Application of Statistical Pattern Recognition Techniques in Structural Health Monitoring

Mohammad Sajjadul Islam

Thesis

in

The Department

of

Building Civil and Environmental Engineering

Presented in Partial Fulfillment of the Requirements for

The Degree of Master of Applied Science (Civil Engineering) at

Concordia University

Montreal, Quebec, Canada

November 2009

Your file Votre référence
ISBN: 978-0-494-67237-2
Our file Notre référence
ISBN: 978-0-494-67237-2

Abstract

In Structural Health Monitoring (SHM), various sensors are installed in the critical locations of a structure. The signals from sensors are either continuously or periodically analyzed to determine the state and performance of the structure. The objective of this thesis is to apply statistical pattern recognition techniques to determine the relation among signals or engineering data from various sensors installed on a structure. An objective comparison of the sensor data at different time ranges is essential for assessing the structural condition, detect any malfunction of sensors, or excessive load experienced by the structure which leads to potential damage in the structure. The objectives of the current research are to establish a relationship between the data from various sensors to estimate the reliability of the data, and to determine defective sensor using the statistical pattern matching techniques. In order to achieve these goals, new methodologies based on statistical pattern recognition techniques have been developed and implemented using the MATLAB environment. The proposed methodologies have been developed and validated using sensor data obtained from an instrumented bridge and road test data from industry. The statistical pattern matching techniques are quite new in SHM data interpretation and current research demonstrate that it has high potential in assessing structural conditions, especially when the data is noisy and susceptible to environmental disturbances.

Acknowledgements

First of all I would like to express my deepest gratitude to my thesis supervisor, Dr. Ashutosh Bagchi, for his continuous guidance and support throughout my thesis work. During my work he always provides me key directions and inspiration with his great personality, vast experience and immense knowledge.

I want to acknowledge International Truck and Engine Corporation who supported this research and I also acknowledge ISIS (Intelligent Sensing for Innovative Structures) Canada Research Network.

I am forever indebted to my parents, sisters and friends for their countless support and inspiration.

Table of Contents

List of Figures

List of Tables

List of Abbreviation

ARX	Auto Regressive Xeogenous
ECT	Eddy Current Testing
FEA	Finite Element Analysis
FEM	Finite Element Model
FHWA	Federal Highway Administration
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
ISIS	Intelligent Sensing for Innovative Structures
LPT	Liquid Penetration Testing
MISO	Multiple Input Single Output
MT	Magnetic Particle Testing
NBIP	National Bridge Inspection Program
NDE	Non Destructive Evaluation
NDI	Nondestructive Inspection
RT	Radiographic Testing
SHM	Structural Health Monitoring
SLT	Static Load Test
SPRT	Sequential Probability Ratio Test
UT	Ultrasonic Testing
VBDI	Vibration Based Damage Identification

Chapter1: Introduction

1.1 General

Bridge condition information is primarily obtained through scheduled biennial visual inspection which is enforced by the National Bridge Inspection Program (NBIP) in the limited states. However, a study funded by the Federal Highway Administration (FHWA) concluded that these are labor and cost intensive activities (Dubin and Yanev, 2001), and that visual inspections are subjective and unreliable (Phares, 2001). In addition, the effectiveness of these periodic inspections is highly constrained by the shortage of timely damage detection ability. Therefore, to maintain overall highway operational safety with current funding limitations, the development of continuous, automatic, and low cost bridge structural health monitoring (SHM) systems is highly and urgently needed(Ping Lu, 2008).

Benefiting from the rapid development of computing, sensing, and tele-communication technology during the last two decades, computer-based long term SHM systems have been developing and are more and more widely utilized to provide timely condition information. A SHM system includes data acquisition, storage and evaluates the status of entire structure or structural component continuously. It is desired monitoring a structure would assist in assessing structural conditions and detecting damage in structures. Damage in a structure may be caused by many factors such as cracking, corrosion, reduction in material properties, reinforcement rupture fractured welds and loosened bolts. In general the context, damage assessment can be defined at four levels (Rytter, 1993).

They are

1. To detect whether there is damage
2. To determine the location of damage
3. To quantify the extent of damage and
4. To carry out prognosis such as safety evaluation and remaining life prediction

It is probable that only global changes such as foundation settlement, bearing failure or major defects, such as the loss of main cable tension or the rupture of the deck, would be detectable by global SHM procedures with a minimum of optimally located sensors, as Saint Venant's principle indicates that the zone of influence is typically small. It is hard, if not impossible, to find a global measurement that can be monitored by a few sensors while sensitive to the localized structural response change. Therefore, a relatively dense and wide spread sensor network is necessary to achieve the required coverage if the damage location is not already known. At the same time, the data acquisition frequency should be sufficiently high so that enough information can be obtained to support the decision making. High data collection frequencies coupled with large sensor numbers leads to extremely large data volumes. Extremely large data volumes have been seen as problematic attributes of some long term SHM systems. Data reduction that results in the extraction of useful information from the original data is a key step toward the development of a real-time/near real-time damage detection approach.

1.2 Motivation

SHM can reduce the chance of catastrophic failure, maintenance costs and down time for rehabilitation. According to Mufti, 2001, more than 40% of the bridges in service in Canada are over 30 years old. Therefore many of these bridges need proper diagnosis, rehabilitation or even partial re-construction in order to make them safe for traffic and also prevent long down-time if sudden collapse occurs. Chase and Washe 1997, conducted a similar survey for the bridges in United States of America, and found that about 33% of the total bridges were deficient. Most of these bridges were built before 1970, and their health condition is yet to be determined by any instrumental and scientific approach. Therefore, in the context of structural safety, the need for the application of SHM has become highly important. Moreover for maintenance and rehabilitation purposes, the need for SHM has increased recently. Within the last two decades, long-term SHM of bridges has been increasing dramatically due to the following factors (Farrar and Doebling, 1997):

1. Aging of bridge infrastructure
2. Bridge failures
3. Realization of the ineffectiveness of visual inspection
4. Technology development

According to Sohn et al 2000, sensors measuring strains and vibration of a structure produce signals that always respond to the change of environmental and operational conditions. Each group of signals can be considered a pattern that has some relation to the structural and ambient condition. They proposed that if the effect of ambient condition to the patterns is normalized, they should be clearly identical or close to one another for similar vibration effect as long as structural vibration property remains same. However, it can be assumed, the change in physical properties, mainly stiffness, should be reflected on the processed signal blocks or patterns. Based on this assumption, various methods of damage detection by pattern recognition have been developed. Pattern recognition is aimed for machine learning process, ability of a computer to identify and classify them to make a decision. It is this feature that makes it attractive to create automated SHM system.

1.3 Thesis objective

The main objectives of this paper are to

1. Assess the structural conditions using SHM data
2. Assess the reliability of sensor data
3. Detect defective sensors or potential damage in structure.

1.4 Thesis organization

This thesis has been organized into seven chapters. Introduction and objective of this thesis are presented in the current chapter i.e. Chapter 1. A thorough review of literature on structural health monitoring in civil structures are presented in Chapter 2. In Chapter 3 detail methodology applied in this thesis has been described. Application of methodologies, test, results, figures, tables are in Chapter 4 and Chapter 5 respectively. Finally summary and conclusion are in Chapter 6. The thesis ended with a list of references and appendices.

Chapter2: Literature review

2.1 General

Damage detection methods for in-service structural components are non destructive. Canadian Institute of NDE describes non-destructive examination (NDE), also referred to as NDT (nondestructive testing) and NDI (nondestructive inspection), provide information about the condition of materials and components without destroying them (CINDE 2008). Visual inspection, liquid penetration testing (LPT), magnetic particle testing (MT), radiographic testing (RT), ultrasonic testing (UT), eddy current testing (ECT), static load test (SLT) are the examples of nondestructive testing. Visual inspection is difficult for large and complex structures in addition, the structure must be accessible. LPT is not applicable for the determination of the strength of the material and needs prior knowledge of the location of the damage like UT and ECT. MT is not suitable for concrete and wood. In RT, two dimensional views hide additional defects in a structural component. SLT cannot be used for the prior warning of occurrence of damage and also the structure may have to be evacuated for the test. For the last few years there has been noticeable research work done in the Vibration Based Damage Identification (VBDI) method. A number of different analytical techniques have been developed for the VBDI method. They are based on frequency changes, method based on mode shape change, mode shape curvature method, method based on change in flexibility, damage index method, method based on modal residual vector, matrix update methods and neural network (Humar et al, 2006).These methods are usually applied on a finite element model (FEM), to study two real structures for simulated damage detection based on practical input of characteristics parameters, such as frequency and mode shapes. FEM should very closely represent the real structure. However, it is difficult to achieve real life as structures are relatively large in size with the inherently greater uncertainties in material properties, support conditions, and connectivity of components. In a complex structure, the change of modal frequencies is not very sensitive to the damage of some member, and damaged portions may have larger deviation in mode shapes than rest of the structure such that it is difficult to calculate the curvature from the measured mode shape. Considering the limitations of ability as VBDI methods for damage detection, pattern recognition approaches have been developed in recent years. They are free from modeling error because they do not need any modeling of the structure like FEM methods. In the present study, a statistical pattern recognition technique has been described in detail.

2.2 Pattern recognition by statistical methods

Fugate et al 2001 have proposed a generalized approach for SHM by statistical pattern recognition.

2.2.1 Operational evaluation

It defines damage for the system being monitored and also the operational and environmental conditions under which the system structure function.

2.2.2 Data acquisition

It involves selection of the types of sensors and location where they should be placed, determination of optimal number of sensors to be used and setup of data acquisition

,storage or transmission hardware. Vibration response can be only obtained by ambient excitation.

2.2.3 Data cleansing

The non-structural conditions such as civil loading or climate conditions always vary with time. Therefore it is needed to normalize the data to make them compatible to analyze for damage detection. In the case of varying environmental or operational conditions, normalized data can be compared at similar times of an environmental or operational cycle. Sources that affect the variation of data and the structure monitored are to be identified and minimized. For those variability sources which can be eliminated, they should be made available to be statistically quantified. Signals are usually gathered continuously. Strain data is significantly influenced by temperature and external loading. Data needs to be corrected for all of these external noises on the signals. There are various ways to denoise data. Some are briefly explored in the following section.

2.2.4 De noising

De-noising is a process of signal recovery from noisy data. This problem is easy to understand by looking at the following simple example, shown in figure 2.1, where a slow sine wave is corrupted by white noise. The general de-noising procedure involves three steps. The basic version of the procedure follows the steps described below.

Decompose: Choose a wavelet, choose a level N. Compute the wavelet decomposition of the signal, s, at level N.

Threshold detail coefficients: For each level from 1 to N, select a threshold and apply soft thresholding to the detailed coefficients.

Reconstruct: Compute the wavelet reconstruction using the original approximation coefficients of level N and the modified detail coefficients of levels from 1to N.

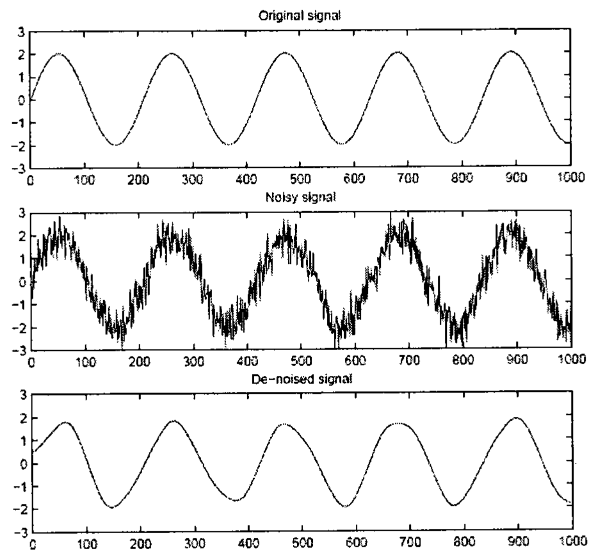


Figure 2.1 A sample de-noising

Struzik et al 1999 have presented that Haar wavelet transform is a simple and powerful technique which allows for the rapid evaluation of similarity between time series in large data bases.

2.2.5 Filter

A filter is usually needed to perform frequency dependent alteration of a data sequence. For example, a filter could be applied to remove noise above 30 Hz from a data sequence sampled at 100 Hz. A more rigorous specification might call for a specific amount of passband ripple, stopband attenuation, or transition width. A very precise specification could ask to achieve the performance goals with the minimum filter order, or it could call for an arbitrary magnitude shape, or it might require an FIR filter. Filter design is the process of creating the filter coefficients to meet specific filtering requirements. Filter implementation involves choosing and applying a particular filter structure to those coefficients. Only after both design and implementation have been performed can data be filtered. To meet the specifications with more rigid constraints like linear phase or arbitrary filter shape, FIR (finite impulse response) and direct IIR (Infinite impulse response) filter design routines are followed. The primary advantage of IIR filters over FIR filters is that they typically meet a given set of specifications with a much lower filter order than a corresponding FIR filter. Although IIR filters have nonlinear phase, data processing within MATLAB software is commonly performed "offline," that is, the entire data sequence is available prior to filtering. This allows for a non causal, zero-phase filtering approach (via the "filtfilt" function), which eliminates the nonlinear phase distortion of an IIR filter. The classical IIR filters, such as, Butterworth approximate the ideal "brick wall" filter in different ways. Roy et al 1997 have explored the implications of the low-pass Butterworth filter on the characteristics of correlation analyses. It has also proposed that knowing the filter response, it is possible to reconstruct the original signal spectrum and to allow comparisons between data collected with different instruments. The autocorrelation function also is affected by filtering which increases the value of the coefficients in the first lags, resulting in an overestimation of the integral length scale of coherent structures. These important effects add to those related to size and shape differences in electromagnetic current meters sensors and must be taken into account in comparative studies.

2.2.6 Data normalization

Sohn et al (2001a, 2001b) show that normalizing and standardizing the acceleration time history x(t) by

$$x = \frac{x-\mu}{\sigma} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2.1)$$

where μ is the mean of the signal and σ is its standard deviation.

2.2.7 Feature extraction

It is the process of identifying damage-sensitive properties derived from the measured vibration response that allows one to distinguish between the undamaged and damage structure. Silva et al 2007 deals with the application of a two-step auto-regressive and auto-regressive with exogenous inputs (AR-ARX) model for linear prediction of damage diagnosis in structural systems. This damage detection algorithm is based on the monitoring of residual error as damage-sensitive indexes, obtained through vibration response measurements.

The basic input-output configuration of ARX model is shown in figure 2.2. Assuming unit sampling interval, there is an input quantity or signal u(t) and output quantity or signal y(t) , t=1,2......n. Assuming that the signals are related by a linear system , input-output relationship can be written as

$$y(t)= G(q)u(t)+v(t) \quad .... (2.2)$$

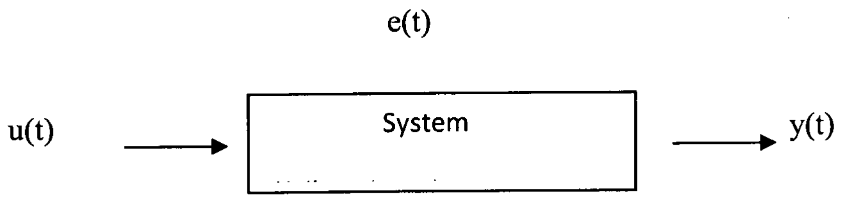


Figure 2.2 Basic Input-output configuration of ARX model.

where q is the shift operator and G(q) is the transfer function of the deterministic part of the system(t) is the disturbance of the system which can be described as filtered white noise.

$$V(t)=H(q)e(t) \text{ .... } (2.3)$$

Where e(t) is white noise with variance and H(q) is the transfer function of the stochastic part of the system. where e(t) is white noise with variance and H (q) is the transfer function of the stochastic part of the system. Equations (2.2) and (2.3) together, give a time-domain description of the system,

$$y(t)= G(q) + H(q)\, e(t) \text{............. } (2.4)$$

A commonly used parametric model is the ARX model that corresponds to

$$\mathbf{G(q) = q - nk\frac{B(q)}{A(q)};\ \ H(q) = \frac{1}{A(q)}} \text{... } (2.5)$$

The number nk is the number of delays from input to output. Where A(q) and B(q) are polynomials in the shift operator q -1

$$B(q)=\begin{bmatrix} b_{11} & b_{12} & \cdots\cdot & \cdots\cdot \\ b_{21}q^{-1} & b_{22}q^{-1} & \cdots\cdot & \cdots\cdot \\ \cdots\cdot & \cdots\cdot & \cdots\cdot & \cdots\cdot \\ b_{nb1}q^{-nb+1} & b_{nb2}q^{-nb+1} & \cdots\cdot & b_{nbnu}q^{-nb+1} \end{bmatrix}$$

Here, B(q) is an *nb x nu* matrix. The numbers na and nb are the orders of their respective polynomials, and nu is the number of input variables. For the SISO (Single Input Single Output) model, nu = 1. The general structure of the SISO or MISO (Multiple Input Single Output) ARX models is given by

$$A(q)\, y(t) = B\,(q)\, u\,(\,t - nk\,) + e\,(t) \text{..........} (2.6)$$

One of the simplest models in the system identification literature is the ARX model. Where AR refers to the Auto-Regressive part A(q) y(t) and X to the extra input B(q) u (t) part. Eq. (2.7) can also be written explicitly for a first-order model with a delay of two sampling times as,

$$y(t) = -\alpha_1 y(t-1) + b_{11}u_1(t-2) + b_{12}u_1(t-2) + \dots + b_{1n}u_n(t-2) + e(t) \dots\dots (2.7)$$

Given a description and having observed the input–output quantities u; y, the errors or residuals e(t) in Eq. (2.8) can be computed as

$$e(t) = H^{-1}(q)\,[\, y(t) - G(q)\,u(t)\,] \dots\dots\dots\dots \quad (2.8)$$

These residuals are, for given observations y and u, functions of G and H . These in turn are parametrized by the polynomial in Eq. (2.9). The most common parametric identification method is to determine the estimates of G and H by minimizing

$$V_n(G,H) = \sum_{t=1}^{n} e^{\wedge}2(t) \dots\dots\dots (2.9)$$

That is

$$[\hat{G}_n \hat{H}_n] = \arg\min \sum_{t=1}^{n} e^{\wedge}2(t) \dots\dots\dots\dots\dots (2.10)$$

This is a prediction error method. The identification method for the ARX model is the LS(Least Square) method, which is a special case for the prediction error method. The LS method is the most efficient polynomial estimation method because this method solves linear regression equations analytically.

For linear models, model estimation can be done using time-domain data, and then model validation can be done using frequency domain data. For nonlinear models, only time-domain data can be used for both estimation and validation. Measured and simulated model output pattern matching can be computed using the following equation:

Best fit = $1 - \frac{|y-\hat{y}|}{|y-\bar{y}|} x100$..................(2.11)

In this equation, y is the measured output, $\hat{y}$ is the simulated or predicted model output, $\bar{y}$ and is the mean of y. 100% corresponds to a perfect fit, and 0% indicates that the fit is no better than guessing the output to be a constant ($\hat{y} = \bar{y}$). Because of the definition of Best Fit, it is possible for this value to be negative. A negative best fit is worse than 0% and can occur for the following reasons: The estimation algorithm failed to converge. The model was not estimated by minimizing $|y - \hat{y}|$. Best Fit can be negative when you minimized 1-step-ahead prediction during the estimation, but validate using the simulated output $\hat{y}$. The validation data set was not preprocessed in the same way as the estimation data set. (Matlab help file)

Sohn et al 2000 proposed difference between the actual acceleration measurement for the new signal and prediction obtained from the auto –regressive and auto-regressive with exogenous model developed from the selected reference signal, is defined as the damage –sensitive feature. The applicability of this approach is demonstrated using acceleration time histories obtained from an eight degree-of –freedom mass-spring system.

Nair and kiremidjian 2006 proposed two algorithms for detection of damage with its location. Both algorithms are based on the time series analysis of vibration data, and the feature vectors obtained are classified using a pattern classification technique. The vibration signals obtained from the structure are modeled as auto-regressive moving average (ARMA) processes. The feature vector used in both algorithms is the first three autoregressive (AR) coefficients. In the first algorithm, a damage index is proposed based on first three AR coefficients and a metric in the AR coefficient space is used for damage localization. ASCE Benchmark Structure is used that is a four story, two-bay by two-bay steel braced frame.

2.2.8 Statistical model development

It is concerned with the implementation of the algorithms that analyze distribution of the extracted features in an effort to determine the damage state of the structure. The appropriate algorithm to use will depend on the ability to perform supervised and unsupervised learning. Supervised learning refers to the case where examples of data from damaged and undamaged structures are available. Unsupervised learning refers to the case where data is only available from the undamaged structure.

2.3 Summary

In this chapter, relevant literature on pattern recognition using statistical methods have been reviewed for the purpose of understanding the field of thesis and identifying the scope of work. It is found that structural damage affects the dynamic properties of a structure, causing a change in the vibration signals i.e. strain and acceleration time histories. Damage detection can be performed using time series analysis of vibration signals measured from a structure before and after damage. The references cited in this review propose different techniques of statistical pattern recognition for extracting damage-sensitive features from vibration response of laboratory based simple structure. In the thesis, statistical pattern recognition techniques are applied for damage detection of real structure with operational and environmental variability.

Chapter3: Methodology

3.1 General

In this study data are collected from two sources. One is from industry which is vibration response, acceleration data, of a heavy vehicle during road load test and same data produced by software simulation by the industry partner. Another source of data is from an instrumented bridge, Portage Creek Bridge, Victoria, BC. Bridge data are collected from the ISIS website [ISIS Canada SHM Database].

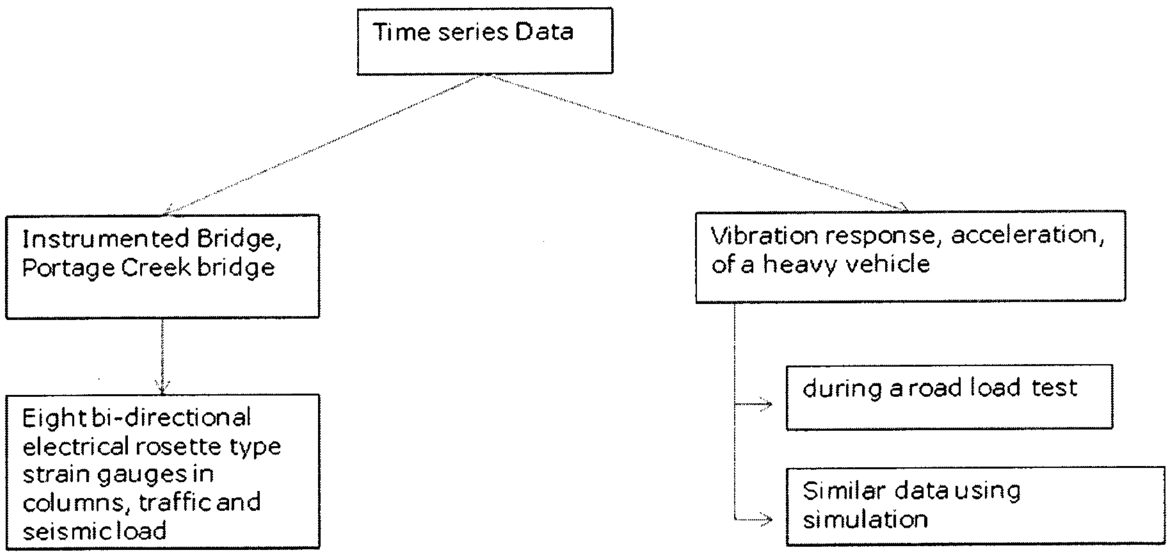

Time series Data
Instrumented Bridge, Portage Creek bridge
Vibration response, acceleration, of a heavy vehicle
Eight bi-directional electrical rosette type strain gauges in columns, traffic and seismic load
during a road load test
Similar data using simulation

Figure 3.1 Data sources

3.2 Details of the monitored structure used in this study

While the details of the heavy vehicle road test data are not provided here to protect the confidentiality of the information as required by the industry partner, details of the Portage Creek Bridge monitoring are provided below.

