Use Authorization

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at Idaho State University, I agree that the Library shall make it freely available for inspection. I further state that permission to download and/or print my thesis for scholarly purposes may be granted by the Dean of the Graduate School, Dean of my academic division, or by the University Librarian. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Signature _____

Date _____

MODELING COMPRESSOR DYNAMICS USING SYSTEM IDENTIFICATION

By

Sujan Prasai

A thesis submitted in partial fulfillment of the

Requirements for the degree of

MASTER OF SCIENCE

IN

MECHANICAL ENGINEERING

IDAHO STATE UNIVERSITY

Spring 2015

To the Graduate Faculty:

The members of the committee appointed to examine the thesis of SUJAN PRASAI find it satisfactory and recommend that it be accepted.

_____

Dr. Marco Schoen, Major Advisor

_____

Dr. Kenneth Bosworth, Co-Advisor

_____

Dr. Gene Stuffle, GFR

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my special thanks and gratitude to my major advisor, Dr. Marco P. Schoen for his continued support, guidance, and patience throughout this research and writing this thesis. This research would not have been possible without his expertise and motivation.

I would like to thank my colleague Dane Sterbentz for his help and co-operation throughout this research. I would also like to thank Mary Hofle, Tom Walters, Dr. Jichao Li, Dr. Feng Lin, and Dr. Kenneth Bosworth for their help in various aspects of this research.

Last, but not least, I would like to thanks my beautiful wife for her patience and continued support in my studies and this research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This thesis deals with data analysis and dynamic modeling of an axial compressor system. The data processing includes filtering of the time domain data using a Bayesian filter, low pass Butterworth filter, and a notch filter. The output of each filter is analyzed and the purposed Bayesian filter is found to be the most suitable filter for this research. A few techniques used in estimating missing data are also discussed and an algorithm is developed in Matlab™ to estimate missing data. This algorithm is based on an autoregressive model with exogenous input having missing data. In most part of this thesis, the linear and nonlinear techniques of modeling a one - stage compressor system is discussed. The modeling is done using System Identification Toolbox in Matlab™. The output error model and Non - Linear Hammerstein - Wiener (NLHW) models are used to capture the slow dynamics of the compressor. The NLHW model is used to model the fast dynamics of the compressor.

CHAPTER 1: INTRODUCTION

Axial flow compressors are primarily designed to create a pressure rise across a flow moving parallel to the axis of rotation of the rotating compressor blades. In an axial compressor, the rotating blades or rotor blades are used to thrust the flow forward in order to increase the fluid velocity. Between each row of rotor blades is a row of stationary blades known as stator blades. Stator blades spread the fluid flowing through the compressor and consequently compress the fluid. The combination of one rotor and one stator blade rows is known as a compressor stage. The rotor and stator blade rows function together as a compressor stage in order to create the rise in the fluid pressure. Most types of axial compressors have multiple stages in which each stage produces a successive pressure rise for the flow. Axial flow compressors are currently used for a number of industrial, aerospace, and research applications, [1].

The schematics of a one stage compressor system is shown in Figure 1. Air gets sucked in though the inlet and is served to the compressor stage. Air is then compressed by the compressor stage and is sent into the plenum guided by the exit duct. A throttle is used to vary the operating point of the compressor and it is done by changing the size of the opening of the outlet to the atmosphere.

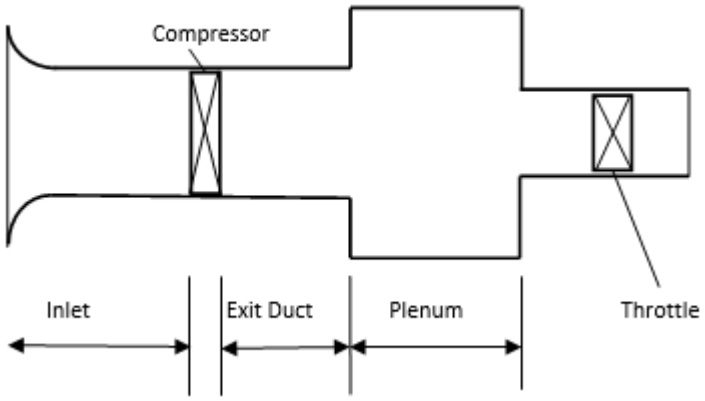

Figure 1: One - Stage Compressor Schematics.

1.1: STALL AND SURGE

Compression systems used in gas turbine engines and industrial processes undergo severe aerodynamic instabilities. During the normal operation of such a system, instabilities cannot be endured because they cause a large amount of mechanical load on the structure, [2]. Compression system instabilities are generally caused by either rotating stall or surge. The type and magnitude of these instabilities are dependent on the dynamics of the compressor, [2-4]. Rotating stall is a phenomenon which causes disturbances in the circumferential flow pattern of the compressor and is usually considered as a two or three dimensional phenomenon, [3, 4]. The rotating stall phenomenon is caused by a flow moving slower than the rotors (generally 30 percent to 70 percent of the rotor speed) around the compressor annulus, [3-7]. Surge is a large amplitude oscillation which causes an overall reversal of the flow in the compressor, [2-7]. The surge phenomenon is dependent on compressor geometry as well as the dynamic properties of the system such as inlet and outlet channels, volume, and throttle resistance, [2]. Surge can also be described by fluctuations in flow, rise in pressure, and the rotational speed of the compressor, [4].

In the design of axial compressors and other types of turbomachinery, it is of primary importance to consider the prevention of rotating stall and surge. Rotating stall causes severe uneven loading on the compressor blades. This uneven loading can then cause extreme blade vibration, decrease in pressure rise, decrease in compressor efficiency, overheat the burner, and sometimes surge, [3, 7, 8]. Surge is a highly undesirable phenomenon due to the severe damage it causes to the compressor as well as the flow system. Damage to a jet engine caused by compressor surge includes overstress on the compressor blades and a lack of air provided to the jet engine combustor which may lead to flame-out, [3, 4, 9]. The traditional way to avoid surge is to run compressors at an operating level away from the surge line, [4]. However, this limits the operating range and achievable efficiency of the machine, [4]. For obtaining extreme pressure ratios in every stage of the compressor, aircraft engines these days oftentimes

use transonic axial flow compressors. Engine weight and size can be reduced if high pressure ratios can be obtained from each stage of the compressor, which in turn reduces the cost of operation and investment, [10].

As mentioned earlier, it is necessary to operate compressor at a safe region away from the peak performance of the system. This operation avoids the flow perturbation while the system is running. The characteristics of the one – staged compressor is shown in Figure 2. In the Figure 2, the flow coefficient ϕ versus the pressure rise Ψ curve, also known as performance curve is depicted. The entire cycle of the compressor operation is shown at constant rotor speed. The cycle starts off with higher flow coefficients and the pressure gradually rises up to certain point. At the same time the flow coefficient gradually decreases to certain point, it is the point of the optimum performance of the compressor. The pressure rise beyond this point drives compressor to stall and surge.

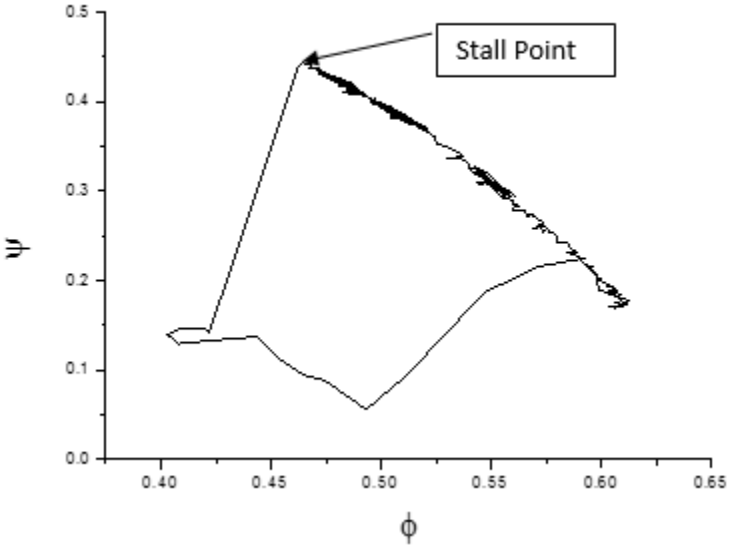

Figure 2: Characteristics of One - Stage Compressor.

A number control schemes have been purposed to mitigate flow instabilities and increase the efficiency of the system without affecting the safety of the system. The purposed control methods are dynamic schemes. These schemes can be designed and applied only if one is familiar with the dynamics of the compressor. In these actuation

schemes, the operating point of the compressor is driven up along the performance curve. The performance increase of the compressor is depicted on the characteristic plot (as shown in Figure 3) of the system with controls applied, [3].

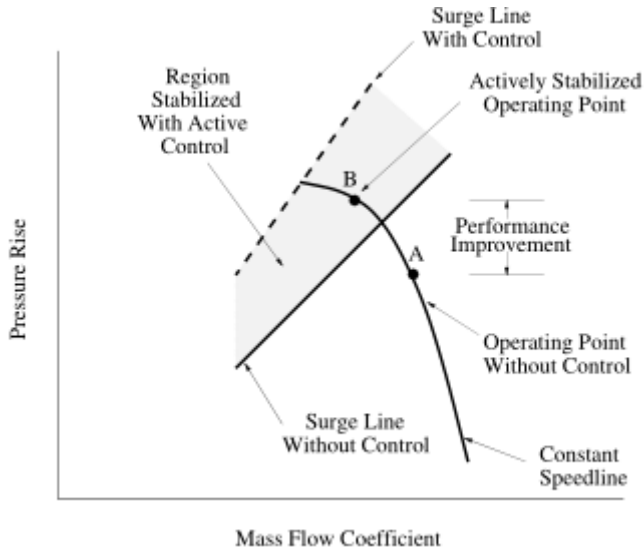

Figure 3: Characteristics of a Compressor with Controls, [3].

As it can be seen in Figure 3, the operating point of the compressor is away from the surge line without controls. Once the control is implemented, the surge line is moved up along the performance curve along with the operating point. Therefore it can be seen that optimization and control of the flow in compressor can result in enhancements of its performance. This improvement results are also reflected in big economic and environmental improvement. A small improvement can results in large cost savings for the operation of compressors. As mentioned in [11], one compressor stage was reduced using active control in the jet engine which resulted in 1.5 % reduction in fuel consumption and 5 % increase in thrust – to – weight ratio. This saving is just for one engine, if the number of airplanes in the sky at a certain time is considered, this could result in huge amount of savings. The air pollution caused by airplanes reduces as the fuel consumption decreases. This gives motivation to understand the dynamics of the compressor and possibly design and implement controls.

1.2 HISTORY OF RESEARCH AND MODELING TECHNIQUES

The research on flow inside the compressor started as early as the 1950's. Since axial flow compressors were used on jet engines, it became a necessity to understand the stall and surge phenomenon in order to make jet engine safe and efficient. Emmons [12] presented a blade – passage level theory. This theory explained how the stall inception of stall occurs in the compressor. He theorized that the flow separation inside the blade passage was culprit to the instability in compressor. In 1970 at University of Tokyo, Saburo Nagano and Hiroyuki Takata [13] established a model to explain rotating stall. They took on finite difference method and applied it on blade row(s) to clarify the features of rotating stall. Greitzer [8] in 1970's developed a system - level model to define surge phenomenon. This model doesn't incorporate any flow information within the blade passages of the compressor. Moreover, he also collected experimental results at Massachusetts Institute of Technology to verify his theoretical model. Moore and Greitzer [14] together in 1980's integrated blade passage flow into Greitzer's model and developed a new model. The resulting model was a highly conceptual 1-D model that can capture both rotating stall and surge in compressors. This Moore – Greitzer model was a huge success, a new engine type "smart engine" concept was purposed. New engine type concept and Moore – Greitzer model initiated a wave of research on active control of compressors. Mansoux et al. [15] in 1994 used nonlinear method of the Moore – Grietzer model to observe the nonlinearities responsible for the initiation of rotating stall. An experiment was conducted using three different compressors and the results were compared to the simulation results. The important features were similar in all comparisons but stall initiation manners were found to be different.

A study done by Mc Dougall et al. [16] (1990) and Day [17] (1993b) in low speed compressors found two different forms of stall initiation. They are generally known as spike and Modal oscillation. Origination of Spike is regarded as a disruption of short length – scale, [18]. The first appearance of spike in the velocity traces indicates small stall cell in circumference which spreads its size around the perimeter quickly, [18]. Another type of stall initiation form is modal oscillation, it is an interference that is formed gradually in

long length – scale, [18]. It appears as small waves in the velocity traces which gradually initiates rotating stall, [18]. Among two type of stall initiation forms, modal initiation is supposed to be well understood however, spike initiation is not quite understood, [19]. It is required to understand flow behavior inside the blade passage to understand the spike initiation whereas it is not necessary for modal initiation, [19].

In 1976 Greitzer [8] developed a model to describe transitory behavior of compressors, it showed the relation between a Helmholtz oscillator and surge oscillation. Greitzer model did not describe the occurrence of acoustic waves in the system. Helvoirt and Jager [20] showed experimentally the occurrence of acoustic waves and has significant influence on the nature of surge oscillation in a compressor. Yoon et al. [21] in 2011 presented similar work modeling compressor plenum as a pipe. They developed a mathematical model to describe the pipeline dynamics during the stable operation of the compressor. They confirmed their mathematical model with data while Helvoirt and Jager [20] failed to do so. Recently in 2013 Joseph Pismenny et al. [22] used the method of spectrograms to correctly understand pressure fluctuations, mistakenly identified as a beating effect in axial compressor during rotating stall. This method showed that the minor change in rotating stall frequency and the beats were absent during the process.

Among the various ways to investigate the flow inside the compressor, numerical simulation is one of the method which is limited due to the instability in occurrence, [23]. Hwang and Kang [23] in 2010 used three dimensional and unsteady numerical simulation in the first stage with inlet guide vanes of a low speed axial flow compressor near the stall line.

Venturini [24] established a non-linear mathematical model to capture the dynamics of the compressor. His approach is based on physics of phenomenon occurring inside the compressor. He used laws of conservation and heat balances to develop the model. To confirm the good quality of the calibration process influence of parameters such as “volume, friction factor, and heat transfer coefficient” on the model were analyzed. A similar method was used by Morini et al. [25] in 2006 to model the

compressor transient characteristics. This model also used lumped parameter approach to account for rotating mass and this model is able to duplicate the system characteristics in occurrence of surge.

1.3 FAST DYNAMICS

The dynamics within the blade passages for different flow conditions is referred to as fast dynamics. As mentioned earlier in this Chapter, stall occurs on a compressor in two different modes. The blade geometry of the compressor and the angles associated with the incoming flow (rotor tip incidence angle) determine the type stall. The two modes of stall inception are modal stall inception and spike stall inception. The slope of the characteristic curve is zero for modal stall inception while it is not necessary for spike stall inception, [18]. In Figure 4 [18], modal and spike stall inception are shown on the compressor characteristics plot.

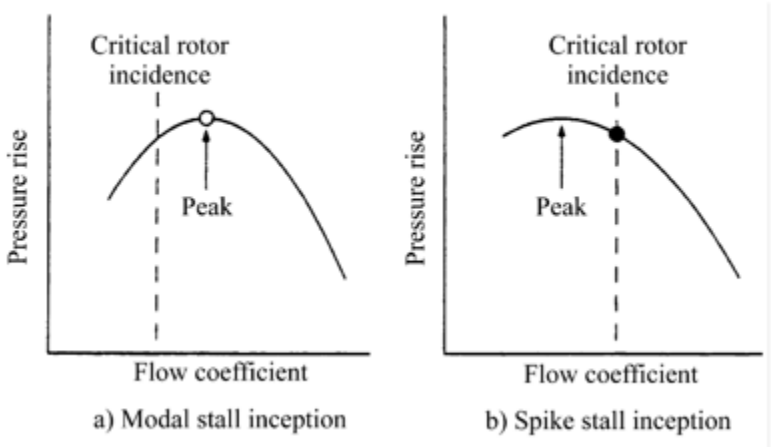

Figure 4: Modal and Spike stall inception shown on compressor characteristic curve, [18].

This section will focus on spike stall inception phenomenon because the compressor located at CAS follows spike stall inception. Spike stall inception is a precursor to fully developed rotating stall and it has been used to implement controls on compressors, [18]. However, spike inception is measured only one to two rotations before

the stall occurs. Therefore, this precursor is not viable for control because controls system requires some time to react, as it needs time to detect, compute, and activate accordingly to the situation. In Figure 5 [18], spike stall inception is shown. In the plots of Figure 5, a spike suddenly appears and after one or two rotations more spikes are seen. Following the spikes compressor suddenly goes to stall after one or two rotations.

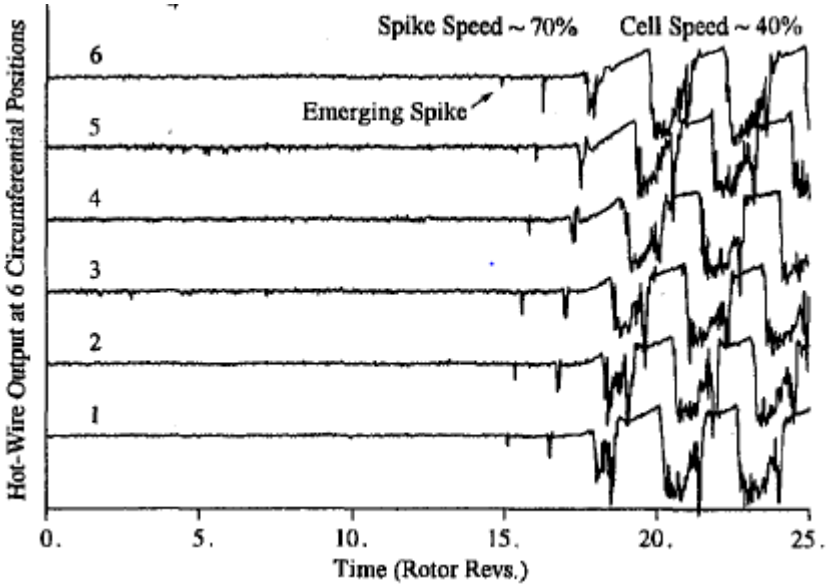

Figure 5: Spike Stall Inception, [18].

The spike inception is followed by an unsteady flow pattern within the blade passage, [26]. Vo et al. [26] depicted that the interface between the incoming flow and the tip clearance flow can be used to characterize the flow within the blade passage. As shown in Figure 6 [26], an interface line is drawn graphically to show where the incoming and the tip clearance flow meet.

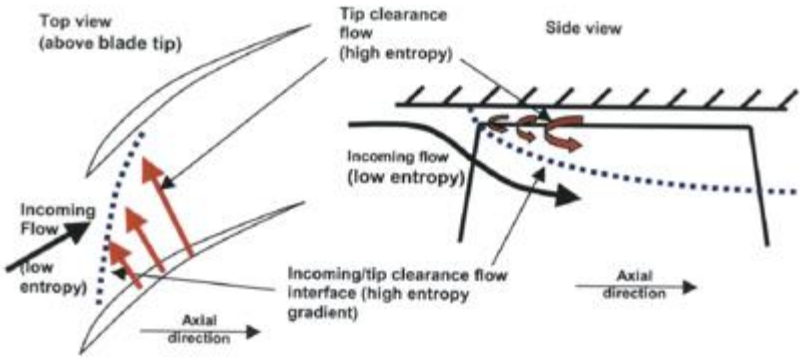

Figure 6: Interface between incoming flow and tip leakage flow, [26].

The various flow stages or conditions within the blade passage can be distinguished using the interface. This hypothesis describes the flow development within blade passage from stable to unsteady, to stall inception, and lastly to unstable flow condition. This hypothesis has been accepted by number of research teams however, a team in Japan has a different hypothesis. Their hypothesis indicates that the vortex growth behind the interface line is the cause of the problem, [27].

The interface line can be used to identify the condition of the flow. The interface lies between the blade passages, it starts from the low pressure side of the first blade and extends into the blade passage area and falls short in reaching the high pressure side of the second blade. This creates a gap between the interface line and the high pressure side of the blade. This gap allows the incoming flow to pass through the blade passage without being blocked or disturbed. This process is depicted in the Figure 7.

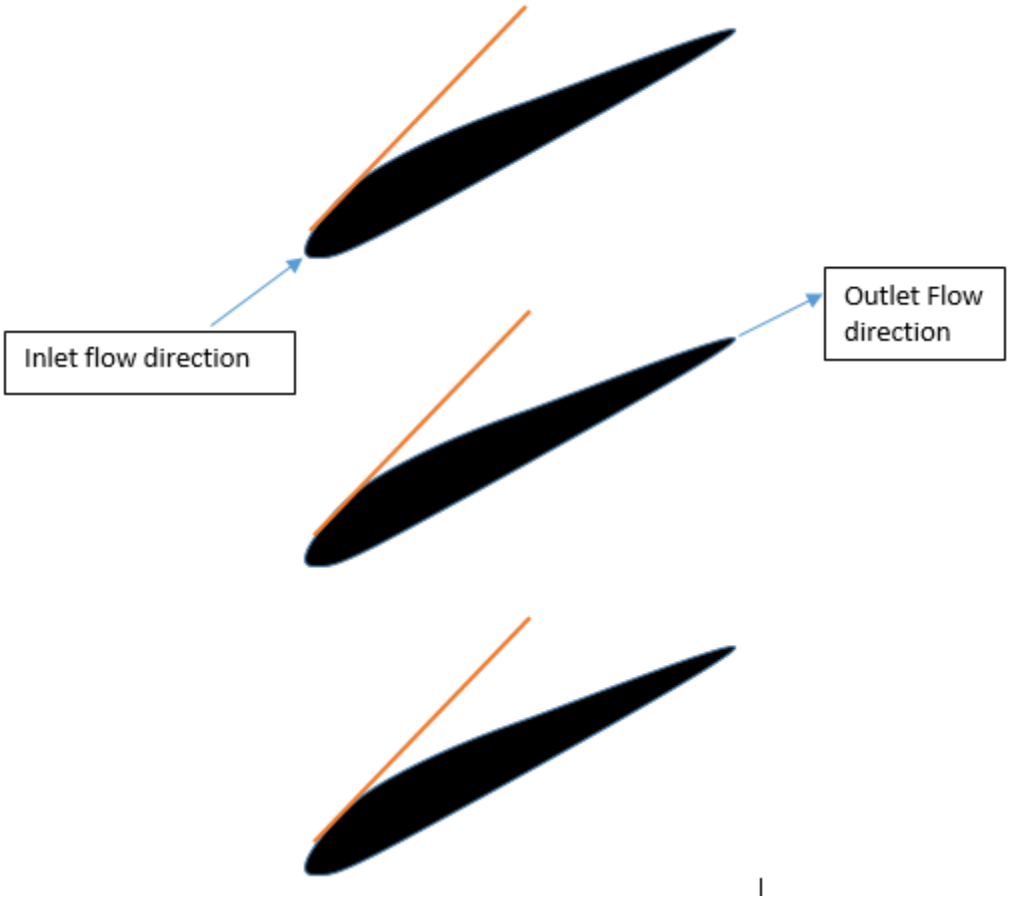

Figure 7: Interface line depicting stable flow within blade passage.

The flow becomes or is considered unsteady once the interface line meets the high pressure side of the second blade. The interface line can oscillate on its angle and move forward towards the leading edge as well as backward. The growth of the interface causes incoming flow blockage. This behavior of the interface line is shown in Figure 8.

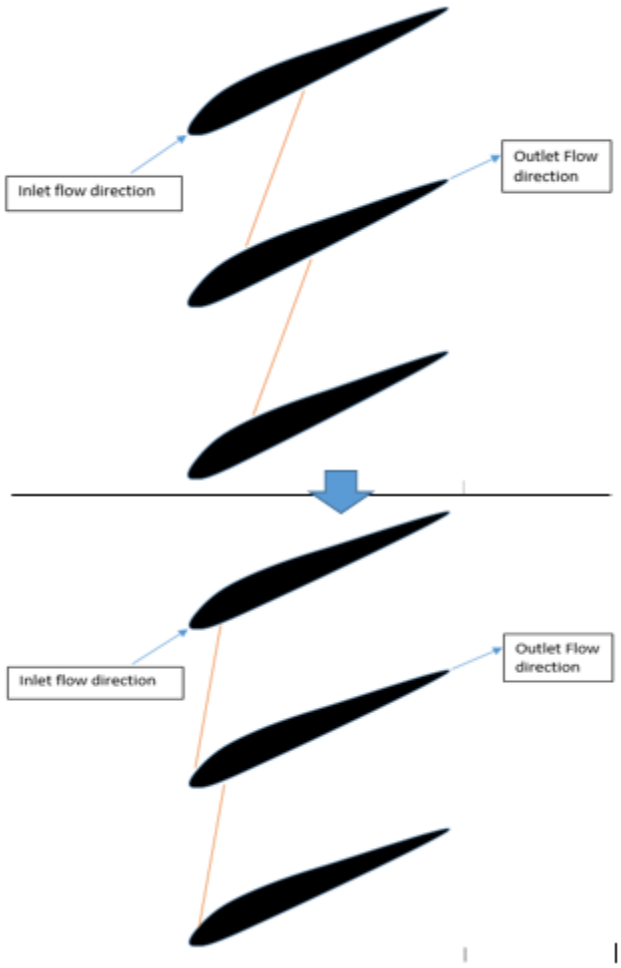

Figure 8: Interface line depicting unsteady flow within blade passage.

As the interface line reaches the leading edge, the incoming flow can escape and this causes the spike inception of the rotating stall, [26]. In Figure 9 [26], the unsteady flow behavior is depicted (a) and the spike inception of the rotating stall is depicted (b), where LE means leading edge. As it can be seen in Figure 9 (a) because of the interface line blockage there is back flow of fluid and leakage of the fluid through the tip. In Figure 9 (b) the flow is spilling through the leading edge as the interface line has travelled out of the leading edge and the incoming flow is blocked.

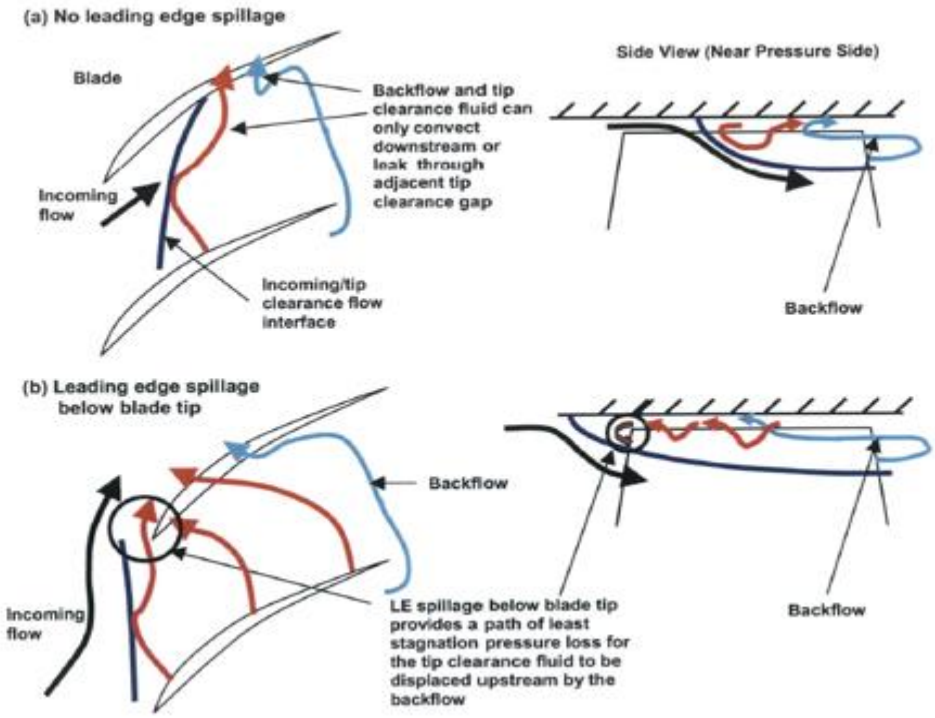

Figure 9: (a) Unsteady flow and (b) Spike stall inception of rotating stall, [26].

1.4 TIP - CLEARANCE

Tip clearance is the gap between the compressor blade tip and the casing of the compressor. As mentioned earlier, flow leaks through this gap. This leakage can induce the vortices on the suction side of the blades which in turn may disturb the incoming flow. Langston [28] identified the two parameters of tip leakage flow. One is the magnitude of the tip clearance gap and the other one is the pressure difference between pressure and suction side of the blade. One study found 5% decrease in efficiency just by increasing the gap by 1%. The performance of the compressor is highly affected by the tip leakage flow. Storer and Cumpsty [29] related the efficiency loss and flow instabilities within the compressor to the tip clearance and the resulting tip leakage. They developed a method to estimate efficiency loss due to tip leakage based on experimental data and numerical solutions. The relationship between tip clearance flow and the three - dimensional flow separation on the suction side of the blade was studied by Gbadebo et al. [30]. They

concluded that the interaction between the clearance flow and end wall boundary layer influences the separation.

It is evident that the instabilities in the flow through a compressor drives the compressor to rotating stall and surge. To overcome this problem Tahara et al. [31] developed a simple technique to reduce the stall margin of an axial compressor. They purposed a way to compute correlation coefficients using the pressure measurements for the current and one previous rotations for each blade pitch. The correlation coefficient was used to relay a warning signal to allow corrective actions to act and prevent stall. They found out the correlation coefficient degraded as the flow was decreased in the axial direction. The significant degradation of the correlation was seen on the mid chord of the blade and as the stall was approached the degradation shifted gradually towards the leading edge. It was also distinguished from the experimental data that the flow fluctuation started on the pressure side of the blade and the cause of the tip leakage may be related to degradation of correlation coefficient. Experiments were conducted focusing on the circumferential direction of the compressor under uniform inlet flow and the same flow throttling, the time scale and origin placement. This experiment found that the degradation first appeared at the largest tip clearance location and spread in both circumferential direction. Therefore it was expected that the larger tip gaps causes larger leakage flow which may in turn degrade the correlation coefficient because of flow disturbance caused by tip leakage, [31].

