Combining Artificial Intelligence and Advanced Techniques in Fault-Tolerant Control

A. Vargas-Martínez*[1], L. E. Garza-Castañón[2]

[1,2] Tecnológico de Monterrey, Campus Monterrey
Av. E. Garza Sada # 2501, 64849, Monterrey, México
e-mail: A00777924@itesm.mx, legarza@itesm.mx

ABSTRACT
We present the integration of artificial intelligence, robust, nonlinear and model reference adaptive control (MRAC) methods for fault-tolerant control (FTC). We combine MRAC schemes with classical PID controllers, artificial neural networks (ANNs), genetic algorithms (GAs), H∞ controls and sliding mode controls. Six different schemas are proposed: the first one is an MRAC with an artificial neural network and a PID controller whose parameters were tuned by a GA using Pattern Search Optimization. The second scheme is an MRAC controller with an H∞ control (H∞). The third scheme is an MRAC controller with a sliding mode controller (SMC). The fourth scheme is an MRAC controller with an ANN. The fifth scheme is an MRAC controller with a PID controller optimized by a GA. Finally, the last scheme is an MRAC classical control system. The objective of this research is to generate more powerful FTC methods and compare the performance of above schemes under different fault conditions in sensors and actuators. An industrial heat exchanger process was the test bed for these approaches. Simulation results showed that the use of Pattern Search Optimization and ANNs improved the performance of the FTC scheme because it makes the control system more robust against sensor and actuator faults.

Keywords: Fault-tolerant control, MRAC, ANN, PID, GA, H∞, SMC.

RESUMEN
Se presenta la integración de métodos de Inteligencia Artificial, Control Robusto, Control No lineal y Control Adaptable por Modelo de Referencia (MRAC) en el Control Tolerante a Fallas (FTC). Se combinan diferentes esquemas de MRAC con controladores PID clásicos, redes neuronales, algoritmos genéticos, controladores H∞ y controladores por modo deslizante. Se proponen seis diferentes esquemas: el primer esquema es un MRAC con una red neuronal y un PID cuyos parámetros fueron optimizados con un algoritmo genético utilizando Optimización por Búsqueda de Patrones. El segundo esquema es un controlador MRAC con un controlador H∞. El tercer esquema es un controlador MRAC con un controlador por modo deslizante. El cuarto esquema es un controlador MRAC con una red neuronal. El quinto esquema es un controlador MRAC con un controlador PID optimizado por un algoritmo genético. Finalmente, el último esquema es un controlador clásico MRAC. El objetivo de esta investigación es generar métodos de FTC más poderosos y comparar su desempeño bajo diferentes condiciones de fallas tanto en sensores como en actuadores. Se utilizó un intercambiador de calor industrial para realizar dichos experimentos. Los resultados obtenidos de la simulación demostraron que el uso de Optimización por Búsqueda de Patrones con redes neuronales mejoró el desempeño del esquema FTC, ya que el sistema de control se volvió más robusto ante fallas en sensores y en actuadores.

1. Introduction

Nowadays, an increasing demand on products quality, system reliability, and plant availability has allowed engineers and scientists to give more attention to the design of methods and systems that can handle certain types of faults. In addition, the global crisis creates more competition between industries and plant shutdowns are not an option because they cause production losses and consequently lack of presence in the markets; primary services such as power grids, water supplies, transportation systems, and communication and commodities production cannot be interrupted without putting at risk human health and social stability. On the other hand, modern systems and challenging operating conditions (i.e. fully integrated processes) increase the possibility of system failures which can cause the loss of human lives and damages in equipments; also, some dangerous environments in nuclear or chemical plants, set restrictive limits to human work. In all these environments the use of automation and intelligent systems is fundamental to minimize the impact of faults. For those reasons, fault-tolerant control (FTC) methods have been proposed, with the most important benefit being the plant to continue operating in spite of a fault, no matter if the process has certain degradation on its performance. This strategy prevents a fault from developing into a more serious failure.

Although, most of the FTC methods that have been developed are based on the classical control theory [1], the use of artificial intelligence (AI) in FTC has emerged recently [2] [3] [4]. Classical artificial intelligence (AI) approaches such as artificial neural networks (ANNs), fuzzy logic (FL), ANN-FL and genetic algorithms (GAs) offer an advantage over traditional methods used by the control community such as state observers, statistical analysis, parameter estimation, parity relations and residual generation [5] [6] because AI approaches can reproduce the behavior of nonlinear dynamical systems with models extracted from data. Also, on the AI field there are many learning processes that improve the FTC performance. This is a very important issue in FTC applications on automated processes, where information is easily available, or processes where accurate mathematical models are hard to obtain.

ANNs have been applied in FTC because they are helpful to identify, detect and accommodate system faults. ANNs are used as fault detectors by estimating changes in process models dynamics [3] [7], such as process controllers [8], and performing both functions: fault detection and control [9] [10].

Recently genetic algorithms have been applied in fault-tolerant control as a strategy to optimize the controlled system in order to accommodate system failures. For instance, in [11] a FTC approach using multi-layer ANNs with a GA was presented. The proposed FTC scheme uses hardware redundancy and weight retraining based on a GA in order to reconfigure the ANN to accommodate the fault. The objective of the genetic algorithm is to optimize the weights of ANNs to reduce the error between the actual output and the desired output.

On the other hand, advanced techniques from robust control such as $H\infty$, and non-linear control, such as sliding mode control, have also been applied to FTC with encouraging results. For example, in [12] an $H\infty$ FTC approach was used against sensor failures for uncertain descriptor systems (systems which capture the dynamical behavior of natural phenomena). Besides, in [13] an FTC system that includes a sliding mode control and an artificial neural network was presented, this system is able to control the desired trajectories tracking problems when a fault is present.

The main objective of this paper is to improve the fault tolerance capabilities of an MRAC structure by aggregating known and powerful control strategies to it. Different FTC schemes which combine model reference adaptive control (MRAC) with AI techniques such as artificial neural networks, genetic algorithms, and methods from nonlinear control (sliding mode control) and robust control ($H\infty$ Control Theory) are developed. A total of six different schemes for fault tolerant control proposed: the first scheme is a model reference adaptive control with an artificial neural network and a PID controller optimized by a genetic algorithm (MRAC-PID-ANN), the second method is a combination of a model reference adaptive controller and an $H\infty$ controller (MRAC-$H\infty$), the third scheme is a model reference adaptive controller with a sliding mode controller (MRAC-SMC), the fourth method is a model reference adaptive controller with an artificial neural network (MRAC-ANN), the fifth method is a combination of a model reference adaptive controller with a PID controller optimized by a genetic algorithm (MRAC-PID), and finally, the last scheme is a classical model reference adaptive controller (MRAC).

This paper is organized as follows: Section 2 describes the article background, Section 3 shows the proposed schemes, Section 4 explains the experiments and results, in Section 5 a qualitative comparison with similar approaches is made, and finally, Section 6 addresses the conclusions.

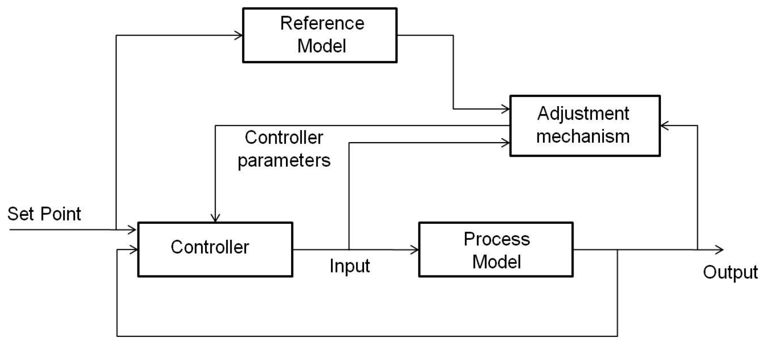


Figure 1. Model Reference Adaptive Controller (MRAC) general scheme [14].

2. Background

2.1 Model reference adaptive control

The MRAC, shown in Figure 1, implements a closed-loop controller that involves the parameters that should be optimized in order to modify the system response to achieve the desired final value. The adaptation mechanism adjusts the controller parameters to match the process output with the reference model output. The reference model is specified as the ideal model behavior that the system is expected to follow

The controller error is calculated as follows:

$$e = y_{process} - y_{reference} \tag{1}$$

where *e*, $y_{process}$ and $y_{reference}$ represent the error, process output and reference output, respectively. To reduce the error, a cost function was used, in the form of

$$J(\theta) = \frac{1}{2} e^2(\theta) \tag{2}$$

where *θ* is the adaptive parameter inside the controller.

The function above can be minimized if parameters *θ* change in the negative direction of gradient *J*, this is called the gradient descent method and is represented by

$$\frac{d\theta}{dt} = -\gamma \frac{\partial J}{\partial \theta} = -\gamma \frac{\partial e}{\partial \theta} e \tag{3}$$

where *γ* is the parameter to adjust the speed of learning. The above equation is known as the MIT (Massachusetts Institute of Technology) rule and determines how parameter *θ* will be updated to reduce the error [15]. The implemented MRAC scheme for the heat exchanger process (shown in Figure 2) has two adaptation parameters: adaptive feedfoward gain *(θ_1)* and adaptive feedback gain *(θ_2)*. These parameters will be updated to follow the reference model. The mathematical procedure to design the MRAC system is the following: First, Equation (1) should be transformed in order to include the process model and the reference model with their respective inputs:

$$e = G_p(s) * u - G_{ref}(s) * u_c \tag{4}$$

Where G_p, *u*, G_{ref} and u_c represent the process model, process input, reference model and controller input, respectively. Then, the input is rewritten in terms of the adaptive feedforward (θ_1) and adaptive feedback (θ_2) gain:

$$u = \theta_1 u_c - \theta_2 y_{process} \tag{5}$$

Equation (5) was defined to derive an equation for the process output ($y_{process}$):

$$y_{process} = G_p(s) * u = \left(\frac{b_r}{s^2 + a_{1r}s + a_{0r}} \right) \left(\theta_1 u_c - \theta_2 y_{process} \right) = \left(\frac{b_r \theta_1}{s^2 + a_{1r}s + a_{0r} + b_r \theta_2} \right) u_c \qquad (6)$$

Using Equation (6), the error can be redefined as

$$e = \left(\frac{b_r \theta_1}{s^2 + a_{1r}s + a_{0r} + b_r \theta_2} \right) u_c - \left(G_{ref}(s) * u_c \right) \qquad (7)$$

Therefore, the error partial derivatives with respect to the adaptive feedforward (*θ1*) and adaptive feedback (*θ2*) gain are specified as

$$\frac{\partial e}{\partial \theta_1} = \left(\frac{b_r}{s^2 + a_{1r}s + a_{0r} + b_r \theta_2} \right) u_c \qquad (8)$$

$$\frac{\partial e}{\partial \theta_2} = \left(\frac{b_r{}^2 \theta_1}{\left(s^2 + a_{1r}s + a_{0r} + b_r \theta_2 \right)^2} \right) u_c = -\left(\frac{b_r \theta_1}{s^2 + a_{1r}s + a_{0r} + b_r \theta_2} \right) y_{process} \qquad (9)$$

