JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 16, No. 6, November–December 1993

Aircraft Failure Detection and Identification Using Neural Networks

Marcello R. Napolitano,[*] Ching I. Chen,[†] and Steve Naylor[‡]
West Virginia University, Morgantown, West Virginia 26506

In this paper, a neural network is proposed as an approach to the task of failure detection following damage to an aerodynamic surface of an aircraft flight control system. Several drawbacks of other failure detection techniques can be avoided by taking advantage of the flexible learning and generalization capabilities of a neural network. This structure, used for state estimation purposes, can be designed and trained on line in flight and generates a residual signal indicating the damage as soon as it occurs. From an analysis of the cross-correlation functions between some key state variables, the identification of the damage type can also be achieved. The results of a nonlinear numerical simulation for a damaged control surface are reported and discussed.

Nomenclature

a_y	= output of accelerometer aligned with the aircraft y body axis, g
a_z	= output of accelerometer aligned with the aircraft z body axis, g
b	= wing span, ft
c	= aerodynamic coefficient
f	= generic function
H	= value of the neuron of the hidden layer
I	= value of the input layer coefficient
I_x	= moment of inertia about the x body axis, slug ft^2
I_y	= moment of inertia about the y body axis, slug ft^2
I_z	= moment of inertia about the z body axis, slug ft^2
I_{xy}	= product of inertia in the x-y body plane, slug ft^2
I_{xz}	= product of inertia in the x-z body plane, slug ft^2
I_{yz}	= product of inertia in the y-z body plane, slug ft^2
i	= index of the neurons of the hidden layer
j	= index of the neurons of the output layer
K	= discrete time index
k	= number of neurons in the hidden layer
L	= lower bound of the modified sigmoid function
l	= index of the input layer coefficients
M	= measurement vector
m	= number of input layer coefficients
mac	= mean aerodynamic chord, ft
n	= number of neurons in the output layer
net	= argument of the modified sigmoid function
O	= neural network output vector
p	= aircraft angular velocity around the x body axis, rad/s
q	= aircraft angular velocity around the y body axis, rad/s
R	= vector of the cross-correlation function
r	= aircraft angular velocity around the z body axis, rad/s
S	= wing surface, ft^2
SC	= surface command displacement
T	= slope of the modified sigmoid function
t	= time
U	= upper bound of the modified sigmoid function
V	= matrix of the weights between the input and the hidden layers
W	= aircraft weight
W	= matrix of the weights between the hidden and the output layers
Y	= vector of the parameters to be estimated
α	= angle of attack, rad
β	= angle of sideslip, rad
Γ	= vector of the thresholds of the neurons of the hidden layer
Δ	= increment
δ	= surface deflection, rad or deg
η	= learning rate
Θ	= vector of the thresholds of the neurons of the output layer
θ	= pitch angle, rad

Subscripts

A	= aileron
D	= differential stabilator
dam	= damaged conditions
H	= symmetric stabilator
init	= starting of the estimation
L	= total aerodynamic lift, lb
L	= left
m	= total aerodynamic pitching moment, lb ft
R	= right
R	= rudder

Introduction

BECAUSE of the ever rising costs of new military aircraft and their relatively low procurement, 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. Contemporary fly-by-wire control systems automatically detect sensor failures and reconfigure to mask the effects of these failures. They do not, however, cope with damaged aircraft surfaces.

In this study we will consider what is believed to be the worst possible scenario, i.e., battle damage to a control surface, involving a stuck actuator along with a missing part of the control surface. This has a realistic condition for a typical air-to-ground maneuver of an aircraft subjected to ground fire or even for an air-duel combat situation. From a mathematical point of view, such a condition implies changes of the values of several aerodynamic stability derivatives along with requiring a constant value for the deflection of the damaged control surface.[1-3] However, the approach that will be introduced later can be applied, without any loss of generality, to the

Received Dec. 5, 1991; revision received Dec. 20, 1992; accepted for publication Dec. 20, 1992. Copyright © 1993 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Assistant Professor, Department of Mechanical and Aerospace Engineering. Member AIAA.

†Research Associate, Department of Mechanical and Aerospace Engineering.

‡Graduate Student, Department of Mechanical and Aerospace Engineering. Member AIAA.

more common sensor and actuator failure problem. Note that this classification of failures does not include unsolvable problems (for example, wings falling off) where the aircraft cannot be saved. Following the battle damage and/or the generic failure, the flight control task is to utilize whatever control resources remain to regain control of the aircraft, to prevent further damage by excessive air loads, and to give the crew enough time to assess the options. In a time sequence, the failure detection and identification task can be considered to be just a step, particularly the first and, for this reason, most crucial step of the overall flight control reconfiguration problem.

To implement a reconfiguration strategy, a variety of control surfaces [speed brakes, wing flaps, differential (dihedral) canards, spoilers, rudder below fuselage] and thrust control mechanisms (differential thrust, thrust vectoring, canted engines) may be introduced. The selection of the control mechanism to be used in a reconfiguration is a function of several factors such as control effectiveness, increased aircraft complexity and costs, weight penalties, and increased aerodynamic drag due to the increased wetted area, with applicability depending on aircraft type. The following quantities from operative sensors along with a fully operational flight computer are assumed to be available for reconfiguration purposes[1-3]: aircraft angular velocity and linear velocity in the three body axes and aircraft attitude and angle of attack. It would also be desirable to have information on the actuator position for each control surface. The task of battle damage and/or generic failure accommodation may include all or some of the following tasks:

1) Executive control task essentially provides synchronization of the remaining tasks.

2) Failure detection and identification task detects significant abnormalities and searches for the cause or a set of probable causes.

3) Damage model estimation task generates a mathematical model of the aircraft dynamics considered to reflect changes due to the damage.

4) Reconfiguration law design task determines what actions should be taken to recover the damaged aircraft, by using alternative control surfaces or thrust control mechanisms.

5) Feedback structure redesign task calculates a new set of feedback gains to retain stability and desirable handling qualities even after the damage.

Eventually another task to be introduced is a pilot advisory function. While the computers of the flight control system are reconfiguring the aircraft, it would be desirable for the pilot to be able to see on a cockpit display which control surface has been damaged and what actions, if possible under current conditions, could eventually benefit the reconfiguration.

In this paper, a neural network (NN) and an analysis of the cross-correlation functions of key state variables are suggested as alternative approaches to the failure detection and identification problem. The paper is organized as follows. The next section describes the goal of the failure detection, lists some of the problems encountered with common techniques, and suggests the implementation of an NN as an on-line state estimator. Then, the extended back-propagation algorithm to be used for the on-line training of the NN is reviewed. Next, the subsequent section discusses the aerodynamic modeling of the aircraft with a damaged elevator, shows the numerical simulation of the failure detection, and explains the cross-correlation function approach for the failure identification. The final section summarizes the paper with conclusions and recommendations.

Failure Detection via Neural Networks

Failure detection (FD) theory is a widely investigated problem addressed in flight control research. Generally speaking, FD implies a constant monitoring of the measurable output variables of the aircraft system. At nominal conditions these variables follow some known patterns within a degree of uncertainty due to system disturbance (such as atmospheric turbulence) and sensor measurement noise. However, when failures of any component of the flight control system occur, the observable output variables deviate from their nominal and somewhat predictable trajectories.[4-8] Most theoretical failure detection and identification (FDI) techniques described in the literature are based on spotting these deviations of the output variables from predictable trajectories. It is clear that this approach implies a knowledge by the flight control system of these predictable or somewhat reasonable trajectories of the system at each possible operating condition.

Typical problems associated with the application of these FD techniques are the following: they are applicable only to linear time-invariant systems; demonstrated only for low-order systems; demonstrated only when the system model is identical to the filter or the observer model, even if, due to constraints on computational speed and power, only reduced-order filters or observers are practically implemented; demonstrated only for large signal-to-noise ratios; and impossible to account for correlations in the estimates.