Some other experiments are done by other researchers to find out the effects of the unsteadiness of the flow at the tip clearance on stall inception. Tong et al. [27] experimented with as the compressor is throttled from fully open condition. At the open condition tip leakage vortices flow easily from the blade passages. As throttling increases the smooth flow remains until a critical value is reached and then the vortices starts to move across the blade passage and invade on the pressure side of the neighboring blade. The compressor is stable until the tip leakage vortices reaches the leading edge of the blade. As the vortex reaches the leading edge, there is flow reduction which induces spike. This originates at the largest tip clearance, [27].

1.5 MODELING OF THE COMPRESSOR

The overall system dynamics of a compressor is described by slow and fast dynamics combined. Therefore, the system dynamics can be represented by a matrix composed of four distinct submatrices. This allows easy distinction between different flow conditions. Different flow conditions include stable flow, unsteady flow, pre – unstable flow where spike - stall inception occurs, and fully unstable flow. The elements of the matrix represent the fast and slow dynamics and the relationship between the disturbances. This matrix can describe the stable operation of the compressor, unsteady flow within blade passage, and linkage between slow and fast dynamics. The matrix can be written in the form as shown in Equation 1, [32].

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \tag{1}$$

In Equation 1, A_{11} represents the slow dynamics, A_{22} represents the fast dynamics, and A_{12} including A_{21} represent relation between the slow and the fast dynamics and the disturbance. This matrix define the unsteady flow condition. The vectors inside the A – matrix contains five variables to define fast dynamics, slow dynamics, and the disturbances. The two variables of the vectors are Ψ and ϕ representing non – dimensional pressure and flow coefficients, respectively. $\widehat{\Psi}$ represents the pressure disturbance coefficients. The other two variables are θ and ω represents the angle of the incoming flow streams with respect to compressor blade and the frequency of the incoming flow streams oscillations, respectively. These two variables are representatives of the fast dynamics of the compressor. The vectors of the A – matrix for unsteady flow can be written as follows:

$$A_{11} = \begin{Bmatrix} \Psi(t) \\ \phi(t) \\ \widehat{\Psi}(t) \\ \theta(t) \\ \omega(t) \end{Bmatrix} \quad A_{12} = \begin{Bmatrix} \Psi(\theta,t) \\ \phi(\theta,t) \\ \widehat{\Psi}(\theta,t) \\ \theta(t) \\ 0 \end{Bmatrix} \quad A_{22} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ \theta(t) \\ \omega(t) \end{Bmatrix} \tag{2}$$

The A – matrix for pre – unstable and unstable contains all the vectors however, the variables of the vectors must be determined.

For the stable flow condition the A - matrix can be written as follows:

$$\begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix} \tag{3}$$

where, A_{12} and A_{21} are zero because the flow is stable and no disturbances are present.

CHAPTER 2: THEORY ON FILTERING

In this Chapter, a few ways of filtration techniques are discussed. The filters are used to reduce noise influence in the data extracted during compressor operation when the compressor approaches stall.

2.1 BAYESIAN FILTERING

Bayesian filtering is named after the English mathematician Thomas Bayes. A theory of probability interference was developed by this mathematician which was used for development of the Bayesian filter. It is also known as a Bayes filter. It is a probabilistic method for estimating an unknown probability density function recursively over time. It uses online measurements and a mathematical process model for estimation, [33, 34].

A Bayesian filter inspects a data set which is identified to be unrelated (noise) to the measurement. It also observes a data set that is identified to be genuine and compares it and filters out the unrelated data (noise). These filters adapt to identify new patterns of noise. An added advantage of this filter is that it takes the entire data set into consideration before filtering out anything out of it [33-35]. In what follows, the method to prepare experimental data ($OriginalData$) and filter theory are presented. The relationship between the filtered data and the original data is given by a conditional probability density function, [36].

$$P(OriginalData|x) \tag{4}$$

where,

x = Average rate of events for the system.

The number of events *n* is modeled as a Poisson process [36] as shown below:

$$P(OriginalData|x) \approx (x^n e^{-x})/n! \tag{5}$$

The data obtained from the experiment is rectified for the filtration purpose. This data is also defined as amplitude modulated zero – mean Gaussian noise, [36]. The expression for the rectified data is given by Equation 6.

$$P(OrignalData|x) = 2 \times \frac{\exp\left(-\frac{(OriginalData)^2}{2x^2}\right)}{(2\pi x^2)^{1/2}} \quad (6)$$

The density function of the original data set can be approximated using a Laplacian density, [36]. Mathematically it can be represented for rectified data as in Equation 7.

$$P(OriginalData|x) = \exp\left(-\frac{OriginalData}{x}\right)/x \quad (7)$$

Assuming the rectified data be represented by $OriginalData(t)$ at a given t, where t is the discrete time index. The probability of each likely value of $x(t)$ can be specified by the function $P[OriginalData(t)|x(t)]$ by using Bayes rule for posterior density [36] as shown in Equation 8.

$$P[x(t)|OriginalData(t)] = P[OriginalData(t)|x(t)] \times P[x(t)|OriginalData(t)] \quad (8)$$

where,

$P[x(t)] =$ Probability density for $x(t)$ instantly before the measurement of $OriginalData(t)$.

Preceding $P[x(t)]$ depends entirely on the past measurements and it can be estimated using Bayes' rule [36] on recursive algorithm based on discrete time measurements as shown in Equation 9.

$$P[x(t)|OD(t), OD(t-1), \dots] = P[OD(t)|x(t)]P[x(t)|OD(t-1), OD(t-2)..]/C \quad (9)$$

where,

$OD = OriginalData$

The recursive algorithm to estimate the probability density function is derived by and can be used for the filtration of the recorded data, [36]. The mathematical form of this relation can be written as:

$$P(x, t-1) \approx \alpha P(x-\varepsilon), t-1) + (1-2\alpha)P(x, t-1) + \alpha P(x+\varepsilon, t-1) + \beta + (1-\beta)P(x, t-1) \quad (10)$$

where,

$\alpha =$ Expected rate of gradual drift.

$\beta =$ Expected rate of sudden shifts in data.

This filter can be implemented in Matlab™ based on the theory described above. Matlab™ implementation of this filter is done by modeling the measured data as a random process with a concentration in exponential group and desired signal rate, [35]. The filtered data is modeled as a collective diffusion and jump process. The signal rate is estimated by utilizing all past measurements to calculate the full conditional density. The filtered data, outcome of Bayesian estimation best describes the measured data. It is a nonlinear filter and it considerably decreases noise from the signal, [35]. The implementation of Bayesian filter on Matlab™ is included in Appendix A1, where 'alpha' is diffusion drift and 'beta' is the Poisson jump has to be chosen empirically. These parameters determine the amount of filtering done.

2.2 LOW PASS BUTTERWORTH FILTER

Classification of filters are done based on the function they accomplish. Range of frequencies are used to classify filters. A filter is called a low pass filter if it has the property of transmitting low frequencies up to a specified limit and blocking the higher frequencies above that limit up to infinity, [37]. A typical ideal low pass filter characteristic is shown in the Figure 10, [38].

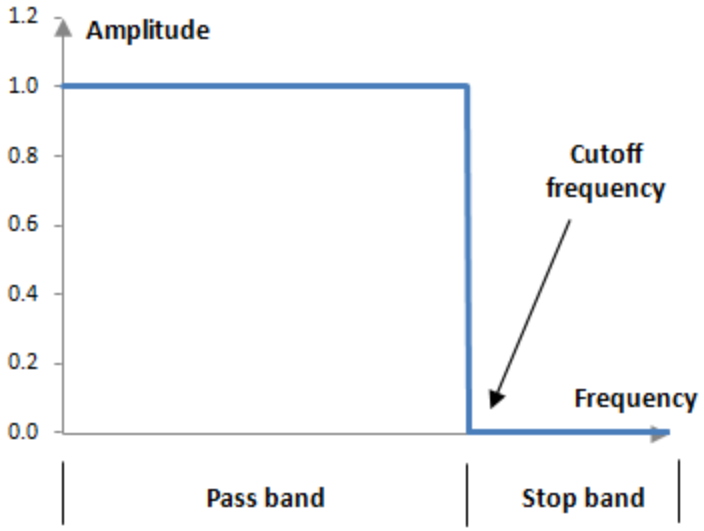

Figure 10: Ideal Low Pass Filter, [38].

However, a filter cannot behave ideally. The typical response of the low pass filter is shown in Figure 11, [38].

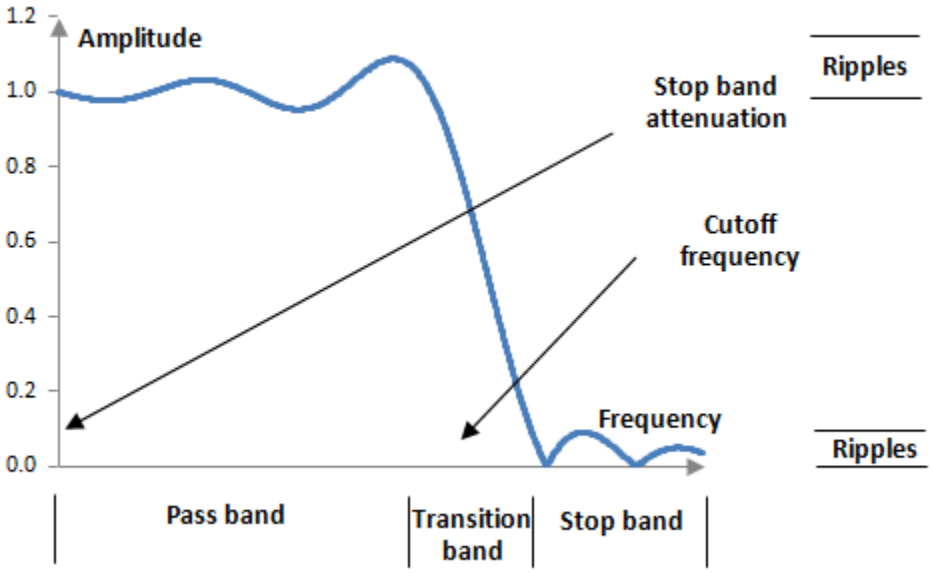

Figure 11: Typical Behavior of Low Pass Filter, [38].

The Butterworth filter was first presented by the British engineer and physicist Stephen Butterworth in his paper “On the Theory of Filter Amplifiers.”, [39]. This filter is mostly used in signal processing and the result of this filter has a flat frequency response

in a specified range of frequency. It is also called a "maximally flat magnitude filter." Butterworth filter are identified with two parameters, the filter order and the cutoff frequency. The filtered output is dictated by filter order, higher filter order must be chosen if higher rate of filtering is necessary, [39]. This filter can be easily programmed in Matlab™, in fact Matlab™ has a built-in command 'butter'. The syntax for this filter to be implemented in Matlab™ is shown in Equation 11.

$$[b\ a] = butter(n, wn, '\ low') \tag{11}$$

where,

$[b\ a]$ = Transfer function coefficients (b = numerator, a = denominator)

n = filter order

wn = cutoff frequency

low = tells Matlab™ that it is a low pass filter.

Once the transfer function coefficients $[b\ a]$ are determined, the signal can be easily filtered using the Matlab™ command below:

$$y = filter(b, a, x) \tag{12}$$

where,

y = filtered signal

x = input signal

This filter is programmed in Matlab™ and is included in Appendix A2.

2.3 NOTCH FILTER

A notch filter is a type of band - stop filter with a very slim stop band. These filters discard a portion of the frequency spectrum and passes all other wavelengths, [40, 41].

These filters can be designed by a slight variation on the band - pass filters. Band pass filters have the pole and zero relatively equidistant logarithmically from the unit circle. A slight modification to this i.e. moving that zero closer to or on the circle makes it a notch filter, [41]. A typical notch filter is shown in the Figure 12, [42].

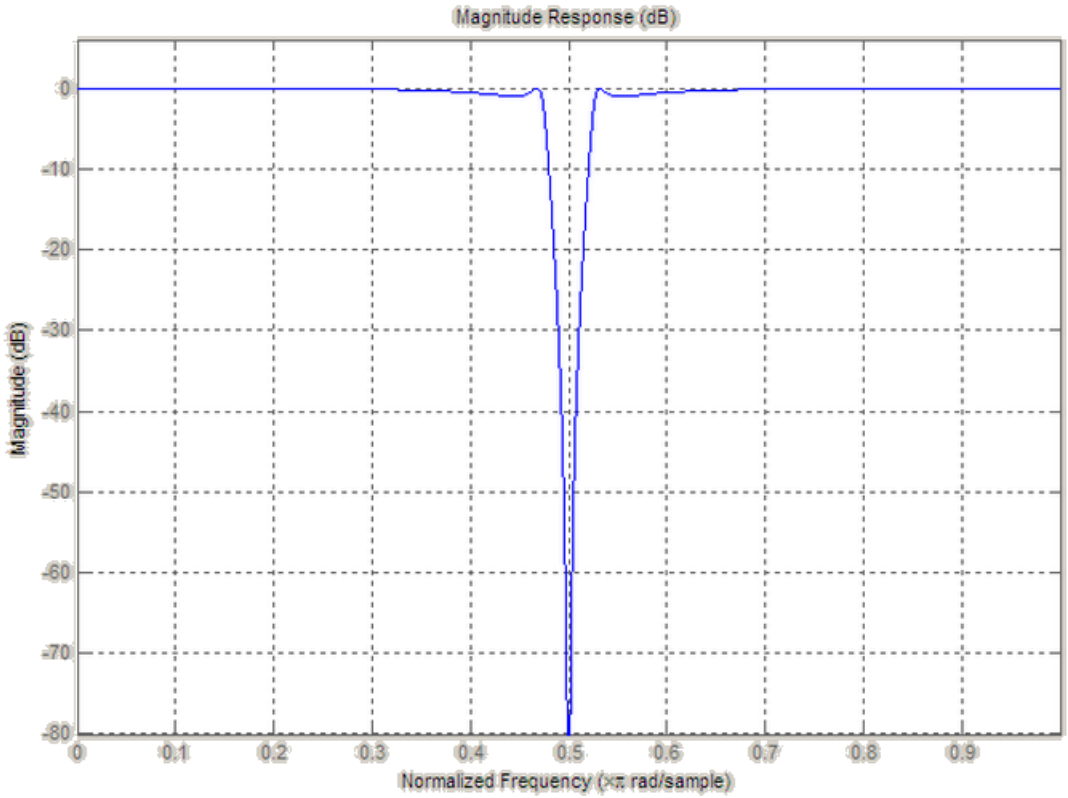

Figure 12: Notch Filter, [42].

A notch filter can be easily implemented in Matlab™. The syntax for implementing this filter in Matlab™ is given by Equation 13.

$$d = fdesign.notch('N, F0, Q', value\ N, value\ F0, value\ Q, Fs) \qquad (13)$$

The command above creates a notch filter of specification'd'.

where,

N = Filter order (should be even)

$F0$ = Center Frequency

Q = Quality Factor

$value\ N$ = Value of N

$value\ F0$ = Value of center frequency

$value\ Q$ = Quality factor value

Fs = Sampling frequency

The command below designs the filter H from the specification in d.

$$H = design(d) \tag{14}$$

Once the above steps are completed, following command is used in Matlab™ to filter out data.

$$y = filter(H, x) \tag{15}$$

where,

y = filtered data

x = input data

This filter is programmed in Matlab™ and is included in Appendix A3.

CHAPTER 3: SYSTEM IDENTIFICATION THEORY

An object that yields evident signals because of the interaction of different variables at various time and space scales is considered to be a system. System Identification (SI) is a process of determining the mathematical model of a system using the measured input and output of the system, [43]. The application of SI can be used in any kind of systems where input and output data can be measured. This process can be easily done in Matlab™. The SI toolbox in Matlab™ contains Matlab™ functions, Simulink blocks, and a Graphical User Interface (GUI) application for predicting mathematical models. A dynamic system can be modeled mathematically using SI toolbox in Matlab™ based on measured input – output data. It can be done by adjusting model parameters until it agrees best with the measured output. A test can be conducted to compare model output and the measured output by using a data set that was not used for model estimation. The SI toolbox allows use of linear and non –linear model for estimation. It can also be used to preprocess the measured data, [43-45].

The main features of the SI toolbox includes continuous and discrete – time transfer functions, state – space models, and process model estimation utilizing frequency and time domain measured data. The SI toolbox also utilizes maximum likelihood, prediction – error minimization, and subspace system identification techniques to identify Autoregressive (ARX), Box-Jenkins, and Output – Error models. It can also be used to identify nonlinear ARX models and Hammerstein – Wiener models. These nonlinear models contain input – output nonlinearities such as piecewise polynomial, 1D polynomial, and dead zone, [43-45].

The general procedure involved in estimating a model of a dynamical system includes the input – output data, the model structure, and the identification method. SI is basically a cycle because it involves selecting various model structures until it best describes the system, [43, 44]. The basic steps in the cycle of SI are laid out as follows:

1. Collect input – output data by designing an experiment to be identified using SI, [43, 44].

2. Process the collected data. Discard the un-necessary portion of data, filter data if noisy, remove trends and outliers, etc., [43, 44].
3. Select a relevant model structure to be used in describing the original system, [43, 44].
4. Describe the model as required, [43, 44].
5. Obtain the best model fit to the original system by inspecting the fit percent and the resulting plot of measured output and estimated output, [43, 44].
6. Inspect if the properties of the identified model resembles the original system, [43, 44].
7. If a good result is obtained, then discontinue. If not, go back to step three or try a different estimation model. If necessary, process input – output data differently or redesign experiment and collect another set of data, [43, 44].

The theory and procedure to estimate output – error model and non-linear Hammerstein – Wiener model using the SI toolbox are explained in what follows.

3.1 OUTPUT ERROR MODEL

The Output Error (OE) model is based on the difference between the measured output and the model's simulated output. The OE model is a variant of an ARX model and its structure is given by, [43]:

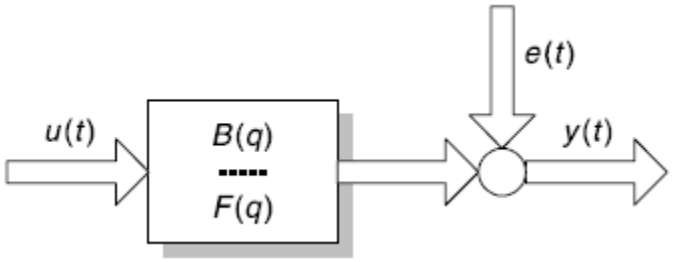

Figure 13: Output - Error Model Structure, [43].

$$y(t) = \frac{B(q)}{F(q)}\, u(t - nk) + e(t) \tag{16}$$

where,

$y(t)$ = output

$u(t)$ = input

nk = input delay

$e(t)$ = error

$B(q)$ & $F(q)$ = polynomial with shift operator 'q'.

An OE polynomial model can be estimated using Matlab™ function [44] as shown in Equation 17.

$$sys = oe(data, [nb\ nf\ nk]) \tag{17}$$

where,

sys = Estimated OE model

$data$ = 'iddata' object containing the input and output signal values.

nb = B polynomial order plus 1

nf = F polynomial order

nk = Input delay

This model can be obtained using the SI GUI in Matlab™. This GUI or application can be accessed by entering 'ident' command in Matlab™ command window. This application is capable of doing all the functions of SI such as import and employ input – output experimental data, estimate linear and non-linear models, inspect identified models, etc, [46].

In what follows, it is shown how to obtain OE model using SI application of the given experimental data.

As mentioned earlier, SI toolbox application can be accessed by entering 'ident' on workspace. The data that is to be imported to SI toolbox should be loaded by using 'load data' command on Matlab™ workspace or other methods of loading data may be used too. Once the data is loaded, one can import data to the SI toolbox using 'import data' drop down menu in the SI dialogue box as shown in Figure 14.

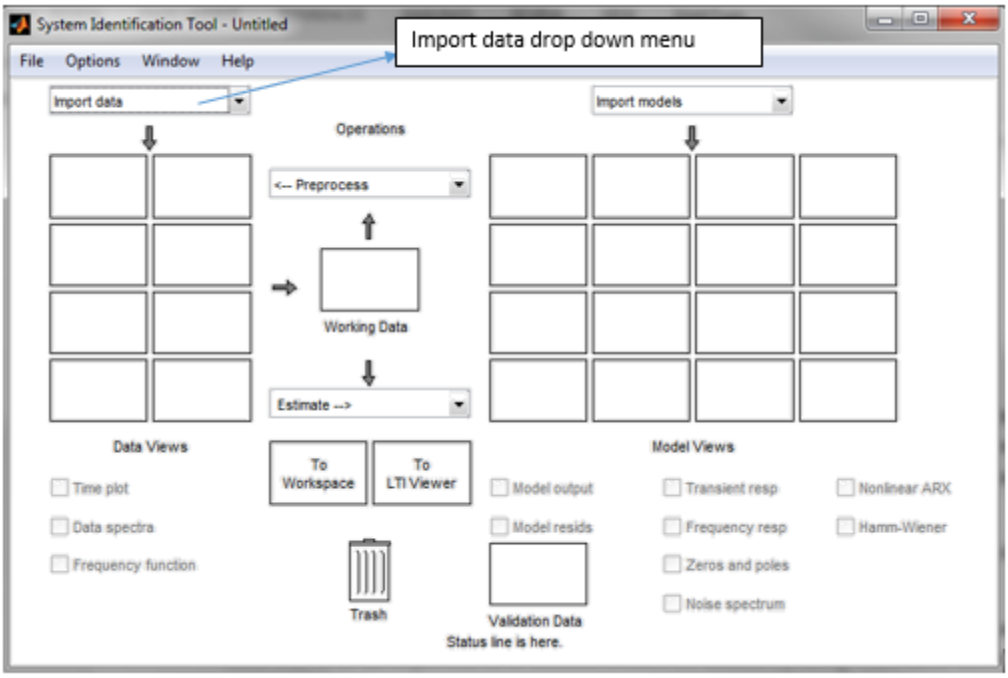

Figure 14: SI Toolbox Application Window

If the input – output data is time - domain then 'Time domain data' must be selected from the drop down menu. If the data is frequency – domain then 'Frequency domain data' must be selected. Once one selects the 'Time domain data' from down menu, a dialogue box appears where input – output data should be specified that is loaded into the workspace. The 'starting time' and the 'sampling interval' can be enter if applicable. Once all the boxes are filled with relevant data, one must click 'import' to import the data to the SI toolbox. The dialogue box is shown in Figure 15.

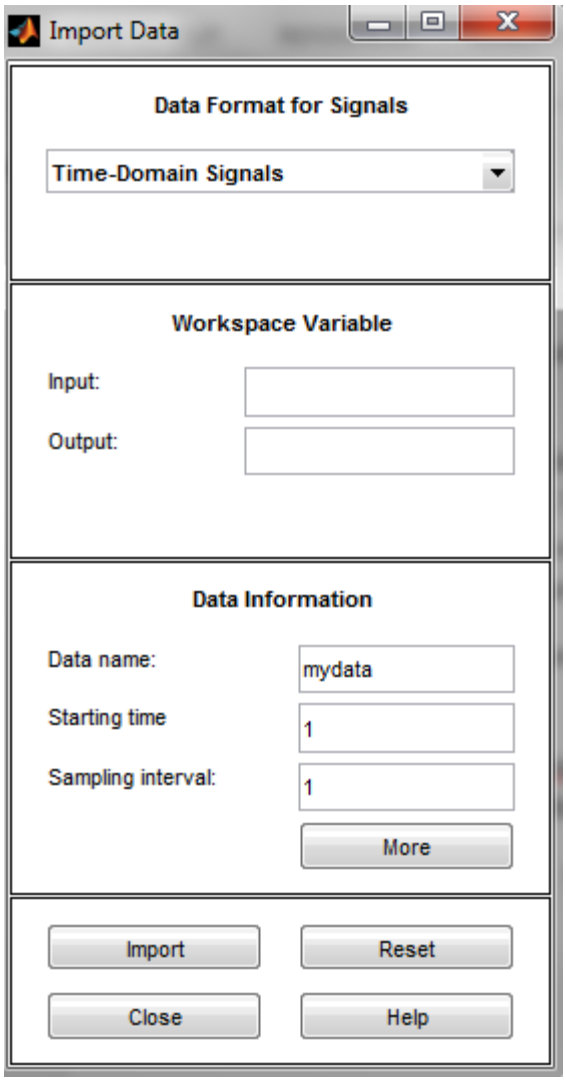

Figure 15: Import Data Window of SI Toolbox.

After the data is imported to the SI toolbox, one is ready to estimate the OE model.

1. Select 'Polynomials Models' from 'Estimate' drop down menu on SI dialogue box. Figure 16 shows SI toolbox window where the specified selection can be made.

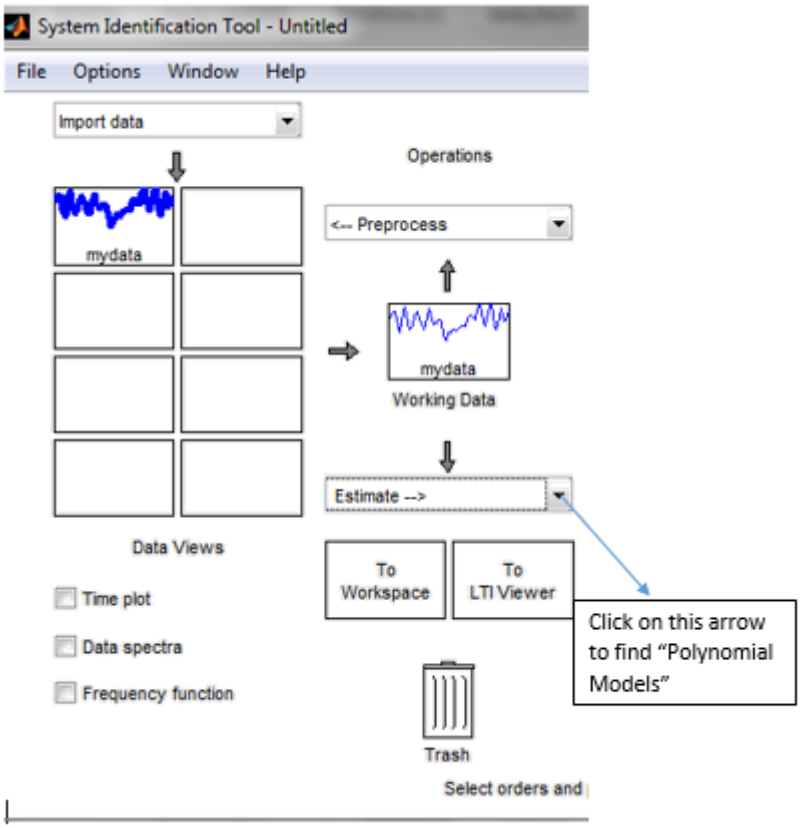

Figure 16: Finding Polynomials Model Menu.

2. A new dialogue box appears as shown in Figure 17. In the box to the right of 'orders', polynomial orders i.e. B and P should be specified along with the time delay. As shown in the Figure (), 'order' box is filled with 2 2 1 inside the square brackets. The first '2' is the order of the B polynomial, the second '2' is the order of the F polynomial, and the '1' is the time delay. They should remain inside the square brackets.

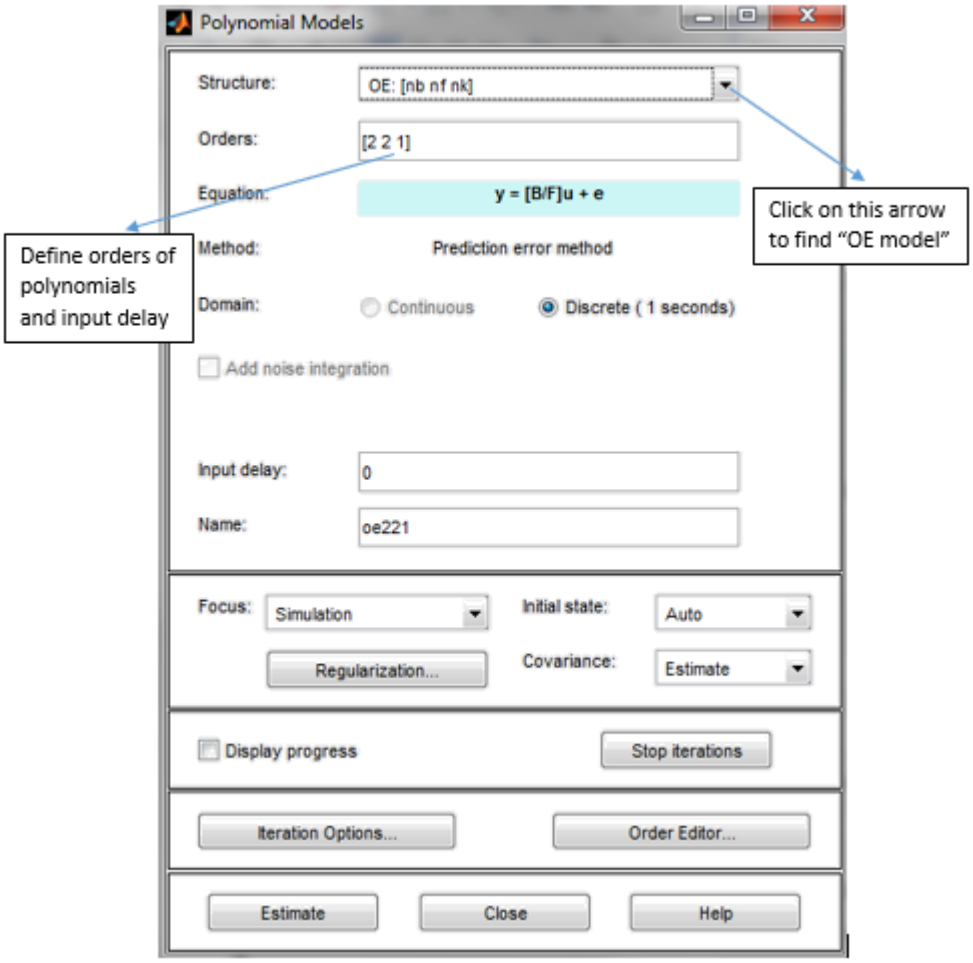

Figure 17: Polynomial Models Estimation Dialogue Box.

