Comparison of MATLAB, Simulink and AnyLogic Approach to ARGESIM Benchmark C9 'Fuzzy Control of a Two Tank System'

Sebastian Hödlmoser*, Florian Kitzler

Inst. of Analysis and Scientific Computing, Vienna University of Technology, Wiedner Haupstraße 8-10, 1040 Vienna, Austria; * *sebastian.hoedlmoser@ tuwien.ac.at*

Simulation Notes Europe SNE 23(3-4), 2013, 195 - 200
DOI: 10.11128/sne.23.bn09.10219
Received: September 20, 2013; Revised: October 10, 2013;
Accepted: November 30, 2013;

Abstract. This contribution compares modelling and simulation of the ARGESIM Benchmark C9 'Fuzzy Control of a Two-Tank System' with three approaches: (1) programming directly in MATLAB (2) using SIMULINK and the MATLAB Fuzzy Toolbox, and (3) using AnyLogic, a Java-based grapic simulation environment.

The MATLAB implementation required direct programming, whereby the nonlinear ODE model for the two-tank system was simulated by MATLABs ODE solvers, and fuzzification, inference, and defuzzification was programmed by 'pure' vector handling feature. The Simulink implementation is straightforward: graphical blocks for the ODE model, and use of the Fuzzy Toolbox, wich supports graphical design of the fuzzy controller. Anylogic offers various graphical modelling methods, also classic block diagrams. But for tis comparison the System Dynamics modelling capability was used, which allows a genuine mapping of 'tanks' as reservooir variables; the fuzzy controller was programmed directly in Java embedded into the simulation environment.

The contribution discusses advantages and disadvantages of the modelling approaches – in modelling, in implementation and in simulation and efficiency.

1 Model Description

We consider a two tank system as showed in Figure 1. Tank 1 has an inflow u and is coupled with tank two by a valve v_1. Tank 2 has a second valve v_2, acting as a sink. The system is characterized by the nonlinear ODE set:

$$f = 0.06624 v_1 \sqrt{|x_1 - x_2|} sign(x_1 - x_2)$$
$$\dot{x}_1 = 0.067u - f$$
$$\dot{x}_2 = f - 0.0605 r v_2 |x_2|^{0.48}$$

with

$$r = \begin{cases} 1.2 & if\ x_2 < 16\ cm \\ 1 & if\ x_2 \geq 16\ cm \end{cases}$$

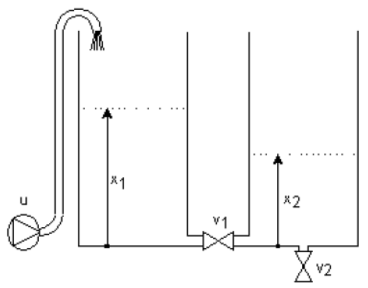


Figure 1: Two tank system with source u, liquid levels x_1, x_2 and valves v_1, v_2.

The valve positions are $v_1=0.4$ and $v_2=0.3$. The aim of the model is to control the liquid level of the second tank with a fuzzy controller. The model can be subdivided into two different systems. The fuzzy controller (FC1, FC2, FC3) calculates the inflow parameter u for the given membership functions for x_1 (liquid level of tank 1) and ex_2 (difference to the desired liquid level in tank 2). These are given as stepwise linear functions. The membership function of u is calculated combining the given rules (MIN for AND, MAX for OR) using MAX-PROD-Inference.

Defuzzification is done by the centroid method. For calculating the center of gravity, two intergrals are approximated using trapezoid rules with stepsize *0.25* (considering the membership functions of u, which change their slopes on vertices *mod 0.25*).

With the calculated value of u we can then solve the differential equation for x_1 and x_2 for the next timestep.

2 Implementation

As mentioned before the two tank system can be subdivided into two different subsystems that can be implemented independently.

First the fuzzy controller has to compute the variable *u* for given values of x_1 and ex_2. With x_1, ex_2 and *u* we can then solve the ODE system for the next time step. This was implemented using the explicit Euler method with fixed time step of *1* in all three environments.

