Rule Base Updating Method for Fuzzy Gain Scheduling of PID Controller

PAULI VILJAMAA
Automation and Control Institute
Tampere University of Technology
P. O. Box 692, FIN-33101 Tampere
FINLAND

HEIKKI N. KOIVO
Control Engineering Laboratory
Helsinki University of Technology
P. O. Box 5400, FIN-02015 HUT
FINLAND

Abstract: — A novel algorithm is developed to update the rule base of the fuzzy gain scheduling of the PID controller. The algorithm fulfills the following requirements: all data needed are stored in a rule base, this keeps the rule base small so that it can also be maintained manually, and it stores the PID parameters with a common base of operation points.

Key-Words: — nonlinear control, controller design

1 Introduction

Fuzzy control is successfully used for feedback control in many applications, see *e.g.*, [7, 1, 21, 5, 12]. However, because of the lack of general tuning methods, tuning is done experimentally. Thus there is no guarantee to obtain a good performance within certain time and effort limits for a particular application.

When fuzzy control is considered, an attractive alternative is to utilize fuzzy computing as a supervisory block for a conventional controller. This approach takes full advantage of the combination because fuzzy computing can thus be more easily applied in different control applications. Therefore the fuzzy system adds power to design methods of linear controllers.

Fuzzy gain scheduling can take the form of three different schemes. The fuzzy system can schedule the controller parameters with changing operation conditions, *e.g.*, according to the process output as is done by Ling and Edgar [11]. They used two fuzzy sets for the input which was the measurement signal, *i.e.*, the output of the process. Due to the low number of fuzzy sets, their proposal cannot be considered as a full scale gain scheduling scheme applicable for different nonlinear processes. Tan *et al.* [16] extended the scheme to neuro-fuzzy techniques by optimizing the shape of the two membership functions with respect to several operating point and parameter value pairs. It seems less applicable in practice than the previous approach.

Fuzzy gain scheduling can also mean a scheme that the controller parameters are changed as a function of the control error and change in the error, *e.g.*, [23]. This approach differs considerably from the conventional gain scheduling, *e.g.*, [3, Section 6.3], because the controller parameters are changed during transients even if the operation conditions are not changed.

The third meaning for the fuzzy gain scheduling is a fuzzy controller with Sugeno type rules [8]. Thus, the consequence of each rule can be interpreted as a local controller for the operating point that the premises define. Usually the structure of the controllers is the same for each rule, *e.g.*, a textbook PID controller or a state feedback controller, *e.g.*, in [13].

In practice, even the simplest PID control algorithm is much more complex than the textbook version of it. Thus it is very inefficient to perform the controller function for each rule separately as is done with Sugeno type rules. It is more natural to separate the fuzzy supervisor and the PID algorithm. It is also easier in design and tuning. Therefore Mamdani type reasoning [6] is utilized to obtain current parameter values for the PID controller. Additionally the controller parameters are usually needed to update more rarely than the controller output, and it also advocates Mamdani type fuzzy rules.

The presentation is organized as the following. The gain scheduling is described in Section 2. The method to update the rule base of the scheduler is developed in Section 3. The results are summarized in Section 4.

2 Problem Statement

This study considers *a controller driven approach* of gain scheduling implemented by fuzzy logic. Controller driven means that the scheduling block is interested in the tuning parameters of the controller and not in the design method of the controller and possible process model behind the design. The approach is mainly aimed for simple controllers like PID con-

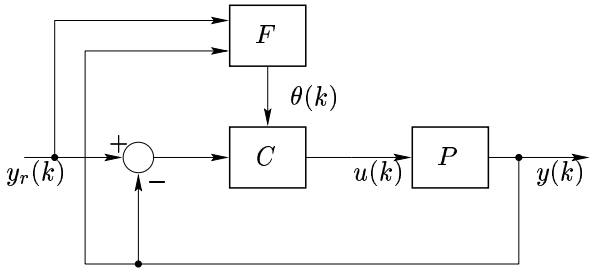

Figure 1: Gain scheduling with fuzzy logic. The fuzzy supervisor F above the process controller C detects the current operation point of the process P. The reference signal is used as a scheduling variable too.

trollers widely used in process control.

