Design of a fuzzy servo-controller

M.S. Ahmeda,*, U.L. Bhattib, F.M. Al-Sunnib, M. El-Shafeib

aAdvanced E/E Development, DaimlerChrysler Corporation, CIMS # 484-02-15, 800 Chrysler Drive East, Auburn Hills, MI 48326-2757, USA
bDepartment of Systems Engineering, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia

Received 22 July 1998; received in revised form 3 March 2000; accepted 5 July 2000

Abstract

A design method of a fuzzy servo-controller for nonlinear plants has been presented. The proposed method is an error feedback scheme, where the controller also receives signals representing the plant operating points. Integrator is used in the control loop to ensure setpoint following, low-frequency disturbance rejection, and to enhance the robustness of the closed-loop system. A training scheme for the fuzzy controller is derived that minimizes the output error between a reference model and the plant. The training is conducted off-line for a class of setpoints conforming to the normal operating condition of the plant. Results of simulation studies are also presented. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Engineering; Control theory; Optimization

1. Introduction

Linear control theory is a well-established subject having a large number of powerful techniques and with a long history of successful industrial applications. However, in practice, most plants are nonlinear. The application of linear control theory to these plants rely on the key assumption of small range of operation for the linear model to be valid. When the required operation range is large, a linear controller is likely to perform unsatisfactorily.

In many cases, it would be advantageous to design controller for the plant considering its nonlinearities. This would result in an improved control. The response of a nonlinear plant generally cannot be shaped to a desired pattern using a linear controller. Consequently, a nonlinear controller is required to satisfactorily control such plants. However, one of the main difficulty in designing the nonlinear controller is the lack of a general structure for it.

* Correspondence address: 414 Coachman Dr. #3A, Troy, MI 48083, USA. Tel./fax: +1-248-528-9813.
E-mail address: msa7@daimlerchrysler.com (M.S. Ahmed).

0165-0114/01/$-see front matter © 2001 Elsevier Science B.V. All rights reserved.
PII: S0165-0114(00)00121-4
Nonlinear controller design may be viewed as a nonlinear function approximation problem. In recent years, the fuzzy logic (FL) has come as an important element in designing the nonlinear controllers. A fuzzy logic system (FLS) can be viewed as a nonlinear mapping function. It can be shown that any given real function on a compact set can be approximated by an FLS with arbitrary accuracy [11,19]. This property often is referred to as the universal approximation property. The other advantage of using an FLS is that the linguistic information enables the construction of fuzzy systems, which can be utilized in obtaining an initial fuzzy controller and as well in its refinements.

Fuzzy logic was introduced by Zadeh [21] in 1965. Applications to control theory sprung up in the 1970s when Mamdani et al. [10,13] applied fuzzy logic in process control. Although fuzzy control has been in use for about a decade it was only recently that Kosko et al. [11] and Wang et al. [19] provided the mathematical foundation by showing that fuzzy systems are universal approximators. A large number of applications in the decade of 1980 revolutionized the market especially in the domain of the consumer products [16].

By the advent of 1990s the trend of combining fuzzy logic with the neural networks started receiving attention. Most of the approaches based their work on the Takagi–Sugeno model [18] which is essentially a generalization of the fuzzy logic concept. The idea of blending the fuzzy logic and neural networks was presented among others by Ching et al. [7]. The training of the FLC both off-line and on-line began on a large scale in the nineties.

Ayoubi [6] proposed a method to train the fuzzy network using the back-propagation in which the desired training set is needed as a prerequisite. Watanabe et al. [20] used fuzzy Gaussian neural networks for control of a mobile robot. A backpropagation rule has been used for the training. Spooner et al. [17] presented a fuzzy logic adaptive control technique for nonlinear systems using a Lyapunov function. However, the design is restricted to feedback linearizable plants.

In this article, we present the design of a fuzzy logic servo-controller for a wide class of nonlinear plants. It is assumed that the plant is described by a known nonlinear function that is first-order differentiable and that a stabilizing controller of the assumed nonlinear structure exists. The proposed control configuration stems from the well-known and proven control strategies in the linear systems. The control scheme uses integrators to ensure low-frequency command tracking, low-frequency disturbance rejection and to enhance robustness of the closed-loop system.

The proposed control scheme is based on error feedback. The error feedback imitates the classical error feedback systems in linear control design. However, the controller also receives signals to determine the plant operating points. These signals may include, plant input–output and variables representing other ambient conditions. The fuzzy controller is trained off-line by minimizing the error between the plant output and the output of a reference model while the training setpoints conform to the normal operating condition of the plant. The feasibility and effectiveness of the proposed scheme are illustrated by conducting simulation studies.

The organization of the paper is as follows: Section 2 introduces the control problem. Section 3 describes the proposed fuzzy control scheme. This section also examines the use of integrators in controlling nonlinear plants. Section 4 poses the control design as an optimization problem and as well describes the optimization method. Section 5 discusses some heuristics gained from the computational experience. Section 6 presents the simulation study.

