

EE682
HW1 - FLC Design

Herman Kolstad Jakobsen
20196493

September 29, 2019

Contents

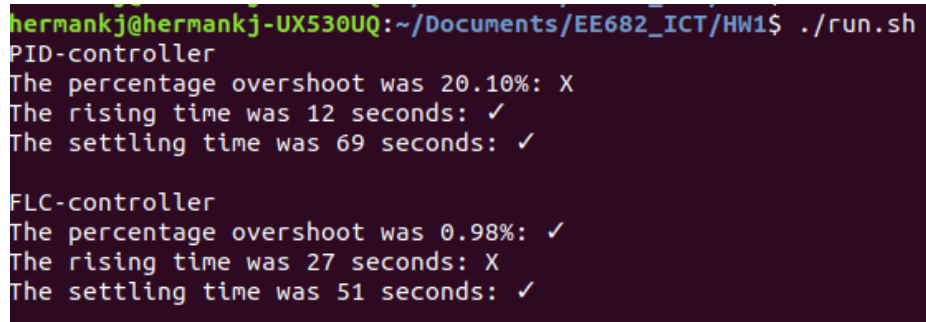
1	Introduction	3
2	Framework	3
3	PID controller	3
4	FLC	4
5	Comparison of controllers	7

1 Introduction

The task was to design a FLC for a system given in the source program. The controller should have an output response of rising time 20.0 sec, 5.0% overshoot, and a settling time of 70.0 sec defined by $\pm 2\%$ of the reference value. A unit step input was given to the controller. Finally, the performance of the FLC will be compared with that of the PID controller.

2 Framework

To start off, a framework was created in order to display the characteristics of the two controllers. The framework consisted of three files; *spec.cpp*, *spec.hpp* and *plotting.m*. By reading the output from the system, *spec.cpp* prints the characteristics of the controllers to the terminal. An example of the printed characteristics is shown in fig. 1. In hindsight, the MATLAB function *stepinfo(x, y, stepvalue)* should rather be used instead of *spec.cpp*. Further, *plotting.m* plots the response of the two controllers together with the membership functions of the FLC. In order to streamline the process of building and running the code, a simple bash script named *run.sh* was created.



```
hermankj@hermankj-UX530UQ:~/Documents/EE682_ICT/HW1$ ./run.sh
PID-controller
The percentage overshoot was 20.10%: X
The rising time was 12 seconds: ✓
The settling time was 69 seconds: ✓

FLC-controller
The percentage overshoot was 0.98%: ✓
The rising time was 27 seconds: X
The settling time was 51 seconds: ✓
```

Figure 1: Framework printing controller characteristics

3 PID controller

After the framework was created, the PID controller was tuned in order to achieve a desired output. To get some rough parameter estimates, the *PID Tuner* from MATLAB was used. In order to use the *PID Tuner* the motor system, with the PID controller, was implemented in *Simulink*. The *Simulink* model is called *pid_controller.slx* and is shown in fig. 2. The discrete transfer function for the motor system was derived by performing a *z*-transformation on the motor difference equation given in the homework description. This resulted

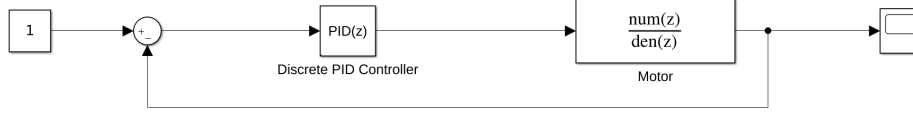


Figure 2: Implementation of motor system with PID controller

Table 1: Parameters for PID controller

Parameters	Value
K_p	1.457
K_i	0.02871
K_d	3.093

in the following equation

$$\frac{y}{m}(z) = \frac{b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2} + b_3 \cdot z^{-3}}{a_0 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2} + a_3 \cdot z^{-3}} \quad (1)$$

where all the consonants are defined in *motor_system.m*.

