THE FOUR PARAMETER CONTROLLER A NEURAL NETWORK IMPLEMENTATION

IEEE Meditteranean Symposium on new Directions in Control Theory and Applications
Chania, GREECE
June 21-23, 1993

Ioannis K. Konstantopoulos and Panos J. Antsaklis
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556
email: antsakli@saturn.ece.nd.edu

THE FOUR PARAMETER CONTROLLER A NEURAL NETWORK IMPLEMENTATION

Ioannis K. Konstantopoulos, and Panos J. Antsaklis
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556, U.S.A.

June 21, 1993

Abstract

Controllers capable to perform failure diagnosis have an additional diagnostic output to detect and isolate sensor and actuator faults. A linear such controller is usually called a four-parameter controller. Neural networks have proved to be a very powerful tool in the control systems area, where they have been used in the modelling and control of dynamical systems. In this paper, a neural network model of a controller with diagnostic capabilities (CDC) is presented for the first time. This nonlinear neural controller is trained to operate as a traditional controller, while at the same time it provides a reproduction of the failure occuring either at the actuator or the sensor. The cases of actuator and sensor failure are studied independently. The validity of the results is verified by extensive simulations.

1 INTRODUCTION

The four-parameter controller is a linear controller used for both control and diagnostics; it was first introduced in [10]. This controller is a generalization of the familiar two-parameter linear controller. It has an additional output that can be viewed as a diagnostic output, which is monitored to detect and isolate sensor and actuator faults. Therefore, beside conventional specifications such as reference input tracking, the controller must also provide information about possible failures that occur. Combining control and diagnostic tasks in a single device recognizes the fact that the control law used affects the diagnostic capabilities of our system, as these tasks compete against each other in general.

Neural networks have proved to be a very powerful tool in the control systems area, where they have been used in the modelling and control of dynamical systems. A neural network model of the controller with diagnostic capabilities (CDC) is implemented here for the first time. This paper is based on work that was reported in [7]. Note that this neural network implementation is applicable to both linear and nonlinear systems, thus extending the applicability of the concept of the four-parameter controller.

It is quite common in the failure diagnosis literature to establish an upper limit-bound, so that a failure is announced, when this limit is exceeded. Declaring a failure when the limiting value of the monitored quantity is exceeded has certain flaws concerning the way these limits are established. The major issue is the reliability of the final decision, since the establishment of such a limit specification may result in an undeclared failure or in the announcement of a failure that is due to noise rather than to a real fault. The focus of this paper is on reproducing the failure; that is the controller is trained, so that the failure that occured will appear in its diagnostic output. The idea of reproducing the failure appears attractive, because it enables us to use a more sophisticated method than a single threshold, in order to detect a failure.

Note that here the case of actuator and the case of sensor failures are considered. The two cases are considered separately, that is for each the assumption is made that only one kind of failure could occur. Another aspect investigated is the establishment of priorities between the control and diagnostic objectives. Since the CDC controller has both a control and diagnostic task, we are able to specify the task that we are more interested in and train our controller accordingly. The performance of the controller changes according to the priorities established. Therefore for applications, where the real concern is the recognition of failures, the controller can be trained with emphasis on the diagnostic performance. It is shown via examples that this can be done, while the control performance is maintained at satisfactory levels.

The paper is organized as follows. In section 2 a brief outline of the original linear CDC, the four-parameter controller, is presented. Problems that can arise due to the interaction of controls and diagnostics are discussed and previous research work on the four parameter-controller and its applications is also presented. A brief description of neural networks fundamentals is then introduced and a discussion of the most common architecture, the feedforward multilayer neural networks is given. In addition, the back-propagation algorithm and the applications of neural networks in the control of systems are briefly discussed. In sections 3 and 4 a feedforward multilayer neural network is used to implement the controller with diagnostic capabilities. Two cases are investigated: actuator failures (section 3), and sensor failures (section 4). It is shown that this implementation can clearly reproduce the failures introduced in the system. Also, methods to emphasize either the control or the diagnostic aspects of the CDC controller are presented. In section 5 the contributions of this paper are discusssed and a critical evaluation of the suggested implementation is given, where the specific advantages are clearly stated.

2 PRELIMINARIES AND PROBLEM FORMULATION

2.1 The Four-Parameter Controller

The four-parameter controller, which is illustrated in Fig. 1, is a generalization of the familiar two-parameter linear controller, [15]. This controller has two vector inputs and two vector outputs, resulting to a controller with four matrix parameters. The concept of the four-parameter controller is attributed to C. Nett, who also proposed the general control system configuration of Fig. 2.

The signals appearing in Fig. 2 represent : r is the command or the reference input to the controller, that is an independent controller variable that is not manipulated by the plant; a is the diagnostic output, that is the output of the controller that is designed to reproduce the failures and is not manipulated by the plant; y_c is the controller output that is manipulated by the plant and should be considered as the ideal actuator input; u_c is the manipulated controller input, that is the independent controller variables that are manipulated by the plant; n_a is an exogenous input which accounts for unmodeled signals that inevitably appear at the actuator; n_s is an exogenous input which accounts for unmodeled signals that inevitably appear at the sensor; u is the actual actuator output. that is the actual signal that is manipulated by the plant; y is the utilized plant output that

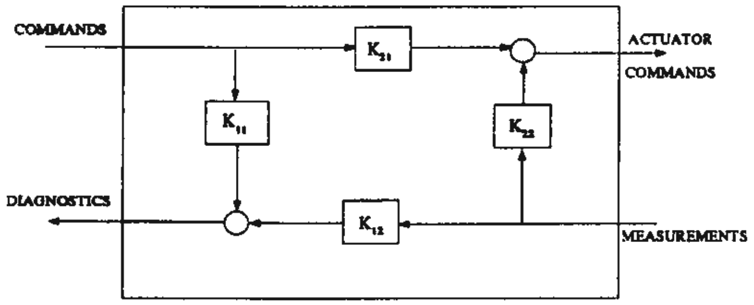

Figure 1: The structure of the four-parameter controller

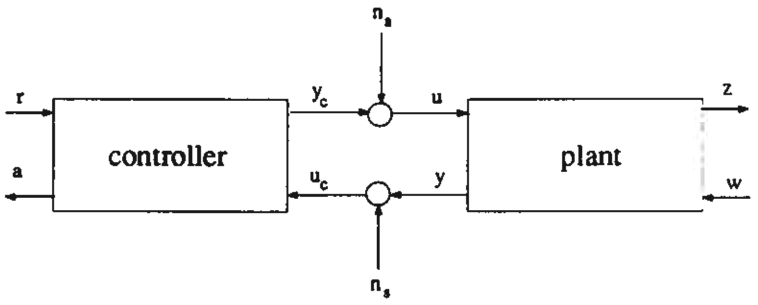

Figure 2: General control system configuration

should be considered as the ideal sensor input; z is the unutilized plant output, that is the plant variables that are not used by the controller, and w is the unmanipulated plant input, that is the disturbance that can't obviously be manipulated by the controller.

In view of Fig. 1, Fig. 2, the linear controller can be described by the following relation :

$$\begin{pmatrix} a \\ y_c \end{pmatrix} = \begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix} \cdot \begin{pmatrix} r \\ u_c \end{pmatrix} \tag{1}$$

It is for these matrix parameters that the controller has been named four parameter controller. It is obvious that for $K_{11} = K_{12} = 0$, which implies that $a = 0$, the controller of (1) reduces to the familiar two-parameter linear controller. We should be able to see by now that the above relation describes an integrated control/diagnostic module and because of its general nature every module of this kind could sufficiently be described by this general

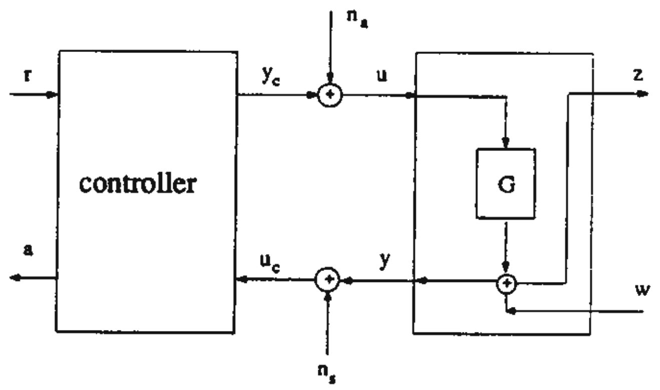

Figure 3: General controller and restricted plant

relation.