2.1 MATLAB Model and Implementation

To implement the fuzzy controller we needed to define the membership functions for the variables x_1, ex_2 and *u*. These are generalized indicator functions that can reach any value between *0* and *1*. For the variable x_1 we used trapezoid membership functions (Figure 2), whereas triangular membership functions are used for the variables ex_2 and *u* (FC1). In FC2 and FC3, the membership functions of *u* are singletons, i.e. functions with a pointwise value of *1* and *0* in between (imagine triangular functions with an 'infinte slope').

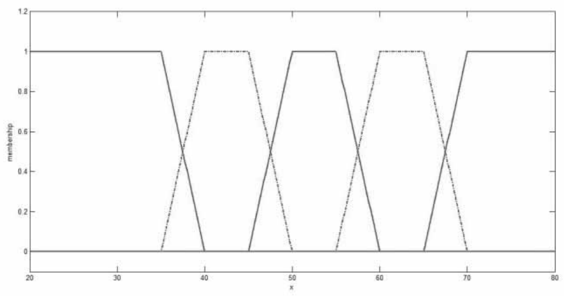


Figure 2: Trapezoidal membership functions for variable x_1.

		x_1				
		nl	**p1**	**p2**	**p3**	**p4**
	p1	p8	p7	p5	p3	nl
ex_2	**p2**	p7	p6	p4	p3	nl
	p3	p7	p5	p3	p2	nl
	nl	p4	p3	p2	p1	nl
	n1	nl	nl	nl	nl	nl

Table 1: Linguistic rules for the output membership function for u of the fuzzy controller.

In Table 1, a set of linguistic rules is given to show how to calculate the membership function for the output variable *u*. To combine two or more of this rules we need the combination rules for AND (Minimum) and OR (Maximum). Under these combination rules, we reach the membership function p_2 for *u* as follows:

```
IF (ex2 = nl AND x1 = p2) OR
 (ex2 = p1 AND x1 = p3) THEN u = p2
```

is translated in MATLAB to

```
mb_FC1(3)=
max([min(mb_ex2(2),mb_x1(3)),...
    min(mb_ex2(3),mb_x1(4))]);
```

Given values for x_1 and ex_2 can be contained in more than just one membership function, with different grades of membership. These values imply more than one membership function for the variable *u,* which are combined using the MAX-PROD-inference. This means that the different membership functions of *u* get multiplied by the grade of membership and then combined using the maximum of all these functions. Figure 3 shows an example of such a membership function for *u* when $x_1=40$ *and* $ex_2=13$.

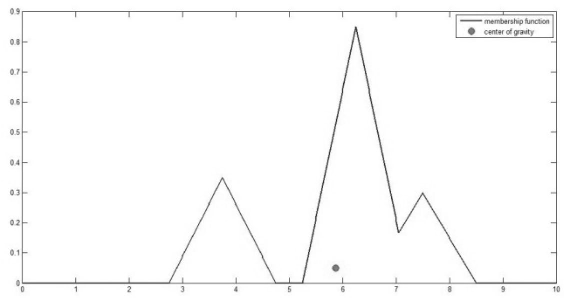


Figure 3: Calculated membership function for output variable u and center of gravity.

The last step of the fuzzy controller is the defuzzification of the membership function of *u* to compute a value to set the inflow valve. This was done by the centroid method, which sets the sharp value of *u* to the x-coordinate of the center of gravity of the membership function.

We computed the center of gravity using the trapezoid rule to calculate the occurring integrals (FC1). All other steps were implemented in a MATLAB routine and only required evaluation of membership functions and the MIN and MAX functions. For solving the ODE the Euler-method was used.

2.2 SIMULINK Model / Implementation

In Figure 4 the SIMULINK implementation is displayed. One can obtain the controlling circuit with the control block 'Fuzzy Logic Controller' and the tank system itself (in the 'tank system' subsystem, which we won't discuss any further).

