Method to build an initial adaptive Neuro-Fuzzy controller for joints control of a legged robot

J-C Habumuremyi, P. Kool and Y. Baudoin
Royal Military Academy-Free University of Brussels
08 Hobbema str, box:MRTM, 1000, Brussels, Belgium
E-mail:Jean-Claude.Habumuremyi@rma.ac.be;Pkool@vub.ac.be;
Yvan.Baudoin@rma.ac.be

Abstract

Due to the complexity of walking robots which has in general a great number of degrees of freedom, cognitive modelling controller such as Fuzzy Logic, Neural Networks…seems to be reasonable. Fuzzy Logic Controller is more used because it lets you describe desired system behaviour with simple "if-then" relations. But it has a major limitation because in many applications, the designer has to derive "if-then" rules manually by trial and error. In this paper, we show an original method to fix initial parameters of a Sugeno fuzzy controller apparently to Ziegler-Nichols rules. Simulations and application to a six-legged robot named AMRU5 has proved the effectiveness of this method. Then, we make the fuzzy controller obtained adaptive by combining it to Neural Networks technologies.

1 Introduction

Many robots (manipulator and mobile robots), until now, are controlled using linear controllers such as PD, PID…which are independent for each joint. It can be proven that those controllers are fairly effective. The two main reasons are [1]:

- The large reduction ratios between the actuators and the link mechanism (non-linearities and coupling terms become less important)
- The large feedback gains in the control loops (they enlarge the domain where the complete robot dynamics is locally equivalent to a linear model).

These controllers limit the use of robots to slow motion applications and they operate over a small closely controlled range. However, the normal operational range of a robot is large and its payload also can vary. To have a controller which works on different operational range and take into account the change of the payload, the environment and the uncertainties (friction, flexibility…), necessitate a sort of on-line parameter estimation scheme in it (an adaptive controller). Most of classical adaptive controllers are based on the well-known dynamic properties of robots which stipulate that parameters models (mass, moment inertia, link lengths…) are linear. They were applied successfully in simple cases (such as inverted pendulum system, planar two-link manipulator, five bar linkage manipulator…) where mathematical model can be deduced. They were also applied in complex cases but simplified by limiting for example, the normal operational range of the robot due to the difficulty of estimating uncertainties of the model. The problem become more complex for a walking robot which has in general a large number of degrees of freedom (we have 18 just for the robot to walk) and which requires changing internal parameters depending on the environment that it explores. Also, it seems practically difficult to build a representative model of a walking robot due to the problem of having accurate internal parameters (distance between joints, moment inertia…) and to accurately model some complex phenomena such as backslash, friction…In this case, cognitive modelling such as Fuzzy Control and Neural Networks seems to be reasonable. That's why, we have investigate the way to control (joints and gait control) a six-legged robot named AMRU5 shown in Figure 1 using ANFIS (Adaptive Neuro-Fuzzy Inference System). In this paper, we are more focused on the joints control of the robot and we show an original method on how to fix initial parameters of a Sugeno fuzzy controller.

