Genetic optimization of fuzzy networks

Uwe D. Hanebeck*, Günther K. Schmidt

Department for Automatic Control Engineering (LSR), Technical University of Munich, 80290 München, Germany

Abstract

A novel fuzzy network controller is introduced which is interesting from both a theoretical and a practical viewpoint. It is similar to a radial basis function neural network, contains structured information and may be characterized by a few parameters only. For training of these networks with experiments or by examples, a nonstandard genetic algorithm is applied, using a real-valued parameter encoding scheme and an appropriate cross-over. The adaptation of a direct fuzzy controller for a simple system illustrates the procedure. In a second example the integrated design and optimization approach is shown for a typical industrial controller stabilizing a laboratory size magnetic levitation system. It includes nonlinear components for fuzzy anti-windup.

Keywords: Control theory; Fuzzy networks; Genetic algorithms; Optimization

1. Introduction

For many real-world control problems, it is possible to find a working fuzzy controller by formulating heuristic knowledge and by using a "trial and error" approach for fine-tuning. This may not, however, always yield the anticipated results and is undoubtedly a tedious task because of the huge number of tuning parameters involved. To overcome this problem, a number of advanced approaches have been reported in the literature. In [6], fuzzy rules are generated from training data by what is called product-space clustering, in [7] a fuzzy plus PD controller with look-up table adaptation is introduced, and in [9] the rules are changed to increase performance. This paper presents an approach for adapting the membership functions of a given initial fuzzy controller by supervised or unsupervised training. After introducing the concept of fuzzy networks (FNs) in Section 2, Section 3 focuses on approximating a desired mapping by a fuzzy network and on the on-line adaptation of a fuzzy network to a given system by using a performance evaluator as the only information source. In both cases, structured information is stored in the network together with a linguistic explanation component. For fast and robust guidance of the optimization procedure with respect to problem-related performance criteria, an appropriate form of a genetic algorithm is introduced in Section 4. It employs a real-valued parameter encoding scheme, suitable cross-over, and provides valid fuzzy controller parameters without the need for penalty functions. Scaling evolves as a by-product. In Section 5 a simple experiment demonstrates how to proceed, and a second example qualifies the proposed method for real-world applications.

* Corresponding author.

2. Fuzzy networks

2.1. Theoretical background

To formulate heuristic knowledge about a mapping $x \rightarrow y, x \in \mathbb{R}^N, y \in \mathbb{R}^M$, we may use rules like

IF input vector $\boldsymbol{x}$ is in region A,
THEN output vector $\boldsymbol{y}$ should be in region B.

Regions A and B are defined by multidimensional membership functions (MFs)

$$\mu_A(\boldsymbol{x}) = \mathrm{e}^{-(\boldsymbol{x}-\bar{\boldsymbol{x}})^{\mathrm{T}} \cdot \boldsymbol{W}_x \cdot (\boldsymbol{x}-\bar{\boldsymbol{x}})}, \tag{1}$$

$$\mu_B(\boldsymbol{y}) = \mathrm{e}^{-(\boldsymbol{y}-\bar{\boldsymbol{y}})^{\mathrm{T}} \cdot \boldsymbol{W}_y \cdot (\boldsymbol{y}-\bar{\boldsymbol{y}})}, \tag{2}$$

with $\boldsymbol{W}_x, \boldsymbol{W}_y$ symmetric, positive-definite matrices and $\bar{\boldsymbol{x}}$, $\bar{\boldsymbol{y}}$ the centroid vectors.

The grade of membership γ of a given $\boldsymbol{x}$ in region A is used to multiply the MF for region B. After performing this for all (implicitly OR-connected) rules, the resulting output MFs are summed up and a crisp output value is obtained by calculating the center of gravity. For the uncorrelated case, i.e. $\boldsymbol{W}_x, \boldsymbol{W}_y$ are diagonal matrices, Eqs. (1) and (2) may be separated into one-dimensional MFs

$$\mu_A(\boldsymbol{x}) = \mu_A^0(x_0) \cdot \mu_A^1(x_1) \cdots \mu_A^{N-1}(x_{N-1}), \tag{3}$$

$$\mu_B(\boldsymbol{y}) = \mu_B^0(y_0) \cdot \mu_B^1(y_1) \cdots \mu_B^{M-1}(y_{M-1}). \tag{4}$$

One rule for N inputs and M outputs may now be written as

IF x_0 in A^0 AND x_1 in A^1 AND ... x_{N-1} in A^{N-1},

THEN y_0 in B^0 AND y_1 in B^1 AND ... y_{M-1} in B^{M-1}.