2. Problem statement

Consider a discrete-time nonlinear minimum phase plant described by

\[ y(t) = f\{y(t-1), \ldots, y(t-n), u(t-\tau), \ldots, u(t-n)\}, \]

where \( u(t) \) and \( y(t) \), respectively, are the scalar plant input and output, \( n \) is the plant order and \( \tau \) is the plant delay. The function \( f\{\cdot\} \), the order \( n \) and the delay \( \tau \) are assumed to be known. The function \( f\{\cdot\} \)
is also assumed to be first order differentiable. The problem addressed in this paper is the design of a fixed (nonadaptive) feedback controller that causes the plant output to follow a given class of command input, in a satisfactory manner.

We propose to design a fixed output-feedback fuzzy controller for a given nonlinear plant. The fuzzy logic controller has a few free parameters such as the scaling factors, and the center and spread of the membership functions. These parameters are to be suitably chosen in order to come up with a satisfactory controller. In this paper, these parameters are tuned based on the minimization of a given performance index.

3. Control of nonlinear plants

In the design of practical controller for nonlinear systems, local linearization is a widely used technique. Linearization produces a linear approximation of the plant for small change of variables around an operating point. Controller synthesis based on this linear model often brings about satisfactory control. However, any change in the system operating point may alter the parameters of the linear approximation. In order to cope with this difficulty, in some cases the controller is modified with the operating point. In situations where the operating points can be inferred through measurement of suitable variables, this scheme can be effectively utilized. Such a scheme is known as gain scheduling or parameter scheduling [5].

In gain scheduling, the nonlinear plant is represented by a set of finite number of locally linearized plants. In contrast, control design based on a continuous change of locally linearized model with the operating point can also be considered [4]. The locally linearized model, however, is valid for small deviation of signals. As a result, control based on such models may not yield good performance in many cases.

A generalization of the locally linearized model for the plant described by Eq. (1) can be expressed as

\[ \ddot{y}(t) = a(I_t)\ddot{y}(t-1) + \cdots + a_n(I_t)\ddot{y}(t-n) + b_0(I_t)\dot{u}(t-\tau) + \cdots + b_{n-\tau}(I_t)\dot{u}(t-n), \]  \hspace{1cm} (2)

where

\[ I_t = [y(t), \dot{y}(t), \ddot{y}(t), \dddot{y}(t)]^T, \]
\[ O_t = [y(t-1), \ldots, y(t-n), u(t-\tau), \ldots, u(t-n)]^T \]

and the tilde (~) indicates the incremental values. The coefficients \( a_i \)'s and \( b_i \)'s are given as

\[ a_i(I_t) = ([\partial f(O_t)/\partial y(t-i)]\ddot{y}(t))/M(t) \]

and

\[ b_i(I_t) = ([\partial f(O_t)/\partial u(t-\tau-i)]\ddot{u}(t))/M(t) \] \hspace{1cm} (3)

with

\[ M(t) = \sum_{i=1}^{n} \frac{\partial f(O_t)}{\partial y(t-i)}\ddot{y}(t-i) + \sum_{i=\tau}^{n} \frac{\partial f(O_t)}{\partial u(t-i)}\dot{u}(t-i). \]

The above generalization simply scales elements of the local linearized model appropriately to ensure its validity for a particular large signal change. In the scalar case this is equivalent to replace the slope of the tangent by the slope of an appropriate chord. While the linearized model is only valid for small signal changes, the above generalization is valid for arbitrary change in variables, as can be verified by direct substitution. For small signal variations \( \ddot{y}(t-i) \) and \( \dot{u}(t-i) \), respectively, approach \( \ddot{y}(t-i) \) and \( \ddot{u}(t-i) \). As a result, \( M(t) \) approaches \( \dot{y}(t) \). Consequently, representation (2) collapses to the locally linearized model.
Although the pseudo-linear representation given by Eqs. (2) and (3) are exact, it is evident that the coefficients of this equation depend not only on the operating point of the nonlinear model but also on the signal increments in every time step. Depending on the signal increment, a family of coefficients (i.e., $a_i$'s and $b_i$'s) may exist for each operating point. It appears that a satisfactory controller design based on all such linear models at every operating point would perform better than the one based on only the locally linearized models.

Although the family of these linear models at each operating point depends on the normal operating input–output of the plant, their determination and controller design based on them are formidable tasks. One practical solution to this problem is to exploit the super function approximation capability of the fuzzy logic system (FLS) [11,19] and adjust the parameters to make it act as a satisfactory controller at all possible operating points. In the proposed design, the FLS is directly trained within a simulated control loop. The fuzzy logic controller (FLC) receives appropriate input to infer the operating point of the plant while the training setpoint conforms to the normal operating condition of the plant. The trained FLC results in an aggregate of satisfactory controllers at each conceivable operating points for all possible signal variations as determined by the training setpoint.