The goal of this paper is to introduce an alternative approach to these techniques.[4-8] Most of the previous problems are due to the fact that the flight control system uses on-line calculated or precalculated system models to generate a finite knowledge of predicted trajectories, sometimes at the cost of a great computational burden. These models are usually linear and time invariant with Gaussian modeled system and sensor noises. It is clear that the success of the FD is not guaranteed when some of these key assumptions are violated with the signal-to-noise ratio playing a key rule. An approach for achieving higher degrees of success for the FD is the introduction of a more general technique where a knowledge of the system dynamics can be developed on line without an excessive computational cost. This can be possible through the introduction of modern neural network theory.[9-12]

Neural network is a re-emerging technology; there is a vast literature regarding NNs and their various applications that dates back to the 1940s. However, it is only since the mid-1980s that there has been an increasing number of applications of neural networks.[9-12] A turning point was given by the introduction of the back-propagation algorithm in 1985.[10] However, to date, there have been a relatively limited number of applications of NN for the design of flight control systems, even though much interest has been generated very recently. Particularly, a number of investigations have suggested the use of NN for parameter estimation[13] and for controller design.[14,15] The suggested use of an NN in this study is quite different and is essentially as an on-line state estimator.[16] In this sense it may be said that the neural network is an alterna-

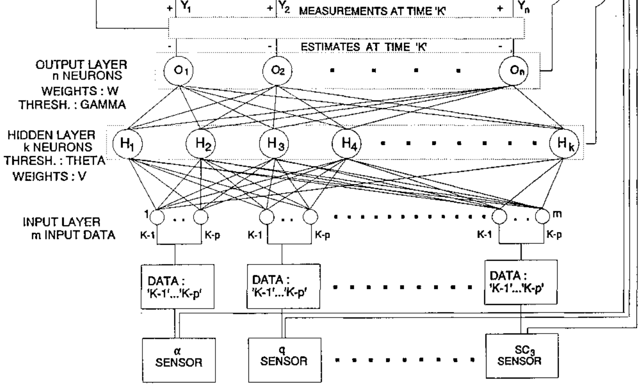


Fig. 1 Three-layer neural network implemented for failure detection.

tive to classic state estimation structures such as Leunberger observers and Kalman filters (full or reduced order).[17] A key feature of a neural network used for state estimation is that it will learn the dynamics of the system during a training session made of several training cycles, with training data coming from either previous computer numerical simulations or from actual on-line data, as is the case of failure detection. After each training cycle the NN will know a little more about the dynamics of the system. Mathematically this will mean that the estimation error has decreased somewhat. Among the other features of an NN is the ability to learn the dynamics of a nonlinear system as well as those of a linear system. The main difference is that nonlinear dynamic systems require a longer training. Also, note that there is no need for assuming the initial conditions of the system. For these reasons an NN has several potential advantages over classic estimation structures. References 9-12 provide a solid background of the theoretical principles on which the functioning of an NN is based.

In this study we refer to an on-line learning NN structure of feed forward type with interlayer connections. For this type of NN, Ref. 16 suggests that a three-layer structure (that is with one input, one hidden, and one output layer) is the best compromise between simplicity of the NN and accuracy of the state estimates. For such an architecture Ref. 16 also shows that the accuracy of the estimates after a certain number of training cycles is practically independent of the number of neurons of the hidden layer. Clearly a smaller number of neurons would imply a lower complexity of the NN structure and greater computational speed. Furthermore, Ref. 16 suggests that the accuracy of the state estimates is improved by increasing the number of input data of the NN, that is, by using sensor data from previous instants as in classic estimation theory. However, this will add complexity to the structure. Finally, the advantages of using an NN structure with more than two layers appear to be justified only if the state estimation is attempted for a nonlinear dynamic system.

A suitable architecture for a three-layer NN implemented for failure detection is shown in Fig. 1. The input of this NN will be the sensor data for the measurements of α, q, θ, p, r, a_z, a_y, SC_1, SC_2, and SC_3, from time $K-1$ down to time $K-p$ where p is an integer. Note that the surface command displacements, sum of the pilot and autopilot commands, given as input to the NN will allow the NN to distinguish between an abrupt maneuver and an atmospheric turbulence.

A weighted manipulation of these data through particular activation functions, with the options of adding thresholds, at each neuron of the hidden and output layer will furnish state estimates of the same motion variables α, q, θ, p, r, a_z, and a_y at time K; these estimates are then compared with the value of the states from the sensors for the same variables at the same time K. Given that a high-performance military aircraft is essentially a nonlinear time varying dynamic system, the main idea is to have a permanent on-line learning for the NN (shown in Fig. 1) in the flight computer. The difference between the actual and the estimated values of the state variables is used by a gradient-based optimization method to calculate adjustments of the weights and thresholds of the NN to decrease the estimation error at the next time step. Note that this is essentially an on-line in-flight training with the NN being just a piece of software of the flight control system. Let us assume now that the aircraft is flying at a given condition of its flight envelope and, at a certain point, battle damage occurs to the elevator. It is clear that the aircraft will experience one or more of the following[3]: altered trim conditions, changes in the aerodynamic forces and moments, and changed control effectiveness.

This implies that the vector of the measurements at the instants immediately after the damage will be very different from the estimates provided by the NN. If the estimation error becomes bigger than a given threshold (to account for false alarms, such as the ones induced by atmospheric turbulence or noise corrupted sensors data) then this will be an effective tool for failure detection. The determination of the weighting coefficients and the thresholds for the hidden and output layers of the NN structure shown in Fig. 1 is performed on line using the extended back-propagation (BP) algorithm,[18] which is a gradient-based optimization method, as the standard BP from which it originated.[10] Such a technique has been developed very recently. Note that if we had a number C of hidden layers then we would have had to determine $C+1$ sets of weighting coefficients and relative thresholds.

Extended Back-Propagation Algorithm

There are some drawbacks associated with the standard BP algorithm.[10-12] First, the learning speed is slow for large-order systems, implying long training processes. Second, in the presence of local minimums the standard BP may sometimes not be able to find a set of weights for the NN. These two problems are mutually related and interactive. For example, it is has been shown[11] that the learning speed can be improved by setting a larger learning rate. Unfortunately, this also enhances the possibility for the standard BP algorithm to be trapped in a local minimum or to oscillate around the global minimum. However, the biggest problem for the application of the standard BP algorithm to the failure detection problem is that the sigmoid activation function usually implemented has a very limited output range, that is, [0,1]. This problem could be overcome by simply modifying the sigmoid function to have $[-1,1]$ as output range and then normalizing all of the NN outputs with respect to that range. However, this approach implies a preliminary knowledge of the maximum and minimum values of each of the states to be estimated and does not guarantee the same degree of accuracy for each variable to be estimated. All of these problems may be solved by introducing a heterogenous network.[18]

The heterogenous network means that each neuron in the hidden and output layers of the NN has its own output capability of updating free parameters. The main idea in this modification is to incorporate three new free parameters into each neuron such that each neuron is able to change its output range and the slope of the sigmoid activation function. The algorithm described in this section is a simple extension of the standard BP. Although this modification implies the introduction of several equations, they are very easy to follow and very suitable for software implementation.

In this section we refer to the three-layer NN architecture shown in Fig. 1 with m input data. In the forward path of the standard BP algorithm, each of the k neurons of the hidden layer is calculated as a manipulation of the weighted sum of the input layer coefficients plus thresholds

$$\text{net}_i = \sum_{l=1}^{m} v_{li} I_l + \Theta_i \qquad i = 1, \ldots, k \tag{1}$$

Using the standard sigmoid activation function, H_i would be given by

$$H_i = f(\text{net}_i) = \frac{1}{1 + e^{-\text{net}_i}} \qquad i = 1, \ldots, k \tag{2}$$

Similarly, in the same forward path, for each of the n neurons of the output layer there results

$$\text{net}_j = \sum_{i=1}^{k} w_{ij} H_i + \Gamma_j \qquad j = 1, \ldots, n \tag{3}$$

$$\hat{y}_j = O_j = f(\text{net}_j) = \frac{1}{1 + e^{-\text{net}_j}} \qquad j = 1, \ldots, n \tag{4}$$

The goal of the extended BP algorithm is to enhance the capability of each neuron of both the hidden and output layers by introducing a modified activation function which would give for a neuron of the output layer

$$O_j = f(\text{net}_j, U_j, L_j, T_j) = \frac{U_j - L_j}{1 + e^{-\text{net}_j / T_j}} + L_j \qquad j = 1, \ldots, n \tag{5}$$

and for a neuron of the hidden layer

$$H_i = f(\text{net}_i, U_i, L_i, T_i) = \frac{U_i - L_i}{1 + e^{-\text{net}_i/T_i}} + L_i \qquad i = 1, \ldots, k \tag{6}$$

The parameters $U_{i,j}$ and $L_{i,j}$ represent the upper and lower bounds of the output of a generic neuron of the hidden and output layers, respectively. The temperatures $T_{i,j}$ represent the slope of the modified sigmoid function. Note that the original sigmoid activation function corresponds to the case of $U = 1$, $L = 0$, and $T = 1$. The gradient descent is calculated to perform the steepest descent as suggested in the standard BP algorithm. The only difference is that at this time the independent variables net, U, L, and T will be used. Therefore, for each neuron of the output layer, there results

$$f'_{\text{net}_j} = \frac{\partial f(\text{net}_j, U_j, L_j, T_j)}{\partial \text{net}_j} = -\frac{(O_j - U_j)(O_j - L_j)}{T_j(U_j - L_j)} \qquad j = 1, \ldots, n \tag{7}$$

$$f'_{U_j} = \frac{\partial f(\text{net}_j, U_j, L_j, T_j)}{\partial U_j} = \frac{1}{1 + e^{-\text{net}_j/T_j}} \qquad j = 1, \ldots, n \tag{8}$$

$$f'_{L_j} = \frac{\partial f(\text{net}_j, U_j, L_j, T_j)}{\partial L_j} = 1 - \frac{1}{1 + e^{-\text{net}_j/T_j}} = 1 - f'_{U_j} \qquad j = 1, \ldots, n \tag{9}$$

$$f'_{T_j} = \frac{\partial f(\text{net}_j, U_j, L_j, T_j)}{\partial T_j} = \frac{\text{net}_j(O_j - U_j)(O_j - L_j)}{T_j^2(U_j - L_j)} = -\frac{\text{net}_j f'_{\text{net}_j}}{T_j} \qquad j = 1, \ldots, n \tag{10}$$

