Fuzzy systems - control of a dc motor using FLC

Christophoros Bekos (mpekchri@auth.gr)

September 29, 2017

1 Problem statement

In this project we will simulate a fuzzy logic controller (FLC) using matlab's fuzzy toolbox . The system to be monitored is a dc motor with independent excitation. Our motor's state equations and parameters are given below :

$$L_a \frac{di_a}{dt} + R_a i_a + K\omega = V_a \qquad R_a = 44\Omega \qquad B = 8\times10^{-5}\ Nms/rad$$

$$L_a = 0.1H \qquad K = 0.64\ Vs/rad$$

$$J\frac{d\omega}{dt} + B\omega + T_L = K_M i_a \qquad J = 8\times10^{-4}\ Kgm^2 \qquad K_M = 0.64\ Nm/A$$

Those parameters lead us to the following system :

$$\Omega(s) = \frac{8000}{s^2+440.1s+5146}V_a(s) - \frac{1250(s+440)}{s^2+440.1s+5146}T_L(s)$$

In our approach we will not consider the most negative poles , so the system we will deal with becomes :

$$\Omega(s) = \frac{18.69}{s+12.064}V_a(s) - \frac{2.92}{s+12.064}T_L(s)$$

The supply voltage V_a is the control input to the system while the load torque T_L is considered as a type of disorder. The purpose of the system control procedure is that ω *(angular velocity of the cursor)* is kept constant or slightly affected by the load torque T_L.
In this design we will study only V_a 's effect , so we make the assumption that our system is described as :

$$\Omega(s) = \frac{18.69}{s+12.064}V_a(s)$$

Our task is to design a fuzzy controller in a way that the following conditions are met :

- For a cyclical frequency of disturbances, the gain of disturbance must be at most 20 db
- Maximum 5% elevation when input is step function
- Position error equal to zero
- $t_r \leq 160$ msec
- $V_a(t) \leq 200$ V for each t>0

Our sampling interval is given as T = 0.01 sec .
FLC will have two inputs , E (error), $\dot{E}$ (error's change) and one output $\dot{U}$.
After the FLC's $\dot{U}$ is normalized to u(k) ,u is given as input in the system described by $\Omega(s)$ and the result is $\omega(k)$.In addition ω and r (reference signal) must obey the restrictions :

$$\omega_{max} = 150 \text{ (rad/sec) and r} \in [0,150] .$$

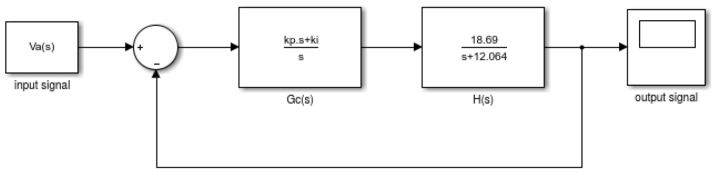
2 Solution steps

We will follow the steps :

- theoretical analysis and description of the PI controller
- design a PI controller in simulink and check if conditions are met
- design a fuzzy PI controller
- tune the fuzzy PI controller in order to fit our needs

3 Theoretical analysis

The following picture , where $H(s) = \frac{18.69}{s+12.064}$ and $G_c(s) = \frac{k_p(s+c)}{s}$, $c = \frac{k_i}{k_p}$ describes our PI controller :



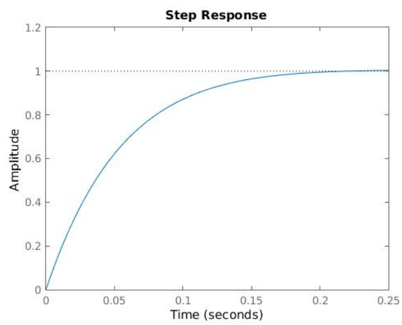
Solving by hand we get the following restrictions for k_p and k_i :

- restriction [1] : $k_p\ \sqrt{c^2+1} = 6.4548$
- restriction [2] : $0.2\ (\ 145.5 + 349\ k_p^2 + 451\ k_p\ ) + 54\ k_i \leqslant 0$
- restriction [3] : restriction for zero position error is fulfilled through the use of PI controller
- restriction [4] : $\frac{k_p}{k_i} \leqslant 160 * 10^{-3}$
- restriction [5] : $\lim\limits_{s\to \inf} \frac{150*k_p*(s+c)}{s^2} \leqslant 0 \implies k_p \leqslant \frac{1}{3}$