3. To identify the OE model the 'Estimate' button at the bottom of the dialogue box should be pressed after the model orders are specified.
4. Matlab™ requires some computation time depending on the input – output data size. Larger the data size results into longer computation time. Matlab™ estimates the OE model and the estimated model fit with the measured output which can be seen by checking a box left to the 'model output' in SI toolbox application window. It is shown in the Figure 18.

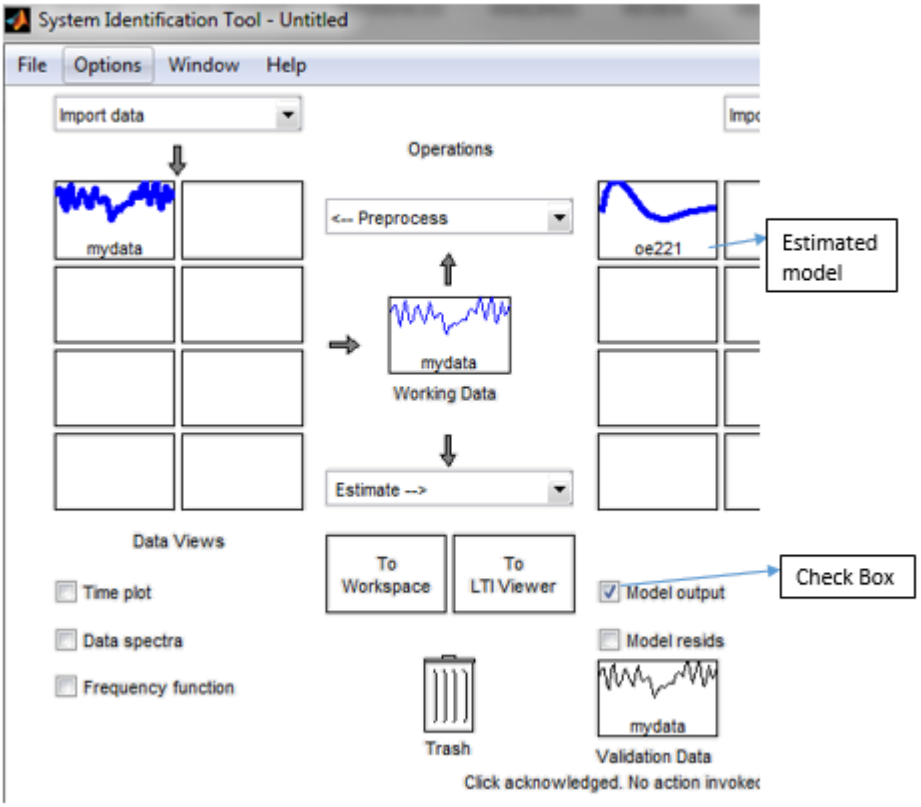

Figure 18: Viewing Estimated Model Fit.

5. If the estimated model fit is low, one can go back to step two and follow the procedure again until a better fit is obtained.
6. As shown in Figure 18, Matlab™ names the estimated model as 'oe221, oe231, etc.' To further investigate the properties of model, one can import this model to the workspace by clicking the model and dragging and dropping the model to 'To Workspace' square box in SI toolbox application window.
7. After the model is imported to workspace, details on the estimated model can be seen by entering the name of the model in the workspace. In this case, the name of the model is 'oe221' and the details of the model given by Matlab™ after entering 'oe221' in workspace is shown in Figure 19.

```
oe221 =
Discrete-time OE model:  y(t) = [B(z)/F(z)]u(t) + e(t)
  B(z) = 0.1081 z^-1 - 0.03288 z^-2

  F(z) = 1 - 1.65 z^-1 + 0.7569 z^-2

Name: oe221
Sample time: 1 seconds

Parameterization:
   Polynomial orders:   nb=2   nf=2   nk=1
   Number of free coefficients: 4
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using POLYEST on time domain data "mydata".
Fit to estimation data: -22.28% (simulation focus)
FPE: 0.000686, MSE: 0.0006855
```

Figure 19: Estimated OE Model Details.

3.2 NON – LINEAR HAMMERISTEIN – WIENER MODEL

A non –linear system is any system that does not satisfy the superposition principle or that is not linear. In this case, the output of a system does rely non - linearly on its inputs. It is possible to differentiate the relationship between the input and output into two or more interrelated features. One of the possible method of doing this is using a Hammerstein – Wiener (HW) model in the SI toolbox to identify the system. In this method of modeling, dynamics of the system can be modeled by a linear transfer function and nonlinearities present in the input and output of the linear system are captured using non – linear functions, [44, 47]. The structure of a HW model is shown in Figure 20, [47].

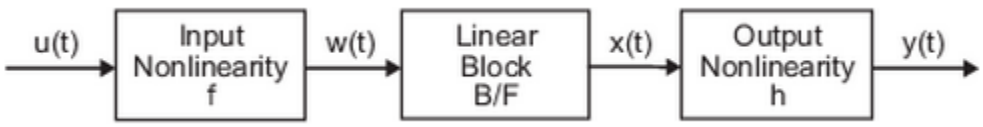

Figure 20: Structure of Non-Linear HW Model, [47].

where,

$u(t)$ = Input data

$w(t) = f(u(t))$ is a non – linear function that transfers input data

$x(t) = \left(\frac{B}{F}\right) w(t)$ is a linear transfer function. Where B and F are as described in OE model.

$y(t) = h\big(x(t)\big)$ is a non – linear function that transfer output of the linear block to the system output.

In Figure 20 structure it can be seen that $w(t)$ and $x(t)$ are the internal variables defining the input and the output of the linear block, respectively. Function f acts on the input of the linear system and the function h acts on the output of the linear system. They are called input and output nonlinearity, respectively. If the model only consists of input non – linearity, it is called Hammerstein model and if the model only contains an output non – linearity, it is a Wiener model. Input and output non – linearity's combined makes the HW model, [47].

The HW model requires three steps in computing the output y, the steps are listed below:

1. Matlab™ computes $w(t)$ at the first step and it is an input to the linear block. The input non – linearity can be captured using sigmoid network, wavelet network, piecewise linear function, 1D polynomial, etc, [47]. A brief description of these methods are as follows:
    a. **Sigmoid Network**: A sigmoid function is a curve of "S" shape. This function is generally is real – valued and differentiable, having a non – negative or non – positive first derivative, one local minimum, and one local maximum. This function is mostly used to induce nonlinearity in artificial neural networks, [47, 48].
    b. **Wavelet Network**: Wavelet is a 'small wave' function used for localize the position and scaling of a given function. These small waves grows and decays in certain time period, [49]. A wavelet network is obtained by cascading a

multi-dimensional wavelet. This network is utilized to approximate arbitrary nonlinear functions, [50].

c. **Piecewise Linear Function**: Piecewise linear function is a function which is made by number of straight – line sections combined. The slope of the function is not constant throughout the graph, [51].

d. **1D Polynomial:** The sum of one or more monomials with positive integer exponents and real coefficients is called a polynomial. The order of a polynomial function is the highest exponent, [52].

2. The second step involves computing $x(t)$ using the input $w(t)$, the polynomials in the numerator and denominator of the linear block should be configured, [47].
3. In the last step of the process, the output of the model is computed using the output of the linear block. As the input nonlinearity, the output nonlinearity can be captured using sigmoid network, wavelet network, piecewise linear function, 1D polynomial, etc. Both input and output nonlinearity's are "static" function whereas the linear block is the dynamic function, [47].

In what follows, the method to estimate HW model using Matlab™ function is described. The 'nlhw' command is used to estimate HW model in Matlab™. The function as in Equation 18 estimates non – linear HW model using 1D polynomial of order 3 to capture the input and output non-linearity, [47]. The linear block has the numerator of 2nd order and denominator of order 3 and input delay of 1.

$$M1 = nlhw(data, [2\ 3\ 1], poly1d(3), poly1d(3)) \quad (18)$$

where,

$M1$ = Estimated non – linear HW model.

$data$ = 'iddata' object containing the input and output signal values.

$[2\ 3\ 1]$ = OE model (specified order for the linear model)

$poly1d(3)$ = Input non-linearity captured using 1D polynomial of 3rd order.

$poly1d(3)$ = Output non – linearity captured using 1D polynomial of 3rd order.

In the similar manner more than one model can be obtained and compared to each other to determine the best one. This can be done by using a Matlab™ function as shown in Equation 19.

$$compare(v, M1, M2, M3) \tag{19}$$

where,

$compare$ = Matlab™ command to compare various model.
v = 'iddata' object containing the input and output signal values to validate the model.
$M1, M2, M3$ = Three differently estimated models.

Once the command as shown above is executed, a window with the plot of different models and measured input - output data appears. This plot also includes the fit percentage of each model to the measured output. Therefore, by inspecting plots and utilizing fit percent's one can select the best model out of all the estimated models. The algorithm to estimate Non – linear HW model is programmed in Matlab™ and is included in Appendix A4.

Non – linear HW models can be estimated using the SI application of Matlab™. In what follows, the steps required to estimate model using the SI application is described.

The procedure to open the SI application and importing data to the application is described in an earlier section of this chapter.

After the data is imported to the SI toolbox, one is ready to estimate the nonlinear Hammerstein – Wiener model.

1. Select 'Nonliner Models' from the 'Estimate' drop down menu on SI dialogue box. Figure 21 shows SI toolbox window where the specified selection can be made.

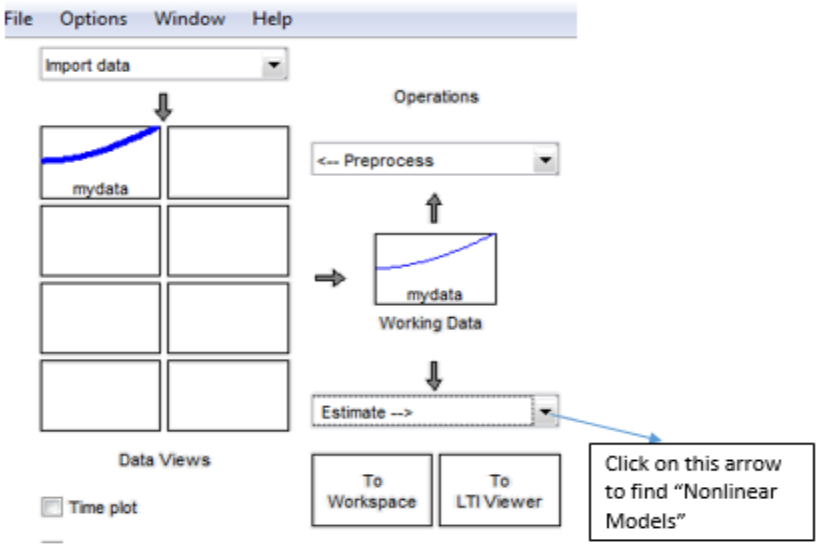

Figure 21: Estimate drop down menu.

2. A new dialogue box appears as shown on Figure 22.

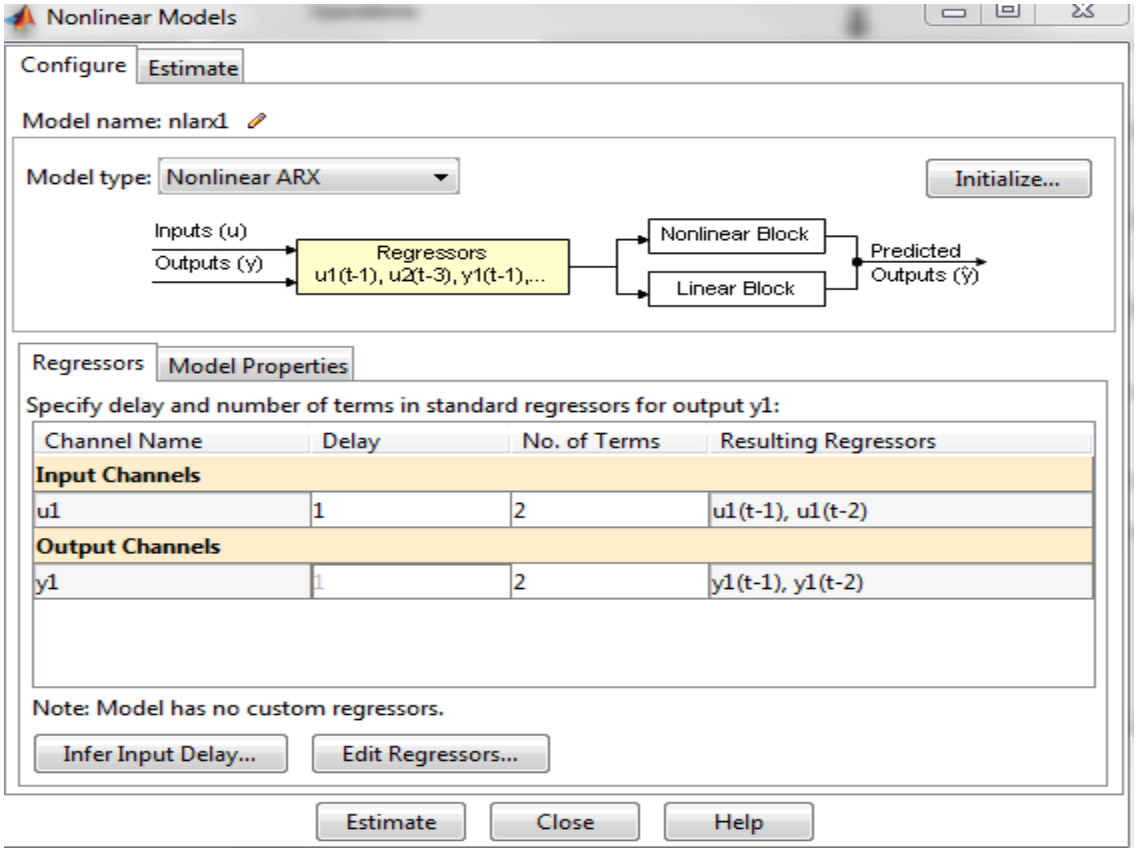

Figure 22: Non-Linear Model Estimation Dialogue Box.

3. On the dialogue box as shown on Figure 23, under configure tab select 'Hammerstein – Wiener' from the drop down menu located to the right of 'Model Type'. The dialogue box appearance changes, and it is shown in Figure 23.

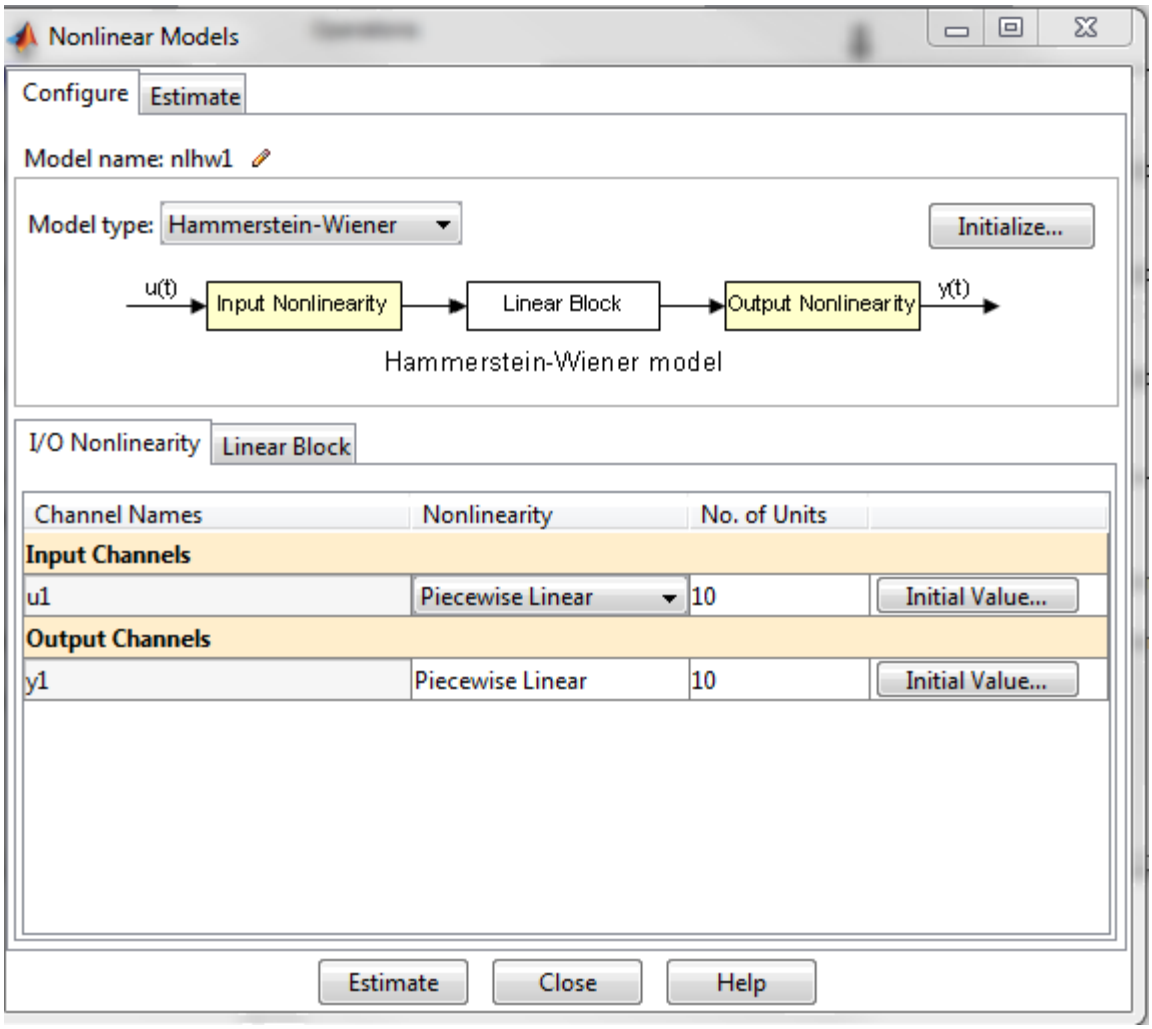

Figure 23: Hammerstein - Wiener Model Estimation Dialogue Box.

4. It is necessary to specify the type of model to estimate the nonlinearities present in the input and output data. Various types of models are available to do so. In this case, one selects to use 'One – dimensional polynomial' model to estimate the nonlinearities. It can be selected by clicking 'Piecewise Linear' present on the 'Nonlinearity' column under "I/O Nonlinearity" tab. Clicking 'Piecewise Linear' turns it into a dropdown menu and 'One – dimensional polynomial' can be

selected from the drop down menu. It is necessary to do this for both input and output. It is also necessary to specify the order of the polynomial to be used for estimating the nonlinearities. It can be specified by entering numbers in the white box to the right of the drop down menu used to select the polynomial model as shown in Figure 24.

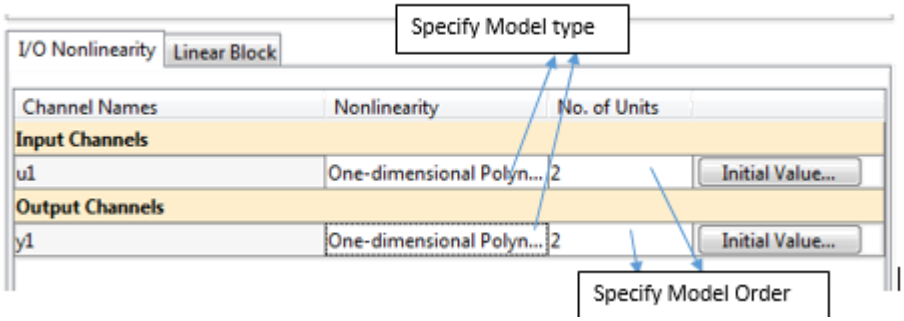

Figure 24: Specifying Model and Model Order.

5. Similarly, the number of Zeros and Poles of the linear block or Output error model should be specified. Based on the formulation of the Hammerstein – Wiener model it must contain at least one Zero on its Output error model. To specify the number of Zeros and poles click on 'Linear Block' tab under 'Configure' tab on the nonlinear model estimation dialogue box. One should see a table after clicking on 'Linear Block' tab where number of Zeros and Poles can be specified. See Figure 25 for details.

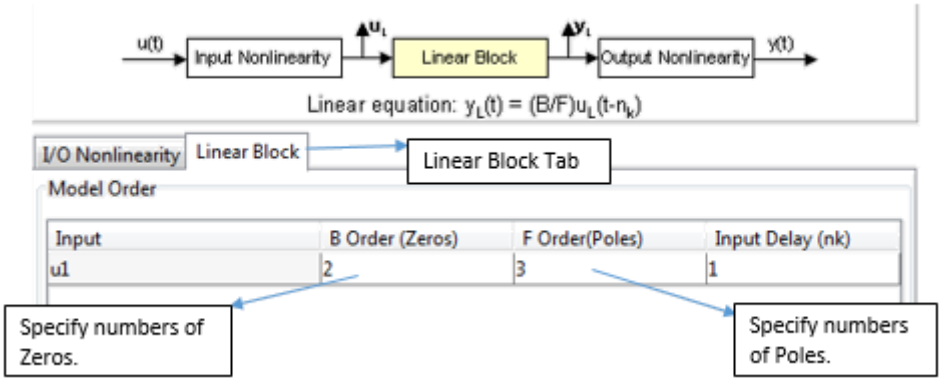

Figure 25: Configuring Linear Block.

6. After all the configurations are done, one is ready to estimate the model. It can be done by clicking the “Estimate” button located in the bottom of the dialogue box.
7. The estimated model’s fit can be checked using the SI toolbox. If the fit percentage is low, one can re-configure the nonlinear model estimation dialogue box and estimate the model again.
8. After the estimation is complete, Matlab™ names the model such as ‘nlhw1, nlhw2, etc.’ and it appears on the SI toolbox window. To further investigate the estimated model it can be imported to the Matlab™ workspace and extracted.
9. It can be done by dragging the desired model from the SI toolbox window and dropping it to the ‘To Workspace’ box present in the SI toolbox window.
10. The model can now be seen on Matlab™ work space window. In this case model is named ‘nlhw2’. If one enters ‘nlhw2’ command on the command window it displays the information about the model as shown in the Figure 26.

```
>> nlhw2

nlhw2 =
Hammerstein-Wiener model with 1 output and 1 input
 Linear transfer function corresponding to the orders  nb = 2, nf = 3, nk = 1
 Input nonlinearity: poly1d of degree 2
 Output nonlinearity: poly1d of degree 2
Loss function: 4.6194e-05
Sampling interval:  0.0005
Estimated by PEM
```

Figure 26: Model Information.

11. To extract the input nonlinearity and output nonlinearity of the estimated model in mathematical form, follow the commands as shown in Figure 27 which are entered in the Matlab™ command window. It presents the coefficients of the polynomials.

```
>> nlhw2.InputNonlinearity
One-dimensional polynomial estimator:
          Degree: 2
    Coefficients: [0.0436 0.9988 -4.6522e-04]

>> nlhw2.OutputNonlinearity
One-dimensional polynomial estimator:
          Degree: 2
    Coefficients: [-3.3174e-04 0.0155 0.0031]
```

Figure 27: Input - Output Non - Linearity Model Extraction.

12. Similarly, the Output Error model or linear model can be extracted by entering the command as shown in Figure 28.

```
>> nlhw2.LinearModel

ans =
Discrete-time OE model:  y(t) = [B(z)/F(z)]u(t) + e(t)
  B(z) = z^-1 - z^-2

  F(z) = 1 - 0.9908 z^-1 - z^-2 + 0.9908 z^-3

Name: nlhw2
Sample time: 0.0005 seconds

Parameterization:
   Polynomial orders:   nb=3   nf=3   nk=0
   Number of free coefficients: 6
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Extracted from an IDNLHW model
```

Figure 28: Linear Model Extraction.

13. Beside extracting the models, one can also plot the entire model by entering 'plot(nlhw2)' command on the command window. On the plot window, one can observe the input/output nonlinearities model plot as shown in Figure 29 and 30.

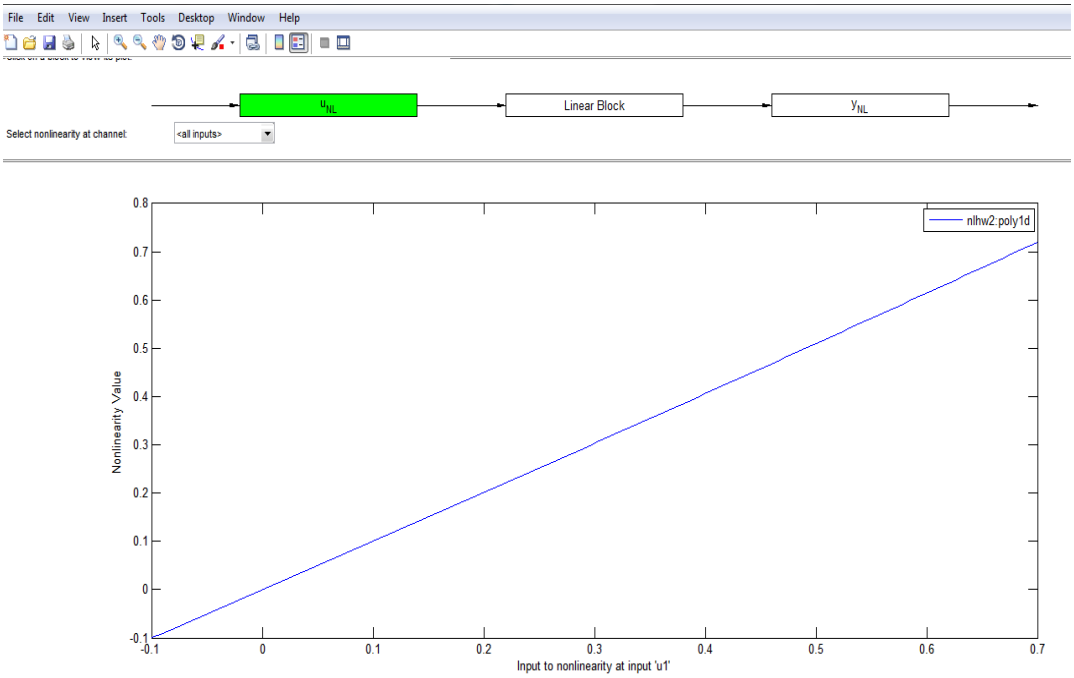

Figure 29: Input Non - Linearity Model Plot.

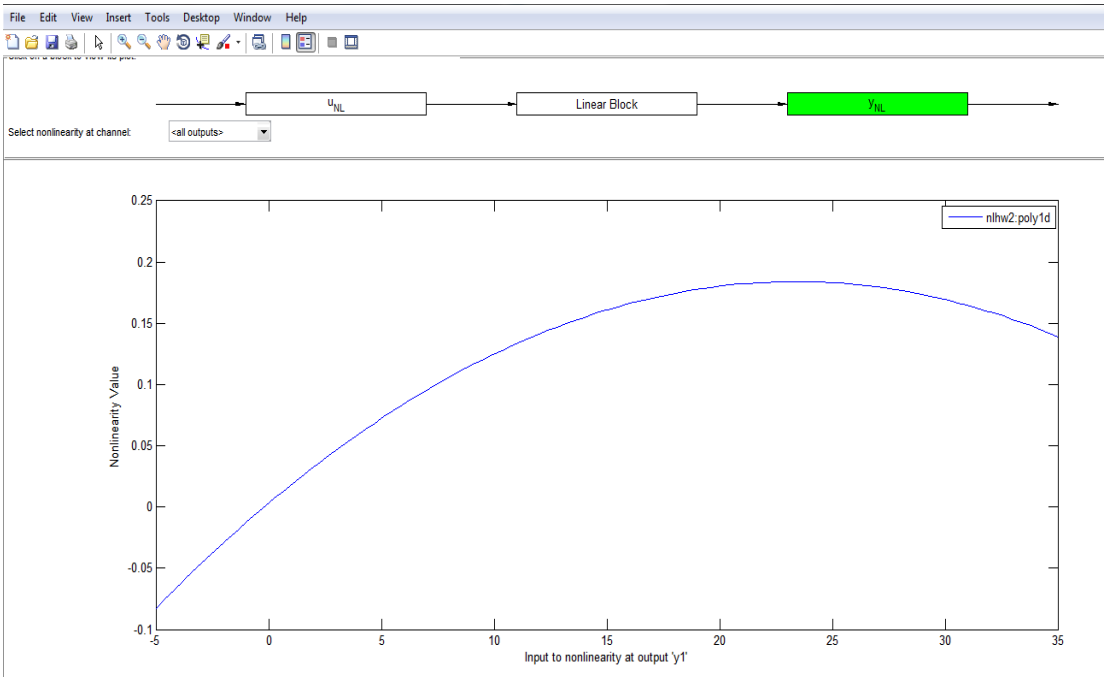

Figure 30: Output Non - Linearity Model Plot.

14. For the linear part of the model, one can view the step response plot, Bode plot, impulse response plot, and Pole – Zero map. It can be done by selecting 'Linear

Block' box on the plot window. The box turns green once the selection is made and displays the plot. A drop down menu appears on the top right corner of the plot window after the Linear Block box is selected. All the plots that can be generated are shown in Figure 31 through 34.

