Multivariate Statistical Process Monitoring On Structural Fault

by

Afifah Binti Mohamad Saidi

16968

Dissertation submitted in partial fulfillment of

the requirements for the

Bachelor Of Engineering (Hons)

(Chemical Engineering)

MAY 2015

Universiti Teknologi PETRONAS

32610, Bandar Seri Iskandar

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Multivariate Statistical Process Monitoring On Structural Fault

by

Afifah Binti Mohamad Saidi

16968

A project dissertation submitted to the

Chemical Engineering Programme

Universiti Teknologi PETRONAS

in partial fulfilment of the requirement for the

BACHELOR OF ENGINEERING (Hons)

(CHEMICAL ENGINEERING)

Approved by,

____

(Ir. Dr. Abd Halim Shah Bin Maulud)

UNIVERSITI TEKNOLOGI PETRONAS

BANDAR SERI ISKANDAR, PERAK

May 2015

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

____

AFIFAH BINTI MOHAMAD SAIDI

ABSTRACT

In modern plants there are many operating variables measured by sensors and logged into the process system database. Thus the amount of available data needs to be analyzed is enormous and they are highly correlated. This creates a demand for a system to monitor, control and analyzes this complex processes data to ensure the monitored process stays within desired conditions, by recognising anomalies in the process behaviour and subsequently correcting it. Statistical Process Monitoring (SPM) meets the demands; it is a system capable of detecting fault occurrence. In this context, the anomalies or faults studied is specifically the fault in structure, which is a type of fault resulted from an alteration of the processes main characteristics. SPM can be broken down into two methods which are univariate and multivariate methods. Multivariate methods or multivariate statistical process monitoring (MSPM) method take into account the correlation among the process variables and measurements; and it is capable to accurately characterize the behaviour of the processes, subsequently detecting faults, for which univariate method unable to adequately perform. MSPM method studied in this research project is Dynamic Principal Component Analysis (DPCA) method specifically on its structural fault detection ability, along with Hotelling's (T^2-statistic) and Squared Prediction Error (Q-statistic) techniques. The accuracies of fault detection ability of DPCA method will be compared with Principal Component Analysis (PCA) method.

ACKNOWLEDGEMENTS

The author gratefully acknowledges the supervisor for this project, Ir. Dr. Abd Halim Shah Bin Maulud, for his continuous support and assistance provided throughout this project and Universiti Teknologi PETRONAS for providing the necessary equipment and facilities to conduct this project.

TABLE OF CONTENTS

CERTIFICATION OF APPROVAL		ii
CERTIFICATION OF ORIGINALITY		iii
ABSTRACT		iv
ACKNOWLEDGEMENT		v
TABLE OF CONTENTS		vi
LIST OF FIGURES		viii
LIST OF TABLES		ix
LIST OF APPENDICES		x
CHAPTER 1:	**INTRODUCTION**	1
	1.1 Background Study	1
	1.2 Problem Statement	3
	1.3 Objectives & Scopes of Study	4
CHAPTER 2:	**LITERATURE REVIEW & THEORY**	5
	2.1 Types of Process Fault	5
	2.2 Statistical Process Monitoring	6
	2.3 Principal Component Analysis (PCA)	7
	2.4 Dynamic Principal Component Analysis (DPCA)	9
	2.5 Fault Detection	9
	2.5.1 Hotelling's (T^2-Statistic)	9
	2.5.2 Squared Prediction Error, SPE (Q-Statistic)	11
CHAPTER 3:	**METHODOLOGY**	12
	3.1 CSTR Model Development	12

3.1.1 CSTR Modelling Equation 14

3.2 Simulation of Faults 16

3.3 Project Tools 17

3.4 Gantt Chart and Key Milestone 18

3.5 Project Process Flow 19

CHAPTER 4: RESULT AND DISCUSSION 20

4.1 CSTR Simulation Model 20

4.2 Base Data 21

4.3 Fault Data 23

4.3.1 Drift in Reaction Kinetics 23

4.3.2 Drift in Heat Transfer Coefficient 25

4.3.3 Drift in Heat Transfer Coefficient and Reaction Kinetics) 27

4.4 Statistical Analysis 29

4.4.1 PCA and DPCA 29

4.4.2 T^2-Statistics and Q-Statistic 29

4.5 Summary of Fault Detection Time 36

CHAPTER 5: CONCLUSION AND RECOMMENDATION 41

REFERENCES 42

APPENDICES 45

LIST OF FIGURES

Figure 1.1	Process Monitoring Loop	2
Figure 1.2	General Fault Diagnosis Framework	3
Figure 3.1	Schematic representation of CSTR	13
Figure 3.2	Matlab Simulink	17
Figure 4.1	CSTR MATLAB Simulation Model	20
Figure 4.2	Product concentration base data .	21
Figure 4.3	Product temperature base data	21
Figure 4.4	Product flowrate base data	22
Figure 4.5	Product concentration fault data .	23
Figure 4.6	Product temperature fault data	23
Figure 4.7	Product flowrate fault data	24
Figure 4.8	Product concentration fault data .	25
Figure 4.9	Product temperature fault data	25
Figure 4.10	Product flowrate fault data	26
Figure 4.11	Product concentration fault data	27
Figure 4.12	Product temperature fault data	27
Figure 4.13	Product flowrate fault data	28
Figure 4.14	PCA T^2-statistics	30
Figure 4.15	DPCA T^2-statistics	30
Figure 4.16	PCA T^2-statistics	31
Figure 4.17	DPCA T^2-statistics	31
Figure 4.18	PCA T^2-statistics	32
Figure 4.19	DPCA T^2-statistics	32
Figure 4.20	PCA Q-statistics	33
Figure 4.21	DPCA Q-statistics .	33
Figure 4.22	PCA Q-statistics	34
Figure 4.23	DPCA Q-statistics	34
Figure 4.24	PCA Q-statistics	35
Figure 4.25	DPCA Q-statistics	35

LIST OF TABLES

Table 3.1 Default operating parameters 14

Table 3.2 Gantt Chart and Key Milestone 18

Table 4.1 T^2-statistics thresholds 29

Table 4.2 Detection time Drift in Reaction Kinetics (T^2-statistic) 36

Table 4.3 Detection time Heat Transfer Coefficients (T^2-statistic) 36

Table 4.4 Detection time Drift in Reaction Kinetics & Heat Transfer Coefficients (T^2-statistic) 36

Table 4.5 Detection time Drift in Reaction Kinetics (Q-statistic) 37

Table 4.6 Detection time Heat Transfer Coefficients (Q-statistic) 37

Table 4.7 Detection time Drift in Reaction Kinetics & Heat Transfer Coefficients (Q-statistic) 37

LIST OF APPENDICES

Appendix 1	Matlab Source Code of CSTR System (for base data)	45
Appendix 2	PCA for Base Data	50
Appendix 3	PCA for Drift in Reaction Kinetics	52
Appendix 4	PCA for Drift in Heat Transfer Coefficient	54
Appendix 5	PCA for Drift in Reaction Kinetics & Heat Transfer Coefficient	55
Appendix 6	DPCA for Base Data .	57
Appendix 7	DPCA for Drift in Reaction Kinetics	59
Appendix 8	DPCA for Drift in Heat Transfer Coefficient	62
Appendix 9	DPCA for Drift in Reaction Kinetics & Heat Transfer Coefficient	64

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND STUDY

Robert M. Solow, an economist at Massachusetts Institute of Technology received a Nobel Prize in 1987, for his work in identifying the sources of economic growth. Professor Solow came with a conclusion that the immensity of an economy's growth is the result of technological advances (Crowl & Louvar, 2011). It is also can be deduce that the industrial growth is too, rely on technology advancement. Industrial evolution, which mean more complex processes, high pressure facilities operation, more reactive chemicals, and many more complexities in controlling this industrial plant operations. Not to forget that, this industry has to cope with the demands on production efficiency, to maintain and improve product quality, and the pressure to comply with increasingly stringent safety and environmental regulations.

Modern industrial processes are dealing with enormous amount of data, coming from many variables that need to be monitored and recorded continuously day to day. Standard (traditional) process controllers maintain operation conditions, by compensating disturbances and changes occurring in the process, however its ability is limited. Changes in processes, in which standard controllers cannot handle adequately, are called faults. A fault is defined as an unpermitted deviation of characteristic property or variable of the system. Fault can severely deteriorate production process, causing production outage, and worst case scenario, catastrophic industrial incident. Thus, it is significant for these processes, a rapid fault detection method that response effectively to fault occurrences. Fault detection, along with fault identification, fault diagnosis and process recovery form the basis of process monitoring.

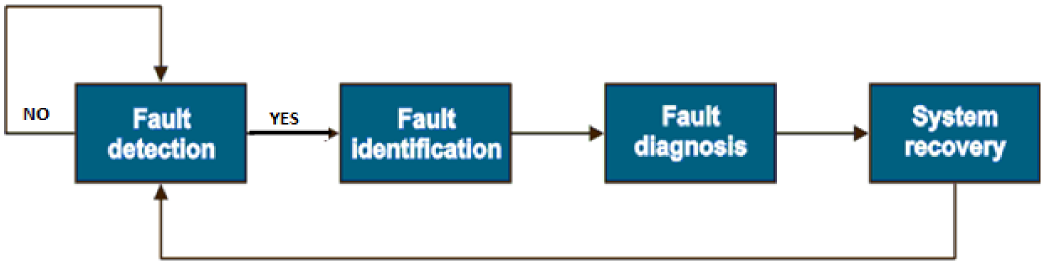


FIGURE 1.1 Process Monitoring Loop

Process monitoring goal is to ensure the success of the planned operations by recognizing anomalies in the process behavior. This project studies a process monitoring method known as multivariate statistical process monitoring (MSPM), which capable of analyzing a vast number of highly correlated variable data in a process plant control system. The MSPM methods studied are Principal Component Analysis (PCA) and Dynamic Principal Component Analysis (DPCA) for relationship modelling which exists between process variables, along with Hotelling's T^2-statistic, and Q-statistic (Squared Prediction Error) are used for fault detection, due to their capability in providing an indication of abnormal variability within and outside normal process workspace.

1.2 PROBLEM STATEMENT

As mentioned earlier, complex processes, with thousand of operating variables data needed to be continuously monitored and controlled to ensure the processes operating within the specified operating conditions. Commonly in industry, the monitoring system of these processes works based on a general principle framework, in this context it is the general fault diagnosis framework. Based on Figure 1, there are many types of disturbances or faults that could occur in a process.

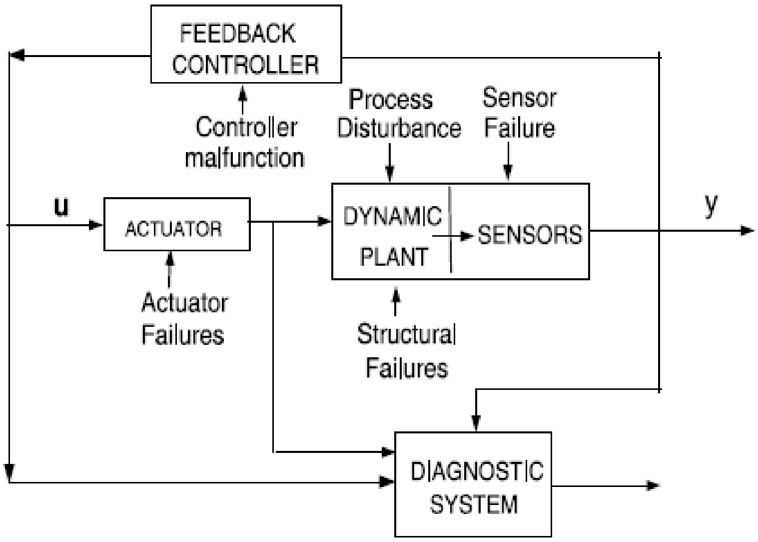


FIGURE 1.2 General Fault Diagnosis Framework (Venkatasubramanian et.al, 2003)

In this project, focus is given on structural failures, of which it is capable to "result in a change in the information flow between various variables" (Venkatasubramanian, et.al. 2003). An example of a structural failure would be failure of a controller, a stuck valve, or a leaking pipe and so on. In this project, structural fault are simulated in Continuous Stirred Tank Reactor (CSTR) in Matlab Simulink, the fault data obtained will be statistically analyzed using Principal Component Analysis (PCA) and Dynamic Principal Component Analysis (DPCA) method, along with Hotelling's T^2-statistic, and SPE's Q-statistic, and their fault detection ability will be mutually compared. Principal Component Analysis (PCA) is a variable-reduction technique, and it determines the number of components to retain, for which technically, a principal component can be defined as a linear combination of optimally-weighted observed variables. An extension of PCA to handle process dynamics are known as Dynamic Principal Component Analysis (DPCA).

In this project, structural fault is the main object of study. Fault in structure "occur due to hard failures in equipment" (Venkatasubramanian et.al 2003). Change in a process structure will caused a change in the information flow between various variables. Thus, it means that the data collected from various variables are highly correlated when they are affected by a structural fault.

1.3 OBJECTIVES AND SCOPE OF STUDY

The main objective of this project is to investigate the structural fault detection performance of DPCA method, along with Hotelling's T^2-statistic, and Q-statistic (Squared Prediction Error) technique, for which DPCA method fault detectability accuracies will be compared with PCA method. The scope of this project is to utilize CSTR simulation model to generate structural fault.

To achieve the main objective, the sub-objectives of this project are as the following:

1. To develop structural fault case study using Continuous Stirred Tank Reactor (CSTR) computer simulation model in MATLAB Simulink
2. To compare fault detection performance of PCA and DPCA using, Hotelling's T^2-statistic and Q-statistic (SPE) techniques

CHAPTER 2
LITERATURE REVIEW & THEORY

2.1 TYPES OF PROCESS FAULTS

In general, fault can be defined as a deviation from an acceptable range of an observed variable or designated parameter associated with a process (Himmelblau, 1978). This defines a fault as a process anomaly; it can be a case such as high temperature in a reactor or low product quality and so on. Referring to Figure 1, there are three types of faults, which are “structural fault (structural failures & controller malfunction), variable fault (process disturbance), and gross errors (sensor and actuator failure)” (Venkatasubramanian et.al. 2003).

Variables fault is a change of variable parameter that exceeds the acceptable range of the observed variables. “Failures in parameter happen when there is a disturbance entering the process from the environment through one or more variables” (Venkatasubramanian et.al. 2003). An example for such fault is a change in temperature or change in concentration of reactant from its steady state value in reactor feed. Variable faults are rather easily detectable via univariate process monitoring methods. Fault in structures is the changes or deviation of the main characteristics governing the processes from ideal operating conditions (Hollender, 2010). Structural faults are rather difficult to detect compared to variable faults and this is the type of faults being studied in this project.

2.2 STATISTICAL PROCESS MONITORING

The objective of SPM is to ensure success of the planned operation by recognizing anomalies of the behaviour (Chiang & Russell, 2001). SPM can be divided into univariate methods and multivariate methods. Modern process control system becoming more complex and the current commonly employed SPM method (univariate) are inadequate. Univariate statistical charts ignore the correlation among other variables and measurements; thus it is incapable to accurately characterize the behaviour of the current industrial process (Chiang & Russell, 2001).

Univariate approaches are adequate for investigating and understanding simple systems. Classical univariate data analysis plot the columns of data together two at a time, followed by plotting each variable for all samples and look for trends. These approaches lead to frustration because of information overload and the time and effort required making each plot. Therefore, assumptions are made due to absence of an effective way to analyse all data simultaneously, which eventually lead to inconclusive result. Univariate methods may provide an oversimplistic and overoptimistic assessment of the data. Mean, average, and standard deviations and a lot of other statistics that describe one variable, but provide no information on how that one variable relates to others. Univariate method failed to detect the variable inter-relationships that may exist, as it assumes all variables are independent of each other.

Multivariate Statistical Process Monitoring (MSPM) address some of the limitations of univariate monitoring techniques by considering all the data simultaneously and extracting information on the ‘directionality’ of the process variations. MSPM are a commonly used method to solve problems occurred in plant operation caused by uncertainty, disturbances, faults and incomplete knowledge of the process, and its’ performance depends on how well the model describes relationships between the variables (Venkatasubramanian et.al. 2003).

MSPM also provides monitoring charts that can detect fault and gives warning signal earlier than the classical univariate chart (Chiang & Russell, 2001).

Multivariate methods can be as simple as analysing two variables right up to millions, "it address univariate methods weakness by considering all the data simultaneously, and extracting information on the directionality of the variations" of one variable relative to the other (Martin et.al. 1996). This is known as covariance or correlation, it describe the influence that variables has on each other. Multivariate methods identify variables that contribute most to the overall variability in the data, it helps to isolate those variables that are related – in other words, that co-vary with each other. Multivariate methods present the result in a highly graphical manner, into a one interpretable picture.

2.3 PRINCIPAL COMPONENT ANALYSIS (PCA)

The basis of MSPM is the projection methods of principal components analysis (PCA), for which the primary objectives are data summarisation, classification of variables, outlier detection, early warning of potential anomalies, as well as providing data for fault identification. PCA reduce the dimensionality of the variables data, by forming a new set of variables called principal components, which are a linear combination of the original measured variables and which explain the maximal amount of variability in the data (Russell et.al. 2000). These principle components are sets of new uncorrelated variables.

PCA determines a set of orthogonal vectors called loading vector and it is ordered by the amount of variance explained in the loading vector direction. PCA can greatly simplify data into lower-dimensional space without any loss of variance between variables, and finding a few linear combinations which can be used to summarized the data with a minimal loss of information. It preserves the correlation structure between process variables and captures variability of the data.

A set of *n observations* and *m process variables* stacked into a *matrix X*:

$$X = \begin{bmatrix} x_{11} & x_{12} & . & . & . & x_{1m} \\ x_{11} & x_{12} & . & . & . & x_{1m} \\ \vdots & \vdots & . & . & . & \vdots \\ x_{11} & x_{12} & . & . & . & x_{1m} \end{bmatrix} \tag{1}$$

The *loading vectors* are calculated by solving the stationary points of the optimization problem

$$\max_{v \neq 0} \frac{v^T X^T X v}{v^T v} \tag{2}$$

where $v \in R^m$

The stationary points of (1) can be computed via the singular value decomposition (SVD):

$$\frac{1}{\sqrt{n-1}} X = U \Sigma V^T \tag{3}$$

where $U \in R^{n \times n}$ and $V \in R^{m \times m}$ are unitary matrices and $\Sigma \in R^{n \times m}$ is the matrix containing the non-negative singular values.

