The use of Fuzzy Logic to balance a LEGO robot based on the Inverted Pendulum Problem

Erik Barrow
Department of Computing
Coventry University
Coventry, United Kingdom
Email: http://www.erikbarrow.com

***Abstract*—Fuzzy logic acts as a suitable controller method for use with the inverted pendulum problem and can be applied in its simplest form to a balancing robot to keep the robot balancing upright.**

Keywords*—*Fuzzy Logic, LEGO, Inverted Pendulum, Defuzzification, Fuzzification.

I. Introduction

The Inverted Pendulum problem is a interesting problem in the field of dynamics involving the balancing of an inverted pendulum [3]. Many different variables need to be taken into account including fall direction, fall speed, motor speed, motor direction, and motor backlash. Other variables on top of these have to be considered in a robot that is also designed to move with an inverted pendulum on it.

Fuzzy logic can be used as a controller to help solve this problem [3], and this paper will investigate the use of fuzzy logic with a self balancing robot that has to balance it's inverted pendulum to stay upright.

The robot that will be used for this problem will be a LEGO NXT robot that has been built much like a segway. The robot has to balance on two wheels and has a number of sensors available (angle, gyro, light, ultrasonic), although only one or two sensors should be needed for balancing.

II. Design and Implementation of the FLC

The fuzzy logic system was implemented on the NXT using the NXC language in the Bricx editor. You can see a fuzzy implementation on JuzzyOnline [8] and can be recreated using the following link http://goo.gl/hNwuYj. The JuzzyOnline implementation is intended as a graphical guide of how the robot decides the output.

The NXC code for the project is available on GitHub at https://github.com/ejbarrow/Fuzzy1. To run the code you will need to install Bricx and the NXT drivers, you will also need an NXT robot on which to download and run the code on.

The fuzzy logic controller for this system has one input and one output. The input is degrees of rotation, which is measured using acceleration on a gyroscope. The outputs of the robot are two motors, which as we are just balancing for this project will be the same, so the fuzzy controller can have just one output value which will be used for both motors.

The light sensor is also used to account for drift in the gyroscopes angle value, if the change in light from calibration is high while the angle is low the program will assume the Gyroscope has drifted and will add or minus a fraction of a degree to correct for drift.

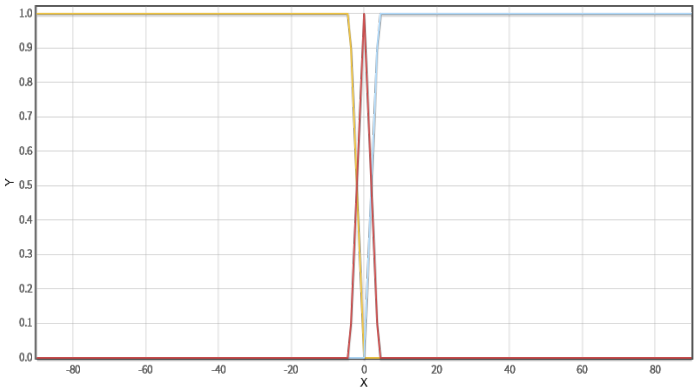

Fig. 1. Fuzzy Inputs (Yellow: Falling Backwards, Red: Stationary, Blue: Falling Forwards)

Figure 1 shows a graphical representation of the fuzzified input. Degrees of rotation are turned into 3 fuzzy sets that represent how the robot is falling. The 3 sets are Falling Forwards, Falling Backwards, and Stationary, which represents the robot is upright and not falling. The x axis is the crisp input as degrees and the y axis is the intensity of the fuzzy sets.

