An approach to the design of a fuzzy self-tuning PID controller

An approach to the design of a fuzzy self-tuning PID controller

This paper presents an approach to the design of a self-tuning PID controller using a fuzzy adaptive mechanism. The main idea is to tune the controller gains on-line by a fuzzy adaptive mechanism, i.e. the controller gains are subjected to a single parameter that results from a fuzzy inference mechanism. The controller is simulated with a 2-DOF robot and simulations show a considerable improvement in performance, in terms of setpoint responses, as compared to conventional well-tuned PID controllers.

M.A.D. BAYENS
Eindhoven University of Technology, Faculty of Mechanical Engineering, Section Dynamics, Eindhoven, The Netherlands

M.J.G. VAN DE MOLENGRAFT
Eindhoven University of Technology, Faculty of Mechanical Engineering, Section Systems and Control, Eindhoven, The Netherlands

Introduction

Most real processes that require automatic control are nonlinear in nature. That is, the system matrices (mass, dampings and stiffness matrix) vary as the operating point changes. A linear controller can only be tuned to give good performance at one particular operating point or for a limited period of time. The controller needs to be retuned if the operating point changes, or retuned periodically if the process is time-varying. This necessity to retune has increased the need for adaptive controllers that can automatically retune themselves to match the current process characteristics.

Despite this, the majority of industrial processes nowadays are still regulated by conventional PID controllers mainly because of their simplicity. For example in the case of a robot arm that has to move objects with different masses along a predefined path, good and exact models are available so it is not too big a problem to create a conventional well-tuned PID controller. However, substantial changes in the system matrices due to non-linearities, or major external disturbances, lead to a sharp decrease in performance. In the presence of such disturbances, PID systems are usually faced with a trade-off between fast reactions with significant overshoot or smooth but slow reactions.

In order to improve the performance of such systems a fuzzy self-tuning PID controller was introduced. The central idea of this controller was first introduced in [1]. The main contribution of this paper involves the extension of the control scheme in [1] for application in nonlinear mechanical systems such as a robot arm. This extension required several changes, such as the adaptation of the rule-base for mechanical systems; however, the basic idea of [1] remains unaltered.

In [1] the controller evaluates the trend of the controlled process output at every timestep, to detect a possible deviation from the prescribed course. If a deviation is found, an appropriate control action according to the nature of the deviation will be generated instantaneously by the fuzzy mechanism to adapt the PID parameters, if necessary.

There are two basic ideas. Firstly: the three controller gains are subjected to a single parameter α. These dependencies are arranged so that an increase(decrease) in α will lead to a decrease(increase) in the differential term and an increase(decrease) in both the proportional and integral terms. Secondly: in order to use the qualitative relationship between the controller gains and the resulting process output, the parameter α is updated on-line by a simple tuning formula driven by a fuzzy inference mechanism. This fuzzy mechanism updates α taking into account the current process state. This adjustment of α will in turn lead to a change in the controller gains resulting in a faster convergence of the process output to a set-point, and a slower divergence away from the set-point. As stated before, this article is concerned with the implementation of this basic idea in the control of a robot arm.

One of the differences between this implementation and the one proposed in [1] is that in [1] the form of the parameterisations is inspired by the Ziegler-Nichols tuning formula, whereas in this simulation the chosen parameterisations of the controller gains are derived empirically. This has the following reason: in the case of the robot arm the set-point responses keep improving if larger control actions are allowed, in other words if a larger P-parameter is used associated with a larger necessary damping parameter D. So it is impossible to specify one optimal combination of the three controller gains because it depends on the maximum allowed control action.

The combination of the three PID parameters must be such that the maximum performed control action does not exceed this maximum allowed control action. The advantage of the fuzzy self-tuning controller over the standard PID controller is the ability to alter the three PID parameters on-line with the maximum allowed control action as only restriction.

The controller and the fuzzy adaption mechanism

Basic Design

To understand the basic structure of the fuzzy self-tuning PID controller which has been developed with the help of [1], the block diagram is drawn in *Figure 1*. As we can see in fig. 1 the fuzzy self-tuning PID controller consists of a fuzzy self-tuning mechanism that computes the parameter α, depending on the error and error derivative, which in turn will be the input of the adaptive part of the controller.

