Tuning Of Fuzzy PID Controllers

Jan Jantzen <jj@iau.dtu.dk>[1]

Abstract

Since fuzzy controllers are nonlinear, it is more difficult to set the controller gains compared to proportional-integral-derivative (PID) controllers. This research paper proposes a design procedure and a tuning procedure that carries tuning rules from the PID domain over to fuzzy single-loop controllers. The idea is to start with a tuned, conventional PID controller, replace it with an equivalent linear fuzzy controller, make the fuzzy controller nonlinear, and eventually fine-tune the nonlinear fuzzy controller. This is relevant whenever a PID controller is possible or already implemented.

1. Introduction

When the control problem is to regulate the process output around a setpoint, it is natural to consider *error* as an input, even to a fuzzy controller, and it follows that the integral of the error and the derivative of the error may be useful inputs as well. In a fuzzified PID controller, however, it is difficult to tell the effect of each gain factor on the rise time, overshoot, and settling time, since it is most often nonlinear and has more tuning gains than a PID controller. The objective in this paper is to find a systematic tuning procedure by carrying PID tuning rules over to the fuzzy domain. A systematic tuning procedure would make it easier to install fuzzy controllers, and it might pave the way for auto-tuning of fuzzy controllers.

PID controllers may be tuned in a variety of ways, including hand-tuning, Ziegler-Nichols tuning, loop shaping, analytical methods, by optimisation, pole placement, or auto-tuning (Smith, 1979; Åström & Hägglund, 1995). Furthermore, fuzzy controllers show similarities with PID controllers under certain assumptions (Siler & Ying, 1989; Mizumoto, 1992; Qiao & Mizumoto, 1996; Tso & Fung, 1997). But there is still a gap, it seems, between the PID tuning methods and a design strategy for fuzzy controllers of the PID type.

This paper proposes a design strategy, which makes use of known PID design techniques, before implementing the fuzzy controller:

1. Tune a PID controller
2. Replace it with an equivalent linear fuzzy controller
3. Make the fuzzy controller nonlinear

[1] Technical University of Denmark, Department of Automation, Bldg 326, DK-2800 Lyngby, DENMARK. Tech. report no 98-H 871 (fpid), 30 Sep 1998.

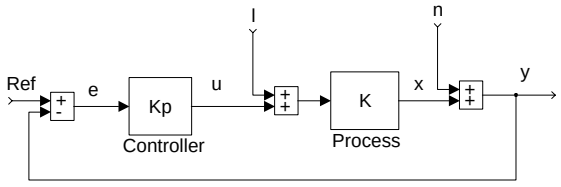

Figure 1: Proportional control with load l and noise n.

4. Fine-tune it

It seems sensible to start the controller design with a crisp PID controller, maybe even just a P controller, and get the system stabilised. From there it is easier to go to fuzzy control. Each step will be investigated in the following.

2. Tuning a PID controller

The first step in the design strategy is to install and tune a PID controller. The ideal continuous PID controller

$$u = K_p \left(e + \frac{1}{T_i} \int_0^t e * d\tau + T_d \frac{de}{dt} \right) \tag{1}$$

returns the controller output u, the constant K_p is the *proportional gain*, T_i is the *integral time*, T_d the *derivative time*, and e is the *error* between the reference and the process output y ($e = Ref - y$). We are concerned with digital control, and for small sampling periods T_s, the equation may be approximated by a discrete approximation. Replacing the derivative term by a backward difference and the integral by a sum using rectangular integration, an approximation is

$$u_n = K_p \left(e_n + \frac{1}{T_i} \sum_{j=1}^{n} e_j T_s + T_d \frac{e_n - e_{n-1}}{T_s} \right) \tag{2}$$

Index n refers to the time instant. By *tuning* we shall mean the activity of adjusting the parameters K_p, T_i, and T_d.

Several tuning aspects may be illustrated by static considerations (Åström & Hägglund, 1995). For purely proportional control ($T_d = 0$ and $1/T_i = 0$), the control law (2) reduces to

$$u_n = K_p e_n \tag{3}$$

Consider the feedback loop in Fig. 1, where the controller has the proportional gain K_p and the process has the gain K in steady state. The process output x is related to the reference Ref, the load l, and the measurement noise n by the equation

$$x = \frac{K_p K}{1 + K_p K}(Ref - n) + \frac{K}{1 + K_p K} l \tag{4}$$

Controller	K_p	T_i	T_d
P	$0.5K_u$		
PI	$0.45K_u$	$T_u/1.2$	
PID	$0.6K_u$	$T_u/2$	$T_u/8$

Table 1: The Ziegler-Nichols rules (frequency response method)

If n and l are zero, then K_p should be high in order to ensure that the process output x is close to the reference Ref. Furthermore, if l is nonzero, a high value will make the system less sensitive to changes in the load l. But if n is nonzero K_p should be moderate, otherwise the system will be too sensitive to noise. If the process dynamics are considered, the closed loop system will normally be unstable if K_p is high. Obviously the setting of K_p is a balance between the control objectives: stability, noise sensitivity, and load regulation. A PID controller may be tuned using the *Ziegler-Nichols frequency response method* (Ziegler & Nichols in Åström & Hägglund, 1995).

Procedure Ziegler-Nichols.

(a) Increase the proportional gain until the system oscillates; that gain is the ultimate gain K_u.
(b) Read the time between peaks T_u at this setting.
(c) Table 1 gives approximate values for the controller gains.

□

The sample period may be related to the derivative gain T_d. Åström and Wittenmark (1984) suggest that the sample period should be between $1/10$ and $1/2$ of T_d. In connection with the Ziegler-Nichols rules, this implies that T_s should approximately equal $1-5$ percent of the ultimate period T_u. Another rule says that it should be chosen somewhat smaller than the dominating time constant in the process, for instance between $1/10$ and $1/5$ of that time constant.

Ziegler and Nichols also give another method called the *reaction curve* or *step response* method (see for example Åström & Hägglund, 1995). That method uses the open loop step response to find the gains, and this is an advantage if oscillations in the closed loop system cannot be tolerated.

