Robust Fuzzy Controllers Using FPGAs

Author Gene S. Monroe, Jr.

NASA LaRC

Gene.S.Monroe@NASA.gov

Abstract

Electro-mechanical device controllers typically come in one of three forms, proportional (P), Proportional Derivative (PD), and Proportional Integral Derivative (PID). Two methods of control are discussed in this paper; they are (1) the classical technique that requires an in-depth mathematical use of poles and zeros, and (2) the fuzzy logic (FL) technique that is similar to the way humans think and make decisions [1]. FL controllers are used in multiple industries; examples include control engineering, computer vision, pattern recognition, statistics, and data analysis [2]. Presented is a study on the development of a PD motor controller written in very high speed hardware description language (VHDL), and implemented in FL.

Four distinct abstractions compose the FL controller, they are the fuzzifier, the rule-base, the fuzzy inference system (FIS), and the defuzzifier [3]. FL is similar to, but different from, Boolean logic; where the output value may be equal to 0 or 1, but it could also be equal to any decimal value between them. This controller is unique because of its VHDL implementation, which uses integer mathematics. To compensate for VHDL's inability to synthesis floating point numbers, a scale factor equal to $10^{(N/4)}$ is utilized; where N is equal to data word size. The scaling factor shifts the decimal digits to the left of the decimal point for increased precision.

PD controllers are ideal for use with servo motors, where position control is effective. This paper discusses control methods for motion-base platforms where a constant velocity equivalent to a spectral resolution of $0.25cm^{-1}$ is required [4]; however, the control capability of this controller extends to various other platforms.

1. Introduction

The PD form of a controller is an ideal method for controlling servo motors with respect to motion base platforms, because it is a position algorithm based on the current position and the rate at which that position is changing [5]. The focus of this research is on the design and development of a FL PD servo motor controller implemented in VHDL. A basic diagram of a classical control system with a feedback loop is shown in Figure 1.

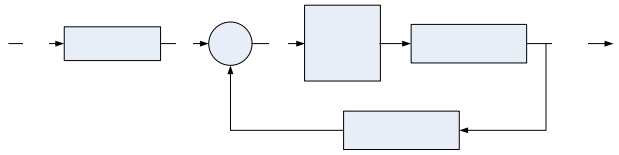

Figure 1. Diagram of a typical control circuit.

FL works much like a human brain; an example is monitoring the water temperature while preparing a bath. If the water is too cold then increase the hot water flow, if it is just right then do nothing and let the water continue to fill, but if it is too hot then increase the cold water flow. In this example there are three temperature conditions used as criteria; too cold, just right, and too hot. These temperature conditions are called fuzzy variables and compose a fuzzy set. There is no exact measure of heat for the perfect bath, because that temperature is at differing degrees for different people, which implies that the final assessment will be made by the individual taking the bath.

FL is about relative truth and the importance of being exact or approximate [6], it refers to the degree of truth, not the likelihood of truth. The "degree of truth" refers the degree of membership (DOM), Equation 1 demonstrates how the fuzzy set *opinion* could be used to calculate this value. *Opinion* could be a piece-wise linear function, an elliptical function, or some other descriptive analysis used to determine an approximate value.

$$DOM = f(opinion) \quad (1)$$

The value placed on the temperature is a fuzzy one, so there must be a numerical value associated with it if a computer is to make human-like decisions. One might

say that a given water temperature is a little too cool, but overall it is almost OK. Mathematically that may translate to 35% too cold, and 65% just right. A human inputs the numeric range values that the fuzzy variables will be equated to. An expert may derive these values using their intuitive assessment based on their experience or what seem to be logical [7].

The range limits of these fuzzy variables can be viewed as linguistic in the design stages of the system; the variables used for this example are defined as: (too cold, just_right$_{min}$], [just_right$_{min}$, just_right$_{max}$], and [just_right$_{max,}$ too hot). See Figure 2 for a visual depiction of the fuzzy set, and note that this is the precept to the formulation of membership functions (MFs). MFs are the functions already discussed for the right-hand side of Equation 1, and the DOM is the derived confidence that the temperature of the water is too cold or just right or too hot.

Figure 2. Visual depiction of the fuzzy set.

