**Fuzzy regulator for UAV**

Simulation approach

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

One of the most important parameters of every flying vehicle is its ability to reach and maintain desired attitude and height without falling in oscillations and with as short delay as possible. This is normally achieved with a classic PID algorithm which proved to be insufficient for more advanced UAV, especially for tasks like transporting cargo which imply changing mass during flight.

Jakub Mnich – MikroCpp (Universal Flying Platforms) 26 October 2016

1. **Defining the problem**

It is often difficult to test ideas even for simple engineering solutions when access to evaluation platforms is limited or none. Then the most reasonable way is to use simulations.

Simulations are especially useful as an evaluation tool for regulators used in flying robots as platforms of this kind are expensive, relatively difficult to obtain and extremely vulnerable to crashes. The additional advantage of simulations, in this case, is very simple physical model of a multirotor flying robot – at least as long as aerodynamic properties of the vehicle are ignored. And they can be ignored as the main concern of regulators is the attitude of a vehicle which is negligibly affected by the aerodynamics – those apply mainly to transitional movement.

There are currently a few basic types of regulator algorithms that are in use. The dominant one is classic PID algorithm which owes its popularity to extremal simplicity of implementation. However, PID works best in linear systems while multirotors are highly nonlinear and therefor classic PID handles difficult situations, like change in mass or extreme errors, rather poorly. Due to this issues a need arises for a new algorithms which can replace PID and eliminate its weaknesses.

Numerous algorithms are investigated for future use in modern multirotors. **[1]** One of the most promising is regulator based on fuzzy logic (later called: fuzzy regulator) which eliminates most of the PID drawbacks without consuming much CPU time.

1. **Materials and methods**

The environment chosen for this experiment is Python 3.5 with Anaconda toolchain form Continuum Analytics.**[2]** Script was edited with the use of PyCharm IDE from JetBrains.**[3]** These tool proved to be robust and have a very friendly learning path.

All visualizations were prepared with Matplotlib package **[4]** and Microsoft Mathematics.

1. **Model**

The first step in simulating any physical system is creating its mathematical model. Clear comparison of regulators requires only one degree of freedom so it is not needed to extend the model to embrace a whole multirotor.

In this experiment a model of an arm balanced with two thrusters was implemented and used to evaluate algorithms.

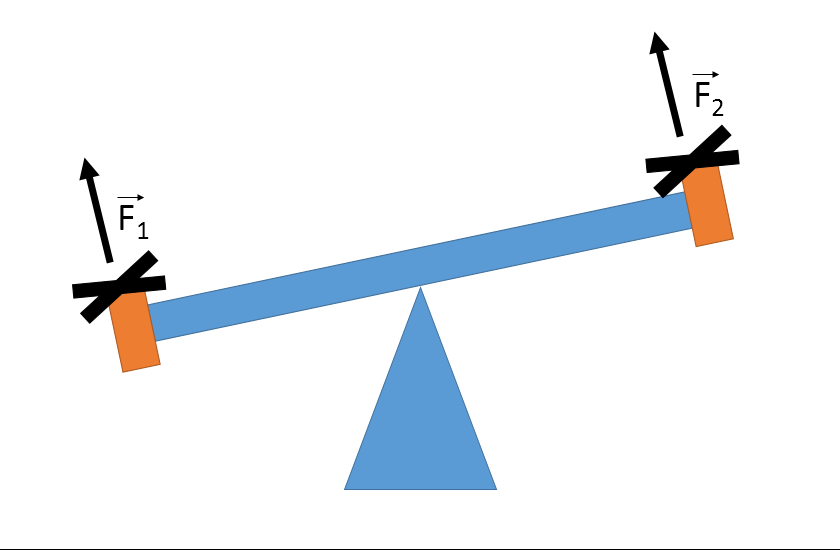


Fig. 1 - balancing arm with two rotors

An arm from figure 1 can be easily modelled with the following set of basic equations:

Now that angular acceleration (ε) can be derived from the mass, length and inertial moment of the arm the state (angle and angular velocity) of it can easily be calculated with two thrusts (F­1 and F2) as inputs. Applied forces cannot be greater than 10N per rotor and the arm’s mass is 1kg.

1. **Reference regulator**

A reference solution is needed to clearly indicate success or failure of the experiment. The obvious choice in this case is a PID regulator applied to the same physical model.

