NEURAL NETWORK-BASED NOISE SUPPRESSOR & PREDICTOR FOR QUANTIFYING VALVE STICTION IN OSCILLATORY CONTROL LOOPS

A Thesis

by

CARL ASHIE ANNAN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Muhammad N. Karim
Committee Members,	Mahmoud El-Halwagi
	Mahboobul Mannan
Head of Department,	Muhammad N. Karim

December 2014

Major Subject: Chemical Engineering

ABSTRACT

Valve stiction-induced oscillations in chemical processing systems adversely affects control loop performance and can degrade the quality of products. Estimating the degree of stiction in a valve is a crucial step in compensating for the effect.

This work proposes a neural network approach to quantify the degree of stiction in a valve once the phenomenon has been detected. Several degrees of stiction are simulated in a closed loop control system by specifying the magnitude of static (fs) and dynamic (fd) friction in a physical valve model. Each simulation generates controller output OP(t) and process variable PV(t) time series data. A feed-forward neural network (the predictor) is trained to model the relationship between a given OP and PV pattern, and the stiction parameters.

To test the models predictive capability, a separate set of stiction patterns are generated with and without added process noise. An inverse neural network-based nonlinear principal component analysis (INLPCA) noise-suppressor effectively extracts the underlying stiction behaviour from the noise-corrupted OP and PV stiction patterns. In the noiseless test patterns, the predictor is shown to estimate fs and fd with a *0.65%* average error. In the case of the noisy test patterns, the average error achieved was *1.85%.*

Since the predictor is developed offline, the use of computationally intensive real-time search/optimization routines to quantify stiction is avoided. The neural networks proved to be easily implementable, highly flexible models for extracting stiction behavior from control loops and accurately quantifying stiction, as long as an adequate first-principles description of the process dynamics can be developed.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF 4 ! " , %3

1. INTRODUCTION

1.1 Motivation

Chemical processing systems are comprised of many physical components and exhibit highly complex, non-ideal and nonlinear dynamics. A chemical refinery operates around the clock and production is halted if or when physical maintenance is necessary. Any opportunity to make such systems even marginally more autonomous, efficient or perform better in the presence of equipment faults can drastically increase product quality and reduce operation costs. Ideally, these improvements will be achieved through the development of easily-implementable, computationally efficient software applications as physical maintenance is costly and sometimes infeasible.

A very common problem in a chemical plant is the presence of oscillatory behaviour in control loops which can significantly hamper control system performance. There are several causes of sustained oscillations in control loops e.g. aggressive or poor tuning of controllers, external disturbances, valve nonlinearities such as stiction, hysteresis, deadband, saturation and backlash, and the use of linear controllers for the control of processes with highly nonlinear dynamics. External disturbances that cause oscillations in control loops can arise from cyclic events such as fluctuations in raw material quality and ambient temperature[1]. Process nonlinearities such as stiction that cause sustained oscillatory behaviour result from physical defects in valves. These defects are typically caused by seal degradation, lubricant depletion, inclusion of foreign matter or tight packing around the valve stem [2].

Stiction occurs when the controller's demand for the valve to achieve a certain opening or closing position is not met because the stem, a physical component in the valve , is stuck due the presence of static friction[3]. Integral action in the controller or the process causes the control signal to increase in the same direction until the force that moves the stem is great enough to overcome the static friction. The stem position then overshoots the desired opening position , and the controller attempts to compensate for this by sending an aggressive signal that acts in the reverse direction of the original stem movement in an attempt to drive the process variable back to its set point. The valve stem gets stuck again and a control signal is sent of high enough magnitude to overcome static friction, leading to an undershoot of the setpoint. This successive overshooting and undershooting of the set point continues leading to sustained oscillations in the process variable.

A chemical processing system can have hundreds or thousands of valves. It is crucial to detect the presence of and diagnose stiction (distinguish it from from other sources of oscillations), then quantify the degree or extent of it so that appropriate action can be taken to compensate for it. Compensating for stiction, either through physical maintenance or improved controller design, helps maximize the quality of final products.

This work is dedicated to the problem of stiction quantification once the phenomenon has already been detected. Quantification is accomplished through closed loop simulation of stiction in a single-input single-out process by specifying the degree of stiction in a valve stiction model. Two neural network models are used to extract underlying stiction behaviour from noise-corrupted stiction loops as well as estimate the degree of stiction. The first model is a conventional single-hidden layer feedforward network trained to model the relationship between a stiction pattern (a data series) and the degree of stiction (the magnitude of static and dynamic friction). The second model is an inverse neural network that models the underlying relationship between the controller output and process variable for a particular stiction control loop, thereby effectively removing the noise from the OP versus PV stiction plots.

Artificial Neural Networks (ANN's) are composed of a number of neuron-like nodes or processing elements that interact with each other through a set of weighted connections. They adapt themselves to inputs from actual processes by modifying these weights, thereby 'learning' certain relationships which allow for representation of complex systems. ANN's are efficient at processing noisy, incomplete or inconsistent data. These models are used in this study for the purpose of noise filtering and prediction or estimation of the degree of stiction present in a control loop.

1.2 Literature Review: Stiction Quantification

An extensive amount of literature has been produced on stiction detection and diagnosis but significantly less work has been done in the area of estimation and quantification techniques. Several of the stiction quantification literature is based on Hammerstein system identification[4].

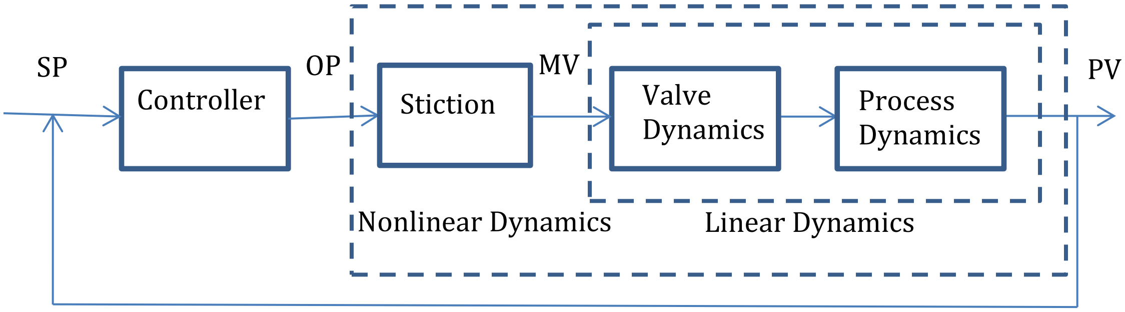


Figure 1.1: Hammerstein System

Figure 1.1 is a schematic of a typical closed-loop feedback process control system where OP is the **controller output**, MV is the **manipulated variable** (the valve position or flow rate) ,PV is the **process** (or controlled) **variable** and SP is the desired set point of PV . In a hammerstein identification-based stiction estimation technique, the MV is usually not explicitly available but OP(t) and PV(t) data are. The goal is essentially to separate the linear dynamics of the process from the static nonlinearity induced by stiction. An empirical data-driven valve model, which relates OP to MV and also contains parameters dictating the degree of stiction, is used to estimate the MV signal based on the known OP. Then a system identification method is used to model the linear dynamics of the process, that is, determine PV(t) from the estimated MV(t). The procedure is repeated until stiction parameters in the valve model are determined such that a certain minimum error criterion between the predicted PV and actual PV(t) is satisfied.

The methods presented by different authors vary in the search or optimization routine and type of process identification used. Lee, Ren and Huang propose a method for stiction estimation using constrained optimization and contour map[5]. The empirical model developed by He et al., parametrized by static and dynamic friction (fs and fd) present in the valve, was selected as the suitable description of the valve stiction nonlinearity[6]. To reduce computational cost, a search space for the stiction model parameters is effectively defined. Process identification is achieved by ordinary least squares method which is based on linear regression. A set of the stiction parameters is selected and MV estimates are generated based on OP(t) data and He's stiction model. Different process models are obtained for each MV data and corresponding PV data in the identification. A multi-start adaptive random search is used to find fs and fd associated with the minimum model error.

Jelali et al use separable least-squares and gradient-free global search algorithms to quantify stiction[7]. An Auto-Regressive Moving Average Exogeneous (ARMAX) output model for process identification and genetic algorithm as a heuristic search of optimal stiction parameters. Karra and Karim proposed a comprehensive approach for distinguishing between stiction and other sources of sustained oscillations in control loops and quantification[1]. In this work, root-cause detection and quantification are accomplished simultaneously. Power Spectral Density (PSD) and Auto Correlation

Function (ACF) are used uncover periodic patterns in OP(t) and PV(t) data from control loops and detect the presence of stiction, while the identification task is performed with an extended ARMAX model, which incorporates a model of both stationary stochastic disturbance and additive non-stationary disturbance affecting the process simulatenously. Srinivasan and Rengaswamy adopt an ARMAX model structure for process identification and a grid search technique[4]. A grid of stiction values *S* and *J* (S=fs+fd, J=fs-fd)is created and each point represents a different estimation of MV(t) and an identified linear process model. The point corresponding to minimum mean-squared error (MSE) is the optimal value of S and J. He and Wang use a linear and nonlinear least squares to quantify stiction based on a semiphysical valve model[8]. The proposed technique uses a curve fitting method to determine fs and fd. Ivan and Lakshminarayanan's approach to stiction quantification is similar to Srinivasan and Rengaswamy's but based on a one parameter stiction model motivated by He[9].

More recently, Zabiri and Omar have applied a neural network based algorithm to the Hammerstein system identification problem. The MV(t) signal is estimated using Choudhury, Thornhill and Shah's empirical data-driven stiction model, a two parameter (S, J) stiction model which is considerably more complex than He's model. An initial guess for S and J is chosen and MV(t) is calculated based on the valve stiction model. Then a NARX (Nonlinear Autoregressive with Exogeneous Input) neural network, which allows for time series forecasting, is used for identification of the process. This type of neural network takes as inputs the MV(t) and PV(t) signal at time steps i to k (that is, a time window of k) and output PV(t) signal at time step k+1. The window is then moved one time step and the next set of input-output data is MV and PV signal at time steps i+1 to k+1, and the PV(t) signal at time step k+2. The NARX network uses several such sets of input-output data to relate past MV(t) and PV(t) time series information to PV(t) signal one time step into the future. The resulting process model is then tested to predict PV(t) based on estimated MV and the root mean-squared error (RMSE) is calculated for the predicted PV(t). In the next iteration the parameter J is fixed and S_2 is chosen such that $S_2 < S_1$. All the steps in the previous iteration are repeated and $RMSE_2$ is calculated. If $RMSE_2 > RMSE_1$, then all values of $S < S_1$ will yield larger errors and they are discarded. The S and J values corresponding to minimum RMSE represents the optimal solution.

The aforementioned techniques for stiction quantification have certain drawback. All involve using real–time search or optimization routines for the estimation of stiction parameters which can be computationally intensive. The genetic algorithm-based search technique used by Jelali is notoriously slow especially if the search space for fs and fd cannot be significantly reduced. All employ either He's or Choudhury's empirical stiction models. Although these models can satisfactorily emulate real stiction behaviour in industrial control loops, they are not as accurate as the physical model, which is less favored in stiction quantification works due to need for numerical integration. Some of these techniques are very sensitive to even low process noise levels or cases where stiction is present along with external disturbances. Zabiri's method admittedly struggles to estimate J values correctly in the presence of external oscillatory disturbances.

1.3 Scope of Study

This work focuses on the problem of stiction quantification using neural network models, assuming oscillations have already been detected in a control loop and diagnosed as being stiction-induced. The aim is to show that neural network models are very flexible models capable to modeling complex nonlinear relations by showing that

1) Simply by using a small section of a stiction pattern, which contains all information about the shape, amplitude and frequency of the oscillations, the degree of stiction can be accurately estimated.
2) When there is process noise or disturbance in addition to stiction oscillations in a control loop, such models can effectively ‘denoise’ such loops and extract the underlying stiction trend.

Two neural network models are used: an inverse neural network to extract the important stiction pattern for cases were both stiction along with external process input and output disturbances (either oscillatory or in the form of colored noise) is present. The second neural network is a fully connected feedforward architecture used to predict the stiction parameters fs and fd given a stiction pattern.

Chapter 2 begins with a detailed description of what causes stiction, a discussion of stiction data from real industrial control loops and models for valve stiction. Empirical data-driven models and a physical model (based on a pneumatic control valve) are compared in terms of how well they capture real stiction behaviour. The chapter concludes with an algorithm for closed loop simulation of stiction in a Single-Input Single-Output (SISO) process using Proportional-Integral (PI) controller and the physical valve model.

In Chapter 3, the theory behind neural networks and their application to predictive modeling and noise filtering is presented. Feed-forward and inverse neural network algorithms are described. Inverse networks are a derivative/extension of auto-associative networks (first proposed by Kramer as a nonlinear principal component analysis technique), which will first be described first.

Chapter 4 contains the comprehensive procedure for quantifying stiction in a closed loop SISO process using OP(t) and PV(t) time series data, based on the physical valve stiction model. Starting with the generation of over one-thousand stiction patterns of varying degrees of stiction (different values of fs and fd), twenty of which will be allocated for testing the accuracy of the stiction quantification approach. Ten of the stiction loops will have no external disturbances and 10 will contain stationary and non-stationary process disturbances. Then a discussion of how the auto-corerelation function and neural network models are used for preprocessing and de-noising of the OP(t) and PV(t) data and ultimately, the prediction of fs and fd for each test pattern.

Finally, the prediction accuracy of the proposed procedure is discussed in Chapter 5. In most cases, control systems will be based on Multiple-Input Multiple-Output (MIMO) type: comprised of multiple manipulated variables and process variables. For such multivariate processes, control loops are not necessarily isolated from one another and may be highly interacting. The validity of treating each loop in a MIMO system as a SISO process for the purpose of stiction quantification is discussed.

Although this work is centered on stiction estimation, neural networks and other data based machine learning algorithms can be applied in stiction detection and diagnosis. A distinguishing characteristic of stiction control loops is the square or parallelogram-like shape of the OP-MV plot. Although in past works, MV operational data is said to be unavailable, this data is becoming more and more readily available in chemical plants. In that case, stiction can be detected and distinguished from other sources of oscillatory phenomena using machine-learning based classification methods such as support vector machines and neural networks. Also, the significance of quantifying stiction accurately in order to properly compensate for the effect will be briefly investigated. MATLAB's Neural Network Toolbox and Dr. Matthias's Scholz's opensource Inverse neural network code is used for the development of the models for stiction quantification.

2. VALVE STICTION

2.1 Introduction

A typical chemical plant has thousands of valves for controlling several key process variables and maintaining them at their desired setpoints. Sustained oscillations caused by valve stiction cause these variables to fluctuate, leading to poor performance of the control system which can potentially impact the variability in product quality and economic profits[10].

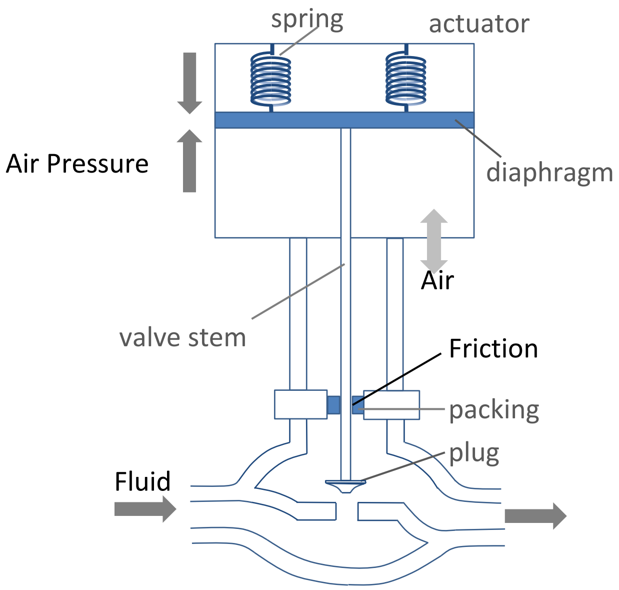


Figure 2.1: Pneumatic Control Valve [2]

Figure 2.1 is a schematic of a typical ‘air-to-open’ pneumatic control valve. To open the valve, a control signal is sent to the valve actuator demanding a certain amount of air pressure to be applied to the diaphragm. The amount of force acting on the diaphragm must be enough to compress the spring in order to get the valve stem and attached plug to move upward, allowing the flow of fluid through the valve. To close the valve, enough air pressure is released so that the plug falls into seat and obstructs fluid flow.

In a valve impaired by stiction (or 'static friction') the stem does not respond to a demand by the control signal to achieve a particular opening position because of frictional resistance, typically between the stem and packing.

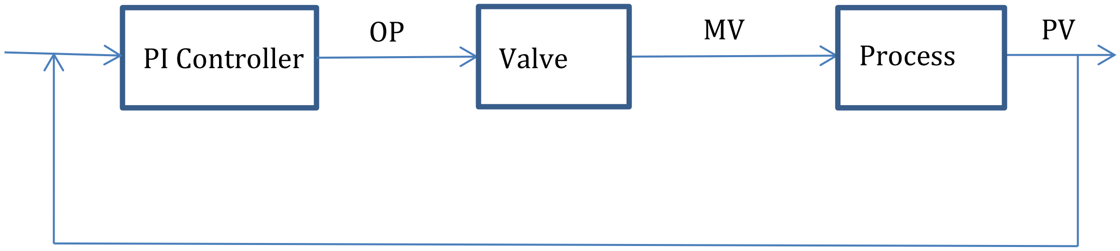


Figure 2.2: Closed-loop Feedback Single-Input Single-Output Control System

For a closed loop feedback control system such as that shown in Figure 2.2, with a conventional proportional integral controller, the following equation governs the relationship between deviations of PV(t) from the setpoint and MV(t):

$$MV(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau$$

To achieve a certain valve opening position, the controller output OP signals the valve actuator to apply adequate force(air pressure) to effect valve stem movement. If stiction is present, the stem does not move. Integral action in the controller causes the control signal to increase in the same direction until the actuator applies a force of magnitude greater than the static friction. Once static friction is overcome, the valve slips and overshoots the desired opening position. The integral part of the controller, which causes control action to be taken according to 1) how long and 2)by how much PV(t) has deviated from the set point, will cause aggressive control action in order to compensate for the overshoot and return the PV(t) back to setpoint. However if stiction is still present the same scenario happens in the reverse direction, leading to an undershoot. These successive overshoots and undershoots lead to continuous oscillations in the control loop.

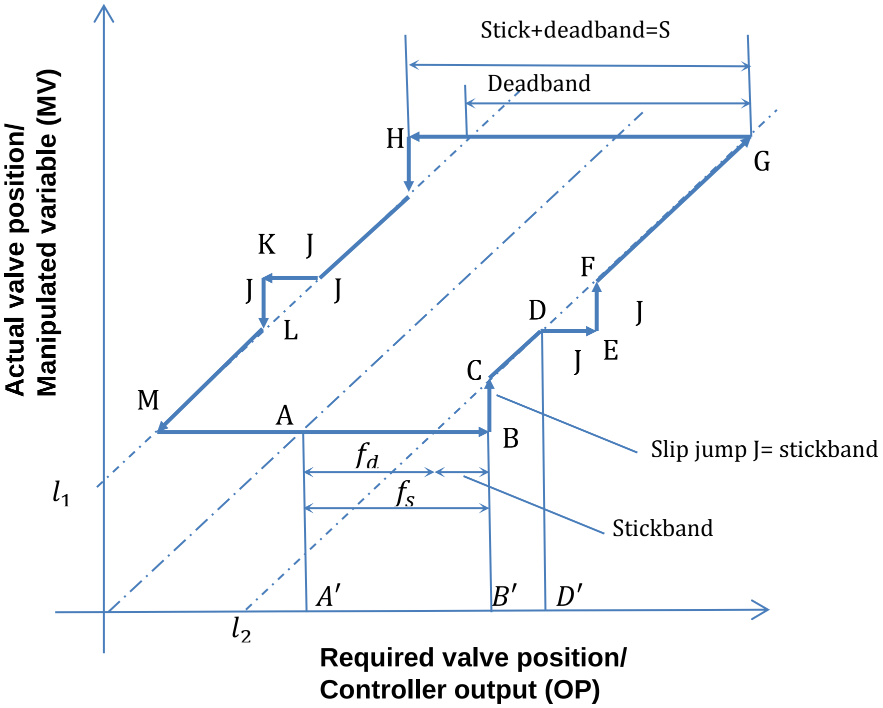


Figure 2.3: Relationship between controller output (OP) and valve position (MV) under stiction [7]

Figure 2.3 describes the relationship between OP and MV in a ‘sticky’ valve. Ideally, MV will vary linearly *and proportionally* with adjustments to the controller output OP according to the line between $l_1$and l_2. For illustration purposes the bolded lines represent the non-ideal behaviour of a valve with stiction. Let A indicate the resting position of the valve. Here, OP,MV, f_s and f_d are all expressed as a percentage of the total valve travel range. To open the valve, the OP signal is increased. Due to stiction present, the valve stem does not move (MV remains constant) until the controller demands an amount of force greater than the static friction f_s (represented by $A'B'$). At some value of OP (D') the f_s is overcome and the valve stem suddenly accelerates and jumps to a new opening position C where its movement varies linearly with OP but offset by an amount f_d, since it is now working against dynamic friction (resistance when the valve is moving). It is possible for the valve to get stuck again at D. In that case, it will have to overcome frictional resistance in the amount of J=$f_s - f_d$. The same trajectory is followed in the reverse direction when the valve is closing. Typically, the valve stem gets stuck once in each direction leading to a square-like or parallelogram-like shape of the MV-OP plot that, which is a way of distinguishing stiction oscillations from other types of oscillations, that is, if MV(t) data is available.

2.2 Industrial Control Loops with Stiction-Induced Oscillations

In general, stiction-induced oscillations in process control loops have characteristic shapes that help distinguish them from other sources of oscillatory behaviour such as aggressive controller tuning and external disturbances. Several authors have collected real stiction data from various industrial systems such as chemical refining, mining and metal processing, mineral processing, pulp and paper, and power plants. These have been gathered into a database of over 100 sets of OP(t) and PV(t) stiction data.

Stiction oscillations typically have sinusoidal, triangular, saw-tooth or square-like shapes. *Figure 2.4* shows some OP(t) and PV(t) patterns from pressure, temperature, flow and level control loops with stiction. MV(t) data is unavailable in these stiction loops. However, if this data is available, detecting stiction and distinguishing it from other types of oscillations becomes possible. As mentioned earlier, this is because a distinguishing characteristic of stiction phenomena is that the OP-MV plot will adopt a parallelogram-like shape due to stick-slip behaviour of the valve in the opening and closing directions.

Generally, stiction loops will have the following distinguishing features: OP(t) and PV(t):

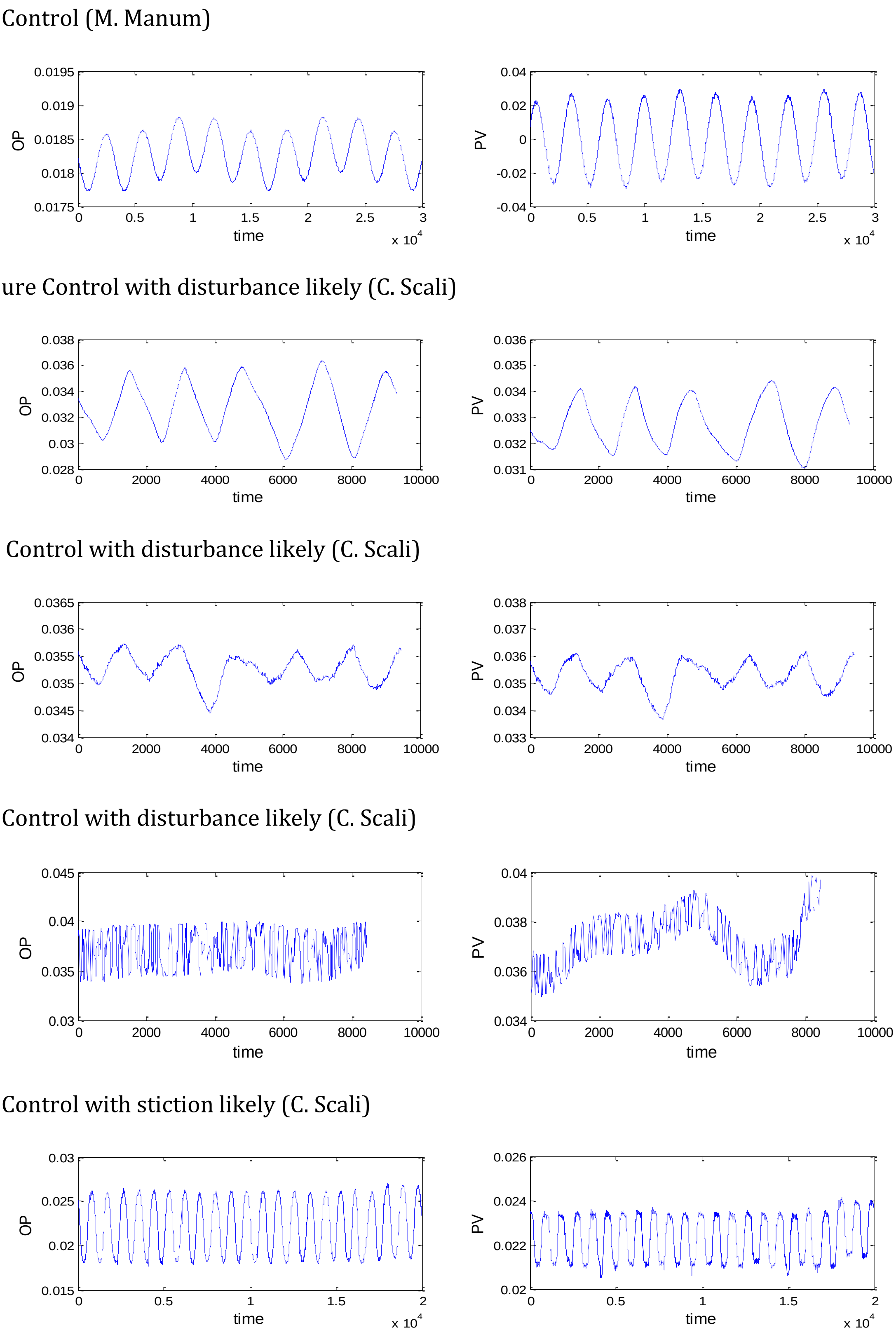
- Oscillations due to stiction will contain harmonics. The harmonic of a periodic signal is a component frequency of the signal which is an integer multiple of the fundamental frequency. Aggressive controller tuning and external disturbances usually lead to sinusoidal waves, which contain only one frequency. Stiction patterns will be rectangular, saw-tooth(asymmetric triangular) or triangular[11]. A rectangular wave leads to odd harmonics, that is, it contains its fundamental frequency f and 3f, 5f, 7f, etc.; while triangular or saw-tooth signals contain both even and odd harmonics[11].