Confidence levels are physical measures from where the input value on the independent axis measures perpendicularly to intersect the MF, and from there measures horizontally to the dependant axis, which will be some value between 0 and 1. MFs are line equations and each fuzzy range has its own MF. MFs can be shaped as a straight line, an elliptical form, a bell shape, etc. They are used to define the degrees of membership (DOM) for input and output values. MATLAB was used to depict how this ties together, and is shown in Figure 3. This combination of fuzzy ranges, MFs and confidences is replicated for each input and each output. Later during a discussion on *fuzzy rules* the interpretation of this chart would read that the computer is 35% confident that the water is too cold, because the analysis can only be as sure as the least confidence.

After all the inputs have been fuzzified, the computer must use a fuzzy inference system (FIS) to combine these values to derive the inputs to the output's fuzzy set. This is done through a set of rules, composed of IF… THEN… statements. The IF's conditional statements for the rules compare the two inputs of velocity and acceleration and join them using the fuzzy 'AND', this is how a FL PD control design is created. The fuzzy 'AND' yields the minimum value from a comparison between the two inputs.

The final stage in FL defuzzifies the output variables from the FIS. This algorithm combines the results from the rules and multiplies each output variable by its associated weight. The weight is a value given to each fuzzy output variable of the fuzzy output set. It could be a standard non-biased value equal to the center of the variable's range, or some other value based on an expert's opinion. The sum of these variables composes the crisp output delivered to the external object being controlled. The method of defuzzification used in this research was the root-sum-square; five other candidate methods include: Centroid, Maximum, Mean of Maximum, Height, and Modified Height [8].

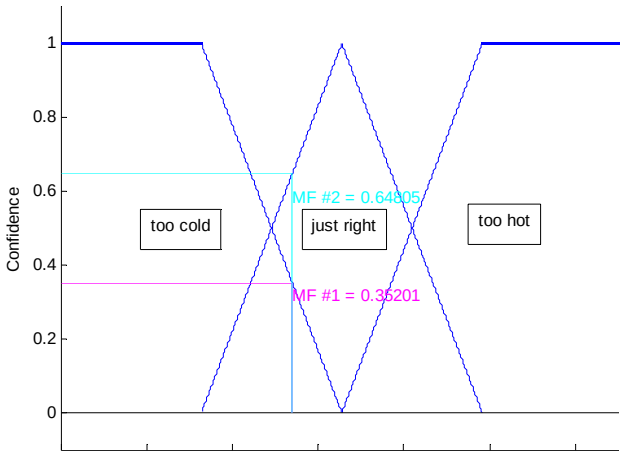

Figure 3. Visual diagram depicting the DOM.

Using fuzzy sets to establish the control parameters of a system offers significant advantages in non-linear cases provided that the rule base does not get too large [9]. Using these fuzzy sets maintains precise control over the controlled system as the hardware may degrade over time, or one or more of the inputs may vary. A significant advantage of this FL controller is that it has been coded in VHDL and programmed into a single field programmable gate array (FPGA), because this reduces the number of electronic components used to implement the controller, it enables redundancy by having multiple copies/images of the code, and yields robustness as a controller that has multiple systems capability.

The final product of this research is a FL controller that has been developed using VHDL to program a Xilinx FPGA, for use in motion base systems or other types of systems that require precise signal response. The controller is capable of overseeing multiple types of control signals; an infant software version of this firmware system was tested and successfully implemented in the robotics lab of the Electronics System Branch at the NASA Langley Research Center (LaRC). This VHDL-coded FL controller is planned to be utilized in the spring of '2007 on-board an aerodynamic structure mounted in a NASA LaRC wind tunnel. The controller will be part of project that will programmatically maneuver ailerons and other wing-tip objects during the tests without manual manipulations, using a multiple input multiple output (MIMO) general predictive algorithm.

2. Technology Review

FL controllers work well with both linear and non-linear systems. FL is rapidly spreading throughout the technology world in the areas of aircraft/spacecraft, automated highway systems, automobiles, autonomous vehicles, manufacturing systems, power industry, process control, and robotics [10]. PD control is also referred to as a "lead network," a term relevant to classical controllers. Evidence will be provided to conclude that classical PD controllers may not meet design specifications for non-linear systems. The Mathworks' Simulink and MATLAB software packages are used to demonstrate through simulation that FL controllers are easier to design and are more robust than classical controllers.

CLASSICAL DESIGN METHOD

The following example uses a hypothetical plant that is non-linear, using the open loop equation form from Equation 2 the open loop system is defined in Equation 3.

$$G_{OL}(s) = \frac{N}{S^2 + as + b} \tag{2}$$

$$G_{OL}(s) = \frac{7570}{(s + 62.61)(s - 62.61)} \tag{3}$$

Design engineers generate control parameters to satisfy system requirements, such as the ones provided in Table 1. A thorough understanding of mathematics is required for this technique, and it should be noted here that all plants and compensators are defined by polynomial equations, with zeros in the numerator and poles in the denominator.

