Model-free adaptive control design using evolutionary-neural compensator

Leandro dos Santos Coelho \textsuperscript{a,}*, Marcelo Wichhoff Pessôa \textsuperscript{a}, Rodrigo Rodrigues Sumar \textsuperscript{b}, Antonio Augusto Rodrigues Coelho \textsuperscript{b}

\textsuperscript{a} Pontifical Catholic University of Paraná, PUCPR/CCET/PPGEPS, Automation and System Laboratory, Imaculada Conceição, 1155, Zip Code 80215-901, Curitiba, PR, Brazil
\textsuperscript{b} Federal University of Santa Catarina, Department of Automation and Systems, P.O. Box 476, Zip Code 88040-900, Florianópolis, SC, Brazil

\textbf{Keywords:}
Model-free adaptive control
Radial basis function neural network
Nonlinear systems
Differential evolution

\textbf{A B S T R A C T}

It is well-known that conventional control theories are widely suited for applications where the processes can be reasonably described in advance. However, when the plant’s dynamics are hard to characterize precisely or are subject to environmental uncertainties, one may encounter difficulties in applying the conventional controller design methodologies. In this case, an alternative design is a model-free learning adaptive control (MFLAC), based on pseudo-gradient concepts with compensation using a radial basis function neural network and optimization approach with differential evolution technique presented in this paper. Motivation for developing a new approach is to overcome the limitation of the conventional MFLAC design, which cannot guarantee satisfactory control performance when the nonlinear process has different gains for the operational range. Robustness of the MFLAC with evolutionary-neural compensation scheme is compared to the MFLAC without compensation. Simulation results for a nonlinear chemical reactor and nonlinear control valve are given to show the advantages of the proposed evolutionary-neural compensator for MFLAC design.

\copyright 2009 Elsevier Ltd. All rights reserved.

\section{1. Introduction}

Model-based control techniques are usually implemented under the assumption of good understanding of process dynamics and their operational environment. These techniques, however, cannot provide satisfactory results when applied to poorly modeled processes, which can operate in ill-defined environments. This is often the case when dealing with complex dynamic systems for which the physical processes are either highly nonlinear or are not fully understood (Karray, Gueaieb, & Al-Sharhan, 2002).

The conventional proportional-integral-derivative (PID) algorithm is still widely used in process industries. The main reason is due to their simplicity of operation, ease of design, inexpensive maintenance, low cost, and effectiveness for most linear systems. PID controllers are the most common controllers in industry. In fact, 95% of control loops use PID and the majority is PI control (Åström & Hägglund, 1995). However, its performance may degrade when applied to highly nonlinear processes, such as many chemical processes. A change in the signal and the directionality of the process gain is a complex practical situation and, so, still becoming complex the design of a control system (Bisowarno, Tian, & Tade, 2003).

Because the PID controller has difficulties when applied to highly nonlinear processes, there is a significant incentive to apply modern insights to the improvement of controller design methods. In this context, most nonlinear control strategies require a nonlinear dynamic model of the process to be controlled. Unfortunately, first-principles modeling is difficult to apply to processes that are poorly understood and/or highly complex. An alternative approach is to develop nonlinear adaptive control strategies.

The field of nonlinear adaptive control has had a rapid development in the last decades. The papers by Narendra and Parthasarathy (1990), Polycarpou and Ioannou (1991), Sanner and Slotine (1992) and Ordóñez and Passino (2001) gave birth to an important branch of adaptive control theory, the nonlinear function approximation based control (which includes neural and fuzzy approaches).

The traditional adaptive control theory has two main problems, the dependence on the mathematical model structure of the controlled plant and mismatch plant dynamics. Recently, motivated by the rapidly developed advanced microelectronics and digital processors, more studies have focused on the development of nonlinear adaptive control strategies using intelligent control algorithms that can be directly applied to complex processes whose dynamics are poorly modeled and/or have high nonlinearity. These fields widely combine adaptive control theories with intelligent paradigms such as artificial neural networks (Dote & Ovaska, 2001), fuzzy systems (Passino & Yurkovich, 1998), and evolutionary algorithms (Fleming & Purshouse, 2002; Jamshidi, Coelho, Krohling, & Fleming, 2003) in controllers design.
An adaptive control approach, also called “learning control”, is categorized in this type of approach. In a basic sense, learning control can be considered as a technique that is able to solve control problems that lack sufficient a priori information for one to perform a complete and fixed control system design in advance.