3.1. Integrators in nonlinear plants

In the classical controller design of linear plants setpoint following is achieved through introduction of integrators. This ensures that the steady-state output follows the constant setpoint and at the same time, it makes the control system robust against low frequency perturbation in the process model. It is natural to explore its extension to nonlinear plants. However in nonlinear control design, this approach does not generally ensure constant setpoint following. In the following, we provide a proposition that sheds guideline on the use of integrator in nonlinear control system. Consider the integrator based nonlinear control system of Fig. 1. Let the controller-plant combination be described by the function $\psi(\cdot)$ such that $\psi(t)$ can be represented as

$$y(t) = \psi(y(t-1), \ldots, y(t-n_1), e(t-1), \ldots, e(t-n_2), z(t-1), \ldots, z(t-n_3)) + d,$$  \hspace{1cm} (4)

where as shown in the figure, $e(t)$ is the setpoint error, $z(t)$ is its integrated value and $x$ is a constant setpoint. For the sake of clarity, the controller-plant delay has been assumed as 1. Assume that $d$ is a constant disturbance.

Proposition 1. If $\psi(\cdot)$ is monotonic in $z(t-1)$ for $y(t) \in A \forall t$; $A$ being a subset of $R$, $R$ being the set of all real numbers then for each $x \in A$, the steady-state value of $y(t)$ defined as $y_{ss(t)} \triangleq \lim_{t \to \infty} y(t)$ either (a) converges to $x$ or (b) oscillates indefinitely or (c) moves out of $A$.

Proof. In simpler words, the proposition asserts that if $y(t)$ converges within $A$, it converges to $x$. This can be proved through contradiction as follows:
Assume that $y(t)$ converges to $x \neq x$, where $x \in \mathbb{A}$. This implies that

$$\lim_{t \to \infty} e(t) = x - x \neq 0,$$

which in turn implies that the steady-state integrator output $\lim_{t \to \infty} \zeta(t)$ keeps on changing instead of converging to a constant value. Since $\psi(\cdot)$ is monotonic in $\zeta(t-1)$, this entails that $y_n(t)$ also changes with time, which is a contradiction to the original assumption that $y_n(t)$ converged to $x$. This completes the proof. \qed

**Remark 1.1.** The proposition establishes sufficient conditions for effective use of integrator in nonlinear servo systems. It further establishes the analogy of Fig. 1 with the linear control system. If the complement of $\mathbb{A}$ is regarded as infinity, the two cases become identical. Obviously, in a nonlinear integrator-based control design one should ensure that $y(t)$ is restricted in $\mathbb{A}$ and further, $e(t)$ and $\zeta(t)$ are restricted in a region such that $\psi(\cdot)$ is monotonic in $\zeta(t-1)$. Since a fuzzy controller cannot be designed through synthesis, it requires that it should be adequately trained to achieve these properties for all possible regimes of the normal operating condition.

**Remark 1.2.** The proposition requires the monotonicity of the controller-plant combination. It is not required that the plant be monotonic individually. Therefore, it may be possible to arm a class of nonmonotonic plants with integrator based design along with proper choice of the controller. This will, not only ensure setpoint following but as well make the control system robust against constant and low-frequency disturbances. In the proposed approach, the FLC is trained to behave as such a controller by ensuring that it follows all the step changes in the training setpoints.

**Remark 1.3.** The above result is quite general. No assumption is made on the stability of the open-loop plant. More specific case has been considered by Desoer and Lin [8], where it has been shown that if an open-loop stable nonlinear plant has a strictly increasing dc input/output map, then a simple PI compensator may achieve identical objective. However, their result does not apply to open-loop unstable plants.

### 3.2. The proposed control scheme

Bringing together all of the above arguments, we propose an output and error feedback FLC for nonlinear plants as depicted in Fig. 2. The FLC receives as input, the setpoint error $e(t)$, $e(t-n_1)$, integrated setpoint error $\zeta(t)$, $\zeta(t-n_2)$, plant input $u(t-1)$, $u(t-n_3)$ and the plant output $y(t)$, $y(t-n_4)$, where the parameters $n_1$, $n_2$ and $n_4$ are to be determined from trial and error. Thus, the FLC is assumed to approximate
u(t) = g\{e(t), \ldots , e(t-n_t), \xi(t), \ldots , \xi(t-n_s), y(t), \ldots , y(t-n_p), u(t-1), \ldots , u(t-n_r)\} \tag{5}

such that the closed-loop system responds to a class of setpoints in a prescribed manner.

Imitating the proportional plus integral (PI) control, the FLC receives the setpoint error directly as well as through an integrator. While the direct application of the setpoint error enables the controller to quickly respond to a change in the setpoint, the application of the setpoint error through the integrator ensures low-frequency command tracking. The injection of the plant input and output to the FLC enables it with the operating point information and as well provides additional dynamics to the controller. In a more realistic situation, other variables representing the plant ambient conditions may be required to infer the operating point.

Since the transfer function and frequency-domain representations are not valid for nonlinear plants and controllers, the controller design in this case may rely on inducing the closed-loop plant output to follow the response of a suitably chosen reference model. The dashed lines in Fig. 2 schematically depicts a training scheme to accomplish this. During the training, the FLC parameters are optimized to minimize the sum squared errors between the output of the control system and that of the reference model.