Eq (3) is equivalent to solving an *eigenvalue decomposition* of *the covariance matrix S* of the data set:

$$S = \frac{1}{n-1} X^T X = V \Lambda V^T \tag{4}$$

where *V (loading matrix) is the orthogonal* and *Λ (eigenvalue) is the diagonal* (Golub & Loan, 1983) and the *new loading matrix P is then built from the columns of V.*

The data set is then projected into a lower dimensional score matrix *T*:

$$T = XP \tag{5}$$

2.4 DYNAMIC PRINCIPAL COMPONENT ANALYSIS (DPCA)

Monitoring system based on PCA approach assumes observations at any time instant statistically independent to observations in the past time. However, in typical industrial processes, assumption is only valid for long sampling intervals i.e. 2 to 12 hours. X^T is the m-dimensional observation vector in the training set at time instance t. By performing PCA on the *data matrix in Eq. (6), a multivariate autoregressive (AR) model* is extracted from the data. The *Q-statistic is then the squared prediction error of the model*. If enough *lags l* are included in the data matrix, the Q-statistic becomes statistically independent between time instances, and the *threshold Eq. (12)* is justified. This approach is known as dynamic PCA (DPCA).

For shorter sampling intervals and faster fault detection, the PCA methods are extended to take into account the serial correlations, by augmenting each observation vector with the previous l observations and stacking the data matrix as following (Chiang & Russell, 2001):

$$X(l) = \begin{bmatrix} X_t^T & X_{t-1}^T & \cdots & X_{t-l}^T \\ X_{t-1}^T & X_{t-2}^T & \cdots & X_{t-l-1}^T \\ \vdots & \vdots & \ddots & \vdots \\ X_{t+l-n}^T & X_{t+l-n-1}^T & \cdots & X_{t-n}^T \end{bmatrix} \tag{6}$$

2.5 FAULT DETECTION

2.5.1 Hotelling's (T^2-Statistic)

Multivariate process control techniques were established by Hotelling in his 1947 pioneering paper, in which he applied multivariate process control methods to a bombsights problem (Hotelling, 1947). Multivariate process control procedure should fulfill four conditions: (1) an answer to the question 'Is the process in control?' must be available; (2) an overall probability for the event ‘Procedure diagnoses an out-of-control state inaccurately’ must be specified; (3) the relationships among the variables–attributes should be taken into account; and (4) an answer to the question ‘If the process is out of control, what is the problem?’ should be available (Jackson, 1991). Multivariate control chart for process mean is based greatly upon Hotelling's-T^2 distribution, which was introduced by Harold Hotelling on 1947. Hotelling's T^2-distribution is the multivariate analogue of the univariate t-distribution for the use of standard value μ or individual observations [(Sultan, 1986), (Blank, 1988), & (Morrison, 1990)].

For an *observation vector x* and assuming that *$\Lambda=\Sigma T\Sigma$ is invertible*, the T^2 –statistic can be obtained from the PCA and DPCA representation respectively Eq. (5) as such:

$$T^2 = x^T V (\Sigma^T \Sigma)^{-1} V^T x \tag{7}$$

By including the matrix P the *loading vectors* associated with the *a largest singular values*, the T^2-statistic is then computed as (Jackson & Mudholkar, 1979) :

$$T^2 = x^T V P \Sigma_a^{-2} P^T x \tag{8}$$

where, *Σ_a contains the first a rows and columns of Σ,* and *x is an observation vector of dimension m*. The *T^2-statistic threshold* is:

$$T_\alpha^2 = \chi_\alpha^2(a) \tag{9}$$

and for a normalised data set, the *T^2-statistic threshold is derived as:*

$$T_\alpha^2 = \frac{(n^2-1)a}{n(n-a)} F_\alpha(a, n-a) \tag{10}$$

where *F_α (a, n - a)* is *the upper 100α% critical point of the F-distribution* with *a* and *n - a degrees of freedom* (Kourti & MacGregor, 1995).

2.5.2 Squared Prediction Error, SPE (Q-Statistic)

The Q-statistic, also known as the squared prediction error (SPE), is a squared 2-norm measuring the deviation of the observations to the lower dimensional PCA representation (Russell et.al. 2000).

For robust monitoring of the portion of the measurement space corresponding to the *m – a smallest singular values, Q-statistic* would be of better use:

$$Q = r^T r, \qquad r = (1 - PP^T)x \tag{11}$$

where *r is the residual vector, a projection of observation x into residual space* (Russell et.al. 2000). *The threshold for Q-statistic* can be approximated as (Jackson & Mudholkar, 1979) :

$$Q_a = \theta_1 \left[\frac{h_0 C_\alpha \sqrt{2\theta_2}}{\theta_1} + 1 + \frac{\theta_2 h_0 (h_0 - 1)}{\theta_1^2} \right]^{1/h_0} \tag{12}$$

Where c_α is the normal deviate which corresponds to *(1- α) percentile*.

$$\theta_i = \sum_{j=a+1}^{n} \sigma_j^{2i} \qquad\qquad h_0 = 1 - \frac{2\theta_1\theta_3}{3\theta_2^2} \tag{13}$$

CHAPTER 3
METHODOLOGY

3.1 CSTR MODEL DEVELOPMENT

In a normal operation of a continuous flow stirred-tank reactor (CSTR), the contents are well stirred and it runs with continuous flow of reactants, as well as the products. The CSTR normally runs at a steady state condition, with a uniform distribution of concentration and temperature throughout the reactor. For this project, the CSTR system will be modelled using Matlab Simulink software. This model is then used to generate a sample of baseline data (without faults) to be tested and used as benchmark later on, thus the following assumption are made:

1. Heat losses from the process are negligible (well insulated)
2. The mixture density and heat capacity are assumed constant
3. There are no variations in concentration, temperature, or reaction rate throughout the reactor as it is perfectly mixed
4. The exit stream has the same concentration and temperature as the entire reactor liquid
5. The overall heat transfer coefficient is assumed constant

6. No energy balance around the jacket is considered. This means that the jacket temperature can directly be manipulated in order to control the desired reactor temperature.
7. The reactor is a flat bottom vertical cylinder and the jacket is around the outside and the bottom.

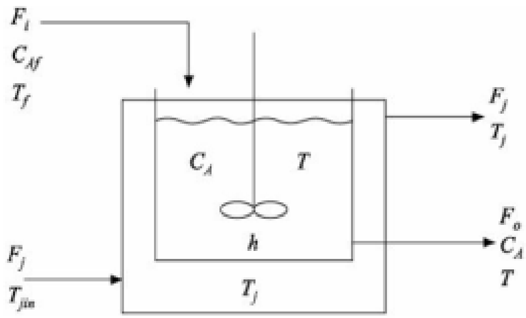


FIGURE 3.1: Schematic representation of CSTR

The CSTR simulation model in Matlab Simulink will be build based on these predefined parameters and operating conditions (Table 3.1).

TABLE 3.1 Default operating parameters (Jana, 2011).

Operating Parameter	**Notation**	**Value**
Cross sectional area of the reactor, ft^2	A_c	10.36
Concentration of reactant A in the exit stream, lb-mol/ft^3	C_A	0.05
Concentration of reactant A in the feed stream, lb-mol/ft^3	C_{Ain}	0.9
Diameter of cylindrical reactor, ft	D	3.6319
Activation energy, BTU/ lb-mol	E	30000
Volumetric feed flow rate, ft^3/h	F_{in}	20
Height of the reactor liquid, ft	h	3.8610
Heat of reaction, BTU/ lb-mol	$-\Delta H_r$	-30000

Universal gas constant, BTU/ (lb-mol)(R)	R	1.987
Frequency factor, h^{-1}	α	7.08×10^{10}
Multiplication of mixture density and heat capacity, BTU/ (ft^3)(R)	ρC_p	37.5
Reactor temperature, R	T	650
Feed temperature, R	T_{in}	600
Jacket temperature, R	T_j	70.0
Overall heat transfer coefficient, BTU/(ft^2)(R)(h)	U_i	150

3.1.1 CSTR Modelling Equation

Total Continuity Equation:

Mass inflow rate = F_i

Mass outflow rate = F_o

Rate of mass accumulation within reactor = $\dfrac{d(\rho V)}{dt} = \dfrac{d(\rho A_c h)}{dt}$ (14)

A_c is cross-sectional area of reactor and h is the height of the reactor liquid.

$$\frac{d(\rho V)}{dt} = (F_i - F_o)\rho$$

$$\frac{dV}{dt} = F_i - F_o \tag{15}$$

The reactor holdup, V and the exit flow rate F_o can be related as:

$$F_o \propto \sqrt{V}$$

For this CSTR, $F_o = \sqrt{10 A_c h}$ (16)

Combining equations 15 and 16:

$$\frac{dh}{dt} = \frac{F_i}{A_c} - \sqrt{\frac{10h}{A_c}} \tag{17}$$

Component Continuity Equation:

Mass inflow rate component A = F_iC_{Af},

Mass outflow rate component A = F_oC_A,

Rate of generation of component A = – $(-r_A)V$

Rate of accumulation of component A within the reactor = $\frac{d(VC_A)}{dt}$

where $-r_A$ is the rate of consumption of chemical species A. The basic balance equation then becomes,

$$\frac{d(VC_A)}{dt} = F_iC_{Af} - F_oC_A - (-r_A)V$$

$$C_A\frac{dV}{dt} + V\frac{dC_A}{dt} = F_iC_{Af} - F_oC_A - (-r_A)V \tag{18}$$

Substituting equation 15 into 17 and simplifying,

$$\frac{dC_A}{dt} = \frac{F_i}{A_ch}(C_{Af} - C_A) - (-r_A) \tag{19}$$

For the given first-order reaction,

$$\begin{aligned} -r_A &= kC_A \\ &= \alpha \exp\left(\frac{-E}{RT}\right)C_A \end{aligned} \tag{20}$$

Combining equations 18 and 19,

Energy input rate = $F_iC_pT_f$

Energy output rate = $F_oC_pT + U_iA_h(T-T_j)$

Energy added by exothermic reaction = $(-\Delta H)V\alpha \exp\left(\frac{-E}{RT}\right)C_A$ (21)

Energy accumulation rate:

$$\frac{d(V\rho C_pT)}{dt} = F_i\rho C_pT_f - F_o\rho C_pT - U_iA_h(T-T_j) + (-\Delta H)V\alpha \exp\left(\frac{-E}{RT}\right)C_A \tag{22}$$

Using equation 22 and further simplifying:

$$\frac{dT}{dt} = \frac{F_i}{A_c h}\left(T_f - T\right) + \left(\frac{-\Delta H}{\rho C_p}\right)\alpha \exp\left(\frac{-E}{RT}\right)C_A - \frac{U_i A_h}{\rho C_p A_c h}\left(T - T_j\right) \qquad (23)$$

And therefore, this is the final form of the energy balance equation.

3.2 SIMULATION OF FAULTS

Once base data is generated by MATLAB Simulink model, a simulation of non-ideal operating conditions is fabricated and run to obtain a set of structural fault data. This is done in MATLAB Simulink by adding disturbances, for example; a sine wave function or random number function for measurement noise. The data will be studied using the PCA, DPCA, Hotelling's and SPE to detect the presence of any such faults. The results from each fault detection method would then be compared to investigate the accuracies of each methods utilized. For fault simulation, the Simulink model is slightly modified. The structural fault studied in this project will be as following:

1) Drift in heat transfer coefficient
    - Eg: fouling of heat exchanger
    - Drift ranges: 1%, 5%, and 10%

2) Drift in reaction kinetics
    - Eg: catalyst deactivation
    - Drift ranges: 1%, 5% and 10%

3) Simultaneous simulation (Drift in heat transfer coefficient and reaction kinetics)
    - Eg: simultaneous simulation (Eg: fouling of heat exchanger and catalyst deactivation)
    - Drift ranges: 1%, 5% and 10%

A sine wave function, random number function or ramp tool can be added added as disturbances input to simulate the structural faults occurrence at 1%, 5% and 20% drift. Therefore, nine fault data sets corresponding to the different drift levels are generated, with each set consist of n = 5000 observations.

3.3 PROJECT TOOLS

The primary tool used for this project is the MATLAB software, specifically the SIMULINK simulation environment. MATLAB, (MATrix LABoratory), is a high level computing software for numerical computations and graphics developed by MathWorks. The SIMULINK simulation environment is a separate feature of MATLAB which is a data graphical programming tool. It is widely used in control theory as it allows the user to model, simulate and analyse dynamic systems.

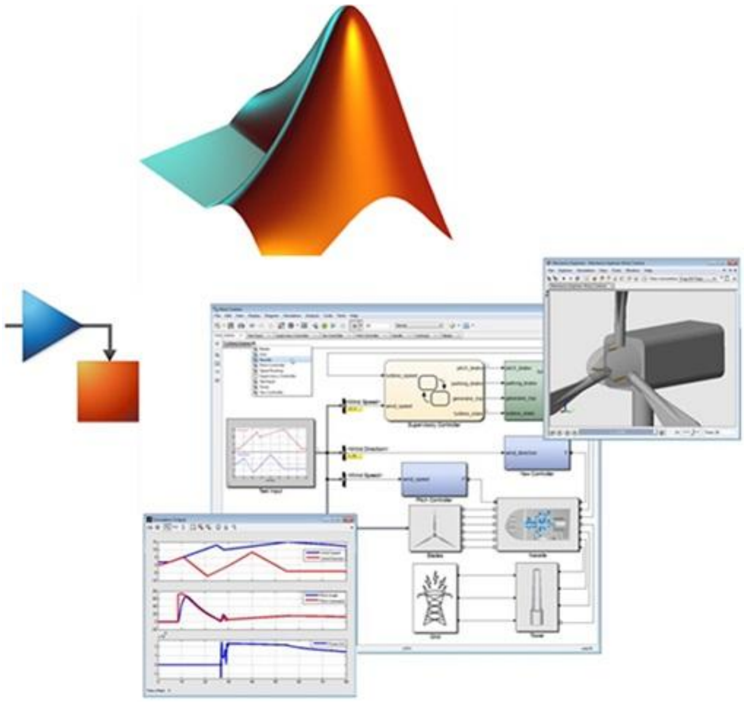


FIGURE 3.2 MATLAB Simulink

3.4 GANTT CHART AND KEY MILESTONE

Table 3.2 Gantt Chart and Key Milestone

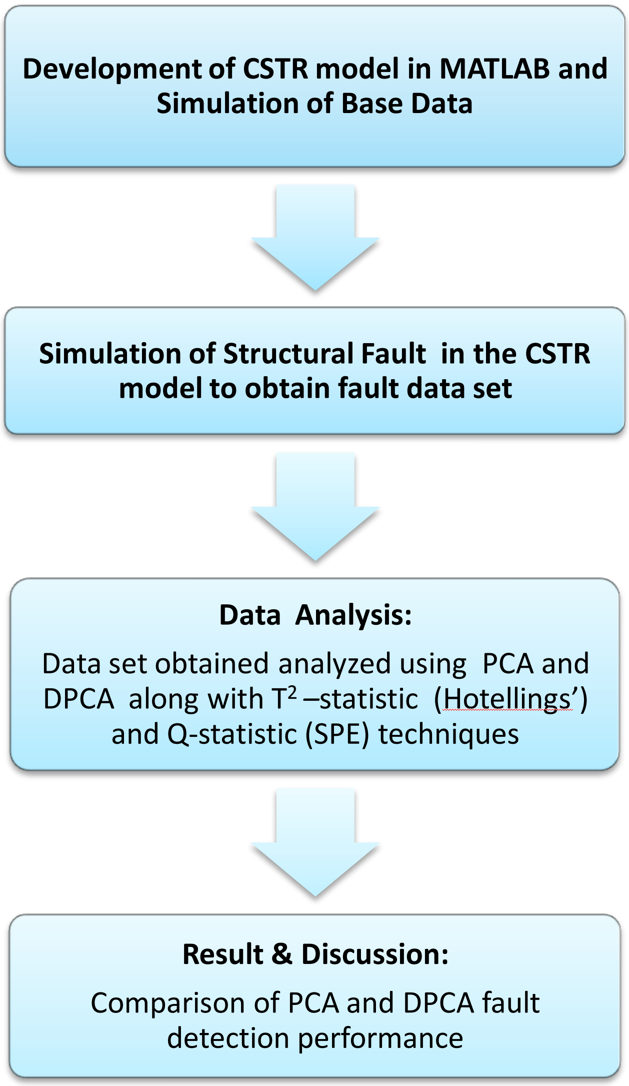
No	Details	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1.	Development of CSTR Simulation Model														
2.	Generation of Base Data and Fault Data														
3.	Data Analysis using using PCA and DPCA method, with T^2-Statistic and Q-statistic Technique														
4.	Preparation and Submission of Progress Report														
5.	Project Work Continues														
6.	Pre-EDX														
7.	Submission of Draft Report														
8.	Submission of Dissertation (soft bound)														
9.	Submission of Technical Paper														
10.	Oral Presentation														
11.	Submission of Project Dissertation (hard bound)														

Gantt chart

Key Milestone

3.5 PROJECT PROCESS FLOW

This is the process flow for this research project that must be so that the objectives of the study can be successfully achieved.



CHAPTER 4

RESULT AND DISCUSSION

4.1 CSTR SIMULATION MODEL

Based on the CSTR simulation model, there are three main inputs, namely feed flowrate, temperature and concentration denoted as F_{in}, T_{in} and $C_{a,in}$. The three main outputs which are recorded are the product flowrate, temperature and concentration denoted as F, T and C_a. The CSTR computer simulation modelled in MATLAB Simulink is shown in Figure 4.1.

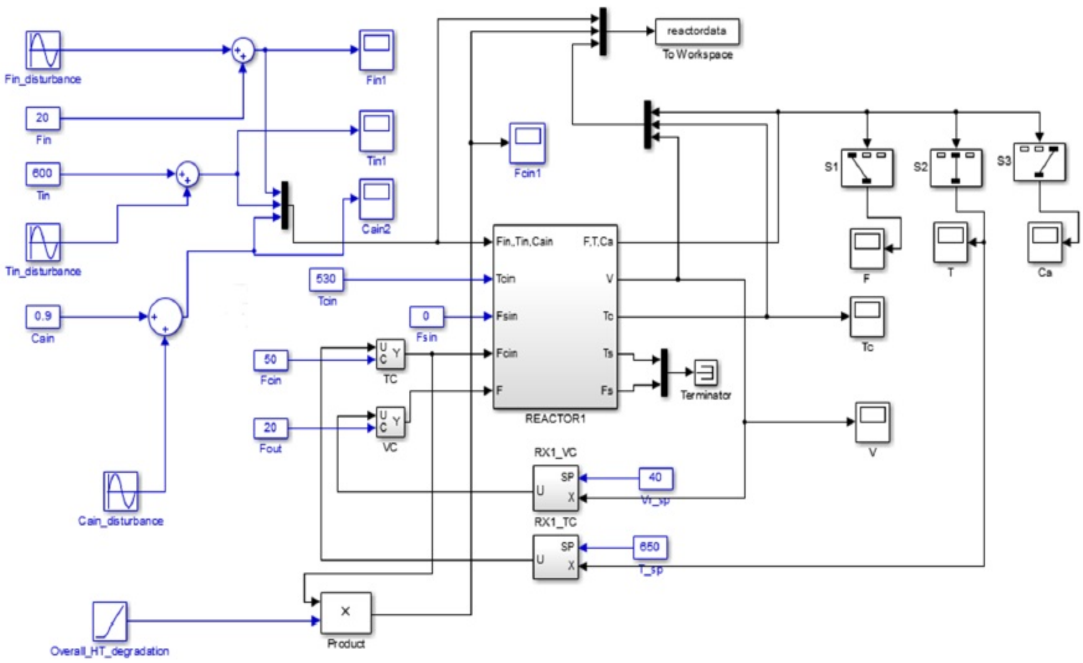


FIGURE 4.1 CSTR MATLAB Simulation Model

4.2 BASE DATA

CSTR simulation model will generate two types of output data, the first one is a base data. This base data set is then *normalized to 0 mean and unit variance* and becomes the input for the *MATLAB function princomp*, which calculates *the loading vector (V), eigenvalues (Λ) and the score matrix (T).* Only the loading vectors corresponding to the *a largest singular values* will be retained.