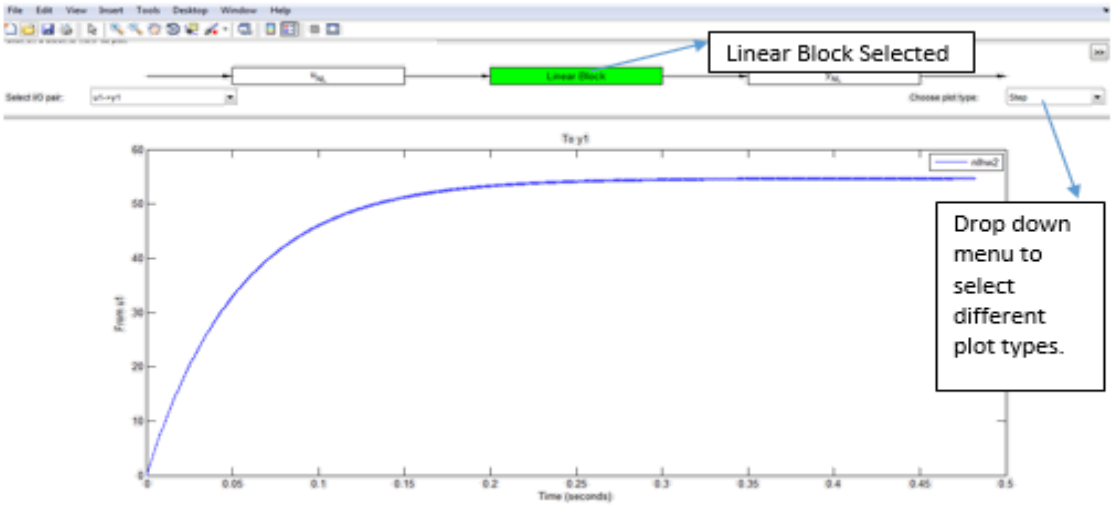

Figure 31: Step Response Plot.

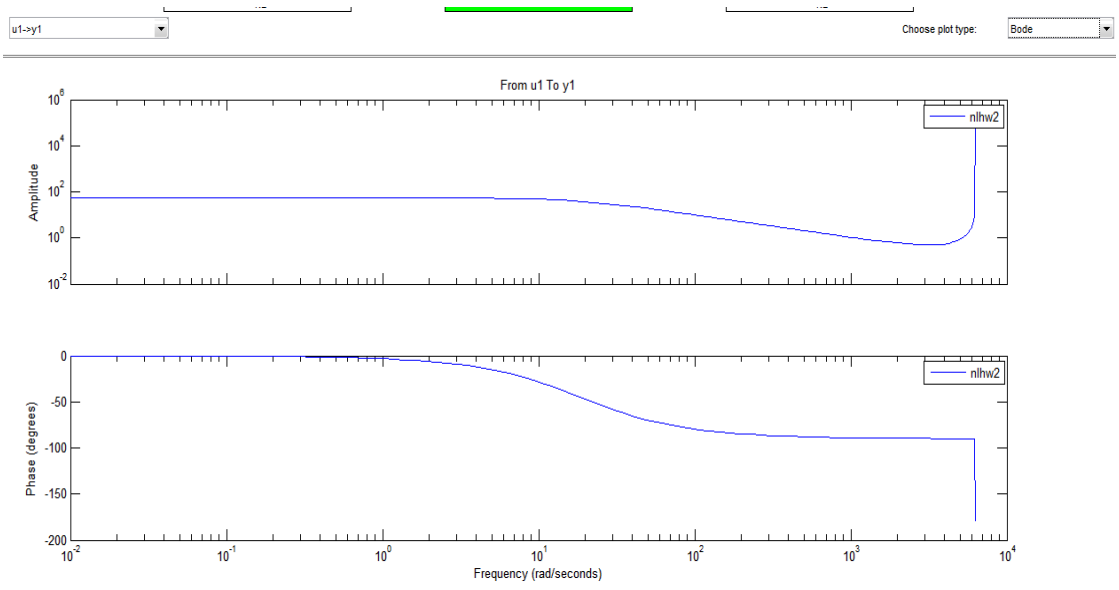

Figure 32: Bode Plot.

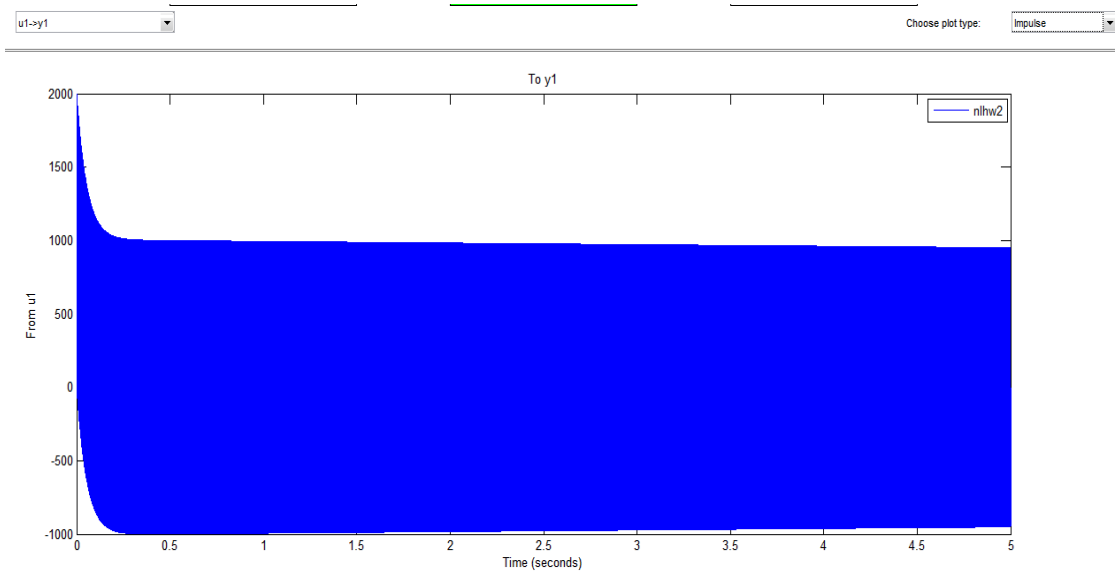

Figure 33: Impulse Response Plot.

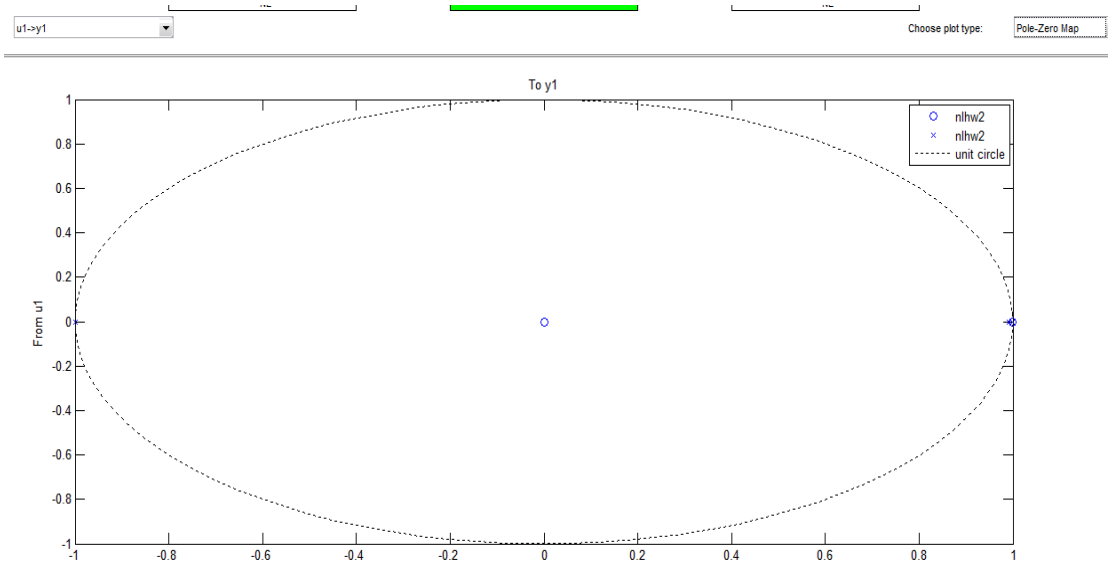

Figure 34: Poles - Zeros Map.

CHAPTER 4: EXPERIMENTAL SET-UP

The system used for the experiment is a one – stage axial compressor. It is located at the research facility of Chinese Academy of Sciences (CAS), Beijing, China. It is used for obtaining data through various experiments. This compressor is equipped with an array of transducers capable of capturing slow dynamics data, changes in axial direction across the compressor are measured and the fast dynamics data. The fast dynamics data are associated with the flow between individual blades. The major components schematics of the compressor are shown in Figure 1 of Chapter 1.

The major components of the compressor as shown in Figure 1 of Chapter 1 are the inlet, the compressor stage, the exit duct, plenum, and the throttle. The one used for the experiment housed at CAS has one compressor stage. The overall compressor used for experimental purpose is shown in Figure 35.

Figure 35: Side View of the Experimental Compressor.

In the Figure 35, the inlet of the compressor is on the left hand side and the outlet is located to the right. However, the outlet (cone and outer cylinder) of the compressor as shown in Figure 35 has been modified and is now passed through the stationary wall. The outer cylinder also passes through the wall and slides back and forth. The new setup of the compressor is shown in Figure 36. The outlet consists of two pieces, a stationary throttle cone and a moveable throttle outer cylinder. The cylinder is used to regulate the flow rate and is controlled by a computer. The cylinder is connected to a stepper motor

(controlled by a computer) by a mechanism to help cylinder change its position with respect to the stationary cone creating small openings of the outlet, driving compressor to stall.

Figure 36: Compressor Outlet Setup.

The outer cylinder has the diameter of 540 mm and is made by using aluminum. The mechanism that moves the cylinder is a screw mechanism, connected to the stepper motor. The clockwise and anti-clock wise rotation of the stepper motor drives the cylinder in forwards and backwards directions. This mechanism is shown in Figure 37. As seen in Figure 37, the outer cylinder sits on a structure that moves along the rail and this structure is connected to the screw. The stepper motor drives the outer cylinder at a constant speed of 2mm/sec.

Figure 37: Throttle Actuation Mechanism.

A different motor (blue color) as shown in Figure 36 is used to drive the compressor fan. The speed of this motor is controlled by using a variable frequency drive (VFD).

In order to capture the fast dynamics and slow dynamics data, ten anemometer type pressure sensors are attached to compressor casing in the vicinity of blade passage. It can be seen in Figure 38. Variable DC power supplies are used to power these sensors. Sampling rates are set differently for acquiring fast and slow dynamics data. Higher sampling rate (e.g. 20,000 Hz) is set for fast dynamics data and low sampling frequency (e.g. 100 Hz, 200 Hz) is used for slow dynamics.

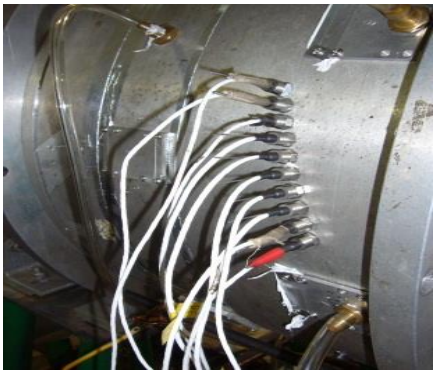

Figure 38: View of Mounted Pressure Sensors.

The relative position of these sensors are shown in the Figure 39. The experimental compressor has 58 blades. The position of the first pressure sensor is just ahead of the tip of a blade and they are laid out so that they gradually pass over the span of the blade tip and ends little behind the edge of the tip.

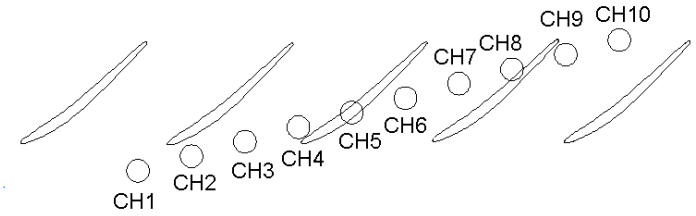

Figure 39: Distribution of Pressure Sensors.

(Include the dimension of the pressure sensor locations)

4.1 SET – UP FOR AN OLDER EXPERIMENT

This compressor was set-up differently when an experiment was conducted in 2013. In order to capture the pressure data eight pressure sensors are used and a Hall Effect sensor to track the position of the compressor blades. Sensor one and sensor eight are arranged in the inlet and outlet of the compressor, respectively. Sensors two through seven are installed over the blade tip from the leading edge to the trailing edge. The arrangement of pressure sensor are shown in Figure 40.

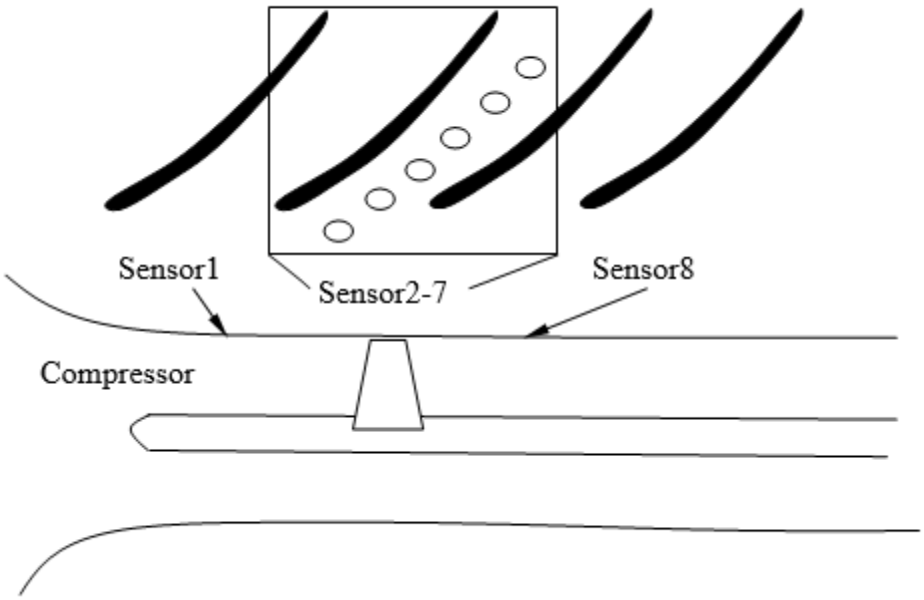

Figure 40: Pressure Sensors Arrangements.

The location of the sensors installed along the blade chord in axial and circumferential direction of the compressor is included in Table 1. The sensors location are with respect to the first sensor in the circumferential direction and for the axial direction, the leading edge of the compressor blade is used as reference.

Table 1: Location of Sensors

Sensors	Axial Co – ordinate (mm)	Circumferential Co – ordinate(mm)
Sensor 2	-4.387	0
Sensor 3	0.033	5.327
Sensor 4	4.583	5.033
Sensor 5	8.993	5.42
Sensor 6	13.83	4.987
Sensor 7	18.09	5.283

4.2 ALTERNATIVE METHOD OF THROTTLE ACTUATION

As mentioned earlier, the stepper motor drives the outer cylinder at the speed of 2mm/sec (axial speed). This is relatively slow and is not enough to perform experiments to excite the overall system dynamics. For System Identification purpose this is not sufficient because better result can be obtained if the input – output data's are rich in information about the system. In order to achieve this, all the system modes should be excited and reflected in the output signal. Therefore, an alternative method of throttle actuation scheme prototype is designed and is being implemented on the compressor. This design will incorporate a wider range of throttle actuation input frequencies. In this new design prototype, a series of flaps are added around the throttle cylinder and a new mechanism to drive the flaps is designed. High frequency rotatory solenoids are used to drive the flaps back and forth in the axial direction of the compressor. A rack and pinion design with the flaps attached to it is also designed using SolidWorks™ and is shown in the Figure 41.

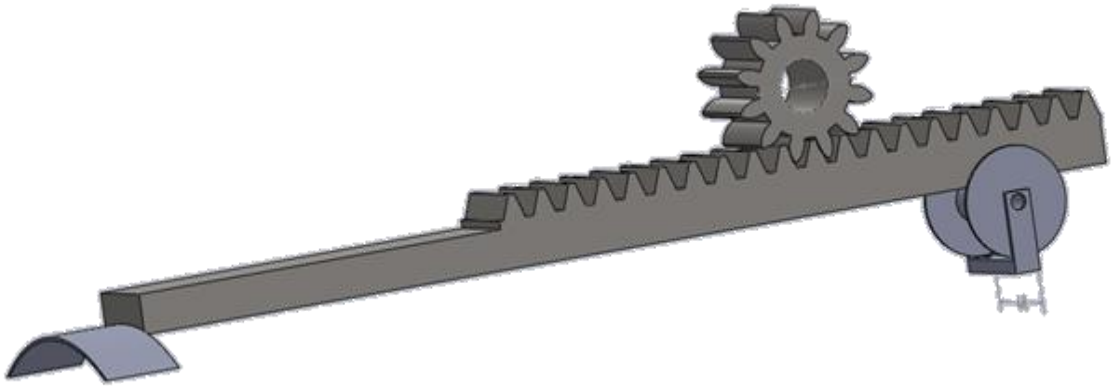

Figure 41: Rack and Pinion Design with Flap.

The final design of the rack and pinion with flaps attached to rotating solenoid in the throttle is shown in Figure 42. The overall weight of the module including rack, pinion, and flaps is 180 grams with 1060 aluminum used as material. The solenoid to drive this mechanism is manufactured by LEDEX. The particular solenoid used for this mechanism

is “Ultimag® Size 4EM”. This solenoid can produce maximum torque of 0.32 Nm and the maximum frequency response of this rotary solenoid is 78 Hz. The specification sheet of this solenoid is included in Appendix A5.

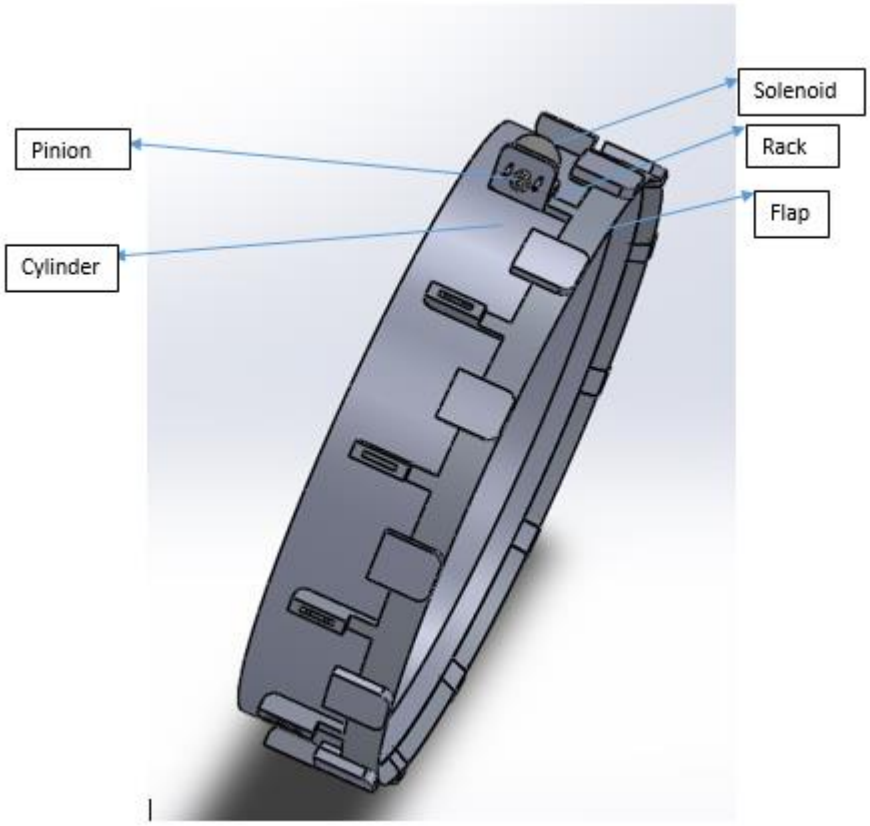

Figure 42: Rack and Pinion Assembled in Throttle Cylinder, [53].

CHAPTER 5: THEORY OF MISSING DATA

The missing data problem arises in almost all research work and statistical analysis, [54, 55]. There are a number of ways for this problem to appear. The causes are briefly discussed below:

- **Missing Completely At Random (MCAR):** In this case, the missing data doesn't depend on the observation and other dependent variables. Any piece of data is as likely as any other piece to be missing, [56].
- **Missing At Random (MAR):** In this case, the missing data depends on the observed dependent variables. This is possible by controlling other variables, [56].
- **Missing Not At Random (MNAR):** In this case, missing data depends on the dependent variables that are not observed, [56].

5.1 TREATING MISSING DATA TECHNIQUES

The traditional way of treating missing data is list-wise deletion or simply remove the pieces where the data is missing. However, this approach results in reduction of available data size for analysis purposes. Another traditional way of dealing with missing data is to substitute the mean of the available data for the missing data. This method is not very reliable because it does not add new information to the existing data. The overall mean of the data would be the same even after the missing data is filled, [54].

Linear regression substitution method is also used to substitute missing data. This method predicts the missing values on the basis of other data that is present. This method is being used for a long time. It inputs a value to the missing place that is conditional on other available information which is advantageous over the mean substitution. However, there is no new information being added based on existing information. The problem due to variance error remains but this method definitely increases the sample size and decrease the standard error, [54].

In 1994, Cook and Stefanski [57] proposed a method of simulation extrapolation. This method is simulation based, utilized to do inference in non –linear models where measurements error exists in co-variates. This method is completed in two steps. First, the effect of increasing the measurement error magnitude on an estimator is revealed by simulation. As the measurement error increases, the estimator bias tends to increase too. The association of the estimator bias with measurement error is estimated in the first step. In the second step, the trend obtained from the first step is used to get the reduced biased estimate by extrapolation. Once, the measurement error is disappeared, a function to approximate true the parameters is estimated, [57].

Gopaluni et al. [58] presented a novel way to deal with the missing data because of multi-rate sampled data systems. A common technique used in dealing with multi-rate data is to interpolate (linearly and quadratically) within the sampled data. However, with linear and quadratic interpolation the variation in the input during the period of interpolation is considered. Another way to deal with this situation is by using lifting techniques. In this technique, multi-rate identification problem is converted to multivariable identification problem using lifting operator. A novel idea forwarded by Gopaluni et al. is an iterative identification algorithm. In the first step of this approach, a simple model is identified using multi-rate data. By using this simple model and expectation maximization approach, missing data points in slow sampled data are obtained. The estimated data points and the original data points are converged to make a new model and this procedure is repeated until the models converge, [58].

Various ways are used to analyze time series data and depending on the type of the data, appropriate method are used to find the missing values. Deterministic or stochastic methods are used in handling time series data. Deterministic modeling is also known as numerical analysis modeling. In this method, an appropriate way of fitting a function to the time series data is used. The resulting function is used to estimate the missing value. A variety of curves are used to obtain best fit to the time series data. Stochastic modeling is also known as time series modeling. Box – Jenkins' Autoregressive Integrated Moving Average (ARIMA) models are most commonly used models in

modeling time series data. It is a statistical approach, it identifies the pattern of the available data and finds an appropriate procedure to generate data based on the identified pattern, [59].

5.2 TREATMENT TECHNIQUES USED IN THIS RESEARCH

The missing data problem for this research arises from the particular measurement setup employed. The compressor used in this research consists of one Hall Effect sensor to track the position of a compressor blade while it is rotating. One compressor blade is tracked out of 58 blades and the pressure data recorded when the compressor blade returns to the same position is used to analyze the dynamics of the compressor. About 30 – 34 data points are recorded once the compressor blade returns to the same position in every rotation. Due to the lack of more pressure sensors other positions of the compressors blades during rotations could not be tracked. Therefore, the events that are occurring all around the compressor are not captured. This creates a situation of missing data. In one rotation the number of data point captured at the sampling rate of 20000 Hz is about 480 – 490. Therefore, only a small chunk of data is available to analyze the compressor dynamics.

An autoregressive model is programmed using Matlab™ to deal with the issue of missing data. This program estimates the missing data stochastically. This algorithm was developed based on autoregressive model with exogenous input (ARX) having missing data. First, this programs separates the 30 data points captured when the compressors blade returns to the same position in every rotation and fills in zeros for rest of the data points. Based on the model order selected it constructs matrices for estimation. Finally, the estimated data's and the data set with the missing values are plotted to compare the result of the estimation with the original data set. The higher model order selection gives the better estimation of the missing data in expense of some computation time. This program (Newestmissdata_mod.m) is included in Appendix A6. The other Matlab™ file that relate to this algorithm is also included in Appendix A7.

In what follows, mathematical formulation of this algorithm is described. A model structure of the form as shown in Equation 20 is assumed. The output of this assumed model are missing. Therefore, an algorithm to estimate the missing parameters is formulated based on least – squares.

$$\sum_{i=1}^{p_1} a_i y_{k-i} + \sum_{i=1}^{p_1} b_i u_{k-i} + \varepsilon_k \tag{20}$$

where, $\{y\} \in \mathbb{R}^{n_y \times L}$, $\{u\} \in \mathbb{R}^{n_u \times L}$, are the input and output data vectors captured during an experiment. The sampling frequency of the experiment is f_s and the number of discrete data points in input and output vector is *L*. n_y is the number of outputs, and n_u represents the number of inputs. Since the output is not recorded all the time, then

$$y^m(k) = y(k) * g(k) \tag{21}$$

where, $g(k) = \begin{cases} 1 & \text{if } y \text{ is measured} \\ 0 & \text{if } y \text{ is not measured} \end{cases}$

The probability of measuring the output, i.e. $E[g(k)] = P_g$ defines the expected value of $g(k)$. The parameters of the parametric model in Equation 20 can be written as in Equation 22.

$$\Theta = \begin{bmatrix} a_1 & a_2 & \dots & a_{p_1} & b_1 & b_2 & \dots & b_{p_1} \end{bmatrix}^T \tag{22}$$

The data vectors can be defined as follows:

$$\phi(k) = \begin{bmatrix} y_{k-1} & y_{k-2} & \dots & y_{k-p_1} & u_{k-1} & u_{k-2} & \dots & u_{k-p_1} \end{bmatrix}^T \tag{23}$$

$$\varphi(k) = \begin{bmatrix} y^m_{k-1} & y^m_{k-2} & \dots & y^m_{k-p_1} & u_{k-1} & u_{k-2} & \dots & u_{k-p_1} \end{bmatrix}^T \tag{24}$$

The least – squares approach as in Equation 25 could be used to compute the parameter vector given in Equation 22.

$$\Theta = \left\{ E\left[\phi(k) \phi(k)^T \right] \right\}^{-1} E\left[\phi(k) y(k) \right] \tag{25}$$

Assume, $E\left[y^2(k)\right]=\sigma_y^2$, the variance of the true output, the following quantities can be computed.

$$E\left[y^m(k_1)y^m(k_2)\right]=E\left[y(k_1)y(k_2)\right]*E\left[g(k_1)g(k_2)\right]$$
$$=\begin{cases}P_g^2E\left[y(k_1)y(k_2)\right] & \text{if } k_1\neq k_2\\ P_g^2\sigma_y^2 & \text{if } k_1\neq k_2\end{cases} \tag{26}$$

$$E\left[y^m(k_1)u(k_2)\right]=P_gE\left[y(k_1)u(k_2)\right] \tag{27}$$

$$E\left[\varphi(k_1)y^m(k_2)\right]=E\left[\begin{Bmatrix}y(k-1)y(k)g(k-1)g(k)\\ y(k-2)y(k)g(k-2)g(k)\\ \vdots\\ y(k-p_1)y(k)g(k-p_1)g(k)\\ u(k-1)y(k)g(k)\\ \vdots\\ u(k-p_1)y(k)g(k)\end{Bmatrix}\right]=E\left[\phi(k)y(k)\right]\circ\begin{Bmatrix}P_g^2\\ P_g^2\\ \vdots\\ P_g^2\\ P_g\\ \vdots\\ P_g\end{Bmatrix} \tag{28}$$

where $\circ$ implies an element by element multiplication.

$$\begin{aligned}&E\left[\varphi(k_1)y^m(k_2)\right]=\\ &=E\begin{bmatrix}P_g^2\sigma_y^2 & P_g^2E\left[y(k_1)y(k_2)\right] & \cdots & P_g^2E\left[y(k_1)y(k_2)\right] & P_gE\left[y(k_1)u(k_2)\right] & \cdots & P_gE\left[y(k_1)u(k_2)\right]\\ P_g^2E\left[y(k_1)y(k_2)\right] & P_g^2\sigma_y^2 & \cdots & P_g^2E\left[y(k_1)y(k_2)\right] & P_gE\left[y(k_1)u(k_2)\right] & \cdots & P_gE\left[y(k_1)u(k_2)\right]\\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots\\ P_g^2E\left[y(k_1)y(k_2)\right] & P_g^2E\left[y(k_1)y(k_2)\right] & \cdots & P_g^2\sigma_y^2 & P_gE\left[y(k_1)u(k_2)\right] & \cdots & P_gE\left[y(k_1)u(k_2)\right]\\ P_gE\left[y(k_1)u(k_2)\right] & P_gE\left[y(k_1)u(k_2)\right] & \cdots & P_gE\left[y(k_1)u(k_2)\right] & \sigma_u^2 & \cdots & E\left[u(k_1)u(k_2)\right]\\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots\\ P_gE\left[y(k_1)u(k_2)\right] & P_gE\left[y(k_1)u(k_2)\right] & \cdots & P_gE\left[y(k_1)u(k_2)\right] & E\left[u(k_1)u(k_2)\right] & \cdots & \sigma_u^2\end{bmatrix}\\ &=\begin{bmatrix}\boxed{P_g^2} & \boxed{P_g}\\ \boxed{P_g} & \boxed{1}\end{bmatrix}\circ E\left[\phi(k)\phi(k)^T\right]\end{aligned} \tag{29}$$

Substituting the quantities derived above into Equation (25) gives

$$\hat{\Theta}=\left\{\begin{bmatrix}\boxed{1/P_g^2} & \boxed{1/P_g}\\ \boxed{1/P_g} & \boxed{1}\end{bmatrix}\circ E\left[\varphi(k)\varphi(k)^T\right]\right\}^{-1}E\left[\varphi(k)y^m(k)\right]\circ\begin{Bmatrix}1/P_g^2\\ 1/P_g^2\\ \vdots\\ 1/P_g^2\\ 1/P_g\\ \vdots\\ 1/P_g\end{Bmatrix} \tag{30}$$

$$\hat{\Theta}=\left\{\frac{1}{L}\sum_{k=1}^{L}\varphi(k)\varphi(k)^{T}\circ\left[\begin{array}{cc}\boxed{1/P_g^2} & \boxed{1/P_g}\\ \boxed{1/P_g} & \boxed{1}\end{array}\right]\right\}^{-1}\left\{\frac{1}{L}\sum_{k=1}^{L}\varphi(k)y^{m}(k)\right\}\circ\left\{\begin{array}{c}1/P_g^2\\ 1/P_g^2\\ \vdots\\ 1/P_g^2\\ 1/P_g\\ \vdots\\ 1/P_g\end{array}\right\} \tag{31}$$

where, $\boxed{x}\in\mathbb{R}^{p_1\times p_1}$ unity matrices multiplied by *x*.