Example 1 (Z-N tuning) *Assume the process in Fig.* 1 *has the transfer function*

$$G(s) = \frac{1}{(s+1)^3}$$

Insert a PID controller with differential and integral action removed by setting $T_d = 0$ *and* $1/T_i = 0$. *Gradually increase the proportional gain until it reaches a stable oscillation (Fig.* 2*). This gain is* $K_u = 8$ *and the ultimate period is approximately* $T_u = 15/4$. *There is a load on the system (cf. Fig.* 1*), therefore the controller must contain an integrator. The third row in Table* 1 *implies* $K_p = 0.6 * K_u = 4.8$, $T_i = T_u/2 = 15/8$, *and* $T_d = T_u/8 = 15/32$. *Figure* 3 *shows the closed loop response after a step in the reference (at time equal*

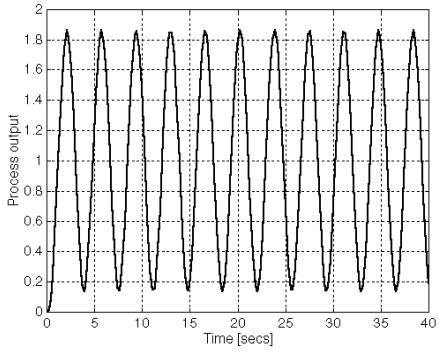

Figure 2: Ziegler-Nichols oscillation of process $1/\left(1+s\right)^3$.

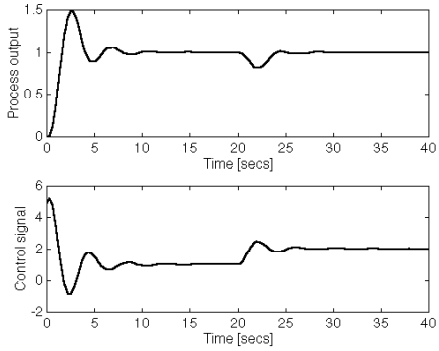

Figure 3: PID control of $1/(1+s)^3$. A reference step at 0 seconds is followed by a load step at 20 seconds; Ziegler-Nichols tuning.

Action	Rise time	Overshoot	Stability
Increase K_p	faster	increases	gets worse
Increase T_d	slower	decreases	improves
Increase $1/T_i$	faster	increases	gets worse

Table 2: Rules of thumb for tuning PID controllers.

to zero) and the load (after 20 seconds).

Ziegler and Nichols derived the rules for a linear system with a time lag and an integrator. Their design criterion is to obtain a decay ratio of one quarter; *decay ratio* is the ratio between two consecutive peaks of the error after a step change in reference or load. Thus in a quarter-decay response the second overshoot is 25 % of the first − a compromise between a fast response and a small overshoot. The results are poor for systems where the time lag is much greater than the dominating time constant. In general, the rules often result in rather poor damping, but they do provide the right magnitude of the gains.

A related, more accurate method is *Kappa-Tau tuning* based on the dimensionless parameters: relative gain κ and relative deadtime τ (Åström & Hägglund, 1995; Åström, Hang, Persson & Ho, 1992).

The gains found by either method, must sometimes be regarded as approximate values, a starting point for a hand-tuning. Hand-tuning is based on certain rules of thumb used by experienced process engineers (Table 2). The tuning is a compromise between fast reaction and stability. There are exceptions to the rules in the table. If, for example, the process contains an integrator, an increase in K_p often results in more stable control. The rules of thumb may also be illustrated in *tuning maps* (see, e.g., Åström & Hägglund, 1995). The following is a hand-tuning procedure adapted from Smith (1979).

Procedure Hand-tuning.

(a) Remove all integral and derivative action by setting $T_d = 0$ and $1/T_i = 0$.
(b) Tune the proportional gain K_p to give the desired response, ignoring any final value offset from the setpoint.
(c) Increase the proportional gain further and adjust the derivative gain T_d to dampen the overshoot.
(d) Adjust the integral gain $1/T_i$ to remove any final value offset.
(e) Repeat until the proportional gain K_p is as large as possible.

□

The procedure adjusts the derivative gain before the integral gain, but in practice the sequence may be reversed. The advantage of hand-tuning is that a process engineer can use the procedure right away, on-line, and develop a feel for how the closed loop system behaves. A disadvantage is that it may take a long time to develop this feel, and it is difficult to sense whether the final settings are optimal.

Example 2 *In a laboratory water rig the control objective is to adjust the water level in a tank after a step in the reference. The rig consists of a feed pump, a tank, and an outlet.*

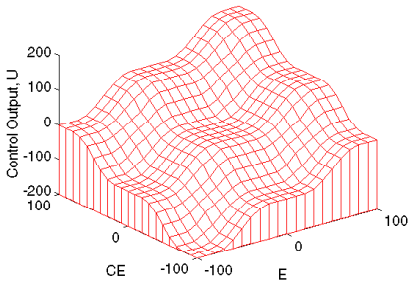

Figure 4: Example of a control surface.

(a) Assume first that the outlet is closed. For this problem a P controller is sufficient. As soon as the water reaches the setpoint level, the error becomes zero, and the controller will stop the feed pump (provided the rule base is sensible).

If there is overshoot, for example if the feed pump reacts sluggishly, the controller should start braking before the water reaches the setpoint. A PD controller is then necessary.

(b) Assume now that the outlet is open. The controller must try and reach the setpoint and keep pumping while the water runs out of the outlet. A sustained control signal in steady state is necessary to balance the outflow. Integral action is necessary and a PI or PID controller will be appropriate.

3. Linear fuzzy controllers

The second step in the design procedure is to replace the summation in PID control by a linear fuzzy controller acting like a *summation*. The closed loop system should thus show exactly the same step response; this is a check that the implementation is correct.

