Embedded Helicopter Heading Control using an Adaptive Network-Based Fuzzy Inference System

Benjamin N. Passow **Simon Coupland** **Mario Gongora**
Centre for Computational Intelligence
De Montfort University
Leicester, LE1 9BH
UK
cir06np@dmu.ac.uk *simonc@dmu.ac.uk* *mgongora@dmu.ac.uk*

Abstract

This work introduces an embedded small indoor helicopter heading controller based on fuzzy logic. The work addresses the problem of system identification when implementing a Takagi-Sugeno-Kang type fuzzy logic controller. Instead of identifying the system formally beforehand, the fuzzy controllers consequent parameters are learned using a Neuro-Fuzzy Inference System with data collected from an existing, previously implemented proportional controller. The controllers are implemented on an embedded microcontroller driven system attached to the helicopter, also presented in this paper. Finally, the fuzzy logic based controller is tested and compared to the proportional controllers performance.

1 Introduction

An autonomous swarm of small indoor helicopters is currently being developed at the Centre for Computational Intelligence at the De Montfort University, using high-efficiency and compact embedded control and monitoring. Before an autonomous helicopter can do complex manoeuvres such as fly to defined way points, take-off and land or avoid obstacles, it has to be able to maintain a stable position. Stability and robustness from external influences such as wind as well as sensor and actuator error is needed for all three rotational and all three translational degrees of freedom (DOF). Fuzzy logic is advocated as a methodology that is able to efficiently deal with uncertainty and disturbance to the controller [7]. The advantage of the fuzzy approach is that a crisp mathematical model of the system need never be identified. As a first step in this larger project, an embedded fuzzy logic based heading control system is developed and introduced in this work.

A helicopter is a non-linear and unstable system which is difficult to control. There is much research being done in this area using a wide variety of methods including a number of artificial intelligence techniques in addition to the more traditional PID approach. In [9] Sanchez *et al* introduce an unmanned helicopter control system combining a Mamdani type fuzzy logic controller [8] with conventional proportional-integral-derivative (PID) controllers. The Fuzzy Inference System (FIS) is controlling the translational movement while the PID controllers handle the altitude and attitude of the helicopter. The system is tested via simulation on hovering and slow velocities and showed good performance. Sanchez *et al* state that the controller will be implemented in real time in the future and might be designed as a Takagi-Sugeno-Kang (TSK) fuzzy system [12]. TSK type fuzzy systems utilise consequent parameters that usually need to be identified beforehand the implementation of such a system. Other work [6] introduced a TSK type fuzzy system to control the altitude and attitude of a small size unmanned helicopter. Kadmiry and Driankov state that the large class of non-linear plants can well be represented by the TSK models with only minor changes. In their work, the non-linear helicopter multiple-input-multiple-output model was obtained from an existing real platform before any work on the FIS was done. As part of a comprehensive study [11], Shin *et al* introduced TSK based fuzzy logic controllers for the control actuators collective pitch, tail rotor pitch, longitudinal and lateral cyclic pitch. A genetic algorithm is used to identify and tune the consequent parameters of the four controllers using fitness evaluated from a simulation. In conclusion, the authors state that the fuzzy controllers are capable of handling uncertainties and disturbances.

The second paper described above [6] uses a mathematical model obtained from an existing platform to model the TSK fuzzy system and consequent parameters. The third paper [11] used a genetic algorithm to identify and tune these parameters. In this work, the TSK fuzzy inference system is implemented on a low-specification embedded hardware board. The FIS is learned from data taken from an existing system using an Adaptive Network-Based

Fuzzy Inference System (ANFIS). This method provides fast and effective means to develop TSK based fuzzy systems close to the original model without the need to formally identify the controllers model. The remainder of this paper is structured as follows. In section 2 the background of this work is discussed, including details of the helicopter model employed, some required flight dynamics and the hardware on which the controllers are running on. Section 3 gives an overview of the complete software and hardware setup used to implement and test the controllers. Section 4 describes the implementation process with details on the methodologies used. Finally, section 5 shows the results obtained in this work and section 6 presents the conclusions.

