IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 7, Issue 10, October 2020

DOI 10.17148/IARJSET.2020.71014

Advanced Lecture Skill of Fuzzy Control in Python

Dong Hwa Kim[1]

EPCE, ASTU, Ethiopia[1]

Abstract: This paper aim is to provide advanced lecture skill of fuzzy control as intelligent control. Fuzzy set is a quite linguistic expression tool for good intelligent control. Currently, many are interested in deep learning or machining to recognize pattern and to classify figures, but it is not powerful as control tool in small size and lab-scale. However, fuzzy has a long history and many have been studying as linguistic expression tool for human being life. Therefore, it is an important to teach effectively and cooperate with Python because deep learning is using Python. This paper describes on how we can have a good lecture of fuzzy control for intelligent control.

Keywords: Fuzzy control, Lecture, Intelligent control, Python.

I. INTRODUCTION

The fuzzy set stared from 1965 as the linguistic expression of fuzzy is one of AI expressions and fuzzy control implemented by Mamdani 1974. Research and teaching about fuzzy control and industrial application were boomed till 1990s because neural network has winter season. The mathematical foundations of fuzzy logic rest in fuzzy set theory, which can be thought of as a generalization of classical set theory. A familiarity with the novel notions, notations, and operations of fuzzy sets is useful in studying fuzzy logic principles and applications. Fuzziness is a property of language. It is not a set uniquely. It all depends on what we mean by the word action. Words like function have many shades of meaning and can be used in many different ways. Their meaning and use may vary with different persons, circumstances, and purposes. On the other hand, fuzzy control primarily refers to the control of processes through fuzzy linguistic descriptions. Since 1974, when E.H. Mamdani and S. Assilian (Mamdan, 1974) demonstrated that fuzzy if/then rules could regulate a model steam engine, a great number of fuzzy control applications have been successfully developed. The list is very long and growing and includes cement kilns, subway trains, unmanned helicopters, autonomous mobile robots, process heat exchangers, and blast furnaces (Mamdani, 1977; Ostergaard, 1982; Yasunobu and Miyamoto, 1985; King and Karonis, 1988).
In the 1970s and early 1980s most applications were minicomputer-based, often found in the process industry in areas where automatic control was rather difficult to realize and hence left in the hands of human operators. The advent of fuzzy microprocessors, a growing number of fuzzy control applications have emerged in consumer electronics and home appliances such as hand-held cameras, vacuum cleaners, air conditioners, and washing machines (Hirota, 1993; Yamakawa, 1989; Schwartz, 1992; Terano et al., 1992).
As identification tool, in each fuzzy inference, there can be several types of membership functions, namely Gaussian membership functions with modified or fixed slopes, and triangular membership functions. According to the proposed auto-tuning algorithm, membership function can be easily adjusted. And also, according to the proposed rule, fuzzy modeling method of fuzzy control can be different for plant. Identification method also will be different by means of fuzzy inference and the premise identification is determined by the membership values of fuzzy variables.
In case of control, some of the most successful applications of fuzzy control have been in conjunction with conventional controllers such as the proportional integral derivative (PID) controller (Lee, 1990a, b). In fuzzy control we are concerned with two broad questions. How can we implement a control strategy as a fuzzy linguistic description and What are the crucial factors involved in fuzzy algorithmic synthesis and analysis?
Although fuzzy linguistic descriptions are a subject of wider interest than the replacement or enhancement of PID controllers, their application to control serves to illustrate some of the basic ideas we encountered in earlier season 1990s. However, currently, almost are interested in deep learning as recognition function of deep learning. As linguistic approach of AI and identification of model, fuzzy is still good AI tool. Fuzzy set, fuzzy relationship, and control technology using linguistic of fuzzy is useful in control system. Its combination technology to other AI tool such as neural network, deep learning, blockchain, and others is also possible and useful research area. Depending on teaching method and skill, some AI tools can be developed and winter season in University and industrial areas. Especially, we need use new programming language like Python and Pytham to extend our students future instead of Matlab. Matlab is basically commercial program and they are focusing on business. It means it is not useful as education. WE do not have any option to teach and learn in simulation and teach. However, now, there is a good alternative language like Python with huge community. Now, it is time to change for good education infrastructure of young generation. This paper deal with on how we can teach fuzzy control using Python through comparison fuzzy control system with the conventional PID control. Thai paper uses two types of Mamdani and Seguno as fuzzy inferences.