This leads to multiplication as the natural implementation for the AND-operator. Now, let us consider a fuzzy network with N inputs, M outputs, Gaussian MFs $\mu_{i_j}^j$, $i_j = 0, 1, \ldots, N_j - 1$, for linguistic input variable x_j, and Gaussian MFs μ_k^i described by m_k^i, v_k^i with index k for each output variable y_i. The user-defined connections between input and output MFs are stored in multidimensional connection arrays I_i, $i = 0, 1, \ldots, M-1$. When indexed with a combination of input labels, I_i returns the index of the resulting MF of output i. The AND-operator as well as the implication are implemented via multiplication. Addition is used to perform the composition of the single rules. Defuzzyfication is done by calculating the center of gravity. Analytical expressions for the output vector $\boldsymbol{y} = (y_0, y_1, \ldots, y_{M-1})^{\mathrm{T}}$ given the input vector $\boldsymbol{x} = (x_0, x_1, \ldots, x_{N-1})^{\mathrm{T}}$ are obtained with the following steps. For simplicity, a fully connected fuzzy system is first assumed, i.e. every combination of input MFs is connected to output MFs.

Fuzzyfication:

$$\gamma(i_0, i_1, \ldots, i_{N-1}) = \mu_{i_{N-1}}^{N-1}(x_{N-1}) \cdots \mu_{i_1}^1(x_1)\, \mu_{i_0}^0(x_0) \quad i_j = 0, 1, \ldots, N_j - 1, \; j = 0, 1, \ldots, N-1. \tag{5}$$

Implication:

$$\mu_{\mathrm{Impl.}}(i_0, i_1, \ldots, i_{N-1}, y_0, y_1, \ldots, y_{M-1}) = \gamma(i_0, i_1, \ldots, i_{N-1}) \mu_{I_{M-1}}^{M-1}(y_{M-1}) \cdots \mu_{I_1}^1(y_1) \mu_{I_0}^0(y_0) \quad i_j = 0, 1, \ldots, N_j - 1, \; j = 0, 1, \ldots, N-1. \tag{6}$$

x_1 \ x_0	N	Z	P
N	NG	N	Z
Z	N	Z	P
P	Z	P	PG

Fig. 1. Example rule base in control matrix form.

Composition:

$$\mu_{\text{Comp.}}(y_0, y_1, \ldots, y_{M-1}) = \sum_{i_{N-1}=0}^{N_{N-1}-1} \cdots \sum_{i_1=0}^{N_1-1} \sum_{i_0=0}^{N_0-1} \mu_{\text{Impl.}}(i_0, i_1, \ldots, i_{N-1}, y_0, y_1, \ldots, y_{M-1}). \tag{7}$$

Defuzzyfication:

$$y_i = \frac{\text{Num}_i}{\text{Den}}, \quad i = 0, 1, \ldots, M-1, \tag{8}$$

$$\text{Num}_i = \int_{y_{M-1}=-\infty}^{\infty} \cdots \int_{y_1=-\infty}^{\infty} \int_{y_0=-\infty}^{\infty} y_i \mu_{\text{Comp.}}(y_0, y_1, \ldots, y_{M-1})\, \mathrm{d}y_0\ \mathrm{d}y_1 \cdots \mathrm{d}y_{M-1}, \tag{9}$$

$$\text{Den} = \int_{y_{M-1}=-\infty}^{\infty} \cdots \int_{y_1=-\infty}^{\infty} \int_{y_0=-\infty}^{\infty} \mu_{\text{Comp.}}(y_0, y_1, \ldots, y_{M-1})\, \mathrm{d}y_0\ \mathrm{d}y_1 \cdots \mathrm{d}y_{M-1}. \tag{10}$$

After some calculations, the result is

$$\text{Num}_i = \sum_{i_{N-1}=0}^{N_{N-1}-1} \cdots \sum_{i_1=0}^{N_1-1} \sum_{i_0=0}^{N_0-1} \left(\mu_{i_{N-1}}^{N-1}(x_{N-1}) \cdots \mu_{i_1}^{1}(x_1)\, \mu_{i_0}^{0}(x_0)\, m_{I_i}^{i} \prod_{j=0}^{M-1} v_{I_j}^{j} \right), \tag{11}$$

$$\text{Den} = \sum_{i_{N-1}=0}^{N_{N-1}-1} \cdots \sum_{i_1=0}^{N_1-1} \sum_{i_0=0}^{N_0-1} \left(\mu_{i_{N-1}}^{N-1}(x_{N-1}) \cdots \mu_{i_1}^{1}(x_1)\, \mu_{i_0}^{0}(x_0) \prod_{j=0}^{M-1} v_{I_j}^{j} \right). \tag{12}$$