- For self-regulating processes OP(t) oscillations are typically triangular and for integrating processes PV(t) follows triangular wave form [2]. If the MV(t) signal (which describes the movement of the valve stem with time) is explicitly available, it will exhibit rectangular or square-like oscillations due to the stick slip behaviour of the valve stem. In flow control processes ,where the MV(t) signal varies proportionally to PV(t) (in this case, the flow rate), PV(t) will follow a rectangular wave if stiction behaviour is nearly 'ideal'.
- When the magnitude of stiction is higher, the amplitude of the oscillations in PV(t) and OP(t) is higher and the peaks are sharper and pronounced and when process lag is high, the peaks are more blunt or curved and the patterns is close to sinusoidal[11].

These are only meant to be general guidelines based on extensive studies performed in the literature. Of course, process dynamics and controller parameters will affect the shapes of stiction-induced oscillations. Table 2.1 is a summary of typical shapes of stiction patterns for different process types and control types.

Process type	Fast processes (flow)		Slow processes	Integrating processes	Level with PI control
	Dominant I action	Dominant P action	Pressure & temaperature	Level	
OP	Triangular (Sharp)	Rectangular	Triangular (Smooth)	Triangular (Sharp)	Triangular (Sharp)
PV	Square	Rectangular	Sinusoidal	Triangular (Sharp)	Parabolic

Table 2.1:Typical stiction pattern shapes for different process types [12, 13]



Flow Control (M. Manum)

Pressure Control with disturbance likely (C. Scali)

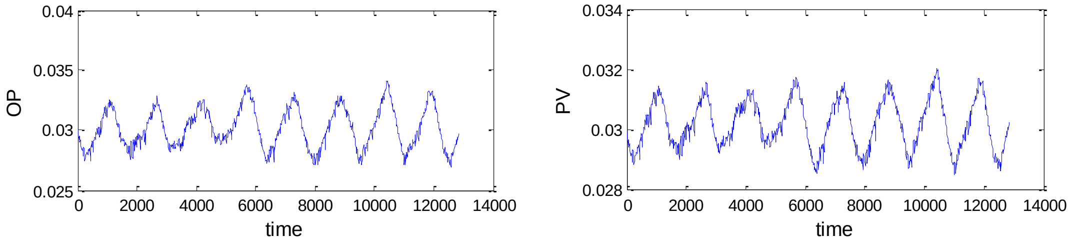
Level Control with disturbance likely (C. Scali)

Flow Control with disturbance likely (C. Scali)

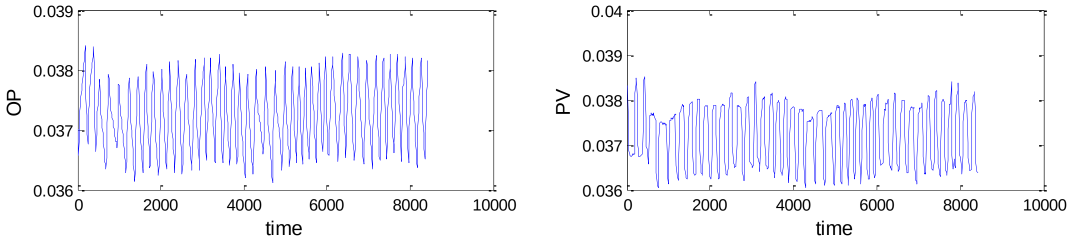
Flow Control with stiction likely (C. Scali)

Figure 2.4: Industrial Control loops with stiction

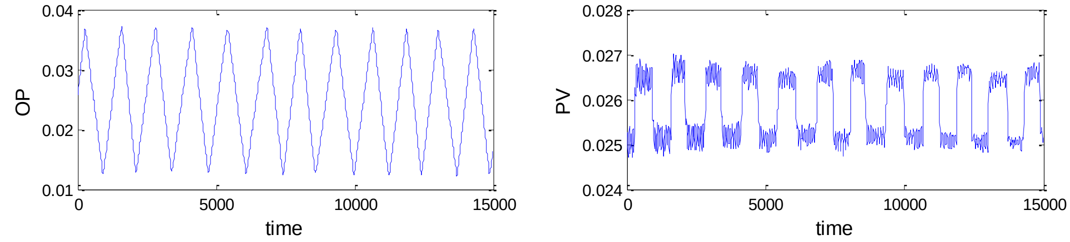
Level Control with stiction likely (C. Scali)



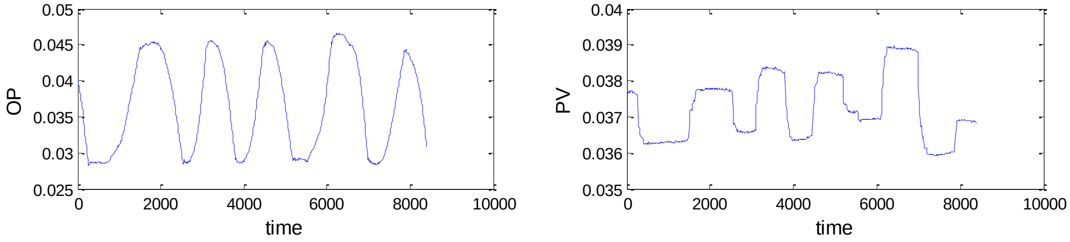
Pressure Control with possibility of marginal stability



Flow Control with stiction likely (C. Scali)



Flow Control with stiction likely (C. Scali)



Flow Control with stiction likely (B. Huang)

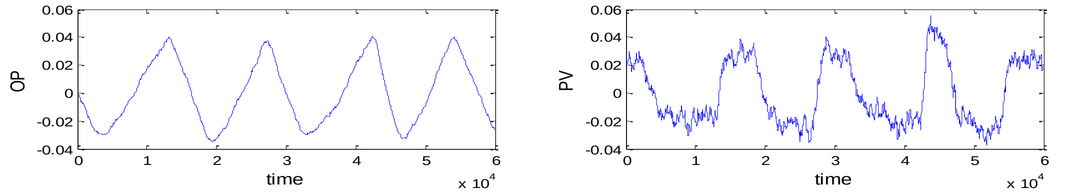
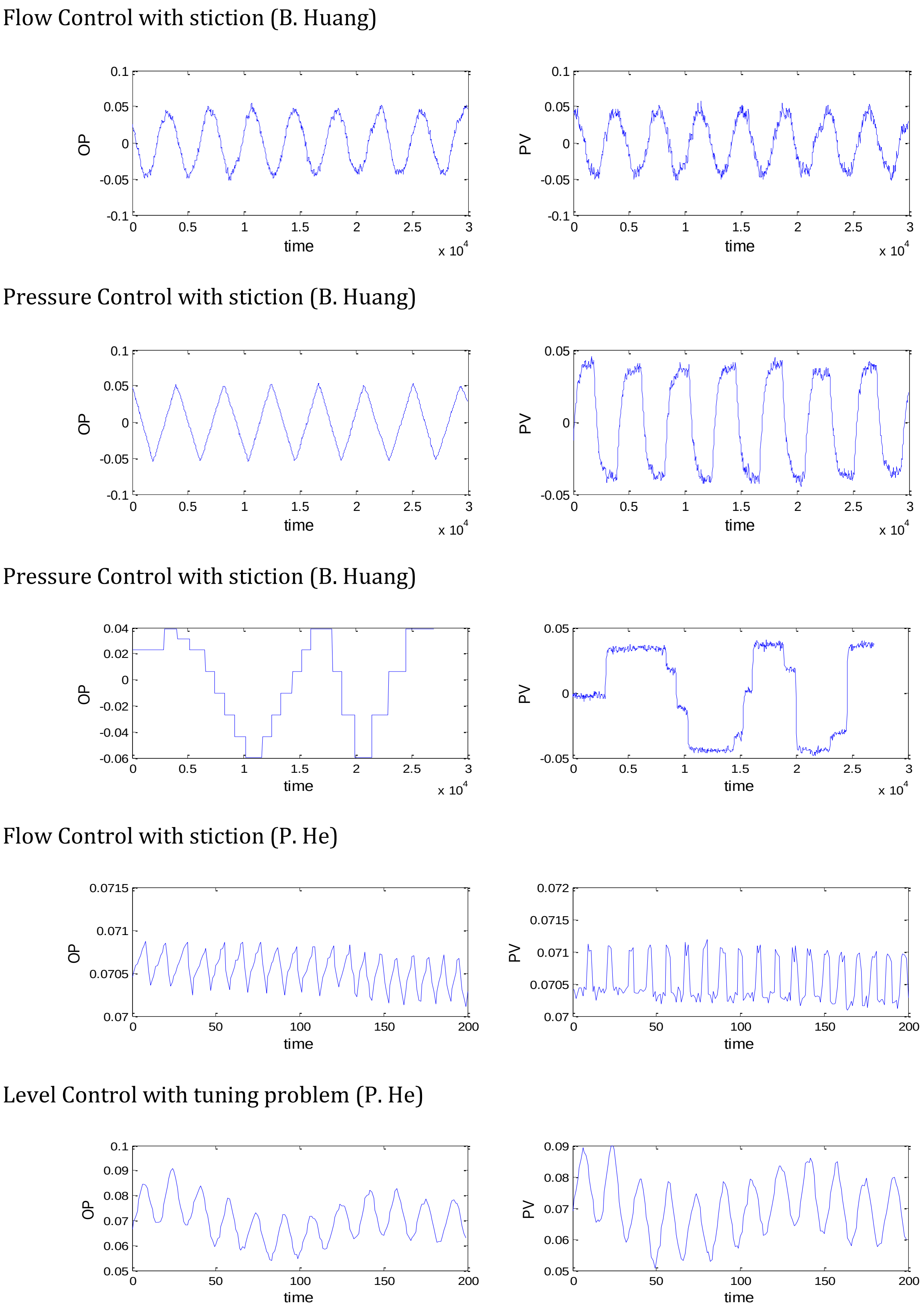


Figure 2.4 Continued



Flow Control with stiction (B. Huang)

Pressure Control with stiction (B. Huang)

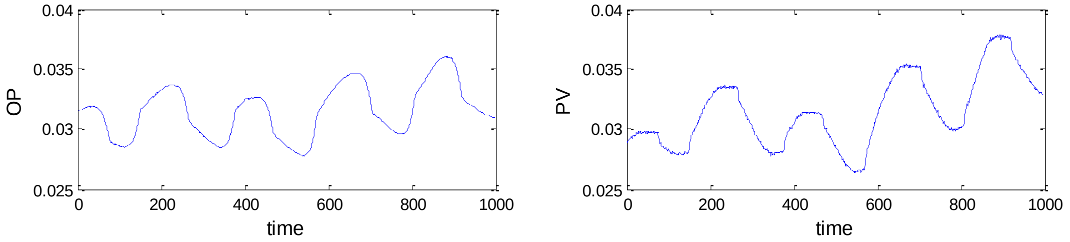
Pressure Control with stiction (B. Huang)

Flow Control with stiction (P. He)

Level Control with tuning problem (P. He)

Figure 2.4 Continued

Flow Control with stiction (A. Horch)



Flow Control with stiction (A. Horch)

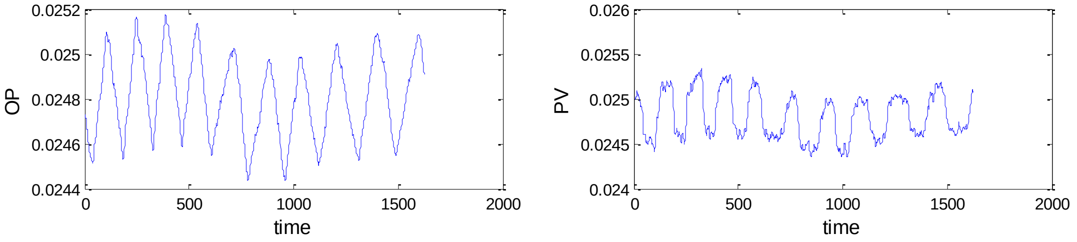
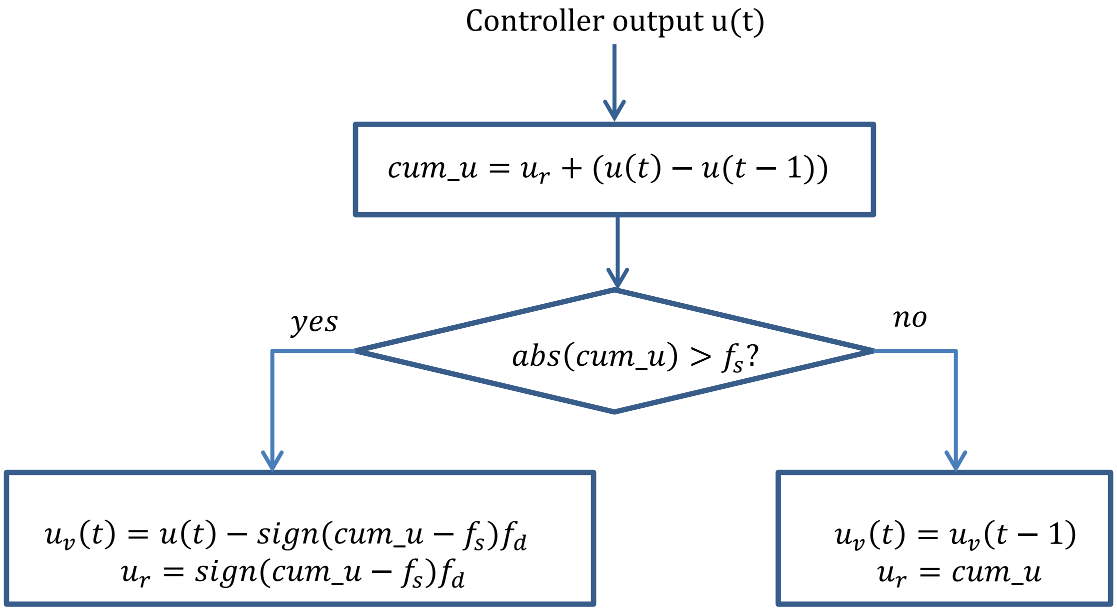


Figure 2.4 Continued

2.3 Modeling Valve Stiction

Empirical or data-driven valve stiction models have been developed [3, 6, 14]. He's two and three parameter model employs the most simplistic logic of the valves movement under stiction and has been shown to emulate real stiction behaviour in industrial control loops very well compared to the others[2]. Figure 2.5 shows the algorithm for He's two-parameter model



Controller output u(t)
$cum\_u = u_r + (u(t) - u(t-1))$
yes
$abs(cum\_u) > f_s?$
no
$u_v(t) = u(t) - sign(cum\_u - f_s)f_d$
$u_r = sign(cum\_u - f_s)f_d$
$u_v(t) = u_v(t-1)$
$u_r = cum\_u$

Figure 2.5: He's two-parameter stiction model [15].

Here, all variables have been translated to be in terms of percent valve travel range. OP(t), MV(t) are now expressed as $u(t)$ and $u_v(t)$. When stiction is present, the residual amount of force being applied that has not effected valve stem movement is u_r. The cumulative force cum_u acting on the valve in the current control instance is the sum of u_r and the change in controller output signal from previous to current control instance $u(t) - u(t-1)$. If cum_u is of high enough magnitude to overcome the static friction f_s, then the valve moves to position $u_v(t)$. The direction of excess force applied above what was required to overcome the stiction band is given by $sign(cum\_u - f_s)$ and the dynamic or moving friction that resists valve movement is f_d. So $sign(cum\_u - f_s)f_d$ represents how much valve travel will not be achieved as a result of the dynamic friction f_d that the valve is working against in the direction $sign(cum\_u - f_s)$. If the valve stem does not overcome f_s, then it does not move, that is, $u_v(t) = u_v(t-1)$ and $u_r = cum\_u$.

The data-driven models are typically validated against physics-based models, which are the most descriptive models of the physics of valves with stiction but are often avoided due to the computational cost incurred in solving them. However the purpose of this work is to generate stiction patterns offline (i.e. when the chemical processing system is not in operation) and use these stiction patterns to develop a predictor (also offline) capable of quantifying stiction in real time when the process is in operation.

Here, the motion (position and velocity) of the valve stem of the pneumatic control valve shown in Figure 2.1 is governed by newtons law of motion Force = mass × acceleration. This model is obtained from [15].

$$M\frac{d^2x}{dt^2} = \sum \text{Forces} = F_a + F_r + F_f + F_p + F_i$$

x: relative valve stem position

$F_a = Au$
F_a: force applied by pneumatic actuator

A : area of the diaphragm
u : actuator air pressure or the valve input signal

$F_r = -kx$
F_r: spring force
k : spring constant

$F_p = -A_p \Delta p$

F_p : force due to fluid pressure
A_p: plug unbalance area
Δp: fluid pressure drop across the valve

F_i : extra force required to force the valve into seat

F_f: friction force (includes static and dynamic/moving friction)

F_p & F_i are assumed negligible

F_d: Dynamic friction (velocity independent term)
vF_v: viscous friction term that depends linearly on the velocity

$$F_f = \begin{cases} -\mathrm{F_d sign(v)} - \mathrm{vF_v} - (\mathrm{F_s} - \mathrm{F_d})e^{-\left(\frac{v}{v_s}\right)^2} \mathrm{sign(v)}, & \text{if } \mathrm{v} \neq 0 \\ -(\mathrm{F_a} + \mathrm{F_r}), & \text{if } \mathrm{v} = 0 \text{ and } |\mathrm{F_a} + \mathrm{F_r}| \leq \mathrm{F_s} \\ -\mathrm{F_s sign}(\mathrm{F_a} + \mathrm{F_r}), & \text{if } \mathrm{v} = 0 \text{ and } |\mathrm{F_a} + \mathrm{F_r}| > \mathrm{F_s} \end{cases}$$

$\mathrm{F_s}$: maximum static friction
$\mathrm{v_s}$: empirical stribeck velocity parameter

The friction term is represented as a piecewise function, each describing the amount of friction present when the valve is stuck, about to become unstuck, or moving. The first line is the condition when the valve is moving. In this case, the valve acts against dynamic friction, which acts in the opposite direction of the valves intended movement -sign(v). The second expression is the viscous friction accounts for resistance to the valve stems motion due to properties of the flowing fluid and is proportional to the velocity of the valve. Finally, the last the stribeck term $(\mathrm{F_s} - \mathrm{F_d})\exp[-(v/v_\mathrm{s})^2]\mathrm{sign(v)}$ addresses the discontinuity at the stick-slip moment , where the valve goes from acting against friction of amount $\mathrm{F_s}$ to $\mathrm{F_d}$, when the valve stem just begins to move. An empirical velocity parameter v_s controls the rate at which that point of discontinuity is approached. The second line indicates the condition where the valve is stuck, in which case the force $|\mathrm{F_a} + \mathrm{F_r}|$ acting to move the valve is less than $\mathrm{F_s}$. At the instance where the stem breaks free of the stiction, that is, $|\mathrm{F_a} + \mathrm{F_r}| > \mathrm{F_s}$, the valve stem is resisted by a force $\mathrm{F_s sign(F_a + F_r)}$. The negative signs on all terms indicate that the friction acts in opposition to the direction in which the valve moves.

The equation describing the motion of the valve stem can be rewritten as follows:

$$\dot{x} = v$$

$$m\dot{v} = \mathrm{F_a} + \mathrm{F_r} + \mathrm{F_f} + \mathrm{F_p} + \mathrm{F_i}$$

This stiff system of ordinary differential equations can be solved by numerical integration. However, But if used directly, difficulties in numerical integration exist due to hash discontinuity caused by sign function at zero velocity (Detection and Diagnosis of stiction in control loops).

So $\mathrm{F_f}$ is approximated using piecewise function, where $|v| < \delta$ is used to approximate $v = 0$.

Therefore, the ODEs we use to simulate the sticky valve are the following, with $\delta = 1 \times 10^{-6} in/s$:

$$F_f = \begin{cases} -\mathrm{F_c sign(v)} - \mathrm{vF_v} - (\mathrm{F_s} - \mathrm{F_c})e^{-\left(\frac{v}{v_\mathrm{s}}\right)^2}\mathrm{sign(v)}, & \text{if } \mathrm{v} \neq 0 \\ -(\mathrm{F_a} + \mathrm{F_r}), & \text{if } \mathrm{v} = 0 \text{ and } |\mathrm{F_a} + \mathrm{F_r}| \leq \mathrm{F_s} \\ -\mathrm{F_s sign(F_a + F_r)}, & \text{if } \mathrm{v} = 0 \text{ and } |\mathrm{F_a} + \mathrm{F_r}| > \mathrm{F_s} \end{cases}$$

$$\dot{x} = v$$

$$m\dot{v} = \begin{cases} \mathrm{S_a}u - kx - \mathrm{F_c} - v\mathrm{F_v} - (\mathrm{F_s} - \mathrm{F_c})e^{-\left(\frac{v}{v_\mathrm{s}}\right)^2}, & \text{if } \mathrm{v} > \delta \\ \mathrm{S_a}u - kx - \mathrm{F_c}, & \text{if } -\delta \le \mathrm{v} \le \delta \text{ and } (\mathrm{S_a}u - kx) > \mathrm{F_s} \\ 0, & \text{if} - \delta \le \mathrm{v} \le \delta \text{ and} - \mathrm{F_s} \le (\mathrm{S_a}u - kx) \le \mathrm{F_s} \\ \mathrm{S_a}u - kx - \mathrm{F_c}, & \text{if} - \delta \le \mathrm{v} \le \delta \text{ and } (\mathrm{S_a}u - kx) < -\mathrm{F_s} \\ \mathrm{S_a}u - kx - \mathrm{F_c} - v\mathrm{F_v} - (\mathrm{F_s} - \mathrm{F_c})e^{-\left(\frac{v}{v_\mathrm{s}}\right)^2}, & \text{if } \mathrm{v} < -\delta \end{cases}$$

$$J = \begin{bmatrix} 0 & 1 \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial v} \end{bmatrix}$$

$$\frac{\partial f_2}{\partial x} = \begin{cases} 0, & if\ -\delta \le \mathrm{v} \le \delta \text{ and } -\mathrm{F_s} \le (\mathrm{S_a}u - kx) \le \mathrm{F_s} \\ -\frac{k}{m}, & else \end{cases}$$

$$\frac{\partial f_2}{\partial v} = \begin{cases} \frac{1}{m}\left[-\mathrm{F_v} - \frac{2v}{{v_\mathrm{s}}^2}(\mathrm{F_s} - \mathrm{F_c})e^{-\left(\frac{v}{v_\mathrm{s}}\right)^2}\right], & if\ -\delta \le \mathrm{v} \le \delta \text{ and } -\mathrm{F_s} \le (\mathrm{S_a}u - kx) \le \mathrm{F_s} \\ 0, & if\ -\delta \le \mathrm{v} \le \delta \\ \frac{1}{m}\left[-\mathrm{F_v} + \frac{2v}{{v_\mathrm{s}}^2}(\mathrm{F_s} - \mathrm{F_c})e^{-\left(\frac{v}{v_\mathrm{s}}\right)^2}\right], & if\ v < -\delta \end{cases}$$

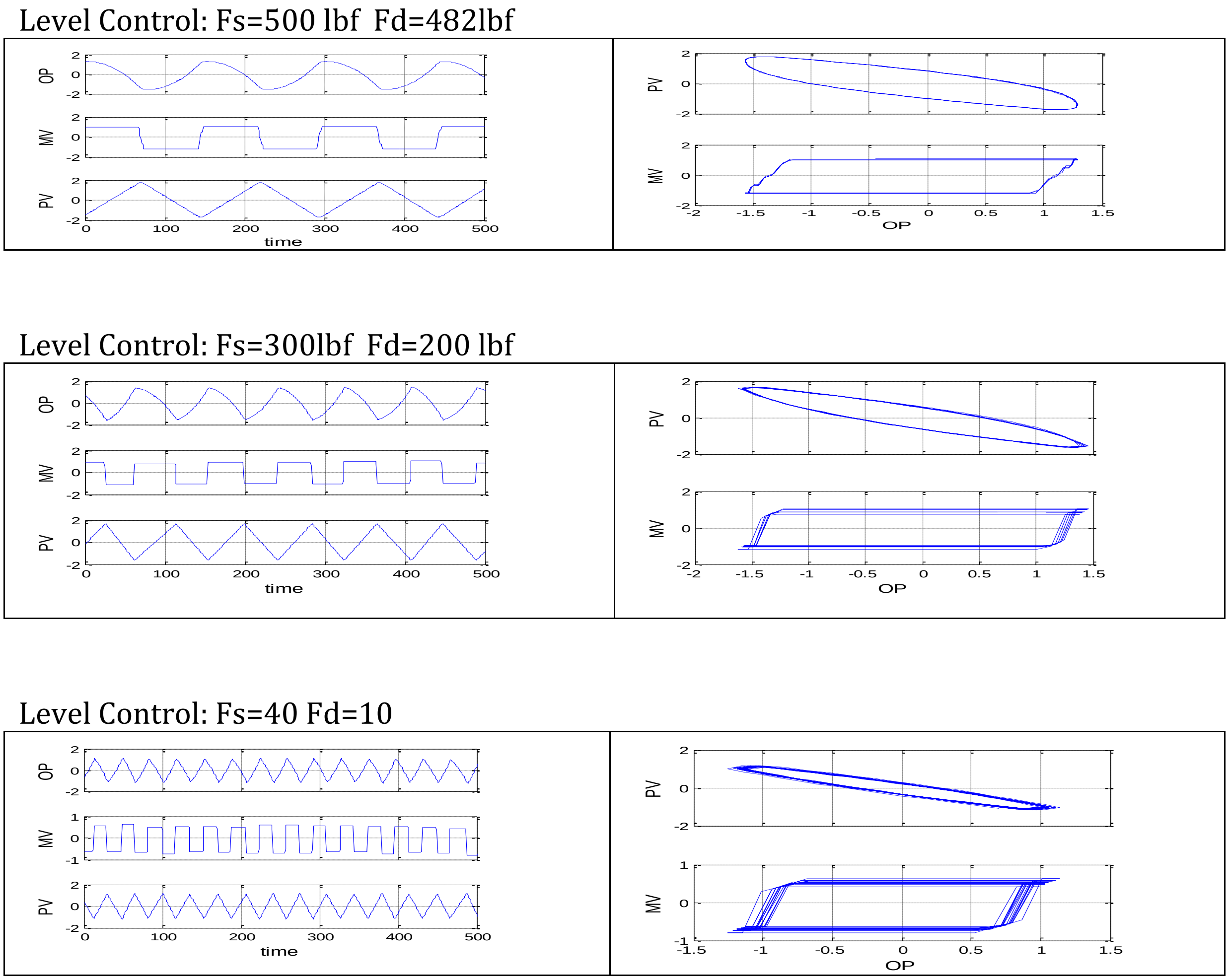
In some cases, especially for stiff systems, ODE solver performance can be enhanced or accelerated by including analytically computed jacobian matrix specially coding your ODE file. Software such as MatLab contain built in functions that compute the jacobian numerically, and in this work, the analytically computed jacobian did not lead to any noticeable speed up.

