Genetically Generated Double-Level Fuzzy Controller with a Fuzzy Adjustment Strategy

Sofiane Achiche
Technical University of Denmark
Nils Koppels Allé, Building 404
2800 Kgs. Lyngby, DK
+45 4525 4166
sac@gst.mek.dtu.dk

Wang Wei
Technical University of Denmark
Nils Koppels Allé, Building 404
2800 Kgs. Lyngby, DK
+45 4525 6271
ww@mek.dtu.dk

Zhun Fan
Technical University of Denmark
Nils Koppels Allé, Building 404
2800 Kgs. LyngbyDK
+45 4525 6271
zf@mek.dtu.dk

Ali Ozkil
Technical University of Denmark
Nils Koppels Allé, Building 404
2800 Kgs. Lyngby, Copenhagen, DK
+45 4525 6271
s053675@student.dtu.dk

Torben Sørensen
Technical University of Denmark
Nils Koppels Allé, Building 404
2800 Kgs. Lyngby, Copenhagen, DK
+45 4525 6278
ts@mek.dtu.dk

Jiachuan Wang
Systems Department
United Technologies Research Center
East Hartford, 06128, USA
+1 860 610 7549
WangJ2@utc.com

Erik Goodman
Michigan State University
East Lansing, 48823
Michigan, USA
+1 517 355 6453
goodman@msu.edu

ABSTRACT
This paper describes the use of a genetic algorithm (GA) in tuning a double-level modular fuzzy logic controller (DLMFLC), which can expand its control working zone to a larger spectrum than a single-level FLC. The first-level FLCs are tuned by a GA so that the input parameters of their membership functions and fuzzy rules are optimized according to their individual working zones. The second-level FLC is then used to adjust contributions of the first-level FLCs to the final output signal of the whole controller, i.e., DLMFLC, so that it can function in a wider spectrum covering all individual working zones of the first-level FLCs. The second-level FLC is again optimized by a GA. An inverted pendulum system (IPS) is used to demonstrate the feasibility of the approach.

Categories and Subject Descriptors
I.2 [Artificial Intelligence]: Problem Solving, Control Methods, and Search – *Heuristic methods.*

General Terms: Design

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

Keywords
Genetic algorithm, fuzzy logic controller, modularity.

1. INTRODUCTION
Several research and industrial applications concentrated their efforts on providing simple and easy control algorithms to cope with the increasing complexity of the controlled processes/systems [1]. The design method for a controller should enable full flexibility in the modification of the control surface [2]. The systems involved in practice are, in general, complex and time variant, with delays and nonlinearities, and often with poorly defined dynamics. Consequently, conventional control methodologies based on linear system theory have to simplify/linearize the nonlinear systems before they can be used, but without any guarantee of providing good performance. To control nonlinear systems satisfactorily, nonlinear controllers are often developed. The main difficulty in designing nonlinear controllers is the lack of a general structure [3]. In addition, most linear and nonlinear control solutions developed during the
last three decades have been based on precise mathematical models of the systems. Most of those systems are difficult/impossible to be described by conventional mathematical relations, hence, these model-based design approaches may not provide satisfactory solutions [4]. This motivates the interest in using FLC; FLCs are based on fuzzy logic theory [5] and employ a mode of approximate reasoning that resembles the decision making process of humans. The behavior of a FLC is easily understood by a human expert, as

knowledge is expressed by means of intuitive, linguistic rules. In contrast with traditional linear and nonlinear control theory, a FLC is not based on a mathematical model and is widely used to solve problems under uncertain and vague environments, with high nonlinearities [6][7]. Since their advent, FLCs have been implemented successfully in a variety of applications such as insurance and robotics [8][9][10]. Unlike neural networks, the first-generation fuzzy systems are not able to learn from data. However, several techniques have been proposed to extract fuzzy rules from training data gathered from observation of the operator control strategy [11][12][13]. As a consequence, an emerging topic in the FLC research community has been to investigate ways to help designers to automatically design FLCs. The majority of the work uses computational intelligence techniques, including neural networks [14][15][16] and genetic algorithms (GA) [17][18][19]. This paper concentrates on the use of a GA. Various approaches have been used to design FLCs using a GA, either concentrating on the fuzzy control rules [20], the membership functions [21], or both [22][23][24]. However, a GA applied to design FLCs has some shortcomings too, mainly because the FLC is built and selected according to the fitness value, which only measures how close the solution is to the desired result under predefined environments, and may therefore decrease its generalization to changing or uncertain environments. FLCs tend to show good control behavior only in ranges near the values of the training sets used to design them [25]. To avoid this problem, the approach inspired by [26] is used in this paper; an evolutionary modular fuzzy system is used to get a combined system that performs better, by combining different modules together; every module in the final system produces an output. These outputs are combined by a function (linear or nonlinear) that produces the final output. However, finding the right function for the right application is a tedious task. In this paper, a genetically tuned FLC is used to combine the outputs of the various FLCs, resulting in a double-level modular FLC (DLMFLC). The existing FLCs will be the modules (first-level FLCs) and the second-level FLC will combine these modules. To validate the approach, an experiment is carried out to control and stabilize an inverted pendulum system with changing force disturbances using DLMFLC. The simulation results demonstrate that DLMFLC can successfully stabilize the inverted pendulum system under the whole spectrum of changing force disturbances, while the first-level FLCs can usually only work in narrower work zones. This paper is organized as follows: section 2 presents some introductory material about fuzzy logic; section 3 gives a short introduction to the GA. Section 4 explains the details of DLMFLC. As a case study, the problem of an Inverted Pendulum system (IPS) is defined in section 5, with simulation results presented and discussed. The paper concludes with section 6, in which some future research directions are also pointed out.