Table 1. Design requirements.

REQ'T	DESCRIPTION
1	%OS < 50%
2	T_s < 1 second
3	e(∞) < 10%; e_{step}, e_{ramp}, or $e_{parabolic}$
4	GM > 8dB
5	PM > 45°

The plant described in Equation 2 reveals a pole in the right hand plane (RHP), and is an indication that the plant is non-linear. As a side this can be further verified by using the Routh-Hurwitz method [11] on the closed-loop version of the open loop transfer function. Equation 4 is the closed loop transfer function form.

$$G_{CL}(s) = \frac{G_{OL}(s)}{1 + G_{OL}(s)} \tag{4}$$

Using Equation 4, the closed loop transfer function is:

$$G_{CL}(s) = \frac{7570}{s^2 - 3{,}649.99} \tag{5}$$

Expanding the denominator polynomial is how the Routh-Hurwitz table is comprised, such as in Table 2. Routh-Hurwitz states when a zero is displayed in the middle column, that the system is non-linear.

Table 2. Routh-Hurwitz table.

S^2	1	3,649
S^1	0	0

CLASSICAL DESIGN PROCESS IN MATLAB

1. At the MATLAB command prompt, define "Gs = tf(7570,conv([1 62.61],[1 -62.61])"; open the root locus tool.
2. Right click on the graph and input the system characteristics derived in step one using equations.
3. Click on the step function and adjust the gain to see if the given requirements can be met.
4. Since the system must have more compensation, add a zero and a pole to make it a lead network, i.e., a PD controller.
5. It can be seen from step 4 that the transient responses have been met, see Figure 4. The next step is to calculate the Steady-State error.
6. Unfortunately the classical PD design does not meet the steady-state error requirement. Add a lag component by inserting another zero and another pole; the system now meets all design criteria, but as a PID Controller.
   a. With a steady-state error $e_{parabolic}(\alpha) = 7.9\%$
   b. Figure 5 shows that the GM and PM were also met, where the GM is nearly 50dB, and the PM is 64.4°.

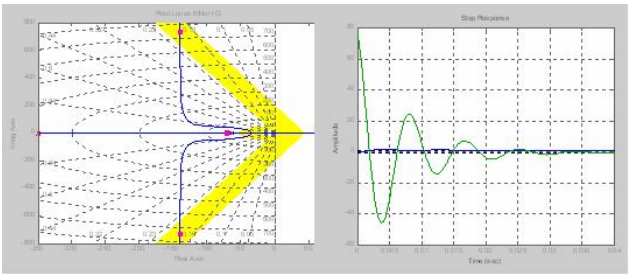

Figure 4. Root locus and step response from step 5.

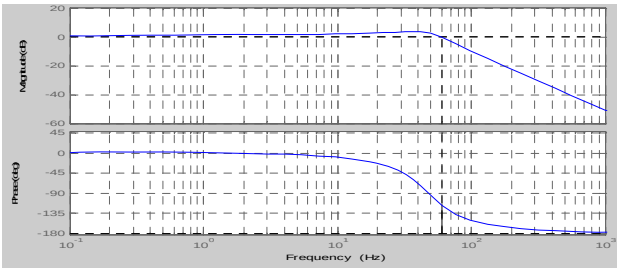

Figure 5. Verification using Bode plots.

FUZZY LOGIC DESIGN METHOD

The linguistic fuzzy variables 'Slow', 'OK', 'Fast', 'NEG', 'OK', and 'POS' comprise the two fuzzy sets; error and derror. To quantify the input confidence levels, use the line equation formula as the input to Equation 1 and refer to Figure 3 to better understand the fuzzification process. A Simulink model of the FL controller is shown in Figure 6, note the slight difference from the classical model shown in Figure 1 as there are two inputs to the FL compensator, "error" and "derror". Each of these two inputs has three fuzzy variables, as shown in Figure 7, and used to develop Linguistic Rules in Table 3. The products of these rules are then aligned to determine the polarity of the output.

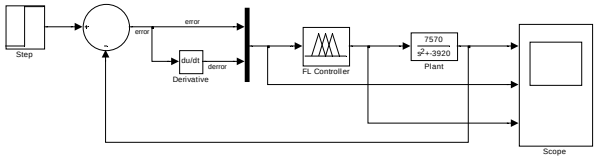

Figure 6. FL feedback control system.