2.4 Simulating Stiction in Single-Input Single-Output (SISO) Closed Loop Processes

The following are some valve stiction simulations based on this physical model and the processes described by the transfer functions in Table 2.2.

	Level	**Concentration**
Process	$\frac{1}{15s}$	$\frac{3}{10s+1}$
PI Controller	$G_c = 3\left(1+\frac{1}{30s}\right)$	$G_c = 0.2\left(1+\frac{1}{2s}\right)$

Table 2.2 Simulating stiction in linear time invariant processes

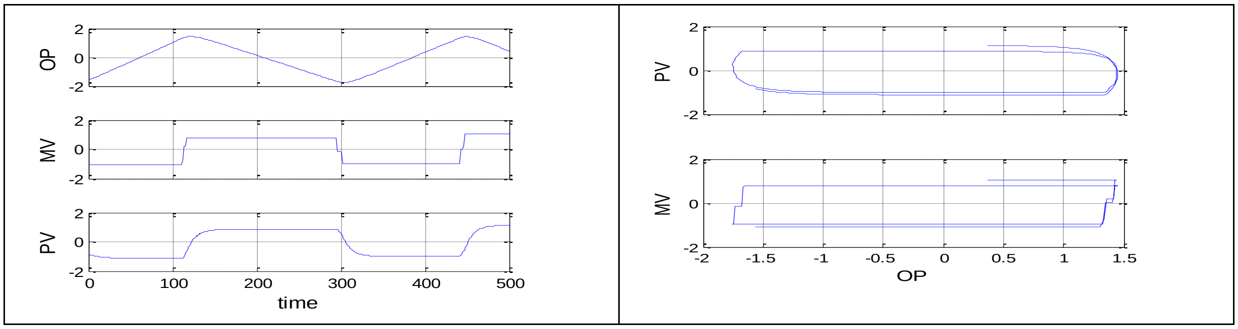

Level Control: Fs=500 lbf Fd=482lbf

Level Control: Fs=300lbf Fd=200 lbf

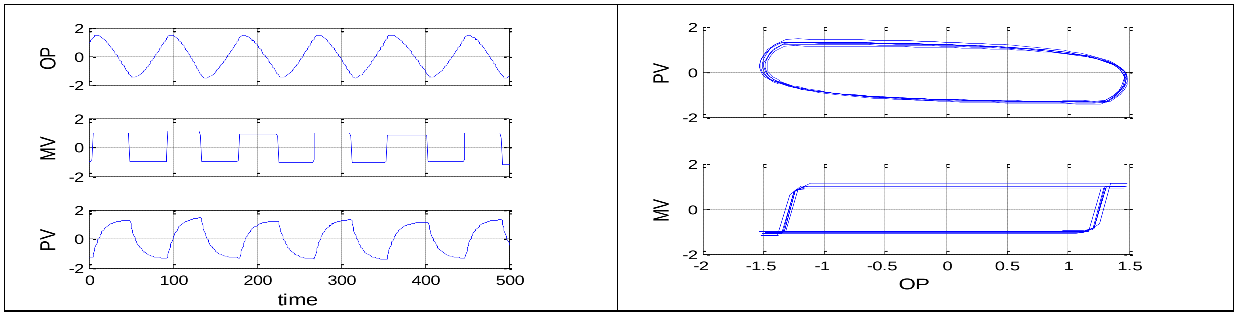
Level Control: Fs=40 Fd=10

Figure 2.6: Simulating stiction in linear time invariant processes

Concentration Control: Fs=500 Fd=482



Concentration Control: Fs=300 Fd=200



Concentration Control: Fs=40 Fd=10

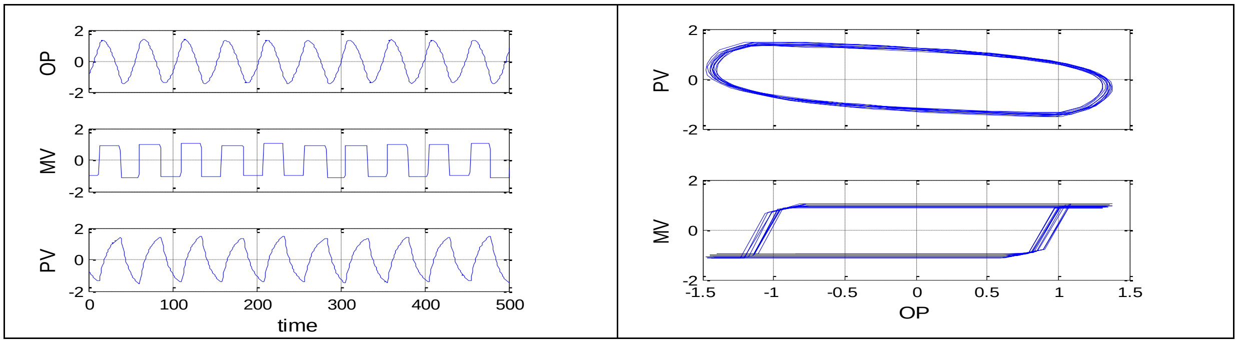


Figure 2.6 Continued

3. DATA-BASED MODELING USING NEURAL NETWORKS

3.1 Introduction

It is difficult to develop first principles mathematical models that describe the dynamics of complex systems with high non-linearity, uncertainty and noise. However, if sufficient data is available from such systems, models can be developed based on this data using powerful data-based modeling algorithms from the field of machine learning. Artificial neural networks (ANN's) are a class of machine learning algorithms containing nonlinear mapping functions that together can approximate any function or 'fit' any data to arbitrary accuracy [14]. ANNs which are imported from statistical learning theory [16] , can perform a nonlinear mapping of input-output data, learn relationships and produce a certain desired output given new inputs.

$$u_k = \sum_{j=1}^{n} x_j\, w_{kj} \qquad v_k = u_k + b_k$$

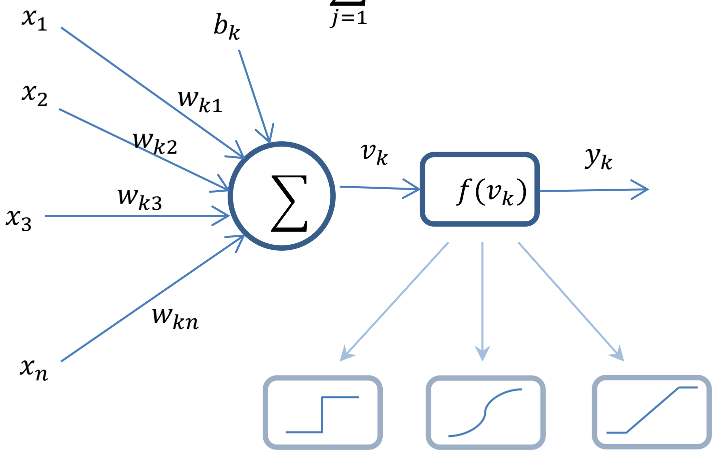


Figure 3.1 Perceptron Model

ANN's are composed of computational nodes (or 'neurons') that interact with each other via weighted, adjustable interconnections. Figure3.1 shows a perceptron model, the basic building block of a neural network. For illustration purposes. Say one has a set X of N vectors

$X = \hat{x}_i \quad i = 1, \ldots\ldots, N$

and each vector $\hat{x}_i$ is of length n, as shown in the model. Suppose each vector has an associated value y_i.This perceptron model can 'learn' or model the relationship between a given $\hat{x}_i$ and its associated y_i as long as the $\hat{x}_i{}'s$ are related with one another in some way. For instance, each $\hat{x}_i$ is data collected from a specific type of experiment where numbers were to be recorded at equally spaced time intervals (i.e. time series data was obtained) and each $\hat{x}_i$ represents a different trial of this experiment run under different conditions. For a particular $\hat{x}_i$, its elements are weighted and summed up, then passed through some linear or nonlinear mapping function $f(u_k)$ to obtain an estimate of y_k, which will be denoted y'_k. The subscript k is simply a 'label' for the node, and it will be clear why it is needed when the perceptron model is extended to a neural network model which has multiple nodes. The weights w_{kj} are initially randomly generated. Then the error between $y'_k$and the actual $y_k$will be determined. This procedure, known as forward propagation, will be repeated for all the vectors using the same randomly generated weights so that N y'_k values are obtained. The total error will be propagated back into the network to determine how the parameters w_{kn} should be adjusted such that the model produces a better prediction of y'_k values in the subsequent iteration. The goal is to execute enough iterations so that the total error satisfies some minimum criterion. So the final model is defined by the weight parameters, which essentially determine the contribution of a particular element in the vector in describing the relationship between the input vectors and the outputs.

So given a new $\hat{x}_i$ which the network has not previously seen, the network predicts y'_k. The mapping function is typically a linear, sigmoid or hyperbolic tangent function. The sigmoid function is given by

$$f(\,v_k\,) = \frac{1}{1+\exp(-av_k)} = y_k \qquad\qquad v_k = u_k + b_k$$

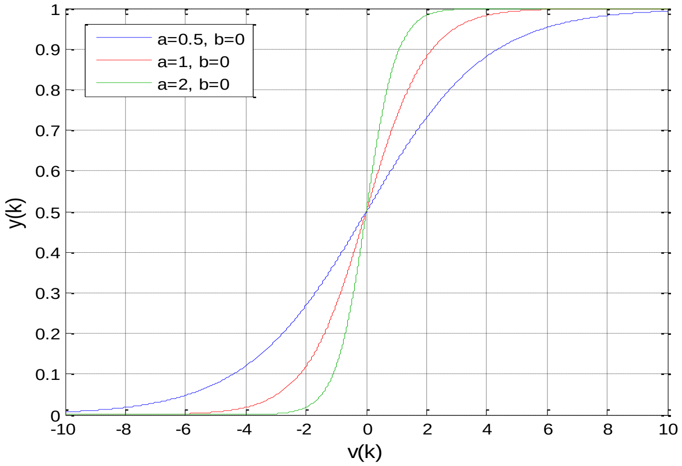


Figure 3.2 The sigmoid function

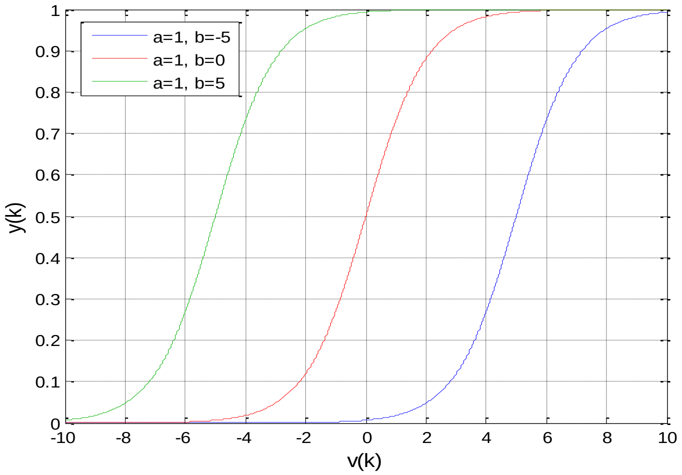


Figure 3.3 Effect of bias on the sigmoid function

Modifying the weights therefore affects the shape and curvature of the sigmoid transfer function whereas the bias neurons adds an additional degree of freedom that allows the curve to shift left or right. Without the sigmoid function, the ANN can only model functions which are linear combinations of the inputs. The function varies from 0 to 1, which is an important property in neural networks used for binary classification problems where the inputs vectors are to be categorized into one of two classes. Thus if the network yields a prediction closer to one, the input vector is classified as being in one category and if closer to 0, the vector belongs to the other category. Another important property of the sigmoid function that makes it useful in regression tasks is that for very high or low values of the input v(k) , that is, near the limiting values 0 and 1 of the function, the rate of change of the slope of the curve is very small. Therefore large changes in the input near these limits do not cause the output to 'blow up'. This is a very useful property in the back-propagation learning algorithm, which is based on gradient descent optimization. This prevents large changes in the calculated error and consequently, modest changes in the weights when those limits are approached.

3.2 Feed-forward Neural Network Model

Here the perceptron model, which can only be used for simple modeling to multilayer, is extended to a multimode network which are meant for complex nonlinear modeling. The feed forward network shown in the Figure 3.4 contains three layers: the input layer which receives the input vector, the hidden layer which performs nonlinear mapping of the inputs to the outputs, and an output layer. Here, the blue dot on the left of a particular node implies the elements entering a node are weighted according to their associated connection and summed before entering the node. It is also implied that there are biases to each node.

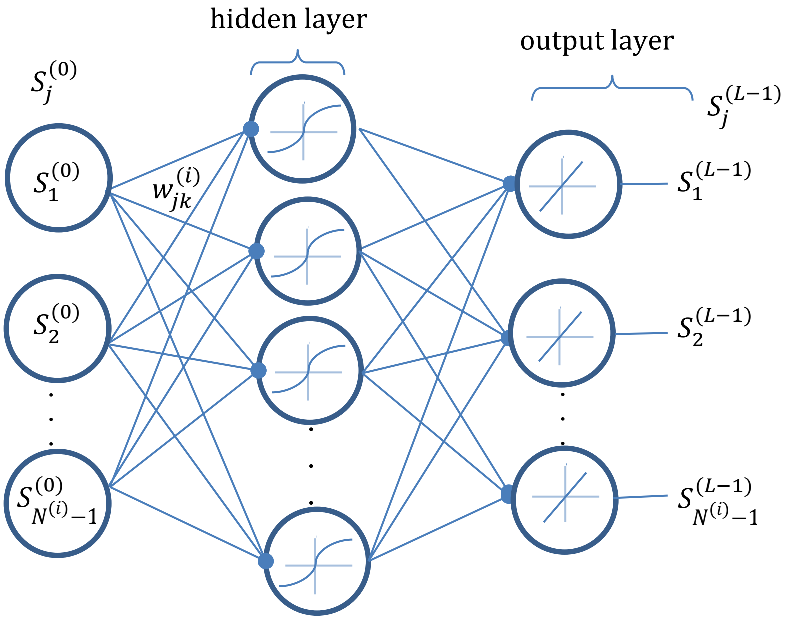


Figure 3.4 Feed-forward neural network

Throughout this presentation, the compact notation of Kirby and Miranda for mathematical theory behind neural networks is adopted [17]. The goal here is to model the relationship of several sets of input vectors of length n and their corresponding output vectors of length m, such that given new input vectors previously unseen by the network, their output vectors are predicted accurately. The first step in neural network learning is **forward propagation.**

The weighting parameters of the network are randomly generated and for each input vector, an output vector is calculated according to the following expressions. There are L layers in the neural network , the first layer being layer 0 and the last layer, L-1. Each layer contains N^inodes numbered j=0 to j=$N^{(i)} - 1$. So a node j in layer i which will be identified as $N_j^{(i)}$. The state value of each node is $S_j^{(i)}$ and $P_j^{(i)}$ denotes the value of the input to a node or pre-state value. Each node $N_j^{(i)}$ has an associated weight $w_{kj}^{(i-1)}$ that connects node $N_k^{(i-1)}$ from the previous layer to it (k is the index for nodes in the previous layer). Thus the inputs to a particular sigmoidal node $N_j^{(i)}$ $(i \geq 1)$ its pre-state value is

$$P_j^{(i)} = \sum_{k=0}^{N^{i-1}-1} w_{kj}^{(i-1)} S_k^{(i-1)} + b_j^{(i)} \qquad S_j^{(i)} = \sigma_j^{(i)}\left(P_j^{(i)}\right) = \frac{1}{1 + \exp(-a * P_j^{(i)})} \qquad S_j^{(i)}$$

The sum squared error between calculated outputs $S_k^{(L-1)}$ and actual outputs G_j is given by

$$E = \frac{1}{2} \sum_{j=0}^{N^{L-1}-1} \left(S_k^{(L-1)} - G_j\right)^2$$

Once forward propagation is complete and the error is calculated, the back-propagation algorithm is implemented. The gradient of the error with respect to all the weights $w_{jk}^{(i)}$ in the network are calculated according to the following expressions:

$$\frac{\partial E}{\partial S_j^{(L-1)}} = S_j^{(L-1)} - G_j$$

$$\frac{\partial E}{\partial b_j^{(i)}} = \frac{\partial E}{\partial S_j^{(i)}} \frac{\partial S_j^{(i)}}{\partial b_j^{(i)}}$$

$$\frac{\partial E}{\partial w_{jk}^{(i)}} = \frac{\partial E}{\partial S_j^{(i+1)}} \frac{\partial S_j^{(i+1)}}{\partial w_{jk}^{(i)}} = \frac{\partial E}{\partial S_j^{(i+1)}} \frac{\partial \left[\sigma_j^{(i)}\left(P_j^{(i)}\right)\right]}{\partial w_{jk}^{(i)}}$$

$$\frac{\partial E}{\partial S_j^{(i-1)}} = \sum_{j=0}^{N^{(i)}-1} \frac{\partial E}{\partial S_j^{(i)}} \frac{\partial S_j^{(i)}}{\partial S_k^{(i-1)}}$$

$$\frac{\partial S_j^{(i)}}{\partial S_k^{(i-1)}} = \frac{\partial S_j^{(i)}}{\partial b_j^{i}} \frac{\partial b_j^{i}}{\partial S_k^{(i-1)}}$$

Adjustments to the weights are given by the following expression

$$\Delta w_{jk}^{(i)} = -\mu \frac{\partial E}{\partial w_{jk}^{(i)}}$$

Where μ is the learning rate parameter and controls how fast convergence to the optimal solution is reached. Too large a learning rate will lead to large adjustments in the weight but may lead to oscillations around the optimum and ultimately, divergence. To illustrate this point, consider the one-dimensional optimization problem where one adaptable parameter w is used to determine the minimum of some convex error function E(w) in Figure 3.5. Say, the initial guess for the weight w and the associated error E is indicated by the red point on the graph.

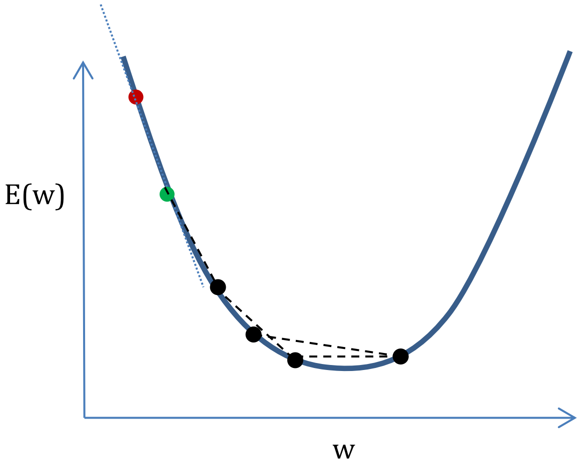


Figure 3.5 Gradient descent for 1-dimensional optimization problem

The dotted line is the derivative (tangent line) of the curve at that point and moving in the direction of steepest descent by an amount $-\mu \ \partial E/\partial w$, the new estimate for w is shown by the green dot. The figure demonstrates what can happen when the learning rate is too large. The solution oscillates about the optimum (the minimum of the curve). So the learning rate should be high enough for fast convergence but not too high.

Multidimensional (higher than three dimensions) optimization problems, such as back-propagation networks with more than two weighting parameters are more difficult to visualize, but the same idea applies.

3.3 Principal Component Analysis

Principal component Analysis (PCA) is a method for extracting the most significant features of a data set. More specifically, it is a statistical technique used to extract linear factors that represent the maximum variation in a multidimensional data set. These sources of variation are expressed as vectors of the original data set. As an illustration, consider a line in three-dimensional space described by the parametric equations of variables x, y and z:

$x = 20 + 4t \quad y = 50 + 8t \quad z = 30 + 6t \qquad \text{where t} = 1, 2, 3, \dots ,50$

This line is shown in black in Figure3.6. Normally distributed random noise is added to the points of the line and the result is the 'noisy line' whose points are shown in blue. Then the objective of PCA is to find D orthogonal vectors that represent the maximum variation in the data set. D, which is the dimensionality of the data set, is 3 in this case. Alternatively, PCA seeks to find vectors (lines in this case) such that if all the points of the data set are projected or mapped onto those vectors, we retain as much of the variation in or information from the original data set as possible, in other words, the variance is minimized. So PCA can be thought of as three-dimensional regression.

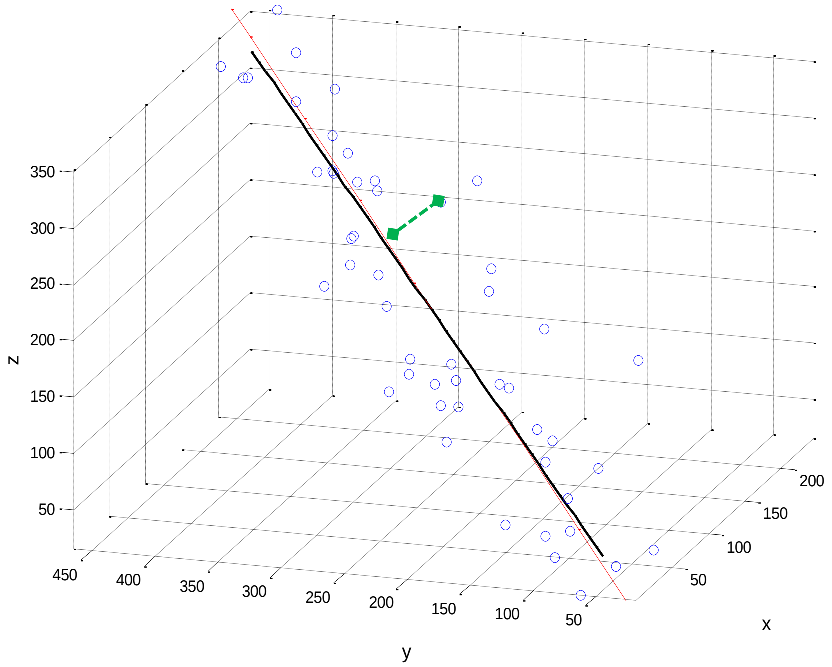


Figure 3.6 First factor extracted in linear PCA

In Figure 3.6, the red line shows the first factor (or component) obtained using PCA. The second factor found by PCA is a line perpendicular to the first factor, indicated by the orange line in Figure 3.7. It is found by imposing the constraint that this line is orthogonal to the first factor. An initial guess for the line is constructed and it is then rotated about the axis of the first factor until the axis of largest variation is found in the data set. There would be a third component (not shown in the figure) perpendicular to the plane formed by the first two factors.