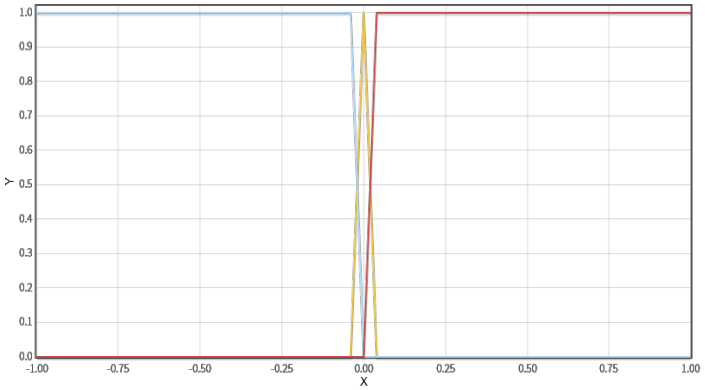

Fig. 2. Fuzzy Outputs (Blue: Motor Backwards, Yellow: Motor OFF, Red: Motor Forwards)

Figure 2 shows a graphical representation of the fuzzified output. There are 3 fuzzy output sets, these are Motor Forwards, Motor Backwards, and Motor off. When defuzzified the value is duplicated and used as the power value for each motor.

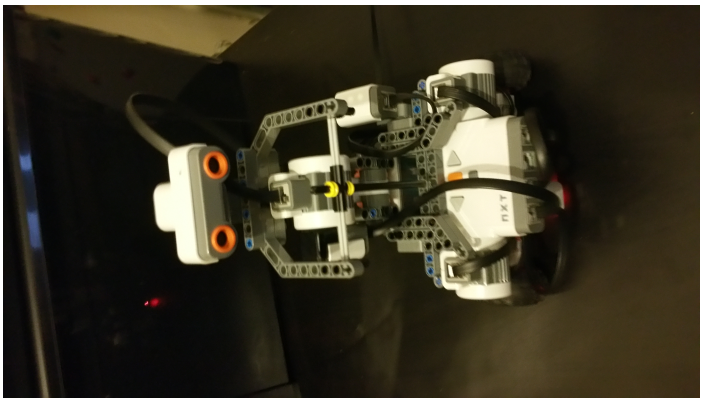

Fig. 3. Robot during balancing

Figure 3 shows a picture of the NXT robot build while it is performing balancing. A video of the robot is also available on Youtube at https://www.youtube.com/watch?v=-0bKrsVGNn0.

Figure 4 shows the rules that I defined for the fuzzy controller. It maps the fall direction to the direction that the motors need to move, and the fuzzy membership value is then defuzzified into a speed.

If Degrees is Backwards **then** MotorSpeed is Backwards

If Degrees is Forwards **then** MotorSpeed is Forwards

If Degrees is Stationary **then** MotorSpeed is Off

Fig. 4. Fuzzy Rules

In the NXC program the defuzzification process is done by taking the fuzzy membership value and multiplying it by 100 or -100 to get the power range. The forwards and backwards fuzzy sets are added as they do not cross over, so only one or the other will be positive at any time. The power is then multiplied by the inverse of the stationary fuzzy membership set. For example if the membership to forwards is 1 (100 percent), the membership to backwards is 0 (0 percent) and the membership to stationary/off is 0.75 (75 percent) then the power returned will be 25 percent of the forwards fuzzy set, which is 25 percent power.

With the addition of acceleration as an input to the Juzzy-Online controller we can also infer that a higher acceleration in one direction will require more power to exit. This also allows us to infer that a slow fall doesn't need as much power. Figure 5 shows the added input and figure 7 shows the surface plane of the controller. Rules where added to this controller to add the extra functionality as shown in figure 6. This controller can be viewed by visiting http://goo.gl/mnNZXC.

We can also add positional drift into the mix as another input. Using the motors as a rotation sensor to tell how far the robot has travelled forward or backwards during balancing.