2 Background

In order to implement a fuzzy controller capable of controlling a helicopters heading, much additional hardware, tools and theoretical knowledge needs to be considered and used.

2.1 Helicopter model - Twister Bell 47

The helicopter used in this work is a *Twister Bell 47* with twin counter rotating rotors with 340 mm span, driven by two high performance DC motors, two servos to control rotor blade angles (not used in this work) and a receiver to control the motors and servos over radio control (also not used in this work). The weight of the helicopter without microcontroller, driver, sensor and batteries is approximately 200 grams.

A helicopters' rotor momentum combined with friction results in a reaction by the helicopters body to turn the opposite direction of the rotors. This effect is known as the torque effect. Usually single rotor helicopters have either some kind of tail rotor or a NOTAR (NOt a TAil Rotor) system to counteract this effect and to change heading. This helicopter doesn't have a tail rotor of any kind. Rather, such a system comprises of two counter rotating blades is used to give both lift and direction. When both rotors are moving at the same speed a constant heading is maintained. If either rotor's speed is reduced the heading will change and lift will be reduced. The change in heading results from the differing levels of torque effect being produced by each rotor. If one rotor's speed is reduced, whilst the others speed is increased by an identical amount, the heading will change whilst a constant amount of lift will be maintained. For a more stable and safe test scenario, the top rotor blades have been turned around. Thus the rotors are working against each other, almost completely neutralising the lift on the helicopter. Further more, the helicopter has been fixed to a plate on top of a ball bearing, thereby enabling the helicopter to turn freely while not being able to lift off nor tilting to any side.

As figure 1 shows, usually helicopters have 3 rotational degrees of freedom (DOF) called pitch, roll and yaw, as well as 3 translational DOF called up / down, left / right and forwards / backwards. All of these outputs are controlled by 4 inputs, the amount of lift with the speed of the rotor (eventually with an adjustable rotor angle), the heading with the tail rotor or the differential of two rotors, and the pitch and roll rotational angles by adjusting the rotors angle that is depending on the rotors position. For more information see the comprehensive study at [11].

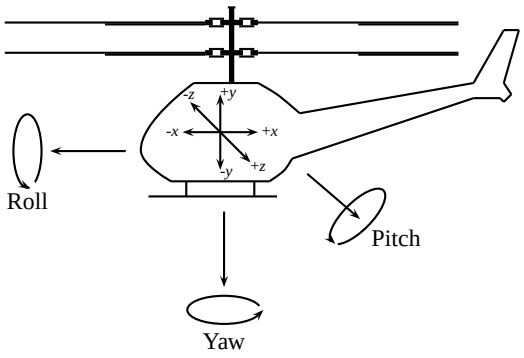

Figure 1: A dual rotor helicopters degrees of freedom

Taken together one can say that a helicopter is an unstable non-linear system that is very sensitive to external disturbances [3] and therefore very difficult to control.

2.2 Microcontroller - OOPIC-R

The heading controller introduced in this work is implemented on a OOPIC-R version B.2.2+ microcontroller. This embedded system has a low power consumption while providing many inputs and outputs great for driving models and robots. Although the microcontroller has a rather low performance and only 168 Byte of program accessible RAM, it can be programmed in C++, Java and Visual Basic like languages using a object oriented paradigm. Unfortunately, the microprocessor and compiler are not able to work with floating point arithmetic. Clearly this affects the development of a fuzzy inference system on such a device.

The microcontroller also provides regulated power supply to the devices connected to it. Except for the external sensor as well as the driver attached to it, the microcontroller board has all the technology required to drive the controller for this work.

2.3 Sensor - Honeywell HMR3000

The sensor used to read the current heading of the helicopter is the Honeywell HMR3000 Digital Compass Module. This device is highly accurate up to 0.5°, has a fast response time of up to 20 Hz and a low power consumption smaller 35 mA. It is communicating over the RS232 using a NMEA (National Marine Electronics Association) protocol.