The Portage Creek Bridge located in the British Columbia (BC), Canada has been used as a case study and SHM data source for the current thesis. The Ministry of Transportation in BC designed the Portage Creek Bridge, as shown in Figure3.2. Located in the City of Victoria, British Columbia, the bridge crosses Interurban Road and Colquitz River at McKenzie Avenue. The bridge is described as a 124 m (407 ft) long, three-span structure with a reinforced concrete deck supported on two reinforced concrete piers, and abutments on H piles . The deck has a roadway width of 16.2 m (53 ft) with two 1.98 m (6′6″) sidewalks and aluminum railings. There are eight bi-directional electrical strain gauge rosettes on each column, four long gauge fiber optic sensors on each column and

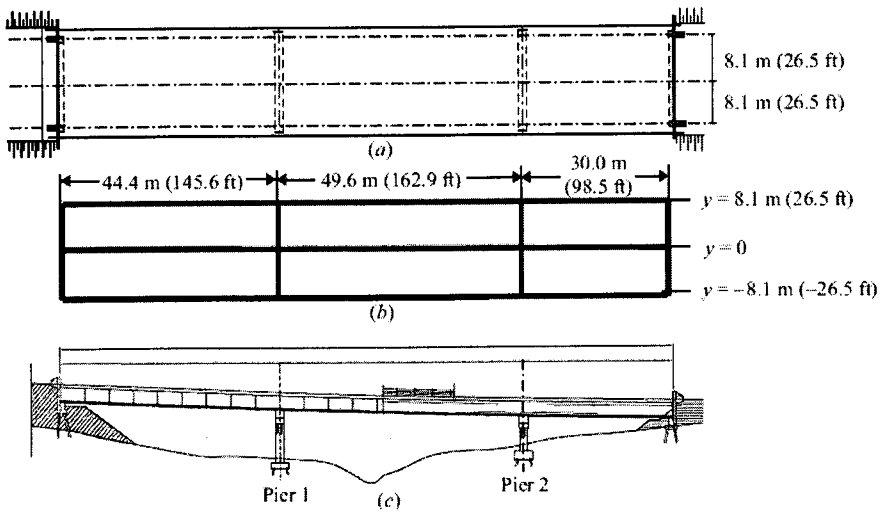


Figure 3.2 Plan and elevation of Portage Creek Bridge (Huffman et al 2006)

one 3-D accelerometer on top of the pier cap of each column. An elevation view of the instrumented pier No. 2 is shown in Figure 3.3 with sensor locations.

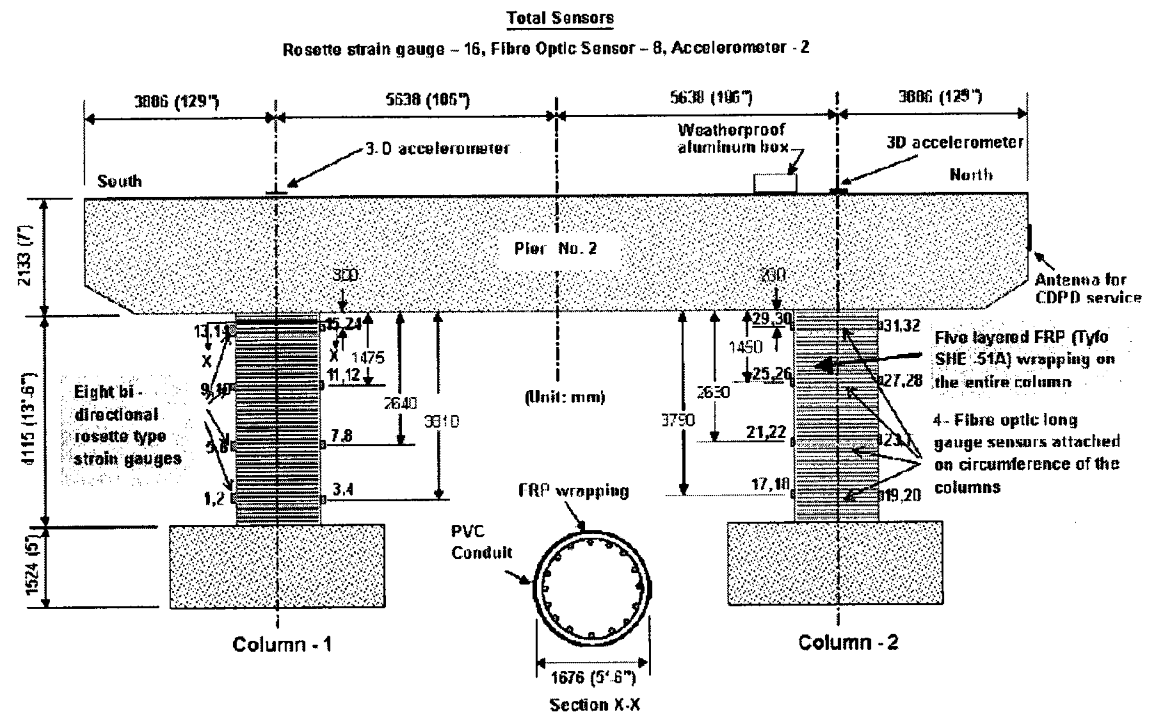

Total Sensors
Rosette strain gauge - 16, Fibre Optic Sensor - 8, Accelerometer - 2
3886 (129")
5638 (186")
5638 (186")
3886 (129")
South
3-D accelerometer
Weatherproof aluminum box
3D accelerometer
North
Pier No. 2
Antenna for CDPD service
2133 (7')
4115 (13'-6")
1524 (5')
Eight bi-directional rosette type strain gauges
Five layered FRP (Tyfo SHE-51A) wrapping on the entire column
4-Fibre optic long gauge sensors attached on circumference of the columns
(Unit: mm)
FRP wrapping
PVC Conduit
Column - 1
Column - 2
1676 (5'-6")
Section X-X

Figure 3.3 Sensor locations on Pier-2 columns of the Portage Creek Bridge (Huffman et al 2006).

In this study, the signals produced in every minute from eight bi-directional strain gauges installed at column-2 (C2) of pier-2 of the Portage Creek Bridge in Victoria, BC (Figure 3.2) have been analyzed. Each bi-directional strain gauges produces two data, one for horizontal movement and another for vertical movement. So, from eight strain gauges, sixteen output signals are obtained. In the Table 3.1 column (1), (2), (3) are same sensor of pier-2 with different notation same as column (4), (5), (6). Table 3.2 is same as Table 3.1 for pier 2. The Caltrans (California Department of Transportation) recommended that any approach should include enough modes of vibration to achieve a total mass participation of not less that 90% for a given bridge. To capture a sufficient number of modes that gives mass participation more than 95 %, it is required to collect data up to 16 Hz. But considering the Nyquist frequency, frequency data collected is double (32 Hz).the monitoring data was available from the ISIS Canada Research Network. The web page for Portage Creek Bridge real time monitoring is hosted by a centralized SHM system ISIS Canada website.The data available from the bridge site covers a period between 2004 and 2006. A user can pick up and access individual sensor's data from the sensors list. The data sampling rate is 32Hz. The approaches followed in this study to process the data are:

1. Time interval between points, number of data points, data channels to query are selected.
2. Data points are saved as comma separated values.
3. Data collected in every second are converted into minute data and then again minute data are converted into hourly data, micro-strain/hr.
4. For the training, time data are taken in the month of December/05, January/06 and February/06 data. Total number of data points in a segment for each strain gauge sensor is 1738. (Out of 2160 data points, 1738 are valid and rests are NaN, not a number, which are removed from data set.)
5. The testing data are taken from the month of March/06 because of the presence of peaks or novel events in that period.

Table 3.1 Sensors on Pier-2 of the Portage Creek Bridge

(1)	(2)	(3)	(4)	(5)	(6)
Sensor# 17	1-1	Strain1_1_C2	Sensor# 18	1-2	Strain2_2_C2
Sensor# 19	2-1	Strain3_1_C2	Sensor# 20	2-2	Strain4_2_C2
Sensor# 21	3-1	Strain5_1_C2	Sensor# 22	3-2	Strain6_2_C2
Sensor# 23	4-1	Strain7_1_C2	Sensor# 24	4-2	Strain8_2_C2
Sensor# 25	5-1	Strain9_1_C2	Sensor# 26	5-2	Strain10_2_C2
Sensor# 27	6-1	Strain11_1_C2	Sensor# 28	6-2	Strain12_2_C2
Sensor# 29	7-1	Strain13_1_C2	Sensor# 30	7-2	Strain14_2_C2
Sensor# 31	8-1	Strain15_1_C2	Sensor# 32	8-2	Strain16_2_C2

3.3 Data preprocessing

In this thesis data has been preprocessed by De noising, Normalization and Filtering following the methods proposed in literature review chapter. All data are normalized using equation 2.1 which removes the mean from each data series. AR-ARX are zero mean Gaussian process. The above normalization approximates the monitoring data to have such characteristics. For the training, time data are taken in the month of December, January and February and the testing data are taken in the month of March to avoid large environmental variation. To remove the remaining small variation normalization is done for both the training and test data because after normalization the features extracted from the signals from the undamaged structure would have similar statistical characteristics and the two signals can be compared (Nair et al 2006).

3.4 Damage identification approach by pattern comparison

The basic concept of this approach was first proposed by Sohn et al 2001. It is logical to assume that the patterns in data a certain state, either steady or agitated, taken at various points of time of the structure will not vary significantly if the structure does not change significantly. Conversely if the structure has undergone a significant change, it should reflect in the pattern of data in a given state. In order to observe the variation of structure by studying the pattern of signals or data blocks, it is necessary to nominate certain block as reference data block with which patterns of the other data series or blocks are compared. Usually, the reference data blocks for particular conditions are taken from the earlier time of the observation of the structure and other data blocks are called test block. The time series model (e.g. ARX model) particularly developed for reference block is defined as reference model. As structure undergoes change, usually, degradation, so will the pattern of data series change. Therefore the pattern of other data blocks will not match closely with that of reference block. In this study two types of approach are proposed for damage identification.

1. Assessing degree of similarity among sensors data by sets of statistical parameters using tool developed in MATLAB.
2. Detection of defective sensor by ARX model build up and application of binary and sequential search method.

3.5 Assessing degree of similarity among sensors' data by sets of statistical parameters using tool developed by MATLAB

The following steps have been followed to achieve the goal.

Step1: Two kinds of data sets have taken. In 1st case same strain gauges data but at different time. In 2nd case same time two different strain gauges data considered.

Step2: Mean, Standard deviation, Skewness are calculated without preprocessing data and cross correlation applied with preprocessing data.

Step3: Finally all results are taken to find out the relationship between sensors signals.

To run the analysis for various sensors a tool developed in Matlab which is described in section 3.5.2. In Figure 3.3 all steps are shown by drawing flow chart to get a quick view of the proposed method.

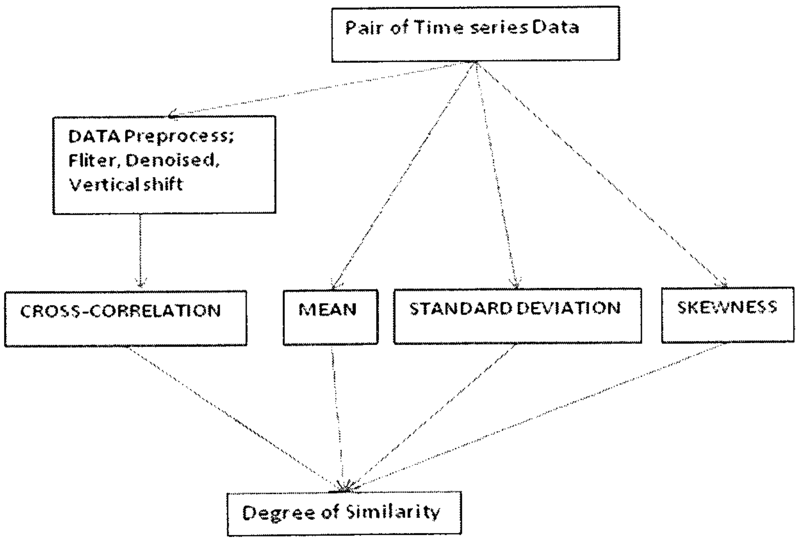


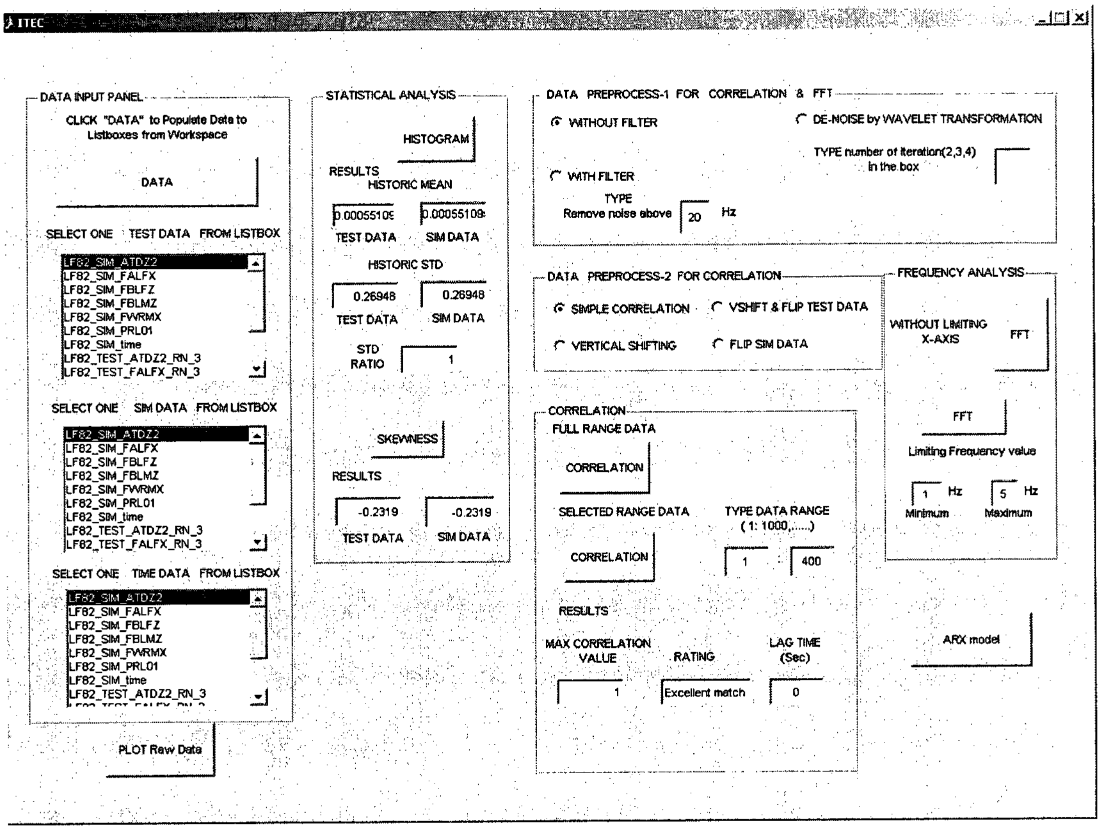
Figure 3.3 Schematic diagram for assessing degree of similarity among sensor data.

3.5.1 Interpretation of the correlation coefficient for assessing degree of similarity

Several authors have offered guidelines for the interpretation of a correlation coefficient. Cohen for example, has suggested the following interpretations for correlations in psychological research indicate 0.1 to 0.3 small correlation, 0.3 to 0.5 medium correlation and 0.5 to 1.0 large correlation. However, all such criteria are in some ways arbitrary and should not be observed too strictly. This is because the interpretation of a correlation coefficient depends on the context and purposes. A correlation of 0.9 may be very low if one is verifying a physical law using high-quality instruments, but may be regarded as very high cases where there may be a greater contribution from many complicating factors. Accordingly, it is important to remember that "large" and "small" correlation should not be taken as synonyms for "good" and "bad" correlation in terms of determining that a correlation is of a certain size. For example, a correlation of 1.0 or −1.0 indicates that the two variables analyzed are equivalent modulo scaling. Scientifically, this more frequently indicates a trivial result than an important one. But in the context of SHM data, the following interpretations are proposed by Islam and Bagchi, 2008. Correlation coefficient > 0.89 indicates excellent match. Correlation coefficient > 0.69 indicates good match. Correlation coefficient > 0.49 indicates fair match. Correlation coefficient < 0.49 indicates not acceptable.

3.5.2 SHM data processing tool developed in Matlab

A software tool has been developed in the MATLAB environment to implement statistical pattern recognition techniques. The user interface of the tool is shown in Figure 3.4. As the statistical pattern recognition method does not required any modeling of bridge structure, this tool can be efficiently used by selecting a time history data of acceleration and strain gauge sensors of any type of bridge. The detailed procedure for operating the tool is described below.



SELECT ONE TEST DATA FROM LISTBOX
LF82_SIM_ATDZ2
LF82_SIM_FALFX
LF82_SIM_FBLFZ
LF82_SIM_FBLMZ
LF82_SIM_FWRMX
LF82_SIM_PRL01
LF82_SIM_time
LF82_TEST_ATDZ2_RN_3
LF82_TEST_FALFX_RN_3

SELECT ONE SIM DATA FROM LISTBOX
LF82_SIM_ATDZ2
LF82_SIM_FALFX
LF82_SIM_FBLFZ
LF82_SIM_FBLMZ
LF82_SIM_FWRMX
LF82_SIM_PRL01
LF82_SIM_time
LF82_TEST_ATDZ2_RN_3
LF82_TEST_FALFX_RN_3

SELECT ONE TIME DATA FROM LISTBOX
LF82_SIM_ATDZ2
LF82_SIM_FALFX
LF82_SIM_FBLFZ
LF82_SIM_FBLMZ
LF82_SIM_FWRMX
LF82_SIM_PRL01
LF82_SIM_time
LF82_TEST_ATDZ2_RN_3

RESULTS
HISTORIC MEAN
0.00055109 TEST DATA 0.00055109 SIM DATA
HISTORIC STD
0.26948 TEST DATA 0.26948 SIM DATA
STD RATIO 1

RESULTS
-0.2319 TEST DATA -0.2319 SIM DATA

[✓ WITHOUT FILTER] WITH FILTER DE-NOISE by WAVELET TRANSFORMATION
TYPE Remove noise above 20 Hz
TYPE number of iteration(2,3,4) in the box

[✓ SIMPLE CORRELATION] VSHIFT & FLIP TEST DATA VERTICAL SHIFTING FLIP SIM DATA

TYPE DATA RANGE (1: 1000,......) 1 400
RESULTS
MAX CORRELATION VALUE 1 RATING Excellent match LAG TIME (Sec) 0

WITHOUT LIMITING X-AXIS
Limiting Frequency value
1 Hz Minimum 5 Hz Maximum

Figure 3.4 Graphical user interface of the tool

To run the software tool in the MATLAB environment, the relevant time history data files need to be loaded to the MATLAB workspace. Once the user interface is loaded, clicking “DATA” button the listbox on the left side of the user interface will be populated by the time history data files that were loaded into the workspace. Now a test data needs to be selected from the first listbox, corresponding SIM data needs to be selected from the second listbox and time data needs to be selected from the third listbox. In Statistical analysis panel “STATISTICS” button shows mean, standard deviation, standard deviation ratio and "SKEWNESS" button shows skewness of the selected time series data. DATA PREPROCESS (1) and DATA PREPROCESS (2) panels are needed for correlation analysis. Selecting data pre-process option correlation has been conducted in two stages. First, full data range are used to quantify the correlation and lag time by using "CORRELATION (1)" button. Second, data are divided into parts and then correlation and lag time are determined by typing data range in the text box and using "CORRELATION (2)" button. DATA PREPROCESS (1) panel is needed for FFT analysis. Selecting data pre-process option FFT has been conducted. Detailed codes are described in appendices.

3.6 Detection of defective sensor by ARX model build up and application of binary and sequential search method.

Binary and sequential search methods commonly used in computer science. The concept has been used in this study for defective sensors detection. Before explaining the total approach , definition of the two methods are described in the next to subsection.

3.6.1 Binary search

Binary search is an algorithm for locating the position of an element in a sorted list by checking the middle, eliminating half of the list from consideration, and then performing the search on the remaining half. If the middle element is equal to the sought value, then the position has been found; otherwise, the upper half or lower half is chosen for search based on whether the element is greater than or less than the middle element. The method reduces the number of elements needed to be checked by a factor of two each time, and finds the target value (Knuth, 1997).

3.6.2 Sequential search

Linear search is a search algorithm, also known as sequential search, that is suitable for searching a list of data for a particular value. It operates by checking every element of a list one at a time in sequence until a match is found. The best case is that the value is equal to the first element tested, in which case only 1 comparison is needed. The worst case is that the value is not in the list (or it appears only once at the end of the list), in which case n comparisons are needed. The simplicity of the linear search means that if just a few elements are to be searched it is less trouble than more complex methods that require preparation such as sorting the list to be searched or more complex data structures, especially when entries may be subject to frequent revision. Another possibility is when certain values are much more likely to be searched for than others and it can be arranged that such values will be amongst the first considered in the list (Knuth, 1997).

Data harvested by the sixteen strains and one temperature data from the SHM database of the Portage Creek Bridge. Then these data are divided into training and input groups. In this study it has chosen simultaneously one strain gauge data as target and remaining sensor data (15 strain gauge and 1 temperature gauge) as input to create a representative set of model by proper training. These models are created to produce the data pattern at a particular period of time with respect to the corresponding input at that time. In Figure 3.5 it is shown that strain gauges data which is time series data first pre-processed by denoising and normalization method. Then 16 sensors data are used to get simulated output with the help of ARX model which is trained by similar sensors data of previously collected undamaged structure. Comparing the output pattern of measured or real time output with simulated output by sequential or binary search method help to find the defective sensor.