In the course of the experiment a basic PID algorithm was designed and evaluated. **[5]**

This algorithm is fairly simple and is not an object of interest for this paper so it’s detailed description isn’t presented.

1. **Basic fuzzy algorithm implementation [6]**

Systems described with a fuzzy logic are capable of “being in a few states at the same time” while each state is associated with a value of confidence that this state is adequate. This property allows fuzzy controllers to approximate the real state of the system by combining multiple discrete states with their certainty level and calculate optimal output based on reactions recommended for each combination of active discrete states.

The first implementation of fuzzy algorithm was simplified to the maximal possible level. It was created mainly for educational purpose.

The very first step in designing a fuzzy regulator is defining its membership functions for input and output. In this case the regulator has two inputs: angle error and angular velocity, and one output – recommended thrust. The first version of fuzzy regulator uses 5-stage triangular membership functions for both inputs (Fig.2) and similar 4-stage functions for thrust recommendations witch start at 0[N] and end at 10[N]. Negative side does not need to be covered in this case as motors cannot generate negative thrust. It can only be assumed that it negative thrust is recommendation for the first engine while positive for the second. First model does not include motors’ own inertia.

Fuzzy regulator requires also a set of rules which define right output for every possible set of input values. Rules used in this early version are presented in a table (Tab.1).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Angular velocity** | | | | | |
| **Angle** |  | -2 | -1 | 0 | 1 | 2 |
| -2 | 3 | 3 | 2 | 1 | 0 |
| -1 | 3 | 2 | 1 | 0 | -1 |
| 0 | 2 | 1 | 0 | -1 | -2 |
| 1 | 1 | 0 | -1 | -2 | -3 |
| 2 | 0 | -1 | -2 | -3 | -3 |

Tab. 1 - rules for thrust recommendations (version for 5-stage inputs)

The last component required to launch this fuzzy regulator is defuzyfication engine which would combine all recommendations from active rules into one numeric output. Two different algorithms were considered for this purpose: COG (Centre Of Gravity) and CA (Centre-Average) calculation. They both give similar results but COG is considered superior to CA. However COG requires quite memory and calculation-intensive implementation of membership functions as they need to provide full information on the cures’ shape and therefor CA was chosen for the first implementation of defuzyfication mechanism.

