**Proceedings of the 17th World Congress**
**The International Federation of Automatic Control**
**Seoul, Korea, July 6-11, 2008**

# Adaptive Neural-based Fault Tolerant Control for Nonlinear Systems ⋆

**M.J. Fuente, V. Mateo, G.I. Sainz, S. Saludes** *

* *Department of Systems Engineering and Control, University of Valladolid, Science Faculty. Prado de la Magdalena s/n. 47011 Valladolid. Spain. (e-mail: maria@autom.uva.es, gresai@eis.uva.es, sersal@cartif.es)*

**Abstract:** A neural network based fault tolerant control for unknown nonlinear systems is proposed. The faultless system is controlled by a nonlinear Internal Model Controller (IMC), where both the direct and inverse models of the plant are carried out by neural networks. Using the residual signal generated from the fault detection path, an extra neural-network fault compensation loop is introduced. This neural network is a two layer perceptron and the weights and bias are updated on-line by the modified-gradient approach, which tries to minimize the control error induced by the fault. In this context, a fault tolerant control scheme is obtained. This scheme is tested in simulation in a pH plant with good results.

Keywords: Fault tolerant control, adaptive nerual networks, IMC Control, non-linear systems, pH plant

## 1. INTRODUCTION

It is well known that every control system will inevitably be subjected to faults which can be caused by actuators, sensors or system faults. Therefore, how the system is kept in a stable and acceptable control performance when a failure occurs is an important issue in control-systems design. This leads to the fault tolerant control (FTC) (Blanke et al., 2003).

Currently, in most real industrial systems FTC are realized by hardware redundancy. For example, the majority-voting scheme is used with redundant sensors to cope with sensor faults (Patton et al., 2000). However, due to two main limitations of the hardware redundancy, high cost and taking up more space, solutions using analytical redundancy have been investigated over the last two decades. There are generally two different approaches using analytical redundancy: (1) passive approaches, and (2) active approaches.

Passive approaches use robust control techniques to design closed-loop systems so that there are insensitive to certain faults (Niemann and Stoustrup, 2004). Active approaches use on-line fault detection information and reconfigurable controllers. When a fault is detected using analytical or hardware redundancy, the controller is reconfigured to guarantee the post-fault stability and maintain acceptable performance. Active FTC has been investigated using different methods including feedback linearization (Ochi, 1993), model following control (Morse and Ossman, 1990) or state feedback using the pseudo-inverse method (Staroswiecki, 2005). However these studies were based on linear models so they are not suitable for non-linear processes.

More recently, it was reported that neural networks had been employed to tackle FTC problems for nonlinear systems. So, Wang and Wang (1999) use a neural network in a FTC system to form an additional feedback loop, which was used to compensate for the degradation of system performance caused by component faults. Polycarpou and Helmicki (1995) use an on-line approximator (that could be a neural network, a fuzzy system, etc.) that is adapted when a fault occurs by a learning scheme based on the Lyapunov theory. In the event of a failure the control law in normal conditions is augmented with an additional term, which takes into account the on-line approximator and a term to assure the system's stability. Saludes and Fuente (1999) use a predictive controller with a non-linear model calculated by a recurrent neural network to control a reactor tank, after that, when a fault in a sensor is detected, they substitute the fault measurement by the output of the recurrent neural network. Finally, Yu et al. (2005) use a multi-layer perceptron network as the process model which is adapted on-line using the extended Kalman filter to learn changes in the process dynamics. Then, the inversion of the neural model is used as a controller to maintain stability and control performance after a fault occurrence.

In this paper, a new FTC approach for unknown non-linear dynamic systems is proposed. The approach uses a nonlinear Internal Model Controller (IMC) as faultless controller, in which a backpropagation network is used for the construction of the plant model, and its inverse has been implemented by means of an Elman neural network (Elman, 1990). Both are used directly within the IMC control structure. Besides this, a compensation loop is introduced when a fault is detected in the system. This

⋆ This work was supported by the Education and Science Ministry of Spain through the project DPI2006-15716-C02-02 and regional government of Castilla y Leon through the project VA052A07.

