FuzzyCAT – Towards a Mathematical Analysis of Fuzzy Controllers

KRISTER FORSMAN, ANDERS STENMAN and JAN-ERIK STRÖMBERG

Department of Electrical Engineering
Linköping University
S-581 83 Linköping
Sweden

email: `krister@isy.liu.se`

1993-03-05

Technical reports from the automatic control group in Linköping are available by anonymous ftp at the address 130.236.24.1 (`joakim.isy.liu.se`). This report is contained in the compressed Postscript file named `/pub/reports/LiTH-ISY-R-1452.ps.Z`

FuzzyCAT – Towards a Mathematical Analysis of Fuzzy Controllers

KRISTER FORSMAN¹, ANDERS STENMAN and JAN-ERIK STRÖMBERG

Department of Electrical Engineering
Linköping University
S-581 83 Linköping
Sweden

email: `krister@isy.liu.se`

1993-03-05

Abstract. The aim of this paper is to show that, contrary to what is commonly believed, it is possible to get rather simple analytic expressions for a class of so called fuzzy controllers and to describe a Maple package called FuzzyCAT to support analysis and simulation of fuzzy controllers. The restrictions made in the program concern the shape of the membership functions, the defuzzification methods available and the number of input and output variables.

Keywords: fuzzy control, nonlinear controllers, symbolic computation, CACSD, stability

1 Basic Definitions

Fuzzy control (FC) and fuzzy logic are becoming more and more common in industrial applications. However this trend has not yet been accompanied by a similar surge of interest in the academic control society. The reasons for this is maybe that control theorists traditionally have very high demands on mathematical rigor and analysis before accepting new techniques, and such mathematical analyses are still scarce in the available FC literature. Nevertheless, FC has so many advantages that its importance cannot be neglected. Some references to the basics of FC and fuzzy logic are [1, 3, 5, 7, 12, 15].

We devote this introductory section to recalling the very basic ideas of FC, since there are comparatively few transparent accounts for this in the literature and the perspective is rarely the one a control theorist would prefer. Also the numerous approaches to various FC technicalities add to this confusion.

From a functional point of view a fuzzy controller simply constitutes a nonlinear mapping of its input to its output signals, where the nonlinearity is given by fuzzy logic. The main contribution of FC comes from the unconventional (from a control theorist's point of view) specification (parametrization) of this nonlinearity. Since there is no dynamics in the actual controller dynamics has to be introduced externally, by some filters. Given a control error e we thus implement a nonlinear PI-controller by feeding the two signals e and $\int e$ into the nonlinear function in question.

From an implementational point of view the nonlinear function can be seen as built up of three parts: fuzzification, inference engine and defuzzification, see figure 1. The basic idea

¹All correspondence to the first author (KF).

Figure 1: The internal of a fuzzy control law and its position in the standard controller/plant configuration.

of fuzzy control is to use fuzzy logic to define the control action as a function of measured signals. Fuzzy logic is an extension of ordinary, boolean logic, where the concepts of “true” and “false” are extended to denote “degree” of truth. If 0 means “false” and 1 means “true”, then we allow the veracity of a statement to take any value in the interval $[0, 1]$. Thus, we can regard the number 0.5 as meaning “almost true” or “almost false”. To a property p defined on a set $\mathcal{S}$ we associate a *membership function* $\mu_p : \mathcal{S} \rightarrow [0, 1]$ that specifies to which extent the property is satisfied at each point of $\mathcal{S}$.

In a fuzzy controller the measured signals are first subject to “fuzzification”, which is simply an application of a number of membership functions μ to each measured signal. As seen in figure 1 the measured signals may either be the outputs of the plant or, more generally, the control errors.

After that, fuzzy logic “inference rules” are applied to determine the resulting control action, see the “Inference engine” box in figure 1. These rules are of the type

IF conditions on measurements THEN certain control action

The ability to express a control law in a “high-level” manner like this, is one of the reasons why fuzzy control has attracted people outside the control society to such an extent.

Both the condition and the control action are supposed to be given in terms of membership functions or fuzzy logic expressions in membership functions. This means that we can abbreviate each rule of the above type as $\mu(e) \rightarrow \pi(u)$ for some membership functions μ, π . The first step towards calculating the actual control action, is to compute the fuzzy output function.