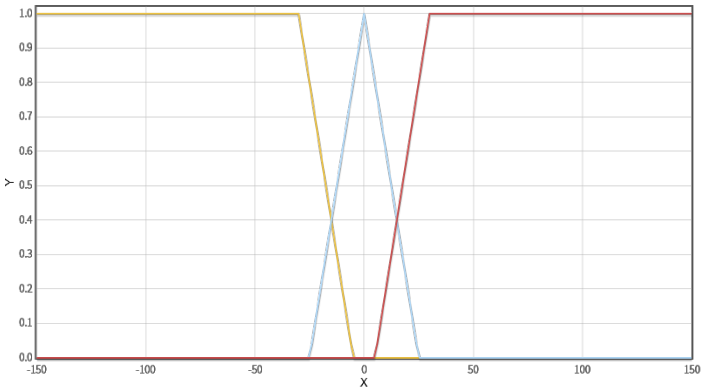

Fig. 5. Acceleration as degrees per second

When the robot is stationary and not accelerating we can use this time to move the robot back toward the starting position if it has drifted away while it was balancing. Without this the robot could have a directional bias and continue to head in that direction. The Fuzzy controller is available to view on JuzzyOnline at goo.gl/9KMeOt.

Figure 9 shows the new fuzzy input to the system, it looks similar to the other input sets as it has drift in either direction as well as a centre set to say that it has reached the home position. The new rule set is shown in figure 10, and the surface plane can be seen in figure 11. In the plot you can see the effect that the new rules have on the output around the centre point where degrees and acceleration are 0.

III. Design Justification

The fuzzy sets I chose for input was the direction of fall (forwards, backwards, upright), I chose this as it indicates how upright the robot is. While the robot can be falling forwards or backwards we want to know by how much the robot is falling, so a fuzzy set allows us to see how much the robot is falling in any direction.

The output sets that I chose where the speed of the motors. A negative speed inferred a reverse direction of that speed. This prevented the need for two outputs (speed & direction) as an amount of direction would in this case indicate a speed and make the need for a second output set. The fuzzy membership for motor speed was off, forwards, and backwards. These sets make sense as the motor can either be off or moving on one of two directions. A motor cannot move in two directions at once hence forwards and backwards do not overlap, much like the input sets for fall direction do not overlap.

For defuzzification of the robot I used a quick calculation that gives a result similar to centroid defuzzification. The calculation involved talking the speed in either direction and multiplying by the inverse of the stationary membership set. This calculation was quick meaning that it would not cause a delay in the robots reaction time, which could cause the robot to fall even further. This essentially slows the motor down as it returns to the upright position so that it does not over compensate and fall in the opposite direction.

The defuzzification of the second controller on JuzzyOnline used centroid defuzzification which is a common defuzzifi-

1. **If** Degrees is Backwards **then** Direction is Backwards
2. **If** Degrees is Forwards **then** Direction is Forwards
3. **If** Degrees is Stationary **then** Direction is OFF
4. **If** Acceleration is Fast_Back **then** Direction is Backwards
5. **If** Acceleration is Fast_Forwards **then** Direction is Forwards
6. **If** Acceleration is Slow **then** Direction is OFF
7. **If** Degrees is Backwards and Acceleration is Fast_Forwards **then** Direction is Backwards
8. **If** Degrees is Forwards and Acceleration is Fast_Back **then** Direction is Forwards
9. **If** Degrees is Backwards and Acceleration is Slow **then** Direction is OFF
10. **If** Degrees is Forwards and Acceleration is Slow **then** Direction is OFF
11. **If** Degrees is Stationary and Acceleration is Fast_Back **then** Direction is OFF
12. **If** Degrees is Stationary and Acceleration is Fast_Forwards **then** Direction is OFF

Fig. 6. Extra Fuzzy Rules for Controller 2

cation method. The defuzzified output of the controller is appropriate unlike the height defuzzification method which provides an inappropriate and ineffective output.

