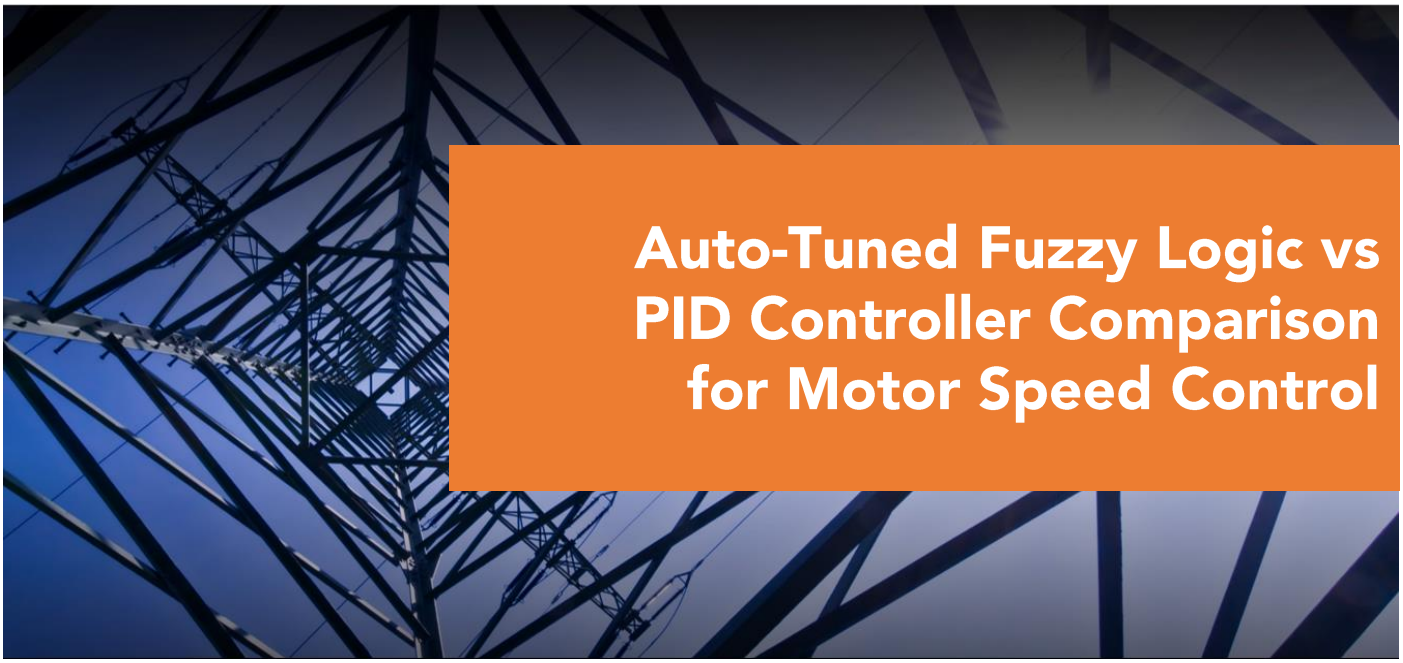


Auto-Tuned Fuzzy Logic vs PID Controller Comparison for Motor Speed Control

RESEARCH TEAM: Rutul Bhavsar
Laith Al-Musawi
Mariano Arriaga

Mohawk IDEAWORKS
ENERGY & POWER
INNOVATION CENTRE

FUNDING AGENCY: Ontario Centres of Excellence

DATE: 17-Mar-2020

REPORT #: EPIC-EP_20-01/1

PREPARED FOR: Xiera Technologies Inc.

Xiera Intelligence in Decision Making

PROJECT PARTICIPANTS – INDUSTRY PARTNER:

Xiera Technologies Inc.

Abstract

The objective of this report is to setup a Single-Input/Single-Output (SISO) control system test to compare the capabilities of a conventional PID controller vs. a novel fuzzy logic-based controller (FLC) developed by Xiera Technologies Inc. The selected test case consists of controlling the speed response of a DC servo motor for single-step input, as well as a torque disturbance scenario. Both controllers were wired/configured similarly to provide a valid comparison framework.

For the base case scenario, the PID controller presents a faster rise time than Xiera's FLC; however, the PID system presents a significant overshoot and longer settling time than the FLC system. Furthermore, the torque disturbance scenario shows that the FLC system stabilizes the motor speed faster and without oscillations when compared to the PID controller for both cases, when torque is applied and removed.

Table of Contents

1 Motivation

For decades, the conventional Proportional-Integral-Derivative (PID) controller has been used to control industrial processes mostly due to its relatively simple implementation and reliability. However, real industrial processes are frequently complex, have a non-linear behaviour, and/or require multiple control inputs. From a control perspective, these characteristics can represent implementation challenges in the mathematical model representation and in the PID tuning process (Babuska & Mamdani, 2011).

On the other hand, intelligent control strategies, such as Fuzzy Logic Control (FLC), can deal with the complexity and non-linear nature of advanced industrial processes. Hence, it can potentially increase process efficiency and economic savings. This advantage is based on the core FLC idea of control action expressed in terms of human operator experience achieving smooth interconnection between distinct control outputs (Arzen & Johansson, 2001). Yet, FLC parameters (knowledge-base) consists of a number of rules, Membership Functions (MFs), and gains that considerably increase with the level of complexity of the problem and the number of variables to control. As a result, one of the main challenges of FLC implementation is the complexity and time-consuming process of generating and tuning FLC rules, MFs and gains, which are generally tuned manually.

In this work Mohawk College, in collaboration with Xiera, has created a validation platform aimed at investigating the capability of Xiera's FLC auto-tuner, and to compare its performance to a conventional PID controller. Here, both controllers are used to regulate the speed of a Servo DC motor. The FLC parameters are auto-tuned using Xiera's software platform and compared to a PID controller auto-tuned using MATLAB PID tuner software.

2 Background

2.1 Proportional-Integral-Derivative Control

PID controllers have been widely used in industry for almost a century (Ho, 2014); it is estimated that 95% of the industrial controllers in the market are PID-type because of their simplicity, clear functionality, applicability and ease of use (Zhang, Wang, & Wang, 2004). Their use extend from simple single variable to multi-variable control systems; including motor drives, automotive & flight control, as well as industrial process control.

One of the main advantages of PID controller is its low number of tuning parameters (proportional, integral and derivative); which for simple systems, Single-Input/Single-Output (SISO), can be relatively easy to tune manually and/or automatically. Auto-tuning processes are common on control systems software; in the case of this study, Matlab/Simulink PID tuner was used. The PID tuner, computes a linear plant model from the Simulink model and designs an initial controller aimed to increase the stability of the system (The MathWorks, Inc., 2019). Furthermore, the PID parameters can be manually modified to improve the response time and transient behaviour of the system.

2.2 Fuzzy Logic Control

Fuzzy logic, a branch of artificial intelligence, was proposed by Lotfi A. Zadeh of the University of California at Berkeley in 1965. The initial FLC applications date from 1974 and focused on controlling a steam engine model industrial plant (Mamdani & Assilian, 1975). By early nineties, Japan started to commercially use FLC technology for home appliance control (Arzen & Johansson, 2001). In recent years, FLC has permeated different commercial products ranging from washing machines, video camera, air conditioning, and automobile anti-lock brake systems (Singh, et al., 2013).

Despite system complexity and vagueness of situations, human operators can make decisions by employing heuristics in the form of linguistic control rules. The concept of fuzzy reasoning expresses such linguistic rules in a rigorous mathematical framework. Fuzzy logic is developed to handle the fuzziness found in human concepts such as those embedded in the knowledge base of an intelligent system, as well as being able to build a framework that can handle linguistic quantifiers such as most, very, somewhat etc. In addition, fuzzy logic can address uncertainties associated with data inaccuracy and process complexity which are problems encountered in industrial applications (Okwu & Nwachukwu, 2018).

2.2.1 Fuzzy Controller Structure

An FLC has three basic blocks (Figure 1):

- Fuzzifier block that maps the measurement signals into fuzzy terms (e.g., high, low etc.),
- Inference engine, or approximate reasoning mechanism, which deduces the control actions in the form of fuzzy terms, and
- Defuzzifier block that translates the fuzzy control actions into crisp output control signals.