2. FUZZY LOGIC

In this section we present a rule-based approach to decision making and control using fuzzy logic techniques, based on the compositional rule of inference (CRI). This approach is used to handle uncertain (imprecise) knowledge and was developed in the sixties by L.A. Zadeh [27]. Such knowledge can be collected and delivered by a human expert (e.g. decision-maker, designer, process planner, machine operator, etc.) or automatically generated by a learning algorithm using synthetic or experimental data. The CRI may be written in the form:

$$U' = (C' \times \ldots \times B' \times A') \circ R \qquad (1)$$

where R represents the global relation that aggregates all the rules (knowledge base). A′, B′, ..., C′ represents the inputs (observations) and U′ represents the output (conclusion). The symbol ∘ represents the CRI operator. The knowledge base consists of two components: the linguistic term base (database) and the fuzzy production rule base, while the database is divided in two parts: fuzzy premises and fuzzy conclusions. More about fuzzy logic and fuzzy logic terms can be found in [28].

3. GENETIC ALGORITHM

Genetic algorithms are stochastic optimization techniques based on an analogy to the mechanics of biological genetics and imitate the Darwinian survival-of-the-fittest phenomenon. Each individual of a population is a potential FLC. In this research, FLCs are *encoded* into a genotype before applying four evolutionary operations: *reproduction, mutation, evaluation* and *natural selection,* and finally *decoded*.

3.1 GA Applied to FLC Generation

The key part of a FLC is the Fuzzy Inference System, and it is composed of fuzzy sets and fuzzy rules [28]. One can use some parameters to describe both of them and encode these parameters into a chromosome. There are many ways to encode an FLC into a GA [29]. In this paper, the encoding is based on the scheme proposed in [30], chosen for its simplicity since only few parameters are used to describe the FLC—namely, the number and center values of the input and output membership functions and the linguistic control rules.

3.1.1 Coding of the Membership Functions

In this paper the number of membership functions (MF) is assumed to be an odd number greater than unity and the MFs are distributed symmetrically around a center of zero. Only triangular fuzzy membership functions are considered for the sake of coding simplicity and without loss of generality. The center value is given by:

$$\phi_{MF} = \mathrm{sgn}(i).c_m |i|^{P_m} \text{ where } i = -\frac{n+1}{2},\cdots,0,\cdots,\frac{n+1}{2} \qquad (2)$$

Where n and P_m are the number of MFs and the exponent of the power function and *sgn(•) denotes the SIGN function:*

$$\mathrm{sgn}(x) = \begin{cases} 1 \ \ if\ x \geq 0 \\ -1 if\ x < 0 \end{cases} \qquad (3)$$

Finally, the characteristic function gives the values of c_m:

$$c_m = \left(\frac{n+1}{2}\right)^{-P_m} \qquad (4)$$

3.1.2 Coding of the Fuzzy Rule Base

A FLC is constructed using the following linguistic rules:

$$R_1 = if\ \ x_1 \text{ is } X_{1i} \text{ and } x_2 \text{ is } X_{2i} \text{ then } u \text{ is } U_i,\ i=1, ..,n \qquad (5)$$

where x_1, x_2 and u are input and output variables in the controller; these are described as state error, change-of-state error and a control input, in this paper.

The rules are coded using a combination of a seed line and seed points, which are positioned into a phase plane containing all the possible fuzzy rules one can fire. The locations of the seed points are determined as follow:

$$x_{si} = L.\sin c_s |i|^{P_s}$$
$$x_{si} = L.\tan(\phi_s) \quad (6)$$
$$i = -\frac{n_o+1}{2},\cdots,0,\cdots,\frac{n_o+1}{2}$$

where ϕ_s and P_s are seed angles and are defined as characteristic parameters for the fuzzy linguistic rules. Also C_s is obtained the same way as C_p using the number of MFs on the output (n_o).

L limits the locations of the seed points. For more details on the coding, see [30].