The base data obtained in the graphs (figure 6, 7, and 8) shown below above are the base data generated without structural faults. The effects of noise are evident in all the graphs.

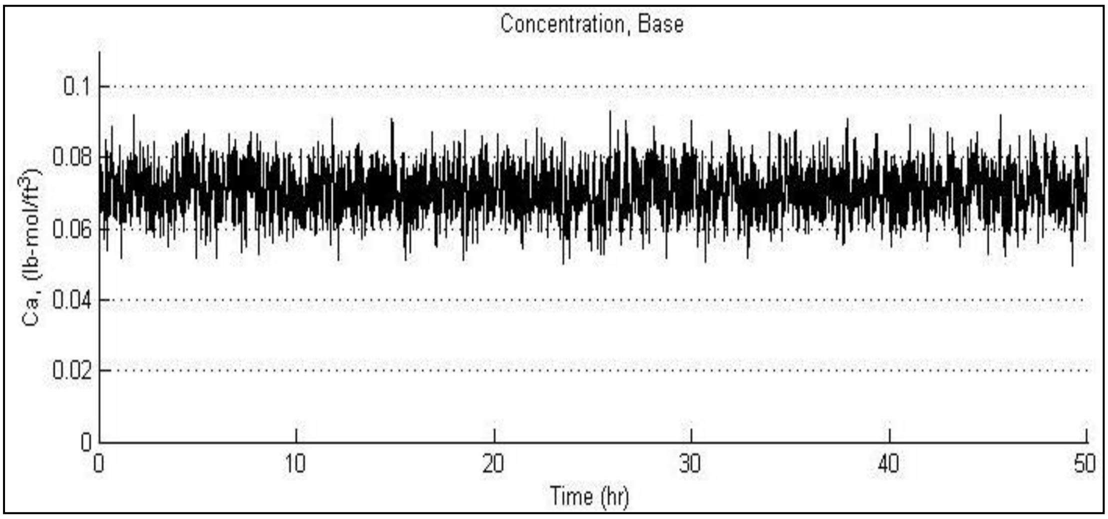


FIGURE 4.2 Product concentration base data

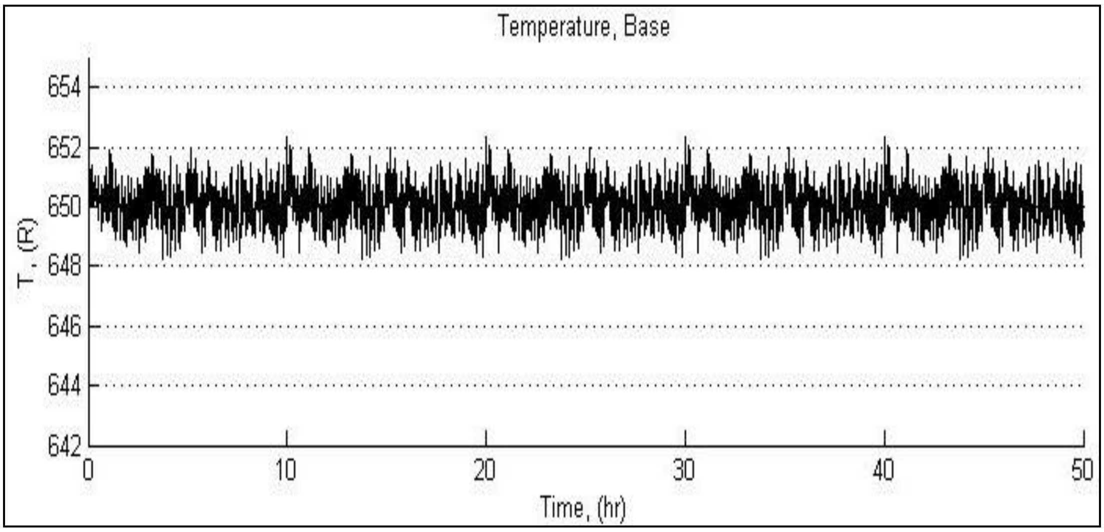


FIGURE 4.3 Product temperature base data

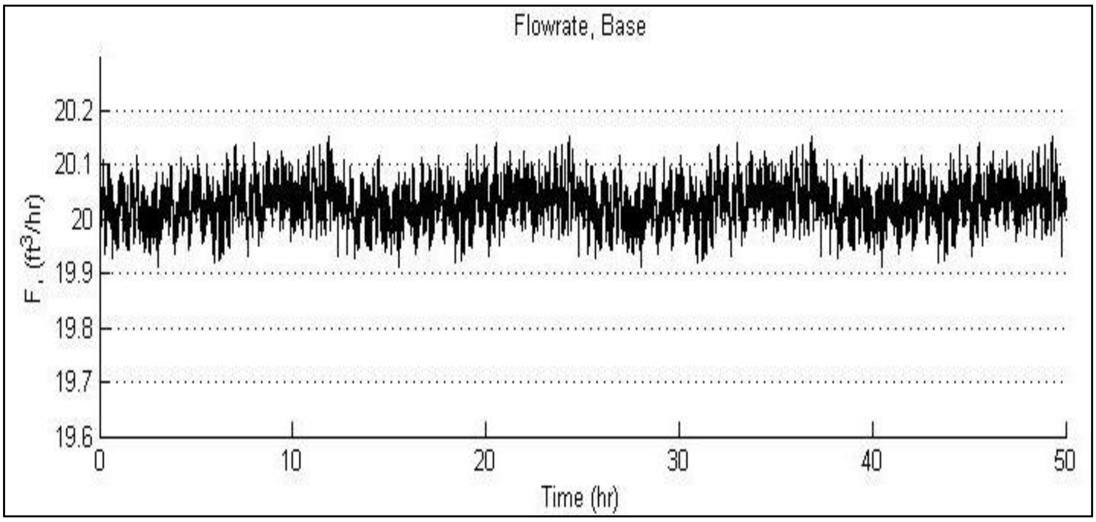


FIGURE 4.4 Product flowrate base data

4.3 FAULT DATA

4.3.1 Drift in Reaction Kinetics

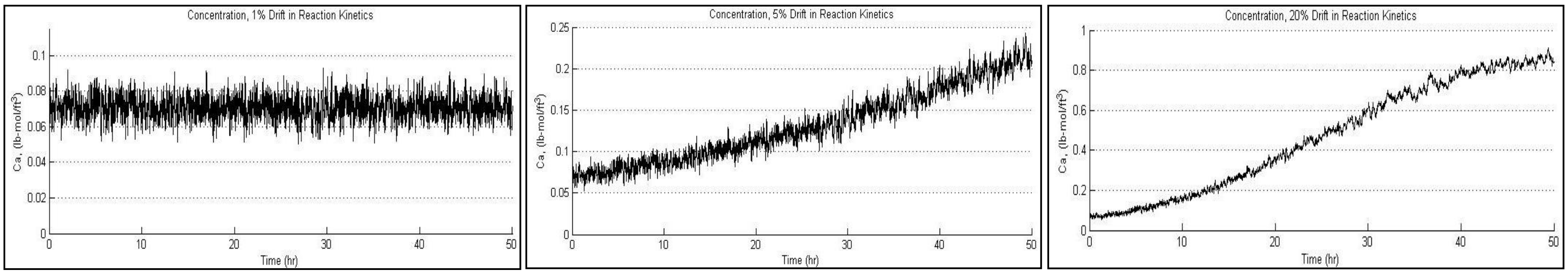


FIGURE 4.5 Product concentration fault data

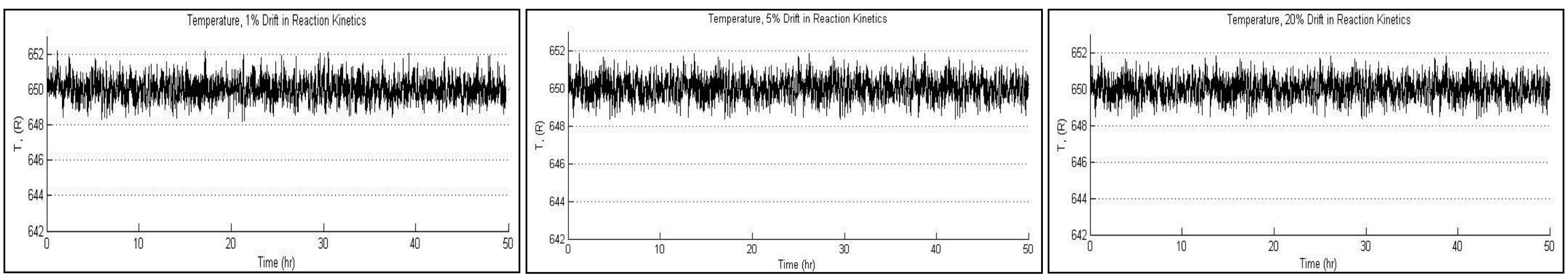


FIGURE 4.6 Product temperature fault data

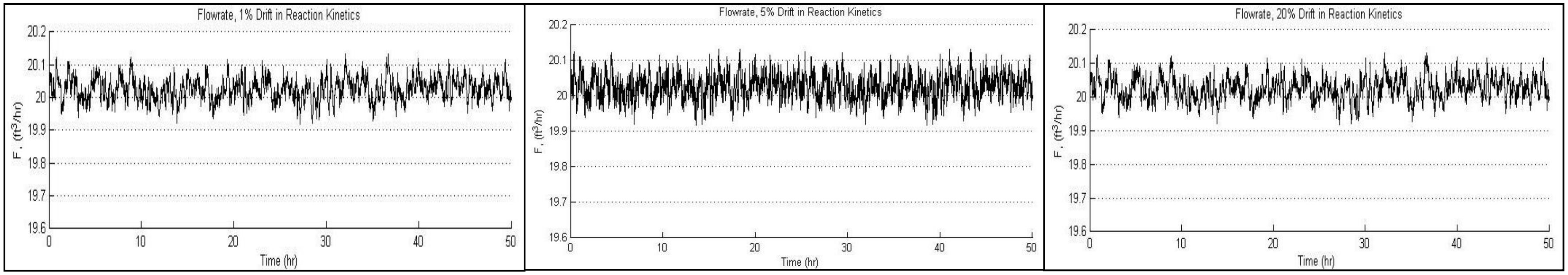


FIGURE 4.7 Product flowrate fault data

As can be seen in the fault data for drift in reaction kinetics, the variation among flowrate and temperature for different fault levels are almost non-existent but stark differences can be seen among concentration data. This could be due to the fact that at higher activation energies, reaction would not proceed fast and thereby consuming lesser reactants.

4.3.2 Drift in Heat Transfer Coefficient

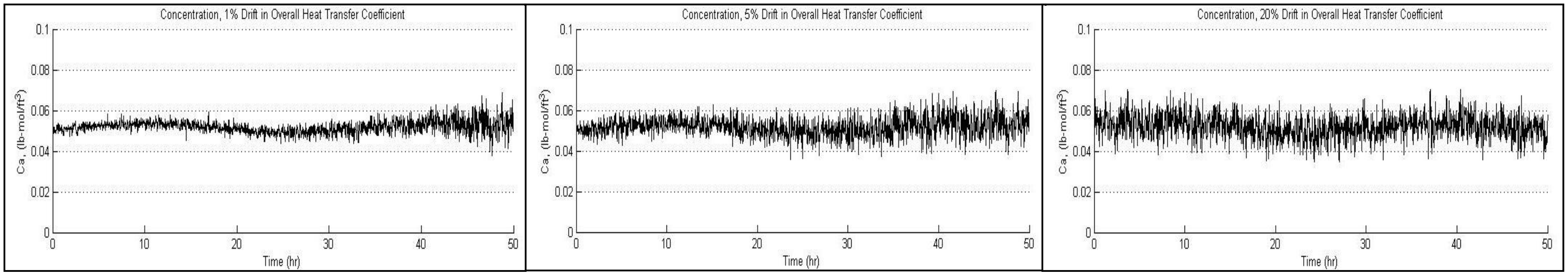


FIGURE 4.8 Product concentration fault data

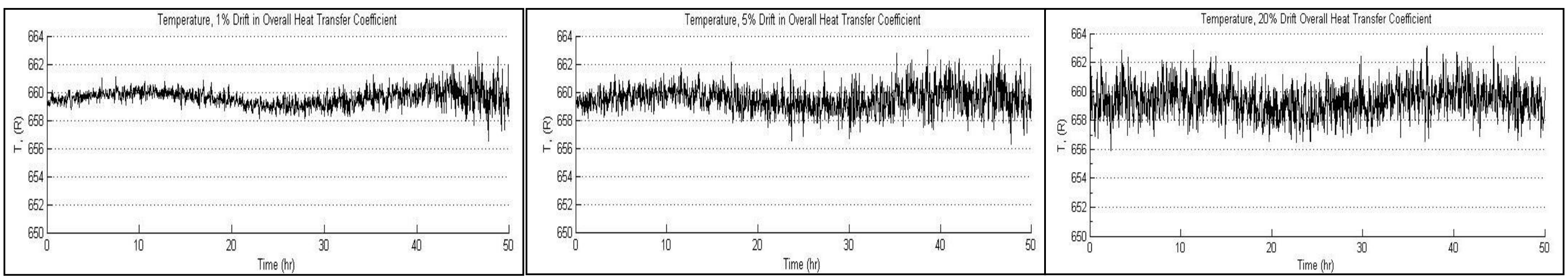


FIGURE 4.9 Product temperature fault data

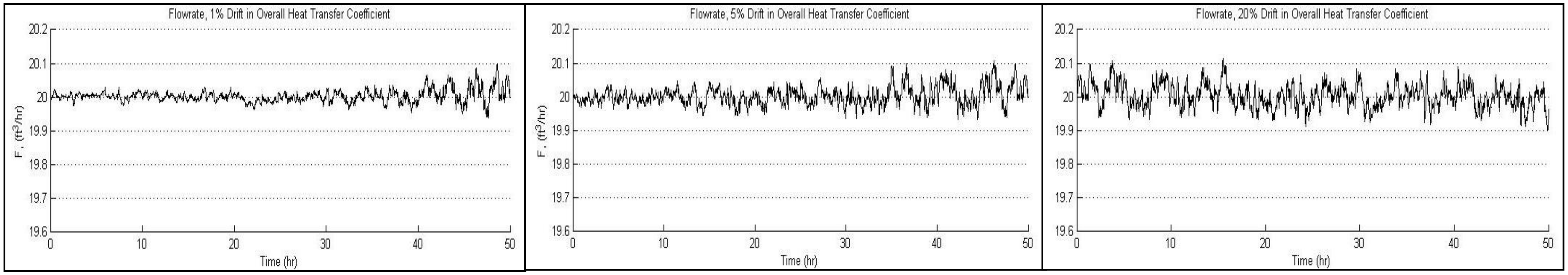


FIGURE 4.10 Product flowrate fault data

As for fault data of drift in overall heat transfer coefficients, it can be observed that drift for different fault levels are present in all three variables of product flowrate, concentration and temperature. This could be due to the fact that drifts in overall heat transfer coefficients capable of affecting all three variables. For example, temperature increased reduces the rate of reaction (as the reaction is not in favour of high temperature). Thus, lesser reactants consumed resulting in overflow of unreacted reactant in the CSTR system.

4.3.3 Drift in Heat Transfer Coefficient & Drift in Reaction Kinetics)

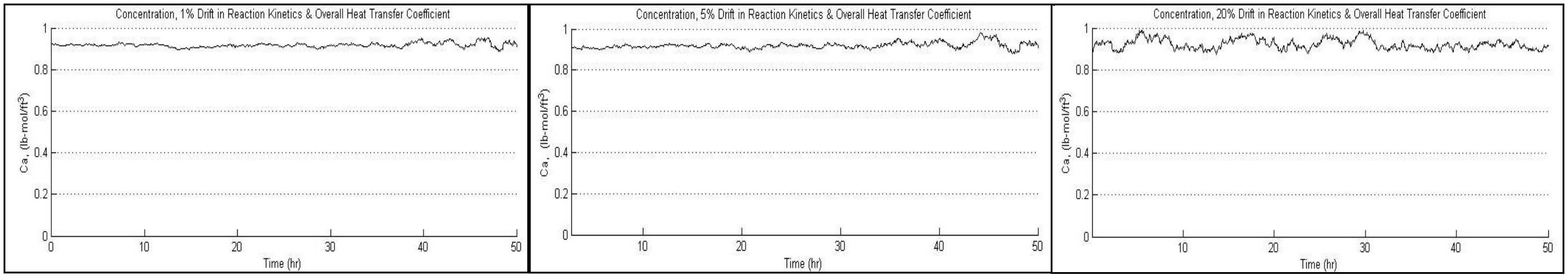


FIGURE 4.11 Product concentration fault data

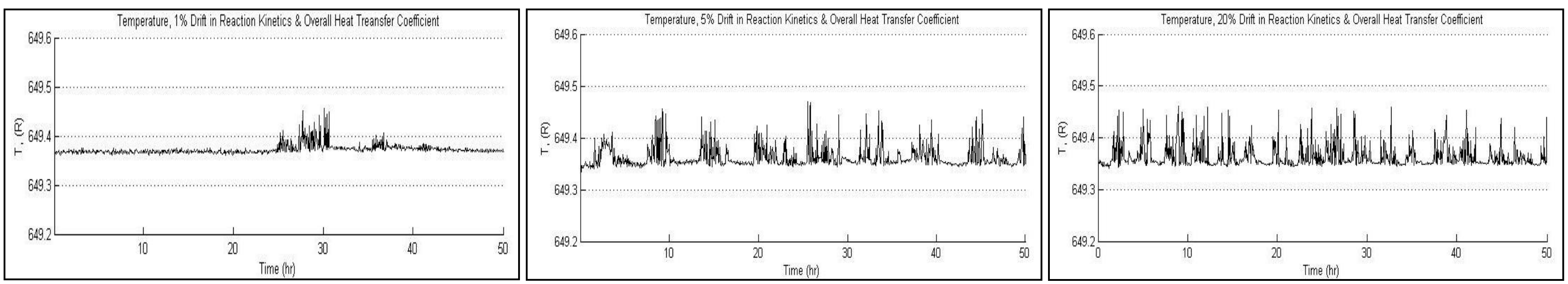


FIGURE 4.12 Product temperature fault data

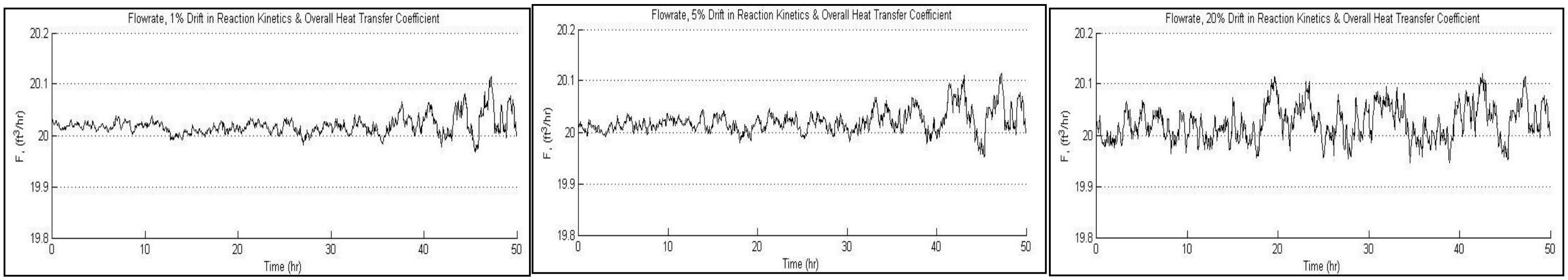


FIGURE 4.13 Product flowrate fault data

Simultaneous simulation of structural fault; drift in overall heat transfer coefficients and reaction kinetics shows stark differences in all three variables of product flowrate, concentration and temperature. This could probably because simultaneous structural fault occurence can affect or upset a system more severely compared to the effect of only one structural fault occurrence

4.4 STATISTICAL ANALYSIS

4.4.1 PCA and DPCA

The PCA is done for normalized base data to obtain the loading matrix, score matrix and latent. PCA for the fault data is then done by using loading matrix of base data and T^2 calculated using latent (which store variance) of the base data. For DPCA, the time lag shift is introduced to the input/output variables matrix. The chosen time lag shift was 2 sample lag.. Two PCs is retained for PCA and three PCs is retained for DPCA method. Source code for simulation of PCA and DPCA in Matlab are available on appendices.