The 'Fuzzy Logic Controller' block calls the fuzzy controller, which is generated with the Fuzzy Toolbox (see below) and saved as a '.fis' file. The coupling of SIMULINK and '.fis' files is just one way to use the Fuzzy Logic Toolbox.

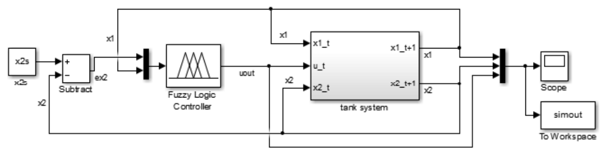


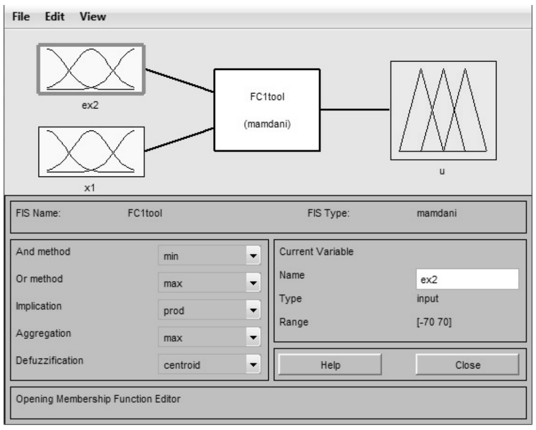
Figure 4: Model structure of the two tank system with fuzzy controller implemented in SIMULINK using the Fuzzy Toolbox.

With the Fuzzy Toolbox it is very easy to define and visualize the used membership functions and combination rules. Figure 5 depics the GUI of the Fuzzy Toolbox (FIS-Editor, 'Fuzzy Inference System') opened via the command `'fuzzy'` in the MATLAB workspace. In the first shell one can specify the number of input and output variables and set the Fuzzy Control options like rules for AND and OR, type of inference (aggregation and implication) and method for defuzzyfication.

The membership functions can be drawn via a graphic interface as seen in Figure 6. The linguistic rules don't have to be programmed but directly declared in a respective window.

To generate the controllers FC2 and FC3 we had to change the type of the controller to 'sugeno' (instead of default 'mamdani'). This type of controller allows singleton membership functions for the output variable.

The Fuzzy Toolbox Editor allows changing the controller options and comparing the effects on the model without changing the source code, like it would be necessary in the MATLAB version.



FC1tool (mamdani)

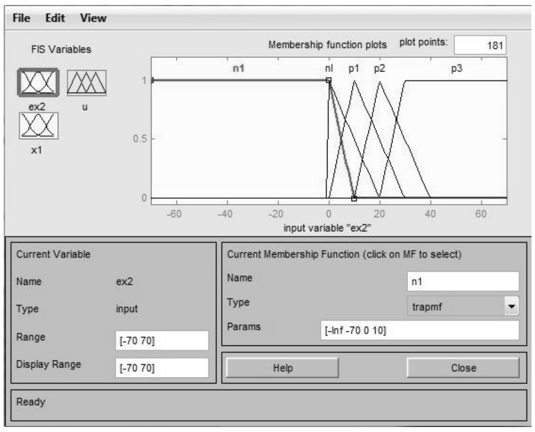
FIS Name: FC1tool FIS Type: mamdani

And method: min
Or method: max
Implication: prod
Aggregation: max
Defuzzification: centroid

Current Variable
Name: ex2
Type: input
Range: [-70 70]

Opening Membership Function Editor

Figure 5: GUI of the Fuzzy Logic Toolbox.



plot points: 181

Current Variable
Name: ex2
Type: input
Range: [-70 70]
Display Range: [-70 70]

Current Membership Function (click on MF to select)
Name: n1
Type: trapmf
Params: [-Inf -70 0 10]

Ready

Figure 6: Interface to draw the Membership Functions

2.3 AnyLogic Model / Implementation

As mentioned above, AnyLogic contains a variety of simulation methods. For our purposes, AnyLogic's system dynamics palette offers an intuitive, graphic approach to implement the tank system. Here a flow chart is generated, where the ODEs and other parts of the system lie within.