Similarly, for each neuron of the hidden layer, there results

$$f'_{\text{net}_i} = \frac{\partial f(\text{net}_i, U_i, L_i, T_i)}{\partial \text{net}_i} = -\frac{(H_i - U_i)(H_i - L_i)}{T_i(U_i - L_i)} \qquad i = 1, \ldots, k \tag{11}$$

$$f'_{U_i} = \frac{\partial f(\text{net}_i, U_i, L_i, T_i)}{\partial U_i} = \frac{1}{1 + e^{-\text{net}_i/T_i}} \qquad i = 1, \ldots, k \tag{12}$$

$$f'_{L_i} = \frac{\partial f(\text{net}_i, U_i, L_i, T_i)}{\partial L_i} = 1 - \frac{1}{1 + e^{-\text{net}_i/T_i}} = 1 - f'_{U_i} \qquad i = 1, \ldots, k \tag{13}$$

$$f'_{T_i} = \frac{\partial f(\text{net}_i, U_i, L_i, T_i)}{\partial T_i} = \frac{\text{net}_i(H_i - U_i)(H_i - L_i)}{T_i^2(U_i - L_i)} = -\frac{\text{net}_i f'_{\text{net}_i}}{T_i} \qquad i = 1, \ldots, k \tag{14}$$

The output of each neuron of the output layer O_j is just the estimates of the measurements at each training step. The error signals δ with respect to the variables net, U, L, and T for the output units, which have specific desired targets Y_j, are given by

$$\delta_{\text{net}_j} = f'_{\text{net}_j}(Y_j - O_j) \qquad j = 1, \ldots, n \tag{15}$$

$$\delta_{U_j} = f'_{U_j}(Y_j - O_j) \qquad j = 1, \ldots, n \tag{16}$$

$$\delta_{L_j} = f'_{L_j}(Y_j - O_j) \qquad j = 1, \ldots, n \tag{17}$$

$$\delta_{T_j} = f'_{T_j}(Y_j - O_j) \qquad j = 1, \ldots, n \tag{18}$$

The neurons of the output layer are the only ones that have a desired value during the on-line learning. Since desired outputs are not available for the neurons of the hidden layer, the error signals for each neuron of the hidden layer are determined at each step in terms of the error signals of the units to which it directly connects and the weights of those connections. Therefore,

$$\delta_{\text{net}_i} = f'_{\text{net}_i} \sum_j^n \delta_{\text{net}_j} w_{ij} \qquad i = 1, \ldots, k \tag{19}$$

$$\delta_{U_i} = f'_{U_i} \sum_j^n \delta_{U_j} w_{ij} \qquad i = 1, \ldots, k \tag{20}$$

$$\delta_{L_i} = f'_{L_i} \sum_j^n \delta_{L_j} w_{ij} \qquad i = 1, \ldots, k \tag{21}$$

$$\delta_{T_i} = f'_{T_i} \sum_j^n \delta_{T_j} w_{ij} \qquad i = 1, \ldots, k \tag{22}$$

With the error signals calculated from the preceding equations in the backward-computing phase, each neuron of the hidden and output layers adjust its connection weights, thresholds, U, L, and T for the next time step by using

$$\Delta w_{ij}(K+1) = \eta_{\text{net}} \delta_{\text{net}_j} O_j + \alpha_{\text{net}} \Delta w_{ij}(K) \qquad j = 1, \ldots, n; \quad i = 1, \ldots, k \tag{23}$$

$$\Delta v_{li}(K+1) = \eta_{\text{net}} \delta_{\text{net}_i} H_i + \alpha_{\text{net}} \Delta v_{li}(K) \qquad l = 1, \ldots, m; \quad i = 1, \ldots, k \tag{24}$$

$$\Delta \Gamma_j(K+1) = \eta_{\text{net}} \delta_{\text{net}_j} + \alpha_{\text{net}} \Delta \Gamma_j(K) \qquad j = 1, \ldots, n \tag{25}$$

$$\Delta \Theta_i(K+1) = \eta_{\text{net}} \delta_{\text{net}_i} + \alpha_{\text{net}} \Delta \Theta_i(K) \qquad i = 1, \ldots, k \tag{26}$$

$$\Delta U_j(K+1) = \eta_{UL} \delta_{U_j} + \alpha_{UL} \Delta U_j(K) \qquad j = 1, \ldots, n \tag{27}$$

$$\Delta L_j(K+1) = \eta_{UL} \delta_{L_j} + \alpha_{UL} \Delta L_j(K) \qquad j = 1, \ldots, n \tag{28}$$

$$\Delta T_j(K+1) = \eta_T \delta_{T_j} + \alpha_T \Delta T_j(K) \qquad j = 1, \ldots, n \tag{29}$$

$$\Delta U_i(K+1) = \eta_{UL} \delta_{U_i} + \alpha_{UL} \Delta U_i(K) \qquad i = 1, \ldots, k \tag{30}$$

$$\Delta T_i(K+1) = \eta_T \delta_{T_i} + \alpha_T \Delta T_i(K) \qquad i = 1, \ldots, k \tag{31}$$

$$\Delta L_i(K+1) = \eta_{UL} \delta_{L_i} + \alpha_{UL} \Delta L_i(K) \qquad i = 1, \ldots, k \tag{32}$$

where η_{net}, η_{UL}, and η_T are the learning rates associated with the independent variables net, U, L, and T, respectively. As in the standard BP algorithm[10-12] the momentum terms α_{net}, α_{UL}, and α_T are also included; they determine the effects of past values in the adjustment equations just introduced to increase the learning speed. A flow chart of the extended back-propagation algorithm is shown in Table 1.

Numerical Simulation of the Failure Detection and Identification

The process described in the preceding section has been numerically simulated using the software recently distributed for the AIAA Control Design Challenge. This software provides a numerical nonlinear simulation of an aircraft model, with full-envelope nonlinear aerodynamics and full-envelope thrust with first-order engine response data.[19] The aircraft modeled is a high-performance, supersonic vehicle representative of current day fighters. The aircraft primary flight-control surfaces consist of horizontal stabilators which are capable of

Table 1 Extended back-propagation algorithm

```
Step 1  Assign random values to all weights and thresholds with
        $[-a,a]$. Initialize upper and lower bounds, temperatures:
        $U_{ij}=1$, $L_{ij}=0$, $T_{ij}=1$

        Do {

        For each time step
          Do {
Step 2  Set up the input $[Y(k-1)\ldots Y(k-p)]$ and desired output
        $Y(k)$
Step 3  /*Forward phase: from input to hidden layer*/
        Compute for each neuron $net_i$ and $H_i$ using Eqs. (1) and (6)
        /*Forward phase: from hidden to output layer*/
        Compute for each neuron $net_j$ and $O_j$ using Eqs. (3) and (5)
Step 4  /*Backward computing phase: output layer*/
        Compute errors $(\delta_j)$ using Eqs. (7-10) and (15-18)
        /*Backward computing phase: hidden layer*/
        Compute errors $(\delta_i)$ using Eqs. (11-14) and (19-22)
Step 5  /*Bounds updating phase: output layer*/
        If (desired $Y_j > U_j$) then $U_j$ = desired $Y_j$
          else use Eq. (27) to update $U_j$
        If (desired $Y_j < L_j$) then $L_j$ = desired $Y_j$
          else use Eq. (28) to update $L_i$
        Bounds updating phase: hidden layer*/
        If $(\delta_{U_i} > 0)$ the use Eq. (30) to update $U_i$
        If $(\delta_{L_i} < 0)$ then use Eq. (32) to update $L_i$
Step 6  /*Weights, thresholds, and temperatures updating phase*/
        Use Eqs. (23) and (25) to update $w_{ij}$ and $\Gamma_j$
        Use Eqs. (24) and (26) to update $v_{li}$ and $\Theta_i$

        Use Eq. (29) to update $T_j$
        Use Eq. (31) to update $T_i$

          }
Step 7  /*Error computing phase*/
        Compute estimation error ERRTOT

        }while ERRTOT < desired threshold (or keep learning)

        Stop
```

Table 2 Aircraft data

Mass and geometry characteristics

Parameter	Symbol	Value
Wing area, ft^2	S	608.0
Wing span, ft	b	42.8
Mean aerodynamic chord, ft	mac	15.95
Aircraft weight, lb	W	45,000.0
Moments of inertia		
Roll, slug ft^2	I_x	28,700.0
Pitch, slug ft^2	I_y	165,100.00
Yaw, slug ft^2	I_z	187,900.0
Products of inertia, slug ft^2	I_{xz}	−520.0
	I_{xy}	0.0
	I_{yz}	0.0

Command input limits and sign conventions

Surface	Symbols	Limits, deg	Positive sign convention
Aileron	δ_A	± 20	Left trailing-edge down
Symmetric stabilator	δ_H	$+15 \leftrightarrow -25$	Trailing-edge down
Differential stabilator	δ_D	± 20	Left trailing-edge down
Rudder	δ_R	± 30	Trailing-edge left

Surface deflection equations

Surface	Deflection definition	Rate limit, deg/s
$\delta_{A\text{left}}$	$\delta_A/2$	24
$\delta_{A\text{right}}$	$-\delta_A/2$	24
$\delta_{H\text{left}}$	$(2\delta_H-\delta_D)/2$	24

symmetric or differential movement, conventional ailerons, and a single vertical rudder. There are a total of five actuators: two aileron, two stabilator, and one rudder. The model includes identical actuators for all of the surfaces. These actuators are limited at 24 deg/s. The stabilator command block diagram is shown in Fig. 2. The mass and geometry characteristics along with information on the control surfaces are given in Table 2.