If the user does not specify an appropriate action for all situations, some combinations of input MFs are not connected to output MFs. In this case the respective terms in Eqs. (11), (12) are skipped. Though the proposed algorithm is computationally light, it may be cast into hardware if required. The structure is similar to a radial basis function neural network. It contains structured information and, in general, only needs a small number of parameters. The MFs are normalized to the range $[-1, 1]$; actual ranges are obtained by scaling. To save computing time, a MF $\mu(x) = \mathrm{e}^{-((x-m)/v)^2}$ should not be evaluated for $\mu(x) < \varepsilon \ll 1$, which is equivalent to $|x - m| > v \cdot \sqrt{-\ln(\varepsilon)}$.

2.2. An example of a fuzzy network

Fig. 1 shows the rule base in control matrix form. The number of inputs is $N = 2$, the number of outputs is $M = 1$, the number of MFs for each input $N_0 = N_1 = 3$. The linguistic labels N, Z, P are mapped to real numbers with N $\rightarrow$ 0, Z $\rightarrow$ 1, P $\rightarrow$ 2. There are 5 output MFs NG, N, Z, P, PG and they are referred to with NG $\rightarrow$ 0, N $\rightarrow$ 1, Z $\rightarrow$ 2, P $\rightarrow$ 3, PG $\rightarrow$ 4. From Eqs. (11), (12), analytical expressions for the output

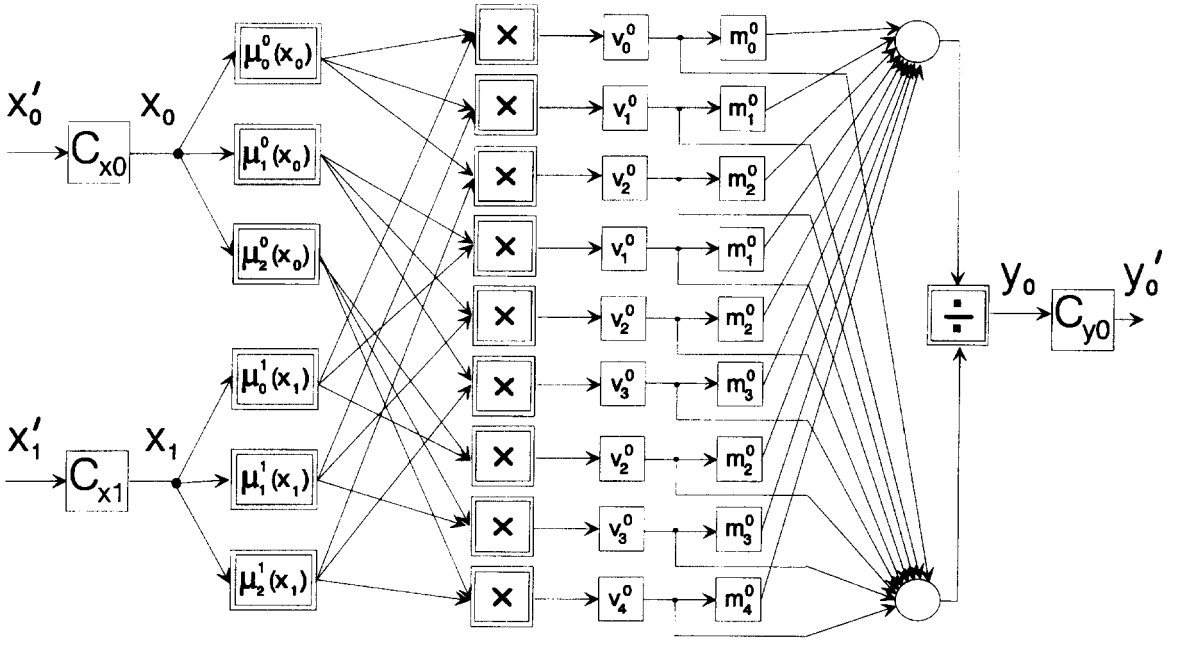

Fig. 2. Example of the fuzzy network processing scheme including scaling.