II. CONCEPTION OF PID CONTROL FOR TEACHING OF FUZZY CONTROL

A. The Basic Conception of PID control

Even though several control theories have been developed significantly, the proportional-integral-derivative (PID) controllers have been widely used owing to their simple structure, which can be easily understood and implemented for a wide range of process control, motor drives, flight control, and instrumentation. In industrial applications, more than 90% of all control loops are PID type. Owing to their popularity in the industrial world, over the past 50 years, several approaches for determining PID controller parameters have been developed for stable processes that are suitable for auto-tuning and adaptive control and for single input single output (SISO) systems.

To improve the performance of PID tuning for processes with changing dynamic properties, several tuning strategies have been suggested, such as automatic tuning PID [3], and adaptive PID [36]. As tuning methods based on the automatic measurement of the ultimate gain and period, various techniques, such as relay excitation feedback [3] and rule-based auto tuning [37], have been developed. These tuning approaches have recalibration features to cope with little a priori knowledge and significant changes in the process dynamics. However, the PID controller parameters must be computed using the classic tuning formulae and therefore these cannot provide good control performance in all situations.

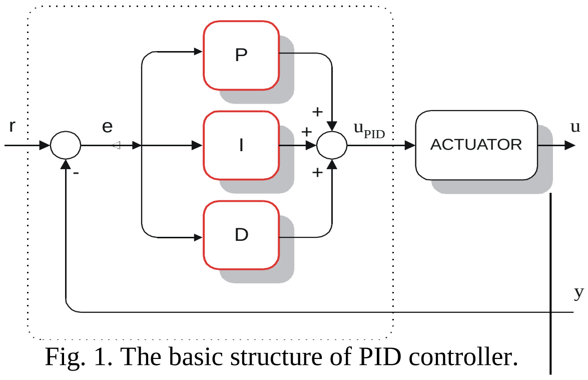


Fig. 1. The basic structure of PID controller.

$$u = K\left[e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right] \qquad (1)$$

To implement in Python, the previous paper [34] modified signal as

$$MV_k = MV_{bar} + K_p e_k + K_i \sum_{k=0}^{n} e_k(T_k - T_{k-1}) + \frac{e_k - e_{k-1}}{T_k - T_{k-1}} \qquad (2)$$

$$e_k = SP_k - PV_k, \qquad (3)$$

I published about theory, lecture skill, code by Python, and lecture procedure of the basic PID Controller in Journal IARJSET (Sept 2020). So, I will skip this content in this paper.

Consider the simple process system shown in Fig.1. The output of the controller u serves as input to the process. The error $e = r - y$ is actually smoothed and scaled before input to the controller. The most common controller in the process industry is the PID controller, where the control relation associated with equation (4) takes the form

$$u(t) = K_P e(t) + K_P K_I \int_{t=0}^{t} e(t)dt + K_P K_D \frac{de(t)}{dt} + u(0) \qquad (4)$$

where K_P is the controller gain representing a proportionality constant between error and controller output (dimensionless), K_I is the reset constant relating the rate to the error in units of [%/(%-sec)], K_D is the rate constant (or derivative gain constant) in units of [(%-sec)/%], and $u(0)$ is the controller output at $t = 0$ (when a deviation from setpoint starts). The first term in equation (4) is called the proportional term, and if it was the only term in the equation it would represent a mode of control where the output of the controller $u(t)$ is changed in proportion to the error $e(t)$, which is the percent deviation from the setpoint. The second term is called the integral term and represents a mode of control where the present controller output depends on the history of errors from when observations started at $t = 0$. The amount of corrective action due to integral mode is directly proportional to the length of time that the error has existed. The reset constant K_I expresses the scaling between error and controller output. A large value of means that a small error produces a large rate of change of u and vice versa. If this term alone was used in equation (4), in addition to the constant $u(0)$, then we would have a mode of control called integral mode. The third term in equation (4) represents the derivative mode of control. This mode provides that the controller output depends on the rate of change of error. Derivative mode tends to minimize oscillation of the system and prevent overshooting. Since derivative control is based solely on the rate of change of error, the controlled variable can stabilize at a value different from r, a condition termed "offset". In pure derivative mode the output depends upon the rate at which the error is changed and not on the value of the error. Integral control is used to address situations when permanent offset of slow returns to desired values cannot be tolerated. The combination of these three modes is called proportional integral derivative or (PID) control. PID is a powerful composite mode of control that has been used for virtually any linear process condition.