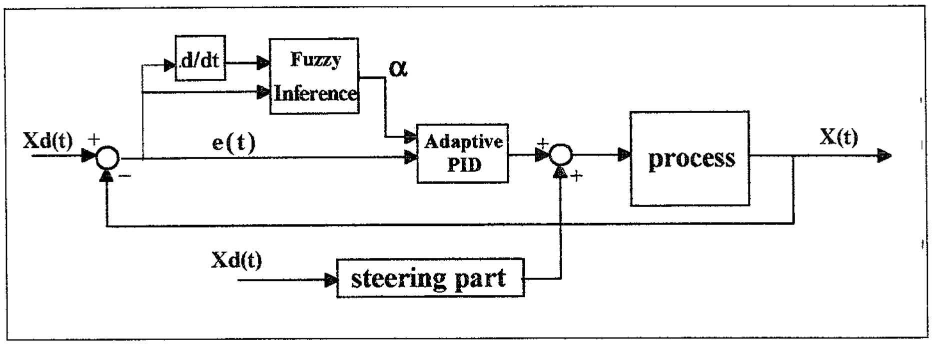


Fig. 1 : Scheme of the fuzzy self-tuning PID controller.

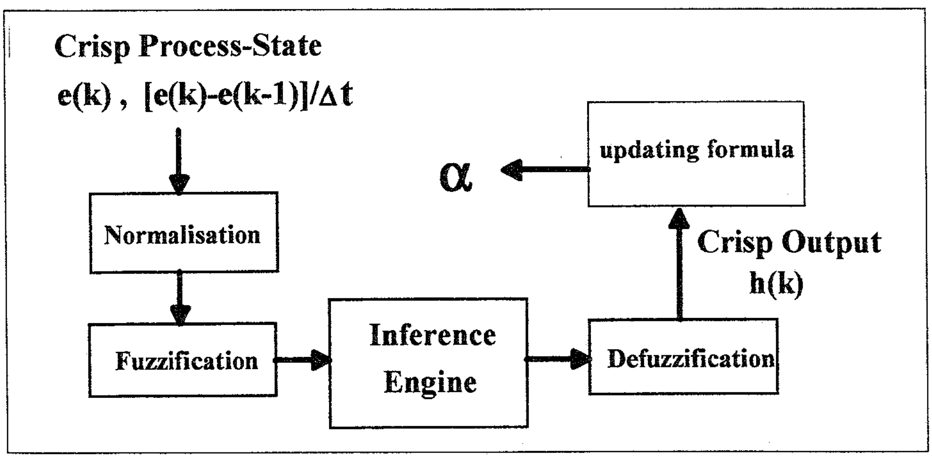


Fig. 2 : The structure of the fuzzy inference mechanism.

The inverse-dynamics based steering part compensates for inertia and non-linear effects in the system. Although a real computed torque feedback term could yield better performances, the proposed scheme provided sufficient control for the non-linear 2DOF robot arm in our case. Morever stability can be negatively affected by a computed torque feedback term.

Adaptive Part

A conventional PID controller generates a control action u(k), based on the error $e(k) = X_d - X(k)$ in the following standard form :

$$u(k) = K_P e(k) + \frac{K_D}{\Delta t}\,[e(k) - e(k-1)] + K_I \sum_{i=1}^{k} [e(i)],$$

$$t_k = t_0 + k.\Delta t \qquad (1)$$

Where K_P, K_D and, K_I are the proportional, derivative and integral gain respectively and Δt is the discrete sampling time.

In the case of the fuzzy self-tuning PID controller these gains are all parameterised by a single parameter α. With this parameterisation we want to achieve a faster convergence to the set-point in cases of large deviations and a slower convergence when approaching the set-point in order to prevent or minimise a possible overshoot. So in the first case we need a larger P-action and a smaller D-action whereas in the second case we need a larger D-action and a smaller P-action. The basic form of the parameterisation is chosen in such a way that an increasing (decreasing) value of α causes an increase (decrease) in both K_P and K_I and a decrease (increase) in K_D. So in the first case α must be increased and in the second case decreased.

An appropriate parameterisation could be the following:

$$K_P = \alpha^2 K_{P0} \quad , 0 < \alpha < 1 \qquad (2)$$

$$K_D = 1/(1+\alpha)\, K_{D0} \qquad (3)$$

$$K_I = \alpha K_{I0} \qquad (4)$$

In these formulas the constants K_{P0}, K_{D0} and K_{I0} are used to adjust the average level of the PID parameters (K_P, K_D and K_I) during a simulation. So they can be used to influence the average level of the control action. Their values are set at the beginning of the process or simulation. The parameterisation of the proportional gain is quadratic in α because various simulations have shown that this improved the performance as compared to a linear relationship. It must be stressed here that the precise form of the above parameterisations is not unique. So many other parameterisations can be used as long as the basic forms correspond with the basic form stated above.

Fuzzy Part

Because the knowledge of the relationship between the proportions of the PID feedbacks and the profiles of the process outputs is mostly qualitative, fuzzy logic provides an opportunity to use this linguistic information in a controller.

