Rule base reduction on a self-learning fuzzy controller

Ferreiro García, Pardo Martinez, Vidal Paz
Dep. Electronics & Systems University of A Coruna
E.S.Marina Civil, Paseo de Ronda,51-15011. A Coruna
Email:Ferreiro & udc.es, Tel. 34 81 167000 ext 4205

Introduction

In fuzzy control developing techniques, it is of great importance how to obtain the control rules. A traditional method widely used to obtain control rules is extracting the knowledge from experienced operators. Knowledge is supplied by means of an expert operator and commonly, the empirical knowledge is incomplete. For the case of different experts, results that even contradict each other.

Fuzzy modelling is another method based on structure identification and parameter estimation but is in contradiction with the advantage of no requiring a mathematical model for a fuzzy controller.

Self-organising fuzzy control proposed first by Mamdani, has been applied successfully in many application on fuzzy process control. Actually self-organising fuzzy control is being enhanced with the use of neural networks giving a class of neuro-fuzzy controller. In all of them, there are a common characteristic: Rule evaluation and network training are in most applications excessively time-consuming.

In contrast with above methods, it is proposed in this work an intuitive approach to automatically generate fuzzy control rules, which are to be on-line updated. The purpose is to develop a self-organising rule based fuzzy controller by means of solving two main problems:

- Generating the control rules on the basis of achieve a virtual controller model to act as trainer by auto tuning that supply inherent process knowledge to a rule base
- Develop an inference engine based on a simple strategy to reduce drastically the rule base.

A virtual controller is defined as the one that can be used as trainer or teacher on generating a consistent rule base.

Any strategy applied on rule base reduction, requires that the control function developed by a virtual controller or its output be synthesised by additive sub-functions of the form

$$f1(e)=u_1= f(e)$$
$$f_2(e)=u_2=f(\Sigma e)$$
$$f_3(e)=u_3=f(\Delta e)$$

then,

$$u = \sum_{i=1}^{n} f_i(e) \qquad (1)$$

where u is the controller output, $f_i(e)$ is any component sub-function of the controller actions such as derivative or integral actions. Consequently, the selected virtual controller (fuzzy controller trainer) must satisfy the requirement of being composed by partial additive sub-functions. Controllers of such characteristics can be achieved by some on-line auto-tuning techniques among the most popular self-tuning methods.

Achieving Virtual Controllers

Training a fuzzy rule base requires a pattern algorithm to be learned by a fuzzy rule base. Such a pattern algorithm is used as trainer on the rule base. The trainer is a virtual control algorithm, which satisfy dynamically some performance criteria according operating requirements. Several virtual control algorithms to be used as patterns can be achieved. The most important due to its reliability are autotuning methods, and among them, two frequency domain methods are selected to automatically achieve virtual controllers in this work.

- Method based on relay feedback named method of harmonic balance [1,2,12]
- Method based on determining the phase and magnitude at operating frequency under phase and gain margins specification. [3]

The idea of using two autotuning procedures, which can be applied alternatively or in parallel, is due to the possibility of failure. For instance, the method of harmonic balance is not useful to be applied on a type of second order systems, a system described by a single pair of integrator in series, such as satellite position control, ship position control and in general all inertial loads under low dumping coefficient. Consequently, using two autotuning methods contributes to increase the possibility of successes.

Inference procedure

When there are many variables in premises, the direct method of fuzzy reasoning has the following difficulties

- The number of rules increases exponentially with the number of premise-part variables
- As the number of rules increases, the task of constructing rules becomes excessively burdensome.

Takagi, Sugeno and Kang, [5], [9], proposed a method to solve these problems, based in a fuzzy reasoning mechanism, which consist in using linear functions for the consequence part. Such a reasoning method has the following features:

- The consequence part of the rule uses linear input-output functions
- It is possible the identification of rules on the basis of input-output data modelling.