Consequently, the process characteristic equation can be transformed into Equation (10) because the purpose of the MRAC system is to approximate the process model with the reference model.

$$s^2 + a_{1r}s + a_{0r} + b_r \theta_2 \approx s^2 + a_{1r}s + a_{0r} \qquad (10)$$

Finally, with Equation (10) defined, the error partial derivatives are transformed and, by employing the MIT rule, the update rules for the adaptive feedforward and adaptive feedback gain are written. This can be observed in Equations (11) and (12).

$$\frac{\partial e}{\partial \theta_1} = \left(\frac{a_{1r}s + a_{0r}}{s^2 + a_{1r}s + a_{0r}} \right) u_c \rightarrow \frac{d\theta_1}{dt} = -\gamma \frac{\partial e}{\partial \theta_1} e = -\gamma \left(\frac{a_{1r}s + a_{0r}}{s^2 + a_{1r}s + a_{0r}} u_c \right) e \qquad (11)$$

$$\frac{\partial e}{\partial \theta_2} = -\left(\frac{a_{1r}s + a_{0r}}{s^2 + a_{1r}s + a_{0r}} \right) y_{process} \rightarrow \frac{d\theta_2}{dt} = -\gamma \frac{\partial e}{\partial \theta_2} e = \gamma \left(\frac{a_{1r}s + a_{0r}}{s^2 + a_{1r}s + a_{0r}} y_{process} \right) e \qquad (12)$$

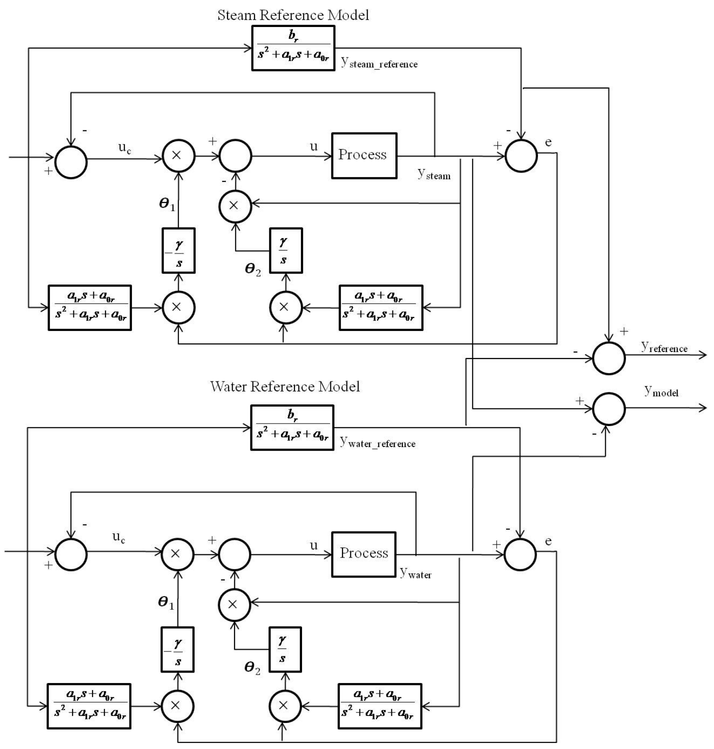
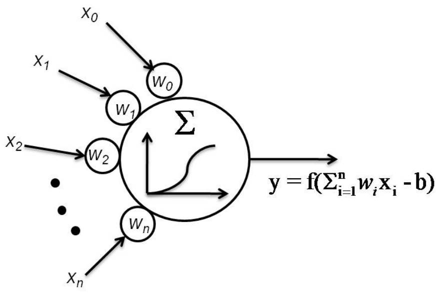


Figure 2. Industrial Heat Exchanger Fault Tolerant MRAC scheme.

2.2 Artificial neural networks

ANNs are mathematical models that try to mimic the biological nervous system. An artificial neuron has multiple input signals x_1, x_2, …,x_n entering the neuron using connection links with specific weights w_1, w_2, …, w_n or $\sum_{i=1}^{n} w_n x_i$ named the net input, and also have a firing threshold b, an activation function f and an output of the neuron that is represented by $y = f\left(\sum_{i=1}^{n} w_n x_i - b\right)$. The firing threshold b or bias can be represented as another weight by placing an extra input node x_0 that takes a value of 1 and has a w_0=-b [16]. This is shown on the left side of Figure 3. On the other hand, an ANN with more than one input layer of neurons, a middle layer called the hidden layer and an output layer is named a multi-layer neural network (Figure 3, right side).



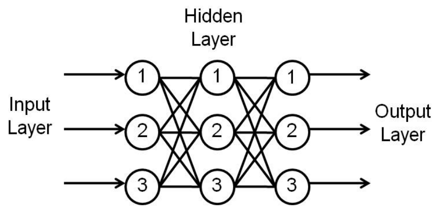


Figure 3. Basic artifical neuron (left side) and general multi-later neural network (right side).

An ANN can have a feedback or a feed-forward structure. In the feedback structure the information can move back and forward. In the feed-forward structure, the information moves only forward from the input nodes through the outputs nodes with no cycles in the network [17].

The ANN needs to be trained from examples, in a process called supervised learning. Once a successfully training is done, the ANN is ready if and only if the networks reproduce the desired outputs from the given inputs. The most common methodology for this kind of learning is the backpropagation algorithm, where the weights of the ANNs are determined by using iteration until the output of the network is the same as the desired output. In addition, unsupervised learning uses a mechanism for changing values of the weights according to the input values, this mechanism is named self-organization. An example of this algorithm is the Hebbian learning algorithm [17].

In this work, to create and train the ANN controller, the original process inputs were introduced as well as the desired outputs. The created ANN is a two-layer feed forward neural network with 20 sigmoid hidden neurons and a linear output neuron. To train the network, the Levenberg-Maquard backpropagation algorithm was used. This training algorithm is a combination of Gauss-Newton and gradient descent methods which integrates the benefits of the global and local convergence properties from the gradient descent and Gauss-Newton methods, respectively. The Levenberg-Marquardt method approaches the Hessian matrix in the form of the product of a Jacobian matrix by its transpose, the same form as the quasi-Newton methods [18].

$$J = \begin{bmatrix} \frac{\partial e_1}{\partial x_1} & \frac{\partial e_1}{\partial x_2} & \cdots & \frac{\partial e_1}{\partial x_N} \\ \frac{\partial e_2}{\partial x_1} & \frac{\partial e_2}{\partial x_2} & \cdots & \frac{\partial e_2}{\partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_n}{\partial x_1} & \frac{\partial e_n}{\partial x_2} & \cdots & \frac{\partial e_n}{\partial x_N} \end{bmatrix} \quad (13)$$

$$H \approx J^T J \quad (14)$$

Then, the gradient can be estimated as the product of the transpose Jacobian matrix by a vector which contains the minimized errors.

$$\nabla = J^T e \quad (15)$$

The combination of the above equations creates a weight-update formula, where *μ* is the control parameter and *I* is an identity matrix.

$$\Delta\omega = -\left(J^T J + \mu I\right)^{-1} J^T e \quad (16)$$

It is important to mention that if *μ* is a large number, the *Δω* equation will be similar to the gradient descent method. On the other hand, if *μ* is zero, the equation will be similar to the Newton method. The next diagram (Figure 4) demonstrates the steps of the Levenberg-Maquard backpropagation algorithm [18].

2.3 Genetic algorithms

GAs are searching and optimizing algorithms motivated by natural selection evolution and natural genetics [19]. Basically, GAs initially generate a random set of solutions (initial population of chromosomes) which improve from iteration to iteration, by changing their features (mutation) and combining with other solutions (crossover). The simplest GAs follow the next steps: they generate a random initial population of chromosomes, they calculate the fitness of every chromosome in the population, they apply selection, crossover and mutation and replace the actual population with the new population until the required solution is achieved. The main advantages of GAs are that they have a powerful computational effect, robustness, fault tolerance, fast convergence to a global optimal, capability of searching in complex landscape where the fitness function is discontinuous; also, they can be combined with traditional optimization techniques (Tabu search) and have the ability to solve problem without needing human experts [17] [19]. The genetic algorithm used in this research is represented in the flow diagram (Figure 5). The parameters used in the genetic algorithm are shown in Table 1. In this work, the PID parameters

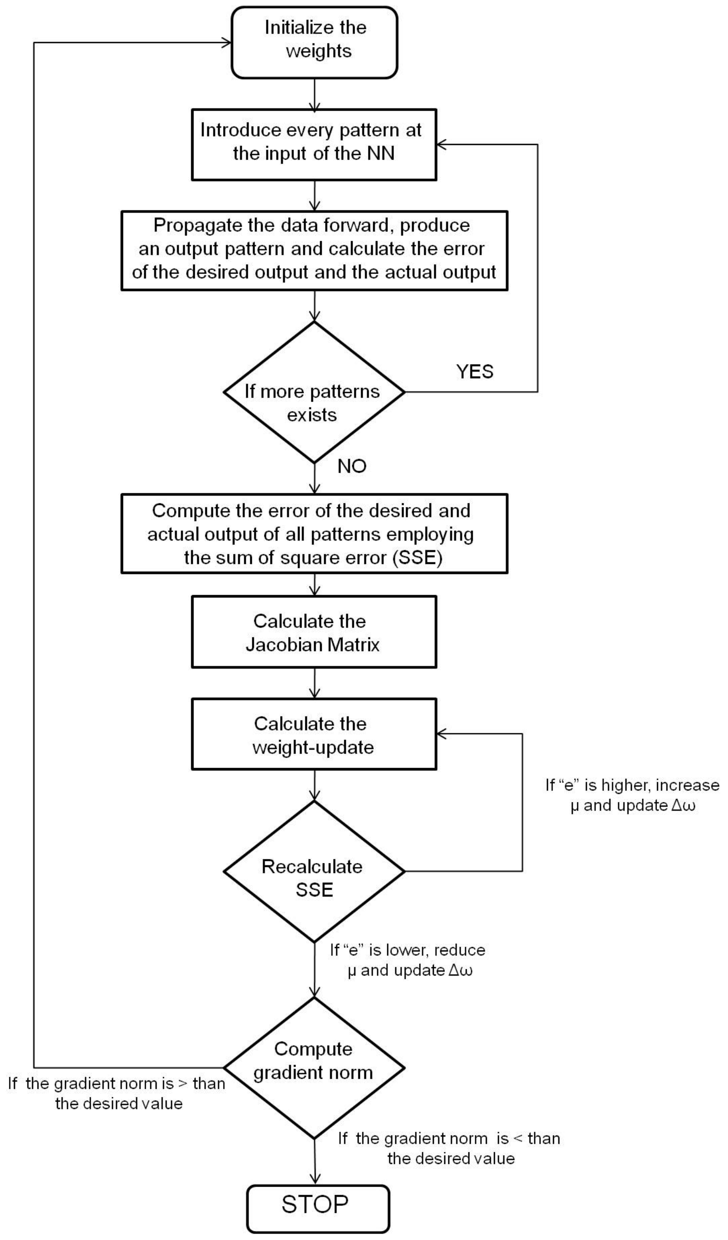

Initialize the weights
Introduce every pattern at the input of the NN
Propagate the data forward, produce an output pattern and calculate the error of the desired output and the actual output
If more patterns exists
YES
NO
Compute the error of the desired and actual output of all patterns employing the sum of square error (SSE)
Calculate the Jacobian Matrix
Calculate the weight-update
If "e" is higher, increase μ and update Δω
Recalculate SSE
If "e" is lower, reduce μ and update Δω
Compute gradient norm
If the gradient norm is > than the desired value
If the gradient norm is < than the desired value
STOP

Figure 4. Levenberg-Maquard backpropagation algorithm steps [18].

were obtained by using a genetic algorithm pattern search to track the desired system trajectory. The desired parameters for the system were first established as the model reference desired trajectory (no faults in the system) and then the genetic algorithm obtained the best parameter optimization (see Table 2).