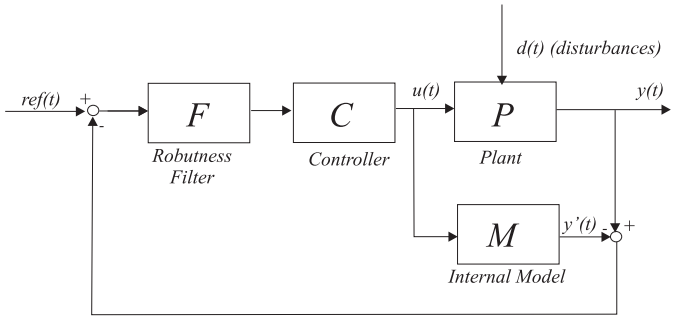



Fig. 1. Structure of the IMC control scheme

loop is implemented by a two layer neural network, whose weights and bias are updated on-line using a modified-gradient algorithm in terms to reduce the control error induced by the fault. Results for a pH plant, i.e., a nonlinear system are presented.

This paper is organized as follows. In section 2 the nonlinear IMC control scheme is described. In section 3 the fault-tolerant scheme is presented. The next section describes the application of this methodology to the pH plant, with a description of the plant, of the fault-tolerant control design system, and the results when different faults have occurred in the plant. Finally, section 5 concludes the paper.

## 2. THE NONLINEAR IMC

The basic idea of the linear Internal Model Control (IMC) is illustrated in Figure 1. The key characteristic of this control design approach is the inclusion of a plant model within the control structure. If the model is a perfect representation of the process, the influence of the process output on the feedback signal vanishes. The feedback signal then only carries the influence of disturbances. However, in practice, a model is not a perfect description of the plant. The feedback signal then combines the model error with the disturbances. Based on this structure, perfect control is obtained if the controller, $C$, is chosen as the inverse of the internal model, $M$. IMC controllers have been extensively studied in the case of linear modeling of the process (Morari and Zafirou, 1989), and have been shown to have good robustness properties against disturbances and model mismatch.

The nonlinear version of the IMC is analogous to the linear one. The difference is that the model of the plant, $M$, and the controller, $C$, are nonlinear functions, as neural networks (Rivals and Personnaz, 2000). The system $F$ in Fig. 1 is a linear filter that can be designed in order to reach some desired robustness or tracking properties in the closed loop system.

In this paper the plant model is carried out by a back-propagation neural network. This network is trained in the classical way, i.e., the error signal used to adjust the net weights is the difference between the plant output and the network output. Thus, the net is forced towards learning the plant dynamics. The second step is to obtain a plant inverse model, which is calculated by training an Elman neural net. Here, the plant model (obtained in the first learning step) is used in the inverse learning architecture rather than the plant itself. For inverse modeling, the error signal used to adjust the network is defined as the difference between a synthetic signal (the desired network output) and the network output. This tends to force the transfer function between the reference and the output of the model to unity; i.e., the network being trained is forced to represent the inverse model of the plant model.

## 3. FAULT TOLERANT CONTROL

A fault tolerant controller should be able to maintain the closed-loop operation to a certain degree in presence of faults. This can be achieved by minimizing the difference between the reference and the output of the system when a fault occurs. For this, when a fault is detected the control input $u_1(t)$ calculated by the nominal controller (IMC) is not enough to compensate the fault effect, and in this case a new control signal $u_c(t)$ is calculated by the compensation loop introduced in this paper, for the total control signal $u(t) = u_1(t) + u_c(t)$ to be able to maintain the control objectives. With this fault compensation loop, the structure of the closed loop system can now be obtained as shown in Fig. 2, where the FDI module should be some fault detection and identification algorithm used to detect the faults. In this form the value of the compensator, $u_c(t)$ is zero until the FDI algorithm detects a fault.

