Controllers with Diagnostic Capabilities.
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. e-mail: antsakli@saturn.ee.nd.edu

(Received: 10 June 1994)

Abstract. Designing controllers with diagnostic capabilities is important as in a feedback control system, detection and isolation of failures is generally affected by the particular control law used. Therefore, a common approach to control and failure diagnosis problems has significant merit. Controllers capable of performing failure diagnosis have additional diagnostic outputs 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 reproduction of the failure occurring 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.

Key words. Integration of controls and diagnostics, feedforward multilayer neural networks, failure diagnosis, four-parameter controller.

1. Introduction

The four-parameter controller is a linear controller used for both control and diagnostics; it was first introduced in [15]. This controller is a generalization of the familiar two-parameter linear controller [22]. 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, in addition to attaining conventional control goals 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.

*Author to whom all correspondence should be addressed.
**A version of this paper under the title ‘The Four-Parameter Controller. A Neural Network Implementation’ was presented at the IEEE Mediterranean Symposium on New Directions in Control Theory and Applications, Chania, Crete, Greece, June 21–23, 1993.
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 [10] and presented at [11]. 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 occurred will appear in its diagnostic output. The idea of reproducing the failure appears attractive, because it allows more sophisticated methods than a single threshold to be used, in order to detect a failure.

Note that here the cases of actuator and 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 outlined. A brief description of neural networks fundamentals is then introduced and a brief 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 discussed 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 Figure 1, is a generalization of the familiar two-parameter linear controller [22]. This controller has two vector inputs and two vector outputs, resulting in a controller with four matrix parameters. The concept of the four-parameter controller is attributed to Nett [15], who also proposed the general control system configuration of Figure 2 in [16].

The signals appearing in Figure 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 unmodelled signals that inevitably

---

Fig. 1. The structure of the four-parameter controller.

Fig. 2. General control system configuration.
appear at the actuator; \( n_s \) is an exogenous input which accounts for unmodelled 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 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 not obviously be manipulated by the controller.

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

\[
\begin{pmatrix}
  a \\
  y_c \\
  K_{11} \\
  K_{21}
\end{pmatrix} =
\begin{pmatrix}
  K_{11} & K_{12} \\
  K_{21} & K_{22}
\end{pmatrix}
\begin{pmatrix}
  r \\
  u_c
\end{pmatrix}
\]  

(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 relation.

In the literature on the theory and applications of the four-parameter controller, the control configuration of Figure 3 has been studied extensively. This configuration consists of a controller of the general form depicted in Figure 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 Figure 3 is sufficient enough to illustrate the ideas of the controller with diagnostic capabilities. Our approach, however, can easily handle the general plant configuration of Figure 2.

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 actuator and sensor outputs respectively. The term ideal implies the absence of any noises or failures in the actuators and sensors. Since

![Fig. 3. General controller and restricted plant.](image)

<table>
<thead>
<tr>
<th>a</th>
<th>y_c</th>
<th>u</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>n_s</td>
<td>u_c</td>
<td>y</td>
<td>w</td>
</tr>
</tbody>
</table>

Fig. 3. General controller and restricted plant.
we have been talking about noise and failures, we should clarify this distinction by defining

\[ n_a = f_a + \eta' \]  
(2)

\[ n_s = f_s + \eta \]  
(3)

where \( f_a, f_s \) represent the failures in the actuator and sensor and \( \eta', \eta \) the respective noises. Having made this distinction, we can now specify the objective of the introduction of the additional controller output \( a \), which is to be able to identify and reproduce \( F \) at \( a \), where

\[ F = \begin{pmatrix} f_a \\ f_s \end{pmatrix} \]  
(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 (\( \eta', \eta \)) 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 \( \eta', \eta \) at the diagnostic output \( a \),
3. Reject the disturbance \( w \) at the diagnostic output \( a \), and
4. Reject the reference input \( r \) at the diagnostic output \( a \).