Control Surface With two inputs and one output the input-output mapping is a surface. Figure 4 is a mesh plot of an example relationship between *error* E and *change in error* CE on the input side, and controller output u on the output side. The plot results from a rule base with nine rules, and the surface is more or less bumpy. The horizontal plateaus are due to flat peaks on the input sets. The plateau around the origin implies a low sensitivity towards changes in either *error* or *change in error* near the reference. This is an advantage if noise sensitivity must be low when the process is near the reference. On the other hand, if it is difficult to keep the process on the reference, as with an inverted pendulum, it is necessary to have a larger gain around the origin.

There are three sources of nonlinearity in a fuzzy controller.

- *The rule base.* The position, shape and number of fuzzy sets as well as nonlinear input scaling cause nonlinear transformations. The rules often express a nonlinear control

strategy.

- *The inference engine.* If the connectives **and** and **or** are implemented as for example **min** and **max** respectively, they are nonlinear.
- *The defuzzification.* Several defuzzification methods are nonlinear.

It is possible to construct a rule base with a linear input-output mapping that acts like a summation (Siler & Ying, 1989; Mizumoto, 1992; Qiao & Mizumoto; 1996; Jantzen, 1998).

Input universes The input universes must be large enough for the inputs to stay within the limits (no *saturation*). Each input family should contain a number of terms, designed such that the sum of membership values for each input is 1. This can be achieved when the sets are triangular and cross their neighbour sets at the membership value $\mu = 0.5$; their peaks will thus be equidistant. Any input value can thus be a member of at most two sets, and its membership of each is a linear function of the input value.

Number of rules The number of terms in each family determines the number of rules, as they must be the **and** combination (*outer product*) of all terms to ensure completeness. The output sets should preferably be singletons s_i equal to the sum of the peak positions of the input sets. The output sets may also be triangles, symmetric about their peaks, but singletons make defuzzification simpler.

Connective To ensure linearity, we must choose the algebraic product for the connective **and**. Using the weighted average of rule contributions for the control signal (corresponding to *centre of gravity* defuzzification, *COG*), the denominator vanishes, because all firing strengths add up to 1.

What has been said can be generalised to input families with more than two input sets per input, because only two input sets will be active at a time. The following checklist summarises the general design choices for achieving a fuzzy rule base equivalent to a summation:

- □ Use triangular input sets that cross at $\mu = 0.5$;
- □ use the algebraic product (*) for the **and** connective;
- □ the rule base must be the outer **and** product of all input families;
- □ use output singletons, positions determined by the sum of the peak positions of the input sets;
- □ use *COG* defuzzification.

With these design choices the control surface (Fig. 4) degenerates to a diagonal plane. A flexible fuzzy controller, that allows these choices, is two controllers in one so to speak. When linear, it has a transfer function and the usual methods regarding tuning and stability of the closed loop system apply.

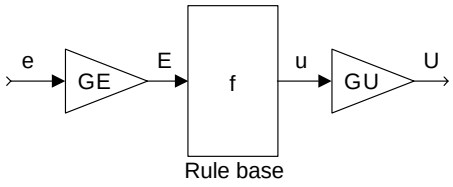

Figure 5: Fuzzy proportional controller (FP).

4. Transferring gains from PID to fuzzy

The third step in the design procedure is to transfer the PID gains to the linear fuzzy controller.

Proportional control Input to a *fuzzy proportional* (FP) controller is *error,* and the output is the control signal, cf. the block diagram in Fig. 5. This is the simplest fuzzy controller there is. It is relevant for state- or output-feedback in a state space controller. Compared to crisp proportional control the fuzzy P controller has two gains GE and GU instead of just one. As a convention, signals are written in lower case before gains and upper case after gains, for instance $E = GE * e$. The gains are mainly for tuning the response, but since there are two gains, they can also be used for scaling the input signal onto the input universe to exploit it better.

The controller output is the control signal U_n, a nonlinear function of e_n,

$$U_n = f(GE * e_n) * GU \tag{5}$$

The function f is the fuzzy input-output map of the fuzzy controller. Using the linear approximation $f(GE * e_n) = GE * e_n$, then

$$U_n = GE * e_n * GU = GE * GU * e_n \tag{6}$$

Compared with (3) the product of the gain factors is equivalent to the proportional gain, i.e.,

$$GE * GU = K_p \tag{7}$$

The accuracy of the approximation depends mostly on the membership functions and the rules. The approximation is best, however, if we choose the same universe on both input and output side, for example $[-100, 100]$. The rule base

1. If E is Pos then u is 100
2. If E is Neg then u is -100

with Pos and Neg triangular as defined previously, is equivalent to a P-controller. Given a target K_p, for example from the Ziegler-Nichols rules, equation (7) helps to choose the gains. The equation has one degree of freedom, since the fuzzy P controller has one more gain factor than the crisp P controller. This is used to exploit the full range of the input universe. If for example the maximal reference step is 1, whereby the maximal error e_n is 1, and the universe for E is $[-100, 100]$, then fix GE at 100. Since GE is now fixed, GU

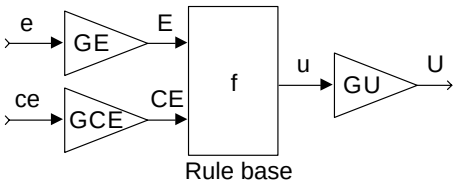

Figure 6: Fuzzy PD controller (FPD).

is determined by (7).

Because of the process dynamics it will take some time before a change in the control signal is noticeable in the process output, and the proportional controller will be more or less late in correcting for an error.

Proportional and derivative control Derivative action helps to predict the error and the proportional-derivative controller uses the derivative action to improve closed-loop stability. The basic structure of a PD controller is

$$u_n = K_p \left(e_n + T_d \frac{e_n - e_{n-1}}{T_s} \right) \tag{8}$$

The control signal is thus proportional to an estimate of the error T_d seconds ahead, where the estimate is obtained by linear extrapolation. For $T_d = 0$ the control is purely proportional, and when T_d is gradually increased, it will dampen oscillations. If T_d becomes too large the system becomes *overdamped* and it will start to oscillate again.