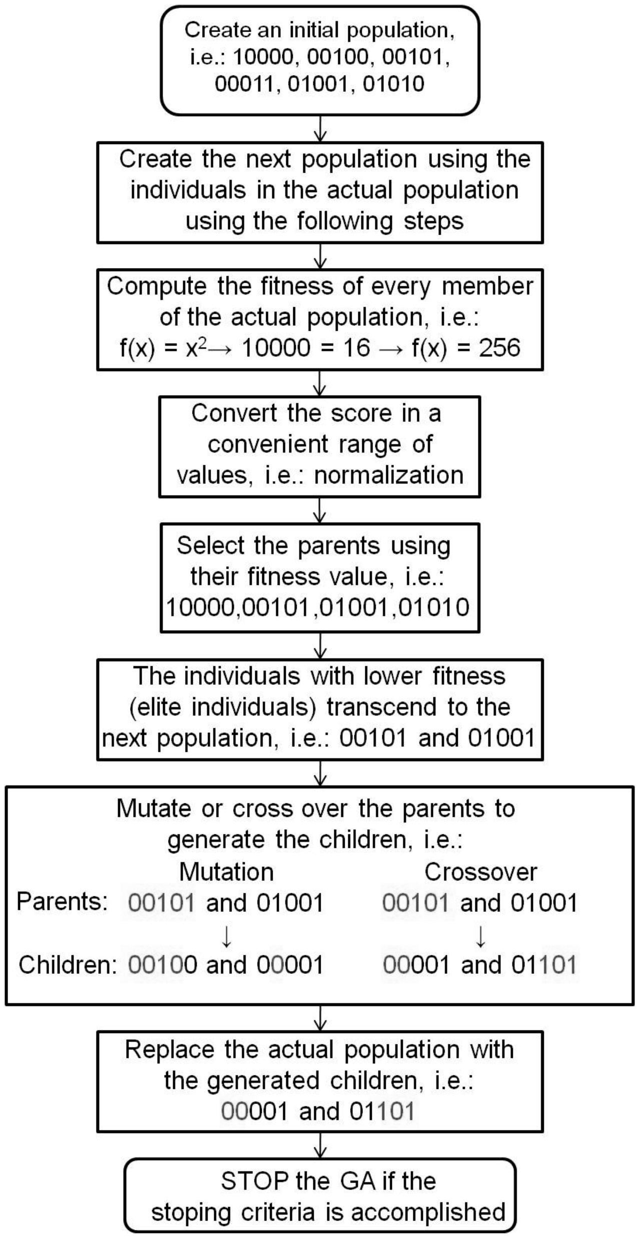

Create an initial population, i.e.: 10000, 00100, 00101, 00011, 01001, 01010
Create the next population using the individuals in the actual population using the following steps
Compute the fitness of every member of the actual population, i.e.: f(x) = x²→ 10000 = 16 → f(x) = 256
Convert the score in a convenient range of values, i.e.: normalization
Select the parents using their fitness value, i.e.: 10000,00101,01001,01010
The individuals with lower fitness (elite individuals) transcend to the next population, i.e.: 00101 and 01001
Mutate or cross over the parents to generate the children, i.e.:
Mutation
Crossover
Parents: 00101 and 01001
00101 and 01001
Children: 00100 and 00001
00001 and 01101
Replace the actual population with the generated children, i.e.: 00001 and 01101
STOP the GA if the stoping criteria is accomplished

Figure 5. Genetic algorithm steps.

$$PID = K_p + K_i/_s + K_d s \qquad (17)$$

GA Parameter	Unit
Population Size	20
Generations	100 Epoch
Selection Function	Tournament
Crossover Function	Scattered
Crossover Fraction	0.7
Mutation Function	Adapt Feasible
Stall Generation	50
Stall Time Limit	20
Function Tolerance	1e-06
Constraint Tolerance	1e-06
Time Limit	Inf
Maximum Iterations Number	100
Maximum Evaluation Number	100000
Variable Tolerance	1e-06
Mesh Size Tolerance	1e-06

Table 1. Genetic algorithm parameters.

Parameter	Steam Plant	Water Plant
Kp	1.7764	0.8276
Ki	1.2229	0.0010
Kd	-0.0047	1.1475

Table 2. Obtained best PID controller parameters using GA optimization.

2.4 Sliding mode control

The sliding mode controller is a technique in which the states of the systems reach a sliding surface (denoted by s) and are maintained there by a shifting law design in order to stabilize the system using a state feedback control law [20]. In order to develop the procedure used to design this controller, first the original transfer function is decomposed in a cascade system, and the following equations are obtained:

$$\dot{x}_1 = -\alpha_1 x_1 - \alpha_2 x_2 + u \qquad \dot{x}_2 = x_1$$
$$y = \alpha_3 x_2 \tag{18}$$

Where α_1, α_2, and α_3 are constants, x_2 is equal to the system output, $\dot{x}_2$ is the derivative of x_2 and $\dot{x}_1$ is the derivative of x_1. With the above variables defined, the following equations can be established:

$$\dot{x}_1 = f(x) + g(x)u \qquad \dot{x}_2 = x_1 \tag{19}$$

Where *f(x)* and *g(x)* are nonlinear functions, *g(x)>0*, *f(x)* and *g(x)* need not to be continuous and x_2 is stable if

$$\dot{x}_2 = -ax_2 \qquad a > 0 \tag{20}$$

On the other hand, if

$$s = x_1 + ax_2 \rightarrow \dot{x}_2 = x_1 = -ax_2 + s \tag{21}$$

Then, the time derivate of *s* is

$$\dot{s} = \dot{x}_1 + a\dot{x}_2 = f(x) + g(x)u + ax_1 \qquad (22)$$

Therefore, the Lyapunov candidate function is

$$V = \tfrac{1}{2}s^2 \qquad (23)$$

Where

$$\dot{V} = \dot{s}s = s\left[f(x) + g(x)u + ax_1\right] \qquad (24)$$

$\dot{V}$ is negative definite if

$$f(x) + g(x)u + ax_1 \begin{cases} < 0 \rightarrow s > 0 \\ = 0 \rightarrow s = 0 \\ > 0 \rightarrow s < 0 \end{cases} \qquad (25)$$

The stability is ensured if

$$u_2 \begin{cases} < \beta \rightarrow s > 0 \\ = \beta \rightarrow s = 0 \\ > \beta \rightarrow s < 0 \end{cases} \qquad \beta(x) = \frac{f(x) + ax_1}{g(x)} \qquad (26)$$

Finally, the control law that will be used is

$$u = \beta(x) - K\operatorname{sign}(s) \qquad (27)$$

where K>0.

2.5 H∞ Controller

The H∞ control theory is a robust technique implemented in [21] to achieve robust performance and stabilization in an implemented system. This control theory uses the H∞ norm which is the frequency response magnitude to the maximum singular value of the interested transfer function (i.e. peak gain or worst case disturbances). The standard configuration problem for an H∞ controller is shown next (Figure 6, left side) [22]:

Where *K* is the H∞ controller, *P* is a generalized plant, *u* are the control variables, *w* the exogenous signals, *z* the error signals which have to be minimized to achieve the control objectives, and *v* the measured variables. In terms of state space, the above is rewritten as [22]

$$\begin{bmatrix} z \\ v \end{bmatrix} = P(s)\begin{bmatrix} w \\ u \end{bmatrix} = \begin{bmatrix} P_{11}(s) & P_{12}(s) \\ P_{21}(s) & P_{22}(s) \end{bmatrix}\begin{bmatrix} w \\ u \end{bmatrix} \qquad (28)$$

$$u = K(s)v \qquad (29)$$

$$P = \begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix} \qquad (30)$$

In which the linear fractional transformation of *w* to *z* is given by

$$z = F_l(P,K)w \rightarrow F_l(P,K) = P_{11} + P_{12}K(I - P_{22}K)^{-1}P_{21}$$

(31)

where the H∞ control implicates the minimization of H∞ norm of $F_l(P,K)$.

$$\left\| F_l(P,K) \right\|_\infty = \sup \bar{\sigma}\left(F_l(P,K)(j\omega)\right) \qquad (32)$$

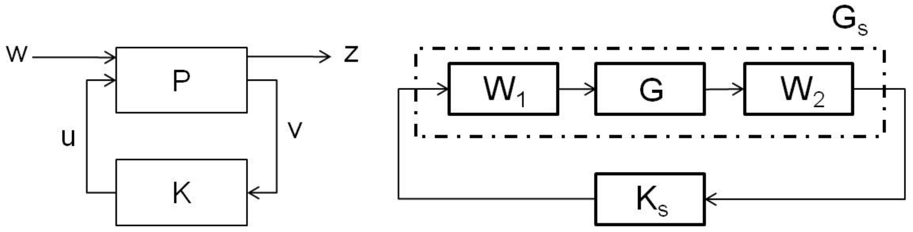


Figure 6. General H∞ controller configuration (left side) and shaped plant and controller (right side).

where $\overline{\sigma}$ means the upper singular value.

