Development and Evaluation of Fault-tolerant Flight Control Systems

Final Report

(Period Covered: June 15, 2000 ~ December 14, 2003)

by
Yong D. Song, Ph.D., P.E.

Department of Electrical Engineering
North Carolina A&T State University, Greensboro, NC 27411

NAG4-211

<u>Technical Monitor</u>

Dr. Kajal Gupta

February 27, 2004

NAG4-211

Summary

The research is concerned with developing a new approach to enhancing fault tolerance of flight control systems. The original motivation for fault-tolerant control comes from the need for safe operation of control elements (e.g. actuators) in the event of hardware failures in high reliability systems. One such example is modern space vehicle subjected to actuator/sensor impairments. A major task in flight control is to revise the control policy to balance impairment delectability and to achieve sufficient robustness. This involves careful selection of types and parameters of the controllers and the impairment detecting filters used. It also involves a decision, upon the identification of some failures, on whether and how a control reconfiguration should take place in order to maintain a certain system performance level.

In this project new flight dynamic model under uncertain flight conditions is considered, in which the effects of both ramp and jump faults are reflected. Stabilization algorithms based on neural network and adaptive method are derived. The control algorithms are shown to be effective in dealing with uncertain dynamics due to external disturbances and unpredictable faults. The overall strategy is easy to set up and the computation involved is much less as compared with other strategies. Computer simulation software is developed. A serious of simulation studies have been conducted with varying flight conditions.

Table of Contents

1. Introduction

A growing importance has been given in the last few years to the design of a flight control system able to accomplish the reconfiguration of the aircraft following battle damage [1]-[5]. While there exists many control methods for flight control, most of the existing methods are based on "healthy" aircraft model in that the control algorithms are derived from system dynamics without any faults in the system. Evidently, automatic control plays a crucial role in achieving these features.

This work is concerned with the development and analysis of neural network based control algorithms for flight vehicles with faulty dynamics. A neural-adaptive method to enhance flight vehicle stability under unpredictable faults is presented. The method is based on the dynamic model coupled with the effect of unpredictable faults and disturbances. It is shown that with the proposed strategy the effect of faults on the system performance can be effectively suppressed. The novelty of the proposed approach also lies in the fact that it is fairly easy to set up and the computation involved is much less as compared with other strategies. An example is used to verify the validity of the proposed approach.

2. Problem Formulation

The major thrust of this research is directed toward the stabilization of flight vehicle subject to subsystem failures. The dynamic model considered is of the following form,

$$M(.)\ddot{r}+N(\dot{r},r)+d(t)=Bu \tag{1}$$

with

$$M(.)=M_0\pm\beta(t-T_f)M_f(.)$$

$$N(.)=N_0\pm\beta(t-T_f)N_f(.)$$

where $M\in R^{nxn}$ is the inertia matrix depending on the mass property of the vehicle structure, $r\in R^n$ denotes the coordinates/states of the system, $N\in R^{nxn}$ represents the term related to damping and stiffness properties of the system, $d \in R^n$ represents the external disturbance, $u \in R^m$ denotes the control signals of the actuator, and $B \in R^{nxm}$ is the control input influence matrix reflecting the location of the actuator distributed along the vehicle structure. In most cases B is a matrix with 0 and 1 as its elements.

The notation "$\pm$" is used in the matrices M and N to emphasize the fact that both the direction and the magnitude of the uncertainties due to modeling approximation and possible faults are not available for control design.

Note that the $\beta(t-T_f)M_f(.)$, $\beta(t-T_f)N_f(.)$, and $\beta(t-T_f)B_f(.)$ are used to model the unknown portion of the dynamics due to a fault occurring at time $t \geq T_f$. The function $\beta(t-T_f)$ takes the form

$$\beta(t-T_f)=\begin{cases} 0 & t<T_f \\ 1-e^{-\alpha_f(t-T_f)} & t \geq T_f \end{cases}$$

which covers both jump and continuously time-varying (ramp) faults. For the system to admit a feasible solution under faulty condition, we need to assume that there exists a positive constant $c_0 > 0$ such that

$$\left\|\beta(t-T_f)M_f(.)\right\| + c_0 \leq \left\|M_0\right\| \tag{2a}$$

This assumption implies that the fault condition under consideration is within attackable range in that it does not dominate the value of M under normal conditions. In such a case, the symmetric and positive definite property of the mass matrix still holds, i.e.,

$$M(.) = M(.)^T > 0 \tag{2b}$$

Whenever a fault occurs, system dynamics may be changes in many different ways. In such a case, our primary objective is to design a proper control to maintain the stability of the system. Note that essentially two system modes are involved under any fault: *critical modes* (denoted by r_c) and *non-critical modes* (denoted by r_{nc}). The critical modes should be stabilized explicitly for they are directly related to system safety.

For control design we assume that the rank of B is m, which implies physically that there are m actuators mounted at appropriate positions in the vehicle for fault accommodation. Obviously, only the mode, with which there is an actuator, can be actively damped out. Therefore, to make full use of the available actuators, we consider the case of dim(r_c) = m, thus dim(r_{nc}) = n-m. Under the collocation condition (i.e., sensors and actuators are located at the same places), we have

$$r_c = B^T r \in \Omega_c \subset R^m, \quad r_{nc} = P_j r \in \Omega_{nc} \subset R^{n-m} \tag{3}$$

where P_j is a projection matrix. Clearly, $\Omega = \Omega_c \oplus \Omega_{nc}$, here Ω denotes the whole coordinate space of practical interest. In this study, our objective is to develop a control scheme that reduces both $\|r_c\|$ and $\|\dot{r}_c\|$ to an acceptable level, i.e.,

$$\|r_c\| \le \varepsilon_1, \quad \|\dot{r}_c\| \le \varepsilon_2$$

where ε_1 and ε_2 are small numbers. A strategy achieving this objective is developed in the next section. The following lemma is needed to develop the control strategy.