The aerodynamics are modeled for the full aircraft envelope using multidimensional tables and linear interpolation to form nonlinear function generators. The software does not allow the user to input a pilot maneuver; it only allows to trim the aircraft at a specified flight condition starting from another specified initial condition. The full control autopilot will consider the damage just as an ordinary atmospheric turbulence and will try to bring the aircraft back to an equilibrium condition. No efforts were made in this study to implement a reconfiguration logic in the autopilot control laws because the FDI problem is the main object of this investigation.

To have a realistic simulation it was necessary to model the aerodynamic effects of the damage. For this purpose, it may be said that the changed aircraft dynamics following damage 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. Additionally, there is a damage-induced rolling moment. Using the procedures introduced in Ref. 20 it is possible to implement a set of closed-form expressions of the nondimensional stability and control derivatives as functions of the normal force coefficient of the damaged stabilator. A similar approach could be used for simulating damage to other control surfaces. The closed-form expressions of these derivatives for the longitudinal dynamics, at both nominal and damaged conditions, are given by[2,20]:

$$c_{L_\alpha} = a_1 + b_1 c_{L_{\delta_H}}; \qquad (c_{L_\alpha})_{\text{dam}} = a_1 + b_1 (c_{L_{\delta_H}})_{\text{dam}} \tag{33}$$

$$c_{m_\alpha} = a_2 + b_2 c_{L_{\delta_H}}; \qquad (c_{m_\alpha})_{\text{dam}} = a_2 + b_2 (c_{L_{\delta_H}})_{\text{dam}} \tag{34}$$

$$c_{L_{\dot\alpha}} = a_3 + b_3 c_{L_{\delta_H}}; \qquad (c_{L_{\dot\alpha}})_{\text{dam}} = a_3 + b_3 (c_{L_{\delta_H}})_{\text{dam}} \tag{35}$$

$$c_{m_{\dot\alpha}} = a_4 + b_4 c_{L_{\delta_H}}; \qquad (c_{m_{\dot\alpha}})_{\text{dam}} = a_4 + b_4 (c_{L_{\delta_H}})_{\text{dam}} \tag{36}$$

$$c_{L_q} = a_5 + b_5 c_{L_{\delta_H}}; \qquad (c_{L_q})_{\text{dam}} = a_5 + b_5 (c_{L_{\delta_H}})_{\text{dam}} \tag{37}$$

$$c_{m_q} = a_6 + b_6 c_{L_{\delta_H}}; \qquad (c_{m_q})_{\text{dam}} = a_6 + b_6 (c_{L_{\delta_H}})_{\text{dam}} \tag{38}$$

where a_i and b_i $(i = 1, \ldots, 6)$ are constant coefficients related to the aerodynamic modeling with

$$c_{L_{\delta_H}} = c_{L_{\delta_{HR}}} + c_{L_{\delta_{HL}}}; \qquad (c_{L_{\delta_H}})_{\text{dam}} = c_{L_{\delta_{HR}}} + (c_{L_{\delta_{HL}}})_{\text{dam}} \tag{39}$$

$$|(c_{L_{\delta_H}})_{\text{dam}}| < |c_{L_{\delta_H}}|; \qquad |(c_{L_{\delta_{HL}}})_{\text{dam}}| < |c_{L_{\delta_{HL}}}| \tag{40}$$

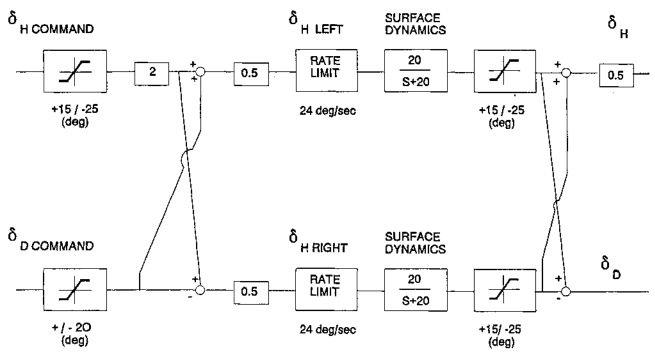


Fig. 2 Block diagram of the stabilator command.

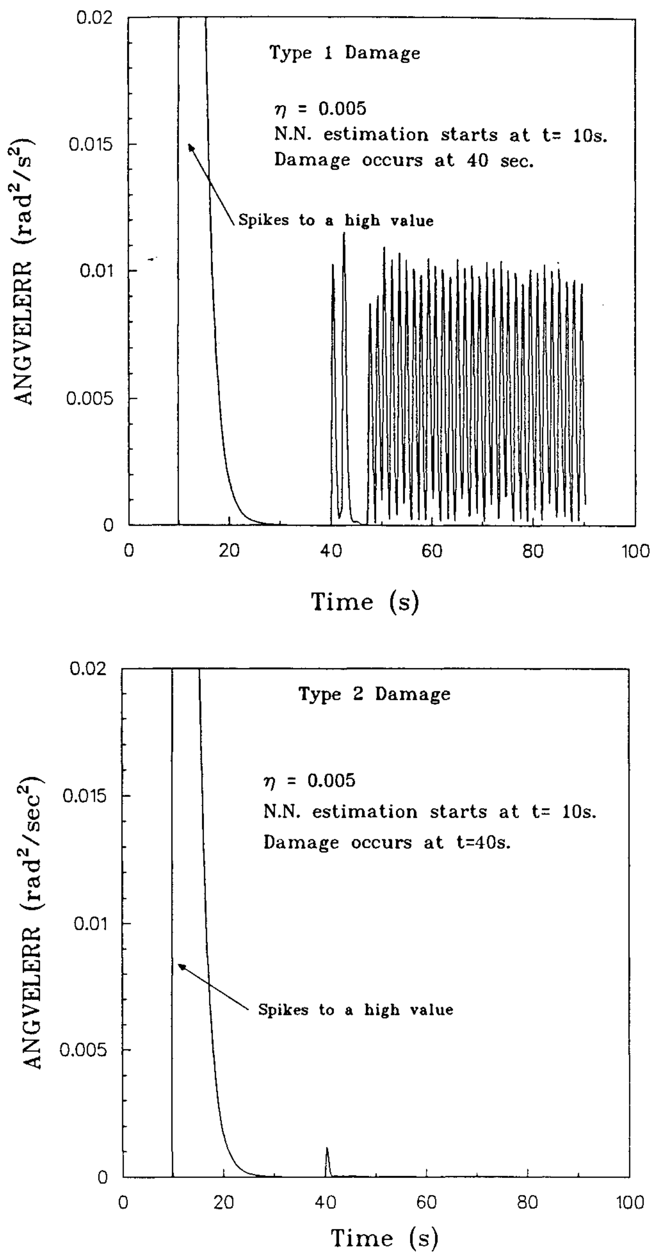


Fig. 3 ANGVELERR vs time for types 1 and 2 damage with learning rates = 0.005.