In the literature on the theory and applications of the four parameter controller, the control configuration of Fig. 3 has been studied extensively. This configuration consists of a controller of the general form depicted in Fig. 2 and a restricted plant. Note that our interest in the present context is concentrated on the controller rather than the plant and therefore the configuration of Fig. 3 is sufficient enough to illustrate the ideas of the controller with diagnostic capabilities.

As mentioned before, the diagnostic output was introduced in order to reproduce any sensor or actuator failures. The exogenous inputs n_a, n_s represent the deviations from the ideal sensor and actuator outputs respectively. The term ideal implies the absence of any noises or failures in the actuators and sensors. Since we have been talking about noise and failures, we should clarify this dinstinction by defining

$$n_a = f_a + \eta' \tag{2}$$

$$n_s = f_s + \eta \tag{3}$$

where f_a, f_s represent the failures in the actuator and sensor and η', η the respective noises. Having made this dinstinction, we can now specify the objective of the introduction of the additional controller output a, which is <u>to be able to identify and reproduce F at a</u>, where

$$F = \begin{pmatrix} f_a \\ f_s \end{pmatrix} \tag{4}$$

Since our configuration consists a control/diagnostic module, we need first to establish separately the control and diagnostic objectives. The control objectives can be stated as follows :

1) Achieve set point tracking, that is follow or track the reference input r at the plant output z, 2) reject the unmeasured disturbance w at the plant output z, 3) reject both the actuator and sensor noises (η', η) and failures(f_a, f_s) at the plant output z, and 4) achieve the above objectives with limited control action u.

For all these requirements, we have to consider a non ideal model for the plant; in other words, the control requirements have to be satisfied in the context of plant modelling errors. In the same context, the diagnostic objectives can be stated as follows:

1) Isolate faults in sensors and actuators and reproduce them at the controller diagnostic output a, 2) attenuate the actuator and sensor noises η', η at the diagnostic output a, 3) reject the disturbance w at the controller output a, and 4) reject the reference input r at the diagnostic output a.

It has been shown in [5] and [10] that the above requirements lead to certain conflicts. Specifically, the requirement for reproduction of the sensor failure f_s at the diagnostic output contradicts the requirement for the rejection of the sensor noise η and the disturbance. The same problem arises for the actuator, that is the actuator noise rejection contradicts the actuator failure reproduction. Another major tradeoff has to do with the fact that the sensor diagnostic perfomance and the actuator diagnostic performance have to be traded against one another in general.

An interesting application of the four parameter controller was presented by Valavanis *et al.* in [13], [14]. It deals with the robot payload variation problem, which is a crucial problem for the planning and control of robot manipulators. Another recent paper by Juarez *et al.*, [6], formulates the problem of integrated design of control/diagnostics as a linear programming problem and provides a l_∞ framework to delineate the tradeoffs that characterize the integrated control and diagnostic objectives. The four parameter controller has also been implemented on a heat exchanger system, [17], to conduct control simulations and experiments.

Another application of the four parameter controller, [8], is in the design of a sensor failure classifier based on back-propagation neural networks, which distinguish between patterns that are caused by sensor failures and patterns that are caused by model-plant mismatch and disturbances.

2.2 Neural Networks

An ***artificial neural network*** (ANN) or ***connectionist model*** can be defined as a computing system that is made up of a number of simple, interconnected elements that are capable of processing information. The ANN is made up of many parallel elements that are called *neurons*, which can be described by a standard nonlinear algebraic or differential equation. and are interconnected via adaptive multiplicative parameters, called *weights*.

The type of neural network most commonly used is the *feedforward multilayer neural network*. Such a network is made up of any number of layers with any number of neurons in each of these layers. The first layer is designated as the input layer. the last as the output layer, and those in between as the hidden layers. The feedforward multilayer neural network is illustrated in Fig. 4.

We consider an M-layer neural network, that is a network with $M-1$ hidden layers. The output of a node i of the m-layer, where $m = 1,..,M$, is determined by first summing all the weighed inputs to this node and then processing the results through the nonlinear fuction. Hence, the output for a given pattern p is given by

$$V^m_{i,p} = f(h^m_{i,p}) = f\Big(\sum_{j=1}^{n_{m-1}+1} w_{ij}\, V^{m-1}_{j,p}\Big) \quad m = 1,..,M \;\; and \;\; i = 1,\ldots n_m \qquad (5)$$

where $V^m_{i,p}$ is the output of the i node of the m-layer for the specific pattern p, $h^m_{i,p}$ is the respective input to that node, the function f is usually the hyperbolic tangent or the sigmoid function, w_{ij} is the weight connection from $V^{m-1}_{j,p}$ to the i node of the m-layer and n_m is the number of nodes in the m-layer. The input vector to the m layer is the augmented input vector, which takes the bias into account, that is the input vector to the m-layer is given by

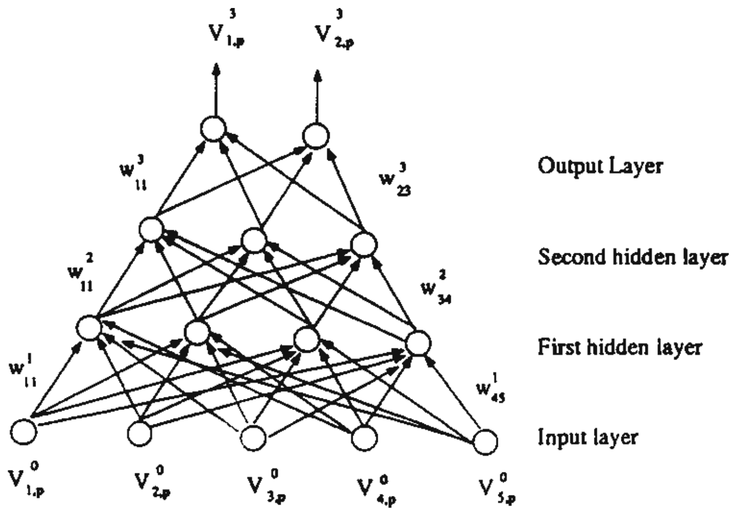

Figure 4: A feedforward multilayer neural network with two hidden layers

$$V_p^{m-1} = (\ V_{1,p}^{m-1} \quad \dots \quad V_{n_{m-1},p}^{m-1} \quad 1\ ) \tag{6}$$

A robust learning heuristic for multilayer feedforward neural network called the *generalised delta rule* (GDR) or *back propagation learning rule* (BP) has been developed by Rumelhart *et al.*, [11]; actually the same algorithm has been invented independently several times in the past. This is the most popular algorithm currently in use and the one that has been adopted in this paper. The BP algorithm is a constant-step-size, gradient descent technique that minimizes the objective function $J(w)$, that is differentiable with respect to each of the weights in the network. This objective function to be minimized is the mean squared error between the actual outputs from the output layer and the desired outputs for all the given patterns and is given as

$$J(w) = \sum_p E_p \tag{7}$$

where E_p is given as

$$E_p = \frac{1}{2} \ \| \ d_p - y_p \ \|^2 \tag{8}$$

Hence, given an input pattern vector x_p, y_p is the output vector generated by the forward propagation of the activation through the network, and d_p represents the desired output vector associated with the x_p.

The weights are adjusted after every iteration, that is after every pass of the training set, by the constant-step-size gradient descent rule :

$$w_{ij}^m(k+1) = w_{ij}^m(k) + \eta \sum_p (\delta_{i,p}^m V_{j,p}^{m-1}) + \alpha \, \Delta w_{ij}^m(k) \tag{9}$$

where η is a gain term, called the *learning rate*, and α is a *momentum* term, typically between 0 and 1, that smooths the effect of dramatic weight changes by adding a fraction of the weight change $\Delta w_{ij}^m(k)$ from the previous iteration, that is $\Delta w_{ij}^m(k) = w_{ij}^m(k) - w_{ij}^m(k-1)$. The error signal $\delta_{i,p}^m$ is a measure of the distance of the activation level of node i of the m-layer from its desired level. The details that entail this derivation, together with the rules for calculating the error signal of a node can be found in any book in neural networks, as in [3].

The neural networks have extensively been used in control systems. Although a number of key theoretical problems remain, it has been demonstrated successfully, [4], [9], that the neural networks have great promise in the modelling of nonlinear systems. A neural network can also be used as a conventional controller in both open and closed loop configurations. A detailed and exhaustive presentation of applications of neural networks in control can be found in [1], [2], [7], [12], [16]. An illustrative way of how neural nets can be used in control is their application on the implementation of the CDC controller in the sections that follow. In these sections, we shall present in detail useful and poweful techniques entailed in neural network control design and present implicitly those parameters that determine the optimal design.

