ON-LINE FAULT DIAGNOSIS AND ACCOMMODATION FOR SYSTEM FAILURES

By

LIANG-WEI HO

Bachelor of Science
Tamkang University
Taipei, Taiwan
1993

Master of Science
Oklahoma State University
Stillwater, Oklahoma
1997

Submitted to the Faculty of the
Graduate Collage of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
December, 2000

ON-LINE FAULT DIAGNOSIS AND

ACCOMMODATION FOR

SYSTEM FAILURES

Thesis Approved:

[illegible]

Thesis Adviser

[illegible]

[illegible]

[illegible]

[illegible]

Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my major adviser, Dr. Gary G. Yen, for his kind support, encouragement, and friendship throughout my study. Special thanks goes to Dr. Martin Hagan for his meticulously generous suggestions in using MATLAB Neural Network Toolbox for the on-line simulation. His professional comments were very helpful to this research work.

Moreover, I also wish to give my appreciation to my other committee members, Dr. Carl D. Latino and Dr. Marvin Stone, their support and friendship were also invaluable to me. Finally, I would like to dedicate this dissertation to my beloved parents, Chih-Kuo Ho and Hsiu-Lien Weng, and my family for their love, support, and understanding.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

CHAPTER I

INTRODUCTION

1.1 Overview of fault diagnosis and accommodation

Modern engineering technology is leading to increasingly complex systems with ever more demanding performance criteria. However, time-critical control recovery due to catastrophic failures is often left unsolved. The ultimate pursuit of a higher degree of autonomous behavior that provides constant health monitoring and fault tolerance for a complex dynamic system with minimum human intervention has high priority in order to achieve a successful control mission. As dynamic systems become more complex, experience rapidly changing environments, and encounter a greater variety of unexpected failures, the system is no longer reliable to perform properly because of the deviations of the system dynamics. Those deviations can be characterized by drastic changes of the system parameters or, more seriously, the inherent dynamical structure of the system. The system stability becomes a critical issue after those dramatic dynamic changes. In many safety-critical systems such as aircrafts or nuclear plants, system stability under failure situations seriously impacts human survivability.

Urging by these growing demands in system safety and reliability, extensive research activities have been focusing on developing Fault Diagnosis and

Accommodation (FDA) or so-called Fault Tolerant Control (FTC) methodology to maintain the system stability and to avoid the loss of human life under various failure scenarios during the past few decades. The major objective of FDA is to detect, diagnose, and accommodate any system failures. Strictly speaking, the term, failure, usually refers to a complete breakdown and the term, fault, usually suggests that the situations are tolerable. Following the common terminology used in the FDA research community [1], the terms failure and fault will be used as synonyms throughout this dissertation.

Failure situations can be typically characterized into three different categories, sensor failures, actuator failures, and component failures. Traditional FDA approaches are based upon the so-called physical or hardware redundancy, where redundant hardware components or systems are used for backup. In the event of a failure or malfunction, a backup system is switched. Due to the increasing complexity in modern hardware systems and the extra space and cost needed for the redundancies, this approach is both realistically and economically infeasible and unattractive. The major attention has moved toward the so-called model-based analytical redundancy where powerful computing devices and a mathematical model of the system are used to create the necessary redundancy to monitor and analyze the system behavior. However, because of the difficulty in obtaining an analytical model of a complex system and the inherent complexity of the system dynamics under failure situations, most of the FDA research works are aimed especially either at the linear system [1,3-8,11-21,89-92,104-108] or a certain type of nonlinear systems under simple failure situations [2,9-10,22-26,63,85-86], where the faults are usually modeled as the deviation of the system parameters, or additive disturbances, which limits the possible failure scenarios and restricts the usefulness of these techniques. Nevertheless, for a dynamic system under totally unanticipated catastrophic system failures, it is not a reasonable approach to assume certain types of dynamic change caused by those unexpected failures. A truly autonomous FDA system is the one that can detect the failures, identify them, and perform effective control law reconfiguration to accommodate the tolerable failures in on-line situation without human intervention. The representative fault diagnosis (FD) and fault accommodation (FA) techniques reported in literatures are systematically shown in Figures 1.1 and 1.2, respectively.

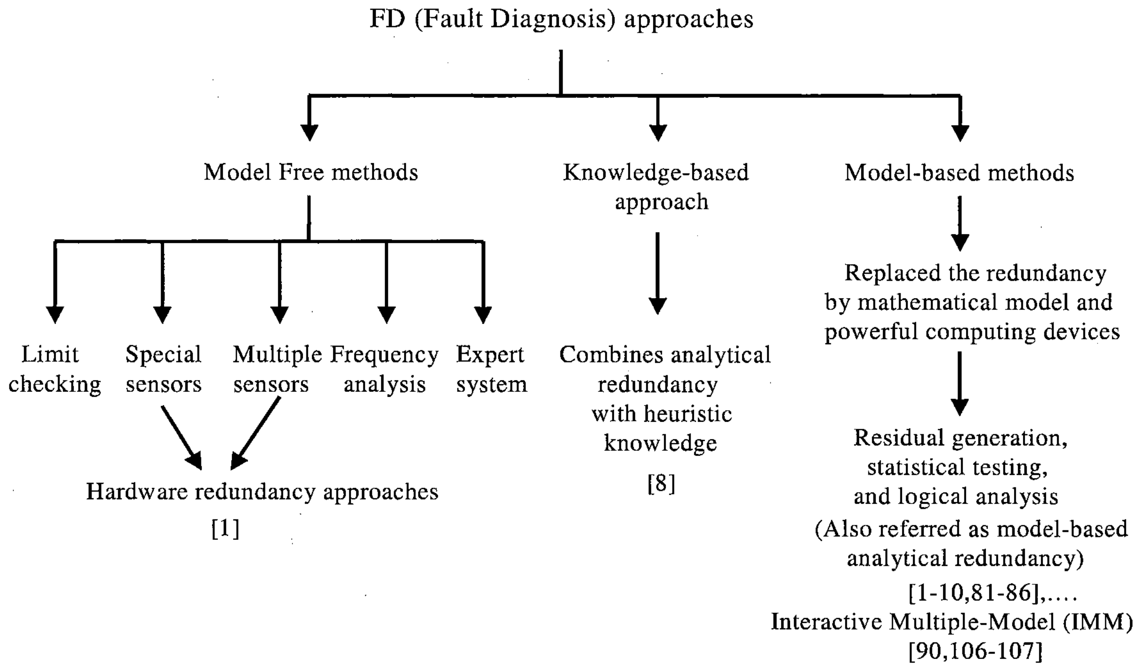
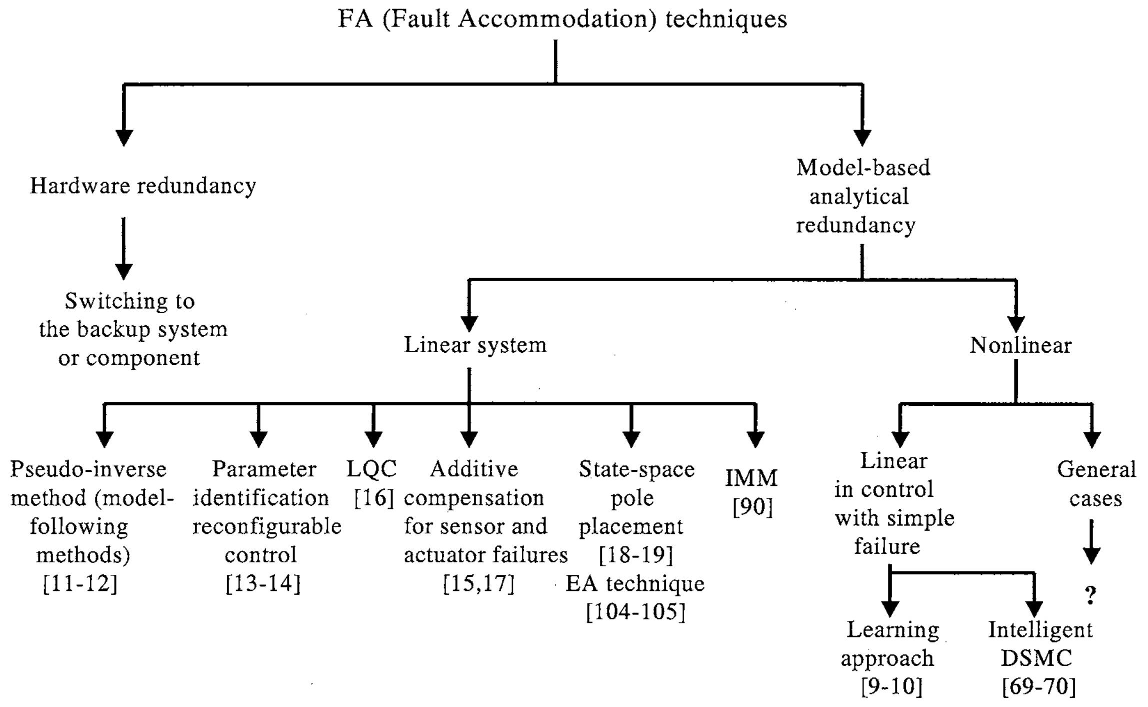

FD (Fault Diagnosis) approaches
Model Free methods
Knowledge-based approach
Model-based methods
Limit checking
Special sensors
Multiple sensors
Frequency analysis
Expert system
Hardware redundancy approaches [1]
Combines analytical redundancy with heuristic knowledge [8]
Replaced the redundancy by mathematical model and powerful computing devices
Residual generation, statistical testing, and logical analysis (Also referred as model-based analytical redundancy) [1-10,81-86],....
Interactive Multiple-Model (IMM) [90,106-107]

Figure 1.1 Typical FD approaches

With the inherent nonlinear features and self-adaptation capability, modern intelligent techniques such as neural network, fuzzy logic, and evolutionary algorithms have received extensive interests from various academic and industrial communities [57,71,73-74,96-103]. The synergistic combination of these intelligent techniques and modern control technologies for the more general and sophisticated fault diagnosis and accommodation scheme to achieve the successful autonomous control mission has become a challenging research discipline.



FA (Fault Accommodation) techniques
Hardware redundancy
Switching to the backup system or component
Model-based analytical redundancy
Linear system
Nonlinear
Pseudo-inverse method (model-following methods) [11-12]
Parameter identification reconfigurable control [13-14]
LQC [16]
Additive compensation for sensor and actuator failures [15,17]
State-space pole placement [18-19] EA technique [104-105]
IMM [90]
Linear in control with simple failure
General cases
?
Learning approach [9-10]
Intelligent DSMC [69-70]

Figure 1.2 Typical FA approaches

1.2 Problem statement

The general SISO dynamic system can be represented by Equation (1.1) [37],

$$\begin{aligned} x(k+1) &= f(x(k),u(k)) \\ y(k) &= h(x(k)) \end{aligned} \tag{1.1}$$

where $x(k) \in \Re^n, y(k) \in \Re$, and $u(k) \in \Re$ denote the system state variables, output, and input, respectively, and $f: \Re^n \times \Re \to \Re^n, h: \Re^n \to \Re$. The problems of control related to system (1.1) can be divided into the following three cases [37]:

1). f and h are known, and the state variables $x(k)$ are accessible.

2). f and h are unknown, but the state variables $x(k)$ are accessible.

3). f and h are unknown, and only the input $u(k)$ and system output $y(k)$ are available.

In the case 1), the system is completely known. Traditional control techniques can be applied directly to analyze the system and, based upon the realization of the system characteristics, the appropriate controllers for different control objectives can be developed. Case 2) corresponds to an adaptive control problem in which both f and h have to be estimated. However, the fact that the state variables, $x(k)$, are measurable makes the control problem relatively simpler than that in case 3) where both system identification and control have to be carried out using the only information available, the input-output data. Nevertheless, due to the complexity of the system dynamics under various failure scenarios, the case 3) is the major problem we are most interested in. Observing the system (1.1) closely, it is easy to show that the system output, $y(k+1)$, is actually a function of past outputs and inputs [37], $[y(k), y(k-1),\ldots, y(k-n+1), u(k), u(k-1),\ldots, u(k-n+1)]$. For the remaining of this dissertation, the least available system information will be assumed and the main focus will be placed in the general case 3). In other words, only the system input-output measurements will be assumed available for the on-line fault tolerant control problems of our interest.

Consider a general MIMO dynamic system that can be described by Equation (1.2),

$$\begin{aligned}
&y_l(k+d) = f_l(\bar{y}_1, \bar{y}_2, \ldots \bar{y}_m, \bar{u}_1, \bar{u}_2, \ldots \bar{u}_n),\\
&\bar{y}_i = \{y_i(k+d-1), y_i(k+d-2), \ldots, y_i(k+d-p_i)\},\\
&\bar{u}_j = \{u_j(k), u_j(k-1), \ldots u_j(k-q_j)\},\\
&p_i, q_j \in \Re^+, i = 1, 2, \ldots, m., j = 1, 2, \ldots, n., \mathit{and}\ l = 1, 2, \ldots, m.,
\end{aligned} \tag{1.2}$$

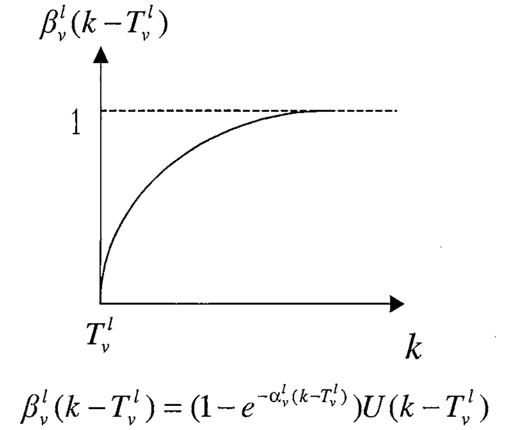
where $f_l : \Re^P \times \Re^Q \mapsto \Re$, with $P = \sum_{i=1}^{m} p_i$, $Q = \sum_{j=1}^{n} q_j$ is the mathematical realization of the system dynamics for the lth output. $y_l, y_i, u_j \in \Re$ are the lth and ith system outputs and jth input, respectively. d is the relative degree of the system (the smallest delay from the input signal to the system output). In general, f_l may not be readily available all the time in mathematical format due to the difficulty of modeling a complex dynamic system. However, it is possible to develop a realization to describe the system behavior with a known bounded uncertainty within the desired working region of the system using all the existing modeling techniques, provided enough resource and sufficient time for the development of the realization [29,34,37-39], as shown in Equation (1.3). These techniques may include the modern intelligent technology such as artificial neural networks or fuzzy-nets.

$$y_l(k+d) = \hat{f}_l(\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_m, \bar{u}_1, \bar{u}_2, \ldots \bar{u}_n) + \eta_l(y,u), \tag{1.3}$$

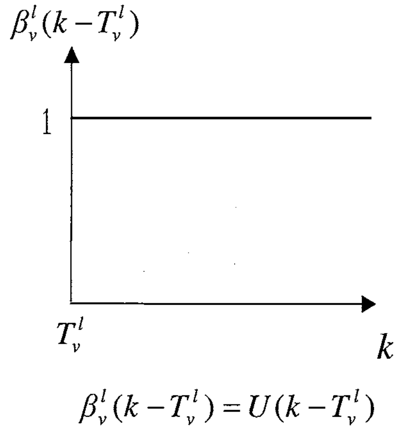
where $\|\eta_l(y,u)\|_2 \le \delta_0$, $\forall (y,u) \subseteq (\mathbf{Y}, \mathbf{U})$, $(\mathbf{Y}, \mathbf{U})$ represents the desired working regime, and $\delta_0 \in \Re$ is a known constant. Thus, $\hat{f}_l$, the realization of the real system with a known bounded uncertainty within the desired working region of the system, will be either a mathematical, numerical, or a combined realization and it is assumed that this realization is developed off-line and available. Equation (1.2) denotes a healthy system under the fault-free situation and Equation (1.3) is the corresponding nominal model. Under different component failures, the system dynamics is changed and represented by the following equation:

$$y_l(k+d) = f_l(\bar{y}_1, \bar{y}_2, \ldots \bar{y}_m, \bar{u}_1, \bar{u}_2, \ldots \bar{u}_n) + \sum_{v=1}^{r} \beta_v^l(k - T_v^l) F_v^l(\bar{y}_1, \bar{y}_2, \ldots \bar{y}_m, \bar{u}_1, \bar{u}_2, \ldots \bar{u}_n, k), \quad (1.4)$$

where $F_v^l(\cdot): \Re^P \times \Re^Q \times \Re^+ \mapsto \Re$ with $P = \sum_{i=1}^{m} p_i$, $Q = \sum_{j=1}^{n} q_j$ represents the dynamic change (a general *time-varying* function depends upon past system outputs, past control inputs, and the current control input) caused by the unknown and possibly unanticipated failure mode v for the lth output and $\beta_v^l(\cdot)$ denotes the corresponding time profile. $F_v^l(\cdot)$, $\beta_v^l(\cdot)$, and T_v^l are assumed unknown due to the possible occurrence of unanticipated failures. r is the number of system failures. All the cases in which $r > 1$ are referred to as multiple-failure cases. Two typical faults, incipient faults and abrupt faults, are considered to be involved on-line. Their characteristics can be described by the time varying constant gain, $\beta_v^l(\cdot)$, shown in Figure 1.3.



a). Incipient fault



b). Abrupt fault

Figure 1.3 Time profiles of the incipient and abrupt faults

$\alpha_v^l \in \Re^+$ is an unknown constant which defines the time profile of the incipient failure mode v and $U(k)$ denotes the unit step function. Abrupt failures are used to represent the sudden change of the system dynamics due to catastrophic malfunction or failure of the system component and the incipient failures are used to describe the time-varying effect of the system component-aging problem. The control objective is to generate appropriate control signals to stabilize the system and, possibly, drive the system outputs back to the desired trajectories, $y_{dl}(k+d) \in \Re, l = 1, 2, \ldots, m.$, in on-line situations with the presence of the abrupt and/or incipient faults.

1.3 Motivation and objective of study

Clearly, observing Equation (1.4), it is easy to find that the contemporary control theories and technology are unable to solve the control problem with the presence of the unknown failure dynamics and their corresponding time-varying profiles in on-line situations. Under the general structure of system dynamics and various possible failure scenarios, existing FDA approaches are inadequate to achieve the successful control mission. From the on-line control point of view, it is, at least now, impossible to perform exact fault detection and identification right after the occurrence of failures. Conflicting requirements exist between the amount of time that the fault detection scheme takes and the information that the scheme can provide [19]. In on-line situations, the interesting and important question becomes how to properly control the system behavior in time to prevent the failure from causing more serious lost, if the system under failures is still controllable at that time. Afterwards, how to possibly recover the system performance based upon only imprecise or insufficient information while the system dynamics may suddenly change dramatically or continuously change with time due to the failures. The development of a more sophisticated intelligent on-line control methodology to identify the failure dynamics and perform control law adjustments or re-configuration with the least human intervention is in critical need.

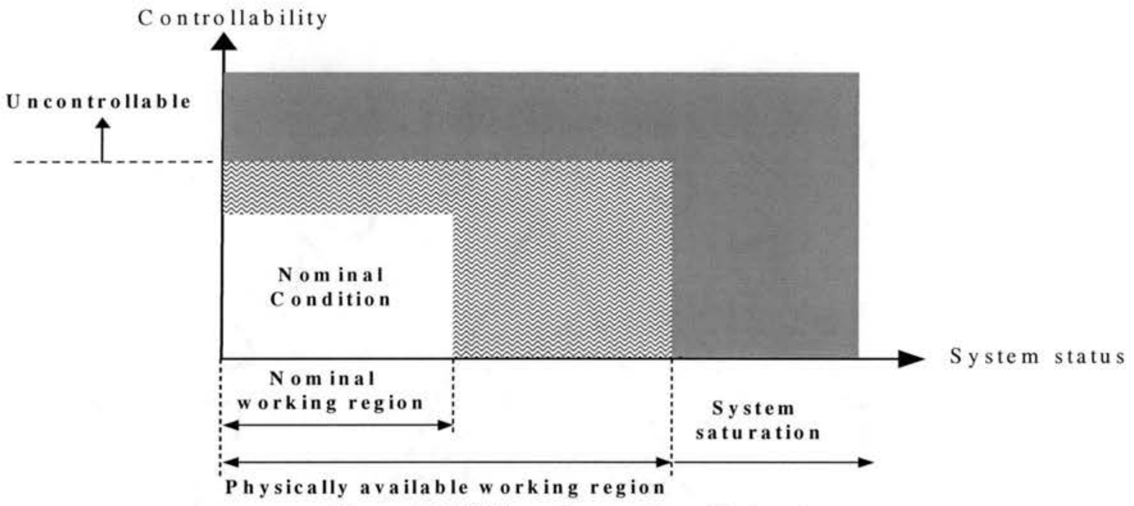


Figure 1.4 FTC problem region of interest

Of course, many system failure situations are catastrophic and uncontrollable. For example, if the sensor loop malfunctions such that all the readings from the sensor are lost or meaningless, without knowing the true failure, the only way to possibly maintain the system safety is by having human interference such as shut-down of the system, failure diagnosis, and replacement of the faulty parts. If the failures actually break down the control input to the system, there is no way to perform any control recovery. Figure 1.4 shows the interest problem region of this research, the curve-line area, where the system behavior under failures are out of the nominal working area, but still controllable and within the physically available working region. The major FTC objective is to prevent the faulty system from moving into the saturation region and possibly drive it back to the nominal condition, which brings out the following important questions:

1. Are there any conditions or constraints under which the system on-line safety and performance can be guaranteed?

2. If the answer of the first question is positive, how do we quantify those conditions or constraints?
3. If the systems under failures are still controllable and the faults are still tolerable, what is the detail systematic procedure to effectively and efficiently detect the failures, identify the change of the system dynamics, reconfigure or adjust the control actions to accommodate the failures, maintain the system on-line safety, and possibly recover the system performance by using only the insufficient information in the on-line situation without human intervention?

This research work is dedicated to the investigation of on-line fault tolerant control problems for unanticipated catastrophic system failures and to provide answers to the questions listed above. The major interest will focus on system component failures such that the system dynamics under these failures can be represented by Equation (1.4). A general intelligent on-line fault accommodation control technique and framework are proposed to deal with the on-line fault detection and control law reconfiguration problems for proper failure accommodations. Through a theoretical analysis of the on-line fault tolerant control problems based upon discrete-time Lyapunov stability theory, the necessary and sufficient conditions to guarantee the system on-line stability and survivability have been established. Incorporating a cost-effective failure detection and a multiple-model based failure diagnosis process with the developed fault accommodation framework, a more complete architecture of the multiple-model based fault diagnosis and accommodation is also presented to efficiently handle the false alarms, the accommodation of the anticipated failures, and to reduce the unnecessary control effort and computational complexity in on-line situations.

This dissertation is organized as follows. Chapter 2 provides a more detail overview of the traditional and existing FDA approaches and the modern intelligent techniques related to the control issues. A theoretical foundation and analysis of the on-line fault tolerant control problems is established in Chapter 3 together with the proposed intelligent on-line fault accommodation control methodology. In Chapter 4, extensive on-line numerical simulations are presented to evaluate and validate the proposed methodology. Real-time hardware experiments are presented in Chapter 5 to substantiate the effectiveness of the proposed fault accommodation technique and to demonstrate the possibility of successful fault tolerance in real applications. A complete architecture of the multiple-model based fault diagnosis and accommodation framework is presented in Chapter 6 together with on-line simulation study and discussions. The conclusions and recommended future research work are given in Chapter 7.

CHAPTER II

LITERATURE REVIEW

2.1 Fault diagnosis

Fault diagnosis typically consists of three different tasks, failure detection, failure isolation, and failure identification. Failure detection usually refers to the process of detecting system abnormal behavior due to faults, failure isolation is the task of determining the exact location of the failure, and failure identification refers to the determination of the size of the failure [1].

According to one early extensive survey [1], the approaches to failure detection and isolation fall into two major categories, model-free methods and model-based methods. The representative model-free approaches include the following methods:

1. Limit checking: different system measurements are compared to their corresponding pre-specified limits (thresholds). Exceeding the corresponding limit indicates that a failure occurs in the corresponding location.
2. Installation of special sensors: using special sensors to monitor hardware limits (e.g., limit temperature or pressure) or measure some special variables (e.g., sound, vibration, etc.).
3. Installation of multiple sensors: this method is aimed especially at detecting and isolating sensor failures. Measurements of the same variable from different sensors

are compared to decide the faulty one. The considered correct reading can be decided by a majority vote.

4. Frequency analysis: some plant measurements have a typical frequency spectrum under normal operating conditions. Certain types of failure may have a characteristic signature in the spectrum that can be used for both failure detection and isolation.
5. Expert system approach: an expert system consisting of a rule database in the form of "IF symptom1 AND symptom2 THEN conclusion" is used for failure detection and isolation.

Approaches 2 and 3 are also referred to as physical redundancy or hardware redundancy. The additional cost, space, and complexity of incorporating redundant hardware make those approaches unattractive [2]. With the availability of the inexpensive and powerful microprocessor, model-based methods or so-called model-based analytical redundancy have dominated the FDA research activities and received substantial attention in the past two decades [1-26,63,75,81-86]. In the analytical redundancy approach, a mathematical model of the physical system is used for monitoring and comparison. Obviously, the major reason this approach is so prevalent is the fact that the information processing techniques, which use powerful computing devices and memory systems, can also be used to create the necessary redundancy without the need of the hardware instrumentation in the system [2].

Unlike the hardware redundancy, sensory measurements are now being compared with the analytically obtained values of the respective variable through the mathematical model. The resulting differences are so-called residuals [1]. In the ideal situation, the residuals will be zeros in the fault-free system and any deviation will be interpreted as an indication of faults. But this is rarely true in practice with the presences of measurement noises and modeling errors. The deviation now can be the combinational results of noises, modeling errors, and faults. Naturally, with the presence of significant noises, statistical analysis of the residuals becomes a necessary and reasonable procedure to generate a logical pattern, called signature of the failure, for the proper fault detection and isolation. Many research activities have also been dedicated to investigate a proper residual generation to facilitate the fault isolation process [81-84,87-89]. Generally speaking, residual generation, statistical testing, and logical analysis are usually combined as the three stages of the fault detection and isolation procedure [3].

Three typical properties of the failure isolation procedure, isolability, sensitivity, and robustness significantly affect the usefulness of the procedure. Isolability is the ability of the procedure to distinguish specific failures, sensitivity is a measure characterizing the size of the faults that can be isolated with the presences of noises and disturbances, and robustness is the ability to isolate the faults in the presence of modeling uncertainties or errors [1]. Apparently, a good fault diagnosis scheme should be robust with respect to the modeling errors and sensitive to the failures. Unfortunately, the presences of noises, disturbances, and modeling uncertainties will obviously obscure the effect of faults in the residuals and possibly cause miss detection of failures and false alarm situations. The trade-off problems between the robustness and sensitivity of the fault diagnosis scheme have attracted many research efforts [4-6].

Other than the hardware redundancy and model-based analytical redundancy, in the cases where only the poor or imprecise analytical models are available, an approach called the knowledge-based method is suggested by [8]. This approach combines both analytical and heuristic knowledge to form a knowledge base. Such knowledge may include the degree of component aging, history of fault statistics, etc. The core of the knowledge-based approach is an on-line expert system that combines the analytical model-based redundancy methods for the fault detection and isolation with the method of fault diagnosis by evaluation of heuristic knowledge about the process through an inference engine [8].

Due to the inherent complexity of nonlinear systems, most of model-based analytical redundancy fault diagnosis studies deal with the linear system that is subject to simple additive or multiplicative faults. A basic framework of the fault diagnosis and accommodation architecture for more general systems and failure situations is suggested by [2,9,10,23-25] and shown in Figure 2.1.

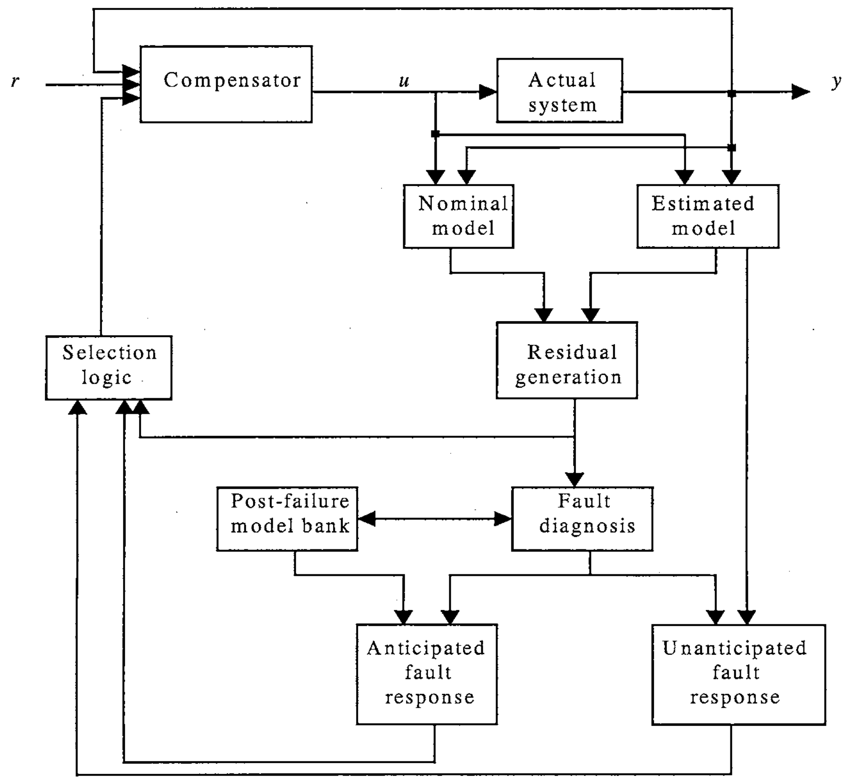


Figure 2.1 Description of the fault diagnosis and accommodation (FDA) architecture

The primary function of the estimated model is to track the actual system dynamics. In order to detect any off-nominal system behavior, the estimated model is also compared with the nominal model to generate the residuals, which serve as a measure of the deviation between the estimated model and the nominal plant. Based on the residual vector and its trending, a decision can be made for whether a system failure is emerging or not through a fault diagnosis mechanism. Once a failure is detected, the characteristics of the failure are compared with the signatures of any known failure modes to decide whether the fault is anticipated (known) or unexpected by using a post-failure model bank which is updated periodically to incorporate the signatures of any new failures.

The estimated model combines the knowledge of the nominal model and an on-line approximator to keep tracking the mathematical representation of the actual system dynamics. The on-line approximator is used not only for the failure detection, but also for estimation of the size of the failures (failure identification). During the nominal situation which corresponds to the fault-free condition, the on-line approximator will not be identically zero because of the existence of the modeling uncertainty and noises. However, once a fault occurs, the output of the approximator should significantly deviate from zero, which will be considered as a failure situation. The research work [85-86] discusses the corresponding fault isolability condition if there is only one failure happened. This approach is inspired by the inherent adaptive features of neural networks to provide a powerful learning scheme for automated fault diagnosis [2]. It does require substantial computations to implement since the parameters of the nonlinear adaptive observers have to be adjusted on-line. Thus, the successful isolation mission can be achieved only when all the assumptions and conditions are met. In other words, more practical and complicated FDD (Fault Detection and Diagnosis) problems such as detection and diagnosis of possible multiple failures still remain to be solved.

2.2 Fault accommodation

The ultimate objective of fault diagnosis is to properly accommodate the failures and maintain the system safety and reliability with the least human interference. In the hardware redundancy approach, fault accommodation process is completed simply by switching to the backup system or components. Typical fault accommodation schemes using model-based analytical redundancy can be divided into the following approaches:

1. If the faults are known as the parameter variation of a linear system, the so-called pseudo-inverse method is used to adjust the feedback gains for proper accommodation of the failures. It has been shown that this method is a special case of classical linear model-following control [11,12].
2. The faulty effects appear as the changes in model parameters, which can be identified on-line. The control law is reconfigured automatically based upon the identified parameters [13,14].
3. Linear-quadratic control methodology is used and the reconfiguration is achieved by choosing new values of the weighting matrices in the performance index to offset the effect of faults [16].
4. Compensation via additive input design for sensor failures and actuator failures [15,17].

5. If the system subject to faults can be modeled as a linear time invariant model with changed parameters, state-space based pole placement together with the system identification process or eigenstructure assignment can be used for the reconfigurable control [18-19,104-105].

Similar to, but different from adaptation, a learning control system concept is suggested by [20]. A learning system is the one that has ability to improve its performance in the future, based upon the information it learned from the past. According to [20], the goal of adaptation is to update behavior through time while the learning control system correlates past experiences with past situations and can recall and exploit those experiences. Possibly inspired by the advancing intelligent neural and fuzzy logic techniques, the learning system concept possesses promising potential in dealing with the ultimate autonomous and fault tolerant control problems. However, so far, the effort has only been focused on the linear control system.

Similar to fault diagnosis research work, traditional fault accommodation schemes are mainly based upon the powerful and well-developed linear design methodology to obtain the desired objectives. However, this is rarely the case in practice since all the systems are inherently nonlinear and the system dynamics under failure situations are more likely to be nonlinear and time varying. Although the control law reconfigurations are relatively easy to realize and solve under the assumption of linear situations, it actually limits the possible failure conditions and restricts the usefulness in practical situations. A series of research works that is devoted to more general failure cases is reported in [2,9,22] and also shown in Figure 2.1.

The failures are divided into two different groups, the anticipated faults and the unanticipated faults. The formers represent those well-known faults and the faults which happened in the past whose characteristics and effects are well characterized such that their corresponding signatures can be stored in the post failure-model bank for proper failure isolation and identification. The appropriate corresponding control actions can then be constructed based upon the well-understood knowledge of the failure dynamics. The latter refer to those failure situations whose signatures cannot be identified or recognized from the existing failure features. In the cases of the unanticipated faults, an on-line identification algorithm for the failure dynamics with control law reconfiguration strategy is required to properly control the system. The basic idea is outlined as follows:

Consider the following dynamic system with modeling uncertainty and unknown failure dynamics:

$$\dot{x} = \zeta(x) + G(x)[u + \eta(x,t) + \beta_i(t - T_i)f_i(x)] \tag{2.1}$$

where ζ and G represent the nominal plant dynamics, i.e. $\zeta + Gu$, η is the modeling uncertainty, and β_i, f_i represent the failure mode dynamics. The fault function, f_i, and the modeling uncertainty, η, are assumed to be independent of u and they are in the range space of G. This is known as a matching condition [27]. Assume that no fault occurs in the dynamic system during a specified initial time period of operation. Then, any differences between the system dynamics and its nominal model are due to modeling uncertainties. Therefore, neural networks can be used to learn the modeling uncertainties. It is assumed that, after the certain period of time, the differences belong to the failures. Neural networks are used to approximate the fault dynamics. The authors in [22] suggested that two neural networks can be used, one for improving the accuracy of the nominal model and one for fault monitoring. The control strategy is simply explained as follows.

Supposed that $u = \alpha(x)$ is the desired controller for the nominal model, which causes the nominal system to exhibit the desired behavior. In the event of system failures, a new augmented control law, $u' = \alpha(x) + \varphi(x,\hat{\theta})$, is required to accommodate the failure. $\varphi(x,\hat{\theta})$, corrective control law, is selected and adjusted based on available information ($\hat{\theta}$ represents the parameters of the neural network). Obviously, in the above case, φ can be chosen as $-\hat{\eta}-\chi$ where $\hat{\eta}$ and χ are the NN approximators for modeling uncertainty and failure mode dynamics, respectively. Once the approximation reaches a satisfactory result, the new control law should drive the fault system to the desired trajectory. The idea of using neural networks to approximate the unknown failure mode dynamics on-line makes the control methodology more flexible to accommodate the system failures.

However, observing Equation (2.1) closely, we can find that the unknown failure dynamics is modeled as an unknown function of past state variables only. In the cases of more general failure situations whose dynamics depends not only on the past state variables, but also on the past and current system inputs (control input), the control law reconstruction is substantially more difficult. As the authors pointed out in [2], one of the nonlinear control techniques available for such problems is that of feedback linearization whose idea is to transform the nonlinear system into a linear one through a change of coordinates and nonlinear feedback. Only when the feedback linearization is achievable, can the powerful linear control design techniques be used to attain the desired control objective. However, a system is feedback linearizable only under certain conditions [28].

Modern intelligent techniques involving artificial neural networks and fuzzy logic have made significant progress and attracted a great deal of attention from various research fields during the past decade. With the capability of *self-optimization* and *on-line adaptation*, these techniques have been successfully demonstrated and used in many real applications including pattern recognition, classification, automatic control, manufacturing, medical, telecommunications, banking, speech, oil and gas, etc [57]. It is a well-known fact that the traditional control design methods require mathematical description of the system to realize the system dynamics. Based upon those system features, the appropriate control law can then be designed to regulate the system behavior. However, it is substantially difficult to obtain a good mathematical model as the system gets larger, more complex, and/or subject to unanticipated faults. The artificial neural network has been proven to have the ability to approximate any piecewise continuous function given sufficient neurons in the hidden layer [56,57]. Naturally, it becomes a promising candidate to relax the complicated and difficult mathematical modeling process. Many intensive research activities have been devoted to explore the possibilities of using these techniques in dealing with the control problems involving unknown nonlinear systems. These techniques can be typically divided into three different approaches, namely indirect adaptive control, direct adaptive control, and multiple model techniques.

2.3 Indirect adaptive control

The idea of indirect adaptive control is simply shown in Figure 2.2. In the cases of unknown plant dynamics, an identification model is used to approximate the real plant dynamics by adjusting its parameters to reduce the error between its output and the output of the real plant. Once the identification process completes, the controller design is based upon the realization of the identification model toward the real system.

Narendra first showed the feasibility of using neural networks to identify and control an unknown nonlinear dynamic system in 1990 [29]. His idea is clearly shown in Figure 2.2. e_i is used to adjust the identification model parameters and the controller is designed based on the identification (estimated) model. The idea is best illustrated by the following example.

Plant : $y_p(k+1) = f(y_p(k), y_p(k-1)) + u(k)$ (2.2)

Reference model : $y_m(k+1) = 0.6y_m(k) + 0.2y_m(k-1) + r(k)$

Controller : $u(k) = -NF(y_p(k), y_p(k-1)) + 0.6y_p(k) + 0.2y_p(k-1) + r(k)$. (2.3)

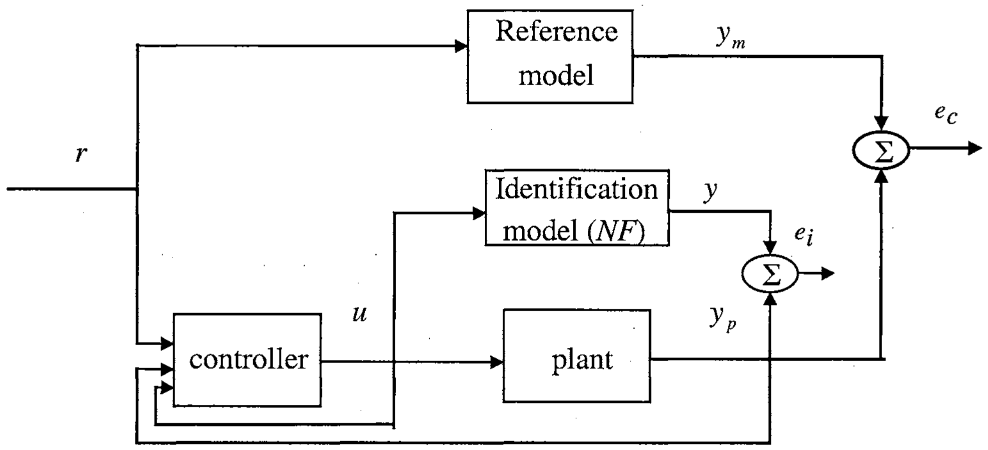


Figure 2.2 Indirect adaptive control

The nonlinear function, f , is assumed to be unknown. The only available information is that it is a function of current and past system outputs, $y_p(k)$ and $y_p(k-1)$ (i.e.

$f : \Re^2 \rightarrow \Re$). The approach is using a neural network to learn the nonlinear input-output mapping such that the unknown system dynamical behavior can be realized by a neural network model. Once the neural network identification model, *NF*, is trained to approximate the real nonlinear function *f*, the control signal generated by the controller will cancel the system non-linearity and replace it with the desired dynamics. However, this approach works properly only when the plant dynamics is not changing and the identification process has to be completed before the controller can effectively function. Otherwise, the incomplete identification model may mislead the controller and drive a stable system into an unstable situation.

The control problem in the above case is relatively easy to solve, since the system is actually affine in control (i.e. the system is linear in control input). The control signal can be easily computed by using Equation (2.3) and the control problem is actually reduced to properly identifying the unknown function (nonlinear mapping), f. Although a large class of dynamic systems can be represented by the "affine in control" relation, in the cases of more complex system dynamics, the more general cases have to be considered.

For the desired trajectories tracking control problem, the control design objective is to specify the appropriate control signal such that $y(k+1) = y_d(k+1)$ where $y_d(k+1)$ represents the desired output at time step $k+1$. By the implicit function theorem [30], the appropriate control input related to system (1.1) should be a function of past system outputs, past control inputs, and the desired output. It can be represented by Equation (2.4),

$$u(k) = G(y(k), y(k-1), \ldots, y(k-n+1), y_d(k+1), u(k-1), \ldots, u(k-n+1)), \qquad (2.4)$$

where $G : \Re^{2n} \rightarrow \Re$. The control problem is to determine the map G from the measured inputs and outputs as well as the desired output to the control signal. One approach is using a neural network to serve as the controller (i.e., using a neural network to approximate the function, G [35,36,72]). However, this technique is computationally expensive because the static back propagation algorithm [57] cannot be used directly to adjust the network parameters. A more complicated training algorithm such as real-time backpropagation or backpropagation through time is required for the parameter searching process [31-34].

Another approach is suggested by Narendra [37]. The idea is to linearize the general representation by using Taylor's series expansion to generate approximation models. The main feature of these models is that they are all "affine in control" such that the control input is easy to compute without requiring an additional neural network controller. The control problem is again simplified to be the identification problem of an unknown dynamic system. His approach is briefly shown as follows.

Consider the system (1.1) that can be described by Equation (2.5) using only the system input-output information,

$$y(k+d) = F(y(k), y(k-1),.., y(k-n+1), u(k), u(k-1),.., u(k-n+1)). \qquad (2.5)$$

Two approximation models, NARMA-L1 and NARMA-L2, can be generated by expanding the general representation (2.5) in Taylor's series around different operating points, respectively. The NARMA-L1 model is generated around the point,

$$(y(k), y(k-1),.., y(k-n+1), u(k)=0, u(k-1)=0,.., u(k-n+1)=0)$$

and the NARMA-L2 is generated around the point,

$$(y(k), y(k-1),.., y(k-n+1), u(k-1),.., u(k-n+1), u(k)=0).$$

The NARMA-L1 model is then represented by Equation (2.6),

$$y(k+d) = f[y(k), y(k-1),.., y(k-n+1)] + \sum_{i=0}^{n-1} g_i(y(k), y(k-1),.., y(k-n+1)]u(k-i), \tag{2.6}$$

and the NARMA-L2 model is described by Equation (2.7),

$$y(k+d) = \bar{f}(y(k),.., y(k-n+1), u(k-1),.., u(k-n+1)] + \bar{g}(y(k),.., y(k-n+1), u(k-1),.. \\ , u(k-n+1)]u(k). \tag{2.7}$$

If the NARMA-L1 model is used to approximate the system, $n+1$ networks are required to approximate the functions f and $g_i\,(i = 0,1,..,n-1)$ with n arguments in each function. If the NARMA-L2 model is used, only two networks are needed to approximate the system dynamics and each function has $2n-1$ inputs.

This approach uses approximation models to represent the real system dynamics based upon the Taylor series expansion up to the first order such that the control input can be easily computed as follows.

For NARMA-L1 model,

$$u(k) = \frac{y_d(k+1) - f[y(k), y(k-1),.., y(k-n+1)] - \sum_{i=1}^{n-1} g_i[y(k),.., y(k-n+1)]u(k-i)}{g_0[y(k),.., y(k-n+1)]}. \tag{2.8}$$

For NARMA-L2 model,

$$u(k) = \frac{y_d(k+1) - \bar{f}[y(k), y(k-1),.., y(k-n+1), u(k-1),.., u(k-n+1)]}{\bar{g}[y(k),.., y(k-n+1), u(k-1),.., u(k-n+1)]}. \tag{2.9}$$

However, using approximation models instead of the actual ones to reduce computational cost will degrade the control accuracy at the same time. Besides, when the system goes far away from the linearized points, the approximation models are no longer valid to represent the actual system dynamics. In the cases that only the system input-output data are measurable, this approach will also experience difficulty in the training process. It is believed that these indirect adaptive control techniques can reach the desired control objectives as long as the network training process converges to the desired accuracy, given a sufficient network structure and training time in the off-line situation.

Apparently, the idea of the indirect adaptive control is mainly based upon the identification of the unknown plant dynamics. The control problems are solved through the realization of the identification model toward the unknown plant dynamics and the control action can effectively regulate the system performance only after the completion of the identification process. Typical identification techniques include polynomials, rational functions, spline functions, multiplayer neural networks, radial-basis-function networks, and adaptive fuzzy systems. Other than these techniques, two novel intelligent system identification techniques will be briefly described in the following subsection.

2.3.1 Mixture of expert networks

Jordan and Jacobs proposed a novel identification structure using neural networks called Hierarchical Mixture of Experts [38]. Based on the principle of divide-and-conquer, it attacks a complex problem by dividing it into simpler problems whose local solutions can be combined to yield a solution to the complex problem. They proposed to solve the nonlinear supervised learning problems by dividing the input space into a nested set of regions and fitting simple surfaces to the data that fall in these regions. The structure of Mixture of Experts is shown in Figure 2.3.

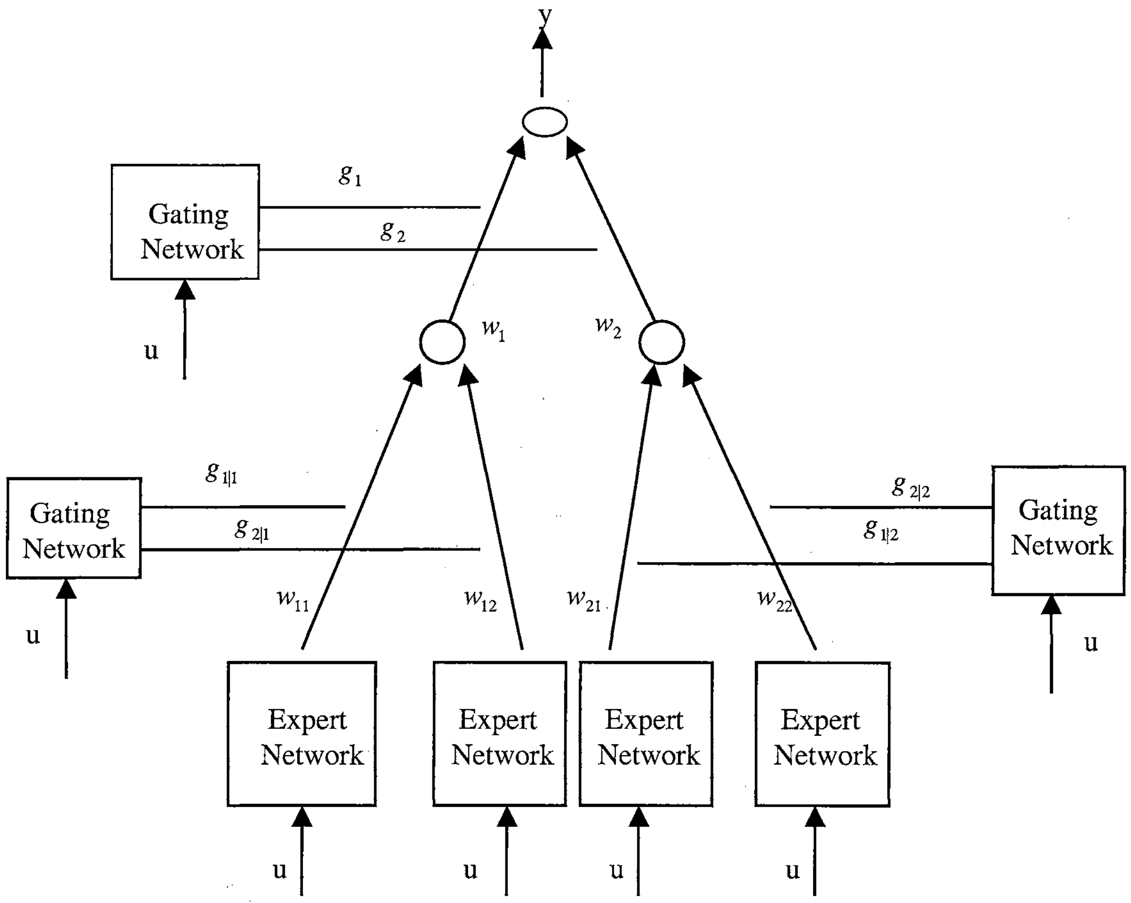


Figure 2.3 Hierarchical Mixture of Expert Networks

u and y represent the input and output, respectively. All of the expert networks in the tree are linear with a single output nonlinearity.

$$w_{ij} = f(U_{ij}\mathrm{u}), \tag{2.10}$$

where U_{ij} is a weight matrix and f is a fixed nonlinear continuous function. The upper level gating network is defined as follows:

$$\xi_i = v_i^T \mathrm{u} \text{ and } g_i = \frac{e^{\xi_i}}{\sum_k e^{\xi_k}}, \tag{2.11}$$

where v_i is a weight vector and ξ_i is an intermediate variable. g_i is positive and sum to one for each u. They can be interpreted as providing a "soft" partitioning of the input space. Similarly, the lower level gating networks are defined as follows:

$$\xi_{ij} = v_{ij}^T \mathrm{u} \text{ and } g_{j|i} = \frac{e^{\xi_{ij}}}{\sum_k e^{\xi_{ik}}}. \tag{2.12}$$

The output of the network is just the weighted sum of the lower level network outputs.

$$w_i = \sum_j g_{j|i} w_{ij} \text{ and } y = \sum_i g_i w_i . \tag{2.13}$$

The problem of training a Mixture of Experts can be treated as a maximum likelihood estimation problem. The training method is called EM (Expectation-Maximization) algorithm. It is an iterative approach to maximum likelihood estimation [38].

Comparing with backpropagation networks, using Hierarchical Mixture of Experts (HME) in the identification process for a dynamic system can be summarized as follows [38]:

1. No free parameter is required for training HME using EM algorithm while the backpropagation networks have some free parameters to adjust such as the learning rate and the momentum term.
2. Backpropagation networks usually produce lower error than the HME, although they sometimes have difficulties with the local minima.
3. Training HME is computationally expensive when the network structure is large.
4. EM algorithm assumes known output density function for training and the choice of density function is problem-dependent.

2.3.2 Neural fuzzy inference network

Juang and Lin proposed a self-constructing neural fuzzy inference network (SOFIN) with the capability of the on-line learning and self-pruning of the network structure for system identification [39]. The network are created and adapted as learning proceeds via simultaneous structure and parameter identification. It contains six layers and realizes a fuzzy model of the following form:

Rule i: **if** x_i **is** A_{i1} **and ...** x_n **is** A_{in} **then** y **is** $m_{0i} + a_{ji}x_j + ...$,

where A_{ij} is a fuzzy set, m_{0i} is the center of a symmetric membership function on y, and a_{ji} is a consequent parameter. The network structure is shown in Figure 2.4 and briefly introduced as follows.

Layer 1: Each node in this layer corresponds to one input variable and it transmits the input values to the next layer directly.

Layer 2: Each node in this layer corresponds to one linguistic label of one of the input variables in Layer 1. The membership value that specifies the degree to which an input variable belongs a fuzzy set is computed in this layer [39]. The operation performed in this layer is

$$f(u_{ij}^{(2)}) = -\frac{[u_i^{(2)} - m_{ij}]^2}{\sigma_{ij}^2} \text{ and } a^{(2)}(f) = e^f , \tag{2.14}$$

where $u_i^{(2)}$ is the ith input of this layer and is the same as the ith output of the first layer, m_{ij} and σ_{ij} are the center and the width of the Gaussian membership function of the jth term of the ith input variable x_i. $a^{(2)}(f)$ represents the output of the second layer. Please note that the number of fuzzy sets for each input variable is not necessarily identical in this network.

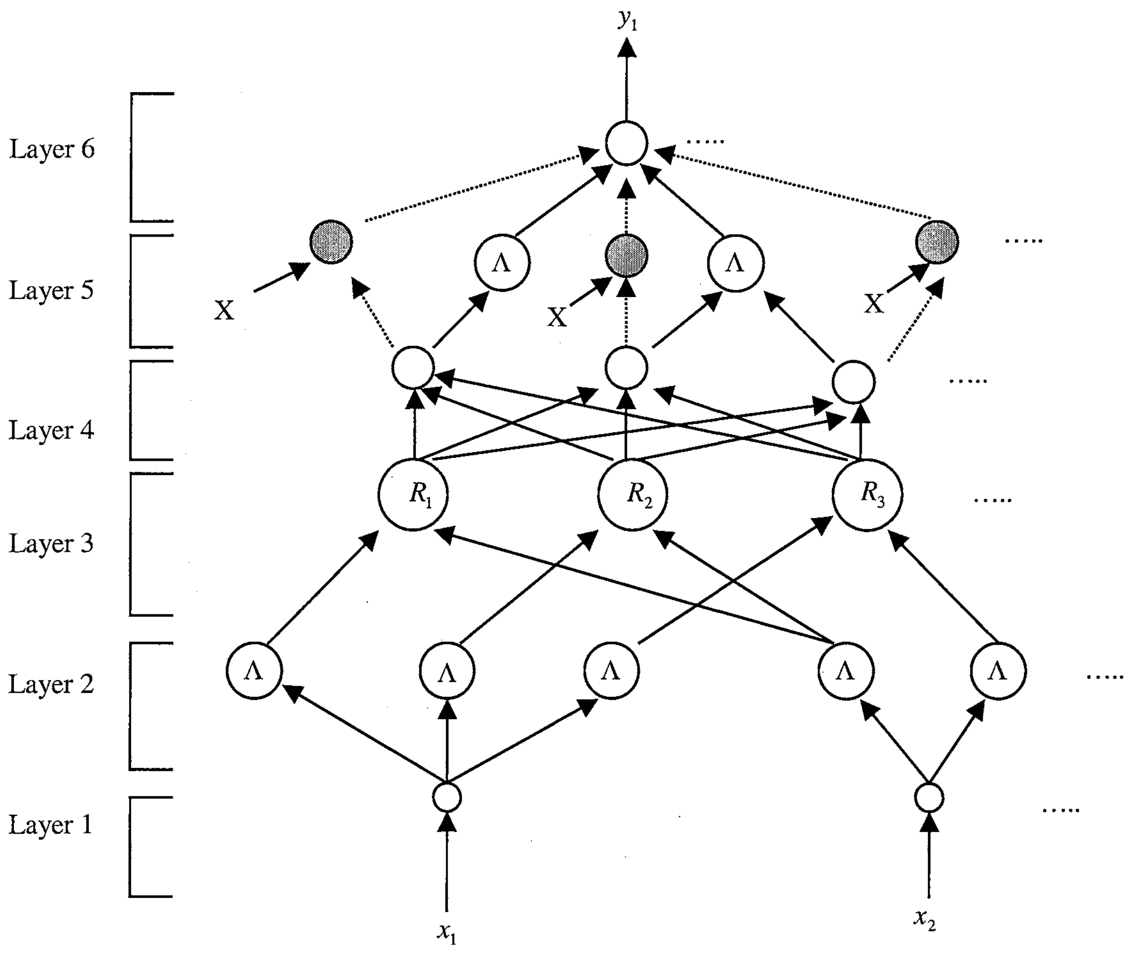


Figure 2.4 Structure of SOFIN

Layer 3: A node in this layer represents one fuzzy logic rule and performs precondition matching of a rule. The AND operation is used for each node in the Layer 3

$$f(u_i^{(3)}) = \prod_i u_i^{(3)} = e^{-[D_i(X-M_i)]^T[D_i(X-M_i)]} \text{ and } a^{(3)}(f) = f, \qquad (2.15)$$

where $D_i = diag(1/\sigma_{i1}, 1/\sigma_{i2}, ..., 1/\sigma_{in})$, $M_i = (m_{i1}, m_{i2}, ..., m_{in})^T$, and $X = (x_1, x_2, .., x_n)$. n is the number of Layer-2 nodes participating in the **IF** part of the rule. The output of a Layer-3 node represents the firing strength of the corresponding fuzzy rule.

Layer 4: The number of nodes in this layer is equal to that in Layer 3 and the firing strength computed in the Layer 3 is normalized in this layer by

$$f = \sum_i u_i^{(4)} \text{ and } a^{(4)}(f) = \frac{u_i^{(4)}}{f}, \tag{2.16}$$

Layer 5: This is the consequent layer. There are two types of nodes used in this layer. The note denoted by a blank circle is the essential node representing a fuzzy set of the output variable. Only the center of each Gaussian membership function is delivered to the next layer for the local mean of maximum defuzzification operation [43] and the width is used for output clustering only. The function of the blank node is

$$f = \sum_i u_i^{(5)} \text{ and } a^{(5)}(f) = f \cdot a_{0i}, \tag{2.17}$$

where $a_{0i} = m_{0i}$, the center of the Gaussian membership function. The shaded node is generated only when necessary. Each node in Layer 4 has its corresponding shaded node in Layer 5. One of the inputs to a shaded node is the output from Layer 4 and the others are the input variables from Layer 1. The shaded node function is

$$f = \sum_j a_{ji} x_j \text{ and } a^{(5)}(f) = f \cdot u_i^{(5)}, \tag{2.18}$$

where the summation is over the significant terms connected to the shaded node only and a_{ji} is the corresponding parameter. The whole function performed in Layer 5 is

$$a^{(5)}(f) = (\sum_j a_{ji}x_j + a_{0i})u_i^{(5)} . \tag{2.19}$$

Layer 6: Each node in this layer corresponds to one output variable. It combines all the actions suggested by Layer 5 and acts as a defuzzifier with

$$f(u_i^{(6)}) = \sum_i u_i^{(6)} \text{ and } a^{(6)}(f) = f . \tag{2.20}$$

Initially, there is no rule in the network. They are created from the incoming training data received by performing the following learning processes:

1) input/output space partitioning
2) construction of fuzzy rules
3) optimal consequent structure identification
4) parameter identification

For the details of each process and the training algorithm, please refer to [39] and the related methods [40-42].

2.4 Direct adaptive control

Unlike the indirect adaptive control techniques, the direct adaptive control methods intend to skip the complex modeling process and directly focus on solving the control problems. The idea of the direct adaptive control problem is shown in Figure 2.5. The control problem is solved by the effective control signal that reduces the system performance error (usually defined by the tracking error between the system output and the desired trajectory). Based upon the truth that tuning the parameters of the neural controller by the gradient descent type optimization algorithm requires only the Jacobian of the system model with respect to the control inputs and the searching of better parameters is along the negative gradient direction, the approach of the direct adaptive control is to either compute or estimate the plant Jacobian matrix with respect to the control inputs for MIMO (multiple-inputs multiple-outputs) system or the plant gradient with respect to the control input for SISO (single-input single-output) system.

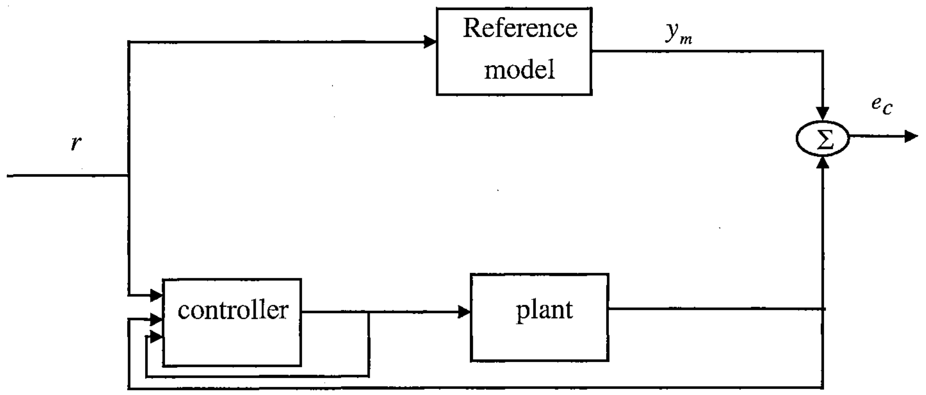


Figure 2.5 Direct adaptive control

Psaltis and Saerens suggested an approach that uses the sign of the Jacobian, instead of its real value in the training of the Neural Controller [35,36,44]. With the help of on-line estimating the sign of the plant Jacobian, the identification process and the NN identification model for the real system dynamics is not required for the control purpose, since the control input can be adjusted by using the sign of the plant Jacobian to reduce the tracking error.

The idea is using numerical on-line approximation by changing each input to the plant slightly and measuring the change at the output, or by comparing changes from the previous iterations. Taking a SISO nonlinear system as an example, the latter can be expressed as follows,

$$J(k+1) \approx \frac{y_p(k+1) - y_p(k)}{u(k) - u(k-1)} \text{ and } J(k+1) \text{ is the estimated gradient.} \tag{2.21}$$

For tracking a constant desired trajectory, Equation (2.21) may have the problem of division by zero. So, the following equation is suggested [44],

$$J(k+1) \approx sign[y_p(k+1) - y_p(k)] \times sign[u(k) - u(k-1)]. \qquad (2.22)$$

However, this approach may be sensitive to noise in the real applications. Also, tuning of the learning rate to compensate for the estimated plant Jacobian is a challenging job. Apparently, finding the plant Jacobian through the model is more accurate than just taking the sign of the plant Jacobian value. For on-line control implementation, the approach is only suitable for fixed or slowly varying plants [44].