4.4.2. T^2-statistics and Q-statistics

The PCA and DPCA data are first tested using T^2-statistics. The thresholds calculated using Eq. (2.10) is shown in the table below

Parameter	PCA	DPCA
a	2	4
n	5000	5000
T_α^2 ***(95.0% confidence limit)***	6.641	7.825
T_α^2 ***(99.0% confidence limit)***	9.223	11.36

TABLE 4.1 T^2-statistics thresholds

To model using Q-statistics, the residual matrix of the data is required. This residual matrix captures the variations beyond the *a* loading vectors. The residual matrix for PCA and DPCA is obtained from Eq.(2.10). The threshold for Q-statistics is obtained from Eq.(2.12). Source code for simulation of T^2-statistic and Q-statistic in Matlab are available on appendices

T^2-statistics (Drift in Reaction Kinetics)

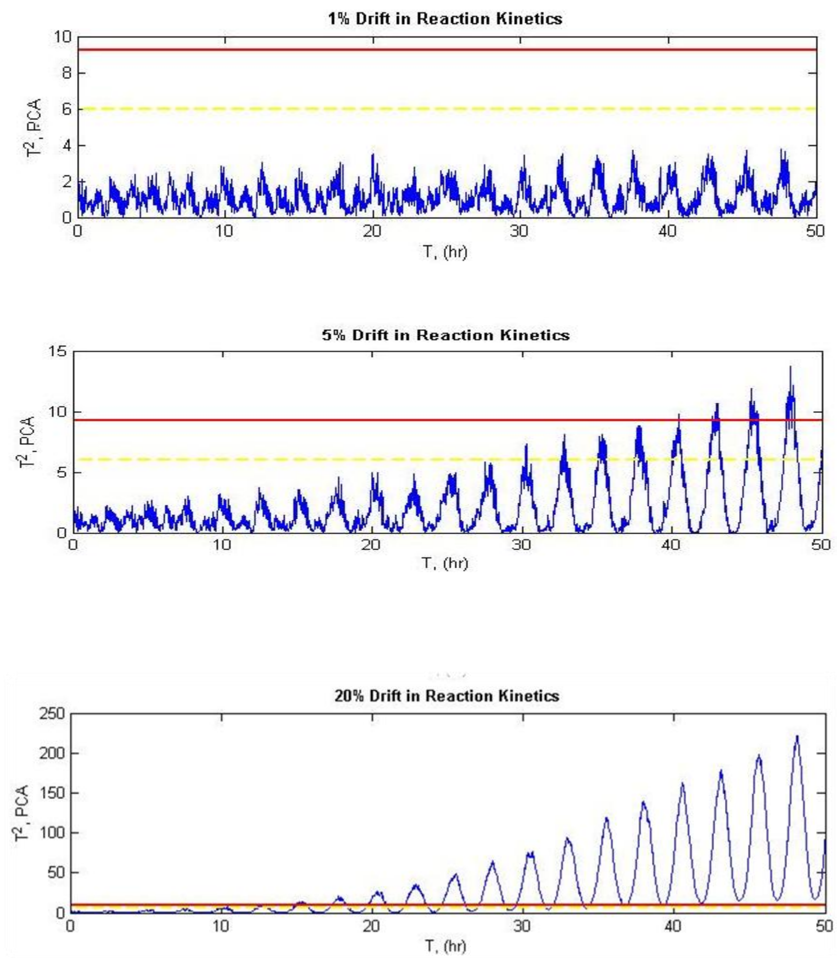


FIGURE 4.14 PCA T^2-statistics

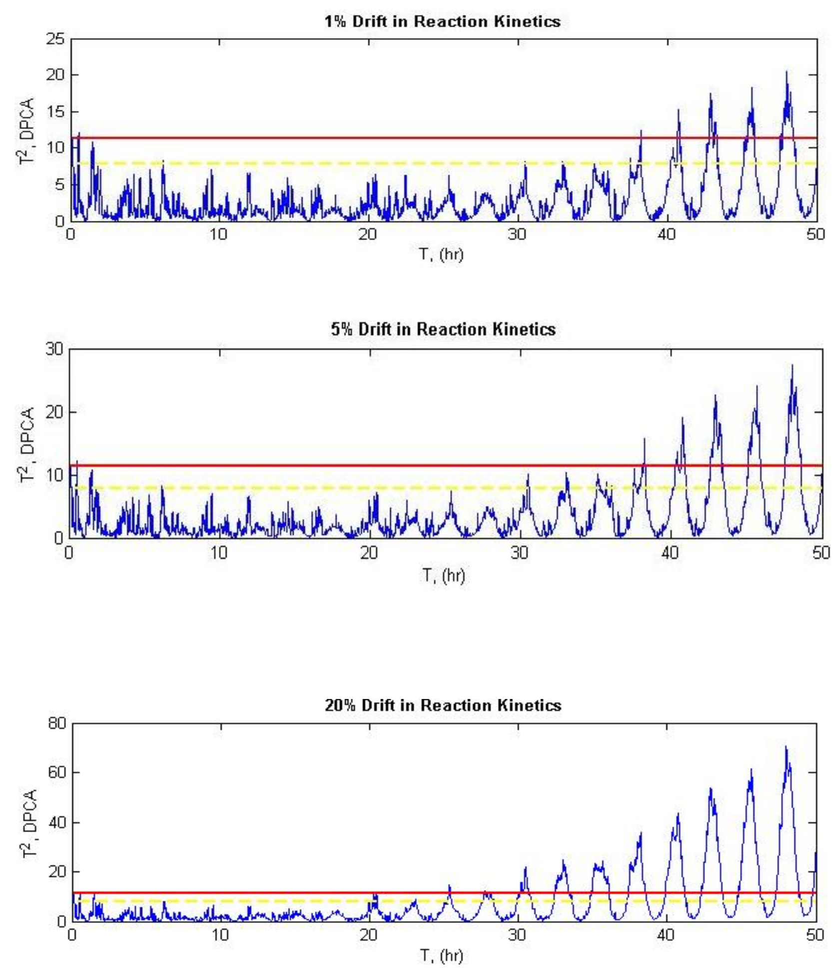


FIGURE 4.15 DPCA T^2-statistics

T^2-statistics (Drift in Heat Transfer Coefficient)

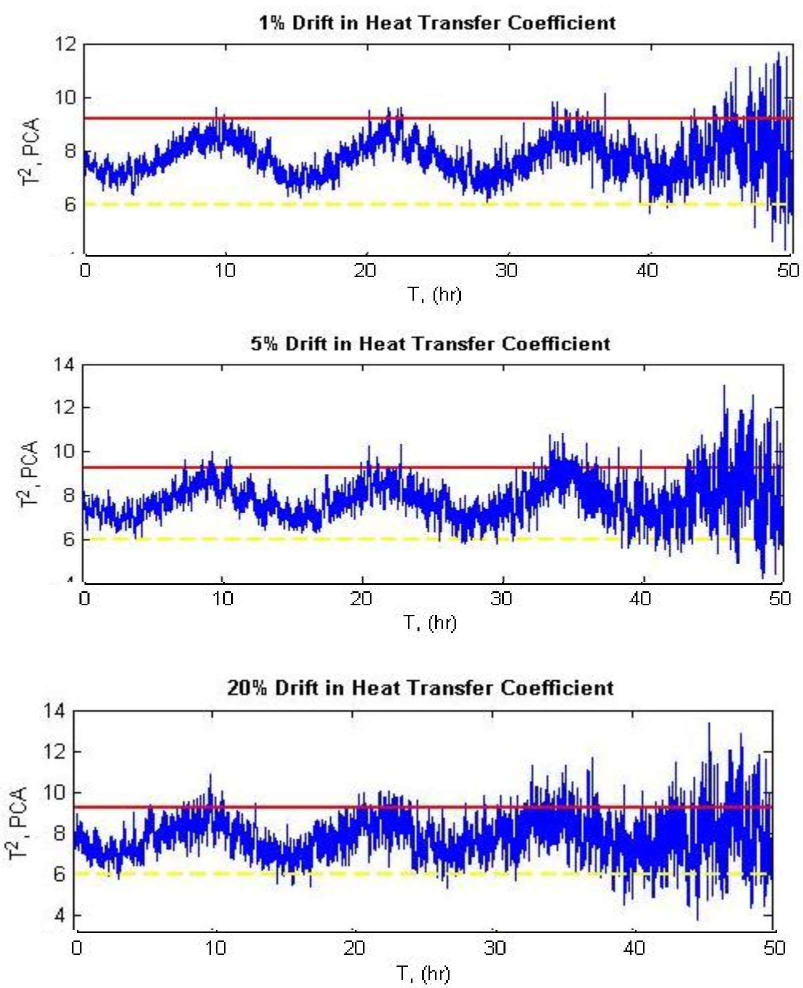


FIGURE 4.16 PCA T^2-statistics

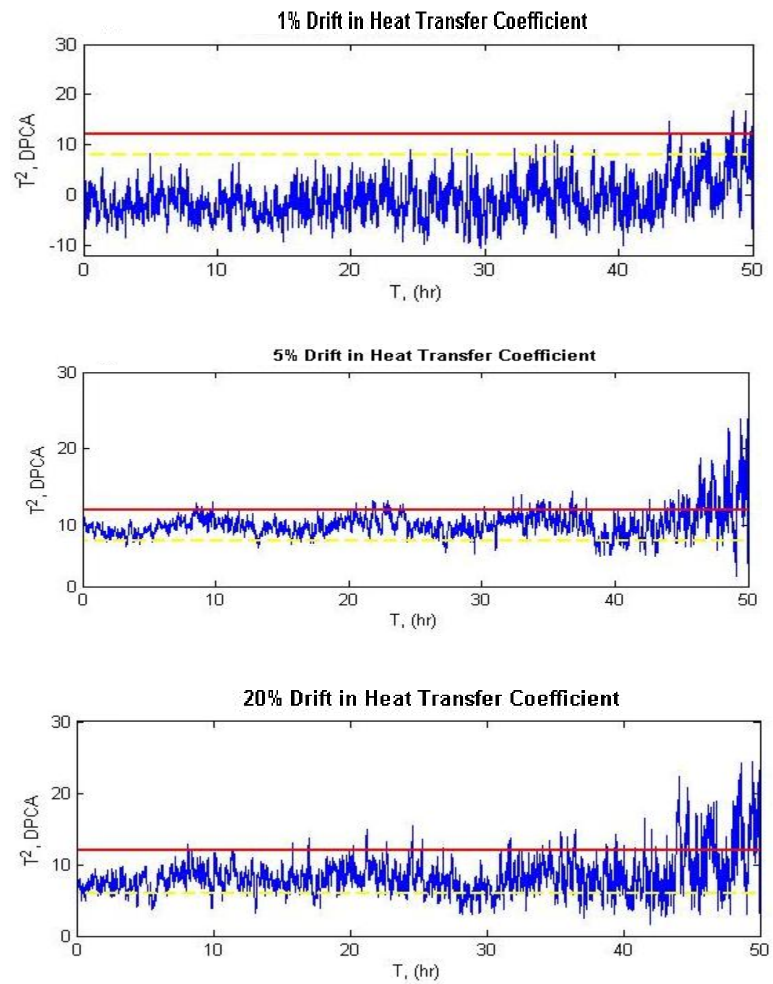


FIGURE 4.17 DPCA T^2-statistics

T^2-statistics (Drift in Heat Transfer Coefficient & Drift in Reaction Kinetics)

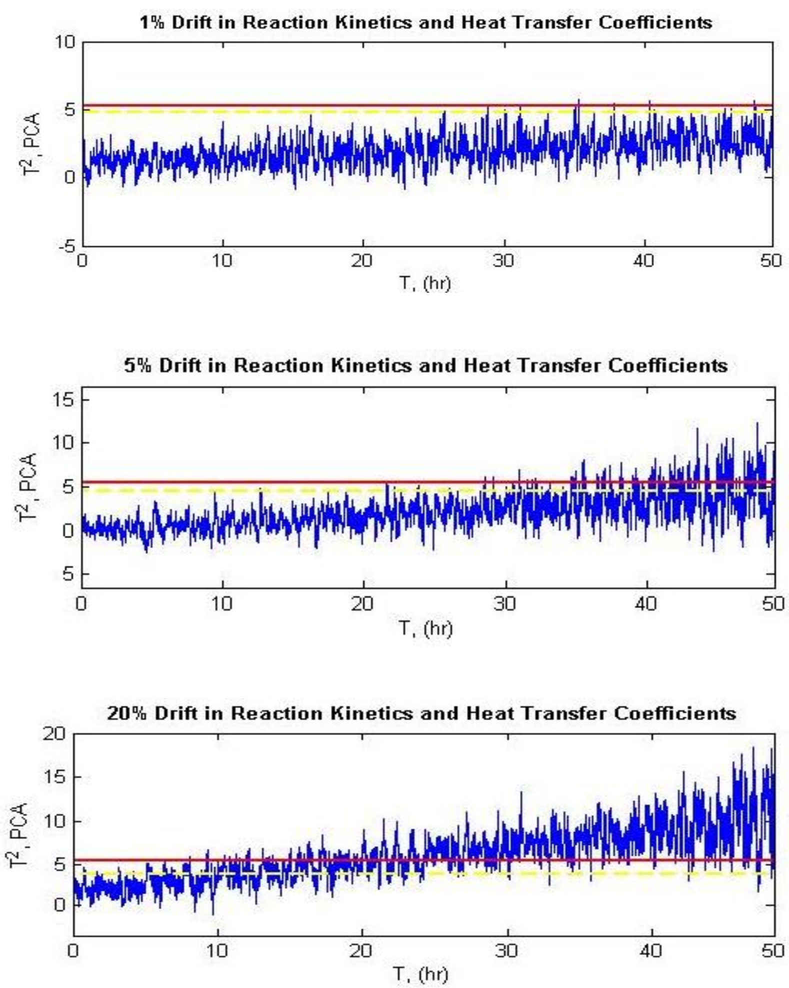


FIGURE 4.18 PCA T^2-statistics

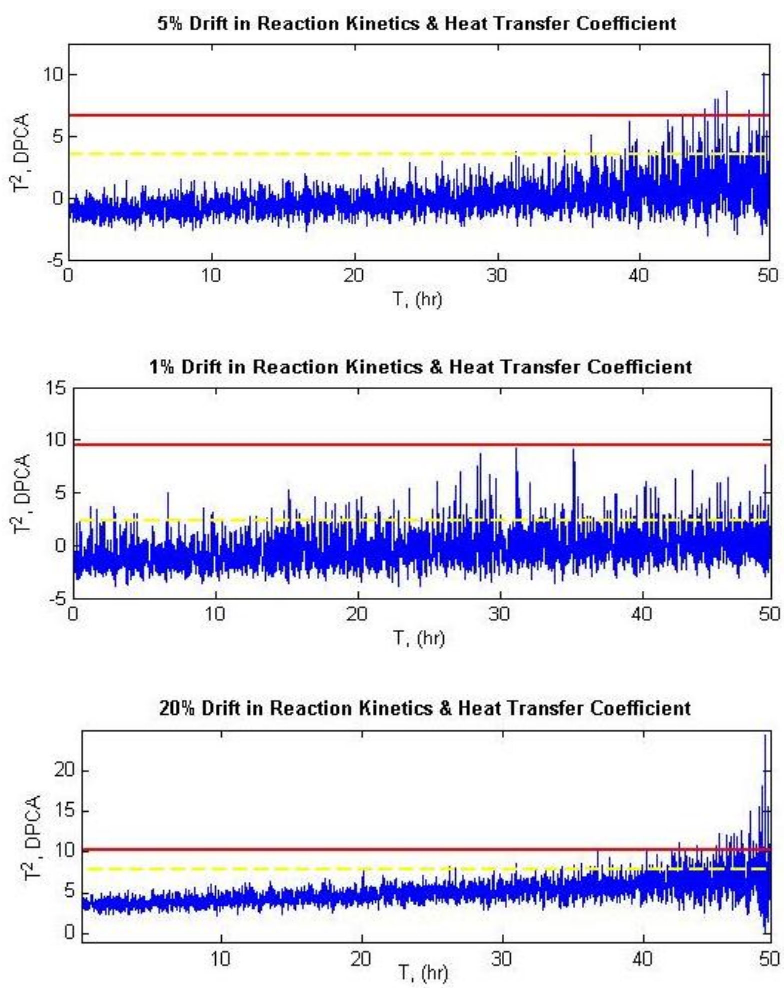


FIGURE 4.19 DPCA T^2-statistics

Q-statistics (Drift in Reaction Kinetics)