3 ACTUATOR FAILURES

3.1 Introduction

In this section we assume that we have only actuator failures. Therefore, for the time being, the sensor failures are neglected and we concentrate exclusively on the study of the system, when only actuator failures take place. Our task is to train and test a neural network model of the CDC controller, that will have the following task:

"to achieve set point tracking, that is to follow the reference input r at the plant output y_p, and also to isolate the actuator faults and reproduce them at the diagnostic output a_{act}."

At this point, we should clarify what we mean by saying *reproduce*. It is the common trend-direction in the failure diagnosis research work to establish an upper limit-bound, so that a failure is announced when this limit is exceeded. That means that an alarm signal is activated, when a failure occurs. The above mentioned limit actually works as a safety, or better, tolerance level, beyond which the operation of the system is not acceptable, that is the system operates under failure. Although this limit can be defined theoretically in many possible ways, depending on the theory that one uses, H_∞ for instance, it is basically quite arbitrary, since each theoretical foundation relies on specific assumptions, which are usually arbitrary themselves.

In practice, this safety limit is given by the designer depending on the specific application. Its value designates the magnitude of failure that can be tolerated in each specific case. Therefore, it appears more reasonable and realistic to try to reproduce the failure as accurately as possible, and then allow ourselves to make the decision whether the alarm should be activated or not. This is exactly the diagnostic objective of this paper and this was actually the objective of the first paper, [10], where the theory of the four-parameter controller was introduced.

The configuration that we are going to study is shown in Fig. 5. As we can see a neural network model for the CDC controller has been adopted. This controller has two general inputs: the reference signal r and the output of the plant y_p. The controller has also two outputs: the diagnostic output a_{act}, and the y_c output, which can be considered as the ideal actuator input. The diagnostic output a_{act} will be designed to give us the reproduction of the actuator failure, if a failure does occur at the actuator.

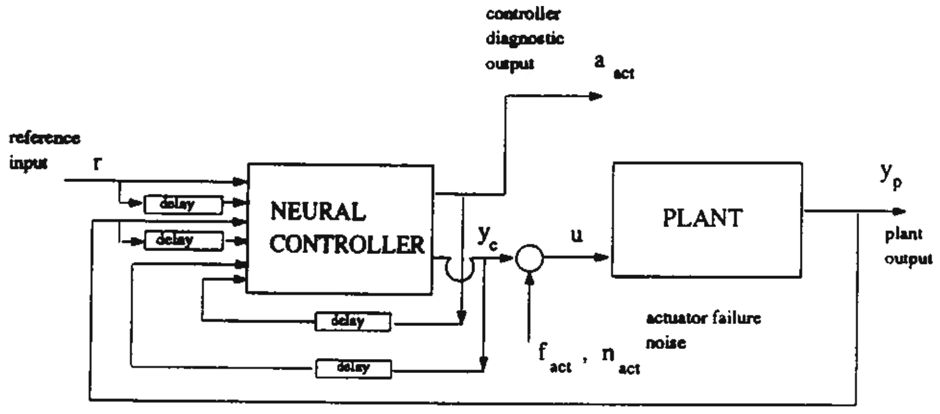

Figure 5: Neural network CDC controller for actuator failures

We also have some more additional inputs to the neural controller. These are the delayed reference inputs, delayed plant outputs, as well as delays of both the controller outputs. The participation of the above signals is typical in the neural network controller design, or generally we should say that the inclusion of delays of both the inputs and outputs of a neural model in the input of the model is characteristic of neural networks in control applications, since this inclusion improves the behavior and effectiveness of the neural model.

For all the neural networks there are always two stages involved. The first stage is training and the second stage is testing. The training has to be done according to the specifications of the problem. In the present case, that implies that our controller has to be trained, so that it can give us the failure that occurs at the actuator. At the same time, the controller has also to operate as a typical controller, that is it has to be trained so that it also follows the reference input at the output of the plant under the presence of failures.

In order to train the neural model of the controller, we need to know the error at the output of the controller. As far as the error at the diagnostic output a_{act} is concerned, we simply compute the difference $(a_{act}^{ideal} - a_{act})$. On the other hand, we have no information about what the ideal output y_c^{ideal} should be. If what we are given is a neural model of the plant, then we proceed as explained below. If we don't have a model of the plant, then we need to find a neural identification model of the plant. The purpose of this neural model is to provide a path for the plant output error to reach the controller, so that the error at the controller output y_c can be computed. This is illustrated in Fig. 6.

The back-propagation algorithm is used to train the neural controller, as well as to back-

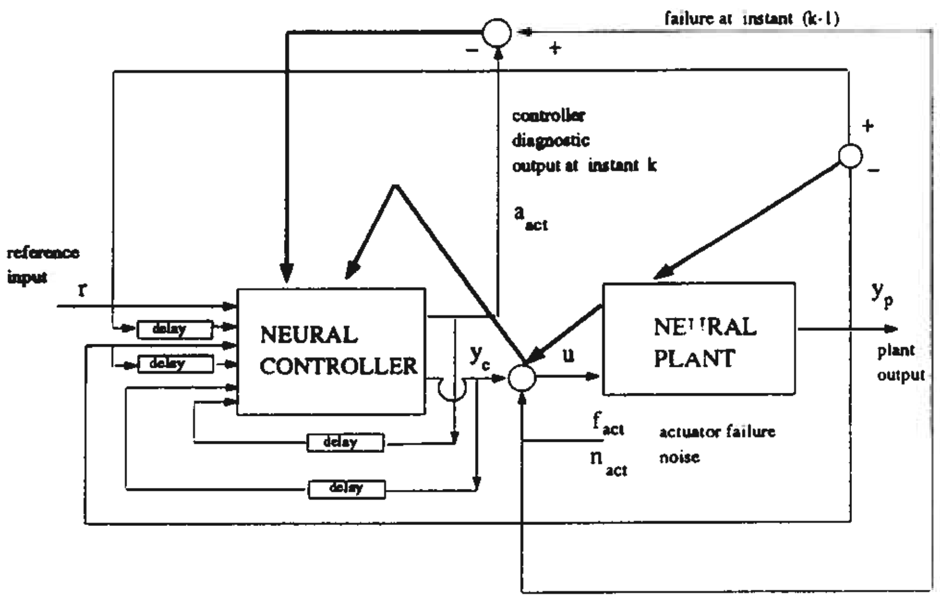

Figure 6: Training of neural CDC controller for actuator failures

propagate the error through the plant, that is from the output of the plant identification model to its input. Since the propagation of the error from the neural plant output to the controller output has to be modified to account for the participation of failures at the actuator, we should now see in detail how, for every single iteration, the signals move from the controller input towards the plant output first, and then how the errors from the neural plant output to the controller inputs are computed.

For a specific time instant, the current and delayed reference inputs, the current and delayed neural plant outputs, as well as previous neural controller outputs, both of a_{act} and y_c, are applied to the input of the controller and propagate to its outputs. The diagnostic output, denoted by $a_{act}(k)$, reproduces the failure. The second output, denoted by $y_c(k)$, comes to the actuator and is added to the actuator failure, denoted by $f_{act}(k)$. The resulting signal, denoted by $u(k)$, is the input to the neural plant, according to the relation

$$u(k) = y_c(k) + f_{act}(k) \tag{10}$$

This input, together with the previous plant outputs, propagate through the neural model to its output. As we can see, no actuator noise is considered during the training. At this point, it should be stated that the desired output at the plant is the current reference input, denoted by $r(k)$, considered at the beginning of this description, and that the desired diagnostic output is the failure at the previous time instant, that is $f_{act}(k-1)$.

The error at the neural plant output is obviously $r(k) - y_p(k)$. This error is back-propagated to the input of the neural plant. The error that corresponds to the input $u(k)$ is given by

$$error_{u(k)} = u^{ideal}(k) - u(k) \tag{11}$$

from which we readily have that

$$u^{ideal}(k) = error_{u(k)} + u(k) \tag{12}$$

Now the ideal output $y_c^{ideal}(k)$ of the controller, that is the one that would satisfy the control requirement of the controller, is given by

$$y_c^{ideal}(k) = u^{ideal}(k) - f_{act}(k) \tag{13}$$

which implies that the error that corresponds to the controller output $y_c(k)$ is given by

$$error_{y_c(k)} = y_c^{ideal}(k) - y_c(k) \tag{14}$$

This error, together with the diagnostic output error mentioned above, are used by the BP algorithm to adjust the weights of the neural controller. The nonlinear plant model that has been used, [9], is described by the following equation:

$$y_p(k+1) = \frac{y_p(k)y_p(k-1)(y_p(k)+2.5)}{1+y_p^2(k)+y_p^2(k-1)} + u(k) \tag{15}$$

For the above plant model an identification neural model has already been found in [4]. This model was used for the training of the neural controller, according to the procedure that was described in the previous section.