The contribution of this paper is to merge for nonlinear systems, the model-free learning adaptive control structure (Hou & Huang, 1997; Hou, Han, & Huang, 1998) with the design based on a radial basis function neural network (RBF-NN) (Billings & Zheng, 1999; Lo, 1998) compensation and optimization procedure using differential evolution (Storn & Price, 1995).

Nonlinear techniques using artificial neural networks have received a great deal of attention and have been employed in many applications in nonlinear system identification and complex modeling (Billings & Zheng, 1999; Hussain, Knowles, Lisboa, & El-Deredy, 2008; Lo, 1998; Mcloone, Brown, Irwin, & Lightbody, 1998; Otawara et al., 2002). Artificial neural networks have the ability to approximate arbitrary linear or nonlinear mapping by means of learning. Because, of the learning ability, the neural networks have been developed to compensate for the nonlinearities and uncertainties in design of control systems. Particularly, the RBF-NNs are flexible modeling tools that have the ability to rapidly learn complex patterns and adapt to changes.

On the other hand, over the past few years, the global optimization field has been very active, producing different kinds of techniques for optimization in the continuous domain. Stochastic and heuristic optimization techniques such as evolutionary algorithms (EAs) have emerged as efficient tools for global optimization and have been applied to a number of numerical optimization problems in diverse fields in recent years (Cao, Feng, & Ma, 2007; Chang, 2007; Chang, Yang, Liao, & Yan, 2008; Chen, Chang, Yan, & Liao, 2008; Cao et al., 2008; Grosman & Lewin, 2004; Guri, Bhatcharya, & Gupta, 2005; Jang, Hahn, & Hall, 2005; Kately, Bhan, Caruthers, Delgass, & Venkatasubramanian, 2004; Lee & Doong, 2008; Link, Vera, Weuster-Botz, Darias, & Franco-Lara, 2008; Mwembehs, Kent, & Salhi, 2004; Passino, 2001; Ponce-Oregia, Serna-González, & Jiménez-Gutiérrez, 2008; Rada, 2008; Su & Hou, 2008; Wongrat, Srinophakun, & Srinophakun, 2005). EAs use a population of structures (individuals) in which each one is a candidate solution for the optimization problem. Since they are population-based methods, they make a parallel search of the space of possible solutions, and are less susceptible to local minima. Most current approaches of EAs descend from principles of main methodologies: genetic algorithms, evolutionary programming, evolution strategy, and differential evolution (DE). DE as developed by Storn and Price (1995) is one of the best EAs, and has proven to be a promising candidate to solve real-valued optimization problems. The computational algorithm of DE is simple and easy to implement, with only a few parameters required to be set by a user.

This paper is organized as follows. In Section 2, the control law of MFLAC is reviewed. In Section 3, the MFLAC design with neural compensation is presented. In Section 4, the optimization by differential evolution is discussed. Simulation results for controlling a continuous stirred tank reactor and a nonlinear control valve are presented in Section 5. Conclusion and future works are given in Section 6.

### 2. Model-free learning adaptive control

In this paper, the direct adaptive control of the following general discrete SISO (Single-Input and Single-Output) nonlinear system is considered

\[
y(k + 1) = f(y(k), \ldots, y(k - n_y), u(k), \ldots, u(k - n_u)) \tag{1}
\]

where \( n_y \) and \( n_u \) are the orders of system output, \( y(k) \), and input, \( u(k) \), respectively, and \( f(\cdot) \) is a general nonlinear function. The plant (Eq. (1)) can be rewritten as follows:

\[
y(k + 1) = f(Y(k), u(k), U(k - 1)) \tag{2}
\]

where \( Y(k) \) and \( U(k - 1) \) are the sets of system outputs and inputs up to sampling instant \( k \) and \( k - 1 \).

The following assumptions are considered about the controlled plant: (A1) the system (1) and (2) is observable and controllable; (A2) the partial derivative of \( f(\cdot) \) with respect to control input \( u(k) \) is continuous; and (A3) the system (1) is generalized Lipschitz.