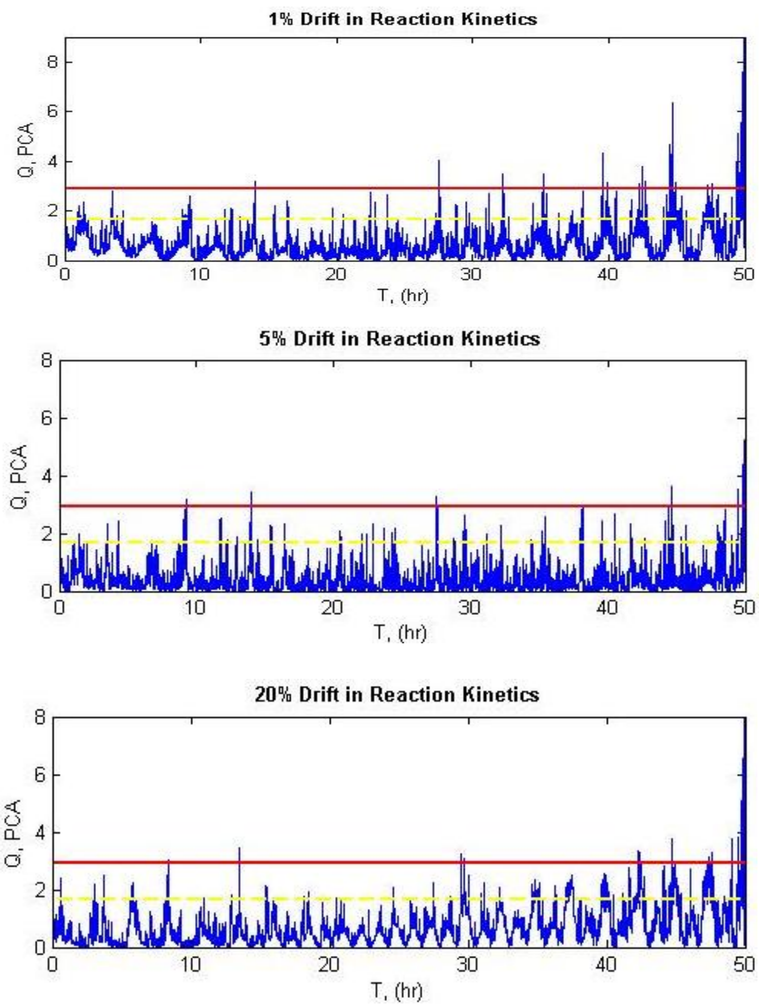


FIGURE 4.20 PCA Q-statistics

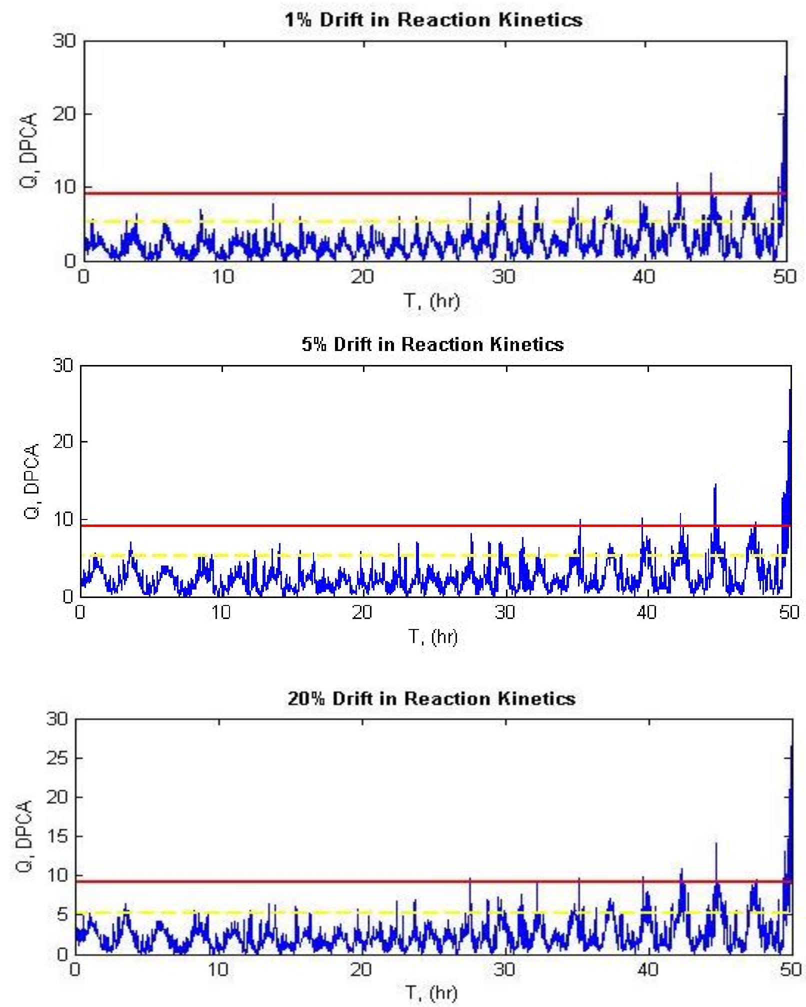


FIGURE 4.21 DPCA Q-statistics

Q-statistics (Drift in Heat Transfer Coefficient)

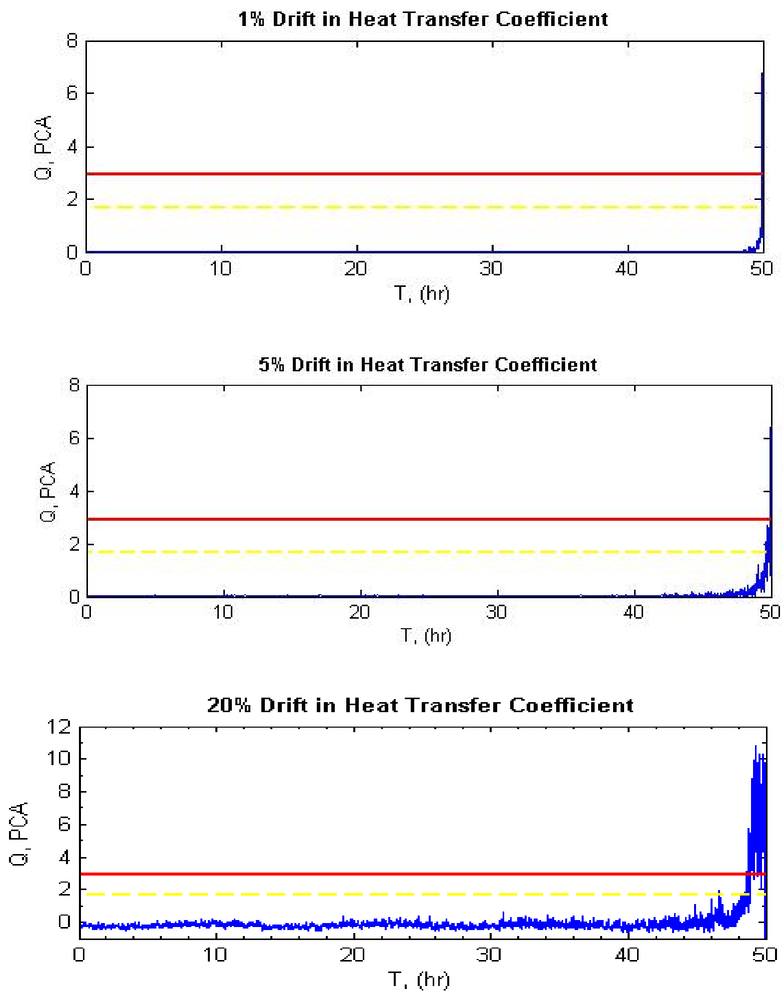


FIGURE 4.22 PCA Q-statistics

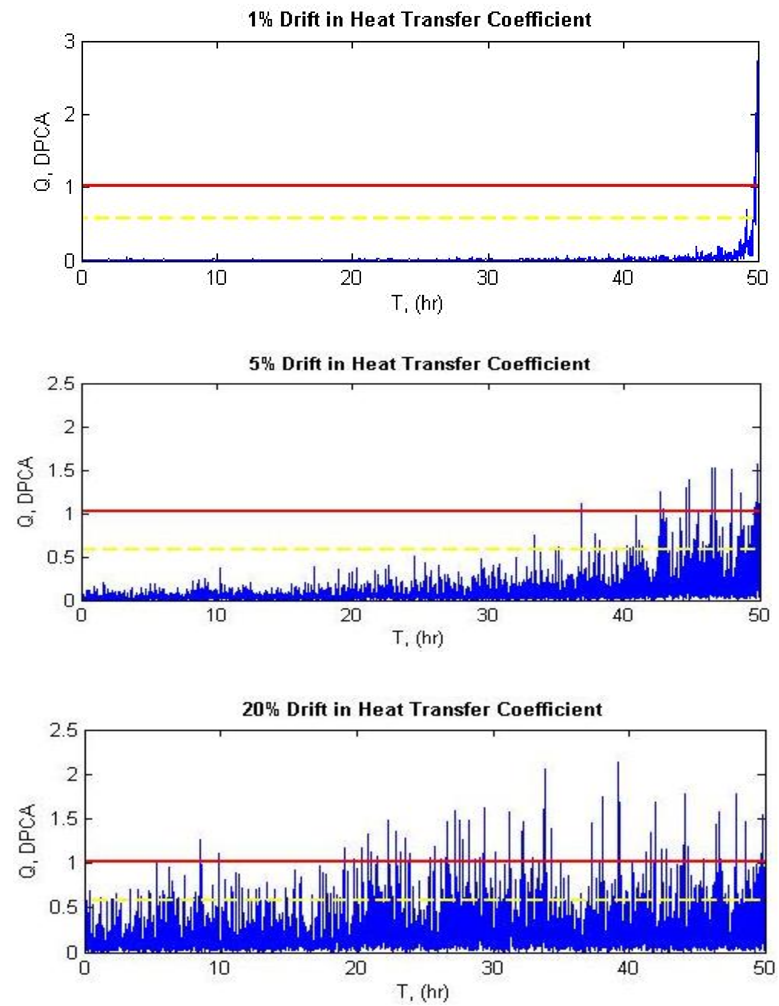


FIGURE 4.23 DPCA Q-statistics

Q-statistics (Drift in Heat Transfer Coefficient & Drift in Reaction Kinetics)

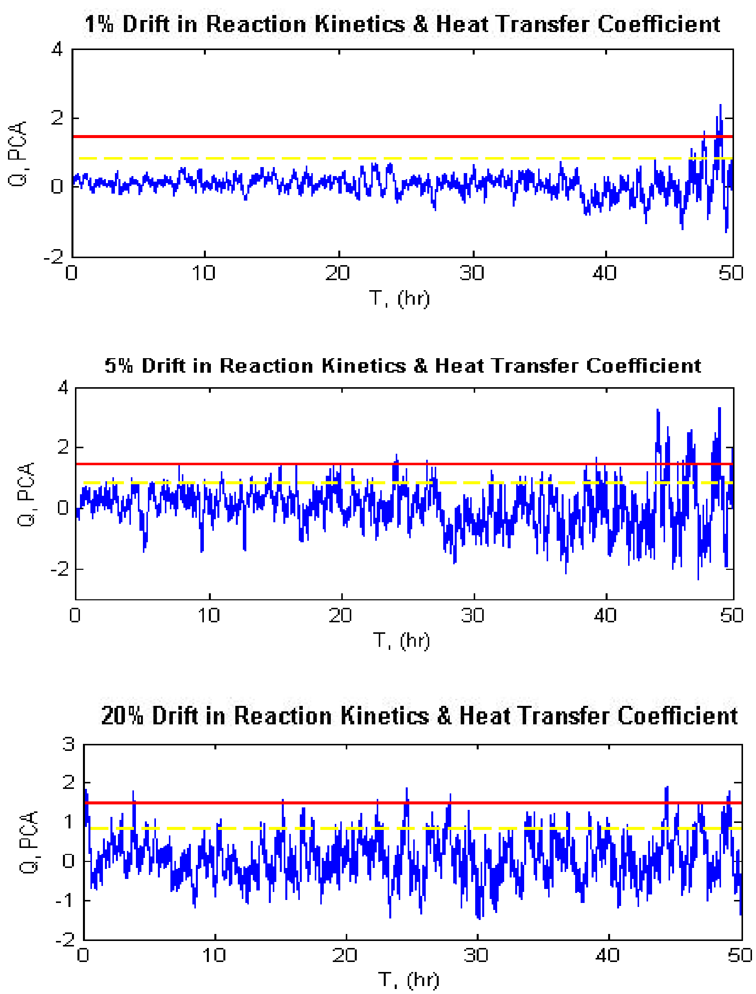


FIGURE 4.24 PCA Q-statistics

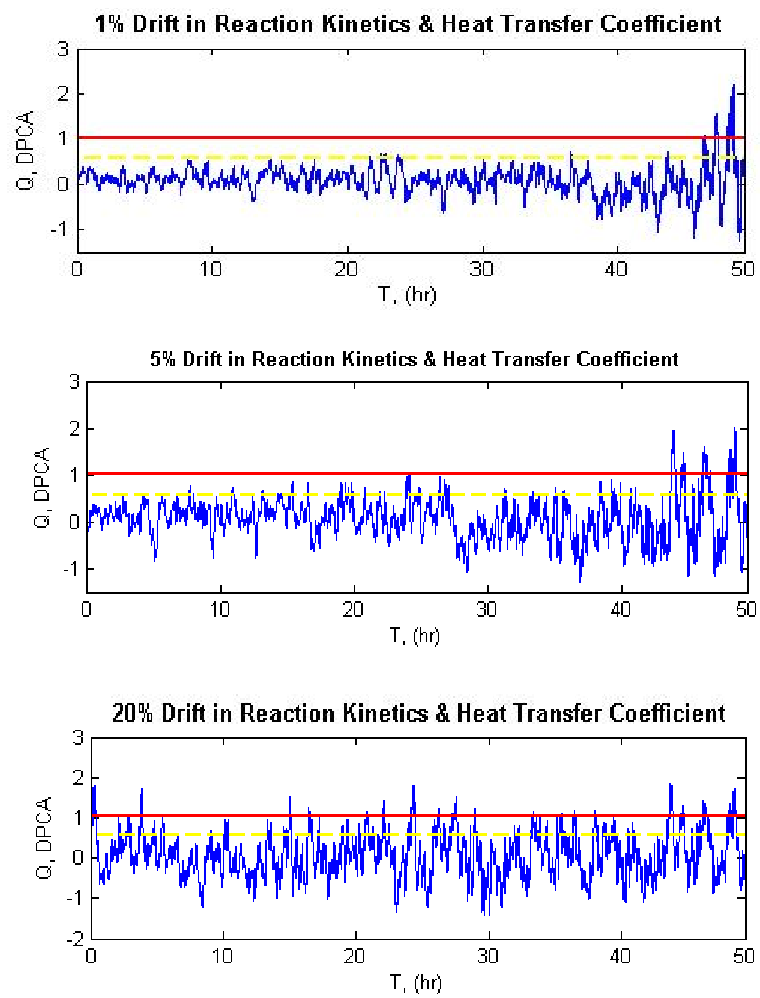


FIGURE 4.25 DPCA Q-statistics

4.5 SUMMARY OF FAULT DETECTION TIME

The summary of the fault detection time for the three types of simulated structural faults using Hotelling's T^2-statistic are in Table 4.2, 4.3 , and 4.4 while the fault detection time for the three types of simulated structural faults using SPE's Q-statistic are in Table 4.5, 4.6, and 4.7.

TABLE 4.2 Detection time for Drift in Reaction Kinetics; (Hotelling's T^2-statistic)

Case	**PCA**		**DPCA**	
	95.0% confidence limit	**99.0% confidence limit**	**95.0% confidence limit**	**99.0% confidence limit**
1%	0	0	0	0
5%	30.20	40.50	0	0
20%	7.20	13.10	0	0

TABLE 4.3 Detection time for Drift in Heat Transfer Coefficients; (T^2-statistic)

Case	**PCA**		**DPCA**	
	95.0% confidence limit	**99.0% confidence limit**	**95.0% confidence limit**	**99.0% confidence limit**
1%	0	9.50	4.9	44.20
5%	0	7.21	0	9.10
20%	0	5.09	0	8.00

TABLE 4.4 Detection time for Simultaneous Simulation of (Drift in Reaction Kinetics & Heat Transfer Coefficients); (Hotelling's T^2-statistic)

Case	**PCA**		**DPCA**	
	95.0% confidence limit	**99.0% confidence limit**	**95.0% confidence limit**	**99.0% confidence limit**
1%	25.5	28.0	1.5	0
5%	8.9	21.9	31.5	44.0
20%	0.9	4.92	20.0	36.5

TABLE 4.5 Detection time for Drift in Reaction Kinetics; (SPE's Q -statistic)

Case	**PCA**		**DPCA**	
	95.0% confidence limit	**99.0% confidence limit**	**95.0% confidence limit**	**99.0% confidence limit**
1%	0.9	13.0	0.3	43.2
5%	1.0	9.5	0.34	36.1
20%	0.5	8.1	0.3	27.9

TABLE 4.6 Detection time for Drift in Heat Transfer Coefficients; (SPE's Q -statistic)

Case	**PCA**		**DPCA**	
	95.0% confidence limit	**99.0% confidence limit**	**95.0% confidence limit**	**99.0% confidence limit**
1%	49.89	49.9	49.0	49.9
5%	48.0	49.9	34.1	36.9
20%	46.5	48.5	0.1	0.56

TABLE 4.7 Detection time for Simultaneous Simulation of (Drift in Reaction Kinetics & Heat Transfer Coefficients); (SPE's Q -statistic)

Case	**PCA**		**DPCA**	
	95.0% confidence limit	**99.0% confidence limit**	**95.0% confidence limit**	**99.0% confidence limit**
1%	48.0	48.9	8.1	48.2
5%	3.5	7.65	2.0	24.5
20%	0	0	0	0

Figure 4.14 shows the result of statistical analysis done on structural fault data (Drift in Reaction Kinetics); by using PCA method with Hotelling's T^2-statistics. The 95% and 99% threshold are obtained from T^2-statistics analysis on the data. The result of PCA analysis on the 1% Drift in Reaction Kinetics data shows that no fault is detected. This probably due to small drifts value or could probably because PCA method is incapable or not sensitive enough of detecting the fault occurrences. In contrary, fault is detected by PCA analysis on 5% and 20% Drift in Reaction Kinetics data, and the fault detection time are summarized in Table 4. Fault is detected earlier on 20% Drift in Reaction Kinetics compared 5% drift, probably because larger value of fault is easier to be detected.

Figure 4.15 shows the result of statistical analysis done on structural fault data (Drift in Reaction Kinetics); by using DPCA method with Hotelling's T^2-statistics. The 95% and 99% threshold are obtained from T^2-statistics analysis on the data. The result of DPCA analysis on 1%, 5% and 20% Drift in Reaction Kinetics data shows that all fault are detected at time zero. This probably because DPCA method has better capability of detecting this faults occurrence compared to PCA method. The fault detection time are summarized in Table 4.

Figure 4.16 shows the result of statistical analysis done on structural fault data (Drift in Heat Transfer Coefficients); by using PCA method with Hotelling's T^2-statistics. The 95% and 99% threshold are obtained from T^2-statistics analysis on the data. The result of PCA analysis on 1%, 5% and 20% Drift in Heat Transfer Coefficients data shows that all fault are detected (by 95% thresholds) at time zero. As for 99% threshold, the earliest fault detection time is recorded on 20% drift, followed by 5% drift and 1% drift. The fault detection time are summarized in Table 5.