The first issue that was investigated was the number of hidden layers. The performance of neural models for the CDC controller with only one hidden layer was very poor. The convergence was very slow and the final result could not even be accepted as satisfactory, since the task of tracking the reference input and reproducing the failure at the actuator was not performed to such a point that this implementation of the CDC controller could be characterized as successful. Therefore, a second hidden layer was considered, which resulted in a dramatic change in the neural controller's performance. It should be noted that the choice of two hidden layers is in accordance with previous work, as reported in [4], [9].

After the number of hidden layers was decided and finally adopted, the next issue to be investigated was the number of neurons in each layer. After several trials, it was decided that a structure with 20 neurons in the first hidden layer and 5 neurons in the second hidden layer was sufficient enough to evaluate the performance of the controller while keeping the *cpu time* needed for the simulations within a reasonable range. While several combinations of more neurons in both layers didn't improve the performance of the controller significantly, they did increase the *cpu time* needed for the simulations.

A critical issue in the design of the controller was found to be the number of delays for the reference input r, the plant output y_p, as well as the previous outputs of the neural controller. The best performance was observed when 3 delays were considered for the inputs of the controller. Although it could be expected that the more delays, the better the performance, here we see that, when the number of delays exceeds an upper limit, 3, both the control and diagnostic performance deteriorate. This can be easily justified, because as we see next, we have forced the neural controller to operate under totally random excitations, and therefore additional memory, in terms of the delayed signals, not only fails to improve the performance, but also imposes an excessive and unnecessary task to the neural controller.

This result is firmly associated with the training set that was used during the training of the controller. Specifically, a random signal uniformly distributed in the range [-2, 2] was considered for the reference input. The failures are usually in the form of step or ramp functions, that is step or ramp functions are introduced as failures at a specific time instant in the system. Therefore, it was considered sufficient that by applying as failures the samples of a typical continuous function, as a sinusoidal, we could submit the network to a sufficient strain, were it to be able to reproduce a step or ramp failure, if such a failure occurs at the system. Therefore, the failure training signal was chosen as

$$f_{act}(k) = 0.8\ sin(2\pi(k + 1.1)/30) \tag{16}$$

The points taken from this continuous function, together with the random points for the reference input, prepare the neural network sufficiently, in order to reproduce a continuous failure, as a step or a ramp.

Another issue that has to be discussed is the criterion used to evaluate both the control and diagnostic performance of the neural model of the four-parameter controller. After a specific number of iterations, a sinusoidal signal was tested as the reference input and a step signal was tested as the actuator failure. Specifically, the reference signal was chosen to be

$$r = 2\ sin(2\pi k/25) \tag{17}$$

whereas the failure was typically taken as a step of 0.8. It should also be mentioned that for the testing actuator noise, uniformly disributed in the range [-0.05, 0.05], was applied as an input to the actuator for all the experiments. This actuator noise should be considered as simulating the plant modelling errors that enter any physical system.

The difference vector $(R - Yplant)$ was considered, where R is the vector with the sampling values of the reference signal and $Yplant$ is the vector with the respective outputs of the neural plant. Then, the absolute values of all the elements of this difference vector were defined and the mean value was found. This *mean absolute value*, as it can be called, defined as *rmean*, is the criterion of whether the tracking is satisfactory or not.

The same approach was followed for the evaluation of the diagnostic requirement. In this case, the difference vector $(A_{act} - F_{act})$ was considered, where A_{act} is the vector with the values of the diagnostic output of the controller and F_{act} is the vector with the respective samples of the failure introduced at the actuator. Again, the absolute values of all the elements of this difference vector were defined and the mean value was found. This *mean absolute value*, defined as *actmean*, is now the criterion of whether the failure reproduction is satisfactory or not.

The training of the neural controller was done in two steps. First, the four-parameter controller was trained as a conventional controller, that is as a controller without diagnostic capabilities. Therefore, in this step, no failure signals participated in the training and the controller was trained, so that it tracks the reference input r at the plant output, and at the same time it gives a 0 at the diagnostic output a_{act}. The weights found in this step are used as initial conditions for the training of the second step, where the controller is trained, in order to meet both the control and diagnostic requirements.

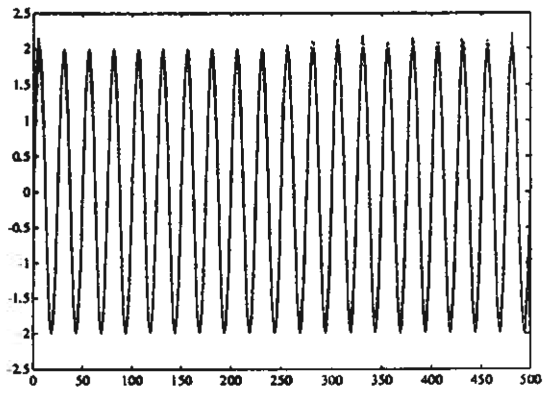

Figure 7: Neural plant during the test. Reference input (solid line) and plant output (dotted line). 3 delays, 200,000 iterations, rmean=0.1155.

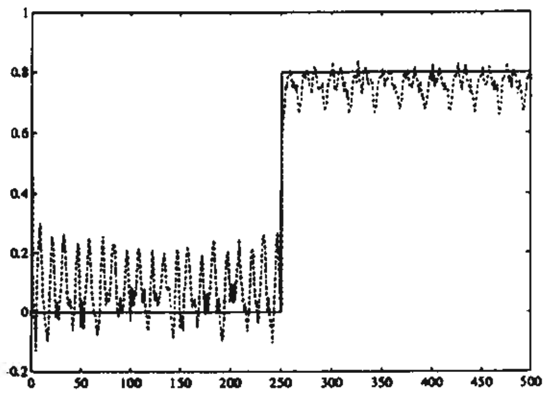

Figure 8: Neural plant during the test. Actuator failure (solid line) and diagnostic output (dotted line). 3 delays, 200,000 iterations, actmean=0.0753.

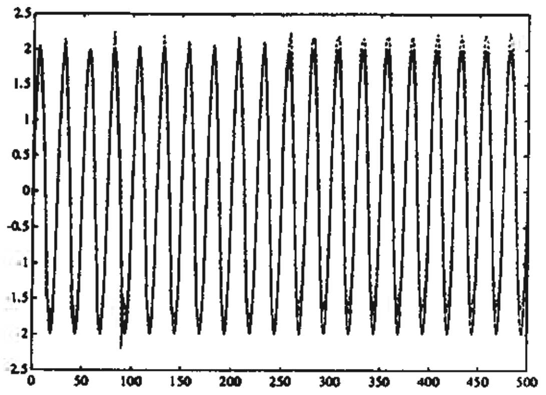

Figure 9: Real plant during the test. Reference input (solid line) and plant output (dotted line). 3 delays, 200,000 iterations, rmean=0.2045.

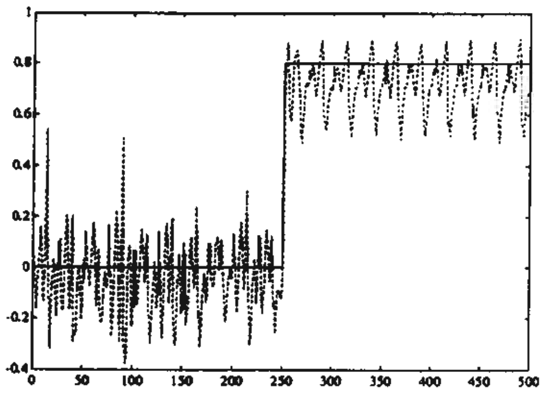

Figure 10: Real plant during the test. Actuator failure (solid line) and diagnostic output (dotted line). 3 delays, 200,000 iterations, actmean=0.1132.