If the controller must be retuned at different operation conditions, it is natural to store the new parameters in a table. During the control the current parameters can be selected from the table according to the current operation conditions. Because the operating range of the parameters cannot be outlined exactly, the scheduling problem is suitable solving using fuzzy logic. Here the main benefit of fuzzy logic is that it provides a flexible and transparent mechanism for gain scheduling.

Usually the scheduling is performed according to the variable, which mostly affects the dynamics of the system [4]. It is called the scheduling variable. There can also be several scheduling variables. Most typical variable is the operation point. Disturbances can also change the dynamics. In this study the system output is utilized as a scheduling variable.

Although the controller parameters are scheduled based on the system output, the reference signal is also utilized as a scheduling variable (Fig. 1) in order to achieve better response immediately after the set-point changes. Thus, the fuzzy system provides tighter or looser tuning based on the next operating point as presented in [20]. If the set-point is usually kept constant or it is changed only slowly, it is not necessarily needed as a scheduling variable.

The parameters of the controller are assumed to be functions of the system output and the reference signal $\theta(y(k), y_r(k))$. Thus the fuzzy logic rule base must include rules of the form

> if y is medium and y_r is big then K_p is small and T_i is medium and T_d is big.

Because the controller parameters are tuned at several separate operating points, the system output will have the same number of fuzzy sets as the number of the tunings. The membership functions can be triangles, the cores of which are the operating points where the parameters of the controller are tuned. The fuzzy sets and the membership functions of the reference signal will be identical with the system output. Singletons are used as membership functions for the output of the fuzzy system. The parameterization θ_in, θ_rules, θ_out suggested in [22, 19] is adopted in this study.

The rule base is composed of the table of the tuning parameters. The operation points where the tuning is performed are stored in a knot vector

$$p_y = \begin{bmatrix} y_0^{(1)} & y_0^{(2)} & \dots & y_0^{(m_y)} \end{bmatrix}^T \qquad (1)$$

which defines the membership functions of the process variable. The operation points $y_0^{(i)}$ must be stored in an increasing order. The membership functions of the reference signal are chosen to be the same, $p_{y_r} = p_y$. Actually, the signals describe the same information about the operation conditions and therefore there is no reason to have unequal fuzzy sets. Different values of the tuning parameters $\theta^{(i)}$ are stored in the matrix

$$\theta = \begin{bmatrix} \theta^{(1)T} & \theta^{(2)T} & \dots & \theta^{(m_y)T} \end{bmatrix}^T \qquad (2)$$

correspondingly, where $\theta^{(i)}$ denotes the ith row of θ.

Now we have two alternatives to build up the fuzzy system. The supervisor block can have a two-input fuzzy system or can consist of two one-input fuzzy systems. In the latter case, both fuzzy systems have exactly the same parameters, $\theta_\mathrm{in} = p_y^T$, $\theta_\mathrm{rules}^{(i)} = i$ ($i = 1, 2, \dots, m_y$), and $\theta_\mathrm{out} = \theta$. The parameter value applied to control is a weighted sum

$$\begin{aligned} \theta(k) = {} & w_y F(y(k), \theta_\mathrm{in}, \theta_\mathrm{rules}, \theta_\mathrm{out}) \\ & + w_{y_r} F(y_r(k), \theta_\mathrm{in}, \theta_\mathrm{rules}, \theta_\mathrm{out}), \end{aligned} \qquad (3)$$

where w_y and w_{y_r} are the weighting factors for the process variable and the reference signal, and $F(\cdot)$ stands for fuzzy computing. Usually the effect of the reference signal is retained to be slight by selecting $w_y \approx 0.9$ and $w_{y_r} = 1 - w_y$.

The weights w_y and w_{y_r} can be adjusted such that first check if the scheduling works with $w_y = 1$ and $w_{y_r} = 0$ with small setpoint changes relative to the changes in θ_in. If that is acceptable but larger setpoint changes cause remarkable overshoot or undershoot in the response, adjustment of the weights is necessary. Reduce the effect of the system output decreasing w_y slightly and compute w_{y_r} respectively until the response is acceptable. This makes the scheduling algorithm predict the near future operating point.