The process of adjusting the coefficients of each mode of control in equation (4) is called tuning. There are several methods for determining the optimum value of these gains such as frequency response methods and the Ziegler-Nichos method (Johnson, 1977). Fuzzy and neural approaches with adaptive characteristics have also been used for PID tuning and more generally for emulation and enhancing PID controllers (Matia et al., 1992; Maeda and Murakami, 1992; Shoureshi and Rahmani, 1992; He et al., 1993).

III. DESCRIPTION OF TEACHING OF FUZZY CONTROL FOR PYTHON

A. Overview

The core or a fuzzy controller is a linguistic description prescribing appropriate action for a given state. As we saw in introduction, fuzzy linguistic descriptions involve associations of fuzzy variables and procedures for inferencing. Whereas in a conventional PID controller what is modelled the physical system or process being controlled, in fuzzy controller the aim is to incorporate expert human knowledge in the control algorithm. In this sense, a fuzzy controller may be viewed as a real-time expert system. That is, a model of the thinking processes an expert might go through in the course of manipulating the process.

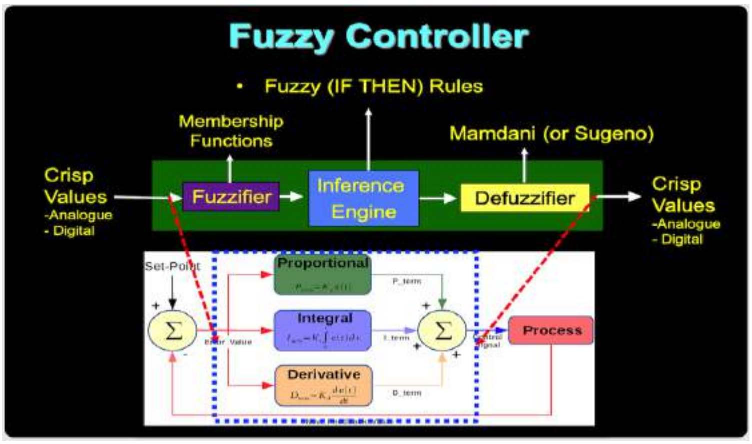


Fig. 2. The basic structure of fuzzy controller (from lecture ppt).

The basic structure of a fuzzy controller is outlined in Fig. 2. The fact that measuring devices give crisp measurements and that actuators require crisp inputs calls for two additional considerations when linguistic description is employed for control purposes: fuzzifying the input of the controller and defuzzifying its output. Fuzzification can be achieved through a fuzzifier kernel as we saw in Fig. 2, and defuzzification can be achieved through special procedures that select a crisp value representative of the fuzzy output. Many controllers, however, use directly crisp inputs. Fig. 3 shows that in addition to a set of if/then rules, a fuzzy controller has an input interface and an output interface handling fuzzification and defuzzification as well as various signal manipulations such as normalization, scaling, smoothing, and quantization. Scaling maps the range of values of the controlled variables into predefined universes of discourse, and quantization procedures assist in the mapping when discrete membership functions are used (Larkin, 1985: Efstathiou, 1987; Yager and Filev, 1994). Fuzzy controllers operate indiscrete time intervals. The rules are evaluated at regular intervals in the same way as in conventional digital control, with several rules being executed together (in parallel) within the same time interval. This parallel feature makes it possible to develop highly dispersed fuzzy algorithms. The choice of sampling interval depends on the process being controlled and is usually selected so settling time (King and Mamdani, 1997).