The implementation of this method requires a modelling approach based in input-output data, which will be achieved from the virtual controller.
The general reasoning model by means of rules is

$$\text{IF } x_1 \text{ is } A_1 \text{ and. } x_n \text{ is } A_n \text{ THEN } Y_1 = f(x_1,...x_n) = C_0.x_1 + ...+ C_n.x_n \qquad (2)$$

This method is extremely useful in cases where the number of variables is high. For instance, processes such a position servo, a temperature controlled process, a power engine or a level controlled process, can operate with several level set points under different loads, and so on. So that, it is of particular interest to design controllers capable for control a process having into account the actual operating point and actual load for a wide range of set points and process loads. Those requirements add to the achieved controller two degree of freedom, which means two input variables more despite some classic error based controller variables as is the error, its derivative and /or its integral.

Theoretically, the number of rules to cover all possible input variations for a five term fuzzy controller is

$$n_1.x.n_2.x.n_3.x.n_4.x.n_5\,, \qquad (3)$$

where $n_1.x.n_2.x.n_3.x.n_4.x.n_5$ are the number of membership functions or linguistic labels of the five input variables. In a particular case, if $n_1 = n_2 = n_3 = n_4 = n_5 = 5$, then the number of rules will be 3125 as shown in figure 1. In practical applications, the implementation of such a large rule base will take a lot of reasoning time besides a large amount of process memory.

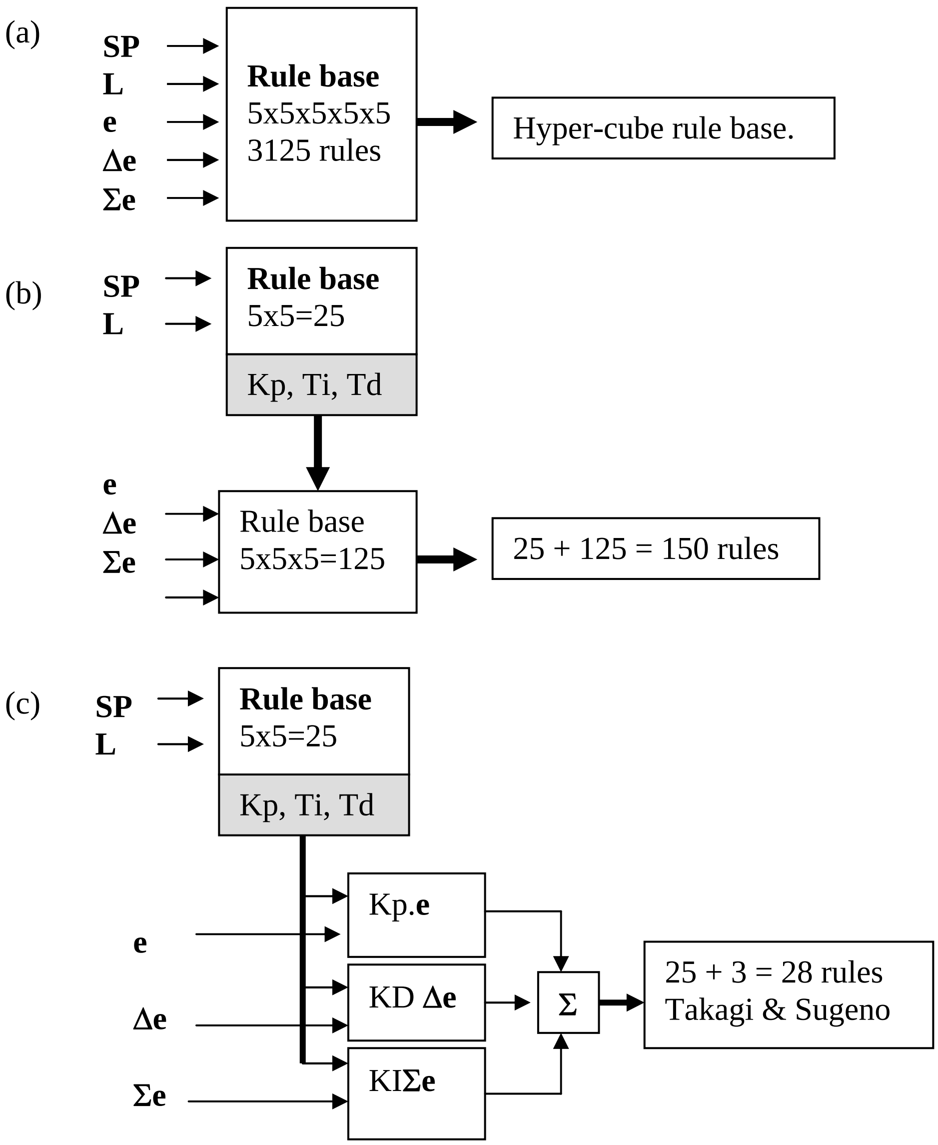
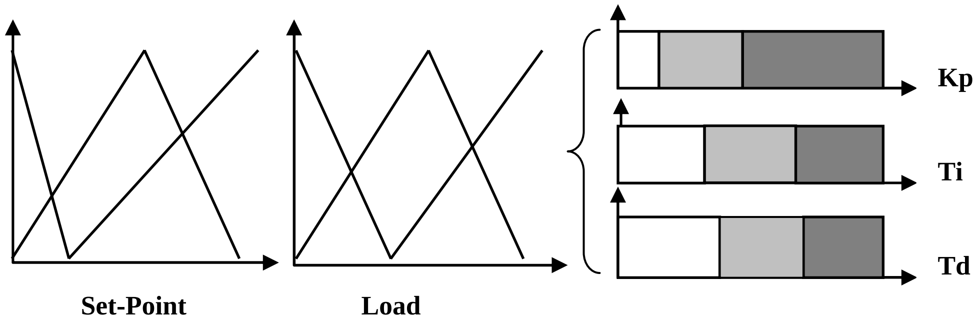
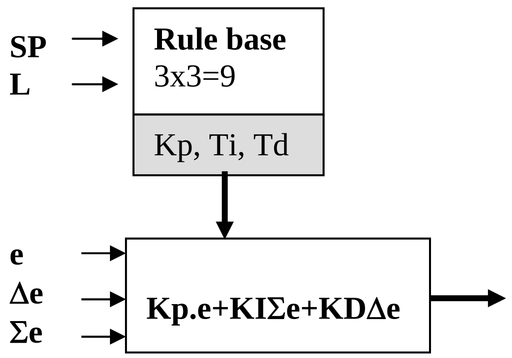