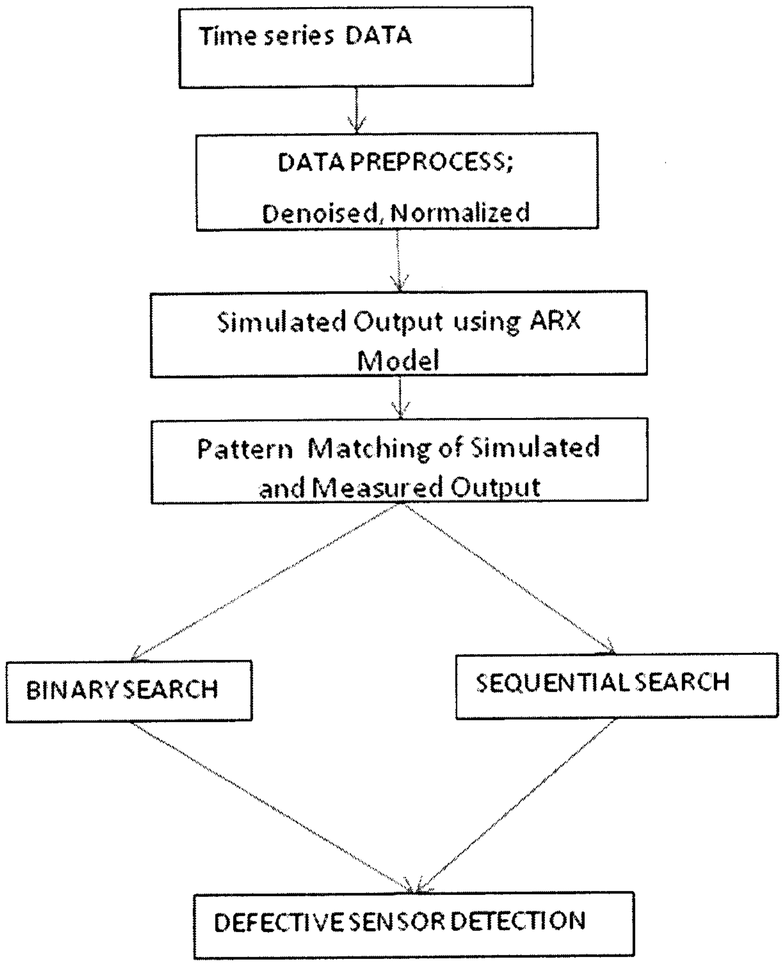
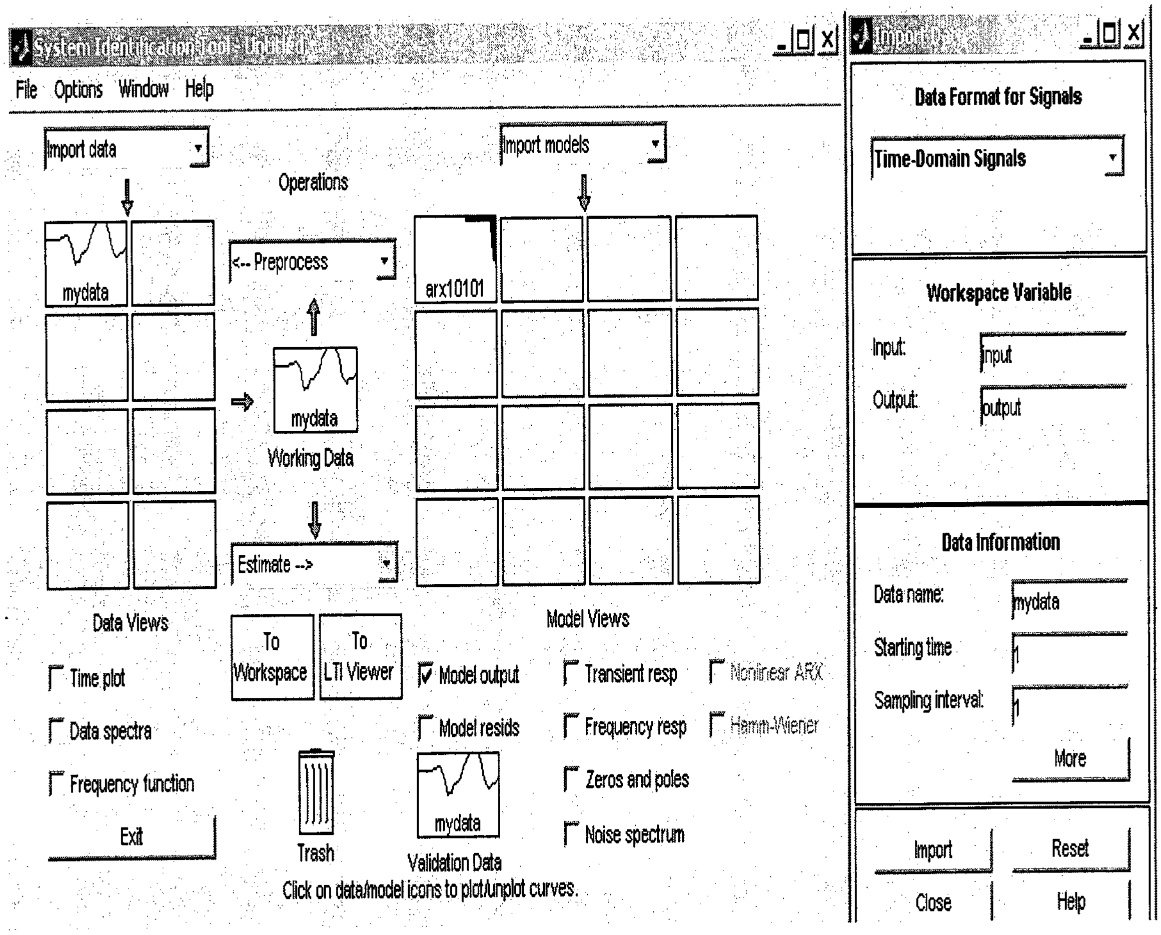

Time series DATA
DATA PREPROCESS; Denoised, Normalized
Simulated Output using ARX Model
Pattern Matching of Simulated and Measured Output
BINARY SEARCH
SEQUENTIAL SEARCH
DEFECTIVE SENSOR DETECTION

Figure 3.5 Flow chart describing methods for identifying defective sensor in structure.

To compare the measured and actual output data, system identification tool in MATLAB (Figure 3.6) has been used. This tool is already available in MATLAB. Some basic steps to construct the model are

1. Strain data from various sensors are first uploaded to MATLAB workspace.

2. 17 strain data are imported as input data and 1 sensor data is imported as output data. Then the tool builds the model. These data are collected from undamaged condition of the structure.
3. Again 17 strain data are imported as input data and 1 sensor data is imported as output data. This output data is used for validation. These data are collected from damaged condition of the structure.
4. Finally using the model the tool builds simulated output and compare it with measured output.

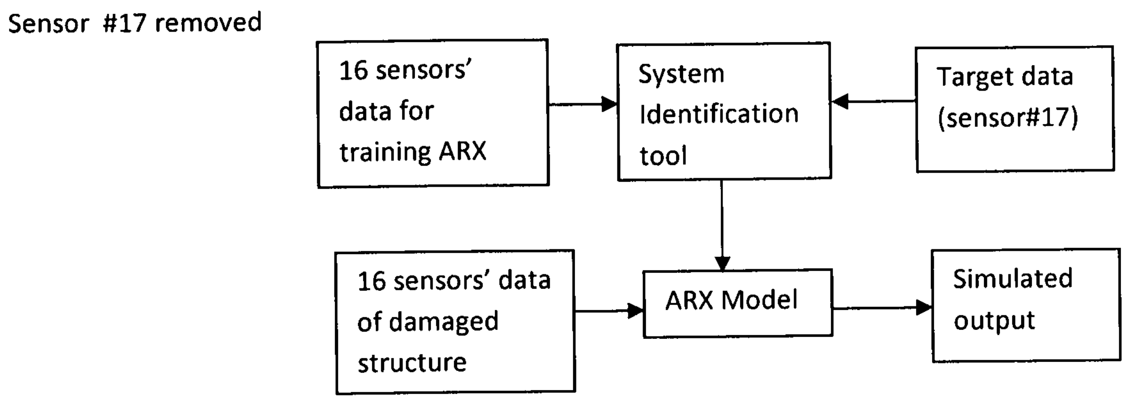


Input: input
Output: output
Data name: mydata
Starting time: 1
Sampling interval: 1

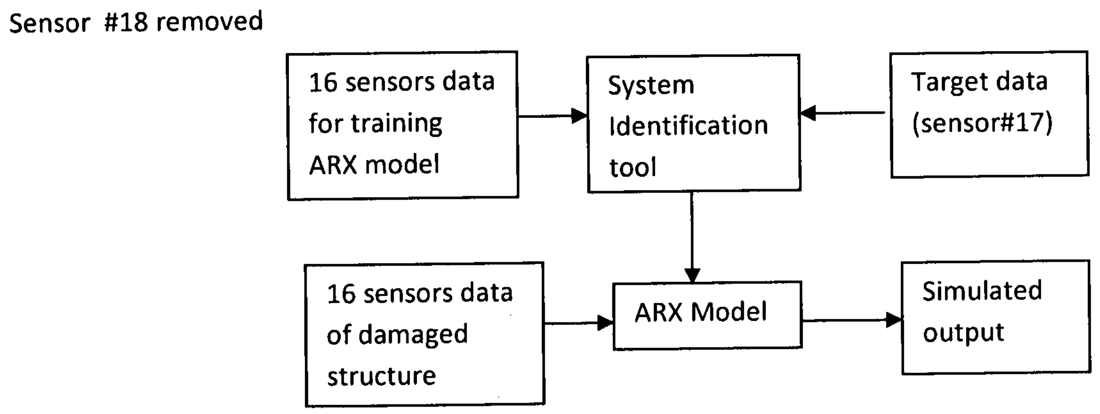
Figure 3.6 Graphical user interface of the system identification tool

3.6.3 ARX model with sequential search method

To build the ARX model to produce the data pattern at a particular period of time, December, January, February 2005. One sensor, in this study sensor 17, is always fixed as the target. For sequential search method 16 trained model has been developed. Each time, one sensor is removed from 17 sensors including temperature sensor data (sensor# 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, T) and used for training the model. Then corresponding sensors' data for a different time, March 2006, but similar environmental and operational condition, has been used as input to get 16 simulated outputs. In the Figure 3.7(a) it is shown that sensor 17 is used as target and for training the 1st model sensor 17 is removed from seventeen sensors, including the temperature sensor. Then, corresponding sensors' data of different time are used to get the 1st simulated output. In the Figure 3.7(b), it is shown that sensor 17 is used as the target and for training the 1st model. Sensor 18 is removed from seventeen sensors including temperature sensor. Then, corresponding sensors' data of different time are used to get 2nd simulated output. In this way, all other simulated outputs are obtained. Each simulated output pattern is compared with corresponding measured output shown in Figure 3.8. The highest best fit would indicate that the excluded sensor is responsible for the change in the data pattern. For instance if sensor 32 is removed from training data and input data and in that case measured and simulated output shows the highest best fit among all the 16 fits then it indicate sensor 32 is defective.


Sensor #17 removed
16 sensors' data for training ARX
System Identification tool
Target data (sensor#17)
16 sensors' data of damaged structure
ARX Model
Simulated output

(a)


Sensor #18 removed
16 sensors data for training ARX model
System Identification tool
Target data (sensor#17)
16 sensors data of damaged structure
ARX Model
Simulated output

(b)

Figure 3.7 Schematic diagrams for identifying defective sensor by ARX model with Sequential search method. (a) simulated output when sensor #17 is removed from training and actual data, (b) simulated output when sensor #18 is removed from training and actual data.

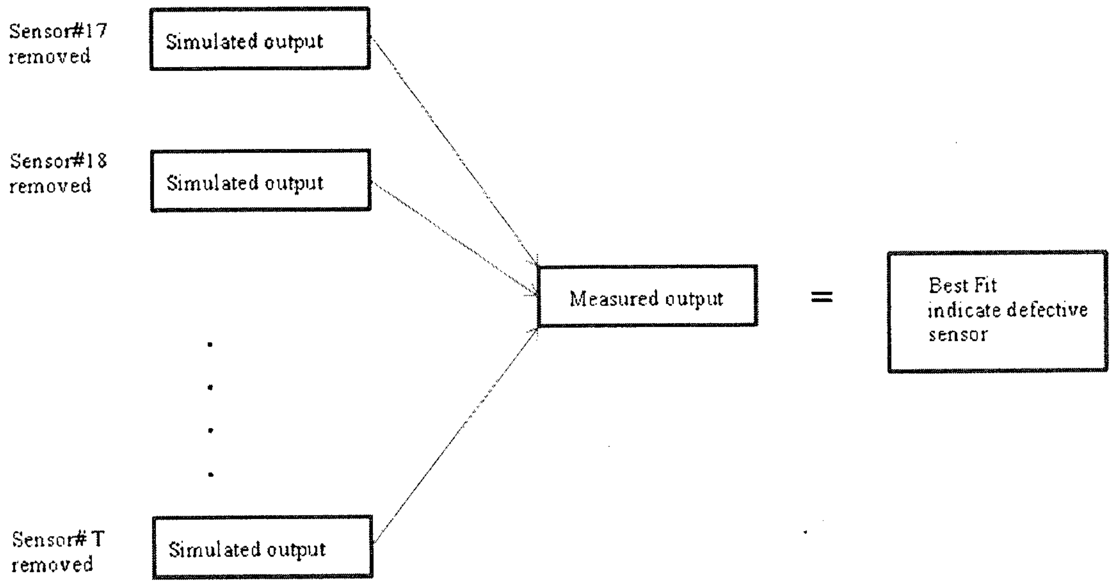


Figure 3.8 Schematic diagrams for comparing all simulated output with measured output. The highest best fit would indicate that the excluded sensor is responsible for the change in the data pattern.

3.6.4 ARX model with binary search method

To build ARX model to produce the data pattern at a particular period of time, December, January, February 2005. One sensor, in this study sensor 17, is always fixed as target. For the binary search method, the first 16 sensors including temperature sensor are divided into two groups. The first 8 sensors (17, 18, 19, 20, 21, 22, 23, 24) are used for training the model. Then the corresponding sensors' data for a different time, March 2006, but similar environmental and operational conditions, has been used as input to get simulated output. In the same way, the second 8 sensors (25, 26, 27, 28, 29, 30, 31, 32, T) are used for training the model. Then corresponding sensors data of different time, March 2006, but similar environmental and operational condition, has been used as input to get another simulated output. Each simulated output pattern is compared with the corresponding measured output. The best fit indicates a defective sensor is in other group. For instance, in Figure 3.9 it is shown that the best fit value of measured and simulated output of the 2nd group of sensors is higher than that of 1st group. So, the defective sensor is in 1st group. Then, the 1st group is divided into two group of four. In this way, the defective sensor is identified.

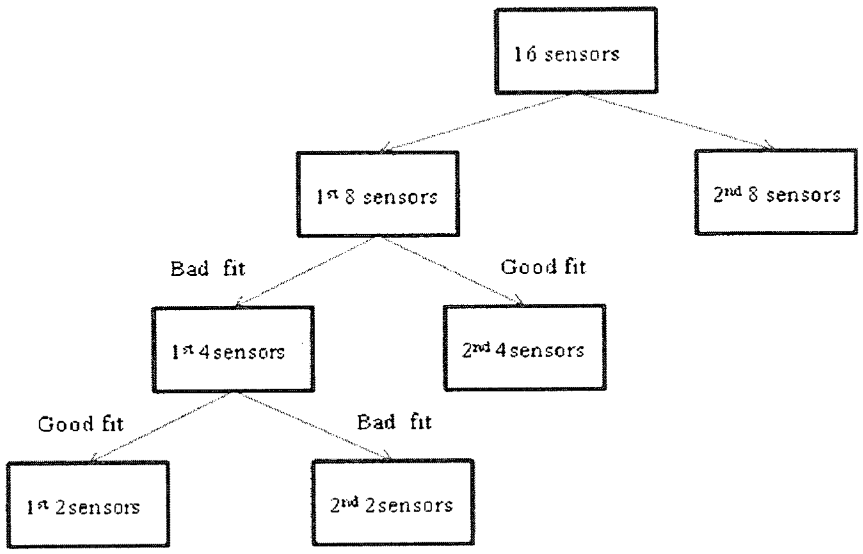


Figure 3.9 Schematic diagram for identifying defective sensor by ARX model with binary search method.

3.7 Sensitivity analysis

Sensitivity analysis is used to determine how "sensitive" a model is to changes in the value of the parameters of the model and to changes in the structure of the model. By showing how the model behavior responds to changes in parameter values, sensitivity analysis is a useful tool in model building as well as in model evaluation. Sensitivity analysis helps to build confidence in the model by studying the uncertainties that are often associated with parameters in models (Breierova and Choudhari, 1996). In this study, parameter sensitivity has been focused. Parameter sensitivity is usually performed as a series of tests in which the modeler sets different parameter values to see how a change in the parameter causes a change in the dynamic behavior of the stocks. Many parameters in system dynamics models represent quantities that are very difficult, or even impossible to measure to a great deal of accuracy in the real world. Also, some parameter values change in the real world. Sensitivity analysis allows the modeler to determine what level of accuracy is necessary for a parameter to make the model sufficiently useful and valid. If the tests reveal that the model is insensitive, then it may be possible to use an estimate rather than a value with greater precision. Sensitivity analysis can also indicate which parameter values are reasonable to use in the model. In Chapter 5, Case 1, sensitivity analysis has been conducted by replacing one sensor's data with random values. Also a number of test cases were studied in Chapters 4 and 5 to evaluate the sensitivity of the proposed methods. Such studies could be further expanded to study the effects randomness in parameters and organization of model constructs. However, based on the limited studies presented here, the proposed methods are found to be robust.

3.8 Summary

To take care of the issues discussed in the objective, two methods are proposed in this chapter. The first method, which is degree of similarity, is used to assess the structural condition and sensor reliability. Time series data pattern of the same sensor at different times and time series data pattern of two different sensors at the same time should always follow the similar pattern. The reason of their dissimilarity may either be defective sensor or damage in structure assuming other conditions i.e. load condition, environmental condition remain the same. The second method is identifying the defective sensor. In the case where removing one or two sensor data (one at a time), improves the fitness of the measured and simulated output pattern indicate that the excluded senor(s) might be defective. This can be used as a tool for testing of reliability of sensor data. But, in the case where removing a bunch of sensors' data (one at a time), does not improve the fitness of the measured and simulated output pattern, it is that there may be presence of damage in structure.

Chapter4: Assessing the relation among sensors' data in SHM.

4.1 General

In this chapter, statistical pattern recognition approach as described in Chapter 3 has been applied using acceleration and strain data of real structures to assess the reliability of sensor data. The goal here is to evaluate the feasibility of the proposed approach to structural damage detection and determination of defective sensor. To do the various tests described in the following sections, tool developed in Matlab has been used.

4.2 Test 1: Road test data

The data set in Test 1 contains a pair of time series data, one corresponding to vibration response of a heavy vehicle obtained during a road load test as mentioned in section 3.1, while the other corresponds to similar data produced using simulation. It is required to determine the level of similarity between the experimental and simulation data to test reliability of the simulation data. The results of the similarity test between the signals are outlined below.

1. 'Plot raw data' button plots the raw data to get preliminary idea of the relation between test data and simulation data, Figure 4.1.

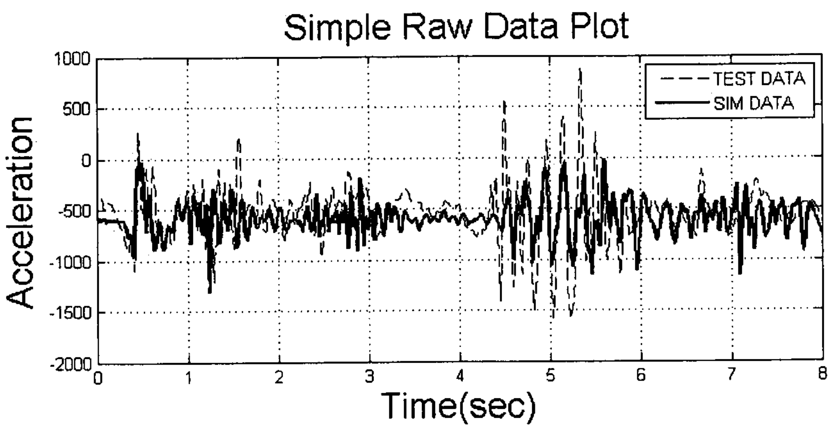


Figure 4.1 Plots of TEST DATA and SIM DATA in their raw forms

Table 4.1 Various statistical parameters on the time series data

	TEST DATA	SIM DATA
MEAN	-532.64	-606.34
STD	283.47	170.52
SKEWNESS	0.0788	-0.024

3. In this stage, the data are processed in a number of ways, such as removing the mean for each series to eliminate the effect of bias from both signals, and filtering the test data using Fourier Transform or de-noising using Wavelet Transform. This is performed by selecting the options in Data Preprocess (1) and Data Preprocess (2) panels sequentially. The correlation on the full range data is performed by clicking the 'CORRELATION (1)' button, which also calculates the lag between the signals to achieve maximum correlation. Results are shown in Table 4.2.

Table 4.2 Effect filtering and de-noising of time series data

Data Preprocess (1) (2)		Maximum Correlation	Lag time (sec)
Without filter	Simple Correlation	0.89	0
With filter(30Hz)		0.89	0
De noise (4)		0.91	0

4. In order to establish the relation between the two series, at different time windows, the series are segmented into multiple segments and each pair of them are analyzed. The results are shown in Tables 4.3, 4.4, 4.5.

5. Filtering (e.g., removing noise above 30 Hz) is performed by specifying the cut-off frequency (i.e., 30 Hz) in the text box and using the FFT button frequency analysis is performed. The frequency domain data does not seem to be very useful.

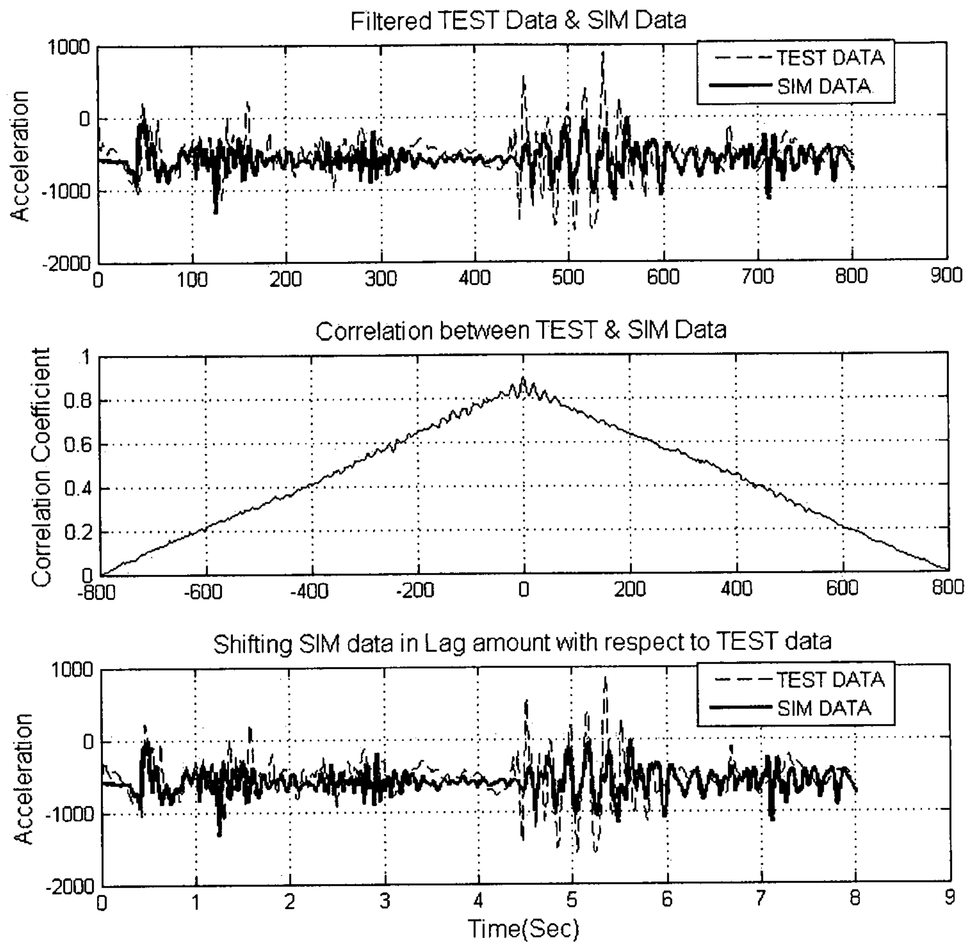


Figure 4.2 Correlation between TEST DATA and SIM DATA.

Table 4.3 Analysis of subsets of time series data – raw data

Data Range	Data Preprocess (1) (2)		Maximum Correlation	Lag time (sec)
1:200	Without filter	Simple Correlation	0.89	0.06
200:400			0.95	0
400:600			0.85	0
600:800			0.95	0

Table 4.4 Analysis of subsets of time series data – filtering

Data Range	Data Preprocess (1) (2)		Maximum Correlation	Lag time (sec)
1:200	With filter(30Hz)	Simple Correlation	0.896	0
200:400			0.94	0
400:600			0.85	0.09
600:800			0.94	0.12

From the results presented here, data sets are found to be similar. The correlation coefficient of 0.89 on the raw data or 0.91 on filtered or de-noised data indicates a high degree of similarity. The correlation improves when smaller time windows are used on data sets. The test data filtered in frequency domain using a low-pass filter with cut-off frequency of 30 Hz in one case, and de-noised with wavelet transform, in another case. The statistical comparison metrics such as correlation coefficient are not affected too much for these signals.