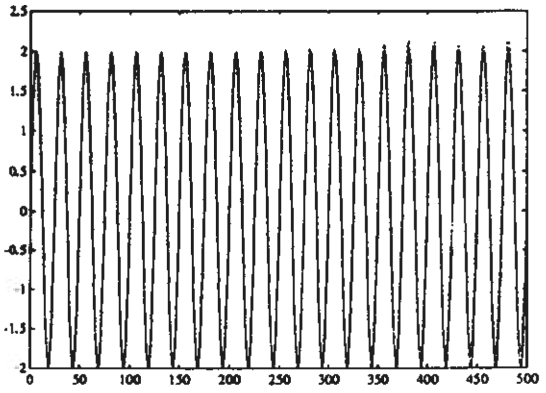

Figure 11: Neural plant during the test. Reference input (solid line) and plant output (dotted line). 3 delays, 200,000 iterations, rmean=0.0872. Ramp failure.

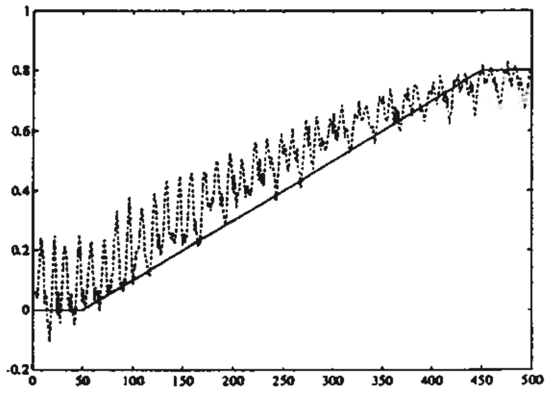

Figure 12: Neural plant during the test. Actuator failure (solid line) and diagnostic output (dotted line). 3 delays, 200,000 iterations, actmean=0.0844. Ramp failure.

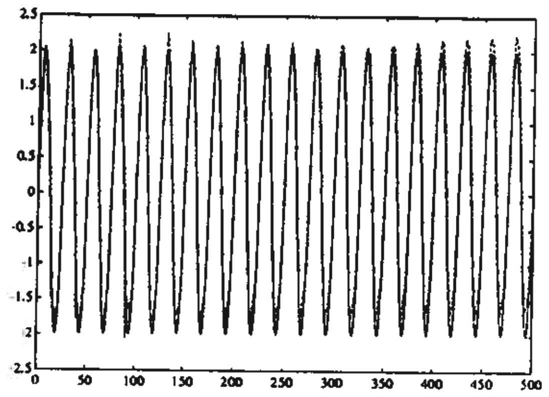

Figure 13: Real plant during the test. Reference input (solid line) and plant output (dotted line). 3 delays, 200,000 iterations, rmean=0.1681. Ramp failure.

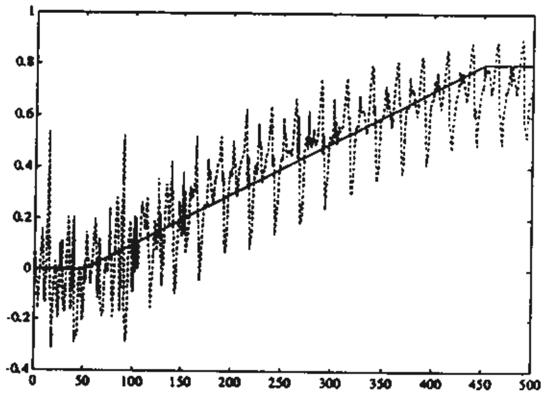

Figure 14: Real plant during the test. Actuator failure (solid line) and diagnostic output (dotted line). 3 delays, 200,000 iterations, actmean=0.1059. Ramp failure.

The behavior of the plant is shown in Fig. 7 - Fig. 10. The results in Fig. 7 and Fig. 8 correspond to the test that has taken place with the testing reference signal a[illegible] failure mentioned above, when the neural model is in place of the real plant. Hence, the output of the neural plant, when tested with the sinusoidal reference input given in (17), is given on Fig. 7 after 200,000 iterations . Next the diagnostic output of the controller is given on Fig. 8. When the real plant was placed instead of its neural network model, then the same test took place, as far as the testing signals are concerned, and the results are given next. The results are given in Fig. 9 and Fig. 10 for the same number of iterations.

As mentioned before two failures often seen in literature are the step and ramp functions. Therefore, it should be of interest to see how the system, that we have already trained, will behave under a ramp failure. As before, two cases will be considered: the case when only a neural network model of the plant is available, and the case when the neural network is only used to train the controller by back-propagating the error. For the first case, we see that our controller is capable of reproducing the ramp failure under the presence of uniformly distributed noise. Indeed, from Fig. 11 and Fig. 12 it is clear that not only does the controller track the reference input at the plant output, but also reproduces clearly the failure that occured at the actuator. Then, the real plant is introduced and the same testing signals are applied. This case is illustrated in Fig. 13 and Fig. 14. As we can easily see, even with the real plant the controller meets satisfactorily its specifications. It should be noted that for all the tests in this section the performance of the CDC controller is slightly better when using the neural plant model during the testing, which was expected since it was this neural model that was used during the training. For all the simulations conducted in this part a momentum term of 0.9 has been used and the learning rate has been 0.01.

3.2 Weight Assignment on Control and Diagnostic Objectives

The BP algorithm minimizes the mean squared error between the actual outputs from the output layer and the desired outputs. Therefore, for a neural network with N outputs at the output layer, the sum of N squares has to be minimized, each of these squares being the difference between the anticipated and the real output. Hence, the minimizing quantity can be written as

$$\sum_{i=1}^{N} \frac{1}{2} (d_i - y_i)^2 \tag{18}$$

where d_i is the desired value, and y_i is the real value at the i^{th} output of the neural network. In minimizing this quantity, we can assign weights to the square terms that participate in the sum, and by doing so we emphasize those terms that are assigned the largest weights. Therefore, the previous relation can be written as

$$\sum_{i=1}^{N} \frac{1}{2}\, \omega_i (d_i - y_i)^2 \tag{19}$$

At this point, we should try to understand the significance of these weights for the CDC controller we are studying here. We have already discussed the interactions between the control and the diagnostic objectives. We have seen that they are so closely associated that whenever we try to enhance the performance of one of them, the performance of the other deteriorates. Hence, depending on the specific application, we either assign a larger weight to the control objective, which results in a better tracking of the reference input at the output of the plant, or to the diagnostic objective, which results in a better and more reliable reproduction of the actuator failures, if any, at the output of the controller.

The way to do all these is by looking at the controller outputs. Since the CDC controller has two outputs, the control and the diagnostic, by assigning various weights to these two outputs, we simply train the controller in the direction we desire; in other words, if what we really care about is the conventional aspects of the controller we only have to assign a small weight to the error that is associated with the diagnostic output a_{act}. If, on the other hand, what we need to know is whether a failure has occured at the system, we emphasize on the weights, that are associated with the diagnostic output. Therefore, the minimizing quantity at the output of the controller is given as follows:

$$J = \frac{1}{2}\, \omega_1\, error^2_{a_{act}} + \frac{1}{2}\, \omega_2\, error^2_{y_c} \tag{20}$$

where

$$error_{a_{act}} = a^{ideal}_{act} - a_{act} \tag{21}$$

$$error_{y_c} = y^{ideal}_{c} - y_c \tag{22}$$

weights	rmean	actmean
0.2 $error_{y_c}$	0.1579	0.0889
0.3 $error_{a_{act}}$	0.1287	0.1670

Table 1: Rmean, actmean when assigning weights to the controller outputs. The neural model for the plant is used for the test.

weights	rmean	actmean
0.2 $error_{y_c}$	0.2303	0.1122
0.3 $error_{a_{act}}$	0.2491	0.2260

Table 2: Rmean, actmean when assigning weights to the controller outputs. The real plant is used for the test.

In order to illustrate the above points, two cases have been considered. The neural model for the controller is the one chosen before, with 3 delays. Again, the weights obtained by training the controller as a conventional controller, that is by requiring the diagnostic output to be equal to zero, have been used as initial condition. In the first case, the diagnostic output is assigned the weight $\omega_2 = 0.2$ for 200,000 iterations. That implies that we are really interested in the control aspects of our controller, whereas in the other case the second output of the controller, that is the one that goes through the actuator, is assigned the weight $\omega_1 = 0.3$ for 200,000 iterations, implying that now we rather prefer to concentrate on the diagnostic aspects.