Figure 4.17 shows the result of statistical analysis done on structural fault data (Drift in Heat Transfer Coefficients); by using DPCA method with Hotelling's T^2-statistics. The 95% and 99% threshold are obtained from T^2-statistics analysis on the data. The result of DPCA analysis on Drift in Heat Transfer Coefficients data shows that all fault are detected (by 95% thresholds) at time zero except for 1% fault drift. As for 99% threshold, the earliest fault detected is on 20% drift, followed by 5% drift and 1% drift fault. The fault detection time are summarized in Table 5.

Figure 4.18 shows the result of statistical analysis done on structural fault data (Drift in Reaction Kinetics & Heat Transfer Coefficients); by using PCA method with Hotelling's T^2-statistics. The 95% and 99% threshold are obtained from T^2-statistics analysis on the data. The results of PCA analysis on all the drift level data shows that fault are detected and the fault detection times are summarized in Table 6.

Figure 4.19 shows the result of statistical analysis done on structural fault data (Drift in Reaction Kinetics & Heat Transfer Coefficients); by using DPCA method with Hotelling's T^2-statistics. The 95% and 99% threshold are obtained from T^2-statistics analysis on the data. The results of DPCA analysis on all the drift level data shows that fault are detected and the fault detection times are summarized in Table 6.

Figure 4.20 shows the result of statistical analysis done on structural fault data (Drift in Reaction Kinetics); by using PCA method with SPE's Q-statistics. The 95% and 99% threshold are obtained from Q-statistics analysis on the data. The results of PCA analysis on all the drift level data shows that fault are detected and the fault detection times are summarized in Table 7.

Figure 4.21 shows the result of statistical analysis done on structural fault data (Drift in Reaction Kinetics); by using DPCA method with SPE's Q-statistics. The 95% and 99% threshold are obtained from Q-statistics analysis on the data. The results of DPCA analysis on all the drift level data shows that fault are detected and the fault detection times are summarized in Table 7.

Figure 4.22 shows the result of statistical analysis done on structural fault data (Drift in Heat Transfer Coefficients); by using PCA method with SPE's Q-statistics. The 95% and 99% threshold are obtained from Q-statistics analysis on the data. The results of PCA analysis on all the drift level data shows that fault are detected and the fault detection times are summarized in Table 8.

Figure 4.23 shows the result of statistical analysis done on structural fault data (Drift in Heat Transfer Coefficients); by using DPCA method with SPE's Q-statistics. The 95% and 99% threshold are obtained from Q-statistics analysis on the data. The results of DPCA analysis on all the drift level data shows that fault are detected and the fault detection times are summarized in Table 8.

Figure 4.24 shows the result of statistical analysis done on structural fault data (Drift in Reaction Kinetics & Heat Transfer Coefficients); by using PCA method with SPE's Q-statistics. The 95% and 99% threshold are obtained from Q-statistics analysis on the data. The results of PCA analysis on all the drift level data shows that fault are detected and the fault detection times are summarized in Table 9.

Figure 4.25 shows the result of statistical analysis done on structural fault data (Drift in Reaction Kinetics & Heat Transfer Coefficients); by using DPCA method with SPE's Q-statistics. The 95% and 99% threshold are obtained from Q-statistics analysis on the data. The results of DPCA analysis on all the drift level data shows that fault are detected and the fault detection times are summarized in Table 9.

CHAPTER 5
CONCLUSION AND RECOMMENDATION

The objective of this work is to investigate structural fault detection performance of DPCA method with comparison to PCA. The significant of the study is to fill the gap of knowledge in fault detection specifically on structural fault detection. The structural change in CSTR model was successfully simulated using Simulink in MATLAB and the data obtained was used as feeding data to PCA based monitoring approaches i.e. PCA, and DPCA.

Based on the result obtained, in general DPCA shows a better performance by detecting faults earlier compared to PCA. Therefore, it can be concluded that structural fault can be detected using Multivariate statistical Process Monitoring based methods i.e. PCA, DPCA and so on. Thus it is too can be concluded that the objectives of the project are successfully achieved.

Nevertheless, the suggested work for future is as below:

1. Continuing this research work by doing more types of structural fault simulation.
2. Replacing CSTR simulation model with a more complex system/process to increase the similarities of this research work to the real operating process plant.
3. The continuation of the structural fault detection using other MSPM techniques and proceeded with fault identification and fault diagnosis.

REFERENCES

Kourti, T., & MacGregor, J. F. (1995). Process analysis, monitoring and diagnosis, using multivariate projection methods. *CHEMOM</cja:jid> Chemometrics and Intelligent Laboratory Systems, 28*(1), 3-21.

MacGregor, J., & Cinar, A. (2012). Monitoring, fault diagnosis, fault-tolerant control and optimization: Data driven methods. CACE Computers and Chemical Engineering, 47, 111-120.

Villegas, T., Fuente, M. J., Rodriguez, M., th Wseas International Conference on Computational Intelligence, M.-M. S., & Cybernetics, C. (2010). Principalcomponent analysis for fault detection and diagnosis. Experience with a pilot plant. Int. Conf. Comput. Intell. Man-Mach. Syst. Cybern. Proc. International Conference on Computational Intelligence, Man-Machine Systems and Cybernetics - Proceedings, 147-152.

Zhang, Y., Zhang, L., & Zhang, H. (2012). Fault Detection for Industrial Processes. Mathematical Problems in Engineering Mathematical Problems in Engineering, 2012 (8-9), 1-18.

Venkatasubramanian, V., Rengaswamy, R., Yin, K., & Kavuri, S. N. (2003). A review of process fault detection and diagnosis:Part I: Quantitative model-based methods. *CACE</cja:jid> Computers and Chemical Engineering*, *27*(3), 293-311.

Himmelblau, D. M. (1978). Fault detection and diagnosis in chemical and petrochemical processes. Amsterdam: Elsevier press.

Chiang, L. H., Braatz, R. D., & Russell, E. (2001). Fault detection and diagnosis in industrial systems. London; New York: Springer.

Russell, E. L., Chiang, L. H., & Braatz, R. D. (2000). Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis. Chemometrics and Intelligent Laboratory Systems Chemometrics and Intelligent Laboratory Systems, 51(1), 81-93

J., M., & C, V. (2005). Fault Detection Using Dynamic Principal Component Analysis by Average Estimation. 2nd International Conference on Electrical and Electronics Engineering (ICEEE) and XI Conference on Electrical Engineering (CIE 2005). Mexico City, Mexico.

Hotelling, H., (1947), "Multivariate Quality Control Illustrated by Air Testing of Sample Bombsights", C.Eisenhart et. al. Pp.111-184

Jackson JE. (1991). A User Guide to Principal Components. Wiley: New York.

Jana, A.K. (2011). Chemical Process Modelling and Computer Simulation (2nd ed.) Delhi: PHL Learning Private Limited.

Haridy, S., & Wu, Z. (2009). Univariate and multivariate control charts for monitoring dynamic-behavior processes: a case study. from http://hdl.handle.net/2099/8485

Hotelling, H. (1947). Multivariate Quality Control: Techniques of Statistical Analysis. New York: McGraw Hill.

Blank, R. E. (1988). Multivariate X-bar and R Charting Techniques. Presented at the ASQC's Annual Quality Congress Transactions.

Morrison, D. F. (1990). Multivariate statistical Methods, 3rd Ed. New York: McGraw-Hill, Inc.

Sultan, T. I. (1986). An Acceptance Chart for Raw Materials of Two Correlated Properties. Quality Assurance, 12, 70-72

Alt, F. B. (1985). Multivariate Quality Control. Encyclopedia of Statistical Sciences, 6, 110-122.

Jackson, J. E., & Mudholkar, G. S. (1979). Control Procedures for Residuals Associated With Principal Component Analysis. Technometrics Technometrics, 21(3), 341-349.

Crowl, D. A., & Louvar, J. F. (2011). Chemical Process Safety : Fundamentals With Applications. Upper Saddle River, NJ: Prentice Hall.

J.J. Downs, E.F. Vogel, (1993). Comput. Chem. Eng. 17. 245–255.

Slišković, D., Grbić, R., & Hocenski, Ž. (2012). Multivariate Statistical Process Monitoring 9.

Martin, E. B., Morris, A. J., & Zhang, J. (1996). Process performance monitoring using multivariate statistical process control. IEE Proceedings - Control Theory and Applications, 143(2), 132-144.

Hollender, M. (2010). Collaborative process automation systems. from http://app.knovel.com/hotlink/toc/id:kpCPAS0001/collaborative-process-automation

APPENDICES

Matlab Source Code of CSTR System (for base data)

```
function [sys,x0,str,ts] = ...
   s_reactor1a (t,x,u,flag,xinit)

%------------------------------------------------------------------------
% S-function for Reactor block
%       States: x = [V, VCa, VT, Tc, Ts]
%       Input: u = [Fin, Tin, Cain, Tcin, Fsin, Fcin, F]
%       Output: y = [F, T, Ca, V, Tc, Ts, Fs]
%       Parameters: params = [Ma, rhoa, Cpa,...
%                             At, Ap, Lp, g, ...
%                             alpha, E, R, lambda,...
%                             Uc, Ac, rhoc, Cpc, Vc,...
%                             Vj, hos, Aos, Ms, Avp, Bvp, Hs_hc ...
%                             xinit]
%               xinit = [48, 11.76, 28800, 594.64, 0.245, 600, 40, 49.9, 0.0803, 719]
%------------------------------------------------------------------------
xinit=[40, 40*0.05, 40*650, 640, 650]; %V, VCa, VT, Tc, Ts

params = [ 100, 50, 0.75, ...
    10.36, 0.1076, 6.56, 0.414720000000E+09 , ... %g =  31.99846 (s)
    7.08e10 , 30000, 1.99, -30000, ...%alpha =  1.9667e+007 (s)
    120, 250, 62.3, 1, 3.85, ...%Uc = 0.0417 (s), 150 (h)
    18.83, 1000, 56.5, 18, -8744.4, 15.70, 939, 30000 ... %hos = 0.2778 (s), 1000(h)
    xinit];%Units in Btu etc

switch flag,

  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(xinit);

  %%%%%%%%%%%%%%%
  % Derivatives %
  %%%%%%%%%%%%%%%
  case 1,
    sys=mdlDerivatives(t,x,u,params);


  %%%%%%%%%%
  % Update %
  %%%%%%%%%%
  case 2,
    sys=mdlUpdate(t,x,u,params);

  %%%%%%%%%%%
  % Outputs %
  %%%%%%%%%%%
  case 3,
    sys=mdlOutputs(t,x,u,params);

  %%%%%%%%%%%%%%%%%%%%%%%
  % GetTimeOfNextVarHit %
```

```
  %%%%%%%%%%%%%%%%%%%%%%%
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u,params);

  %%%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%%
  case 9,
    sys=mdlTerminate(t,x,u,params);

  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);

end

% end sfun_sub

% =======================================================================
function [sys,x0,str,ts]=mdlInitializeSizes(xinit)

sizes = simsizes;
sizes.NumContStates  = 5; %i.e. V, VCa, VT, Tc, Ts
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 7; %i.e. F, T, Ca, V, Tc, Ts, Fs
sizes.NumInputs      = 7; %i.e. Fin, Tin, Cain, Tcin, Fsin, Fcin, F
sizes.DirFeedthrough = 7;
sizes.NumSampleTimes = 1;   % at least one sample time is needed

sys = simsizes(sizes);

%
% initialize the initial conditions
%
%statenum = sizes.NumContStates;
%pend = length(params);
%xinit = params(pend-statenum+1:pend);
x0  = xinit;

%
% str is always an empty matrix
%
str = [];

%
% initialize the array of sample times
%
ts  = [0 0];

% end mdlInitializeSizes

% =======================================================================
function sys=mdlDerivatives(t,x,u,params)

%disp('---------------------------------------------------');
t;

Ma = params(1);
rhoa = params(2);
Cpa = params(3);
```

```
At = params(4);
Ap = params(5);
Lp = params(6);
g = params(7);
alpha = params(8);
E = params(9);
R = params(10);
lambda = params(11);
Uc = params(12);
Ac = params(13);
rhoc = params(14);
Cpc = params(15);
Vc = params(16);
Vj = params(17);
hos = params(18);
Aos = params(19);
Ms = params(20);
Avp = params(21);
Bvp = params(22);
Hs_hc = params(23);% btu/lbm, i.e. = 522.4877cal/g;

rho = rhoa;
Cp = Cpa;

%Inputs
%i.e. Fin, Tin, Cain, Tcin, Fsin, Fcin, F
Fin = u(1);
Tin = u(2);
Cain = u(3);
Tcin = u(4);
Fsin = u(5);
Fcin = u(6);
Fout = u(7);

%States
%i.e. V, VCa, VT, Tc, Ts
V = x(1);
VCa = x(2);
VT = x(3);
Tc = x(4);
Ts = x(5);

Ca = VCa/V;
if Ca < 1e-30, Ca = 0; end
T = VT/V;

%Start: Calculate derivative for Ts (Version 2)----------------------------
%Xs = 0.5; %open loop, fixed steam inlet valve
Pj = exp(Avp/Ts+Bvp);
rhos = Ms*Pj/(R*Ts);
drhos_over_dTs = Ms/R * (-1-Avp/Ts) * Pj /Ts/Ts;
%ws = Fsin * rhos;
if Pj<35,
    ws = 112*Fsin*sqrt(35-Pj);
else
    ws=0;
end
Fs = ws/rhos;
Qj = -hos*Aos*(Ts-T);
wc = -Qj/Hs_hc; %steam condensate outlet
%dTs = (ws-wc) / Vj / drhos_over_dTs;
dTs = 0 ;% Set dTs always zero for this case
```

```
%end--------------------------------------------------------------------------

%Derivatives
F = Fout;

dV = Fin - F;
dVCa = Fin*Cain - F*Ca - V*alpha*exp(-E/(R*T))*Ca;
dVT = Fin*Tin - F*T - lambda*V*alpha*exp(-E/(R*T))*Ca / (rho*Cp) - Uc*Ac/(rho*Cp)*(T-Tc)
- Qj/(rho*Cp);
dTc = Fcin*(Tcin-Tc)/Vc + Uc*Ac/(rhoc*Vc*Cpc)*(T-Tc);

%Start: Gravity Flow --------------------------------------------------
%gc = g;
%Dt = sqrt(At*4/pi); %tank diameter
%ff = 3.8488e+003;%12.6239; %friction factor
%Kf = Ap*rho*2*ff/gc/Dt;
%h = V/At;
%dF = g*h/Lp*Ap - Kf*gc*F*F/(Ap*rho)/Ap; %dv = 0.0107*h - 0.00205*v*v;
%End----------------------------------------------------------------------

%States: V, VCa, VT, Tc, Ts
sys(1) = dV;
sys(2) = dVCa;
sys(3) = dVT;
sys(4) = dTc;
sys(5) = dTs;

% end mdlDerivatives

% =======================================================================
function sys=mdlUpdate(t,x,u,params)

sys = [];

% end mdlUpdate

% =======================================================================
function sys=mdlOutputs(t,x,u,params)

%States
%i.e. V, VCa, VT, Tc, Ts
%disp('mdlOutputs --------------------------')
V = x(1);
VCa = x(2);
VT = x(3);
Tc = x(4);
Ts = x(5);

Ca = VCa/V;
if Ca < 1e-30, Ca = 0; end
T = VT/V;

%Inputs
%i.e. Fin, Tin, Cain, Tcin, Fsin, Fcin, F
F = u(7);

Ms = params(20);
Avp = params(21);
Bvp = params(22);
R = params(10);
Pj = exp(Avp/Ts+Bvp);
Fsin = u(5);
```

```
rhos = Ms*Pj/(R*Ts);
if Pj<35,
    ws = 112*Fsin*sqrt(35-Pj);%Fsin = extent of valve opening
else
    ws=0;
end
Fs = ws;%/rhos;

%Output
%i.e. F, T, Ca, V, Tc, Ts, Fs, alphaout
sys(1) = F;
sys(2) = T;
sys(3) = Ca;
sys(4) = V;
sys(5) = Tc;
sys(6) = Ts;
sys(7) = Fs;
% end mdlOutputs

% ======================================================================
function sys=mdlGetTimeOfNextVarHit(t,x,u,params)

sys = [];

% end mdlGetTimeOfNextVarHit

% ======================================================================
function sys=mdlTerminate(t,x,u,params)

sys = [];

% end mdlTerminate

%F = 40 - 10*(48-V);%control
%Fc = 49.9 - 4*(600-T);%control
%%Fc = Fc_prev;

%Start: control for steam inlet valve-------------------------
%Ptt = 3+(T-510)*12/200;%control for steam inlet valve
%Pc = 7+2*(12.6-Ptt);%control for steam inlet valve
%Xs = (Pc-9)/6;%control for steam inlet valve
%if Xs>1, Xs=1; end%control for steam inlet valve
%if Xs<0, Xs=0; end%control for steam inlet valve
%end----------------------------------------------------------

%Start: Calculate derivative for Ts (Version 1)----------------------------
%Xs = 0.5; %open loop, fixed steam inlet valve
%Pj = exp(Avp/Ts+Bvp);
%if Pjin < Pj, Xs=0;end %
%ws = 112*Xs*sqrt(Pjin-Pj);
%Qj = -hos*Aos*(Ts-T);
%wc = -Qj/939; %steam condensate outlet

%drhos = (ws-wc)/Vj;
%rhos_new = rhos + 0.002*drhos;%assume step size of simulation 0.002
%PjTj = fsolve(@steam,[Pj Ts],optimset('Display','off'),rhos_new,Ms,R,Avp,Bvp);
%Ts_new = abs(PjTj(2));
%dTs = (Ts_new - Ts)/0.002;
%end-----------------------------------------------------------------------
```