Table 4.5 Analysis of subsets of time series data – de-noising

Data Range	Data Preprocess (1) (2)		Maximum Correlation	Lag time (sec)
1:200	De noise	Simple Correlation	0.93	0
200:400			0.95	0
400:600			0.90	0
600:800			0.95	0

4.2 Test 2: Portage creek bridge data

The data in Test 2 contains a pair of time series representing strain data in two different time windows from a strain gauge (Channel Strain1_1_c1) installed on the bridge pier of Portage Creek Bridge. These statistical patterns of these two data series' are compared to determine whether the response of the structure at the corresponding time windows is similar. Data characteristics: Strain1_1_c1 data, Starting Time of Query:2003/12/25,11.00.00 and 2006/08/25,11.00.00, Time Interval between Points: 1sec, Number of Data Points: 302, Data Points (Comma Separated Values).

Table 4.6 Various statistical parameters on Strain1_1_c1 data

	DATA 2003/12/25	DATA 2006/08/25
MEAN	-505.52	165.49
STD	1.34	0.96
SKEWNESS	0.31	0.51

Table 4.7 Correlation results

Data Preprocess (1) (2)		Maximum Correlation	Lag time (sec)
Without filter	Vertical Shifting	0.74	0

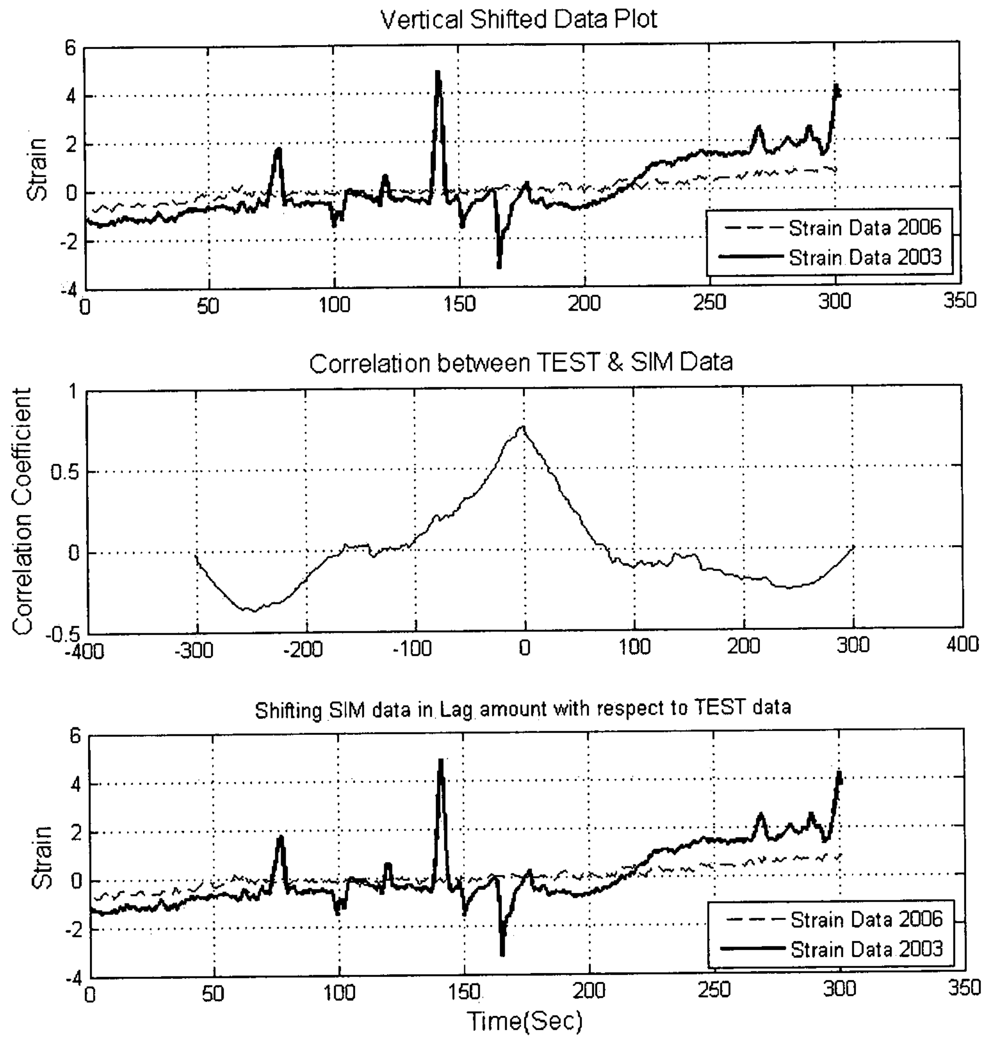


Figure 4.3 Correlation of two data sets, starting time of query 2003/12/25,11.00.00 and Starting Time of Query 2006/08/25,11.00.00.

The mean values of the two data sets are different and correlation value indicates they are 74% similar. The data sets are produced by the same sensor and time difference between them is two years eight months. So it can be said there might some change occur in structure during this period. Standard deviations are almost same and they skewed at the same direction that shows they follow same pattern which indicate the data sets are from same sensor.

4.3 Test 3: Portage creek bridge data

The data in Test 3 contains a pair of time series' representing the strain data in a given time window from two different strain gauges (Channels Strain1_1_c1 and Strain4_1_c2) installed on the bridge pier. The objective in this test is to identify the relationship between the data generated at the same time, but from different sensors. Data characteristics: Strain1_1_c1, Strain4_1_c2, Starting Time of Query:2006/04/01,10.00.00, Time Interval between Points: 1sec, Number of Data Points: 220, Data Points (Comma Separated Values).

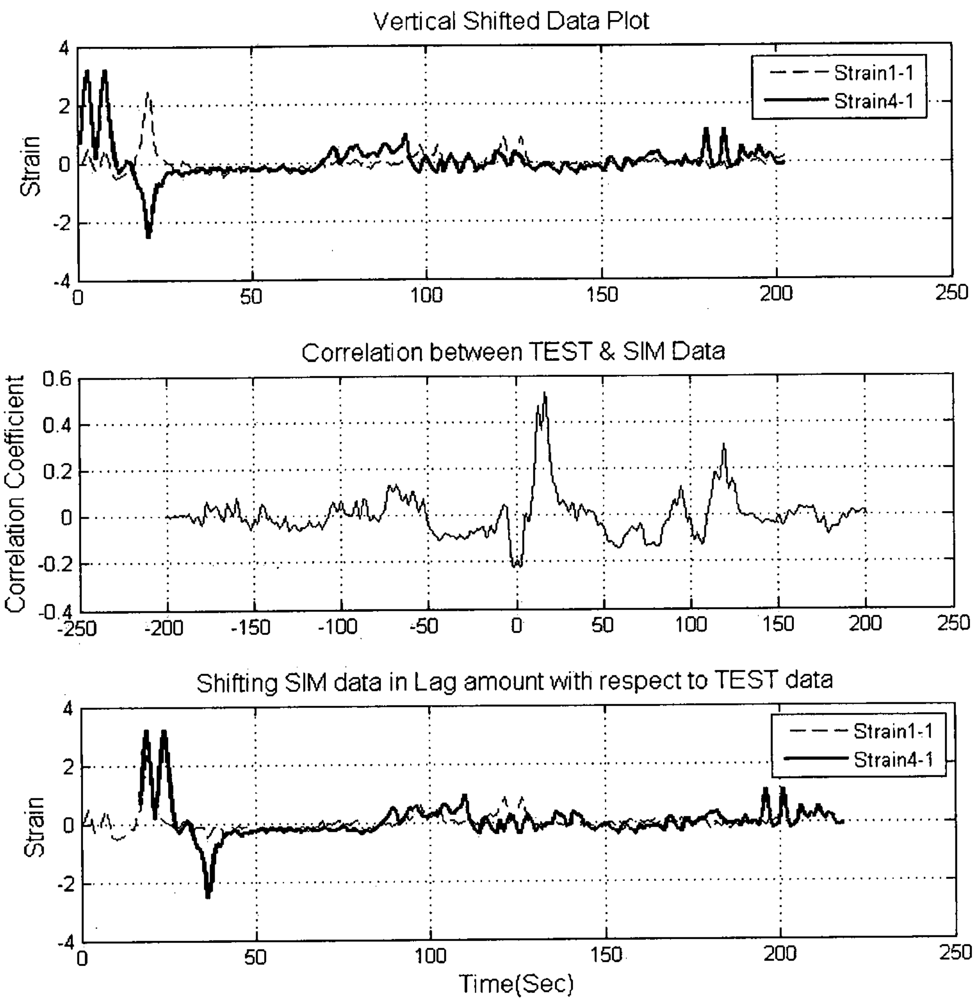


Figure 4.4 Correlation of Strain1_1_c1and Strain4_1_c2 data

Table 4.8 Summary of statistics on Strain1_1_c1 and Strain4_1_c2 data

	Strain1_1_c1	Strain4_1_c2
MEAN	-261.335	42.9626
STD	0.47	0.46
SKEWNESS	1.58	6.1

Table 4.9 Correlation of Strain1_1_c1and Strain4_1_c2 data

Data Preprocess (1) (2)		Maximum Correlation	Lag time (sec)
Without filter	Vertical Shifting	0.31	414.2

The mean values of the two data sets are different and correlation values indicate they are only 31% similar. The data sets are produces by two different sensors at the same time window. They are skewed in the same direction but their skewness values are quite different which indicates that the data sets are not similar.

4.4 Summary

In this chapter, various test results are presented to assess the structural condition and reliability of sensors' data by quantifying the degree of similarity between the pairs of sensor data. In Test 1, mean, standard deviation and skewness values of the two data sets are close, which indicate the simulation data are reliable. So, expensive and time consuming tests can be avoided generating data by simulation. In Test 2, the mean values of the two data sets are different, standard deviation and skewness values are close and correlation 74%. So, the degree of similarity of strain data in two different time window is high. It can be implied that structural conditions are not degraded from the initial condition. In Test 3, the mean, standard deviation values of the two data sets are not close. Correlation values indicate they are only 31% similar. They are skewed in the same direction but their skewness values are quite different which indicates that the degree of similarity is low because they are located in different location of the column. It can be implied that sensors are working reasonably.

Noman, 2008 worked on the same Portage Creek Bridge using different statistical pattern recognition algorithm. In his study, first AR process has been applied to extract coefficients which are then statistically modeled for damage classification by X-bars. From the X-bars of strain and vibration readings, percentage of outliers found was not so high to indicate any damage in the structure or structural degradation. Secondly, pattern comparison based on fitting of the reference models to test blocks was performed. Computed R- values that represent the goodness of fit also did not show any trend or consistent discrepancies to indicate any damage in the structure. The comparison of two different approach of same structure show consistent results which justify the proposed statistical approach of the current thesis.

Chapter5: Detection of defective sensors in SHM

5.1 General

In this chapter, the damage detection method using statistical pattern recognition approach as described in Chapter 3 has been applied to acceleration and strain data of real structures. The goal here is to evaluate the feasibility of the proposed approach to structural damage detection and determination of defective sensor. This chapter demonstrates how statistical pattern recognition methods can be used for detection of defective sensor or detection of damage in the structure.

5.2 Case 1: Identifying a sensor known to be defective (sequential search technique)

In the first run it has found that there is a spike in 2nd half of the March 2006 which is shown in Figure 5.1. In this figure dotted line represents measured output and solid line represents simulated output. It is assumed that all sensors are in good condition in the1st half of the month March 06. So Simulated output generated by feeding all 15 sensors data of 1ST 15 days of March 2006 to the ARX model. Each time one sensor is removed from input data to see which output give best fit. In Case 1, sensor 6-1 has been made defective by replacing original data by random numbers. In Figure 5.2 it is found that

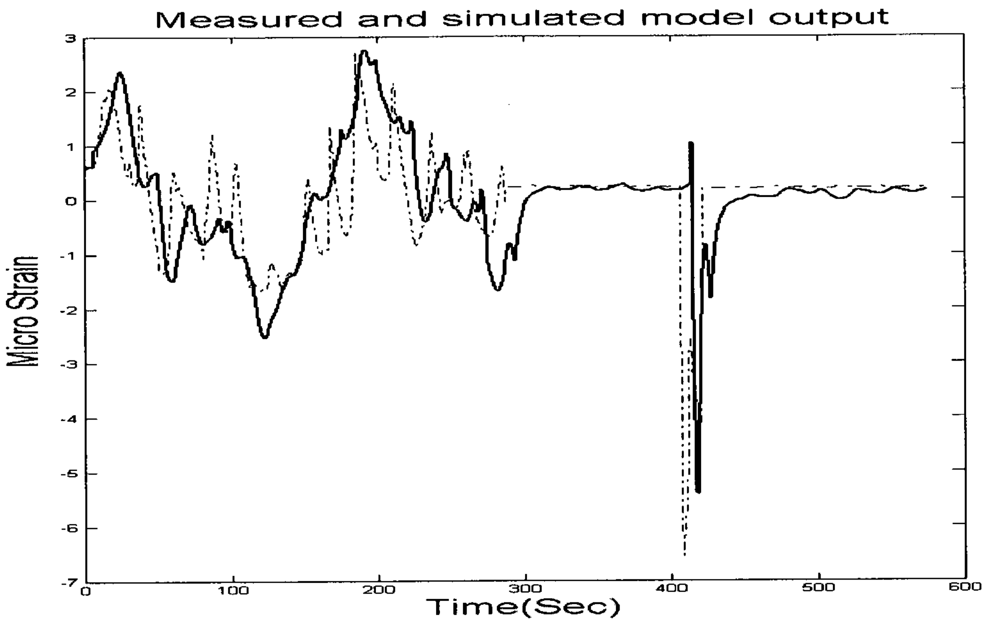


Figure 5.1 Measured and Simulated output of March 2006

when sensor 2-1 is removed from the input then measured and simulated output does not fit well, -267%. In this way all the 16 combination is tested and finally best fit is taken.

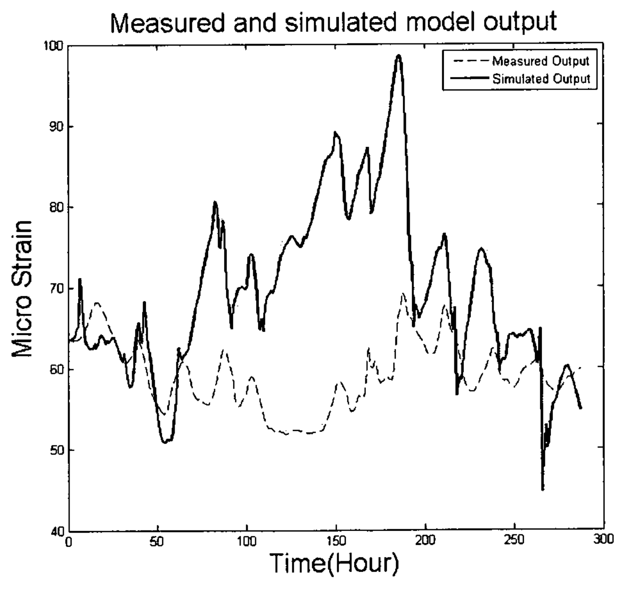


Figure 5.2 Measured and Simulated output of March 2006, sensor# 2-1 removed

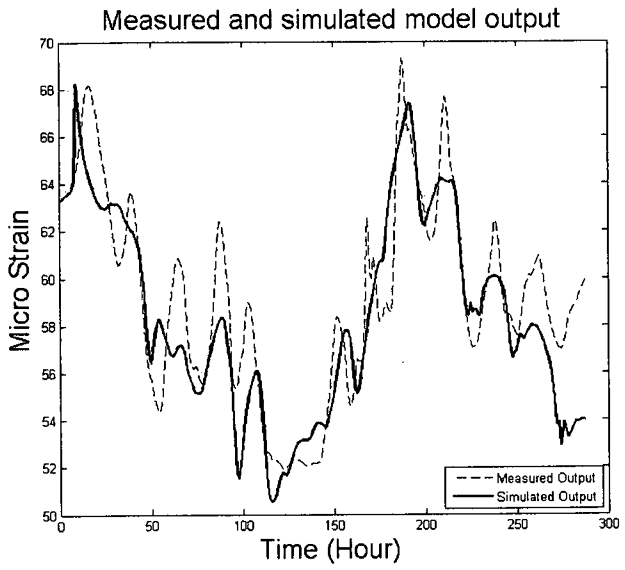


Figure 5.3 Measured and Simulated output of March 2006, sensor# 6-1 removed

It is found that when sensor 6-1 is removed from the input then the simulated output fit best with the measured output which indicates sensor 6-1 is defective. In Table 5.3 when sensor 6-1 is removed measured and simulated output gives maximum positive fitting 45%. So this method works to detect malfunction sensor. While the sample figures are presented here in discussing results produced by proposed algorithms, remaining figures are presented in appendix B. It also shows random change of sensor data affect ARX model.

Table 5.1 Fitting percentage of measured and simulated output for each sensor removal

Sensor Removed	Fit (%)
1-2	31.6
2-1	-267
2-2	36.9
3-1	40.5

3-2	35.5
4-1	-26.5
4-2	24.4
5-1	21.7
5-2	36
6-1	45
6-2	-32
7-1	38
7-2	43
8-1	-59
8-2	-36
9-1	35

5.3 Case 2: Identifying unknown defective sensor (sequential search technique)

In Figure 6.1 it is found that defective sensors are in 2nd half of the of the March 2006. So Simulated output generated by feeding all 15 sensors data of 2nd 15 days of March 2006 to the ARX model. Each time one sensor is removed from input data to see which output give best fit to detect the defective sensor. In Figure 5.4 it is found that when sensor 1-2 is removed from the input then measured, dotted line and simulated, solid line in the figure, output does not fit 10.54% and in Figure 5.5 it is found that when sensor 4-2 is removed from the input then measured and simulated output does not fit well, -109.7%. In this way all the 16 combination is tested and finally best fit is taken.

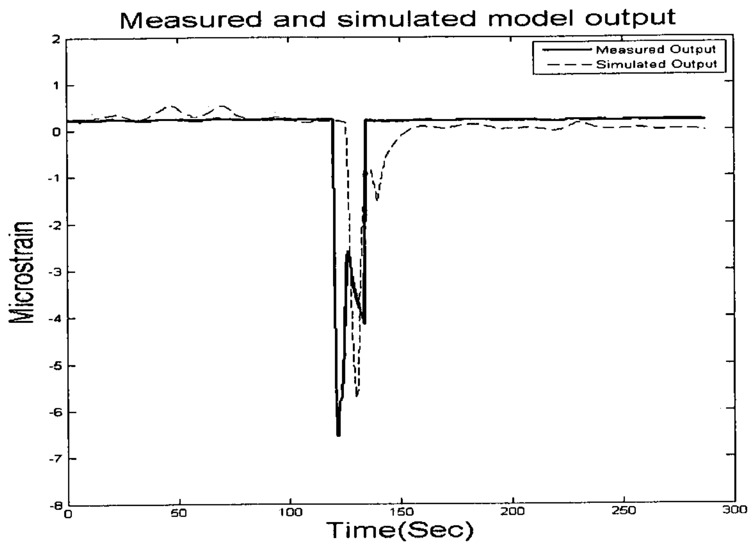


Figure 5.4 Measured and Simulated output of March 2006, sensor# 1-2 removed

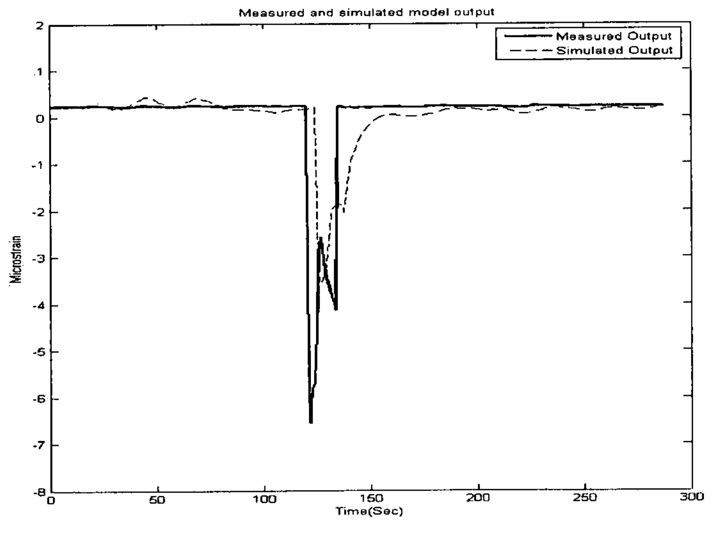


Figure 5.5 Measured and simulated output of March 2006, sensor# 4-2 removed

In Table 5.2 shows that when sensor 4-2 is removed measured and simulated output gives maximum positive fitting in percentage. So sensor 4-2 is the defective sensor While the sample figures are presented here in discussing results produced by proposed algorithms, remaining figures are presented in appendix C.

Table 5.2 Fitting percentage of measured and simulated output for each sensor removal

Sensor removed	Fit (%)
1-2	10.54
2-1	5.94
2-2	-8.32
3-1	7.98
3-2	7.95
4-1	8.53
4-2	15.28
5-1	11.2
5-2	4.664
6-1	-22.95
6-2	-109.7
7-1	8.22
7-2	10.1
8-1	7.89
8-2	0.34
9-1	10.48

5.4 Case 3: Identifying unknown defective sensor (binary search technique)

In Case 3, binary search method has followed. In this case first 16 sensors are divided into 2 groups of 8 sensors. Feeding 1st 8 sensors data as input to trained ARX model and sensor 1-1 as target or measured output. It is observed in Figure 5.6 and Figure 5.7 that measured and simulated model output of 2nd 8 sensors fit well than 1st 8 sensors.

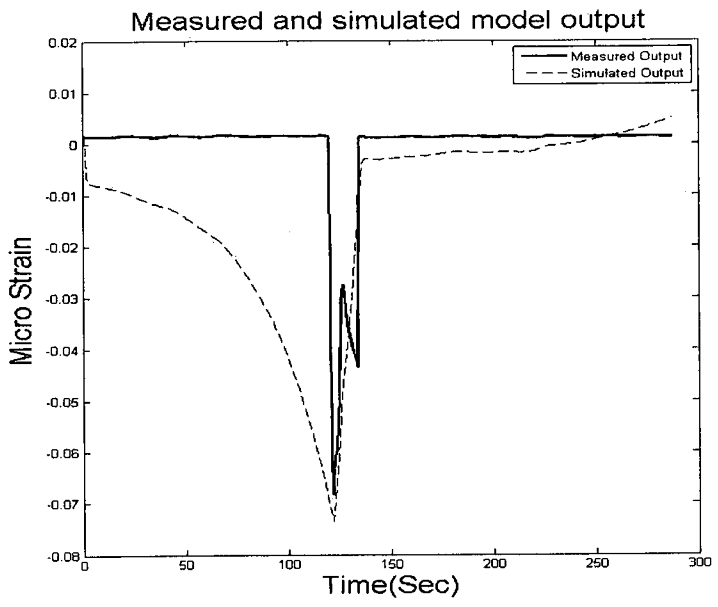


Figure 5.6 Measured and Simulated output of March 2006, 1st 8 sensors

Therefore malfunction sensors are located in 1st 8 sensors group, Table 5.3 (a).

