# A Dymola-Implementation of Benchmark C9R 'Extended Fuzzy Control of a Two-Tank System' with Directly Programmed Fuzzy Control

Anto Sodja, University of Ljubljana, Slovenia; *asodja@gmail.com*

**Simulator:** *Dymola* 5.3d, Dynamic Modeling Laboratory, is an object oriented simulation environment for acausal modeling, simulation and visualisation of continuous, hybrid and discrete models. Dymola is able to understand Modelica, a unified textual and graphical modelling language offering many libraries for applications (WWW.DYNASIM.COM).

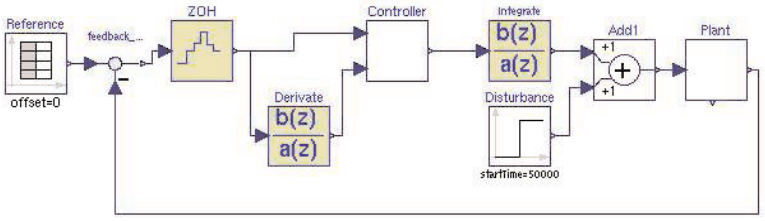


Figure 1: Dymola / Modelica model of the overall system.

**Modelling**. The model of the whole system was build by using standard blocks from the Modelica Standard Library, except model of the fuzzy controller. The model is composed of the *Controller* model, the *Plant* model, and standard blocks from the Modelica Block Library (zero-order-hold, discrete transfer function, and sources; Figure 1). The plant model is described textually by the system governing ODEs using Dymola notation, put into a graphical block.

No tool for fuzzy control is supplied in Dymola (5.3d), neither in the standard Modelica library (2.2). The fuzzy control has to be programmed directly in Dymola, using the `algorithm` section. The fuzzy controller model is hierarchically comprised from simpler blocks, integrated into a graphical block (Figure 2).

The input fuzzification block is made up of blocks which represent single fuzzy set and linearly interpolate input provided by parameters in a table:

```
parameter Real values[2];
parameter FuzzySetMembership membership[2];
algorithm
  if u <= values[1] then y := membership[1];
  elseif u <= values[2] then
     y := interpolate(values,membership,u);
  else y := membership[2];
end if;
```

Inputs and outputs are routed to input and output of the parent block respectively. The same holds in case of output blocks where two outputs, center and surface must me provided by each block:

```
parameter FuzzySetMembership height = 1.0;
parameter Real points[3];
protected
parameter Real C =
        points[2]+points[3]+points[1])/3
parameter Real S =
          height*(points[3]-points[1])/2;
algorithm
center := C;
surface := membership * S;
```

All outputs are then routed to a block which calculates the output by center of gravity formula, and finally, the interface engine block is implemented as an explicit set of rules due to only four simple rules:

```
   y = sum(vars*weights) / sum(weights);
algorithm
   y[1] := set_h[1]; y[4] := set_h[3];
   y[2] := set_h[2] * set_v[2];
   y[3] := set_h[2] * set_v[1];
```

All components (input/output blocks and interface engine) are then connected to the controller model `ProcessModelFC1` as depicted in Figure 2. As the control is programmed directly, I-FC1 and I-FC2 are implemented very similar. The controller model is a discrete Dymola model, which can be used stand-alone or in combination with other Dymola models.

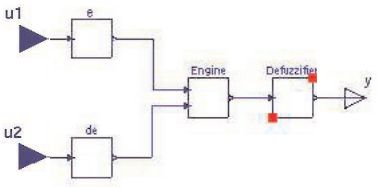


Figure 2: Controller, Dymola blocks.

**A-Task: Controller Surfaces.** The stand-alone con-troller model, with a discrete ramp signal on both inputs of the fuzzy controller, calculates the controller surface (control calculated at 2.500 points). For plotting the results, due to lack of any 2-D plotting abilities of Dymola, MATLAB was used. Data from MATLAB's file `SurfaceFC1.mat` that Dymola produces were extracted with help of `dymload.m` and `dymget.m`. The vectors need to be reshaped into a 2D matrix, which is suitable for surface plotting in MATLAB. Figure 3 shows the control surfaces: after rescaling, the shapes for I-FC1 (max. 5) and I-FC2 (max. 2) look very similar, but I-FC2 calculates significantly lower control values (slower control action).