The H∞ loop-shaping method proposed in [23] consists of a two stage process. In the first stage, the plant is augmented by pre and post compensators (W_1 and W_2, respectively) in order to obtain the desired shape of the singular values in an open-loop system. Second, the shaped plant is stabilized with H∞ optimization. The shaped plant is denoted by (see right side of Figure 6):

$$G_s = W_2 G W_1 \tag{33}$$

Controller K_s is obtained solving the robust stabilization problem for the shaped plant G_s with left coprime factorization.

$$G_s = M^{-1}N \tag{34}$$

Therefore, the perturbed plant model G_p can be represented as

$$G_p = (M + \Delta_M)^{-1} + (N + \Delta_N) \tag{35}$$

Where Δ_M and Δ_N represent the uncertainty of the nominal plant *G*. Then, the goal of robust stabilization is to stabilize *G* and the family of perturbed plants G_p.

$$G_p = \{(M + \Delta_M)^{-1} + (N + \Delta_N) : \|\Delta_M \Delta_N\|_\infty < \varepsilon\} \tag{36}$$

Where *ε>0* is the stability margin. For the above problem, the stability property is stable if and only if the nominal feedback system is stable and

$$H_\infty norm \overset{\Delta}{=} \left\| \begin{bmatrix} K \\ I \end{bmatrix} (I - GK)^{-1} M^{-1} \right\|_\infty < \varepsilon \tag{37}$$

Where $(I\text{-}GK)^{-1}$ is the sensitivity function. Finally, the feedback controller for plant *G* of Figure 6 will be

$$K = W_1 K_s W_2 \tag{38}$$

The H∞ control proposed in this paper was designed by using the loop-shaping method and the following steps were realized: First, the worst case of system faults (steam process faults and water process faults) was simulated and identified in the form of a Laplace function obtaining the following equations:

$$G_{steam\_faults} = \frac{0.00002}{s^2 + 0.00004309\,s + 0.00002} \tag{39}$$

$$G_{water\_faults} = \frac{-0.000013}{s^2 + 0.007819\,s + 0.000082} \tag{40}$$

Where G_{steam_faults} and G_{water_faults} represent the worst case of faults in the steam process and water process, respectively. Second, the above Laplace functions are compared against the non fault steam and water process:

$$G_{steam} = \frac{0.00002}{s^2 + 0.004299s + 0.00002} \tag{41}$$

$$G_{water} = \frac{-0.000013}{s^2 + 0.007815s + 0.00008} \tag{42}$$

Where G_{steam} is the non fault function describing the steam process and G_{water} is the non fault function describing the water process (the above will be explained in more detail in Section 3). Third, a loop-shaping control synthesis is performed to calculate an optimal H∞ controller for the Laplace fault functions (e.g. G_{water_faults}). This controller shapes the sigma plot of the Laplace fault function and obtains the desired loop shaping (non fault steam and water function) with a precision parameter called "GAM" (e.g. if GAM should be ≥ 1 with GAM = 1 being a perfect match).The next procedure is to calculate the stable minimum-phase loop shaping. This was realized squaring down a pre-filter *W*:

$$W_{steam} = C(sI - A)^{-1}B + D \tag{43}$$

$$W_{steam} = \begin{bmatrix} 1.311e^{5} & 6.872e^{10} & 0 & -6.872e^{10} \end{bmatrix} \left(sI - \begin{bmatrix} -8192 & -4.295e^{9} & 0 & 4.295e^{9} \\ 0.003906 & -1.819e^{-12} & 0 & -0.001703 \\ 0 & 0 & -0.004299 & -0.00512 \\ 0 & 0 & 0.003906 & -4.748e^{-20} \end{bmatrix} \right)^{-1} \begin{bmatrix} -3.469e^{-18} \\ 2.367e^{-30} \\ -0.0625 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} \tag{44}$$

$$W_{steam} = \frac{3.638e^{-12}s^3 + 1.678e^7 s^2 + 2.929e^4 s - 348.6}{s^4 + 8192s^3 + 1.678e^7 s^2 + 7.213e^4 s - 335.5} \tag{45}$$

In a manner that the shaped plant G_s is square in state space formulation:

$$G_s = G_{steam\_faults} W = C(sI - A)^{-1} B + D \tag{46}$$

$$G_s = \tag{47}$$

$$\begin{bmatrix} 1.057e^{-14} & -167.8 & -6.617e^{-24} & -6.939e^{-18} & -6.617e^{-24} & -1.388e^{-17} \end{bmatrix} \left(sI - \begin{bmatrix} -8192 & -4096 & 3.815e^{-6} & 1 & 4.235e^{-22} & 2 \\ 4096 & -1.819e^{-12} & 2.541e^{-21} & 4.441e^{-16} & -2.118e^{-22} & 4.441e^{-16} \\ 0 & 0 & -4.309e^{-5} & -0.002561 & 1.291e^{-26} & 0.0001775 \\ 0 & 0 & 0.007813 & -3.939e^{-13} & 4.136e^{-25} & 0.003405 \\ 0 & 0 & 0 & 0 & -0.004299 & -0.00512 \\ 0 & 0 & 0 & 0 & 0.003906 & 4.758e^{20} \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 \\ -7.175e^{-43} \\ -9.861e^{-32} \\ 0 \\ -0.0625 \\ 0 \end{bmatrix} + \begin{bmatrix} -5.049e^{-29} \end{bmatrix}$$

$$G_s = \frac{3.725e^{-9}s^4 + 5.384e^{10}s^3 + 335.5s^2 + 0.5858s - 0.00697}{s^6 + 8192s^5 + 1.678e^7 s^4 + 7.285e^4 s^3 + 674.3s^2 + 1.457s - 0.006712} \tag{48}$$

Then, the desired shape of G_{steam} is accomplished with high precision in the frequency range by the shaped plant. After the above, the normalized coprime factor control synthesis is used to calculate the ideal loop-shaping controller (K_s):

$$K_s = C(sI - A)^{-1} B + D \tag{49}$$

$$K_s = \tag{50}$$

$$\begin{bmatrix} 0.1002 & -0.03229 & 0.06381 & 0.0001991 & 3.415e^{-5} \end{bmatrix} \left(sI - \begin{bmatrix} -0.006639 & 0.004199 & -0.005592 & -1.938e^{-5} & -3.324e^{-6} \\ -0.003146 & -0.0003806 & 0.002807 & 2.63e^{-6} & 4.512e^{-7} \\ -0.004052 & -0.002715 & -0.005551 & -2.675e^{-5} & -4.59e^{-6} \\ 2.125e^{-5} & 8.556e^{-5} & 1.548e^{-5} & -2648 & -1448 \\ -3.443e^{-5} & -5.083e^{-6} & 1.278e^{-5} & 1448 & -5544 \end{bmatrix} \right)^{-1} \begin{bmatrix} -0.1399 \\ -0.03378 \\ -0.06457 \\ 0.0001991 \\ -3.415e^{-5} \end{bmatrix} + \begin{bmatrix} 1.714 \end{bmatrix}$$

$$K_s = \frac{1.714s^5 + 1.404e^4 s^4 + 2.876e^7 s^3 + 7.557e^4 s^2 + 417.2s - 0.5835}{s^5 + 8192s^4 + 1.678e^7 s^3 + 2.109e^5 s^2 + 665.6s - 3.773} \tag{51}$$

Finally, with Equation 51, the H∞ controller is computed using:

$$H_{\infty\_steam} = W * K_s \tag{52}$$

$$H_{\infty\_steam} = \frac{-3.63e^{-12}s^{8} + 2.876e^{7}s^{7} + 2.356e^{11}s^{6} + 4.825e^{14}s^{5} + 2.11e^{12}s^{4} + 1.924e^{10}s^{3} + 3.208e^{7}s^{2} + 1.427e^{5}s - 127}{s^{9} + 1.638e^{4}s^{8} + 1.007e^{8}s^{7} + 2.749e^{11}s^{6} + 2.815e^{14}s^{5} + 4.749e^{12}s^{4} + 3.201e^{10}s^{3} + 1.821e^{8}s^{2} + 4.954e^{5}s - 1266} \quad (53)$$

The parameter GAM, which is a precision parameter for this controller, is 1.98461 which is equal to 5.9535 dB. Also, in Figure 8, the singular value plot of this controller can be viewed. In Figure 7, parameter *L* is the open loop, *T* is the complementary sensitivity or closed-loop function and *S* is the sensitivity function. It can be observed that these parameters are within the specified boundaries denoted by the singular values of the non fault system +/- the value of GAM.

$$L = G_{steam\_faults} * K \quad (54)$$

$$L = \frac{-1.49e^{-8}s^{9} - 6.104e^{-5}s^{8} + 575.3s^{7} + 4.712e^{6}s^{6} + 9.651e^{9}s^{5} + 4.221e^{7}s^{4} + 3.011e^{5}s^{3} + 355.6s^{2} + 1.352s - 0.00237}{s^{11} + 1.638e^{4}s^{10} + 1.007e^{8}s^{9} + 2.749e^{11}s^{8} + 2.815e^{14}s^{7} + 4.761e^{12}s^{6} + 3.784e^{10}s^{5} + 2.784e^{8}s^{4} + 1.143e^{6}s^{3} + 4929s^{2} + 9.963s + 0.02532} \quad (55)$$

$$S = \frac{1}{1 + G_{steam\_faults} * K} \quad (56)$$

$$S = \frac{s^{11} + 1.638e^{4}s^{10} + 1.007e^{8}s^{9} + 2.749e^{11}s^{8} + 2.815e^{14}s^{7} + 4.761e^{12}s^{6} + 3.784e^{10}s^{5} + 2.784e^{8}s^{4} + 1.143e^{6}s^{3} + 4929s^{2} + 9.963s + 0.02532}{s^{11} + 1.638e^{4}s^{10} + 1.007e^{8}s^{9} + 2.749e^{11}s^{8} + 2.815e^{14}s^{7} + 4.761e^{12}s^{6} + 4.749e^{10}s^{5} + 3.206e^{8}s^{4} + 1.445e^{6}s^{3} + 5284s^{2} + 11.32s + 0.02295} \quad (57)$$

$$T = \frac{G_{steam\_faults} * K}{1 + G_{steam\_faults} * K} \quad (58)$$

$$T = \frac{-1.49e^{-8}s^{9} - 6.104e^{-5}s^{8} + 575.3s^{7} + 4.721e^{6}s^{6} + 9.651e^{9}s^{5} + 4.221e^{7}s^{4} + 3.011e^{5}s^{3} + 355.6s^{2} + 1.352s - 0.00237}{s^{11} + 1.638e^{4}s^{10} + 1.007e^{8}s^{9} + 2.749e^{11}s^{8} + 2.815e^{14}s^{7} + 4.761e^{12}s^{6} + 4.749e^{10}s^{5} + 3.206e^{8}s^{4} + 1.445e^{6}s^{3} + 5284s^{2} + 11.32s + 0.02295} \quad (59)$$

The above procedure is repeated for the G_{water_faults} process and the following H∞ controller (H_{∞_water}) is obtained:

$$T = \frac{-1.49e^{-8}s^{9} - 6.104e^{-5}s^{8} + 575.3s^{7} + 4.721e^{6}s^{6} + 9.651e^{9}s^{5} + 4.221e^{7}s^{4} + 3.011e^{5}s^{3} + 355.6s^{2} + 1.352s - 0.00237}{s^{11} + 1.638e^{4}s^{10} + 1.007e^{8}s^{9} + 2.749e^{11}s^{8} + 2.815e^{14}s^{7} + 4.761e^{12}s^{6} + 4.749e^{10}s^{5} + 3.206e^{8}s^{4} + 1.445e^{6}s^{3} + 5284s^{2} + 11.32s + 0.02295} \quad (60)$$

The parameter GAM for this controller is 1.0152445; it is equivalent to 0.1314129 dB.

