A report

on

Fuzzy control of an inverted pendulum system

Sashank Krishna S	2019A8PS0184P
Satwik Vats	2019A8PS0194P
Aryan Balyan	2019A8PS0193P

Contents

1 Introduction

When one is asked to think of stabilization problems, inverted pendulum systems are among the first to come to mind. Inverted pendulums are a simple and common example of systems that can be at an unstable equilibrium. Why would we want to study such a system? We would like to, because more complex systems of interest, such as rockets right after launch, and the hoverboard (2-wheeled self-balancing scooter) also involve similar stabilization problems.

The aim of this study is to simulate and visualize the inverted pendulum system, and observe the actions and performances of fuzzy logic controllers with respect to PID controllers. The differential equations stemming from the inverted pendulum system, a PID controller and a fuzzy logic controller were all implemented using MATLAB. Code was also written to visualize the pendulum system in the form of a video.

MATLAB was used, and not Simulink or LabView. This was because the goal of the study was to explore the fundamentals of both PID and fuzzy logic controllers. The MATLAB programming environment allowed for more flexibly visualize the system as a video, as well as allowed for tinkering of the system. Implementing the fuzzy logic controller from scratch allowed for a better understanding of the same than the use of a block diagram implementation.

2 Mathematical setup

The inverted pendulum system could be modelled using the following equation:

$$(I + ml^2)\phi'' + mglsin(\phi) = mlx''cos(\phi)$$

The linearized version of the same is given by:

$$(I + ml^2)\phi'' - mgl\phi = mlx''$$

The model assumes that the pendulum system is a rigid body, and that there is no air resistance. More elaborate models can be formulated in case the factors will be of significance.

The differential equations were converted into difference equations using central the central differencing approach. Both the non-linear and linear differential equations were converted into this form, but only the non-linear versino is presented below. The rest of this study uses the more accurate non-linear model as well.

$$\phi[i] = \frac{mlcos(\phi[i-1])(u[i] - 2u[i-1] + u[i-2]) + mglsin(\phi[i-1])dt^2}{I + ml^2} + 2\phi[i-1] - \phi[i-2]$$

The initial conditions used correspond to zero inputs and a mild disturbance in ϕ. If we simulate this model alone, we can see the inverted pendulum fall down, rise back up, and repeat the motion.

To make the situation more realistic, gaussian noise was also added to this value of $\phi[i]$. i.e., $\phi[i] = \phi[i] + N(0, \sigma^2)$ was performed. This model for noise is not that realistic, since it can be rather erratic. But if our system can be stabilized despite this level of noise, we should be fine. Furthermore, σ can be controlled in the simulation environment.

The Pendulum parameters considered for this study are as follows: [2]

Mass	200 g
Length	30 cm
Moment of Inertia	0.006 kgm^2

3 PID controller

The PID controller was implemented in the velocity form as shown:

$$du = K_p(\phi[i] - \phi[i-1]) + K_i\phi[i] + K_d(phi[i] - 2phi[i-1] + phi[i-2])$$

$$u[i+1] = u[i] + du$$

The PID coefficients used are shown below:

K_p	0.48
K_I	-1.6
K_d	-0.025

Negative coefficients were used because of the unstable nature of the system. The tuning was done by hand, first by trial and error, and then by manually tweaking the coefficients to improve performance. Approaches such as ZN tuning and Cohen-Coon tuning were not applicable since the system at hand is intrinsically unstable.

Tuning could have been done better by using genetic algorithms to guess a set of PID parameters, and then evolve them closer towards the optimal set of parameters. This however was not done due to the lack of time and expertise.

4 Fuzzy logic controller

The designed fuzzy logic controller has 3 simple steps. Fuzzification, inference and defuzzification. First, we defined sets to which the state of the system can belong to. Five sets were defined. VN, N, Z, P and VP. ie Very Negative, Negative, Zero, Positive and Very Positive. The sets classify the deviation angle from the unstable equilibrium point.