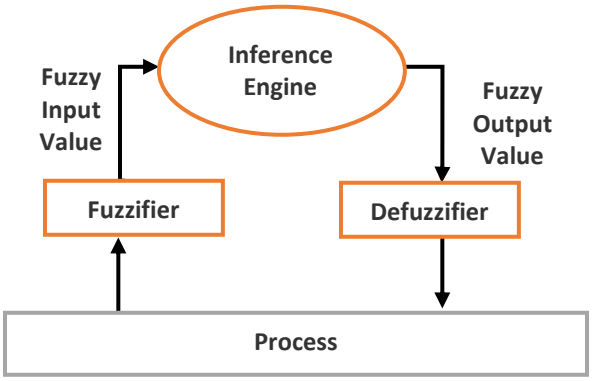


Figure 1: Fuzzy logic controller cycle

2.2.2 The Tuning Problem

There are two problems when fuzzy logic is applied to a real system:

1. The definition of the set of rules, part of the inference engine, is a complicated process that depends on expert knowledge and there is no formal procedure to determine the parameters of the fuzzy system (Kavka, Roggero, & Apolloni, 2003).

2. The inherent complexity of tuning of the parameters associated to the MFs, FLC rules and gains.

When the rules are set properly and parameters tuned correctly, an FLC system can yield a high-performance controller even for complex systems (Azar & Vaidyanathan, 2016). The problem is that tuning fuzzy controllers is a non-trivial task. At present it is performed manually by fuzzy logic experts and, even for a simple application, can take a significant amount of time due to the large number of parameters of the knowledge base and the related fuzzy rules, MFs and gains. Due to this tuning complexity, applications of fuzzy logic have been limited to relatively simple systems, such as home appliance products, and very limited application in large and complex industrial systems, to date.

2.3 Xiera's edeX Platform and Auto-Tuner System

2.3.1 edeX Platform

In recent years, Xiera Technologies Inc. (Xiera) has developed an intelligent auto-tuner addressing the time consuming and complexity issue of FLC knowledge-base parameter tuning. The auto-tuner is embedded as part of a hardware and software FLC platform, edeX (Figure 2); targeted to simplify and expand the implementation of FLCs in complex control systems; while addressing the time consuming and complexity issue of FLC knowledge-base tuning.

Xiera's edeX platform is an interactive fuzzy logic design and development environment intended to offer flexibility and ease-of-use for FLC design and tuning. The platform enables users to model and simulate the target process; as well as designing and testing the FLC system with the aid of an auto-tuner application. In addition, the edeX platform supports the FLC validation process by providing data acquisition capabilities when implementing the controller. edeX is also designed with capabilities to link to third party hardware systems, such as custom-design boards and off the shelf single board controllers (SBCs).

Based on Xiera's experience, edeX can:

- Handle Multi-Input Multi-Output (MIMO),
- Combine fuzzy and conventional controllers in one control system,
- Design and configure fuzzy and/or conventional controllers,
- Run unlimited number of separate fuzzy/conventional controllers on one microcontroller chip, only limited by the chip memory, and
- Handle 32 configurable I/O's.

The hardware component of the edeX platform consists of:

- The fuzzy controller board (Xiera Controller Module or XCM), which carries the microcontroller programmed with the fuzzy algorithms (Figure 3 [a]).
- Up to eight (8) connectable Evaluation Boards (EVBs), each with modular analog Input/output (IO) capability (Figure 3 [b]).
- The XCM features a USB interface to the edex software, for:
    - Data-acquisition,
    - Downloading controller parameters to the XCM's microcontroller,
    - Validating the control loop during development.
- A second serial interface to the XCM used to communicate with EVB or third party host.

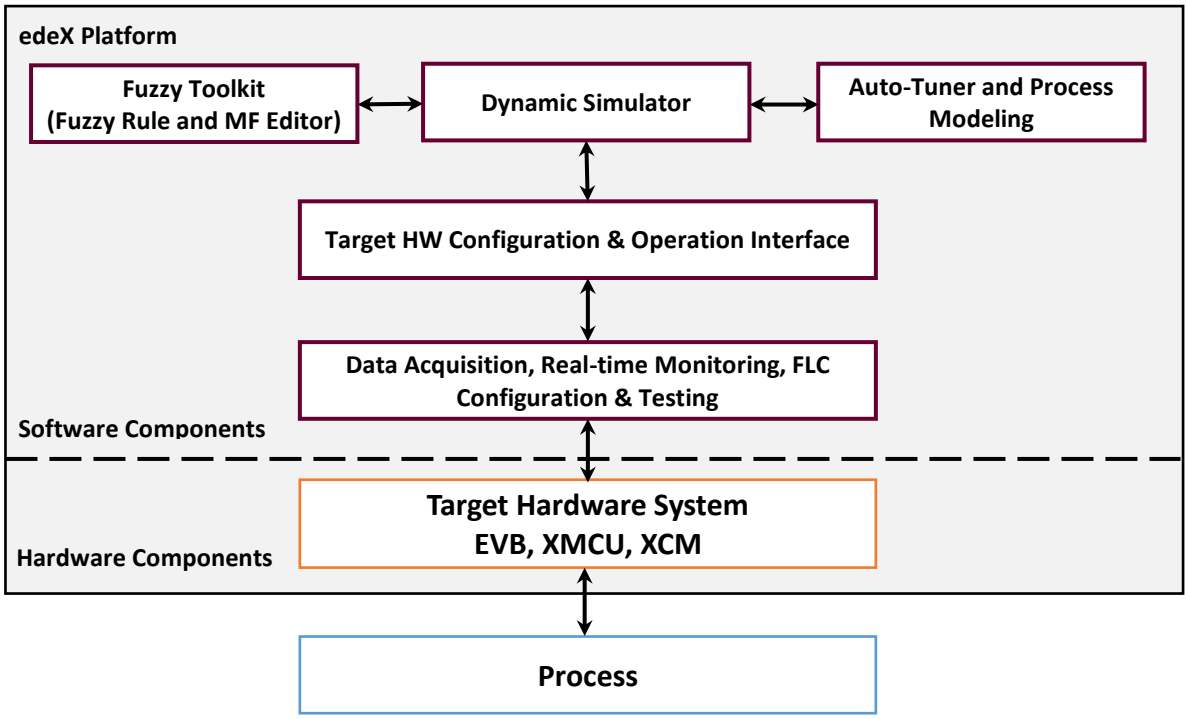
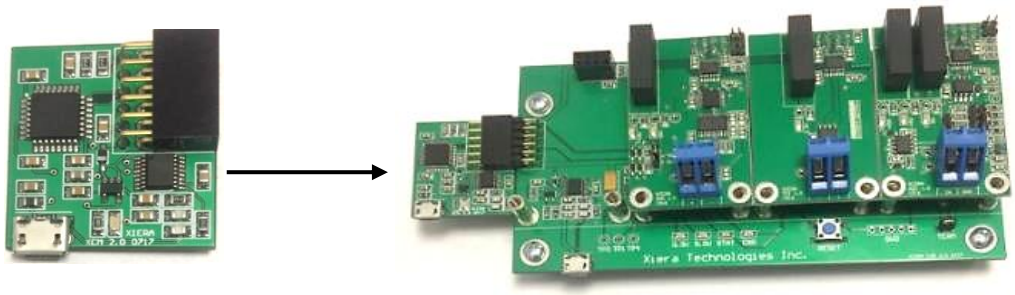


Figure 2 edeX software and hardware components and process interface



[a] [b]

Figure 3 edeX hardware [a] XCM board, [b] XCM board with Evaluation Board and I/O boards

2.3.2 Auto-Tuner System

After the control process model has been properly built on the edeX software platform, the configuration for the edeX auto-tuning parameters can be started. The edeX auto-tuning process can be initiated using default parameters and achieve a stable control process. However, to obtain enhanced control results edeX has the flexibility to modify:

- **Simulation time:** Adjust the sampling frequency of the FLC controller; which can be adjusted depending on the physical characteristics of the control system.
- **Performance evaluation:** Modify the thresholds and constraints involving the auto-tuning (optimization) process including rise time, overshoot, settling time, steady state error, and oscillations.
- **Design variables:** Select the number of parameters/blocks to include as part of the auto-tuning (optimization) process. For example, the user can define to just tune the FLC process gains, or also include MFs and fuzzy rules-related parameters in the auto-tuning process.
- **General Settings:** Specify the number of auto-tuning iterations and the number of best solutions, based on performance evaluation criteria, to keep at the end of the process.