y_0 given input values x_0, x_1 are as follows:

$$\mathrm{Num}_0 = \sum_{i_1=0}^{2} \sum_{i_0=0}^{2} \left(\mu_{i_1}^1(x_1)\mu_{i_0}^0(x_0) m_{I_0}^0 v_{I_0}^0\right), \tag{13}$$

$$\mathrm{Den} = \sum_{i_1=0}^{2} \sum_{i_0=0}^{2} \left(\mu_{i_1}^1(x_1)\mu_{i_0}^0(x_0) v_{I_0}^0\right), \tag{14}$$

$$I_0(i_0, i_1) = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{pmatrix}, \tag{15}$$

$$y_0 = \frac{\mathrm{Num}_0}{\mathrm{Den}}. \tag{16}$$

As a straightforward implementation, we obtain the network depicted in Fig. 2. Of course, it may be simplified to reduce the computational burden.

3. Optimization methodologies

Here we focus our attention on problems that are difficult to solve with the standard design approaches used in control. To systematically adapt systems to their assigned task, two approaches are presented: learning by experience, which is performed through systematic trials, and learning by example, which is achieved by observing a human operator already mastering the task.

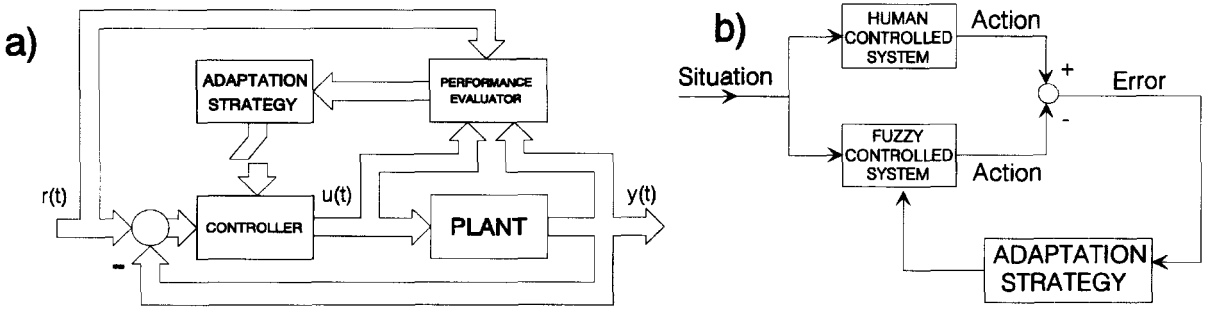

Fig. 3. Optimization schemes: (a) "Learning by experience" and (b) "Learning by example".

3.1. Learning by experience (unsupervised)

Fig. 3 illustrates the basic procedure. A certain trajectory $r(t)$ including all interesting set-point changes is presented in a repetitive fashion. The controller's behaviour is evaluated with prespecified performance criteria which may be of integral or nonintegral type. No a priori information about the system is needed. With every iteration a genetic algorithm supplies a new parameter set for the controller.

3.1.1. Direct optimization

The system is used directly for evaluating controller performance. There are no modelling errors and, of course, plant uncertainties and noise are taken into account. This scheme is ideally suited for repetitive control tasks such as pick & place operations in robotics. It is easily applied since no modelling phase is required.

3.1.2. Indirect optimization

If a system model is available, it can be used for evaluating controller performance concurrently with system operation. The system follows arbitrary trajectories; only the model is evaluated repetitively. It may be continuously adapted to track system parameter variations. Controller parameters for the actual system are changed only when a better performing parameter set is available from the optimization procedure. Speeding up the search process is possible by exploiting the parallel processing capabilities of a genetic algorithm and by running several instantiations of the system model at the same time. This is in contrast to the direct optimization approach, where the search process may only be accelerated by the parallel evaluation of several physical models.

3.2. Learning by example (supervised)

By observing a human operator controlling the system under consideration, we obtain input/output pairs. These define a mapping to be approximated by an initial fuzzy controller specified by the user, Fig. 3. The structure of the proposed fuzzy networks permits the use of gradient techniques – in particular error backpropagation – for controller fine-tuning in conjunction with exploring promising parameter regions by means of a genetic algorithm.