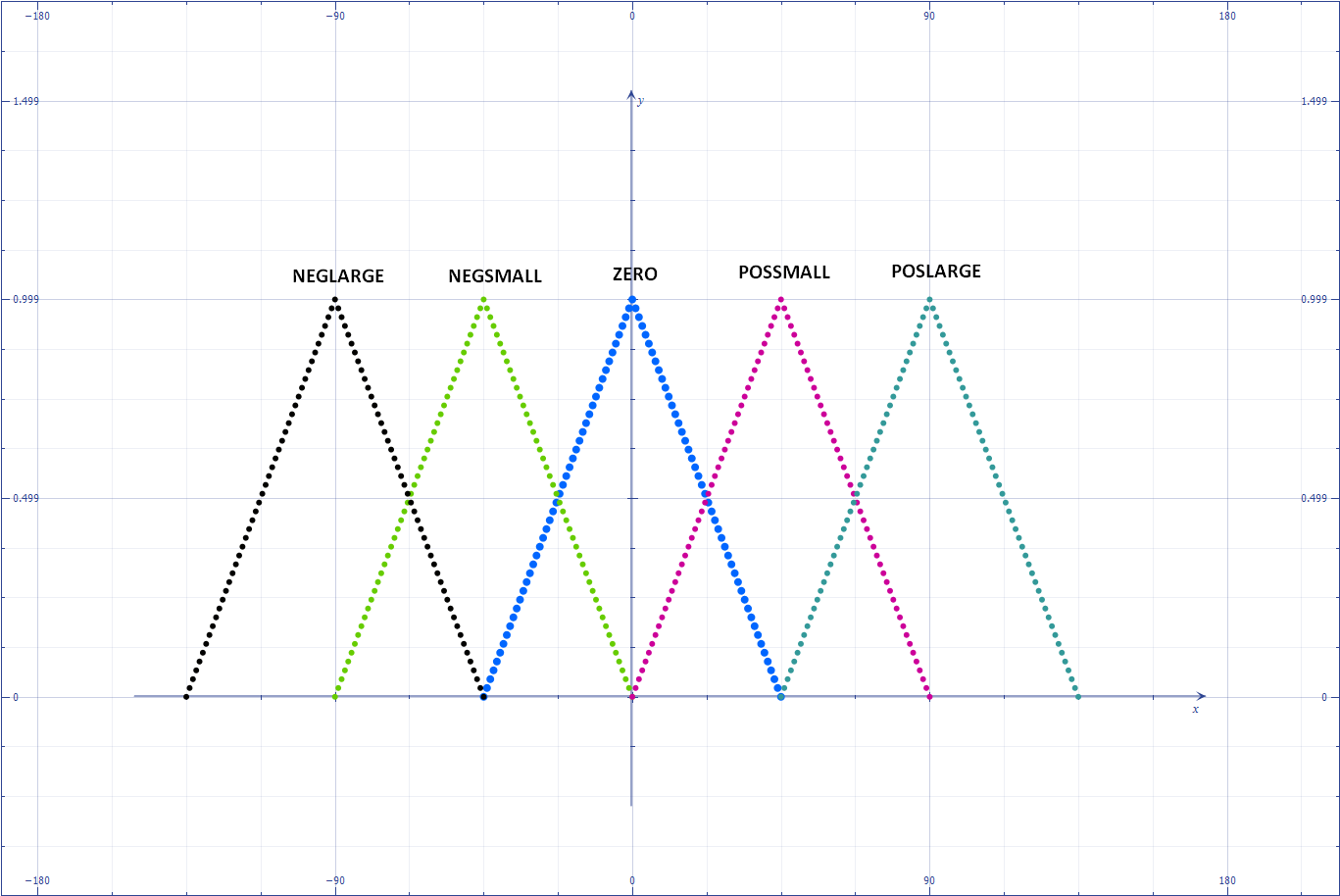
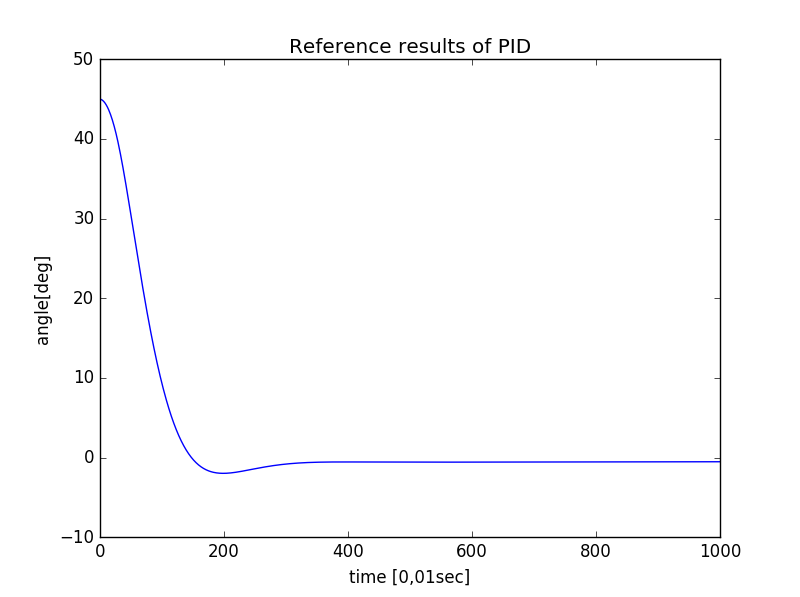


Fig. 2 - membership functions for angle error (for angular speed the extreme values are +-60deg/s and the rest is proportional)

1. **First results**

The first result obtained with the simple fuzzy regulator discussed above is showed on Fig.3.

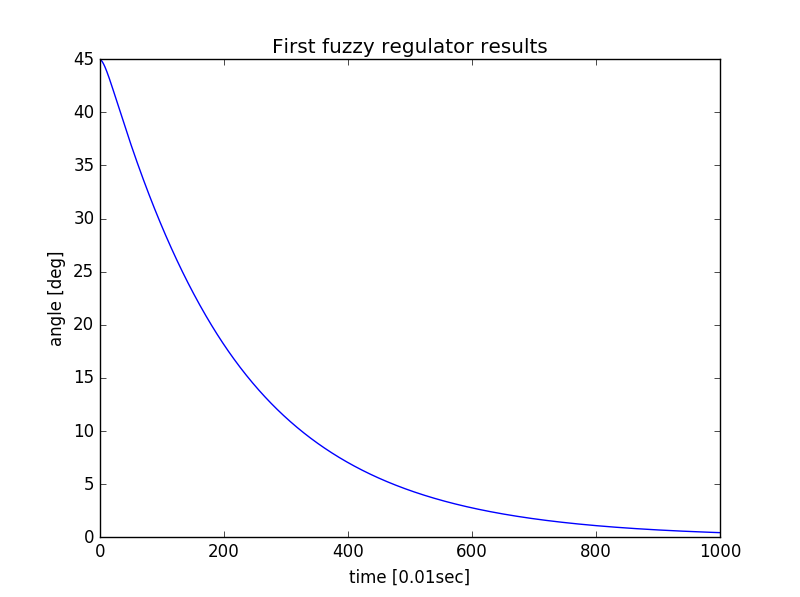
Two numerical parameters will be used to compare algorithms: Area Under the Curve (AUC) and Time taken to reach the Target Value (TTV) with 0,5 degree accuracy. All algorithms will be compared based on their results in stabilizing an arm rotated initially by a large angle – 45 degrees.