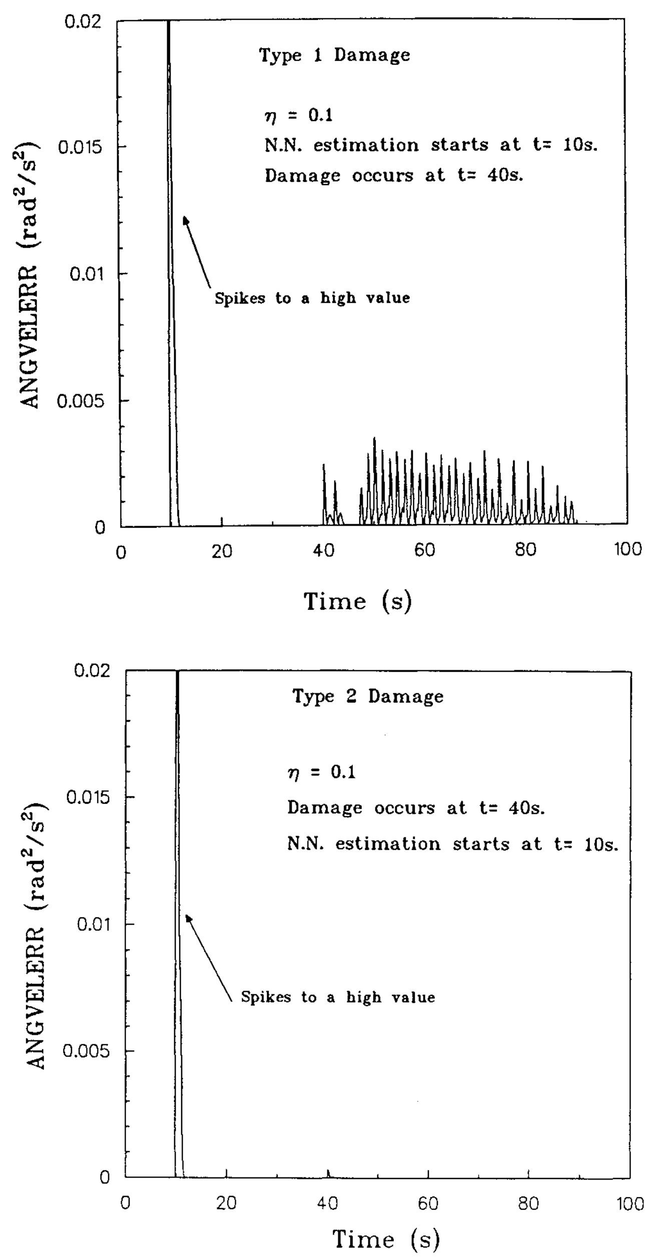


Fig. 4 ANGVELERR vs time for types 1 and 2 damage with learning rates = 0.1.

The damaged-induced rolling moment was introduced by multiplying the stabilator normal force coefficient, considered to be acting at the stabilator mean aerodynamic chord, with the distance from the aircraft x axis divided by the wing span for adimensionalization purposes. The damage was aerodynamically modeled in the code with modifications of the coefficient of lift, pitching moment, and rolling moment.

It must be emphasized that this mathematical model of the damaged aircraft, based on the preceding assumptions on the aerodynamic behavior of the damaged surface, is essentially linear and does not claim to be extremely accurate. It has only been introduced for the purpose of simulating the FD and it is not intended to be used by the flight control system for any reconfiguration purposes.

The time increment Δt of the simulation was selected to be 0.01. The values of the data of the measurement vector M

$$M = [\alpha, q, \theta, p, r, a_z, a_y, SC_1, SC_2, SC_3]^T \tag{41}$$

from time instant $K - 1$ down to $K - p$ ($p = 4$) were given as input to the three-layer NN of Fig. 1, which is continuously training on line. The hidden layer of the NN is assumed to have $k = 10$ neurons. The output of the NN is given by the $n = 7$ size vector O.

$$O = [\hat{\alpha}, \hat{q}, \hat{\theta}, \hat{p}, \hat{r}, \hat{a}_z, \hat{a}_y]^T \tag{42}$$

At each instant K the sensors data grouped in the vector Y

$$Y = [\alpha, q, \theta, p, r, a_z, a_y]^T \tag{43}$$

are compared with the estimates of the NN, the vector O.

The simulation starts at an altitude of 19,700 ft with a speed of 670 ft/s. The desired trimming conditions are 20,000-ft altitude with a 700-ft/s speed. The motion of the aircraft at the nominal configuration starts at time instant $K = 1$ without the on-line state estimation. At time instant $K = K_{\text{init}}$ ($t = K_{\text{init}}$ $\Delta t =$ 10 s) the NN is turned on and, after a training process, should be able to provide accurate state estimation of the aircraft dynamics. To avoid false alarms, it will be desirable to automatically disengage the FD every time a pilot voluntarily changes the aerodynamic configuration, as in the case for

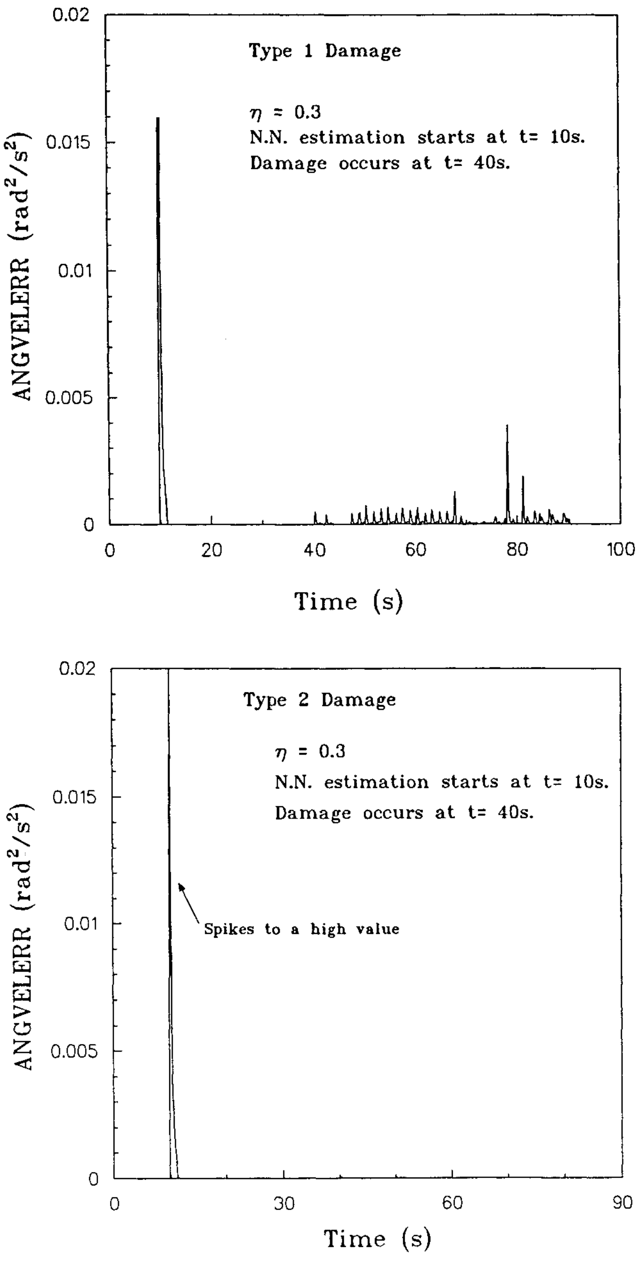


Fig. 5 ANGVELERR vs time for types 1 and 2 damage with learning rates = 0.3.

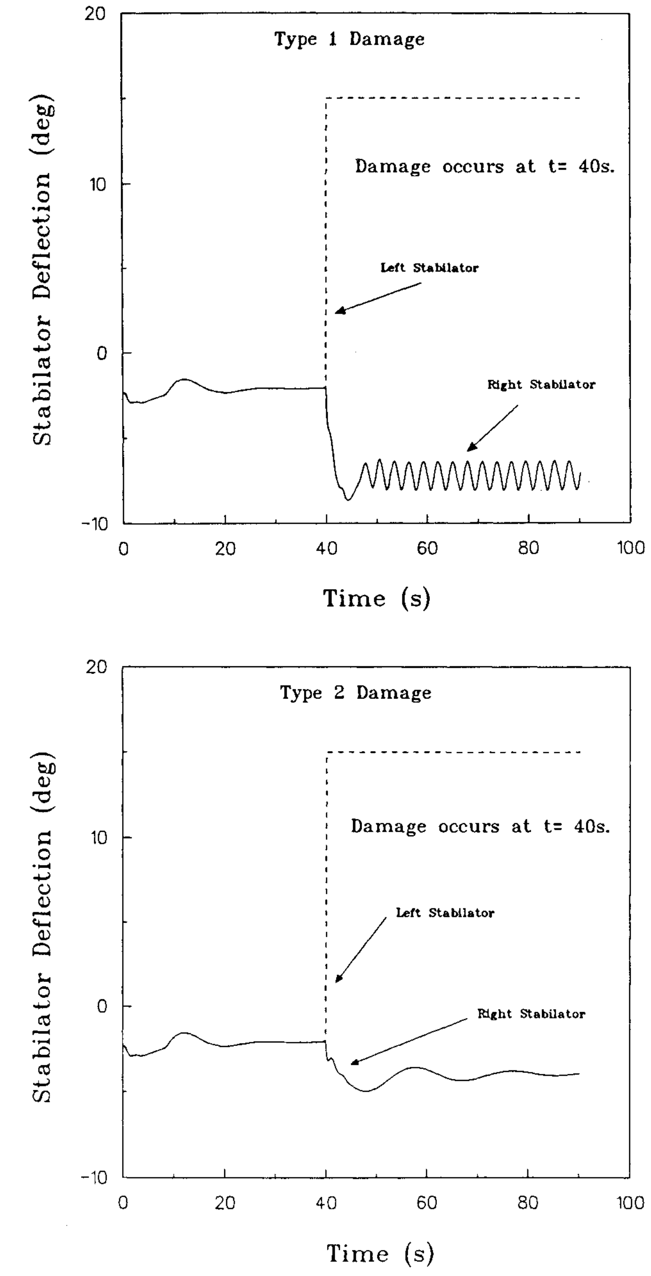


Fig. 6 Left and right stabilator time histories.

variable wing geometry or for dropping empty fuel tanks or war loads. At a certain time instant $K = K_{\text{dam}}$ ($t = K_{\text{dam}} \Delta t = 40$ s) the damage takes place on the left stabilator, causing an instantaneous change of the lift and pitching and rolling moment coefficients along with a stuck value of δ_{HL}. For this study, two damage types were considered:

1) As a result of damage, the left stabilator loses 50% of its effectiveness and it is stuck at $\delta_{HL} = 15$ deg.