A choice of $k_p = 1$ and $k_i = 13.5$ seems to agree with all the restrictions , so we will define the PI controller's transfer function as $G_c(s) = \frac{s+13.5}{s}$

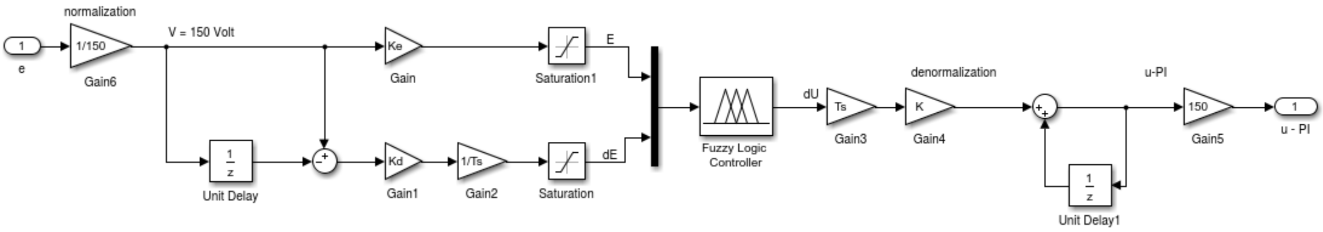
4 Design using simulink

The file "sim1.slx" can be opened by simulink , and describes our PI controller . Also , the script pi-controller can be used to verify that our choice of k_p and k_i is suitable . Indeed, if we run the script we take :
RiseTime: 0.1051 — Overshoot: 0.4150 — Undershoot: 0
System responce in step function input can be shown in figure below :



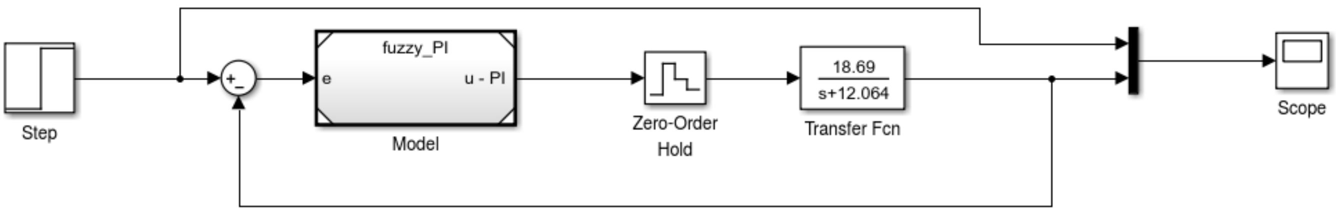
5 Design of the fuzzy PI

Considering the gains of the PI controller , and also the restriction [4] , we design a fuzzy PI controller with $K = \frac{k_p}{F(aK_e)} = 13.5$, $K_e = 1$ and $K_d = a * K_e$, where a $< t_r$, lets assume a $=$ 150 msec .
The structure of our fuzzy PI controller as a simulink model is shown in the figure below :



fuzzy PI controller in simulink

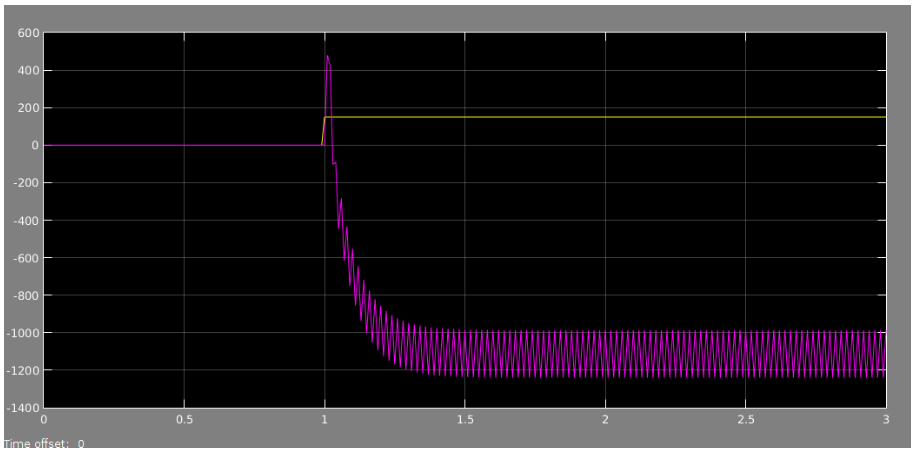
Our final system is now described by the following model :