Definition 1.1 An inference rule of the kind $\mu(e) \rightarrow \pi(u)$ translates to the *fuzzy output function*

$$\rho(u, e) = \mu(e) \wedge \pi(u) = \min\{\mu(e), \pi(u)\} \quad (1)$$

where μ and π are any membership functions and e the measured signal, which may in general be vector valued. Notice that ρ is a function of two variables if e and u are scalars: the min operator just chooses the smallest of two numbers (the minimization is pointwise, *not* over all values of e or u). $\mu(e)$ is sometimes referred to as the *degree of fulfillment*. $\square$

Now there is only one more step required to find the control output u : the “defuzzification step”. In mathematical language, the defuzzification is nothing else than a functional, i.e. a mapping from a space of functions to $\mathbb{R}$. Several defuzzification methods have been suggested, but the most commonly applied in continuous valued control problems, is the *Center of Area* (COA) method. The idea is to generate the output u that divides the area under the fuzzy output function in two equally large parts. More formally, we have:

Definition 1.2 The Center of Area (COA) defuzzification method produces the control law $u = c(e)$ where

$$c(e) = \frac{\int_{\mathcal{P}} \rho(v, e) v \, dv}{\int_{\mathcal{P}} \rho(v, e) \, dv} \quad (2)$$

where $\rho(u, e)$ is the fuzzy output function derived by the inference engine, $u \in \mathcal{P} \subset \mathbb{R}$ the control output and $e \in \mathcal{S} \subset \mathbb{R}^n$ the control error (which might be a vector). If e is a scalar we will refer to $c(e)$ as the *gain function*. $\square$

Note the close resemblance to *expected value* in probability theory. Since in all applications $\mathcal{P}$ is a compact set and ρ only takes values in the interval $[0, 1]$ the integrals involved converge. If the denominator $\int_{\mathcal{P}} \rho(v, e) \, dv$ is zero then so is the numerator; in this case we assign $c = 0$. We now see that the fuzzy control law is indeed a *static* non-linear function of e . The control law can be considered as a P-controller with variable gain, if e is the (scalar) control error.

If we have N different rules the ordered set $\{\rho_1, \dots, \rho_N\}$ is called the *fuzzy output vector*. Before we can apply COA we have to combine the rules to a single real valued function. The most common idea is to pretend there is a logic **or** between the rules. Thus we obtain the *combined fuzzy output function*

$$\rho(u, e) = \rho_1(u, e) \vee \rho_2(u, e) \vee \dots \vee \rho_N(u, e) = \max_i \{\rho_i(u, e)\} \quad (3)$$

Another possibility is to add the ρ_i before computing the center of area:

$$\rho(u, e) = \sum_i \rho_i(u, e) \quad (4)$$

A comparison between these and other defuzzification methods can be found in [5].

2 FuzzyCAT: Ideas, Features and Principles

Usually the integrals in (2) are solved numerically, either on-line while running the controller, or off-line while compiling the code for the controller. This however gives us no support in doing mathematical analysis of the result, i.e. we are left with experimental or simulation based analysis. But if proper restrictions are introduced to narrow down the set of possible membership functions, we are in fact able to solve (2) to get closed expressions in e . This was

Figure 2: Membership function.

the main motivation for initiating the development of FuzzyCAT (Fuzzy Control Analysis Tool).

FuzzyCAT has been developed as a M.Sc. thesis work by the second author and is implemented as a software package for the computer algebra system Maple [4]. A detailed description of the package is found in [11], but we will here present some of its main characteristics to give some flavor of the general idea. FuzzyCAT translates fuzzy logic rules to explicit (closed) mathematical expressions for the resulting control law as defined by the gain function. The mathematical expressions can either be rendered as ordinary Maple code or as a MATLAB [9] m-file suitable for incorporation in a simulation model in e.g. Simulink [8].

There is of course other software available for FC. One example is Togai's TIL-shell [13] which is a commercial package that lets one define fuzzy control laws graphically, plot these, simulate them and translate them into executable C-code. What is unique with FuzzyCAT is that *analytic expressions* for the control law are obtained.

FuzzyCAT is freely available by anonymous ftp at the address 130.236.24.1 or joakim.isy.liu.se, under the directory /pub/src/maple/fuzzycat. The size of the file containing all the Maple source code for FuzzyCAT, including help texts, is approximately 40 kbytes.

