Fuzzy systems - project 1

Christophoros Bekos (mpekchri@auth.gr)

October 1, 2017

1 Introduction

In this project we are about to simulate a fuzzy system which has two inputs, x1 and x2 and one output y . The domain for x1 is X1 = $[0, 1]$, for x2 is X2 = $[0, 50]$ and for the output y $\in$ Y = $[0, 1400]$. The domain of x1 is divided in two fuzzy sets S(small) and L(large) ,where S = $trap - MF(x1; 0, 0, 0.25, 0.75)$ and L = $trap - MF(x1; 0.25, 0.75, 1.0, 1.0)$. The domain of x2 is also divided in two fuzzy sets ,just like before : S = $trap - MF(x2; 0, 0, 10, 40)$ and L = $trap - MF(x2; 10, 40, 50, 50)$ .The domain of y is described by four sets : $B1 = tri - MF(y; 0, 0, 11), B2 = tri - MF(y; 0, 11, 130), B3 = tri - MF(y; 11, 130, 800)$ and $B4 = trap - MF(y; 130, 800, 1400, 1400)$.
The following rules are given :

- Rule R1 : IF x1 is S AND x2 is S ,THEN y is B1
- Rule R2 : IF x1 is S AND x2 is L ,THEN y is B3
- Rule R3 : IF x1 is L AND x2 is S ,THEN y is B2
- Rule R4 : IF x1 is L AND x2 is L ,THEN y is B4

Rules are implemented with the Larsen operator (R_p)
The binder AND is implemented with t-norm min operator ($\wedge$)
The binder ALSO is implemented with max operator ($\vee$)
We will implement the Base deduction method and our synthesis operator will be max-product operator.

2 Implementation of the rules R_i (i = 1,2,3,4)

2.1 How to implement the rules

Since we implement the base deduction method ,assuming we have two inputs A'_1 and A'_2 we must first compute the degree of occupancy (DOF) $w_{r_k} = w_1 \wedge w_2$ for each rule (k = 1,...,4) .
Each one of w_i's can be computed as $w_i = \vee(\mu_{A'_i}(x_i) * \mu_{A_i}(x_i))$ *(max-product operator)* .
For example , $w_{r_1} = w_1 \wedge w_2$,where $w_1 = \vee(\mu_{A'_1}(x_1) * \mu_S(x_1))$ and $w_2 = \vee(\mu_{A'_2}(x_2) * \mu_S(x_2))$.
On the other hand, if we are interested in rule R2 will get :
$w_{r_2} = w_1 \wedge w_2$, where $w_1 = \vee(\mu_{A'_1}(x_1) * \mu_S(x_1))$ and $w_2 = \vee(\mu_{A'_2}(x_2) * \mu_L(x_2))$.
After all , for each rule r_i we may write : $\mu_{B'_{r_i}}(y) = w_{r_i} * \mu_{B_{r_i}}(y)$ *(because we use Larsen operator)* and finally the result would be :

$$\mu_{B'}(y) = \vee(\mu_{B'_{r_i}}(y)) \text{ for each i} \in (1, 4) \text{ } \textit{(using max operator for ALSO)}$$

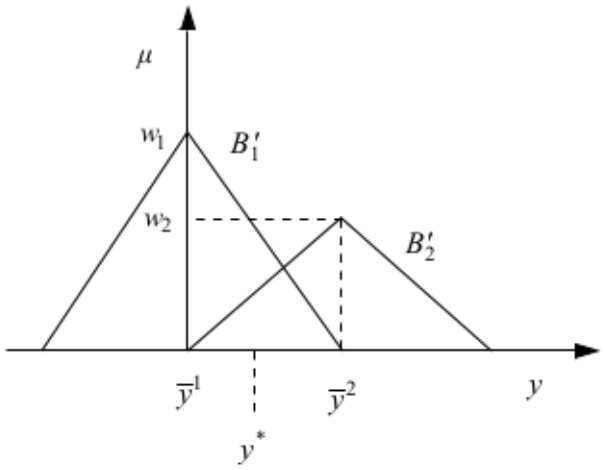
2.2 Defuzzifier CAD

In many cases we need a singleton output instead of a fuzzy set ,so since we compute $\mu_{B'}(y)$, we must use a defuzzifier in order to retrieve the singleton output .
CAD is simple ,yet powerful defuzzifier , in the sense of being able to compute values close to the

more complex COA (Center of Area defuzzifier) .
In order to implement CAD we need to compute :

$$y^* = \frac{\sum_{i=1}^{N} y_i * ww_i}{\sum_{i=1}^{N} ww_i}$$

where : y^* is the singleton output, N is equal to number of rules (in this project 4) ,
y_i is center of mass of $\mu_{B_i'}(y)$ and ww_i is equal to maximum value of $\mu_{B_i'}(y)$.
An example of CAD implementation in a two fuzzy set system is given in the image bellow .