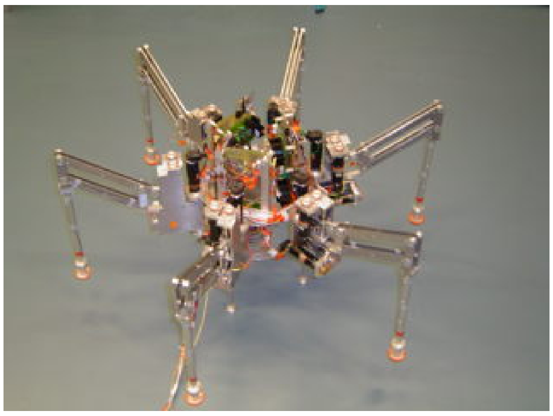
Figure 1: Walking Robot AMRU5

2 ANFIS[1] Architecture

2.1 Fuzzy Logic and Neural Networks

Fuzzy Logic Controller (FLC) is more used because it lets you describe desired system behaviour with simple « if-then » relations. In many applications, this gets you a simpler solution in less design time. In addition, you can use all available engineering know-how to optimise the system performance directly. While this is certainly the beauty of fuzzy logic, at the same time it is a major limitation. In many applications, knowledge that describes desired system behaviour is contained in data sets. The designer has to derive the « if-then » rules from the data sets manually, which requires a major effort with large data sets. This is often done by trial and error. Without adaptive capability, the performance of FLCs relies on two factors: the availability of human experts, and knowledge acquisition techniques to convert human expertise into appropriate fuzzy « if-then » rules and membership functions. These two factors substantially restrict the application domain of FLCs. Changing shapes of membership functions can drastically influence the quality of the FLC. Thus methods for tuning fuzzy controllers are necessary.
Artificial neural networks are highly parallel architectures consisting of simple processing elements, which communicate through weighted connections. They are able to approximate or to solve certain tasks by learning from examples. When data sets contain knowledge about the system to be designed, a neural net promises a solution because it can train itself from the data sets. However, only few commercial applications of neural nets exist due to the lack of interpretation of the solution, the prohibitive computational effort and the difficulty to select the appropriate net model .
It becomes obvious that a clever combination of the two technologies delivers the best of both. Neuro-Fuzzy [2] is a combination of the explicit knowledge representation of the fuzzy logic with the learning power of the neural nets.

2.2 Neuro-Fuzzy systems

There are many approaches to combine fuzzy logic and Neural Networks, known among them are:

- Cooperative Neuro-Fuzzy systems where ANN learning mechanism determines the FIS membership functions or fuzzy rules from the training data after ANN goes to the background.
- Concurrent Neuro-Fuzzy systems where ANN assists the FIS continuously to determine the required parameters.
- Fused Neuro-Fuzzy systems is the methods where FL and ANN share data structures and knowledge representations. In the above methods FL and ANN are separated. In fused systems, ANN learning algorithms are used to determine the parameters of FIS. For that, Fuzzy system is represented in a special ANN like architecture. Some of major works in this area are NEFCON, ANFIS[3], FALCON, GARIC, FINEST and many others.

[1] Adaptive Neuro-Fuzzy Inference Systems

2.3 ANFIS Architecture

In our application, we use the ANFIS [3] architecture. It keeps the structure of the fuzzy controller that is determined by the fuzzy rules as depicted in Figure 2.

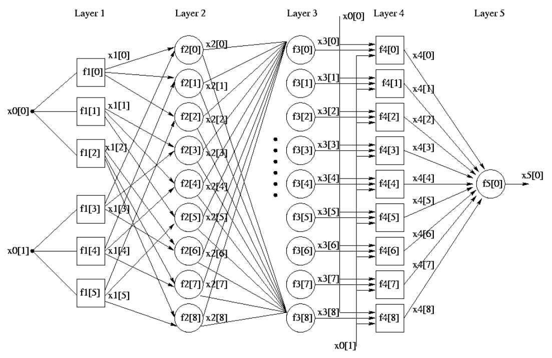

Figure 2: ANFIS Architecture

At layer 1, every node is adaptive (premise parameters) with a node function which is the membership function. Node output at layer 2 represents the firing strength of a rule. In our application, it is a product of all the incoming signals but can be in general any T-norm operators that performs fuzzy AND. At layer 3, a normalised firing strengths is realised by making a ratio of rule's firing strength to the sum of all rule's firing strengths. Nodes at layer 4 are adaptive (consequent parameters) with node function which can be a first-order Sugeno, zero-order Sugeno, Mandani or Tsukamoto fuzzy model.

