Additional Mathematical Pre-processing for the Fuzzy Control of a Servodrive

DAN MIHAI
Department of Electrical Drives
University of Craiova
Blvd. Decebal 107, 200 440
ROMANIA

Abstract: -The paper deals with an analytical tool proposed as a pre-processing stage for the fuzzy control of a servodrive. Several relations are aimed to optimize both the correlation of the plant / controller parameters and the real-time implementation. This study is focused on positioning systems, where the fast response and the accuracy are very important. The results as simulation diagrams, on-line timing and main system variables recordings prove the utility of the relations and validate different fuzzy control algorithms: based on look-up table, with on-line inference and an adaptive version one.

Key-Words: - Fuzzy Control, Servodrive, Mathematical Pre-processing

1 Introduction

The most obvious advantage of the fuzzy control technique seems to be its simplicity, in term of the effort design. A quite general perception is that fuzzy control is a very fast way for implementing controllers with very good results for systems no having a model (or an accurate one) or / and not involving a certain precision. The author's experience brings some adjustments to this statement. A more or less systemic study is necessary for no matter what control algorithm is applied. The aim of the paper is to prove that adding an analytical / systemic study, a Fuzzy Logic Controller (FLC) is able to ensure a better quality for some high performance applications, like the precise and fast positioning systems. The basic improvements proposed here concern an off – line multicriterial conditioning analyze for the best correlation between the sampling rate, the controlled system parameters, the hardware restrictions and the overall performance, the combination of the pure fuzzy controller technique with non-fuzzy strategies, the control of the servodrive at a low speed in the final time segment so that the inertial effect would be minimized. This low speed is pre-computed and defines a "motion threshold".

Many of the proposed relations are available for other control algorithms too, optimizing parameters and data for servodrives with position and speed loops. The results are essential for low-cost microcontrollers and very useful even for high-end DSP-controllers, allowing a shorter control task and ensuring a good balance computation precision – computation speed.

2 The Target System Configuration and Basic Variables

The servodrive has an external position loop based on a hardware transducer (encoder) and an inner one for the speed, implemented in software - fig. 1, 2. The meaning of the notations is:

$Norm_i$: normalization blocs for each FLC entries; CPB: control processing blocks; PS: power supply; T_{gen}: torque generator; M: motor; En: encoder.

N_α^* and $N_{\alpha k}$: position set-point and actual position, in encoder pulses number;

$\varepsilon_{\alpha k}$, $\Delta\varepsilon_{\alpha k}$ and same variables with index n: position error, its variation referred to a sampling period T and their corresponding normalized values;

$\Delta N_{\alpha k}$: pulse encoder number delivered by encoder during T;

c_k, c_{kout}: the computed control and its outputted value;

k_{div}: division or multiplication factor for pulses from encoder, by additional hardware;

The input FLC variables are:

- the position error $\varepsilon_{\alpha k}$:

$$\varepsilon_{\alpha k} = N_\alpha^* - N_{\alpha k} \quad (1)$$

- the position error variation during T, $\Delta\varepsilon_{\alpha k}$:

$$\Delta\varepsilon_{\alpha k} = \varepsilon_{\alpha k} - \varepsilon_{\alpha k-1} = \Delta N_{\alpha k} \quad (2)$$

ω_k, the "software" speed, is computed by:

$$\omega_k = \frac{2\pi \cdot k_{div} \cdot \Delta N_{\alpha k}}{N_{p/r} \cdot T} = c_{sp} \cdot \Delta N_{\alpha k} \quad (3)$$

$N_{p/r}$ means the encoder resolution in pulses number per revolution. (2) and (3) reveals a physical meaning of $\Delta\varepsilon_{\alpha k}$, useful for an easy identification of the fuzzy rules table.

3 The Real-Time Control Tasks

Fig. 3 gives the image of the on-line timing for the main tasks. INT0 is a high level priority interrupt generated by encoder pulses (the falling edge). The low level priority interrupt is software generated, marking the sampling period T. PULSE are the encoder signal, ALG means the control algorithm processing and MAIN concerns the main program interrupted. This strategy ensures a fast response of the controller to the most important system variable: the position. Each encoder pulse requires an up-dating of the FLC input variables.