4. Genetic algorithms for adaptation of fuzzy networks

Genetic algorithms (GA) are optimization procedures inspired by natural evolution. They combine robustness with the ability to explore vast search spaces quickly. Detailed introductions may be found in [1–3]. Here parameters are represented as strings of real numbers. Selection is performed with a probability proportional to

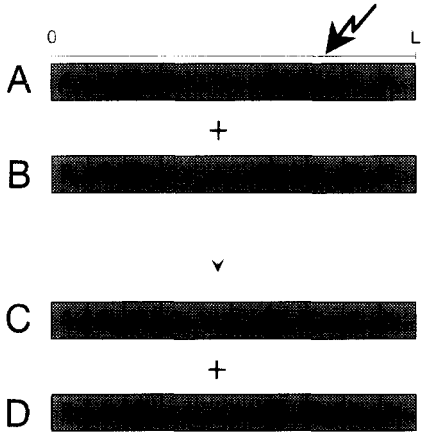

Fig. 4. One–point weighted average cross-over.

the individuals' fitness. After selecting two individuals, recombination and mutation take place with probability P_C and P_M, respectively, and the two members with the lowest fitness of the old population are replaced. Recombination is done via weighted average crossover. For the two strings A and B, both of length L, the realization y of a random variable Y equally distributed in $[0, L]$ is used to define one cross-over point. Define k as the largest integer contained in y. The string positions 0 to $k-1$ stay in place and the elements $k+1$ to $L-1$ are exchanged. The remaining elements of the new strings C and D are calculated as a weighted average with $W = y - k$

$$C(k) = W \cdot A(k) + (1 - W) \cdot B(k), \tag{17}$$

$$D(k) = W \cdot B(k) + (1 - W) \cdot A(k), \tag{18}$$

as shown in Fig. 4. Mutation is performed by adding a random number to the chosen string position such that the allowed range is not exceeded. In analogy to simulated annealing, global mutations are preferred at the beginning of the optimization procedure and small changes at the end. For this purpose we map a random variable X equally distributed in $[-1, 1]$ via

$$h(x) = \begin{cases} (s_{\max} - s) \cdot \left(1 - x^{\left(1 - \frac{n}{N_{\max}}\right)^2}\right) & x \geqslant 0, \\ (s - s_{\min}) \cdot \left(1 - (-x)^{\left(1 - \frac{n}{N_{\max}}\right)^2}\right) & x < 0 \end{cases} \tag{19}$$

to a random variable $Y = h(X)$ with the desired distribution [8]. $s_{\min}$ and $s_{\max}$ define the allowed range, $N_{\max}$ is the maximum number of iterations. In order to avoid implicit change of the rule base, the MFs for one linguistic variable should stay ordered during the optimization. For gaussian MFs this is equivalent to $m_0 < m_1 < \cdots < m_{L-1}$. An elegant approach which avoids penalty functions is to map the variables u_i provided by the GA

$$0 < u_i < 1, \quad i = 0, 1, \ldots, L - 1 \tag{20}$$

to the ordered variables m_i with upper and lower bounds U_i, L_i

$$m_0 < m_1 < \cdots < m_{L-1}, \quad L_i < m_i < U_i, \tag{21}$$

where

$$m_0 = (U_0 - L_0) \cdot u_0 + A_0\,, \quad m_i = (U_i - L_i) \cdot \left(1 + (u_0 - 1) \cdot \prod_{j=1}^{i} u_j\right) + L_i\,,\ i = 1,2,\ldots,L-1\,. \tag{22}$$

To optimize correctly all degrees of freedom, we need an appropriate input and output scaling. Input scaling maps the input ranges to ranges that include $[-1,1]$. Values that fall outside of this interval are mapped to the border values. Output scaling helps to exploit the permitted output range. This is achieved by preferring parameter sets that produce valid scaling. Here is a vantage point for hybridization: In the beginning of the optimization a GA is used alone for quick exploration of optimal parameter regions; a hill-climbing routine would get stuck. In the following iterations the procedure of Hooke-Jeeves [5] is used to optimize the scaling for each parameter set provided by the GA. The third phase is used for locally fine-tuning all parameters provided by the GA by means of the Hooke-Jeeves procedure.

5. Experimental results

Let us constrain our discussion to experiments illustrating some sort of self-organization. For optimization, we use the direct approach introduced in Section 3.1.1. A GA with cross-over probability $P_C = 0.9$, mutation probability $P_M = 1/(\text{number of parameters})$, and a 10 member population is assumed. The parallel processing capabilities of a GA have not been used in this context.