Input to the *fuzzy proportional-derivative* (FPD) controller is the *error* and the *derivative of the error* (Fig. 6). In fuzzy control the latter term is usually called *change in error*,

$$ce_n = \frac{e_n - e_{n-1}}{T_s} \tag{9}$$

This is a discrete approximation to the differential quotient using a backward difference. Other approximations are possible, as in crisp PD controllers. Notice that this definition deviates from the straight difference $ce_n = e_n - e_{n-1}$ used in the early fuzzy controllers.

The controller output is a nonlinear function of *error* and *change in error*,

$$U_n = f(GE * e_n, GCE * ce_n) * GU \tag{10}$$

Again the function f is the input-output map of the fuzzy controller, only this time it is a surface. Using the linear approximation $GE * e_n + GCE * ce_n$, then

$$U_n = (GE * e_n + GCE * ce_n) * GU \tag{11}$$

$$= GE * GU * (e_n + \frac{GCE}{GE} ce_n) \tag{12}$$

By comparison, the gains in (8) and (11) are related in the following way,

$$GE * GU = K_p \tag{13}$$

$$\frac{GCE}{GE} = T_d \tag{14}$$

The approximation corresponds to replacing the fuzzy input-output surface with a plane. The approximation is best if we choose the output universe to be the sum of the input universes. Assume, for example, that the input universes are both $[-100, 100]$ and we choose output singletons on $[-200, 200]$, then the input-output map will be the plane $u = E + CE$. Therefore, by that choice, we can exploit (13) and (14).

The fuzzy PD controller may be applied when proportional control is inadequate. The derivative term reduces overshoot, but it may be sensitive to noise as well as an abrupt change of the reference causing a *derivative kick*. The usual counter-measures may overcome these problems: in the former case insert a filter, and in the latter use the derivative of the process output y_n instead of the error.

Incremental control If there is a sustained error in steady state, integral action is necessary. The integral action will increase the control signal if there is a small positive error, no matter how small the error is; the integral action will decrease it if the error is negative. A controller with integral action will always return to zero in steady state.

It is possible to obtain a fuzzy PI controller using error and change in error as inputs to the rule base. Experience shows, however, that it is rather difficult to write rules for the integral action. Problems with *integrator windup* also have to be dealt with. Windup occurs when the actuator has limits, such as maximum speed for a motor or maximum opening of a valve. When the actuator saturates, the control action stays constant, but the error will continue to be integrated, the integrator winds up. The integral term may become very large and it will then take a long time to wind it down when the error changes sign. Large overshoots may be the consequence. There are methods to avoid it (Åström & Hägglund, 1995).

It is often a better solution to configure the controller as an incremental controller. An incremental controller adds a *change* in control signal Δu to the current control signal,

$$u_n = u_{n-1} + \Delta u_n \Rightarrow \tag{15}$$

$$\Delta u_n = K_p \left(e_n - e_{n-1} + \frac{1}{T_i} e_n T_s \right) \tag{16}$$

using (2) with $T_d = 0$. It is natural to use an incremental controller when for example a stepper motor is the actuator. The controller output is an increment to the control signal, and the motor itself performs an integration. It is an advantage that the controller output is driven directly from an integrator, then it is easy to deal with windup and noise. A disadvantage is that it cannot include D-action well.

The *fuzzy incremental* (FInc) controller in Fig. **??** is almost the same configuration as the FPD controller except for the integrator on the output The output from the rule base is therefore called *change in output* (cu_n) and the gain on the output has changed name accordingly to GCU. The control signal U_n is the sum of all previous increments,

$$U_n = \sum_i (cu_i * GCU * T_s) \tag{17}$$

Notice again that this definition deviates from the early fuzzy controllers, where $U_n =$

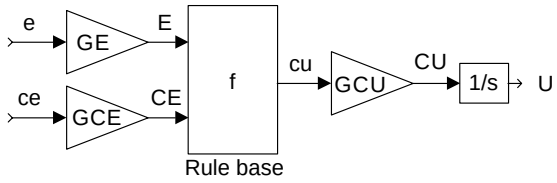

Figure 7: Fuzzy incremental controller (FInc).

$\Sigma(GCU * cu_i)-$ the difference is the sampling period T_s. The linear approximation to this controller is

$$\begin{aligned} U_n &= \sum_{i=1}^{n}(E_i + CE_i) * GCU * T_s \\ &= GCU * \sum_{i=1}^{n}\left[GE * e_i + GCE * \frac{e_i - e_{i-1}}{T_s}\right] * T_s \\ &= GCU * \left[GE * \sum_{i=1}^{n} e_i * T_s + GCE * \sum_{i=1}^{n}(e_i - e_{i-1})\right] \\ &= GCE * GCU * \left[\frac{GE}{GCE}\sum_{i=1}^{n} e_i * T_s + e_n\right] \end{aligned} \tag{18}$$

By comparing (8) and (18) it is clear that the gains are related in the following way,

$$\begin{aligned} GCE * GCU &= K_p \\ \frac{GE}{GCE} &= \frac{1}{T_i} \end{aligned} \tag{19}$$

Notice that the proportional gain now depends on GCE. The gain on the integral action is determined by the ratio between the two input gains, and it is the inverse of the derivative gain in FPD control. It is as if GE and GCE have changed roles.

The controller is really a PI controller, compare (2) and (18). The usual problem of *integrator windup* can be overcome by simply limiting the integrator.

Proportional, integral and derivative control It is straight forward to envision a *fuzzy PID controller* with three input terms: *error*, *integral error*, and *derivative error*. A rule base with three inputs, however, easily becomes rather big and, as mentioned earlier, rules concerning the integral action are troublesome. Therefore it is common to separate the integral action as in the *fuzzy PD+I* (FPD+I) controller in Fig. 8. The integral error is computed as,

$$ie_n = \sum_i (e_i * T_s) \tag{20}$$

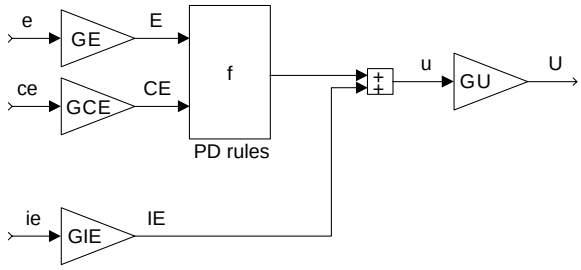

Figure 8: Fuzzy PD+I controller (FPD+I).