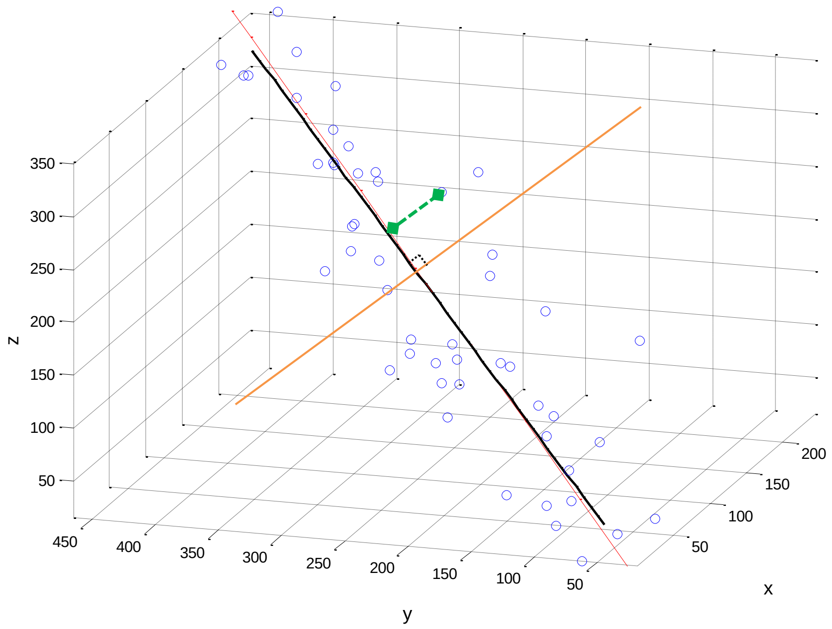


Figure 3.7 Second factor extracted in linear PCA

These component vectors are obtained as follows. First the covariance matrix is calculated. The covariance is a statistical measure of the linear relationship between two variables. If a M by N data matrix D is created for a data set of M variables each with N observations or elements, the covariance matrix which relates the variables with one another can be found with the following equations:

$$R_{(i,j)} = D_{(i,j)} - \frac{1}{N}\sum_{j=1}^{N} D_{(i,j)} \qquad V = \frac{R * R^T}{N-1}$$

Where i and j represent the row and column number of matrix D and the residuals matrix is R. So $R_{(i,j)}$ is the deviation between all observations of variable i and the average of the observations. Then the covariance matrix V will be an M by M symmetric matrix whose diagonal elements are the covariance between each individual variable and itself, that is, the *variance* of the variable. The covariance matrix will have same dimensionality as the data matrix.

The goal is to minimize $R_{(i,j)}$ and it can be shown that in order to minimize $R_{(i,j)}$, the covariance matrix V needs to be maximized. Once the covariance matrix is found, eigenvalue decomposition algorithms are applied on the matrix to find a unit vector $\vec{w}$ such that

$$V\vec{w} = \lambda\vec{w} \qquad (V - \lambda I)\vec{w} = 0$$

Where $\vec{w}$ is an eigenvector of V, λ is the eigenvalue of V and I is the identity matrix. It can be shown that in order to minimize the residuals or maximize variance this relation is satisfied. It can also be proved that since V is symmetric (V^T), its eigenvectors are orthogonal. The eigenvectors of V are the **principal components** of the data matrix. The result for the M by N data matrix is a vector of M eigenvalues which represent the magnitude of the components and an M by M matrix with each column containing an eigenvector that corresponds to the eigenvalue of the same index. Each column/eigenvector describes a factor or component.

Lastly, the magnitude of the components (the eigenvalues) are used to select components that contribute the most to the variation in the data set. The percent contribution of each eigenvector i is calculated by dividing its associated eigenvalue by the sum of all eigenvalues:

$$\frac{\lambda_i}{\sum_{j=1}^{M} \lambda_j}$$

Typically the aim is to select as few components as possible whose total percent contribution is greater than a certain desired threshold. Then say the largest q eigenvalues are selected, their eigenvectors will be used to construct a $P \times Q$ dimensional matrix T which is used to transform the $M \times N$ data matrix D into a $P \times N$ data matrix ($P < M$) of principal components.

As mentioned earlier linear PCA techniques aim to project high-dimensional data into a lower dimensional representation. This linear transformation then takes on the form

$$T = P \cdot D$$

Naturally, expressing D as it's lower dimensional counterpart T leads to a loss of some of the information in D. The following reverse transformation reconstructs the data set so that the amount of information lost can be determined:

$$D' = TP^T$$

P is such that the error between X and the reconstructed data set D' (of the same dimension as D) is minimized[18]. The loss of information referred to earlier translates into application of PCA in noise-filtering.

3.4 Nonlinear Principal Component Analysis and Inverse Neural Network Model

Nonlinear principal component analysis (NLPCA) is a generalization of standard PCA that allows for dimensionality reduction through the use of nonlinear mapping functions. Auto-associative neural network-based NLPCA [18, 19] is a three hidden-layer feed-forward network with a mapping, bottleneck and de-mapping layer. Once again, the notation of Kirby and Miranda[17] is adopted in the following explanations of the mathematical theory behind neural networks.

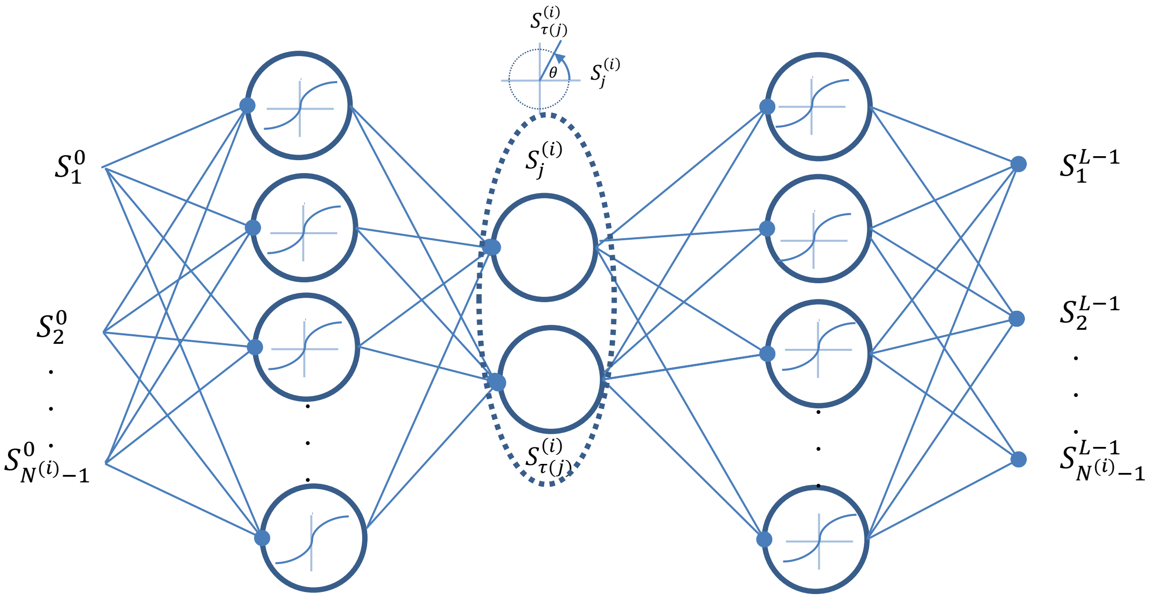


*Figure 3.8 Auto-*Associative *Neural Network [17, 20]*

The nodes in the mapping and de-mapping layers are nonlinear transfer functions (usually sigmoidal or hyperbolic tangent). The bottleneck layer encodes the lower dimensional representation of the original data set, it's outputs are the nonlinear principal components. This layer may contain either a linear or nonlinear function without affecting the nonlinear modeling capability of the entire network. All three layers are essential to achieve optimal nonlinear feature extraction. The mapping and bottle neck layers perform nonlinear feature extraction and data compression, while the de-mapping layer does the data reconstruction. If the pre and post bottleneck layers are omitted, the network merely performs linear PCA regardless of the nonlinear bottleneck node[18]. This is because linear combinations of the inputs are simply passed through the circular node. Also omitting the bottleneck layer and keeping the pre and post bottle-neck layers will lead to a trivial identity mapping: data will not be compressed to a lower dimension

The low dimensional features are transmitted to the demapping layer which performs data re-construction. The presence of the bottle neck layer of fewer nodes (lower dimension) than the input and output layers guarantees this underfitting or imperfect reconstruction of the data set which as will be seen later, translates to the use of NLPCA in noise-filtering applications[21].

In contrast to conventional NLPCA which is meant for open curve solutions , Kirby and Miranda introduced the use of circular nodes in the bottle neck layer to approximate data using closed continuous curves. Otherwise this network architecture is totally identical to that proposed by Kramer. This type of network has proved to be useful for cases were the original data set contains periodic or oscillatory patterns or circular data structures [21-24], as will be seen once inverse networks and their specific application to data sets with stiction-induced oscillations is discussed.

The bottleneck layer of the network consists of a circular unit. This unit is internally represented by a pair of nodes $S_j^{(i)}$ and $S_{\tau(j)}^{(i)}$ but together they represent a single angular variable θ since their outputs are constrained to lie on a point on the unit circle:

$$P_j^{(i)} = \cos(\theta) \qquad P_{\tau(j)}^{(i)} = \sin(\theta) \qquad \left(S_j^{(i)}\right)^2 + \left(S_{\tau(j)}^{(i)}\right)^2 = 1$$

$$R_j^{(i)} = \sqrt{\left(P_j^{(i)}\right)^2 + \left(P_{\tau(j)}^{(i)}\right)^2} \qquad S_j^{(i)} = \frac{P_j^{(i)}}{R_j^{(i)}} \qquad S_{\tau(j)}^{(i)} = \frac{P_{\tau(j)}^{(i)}}{R_j^{(i)}}$$

A recent modification of auto-associative neural network architecture , proposed by Scholz, is the Inverse neural network. This network is shown in figure 3.9 and only contains the 'inverse mapping' or data reconstruction section of an auto-associative network.

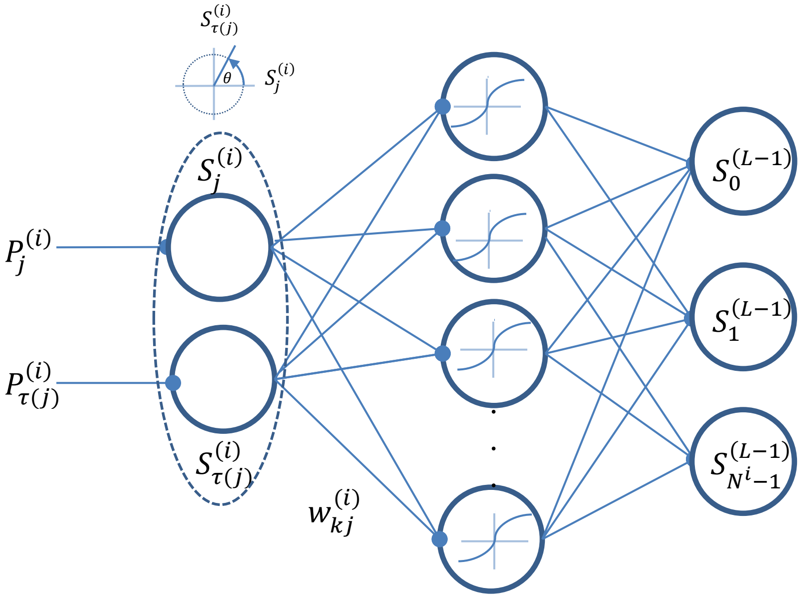


Figure 3.9 Inverse Neural Network [20]

Given the original data set as target outputs, the network estimates principal components such that the squared error between the reconstructed and original data sets is minimized. So this network estimates the principal components that reconstruct the data set such that the error between the reconstructed and actual data set is minimized. But just as an auto-associative network, since the principal components are a 'compressed' or low dimensional representation of the data set, there is a loss of information in the reconstructed data set.

The weights are denoted as $w_{kj}^{(i)}$, the weight that connects node j in the previous layer to node k in the current layer i. For instance $w_{31}^{(2)}$ is the weight that connects node 1 in the bottle-neck layer (first layer) to node 3 in the mapping layer (second layer).

The unknown inputs $P_j^{(i)}$ and $P_{\tau(j)}^{(i)}$ are to be determined along with the weights $w_{kj}^{(i)}$in the network by the back-propagation algorithm. Equations (2) are parametric equations that force $P_j^{(i)}$and $P_{\tau(j)}^{(i)}$to be the coordinates of points on a circle. These equations ensure that the trigonometric constraint in (1) that is, point $S_j^{(i)}, S_{\tau(j)}^{(i)}$ lies on a unit circle, is satisfied. The point $S_j^{(i)}, S_{\tau(j)}^{(i)}$ is represented by a single angle θ, the principle component for one sample (a d-dimensional vector) of the original data set.

Initially, the parameters $P_j^{(i)}, P_{\tau(j)}^{(i)}$ and $w_{kj}^{(i)}$ are randomly generated and $G_j's$ are calculated.The sum squared error over all samples/observations N:

$$P_j^{(i)} = \sum_{j=1}^{N^{L-1}-1} w_{kj}^{(i-1)} S_k^{(i-1)} + b_j^{(i)}$$

$$E = \frac{1}{2} \sum_{j=0}^{N^{L-1}-1} \left(S_k^{(L-1)} - G_j\right)^2$$

	For Circular Node	For Sigmoidal Nodes
$S_j^{(i)} =$	$\frac{P_j^{(i)}}{\sqrt{\left(P_j^{(i)}\right)^2 + \left(P_{\tau(j)}^{(i)}\right)^2}}$	$\sigma_j^{(i)}\left(P_j^{(i)}\right) = \frac{1}{1 + \exp(-P_j^{(i)})}$
$\frac{\partial E}{\partial b_j^{(i)}} =$	$\frac{\partial E}{\partial S_j^{(i)}} \frac{\partial S_j^{(i)}}{\partial b_j^{(i)}} + \frac{\partial E}{\partial S_{\tau(j)}^{(i)}} \frac{\partial S_{\tau(j)}^{(i)}}{\partial b_j^{(i)}}$	$\frac{\partial E}{\partial S_j^{(i)}} \frac{\partial S_j^{(i)}}{\partial b_j^{(i)}}$
$\frac{\partial E}{\partial w_{jk}^{(i)}} =$	$\frac{\partial E}{\partial S_j^{(i+1)}} \frac{\partial S_j^{(i+1)}}{\partial w_{jk}^{(i)}} + \frac{\partial E}{\partial S_{\tau(k)}^{(i)}} \frac{\partial S_{\tau(k)}^{(i)}}{\partial w_{jk}^{(i)}}$	$\frac{\partial E}{\partial S_j^{(i+1)}} \frac{\partial S_j^{(i+1)}}{\partial w_{jk}^{(i)}} = \frac{\partial E}{\partial S_j^{(i+1)}} \frac{\partial \left[\sigma_j^{(i)}\left(P_j^{(i)}\right)\right]}{\partial w_{jk}^{(i)}}$
$\frac{\partial S_j^{(i)}}{\partial S_k^{(i-1)}} =$	$\frac{\partial S_j^{(i)}}{\partial b_j^i} \frac{\partial b_j^i}{\partial S_k^{(i-1)}} + \frac{\partial S_j^{(i)}}{\partial b_{\tau(j)}^i} \frac{\partial b_{\tau(j)}^i}{\partial S_k^{(i-1)}}$	$\frac{\partial S_j^{(i)}}{\partial b_j^i} \frac{\partial b_j^i}{\partial S_k^{(i-1)}}$

$$S_j^{(i)} = \frac{P_j^{(i)}}{\sqrt{\left(P_j^{(i)}\right)^2 + \left(P_{\tau(j)}^{(i)}\right)^2}} \qquad R_j^{(i)} = \sqrt{\left(P_j^{(i)}\right)^2 + \left(P_{\tau(j)}^{(i)}\right)^2} \qquad S_j^{(i)} = \frac{P_j^{(i)}}{R_j^{(i)}}$$

$$\frac{\partial E}{\partial S_j^{(L-1)}} = S_j^{(L-1)} - G_j$$

$$\frac{\partial E}{\partial S_j^{(i-1)}} = \sum_{j=0}^{N^{(i)}-1} \frac{\partial E}{\partial S_j^{(i)}} \frac{\partial S_j^{(i)}}{\partial S_k^{(i-1)}}$$

4. PROCEDURE FOR STICTION QUANTIFICATION

4.1 Introduction

This section demonstrates the development of two neural network models: the noise suppressor, which removes noise from the stiction patterns and the predictor, which estimates the degree of stiction given the controller output OP(t) signal from a stiction pattern. There are also data pre-processing and post processing steps involved in the development of both models. The procedure is outlined as follows:

1. 1275 degree of stiction patterns i.e. OP(t) and PV(t) are generated by closed loop simulation of a Single-Input Single Output process. Each pattern is generated by specifying a particular value of static and dynamic friction (fs and fd) in the valve stiction model.
2. 20 separate degrees of stiction are simulated and will be used as the'test' patterns. 10 of these will have no added noise and the other 10 will contain noise.
3. A 100 time step interval of the controller output OP(t) signal from each of the1275 patterns and associated (fs,fd) values are used to train a feedforward network to predict fs given OP(t). The is the **predictor** model
4. The OP(t)-PV(t) signal from the 10 'noisy' test cases are denoised using the inverse neural network model described in the previous section. This is the noise-suppresor model. The OP(t) signal is then extracted.

5. The OP(t) ‘test’ signals, whose degrees of stiction are known, are then fed into the predictor model to test if the model is capable of accurately estimating fs and fd.

Neural networks (or in general, data based modeling) are used here for 3 major reasons:

1. Ease of implementation and the speed at which they model complex nonlinear behavior
2. The high flexibility of these models. The same exact neural network architectures used here can be used in a different application, regardless of process type, nature of valve model or controller.
3. It is difficult to find non-data based models that can perform the type of complex modeling involved in this work. That is, take several sets of large time series oscillations and model the relation between these series and other parameters.

4.2 Generation of Stiction Patterns by Closed-Loop Simulation of Single- Input Single-Output (SISO) Process

The goal is to create a predictive model that takes a controller output OP(t) or process variable PV(t) stiction pattern from a particular process control loop as input and produces an estimate of the static and dynamic friction (f_s, f_d). The process model used throughout this demonstration is a single-input single-output (SISO) linear time invariant (LTI) level control process described by the following discrete time state space equations:

$$x(t+1) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t)$$

Note that there is no noise or disturbances in the model so that predictor is based on 'ideal' stiction behavior in this particular process. The physical valve model for the pneumatic control valve described in section 2.3 serves as the valve stiction model and by specifying a value for f_s, f_d and simulating the closed loop system shown in Figure 4.1 OP(t), MV(t) and PV(t) stiction-induced oscillations can be generated. Both f_s and f_d both have units of pound forces lbf.

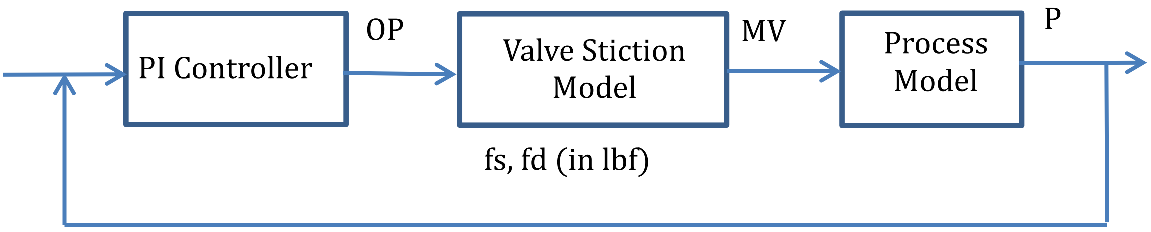


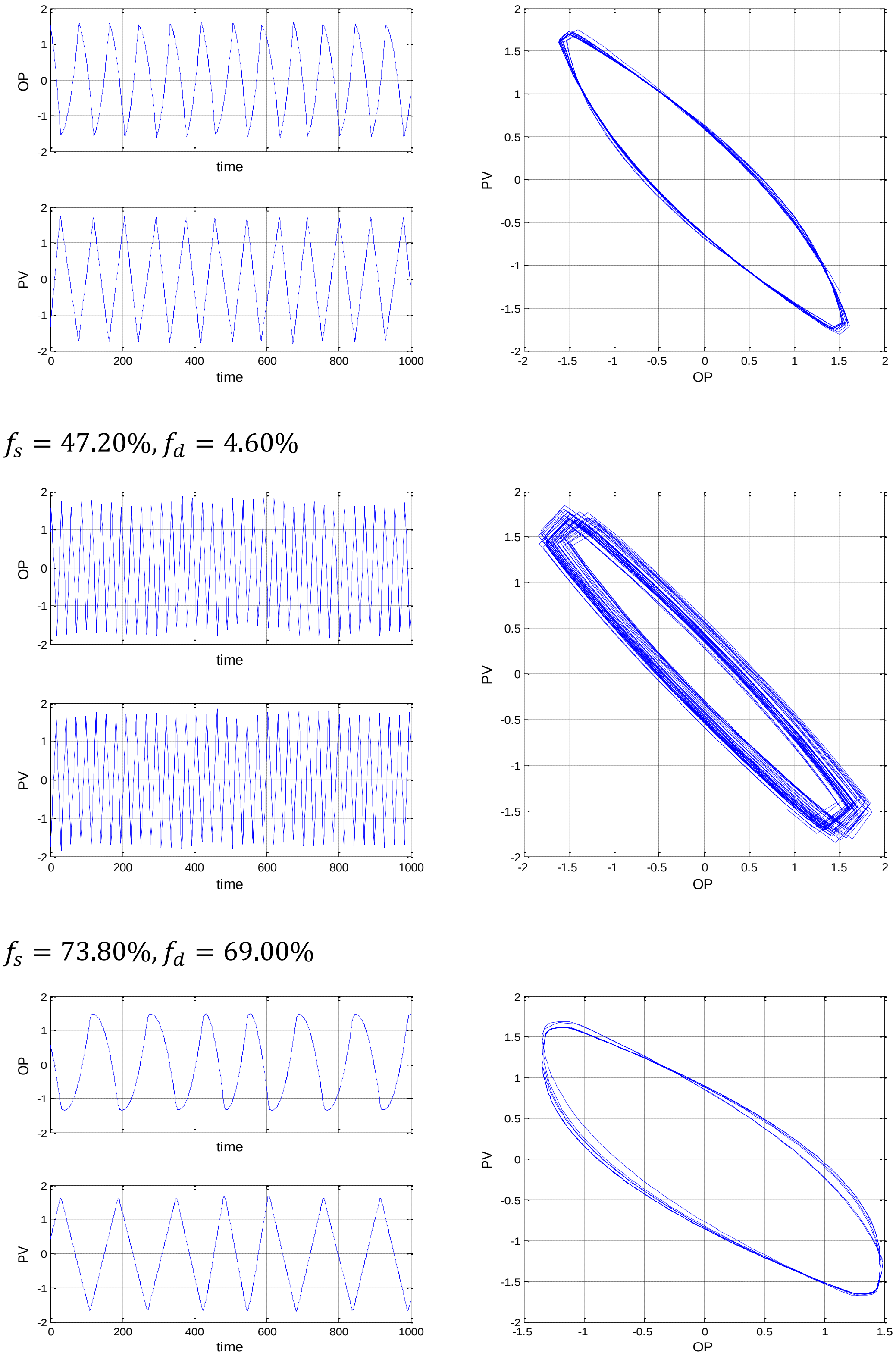
Figure 4.1: Stiction control loop: no process noise or disturbance

1275 such simulations were run to generated 1275 sets of OP(t=1…1000) and PV(t=1…1000). An assumption made in all runs is that f_d is less than f_s, which is typically the case. If static friction is overcome and the valve stem experiences a slip jump and begins to move, then the dynamic or moving friction f_d must be less than the friction f_s that was preventing the valves movement in the first place. A maximum value of f_s is arbitrarily chosen as 500 lb_f. Keeping in mind that $f_d < f_s$ the following algorithm was implemented in MatLab to generate the 1275 sets of $\langle f_s, f_d \rangle$ values, each set representing a different degree of stiction:

```
$for\ f_s = 500{:}-10{:}\,10$
    $for\ f_d = f_s - 10{:}-10{:}\,0$
            simulate $\langle f_s, f_d \rangle$ for the closed loop system
    end
end
```

So in the first simulation $\langle f_s, f_d \rangle = \langle 500{,}490 \rangle$,in the second $\langle f_s, f_d \rangle = \langle 500{,}480 \rangle$ $\langle f_s, f_d \rangle = \langle 500{,}470 \rangle$ in the third and so on until $\langle f_s, f_d \rangle = \langle 500{,}0 \rangle$. Then in the second iteration of the outer loop, f_s is decremented by 10 to a fixed value of 490, and f_d is reduced from 480 to 0 also in decrements of 10 lbf. A total of 1275 stiction patterns are generated. The degree of stiction f_s and $f_d$will be expressed in terms of percentages from now on, with 500 lbf being the maximum possible amount of friction

Next, a separate set of 20 stiction patterns with different degrees of stiction than the 1275 patterns generated as the ‘test’ cases: 10 of these contain no added noise and the other 10 will contain input and output process noise.