4 Parameters / variables correlation and processing

T must be correlated with - [7]:

a. The specific time constants of the plant (a standard condition);
b. n_{min} - the minimum accepted value of the real speed for which $\omega_{k\ soft}$ is detectable;
c. l_w - the size of the word (register) for ω_k ;
d. c_f - the fineness coefficient for speed;

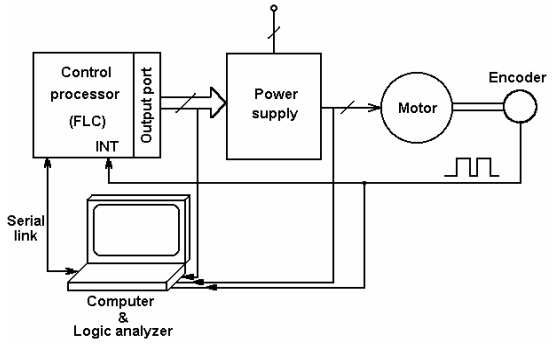

Fig 1. The system structure

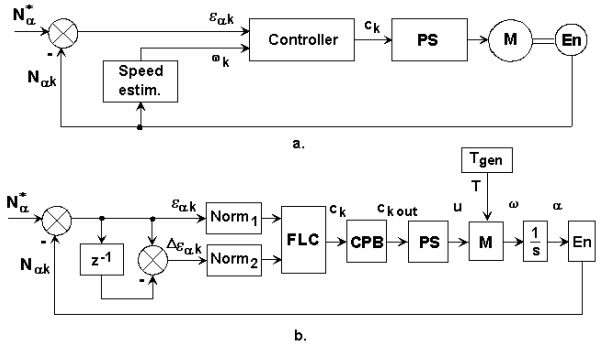

Fig. 2 The processing blocks (a) with their details (b)

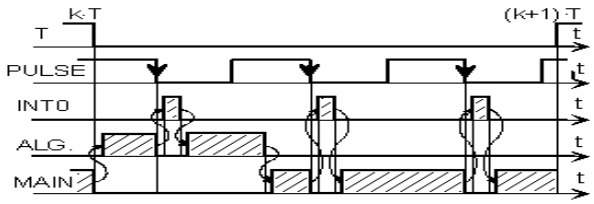

Fig. 3 The timing for on-line control based on interrupts

e. The duration of on-line processing;
f. The amount of memory reserved for on-line recordings in microcontroller NVRAM.

b. and **c.** give next restrictions for T:

$$\frac{60 \cdot k_{div}}{N_{p/r} \cdot n_{min}\ [RPM]} \le T \le \frac{(2^{n_{lw}} - 1) \cdot 60 \cdot k_{div}}{N_{p/r} \cdot n_{max}\ [RPM]} \quad (4)$$

d. A too big value for c_f may imply a speed loop ringing. The ideal value is given by the condition that the maximum number for software values of speed N_{smax} fills the whole speed register:

$$N_{s\,max} = \frac{N_{p/r} \cdot T \cdot n_{max}[RPM]}{2\pi \cdot k_{div}} \cdot \frac{2\pi}{60} = 2^{n\ reg} - 1 \quad (5)$$

$$c_{\ f\ ideal\ (min)} = \frac{2\pi}{60} \cdot \frac{n_{max}[RPM]}{2^{nreg} - 1} \quad (6)$$

e. The designer must check that the processor is able to perform all computations at motor maximum speed. The hardware interrupt INT0 requested by the falling edge encoder pulses is a fast task which mainly computes the position error $\varepsilon_{\alpha k}$ and ΔN_k. The minimum interval between 2 successive interrupts:

$$t_{min\ INT0} = \frac{60 \cdot k_{div}}{n_{max}[RPM] \cdot N_{p/r}} \quad (7)$$

During T, the maximum call number for INT0 is:

$$N_{max\ calls\ INT0} = \frac{T}{t_{min\ INT0}} \quad (8)$$

Δt_{INT0} is the time necessary to perform the on-line processing for INT0. The maximum speed for which the controller is able to perform the processing is:

$$n_{max} \le \frac{60}{\Delta t'_{INT0}} \cdot \frac{k_{div}}{N_{p/r}} \quad [RPM] \quad (9)$$