Fig. 1. Some strategies in reducing Rule bases. (a), Original hyper-cube rule base. (b), Two rule bases in series. (c), Reduction based on Takagi & Sugeno method.

Load / **Set-Point**	**Large**	**Medium**	**Small**
Large	Kp_1,Ti_1,Td_1	Kp_2,Ti_2,Td_2	Kp_3,Ti_3,Td_3
Medium	Kp_4,Ti_4,Td_4	Kp_5,Ti_5,Td_5	Kp_6,Ti_6,Td_6
Small	Kp_7,Ti_7,Td_7	Kp_8,Ti_8,Td_8	Kp_9,Ti_9,Td_9





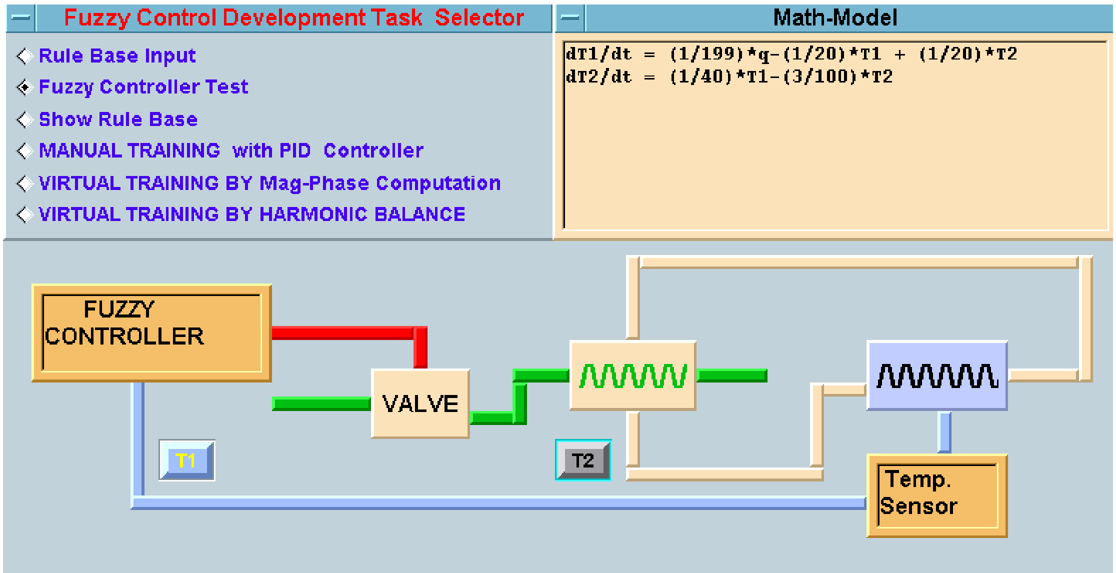
In contrast with last results, in the classical approach of an adaptive controller by gain scheduling strategy, the controller algorithm will quite similar:

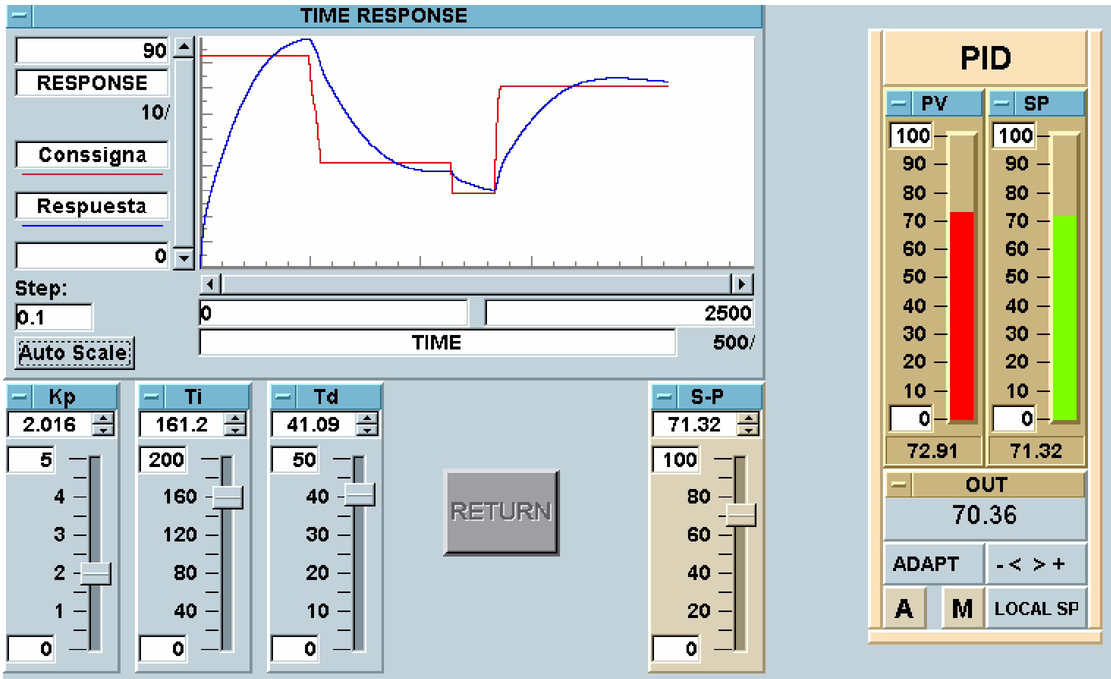
$$\left.\begin{array}{l} Kp = fp(SP,L) \\ Ki = fi(SP,L) \\ Kd = f(SP,L) \end{array}\right\} \quad U=f(Kp, Ki, Kd, e\ , \Delta e, \Sigma e)$$