The device needs to be initialised after startup with certain NMEA strings sent over the serial port in order to configure and start the device. Depending on these settings, the device returns different readings in different time intervals using different message formats. In section 4 more details are given on how the sensor is configured and read for this particular task.

2.4 Adaptive Network-Based Fuzzy Inference Systems (ANFIS)

ANFIS [4] is a neuro-fuzzy approach which brings together the machine learning abilities of artificial neural networks with the tractability and robustness of fuzzy logic. ANFIS models a rule based fuzzy inference system as feed forward network, the rules can be of Mamdani, TSK or standard additive model type. The purpose of the ANFIS model is to allow the fuzzy sets to be tuned using data the has already been gather about a given system. The fuzzy rules, or whichever type, have an associated set of parameters for the antecedent and consequent part of the rules. ANFIS uses a two pass hybrid learning algorithm to tune these parameters, so that the fuzzy inference system fits the data. For the forward pass of the learning algorithm the antecedent parameters are fixed and the consequent parameters are estimated using the least squares estimator. Data is fed through the system and the least squares regression technique finds the best fit for this data, giving the consequent parameters. For the backwards pass of the learning algorithm the consequent parameters are now fixed and the antecedent parameters are tuned. The errors between the output from the system and the actual data points are propagated back through the system and gradient decent is used to minimise this error by tuning the antecedent parameters. In this paper we use ANFIS to model a where the performance is given by a proportional controller.

3 System Setup

The complete controller system persists of components as described previously. Figure 2 shows all of these components connected together as required.

The battery powering the microcontroller and motors is a 7.4 V Li-ion Polymer with 800mAh. The microcontroller board is supplying all electrical devices except for the motors with regulated 5 V. Three buttons on the microcontroller enable user interaction with the controller application. The first button is set to save the current heading as the fixed heading the controller will use to hold or move back to. The second and third button enable a user to increase and decrease respectively the total power to the rotors.

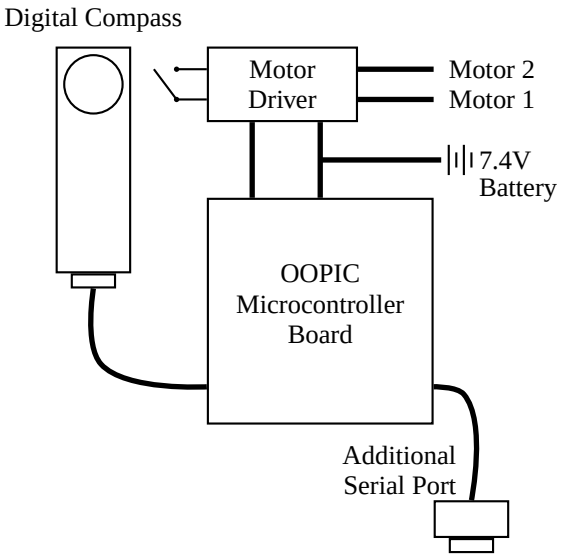

Figure 2: Complete system setup with sensor, driver, battery and serial debug interface

The HMR3000 sensor is connected to the microcontroller over the serial port using a special cable that also provides the sensor with power. The driver is connected to the OOPICs digital motor I/Os as well as to its 5 V power supply. An additional safety switch is connected to the drivers enable input to quickly disable the driver, if needed. Two cables from the driver connect to the motors and supply these with up to full power from the battery, depending on the controllers PWM signal sent to the driver. An additional and optional TTL logic to serial converter chip is attached to the digital I/Os of the OOPIC to provide a debug interface as the primary serial port is already used by the sensor. The complete system is then attached to the helicopter.

4 Implementation

Using the complete system as previously described, the controllers can now be implemented. This section gives details on how exactly this is done.