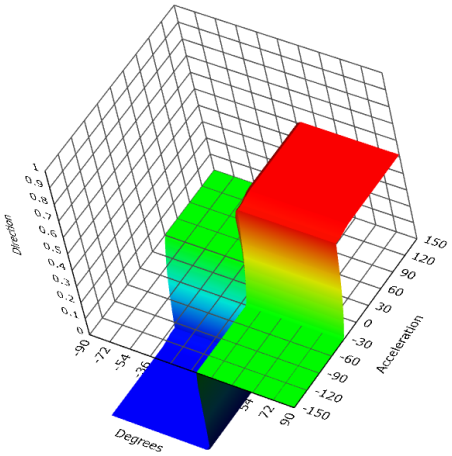

Fig. 7. Surface Plane of fuzzy controller

The rules I have chosen are logical for the inverted pendulum problem. When the top of the inverted pendulum starts to fall the robot needs to move the bottom of the pendulum where the Axis of rotation is, in the same direction of the fall. So when the fall is forwards, the robot will move forwards to compensate and will do the same in the opposite direction. The rule that makes the robot stationary when the robot is upright is needed to prevent the robot from moving constantly and oscillating, this rule also adds a deceleration effect on the robot as it nears back to its upright centre of gravity, which prevents a immediate fall in the opposite direction.

The fuzzy logic controller uses the Mamdani inference system which maps the inputs to the outputs of the controller. Mamdani is one of the most used inference systems in fuzzy logic and it is logical to use for this problem. Figure 8 shows a Mamdani map of an output from one of the fuzzy controllers, you can see the combined membership of stationary speed and forwards speed in this figure.

The third controller is the next logical step in the balancing problem. If you stray too far from the start you may find that the space you are moving around ends. The input fuzzy sets (Figure 9) can be justified as the distance travelled by the robot will be given in either degrees or wheel rotations in either a positive or negative direction from the calibration point.

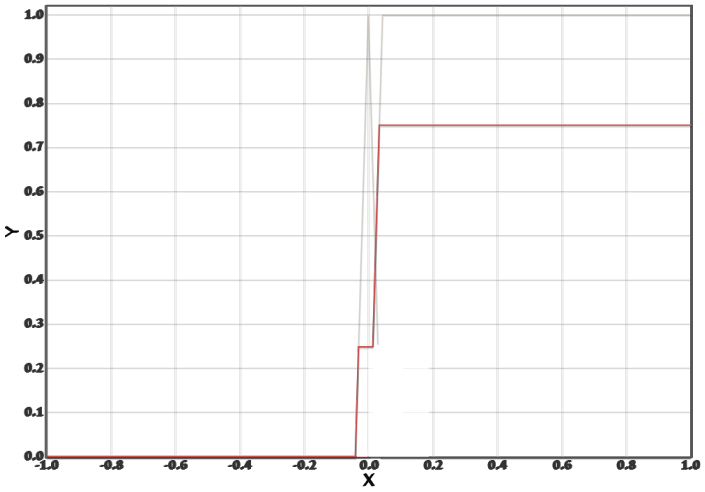

Fig. 8. Mamdani Inference

IV. Analysis of Controller Performance

The controller achieved a acceptable performance in balancing as can be seen in the video evidence of the robot balancing. The robot used the first controller design with just the angle as an single input and the motor power (positive and negative) as a single output.

The second controller is similar to the first except that it has a re-enforcing effect to make sure that the robot will be heading faster to upright itself if it is falling faster. The surface plot in figure 7 shows an effective activation of the rules, and that a small change in fall direction is immediately countered by moving the motor an appropriate speed in the fall direction.

The plot shows that as acceleration and degrees increase the amount of power on the motors increase in the same direction of the fall. You can see from the plot that the increase happens very quickly as a fall in either direction quickly requires more power to correct itself. The plot itself is much like a inverted pendulum, where the speed quickly decelerates towards the centre of gravity and then quickly accelerates in the opposite direction.

The third controller shows that the addition of the positional drift has only a very small impact on the output (Figure 11), and looks like a small dent on the surface plane. While it may not look like much, a big impact in speed would cause the robot to have to start balancing again, and possibly drift even further, so it is far better to regain the starting position slowly.