3 DC Motor Speed Control Test Setup

The objective of this project is to compare the performance of the FLC against a PID Controller using a SISO system where PID is typically used. Thus, a DC motor speed control system was selected to test and compare the performance of both FLC and PID controllers. Figure 4 represents the block diagram for the speed control system. The configuration shown in the figure applies to both controller cases and the specific parameters for the motor and transfer functions is described in Section 3.1. A step signal from a generator represents the input signal, set-point, to the system. For the PID controller, a real-time PID model emulated on an OPAL-RT real-time simulator was used and connected to the DC motor, Hardware-In-the-Loop (HIL), further detailed in Section 3.2. For the FLC system, the edeX hardware components were directly connected to the DC motor and the function generator. In addition, an oscilloscope was connected to monitor the tachogenerator voltage. For the PID controller and FLC, the input, process, and output signals were logged using the real-time simulator and the edeX platform, respectively.

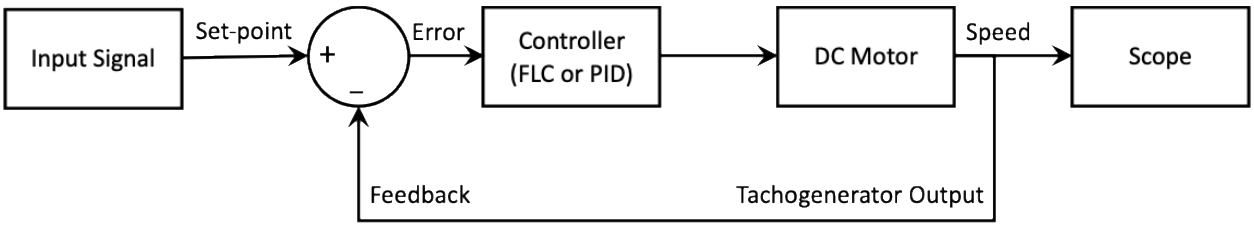


Figure 4: DC motor speed control block diagram

3.1 DC Motor Considerations

3.1.1 DC Motor Physical Characteristics

The characteristics of the selected DC permanent magnet motor are listed onTable 1. Figure 5 shows a picture of the motor with the integrated tachogenerator used as an output signal of the control system. Refer to Appendix I: Function Generator and Motor Drive Specifications for further details of the equipment used in the DC motor test setup.

Table 1: Permanent magnet DC motor specifications – MS 2225 05

Parameter	Value
Viscous Friction	B_m = 5.6 X 10^{-5} N.m/rad/sec.
Inertia	J_m = 6.5 x 10^{-5} Kg/m^2
Back EMF Constant	K_e = 0.115 V/rad/sec.
Torque Constant	K_T = 0.113 N.m/Amp.
Armature Resistance, Total	R_a = 5.8 Ω
Armature Inductance	L_a = 9.2 x 10^{-3}H
Rated Current	2 Amps
Rated Voltage	64 volts
Rated Speed	N = 4900 RPM
Tacho-Generator Constant	K_{tach} = 3v per 1000 RPM

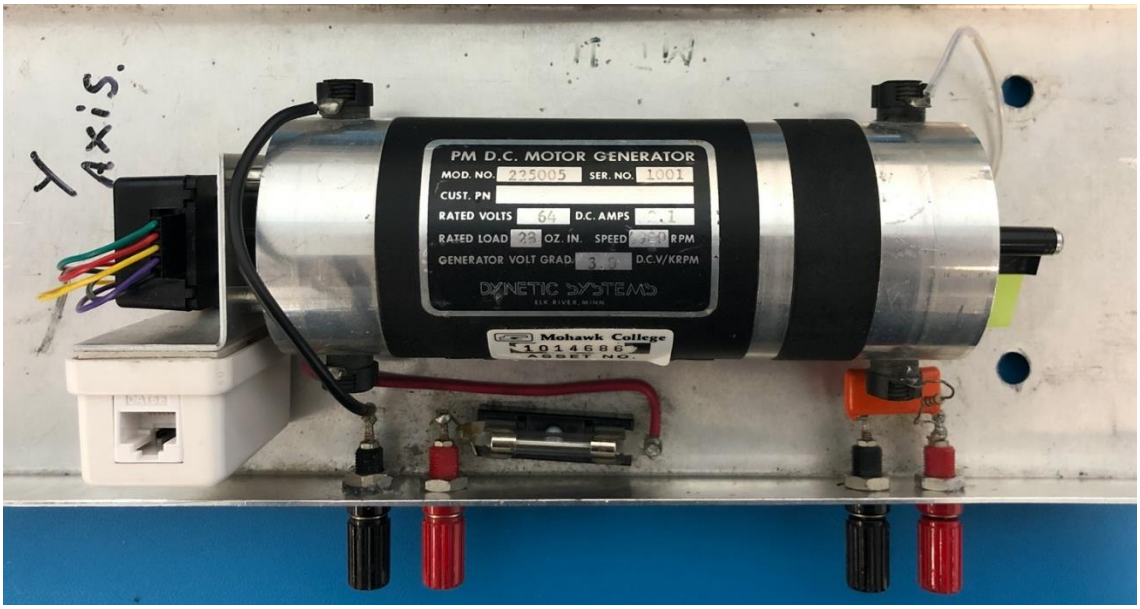


Figure 5: Permanent magnet DC motor generator used for controller testing

3.1.2 DC Motor Transfer Function Model

The initial model with the motor Transfer Function (TF) was obtain by using the following block diagram representation (Figure 6):

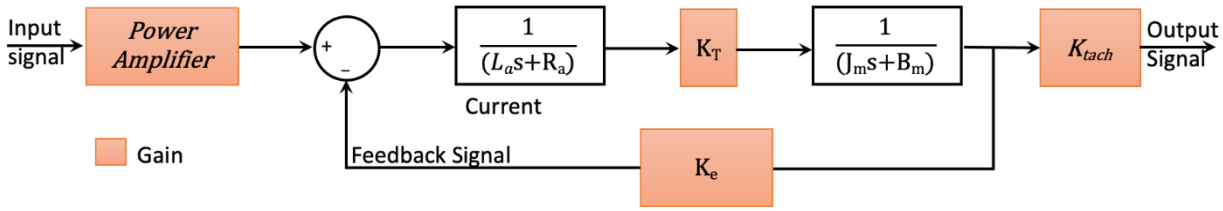


Figure 6: Initial speed control DC motor block diagram

Then, by inputting the DC motor parameters from the previous section, the resulting block diagram (Figure 7), is as follows:

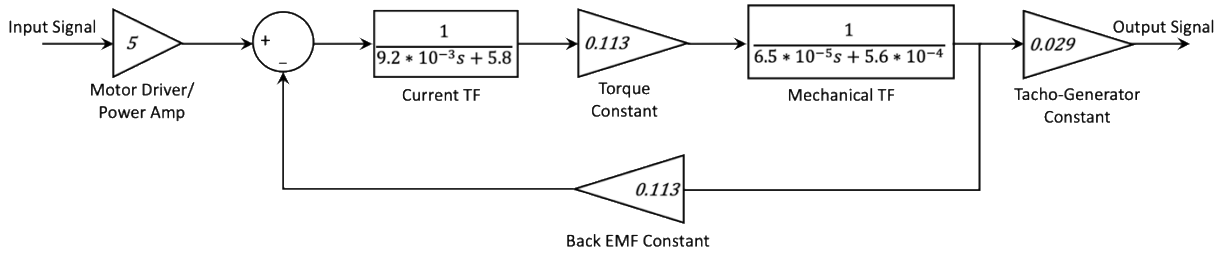


Figure 7: Initial speed control DC motor block diagram with specific motor parameters

The above model was mathematically modified to simplify the control system implementation process (Figure 8); refer to Appendix II: DC Motor Transfer Function Calculation for details on this simplification process.