$$L = G_{water\_faults} * K \quad (61)$$

$$L = \frac{-6.1042e^{-5}s^{6} + 38.19s^{5} + 3.132e^{5}s^{4} + 6.414e^{8}s^{3} + 5.626e^{6}s^{2} + 5.602e^{4}s + 48.82}{s^{9} + 1.638e^{4}s^{8} + 1.007e^{8}s^{7} + 2.749e^{11}s^{6} + 2.815e^{14}s^{5} + 8.133e^{12}s^{4} + 1.206e^{11}s^{3} + 1.177e^{9}s^{2} + 6.471e^{6}s + 2.389e^{4}} \quad (62)$$

$$S = \frac{1}{1 + G_{water\_faults} * K} \quad (63)$$

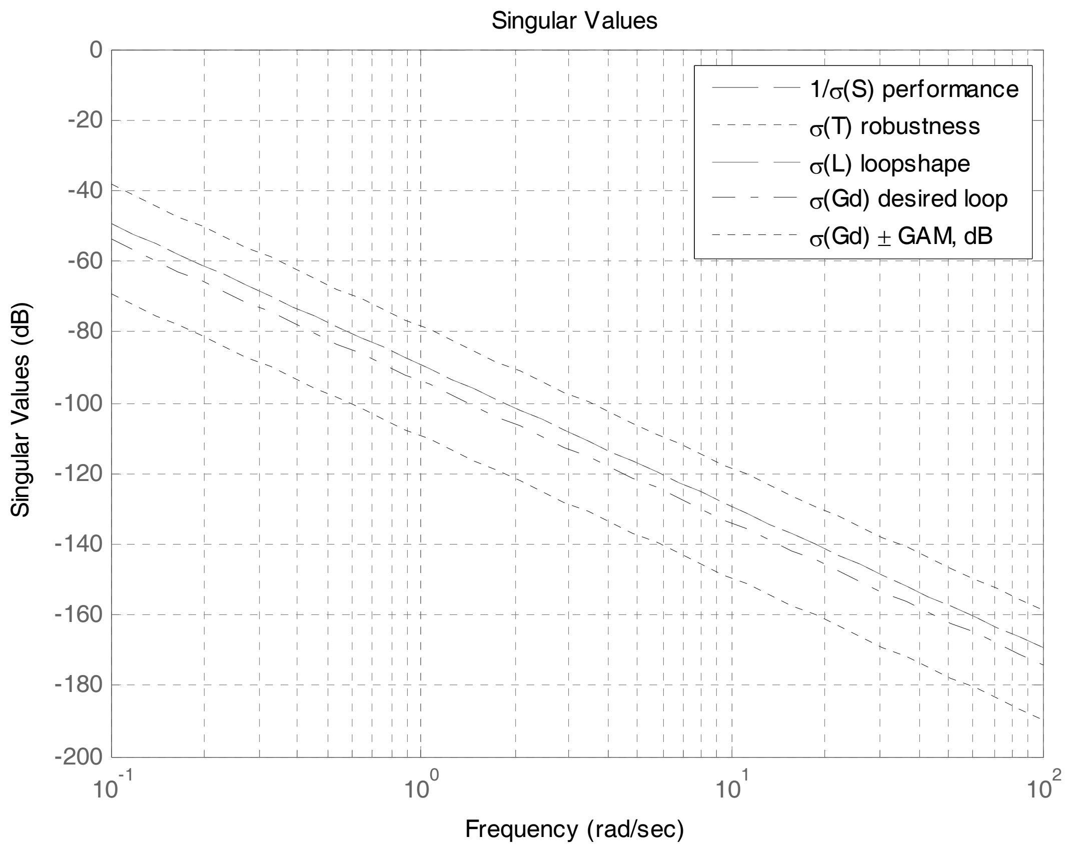


Figure 7. H∞ controller singular value plot for the steam process.

$$S = \frac{s^9 + 1.638e^4 s^8 + 1.007e^8 s^7 + 2.749e^{11} s^6 + 2.815e^{14} s^5 + 8.133e^{12} s^4 + 1.206e^{11} s^3 + 1.177e^9 s^2 + 6.471e^6 s + 2.389e^4}{s^9 + 1.638e^4 s^8 + 1.007e^8 s^7 + 2.749e^{11} s^6 + 2.815e^{14} s^5 + 8.133e^{12} s^4 + 1.212e^{11} s^3 + 1.183e^9 s^2 + 6.527e^6 s + 2.394e^4} \quad (64)$$

$$T = \frac{G_{water\_faults} * K}{1 + G_{water\_faults} * K} \quad (65)$$

$$T = \frac{-6.1042e^{-5} s^6 + 38.19 s^5 + 3.132e^5 s^4 + 6.414e^8 s^3 + 5.626e^6 s^2 + 5.602e^4 s + 48.82}{s^9 + 1.638e^4 s^8 + 1.007e^8 s^7 + 2.749e^{11} s^6 + 2.815e^{14} s^5 + 8.133e^{12} s^4 + 1.212e^{11} s^3 + 1.183e^9 s^2 + 6.527e^6 s + 2.394e^4} \quad (66)$$

3. Proposed scheme

An industrial heat exchanger was used to test these approaches (shown in Figure 8 and Table 3). This process is a shell and tube industrial heat exchanger that has two inputs: water and steam flows controlled by pneumatic valves (FSV_1 and FSV_2, respectively), the water passs inside the tubes at room temperature and the steam passes through the tube walls in order to transfer heat to the water. In addition, the industrial heat exchanger has one output in which the water temperature is measured by a thermistor (TT_2). Variations in water and steam flows are determined by flow transmitters (FT_1 and FT_2, respectively).

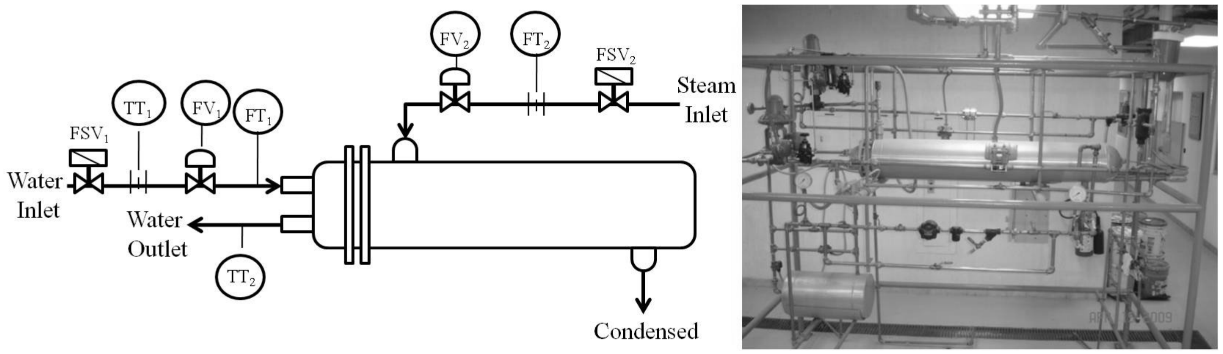


Figure 8. Industrial heat exchanger used in the experiments.

Tag Name	Description
FSV_1	Solenoid valve in the water inlet
TT_1	Temperature transmitter of the water inlet
FV_1	Pneumatic control valve in the water inlet
FT_1	Flow transmitter in the water inlet
TT_2	Temperature transmitter of the water outlet
FV_2	Pneumatic control valve in the steam inlet
FT_2	Flow transmitter of the steam inlet
FSV_2	Solenoid valve in the steam inlet

Table 3. Industrial heat exchanger sensors/transmitters description.

$$G_p = G_{steam} + \left(-G_{water}\right) \tag{67}$$

$$G_p = \frac{0.00002}{s^2 + 0.004299s + 0.00002} + \frac{-0.000013}{s^2 + 0.007815s + 0.00008} \tag{68}$$

$$T(s) = \frac{0.00002}{s^2 + 0.004299s + 0.00002} F_{steam}(s) - \frac{0.000013}{s^2 + 0.007815s + 0.00008} F_{water}(s) \tag{69}$$

$$G_{steam} = \frac{T(s)}{F_{steam}(s)} = \frac{0.00002}{s^2 + 0.004299s + 0.00002} \qquad (70)$$

$$G_{water} = \frac{T(s)}{F_{water}(s)} = \frac{-0.000013}{s^2 + 0.007815s + 0.00008} \qquad (71)$$

where G_p represents the process model, G_{steam} and G_{water} describe the steam and water model of the industrial heat exchanger, respectively. T(s) describes the water temperature at the exit and F_{steam}(s) and F_{water}(s) represent the steam and water flow, respectively. In this research, a total of six different schemas for fault-tolerant control were proposed (see Figure 9).

The first scheme is a model reference adaptive control with an artificial neural network and a PID controller optimized by a genetic algorithm (MRAC-PID-ANN). The second method is a combination of a model reference adaptive controller and an H∞ controller (MRAC-H∞). The third scheme is a model reference adaptive controller with a sliding mode controller (MRAC-SMC). The fourth method is a model reference adaptive controller with an artificial neural network (MRAC-ANN). The fifth method is a combination of a model reference adaptive controller with a PID controller optimized by a genetic algorithm (MRAC-PID) (see Figure 9). Finally, the last scheme is a classical model reference adaptive controller (MRAC) (see Figure 2).

4. Experiment and results

Two different types of faults were simulated on the implemented schemes: abrupt faults and gradual faults. In abrupt faults, the whole magnitude of the fault is developed in one moment of time and was simulated with a step function; gradual faults are developed during a period of time and are implemented with a ramp function.

Both types of faults, abrupt and gradual, do not affect the process properties but can change its behavior and can be implemented in sensors (feedback) and actuators (process entry).

These types of faults were tested in the six proposed schemes: MRAC-PID-ANN, MRAC-H∞, MRAC-SMC, MRAC-ANN, MRAC-PID and classical MRAC. In order to compare the different system structures, a total of seventy-two different experiments were simulated. In each experiment, a fault was introduced at time 5000 seconds. Three different fault levels were simulated: 5%, 15% and 25%. The fault size is in terms of percentage deviation from de normal operational value, *f=fault*. In the next figures, three different results are explained: robust (no changes occur in the system after the fault), adaptation mechanism (the system tolerates and accommodates the fault) and degraded system (the system does not tolerate the fault). For the gradual faults, the slope was of 10% deviation from the normal operational value per second but had a saturation block stopping this percentage at the different fault percentage values (5%, 15% and 25%).

In addition, the mean square error (MSE) was calculated for all the experiments, the results are shown in Table 4 and Table 5 for sensor and actuator faults, respectively.

$$MSE = \frac{\sum (y_{reference} - y_{process})^2}{n-2} \qquad (72)$$

where $y_{reference}$ is the output of the reference model, $y_{process}$ is the output of the actual process and n is the sampling period.