After the parameters were obtained from the *PID Tuner*, the parameters were fine-tuned. This was necessary due to deviation between the *Simulink* model and the model implemented in *control.hpp*. After trying different types of tuning, it was concluded that it was extremely difficult to meet all of the specifications with a PID controller. The final choice of parameters are shown in table 1.

The step response of the PID controller is shown in fig. 3 and the characteristics are shown in fig. 4. Even though only the specification for settling time was properly met, it can be concluded from fig. 4 that the specifications regarding rising- and settling time were favored over overshoot percentage. Hence, the response in fig. 3 is characterized by a oscillatory behaviour. It is also worth noting that the response has a stationary deviation. By taking this into account, it may be argued that this is not the best possible tuning of the controller. It would require some kind of optimization techniques in order to obtain an optimal tuning and response. These kinds of techniques were not considered in this homework.

4 FLC

Initially, the gain parameters for the FLC were hand-tuned in order to get meet the rising time specification. The tuned parameters are shown in table 2. The response of the controller and its characteristics are shown in fig. 5 and fig. 6, respectively.

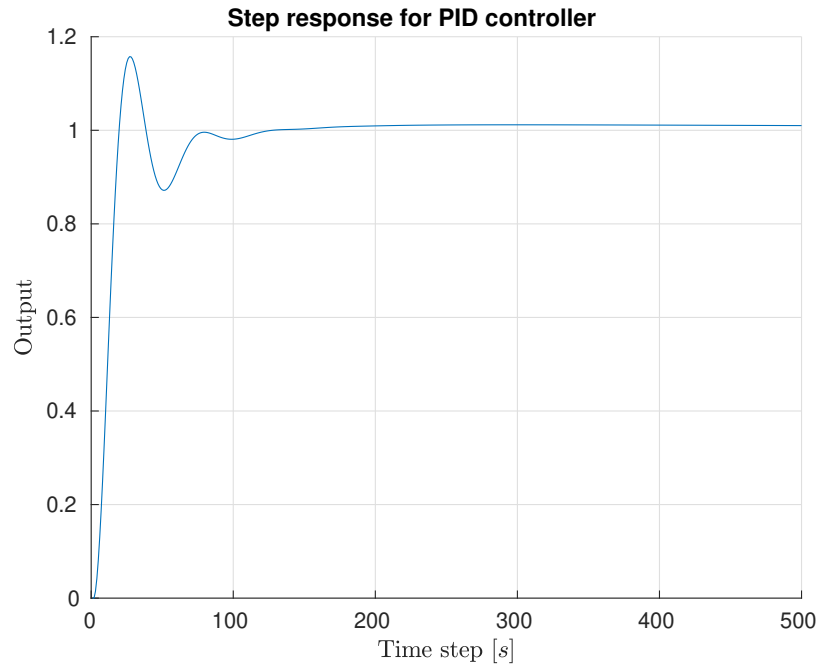


Figure 3: Step response of motor system with PID controller

```
hermankj@hermankj-UX530UQ:~/Documents/EE682_ICT/HW1$ ./run.sh
PID-controller
The percentage overshoot was 15.74%: X
The rising time was 12 seconds: ✓
The settling time was 70 seconds: ✓

Plotting...
hermankj@hermankj-UX530UQ:~/Documents/EE682_ICT/HW1$
```