4.1 Overview

The first step in the development of the fuzzy logic based heading controller is the implementation of a simple proportional controller. This controller will serve as a raw model to build the TSK fuzzy system from. Figure 3 shows the three implementation steps needed to get to a good working FIS. The following subsections explain the implementation details for the proportional controller as well as the design, training and implementation of the FIS. The most important input to the controller is the current heading from the sensor.

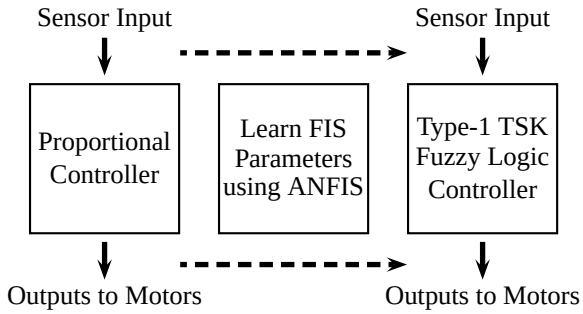

Figure 3: Fuzzy Inference System Developmental Methodology Overview.

4.2 Reading the Sensor

The HMR3000 uses the NMEA (National Marine Electronics Association) protocol to send and receive information. The sensor needs to be initialised and configured in order to send sensor data in the format and refresh speed required. Thus, the sensor is configured to transmit new readings using ASCII messages in degrees with 13.75 Hz. A final command initiates the sensor which starts sending information over the serial port, always overwriting old information on the port of the microcontroller. The following algorithm is used to successfully read the current heading from the sensor data.

As the information is sent single char wise, one does not know exactly at what point the information is read and analysed. A heading of 310.1 degrees could be read as 0.1 degrees as the first two chars were sent already and not analysed by the application. Therefore synchronisation is required and achieved by reading all information until no more information is sent by the sensor. This sending pause is only happening between sensor readings. At that point the program is synchronised and the sensor reading is taken with the following procedure:

1. Wait for new reading.
2. Read byte by byte into a character array.
3. Generate number out of read information.

4.3 Proportional Controller

With the heading read from the sensor, the application has now the first and most important of the two inputs needed by the controller. The second input is entered by the user by means of the buttons on the microcontroller: The total power to drive the rotors.

In order to make the heading meaningful to the controller, the difference between the previously fixed and the current heading is calculated by the following equation.

$$error = current\_heading - fixed\_heading \quad (1)$$

Now the error is in between -360 and 360 degrees and therefore has to be pushed back into a valid range using some conditional statements. It should be mentioned here that the error is computed for negative and positive errors using a boolean flag which is needed due to a bug within the OOPIC compiler. Negative numbers are not correctly analysed in conditional statements without this flag.

The proportional controller reacts based on the following two rules:

- The bigger the error, the bigger the turning speed difference
- The bigger the total power, the smaller the turning speed difference

Although this is not exactly implemented in a proportional way, the controller is closely related to a proportional controllers and therefore called “proportional controller” in this work.

4.4 Fuzzy Inference System

As mentioned previously, a Takagi-Sugeno-Kang fuzzy inference system has been chosen for this work. Such a fuzzy system is usually much less computational expensive than a Mamdani type FIS [1]. It also needs less variable space in terms of RAM which is also extremely small on the microcontroller used in this work. Although a Mamdani type FIS might be implementable on this embedded system, the TSK fuzzy system is the better choice in terms of performance as well as future work [5, 6].

The rules bases for the controller contained six rules:

$$R_1 : if\ \mu_{low}(pl)\ and\ \mu_{low}(e)\ then\ w_1$$
$$R_2 : if\ \mu_{low}(pl)\ and\ \mu_{medium}(e)\ then\ w_2$$
$$R_3 : if\ \mu_{low}(pl)\ and\ \mu_{high}(e)\ then\ w_3$$
$$R_4 : if\ \mu_{high}(pl)\ and\ \mu_{low}(e)\ then\ w_4$$
$$R_5 : if\ \mu_{high}(pl)\ and\ \mu_{medium}(e)\ then\ w_5$$
$$R_6 : if\ \mu_{high}(pl)\ and\ \mu_{high}(e)\ then\ w_6$$

where R_i is the i^{th} rule, *pl* is power level, *e* is error in the current heading and $f_1 \ldots f_6$ are fuzzy parameters that are to be learned using the ANFIS hybrid learning algorithm. The power level input has two associated fuzzy set *low* and *high* and the error input has three associated fuzzy set *low*, *medium* and *high*. The ANFIS topology for this fuzzy system is depicted in Figure 4. Figures 5 and 6 depict