During the optimization, the training data set must conform to the normal operating condition of the plant and should represent all regimes of expected operation of the plant. For example, if the plant is expected to experience step changes in the setpoint between \(x_{\text{min}}\) and \(x_{\text{max}}\), during the training the setpoint must contain random step changes between \(x_{\text{min}}\) and \(x_{\text{max}}\). Further the sampling-sequence methodology during the training and implementation stages should be consistent.

The concept of controllability is one of the basic ingredients in the satisfactory control of a plant. Controllability is associated with the existence of a control input to drive the plant to the desired equilibrium point. Although controllability is a global concept in linear plants, in general, a nonlinear plant may only be locally controllable. The local controllability of nonlinear plants requires existence of a suitable inverse operator in the specified region. If the input is generated through a specific controller structure, further assumptions of existence of appropriate fixed control parameters is required to achieve the control objective [12]. However in most realistic conditions, these are hard to verify.

It can be shown that if the linearized plant is controllable at an equilibrium point (as can be verified through a locally linearized plant equation), then there exists a neighborhood around the equilibrium and a constant feedback law of the form of Eq. (5) that will drive the nonlinear plant to the equilibrium in a finite amount of time [12]. In this paper, it will be assumed that a stabilizing feedback controller of the specified structure (i.e. with the assumed values for \(n_s\), \(n_t\), and \(n_p\) and the assumed structure of the FLC) exists. The purpose of this paper is to realize the approximate mapping for the controller.

4. Controller training

Once the controller structure is selected, it remains to adjust the parameters of the FLC so that the performance index is minimized. The proposed optimization scheme utilizes the concept of Block Partial Derivatives [1,2], which is described next.

4.1. Block partial derivatives (BPD)

Definition. Consider a block \(\mathbf{A}\) with the input \(x \in \mathbb{R}^n\) and the output \(y \in \mathbb{R}^m\). The block partial derivatives (BPDs) [1] of \(\mathbf{A}\) are defined as

\[
\frac{\partial^k y_j}{\partial x_j} = \lim_{\Delta x_k \to 0} \frac{\Delta y_j}{\Delta x_j} |_{\Delta x_k = \text{0 for } k \neq j}.
\]
Remark 1. In the above definition all inputs other than $x_i$ are held constant. However, the other signals within the block are allowed to vary owing to the change in $x_i$.

Remark 2. If a parameter set $w \in \mathbb{R}^{d_x}$ in the block also is allowed to vary, its effect may be accommodated by representing the weights as auxiliary inputs. This representation facilitates computation of BPDs with respect to the parameters.

Remark 3. When $x$ and $y$ are functions of time and the block is dynamic (i.e. has memory), the BPD's may assume rational dynamic operators.

Remark 4. The BPDs are similar to ordinary partial derivatives except that a block (which may be imaginary) characterizes the explicit dependence of a variable on other variables. The 'other' variables are analogous to the independent variables in the ordinary partial derivatives. Thus the algebra of ordinary partial derivatives can be easily modified for the BPDs.

A computational method for the BPD is proposed in [2] which is based on the argument that the definition of BPD involves infinitesimal change in one of its variables, it causes infinitesimal change among all other variables in the block. Thus in order to compute the BPD of a block, all blocks inside the given block can be replaced by their linearized equivalents. The resulting network can then be reduced using the classical graph reduction technique. This reduced graph depicts the relationship between the incremental variables. For details and examples on this BPD computation technique the reader is referred to the above reference.

4.2. Controller structure

Fuzzy logic utilizes the linguistic variables that do not have precise values. This allows the use of traditional human heuristics and experience in designing systems. In order to combine the linguistic variables with real process input–output, Takagi and Sugeno [18] proposed a fuzzy system whose input is fuzzy but the output part is crisp. This system has been successfully applied in many practical problems.

The defuzzification method used in our controller is identical to that used by Takagi and Sugeno as reported in the above reference. In this method, the crisp output is obtained using the following equation:

$$u(x) = \frac{\sum_{j=1}^{M} u_j \prod_{i=1}^{L} m_i^j (x_i)}{\sum_{j=1}^{M} \prod_{i=1}^{L} m_i^j (x_i)} + b. \quad (7)$$

In the above $x \in \mathbb{R}^L$ and $u(.) \in \mathbb{R}$, respectively, are the input and the output of the FLC. In our context, $x$ includes $c, \zeta, y$ and delayed $u$ values. Thus the value of $L$ in our case equals 3. $M$ is the number of rules, $b$ is a bias term, $u_j$ is the output due to the rule $j$, and $m_i^j$ is the appropriate membership function. Although the proposed method applies to any smooth membership functions, we demonstrate it assuming Gaussian membership functions. In this case, the above takes the following form:

$$u(x) = \frac{\sum_{j=1}^{M} u_j \prod_{i=1}^{L} \exp(-((c_i - \mu_i^j)/\sigma_i^j)^2))}{\sum_{j=1}^{M} \prod_{i=1}^{L} \exp(-((c_i - \mu_i^j)/\sigma_i^j)^2))} + b. \quad (8)$$

where $\mu_i^j$ and $\sigma_i^j$ are the corresponding mean and spread of the membership functions.