3 Design of an initial zero-order Sugeno Fuzzy Logic controller

It is important to have an initial controller which works properly in a closed range. One of the reasons is that during the on-line learning, optimization algorithm will be used and the solution cannot converge to the good one if parameters of the controller are set far away of the true. We have to make a fuzzy controller which work properly in a closed range then make it adaptive to take into account uncertainties of the model. Many controller design avoid this problem by making first a classical controller (PD usually) then add another controller (Fuzzy or Neural Network) to deal with uncertainties. This makes the controller more complex. In our method,

we design an initial Neuro-fuzzy controller which works similarly as a classical one. After, we make it adaptive to deal with uncertainties. To find good parameters of a fuzzy logic controller is not an easy task because they have in general a lot of parameters. If we have *n* input, *m* triangular membership functions (2 parameters to adjust by membership function) and a zero-order Sugeno FIS is used, we have to fix $3m^n$ parameters. For illustration of the method, we will use a fuzzy system with 2 triangular membership functions (N, P), 3 input (*e(n), e(n-1)* and *e(n-2)*) and a zero-order Sugeno FIS as shown on Figure 2, but this method can be generalised. In this case, we have to fix 24 parameters. But if we fix parameters of the membership function, we have only 8 parameters to fix. A typical rule of such a system has the form:
If *e(i)* is N, *e(i-1)* is P and *e(i-2)* is N then the output

$$z_1 = p_1 e(i) + q_1 e(i-1) + r_1 e(i-2) + s_1 \quad (1)$$

where $\{p_1, q_1, r_1, s_1\}$ is the parameter set of one node. The equation (1) become $z_1 = s_1$ (2) for a zero-order Sugeno FIS.

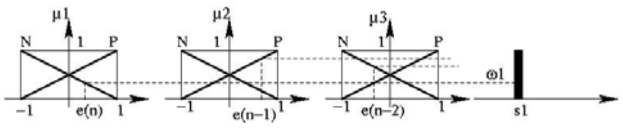

Figure 2: Zero-Order Sugeno: two triangular MF and three input

The first method to fix parameters of a controller is simple trial and error. Unfortunately, intuitive tuning procedures can be difficult to develop in the case of Sugeno FIS because a change in one tuning constant tends to affect the performance of others terms in the controller's output. Also, the great number of parameters makes this method practically impossible. The second method can be the analytical approach to the tuning problem. It involves a mathematical model of the process. This method cannot be used because the advantage of fuzzy logic is precisely the fact that it is used on complex processes where the establishment of a reliable model is unimaginable. The third approach to the tuning problem is something of a compromise between purely self-teaching trial and error techniques and the more rigorous analytical techniques. It was originally proposed by John G. Ziegler and Nathaniel B. Nichols [5] and remains popular today because of its simplicity and its applicability to process which can be describes by a "gain", a "time constant" and a "deadtime" (which is the case of joints of robots actuated by DC motor). Ziegler and Nichols came up with a practical method for estimating the proportional, the integral and the derivative parameters of a PID controller. In this paper, we show how these techniques can be applied in the design of a fuzzy controller.

3.1 How the method was developed?

Many techniques used to turn a Mandani Fuzzy Model (which has less parameter compare to Sugeno Fuzzy Model) are intuitive. In many papers, books…they show which parameters to increase or to decrease by considering the rise time, the overshoot and the steady state error [4]. These techniques seem more like the art than engineering and they are difficult to apply them to Sugeno Fuzzy Model. The best solution to turn parameters of a Sugeno Fuzzy Model is by fusing Neural Networks to Fuzzy Logic Systems. But we need data to train the system. The first solution can be to collect them from a classical controller implemented to the real robot by giving random trajectories to the actuators. We noticed that we cannot cover all possible operating regions, the time to read data (on encoders, actuators,…) and to write them on a stored device is too short (the microcontroller has no much time sometime to write all the value) and there is noise on the data. The error obtained after training is still big by using these data. Another original solution could be the use of Ziegler-Nichols rules originally applied to PID controllers. The analogue PID controller is expressed by the equation:

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de}{dt} \quad (2)$$

where *e* is the difference between the set point and the process output and *u* the command signal. K_p, K_i and K_d are controller parameters.

Two practical methods can be used to have a first estimate of the PID controller parameters:

- the step-response method
- and the frequency response method (only this method will be considered in this paper)

3.1.1 The step-response method

This method is based on a registration of the open-loop response of the system, which is characterized by two parameters. The parameters (a and L) are determined from a unit step response of the process, as shown in Figure 3. When those parameters are known, the controller parameters are obtained from Table 1.