As we can see in *figure 2* the fuzzy scheme consists of four modules and an additional module for the updating of α. In the normalisation module the crisp input values are normalised to a standardised domain by linear scaling. The fuzzification module is concerned with transforming the normalised crisp inputs to membership values corresponding to linguistic states. An inference module working with predetermined rules calculates membership values for the linguistic output state H. These rules determine the behaviour of the controller in the closed loop system.

The transformation from the linguistic state (H) to the crip output parameter (h(t)) is done by the defuzzification module. Finally, this parameter is used for updating α in the updating module. Each of the modules will now be looked at more closely in order to explain the fuzzy part of the fuzzy self-tuning controller in detail.

The normalisation module maps the actual physical domain of the input variables on a domain of real numbers between -6 and 6. This domain is a frequently used standard domain in literature [2]. In this way the same domain can be used for all fuzzy variables. As we can see in fig. 2 the input variables are e(k) and [e(k)-e(k-1)] /Δt.

The fuzzification module then transforms the measured variables to linguistic states. In order to do this each linguistic state is defined by a membership function. With a membership function the connection between a physical state and a linguistic state is specified. The normalised domain consists of seven different fuzzy sets (linguistic states) as shown below,

$$X = \{NL, NM, NS, ZO, PS, PM, PL\} \quad (5)$$

in which, as usual, NL stands for *negative large*, NM stands for *negative medium* etc. The membership functions $\mu_i(x)$ used to represent these sets are all bell-shaped and have the following mathematical formula:

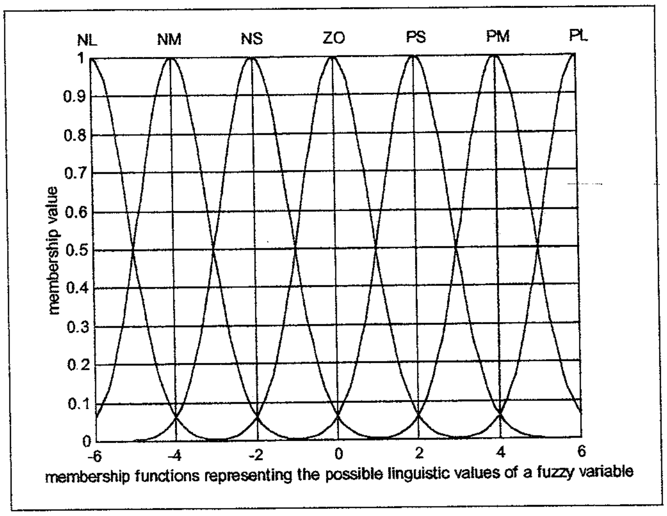


Fig. 3: Membership functions defining fuzzy sets.

H (error \ error derivative)	NL	NM	NS	ZO	PS	PM	PL
NL	NL	NL	NM	PL	PM	PS	ZO
NM	NL	NM	NS	PM	PM	ZO	NS
NS	NM	NM	NS	PS	ZO	NS	NS
ZO	NM	NS	NS	ZO	NS	NS	NM
PS	NS	NS	ZO	PS	NS	NM	NM
PM	NS	ZO	PM	PM	NS	NM	NL
PL	ZO	PS	PM	PL	NM	NL	NL

Fig. 4: Fuzzy map showing the fuzzy rules.

$$\mu_i(x) = e^{-\left(\frac{x-x_i}{\sigma}\right)^2} \quad (6)$$

with: x the crisp input value
x_i a constant that determines the centre of membership function i
σ a constant that determines the width of the membership functions

All seven fuzzy states with the corresponding membership functions are shown in *Figure 3.*

The inference engine is the place where the fuzzy rules, derived from linguistic knowledge are applied, thus transforming the linguistic input states to an appropriate linguistic output. The Mamdani implication is used to represent the meaning of the if-then rule, as is usually done in fuzzy control. The type of rule firing is Individual-rule based inference. A possible set of rules which would be used is shown in *figure 4.* The rules are of the following form:

if error is PL *and* error rate is ZO *then* H is PL
if error is PM *and* error rate is NS *then* H is PM
if error is PS *and* error rate is NL *then* H is NS
etc.

Explanation of rulebase:

As stated before the purpose of the new scheme is to speed up convergence in case of large deviations from the

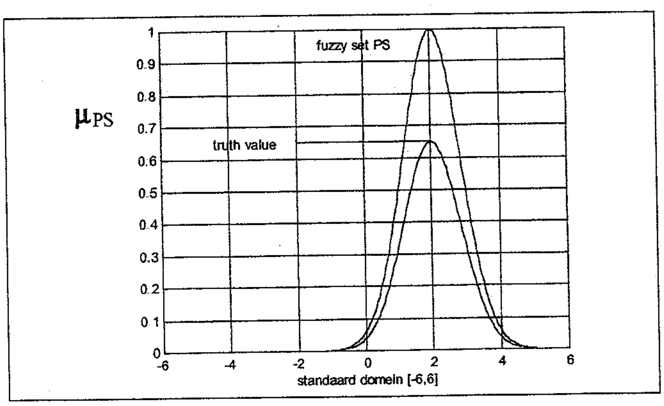


Fig. 5: One scaled fuzzy set (PS).

set-point and to slow down the response when approaching the set-point, by adapting the PID parameters on-line. So keeping in mind the global forms of the chosen parameterisations in the adaptive part, α should be increased in the first case and decreased in the second case. It will become clear in the additional updating module on the next page that the output H (and its corresponding crisp value h(t)) determine the increase in α. So considering the three aforementioned rules we can conclude that these rules indeed force α to a higher level in the first situation and a lower level in the second.