In **Figure 1**, the relationship between input and output in the controller can be represented as a function of these design parameters as follows.

$$u(t) = f(x_1(t), x_2(t)|\Psi) \quad (7)$$

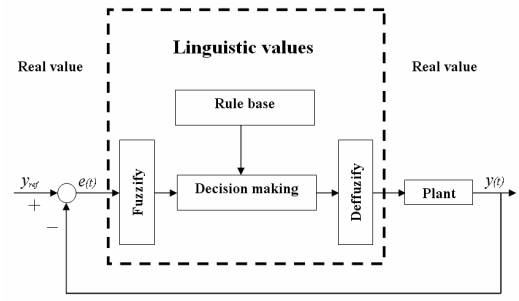

Figure 1: Control Block Diagram

Where $f(\bullet|\Psi)$ denotes a fuzzy function constructed with the design parameter vector Ψ. The parameters of Ψ are coded into a chromosome using binary coding.

3.1.3 Genotype of FLC

The characteristic parameters of the FLC are encoded into a chromosome as shown in Figure 2. The GA will evaluate this chromosome using the evaluation function to find the best individual and the FLC defined by it. More details are given in [30].

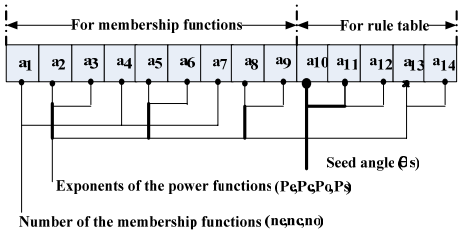

Figure 2: FLC's Parameters Encoded into a Chromosome

To perform the GA, MatLab was used in conjunction with the GAOT toolbox, which is open-source code provided by Houck et al. [31].

3.2 Evaluation Function

The evaluation function is called by the GA to compute the fitness of a set of parameters. The parameters are passed to the evaluation function, which processes them and returns a value corresponding to how well the controller defined by those parameters performed the task at hand. In this paper, the evaluation function returns the root mean square error between the desired and actual (computed) positions of the pendulum. This function first extracts the relevant parameters from the chromosome passed in. A SIMULINK model is then called, from which a record of the error in the pole angle throughout the duration of the simulation is obtained. The square of the error is multiplied by a time weight and the sum of this time-weighted square error is inverted to give a fitness value. If the pole angle should at any stage of the simulation saturate—i.e., reach ±90°, then the simulation is stopped immediately so that time is not wasted modeling controllers that fail to balance the pole. The MatLab source code for this evaluation function can be found in [33].

4. DOUBLE-LEVEL MODULAR FLC

FLCs designed automatically by GAs or other techniques generally exhibit a drop in performances with an increase in the working zone. The working zone (also called the universe of discourse) is defined as the interval between the minimum and maximum values in which the control actions occur. This drop in efficiency is due to the GA's learning paradigm, which generally selects the desired solution by evaluating the fitness function. A fitness function typically measures how closely a FLC's output fits the data in working zone used in the training set, which is generally smaller than the overall working zone (smaller interval and/or fewer values or even one single value). The difficulty this causes will be explained further below.

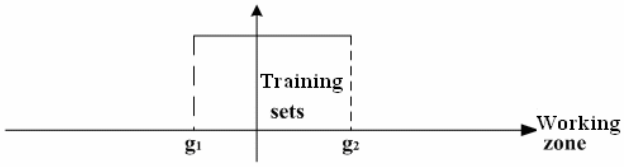

Figure 3: Training Set and the Working Zone

In **Figure 3**, g_1 and g_2 denote the minimum and maximum values of the FLC's working zone, respectively. The horizontal axis is the whole working zone. For the rest of the paper, the interval G (g1, g2) will refer to the working zone of the FLC and ts will refer to the training sets. G and ts are linked by the following expression:

$$ts \in G \ \ or \ \ ts = G \quad (8)$$

There is no established formula that can describe explicitly this link. But it is important to note that the performance quality depends on the selection of ts, since it determines the fitness function evaluation.

When the *ts* working zone is smaller than *G*, the near-optimal FLC may not work properly for the whole *G*, as shown in **Figure 4**. But increasing the *ts* working zone would not only increase learning time, but also causes the GA to get stuck in local optima because of larger/more complex search space, yielding an FLC that is not optimum, and therefore cannot solve the problem either efficiently or effectively.

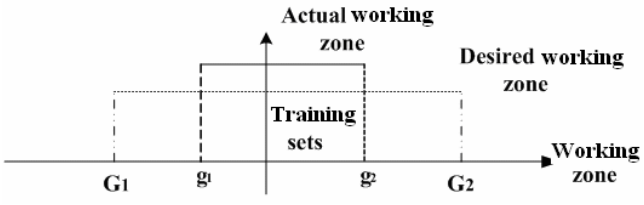

Figure 4: FLC's Working Zone

One way to solve this problem is to divide *G* into smaller working zones, choose a *ts* from each (can be a single value), and construct an FLC that works well for each. A method to integrate all FLCs is then needed so that the combination covers the entire working zone *G*. However, how to divide *G* properly so that each smaller working zone can have the right (producing optimum FLC) *ts* is still an open problem.