FLC controlling a dc monitor

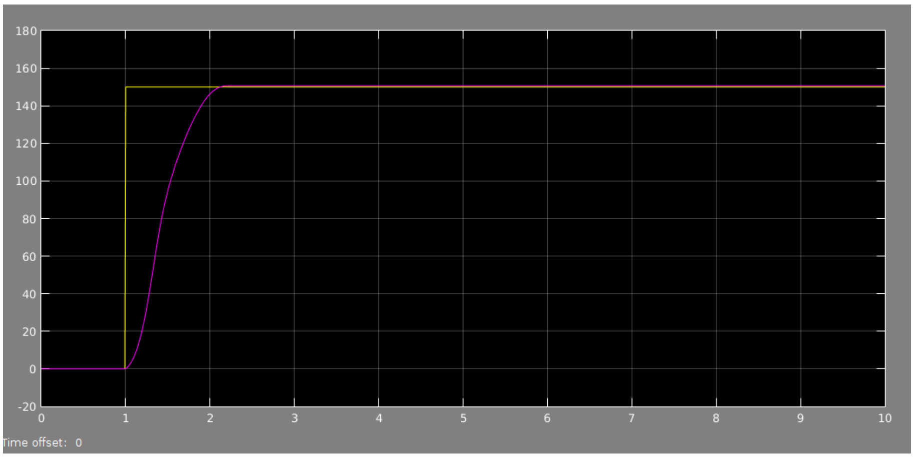
6 FLC tunning

Using the gains that emerged using the gains of the PI controller , with a step function (150 Volt as max) as input the response of our system is :



simulink results ,using Ts = 0.01 sec and our FLC
yellow line stands for the input function ,where purple stands for our system output

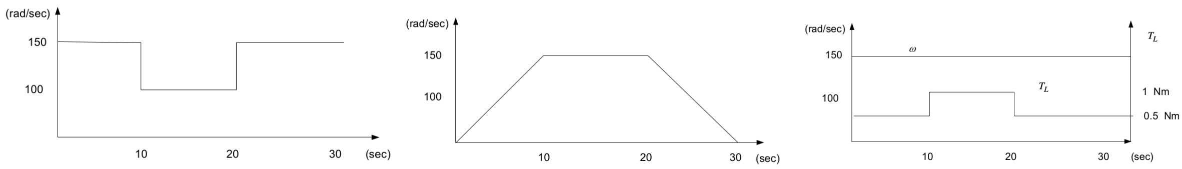
It's obvious that the behavior of our FLC is not the desired , so we must tune the gains :
by selecting a $=$ 0.150 , $K = 0.028$, $K_e = 0.5$ (and of course $K_d =$ a*K_e , we manage to fulfill all the given restrictions , and now system response looks like :



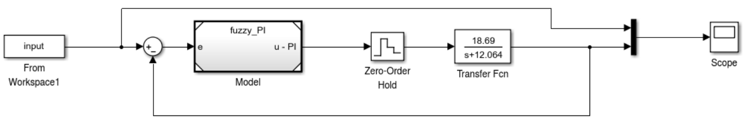
system's output when FLC is using the new values for gains K and K_e

7 Project's tasks

In this project we are asked to demonstrate our system response for a number of different inputs. Those inputs are described by the following figures :



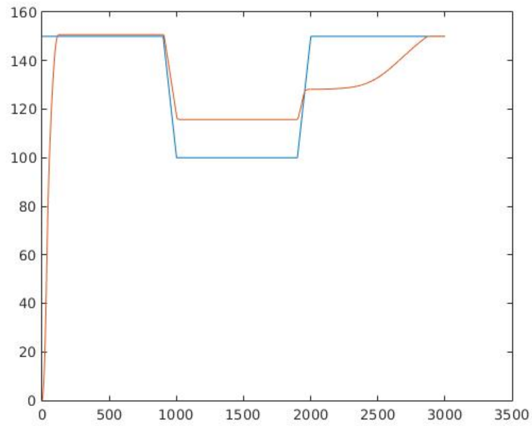
In tasks 1 and 2 , we assume that T_L is zero , and only ω affects the response . In task 3 we observe both ω's and T_L's effect in the response .
In order to simulate those signals using our simulink model, we need a script in matlab to run in first place . In this script we have to define FLC's gains , and also a timeseries matlab object called input . Timeseries objects are used in matlab in order to simulate signals in time (using a vector of values) and then connect these signals to our simulink model , using **"simin" block** (belongs to source library of simulink) . Our system now looks like :



Also we adjust scope properties in order to output simulation's results in an matrix called "ScopeData" , in order to easily plot system's responses .