B. if/then Rules and Inference

Often, but not always, LHS and RHS variables are scaled to the same universe of discourse and possess fuzzy values that have the same form. Scaling to a common universe of discourse with a common set of values for all variables may offer considerable savings in memory and speed as far as the computer implementation of a fuzzy algorithm is concerned. In

IARJSET

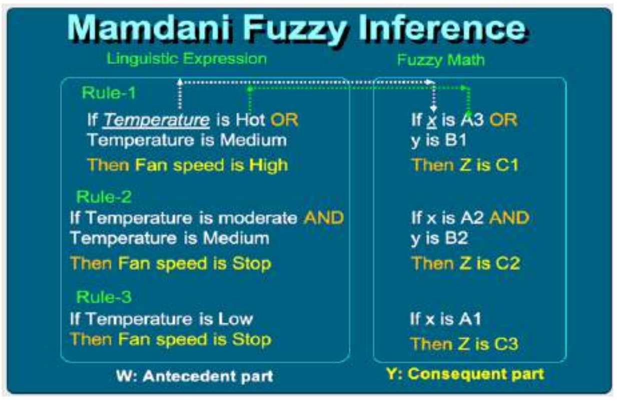
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 7, Issue 10, October 2020

DOI 10.17148/IARJSET.2020.71014

addition, it may be helpful in analyzing the behavior of the controller itself, as we will see later in this chapter. With the advent of fuzzy microprocessors and fuzzy development shells, it is no longer necessary for a user to do scaling because it is done by the system automatically. Nonetheless, scaling help to simplify analysis an exp, consider the fuzzy values of the variables $error$, $\Delta error$, and Δu shown in Fig. 3 in connection with a fuzzy controller that emulates the derivative mode of a conventional controller (Sugeno, 1985; Mizumoto, 1988). The common fuzzy values are as follows:



Mamdani Fuzzy Inference

Linguistic Expression	Fuzzy Math
Rule-1 If Temperature is Hot OR Temperature is Medium Then Fan speed is High	If x is A3 OR y is B1 Then Z is C1
Rule-2 If Temperature is moderate AND Temperature is Medium Then Fan speed is Stop	If x is A2 AND y is B2 Then Z is C2
Rule-3 If Temperature is Low Then Fan speed is Stop	If x is A1 Then Z is C3
W: Antecedent part	Y: Consequent part

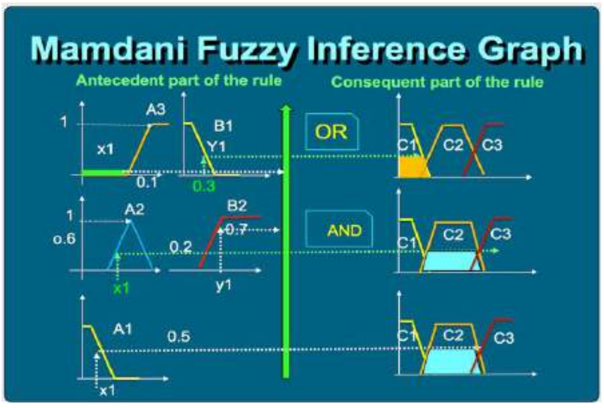

Mamdani Fuzzy Inference Graph
Antecedent part of the rule
Consequent part of the rule
1
A3
x1
0.1
B1
Y1
0.3
OR
C1
C2
C3
1
0.6
A2
x1
B2
0.7
0.2
y1
AND
C1
C2
C3
A1
0.5
x1
C1
C2
C3

Fig. 3. Mandani If/Then rule (from lecture ppt).

All variables share the same universe of discourse ranging as shown in Fig. 3. In computer implementations, fuzzy values are usually quantized and stored in memory in the form of a look-up table as with every row in the table representing a quantized fuzzy value. The fuzzy algorithm of a controller that emulates a derivative mode is comprised of the above $if\ /\ then$ rules. When two LHS and one RHS variables are used, the algorithm can be visualized in the form of a table as shown in Table 1. Such an arrangement is sometimes called a "fuzzy associative memory (FAM) matrix." Blank items in the table indicate that there is no rule present for the particular combination of LHS variables. Obviously for algorithms with more than two LHS variables a tabular representation requires additional dimensions.