**B- Task: Transient Response**. The sources (Figure 1) are defined as table (`Reference`) and as step function (`Disturbance`). Simulation is done in

the Dymola menu-driven environment, which also produces the result plots (Figure 4 for I-FC1). For simulation of I-FC2, in the controller model the output defuzzifier of the control was redeclared and replaced by singleton defuzzifier:

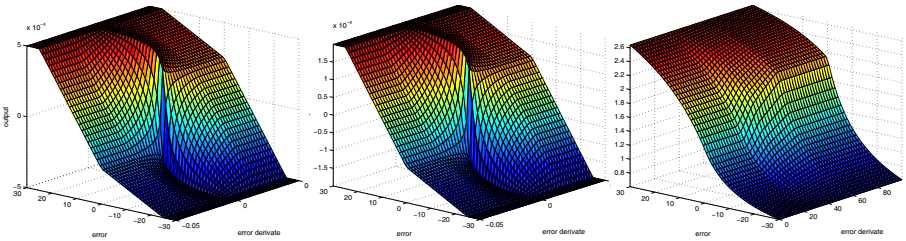

Figure 3: Control surfaces for I-FC1 (a - left), I-FC2 (b - midst) and P-FC (c - right)

```
model ProcessModelFC2
extends ProcessModelFC1(Controller(redeclare
        Components.FuzzyOutput_Singleton
        Defuzzifier));
end ProcessModelFC2;
```

Results for I-FC2 in Figure 5 show, that due to lower gain the response is slower and that also the overshot is smaller, compared with I-FC1.

**C-Task: Comparison with Proportional Fuzzy Controller.** To model the proportional fuzzy controller P-FC, input and output fuzzy block and the interface engine need to be rebuild (different membership functions, different rule base), but the base input (and output) fuzzy set blocks can be reused and almost no code needs to be written additionally.

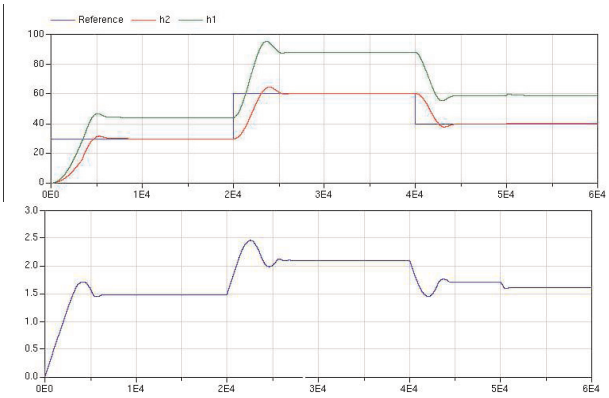


Figure 4: State variables $h_1$, $h_2$ and reference signal (upper) and control signal (lower) over time for I-FC1.

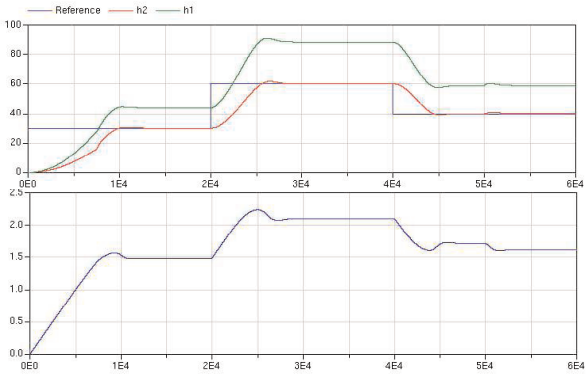


Figure 5: State variables $h_1$, $h_2$ and reference signal (upper) and control signal (lower) over time for I-FC1.

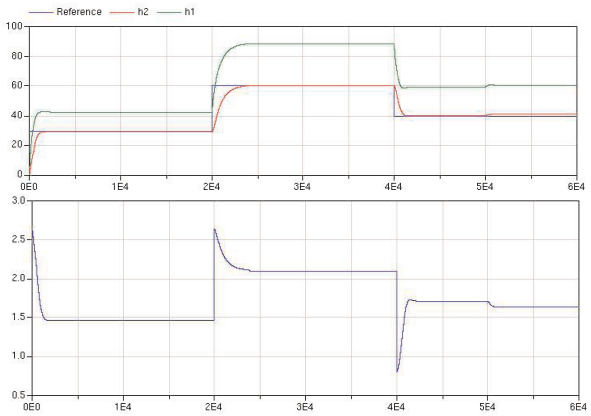


Figure 6: State variables $h_1$, $h_2$ and reference signal (upper) and control signal (lower) over time for P-FC.

The overall model becomes simpler, because the discrete transfer functions can be omitted, whereby the controller input has to be changed to $e(t)$ and $h_2(t)$.

The control surface (Figure 3c) is calculated by the modified stand-alone P-FC controller model and plotted in MATLAB. Figure 6 shows the results for time domain. While the control surface for P-FC is smoother (max. control 2.6), the control itself is discontinuous and the disturbance cannot be compensated because of the missing integral action. As advantage, this controller reacts faster.

**Résumé:** In this Dymola / Modelica solution, for plant model and for the discrete control structure standard Modelica blocks are used. Fuzzy control has to be programmed directly in textual Modelica code, making use of table function features and of very similar structure for all controller types. The Dymola fuzzy control model can be used standalone, computing control values for arbitrary input values directly. Experiments are controlled by table functions.

**Corresponding Author:** Anton Sodja, *asodja@gmail.com*
Laboratory of Modelling, Simulation and Control (LMSC),
Faculty of Electrical Engineering
Univ. of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenia

Received: February 10, 2007
Revised: February 23, 2007
Accepted: February 28, 2007