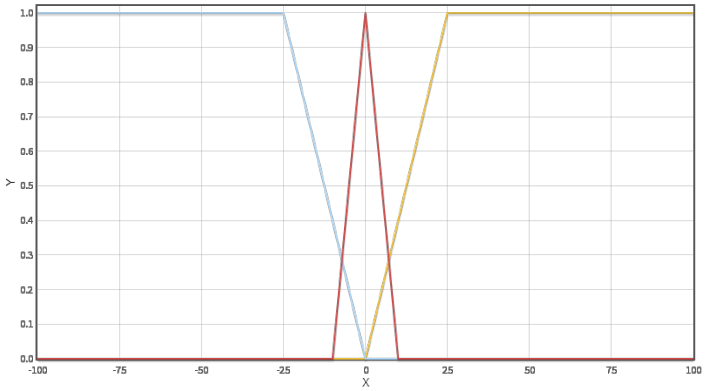

Fig. 9. Positional Drift

V. Description & Explanation of FLC Hybrid Systems

The use of Hybrid and/or intelligent systems (computational intelligence) allow for a system to learn and adapt. A traditional fuzzy system requires rules to be set manually by the programmer, but with hybrid fuzzy systems that use intelligence, the rules can be tweaked to learn from the outputs.

One possible hybrid is a Neo-Fuzzy Neural Network [2], a Neo-Fuzzy Network uses the rules of a fuzzy system as a replacement to the activation functions of neurons. The weights of the network can then be trained to improve accuracy of the rules of the fuzzy system. This type of hybrid system requires some sample data to compare inputs and outputs to find any error in the fuzzy network.

Another hybrid fuzzy system that can be used is a Fuzzy Logic PID controller [1]. A PID (proportional-integral-derivative) controller contains three controllers [7], proportional, integral, and derivative. In a fuzzy PID system these controllers are fuzzified [6]. PID controllers can be applied quite successfully to the inverted pendulum problem that this paper discusses, so a hybrid Fuzzy PID system could work very well in balancing the robot built for this paper.

Fuzzy PID or Fuzzy PD controllers can also be used with quad rotor helicopters to control the flight of the aerial vehicle. The fuzzy controller would be able to keep the drone stable and hovering whilst also moving in the required direction [4]. This application has been looked at by several people, and videos of dual blade copters balancing around an axis.

1. **If** Degrees is Backwards **then** Direction is Backwards
2. **If** Degrees is Forwards **then** Direction is Forwards
3. **If** Degrees is Stationary **then** Direction is OFF
4. **If** Acceleration is Fast_Back **then** Direction is Backwards
5. **If** Acceleration is Fast_Forwards **then** Direction is Forwards
6. **If** Acceleration is Slow **then** Direction is OFF
7. **If** Degrees is Stationary **and** Acceleration is Fast_Back **then** Direction is OFF
8. **If** Degrees is Stationary **and** Acceleration is Fast_Forwards **then** Direction is OFF
9. **If** Degrees is Stationary **and** Acceleration is Slow **and** Positional_Drift is Home **then** Direction is OFF
10. **If** Degrees is Stationary **and** Acceleration is Slow **and** Positional_Drift is Forward_drift **then** Direction is Backwards
11. **If** Degrees is Stationary **and** Acceleration is Slow **and** Positional_Drift is Drift_Backwards **then** Direction is Forwards

Fig. 10. Rule Set for Controller 3

Fuzzy Logic Controllers can also be combined with genetic algorithms [5]. The genetic algorithm can be applied to the membership function and the defuzzification method to fine tune the fuzzy logic controller. Sample input and output data are used to find the error between the output and the preferred output to evaluate the effectiveness of the genetically altered fuzzy memberships. This is an advantage for a fuzzy controller as the outputs are tuned, and would be good for the inverted pendulum as it takes a lot of manual fine tuning to get the robot to balance well.

VI. Conclusion and Recommendations

Fuzzy Logic has shown to be a suitable controller to balance an inverted pendulum for use with a LEGO NXT robot. The robot is able to balance for a reasonable amount of time and account for gyroscopic drift.