Our system dynamics model is depicted in Figure 7. The clouds are a source and a sink, in between are two so called stocks, which represent the two tanks (x_1 and x_2). These are coupled with flows which depend on the signs of equations in the ODE set, e.g. the inflow into x_1 is set to $0.067u$ and the flow from x_1 to x_2 is f, giving the change over time $\dot{x}_1$ (compare the second equation of the ODE set).

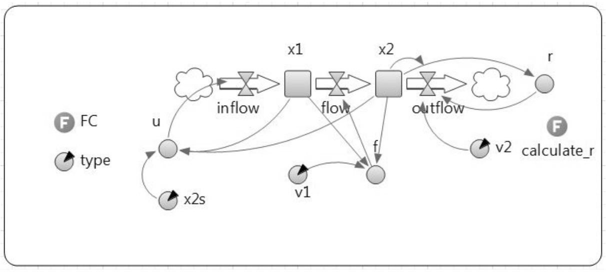
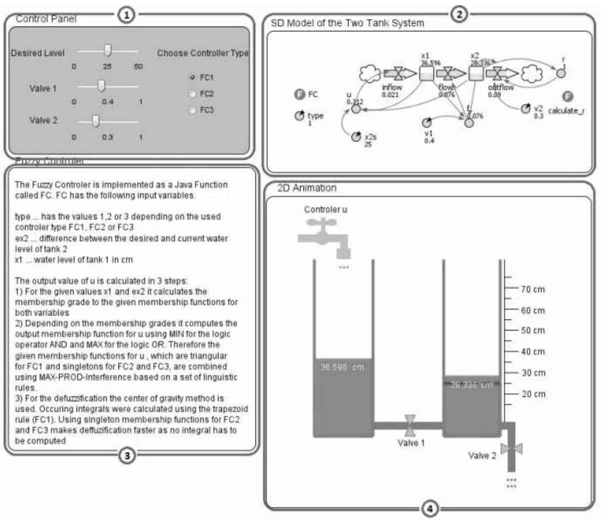


Figure 7: System Dynamics Model of the Two Tank System.

In general, circles are dynamic variables (like *u*, *r* and *f*), circles with black arrows are parameters (*v1*, *v2* and *x2s*).

The controller is added as the function FC, written in JAVA, and lies within the dynamic vaiable *u*. The parameter 'type' denotes which controller should be used (FC1, FC2 or FC3). Besides that, the function FC is basically the JAVA translation of the controller funcitons used in MATLAB.


Control Panel

Desired Level 0 25 50

Valve 1 0 0.4 1

Valve 2 0 0.3 1

Choose Controller Type
[✓ FC1]
FC2
FC3

Fuzzy Controler

The Fuzzy Controler is implemented as a Java Function called FC. FC has the following input variables.

type ... has the values 1,2 or 3 depending on the used controler type FC1, FC2 or FC3
ex2 ... difference between the desired and current water level of tank 2
x1 ... water level of tank 1 in cm

The output value of u is calculated in 3 steps:
1) For the given values x1 and ex2 it calculates the membership grade to the given membership functions for both variables
2) Depending on the membership grades it computes the output membership function for u using MIN for the logic operator AND and MAX for the logic OR. Therefore the given membership functions for u , which are triangular for FC1 and singletons for FC2 and FC3, are combined using MAX-PROD-interference based on a set of linguistic rules.
3) For the defuzzification the center of gravity method is used. Occuring integrals were calculated using the trapezoid rule (FC1). Using singleton membership functions for FC2 and FC3 makes deffuzification faster as no integral has to be computed

Figure 8: Presentation of the simulation in AnyLogic: (1) control panel, (2) system dynamics model, (3) brief description, (4) 2D visualization.