7.1 Task 1

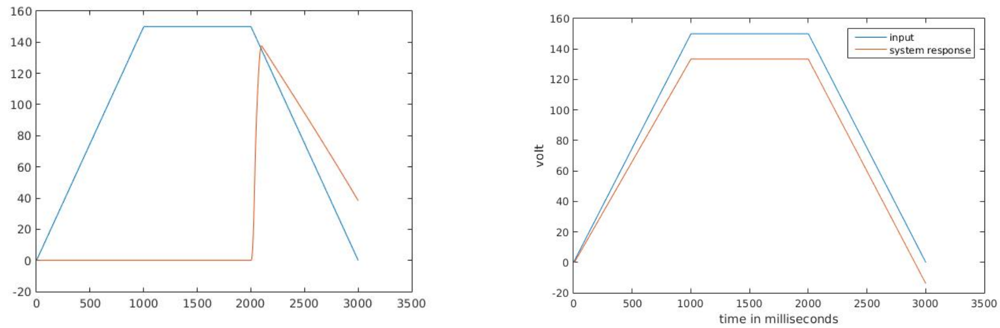
By applying script "task1" and then running the simulation for 30secs , we get :



As you can see, our system's output can follow the input signal (x axis in milliseconds).

7.2 Task 2

By applying script "task2" and then running the simulation for 30secs , we get :

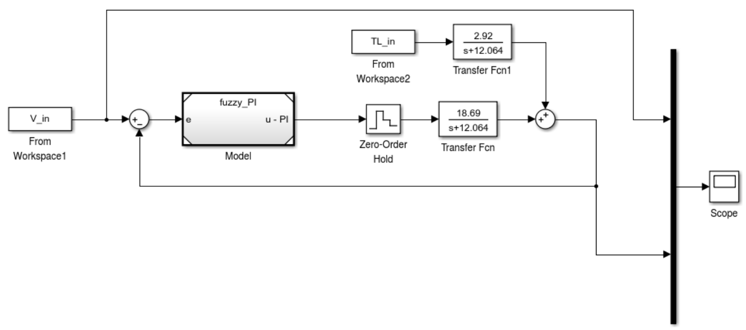


system's response in task's 2 input, before and after tunning

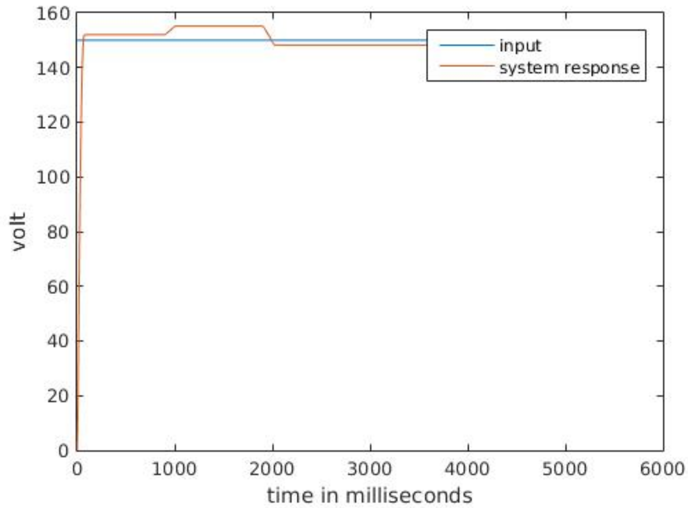
As you can see, our system's output can **not** follow the input signal. We start tunning and find the desired values : a = -0.41 , K = 11.7 , $K_e = -0.0038$, $K_d = aK_e = 0.0016$. Input and system's response can be found by running Task2 script with the new values (second figure above).

7.3 Task 3

In this task we want to observe T_L effect on our system response , so we must use a new model that includes T_L ("model-for-tsk-3.slx") . This model looks like this :



After running script task3.m we observe that our system is stable and although the existence of destruction T_L it can return to it's initial status (having a small error) :



7.4 How to simulate tasks 1-3 and view results

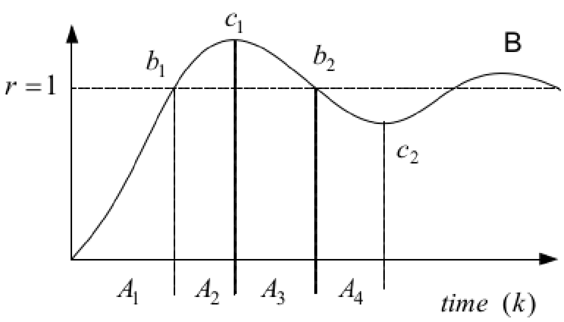
In order to simulate/tune or view the codes that have been used in the previews tasks , you need to keep in mind that :