Controller Type	K_p	K_i	K_d
P	$\frac{1}{a}$		
PI	$\frac{0.9}{a}$	$\frac{0.3}{aL}$	
PID	$\frac{1.2}{a}$	$\frac{0.6}{aL}$	$0.6aL$

Table 1: Parameters obtained from step-response method

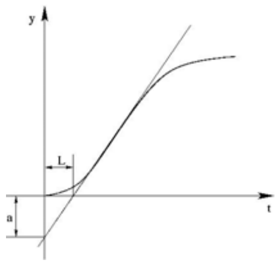

Figure 3: Step-response method

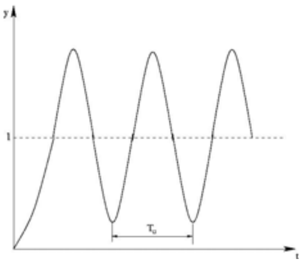

Figure 4: Frequency response method

3.1.2 The frequency-response method

The idea of this method is to determine the point where the Nyquist curve of the open-loop system intersects the negative real axis. This is done by connecting the controller to the process and setting the parameters so that pure proportional loop system is obtained. The gain of the controller is then increased until the closed-loop systems reaches the stability limit. When this occurs, the gain K_u and the period of oscillation T_u shown on Figure 4 are determined. The controller parameters are then given by the Table 2.

Controller Type	K_p	K_i	K_d
P	$0.5K_u$		
PI	$0.45K_u$	$\frac{0.54K_u}{T_u}$	
PID	$0.6K_u$	$\frac{1.2K_u}{T_u}$	$0.075K_uT_u$

Table 2: Parameters obtained from frequency-response method

3.1.3 Method of turning an UFLC based on the frequency-response

If the ultimate gain K_u and the ultimate period T_u of the process was determined by experiment or simulation, equation…can be write as follow:

$$u(t) = 0.6K_u e(t) + \frac{1.2K_u}{T_u}\int e(t)dt + 0.075K_uT_u\frac{de}{dt} \quad (3)$$

There exist different methods to convert equation (3) into discrete form for digital implementation such as Tustin approximations (or trapezoidal approximations), ramp invariance, rectangular approximations…When the sampling time T is short, all these methods have nearly the same performance. We'll use rectangular approximations. Equation (3) becomes:

$$u(n) = 0.6K_u e(n) + \frac{1.2K_u}{T_u}\sum_{i=1}^{n} e(i)T + 0.075K_uT_u\frac{e(n)-e(n-1)}{T} \quad (4)$$

$$u(n-1) = 0.6K_u e(n-1) + \frac{1.2K_u}{T_u}\sum_{i=1}^{n-1} e(i)T + 0.075K_uT_u\frac{e(n-1)-e(n-2)}{T} \quad (5)$$

(4)-(5) gives

$$\Delta u(n) = u(n) - u(n-1) = K_1e(n) + K_2e(n-1) + K_3e(n-2) \quad (6)$$

Where $K_1 = \frac{3K_u}{40}\left(\frac{T_u^2 + 8TT_u + 16T^2}{TT_u}\right)$, $K_2 = \frac{-3K_u}{20}\left(\frac{T_u - 4T}{T}\right)$ and

$$K_3 = \frac{3}{40}\frac{K_uT_u}{T}$$

Equation (6) can now be used to build a FLC controller that we called a UFLC (Unit FLC). UFLC will be determined by the equation:

$$\Delta u_u(n) = K_1e_u(n) + K_2e_u(n-1) + K_3e_u(n-2) \quad (7)$$

where $e_u()$ is between -1 and 1. If we define a step t (0.001 for example), we can define a set A of numbers between -1 and 1 as follow:

$$A = \{-1, -1+t, -1+2t, \ldots, 1-2t, 1-t, 1\}$$

Then we constitute all possible set $\{e_u(n), e_u(n-1), e_u(n-2)\}$ with numbers which belong to the set *A*. From each set, we calculate $\Delta u_u(n)$ using equation(7). Finally, we can use the set $\{e_u(n), e_u(n-1), e_u(n-2), \Delta u_u(n)\}$ to train the Neuro-Fuzzy Controller. Using a hybrid learning paradigm (least square error algorithm for consequent parameters which are linear and backpropagation for premise parameters), we noticed that the initial membership functions did not change (premise parameters remain the same), only consequent parameters change) With 2 triangular membership functions choose for our illustration, we have analytical expression shown in the Table 3.