Lemma:

Let $b > 0$ be a constant chosen by the designer, and define

$$s = (B^T M^{-1} B)^{-1} B^T (\dot{r} + br) \tag{4a}$$

If $ran(B) = m$, then

(i) $\|s\| \to 0$ as $t \to \infty$ implies that $\|r_c\| \to 0$ and $\|\dot{r}_c\| \to 0$ as $t \to \infty$.

(ii) $\|s\| \le c < \infty$ as $t \ge T_c$ implies that $\|r_c\| \le c_r < \infty$ and $\|\dot{r}_c\| \le c_{\dot{r}} < \infty$ as $t \to \infty$.

Proof:

Let

$$z = \dot{r}_c + br_c \tag{4b}$$

then from (3) and (4a)-(4b) we have

$$s = (B^T M^{-1} B)^{-1} z \tag{4c}$$

With $\mathrm{rank}(B) = m$ and the property of M as in (2b), it follows that $B^T M^{-1} B$ is symmetric and positive definite. Therefore $\|s\| \to 0$ as $t \to \infty$ implies that $\|z\| \to 0$ as $t \to \infty$. By the definition of z as in (4b), the results as stated in the lemma are readily established.

As motivated by this lemma, we first construct the following signal

$$z = \dot{r}_c + b r_c = B^T (\dot{r} + b r) \tag{5}$$

where $b > 0$ is a positive number chosen arbitrarily, then we focus on making z go to zero or be less than a small constant which is determined by the required precision.

Note that this signal is available as long as r_c and $\dot{r}_c$ are measurable. From (1) and (5) it can be shown that

$$\dot{z} = P(.)u + L(.) \tag{6a}$$

where

$$P(.) = B^T M^{-1} B \tag{6b}$$

$$L(.) = bB^T \dot{r} - B^T M^{-1}(N + d) \tag{6c}$$

It should be emphasized that P and L are unavailable due to the fact that the structure contains fault and external disturbance. This calls for a more dedicated control design approach for such a system.

It is interesting to note that the system model (1), although uncertain, possesses the following properties.

Properties:

1) $M^{-1} = M^{-T} > 0$.

2) If B is full row rank, then there exists constants $c_{\min}$ and $c_{\max}$ such that

$$c_{\min} \|z\|^2 \le z^T B^T M^{-1} B z \le c_{\max} \|z\|^2$$

where $c_{\max} \geq c_{\min} > 0$ denotes the maximum and minimum eigenvalues of $B^T M^{-1} B$ respectively.

Proof:

The proposition can be easily shown by noting that when B is full row rank $B^T M^{-1} B$ is symmetric and positive definite.

In our later controller design and stability analysis, we only use the fact that such constants $c_{\min}$ and $c_{\max}$ exist, while the actual estimation of these constants is not needed. To close this section, we re-state the control objective as follows: find a smooth control u without using the detailed information of $P(.)$ and $L(.)$ such that the system tracking error vector z locks in a compact set containing origin as time goes by, i.e., $\|z\| \leq \mu$ as $t > T$ (μ is a small constant related to control precision).

3. Neuro-adaptive Fault-tolerant Control

The nonlinear and uncertain nature of $P(.)$ and $L(.)$ represents the major challenge in control system design. The typical approach is to re-organize $L(.)$ as a linear combination of some nonlinear functions. This has been the basis for most stable adaptive controllers, known as the *regressor based method* in literature [6]. For complex systems such as composite structures, this method involves complicated design procedures and heavy computations. In the early work of neural network based control, a common practice is to construct multilayer NNs and train these networks via gradient algorithms [7]-[9], [12]-[13] to approximate $P(.)$ and $L(.)$ respectively. Recognizing the potential instability associated with these algorithms, several researchers have proposed NN control schemes using on-line and stable training mechanisms based on Lyapunov stability theory [11]-[12].

In this work we develop a neuro-adaptive control scheme for the system (1). As a first step, we "reconstruct" $L(.)$ via the first NN unit as

$$L_{NN} = L_0 + W_I^T \psi_I(z) \tag{7}$$

where L_0 represents the priori (crude) estimate of L, $W_I \in R^{m \times m}$ is the "optimal" weights of the NN, $\psi_I \in R^m$ is the basis function whose selections typically include linear function, sigmoidal function, hyperbolic tangent function, radial basis functions etc. [7]-[15].

Let the discrepancy between L and L_{NN} be denoted by

$$\varepsilon = L - L_{NN} \tag{8}$$

which is termed as *NN reconstruction error* in literature. With the support of the approximate theory of multilayer neural networks[7]-[8], such an error can be made arbitrarily small provided that certain conditions (e.g., sufficiently large number of neurons, sufficient smoothness of L etc.) are satisfied. This has been the basis for many NN based control laws [16]-[19].

Note that, however, in reality one can only construct a network with a finite number of neurons and that the nonlinear function being reconstructed is not smooth enough, as in the case of a jump fault. Thus the approximation capabilities of the network can be guaranteed only on a subset of the entire system state space. In other words, during the system operation, the reconstruction error is bounded by a constant sometimes, but unbounded at other times. For this reason, the NN reconstruction error should be treated carefully in control design because simply assuming that $\|\varepsilon\|$ is bounded by a constant does not warrant the success of the control scheme. For executing the idea, in our development, we remodel reconstruction error as follows:

$$\varepsilon = c\varepsilon_1 + (1-c)\varepsilon_2 \tag{9}$$

with

$$c = \begin{cases} 1 & \text{if the NN works satisfactorily} \\ 0 & \text{if the NN fails to work well} \end{cases}$$

To make the problem solvable, we make the assumption that

$$\|\varepsilon_1\| \le d_0 < \infty, \quad \text{and} \quad \|\varepsilon_2\| \le \| L - L_0 - W_I^T \psi_I \| \equiv d_1(.).$$

Remark 2

The above model seems to be more reasonable to reflect the fact that in general the reconstruction error could take any form, depending on many factors such as the basis function, the number of neurons, the structure of the networks, the weight updating algorithms as well as the original nonlinear function to be approximated.