2) As a result of damage, the left stabilator loses 25% of its effectiveness and it is stuck at $\delta_{HL} = 15$ deg.

The autopilot will treat the damage just as an ordinary perturbation of the motion and will try to restore trimming conditions. Following the first damage type the aircraft will be in a marginally stable condition; the autopilot will not be able to bring back the aircraft to an equilibrium condition. Following the second damage type the aircraft will still be stable and the autopilot will have enough control authority to bring back the aircraft to an equilibrium condition. In both cases the aircraft dynamics have changed immediately after damage; this implies that an angular velocity estimation error parameter ANGVELERR, defined by

$$\text{ANGVELERR}(k) = \tfrac{1}{2}\{[p(k) - \hat{p}(k)]^2 + [q(k) - \hat{q}(k)]^2 + [r(k) - \hat{r}(k)]^2\} \quad (44)$$

will become bigger than a selected threshold and this is a quick indication that a failure has indeed occurred.

A small-scale parametric study was conducted for both damage types to investigate the effects of different values for the learning rates (η_{net}, η_{UL}, η_T). The considered values are reported in Table 3.

The results of this study in terms of the trend of the parameter ANGVELERR are shown in Figs. 3–5 for both damage types. In each of the plots there is a peak as soon as the NN estimation is turned on. All of the plots have the same upper limit for ANGVELERR to allow a comparison of the trends for different learning rates. It can be noticed that the parameter ANGVELERR has higher peaks after damage for the first damage type, as one would expect. Also, it was noticed that

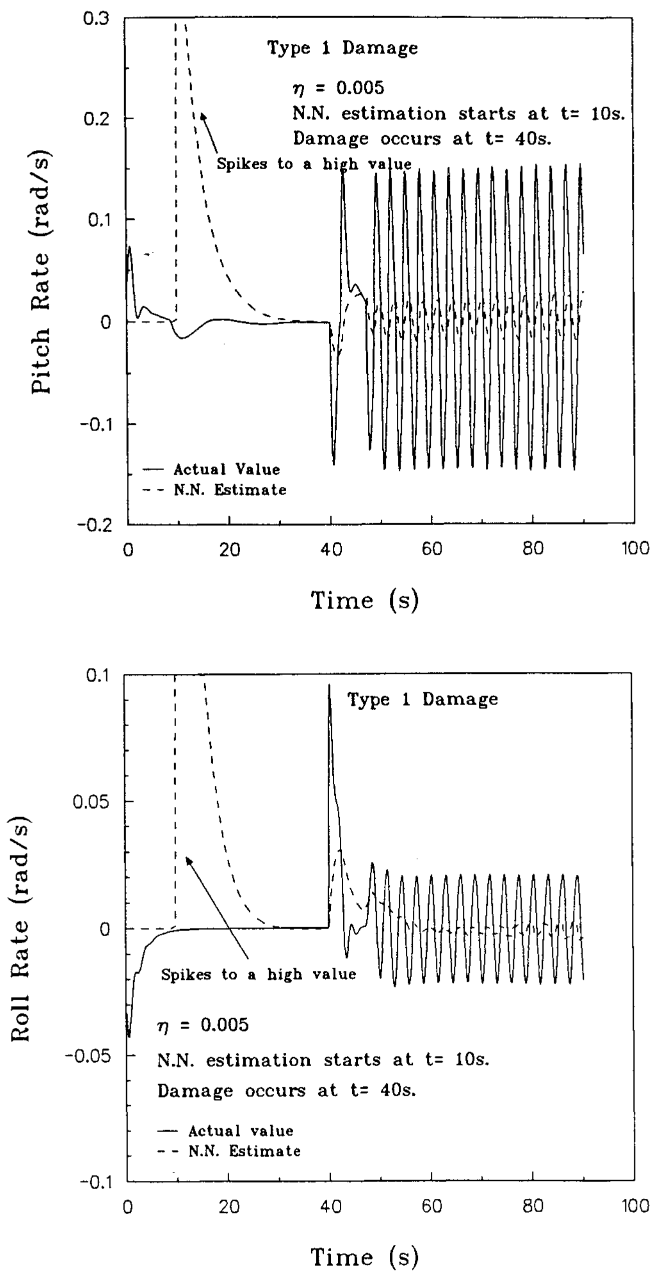


Fig. 7 Actual and estimated time histories for type 1 damage.

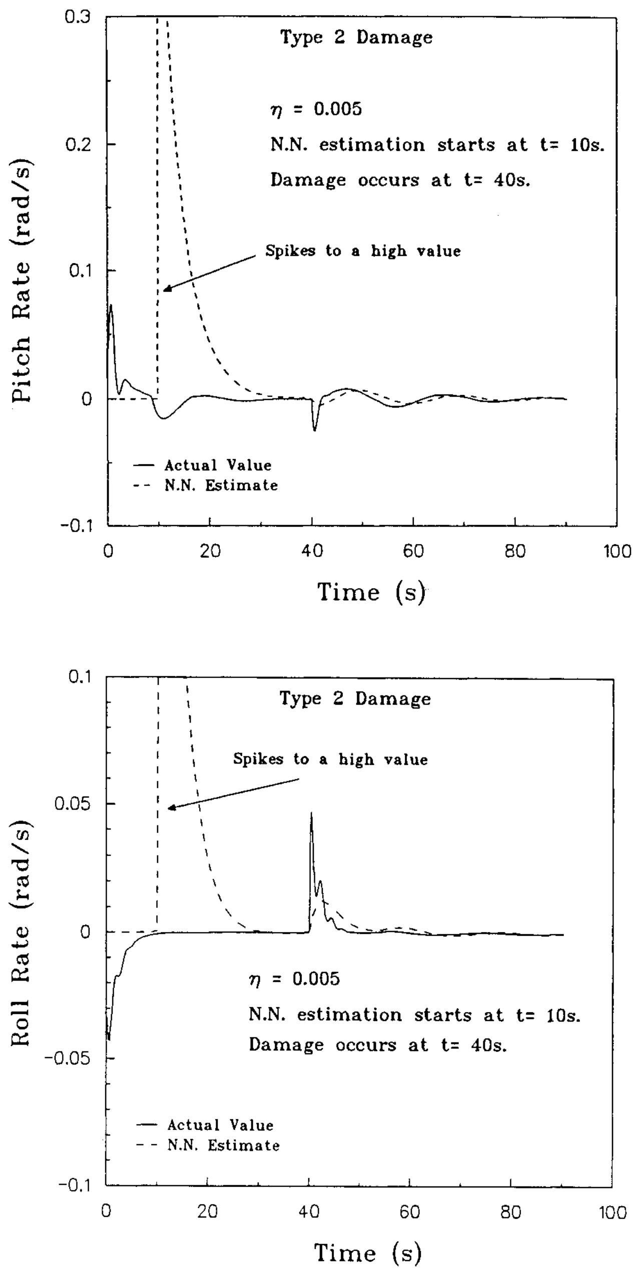


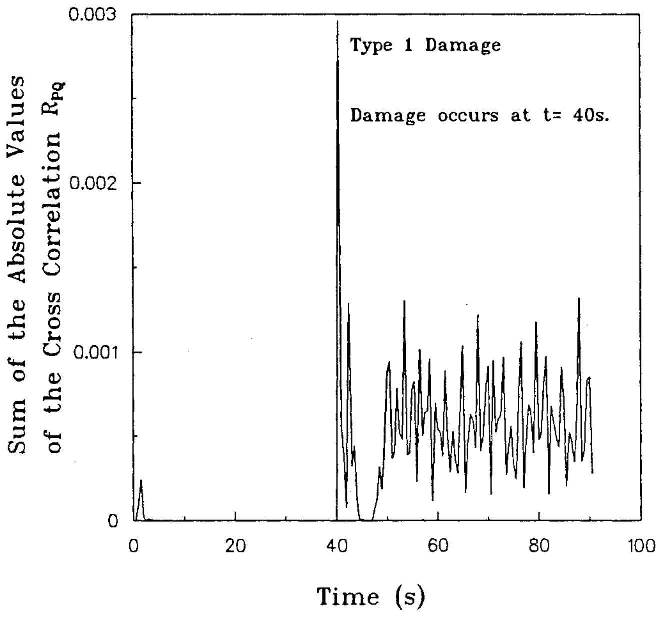
Fig. 8 Actual and estimated time histories for type 2 damage.