Another interesting technique for plant Jacobian estimation is suggested by Spall [45-47]. Instead of using the finite difference approximation, an algorithm called SPSA (Simultaneous Perturbation Stochastic Approximation) is used to estimate the gradient of the objective function with respect to the parameters being optimized. The idea of the SPSA algorithm is described as follows.

Assume $L_k(\theta_k)$ is the differentiable cost function to be minimized by optimizing the parameter vector, θ_k. The subscript, k, represents the time step or the number of iterations. The objective is to find an optimal parameter vector, θ^*, satisfying Equation (2.23),

$$g(\theta^*) = \left.\frac{\partial L(\theta)}{\partial \theta}\right|_{\theta=\theta^*} = 0. \qquad (2.23)$$

Consider a stochastic approximation algorithm of the form,

$$\hat{\theta}_k = \hat{\theta}_{k-1} - a_k \hat{g}_k(\hat{\theta}_{k-1}), \qquad (2.24)$$

to estimate θ_k, where $\hat{g}_k$ is the estimate of real gradient g_k at kth iteration, a_k is a scalar gain sequence. The estimate gradient is computed at each iteration using Equation (2.25),

$$\hat{g}_k(\hat{\theta}_{k-1}) = \frac{\hat{L}_k^{(+)} - \hat{L}_k^{(-)}}{2c_k \Delta_k}, \tag{2.25}$$

where $\hat{L}_k^{(\pm)}$ are estimated (observed) values of $L_k(\hat{\theta}_{k-1} \pm c_k \Delta_k)$, c_k is another design gain sequence, and $\Delta_k = (\Delta_{k1}, \Delta_{k2}, ..., \Delta_{kp})^T$ is a random vector. Δ_{ki} is selected as an independent, bounded, symmetrically distributed zero mean random variable satisfying certain conditions [45]. A simple choice for each component of Δ_k is to use a Bernoulli ± 1 distribution with probability of $\frac{1}{2}$ for each ± 1 outcome [46]. After the evaluation of the estimated gradient, the parameters, $\hat{\theta}_k$, are updated using Equation (2.24) and the algorithm is terminated if there is little change in several successive iterations or the maximum allowable number of iterations has been reached.

Observing Equation (2.25), it is easy to find that only two measurements are required to compute the p estimated gradients at each time step. It is much more efficient comparing with the standard FDSA (Finite-Difference SA) algorithm [52,53] where $2p$ measurements are required. For fast changing systems, one-measurement form of the gradient approximation is suggested by [45] as shown below,

$$\hat{g}_k(\hat{\theta}_{k-1}) = \frac{\hat{L}_k^{(+)}}{c_k \Delta_k}. \tag{2.26}$$

In the high-noise environment, a variation on the gradient approximation is to average several gradient estimates with each estimate being made based upon an independent value of Δ_k. Another variation of the gradient approximation is to smooth the gradient approximation by using a weighted average of the previous and current gradient estimates. Based upon the same algorithm, some application-oriented research work has been reported [48-51].

The basic idea of how the SPSA algorithm works is by sending stochastic random testing signals into the system, computing the estimated gradients based upon observations of the system outputs, and adjusting the parameters according to the estimated gradients. The convergence property of the parameters has been studied and it has been shown that SPSA algorithm is theoretically more efficient than the FDSA algorithm [45]. However, it also shares the same deficiency as the FDSA algorithm, which is the question whether or not it is possible to reset the system from a given state to the previous state. To estimate the gradients, stochastic testing signals will first be sent to the system and, during the testing phase, no parameter update will take place until all the testing signals (i.e., the stochastic dither signals) go through the system and the testing results show in the system outputs. In on-line situations, this becomes an important issue since all the stochastic testing signals will go through the dynamic system and affect the system behavior later on.

2.5 Multiple model approaches

The basic idea of multiple model approach is to create different models and their corresponding controllers according to different operating conditions. With all the existing fixed models and their corresponding controllers stored in a database, an appropriate switching algorithm is used to choose (or switch to) the controller that corresponds to the model with the minimum identification error by monitoring the system and evaluating a certain criterion function [54-55]. The idea is shown in Figure 2.6 [58].

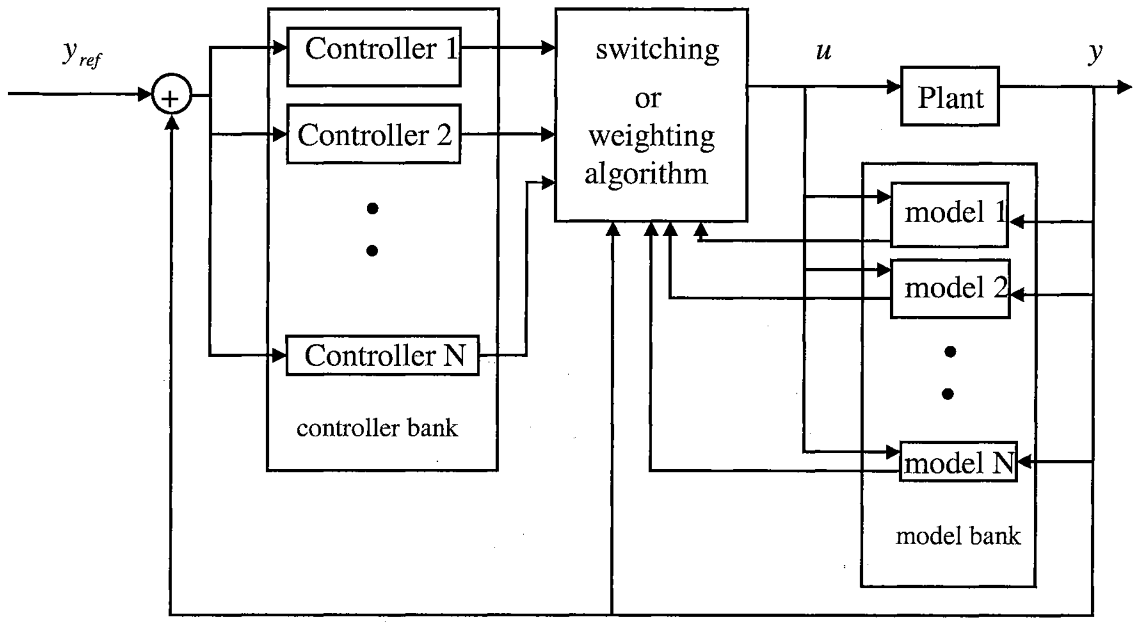


Figure 2.6 Basic multiple-model adaptive control structure

Obviously, switching to the controller corresponding to the model which best describes the current system dynamics can quickly improve the system transient performance. However, if all the models are fixed and the closest one is still far away from the current system dynamics, the performance cannot be guaranteed. Typical adaptive multiple model approaches proposed to overcome this drawback are briefly discussed as follows.

2.5.1 Model switching and tuning

Narendra combined multiple models and neural networks as the adaptive multiple model approach [59-61]. Apparently, the main reason of using fixed multiple models is to improve system transient performance. When the actual system dynamics doesn't fall into any one of the existing model dynamics, an appropriate adaptive model will be required to adapt itself to follow the actual system and adjust the corresponding controller at the same time. So, he suggested using an efficient combination of fixed and adaptive models. Right after the switching, an adaptive model should take over from the closest model and should be tuned on-line in order to keep reducing the difference between the model and the actual system so that the system performance is improved with time and the steady state error can finally reach an acceptable level. The idea is to use neural networks as the adaptive model for on-line adaptation and tuning.

However, so far, all the work is done for linear systems which are both theoretically and practically easy to be carried out for the proposed methodology. For the nonlinear systems, the problems of on-line identification and control are substantially difficult. Also, as mentioned in [60], the answers for the creation, modification and pruning of models, the acquisition of their sensitivity characteristics, and the generation of equivalence classes of models are remaining unsolved at present.

2.5.2 Multiple-model weighting

Another interesting approach was proposed by Rajagopal and Krishnamurthy [62]. Instead of the real switching algorithm, multiple model weighting is used based upon the assumption that the actual system output can be represented by a weighted sum of all the existing model (NN model) outputs. The idea is described as follows.

Consider a nonlinear plant described by

$$y_p(k+1) = F(y_p(k), y_p(k-1), .., y_p(k-p), u(k-1), .., u(k-q)) + G(y_p(k), y_p(k-1), .., y_p(k-p), u(k-1), u(k-2), .., u(k-q))u(k). \qquad (2.27)$$

The approach is to create N identification models I_i, $i = 1,2,..,N$, one for each one of the N environments that the system is expected to encounter during the operation, i.e. $I_i = NF_i(.) + NG_i(.)u(k)$. Please note that $(.)$ is used to represent the same arguments in Equation (2.27). The output of the actual plant, $y_p(k+1)$, can be approximated by the weighted sum of each NN identification model,

$$y_p(k+1) \approx \sum_i w_i(k)NF_i(.) + w_i(k)NG_i(.)u(k). \tag{2.28}$$

So, the control input can be generated as follows:

$$u(k) = \left[\phi^T \phi\right]^{-1} \phi^T \left[y_d(k+1) - \psi\right],\ \psi = \sum_i w_i(k)NF_i(.),\ \text{and}\ \phi = \sum_i w_i(k)NG_i(.), \tag{2.29}$$

where $y_d(k+1)$ is the desired output at time step $k+1$. Hence, the key point of finding the appropriate control input becomes how to decide the correct weight (contribution) of each existing NN model. The authors proposed using an evolutionary strategy as the model weight selection by minimizing the following cost function [62],

$$J(k) = \sum_{i=0}^{p} \beta^i \sum_{j=1}^{n} \left| \frac{y_{pj}(k-i) - y_{wj}(k-j)}{\varepsilon_j} \right|^2 + k1 \sum_{i=1}^{N} \left[w_i(k) - w_i(k-1)\right]^2 + k2 \left[1 - \left| \sum_{i=1}^{N} w_i(k) \right| \right]^2, \tag{2.30}$$

where $y_{wj}(k) = \sum_{i=1}^{N} w_i(k) I_{j,i}$, $I_{j,i}$ is the jth output of the ith identification model. $\varepsilon_j, k1, k2$ are design parameters, β is the forgetting factor that places a higher weight on the recent data, and p is the effective windows of past data points. Observing the cost function closely, we can find that the first term accounts for the estimation error, the second term ensures that the weight of each model doesn't change drastically during the optimization process, and the last term is to make sure that the sum of each weight is close to 1, which can be interpreted as adding stability guarantee to the whole system.

The switching between multiple models becomes smoother by adjusting the corresponding weight of each model. Using model weighting to replace the switching algorithm seems to have better results when the actual system is working under an environment that doesn't fall into the parameter space of any one of the models but is close to some or all of them.

2.5.3 Interactive multiple model (IMM)

Similar to the multiple-model weighting, an Interactive Multiple Model (IMM) FDD approach has been proposed in [90,106-107] where the occurrence or recovery of a failure in a dynamics system is modeled as a finite-state Markov chain with known transition probabilities. The idea can be simply described as follows.

Assume that a set of N models has been used to represent different failure situations,

$$\begin{aligned} x(k+1) &= (F(k)+\Delta F_j(k))x(k)+(G(k)+\Delta G_j(k))u(k)+\zeta_j(k) \\ &= F_j(k)x(k)+G_j(k)u(k)+\zeta_j(k) \end{aligned} \tag{2.31}$$

$$\begin{aligned} z(k+1) &= (H(k)+\Delta H_j(k))x(k)+\eta_j(k) \\ &= H_j(k)x(k)+\eta_j(k), \qquad\qquad j=1,2,\ldots,N. \end{aligned} \tag{2.32}$$

where $\Delta F_j(k)$, $\Delta G_j(k)$, and $\Delta H_j(k)\,(j=2,\ldots,N)$ represent the fault-induced changes in system components, actuator, and sensors, respectively. They are zero for $j=1$ which denotes the fault-free condition. The FDD and FTC problem then turns into determining the current model state from a sequence of noisy measurements, and to select the most appropriate control strategy from a set of pre-computed controllers to compensate the failure. The probability of a mode in effect plays a key role in determining the weights in the combination of state estimates and covariances for the overall state estimation [90]. Several different criterions of the fault detection and diagnosis scheme based upon the probabilities of system modes are suggested in [90]. The effectiveness of the IMM FDD approach has been demonstrated on linear systems with simple failures that can be represented by Equations (2.32)-(2.33). However, there is no solid theoretical result to support the successful failure accommodation especially for unanticipated failures in nonlinear format due to the complexity of the problem involved.

Generally speaking, among all existing intelligent control techniques, multiple model methods are probably, at present, the most efficient approach in dealing with the fault tolerant control problems involving *anticipated* faults since the anticipated failures, the corresponding model, and the control actions can be realized off-line. For unanticipated system failures in on-line situations, it is still an open question that remains to be solved.

CHAPTER III

THE ON-LINE FAULT ACCOMMODATION TECHNIQUE FOR UNANTICIPATED SYSTEM FAILURES

While most research attention has been focusing on fault detection and diagnosis, much less effort has been dedicated to general fault accommodation mainly because of the lack of well-developed control theory and techniques for general nonlinear systems. Existing fault accommodation techniques are mainly designed for either linear systems or certain classes of nonlinear systems under simple failure scenarios [2, 11-21]. However, for a dynamic system under unanticipated catastrophic system failures, it is not a reasonable approach to assume certain types of dynamic changes. In this chapter, the general on-line fault accommodation problems will be analyzed from both theoretical and realistic points of view. The main focus will be placed in developing an on-line fault accommodation technique for general nonlinear dynamic systems under general unanticipated catastrophic system failures.

3.1 Theoretical foundation and analysis

Without loss of the generality, we let $d = 1$ in Equation (1.4) and consider the SISO system to facilitate the analysis and derivation. Define a sliding surface function as shown in Equation (3.1),

$$S(k) = \frac{y_d(k) - y_d(k-1)}{\Delta t} - \frac{y(k) - y(k-1)}{\Delta t} + a(y_d(k) - y(k)), \qquad (3.1)$$

where $y_d(k)$ and $y(k)$ represent the desired system output and the actual system output at time step k, respectively. Δt represents the sampling period and $a \in R^+$ defines how fast the system output will converge to the desired output. The desired dynamics can then be described by setting the sliding surface function equal to zero (i.e., $S(k) = 0$) and it is a function of tracking error, $e(k) = y_d(k) - y(k)$. According to the discrete-time Lyapunov stability theory [77-80], if we choose $V(e(k)) = S^2(e(k))$ as the Lyapunov function candidate, the controller design objective becomes seeking the control input that will satisfy $S^2(k+1) < S^2(k)$ which is equivalently to say $[S(k+1) + S(k)][S(k+1) - S(k)] < 0$. (To simplify the notation, e will be eliminated from the remaining sections of this dissertation.) This is the same as satisfying the following inequalities

$$\begin{aligned} -S(k) < S(k+1) < S(k) \quad & when\ S(k) > 0, \\ S(k) < S(k+1) < -S(k) \quad & when\ S(k) < 0. \end{aligned} \qquad (3.2)$$

For $S(k) > 0$, plugging in Equation (3.1), we have

$$-S(k) < \frac{y_d(k+1) - y_d(k)}{\Delta t} - \frac{y(k+1) - y(k)}{\Delta t} + a(y_d(k+1) - y(k+1)) < S(k). \qquad (3.3)$$

Reorganizing the inequality, we get

$$-S(k) - \overline{Y}(k) < (-a - \frac{1}{\Delta t})y(k+1) < S(k) - \overline{Y}(k), \qquad (3.4)$$

where $\overline{Y}(k) = \frac{y_d(k+1) - y_d(k) + y(k)}{\Delta t} + ay_d(k+1)$. This can be further simplified as

$$(\bar{Y}(k)+S(k))(a+\frac{1}{\Delta t})^{-1} > y(k+1) > (\bar{Y}(k)-S(k))(a+\frac{1}{\Delta t})^{-1}. \tag{3.5}$$

In a similar manner for $S(k)<0$, we obtain

$$(\bar{Y}(k)-S(k))(a+\frac{1}{\Delta t})^{-1} > y(k+1) > (\bar{Y}(k)+S(k))(a+\frac{1}{\Delta t})^{-1}. \tag{3.6}$$

Notice that the left hand side and the right hand side of the inequalities (3.5)-(3.6) are known and can be computed at each time step. Thus, the on-line fault tolerant control problems become *finding the effective control signal that satisfies inequality* (3.5), *when* $S(k)>0$ *or* (3.6), *when* $S(k)<0$ *at every time step.*

Let $y(k+1)=\bar{\Theta}[\bar{y},\bar{u}]$, which represents the system dynamics under failures, where $\bar{y}$ and $\bar{u}$ represent the regression vectors of system outputs and inputs, recpectively. Based upon the implicit function theorem [30], the control law can be written as $u(k)=\bar{G}[\bar{y},\bar{Y}(a+\frac{1}{\Delta t})^{-1},\bar{u}\setminus\{u(k)\}]$ provided $\bar{G}$ exists (i.e., $\bar{u}\setminus\{u(k)\}$ denotes the set containing the regression vector of input excluding the current input, $u(k)$). Since the nonexistence of $\bar{G}$ corresponds to the cases where the system becomes uncontrolable under the failure situations, the existence problem becomes trivial. Unfortunately, the realization of $\bar{G}$ can not be provided without knowing the true structure of the system dynamics and the failure dynamics. Thus, the control law is not implementable in reality. However, through the modern intelligent techniques, the effective control input satisfying inequalities (3.5) or (3.6) can be estimated using optimization algorithms without a complete realization of $\bar{G}$. The systematic procedure is described as follows.

The unknown failure dynamics can be realized through an on-line estimator. The true system output, $y(k+1)$, can be approximated by the sum of the outputs from the nominal model and the on-line estimator as follows:

$$\begin{aligned} y(k+1) &= Ny(k+1) + fy(k+1) \\ &= N\hat{y}(k+1) + N\tilde{y}(k+1) + nfy(k+1) + nf\tilde{y}(k+1), \end{aligned} \tag{3.7}$$

$Ny(k+1)$: the output of the actual system,

$fy(k+1)$: the output of the failure dynamics,

$N\hat{y}(k+1)$: the output of the nominal model,

$N\tilde{y}(k+1)$: the remaining uncertainty between the nominal system and the nominal model,

$nfy(k+1)$: the output of the on-line estimator,

$nf\tilde{y}(k+1)$: the remaining uncertainty between the estimator and the failure dynamics,

and $Ny(k+1) = N\hat{y}(k+1) + N\tilde{y}(k+1)$; $fy(k+1) = nfy(k+1) + nf\tilde{y}(k+1)$.

Using Equation (3.7), the inequalities become:

for $S(k) > 0$,

$$(S(k) + \overline{Y}(k))(a + \frac{1}{\Delta t})^{-1} > N\hat{y}(k+1) + N\tilde{y}(k+1) + nfy(k+1) + nf\tilde{y}(k+1) > (\overline{Y}(k) - S(k))(a + \frac{1}{\Delta t})^{-1}, \tag{3.8}$$

for $S(k) < 0$,

$$(\overline{Y}(k) - S(k))(a + \frac{1}{\Delta t})^{-1} > N\hat{y}(k+1) + N\tilde{y}(k+1) + nfy(k+1) + nf\tilde{y}(k+1) > (\overline{Y}(k) + S(k))(a + \frac{1}{\Delta t})^{-1}. \tag{3.9}$$

Modern intelligent optimization techniques such as genetic algorithm, immune algorithm, simulated annealing, reinforcement learning, etc., have been exploited in a variety of areas and applications [96-103]. However, although the effectiveness in achieving successful optimization objectives has been demonstrated, most of them are applicable in off-line situations at present, due to the time-consuming iterative process. From the computational complexity point of view, the well-known and efficient gradient descent algorithm will be considered and used in the remaining of this dissertation because of its popularity and effectiveness in on-line applications. The optimization procedure is shown as follows.

The desired point at every time step is

$$Desire(k) = [(\bar{Y}(k) + S(k))(a + \frac{1}{\Delta t})^{-1} + (\bar{Y}(k) - S(k))(a + \frac{1}{\Delta t})^{-1}]/2$$

$$= \bar{Y}(k)(a + \frac{1}{\Delta t})^{-1}. \tag{3.10}$$

Define the error as

$$Error(k) = Desire(k) - N\hat{y}(k+1) - N\tilde{y}(k+1) - nfy(k+1) - nf\tilde{y}(k+1). \tag{3.11}$$

The effective control input can be searched based upon the gradient descent algorithm for square error

$$\frac{\partial Error(k)^2}{\partial u(k)} = 2Error(k)\frac{\partial Error(k)}{\partial u(k)}$$

$$= -2Error(k)[\frac{\partial N\hat{y}(k+1)}{\partial u(k)} + \frac{\partial N\tilde{y}(k+1)}{\partial u(k)} + \frac{\partial nfy(k+1)}{\partial u(k)} + \frac{\partial nf\tilde{y}(k+1)}{\partial u(k)}]. \tag{3.12}$$

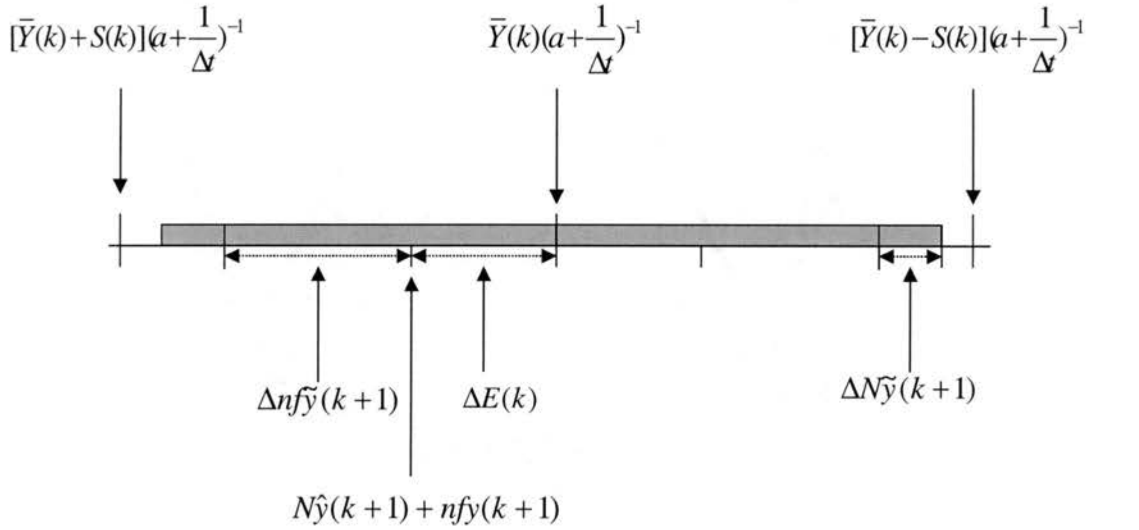
The resulting control input will be updated by

$$u(k)_{new} = u(k)_{old} - \alpha\frac{\partial Error(k)^2}{\partial u(k)}, \tag{3.13}$$

where α is the learning rate parameter. The searching procedure is repeated until inequalities (3.8) or (3.9) hold, the control input converges, or the maximum number of iterations is reached. Of course, the term, $nf\tilde{y}(k+1)$, the remaining uncertainty of the failure dynamics, and $N\tilde{y}(k+1)$, the remaining uncertainty of the nominal system, are unknown and the terms, $\frac{\partial nf\tilde{y}(k+1)}{\partial u(k)}$ and $\frac{\partial N\tilde{y}(k+1)}{\partial u(k)}$, cannot be computed either. So, the actual searching procedure is based upon the approximated values:

$$Er\hat{r}or(k) = Desire(k) - N\hat{y}(k+1) - nfy(k+1) \tag{3.14}$$

$$\frac{\partial Er\hat{r}or(k)^2}{\partial u(k)} = -2Er\hat{r}or(k)[\frac{\partial N\hat{y}(k+1)}{\partial u(k)} + \frac{\partial nfy(k+1)}{\partial u(k)}]. \tag{3.15}$$


$[\bar{Y}(k)+S(k)](a+\frac{1}{\Delta t})^{-1}$
$\bar{Y}(k)(a+\frac{1}{\Delta t})^{-1}$
$[\bar{Y}(k)-S(k)](a+\frac{1}{\Delta t})^{-1}$
$\Delta nf\tilde{y}(k+1)$
$\Delta E(k)$
$\Delta N\tilde{y}(k+1)$
$N\hat{y}(k+1)+nfy(k+1)$

$$\Delta E(k) = \sup_{\forall k>T_f}\{|\Delta Error(k)|\}\ ;\ \Delta nf\tilde{y}(k+1) = \sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\}\ ;\ \Delta N\tilde{y}(k+1) = \sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\}$$

▨ : bounded area

Figure 3.1 The bound of the sliding surface function

At every time step, the desired point, $\bar{Y}(k)(a+\frac{1}{\Delta t})^{-1}$, is computed, and the effective control signal is searched to ensure that the actual result is as close to the desired point as possible, through the realizations of the nominal system dynamics and the failure dynamics. Observing inequalities (3.8) and (3.9) closely, if $nf\tilde{y}(k+1)$ and

$N\tilde{y}(k+1)$, the remaining uncertainty of the failure dynamics and the nominal system, are bounded, combining these results with Equations (3.10)-(3.11), we can prove that the desired dynamics, sliding surface function, S, is also bounded. Figure 3.1 indicates how the S function is bounded by the upper bounds of the nominal model uncertainty, $\Delta N\tilde{y}(k+1)$, optimization error, $\Delta E(k)$, and the prediction error of the failure dynamics, $\Delta nf\tilde{y}(k+1)$. This can be proven given the following assumptions and detailed in Theorem 1.

Assumptions:

1. The nominal model, $N\hat{y}(k+1)$, is accurate and precise enough such that $N\tilde{y}(k+1)$, the remaining uncertainty of the nominal system, is bounded by $\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\}$, where T_f is the starting time step that the control input is reconfigured for proper failure accommodation. (Note that this constraint can be possibly relaxed since the accuracy of the nominal model can be achieved off-line provided sufficient time for the development.)
2. The remaining uncertainty of the failure dynamics, $nf\tilde{y}(k+1)$, is the residue resulting from the difference between the actual $fy(k+1)$ and the best estimation of the on-line estimator and it is bound by the least upper bound, $\sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\}$.
3. The error caused by the optimization algorithm is bounded by $\sup_{\forall k>T_f}\{|\Delta Error(k)|\}$.

Theorem 1: If the nominal system model is accurate and precise enough within the system working region such that $N\tilde{y}(k+1)$, the remaining uncertainty of the nominal system, is bounded by $\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\}$, and the on-line

estimator is accurate enough such that the remaining uncertainty of the failure dynamics, $n\tilde{f}y(k+1)$, is bounded by the least upper bound, $\sup_{\forall k>T_f}\{|n\tilde{f}y(k+1)|\}$, and $\Delta Error(k)$, the error after the searching effort of the optimisation algorithm, is finite (i.e., bounded by $\sup_{\forall k>T_f}\{|\Delta Error(k)|\}$), the system stability after time step T_f under arbitrary unanticipated system failures is guaranteed in on-line situation and the sliding surface function, S, defined by the system performance error is also bounded as follows:

$$\Sigma \le S(k+1) \le \Xi,$$

where

$$\Sigma = -\left[\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\} + \sup_{\forall k>T_f}\{|n\tilde{f}y(k+1)|\} + \sup_{\forall k>T_f}\{|\Delta Error(k)|\}\right](a+\frac{1}{\Delta t}), \text{ and}$$

$$\Xi = \left[\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\} + \sup_{\forall k>T_f}\{|n\tilde{f}y(k+1)|\} + \sup_{\forall k>T_f}\{|\Delta Error(k)|\}\right](a+\frac{1}{\Delta t}).$$

The sliding surface function, S, is defined as:

$$S(k) = \frac{y_d(k)-y_d(k-1)}{\Delta t} - \frac{y(k)-y(k-1)}{\Delta t} + a(y_d(k)-y(k)); \quad a \in R^+,$$

where $y_d(k)$ is the desired trajectory at time step k.

Proof:

Let $\Delta Error(k)$ represents the error after the searching effort of the optimization algorithm. Then,

$$\Delta Error(k) = Desire(k) - N\hat{y}(k+1) - nfy(k+1).$$

By Equation (3.10),

$$\bar{Y}(k)(a+\frac{1}{\Delta t})^{-1} - \Delta Error(k) = N\hat{y}(k+1) + nfy(k+1)\,. \tag{3.16}$$

For $S(k) > 0$:

Plugging in Equation (3.16) into inequality (3.8), we have

$$(\bar{Y}(k)+S(k))(a+\frac{1}{\Delta t})^{-1} > \bar{Y}(k)(a+\frac{1}{\Delta t})^{-1} - \Delta Error(k) + N\tilde{y}(k+1) + nf\tilde{y}(k+1) > (\bar{Y}(k)-S(k))(a+\frac{1}{\Delta t})^{-1}\,.$$

Simplifing the inequality, we get

$$S(k) > [N\tilde{y}(k+1) + nf\tilde{y}(k+1) - \Delta Error(k)](a+\frac{1}{\Delta t}) \text{ and}$$

$$-S(k) < -[N\tilde{y}(k+1) + nf\tilde{y}(k+1) - \Delta Error(k)](a+\frac{1}{\Delta t})\,. \tag{3.17}$$

Since $S(k) > 0$, $-S(k) < [\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\} + \sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\} + \sup_{\forall k>T_f}\{|\Delta Error(k)|\}](a+\frac{1}{\Delta t})$ is always true. By assumptions 1, 2, and 3, the following inequalities will hold for the worst condition,

$$S(k) > [\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\} + \sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\} + \sup_{\forall k>T_f}\{|\Delta Error(k)|\}](a+\frac{1}{\Delta t})\,. \tag{3.18}$$

Apparently, $[\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\} + \sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\} + \sup_{\forall k>T_f}\{|\Delta Error(k)|\}](a+\frac{1}{\Delta t}) = \inf_{\forall k>T_f}\{S(k)\}$, which is the greatest lower bound of $S(k)$ and $-[\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\} + \sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\} + \sup_{\forall k>T_f}\{|\Delta Error(k)|\}](a+\frac{1}{\Delta t}) = \sup_{\forall k>T_f}\{-S(k)\}$, which is the least upper bound of $-S(k)$, and since $S(k) > S(k+1)$ and $-S(k) < -S(k+1)$, the following inequalities will always hold

$$[\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\} + \sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\} + \sup_{\forall k>T_f}\{|\Delta Error(k)|\}](a+\frac{1}{\Delta t}) \geq S(k+1) \text{ and}$$

$$-[\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|\Delta Error(k)|\}](a+\frac{1}{\Delta t})\leq -S(k+1)\,, \quad (3.19)$$

for both situations, $S(k+1)>0$ and $S(k+1)<0$, which implies

$$\Sigma \leq S(k+1)\leq \Xi\,, \quad (3.20)$$

where

$$\Sigma = -[\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|\Delta Error(k)|\}](a+\frac{1}{\Delta t}) \text{ and}$$

$$\Xi = [\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|\Delta Error(k)|\}](a+\frac{1}{\Delta t})\,.$$

For $S(k)<0$:

Plugging in Equation (3.16) into inequality (3.9), we have

$$(\bar{Y}(k)-S(k))(a+\frac{1}{\Delta t})^{-1} > \bar{Y}(k)(a+\frac{1}{\Delta t})^{-1}-\Delta Error(k)+N\tilde{y}(k+1)+nf\tilde{y}(k+1) > (\bar{Y}(k)+S(k))(a+\frac{1}{\Delta t})^{-1}\,.$$

Simplifing the inequality, we get

$$-S(k) > [N\tilde{y}(k+1)+nf\tilde{y}(k+1)-\Delta Error(k)](a+\frac{1}{\Delta t})\,, \text{ and}$$

$$S(k) < [N\tilde{y}(k+1)+nf\tilde{y}(k+1)-\Delta Error(k)](a+\frac{1}{\Delta t})\,. \quad (3.21)$$

Since $S(k)<0$, $S(k)<[\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|\Delta Error(k)|\}](a+\frac{1}{\Delta t})$ is always true. By assumptions 1, 2, and 3, the following inequalities will hold for the worst condition,

$$-S(k) > [\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|\Delta Error(k)|\}](a+\frac{1}{\Delta t})\,. \quad (3.22)$$

Apparently, $[\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|\Delta Error(k)|\}](a+\frac{1}{\Delta t})=\inf_{\forall k>T_f}\{-S(k)\}$,

which is the greatest lower bound of $-S(k)$ and $-[\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|\Delta Error(k)|\}](a+\frac{1}{\Delta t})=\sup_{\forall k>T_f}\{S(k)\}$, which is the least upper bound of $S(k)$, and since $-S(k)>S(k+1)$ and $S(k)<-S(k+1)$, the following inequalities will always hold

$$[\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|\Delta Error(k)|\}](a+\frac{1}{\Delta t})\geq S(k+1),\text{ and}$$

$$-[\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|\Delta Error(k)|\}](a+\frac{1}{\Delta t})\leq -S(k+1), \qquad (3.23)$$

for both situations, $S(k+1)>0$ and $S(k+1)<0$, which implies

$$\Sigma\leq S(k+1)\leq\Xi, \qquad (3.24)$$

where

$$\Sigma=-[\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|\Delta Error(k)|\}](a+\frac{1}{\Delta t}),\text{ and}$$

$$\Xi=[\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\}+\sup_{\forall k>T_f}\{|\Delta Error(k)|\}](a+\frac{1}{\Delta t}).$$

We arrive at exactly the same result as Equation (3.20). Thus, the sliding surface function, S, is bounded by the value defined by the least upper bounds of the remaining uncertainty of the nominal system, the residue of the on-line estimator, and the error by the optimization algorithm.

Q.E.D.

The discrete-time Lyapunov stability theory indicates that the control problem can be solved as long as *the numerical value of the failure dynamics* is realizable at each time step, which is a measure of how far the fault drives the system dynamics away from the desired dynamics. Based upon the above theoretical analysis, the system under unexpected catastrophic failures can be stabilized and the performance can be recovered provided an effective on-line estimator for the unknown failure dynamics such that the necessary and sufficient conditions are satisfied (i.e., assumptions 1, 2, and 3). Moreover, since the on-line estimator is used to provide the approximated numerical value of the failure dynamics at each time step based upon the most recent measurements (i.e., the failure may be time-varying), no specific structure or dynamics is required for the estimator. In other words, only a static function approximator that approximates the most recent behavior of the failure is needed for the control purpose.

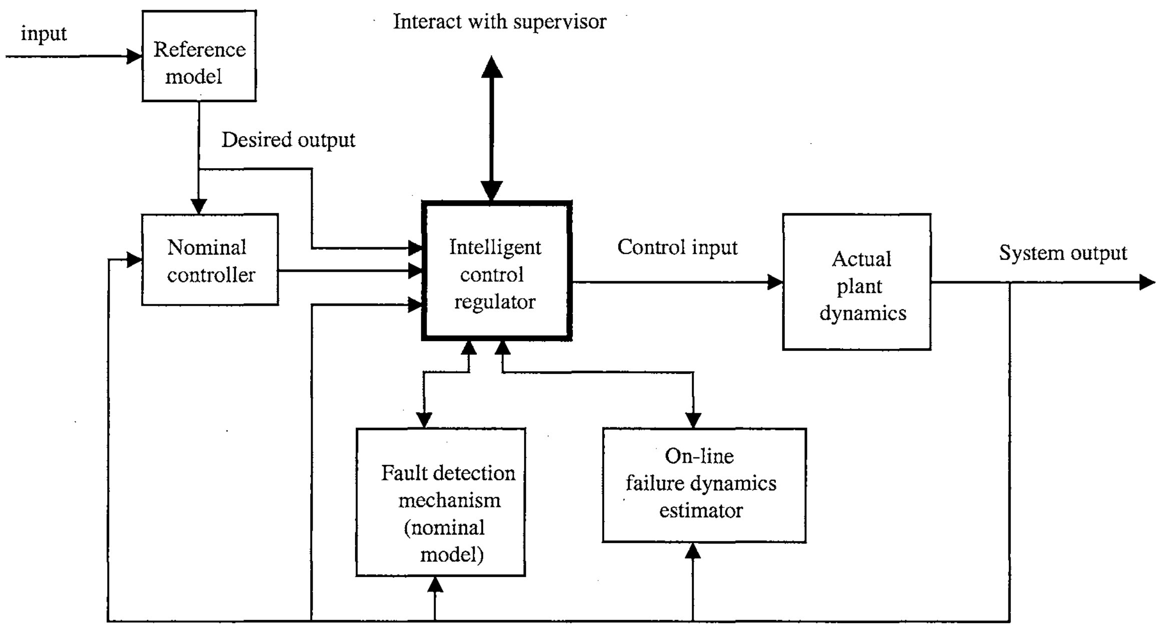


Figure 3.2 Basic framework of the intelligent on-line fault accommodation strategy

Figure 3.2 shows the basic framework of the intelligent control strategy for on-line control of the system that may be subject to the unanticipated catastrophic failures. In on-line situations, the nominal control signal will have to go through an intelligent control regulator before it can reach the system. The intelligent control regulator monitors and evaluates the system behavior at every time instant through a fault detection mechanism. During the normal operation mode that corresponds to the non-failure situation, the nominal control signal will be passed to the system to control its behavior. Once an abnormal system behavior is detected by the fault detection mechanism, an on-line estimator is initialized and starts estimating the unknown failure dynamics. When the learning process converges, the control law is reconfigured and computed by the regulator based upon the current knowledge of the failure dynamics provided by the on-line estimator. The intelligent control regulator also has to interact with the supervisor to accept higher priority commands, such as changes of the control objective or design parameters, and warn the supervisor for emergency shutdown of the system in cases that the unanticipated system failures are serious and the system is actually uncontrollable.

3.1.1 On-line learning of the failure dynamics

With the universal approximation capability for any piecewise continuous function [56,57], Artificial Neural Network becomes one of the most promising candidates for the on-line control problems of our interest. In this research work, neural network is exploited and used as the on-line estimator for the unknown failure dynamics. Some important features of the on-line learning using neural networks should first be addressed here. The structure of the on-line estimator needs to be decided (i.e., in neural networks, the number of hidden layers, number of neurons in each layer, and neuron transfer functions). It is known that neural networks are sensitive to the number of neurons in the hidden layers. Too few neurons can result in underfitting problems (poor approximation), while too many neurons may lead to overfitting problem, where all the training patterns are well fit, but the fitting curve may take wild oscillations between the training data points [65]. The criterion for stopping of the training process is another important issue in the real applications. If the mean square error of the estimator is forced to reach a very small value, the estimator may perform poorly for the novel input data slightly away from the training patterns. This is the well-known generalization problem. Besides, in the real applications, the training patterns may be contaminated by measurement noises since they are the measurements from sensors. The estimator may adjust itself to fit the noise instead of the real failure dynamics. Some methods proposed to improve these problems, such as early stopping criterion and generalization network training algorithms, may be useful to remedy these situations [65,66].

In the on-line situation, the number of input-output data for the training process becomes a very important design parameter. The system dynamics may keep changing because of progressing fault severity (i.e., the incipient fault, abrupt fault, and multiple faults). Apparently, using all input-output measurements to train the on-line estimator does not make too much sense since we may use invalid training patterns to mislead the estimator and it is also unrealistic for on-line applications. In other words, only finite and limited number of data sets should be considered as training patterns to adjust the parameters of the estimator. A reasonable approach is to use the most recent input-output measurements. A set, B, that contains the most recent measurements within a fixed length of a time-shifting data window is used to collect the training patterns,

$$B = \{(\underline{p}(m), \underline{t}(m)) | \underline{p} \in \Re^S ; \underline{t} \in \Re^T ; k - j + 1 \le m \le k\}, \tag{3.25}$$

where $\underline{p}(m)$ and $\underline{t}(m)$ are the network input vector and desired output vector at time step m, respectively. k is the current time step and j represents the length of the time-shifting data window which is a design parameter. This parameter has to be decided based upon the system computing capability, sampling rate, and the performance criteria. In additions, the maximum number of the effective control signal searching iterations is another important design parameter in real-time applications. It has to be within an allowable range according to the system computing capacity in the on-line situation. For gradient descent type of optimization algorithms, time-varying learning rates can be used to possibly reduce the searching time. A simple adjustment algorithm of the time-varying learning rate used in the on-line simulations in the next Chapter is shown as follows:

```
Define initial learning rate, $\alpha_{old}$, and min, max
Inside the searching process
{......................
 if $error(i) \geq error(i-1)$
   if $\alpha_{old} \geq \min$
     $\alpha_{new} = \alpha_{old} \times q$
     restore the last searching position
   end
 else
   if $\alpha_{old} \leq \max$
      $\alpha_{new} = \alpha_{old} \times g$
   end
 end
 check the stopping criterion
}
```

where i and $error(i)$ are the searching iteration number and the searching error at the ith iteration, respectively. $\min$ and $\max$ are the pre-specified minimum and maximum of the learning rates. g and q are pre-specified constant gains that satisfy the conditions, $0 < q < 1$ and $1 < g < 2$.

In the following sections, the on-line fault accommodation control problems under catastrophic system failures will be further divided into different cases according to prior knowledge of the nominal system and failure dynamics. Further analysis and discussion are provided case by case.

3.2 Further analysis for different cases

3.2.1 Case 1

Consider a dynamic system under catastrophic failures, which can be represented by Equation (3.26),

$$y(k+d) = f(y(k+d-1),\dots,y(k+d-p)) + g(y(k+d-1),\dots,y(k+d-q))u(k) + \sum_{i=1}^{n} \beta_i(k-T_i) f_i(y(k+d-1),\dots,y(k+d-p_i)), \tag{3.26}$$

where the nominal model is in "linear in control" format, $f_i(\cdot)$ represents the dynamics of the failure mode i. The prior knowledge shows that the failure dynamics is an explicit function of past system outputs. $\beta_i(\cdot)$ is the corresponding time-varying constant gain and n is the number of system failures. $f_i(\cdot)$, $\beta_i(\cdot)$, and n are assumed unknown. The nominal model is well developed and precise enough within the working regime.

In this case, the prior knowledge of both nominal model and failure dynamics provides substantial useful information for the on-line control problem. According to the analysis discussed in the last section, the effective on-line real-time control law can be easily found by plugging the nominal model and the estimated failure dynamics into the inequalities (3.8) and (3.9). Ignoring the uncertainties, when $S(k) > 0$, we have

$$(\bar{Y}(k) + S(k))(a + \frac{1}{\Delta t})^{-1} > f(\cdot) + g(\cdot)u(k) + F(k) > (\bar{Y}(k) - S(k))(a + \frac{1}{\Delta t})^{-1}, \tag{3.27}$$

which is equal to

$$(\bar{Y}(k) + S(k))(a + \frac{1}{\Delta t})^{-1} - f(\cdot) - F(k) > g(\cdot)u(k) > (\bar{Y}(k) - S(k))(a + \frac{1}{\Delta t})^{-1} - f(\cdot) - F(k). \tag{3.28}$$

When $S(k) < 0$, we have

$$(\bar{Y}(k) - S(k))(a + \frac{1}{\Delta t})^{-1} - f(\cdot) - F(k) > g(\cdot)u(k) > (\bar{Y}(k) + S(k))(a + \frac{1}{\Delta t})^{-1} - f(\cdot) - F(k). \tag{3.29}$$

where $(\cdot)$ is the short notation of the arguments for f and g in Equation (3.26) and $F(k)$ represents the numerical value of the failure dynamics at time step k in Equation (3.26). If $F(k)$ is known, the best desired control input is just the sum of the left hand side and right hand side of inequality (3.28) or (3.29) divided by $2g(\cdot)$ (in single input case). Since we are interested in the unanticipated failure situations, $F(k)$ is actually unknown. If we deploy an on-line estimator $NF(k)$ to approximate the unknown failure dynamics, $F(k)$ can be realized as follows:

$$F(k) = NF(k) + \Delta NF(k),$$

where $\Delta NF(k)$ denotes the residue between the on-line estimator and the actual failure dynamics. Plugging this realization into inequalities (3.28) and (3.29) and ignoring the unknown term, $\Delta NF(k)$, we have the first control law shown in Equation (3.30),

$$u(k) = [\bar{Y}(k)(a + \frac{1}{\Delta t})^{-1} - f(\cdot) - NF(k)]\frac{1}{g(\cdot)}. \tag{3.30}$$

If $\Delta NF(k)$ is bounded such that the following condition is satisfied,

$$\sup_{\forall k \geq T_f} \{|\Delta NF(k)|\} \leq \varpi, \varpi \in \Re^+, \tag{3.31}$$

where T_f is the starting time step when the on-line estimator starts compensating the control signal, it can be shown that the system stability under unanticipated system failures is guaranteed and the system performance error is also bounded (i.e., the sliding surface function, S, is bounded). This result can be proven and is summarized in the following theorem.

Theorem 2: If the remaining uncertainty of the failure dynamics is bounded such that Equation (3.31) is satisfied, using the intelligent on-line control law, Equation (3.30), the system stability after time step T_f is guaranteed and the performance error is also bounded for the system described by Equation (3.26) under unanticipated system failures.

Proof:

For $S(k) > 0$, plugging the control law, Equation (3.30), into inequality (3.28), we have

$$(\overline{Y}(k) + S(k))(a + \frac{1}{\Delta t})^{-1} > \overline{Y}(k)(a + \frac{1}{\Delta t})^{-1} + \Delta NF(k) > (\overline{Y}(k) - S(k))(a + \frac{1}{\Delta t})^{-1}. \tag{3.32}$$

Simplifying the inequality, we have

$$S(k) > [\Delta NF(k)](a + \frac{1}{\Delta t}) > -S(k). \tag{3.33}$$

So,

$$S(k) > [\Delta NF(k)](a + \frac{1}{\Delta t}) \text{ and } -S(k) < [\Delta NF(k)](a + \frac{1}{\Delta t}). \tag{3.34}$$

Since $S(k) > 0$ and $|\Delta NF(k)| \geq 0$, $-S(k) < \sup\limits_{\forall k \geq T_f}\{|\Delta NF(k)|\}(a + \frac{1}{\Delta t})$ is always true. If $\Delta NF(k)$ is bounded such that the condition in Equation (3.31) is satisfied, the following inequality will hold for the worst condition

$$S(k) > \sup_{\forall k \geq T_f}\{|\Delta NF(k)|\}(a + \frac{1}{\Delta t}). \tag{3.35}$$

Apparently, $\sup\limits_{\forall k \geq T_f}\{|\Delta NF(k)|\}(a + \frac{1}{\Delta t}) = \inf\limits_{\forall k \geq T_f}\{S(k)\}$, which is the greatest lower bound of $S(k)$ and $-\sup\limits_{\forall k \geq T_f}\{|\Delta NF(k)|\}(a + \frac{1}{\Delta t}) = \sup\limits_{\forall k \geq T_f}\{-S(k)\}$, which is the least upper bound of $-S(k)$. Since $S(k) > S(k+1)$ and $-S(k) < -S(k+1)$, the following inequality will always hold

$$S(k+1) \leq \sup_{\forall k \geq T_f}\{|\Delta NF(k)|\}(a + \frac{1}{\Delta t}) \text{ and } -\sup_{\forall k \geq T_f}\{|\Delta NF(k)|\}(a + \frac{1}{\Delta t}) \leq -S(k+1), \tag{3.36}$$

for both situations, $S(k+1) > 0$ and $S(k+1) < 0$, which implies

$$-\sup_{\forall k \geq T_f}\{|\Delta NF(k)|\}(a + \frac{1}{\Delta t}) \leq S(k+1) \leq \sup_{\forall k \geq T_f}\{|\Delta NF(k)|\}(a + \frac{1}{\Delta t}). \tag{3.37}$$

For $S(k) < 0$,

$$(\bar{Y}(k) - S(k))(a + \frac{1}{\Delta t})^{-1} > \bar{Y}(k)(a + \frac{1}{\Delta t})^{-1} + \Delta NF(k) > (\bar{Y}(k) + S(k))(a + \frac{1}{\Delta t})^{-1}. \tag{3.38}$$

Simplifying the inequality, we have

$$-S(k) > [\Delta NF(k)](a + \frac{1}{\Delta t}) > S(k). \tag{3.39}$$

So,

$$-S(k) > [\Delta NF(k)](a + \frac{1}{\Delta t}) \text{ and } S(k) < [\Delta NF(k)](a + \frac{1}{\Delta t}). \tag{3.40}$$

Since $S(k)<0$ and $|\Delta NF(k)|\geq 0$, $S(k)<\sup\limits_{\forall k>T_f}\{|\Delta NF(k)|\}(a+\frac{1}{\Delta t})$ is always true. If $\Delta NF(k)$ is bounded such that the condition in Equation (3.31) is satisfied, the following inequality will hold for the worst condition

$$-S(k)>\sup_{\forall k\geq T_f}\{|\Delta NF(k)|\}(a+\frac{1}{\Delta t}). \tag{3.41}$$

Apparently, $\sup\limits_{\forall k\geq T_f}\{|\Delta NF(k)|\}(a+\frac{1}{\Delta t})=\inf\limits_{\forall k\geq T_f}\{-S(k)\}$ is the greatest lower bound of $-S(k)$ and $-\sup\limits_{\forall k\geq T_f}\{|\Delta NF(k)|\}(a+\frac{1}{\Delta t})=\sup\limits_{\forall k\geq T_f}\{S(k)\}$ is the least upper bound of $S(k)$. Since $-S(k)>S(k+1)$ and $S(k)<-S(k+1)$, the following inequality will always hold

$$S(k+1)\leq\sup_{\forall k\geq T_f}\{|\Delta NF(k)|\}(a+\frac{1}{\Delta t})\text{ and }-\sup_{\forall k\geq T_f}\{|\Delta NF(k)|\}(a+\frac{1}{\Delta t})\leq -S(k+1), \tag{3.42}$$

for both situations, $S(k+1)>0$ and $S(k+1)<0$, which implies

$$-\sup_{\forall k\geq T_f}\{|\Delta NF(k)|\}(a+\frac{1}{\Delta t})\leq S(k+1)\leq\sup_{\forall k\geq T_f}\{|\Delta NF(k)|\}(a+\frac{1}{\Delta t}).$$

We arrive at the same result as inequality (3.37). Thus, the system stability is guaranteed and the sliding surface function, S, defined by the system performance error is also bounded.

Q.E.D.

3.2.1.1 Discrete-time Sliding Mode Control (DSMC) technique

One variation of the on-line control law can be derived based upon the discrete-time sliding mode control technique proposed in [67]. The basic principle behind Sliding Mode Control technique is called invariant set theory [68]. The approach is to define a sliding surface based on the desired dynamics and the control problem is solved by the appropriate control input to make the sliding surface an invariant set and attractive to all system trajectories using Lyapunov theory. The state variable trajectories starting at different initial conditions will eventually move close to the sliding surface. Once the trajectories move into the boundary layer, they will be forced to stay inside the boundary layer because the appropriate sliding mode control signal will drive the boundary layer to be an invariant set of the dynamic system.

The discrete-time sliding model control law can be derived for a given discrete-time nonlinear dynamic system with unmatched uncertainties. A controllable nonlinear system with uncertainty can be represented by Equation (3.43)

$$\dot{x}(t) = \hat{f}(x(t)) + g(x(t))u(t) + \Delta f(\cdot), \tag{3.43}$$

where $g(x(t)) = \hat{g}(x(t))\Delta g(\cdot)$ and both $\hat{f}(x(t))$ and $\hat{g}(x(t))$ are known functions of state vector $x(t)$. Uncertainties (Δf's) are bounded by constants and are explicit functions of the state vector, $x(t)$. The scalar uncertainty factor $\Delta g(\cdot)$ is also bounded such that $1/\mu \le \Delta g \le \mu$ for some $\mu \ge 1$. The discrete-time model is derived by using forward Euler approximation,

$$\dot{x}_i \approx \frac{x_i(k+1) - x_i(k)}{\Delta t}, \quad i=1,2,..,n, \tag{3.44}$$

where Δt is the sampling period. So, the discrete-time model can be written as

$$x(k+1) = x(k) + \Delta t\hat{f}(x(k)) + \Delta t g(x(k))u(k) + \Delta t\Delta f(\cdot). \tag{3.45}$$

Assume the uncertainty term $\Delta f(\cdot)$ is bounded by

$$|D(k)\Delta f(\cdot)| \le \gamma(k), \tag{3.46}$$

where $\gamma(k)$ is a constant vector and $D(k) = \left.\frac{\partial S}{\partial \tilde{x}}\right|_{\tilde{x}(k)} = \left[\frac{\partial S}{\partial \tilde{x}_1} \quad \frac{\partial S}{\partial \tilde{x}_2} \quad \cdots \quad \frac{\partial S}{\partial \tilde{x}_n}\right]_{\tilde{x}(k)}$.

$\tilde{x}(k) = x_d(k) - x(k)$, $x_d(k)$ is the desired trajectory at time step k, and S represents the sliding surface which is a function of $\tilde{x}(k)$ and is defined based upon the desired dynamics. The desired control law for the system (3.43) to guarantee the boundary layer being attractive is determined in [67] as

$$u(k) = \frac{\mu^2+1}{2\mu}\hat{p}(k) + \left[\left(\frac{\mu^2-1}{2\mu}\right)|\hat{p}(k)| + \frac{\mu K(k)}{D(k)\hat{g}(x(k))}\right]\text{sat}\left(\frac{S(k)}{\phi(k)}\right), \tag{3.47}$$

where

$$\hat{p}(k) = \frac{D(k)}{D(k)\hat{g}(x(k))}\left[-\hat{f}(x(k)) + \frac{\Delta x_d}{\Delta t}\right], \tag{3.48}$$

$$\phi(k) = \frac{\Delta t}{2}\{D(k)\hat{g}(k)[(3\mu^2+1)|\hat{p}(k)| + (\mu^2-1)\hat{p}(k)sat(S(k)/\phi(k-1))](\mu^2-1)/2\mu^2 +$$

$$(1+\mu^2)\eta(k) + 2\mu^2\varepsilon\}, \tag{3.49}$$

$$K(k) = \eta(k) + 2\varepsilon, \text{ and } \eta(k) = \gamma(k). \tag{3.50}$$

$\Delta x_d = x_d(k+1) - x_d(k)$, ε is an arbitrary positive constant, $\phi(k)$ is the boundary layer thickness, and the saturation function is defined to be

$$\text{sat}\left(\frac{S(k)}{\phi(k)}\right) = \begin{cases} +1 \ , & \text{if } S(k) > \phi(k) \\ \frac{S(k)}{\phi(k)} \ , & \text{if } |S(k)| \le \phi(k) \\ -1 \ , & \text{if } S(k) < -\phi(k) \end{cases}.$$

For the detail of the control law derivations, please refer to the Appendix in [67].

With the sliding mode control signal, the system trajectories will be guaranteed to converge and be confined inside the boundary layer. This research work [67] presents a new technique to design a robust discrete-time sliding mode controller off-line based upon discrete-time Lyapunov stability theory and guarantees the system stability with the known upper bound of the unmatched uncertainty.

3.2.1.2 The alternative corrective control law

Based upon the similar idea, $\Delta NF(\cdot)$ can be treated as the remaining uncertainty of the failure dynamics and the on-line approximation error can be used to estimate the upper bound of $\Delta NF(\cdot)$ in order to further improve the performance and increase the robustness property. Under different unknown failure modes, $u(k)$, the effective control law to accommodate the failures can be revised by adding a corrective control input, $u_2(k)$, such that $u(k) = u_1(k) + u_2(k)$ (i.e., $u_1(k)$ represents the nominal control law). The *corrective* sliding mode control law for the control problems of our interest is developed as follows:

$$u_2(k) = \frac{K(k)}{D(k)g(\cdot)} sat\left(\frac{S(k)}{\varphi(k)}\right) + U(k - T_c)\frac{-NF(\cdot)}{g(\cdot)}, \tag{3.51}$$

where T_c denotes the specific time step at which the difference of the sum square approximation error of the on-line estimator during two consecutive windows, Ω, is below a pre-specified threshold, δ, at the first time. This implies that the on-line learning result cannot be further significantly improved (i.e., $\Omega \leq \delta$). The boundary thickness and the controller gain are defined as

$$\phi(k) = \eta(k) + \varepsilon, \tag{3.52}$$

$$K(k) = \eta(k) + 2\varepsilon, \tag{3.53}$$

where $\eta(k)$ will be updated using the following equation

$$\eta_{new}(k) = \begin{cases} \sup\limits_{L}\left\{\left|D(k)\Delta\hat{f}(.)\right|\right\} = \sup\limits_{L}\left\{\left|D(k)\left(\sum_{i=1}^{n}\beta_i(\cdot)f_i(\cdot) - NF(\cdot)\right)\right|\right\}, \ if\ \Omega \le \delta \\ \eta_{old}\ , \text{otherwise} \end{cases}, \quad (3.54)$$

and

$$D(k) \approx \frac{S(k) - S(k-1)}{\tilde{y}(k) - \tilde{y}(k-1)} \quad ; \quad \tilde{y}(k) = y_d(k) - y(k)\,, \quad (3.55)$$

where $g(\cdot)$ and $\beta_i(\cdot)$ represent $g(y(k+d-1),..,y(k+d-q))$ and $\beta_i(k-T_i)$, respectively, and $y_d(k)$ is the desired system output at time step k. The first term on the right hand side of Equation (3.51) can be obtained by setting $\mu = 1$ in Equation (3.47) and ignoring the nominal controller part and the second term is the corrective control signal used to compensate the nominal controller. Equation (3.52) is obtained similarly by setting $\mu = 1$ in Equation (3.49) and Equation (3.53) is the same as that in Equation (3.50). The boundary layer thickness is now redefined by the least upper bound of the remaining uncertainty, on-line identification error, as shown in Equations (3.52) and (3.54). $NF(\cdot)$ denotes the on-line estimator which tracks $\sum_{i=1}^{n}\beta_i(\cdot)f_i(\cdot)$ on-line and the identification error is defined as $\Delta\hat{f}(\cdot) = \sum_{i=1}^{n}\beta_i(\cdot)f_i(\cdot) - NF(\cdot)$ which is the remaining uncertainty of the failure dynamics. The design parameter l represents a time period such that the least upper bound of the identification error is evaluated every time period, $L = [k-l, k]$. Equations (3.52)-(3.54) state that both the boundary layer thickness and the controller gain are automatically estimated and adjusted on-line by the estimator to further reduce control error. The on-line learning result is monitored and evaluated by the regulator using the following criterion:

$$SSAE0=\sum_{k=k_0}^{k_0+l-1}(fy(k)-nfy(k))^2\,,$$

$$SSAE1=\sum_{k=k_0+l}^{k_0+2l-1}(fy(k)-nfy(k))^2\,, \tag{3.56}$$

$$\Omega=\left|SSAE1-SSAE0\right|\,,$$

where $SSAE0$ and $SSAE1$ stand for the sum square approximation errors of the on-line estimator during two consecutive windows, $nfy(k)$ and $fy(k)$ are the output of the estimator and the difference between the measurement and the output of the nominal model at time step k, respectively. The certain threshold value, δ, is defined such that once Ω is less than or equal to the threshold value, the on-line estimation result is considered to be accurate enough and both the identification result and error can be used to further estimate a new least upper bound for the remaining uncertainty.

Both on-line control laws are derived based upon discrete-time Lyapunov stability theory. Their stability constraints remain the same as shown in Theorem 2. The alternative corrective control law is different from the first one (i.e., Equation (3.30)) by estimating the upper bound of $\Delta NF(\cdot)$ and redefining the boundary layer thickness on-line to provide more robustness property because of the attraction of the boundary layer. The price is more computations and design parameters to implement while the first control law is simple and straightforward. Another important point that should be mentioned here is that by adding $U(k-T_c)$ in the second term of the right hand side in Equation (3.51) to delay the compensation of the nominal control law, it is assumed that the system under nominal control law will not lose the stability before time step T_c. In other words, Theorem 2 is not effective to describe the system stability during this time period for the second control law because of the delay of compensation. However, later in the simulation, this delay results in less sensitivity in the selection of the learning rate in the learning process and a better transient performance.

The intelligent on-line fault tolerant control scheme for case 1 can be summarized as follows:

Off-line stage:

Step 1. Obtain the nominal model, design the nominal controller, and test the performance of the controller with selected criterions (i.e., for example, mean square control errors, sum square errors). Decide the range within which the system is working under the nominal condition (i.e., decide the fault detection threshold value) based upon the testing results. A simple but computationally cost-effective fault detection method used in the on-line simulations in Chapter 4 is shown as follows:

$$\psi = \frac{1}{\omega}\sum_{k=k_0}^{k_0+\omega-1}(y_d(k) - y(k))^2, \qquad (3.57)$$
$$\psi > \lambda \text{, } \textit{failure alarm.}$$

On-line stage:

Step 2. (For the alternative corrective control law only): Set the *initial* upper bound, η_0, for the unknown failure mode dynamics. Usually, the physical limitations of the system are useful information for deciding the upper bounds. Decide the threshold value, δ, for the convergence criterion of the on-line estimator and the design parameter, l.

Step 3. Keep monitoring the system behavior after the system is working and compare it with the nominal model response to decide if a fault has occurred. If the system is still in the nominal condition range, nothing has to be done (i.e., evaluating Equation (3.57) and comparing with the fault detection threshold value, λ).

Step 4. If a fault is detected: initialize the on-line estimator to learn the failure mode dynamics by using the difference between the actual measured system output and the output of the nominal model as the desired target for on-line training.

Step 5. For the first control law:

Use the output of the estimator and control law in Equation (3.30) to compensate for the system failures.

For the alternative corrective control law:

Add the corrective control signal (i.e., $u(k) = u_1(k) + u_2(k)$, Equations (3.51)-(3.55)). Evaluate Ω at every time period l (i.e., Equation (3.56)). If $\Omega \le \delta$, set T_c equal to the current time step, k, and adjust upper bound of the remaining uncertainty (Note that T_c is set once and only once when the condition, $\Omega \le \delta$, satisfies the first time.)