The fuzzy system provides the parameters of the controller according to the current values of the reference signal and the system output, based on the fuzzy

parameters θ_{in}, θ_{rules} and θ_{out} at each sampling instant. The rule base is computed from p_y and θ. The rule base can be formed when the controller is tuned at different operation points. The problem how to restrict the size of θ_{in}, θ_{rules} and θ_{out} is addressed in the next section.

3 Updating of Rule Base

The maintenance of the rule base of the fuzzy gain scheduler needs more detailed consideration.

One idea is that the operator is supervising the system. When the performance is observed to be less successful, the controller is retuned for the current operation point. The tuning task can be automated. The automatic tuning is not dealt here because many well-known methods exist [2, 15]. To obtain the whole benefit of fuzzy gain scheduling, the new tuning parameters at the new operation point should be added automatically to the rule base.

The problem is reduced to an update of the scalar approximation between the operation point and the tuning parameter. Fuzzy logic is utilized to implement a piece-wise linear approximation. The approximation parameters are updated when a new controller tuning is obtained. The tuning task cannot be performed often. Thus, sample points of the mapping are obtained only rarely and they cannot be selected arbitrarily.

An adaptive approximation [10] is a method where new knots, *i.e.*, new elements for p_y in (1), are selected based on the maximum deviation. Interpolation intervals are subdivided to smaller subdomains in order to obtain more accurate approximation. The function to be approximated must be known or a huge amount of sample points are needed. Thus, the adaptive approximation is not suitable for this application.

The objective of this research is to develop a simple method for the updating. The requirements for the method are the following. The amount of needed data must be small. The number of accepted knots must be restricted so that the maintenance of the rule base can also be performed manually. The method must take care of addition, replacement and update of tuning points, *i.e.*, the elements of the knot vector (1). All stored values in the rule base should be untouched, *i.e.*, they must be actual values obtained from the procedure of the controller tuning. The developed method is presented in more detail in the next section.

3.1 Updating of piece-wise linear approximation

Consider the problem where an unknown function

$$y = f(x), \tag{4}$$

where $x, y \in \mathbb{R}$, must be approximated by a piece-wise linear approximation. The approximation is parametrized with the knot vector

$$x = \begin{bmatrix} x^{(1)} & x^{(2)} & \dots & x^{(m)} \end{bmatrix}^T \tag{5}$$

and the corresponding ordinate vector

$$y = \begin{bmatrix} y^{(1)} & y^{(2)} & \dots & y^{(m)} \end{bmatrix}^T \tag{6}$$

where $y^{(i)} = f(x^{(i)})$. The elements of the knot vector are stored in an increasing order.

The approximation is implemented with fuzzy logic so that vector x includes the cores of the input membership functions and vector y includes the places of the output singletons. The approximation is piece-wise linear interpolation between the knots.

The fuzzy logic gain scheduling can be utilized after the controller is tuned at least at two distinct operation points. Thus even in the simplest case, two knots are needed. From the practical point of view, the number of the knots must also be limited. The memory requirements and the computational load must be considered, but the most important aspect is the maintenance of the rule base. It must be also possible manually. Thus, the maximum number of the knots M is relatively small.

Consider first a situation where only two knots have been obtained. The method can add new knots to the rule base until the maximum M is reached. The new knot x_{f} is stored in the knot vector x so that the increasing order of the knots is preserved. The new ordinate y_{f} is stored in the ordinate vector y respectively. If x_{f} is the same with some old knot $x^{(i)}$, then the corresponding ordinate $y^{(i)}$ is replaced with the new ordinate y_{f}.

When the maximum number of knots M has been reached and a new tuning, *i.e.*, a new knot x_{f} and a new ordinate y_{f}, is obtained, the basic method cannot be applied. Now there are several alternatives: the new knot could be ignored, an old knot could be replaced with it, or the nearest knot could be updated. The selection of the cases must be based on a criterion. Because function f is unknown except for the stored knots and ordinates, the criterion must depend only on the observations. The developed criterion is described next.