Figure 4: Characteristics for motor system with PID controller

Table 2: Gain parameters for FLC

Parameters	Value
K_e	1.6
K_{ce}	1.2
K_u	3.6

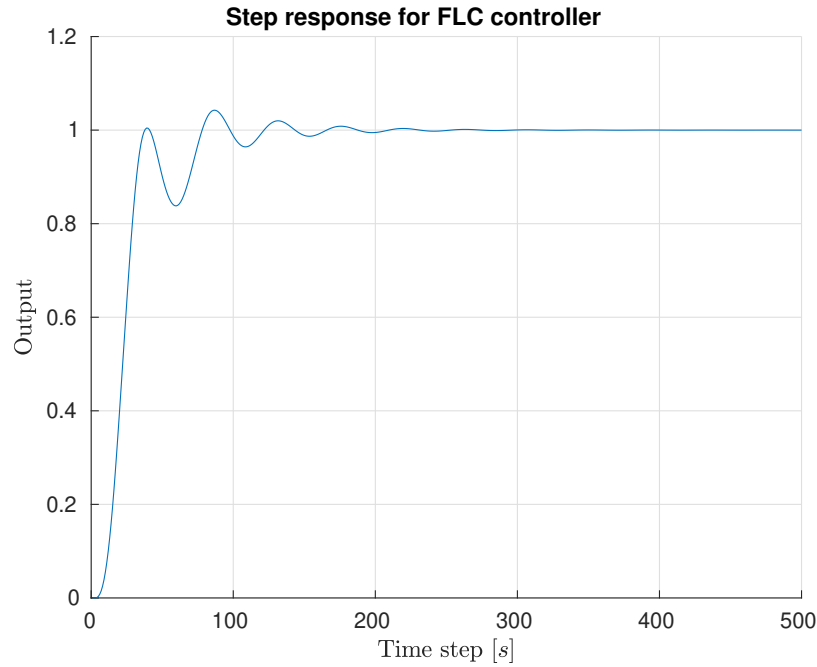


Figure 5: Step response of motor system with FLC which meets rising time specifications

As seen in, fig. 5, the response gets an oscillatory response when the output hits the stepvalue for the first time. This phenomenon makes the settling time exceed the specification and is therefore not desired. In addition, the overshoot specification is not met either. It was thought that the reason behind this response was that the rule base gave out a too big of a negative output when $e = 0$, where e symbolizes the error (i.e $e = \text{stepvalue} - \text{output}$). Hence, the *out*-values corresponding to rules where $e = 0$ were changed to lower valued membership functions, hoping this would shift the oscillatory behaviour upwards the y -axis.

The new rule base is shown in fig. 7, while the new response and characteristics are shown in fig. 8 and fig. 9, respectively. The characteristics in fig. 9 shows that the settling time was somewhat improved, but the specification was still not met. This can be verified by looking at the response shown in fig. 8. The response still has a smaller overshoot than the specification. In conclusion, changing outputs in the rule base did not yield the desired response. It was also tested out to both change the shape of the membership functions and add new rules (which mostly affected small error and change in error) to the rule base, but this did not improve the response. The final membership functions are shown in fig. 10. Ultimately, the settling time and overshoot specifications

```
hermankj@hermankj-UX530UQ:~/Documents/EE682_ICT/HW1$ ./run.sh
FLC-controller
The percentage overshoot was 4.26%: ✓
The rising time was 20 seconds: ✓
The settling time was 116 seconds: X
Plotting...
hermankj@hermankj-UX530UQ:~/Documents/EE682_ICT/HW1$
```