4.3. Back propagation in fuzzy systems

Like the artificial neural networks, a class of fuzzy systems can also be trained using the back propagation algorithm. Back propagation is a simple minded gradient algorithm in order to optimize a performance index.
Back propagation is referred to the systematic computation procedure in a multi-layered feed-forward network. This basic idea also can be extended to the fuzzy system represented by Eq. (8).

Assuming \( b, u_i, \mu_i^j, \) and \( \sigma_i^j \) as the adjustable parameters, the gradients needed for the optimization can be obtained as follows:

\[
\frac{\partial F}{\partial b} = 1,
\]

\[
\frac{\partial F}{\partial w^i} = \phi^j,
\]

\[
\frac{\partial F}{\partial \mu_i^j} = (u^j - u) \phi^j \frac{2(x_i - \mu_i^j)}{(\sigma_i^j)^2},
\]

\[
\frac{\partial F}{\partial \sigma_i^j} = (u^j - u) \phi^j \frac{2(x_i - \mu_i^j)^2}{(\sigma_i^j)^3}
\]

and

\[
\frac{\partial F}{\partial x_i} = -\sum_{j=1}^{M} (u^j - u) \phi^j \frac{2(x_i - \mu_i^j)}{(\sigma_i^j)^2},
\]

where

\[
\phi^j = \frac{\prod_{i=1}^{L} \exp((- (x_i - \mu_i^j)/\sigma_i^j)^2)}{\sum_{j=1}^{M} \prod_{i=1}^{L} \exp((- (x_i - \mu_i^j)/\sigma_i^j)^2))}.
\]

In the above, we introduced the BPD notation. The superscript \( F \) in the derivatives indicates that these derivatives correspond to the block \( F \) that includes the fuzzy regulator (i.e., external connections or feedback, if any, are disregarded). Note that some of the centroid and spread parameters may represent the same coefficients that must be updated together. In this case, the computation of the partial derivatives must add up the corresponding terms from the right hand sides of Eqs. (11) and (12). Eqs. (10)–(12) are derived in [19], while Eqs. (9) and (13) can be derived in an analogous way.

In Ref. [19], the above expressions have been obtained using the chain rule. However, the same results can be obtained by employing the BPD technique, where the block \( F \) is decomposed into sub-blocks containing simpler functions [2]. However, we will be using the BPD technique in the next section where the use of chain rule would become unrealistically cumbersome.

**4.4. Parameter adjustment**

In this section, we will derive an algorithm to train the FLC. The algorithm is based on a gradient scheme. Reconsider the control configuration of Fig. 2. Given an arbitrary training setpoint \( x(t), \ t = 1, \ldots, N \), the off-line training minimizes the following criterion function:

\[
J(W) = \frac{1}{2} \sum_{t=1}^{N} \varepsilon^2(W, t)
\]

where

\[
\varepsilon(W, t) = y_m(t) - y(W, t)
\]
and the $n_W \times 1$ vector $W$ contains all the adjustable parameters of the FLC i.e. $b$, $u^j$, $\mu^j_i$ and $\sigma^j_i$. For an FLC with $L$ membership functions, and $M$ rules, the vector $W$ is given by

$$W = [\mu^1, \ldots, \mu^M, |\ldots|, \mu^1, \ldots, \mu^M, |\ldots|, \sigma^1, \ldots, \sigma^M, |\ldots|, \sigma^1, \ldots, \sigma^M, |u^1, \ldots, u^M| b].$$ (16)

The variables $y_m(t)$ and $y(W, t)$ respectively, denote the reference model output and the plant output, as depicted in the figure. The presence of the vector $W$ in the argument list indicates its explicit dependence on $W$. To minimize $J(W)$, one may adjust the parameter vector $W$ along the negative gradient direction. This gives the following parameter adjustment rule:

$$W_{k+1} = W_k - \eta_k \left( \frac{\partial^C J}{\partial W_k} \right)^T = W_k + \eta_k \sum_{t=1}^{N} \Gamma(W_k, t) \sigma(W_k, t),$$ (17)

with

$$\Gamma(W_k, t) = \left( \frac{\partial^C y(t)}{\partial W_k} \right)^T,$$

where $\eta_k$ is the ‘adaptation rate’ and $\Gamma(t)$ is the ‘sensitivity derivative’. Further, the sensitivity derivative $\Gamma(t)$ is required to be computed for the hypothetical block $C$ that contains the entire closed-loop system as shown in Fig. 2.