The new observation is firstly appended to the knot vector and to the ordinate vector. Thus, the size allocated for the vectors must be $M + 1$ elements. Then

each knot and ordinate are removed from the approximation, and the difference between the long and the short approximation is evaluated at the removed knot

$$e^{(i)} = y^{(i)} - \left(\frac{y^{(i+1)} - y^{(i-1)}}{x^{(i+1)} - x^{(i-1)}} \left(x^{(i)} - x^{(i-1)} \right) + y^{(i-1)} \right) \quad (7)$$

for $i = 2, \ldots, M$, where the first term, $y^{(i)}$, is the value of the long approximation and the remaining part is the interpolated value of the approximation if the data pair $(x^{(i)}, y^{(i)})$ is ignored. The criterion for ignoring the first and the last data pair is expressed as

$$e^{(1)} = y^{(1)} - y^{(2)} \quad (8)$$

$$e^{(M+1)} = y^{(M+1)} - y^{(M)}, \quad (9)$$

because the approximation is implemented with fuzzy logic where the first and the last membership functions saturate to one outside the knots.

Criterion (7)–(9) is utilized so that the data pair $(x^{(i)}, y^{(i)})$, for which the absolute value of e is the smallest, is removed. Thus the approximation accuracy deteriorates the least. Therefore the method is called the *principle of the slightest detriment*.

The algorithm can be further improved so that the new knot is directly accepted to the approximation, if some of the input membership functions exceed the adjustable threshold value μ_{thr}. Thus the corresponding old knot is directly replaced with the new knot

$$x^{(i)} = x_{\mathrm{f}},\ y^{(i)} = y_{\mathrm{f}},\ \text{ if } \mu^{(i)}(x_{\mathrm{f}}) > \mu_{\mathrm{thr}} \quad (10)$$

for $i = 2, \ldots, m-1$, where m is the number of the knots and $\mu_{\mathrm{thr}} \in (\frac{1}{2}, 1]$. If $\mu_{\mathrm{thr}} = 1$, the direct replacement is not performed. When $\mu_{\mathrm{thr}} \approx 0.5$, a new knot is nearly always accepted. An advantage of the improvement is that a randomly disturbed observation or an obsolete knot can be forgotten when new information is obtained. In the basic criterion, an incorrect observation can be preserved, if the criterion is not the smallest for the knot. On the boundaries, the saturation of the membership functions must be taken into account and the direct replacement is allowed only if

$$\mu^{(i)}(x_{\mathrm{f}}) = \max_j(\mu^{(j)}(x_{\mathrm{f}})) \ \wedge \ \min(\mu^{(i)}(x_{\mathrm{f}}), \mu^{(i)}(2x^{(i)} - x_{\mathrm{f}})) > \mu_{\mathrm{thr}} \quad (11)$$

for $i = 1, m$. Thus the smallest and the largest fuzzy sets are interpreted as symmetric triangles and the observation far from the saturation point does not distort the approximation.

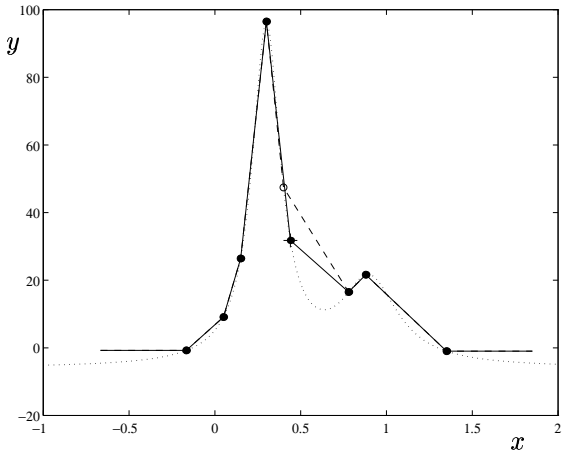

Figure 2: An old observation ($\circ$) is ignored and the new observation ($+$) is added to the approximation.