The point to be made is that d_0 and $d_1(.)$ are still unavailable and should not be included in the control scheme directly. Another major challenge stems from the fact that we do not know when the NN behaves well. Sanner and Slotine [16] proposed a method of partitioning the state space into A_d and A_d^c and assumed that the NN works satisfactorily within A_d. The problem with this method is that it may be practically hard to find such subset A_d^C at which the NN works fine during the system operation.

Here we take a different method to address this issue. Namely, we incorporate two NN units in the control scheme, with the first NN unit compensating the lumped nonlinearities/uncertainties and the second NN unit attenuating the effects due to the NN reconstruction error and the other resulting uncertainties. The overall control scheme is given by

$$u = -k_0 z + \hat{L}_{NN} + u_c \tag{10a}$$

where

$$\hat{L}_{NN} = L_0 + \hat{W}_I^T \psi_I(z) \tag{10b}$$

denotes the control action of the first NN unit and u_c is a compensating signal from the second NN unit to be specified. Here $k_0 > 0$ is a constant chosen by the designer, $\hat{W}$ is the estimation of the ideal weight values, L_0 is the prior (crude) estimation of L and ψ_I is again the basis function vector. With the control of the above scheme, the original system (6a) becomes

$$\dot{z} = -k_0 P z + \tilde{W}_I^T \psi_I - P u_c + \eta \tag{11}$$

where $\tilde{W}_I = W_I - \hat{W}_I$ is weight estimation error and

$$\eta = \varepsilon + (I-P)(L_0 + \hat{W}_I^T \psi_I) \tag{12}$$

is the resultant uncertainty to be attenuated by the second NN unit.

The design of the second NN unit consists of two steps. The first step is to formulate the "magnitude" of η, which allows the m-dimensional nonlinear function to be "funneled" into a scalar nonlinear function. The second step is to build a NN to fight against the resulting scalar quantity, i.e., the "funneling effect" of η. For this reason the second NN is named as a funneling NN.We use the Euclidean norm to funnel the effect of η, i.e.

$$\|\eta\| \leq \gamma \tag{13a}$$

To find the expression for γ, we make the assumption that there exists some unknown constants ω_i and some scalar nonlinear functions Y_i such that

$$\|L - L_0\| \leq \sum_{i=1}^{N_1} \omega_i Y_i$$

where N_1 is an integer, which, together with (8) and (12), leads to

$$\gamma = \sum_{i=1}^{N_1+3} \omega_i Y_i \tag{13b}$$

with

$$\omega_{N_1+1} = d_0, \quad \omega_{N_1+2} = \| W_I \|, \ \omega_{N_1+3} = \sup_{z \in R_c \subset R^m} \| I - P \|,$$

$$Y_{N_1+1} = 1, \ Y_{N_1+2} = \| \psi_I \|, Y_{N_1+3} = \| \hat{W}_I \psi_I \|$$

Note that while Y_i are known scalar functions, ω_i are unknown in general, which makes γ unobtainable. With the funneling NN, γ is estimated as

$$\hat{\gamma} = \sum_{i=1}^{N} \hat{\omega}_i \psi_{II}(Y_i) = \hat{\omega}^T \psi_{II}(Y) \tag{14}$$

where $N = N_1 + 3$, $\hat{\omega}_i$ denotes the estimated weights and $\psi_{IIi}(.)$ are some basis functions satisfying

$$\psi_{\text{II}i}(Y_i) \geq Y_i \tag{15}$$

Based upon $\hat{\gamma}$, the control action of the second *NN* unit is given by

$$u_c = \frac{\sum_{\text{i=1}}^{N} \psi_{\text{II}_\text{i}} \hat{\omega}^T \psi_{\text{II}}(Y)}{\sum_{\text{i=1}}^{N} \psi_{\text{II}_\text{i}} \|z\| + v} z \tag{16}$$

where $v > 0$ is a small number chosen by the designer. To synthesize the tuning algorithms for $\hat{W}_\text{I}$ and $\hat{\omega}$, we consider the following Lyapunov candidate function

$$V = V_1 + V_2 + V_3 \tag{17a}$$

with

$$\begin{aligned} V_1 &= \frac{1}{2} z^T z, \quad V_2 = \frac{1}{2g_\text{I}} tr(\tilde{W}_\text{I}^T \tilde{W}_\text{I}), \\ V_3 &= \frac{1}{2g_{\text{II}} c_{\text{min}}} (\omega - c_{\text{min}} \hat{\omega})^T (\omega - c_{\text{min}} \hat{\omega}) \end{aligned} \tag{17b}$$

where $g_\text{I} > 0$ and $g_{\text{II}} > 0$ are free design parameters affecting weight learning rate and control performance, $c_{\text{min}} > 0$ is the minimum eigenvalue of P. Its derivative along the vector field (11) is

$$\begin{aligned} \dot{V} &= -k_0 z^T P z + z^T \tilde{W}_\text{I}^T \psi_\text{I} + z^T \eta - z^T P u_c + \dot{V}_2 + \dot{V}_3 \\ &\leq -k_0 c_{\text{min}} \|z\|^2 + Q + tr\left[\tilde{W}_\text{I}^T (\psi_\text{I} z^T - g_0^{-1} \dot{\hat{W}}_\text{I})\right] \end{aligned} \tag{18}$$

with

$$Q = z^T \eta - z^T P u_c - \frac{1}{g_{\text{II}}} (\omega - c_{\text{min}} \hat{\omega})^T \dot{\hat{\omega}}$$