The controller is thus a function of the three inputs

$$U_n = [f(GE * e_n, GCE * ce_n) + GIE * ie_n] * GU \tag{21}$$

Its linear approximation is

$$\begin{aligned} U_n &= [GE * e_n + GCE * ce_n + GIE * ie_n] * GU \\ &= GE * GU * \left[e_n + \frac{GCE}{GE} * ce_n + \frac{GIE}{GE} ie_n \right] \end{aligned} \tag{22}$$

In the last line we have assumed a nonzero GE. Comparing (2) and (22) the gains are related in the following way,

$$GE * GU = K_p \tag{23}$$

$$\frac{GCE}{GE} = T_d \tag{24}$$

$$\frac{GIE}{GE} = \frac{1}{T_i} \tag{25}$$

This controller provides all the benefits of PID control, but also the disadvantages regarding derivative kick and integrator windup.

Example 3 *We will try fuzzy PD+I control on the process from Example 1. We set* $GE = 100$ *since, according to the plot in Fig.* 3*, the maximal error is* 1*. By (23)* GU *is now fixed by the relation*

$$GU = K_p/GE = 4.8/100$$

The gain on the change in error is then fixed by (24),

$$GCE = GE * T_d = 100 * 15/32$$

The last gain is, by (25),

$$GIE = GE * 1/T_i = 100 * 8/15$$

The step response with a linear FPD+I is exactly identical to Fig. 3*. A check on the input universes showed that* $|E_{\max}| \leq 100$ *and* $|CE_{\max}| \leq 55-$ *thus there was no saturation.*

Controller	Advantage	Disadvantage
FP	Simple	Maybe too simple
FPD	Less overshoot	Noise sensitive, derivative kick
FInc	Removes steady state error, smooth control signal	Slow
FPD+I	All in one	Windup, derivative kick

Table 3: Quick reference to controllers

Summary Advantages and disadvantages of all four fuzzy controllers are listed in Table 3. The fuzzy P controller may be used in state space models or for practising. To improve the settling time and reduce overshoot, the fuzzy PD is the choice. If there is a problem with a steady state error, a fuzzy incremental controller or a fuzzy PD+I is the choice.

The relationships between the PID gains and the fuzzy gains are summarised in Table 4. To emphasise, the controllers with f replaced by a summation are linear approximations to the corresponding fuzzy configurations; the relations hold for the *approximations* only. They are valid when *ce* and *cu* are true difference *quotients* instead of just differences. If these are implemented as differences anyway, ($ce_n = \Delta e_n = e_n - e_{n-1}$ and $cu_n = \Delta u_n = u_n - u_{n-1}$), the sample period must be taken into account in the equations, and the table modified accordingly. Also, for fixed universe controllers, the output universe must be the sum of the input universes. With input universes $[-100, 100]$, for instance, the output universe of an FPD controller should be $[-200, 200]$.

Example 4 *What if we come across other controller implementations than the above; what are the approximations then?*

(a) In a fuzzy PD controller ce is implemented as a difference Δe. Comparing (2) and (11) implies $(GCE/GE) * \Delta e = T_d * \Delta e/T_s$, and this implies $T_d = (GCE/GE) * T_s$. Similarly with the FPD+I controller. Then the last column in Table 4 should be multiplied by T_s. As a consequence an increase in the sampling period will increase the differential time.

(b) In a fuzzy incremental controller $ce = \Delta e/T_s$, but $U_n = \sum u_i * GCU$ (without the multiplication by T_s). Then (18) becomes

$$U_n = \frac{GCE}{T_s} * GCU * \left[\frac{GE}{GCE} \sum_{i=1}^{n} e_i * T_s + e_n \right]$$

The comparison with (2) yields $K_p = GCE * GCU/T_s$, and the integral time is unchanged. As a consequence an increasing sampling period implies a decreasing proportional gain.

(c) A fuzzy PD has the output universe $[-100, 100]$. With a maximum value of 100 on both inputs E and CE, the output is at most 100. It is equivalent to the usual linear approximation with half the output gain. Thus Table 4 must be used with $GU/2$ instead of GU. The general rule is to use the table with GU/r (or GCU/r) where r is the number of inputs when the output universe equals the input universes.

Controller	K_p	$1/T_i$	T_d
FP	$GE * GU$		
FInc	$GCE * GCU$	GE/GCE	
FPD	$GE * GU$		GCE/GE
FPD+I	$GE * GU$	GIE/GE	GCE/GE

Table 4: Relationship between fuzzy and PID gains

Saturation, quantisation, and noise A few practical considerations are necessary at this place. It is important to be aware whether the input signals saturate in their universes. Take for example the third order process $1/(s+1)^3$ from Example 1. A suitable value of the gain on *error* is $GE = 100$. For the sake of illustration Fig. 9 shows what happens if the error signal saturates – the gain is now $GE = 400$, and all other gains are according to Table 4 whereby the proportional gain, differential time and integral time remain the same. The error signal saturates early in the transient response as the phase plot clearly shows (Fig. 10). The result is a larger overshoot, a slower response, and an irregular error signal.

If the input universes in a controller are discrete, it is always possible to calculate all thinkable combinations of inputs before putting the controller into operation. In a *table-based controller* the relation between all input combinations and their corresponding outputs are arranged in a look-up table. The table implementation improves execution speed, as the run-time inference is reduced to a table look-up which is faster, at least when the correct entry can be found without too much searching. In the table-based controller without interpolation the quantisation in the table affects the performance.