Here we built a 2D visualization of the two tanks and linked the liquid levels to the sizes of the blue rectangles which represent the liquid. Now instead of just showing plots of the liquid levels over time (compare Figure 11), one can actually see a model of the tanks and how the system changes in real time. Also by adding control units like sliders and radio buttons, we built a panel where one can change the parameters and controllers during simulation.

Figure 8 shows a possible presentation of the simulation. In the left upper corner we have the control panel to set the desired liquid level and vale parameters and choose the type of controller, beneath a brief description of the simulation. On the right side we see the system dynamics model and the corresponding 2D model.

3 Tasks and Results

The following section deals with the tasks of ARGESIM Benchmark C9. Here the performances of the MATLAB and SIMULINK implementation are compared. There are no qualitative differences between the following results and the AnyLogic simulations. However the computation times in AnyLogic are significantly higher.

Task a: Computation of Control Surfaces. The characteristic surfaces of the two fuzzy controllers FC1 and FC2 where computed. Since the characteristic surfaces are the same in both versions (up to little differences due to defuzzification), this was done only in MATLAB (here a plot command had to be programmed; in the Fuzzy Toolbox a surface view command is already implemented, see Figure 10).

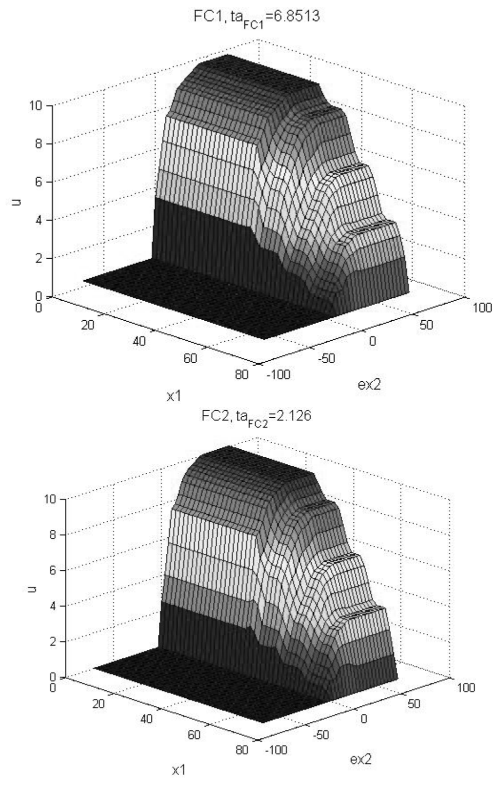
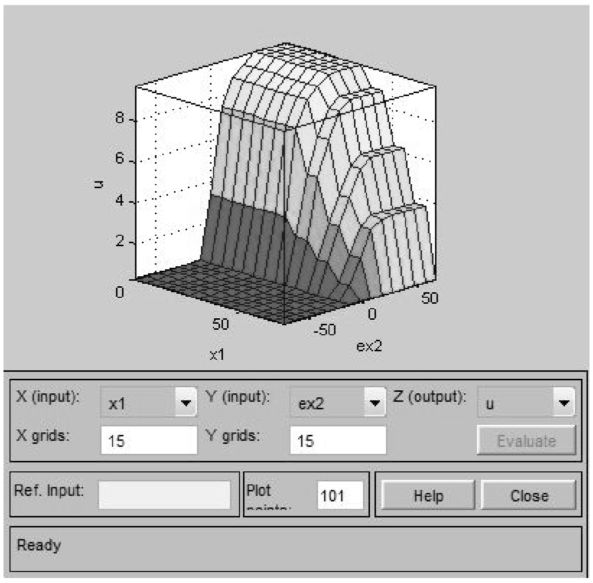


Figure 9: Surface plots for FC1 and FC2 in MATLAB; the characteristic surface with the fuzzy toolbox (nearly) coincides.