It has been shown in \([8] \) and \([16] \) 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 \( \eta \) 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 performance 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 \([20, 21] \). 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. \([9] \), formulates the problem of integrated design of control/diagnostics as a linear programming problem and provides an
$l_\infty$ 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 [25], to conduct control simulations and experiments. Another application of the four-parameter controller [13], 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 Figure 4.

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

![Feedforward multilayer neural network with two hidden layers.](image)

Fig. 4. A feedforward multilayer neural network with two hidden layers.
by

\[ V_{i,p}^m = f(h_{i,p}^m) \]
\[ = f \left( \sum_{j=1}^{n_{m-1}+1} w_{ij} V_{j,p}^{m-1} \right) \quad m = 1, \ldots, M \text{ and } i = 1, \ldots, n_m \]  \hspace{1cm} (5)

where \( V_{i,p}^m \) is the output of the \( i \) node of the \( m \)-layer for the specific pattern \( p \), \( h_{i,p}^m \) is the respective input to that node, the function \( f \) is usually the hyperbolic tangent or the sigmoid function, \( w_{ij}^m \) is the weight connection from \( V_{j,p}^{m-1} \) 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

\[ V_{p}^{m-1} = \left( V_{1,p}^{m-1}, \ldots, V_{n_{m-1},p}^{m-1}, 1 \right) \]  \hspace{1cm} (6)

A powerful learning heuristic for multilayer feedforward neural networks called the generalized delta rule (GDR) or back-propagation learning rule (BP) has been developed by Rumelhart et al. [17]; 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 of the output layer and the desired outputs for all the given patterns and is given by

\[ J(w) = \sum_p E_p \]  \hspace{1cm} (7)

where \( E_p \) is given by

\[ E_p = \frac{1}{2} \left\| d_p - y_p \right\|^2 \]  \hspace{1cm} (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:

\[ w_{ij}^m(k+1) = w_{ij}^m(k) + \eta \sum_p \left( \delta_{i,p} V_{j,p}^{m-1} \right) + \alpha \Delta w_{ij}^m(k) \]  \hspace{1cm} (9)
where \( \eta \) is a gain term, called the learning rate, and \( \alpha \) 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 \( \Delta w_{ij}(k) \) from the previous iteration, that is \( \Delta w_{ij}(k) = w_{ij}(k) - w_{ij}(k - 1) \). The error signal \( \delta_{i,j,m} \) 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 on neural networks, as in [5].

Neural networks have been used to detect and identify failures, especially in chemical processes. Some of these applications can be found in [4, 6, 12, 13, 19, 24, 26]. Neural networks have extensively been used in control systems. Although a number of key theoretical problems remain, it has been demonstrated successfully [7, 14], 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 – 3, 10, 18, 23]. An illustrative way of how neural networks can be used in control is their use in the implementation of the CDC controller in the sections that follow. In these sections, we shall present in detail useful and powerful techniques employed in neural network control design and study issues, such as the number of layers, momentum, and training sets, that are entailed in our proposed 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 common 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, $H_{\infty}$ 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 [16], where the theory of the four-parameter controller was introduced.

The configuration that we are going to study is shown in Figure 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 also has 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.

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 feedforward neural networks in control applications, in order to model the dynamical systems.

In neural network modelling there are typically 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 also has to operate as a typical

![Fig. 5. Neural network CDC controller for actuator failures.](image-url)
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 $(a^{ideal}_{act} - a_{act})$. On the other hand, we have no information about what the ideal output $y^{ideal}_c$ should be. The error at the controller output $y_c$ is easily computed, when a neural model of the plant is available, as is the case here. 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 Figure 6.