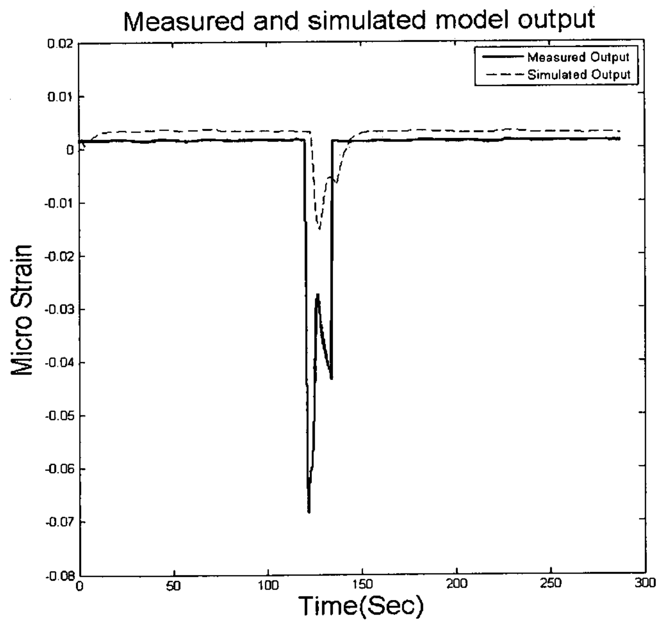


Figure 5.7 Measured and Simulated output of March 2006, 2nd 8 sensors

Dividing 1st 8 sensors data into 2 sub groups of 4 sensors and repeating the same procedure it is found that measured and simulated model output of 1st 4 sensors does not fit well than 2nd 4 sensors. Therefore malfunction sensor is located in 1st 4 of 1st 8 sensors group, Table 5.3 (b). More subgroup reduces the data points so much that both subgroups do not give good results. Now sequential search can be used to find the defective sensor. Finally it is found that 4-2 is defective. While the sample figures are presented here in discussing results produced by proposed algorithms, remaining figures are presented in appendix D.

Table 5.3 Fitting percentage of measured and simulated output

Sensors	Fit (%)	Comments
1st 8	-94.34	Defective sensor group
2nd 8	7.01	Non-defective sensor group

(a)

Sensors	Fit (%)	
1st 4 of defective group	-10.49	Defective sensor group
2nd 4 of defective group	3.03	Non-defective sensor group

(b)

Sensors	Fit (%)	
1st 2 of defective group	49.63	Non-defective sensor group
2nd 2 of defective group	-21.92	Defective sensor group

(c)

Sensors	Fit (%)	
1st of defective group	-2.32	
2nd of defective group	-27.66	Defective sensor

(d)

5.5 Summary

In this chapter the proposed binary and sequential search methods for detecting defective sensors have been tested using the data obtained from the portage Creek Bridge. Several test cases have been presented to show the effectiveness of the proposed methods. In Case 1, it is found that when sensor 6-1 is removed from the input data then the simulated output fit best to measured output. In Case 2, defective sensor was unknown initially. Using the same methodology of Case 1, it is found that when sensor 4-2 is removed from the input data then the simulated output fit best to measured output. So sensor 4-2 is defective. In Case 3, it is found that the same sensor that has been found defective with sequential search is detected correctly with the binary search technique which is faster than sequential search technique. Bagchi et al. (2007) describes the implementation of intelligent sensing for the remote health monitoring of the seismic strengthened pier of the Portage Creek Bridge. In that study it is found that both strain gauge 4-1 and 6-1 experienced an increase of strains during dynamic test. The comparison of two study show consistent result which justify the proposed statistical approach of the thesis. Besides, Case 1 shows Sensitivity analysis

Chapter6: Summary and Conclusions

6.1 Summary

Structural Health Monitoring (SHM) has been widely studied during the past two decades and significant progress has been achieved through the development of new sensors and system that are capable of monitoring the performance of a structure. In literature review it is found that structural damage affects the dynamic properties of a structure, causing a change in the vibration signals i.e. strain and acceleration time histories. Damage detection can be performed using time series analysis of vibration signals measured from a structure before and after damage. In this thesis, two methods are proposed.

The first method, which is degree of similarity, is used to assess the structural condition and sensor reliability. Time series data pattern of the same sensor at different times and time series data pattern of two different sensors at the same time should always follow the similar pattern. The reason of their dissimilarity may be due to either defective sensor or damage in structure assuming other conditions i.e. load condition, environmental condition remain the same. Numerous tests have been done to assess the structural condition and reliability of sensors' data by quantifying the degree of similarity between the pairs of sensor data. In chapter 4, three test results have been presented. In Test 1, mean, standard deviation and skewness values of the two data sets are found to be close, which indicate that the simulation data are reliable. Thus, expensive and time consuming tests can be avoided by generating data using simulation. In Test 2, while the mean values of the two data sets from the same sensor, but taken at different time windows are found to be different, the standard deviation and skewness values are close and correlation 74%. Thus, the degree of similarity of strain data in two different time window can be considered high. It can be implied that structural conditions are not degraded from the initial condition. In Test 3, data from two different sensors at the same time window have been considered, where the mean, standard deviation values of the two data sets are not found to be close, correlation values indicate they are only 31% similar, and while the data sets are skewed in the same direction, their skewness values are quite different. It can be implied that sensors are different and the data are related. In all cases the sensors can be said to be working reasonably for the time windows considered in the study. Finally, the test results were also compared with Noman (2008) who worked on the bridge data using different statistical pattern recognition algorithms. The comparison of two different approach of same structure show consistent results.

The second method is proposed for identifying defective sensors or damage. In the case, the statistical pattern recognition techniques such as, ARX model have been used to develop a method for automatically build relationships among the data from various sensors installed in a structure. Such relationships are tracked over time and abnormal changes in the data patterns are identified using predefined metrics (e.g., coefficient of determination between the simulated and real data from a reference sensor). In case of a deviation in the data pattern, the data segment responsible is processed by eliminating one or more sensors from the input vector to the ARX model utilizing the concept of the well known sequential and binary search techniques. In the event of removing a particular set or sensors at a time improves the fitness of the established relationship among the remaining sensors, the excluded sensors are identified as the responsible ones for the change in the relationship. Once the responsible sensor is identified, physical tests could be conducted to ascertain if it indeed malfunctioned. On the other hand, if a group of sensors is found to be responsible, it is likely that the structural or load condition has changed. In that case, further analysis of the SHM and structural systems would be necessary. In the absence of unusual patterns, the relationships would be simply updated with the new data, and compared with the initial pattern of relationship to determine the rate of gradual change in the data pattern which would indicate the rate of deterioration in the structure. Numerous tests have been performed to identify the defective sensors or anomaly in the structural response at different time segments. In chapter 4, three case study results have been presented. In Case 1, it is found that when sensor 6-1 with known error in the data is removed from the input set, the simulated output fits best to the measured output. The sequential search technique as proposed here has been used in this case. In Case 2, the defective sensor was unknown initially. Using the proposed methodology in the same manner as in Case 1, it is found that when Sensor 4-2 is removed from the input set, the simulated output fit best to measured output. In that case Sensor 4-2 may be defective, which should be verified using manual tests. In Case 3, using the binary search technique as proposed here, the same sensor has been detected as the defective one as in the case with sequential search in Case 2. As the binary search technique is faster than sequential search technique, it should be considered in practical implementation of the proposed methods. Finally, the test results were also compared with that reported in Bagchi et al. (2007) which describes the implementation of intelligent sensing for the remote health monitoring of the seismic strengthened pier of the Portage Creek Bridge. In that study it is found that both strain gauge 4-1 and 6-1 experienced an increase of strains during dynamic test. The comparison of two studies shows consistent results.

6.2 Conclusions

- To determine the reliability of the sensors' data and to assess the structural condition, a set of statistical parameters, mean, standard deviation, skewness, correlation can be used in a holistic manner as demonstrated in the present study.
- The ARX model with proposed sequential search technique and or binary search techniques can be used successfully used in identifying defective sensor or changes in the behavior of the structure.
- The information about the degree of similarity among various sensors data in a structure and detection of defective sensors using the proposed methods can be extended to detect damage in structure.
- Low degree of similarity among sensors data or multiple sensors detected as the responsible ones for the change of the statistical patterns of data indicates either the presence of damage/degradation in structure or change in load/environmental conditions.
- Limited sensitivity analysis shows that the proposed methods are robust.

6.3 Limitations and Future works

- In the present study, it is assumed that data dissimilarity occurs due to the presence of damage in structure. But data dissimilarity may be also occurs due to the presence of individual or combined effect of sensor malfunction or excessive load or presence of damage in the structure. Further work should consider these effects in isolating the individual factors.
- For correlation analysis, there is no concrete benchmark to rating the similarity. the coefficient of determination (i.e., R^2 measure) or other similarity metric might be used to define the rating.
- For the measured and simulated output, the best fit algorithm used here may not be sufficient. Other metrics could be used as mentioned above to determine the degree of fitness between simulated and actual data.
- To validate the results of the proposed methods which are based on statistical pattern recognition techniques, other methods like neural network might be used.
- For sensitivity analysis limited cases are studied which needs more elaborate work.

References

1. Bagchi, A.; Huffman, S.; Mufti, A.A.; "Seismic strengthening and SHM of Portage Creek Bridge". Ninth Canadian conference on earthquake engineering, Ottawa, Canada, June, 2007.
2. Breierova,L.; Choudhari,M.;"An Introduction to Sensitivity Analysis". MIT System Dynamics in Education Project. September 6, 1996.
3. Chase,S.B.; Washer,G.; " Non –destructive evaluation for bridge management in the next century". Public road, 61 (1), Available from http://www.tfhrc.gov/pubrds/july97/ndejuly.htm; 1997.
4. CINDE, Canadian Institute for Non Destructive Evaluation, 2008. http://www.cinde.ca/ndt.shtml.
5. da Silva, S.; Junior, M.D.; Junior, V.L.; "Damage detection in a benchmark structure using AR-ARX models and statistical pattern recognition". Journal of the Brazilian Society of Mechanical Sciences and Engineering, Vol. 29 no.2 Rio de Janeiro, June, 2007.
6. Dubin, E.E.; Yanev B.S.; "Managing the east river bridge in New York city",Proceedings of the 7th international symposium on smart structures and materials, Newport, CA, 4-8 March, 2001.
7. Fugate, M.; Sohn,H.; Farrar, C.R.; "Vibration-Based Damage Detection Using Statistical Process Control", Mechanical Systems and Signal Processing, Vol.15(4), 707-721, 2001.
8. Farrar, C. R.; Doebling, S.W.; "An overview of modal-based damage identification methods". Proc. of EUROMECH 365 International Workshops: DAMAS 97- Structural Damage Assessment Using Advanced Signal Processing Procedures, June/July, Sheffield, U.K., 1997.
9. Huffman,S.; Bagchi, A.; Mufti,A.; Neale,K. ; Sargent,D.; Riveraa,E.; "GFRP seismic strengthening and structural health monitoring of portage creek bridge concrete columns".The Arabian Journal for Science and Engineering, Volume 31, Number 1C, Dec, 2006.

10. Humar. J.; Bagchi, A. and Xu, H.; "Performance of vibration based techniques for the identification of structural damage, SHM-An international Journal, 5(3): 215-227; 2006
11. Knuth, D. The Art of Computer Programming, Volume 3: Sorting and Searching, Third Edition. Addison-Wesley, 1997. ISBN 0-201-89685-0. Section 6.2.1: Searching an Ordered Table, pp. 409–426.
12. Mufti,A.A.; "Guidelines for structural health monitoring". ISIS Canada design manual No.2, Sept. 2001.
13. Nair, K.K.; Kiremidjian A.S.; "A comparison of local damage detection algorithm based on statistical processing of vibration based measurements". SHM and intelligent Infrastructure –Qu,Li & Duan, Taylor & Francis Group, London, 2006.
14. Noman, A.S.; "Application of vibration based methods and statistical pattern recognition techniques to structural health monitoring". M.A.Sc thesis, Concordia University, 2008.
15. Phares, B.; "Highlights of study of reliability of visual inspection", Presentation at the Annual Meeting of TRB Subcommittee A2C005 (1) non-destructive Evaluation of Structures, 2001.
16. Ping, Lu; "A statistical based damage detection approach for highway bridge structural health monitoring", Ph.D dissertation, Iowa State University, 2008.
17. Rytter,A; "Vibration based inspection of civil engineering structure. Ph.D. dissertation, Department of building Technology and Structural Engineering, university of Aalborg, Denmark, 1993.
18. Roy A.G; Biron P.M; Lapointe M.F."Implications of low-pass filtering on power spectra and autocorrelation functions of turbulent velocity signals".Source: Mathematical Geology, Volume: 29,Issue: 5,Pages: 653-668,Published: Jul 1997.
19. Sohn, H.; Czarnecki,J. J.; Farrar, C.R.; "Structural Health Monitoring Using Statistical Process Control", Journal of Structural Engineering, Vol. 126 (11), 1356-1363, 2000.
20. Struzik, Z.R.;Siebes, A."The Haar Wavelet Transform in the Time Series Similarity Paradigm", Principles of Data Mining and Knowledge Discovery.

Third European Conference, PKDD'99. Proceedings. (Lecture Notes in Artificial Intelligence Vol.1704), pp 12-22, 1999.

21. Sohn, H.; Farrar,C.R.; Hunter N. F.; and Worden, K.; "SHM using statistical pattern recognition techniques", Transactions of the ASME, Vol.123, 2001.
22. Sohn, H.; Czarnecki,J. J.; Farrar, C.R.; "Structural Health Monitoring Using Statistical Process Control", Journal of Structural Engineering, Vol. 126 (11), 1356-1363, 2000.