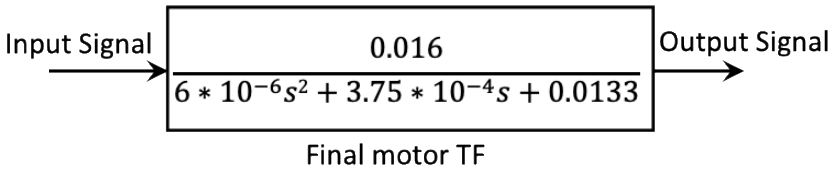


Figure 8: Implemented transfer function model for the speed control system

3.2 PID Controller - DC Motor Test Setup

The PID controller was implemented using an OPAL-RT real-time simulator which enables the implementation of a HIL test environment, refer to Appendix III: Real-Time Simulator Characteristics and Specifications for specifications of the system. The PID controller model was

emulated with the OPAL-RT wired to the DC motor IO, as presented in Figure 4. The OPAL-RT system uses the available models from MATLAB and Simulink libraries to compute the block diagram in Figure 8 as well as the PID controller model (ver. 9.2.0.556344 2017a). Figure 9 shows the diagram of the test setup for the PID configuration in which the motor speed, feedback, signal from the DC motor is connected to the analog input pin of the OPAL-RT and the simulated PID controller signal (analog output) is connected to be the control signal for the DC motor. In addition, the OPAL-RT is used as the data acquisition (DAQ) system, recording the motor speed set-point, controller output and motor speed signals.

The simulation time for both PID and FLC model was kept constant to 50ms, as this is suitable for the mechanical nature of the motor speed control process. The sampling time of the DAQ process was also set at 50ms to adjust to the edeX DAQ sampling rate. The PID controller was auto-tuned using the default parameters of the PID tuner function of MATLAB/Simulink. The resulting parameter values for the PID controller are k_p = 0 (proportional), k_i = 5.52 (integral), and k_d = 0 (derivative).

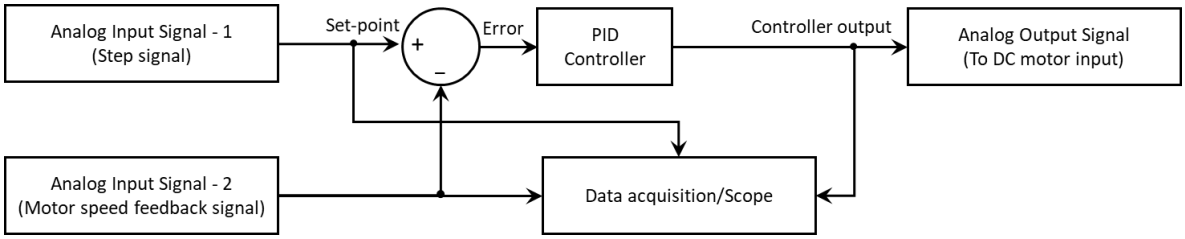


Figure 9: PID controller test setup using a Hardware-In-the-Loop configuration using OPAL-RT

3.3 Fuzzy Logic Controller - DC Motor Test Setup

The FLC test setup is shown in Figure 10 and it is based on the same control process model shown in Figure 4. The function generator (top left) creates a step signal, set-point, that is fed to an analog input module of the edeX EVB. Based on this input, the FLC provides the speed controller signal which is connected to the motor drive, clear box left of DC motor; and the DC motor tachogenerator is connected to an analog input module of the EVB, which is the feedback signal for the FLC. As previously mentioned, the FLC execution and DAQ-sampling time was set at 50ms.

As part of the edeX software setup, the number and type of MFs for the FLC were modified to increase the performance of the control system, based on Xiera's experience. First, the performance evaluation criteria was modified to the values shown in Table 2. The default value for all parameters is zero; however, from an optimization perspective the default settings can

result in an unnecessary longer auto-tuning time and requesting the system to reach as a target an unfeasible and/or unrealistic solution.

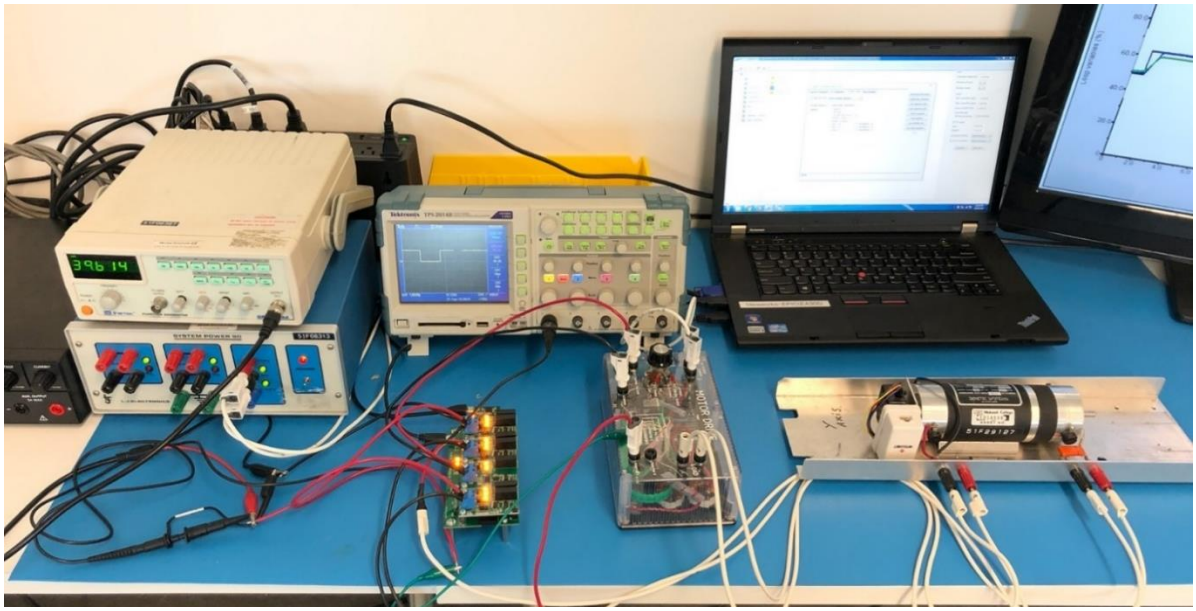


Figure 10: FLC controller test setup

Table 2: Performance evaluation constraints parameters

Constraints	Modified Value
Overshoot (%)	0.01
Rise-time (s)	0.20
Settling time (s)	0.50
Steady-state error (%)	0.01
Oscillations (%)	0.01
Asymptotic slope (%)	0.01

Once the performance evaluation constraints were updated; as part of the design variables, the rule actions, gains and integral were selected to be tuned as a part of the optimization process. The resulting knowledge-base rules, error MFs, change of error MFs, and FLC output MFs are shown in Figure 11 to Figure 14, respectively.