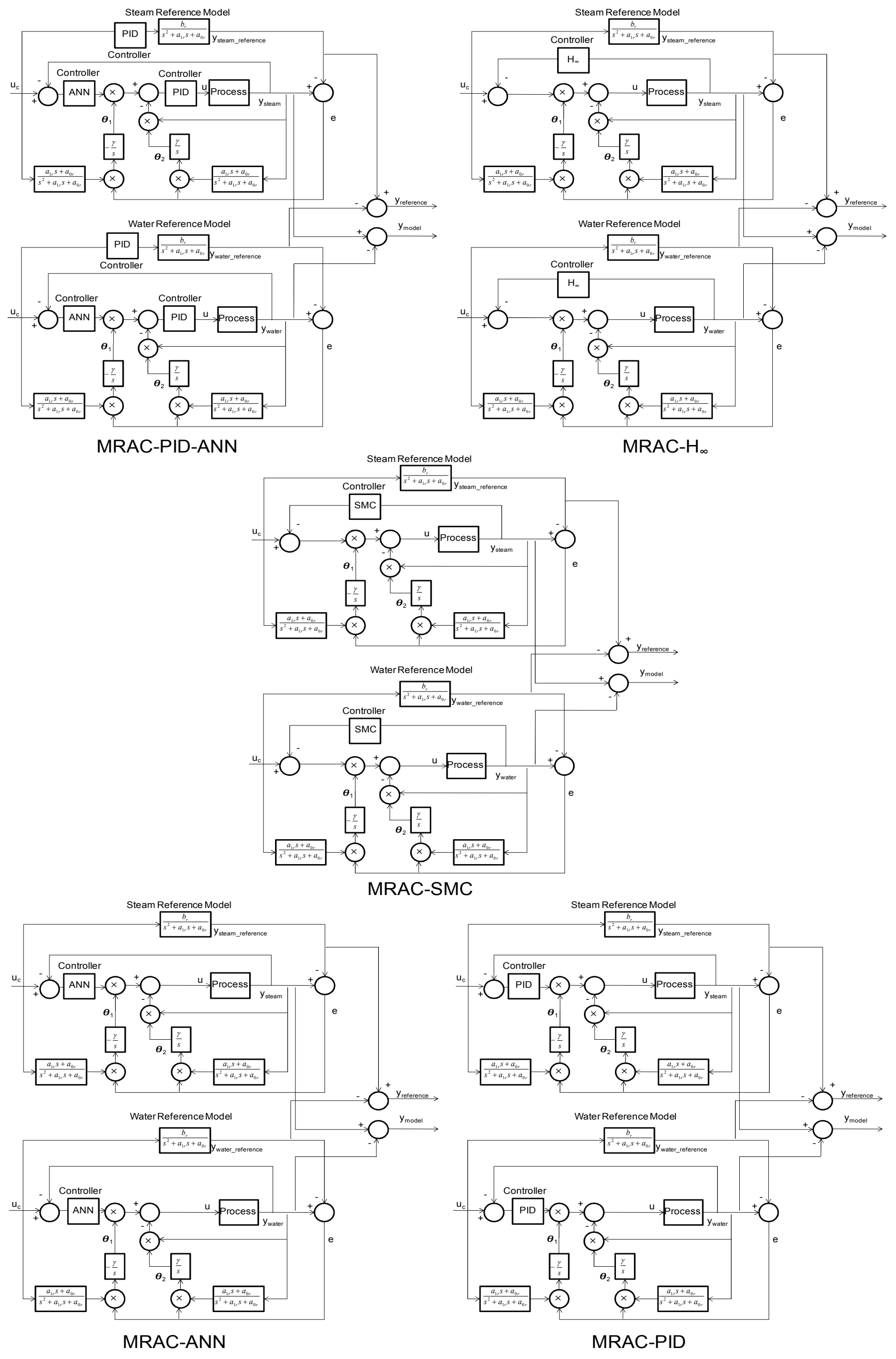


Figure 9. Industrial heat exchanger fault-tolerant MRAC-PID-ANN controller structure (upper left corner), MRAC-H∞ controller structure (upper right corner), MRAC-SMC controller structure (center), MRAC-ANN controller structure (lower left corner) and MRAC-PID controller structure (lower right corner).

Approaches	*Abrupt Sensor Faults*			*Gradual Sensor Faults*		
	f = 5%	f = 15%	f = 25%	f = 5%	f = 15%	f = 25%
MRAC-PID-ANN	1.107E-05	1.107E-05	1.107E-05	1.107E-05	1.107E-05	1.107E-05
MRAC-H∞	4.803E-06	4.803E-06	0.0007472	4.804E-06	4.804E-06	0.0007358
MRAC-SMC	1.843E-06	1.843E-06	1.843E-06	3.922E-06	3.922E-06	3.922E-06
MRAC-ANN	2.069E-05	2.069E-05	2.069E-05	2.069E-05	2.069E-05	2.069E-05
MRAC-PID	1.521E-05	1.521E-05	0.0698587	1.522E-05	1.522E-05	0.0698464
MRAC	0.0005245	0.0097135	1.8321538	0.0005246	0.0097138	1.8320687

Table 4. Mean square error for the abrupt and gradual sensor faults.

In the above results, it can be seen that in general the MRAC-SMC has the lower MSE for the abrupt and gradual sensor faults (MSE = 1.843E-06 for the abrupt sensor faults and MSE = 3.922E-06 for the gradual sensor faults).

Approaches	*Abrupt Actuator Faults*			*Gradual Actuator Faults*		
	f = 5%	f = 15%	f = 25%	f = 5%	f = 15%	f = 25%
MRAC-PID-ANN	1.107E-05	1.107E-05	1.107E-05	1.107E-05	1.107E-05	1.107E-05
MRAC-H∞	0.0003495	0.0023871	0.0050391	0.0003494	0.0023870	0.0050384
MRAC-SMC	0.0003501	0.0024160	0.1223235	0.0003504	0.0024173	0.1223123
MRAC-ANN	0.2993774	0.1766571	0.1210197	0.2993563	0.1766339	0.1210041
MRAC-PID	0.2973728	0.1771648	0.1205219	0.2973542	0.1771336	0.1205066
MRAC	0.1182904	0.0855958	0.1187836	0.1182899	0.0855955	0.1187681

Table 5. Mean square error for the abrupt and gradual actuator faults.

On the other hand, the MRAC-PID-ANN scheme has the lower MSE for the abrupt and gradual actuator faults (MSE = 1.107E-05). The following graphs (Figure 10 and Figure 11) represent a comparison between the different simulated experiments, in these graphs a fault was introduced at time 5000 seconds.

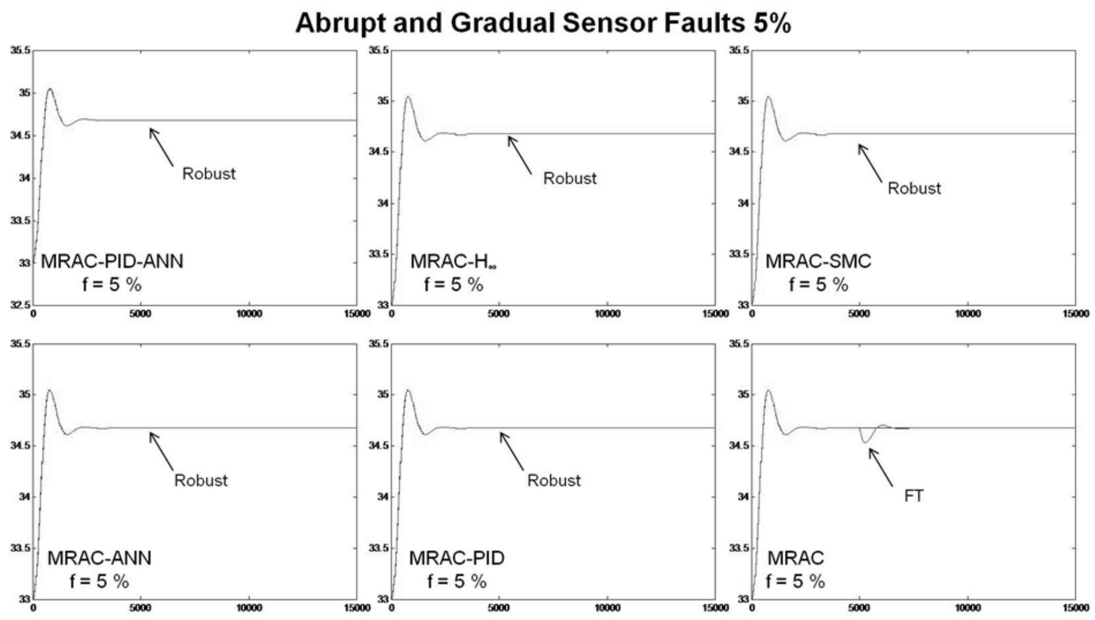
In Figure 10, it is observed that for abrupt and gradual sensor faults of 5% and 15% of system deviation almost all of the schemes are robust against the fault except for the classical MRAC approach which was fault-tolerant against these types of faults. For faults of 25%, it can be viewed that the robust schemes are the MRAC-PID-ANN (MSE = 1.107E-05 for the abrupt and gradual sensor faults), MRAC-SMC (MSE = 1.843E-06 and 3.922E-06 for abrupt and gradual sensor faults, respectively) and MRAC-ANN (MSE = 2.069E-05 for abrupt and gradual sensor faults). On the other hand, the MRAC-H∞ scheme was fault-tolerant against this type of faults with an MSE of 0.0007472 and 0.0007358 for abrupt and gradual faults, respectively. In addition, the MRAC-PID and the classical MRAC scheme resulted in a degraded system with these types of faults.

In Figure 11, it is observed that for abrupt and gradual actuator fault case of 5% and 15%, the best scheme is the MRAC-PID-ANN because is robust against the faults. The second best schemes are the MRAC-H∞ and the MRAC-SMC approaches because these schemes are fault-tolerant to faults of 5% and 15% of deviation. The MRAC-ANN, MRAC-PID and MRAC schemes had certain system degradation after the occurrence of the fault. Furthermore, these results are confirmed with the MSE of Table 5. For example, for abrupt and gradual actuator faults of 15% of deviation, the MSE are around 1.107E-05, 0.00238, 0.00241, 0.17665, 0.17716 and 0.08559 for the MRAC-PID-ANN, MRAC-H∞, MRAC-SMC, MRAC-ANN, MRAC-PID and MRAC, respectively. On the other hand, for abrupt and gradual actuator fault case of

25% of deviation, the best scheme is the MRAC-PID-ANN because is robust against the faults. The second best scheme is the MRAC-H_∞ approach because this scheme is fault-tolerant to the 25% of deviation faults. On the other hand, the MRAC-SMC, MRAC-ANN, MRAC-PID and MRAC schemes had certain system degradation after the occurrence of the fault. Also, these results are confirmed with the MSE of Table 5.