5.1. Synthetic system

The unstable, discrete-time system

$$x_{k+1} + 1.1 \cdot x_k = \operatorname{arsinh}(u_k) \tag{23}$$

may be feedback linearized to exactly track the desired output w_k with

$$u_k = \sinh(w_k + 1.1 \cdot x_k) = f(w_k, x_k). \tag{24}$$

The goal is to achieve the system response $x_{k+1} = w_k$. For this purpose, $f(w_k, x_k)$ in Eq. (24) is approximated with a fuzzy controller only by performing experiments and evaluating the following performance measure:

$$G = \sum_{i=0}^{N_{\text{sim}}-1} |w_i - x_{i+1}|. \tag{25}$$

There exists no further information source. With $x_0' = x_k$, $x_1' = w_k$, $y_0' = u_k$ the fuzzy network from Section 2.2 is used as a direct controller. Fig. 6 depicts the MFs of the user-specified fuzzy controller and the controller's performance for a certain trajectory. The starting population consists of 3 prespecified parameter sets and 7 random sets. The optimization takes place in three phases: 95 iterations with the GA alone; 200 iterations with the Hooke-Jeeves procedure for local optimization of the scaling factors; and 30 iterations with Hooke-Jeeves for the local optimization of all parameters provided by the GA. This results in 660 parameter sets to be evaluated. Fig. 6 shows the MFs of the optimized controller and its performance for the desired trajectory. The scaling factors are $C_{x0} = 1.0046$, $C_{x1} = 1.0012$, and $C_{y0} = 4.00597$. Fig. 7 shows the resulting control surface and the evolution of performance measure G of the best parameter set so far.

5.2. Magnetic levitation system

A schematic overview of the experimental set-up is depicted in Fig. 5. An iron mass (160 g) is to be held free-flying at a certain distance r by the magnetic field of a coil. An optical sensor measures the actual distance

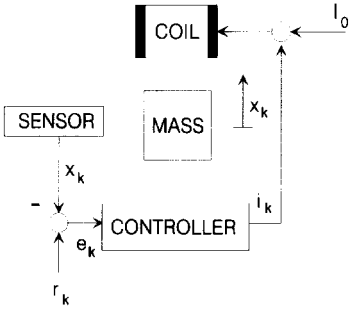

Fig. 5. Magnetic levitation system.