Before presenting the weight tuning algorithms, let us focus on Q. With u_c as in (16) and the upper norm bound on η as in (13a), we have

$$Q \leq \| z \| \omega^T Y - \frac{c_{\min} \sum_{i=1}^{N} \psi_{\mathrm{II}_i} \| z \|^2 \hat{\omega}^T \psi_{\mathrm{II}}}{\sum_{i=1}^{N} \psi_{\mathrm{II}_i} \| z \| + v} - \frac{1}{g_{\mathrm{II}}} (\omega - c_{\min} \hat{\omega})^T \dot{\hat{\omega}}$$

$$= (\omega - c_{\min}\hat{\omega})^T \left[\frac{\sum_{i=1}^{N} \psi_{\mathrm{II}_i} \| z \|^2 \psi_{\mathrm{II}}}{\sum_{i=1}^{N} \psi_{\mathrm{II}_i} \| z \| + v} - \frac{1}{g_{\mathrm{II}}} \dot{\hat{\omega}} \right] + v\omega_m \tag{19}$$

where $\omega_m = \max\ (\omega_i)$. Note that in deriving (19) we have used the facts that

$$\omega^T Y \leq \omega^T \psi_{\mathrm{II}} \leq \omega_m \sum_{i=1}^{N} \psi_{\mathrm{II}_i}$$

and

$$\frac{\sum_{i=1}^{N} \psi_{\mathrm{II}_i} \| z \|}{\sum_{i=1}^{N} \psi_{\mathrm{II}_i} \| z \| + v} \leq 1, \quad \forall v > 0$$

Combining (18) and (19), we come up with the following weight tuning algorithms:

$$\dot{\hat{W}}_{\mathrm{I}} = -\sigma_{\mathrm{I}} \hat{W}_{\mathrm{I}} + g_{\mathrm{I}} \psi_{\mathrm{I}} z^T$$

$$\dot{\hat{\omega}} = -\sigma_{\mathrm{II}} \hat{\omega} + g_{\mathrm{II}} \frac{\sum_{i=1}^{N} \psi_{\mathrm{II}_i} \| z \|^2 \psi_{\mathrm{II}}}{\sum_{i=1}^{N} \psi_{\mathrm{II}_i} \| z \| + v} \tag{20}$$

Consequently,

$$\dot{V} \leq -c_{\min} k_0 \| z \|^2 + \frac{\sigma_{\mathrm{I}}}{g_{\mathrm{I}}} tr(\tilde{W}_{\mathrm{I}}^{\mathrm{T}} \hat{W}_{\mathrm{I}})$$

$$+ \frac{\sigma_{\mathrm{II}}}{g_{\mathrm{II}}} (\omega - c_{\min} \hat{\omega})^T \hat{\omega} + v\omega_m \tag{21}$$

Completing the square we can further rewrite (21) as:

$$\begin{aligned}\dot{V} &\le -c_{\min}k_0 \| z \|^2 - \frac{\sigma_{\mathrm{I}}}{2g_{\mathrm{I}}} tr(\tilde{W}_{\mathrm{I}}^{\mathrm{T}} \tilde{W}_{\mathrm{I}}) \\ &\quad - \frac{\sigma_{\mathrm{II}}}{2g_{\mathrm{II}}c_{\min}} (\omega - c_{\min}\hat{\omega})^T (\omega - c_{\min}\hat{\omega}) + \delta \\ &= -\lambda_1 V_1 - \lambda_2 V_2 - \lambda_3 V_3 + \delta\end{aligned} \tag{22}$$

where

$$\delta = v\omega_m + \frac{\sigma_{\mathrm{I}}}{2g_{\mathrm{I}}} tr(W_{\mathrm{I}}^{\mathrm{T}} W_{\mathrm{I}}) + \frac{\sigma_{\mathrm{II}}}{2g_{\mathrm{II}}} \omega^T \omega$$

$$\lambda_1 = 2c_{\min}k_0, \lambda_2 = \sigma_{\mathrm{I}}, \lambda_3 = \sigma_{\mathrm{II}} \tag{23}$$

are constants independent of z. Note that (22) implies that $\|z\|, \hat{W}_{\mathrm{I}}$ and $\hat{\omega}$ are bounded. Furthermore, from (22), we have

$$\dot{V} \le -k_0 c_{\min} \|z\|^2 + \delta \tag{24}$$

It is seen that $\dot{V}$ is negative as long as

$$\|z\| \ge \left(\frac{\delta}{k_0 c_{\min}} \right)^{1/2} \tag{25}$$

Therefore, z is confined in the region $\Omega = \left\{ z \middle| \|z\| \le \left(\frac{\delta}{k_0 c_{\min}} \right)^{1/2} \right\}$. From previous lemma, we establish that the control error is bounded.

4. Simulation Verification

Simulation on a numerical example is conducted to test the effectiveness of the proposed method. The parameters involved in the algorithms are chosen as

$$k_0 = 10,\ \sigma_I = \sigma_{II} = 0.5,\ g_I = g_{II} = 0.5,\ v = 0.01,\ \Delta T = 0.01\sec$$

The basis functions for the first NN unit are chosen as:

$$\psi_I(c_{1i}, X, \alpha) = \frac{1 - e^{-\alpha(\|X\| - c_{1i})^2}}{1 + e^{-\alpha(\|X\| - c_{1i})^2}}$$

$$(c_{1i} = 0.2, 0.4, \ldots, \|X\| = \begin{bmatrix} r \\ \dot{r} \end{bmatrix})$$

where $\alpha = 0.5$. The basis functions for the second NN unit are selected as in the following:

$$\psi_{II}(c_{2i}, X, \alpha) = \frac{c_{2i} - c_{2i} e^{-\alpha(\|X\| - c_{2i})^2}}{\mu + e^{-\alpha(\|X\| - c_{2i})^2}} \qquad (c_{2i} = 0.3, 0.7, \ldots, \|X\| = \begin{bmatrix} r \\ \dot{r} \end{bmatrix})$$

where $\mu = 0.5$. Ten NN basis functions were used for each of the NN units: $\psi \in R^j, \quad j = 10$. All the weights are initialized at zero in the simulation.