- testing the PI controller can be done using script named "pi-controller.m"
- the Fuzzy logic controller is the simulink model named "fuzzy-PI.slx" , and as rule base it uses "fis.fis" .
- in order to get task's 1 results ,you need to run "task1.m" script , simulate "model-for-tsk-1-2.slx" in simulink , then go back to "task1.m" script , remove the commented code section and run the code in it .
- same goes for task 2 , but here you need to use script "task2.m"
- same goes for task 3 , but here we use "task3" as our matlab script and "model-for-tsk-3.slx" as our simulink model
- the initial FLC's response can be tested using a step function ,by running "fuzzy-PI-gains.m" script and "main-model" as our simulink model

8 Rule Base - creation and explanation

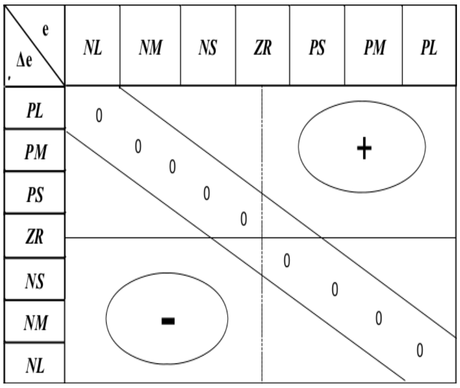
8.1 How to construct the rule base for FLC

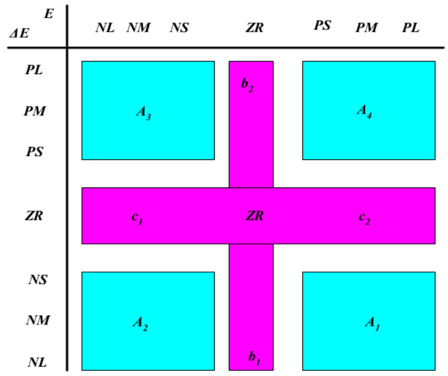
In order to construct the rule base we need to study system's step response . The most general form , that step response may take is shown in the figure :



The **rule base** for a FZ-PI controller is determined by the following rules :

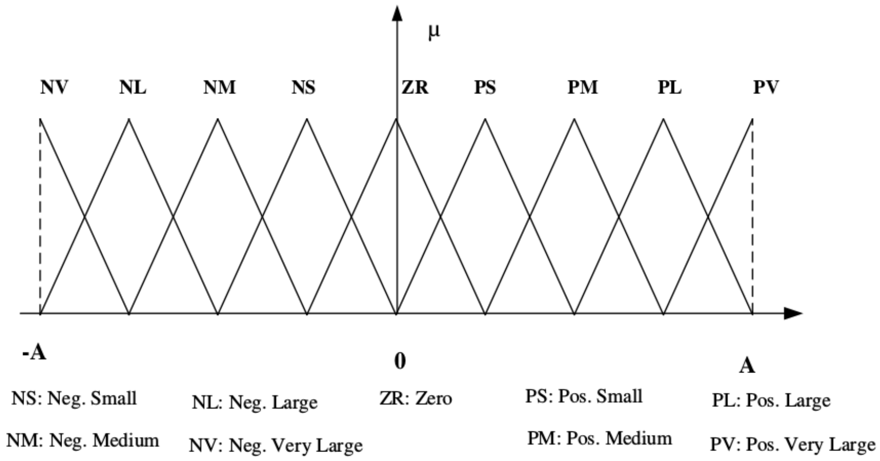
- **[R1] :** $\frac{\partial U}{\partial t} = \text{E AND } \frac{\partial E}{\partial t} = 0$ for points c_1 and c_2
- **[R2] :** for the rest of the points we use the equation : $\frac{\partial U}{\partial t} = \text{E} + \frac{\partial E}{\partial t}$ where symbol "+" is not used for classical addition ,but for verbal addition .
  This way , the final rule base will be in the form :