Fig. 3 - simulation results of first regulator. AUC = 98,6; TTV = 10s

PID algorithm tested on the same physical model provided output presented on a graph below (Fig.4).

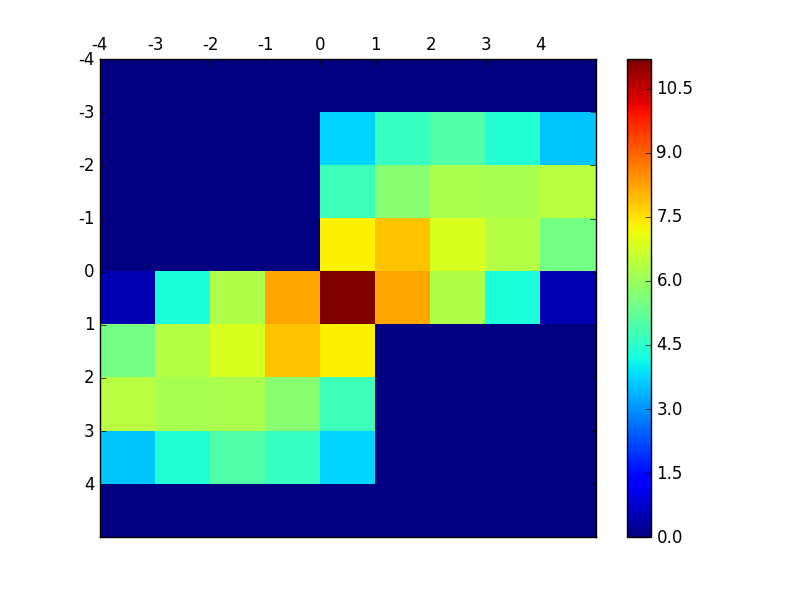
Fig. 4 - reference results of PID. AUC = 38,0; TTV = 2,7s

These results clearly indicate that fuzzy algorithm requires major upgrades in order to get at least as good results as much simpler in implementation and less computationally complex PID.

1. **Reflections on improvements**

The main issue of the first prototype of fuzzy algorithm is the time it needs to reach a desired value. It has no visible tendency to fall into oscillations or overshoot the setpoint value.

The possible solution would be to increase the number of input stages in fuzzyfication engine (increase the number of fuzzy states that input can be encoded into). Together with increased number of output fuzzy states this should provide enough flexibility to allow the algorithm for highly nonlinear behaviour – it needs to accelerate faster and break more intensively in order to reach a desired value faster and still not fall into oscillations.

The next version of algorithm has 9 input fuzzy states instead of 5 (which results in an increase of number of rules from 25 to 81) and 9 output fuzzy states.

