Fuzzy logic control systems. Fuzzy temperature controller.

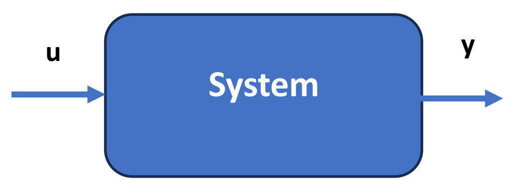
Objectives: understanding how a classical controller works, understanding the differences between classical and fuzzy logic controllers, visualising the output of a fuzzy controller.

Note: MATLAB/Simulink is accessed online (https://matlab.mathworks.com/), by logging in with the MS Teams student credentials (surname.name@student.utcluj.ro).

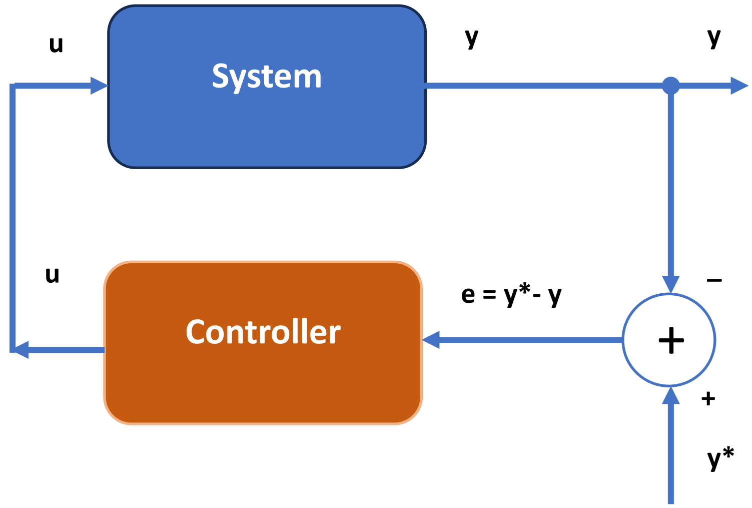
Terms and abbreviations: *classical controller, fuzzy controller, PID.*

○ Classical control systems

Consider an open loop system, with a control input **u** and an output **y**:



The goal is to provide a desired output, **y***. The system is designed so that, in the absence of disturbances and variations of system parameters, **y = y*** for a certain input **u = u***. The absence of disturbances is impossible in real life situations, so **y*** is different from **y**, if the system works in an open loop, for an input **u = u***. Thus, to ensure **y = y*** when disturbances are present, **u** needs to be different from **u***, to compensate the disturbances. The change in **u** depends on the change of **y** with respect to **y***, and is achieved by connecting another system, called ***controller***, between the output and the input of the initial system:



The new system is called *closed loop system*, or *feedback system*. The output **u** of the controller represents the command input of the system and generally depends on the errors computed at previous times: differences between output **y** and desired output **y***, but also on the previous commands **u**:

$$u(k) = f(e(k), e(k-1), \dots, e(k-t), u(k-1), \dots, u(k-t))$$

where *f* is the control law, and *t* is the order of the controller. For t > 0, the controller has a memory of *t*.

The error *e* is computed as:

$$e(k) = y^* - y$$

Generally, the control law *f* is nonlinear. In classical control theory, the control law *f* is inferred based on the mathematical model of the open loop process.