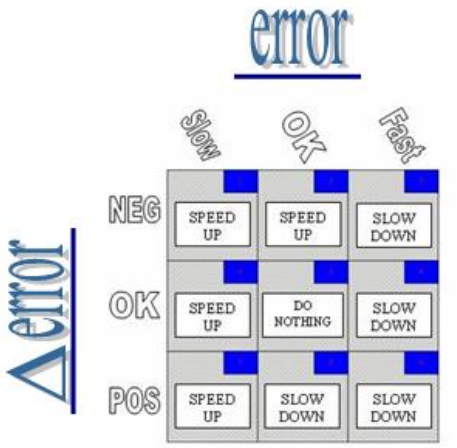

Figure 7. Matrix for Rule-Base.

The following statements demonstrate how to use each input's linguistic variables to derive the output fuzzy sets, which are Slow Down, Do Nothing, and Speed Up. The matrix in Figure 7 shows the two inputs with their respective fuzzy variables outside the matrix, and the output's fuzzy variables inside the matrix. The rules are numbered from left to right beginning with the top row, Rule #1 to Rule #9.

Rule 1. "error is Slow and derror is negative" says the platform is moving too slow and is slowing down. Output is a positive voltage.

Rule 2. "error is OK and Δerror is negative" says the platform is moving at an acceptable speed and is slowing down. The output is a positive voltage.

Rule 3. "error is Fast and Δerror is negative" says the platform is moving too fast and is slowing down. The output is a negative voltage.

Rule 4. "error is Slow and Δerror is OK" says the platform is going too slow and is not changing. The output is a positive voltage.

Rule 5. "error is OK and Δerror is OK" says the platform is moving at an acceptable speed and is not changing. The output is a zero voltage.

Rule 6. "error is Fast and Δerror is OK" says the platform is moving too fast and is not changing. The output is a negative voltage.

Rule 7. "error is Slow and Δerror is positive" says the platform is moving too slow and is speeding up. The output is a positive voltage.

Rule 8. "error is OK and Δerror is positive" says the platform is moving at an acceptable speed and is speeding up. The output is a negative voltage.

Rule 9. "error is Fast and Δerror is positive" says the platform is moving too fast and is speeding up. The output is a negative voltage.

Table 3. Fuzzy Rules.

Rule #	Fuzzy Equation
1	Slow AND NEG
2	OK AND NEG
3	Fast AND NEG
4	Slow AND OK
5	OK AND OK
6	Fast AND OK
7	Slow AND POS
8	OK AND POS
9	Fast AND POS

The inputs have been fuzzified to obtain confidence levels, and the inference method has generated a rule matrix. The next process in the FL algorithm is *defuzzification*. MATLAB's default procedure is the Centroid Method, which uses the center of gravity (COG) equation shown in Equation 5, with the solutions from each of the rules listed in Table 3. The equation used to derive the COG calculates the area in a particular output MF at height equal to that confidence level derived by that rule. Where μ^{crisp} is the crisp output value. b_i is the center of each output MF and is also called the weight factor, and providing that the MFs are symmetrical it can be shown that the area $\mu_{(i)}$ beneath the confidence level can be derived by multiplying the MF's width by a factor of its height. The equation is listed below, as Equation 6.

$$\mu^{crisp} = \frac{\sum_i b_i \int \mu_{(i)}}{\sum_i \int \mu_{(i)}} \quad (5)$$

$$\int \mu_{(i)} = w_i \left(h_i - \frac{h_i^2}{2} \right) \quad (6)$$

FL DESIGN PROCESS IN MATLAB

1. The FL equivalent of a PD controller is composed of two inputs and one output. The two inputs are defined as the 'error' signal between the source and the feedback, and the 'derror' signal between the current error and the previous error with respect to time. Using MATLAB the first objective is to define the FIS, see Figure 8.
2. Define the fuzzy sets' range values. Select the MF editor through the edit menu, and declare the range of interest (ROI) for each variable. Then use the *edit* menu to add MFs.
3. Generate a matrix using the input variables and their MF names to derive the consequent to each premise. This is accomplished through a series of "IF premise THEN consequent" statements, which are inserted through the Rule-Editor using the fuzzy 'AND' method.
4. Once the rules have been defined then use the built-in Rule Viewer to verify any concept issues that may arise in the future.
5. A FIS has now been created, export it to the MATLAB workspace. Open Simulink and create a circuit using the fuzzy logic controller. Open the controller and insert the name of the FIS just created. The Simulink program is now ready to be run.