$\Delta t'_{INT0}$ includes the additional delay for an interrupt answer. The total time allocated to INT0 interrupts during T is:

$$\Sigma \Delta t_{INT0}\ /T = \frac{T \cdot n_{max} \cdot N_{p/r}}{60 \cdot k_{div}} \cdot \Delta t'_{INT0} \le k_1 \cdot T \quad (10)$$

$$k_1 = 0.2 \div 0.3$$

(10) means that the most part of T must be available for tasks related with the control algorithm, auxiliary tasks (saving and displaying data) and the main program. k_2 is the fraction of T allocated to these tasks. Because some non-lucrative time (calls, settings), $k_2 < 1 - k_1$. With f_{INT0} - the encoder pulses frequency,

$$\frac{T}{\frac{1}{f_{INT0}}} \cdot \left(\frac{1}{f_{INT0}} - \Delta t_{INT0} \right) \ge k_2 \cdot T \quad (11)$$

$$f_{INT0} = \frac{N_{p/r} \cdot n}{60 \cdot k_{div}} \le \frac{1 - k_2}{\Delta t_{INT0}};\ n \le \frac{60 \cdot k_{div}}{N_{p/r}} \cdot \frac{1 - k_2}{\Delta t_{INT0}} \quad (12)$$

ΔN_k being stored into a register with n_{reg} bits, another condition is:

$$\frac{T}{t_{INT0\,min}} = \frac{T}{\frac{60 \cdot k_{div}}{n \cdot N_{p/r}}} \le 2^{n_{reg}} - 1 \quad (14)$$

$$n_{max} \le \frac{(2^{n_{reg}} - 1) \cdot 60 \cdot k_{div}}{N_{p/r}} \quad [RPM] \quad (15)$$

The designer must also have an evaluation of the length for the interrupt routines (INT0 and ALG. - fig. 3, written in assembler) as maximum instruction number. The software interrupt routine associated with the control algorithm processing being called by a timer, this routine will be noted with T2. With $\Delta t_{av.instr.}$-the average time for a processor instruction:

$$N_{max\ INT0\ instr.} \approx \frac{\Delta t_{INT0}}{\Delta t_{av.\ instr.}} \quad (16)$$

$$N_{max\ T2\ instr.} \approx \frac{T - \Sigma \Delta t_{INT0}}{\Delta t_{av\,/\,instr.}} \le \frac{k_2 \cdot T}{\Delta t_{av\,/\,instr.}} \quad (17)$$

$$N_{max\ T2\ instr.} \le T \cdot \left(1 - \frac{n_{max} \cdot N_{p/r} \cdot \Delta t_{INT0}}{60 \cdot k_{div}}\right) \cdot \frac{1}{\Delta t_{av.\ instr.}} \quad (18)$$

f. The data memory space for all on-line recordings is given by the number of data bytes n_{Bytes} saved for each T and the regime duration Δt_{pos}. For avoid A an outrunning of the available memory $V_{data\ mem.}$:

$$T \ge \frac{\Delta t_{pos.\ max} \cdot n_{Bytes}}{V_{data\ mem.}} \quad (19)$$

4.1 Look-Up-Table based FLC

The most reduced on – line computational effort is obtained by means of a look-up table (LUT) filled off - line. Fig. 4 gives the image of the equivalent systemic structure. The normalized variables are:

$$\varepsilon_{\alpha n\,k} = (N^* - N_k) \cdot \frac{\varepsilon_{\alpha n\,k\,max}}{N^*} \ge 0 \quad (20)$$

$$\Delta\varepsilon_{\alpha n\,k} = \frac{\Delta N_k \cdot c_{sp} \cdot \Delta\varepsilon_{\alpha n\,k\,max}}{\Omega_{max}} \le 0 \quad (21)$$

When the LUT has N_L lines, the address of the control c_k is found on-line easily by:

$$Adr_{ck} = Adr_{Base\ LUT} + \varepsilon_{\alpha n\,k} \,.\, N_L + \Delta\varepsilon_{\alpha n\,k} - \Delta\varepsilon_{\alpha n\,k\,min} \quad (22)$$