For a nonlinear system (2), satisfying assumptions (A1–A3), then there must exist \( \phi(k) \), called pseudo-gradient vector, when control change \( \Delta u(k) \neq 0 \), and

\[
\Delta y(k + 1) = \phi^T(k)\Delta u(k) \tag{3}
\]

where the control change \( \Delta u(k) = u(k) - u(k - 1) \); \( \| \phi(k) \| \leq L \), and \( L \) is a constant.

Details of the theoretical basis and the mathematical proof of the MFLAC are given in Hou and Huang (1997) and Hou et al. (1998). In this proof, the equation \( y(k + 1) = f(Y(k), u(k), U(k - 1)) \) gives

\[
\Delta y(k + 1) = f(Y(k), u(k), U(k - 1)) - f(Y(k), u(k - 1), U(k - 2)) \tag{4}
\]

or

\[
\Delta y(k + 1) = f(Y(k), u(k), U(k - 1)) - f(Y(k), u(k - 1), U(k - 1)) + f(Y(k), u(k), U(k - 1)) - f(Y(k - 1), u(k - 1), U(k - 2)) \tag{5}
\]

Using assumption (A2) and the mean value theorem, the Eq. (5) gives

\[
\Delta y(k + 1) = \frac{df}{du(k)}\Delta u(k) + \xi(k) \tag{6}
\]

where \( \frac{df}{du(k)} \) denotes the value of gradient vector of \( f(Y(k), u(k), U(k - 1)) \) with respect to \( u \) at some point between \( u(k - 1) \) and \( u(k) \), and given by

\[
\xi(k) = f(Y(k), u(k - 1), U(k - 1)) - f(Y(k - 1), u(k - 1), U(k - 2)) \tag{7}
\]

Considering the following equation

\[
\zeta(k) = \eta^T(k)\Delta u(k) \tag{8}
\]

where \( \eta(k) \) is a variable. Since condition \( \Delta u(k) \neq 0 \), Eq. (8) must has solution \( \eta(k) \).

Let

\[
\phi(k) = \frac{df}{du(k)} + \eta(k) \tag{9}
\]

from (8) and (9), then (7) can be rewritten as

\[
\Delta y(k + 1) = \phi^T(k)\Delta u(k) \tag{10}
\]

This is the same as (3). In this case, by using (3) and assumption (A3), and \( \Delta u(k) \neq 0 \), we have

\[
|\phi^T(k)\Delta u(k)| \leq L\|\Delta u(k)\| \tag{10}
\]

Hence \( \| \phi(k) \| \leq L \). For the learning control law algorithm, a weighted one-step-ahead control input cost function is adopted, and given by

\[
J(u(k)) = y(k + 1) - y_c(k + 1)^2 + \lambda\|\Delta u(k)\|^2 \tag{11}
\]

For the control design, where \( y_c(k + 1) \) is the expected system output signal (true output of the controlled plant), and \( \lambda \) is a positive weighted constant. The Eq. (3) can be rewrite as follows

\[
y(k + 1) = y(k) + \phi^T(k)\Delta u(k) \tag{12}
\]
Substituting (12) into (11), differentiating (11) with respect to \( u(k) \), solving the equation \( \partial J(u(k))/\partial u(k) = 0 \), and using the matrix-inversion lemma gives the control law as follows:

\[
    u(k) = u(k-1) + \frac{\rho_d \phi(k)}{\hat{l} + \|\phi(k)\|^2} [y_r(k+1) - y(k)]
\]

(13)

The control law (13) is a kind of control that has no relationship with any structural information (mathematical model, order, structure, etc.) of the controlled plant. It is designed only using I/O data of the plant.

The cost function proposed by Hou et al. (1998) for parameter estimation is used in this paper as

\[
    J(\phi(k)) = [y(k) - y(k-1) - \phi^T \Delta u(k-1)]^2 + \mu \|\phi(k) - \hat{\phi}(k-1)\|^2
\]

(14)

Using the similar procedure of control law equations, we can obtain the parameter estimation algorithm as follows:

\[
    \hat{\phi}(k) = \hat{\phi}(k-1) + \frac{\eta \Delta u(k-1)}{\mu + \|\Delta u(k)\|^2} [\Delta y(k) - \phi^T(k-1) \Delta u(k-1)]
\]

(15)

Summarizing, the MFLAC scheme is

\[
    \hat{\phi}(k) = \hat{\phi}(1) \text{ if } \text{sign}(\phi(1)) \neq \text{sign}(\hat{\phi}(1))
\]