4.4.1. Computation of the sensitivity derivative

In order to compute the sensitivity derivative, in Fig. 3 we draw the linearized network for the closed loop system of Fig. 2 (for details see [2]). In Fig. 3, the $1 \times n_W$ vector $\frac{\partial^F u}{\partial W}$ denotes the BPDs with respect to the block $F$ (i.e. the fuzzy block). Further, the polynomials $P(q)$, $R(q)$, $S(q)$ and $T(q)$ are given as

$$P(q) = p_0 + p_1 q^{-1} + \cdots + p_m q^{-m},$$

$$T(q) = 1 + t_1 q^{-1} + \cdots + t_m q^{-m},$$

$$P(q) = p_0 + p_1 q^{-1} + \cdots + p_m q^{-m},$$

$$T(q) = t_0 + t_1 q^{-1} + \cdots + t_m q^{-m},$$ (18)

with

$$p_i = -\frac{\partial^F u(t)}{\partial y(t-i)}, \quad r_i = \frac{\partial^F u(t)}{\partial u(t-i)}, \quad s_i = \frac{\partial^F u(t)}{\partial z(t-i)} \quad \text{and} \quad t_i = \frac{\partial^F u(t)}{\partial e(t-i)}$$

and $q$ is the forward shift operator. All of the above partial derivatives can be computed using Eqs. (9)–(14).
The polynomials $A$ and $B$ in the figure, are given as

$$A \equiv A(q) = 1 + a_1 q^{-1} + \ldots + a_m q^{-m},$$

$$B \equiv B(q) = b_0 + b_1 q^{-1} + \ldots + b_n q^{-n}$$

with

$$a_i = \frac{\partial G y(t)}{\partial y(t-i)} \quad \text{and} \quad b_i = \frac{\partial G y(t)}{\partial u(t-i)}.$$  

where the block $G$ contains the open-loop plant as shown in Fig. 2. Since the plant equation is assumed to be known, these BPDs can be readily calculated for known input-output values.

From the linearized network drawn in Fig. 3, Meson’s rule [9] can be applied to get the following expression

$$I(W, t) = \left( \frac{\partial e y(W, t)}{\partial W} \right)^T G_w(q) \left( \frac{\partial e u(t)}{\partial W} \right)^T,$$

where

$$G_w(q) = \frac{(1 - q^{-1})^T B q^{-i}}{(1 - q^{-1})^T [A R + P B q^{-i}] + [S + (1 - q^{-1})^T B q^{-i}]}$$

with $A \equiv A(q), \ B \equiv B(q), \ P \equiv P(q), \ R \equiv R(q), \ S \equiv S(q)$ and $T \equiv T(q)$.

4.5. Stability of the training algorithm

The proposed training algorithm cannot be shown to be globally stable. However, this should not be treated as a serious drawback in the scheme, as the proposed control design is not an adaptive scheme. For the proposed controller design, the training is conducted off-line, where the algorithm can be stabilized by adjusting the adaptation rate. Since a gradient algorithm does not guarantee convergence to a global minimum, the training algorithm can also be conducted with different initial parameters and different adaptation rate parameters until a good minimum is achieved, as evident from the model following capability of the control system during the training.

It should be realized that we are addressing the controller design of a general class of nonlinear plants for which no theoretical stability results are available. It is (not only) expected (but inevitable) that during the off-line training, one will get into stability problems. However, once a controller is trained to stay stable for a class of setpoints, it will remain stable during the implementation as long as the setpoint lingers within the domain of the training values.

In the following we show that the training algorithm has good local stability properties when the FLC parameters are in the vicinity of their optimal values.

4.5.1. Stability in the gradient computation

First, we would like to establish the stability of the gradient computation algorithm given by Eq. (17). From the linearized diagram of Fig. 3, one may derive the locally linearized relationship as

$$\tilde{y}(t) = G_c(q) \tilde{x}(t),$$

where

$$G_c(q) = \frac{[S + (1 - q^{-1})^T B q^{-i}]}{(1 - q^{-1})^T [A R + P B q^{-i}] + [S + (1 - q^{-1})^T B q^{-i}]}$$
with the coefficients of the polynomials $A, B, P, R, S$ and $T$ are given in Eqs. (18) and (19). Near the optimal values of the FLC parameters, the plant output of the closed-loop system will closely follow the reference model output. Further, $G_c(q)$ would approach the reference model $G_m(q)$ such that one may write

$$G_c(q) \approx G_m(q) = \frac{p^{-1}B_m(q)}{A_m(q)}.$$  \hfill (22)

Since the training setpoints represent all possible changes in a specified region, it is evident that the optimal FLC parameters as well furnish stable locally linearized closed-loop systems. This implies that near the optimal parameters, the denominator of $G_c(q)$ would be Schur. Since $G_c(q)$ and $G_H(q)$ (see Eqs. (20) and (21)) share the same denominator, one may assert that the gradient computations by Eq. (20) would be as well stable when the FLC parameters approach their optimum values.

With $\Gamma(t)$ being bounded, the local stability of the weight update algorithm given by Eq. (17), however, depends on the adaptation rate $\eta_t$. In the following, we provide some guide lines on the choice of $\eta_t$.