1. **Results with increased number of fuzzy states**

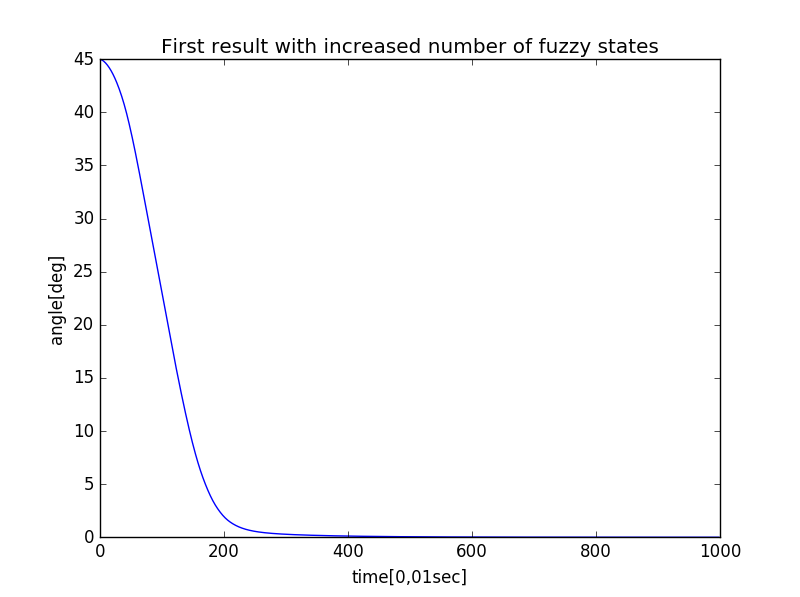
First result obtained with a new version of the algorithm is presented on a graph below (Fig.5).

Fig. 6 - heatmap of rules' usage frequency. Unit is arbitrary and values are proportional to logarithms of this frequency.

Fig. 5- results of the improved fuzzy regulator. AUC = 48,4; TTV = 2,7s

A new set of rules is presented in Tab.2. These rules were prepared based only on author’s own intuition and were not calibrated.

This result clearly shows that improving non-linear capabilities of fuzzy regulator is the right way forward. It also shows that the regulator can be completely prevented from falling into oscillations even if the arm reaches very high angular velocities during the swing.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Angular velocity** | | | | | | | | | |
| **Angle** |  | -4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 |
| -4 | 8 | 8 | 8 | 7 | 8 | 4 | 4 | 5 | 6 |
| -3 | 8 | 7 | 7 | 5 | 6 | 3 | 3 | 4 | 2 |
| -2 | 8 | 7 | 5 | 3 | 4 | 2 | 2 | 3 | 0 |
| -1 | 7 | 6 | 4 | 2 | 2 | 1 | -1 | -2 | -3 |
| 0 | 7 | 5 | 3 | 2 | 0 | -2 | -3 | -5 | -7 |
| 1 | 3 | 2 | 1 | -1 | -2 | -2 | -4 | -6 | -7 |
| 2 | 0 | -3 | -2 | -2 | -4 | -3 | -5 | -7 | -8 |
| 3 | -2 | -4 | -3 | -3 | -6 | -5 | -7 | -7 | -8 |
| 4 | -6 | -5 | -4 | -4 | -8 | -7 | -8 | -8 | -8 |

Tab. 2 - intuitively prepared rules for the second prototype

In order to provide a better understanding of the experiment a heatmap reflecting frequency of each rule’s usage was created (combined results of two start points at 45 and -45 degrees for symmetry check).

Due to the character of the experiment only two quarters of the entire table are in use. It is important to notice that this heatmap will significantly vary in a real application. Its aimed only at improving observational knowledge on behaviour of fuzzy algorithms.

1. **Automatic calibration attempt**

Calibrating the new version of regulator is much more difficult than the previous one as it requires 81 rules instead of 25 and the process takes significantly more time than in case of PID which has only 3 parameters.

This issue can be addressed in three different ways:

1. Brute force – generating and testing all possible combinations of parameters.
2. Simple AI optimization (genetic algorithms)
3. Introduction of a limited set of parameters which all rules would depend upon.

The most promising and versatile approach would be to introduce solution c) but an attempt to do so ended in failure. It occurred to be too difficult to subject all rules to a few parameters and preserve the unique ability of the regulator to handle very high angular velocities without oscillations.