Few other ways of estimating missing data are investigated. Various commands that are used in Matlab™ to fill in missing data are investigated including '*ecmlsrmle*', *'resample'*, and *'interp'*. Command 'ecmlsrmle' is included in financial toolbox of Matlab™ and uses least square regression method to calculate missing data. The syntax for the command is as shown in Equation 32.

$$[Parameters, Covariance] = ecmlsrmle(Data, Design) \tag{32}$$

where,

$Parameters$ = Vector of estimates.

$Covariance$ = Matrix of estimates for the covariance of the regression model's residuals.

$Data$ = Data set where the values are missing and are represented by NaNs.

$Design$ = The standard form for regression.

This method didn't prove any useful with existing problem because of the lack of the design matrix information.

The next Matlab™ command that is investigated is 'resample'. The syntax for this command in Matlab™ is given in Equation 33.

$$y = resample(x, p, q) \tag{33}$$

The above command resamples the vector x at p/q times the original sampling rate. This method is only good for the data that has no missing values.

The other investigated Matlab™ command is *'interp'*, the syntax of this command is shown in Equation 34.

$$y = interp(x, r) \tag{34}$$

This command increases the sampling rate of x (original data set) by a factor of r. The output vector y is r times as long as the original data.

This method is not considered to be beneficial to the existing problem because the linear interpolation is not a good prediction for missing data. Also, increasing the data set by a factor doesn't ensure accurate number of total data points because the number of data points captured in each rotations varies. Each correlation coefficient should be located in between each rotation data, which is not possible with this command.

CHAPTER 6: EXPERIMENTAL RESULTS

This Chapter includes the results of the System Identification (SI) conducted on the slow and fast dynamics data. It also includes the procedure used for processing data prior to SI.

6.1 SLOW DYNAMICS SYSTEM IDENTIFICATION AND RESULTS

Experiments are conducted on a single stage compressor during the summer of 2014 with the objective to capture its dynamics. The slow dynamics data of the compressor consists of eleven columns. Column one of the data file contains the flow coefficient data (input) and column two of the data file consists of pressure rise coefficient data (output). The data obtained from the experiment are corrupted with noise. Therefore, it is necessary to remove noise out of the data as much as possible. The sampling frequency of the data is 2000 Hz. The plot of input data (column 1) of flow coefficient 0.50 is shown in Figure 43.

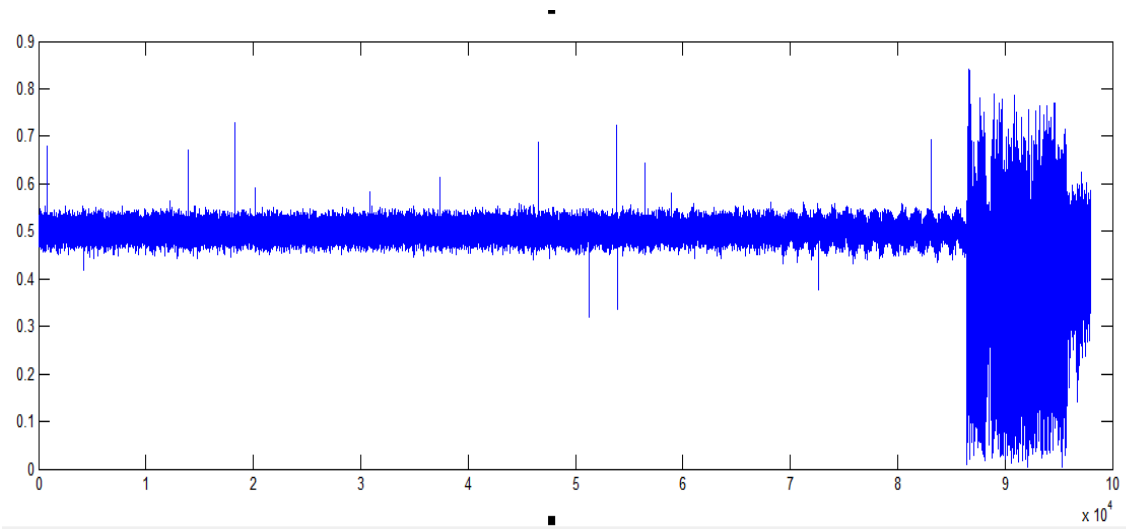

Figure 43: Input Data of Flow Coefficient 0.50.

As it can be seen in the Figure 43, no compressor dynamics is captured until 70,000 data points and it is just the noise. Towards the end of the plot, big magnitude data can be seen, it is noise caused during the stall of the compressor. The compressor dynamics

is represented by the data points located between 70,000 and 85,000 data points. That portion of the data is separated for filtration and system identification.

Various filters are investigated to reduce the noise influence. A Bayesian filter as described in Chapter 2, is used in removing noise from the Electromayography (EMG) signal. This filter is programmed in Matlab™ used to filter the compressor data. The results of the filtration is shown in Figure 44. The abscissa of the Figure 44 is the number of data points and the ordinate is the magnitude of the data. The blue plot is of the original data whereas the red plot is of the filtered data. The following result is obtained by choosing “alpha” or the diffusion rate to be five and “beta” or the probability of sudden jumps to be eight. These are the two parameters of the Bayesian filter, which are changed in order to optimize the performance of the filter. A number of iterations is performed and the result is analyzed using the power spectrum plot. A combination of alpha and beta is chosen that cancelled most of the higher frequency data. In what follows, power spectrum plots of original data and filtered data are discussed.

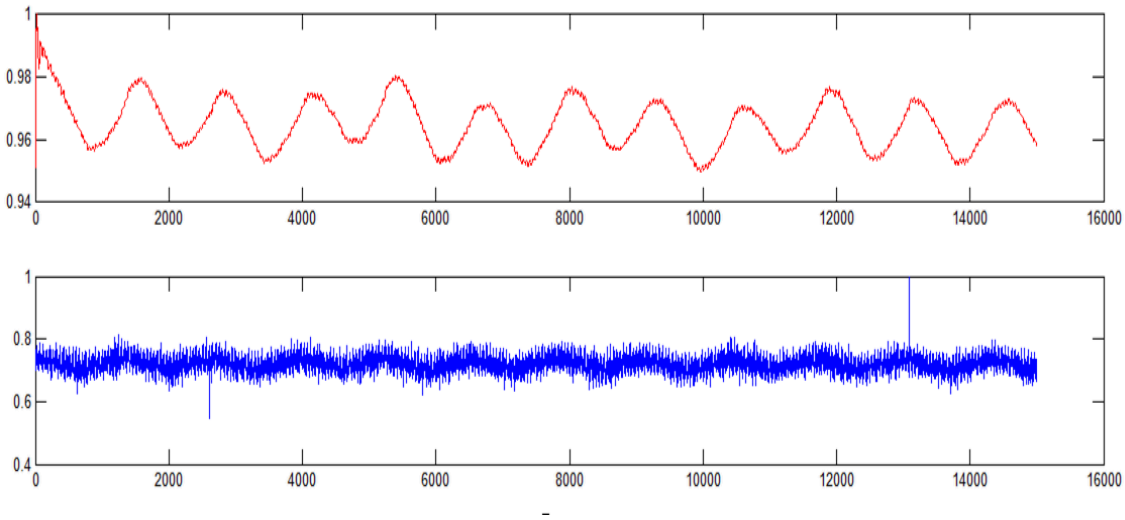

Figure 44: Input Data Filtered.

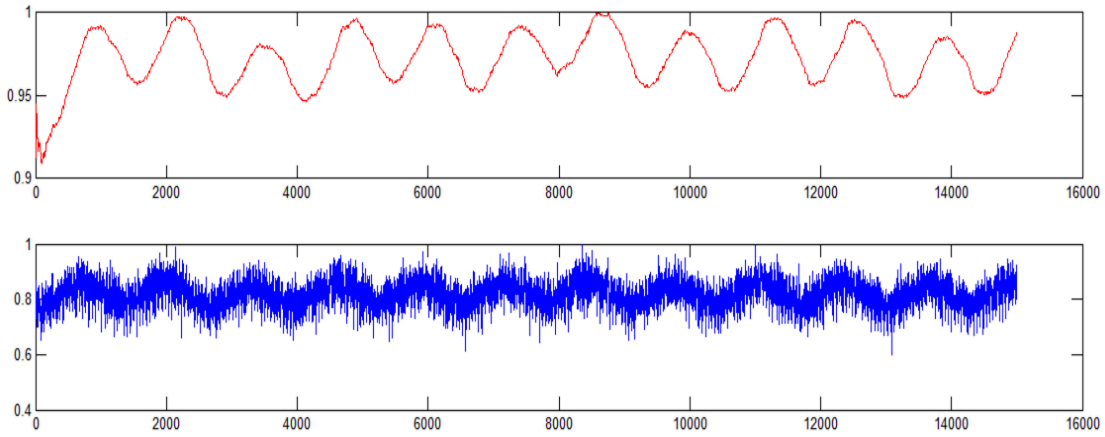

Figure 45: Output Data Filtered.

As it can be seen from the Figures 44 and 45, the filtered data is less noisy than the original data. In order to verify higher frequencies are taken out from the input/output data, the power spectrum plot is used. Once the input/output data's are imported to the SI toolbox, the power spectrum plot can be obtained by checking "Data Spectra" checkbox on the SI toolbox window. The power spectrum plot of the original data is plotted first and then compared it with the power spectrum plot of the filtered data. This procedure is applied to both input and output data and the filtered output is used to do the SI. The power spectrum plot of the data before filtration and filtered data are shown in Figures 46 and 47, respectively. In the Figures 46 and 47, "u1" is the input data and "y1" is the output data. As it can be seen in Figure 46, there is a peak at 40 Hz (marked with red circle) which is the blade passing frequency.

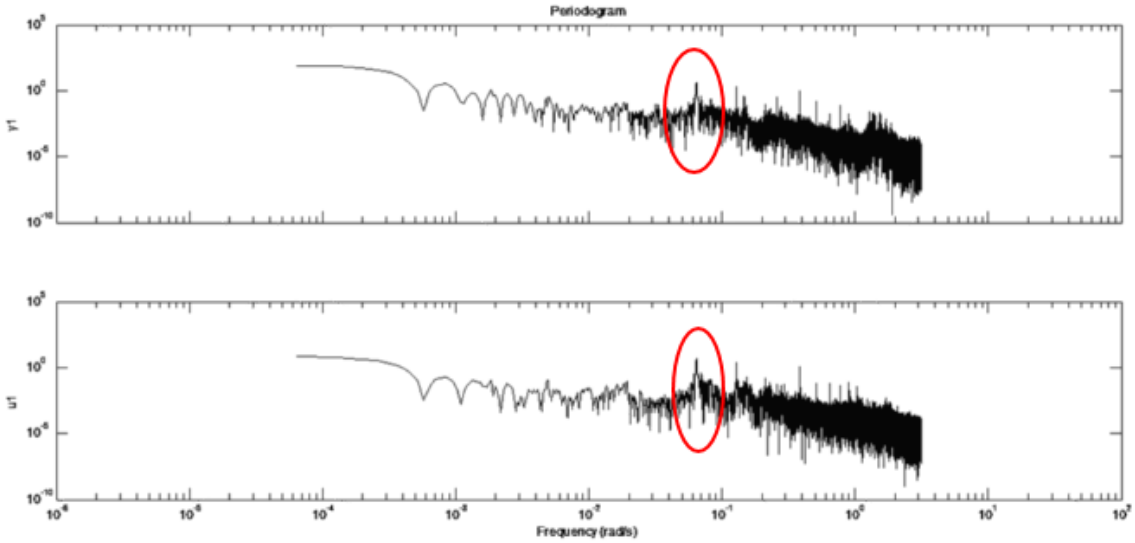

Figure 46: Power Spectrum Plot of Original Data.

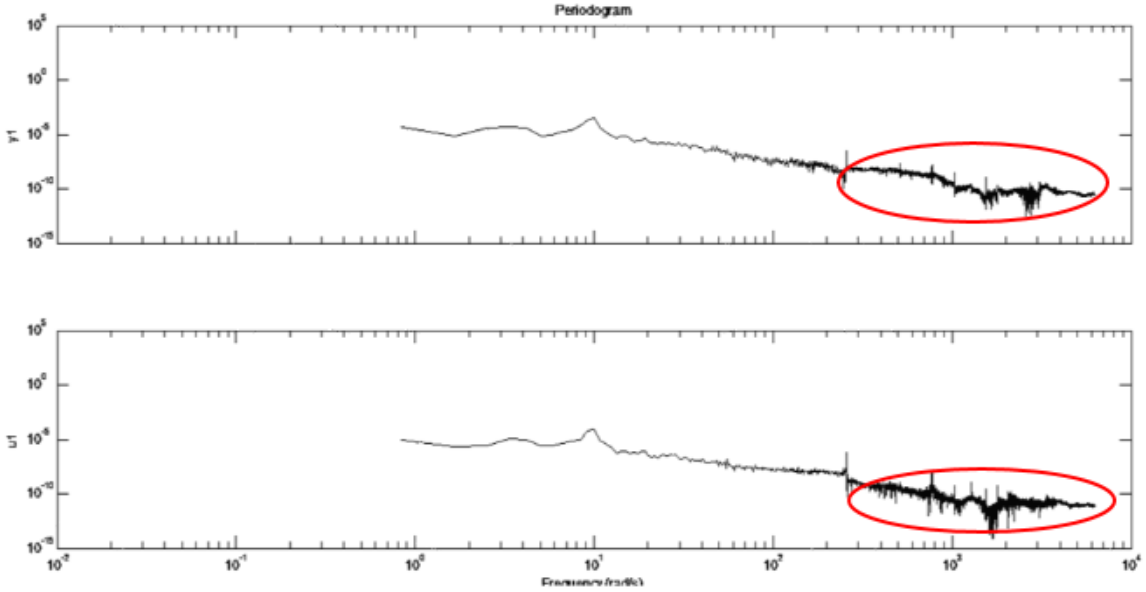

Figure 47: Power Spectrum Plot of Filtered Data.

As it can be seen from the Figure 47, higher frequencies (marked with red circle) has been filtered out. It is necessary to understand that the number of data points of the input and output should match. Therefore, while cutting off the data points it is necessary to cut off the data for both input and output (in this case from 70,000 to 85,000). Linear Output Error (OE) model is used to model the compressor dynamics. The procedure explained in Chapter 3 is followed for SI. Figure 48 consists of simulated (green plot) and

measured (black plot) output obtained by using OE model. The simulated model is obtained by using three poles and two zeros, which produced the fit of 62.35 %.

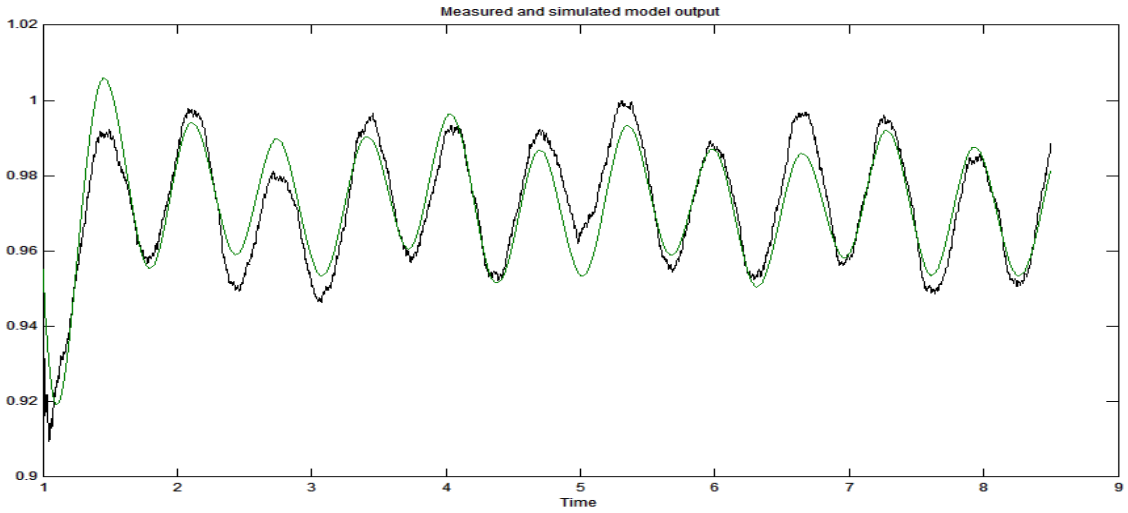

Figure 48: Measured and Estimated Output for 0.50 Flow Coefficient.

A similar approach as described above is conducted with Low Pass Butterworth filter. As described in Chapter 2, Butterworth filters are commonly used to filter noisy EMG signals. As mentioned earlier compressor's data is noisy too. This filter is programmed in Matlab™ as explained in Chapter 2. The data that is used for Bayesian filter is also used for this filter. Numerous attempts are made to optimize the filter. A filter order of 2 produced good result based on the power spectrum plot analyzation (discussed later) and the cutoff frequency is set to 50 Hz. The plot of the filtered data and the original data is shown in Figures 49 and 50. Figure 49 is the input data filtered, where the first plot is of the original data and the second plot is of the filtered data. Figure 50 is the output data filtered, where the first plot is of the original data and second of the filtered data.

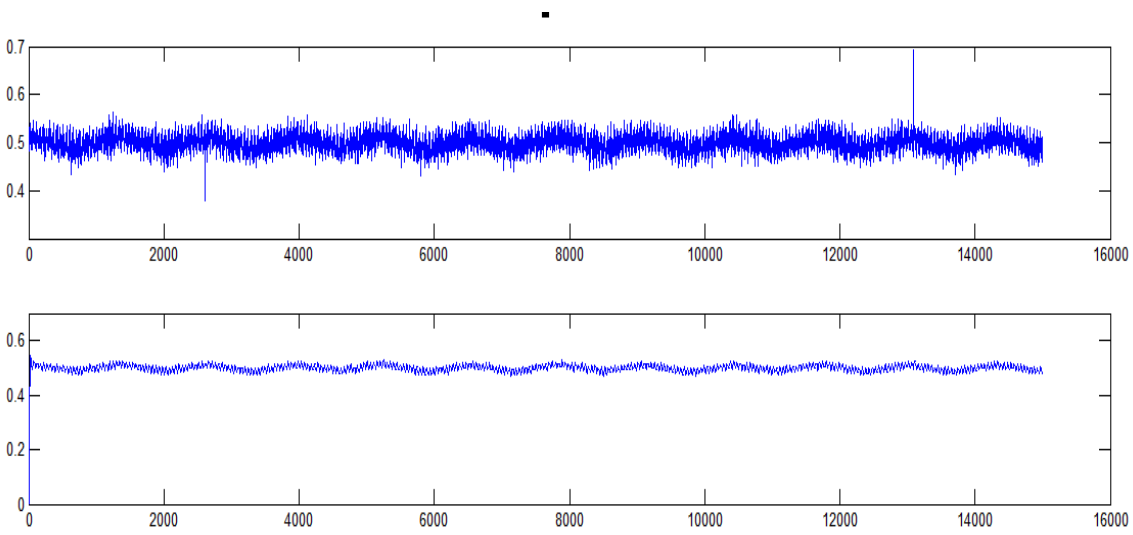

Figure 49: Input Data Filtered Using Butterworth Filter.

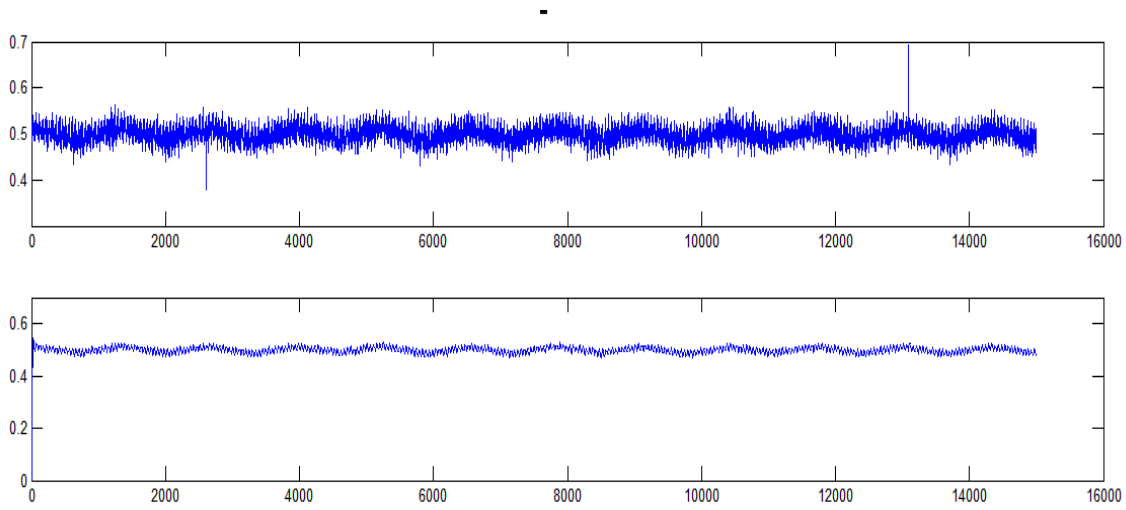

Figure 50: Output Data Filtered Using Butterworth Filter.

Once the data is filtered, the filtered data is imported to the SI toolbox. To verify the higher frequencies are taken out by filtering, a power spectrum plot is generated as described earlier. The power spectrum plot of the filtered data is shown below. It can be seen that the higher frequencies are filtered out or reduced. In Figure 51, 'u1' is the plot of input data and 'y1' is the plot of the output data.

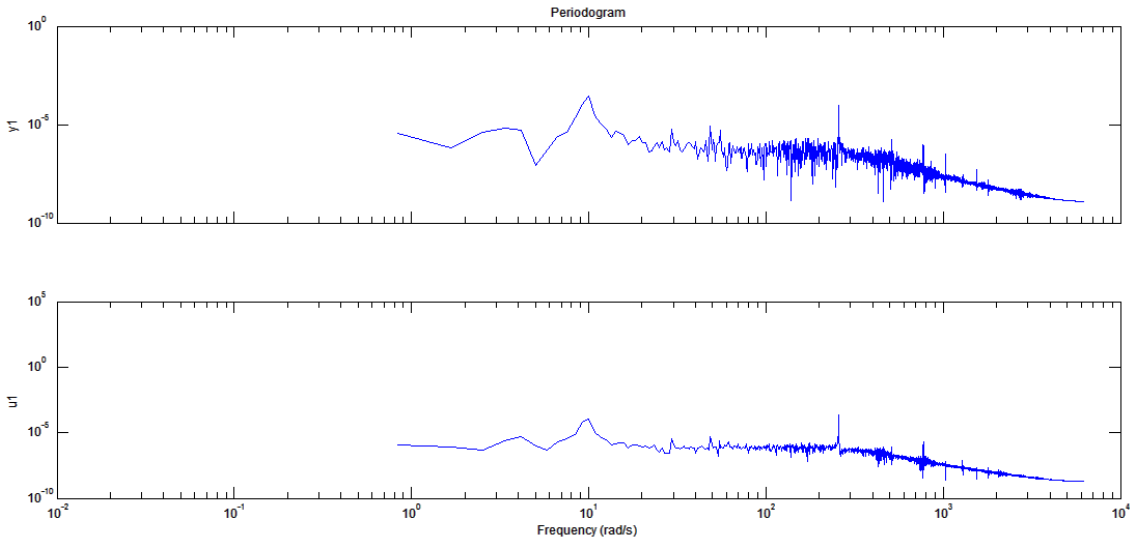

Figure 51: Power Spectrum Plot of Input/Output Data - Butterworth Filter Filtration.

As before, an output error model is used to model the compressor dynamics. The result of the SI is shown in Figure 52. The green curve is representing the simulated model and the black curve is the measured output. The simulated plot is obtained by using two zeros and three poles for the assumed transfer function model structure. As it can be seen from the plots, the fit is very poor.

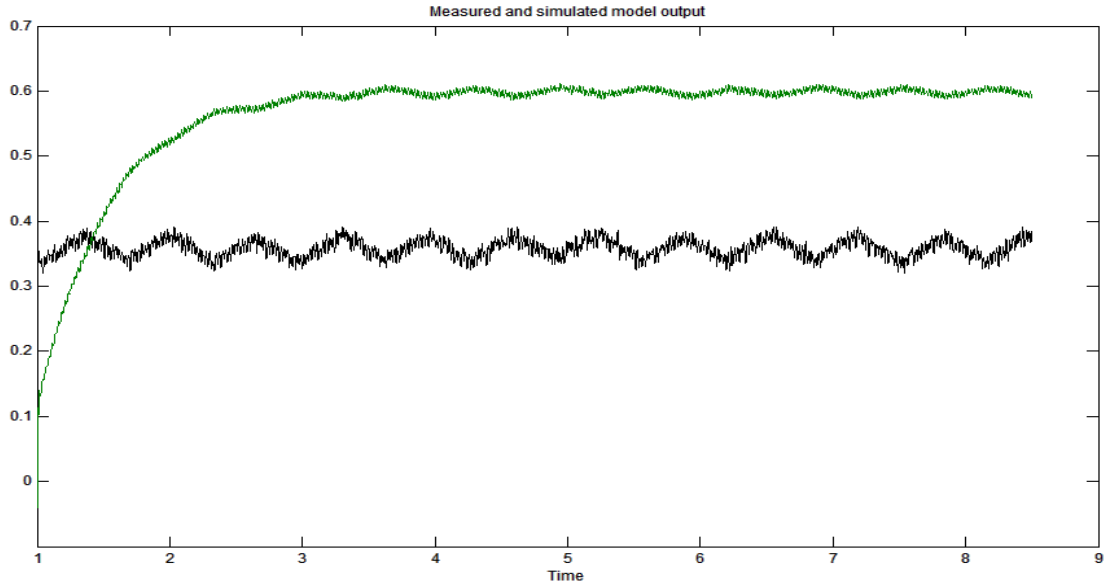

Figure 52: Measured and Simulated Model - SI Butterworth Filter.

Analyzing the SI from two separate data filtrations, a few conclusions can be drawn. The low pass Butterworth filter may not be a viable filter to be used in this case. It may be because it is unknown what should be filtered out of the data. A lack of better understanding of compressor dynamics rules out the use of a low pass filter. The output error model with three poles and two zeros is able to model the dynamics with the accuracy of 62.35% model fit when the Bayesian filter is used, but the fit is very poor with the same model when Butterworth filter is used. This implies that the low pass filter has filtered out much of the compressor dynamics or the original data doesn't contain much of the compressor dynamics. This may be due to the result of poor experiment or lack of enough excitation of the compressor to all the frequencies and a lack of sensors to capture enough dynamics of the compressor.

In order to verify this, data from different experiments are used in filtration and SI. The data for the flow coefficient of 0.50 with the sampling rate of 2000 Hz is taken. As described earlier, noise at the beginning and toward the end of the data set are taken off. The remaining data belonging to compressor dynamics is filtered using Bayesian filter and is used for SI. The plot of the filtered data is shown in the Figure 53.

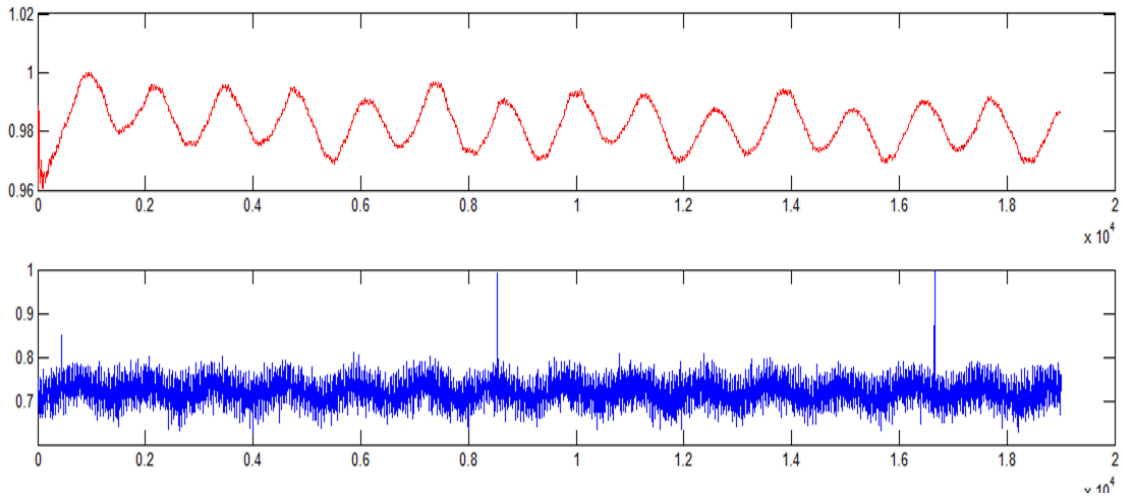

Figure 53: Input Data Filtered - Bayesian Filter.