Note that the general rulebase shown above can be slightly adjusted to each specific application which is also done in this case.

The rules include two propositions in the *if* part and one in the *then* part. The *and* operator which combines the two propositions in the antecedent (if part) is calculated as the product. In this way, computing the truth values of the antecedent parts of all forty-nine rules is reduced to a simple outer vector product.

The transformation of linguistic input to linguistic output is done by scaling (multiplying) [2] all seven membership functions defining the linguistic output states with the related membership of the antecedent part. Finally, the overall output H is obtained as the union of the scaled fuzzy sets.

In the defuzzification module the Center-of-Sums defuzzification method [2] is used to transform these scaled fuzzy sets into a crisp output h(t) in order to reduce computation time in comparison with the Centre-of-Area method which is mostly used in literature.

Additional updating module

Following the researchers in [1] we use the formula below to update the parameter α.

$$\alpha(k+1) = \alpha(k) + \gamma h(k)\,(1 - \alpha(k)), \quad \text{for } \alpha(k) \geqslant 0.5 \qquad 0<\alpha<1 \qquad (7)$$
$$\alpha(k+1) = \alpha(k) + \gamma h(k)\,\alpha(k), \quad \text{for } \alpha(k) < 0.5$$

in which γ is a positive constant used to modify the convergence rate of the updating formula.

By using this kind of formula we obtain a smooth and bounded variation of $\alpha < 0 ; 1 >$ and therefore a smooth and bounded variation of the PID parameters.

Note that this fuzzy mechanism has no denormalisation module because this is in fact done by the parameter γ, which turns out to be of great importance to the controllers performance.

After having explained the new control scheme, in particular the α-updating formula, we are able to discuss stability properties. By linearising the differential equations which describe the closed-loop behaviour around a constant set-point (r_0, φ_0) we can derive the following linear differential equation:

$$\ddot{e} + C_1\,\dot{e} + C_2 e = 0 \qquad (8)$$

in which: $e = r_0 - r$ or $e = \varphi_0 - \varphi$

For a conventional PD controller C_1 and C_2 only depend on the controller parameters K_P and K_D and some system constants (appendix A).

$$C_1 = cK_D \qquad (9)$$
$$C_2 = cK_P$$

in which the positive constant c depends on the system constants.

In order to prove stability we try the following candidate Lyapunov function:

$$V = \tfrac{1}{2}\,\dot{e}^2 + \tfrac{1}{2} C_2 e^2 \qquad (10)$$

With eq. 8 we can derive:

$$\dot{V} = -C_1\,\dot{e}^2 \qquad (11)$$

Note that in order to be a Lyapunov function, V must satisfy:

$$V > 0 \text{ if } e \neq 0 \text{ or } \dot{e} \neq 0 \qquad (12)$$
$$\dot{V} < 0 \text{ if } \dot{e} \neq 0 \qquad (13)$$

Therefore V is a Lyapunov function when the control parameters are positive and constant resulting in an assymptotically stable closed-loop response.

In case of the fuzzy self-tuning PD controller the coefficients C_1 and C_2 depend on the time-varying parameter α (see eq. 2 and eq. 3):

$$C_1 = \frac{c}{(1 + \alpha(t))} \qquad (14)$$
$$C_2 = c\alpha^2(t)$$

Note that because α is bounded ($\alpha(t) \in <0,1>$), C_1 and C_2 are positive and bounded as well. Because $\alpha = \alpha(t)$ the linear differential equation (8) becomes time-varying:

$$\ddot{e} + C_1(\alpha(t))\,\dot{e} + C_2(\alpha(t))\,e = 0 \qquad (15)$$

Using the same Lyapunov function (10) now yields:

$$\dot{V} = -C_1\,\dot{e}^2 + \tfrac{1}{2}\,\dot{C}_2\,e^2 \qquad (16)$$

with:

$$\dot{C}_2 = 2c\alpha(t)\,\dot{\alpha}(t) \qquad (17)$$

Because $\dot{C}_2$ can be positive, condition (13) may no longer be met, so candidate Lyapunov function (10) cannot be used in this case.