The controller was trained with a neural model for the plant, and then the testing signals mentioned before were considered. The results are given in Table. 1. As expected, we see that in the case, where the weight $\omega_1 = 0.3$ was assigned to the diagnostic output the diagnostic capabilities of the controller became less powerful, whereas the reference tracking seemed to be quite successful. Similarly, in the second case, where the weight $\omega_2 = 0.2$ was assigned to the second controller output, the diagnostic performance is excellent, but the control performance is not as good as in the first case.

After these tests were completed, the real plant was considered, instead of its neural model, and exactly the same tests were performed. The performance of the controller is the same as before, that is depending on the weights, either the diagnostic or the control aspects are emphasized. The results are given in Table. 2. From this table, compared to Table. 1, we can see again that the replacement of the neural model by the real plant results to a performance not as excellent as the one with the neural model, but still quite illustrative of the points that have been highlighted here.

4 SENSOR FAILURES

4.1 Introduction

In this section we assume that we have only sensor failures. This assumption implies that, in the present context, actuator failures either don't occur at all, or when they do occur, their influence on the overall system is negligible. Now, our objective is to train a neural network model of the CDC controller that will have the following task:

"to achieve set point tracking, that is to follow the reference input r at the plant output y_p, and also to isolate the sensor faults and reproduce them at the diagnostic output a_{sens}."

The importance of this choice, to reproduce the failure signal rather, than check whether it exceeds a certain upper limit-bound, has already been diccussed. At this point, it should be noted that for the configuration of interest here, which is depicted in Fig. 15, the sensor failures account for the disturbances that occur at the output of the plant: hence it is these disturbances that we are trying to isolate at the diagnostic output a_{sens}.

Once again, we realize that although it is easy to train the neural network of the controller for the failure identification objective, since we know at any time instant, during the training, what the diagnostic output of the controller should be, we have no information however about the ideal output at the second output of the controller, that is the one which is an input to the plant.

If the information about the behavior of the plant is given by a neural network representation, then this neural network can be used to provide us with all the information we

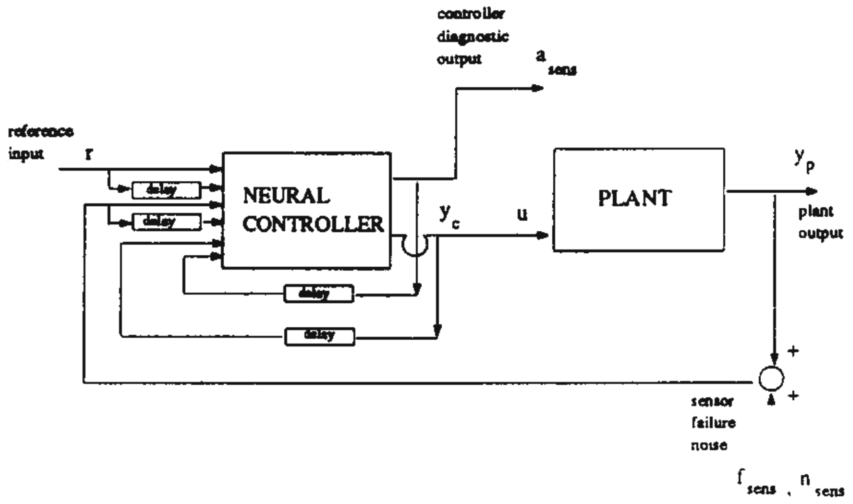

Figure 15: Neural network CDC controller for sensor failures.

need about the error at the output of the controller. If such a model is not available, or the plant is described by an accurate and reliable representation, differential or state equations for the continuous case, difference equations for the discrete, then we need to find first an identificaton model of the plant. This situation, that is the situation where a neural model of the plant is used for the training, is illustrated in Fig. 16.

The situation here resembles the one we had in the actuator case: the main difference, though, is that now the propagation of the signals and the errors is quite straightforward. Specifically, the signal at the second output of the controller moves towards the neural model of the plant, without being 'annoyed' by any failures at the actuator at this time. After propagating through the layers of the neural model, it gets to the output, where it is added to the sensor failure, denoted by f_{sens}. The resulting signal is going to be the second general input to the CDC controller; the other general input is the reference signal as known, where by general the current, together with the previous values of the signal are implied.

As far as the errors are concerned, the easily computed error at the output of the plant - it is easily computed, because we know that the expected plant output is the reference input at the controller - is back-propagated to the plant input. One of the components of the error vector at the input of the plant is the error at the second output of the controller $error_{y_c}$. This error, together with the easily computed error at the diagnostic output of the controller are used by the BP algorithnm to train the neural network.

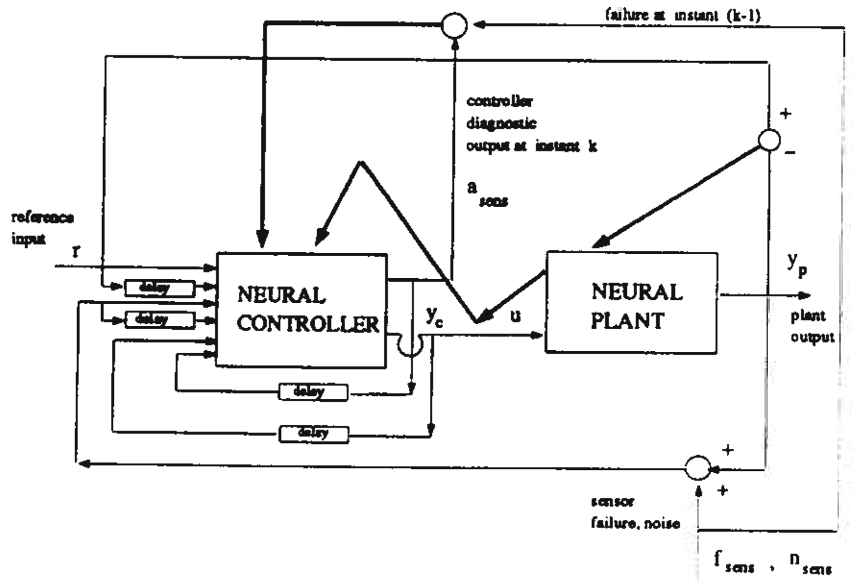

Figure 16: Training of neural CDC controller for sensor failures.

The same issues, with the actuator case, should be adressed here. The 3 delays at the input of the controller and the internal structure of the neural model of the controller (number of layers, neurons per layer) have already been tested and found to perform satisfactorily, as far as the fulfillment of both the control and diagnostic requirements for the controller are concerned, and there is no need to be changed now. The same is true for the learning rate. as well as the training and testing signals.

The same plant model was used with the actuator case, that is the one described by the input-output equation (15). Once more, the criterion for the evaluation of the control performance of the CDC controller was the mean absolute value *rmean*, that is the mean of the absolute values of all the elements that constitute the difference vector $R - Yplant$, where these vectors have as elements the samples of the reference signal and the output of the plant. Similarly, the criterion for the evaluation of the diagnostic performance was the mean of all the absolute values *sensmean* of the elements of the difference between the vector F_{sens}, that contains the samples of the testing sensor failure and the vector A_{sens}, that contains the samples of the controller diagnostic output.

The initial vectors for the training were not arbitrary; actually, as in the actuator case the controller was trained initially as a conventional controller, that is it was trained to

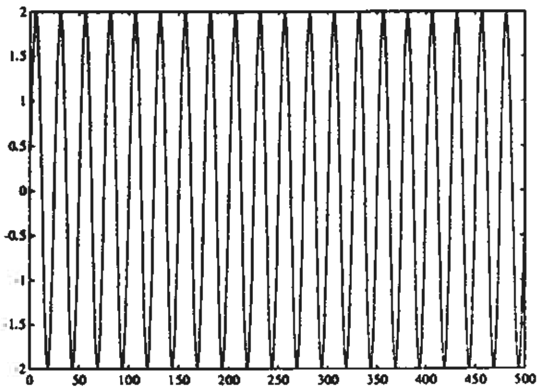

Figure 17: Neural plant during the test. Reference input (solid line) and plant output (dotted line). 3 delays, 200,000 iterations, rmean=0.0487.

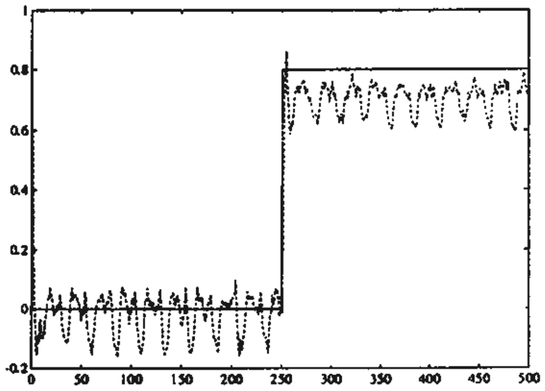

Figure 18: Neural plant during the test. Sensor failure (solid line) and diagnostic output (dotted line). 3 delays, 200,000 iterations, sensmean=0.0794.