With $GE = 100$ and a quantum size of 20 in the input universes corresponding to a resolution of 10% of full range, Fig. 11 shows the effect. The overshoot is the same, but there are now *limit cycles*, i.e., stable oscillations in steady state. The controller allows the process to drift within a cell in the look-up table until it shifts to another cell with another control action. This is especially noticeable in steady state. The amplitude of the limit cycle is affected by the input gains, and with $GE = 400$ the limit cycle diminishes (Fig. 12), while the overshoot increases. There are three ways to reduce the limit cycle: 1) to increase GE, 2) to make the discretisation finer, and 3) to use interpolation. The first option makes the controller more sensitive to small deviations from the setpoint, and may also cause saturation that changes the dynamics. A variant of the second option is to use a new table with a finer resolution when the process gets near the setpoint. The third option gets rid of the quantisation effect all together, but the second is a good alternative if interpolation is too time consuming and if computer memory space permits.

Measurement noise also affects the performance. With a linear control table and $GE = 100$ as in the original example, Fig. 13 shows the response with noise added. The control signal is drastically affected. The system is still stable, but it shows more oscillation with noise. The initial overshoot is also slightly higher.

If the noise is bad enough it will drive the phase plot to the edges of the *change in error* universe; beyond that the input universe will limit how bad it gets. If the noise causes instability or disturbs the control, a filter may be necessary.

Another option is to use the fuzzy incremental controller. The integrator in the out-

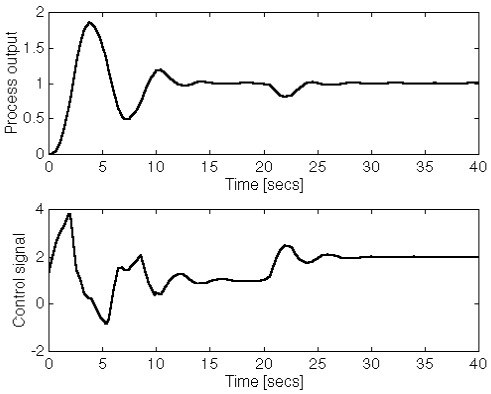

Figure 9: FPD+I control of $1/(1+s)^3$ when $GE = 400$ causing saturation.

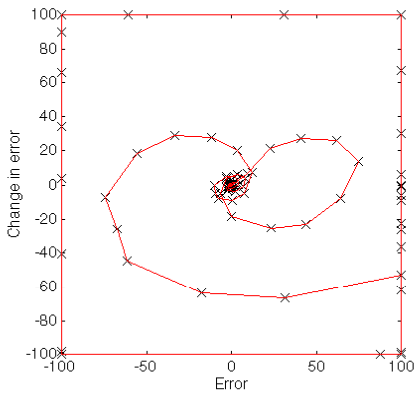

Figure 10: Phase plot showing saturation along the edges of the look-up table.

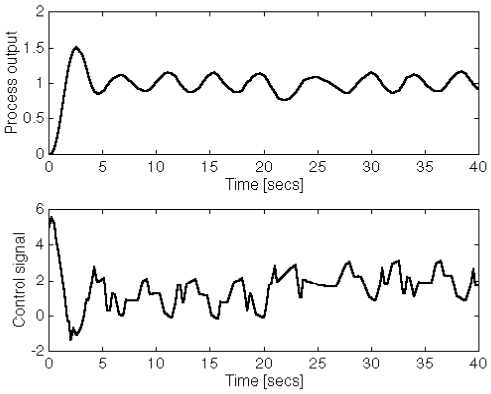

Figure 11: FPD+I control of $1/(1+s)^3$ with quantisation.

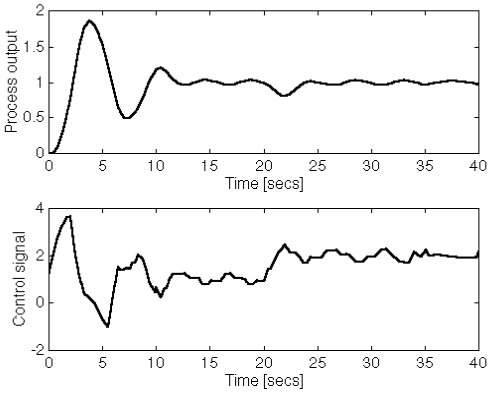

Figure 12: FPD+I control of $1/(1+s)^3$ with quantisation and $GE = 400$.

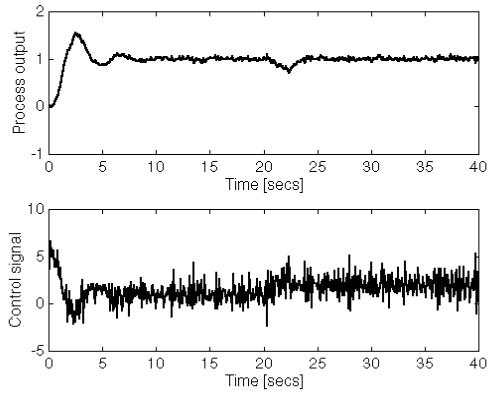

Figure 13: FPD+I control of $1/(1+s)^3$ with noise.

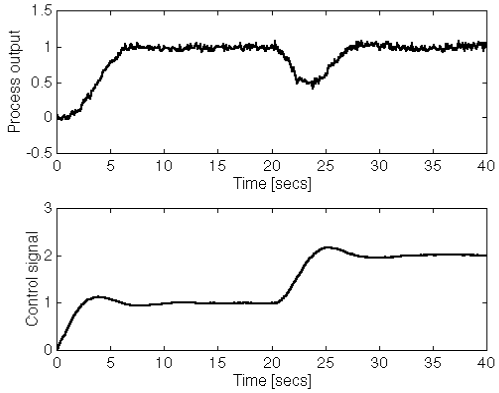

Figure 14: FInc control of $1/(1+s)^3$ with noise.