In Figure 54, the red plot is of the filtered data and the blue plot is of the original data. To be consistent with the earlier filtration results, the parameters of the Bayesian

filter “alpha” and “beta” are set to be five and eight, respectively. Similarly, consistency is maintained on the output data filtration by keeping “alpha” and “beta” to be four and seven, respectively. The plot of the output filtered data (red) and original data (blue) is shown below. As it can be seen from both the plots the filtered data is cleaner than the original data.

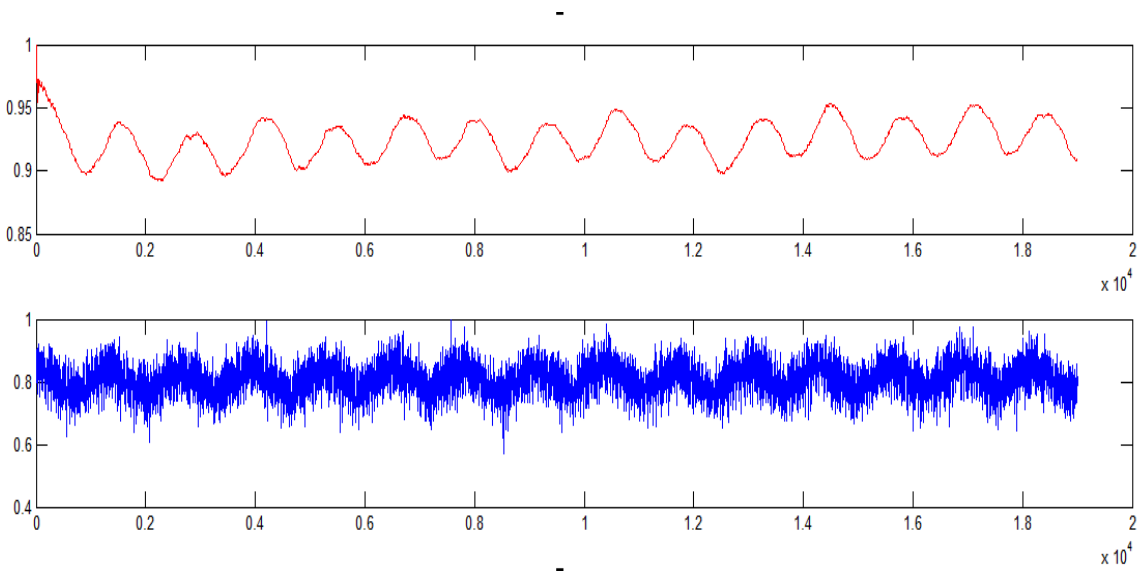

Figure 54: Output Data Filtered - Bayesian Filter.

The output of Bayesian filter is used for SI. Filtered input/output data are imported to the SI toolbox and used to model an output error model with two zeros and three poles. The SI result is shown in the Figure 55.

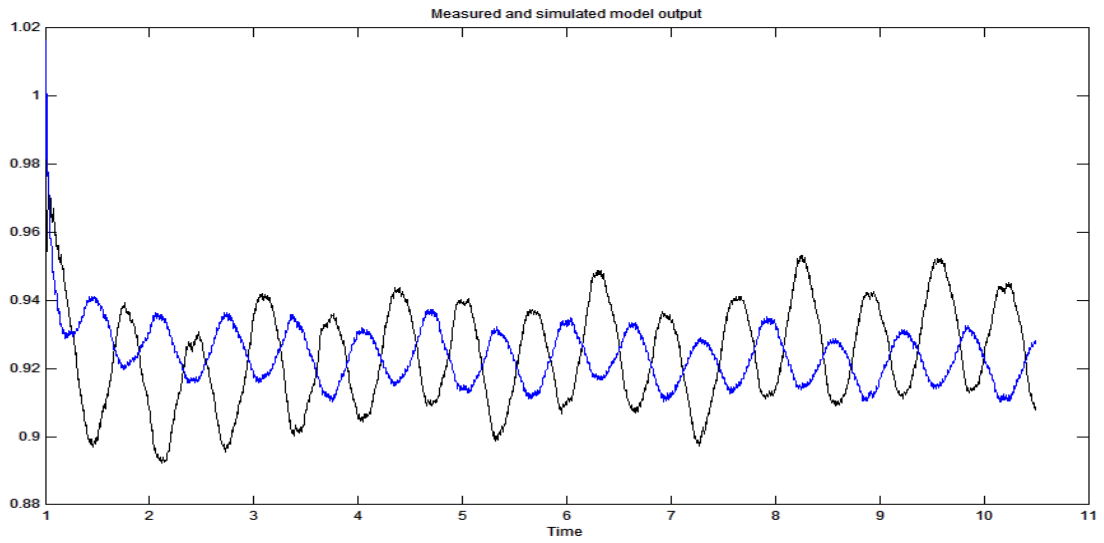

Figure 55: Measured and Simulated Output - OE231

It can be seen from the Figure 55, the simulated model fits poorly with the measured output. The calculated fit percentage is -36.38%. This makes it clear that the data does not contain actual dynamics of the compressor. If it models the dynamics, the SI results should be giving similar fit values with the same model type and same number of poles and zeros. The compressor behavior cannot be different every time at the same flow coefficient.

In order to further verify this thought, a new SI approach is investigated. As discussed in previous Chapters, the compressor is a non-linear system, therefore the experimental data contains non-linearities. Matlab™ SI toolbox has models that are able to separate linear and nonlinear part. One of the model is called Nonlinear Hammerstein - Wiener (NLHW) model. As mentioned in Chapter 3, this model is able to capture non-linearities present in the input and output data separately. It also gives the linear relation that exists between input and output data in the form of transfer function. This approach eliminates the need of data filtration this approach is also safe approach because filtration is not necessary. In this approach, the risk of filtering actual compressor dynamics along with noise is eliminated.

The same data as before from two different experiments is used for modeling the compressor dynamics using NLHW model. The input and output nonlinearities are modeled by using third order 1D polynomial and the linear model is given by an OE model with three poles and four zeros. The SI result is shown in Figure 56. The calculated fit of the simulated model (red) with measured model (black) is 15.73%. Although, the calculated fit is 15.73%, if the plots are examined closely the simulated model follows the measured model closely. The calculated fit is low may be because of the noise in the original data.

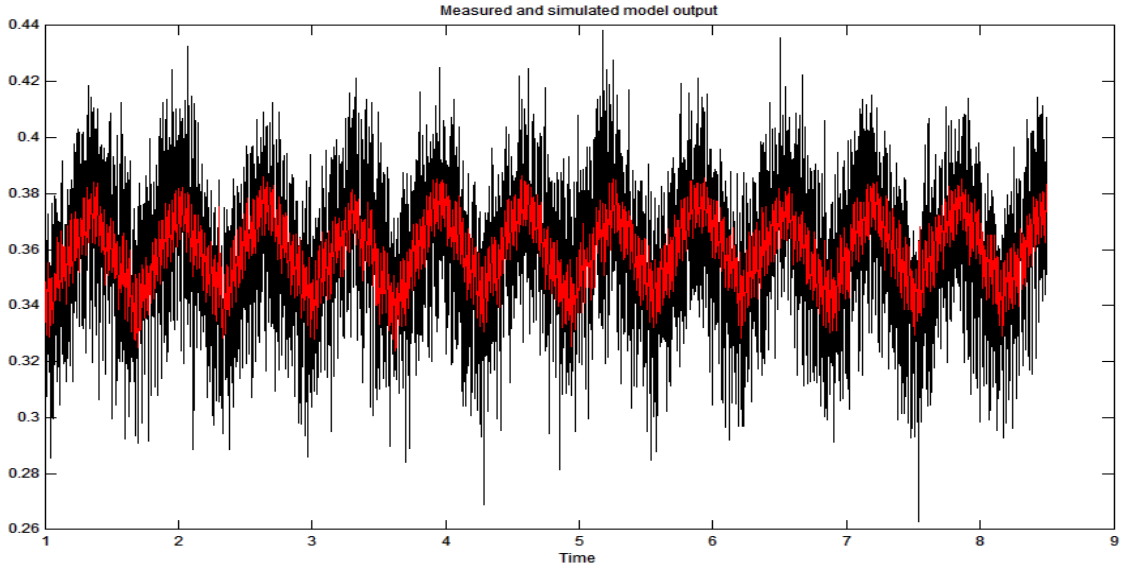

Figure 56: Measured and Simulated Model Output - NLHW (Experiment 1)

Now to verify, that the same model i.e. NLHW model with third order 1D polynomial and transfer function with three zeros and four poles produces the similar result for the data from the another experiment, a new SI for another set of data is done. The result of the SI is shown in Figure 57.

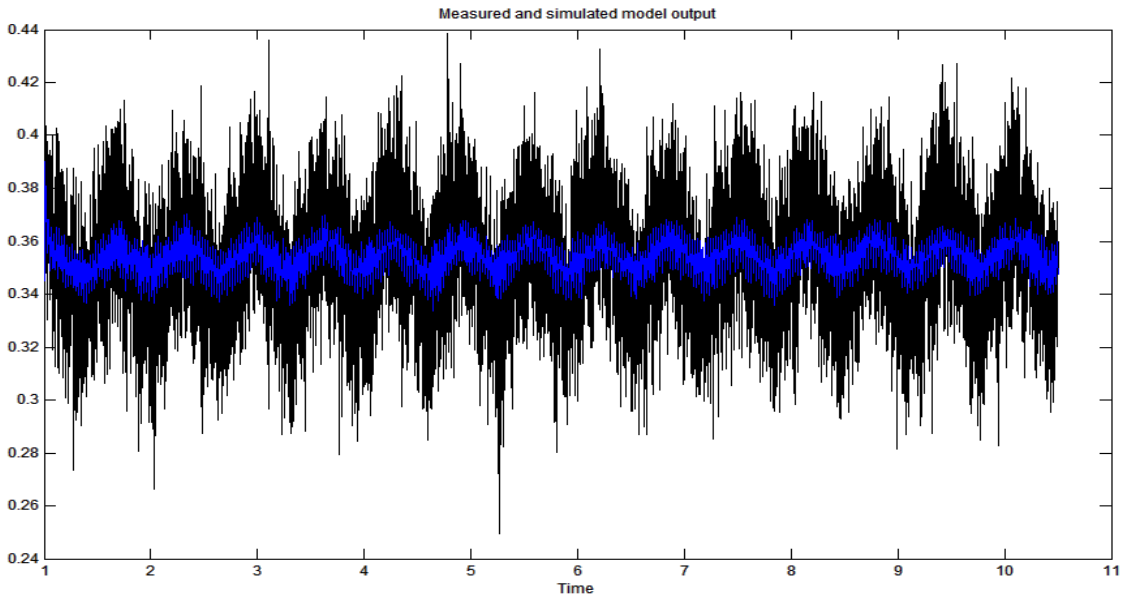

Figure 57: Measured and Simulated Output Model - NLHW (experiment 2)

In Figure 57, the blue plot is the simulated model and the black plot is the measured model output. The SI produced the fit percentage of 3.99% which is less than the fit that is calculated before with the same model. This verifies that the experimental data lacks the compressor dynamics as mentioned earlier.

In order to excite the compressor and capture enough dynamics a modification to throttle is purposed as mentioned in Chapter 4. In this modification, series of flaps will be added around the throttle and a mechanism to drive those flaps will be designed. It is decided to use high frequency solenoids to drive those flaps back and forth in axial direction. This design will excite the compressor to higher frequencies, therefore the actual dynamics of the compressor can be captured. Until this installation is done on the existing throttle, it is decided to work on the fast dynamics data.

6.2 FAST DYNAMICS SYSTEM IDENTIFICATION AND RESULT

The fast dynamics data obtained from the experiment conducted in Summer of 2014. The Hall Effect sensor is placed on the compressor casing above the blade passage to track the rotation of a particular compressor blade.

The data of fast dynamics consists of injection data which is the input to the system. A plot of the injection data for the flow coefficient of 0.52 is generated and is shown in Figure 58.

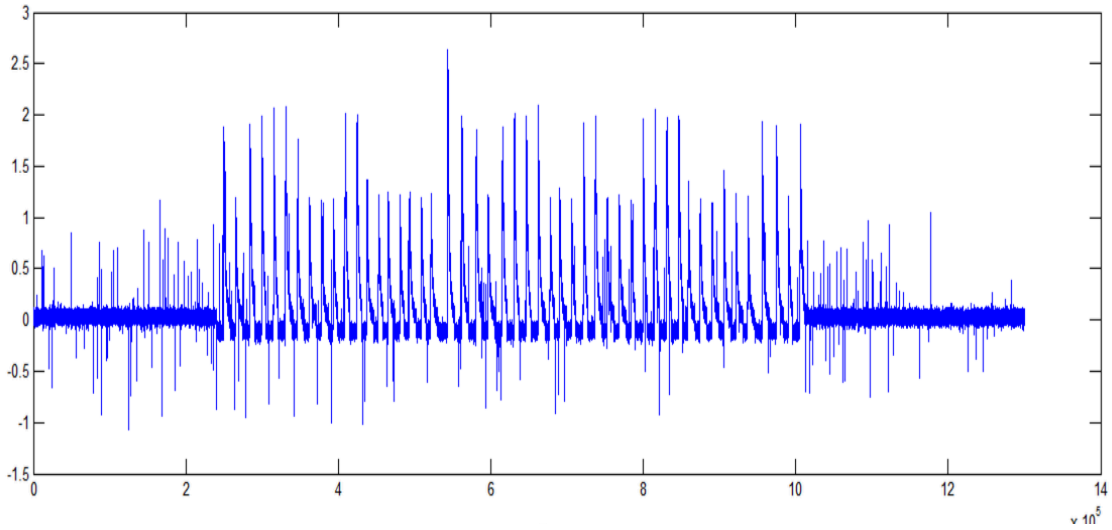

Figure 58: Injection Data Plot.

Figure 58 shows the plot of the injection pressure. At the beginning and towards the end of the plot it can be seen that the plot has no activity or the plot is just flat. In the middle of the plot, ups and downs can be seen. The area of no activity in the plot is just the time when no injection is administered. The data towards the end with no activity is separated and the plot is shown in the Figure 59.

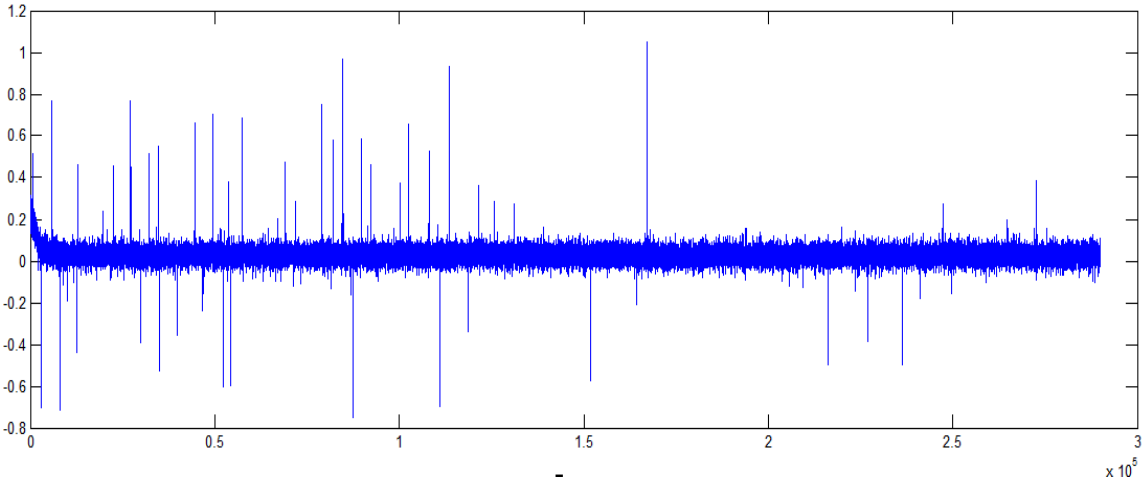

Figure 59: Plot of No Activity (no injection) Data.

In a similar way, the data of the injection pressure is plotted and is shown in Figure 60.

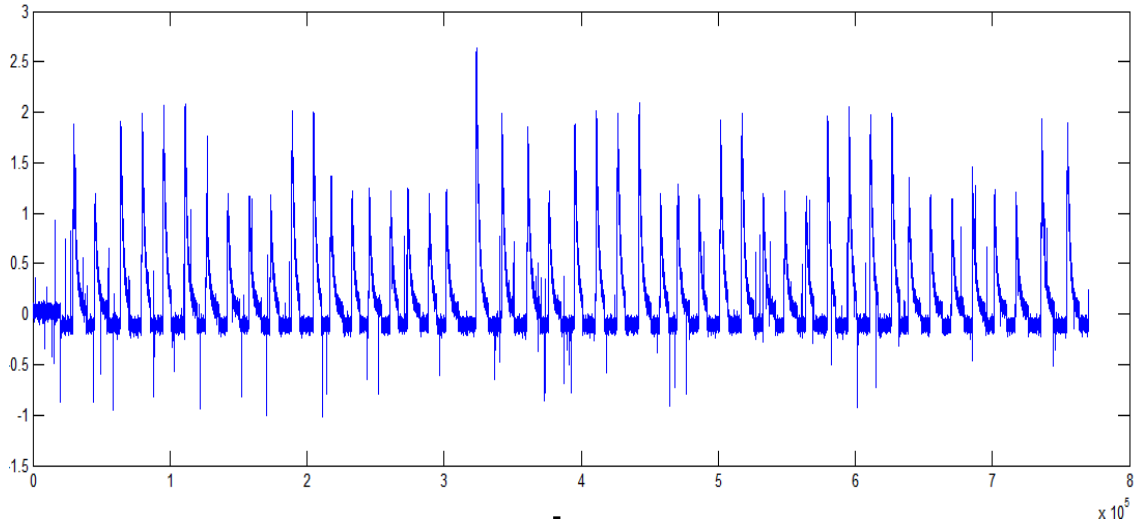

Figure 60: Injection Plot.

The correlation coefficient of the compressor data is widely used to predict the compressor behavior and to implement active controls on the system, as mentioned in Chapter 1. Therefore, for the analysis the correlation coefficient is used by using the rotation data of a particular blade and the rotation data after one rotation for that particular blade. The particular blade here means the blade that is tracked by the Hall Effect sensor. The data is recorded once the blade comes to the same position as it was during its last rotation. Therefore, the pressure data that is collected by each sensors at the time of blade passage through the Hall Effect sensors are correlated from rotation one to rotation two, rotation two to rotation three, and so on. Each passage of the blade has at-least 30 data points that are captured by the sensors. As the blade passes below the Hall Effect sensor, the magnitude of the data recorded by the Hall Effect sensor are less than one and as the blade goes away the magnitude of data recorded is more than one. A Matlab™ program is written to extract the pressure readings from the pressure sensors at the time of the blade passage through the Hall Effect sensor. Thirty data points of the pressure readings are pulled out and correlated to one after another rotation. The Matlab™ program is included in the Appendix A8.

The mean and variance of the correlation coefficients of the data from the pressure sensors corresponding to the data of no injection (no activity) is examined. The column ten data of each flow coefficient are plotted and based on the plot the data of non-injection towards the end are separated. A Matlab™ program is written (AllFlowC.m), which separates the data from sensor one through nine corresponding to the non-injection data. This program is included in Appendix A9. The correlation coefficients for each sensor are calculated. Once the correlation coefficient is calculated, mean and variance of the correlation coefficient for each sensor are calculated using the trailing window of 100 data points of correlation coefficients. This procedure is followed for the flow coefficients of 0.50, 0.52, 0.54, 0.58, and 0.60 and a plot of mean and variance is generated for all the flow coefficients. Figure 61 shows the plot of the mean (red) and the variance (blue) for the all the flow coefficients.

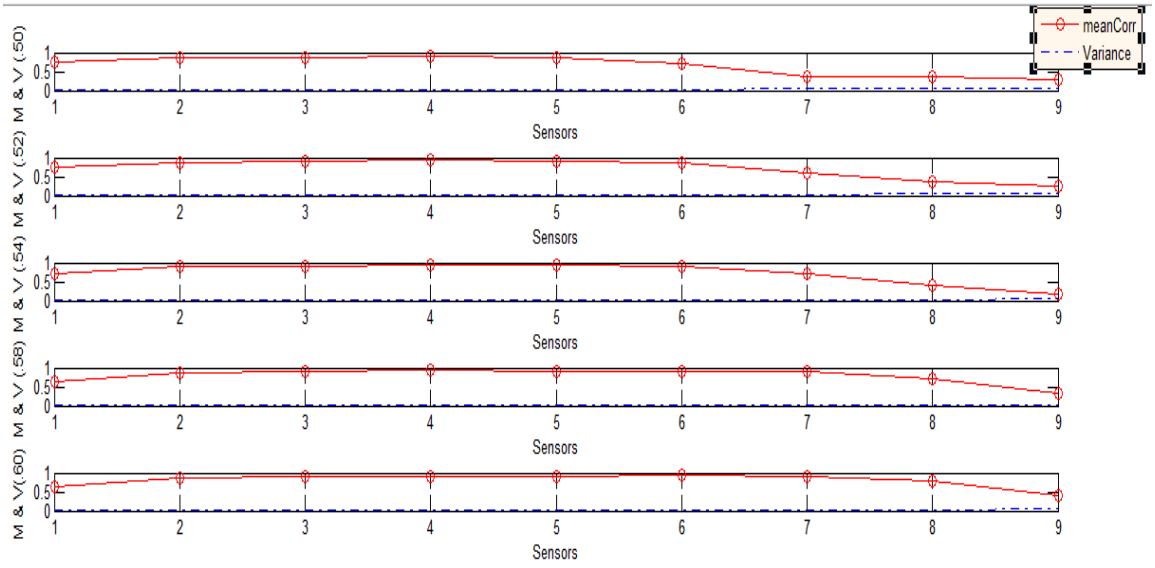

Figure 61: Plot of Mean and Variance.

As it can be seen from the plot above that the mean value of correlation coefficients for sensor 7, 8, and 9 is lower than the other sensors. As the compressor goes to stall (i.e. low flow coefficient) the mean value is even lower. From the figure above, it can be said that as the compressor goes to stall value of the correlation coefficient

decreases. It can also be seen from the Figure 61 the variance of the sensor 7, 8, and 9 increases when the compressor is near stall (near flow coefficient 0.52 and 0.50).

Correlation coefficients are calculated using the readings from each sensors and are treated as the output for the purpose of SI. There is only one Hall Effect sensor which is tracking only one blade. The other 57 blades are left out and their data is not used to calculate the correlation coefficient. As mentioned in Chapter 5, a situation of missing data arises and only one correlation coefficient is computed for 30 data points. Therefore a large number of data points are missing i.e. one data point for one rotation. Injection is the input to the system and used as the input for SI purpose. About 480 - 490 data points are captured in each rotation. Injection has no missing data but the output will have only one data point. This arises the need of filling in missing data in the output. As explained in Chapter 5, an ARX model method of predicting missing data is proposed and used to fill in the missing data.

The correlation coefficients of sensor one data are computed and the missing values are filled in with zeros. The algorithm of the written program estimates the missing data and replaces the zeros with the estimated values as stated in Chapter 5. The recorded data are in Volts (V) and has to be converted to pressure data before computing correlation coefficient. The calibration coefficients to convert pressure readings of pressure sensors in volts to pressure (Pa) is included in Table 2.

Table 2: Calibration Coefficients for Pressure Sensors

Sensors	Calibration Coefficients [Pa/V] - k
Sensor 1	968.3
Sensor 2	879.9
Sensor 3	891.9
Sensor 4	603.7
Sensor 5	599.2
Sensor 6	599.5
Sensor 7	596.8
Sensor 8	600
Sensor 9	917.7

The estimated values along with correlation coefficient are plotted and are shown in Figure 62.

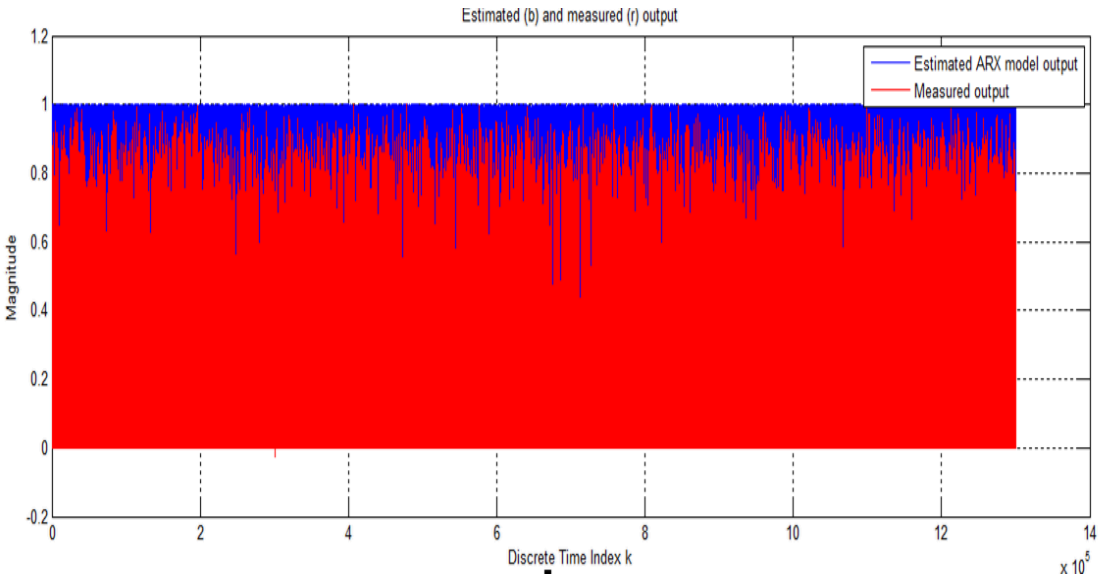

Figure 62: Estimation of Missing Data Model Order 10.

The plot shown in Figure 62 is generated by keeping the model order of 10 for flow coefficient of 0.52. The blue plot is of the estimated data and red plot is of the correlation coefficient with missing data. The plot shown in Figure 63 is for a model order of 100 and

for a flow coefficient of 0.52. The estimation is better than it is for using a model order of 10.

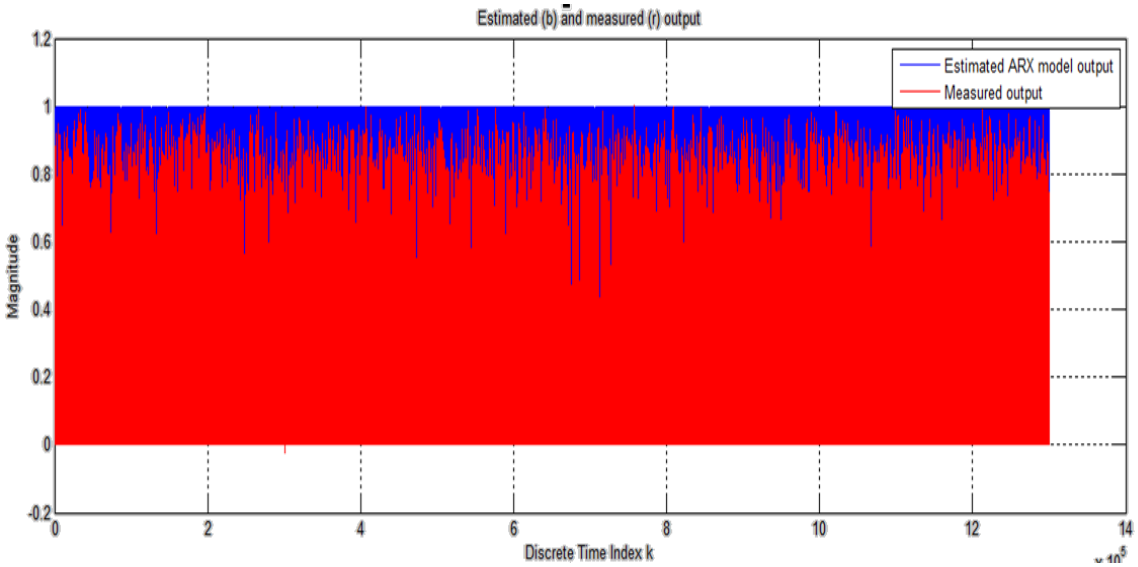

Figure 63: Estimation of Missing Data Model Order of 100.

By closely examining the estimated data plot in Figure 63, it is noted that the estimated data follows the pattern of the injection. By examining the plot as shown in Figure 64, it can be said that the estimation algorithm provides a correlation that fits the expected characteristics.

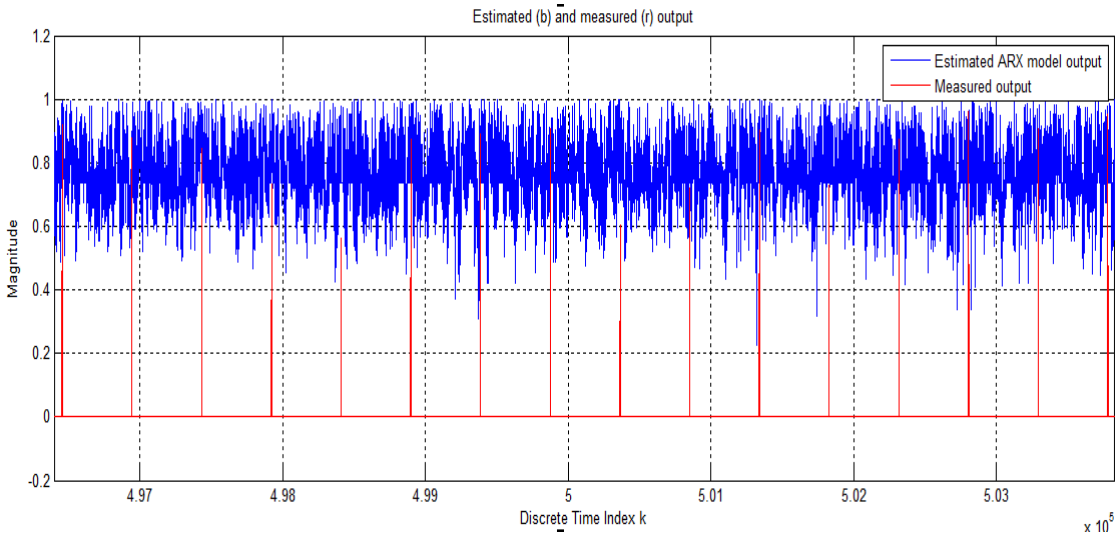

Figure 64: Estimated Data Zoomed.