Step 6. Collect the next training pattern from the measurement, keep training the estimator, and observing the identification error of the estimator. Go back to the ***Step 5*** for the control process.

3.2.2 Case 2

Consider a dynamic system under catastrophic failures, which can be represented by Equation (3.58).

$$y(k+d) = f(y(k+d-1),\dots,y(k+d-p)) + g(y(k-d-1),\dots,y(k+d-q))u(k)$$
$$+\sum_{i=1}^{n} \beta_i(k-T_i) f_i(y(k+d-1),\dots,y(k+d-p_i),u(k),\dots,u(k-q_i)) . \quad (3.58)$$

This case corresponds to the situation that nominal system dynamics can be derived mathematically and it is in "linear in control" format. However, except for the nominal system dynamics, there is not much prior information for the multiple-failure dynamics available. It can only be described by a general function that depends upon the past system outputs, past control inputs, and the current control input. According to the analysis discussed in the last section, we need to determine the effective control signal to satisfy inequality (3.8) or (3.9) at each time step. In other words, the following inequalities have to be satisfied (i.e., ignore the remaining uncertainties),

$$(\bar{Y}(k)+S(k))(a+\frac{1}{\Delta t})^{-1} > N\hat{y}(k+1)+nfy(k+1) > (\bar{Y}(k)-S(k))(a+\frac{1}{\Delta t})^{-1}$$

if $S(k)>0$, or

$$(\bar{Y}(k)-S(k))(a+\frac{1}{\Delta t})^{-1} > N\hat{y}(k+1)+nfy(k+1) > (\bar{Y}(k)+S(k))(a+\frac{1}{\Delta t})^{-1}$$

if $S(k)<0$,

which is equivalent to

$$(\bar{Y}(k)+S(k))(a+\frac{1}{\Delta t})^{-1}-f(\cdot) > g(\cdot)u(k)+nfy(k+1) > (\bar{Y}(k)-S(k))(a+\frac{1}{\Delta t})^{-1}-f(\cdot)$$

if $S(k)>0$, or

$$(\bar{Y}(k)-S(k))(a+\frac{1}{\Delta t})^{-1}-f(\cdot) > g(\cdot)u(k)+nfy(k+1) > (\bar{Y}(k)+S(k))(a+\frac{1}{\Delta t})^{-1}-f(\cdot),$$

, if $S(k)<0$,

where $N\hat{y}(k+1) \cong f(y(k+d-1),\ldots,y(k+d-p))+g(y(k-d-1),\ldots,y(k+d-q))u(k)$,

and

$$nfy(k+1)+n\tilde{f}y(k+1)=\sum_{i=1}^{n}\beta_i(k-T_i)f_i(y(k+d-1),\ldots,y(k+d-p_i),u(k),\ldots,u(k-q_i)).$$

The Jacobian of the nominal model with respect to the current control signal is clearly equal to $g(y(k+d-1),\ldots,y(k+d-q))$ and the Jacobian of the actual system dynamics with respect to the current control input is the sum of $g(y(k+d-1),\ldots,y(k+d-q))$ and the Jacobian of the failure dynamics with respect to the current control signal. Since the failure dynamics is realized through an on-line estimator, the latter can be approximated and computed through the estimator (i.e., computing $\frac{\partial nfy(k+1)}{\partial u(k)}$). However, the fact that the failure dynamics depends explicitly upon the current control input makes the on-line control problems more complicated. System stability and performance are directly related to the accuracy of the on-line estimator (i.e., the necessary condition in assumption 2). In the on-line situation, the computational time becomes a critical issue for both learning of the unknown dynamics and searching of the effective control input. Few training patterns from measurements may not be good enough to represent the real failure dynamics such that the on-line estimator may perform poorly for the searching of the effective control signal. However, as mentioned before, only a limited number of training patterns are allowed to use from both reality and accuracy point of views. Under this condition, the estimator may represent the failure dynamics only locally inside the domain of the limited training patterns and its confined neighborhood. Once the searching of the effective control actions moves far enough away from the domain of the training patterns, the extrapolation problems of the on-line estimator may emerge and degrade the system performance (i.e., possible violation of the necessary condition 2). This is the major difficulty to guarantee the system on-line stability under unanticipated catastrophic failures in real applications.

3.2.3 Case 3

Consider a nominal system under catastrophic failures, which can be represented by Equation (3.59),

$$y(k+d) = f(y(k+d-1),\ldots,y(k+d-p),u(k),\ldots,u(k-q))$$
$$+\sum_{i=1}^{n}\beta_i(k-T_i)f_i(y(k+d-1),\ldots,y(k+d-p_i)). \qquad (3.59)$$

This case corresponds to the situations where the mathematical format of the nominal system is not 'linear in control' or not easy to be derived. Nominal system dynamics is represented by a general function that explicitly depends upon the past system outputs, past control inputs, and current control input. Under these conditions, the nominal model may be realized by off-line modeling techniques (i.e., neural network or neural fuzzy network, etc.) and the nominal controller is also designed by the similar techniques [31-34]. Some prior information of the failure dynamics is available. It is known that the failure dynamics is only an explicit function of system past outputs, which makes this problem relatively easier to solve. According to inequalities (3.8) and (3.9), the control problem becomes to satisfy the following inequalities (i.e., ignore the remaining uncertainties),

$$(\bar{Y}(k)+S(k))(a+\frac{1}{\Delta t})^{-1}-nfy(k+1) > N\hat{y}(k+1) > (\bar{Y}(k)-S(k))(a+\frac{1}{\Delta t})^{-1}-nfy(k+1),$$

if $S(k)>0$, or

$$(\bar{Y}(k)-S(k))(a+\frac{1}{\Delta t})^{-1}-nfy(k+1) > N\hat{y}(k+1) > (\bar{Y}(k)+S(k))(a+\frac{1}{\Delta t})^{-1}-nfy(k+1),$$

if $S(k)<0$.

Because the failure dynamics does not depend explicitly on the current control input, the real plant Jacobian with respect to the current control input is exactly the same as the nominal plant Jacobian which we have good confidience with. This is because the nominal model can be obtained off-line and the accuracy within the desired system working range can be developed. At every time step, the effective control input to reduce the absolute value of the sliding surface function is searched based upon the negative gradient direction of the nominal model with respect to the current control input. Both sides of the inequalities are fixed during the whole searching process. In other words, the extrapolation problem caused by the on-line estimator because of the availability of only the partial information of the unknown failure dynamics at each time step in on-line situations does not exist during the searching process in this problem. Similar to the first case where the effective control input can be obtained by solving the inequalities, according to Theorem 1, since the first necessary condition can be relaxed by off-line manipulative effort, the system stability under catasrophic failures in this case can be guaranteed and the system performance error is also bounded as long as the necessary condition 2 is satisfied by the on-line estimator and the third condition is provided by the optimization algorithm.

3.2.4 Case 4

Consider a nominal system under catastrophic failures, which can be represented by Equation (3.60),

$$\begin{aligned} y(k+d) &= f(y(k+d-1),\dots,y(k+d-p),u(k),\dots,u(k-q)) \\ &\quad + \sum_{i=1}^{n} \beta_i (k-T_i) f_i (y(k+d-1),\dots,y(k+d-p_i),u(k),\dots,u(k-q_i)) . \end{aligned} \tag{3.60}$$

This is the most general form covering all of the above cases. Similar to case 3, the mathematical representation of the nominal system dynamics is not 'linear in control'. A general function is used to describe the system dynamics and the prior information of the failure dynamics is not available either, such that the failure dynamics is represented by a general function as well. To deal with the on-line fault accommodation control problems in this case, the inequalities (3.8) or (3.9) have to be solved in the most general way,

$$(\overline{Y}(k)+S(k))(a+\frac{1}{\Delta t})^{-1} > N\hat{y}(k+1)+nfy(k+1) > (\overline{Y}(k)-S(k))(a+\frac{1}{\Delta t})^{-1},$$

if $S(k)>0$, or

$$(\overline{Y}(k)-S(k))(a+\frac{1}{\Delta t})^{-1} > N\hat{y}(k+1)+nfy(k+1) > (\overline{Y}(k)+S(k))(a+\frac{1}{\Delta t})^{-1},$$

if $S(k)<0$,

where $N\hat{y}(k+1) \cong f(y(k+d-1),\ldots,y(k+d-p),u(k),\ldots,u(k-q))$, and

$$nfy(k+1)+nf\tilde{y}(k+1)=\sum_{i=1}^{n}\beta_i(k-T_i)f_i(y(k+d-1),\ldots,y(k+d-p_i),u(k),\ldots,u(k-q_i)).$$

The effective control signals satisfying the above inequalities have to be searched at each time step through the nominal model and the on-line estimator. Similar to case 2, the major difficulty of guaranteeing system on-line safety may arise when the extrapolation problem emerges through the on-line estimator in the optimization process.

The general strategy of the on-line fault accommodation technique for unanticipated catastrophic system failures is summarized in the following steps:

Initial off-line stage:

1. Obtain the nominal model and achieve its accuracy with the help of off-line modeling techniques.

2. Design the nominal controller with the required accuracy based upon the nominal model.
3. Test both nominal model and controller, decide the fault detection threshold value, λ, and ω for the fault detection scheme based upon expected measurement noises, modeling error, and testing results of system behavior under nominal controller.
4. Choose the design parameters (maximum number of searching iterations, the structure of the on-line estimator, stopping criterion of the training process, and the length of the time-shifting data window, j) according to the system computing capability, sampling rate, and the performance requirement.

On-line stage:

5. Keep monitoring the system performance once the system starts working.
6. If system abnormal behavior is detected according to the fault detection mechanism, initialize the on-line estimator for learning process.
7. Collect training patterns (the difference between the output of the nominal model and the measurement), adjust the parameters of the on-line estimator, and check the stopping criterion.
8. Search the effective control signal to satisfy inequality (3.8) or (3.9).
9. Repeat step 8 until the inequality holds, maximum number of iteration reaches, or the control input converges.
10. Go back to step 7 and repeat for the control process.

CHAPTER IV

SIMULATION STUDY OF THE ON-LINE FAULT ACCOMMODATION TECHNIQUE FOR UNANTICIPATED SYSTEM FAILURES

4.1 CASE 1

4.1.1 Single failure case

Consider the following SISO nominal plant characterized by a NARMA model

$$y(k+1) = y(k) - \frac{\Delta t}{m}(cy(k)^3 + k_1 y(k-1)) + \frac{\Delta t}{m}u(k), \tag{4.1}$$

where $y(k+1)$ and $u(k)$ represent the system output and control input at time step $k+1$ and k, respectively. Δt is the sampling period and c, m, and k_1 are the system parameters which are assumed to be known. Under unexpected failure modes, the system is represented by the following equations

$$y(k+1) = y(k) - \frac{\Delta t}{m}(cy(k)^3 + k_1 y(k-1)) + \frac{\Delta t}{m}u(k) + \beta(k-T)f(y(k), y(k-1)), \tag{4.2}$$

where $f(y(k), y(k-1)) = b\sin(y(k) \times y(k-1))$ is assumed to be unknown and b denotes an unknown constant gain. Two possible fault scenarios will be considered:

abrupt fault: $\beta(k-T) = U(k-T)$, and

incipient fault: $\beta(k-T) = (1 - e^{-\alpha(k-T)})U(k-T)$, where $\Delta t = 0.01$, $c = 5$, $k_1 = 100$, $m = 1$, $T = 100$, and $\alpha = 0.05$. No information is available about the failure mode dynamics. In this simulation, b is chosen to be 0.5. In order to track the desired trajectory, $y_d(k+1)$, the nominal control input, $u_1(k)$, for Equation (4.1) is chosen as

$$u_1(k) = \frac{m}{\Delta t}(-(y(k) - \frac{\Delta t}{m}(cy(k)^3 + k_1 y(k-1))) + y_d(k+1)). \tag{4.3}$$

The desired trajectory was generated by the following reference model:

reference input: $r(k) = 0.2\sin(\frac{k\pi}{100})$,

desired output: $y_d(k+1) = 0.6y_d(k) + 0.2y_d(k-1) + r(k)$, and

the sliding surface, S, is selected as

$$S(k) = \frac{y_d(k) - y_d(k-1)}{\Delta t} - \frac{y(k) - y(k-1)}{\Delta t} + 10(y_d(k) - y(k)). \tag{4.4}$$

The design parameters of the proposed intelligent control scheme for the simulations are selected as follows:

$\omega = 5$, and $\lambda = 10^{-5}$.

For the alternative corrective control law only, we choose

$\eta_0 = 0.5$, $\delta = 0.001$, and $l = 10$.

4.1.1.1 Abrupt fault case

Figure 4.1 shows the actual system output and the desired output within the total simulation time steps, 2,000, when the nominal controller is applied alone. As seen, the system performance degrades and a large deviation from the desired trajectory starts after the time step reaches 100, when the system suddenly experiences an abrupt fault. Figure

4.2 shows the system response when the proposed intelligent control scheme is applied with the alternative corrective control law. As clearly shown, the system performance is greatly improved. The controller successfully drives the output of the unknown faulty system back to the desired trajectory with a small range of error bounded by the estimated uncertainty. The fault is actually detected by the control regulator at time step 115. The control signal is adjusted by adding the corrective control signal, $u_2(k)$, and a Multi-Layer Perceptron (MLP) neural network with two-input neurons, 30 neurons in the first and second layer, and one-output neuron (2-30-30-1) is initialized and learns the unknown failure mode dynamics on-line by using the static back-propagation algorithm in a non-batch mode [29,34,57]. At time step, 180, the on-line identification error converges. The control input is tuned again by using the output of the NN identification model and the new least upper bound for the remaining uncertainty is estimated by using the converged identification error to further reduce the control error.

Figure 4.3 shows the actual control error (difference between the actual output and the desired output) at each time step. Figure 4.4 is the on-line NN identification error plot. Note that there is no identification error defined before time step 115 because the learning of the neural network is initialized right after a fault is detected. Figure 4.5 shows the actual S function and the estimated boundary layer thickness. As shown, the intelligent controller adjusts the boundary layer thickness on-line, which represents the upper bound of the remaining uncertainty in order to improve the system performance and to reduce control effort. Because of the sliding mode control signal, the S function will be confined within the boundary layer, as shown in Figure 4.5. Figure 4.6 shows the actual control signal at each time step. Figure 4.7 shows the system response when the proposed intelligent control scheme is applied with the first control law as given in Equation (3.30). The same neural network on-line estimator with the same structure is used for the first control law and the learning rate for the on-line training is set to be 0.05. Comparing this response with Figure 4.2, we find that both control laws successfully drive the system output back to the desired trajectory within a small range of error. Although the system response under the first control law seems to have relatively larger error than that under the alternative corrective control law, the response under the first control law is much smoother while the system output under the alternative corrective control law seems to oscillate around the desired trajectory. The reason is obviously from the fact that the alternative corrective control law is based upon the discrete-time sliding mode control technique and it is the nature of the sliding mode control to drive the system output bouncing around the desired dynamics within the boundary layer. Figure 4.8 is the corresponding S function plot at each time step under the first control law.

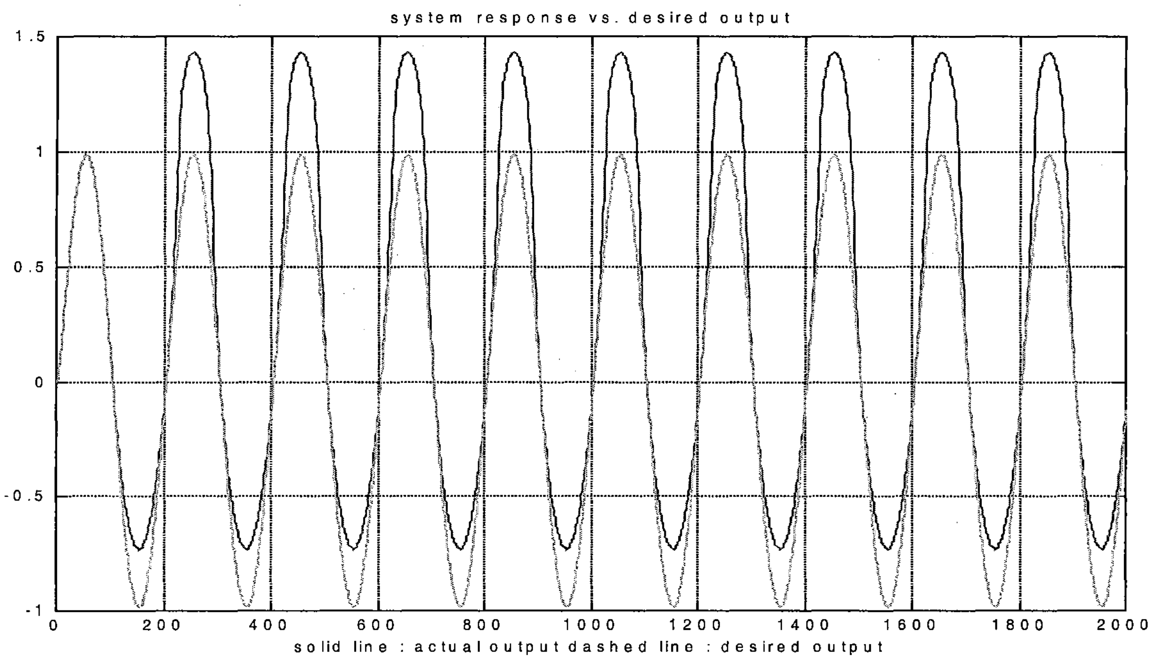


Figure 4.1 System response vs. desired output with nominal controller only (single abrupt fault case; case 1)

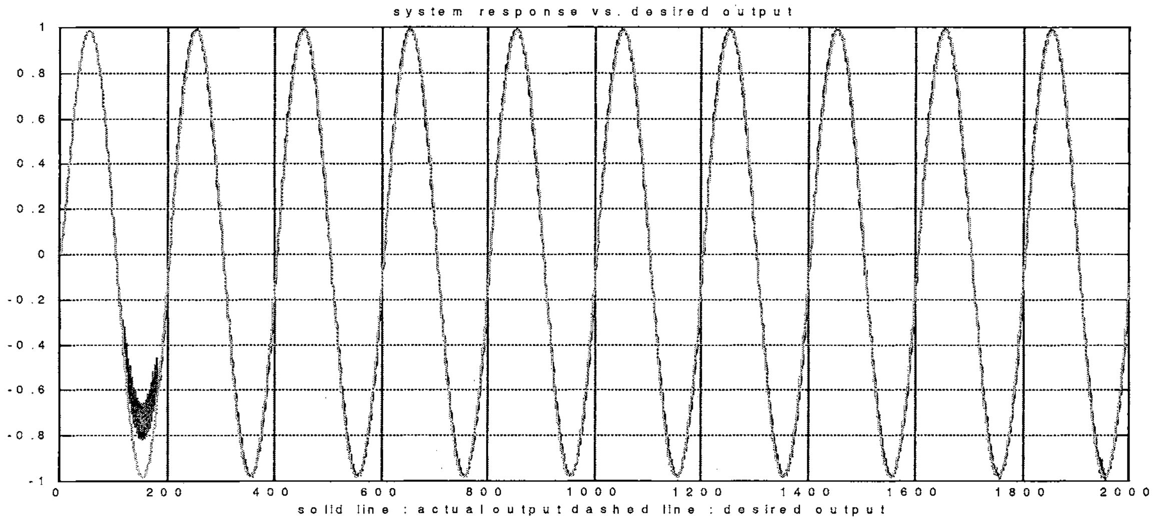


Figure 4.2 System response vs. desired output with the alternative corrective control law (single abrupt fault case; case 1)

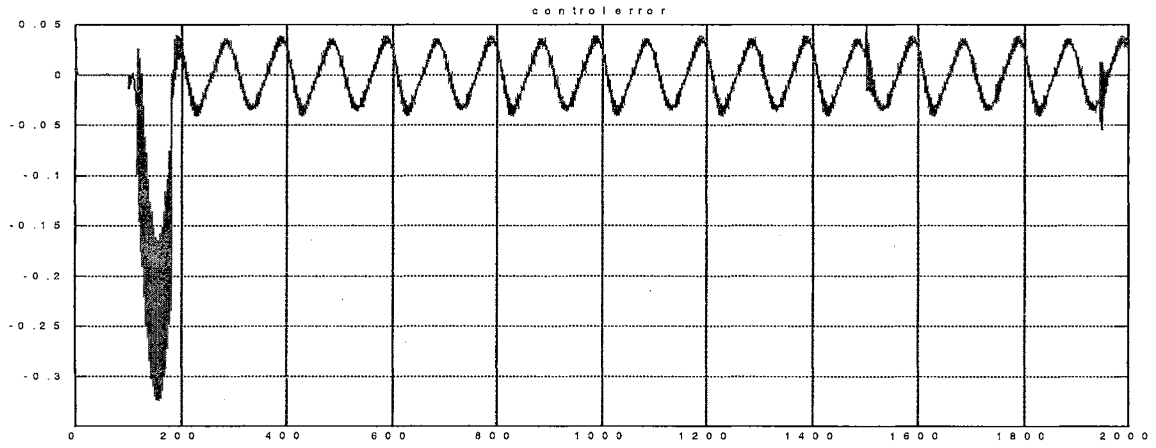


Figure 4.3 Control error with the alternative corrective control law (single abrupt fault case; case 1)

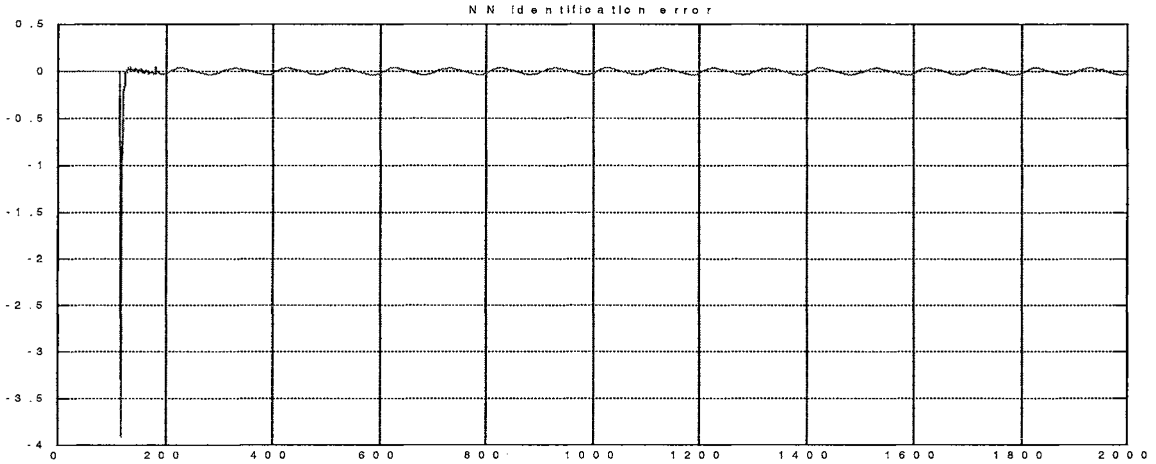


Figure 4.4 NN model identification error with the alternative corrective control law (single abrupt fault case; case 1)

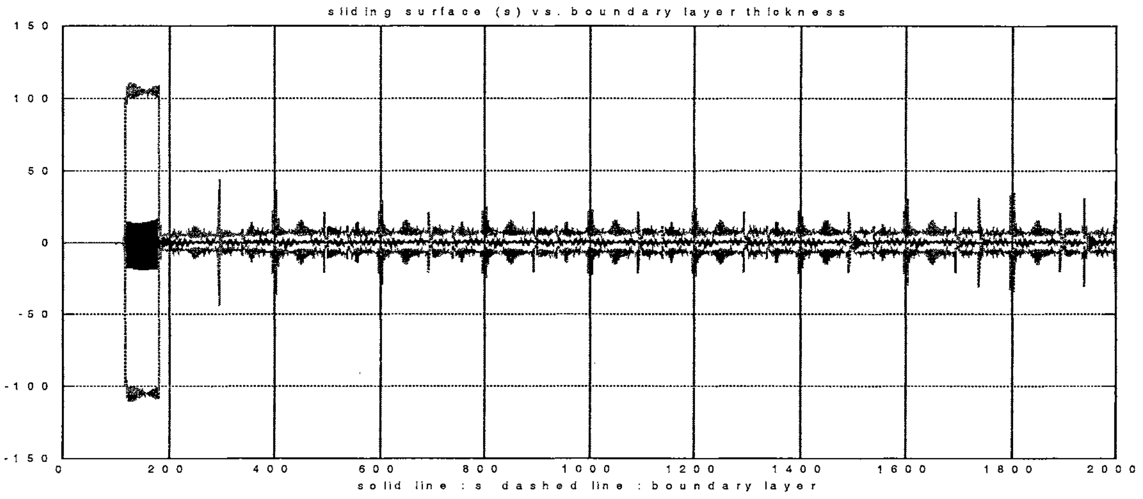


Figure 4.5 S function vs. on-line estimated boundary layer with the alternative corrective control law (single abrupt fault case; case 1)

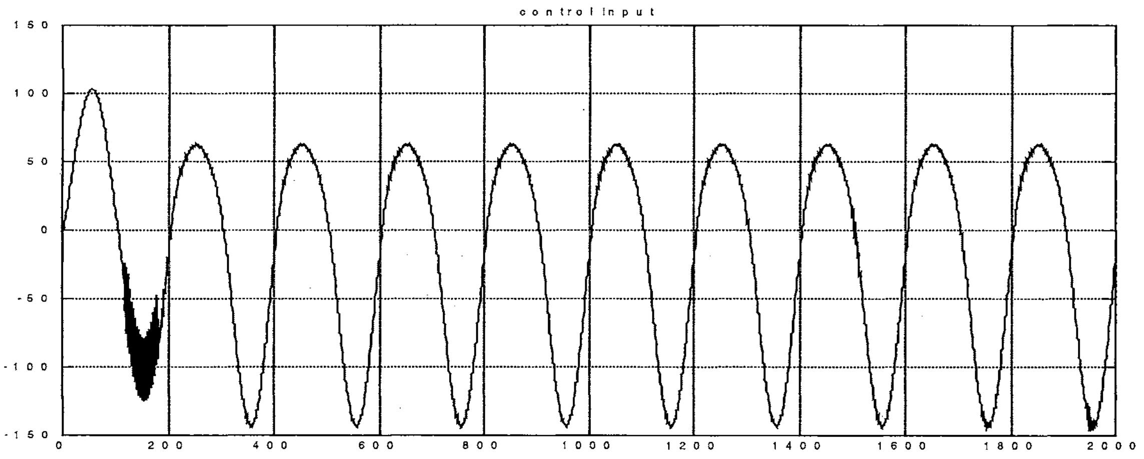


Figure 4.6 Actual control input with the alternative corrective control law (single abrupt fault case; case 1)

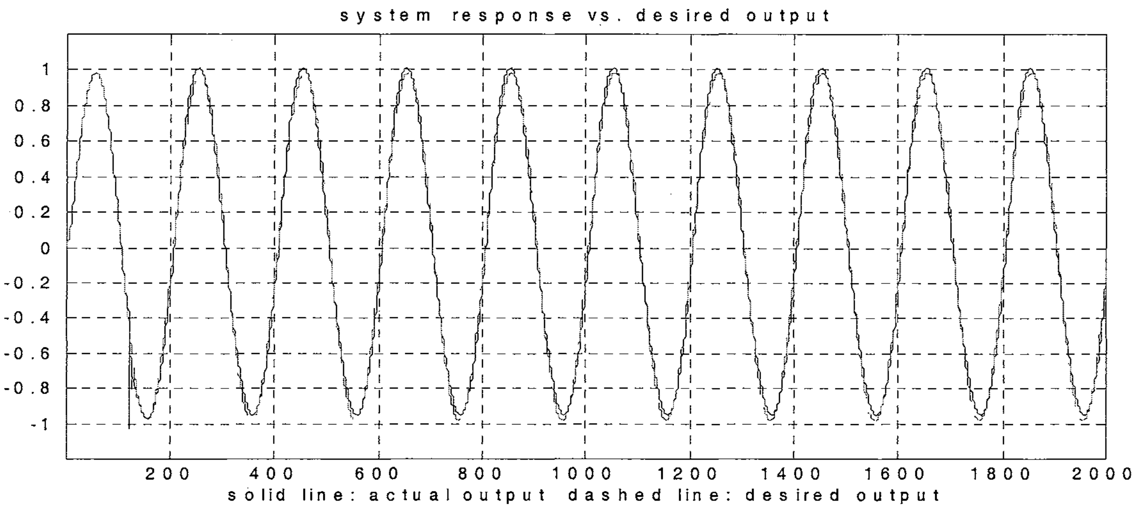


Figure 4.7 System response vs. desired output with the first control law (single abrupt fault case; case 1)

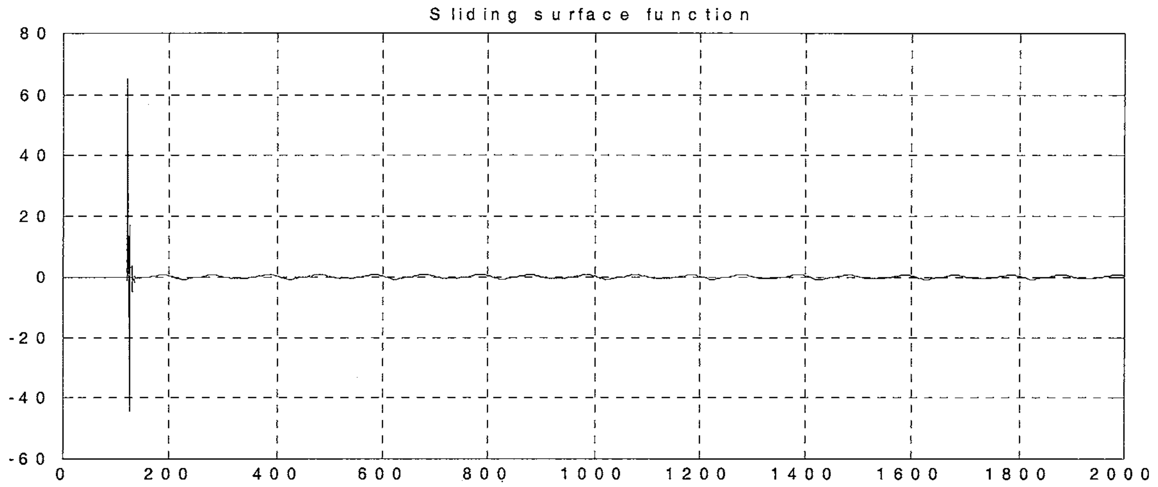


Figure 4.8 S function with the first control law (single abrupt fault case; case 1)

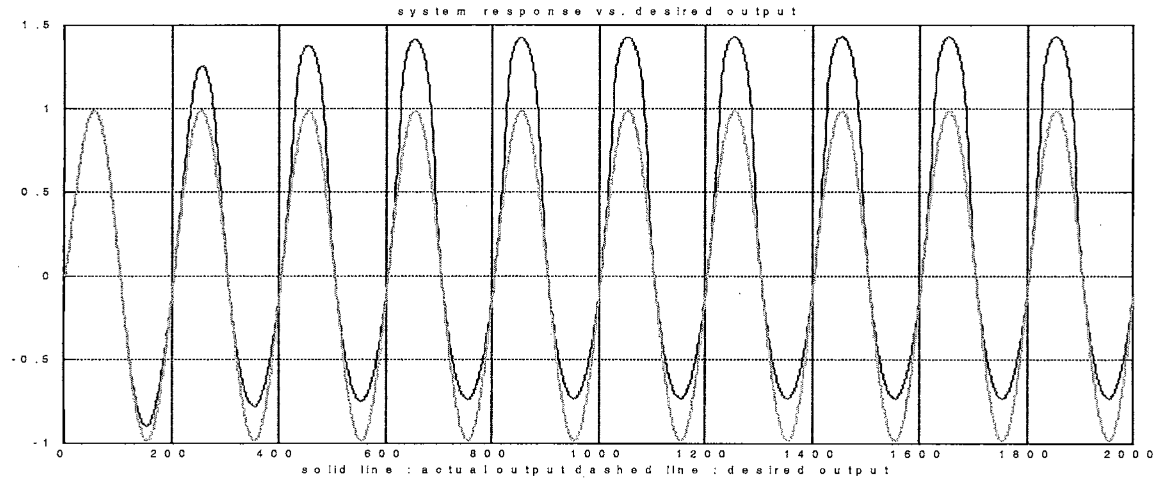


Figure 4.9 System response vs. desired output with nominal controller only (single incipient fault case; case 1)

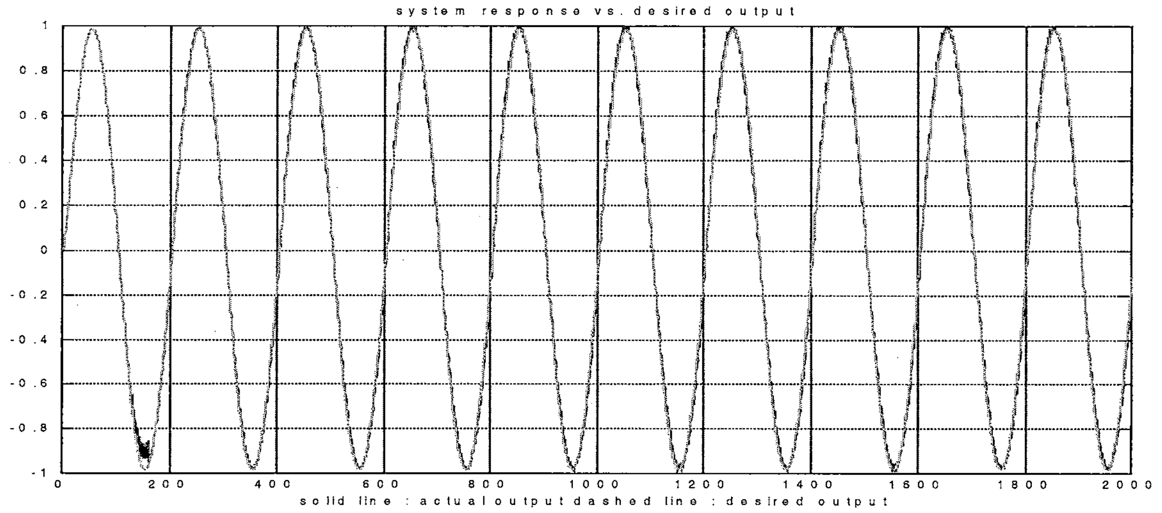


Figure 4.10 System response vs. desired output with the alternative corrective control law (single incipient fault case; case 1)

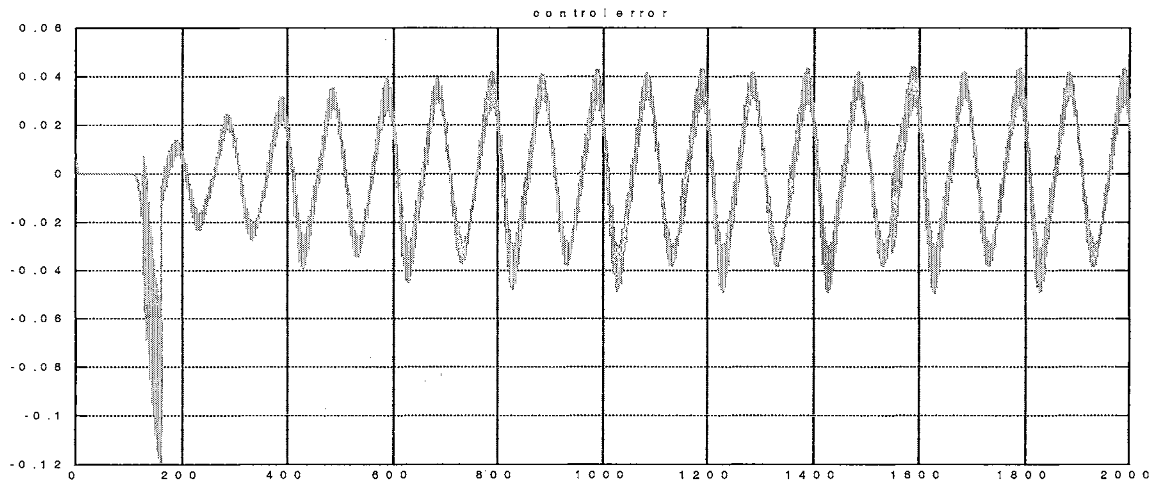


Figure 4.11 Control error with the alternative corrective control law (single incipient fault case; case 1)

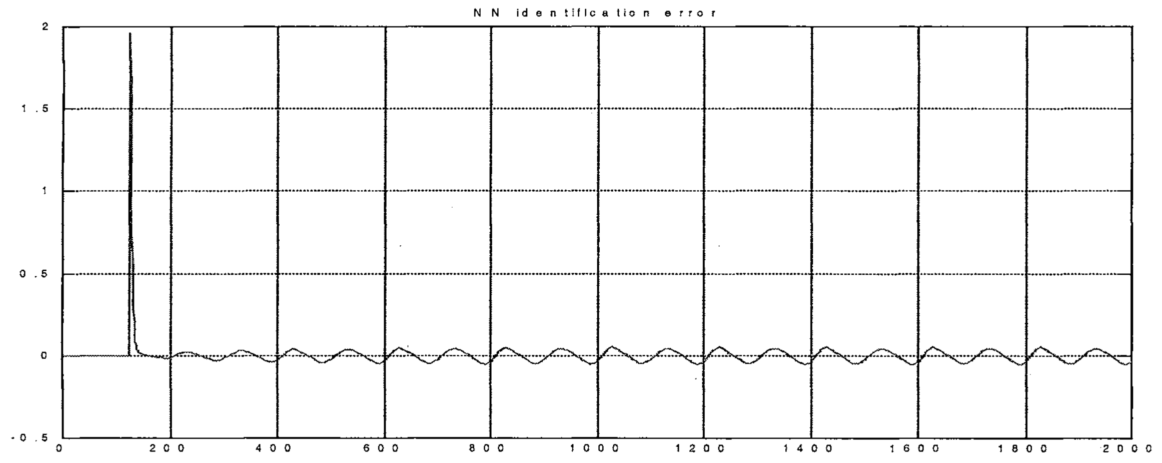


Figure 4.12 NN model identification error with the alternative corrective control law (single incipient fault case; case 1)

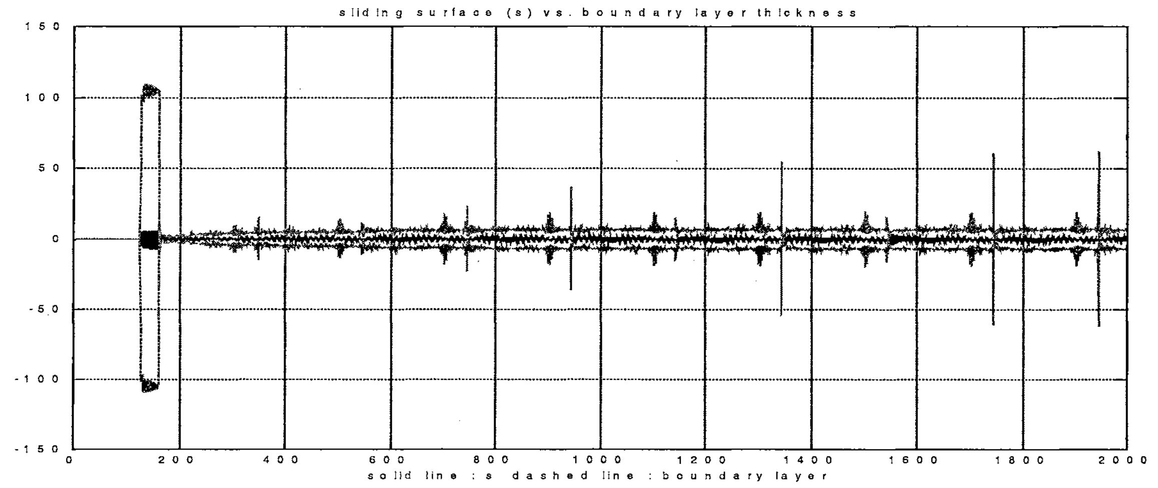


Figure 4.13 S function vs. on-line estimated boundary layer with the alternative corrective control law (single incipient fault case; case 1)

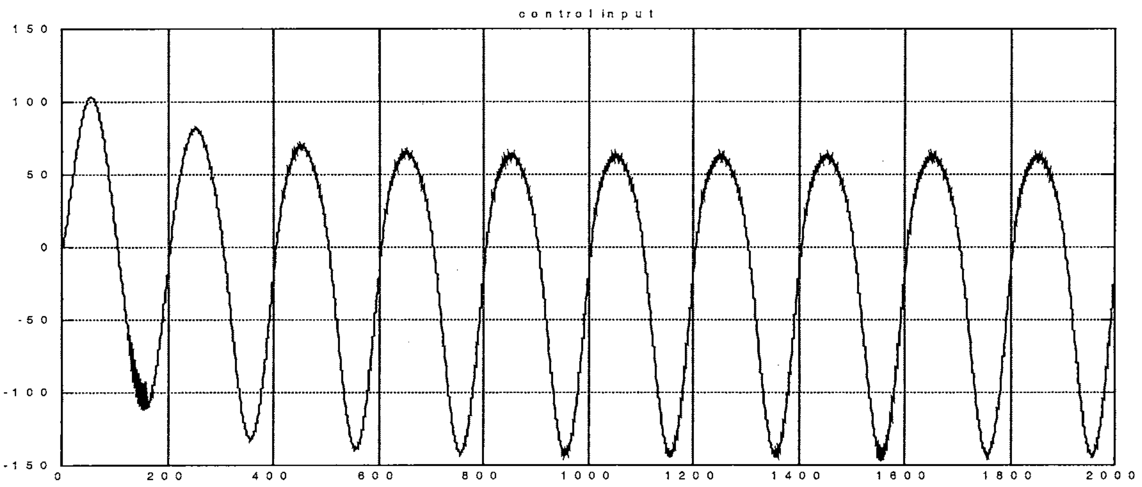


Figure 4.14 Actual control input with the alternative corrective control law (single incipient fault case; case 1)

4.1.1.2 Incipient fault case

Figure 4.9 shows the system response together with the desired output when the nominal controller is applied alone and the system suffers from an incipient fault. Again, the performance degrades slowly after the incipient fault occurs at time step 100. Figure 4.10 shows the response of the same system that suffers from the incipient fault when we use the proposed intelligent control scheme with the alternative corrective control law. Apparently, the result is greatly improved. Figure 4.11 is the actual control error plot. The system dynamics keep changing with time until the time step reaches 900. After that, the incipient fault dynamics converges. However, the intelligent controller still tries to on-line estimate the bound for the remaining uncertainty in order to reduce the performance error after it detects the fault as seen from Figures 4.12 and 4.13. The actual fault is detected at time step 125 in this case. Figure 4.14 shows the actual control input at each time step in this incipient fault case.

4.1.2 Multiple failures case

In this Subsection, multiple system component failures are generated to validate the proposed intelligent control technique. The same nominal model is used again in this Subsection. However, the system model under multiple-failure modes is represented by the following equation

$$y(k+1) = y(k) - \frac{\Delta t}{m}(cy(k)^3 + k_1 y(k-1)) + \frac{\Delta t}{m}u(k) + \sum_{i=1}^{n} \beta_i(k-T_i) f_i(y(k), y(k-1)), \tag{4.5}$$

where $\beta_i(k-T_i)$ and $f_i(\cdot)$ are the time profile and the failure dynamics under failure mode i, respectively. Without loss of generality, consider $n=2$ where $f_1(\cdot)$ and $f_2(\cdot)$ represent unknown failure dynamics and $\beta_1(\cdot)$ and $\beta_2(\cdot)$ are their corresponding time profiles, respectively.

4.1.2.1 Consecutive abrupt failures case

Two abrupt failures occur at time step 20 and 100, respectively, as shown below:

$f_1(y(k), y(k-1)) = 0.5 \times \sin(y(k) \times y(k-1))$, $\beta_1(k-T_1) = U(k-T_1)$, where $T_1 = 20$, and

$f_2(y(k), y(k-1)) = 0.5 \times y(k) \times y(k-1)$, $\beta_2(k-T_2) = U(k-T_2)$, where $T_2 = 100$.

Figure 4.15 shows the system behavior driven by the nominal controller only. As shown, without properly accommodating the failures, the system finally becomes unstable and the output goes unbounded after time step 230. On the other hand, Figure 4.16 shows the system output together with the desired output when the proposed intelligent alternative corrective control law is applied to the same system. Apparently, the failure dynamics have been properly accommodated and the control signal has been corrected by the alternative corrective control law. Figure 4.17 shows the system response under the same failure situations with the first control law.

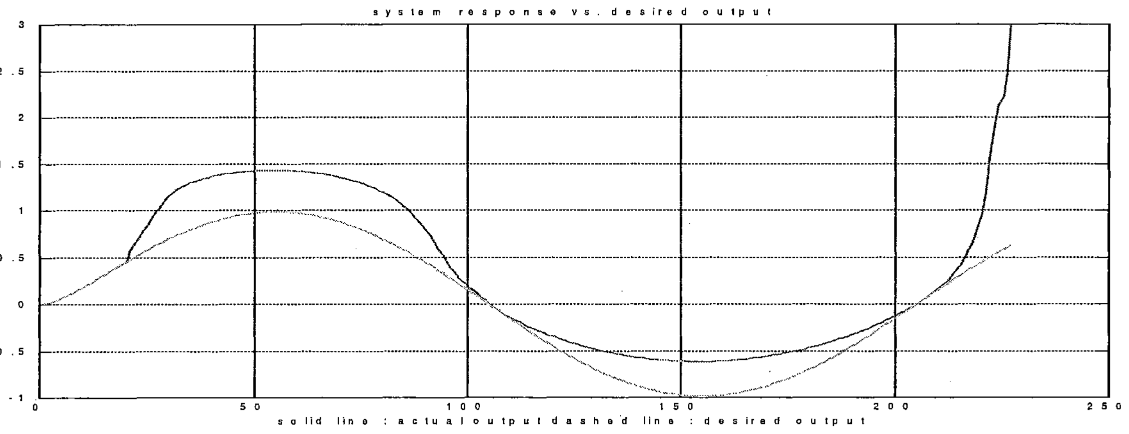


Figure 4.15 System response vs. desired output with nominal controller only (consecutive abrupt faults case; case 1)

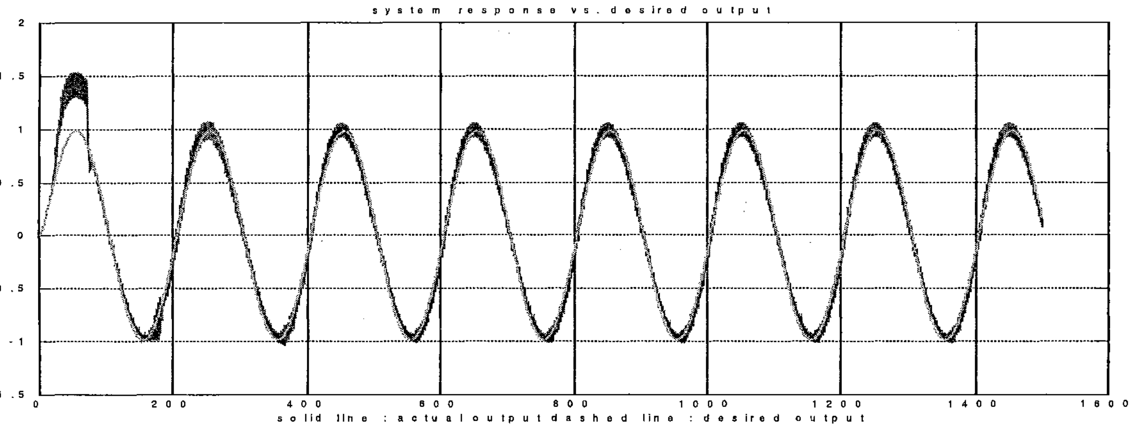


Figure 4.16 System response vs. desired output with the alternative corrective control law (consecutive abrupt faults case; case 1)

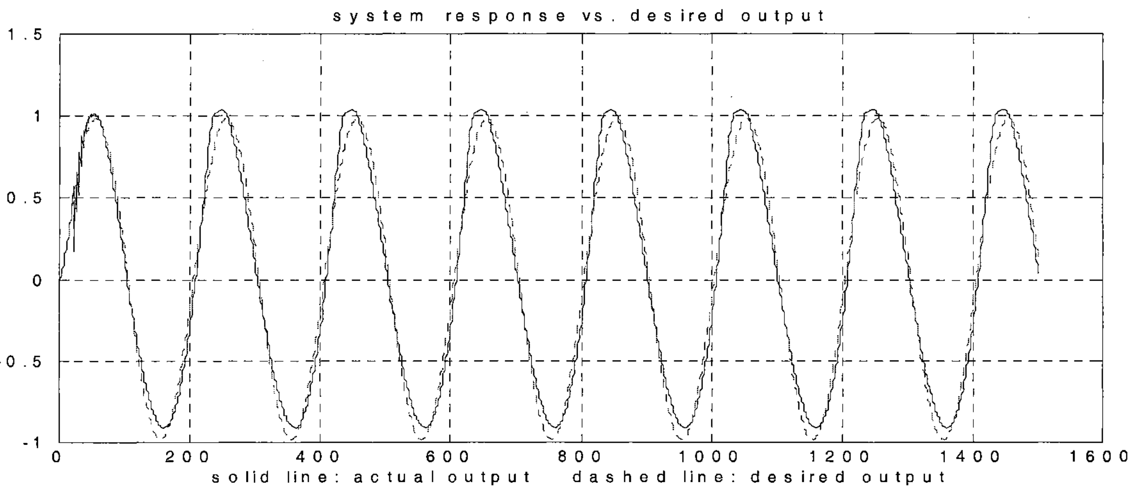


Figure 4.17 System response vs. desired output with the first control law (consecutive abrupt faults case; case 1)

4.1.2.2 Consecutive incipient failures case

Figure 4.18 shows the system response controlled by the nominal controller only when the system suffers consecutive incipient faults. The first incipient failure starts at time step 20, and, before the time profile of this fault converges, another incipient failure initiates at time step 60. The actual time profiles of these two incipient failures are given as follows:

$$\beta_1(k-T_1)=(1-e^{-\alpha_1(k-T_1)})U(k-T_1)\text{, where } \alpha_1=0.01 \text{ and } T_1=20,$$

$$\text{and } \beta_2(k-T_2)=(1-e^{-\alpha_2(k-T_2)})U(k-T_2)\text{, where } \alpha_2=0.001 \text{ and } T_2=60.$$

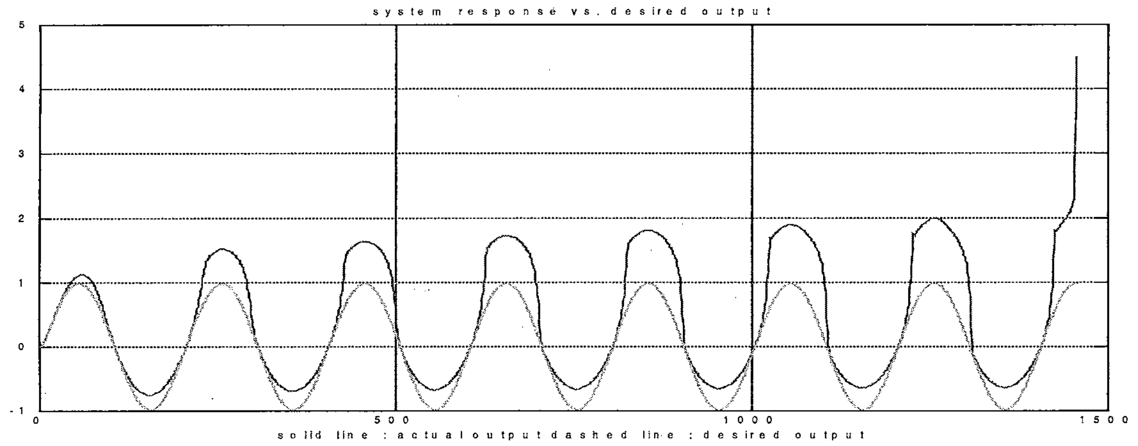


Figure 4.18 System response vs. desired output with nominal controller only (consecutive incipient faults case; case 1)

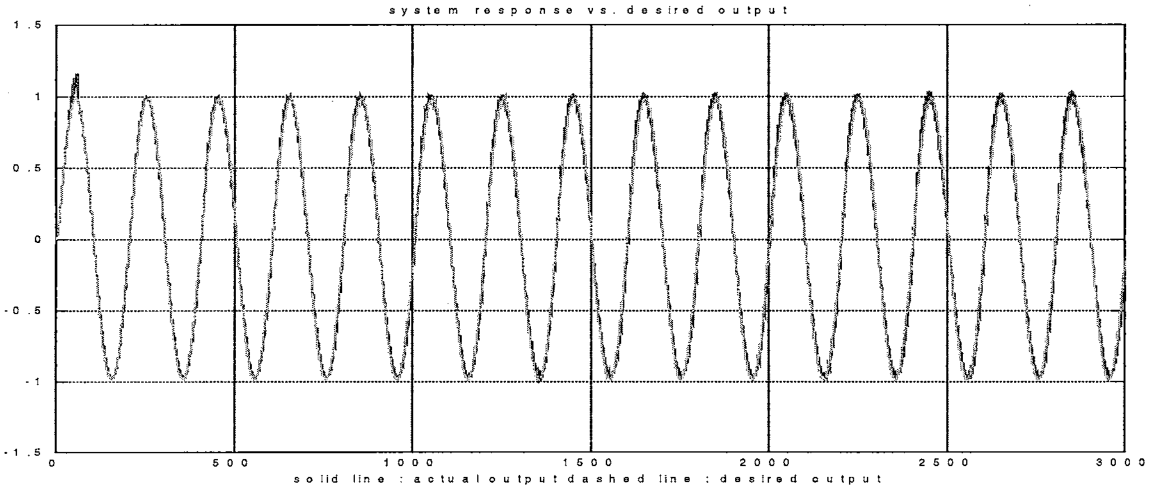


Figure 4.19 System response vs. desired output with the alternative corrective control law (consecutive incipient faults case; case 1)

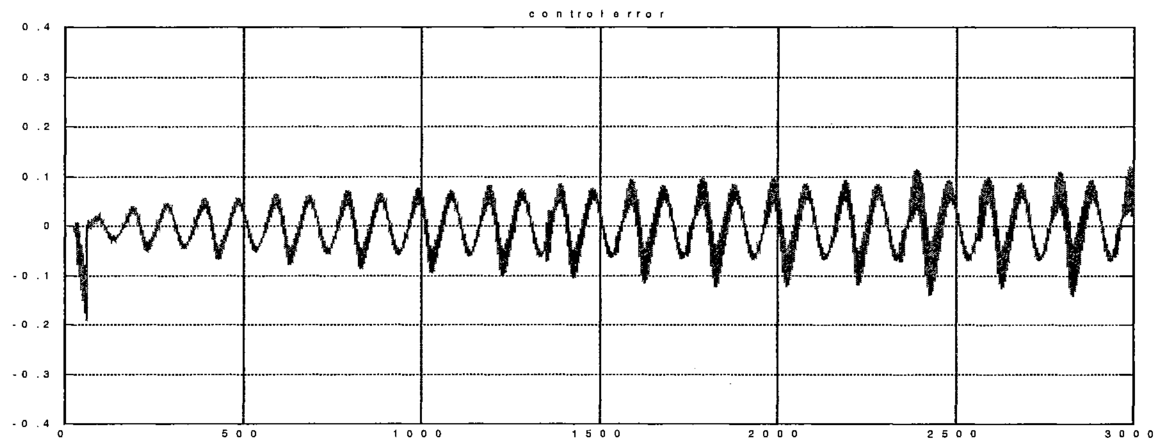


Figure 4.20 Control error with the alternative corrective control law (consecutive incipient faults case; case 1)

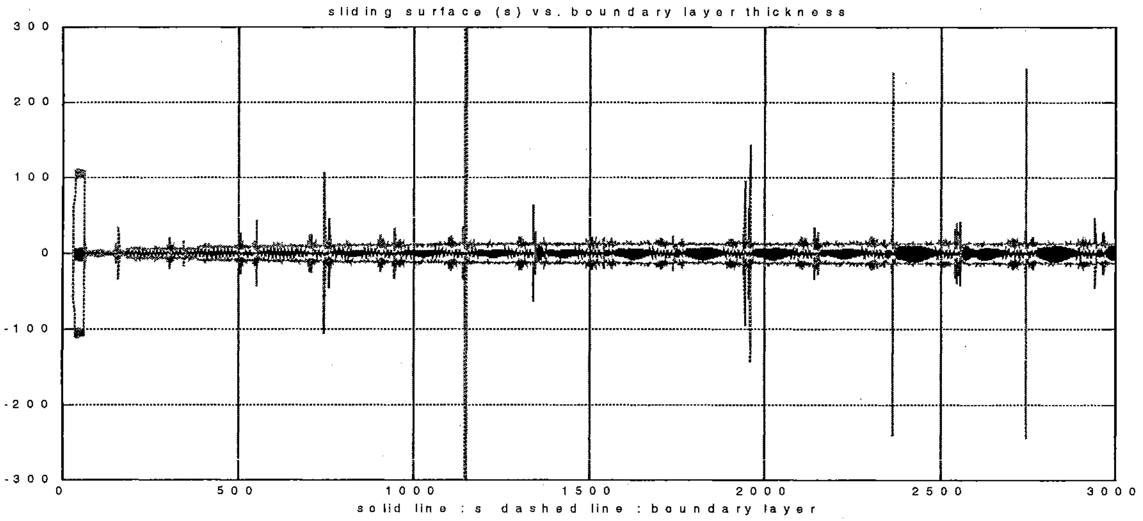


Figure 4.21 S function vs. on-line estimated boundary layer with the alternative corrective control law (consecutive incipient faults case; case 1)

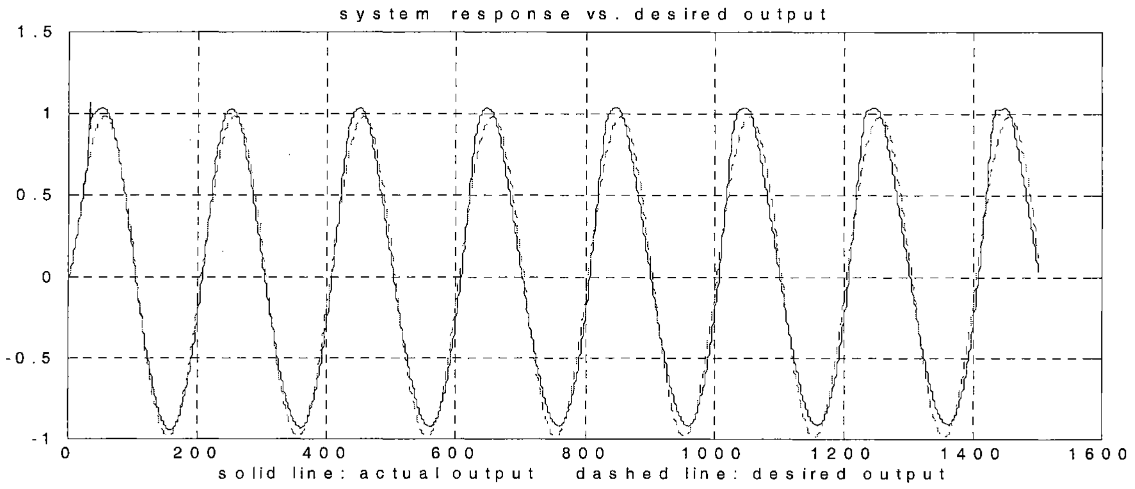


Figure 4.22 System response vs. desired output with the first control law (consecutive incipient faults case; case 1)

The failure dynamics are defined to be the same as those in *4.1.2.1.* As shown in Figure 4.18, the system output diverges to infinity because the nominal controller cannot properly accommodate the first failure, and, after the second failure happens, the system behavior is eventually out of control. Figure 4.19 indicates how the proposed alternative corrective control law handles the consecutive incipient system component failures. Observing Figures 4.20 and 4.21, we can easily conclude that the alternative corrective control law is adjusting itself on-line to accommodate the system failures and confines the performance error within a bounded range. Figure 4.22 shows the response under the first control law.

4.1.2.3 Mixed incipient and abrupt failures case

In this Subsection, the situations where the system suffers both an incipient fault and an abrupt fault are simulated. Figure 4.23 shows the system behavior when an incipient fault starts first at time step 50 and then an abrupt fault occurs at time step 100. The failure mode dynamics are given as follows:

incipient failure dynamics : $f_1(\cdot) = 1 - e^{-0.7|y(k)-y(k-1)|}$, and

abrupt failure dynamics : $f_2(\cdot) = 0.46 \times y(k) \times y(k-1)$,

where time profiles $\beta_1(k-T_1) = (1-e^{-\alpha_1(k-T_1)})U(k-T_1)$, $\alpha_1 = 0.02$, $T_1 = 50$, and

$\beta_2(k-T_2) = U(k-T_2)$, $T_2 = 100$.

Again, the system becomes unstable at time step 240 when it is controlled only by the nominal controller. Figures 4.24 and 4.25 show the proposed alternative corrective control law and the first control law successfully accommodate both system failures, respectively. Figure 4.26 shows the system response plot under the nominal controller when it first suffers an abrupt fault at time step 100 and then an incipient fault starts at time step 140. The failure mode dynamics are defined as follows:

abrupt failure dynamics : $f_1(\cdot) = 1 - e^{-0.43|y(k)-y(k-1)|}$, and

incipient failure dynamics : $f_2(\cdot) = 0.6 \times \cos(y(k) \times y(k-1))$,

where time profile $\beta_1(k-T_1) = U(k-T_1)$; $T_1 = 100$, and

$\beta_2(k-T_2) = (1-e^{-\alpha_2(k-T_2)})U(k-T_2)$, $\alpha_2 = 0.08$, $T_2 = 140$.