As mentioned earlier, any fault due to damage, element malfunction, and so forth, plays an important role in the behavior of the system. To have a realistic simulation it is necessary to model the aerodynamic effects of the damage. For this purpose, it may be said that the changed aircraft dynamics following damages to a stabilator surface is mainly due to an instantaneous change of the normal force coefficient of the surface, with the axial force coefficient being usually negligible and with the moment coefficient being proportional, through the geometry, to the normal force coefficient. Therefore we consider the expressions for fault in the simulation as follows,

$$M_f(.) = \beta \cdot 0.3 \cdot M_0$$

$$N_f(.) = \beta \cdot (N_1 \dot{r} + N_2 r + N_3)$$

where $\beta = 1 - e^{-m(t - T_h)}$ is the fault function, *Th=15 sec* is the time instant at which the fault occurs, *m=0.2* is the fault coefficient. The results are presented in Figures 1-4, where Figure 1 is the stabilization error for critical and non-critical modes, Figure 2 shows the control signals, Figure 3 and Figure 4 illustrated the on-line updating of weights for the two NN units.

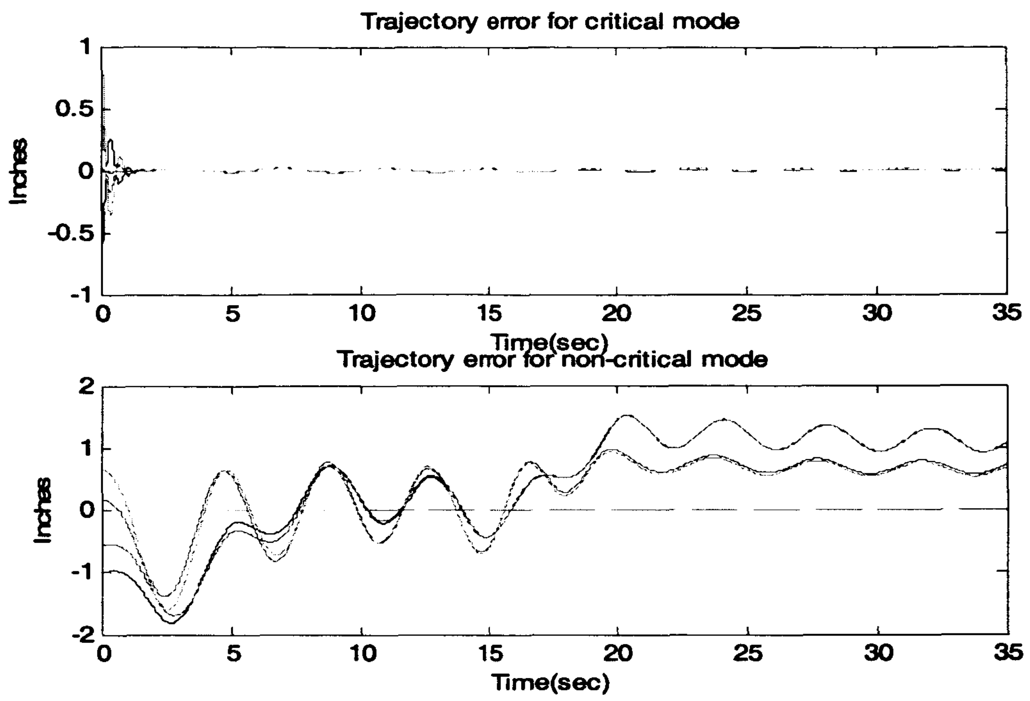


Figure 1 - System Performance Under Faults

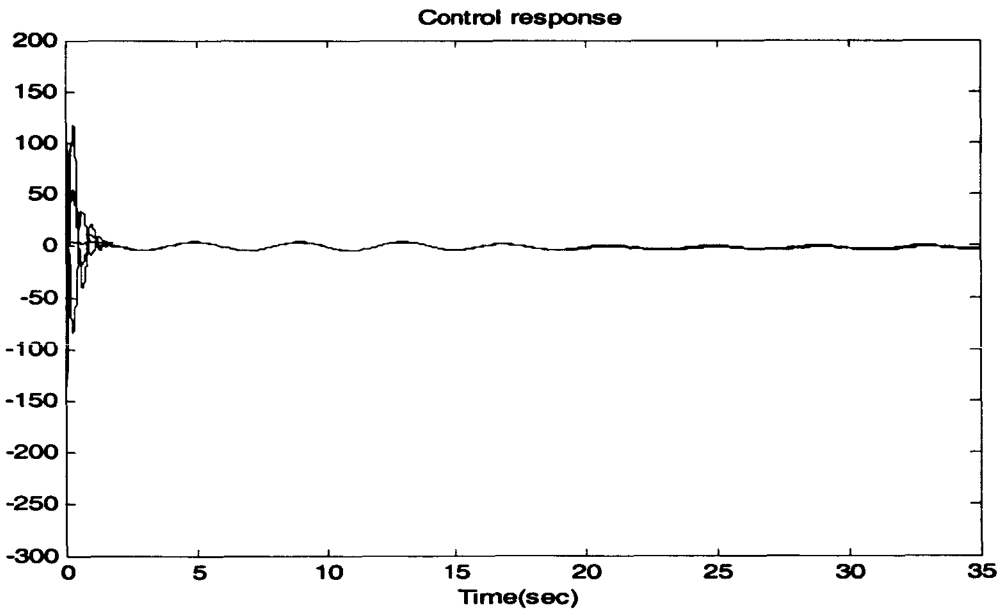


Figure 2 - Control Input Under Faults

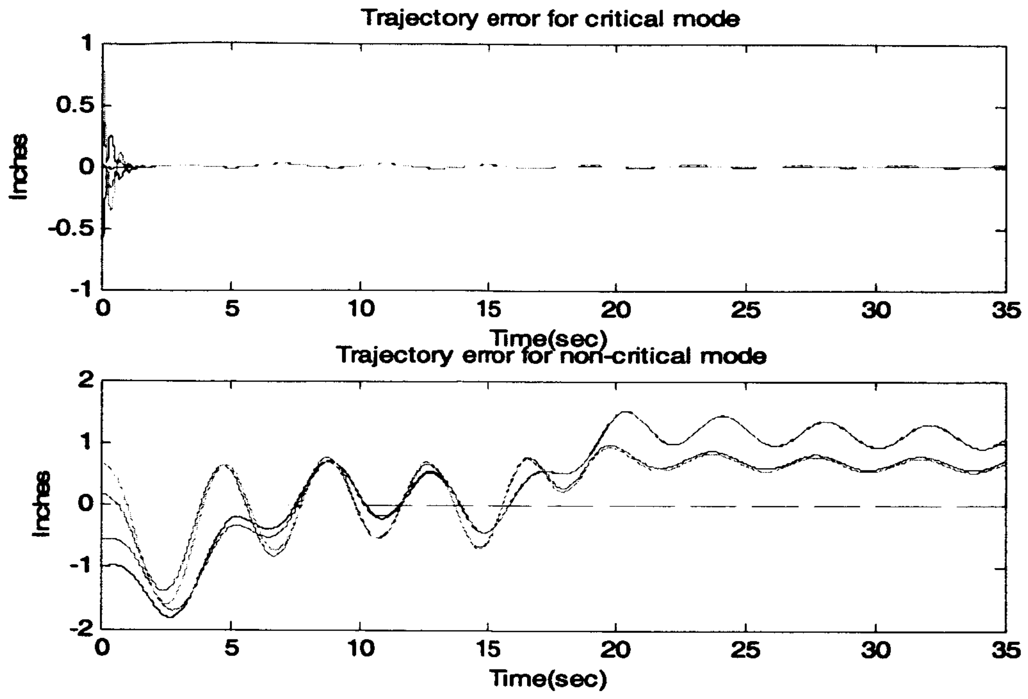


Figure 1 - System Performance Under Faults

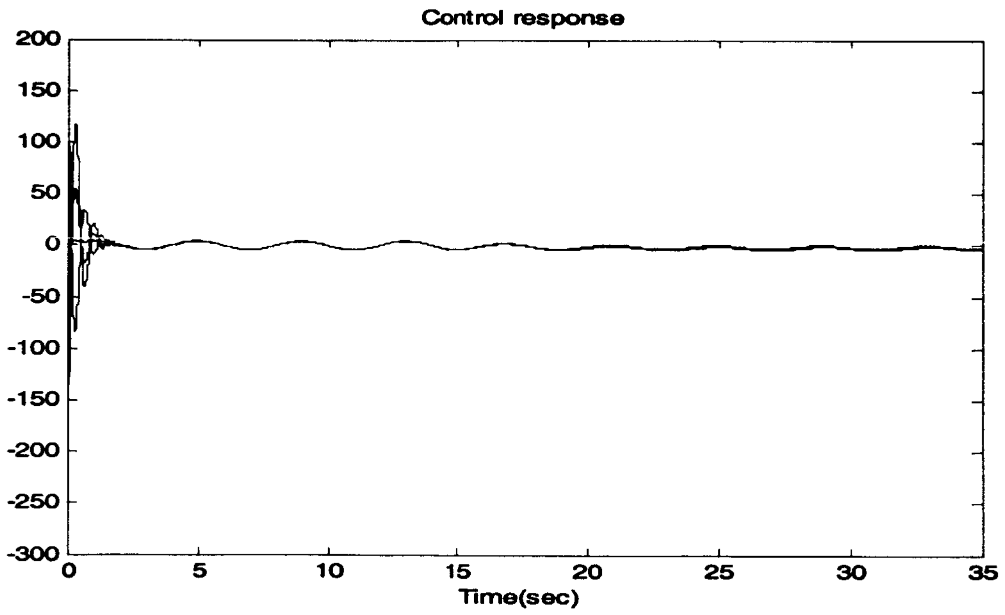


Figure 2 - Control Input Under Faults

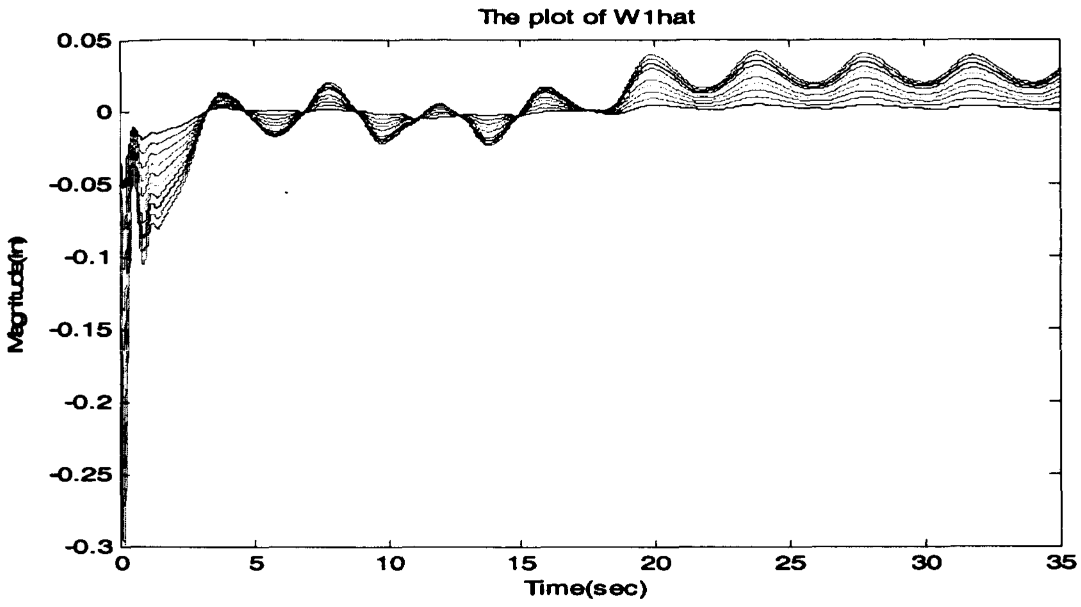


Figure 3 - Weights Updating for 1st Neural Network

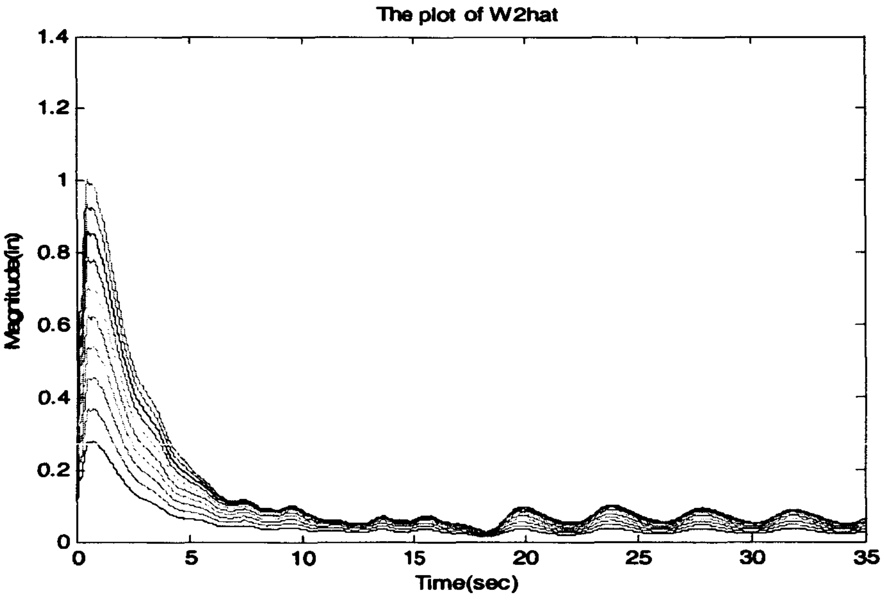
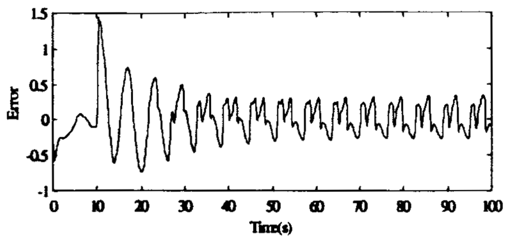
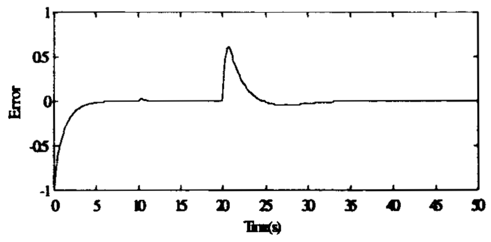


Figure 4 - Weights Updating for 2nd Neural network

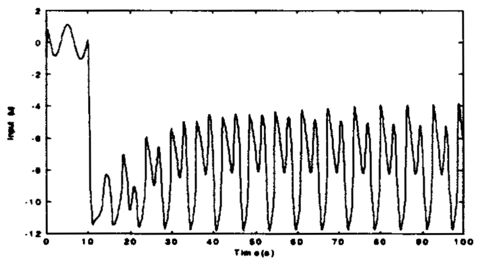
Also, for test purpose, we conducted a simulation on one critical mode with both jump and ramp faults. The results are presented in Figures 5-8, where Figure 5 is the control