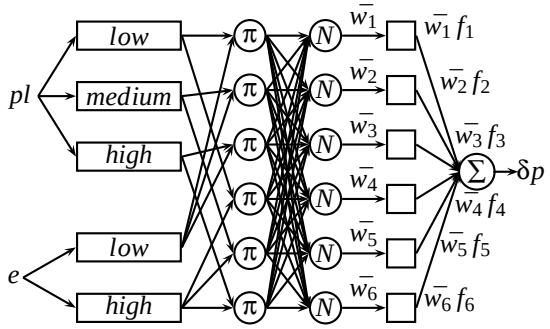

Figure 4: The ANFIS Topology

the membership functions for the power level and error inputs respectively, which were trained using data from the proportional controller using the ANFIS hybrid learning algorithm. The firing strength w_i of each rule is calculated with the minimum t-norm, these are also ($\bar{w}_i$) normalised prior to centroid calculation. The final output of the FIS, change in power ratio, is calculated using equation 2.

$$\delta p = \sum_{i=1}^{6} \bar{w}_i f_i \qquad (2)$$

where $\bar{w}_i$ is the firing strength of the i_{th} rule and f_i is a consequent parameter tuned using ANFIS.

4.4.1 Training parameters with ANFIS

ANFIS uses a hybrid learning approach, the back-propagation gradient descent method to fine-tune the membership functions and the least-squares estimator to identify the consequent parameters. Using this system, the "constant" consequent parameters in table 1 were identified.

Based on these parameters the FIS showed to be performing well as can be seen in figure 7. The Figure show that the choice of membership functions for the power input is resulting in a plateau on the rule surface. This plateau represents an area of control that is of no practical use.

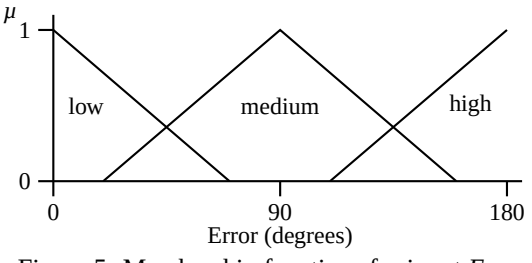

Figure 5: Membership functions for input *Error*.

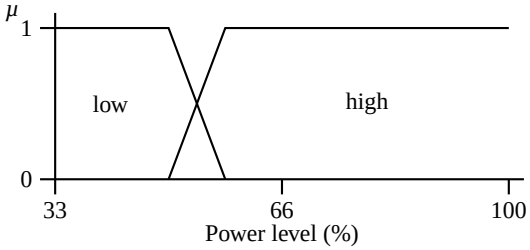

Figure 6: Membership functions for inputs *Power-Level*.

Table 1: Learned Parameters of the Fuzzy Rules.

Parameter	Value
f_1	0.2744
f_2	0.3435
f_3	1.7486
f_4	1.0858
f_5	2.0525
f_6	0.9821

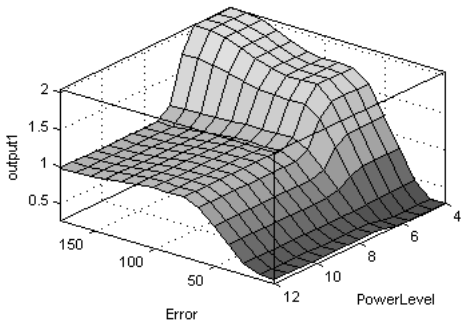

Figure 7: Rule output surface of trained FIS

4.4.2 Fixed Point Arithmetic

The methods presented in this Section are used when fuzzifing the antecedent membership functions in the controllers inference engine. The microcontroller is not capable of computing floating point numbers, usually needed for the implementation of a FIS. One solution to this problem could be to recreate the FIS to use higher numbers for inputs and output, for example choosing a membership grade range between 0 and 100. But this would mean that the *intuitive* and *clear* design of the FIS would be tweaked and sized in many ways. Rather than changing the design, another method can be used to compute the numbers needed: *Fixed Point Arithmetic*