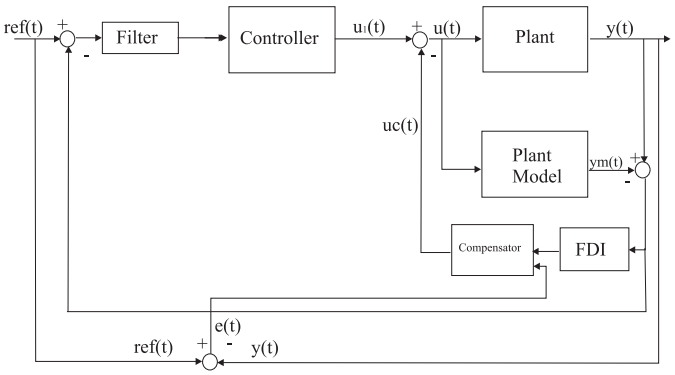



Fig. 2. Structure of the nonlinear fault-tolerant controller

A two-layer perceptron network is used as the compensator; this leads to the input and the output relationship of the selected neural network compensator being given by:

$$u_c = \sum_{j=1}^{h+1} \nu_j \tilde{\varphi}\left(a_j\right) \tag{1}$$

$$a_j = \sum_{i=1}^{n+1} w_{i,j} \tilde{e}_i \tag{2}$$

where $\tilde{e} = [e_1, e_2, \ldots, e_n, 1]$, and $e_i = e(t-i+1)$ the inputs to the neural network, with $e(t) = ref(t) - y(t)$ the control error, i.e., the difference between the reference and the plant output. $u_c(t)$ is the output of the neural network compensator, and $\nu$ and $w$ are the weights. $n$ and $h$ are the number of neurons in the input and hidden layer, respectively, and

$$\tilde{\varphi}(a_j) = [\varphi(a_1), \varphi(a_2), \ldots, \varphi(a_h), 1] \tag{3}$$

The activation function can be each nonlinear function, but in this case it is represented in eq. (4) and its derivative is in eq. (5):

$$\varphi(x) = \frac{2}{1+e^{-2x}} - 1 \tag{4}$$

$$\frac{d\varphi(x)}{dx} = \dot{\varphi}(x) = \frac{4e^{-2x}}{(1+e^{-2x})^2} \tag{5}$$

### *3.1 On-line training*

The gradient-based updating rule for the parameters $w$ and $\nu$ can be expressed as:

$$w_{i,j}(t) = w_{i,j}(t-1) - \eta \frac{\partial E(t)}{\partial w_{i,j}(t)} \tag{6}$$

$$\nu_i(t) = \nu_i(t-1) - \eta \frac{\partial E(t)}{\partial \nu_i(t)} \tag{7}$$

where $\eta$ is the learning rate and $E(t)$ is a function of the tracking error ($e(t) = ref(t) - y(t)$) defined as:

$$E(t) = \frac{1}{2}\sum_{k=1}^{t} e(k)^2 \tag{8}$$

The derivatives that appear in eqs. (6) and (7) can be calculated using the chaining rule as:

$$\frac{\partial E(t)}{\partial \nu_i(t)} = \frac{\partial E(t)}{\partial e(t)}\frac{\partial e(t)}{\partial y(t)}\frac{\partial y(t)}{\partial u(t)}\frac{\partial u(t)}{\partial u_c(t)}\frac{\partial u_c(t)}{\partial \nu_i(t)} \tag{9}$$

$$\frac{\partial E(t)}{\partial w_{ij}(t)} = \frac{\partial E(t)}{\partial e(t)}\frac{\partial e(t)}{\partial y(t)}\frac{\partial y(t)}{\partial u(t)}\frac{\partial u(t)}{\partial u_c(t)}\frac{\partial u_c(t)}{\partial w_{ij}(t)} \tag{10}$$

It is possible to solve all the partial derivatives in eqs. (9) and (10) easily, except for $\frac{\partial y(t)}{\partial u(t)}$, the plant sensitivity, which is unknown if a model of the process is not available. But this equation can be substituted by its sign, which is the sign of the process gain (Cui and Shin, 1993). These derivatives are:

$$\frac{\partial E(t)}{\partial e(t)} = e(t) \tag{11}$$

$$\frac{\partial e(t)}{\partial y(t)} = -1 \tag{12}$$

$$\frac{\partial y(t)}{\partial u(t)} = sign(K) \tag{13}$$

$$\frac{\partial u(t)}{\partial u_c(t)} = 1 \tag{14}$$

Finally, for the calculation of the other derivatives, it is also possible to use the chaining rule:

$$\frac{\partial u_c}{\partial \nu_i} = \begin{bmatrix} \varphi\left(\sum_{i=1}^{n+1} w_{i,1}\tilde{e}_i\right) \\ \varphi\left(\sum_{i=1}^{n+1} w_{i,2}\tilde{e}_i\right) \\ \vdots \\ \varphi\left(\sum_{i=1}^{n+1} w_{i,h}\tilde{e}_i\right) \\ 1 \end{bmatrix} \tag{15}$$