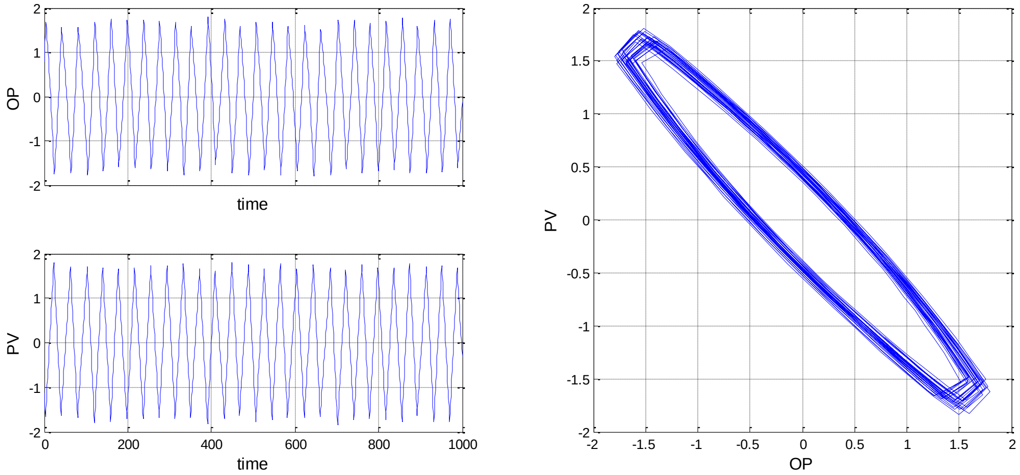
Loop 1: $f_s = 97.40\%, f_d = 64.40\%$

Loop 2: $f_s = 47.20\%, f_d = 4.60\%$

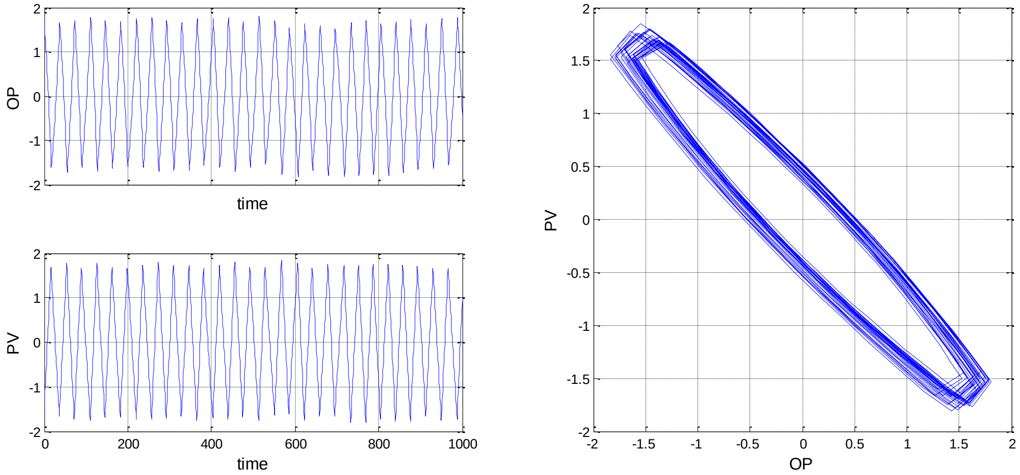
Loop 3: $f_s = 73.80\%, f_d = 69.00\%$

Figure 4.2: Stiction test cases: no process noise

Loop 4: $f_s = 70.40\%, f_d = 21.00\%$



Loop 5: $f_s = 62.60\%, f_d = 16.80\%$



Loop 6: $f_s = 10.60\%, f_d = 4.80\%$

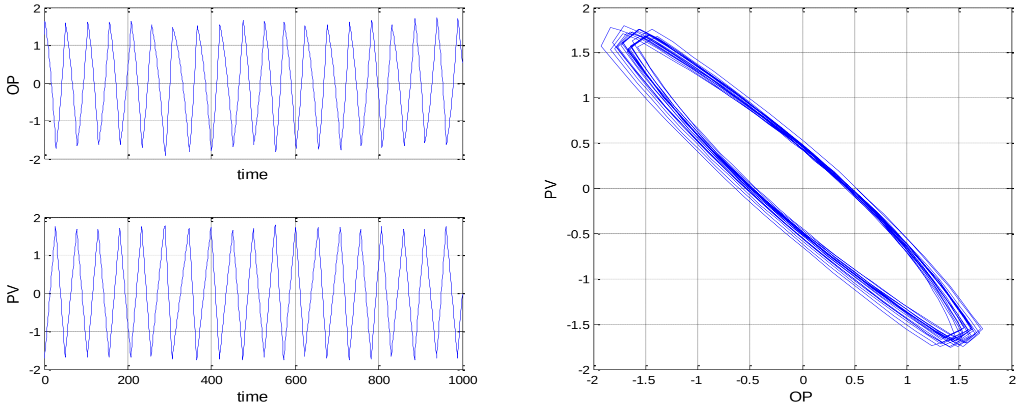
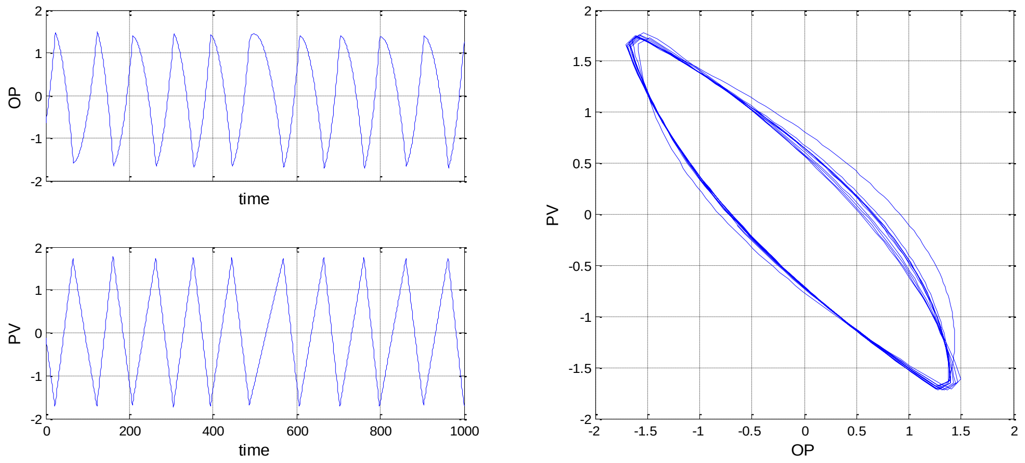
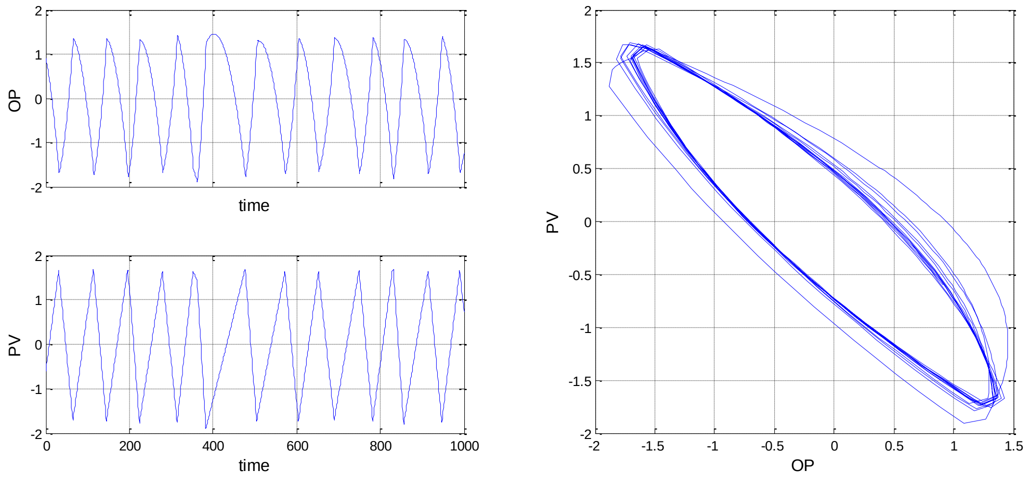


Figure 4.2 Continued

Loop 7: $f_s = 16.00\%, f_d = 11.20\%$



Loop 8: $f_s = 9.40\%, f_d = 6.00\%$



Loop 9: $f_s = 33.20\%, f_d = 19.40\%$

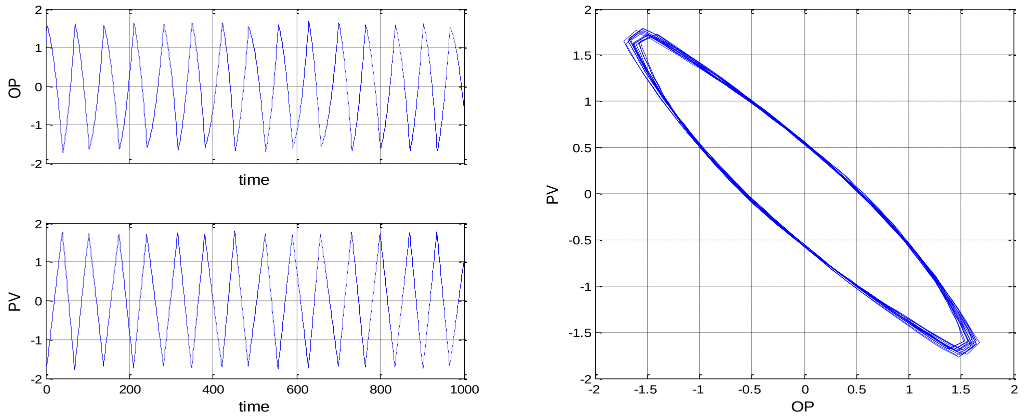


Figure 4.2 Continued

Loop 10: $f_s = 64.80\%, f_d = 7.20\%$

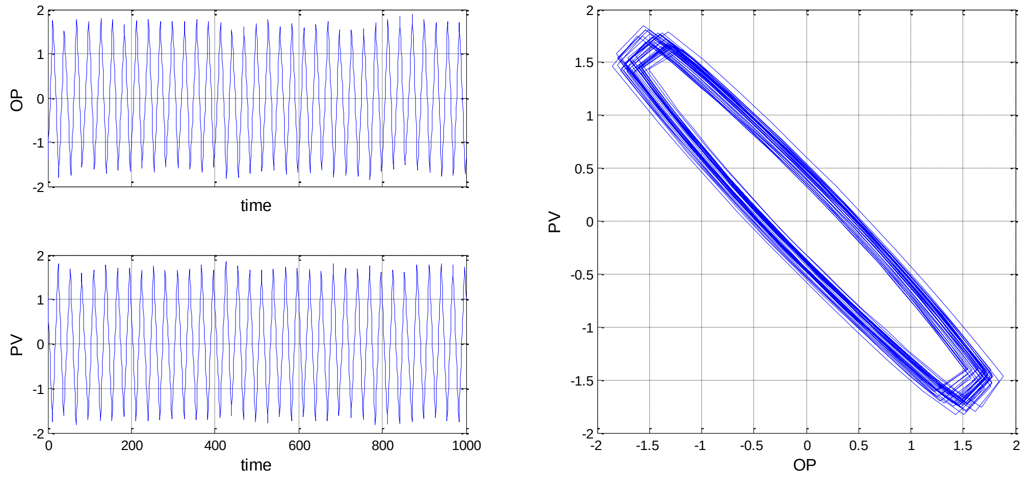


Figure 4.2 Continued

The 10 noisy patterns are generated by incorporating disturbance terms into the state space model

$$x(t+1) = Ax(t) + B[u(t) + w(t)]$$

$$y(t) = Cx(t) + z(t)$$

where $w(t)$ and z(t) are input and output process noise, respectively. Each case has stiction case has either one or the other as shown in Figures 4.3 and 4.4.

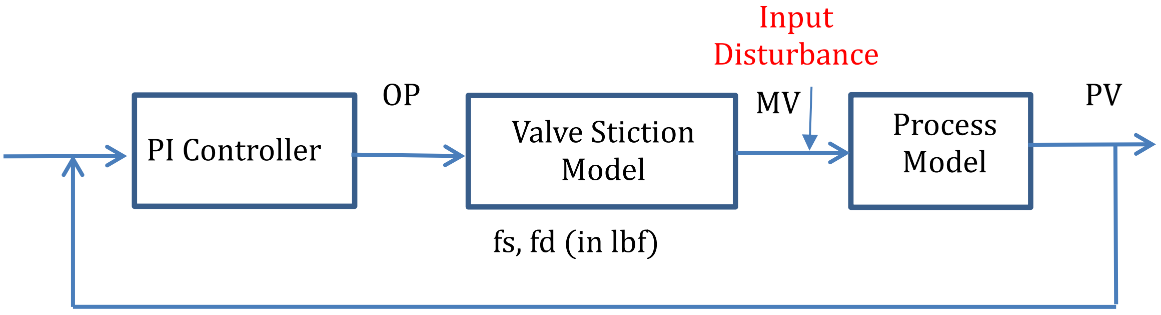


Figure 4.3: Stiction control loop: with input process noise

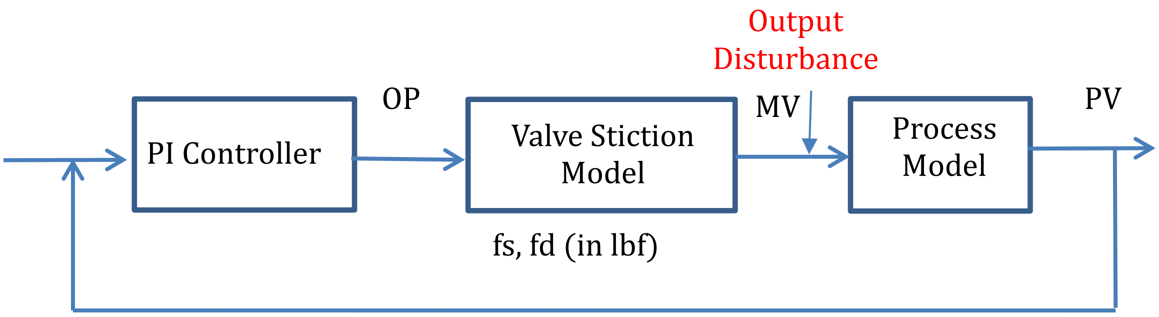


Figure 4.4: Stiction control loop: with output process noise

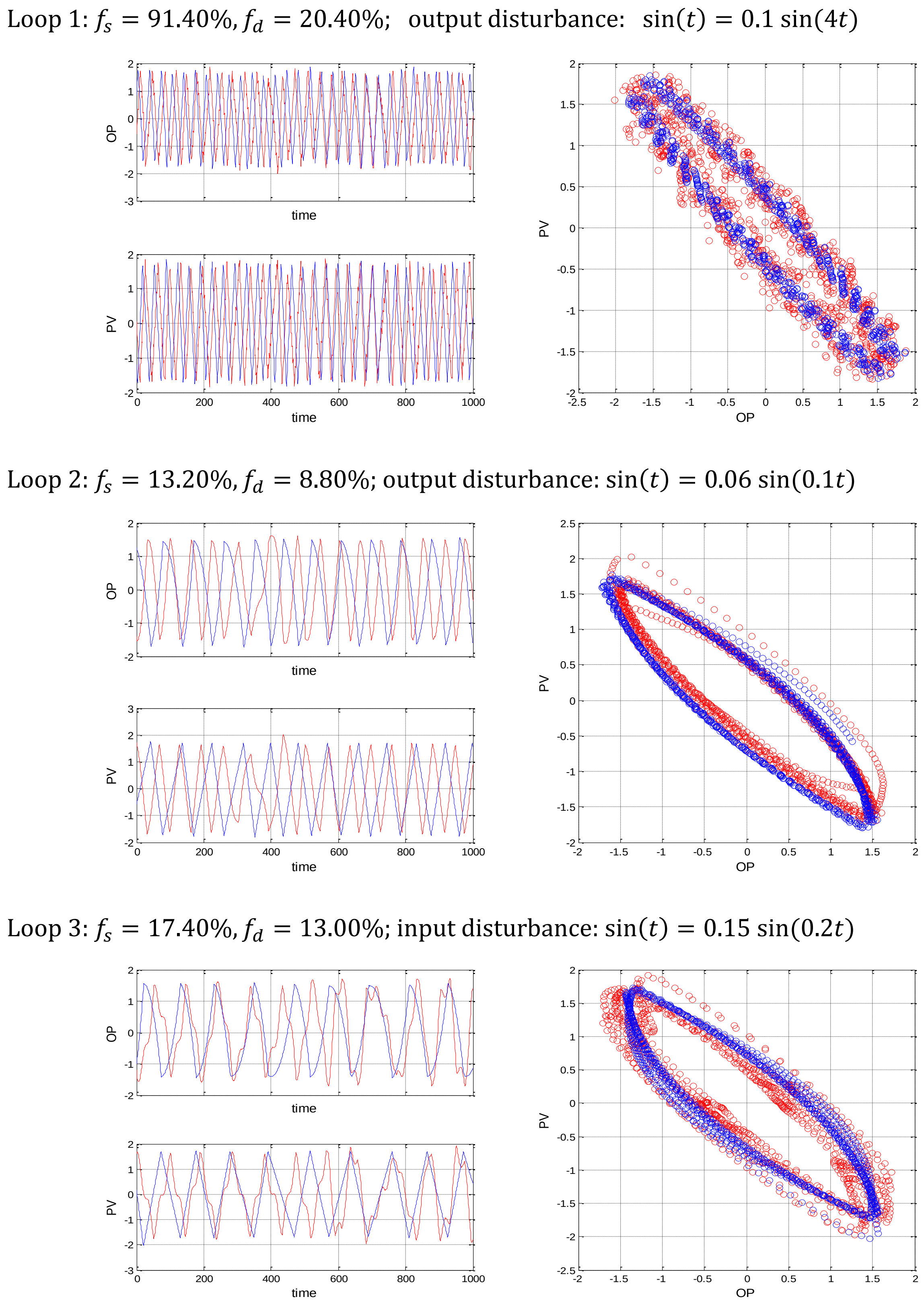
Three noise models are used:

- Sinusdoidal: $A * \sin(f * t)$, where A is the amplitude and f is the frequency
- Randomly generated normally distributed, gaussian or white noise
- Brownian colored noise is generated filtering zero mean white noise with an autoregressive model of order 63. MatLab's model is given by:

$$\sum_{k=0}^{63} a_k y(n-k) = w(n) \qquad a_0 = 1 \qquad a_k = \left(k - 1 - \frac{\alpha}{2}\right)\frac{a_{k-1}}{k} \quad k = 1,2,\ldots$$

Where $w(n)$ is a zero mean white noise process. The reader is referred to [25] for a description of the autoregressive model and power law colored noise generation.

Figure 4.5 shows OP versus time, PV versus time and OP versus PV plots for stiction cases with and without process noise.

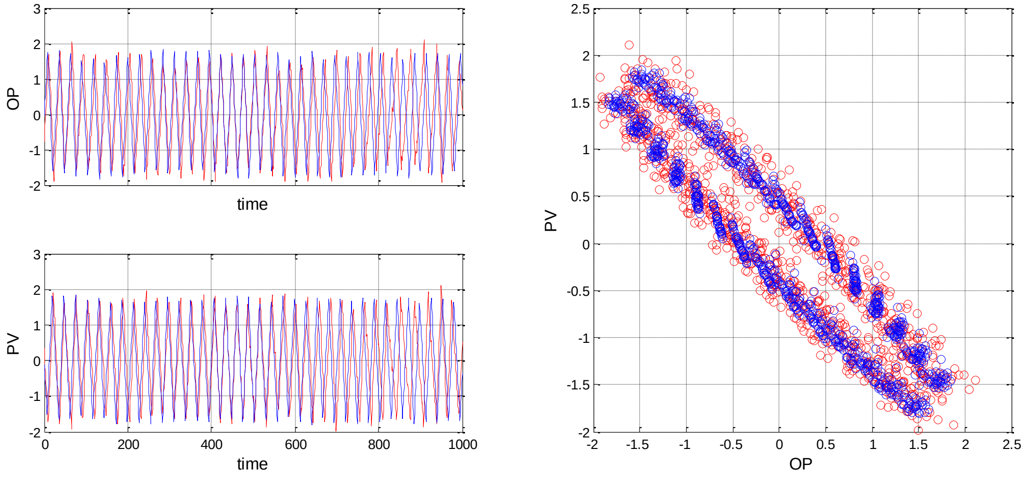

Loop 1: $f_s = 91.40\%, f_d = 20.40\%$; output disturbance: $\sin(t) = 0.1\sin(4t)$

Loop 2: $f_s = 13.20\%, f_d = 8.80\%$; output disturbance: $\sin(t) = 0.06\sin(0.1t)$

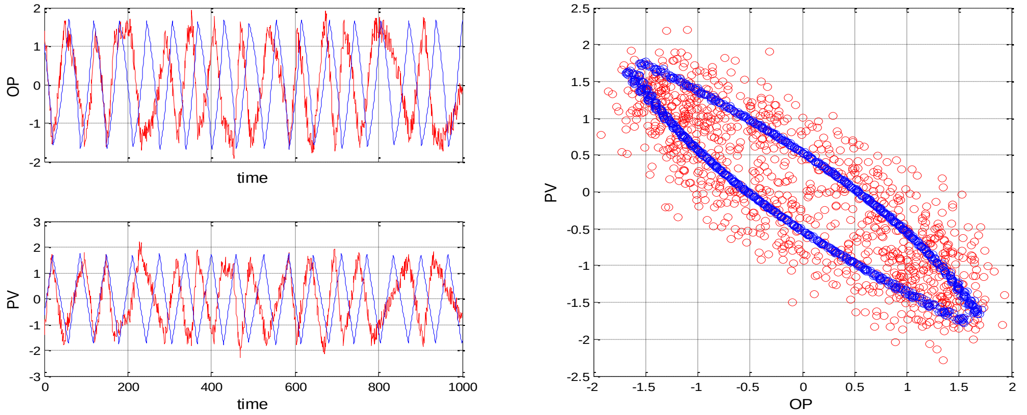
Loop 3: $f_s = 17.40\%, f_d = 13.00\%$; input disturbance: $\sin(t) = 0.15\sin(0.2t)$

Figure 4.5: Stiction Test Cases: With Process Noise

Loop 4: $f_s = 46.80\%, f_d = 4.00\%$: output disturbance: $0.1 *$ random gaussian noise



Loop 5: $f_s = 29.00\%, f_d = 16.00\%$: output disturbance: $0.2 *$ random gaussian noise



Loop 6: $f_s = 17.80\%, f_d = 8.00\%$; input disturbance: $0.3 *$ random gaussian noise

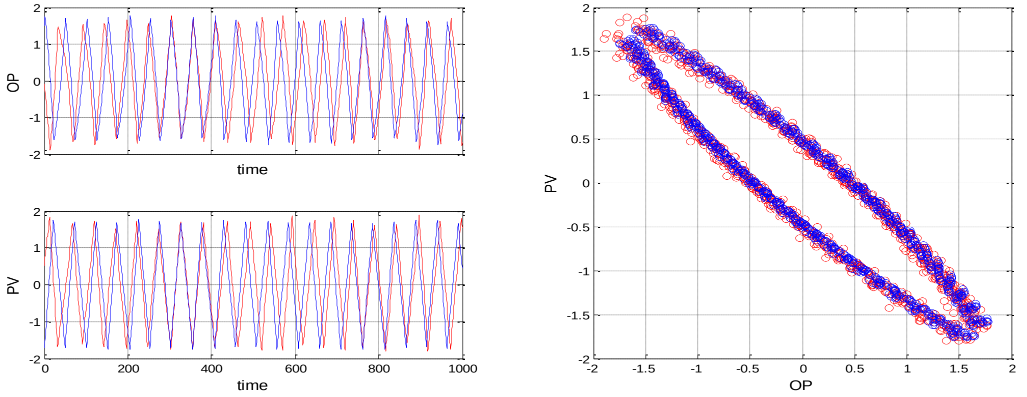
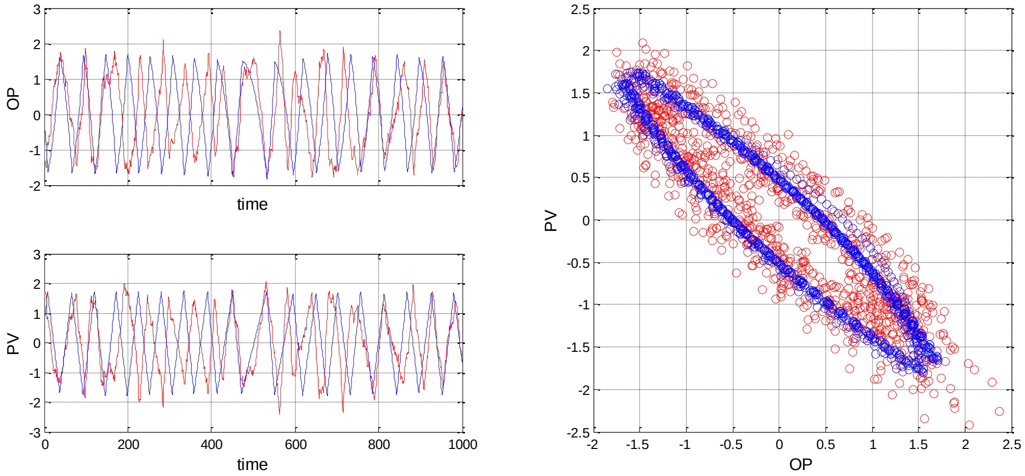
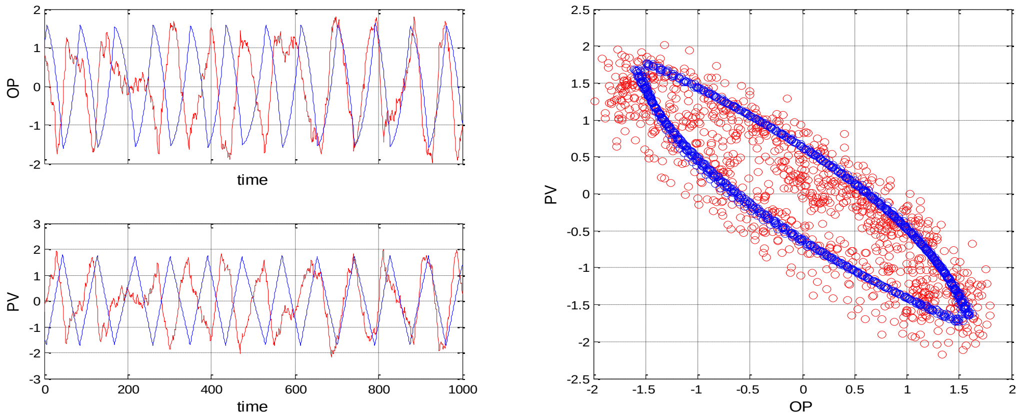