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

![Diagram](image)

Fig. 6. Training of neural CDC controller for actuator failures.
CONTROLLERS WITH DIAGNOSTIC CAPABILITIES

input to the neural plant, according to the relation

\[ u(k) = y_c(k) + f_{act}(k) \]  \hspace{1cm} (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

\[ error_{u(k)} = u_{\text{ideal}}(k) - u(k) \]  \hspace{1cm} (11)

from which we readily have that

\[ u_{\text{ideal}}(k) = error_{u(k)} + u(k) \]  \hspace{1cm} (12)

Now the ideal output \( y_{c_{\text{ideal}}}(k) \) of the controller, that is the one that would satisfy the control requirement of the controller, is given by

\[ y_{c_{\text{ideal}}}(k) = u_{\text{ideal}}(k) - f_{act}(k) \]  \hspace{1cm} (13)

which implies that the error that corresponds to the controller output \( y_{c}(k) \) is given by

\[ error_{y_{c}(k)} = y_{c_{\text{ideal}}}(k) - y_{c}(k) \]  \hspace{1cm} (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 [14], is described by the following equation:

\[ y_p(k + 1) = \frac{y_p(k)y_p(k - 1)(y_p(k) + 2.5)}{1 + y_p^2(k) + y_p^2(k - 1)} + u(k) \]  \hspace{1cm} (15)

For the above plant model an identification neural model has already been found in [7]. This model was used for the training of the neural controller, according to the procedure that was described in the previous section. Note that this particular plant was used for convenience only; any nonlinear plant that can be modelled by a feedforward neural network could have been used.

The first issue that was investigated was the number of hidden layers. The performance of neural models for the CDC controller with the input layer and 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, as in Figure 4, 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 [7, 14].

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 did not improve the performance of the controller significantly, they did increase the cpu time needed for the simulations.

Another issue in the design of the controller was the number of delays for the reference input \( r \), the plant output \( y_p \), as well as the previous outputs of the neural controller. When 3 delays were considered for the inputs of the controller, the obtained results were very satisfactory, whereas the cpu time was kept reasonable.

As far as the training set that was used during the training of the controller is concerned, 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 excitation, in order 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

\[
f_{\text{act}}(k) = 0.8 \sin \left(\frac{2\pi (k + 1.1)}{30}\right)
\]

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

\[
r = 2 \sin \left(\frac{2\pi k}{25}\right)
\]

whereas the failure was typically taken as a step of 0.8. It should also be mentioned that for the testing phase, actuator noise, uniformly distributed 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 \((R - Y_{\text{plant}})\) was considered, where \(R\) is the vector with the sampling values of the reference signal and \(Y_{\text{plant}}\) is the vector with the respective outputs of the neural plant. Then, the absolute values of all the elements of this difference vector were computed and the mean value was found. This mean absolute value, as it can be called, defined as \(r_{\text{mean}}\), 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 \((A_{\text{act}} - F_{\text{act}})\) was considered, where \(A_{\text{act}}\) is the vector with the values of the diagnostic output of the controller and \(F_{\text{act}}\) 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 computed and the mean value was found. This mean absolute value, defined as \(act_{\text{mean}}\), 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 CDC 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 \(a_{\text{act}}\). The weights are assigned random initial values at this stage. The weights found after 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.

The behavior of the plant is shown in Figures 7–10. The results in Figure 7 and Figure 8 correspond to the test that has taken place with the test reference and failure signals mentioned above, when the neural model was 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 in Figure 7 after 200,000 iterations. Next the diagnostic output of the controller is given in Figure 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 Figure 9 and Figure 10 for the some number of iterations. Note that for all the experiments in this and all the following sections, the initial conditions have been considered the same for the real and the neural plant.

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, would behave under a ramp failure. We should mention that these types of failure have been chosen for our tests, because they are usually encountered in literature. Our system, however, has been trained to deal with other kinds of failure as well. As before, two cases were considered: the case
Fig. 7. Neural plant. Reference input (solid line) and plant output (dotted line). 3 delays, 200,000 iterations, $r_{\text{mean}} = 0.1155$.

Fig. 8. Neural plant. Actuator failure (solid line) and diagnostic output (dotted line). 3 delays, 200,000 iterations, $act_{\text{mean}} = 0.0753$. 
Fig. 9. Real plant. Reference input (solid line) and plant output (dotted line). 3 delays, 200,000 iterations, \( r_{\text{mean}} = 0.2045 \).

Fig. 10. Real plant. Actuator failure (solid line) and diagnostic output (dotted line). 3 delays, 200,000 iterations, \( act_{\text{mean}} = 0.1132 \).
Fig. 11. Neural plant. Reference input (solid line) and plant output (dotted line). 3 delays, 200,000 iterations, $r_{\text{mean}} = 0.0872$. Ramp failure.