The control values, computed off-line by the Mamdani inference method, were stored by concatenation of columns. This matrix has the final control values (no more necessary a CPB block). For the drive fed from a power supply delivering the range $[U_{min}, U_{max}]$, the motor voltage is:

$$U_M = \frac{U_{PS\,max}}{2^{n_{DAC}} - 1} \cdot (P_{out}) - U_D \quad (23)$$

PS includes a digital analog converter with n_{DAC} bits. (P_{out}) means the word sent to the output port of the microcontroller. U_D must be subtracted outside the power supply (by diodes) allowing to U_M to vary from zero, or, anyway, to be above the sensibility voltage level of the motor.

The fuzzy sets and the 3D surface control are presented by the fig. 5, with: **az**: almost zero; **vs**: very small; **s**: small; **qs**: quite small; **m**: medium; **b**: big; **vb**: very big. A FLC tuning leads to a non-uniform distribution of the fuzzy sets. According to the interface for the power supply, the control is computed directly for the necessary range (7 bits).

4.2 An adaptive Look-Up-Table based FLC

In the context of a wide interest in the adaptive fuzzy control - [6], [10], the author brings the next ideas, related to the control of the servodrive:

- the on - line computation of the threshold control $c_{k\,th}$ (allowing a motion without mechanical inertia, just above the limit for slow continuous speed) for a current (variable) load;
- the on-line computation of an adaptation factor F dependent of the load too and its multiplication with the control delivered by a reference LUT with values for no load conditions.

Under a limit space, c_k delivered by the LUT is sent to the motor. After that, basically, for $c_k \ge c_{k\,th}$, $c_{kout} = c_k$, otherwise $c_{kout} = c_{k\,th}$.

With a selected threshold speed Ω_{th}, according the inertia of the drive, the associated control c_{kth} is –[8]:

$$c_{k\,th} = \frac{2^{n_{DAC}} - 1}{U_{PSmax}} \cdot \left(R \cdot i_k + C_e \cdot \Omega_{th} + U_D\right) \quad (24)$$

C_e - the flux constant; T_e - electromagnetic time constant; R - motor resistance (DC rotor disk type).

The current i_k is an image of the motor torque and is estimated with the next relation set:

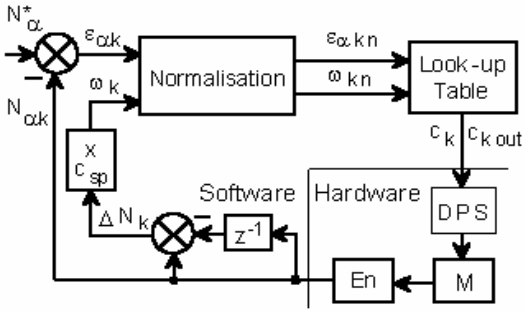

Fig. 4 Fuzzy control loop – LUT variant

$$\omega_k \cong c_{sp} \cdot \Delta N_k$$

$$i_k = \frac{u_k}{R \cdot \left(1 + \frac{T}{T_e}\right)} + \frac{i_{k-1}}{1 + \frac{T}{T_e}} - \frac{C_e \cdot \omega_k}{R \cdot \left(1 + \frac{T}{T_e}\right)} - \frac{U_D}{R \cdot \left(1 + \frac{T}{T_e}\right)} = a \cdot u_k + b \cdot i_{k-1} + c \cdot \omega_k + d \quad (26)$$

i_k is dependent of the system inertia indirectly, because of the correlation with ω_k. The factor F:

$$F = 1 + \frac{i_k - I_{min}}{I_{max} - I_{min}} \cdot \left(\frac{I_{max} \cdot c_{th\,min}}{I_{min} \cdot c_{th\,max}} - 1 \right) \quad (27)$$

for $I_{min} \leq i_k \leq I_{max}$

$F = 1$ for $i_k < I_{min}$

$F = F_{max}$ for $i_k > I_{max}$

$c_{th\,min}$ and $c_{th\,max}$ are the threshold controls for minimum and maximum load. i_k is the single variable computed on-line, all other variables from rel. (27) being prepared off-line.