Figure 4.5 Continued

Loop 7: $f_s = 86.00\%, f_d = 42.00\%$; output disturbance: $0.1 * \text{colored noise}$



Loop 8: $f_s = 100.00\%, f_d = 67.60\%$; output disturbance: $0.1 * \text{colored noise}$



Loop 9: $f_s = 47.20\%, f_d = 10.40\%$; input disturbance: $0.1 * \text{colored noise}$

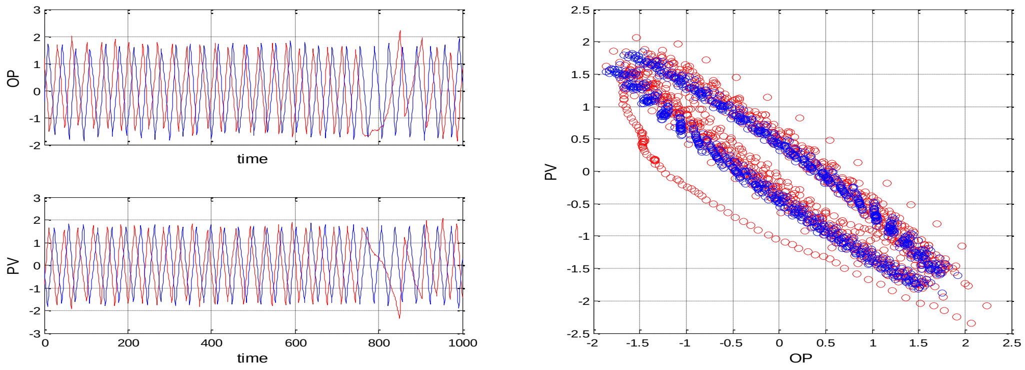


Figure 4.5 Continued

Loop 10: $f_s = 57.80\%, f_d = 22.20\%$; output disturbance: $0.1 * $ colored noise

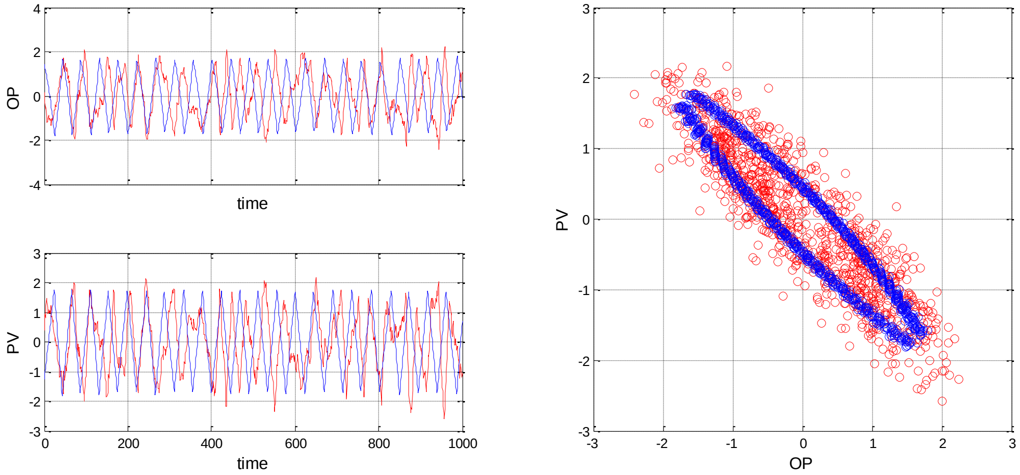


Figure 4.5 Continued

4.3 Correlation between OP(t) and PV(t) Signals

The cross-correlation coefficient measures magnitude and direction of the linear relationship between two signals. It is defined as the sample covariance of the two signals divided by their sample standard deviations. For each of the 1275 sets of OP(t) and PV(t) stiction patterns (each representing a different degree of stiction) the correlation coefficient is determined. The aim is to show that if OP(t) and PV(t) are strongly positively or negatively correlated, then it will be redundant to use both signals to develop the predictor model that estimates the stiction parameters fs and fd given a stiction pattern. Furthermore, it is assumed that the behaviour of PV(t) is implicitly accounted for by the OP(t) signal and vice versa. Then it may be possible to use either the OP(t) or PV(t) signal alone in developing the predictor model that estimates fs and fd given either OP(t) or PV(t) data. The closer the correlation coefficient is to -1 or +1, the stronger the linear relationship between the two signals. The correlation coefficient is given by:

$$r = \frac{c}{std1\ \times std2}$$

$$c = \frac{1}{N-1}\sum_{t=1}^{N}\left(OP_t - \overline{OP}\right)\left(PV_t - \overline{PV}\right)$$

$$std1 = \sqrt{\frac{\sum_{t=1}^{N}\left(OP_t - \overline{OP}\right)^2}{N}} \qquad std2 = \sqrt{\frac{\sum_{t=1}^{N}\left(PV_t - \overline{PV}\right)^2}{N}}$$

where c is the covariance between OP_t and PV_t, and $std's$ are the standard deviations. So for each degree of stiction, the correlation coefficient netween between OP_t and PV_t $r's$ is calculated.

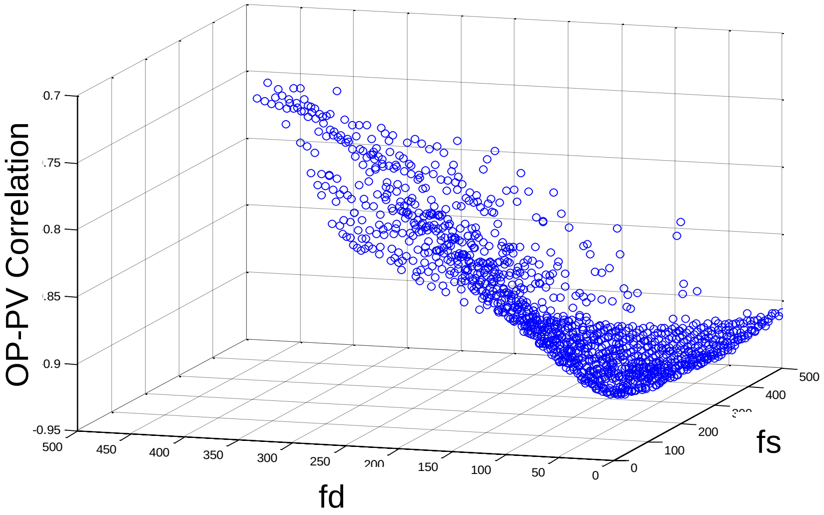


Figure 4.6: Correlation between OP(t) and PV(t)

Figure 4.6 plots the correlation coefficient for the 1275 degrees of stiction. The plot shows a high negative correlation between OP(t) and PV(t), and a particularly high concentration of points at correlation coefficients closer to -1. This indicates a *strong* negative correlation between the two signals for all degrees of stiction (for all possible combinations of fs and fd values, with the constraint $fs \leq 500$ and $fd < 500$. So there is a strong linear relationship between the two signals. Unlike the covariance (which is used to calculate the correlation), the correlation provides information about the *strength* of the linear association between two variables. In this work, It is assumed that MV(t) may not be explicitly available, as is the case in process control loop from industrial systems. The OP(t) signals are selected for use as the input into the feedforward network predictor discussed in the next section.

4.4 Noise Suppression of Test Stiction Patterns using Inverse Network-based Nonlinear PCA (INLPCA)

As seen in the figures in section 4.2, the relationship between PV(t) and OP(t) follows a closed curve since both are cyclic pattern. As discussed in Section 3.4, the inverse neural network with a circular node is efficient at approximating circular data structures. This inverse network model is used to filter the noise from the PV versus OP(t) trend of the noisy stiction test patterns so that the underlying stiction behavior can be extracted. The network takes as output a 2 by 100 matrix, the first row being 100 time step interval of the OP(t) signal and the second row is the PV(t) signal for the same time interval. In Figure 4.7, the network is depicted as a black box model which takes in the noisy 2-dimensional, 100-sample data set and attempts to construct noiseless PV(t) and OP(t) signals.

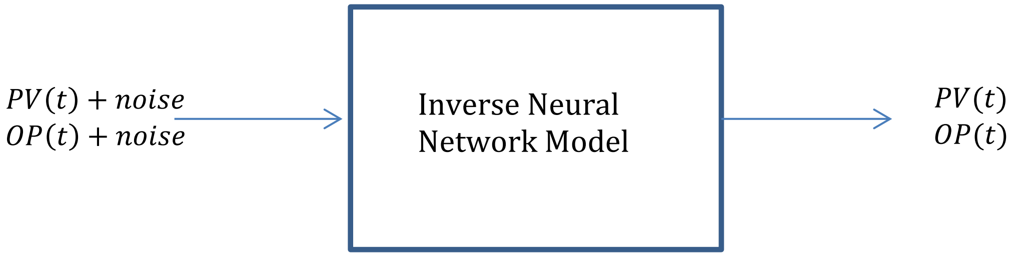


Figure 4.7: Inverse neural network denoises the circular data structure formed in OP(t) versus PV(t)

The input to the network is a 2 by 1000 data matrix. The first row containing PV(t) and the second row, OP(t) from the noisy test cases.

4.5 Approximating Periodicity of Noisy Test Stiction Patterns using Auto-Correlation Function

Next, in order to construct a new OP(t) for each of the 10 noisy test cases, the period of the oscillations in the original noisy signals needs to be approximated using autocorrelation function. It is a sequence representing the similarity between two wave signals as a function of the time lag applied to one of them. Autocorrelation is the cross-correlation between a signal and itself and is useful in uncovering patterns or repetitions in a signal and can therefore determine durations of cycles in periodic phenomena. In this case, it is used to estimate the periodicity of the noisy oscillatory signal.

The auto-correlation is obtained using the following expressions [26]

$$r_k = \frac{c_k}{c_0} \quad c_k = \frac{1}{N-1}\sum_{t=1}^{N-k}(y_t - \overline{y})\,(y_{t+k} - \overline{y}) \qquad c_0 = \sum_{t=1}^{N-k}(y_t - \overline{y})^2$$

Where c_k is the covariance between y_t and y_{t+k}, and c_0 is the sample variance of y_t. So for all t’s, the autocorrelation coefficient $r_k's$ are calculated. The resulting autocorrelation coefficients. Figure 4.8 shows a pictorial representation of the autocorrelation of a signal y.

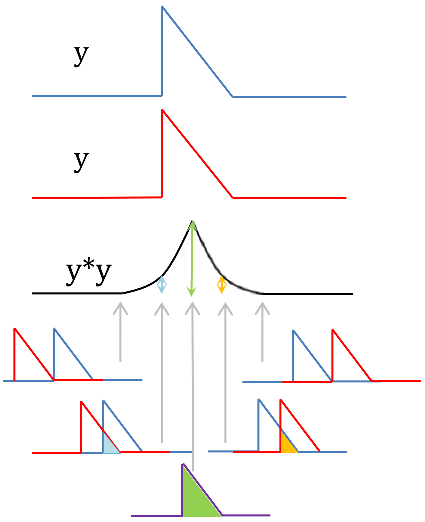


Figure 4.8: Pictorial Representation of the Auto-Correlation of a signal

Each of the 1275 OP(t) patterns are 1000 time steps long. However a particular degree of stiction can be defined by just one cycle of the oscillations, since one cycle contains information about the frequency, amplitude and shape of the stiction oscillation. In this work, 100 time interval section of each OP(t) pattern is used from each of the OP(t) stiction

patterns. Figure 4.4 below shows how the 100 time step intervals were chosen in order to normalize all the data. For each OP(t) pattern, The first the first point was chosen as the beginning of the first cycle of oscillations, then the 100 time step window after that point was extracted as shown in Figure 4.9.

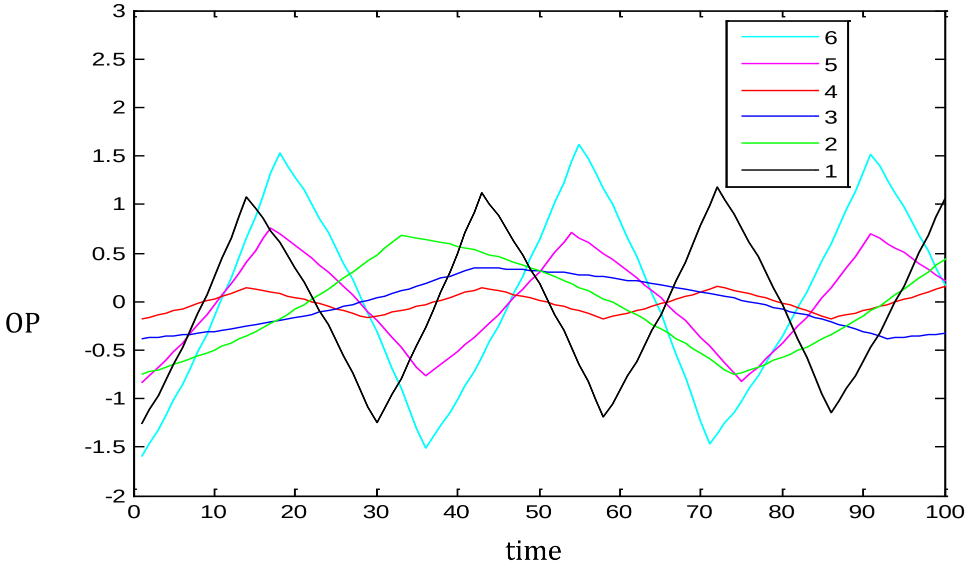


Figure 4.9: Normalizing the stiction signals: 6: strong stiction, 1: weak stiction

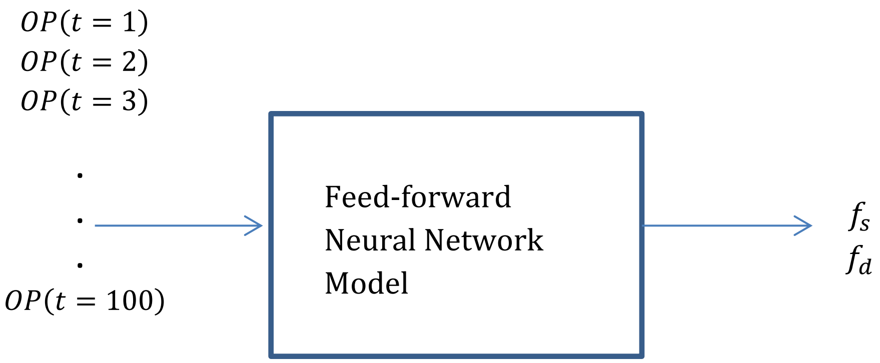


Figure 4.10: Feedforward neural network predictor

Matlab’s extensive and highly efficient neural network toolbox was used to develop the feedforward neural network predictor. In Figure 4.10, it is shown as a black box model relating certain input vectors to their corresponding output vectors.

In the **training** and **validation** phase of developing this neural network-based predictor, the network learns the relationship between the 1275 OP(t) signals and their corresponding f_s, f_d values. 80% of the 1275 OP(t) signals are allocated for training and 20% for validation. During training, the first input OP(t) vector of the training set is fed into the network, initial weights are randomly generated in the network and fs and fd is calculated based on the feed-forward network architecture described in Section 3.2. Note that the number of nodes in the hidden layer of this network is chosen as 25 based on tests done to see the effect of number of nodes on model accuracy. Generally, the goal is to use the fewest number of nodes necessary for accurately modeling the relationships between the input and output vectors. Too many nodes may lead to over-fitting the relationships, which will be discussed shortly.

fs and fd are calculated for all OP(t) patterns from the training set . The error between the calculated stiction parameters ($\hat{f}_s, \hat{f}_d$) and actual f_s, f_d is determined., then the backpropagation algorithm is applied to update all the weights in the network and subsequent iterations are implemented until the total error between $\hat{f}_s, \hat{f}_d$ and $f_s, f_d$is minimized.

Simultaneously, the same procedure is applied to the validation set. The purpose of validation is to ensure that the network does not overfit the relationships. So if at a particular iteration, the mean squared error of the testing cases decreases but the error of validation sets increases or remains constant, training is stopped to prevent 'overtraining' the network.

It is important to note that the network does not 'fit' the relationships for validation sets, only for the testing sets. In other words the weights are adjusted based on errors of the testing set alone. The validation set only serves as a test of the networks generalization capability, that the network has effectively 'understood' and modeled the relationship between OP(t) and f_s, f_d, instead of merely 'memorizing' or overfitting. The network is able to take as input 100 time steps of OP(t) pattern and predicts f_s and f_d

5. STICTION QUANTIFICATION RESULTS AND DISCUSSION

5.1 Noise Suppression of Test Stiction Patterns using Inverse Network-based NLPCA

As mentioned in the previous sections, there are 10 test cases of stiction patterns with no noise and 10 cases of stiction patterns with added noise. All the stiction simulations generate OP(t) and PV(t) data. For each of the 'noisy' cases, the goal is to use an inverse neural network-based nonlinear principal component analysis (INLPCA) to remove noise from the OP(t) versus PV(t) trend, which has a closed curve data structure. Since the actual or desired trend is known. That is, all stiction patterns are based on simulation where the degree of stiction (a particular value of fs and fd) is specified.

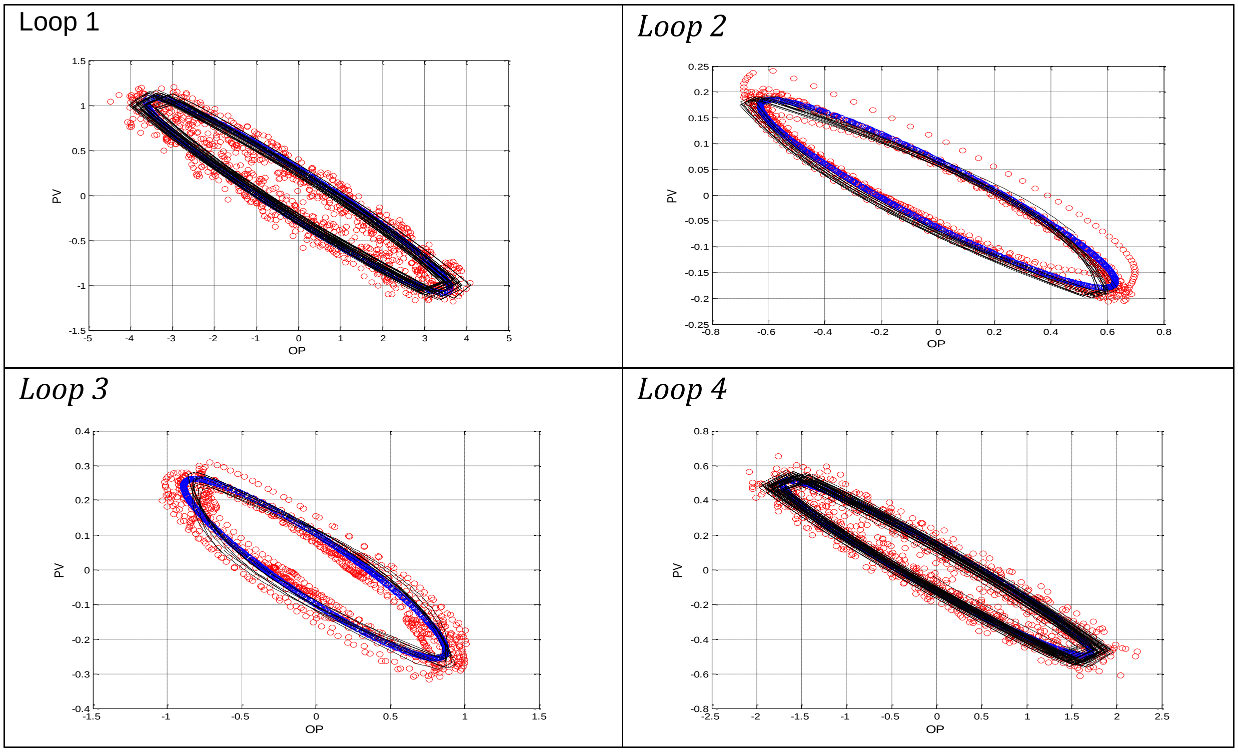
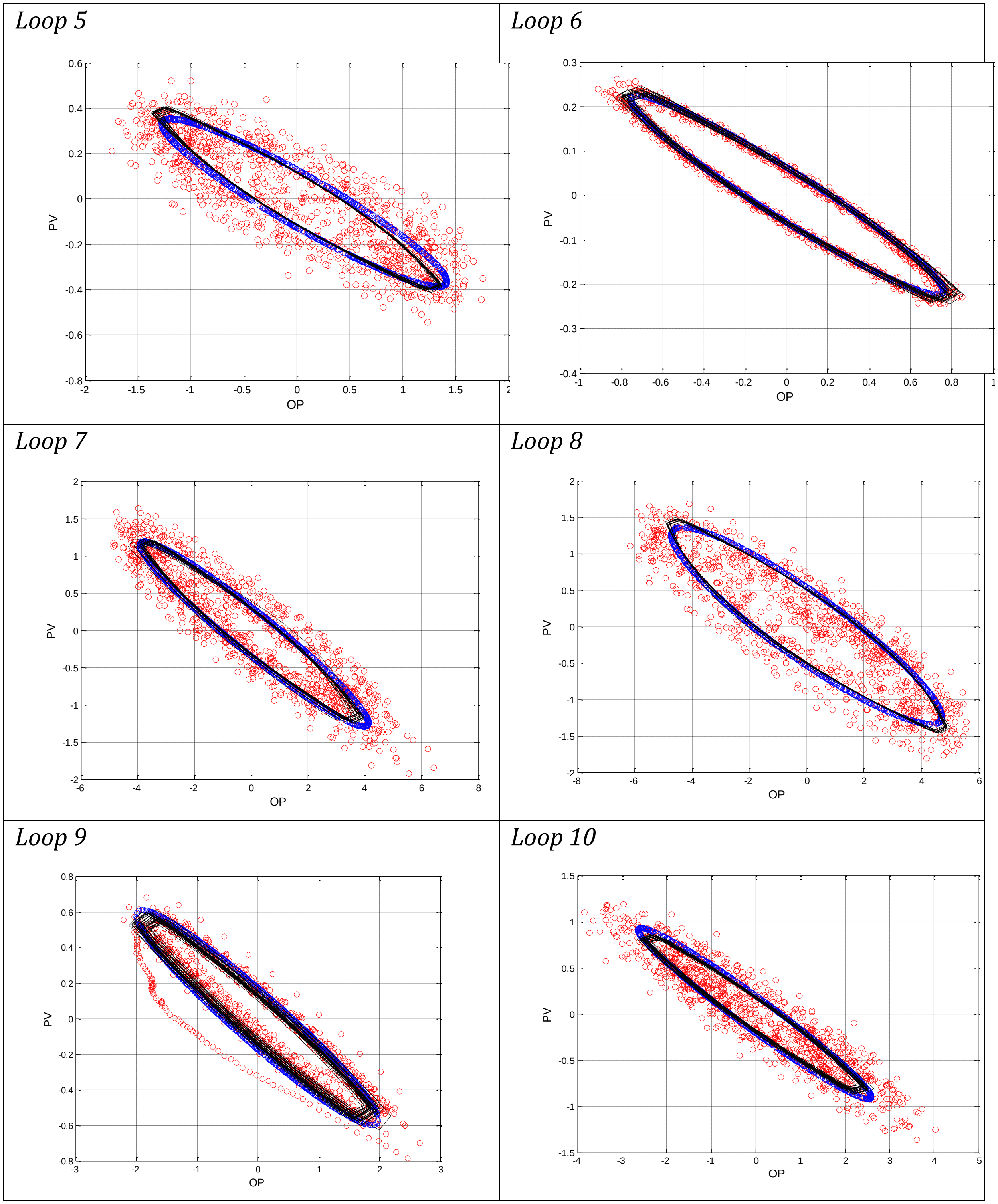


Figure 5.1: OP versus PV plots for the 10 test cases Red: Noisy Trend, Black: Actual Trend, Blue: Approximation of Actual Trend using INLPCA



Loop 5

Loop 6

Loop 7

Loop 8

Loop 9

Loop 10

Figure 5.1 Continued

5.2 Estimating Periodicity of OP(t) Patterns

Next, by the cross-correlation analysis performed in section 4.4, it was concluded that it is unnecessary to use both OP(t) and PV(t) to determine the degree of stiction, and that only the OP(t) pattern is needed. In order to construct a 'noise-free' OP(t) pattern from the INPLCA solution from the previous section, the periodicity of the OP(t) pattern for all ten 'noisy' test cases was approximated using the autocorrelation function discussed in Section 4.5. The INLPCA technique does not know that OP(t) and PV(t) are time dependent trends, it only estimates the *relationship* between OP and PV. By approximating the period of the original noisy OP(t) trend, the frequency of the signal is approximated as well. In other words, this techniques allows for the determination of how many points on the closed curve are in one cycle of the oscillation. Then the OP(t) signals were reconstructed.