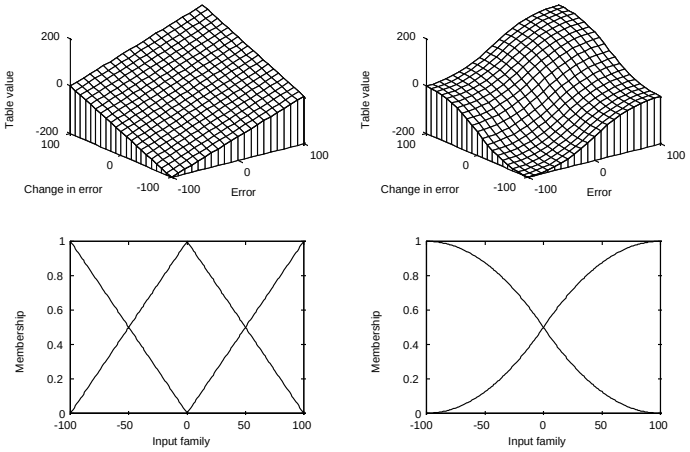

Figure 15: Linear surface (top left) and steep surface (top right) with the input families that generated them (bottom left and right respectively).

put end smooths the control signal, but it performs somewhat differently. With an FInc controller the Ziegler-Nichols rules give poor results, but after hand-tuning to $GE = 100$, $GCE = 200$, $GCU = 0.005$ the response is rather good (Fig. 14). The rise time is slower than FPD+I, and the dip from the load change is larger, but it has less overshoot. Noise makes the system less stable, but the noise in the control signal is dampened compared to FPD+I control.

The similarity with the PID controller implies some traditional PID problems. The *change in error* is sensitive to noise, although a finite universe may limit spikes and outliers. Often some sort of filter will be necessary. The *integral error* may cause integrator windup when the control signal U is limited by the actuator equipment. It is possible, though, to overcome these problems with the known PID techniques (see e.g., Åström & Hägglund, 1995).

5. Making the fuzzy controller nonlinear

The fourth step in the design procedure is to gradually make the linear fuzzy controller nonlinear. It is common practice to build a rule base from *terms* such as Pos, Zero, and Neg, representing *labels* of fuzzy sets. An *input family* may consist of those three terms. Consequently, with two inputs it is possible to build $3 \times 3 = 9$ rules. Nine rules is a manageable amount often used in practice.

The shape of the sets and the choice of rules affect the control strategy and the dynamics of the closed loop system. There are essentially four characteristic shapes of the control surface.

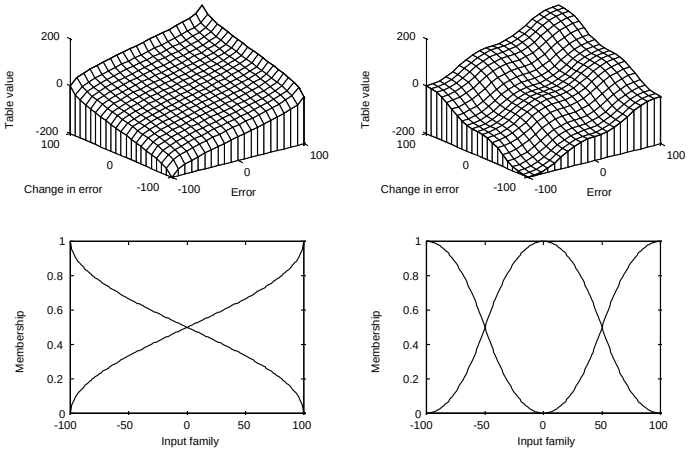

Figure 16: Gently sloping surface (top left) and bumpy surface (top right) with the input families that generated them (bottom left and right respectively).

- A *linear* surface (Fig. 15, left) results from the rules below with triangular input sets,

  1. If *error* is Neg and *change in error* is Neg then output is -200
  2. If *error* is Neg and *change in error* is Zero then output is -100
  3. If *error* is Neg and *change in error* is Pos then output is 0
  4. If *error* is Zero and *change in error* is Neg then output is -100
  5. If *error* is Zero and *change in error* is Zero then output is 0 (26)
  6. If *error* is Zero and *change in error* is Pos then output is 100
  7. If *error* is Pos and *change in error* is Neg then output is 0
  8. If *error* is Pos and *change in error* is Zero then output is 100
  9. If *error* is Pos and *change in error* is Pos then output is 200

  The surface in the figure is actually equivalent to a summation of the two inputs (cp. the values on the axes). Since the surface is equivalent to a summation, the controller is equivalent to a PD controller.
- The *steep* surface (Fig. 15, right) is built using only rules 1, 3, 7, and 9 together with the input sets shown in the figure; they are segments of cosine functions. Notice the absence of the centre rule (no. 5) with zero *error* and zero *change in error*. That surface has a steeper slope, or higher gain, near the centre of the table than the linear surface has, but they have the same values pairwise in the four corners.
- The *gentle* surface (Fig. 16, left) is built using the same four rules 1, 3, 7, and 9 but the input sets have been reflected. They are based on inverse trigonometric functions, i.e., $Pos(x) = 0.5 + \sin^{-1}(x/100)/\pi$, and $Neg(x) = \cos^{-1}(x/100)/\pi$. That surface has a more gentle slope, or lower gain, near the centre of the table than the linear surface has; but it too has the same values pairwise in the four corners.

- The *bumpy* surface (Fig. 16, right) is a blend of the previous two surfaces. It is built using the set of nine rules (26) with nonlinear input sets as shown in the figure. This is often the default. It has a flat plateau near the centre and bumps in several other places. Even this surface has the same values as the other surfaces in the four corners.

It is difficult to make a stringent, fair and objective comparison of their control characteristics. For a study of how the response reacts to the various nonlinear surfaces, see Jantzen (1997).