(16)

\[
    \phi(k) = \hat{\phi}(1) \text{ if } \|\phi(k)\| \geq M, \text{ or } \|\phi(k)\| \leq \varepsilon
\]

(17)

\[
    u(k) = u(k-1) + \frac{\rho \phi(k)}{\hat{l} + \|\phi(k)\|^2} [y_r(k+1) - y(k)]
\]

(18)

where step-size series \( \rho \) and \( \eta \), and the weighted constants \( \hat{l} \) and \( \mu \) are design parameters optimized by differential evolution in this paper. The parameter \( \varepsilon \) is a small positive constant (adopted 0.00001), \( M \) is adopted with value 10, and \( \phi(1) \) is the initial estimation value of \( \phi(k) \).

3. MFLAC design with RBF-NN compensation

It is well-known that a simplified linear/linearizable system can often represents dynamics of a nonlinear plant (Funahashi, 1989; Nounou & Passino, 2004; Sastry & Isidori, 1989; Wang, Ge, & Lee, 2004) around its operational point and provides good basis for the control design. Furthermore, a linear controller can work as good as any nonlinear controller when a system behaves sufficiently close to its operational point. However, the challenge for implementing a model-free controller to dealing plants with nonlinearities and time varying behavior is a complex task. In this paper, a new approach of neural compensation is applied to reduce the control errors of a MFLAC which are not completely known. Because of the learning ability, compensation based on neural network is developed to deal with the nonlinearities and uncertainties in a control system.

Artificial neural networks are originally inspired by the functionality of biological neural networks, which are able to learn complex functional relations based on a limited number of training data. Artificial neural networks may serve as black-box models of nonlinear multivariable dynamic systems and may be trained using input–output data observed from the system. The usual artificial neural network consists of multiple simple processing elements, called neurons, interconnections among them and the weights attributed to the interconnections. The relevant information of this methodology is stored in the weights.

In this work, the neural network used is a RBF-NN. RBF-NNs offer a framework for the compensation nonlinear, because they are simple topological structure and they can get a precise behavior in nonlinear dynamics approximations. The compensation based on RBF-NN is a single hidden layer feedforward setup and has the form

\[
    u_{NN}(k,x) = \sum_{i=1}^{n} \omega_i G(\|x(k) - t_i\|), \quad 1 \leq t \leq N
\]

(20)

where \( k \) is the current sample, \( x \in \mathbb{R}^m \) is the input vector, \( G(\cdot) \) is the continuous function from \( \mathbb{R}^1 \) to \( \mathbb{R} \), called the radial basis function, \( t_i \in \mathbb{R}^m \) are some appropriately chosen centers from the given data and \( \omega_i \) are constant coefficients or weights of the network with \( n < N \), where \( N \) is the number of input and output samples. For the nonlinearity \( G(\cdot) \), several functions which form good radial basis function approximation can be chosen. In this paper, the Gaussian function was selected and is given by

\[
    G(v) = e^{-v^2/\sigma^2}
\]

(21)

where \( v \equiv \|x - t_i\| \) and the variance \( \sigma \) is a parameter specified by the user or by optimization.

Learning of the RBF-NN corresponds to determination of the centers \( t_i \), variance (spread) \( \sigma \), and the coefficients \( \omega_i \). Since RBF-NN networks are linear-in-the-parameters for fixed \( t_i \) and \( \sigma \), the coefficients \( \omega_i \) can be determined using the linear least-squares method. The choice of the values of \( t_i \) and \( \sigma \) is crucial for the performance of the neural compensation.

An optimization algorithm was used to find the proper values of these RBF-NN parameters. The parameter optimization method adopted to tune the centers and variances of the RBF-NN (with two Gaussian functions for each input) is the differential evolution. The neural compensation has three normalized input vectors: \( x(k-1) \), control change \( \Delta u(k) = u(k) - u(k-1) \), and reference signal \( y_r(k-ny) \), where \( ny = 1 \) is adopted in this work. The output (incremental control) of the neural compensation is \( \Delta u_{NN}(k,x) = u_{NN}(k,x) - u_{NN}(k-1,x) \), as shown in Fig. 1.