PCA for Base Data

```
clear all; clc;
load('cstr_data.mat')
cstr=cstr(1:5000,2:8);
mn=mean(cstr); %mean
sd=std(cstr); %standard deviation

save('cstr_data_HE.mat','mn','sd','-append');
save('cstr_data_CAT.mat','mn','sd','-append');

[cstr_row, cstr_column]=size(cstr); %state column size for normalization loop

for i=1:cstr_column, %normalization loop
norm_column=(cstr(:,i)-mn(i))./sd(i);
cstr_norm(:,i)=norm_column;
end
i=0;

[coeff,score,latent,tsquare,explained,mu] = princomp(cstr_norm); %principal component
analysis

save('cstr_data_HE.mat','coeff','latent','-append');
save('cstr_data_CAT.mat','coeff','latent','-append');

[coeff_row,coeff_column]=size(coeff);

for i=1:8, %convert latent to square matrix
latent_mat(i,i)=latent(i,1);
end
i=0;

mn_score=mean(score); %mean of score matrix
sd_score=std(score); %standard deviation of score matrix

save('cstr_data_HE.mat','mn_score','sd_score','-append');
save('cstr_data_CAT.mat','mn_score','sd_score','-append');

[score_row, score_column]=size(score); %state column size for normalization
loop
for i=1:score_column, %normalization loop
nSCORE_column=(score(:,i)-mn_score(i))./sd_score(i);
score_norm(:,i)=nSCORE_column;
end
i=0;


no_princomp=2;                                      %no of retained component.
score=score(:,1:no_princomp);
score_square=(score_norm.^2);
%[coeff,score,latent,tsquare] = princomp(score_norm);

for i=1:5000,
column_t(i,:)=score_square(i,1:no_princomp)./(latent(1:no_princomp,1))’;
tsquare(i,1)=sum(column_t(i,:));
end
i=0;

%r=pcares(cstr_norm,no_princomp);     %pcares return residual from PCA
%q=r.*r;
%[r_row, r_column]=size(r);
```

```
backprojection=score_norm*(coeff)';
r=cstr_norm-backprojection;
q=r.*r;
[r_row, r_column]=size(r);
SPE=sum(q');

figure(2)
subplot (2,1,1);
plot (SPE) %SPE Chart

chisquare_99=chi2inv(0.99,cstr_column-1);
chisquare_95=chi2inv(0.95,cstr_column-1);

theta1=sum(latent((no_princomp+1):cstr_column,1));
theta2=sum(latent((no_princomp+1):cstr_column,1).^2);
theta3=sum(latent((no_princomp+1):cstr_column,1).^3);
g=theta2/theta1;
h=(theta1^2)/theta2;
h0=1-((2*theta1*theta3)/(3*(theta2^2)));
z_99=norminv(1-0.01);
z_95=norminv(1-0.05);

SPE_threshold99=theta1*((1-(theta2*h0*(1-h0)/(theta1^2)) +
((z_99*((2*theta2*(h0^2))^0.5))/(theta1)))^(1/h0)) %SPE limit alternative 1
SPE_threshold95=theta1*((1-(theta2*h0*(1-h0)/(theta1^2)) +
((z_95*((2*theta2*(h0^2))^0.5))/(theta1)))^(1/h0))

%SPE_threshold99=g*chisquare_99 %SPE limit alternative 2
%SPE_threshold95=g*chisquare_95
%SPE_threshold99=g*h*((1-(2/(9*h))+(z_99*((2/(9*h))^0.5)))^3) %SPE limit alternative 3
%SPE_threshold95=g*h*((1-(2/(9*h))+(z_95*((2/(9*h))^0.5)))^3)

line('XData', [0 5000], 'YData', [SPE_threshold99 SPE_threshold99], 'LineStyle', '-', ...
'LineWidth', 2, 'Color','r');
line('xData', [0 5000], 'yData', [SPE_threshold95 SPE_threshold95], 'LineStyle', '--', ..
'LineWidth', 2, 'Color','y');
legend('residual','99.0% confidence limit','95.0% confidence limit',...
'Location','NorthEastOutside')

title('SPE-Chart','FontWeight','bold')
xlabel('T, (hr)')
ylabel('residual')

finv_99=finv(0.99,no_princomp,(score_row-no_princomp));
%F alpha (no_princomp, (no of sample - no_princomp))
finv_95=finv(0.95,no_princomp,(score_row-no_princomp));
%F alpha (no_princomp, (no of sample - no_princomp))

thold_99=(((no_princomp)*(score_row-1)*(score_row+1))/(score_row*(score_row-
no_princomp)))*finv_99;
thold_95=(no_princomp)*(score_row-1)*(score_row+1)/(score_row*(score_row-
no_princomp))*finv_95;

subplot (2,1,2);
plot(tsquare) %T2 chart

line('XData', [0 5000], 'YData', [thold_99 thold_99], 'LineStyle', '-', ...
'LineWidth', 2, 'Color','r');
line('xData', [0 5000], 'yData', [thold_95 thold_95], 'LineStyle', '--', ...
'LineWidth', 2, 'Color','y');
legend('T-square','99.0% confidence limit','95.0% confidence limit',...
'Location','NorthEastOutside')
```

```
title('T-square Chart','FontWeight','bold')
xlabel('T, (hr)')
ylabel('T^2')

'--------------------end of program----------------------'
```

PCA for Drift in Reaction Kinetics

```
clear all; clc;
load('cstr_data_CAT.mat')
cstr=cstr(1:5000,2:8);

[cstr_row, cstr_column]=size(cstr); %state column size for normalization loop

for i=1:cstr_column, %normalization loop
norm_column=(cstr(:,i)-mn(i))./sd(i);
cstr_norm(:,i)=norm_column;
end
i=0;

score=cstr_norm*coeff;
[coeff_row,coeff_column]=size(coeff)

for i=1:8, %convert latent to square matrix
latent_mat(i,i)=latent(i,1);
end
i=0;

[score_row, score_column]=size(score); %state column size for normalization loop

for i=1:score_column, %normalization loop
nSCORE_column=(score(:,i)-mn_score(i))./sd_score(i);
score_norm(:,i)=nSCORE_column;
end
i=0

no_princomp=2;                                     %no of retained component
score_square=(score_norm.^2)
%[coeff,score,latent,tsquare] = princomp(score_norm);

for i=1:5000, %T-square loop
column_t(i,:)=score_square(i,1:no_princomp)./(latent(1:no_princomp,1))';
tsquare(i,1)=sum(column_t(i,:));
end
i=0

%r=pcares(cstr_norm,no_princomp); %pcares return residual from PCA
%q=r.*r;
backprojection=score_norm*(coeff)';
r=cstr_norm-backprojection;
q=r.*r;
[r_row, r_column]=size(r);
SPE=sum(q');

figure(2)
subplot (2,1,1);
plot (SPE) %SPE Chart
```

```
chisquare_99=chi2inv(0.99,cstr_column-1);
chisquare_95=chi2inv(0.95,cstr_column-1);

theta1=sum(latent((no_princomp+1):7,1));
theta2=sum(latent((no_princomp+1):7,1).^2);
theta3=sum(latent((no_princomp+1):7,1).^3);
g=theta2/theta1;
h=(theta1^2)/theta2;
h0=1-((2*theta1*theta3)/(3*(theta2^2)));
z_99=norminv(1-0.01);
z_95=norminv(1-0.05);

SPE_threshold99=theta1*((1-(theta2*h0*(1-h0)/(theta1^2)) +
((z_99*((2*theta2*(h0^2))^0.5))/(theta1)))^(1/h0)) %SPE limit alternative 1
SPE_threshold95=theta1*((1-(theta2*h0*(1-h0)/(theta1^2)) +
((z_95*((2*theta2*(h0^2))^0.5))/(theta1)))^(1/h0))

line('XData', [0 5000], 'YData', [SPE_threshold99 SPE_threshold99],
'LineStyle', '-', ...
'LineWidth', 2, 'Color','r');
line('xData', [0 5000], 'yData', [SPE_threshold95 SPE_threshold95], 'LineStyle', '--',
...
'LineWidth', 2, 'Color','y');
legend('residual','99.0% confidence limit','95.0% confidence limit',...
'Location','NorthEastOutside')
title('Drift in Reaction Kinetics','FontWeight','bold')
xlabel('T, (hr)')
ylabel('Q, PCA')

finv_99=finv(0.99,no_princomp,(score_row-no_princomp)) %F alpha (no_princomp, (no of
sample - no_princomp))
finv_95=finv(0.95,no_princomp,(score_row-no_princomp)) %F alpha (no_princomp, (no of
sample - no_princomp))

thold_99=(((no_princomp)*(score_row-1)*(score_row+1))/(score_row*(score_row-
no_princomp)))*finv_99;
thold_95=(no_princomp)*(score_row-1)*(score_row+1)/(score_row*(score_row-
no_princomp))*finv_95;

subplot (2,1,2);
plot(tsquare) %T2 chart

line('XData', [0 5000], 'YData', [thold_99 thold_99], 'LineStyle', '-', ...
'LineWidth', 2, 'Color','r');
line('xData', [0 5000], 'yData', [thold_95 thold_95], 'LineStyle', '--', ...
'LineWidth', 2, 'Color','y');
legend('T-square','99.0% confidence limit','95.0% confidence limit',...
'Location','NorthEastOutside')
title('Drift in Reaction Kinetics','FontWeight','bold')
xlabel('T, (hr)')
ylabel('T^2, PCA')

'--------------------end of program----------------------'
```

PCA for Drift in Heat Transfer Coefficient

```
clear all; clc;
load('cstr_data_HE.mat')
cstr=cstr(1:5000,2:8);
[cstr_row, cstr_column]=size(cstr); %state column size for normalization loop

for i=1:cstr_column, %normalization loop
norm_column=(cstr(:,i)-mn(i))./sd(i);
cstr_norm(:,i)=norm_column;
end
i=0;

score=cstr_norm*coeff;
[coeff_row,coeff_column]=size(coeff)

for i=1:8, %convert latent to square matrix
latent_mat(i,i)=latent(i,1);
end
i=0;
[score_row, score_column]=size(score); %state column size for normalization loop

for i=1:score_column, %normalization loop
nSCORE_column=(score(:,i)-mn_score(i))./sd_score(i);
score_norm(:,i)=nSCORE_column;
end
i=0

no_princomp=2;                                              %no of retained component
score=score(:,1:no_princomp);
score_square=(score_norm.^2)
%[coeff,score,latent,tsquare] = princomp(score_norm);

for i=1:5000,
column_t(i,:)=score_square(i,1:no_princomp)./(latent(1:no_princomp,1))';
tsquare(i,1)=sum(column_t(i,:));
end
i=0;

%r=pcares(cstr_norm,no_princomp); %pcares return residual from PCA
backprojection=score_norm*(coeff)';
r=cstr_norm-backprojection;
q=r.*r;
[r_row, r_column]=size(r);
SPE=sum(q');

figure(2)
subplot (2,1,1);
plot (SPE) %SPE Chart
chisquare_99=chi2inv(0.99,cstr_column-1);
chisquare_95=chi2inv(0.95,cstr_column-1);
theta1=sum(latent((no_princomp+1):7,1));
theta2=sum(latent((no_princomp+1):7,1).^2);
theta3=sum(latent((no_princomp+1):7,1).^3);
g=theta2/theta1;
h=(theta1^2)/theta2;
h0=1-((2*theta1*theta3)/(3*(theta2^2)));
z_99=norminv(1-0.01);
z_95=norminv(1-0.05);
SPE_threshold99=theta1*((1-(theta2*h0*(1-h0)/(theta1^2)) +
((z_99*((2*theta2*(h0^2))^0.5))/(theta1)))^(1/h0)) %SPE limit alternative 1
```

```
SPE_threshold95=theta1*((1-(theta2*h0*(1-h0)/(theta1^2)) +
((z_95*((2*theta2*(h0^2))^0.5))/(theta1)))^(1/h0))

line('XData', [0 5000], 'YData', [SPE_threshold99 SPE_threshold99], 'LineStyle', '-', ...
'LineWidth', 2, 'Color','r');
line('xData', [0 5000], 'yData', [SPE_threshold95 SPE_threshold95], 'LineStyle', '--',
...
'LineWidth', 2, 'Color','y');
legend('residual','99.0% confidence limit','95.0% confidence limit',...
'Location','NorthEastOutside')
title('Drift in Heat Transfer Coefficient','FontWeight','bold')
xlabel('T, (hr)')
ylabel('Q, PCA')

finv_99=finv(0.99,no_princomp,(score_row-no_princomp)) %F alpha (no_princomp, (no of
sample - no_princomp))
finv_95=finv(0.95,no_princomp,(score_row-no_princomp)) %F alpha (no_princomp, (no of
sample - no_princomp))
thold_99=(((no_princomp)*(score_row-1)*(score_row+1))/(score_row*(score_row-
no_princomp)))*finv_99;
thold_95=(no_princomp)*(score_row-1)*(score_row+1)/(score_row*(score_row-
no_princomp))*finv_95;

subplot (2,1,2);
plot(tsquare) %T^2 chart
line('XData', [0 5000], 'YData', [thold_99 thold_99], 'LineStyle', '-', ...
'LineWidth', 2, 'Color','r');
line('xData', [0 5000], 'yData', [thold_95 thold_95], 'LineStyle', '--', ...
'LineWidth', 2, 'Color','y');
legend('T-square','99.0% confidence limit','95.0% confidence limit',...
'Location','NorthEastOutside')
title('Drift in Heat Transfer Coefficient','FontWeight','bold')
xlabel('T,  (hr)')
ylabel('T^2, PCA')
'--------------------end of program----------------------'
```

PCA for Drift in Reaction Kinetics & Heat Transfer Coefficient

```
clear all; clc;
load('cstr_data_CAT_HE.mat')
cstr=cstr(1:5000,2:8);
[cstr_row, cstr_column]=size(cstr); %state column size for normalization loop

for i=1:cstr_column, %normalization loop
norm_column=(cstr(:,i)-mn(i))./sd(i);
cstr_norm(:,i)=norm_column;
end
i=0;

score=cstr_norm*coeff;
[coeff_row,coeff_column]=size(coeff)

for i=1:8, %convert latent to square matrix
latent_mat(i,i)=latent(i,1);
end
i=0;
[score_row, score_column]=size(score); %state column size for normalization
```

```
for i=1:score_column, %normalization loop
nSCORE_column=(score(:,i)-mn_score(i))./sd_score(i);
score_norm(:,i)=nSCORE_column;
end
i=0

no_princomp=2;                                              %no of retained component
score=score(:,1:no_princomp);
score_square=(score_norm.^2)
%[coeff,score,latent,tsquare] = princomp(score_norm);

for i=1:5000,
column_t(i,:)=score_square(i,1:no_princomp)./(latent(1:no_princomp,1))';
tsquare(i,1)=sum(column_t(i,:));
end
i=0;

%r=pcares(cstr_norm,no_princomp); %pcares return residual from PCA
backprojection=score_norm*(coeff)';
r=cstr_norm-backprojection;
q=r.*r;
[r_row, r_column]=size(r);
SPE=sum(q');

figure(2)
subplot (2,1,1);
plot (SPE) %SPE Chart
chisquare_99=chi2inv(0.99,cstr_column-1);
chisquare_95=chi2inv(0.95,cstr_column-1);
theta1=sum(latent((no_princomp+1):7,1));
theta2=sum(latent((no_princomp+1):7,1).^2);
theta3=sum(latent((no_princomp+1):7,1).^3);
g=theta2/theta1;
h=(theta1^2)/theta2;
h0=1-((2*theta1*theta3)/(3*(theta2^2)));
z_99=norminv(1-0.01);
z_95=norminv(1-0.05);
SPE_threshold99=theta1*((1-(theta2*h0*(1-h0)/(theta1^2)) +
((z_99*((2*theta2*(h0^2))^0.5))/(theta1)))^(1/h0)) %SPE limit alternative 1
SPE_threshold95=theta1*((1-(theta2*h0*(1-h0)/(theta1^2)) +
((z_95*((2*theta2*(h0^2))^0.5))/(theta1)))^(1/h0))

line('XData', [0 5000], 'YData', [SPE_threshold99 SPE_threshold99], 'LineStyle', '-', ...
'LineWidth', 2, 'Color','r');
line('xData', [0 5000], 'yData', [SPE_threshold95 SPE_threshold95], 'LineStyle', '--',
...
'LineWidth', 2, 'Color','y');
legend('residual','99.0% confidence limit','95.0% confidence limit',...
'Location','NorthEastOutside')
title('Drift in Reaction Kinetics & Heat Transfer Coefficient','FontWeight','bold')
xlabel('T, (hr)')
ylabel('Q, PCA')

finv_99=finv(0.99,no_princomp,(score_row-no_princomp)) %F alpha (no_princomp, (no of
sample - no_princomp))
finv_95=finv(0.95,no_princomp,(score_row-no_princomp)) %F alpha (no_princomp, (no of
sample - no_princomp))
thold_99=(((no_princomp)*(score_row-1)*(score_row+1))/(score_row*(score_row-
no_princomp)))*finv_99;
thold_95=(no_princomp)*(score_row-1)*(score_row+1)/(score_row*(score_row-
no_princomp))*finv_95;
```

```
subplot (2,1,2);
plot(tsquare) %T2 chart
line('XData', [0 5000], 'YData', [thold_99 thold_99], 'LineStyle', '-', ...
'LineWidth', 2, 'Color','r');
line('xData', [0 5000], 'yData', [thold_95 thold_95], 'LineStyle', '--', ...
'LineWidth', 2, 'Color','y');
legend('T-square','99.0% confidence limit','95.0% confidence limit',...
'Location','NorthEastOutside')
title('T-square Chart Drift in Reaction Kinetics & Heat Transfer
Coefficient','FontWeight','bold')
xlabel('T, (hr)')
ylabel('T^2, PCA')
'--------------------end of program----------------------'
```