Figure 4.27 is the same system response plot when the alternative corrective control law is applied. As shown, the significant performance improvement has been achieved by using the proposed intelligent controller.

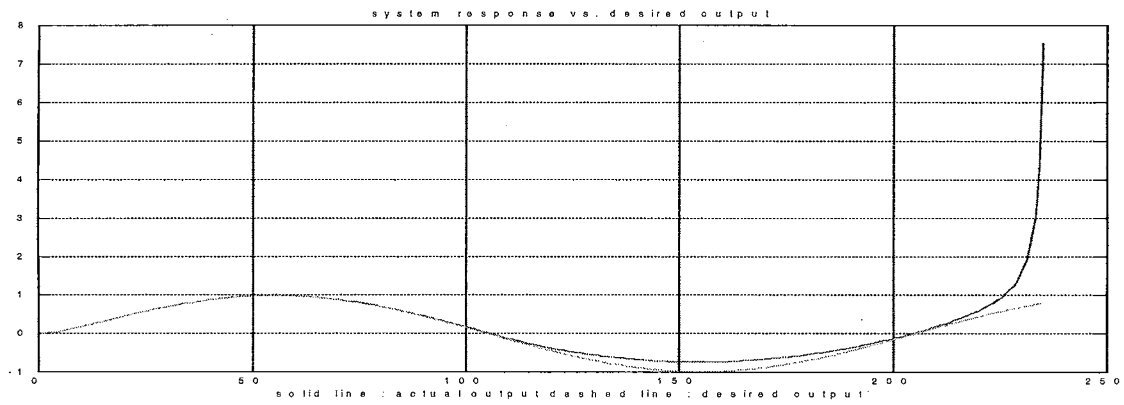


Figure 4.23 System response vs. desired output with nominal controller only (incipient-abrupt fault case; case 1)

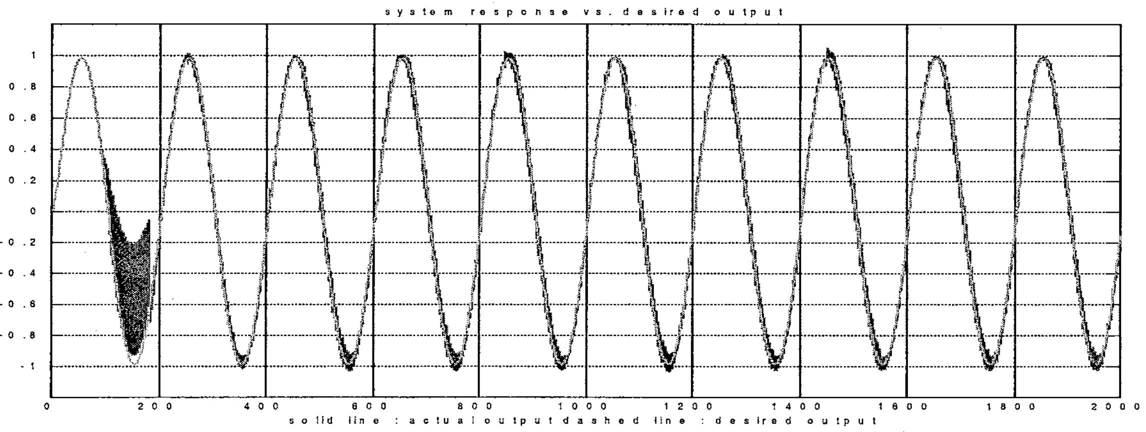


Figure 4.24 System response vs. desired output with the alternative corrective control law (incipient-abrupt fault case; case 1)

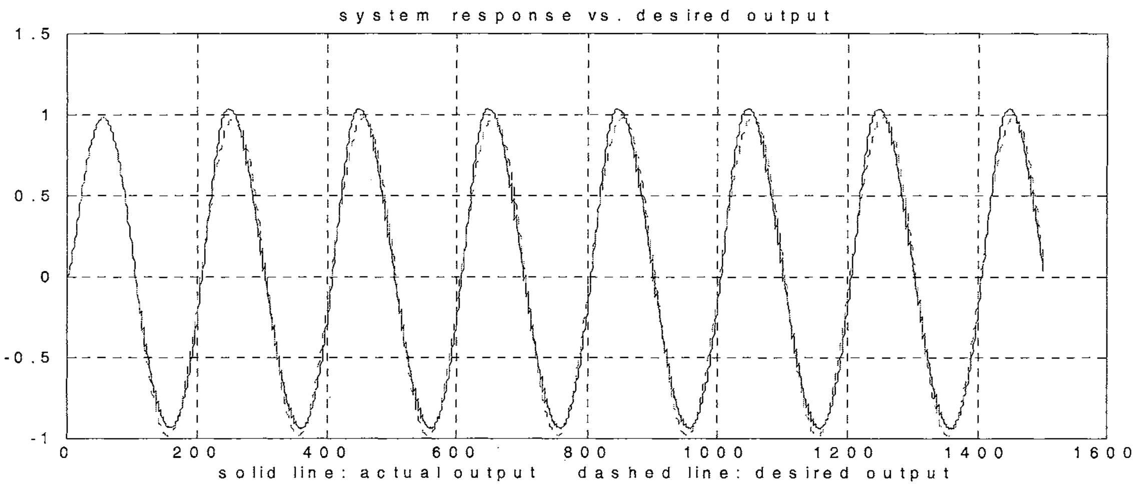


Figure 4.25 System response vs. desired output with the first control law (incipient-abrupt fault case; case 1)

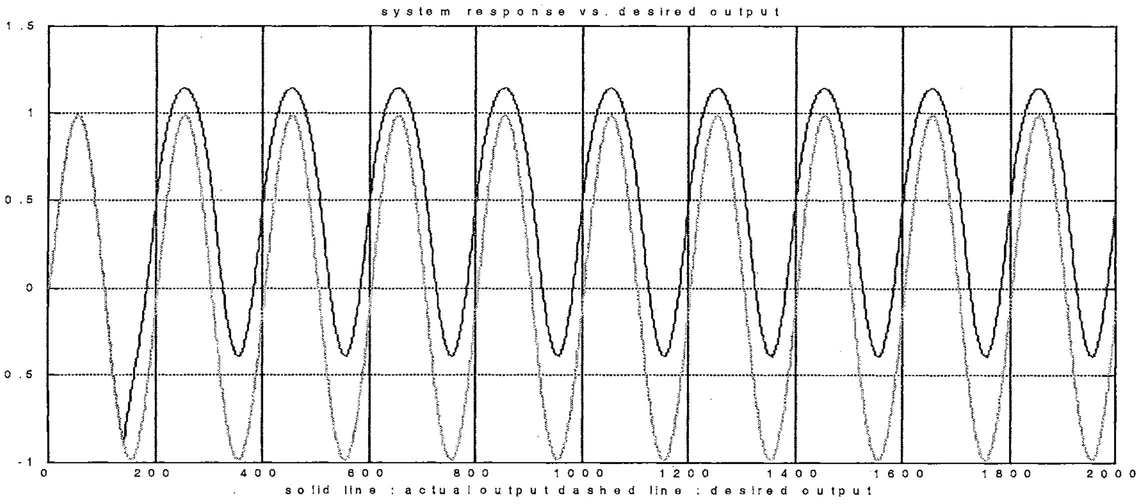


Figure 4.26 System response vs. desired output with nominal controller only (abrupt-incipient fault case; case 1)

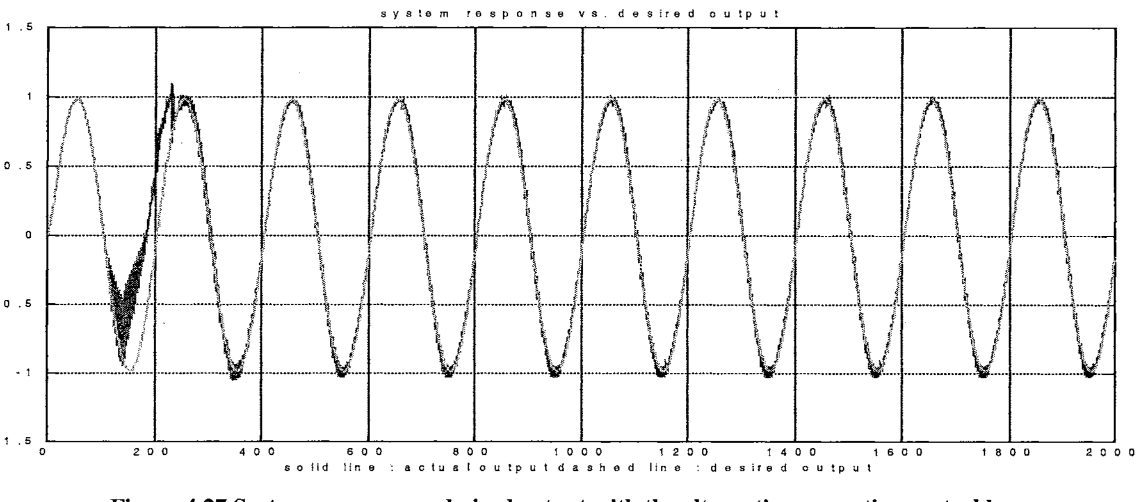


Figure 4.27 System response vs. desired output with the alternative corrective control law (abrupt-incipient fault case; case 1)

4.1.3 A benchmark problem: three-tank system

A well-regarded FDI benchmark problem shown in Figure 4.28 [85], the controlled three-tank system, is used in this Subsection to demonstrate the design of the nominal controller, the selection of the design parameters, and how to implement the proposed control technique in real application. The state equations of the system are given as

$$\begin{aligned}
\dot{x}_1 &= (-c_1 S_p sign(x_1 - x_3)\sqrt{2g|x_1 - x_3|} + u_1)/A + \eta_1(x,u) \\
\dot{x}_2 &= (-c_3 S_p sign(x_2 - x_3)\sqrt{2g|x_2 - x_3|} - c_2 S_p \sqrt{2gx_2} + u_2)/A - q_{20} + \eta_2(x,u) \\
\dot{x}_3 &= (c_1 S_p sign(x_1 - x_3)\sqrt{2g|x_1 - x_3|} - c_3 S_p sign(x_3 - x_2)\sqrt{2g|x_3 - x_2|})/A + \eta_3(x,u)
\end{aligned} \tag{4.6}$$

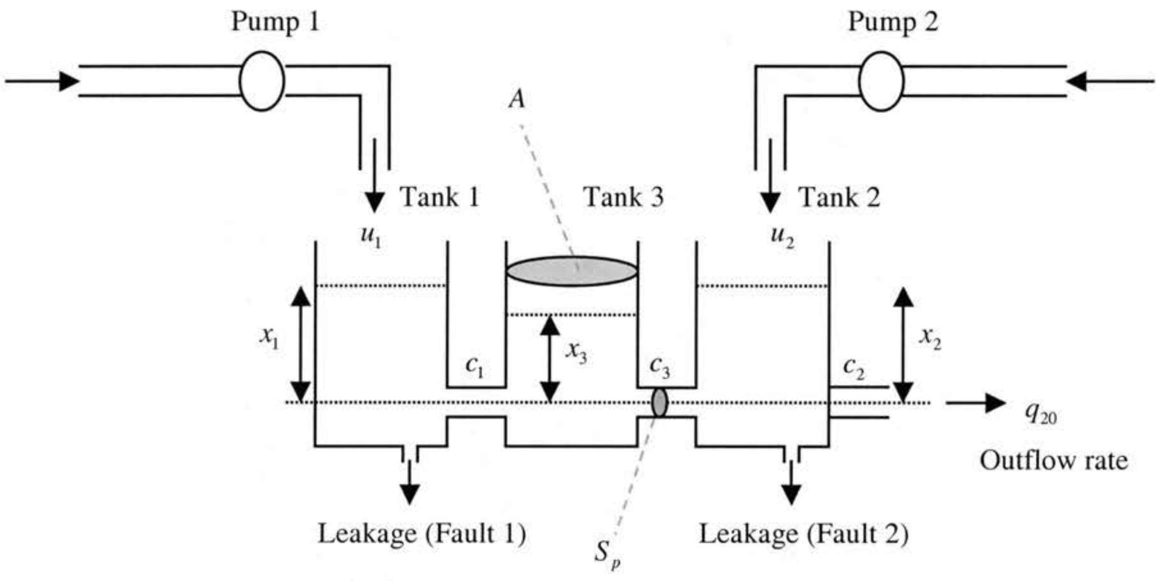


Figure 4.28 A benchmark problem (three-tank system)

Three tanks are identical and have a cylindrical shape with cross section $A = 0.0154\ m^2$. The cross section of the connection pipes is $S_p = 5 \cdot 10^{-5}\ m^2$ and the liquid levels in the three tanks are denoted by x_1, x_2, and x_3, respectively with $(0 \le x_i \le 0.69\ m, \forall i = 1,2,3)$. The control inputs, u_1 and u_2, are the flow rates coming from pumps 1 and 2 to tanks 1 and 2, respectively. $q_{20} = c_1 S_p \sqrt{2gx_2}$ is the outflow rate from tank 2. $c_1 = 1$, $c_2 = 0.8$, and $c_3 = 1$ denote the non-dimensional outflow coefficients, g is the gravity acceleration, and η_i, $\forall i = 1, 2, 3$ represent the corresponding modeling uncertainties due to the inaccuracy on the cross section of connection pipes. The discrete-time model is derived by using forward Euler approximation,

$$\dot{x}_i \approx \frac{x_i(k+1) - x_i(k)}{\Delta t}, i = 1,2,3, \tag{4.7}$$

where $\Delta t = 0.1$ second represents the sampling period. Plugging in Equation (4.7) and re-arranging the state equations, we have the nominal model in the form of Equation (3.26) such that the corresponding $f(\cdot)$ and $g(\cdot)$ are found for each state equation. Initial condition is set to be the liquid levels $x_1(0) = x_2(0) = x_3(0) = 0.15\, m$ and the control objective is to keep the liquid levels at $0.2\, m$ (i.e., $x_{1d}(k) = x_{2d}(k) = x_{3d}(k) = 0.2, \forall k > 0$). The modeling uncertainty is assumed to satisfy

$$|\eta_i(x,u)| \le \bar{\eta}_i, \forall (x,u) \in \mathrm{X}, \; i = 1, 2, 3, \tag{4.8}$$

where X represents the region of interest. In order to simulate the effects of modeling uncertainty and possible noises in the measurements, uniformly distributed random values satisfying Equation (4.8) with $\bar{\eta}_1 = 3.5\times10^{-4}$, $\bar{\eta}_2 = 2.05\times10^{-4}$, and $\bar{\eta}_3 = 6.5\times10^{-5}$ are added to the corresponding nominal state equations.

4.1.3.1 Design of nominal controllers

The design of the nominal controllers is based upon Equation (3.30) without the on-line estimator. Only the liquid levels $x_1(k)$ and $x_2(k)$ need to be considered in the controller design process since $x_3(k)$ will eventually reach the same level as long as we can keep $x_1(k) = x_2(k) = 0.2\,m$ based upon the U tube principle. Thus, the nominal controllers are

$$u_1(k) = [\bar{Y}1(k)(a + \frac{1}{\Delta t})^{-1} - f1(\cdot)] / g1(\cdot), \tag{4.9a}$$

$$u_2(k) = [\bar{Y}2(k)(a + \frac{1}{\Delta t})^{-1} - f2(\cdot)] / g2(\cdot), \tag{4.9b}$$

where
$$\bar{Y}1(k) = \frac{x_{1d}(k+1) - x_{1d}(k) + x_1(k)}{\Delta t} + a x_{1d}(k+1),$$

$$\bar{Y}2(k) = \frac{x_{2d}(k+1) - x_{2d}(k) + x_2(k)}{\Delta t} + a x_{2d}(k+1).$$

$f1(\cdot)$, $f2(\cdot)$, $g1(\cdot)$, and $g2(\cdot)$ are the corresponding terms obtained when we re-organized the nominal model into the form of Equation (3.26). Two sliding surface functions S_1 and S_2 are defined for x_1 and x_2, respectively, with the same form as Equation (3.1) and $a = 10$. The sum of the mean square control errors (i.e., $\tilde{x}_1^{\,2}(k) = [x_{1d}(k) - x_1(k)]^2$ and $\tilde{x}_2^{\,2}(k) = [x_{2d}(k) - x_2(k)]^2$) within a fixed length (i.e., 5 time steps) of time-shifting window is selected as a criterion to test performance of the nominal controllers with the presence of modeling uncertainty and possible noises. Based upon the testing results, the fault detection threshold value is then selected as 2.0×10^{-3} in steady state condition.

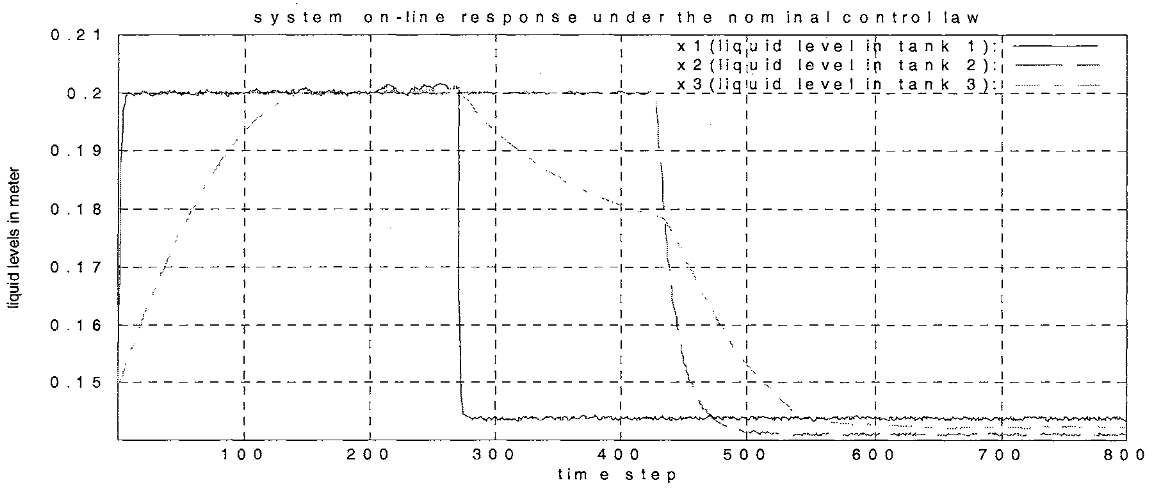


Figure 4.29 System on-line response with the nominal control law (the three-tank system)

4.1.3.2 Multiple failures: leakages in the tanks

Consider an abrupt leakage in tank 1 and an incipient leakage in tank 2 whose failure dynamics are

$$\begin{aligned} &f_1(k) = -c_1\pi r_1^2\sqrt{2gx_1(k)},\ \beta_1(k-T_1) = U(k-T_1), T_1 = 270, \\ &f_2(k) = -c_2\pi r_2^2\sqrt{2gx_2(k)}, \beta_2(k-T_2) = (1-e^{-\alpha_2(k-T_2)})U(k-T_2),\ \alpha_2 = 0.063, T_2 = 426, \end{aligned} \tag{4.10}$$

where $r_1 = 7.3\times10^{-2}$ and $r_2 = 8.4\times10^{-2}$. No information in Equation (4.10) is assumed to be known except for the state variables $x_1(k)$ and $x_2(k)$. The physical knowledge of the system provides us useful information to determine the initial upper bound of the failure dynamics. Since the failures are possible leakage problems in the tanks (i.e., the failures of system components), the maximum effect caused by the failure is suddenly draining out the liquid in the tank, which corresponds to the worst failure condition where the tank is completely broken. Thus, the initial upper bounds for failures can be chosen as the liquid levels in the tanks at the corresponding time step. Two separate 1-5-5-1 MLP neural networks are used to serve as the on-line failure estimators for f_1 and f_2, respectively, with the same static backpropagation method as the training algorithm. The selection of the MLP network structure is a design parameter and may not be optimal in this case. Generally speaking, a more complicated structure may be required for a more complex function with better performance and the computational cost is expected to increase with the complexity. The on-line approximation result is monitored by the criterion shown in Equation (3.56) with $l = 10$ and the least upper bound of the failure uncertainty is computed according to Equation (3.54). Figure 4.29 shows the liquid levels in the tanks under the nominal control law. As the first leakage in tank 1 occurs, the liquid level 1 drops quickly causing dropping of the liquid level in tank 3. As the second leakage problem occurs in tank 2, the liquid levels eventually drop below the initial condition. Applying the proposed control technique with the corrective control law, we observe significant performance improvement by proper reconfiguration of the control inputs which are the flow rates from pumps 1 and 2, as shown in Figure 4.30. The implementation of the first control law for this problem is straightforward and the result is shown in Figure 4.31 (i.e., with the same MLP estimators and the delay of compensation).

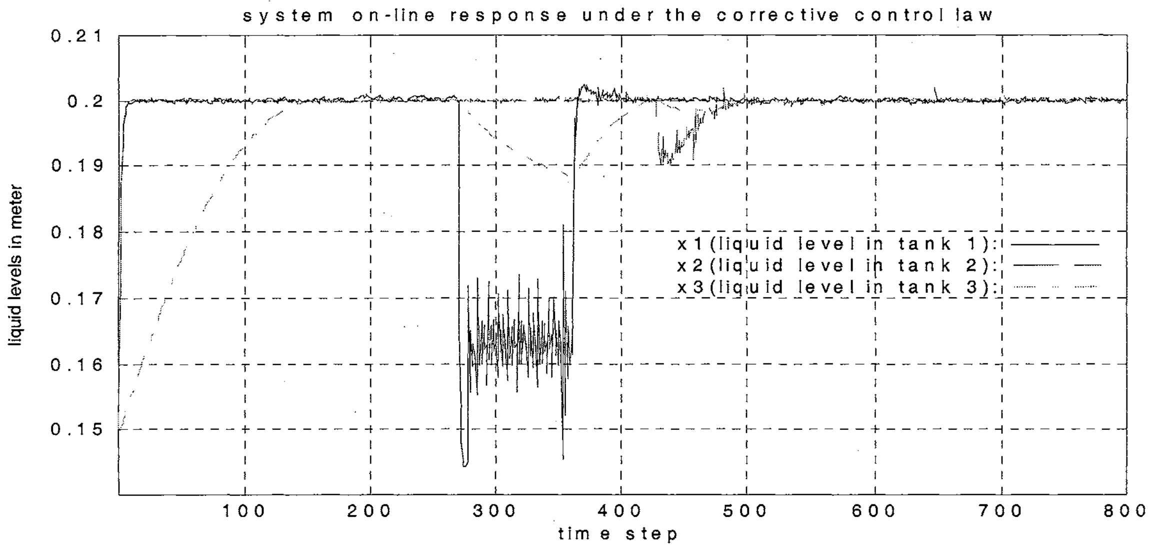


Figure 4.30 System on-line response with the alternative corrective control law (the three-tank system)

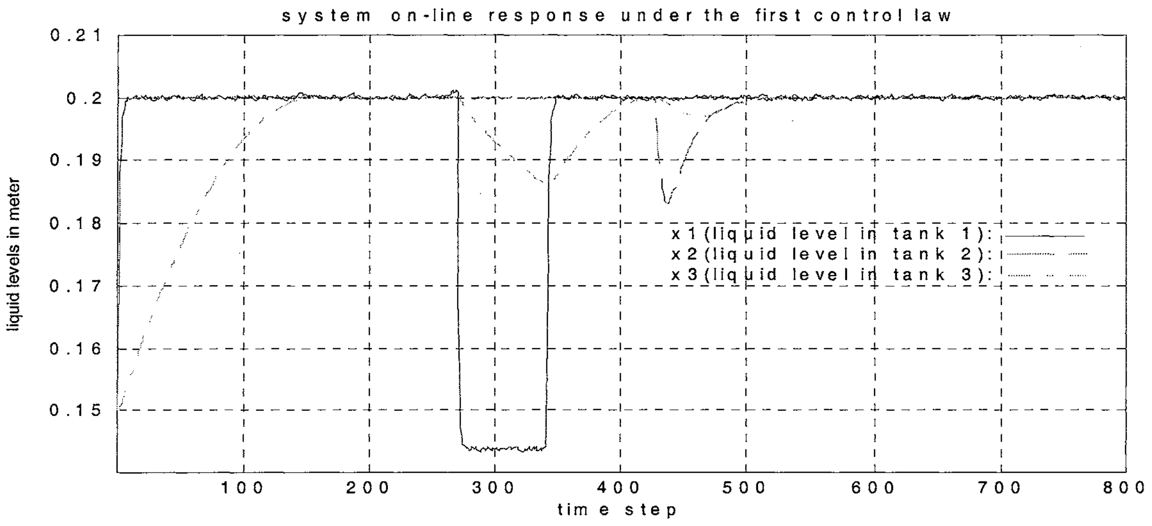


Figure 4.31 System on-line response with the first control law (the three-tank system)

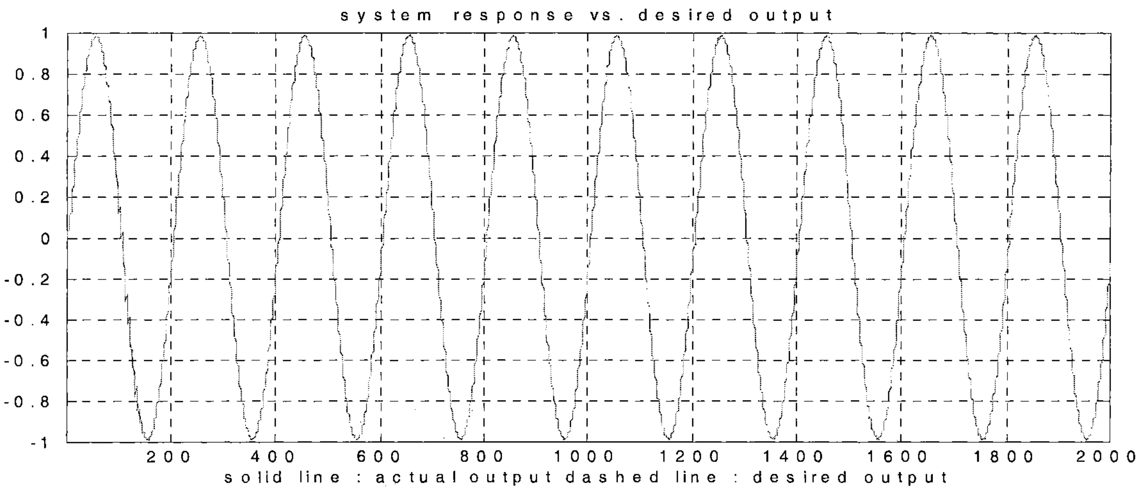


Figure 4.32 System response vs. desired output with the alternative corrective control law in false alarm situation

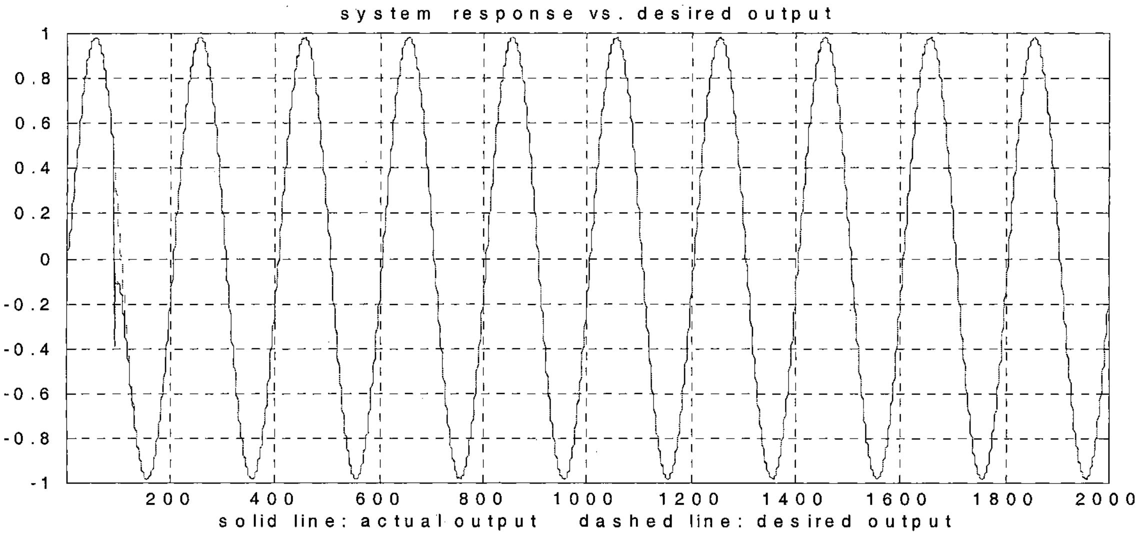


Figure 4.33 System response vs. desired output with the first control law in false alarm situation

4.1.4 Fault detection delay and false alarm

The simulation results shown in Figures 4.24 and 4.25 indicate a fault detection delay situation. One system failure occurred at time step 50 and it was not detected until the time step reached 100 at which time the intelligent control regulator initialized the on-line estimator. However, the system response is still quite satisfactory because the response affected by the failure is so small that no adjustment of the control law is required. The on-line learning and compensation is performed only after the effect caused by the failure is significant enough to degrade the performance. In order to test the proposed intelligent control framework in false alarm situation, a false failure detection signal is generated at time step 100 in the same system with the same failure situations. Right after the initialization of the on-line estimator, the failure dynamics are eliminated to test how the system responds to this situation. Figures 4.32 and 4.33 show the system output plots when the alternative corrective control law and the first control law are applied, respectively. The result shows compliance with the expectation from the analysis. The on-line estimator is triggered to learn the remaining uncertainty between the nominal model and the actual system dynamics in noise-free situations.

4.1.5 Simulation test in noisy measurement situations

This subsection is dedicated to investigate how the intelligent control scheme proposed will react in noisy environments. Without loss of generality, only the multiple-failure situations will be used to test system performance with noisy measurements. In all the simulation tests, random Gaussian white noise with zero mean and different variances will be added to the measurements. All the necessary computations including the on-line identification of the unknown failure dynamics, computation of the sliding surface function, and searching of the effective control input are based upon the noisy measurements without any noise reduction or cancellation process (i.e., assume no prior information of the noise model is available). Figure 4.34 shows the system response in a consecutive abrupt failures case (Subsection *4.1.2.1*) controlled by the intelligent control regulator using the alternative corrective control law in the noisy environment with

Gaussian white noise, variance 4.0×10^{-4} in the measurements. The noisy measurements together with the actual white noise are shown in Figure 4.35. Figures 4.36-4.37 and 4.38-4.39 are the test results for the consecutive incipient faults case (Subsection *4.1.2.2*) and the abrupt-incipient fault case (Subsection *4.1.2.3*), respectively, when the same variance Gaussian white noises are added to the measurements. As clearly seen, the system performance degrades significantly in noisy environments. It is a fully expected result since the measurements contain unpredictable noises such that the on-line estimator will have larger remaining uncertainty with regards to the failure dynamics and all the computations for the effective control inputs are based upon the contaminated measurements.

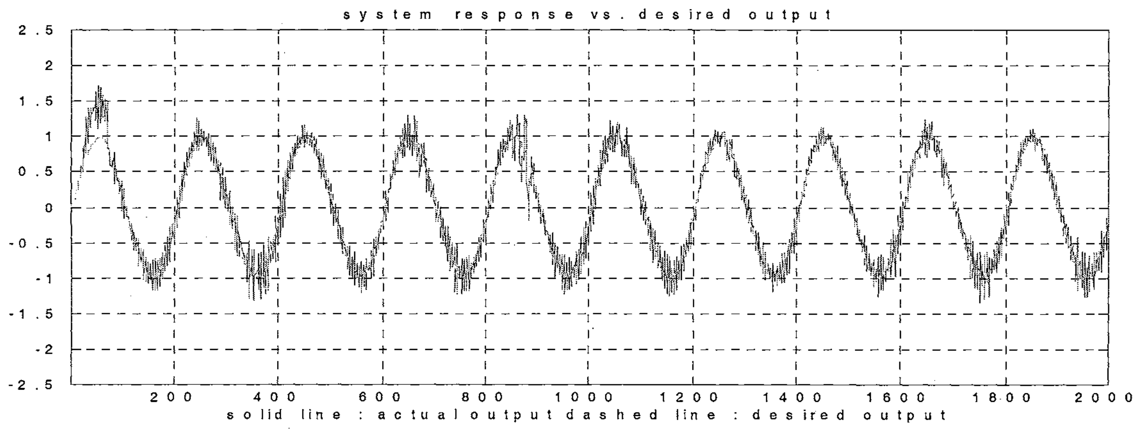


Figure 4.34 System response in 4.1.2.1 consecutive abrupt failure case (with Gaussian white noise variance 4e-4) using the alternative corrective control law

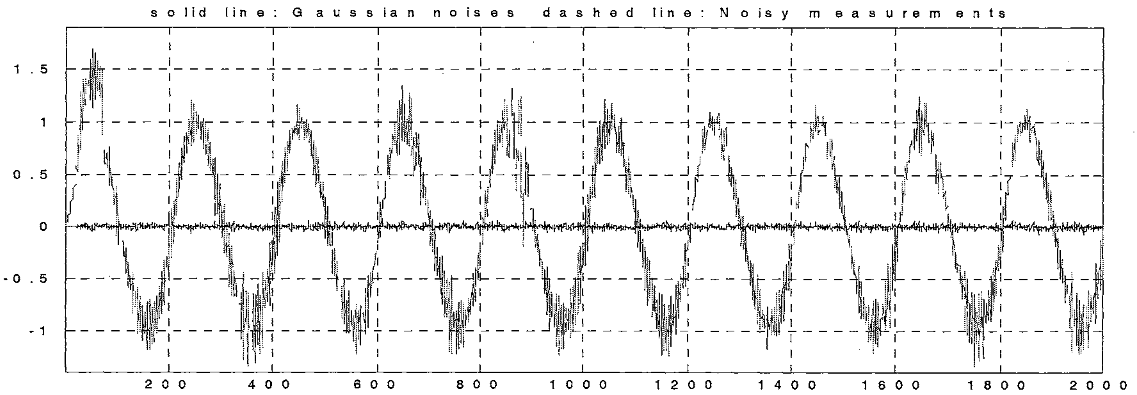


Figure 4.35 Actual noisy measurements vs. Gaussian noises in 4.1.2.1 consecutive abrupt failure case (with Gaussian white noise variance 4e-4) using the alternative corrective control law

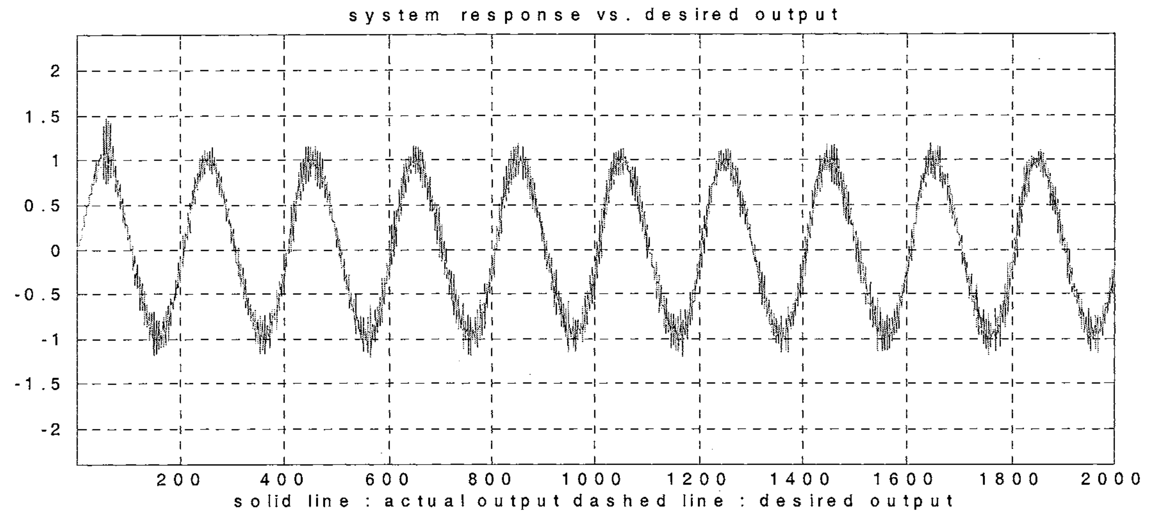


Figure 4.36 System response in 4.1.2.2 consecutive incipient failure case (with Gaussian white noise variance 4e-4) using the alternative corrective control law

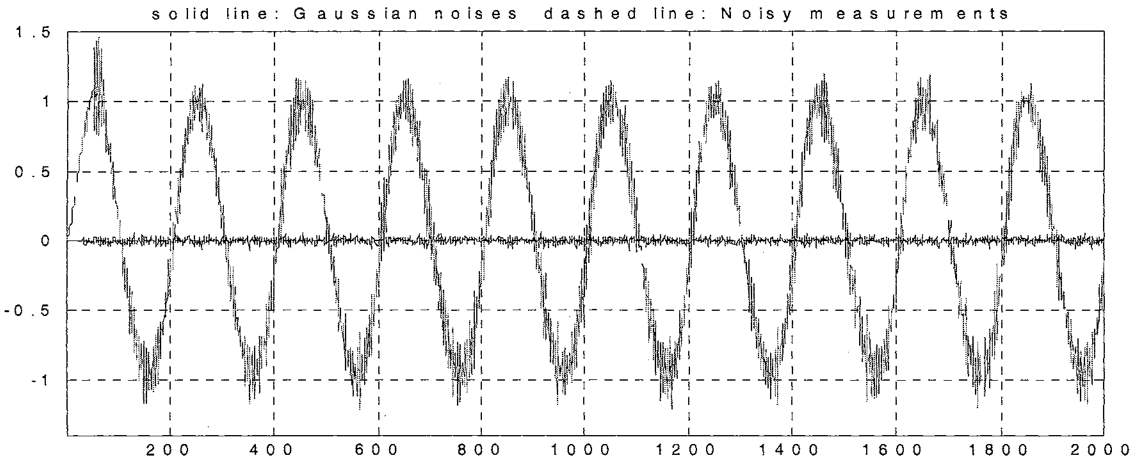


Figure 4.37 Actual noisy measurements vs. Gaussian noises in 4.1.2.2 consecutive incipient failure case (with Gaussian white noise variance 4e-4) using the alternative corrective control law

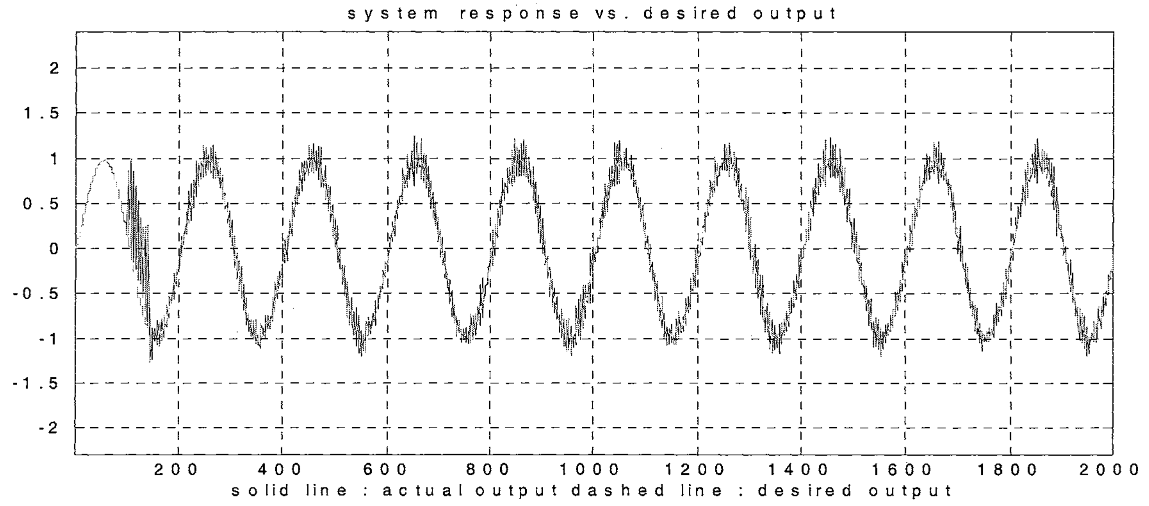


Figure 4.38 System response in 4.1.2.3 abrupt-incipient failure case (with Gaussian white noise variance 4e-4) using the alternative corrective control law

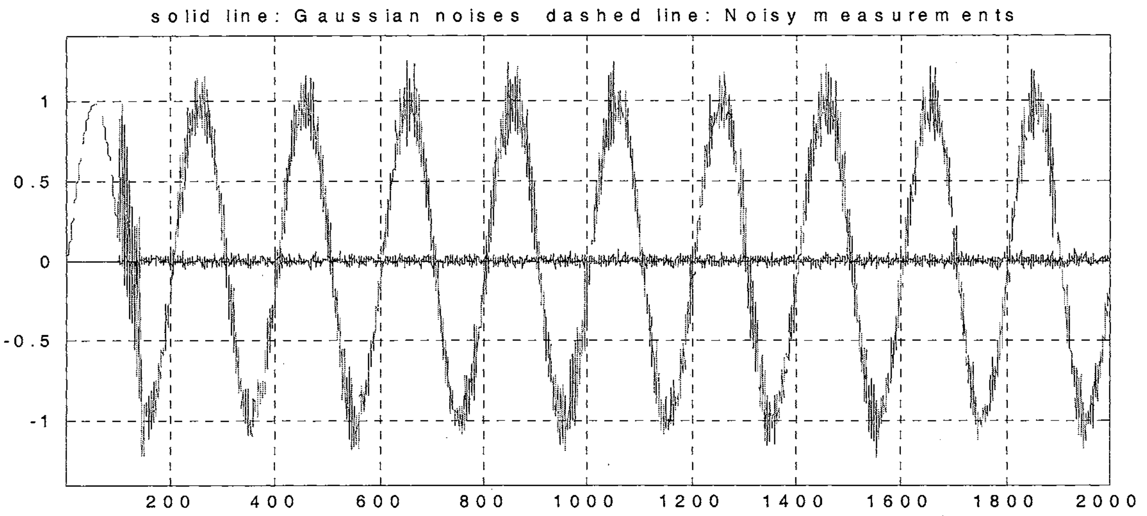


Figure 4.39 Actual noisy measurements vs. Gaussian noises in 4.1.2.3 abrupt-incipient failure case (with Gaussian white noise variance 4e-4) using the alternative corrective control law

Among numerous simulation tests in noisy environments, it is found that the system performance is sensitive to the variance of the noise and the system behavior seems to oscillate around the desired trajectory. For the noise with variance higher than 6.4×10^{-3}, not only is the system performance significantly degraded, but also the possibility of instability is increasing. The pre-processing of the noisy measurements seems to be an important and necessary step for a better system performance.

4.1.6 Comments and discussions

The first control law derived directly from the discrete-time Lyapunov stability theory is simple, straightforward, and requires less computation and fewer design parameters to implement in a real-time situation. Once the abnormal system behavior is detected, the intelligent control regulator switches the control action (nominal controller) to follow the first control law and compensates immediately for the failure dynamics based upon the realization of the neural network on-line estimator for the unanticipated failure dynamics without performing any convergence test of the on-line learning. The simulation results show:

1. Although the system response for the first control law has relatively larger performance error than the error under the alternative corrective control law, it exhibits much smoother response.
2. The system response is sensitive to the learning rate of the on-line training algorithm (i.e., static backpropagation algorithm is used here). The reason is apparent from the fact that the first control law does not perform the convergence test before the on-line learning result is used to compensate for the failure dynamics. Although it is true that the gradient descent algorithm is stable given a small enough learning rate, small learning rates will usually slow down learning and degrade the transient control performance, especially in the on-line situation where the result has a significant and immediate effect on the control error.

The alternative corrective control law based upon the discrete-time sliding mode control technique requires more design parameters and more computational cost to implement. This is a corrective control law that is used to compensate the nominal control law for the accommodation of unexpected failure dynamics. The simulation results show:

1. Generally speaking, the system performance error is smaller than that under the first control law. Although the on-line estimation of the boundary layer thickness for the remaining uncertainty of the failure dynamics based upon the on-line approximation results in much more computational burden, it is well justified in

the system performance and makes the control law almost insensitive to the learning rate of the on-line training process.

2. The delay of nominal control law compensation (i.e., with $U(k-T_c)$ in Equation (3.51)) usually results in a better transient behavior although the system stability cannot be theoretically guaranteed during the time period of delay.
3. Due to the nature of the sliding mode control technique, the system response will oscillate around the desired dynamics within the boundary layer.

4.2 CASE 2

4.2.1 Example 1

Consider the same system model we used in *4.1.2.1, Consecutive abrupt failures case*, where both the failure dynamics are explicit functions of past system outputs only and the system behavior will go unbounded under the nominal controller. It actually belongs to the first case. However, the prior information of the failure dynamics are not available or not accurate enough such that we may misclassify it. Thus, the control strategy of case 2 is used to deal with the on-line fault tolerant control problems.

Assume that the inaccurate or insufficient prior information indicates that the failure dynamics is an explicit function depended upon the past system outputs, $[y(k), y(k-1)]$, but, we are not sure whether or not the current control input is involved in the argument. So, the more general argument, (i.e., $[y(k), y(k-1), u(k)]$), is chosen as the input of the NN on-line estimator. Right after a system fault is detected, a 3-2-1 MLP neural network is initialized as the on-line estimator for the unknown failure dynamics. A time-shifting data window contains the most recent 10 sets of the system input-output measurements that are used as training patterns at each time step and the Levenberg-Marquardt algorithm with Bayesian regularization is used to train the NN estimator [66]. When the learning process converges, a simple gradient descent optimization algorithm with variable learning rate is applied for the searching process of the effective control input.

Figure 4.40 shows the on-line control result, where the solid line represents the actual system output and the dashed line is the desired output. As clearly seen from the figure, after the time step 32 (i.e., the first abrupt fault happened at time step 20, almost being detected immediately, and 10 sets of measurements were collected for the training process), the intelligent controller tries to drive the system output to follow the desired trajectory based upon only partial information of the failure dynamics available at each time step to estimate the Jacobian of the failure dynamics with respect to the current control input and search the best effective control action to satisfy inequalities (3.8) or (3.9). The system output seems to be controlled well even when there is another abrupt fault happened at time step 100. Figure 4.41 shows the output prediction from the on-line estimator for the unknown failure dynamics at each time step together with the actual output from the failure dynamics. The actual control input at each time step is shown in Figure 4.42.

A much better performance can be obtained if we increase the length of the time-shifting data window, j, from 10 to 20, using the same network structure (i.e., 3-2-1 MLP) and the same training algorithm (i.e., Levenberg-Marquardt algorithm with Bayesian regularization). However, instead of using only 10 sets of input-output measurements, the most recent 20 sets of the measurements are used to train the NN on-line estimator at each time step.

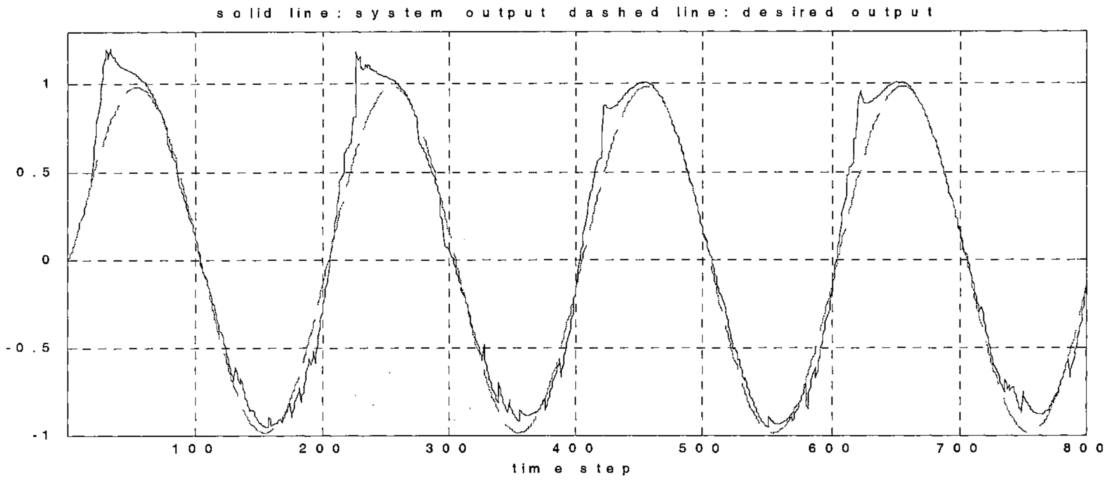


Figure 4.40 System response vs. desired output (10-data window; consecutive abrupt faults; example 1; case 2)

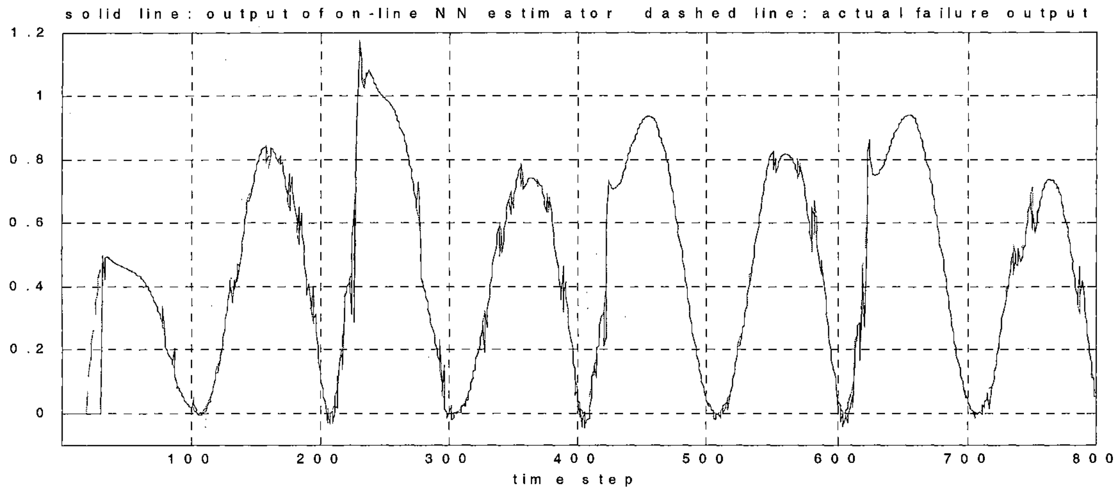


Figure 4.41 Output prediction from the on-line estimator vs. actual failure dynamics output (10-data window; consecutive abrupt faults; example 1; case 2)

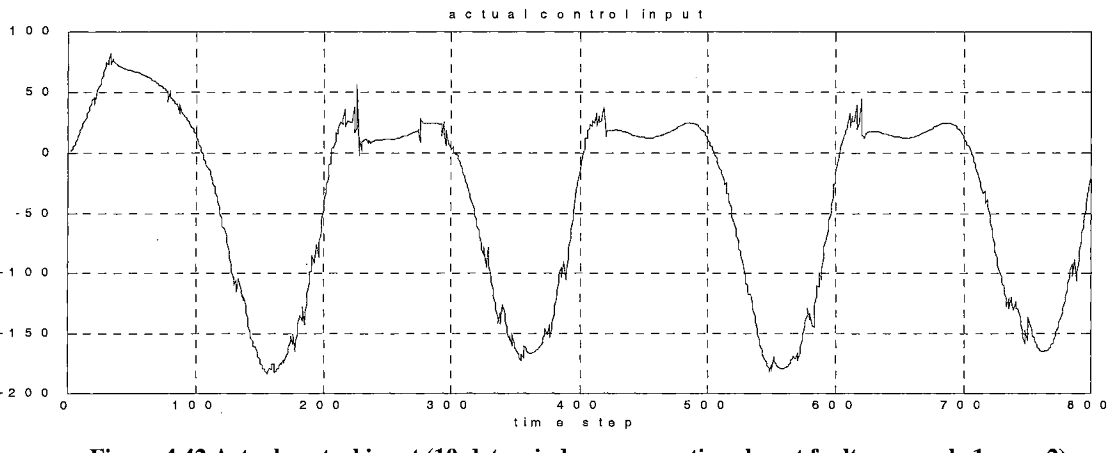


Figure 4.42 Actual control input (10-data window; consecutive abrupt faults; example 1; case 2)

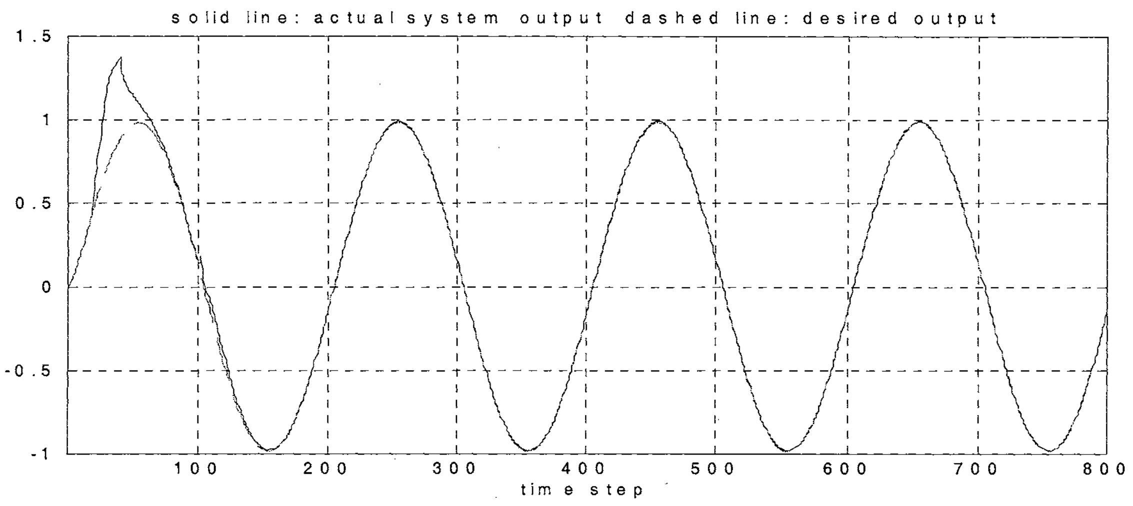


Figure 4.43 System response vs. desired output (20-data window; consecutive abrupt faults; example 1; case 2)

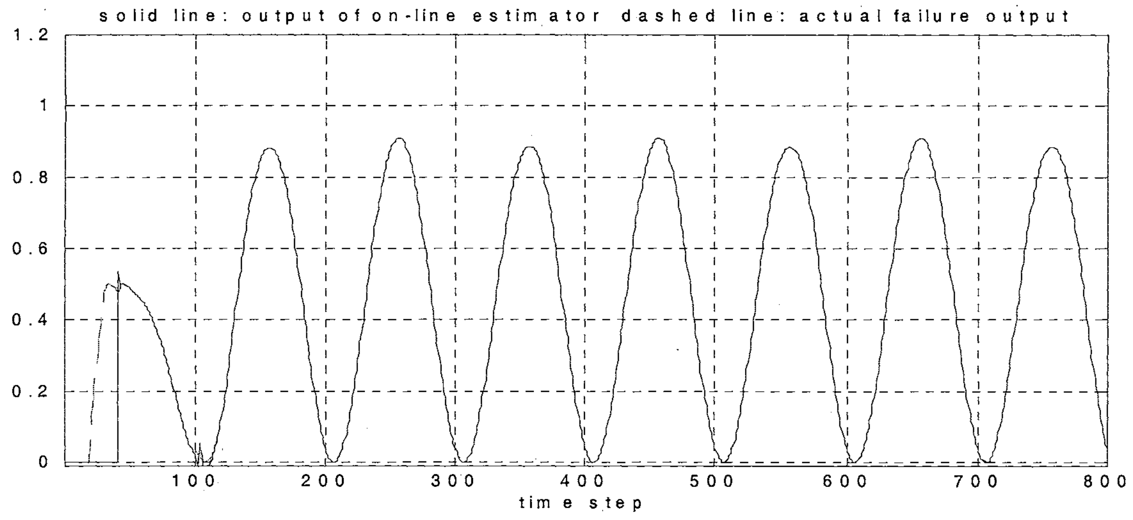


Figure 4.44 Output prediction from the on-line estimator vs. actual failure dynamics output (20-data window; consecutive abrupt faults; example 1; case 2)

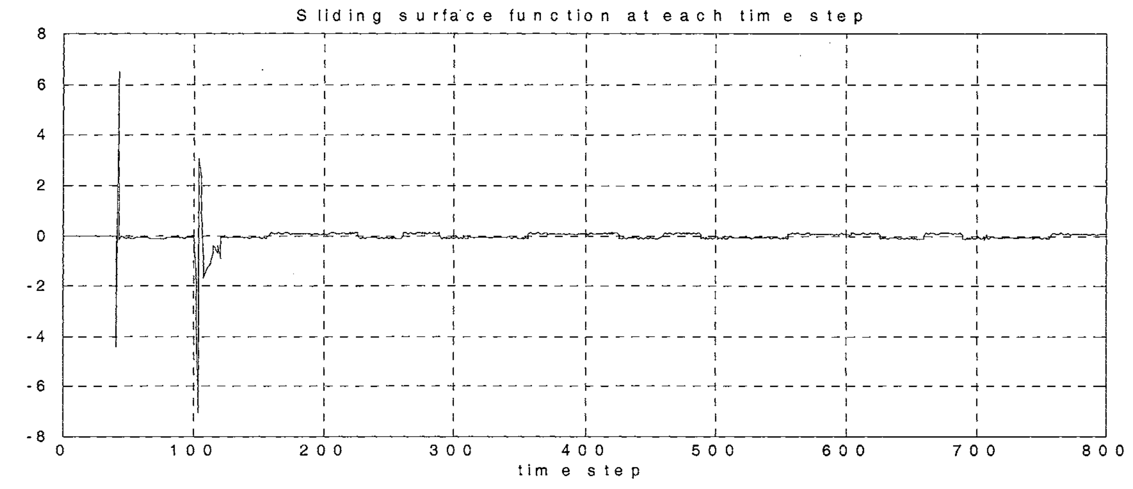


Figure 4.45 Sliding surface function at each time step (20-data window; consecutive abrupt faults; example 1; case 2)

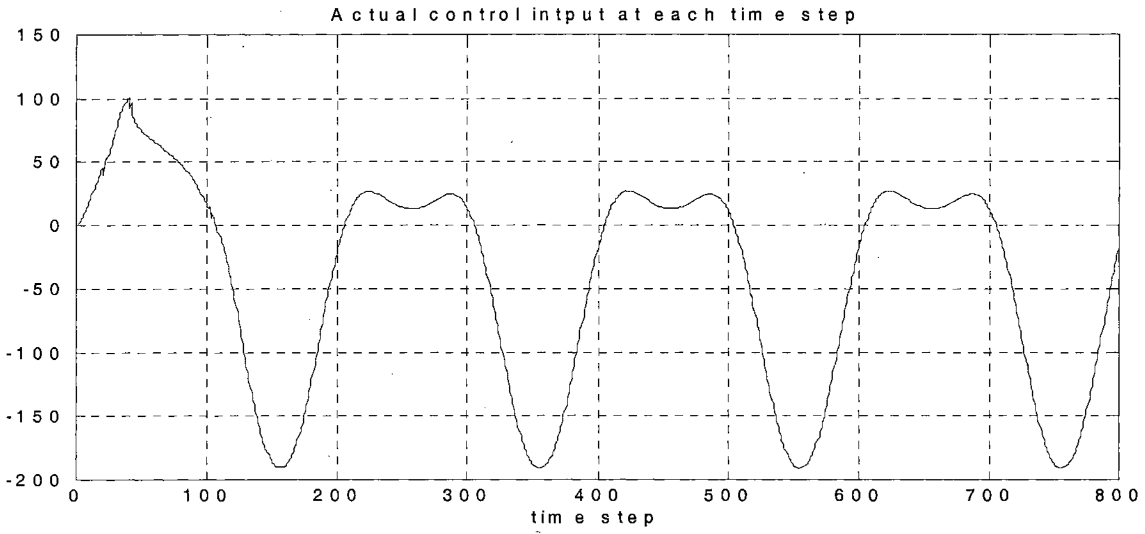


Figure 4.46 Actual control input (20-data window; consecutive abrupt faults; example 1; case 2)

The simulation result is significantly improved as shown in Figure 4.43. Figure 4.44 shows the output prediction of the failure dynamics from the NN on-line estimator together with the actual failure output. As shown, both system performance and prediction are much better and smoother than those in the last result. The second fault, happened at time step 100, caused a slight deviation of the system output from the desired output, but it was quickly controlled by our intelligent on-line control regulator. Figure 4.45 shows the sliding surface function at each time step. It has been controlled within a small range of deviation from zero (i.e., the desired system behavior). The actual control input at each time step is shown in Figure 4.46 and it is much smoother, compared with that in Figure 4.42.

Although the result is significantly improved using these design parameters for this case, we do exert much more computational cost as that in the last simulation since we double the length of the data window and trade the computational cost for the system performance. The result again shows the trade-off dilemma we have to face when we select the design parameters for the on-line fault tolerant control problems. Clearly, all these design parameters have to be reasonable for the hardware computational capability in the real applications.