In order to compare CAD results i also implemented an COA defuzzifier . COA defuzzifier actually computes the center of mass of the output fuzzy set . Computation of center of mass is done by Jered Wells matlab function , which can be found here : https://www.mathworks.com/matlabcentral/fileexchange/41675-center-of-mass .

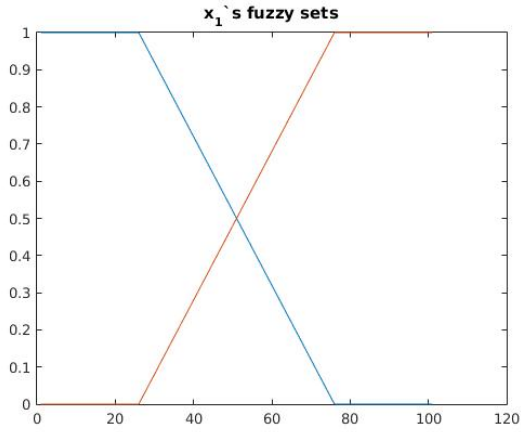
3 Matlab script

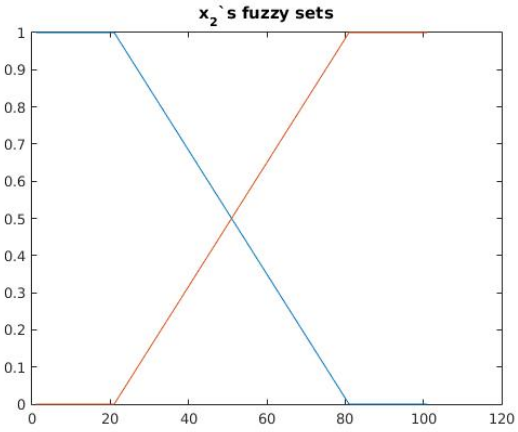
The script 'main.m' is used in order to simulate our rules , and their output when we have two fuzzy input sets : A_1' = MORE OF LESS (Large) - *named X1-in in script* and A_2' = VERY (Large) - *named X2-in in script* . Fuzzy sets for x_1 variable are being represented in script by X1-small and X1-large , while x_2's fuzzy sets by X2-mall and X2-large.
As said before , script first finds the DOF of each rule for these inputs and then calculates the individual conclusions of each rule. Finally we find the union (ALSO operator) of these conclusions and we compute the final fuzzy output .
Since in most circumstances we need a singleton value as output, a defuzzifier is used to convert the output fuzzy set in a single value.
CAD defuzzifier computes 80.0684 as output , while
COA defuzzifier computes 82.0057 as output .
As expected CAD's results are way too close to COA's results although CAD requires considerably fewer calculations .

4 System's behavior

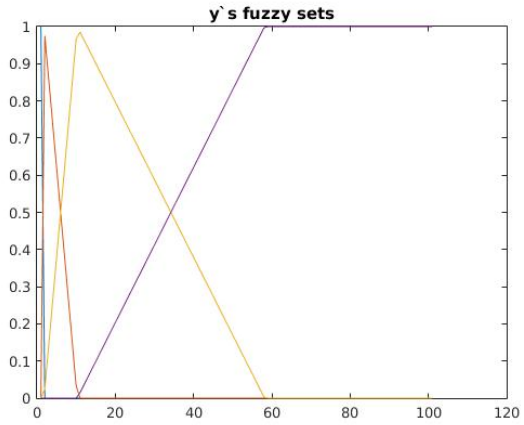
In order to better understand how our system operates we will examine in more detail the effect of A_1' and A_2' inputs (as defined before) .

Fuzzy sets for variables x_1 and x_2 look like :

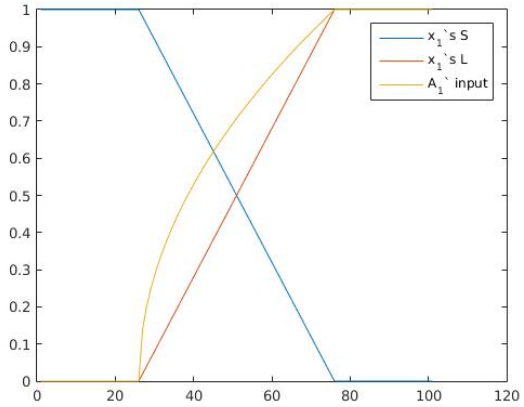


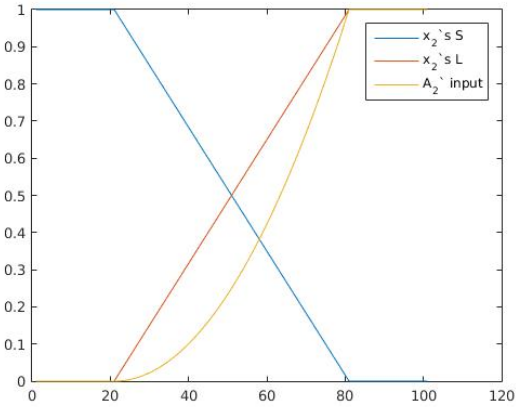


Fuzzy set for output variable y are :