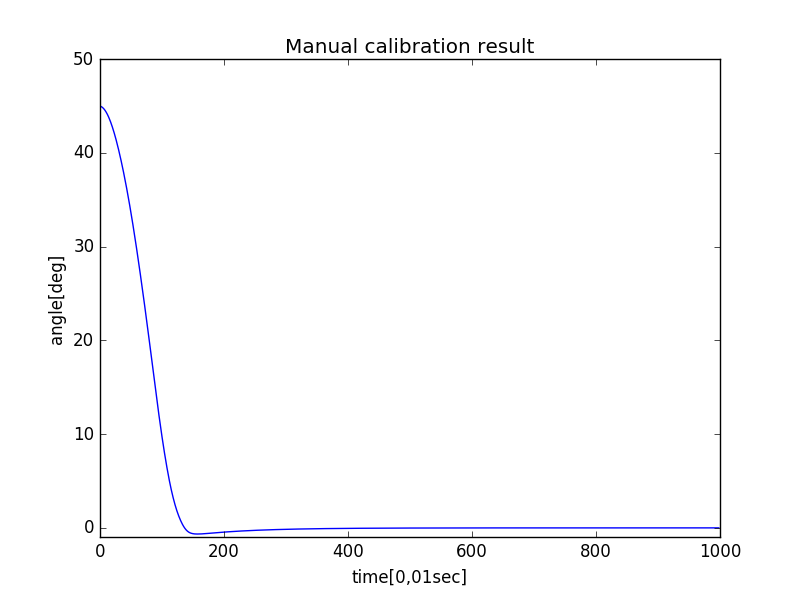
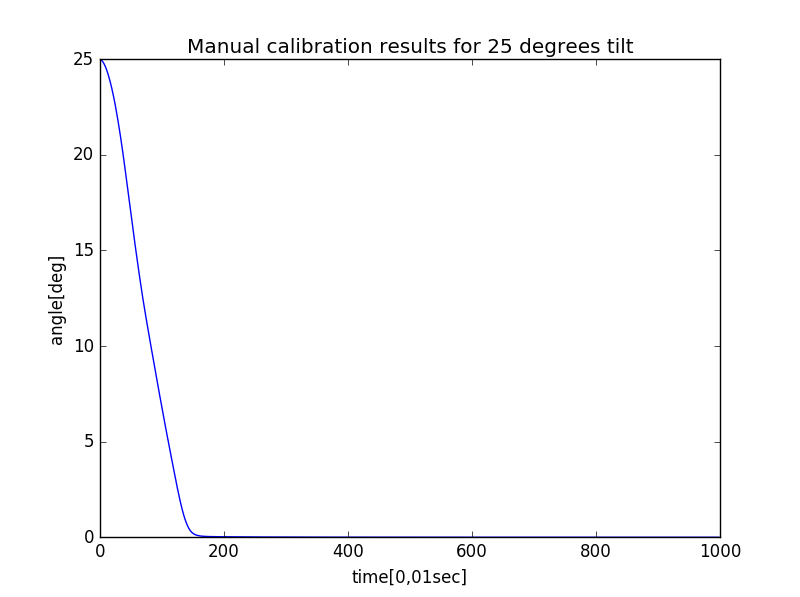
The second best solution – b) would require reimplementing a genetic algorithm developed earlier by the team members but it wouldn’t be able to handle 81 parameters in reasonable time so this option was rejected due to overwhelming amount of required calculations.

Fig. 7 - result after manual calibration. AUC = 34,2; TTV = 1,5s

As b) and c) were rejected it became clear that the least “elegant” option - a) Is the only way forward. However, it is absolutely impossible to test all possible solutions as there are 1781 of them. The problem was therefore simplified: rules table was split into 9 modules with 9 rules in each of them and during a single iteration the algorithm would attempt to increase and decrease all thrust recommendations by 1 instead of testing all possibilities. This would require only 19 683 operations for each module which gives less than 180 000 operations per full iteration (with all 9 modules covered in random order). This version of brute force would, most likely, not generate optimal result but it should at least improve the existing set of rules.

This improved version of brute force algorithm was implemented but with a little success as processing an individual module took 80-120ms which meant that accomplishing calculations would take at least two weeks.

After the last failure concept of fully automatic calibration of rule base was abandoned.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Angular velocity** | | | | | | | | | |
| **Angle** |  | -4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 |
| -4 | 8 | 8 | 8 | 8 | 8 | 8 | 7 | 6 | 5 |
| -3 | 8 | 8 | 8 | 7 | 8 | 6 | 6 | 5 | 4 |
| -2 | 8 | 8 | 7 | 6 | 6 | 5 | 4 | 3 | 2 |
| -1 | 8 | 8 | 6 | 5 | 5 | 5 | 3 | -6 | -8 |
| 0 | 8 | 8 | 6 | 5 | 0 | -5 | -6 | -8 | -8 |
| 1 | 8 | 6 | -3 | -5 | -5 | -5 | -6 | -8 | -8 |
| 2 | -2 | -3 | -4 | -5 | -6 | -6 | -7 | -8 | -8 |
| 3 | -4 | -5 | -6 | -6 | -8 | -7 | -8 | -8 | -8 |
| 4 | -5 | -6 | -7 | -8 | -8 | -8 | -8 | -8 | -8 |

Tab. 3 - manually calibrated rulebase

1. **Manual calibration and final results**

Fig. 8 - final fuzzy result for 25 deg tilt. AUC = 18,6; TTV = 1,4s

Manually calibrated rule base is presented in Tab.3. The result produced with it is presented on Fig.7.