Appendix A

A.1 Detail Matlab code for the tool mentioned in Chapter 5.

```
function varargout = ITEC(varargin)
% ITEC M-file for ITEC.fig
%      ITEC, by itself, creates a new ITEC or raises the existing
%      singleton*.
%
%      H = ITEC returns the handle to a new ITEC or the handle to
%      the existing singleton*.
%
%      ITEC('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ITEC.M with the given input
arguments.
%
%      ITEC('Property','Value',...) creates a new ITEC or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before ITEC_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to ITEC_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ITEC

% Last Modified by GUIDE v2.5 25-May-2008 19:49:30

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ITEC_OpeningFcn, ...
                   'gui_OutputFcn',  @ITEC_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before ITEC is made visible.
function ITEC_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ITEC (see VARARGIN)

% Choose default command line output for ITEC
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ITEC wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = ITEC_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in POPULATE_button.
function POPULATE_button_Callback(hObject, eventdata, handles)
% hObject    handle to POPULATE_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
update_listbox(handles)

function update_listbox(handles)
% hObject    handle to update (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listbox1 contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from
listbox1

% Updates the listbox to match the current workspace
vars = evalin('base','who');
set(handles.TEST_listbox,'String',vars)
set(handles.SIM_listbox,'String',vars)
set(handles.TIME_listbox,'String',vars)

function [var1] = get_var_namest(handles)
```

```
% Returns the names of the two variables to plot
list_entries = get(handles.TEST_listbox,'String');
index_selected = get(handles.TEST_listbox,'Value');
var1 = list_entries{index_selected(1)};


function [var2] = get_var_namess(handles)
% Returns the names of the two variables to plot
list_entries = get(handles.SIM_listbox,'String');
index_selected = get(handles.SIM_listbox,'Value');
var2 = list_entries{index_selected(1)};

function [var3] = get_var_namesti(handles)
% Returns the names of the two variables to plot
list_entries = get(handles.TIME_listbox,'String');
index_selected = get(handles.TIME_listbox,'Value');
var3 = list_entries{index_selected(1)};


% --- Executes on selection change in TEST_listbox.
function TEST_listbox_Callback(hObject, eventdata, handles)
% hObject    handle to TEST_listbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns TEST_listbox contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from
TEST_listbox


% --- Executes during object creation, after setting all properties.
function TEST_listbox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to TEST_listbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in SIM_listbox.
function SIM_listbox_Callback(hObject, eventdata, handles)
% hObject    handle to SIM_listbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns SIM_listbox contents
as cell array
```

```
%       contents{get(hObject,'Value')} returns selected item from
SIM_listbox


% --- Executes during object creation, after setting all properties.
function SIM_listbox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SIM_listbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in TIME_listbox.
function TIME_listbox_Callback(hObject, eventdata, handles)
% hObject    handle to TIME_listbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns TIME_listbox contents
as cell array
%       contents{get(hObject,'Value')} returns selected item from
TIME_listbox


% --- Executes during object creation, after setting all properties.
function TIME_listbox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to TIME_listbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function TIME_edit_Callback(hObject, eventdata, handles)
% hObject    handle to TIME_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of TIME_edit as text
%        str2double(get(hObject,'String')) returns contents of
TIME_edit as a double


% --- Executes during object creation, after setting all properties.
function TIME_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to TIME_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function FREQ_edit_Callback(hObject, eventdata, handles)
% hObject    handle to FREQ_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.Data.FREQ_edit=str2double(get(hObject,'String'));
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of FREQ_edit as text
%        str2double(get(hObject,'String')) returns contents of
FREQ_edit as a double


% --- Executes during object creation, after setting all properties.
function FREQ_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FREQ_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function SAMPLE_edit_Callback(hObject, eventdata, handles)
% hObject    handle to SAMPLE_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.Data.SAMPLE_edit=str2double(get(hObject,'String'));
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of SAMPLE_edit as text
%       str2double(get(hObject,'String')) returns contents of
SAMPLE_edit as a double


% --- Executes during object creation, after setting all properties.
function SAMPLE_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SAMPLE_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function FILTER_edit_Callback(hObject, eventdata, handles)
% hObject    handle to FILTER_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.Data.FILTER_edit=str2double(get(hObject,'String'));
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of FILTER_edit as text
%       str2double(get(hObject,'String')) returns contents of
FILTER_edit as a double


% --- Executes during object creation, after setting all properties.
function FILTER_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FILTER_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function MAXCORR_edit_Callback(hObject, eventdata, handles)
% hObject    handle to MAXCORR_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of MAXCORR_edit as text
%       str2double(get(hObject,'String')) returns contents of
MAXCORR_edit as a double


% --- Executes during object creation, after setting all properties.
function MAXCORR_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MAXCORR_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function LAG_edit_Callback(hObject, eventdata, handles)
% hObject    handle to LAG_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of LAG_edit as text
%        str2double(get(hObject,'String')) returns contents of LAG_edit
as a double


% --- Executes during object creation, after setting all properties.
function LAG_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to LAG_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function RATING_edit_Callback(hObject, eventdata, handles)
% hObject    handle to RATING_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of RATING_edit as text
%        str2double(get(hObject,'String')) returns contents of
RATING_edit as a double
```

```
% --- Executes during object creation, after setting all properties.
function RATING_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to RATING_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function MEANTEST_edit_Callback(hObject, eventdata, handles)
% hObject    handle to MEANTEST_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of MEANTEST_edit as
text
%        str2double(get(hObject,'String')) returns contents of
MEANTEST_edit as a double


% --- Executes during object creation, after setting all properties.
function MEANTEST_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MEANTEST_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function MEANSIM_edit_Callback(hObject, eventdata, handles)
% hObject    handle to MEANSIM_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of MEANSIM_edit as text
%        str2double(get(hObject,'String')) returns contents of
MEANSIM_edit as a double
```

```
% --- Executes during object creation, after setting all properties.
function MEANSIM_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MEANSIM_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function STDTEST_edit_Callback(hObject, eventdata, handles)
% hObject    handle to STDTEST_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of STDTEST_edit as text
%        str2double(get(hObject,'String')) returns contents of
STDTEST_edit as a double


% --- Executes during object creation, after setting all properties.
function STDTEST_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to STDTEST_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function STDSIM_edit_Callback(hObject, eventdata, handles)
% hObject    handle to STDSIM_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of STDSIM_edit as text
%        str2double(get(hObject,'String')) returns contents of
STDSIM_edit as a double


% --- Executes during object creation, after setting all properties.
```

```
function STDSIM_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to STDSIM_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function STDRATIO_edit_Callback(hObject, eventdata, handles)
% hObject    handle to STDRATIO_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of STDRATIO_edit as
text
%        str2double(get(hObject,'String')) returns contents of
STDRATIO_edit as a double


% --- Executes during object creation, after setting all properties.
function STDRATIO_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to STDRATIO_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function SKEWTEST_edit_Callback(hObject, eventdata, handles)
% hObject    handle to SKEWTEST_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of SKEWTEST_edit as
text
%        str2double(get(hObject,'String')) returns contents of
SKEWTEST_edit as a double


% --- Executes during object creation, after setting all properties.
```

```
function SKEWTEST_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SKEWTEST_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function SKEWSIM_edit_Callback(hObject, eventdata, handles)
% hObject    handle to SKEWSIM_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of SKEWSIM_edit as text
%        str2double(get(hObject,'String')) returns contents of
SKEWSIM_edit as a double


% --- Executes during object creation, after setting all properties.
function SKEWSIM_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SKEWSIM_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function BIN_edit_Callback(hObject, eventdata, handles)
% hObject    handle to BIN_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.Data.BIN_edit=str2double(get(hObject,'String'));
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of BIN_edit as text
%        str2double(get(hObject,'String')) returns contents of BIN_edit
as a double


% --- Executes during object creation, after setting all properties.
function BIN_edit_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to BIN_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in HIST_button.
function HIST_button_Callback(hObject, eventdata, handles)
% hObject    handle to HIST_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
figure()
[x] = get_var_namest(handles);
[y] = get_var_namess(handles);
TEST=evalin('base',x);
SIM=evalin('base',y);
%histogram and fitting a normal distribution to the data.
subplot(2,1,1)
histfit(TEST);title('TEST DATA');ylabel('Frequency (counts)');
[historic_mean, historic_stdev1] = normfit(TEST);
set(handles.MEANTEST_edit,'String',historic_mean);
set(handles.STDTEST_edit,'String',historic_stdev1);
subplot(2,1,2)
histfit(SIM);title('SIM DATA');ylabel('Frequency (counts)');
[historic_mean, historic_stdev2] = normfit(SIM);
set(handles.MEANSIM_edit,'String',historic_mean);
set(handles.STDSIM_edit,'String',historic_stdev2);
ratio=historic_stdev1/historic_stdev2;
set(handles.STDRATIO_edit,'String',ratio);



% --- Executes on button press in FULLRANGE_CORR_button.
function FULLRANGE_CORR_button_Callback(hObject, eventdata, handles)
% hObject    handle to FULLRANGE_CORR_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[x] = get_var_namest(handles);
[y] = get_var_namess(handles);
[z] = get_var_namesti(handles);
TEST=evalin('base',x);
SIM=evalin('base',y);
TIME=evalin('base',z);
Fs=length(TEST)/max(TIME);
p=length(TEST);
if get(handles.WITHOUTFILTER_radiobutton,'Value')  ==
get(hObject,'Max')
    if get(handles.CORR_radiobutton,'Value')  ==  get(hObject,'Max')
        figure();
        subplot(3,1,1)
        plot(TEST,'r');hold all;plot(SIM,'k');hold off;
        grid on;title('Simple Raw Data Plot');
        legend('TEST DATA','SIM DATA');
        [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(3,1,2)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(3,1,3)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
        if mcorr>.89
        set(handles.RATING_edit,'String','Excellent match')
        elseif mcorr>.69
        set(handles.RATING_edit,'String','Good match')
        elseif mcorr>.49
        set(handles.RATING_edit,'String','Fair match')
        elseif mcorr<.50
        set(handles.RATING_edit,'String','Not Acceptable')
        end
    elseif get(handles.VSHIFT_radiobutton,'Value')  ==
get(hObject,'Max')
    figure();
    TEST=TEST-mean(TEST);
    SIM=SIM-mean(SIM);
    subplot(3,1,1)
    plot(TEST,'r');hold all;plot(SIM,'k');hold off;
    grid on;title('Vertical Shifted Data Plot');
    legend('TEST DATA','SIM DATA');
    [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(3,1,2)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(3,1,3)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
        if mcorr>.89
        set(handles.RATING_edit,'String','Excellent match')
        elseif mcorr>.69
        set(handles.RATING_edit,'String','Good match')
        elseif mcorr>.49
        set(handles.RATING_edit,'String','Fair match')
        elseif mcorr<.50
        set(handles.RATING_edit,'String','Not Acceptable')
        end

    elseif get(handles.FLIPTEST_radiobutton,'Value')  ==
get(hObject,'Max')
        figure();
        TEST=TEST-mean(TEST);
        SIM=SIM-mean(SIM);
        ratio=std(TEST)/std(SIM);
        TEST=TEST/ratio;
        TEST=TEST*-1;
        subplot(3,1,1)
        plot(TEST,'r');hold all;plot(SIM,'k');hold off;
        grid on;title('VSHIFT & Flipped TEST Data');
        legend('TEST DATA','SIM DATA');
        [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(3,1,2)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(3,1,3)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
            if mcorr>.89
            set(handles.RATING_edit,'String','Excellent match')
            elseif mcorr>.69
            set(handles.RATING_edit,'String','Good match')
            elseif mcorr>.49
            set(handles.RATING_edit,'String','Fair match')
            elseif mcorr<.50
            set(handles.RATING_edit,'String','Not Acceptable')
            end
     elseif get(handles.FLIPSIM_radiobutton,'Value')  ==
get(hObject,'Max')
        figure();
        TEST=TEST-mean(TEST);
        SIM=SIM-mean(SIM);
        ratio=std(TEST)/std(SIM);
        TEST=TEST/ratio;
        SIM=SIM*-1;
        subplot(3,1,1)
        plot(TEST,'r');hold all;plot(SIM,'k');hold off;
        grid on;title('VSHIFT & Flipped SIM Data');
        legend('TEST DATA','SIM DATA');
        [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(3,1,2)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(3,1,3)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
            if mcorr>.89
            set(handles.RATING_edit,'String','Excellent match')
            elseif mcorr>.69
            set(handles.RATING_edit,'String','Good match')
            elseif mcorr>.49
            set(handles.RATING_edit,'String','Fair match')
            elseif mcorr<.50
            set(handles.RATING_edit,'String','Not Acceptable')
            end

    end

elseif get(handles.WITHFILTER_radiobutton,'Value')  ==
get(hObject,'Max')
%Butterworth Infinite Impulse Response(IIR)Filter
r=Fs*0.5;
[b,a] = butter(5,handles.Data.FILTER_edit/r);
%5 stands for fifth order
%handles.Data.FILTER_edit/50 is cuttoff frequency,normalized to half
the
%sampling frequency(the Nyquist frequency)
Hd = dfilt.df2t(b,a); %direct form 2 transposed structure
TESTF = filter(Hd,TEST);
    if get(handles.CORR_radiobutton,'Value')  ==  get(hObject,'Max')
        figure();
        subplot(3,1,1)
        plot(TESTF,'r');hold all;plot(SIM,'k');hold off;
        grid on;title('Filtered TEST Data & SIM Data');
        legend('TEST DATA','SIM DATA');
        [m,lags]=xcorr(TESTF,SIM,'coeff');
        subplot(3,1,2)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(3,1,3)
        plot(TIME,TESTF,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
        if mcorr>.89
        set(handles.RATING_edit,'String','Excellent match')
        elseif mcorr>.69
        set(handles.RATING_edit,'String','Good match')
        elseif mcorr>.49
        set(handles.RATING_edit,'String','Fair match')
        elseif mcorr<.50
        set(handles.RATING_edit,'String','Not Acceptable')
        end
    elseif get(handles.VSHIFT_radiobutton,'Value')  ==
get(hObject,'Max')
        figure();
        TEST=TESTF-mean(TESTF);
        SIM=SIM-mean(SIM);
        subplot(3,1,1)
        plot(TEST,'r');hold all;plot(SIM,'k');hold off;
        grid on;title('Filtered & VShift TEST Data & SIM Data');
        legend('TEST DATA','SIM DATA');
        [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(3,1,2)
        plot(lags,m);
        grid on;title('Correlation between VSHIFT FILTERED TEST DATA &
SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(3,1,3)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
        if mcorr>.89
        set(handles.RATING_edit,'String','Excellent match')
        elseif mcorr>.69
        set(handles.RATING_edit,'String','Good match')
        elseif mcorr>.49
        set(handles.RATING_edit,'String','Fair match')
        elseif mcorr<.50
        set(handles.RATING_edit,'String','Not Acceptable')
        end

     elseif get(handles.FLIPTEST_radiobutton,'Value')  ==
get(hObject,'Max')
        figure();
        TEST=TESTF-mean(TESTF);
        SIM=SIM-mean(SIM);
        ratio=std(TEST)/std(SIM);
        TEST=TEST/ratio;
        TEST=TEST*-1;
        subplot(3,1,1)
        plot(TEST,'r');hold all;plot(SIM,'k');hold off;
        grid on;title('Filtered ,VShift & Flipped TEST Data & SIM
Data');
        legend('TEST DATA','SIM DATA');
        [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(3,1,2)
        plot(lags,m);
        grid on;title('Correlation between FILTERED VSHIFT FLIPPED TEST
DATA & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(3,1,3)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
            if mcorr>.89
            set(handles.RATING_edit,'String','Excellent match')
            elseif mcorr>.69
            set(handles.RATING_edit,'String','Good match')
            elseif mcorr>.49
            set(handles.RATING_edit,'String','Fair match')
            elseif mcorr<.50
            set(handles.RATING_edit,'String','Not Acceptable')
            end

    elseif get(handles.FLIPSIM_radiobutton,'Value')  ==
get(hObject,'Max')
        figure();
        TEST=TESTF-mean(TESTF);
        SIM=SIM-mean(SIM);
        ratio=std(TEST)/std(SIM);
        TEST=TEST/ratio;
        SIM=SIM*-1;
        subplot(3,1,1)
        plot(TEST,'r');hold all;plot(SIM,'k');hold off;
        grid on;title('Filtered & VShift TEST Data & Flipped SIM
Data');
        legend('TEST DATA','SIM DATA');
        [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(3,1,2)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(3,1,3)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
            if mcorr>.89
            set(handles.RATING_edit,'String','Excellent match')
            elseif mcorr>.69
            set(handles.RATING_edit,'String','Good match')
            elseif mcorr>.49
            set(handles.RATING_edit,'String','Fair match')
            elseif mcorr<.50
            set(handles.RATING_edit,'String','Not Acceptable')
            end

    end
elseif get(handles.WAVELET_radiobutton,'Value')  ==  get(hObject,'Max')
```

```
    %To perform a decomposition of the signal using the dmey wavelet),
    [C,L]=wavedec(TEST,handles.Data.WAVELET_edit,'dmey');
    [thr,sorh,keepapp]=ddencmp('den','wv',TEST);
    %De-noise Data by global threshold
    clean=wdencmp('gbl',C,L,'dmey',3,thr,sorh,keepapp);

    if get(handles.CORR_radiobutton,'Value')  ==  get(hObject,'Max')
        figure();
        subplot(3,1,1)
        plot(clean,'r');hold all;plot(SIM,'k');hold off;
        grid on;title('Denoised TEST Data & SIM Data');
        legend('TEST DATA','SIM DATA');
        [m,lags]=xcorr(clean,SIM,'coeff');
        subplot(3,1,2)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(3,1,3)
        plot(TIME,clean,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
        if mcorr>.89
        set(handles.RATING_edit,'String','Excellent match')
        elseif mcorr>.69
        set(handles.RATING_edit,'String','Good match')
        elseif mcorr>.49
        set(handles.RATING_edit,'String','Fair match')
        elseif mcorr<.50
        set(handles.RATING_edit,'String','Not Acceptable')
        end
    elseif get(handles.VSHIFT_radiobutton,'Value')  ==
get(hObject,'Max')
    figure();
    subplot(3,1,1)
    TEST=clean-mean(clean);
    SIM=SIM-mean(SIM);
    plot(TEST,'r');hold all;plot(SIM,'k');hold off;
    grid on;title('Denoised & VShift TEST Data & SIM Data');
    legend('TEST DATA','SIM DATA');
       [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(3,1,2)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(3,1,3)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
        if mcorr>.89
        set(handles.RATING_edit,'String','Excellent match')
        elseif mcorr>.69
        set(handles.RATING_edit,'String','Good match')
        elseif mcorr>.49
        set(handles.RATING_edit,'String','Fair match')
        elseif mcorr<.50
        set(handles.RATING_edit,'String','Not Acceptable')
        end
     elseif get(handles.FLIPTEST_radiobutton,'Value')  ==
get(hObject,'Max')
        figure();
        TEST=clean-mean(clean);
        SIM=SIM-mean(SIM);
        ratio=std(TEST)/std(SIM);
        TEST=TEST/ratio;
        TEST=TEST*-1;
        subplot(3,1,1)
        plot(TEST,'r');hold all;plot(SIM,'k');hold off;
        grid on;title('Filtered,VShift & Flipped TEST Data & SIM
Data');
        legend('TEST DATA','SIM DATA');
        [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(3,1,2)
        plot(lags,m);
        grid on;title('Correlation between DENOISED VSHIFT FLIPPED TEST
DATA & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(3,1,3)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
            if mcorr>.89
            set(handles.RATING_edit,'String','Excellent match')
            elseif mcorr>.69
            set(handles.RATING_edit,'String','Good match')
            elseif mcorr>.49
            set(handles.RATING_edit,'String','Fair match')
            elseif mcorr<.50
            set(handles.RATING_edit,'String','Not Acceptable')
            end
     elseif get(handles.FLIPSIM_radiobutton,'Value')  ==
get(hObject,'Max')
        figure();
        TEST=clean-mean(clean);
        SIM=SIM-mean(SIM);
        ratio=std(TEST)/std(SIM);
        TEST=TEST/ratio;
        SIM=SIM*-1;
        subplot(3,1,1)
        plot(TEST,'r');hold all;plot(SIM,'k');hold off;
        grid on;title('Filtered & VShift TEST Data & Flipped SIM
Data');
        legend('TEST DATA','SIM DATA');
        [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(3,1,2)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(3,1,3)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
            if mcorr>.89
            set(handles.RATING_edit,'String','Excellent match')
            elseif mcorr>.69
            set(handles.RATING_edit,'String','Good match')
            elseif mcorr>.49
            set(handles.RATING_edit,'String','Fair match')
            elseif mcorr<.50
            set(handles.RATING_edit,'String','Not Acceptable')
            end

    end

  end

% --- Executes on button press in FFT_button.
function FFT_button_Callback(hObject, eventdata, handles)
% hObject    handle to FFT_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[x] = get_var_namest(handles);
[y] = get_var_namess(handles);
[z] = get_var_namesti(handles);
TEST=evalin('base',x);
SIM=evalin('base',y);
TIME=evalin('base',z);
Fs=length(TEST)/max(TIME);
N=length(TEST);
f=Fs*(1:N)/N;
SIM=fft(SIM);
c=handles.Data.XAXIS_edit;
if get(handles.WITHOUTFILTER_radiobutton,'Value')  ==
get(hObject,'Max')
    figure();
    TEST=fft(TEST);
    plot(f,abs(TEST),'r');xlim([1 c]);hold all;plot(f,abs(SIM));hold
off;xlabel('Frequency');ylabel('Amplitude');legend('TEST','SIM');
    title('FFT TEST DATA & SIM DATA');
elseif get(handles.WITHFILTER_radiobutton,'Value')  ==
get(hObject,'Max')
```

```
    r=N*0.5;%the nyquist frequency
    [b,a] = butter(5,handles.Data.FILTER_edit/r);
    Hd = dfilt.df2t(b,a);
    TESTF = filter(Hd,TEST);
    TESTF=fft(TESTF);
    figure();
    plot(f,abs(TESTF));xlim([1 c]);hold all;plot(f,abs(SIM));hold
off;xlabel('Frequency');ylabel('Amplitude');legend('TEST','SIM');
    title('FFT FILTERED TEST DATA & SIM DATA');
elseif get(handles.WAVELET_radiobutton,'Value')  ==  get(hObject,'Max')
     figure();
    %To perform a decomposition of the signal using the dmey wavelet),
    [C,L]=wavedec(TEST,handles.Data.WAVELET_edit,'dmey');
    [thr,sorh,keepapp]=ddencmp('den','wv',TEST);
    set(handles.THRESHOLD_edit,'String',thr);
    set(handles.THRTYPE_edit,'String',sorh);
    %De-noise Data by global threshold
    clean=wdencmp('gbl',C,L,'dmey',3,thr,sorh,keepapp);
    TESTW=fft(clean);
    plot(f,abs(TESTW));xlim([1 c]);hold all;plot(f,abs(SIM));hold
off;xlabel('Frequency');ylabel('Amplitude');legend('TEST','SIM');
    title('FFT DENOISED TEST DATA & SIM DATA');
end


function WAVELET_edit_Callback(hObject, eventdata, handles)
% hObject    handle to WAVELET_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.Data.WAVELET_edit=str2double(get(hObject,'String'));
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of WAVELET_edit as text
%        str2double(get(hObject,'String')) returns contents of
WAVELET_edit as a double



% --- Executes during object creation, after setting all properties.
function WAVELET_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to WAVELET_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in WAVELET_button.
function WAVELET_button_Callback(hObject, eventdata, handles)
% hObject    handle to WAVELET_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

figure();
[x] = get_var_namest(handles);
[y] = get_var_namess(handles);
TEST=evalin('base',x);
SIM=evalin('base',y);
subplot(3,1,1)
plot(TEST,'DisplayName','TEST');
title('Simple TEST Data');
%Low-pass filter by wavelet decomposition which gives approximation
%coefficients contain less noise than does the original signal
[C,L]=wavedec(TEST,handles.Data.WAVELET_edit,'dmey');%dmey stand for
Meyer Wavelet
T=wrcoef('a',C,L,'dmey',handles.Data.WAVELET_edit);
subplot(3,1,2)
plot(T);title('Wavelet Transformed TEST Data ');
subplot(3,1,3)
plot(T,'DisplayName','TEST');hold all;plot(SIM,'DisplayName','SIM');
legend('TEST DATA','SIM DATA');hold off;
title('SIM Data and Wavelet Transformed TEST Data ');
[m,lags]=xcorr(T,SIM,'coeff');
[mcorr,lag]=max(m);
set(handles.WAVELETCORR_edit,'String',mcorr)


function WAVELETCORR_edit_Callback(hObject, eventdata, handles)
% hObject    handle to WAVELETCORR_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of WAVELETCORR_edit as
text
%        str2double(get(hObject,'String')) returns contents of
WAVELETCORR_edit as a double


% --- Executes during object creation, after setting all properties.
function WAVELETCORR_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to WAVELETCORR_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function FFTFILTER_edit_Callback(hObject, eventdata, handles)
% hObject    handle to FFTFILTER_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.Data.FFTFILTER_edit=str2double(get(hObject,'String'));
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of FFTFILTER_edit as
text
%        str2double(get(hObject,'String')) returns contents of
FFTFILTER_edit as a double


% --- Executes during object creation, after setting all properties.
function FFTFILTER_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FFTFILTER_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in SKEW_button.
function SKEW_button_Callback(hObject, eventdata, handles)
% hObject    handle to SKEW_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[x] = get_var_namest(handles);
[y] = get_var_namess(handles);
TEST=evalin('base',x);
SIM=evalin('base',y);
%Skewness is a measure of the asymmetry of the data around the sample
mean.
%If skewness is negative, the data are spread out more to the left of
the
%mean than to the right. If skewness is positive, the data are spread
out
%more to the right. The skewness of the normal distribution
%(or any perfectly symmetric distribution) is zero.
TEST3=skewness(TEST);
SIM3=skewness(SIM);
set(handles.SKEWTEST_edit,'String',TEST3);
set(handles.SKEWSIM_edit,'String',SIM3);



% --- Executes on button press in PLOT_button.
function PLOT_button_Callback(hObject, eventdata, handles)
% hObject    handle to PLOT_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[x] = get_var_namest(handles);
[y] = get_var_namess(handles);
[z] = get_var_namesti(handles);
TEST=evalin('base',x);
SIM=evalin('base',y);
TIME=evalin('base',z);
figure();
subplot(2,1,1)
plot(TEST,'r');hold all;plot(SIM,'k');hold off;
grid on;
title('Simple Raw Data Plot');
legend('TEST DATA','SIM DATA');
subplot(2,1,2)
plot(TIME,TEST,'r',TIME,SIM,'k');
grid on;xlabel('Time(sec)');
title('Simple Raw Data Plot');
legend('TEST DATA','SIM DATA');


% --- Executes on button press in REPORT_button.
function REPORT_button_Callback(hObject, eventdata, handles)
% hObject    handle to REPORT_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

report



function edit19_Callback(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit19 as text
%        str2double(get(hObject,'String')) returns contents of edit19
as a double


% --- Executes during object creation, after setting all properties.
function edit19_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function THRESHOLD_edit_Callback(hObject, eventdata, handles)
% hObject    handle to THRESHOLD_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of THRESHOLD_edit as
text
%        str2double(get(hObject,'String')) returns contents of
THRESHOLD_edit as a double


% --- Executes during object creation, after setting all properties.
function THRESHOLD_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to THRESHOLD_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function THRTYPE_edit_Callback(hObject, eventdata, handles)
% hObject    handle to THRTYPE_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of THRTYPE_edit as text
%        str2double(get(hObject,'String')) returns contents of
THRTYPE_edit as a double


% --- Executes during object creation, after setting all properties.
function THRTYPE_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to THRTYPE_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function DataRange_edit_Callback(hObject, eventdata, handles)
% hObject    handle to DataRange_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.Data.DataRange_edit=str2double(get(hObject,'String'));
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of DataRange_edit as
text
%        str2double(get(hObject,'String')) returns contents of
DataRange_edit as a double


% --- Executes during object creation, after setting all properties.
function DataRange_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DataRange_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function DataRange1_edit_Callback(hObject, eventdata, handles)
% hObject    handle to DataRange1_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.Data.DataRange1_edit=str2double(get(hObject,'String'));
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of DataRange1_edit as
text
%        str2double(get(hObject,'String')) returns contents of
DataRange1_edit as a double


% --- Executes during object creation, after setting all properties.
function DataRange1_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DataRange1_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

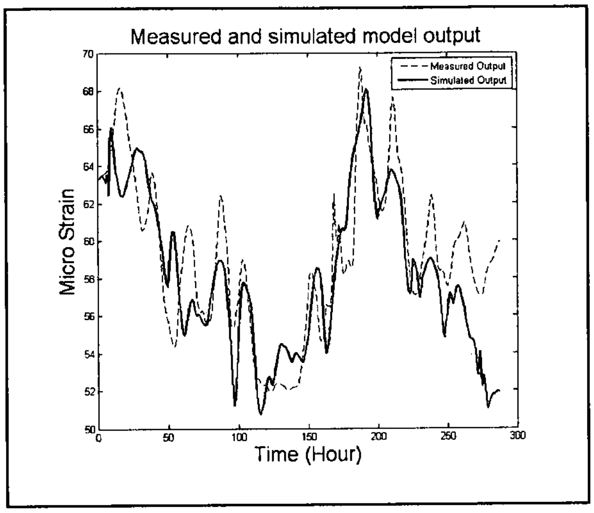
```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function XAXIS_edit_Callback(hObject, eventdata, handles)
% hObject    handle to XAXIS_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.Data.XAXIS_edit=str2double(get(hObject,'String'));
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of XAXIS_edit as text
%        str2double(get(hObject,'String')) returns contents of
XAXIS_edit as a double


% --- Executes during object creation, after setting all properties.
function XAXIS_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to XAXIS_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)




% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in SELECTEDRANGE_CORR_button.
function SELECTEDRANGE_CORR_button_Callback(hObject, eventdata,
handles)
% hObject    handle to SELECTEDRANGE_CORR_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[x] = get_var_namest(handles);
[y] = get_var_namess(handles);
[z] = get_var_namesti(handles);
TEST=evalin('base',x);
SIM=evalin('base',y);
TIME=evalin('base',z);
Fs=length(TEST)/max(TIME);
TEST=TEST(handles.Data.DataRange_edit:handles.Data.DataRange1_edit);
SIM=SIM(handles.Data.DataRange_edit:handles.Data.DataRange1_edit);
TIME=TIME(handles.Data.DataRange_edit:handles.Data.DataRange1_edit);
p=length(TEST);
if get(handles.WITHOUTFILTER_radiobutton,'Value')  ==
get(hObject,'Max')
    if get(handles.CORR_radiobutton,'Value')  ==  get(hObject,'Max')
        figure();
        subplot(3,1,1)
        plot(TEST,'r');hold all;plot(SIM,'k');hold off;
        grid on;
        title('Simple Raw Data Plot');
        legend('TEST DATA','SIM DATA');
        [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(3,1,2)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(3,1,3)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
        if mcorr>.89
        set(handles.RATING_edit,'String','Excellent match')
        elseif mcorr>.69
        set(handles.RATING_edit,'String','Good match')
        elseif mcorr>.49
        set(handles.RATING_edit,'String','Fair match')
        elseif mcorr<.50
        set(handles.RATING_edit,'String','Not Acceptable')
        end
    elseif get(handles.VSHIFT_radiobutton,'Value')  ==
get(hObject,'Max')
    figure();
    TEST=TEST-mean(TEST);
    SIM=SIM-mean(SIM);
    subplot(3,1,1)
    plot(TEST,'r');hold all;plot(SIM,'k');hold off;
    grid on;
    title('Vertical Shifted Data Plot');
    legend('TEST DATA','SIM DATA');
    [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(3,1,2)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(3,1,3)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
        if mcorr>.89
        set(handles.RATING_edit,'String','Excellent match')
        elseif mcorr>.69
        set(handles.RATING_edit,'String','Good match')
        elseif mcorr>.49
        set(handles.RATING_edit,'String','Fair match')
        elseif mcorr<.50
        set(handles.RATING_edit,'String','Not Acceptable')
        end

    elseif get(handles.FLIPTEST_radiobutton,'Value')  ==
get(hObject,'Max')
        figure();
        TEST=TEST-mean(TEST);
        SIM=SIM-mean(SIM);
        ratio=std(TEST)/std(SIM);
        TEST=TEST/ratio;
        TEST=TEST*-1;
        subplot(3,1,1)
        plot(TEST,'r');hold all;plot(SIM,'k');hold off;
        grid on;
        title('Flipped TEST Data');
        legend('TEST DATA','SIM DATA');
        [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(3,1,2)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(3,1,3)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
```