Rule	$e(n)$	$e(n-1)$	$e(n-2)$	$\Delta u_n(n)$
1	N	N	N	$\frac{-6K_uT}{5T_u}$
2	N	N	P	$\frac{-3K_u(8T^2 - T_u^2)}{20T_uT}$
3	N	P	N	$\frac{-3K_u(2T + T_u)^2}{10T_uT}$
4	N	P	P	$\frac{-3K_u(8T^2 + 8T_uT + T_u^2)}{20T_uT}$
5	P	N	N	$\frac{3K_u(2T + T_u)^2}{10T_uT}$
6	P	N	P	$\frac{3K_u(4T^2 + T_u^2)}{10T_uT}$
7	P	P	N	$\frac{3K_u(8T^2 - T_u^2)}{20T_uT}$
8	P	P	P	$\frac{6K_uT}{5T_u}$

Table 3: Rules of the system used for illustration

The same procedure can be applied to the step-response method and to derive rules of a controller which depends from the parameters *a* and *L*.

3.1.4 Use of the UFLC on a real process

In practice, error will not belong always between -1 and 1. We need some transformation to use the UFLC design on a real process. If the minimum error of the system is *a* and the maximum is *b* (*a* and *b* was determined by the limitation of each joint), a reduced error $e_u(n)$ (error between -1 and 1) can be expressed as follow:

$$e_u(n) = \frac{2}{b-a}\left(e(n) - \left(\frac{b+a}{2}\right)\right) \quad (8)$$

and $e(n) = e_u(n)\dfrac{b-a}{2} + \dfrac{b+a}{2}$ (9)

Equation (9) in (6) gives

$$\Delta u(n) = (K_1 e_n(n) + K_2 e_n(n-1) + K_3 e_n(n-2))\frac{b-a}{2} + (K_1 + K_2 + K_3)\frac{b+a}{2} \quad (10)$$

Equation (10) becomes after simplification:

$$\Delta u(n) = \Delta u_u(n)\frac{b-a}{2} + \frac{6K_u T}{5T_u}\frac{b+a}{2} \quad (11)$$

Figure 5 shows how UFLC is used on a real process.

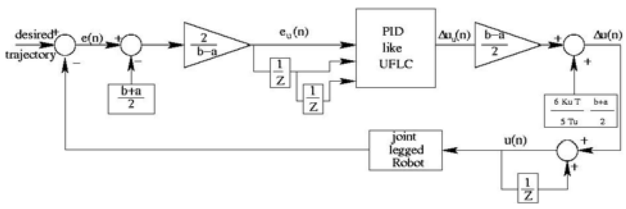

Figure 5: Use of the UFLC on a real process

3.1.5 How to determine K_u ?

Two ways can be used to determine the ultimate gain K_u :

- By using the proportional classical controller and change K_p until the system reach the stability limit
- By using the method mentioned above, we can easily build a UFLC which react exactly as a classical controller. For example, if we change to $-K_p$ the value of $\Delta u_u(n)$ for rule 1, 2, 3, 4 and to K_p for rule 5, 6, 7, 8; the UFLC will react as a proportional classical controller.

3.1.6 Application to a known function transfer

To allow comparison between a classical PID controller and a UFLC, we have applied the method to the process with a transfer function

$$G(s) = \frac{1}{(s+1)^3} \quad (13)$$

This process has the ultimate gain $K_u = 8$ and the ultimate period $T_u = \frac{2\pi}{\sqrt{3}} \approx 3.63$. From the Table 2 and Table 3, we can easily design a PID and an UFLC. Figure 6 shows the Matlab schematic used to compare the two controllers with the step function as the input.