The first restrictions introduced by FuzzyCAT is that the number of controller inputs (measured signals) cannot exceed 2. Furthermore the controller output is assumed scalar. Given these and some additional restrictions to be mentioned below, the following main functions are currently available in FuzzyCAT:

- Generation of membership functions: `mfmake`
- Generation of fuzzy rules: `rule`, `combine_rules`
- Fuzzy logical operations: `And`, `Or`, `Not`
- Defuzzification: `defuzz`
- Generation of simulation code for MATLAB: `matlabfunc`
- Plotting of gain functions: `fuzzyplot`

A membership function is represented as lists of lists:

```
[[<function1>,<range1>],[<function2>,<range2>],...]
```

where each pair [$\langle \text{function} \rangle, \langle \text{range} \rangle$] is called a *subfunction*. The range of each subfunction is the interval on which the first expression of the subfunction defines the membership function. The restrictions introduced for membership functions are

1. The total function has to be continuous.
2. The subfunctions have to be analytically integrable, monotonic and surjective onto the interval $[0, 1]$, unless they are identical to 0 or 1 over their interval of definition.

This certainly allows for non-monotonic membership functions, though. Both the function and the range in a subfunction can depend on parameters. An example of a membership function is

```
> v_hi:=[[0,e=-emax..-a],[e/(2*a)+1/2,e=-a..a],[1,e=a..emax]]:
```

Figure 2 depicts `v_hi` with $e_{min} = -e_{max}$, $\alpha = -a$, $\beta = a$. FuzzyCAT defines a data type called `membershipfunction` so that the call `type(v,membershipfunction)`; returns true if `v` is a valid membership function. The function `mfmake` is a user's support for designing membership functions. When called without arguments it asks questions about the shape of the membership function to be constructed. Otherwise some parameters determining the membership function are given as arguments:

```
mfmake(type,variable_name,list_of_breakpoints,interpolation_type)
```

The function `rule` translates a linguistic rule (a string) to a fuzzy output function, for example

```
> r1:=rule('if v_lo or a_lo then p_hi'):
```

and `combine_rules` combines two or more fuzzy output functions according to (3) or (4), according to the user's choice. The function `defuzz` defuzzifies a combined fuzzy output function according to a user specified defuzzification method. Currently only different versions of COA are supported in FuzzyCAT.

`fuzzypplot` makes a plot of the gain function $c(e)$. If e is scalar then the plot is two dimensional and if $e \in \mathbb{R}^2$ it is three dimensional. Since the control laws often are not C^1 (see example 3.1) linear interpolation is used instead of the default cubic spline when making 2D-plots.

There is also a procedure `diffs_in_origin` which returns the derivative of $c(e)$ in the origin, i.e. $c'(0)$. If e is a vector then the partial derivatives of c in the origin are computed. This might be interesting, since these typically describe the parameters of an approximating linear PID-controller.

3 Analyzing an Example Using FuzzyCAT

In this section we study a simple fuzzy controller and use FuzzyCAT to derive an analytic expression for the control signal u as a function of the measured signal (plant output) y . This expression is then used to prove the stability of the closed loop system, as the controller is applied to a linear minimum phase plant.

Example 3.1 Let $\mathcal{P} = [-b, b] \subset \mathbb{R}$ and $u \in \mathcal{P}$. If the controller output membership function $\pi_{lo}(u)$ is associated with the statement “The signal u is low.”, we can have e.g.

$$\pi_{lo}(u) = \begin{cases} 1, & u < -5 \\ (5 - u)/10, & u \in [-5, 5] \\ 0, & u > 5 \end{cases} \quad (5)$$

and

$$\pi_{hi}(u) = \neg \pi_{lo}(u) = 1 - \pi_{lo}(u) \quad (6)$$

Suppose that the measured signal (plant output) $y \in [-a, a]$ has associated membership functions

$$\mu_{lo}(y) = \begin{cases} 1, & y < -1 \\ (1 - y)/2, & y \in [-1, 1] \\ 0, & y > 1 \end{cases} \quad (7)$$

and $\mu_{hi}(y) \equiv 1 - \mu_{lo}(y)$. Consider a fuzzy controller defined by two rules:

$$\text{IF } y \text{ is hi THEN } u \text{ is lo} \quad (8)$$

$$\text{IF } y \text{ is lo THEN } u \text{ is hi.} \quad (9)$$

Here follows a Maple session where FuzzyCAT is used for deriving an analytic expression for the control law generated by the rules above:

```
> read('fuzzycat.m'):
> ylo:=mfmake('lo',y,[-a,-1,1,a],'linear');

ylo :=

[[1, y = - a .. -1], [1/2 - 1/2 y, y = -1 .. 1], [0, y = 1 .. a]]

> type(ylo,membershipfunction);
true
> yhi:=Not(ylo):
> ulo:=mfmake('lo',u,[-b,-5,5,b],'linear'):
> uhi:=Not(ulo):
> r1:=rule('if ylo then uhi'):
> r2:=rule('if yhi then ulo'):
> rm:=combine_rules(r1,r2,max):
> rs:=combine_rules(r1,r2,add):
> cm:=defuzz(rm);

cm := [[1/6  $\frac{-25 + 3b^2}{b}$ , y = - a .. -1],
[1/6  $\frac{y^2(-3b^2 + 25y)}{b}$ , y = -1 .. 1], [-1/6  $\frac{-25 + 3b^2}{b}$ , y = 1 .. a]]

> cs:=defuzz(rs);

cs := [[1/6  $\frac{-25 + 3b^2}{b}$ , y = - a .. -1],
```

```

      2      2      2
      y (- 3 b  + 25 y )      - 25 + 3 b
[- 1/3 -----, y = -1 .. 1], [- 1/6 -----, y = 1 .. a]]
      2      2
      - 5 - 2 b + 5 y      b

> diffs_in_origin(cs);

      2
      b
[[y, -----]]
      - 5 - 2 b

> diffs_in_origin(cm);

[[y, - 1/2 b]]

> matlabfunc(cs,cssim);
> a:=2: b:=12:
> interface(plotoutput='cspplot.ps',plotdevice=postscript):
> fuzzyplot(cs):

```

The plot that was made by `fuzzyplot` is displayed in figure 3 and the file `cssim.m` produced by the call `matlabfunc(cs,cssim)` is displayed in figure 4. In this case the file `cssim` cannot be used directly in MATLAB since it contains parameters a, b . This is of course trivial to circumvent.

From the Maple computation we draw the conclusion that the rules (8) and (9) together with the membership functions (5), (6) etc. generate the control law $u = c(y)$, where

$$c(y) = \begin{cases} 5.65, & -2 \leq y < -1 \\ \frac{1}{72} (25 y^2 - 432) y, & -1 \leq y \leq 1 \\ -5.65, & 1 < y \leq 2 \end{cases} \quad (10)$$

in the case $a = 2, b = 12$, using COA-defuzzification with max-combination of fuzzy output functions. For these choices of a, b we have $c'(0) = -6$ so around the origin the fuzzy controller can be approximated with a P-controller with gain 6. In the sequel we write c_{max} for the gain function obtained with max-COA and c_{sum} for the one corresponding to sum-COA; i.e. c in formula (10) is c_{max} .

It is trivial to check in Maple that $c'_{max}(y) = \frac{25}{24} y^2 - 6$ in the interval $[-1, 1]$. Since $c'(1) = c'(-1) \neq 0$, c does not have a continuous derivative which may cause trouble in some applications. By keeping b unassigned in c_{max} we can see that $c \in C^1$ is only achieved by putting $b = 5$, whereas c_{sum} cannot be made C^1 by any choice of b !

It is interesting to note that the controller obtained is the simplest possible odd non-linearity (a polynomial with a linear and a cubic term) plus a saturation. Since it is not possible to define a fuzzy controller with one single rule [6] we proverbially conclude:

“The simplest possible fuzzy controller is the simplest possible nonlinear controller.”

□

Now that we have analytic expressions for the nonlinear control law defined by fuzzy logic we can use standard methods from nonlinear control theory to analyze the closed loop. Some suitable tools treated in most basic textbooks on nonlinear control are: the circle criterion, the Popov criterion, the describing function approach and Lyapunov theory [2, 10, 14].

Example 3.2 Let us apply the type of controller derived in the previous example to a linear plant and examine the stability properties of the closed loop system.

Consider the gain function $c(y)$ defined in equation (10). (Note that this is not the function displayed in figure 3.) Let us apply this control law to the linear non-minimum phase plant with transfer function

$$G(s) = 0.1 \frac{1-s}{(s+1)(s+2)} \quad (11)$$

It is easy to check that

$$\forall y : \quad 0 \leq \frac{-c(y)}{y} \leq 6 \quad (12)$$

The circle criterion thus assures closed loop stability if

$$\forall \omega : \quad \operatorname{Re} G(i\omega) > -1/6 \quad (13)$$

and it is straightforward to prove that the inequality (13) actually holds. We have proved that (10) is a stable control law for the system (11). Figure 5 shows the step-response of the closed loop system under a fuzzy control law and a linear P-controller with gain 6. Since there is no other P-controller which gives the closed loop system the same static gain we can conclude that the fuzzy controller performs better than the linear one for this system, but we should remember two things:

- As for all nonlinear controllers, the shape of the step response changes with the amplitude of the reference signal. The fuzzy controller will behave quite differently for other values of r .
- The main reason for the good performance of the fuzzy controller is the saturation present in c – it is simple to check via simulation that a saturating P-controller behaves almost exactly as the fuzzy controller.
- A linear PI-controller will outperform the fuzzy P-controller.