4.3 On-line inference fuzzy control

FLC makes in real-time 3 essential operations: fuzzification, fuzzy inference and defuzzification. Fig. 6 gives the image of 7 overlapped triangular fuzzy sets and the 7 characteristics points (A…G). f_M is the maximum membership function (MBF). In this real-time algorithm, a scaled value for f_M will be used, more appropriate for the control processor. For a standard range on 8 bits, f_M =255. All in/out variables must be normalized in this range.

4.3.1 The fuzzifier

The inputs of the fuzzifier are computed as:

$$\varepsilon_{\alpha kn} = \varepsilon_{\alpha k} \cdot \frac{2^8 - 1}{\varepsilon_{\alpha max}} = \left(1 - \frac{N_{\alpha k}}{N^*_\alpha}\right) \cdot (2^8 - 1) \quad (28)$$

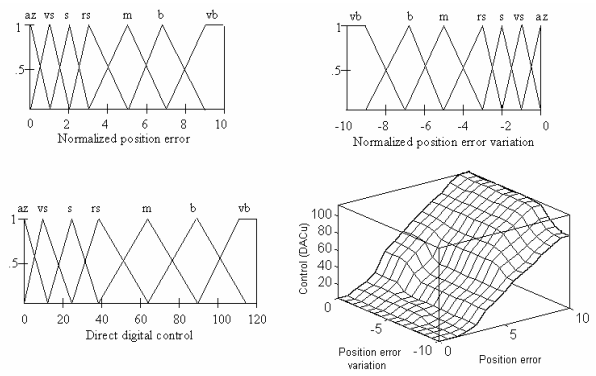

Fig. 5 An image of the fuzzy sets and the surface control

$$\Delta\varepsilon_{\alpha kn} = \Delta\varepsilon_{\alpha k} \cdot \frac{2^8 - 1}{\Delta\varepsilon_{\alpha max}} = \frac{\omega_k}{\omega_{max}} \cdot (2^8 - 1) \quad (29)$$

For each V_x, the neighbor fuzzy sets to that it belongs and their membership function values (MBF) must be computed:

$$f_{S_i}(V_x)\Big|_B = f_M \cdot \frac{V_{i+1} - Vx}{V_{i+1} - V_i} \quad (30)$$

$$f_{S_{i+1}}(V_x)\Big|_A = f_M \cdot \frac{V_x - V_i}{V_{i+1} - V_i} = f_M - f_{S_i}(V_x) \quad (31)$$

$f_{Sn}(V_x) = f_M$ for $V_x \geq V_n$

For a fast on-line implementation of (28):

$$\varepsilon_{\alpha kn} \approx \frac{\varepsilon_{\alpha k}}{N^*_\alpha} \cdot 2^8 \approx \varepsilon_{\alpha k}\Big|_H \cdot \left[\frac{255}{N^*_\alpha}\right] \approx \frac{\varepsilon_{\alpha k}}{N^*_\alpha\Big|_H} \quad (32)$$

The last form is optimal, for the computation accuracy and the on-line effort. For an easy MBF on-line computations, from (30):

$$f_{S_i}(V_x) = \left[\frac{255}{V_{i+1} - V_i}\right] \cdot (V_{i+1} - Vx) \quad (33)$$

$$f_{S_i}(V_x) = 256 \cdot \frac{V_{i+1} - Vx}{V_{i+1} - V_i} \quad (34)$$

(34) is a very fast one (x 256 is realized by a simple byte shifting), but the best compromise accuracy – computation effort is done by (33).