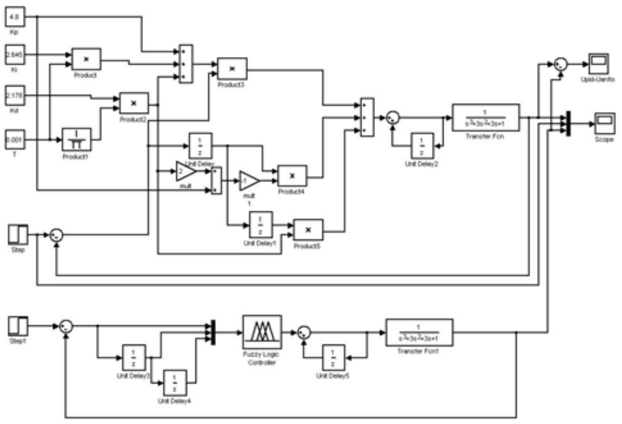

Figure 6: Comparison between PID controller and UFLC

The output of the two controllers and their result of the error (output of the PID controller subtract to the output of the UFLC) are shown on Figure 7 and 8. The error is less than 0.0063.

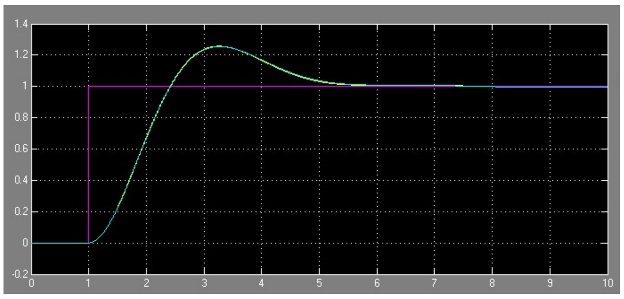
Figure 7: Output of the PID controller and the UFLC

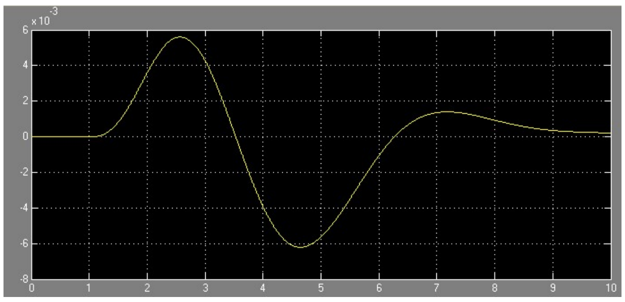
Figure 8: Error between the PID controller and the UFLC

4 Conclusions

We show in this paper original methods to turn an initial ANFIS controller based on Ziegler-Nichols rules. The methods were applied to a zero-order Sugeno with 2 triangular membership functions and 3 inputs. Analytical expression of turning rules was deduced from these methods. These methods can be easily extended to more than 2 triangular membership functions and to other types of fuzy model: first-order Sugeno, Mandani, Tsukamoto. Indeed, the structure of 2 membership functions can be found in a structure with more than 2 membership functions. The zero-order Sugeno is a particular case of a first-order Sugeno (just the constant element is no null), of a Mandani fuzzy model (each rule's consequent is specified by a fuzzy singleton) and of a Tsukamoto (each rule's consequent is specified by a step membership function center at the constant).
We obtain with the methods mentioned above a first approximation of the controller which can be refining after with adaptive paradigm.

5 References

[1] T. Yoshikawa, "Foundations of Robotics: Analysis and Control". *Massachusetts Institute of Technology*, USA, 1990

[2] D. Nauck, F. Klawonn and R. Kruse. "Combining Neural Networks and Fuzzy Controllers" *FLAI'93*, Linz, Austria, Jun. 28-Jul.2, 1993

[3] J. –S. R. Jang, C. T. Sun and E. Mizutani. "Neuro-Fuzzy and Soft Computing" . *Prentice-Hall* (UK), 1997

[4] B. Subudhi, A. S. Morris, "Fuzzy and Neuro-Fuzzy approaches to control a flexible single-link manipulator " *IMechE 2003*, 29 May 2003

[5] J.G. Ziegler and N.B. Nichols, "Optimum settings for automatic controllers", *Trans. ASME*, 64, 759, 1942