5. Comparisons with similar approaches

Although some other MRAC approaches have been developed in the last years, there are several differences from the methods proposed in this paper. In [24], an MRAC based on a PID controller and a GA, in which the GA was used to optimize the PID parameters, was introduced; this scheme was used in a continuous stirred-tank reactor system (CSTR), and an ANN trained by the GA was used to estimate the state value of the CSTR. The differences between our scheme and the one presented in this paper is that the ANN in [24] is used as an estimator of the plant and in this paper the ANN is used as a trajectory controller to follow the ideal system trajectory (normal operation mode). In [25], an ANN-based state feedback MRAC for a type of nonlinear plants was presented. This methodology uses a time-varying pseudo-linear feedback control in which the gain of the state feedback is generated from the ANN output; this scheme is different from the proposed structure presented in this paper, first of all because it does not use a GA to optimize a PID controller and also because the ANN is used just to approximate the controller parameters not as a trajectory controller. In [26], an MRAC application with an aged actuator in order to identify vulnerable devices or control parameters on stability was proposed, this scheme did not use any artificial intelligence method (ANN or GA) and neither used a PID controller.



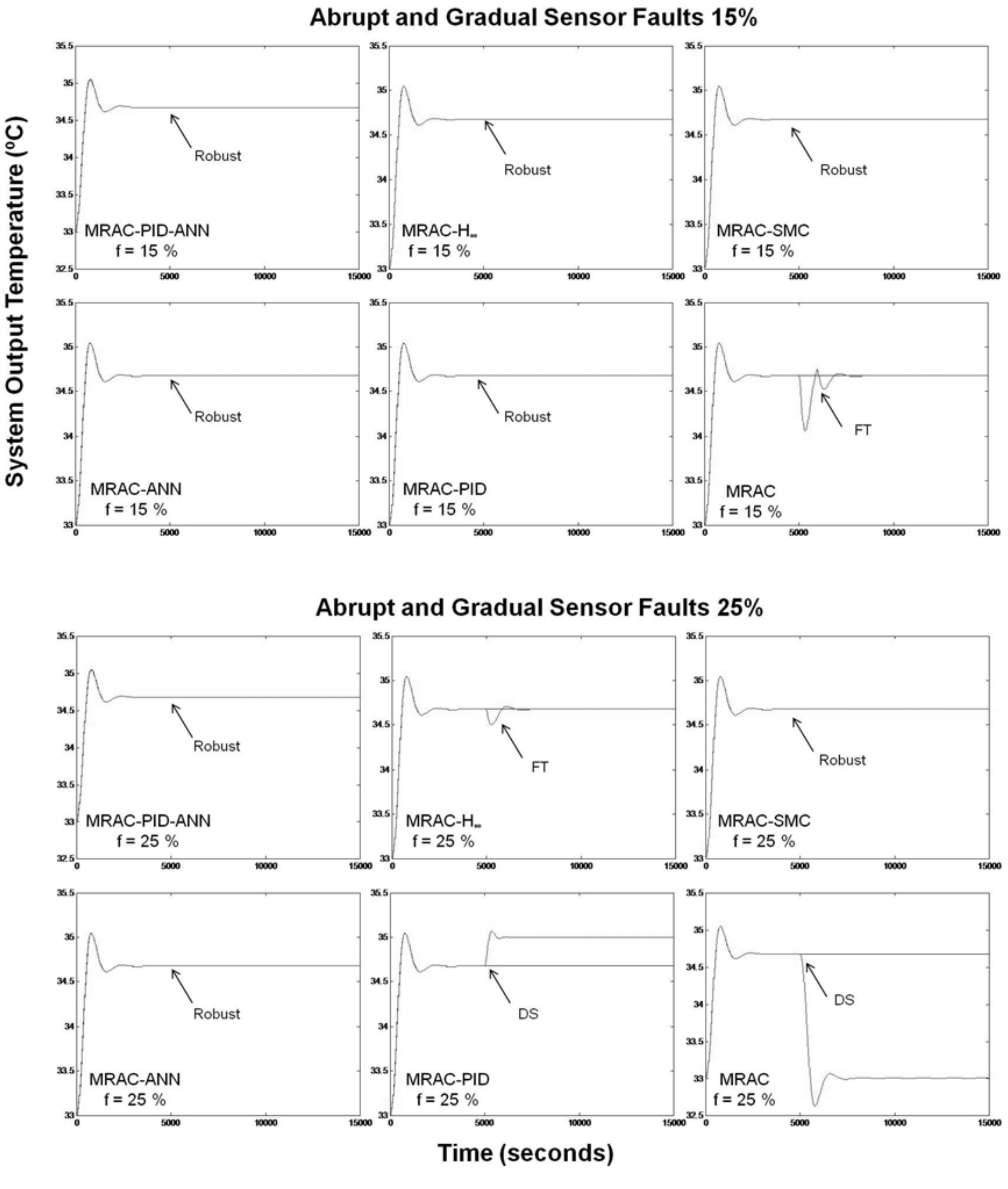
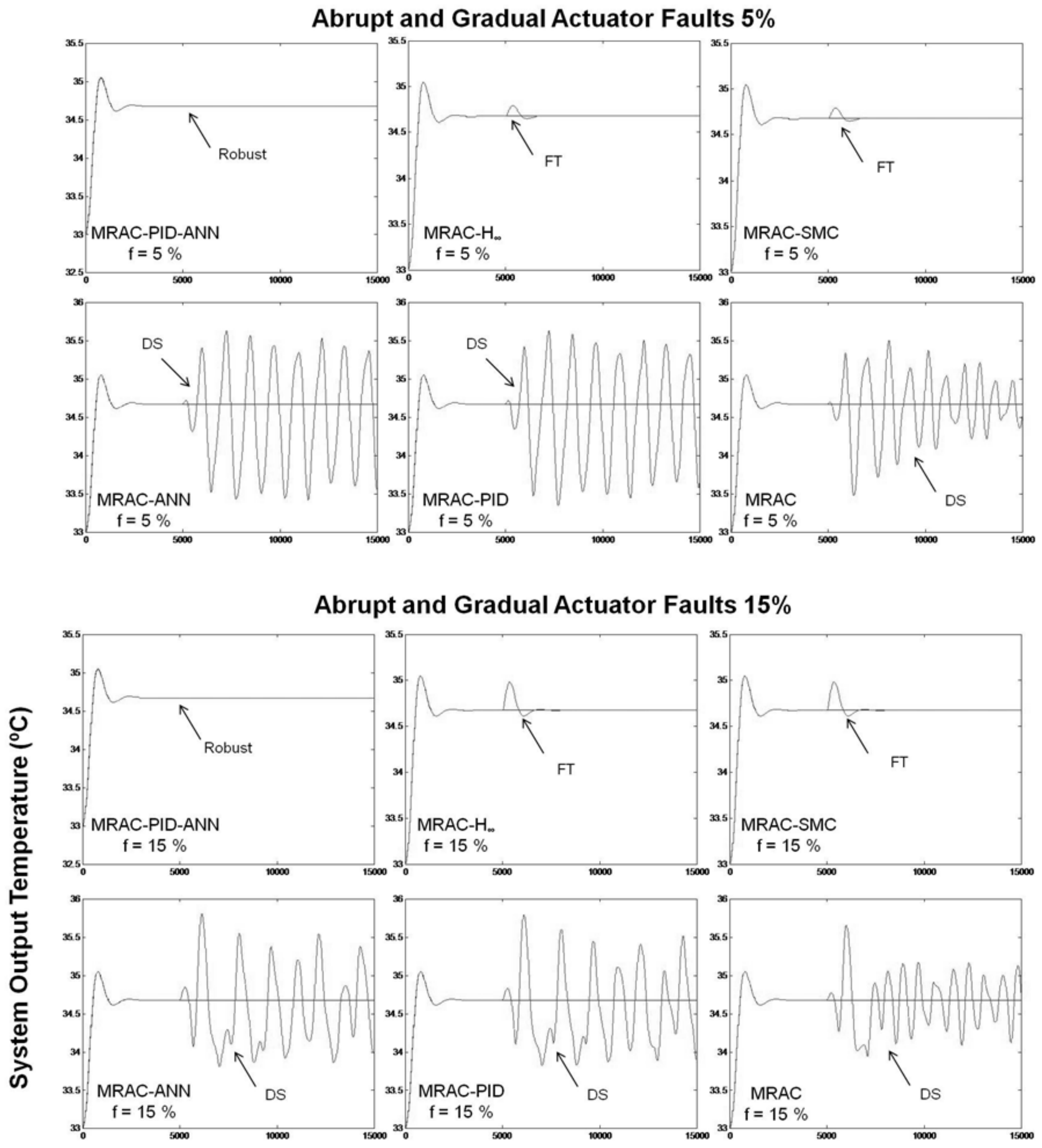


Figure 10. Abrupt and gradual sensor faults of 5%, 15% and 25%; the fault started at time 5000 secs.


Abrupt and Gradual Actuator Faults 5%
MRAC-PID-ANN f = 5 %
Robust
MRAC-H∞ f = 5 %
FT
MRAC-SMC f = 5 %
FT
MRAC-ANN f = 5 %
DS
MRAC-PID f = 5 %
DS
MRAC f = 5 %
DS
Abrupt and Gradual Actuator Faults 15%
MRAC-PID-ANN f = 15 %
Robust
MRAC-H∞ f = 15 %
FT
MRAC-SMC f = 15 %
FT
MRAC-ANN f = 15 %
DS
MRAC-PID f = 15 %
DS
MRAC f = 15 %
DS
System Output Temperature (ºC)

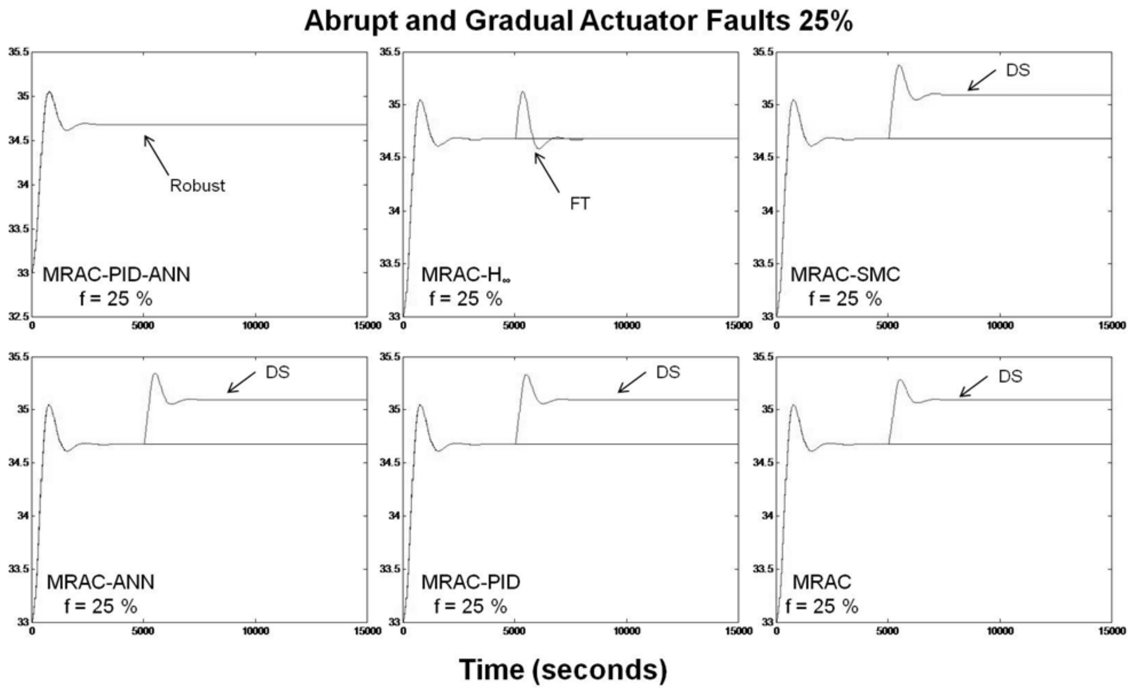


Figure 11. Abrupt and gradual actuator faults of 5%, 15% and 25%; the fault started at time 5000 secs.