4.3.2 The fuzzy inference block

The rules are arranged into a matrix - fig. 8a. This data is re-organized for the storage in FLC memory by rows concatenations, beginning with a base address - BA. The Fuzzy Inference Bloc (FIB) receives 8 inputs from the 2 fuzzifiers (one per FLC entry). The first 4 gives the framing fuzzy sets and the next 4 are MBF values. FIB delivers 8 outputs; the first are code conclusion values, maximum 4. If V and ΔV belong each of them to 2 neighbor fuzzy sets, then 4 rules will be in action, denoted by R1, R2, R3 and R4 - fig. 8a. It is posible to have 1, 2 or 4 fired rules. For V framed in the j^{th} set and ΔV in the i^{th} set, only the R1 rule is active (conclusion C1 is the result). For V framed in the j^{th} and $(j+1)^{th}$ sets and ΔV in the i^{th} and $(i+1)^{th}$ sets, all the R1, R2, R3

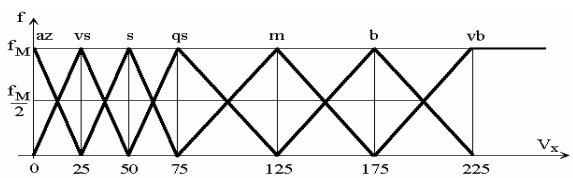

Fig. 6 The fuzzy sets for the application

and R4 rules are active (C1, C2, C3 and C4 are the result). If less than 4 rules are active, the other 'Code Set' values are null. For all cases, R1 is active so this rule will be stated as the 'Basic Rule' - BR. Degrees of fulfillment for each rule / conclusion in action, obtained by a specific fuzzy reasoning (Mamdani, Larsen, Sugeno, Tsukamoto etc.), is denoted with MBF Ri. For the actual inputs, with a framing (i, i+1) and (j,j+1), the first task of the FIB is to find the memory address for the conclusion codes of the fired fuzzy rules - a fast computation of the displacements added to BA:

R1: Disp(R1)=(j+1)xm+i; R2: Disp(R2)=Disp (R1) + 1

R3: Disp(R3)=Disp(R1)+m;R4:Disp (R4)=Disp (R3)+1

The second task is to fetch from the above address the memorized code conclusion. The last task of FIB is to make foro each a MBF value, minimizing / multiplying MBF values for the entry framing sets (Mamdani / Larsen inference).

4.3.1 The defuzzifier

One of the most intensive computational task associated with a FLC is the defuzzification. From many methods: COG, COA, MOM, LOM, ROM, COM, the Center of Maximum (COM) method is the most appropriate for the closed-loop control – [9]. The principle of COM method is illustrated by fig. 9. The representative values V_i in the conclusion sets for the fired rules are those bolded. With the degrees of fulfillment of the i conclusion denoted by f_i, the crisp value delivered is:

$$c_0 = \sum_{i=1}^{4} V_i \cdot f_i \Big/ \sum_{i=1}^{4} f_i \qquad (35)$$

Codes set i are the number 1…7 of the fired rules (0 if is not in action) and MBF i are the degree of fulfillment for the fired rules. For a low-cost control processor (with an 8-16 bits arithmetic), rel. (35) may request a too complex division routine. Indeed:

$V_i \leq 0FF\ H;\ f_i \leq 0FF\ H; \Rightarrow V_i \times f_i \leq 0FFFF\ H \qquad (36)$

$\Sigma V_i \times f_i$ can exceed 16 bits. A 24 by 16 division routine could be avoided through:

$$c_0 = \left(\sum_{i=1}^{4} V_i \cdot f_i \Big/ 4 \right) \Big/ \left(\sum_{i=1}^{4} f_i \Big/ 4 \right) \qquad (12)$$

The arithmetic processing is reduced to: simple operations associated with direct processor instructions (multiply); a maximum 16 bits range for all data; a very fast addition routine 16 + 16 bits; a very fast division routine 16 / 4 bits (by shifting); a single call routine for a division 16 /8 bits.