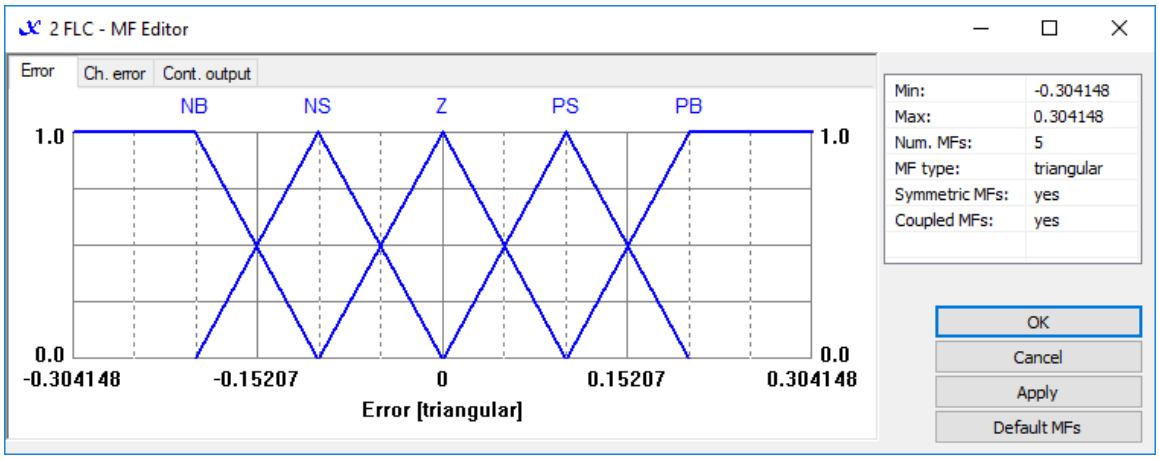


Figure 11: FLC error membership functions

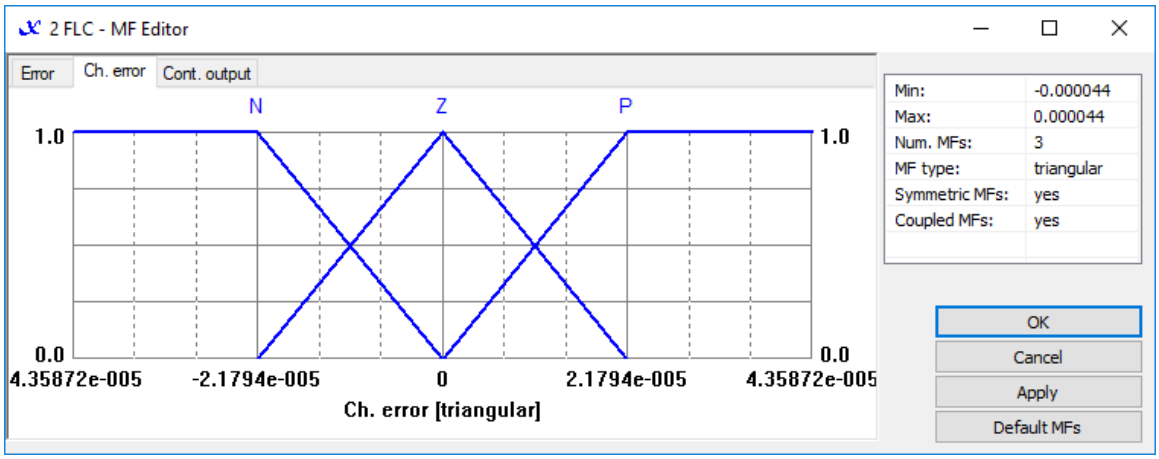


Figure 12: FLC change of error membership functions

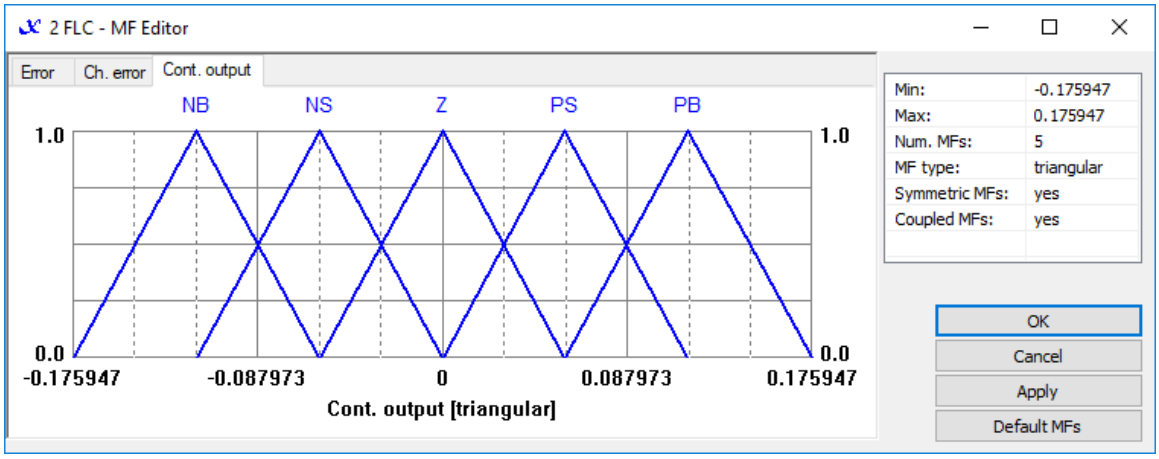
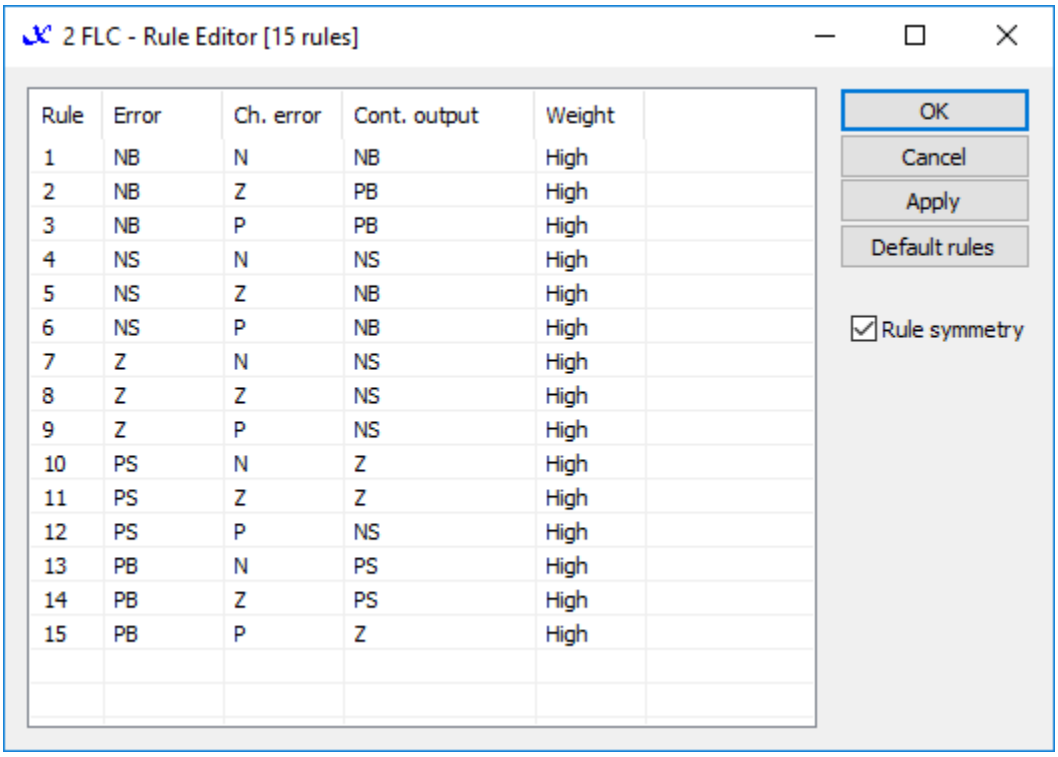


Figure 13: FLC output membership functions



Rule	Error	Ch. error	Cont. output	Weight
1	NB	N	NB	High
2	NB	Z	PB	High
3	NB	P	PB	High
4	NS	N	NS	High
5	NS	Z	NB	High
6	NS	P	NB	High
7	Z	N	NS	High
8	Z	Z	NS	High
9	Z	P	NS	High
10	PS	N	Z	High
11	PS	Z	Z	High
12	PS	P	NS	High
13	PB	N	PS	High
14	PB	Z	PS	High
15	PB	P	Z	High

Figure 14: FLC knowledge-base rules

4 PID and Fuzzy Logic Controller Test Results

This section presents the results for the PID and FLC test system for two case studies with the objective of recording and comparing their performance, the studied cases are:

1) **Base Case:** Step function applied to the DC servo motor.
2) **Torque Disturbance Case:** Once the motor is running at constant speed, torque is applied to the end of the DC motor shaft and then removed.

The tests include the calculation of the following control systems performance characteristics using the default calculation used in MATLAB/Simulink (The MathWorks, Inc., 2019).

- **Rise time:** The time taken for the output to go from 10% to 90% of the set-point input. In this case a step signal from 0 to 1.
- **Steady-State-Error:** Steady state error the difference between the set-point and the feedback signal steady state signal values.
- **Settling Time:** The time it takes for the error between the setpoint and the response value to reach a steady-state response; with the difference being within 2% of setpoint.
- **Overshoot:** Overshoot is the percentage difference of the maximum peak value of the signal to the final value, steady state value.

4.1 Base Case Results

The results recorded for the PID controller using OPAL-RT's are shown in Figure 15. The three signals shown are the set-point (black), the speed control signal to DC motor (red), and the motor speed feedback (blue). The units in the y-axis are in Volts representing a 'per unit' base for the results to simplify the comparison among the three signals. The PID controller results shows that the rise time for the motor speed signal is ≈0.125 sec with a 56% overshoot value. In addition, the PID control system has a settling time of ≈1.6 seconds after going through an oscillation phase; yet the PID controller achieves a steady-state error of practically zero.

Figure 16 shows the response of the auto tuned FLC for the same base case. For comparison purposes, the signals plotted are the same as the PID case. The results show that the rise time for the motor speed signal is ≈0.6 sec, a settling time of ≈0.65 sec, a 0% overshoot and reaching a steady-state error of zero. Given, the selected performance criteria, PID has a faster (37% faster) risetime when compared to the FLC system; however, the FLC outperformed the PID controller from an overshoot perspective since there was zero overshoot; and from a settling time perspective the FLC system was 146% faster than the PID controller system.