6. Fine-tuning the nonlinear fuzzy controller

The fifth step in the design procedure is to fine-tune the gains, now the fuzzy controller is nonlinear. The choice of gains in the fine-tuning phase is traditionally the result of intuition and experience. A few rules of thumb can be derived from the linear approximations:

- *The sample period.* If the sample period is too short, the computation of ce_n will become too sensitive to noise. This normally shows up as a restless control signal. If ce_n happens to be implemented as a difference several gains depend on the sample period; a change in sample period must then be followed by a compensation in a gain factor to keep the proportional, integral and derivative gains intact.
- *GE.* If the controller is supposed to use the whole range of its universe, then the maximal E should equal the limit of the universe, that is

$$|e_{\max} * GE| = |Universe_{\max}| \tag{27}$$

  With a unit reference step and the universe $[-100, 100]$ the equation implies $GE = 100$. If GE is too big, the incremental controller will become less stable, because the integral gain is too high. In an FPD controller, GE affects the proportional gain and the derivative gain. One would like to have GE as large as possible to reduce noise problems and still have a large proportional gain. In the FPD+I controller, one would still prefer GE as large as possible to promote the proportional gain at the expense of the integral gain and the derivative gain.
- *GCE.* A similar argument as above applies if *change in error* is supposed to use the whole range of its universe,

$$|ce_{\max} * GCE| = |Universe_{\max}| \tag{28}$$

  In an FPD controller, a larger GCE means a larger derivative gain with no effect on the proportional gain. To keep noise problems to a minimum, one will therefore try and keep GCE as small as possible. In the FInc controller an increase in GCE will decrease the integral gain and increase the proportional gain; thus one would like to keep GCE as large as possible to preserve stability. In the FPD+I controller, an increase in GCE will increase the derivative gain, so one would keep it as small as possible.
- *GCU or GU.* These affect the proportional gain, so one would like to have them as large as possible without creating too much overshoot. If too small, the system will be too slow, and if too large the system might become unstable.

A procedure for hand-tuning an FPD+I controller is the following (with trivial modifications this procedure covers the FPD and FInc controllers as well):

1. Adjust GE according to stepsize and universe to exploit the range of the universe fully.
2. Remove integral action and derivative action by setting $GIE = GCE = 0$. Tune GU to give the desired response, ignoring any final value offset.
3. Increase the proportional gain by means of GU, and adjust the derivative gain by means of GCE to dampen the overshoot.
4. Adjust the integral gain by means of GIE to remove any final value offset.
5. Repeat the whole procedure until GU is as large as possible.

It seems plausible that the stability margins will be close in some sense to the linear approximation. In simulation, at least, it is possible to experiment with different controller surfaces and get a rough idea of the gain margin and the sensitivity to dead times. As with all nonlinear systems, however, the responses are amplitude dependent and thereby depend on the step size.

7. Summary and conclusions

In summary, we can form a tuning procedure for fuzzy controllers for a step in the setpoint. The following refers to an FPD+I controller, because it is the most general controller, but it covers the other controllers as well with straight forward modifications.

1. Insert a crisp PID controller, and tune it (use Ziegler-Nichols, Kappa-Tau, optimisation, hand-tuning, or another method, cf. e.g. Åström & Hägglund, 1995).
2. Insert a *linear* FPD+I.
3. Transfer K_p, T_d and $1/T_i$ to GE, GCE, GIE and GU using Table 4. If it does not saturate in the universes, the response should be exactly the same.
4. Insert a nonlinear rule base.
5. Fine-tune using hand-tuning; use GE to improve the rise time, GCE to dampen overshoot, and GIE to remove any steady state error.

The FPD+I controller has one degree of freedom, since it has one more gain factor than the crisp PID. This is used to exploit the full range of one input universe. If, for example, the reference step is 1 and the universe for E is $[-100, 100]$, then fix GE at 100 in order to exploit the full range. The free variable should be GE or GCE, whichever signal has the largest magnitude after multiplication by its gain factor.

The performance depends on the control table. With a linear control table the fuzzy PID controller can be made to perform exactly like a crisp PID controller. Sometimes a nonlinear control table can be made to perform better than conventional PID control, but it depends on the process, and one has to be careful to select the right kind of nonlinear control table.

Since a fuzzy PID controller contains a crisp PID controller as a special case, it is true to say that it performs at least as well. It is comforting in process control systems to start

in the PID domain and gradually make it fuzzy.

References

Åström, K., Hang, C., Persson, P. and Ho, W. (1992). Towards intelligent PID control, *Automatica* **28**(1): 1–9.

Åström, K. J. and Hägglund, T. (1995). *PID Controllers - Theory, Design, and Tuning*, second edn, Instrument Society of America, 67 Alexander Drive, PO Box 12277, Research Triangle Park, North Carolina 27709, USA.

Åström, K. J. and Wittenmark, B. (1984). *Computer controlled systems - theory and design*, Prentice-Hall, Englewood Cliffs.

Jantzen, J. (1997). A robustness study of fuzzy control rules, *in* EUFIT (ed.), *Proceedings Fifth European Congress on Fuzzy and Intelligent Technologies*, ELITE Foundation, Promenade 9, D-52076 Aachen, pp. 1222–1227.

Jantzen, J. (1998). Design of fuzzy controllers, *online 98-E-864 (design)*, Technical University of Denmark: Dept. of Automation, http://www.iau.dtu.dk/˜ jj/pubs. Lecture notes, 27 p.

Mizumoto, M. (1992). Realization of PID controls by fuzzy control methods, *in* IEEE (ed.), *First Int. Conf. On Fuzzy Systems*, number 92CH3073-4, The Institute of Electrical and Electronics Engineers, Inc, San Diego, pp. 709–715.

Qiao, W. and Mizumoto, M. (1996). PID type fuzzy controller and parameters adaptive method, *Fuzzy Sets and Systems* **78**: 23–35.

Siler, W. and Ying, H. (1989). Fuzzy control theory: The linear case, *Fuzzy Sets and Systems* **33**: 275–290.

Smith, L. C. (1979). Fundamentals of control theory, *Chemical Engineering* **86**(22): 11–39. (Deskbook issue).

Tso, S. K. and Fung, Y. H. (1997). Methodological development of fuzzy-logic controllers from multivariable linear control, *IEEE Trans. Systems, Man & Cybernetics* **27**(3): 566–572.