4. Optimization of controller design using differential evolution approach

The choice of the differential evolution (DE) algorithm for optimization of MFLAC and MFLAC-NN is based on its useful features such as (Cheng & Hwang, 2001): (i) it is a stochastic search algorithm that is originally motivated by the mechanisms of natural selection, (ii) it is less likely become trapped in a local optimum because it searches for the global optimal solution by manipulating a population of candidate solutions, and (iii) it is effective for solving the optimization problems with nonsmooth objective functions as it does not require the derivative information.

In this paper, a DE-based optimization technique is adopted to obtain \( \phi(1) \), \( \rho \), \( \eta \), \( \hat{l} \), \( \mu \) for the MFLAC design and the centers and variances of RBF-NN used in MFLAC-NN. The RBF-NN learning of MFLAC-NN is realized using an off-line training procedure.

4.1. Fundamentals of differential evolution approach

EAs are a class of stochastic search and optimization methods. These algorithms, based on the principles of natural biological evolution, have received considerable and increasing interest over the past decade. EAs operate on a population of potential solutions, applying the principle of survival of the fittest to produce successively better approximations to a solution. EAs encompass a range
of different ‘nature-inspired’ methods, including genetic algorithms, evolution strategies, evolutionary programming, genetic programming, differential evolution and their variants (Bäck, Fogel, & Michalewicz, 1997; Michalewicz, 1996).

There are many variants of EAs, but the main differences rely on: how individuals are represented, the genetic operators that modify individuals (especially mutation and crossover), and the selection procedure.

DE is a simple but powerful stochastic global optimizer over continuous spaces proposed by Storn and Price (1995, 1997), whose main design emphasis is real parameter optimization using floating point encoding. The crucial idea behind DE is a scheme for generating trial parameter vectors. DE creates new candidate solutions by combining the parent individual and several other individuals of the same population. A candidate replaces the parent only if it has better fitness.

DE is based on a mutation operator, which adds an amount obtained by the difference of two randomly chosen individuals of the current population, in contrast to most EAs, in which the mutation operator is defined by a probability function.

The different variants of DE are classified using the following notation: DE/α/β/δ, where α indicates the method for selecting the parent chromosome that will form the base of the mutated vector, β indicates the number of difference vectors used to perturb the base chromosome, and δ indicates the recombination mechanism used to create the offspring population. The bin acronym indicates that the recombination is controlled by a series of independent binomial experiments.

The fundamental idea behind DE is a scheme whereby it generates the trial parameter vectors. In each step, the DE mutates vectors by adding weighted, random vector differentials to them. If the cost of the trial vector is better than that of the target, the target vector is replaced by the trial vector in the next generation. The variant implemented here was the DE/rand/1/bin, which involved the following steps and procedures (Coelho & Mariani, 2006):

Step 1: Parameter setup
The user chooses the parameters of population size, the boundary constraints of optimization variables, the mutation factor (f_m), the crossover rate (CR), and the stopping criterion of maximum number of iterations (generations), t_max.

Step 2: Initialization of an individual population
Set generation t = 0. Initialize a population of i = 1, ..., M individuals (real-valued n-dimensional solution vectors) with random values generated according to a uniform probability distribution in the n dimensional problem space. These initial individual values are chosen at random from within user-defined bounds (boundary constraints).

Step 3: Evaluation of the individual population
Evaluate the objective function (minimization problem is this paper) of each individual.

Step 4: Mutation operation (or differential operation)
Mutation is an operation that adds a vector differential to a population vector of individuals according to the following equation:

\[ z_i(t+1) = x_i(t) + f_m \cdot [z_{rnbr}(t) - x_i(t)] \]  

(22)

In the above equation, i = 1, 2, ..., N is the individual’s index of population; t is the time (generation); \( x_i(t) \) stands for the position of the ith individual of population of N real-valued n-dimensional vectors; \( z_i(t) \) stands for the position of the ith individual of a mutant vector; \( f_m > 0 \) is a real parameter, called mutation factor, which controls the amplification of the difference between two individuals so as to avoid search stagnation. The mutation operation randomly select the target vector \( x_i(t) \), with \( i \neq i \). Then, two individuals \( x_i(t) \) and \( x_{rnbr}(t) \) are randomly selected with \( i_1 \neq i_2 \neq i_3 \neq i \), and the difference vector \( x_i(t) - x_{rnbr}(t) \) is calculated.