As this result shows fuzzy regulator is superior in all respects to standard PID. It prevents arm from oscillating and takes less time to reach and maintain desired angle.

Behaviour for smaller angles (25 and 5 degrees) is presented on Fig.8 and Fig.9.

For comparison Fig.10 and Fig.11 present reference PID behavior for the same cases.

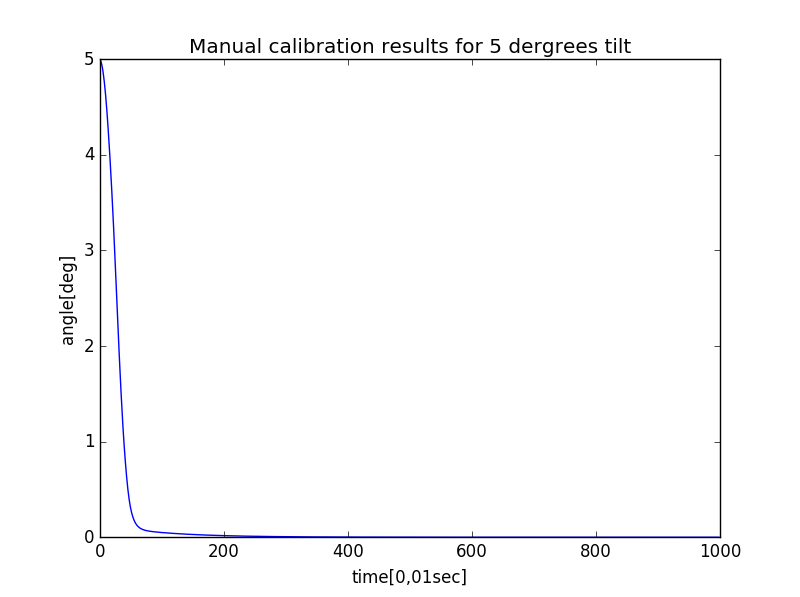
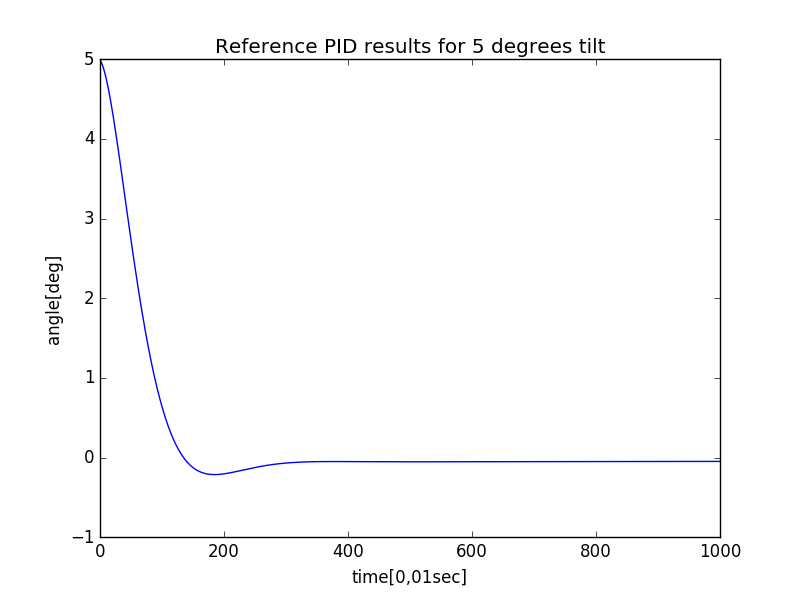
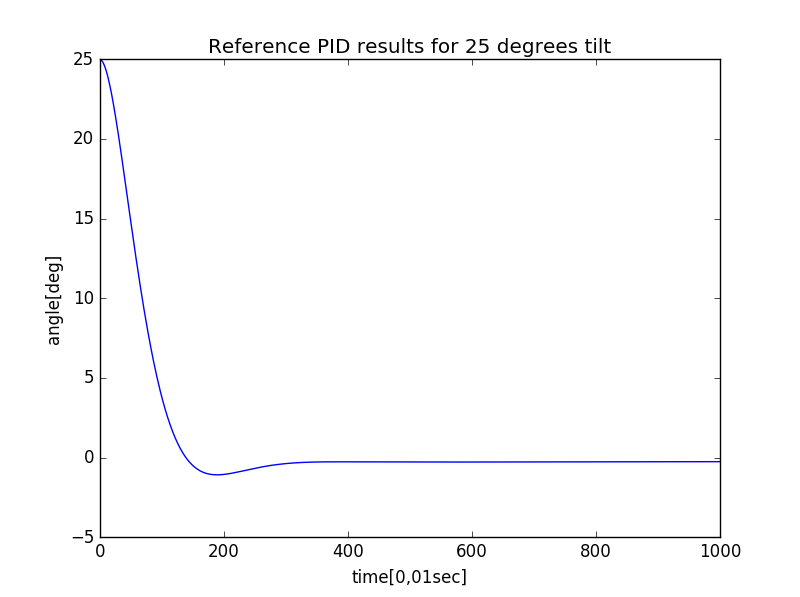