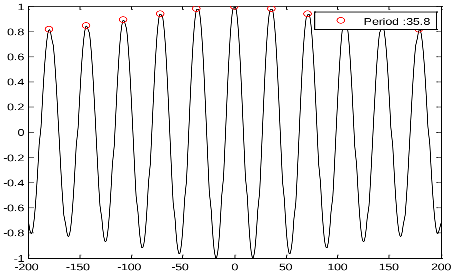
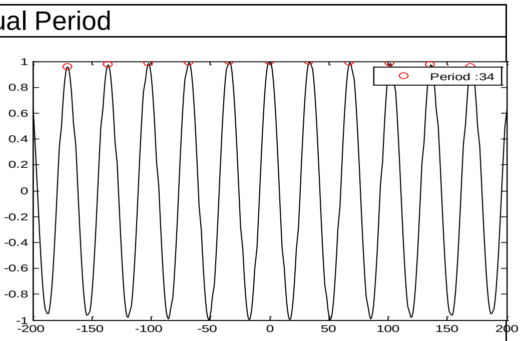
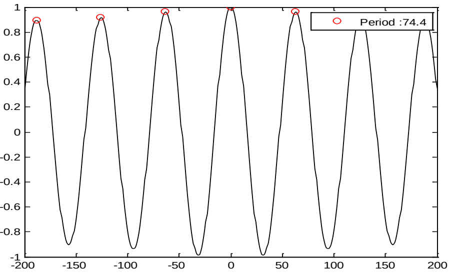
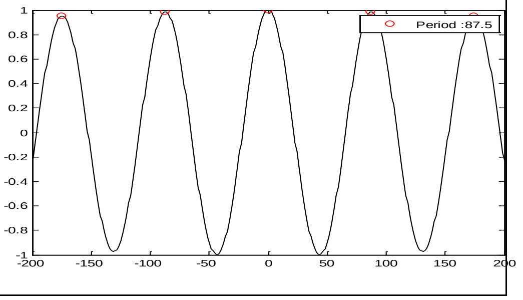
Loop	Approximated Period	Actual Period
1		
2		

Figure 5.2: Auto-Correlation versus Lag

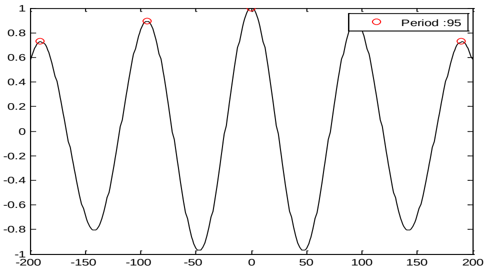
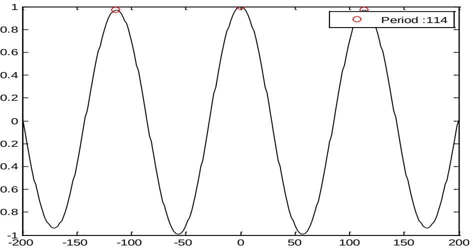
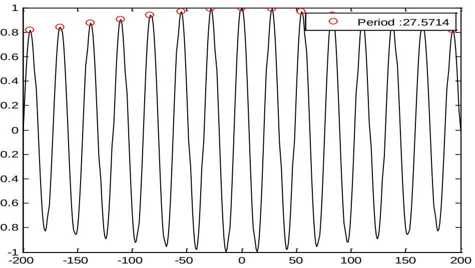
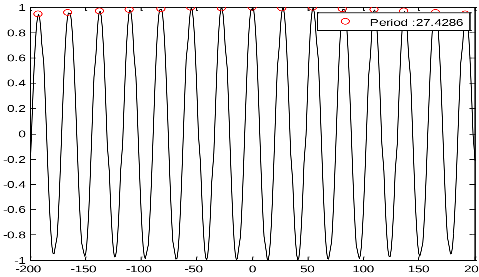
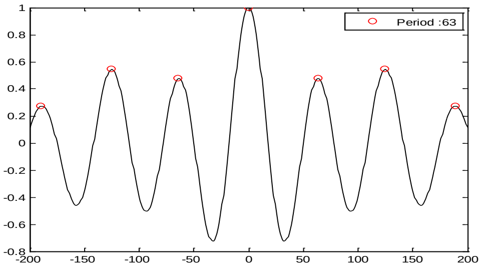
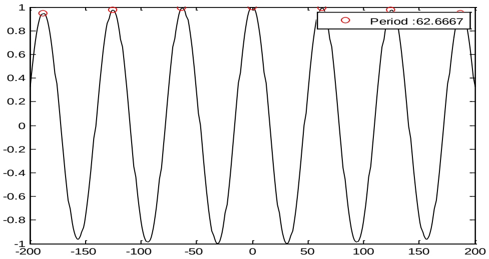
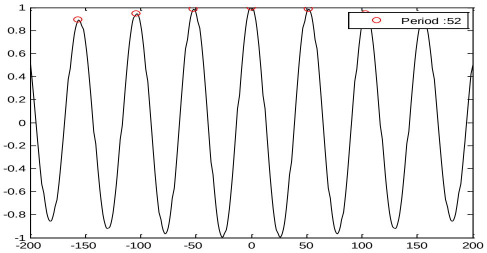
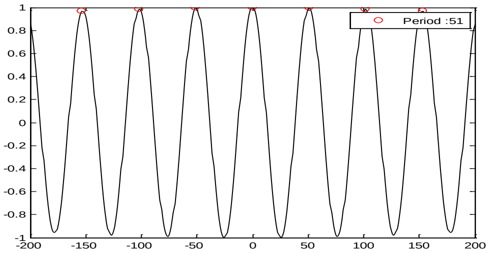
Loop	Approximated Period	Actual Period
3		
4		
5		
6		

Figure 5.2 Continued

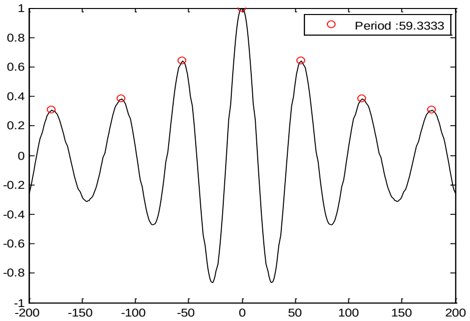
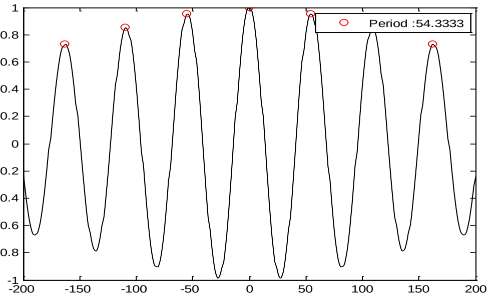
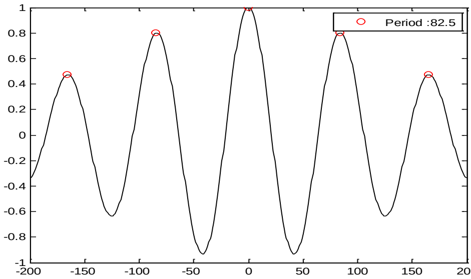
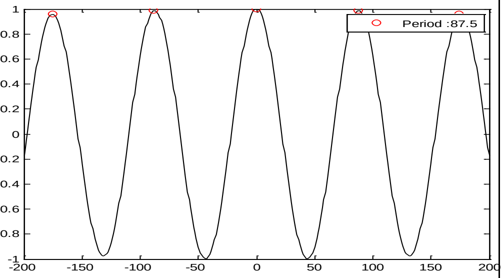
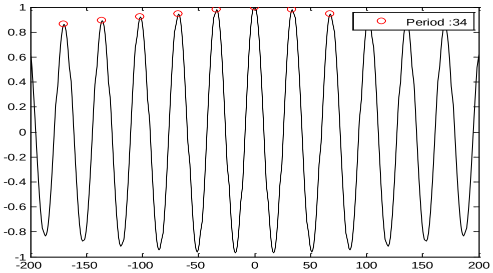
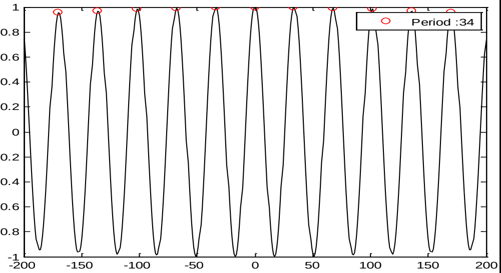
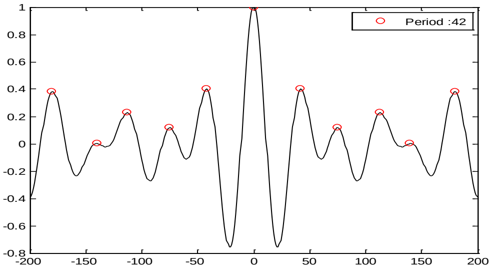
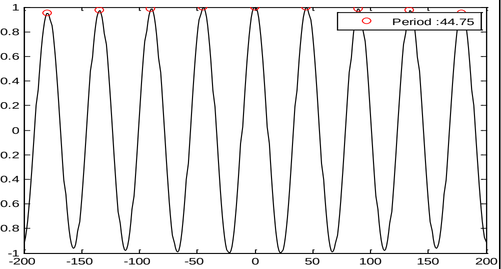
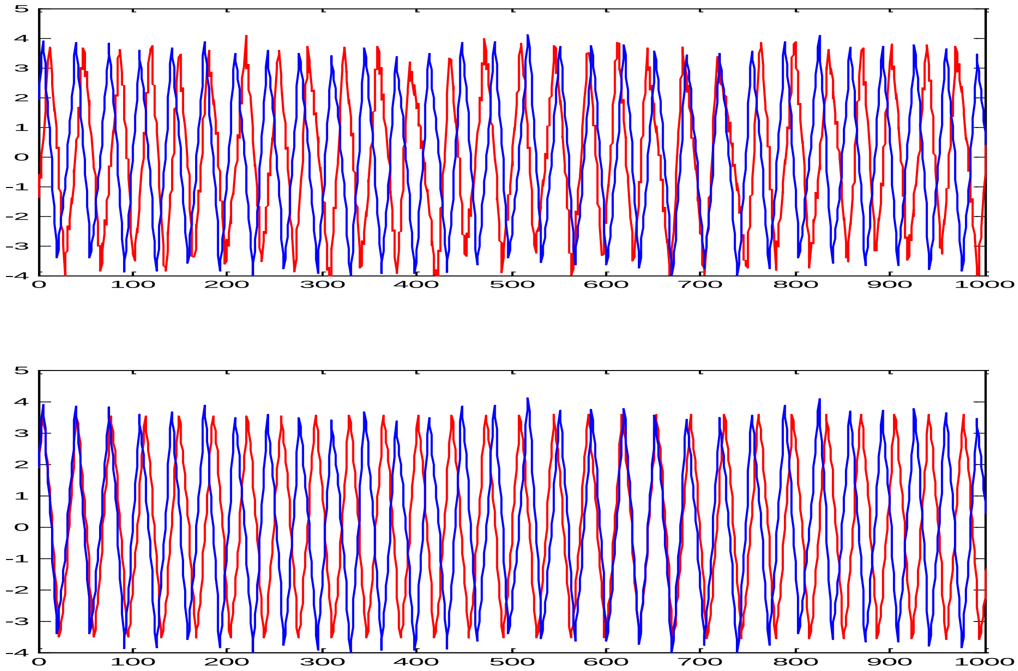
Loop	Approximated Period	Actual Period
7		
8		
9		
10		

Figure 5.2 Continued

Loop 1



Loop 2

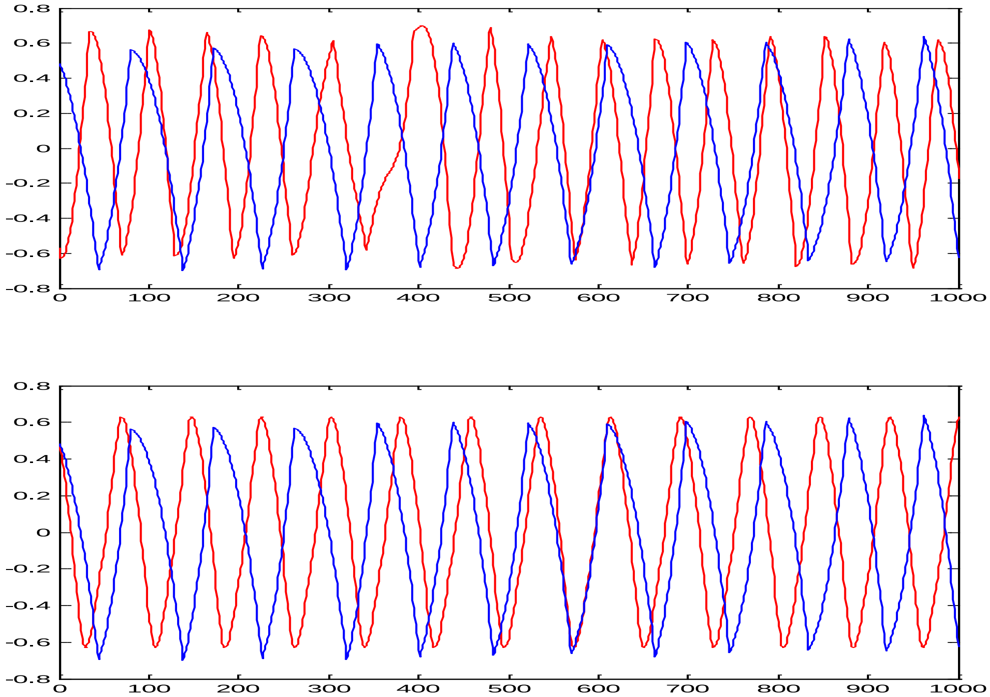
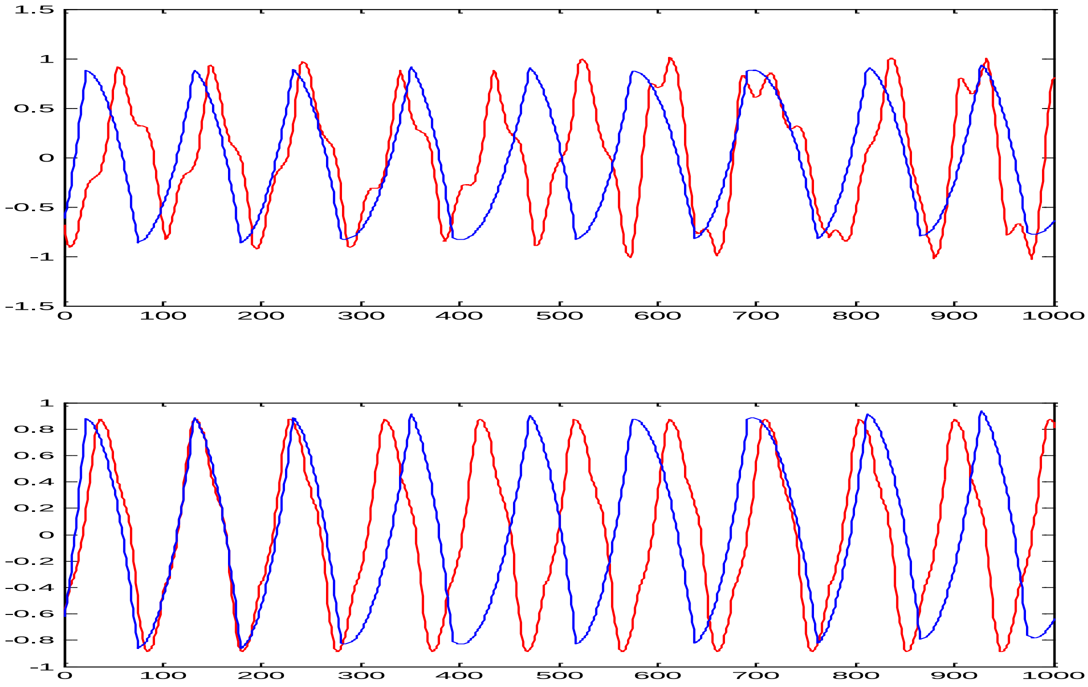


Figure 5.3: OP(t) patterns before and after noise removal and reconstruction: Red: Noisy Trend, Blue: Actual Trend

Loop 3



Loop 4

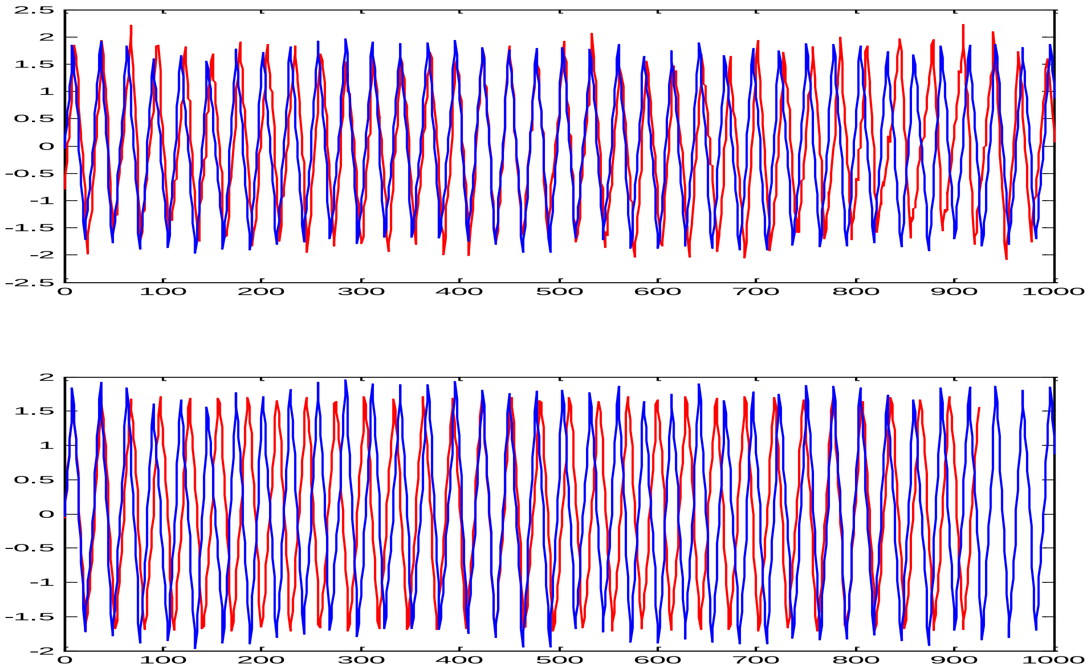
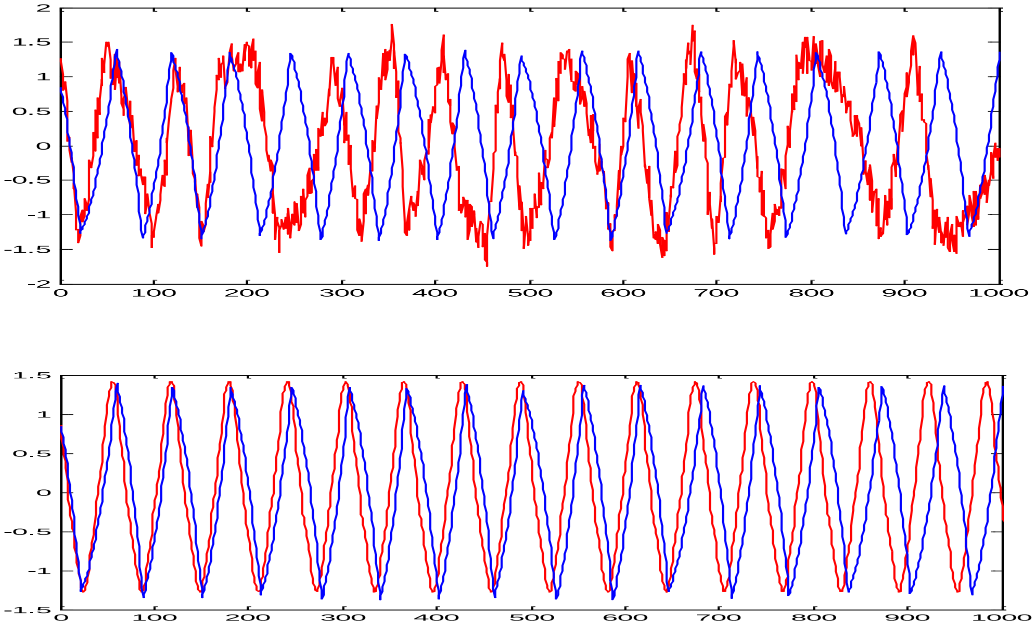


Figure 5.3 Continued

Loop 5



Loop 6

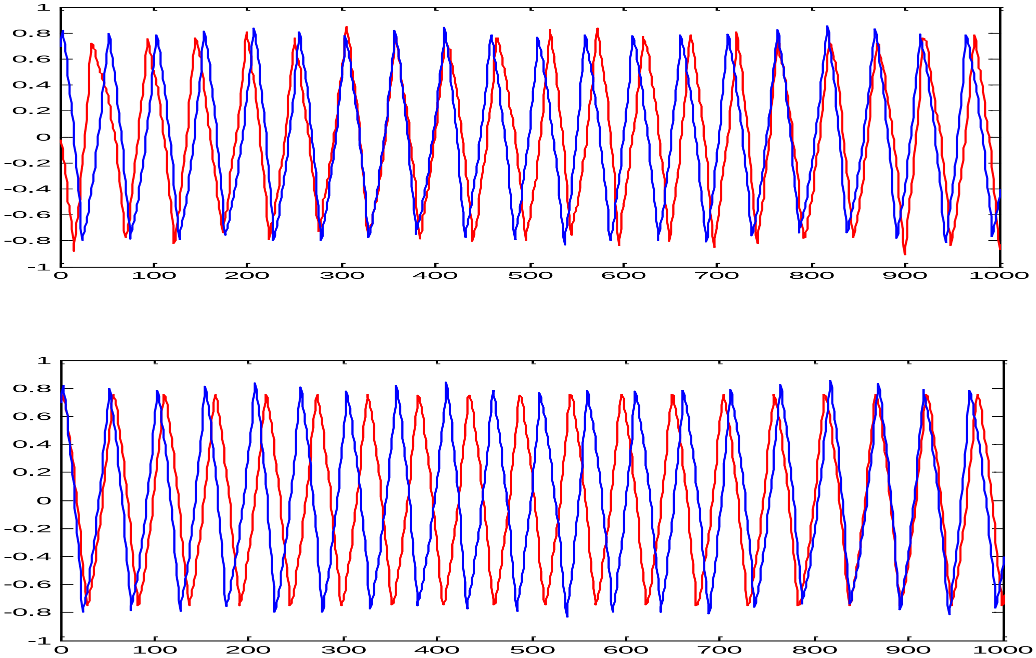
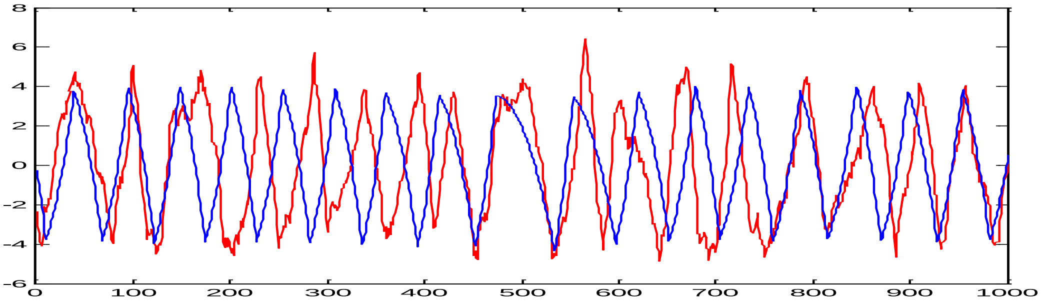
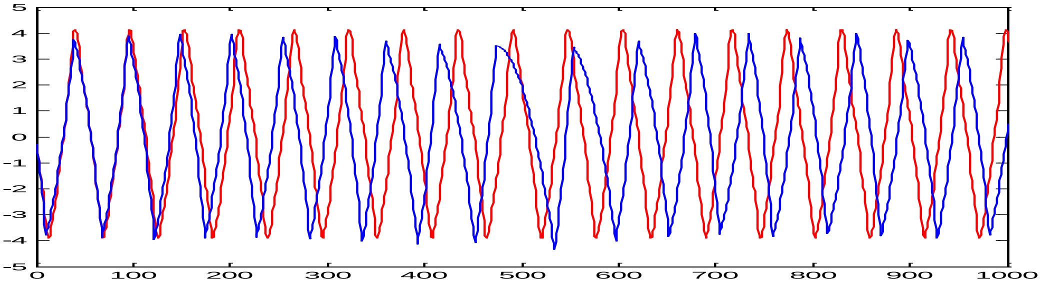