Although simulations show assymptotically stable process responses, all efforts to prove stability when $\dot{\alpha}(t) \neq 0$ have been unsuccessful up to now. Therefore further investigation will be necessary to prove stability of the time-varying system. As a starting point for further research on stability the use of Lyapunov exponents [10] can be investigated.

Simulation and results

The fuzzy self-tuning PID controller is simulated on a robot arm with two degrees of freedom, moving objects in a horizontal plane. These two degrees of freedom are the radial displacement (r), controlled by a radial force and the angle of the robot arm (φ) controlled by a torque. Each degree is controlled separately, so we need two fuzzy self-tuning PID controllers. The geometry and details of this robot arm are described in appendix A.

In order to evaluate the performance of the new controller, we compare its set-point as well as path-tracking and load disturbance responses with conventional well-tuned PID controllers. Due to the nature of the system, the responses improve as the maximum possible control action increases. This means that in order to get a realistic comparison an equal maximum cosntrol action for both the new and the conventional PID controller must be set. There are two control actions, the radial force and the torque of which the maximum values are set at 800N and 800Nm respectively.

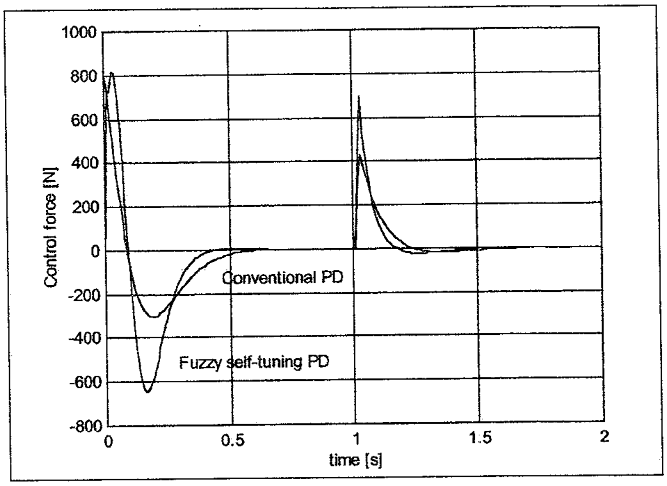


Fig. 8 : Control signal compared with the first PD controller.

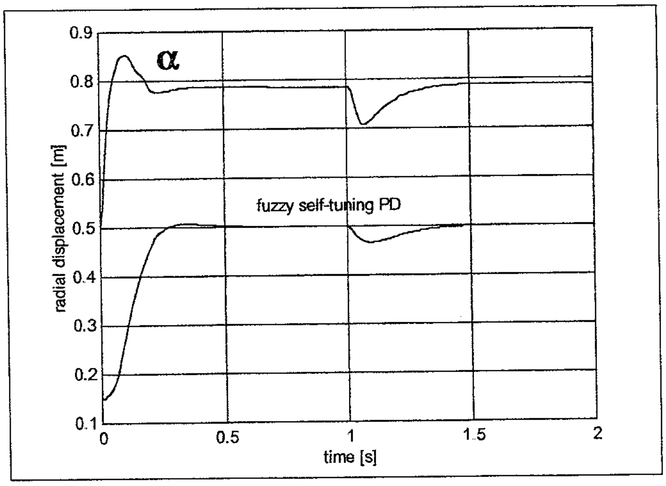


Fig. 9 : Set-point and load disturbance responses together with the updating parameter α.

Set-point and load disturbance responses

In this case we can suffice with a PD action because the I-action could not improve the responses. The tuning of the conventional P(I)D controller starts with finding the maximum value for K_P in such a way that the above mentioned control action limits are not exceeded. Then the damping parameter K_D can be increased to achieve the desired response characteristics in terms of overshoot, rise-time etc.

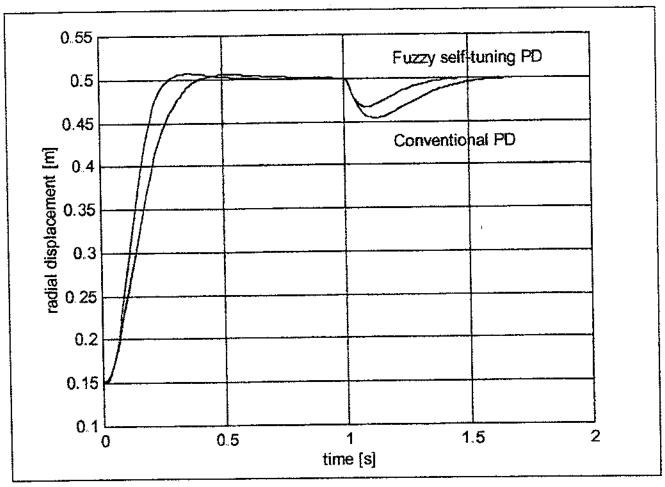


Fig. 6 : Set-point and load disturbance responses compared with the first PD controller.