Fuzzy control algorithms are evaluated using *generalized modus ponens* (GMP). We recall that GMP is a data-driven inferencing procedure that analytically involves the composition of fuzzy relations, usually *max-min composition*. We also saw that max-min composition under a given implication operator affects the RHS in a specific manner—for example, by clipping (when Mamdani min, ϕ_c, is used) or scaling (when Lasen product, ϕ_p, is used). In general, GMP can be thought of as a transformation of the RHS by a degree commensurate with degree of fulfillment (DOF) of the rule and in a manner dictated by the implication operator chosen. In this chapter, instead of explicitly using composition operations, we will mostly focus on such transformations as is often done, for the sake of convenience, in many fuzzy control applications. As far as the entire algorithm is concerned, the connective *ELSE* is analytically modeled as either $OR(\vee)$ or $AND(\wedge)$, again depending on the implication operator used for the individual if/then rules. For example, when the Mamdani min implication is used, the connective *ELSE* is interpreted as *OR*.

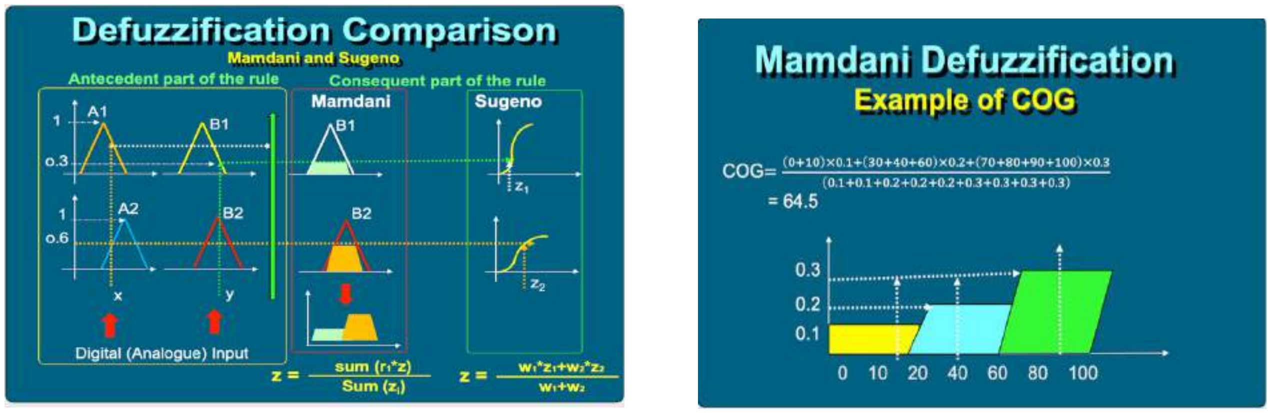
TABLE 1 EXAMPLE OF FUZZY VALUES

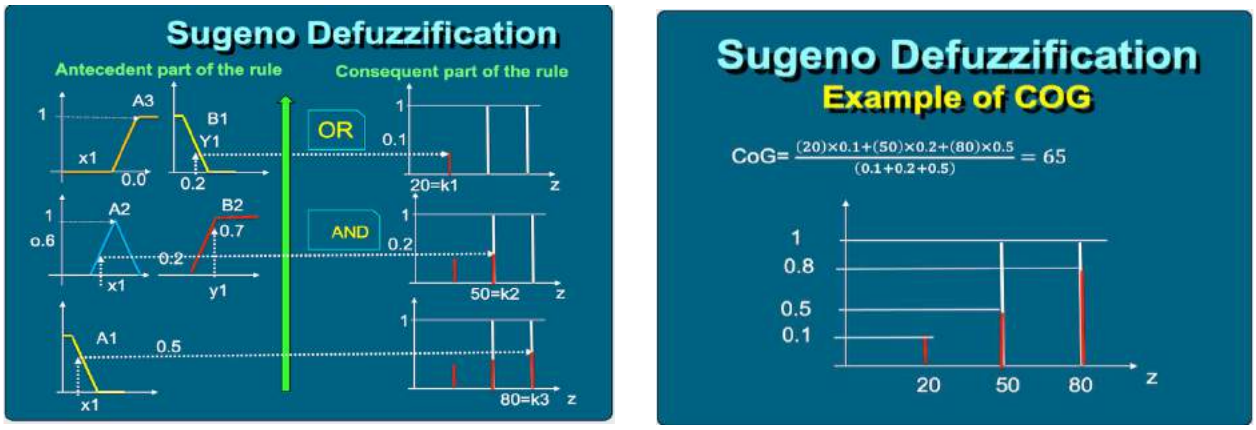
	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
NB	1	0.67	0.33	0	0	0	0	0	0	0	0	0	0
NM	0.33	0.67	1	0.67	0.33	0	0	0	0	0	0	0	0
NS	0	0	0.33	0.67	1	0.67	0.33	0	0	0	0	0	0
ZE	0	0	0	0	0.33	0.67	1	0.67	0.33	0	0	0	0
PS	0	0	0	0	0	0	0.33	0.67	1	0.67	0.33	0	0
PM	0	0	0	0	0	0	0	0	0.33	0.67	1	0.67	0.33
PB	0	0	0	0	0	0	0	0	0	0	0.33	0.67	1