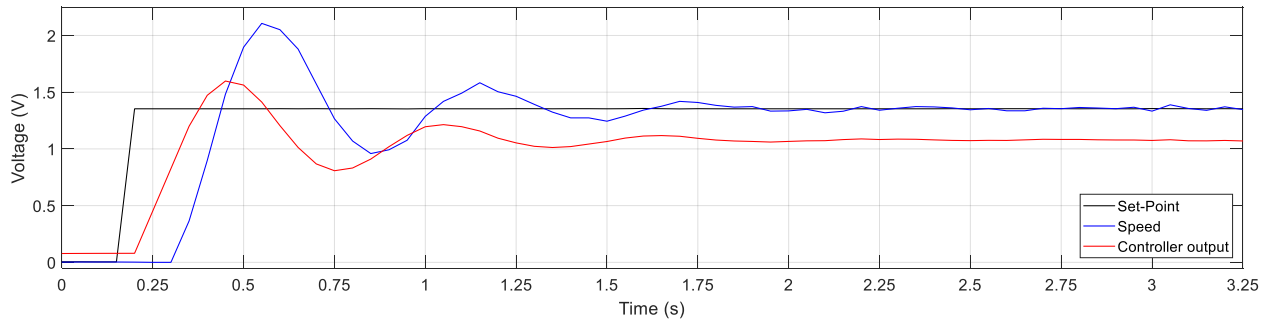


Figure 15: PID controller step response for base case

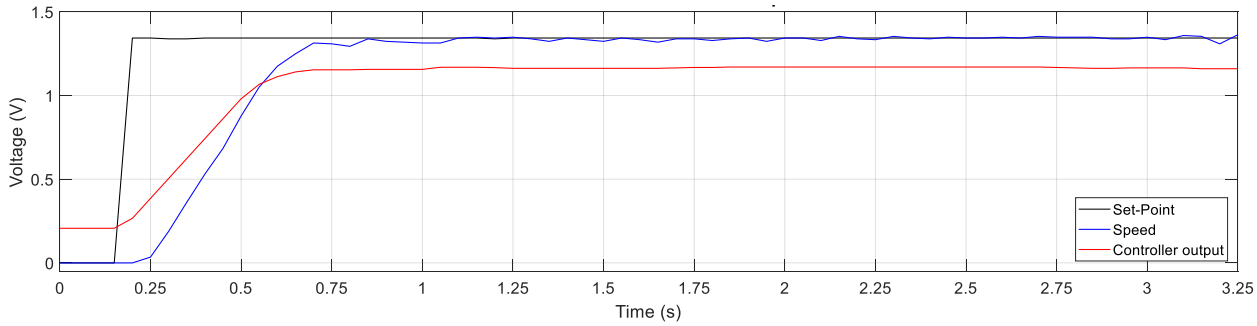


Figure 16: FLC step response for base case

4.2 Torque Disturbance Case Results

The Torque Disturbance Case analyzes the performance of both speed controllers when torque is applied to the motor shaft while running at constant speed. Pressure was applied to the motor shaft to introduce torque (load the motor), hence the torque is neither constant nor consistent for the same test. Yet, as seen from the respective figures below, the torque was maintained as constant and as equal as possible for the PID and FLC case respectively.

Figure 17 presents the auto tuned PID controller response. The figure first shows the base case step response; then, the torque disturbance was applied only after the motor has reached a steady-state (constant) speed. The torque is approximately at $t \approx 3.5$ sec, the PID output (red) increases to compensate for the speed lose in the motor (blue) which initially makes the motor speed to drop by 39% of the step set-point. Then, the PID controller is able to recover the motor speed in ≈1.1 sec and then continue with slight oscillations, likely due to the small torque variations. When the applied torque is removed (t≈5.62 sec), the motor speed increases by 56% of the set-point, followed by similar oscillations as in the PID base case and reaching steady state after ≈1.5 sec of the torque being applied.

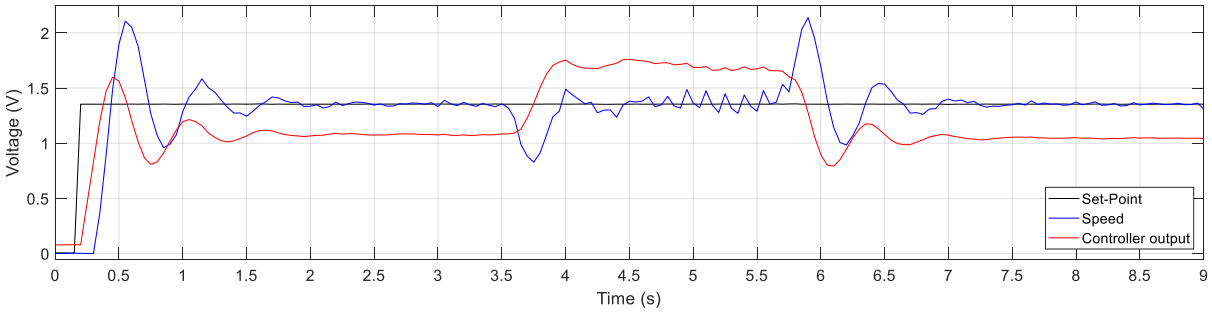


Figure 17: PID controller response for torque disturbance case

The torque testing results for the FLC system are shown in Figure 18. Similar as with the PID controller torque case, the test first applies the step signal and waits until the motor speed remains constant. Then, at t≈3.68 sec, manual torque is applied to the shaft. As a result, the FLC output (red) increases to compensate for the lost motor speed (blue) which drops 27% below the set-point. The FLC controller recovers the motor speed in ≈0.82 sec after the torque is applied and then continue with oscillations, likely due to the small torque variations. When the applied torque is removed (t ≈7.62 sec), the motor speed increases by ≈42% of the setpoint and then reaches steady state in ≈0.7 sec after the torque is removed, without any oscillations.

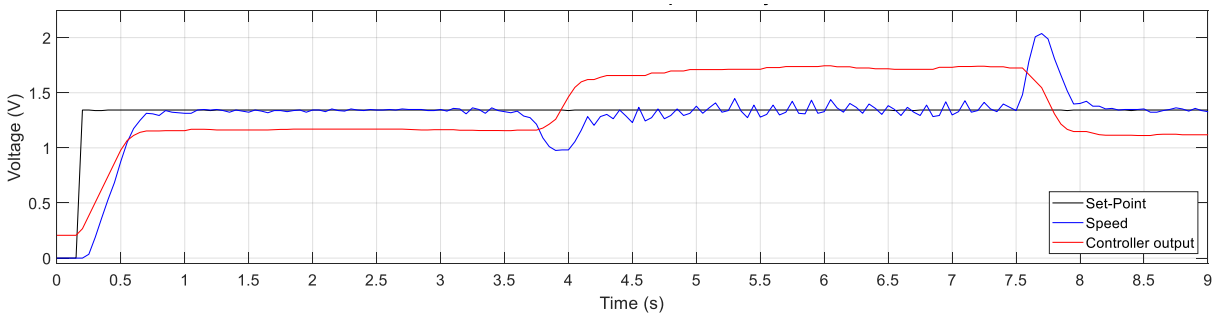


Figure 18: FLC response for torque disturbance case

4.3 Results Summary

Table 3 and Table 4 summarize the values of the performance criteria used to compare the PID and FLC systems for the Base and Torque Disturbance cases[1].

Table 3: Data Summary for Base Case

Controller	Rise time (s)	Settling time (s)	Overshoot (%)
PID	0.125	1.60	56
FLC	0.60	0.65	0

Table 4: Data Summary for Torque Disturbance Case

Controller	Recovery time (s)	Undershoot (%)	Overshoot (%)
Applied Torque			
PID	1.1	39	-
FLC	0.82	27	-
Removed Torque			
PID	1.5	-	56
FLC	0.7	-	42

[1] The steady state errors for all test cases are not shown in the table since from a practical perspective they are equal to zero.

5 Conclusion

This report presented the methodology and results for testing a SISO system, considering the speed control system for a DC motor using a PID controller and an FLC. The performed test included a base case considering a step signal and an applied/removed torque case. The main objective of this test was to compare the control performance between the PID and FLC systems; which is summarized as follows:

- For the base case (step signal), the PID controller presented a faster rise time; however, it also presented oscillations and a longer settling time than the FLC controller. In both cases, the controllers reached a practically zero steady state error.