The simulation result for the same failures in consecutive incipient faults using 20-data window with the following time varying profiles are shown in Figures 4.47-4.50.

$$\beta_1(k-T_1)=(1-e^{-\alpha_1(k-T_1)})U(k-T_1) \quad \text{where } \alpha_1=0.05 \text{ and } T_1=20,$$

$$\beta_2(k-T_2)=(1-e^{-\alpha_2(k-T_2)})U(k-T_2) \quad \text{where } \alpha_2=0.05 \text{ and } T_2=100.$$

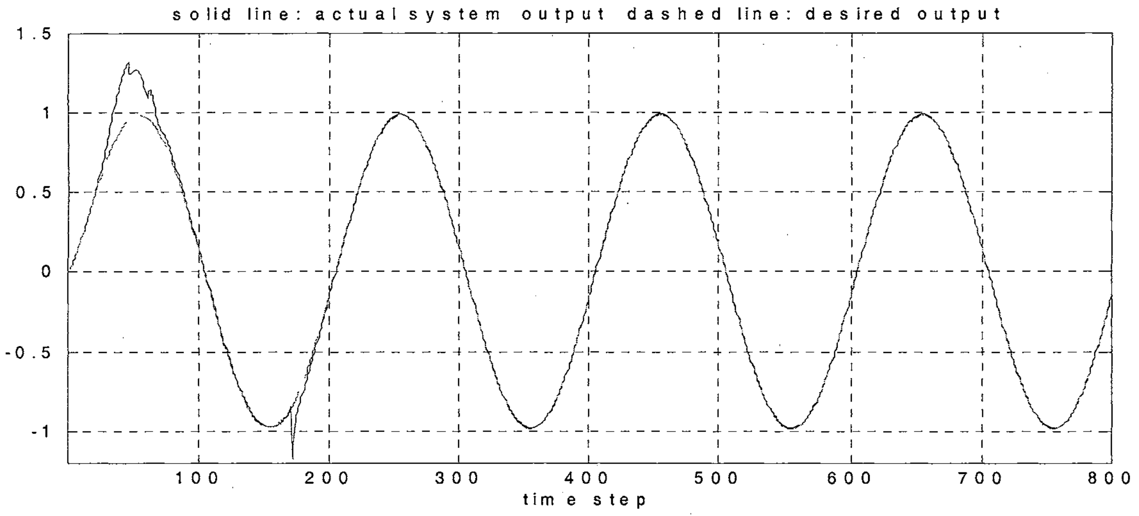


Figure 4.47 System response vs. desired output (20-data window; consecutive incipient faults; example 1; case 2)

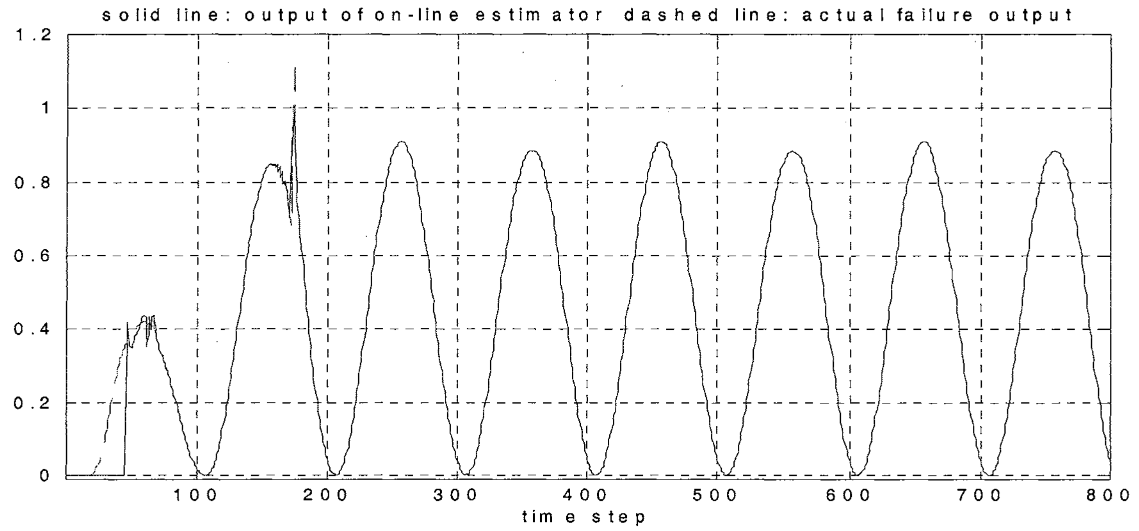


Figure 4.48 Output prediction from the on-line estimator vs. actual failure dynamics output (20-data window; consecutive incipient faults; example 1; case 2)

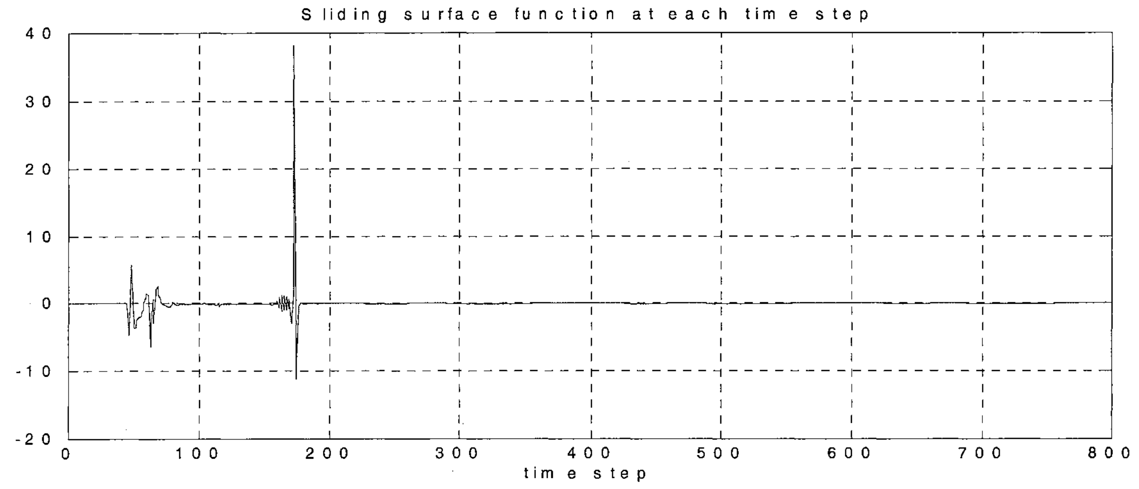


Figure 4.49 Sliding surface function at each time step (20-data window; consecutive incipient faults; example 1; case 2)

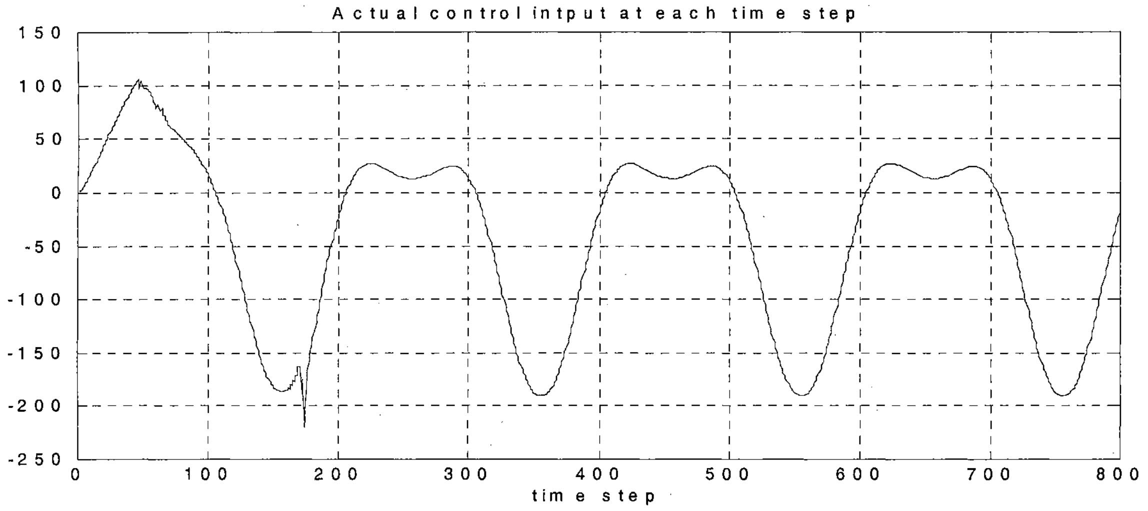


Figure 4.50 Actual control input (20-data window; consecutive incipient faults; example 1; case 2)

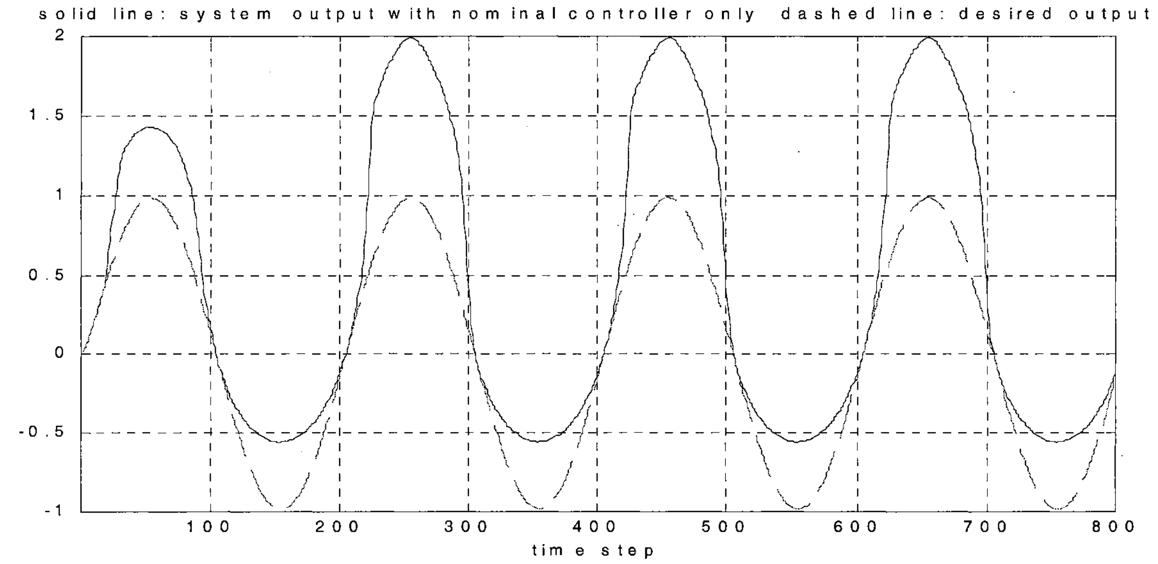


Figure 4.51 System response vs. desired output with nominal controller only (example 2; case 2)

4.2.2 Example 2

Consider the same nominal system model with different failure dynamics as shown in Equation (4.11),

$$y(k+1) = y(k) - \frac{\Delta t}{m}(cy(k)^3 + k_1 y(k-1)) + \frac{\Delta t}{m}u(k) + \sum_{i=1}^{n} \beta_i(k-T_i) f_i(y(k), y(k-1), u(k)), \tag{4.11}$$

$f_1(y(k), y(k-1)) = 0.5 \times \sin(y(k) \times y(k-1))$, $\beta_1(k-T_1) = U(k-T_1)$, where $T_1 = 20$, and

$f_2(y(k), u(k)) = 0.005 \times y(k) \times u(k)$, $\beta_2(k-T_2) = U(k-T_2)$, $T_2 = 100$, $n = 2$.

The first failure dynamics does not explicitly depend on the current control input. However, the second failure dynamics does. Apparently, the intelligent control technique in case 1 is not adequate to handle this problem. A 3-3-1 MLP network with argument, $[y(k), y(k-1), u(k)]$, as network input is used as the on-line estimator in this example. The number of data sets for the on-line training is set to be 20 and the Levenberg-Marquardt algorithm with Bayesian regularization is used in the training process. Figure 4.51 shows the system output when the nominal controller is applied alone. Appreciable performance error appears after the first fault happened and the error gets even larger after the second fault occurred. However, when the proposed intelligent control regulator is applied, the system performance is apparently improved. The simulation results together with the desired output are shown in Figure 4.52. The output prediction from the on-line estimator at each time step together with the actual failure dynamics are plotted in Figure 4.53 and the control input at each time step is shown in Figure 4.54. The S function is plotted in Figure 4.55. Figures 4.56-4.59 show the simulation result for the same failure situation in the consecutive incipient faults with the following time profiles:

$$\beta_1(k-T_1) = (1 - e^{-\alpha_1(k-T_1)})U(k-T_1) \text{ where } \alpha_1 = 0.05 \text{ and } T_1 = 20,$$

$$\beta_2(k-T_2) = (1-e^{-\alpha_2(k-T_2)})U(k-T_2) \text{ where } \alpha_2 = 0.05 \text{ and } T_2 = 100.$$

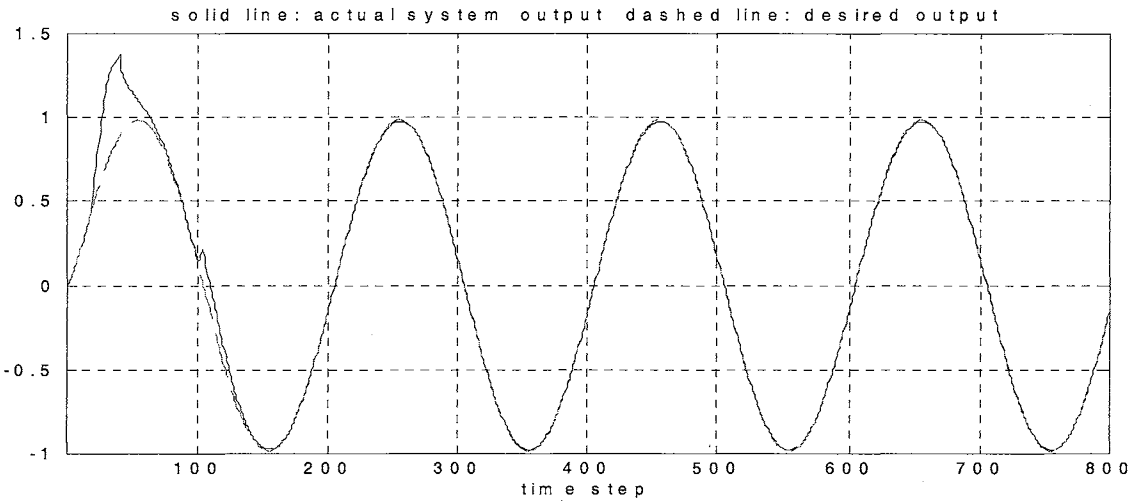


Figure 4.52 System response vs. desired output (20-data window; consecutive abrupt faults; example 2; case 2)

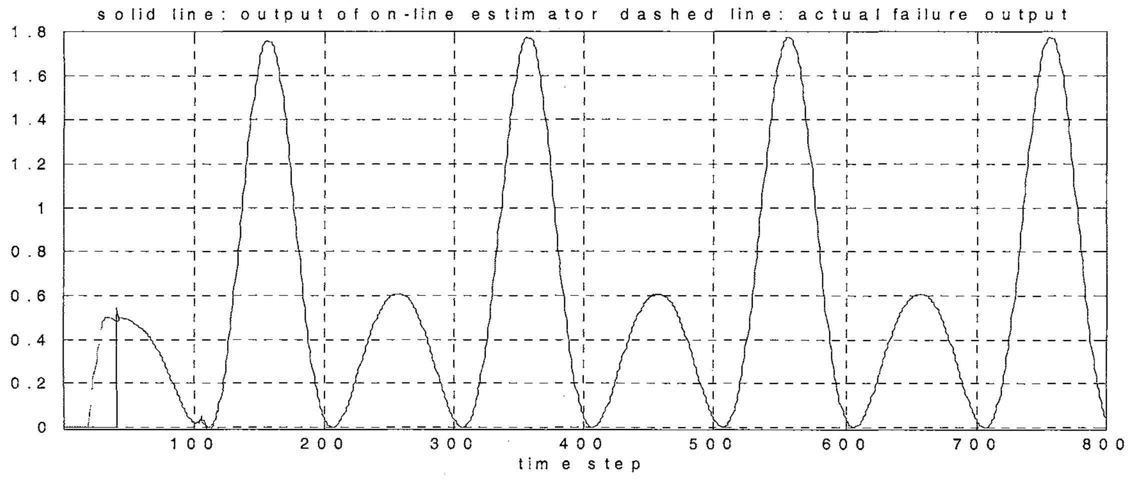


Figure 4.53 Output prediction from the on-line estimator vs. actual failure dynamics output (20-data window; consecutive abrupt faults; example 2; case 2)

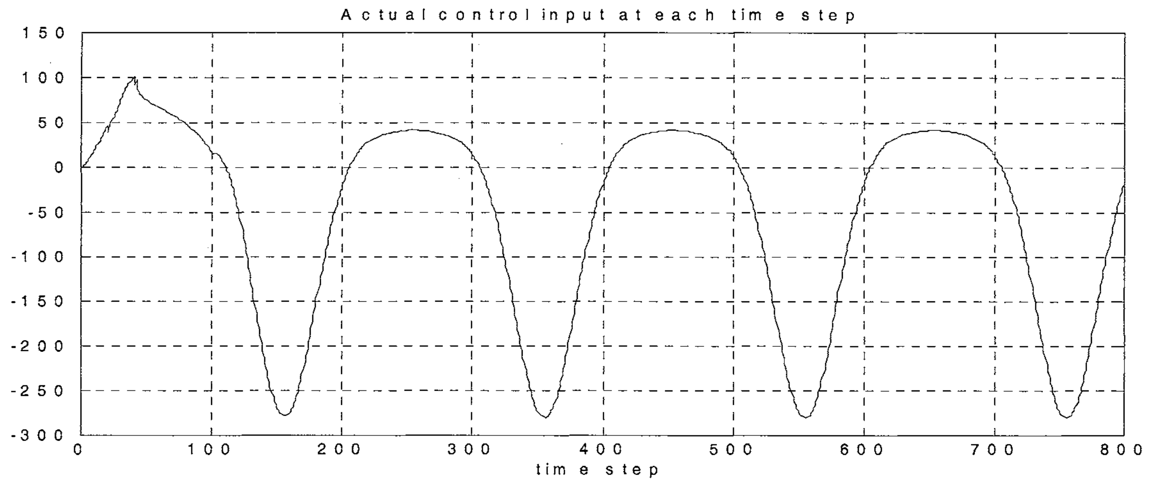


Figure 4.54 Actual control input (20-data window; consecutive abrupt faults; example 2; case 2)

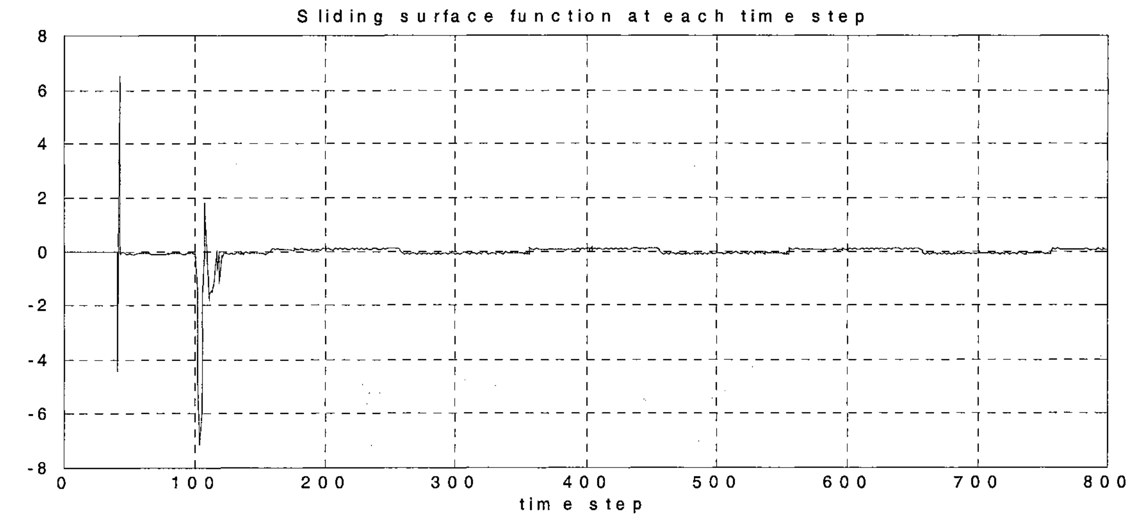


Figure 4.55 Sliding surface function at each time step (20-data window; consecutive abrupt faults; example 2; case 2)

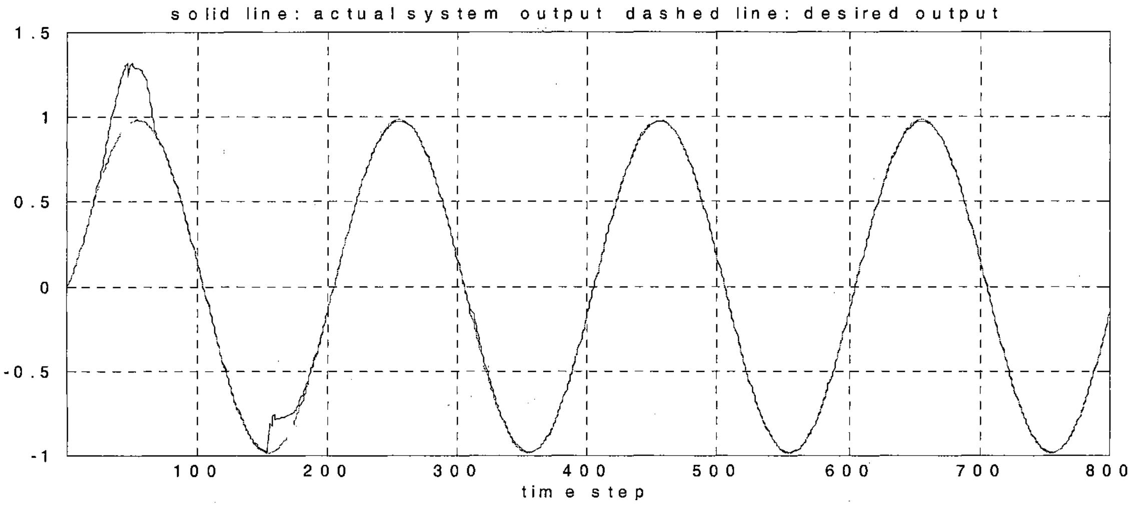


Figure 4.56 System response vs. desired output (20-data window; consecutive incipient faults; example 2; case 2)

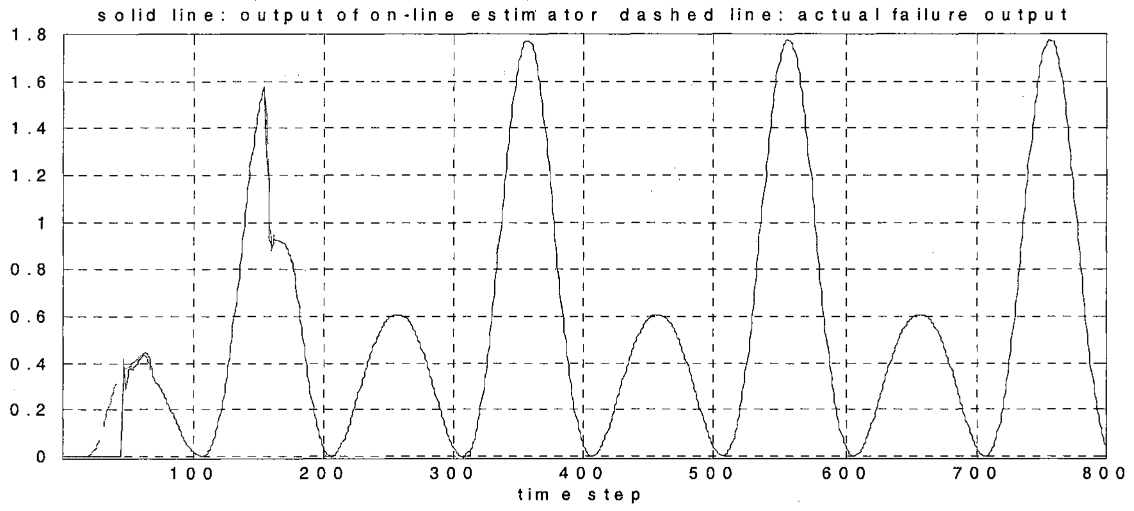


Figure 4.57 Output prediction from the on-line estimator vs. actual failure dynamics output (20-data window; consecutive incipient faults; example 2; case 2)

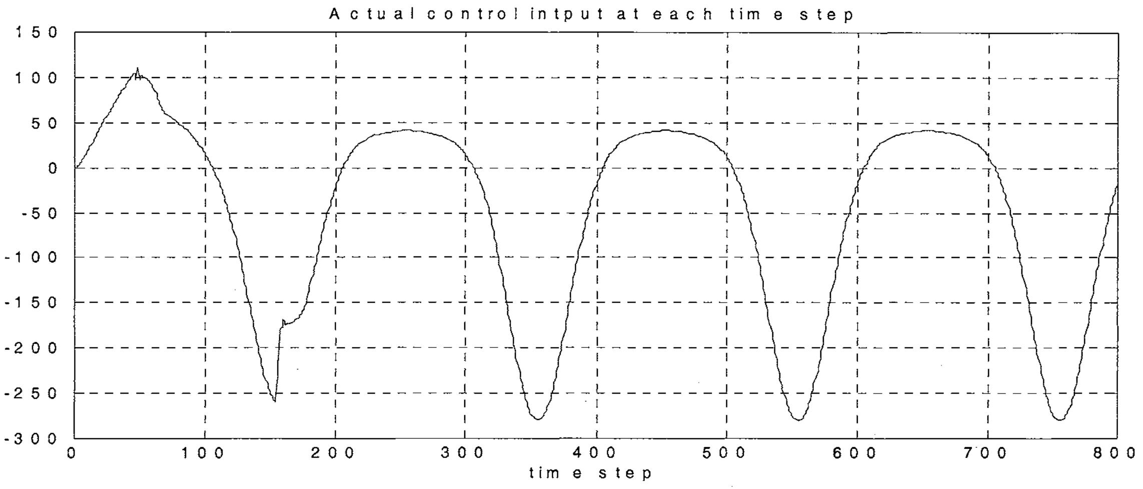


Figure 4.58 Actual control input (20-data window; consecutive incipient faults; example 2; case 2)

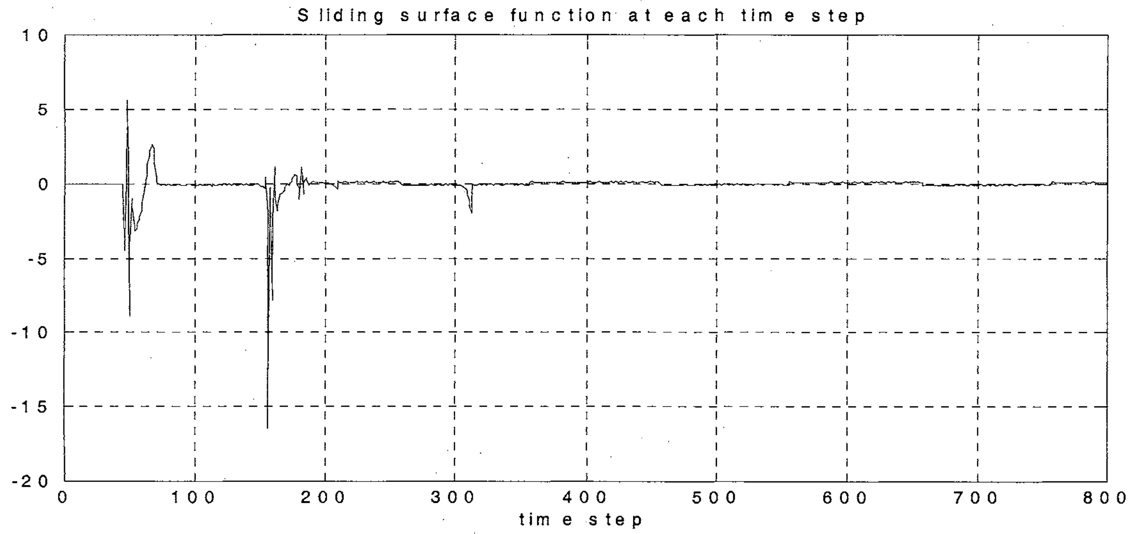


Figure 4.59 Sliding surface function at each time step
(20-data window; consecutive incipient faults; example 2; case 2)

<u>4.2.3 Simulation test in noisy measurement situations</u>

In order to test the proposed intelligent control strategy for case 2 in the noisy environment, the same Gaussian noise with zero mean and different variances are added to the actual output data as noisy measurements. The simulation tests are divided into 2 parts. The first part shows the system behavior, output predictions from the on-line estimator, noisy measurements and the added noises, sliding surface function, and actual control input for Example 1 (Subsection 4.2.1). The second part shows the simulation results for Example 2 (Subsection 4.2.2). The results of the first part for consecutive abrupt faults with variance 2.25×10^{-4} are shown in Figures 4.60-4.64. Apparently, the system performance degrades as the noise variance increases. In the second part, Figures 4.65-4.69 are the results for the consecutive incipient failures in example 2 with noise variance 2.25×10^{-4}.

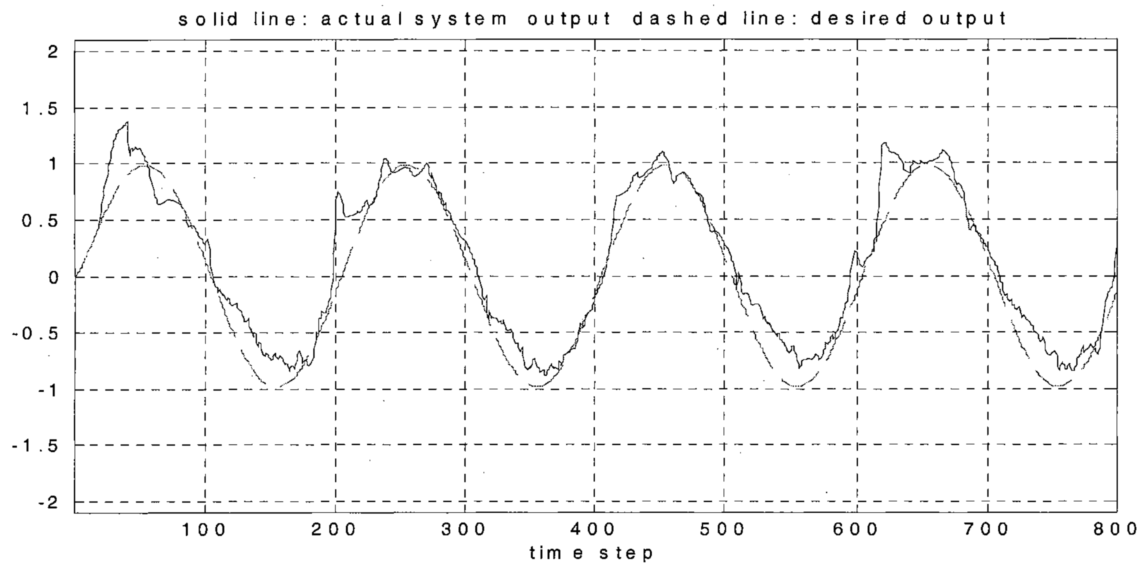


Figure 4.60 System response vs. desired output
(Gaussian white noise variance 2.25e-4; 20-data window; consecutive abrupt faults; example 1; case 2)

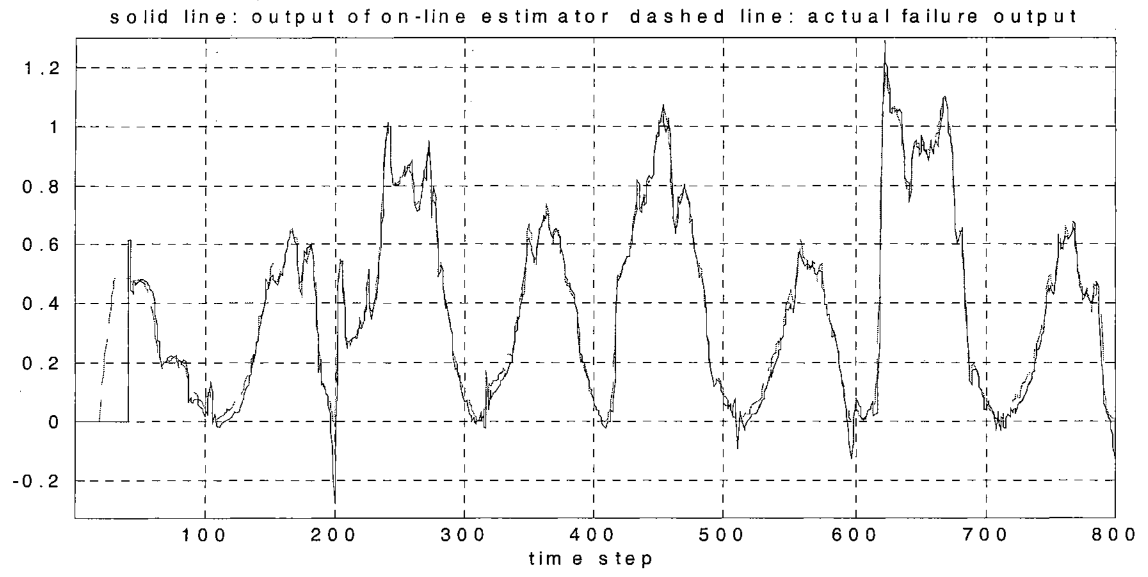


Figure 4.61 Output prediction from the on-line estimator vs. actual failure dynamics output
(Gaussian white noise variance 2.25e-4; 20-data window; consecutive abrupt faults; example 1; case 2)

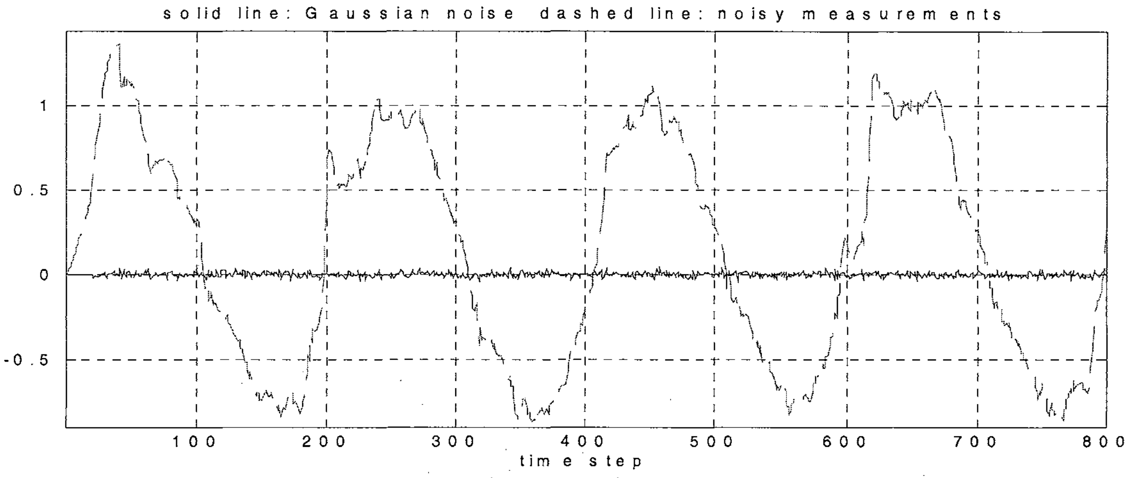


Figure 4.62 Actual noisy measurements vs. Gaussian noises
(Gaussian white noise variance 2.25e-4; 20-data window; consecutive abrupt faults; example 1; case 2)

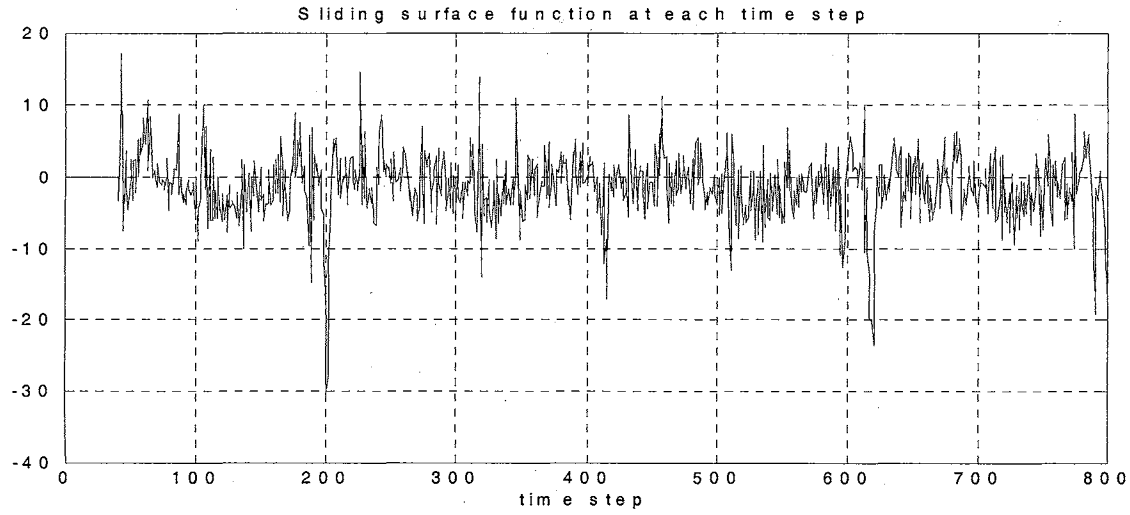


Figure 4.63 Sliding surface function at each time step
(Gaussian white noise variance 2.25e-4; 20-data window; consecutive abrupt faults; example 1; case 2)

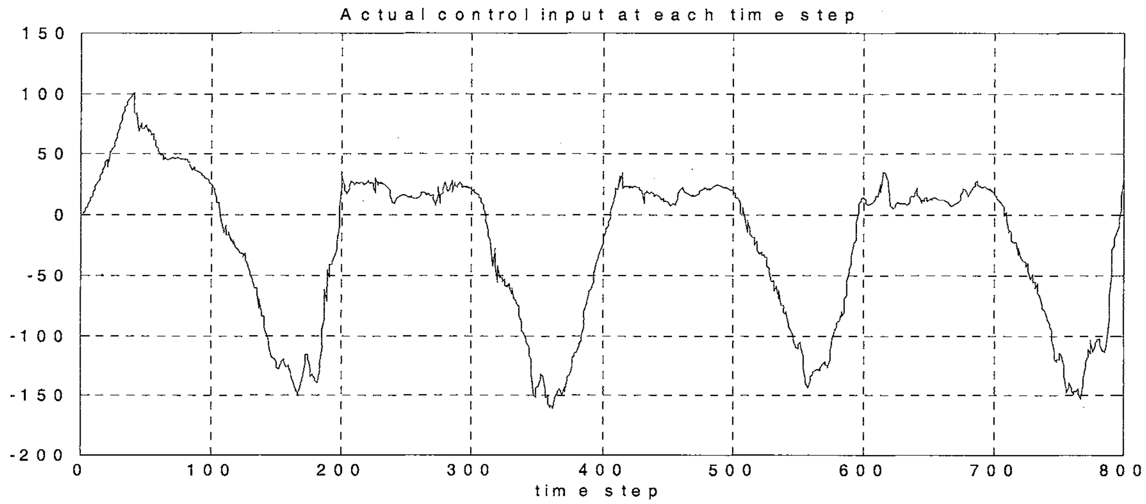


Figure 4.64 Actual control input
(Gaussian white noise variance 2.25e-4; 20-data window; consecutive abrupt faults; example 1; case 2)

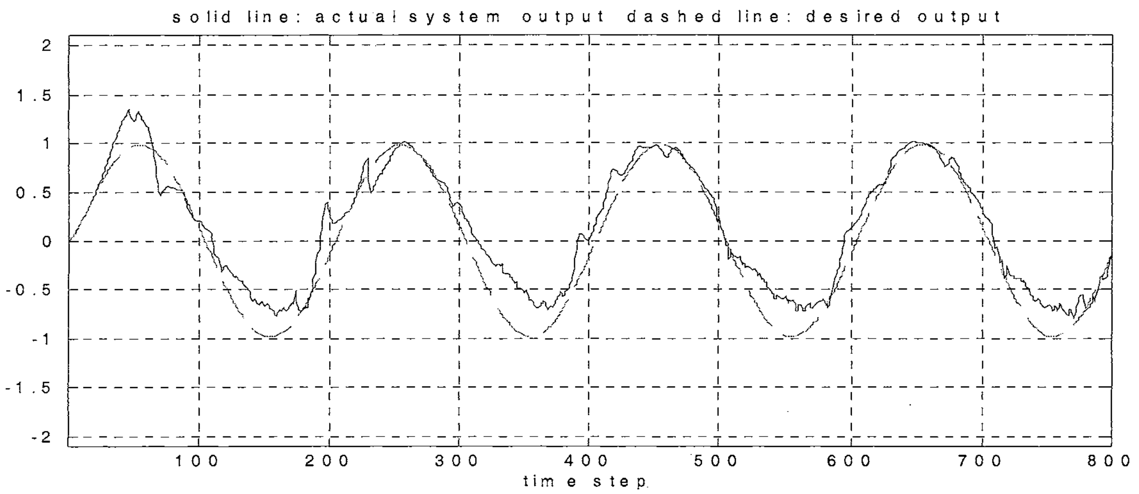


Figure 4.65 System response vs. desired output
(Gaussian white noise variance 2.25e-4; 20-data window; consecutive incipient faults; example 2; case 2)

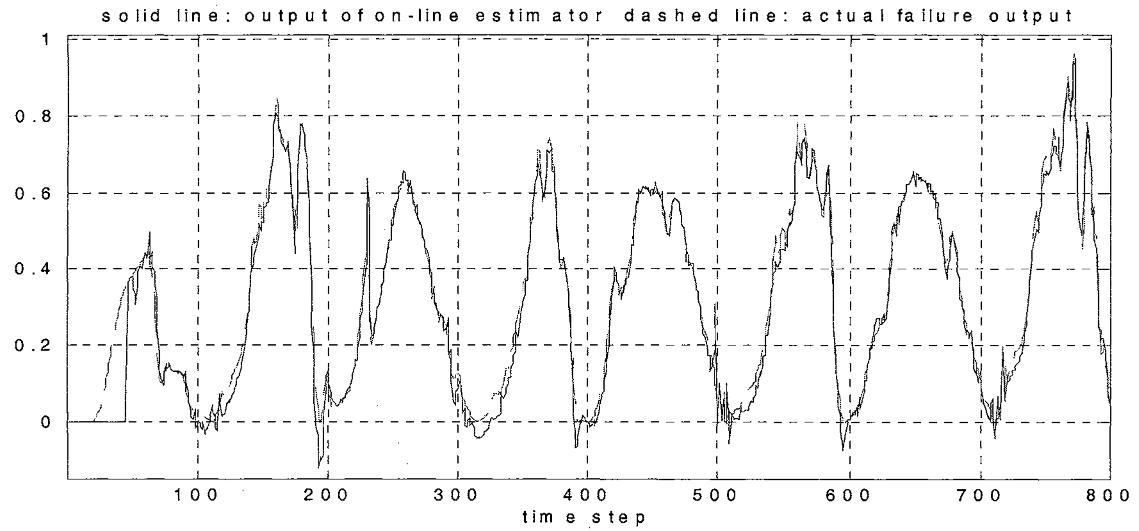


Figure 4.66 Output prediction from the on-line estimator vs. actual failure dynamics output
(Gaussian white noise variance 2.25e-4; 20-data window; consecutive incipient faults; example 2; case 2)

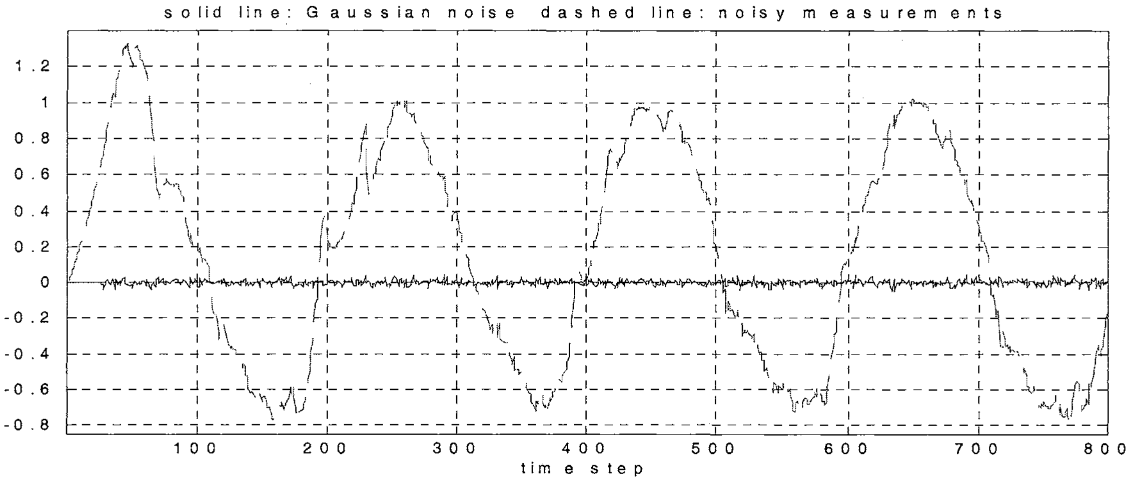


Figure 4.67 Actual noisy measurements vs. Gaussian noises
(Gaussian white noise variance 2.25e-4; 20-data window; consecutive incipient faults; example 2; case 2)

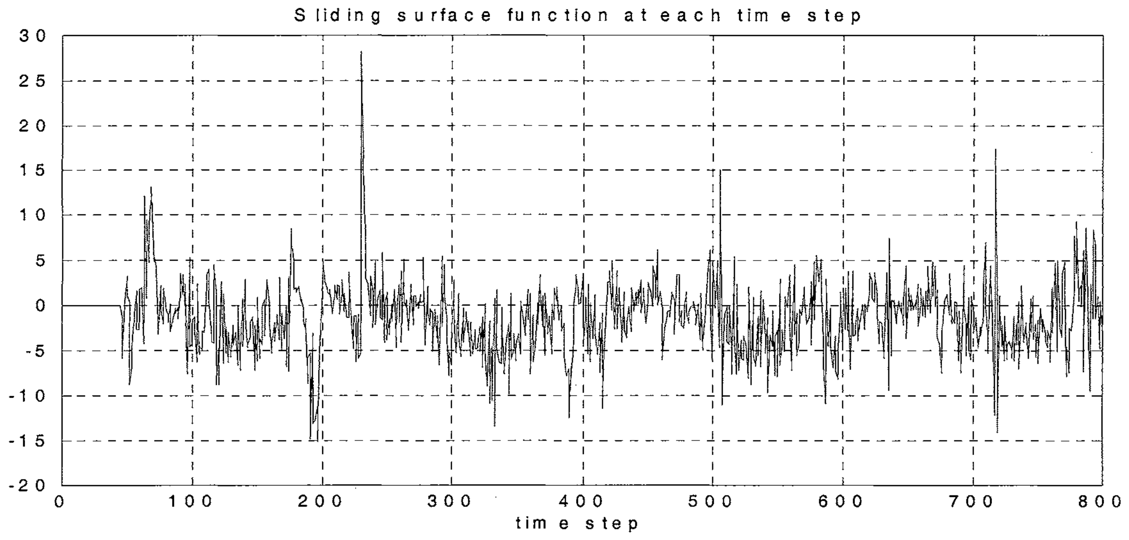


Figure 4.68 Sliding surface function at each time step
(Gaussian white noise variance 2.25e-4; 20-data window; consecutive incipient faults; example 2; case 2)

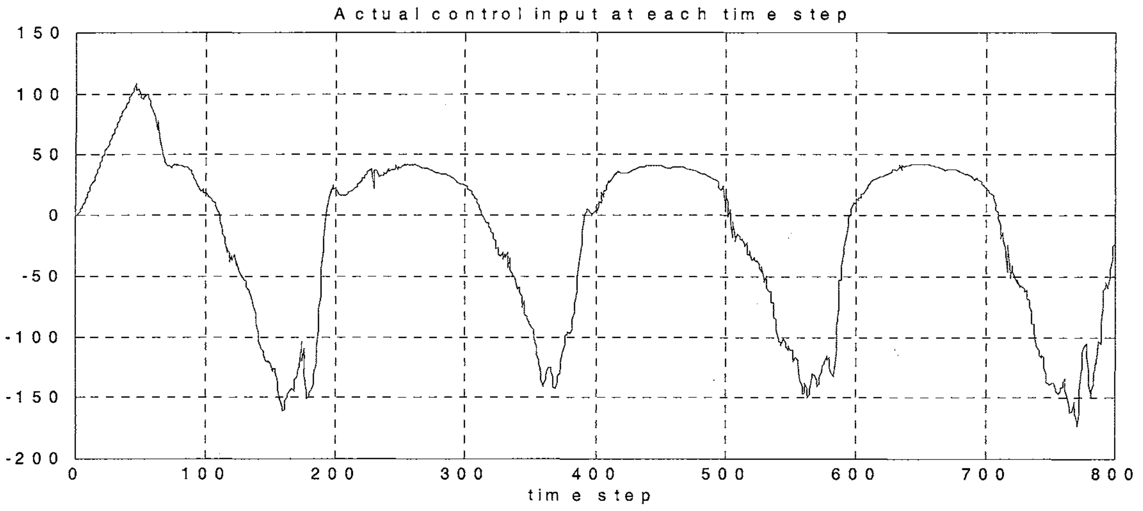


Figure 4.69 Actual control input
(Gaussian white noise variance 2.25e-4; 20-data window; consecutive incipient faults; example 2; case 2)

4.2.4 Comments and discussions

In Section 4.2, the simulation results of the intelligent control strategy for on-line fault tolerant control problems in the situations of case 2 are presented. According to the theoretical analysis presented in Chapter 3, the on-line fault tolerant control problems for unanticipated system failures of case 2 can be solved by deploying an estimator to identify the failure dynamics on-line and searching for the effective control action to satisfy inequalities (3.8) or (3.9). Among the numerous simulation tests, these expected results have been verified and the following observations can be drawn.

1. The design parameters such as the length of the time-shifting data window and training algorithms have substantial effect on the system performance. It is found that using 20 pairs of input-output measurements as the training pattern at each time step has a much better system performance than using 10-data window. Among all simulation tests, there does exist some situations where the system behavior is out of control when the design parameter, the length of the time-shifting data window, 10 is used. It is also obviously shown that using the training algorithm with regularization will generally result in a better system behavior than using the algorithm without regularization since this technique can relax the network over-fitting problem and eliminate the guesswork in determining the optimal network structure [65,66].
2. Although the scenario with large noise is virtually equivalent to the situation with sensor failure where all the measurements have no actual meaning, the system performance still seems to be sensitive to the noise. It is found that the performance error increases quickly in the environment with the high variance white noise and the possibility of system instability increases significantly when the noise variance is larger than 1.0×10^{-3}. Apparently, it is a reasonable result and fact that the contaminated noisy data have significant negative influence in both the computation of the sliding surface function and the searching process of the effective control signal.

4.3 CASE 3

4.3.1 Example 1: time-varying abrupt-incipient failures

The system dynamics under unknown multiple-failure modes is represented by Equation (4.12),

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + u(k)^2 - 20u(k) + \sum_{i=1}^{n} \beta_i(k-T_i) f_i(y(k), y(k-1)), \quad (4.12)$$

where the failure dynamics are defined as

$n = 2$, $f_1(y(k)) = 0.05 \times \frac{k}{20} \times \cos(y(k))$, $\beta_1(k - T_1) = U(k - T_1)$, $T_1 = 20$, $f_2(y(k), y(k-1)) = 0.6 \times \sqrt{|y(k) \times y(k-1)|}$, $\beta_2(k - T_2) = 1 - e^{-0.05(k-100)} U(k - T_2)$, and $T_2 = 103$.

The nominal system dynamics is first realized by a 3-30-1 MLP neural network. Training patterns are collected by feeding 2,000 uniformly distributed random input signals varying from -1.5 to 1.5 into the nominal system and, after a normalization process of the training patterns, the Levenberg-Marquardt algorithm with Bayesian regularization is used to train the NN nominal model off-line [66]. The corresponding nominal controller can be developed using dynamical backpropagation or backpropagation-through-time algorithm [31-34]. However, these two methods are computationally expensive and complex since the training process of the NN nominal controller requires the realization of the NN nominal model. In this simulation, the nominal controller is developed off-line based upon the same technique discussed in Chapter 3. Through the realization of the NN nominal model, the effective control input to reduce the S function is, first, searched, and then, the input-output patterns is collected for the training process of the NN nominal controller. In this way, the training of the NN nominal controller does not require knowledge of the nominal model and the complexity of the dynamical backpropagation or backpropagation-through-time algorithm can be avoided (i.e., the training of the NN nominal controller requires only the static backpropagation algorithm.).

Another 3-2-2-1 MLP network is deployed as the on-line estimator and the on-line learning algorithm used in this case is the static non-batch form backpropagation algorithm (i.e., same as those used in case 1) [57]. Figures 4.71-4.73 show part of the on-line simulation results. As clearly shown, the system failures have been properly accommodated such that the system output has been driven back to the desired trajectory while the system behavior under nominal control law without appropriate adjustment of the control action is eventually out of control due to the time-varying failure dynamics as shown in Figure 4.70. Observing Figure 4.73 closely, we find that the effective control input generated by the intelligent control regulator still keeps changing without periodical pattern within 800 time steps, which is obviously because the regulator keeps adjusting the control input to compensate the time-varying system dynamics.

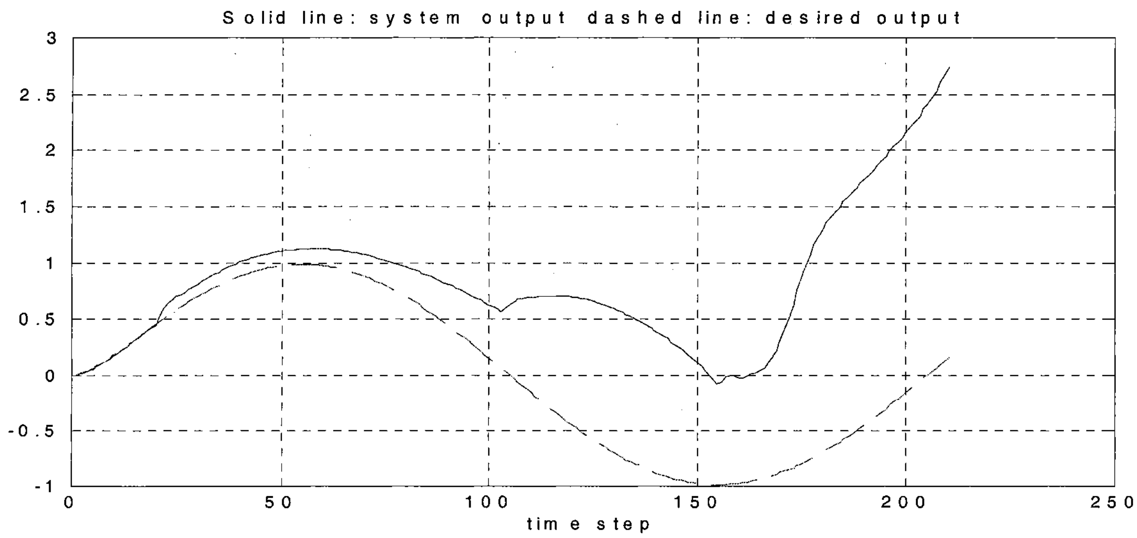


Figure 4.70 System response vs. desired output with nominal controller (abrupt-incipient faults; example 1; case 3)

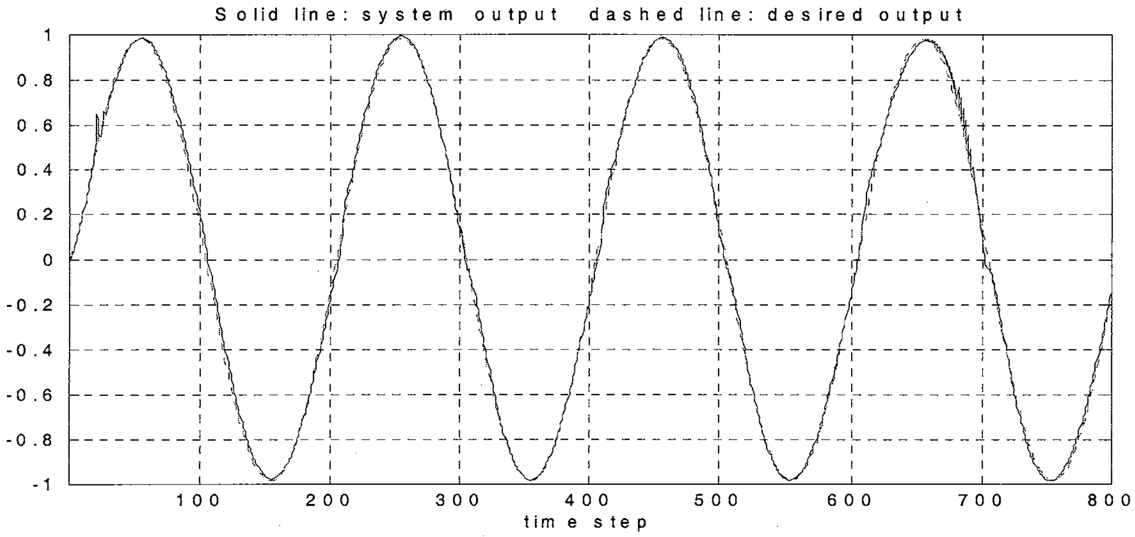


Figure 4.71 System response vs. desired output (abrupt-incipient faults; example 1; case 3)

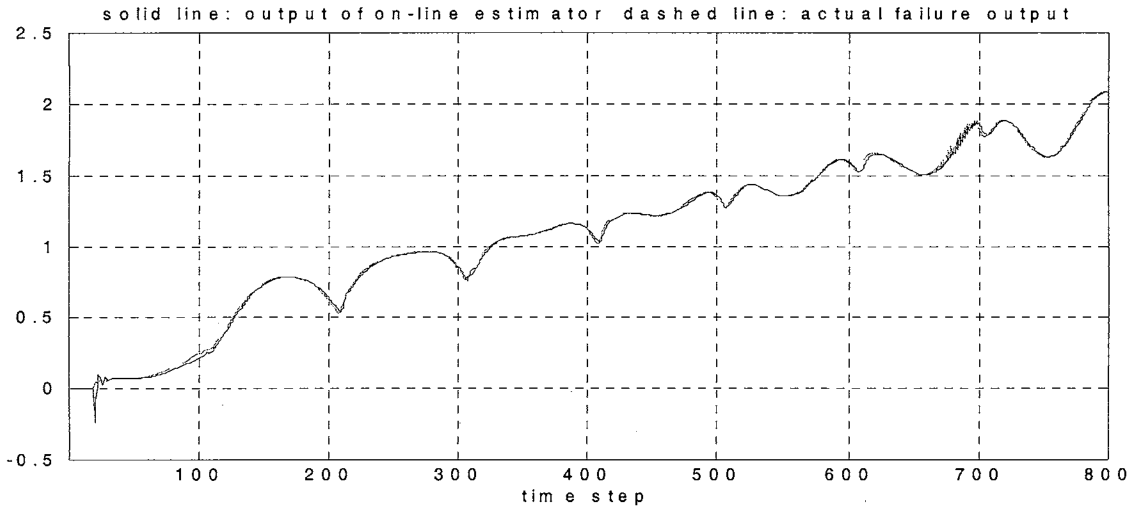


Figure 4.72 Output prediction from the on-line estimator vs. actual failure dynamics output (abrupt-incipient faults; example 1; case 3)

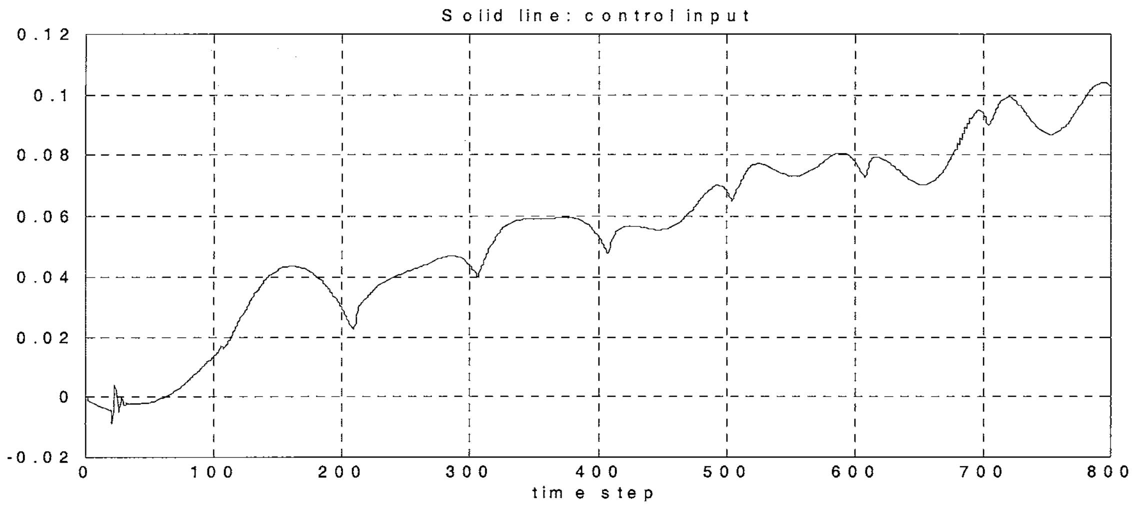


Figure 4.73 Actual control input (abrupt-incipient faults; example 1; case 3)

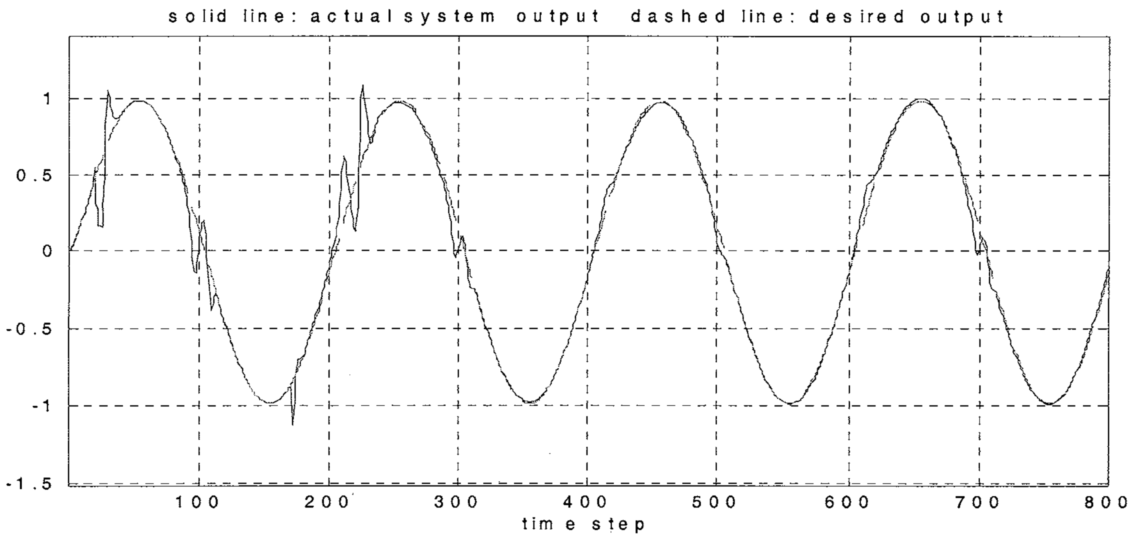


Figure 4.74 System response vs. desired output (consecutive abrupt faults; example 2; case 3)

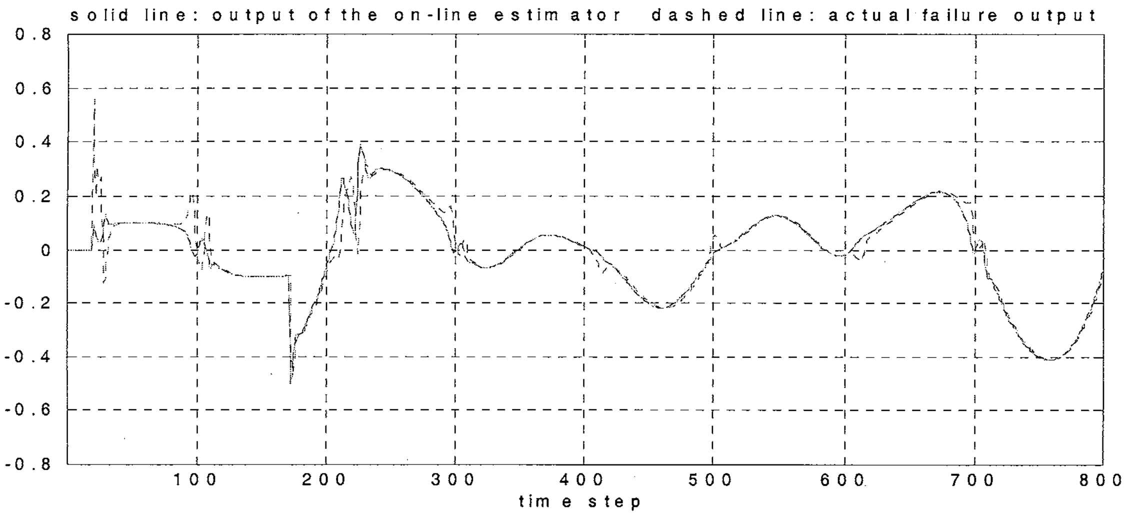


Figure 4.75 Output prediction from the on-line estimator vs. actual failure dynamics output (consecutive abrupt faults; example 2; case 3)

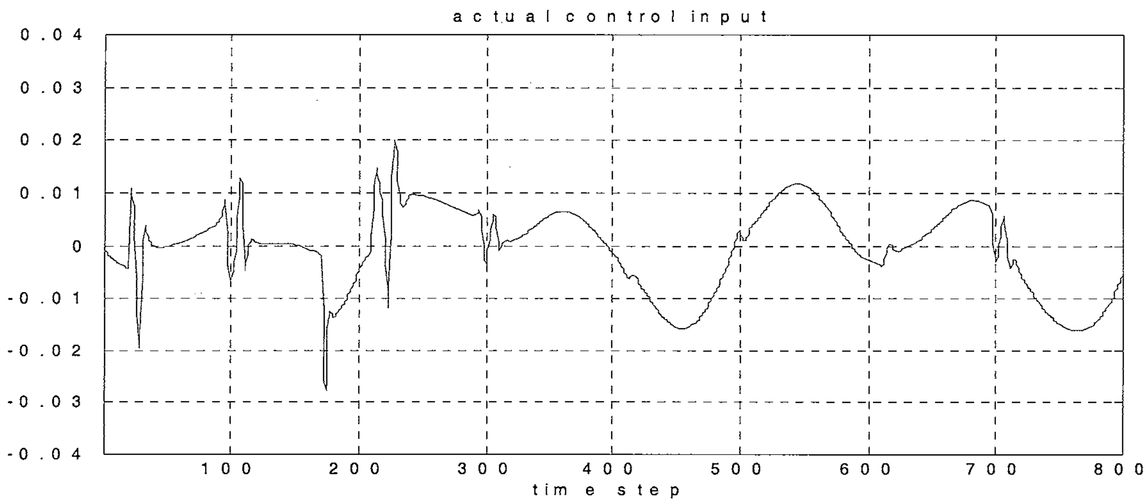


Figure 4.76 Actual control input (consecutive abrupt faults; example 2; case 3)

4.3.2 Example 2: time-varying abrupt failures

Consider the same nominal system with different time-varying failure dynamics as follows:

$n=2$, $f_1(y(k),y(k-1)) = 0.2\times\frac{y(k)}{(y(k-1)^2+1)}$, $\beta_1(k-T_1)=U(k-T_1)$, $T_1=20$, and

$f_2(y(k)) = 0.33\times y(k)\times\sin(\frac{k}{100})$, $\beta_1(k-T_2)=U(k-T_2)$, and $T_2=171$.