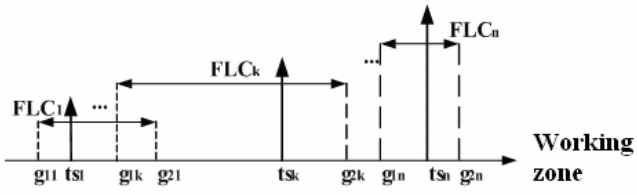

Figure 5: Working Zone Composed of Many Smaller Working Zones

Another similar way is to randomly select values from *G* and use them as *ts,* and then to identify an appropriate working zone for each *ts* by testing. With this method, the working zone of the system is controlled by several FLCs. This approach will be referred to as the *switch* mode in this paper, in which the concatenation of the FLCs' smaller working zones gives the total working zone of the system. In the switch mode approach, the control switches to a different FLC depending on where the value of the external input is located in *G*. Figure 5 shows an example in which all the FLCs have the same external input, but as their working zones are different, a switch chooses the FLC to use to control the target (FLC_2 output in **Figure 6**).

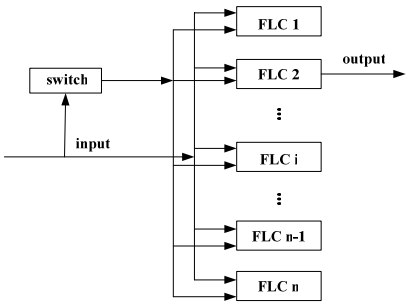

Figure 6: Switch Mode Approach

However, the switch mode approach can still fail quite frequently. One reason for this is that a smoother transition from one FLC to another is needed to control the dynamic systems successfully. An improvement of the switch mode approach is to use a linear combination of the outputs of the FLCs to control the system; such an approach has generated some good results, as described in [26].

Both approaches described above are difficult to realize properly, since the working zone must be divided into several smaller ones, and the learning by GA must be done for each, which is time consuming. The question would be: is it possible to reduce the number of smaller working zones and/or choose a finite and limited number of values from the system's working zone as *ts* in order to get an FLC that works well across its entire working zone? To achieve this, a double-level modular FLC (DLMFLC) is proposed in this paper.

A DLMFLC is a further improvement of the method presented in [26]. The basic idea is to use a second-level FLC to adjust the contribution of the first-level FLCs to the output of the integrated controller. The first-level FLCs would be the modules of the control system. These FLCs are called modular FLCs; they can work in their smaller working zones but not necessarily well in the whole system's work zone. Having several FLCs, how can we integrate them together into a single FLC? How can this second level be constructed? For each input that falls into the working zone of a first-level FLC, the output of this particular FLC module should contribute more to the output of the overall controller. However, this does not necessarily mean that other FLC modules should not contribute at all to the output of the overall controller. They will simply contribute to a lesser extent. The key here is to decide proper 'weights' of all FLC modules so that their overall contributions can control the system successfully across the entire working zone.

The method in [26] utilizes a linear combination of the outputs of the FLC modules, in which the 'weights' of the FLC modules are fixed for all inputs. In this paper, a second-level FLC is used to integrate the modular FLCs so that the 'weights' are adjusted by the second-level FLC according to different inputs. The basic scheme is presented in **Figure 7**.

The existing FLCs constitute the first-level FLCs, which are FLC_1, FLC_2, ..., FLC_n in **Figure 7**. A second-level FLC, which resides above the first-level FLCs (**Figure 7**), is used to integrate outputs of the first-level FLCs.

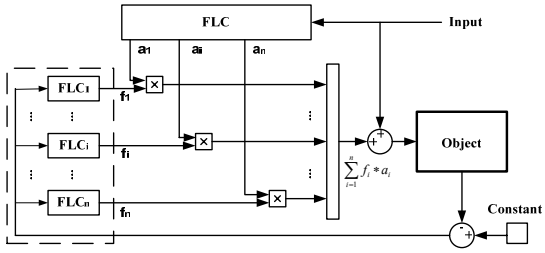

Figure 7: Diagram of Modular Double-Level FLC System

The second level FLC uses the input value of the system to provide the first level FLCs the fuzzy values *f*, where,

$$f_i = a_i, 0 \le a_i \le 1 \qquad (9)$$

Furthermore, every first-level FLC has an output, too. It is

$$out_i = f(input) \tag{10}$$

The output of the FLC to control the object is then given by:

$$OUT = \sum_{i=1}^{n} f_i \times out_i \tag{11}$$

5. Case Study

In order to test and validate the control approach proposed in Section 4, the Inverted Pendulum system (IPS) control problem is used, since it is considered a typical testbed for control experiments.

5.1 Inverted Pendulum System (IPS)

As we chose the IPS as our testbed, it is necessary to introduce some basic variables and components.

The variables shown in **Figure 8** and 9 are as follows:

- M, the mass of the cart,
- m, the mass of the pole,
- l, the distance from the pivot to the center of mass of the pole,
- I, the moment of inertia of the pole around the pivot,
- x, the cart's position,
- θ, the angle the center of the pole makes with the vertical,
- F, the horizontal control force applied to the cart,
- g, gravity.