error under the ramp and jump faults occurring at $T_f = 20$ seconds, Figure 6 shows the control inputs and Figures 7 and Figure 8 are the updated weights of the networks. As can be seen, with the proposed control method the system remains stable under severe faults and all the internal signals are bounded and smooth.

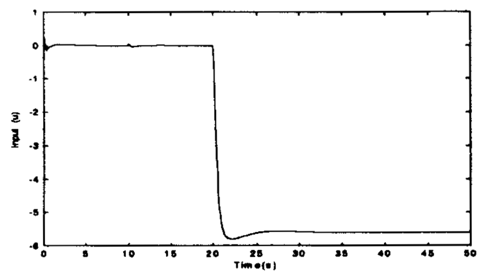




a) ramp fault b) jump fault

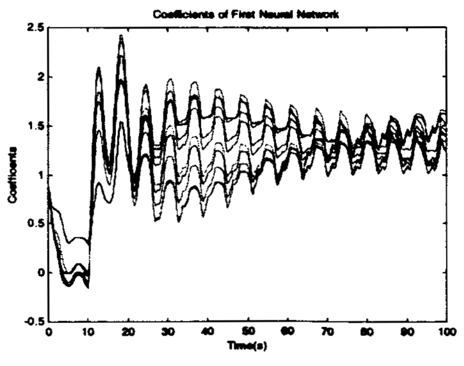
Figure 5 - System Performance Under Faults

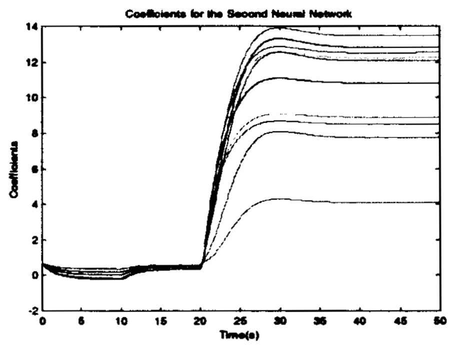




a) ramp fault b) jump fault

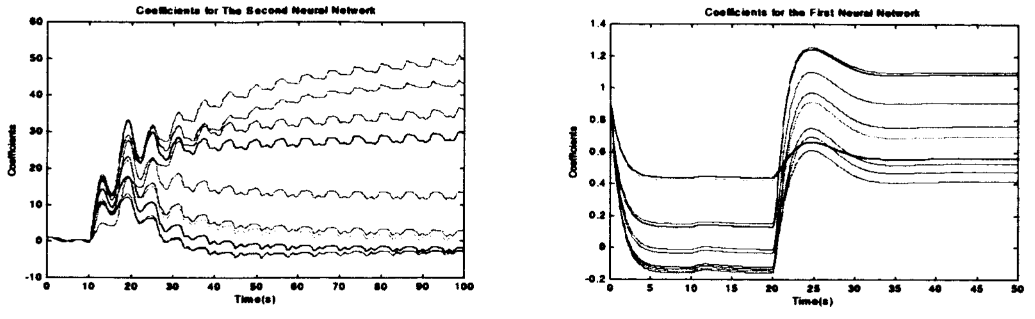
Figure 6 - Control Input Under Faults





a) ramp fault b) jump fault

Figure 7 - Weights Updating for the 1st Network I



a) ramp fault b) jump fault

Figure 8-Weights Updating for the 2nd Network II

5. Conclusion and Future Work

Whenever an actuator fault occurs, the element of *B* is of the form,

$$\Re_{ii} = \begin{cases} 1 \\ \sigma_{ii0} \pm \beta(t - T_f)\sigma_{ii}(.) \end{cases}$$