In [27], an MRAC controller based on an on-line ANN and a PID traditional controller used for servo system tracking control was shown. In this scheme, the ANN controller was implemented to reduce the unknown model dynamics, the disturbances and parameter variations. The ANN weights and the MRAC parameters are updated using Lyapunov stability theory. The differences between this scheme and the one proposed in our paper is that the scheme presented in [27] did not use a GA to optimize the PID controller; also, the controller is not a complete PID but just a proportional controller and the ANN is used to reduce the unknown model dynamics and is not used as a trajectory controller.

In the case of the robust H_∞ controller, although there are some publications where the H_∞ technique has been combined with other schemes [28] [29] [30], to the best of our knowledge there are no reports concerning the combination of MRAC with H_∞.

On the other hand, for the sliding mode control, in [31] the scheme uses a passive FTC approach because the system is designed with robust or adaptive techniques to guarantee certain insensitivity to the same fault, while in our scheme, the SMC uses the information of the fault to restructure the control law. Finally, in [32] a decentralized model reference adaptive SMC based on fuzzy model for MIMO nonlinear systems was developed; the decentralized controller is able to achieve the tracking objective and is globally stable. The SMC is used to generate a control law that involves the modeling and parameter estimation errors to ensure global stability and is not used as a direct feedback controller which is the case of our proposed scheme.

5. Conclusions

The MRAC-PID-ANN methodology showed the best results because it was robust against sensor and actuator faults with a very low error between the reference model and the process model, no matter the size of the fault; this method resulted in the best scheme because it is a combination of two types of controllers: the PID controller optimized by the GA with the best parameters to handle the fault, and the ANN that was trained to follow the desired system trajectory, no matter the fault size. On the other hand, the second best methodology was the MRAC-H∞ approach because this methodology was robust against certain types of faults (5% and 15% of abrupt and gradual sensor faults) and was fault-tolerant to the rest of the fault types (25% of abrupt and gradual sensor faults and 5%, 15% and 25% of abrupt and gradual actuator faults). In addition, the MRAC-SMC resulted in a good FTC scheme because it was robust against sensor faults (5%, 15% and 25%), it was fault-tolerant for some actuator faults (5% and 15%) but it was unstable for actuators faults of 25%. The other three approaches (MRAC-ANN, MRAC-PID and MRAC) were unstable for abrupt and gradual actuator faults, and one of them (MRAC-ANN) was robust against abrupt and gradual sensor faults.

References

[1] Fradkov A., Andrievsky B., & Peaucelle, D., Adaptive Control Design and Experiments for LAAS “Helicopter” Benchmark, European Journal of Control, Vol. 14, No. 4, 2008, pp. 329-339.

[2] Stengel R., Intelligent failure-tolerant control, IEEE Control System Magazine, Vol. 11, No. 4, June, 1991, pp. 14-23.

[3] Patton R., Lopez-Toribio C., & Uppal F., Artificial intelligence approaches to fault diagnosis, IEE Colloquium on Update on Developments in Intelligent Control, 1998, pp. 3/1-312, London, United Kingdom, October.

[4] Schroder P., Chipperfield A., Fleming P., & Grum, N., Fault Tolerant Control of Active Magnetic Bearings, IEEE International Symposium on Industrial Electronics, 1998, pp. 573-578, Pretoria, South Africa, July.

[5] Yu D., Chang T., & Yu D. W., Adaptive Neural Model-Based Fault Tolerant Control for Multi-Variable Processes, Engineering Applications of Artificial Intelligence, Vol. 18, No. 4, June, 2005, pp. 393-411.

[6] Nieto J., Garza-Castañon L., Rabhi A., El Hajjaji A., & Morales-Menendez R., Vehicle Fault Detection and Diagnosis combining AANN and ANFIS, Seventh IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, 2009, pp. 1079-1084, Barcelona, Spain, June.

[7] Polycarpou M., & Helmicki A., Automated fault detection and accommodation: A learning systems approach, IEEE Transactions on Systems, Vol. 25, No. 11, November, 1995, pp. 1447–1458.

[8] Pashilkar A., Sundararajan N., & Saratchandran P., A Fault-tolerant Neural Aided Controller for Aircraft Auto-landing, Aerospace Science and Technology, Vol. 10, No. 1, January, 2006, pp. 49-61..

[9] Perhinschi M., Napolitano M., Campa G., Fravolini M., & Seanor B., Integration of Sensor and Actuator Failure Detection, dentification, and Accommodation Schemes within Fault Tolerant Control Laws, Control and Intelligent Systems, Vol. 35, No. 4, 2007, pp. 309-318.

[10] Patan K., & Korbicz J., Fault detection and accommodation by means of neural networks. Application to the boiler unit. Seventh IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, 2009, pp. 119-124, Barcelona, Spain, June.

[11] Sugawara E., Fukushi M., & Horiguchi S., Fault Tolerant Multi-layer Neural Networks with GA Training, Eighteenth IEEE International Symposium on Defect and Fault Tolerance in VLSI systems, 2003, pp. 328-335, November.

[12] Liang B., & Duan G., Robust H∞ fault-tolerant control for uncertain descriptor systems by dynamical compensators, Journal of Control Theory and Applications, Vol. 2, No. 3, August, 2004, pp. 288-292.

[13] Yen G., & Ho L., Fault Tolerant Control: An Intelligent Sliding Mode Control Strategy, Proceeding of the American Control Conference, 2000, pp. 4204-4208, Chicago, Illinois, United States, June.

[14] Nagrath J., Control Systems Engineering, 3rd Ed., Anshan Ltd, 2006, pp. 715.

[15] Whitaker H., Yamron J., & Kezer A., Design of Model Reference Adaptive Control Systems for Aircraft, Report R-164, M. I. T. Press, 1958.

[16] Nguyen H., Nadipuren P., Walker C., & Walker E., A First Course in Fuzzy and Neural Control, Chapman & Hall/CRC Press Company, 2002, pp. 165.

[17] Ruan, D., Intelligent Hybrid Systems: Fuzzy Logic, Neural Networks, and Genetic Algorithms, Kluwer Academic Publishers, 1997, pp. 9.

[18] Priddy K., and Keller P., Artificial neural networks, SPIE Press, 2005.

[19] Goldberg D., Genetic algorithms in search, optimization, and machine learning, Addison-Wesley, 1989.

[20] Khalil, H., Nonlinear Systems, 3rd Ed., Prentice Hall, 2002, pp. 552.

[21] Zames G., Feedback and optimal sensitivity: model reference transformations, multiplicative seminorms, and approximate inverse, IEEE Transactions on Automatic Control, Vol. 26, No. 2, April, 1981, pp. 301-320.

[22] Skogestad S., & Postlethwaite I., Multivariable Feedback Control. Analysis and Design, Wiley Ed., 2005, pp. 376.

[23] McFarlane D., & Glover K., Robust Controller Design Using Normalized Coprime Factor Plant Descriptions, Springer Verlag, 1989.

[24] Jia L., & Jingping J., The Model Reference Adaptive Control Based on the Genetic Algorithm, International Conference on Neural Networks,1997,pp. 783-787, Houston, Texas, United States, June.

[25] Ahmed M., Neural Net based MRAC for a Class of Nonlinear Plants, Neural Networks, Vol. 13, No. 1, January, 2000, pp. 111-124.

[26] Zhu O., Kim B., House B., & Kim, K. J., An Adaptive Controller for Wolsong NGS Bulk Liquid Zone Control of RRS, Nuclear Science Symposium, 1999, pp. 1699-1703, Seattle, Washington, United States, October.

[27] Hongjie H., & Bo Z., A New MRAC Method based on Neural Network for High-Precision Servo System, IEEE Vehicle Power and Propulsion Conference, 2008, pp. 1-5, Harbin, China, September.

[28] Lian K., Chiu C., & Liu P., Semi-Decentralized Adaptive Fuzzy Control for Cooperative Multirobot Systems with H∞ Motion/Internal Force Tracking Performance, IEEE Transaction on System, Man, and Cybernetics- Part B: Cybernetics, Vol. 32, No. 3, June, 2002, pp. 269-280.

[29] Yu W., H∞ Tracking-based adaptive fuzzy-neural control for MIMO uncertain robotic systems with time delays, Fuzzy Sets and Systems, Vol. 146, 2004, pp. 375-401.

[30] Miyasato Y., Model Reference Adaptive H∞ for Distributed Parameter Systems of Hyperbolic Type by Finite Dimensional Controllers – construction with unbounded observation operator, Forty-sixth IEEE Conference on Decision and Control, 2007, pp. 1338-1343, New Orleans, Louisiana, United States, December.

[31] Vilela J., Costa R., & Hsu L., Cooperative Actuators for Fault Tolerant Model-Reference Sliding Mode Control, IEEE International Symposium on Industrial Electronics, 2003, pp. 690-695, June.

[32] Haijun G., Tianping Z., & Qikun S., Decentralized model reference adaptive sliding mode control based on fuzzy model, Journal of Systems Engineering and Electronics, Vol. 17, No. 1, March, 2006, pp. 182-186.

Authors´ Biographies

Adriana VARGAS-MARTÍNEZ

She was born in 1983. Ms. Vargas-Martínez received the B.Sc. degree in chemical engineering with a minor in industrial systems in 2005 and a M.Sc. degree in environmental systems in 2007. Since 2007, she has been a formal student of the Doctoral Program in Engineering Sciences at Instituto Tecnológico y de Estudios Superiores de Monterrey (Technological Institute of Higher Education of Monterrey), ITESM, Campus Monterrey. Her research fields include fault detection and diagnosis, advanced control methods and fault-tolerant control.

Luis E. GARZA-CASTAÑÓN

He was born on December 15th, 1963, in Monclova, Coahuila. Mr. Garza-Castañón obtained a PhD in artificial intelligence from ITESM, Monterrey in 2001 and a M.Sc. degree in control engineering, in 1988, and in electronic systems engineering in 1986. For about 10 years, he worked in the control and automation area, in the metallurgical and metal-mechanic industry. Now, he is a full-time professor in the Mechatronics and Automation Department at ITESM, Monterrey. He has published more than 50 articles in biometrics, fault detection and diagnosis and fault-tolerant control.