```
            if mcorr>.89
            set(handles.RATING_edit,'String','Excellent match')
            elseif mcorr>.69
            set(handles.RATING_edit,'String','Good match')
            elseif mcorr>.49
            set(handles.RATING_edit,'String','Fair match')
            elseif mcorr<.50
            set(handles.RATING_edit,'String','Not Acceptable')
            end
        elseif get(handles.FLIPSIM_radiobutton,'Value')  ==
get(hObject,'Max')
          figure();
          TEST=TEST-mean(TEST);
          SIM=SIM-mean(SIM);
          ratio=std(TEST)/std(SIM);
          TEST=TEST/ratio;
          SIM=SIM*-1;
          subplot(3,1,1)
          plot(TEST,'r');hold all;plot(SIM,'k');hold off;
          grid on;
          title('Flipped SIM Data');
          legend('TEST DATA','SIM DATA');
          [m,lags]=xcorr(TEST,SIM,'coeff');
          subplot(3,1,2)
          plot(lags,m);
          grid on;title('Correlation between TEST & SIM Data');
          [mcorr,lag]=max(m);
          lag=(lag-p)/Fs;
          subplot(3,1,3)
          plot(TIME,TEST,'r',TIME+lag,SIM,'k');
          grid on;xlabel('Time(Sec)');
          title('Shifting SIM data in Lag amount with respect to TEST
data');
          legend('TEST DATA','SIM DATA');
          set(handles.MAXCORR_edit,'String',mcorr)
          set(handles.LAG_edit,'String',lag)
             if mcorr>.89
             set(handles.RATING_edit,'String','Excellent match')
             elseif mcorr>.69
             set(handles.RATING_edit,'String','Good match')
             elseif mcorr>.49
             set(handles.RATING_edit,'String','Fair match')
             elseif mcorr<.50
             set(handles.RATING_edit,'String','Not Acceptable')
             end

     end

elseif get(handles.WITHFILTER_radiobutton,'Value')  ==
get(hObject,'Max')
%Butterworth Infinite Impulse Response(IIR)Filter
r=Fs*0.5;
[b,a] = butter(5,handles.Data.FILTER_edit/r);
%5 stands for fifth order
%handles.Data.FILTER_edit/r is cuttoff frequency,normalized to half the
%sampling frequency(the Nyquist frequency)
```

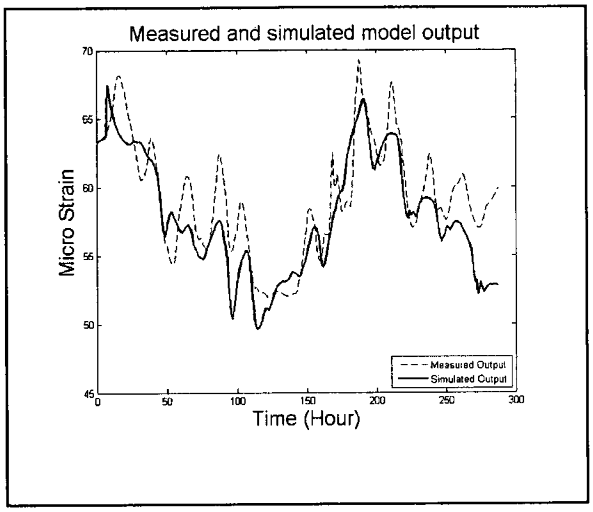
```
Hd = dfilt.df2t(b,a); %direct form 2 transposed structure
TESTF = filter(Hd,TEST);
    if get(handles.CORR_radiobutton,'Value')  ==  get(hObject,'Max')
        figure();
        subplot(4,1,1)
        plot(TEST,'r')
        grid on;title('Simple TEST Data');
        subplot(4,1,2)
        plot(TESTF,'r')
        grid on;title('Filtered TEST Data');
        [m,lags]=xcorr(TESTF,SIM,'coeff');
        subplot(4,1,3)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(4,1,4)
        plot(TIME,TESTF,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
        if mcorr>.89
        set(handles.RATING_edit,'String','Excellent match')
        elseif mcorr>.69
        set(handles.RATING_edit,'String','Good match')
        elseif mcorr>.49
        set(handles.RATING_edit,'String','Fair match')
        elseif mcorr<.50
        set(handles.RATING_edit,'String','Not Acceptable')
        end
    elseif get(handles.VSHIFT_radiobutton,'Value')  ==
get(hObject,'Max')
        figure();
        subplot(4,1,1)
        plot(TEST,'r')
        grid on;title('Simple TEST Data');
        subplot(4,1,2)
        plot(TESTF,'r')
        grid on;title('Filtered TEST Data');
        TEST=TESTF-mean(TESTF);
        SIM=SIM-mean(SIM);
        [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(4,1,3)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(4,1,4)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
        if mcorr>.89
        set(handles.RATING_edit,'String','Excellent match')
        elseif mcorr>.69
        set(handles.RATING_edit,'String','Good match')
        elseif mcorr>.49
        set(handles.RATING_edit,'String','Fair match')
        elseif mcorr<.50
        set(handles.RATING_edit,'String','Not Acceptable')
        end

     elseif get(handles.FLIPTEST_radiobutton,'Value')  ==
get(hObject,'Max')
        figure();
        subplot(4,1,1)
        plot(TEST,'r')
        grid on;title('Simple TEST Data');
        subplot(4,1,2)
        plot(TESTF,'r')
        grid on;title('Filtered TEST Data');
        TEST=TESTF-mean(TESTF);
        SIM=SIM-mean(SIM);
        ratio=std(TEST)/std(SIM);
        TEST=TEST/ratio;
        TEST=TEST*-1;
        [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(4,1,3)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(4,1,4)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
            if mcorr>.89
            set(handles.RATING_edit,'String','Excellent match')
            elseif mcorr>.69
            set(handles.RATING_edit,'String','Good match')
            elseif mcorr>.49
            set(handles.RATING_edit,'String','Fair match')
            elseif mcorr<.50
            set(handles.RATING_edit,'String','Not Acceptable')
            end

    elseif get(handles.FLIPSIM_radiobutton,'Value')  ==
get(hObject,'Max')
        figure();
        subplot(4,1,1)
        plot(TEST,'r')
        grid on;title('Simple TEST Data');
        subplot(4,1,2)
        plot(TESTF,'r')
        grid on;title('Filtered TEST Data');
        TEST=TESTF-mean(TESTF);
        SIM=SIM-mean(SIM);
        ratio=std(TEST)/std(SIM);
        TEST=TEST/ratio;
        SIM=SIM*-1;
        [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(4,1,3)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(4,1,4)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
            if mcorr>.89
            set(handles.RATING_edit,'String','Excellent match')
            elseif mcorr>.69
            set(handles.RATING_edit,'String','Good match')
            elseif mcorr>.49
            set(handles.RATING_edit,'String','Fair match')
            elseif mcorr<.50
            set(handles.RATING_edit,'String','Not Acceptable')
            end

    end
elseif get(handles.WAVELET_radiobutton,'Value')  ==  get(hObject,'Max')
    %To perform a decomposition of the signal using the dmey wavelet),
    [C,L]=wavedec(TEST,handles.Data.WAVELET_edit,'dmey');
    [thr,sorh,keepapp]=ddencmp('den','wv',TEST);
    %De-noise Data by global threshold
    clean=wdencmp('gbl',C,L,'dmey',3,thr,sorh,keepapp);

    if get(handles.CORR_radiobutton,'Value')  ==  get(hObject,'Max')
        figure();
        subplot(4,1,1)
        plot(TEST)
        grid on;
        title('TEST DATA');
        subplot(4,1,2)
        plot(clean)
        grid on;
        title('DENOISED TEST DATA');
        [m,lags]=xcorr(clean,SIM,'coeff');
        subplot(4,1,3)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(4,1,4)
        plot(TIME,clean,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
        if mcorr>.89
        set(handles.RATING_edit,'String','Excellent match')
        elseif mcorr>.69
        set(handles.RATING_edit,'String','Good match')
        elseif mcorr>.49
        set(handles.RATING_edit,'String','Fair match')
        elseif mcorr<.50
        set(handles.RATING_edit,'String','Not Acceptable')
        end
    elseif get(handles.VSHIFT_radiobutton,'Value')  ==
get(hObject,'Max')
    figure();
    subplot(4,1,1)
    plot(TEST)
    grid on;
    title('TEST DATA');
    subplot(4,1,2)
    plot(clean)
    grid on;
    title('DENOISED TEST DATA');
    TEST=clean-mean(clean);
    SIM=SIM-mean(SIM);
       [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(4,1,3)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(4,1,4)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
        if mcorr>.89
        set(handles.RATING_edit,'String','Excellent match')
        elseif mcorr>.69
        set(handles.RATING_edit,'String','Good match')
        elseif mcorr>.49
        set(handles.RATING_edit,'String','Fair match')
        elseif mcorr<.50
        set(handles.RATING_edit,'String','Not Acceptable')
        end
     elseif get(handles.FLIPTEST_radiobutton,'Value')  ==
get(hObject,'Max')
        figure();
        subplot(4,1,1)
        plot(TEST)
        grid on;
        title('TEST DATA');
        subplot(4,1,2)
        plot(clean)
        grid on;
        title('DENOISED TEST DATA');
        TEST=clean-mean(clean);
        SIM=SIM-mean(SIM);
        ratio=std(TEST)/std(SIM);
        TEST=TEST/ratio;
        TEST=TEST*-1;
        [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(4,1,3)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(4,1,4)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
            if mcorr>.89
            set(handles.RATING_edit,'String','Excellent match')
            elseif mcorr>.69
            set(handles.RATING_edit,'String','Good match')
            elseif mcorr>.49
            set(handles.RATING_edit,'String','Fair match')
            elseif mcorr<.50
            set(handles.RATING_edit,'String','Not Acceptable')
            end
     elseif get(handles.FLIPSIM_radiobutton,'Value')  ==
get(hObject,'Max')
        figure();
        subplot(4,1,1)
        plot(TEST)
        grid on;
        title('TEST DATA');
        subplot(4,1,2)
        plot(clean)
        grid on;
        title('DENOISED TEST DATA');
        TEST=clean-mean(clean);
        SIM=SIM-mean(SIM);
        ratio=std(TEST)/std(SIM);
        TEST=TEST/ratio;
        SIM=SIM*-1;
        [m,lags]=xcorr(TEST,SIM,'coeff');
        subplot(4,1,3)
        plot(lags,m);
        grid on;title('Correlation between TEST & SIM Data');
        [mcorr,lag]=max(m);
        lag=(lag-p)/Fs;
        subplot(4,1,4)
        plot(TIME,TEST,'r',TIME+lag,SIM,'k');
        grid on;xlabel('Time(Sec)');
        title('Shifting SIM data in Lag amount with respect to TEST
data');
        legend('TEST DATA','SIM DATA');
        set(handles.MAXCORR_edit,'String',mcorr)
        set(handles.LAG_edit,'String',lag)
            if mcorr>.89
            set(handles.RATING_edit,'String','Excellent match')
            elseif mcorr>.69
            set(handles.RATING_edit,'String','Good match')
            elseif mcorr>.49
            set(handles.RATING_edit,'String','Fair match')
            elseif mcorr<.50
            set(handles.RATING_edit,'String','Not Acceptable')
            end

    end


 end


% --- Executes on button press in FFTbutton_.
function FFTbutton__Callback(hObject, eventdata, handles)
% hObject    handle to FFTbutton_ (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[x] = get_var_namest(handles);
[y] = get_var_namess(handles);
[z] = get_var_namesti(handles);
TEST=evalin('base',x);
SIM=evalin('base',y);
TIME=evalin('base',z);
Fs=length(TEST)/max(TIME);
N=length(TEST);
f=Fs*(1:N)/N;
SIM=fft(SIM);
if get(handles.WITHOUTFILTER_radiobutton,'Value')  ==
get(hObject,'Max')
    figure();
    TEST=fft(TEST);
    plot(f,abs(TEST),'r');hold all;plot(f,abs(SIM));hold
off;xlabel('Frequency');ylabel('Amplitude');legend('TEST','SIM');
elseif get(handles.WITHFILTER_radiobutton,'Value')  ==
get(hObject,'Max')
    r=N*0.5;%the nyquist frequency
    [b,a] = butter(5,handles.Data.FILTER_edit/r);
    Hd = dfilt.df2t(b,a);
    TESTF = filter(Hd,TEST);
    TEST=fft(TESTF);
    figure();
    plot(f,abs(TEST));hold all;plot(f,abs(SIM));hold
off;xlabel('Frequency');ylabel('Amplitude');legend('TEST','SIM');
elseif get(handles.WAVELET_radiobutton,'Value')  ==  get(hObject,'Max')
     figure();
    %To perform a decomposition of the signal using the dmey wavelet),
    [C,L]=wavedec(TEST,handles.Data.WAVELET_edit,'dmey');
    [thr,sorh,keepapp]=ddencmp('den','wv',TEST);
    set(handles.THRESHOLD_edit,'String',thr);
    set(handles.THRTYPE_edit,'String',sorh);
    %De-noise Data by global threshold
    clean=wdencmp('gbl',C,L,'dmey',3,thr,sorh,keepapp);
    TEST=fft(clean);
    plot(f,abs(TEST));hold all;plot(f,abs(SIM));hold
off;xlabel('Frequency');ylabel('Amplitude');legend('TEST','SIM');