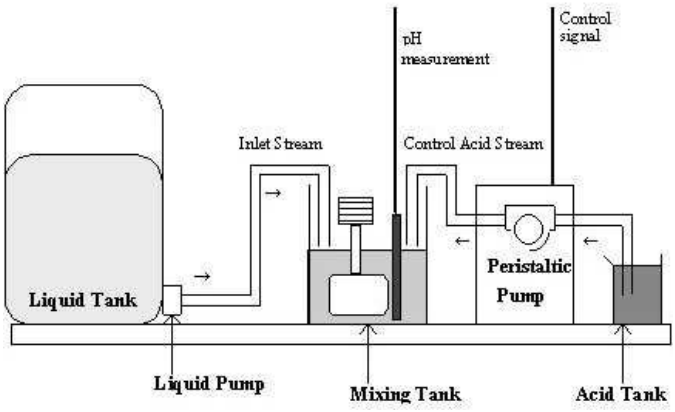



Fig. 3. Laboratory plant

$$\frac{\partial u_c}{\partial w_{i,j}} = \begin{bmatrix} \nu_1 e_1 \dot{\varphi}\left(\sum_i w_{i,1}\tilde{e}_i\right) & \dots & \nu_h e_1 \dot{\varphi}\left(\sum_i w_{i,h}\tilde{e}_i\right) \\ \nu_1 e_2 \dot{\varphi}\left(\sum_i w_{i,1}\tilde{e}_i\right) & \dots & \nu_h e_2 \dot{\varphi}\left(\sum_i w_{i,h}\tilde{e}_i\right) \\ \vdots & \ddots & \vdots \\ \nu_1 e_n \dot{\varphi}\left(\sum_i w_{i,1}\tilde{e}_i\right) & \dots & \nu_h e_n \dot{\varphi}\left(\sum_i w_{i,h}\tilde{e}_i\right) \\ \nu_1 \dot{\varphi}\left(\sum_i w_{i,1}\tilde{e}_i\right) & \dots & \nu_h \dot{\varphi}\left(\sum_i w_{i,h}\tilde{e}_i\right) \end{bmatrix} \tag{16}$$

All these equations can be calculated in real time, because of the reduced size of the network. The most important parameter here is the learning rate, $\eta$. With large values of this parameter, the speed of the convergence increases, but the system can become unstable. With low values of $\eta$ the behavior is better but the response is slower. In this case the parameter $\eta$ is variable depending on the control error, i.e., with large errors the learning rate is faster, and with low regulation errors the learning rate is slower.

$$\eta = \gamma |e(t)| \tag{17}$$

with the $\gamma$ parameter changing in real time, from a very small value, $10^{-6}$ by example, at the beginning of the learning task and it can increase slowly until it reaches the desired behavior.

## 4. APPLICATION OF THE FTC SCHEME TO THE PH PLANT

### *4.1 Description of the experiment setup*

In this study the process is to modify the pH value of an aqueous solution titrated with hydrochloric acid ($HCl$) in a continuous stirred tank reactor (CSTR). The experimental setup is shown in Figure 3. An overflow system (not shown) is applied on the CSTR; therefore the volume can be considered constant. The control variable $u$ is the flowrate of the titrating stream which is feed using a peristaltic pump (ISMATEC MS-1 REGLO/6-160). The output variable $y$ is the hydrogen ions in the effluent stream, measured as pH. The pH mixture is measured using an Ag-AgCl electrode (Kent 1180/700) and transmitted using a pH-meter (Kent EIL9143).