Figure 9 shows the visualization for ex_2=[$-70,70$] and x_1=[$0,70$] with both intervals subdivided 40 times. ta_{FC1} and ta_{FC2} denote the computation times for the controller FC1 and FC2, respectively. With our machine we got $ta_{FC1}=6.8513$ s and $ta_{FC2}=2.126$ s, giving a ratio of $ta_{FC1}/ta_{FC2}=3.2226$.

It should be mentioned that the computation time of FC1 increases with decreasing stepsize for the defuzzyfication (calculation of the center of gravitiy). However, with smaller stepsizes no significant changes can be observed in the behaviour of FC1 (in this computation the stepsize was *0.25*). Since FC2 uses singleton membership functions, no center of gravtiy has to be calculated, thus making defuzzification easier.

A remark: as mentioned, the plots in Figure 9 where programmed in MATLAB. The Fuzzy Toolbox surface viewer opens - without any further programming - a window as depicted in Figure 10.



X (input): x1 Y (input): ex2 Z (output): u
X grids: 15 Y grids: 15

Ref. Input: Plot points: 101

Ready

Figure 10: Surface viewer with the Fuzzy Logic Toolbox.

Task b: Simulation with Fuzzy Control. The system was simulated for $t=1000$ seconds and a desired level $x_{2s}=25$ cm for the second tank. All calculations were done with both methods. Corresponding computation times where $tb_{FC1}=3.5353$ s, $tb_{FC2}=0.78904$ s in MATLAB and $tb_{FC1,Tool}=1.7009$ s, $tb_{FC2,Tool}=0.79509$ s in the SIMULINK/Toolbox version. The two ratios are $tb_{FC1}/tb_{FC2}=4.4805$ and $tb_{FC1,Tool}/tb_{FC2,Tool}=2.1392$.

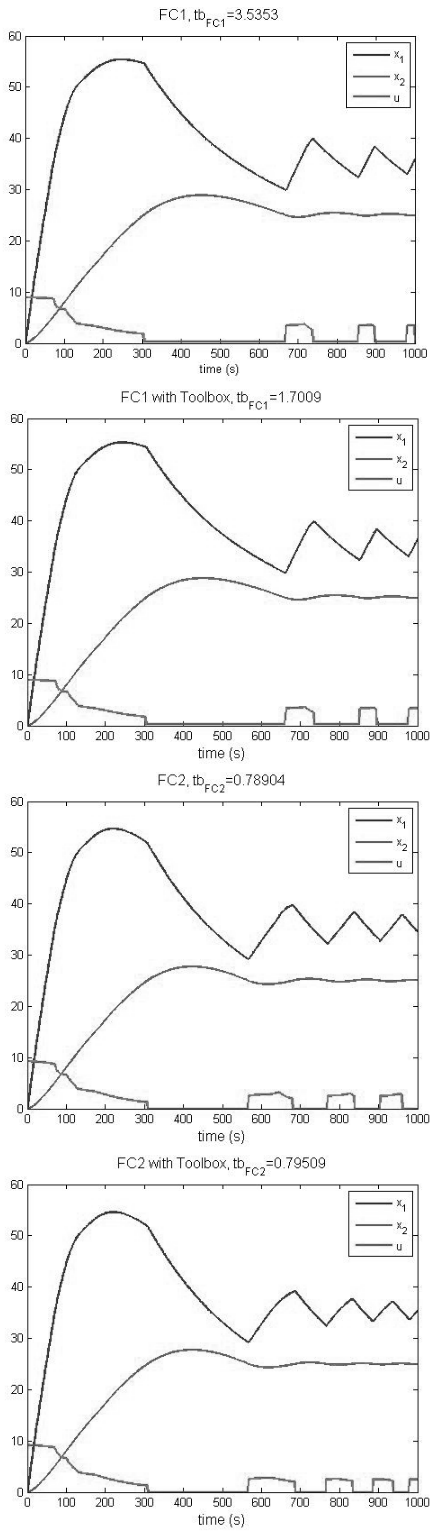


Figure 11: Simulation with FC1/FC2 with MATLAB and SIMULINK/Fuzzy Toolbox; the different performances of the controllers can be observed.