The same NN nominal model, NN on-line estimator, and the training algorithm as Example 1 in Subsection 4.3.1 are used here. Figures 4.74-4.76 show the simulation results when the system suffers from consecutive time-varying abrupt faults. The structure of the NN on-line estimator used in this case is much simpler than those used in case 1, where a 2-30-30-1 neural network was deployed as the on-line estimator. However, the simulation tests in both examples show positive results. This indicates that the on-line tracking performance is not very sensitive to the network structure while the static backpropagation algorithm is used for the parameter adjustments. The only significant difference is the pre-selection learning rate of the training algorithm. A bigger structure NN should use a much smaller learning rate since more parameters are adjusted at the same time to reduce the tracking error. Obviously, the simpler structure is always preferable since it demands less computational cost and possesses less chance of overfitting or falling into a local minimum.

4.3.3 Comments and discussions

In both examples, it is assumed that the NN nominal model is accurate enough to represent the actual nominal system dynamics even when the system is in failure modes. Under this condition, the effective control input to accommodate the failure dynamics can be found through the realization of the NN nominal model. If the desired actual effective control actions are far away from the validated domain of the NN nominal model, the similar extrapolation problem discussed in Chapter 3 (i.e., Subsection 3.2.2) may emerge to degrade the system performance or stability. Thus, it is important and necessary to maintain the validated domain of the nominal model as large as possible for a better and larger failure accommodation coverage.

4.4 CASE 4

4.4.1 Example 1

The system dynamics under unknown multiple-failure modes is represented by Equation (4.13),

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + u(k)^2 - 5u(k) + \sum_{i=1}^{n} \beta_i(k-T_i) f_i(y(k), y(k-1), u(k)), \tag{4.13}$$

where the failure dynamics is defined as

$n = 2$, $f_1(y(k), u(k)) = 0.1 \times \frac{k-25}{20} \times y(k) \times \cos(u(k))$, $\beta_1(k-T_1) = U(k-T_1)$, $T_1 = 25$, $f_2(y(k), y(k-1)) = 0.6 \times y(k) \times y(k-1)$, $\beta_2(k-T_2) = U(k-T_2)$, and $T_2 = 201$.

Similar to case 3, a 3-30-1 MLP network is first used to realize the nominal system dynamics. A 3-3-1 MLP network is then used to approximate the failure dynamics on-line with the same training algorithm in case 2. Figures 4.78-4.79 show part of the on-line simulation results. As shown, the system failures have been properly accommodated when the proposed intelligent control law is applied, while the system response under the nominal control law is shown in Figure 4.77. Similar to case 3, the effective control input keeps changing to accommodate the time-varying failures while the nominal control law cannot place the behavior of the time-varying system under control.

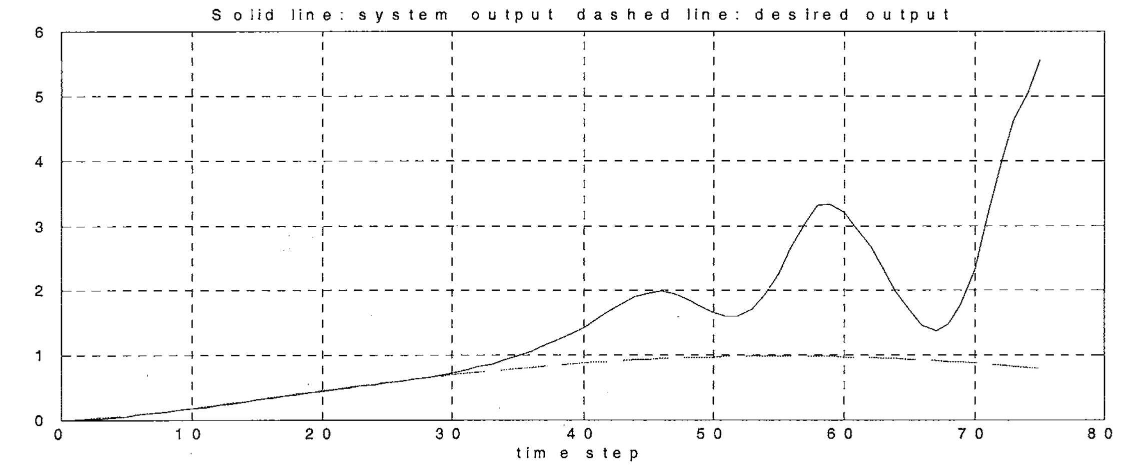


Figure 4.77 System response vs. desired output with nominal controller (consecutive abrupt faults; example 1; case 4)

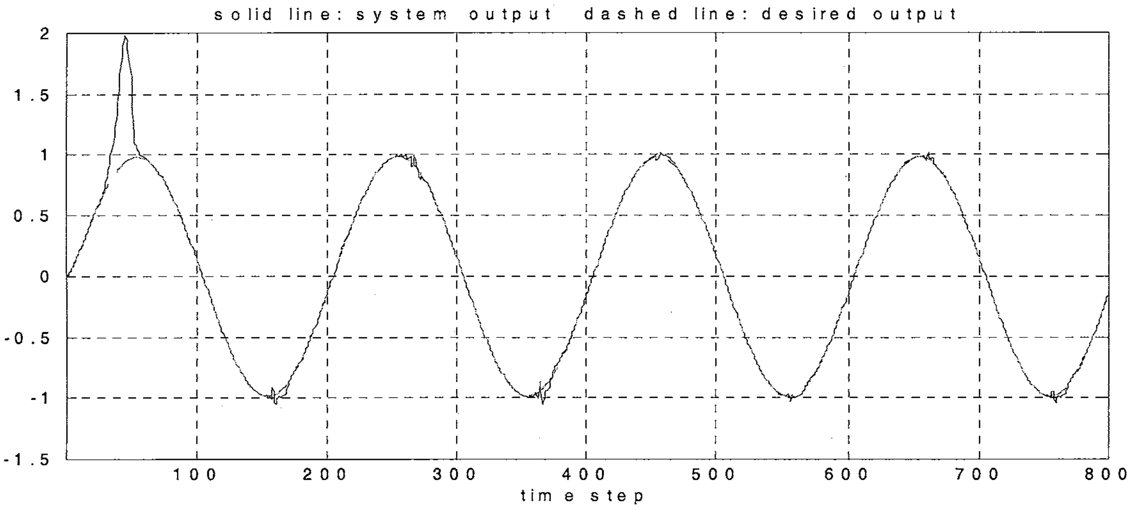


Figure 4.78 System response vs. desired output (20-data window; consecutive abrupt faults; example 1; case 4)

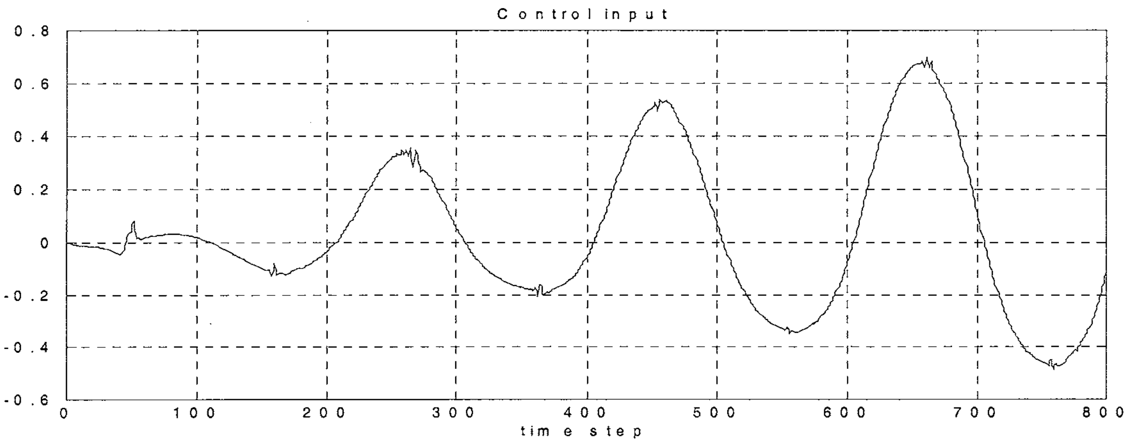


Figure 4.79 Actual control input (20-data window; consecutive abrupt faults; example 1; case 4)

4.4.2 Example 2

Consider the same nominal system model with different failure dynamics as follows:

$$\beta_1(k-T_1)\times\{-[0.3y(k)+0.6y(k-1)+2u(k)^2+5u(k)]+\beta_2(k-T_2)\times 0.6\times y(k)\times y(k-1)+ 0.05\times\frac{k}{20}\times y(k)^2+4u(k)\},$$

where $\beta_1(k-T_1)=U(k-T_1)$, $T_1=25$, $\beta_2(k-T_2)=U(k-T_2)$, and $T_2=171$. Starting from time step, 25, the first failure will cancel the nominal system dynamics and replace it with a totally different time-varying system. At time step 171, the system dynamics will suddenly change dramatically again due to the second failure. Figure 4.80 is the on-line system response plot. A large deviation of the response starts right after the sudden change of the system. However, within 25 time steps, the failure condition has been properly controlled such that the system output is driven back to the desired trajectory until the second failure happens at time step 171. It takes almost the same time for the on-line estimator to catch up with the failure and place the system behavior under control. Figure 4.81 shows the actual control input at each time step.

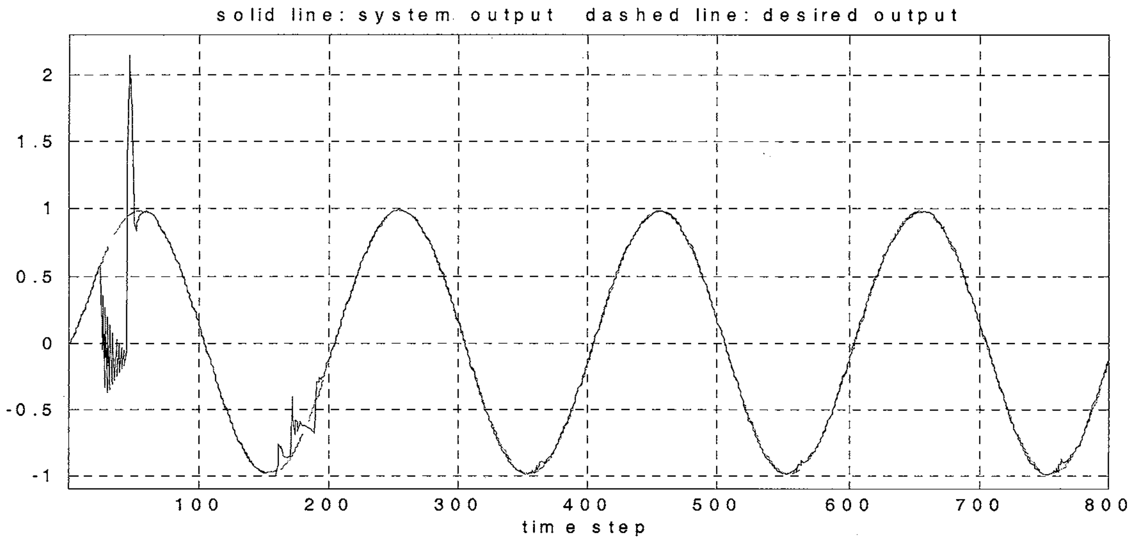


Figure 4.80 System response vs. desired output (20-data window; consecutive abrupt faults; example 2; case 4)

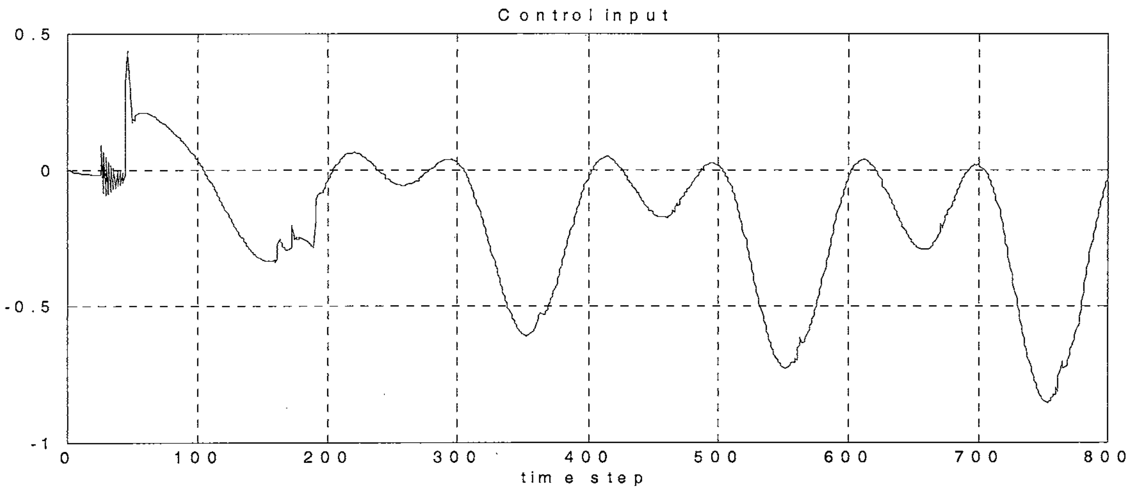


Figure 4.81 Control input (20-data window; consecutive abrupt faults; example 2; case 4)

4.4.3 Example 3: On-line multiple-failures accommodation for a multiple-input multiple-output system

Consider a MIMO system under different failures as shown in Equation (4.14),

$$\begin{aligned}
y_1(k+1) &= \frac{y_1(k)}{1+y_2(k)^2} + u_1(k)^2 + u_2(k)^2 - 16u_1(k) - 20u_2(k) + \Delta f_1(k), \\
y_2(k+1) &= \frac{y_1(k)y_2(k)}{1+y_2(k)^2} + u_1(k)u_2(k) + 20u_1(k) - 5u_2(k) + \Delta f_2(k), \\
\Delta f_1(k) &= \beta_{10}(k-T_{10}) \times 0.1 \times \frac{k-25}{20} y_1(k)\cos(u_1(k)) + \beta_{11}(k-T_{11}) \times 0.6 y_1(k) y_2(k), \\
\Delta f_2(k) &= \beta_{20}(k-T_{20}) \times 0.1 \times y_1(k) y_2(k),
\end{aligned} \tag{4.14}$$

where $\beta_{10}(k-T_{10}) = U(k-T_{10})$, $\beta_{11}(k-T_{11}) = U(k-T_{11})$, $\beta_{20}(k-T_{20}) = U(k-T_{20})$, $T_{10} = 25$, $T_{20} = 15$, and $T_{11} = 123$. The nominal system is first realized through a 4-75-2 MLP network. 2,000 input-output training patterns are collected by supplying uniformly distributed random inputs varying from –1.5 to 1.5. A 3-4-2 MLP network is chosen as the on-line estimator and the Levenberg-Marquardt with Bayesian regularization algorithm is used in the training process for both the NN nominal model and the NN on-line estimator (i.e., a 4-40-2 MLP network is used as a nominal controller trained off-line). Figures 4.82 and 4.83 show the system response for the first output and the response for the second output together with the desired outputs, respectively, while the nominal controller alone fails to maintain the system stability under multiple failures. A relatively large tracking error appears in both system outputs around time step, 110. The same results are observed in Figures 4.84-4.85 and Figures 4.86-4.87 which are the plots for the on-line estimations and the sliding surface functions, respectively. This indicates that a relatively large estimation error has occurred at that time.

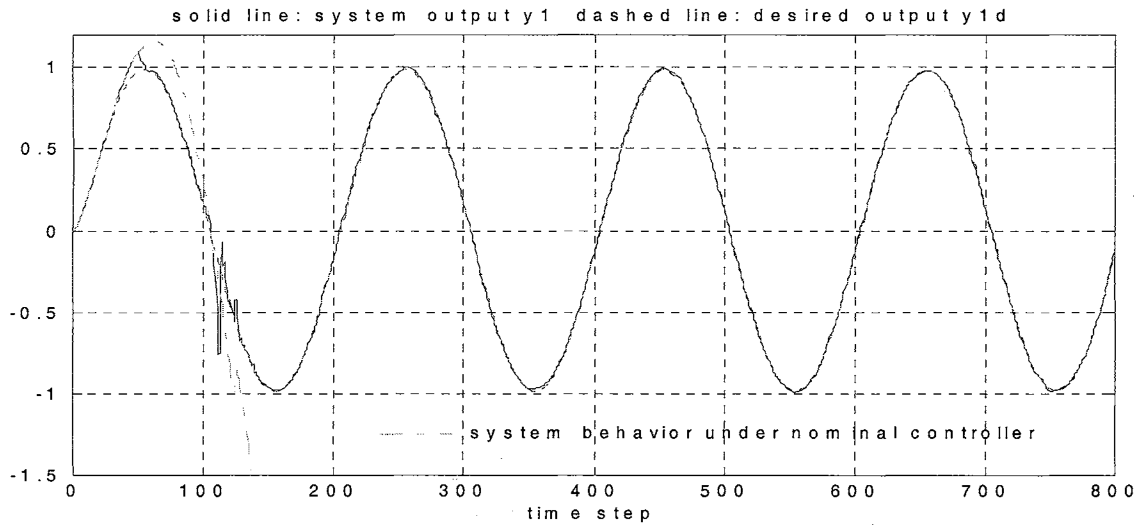


Figure 4.82 System response, y1, vs. desired output, y1d
(MIMO system; 25-data window; consecutive abrupt faults; example 3; case 4)

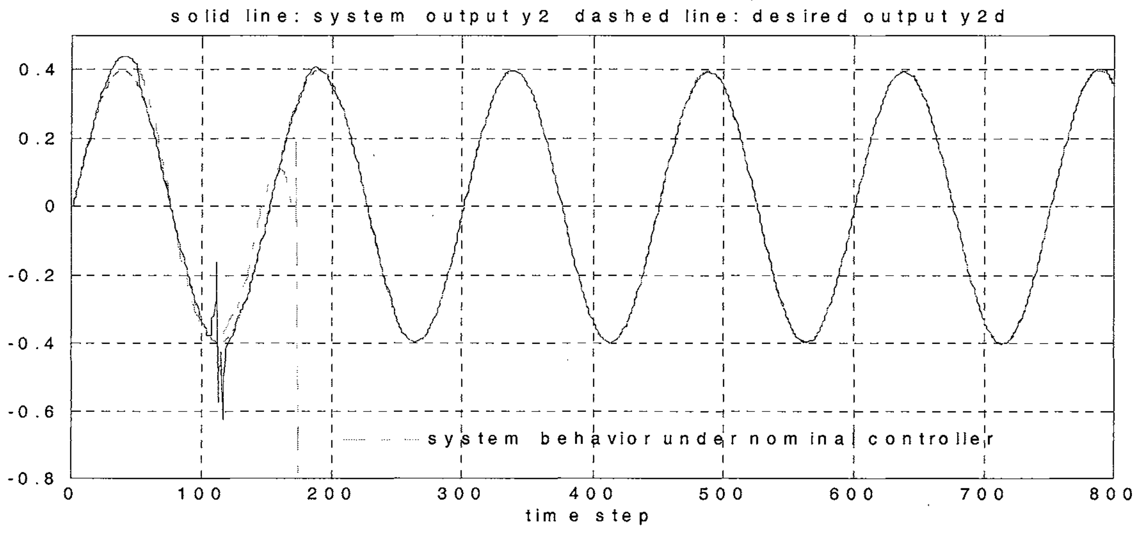


Figure 4.83 System response, y2, vs. desired output, y2d
(MIMO system; 25-data window; consecutive abrupt faults; example 3; case 4)

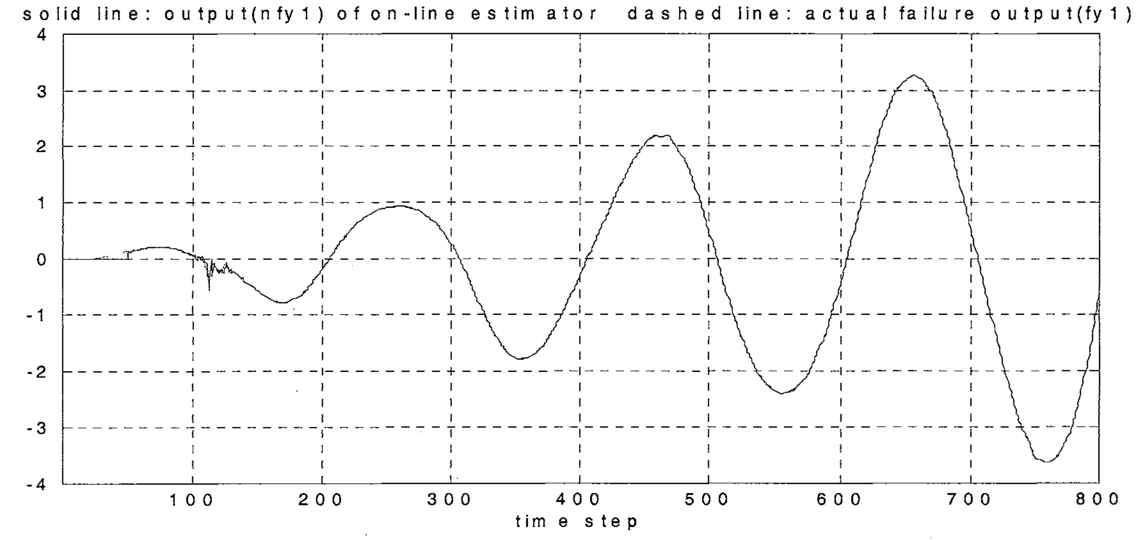


Figure 4.84 Output prediction, nfy1, from the on-line estimator vs. actual failure dynamics output, fy1 (MIMO system; 25-data window; consecutive abrupt faults; example 3; case 4)

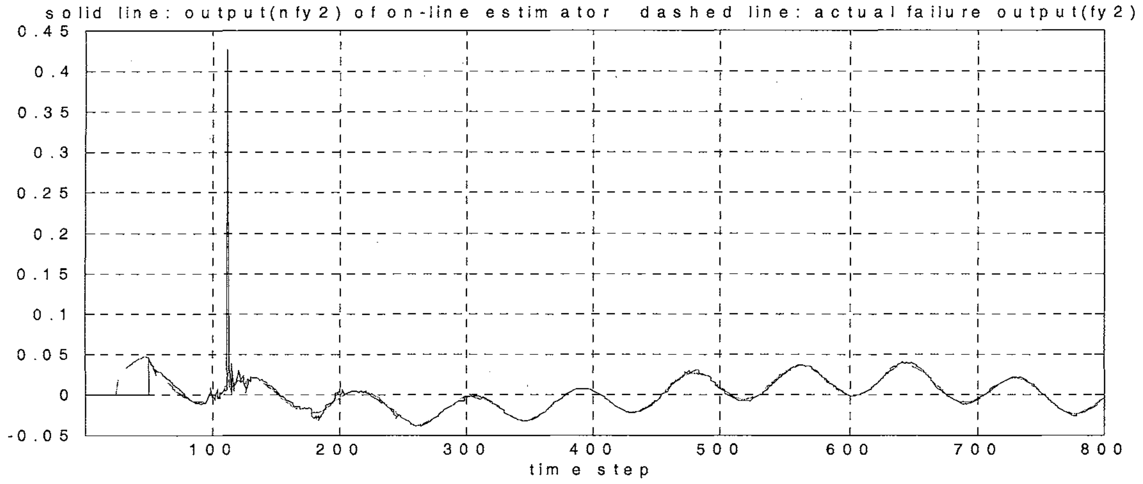


Figure 4.85 Output prediction, nfy2, from the on-line estimator vs. actual failure dynamics output, fy2 (MIMO system; 25-data window; consecutive abrupt faults; example 3; case 4)

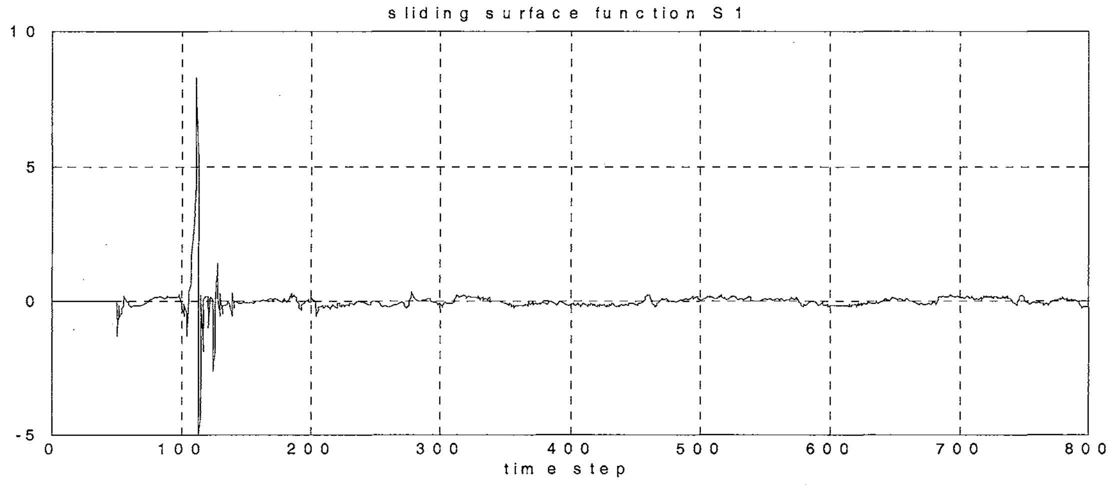


Figure 4.86 Sliding surface function S1 (MIMO system; 25-data window; consecutive abrupt faults; example 3; case 4)

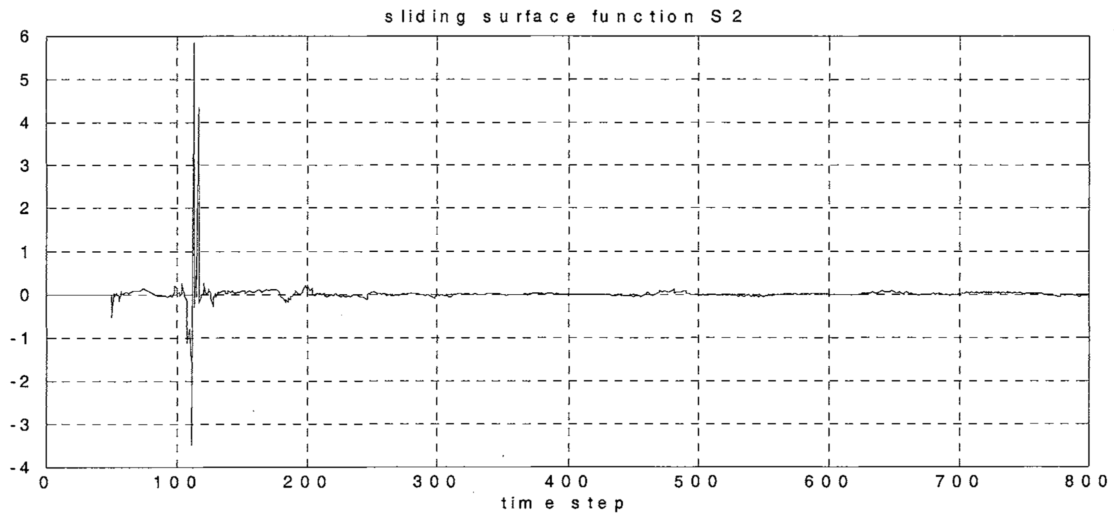


Figure 4.87 Sliding surface function S2
(MIMO system; 25-data window; consecutive abrupt faults; example 3; case 4)

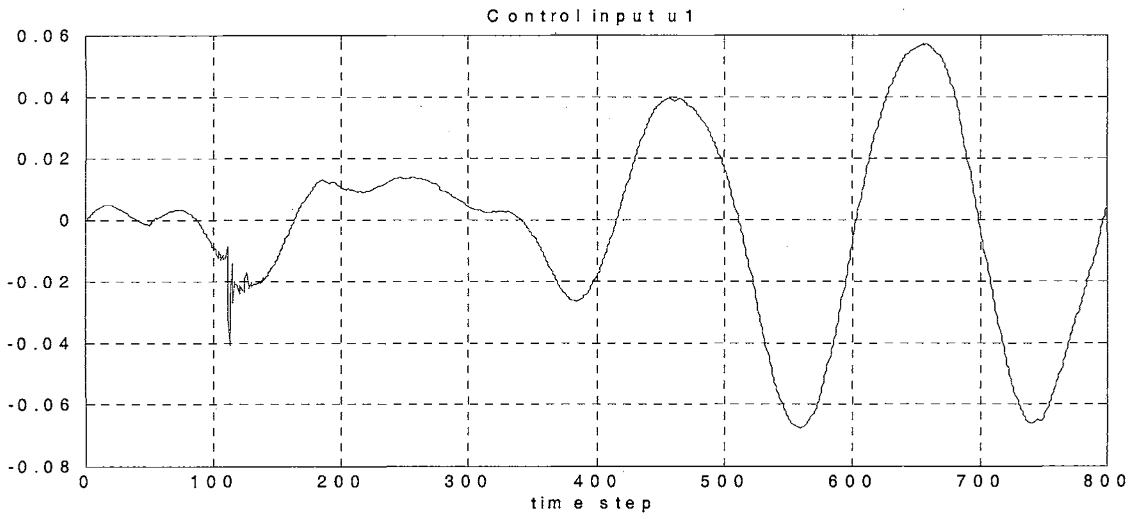


Figure 4.88 Control input u1 (MIMO system; 25-data window; consecutive abrupt faults; example 3; case 4)

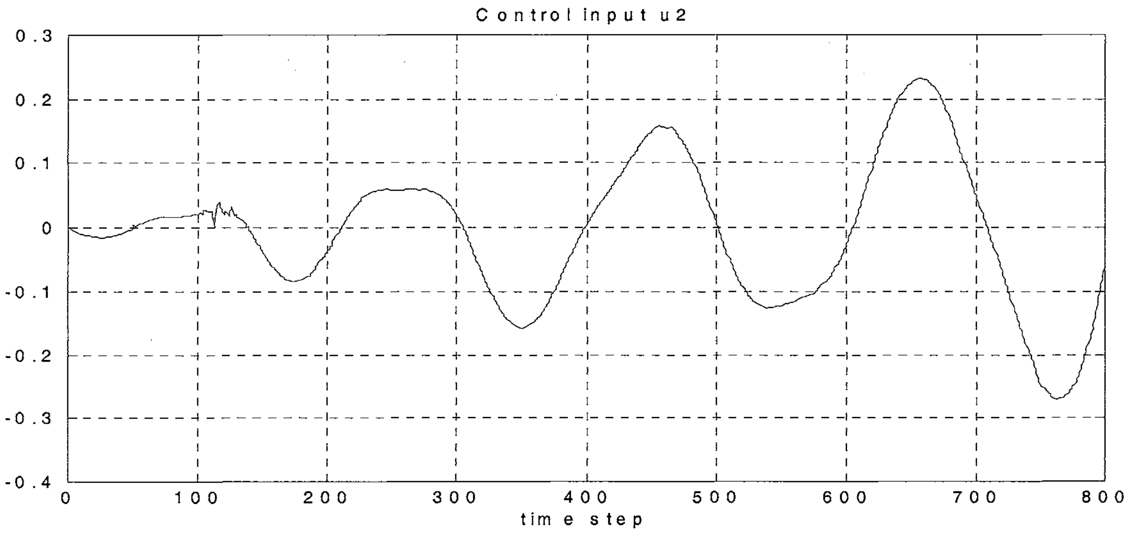


Figure 4.89 Control input u2 (MIMO system; 25-data window; consecutive abrupt faults; example 3; case 4)

The actual control inputs are shown in Figures 4.88-4.89. In this example, the length of the time-shifting data window is selected as 25. Simulation tests show that this is a better trade-off number between the system performance and the computational complexity. A simple mean value of the two estimated gradient directions realized through the NN on-line estimator is used for the searching of the effective control inputs. This approach is based upon the assumption that the searching directions of the effective control signals to accommodate failure dynamics have no confliction, which may not always be true under unanticipated catastrophic system failures. In general, this becomes a multi-objective optimization problem that remains to be an open research issue [76].

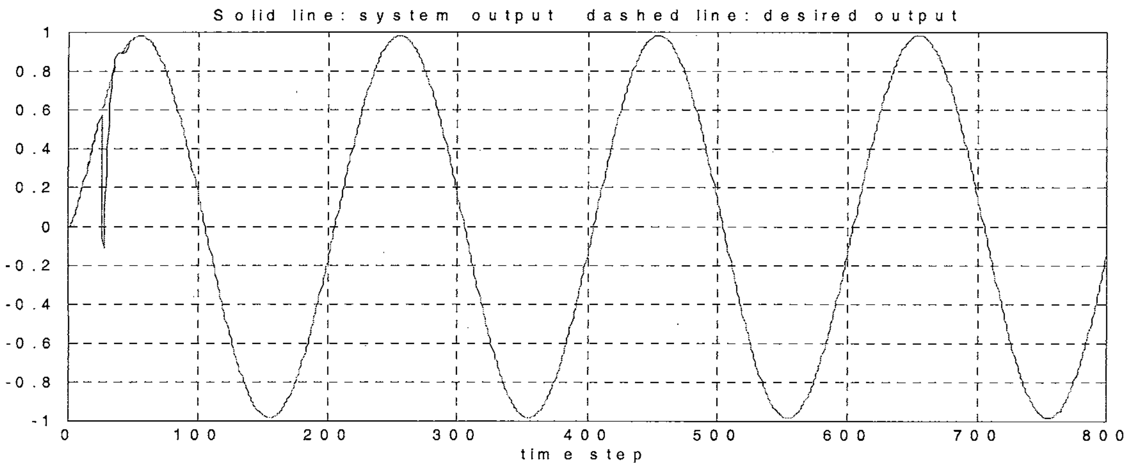


Figure 4.90 System response vs. desired output in false alarm situation (consecutive abrupt faults; example 2; case 4)

4.4.4 False alarm

In order to test the false alarm situations, the same system with the same failures in Example 2 of case 4 is used and, right after the system abnormal behavior detected, the failures are eliminated by setting the time varying constant gains of the failures to zeros. Figure 4.90 shows the on-line system response plot under the false alarm noise-free situation. As expected, the on-line estimator tries to approximate the differences between the outputs of the NN nominal model and the actual measurements, which is the remaining uncertainty between the actual nominal system and the NN nominal model.

4.4.5 Comments and discussions

The proposed on-line fault detection scheme has good resistance in miss detection of system failures. However, it also increases the sensitivity to the false alarm situations. Simulation tests in noise-free false alarm situations indicate that the on-line estimator is used to approximate the remaining uncertainty of the system. In noisy environments, it is possible for the on-line estimator to overfit the noisy model. Thus, the pre-processing of the contaminated noisy measurements becomes an important process for better results.

CHAPTER V

REAL-TIME HARDWARE EXPERIMENT FOR UNANTICIPATED FAULT ACCOMMODATION

As discussed before, several important design parameters will affect the failure accommodation performance in on-line situations. In general, the selection of those parameters is a system-dependent problem that requires the considerations of performance criteria and system computing capacity. To obtain a comprehensive insight for quantification of the design parameters and the real-time control system, an on-line fault tolerant control test bed for validating the proposed on-line fault tolerant control framework in real hardware has been constructed. The hardware setup is shown in Figure 5.1. It consists of the following major components,

1. a BALDOR dc motor with maximum ½ hp,
2. a MAGTROL HD-505-8N dynamometer,
3. a MAGTROL 6200 dynamometer controller/readout,
4. one ADVANCED dc motor amplifier,
5. dSPACE software, DS1102 board and cable box with Texas Instruments TMS320C31 floating-point Digital Signal Processor (DSP), and
6. NT workstation with Intel Pentium II-450 dual processors.

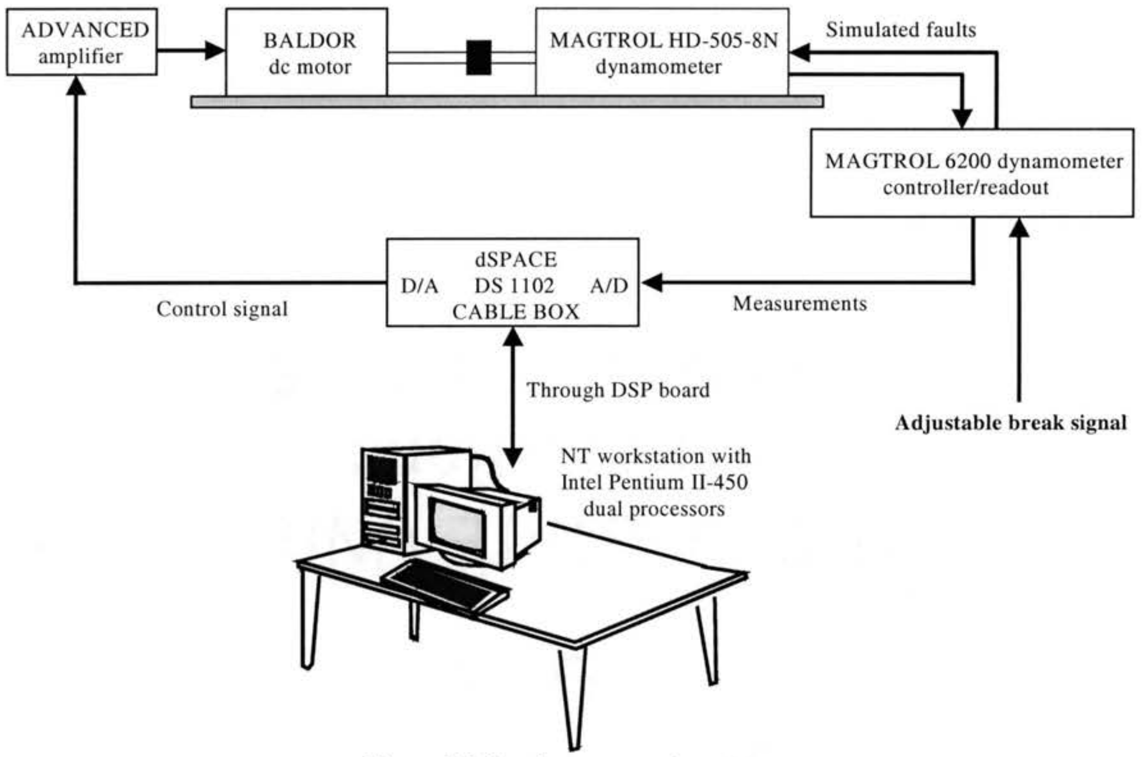

ADVANCED amplifier
BALDOR dc motor
MAGTROL HD-505-8N dynamometer
Simulated faults
MAGTROL 6200 dynamometer controller/readout
dSPACE D/A DS 1102 A/D CABLE BOX
Control signal
Measurements
Through DSP board
Adjustable break signal
NT workstation with Intel Pentium II-450 dual processors

Figure 5.1 Hardware experiment setup

The dc motor is connected to the dynamometer that is used to generate unanticipated friction on the motor shaft to simulate the unanticipated system failures. The control objective is to maintain the rotational speed of the motor (i.e., in terms of rpm) to the desired patterns with the presence of the unanticipated simulated failures. A computer with Intel Pentium II-450 dual processors is used to simulate the intelligent control regulator, fault detection mechanism, and on-line estimator. An embedded encoder and sensor in the dynamometer provide motor rpm and torque measurements in real-time, respectively. The measured signals are connected to a MAGTROL 6200 dynamometer controller/readout with the on-line readings shown on the device screen and the same signals are sent to dSPACE DS1102 cable box that is connected to the workstation through the TMS320C31 DSP board. An adjustable brake dial on the front panel of the dynamometer controller is used to generate the simulated time-varying, unknown and unanticipated brakes (i.e., unanticipated workload on the dc motor). All the necessary computation and the appropriate control input to drive the motor are computed within the workstation. The DSP board and dSPACE software are used to provide the necessary interface (A/D and D/A converter) and the integration of the real-time control with high-level languages such as MATLAB, SIMULINK, and C programs. A picture of the real-time fault tolerant control test bed is shown in Figure 5.2.

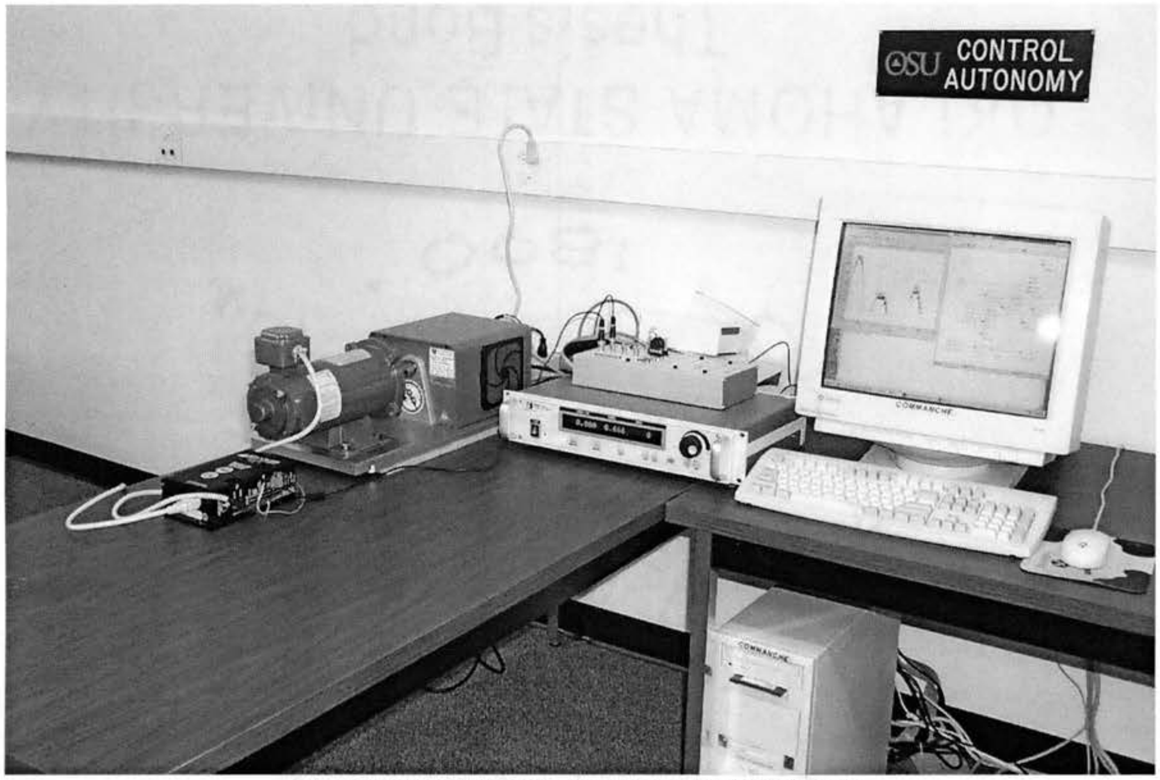


Figure 5.2 Real-time fault tolerant control test bed

In real-time environment, to close the on-line control loop as shown in Figure 5.1, an application source code (i.e., obj file) has to be created and downloaded to the TMS320C31 DSP. The on-line fault detection scheme, failure estimation, and control algorithm are performed under Matlab workspace in the NT workstation which communicates with the DSP through dSPACE MLIB (Matlab-dSPACE Interface Library). Figure 5.3 shows the SIMULINK model that is used to create the application source code for the real-time experiment. One 14-bit D/A converter channel and one 16-bit A/D converter channel are used to generate the control input (i.e., motor input voltage) and collect the torque reading from the dynamometer controller/readout, respectively. A discrete filter is used to reduce the effect of measurement noises in the torque reading. The rotational speed reading is decoded through one DS1102 encoder interface channel with a 24-bit counter. The DSP with generated application code runs the hardware experiments in real-time with sampling period 0.01 second and the control signal generated by the computer is sent to regulate the real-time response by changing the constant value in the SIMULINK model.

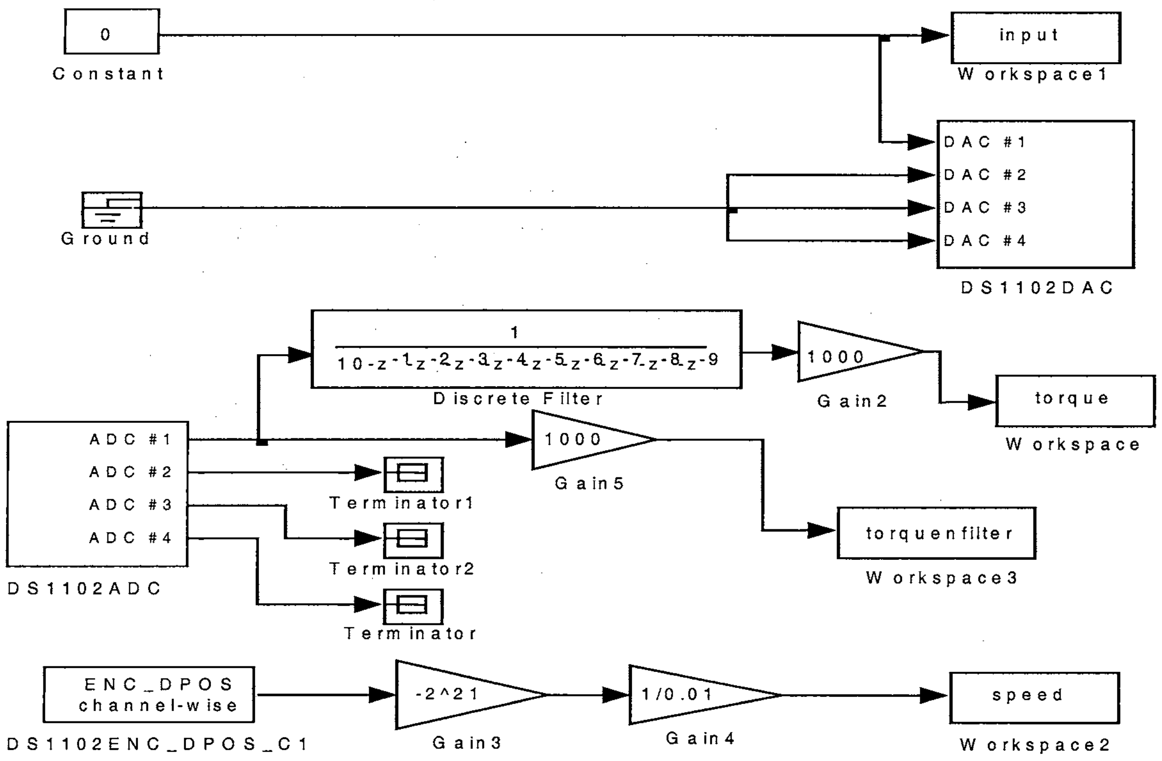


Figure 5.3 The SIMULINK model for the real-time experiment

Following the design procedure shown in Chapter 3, the first step is to obtain a nominal model for the fault-free system. It is well known that a dc motor can be modeled as a linear time-invariant system. For an armature-controlled dc motor with the negligible time constant of the armature, the nominal transfer function can be represented by Equation (5.1) [110],

$$G(s) = \frac{w(s)}{V(s)} = \frac{K_m}{[R_a(Js+f)+K_bK_m]}, \tag{5.1}$$

where $w(s)$ and $V(s)$ denote the rotational speed and motor voltage in s domain, respectively. K_b, K_m, R_a, J, and f are motor constants. Equation (5.1) can be re-organized as Equation (5.2) with A, b, c representing the corresponding constants,

$$A\dot{w} + bw = cV. \tag{5.2}$$

Using the forward Euler approximation shown in Equation (3.44), the discrete-time nominal model can be derived and shown in Equation (5.3),

$$\begin{aligned} w(k+1) &= [1 - b\Delta t / A]w(k) + [c\Delta t / A]V(k) \\ &= f_{linear}w(k) + g_{linear}V(k), \end{aligned} \tag{5.3}$$

which is in the similar form of the nominal model in Equation (3.26). The next step is to identify the parameters, $f_{linear} = [1 - b\Delta t / A]$ and $g_{linear} = [c\Delta t / A]$. Since Equation (5.3) is a linear time-invariant system, the batch form least square estimation method can be used for the identification of the parameters [109]. With the zero initial condition, 20,000 sets of input signals generated by Equation (5.4) are sent to the system for the collection of the system responses, $w(k)$,

$$V(k) = 0.015 \times \sin(\frac{k\pi}{2000}) + 0.015, \; k = 1,2,\ldots,19{,}999. \tag{5.4}$$

The batch form least square method provides the parameter estimation as follows [109],

$$Z = \begin{bmatrix} w(20{,}000) \\ w(19{,}999) \\ \vdots \\ w(2) \end{bmatrix},\ H = \begin{bmatrix} w(19{,}999) & V(19{,}999) \\ w(19{,}998) & V(19{,}998) \\ \vdots & \vdots \\ w(1) & V(1) \end{bmatrix},\ \theta = \begin{bmatrix} f_{linear} \\ g_{linear} \end{bmatrix},\ Z = H\theta + v \text{, and}$$

$$\hat{\theta}_{LS} = \begin{bmatrix} \hat{f}_{linear} \\ \hat{g}_{linear} \end{bmatrix} = [H^T H]^{-1} H^T Z\,, \tag{5.5}$$

where v and $\hat{\theta}_{LS}$ represent the white noise and the least square estimation, respectively. The design of the nominal controller follows Equation (3.30) without the term of fault estimator as shown in Equation (5.6),

$$\overline{Y}(k) = \frac{w_{desired}(k+1) - w_{desired}(k) - w(k)}{\Delta t} + aw_{desired}(k+1) \text{ and}$$

$$V_{no\min al}(k) = [\overline{Y}(k)(a + \frac{1}{\Delta t})^{-1} - \hat{f}_{linear} \times w(k)]\frac{1}{\hat{g}_{linear}} \tag{5.6}$$

with $a = 1$ and the S function defined in the form of Equation (3.1).

The final step in the off-line design stage is to evaluate the nominal model accuracy and the performance of the nominal controller under the fault-free situation for proper selection of the design parameters in the on-line fault detection scheme. The length of the time-shifting evaluation window for the fault detection scheme (i.e., Equation (3.57) with the square operation replaced by the absolute value) is pre-selected as 5 and the system response under nominal controller is tested using this criterion with the presence of measurement noises. The on-line fault detection threshold value is decided as 100 based upon the testing results. Under the unanticipated failures, the system response can be approximated by Equation (5.7),

$$w(k+1) = f_{linear} w(k) + g_{linear} V(k) + F(Torque(k))\,, \tag{5.7}$$

where F denotes the unknown effect that changes the motor rotational speed due to the unanticipated workload, $Torque(k)$. To reduce the negative effect of noisy measurements, the approximation target (i.e., numerical value of F) is computed based upon the average of the differences between the nominal model outputs and the actual speed readings every 10 time steps. A 1-5-5-1 MLP network is used to approximate the unknown failure effect, F, on-line with the static backpropagation algorithm. Four real-time experiments with different desired trajectories and unanticipated faults have been performed to test the proposed failure accommodation technique. Each real-time experiment is complete within 5,000 time steps. The design parameters of the learning result criterion (i.e., Equation (3.56)) for the alternative corrective control law are $l = 20$ and $\delta = 10$.

5.1 Experiment 1

The desired system response is generated by a linear model with the specified reference input as shown in Equation (5.8),

$$\begin{aligned} ref(k) &= 200 \times \sin(\frac{k\pi}{1000}) + 200, \\ w_{desired}(k+1) &= 0.6 w_{desired}(k) + 0.2 w_{desired}(k-1) + ref(k). \end{aligned} \tag{5.8}$$

15-17% unknown and unanticipated workload is generated by on-line adjusting the brake dial on the front panel of the dynamometer controller/readout. The real-time system response under the nominal controller is shown in Figure 5.4. Performance degradation is observed once the unknown workload occurs while the better control performances are shown in Figures 5.5 and 5.6 when the first control law and the alternative corrective control law are applied, respectively.

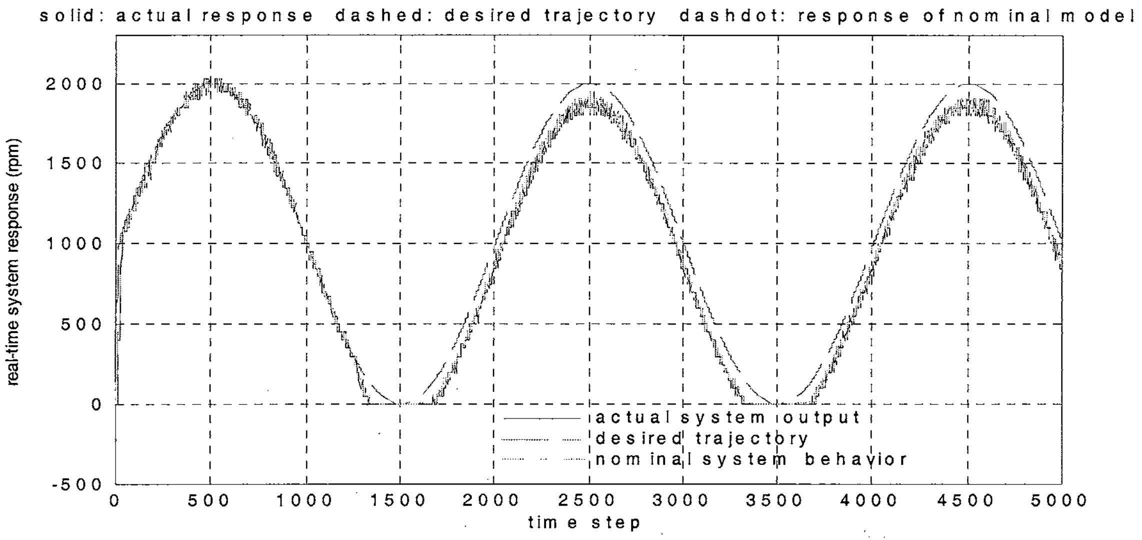


Figure 5.4 On-line system behavior under nominal controller only (experiment 1)

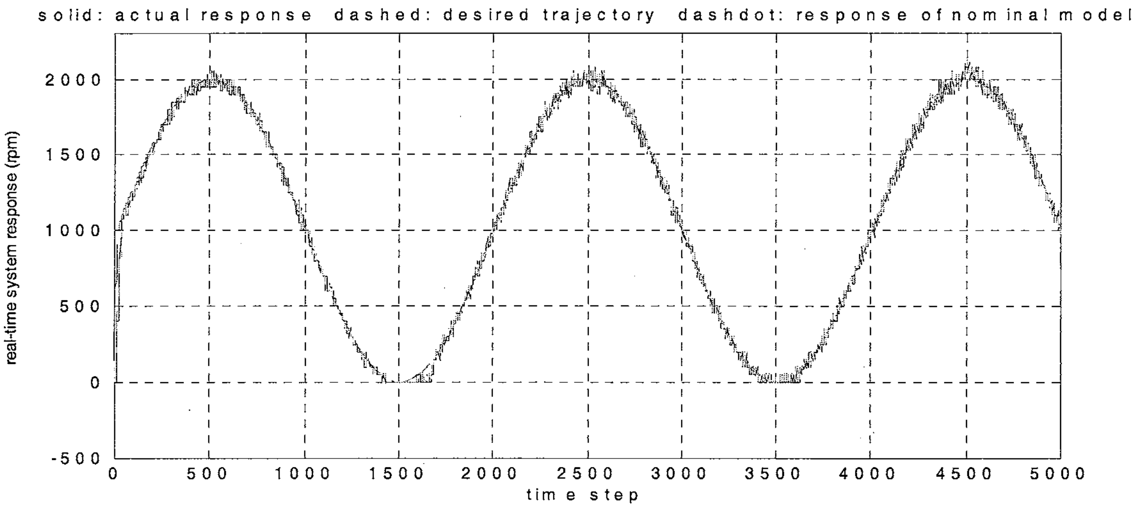


Figure 5.5 On-line system behavior under the first control law (experiment 1)

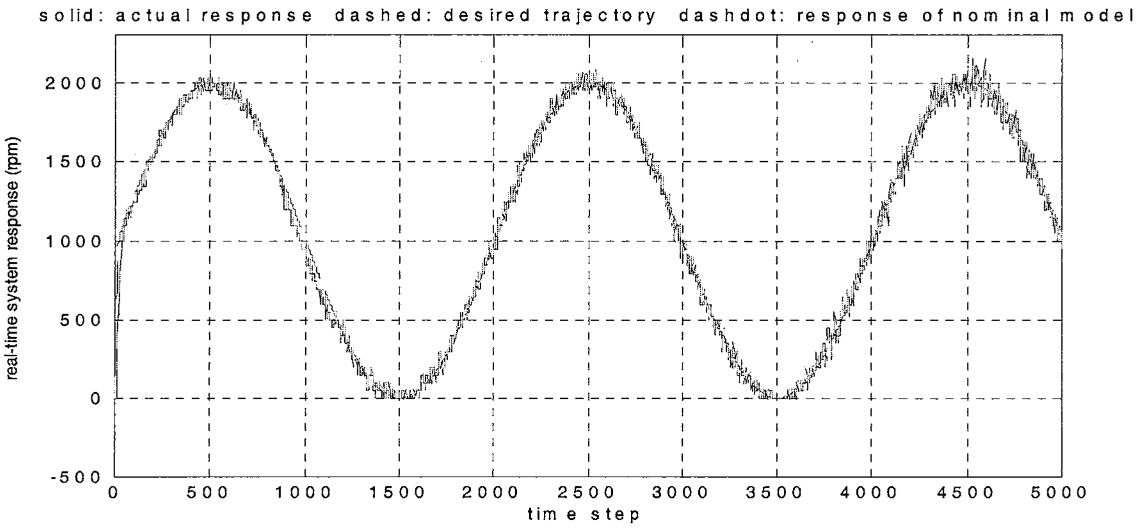


Figure 5.6 On-line system behavior under the alternative corrective control law (experiment 1)

5.2 Experiment 2

The control objective in this experiment is to maintain constant rotational speed at 1200 rpm with the presences of the measurement noise and unanticipated faults. Figure 5.7 is the real-time response plot under the nominal controller when 18-22% of unknown faults occur. Without proper compensation, the nominal control law results in drop of rotational speed up to 300 rpm during the experiment. The significantly improved control performances are obtained through the applications of the first control law and the alternative corrective control law shown in Figures 5.8 and 5.9, respectively.