For the SI, the estimated correlation coefficient (output) and the injection data (input) is used. A non - linear model is obtained using Hammerstein - Wiener model structure. The result of the SI is shown in Figure 65.

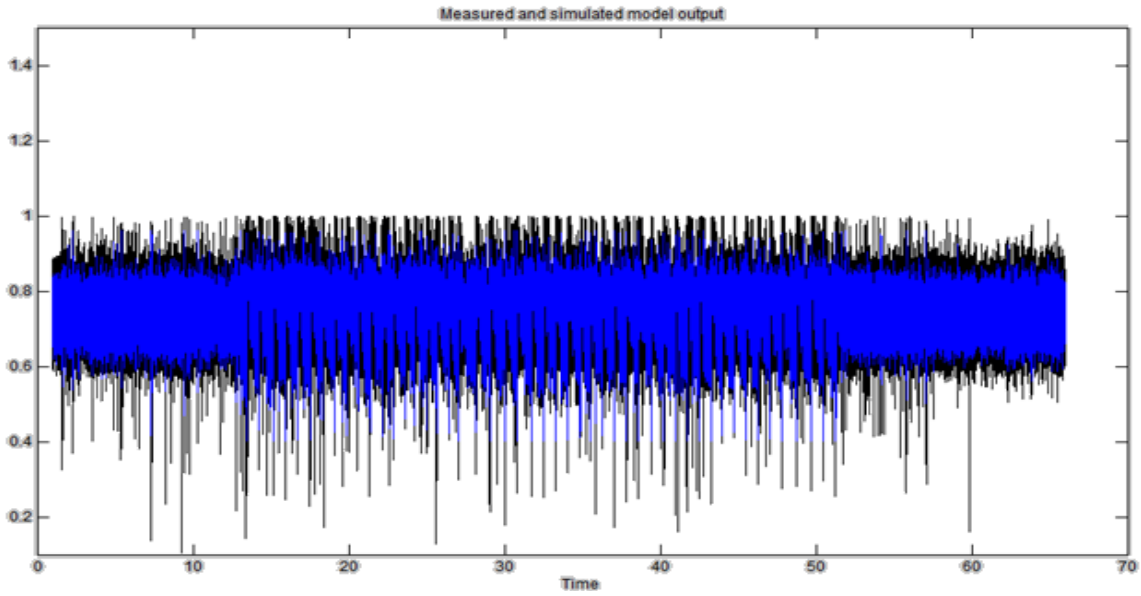

Figure 65: System Identification Result (Fit = 36%).

In the Figure 65, the black plot is of the measured output and the blue plot is of the estimated model resulted from SI. This identified model has a fit of 36% to the measured output. Three poles and three zeros are used to produce this result. Nonlinearities are modeled using a third order polynomial. This model is imported to the workspace of Matlab™ to investigate the natural frequency of the estimated model. A Matlab™ program (conver_dc.m) is written to convert the linear model of Hammerstein - Wiener to the continuous time transfer function and obtain the natural frequencies of the model. This Matlab™ is included in Appendix A10.

Estimating the missing data, SI, and obtaining the natural frequency of the identified model is done for various flow coefficients data including 0.51, 0.52, 0.54, 0.56, and 0.58. Natural frequencies of the model obtained for each SI for each flow coefficient are listed in Table 2.

Table 3: Experimental Results.

Flow Coefficient	Natural Frequency (Hz)	# of Poles	# of Zeros	# of Polynomials	Fit (%)
0.51	276.3	3	3	3	42.5
0.52	0.016	3	3	3	36
0.54	1561	4	4	3	43.6
0.56	260	3	3	3	31
0.58	94.5	4	4	3	1.14

Table 3 shows that the natural frequency of the system is not predictable, there are large fluctuations in different flow coefficients.

It is known that the natural frequency of the compressor near stall is about 17 Hz. However, the SI model for the 0.51 flow coefficient (near to stall) produced the natural frequency of 276.3 Hz. This is higher than expected and as it can be noticed from the table above, the natural frequencies are random and are not increasing as flow coefficient is increased. The natural frequency of compressor increases as the flow coefficient increases but is not reflected on the identified models. It is suspected that estimation of missing data has induced all kinds of frequencies to the estimated data. Since, 98 percent of the data is being estimated, it is obvious that the estimation cannot be as good and it would induce random frequencies in the data.

Since the above approach didn't produce very exciting results, a new approach of down sampling the input data is conducted. The down sampling here means that making the input data as the same size as of the correlation coefficient data. This eliminates the need of estimating missing data. The correlation coefficient of thirty data point is placed in the middle of the thirty data points and the corresponding data point from column 10

is extracted to make the input data equal to the output data. A Matlab™ program (Resample.m) is written that places correlation coefficients in the middle of the thirty data points and extracts the corresponding row data from the column 10. This program is included in the Appendix A11. The input/output data obtained from this procedure is extracted to the Matlab™ SI toolbox and a non - linear model is identified using Hammerstein - Wiener modeling method. The SI is done for the flow coefficient of 0.52 and the results are presented in Figure 66. The pressure data used to calculate the correlation coefficient is used from sensor 5.

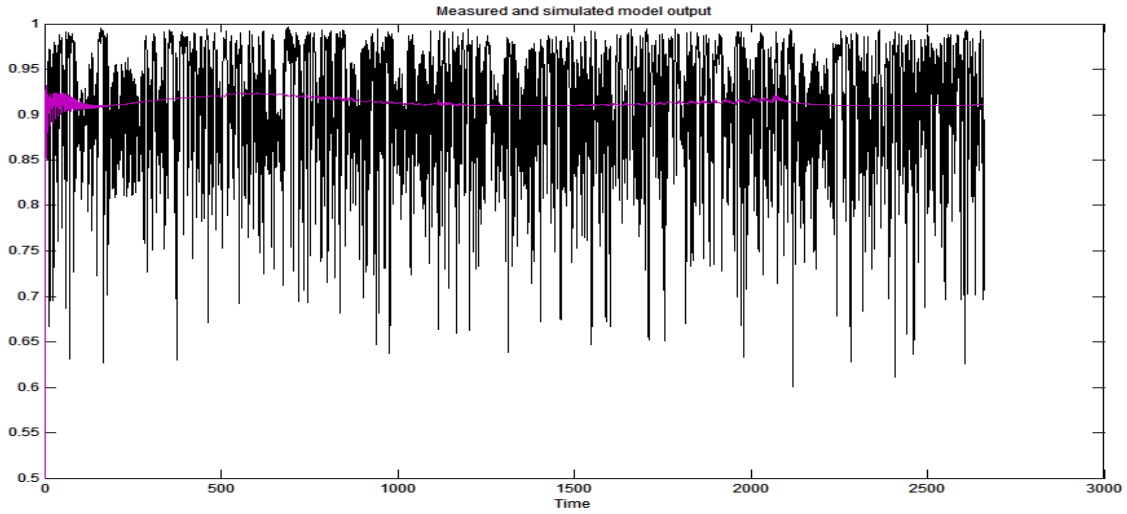

Figure 66: SI of Down Sampled Data.

In Figure 66, the black plot is the measured data and the purple plot is the simulated model obtained through SI. The simulated model is obtained by using a third order polynomial to model its non-linearities. A linear model is obtained by using two zeros and five poles and it is the best estimate obtained by trial and error. The fit % of the simulated model is only 3 %. The model is imported into the Matlab™ workspace and the natural frequency is calculated as described earlier. The natural frequency of the estimated model is calculated to be 0.0002 Hz, this is very low. Therefore, it is concluded that this approach is not valid.

Another approach proposed in order to do better modeling with SI, is given by filtering the data with a notch filter. The notch filter is designed to take out the blade passage frequency from the data and the designed filter has the central frequency of 40.9 Hz. Since the blade passing frequency is about 40.9 Hz, the central frequency of the notch filter is set to be 40.9 Hz.

The output data or pressure data of sensor no 5 for a flow coefficient 0.50 is filtered using the purposed notch filter. The correlation coefficient is calculated. The missing data is estimated and the estimated data and the sensor 10 data are extracted to the SI toolbox. SI is conducted assuming a Hammerstein - Wiener model. The result as shown in Figure 67 is obtained.

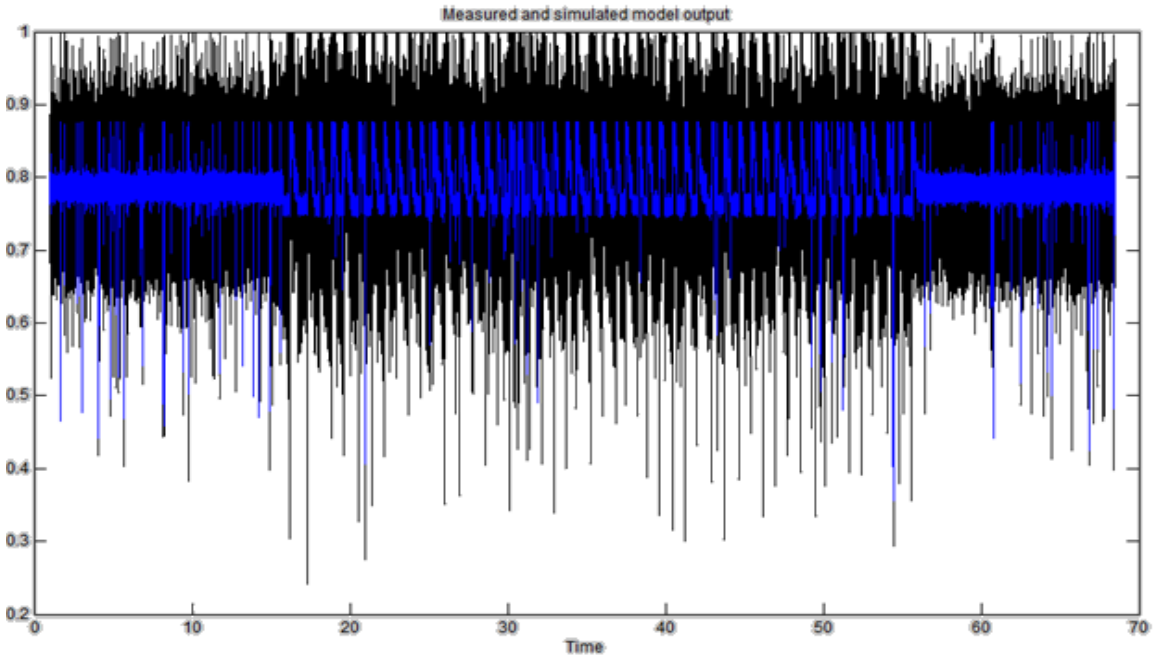

Figure 67: System Identification - Notch Filter.

In Figure (67), the black plot represents the measured output and the blue plot represents the simulated model output. The simulated model is obtained using three poles and three zeros for the linear part and a third order polynomial is used to model the non-linearities. The natural frequency of the simulated model is calculated to be 4413 Hz. The natural frequency is very high compared to what it is expected to be.

6.3 SYSTEM IDENTIFICATION ON NEW DATA

A different set of data from a different experiment is used to do the system identification and to obtain the natural frequency of the identified model. This experimental setup is described in Section 4.1 of Chapter 4. The data recorded by each pressure sensor are the voltage (V) data and this must be converted to actual pressure. The calibration coefficient for each sensor to convert the recorded data to Pascal's (Pa). The calibration coefficient for each sensors are included in Table 3.

Table 4: Calibration Coefficients of the Sensors.

Sensors	Calibration Coefficients [Pa/V] - k
Sensor 1	599.7
Sensor 2	752
Sensor 3	760.5
Sensor 4	665.4
Sensor 5	781.1
Sensor 6	757.8
Sensor 7	737.7
Sensor 8	598.7

Using the captured data in V and k, actual pressure can be easily computed using Equation (23).

$$P = (V - V_{zero})k \tag{35}$$

where,

P = actual pressure.

P_{zero} = mean zero drift of each sensor.

The SI is conducted on this new set of data in order verify if the resulting identified model has the natural frequency as of the compressor near stall. The SI is conducted with

the data of flow coefficient 0.52. The input in this case is from injection and output is the estimated correlation coefficient of sensor five. Same procedure as before is applied to obtain the correlation coefficient. The NLHW model is estimated with two poles and three zeros with the fit of 16.77%. The resulting plot of measured and simulated output models is shown in Figure 68. In Figure 68, the black colored plot is of the measured output while blue colored plot is of the simulated output.

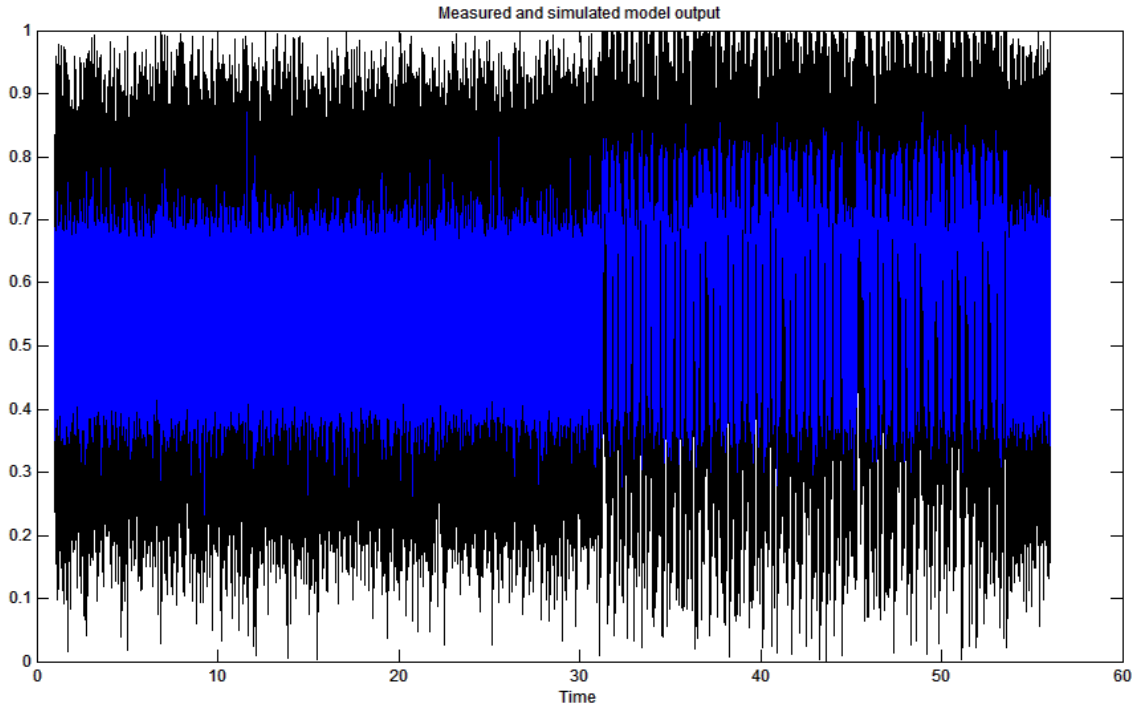

Figure 68: Measured and Simulated Model Output (0.52) - New Data.

The linear model of NLHW model is extracted and the natural frequency is computed following the method as described earlier. The computed natural frequency of the estimated model is 4268 Hz, enormously higher than what it should be. Another SI is performed on flow coefficient 0.51 data, nearest flow coefficient to stall. The NLHW model is estimated using four poles and four zeros. This combination of zeros and poles produced the highest fit of 10.49 % compared to other combination. However, this is a poor fit. The linear model of the NLHW is extracted and natural frequency of the estimated model is computed. The computed natural frequency is 350 Hz, a lot higher

than expected value. Therefore, no acceptable results were obtained from the experimental data of the compressor. The modification in the compressor is vital.

CHAPTER 7: CONCLUSION AND FUTURE WORK

This thesis discusses how the compressor works and the existing problem that arises during its operation. The work that has been done by many researchers to identify and mitigate the problems of rotating stall and surge are discussed. The theories and hypotheses in modeling the compressor dynamics forwarded by researchers are also discussed. A few ways of filtering noise out of the experimental data are explored and based on the power spectrum plot of the filtered data, the Bayesian filter is chosen to be used for this research. A problem of missing data arises because of the setup employed for the compressor. To resolve the problem of missing data, a few ways that are being used are discussed and a new algorithm based on autoregressive model with exogenous input having missing data is programmed in Matlab™. This algorithm is formulated based on least – square estimation. The estimated results of this algorithm provided a correlation that fits the expected characteristics of the compressor. However, the natural frequencies of models obtained after System Identification (SI) are random and don't match the original system's natural frequency. It is expected that the missing data estimation algorithm induced various frequencies that are not part of the compressor's dynamics causing natural frequency of the system to become random. The SI conducted with linear (output error model) and non-linear model (Non – Linear Hammerstein – Wiener model) in slow dynamics data don't produce any consistent model of the compressor. The identified models has different characteristics for each flow coefficients. This might be a cause of poor experiment or lack of equipment to excite all the modes of the compressor. The non – linear modeling of the compressor dynamics using fast dynamics data is affected by the problem of missing data. Since large number of data points are missing it is difficult to estimate those with high accuracy. However, the algorithm developed will be effective if there are less data points missing. Numerous SI with Non – Linear Hammerstein – Wiener model confirms that it is able to model compressor dynamics with higher accuracy than linear models.

To continue this research in future, it is concluded that it is necessary to make changes to the compressor in order to excite all of its modes during an experiment. In addition a few more pressure and the Hall Effect sensors must be added around the compressor to reduce the problem of missing data. With these modifications on the compressor, the data obtained from the experiments can be used to identify a model representing compressor dynamics with better accuracy. This model can be used to design controls system to be implemented on the compressor. The control will have the objective to reduce the problem of the rotating stall and surge and improve the overall efficiency of the compressor.

REFERENCES

[1] Boyce, M. P., "Axial Flow Compressors."

[2] Paduano, J. D., Greitzer, E. M., and Epstein, A. H., 2001, "Compression System Stability and Active Control," Annual Review of Fluid Mechanics, 33(1), pp. 491-517.

[3] Chi, J. N., and Paduano, J. D., 2008, "New Concepts for Active Control of Rotating Stall and Surge," American Control ConferenceSeattle, WA, pp. 2435-2442.

[4] Uddin, N., and Gravdahl, J. T., 2011, "Active Compressor Surge Control Using Piston Actuation," ASME 2011 Dynamic Systems and Control Conference and Bath/ASME Symposium on Fluid Power and Motion ControlArlington, VA, pp. 69-76.

[5] Ohta, Y., Fujita, Y., and Morita, D., 2012, "Unsteady Behavior of Surge and Rotating Stall in an Axial Flow Compressor," Journal of Thermal Science, 21(4), pp. 302-310.

[6] Saxer-Felici, H. M., Saxer, A. P., Inderbitzin, A., and Gyarmathy, G., 2000, "Numerical and Experimental Study of Rotating Stall in an Axial Compressor Stage," AIAA Journal, 38(7), pp. 1132-1141.

[7] Du, W., and Léonard, O., 2012, "Numerical Simulation of Surge in Axial Compressor," International Journal of Rotating Machinery, pp. 1-10.

[8] Greitzer, E. M., 1976, "Surge and Rotating Stall in Axial Flow Compressors, Part I: Theoretical Compression System Model," ASME J. Eng. for Power, 98(2), pp. 190-198.

[9] Chen, J. P., Hathaway, M. D., and Herrick, G. P., 2008, "Prestall Behavior of a Transonic Axial Compressor Stage via Time-Accurate Numerical Simulation," Journal of Turbomachinery, 130(4).

[10] Biollo, R., and Benini, E., 2013, "Recent Advances in Transonic Axial Compressor Aerodynamics," Progress in Aerospace Sciences, 56, pp. 1-18.

[11] Przybylko, S. J., 1997, "Active-Control Technologies for Aircraft Engines," AIAA 97-2769, IAA/ASME/SAE/ASEE 33rd Joint Propulsion Conference & Exhibit.

[12] Emmons, H. W., Pearson, C. E., and Grant, H. P., 1955, "Compressor Surge and Stall Propagation," Trans. ASME, 79, pp. 455-469.

[13] Nagano, S., and Takata, H., 1970, "Nonlinear Analysis of Rotating Stall," Institute of Space and Aeronautical Science, University of Tokyo.

[14] Moore, F. K., and Greitzer E. M., 1986, "A Theory of Post-Stall Transients in Axial Compression Systems: Part I-Development of Equations," Journal of Engineering for Gas Turbines and Power, 108, pp. 68-76.

[15] Mansoux, C. A., Gysling, D. L., Setiawan, J. D., and Paduano, J. D., 1994, "Distributed Nonlinear Modeling and Stability Analysis of Axial Compressor Stall and Surge," American Control Conference, pp. 2305-2316.

[16] McDougall, N. M., Cumpsty, N. A., and Hynes, T. P., 1990, "Stall Inception in Axial Compressors," Journal of Turbomachinery, 112(1), pp. 116-123.

[17] Day, I. J., 1993, "Stall Inception in Axial Flow Compressors," Journal of Turbomachinery, 115(1), pp. 1-9.

[18] Camp, T. R., and Day, I. J., 1998, "A Study of Spike and Modal Stall Phenomena in a Low-Speed Axial Compressor," Journal of Turbomachinery, 120(3), pp. 393-401.

[19] Hah, C., Bergner, J., and Schiffer, H. P., 2006, "Short Length-Scale Rotating Stall Inception in a Transonic Axial Compressor: Criteria and Mechanisms," ASME Turbo Expo 2006: Power for Land, Sea, and Air, pp. 61-70.
[20] Van Helvoirt, J., and De Jager, B., 2007, "Dynamic Model Including Piping Acoustics of a Centrifugal Compression System," Journal of Sound and Vibration, 302(1), pp. 361-378.
[21] Yoon, S. Y., Lin, Z., Goyne, C., and Allaire, P. E., 2011, "An Enhanced Greitzer Compressor Model with Pipeline Dynamics Included," American Control Conference, pp. 4731-4736.
[22] Pismenny, J., Levy, Y., and Modelevsky, A., 2013, "Potential Erroneous Interpretations of Pressure Signals as Beats During Established Rotating Stall," ASME Turbo Expo 2013: Turbine Technical Conference and Exposition.
[23] Hwang, Y., and Kang, S. H., 2010, "Flow and Performance Calculations of Axial Compressor Near Stall Margin," The 10th Asian International Conference on Fluid Machinery, pp. 605-615.
[24] Venturini, M., 2005, "Development and Experimental Validation of a Compressor Dynamic Model," Journal of Turbomachinery, 127(3), pp. 599-608.
[25] Morini, M., Pinelli, M., and Venturini, M., 2007, "Development of a One-Dimensional Modular Dynamic Model for the Simulation of Surge in Compression Systems," Journal of Turbomachinery, 129(3), pp. 437-447.
[26] Vo, H. D., Choon, T. S., and Greitzer, E. M., 2008, "Criteria for Spike Initiated Rotating Stall," Journal of Turbomachinery, 130(1).
[27] Tong, Z., Chen, J., Feng, L., and Nie, C., 2007, "The Self-Induced Unsteadiness of Tip Leakage Vortex and Its Effects on Compressor Stall Inception," Proc. ASME Turbo ExpoMontreal, Canada.
[28] Langston, L. S., 2013, "Blade Tips – Clearance and Its Control," Mechanical Engineering Mag., pp. 64-69.
[29] Storer, J. A., and Cumpsty, N. A., 1994, "An Approximate Analysis and Prediction Method for Tip Clearance Loss in Axial Compressors," Trans. ASME, 116(4), pp. 648-656.
[30] Gbadebo, S. A., Cumpsty, N. A., and Hynes, T. P., 2006, "Interaction of Tip Clearance Flow and Three-Dimensional Separations in Axial Compressors," Journal of Turbomachinery, 129(4), pp. 679-685.
[31] Tahara, N., Nakajima, T., Kurosaki, M., Ohta, Y., Outa, E.,and Nishikawa, T., 2001, "Active Stall Control with Practicable Stall Prediction System Using Auto-Correlation," AIAA Paper No. 2001-3623.
[32] Li, J., 2014, "Evolutions of Scales - Multi - Scale Dynamics in Axial Compressors.."
[33] wikipedia.org, "Recursive Bayesian estimation."
[34] webopedia.com, "Bayesian filter."
[35] Sanger, D. T., 2007, "Bayesian Filtering of Myoelectric Signals," Journal of Neurophysiology, 97, pp. 1839-1845.
[36] Anish, S., Kumar, P., Anugolu, M., Schoen, M. P., Urfer, A., and Naidu, D. S., "Optimization Of Bayesian Filters and Hammeristein-Wiener Models For EMG-Force Signals Using Genetic Algorithm."

[37] uic.edu, 01/23/2015, "Design of 5th Order Butterworth Low - Pass Filter Using Sallen & Key Circuit.."
[38] RecordingBlogs.com, 01/25/2015, "Gibbs phenomenon."
[39] Butterworth, S., 1930, "On the Theory of Filter Amplifiers," pp. 536-541.
[40] www.edmundoptics.com, 02/01/2015, "Notch Filter."
[41] www.thorlabs.us, 02/01/2015, "Notch Filters."
[42] www.mathworks.com, 02/02/2015, "fdesign.notch."
[43] Keesman, K. J., 2011, System Identification: An Introduction, Springer, New York.
[44] Ljung, L., 2015, "System Identification Toolbox: User's Guide," MathWorks.com, Natick, MA.
[45] MathWorks.com, 02/03/2015, "System Identification Toolbox."
[46] www.mathworks.com, 01/20/2015, "System Identification App."
[47] www.mathworks.com, 01/19/2015, "Identifying Hammerstein - Wiener Model."
[48] Kocak, T., 2007, "Sigmoid Functions and Their Usage in Artificial Neural Networks."
[49] Veitch, D., 2005, "Wavelet Neural Networks And Their Application In The Study Of Dynamical Systems," University of York, UK.
[50] Zhang, Q., and Benveniste, A., 1992, "Wavelet networks," pp. 889 - 898.
[51] Barron, T., and Kastberg, S., "Piecewise Linear Functions," The University of Georgia.
[52] www.chegg.com, 02/01/2015, "Definition of Polynomial Functions."
[53] Sterbentz, D., 2014, "Rack Pinion Assembly for Throttle Ring.."
[54] Howell, D. C., "Treatment Of Missing Data - Part 1."
[55] www.stat.columbia.edu,01/22/2015, "Missing - data imputation."
[56] Hsu, Y.-Y., 2013, "Reducing parameter estimation bias for data with missing values using simulation extrapolation," Iowa State University.
[57] Cook, J. R., and Stefanski, L. A., 1994, "Simulation - extrapolation estimation in parametric measurement error models," pp. 1314-1328.
[58] Gopaluni, R. B., Raghavan, H., and Shah, S. L., "System Identification From Multi-Rate Data."
[59] Fung, D. S. C., 2006, "Methods for the Estimation of Missing Values in Time Series,"Perth, Western Australia.