The classical control laws are:

a) the proportional control law (P):

$$u = K_p * e \Rightarrow u(k) = K_p * e(k)$$

b) the integral control law (I):

$$u = K_i * \int e \, dt$$

or the discrete version:

$$u(k) = K_i * \sum_{j=0}^{\tau} e(k-j)$$

c) the derivative control law (D):

$$u = K_d * \frac{d^\tau e}{dt^\tau}$$

d) combinations of these laws, such as the PI controller:

$$u(k) = K_p * e(k) + K_i * \sum_{j=0}^{\tau} e(k-j)$$

Fuzzy controllers

Given a fuzzy logic system with the inputs **e(k), e(k-1),..., e(k-t), u(k-1),..., u(k-t)**, a linguistic dependency between the output **u(k)** and these inputs can be found. Most commonly used fuzzy controllers are for t = 1:

$$u(k) = f(e(k), e(k-1), \dots, u(k-1))$$

Typical fuzzy controllers have a even more compact form, where the previous output **u(k-1)** is not taken into account. The inputs are only **e(k)** and **e(k-1)**, and the output of the controller is the variation of **u**, defined as:

$$\Delta u(k) = u(k) - u(k-1) \Rightarrow u(k) = \Delta u(k) + u(k-1)$$

$$\Delta u(k) = F\big(e(k), e(k-1)\big)$$

where **F** is the transfer function of the control system, given by:

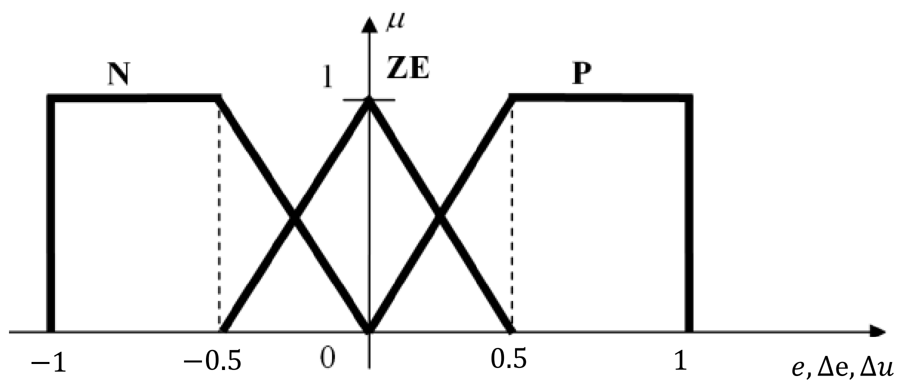
- the fuzzy sets for the inputs and output
- the fuzzy rule base
- the inference mechanism
- the defuzzification method.

Typically, the crisp inputs of the FLS are the error $e(k)$, the variation of the error $\Delta e(k) = e(k) - e(k-1)$, and the output is $\Delta u(k) = F\big(e(k), e(k-1)\big)$.

This fuzzy logic controller with t = 1 was proposed in 1975 Mamdani and Assilian and is called Mamdani-type FLC.

○ Mamdani-type fuzzy logic controller - example

The easiest way to define the fuzzy sets for inputs and output is by using three fuzzy sets (**Negative N**, **Zero ZE**, **Positive P**), identical for the two inputs and the output.



The rule base is deduced knowing that the goal is **y = y***, meaning **e = y* - y = 0**. In other words, the desired output is ***"e is ZE"***. Additionally, it is assumed that the output **y** and the command **u** have the same type of variation:

- if **u** increases, **y** increases;
- if **u** is constant, **y** is constant;
- if **u** decreases, **y** decreases;

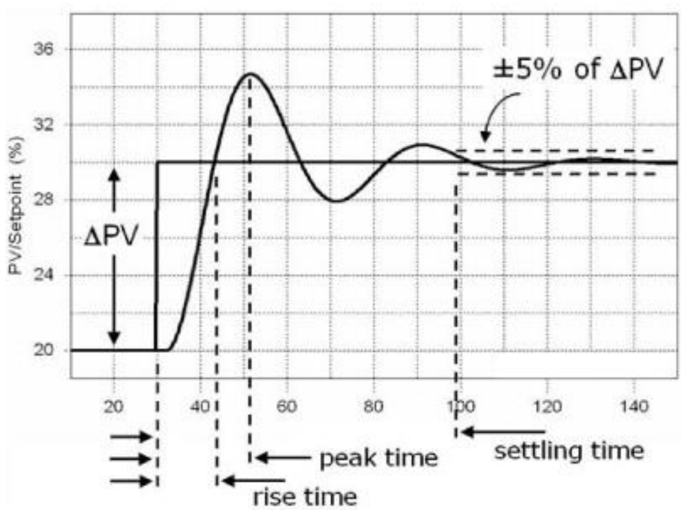
The complete rule base takes into account all possible combinations of the two outputs:

Δe \ e	***N***	***ZE***	***P***
N	N	N	ZE
ZE	N	ZE	P
P	ZE	P	P

The inference mechanism is usually Mamdani, that is max-min inference, and the defuzzification method is COA (centroid).

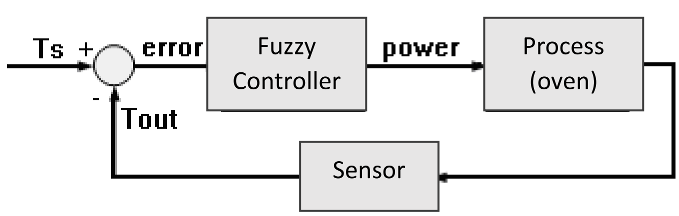
The typical response of a system for automatically adjusting to a step signal is presented in the next figure and can be characterized by several parameters:

- rise time, peak time, settling time
- overshoot.



○ Fuzzy logic temperature controller

The control system consists of a process (oven) and the fuzzy controller:

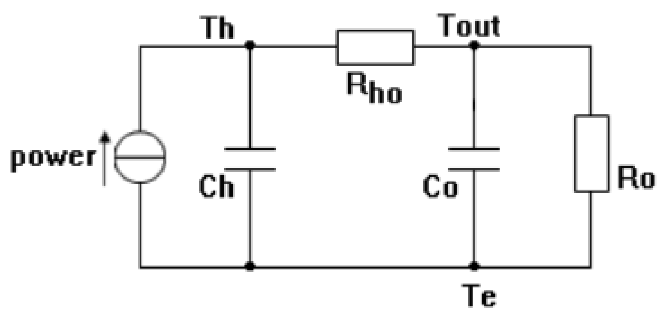


where:

- *Ts* – desired temperature (*Set Point Temperature*)
- *Tout* measured temperature (inside the oven)
- *error* = *Ts* – *Tout*

- *power* - power required for heating/cooling (temperature controller output, and the command input of the process).

The purpose of the controller is to maintain the temperature inside the oven at a constant value, equal to the desired temperature (*Ts*). To model the heat generation and transfer, the following equivalent circuit is used:



The *power* current source (thermal power) represents the power supplied to the heating/cooling element. The system has an electrical heating/cooling element with a capacity of Ch=500[J/°C], connected through a resistor Rho=0.143[°C/W], for a heating capacity of Co=1000[J/°C]. The oven dissipates heat into the exterior (external temperature Te), through the thermal resistor Ro=0.1[°C/W]. The temperature controller adjusts the power dissipated to the heating element *power*, by comparing the temperature inside the oven Tout against the set point (desired) temperature Ts.

Download "*TempControl.zip*" and place the archive (using *drag-and-drop*) in the current directory of MATLAB. Double click to unzip and view the contents of the folder.

http://www.bel.utcluj.ro/dce/didactic/sf/lab/6ControlerTemperatura/TempControl.zip

The Simulink block diagram is shown below. To make sure that the controller is universal, linear conversion blocks were used at the two inputs (*Scale error*, *Scale delta_error*) and at the output (*Scale_power*). The values at the inputs of the controller are limited to [-1, 1] by means of saturation blocks (*Saturation*, *Saturation1*).