4.5.2. Choice of adaptation rate

Adaptation rate is determined by trial and error procedure. Too large adaptation rate may turn the training process unstable. It may as well bypass the global minimum. In a similar way too small adaptation rate will result in increased computation and as well may cause convergence to a local minimum. A suitable value of $\eta_t$ can only be figured out through repeated trials.

The convergence in a steepest descent method also depends on the starting point. In our simulation study, the initial working controller has been obtained through trial and error. However, the initial FLC can also be obtained by other methods. There exists various parameter sets of the controller that may produce an initial working controller. Some of which may be quite close to bad local minima resulting in convergence to these minima. Thus training with various initial working controllers may be needed to get a satisfactory trained controller.

It is to be noted that this training method is quite similar to that of multi-layered feed-forward neural network (MFNN) controllers [3]. However, in the case of MFNN, one is concerned with the training of the weights, which may have no physical significance. On the other hand, the parameters of an FLC may have explicit significance in affecting the system response within a given range of operating points of the plant.

Further, in the case of an FLC, one is confronted with different aspects of the layers. In the first layer, parameters to be tuned are the spreads and centroids of the input membership functions while in the next layer, gains correspond to these membership functions are tuned. It is found from simulation study that a small change in the latter is much more significant as compared to that in the former ones. All these observations justify the use of different adaptation rates for different parameters.

5. Computational considerations

No hard and fast rules can be specified for the training of the FLC parameters. Due to the nature of the algorithm, training requires heuristics. Some of the heuristics established from intensive simulation study are discussed below.

The training method primarily is user guided, analogous to the neural controller training presented in Ahmed [3]. Initially a working controller is achieved by setting up the Fuzzy IF-THEN rules. These rules could be initialized and modified by observing the response of the system under different operating conditions. By playing with these parameters it is almost always possible to get a stable closed-loop control system for at least the first leg of the training setpoints. If this is not possible, other alternatives as explained in the following can be attempted. The complete training process can be described through two phases.
5.1. Phase I

This phase consists of obtaining an initial working FLC. One way of achieving this is to start with random parameter values and apply the training algorithm for only the first leg of the training setpoints. This approach may be repeated a few times with different initial random parameters until a satisfactory response is obtained. In case of random parameters, the output parameters $u'$ should be small to start with. From the simulations it has been found that these parameters change faster. Most of the times, the randomly initialized FLC will start with an unstable closed-loop system. Therefore, it is essential that the initial simulation time be small that will prevent system output from attaining very large magnitude. As soon as a satisfactory output is obtained, the simulation time can be gradually increased.

A second approach to obtain an initial working controller, would be to initialize the FLC parameters employing the human logic and experience. This can be followed by applying the training algorithm to only one leg of the training setpoints with a small simulation time as described earlier. Then the simulation time can be gradually increased as the training proceeds. The amplitude of the setpoint can be reduced temporarily in order to get an initial stabilizing regulator.

A third alternate to acquire a working controller would be to train the isolated FLC (The FLC being outside the loop) to mimic the input-output of a satisfactory controller at hand. Such a controller may stem from the standard PID technique or a linear control design techniques applied on the nominal or locally linearized plant. Once the input-output data from such a controller is collected, an initial working FLC can be trained employing the standard gradient algorithm [14] and the gradient expressions given in Section 4.3.

5.2. Phase II

After phase I is completed, one has an initial working controller that is to be enhanced further so that the system minimizes the chosen criterion function. In this user guided training, the session is closely monitored to ensure that the system does not become unstable during the course of training. The response of the system is monitored closely. The learning rates for each of the parameters are adjusted accordingly.

All parameters of the FLC do not effect the system response uniformly. Generally different learning rates are required for different parameters. To obtain satisfactory training, vigilance is necessary. At times, it may be required to freeze one or more parameters. Since the parameters of an FLC have physical significance, the designer upon observation of the response may conveniently take such decisions.

Once a satisfactory regulation is obtained for one leg of the training setpoints, it is time to add another leg. The parameters obtained from the previous training are preserved and training for the new setpoint is initiated from these parameter values. At times, it may be difficult to train the controller if the added setpoint level is significantly different from the previous ones. In this situation, the initial training should be conducted with a smaller deviation from the previous setpoint value.

During the training, the number of setpoint transition may be gradually increased. Unsatisfactory training at this stage may indicate that the FLC size is inadequate. If this is suspected, the membership function of the FLC must be increased. In that case it may be necessary to go back to phase I. However, the initial training of the FLC at this stage would be relatively easy as the output gain correspond to the new membership function can be inferred from the ones correspond to the old ones through interpolation.

A further consideration is to identify the proper parameters that are to be tuned to improve a particular aspect of the regulation performance. In cases, the other parameters that are not related to the specific aspect, should be frozen. For example, the parameters relating to the membership functions that are not activated by the controller input can be held fixed. For this purpose the corresponding adaptation rate may be set to zero.