Fig. 12. Neural plant. Actuator failure (solid line) and diagnostic output (dotted line). 3 delays, 200,000 iterations, $act_{\text{mean}} = 0.0844$. Ramp failure.
Fig. 13. Real plant. Reference input (solid line) and plant output (dotted line). 3 delays, 200,000 iterations, $r_{mean} = 0.1681$. Ramp failure.

Fig. 14. Real plant. Actuator failure (solid line) and diagnostic output (dotted line). 3 delays, 200,000 iterations, $act_{mean} = 0.1059$. Ramp failure.
when only a neural network model of the plant was available, and the case when the neural network was only used to train the controller by back-propagating the error. For the first case, we see that our controller was capable of reproducing the ramp failure under the presence of uniformly distributed noise. Indeed, from Figure 11 and Figure 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 occurred at the actuator. Then, the real plant was introduced and the same test signals were applied. This case is illustrated in Figure 13 and Figure 14. As we can easily see, even with the real plant the controller met 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 and Section 3.2 that follows, a momentum term of 0.9 has been used and the learning rate has been 0.01.

3.2. WEIGHT ASSIGNMENT TO 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

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

$$\sum_{i=1}^{N} \frac{1}{2} \omega_i (d_i - y_i)^2$$ (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 \( a_{\text{act}} \). If, on the other hand, what we need to know is whether a failure has occurred at the system, we emphasize the weights, that are associated with the diagnostic output. Therefore, the minimizing quantity at the output of the controller is given as follows:

\[
J = \frac{1}{2} \omega_1 \text{error}_{a_{\text{act}}}^2 + \frac{1}{2} \omega_2 \text{error}_{y_c}^2
\]  

(20)

where

\[
\text{error}_{a_{\text{act}}} = a_{\text{act}}^{\text{ideal}} - a_{\text{act}}
\]  

(21)

\[
\text{error}_{y_c} = y_c^{\text{ideal}} - y_c
\]  

(22)

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 prefer to concentrate on the diagnostic aspects.

The controller was trained with a neural model for the plant, and then the test signals mentioned before were considered. The results are given in Table I. 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.

<table>
<thead>
<tr>
<th>Weights</th>
<th>( r_{\text{mean}} )</th>
<th>( a_{\text{act}_{\text{mean}}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2 ( \text{error}_{y_c} )</td>
<td>0.1579</td>
<td>0.0889</td>
</tr>
<tr>
<td>0.3 ( \text{error}<em>{a</em>{\text{act}}} )</td>
<td>0.1287</td>
<td>0.1670</td>
</tr>
</tbody>
</table>
Table II. $r_{\text{mean}}$, $acf_{\text{mean}}$ when assigning weights to the controller outputs. The real plant is used for the test

<table>
<thead>
<tr>
<th>Weights</th>
<th>$r_{\text{mean}}$</th>
<th>$acf_{\text{mean}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2 $error_y_e$</td>
<td>0.2303</td>
<td>0.1122</td>
</tr>
<tr>
<td>0.3 $error_{a_{\text{act}}}$</td>
<td>0.2491</td>
<td>0.2260</td>
</tr>
</tbody>
</table>

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 II. From this table, compared to Table I, we can see again that the replacement of the neural model by the real plant results in a performance not as excellent as the one with the neural model, but still quite illustrative of the points that have been highlighted here. It should be noted that for all the experiments in this section the weights had the same initial values.

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 do not 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 $a_{\text{sens}}$."

The importance of this choice, to reproduce the failure signal rather, than check whether it exceeds a certain upper limit-bound, has already been discussed. At this point, it should be noted that for the configuration of interest here, which is depicted in Figure 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 $a_{\text{sens}}$.

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
Fig. 15. Neural network CDC controller for sensor failures.