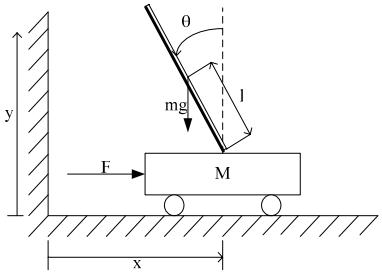

Figure 8: IPS Model.

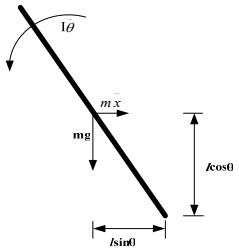

Figure 9: IPS--Pole in Isolation

Taking the second derivative of the position of the pole's center of gravity, one obtains:

$$\ddot{x}_G = \ddot{x} - l\ddot{\theta}\cos\theta + l\dot{\theta}^2 \sin\theta \tag{12}$$

Newton's second law of motion yields:

$$F = M\ddot{x} + m\ddot{x}_G \tag{13}$$

Combining equations 12 and 13 yields:

$$F = \ddot{x}(M+m) + ml\dot{\theta}^2 \sin\theta - ml\ddot{\theta}\cos\theta \tag{14}$$

Considering that, for a rigid body, the sum of the moments about a fixed point, P, is equal to the moment of inertia of the body about P multiplied by the angular acceleration plus the product of the mass of the body, its linear acceleration and the perpendicular distance between point P and the vector representing the acceleration yields:

$$M_p = I_p\alpha + m\vec{a}d \tag{15}$$

Recombination, insertion and rearrangement of these equations yields (for a uniform rod of mass m and length L=2×l):

$$\ddot{x} = \frac{\frac{4}{3}F + mg\cos\theta\sin\theta - \frac{4}{3}l\dot{\theta}^2\sin\theta}{\frac{4}{3}(M+m) - m\cos^2\theta} \tag{16}$$

$$\ddot{\theta} = \frac{(M+m)g\sin\theta + \cos\theta F - ml\dot{\theta}^2\cos\theta\sin\theta}{\frac{4}{3}(M+m)l - ml\cos^2\theta} \tag{17}$$

More details on the inverted-pendulum can be found in [33].

5.2 SIMULINK model of the IPS

Using the equations shown in Section 5.1, the SIMULINK model shown in Figure 10 is developed. In the model, the input is the control action. The angle and angular velocity from the previous time step are combined to compute the cart's acceleration and the pole's angular acceleration, respectively. The speed and position of the cart are gotten by integrating the angle and the angular velocity. The pole's angle is also limited to remaining in the range of [-pi/2, pi/2]. There are four variables considered as outputs: pole's angle and angular velocity along with the cart's position and velocity.

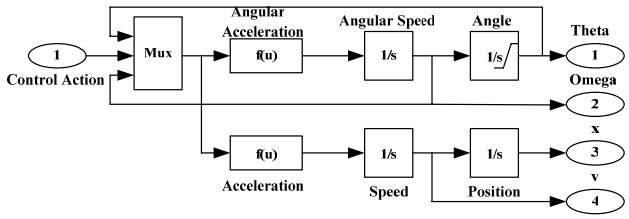

Figure 10: SIMULINK model of the IPS

For the sake of uniformity of the parameters during the test, the values of three variables are fixed:

- M=1kg
- m=0.1kg
- l=0.5m

Gravity is fixed at g = 9.8m/s^2.

5.3 Construction of the first-level FLCs

The number of membership functions (MFs) is arbitrarily restricted to the integers from 3 to 9, inclusively, as most FLCs

are within this range. The number of MFs is limited to odd integers because of the way the FLC is encoded in the GA. In the tests reported in this paper, we selected the number of MFs to be 5.

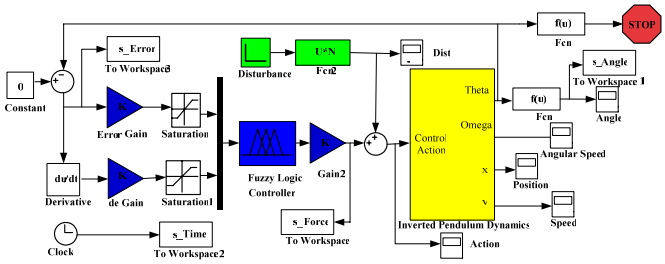

Figure 11: First-level FLC's SIMULINK Model

5.3.1 Running the GA

The parameters for running the GA to evolve the FLCs in both second and first levels are:

- Population size: 50
- Selection mechanism: Stoc.Rem (as in Stochastic Remainder Sampling, a MatLab script added to the GA tool box of MatLab [33])
- Crossover: single-point crossover at 0.5
- Mutation : single-bit, at 0.05/bit

Three different disturbances—250N, 500N and 1000N—were used to create the training sets for evolving the first-level FLCs. Then the working zone for each evolved FLC module was obtained by testing the FLC against different external disturbances ranging from 0N to 1200N, with a 1N increment.