3.2 Numerical example

The updating is illustrated with an example where a function (MATLAB function `humps`)

$$f(x) = \frac{1}{(x-0.3)^2 + 0.01} + \frac{1}{(x-0.9)^2 + 0.04} - 6 \quad (12)$$

will be approximated in the interval $[-0.5, 1.5]$. The number of knots is restricted to $M = 8$. The straightforward replacement is not utilized ($\mu_{\mathrm{thr}} = 1$).

Firstly, let the knot vector be $x = [-0.5, -0.45]$. Then the function (12) is evaluated with the step size of 0.05 up to value $x_{\mathrm{f}} = 1.5$. The interval $[0.4, 0.8]$ is skipped at this stage. The algorithm performs the updating very well and all significant observations are stored in the knot and the ordinate vectors.

Then the input is decreased step-by-step to the value $x_{\mathrm{f}} = -0.5$ with slightly different values. This checks if the order of the observations affects the selection of the knots. In this example it does not.

After that function (12) is evaluated in the interval $[0.4, 0.8]$. Thus the algorithm must select an old observation which could be ignored. In Fig. 2, the new observation is updated to the approximation and the approximation accuracy does not deteriorate between the fourth and the fifth knots. In Fig. 3, the sixth knot has been moved to the left of the local minimum and the new observation cannot be utilized. The approximation accuracy would deteriorate more in any other knot than it will be improved at the observation point.

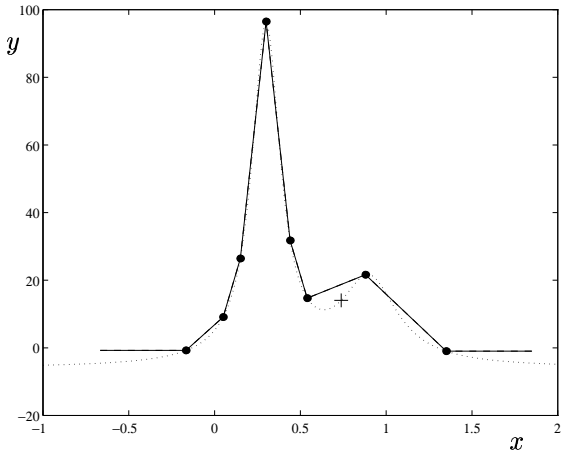

Figure 3: The new observation (+) cannot be added to the approximation.

3.3 Combination of gain scheduling and updating method

Now we have a concept of fuzzy gain scheduling and a method to update scalar piece-wise linear approximation automatically. When PID control is considered, three parameters should be scheduled. This affects an extra requirement for the updating algorithm. It is natural that the tuning values are kept consistent. It means that the array of the operation points is the same for each tuning parameter and therefore $x^{(i)} = y_0^{(i)}$ and $y^{(i)} = \begin{bmatrix} {K_\mathrm{p}}^{(i)} & {T_\mathrm{i}}^{(i)} & {T_\mathrm{d}}^{(i)} \end{bmatrix}$ in (5) and (6). Thus the automatic update mechanism must be adopted slightly.

The problem is that now there are three criteria $J = \begin{bmatrix} |e_\mathrm{p}| & |e_\mathrm{i}| & |e_\mathrm{d}| \end{bmatrix}$ instead of one, $J = |e|$. The point to be removed is suggested based on the principle of the slightest detriment. In the case of the PID parameters, the principle is applied by means of Pareto optimality. Otherwise, the selection criteria must be reformulated to be a scalar criterion. The other choices might be a linear combination of the criteria or a totally different approach like a loss function of the closed loop response.

The linear combination depends on the weights and therefore it has no unique solution. The loss function approach evaluated from the closed loop step response is computationally more demanding. The method needs the process model for each tuning and its performance index in the form of, *e.g.*, sum of squared errors between the reference and the output. Additionally, the performance index must be evaluated for each removed knot in order to achieve the final selection criteria. This might mean simulations for the performance index [9]. The selection of the performance index affects also the time domain performance of the closed loop system. Some methods allow, *e.g.*, more overshoot than others [9].

Therefore, the three criteria are not converted to a scalar criterion, but the selection is done based on methods well-known in multiple criteria optimization.