Step 5: Crossover (or recombination) operation
Following the mutation operation, crossover is applied in the population. For each mutant vector, \( z_i(t+1) \), an index \( mbr(i) \in \{1, 2, ..., n\} \) is randomly chosen using a uniform distribution, and a trial vector \( u_i(t+1) = [u_{i1}(t+1), u_{i2}(t+1), ... , u_{in}(t+1)] \), is generated via

\[ u_i(t+1) = \begin{cases} z_i(t+1) & \text{if } randb(j) \leq CR \text{ or } j = mbr(i), \\ x_i(t) & \text{otherwise.} \end{cases} \]  

(23)

where \( j = 1, 2, ..., n \) is the parameter index; \( x_i(t) \) stands for the ith individual of jth real-valued vector; \( z_i(t) \) stands for the ith
individual of \( j \)th real-valued vector; \( u_i(t) \) stands for the \( i \)th individual of \( j \)th real-valued vector after crossover operation; \( \text{randb}(j) \) is the \( j \)th number generated using a uniform distribution in the range \([0,1]\); \( \text{CR} \) is a crossover rate in the range \([0,1]\).

Step 6: Selection operation

Selection is the procedure of producing better offspring. To decide whether or not the vector \( v_i(t+1) \) should be a member of the population comprising the next generation, it is compared with the corresponding vector \( x_i(t) \). Thus, if \( f \) denotes the objective function under minimization, then

\[
x_i(t+1) = \begin{cases} u_i(t+1), & \text{if } f(u_i(t+1)) < f(x_i(t)), \\ x_i(t), & \text{otherwise} \end{cases}
\]

(24)

In this case, the cost of each trial vector \( v_i(t+1) \) is compared with that of its parent target vector \( x_i(t) \). If the cost, \( f \), of the target vector \( x_i(t) \) is lower than that of the trial vector, the target is allowed to advance to the next generation. Otherwise, the target vector is replaced by the trial vector in the next generation.

Step 7: Verification of stop criterion

Set the generation number for \( t = t+1 \). Proceed to Step 3 until a stopping criterion is met, usually \( G_{\text{max}} \). The stopping criterion depends on the type of problem.

Details of the adopted DE and its operators can be obtained in Cheng and Hwang (2001) and Storn and Price (1997). In this work, the objective of the DE optimization in the MFLAC design is to minimize the objective function given by

\[
f = \sum_{k=1}^{N} |y(k) - y_r(k)| + 0.01 \cdot |u(k) - u(k-1)|^2
\]

(25)

where \( u(k) \) is the control signal, \( y(k) \) is the process output, \( y_r(k) \) is the reference (setpoint), and \( N \) is the number of collected samples.

Fig. 3. Static curve of the CSTR process.

Fig. 4. Input and output signals for the MFLAC in case study 1.
In this paper, the setup of DE used in MFLAC and MFLAC-NN designs was the following: constant mutation factor given by \( f_m = 0.4 \), a crossover rate of \( CR = 0.8 \), population size \( P = 20 \) and stop criterion, \( t_{\text{max}} = 200 \) generations. The space search adopted in DE setup to MFLAC design is: \( 0 < x_1 < 6, 0.5 < 6, 0.00001 < \mu < 1.00 \) and \( 1.00 < \mu < 5.00 \).

### 5. Case studies and simulation results

In this paper, the setup of DE used in MFLAC and MFLAC-NN designs was the following: constant mutation factor given by \( f_m = 0.4 \), a crossover rate of \( CR = 0.8 \), population size \( P = 20 \) and stop criterion, \( t_{\text{max}} = 200 \) generations. The space search adopted in DE setup to MFLAC design is: \( 0 < x_1 < 6, 0.5 < 6, 0.00001 < \mu < 1.00 \) and \( 1.00 < \mu < 5.00 \).

#### 5.1. Case study 1: Control of a nonlinear continuous stirred tank reactor

The first case study consists of a nonlinear Continuous Stirred Tank Reactor (CSTR) process as shown in Fig. 2.