Fig. 9 - final fuzzy result for 5 deg tilt. AUC = 1,5; TTV = 0,5s

Fig. 10 - reference PID result for 25 deg tilt. AUC = 18,9; TTV = 2,7s

Better results of fuzzy algorithm can be easily explained with the fact that it generally uses rotors at higher thrusts than PID in order to provide more robust control over the plant. In other words: it accelerates and breaks witch much higher forces than PID can control.

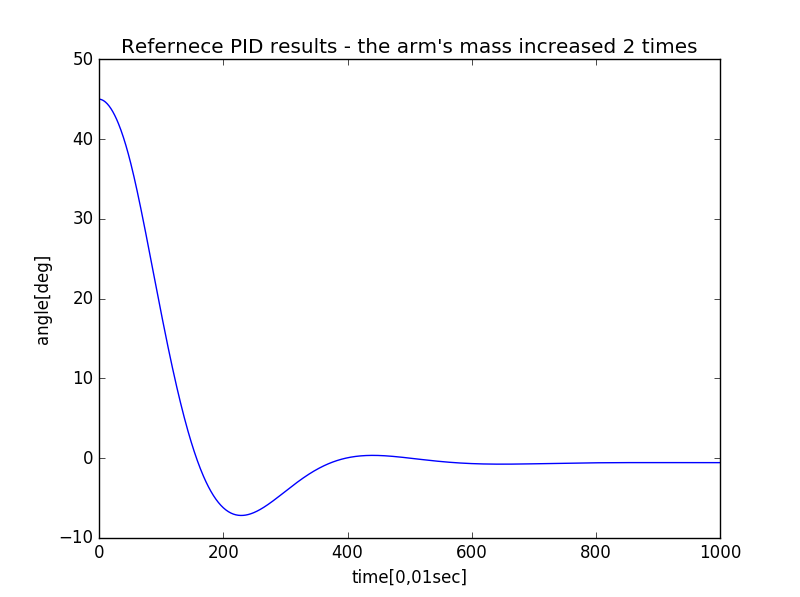
Additionally Fuzzy algorithm handles changes in plant’s inertia better than PID which is presented on Fig.12 and Fig.13.

Fig. 13- heavier arm regulated by fuzzy logic. AUC =38,0; TTV = 2,8s

Fig. 12- heavier arm regulated by PID. AUC = 53,2; TTV = 3,7s

1. **Conclusions**

Conducted simulations clearly show that fuzzy logic should handle the task of flight stabilization better than PID. Developed algorithm will be reimplemented in Java and used in UFP drones.

Fig. 11- reference PID result for 5 deg tilt. AUC = 3,6; TTV = 0,9s

**References:**

1. **Andrew Zulu, Samuel John** (2014) A Review of Control Algorithms for Autonomous Quadrotors. *Open Journal of Applied Sciences 2014, 4, 547-556*
2. Anaconda toolchain overview [https://www.continuum.io/anaconda-overview]
3. PyCharm official site [https://www.jetbrains.com/pycharm]
4. Matplotlib package official site [http://matplotlib.org]
5. Improving the beginner’s PID [http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction]
6. **Kevin M. Passino, Stephen Yurkovich**, The Ohio State University (1998) *Fuzzy Control. Addison Wesley Longman Inc.*
7. Github repository with implemented regulators [https://github.com/jmnich/UFP\_Regulator\_Simulations]