Fixed Point Arithmetic has the performance of integer calculations while providing adequate accuracy [10]. Unlike floating point numbers, fixed point numbers have a certain range of bits only for whole numbers and other bits for the rest. As the name of the variable type already states, the *point* within the variable is fixed. Due to this fact, the precision of these numbers is only a fraction of that of floating point variables. This needs to be considered during the application design and implementation. The point position is very important to take into account when doing calculations with these numbers. Fixed point variables are created by shifting the value by the fixed point offset to the left. The following example will shows how this is done:

fix1 = 3.1415 << fixed_point_offset;

Using a 2 byte or 16 bit variable and a fixed point offset of 8 bit, the shift operation would be the same as a multiplication by $2^8 - 1 = 255$. Thus, a number like 3.1415 would become 801.083 and be stored in the fixed point variable as 801 ($\sim 3.1412 * 255$). For higher precision, more bits could be allocated from the whole number to the rest. Unfortunately, calculations like multiplication and division need many extra bits within the variables as will be shown later. In order to reverse the process of creating fixed point numbers, the variable needs to be shifted back by the fixed point offset. Clearly, in this process the number after the point is simply cut off. Multiplications and divisions need to be performed together with the shift operation. The following example could be used:

*fix3 = (fix1 * fix2) >> fixed_point_offset;*

fix3 = (fix1 << fixed_point_offset) / fix2;

Using previous fixed point number $fix1$ again, the following calculations will serve as an example what is happening here. When multiplying two fixed point numbers like $fix2 = fix1 * fix1$, the point location within the result is shifted by another fixed point offset. In the previous example this means that $801 * 801 = 641601$ which is a far too large number for a 16 bit variable. Therefore, before a multiplication is done one of the fixed point variables needs to be shifted back by the fixed point offset, loosing all precision. Alternatively, both variables can be shifted back half the fixed point offset loosing only some precision. Taken together, every use and calculation with a fixed point variable needs to be thought through carefully always having in mind the range of possible values in the variables. If the system is designed carefully, this method provides adequate accuracy, is very fast and working well. This method provides the means needed to very efficiently compute membership grades, rules and outputs on the embedded system.

4.4.3 Implementation on the OOPIC

Using the system, methods and results described above, the TSK type fuzzy inference system can be realised. In order to calculate the membership grade of each membership function, the range of the input is checked first. If the input is in the range of a membership function, the membership grade is computed by using geometric methods [2]. The membership grades of the these geometric fuzzy sets are given in equations 3 to 5.

$$\mu_{low}(x) = \begin{cases} 255 - (f/70),\ 0 < x < 70 \\ 0,\ x < 0,\ 70 < x \end{cases} \tag{3}$$

$$\mu_{medium}(x) = \begin{cases} (f - 5120)/70,\ 20 < x < 90 \\ 255 - (f - 23040)/160, \\ \quad 90 < x < 160 \\ 0,\ x < 20,\ 160 < x \end{cases} \tag{4}$$

$$\mu_{high}(x) = \begin{cases} (f - 28160)/180,\ 110 < x < 180 \\ 0,\ x < 110,\ 180 < x \end{cases} \tag{5}$$

In equations 3 to 5 f is the $fixDifHead$ variable using for the fixed point arithmetic.

Together with the calculation of the membership grade, a rule specific boolean flag is set to enable easier and faster rule identification. The second input is rather crisp and conditional statements are used to compute the membership grade and rule identification. The final output from the system is calculated using equation 2. The final output is rounded using the same fixed point arithmetic and converted back to an integer value. This output is then used to increase and decrease the rotor speeds respectively to the turn direction.