5.4 Second-level FLC

Figure 12 shows the basic scheme of the DLMFLC. The new blocks on the left are the modular FLC, also called the second-level FLC. There are some differences in the process of getting the second-level FLC, as compared to finding the first-level ones. The disturbance is different—it lasts 0.01s—but the same three disturbances, which are 250N, 500N and 1000N, are used, and the simulation starts from 0s, 2s and 4s, respectively. The FLC's structure is also different; while the first-level FLC's inputs are error and change ratio of error and the output is a force to reach (control) the target (the pole of IPS), the second-level FLC is a one-input, three-output system. The input is the value of the disturbance and the output is the gain (a real value with no actual physical meaning) for every modular FLC (3 outputs).

The parameters for encoding the second-level FLC into the GA are therefore also different. The number of MFs is set to 3 and the output scaling is 0-1 (gain). The parameters for running the GA are the same as those used to evolve the modular FLCs.

In order to test the repeatability of the experiment, the whole process was repeated 5 times. A different modular FLC was selected from each of the 5 experiments. Furthermore, an additional challenge was imposed on the second-level FLC: the sum of the outputs of the overall system should not exceed 500N, which is a way of reproducing the torque limit on a motor that would stabilize the inverted pendulum; however, the modular FLCs' outputs have no such limitations (i.e., the limitation is imposed by the gains evolved for the second-level FLC).

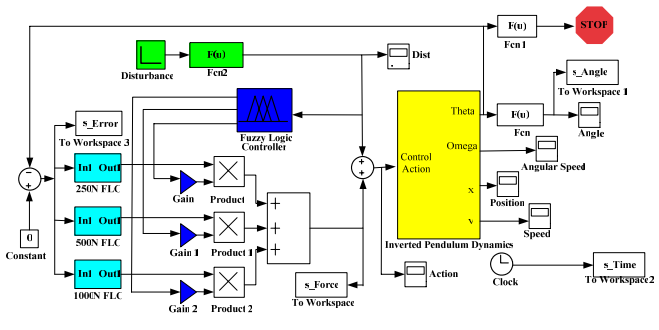

Figure 12: SIMULINK Model of a Double-Level FLC

5.5 Results

In order to test the efficiency of using a second-level FLC, it is compared to the results obtained by training the FLC with the first level only (gains/parameters of the first level FLC are decided by GA).

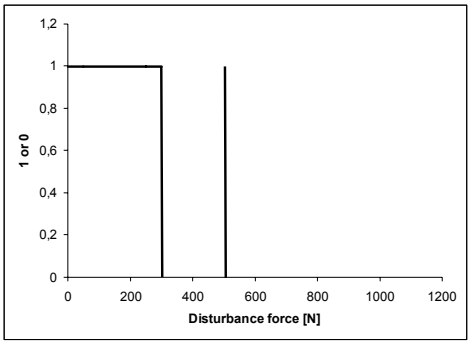

Figure 13: 250N FLC's Working Zones

Figure 14 and 15 show the working zones for each FLC. A reading of '1' on the y-axis means that for the disturbance force shown on the x-axis, the FLC can control and stabilize the IPS successfully. A reading of '0' indicates otherwise.

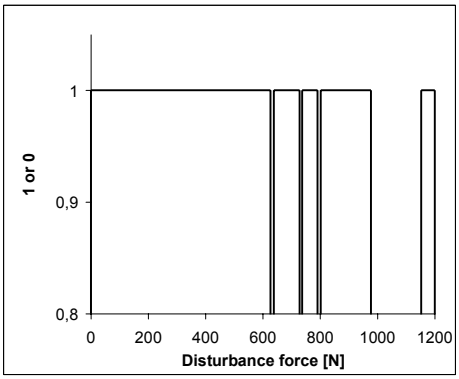

Figure 14: 500N FLC's Working Zones

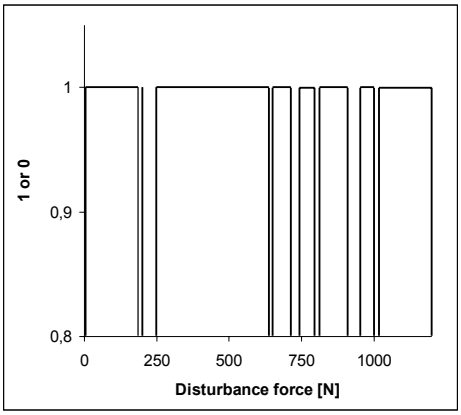

Figure 15: 1000N FLC's Working Zones

It is noteworthy that the FLC obtained for the 250N disturbance has a very limited control range, and examination of the ones obtained for 500N and 1000N shows that there are many intervals where the FLC cannot control the IPS. Hence, using a first-level FLC alone does not give satisfactory results when testing the controller with different values than the ones used for the training.