Figure 6: Characteristics of motor system with PID controller with hand-tuned parameters

```
rule[0] = Fuzzy::strength(error, Fuzzy::NB, d_error, Fuzzy::Z0); out[0] = Fuzzy::NB;
rule[1] = Fuzzy::strength(error, Fuzzy::NB, d_error, Fuzzy::PS); out[1] = Fuzzy::NM;
rule[2] = Fuzzy::strength(error, Fuzzy::NM, d_error, Fuzzy::Z0); out[2] = Fuzzy::NM;
rule[3] = Fuzzy::strength(error, Fuzzy::NS, d_error, Fuzzy::Z0); out[3] = Fuzzy::NS;
rule[4] = Fuzzy::strength(error, Fuzzy::NS, d_error, Fuzzy::PS); out[4] = Fuzzy::PS; // Changed out
rule[5] = Fuzzy::strength(error, Fuzzy::NS, d_error, Fuzzy::PB); out[5] = Fuzzy::PM;
rule[6] = Fuzzy::strength(error, Fuzzy::Z0, d_error, Fuzzy::NB); out[6] = Fuzzy::NS; // Changed out
rule[7] = Fuzzy::strength(error, Fuzzy::Z0, d_error, Fuzzy::NM); out[7] = Fuzzy::NS; // changed out
rule[8] = Fuzzy::strength(error, Fuzzy::Z0, d_error, Fuzzy::NS); out[8] = Fuzzy::NS;
rule[9] = Fuzzy::strength(error, Fuzzy::Z0, d_error, Fuzzy::Z0); out[9] = Fuzzy::Z0;
rule[10] = Fuzzy::strength(error, Fuzzy::Z0, d_error, Fuzzy::PS); out[10] = Fuzzy::PS;
rule[11] = Fuzzy::strength(error, Fuzzy::Z0, d_error, Fuzzy::PM); out[11] = Fuzzy::PS; // changed out
rule[12] = Fuzzy::strength(error, Fuzzy::Z0, d_error, Fuzzy::PB); out[12] = Fuzzy::PS; // changed out
rule[13] = Fuzzy::strength(error, Fuzzy::PS, d_error, Fuzzy::NB); out[13] = Fuzzy::NM;
rule[14] = Fuzzy::strength(error, Fuzzy::PS, d_error, Fuzzy::NS); out[14] = Fuzzy::Z0; // changed out
rule[15] = Fuzzy::strength(error, Fuzzy::PS, d_error, Fuzzy::Z0); out[15] = Fuzzy::PS;
rule[16] = Fuzzy::strength(error, Fuzzy::PM, d_error, Fuzzy::Z0); out[16] = Fuzzy::PM;
rule[17] = Fuzzy::strength(error, Fuzzy::PB, d_error, Fuzzy::NS); out[17] = Fuzzy::PM;
rule[18] = Fuzzy::strength(error, Fuzzy::PB, d_error, Fuzzy::Z0); out[18] = Fuzzy::PB;
```

Figure 7: Rule base for FLC

were never met.

An alternative to hand-tuning would be to use optimization techniques to obtain the optimal membership functions, tuning parameters and rule base for the desired response. There are several papers where people have used *Genetic Algorithm* from MATLAB to find the optimal values for a FLC. The use of optimization techniques was considered to be out of scope for this assignment, and therefore never tested out optimization techniques was considered to be out of scope for this assignment, and therefore never tested out optimization techniques was considered to be out of scope for this assignment, and therefore never tested out optimization techniques was considered to be out of scope for this assignment, and therefore never tested out optimization techniques was considered to be out of scope for this assignment, and therefor

5 Comparison of controllers

By looking at the response of the PID controller in fig. 3 and the response of the FLC in fig. 8, it is seen that the PID controller has a larger overshoot and a stationary deviation. In return, the PID controller managed to meet the speci-