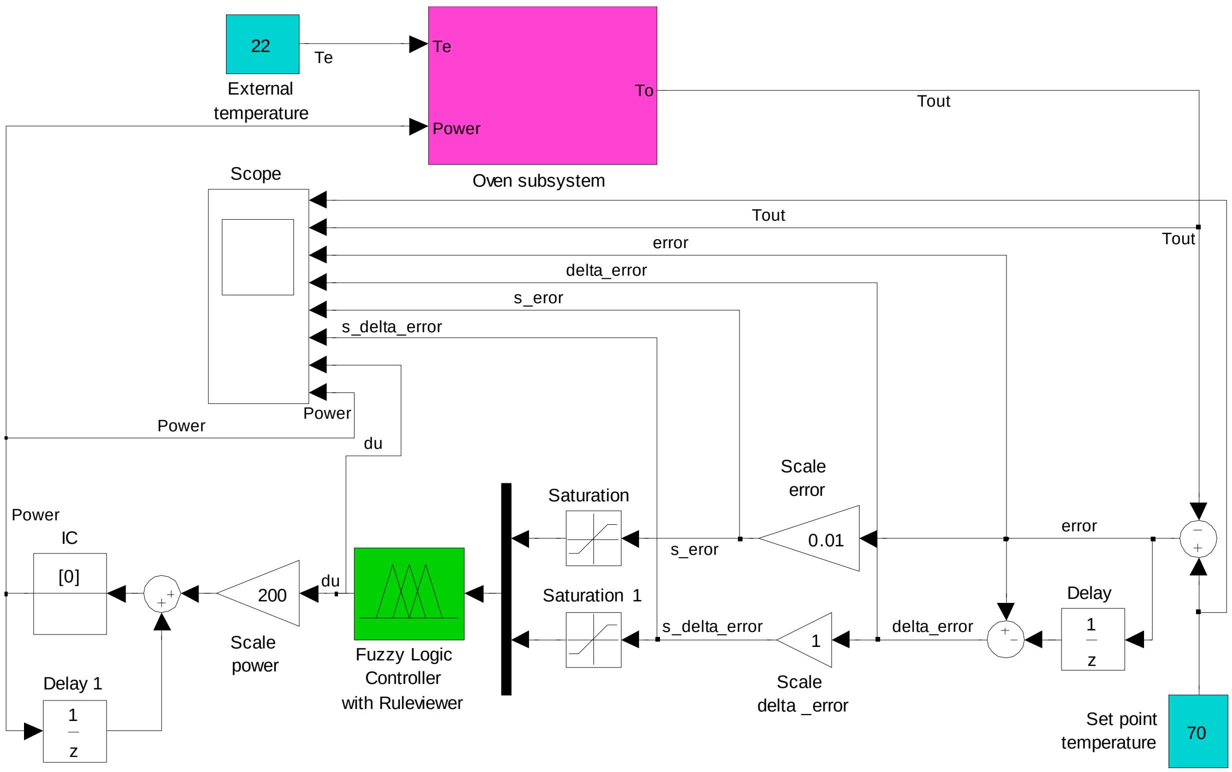
Exercise 1

Implement the Mamdani-type fuzzy logic controller, based on the previous specifications. Visualise the control surface. Save the system as „*TempControlM.fis”.* Read the system into a variable called *fls*, using *readfis*.

Exercise 2

Open the Simulink model of the temperature control process, “*TempControlM.mdl*”. Initialize the parameters of the oven, by running the „*HeaterOven_params.m”* script.

Start the simulation and visualize the waveforms on the oscilloscope (double-click on *Scope*). Measure the parameters of the control system: rise time, settling time, overshoot. Save the measured values.



Exercise 3

Adjust the scaling factors for inputs and output, so that the performance of the control system is improved.

Exercise 4

Repeat *Exercise 2*, this time using a Takagi-Sugeno type controller. The conversion from Mamdani to Takagi-Sugeno is done by using the *Mamdani to Sugeno* option. In this case, the Simulink model is "*TempControlTS.mdl*". Which of the two control systems performs better, for the initial values of the scaling factors?