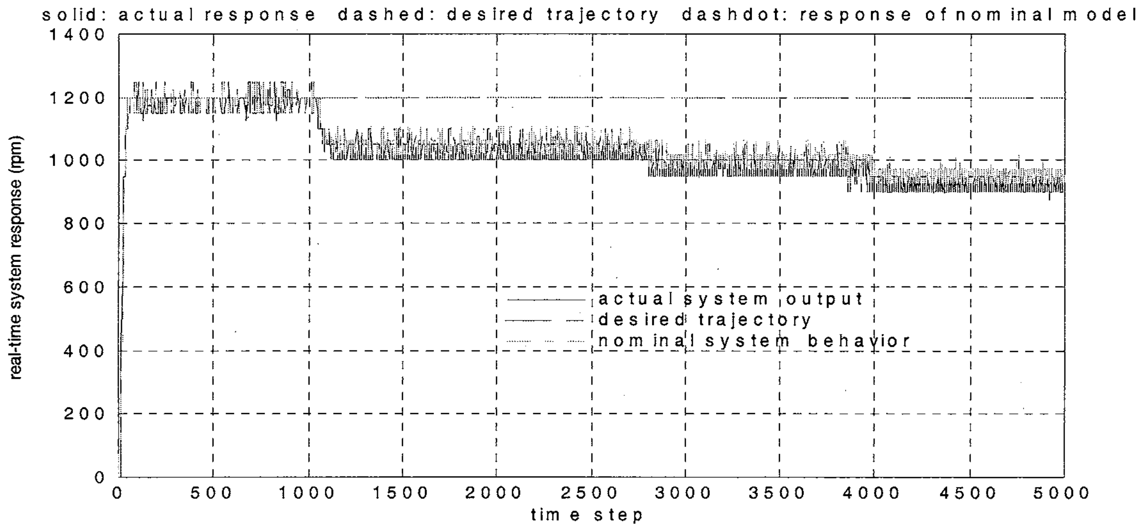


Figure 5.7 On-line system behavior under nominal controller only (experiment 2)

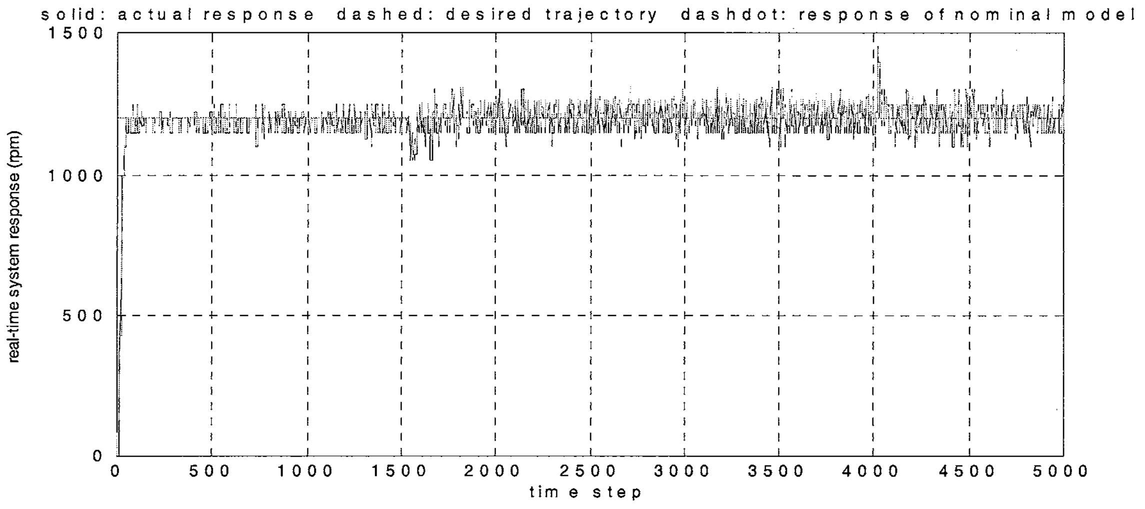


Figure 5.8 On-line system behavior under the first control law (experiment 2)

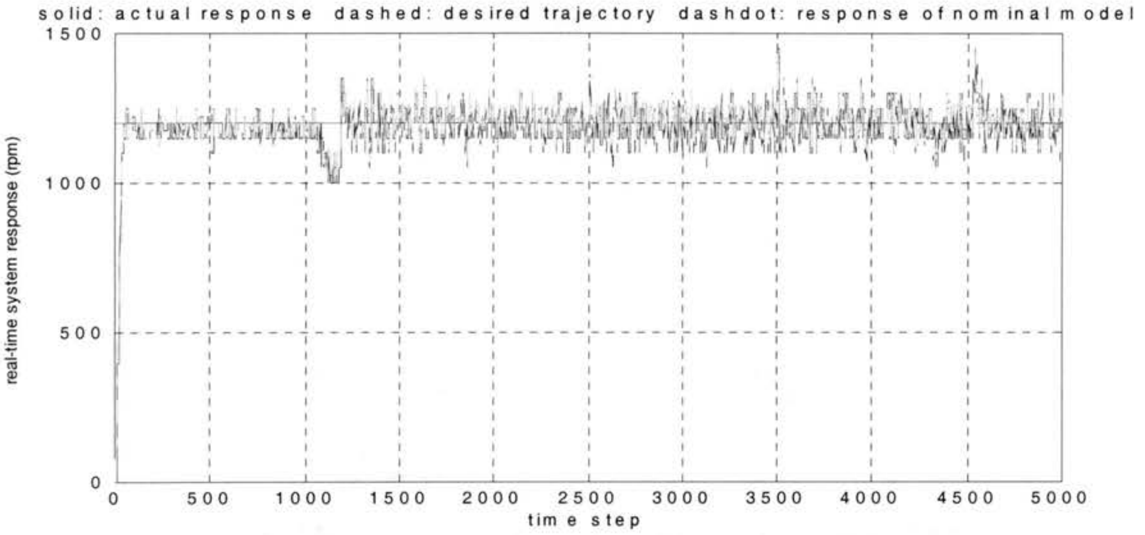


Figure 5.9 On-line system behavior under the alternative corrective control law (experiment 2)

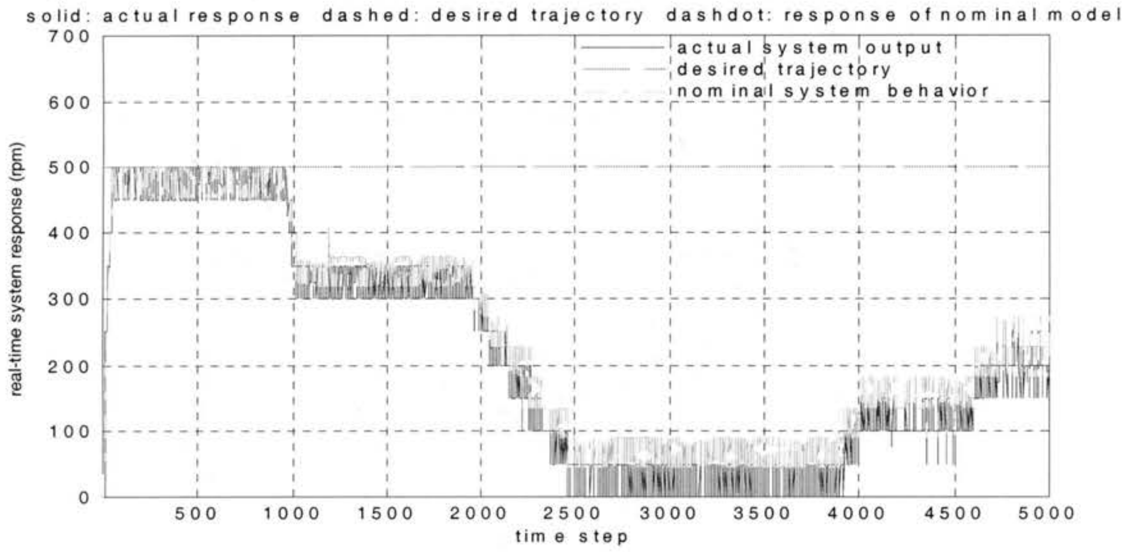


Figure 5.10 On-line system behavior under nominal controller only (experiment 3)

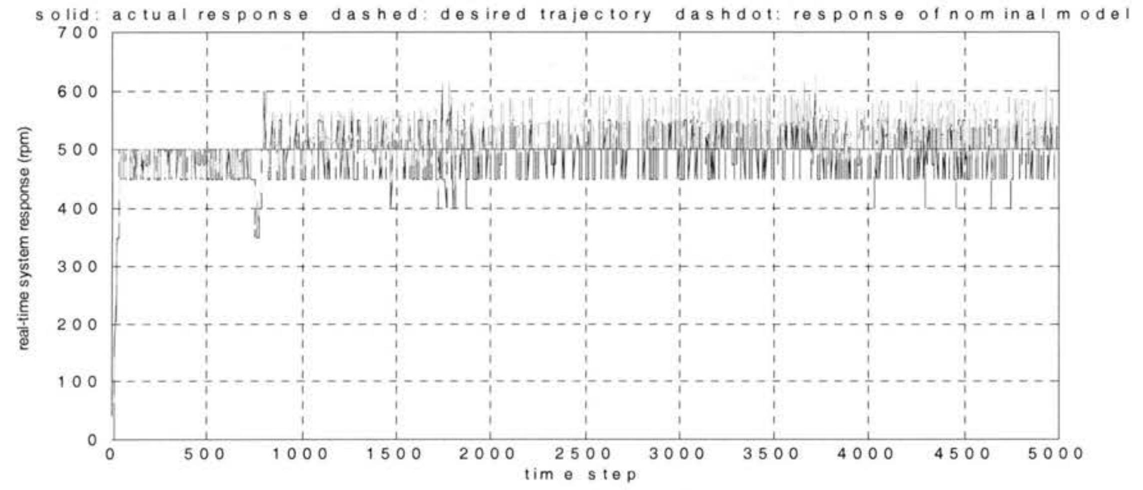


Figure 5.11 On-line system behavior under the first control law (experiment 3)

5.3 Experiment 3

The desired rotational speed is reduced to 500 rpm with increased unanticipated workload 18-30% in this experiment. Figure 5.10 is the real-time system behavior plot under the nominal controller. Due to the increased workload, the motor rotational speed almost reaches zero from time step 2500 to 3850. On the other hand, successful fault tolerant mission has been accomplished through the proposed on-line failure accommodation technique as shown in Figures 5.11 and 5.12.

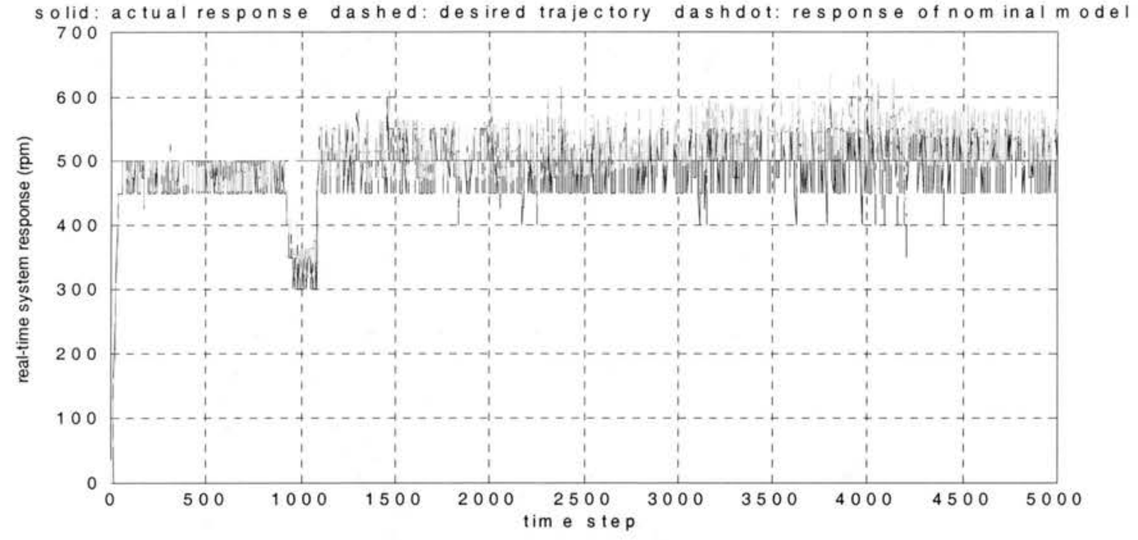


Figure 5.12 On-line system behavior under the alternative corrective control law (experiment 3)

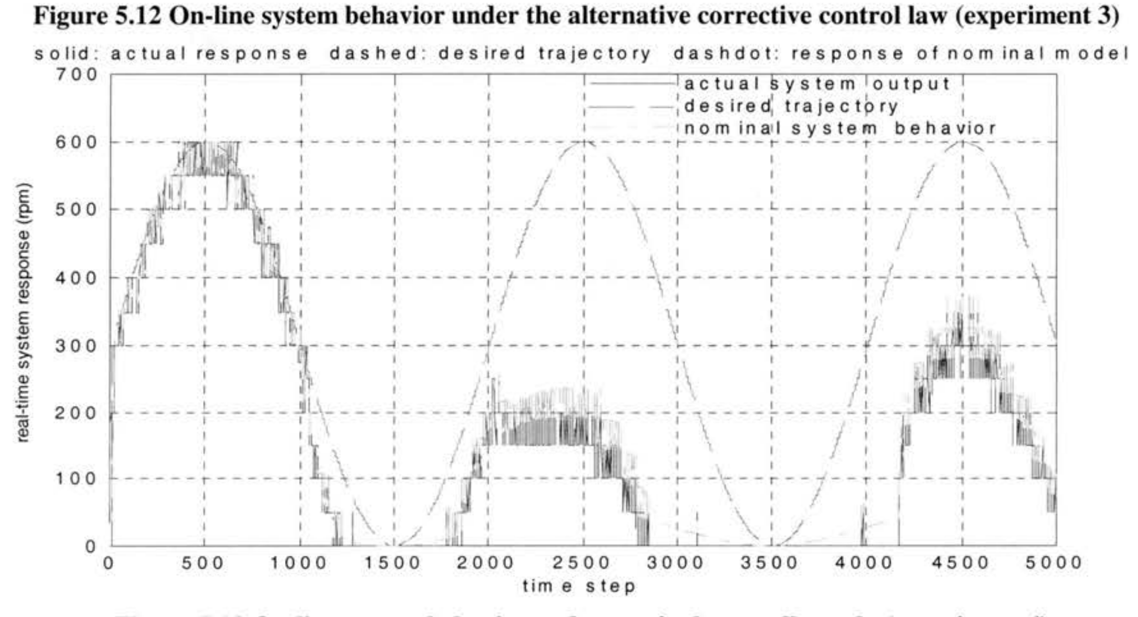


Figure 5.13 On-line system behavior under nominal controller only (experiment 4)

5.4 Experiment 4

Similar to Experiment 1, the desired trajectory in this experiment is selected as a sinusoid curve generated by the same linear model with a different reference input as shown in Equation (5.9),

$$ref(k) = 60 \times \sin(\frac{k\pi}{1000}) + 60. \tag{5.9}$$

The unknown workload used to generate the simulated unanticipated faults ranges from 15% to 28%. The system behavior under the failures with the nominal controller alone is plotted in Figure 5.13. As clearly shown, the performance has been significantly degraded and the rotation actually stops during the time periods, from time step 1300 to 1700 and 3200 to 4000, because of the relatively large unanticipated workload. Figures 5.14 and 5.15 show the satisfactory real-time fault accommodation when the first and the alternative corrective control techniques are applied, respectively.

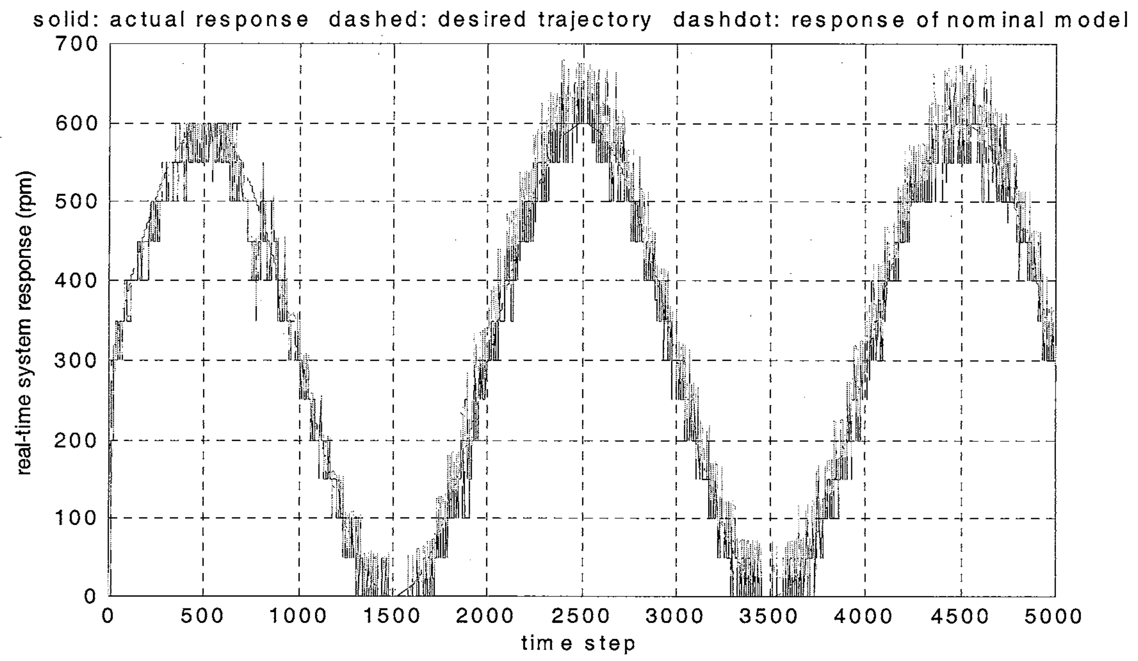


Figure 5.14 On-line system behavior under the first control law (experiment 4)

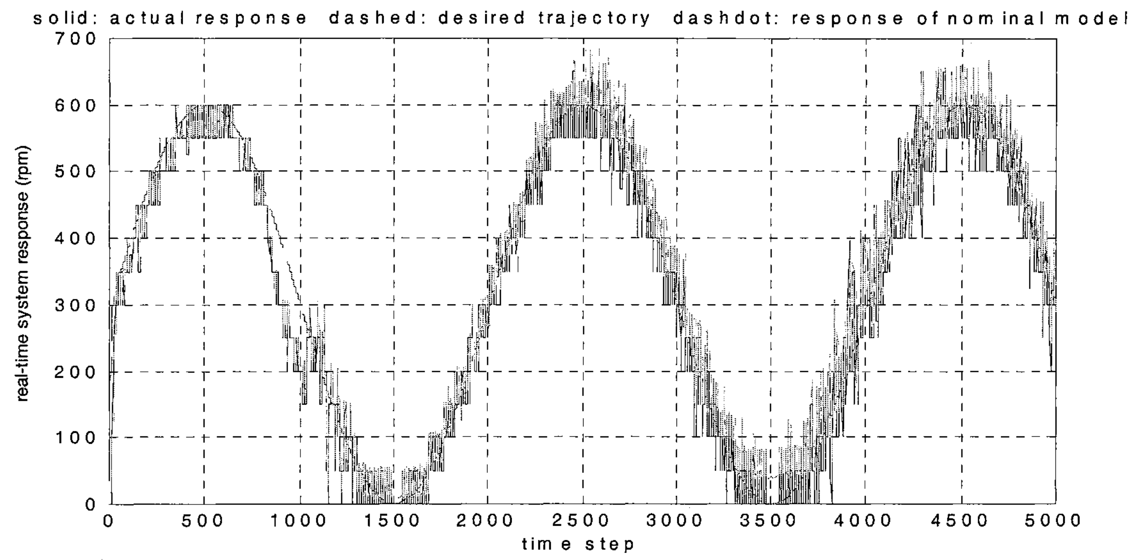


Figure 5.15 On-line system behavior under the alternative corrective control law (experiment 4)

5.5 Summaries and discussions

The effectiveness and efficiency of the proposed on-line failure accommodation technique for case 1 has been shown, and the possibility of successful on-line fault tolerance in real applications has been demonstrated through real-time hardware experiments under different desired control objectives with the presence of measurement noises and various unanticipated failures. Similar to the on-line simulation results shown in Chapter 4, the time constraint becomes a critical issue in the real-time applications. Successful on-line fault accommodation mission relies highly upon the computational capacity within the real-time control systems. The experimental results indicate that reducing the sampling rate in the real-time application source code will result in significant performance degradation due to the large difference between the real-time continuous system and the discrete-time approximation (i.e., system behavior exhibits large oscillation with sampling period higher than 0.01 sec. in the SIMULINK model shown in Figure 5.3). Slowing down the response time in the dual processors by increasing computational complexity will cause even more serious on-line control problems in real-time environment because of the delay of effective control actions. For more general situations (i.e., cases 2-4) where much more computational cost is required for successful on-line failure accommodation, the currently used computer system is obviously not fast enough to carry out the proposed fault accommodation technique in real-time. In other words, a more powerful computing device such as a higher speed processor is mandatory for the on-line real-time fault tolerant control in the real applications under the more general failure scenarios.

CHAPTER VI

A MULTIPLE-MODEL BASED FAULT DIAGNOSIS AND ACCOMMODATION ARCHITECTURE

The major research attention so far has been primarily focused on the on-line fault accommodation control technique for unanticipated catastrophic system failures. Although this technique can be directly applied for the on-line control purpose, under the developed methodology and the suggested on-line fault detection scheme, all the system abnormal behavior will be automatically considered as a consequence resulted from unknown system failures. Thus, the on-line estimator will be triggered in the learning process for the failure dynamics and a substantial amount of computational cost will be spent on both on-line estimation and computation of the effective control actions even when the failures are anticipated and the corresponding control actions are well known. The simulation tests also indicate that, under the on-line fault detection scheme, the false alarm situations will cause unnecessary computational waste since the nominal control actions are adequate to control the system behavior well in fault-free situations.

Apparently, a more sophisticated on-line fault tolerant control scheme should incorporate the proposed failure accommodation technique with a proper fault diagnosis mechanism, the post-failure models, and the corresponding post-failure control actions to avoid these situations. However, the detail of a systematic procedure for the on-line fault diagnosis scheme to avoid the false alarms with the guarantee of miss-free-detection, distinguish the anticipated faults from the failure situations, and select the effective control actions for the anticipated failures still remains to be addressed.

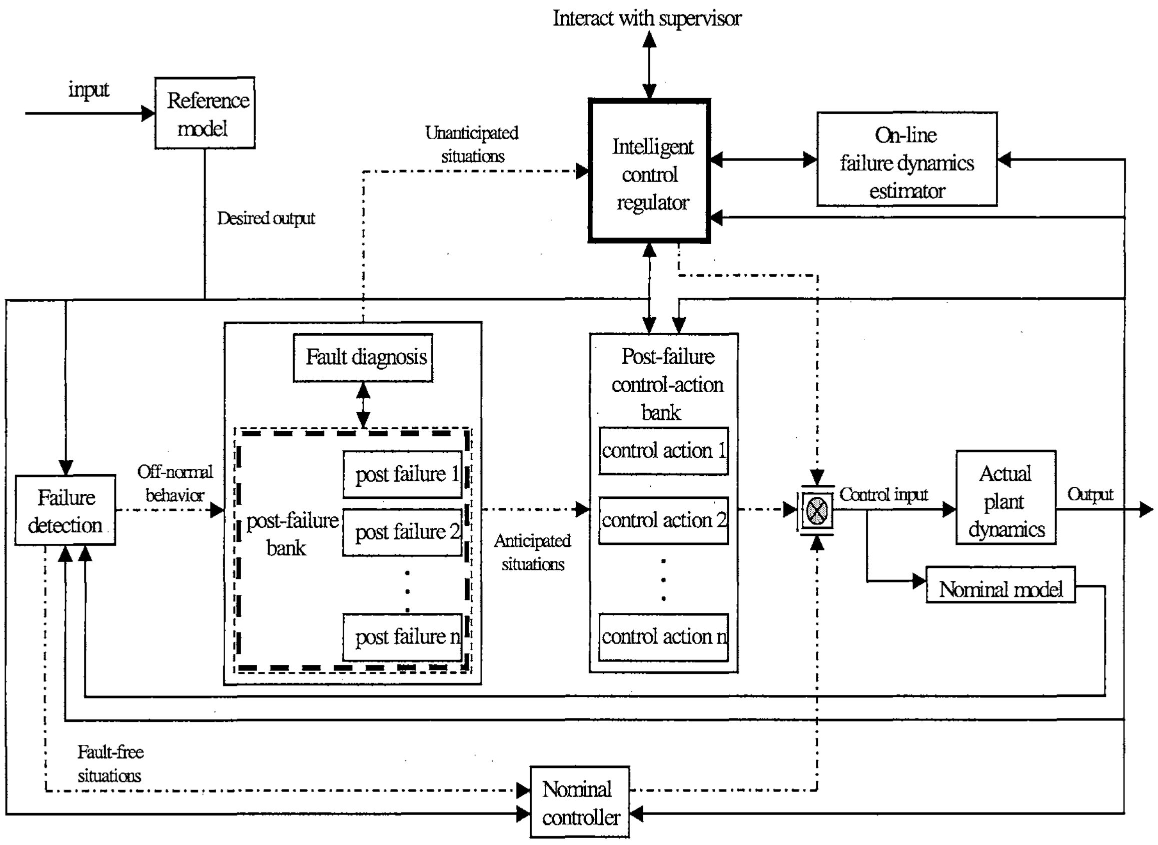


Figure 6.1 An architecture of multiple-model based fault diagnosis and accommodation

6.1 The multiple-model based FDA framework

Figure 6.1 shows a basic architecture of a multiple-model based fault diagnosis and accommodation framework. The developed on-line fault tolerant control technique incorporates a separate fault detection scheme, a failure diagnosis mechanism, and post-failure control actions to form a more sophisticated and complete FDA methodology. Unlike the framework shown in Figure 3.2, the intelligent control regulator is no longer sitting between the nominal controller and the actual system. Instead, it is now parallel with the post-failure control actions and the nominal controller to emphasize that, in on-line situations, the effective control actions that may come from one of the three sources, the nominal controller, the post-failure control actions, and the intelligent control regulator, are decided based upon system behavior or healthiness. The intelligent fault tolerant control technique will be applied only when it is necessary for the control purpose. Under this framework, the unnecessary computational waste for anticipated failures and false alarms could be avoided.

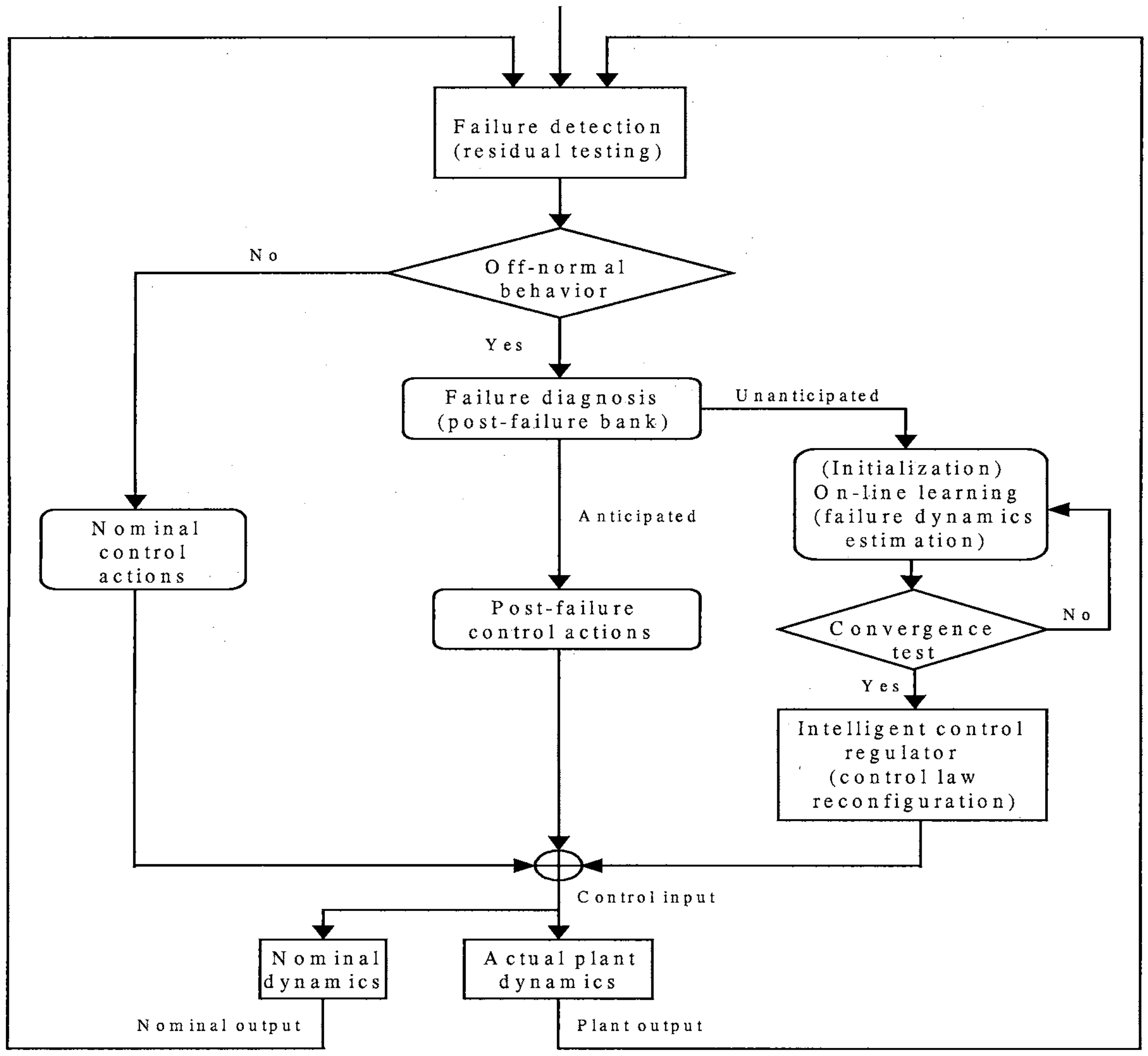

Failure detection (residual testing)
Off-normal behavior
No
Yes
Failure diagnosis (post-failure bank)
Unanticipated
(Initialization) On-line learning (failure dynamics estimation)
Nominal control actions
Anticipated
Post-failure control actions
Convergence test
No
Yes
Intelligent control regulator (control law reconfiguration)
Control input
Nominal dynamics
Actual plant dynamics
Nominal output
Plant output

Figure 6.2 The flow chart of the multiple-model based FDA

A clearer picture of how this idea works is depicted in a flow chart shown in Figure 6.2. The system "healthiness" is continuously monitored by the fault detection scheme with the knowledge of the nominal system behavior every certain period of time. Any off-normal behavior will trigger the failure diagnosis mechanism to analyze the situation and further decide which control actions should be taken. If the failure is recognized as an expected fault, the corresponding post-failure control actions will be selected as the current effective control commends. Otherwise, the developed intelligent FTC technique is initialized. Notice that the dashed line shown in Figure 6.1 indicates that only one action will be taken at every time instant.

6.1.1 On-line fault detection and diagnosis

With the presence of measurement noises, disturbances, and modeling errors, the problems of Fault Detection and Diagnosis (FDD), such as residual generation, sensitivity, robustness, false alarm, miss detection, and failure isolability, are substantially difficult issues to solve. In spite of many research efforts dedicated to address the FDD problems [1,3-6,8,23-26,64,84-90], complete on-line fault detection and diagnosis is still far from complete due to the inherent complexity of the problems and the time constraint in on-line operations. The ultimate goal of fault detection and diagnosis is leading to the failure accommodation. Until the invention of breakthrough technology for FDD, from a realistic point of view, identifying a better trade-off solution for real implementation based upon existing technology is the best an engineer can do. Since the system safety is the top priority of control missions, false alarms are more preferable than the miss detection. Moreover, it is true that a more sophisticated fault detection and diagnosis scheme will result in a better treatment for the failures. However, it also implies much more computational cost in real implementation.

From both system safety and on-line computational complexity point of views, a simple, however, computationally cost-effective criterion will be used as the on-line fault detection scheme in Figure 6.1 and the real-time FDA simulation. This criterion evaluates the mean square tracking error within a certain time window shown in Equation (6.1), which provides an effective miss detection-free scheme.

$$\psi_f = \frac{1}{\omega_f} \sum_{k=k_0}^{k_0+\omega_f-1} (ny(k) - y(k))^2. \tag{6.1}$$

$y(k)$ and $ny(k)$ denote the system output and the nominal model output at time step k, respectively. The design parameter, ω_f, represents the length of the evaluating window for fault detection. In other words, the system healthiness will be examined every ω_f time steps to decide whether or not the system is still under nominal condition by comparing the on-line system performance with the nominal behavior. This also implies that the control actions can be switched as fast as once every ω_f time steps. Unlike the fault detection technique reported in [23-26] where an approximator is deployed to approximate modeling error on-line by monitoring the system behavior and any significant deviation of the approximator output away from the origin is considered as a detection of failures, this fault detection scheme provides more computational efficiency since there is no need to spend computational cost in the on-line approximator only for the simple detection purpose of the abnormal behavior. However, sharing the similar spirit with [23-26], a pre-specified threshold value based upon the modeling uncertainty and the expected measurement noise is used to complete the fault detection shown in Equation (6.1).

Under this simple, conservative fault detection method, miss detection becomes trivial since the control objective is to keep the tracking error as small as possible within an affordable control effort. If the fault cannot be seen on the tracking error or it lasts only a short transient period such that the failure alarm is not triggered, the fault is not within our concern (i.e., its effect on the system performance does not degrade the control performance). Of course, the price of the trivial miss detection and computational simplicity is the increasing possibility of false alarms, which are possibly caused by unexpected interferences or noises. However, under the FDA architecture in Figure 6.1, the fault detection scheme will examine the system healthiness every ω_f time steps. In cases of failure alarms caused by unexpected disturbances or measurement noises, the detection scheme will eventually recognize the false alarm situations and recommend nominal control actions to avoid the unnecessary control effort after the effects resulting from unexpected disturbances or noises decays.

In many real systems, some failures could be well known or anticipated according to the history of system behavior and/or the aging degree of the system components. For those known and/or expected faults, the corresponding failure patterns or signatures and the corresponding control actions can be developed off-line and pre-stored in a database for on-line control purposes. The appropriate on-line failure accommodation actions are then suggested by a proper fault diagnosis scheme that identifies the failure patterns on-line. This approach shares a similar spirit with the well-known multiple-model approach [58]. Although there is no credible theoretical result to guarantee the stability of multiple model switching, this idea has attracted substantial attention and has been widely used in many areas [58-62,90-95,106-108]. Similarly, the conservative diagnostic attitude should be preferable since the price of the misdiagnosis and mistreatment could be instability and unaffordable loss.

An efficient fault isolation method used in the real-time FDA simulation is shown in Equation (6.2),

$$\psi_{fdiag_i} = \sum_{k=k_0-\omega_{fdiag}+1}^{k_0} \left(\left[y(k) - ny(k) \right] - pf_i(k) \right)^2, \tag{6.2}$$

where k_0 is the current time step and pf_i represents the time domain signatures of the post-failure model i. The principle of this method is based upon the multiple model approach where anticipated or possible failures are first analyzed to form a post-failure model bank. The corresponding post-failure control actions are also designed off-line to construct the post-failure control action bank according to the mathematical or numerical realization of the failure situations through the post-failure model bank. The effective control actions to achieve successful failure accommodations for anticipated faults are selected based upon the matching conditions of the signatures between the actual failures and the multiple-model based failures, which is the major fault isolation process. From the computational complexity point of view in the on-line situation, the time domain signature is considered as an appropriate cost-effective criterion for the failure diagnosis process. The fault isolation process is to compare the most recent time domain signatures between the actual failure and the post failures. The differences between the actual measurements and the outputs of the nominal model are considered as the outputs from the failure dynamics and they are compared with the “signatures”, outputs, from the post failure models within a certain length of time window, ω_{fdiag}. A pre-specified threshold value is used to compare with ψ_{fdiag_i} for the proper selection of the anticipated failure condition. If none of the "signatures" of the anticipated failures meets the criterion, the system status will be switched to the unanticipated failure situation and the intelligent on-line FTC approach will be initialized.

6.2 Simulation study for the multiple-model based FDA framework

To obtain a deeper insight into the on-line FDA problems in the real applications, a separate simulation study has been performed to test the FDA framework. The simulation is divided into two parts. They are dedicated to test the FDA framework for the system with different failure situations in a special case (case 1) and general cases, respectively. The design parameters including the threshold value of failure alarms, threshold value in the failure diagnosis process, lengths of the evaluating windows for fault detection and for failure diagnosis are pre-selected as $7.0\times e^{-5}$, $5.0\times e^{-6}$, 5, and 10, respectively.

6.2.1 Example 1 (case 1)

Consider the same nominal system as that in Subsection 4.1.1 with the nominal dynamics represented by Equation (4.1). Four different anticipated failures are assumed known and shown as follows:

$$\begin{aligned}
&\textit{post failure}\ 1: pf_1(k+1) = 1 - e^{-0.7|y(k)-y(k-1)|},\\
&\textit{post failure}\ 2: pf_2(k+1) = 0.46\times y(k)\times y(k-1),\\
&\textit{post failure}\ 3: pf_3(k+1) = 0.5\times \sin(y(k)\times y(k-1)),\\
&\textit{post failure}\ 4: pf_4(k+1) = 0.6\times \cos(y(k)\times y(k-1)).
\end{aligned} \tag{6.3}$$

The nominal control law is described in Equation (4.3) and, since the post failures are known, the corresponding control actions are easily computed as follows:

$$
\begin{aligned}
&post-failure\ control\ action\ 1:\\
&pfu_1(k)=\frac{m}{\Delta t}(-(y(k)-\frac{\Delta t}{m}(cy(k)^3+k_1y(k-1)))+y_d(k+1)-(1-e^{-0.7|y(k)-y(k-1)|})),\\
&post-failure\ control\ action\ 2:\\
&pfu_2(k)=\frac{m}{\Delta t}(-(y(k)-\frac{\Delta t}{m}(cy(k)^3+k_1y(k-1)))+y_d(k+1)-(0.46\times y(k)\times y(k-1))),\\
&post-failure\ control\ action\ 3:\\
&pfu_3(k)=\frac{m}{\Delta t}(-(y(k)-\frac{\Delta t}{m}(cy(k)^3+k_1y(k-1)))+y_d(k+1)-(0.5\times\sin(y(k)\times y(k-1)))),\\
&post-failure\ control\ action\ 4:\\
&pfu_4(k)=\frac{m}{\Delta t}(-(y(k)-\frac{\Delta t}{m}(cy(k)^3+k_1y(k-1)))+y_d(k+1)-(0.6\times\cos(y(k)\times y(k-1)))).
\end{aligned}
\tag{6.4}
$$

6.2.1.1 Scenario 1

Consider a failure situation involving the incipient anticipated failure 1 and the abrupt anticipated failure 2 with the time profiles shown in Equation (6.5),

$$
\begin{aligned}
&\beta_1(k-T_1)=(1-e^{-0.8\times(k-T_1)})U(k-T_1)\,;T_1=125,\\
&\beta_2(k-T_2)=U(k-T_2)\,;T_2=430.
\end{aligned}
\tag{6.5}
$$

System response under the intelligent FDA framework and the design parameters is shown in Figure 6.3. After time step 125, system output starts deviating from the desired trajectory due to the variation of system dynamics caused by the incipient failure. When the error is significant enough to trigger the fault detection scheme for failure alarm, the failure diagnosis process proceeds to identify the failure situation. Figure 6.4 shows the system status from the diagnosis process at every time step during the simulation. As clearly seen, right after the failure alarm and before the signature of the actual failure can match with the signature of any post failure, the system status is switched to the unanticipated situation in order to properly control the failure dynamics, which is a reasonable reaction since proper control actions may be necessary to secure the system behavior before the failure situation can be recognized.

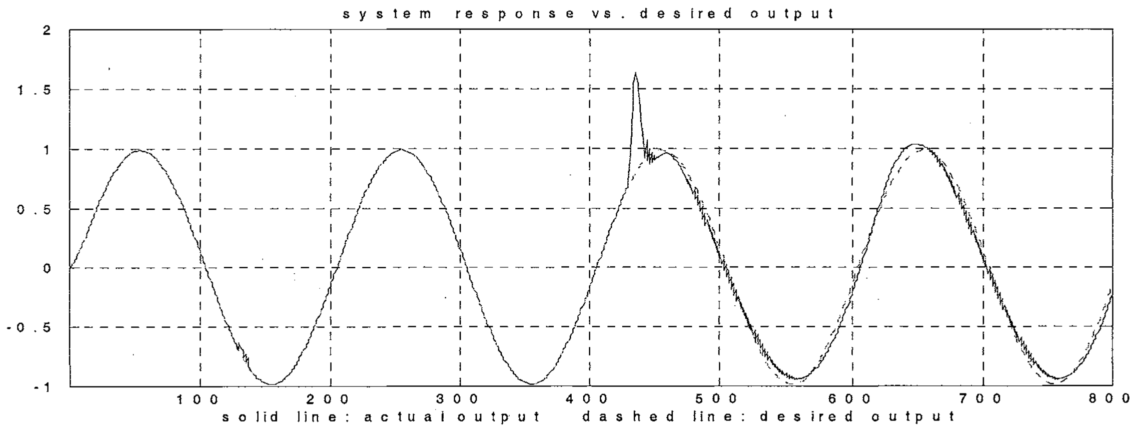


Figure 6.3 System response vs. desired output (scenario 1; case 1; FDA simulation)

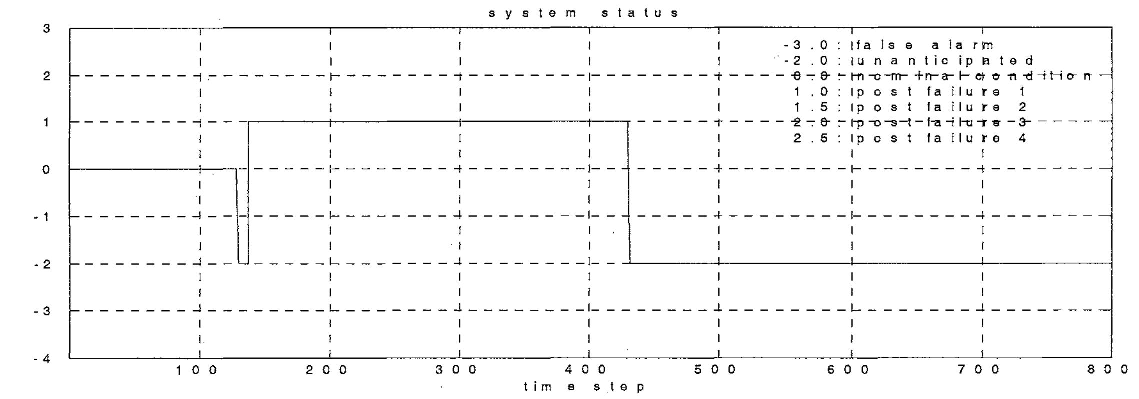


Figure 6.4 System status (scenario 1; case 1; FDA simulation)

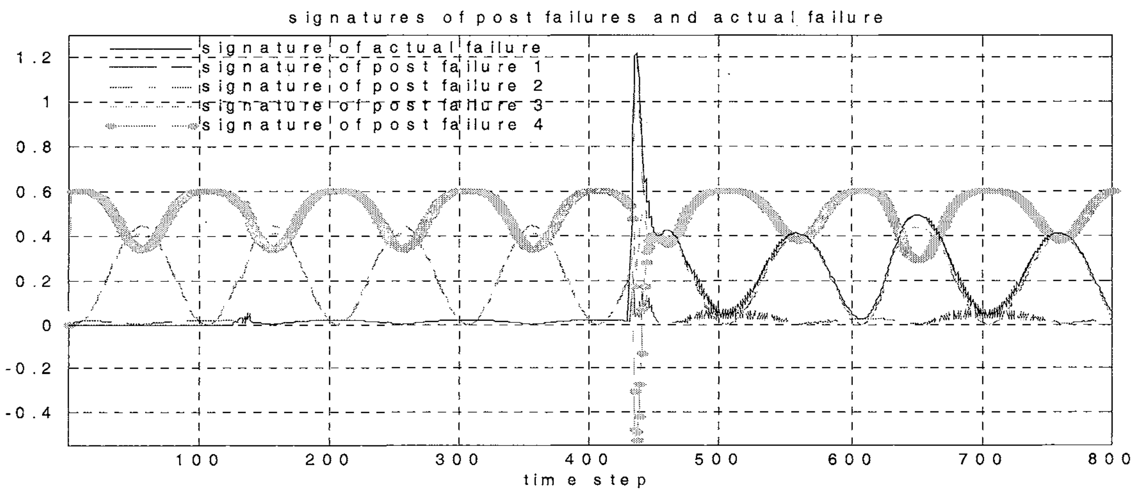


Figure 6.5 Signatures of post failures vs. actual failure dynamics (scenario 1; case 1; FDA simulation)

Approximately 10 to 15 time steps later, the fault diagnosis process recognizes that the failure situation matches with the anticipated failure 1 by comparing their corresponding signatures in the time domain. The pre-stored corresponding control actions are taken as the effective control commands until the time step reaches 430, at which time the second failure occurs. The combination of two failures produces an unexpected dynamics change that can not be recognized by the post-failure bank. Therefore, the system status is diagnosed and changes to the unanticipated condition. This can also be observed from the plot for the time domain signatures of the actual failure and all the post failures shown in Figure 6.5.

Without the failure diagnosis process, the post-failure banks, and the corresponding post-failure control actions, the system status under the intelligent framework in Chapter 3 will be considered as unanticipated situations after the detection of the system abnormal behavior. Now, as indicated in Figure 6.4, approximately 43% of the computational cost has been saved, while the computational complexity for fault diagnosis process is ignorable, compared with the cost under the unanticipated situations.

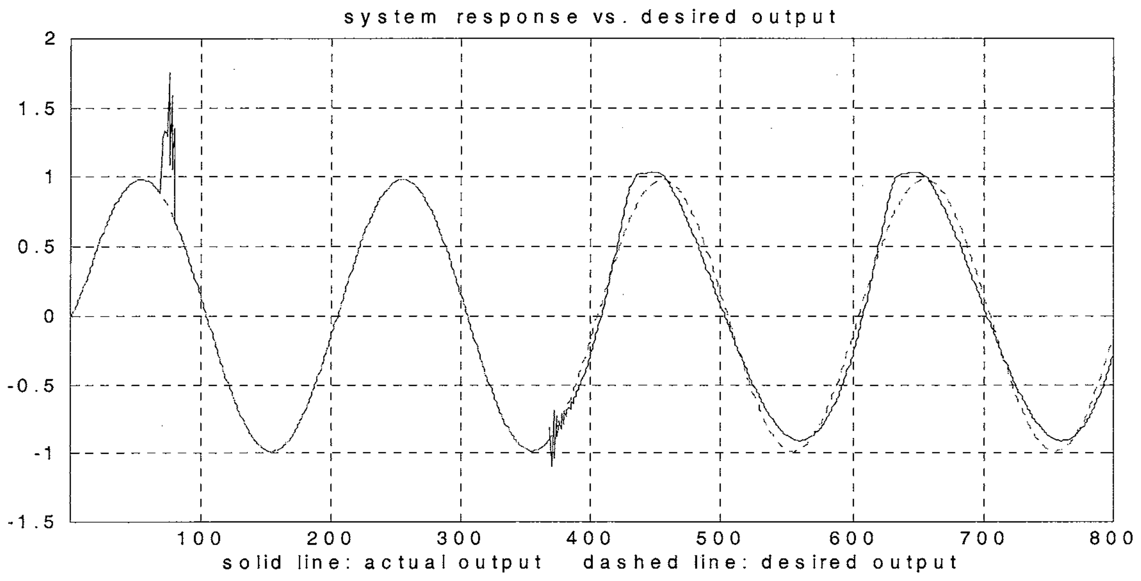


Figure 6.6 System response vs. desired output (scenario 2; case 1; FDA simulation)

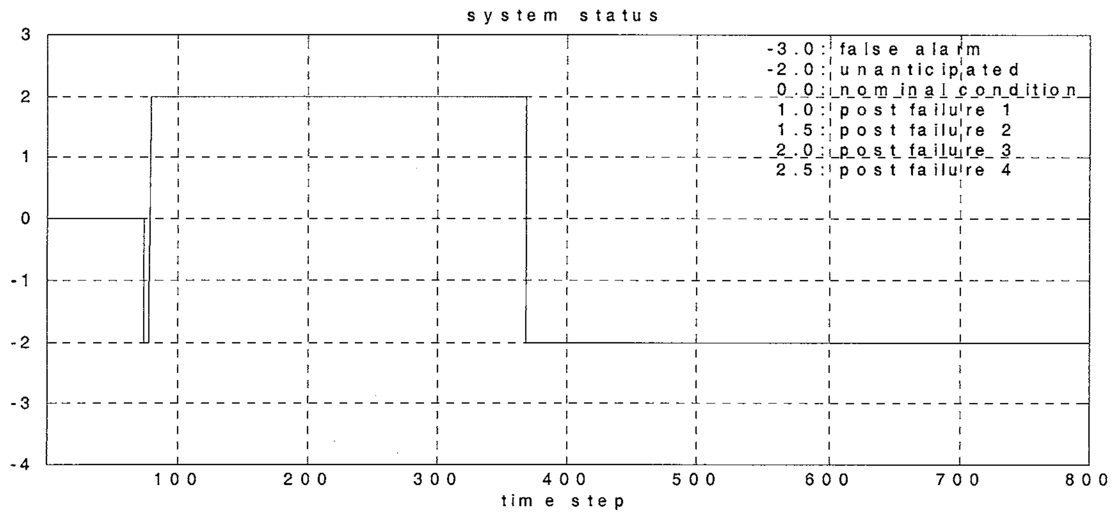


Figure 6.7 System status (scenario 2; case 1; FDA simulation)

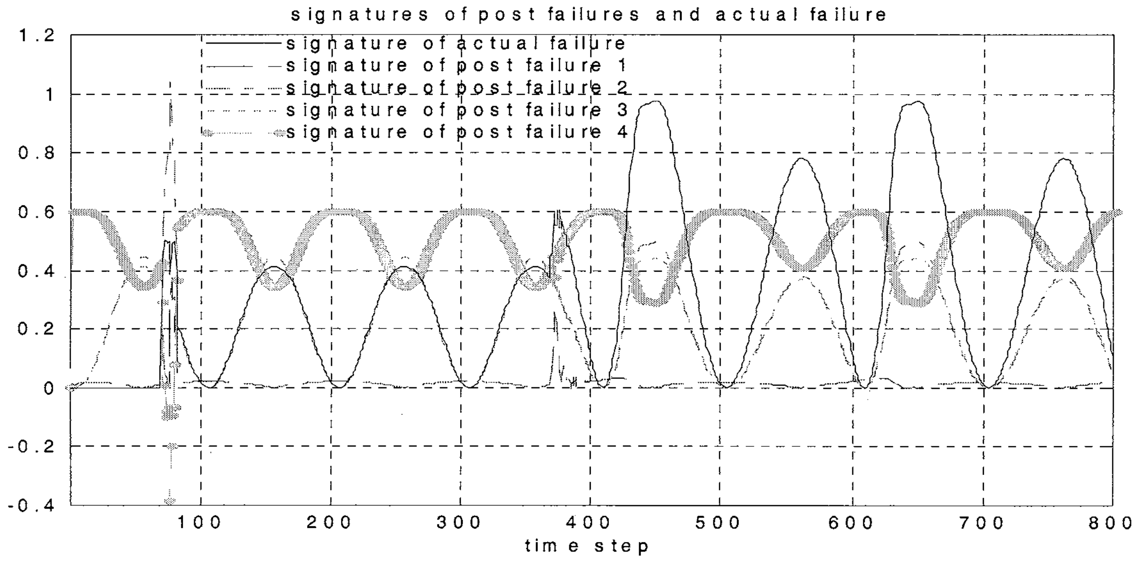


Figure 6.8 Signatures of post failures vs. actual failure dynamics (scenario 2; case 1; FDA simulation)

6.2.1.2 Scenario 2

Consider a failure situation involving the abrupt anticipated failure 3 which happened at time step 70 and an incipient unanticipated failure with the time profiles shown in Equation (6.6),

$$f_2(y(k), y(k-1)) = 0.5 \times y(k) \times y(k-1),$$
$$\beta_2(k-T_2) = (1-e^{-0.2\times(k-T_2)})U(k-T_2)\,;\, T_2 = 367. \qquad (6.6)$$

Figure 6.6 is the system response plot. System output quickly jumps away from the desired point due to the first (abrupt) failure. The failure has been recognized as the post failure 3 and properly accommodated within 10 time steps until the second failure appears. After that, the control strategy is switched to the unanticipated situation for the failure accommodation. Please also note that the first control law shown in Equation (3.30) is used in all the simulation tests of this case. Figures 6.7 and 6.8 are the plots for the system status and the signatures of the failures, respectively.

6.2.1.3 Scenario 3

Re-consider the failure situations in scenario 1 with the following different profiles,

$$\begin{aligned} \beta_1(k-T_1) &= U(k-T_1)\,;T_1=100, \\ \beta_2(k-T_2) &= U(k-T_2)\,;T_2=430. \end{aligned} \tag{6.7}$$

In order to test how the threshold value in the failure diagnosis process affects the FDA response, the design parameter is changed from $5.0\times e^{-6}$ to $1.0\times e^{-4}$. Figures 6.9-6.11 show the plots for the system on-line response, status from the failure diagnosis process, and the signatures of the failures. The significant difference shows in the on-line system status plot where it appears that the on-line failure diagnosis and control actions had been bouncing around between post failure 2 and the unanticipated situation, three times after time step 550. This is obviously the consequence of changing the threshold value in the diagnosis scheme since the actual failure signature is compared with the signatures of all the existing recognizable failures within a fixed length of time window and also since how much the signatures are considered a match depends totally upon the threshold value. It appears that the smaller threshold value gives more restriction in failure recognition and provides a more conservative diagnosis result. It also implies that more computational cost may possibly be spent due to the conservative attitude. On the other hand, if the value appears to be too big, large uncertainty will exist in the diagnosis result. Thus, the control actions may also jump around. Moreover, if the changing rate is too high, it is possible to excite unmodeled system dynamics under serious vibration.

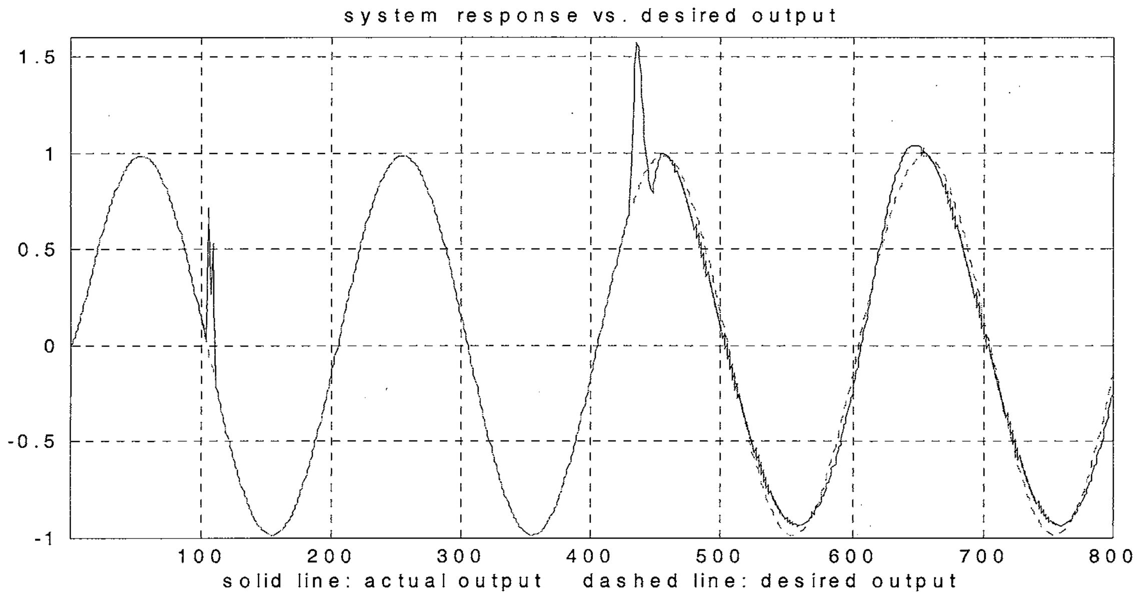


Figure 6.9 System response vs. desired output (scenario 3; case 1; FDA simulation)

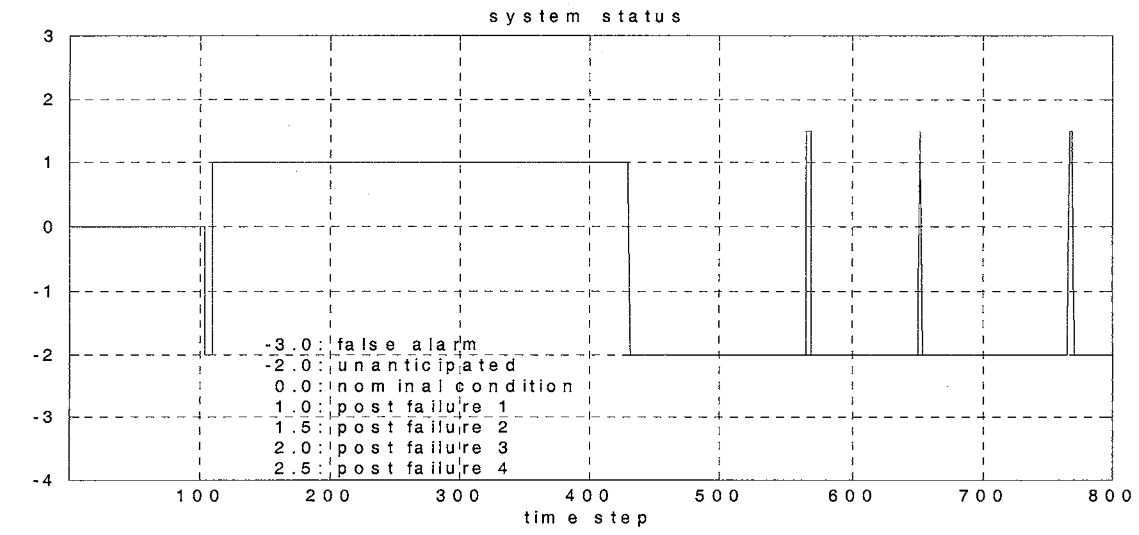


Figure 6.10 System status (scenario 3; case 1; FDA simulation)

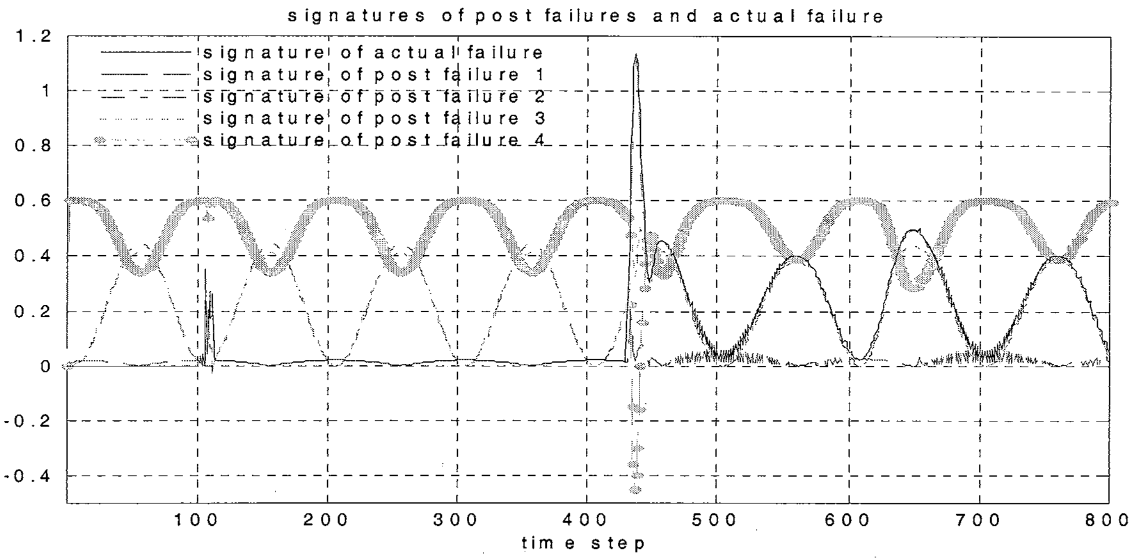


Figure 6.11 Signatures of post failures vs. actual failure dynamics (scenario 3; case 1; FDA simulation)

6.2.2 Example 2 (general cases)

Consider the same nominal system described in Equation (4.13) where the system dynamics is not in linear-in-control format and the nominal model is first realized by a 3-30-1 MLP network. The nominal NN controller is designed using the same technique described in Section 4.3. Three post failures are shown in Equation (6.8),

$$\begin{aligned}
&post\ \ failure\,1: pf_1(k+1) = -(0.3y(k)+0.6y(k-1)+u(k)^2-5u(k))+0.06y(k)^2+4u(k),\\
&post\ \ failure\,2: pf_2(k+1) = 0.05\times y(k)\times\cos(u(k)),\\
&post\ \ failure\,3: pf_3(k+1) = -0.8\times u(k)^2+3.5u(k).
\end{aligned} \tag{6.8}$$

To simulate the failures in a real dynamic system, the specific mathematical formats of these failure dynamics are assumed unknown and have to be realized by separate network models. The idea is described in the following steps.