5 Results

The fuzzy logic based controller has been tested on the embedded system. Figure 8 shows the helicopter with the embedded system attached to it on a turntable. The system was tested using all available speed settings and many different angles.

The Takagi-Sugeno-Kang type fuzzy controller has been shown to be working well under a variety of conditions. The small changes from the previously fixed heading were corrected instantly by slowly rotating the helicopter back to the fixed heading. Medium changes in heading were corrected with more yaw rotation, depending on the total power on the rotors. Large errors between the helicopters and the fixed heading were successfully corrected by the controller although some jiggering appeared. To

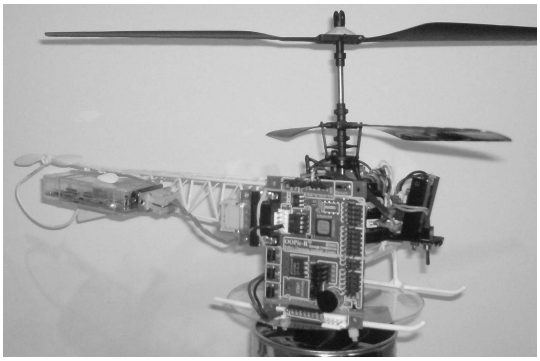

Figure 8: Complete helicopter model with embedded system attached.

give a clearer picture of the controller performance we recorded the response of each controller after manually altering the helicopters heading by 90^o. Figure 9 depicts the response graph of the PID controller a moderate change in desired heading of 90^o. Figure 10 depicts the response graph of the fuzzy controller a moderate change in desired heading of 90^o. Both systems function adequately, however the PID system appears to outperform the fuzzy system in terms of response time and overshoot of the set point. These two factors are obviously related, with the set point overshoot mainly being caused by the slower response time. In order to measure both controllers cycle times a short "." message is sent over the debug serial interface after every control cycle. With this method, the control cycle times were measured. The results are presented in table 2. Despite the significant hardware performance limitations, we were able to successfully create a real-time FLC for a highly unstable system.

The results show that the TSK type fuzzy system used here is only 25% slower than the proportional one. However, performance of both controllers could be significantly improved if we are

Table 2: Controller Cycle Times

Controller Cycle Times	
Proportional Controller	2.4 cycles / second
Fuzzy Logic Controller	1.8 cycles / second

able to process sensor inputs in a more timely fashion. The bottleneck for these results is the OOPIC microcontrollers low performance.

6 Conclusion

In this work, a Takagi-Sugeno-Kang type fuzzy inference system is proposed for the use on an low-specification embedded hardware system controlling the heading of a small indoor helicopter. We were able to successfully create a real-time FLC for a highly unstable system with these significant hardware constraints. The system was developed and trained based on a proportional controller using an Adaptive Network-Based Fuzzy Inference System. The microcontroller employed is a OOPIC-R with a HMR3000 digital compass and a L298N driver to drive the engines of the Twister Bell 47 helicopter model.

In conclusion, it has been shown that the fuzzy logic based controller is performing very well. The speed difference between the proportional and the fuzzy logic based controllers is only about 25%. The error in heading is successfully corrected with only minor problems. The controller has problems with jiggering and in a certain situation when opposite to the previously fixed heading. Both problems were discussed with possible solutions such as adding a heading history to the controllers inputs and exchanging the microcontroller for a better model. Unfortunately, the microcontroller used in this work is rather slow and does not have much memory available. It is clearly the bottleneck of this work and

Figure 9: Response Graph for the PID controller with a Heading Perturbed by 90^o.