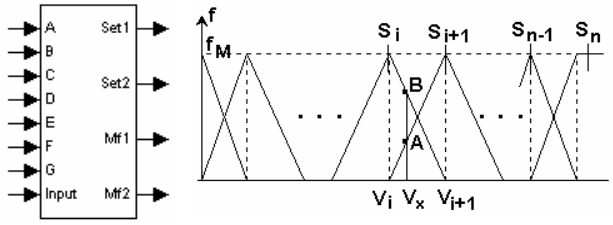

Fig. 7 The main elements of the fuzzifier

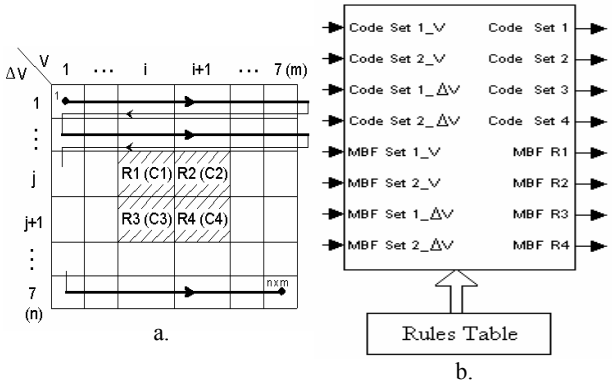

Fig. 8 The main elements of the inference bloc

5 Application results

The experiments concerned a rotor disk motor having: U_N = 12 V; I_N = 13 A; P_N = 74 W; n_N = 300 RPM, R_a = .97 Ω; L_a = 0.246 mH; J = .0011 kgm^2; T_{em} = 50.2 ms; T_e = .253 ms. The parameters set: T = .002456 s, c_{sp} = 1; $N_{p/r}$ = 2500; k_{div} = 1 represents a multicriterial optimum - [7]. Fig. 10 gives the system evolution for a LUT-based FLC. Fig 12 is a real-time analyzer recording, with T (SPER), encoder pulses (PULSE), the hardware interrupt routine (INTO) and the main fuzzy tasks. This diagram reveals the ability of the designed FLC to manage the system, gives precise real data on the timing and makes possible a hardware / software debugging. Fig. 11 shows the results for using the adaptive factor F (FLUT) (setpoint position: 3750 pulses -1.5 turns; and the recording time: 2.5 sec.). ckth is the threshold control for the actual load; m err, n varer – integer index LUT lines / columns). It can be seen the high fidelity of estimated current (Iest), including for the dynamic regime. The position error variation (Er. var.) has really the mirrored speed variation. With FLUTout = FLUTc(computed), the positioning time (when Pos.

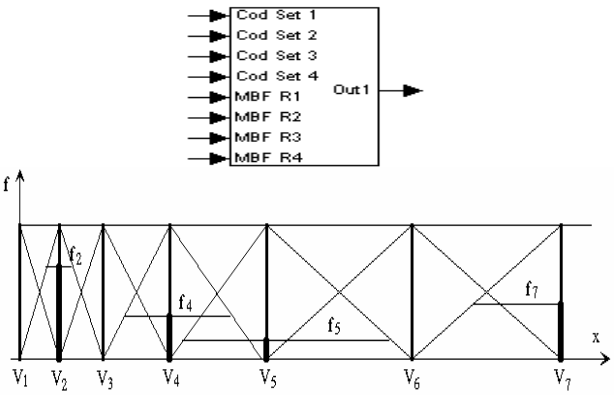

Fig. 9 The main elements of the defuzzifier

er.– position error becomes null) decreases to 60 % when the LUT basic controls are adapted to the load level. The threshold control is modified too in the presence of the adaptive factor, the corresponding voltage diminishing the positioning time. T (2.456 ms) is very appropriate, being very closed to the values for dt – integration step size, acting like a T in simulation, (vertical scale is 6 ms). The effect of control processing is marked by the difference between the voltage shape and the basic control taken from LUT (ck lut).

6 Conclusions

The application of fuzzy logic technique, stated sometimes as simple and fast, must be done after a rigorous study and a data flow and timing optimization. It is possible to have an accurate control of a servodrive Some optimized computations proposed by the author, have been confirmed by real-time recordings in various conditions. The real-time algorithm presented for managing the system has portability and can ensure a good overall control. The study concerned also original hardware and software techniques for debugging.

References

[1] Bogumila Mrozek, Mrozek Z., Modelling and Fuzzy Control of DC Drive, *ESM 2000*, Ghent, pp186-190.

[2] Debowski A., Armature - Current Fuzzy Control for DC Motor Drives, *PEMC*, Warsaw, 1994, pp. 136 – 141.