where

$$\left|\sigma_{ii0} \pm \beta(t - T_f)\sigma_{ii}(.)\right| = \begin{cases} 0 \\ \delta \in (0,1) \\ 1 \end{cases}$$

$\left|\sigma_{ii0} \pm \beta(t - T_f)\sigma_{ii}(.)\right| = 0$ implies that the actuator is totally failing

$\left|\sigma_{ii0} \pm \beta(t - T_f)\sigma_{ii}(.)\right| = 1$ implies the actuation system is health

$\left|\sigma_{ii0} \pm \beta(t - T_f)\sigma_{ii}(.)\right| = \delta < 1$ the fading actuation system is involved

We are currently investigating the control problem under the following fault conditions,

$$0 < \left|\sigma_{ii0} \pm \beta(t - T_f)\sigma_{ii}(.)\right| \leq 1$$

This, physically, involves a fading actuation fault in which the actuating system has not totally failed yet. The results will be reported in the forth coming publication.

Acknowledgment

The PI would like to thank Dr. Kajal Gupta for his technical support and guidance throughout this project.

References

[1]. L. P. Lyanch. and S. S. Banda, "Active control for vibration damping, " In S. N. Atluri and A. K. Amos (Eds.), Springer Series in Computational. Mechanics. New York: Springer, 1988, 239-261.

[2]. J. Lu, J. S. Thorp and H.-D. Chiang,, "modal control of large flexible space structures using collocated actuators and sensors.", IEEE Transactions on Automatic Control,37, 10, 1 (1992).

[3]. Iyer, A., and S. N. Singh, "Variable structure slewing control and vibration damping of flexible spacecraft. Acta Astronautica, 25, 1 (1991), 1-9.

[4]. A. Bhaya and C. A. Desor, "On the design of large space structures" IEEE Transaction on Automatic Control, AC-30, 11 1985.

[5]. Y. D. Song and T. Mitchell, "Active Damping Control of Vibrational Systems," IEEE Trans. On Aerospace and Electronic Systems, Vol. 32, No. 2, 1996, pp. 569-577.

[6]. Y. D. Song, "Adaptive Motion Tracking Control of Robot Manipulators-Non-regressor based Approach," Int. J. Control, Vol. 63, No. 1, 1996, pp. 41-54.

[7]. K. J. Hunt, D. Sbarbaro, R. Zbikowski, and P. J. Gawthrop, "Neural networks for control systems- a survey," Automatica , 26, 1085- 1112, 1992.

[8]. W. T. Miller, R. S. Sutton, and P. J. Werbos, Eds, "Neural network for control, 3rd." MIT Press, Cambridge, MA, 1992.

[9]. K. S. Narendra and A. M. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Trans. on Neural Networks*, Vol. 1, No. 1, 1990, pp. 4-27.

[10]. F. C. Chen and C. C. Liu, "Adaptively Controlling Nonlinear Contiguous-Time Systems Using Multilayer Neural Networks," *IEEE Trans. on Auto. Control*, Vol. 39, No. 6, 1994, pp. 1306-1310.

[11]. F. C. Chen and H. K. Khalil, "Adaptive Control of a Class of Nonlinear Discrete-Time Systems Using Neural Networks," *IEEE Trans. Auto. Control*, Vol. 40, No. 5, 1995, pp. 1306-1310.

[12]. T, Yamada and T. Yabuta, "Neural Network Controller Using Autotuning Method for Nonlinear Functions," *IEEE Trans. on Neural Networks*, Vol. 1, No. 4, July, 1992, pp. 595-601.

[13]. G. Cybenko, "Approximation by Superposition of a Sigmoidal Function," Mathematics of Control, Signals, and Systems, No. 2, 1989, pp. 303-314.

[14]. F. L. Lewis, K. Liu and A. Yesildirek, "Neural Net Robot Controller with Guaranteed Tracking Performance," *IEEE Trans. on Neural Networks*, Vol. 6, No. 3, 1995, pp. 703-715.

[15]. A. Karakasoglu, S. I. Sudharsanan and M. K. Sundareshan, "Identification and Decentralized Adaptive Control Using Dynamical Neural Networks with Application to Robotic Manipulators," *IEEE Trans. on Neural Networks*, Vol. 4, No. 6, 1993, pp. 919-930.

[16]. R. M. Sanner and J. J. Slotine, "Guassian Networks for Direct Adaptive Control", IEEE Trans. Neural Networks, Vol. 3, No. 6, 1992, pp. 837-863.

[17]. E. Tzirkel-Hancock and F. Fallside, "Stable Control of Nonlinear Systems Using Neural Networks," Int. J. of Robust and Nonlinear Control, Vol. 2, No. 1, 1992, pp. 63-85.

[18]. K. Funahshi, "On the Approximate Realization of Continuous Mappings by Neural Networks," Neural Networks, Vol. 2, pp. 183-192, 1989.

[19]. M. Teshnehlab and K. Watanabe, "Self Tuning of Computed Torque Gains by Using Neural Networks with Flexible Structures," *IEE Proc.-Control Theory Appl.*, Vol. 141, No. 4, 1994, pp. 235-242.