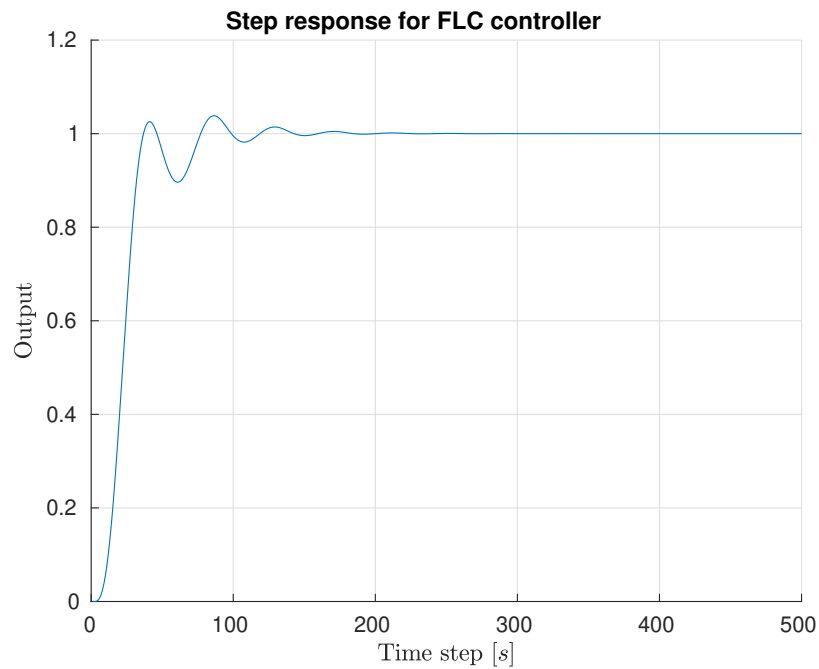


Figure 8: Step response of motor system with finally-tuned FLC

```
hermankj@hermankj-UX530UQ:~/Documents/EE682_ICT/HW1$ ./run.sh
FLC-controller
The percentage overshoot was 3.82%: ✓
The rising time was 20 seconds: ✓
The settling time was 93 seconds: X
Plotting...
hermankj@hermankj-UX530UQ:~/Documents/EE682_ICT/HW1$
```

Figure 9: Characteristics of motor system with finally-tuned FLC

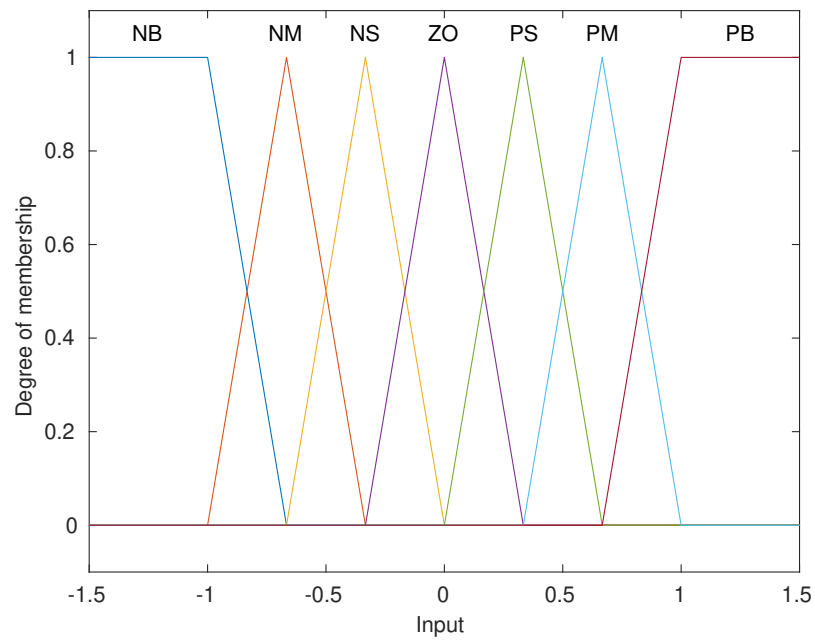


Figure 10: Membership functions for finally-tuned FLC

fication of the settling time. The FLC, even though having a smaller overshoot, has a more oscillatory response. Moreover, the controller only managed to meet the specification of rising time.

The reason behind the FLC under-performing is the tuning. Due to lack of competence, knowledge and experience, the controller was poorly hand-tuned. The PID controller, on the other hand, was tuned using the *PID Tuner* from MATLAB, but it was still difficult to meet all the specifications. By tuning the FLC better, the controller would certainly out-perform the PID controller.

However, this shows one of the downsides of the FLC. While the PID controller only has three parameters to tune, the FLC has three parameters, a rule base and membership functions to tune. This makes it difficult to decide what should be tuned and what should be kept constant. It is also difficult to see how changing the different parameters, the rule base or the membership function affects the response. In order to tune the FLC to meet the specifications, a proper optimization technique should be used.