### *4.2 Plant model*

The model of the plant has been obtained based on first principles, and then validated in the real plant, by carrying

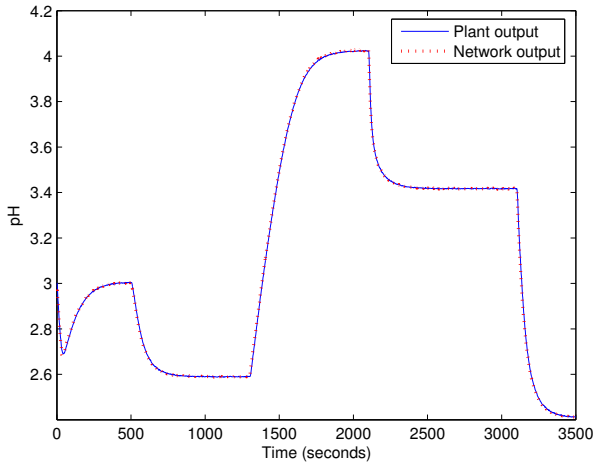



Fig. 4. Plant and neural network outputs for the validation data set, i.e., the direct model $M$.

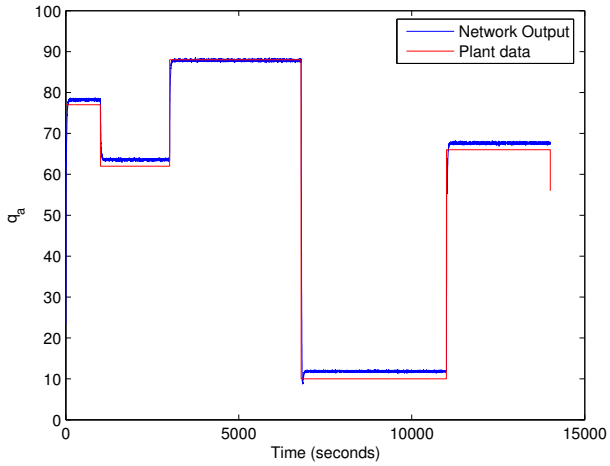



Fig. 5. Acid flow and neural network output for the validation data set, i.e., the inverse model $C$.

out experiments at different working points. Assuming that the liquid is water, the hydrochloric acid has constant concentration, the temperature is constant (25C) and there are perfect dissolution, mixing and no additional buffering effects, the following model can be obtained:

$$\begin{aligned} \frac{dMN_d}{dt} &= q_a N_a - (q_a + q_0)N_d \\ pH^* &= -log(N_d) - \frac{dpH^*}{dt} + w \end{aligned} \tag{18}$$

where $q_a$ is the control acid stream flowrate, $q_0$ the inlet water flowrate, $N_d$ the acid concentration in the solution, $N_a$ the acid concentration in the control acid stream, the measured pH is $pH^*$, which is affected by a time constant $\tau$ and w is the noise.

### *4.3 Nonlinear IMC application*

The three elements that conform the IMC controller are the internal model, $M$, the inverse model, $C$ and the robustness filter F. In this paper the nonlinear plant model used is a multi-layer perceptron network, with five neurons in the hidden layer and trained with the backpropagation algorithm. The output of the network is $pH(t)$ and the inputs are:

$$\mathcal{E}_M(t) = \{q_a(t), ..., q_a(t-2), pH(t-1), ..., pH(t-3)\} \tag{19}$$

Figure 4 shows the validation results obtained after training. It can be seen that the model follows faithfully the plant output.

The controller, i.e., the inverse model of the plant is calculated with an Elman neural network, whose output is $q_a(t)$ and the inputs are presented in eq. (20). This network has been trained using the plant outputs as inputs in the training set and plant inputs become the outputs in the training set.

$$\mathcal{E}_C(t) = \{pH(t), ..., pH(t-7)\} \tag{20}$$

The results for validation data set are shown in Figure 5. Finally the filter, $F$ used in the IMC structure (Fig. 1), is:

$$F(z) = \frac{Tz}{z - (1-T)} \tag{21}$$

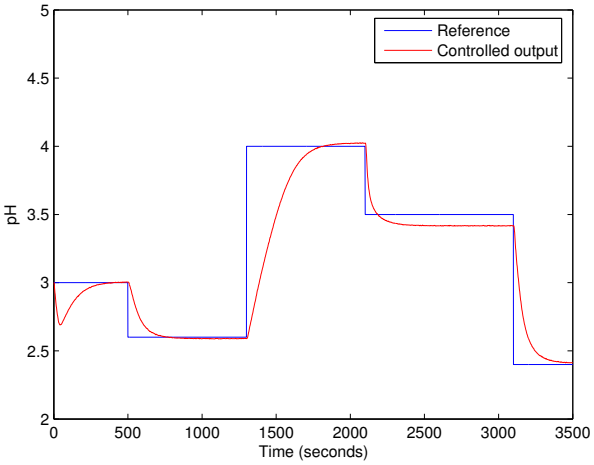



Fig. 6. Controlled plant output in absence of faults and perturbations

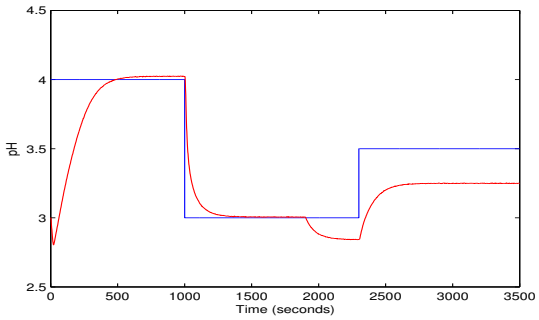
where $T = 1 - \alpha$ and, for ensuring stability, $0 < \alpha < 1$ must be held. The value that has been chosen is $\alpha = 0.9$.

Some control experiments have been carried out in the pH plant, to show the results of the IMC controller. Fig. 6 shows the results when the reference changes over all the operation range of the system.

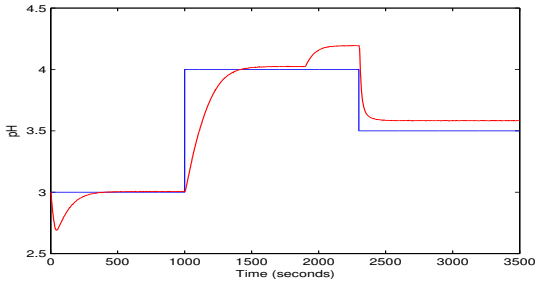
### *4.4 Fault-tolerant control in the pH plant*

Two different fault types have been considered in this work, first a fault in the actuator, the pump that feeds the acid flow in the system, and second a parametric fault is considered, when the flow of water at the input of the plant changes considerably, this flow is not measured in the plant, and it can be considered as an internal fault.

First, Figure 7 shows the output of the plant when a fault in the actuator is simulated at $t = 1900s$ and any accommodation is carried out in the plant. As it is possible to see, the fault causes the output of the plant to go outside the reference, because the nominal controller is not able to compensate the fault, and it is not able to continue




(a) Fault in the actuator.




(b) Fault in the inlet water flowrate.

Fig. 7. Plant output when different faults are simulated and any accommodation is carried out.

controlling the plant adequately. Also, if the fault occurs in the inlet water flow, the controller is not able to maintain the output tracking the reference, as it is possible to see in the same figure.

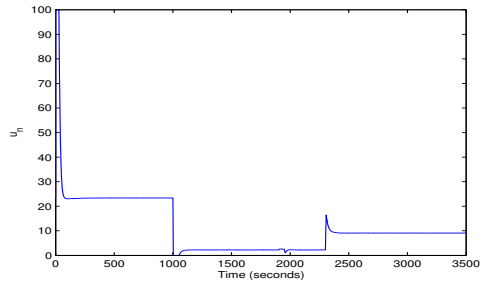
In this approach, when a fault is detected, the nominal controller is modified by the adding of a compensation signal, i.e., the new control signal is calculated as: $u(k) = u_1(k) + u_c(k)$. In order to calculate this compensation signal, $u_c(t)$, a neural network with two layers is used, this network has two inputs, 5 neurons with the hyperbolic tangent as the activation function in the hidden layer and one neuron with linear function in the output layer. This network has been trained on-line with the algorithm shown in section 3.1. The inputs to the network are:

$$\tilde{e}(t) = \{e(t), e(t-1)\} \tag{22}$$

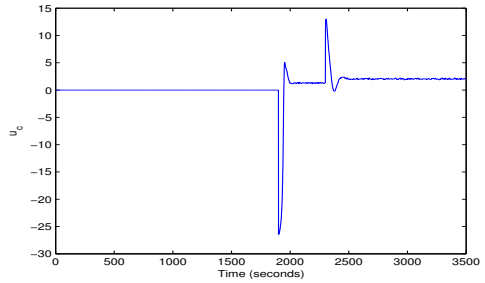
with $e(t) = ref(t) - y(t)$, i.e., the tracking error.