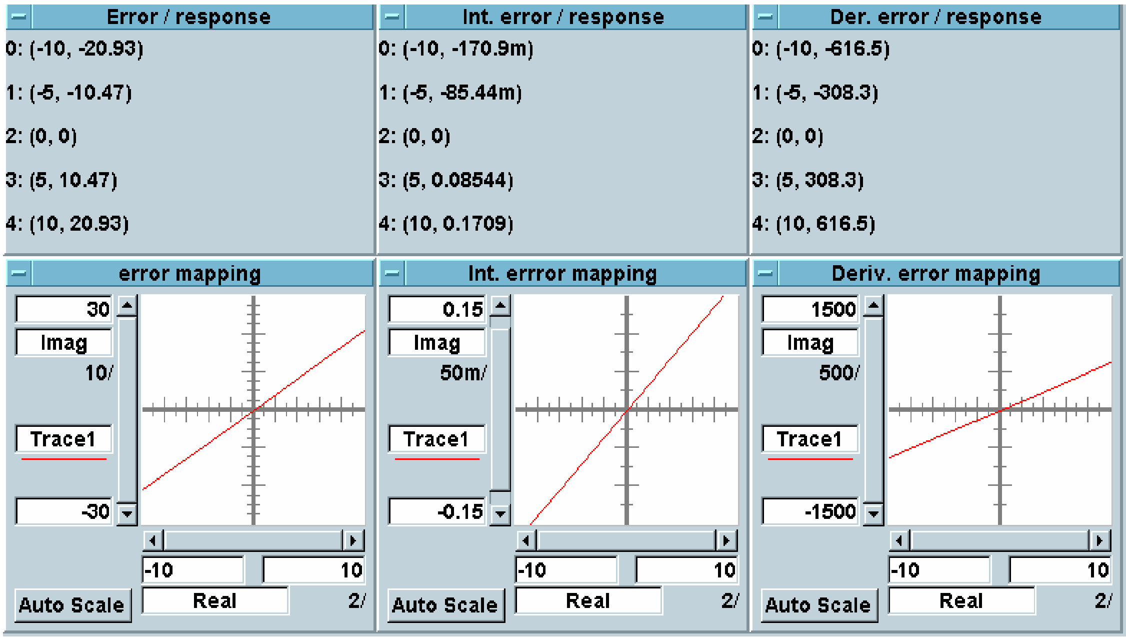
Automatic Rule base Generator Tool

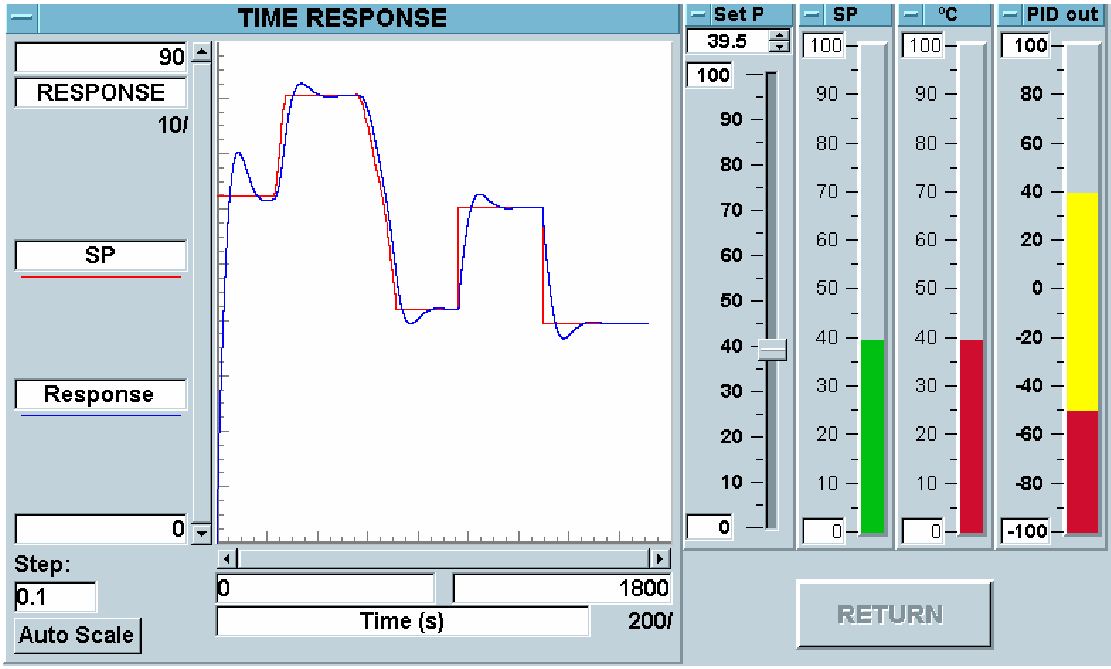
To implement any drastically reduced fuzzy rule base, it was developed an automatic tool with capacity to process in parallel input/output process variables. Such a tool operates on commercial hardware platforms with properly selected drivers, depending on the particular architecture, mainly on PCI, VXI and VME Buses.