- For the torque disturbance case, the FLC presented a faster recovery time when the torque was applied and removed, as well as having a lower overshoot than the PID controller.

References

Arzen, K. E., & Johansson, M. (2001). Fuzzy Control: From Heuristic PID to Optimization-Based Nonlinear Control. In H. B. Verbruggen, & R. Babuska, *Fuzzy Logic Control Advances in Application* (p. 331). World Scientific Publishing Co. Pte. Ltd.

Azar, A. T., & Vaidyanathan, S. (2016). *Advances in Chaos Theory and Intelligent Control.* Switzerland: Springer International Publishing .

Babuska, R., & Mamdani, E. (2011, October 21). *Fuzzy Control.* Retrieved December 10, 2019, from Scholarpedia.org: http://www.scholarpedia.org/article/Fuzzy_control

Control Engineering. (2018, June 01). *Pros and cons of autotuning control: Part 1*. Retrieved March 03, 2020, from https://www.controleng.com/: https://www.controleng.com/articles/pros-and-cons-of-autotuning-control-part-1/

Goel, A., Uniyal, A., Bahuguna, A., Patwal, R. S., & Ahmed, H. (2012). Performance comparision of PID and Fuzzy logic controller using different defuzzificztion techniques for positioning control of DC motor. *Journal of Information Systems and Communication, 3*(1), 235 - 238. Retrieved December 11, 2019, from https://bioinfopublication.org/files/articles/3_1_49_JISC.pdf

Gouda, M. M., Danaher, S., & Underwood, C. P. (2000, September). Fuzzy logic control versus conventional PID control for controlling indoor temprature of a building space. *33*(24), 249-254. Retrieved December 11, 2019

Ho, A. K. (2014). *PDHonline Course E331 (3 PDH).* Retrieved December 11, 2019, from www.pdhonline.com/: https://pdhonline.com/courses/e331/e331content.pdf

Kavka, C., Roggero, P., & Apolloni, J. (2003). An architecture for fuzzy logic controller evolution and learning in microcontroller based environment.

Mamdani, E. H., & Assilian, S. (1975, January). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Made Studies, 7*(1). Retrieved December 09, 2019, from https://www.sciencedirect.com/science/article/abs/pii/S0020737375800022

Okwu, O. M., & Nwachukwu, A. N. (2018). A review of fuzzy logic applications in petroleum exploration, production and distribution operations. *Journal of Petroleum Exploration and Production Technology*, 14.

Singh, H., Gupta, M. M., Meitzler, T., Hou, Z.-G., Garg, K., Solo, A., & Zadeh, L. (2013). Real-Life Applications of Fuzzy Logic. *Hindwani*. Retrieved from Hindwani: https://www.hindawi.com/journals/afs/2013/581879/

The MathWorks, Inc. (2019). *PID Controller Tuning in Simulink*. Retrieved from Mathworks: https://www.mathworks.com/help/slcontrol/gs/automated-tuning-of-simulink-pid-controller-block.html

The MathWorks, Inc. (2019). *Stepinfo*. Retrieved from mathworks: https://www.mathworks.com/help/ident/ref/stepinfo.html

Vaishnav, S. R., & Khan, Z. J. (2007). Design and Performance of PID and Fuzzy Logic Controller with Smaller Rule Set for Higher Order System. *World Congress on Engineering and Computer Sciences .* San Francisco: World Congress on Engineering and Computer Sciences .

Zhang, J., Wang, N., & Wang, S. (2004). A developed method of tuning PID controllers with fuzzy rules for integrating process. *American Control Conference* (pp. 1109 - 1114). Boston: American Control Conference.

Appendix I: Function Generator and Motor Drive Specifications

Table 5: Function Generator (Instek GFG-8216A) Specifications

Parameter	Value
Frequency Range	0.3Hz ~ 3MHz (7 Range)
Amplitude	>10Vpp (into 50Ω load)
Impedance	50Ω±10%
Attenuator	-20dB±1dBx2
DC Offset	<-5V ~ >5V (into 50Ω load)
Duty Control	80% ~ 20%, maximum 1MHz (continuously adjustable)
Display	6 digits LED display
Power Source	AC115V/230V±15%, 50 / 60Hz
Dimensions & Weight	251(W) x 91(H) x 291(D)mm, Approx. 2.1 kg

Table 6: Motor Driver (Electro craft DA4709) Specifications

Parameter	Value
Electrical Data	
Power Supply Voltage	+11 to +70 VDC, (Residual ripple <5 %)
Auxiliary Voltage Input	+5 to +30 VDC
Nominal Current	9 / 18 A (model dependent)
Peak Current	18 / 36 A (model dependent)
Maximum	630 / 1260 W (model dependent)
Switching Frequency	50 kHz
Efficiency	95 %
Digital and Analog Inputs	
Enable	Active High TTL, +24 VDC; Resistance = 4,7 kOhm
Ramp	Active High TTL, +24 VDC; Resistance = 4,7 kOhm
I max	Set value Analog -10 – +10 VDC ; Resistance = 20 kOhm
Tacho	Analog -50 – +50 VDC ; Resistance = 50 kOhm
Outputs	
Auxiliary Voltage Output +5V	+5 V / 50 mA
Auxiliary Voltage Output +10V	+10 V / 20 mA
Auxiliary Voltage Output -10V	-10 V / 20 mA
Error	Open Collector / Push Pull / TTL /+24 VDC; R = 50 Ohm
Monitor I	Analog 0 – +10 VDC ; Resistance = 200 Ohm; max. 20 mA
Monitor n	Analog 0 – +10 VDC ; Resistance = 200 Ohm; max. 20 mA

Appendix II: DC Motor Transfer Function Calculation

The figure below shows the initial setup for determining the transfer function of the motor using the specification of the motor provided in Table 1.

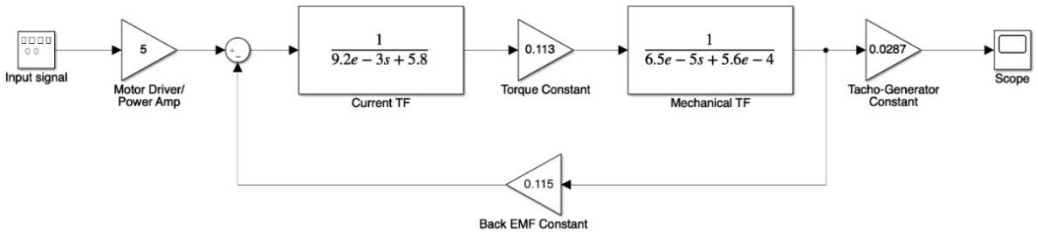


Figure 19: Initial transfer function model for the speed control system

Combining the torque constant and the current/mechanical TF above, the resulting system is:

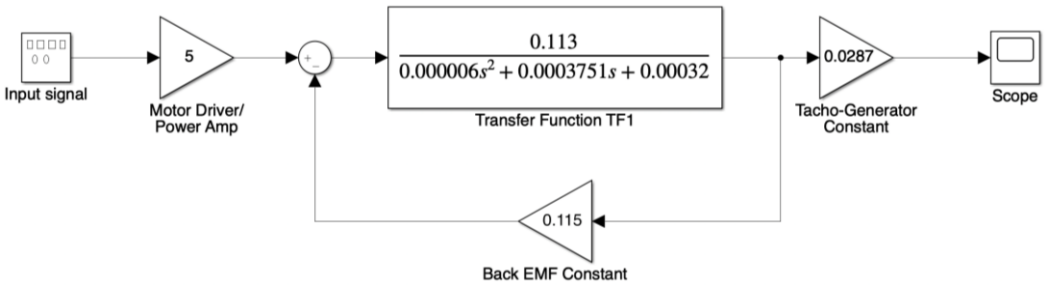


Figure 20: reduced model for new Transfer function (TF1)

Then, solving the parallel connection between TF1 and the back Electromotive force (EMF) constant, and multiplying by the tacho-generator constant gain, the reduced model is:

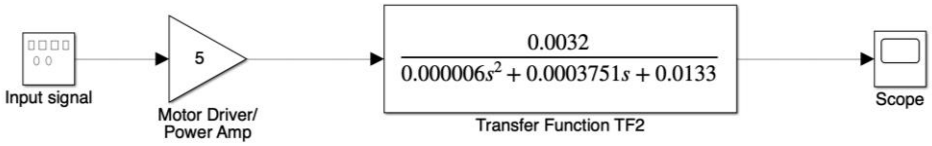


Figure 21: Reduced transfer function TF2

Finally, TF2 can be multiplied by the motor driver/power amp gain, resulting in the final implemented TF given by:

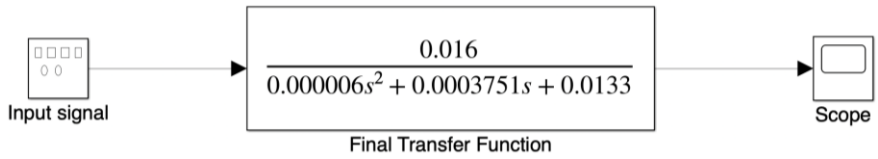


Figure 22: Implemented transfer function model for the speed control system

Appendix III: Real-Time Simulator Characteristics and Specifications

Matlab/Simullink software version details:

```
>> ver
-----------------------------------------------------------------------------------------------------
MATLAB Version: 9.2.0.556344 (R2017a)
MATLAB License Number: 40755337
Operating System: Microsoft Windows 10 Pro Version 10.0 (Build 17134)
Java Version: Java 1.7.0_60-b19 with Oracle Corporation Java HotSpot(TM) 64-Bit Server VM mixed mode
-----------------------------------------------------------------------------------------------------
MATLAB                                                  Version 9.2          (R2017a)
Simulink                                                Version 8.9          (R2017a)
ARTEMIS Blockset                                        Version 7.3.2.1271     (R2017a)
Control System Toolbox                                  Version 10.2         (R2017a)
Data Acquisition Toolbox                                Version 3.11         (R2017a)
Database Toolbox                                        Version 7.1          (R2017a)
Fuzzy Logic Toolbox                                     Version 2.2.25       (R2017a)
Instrument Control Toolbox                              Version 3.11         (R2017a)
MATLAB Coder                                            Version 3.3          (R2017a)
OPC Toolbox                                             Version 4.0.3        (R2017a)
Optimization Toolbox                                    Version 7.6          (R2017a)
RT-EVENTS Blockset                                      Version 4.3.2.749     (R2017A.x)
RT-LAB                                                  Version v11.3.3.62     (R2017a.x)
Signal Processing Toolbox                               Version 7.4          (R2017a)
Simscape                                                Version 4.2          (R2017a)
Simscape Power Systems                                  Version 6.7          (R2017a)
Simulink 3D Animation                                   Version 7.7          (R2017a)
Simulink Coder                                          Version 8.12         (R2017a)
Simulink Control Design                                 Version 4.5          (R2017a)
Simulink Design Optimization                            Version 3.2          (R2017a)
Simulink Desktop Real-Time                              Version 5.4          (R2017a)
Spreadsheet Link                                        Version 3.3.1        (R2017a)
Statistics and Machine Learning Toolbox                 Version 11.1         (R2017a)
Symbolic Math Toolbox                                   Version 7.2          (R2017a)
System Identification Toolbox                           Version 9.6          (R2017a)
eFPGAsim                                                Version v1.5.4.41-4     (R2017a)
```

Figure 23: MATLAB add-ons tools and version

The real-time simulator form OPAL-RT consists of two pieces of hardware; OP5700 RCP/HIL FPGA-Based Real-Time Simulator and OP8660 HIL Controller and Data Acquisition Interface.

The OP5700 simulator contains a target computer in the lower section, which uses OPAL-RT's RT-LAB tools to run simulations and the upper section contains the FPGA and the I/O conditioning modules (Figure 24).

The target computer in lower section can be connected to a network of simulator or can be used as a standalone system and some of its features are::

- ATX motherboard
- Linux-based real-time operating system
- Intel® Xeon® E5 CPU with 4, 8, 16 and 32 processor cores, up to 3.2GHz
- 10MB Cache Memory per 4 cores

- up to 32GB of DRAM,
- 512GB SSD disk,
- 6 PCIe slots1, used to connect the internal FPGA board and PCIe or PCI third party I/O and communication cards.

Figure 24: Opal-RT real time simulator

Features of the FPGA and the I/O conditioning modules include:

- Xilinx® Virtex®7 FPGA programmable from the target computer via PCIe. The FPGA is used to
- execute models designed with the OPAL-RT's RT-XSG tool, manage the I/O lines and execute
- embedded FPGA-based simulations. It exchanges data with the real-time simulations running on
- the target computer CPUs via the PCIe link.
- Flat carrier board capable of connecting any combination of up to 8 digital and analog conditioning modules.
- Each module controls 16 or 32 lines for a total of up to 256 I/0 lines.
- 16 SFP ports for high speed communication with other FPGA-based systems or with external devices.

In general, some of the main features of the simulator are as shown in the table below:

OP5700 SPECIFICATIONS

Product name	OP5700
CPU	Intel® Xeon® E5, 4 core / 3GHz to 32 cores / 2.3GHz
FPGA	Xilinx® Virtex®7 FPGA on VC707 board, 485T, 485 760 Logic cells, 2 800 DSP slices
I/O lines	256 lines, routed to 8 analog or digital, 16 or 32 channel, conditioning modules
High speed communication ports	16 SFP sockets, up to 5GBps
I/O connectors	4 panels of 4 DB37F connectors
Monitoring connectors	4 panels of RJ45 connectors
PC interfaces	Standard PC connectors (monitor, keyboard, mouse and network)
PCIe slots	6 on-board PCIe slots. 4 or 5 slots are available for third party cards or riser boards depending on CPU configuration.
Hard disk	512 GB SSD
Power supply	Universal input and active power factor correction 650W continuous power Input : 100-240VAC, 50-60Hz, 10A/5A Power: 600W
Dimensions	22.27 x 47.7 x 49.3cm (8.75" x 18.8" x 19.4") HxWxD
Weight	17kg (37.5 lbs)
Operating temperature	10 to 35 °C (50 to 95°F)
Storage temperature	-55 to 85°C (-67 to 185°F)
Maximum rated ambient temperature	40C° (104°F)
Relative humidity	10 to 90% non-condensing
Maximum altitude	2000 m (6562 ft.)

Figure 25: OP5700 Specifications

The **OP8660 HIL Controller and Data Acquisition Interface** is designed to be used with a real-time simulator (such as OP5700) to provide supplementary signal conditioning.

The rear of the chassis provides DB37 connectors to connect the OP8660 to the real-time simulator, while the front provides connectors (banana jack or DB9) for connecting devices such inverters, encoders, monitoring and measuring devices for monitoring or testing. Which in our case is the FLC controller.

The unit includes four high current and high voltage input conditioning modules, which convert high current and high voltage signals coming from the external device to ±10V voltage signals compatible with the real-time simulator's inputs. The HIL Controller is useful link between the unit under test (ECU, motor controller, etc.) and the simulator, you can insert a fault at any point in the test to assess how the unit reacts to the fault. Some of the main features and specifications of this unit are as below:

- DB9 inverter and encoder connectors.
- Banana jack high current and high voltage measurement connectors.
- Banana jack analog input (+/-16V) monitoring connectors

- Banana jack analog output (+/-16V) interface connectors
- Banana jack digital input (0-30V) monitoring connectors
- Banana jack digital output (0-5V) interface connectors
- DB37 connectors for quick connections to the real-time simulator (all DB37 use common pin assignments).

General Specifications:

Product name	OP8660 HIL Controller
Part number	310-0055
Form factor	4 U
Dimensions	13.33 x 48.26 x 30.8cm HxWxD (5.25” x 19” x 12.125”)
I/O connectors	DB37F, DB9, banana jacks
Operating temperature	10 to 40 ºC (50 to 104ºF)
Storage temperature	-55 to 85ºC (-67 to 185ºF)
Relative humidity	10 to 90%, non-condensing
Maximum altitude	2,000 m (6562 ft.)

Sensor Specifications:

This specification applies to the High Current and High Voltage Measurement connectors in the front of the OP8660.

	Current Sensors Specification	Voltage Sensors Specification
Input range:	15 A	Up to 600 volts
Signal output range:	± 10 Volts	± 10 Volts
Isolation:	Galvanic, 2.5 Kv	Greater than 200 volts after the resistive divider
Bandwidth:	DC to 100 kHz	DC to 100 kHz
Linearity:	< 0.2%	< 0.2 %
Rise time:	< 2 Microseconds	< 2 microseconds
Power supplies:	±15 Volts	±15 Volts