Some experiments have been carried out to show the results of the fault tolerant controller designed here. Figure 8 shows the results when a fault in the input flow of water to the plant is simulated at $t = 1900s$. First the nominal control signal $u_1(k)$ is shown, second the compensator signal $u_c(k)$, that is zero until the fault is detected, third the real control signal that is introduced into the plant is shown ($u(k) = u_1(k)+u_c(k)$), and finally the output of the system, which changes at the time that the fault occurs, i.e, there is an overshot due to the fault, but the compensator procedure is able to accommodate the fault rapidly and after that the plant output follows the reference at all instants.

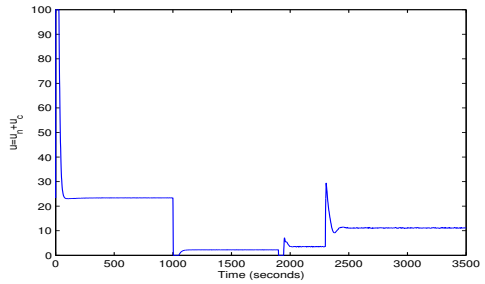
Fig. 9 presents the results when a step fault in the actuator is simulated at $t = 1900s$. As before, it is possible to see that the neural network compensator is able to minimize the tracking error in a few instants of time, and the



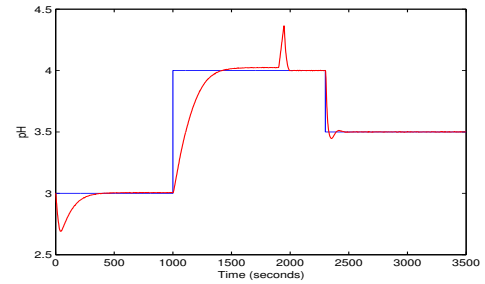

(a) Nominal control signal: $u_1(t)$.




(b) Compensation control signal: $u_c(t)$.




(c) Control signal: $u(t) = u_1(t) + u_c(t)$.




(d) System output.

Fig. 8. Plant output and control signals when a fault is simulated in the inlet water flowrate and the accommodation is carried out.

system can accommodate the fault. The same graphic also shows the behavior of the nominal control signal, $u_1(t)$, the compensator signal, $u_c(t)$ and the control signal $u(t)$.

Also, if both faults are simulated simultaneously, the first one, i.e., the fault in the inflow of water at $t = 1700s$, and the second one, the fault in the actuator at $t = 3200s$ after a change in the reference, the FTC is able to maintain the output of the system in the reference, as can be seen in Figure 10, where only the final control signal ($u(t) = u_1(t) + u_c(t)$) and the plant output are shown. Since time $t = 3200s$ the two faults are presented simultaneously in the system, and the compensator loop is

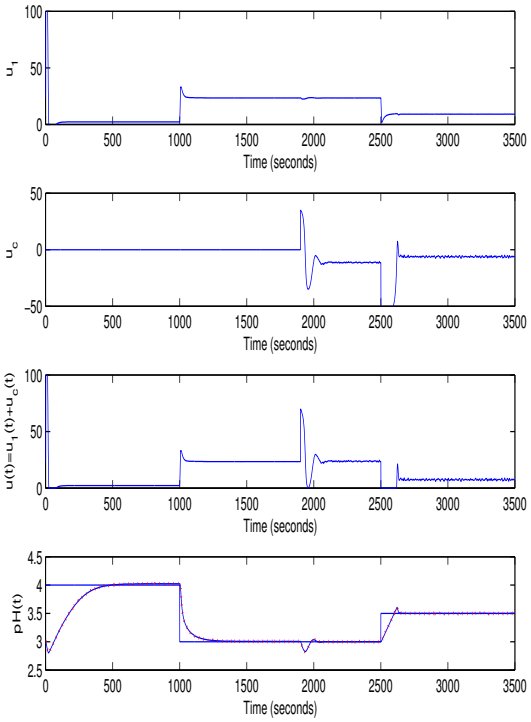



Fig. 9. Plant output and control signals when a fault is simulated in the actuator and the accommodation is carried out.