Equations of the reactor dynamic, which represent a complex behavior, is considering three operational ranges, and are given by Chen and Peng (1997):

\[
\frac{dx_1}{dt} = -x_1 + D_a \cdot (1 - x_1) \cdot e^{x_2/\alpha} + u
\]

\[
\frac{dx_2}{dt} = -(1 - \beta) \cdot x_2 + B \cdot D_a (1 - x_1) \cdot e^{x_2/\alpha} + \beta \cdot u
\]

where \( x_1 \) and \( x_2 \) represent the dimensionless reactant concentration and reactor temperature, respectively. The control input, \( u \), is the dimensionless cooling jacket temperature. The physical parameters of the CSTR model equations are \( D_a, \varphi, B \) and \( \beta \) which corresponds to Damköhler number, the activation energy, heat of reaction and heat transfer coefficient, respectively. Considering system nominal parameters: \( D_a = 0.072, \varphi = 20, B = 8 \) and \( \beta = 0.3 \), the system shows unstable behavior in open-loop. The simulation of reactor dynamic behavior is performed by converting Eqs. (26) and (27) to discrete system equations, by using Euler method (Åström & Wittmark, 1994) as follows:

**Table 1**

<table>
<thead>
<tr>
<th>Variables</th>
<th>MFLAC</th>
<th>MFLAC-NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of ( u )</td>
<td>1.3069</td>
<td>1.3721</td>
</tr>
<tr>
<td>Variance of ( u )</td>
<td>8.7415</td>
<td>4.6216</td>
</tr>
<tr>
<td>Mean of error</td>
<td>0.0236</td>
<td>0.0230</td>
</tr>
<tr>
<td>Variance of error</td>
<td>0.4072</td>
<td>0.0397</td>
</tr>
</tbody>
</table>

**Fig. 5.** Input and output signals for the MFLAC-NN in case study 1.

---

where $T_s$ represents the sampling time (selected by $T_s = 0.2$ s). The relationship between system output and control input can be acquired by solving Eq. (28) on Eq. (29), with $k = k + 1$:

$$y(k + 1) = x_2(k) + T_s \left[ -(1 - \beta) \cdot x_2(k) + B \cdot D_a(1 - x_1(k)) \cdot e^{-\frac{y(k)}{C_0}} + \beta \cdot u(k) \right]$$

(31)

Fig. 3 shows the static curve of $u$ versus $y_{ss}$ (output signal in steady-state) of the CSTR. Due to this static behavior, which is the relationship of steady-state input and output signals, it has distinct operational points. As shown in Fig. 3, the first range is defined between 0 and 1.5, the second between 1.5 and 5.5 and the last one between 5.5 and 7.

For the MFLAC design, the optimization procedure by DE obtains

- $\phi(1) = 0.0001, \rho = 2.3107, \eta = -0.05598, \lambda = 0.000356, \mu = 0.068943$
- objective function $f = 239.5402$ (best result in 30 runs using DE).

For the MFLAC-NN, the DE determines the design parameters of MFLAC (five design parameters) plus six centers and six variances of RBF-NN compensator with two Gaussian functions in hidden layer.

Simulation results for servo and regulatory responses of MFLAC and MFLAC-NN are shown in Figs. 4 and 5, respectively. Regulatory behavior analysis of the controllers was based on parametric changes in the plant output when:

- sample 140: $y(k) = y(k) - 0.4$;
- sample 280: $y(k) = y(k) + 0.4$;

Fig. 7. Input and output signals for the MFLAC in case study 2.
Simulation results presented in Figs. 4 and 5 show that the MFLAC without compensation presents poor performance for servo and regulatory cases of the CSTR, mainly to the reference $y_r = 3$. However, the proposed MFLAC-NN enables the MFLAC to have precise control performance. In this case, the RBF-NN compensation is very useful when the MFLAC cannot properly handle large disturbances and parameter changes. In Table 1, a summary of simulation results and performance of the controllers is presented. The proposed MFLAC-NN controller is able to maintain better set point tracking performance and disturbance rejection capabilities over the range of nonlinear operation of CSTR.

5.2. Case study 2: control of a nonlinear control valve

The control valve system is an opening with adjustable area. Normally it consists of an actuator, a valve body and a valve plug. The actuator is a device that transforms the control signal to movement of the stem and valve plug. Wigren (1993) describes the plant where the control valve dynamic is described by a Wiener model (the nonlinear element follows linear block) and it is given by

\[
x(k) = 1.5714x(k - 1) + 0.6873x(k - 2) + 0.0616u(k - 1) + 0.0543u(k - 2)
\]
\[
y(k) = f_n[x(k)] = \frac{x(k)}{\sqrt{0.10 + 0.90|x(k)|^2}}
\]

where $u(k)$ is the control pressure, $x(k)$ is the stem position, and $y(k)$ is the flow through the valve which is the controlled variable. The input to the process, $u(k)$, is constrained between [0; 1.2]. The nonlinear behavior of the control valve described by Eq. (33) is shown by static curve presented in Fig. 6.