1. Place the system under failure situations (i.e., using the nominal mathematical model with the failure dynamics in simulation stage).

2. Feed 3,000 uniformly distributed random input signals varying from -1.5 to 1.5 with selected initial conditions.

3. Collect the training patterns. The desired outputs are computed by using the differences between the system outputs and the outputs from the NN nominal model.

4. Train the NN model for the failure dynamics.

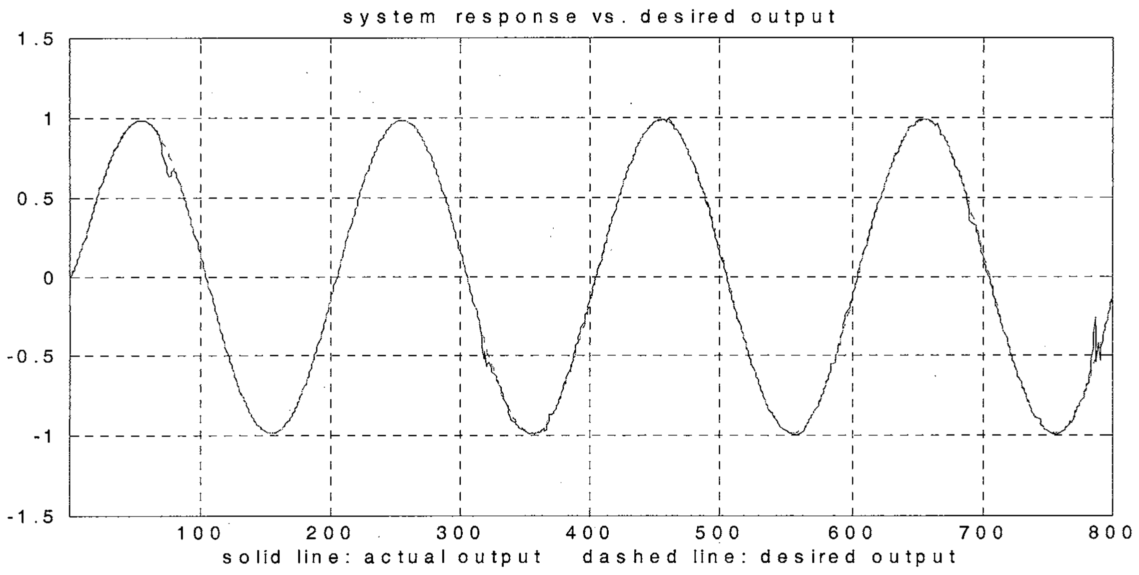


Figure 6.12 System response vs. desired output (scenario 1; general case; FDA simulation)

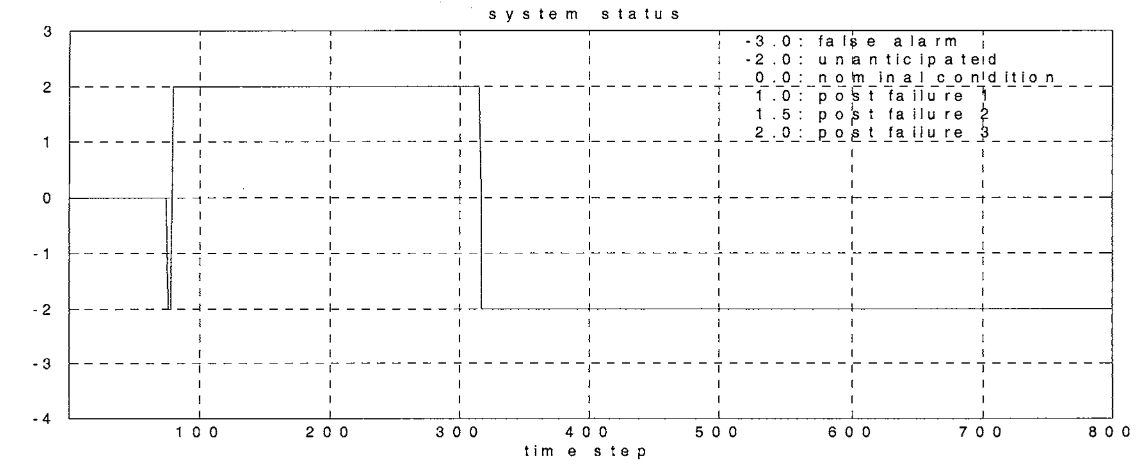


Figure 6.13 System status (scenario 1; general case; FDA simulation)

3 separate MLP networks with different structures, 3-30-1, 2-20-1, 1-20-1, are used to realize the 3 post failures off-line, respectively, by following the above steps. The post-

failure control actions are also realized off-line by using separate NN controllers (i.e., 3-20-1, 3-20-1, and 3-10-1 MLPs) and they are obtained using the same technique as described in Section 4.3. The pre-selected threshold value for the diagnosis process is $1.0 \times e^{-4}$ and the rest of the design parameters are the same as those in the last section.

6.2.2.1 Scenario 1

Consider the failure situation involving an abrupt anticipated failure 3 starting at time step 70 and an incipient unanticipated time-varying failure with the corresponding time profile shown in Equation (6.9),

$$\begin{aligned} f_2(\cdot) &= \frac{ky(k)}{320 \times (1 + y(k-1)^2)}; \\ \beta_2(k - T_2) &= (1 - e^{-0.2\times(k-T_2)})U(k - T_2)\,; T_2 = 314. \end{aligned} \tag{6.9}$$

Figures 6.12-6.14 are the on-line simulation results under the intelligent FDA framework. The abrupt anticipated failure 3, which happened at time step 70, is quickly recognized by the diagnosis process and the effective control commands are then switched to the corresponding post-failure control actions. This is clearly seen in the on-line system status plot shown in Figure 6.13 and the system on-line response shown in Figure 6.12. Observing Figure 6.14 closely, we can also see that the time-domain signatures of the actual failure dynamics and post failure 3 are almost indistinguishable from time step 80 to 320. After the second change of the system dynamics caused by another time-varying incipient failure, the discrepancy between the signatures are getting large, which indicates that an unanticipated situation has occurred.

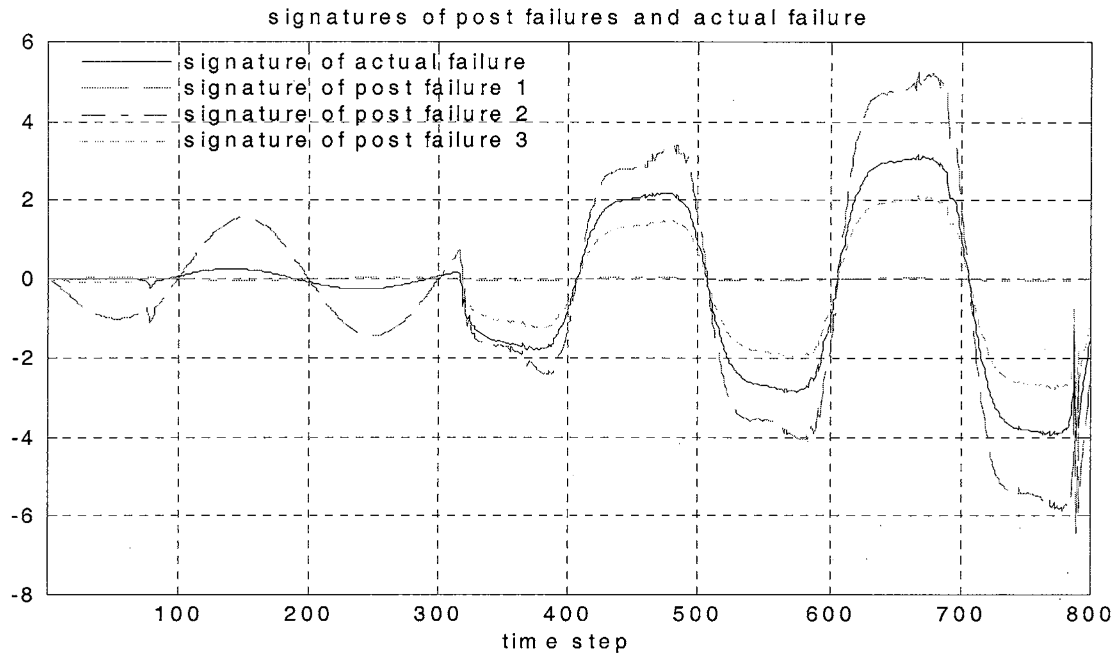


Figure 6.14 Signatures of post failures vs. actual failure dynamics (scenario 1; general case; FDA simulation)

6.2.2.2 Scenario 2

Consider the failure situation involving an incipient anticipated failure 2 with the time profile shown in Equation (6.10) and an abrupt anticipated failure 3 starting at time step 601,

$$\beta_1(k-T_1) = (1-e^{-0.009\times(k-T_1)})U(k-T_1)\,;\, T_1 = 25. \tag{6.10}$$

Figures 6.15-6.17 are the plots of the test results. Due to the slow variation of the failure dynamics (with $\alpha_1 = -0.009$), the incipient fault will not be identified as quickly as an abrupt failure since this incipient time profile will take almost 300 time steps to converge (i.e., 300 time steps are required for the profile to reach 0.9328). Figure 6.16 shows this expected result. The failure diagnosis scheme can not be sure of the actual failure situation until the time step almost reaches 300. This is also a correct decision since both the failure dynamics and the corresponding control actions are realized by NN under the abrupt failure situation. However, we do observe some diagnostic oscillation before time step 290. It is apparent that the diagnostic threshold value, $1.0 \times e^{-4}$, may be too big for the accuracy of the NN post-failure model 2. Observing Figure 6.17 closely, we can easily tell that there are signature differences between the actual failure and post failure 2 from time step 100 to 220. In other words, the NN post-failure model 2 in this case may be accurate enough to use a smaller threshold value in order to have a better diagnostic report. This result suggests that, for better failure diagnosis, the selection of the diagnostic threshold value should be failure-dependent and also should be chosen based upon the accuracy of the NN post-failure model.

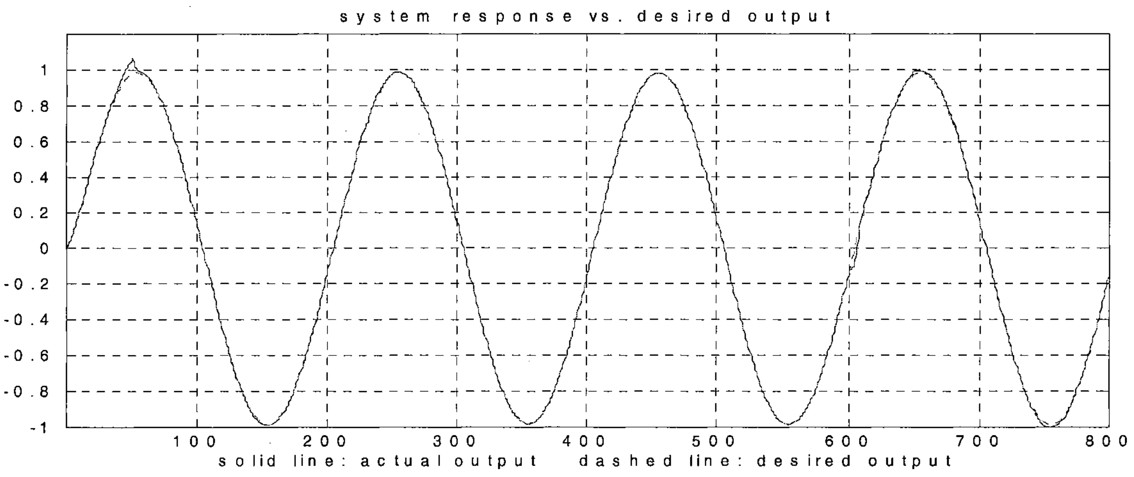


Figure 6.15 System response vs. desired output (scenario 2; general case; FDA simulation)

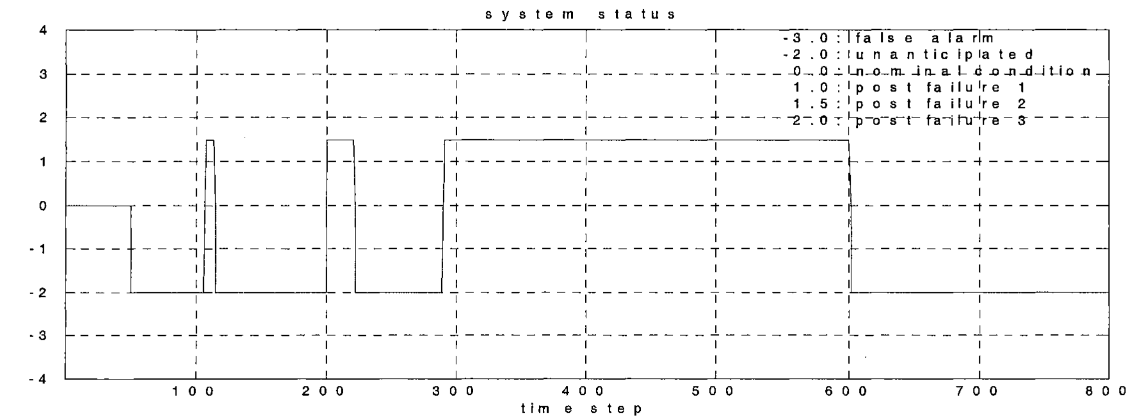


Figure 6.16 System status (scenario 2; general case; FDA simulation)

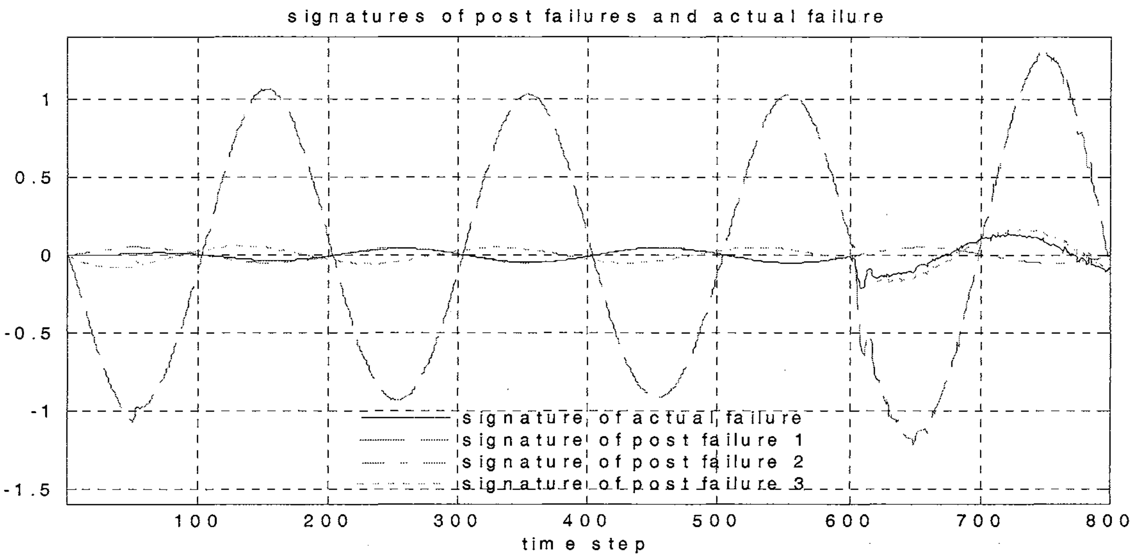


Figure 6.17 Signatures of post failures vs. actual failure dynamics (scenario 2; general case; FDA simulation)

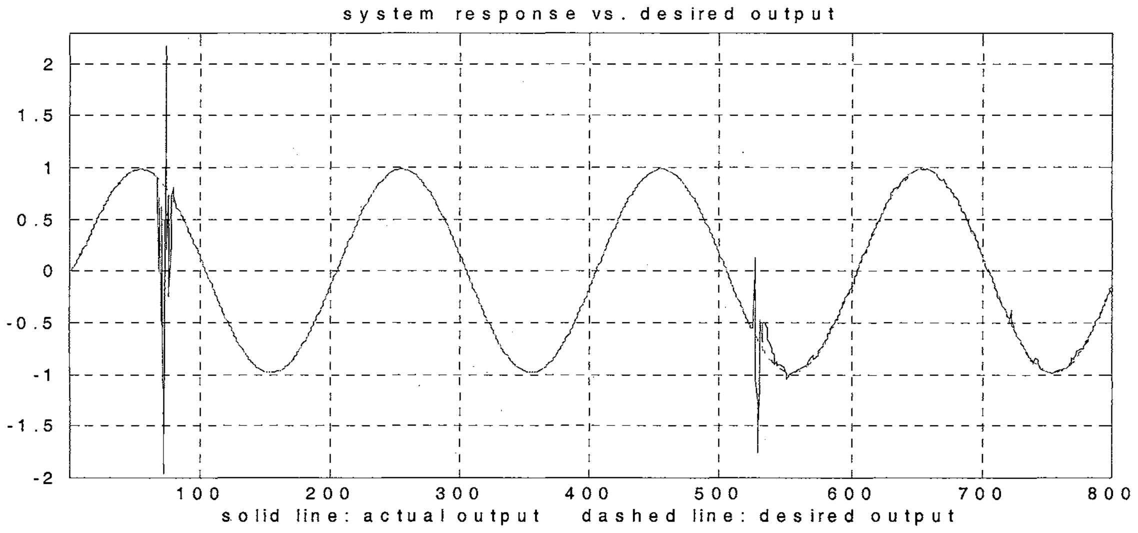


Figure 6.18 System response vs. desired output (scenario 3; general case; FDA simulation)

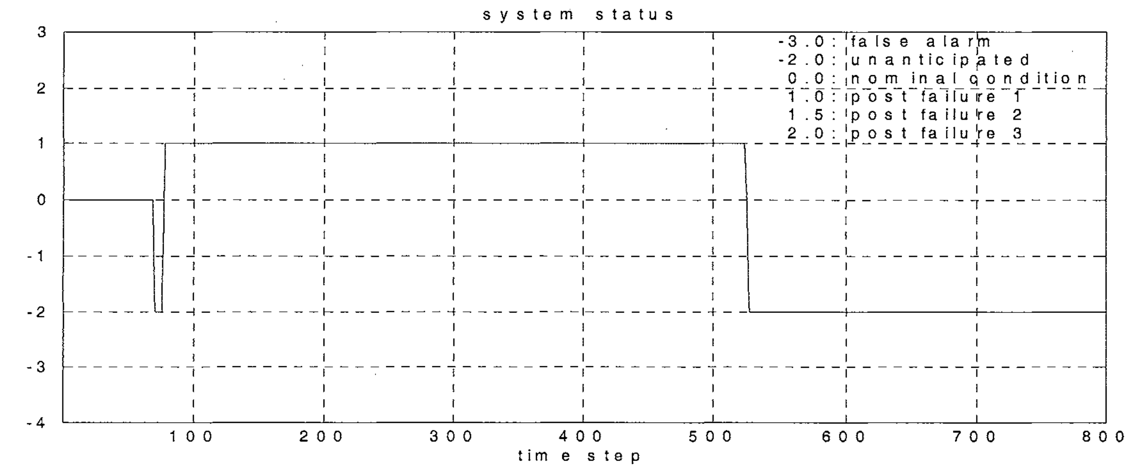


Figure 6.19 System status (scenario 3; general case; FDA simulation)

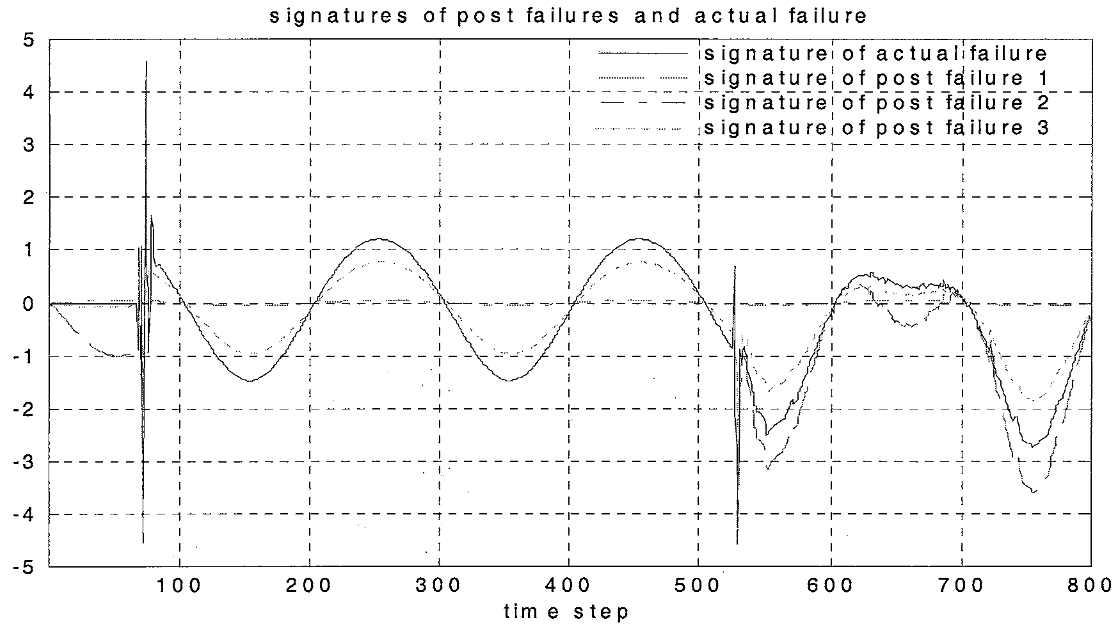


Figure 6.20 Signatures of post failures vs. actual failure dynamics (scenario 3; general case; FDA simulation)

6.2.2.3 Scenario 3

Consider the failure situation involving an abrupt anticipated failure 1 starting at time step 68 and an incipient time-varying unanticipated failure with the time profile shown in Equation (6.11),

$$f_2(\cdot) = 0.6\times\frac{k}{510}\times y(k)\times y(k-1)\,;$$
$$\beta_2(k-T_2) = (1-e^{-0.2\times(k-T_2)})U(k-T_2)\,;\,T_2 = 523. \qquad (6.11)$$

Figure 6.18 indicates that a large control error occurs suddenly right after appearance of the abrupt failure. It is identified as post failure 1 within 10 time steps as shown in Figure 6.19 and is properly accommodated by the corresponding post-failure control actions until a time-varying failure starts at time step 523. Figure 6.20 shows the observable signature discrepancies between the actual failure and the post failures after the time step 530 which corresponds to an unanticipated failure situation.

6.3 False alarm situations

The nominal system and the nominal controller in the general case are selected to test the false alarm situation under the intelligent FDA architecture. In order to simulate the false failure alarm possibly caused by unexpected measurement noises, uniformly distributed random white noises are generated and added to the measurements. The unknown and unexpected white noises are generated during three time periods, time step 78 to 81, 212 to 215, and 465 to 466. The added noises are varying from –0.3 to 0.3, -0.23 to 0.23, and –0.85 to 0.85, respectively. The simulation results are plotted and shown in Figures 6.21-6.23.

Figure 6.22 indicates how the FDA framework reacts to the false alarm situation caused by the unanticipated noises. Once the contaminated measurements trigger the alarm, the system behavior is immediately examined by failure diagnosis and, since the measurements are contaminated by noises, it will not be recognized as any one of the anticipated failures. Thus, the diagnostic result will suggest an unanticipated situation. However, after the effect resulting from unexpected noises decays, the fault detection scheme discovers the system behavior is as normal as it is under the nominal situation. After a double check of this situation, the fault detection scheme flags a false alarm signal and the system status is switched back to the nominal condition. Figure 6.21 shows some slight deviations of the system output away from the desired trajectory during the periods of noisy data, which is apparently caused by the fact that the contaminated measurements are used directly in the fault detection scheme and in the unanticipated failure conditions as the learning targets for failure accommodation.

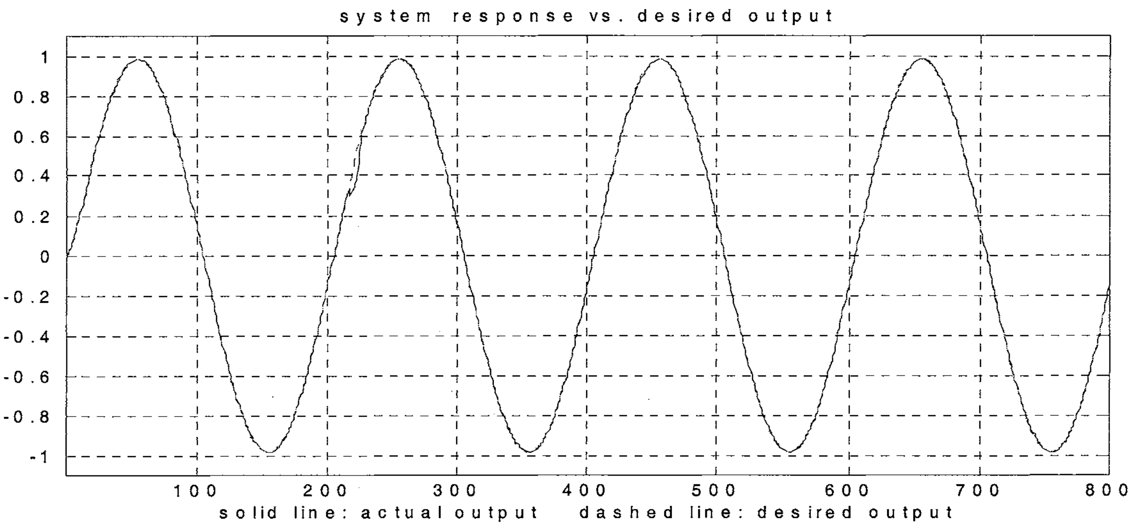


Figure 6.21 System response vs. desired output (false alarm situations; FDA simulation)

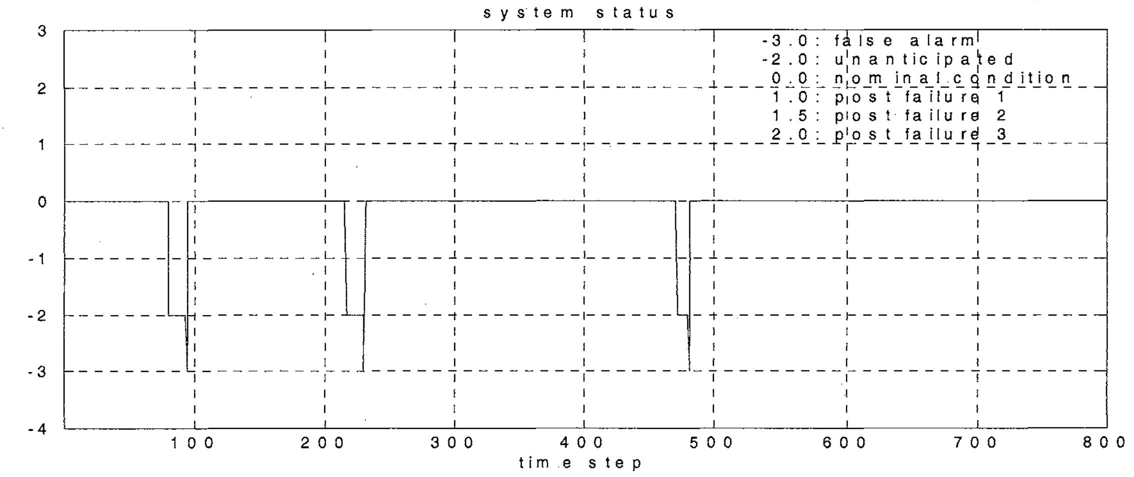


Figure 6.22 System status (false alarm situations; FDA simulation)

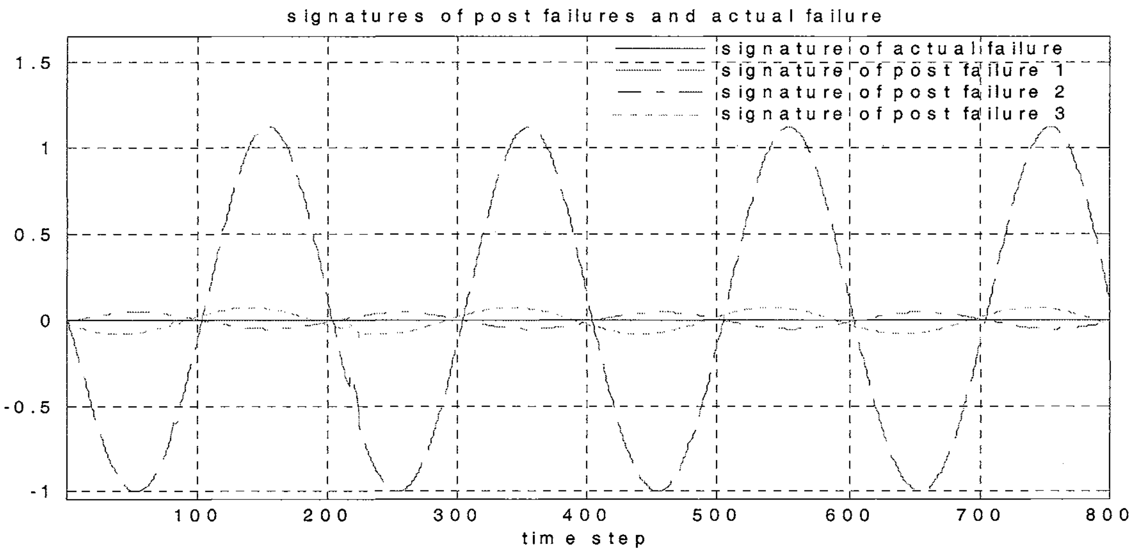


Figure 6.23 Signatures of post failures vs. actual failure dynamics (false alarm situations; FDA simulation)

6.4 Comments and discussions

The design parameters of fault detection and diagnosis, such as the length of the time-shifting evaluating window and the threshold values, all have direct effects on the system performance. Short evaluating windows in fault detection scheme may result in a sensitive and nervous failure detector while a long one may appear to be too slow for proper fault accommodation. Similar conditions also exist for failure diagnosis. Apparently, the best design of these parameters should be on a system-dependent basis. Noisy measurement tests are omitted since the accommodation for the measurement noise should be incorporated within the parameter design process.

CHAPTER VII

CONCLUSIONS

7.1 Summaries of complete research work

Prompted by the increasing demands in system safety and reliability, FDA techniques are quickly becoming one of the most active research areas in the intelligent control community. Yet, many problems remain to be solved. Contemporary fault diagnosis and accommodation techniques are mainly focused on either linear systems or certain classes of nonlinear systems with simple failure scenarios. The major reason is obviously resulted from the fact that the control theory and technology for general nonlinear systems are still not readily available at present. Nevertheless, to face the problems of fault accommodation for a dynamic system in the on-line situation, it is not a reasonable approach to assume that the changes of system dynamics caused by *unanticipated failures* are limited to certain types.

In this dissertation, the on-line fault accommodation control problems under catastrophic system failures are investigated. The major interest is focused on dealing with the unanticipated system failures in the general format. Through discrete-time Laypunov stability theory, the necessary and sufficient conditions to guarantee the system on-line stability and performance under various failure scenarios are derived. An on-line fault accommodation control framework that incorporates an efficient on-line fault detection scheme and effective control law reconfiguration strategy is presented. Because of its capabilities of *self-optimization* and *on-line adaptation*, Artificial Neural Network is used in this research work as the on-line estimator to approximate the unknown failure dynamics.

The on-line fault accommodation control problems are further divided into four different cases according to the prior knowledge of both nominal system and failures. Their corresponding problems and solutions are investigated and discussed case by case through theoretical analysis and extensive simulation studies. The effective control actions to accommodate system failures are automatically computed on-line by the control regulator through the realization of the failure dynamics by the NN estimator based upon partially available information of the failure dynamics. After numerous simulation tests for different cases under various failure situations, the following summaries can be drawn.

1. The prior information of both nominal system dynamics and failures provides substantially useful information for on-line fault accommodation control problems. In both cases 1 and 3, where the multiple-failure dynamics do not explicitly depend upon the current control input, the on-line fault tolerant control problems are relatively easy to be solved since the extrapolation problem of the on-line estimator during the searching process of the effective control signal does not exist. Two different real-time control laws for on-line accommodation of the system failures have been derived for case 1.

2. The design parameter, the length of the time-shifting data window for the training process of the on-line estimator, has substantial effect on the system performance. More training patterns usually result in a better prediction performance. However, it takes much more computational cost to implement. Simulation tests indicate that 20 seems to be a reasonable trade-off number between the system performance and the computational complexity (i.e., the learning process using the Levenberg-Marquardt algorithm with Bayesian regularization usually converges within 10 iterations under the Intel Pentium II-450 dual processors). The network structure of the on-line estimator and training algorithm also have significant influence on the on-line learning process. Using the training algorithm with regularization will generally result in a better system behavior than others and, moreover, this kind of techniques can also relax the network over-fitting problem and eliminate the guesswork in determining an optimal network structure [65].
3. Simulation results suggest that performing the noise reduction or cancellation prior to on-line fault accommodation control will result in a better system performance, if some statistical properties of the noise are available, since the contaminated noisy measurements will mislead all the interpretations of the system behaviors and the on-line estimator.
4. The suggested on-line fault detection scheme has good resistance in miss detection of system failures. However, it also increases the sensitivity of the false alarm situations. Simulation tests in noise-free false alarm situations indicate that the on-line estimator is used to approximate the remaining uncertainty of the

system. In noisy environments, it is possible for the on-line estimator to overfit the noise instead of the actual failure dynamics. Thus, the pre-processing of the contaminated noisy measurements becomes an essential step to prevent the false alarm situations that are obviously a waste of computational source.

5. The proposed intelligent on-line fault accommodation methodology has also been tested on a MIMO system under a multiple unanticipated failure situation. The simulation results indicate the effectiveness of the suggested framework. However, it is important to mention that the simulation is performed under the assumption that there is no conflict to achieve the control missions (performance recoveries for multiple outputs) under the multiple failures. This may not always be true in the real failure situations. The accommodations of some failures may require a certain degree of compromise in other objectives.

To obtain a deeper insight for quantification of the design parameters and the real-time control system, an experimental on-line fault tolerant control test bed for examining the proposed on-line control framework in real hardware is constructed. Four different real-time experiments for case 1 with different control objectives and various unanticipated faults have been performed to evaluate the performance of the proposed fault accommodation technique in real applications under the real-time environment. In general, the effectiveness of the developed on-line fault accommodation control technique for catastrophic system failures has been validated through extensive on-line simulation tests. The successful on-line fault tolerance in real applications has been demonstrated through real-time hardware experiments with the presence of measurement noises.

Based upon the modern intelligent techniques, the unexpected failures can be identified and properly accommodated on-line without a complete realization of the failure dynamics. The price paid for this achievement of the successful control mission relies on a certain degree of computational expense. Simulation results show that, under the Levenberg-Marquardt training algorithm with Bayesian regularization [65-66], the on-line simulation speed can reach 2-3 time steps per second under the Intel Pentium II-450 dual processors. Experimental results indicate that a more powerful computing device such as a computer with higher speed dual processors is mandatory for the on-line real-time fault tolerant control in the real applications under the more general situations (i.e., cases 2-4). Although the currently used dual processors may not be fast enough in many real-time control systems that require higher sampling rate, it is believed that the developed on-line fault accommodation technique can be implemented on-line in most of the real-time control systems in near future, with the continuous performance improvement of microprocessors and semiconductor technology. These results show a promising future of the fault tolerant control for *unknown and unanticipated system failures* in *on-line real-time* fashion based only upon imprecise and insufficient information of the failures and the modern intelligent techniques.

A more sophisticated and complete architecture for intelligent fault diagnosis and accommodation has also been presented by incorporating the developed intelligent fault tolerant control technique with a cost-effective fault detection scheme and a multiple-model based failure diagnosis process to efficiently handle the false alarms, the accommodation of the anticipated failures, and to reduce the unnecessary control effort and computational complexity in on-line situations. A separate simulation study has been performed to test this framework on-line for both special and general cases. Under this architecture, the unnecessary computational waste caused by the false alarm which is the major weakness of the suggested fault detection scheme can be avoided as soon as the effects resulted from the expected disturbance or measurement noises diminished. Simulation results also indicate that, under the multiple-model based failure diagnosis process together with the post-failure control actions, successful fault isolation mission is quickly reached through the multiple-model failure recognition. System performance recovery can be obtained through the multiple-model switching in the post-failure control actions, and significant saving in control effort is achieved during which only the anticipated failure occurs.

7.2 Future research directions

Following the research work completed in this dissertation, several important future research directions are recommended and outlined as follows.

1. The developed on-line fault accommodation technique should be tested under general MIMO cases. For case 1, the on-line control problems become to solve simultaneous equations, which is relatively simple while, in the more general cases, the control problems are both theoretically and technically complicated since the searching of effective control signals to accommodate the multiple failures becomes a multi-objective optimization problem which is still one of the open research issues. Thus, how to systematically reorganize the priorities of the control missions related to both system stability and performance for the general

MIMO system under multiple failures in the on-line situation becomes a challenging work.

2. The presented architecture of the intelligent fault diagnosis and accommodation is designed from both the effective and efficient points of views. By taking advantage of the multiple-model structure, fault recognition can be quickly reached. Nevertheless, under the multiple-model base switching and the on-line system safety point of view, the failure diagnosis and selection of the accommodation actions are restricted to being conservative since the incorrect diagnostic result and the corresponding incorrect and/or ineffective control actions may cause even worse consequences. It is expected that a more sophisticated failure diagnosis process will provide a more precise and quicker "cure" or "treatment" for the faulty system such that more computational burden and risk can be relieved during the accommodation process. (i.e., for example, a better fault diagnosis scheme which is capable of detecting the *"incipient" anticipated failures*.) Thus, improvement on the fault diagnosis technique with the least computational complexity will definitely make its contribution to the fault tolerant control problems.

3. Based upon the developed fault tolerant control technique, the reconfiguration of the effective control actions to properly accommodate the system failures is achieved through an appropriate optimization algorithm and the result of the optimization process directly affects the system performance recovery under the failures. Gradient descent algorithm is considered as an appropriate choice at present due to its reliability and efficiency in on-line situations. However, one

well-known problem of the gradient type of optimization algorithms is the possibility of stucking at a local minimum, which will apparently result in a degrading performance of control recovery. Thus, a more sophisticated and efficient on-line optimization algorithm to possibly relax this problem is worth an investment of research effort.

4. With higher speed processors, hardware experiments should continue for more general situations (i.e., cases 2-4) and more experiments can be designed to further validate the proposed multiple-model based fault diagnosis and accommodation architecture in real-time environments.
5. Further research work should pay more attention on the change of system order (i.e., change of the system relative degree) under failures. In general, if the failure causes the reduction of the relative degree, the control problem may be relatively easy to handle while both system identification and control are more complicated if the system order increases due to failures. In either case, on-line system order estimation may be required prior to the processes of failure estimation and accommodation.

REFERENCES

[1] J. Gertler, "Survey of model-based failure detection and isolation in complex plants," *IEEE Control Systems Magazine*, Vol. 8, No. 6, pp. 3-11, 1988.

[2] M. Polycarpou, and A. Helmicki, "Automated fault detection and accommodation: a learning systems approach," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 25, No. 11, pp. 1447-1458, 1995.

[3] E. Chow, and A. Willsky, "Analytical redundancy and the design of robust failure detection systems," *IEEE Transactions on Automatic Control*, Vol. 29, No. 7, pp. 603-614, 1984.

[4] A. Emani-Naeini, M. Akhter, and S. Rock, "Effect of model uncertainty on failure detection: the threshold selector," *IEEE Transactions on Automatic Control*, Vol. 33, No. 2, pp. 1106-1115, 1988.

[5] W. Ge, and C. Feng, "Detection of faulty components via robust observation," *International Journal of Control*, Vol. 47, No. 2, pp. 581-599, 1988.

[6] X. Lou, A. Willsky, and G. Verghese, "Optimally robust redundancy relations for failure detection in uncertain systems," *Automatica*, Vol. 22, No. 3, pp. 333-344, 1986.

[7] G. Yen, "Reconfigurable learning control in large space structures," *IEEE Transactions on Control Systems Technology*, Vol. 2, No. 4, pp. 362-370, 1994.

[8] P. Frank, "Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy- a survey and some new results," *Automatica*, Vol. 26, No. 3, pp. 459-474, 1990.

[9] M. Polycarpou, and A. Vemuri, "Learning methodology for failure detection and accommodation," *IEEE Control Systems Magazine*, Vol. 15, No. 3, pp. 16-24, 1995.

[10] M. Polycarpou, "Stable learning scheme for failure detection and accommodation," *Proceedings of the IEEE International Symposium on Intelligent Control*, pp. 315-320, 1994.

[11] Z. Gao, and P. Antsaklis, "Reconfigurable control system and design via perfect model following," *International Journal of Control*, Vol. 56, No. 4, pp. 783-798, 1992.

[12] Z. Gao, and P. Antsaklis, "Stability of the pseudo-inverse method for reconfigurable control systems," *International Journal of Control*, Vol. 53, No. 3, pp. 717-729, 1991.

[13] H. Rauch, "Autonomous control reconfiguration," *IEEE Control Systems Magazine*, Vol. 15, No. 6, pp. 37-48, 1995.

[14] M. Bodson, and J. Groszkiewicz, "Multivariable adaptive algorithms for reconfigurable flight control," *IEEE Transactions on Control Systems Technology*, Vol. 5, No. 2, pp. 217-229, 1997.

[15] H. Noura, D. Sauter, and D. Theilliol, "Fault-tolerant control in dynamics systems: application of a winding machine," *IEEE Control Systems Magazine*, Vol. 20, No. 1, pp. 33-49, 2000.

[16] D. Sauter, F. Hamelin, and H. Noura, "Fault tolerant control in dynamic systems using convex optimization," *Proceedings of the IEEE International Symposium on Intelligent Control*, pp. 187-192, 1998.

[17] D. Theilliol, H. Noura, and D. Sauter, "Fault-tolerant control method for actuator and component faults," *Proceedings of the IEEE Conference on Decision and Control*, pp. 604-609, 1998.

[18] J. Jiang, and Q. Zhao, "Fault tolerant control system synthesis using imprecise fault identification and reconfigurable control," *Proceedings of the IEEE International Symposium on Intelligent Control*, pp. 169-174, 1998.

[19] J. Jiang, and Q. Zhao, "Reconfigurable control based on imprecise fault identification," *Proceedings of the American Control Conference*, pp. 114-118, 1999.

[20] J. Farrell, T. Berger, and B. Appleby, "Using learning techniques to accommodate unanticipated faults," *IEEE Control Systems Magazine*, Vol. 13, No. 3, pp. 40-49, 1993.

[21] C. Jacobson, and C. Nett, "An integrated approach to controls and diagnostics using the four parameter controller," *IEEE Control Systems Magazine*, Vol. 11, No. 6, pp. 22-29, 1991.

[22] M. Polycarpou, and A. Vemuri, "Learning approach to fault tolerant control: an overview," *Proceedings of the IEEE International Symposium on Intelligent Control,* pp. 157-162, 1998.

[23] A. Vemuri, and M. Polycarpou, "Neural-network-based robust fault diagnosis in robotic systems," *IEEE Transactions on Neural Networks*, Vol. 8, No. 6, pp. 1410-1420, 1997.

[24] A. Vemuri, and M. Polycarpou, "Robust nonlinear fault diagnosis in input-output systems," *International Journal of Control*, Vol. 68, No. 2, pp. 343-360, 1997.

[25] A. Trunov, and M. Polycarpou, "Automated fault diagnosis in nonlinear multivariable systems using a learning methodology," *IEEE Transactions on Neural Networks*, Vol. 11, No. 1, pp. 1–11, 1999.

[26] M. Demetriou, and M. Polycarpou, "Incipient fault diagnosis of dynamical systems using online approximators," *IEEE Transactions on Automatic Control*, Vol. 43, No. 11, pp. 1612-1617, 1998.

[27] H. Khalil, *Nonlinear Systems*, New York, NY: Macmillan, 1992.

[28] A. Isidori, *Nonlinear Control Systems*, New York, NY: Springer-Verlag, 1989.

[29] K. Narendra, and K. Parthasathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, Vol. 1, No. 1, pp. 4-27, 1990.

[30] T. Apostol, *Mathematical Analysis*, Menlo Park, CA: Addison-Wesley, 1974.

[31] F. Beaufays, and E. Wan, "Relating real-time backpropagation and backpropagation-through-time: an application of flow graph interreciprocity," *Neural Computation*, Vol. 6, No. 2, pp. 297-305, 1994.

[32] M. Chow, and S. Yee, "An adaptive backpropagation through time training algorithm for a neural controller," *Proceedings of the IEEE International Symposium on Intelligent Control*, pp. 170-175, 1991.

[33] P. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of IEEE*, Vol. 78, No. 10, pp. 1550-1560, 1990.

[34] K. Narendra, and K. Parthasathy, "Gradient methods for the optimization of dynamical systems containing neural networks," *IEEE Transactions on Neural Networks*, Vol. 2, No. 2, pp. 252-262, 1991.

[35] D. Psaltis, A. Sideris, and A. Yamaura, "Neural controllers," *Proceedings of the IEEE International Conference on Neural Networks*, pp. IV 551-558, 1987.

[36] M. Saerens, and A. Soquet, "A neural controller," *Proceedings of the First IEE Conference on Artificial Neural Networks*, pp. 211-215, 1989.

[37] K. Narendra, and S. Mukhopadhyay, "Adaptive control using neural networks and approximate models," *IEEE Transactions on Neural Networks*, Vol. 8, No. 3. pp. 475-485, 1997.

[38] J. Michael, and R Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Computation*, Vol. 6, No. 2, pp 181-214, 1994.

[39] C. Juang, and C. Lin, "An on-line self-constructing neural fuzzy inference network and its applications," *IEEE Transactions on Fuzzy Systems*, Vol. 6, No. 1, pp. 12-32, 1998.

[40] J. Platt, "A resource allocating network for function interpolation," *Neural Computation*, Vol. 3, No. 2, pp. 213-225, 1991.

[41] J. Nie, and D. Linkens, "Learning control using fuzzied self-organizing radial basis function network," *IEEE Transactions on Fuzzy Systems*, Vol. 1, No. 4, pp. 280-287, 1993.

[42] R. Reed, "Pruning algorithms- a survey," *IEEE Transactions on Neural Networks*, Vol. 4, No. 5, pp. 740-747, 1993.

[43] H. Berenji, and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, pp. 724-740, 1992.

[44] G. Ng, *Application of Neural Networks to Adaptive Control of Nonlinear Systems*, New York, NY: John Willy & Sons, 1997.

[45] J. Spall, "Multivariable stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control*, Vol. 37, No. 3, pp. 332-341, 1992.

[46] J. Spall, "Implementation of the simultaneous perturbation algorithm for stochastic optimization," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 34, No. 3, pp. 817-823, 1998.

[47] J. Spall, and J. Cristion, "Model-free control of nonlinear stochastic systems with discrete-time measurements," *IEEE Transactions on Automatic Control*, Vol. 43, No. 9, pp. 1198-1210, 1998.

[48] X. Ji, and B. Familoni, "A diagonal recurrent neural network-based hybrid direct adaptive SPSA control system," IEEE *Transactions on Automatic Control*, Vol. 44, No. 7, pp. 1469-1473, 1999.

[49] J. Spall, and J. Cristion, "Nonlinear adaptive control using neural networks: estimation based on a smoothed form of simultaneous perturbation gradient approximation," *Statistica Sinica*, Vol. 4, pp. 1-27, 1994.

[50] J. Spall, and J. Cristion, "A neural network controller for systems with unmodeled dynamics with applications to wastewater treatment," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 27, No. 3, pp. 369-375, 1997.

[51] J. Nechyba, and Y. Xu, "Human-control strategy: abstraction, verification, and replication," *IEEE Control Systems Magazine*, Vol. 17, No. 5, pp. 48-61, 1997.

[52] J. Kiefer, and J. Wolfowitz, "Stochastic estimation of a regression function," *Annals of Mathematical Statistics*, Vol. 23, pp. 462-466, 1952.

[53] J. Bum, "Multidimensional stochastic approximation methods," *Annals of Mathematical Statistics*, Vol. 25, pp. 737-744, 1954.

[54] G. Aly, A. Badr, and Z. Binder, "Multi-model control of MIMO system: location and control algorithm," *International Journal of Systems Science*, Vol. 19, No. 9, pp. 1687-1698, 1988.

[55] G. Nagib, W. Gharieb, and Z. Binder, "Qualitative multi-model control using a learning approach," *International Journal of Systems Science*, Vol. 19, No. 9, pp. 855-869, 1988.

[56] K. Hornik, M. Stinchombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, Vol. 2, No. 5, pp. 359-366, 1989.

[57] M. Hagan, H. Demuth, and M. Beale, *Neural Network Design*, Boston, MA: PWS Publishing, 1996.

[58] R. Murray-Smith, and J. Johansen, *Multiple Model Approaches to Modeling and Control*, Bristol, PA: Taylor & Francis, 1997.

[59] K. Narendra, and J. Balakrishnan, "Improving transient response of adaptive control systems using multiple models and switching," *Proceedings of the IEEE Conference on Decision and Control*, pp. 1067-1072, 1993.

[60] K. Narendra, J. Balakrishnan, and M. Ciliz, "Adaptation and learning using multiple models, switching, and tuning," *IEEE Control Systems Magazine*, Vol. 15, No. 3, pp. 37-51, 1995.

[61] K. Narendra, and J. Balakrishnan, "Adaptive control using multiple models," *IEEE Transactions on Automatic Control*, Vol. 42, No. 2, pp. 171-187, 1997.

[62] A. Rajagopal, and K. Krishnamurthy, "Adaptive control using multiple models and model weighting," *Proceedings of the ASME Dynamics systems and Control Division*, pp. 805-812, 1996.

[63] T. Parisini, and S. Sacone, " Fault diagnosis and controller re-configuration and hybrid approach," *Proceedings of the IEEE International Symposium on Intelligent Control,* pp. 163-167, 1998.

[64] F. Kerestecioglu, *Change Detection and Input Design in Dynamical Systems*, Wiley, NY: John Willy & Sons, 1993.

[65] H. Demuth, and M. Beale, *Neural Network Toolbox User's Guide*, Version 3, Natick, MA: The Math Works, 1998.

[66] D. Mackey, "Bayesian interpolation," *Neural Computation*, Vol. 4, No. 3, pp. 415-447, 1992.

[67] E. Misawa, "Discrete-time sliding mode control for nonlinear systems with unmatched uncertainties and uncertain control vector," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 119, pp. 503-512, 1997.

[68] J. Slotine, and W. Li, *Applied Nonlinear Control*, Englewood Cliffs, NJ: Prentice-Hall, 1991.

[69] G. Yen, and L. Ho, "Fault tolerant control: an intelligent sliding model control strategy," *Proceedings of the American Control Conference*, pp. 4204-4208, 2000.

[70] L. Ho, and G. Yen, "Intelligent on-line fault tolerant control for unanticipated catastrophic failures," submitted to *International Journal of Control.*

[71] J. Zurada, *Introduction to Artificial Neural Systems*, Boston, MA: PWS Publishing, 1992.

[72] K. Narendra, "Neural networks for control: theory and practice," *Proceedings of IEEE*, Vol. 84, No. 10, pp. 1385-1406, 1996.

[73] K. Passino, and S. Yurkovich, *Fuzzy Control*, Menlo Park, CA: Addison Wesley Longman, 1997.

[74] T. Ross, *Fuzzy Logic with Engineering Applications*, New York, NY: McGraw-Hill, 1995

[75] S. Naidu, E. Zafiriou, and T. Mcavoy, "Use of neural networks for sensor failure detection in a control system," *IEEE Control Systems Magazine*, Vol. 10, No. 3, pp. 49-55, 1990.

[76] E. Zitzler, and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 4, pp. 257-271, 1999.

[77] S. Drakunov, and V. Utkin, "On discrete-time sliding modes," *Proceedings of the IFAC Symposium on Nonlinear Control System Design*, pp. 484-489, 1988.

[78] K. Furtua, "Sliding mode control of a discrete system," *System & Control Letters*, Vol. 14, pp. 145-152, 1990.

[79] C. Milosavlievic, "General conditions for the existence of a quasisliding model on the switching hyperplane in discrete variable structure systems," *Automation and Remote Control*, Vol. 46, No. 3, pp. 307-314, 1985.

[80] W. Su, U. Ozguner, and S. Drakunov, "Discrete time sliding mode control in nonlinear sampled-data systems," *Proceedings of the 31st Annual Allerton Conference on Communication, Control, and Computing*, pp. 809-818, 1993.

[81] E. Garcia, and P. Frank, "Multiple fault isolation in linear systems," *Proceedings of the IEEE Conference on Decision and Control*, pp. 3114-3115, 1999.

[82] E. Garcia, R. Seliger, and P. Frank, "Nonlinear decoupling approach to fault isolation in linear systems," *Proceedings of the American Control Conference*, pp. 2867-2871, 1998.

[83] J. Gertler, and Q. Luo, "Direct identification of nonlinear parity relations-a way around the robustness problem," *Proceedings of the IEEE Conference on Decision and Control*, pp. 78-83, 1998.

[84] J. Gertler, and Y. Hu, "Direct identification of optimal nonlinear parity models," *Proceedings of the IEEE International Symposium on Computer Aided Control System Design*, pp. 164-169, 1999.

[85] X. Zhang, T. Parisini, and M. Polycarpou, "Robust parametric fault detection and isolation for nonlinear systems," *Proceedings of the IEEE Conference on Decision and Control*, pp. 3102-3107, 1999.

[86] X. Zhang, M. Polycarpou, and T. Parisini, "Abrupt and incipient fault isolation of nonlinear uncertainty systems," *Proceedings of the American Control Conference*, pp. 3713-3717, 2000.

[87] J. Gertler, M. Costin, X. Fang, Z. Kowalczuk, M. Kumwer, and R. Monajemy, "Model based diagnosis for automotive engines," *IEEE Transactions on Control Systems Technology*, Vol. 3, pp. 61-69, 1995.

[88] J. Gertler, and M. Kunwer, "Optimal residual decoupling for structured diagnosis and disturbance insensitivity," *International Journal of Control*, Vol. 61, pp. 395-421, 1995.

[89] J. Gertler, and D. Singer, "A new structural framework for parity equation based failure detection and isolation," *Automatica*, Vol. 26, pp. 381-388, 1990.

[90] Y. Zhang, and J. Jiang, "An interacting multiple-model based fault detection, diagnosis and fault-tolerant control approach," *Proceedings of the IEEE Conference on Decision and Control*, pp. 3593-3598, 1999.

[91] R. Pickhardt, "Application of adaptive controllers to a solar power plant using a multiple-model description," *Proceedings of the American Control Conference*, pp. 2895-2899, 1998.

[92] X. Li, and Y. Zhang, "Numerical robust implementation of multiple-model algorithms," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 36, No. 1, pp. 266-278, 2000.

[93] A. Blumel, E. Hughes, and B. White, "Design of robust fuzzy controllers for aerospace applications," *Proceedings of the International Conference of the North American Fuzzy Information*, pp. 439-442, 1999.

[94] A. Karimi, and I. Landau, "Robust adaptive control of a flexible transmission system using multiple models," *IEEE Transactions on Control Systems Technology*, Vol. 8, No. 2, pp. 321-331, 2000.

[95] J. Kim, K. Lee, and C. Lee, "On the applications of the interacting multiple model algorithm for enhancing noisy speech," *IEEE Transactions on Speech and Audio Processing*, Vol. 8, No. 3, pp. 349-352, 2000.

[96] J. Chun, M. Kim, H. Jung, and S. Hong, "Shape optimization of electromagnetic devices using immune algorithm," *IEEE Transactions on Magnetics*, Vol. 33, No. 2, pp. 1876-1879, 1997.

[97] J. Chun, H. Jung, and S. Hahn, "A study on comparison of optimization performances between immune algorithm and other heuristic algorithms," *IEEE Transactions on Magnetics*, Vol. 34, No. 5, pp. 2972-2975, 1998.

[98] M. Marchesi, G. Molinari, and M. Repetto, "A parallel simulated annealing algorithm for the design of magnetic structures," *IEEE Transactions on Magnetics*, Vol. 30, No. 5, pp. 3439-3442, 1994.

[99] M. Marchesi, "A new class of optimization algorithm for circuit design and modeling," *Proceedings of the IEEE International Symposium on Circuits and Systems*, Vol. 2, pp. 1691-1695, 1998.

[100] Y. Tanaka, and T. Yoshida, "An applications of reinforcement learning to manufacturing scheduling problems," *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 4, pp. 534-539, 1999.

[101] C. Juang, J. Lin, and C. Lin, "Genetic reinforcement learning through symbiotic evolution for fuzzy controller design," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 30, No. 2, pp. 290-302, 2000.

[102] S. Ottner, "Optimising television commercial air-time by means of a genetic algorithm," *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 761, 2000.

[103] S. Ho, and M. Huang, "An efficient quadratic curve approximation using an intelligent genetic algorithm," *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 766, 2000.

[104] J. Jiang, "Design of reconfigurable control systems using eigenstructure assignment," *International Journal of Control*, Vol. 59, No. 2, pp. 395-410, 1994.

[105] Y. Zhang, and J. Jiang, "Design of proportional-integral reconfigurable control systems via eigenstructure assignment," *Proceedings of the American Control Conference*, pp. 3732-3736, 2000.

[106] P. Maybeck, and R. Stevens, "Reconfigurable flight control via multiple model adaptive control methods," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 27, No. 3, pp. 470-480, 1991.

[107] Y. Zhang, and X. Li, "Detection and diagnosis of sensor and actuator failures using IMM estimator," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 34, No. 4, pp. 1293-1313, 1998.

[108] K. Laparo, M. Buchner, and K. Vasudeva, "Leak detection in an experimental heat exchanger process: a multiple model approach," *IEEE Transactions on Automatic Control*, Vol. 36, No. 2, pp. 167-177, 1991.

[109] J. Mendel, *Lessons in Estimation Theory for Signal Processing, Communications, and Control*, Englewood Cliffs, NJ: Prentice-Hall, 1995.

[110] R. Dorf, and R. Bishop, *Modern Control Systems, 7th edition*, Menlo Park, CA: Addison-Wesley Publishing Co., 1995.

VITA

Liang-Wei Ho

Candidate for the Degree of

Doctor of Philosophy

Thesis: ON-LINE FAULT DIAGNOSIS AND ACCOMMODATION FOR SYSTEM FAILURES

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Taipei, Taiwan, R.O.C., On April 26, 1969, the third son of Chih-Kuo Ho and Hsiu-Lien Weng.

Education: Graduate from Tamkang University, Taipei, Taiwan, R.O.C. with the Degree of Bachelor of Science in Mechanical Engineering in January, 1993; receive the Degree of Master of Science in Mechanical Engineering, Oklahoma State University, in July, 1997. Complete the requirements for the Degree of Doctor of Philosophy with a major in Electrical Engineering at Oklahoma State University in December, 2000.

Experience: Mechanical Engineer, Combustion and Heat Transfer Co., Taipei, Taiwan, R.O.C. (01/1993-05/1994). Sale & Mechanical Engineer, Great Wall Industrial Co., Taipei, Taiwan, R.O.C. (05/1994-08/1995). Teaching Assistant of a senior design class, Aerospace System Design in School of Mechanical and Aerospace Engineering, Oklahoma State University (01/1997-08/1997). Design a CAN (Controller Area Network) simulator program for network behaviors of CAN-Based control systems (07/1997). Assistant for American Control Conference, San Diego, CA, 1999, and IEEE International Conference on Control Applications, Anchorage, Alaska, 2000. Develop "Intelligent on-line Fault Tolerant Control methodology" and the real-time FTC test bed supported by the U.S. Air Force Office of Science Research under the Grant F49620-98-1-0049 and U.S. Environmental Protection Agency under the Grant R826273-01-0. Research Assistant of Intelligent Systems & Control Lab., ECE Dept., Oklahoma State University (09/1997-present).