DPCA for Base Data

```
clear all; clc;
load('cstr_data.mat')
cstr=cstr(1:5000,2:8);

[cstr_row, cstr_column]=size(cstr);
%cstr_tlshift=cstr(3:cstr_row,cstr_column);
no_lag=2;

for i=1:cstr_column,
if i==1;
cstr_tlshift(:,i)= cstr(((no_lag+1):cstr_row),i);
for n=1:no_lag,
cstr_tlshift(:,i+n)=cstr((no_lag+1-n):(cstr_row-n),i);
end
n=0;
else
cstr_tlshift(:,(i*(no_lag+1)-no_lag))= cstr(((no_lag+1):cstr_row),i);
for n=1:no_lag,
cstr_tlshift(:,(i*(no_lag+1)-no_lag+n))=cstr((no_lag+1-n):(cstr_row-n),i);
end
end
n=0;
end

save('dpca_all_column.mat','cstr_tlshift');
%---------------------------DPCA----------------------------------%

clear all; clc;
load('dpca_all_column.mat')
cstr=cstr_tlshift;
mn=mean(cstr); %mean
sd=std(cstr); %standard deviation

save('dpca_all_column_CAT.mat','mn','sd','-append');
save('dpca_all_column_HE.mat','mn','sd','-append');
[cstr_row, cstr_column]=size(cstr); %state column size for normalization loop

for i=1:cstr_column, %normalization loop
norm_column=(cstr(:,i)-mn(i))./sd(i);
cstr_norm(:,i)=norm_column;
```

```
end
i=0;

[coeff,score,latent,tsquare,explained,mu] = princomp(cstr_norm); %principal component
analysis

save('dpca_all_column_CAT.mat','coeff','latent','-append');
save('dpca_all_column_HE.mat','coeff','latent','-append');
[coeff_row,coeff_column]=size(coeff);
for i=1:cstr_column, %convert latent to square matrix
latent_mat(i,i)=latent(i,1);
end
i=0;

mn_score=mean(score); %mean of score matrix
sd_score=std(score); %standard deviation of score matrix
save('dpca_all_column_CAT.mat','mn_score','sd_score','-append');
save('dpca_all_column_HE.mat','mn_score','sd_score','-append');
[score_row, score_column]=size(score); %state column size for normalization loop

for i=1:score_column, %normalization loop
nSCORE_column=(score(:,i)-mn_score(i))./sd_score(i);
score_norm(:,i)=nSCORE_column;
end
i=0;

no_princomp=3;                                      %no of retained component

save('dpca_all_column_CAT.mat','no_princomp','-append');
save('dpca_all_column_HE.mat','no_princomp','-append');

score=score(:,1:no_princomp);
score_square=(score_norm.^2);
%[coeff,score,latent,tsquare] = princomp(score_norm);

for i=1:score_row,
column_t(i,:)=score_square(i,1:no_princomp)./(latent(1:no_princomp,1))';
tsquare(i,1)=sum(column_t(i,:));
end
i=0;

%r=pcares(cstr_norm,no_princomp);            %pcares return residual from PCA
backprojection=score_norm*(coeff)';
r=cstr_norm-backprojection;
q=r.*r;
[r_row, r_column]=size(r);
SPE=sum(q');

figure(2)
subplot (2,1,1);
plot (SPE) %SPE Chart

chisquare_99=chi2inv(0.99,cstr_column-1);
chisquare_95=chi2inv(0.95,cstr_column-1);
theta1=sum(latent((no_princomp+1):cstr_column,1));
theta2=sum(latent((no_princomp+1):cstr_column,1).^2);
theta3=sum(latent((no_princomp+1):cstr_column,1).^3);
g=theta2/theta1;
h=(theta1^2)/theta2;
h0=1-((2*theta1*theta3)/(3*(theta2^2)));
z_99=norminv(1-0.01);
z_95=norminv(1-0.05);
```

```
SPE_threshold99=theta1*((1-(theta2*h0*(1-h0)/(theta1^2)) +
((z_99*((2*theta2*(h0^2))^0.5))/(theta1)))^(1/h0)) %SPE limit alternative 1
SPE_threshold95=theta1*((1-(theta2*h0*(1-h0)/(theta1^2)) +
((z_95*((2*theta2*(h0^2))^0.5))/(theta1)))^(1/h0))
line('XData', [0 5000], 'YData', [SPE_threshold99 SPE_threshold99], 'LineStyle', '-', ...
'LineWidth', 2, 'Color','r');
line('xData', [0 5000], 'yData', [SPE_threshold95 SPE_threshold95], 'LineStyle', '--',
...
'LineWidth', 2, 'Color','y');
legend('residual','99.0% confidence limit','95.0% confidence limit',...
'Location','NorthEastOutside')

title('SPE-Chart','FontWeight','bold')
xlabel('T, (hr)')
ylabel('residual')

finv_99=finv(0.99,no_princomp,(score_row-no_princomp)); %F alpha (no_princomp, (no of
sample - no_princomp))
finv_95=finv(0.95,no_princomp,(score_row-no_princomp)); %F alpha (no_princomp, (no of
sample - no_princomp))
thold_99=(((no_princomp)*(score_row-1)*(score_row+1))/(score_row*(score_row-
no_princomp)))*finv_99;
thold_95=(no_princomp)*(score_row-1)*(score_row+1)/(score_row*(score_row-
no_princomp))*finv_95;

subplot (2,1,2);
plot(tsquare) %T2 chart
line('XData', [0 5000], 'YData', [thold_99 thold_99], 'LineStyle', '-', ...
'LineWidth', 2, 'Color','r');
line('xData', [0 5000], 'yData', [thold_95 thold_95], 'LineStyle', '--', ...
'LineWidth', 2, 'Color','y');
legend('T-square','99.0% confidence limit','95.0% confidence limit',...
'Location','NorthEastOutside')

title('T-square Chart','FontWeight','bold')
xlabel('T, (hr)')
ylabel('T^2')

'--------------------end of program----------------------'
```

DPCA for Drift in Reaction Kinetics

```
clear all; clc;
load('cstr_data_CAT.mat')
cstr=cstr(1:5000,2:8);

[cstr_row, cstr_column]=size(cstr);
%cstr_tlshift=cstr(3:cstr_row,cstr_column);
no_lag=2;

for i=1:cstr_column,
if i==1;
cstr_tlshift(:,i)= cstr(((no_lag+1):cstr_row),i);
for n=1:no_lag,
cstr_tlshift(:,i+n)=cstr((no_lag+1-n):(cstr_row-n),i);
```

```
end
n=0;
else
cstr_tlshift(:,(i*(no_lag+1)-no_lag))= cstr(((no_lag+1):cstr_row),i);
for n=1:no_lag,
cstr_tlshift(:,(i*(no_lag+1)-no_lag+n))=cstr((no_lag+1-n):(cstr_row-n),i);
end
end
n=0;
end

save('dpca_all_column_CAT.mat','cstr_tlshift','-append');
%-----------------------------------DPCA--------------------------%

clear all; clc;
load('dpca_all_column_CAT.mat')
cstr=cstr_tlshift;
[cstr_row, cstr_column]=size(cstr); %state column size for normalization loop

for i=1:cstr_column, %normalization loop
norm_column=(cstr(:,i)-mn(i))./sd(i);
cstr_norm(:,i)=norm_column;
end
i=0;

score=cstr_norm*coeff
[coeff_row,coeff_column]=size(coeff)

for i=1:cstr_column, %convert latent to square matrix
latent_mat(i,i)=latent(i,1);
end
i=0;
[score_row, score_column]=size(score); %state column size for normalization loop

for i=1:score_column, %normalization loop
nSCORE_column=(score(:,i)-mn_score(i))./sd_score(i);
score_norm(:,i)=nSCORE_column;
end
i=0
score_square=(score_norm.^2)

for i=1:score_row, %T-square loop
column_t(i,:)=score_square(i,1:no_princomp)./(latent(1:no_princomp,1))';
tsquare(i,1)=sum(column_t(i,:));
end
i=0

%r=pcares(cstr_norm,no_princomp); %pcares return residual from PCA
%q=r.*r;
backprojection=score_norm*(coeff)'; %back projection from the score matrix
r=cstr_norm-backprojection; %residual
q=r.*r; %the Q/SPE statistics
[r_row, r_column]=size(r);
SPE=sum(q');

figure(2)
subplot (2,1,1);
plot (SPE) %SPE Chart

chisquare_99=chi2inv(0.99,cstr_column-1);
chisquare_95=chi2inv(0.95,cstr_column-1);
theta1=sum(latent((no_princomp+1):score_column,1));
```

```
theta2=sum(latent((no_princomp+1):score_column,1).^2);
theta3=sum(latent((no_princomp+1):score_column,1).^3);
g=theta2/theta1;
h=(theta1^2)/theta2;
h0=1-((2*theta1*theta3)/(3*(theta2^2)));
z_99=norminv(1-0.01);
z_95=norminv(1-0.05);

SPE_threshold99=theta1*((1-(theta2*h0*(1-h0)/(theta1^2)) +
((z_99*((2*theta2*(h0^2))^0.5))/(theta1)))^(1/h0)) %SPE limit alternative 1
SPE_threshold95=theta1*((1-(theta2*h0*(1-h0)/(theta1^2)) +
((z_95*((2*theta2*(h0^2))^0.5))/(theta1)))^(1/h0))

line('XData', [0 5000], 'YData', [SPE_threshold99 SPE_threshold99], 'LineStyle', '-', ...
'LineWidth', 2, 'Color','r');
line('xData', [0 5000], 'yData', [SPE_threshold95 SPE_threshold95], 'LineStyle', '--',
...
'LineWidth', 2, 'Color','y');
legend('residual','99.0% confidence limit','95.0% confidence limit',...
'Location','NorthEastOutside')

title('Drift in Reaction Kinetics','FontWeight','bold')
xlabel('T, (hr)')
ylabel('Q, DPCA')

finv_99=finv(0.99,no_princomp,(score_row-no_princomp)) %F alpha (no_princomp, (no of
sample - no_princomp))
finv_95=finv(0.95,no_princomp,(score_row-no_princomp)) %F alpha (no_princomp, (no of
sample - no_princomp))
thold_99=(((no_princomp)*(score_row-1)*(score_row+1))/(score_row*(score_row-
no_princomp)))*finv_99;
thold_95=(no_princomp)*(score_row-1)*(score_row+1)/(score_row*(score_row-
no_princomp))*finv_95;

subplot (2,1,2);
plot(tsquare)                          %T2 chart
line('XData', [0 5000], 'YData', [thold_99 thold_99], 'LineStyle', '-', ...
'LineWidth', 2, 'Color','r');
line('xData', [0 5000], 'yData', [thold_95 thold_95], 'LineStyle', '--', ...
'LineWidth', 2, 'Color','y');
legend('T-square','99.0% confidence limit','95.0% confidence limit',...
'Location','NorthEastOutside')

title('Drift in Reaction Kinetics','FontWeight','bold')
xlabel('T, (hr)')
ylabel('T^2, DPCA')


'--------------------end of program---------------------'
```

DPCA for Drift in Heat Transfer Coefficient

```
clear all; clc;
load('cstr_data_HE.mat')
cstr=cstr(1:5000,2:8);

[cstr_row, cstr_column]=size(cstr);
%cstr_tlshift=cstr(3:cstr_row,cstr_column);
no_lag=2;

for i=1:cstr_column,
if i==1;
cstr_tlshift(:,i)= cstr(((no_lag+1):cstr_row),i);
for n=1:no_lag,
cstr_tlshift(:,i+n)=cstr((no_lag+1-n):(cstr_row-n),i);
end
n=0;
else
cstr_tlshift(:,(i*(no_lag+1)-no_lag))= cstr(((no_lag+1):cstr_row),i);
for n=1:no_lag,
cstr_tlshift(:,(i*(no_lag+1)-no_lag+n))=cstr((no_lag+1-n):(cstr_row-n),i);
end
end
n=0;
end

save('dpca_all_column_HE.mat','cstr_tlshift','-append');
%--------------------------------------DPCA------------------------%

clear all; clc;
load('dpca_all_column_HE.mat')
cstr=cstr_tlshift;
[cstr_row, cstr_column]=size(cstr); %state column size for normalization loop

for i=1:cstr_column, %normalization loop
norm_column=(cstr(:,i)-mn(i))./sd(i);
cstr_norm(:,i)=norm_column;
end
i=0;

score=cstr_norm*coeff;
[coeff_row,coeff_column]=size(coeff);
for i=1:cstr_column, %convert latent to square matrix
latent_mat(i,i)=latent(i,1);
end
i=0;
[score_row, score_column]=size(score); %state column size for normalization loop

for i=1:score_column, %normalization loop
nSCORE_column=(score(:,i)-mn_score(i))./sd_score(i);
score_norm(:,i)=nSCORE_column;
end
i=0;
score_square=(score_norm.^2);

for i=1:score_row, %T-square loop
column_t(i,:)=score_square(i,1:no_princomp)./(latent(1:no_princomp,1))';
tsquare(i,1)=sum(column_t(i,:));
end
i=0;
```

```
%r=pcares(cstr_norm,no_princomp); %pcares return residual from PCA
%q=r.*r;
backprojection=score_norm*(coeff)'; %back projection from the score matrix
r=cstr_norm-backprojection; %residual
q=r.*r; %the Q/SPE statistics
[r_row, r_column]=size(r);
SPE=sum(q');

figure(2)
subplot (2,1,1);
plot (SPE) %SPE Chart

chisquare_99=chi2inv(0.99,cstr_column-1);
chisquare_95=chi2inv(0.95,cstr_column-1);
theta1=sum(latent((no_princomp+1):score_column,1));
theta2=sum(latent((no_princomp+1):score_column,1).^2);
theta3=sum(latent((no_princomp+1):score_column,1).^3);
g=theta2/theta1;
h=(theta1^2)/theta2;
h0=1-((2*theta1*theta3)/(3*(theta2^2)));

z_99=norminv(1-0.01);
z_95=norminv(1-0.05);

SPE_threshold99=theta1*((1-(theta2*h0*(1-h0)/(theta1^2)) +
((z_99*((2*theta2*(h0^2))^0.5))/(theta1)))^(1/h0)); %SPE limit alternative 1
SPE_threshold95=theta1*((1-(theta2*h0*(1-h0)/(theta1^2)) +
((z_95*((2*theta2*(h0^2))^0.5))/(theta1)))^(1/h0));

line('XData', [0 5000], 'YData', [SPE_threshold99 SPE_threshold99], 'LineStyle', '-', ...
'LineWidth', 2, 'Color','r');
line('xData', [0 5000], 'yData', [SPE_threshold95 SPE_threshold95], 'LineStyle', '--',
...
'LineWidth', 2, 'Color','y');
legend('residual','99.0% confidence limit','95.0% confidence limit',...
'Location','NorthEastOutside')

title('Drift in Heat Transfer Coefficient','FontWeight','bold')
xlabel('T, (hr)')
ylabel('Q, DPCA')
finv_99=finv(0.99,no_princomp,(score_row-no_princomp)); %F alpha (no_princomp, (no of
sample - no_princomp))
finv_95=finv(0.95,no_princomp,(score_row-no_princomp)); %F alpha (no_princomp, (no of
sample - no_princomp))

thold_99=(((no_princomp)*(score_row-1)*(score_row+1))/(score_row*(score_row-
no_princomp)))*finv_99;
thold_95=(no_princomp)*(score_row-1)*(score_row+1)/(score_row*(score_row-
no_princomp))*finv_95;

subplot (2,1,2);
plot(tsquare) %T2 chart
line('XData', [0 5000], 'YData', [thold_99 thold_99], 'LineStyle', '-', ...
'LineWidth', 2, 'Color','r');
line('xData', [0 5000], 'yData', [thold_95 thold_95], 'LineStyle', '--', ...
'LineWidth', 2, 'Color','y');
legend('T-square','99.0% confidence limit','95.0% confidence limit',...
'Location','NorthEastOutside')
title('Drift in Heat Transfer Coefficient ','FontWeight','bold')
xlabel('T, (hr)')
ylabel('T^2, DPCA')
'--------------------end of program----------------------'
```

DPCA for Drift in Reaction Kinetics & Heat Transfer Coefficient

```
clear all; clc;
load('cstr_data_CAT_HE.mat')
cstr=cstr(1:5000,2:8);

[cstr_row, cstr_column]=size(cstr);
%cstr_tlshift=cstr(3:cstr_row,cstr_column);
no_lag=2;

for i=1:cstr_column,
if i==1;
cstr_tlshift(:,i)= cstr(((no_lag+1):cstr_row),i);
for n=1:no_lag,
cstr_tlshift(:,i+n)=cstr((no_lag+1-n):(cstr_row-n),i);
end
n=0;
else
cstr_tlshift(:,(i*(no_lag+1)-no_lag))= cstr(((no_lag+1):cstr_row),i);
for n=1:no_lag,
cstr_tlshift(:,(i*(no_lag+1)-no_lag+n))=cstr((no_lag+1-n):(cstr_row-n),i);
end
end
n=0;
end

save('dpca_all_column_HE.mat','cstr_tlshift','-append');
%--------------------------------------DPCA------------------------%

clear all; clc;
load('dpca_all_column_HE.mat')
cstr=cstr_tlshift;
[cstr_row, cstr_column]=size(cstr); %state column size for normalization loop

for i=1:cstr_column, %normalization loop
norm_column=(cstr(:,i)-mn(i))./sd(i);
cstr_norm(:,i)=norm_column;
end
i=0;

score=cstr_norm*coeff;
[coeff_row,coeff_column]=size(coeff);
for i=1:cstr_column, %convert latent to square matrix
latent_mat(i,i)=latent(i,1);
end
i=0;
[score_row, score_column]=size(score); %state column size for normalization loop

for i=1:score_column, %normalization loop
nSCORE_column=(score(:,i)-mn_score(i))./sd_score(i);
score_norm(:,i)=nSCORE_column;
end
i=0;
score_square=(score_norm.^2);

for i=1:score_row, %T-square loop
column_t(i,:)=score_square(i,1:no_princomp)./(latent(1:no_princomp,1))';
tsquare(i,1)=sum(column_t(i,:));
end
i=0;
```

```
%r=pcares(cstr_norm,no_princomp); %pcares return residual from PCA
%q=r.*r;
backprojection=score_norm*(coeff)'; %back projection from the score matrix
r=cstr_norm-backprojection; %residual
q=r.*r; %the Q/SPE statistics
[r_row, r_column]=size(r);
SPE=sum(q');

figure(2)
subplot (2,1,1);
plot (SPE) %SPE Chart

chisquare_99=chi2inv(0.99,cstr_column-1);
chisquare_95=chi2inv(0.95,cstr_column-1);
theta1=sum(latent((no_princomp+1):score_column,1));
theta2=sum(latent((no_princomp+1):score_column,1).^2);
theta3=sum(latent((no_princomp+1):score_column,1).^3);
g=theta2/theta1;
h=(theta1^2)/theta2;
h0=1-((2*theta1*theta3)/(3*(theta2^2)));

z_99=norminv(1-0.01);
z_95=norminv(1-0.05);

SPE_threshold99=theta1*((1-(theta2*h0*(1-h0)/(theta1^2)) +
((z_99*((2*theta2*(h0^2))^0.5))/(theta1)))^(1/h0)); %SPE limit alternative 1
SPE_threshold95=theta1*((1-(theta2*h0*(1-h0)/(theta1^2)) +
((z_95*((2*theta2*(h0^2))^0.5))/(theta1)))^(1/h0));

line('XData', [0 5000], 'YData', [SPE_threshold99 SPE_threshold99], 'LineStyle', '-', ...
'LineWidth', 2, 'Color','r');
line('xData', [0 5000], 'yData', [SPE_threshold95 SPE_threshold95], 'LineStyle', '--',
...
'LineWidth', 2, 'Color','y');
legend('residual','99.0% confidence limit','95.0% confidence limit',...
'Location','NorthEastOutside')

title(' Drift in Reaction Kinetics & Heat Transfer Coefficient ','FontWeight','bold')
xlabel('T,(hr)')
ylabel('Q, DPCA ')

finv_99=finv(0.99,no_princomp,(score_row-no_princomp)); %F alpha (no_princomp, (no of
sample - no_princomp))
finv_95=finv(0.95,no_princomp,(score_row-no_princomp)); %F alpha (no_princomp, (no of
sample - no_princomp))
thold_99=(((no_princomp)*(score_row-1)*(score_row+1))/(score_row*(score_row-
no_princomp)))*finv_99;
thold_95=(no_princomp)*(score_row-1)*(score_row+1)/(score_row*(score_row-
no_princomp))*finv_95;

subplot (2,1,2);
plot(tsquare) %T2 chart
line('XData', [0 5000], 'YData', [thold_99 thold_99], 'LineStyle', '-', ...
'LineWidth', 2, 'Color','r');
line('xData', [0 5000], 'yData', [thold_95 thold_95], 'LineStyle', '--', ...
'LineWidth', 2, 'Color','y');
legend('T-square','99.0% confidence limit','95.0% confidence limit',...
'Location','NorthEastOutside')
title(' Drift in Reaction Kinetics & Heat Transfer Coefficient','FontWeight','bold')
xlabel('T, (hr)')
ylabel('T^2, DPCA')
'--------------------end of program----------------------'
```