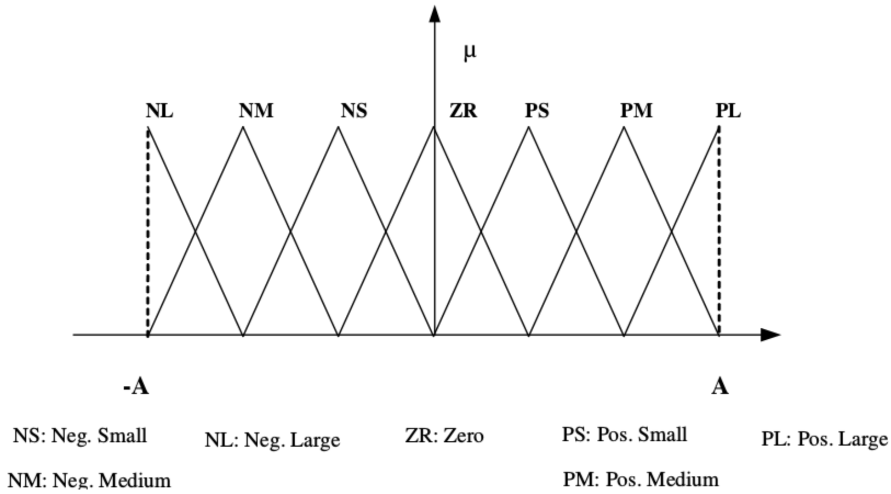


8.2 How our rule base looks like

In our project we are given that error (e or E) and error change (ΔE , or dE , or $\frac{\partial E}{\partial t}$) are described by the following fuzzy sets :



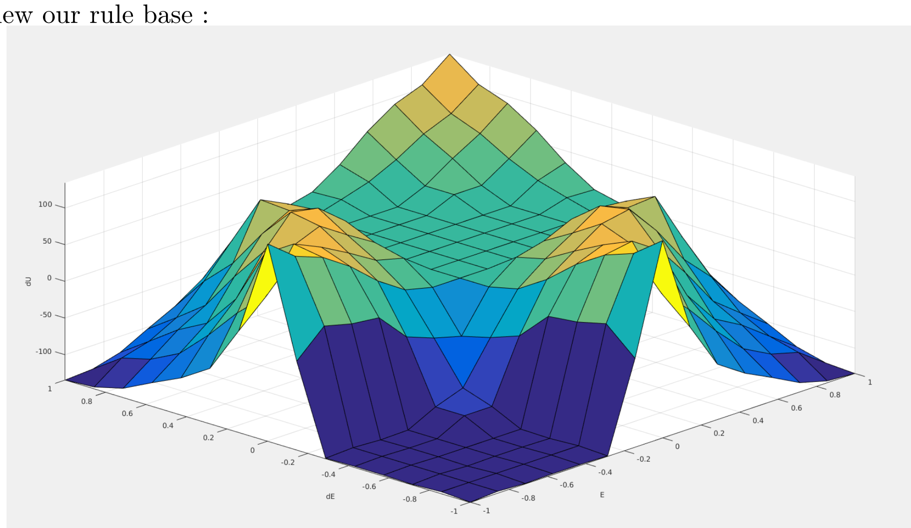
while FLC's output (derivative of ω , dU, or $\frac{\partial U}{\partial t}$) is described by :



Considering all these and of [R1],[R2] we design the rule base for our FLC as :

dE \ E	NV	NL	NM	NS	ZE	PS	PM	PL	PV
PV	ZE	PS	PM	PL	PL	PL	PL	PL	PL
PL	NS	ZE	PS	PM	PL	PL	PL	PL	PL
PM	NM	NS	ZE	PS	PM	PL	PL	PL	PL
PS	NL	NM	NS	ZE	PS	PM	PL	PL	PL
ZE	NL	NL	NM	NS	ZE	PS	PM	PL	PL
NS	NL	NL	NL	NM	NS	ZE	PS	PM	PL
NM	NL	NL	NL	NL	NM	NS	ZE	PS	PM
NL	NL	NL	NL	NL	NL	NM	NS	ZE	PS
NV	NL	NL	NL	NL	NL	NL	NM	NS	ZE

where red letters represent dU's fuzzy sets .
The script main is used to create the fis structure with this rule base , and using gensurf function we can view our rule base :



Just because we are dealing with a 3-D image , the rule base should be better understood by opening (through matlab) the file "rule-base.fig" .
It must almost be mentioned that for the purposes of the project , **dE's domain is set as [-50 ,50] , while E's domain is [-150 ,150]** .

8.3 Response of our FLC in specific input

If we want to observe the effect of different inputs to our system we should type "fuzzy" in matlab's command window , load the fis structure, open rule view and then give the singleton values to our system.