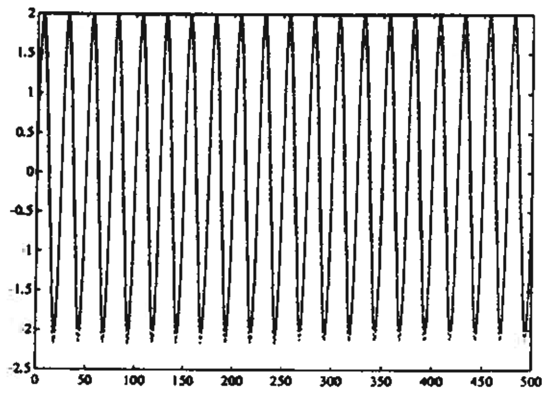

Figure 19: Real plant during the test. Reference input (solid line) and plant output (dotted line). 3 delays, 200,000 iterations, rmean=0.1632. A step failure was considered.

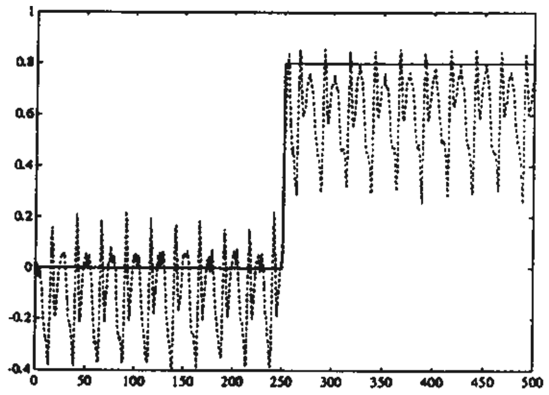

Figure 20: Real plant during the test. Sensor failure (solid line) and diagnostic output (dotted line). 3 delays, 200,000 iterations, sensmean=0.1676. A step failure was considered.

perform under ideal conditions, implying no failures, in order to track the reference input and at the same time give no signal at the diagnostic output a_{sens}. This initial condition for the weights accelerates the convergence of the training, since it emphasizes the control aspect of the controller. It is like warming-up the controller with an easy task, until the really difficult task finally begins.

The training signal for the reference input was again random, uniformly distributed in range [-2, 2], where for the sensor failure a continuous signal like the one in (16) was chosen. For the testing a sinusoidal was chosen as reference input, as in equation (17), where for the failure at the sensor a step of 0.8 was tested. Once more, noise uniformly disributed in the range [-0.05, 0.05], was applied at the output of the plant for all the experiments. This noise simulates the plant modelling errors that enter the system.

All the above cited results can be better seen in the plots that are given next. In Fig. 17 and Fig. 18 the neural model of the plant that was used for training is again used for the testing. We see that both the task of reproducing the sensor failure and tracking the reference input are performed quite successfully. Then the same tests are performed with the real plant and the results are given in Fig. 19 and Fig. 20.

4.2 Weight Assignment on Control and Diagnostic Objectives.

The assignment of weights on the participating terms in the minimizing cost function is again of interest here. Now, the cost function is given as follows

$$J = \frac{1}{2}\,\omega_1\; error^2_{a_{sens}} + \frac{1}{2}\,\omega_2\; error^2_u \tag{23}$$

since, as stated before, the error at the second output of the controller is just the error that corresponds to the input u of the plant. The importance of the assignment of these weights has already been discussed. The results obtained when considering, from the beginning of the training, weights for the controller outputs are similar to the ones reported for the actuator case; that is by assigning a weight to the diagnostic output we improved the control capabilities, while by assigning a weight to the input of the plant we improved the diagnostic capabilities. Here, we considered as initial condition for the training the layer-weights that were found before after 200,000 iterations of training of the controller, without assigning weights to its ouputs. This was done, because we thought it would be interesting to see how

weights	rmean	actmean
0.3 $error_{y_c}$	0.0454	0.0615
0.3 $error_{a_{sens}}$	0.0252	0.0591

Table 3: Rmean, sensmean when assigning weights to the controller outputs. The layer-weights of the controller trained for 200,000 iterations without any weights assigned are used for initial condition. The neural model for the plant is used for the test.

weights	rmean	actmean
0.3 $error_{y_c}$	0.0979	0.1017
0.3 $error_{a_{sens}}$	0.0699	0.0969

Table 4: Rmean, sensmean when assigning weights to the controller outputs. The layer-weights of the controller trained for 200,000 iterations without any weights assigned are used for initial condition. The real plant is used for the test.

the behavior of the CDC controller for this specific plant, without any particular weights assigned to its outputs initially, could change, once we decided to emphasize either of its two objectives.

Two cases were investigated. First the error at the diagnostic output of the controller was assigned a weight of $\omega_1 = 0.3$ and then the error at the input of the plant was assigned the same weight $\omega_2 = 0.3$. The results are in Table. 3. By comparing the results of both cases, we see that although the control behavior was evidently improved, by assigning the weight of $\omega_1 = 0.3$ at the diagnostic output, the diagnostic performance didn't differ significantly in both cases, which implies that for this specific plant and structure (layers and neurons), the weights that correspond to the diagnostic output have reached a level that no further improvement can be accomplished. In other words, for this specific plant the initial training of 200,000 iterations was sufficient for the diagnostic performance of the controller, and hence the additional 200,000 iterations couldn't change the picture dramatically, even when an emphasizing weight was assigned to the diagnostic output.

The results in Table. 3 were taken by testing the neural model of the plant. The same test was also performed for the real plant. The results are given in Table. 4. These results correspond to the training of the controller for 200,000 iterations. From these results, we see again what was mentioned above; specifically that although there is an evident improvement in the control behavior, the diagnostic behavior remains the same, even in the case where the emphasis was given to the diagnostic output.

Now that we have exploited the behavior of the system for different choices of the weights assigned at the output errors, we can test it for ramp failures. The configuration with $\omega_1 = 0.3$ studied before was used. These results are better illustrated in Fig. 21 and Fig. 22 for the test on the neural plant and in Fig. 23 and Fig. 24 for the test on the real plant. From these results, we see that the response of the system to a ramp failure is just as good as with the step failure studied before. This was expected, since the controller has been trained, so that it can reproduce any continuous function introduced as sensor failure in the system. Finally, it should be noted that a momentum term of 0.1 and a learning rate of 0.01 have been used for all the experiments in this section.

5 CONCLUSIONS

In this paper, a new approach to implementing a controller with diagnostic capabilities (CDC) has been presented. The CDC controller, in addition to the tasks of a conventional controller, is designed to provide information about failures taking place in the system. A neural network of the CDC controller was implemented here for the first time. The major advantage of this implementation is that it can also be applied to nonlinear plants, as the example presented in this paper, whereas the literature in the four-parameter controller has been restricted to linear plants so far. It was shown via numerous experimental results in this paper that instead of examining every time whether a specific signal has exceeded an acceptable limit-bound, in which case an alarm is triggered, it is preferable to try to reproduce the failure. This is true even under noise introduced in the system, which accounts for plant modelling errors.

Although the interest is usually in the sensor failures, here a unified approach for both sensor and actuator failures has been introduced for the design of a controller, which is

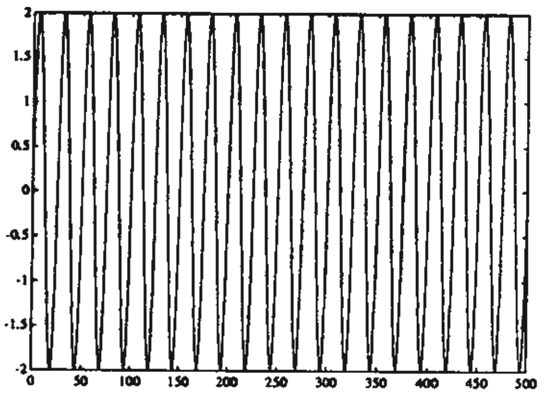

Figure 21: Neural model of the plant during the test. Reference input (solid line) and plant output (dotted line). 3 delays, 200,000 iterations, rmean=0.0255. Diagnostic controller output assigned a weight $\omega_1 = 0.3$. Ramp failure.