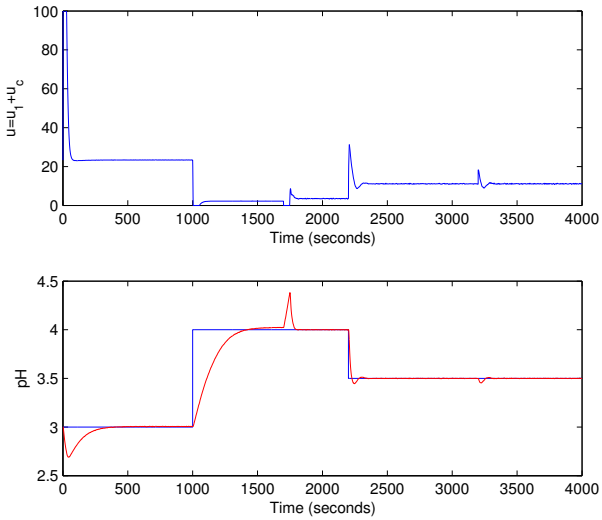



Fig. 10. The plant output and the control signal $u(t)$ when the two faults are simulated simultaneously and the accommodation is carried out.

able to compensate both faults and to maintain the control objective.

## 5. CONCLUSIONS

This paper has dealt with fault-tolerant control in the context of IMC control, but this approach can be generalized to every possible control strategy. This paper proposes a compensator which, when a fault is detected, generates a new control signal that, added to the nominal control signal, calculates the control signal to be introduced into the plant. This compensator is implemented by a neural network trained on-line using a modified-gradient algorithm to reduce the control error in the system. This compensator does not need any a priori knowledge of the plant after the fault, only need the tracking error, i.e., the difference between the reference and the output of the plant generated by the fault. This fault tolerant control has been proved in a pH plant, with different types of faults, actuator and internal faults with good results. Now, the next step is to prove the method in the real plant.

## REFERENCES

M. Blanke, M. Kinnaert, J. Lunze and M. Staroswiecki. *Diagnosis and Fault Tolerant Control*. Springer, 2003.

X. Cui and K. G. Shin. Direct control and coordination using neural networks. *IEEE Transactions on Systems, Man and Cybernetics*, 23(3):686–697, Mayo/junio 1993.

J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

M. Morari and E. Zafirou. *Robust Process Control*. Prentice–Hall, 1989.

W.D. Morse and K.A. Ossman. Model–following reconfigurable flight control system for the AFTI/F–16. *Journal of Guidance, Control and Dynamics*, 13(6):969–976, 1990.

H. Niemann and J. Stoustrup. Passive fault tolerant control of a double inverted pendulum - a case study. *Control Engineering Practice*, 13:1047–1059, 2004.

Y. Ochi. Application of feedback linearisation method in a digital restructurable flight control system. *Journal of Guidance, Control and Dynamics*, 16(1):111–117, 1993.

J.R. Patton, P.M. Frank, and R. Clark. *Fault Diagnosis in Dynamic Systems: Theory and Application*. Prentice Hall, Enlewood Cliffs, NJ, 2000.

M.M. Polycarpou and A.J. Helmicki. Automated fault detection and accommodation: A learning systems approach. *IEEE Trans. on Systems, Man and Cybernetics*, 25(11):1447–1458, 1995.

I. Rivals and L. Personnaz. Nonlinear internal model control using neural networks: Application to process with delay and design issues. *IEEE Trans. on Neural Networks*, 11:80–90, 2000.

S. Saludes and M.J. Fuente. Neural networks-based fault detection and accommodation in a chemical reactor. In *14th IFAC World Congress*,169–174, Beijin, China, 1999.

M. Staroswiecki. Fault tolerant control: The pseudo-inverse method revisited. In *16th IFAC World Congress*, Prague, Czech Republic, 2005

H. Wang and Y. Wang. Neural–network–based fault–tolerant control of unknown nonlinear systems. *IEE Proceedings – Control Theory and Applications*, 146(5): 389–398, Septiembre 1999.

D.L. Yu, T.K. Chang, and D.W. Yu. Adaptive neural model.based fault tolerant control for multi–variable processes. *Engineering Applications of Artificial Intelligence*, 18:393–411, 2005.