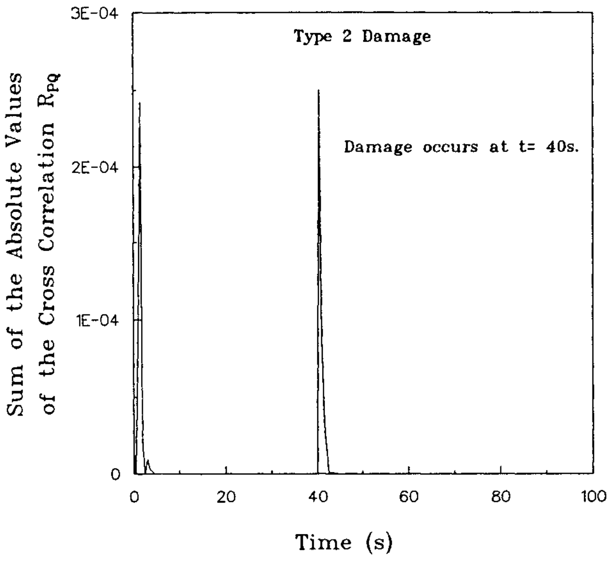
greater learning rates increase the speed and the accuracy of the estimates implying a decrease of the parameter ANGVELERR. For FD purposes a lower learning rate will guarantee a quick detection. The magnitude of the threshold can be the object of discussion. It is desirable to minimize the level of false alarms due to system noise and still be able to detect very quickly significant abnormalities. No efforts were made in this study to evaluate separately the effects of upper and lower bounds and temperature learning rates and momentums (ηs and αs). The time histories of the stabilator for both cases are shown in Fig. 6. It is clear that for the first damage type a greater activity of the "healthy" right stabilator is required. Figures 7 and 8 show the time histories for the real and estimated values of q and p with a 0.005 learning rate for both damage types. The marginal stability condition following the first type of damage is shown clearly in Fig. 7.

To this point, by analyzing the parameter ANGVELERR, the failure detection has been accomplished. However, it is also desirable to be able to interpret the sensors data in the shortest time possible with the goal of identifying the location of the damage. An approach to the failure identification could be given by an analysis of the cross-correlation functions between some key state variables. For example, it is known that ordinarily the values of the pitch and the roll rates are not correlated. However, Figs. 7 and 8 have shown that, immediately following the damage to the left stabilator, a strong rolling moment will take place. Therefore, there will be a certain cross correlation between the pitch and the roll rates. By definition, given two generic random processes $Y(k)$ and $X(k)$, the cross-correlation function is given by[21]

$$R_{YX}(n) = E[Y(k)\,X(k+n)] \tag{45}$$

where E is the expectation operator. To use such a concept for damage identification purposes, not knowing, of course, when the damage is going to occur, the cross-correlation function R_{PQ} should be calculated at each time instant K and stored in temporary memory locations of the flight computer for a time





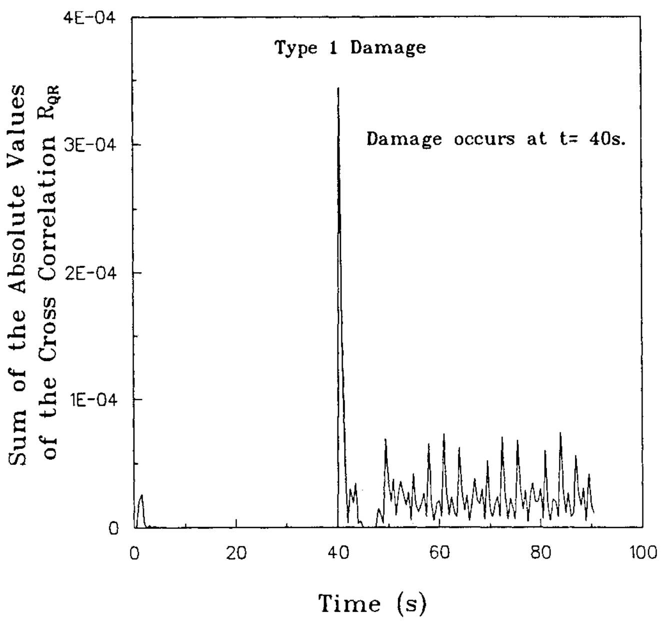


Fig. 9 Sum of the absolute values of the elements of cross-correlation functions vs time for type 1 damage.

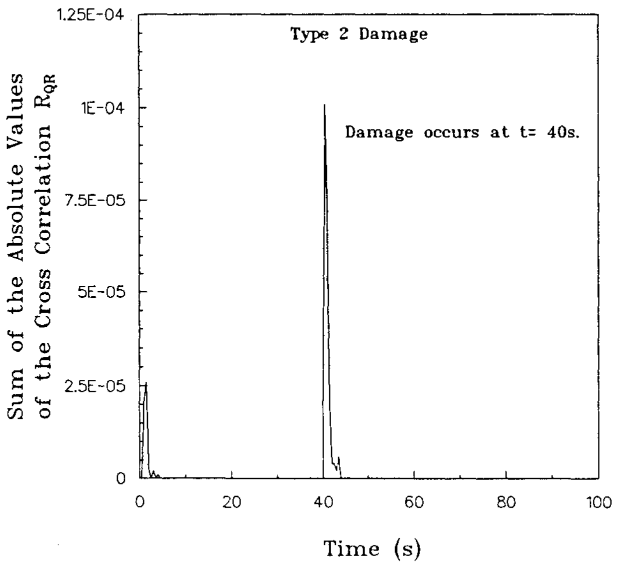


Fig. 10 Sum of the absolute values of the elements of cross-correlation functions vs time for type 2 damage.

window of size win. If no abnormal flight conditions due to the damage are detected, the flight computer will keep on calculating these cross-correlation functions and storing them in temporary memory locations. If a failure is detected (that is, ANGVELERR > $\text{thresh}_{\text{FD}}$) the flight computer will then proceed to analyze the cross-correlation functions data. Particularly, the sum of the absolute values of R_{PQ} needs to be calculated on line at each time instant. If it is observed that the values of these sums are increasing, then the damaged control surface may be identified. In the case of a damage on the left elevator for $K > K_{\text{dam}}$, at which already (ANGVELERR > $\text{thresh}_{\text{FD}}$), we will have

$$\sum [\text{abs}(R_{PQ}\text{coef})](k-1) - \sum [\text{abs}(R_{PQ}\text{coef})](k-2) > \text{thresh}_{\text{FI}(PQ:\text{elev})} \tag{46}$$

The same approach could be used for FDI of damages to other control surfaces; in this case other cross-correlation functions should be analyzed, such as R_{PR} and R_{QR}. A suitable decision process is shown in Table 4. The selection of the values of all of the thresholds can be performed within a flight simulator or with a dynamic simulation program. Even these threshold values will have to be tradeoffs between the desire for a quick detectability and identifiability of the damage and the desire to minimize the rate of false alarms. Note that comparative studies must be performed with the same window's size, the same number of points for each cross-correlation function, and with the same sampling time. It should be pointed out that different values of the same cross-correlation function are required depending on which surface has been damaged. The sums of the absolute values of the elements of R_{PQ} and R_{QR} for both damage types are shown in Figs. 9 and 10 for a time window of 500 instants.

The initial small peaks of the sum of the absolute values of the cross-correlations are because at the beginning not all of the required data are available for the calculation of the cross correlations and the algorithm assigns zero values by default. It can be noticed that immediately after damage there is a sharp increase in the sums of the absolute values of R_{PQ} and

Table 3 Values of learning rates and momentums

Case 1:	$\eta_{\text{net}} = \eta_{UL} = \eta_T = 0.005$
	$\alpha_{\text{net}} = \alpha_{UL} = \alpha_T = 0.0$
Case 2:	$\eta_{\text{net}} = \eta_{UL} = \eta_T = 0.1$
	$\alpha_{\text{net}} = \alpha_{UL} = \alpha_T = 0.0$
Case 3:	$\eta_{\text{net}} = \eta_{UL} = \eta_T = 0.3$
	$\alpha_{\text{net}} = \alpha_{UL} = \alpha_T = 0.0$

Table 4 Rules for the failure detection and identification process

```
If {ANGVELERR($k$) > thresh$_{\text{FD}}$}
  If {sum(abs($R_{PQ}$ coef.)))($k-1$) − sum(abs($R_{PQ}$ coef.)))($k-2$)
      > thresh$_{\text{FI}(PQ:\ \text{STABIL.})}$}
    DAMAGE TO THE STABILATOR
  end

  elseif {sum(abs($R_{PR}$ coef.)))($k-1$) − sum(abs($R_{PR}$ coef.)))($k-2$)
      > thresh$_{\text{FI}(PR:\text{RUDDER})}$}
    DAMAGE TO THE RUDDER
  elseif {{sum(abs($R_{PR}$ coef.)))($k-1$) − sum(abs($R_{PR}$ coef.)))($k-2$)
      > thresh$_{\text{FI}(PR:\text{AILER.})}$}
  and
    {sum(abs($R_{PQ}$ coef.)))($k-1$) − sum(abs($R_{PQ}$ coef.)))($k-2$)
      > thresh$_{\text{FI}(PQ:\text{AILER.})}$}}
    DAMAGE TO THE AILERONS
    else
      False Alarm
    end
  end
end
```