3.3.1 Multiple criteria optimization

In the optimizing tasks there are often multiple objectives to be considered. In general the objectives are conflicting and no solution exists which is the best with respect to all objectives. Hence a trade-off cannot be avoided [18].

The treatment of a multiobjective optimization problem leads to Pareto optimal solutions. The solution of the multiobjective optimization problem

$$\min_{i \in I} J^{(i)}, \; I = \{1, 2, \ldots, M+1\} \qquad (13)$$

can be considered to consist of the set of non-inferior, or Pareto optimal, points [14]. A solution $i^* \in I$ is a Pareto optimal solution of the problem if there exists no other solution $i \in I$ such that $J^{(i,j)} \leq J^{(i^*,j)}$, $\forall j$ and $J^{(i,j)} < J^{(i^*,j)}$ for at least one j. Thus a non-inferior solution to the problem has the property that it is not possible to reduce any of the objectives without making at least one of the other objectives worse [18].

The set of Pareto optimal points are the points which are of interest in multiobjective optimization, and the search for a satisfactory point should be made in this set of points. The point which should be chosen from the set of Pareto optimal points depends on the preferences of the decision maker [17].

An algorithm for solving a multiobjective optimization problem should assist the decision maker to find a satisfactory Pareto optimal point in an efficient way. Clearly, an algorithm for multiobjective optimization should be able to generate Pareto optimal points [17].

In this case, the values of the criterion function can be obtained easily, and the number of them $M + 1$ is relatively low. Therefore, a rather crude algorithm is adequate to search for Pareto optimal candidates. In this study, the non-inferior alternatives are searched by checking each solution with respect to others, if any of them are Pareto dominant to the current alternative. If no Pareto optimal rival candidates exist, the current alternative is Pareto optimal. The algorithm needs $2n_z(M+1)M$ comparisons, where n_z is number of the scheduled parameters [19].

3.3.2 Updating example

The Pareto optimal selection is illustrated with the following example. Let the controller parameters be polynomial functions of the operation point as follows

$$\begin{aligned} K_{\mathrm{p}}(x) &= 18.6x^5 - 9.44x^4 - 12.5x^3 + 9.56x^2 \\ &\quad + 3.91x + 20.1 \\ T_{\mathrm{i}}(x) &= 18.9x^5 - 32.2x^4 + 6.69x^3 + 3.89x^2 \\ &\quad - 1.26x + 8.04 \\ T_{\mathrm{d}}(x) &= -2.83x^5 + 14.9x^4 - 26.1x^3 + 20.1x^2 \\ &\quad - 6.98x + 1.07. \end{aligned} \tag{14}$$

The parameters are approximated with the fuzzy system given by

$$\begin{aligned} \theta_{\mathrm{in}} &= \begin{bmatrix} 0 & 0.1 & 0.2 & \dots & 1 \end{bmatrix} \\ \theta_{\mathrm{rules}} &= \begin{bmatrix} 1 & 2 & \dots & 11 \end{bmatrix}^T \\ \theta_{\mathrm{out}} &= \begin{bmatrix} 20.1 & 8.04 & 1.07 \\ 20.6 & 7.96 & 0.548 \\ 21.2 & 7.95 & 0.292 \\ 21.8 & 7.98 & 0.194 \\ 22.3 & 7.96 & 0.176 \\ 22.9 & 7.80 & 0.185 \\ 23.4 & 7.43 & 0.191 \\ 24.1 & 6.80 & 0.183 \\ 25.2 & 5.95 & 0.163 \\ 27.0 & 4.97 & 0.147 \\ 30.2 & 4.06 & 0.160 \end{bmatrix} \end{aligned} \tag{15}$$

One of the rules $i^* = \{3, 4, 5, 6, 9, 10\}$ can be removed by the means of the Pareto optimality and thus the approximation accuracy would deteriorate

$$J^{(i^*)} = \begin{bmatrix} 0.0000 & 0.0200 & 0.0790 \\ 0.0500 & 0.0250 & 0.0400 \\ 0.0500 & 0.0700 & 0.0135 \\ 0.0500 & 0.1050 & 0.0015 \\ 0.3500 & 0.0650 & 0.0020 \\ 0.7000 & 0.0350 & 0.0145 \end{bmatrix} \tag{16}$$

with respect to the original approximation (15). Rule number 5 is selected to be removed. The output of the system before and after the removal is shown in Fig. 4.