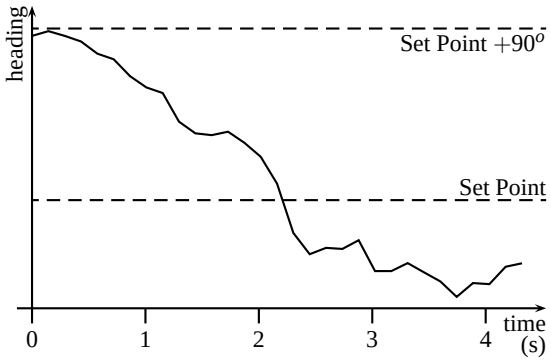

Figure 10: Response Graph for the Fuzzy controller with a Heading Perturbed by 90^o.

for further work another microprocessor would most definitely be used. Still, it has been shown that even on a controller with only 168 Byte of program accessible RAM a fuzzy inference system is feasible. Fuzzy logic based controllers are more robust for unstable systems with high variability and uncertainty such as a helicopter. The TSK type fuzzy logic system has been implemented with parameters identified using ANFIS. This method made the development of the fuzzy system easier as no formal controller model had to be identified. The results of tests showed that the fuzzy based controller is very similar to the proportional prototype. However, the fuzzy logic based controller is more robust to disturbances and unforeseen inputs. Additionally, it has been shown that using fixed point arithmetic one can implement almost any kind of fuzzy system on an embedded device without the need to modify or tweak the fuzzy system beforehand.

Taken together, this work presented a fuzzy logic based heading controller for helicopters running on an embedded system. Clearly, the Takagi-Sugeno-Kang type fuzzy logic approach can be recommended for such tasks. In combination with an Adaptive Network-Based Fuzzy Inference System, the TSK type fuzzy system can be implemented based on an existing model without the need to formally identify a control model. Still, an analysis of the system and methods employed showed that there is scope for improvements and future extensions.

References

[1] P. Bergsten. *Observers and controllers for Takagi-Sugeno systems*. PhD thesis, Oerebro University, Oerobro, Sweden, 2001.

[2] S. Coupland and R. John. Geometric Type-1 and Type-2 Fuzzy Logic Systems. *IEEE Transactions on Fuzzy Systems*, 15(1):3–15, Feburary 2007.

[3] I.Rojas, H. Pomares, C. Puntonet, F.Rojas, M.Rodriguez, and O. Valenzuela. On-line adaptive fuzzy controller: Application to helicopter stabilization of the altitude of a helicopter. In *Proc. Of the International Symposium on Computational Intelligence for Measurement Systems and Applications*, Lugano, Switzerland, 29-31 July 2003.

[4] J.-S. R. Jang. Anfis: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23:665–684, 1993.

[5] B. Kadmiry and D. Driankov. Fuzzy control for an autonomous helicopter. In *Proc. of the 9th IEEE Int. Fuzzy Systems Association (IFSA/NAFIPS) World Congress*, volume 5, pages 2797–2802, Vancouver - Canada, July 2001.

[6] B. Kadmiry and D. Driankov. A fuzzy gain-scheduler for the attitude control of an unmanned helicopter. *IEEE Transactions on Fuzzy Systems*, 12:502–515, 2004.

[7] B. Kosko. *Fuzzy Engineering*. Prentice Hall, New Jersey, 1997.

[8] E. H. Mamdani. Application of Fuzzy Algorithms for Simple Dynamic Plant. *Proc. IEE*, 121:1585 – 1588, 1974.

[9] E. Sanchez, H. Becerra, and C. Velez. Combining fuzzy and pid control for an unmanned helicopter. In *Annual Meeting of the North American Fuzzy Information Processing Society*, pages 235– 240, Unidad Guadalajara, Mexico, 2005.

[10] R. Schulz and A. Lecher. *Algorithmen für Spieleprogrammierung*. TLC The Learning Companie, 1995.

[11] H. Shim, T. Koo, F. Hoffmann, and S. Sastry. A comprehensive study of control design for an autonomous helicopter. In *37th IEEE conference on Dicision Control*, Tampa, FL, 1998.

[12] M. Sugeno. An introductory survey of fuzzy control. *Information Sciences*, 36:59–83, 1985.