Experimental results






Error / response
0: (-10, -20.93)
1: (-5, -10.47)
2: (0, 0)
3: (5, 10.47)
4: (10, 20.93)
Int. error / response
0: (-10, -170.9m)
1: (-5, -85.44m)
2: (0, 0)
3: (5, 0.08544)
4: (10, 0.1709)
Der. error / response
0: (-10, -616.5)
1: (-5, -308.3)
2: (0, 0)
3: (5, 308.3)
4: (10, 616.5)
error mapping
30
Imag
10/
Trace1
-30
-10
10
Auto Scale
Real
2/
Int. errror mapping
0.15
Imag
50m/
Trace1
-0.15
-10
10
Auto Scale
Real
2/
Deriv. error mapping
1500
Imag
500/
Trace1
-1500
-10
10
Auto Scale
Real
2/


TIME RESPONSE
90
RESPONSE
10/
SP
Response
0
Step:
0.1
Auto Scale
0
1800
Time (s)
200/
Set P
39.5
100
90
80
70
60
50
40
30
20
10
0
SP
°C
PID out
100
80
60
40
20
0
-20
-40
-60
-80
-100
RETURN

References

[1] Astrom, K. J. Wittenmark, On Self Tunning Regulators, Automatica 9(1973), 185-199 (autotuning)

[2] Astrom, K. J. And T. Hägglund, Automatic of PID Regulators. Research Triangle Parrk, N.C.: Instrument Society of America (1988) (autotuning)

[3] Charles L. Phillips and H. Troy Nagle, Digital control systems: Analysis and design. Prentice Hall Inc., Englewood Cliffs, N.J. (1982) 230.267

[4] F. Y. Wang and Hung-man Kim, Implementation adaptive fuzzy logic controllers with neural networks: a design paradigm, J Intelligent fuzzy systems 3 (1995) 165-180

[5] M. Sugno and G. T. Kang, Structure identification of fuzzy model, Fuzzy Sets and Systems 28 (1988) 15-33

[6] Minyou Chen and D.A. Linkens, A Hybrid neuro-fuzzy PID controller, Fuzzy sets and Systems 99 (1998) 27-36

[7] S. Tzafestas and N.P.Papanikolopoulos, Incremental fuzzy espert PID control , IEEE Trans. Ind. Electron. 37 (1990) 365-371

[9] Takagi and Sugeno, Fuzzy Identification of Systems and its Applications to Mdeling and Control. IEEE Transactions on Systems, Man aaand Cybernetics, 15,1 (1985) 116-132

[10] T. Brehm, Hybrid fuzzy logic PID controller, proc. 3rd IEEE Conf. On Fuzzy Systems 3 (1994) 1682-1687

[11] T.J. Procyk and E. H. Mamdani, A linguistic self-organizing proces controller, Automatica 15 (1979) 15-30

[12] Tsypkin, Y.Z., Relay Control Systems. Cambridge, U.K.: University Press. (1984) (Relay Autotuning)

[13] Z. Zhang and M. Mizumoto, On rule self-generating for fuzzy control, Int. J. Intelligent Systems 9(1994) 1047-1057