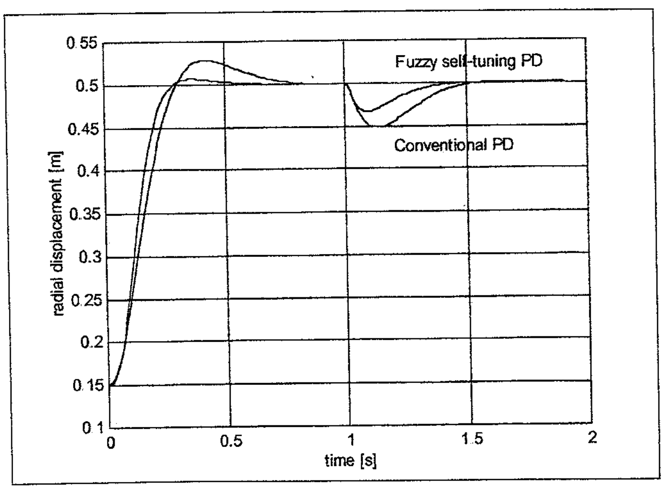


Fig. 7 : Set-point and load disturbance responses compared with the second PD controller.

The tuning of the parameters K_{P0} and K_{D0} is done in a similar way. After that the convergence parameter γ and the rulebase can be adjusted to the specific system to improve the performance without exceeding the control action limits. This concept is compared with two differently tuned PD controllers : the first one having a comparable overshoot from the set-point and the second one with less damping resulting in a faster response with shorter rise-time but more overshoot.

As we can see in *figure 6* and *figure 7*, both set-point and load disturbance (at t = 1s) responses improve considerably. The fuzzy self-tuning PD controller is either much faster or has less overshoot. In *figure 8* we can see that both the control actions of the fuzzy self-tuning controller and the first conventional PD controller stay within the limits that are set above.

The responses and control action for the angle of the robot arm are not displayed here because they show similar results as the radial displacement of the robot arm.

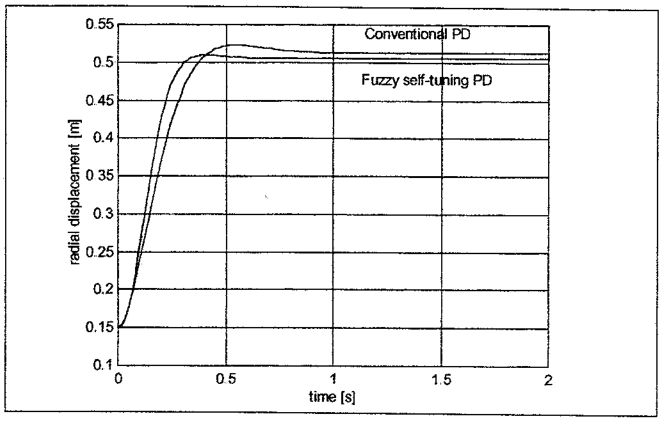


Fig. 10: The path-tracking responses of the fuzzy self-tuning PD controller and the conventional PD controller.

In *figure 9* the course of α together with the set-point response of the fuzzy self-tuning controller are shown, so we can check if the fuzzy rules are applied correctly. We can see that α indeed increases till the set-point is approached (thus in case of a large deviation from the set-point) after which α keeps decreasing till the response approaches the set-point again. The divergence at t = 1s caused by the load disturbance is slowed down by the decreasing α.

Path-tracking responses

In this case the robot arm has to follow a predefined path hih will b a circular course. As we can see in *figure 10* both the fuzzy self-tuning PD and the conventional PD controller discussed above can not provide adequate control because they keep a steady state deviation from the desired radius of the circular course. The reason is as follows: when the robot arm approaches the desired radius the convergence is slow, so the radial control force mainly consists of the proportional action. At a certain moment when the robot arm further approaches the desired radius, the radial control force (≈ proportional control action) equals the centrifugal force so there is no resulting radial force left to eliminate this steady state deviation.

We can see in *figure 10,* however, that this steady state deviation of the fuzzy self-tuning PID controller is much smaller, which is caused by the fact that during the simulation the fuzzy adaptive mechanism has put α at a higher level resulting in a larger proportional parameter.