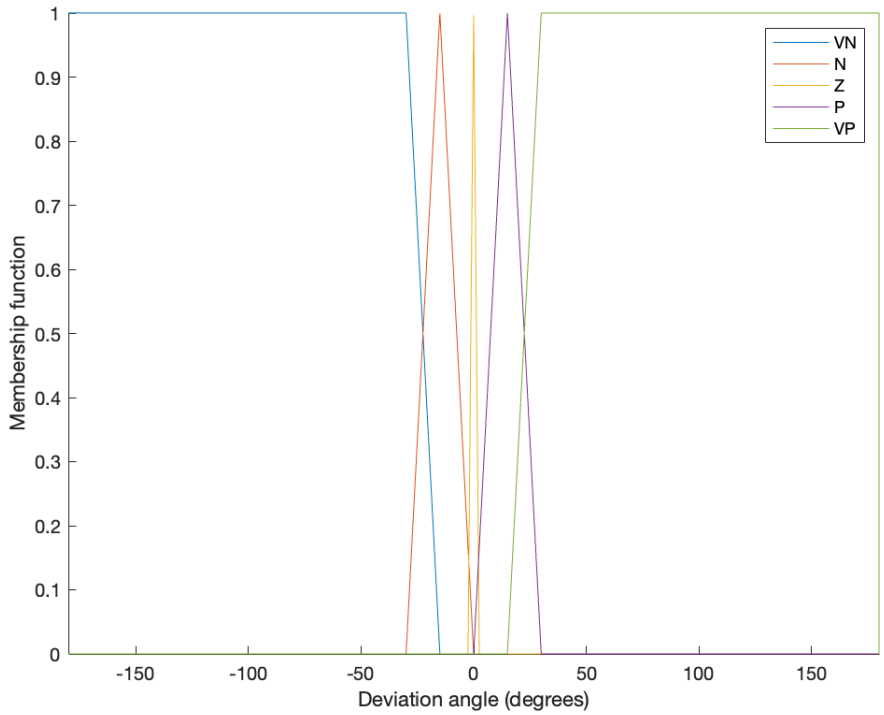

Figure 1: Membership functions of the defined sets

The first step our controller takes, is to determine the extent to which the state of the system belongs to each of the defined sets. We defined the membership functions of the class as shown in fig. 1. The width of the zero class alone is smaller, since we want to take action more even for small deviations from the equilibrium point.

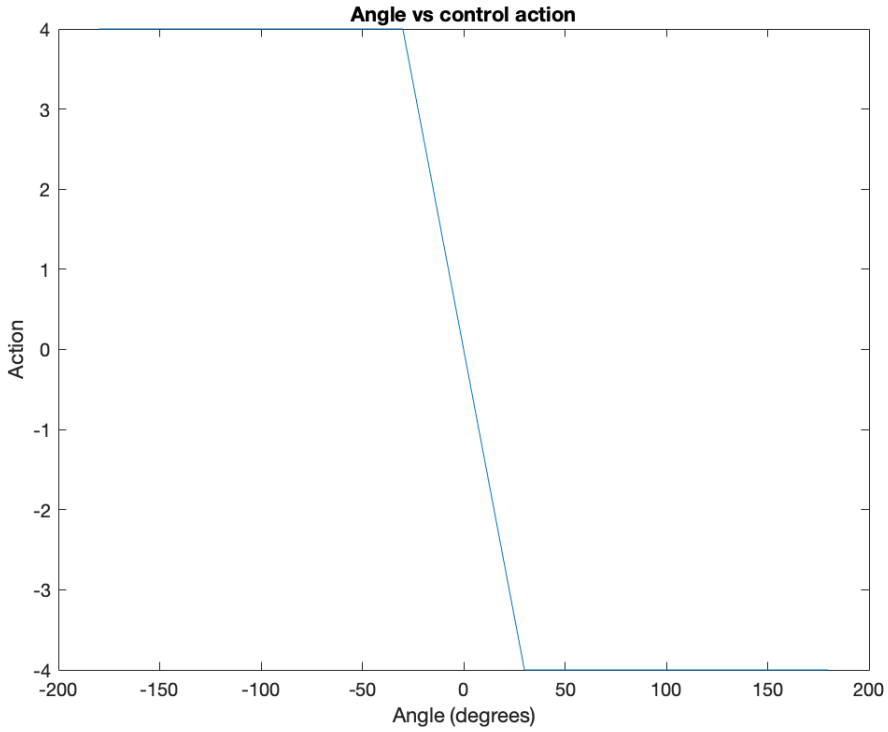

Figure 2: Control action vs System state

We want to define what action our controller should take for each set. Hence we defined control rules for our controller. We then weight this control rule's action by how much the systems state belongs to the set we are considering. We now sum all the actions we need to take, basically for all the sets we have defined. Effectively, the controller responds to the state of the system as shown in fig. 2.

The defined rules are summarized below:

Fuzzy set	Range	Control Action
VN	-180 to -15	4
N	-30 to 0	2
Z	-5 to 5	0
P	0 to 30	-2
VP	15 to 180	-4

Lastly, we want to translate this notion of an output that we have into a real-world action. We need to defuzzify our inference. This was done by using a constant K_{defuz}, which was set to be 0.09 by trial and error.

For the design of this controller, the referenced online course was consulted heavily, since we were unfamiliar with concepts of fuzzy logic [1].

5 Results and discussions

1. PID > FLC
2. Present different situations
3. Link to videos

6 Conclusions

1. PID > FLC under normal conditions 2. large number of sets + customized gains -> smooth / intricate controller, which could outperform PID

7 Learning Outcomes

We learnt how to model simple systems on MATLAB, be it linear or non-linear. We learnt how to stabilize unstable systems with both PID and fuzzy logic control. All three of us were new to fuzzy logic, and in general learnt a lot about how things work. Though we didn't get to implement everything that we learnt, we learnt a lot through this project.

Things we were looking forward to learn but couldn't due to time constraints include standard techniques for tuning PID for unstable systems, how to design fuzzy logic controllers properly for such systems, and multi-input fuzzy logic controllers for the same. Advanced topics that we only glanced at include adaptive or fuzzy PID controllers, and neural networks for the control of such systems.

8 Contributions

The project was undertaken by Sashank Krishna S, Satwik Vats and Aryan Balyan. Sashank wrote the MATLAB scripts and a significant portion of the report, while Satwik and Aryan worked on making the presentation and helped with debugging some of the scripts. Overall, the work split would be as follows:

Sashank Krishna S	40 %
Satwik Vats	30 %
Aryan Balyan	30 %

9 Acknowledgements

We would like to thank Dr. Puneet Mishra and Dr. Surekha Bhanot for teaching the Industrial Instrumentation and Control course, without which this study would not have been possible.

10 References

[1] *Introduction to Soft Computing*. URL: https://nptel.ac.in/courses/106105173.

[2] *Inverted pendulum: System modeling*. URL: https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum%C2%A7ion=SystemModeling.