end
```

Appendix B

B.1 Graphical output of defective sensors detection of Portage Creek Bridge. The details are described in Chapter 5, section 5.2.

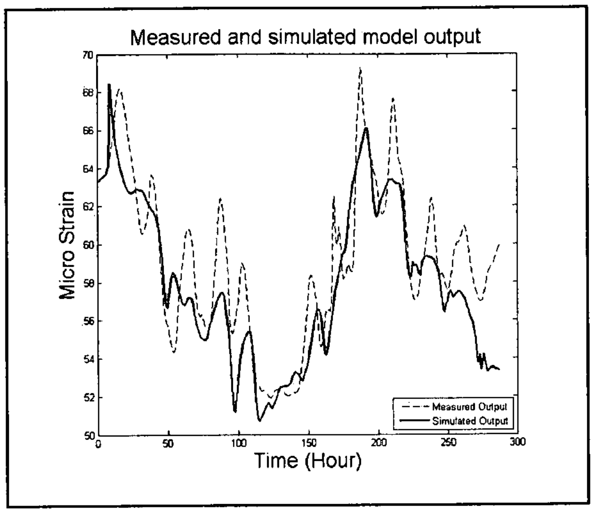


(a)

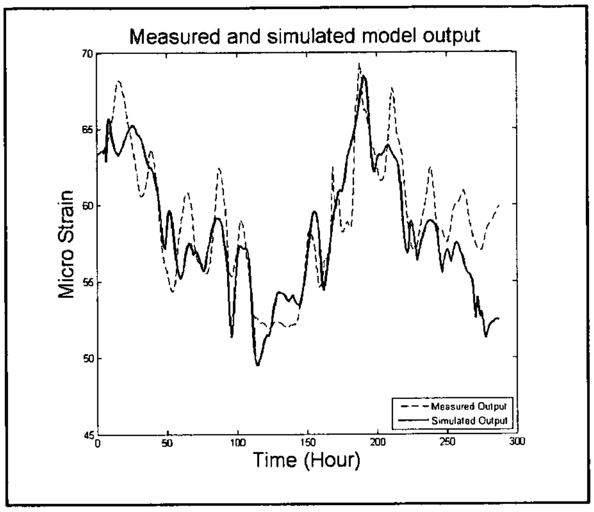


(b)

Figure B.1 Measured and Simulated output of March 2006, (a) sensor# 2-1 removed, (b) sensor# 2-2 removed.

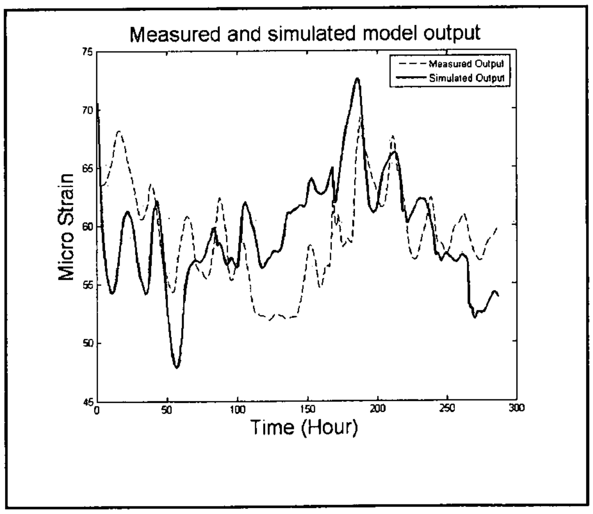


(a)

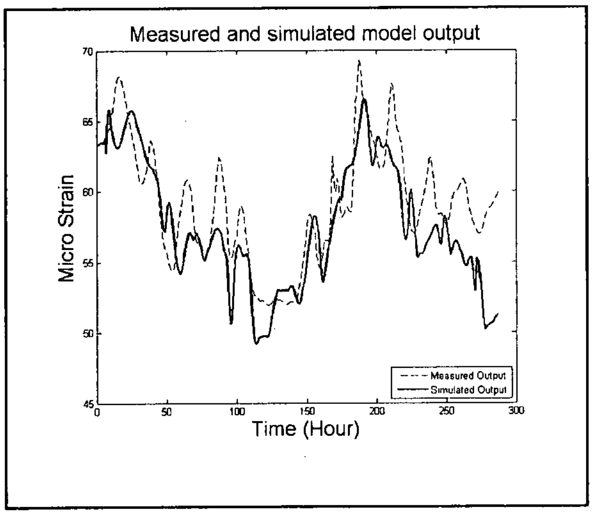


(b)

FigureB.2 Measured and Simulated output of March 2006, (a) sensor# 3-1 removed, (b) sensor# 3-2 removed.

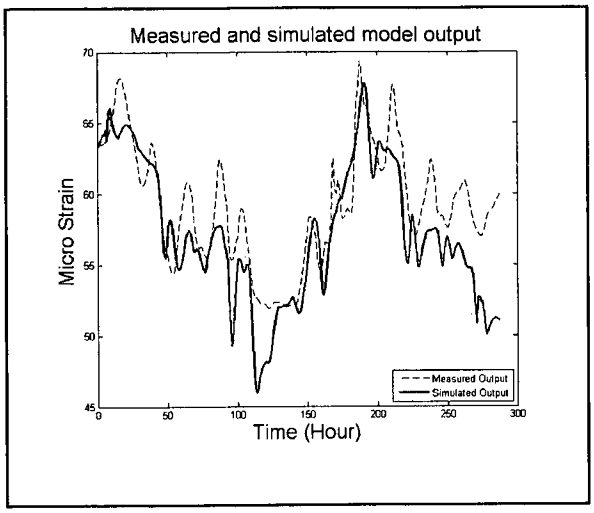


(a)

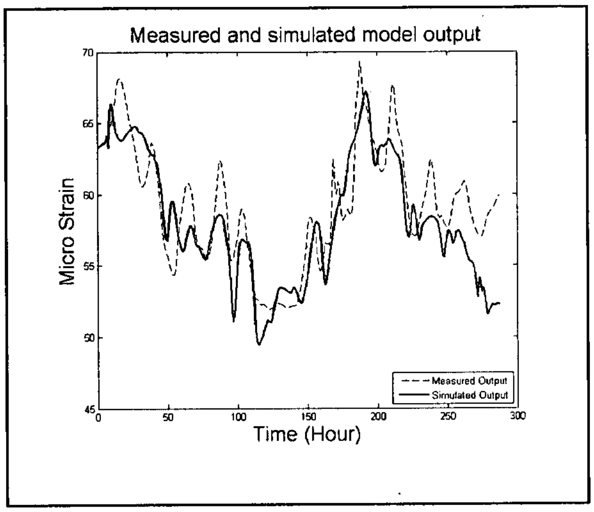


(b)

FigureB.3 Measured and Simulated output of March 2006, (a) sensor# 4-1 removed, (b) sensor# 4-2 removed.

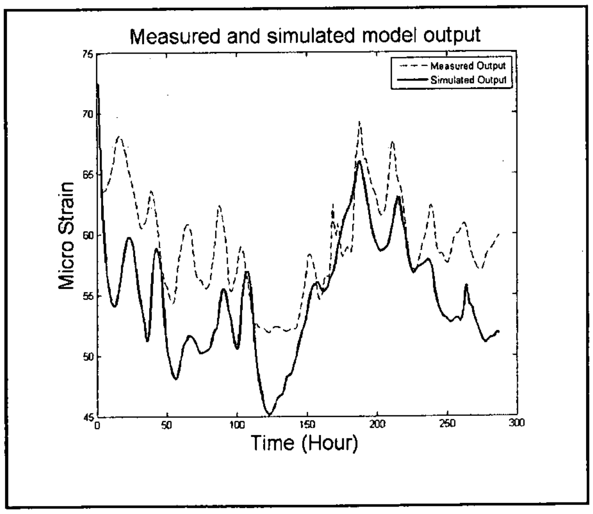


(a)

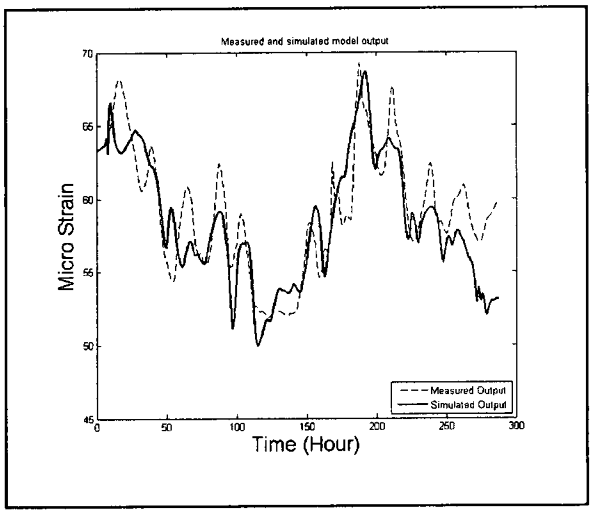


(b)

Figure B.4 Measured and Simulated output of March 2006, (a) sensor# 5-1 removed, (b) sensor# 5-2 removed.

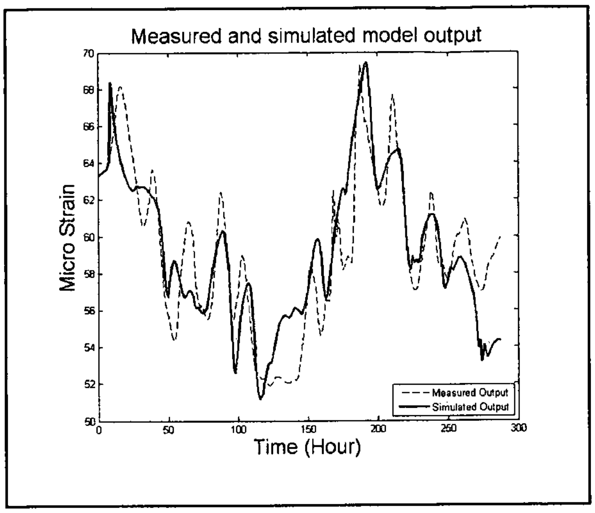


(a)

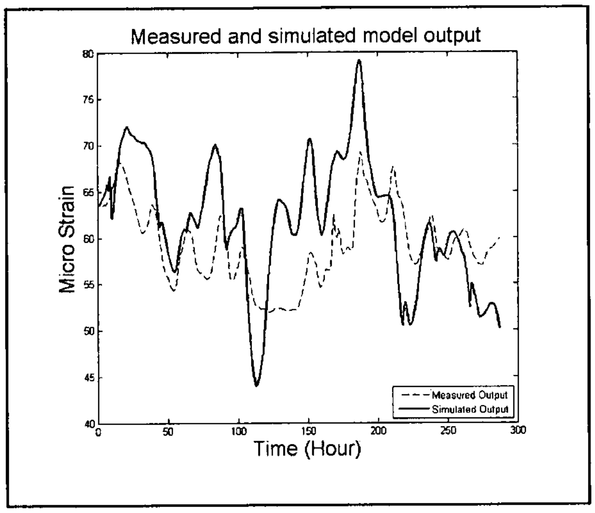


(b)

Figure B.5 Measured and Simulated output of March 2006, (a) sensor# 6-2 removed, (b) sensor# 7-1 removed.

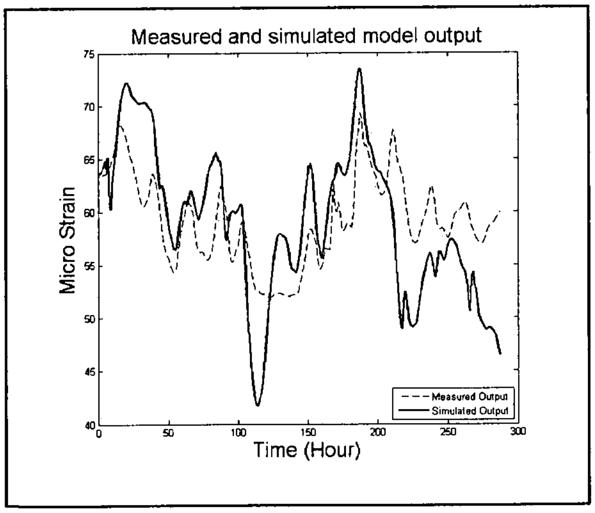


(a)

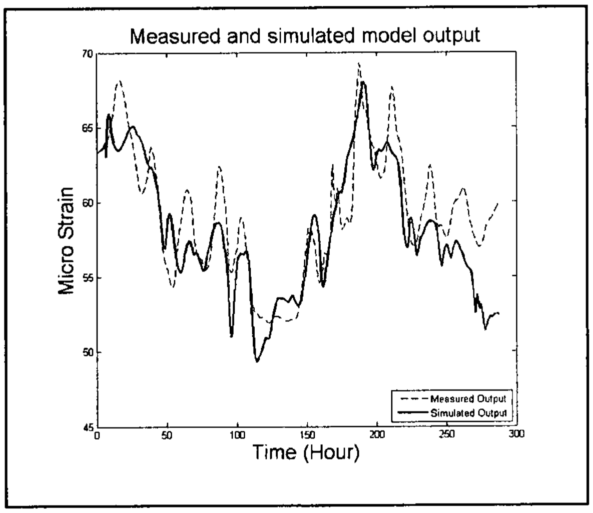


(b)

Figure B.6 Measured and Simulated output of March 2006, (a) sensor# 7-2 removed, (b) sensor# 8-1 removed.



(a)

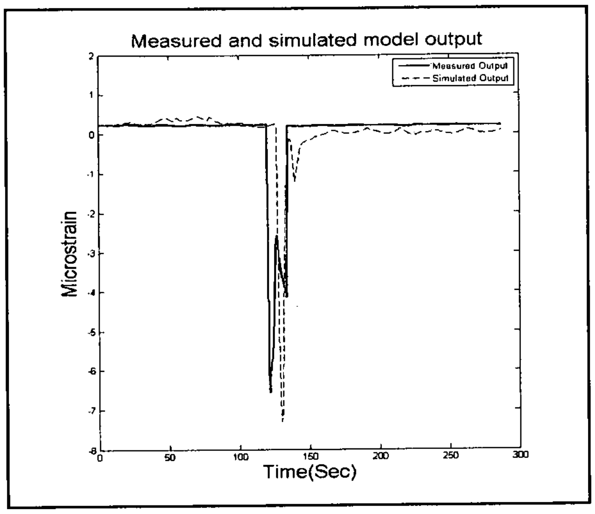


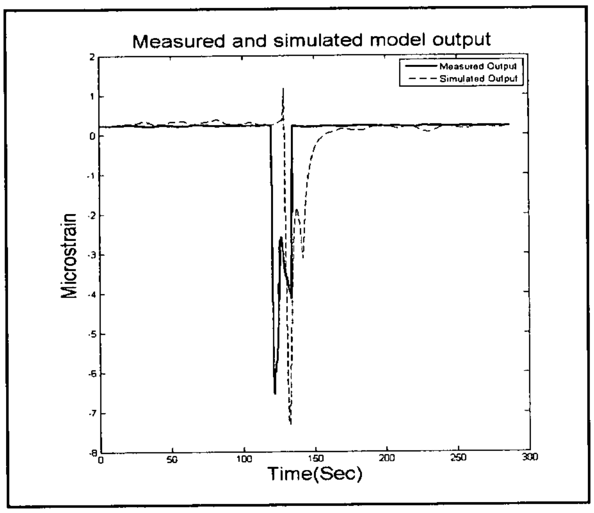
(b)

Figure B.7 Measured and Simulated output of March 2006, (a) sensor#8-2 removed, (b) sensor# 9-1 removed.

Appendix C

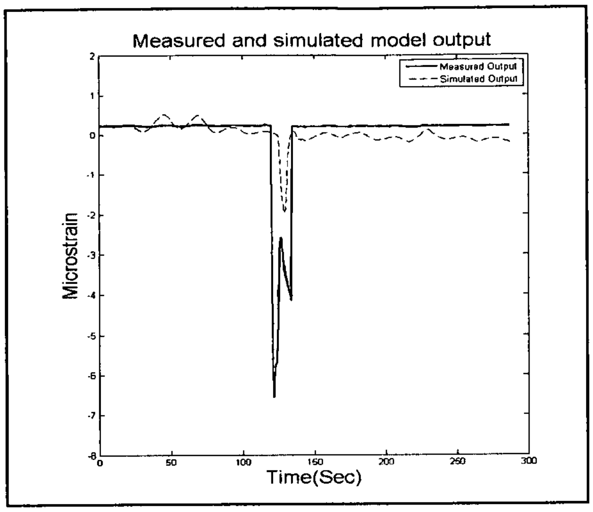
C.1 Graphical output of defective sensors detection of Portage Creek Bridge. The details are described in Chapter 5, section 5.3.



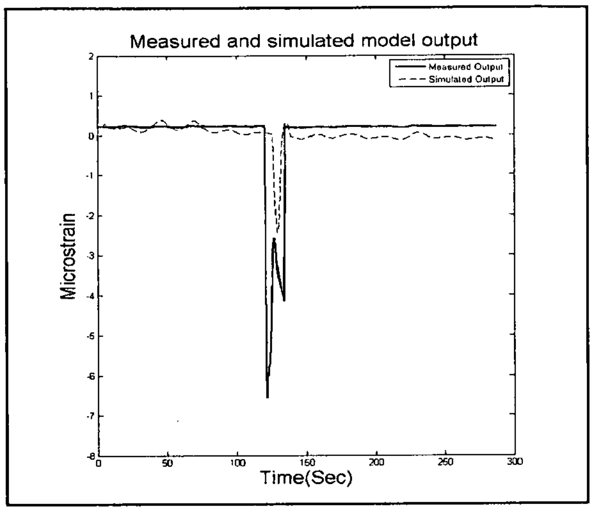


(a) (b)

Figure C.1 Measured and Simulated output of March 2006, (a) sensor# 2-1 removed, (b) sensor# 2-2 removed.

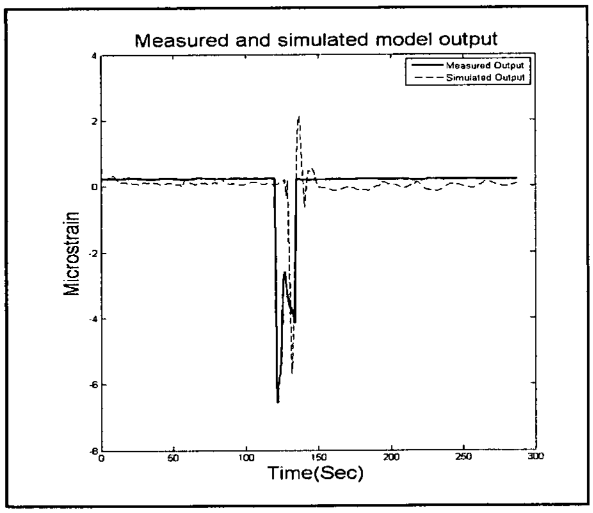


(a)

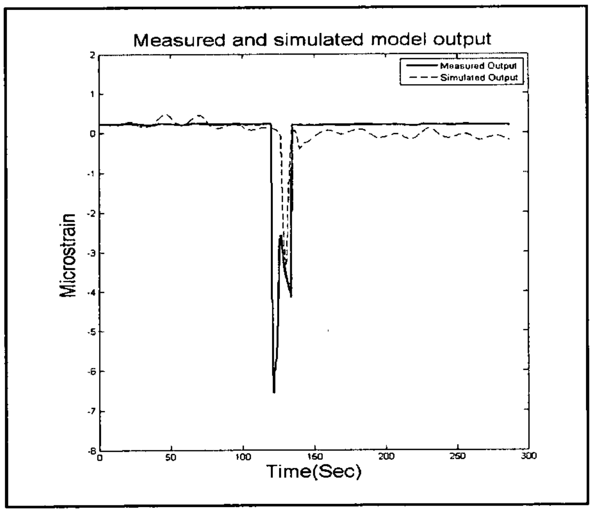


(b)

Figure C.2 Measured and Simulated output of March 2006, (a) sensor# 3-1 removed, (b) sensor# 3-2 removed.

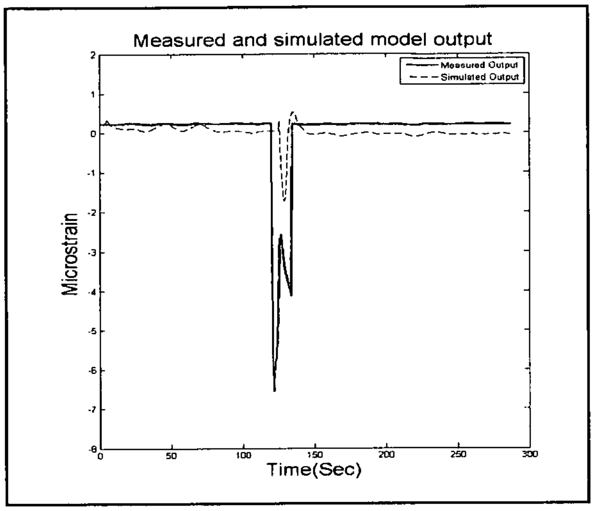


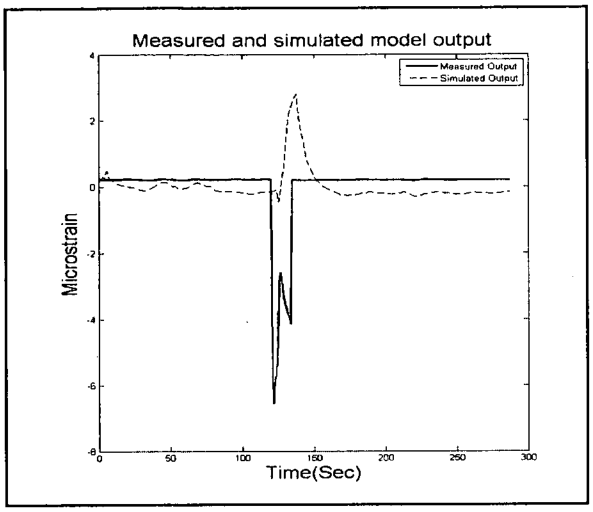
(a)



(b)

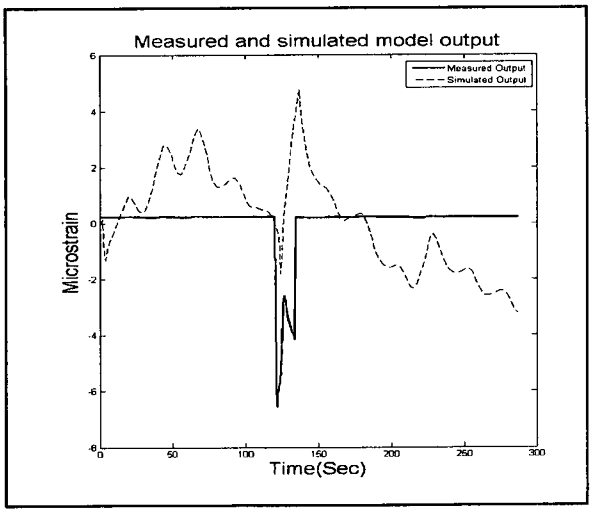
Figure C.3 Measured and Simulated output of March 2006, (a) sensor# 4-2 removed, (b)sensor# 5-1 removed.

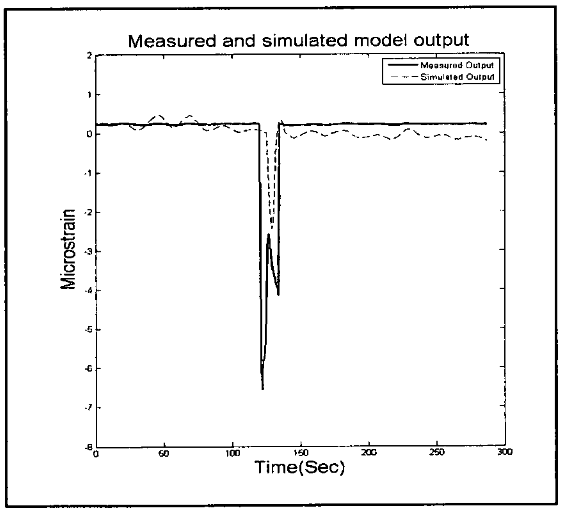




(a) (b)

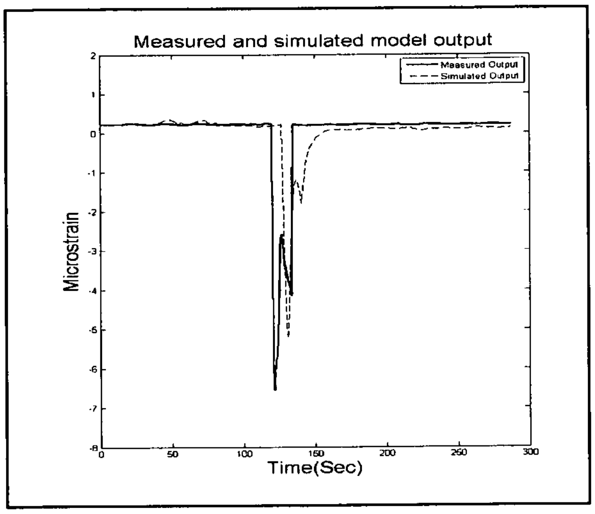
Figure C.4 Measured and Simulated output of March 2006, (a) sensor# 5-2 removed, (b) sensor# 6-1 removed.



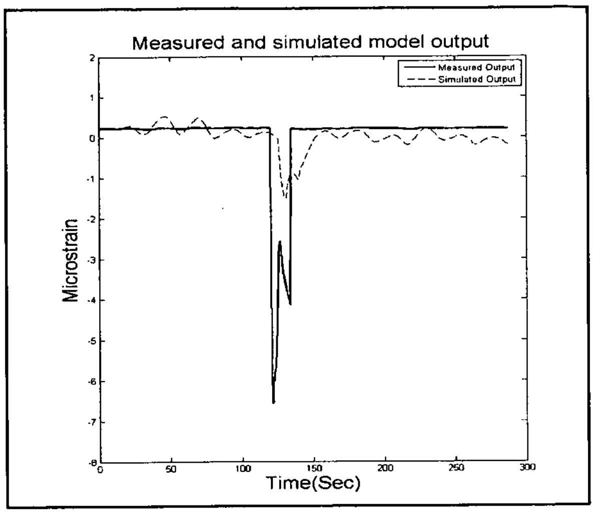


(a) (b)

Figure C.5 Measured and Simulated output of March 2006, (a) sensor# 6-2 removed, (b) sensor# 7-1 removed.

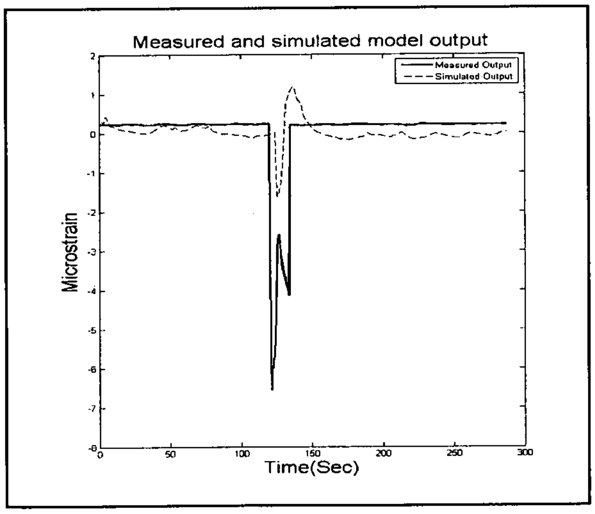


(a)

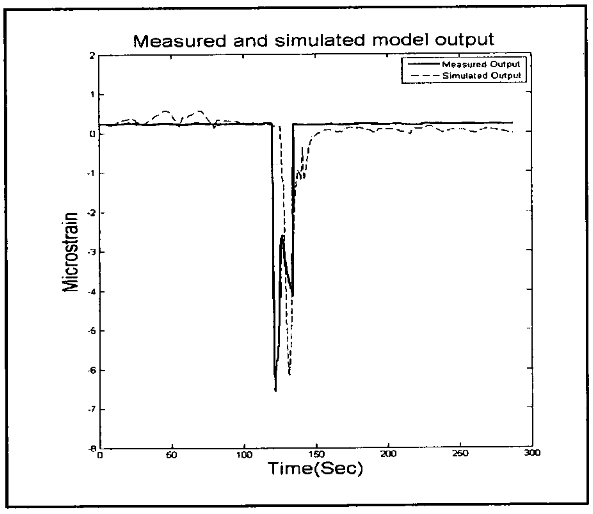


(b)

Figure C.6 Measured and Simulated output of March 2006, (a) sensor# 7-2 removed, (b) sensor# 8-1 removed.



(a)

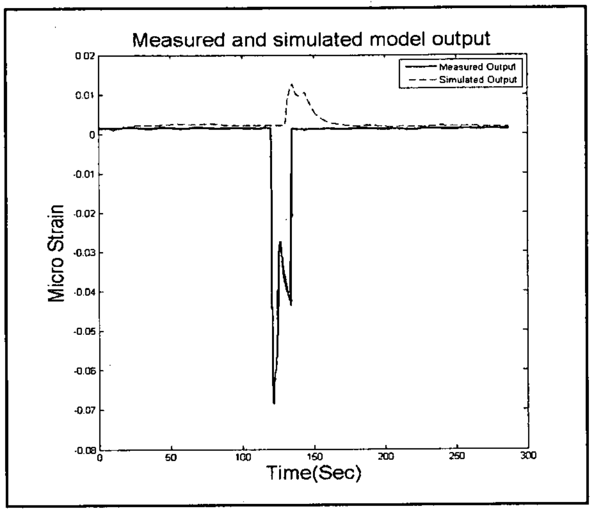


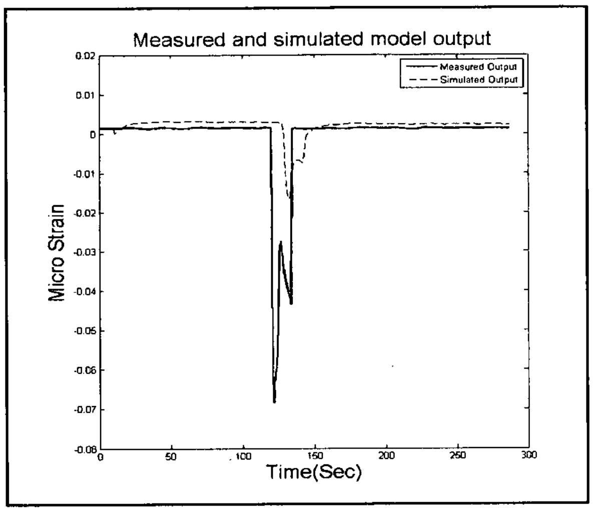
(b)

Figure C.7 Measured and Simulated output of March 2006, (a) sensor# 8-2 removed, (b) sensor# 9-1 removed.

Appendix D

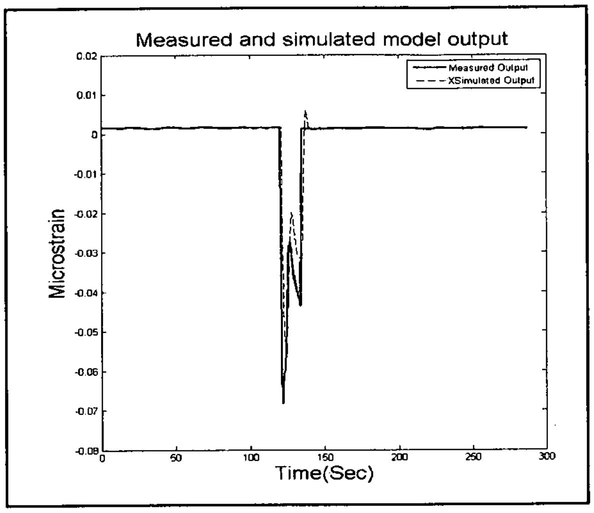
D.1 Graphical output of defective sensors detection of Portage Creek Bridge. The details are described in Chapter 5, section 5.4.

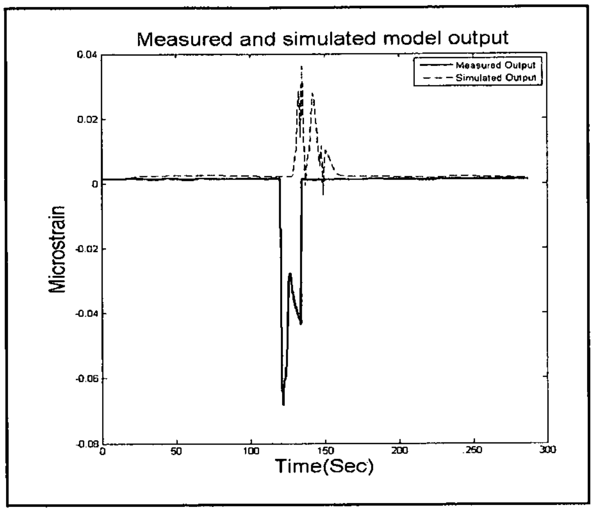




(a) (b)

Figure D.1 Measured and Simulated output of March 2006, (a) 1st 4 sensors, (b) 2nd 4 sensors.

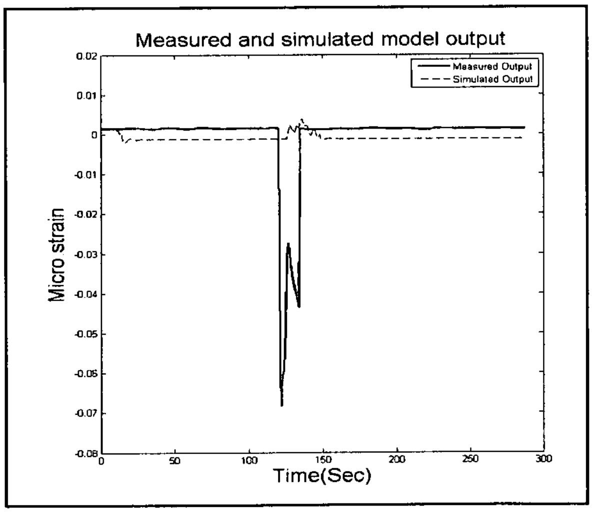


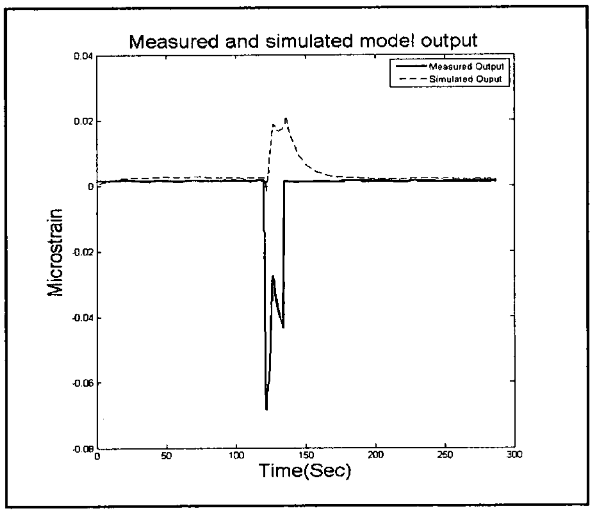


(a) (b)

Figure D.2 Measured and Simulated output of March 2006, (a) 1st 2sensors,

(a) 2nd 2 sensors,





(a) (b)

Figure D.3 Measured and Simulated output of March 2006, (a) 1st 2sensors,

(a) 2nd 2 sensors.