The concept of Pareto optimality can be used to eliminate the possible candidates to a smaller set. In this example, eleven potential candidates are eliminated to six non-inferior solutions, among which the user selects one.

3.4 Limitations of the updating method

The method developed to update the rule base has some limitations. It

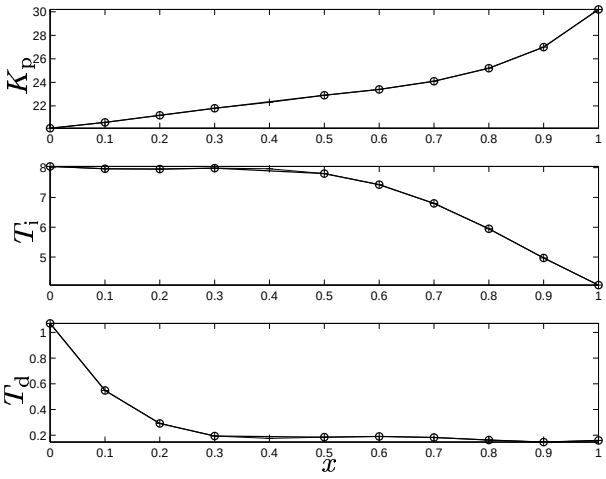

Figure 4: Output of the fuzzy system before (thin $+$) and after (thick $\circ$) the removal of rule number 5.

- needs the intervention of human operators (to select one of the non-inferior solutions),
- is not suitable for processes where an auxiliary input is needed as a scheduling variable in addition to the process output (can not handle rule base of multi-dimensional Cartesian grid), and
- may preserve erroneous tuning parameters.

The first limitation is not serious. Traditionally human operators check the controller parameters before accepting them. When the selection of a point to be removed is combined with the checking, it should not restrict the operator too much. This also has an advantage. The concept is controller driven not model driven gain scheduling. If the new controller parameters are acceptable, they are stored in the rule base, and during control, the actual values of the parameters are calculated on-line from the rule base. If gain scheduling in general works with the process, then probably the concept works also.

The second issue limits the feasibility of the method. If more than one scheduling variable is needed (the process output and the reference signal are regarded as one here), a Cartesian grid of tuning points is necessary. However, the tuning points can not be placed arbitrarily, and therefore the Cartesian grid is impossible to cover. Also the updating algorithm as such does not fit with a multi-dimensional problem. Further work is needed in order to solve the problem.

The latter disadvantage is related to the selection criteria which tends to retain knots that are far from a straight line between its neighboring knots. Thus possible outliers are not suggested to be removed from the rule base. Therefore the algorithm has the threshold parameter which is used in the check, if there is any

old knot near to the fresh observation, and if it should be replaced directly. But in this case outliers should be rare because the new parameters are checked by the operator before any further analysis.

4 Conclusion

The controller driven approach of the gain scheduling is implemented by fuzzy logic. The main benefit of fuzzy logic is that it provides a flexible and transparent mechanism for gain scheduling. The reference signal is suggested for use as a scheduling variable with the system output in order to predict the near future operating point. A new method to automate the updating of the piece-wise linear approximation is proposed. An advantage of the method is that it requires only small amounts of data and it is stored in the knot information, *i.e.*, in the rule base. The method is integrated to the fuzzy gain scheduling of the PID controller. The concept of Pareto optimality is utilized in the multiple criteria optimization needed in the integration.

References

[1] C. von Altrock, *Fuzzy Logic and Neurofuzzy Applications Explained*, Prentice Hall, Upper Saddle River, NJ, 1995.

[2] K. J. Åström and T. Hägglund, *Automatic Tuning of PID Controllers*, Instrument Society of America, Research Triangle Park, NC, 1988.

[3] —, *PID Controllers: Theory, Design, and Tuning*, Instrument Society of America, Research Triangle Park, NC, 2nd ed., 1995.