In Figure 11 we can obtain the different performances of the controllers. Although the controller output *u* is slightly different (due to differences in the calculation of the center of gravity), the behaviour of FC1 stays the same in MATLAB and in SIMULINK. The differences in FC2 are clearer (note the changes at *t~900*). Their origin lies in the interference algorithm and defuzzification of the toolboxes sugeno-type-controller. This controller supports singleton membership functions, but no MAX-PROD-interference. Also the deffuzification is different.

An interesting observation is that the choice of singletons in FC2 instead of triangular membership functions increases the controller's performance. It takes controller FC1 anbout *100* seconds more to reach a stable liquid level of about *25* cm.

Task b: Simulation with Weighted Fuzzy Control.. The rules of FC2 where weighted to gain the controller FC3. In our MATLAB programm this happens in the interference algorithm, where corresponding minima where multiplied by the factor *0.1*. The Fuzzy Toolbox supports weighting in the rule editor. Repeating the calculation of task a, the computation time for FC3 was $tc_{FC3} = 2.1236$.

4 Conclusion

Applying fuzzy logic to control systems bears a lot of advantages. The formulation of membership functions and rules is an intuitive and comparatively accessable way to formularize a problem.

Out of the three methods, implementing a fuzzy controller in MATLAB without the use of any toolbox or software takes the most programming effort. Also some basic knowledge about ODE solving and a good understanding of fuzzy controlling are necessary. Thus the pure MATLAB approach takes the most time to realize, on one hand because of the programming itself, on the other because a certain theoretical backround is needed. However this isn't necessarly a disadvantage since this method gives the best understanding of the underlying processes. The pure programming also leads to a high transparacy and flexibility of the system.

If the theoretical backround is of less interest, the MATLAB Fuzzy Toolbox delivers a convenient way to generate fuzzy controllers.

Via the FIS editor, one can define and edit the membership functions easily, specify the linguistic rules, inference and defuzzification and leave all the programming to the toolbox. All these options and functions can be altered by just a few clicks to compare different controllers and the rule and surface viewer can help get a better understanding of the system. The fuzzy toolbox enables also users with lesser knowledge about fuzzy control theory a rather easy access. Coupled with SIMULINK this is an easy way to implement a fuzzy control system. But while the SIMULINK model gives a nice overview about the control circuit (Figure 4), the implementation of the ODE set can be a rather confusing.

In comparison to MATLAB and SIMULINK, the big strength of AnyLogic lies in its countless ways to present the data and simulations. Instead of relying on usual plots, one can create interactive sheets, build dynamic objects, add external graphics, explanations, comments etc. The parameter manipulation can be done during simulation to get a better understanding about the effects on controlling. Also the system dynamics model is way clearer than the ODE system in SIMULINK (in our case the components of the SD model can be linked - more or less - directly to the schematic model from Figure 7). When it comes to the fuzzy controller itself, in AnyLogic the parts have to be programmed form scratch in JAVA in the same manner as in the MATLAB version. This leads to the same amount of work but again also to a good insight in the control unit.

In summary we belief it is a question of useage which environtment is to be preferred. For pure controlling, the Fuzzy Toolbox/SIMULINK offers the most convenient way. For educational or presentation purposes, AnyLogic is the better alternative.

Model sources

All model sources (MATLAB m-files, Simulink .mdl files, and AnyLogic .alp files) and a short file documentation can be downloaded (zip format) by EUROSIM sociteties' members from SNE website, or are availably from the author.

References

[1] Goldynia J W, Marinits J M. Comparison 9: Fuzzy Control of a Two-Tank System Definition. *SNE Simulation News Europe*. 1996; 6(2): p28 – p29.

[2] Jaouad S, Ottens M. *Einführung in die Regelungstechnik mit Fuzzy Logik,* Skriptum, Technische Fachhochschule Berlin, 2009.