Simulation results for servo (optimization phase of design) and regulatory (validation phase of design) responses of MFLAC are shown in Figs. 7 and 8, respectively. Regulatory behavior analysis of the MFLAC and MFLAC-NN were based on parametric changes in the plant output when:

- sample 40: $y(k) = y(k) + 0.2$;
- sample 120: $y(k) = y(k) - 0.2$;
- sample 210: $y(k) = y(k) - 0.2$; and
- sample 310: $y(k) = y(k) - 0.2$.

For the MFLAC design, the optimization procedure by DE obtains $\phi(1) = 0.020027$, $\rho = 1.990504$, $\eta = 0.736865$, $\lambda = 0.468514$, $\mu = 0.209603$ and objective function $f = 5.4280$ (best result in 30 runs using DE). For the MFLAC-NN, in similar way of first case

![Fig. 8. Input and output signals for the MFLAC-NN in case study 2.](image-url)
Table 2
Indices for the MFLAC and MFLAC-NN design when applied in control valve.

<table>
<thead>
<tr>
<th></th>
<th>Servo behavior</th>
<th>Regulatory behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MFLAC</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean of $u$</td>
<td>0.4227</td>
<td>0.4322</td>
</tr>
<tr>
<td>Variance of $u$</td>
<td>0.0681</td>
<td>0.0748</td>
</tr>
<tr>
<td>Mean of error</td>
<td>0.0112</td>
<td>0.0074</td>
</tr>
<tr>
<td>Variance of error</td>
<td>0.0016</td>
<td>0.0036</td>
</tr>
<tr>
<td><strong>MFLAC-NN</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean of $u$</td>
<td>0.4281</td>
<td>0.4351</td>
</tr>
<tr>
<td>Variance of $u$</td>
<td>0.0705</td>
<td>0.0854</td>
</tr>
<tr>
<td>Mean of error</td>
<td>0.0071</td>
<td>0.0022</td>
</tr>
<tr>
<td>Variance of error</td>
<td>0.0008</td>
<td>0.0018</td>
</tr>
</tbody>
</table>

study, the DE determines the design parameters of MFLAC plus six centers and six variances of RBF-NN compensator with two Gaussian functions in hidden layers.

Numerical results presented in Figs. 7 and 8 show that the MFLAC and MFLAC-NN approaches have good control performance in servo control. However, the results of regulatory behavior demonstrated the proposed MFLAC-NN scheme can solve control more effectively the nonlinear control valve that the MFLAC. In Table 2, a summary of simulation results and performance of the MFLAC and MFLAC-NN designs based on DE optimization is presented.

6. Conclusion and future works

Many systems in the industrial process include nonlinearity, which are often unknown and time varying. Classical control approaches frequently present poor performance since the unknown nonlinearity is a hard task to tackle in that simple domain. In this case, the performance of conventional adaptive algorithms relies heavily also on the accuracy of the a priori information about plant structure, process order, and the system delay.

Artificial neural networks have been used mainly in identification of nonlinear systems, where it can be viewed as the nonlinear dynamic mapping of input to the output. In this work, a MFLAC strategy based on pseudo-gradient concepts with compensation using RBF-NN and DE optimization called MFLAC-NN has been developed. The application and benefits of this novel adaptive strategy is demonstrated through simulation to two case studies of nonlinear processes.

Numerical results for controlling a CSTR process and a nonlinear control valve have shown the correctness and efficiency of the proposed MFLAC-NN scheme. Based on the simulation results, it has been noticed that the performance of the MFLACC-NN is better than the performance of the MFLAC without compensation to regulatory tests in control of a CSTR process and a control valve.

However, it still has a distance to industrial applications and more practical essays must be done. A further investigation can be directed to analyze the RBF-NN compensation for model-free adaptive control methods (Sofonov & Cabral, 2001; Spall & Cristion, 1998) in theoretical issues such as robustness and stability convergence proof.

Acknowledgement

This work was partially supported by the National Council of Scientific and Technologic Development of Brazil – CNPq – under Grant 309646/2006-5/PQ.

References