[5] Ibbini M.S., Jafar A.S., Self-Tuning Fuzzy Logic Controller for a Series DC Motor, Proc. *(369) Power and Energy Systems - 2002* .

[6] Kung C.-C., K.-H. Su, Adaptive Fuzzy Position Control for Electrical Servodrive via Total-Sliding-Mode Pechnique, *IEE Proc.-Electr. Power Appl*., Vol. 152, No. 6, Nov. 2005, pp. 1489-1502.

[7] Mihai D., Multicriterial Conditioning of a Digital Control Problem for a Positioning Electrical Drive System", *Annals of University of Craiova*, 1999, pp. 129-140.

[8] Mihai D., An Adaptive Fuzzy Control Strategy for a Precise Positioning System, *Proc. PCIM*, Nurnberg, 2000, pp. 301-306.

[9] Mihai D., A Fast Algorithm for Fuzzy Inference with 1, 2 or 4 Fired Rules by Standard Microcontrollers. Application on Control of Servodrives", *Proc. SCI* 2003, Orlando, FL, vol VIII, pp. 258-263.

[l0] Ordonez R., Zumberge J., Spooner J. T., Passino K. M., Adaptive Fuzzy Control: Experiments and Comparative Analyses, *IEEE Transactions on Fuzzy Systems*, Vol. 5, No. 2, May 1997, pp 167-188.

[l1] Passino K. M., Yurkovich S., *Fuzzy Control.* Menlo Park, CA Addison-Wesley, 1998.

[12] Pedrycz W., *Fuzzy Control and Fuzzy Systems, Second, Extended Edition*, John Wiley and Sons Inc., 1993.

[13] Silveura, P. E., Souza JR., R. de, Biazotto, V. M., Speed Control of an Autonomous Mobile Robot: Comparison between a PID Control and a Control Using Fuzzy Logic. *J. Braz. Soc. Mech. Sci*., May 2002, vol.24, no.2, pp.127-129.

[14] Vas P., Chen J., Stronach A. F.: "Fuzzy Control of DC Drives", *Intelligent Motion*, Nürnberg, 1994

[15] P. Vas: *Artificial– Intelligence-Based Electrical Machines and Drives*, Oxford University Press, New-York, 1999

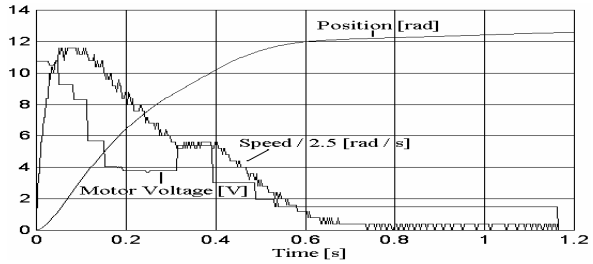

Fig. 10 Fuzzy control of the servodrive – real-time recording of the main variables.

Fig. 11 The positioning results with the adaptive factor for the fuzzy control – simulation results

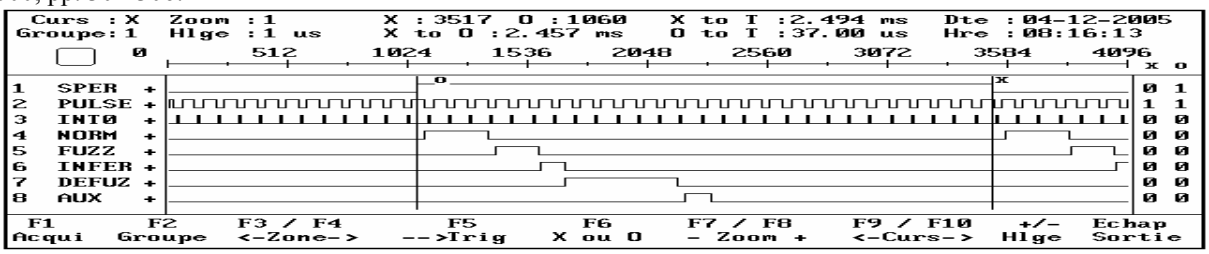

Fig. 12 The real-time timing for the servodrive with fuzzy position closed-loop