A flow chart for the controller training is exhibited in Fig. 4.
6. Simulation studies

In order to validate the proposed scheme, extensive simulation studies have been carried out on a few examples. In the following, we present the simulation results on a laboratory-scale liquid-level system. In [15], such a system has been identified as being described by the following NARX model

\[ y(t) = 0.9722y(t - 1) + 0.3578u(t - 1) - 0.1295u(t - 2) - 0.3103y(t - 1)u(t - 1) - 0.04228y^2(t - 2) \]
The system consists of a dc water pump feeding a conical flask which in turn feeds a square tank, giving the system second-order dynamics. The controllable input is the voltage to the pump motor and the plant output is the height of the water in the conical flask. The aim under simulation conditions, is for the water height to follow some demand signal.

The variations of the steady state gain and the dominant time constant of the linearized plant against the output level are shown in Fig. 5. The time constant is shown in terms of the sampling intervals. The large variation in both of these attributes indicate a high degree of nonlinearity.

It was intended to design an FLC for this plant for step commands ranging between \( \pm 0.5 \). Therefore, a single integrator has been used in the control system. The input to the FLC has been \( e(t) \), \( \zeta(t) \) and \( y(t) \). This corresponds to \( n_e \), \( n_\zeta \), \( n_y \) and \( n_y \), respectively, being 1, 1, 1 and 0 (see Eq. (5)). Each input to the FLC has been assigned five membership functions, requiring the tuning of 15 mean and spread parameters. The number of output gain parameters have been 125. There has been a total of 155 parameters to adjust. All the programs have been written in MATLAB.

The initial FLC has been obtained through trial and error. A considerable amount of trial and error produced a working regulator, the performance of which is shown in Fig. 6. They show the control for a positive and a negative setpoint. It can be seen that although the initial (untrained) FLC showed some setpoint tracking ability, the control is clearly unsatisfactory. While the tracking of the negative setpoint is clearly unacceptable, the transition to the positive setpoint is rather slow. Despite of repeated trials, we failed to improve the FLC performance for the negative setpoint.

In order to train the FLC employing the proposed method, the performance index has been taken as in Eq. (15). A first-order reference model represented by the following equation was chosen:

\[
(1 - 0.8q^{-1})y_m(t) = 0.2q^{-1}x(t).
\]

The algorithm given by Eq. (17) has been applied for the parameter adjustments. As mentioned earlier that there has been a total of 155 parameters to adjust. The gradients have been computed using Eqs. (9)–(12) and (20). The plant polynomials \( A(q) \) and \( B(q) \) were obtained through local linearization of the plant. The
training has been user guided as explained in Section 5. The adaptation rate has been changed employing
users experience and through other heuristics until a stable satisfactory FLC for the entire setpoint has been
obtained. The final training setpoint has been constructed as random step changes in twenty different levels
within the range ±0.5.

After successful training of the controller, the control system has been tested. Fig. 7 depicts the control
of the plant for the same setpoints as in Fig. 6. A comparison of this with Fig. 6 clearly demonstrates the
improvement. The response to the positive setpoint became faster while the large oscillation for the negative
setpoint disappeared. Figs. 8 and 9, respectively, show the plant response to random changes in setpoints
due to the untrained and trained FLC. For the untrained FLC, the control system turned unstable during the
second leg of the setpoint change. The plant input under the closed-loop control with the trained FLC is also
displayed in Fig. 9. It was observed that the swinging in the input can be reduced by allowing a sluggish
plant response by picking a slower reference model.

6.1. Robustness of the FLC

Simulation studies have been also conducted to investigate the sensitivity of the control system against the
plant parameter variations. Simulations have been conducted for different plant parameter values, while the
FLC has been designed for their nominal values. In this study only one parameter has been varied at a time
keeping all others at their nominal values. The control of the plant for variations in typical parameters are
shown in Fig. 10. In this figure the parameter numbering refers their chronological ordering in Eq. (23). It can
be observed that with changes in the parameter values, the response of the control system deteriorates, however,
the closed-loop system remains stable for significant parameter changes. This property can be attributed to the
robust structure of the control system comprising the feedback and integrator, and the interpolation capability
of the FLC.

7. Conclusion

In this paper we considered the fuzzy logic controller (FLC) design for nonlinear plants. The proposed
control configuration stems from the well-known and proven control strategies in the linear systems. The
control scheme uses integrator to ensure low-frequency command tracking, low-frequency disturbance rejection and to enhance robustness of the closed loop system.

The proposed control scheme is an error feedback control that imitates the classical error feedback systems in linear control systems. However, the controller also incorporates the variables representing the plant operating point. These may include the plant input–output and the variables representing other ambient conditions.

The FLC has been trained off-line by minimizing the error between the plant output and the output of a reference model, for a class of setpoints. The feasibility and effectiveness of the proposed scheme are illustrated by conducting simulation. The proposed scheme is quite general and it can be easily extended to control systems of complex structures. The design however is computationally intensive as it requires a user guided training of the fuzzy logic system employing a gradient descent optimization.

Acknowledgements

The authors acknowledge King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia, for supporting this research.
References