[4] K. J. Åström and B. Wittenmark, *Adaptive Control*, Addison-Wesley, Reading, MA, 1989.

[5] J. Franssila and H. N. Koivo, Fuzzy control of an industrial robot in transputer environment, in *IEEE International Conference on Industrial Electronics, Control, Instrumentation and Automation IECON'92*, San Diego, CA, USA, Nov. 9–13, 1992, pp. 624–629.

[6] R. Jager, *Fuzzy Logic in Control*, Ph.D. thesis, Technische Universiteit Delft, Jun. 1995.

[7] L. C. Jain, R. P. Johnson, Y. Takefuji and L. A. Zadeh (eds.), *Knowledge-Based Intelligent Techniques in Industry*, CRC Press, Boca Raton, FL, 1999.

[8] J.-S. R. Jang, C.-T. Sun and E. Mizutani, *Neuro-Fuzzy and Soft Computing*, Prentice-Hall, Upper Saddle River, NJ, 1997.

[9] T. Jussila, *Computational Issues of PID Controllers*, Diss., Tampere University of Technology, 1992.

[10] D. Kincaid and W. Cheney, *Numerical Analysis*, Brooks/Cole Publishing Company, Pasific Grove, CA, 1991.

[11] C. Ling and T. F. Edgar, A new fuzzy gain scheduling algorithm for process control, in *American Control Conference ACC'92*, Chicago, USA, vol. 3, Jun. 1992, pp. 2284–2290.

[12] A. Makkonen and H. N. Koivo, Fuzzy control of a nonlinear servomotor model, in *3rd Int. Workshop on Advanced Motion Control*, Berkeley, CA, USA, Mar. 20–23, 1994, pp. 833–841.

[13] R. Palm and U. Rehfuess, Fuzzy controllers as gain scheduling approximators, *Fuzzy Sets and Systems*, vol. 85, no. 2, 1997, pp. 233–246.

[14] R. E. Steuer, *Multiple Criteria Optimization: Theory, Computation, and Application*, John Wiley & Sons, New York, 1986.

[15] K. K. Tan, *Towards Autonomous Process Control with Relay-Based Feedback Identification*, Ph.D. thesis, National University of Singapore, 1994.

[16] S. Tan, C.-C. Hang and J.-S. Chai, Gain scheduling: from conventional to neuro-fuzzy, *Automatica*, vol. 33, no. 3, 1997, pp. 411–419.

[17] H. T. Toivonen, Lecture notes on linear quadratic control with process application, Tech. rep., Process Control Laboratory, Department of Chemical Engineering, Åbo Akademi, Jun. 1991.

[18] H. T. Toivonen and P. M. Mäkilä, Computer-aided design procedure for multiobjective LQG control problems, *International Journal of Control*, vol. 49, no. 2, 1989, pp. 655–666.

[19] P. Viljamaa, *Fuzzy Gain Scheduling and Tuning of Multivariable Fuzzy Control—Methods of Fuzzy Computing in Control Systems*, Diss., Tampere University of Technology, Tampere, Finland, Jun. 2000.

[20] P. Viljamaa and H. N. Koivo, Fuzzy logic in PID gain scheduling, in *Third European Congress on Fuzzy and Intelligent Technologies EUFIT'95*, ELITE-foundation, Aachen, Germany, vol. 2, Aug. 28–31, 1995, pp. 927–931.

[21] —, Multivariable fuzzy logic controller of pilot headbox process, in *13th World Congress of IFAC*, IFAC, San Francisco, CA, USA, vol. N, Jun. 30 – Jul. 5, 1996, pp. 379–384.

[22] —, Comparison of minimum and product operators in fuzzy systems with more than two inputs, in *IASTED Int. Conf. on Control and Applications CA'98*, IASTED, Honolulu, HI, USA, vol. 1, Aug. 12–14, 1998, pp. 69–73.

[23] Z.-Y. Zhao, M. Tomizuka and S. Isaka, Fuzzy gain scheduling of PID controllers, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 5, 1993, pp. 1392–1398.