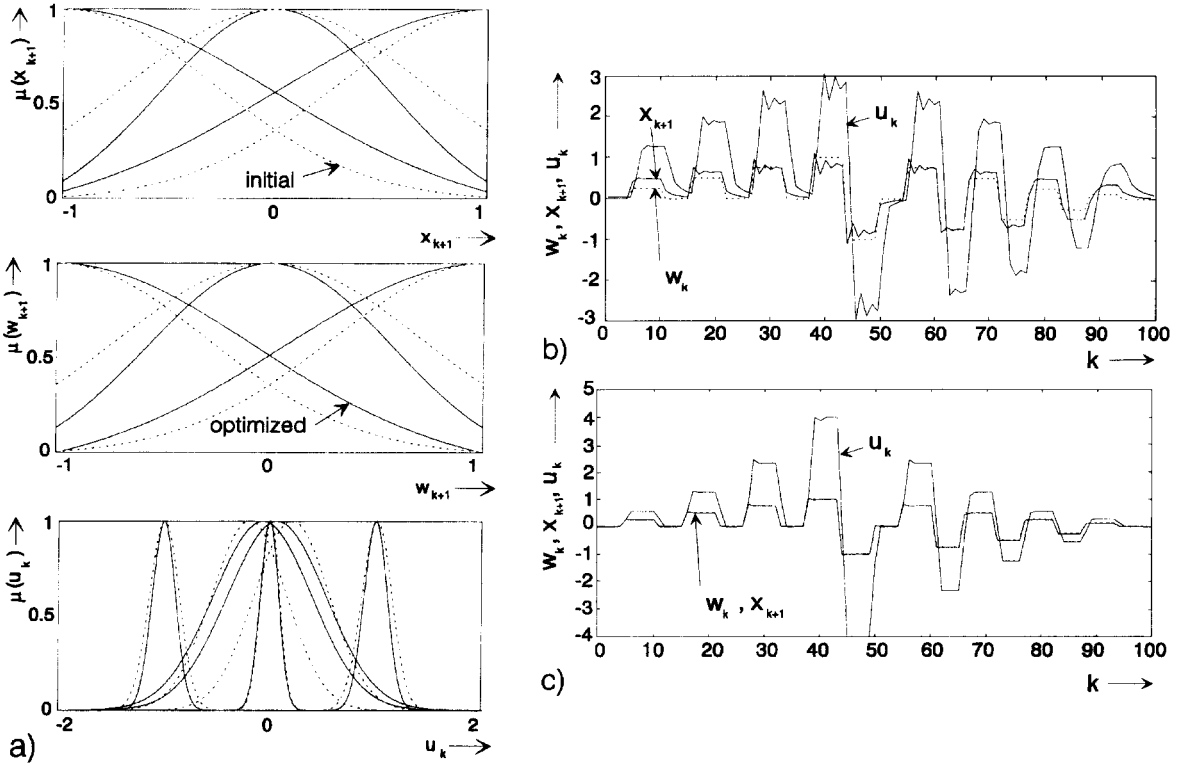

Fig. 6. (a) Initial and optimized MFs. Experiment with (b) initial fuzzy controller (c) optimized fuzzy controller.

x from the normal operating point. The system is unstable and nonlinear. A transputer network controls the system with a sampling interval of 1 ms. The converters have a resolution of 12 bit. The controller is chosen to be of PID type with an adapted integral action:

$$u_k = K_p \cdot \left\{ e_k + \frac{T_v}{T_A} \cdot (e_k - e_{k-1}) \right\} + \sum_{n=0}^{k} K_i(|e_n|) \cdot e_n \,. \tag{26}$$

The conditional integrator is based on ideas presented in [4]. The adaptation rule base is

IF the absolute error $|e|$ is small, THEN K_i large,
IF the absolute error $|e|$ is large, THEN K_i small,

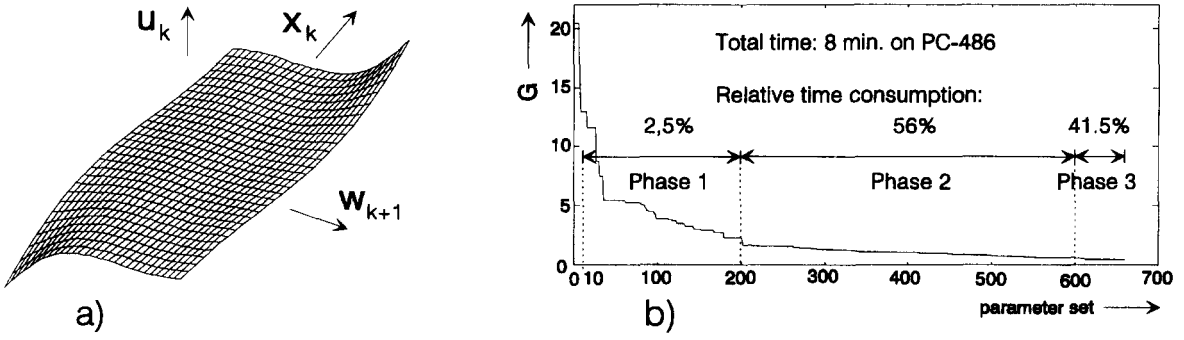

Fig. 7 (a) Control surface of the optimized fuzzy controller. (b) Evolution of performance index G.

with the following MFs:

$$|e|' = \frac{|e_k|}{C_e}, \tag{27}$$

$$\mu^{|e|'}_{\text{small}}(|e|') = \exp\left(-\left[\frac{|e|'}{v_0^e}\right]^2\right), \quad \mu^{|e|'}_{\text{large}}(|e|') = \exp\left(-\left[\frac{(|e|'-1)}{v_1^e}\right]^2\right), \tag{28}$$

$$\mu^{K_i'}_{\text{small}}(K_i') = \exp\left(-\left[\frac{K_i'}{v_0^K}\right]^2\right), \quad \mu^{K_i'}_{\text{large}}(K_i') = \exp\left(-\left[\frac{(K_i'-1)}{v_1^K}\right]^2\right). \tag{29}$$

With the methodology of Section 2.1 an analytic expression

$$K_i' = \frac{1}{1 + v_0^K/v_1^K \exp\left(\left[\frac{|e|'}{v_1^e}\right]^2 - \left[\frac{(|e|'-1)}{v_0^e}\right]^2\right)}, \quad K_i = C_i \cdot (K_i'(|e|') - K_i'(1)) \tag{30}$$

for the adaptation of $K_i(|e_k|)$ is obtained. The optimization is started with a random population. v_0^e, v_1^e, v_0^K/v_1^K, the scaling factors C_e, C_i, and the controller parameters K_p, T_v are optimized with respect to the following performance index:

$$G = \sum_{n=0}^{N_{\text{exp}}-1} \frac{|e_n|}{\text{mm}} + \sum_{n=0}^{N_{\text{exp}}-1} \frac{|u_n|}{1000\text{mA}}. \tag{31}$$