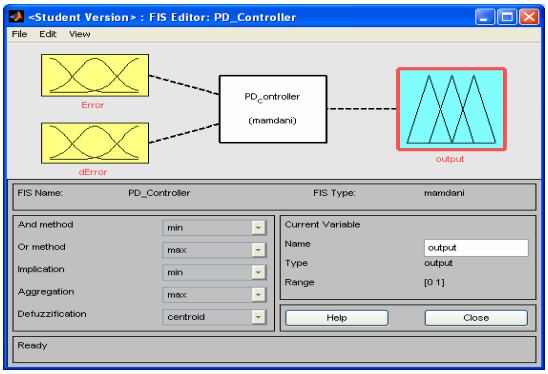

Figure 8. FIS editor defines I/O variables.

3. Conclusions

The FIS is composed of MFs, logical operators, and the linguistic rules. The two types of FIS are Mamdani, and Sugeno. The Mamdani-type FIS was the first of the two to be in the fuzzy world, as Ebrahim Mamdani proposed FL to be used to control a steam engine and boiler in 1975. This method uses the Centroid Method for its defuzzification process. The Sugeno method only works when the output fuzzy sets are linear, or constant.

Development of the two systems certainly seems to be a basic operation, but implementing them in VHDL posed limitations that made them difficult. This was mainly due to the integer mathematics; however, using an offset value equal to $10^{(N/4)}$ solved two issues with the output value: (1) an increased precision during the calculation was realized, and (2) this offset during the derivation process proved to deliver an acceptable output value that did not require compensation. The precision was established by knowing how far to shift the values to the left of the decimal, which is relative to the word size of the project.

The next step is to implement this controller on a motion base here in the lab, and prepare it for use in the wind tunnel test.

4. Acknowledgements

I would like to particularly thank my advisor Dr. Jerry Tucker for his courteous and professional expertise. This association continues professionally in our working together, co-authoring papers, and even co-authoring a mini-course that was taught at the Embedded Systems Conference in San Francisco. As my advisor Dr. Tucker has introduced me to three of his colleagues; two of which are on my thesis review board – Dr. Rosalyn Hobson and Dr. Karla Mossi, and the third presented a mini-course on Advanced DSP topics where I work and tailored it to my needs – Dr. Alen Docef.

I am extending my appreciation to my place of employment as my supervisors throughout my career have allowed me to progress through the education system and achieve my goal of an Advanced Degree in Engineering. These managers include Randy Regan, Steve Jurczyk, Tom Shull, Robert Swain, and James Bell.

Mostly I thank and appreciate my family, who has stuck by my side and helped me no matter how difficult my schedule became. They have given me moral support, financial assistance and emotional care and love so freely with no expectation of a return in their investment.

LIST OF REFERENCES

[1] K.M. Passino and S. Yurkovich, Fuzzy Control. Addison-Wesley, 1998. p. v.
[2] H.T. Nguyen and E.A. Walker, A First Course in FUZZY LOGIC, 3rd Edition. Taylor and Francis Group, LLC, 2006. p. v.
[3] K.M. Passino and S. Yurkovich, Fuzzy Control. Addison-Wesley, 1998. p. 10.
[4] Carl Mills, Gene Monroe, Jeff Herath, Chris Edwards. Adaptive Data Analysis and Processing Technology Final Review. The John Hopkins University Applied Physics Laboratory. P. 50.
[5] Helsinki University of Technology. Fuzzy Logic Control. http://www.control.hut.fi/Kurssit/AS-74.3115/Material/Fuzzy_Control_Slides.pdf
[6] Fuzzy Logic Toolbox. The MathWorks, Inc. 1995
[7] K.M. Passino and S. Yurkovich, Fuzzy Control. Addison-Wesley, 1998. p. 12.
[8] Armin Zinali. Interpolative Fuzzy Inferences Using Least Square Principle. Transactions on Engineering, Computing and Technology V4 February 2005 ISSN 1305-5313. p.245.
[9] Helsinki University of Technology. Fuzzy Logic Control. http://www.control.hut.fi/Kurssit/AS-74.3115/Material/Fuzzy_Control_Slides.pdf
[10] K.M. Passino and S. Yurkovich, Fuzzy Control. Addison-Wesley, 1998. p. 13.

[11] N.S. Nise. Control Systems Engineering third Edition. John Wiley & Sons, Inc. 2000. pp. 329-332.