Future work on this robot would be to include the positional drift controller into the robots calculations. This involves measuring the distance that the wheels have travelled while balancing and making the robot return to the starting position when it regains balance.

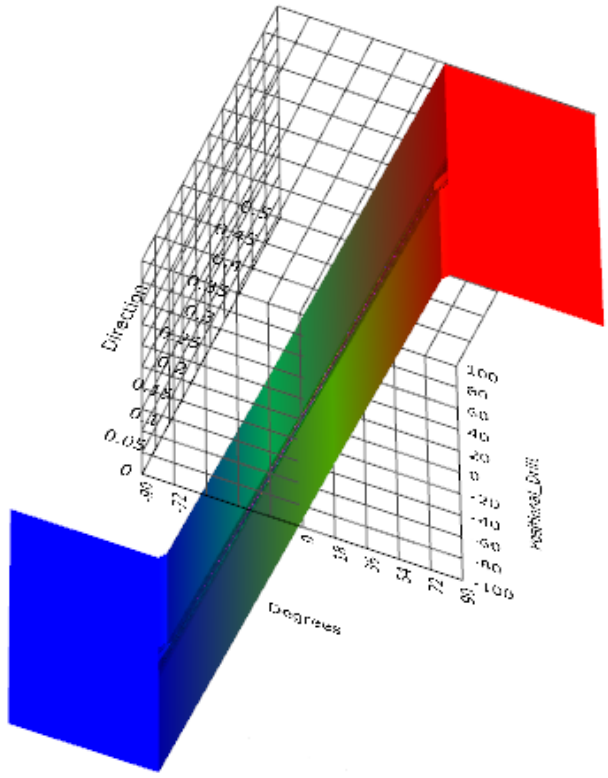

Fig. 11. Surface Plane for fuzzy controller 3

There is also room to expand the robots driving capabilities using the third motor to control the driver and allow the robot to be driven forwards and backwards, with the possibility of editing the fuzzy controller to take turning left and right into account, separating each motor into a separate output.

The robot currently has to be calibrated perfectly upright to balance well, an evolutionary algorithm may be able to learn

and compensate for small errors in calibration to find the true upright value.

Another improvement to the controller would be to separate the output into each motor to allow the a new turn controller input. this would allow the user to steer the balancing robot or allow the robot to steer itself in self controlled applications such as obstacle avoidance or line following.

REFERENCES

[1] Thomas Brehm and Kuldip S Rattan. Hybrid fuzzy logic pid controller. In *Aerospace and Electronics Conference, 1993. NAECON 1993., Proceedings of the IEEE 1993 National*, pages 807–813. IEEE, 1993.

[2] Kwang Bo Cho and Bo Hyeun Wang. Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction. *Fuzzy sets and systems*, 83(3):325–339, 1996.

[3] Mohamed I El-Hawwary, Abdel-Latif Elshafei, Hassan M Emara, and Hossam A Abdel Fattah. Adaptive fuzzy control of the inverted pendulum problem. *Control Systems Technology, IEEE Transactions on*, 14(6):1135–1144, 2006.

[4] Bora Erginer and Erdin Altu. Design and implementation of a hybrid fuzzy logic controller for a quadrotor vtol vehicle. *International Journal of Control, Automation and Systems*, 10(1):61–70, 2012.

[5] F. Herrera, M. Lozano, and J.L. Verdegay. Tuning fuzzy logic controllers by genetic algorithms. *International Journal of Approximate Reasoning*, 12(34):299 – 315, 1995.

[6] Jan Jantzen. Tuning of fuzzy pid controllers. *Technical University of Denmark, Department of Automation, Bldg*, 326, 1998.

[7] E. J. Mastascusa. Pid controllers - general pids.

[8] Christian Wagner, Mathieu Pierfitte, and Amandine Pailloux. Juzzyonline.