Fig. 16. Training of neural CDC controller for sensor failures.
information we 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 discrete, then we first need to find an identification 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 Figure 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 $f_{\text{sens}}$. 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 input, 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 $\text{error}_{y_2}$. This error, together with the easily computed error at the diagnostic output of the controller are used by the BP algorithm to train the neural network.

The same issues, with the actuator case, should be addressed 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 with the actuator case was used, 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 $\tau_{\text{mean}}$, that is the mean of the absolute values of all the elements that constitute the difference vector $R - Y_{\text{plant}}$, 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 the absolute values \( s_{\text{sens,mean}} \) of all the elements of the difference between the vector $F_{\text{sens}}$, that contains the samples of the testing sensor failure and the vector $A_{\text{sens}}$, 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, again with random starting values for the weights. In other words, the controller was trained
Fig. 17. Neural plant. Reference input (solid line) and plant output (dotted line). 3 delays, 200,000 iterations, $r_{\text{mean}} = 0.0487$.

Fig. 18. Neural plant. Sensor failure (solid line) and diagnostic output (dotted line). 3 delays, 200,000 iterations, $sens_{\text{mean}} = 0.0794$. 
Fig. 19. Real plant. Reference input (solid line) and plant output (dotted line). 3 delays, 200,000 iterations, $r_{\text{mean}} = 0.1632$.

Fig. 20. Real plant. Sensor failure (solid line) and diagnostic output (dotted line). 3 delays, 200,000 iterations, $sens_{\text{mean}} = 0.1676$. 
to 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 \(a_{\text{sens}}\). 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 the range \([-2, 2]\), where for the sensor failure a continuous signal like the one in (16) was chosen. For the test 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 distributed 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 Figure 17 and Figure 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 Figure 19 and Figure 20.

4.2. WEIGHT ASSIGNMENT TO CONTROL AND DIAGNOSTIC OBJECTIVES

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

\[
J = \frac{1}{2} \omega_1 \text{error}^2_{\text{sens}} + \frac{1}{2} \omega_2 \text{error}^2_u
\]  

(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 less than one to the diagnostic output we improved the control capabilities, while by assigning a weight less than one 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 outputs. This was done, because we thought it would be interesting to see how 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 III. By comparing the results of both cases, we see that although the control behavior
Table III. $r_{\text{mean}}$, $s\text{ens}_{\text{mean}}$ when assigning weights to the controller outputs. The layer-weights of the controller trained for 200,000 iterations without any weights assigned are used as initial condition. The neural model of the plant is used for the test.

<table>
<thead>
<tr>
<th>Weights</th>
<th>$r_{\text{mean}}$</th>
<th>$s\text{ens}_{\text{mean}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3 $\text{error}_{yc}$</td>
<td>0.0454</td>
<td>0.0615</td>
</tr>
<tr>
<td>0.3 $\text{error}_{sem}$</td>
<td>0.0252</td>
<td>0.0591</td>
</tr>
</tbody>
</table>

was evidently improved, by assigning the weight of $\omega_1 = 0.3$ at the diagnostic output, the diagnostic performance did not 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 could not change the picture dramatically, even when a small weight was assigned to the diagnostic output.

The results in Table III 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 IV. 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 illustrated in Figure 21 and Figure 22 for the test on the neural plant and in Figure 23 and Figure 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

Table IV. $r_{\text{mean}}$, $s\text{ens}_{\text{mean}}$ when assigning weights to the controller outputs. The layer-weights of the controller trained for 200,000 iterations without any weights assigned are used as initial condition. The real plant is used for the test.

<table>
<thead>
<tr>
<th>Weights</th>
<th>$r_{\text{mean}}$</th>
<th>$s\text{ens}_{\text{mean}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3 $\text{error}_{yc}$</td>
<td>0.0979</td>
<td>0.1017</td>
</tr>
<tr>
<td>0.3 $\text{error}_{sem}$</td>
<td>0.0699</td>
<td>0.0969</td>
</tr>
</tbody>
</table>
Fig. 21. Neural plant. Reference input (solid line) and plant output (dotted line). 3 delays, 200,000 iterations, $r_{\text{mean}} = 0.0255$. Diagnostic controller output assigned a weight $\omega_1 = 0.3$. Ramp failure.