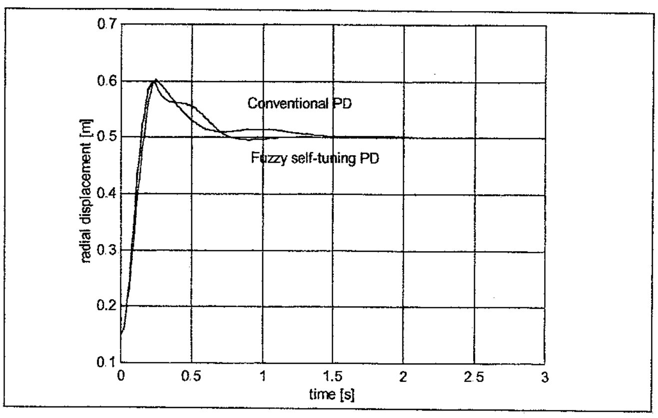


Fig. 11: Path-tracking responses with Integral action and feedforward term.

This steady state deviation can be eliminated by introducing an integral action. So in this case there is a need for all three controller gains. The control is also aided by an inverse-dynamics based feedforward term (see fig. 1) which computes an input based on the desired output and the differential equations of the system. As we can see in *figure 11* the steady state deviation is indeed eliminated. However, the I-action has resulted in a much larger overshoot. The fuzzy self-tuning PID controller is faster in eliminating the steady state deviation than a standard PID controller, with the same rise-time and overshoot, because after the maximum overshoot has occurred, α remains high resulting in a small D-action and large P- and I-actions. In this case the maximum allowed force in radial direction is set at about 1700N and both the controllers are tuned in a way which leads to a compromise between maximum overshoot and the time needed to eliminate the steady state deviation.

Conclusions

From the preceding results we can draw the following conclusions.

In *figure 9* we see that the course of the updating parameter α follows the fuzzy rules that are set in the rule-base in *figure 4* very smoothly and correctly, so we can conclude that the fuzzy logic implemented in this controller works well.

Secondly, the simulations and comparisons with conventional P(I)D controllers lead to the conclusion that this parameter α together with the chosen parameterisations of the controller gains indeed lead to a better performance of the new controller. On the basis of these simulations we expect the proposed scheme to yield better responses in the control of other mechanical systems as well; however, future research is required to verify this.

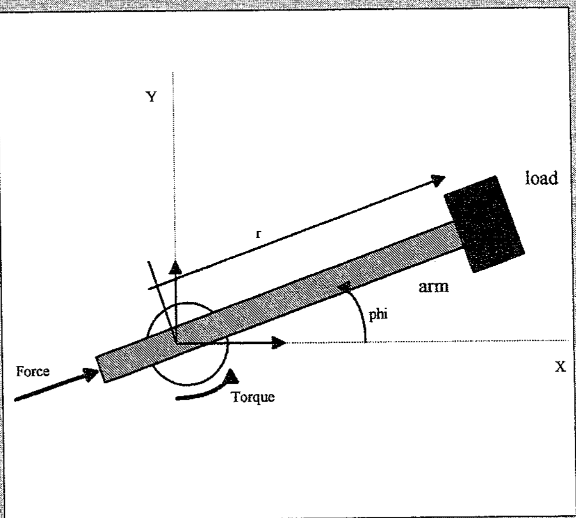
On the other hand it must be emphasized here that the forms of those parameterisations along with the chosen constants and parameters are not unique. So in order to get a fine response one should know the influences of all these aspects, which brings us to the main disadvantage of this fuzzy self-tuning P(I)D controller: the adjustment of the constants and parameters, the formation of the precise form of the rule-base and the parameterisations, in other words the tuning of the fuzzy self-tuning P(I)D controller, is more complicated than tuning a common P(I)D controller. So the question whether or not the extra tuning effort of the fuzzy self-tuning P(I)D is worthwhile, strongly depends on the required process reponses.

Bibliography

1. S.Z.HE, S.H. Tan, F.L. Xu and P.Z. Wang, "PID Self-tuning control using a fuzzy adaptive mechanism, "Proc. 1993 IEEE Intern. Conf. Fuzzy Syst., San Fransisco, pp. 708-713.
2. D. Driankov, H. Hellendoorn and M. Reinfrank "An introduction to fuzzy control", Berlin Heidelberg 1993.
3. K.J. Astrom and T. Hagglund, "Atomatic tuning of PID controllers", Instrument Society of America, USA, 1988.
4. K.J. Astrom and T. Hagglund, "An industrial adaptive PID controller," Proc. 1989 IFAC Symp. Adaptive System in Control and signal Processing, pp. 293-298, 1989.