□

Figure 3: Gain function $c(e)$ in example 3.1.

```
function u=cssim(v)

% This file was created by the Maple function matlabfunc.
% Contains a fuzzy controller.
% (c) 1993 Anders Stenman
% -----

y=v(1);

if ( y >= -a ) & ( y <= -1 ),
    u = 1/6*(-25+3*b^2)/b ;
elseif ( y >= -1 ) & ( y <= 1 ),
    u = -1/3*y*(-3*b^2+25*y^2)/(-5-2*b+5*y^2) ;
elseif ( y >= 1 ) & ( y <= a ),
    u = -1/6*(-25+3*b^2)/b ;
end
```

Figure 4: MATLAB m-file defining the fuzzy controller in example 3.1.

Figure 5: Step responses for controlled non-minimum phase plant (example 3.2). Solid line: fuzzy controller, dotted: linear P-controller.

4 Future Extensions of FuzzyCAT

FuzzyCAT is likely to be extended in the following directions in future versions of the program:

Membership functions. It could be interesting to have more general types of membership functions even if the ones already covered probably are sufficient for most applications.

Other defuzzification methods. Some of the commonly employed defuzzification methods that do not use COA should be supported.

Stability analysis. It is easy to automatically check if a nonlinear P-controller of the above type together with a linear (rational) plant satisfies the circle criterion.

To conclude, we hope that some of the tools presented here contributes to the demystification of fuzzy control so that it may eventually become an accepted part of control engineering both in industry and academia, and receive a fair judgement and treatment from both.

Acknowledgement

This work was financially supported by the Swedish Council for Technical Research (TFR).

References

- [1] K.J. Åström and T.J. MacAvoy. Intelligent control: an overview and evaluation. In D.A. White and D.A. Sofge, editors, *Handbook of Intelligent Control. Neural, Fuzzy, and Adaptive Approaches*, chapter 1, pages 3–35. Van Nostrand Reinhold, 1992.
- [2] D. Atherton. *Nonlinear Control Engineering*. Van Nostrand, 1975.
- [3] S. Boverie, B. Demaya, and A. Titli. Fuzzy logic control compared with other automatic control approaches. In *Proc. 30th CDC*, volume 2, pages 1212–1216, Brighton, UK, 1991. IEEE.
- [4] B. Char, K.O. Geddes, G.H. Gonnet, B.L. Leong, M.B. Monagan, and S.M. Watt. *Maple V Language Reference Manual*. Springer, 1991.
- [5] D. Driankov, H. Hellendorf, and M. Reinfrank. An introduction to fuzzy control. To appear, 1993.
- [6] K. Forsman and J-E Strömberg. A short introduction to fuzzy control for control theorists; with annotated bibliography. Technical Report LiTH-ISY-I-1339, Dept. of Electrical Engineering, Linköping University, S-581 83 Linköping, Sweden, March 1992. Available by anonymous ftp 130.236.24.1.
- [7] K. Forsman and J-E Strömberg. Aspects of fuzzy control. Technical report, Dept. of Electrical Engineering, Linköping University, S-581 83 Linköping, Sweden, 1993. To appear.
- [8] The Math Works Inc. *SIMULINK User's Guide*, January 1991.
- [9] The Math Works Inc. *MATLAB Reference Guide*, August 1992.

- [10] J.E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice-Hall, 1991.
- [11] A. Stenman and K. Forsman. FuzzyCAT – a fuzzy control analysis tool for Maple. Technical report, Dept. of Electrical Engineering, Linköping University, S-581 83 Linköping, Sweden, 1993. To appear.
- [12] M. Sugeno. An introductory survey of fuzzy control. *Inform. Sci.*, 36(1-2):59–83, July-August 1985.
- [13] Togai Infralogic, Inc,. *TILShell User's Manual*, 1991. Version 1.3.0.
- [14] M. Vidyasagar. *Nonlinear Systems Analysis*. Prentice-Hall, second edition, 1993.
- [15] L.A. Zadeh. PRUF - a meaning representation language for natural languages. *Int. J. Man-Machine Studies*, 10:395–460, 1978.