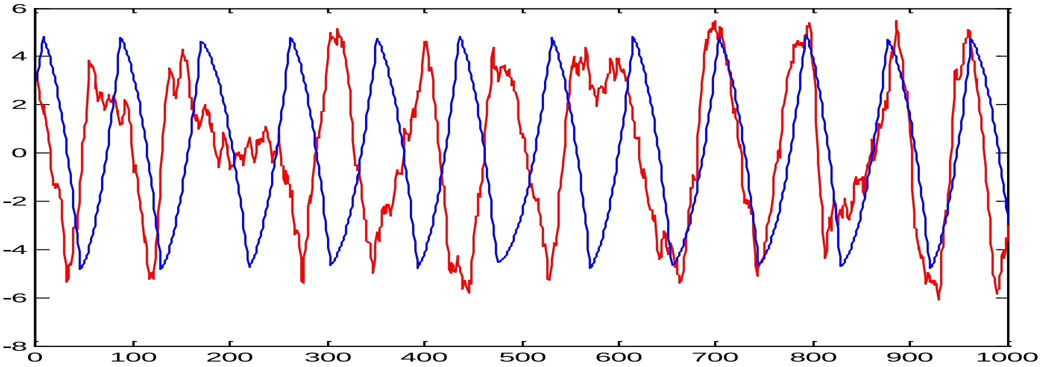
Figure 5.3 Continued

loop 7





Loop 8



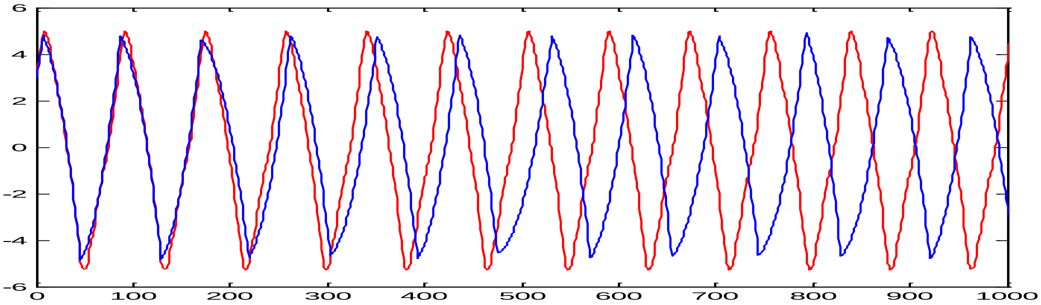
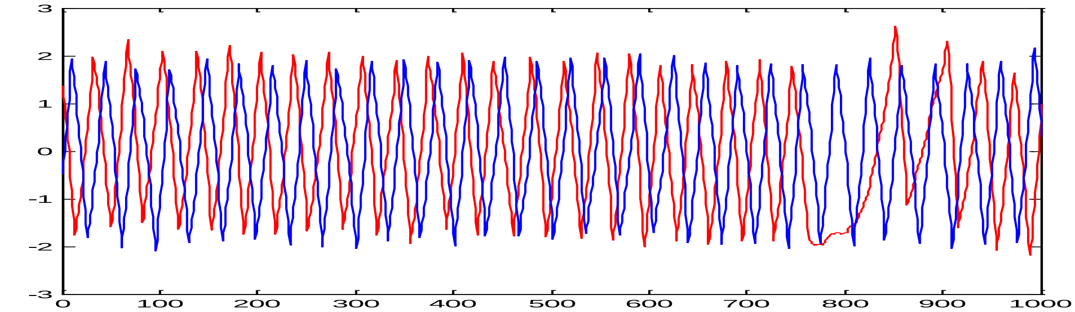
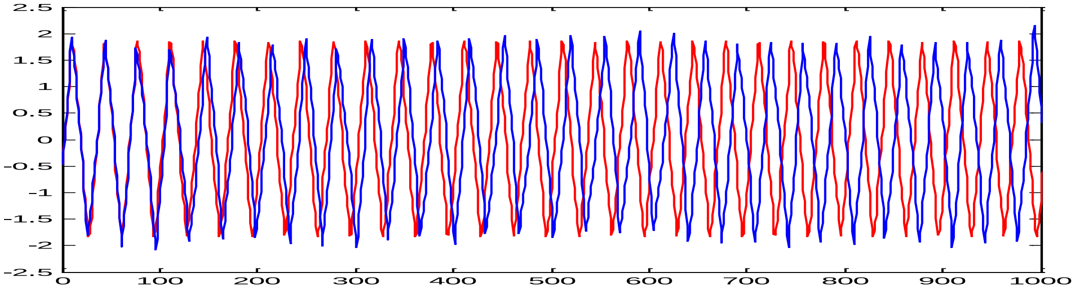


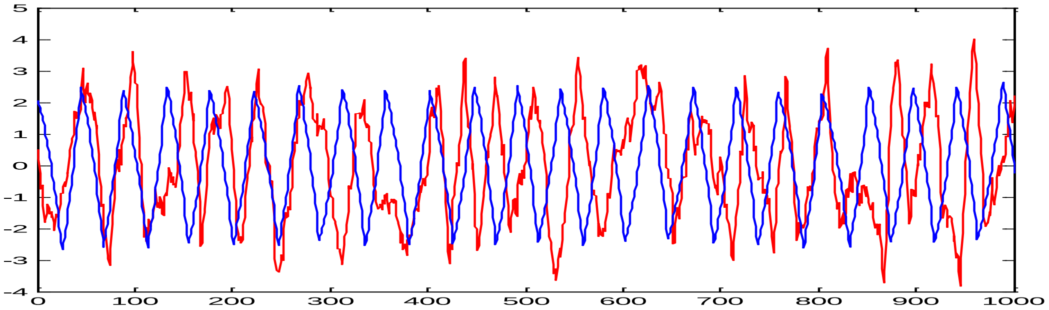
Figure 5.3 Continued

Loop 9





loop 10



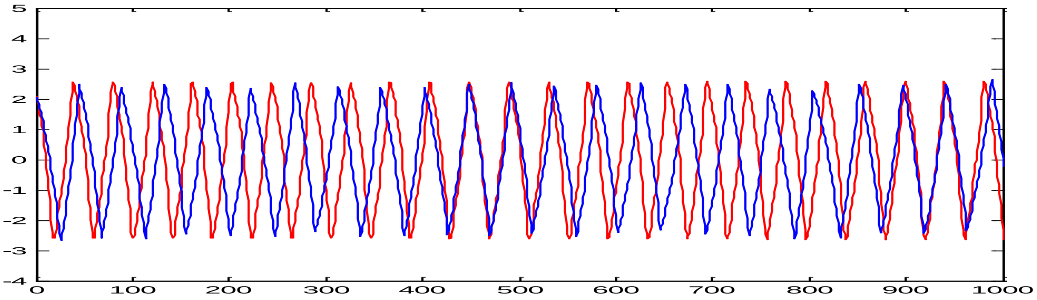


Figure 5.3 Continued

There was a clear improvement in appearance of the trend. Essentially, the goal was to extract the actual trend (shown in blue in Figure 5.3) from the noisy red patterns. Not only has the drifting nature of the original noisy trends (shown in red) been removed, but a fair estimation of the amplitude, frequency and shape of the patterns has been in achieved in most of the test cases.

It is important to note that since only a 100 time step interval from the trends will be used to predict fs and fd (that is, quantify stiction) the long term behavior of the trend does not affect quantification results. Using a 100 time step interval of a particular pattern is justified because only one cycle of a particular stiction pattern can completely characterize the oscillations : their shape, amplitude and frequency.

In other words, although there is a lag in the 1000 time-step reconstructed signals shown, it should not affect quantification results because only 100 time steps from each will be used.

5.3 Prediction of fs and fd using Feed-forward Neural Network

In Section 4.6, the neural network predictor, which estimates the degree of stiction (fs and fd) given a 100 time step section of an OP(t) pattern was developed. Details of training, validating and testing of feed-forward network were described. Figure 5.4 shows the performance of the network

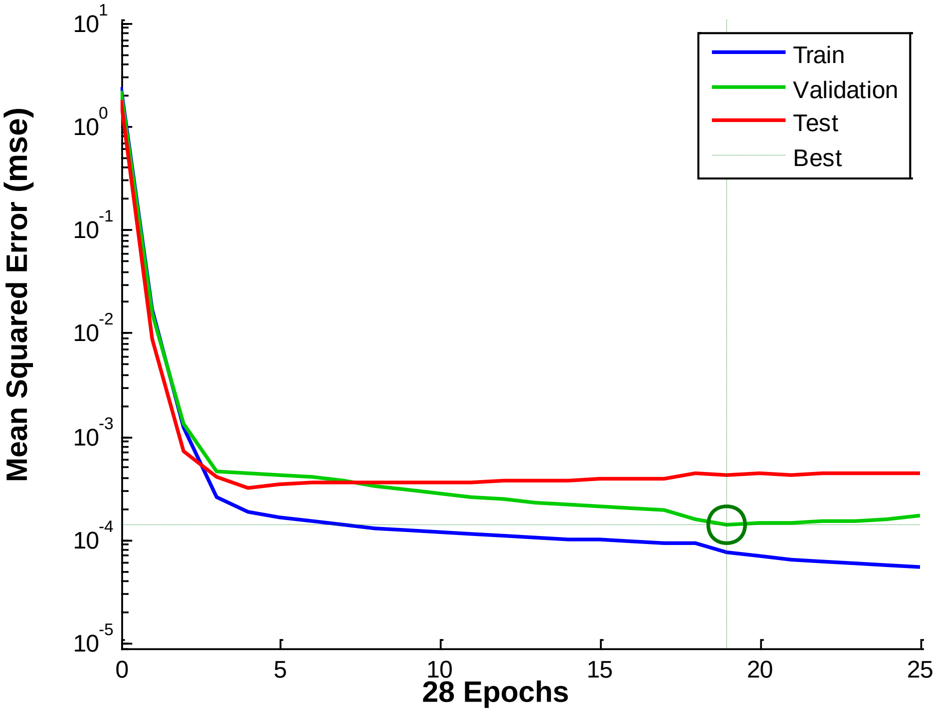


Figure 5.4: Performance of Neural Network Predictor

Each Epoch represents a particular iteration of the back-propagation learning algorithm (Section 3.2). As a reminder, the predictor model was developed by simulating 1275 degrees of stiction. 80% of this data The OP(t) and associated fs and fd values were used to train a feedforward network and 20% to validate that the model had properly learned the relationships and will be able to generalize. That is, given a new stiction pattern form the 20 test cases, it would be able to predict fs and fd.

The smooth convergence of the performance plots indicates that the model has not over-fitted the relationship between OP(t) and fs and fd. Training is stopped at the point circled when the network notices that the training error is decreasing but the validation error starts to increase, a clear sign of 'overfitting'.

The goodness of fit for the training, validation and testing sets are shown in Figures 5.5, 5.6 and 5.7, respectively.

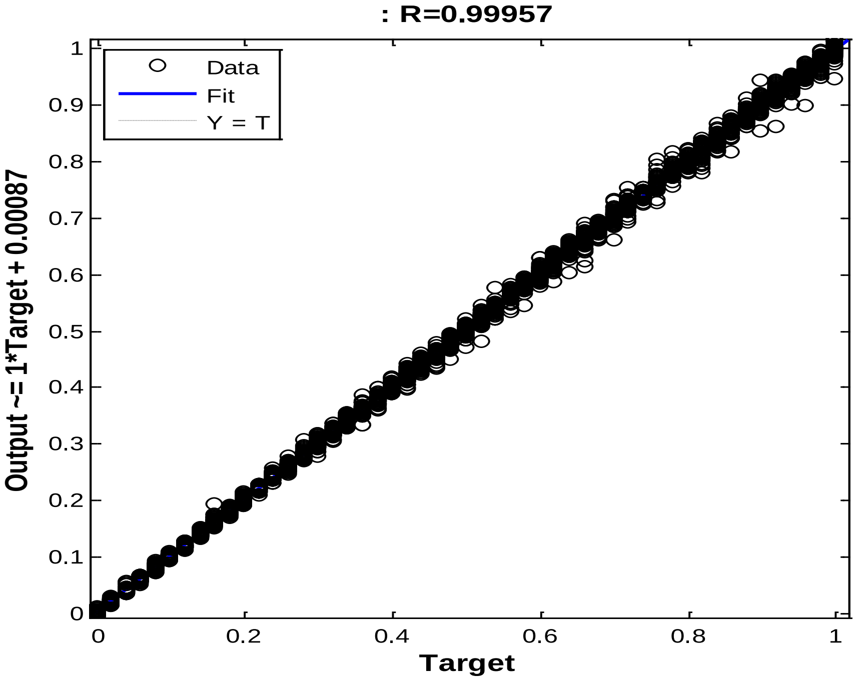


Figure 5.5: Regression Plot Showing relationship between actual and predicted fs and fd values for the training data set (80% of the 1275 stiction patterns)

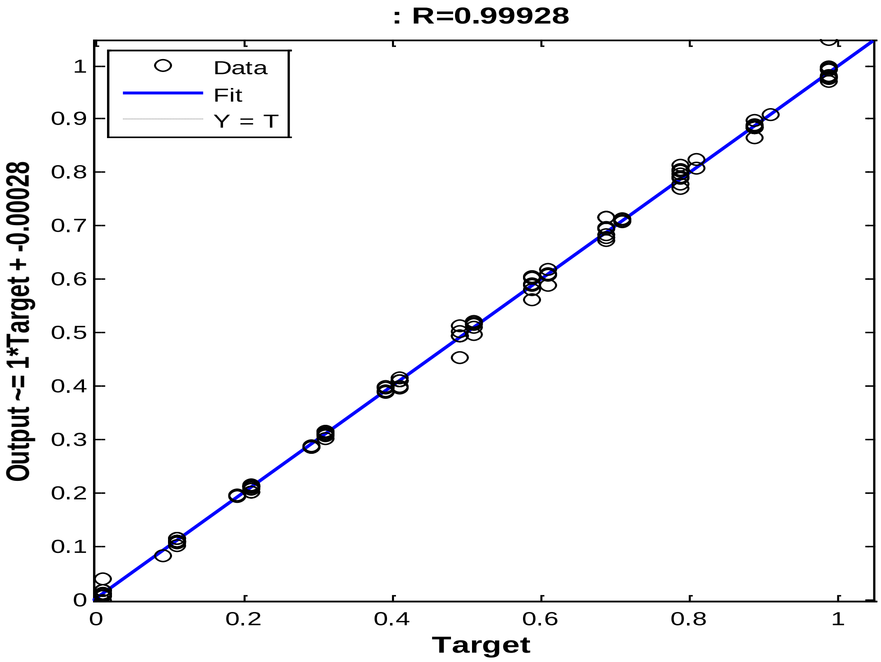


Figure 5.6: Regression Plot Showing relationship between actual and predicted fs and fd values for the validation data set (20% of the 1275 stiction patterns)

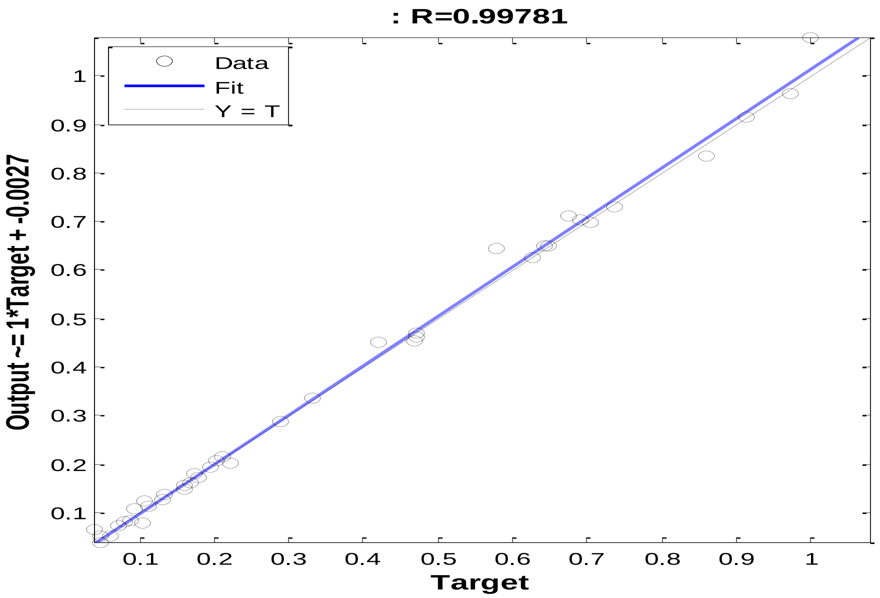


Figure 5.7: Regression Plot Showing accuracy of the developed model in predicting fs and fd for the 20 test cases

Tables 5.1 and 5.2 show the estimation error of the neural network model in prediction the stiction parameters.

Stiction Loop Number	Degree of Stiction (%)	Estimation Error $\mathrm{f_s}$(%)	Estimation Error $\mathrm{f_d}$(%)
1	$f_s = 98.24$ $f_d = 65.53$	*-1.1176*	*0.4864*
2	$f_s = 47.05$ $f_d = 5.40$	*-0.9546*	*0.4864*
3	$f_s = 74.37$ $f_d = 69.45$	*-0.7093*	*1.3517*
4	$f_s = 72.16$ $f_d = 21.19$	*-0.6616*	*0.4497*
5	$f_s = 61.34$ $f_d = 15.83$	*0.0147*	*-0.6926*
6	$f_s = 12.25$ $f_d = 4.03$	*1.7384*	*-1.0131*
7	$f_s = 16.52$ $f_d = 11.24$	*-0.3538*	*0.0389*
8	$f_s = 9.35$ $f_d = 6.27$	*1.4614*	*-0.7636*
9	$f_s = 32.59$ $f_d = 20.11$	*0.4697*	*0.0681*
10	$f_s = 64.86$ $f_d = 7.36$	*0.0825*	*0.1383*
Average Error	*0.6526%*		

Table 5.1: Estimation error for 10 test cases with no added noise

Stiction Loop Number	Degree of Stiction (%)	Estimation Error f_s(%)	Estimation Error f_d(%)
1	$f_s = 91.40$ $f_d = 20.40$	*0.1243*	*0.3617*
2	$f_s = 13.20$ $f_d = 0.09$	*0.4224*	*-0.4987*
3	$f_s = 17.40$ $f_d = 13.00$	*0.5712*	*0.3358*
4	$f_s = 46.80$ $f_d = 4.00$	*-1.4786*	*2.4653*
5	$f_s = 29.00$ $f_d = 16.00$	*-0.2514*	*-1.2963*
6	$f_s = 17.80$ $f_d = 8.00$	*0.6192*	*0.0120*
7	$f_s = 86.00$ $f_d = 42.00$	*-2.6176*	*2.9959*
8	$f_s = 100.00$ $f_d = 67.60$	*7.8813*	*3.5783*
9	$f_s = 47.20$ $f_d = 10.40$	*-0.2595*	*-2.5510*
10	$f_s = 57.80$ $f_d = 22.20$	*6.5867*	*-1.9883*
Average Error	*1.8448%*		

Table 5.2: Estimation error for 10 test cases with no added noise

6. CONCLUSION AND FUTURE WORK

Neural networks were shown to be very flexible models capable of extracting the underlying stiction trend from stiction control loops with added noise and accurately predicting the amount of stiction . The INLPCA model was effective at denoising the closed curve trend formed by plotting the controller output (OP) versus the process variable (PV). For the 10 stiction test cases with no noise, the prediction accuracy was 0.6526%, and for the 10 test cases with no added noise, the prediction accuracy achieved was 1.8448%. Estimating the degree of stiction present in a control loop can be a first step in developing appropriate automatic software compensation techniques through predictive control design coupled with installation of hardware such as valve positioners. If the amount of stiction present and its effect on the process variable can be modeled, stiction can treated as a measured disturbance in a process.

The stiction quantification technique also relies heavily on an accurate process model and valve stiction model, since the idea is to develop this predictor model offline not in real time.

Real systems involve multivariable or multiple-input multiple-output (MIMO) control, where control loops are interacting and not isolated from one another. This introduces complexity in the problem of estimating stiction in a control loop, although, the loops may be treated as if they are isolated from one another and fairly accurate stiction estimation results are still possible.

REFERENCES

[1] S. Karra, M.N. Karim, Comprehensive methodology for detection and diagnosis of oscillatory control loops, Control Engineering Practice, 17 (2009) 939-956.

[2] M. Jelali, B. Huang, Detection and diagnosis of stiction in control loops: state of the art and advanced methods, Springer, 2009.

[3] M. Shoukat Choudhury, N.F. Thornhill, S.L. Shah, Modelling valve stiction, Control engineering practice, 13 (2005) 641-658.

[4] R. Srinivasan, R. Rengaswamy, S. Narasimhan, R. Miller, Control loop performance assessment. 2. Hammerstein model approach for stiction diagnosis, Industrial & engineering chemistry research, 44 (2005) 6719-6728.

[5] K.H. Lee, Z. Ren, B. Huang, Stiction estimation using constrained optimisation and contour map, in: Detection and Diagnosis of Stiction in Control Loops, Springer, 2010, pp. 229-266.

[6] Q.P. He, J. Wang, M. Pottmann, S.J. Qin, A curve fitting method for detecting valve stiction in oscillating control loops, Industrial & engineering chemistry research, 46 (2007) 4549-4560.

[7] M. Jelali, Estimation of valve stiction in control loops using separable least-squares and global search algorithms, Journal of Process Control, 18 (2008) 632-642.

[8] Q.P. He, J. Wang, Quantification of valve stiction based on a semi-physical model, in: American Control Conference (ACC), 2013, IEEE, 2013, pp. 4362-4367.

[9] L.Z. Xiang Ivan, S. Lakshminarayanan, A new unified approach to valve stiction quantification and compensation, Industrial & Engineering Chemistry Research, 48 (2009) 3474-3483.

[10] N.F. Thornhill, A. Horch, Advances and new directions in plant-wide disturbance detection and diagnosis, Control Engineering Practice, 15 (2007) 1196-1206.

[11] M.A.A.S.C. Monir Ahammad, A Simple Harmonics Based Stiction Detection Method, in: Proceedings of the 9th International Symposium on Dynamics and Control of Process Systems (DYCOPS 2010), Leuvan, Belgium, 2010.

[12] K. Forsman, On detection and classification of valve stiction, in: Proc TAPPI conf process control, Williamsburg, USA, 2000.

[13] R. Srinivasan, R. Rengaswamy, R. Miller, Control loop performance assessment. 1. A qualitative approach for stiction diagnosis, Industrial & Engineering Chemistry Research, 44 (2005) 6708-6718.

[14] M. Kano, H. Maruta, H. Kugemoto, K. Shimizu, Practical model and detection algorithm for valve stiction, in: IFAC symposium on dynamics and control of process systems, 2004, pp. 5-7.

[15] S.B. Chitralekha, S.L. Shah, J. Prakash, Detection and quantification of valve stiction by the method of unknown input estimation, Journal of Process Control, 20 (2010) 206-216.

[16] V.N. Vapnik, An overview of statistical learning theory, Neural Networks, IEEE Transactions on, 10 (1999) 988-999.

[17] M. Kirby, R. Miranda, Circular nodes in neural networks, Neural Computation, 8 (1996) 390-402.

[18] M.A. Kramer, Autoassociative neural networks, Computers & chemical engineering, 16 (1992) 313-328.

[19] M.A. Kramer, Nonlinear principal component analysis using autoassociative neural networks, AIChE journal, 37 (1991) 233-243.

[20] M. Scholz, M. Fraunholz, J. Selbig, Nonlinear principal component analysis: neural network models and applications, in: Principal manifolds for data visualization and dimension reduction, Springer, 2008, pp. 44-67.

[21] W.W. Hsieh, Nonlinear principal component analysis of noisy data, Neural Networks, 20 (2007) 434-443.

[22] W.W. Hsieh, Nonlinear principal component analysis by neural networks, Tellus A, 53 (2001) 599-615.

[23] W.W. Hsieh, Nonlinear multivariate and time series analysis by neural network methods, Reviews of Geophysics, 42 (2004).

[24] K. Hamilton, W.W. Hsieh, Representation of the quasi-biennial oscillation in the tropical stratospheric wind by nonlinear principal component analysis, Journal of Geophysical Research: Atmospheres (1984–2012), 107 (2002) ACL 3-1-ACL 3-10.

[25] N.J. Kasdin, Discrete simulation of colored noise and stochastic processes and 1/f α power law noise generation, Proceedings of the IEEE, 83 (1995) 802-827.

[26] G.E. Box, G.M. Jenkins, G.C. Reinsel, Time series analysis: forecasting and control, John Wiley & Sons, 2013.