After the second-level FLC is linked to the first-level ones, comprising the DLMFLC, one can see in Figure 16 that the controller is able to work properly for the whole range of disturbance forces.

Figure 17 shows the necessary time to stabilize the inverted pendulum using a maximal force of 500N, while disturbing the system with different forces; one can conclude that the response time is satisfactory in all cases studied.

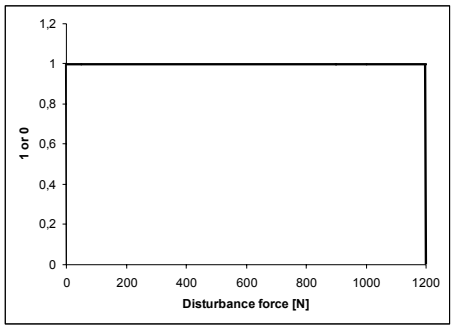

Figure 16: Double Level FLC's Working Zone

6. CONCLUSION

In order to get a FLC that performs efficiently across the whole working zone of a controlled system using a genetic algorithm (GA), a double-level modular FLC (DLMFLC) was proposed and developed successfully in this paper. The approach was tested with an IPS controller design problem.

The simulation results show that the DLMFLC obtained works well across the entire desired working zone and needs only small training sets during the search process of the GA.

More issues must still to be investigated for the DLMFLC. For example, one should investigate the relationship between first-level FLCs and the second-level FLC, so as to guide how to choose the training sets for the first-level FLCs that make it easier for the second-level FLC to evolve a better overall controller—i.e., DLMFLC.

In the test presented in this paper, the second-level FLC is a SIMO system. In this particular case study of IPS controller design, external disturbance is the only input for the second-level FLC. But in practice, several external parameters can influence a system. Should one add more variables as inputs to the second-level FLC, and how? All these questions merit further study to make the DLMFLC usable in more general applications.

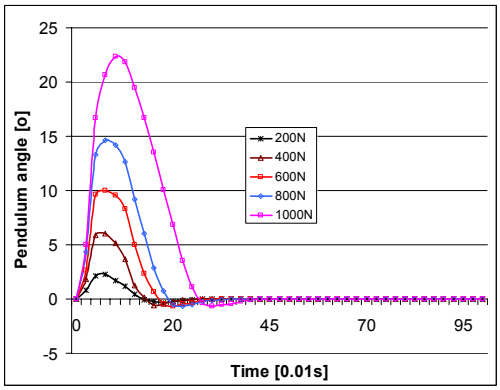

Figure 17 : Oscillation Angle of the IPS vs. Time.

7. ACKNOWLEDGMENTS

Financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC) under Post-Doctoral Grant BP-328508-2006 is gratefully acknowledged.

REFERENCES

[1] Verbruggen, H.B. and Bruijn, P.M., Fuzzy control and conventional control: What is (And Can Be) the Real Contribution of Fuzzy Systems? Fu*zzy Sets Systems, Vol. 90,* 151–160, 1997.

[2] Kowalska, T.O., Szabat, K. and Jaszczak, K., The Influence of Parameters and Structure of PI-Type Fuzzy-Logic Controller on DC Drive System Dynamics, *Fuzzy Sets and Sysems,* Vol. 131, 251–264, 2002.

[3] Ahmed, M.S., Bhatti, U.L., Al-Sunni, F.M. and El-Shafei, M., Design of a Fuzzy Servo-Controller, *Fuzzy Sets and Systems,* vol. 124, 231–247, 2001.

[4] Zilouchian, A., Juliano, M., Healy, T. and Davis, J., Design of Fuzzy Logic Controller for a Jet Engine Fuel System, *Control and Engineering Practices,* Vol. 8, 873–883, 2000.

[5] Zadeh, L.A., Fuzzy sets, *Information Control*, Vol. 8, pp. 339–353, 1965.

[6] Liu, B-D., Design and Implementation of the Tree-Based Fuzzy Logic Controller, *IEEE Transactions on Systems, Man, and Cybernetics*, Part B: Cybernetics., Vol.27, No.3, 475-487, 1997.

[7] Zhiqiang, G., A Stable Self-Tuning Fuzzy Logic Control System for Industrial Temperature Regulation, *IEEE*

Transactions on Industry Applications.Vol.38, No.2, 414-424, 2002.

[8] Shapiro, A.F., Fuzzy Logic in Insurance*, Insurance: Mathematics and Economics*, Vol.35, No.2 , 399-424, 2004.

[9] Hayward, G. and Davidson, V., Fuzzy Logic Applications, *Analyst*, Vol.128, 1304-1306, 2003.

[10] Peri, V.M. and Simon, D., Fuzzy Logic Control for an Autonomous Robot, *North American Fuzzy Information Processing Society, NAFIPS 2005 Annual Meeting*, 337-342, *2005.*

[11] Castellano, G., Attolico, G. and Distante, A., Automatic Generation of Fuzzy Rules for Reactive Robot Controllers, *Robotics and Autonomous Systems*, Vol. 22, 133–149, 1997.