Appendix A

Description of the 2-DOF robot



Model

degrees of freedom : - radial displacement (r)
- angle of the robot arm φ

input signals : - force in radial direction (F)
- torque (T)

assumptions : - all bodies are rigid
- friction influences are neglected

system boundaries : - max. radial displacement 1m
- min. radial displacement 0.15m

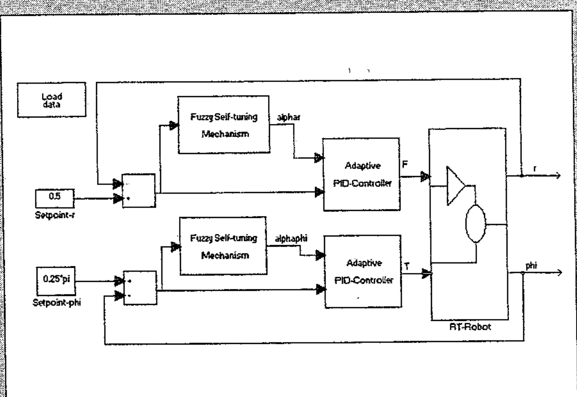
equations of motion :

$$(m_p + m_1)\,\ddot{r} - ((m_p + m_1)\, r\, \dot{\varphi}^2 - \frac{1}{2} m_1\, l\, \dot{\varphi}^2) = F$$

$$(J + (m_p + m_1) r^2 - m_1 r l + \frac{1}{3} m_1 l^2)\,\ddot{\varphi} + (2r(m_p + m_1)\,\dot{r} - m_1 l \dot{r})\,\dot{\varphi} = T$$

constants : - m_p = 20kg; mass of load
- m_1 = 10kg; mass of robot arm
- J = 5kgm^2; inertia of system without robot arm
- l = lm; lenght of robot arm

Simulation



Simulation environment :

matlab-simulink

Simulation method :

Runga-Kutta 5

Sampling time :

0.01

Fuzzy mechanism

Normalisation factors :

17 (e_r) / 2.5 (e_r)

8 (e_φ / l (e_φ)

α-updating formula

Convergence rate γ :

$\gamma_r = 0.007$

$\gamma_\varphi = 0.007$

PID-parameterisations

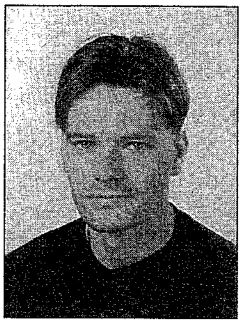
- *set-point control* :

	fuzzy self-tuning PD r-part/φ-part	PD(fast) r-part/φ-part	PD(slow) r-part/φ-part
$K_{P(0)}$	7500/3500	2300/1025	2300/1025
$K_{D(0)}$	1200/370	390/125	465/145

- *Path-tracking control* :

	fuzzy self-tuning PID r-part/φ-part	PID r-part/φ-oart
$K_{P(0)}$	8500/3500	2000/1025
$K_{D(0)}$	550/370	200/125
$K_{I(0)}$	12000/0	3750/0

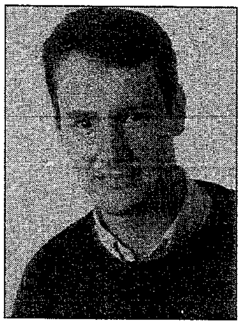
5. C.C. Hang, K.J. Astrom and W.K. Ho, "Refinements of the Ziegler-Nichols tuning formula", IEEE Proc. Part D, Vol. 138, pp. 111-118, 1991.
6. S.Z. HE, S.H. Tan, C.C. Hang and P.Z. Wang, "Design of an on-line rule-adaptive fuzzy control system," Proc. 1992 IEEE Intern. Conf. Fuzzy Syst., San Diego, pp. 83-91.
7. M. Sugeno, "Industrial applications of fuzzy control," North-Holland, Amsterdam, 1985.
8. B. Jager, I. Lammerts, F. Veldpaus, "Course on advanced control," Eindhoven University of Technology
9. J.J. Kok, "Werktuigkundige Regeltechniek 1", Eindhoven University of Technology
10. R.H.B. "Fey, Steady-state behaviour of reduced dynamical systems with local non-linearities," PhD thesis, Eindhoven University of Technology, 1992.



Mark A.D. BAYENS

■ Meckelenburgstraat 25, NL - 5615PZ Eindhoven, The Netherlands.

■■ Mark Bayens was born in Heerlen, The Netherlands in 1972. He started studying at the faculty of mechanical engineering of the University of Technology in Eindhoven in 1990. This article was a result of his first training period.
He is currently in his fifth year and is expecting to complete his engineering study in 1995.



Rene VAN DE MOLENGRAFT

■ Eindhoven University of Technology, P.O. Box 513, NL - 5600 MB Eindhoven, The Netherlands.

■■ Rene van de Molengraft was born in Eindhoven, The Netherlands, in 1963. He graduated in mechanical engineering in 1986 at EUT. He received his Ph.D. in 1990 with a thesis on the identification of non-linear mechanical systems. Since 1991 he is an assistant professor at the EUT in the systems and control section of the faculty of mechanical engineering. His main interest at the moment is control of non-linear mechanical systems.