The allowed ranges are defined to

$$\begin{aligned}
500\,\text{mA/mm} &< K_p &< 1500\,\text{mA/mm},\\
0.01\text{s} &< T_v &< 0.1\text{s},\\
0.5 &< v_0^e &< 1.2,\\
0.5 &< v_1^e &< 1.2,\\
0.5 &< v_0^K/v_1^K &< 1.5,\\
0\,\text{mA/mm} &< C_i &< 120\,\text{mA/mm},\\
0.3\,\text{mm} &< C_e &< 1.2\,\text{mm}.
\end{aligned}$$

If a parameter set leads to instability, it is instantaneously exchanged with the current best one. Due to plant uncertainties and noise, same parameter sets lead to rather different values of G. Thus, hybridization is of no

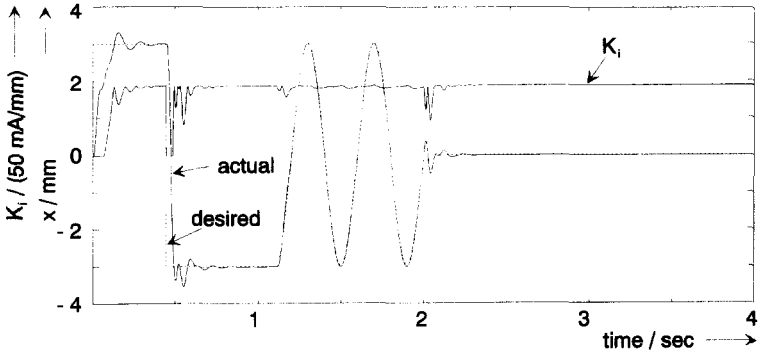

Fig. 8. Experiment with the optimized fuzzy anti windup PID controller.

advantage in this case and the GA is used alone. After 150 iterations, equivalent to 310 experiments, the best parameter set is $K_p = 983.108\,\text{mA/mm}$, $T_v = 0.0447895\,\text{s}$, $v_0^e = 0.638417$, $v_1^e = 0.57288$, $v_0^K/v_1^K = 1.16908$, $C_i = 106.064\,\text{mA/mm}$, $C_e = 1.05207\,\text{mm}$. Experimental results with the parameter set are shown in Fig. 8.

6. Conclusion

A flexible type of fuzzy network is introduced. Optimization of user-specified initial fuzzy controllers is achieved via learning by experiments, i.e. systematic controller evaluations during system operation, or via learning by example, i.e. approximating a mapping given by desired input/output pairs. Genetic algorithms are employed in both cases. Compared to conventional search strategies they locate regions of satisfactory performance very quickly and are insusceptible to noisy and even discontinuous performance measures. The approach allows the user to choose more realistic performance criteria, i.e. others than the usual quadratic ones. For performance enhancement, genetic algorithms are hybridized with proven classical optimization schemes. Two nontrivial examples demonstrate the applicability of the proposed scheme in real-world applications.

References

[1] L. Davis, *Genetic Algorithms and Simulated Annealing* (Pitman Publishing, London, 1987).
[2] L. Davis, *Handbook of Genetic Algorithms* (Van Nostrand Reinhold, New York, NY, 1991).
[3] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison-Wesley, Reading, MA, 1989).
[4] A. Hansson et al., *Fuzzy Anti-Reset Windup for PID Controllers, 12th World Congress International Federation of Automatic Control*, Vol. 10, Sydney (July 18–23, 1993) 389–392.
[5] S.L.S. Jacoby and J.S. Kowalik and J.T. Pizzo, *Iterative Methods for Nonlinear Optimization Problems* (Prentice-Hall, Englewood Cliffs, NJ, 1972).
[6] B. Kosko, *Neural Networks and Fuzzy Systems* (Prentice-Hall, Englewood Cliffs, NJ, 1992).
[7] Y.N. Lee et al., Design of a self-organizing fuzzy plus PD controller using the look-up tables, *Proc. 1992 IEEE/RSJ Internat. Conf. on Intelligent Robots and Systems*, Raleigh, NC (July 7–10, 1992) 775–781.
[8] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs* (Springer, Berlin, 1992).
[9] T.J. Procyk and E.H. Mamdani, A linguistic self-organizing process controller, *Automatica* **15** (1979) 15–30.