[12] Higgins, C. M. and Goodman, R. M., Fuzzy Rule-Based Networks for Control, *IEEE Transactions on Fuzzy Systems*, Vol. 2, No. 1, 82–88, 1994.

[13] Wang, L. X. and Mendel, J. M., Generating Fuzzy Rules By Learning From Examples, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 22, No. 6, 1414–1427, 1992.

[14] Mohamed, A.M.O., Neuro-Fuzzy Logic Model for Evaluating Water Content of Sandy Soils, *Computer-Aided Civil and Infrastructure Engineer,* Vol.19, No.4,274-287, 2004..

[15] Horng, J.H., SCADA System of DC Motor with Implementation of Fuzzy Logic Controller on Neural Network, *Advances in Engineering Software*, Vol.33 Issue.6, pp:361-364, 2002.

[16] Van Cleave, D.W., Tuning of Proportional Plus Derivative Fuzzy Logic Controller using Neural Network, *Proceedings of the 33rd Southeastern Symposium on System Theory*, 365 -370, 2001.

[17] Nandi, A.K. and Pratihar, D.K, Automatic Design of Fuzzy Logic Controller using a Genetic Algorithm—To Predict Power Requirement and Surface Finish in Grinding, *Journal of Material Processing Technology,* Vol. 148, 288-300, 2004.

[18] Achiche, S., Baron, L. and Balazinski, M., Predictive Fuzzy Control of Paper Quality, *Annual Meeting of the North American Fuzzy Information Processing Society*, CD-ROM Version, 2006.

[19] Chiang, C.K., A Self-Learning Fuzzy Logic Controller Using Genetic Algorithms with Reinforcements, *IEEE Transactions on Fuzzy Systems,* Vol.5, No.3, 460-467, 1997.

[20] Chin, T.C., Genetic Algorithms for Learning the Rule Base of Fuzzy Logic Controller, *Fuzzy Sets and Systems*, Vol.97, No.1, 1-7, 1998.

[21] Arslan, A and Kaya, M., Determination of Fuzzy Logic Membership Functions using Genetic Algorithms: Application to Structure-Odor Modeling, *Fuzzy Sets and Systems*, Vol.118, No.2, 297-306, 2001.

[22] Li, R. and Zhang Y., Fuzzy Logic Controller Based on Genetic Algorithms. *Fuzzy Sets and Systems*, Vol.83, No.1, 1-10, 1996.

[23] Liu, B.D., Design of Adaptive Fuzzy Logic Controller Based on Linguistic-Hedge Concepts and Genetic Algorithms, I*EEE Transactions on Systems, Man and Cybernetics,* Part B*,* Vol.31, No.1, 32 -53, 2001.

[24] Chung, H.Y. and Chiang, C.K., A Self-Learning And Tuning Fuzzy Logic Controller Based on Genetic Algorithms and Reinforcements, *International Journal of Intelligent Systems*, Vol.12, Issue.9, 673-694, 1997.

[25] Balazinski, M., Achiche, S.and Baron, L. Influence of Optimization and Selection Criteria on Genetically-Generated Fuzzy Knowledge Bases, *2nd International Conference on Advanced Manufacturing Technology*, 159-164, 2000.

[26] Yuhui, S., Eberhart, R. and Yaobln C., Evolutionary Modular Fuzzy System, *IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence*, 387-391, 1998.

[27] Zadeh, L.A., Outline of New Approach to the Analysis of Complex Systems and Decisions Processes, *IEEE Transactions of Systems, Man and Cybernetics*, Vol.3, 28-44, 1973.

[28] Klir, G.J., Zadeh, L. A., Yuan, B., Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems, World Scientific Series, 1996.

[29] Baron, L., Achiche, S., Balazinski, M., Fuzzy Decision Support System Knowledge Base Generation Using a Genetic Algorithm. *International Journal of Approximate Reasoning,* Vol.28, No. 2-3, 125-148, 2001.

[30] Young J.P., Hyung, S.C. and Dong H.Ch., Genetic Algorithm-Based Optimization Of Fuzzy Logic Controller Using Characteristic Parameters, IEEE International Conference on Evolutionary Computation, Vol.2, 831-836, 1995.

[31] Houck, C.R., Joines, J. and Kay, K., A Genetic Algorithm for Function Optimization: A Matlab implementation, *ACM Transactions on Mathematical Software*, http://www.eos.ncsu.edu/eos/service/ie/research/kay_res/GAToolBox/gaot, 1996.

[32] Lee, M.A., Takagi, Integrating Design States of Fuzzy System using Genetic Algorithms, *Proceedings of the 2nd IEEE International Conference on Fuzzy Systems,* San Francisco, 612-617, 1993.

[33] Foran, J., Optimization of a Fuzzy Logic Controller Using Genetic Algorithms, *Master in Engineering Report*, 108p, 2002.