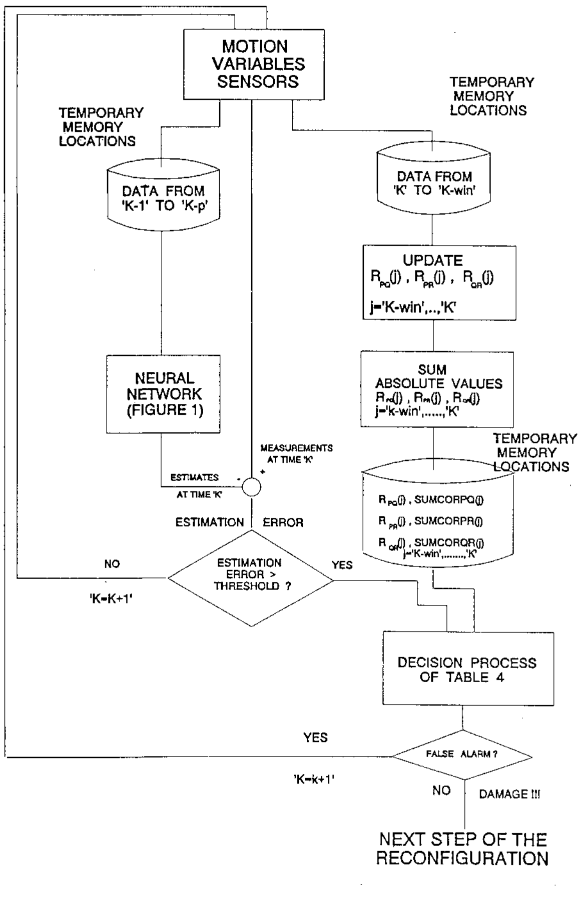

MOTION VARIABLES SENSORS
TEMPORARY MEMORY LOCATIONS
DATA FROM 'K-1' TO 'K-p'
TEMPORARY MEMORY LOCATIONS
DATA FROM 'K' TO 'K-win'
UPDATE $R_{PQ}(j)$, $R_{PR}(j)$, $R_{QR}(j)$ j='K-win',..,'K'
SUM ABSOLUTE VALUES $R_{PQ}(j)$, $R_{PR}(j)$, $R_{QR}(j)$ j='k-win',......,'K'
NEURAL NETWORK (FIGURE 1)
MEASUREMENTS AT TIME 'K'
ESTIMATES AT TIME 'K'
-
+
ESTIMATION ERROR
TEMPORARY MEMORY LOCATIONS
$R_{PQ}(j)$, SUMCORPQ(j)
$R_{PR}(j)$, SUMCORPR(j)
$R_{QR}(j)$, SUMCORQR(j) j='K-win',........,'K'
NO
'K=K+1'
ESTIMATION ERROR > THRESHOLD ?
YES
DECISION PROCESS OF TABLE 4
YES
'K=k+1'
FALSE ALARM ?
NO
DAMAGE !!!
NEXT STEP OF THE RECONFIGURATION

Fig. 11 Block diagram of the failure detection and identification process.

R_{QR} for both damage types. However, the coupling remains high for the first damage type where the aircraft is marginally stable. For the second damage type the coupling is removed as the aircraft is brought back to an equilibrium condition by the autopilot. The block diagram of the overall FDI process is shown in Fig. 11.

Once the failure detection and identification process has been satisfactorily performed, the next step would be to proceed to the remaining tasks of the flight control reconfiguration process. In the same time, given that the instants immediately following a damage are moments of concern and sometimes panic even for a well-trained pilot, it will be desirable to briefly inform the pilot about the location and the type of the damage with an emergency message displayed on the main screen of the cockpit.

Conclusions

This paper has presented an original implementation of failure detection and identification in the flight control system via neural network and analysis of cross-correlation functions. The numerical study presented in this paper has shown a quick failure detection and identification of a typical damage to the left stabilator of a high-performance aircraft involving a missing surface and a stuck actuator. The neural network, implemented in the flight computer software and trained on line using motion variables sensors data, has shown to be an effective state estimator. Its capability in predicting the dynamics of the aircraft make it an attractive alternative to classic estimation structures such as Kalman filters and observers, of both full and reduced order. The relative simplicity of the algorithms and the efficiency of the method may be the key factors for a successful practical implementation of this approach.

References

[1]Napolitano, M. R., "A New Approach to the Flight Control System Reconfiguration Following a Battle Damage and/or a Generic Failure on a Control Surface," Ph.D. Thesis, School of Mechanical and Aerospace Engineering, Oklahoma State Univ., Stillwater, OK, Dec. 1989.

[2]Napolitano, M. R., and Swaim, R. L., "New Technique for Aircraft Flight Control Reconfiguration," *Journal of Guidance, Control, and Dynamics,* Vol. 14, No. 1, 1991, pp. 184–190.

[3]Robinson, A. C., "Totally Robust Control—A New Concept for Design of Flight Control Systems," *Proceedings of the AIAA Guidance, Navigation and Control Conference* (Snowmass, CO), AIAA, New York, 1985 (AIAA Paper 85-1974).

[4]Kerr, T., "False Alarm and Correct Detection Probabilities Over a Time Interval for Restricted Classes of Failure Detection Algorithms," *IEEE Transactions on Information Theory,* Vol. IT-20, No. 4, 1982, pp. 619–631.

[5]Willsky, A. S., "A Survey of Several Failure Detection Methods," *Automatica,* Vol. 12, No. 6, 1976, pp. 601–611.

[6]Willsky, A. S., and Jones, H. L., "A Generalized Likelihood Ratio Approach to the Detection and Estimation of Jumps in Linear Systems," *IEEE Transactions on Automatic Control,* Vol. AC-21, Feb. 1976, pp. 108–112.

[7]Willsky, A. S., "Failure Detection in Dynamic Systems," AGARD LS-109, Neuilly sur Seine, France, Oct. 1980.

[8]Massoumnia, M. A., Verghese, G. C., and Willsky, A. S., "Failure Detection and Identification," *IEEE Transactions on Automatic Control,* Vol. AC-34, No. 3, 1989.

[9]Widrow, B., and Stearns, S., *Adaptive Signal Processing,* MIT Press, Cambridge, MA, 1986.

[10]Rumelhart, D., and McClelland, J., *Parallel Distributed Processing,* MIT Press, Cambridge, MA, 1986.

[11]Simpson, P. K., *Artificial Neural System,* Pergamon, Fairview Park, NY, 1990.

[12]Nielsen, R. B., *Neurocomputing,* Addison-Wesley, Reading, MA, 1990.

[13]Parlos, A. G., Atiya, A. F., and Sunkel, J. W., "Parameter Estimation in Space Systems Using Recurrent Neural Networks," *Proceedings of the AIAA Guidance, Navigation, and Control Conference* (New Orleans, LA), AIAA, Washington, DC, Aug. 1991 (AIAA Paper 91-2716).

[14]Troudet, T., Garg, S., and Merrill, W. C., "Neural Network

Application to Aircraft Control System Design," *Proceedings of the AIAA Guidance, Navigation, and Control Conference* (New Orleans, LA), AIAA, Washington, DC, Aug. 1991 (AIAA Paper 91-2715).

[15]Ha, C. M., "Neural Network Approach to AIAA Aircraft Control Design Challenge," *Proceedings of the AIAA Guidance, Navigation, and Control Conference* (New Orleans, LA), AIAA, Washington, DC, Aug. 1991 (AIAA Paper 91-2672).

[16]Napolitano, M. R., Chen, C. I., and Nutter, R., "Application of a Neural Observer as State Estimator in Active Vibration Control of a Cantilevered Beam," *IOP International Journal on Smart Materials and Structures,* Vol. 1, April 1992, pp. 69–75.

[17]Ogata, K., *Discrete-Time Control Systems,* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[18]Chen, C. L., and Nutter R. S., "An Extended Back-Propagation Learning Algorithm by Using Heterogenous Processing Units," *Proceedings of the International Joint Conference on Neural Network (IJCNN '92),* Baltimore, MD, June 1992.

[19]Brumbaugh R. W., "An Aircraft Model for the AIAA Controls Design Challenge," NASA CR 186019, Dec. 1991.

[20]Hoak, D. E., Ellison, D. E., et al., "USAF Stability and Control Datcom," Flight Control Division, Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, OH, 1960.

[21]Newland, D. E., *Random Vibrations and Spectral Analysis,* Longman Group, Essex, England, UK, 1984.