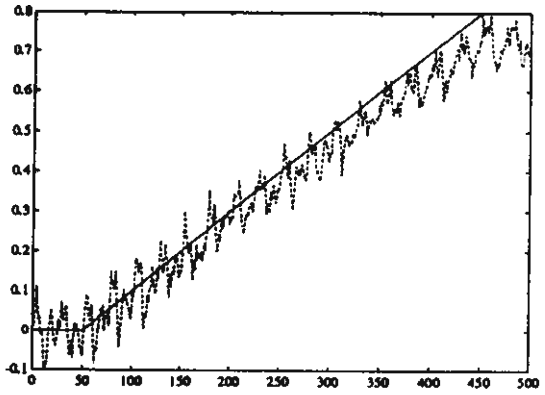

Figure 22: Neural plant during the test. Sensor failure (solid line) and diagnostic output (dotted line). 3 delays, 200,000 iterations, sensmean=0.0488. Diagnostic controller output assigned a weight $\omega_1 = 0.3$. Ramp failure.

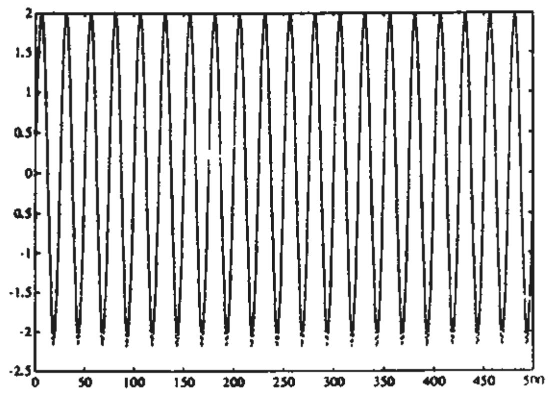

Figure 23: Real plant during the test. Reference input (solid line) and plant output (dotted line). 3 delays, 200,000 iterations, rmean=0.0719. Diagnostic controller output assigned a weight $\omega_1 = 0.3$. Ramp failure.

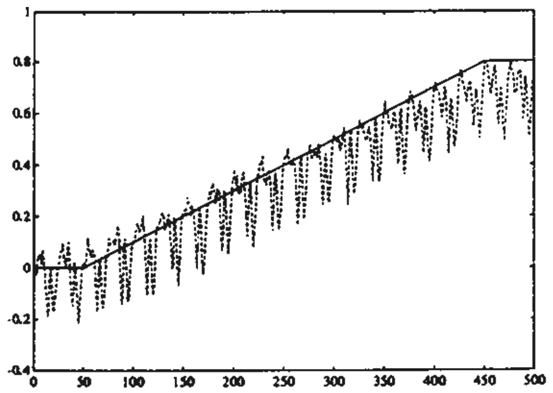

Figure 24: Real plant during the test. Sensor failure (solid line) and diagnostic output (dotted line). 3 delays, 200,000 iterations. sensmean=0.0870. Diagnostic controller assigned a weight $\omega_1 = 0.3$. Ramp failure.

capable of reproducing failures at the actuator as well. The actuator case seems to be harder to train, because of the intervention of the actuator failures in the path of the back-propagating errors, but this was settled by appropriately defining the error at the output layer of the neural controller.

The trained controller was finally tested with both the neural model of the plant and the real plant. The term real plant corresponds to the difference equation that describes it. The results for the neural model were generally better than the one with the real model. This was expected, since it was the neural model that was used for the training of the controller. This remark also leads to the conclusion that if we are really interested in the reproduction of the failure, we should prefer the neural model, since it can gives us a clearer reproduction of the failure that occured in the system.

Since both control and diagnostic objectives characterize diagnostic controller, with our design we have the flexibility of concentrating on one of these objectives by appropriately emphasizing the contribution of this specific objective in the cost function to be minimized by the back-propagation algorithm. This is quite a useful feature, since the requirements are different for each specific application, and therefore at cases, where for instance the failure detection is crucial, this design enables us to concentrate on the identification of failures. while at the same time the control requirements are still sufficiently satisfied.

At this point, it should be mentioned that our neural model of the controller with diagnostic capabilities was trained to reproduce a random failure. which is a general and hard problem. Therefore, in cases where either we suspect or we know in advance what a possible failure may be , the training can be modified appropriately, preparing the controller for this specific form of failure. This can obviously result in a much more accurate reproduction of the failure when it does occur in the system, either in the actuator or the sensor.

An issue that should be adressed in the future concerns the design of a neural controller that could provide diagnostic information, when both sensor and actuator failures occur in the system. That suggests a controller with three outputs, instead of the controller with the two outputs studied here, that is two outputs for diagnostics and one output for control. This seems to be quite a difficult computational problem and possibly other forms of neural networks, besides the multilayer feedforward neural network examined here. or alternative ways of training should be investigated.

References

[1] Antsaklis P. J. , "Neural Networks in Control Systems," *IEEE Control Systems Magazine,* vol. 12, no.2, pp. 8-10, April 1992.

[2] Antsaklis P. J., "Neural Networks for the Intelligent Control of High Autonomy Systems," *Intelligent Systems Technical Report 92-9-1,* Department of Electrical Engineering, University of Notre Dame, September 1992.

[3] Hertz J., Krogh A. and Palmer R. G., *Introduction to the Theory of Neural Computation.* Lecture Notes Volume I, Santa Fe Institute Studies in the Sciences of Complexity. Redwood City, CA: Addison-Wesley Publishing Company, 1991.

[4] Hou Z., "Analysis of Auto Powertrain Dynamics and Modelling Using Neural Networks." Master Thesis, Department of Electrical Engineering, University of Notre Dame, February 1992.

[5] Jacobson C. A. and Nett C. N., "An Integrated Approach to Control and Diagnostics Using the Four Parameter Controller," *IEEE Control Systems Magazine,* vol. 11, no. 6, pp. 22-29, October 1991.

[6] Juarez A. E., Ajbar A. and Kantor J. C., "Multivariable Control with Integrated Diagnostics for Chemical Processes," 1991.

[7] Konstantopoulos I. K., "Controller Design with Failure Diagnostic Capabilities via Neural Networks," Master Thesis, Department of Electrical Engineering, University of Notre Dame, November 1992.

[8] Naidu S. R., Zafiriou E., McAvoy T., "Use of Neural Networks for Sensor Failure Detection in a Control System," *IEEE Control Systems Magazine.* vol. 10, no. 3. pp. 49-55, April 1990.

[9] Narendra K. S. and Parthasarathy K., "Identification and Control of Dynamic Systems Using Neural Networks," *IEEE Transactions on Neural Networks.* vol. 1. no. 1. pp. 4-27. March 1990.

[10] Nett C. N., Jacobson C. A. and Miller A. T., "An Integrated Approach to Controls and Diagnostics : The 4-Parameter Controller," *Proceedings of 1988 American Control Conference,* pp. 824-835, June 1988.

[11] Rumelhart D. E., Hinton G. J. and Williams R., "Learning Internal Representations by Error Propagation," in D. E. Rumelhart, J. L. McClelland, eds, *Parallel Distributed Processing : Explorations in the Microstructure of Cognition, vol. 1 : Foundation*, pp. 318-362. Cambridge, MA: MIT Press, 1986.

[12] Sartori M. A., "Feedforward Neural Networks and their Application in the Higher Level Control of Systems," Ph.D. Dissertation, Department of Electrical Engineering, University of Notre Dame, April 1991.

[13] Valavanis K. P., Jacobson C. A. and Gold B., "An Application of the 4-Parameter Controller to the Robot Payload Variation Problem," *Proceedings of 28th Conference on Decision and Control,* pp. 2658-2663, 1989.

[14] Valavanis K. P., Jacobson C. A. and Gold B., "Integration Control and Failure Detection with Application to the Robot Payload Variation Problem," *Journal of Intelligent and Robotic Systems 4*, pp.145-173, 1991.

[15] Vidyasagar M., *Control System Synthesis: A Factorization Approach.* Cambridge, MA: MIT Press, 1985.

[16] Warwick K., Irwin G. W. and Hunt K. J., eds, *Neural Networks for Control and Systems.* IEE Control Engineering Series. London, UK: Peter Peregrinus Ltd., 1992.

[17] Wu X. and Cinar A., "Reliable Process Control by 4-Parameter Controllers," *Proceedings of 1991 American Control Conference,* pp. 1860-1865, 1991.