Fig. 22. Neural plant. Sensor failure (solid line) and diagnostic output (dotted line). 3 delays, 200,000 iterations, $sens_{\text{mean}} = 0.0488$. Diagnostic controller output assigned a weight $\omega_1 = 0.3$. Ramp failure.
Fig. 23. Real plant. Reference input (solid line) and plant output (dotted line). 3 delays, 200,000 iterations, \( r_{\text{mean}} = 0.0719 \). Diagnostic controller output assigned a weight \( \omega_1 = 0.3 \). Ramp failure.

Fig. 24. Real plant. Sensor failure (solid line) and diagnostic output (dotted line). 3 delays, 200,000 iterations, \( s_{\text{sens,mean}} = 0.0870 \). Diagnostic controller assigned a weight \( \omega_1 = 0.3 \). Ramp failure.
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 both parts of this section. Again, for all the experiments in this section, the weights had the same starting values.

5. Conclusions

In this paper, a new approach to implement 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 implementation of the CDC controller was presented 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 it is possible to reproduce the failure. This approach offers an alternative to examining every time whether a specific signal has exceeded an acceptable limit-bound, in which case an alarm is triggered. This is true even in the presence of noise introduced in the system, which accounts for plant modelling errors.

Although the interest is usually in sensor failures, here a unified approach for both sensor and actuator failures has been introduced for the design of a controller, which is capable of reproducing failures at the actuator as well. The actuator case seems to be harder to train, first because the failure signals are filtered through the plant and also because of the intervention of the actuator failures in the path of the back-propagating errors. Note that the latter 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, where the term real plant corresponds to the difference equation that describes it. This was done, in order to see how much the introduction of the neural model of the plant, which was essential for our training needs, could change the performance of our system. The results for the neural model were generally better than the ones with the real model. This was expected, since it was the neural model that was used for the training of the controller. However, as we can see from all the tests, the behavior of the system with the real plant is not significantly different from the behavior with the neural network model of the plant, which implies that our neural model was a good approximation of the real plant.

Since both control and diagnostic objectives characterize our 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. It should be noted that the weighting of the priority for tracking and failure reproduction
is being decided, when the controller is designed, and this decision has to be
done in advance, whether the system should put emphasis on tight control or the
ability to reproduce the failure. In this first papers describing the 4-parameter
controller theory and applications, no priority at all can be established. This
is another reason why our neural network configuration is advantageous, since
we can establish this priority any way we want. This is quite a useful feature,
since the requirements are different for each specific application, and therefore
in 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.

It should also be mentioned that for all our experiments, the assumption was
made that there is no failure occurring at the neural controller. Since the controller
is designed and tested by the designer, there is typically low probability of such
failure. Besides, our approach follows the guidelines of FDI problems, where
the sources of failure come typically from the actuator, the sensor and the plant.

An issue that should be addressed in the future concerns the design of a neural
controller that could provide diagnostic information, when both sensor and actu-
ator 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. Extended simulation efforts showed
that our neural network configuration can not distinguish sensor from actuator
failures, when both occur at the system. Specifically, although the control part,
that is the tracking of the reference input, worked comparatively well, the re-
production of both actuator and sensor failures occurring simultaneously at the
system was not successful. This came as no surprise, in view of the fact that
similar efforts in the literature had not succeeded either; for instance the method-
ology of [9] works only for one type of failure, specifically sensor failures. The
reproduction of simultaneous sensor and actuator failures 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.

Finally, we should state the disadvantages of the proposed methodology, which
can also be viewed as issues and directions for future research. First, we can
not handle simultaneous sensor and actuator failures, as discussed above. Train-
ing time can also be a disadvantage. We have not derived stability theoretical
results and we have not considered plant failures. We should stress, however,
that stability proofs are an open issue in the literature, even when only control specifications exist for the system. We should also state that unless a reference signal or failure of magnitude much greater than those used in training is applied, our system can meet the desired specifications and has been successfully tested in numerous simulations, avoiding any instabilities.

References