APPENDIX - A

A1 – Bayesian.m

```
% load -ascii exp1.lvm; % load the raw emg and force data
% u1=exp1(:,3); % takes the loaded data and sperates them
into three vectors
% u2=exp1(:,4);
% u3=exp1(:,5);

% Preliminary section: load data, set constants, initialize
variables
%set parameters
close all; clc;
load I_50.mat;
samplerate1 = 2000;  %samples per second
noutputs1 = 50;      %output quantization levels
ratemax1 = 1;        %rectified EMG is normalized to max
value of 1
inscale1 = 1;        %arbitrary input scaling
alpha1 = 5 / samplerate1;                %sets diffusion
rate
beta1 =  8 / (noutputs1 * samplerate1);     %sets
probability of sudden jumps

%load the data
 v1 = I_50;
 biceps1=v1;
%calculate rectified EMG after removing the mean, and
normalize
%here, we use only biceps and thus cannot approximate
extensor torque
emg1 = biceps1;% - mean(biceps1);               %essential
to remove the mean before rectifying
emg1 = inscale1 * ratemax1 * emg1 / max(emg1);       %input
prescaling to use full output range
%emg1(emg1>ratemax1) = ratemax1;                     %make
sure we don't go over
%figure
%plot(emg1)

%initialize variables
%   x is the latent variable (the driving rate)
%   MAP is the output estimate
```

```
x1 = linspace(ratemax1/noutputs1, ratemax1, noutputs1)';
%don't start with zero because requires n=0 exactly to
match
MAP1 = zeros(length(emg1),1);                    %store
the bayes estimates
g1 = [(alpha1/2) (1 - alpha1) (alpha1/2)];
%approximate spatial second derivative operator

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Following is the main section of the algorithm; steps are
numbered as in the text
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%1. Initialize p(x,0) = 1;
prior1 = ones(noutputs1,1) / noutputs1;            %start
with uniform prior

for t1=1:length(emg1) %iterate for each sample of EMG

    %2. Forward propagate p(x,t-) Å
    %           ?p(x-?,t-1)+(1-2?)p(x,t-1)+?p(x+?,t-
1)+?+(1-?)p(x,t-1);
    prior1 = filtfilt(g1, 1, prior1);             %drift
term by convolving with second derivative operator
    prior1 = beta1 + (1-beta1) * prior1;           %sets
probability of a sudden jump

    %3. Measure the rectified emg;
    emgval1 = emg1(t1);                            %if this
were online, would read a new sample here

    %4. Calculate the posterior likelihood function
    %       P(x,t) Å P(emg|x)p(x,t-);
    measurement_model1 = 2*exp(-
(emgval1).^2./(2.*(x1).^2))./(2.*pi.*x1.^2).^(1/2);   %half
Gaussian model for P(emg|x)
    posterior1 = measurement_model1 .* prior1;
%calculate posterior density using Bayes rule

    %5. Output the signal estimate MAP(x(t)) = argmax
P(x,t);
    pp1 = min(find(posterior1 == max(posterior1)));
%find the maximum of the posterior density
    if (pp1 > 1 && pp1 < length(posterior1)),
%interpolate to find the zero
```

```
        dL1 = posterior1(pp1-1) - posterior1(pp1);
        dR1 = posterior1(pp1) - posterior1(pp1+1);
        PeakIndex1 = (pp1 - .5 - (dL1/ (dR1 - dL1)));
%index runs from 1 to noutputs
    else
        PeakIndex1 = pp1;    %if maximum occurs at an
endpoint do not interpolate
    end
    MAP1(t1) = (ratemax1 / (noutputs1-1)) * PeakIndex1;
%convert index of peak value to scaled EMG value

    %6. Divide p(x,t) by a constant C so that  ° p(x,t) dx
= 1;
    posterior1 = posterior1 / sum(posterior1);
%normalize the density

    %7. Repeat from step 2;
    prior1 = posterior1;                          %prior
for next iteration is posterior from this iteration
end

%show results
figure
% subplot(2,1,1)
% plot(emg1), title('emg')
% %hold on
% %plot(torque/max(torque));
% subplot(2,1,2)
subplot(211),plot(MAP1/max(MAP1),'r');
%hold on
subplot(212), plot(emg1);
% hold off
y = MAP1/max(MAP1);
% G = spa(i);
% plot(G)
```

A2 – LowButter.m

```
% creating a low pass butterworth filter
% band pass filter can be created tooo
% type 'help butter'
% for flow coefficient
close all
n = 2; % filter order
w = 2000; % sampling frequency
fc = 50; % cutoff frequency
wn = fc/(w/2); % normalizing cutoff frequency

[b a] = butter(n,wn,'low');
y = filter (b,a,i_50);% y is the filtered output

figure
subplot(211),plot(i_50);
subplot(212), plot(y);

% For pressure rise coefficient

n = 2; % filter order
w = 2000; % sampling frequency
fc = 50; % cutoff frequency
wn = fc/(w/2); % normalizing cutoff frequency

[c d] = butter(n,wn,'low');
yy = filter (c,d,ii_50);% y is the filtered output

figure
subplot(211),plot(ii_50);
subplot(212), plot(yy);
```

A3 – NotchFilter.m

```
% Notch Filter

d = fdesign.notch('N,F0,Q',4,40.9,10,20000);
H = design(d);

load dynamic_data_051.dat;
datav=dynamic_data_051;

datav = filter(H,datav);
```

A4 – NLhw.m

```
% This program estimates Nonlinear Hammerstein Wiener Model
at the Command
% line
% 12/31/2014
% http://www.mathworks.com/help/ident/ug/identifying-
hammerstein-wiener-models.html

clear; clc; close all;
load i_50.mat; % flow coefficient (input)
load ii_50.mat; % pressure rise coefficient (output)

z = iddata(i_50, ii_50, 0.0005);

[R C]= size(z);

ze = z(1:7500); %taking half of the available data to
create the model

zv = z(7501:R); % using rest of the data to verify the fit.

% all data can be used to create the model and all of it
can be used to
% verify.

% Now estimating several models using differnt model
orders, and
% nonlineriy setting

m1 = nlhw(ze,[1 3 1],poly1d(5),poly1d(4)); % first model

m2 = nlhw(ze,[3 3 1],poly1d(2),poly1d(5)); % first model

% now compare the resulting models by plotting the model
outputs on the top
% of the measured output

compare(zv,m1,m2) % displays the plot and fit percentage.

% Use commands below to display the Input/output
nonlinearites and linear
% model

ON = m1.OutputNonlinearity % output nonlinearity
```

```
IN = m1.InputNonlinearity % input nonlinearity

LM = m1.LinearModel % linear model
```

A5 – Ultimag_4EM.pdf

Ultimag® Size 4EM

ROTARY Ultimag®

Part Number: 197124-0XX

All catalog products manufactured after April 1, 2006 are RoHS Compliant

Specifications

Dielectric Strength	1000 VRMS (23 awg); 1200 VRMS (24-33 awg)
Recommended Minimum Heat Sink	Maximum watts dissipated by the Ultimag are based on an unrestricted flow of air at 20°C, with the Ultimag mounted on the equivalent of an aluminum plate measuring 6-1/4" square by 1/8" thick (15.9 cm sq. x 0.32 cm)
Thermal Resistance	7.6°C/watt with heatsink; 15.0°C/watt without heatsink
Rotor Inertia	8.43 x 10^{-7} (kgm^2)
Peak Torque Rating (Tp)	45 oz.in. (0.32 Nm)
Power Input	145 watts (stalled at Tp; 25°C; Pp)
Number of Phases	1
Static Friction (Tf)	1 oz.in. max. (7mNm)
-3dB Closed Loop	78 Hz
Maximum Winding	180°C
Number of Poles	6
Weight:	7.6 oz. (215 gms)
Dimensions:	Ø1.625" x 1.04" L (Ø41.66 mm x 26.3 mm L) See page B10.

Performance

Maximum Duty Cycle	100%	50%	25%	10%
K_M (oz-in/√watt)	5.8	5.1	4.6	4.3
Maximum ON Time (sec) when pulsed continuously[1]	∞	40	15	4
Maximum ON Time (sec) for single pulse[2]	∞	108	34	9
Typical Energize Time (msec)[3]	6	5	4.5	3.5
Watts (@ 20°C)	14.5	29	58	145
Ampere Turns (@ 20°C)	510	721	1020	1613

Coil Data

awg (0XX)[4]	Resistance (@20°C)	# Turns[5]	VDC (Nom)	VDC (Nom)	VDC (Nom)	VDC (Nom)
23	0.71	104	3.2	4.5	6.4	10.1
24	1.54	174	4.7	6.7	9.4	14.9
25	2.15	195	5.6	7.9	11.2	17.6
26	3.01	219	6.6	9.3	13.2	20.9
27	5.78	328	9.2	12.9	18.3	28.9
28	8.09	368	10.8	15.3	21.7	34.3
29	14.40	515	14.5	20.4	28.9	45.7
30	20.11	575	18.9	24.2	37.7	59.6
31	34.40	774	22.3	31.6	44.6	71.0
32	56.60	1008	28.7	40.5	57.0	91.0
33	91.40	1288	36.0	51.5	73.0	115.0

How to Order

Add the coil awg number (0XX) to the part number (for example: to order a 25% duty cycle rated at 18.3 VDC, specify 197124-027).

Please see www.ledex.com (click on Stock Products tab) for our list of stock products available through our North American distributors.

[1] Continuously pulsed at stated watts and duty cycle
[2] Single pulse at stated watts (with coil at ambient room temperature 20°C)
[3] Typical energize time based on no load condition. Times shown are for half of full rotary stroke starting at center-off position.
[4] Other coil awg sizes available — please consult factory
[5] Reference number of turns

WARNING: Exposed Magnet may affect pacemakers. In the event a product unit's magnet is exposed due to product disassembly, Pacemaker Wearers should distance themselves 10 feet from exposed magnet.

All specifications subject to change without notice.

A6 – Newestmissdata_mod.m

```
%                  estmissdata.m
%
%                  Marco P. Schoen/Sujan Prasai
%                  December 10, 2014
%                  version 1.0
%
% Estimating ARX model parameters with missing data (output
data is
% incomplete, input data is complete
%
_____________________________________________________________
__________

clear;clc; close all;

% 1. Define models
Ts=0.00005; % Sampling time or 20000 Hz

%[n,n]=size(An);[n,nu]=size(Bn); % n ... number of states,
nu ... number of inputs
%[ny,n]=size(Cn); % ny ... number of outputs

load corcf.mat

yc = corcf(:,1); % output - sensor 1 (correletion coef.)
yc(isnan(yc))=0;
ycmean=mean(yc);n=2;
ny = 1; % number of outputs

nu = 1; % number of inputs

load dynamic_data_052.dat;

data = dynamic_data_052;

% implementing notch filter

% d = fdesign.notch('N,F0,Q',4,2372.2,10,20000);
% H = design(d);
%
% data = filter(H,data);

u = data(:,9); % input - injection
```

```
[ru cu] = size(u);

%____________
% Now 'NaNs' for the missing data on 'corcf'

% B = zeros(488,1); % making a column vector with NaNs to
fill in for missing data
% z= zeros(1,1);
% newy = [];
% [row col] = size(yc);
% for i = 1:row
%     z(i) = yc(i,:);
%     newy = cat(1,newy,z(i),B);
% end

count=1;
newy=zeros(L,1);
for i=1:L
    x=data(i,11);
    if x<1
        if data(i+1,11)>1
            newy(i-15,:) = corcf(count,1);
            count=count+1;
            newy(i,:)=0;
        else
            newy(i,:)=0;
        end
    else
        newy(i,:)=0;
    end
end

yo = newy; % assign yo to the NaN filled column vector.

% yo should have equal number of row numbers as of the
column 10
%yo = yo(1:ru,:);

% 3. Construct matrices for estimation
p1=input('Model order: ');
%Pg=input('Probability of available data: ');
Pg = 1/58; % one blade data out of 58 blades.
ymax=max(yo);ymean=mean(yo);dpg=round(100*Pg);
y1=(yo-ymean)/ymax;
counter1=1;counter2=1;
%y1 = y1(1:100000,:); % reducing no of datas
[L C] = size(y1);
```

```
% for k=1:100:L
%     for j=1:dpg
%         y(counter1,:)=y1(j+k-1,:);
%         counter1=counter1+1;
%     end
% end

y=y1;
[L2,dc]=size(y);
L=L2;
sum=zeros(2*p1,2*p1);lphi=zeros(2*p1,ny);
for k=p1+1:L
    for i=1:p1
        lphi(i,:)=y(k-i,:);
    end;
    for i=1:p1
        lphi(p1+i,:)=u(k-i,:);
    end;
    lphilphit=lphi*lphi';
    sum=sum+lphilphit;
end;

for i=1:p1
    for j=1:p1
        sum(i,j)=sum(i,j)*(1/(Pg^2));
        sum(i,j+p1)=sum(i,j+p1)*(1/Pg);
        sum(i+p1,j)=sum(i+p1,j)*(1/Pg);
    end;
end;
suminv=pinv(sum)/L;
sum2=zeros(2*p1,1);%lphiy=sum2;
for k=p1+1:L
    for i=1:p1
        lphi(i,:)=y(k-i,:);
    end;
    for i=1:p1
        lphi(p1+i,:)=u(k-i,:);
    end;
    lphiy=lphi*y(k,:);
    sum2=sum2+lphiy;
end;
for i=1:p1
    sum(i,:)=sum2(i,:)/(Pg^2);
    sum(i+p1,:)=sum2(i+p1,:)/Pg;
end;
sum2=sum2/L;
```

```
Theta=suminv*sum2; % Theta = [a1 a2 ... b1 b2 ... b(p1)]'

% 4. Check output of estimated model

% Theta = [a(1) a(2) a(3) ... b(p1)], no D matrix
identification
% they need to be in Thetaarmax=[b(0) a(0) b(1) a(1) ...
b(q) a(q)]'
a=zeros(1:ny,1,nu,p1+1);b=a;Tha=Theta;count=1;
for i=2:p1+1
    a(:,:,i)=Tha(i-1:(i-1)*ny,:);
end;
for i=2:p1+1
    b(:,:,i)=Tha(i+p1-1:(i+p1-1)*ny,:);
end;
count=1;
for i=1:p1+1
    Thetan(count:count*ny,:)=b(:,:,i);
    count=count+ny;
    Thetan(count:count*ny,:)=a(:,:,i);
    count=count+ny;
end

yestarx=arxsim(n,nu,u,y,L,p1,Thetan);
yest=yestarx/(max(yestarx));yest=yest+ycmean;

yn=y/(max(y));yn = yn+ymean;
[Ly,dc]=size(yest);
for j=1:Ly %truncating 0 and 1 to make it mean value as
best guess
    if yest(j)>1
        yest(j)=ycmean;
    elseif yest(j)<0
        yest(j)=ycmean;
    else
    end
end;
figure;plot(yest);hold;
plot(yn,'r');grid;title('Estimated (b) and measured (r)
output');
xlabel('Discrete Time Index
k');ylabel('Magnitude');legend('Estimated ARX model
output','Measured output');
```

A7 – arxsim.m

```
function yestarx=arxsim(n,no,u,y,L,q,Thetaarx);
%Thetaarx=[b(0) a(0) b(1) a(1) ... b(q) a(q)]
yestarx=zeros(L,no);yest_elarx=0;
L0 = L-200;
for i=L-L0:L
    for j=1:q
        m=2*j;
        %yest_elarx=yest_elarx+Thetaarx(m-1,1)*y(i-
j,:)+Thetaarx(m,1)*u(i-j,:);
        % using Theraarx = [a(1) b(1) a(2) b(2) ... a(p1)
b(p1)], no D matrix
        yest_elarx=yest_elarx+Thetaarx(m-1,1)*u(i-
j+1,:)+Thetaarx(m,1)*y(i-j+1,:);
   end;
   yestarx(i,:)=yest_elarx;
   yest_elarx=0;
end;
figure
plot(yestarx,'b');hold;plot(y,'k');grid;title('ARX (b)
estimate and measured output (k)');
xlabel('Time (s)');legend('ARX model output','True measured
output');
```

A8 – CorrCoef.m

```
clear;clc;
% Conversion factors [Pa/V]:
con(1)=968.3;con(2)=879.9;con(3)=891.9;con(4)=603.7;con(5)=
599.2;
con(6)=599.5;con(7)=596.8;con(8)=600;con(9)=917.7;
% read data in
load dynamic_data_054.dat;
datav=dynamic_data_054; % data in [V]
[L,ca]=size(datav);
Ts=1/20000;

% correct for drift
load dynamic_data_zero.dat;
drift=dynamic_data_zero;
[Ld,col]=size(drift);
for i=1:col
    driftmean(i)=mean(drift(:,i));
    datavm(:,i)=datav(:,i)-driftmean(i);% data with drift
adjusted
end

for i=1:9
    datapm(:,i)=datavm(:,i)*con(i); % data in [Pa]
end;
datapm(:,10:12)=datav(:,10:12);

% % with the program below we can calculate correlation
coefficient of each
% % sensors for all the rotations.
%
% x = datav(:,11);
%
% s = find(x<1);
%
% xd = datapm(s,1:9);

counter = 1;
xd = zeros(34,3000,12);
for i = 1:L
    x = datav(i,11);
    if x < 1
       if datav(i + 1,11) > 1
           xd(1:34,counter,:) = datapm(i-34+1:i,:);
           counter = counter+1;
```

```
        else
        end
    else
    end
end

% Compute correlation coefficient
for sensor=1:9
    for j=2:counter % count is the number of blade passages
        xcf=corrcoef(xd(:,j-1,sensor),xd(:,j,sensor));
        corcf(j,sensor)=xcf(2,1);
    end
end

figure
subplot(3,3,1);plot(corcf(:,1));title('Correlation
Coefficient for Sensor 1')
xlabel('Rotation');ylabel('Magnitude');grid;
subplot(3,3,2);plot(corcf(:,2));title('Correlation
Coefficient for Sensor 2')
xlabel('Rotation');ylabel('Magnitude');grid;
subplot(3,3,3);plot(corcf(:,3));title('Correlation
Coefficient for Sensor 3')
xlabel('Rotation');ylabel('Magnitude');grid;
subplot(3,3,4);plot(corcf(:,4));title('Correlation
Coefficient for Sensor 4')
xlabel('Rotation');ylabel('Magnitude');grid;
subplot(3,3,5);plot(corcf(:,5));title('Correlation
Coefficient for Sensor 5')
xlabel('Rotation');ylabel('Magnitude');grid;
subplot(3,3,6);plot(corcf(:,6));title('Correlation
Coefficient for Sensor 6')
xlabel('Rotation');ylabel('Magnitude');grid;
subplot(3,3,7);plot(corcf(:,7));title('Correlation
Coefficient for Sensor 7')
xlabel('Rotation');ylabel('Magnitude');grid;
subplot(3,3,8);plot(corcf(:,8));title('Correlation
Coefficient for Sensor 8')
xlabel('Rotation');ylabel('Magnitude');grid;
subplot(3,3,9);plot(corcf(:,9));title('Correlation
Coefficient for Sensor 9')
xlabel('Rotation');ylabel('Magnitude');grid;
```

A9 – AllFlowC.m

```
% Preparing data for all Flow coef.
% First plot column 10
% Obtain data where there is no injection
% Cut the corresponding data from all sensors 1-9.
% compute correlation coef. for each column 1-9 for each
rotation
% _____________________________________________
% compute Mean and Sample Variance for Each Corr. Coef.
using trailing
% window of 10 - 100 data point.
% _____________________________________________________

clear; clc; close all

load dynamic_data_050.dat
datav = dynamic_data_050;
[L,ca]=size(datav);
Ts=1/20000;

% Conversion Factors [Pa/V]
con(1)=968.3;con(2)=879.9;con(3)=891.9;con(4)=603.7;con(5)=
599.2;
con(6)=599.5;con(7)=596.8;con(8)=600;con(9)=917.7;

% correction for drift
load dynamic_data_zero.dat;
drift=dynamic_data_zero;
[Ld,col]=size(drift);
for i=1:col
    driftmean(i)=mean(drift(:,i));
    datavm(:,i)=datav(:,i)-driftmean(i);% data with drift
adjusted
end

for i=1:9
    datapm(:,i)=datavm(:,i)*con(i); % data in [Pa]
end;
datapm(:,10:12)=datav(:,10:12);

x = datav(:,10);

% figure
% plot(x)
```

```
% Now separate data of non - injection by looking at the
plot

y = x(1.1*10^6:end); % cut off datas.

% figure
% plot(y) % to verify that injection data are excluded

nonInj = datapm(1.1*10^6:end,1:12); % cut off datas.

[R C] = size(nonInj);

counter = 1;
xd = zeros(30,3000,12);
for i = 1:R
    x = nonInj(i,11);
    if x < 1
       if nonInj(i + 1,11) > 1
           xd(1:30,counter,:) = nonInj(i-30+1:i,:);
           counter = counter+1;
       else
       end
    else
    end
end

% Compute correlation coefficient
for sensor=1:9
    for j=2:counter % counter is the number of blade
passages
        xcf=corrcoef(xd(:,j-1,sensor),xd(:,j,sensor));
        corcf(j,sensor)=xcf(2,1);
    end
end


% compute Mean and Variance of the corr. coef. using a
trailing window of
% 10 - 100 data point
% In this case choosing 100 coef.

window = corcf(end - 101: end - 1,:); % Window of 100 corr.
coef.

% calculate Mean

for i=1:9
```

```
   meanCorr050(i)=mean(window(:,i));
end

% calculating Variance

Variance050 = var(window);
Colu = 1:1:9;

% Now Plotting Mean and Variance

figure
subplot(5,1,1); title('Plot of Mean and Variance')
plot(Colu,meanCorr050,'-ro',Colu,Variance050,'-
.b'),legend('meanCorr','Variance')
xlabel('Sensors');ylabel('M & V (.50)');

% Now for Flow coef. .52
%_____________________________________________________________
_____________
%_____________________________________________________________
_____________

clear;
load dynamic_data_052.dat
datav = dynamic_data_052;
[L,ca]=size(datav);
Ts=1/20000;

% Conversion Factors [Pa/V]
con(1)=968.3;con(2)=879.9;con(3)=891.9;con(4)=603.7;con(5)=
599.2;
con(6)=599.5;con(7)=596.8;con(8)=600;con(9)=917.7;

% correction for drift
load dynamic_data_zero.dat;
drift=dynamic_data_zero;
[Ld,col]=size(drift);
for i=1:col
    driftmean(i)=mean(drift(:,i));
    datavm(:,i)=datav(:,i)-driftmean(i);% data with drift
adjusted
end

for i=1:9
    datapm(:,i)=datavm(:,i)*con(i); % data in [Pa]
end;
datapm(:,10:12)=datav(:,10:12);
```

```
x = datav(:,10);

% figure
% plot(x)

% Now separate data of non - injection by looking at the
plot

y = x(1.05*10^6:end);

% figure
% plot(y) % to verify that injection data are excluded

nonInj = datapm(1.05*10^6:end,1:12);

[R C] = size(nonInj);

counter = 1;
xd = zeros(34,3000,12);
for i = 1:R
    x = nonInj(i,11);
    if x < 1
       if nonInj(i + 1,11) > 1
           xd(1:34,counter,:) = nonInj(i-34+1:i,:);
           counter = counter+1;
       else
       end
    else
    end
end

% Compute correlation coefficient
for sensor=1:9
    for j=2:counter % counter is the number of blade
passages
        xcf=corrcoef(xd(:,j-1,sensor),xd(:,j,sensor));
        corcf(j,sensor)=xcf(2,1);
    end
end


% compute Mean and Variance of the corr. coef. using a
trailing window of
% 10 - 100 data point
% In this case choosing 100 coef.
```

```
window = corcf(end - 101: end - 1,:); % Window of 100 corr.
coef.

% calculate Mean

for i=1:9
   meanCorr052(i)=mean(window(:,i));
end

% calculating Variance

Variance052 = var(window);
Colu = 1:1:9;

% Now Plotting Mean and Variance.

subplot(5,1,2);
plot(Colu,meanCorr052,'-ro',Colu,Variance052,'-.b'),
%legend('meanCorr','Variance')
xlabel('Sensors');ylabel('M & V (.52)');

% for flow coef. .54
%_____________________________________________________________
_________
%
_____________________________________________________________
_______

clear;
load dynamic_data_054.dat
datav = dynamic_data_054;
[L,ca]=size(datav);
Ts=1/20000;

% Conversion Factors [Pa/V]
con(1)=968.3;con(2)=879.9;con(3)=891.9;con(4)=603.7;con(5)=
599.2;
con(6)=599.5;con(7)=596.8;con(8)=600;con(9)=917.7;

% correction for drift
load dynamic_data_zero.dat;
drift=dynamic_data_zero;
[Ld,col]=size(drift);
for i=1:col
    driftmean(i)=mean(drift(:,i));
    datavm(:,i)=datav(:,i)-driftmean(i);% data with drift
adjusted
```

```
end

for i=1:9
    datapm(:,i)=datavm(:,i)*con(i); % data in [Pa]
end;
datapm(:,10:12)=datav(:,10:12);

x = datav(:,10);

% figure
% plot(x)

% Now separate data of non - injection by looking at the
plot

y = x(1.1*10^6:end); % cut off datas

% figure
% plot(y) % to verify that injection data are excluded

nonInj = datapm(1.1*10^6:end,1:12); % cut off datas

[R C] = size(nonInj);

counter = 1;
xd = zeros(34,3000,12);
for i = 1:R
    x = nonInj(i,11);
    if x < 1
       if nonInj(i + 1,11) > 1
           xd(1:34,counter,:) = nonInj(i-34+1:i,:);
           counter = counter+1;
       else
       end
    else
    end
end

% Compute correlation coefficient
for sensor=1:9
    for j=2:counter % counter is the number of blade
passages
        xcf=corrcoef(xd(:,j-1,sensor),xd(:,j,sensor));
        corcf(j,sensor)=xcf(2,1);
    end
end
```

```
% compute Mean and Variance of the corr. coef. using a
trailing window of
% 10 - 100 data point
% In this case choosing 100 coef.

window = corcf(end - 101: end - 1,:); % Window of 100 corr.
coef.

% calculate Mean

for i=1:9
   meanCorr054(i)=mean(window(:,i));
end

% calculating Variance

Variance054 = var(window);
Colu = 1:1:9;

% Now Plotting Mean and Variance.

subplot(5,1,3);
plot(Colu,meanCorr054,'-ro',Colu,Variance054,'-
.b'),%legend('meanCorr','Variance')
xlabel('Sensors');ylabel('M & V (.54)');

% For flow coef .58
%_____________________________________________________________
_____________
%_____________________________________________________________
_____________

clear;
load dynamic_data_058.dat
datav = dynamic_data_058;
[L,ca]=size(datav);
Ts=1/20000;

% Conversion Factors [Pa/V]
con(1)=968.3;con(2)=879.9;con(3)=891.9;con(4)=603.7;con(5)=
599.2;
con(6)=599.5;con(7)=596.8;con(8)=600;con(9)=917.7;

% correction for drift
load dynamic_data_zero.dat;
drift=dynamic_data_zero;
```

```
[Ld,col]=size(drift);
for i=1:col
    driftmean(i)=mean(drift(:,i));
    datavm(:,i)=datav(:,i)-driftmean(i);% data with drift
adjusted
end

for i=1:9
    datapm(:,i)=datavm(:,i)*con(i); % data in [Pa]
end;
datapm(:,10:12)=datav(:,10:12);

x = datav(:,10);

% figure
% plot(x)

% Now separate data of non - injection by looking at the
plot

y = x(1.07*10^6:end); % cut off datas

% figure
% plot(y) % to verify that injection data are excluded

nonInj = datapm(1.07*10^6:end,1:12); % cut off datas

[R C] = size(nonInj);

counter = 1;
xd = zeros(34,3000,12);
for i = 1:R
    x = nonInj(i,11);
    if x < 1
       if nonInj(i + 1,11) > 1
           xd(1:34,counter,:) = nonInj(i-34+1:i,:);
           counter = counter+1;
       else
       end
    else
    end
end

% Compute correlation coefficient
for sensor=1:9
    for j=2:counter % counter is the number of blade
passages
```

```
        xcf=corrcoef(xd(:,j-1,sensor),xd(:,j,sensor));
        corcf(j,sensor)=xcf(2,1);
    end
end


% compute Mean and Variance of the corr. coef. using a
trailing window of
% 10 - 100 data point
% In this case choosing 100 coef.

window = corcf(end - 101: end - 1,:); % Window of 100 corr.
coef.

% calculate Mean

for i=1:9
   meanCorr058(i)=mean(window(:,i));
end

% calculating Variance

Variance058 = var(window);
Colu = 1:1:9;

% Now Plotting mean and Variance

subplot(5,1,4);
plot(Colu,meanCorr058,'-ro',Colu,Variance058,'-
.b'),%legend('meanCorr','Variance')
xlabel('Sensors');ylabel('M & V (.58)');

% For flow coef. .60
%_____________________________________________________________
_____________
%_____________________________________________________________
_____________

clear;
load dynamic_data_060.dat
datav = dynamic_data_060;
[L,ca]=size(datav);
Ts=1/20000;

% Conversion Factors [Pa/V]
con(1)=968.3;con(2)=879.9;con(3)=891.9;con(4)=603.7;con(5)=
599.2;
```

```
con(6)=599.5;con(7)=596.8;con(8)=600;con(9)=917.7;

% correction for drift
load dynamic_data_zero.dat;
drift=dynamic_data_zero;
[Ld,col]=size(drift);
for i=1:col
    driftmean(i)=mean(drift(:,i));
    datavm(:,i)=datav(:,i)-driftmean(i);% data with drift
adjusted
end

for i=1:9
    datapm(:,i)=datavm(:,i)*con(i); % data in [Pa]
end;
datapm(:,10:12)=datav(:,10:12);

x = datav(:,10);

% figure
% plot(x)

% Now separate data of non - injection by looking at the
plot

y = x(1.05*10^6:end); % cut off datas

% figure
% plot(y) % to verify that injection data are excluded

nonInj = datapm(1.16*10^6:end,1:12); % cut off datas

[R C] = size(nonInj);

counter = 1;
xd = zeros(34,3000,12);
for i = 1:R
    x = nonInj(i,11);
    if x < 1
       if nonInj(i + 1,11) > 1
           xd(1:34,counter,:) = nonInj(i-34+1:i,:);
           counter = counter+1;
       else
       end
    else
    end
end
```

```
% Compute correlation coefficient
for sensor=1:9
    for j=2:counter % counter is the number of blade
passages
        xcf=corrcoef(xd(:,j-1,sensor),xd(:,j,sensor));
        corcf(j,sensor)=xcf(2,1);
    end
end


% compute Mean and Variance of the corr. coef. using a
trailing window of
% 10 - 100 data point
% In this case choosing 100 coef.

window = corcf(end - 101: end - 1,:); % Window of 100 corr.
coef.

% calculate Mean

for i=1:9
   meanCorr060(i)=mean(window(:,i));
end

% calculating Variance

Variance060 = var(window);
Colu = 1:1:9;

%Now plotting all the mean and variance for comparision
%_____________________________________________________

subplot(5,1,5);
plot(Colu,meanCorr060,'-ro',Colu,Variance060,'-
.b'),%legend('meanCorr','Variance')
xlabel('Sensors');ylabel('M & V(.60)');
```

A10 – conver_dc.m

```
% This program converts discere time TF to Continuous time
TF

TFd = nlhw5.LinearModel; % linear model of Hammeristien
Wiener model

TFc = d2c(TFd) % converting discete to continuos time

[wn,zeta] = damp(TFc);

WnHz = wn./(2*pi)
```

A11 – Resample.m

```
% This program resamples the injection data to match the
number of
% data points in the corcf.mat or match the number of
correlation
% coefficient
% This program obtains the corresponding data point from
injection column

clear; clc; close all;

load corcf.mat;

% Last value of corcf is NaN so taking it off

corcf = corcf(1:end-1,:);

load dynamic_data_052.dat;

data = dynamic_data_052;
[L Col] = size(data);
output = corcf(:,5); % making the output to be the
correlation coefficient
% of sensor 5.

[R C] = size(output);

% input data i.e. data must be resampled.

input = zeros(R,1);

count=1;
newy=NaN(L,1);
for i=1:L
    x=data(i,11);
    if x<1
        if data(i+1,11)>1
            newy(i-15,:) = corcf(count,5);
            count=count+1;
            newy(i,:)=3;
        else
            newy(i,:)=3;
        end
    else
        newy(i,:)=3;
```

```
    end
end

% now find where the correlation coefficients lies

[Rn,Cn] = find(newy >= -1 & newy <= 1 );

input = data(Rn,10);
%
% p = 20000;
%
% q = p * R / L;
% y = resample(data(:,10),p,q);
```