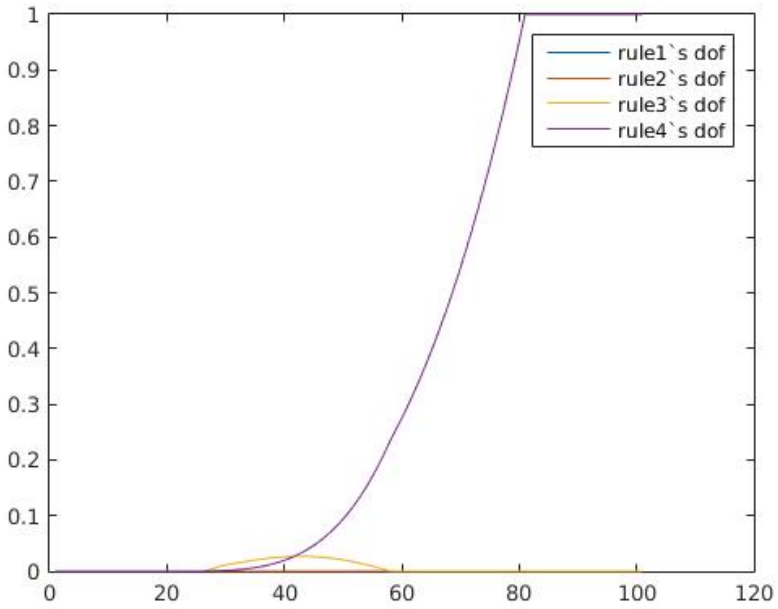


Fuzzy inputs A_1' and A_2' as well as x_1 and x_2 look like :

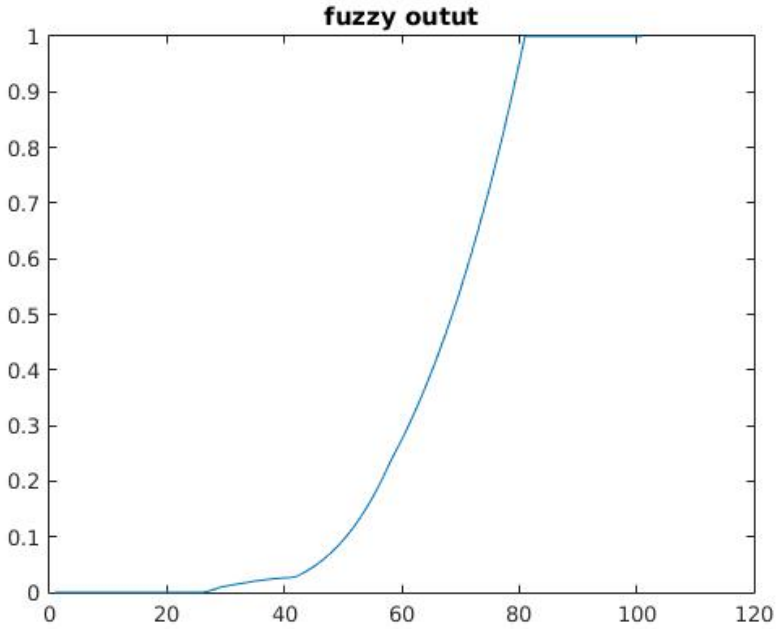




As expected (from both the figures above and script's results) , our system produces the following DOFs for each rule :



Finishing , our fuzzy output is an union of each DOF and it's described by the following figure :



CAD defuzzifier computes 80.0684 as output , while
COA defuzzifier computes 82.0057 as output ,
which are completely reasonable results .

5 Notes

- Variables x_1 and x_2 represent mass m and velocity u, respectively , while our output y represents kinetic energy E_k ($E_k = \frac{1}{2}mu^2$) . The given rules were created empirically .
  We observe that we are able to produce results similar to those of a strict mathematical equation, using fuzzy rules .
- Of course all those rules could be better defined and plotted using fuzzy toolbox (in matlab) , but it wasn't used in order to gain a better understanding of how fuzzy systems are defined and how results are being computed using fuzzy rules .