C. Defuzzification for Design of Fuzzy Controller

After the input of crisp signal transformed by fuzzification to the controller, fuzzy controller must change from fuzzification to crisp signal again. We call this process as defuzzification. Mamdani style and Sugeno style has been suggested but over the years several defuzzification techniques have been suggested (Terano et al., 1992; Pedrycz, 1993; Yager and Filev, 1994). The choice of defuzzification method may have a significant impact on the speed and accuracy of a fuzzy controller. For the final crisp signal, the most frequently used ones are the *centroid* or *center of area* (COA), the *center of sums* (COS), and *mean of maxima* (MOM).

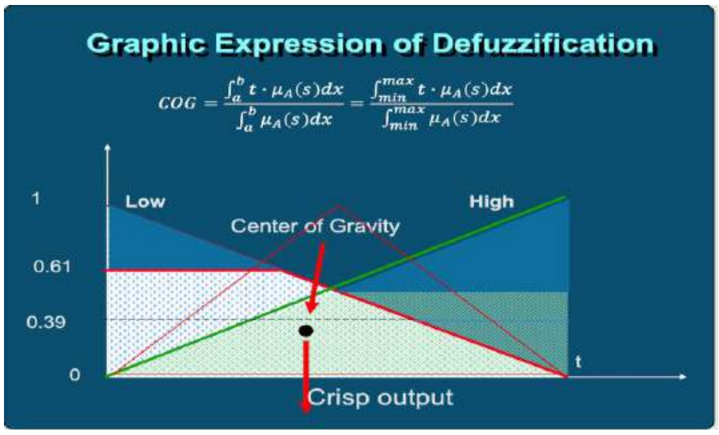
IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Vol. 7, Issue 10, October 2020
DOI 10.17148/IARJSET.2020.71014


Defuzzification Comparison
Mamdani and Sugeno
Antecedent part of the rule
Consequent part of the rule
Mamdani
Sugeno
Digital (Analogue) Input
z = sum (r_i*z) / Sum (z_i)
z = w1*z1+w2*z2 / w1+w2
Mamdani Defuzzification
Example of COG
COG= (0+10)×0.1+(30+40+60)×0.2+(70+80+90+100)×0.3 / (0.1+0.1+0.2+0.2+0.2+0.3+0.3+0.3+0.3)
= 64.5
Fig. 4. Defuzzification of Mamdani (from lecture ppt)


Sugeno Defuzzification
Antecedent part of the rule
Consequent part of the rule
OR
AND
20=k1
50=k2
80=k3
Sugeno Defuzzification
Example of COG
CoG= (20)×0.1+(50)×0.2+(80)×0.5 / (0.1+0.2+0.5) = 65
Fig. 5. Defuzzification of Sugeno (from lecture ppt)

D. Center of Area (COA) Defuzzification

In COA defuzzification the crisp value is taken to be the geometrical center of the output fuzzy value. Output signal is formed by taking the union of all the contributions of rules. Finally, the defuzzified output is defined as shown Fig. 6 and example is in Fig. 4 and Fig. 5.


Graphic Expression of Defuzzification
COG = ∫_a^b t·μ_A(s)dx / ∫_a^b μ_A(s)dx = ∫_min^max t·μ_A(s)dx / ∫_min^max μ_A(s)dx
Low
High
Center of Gravity
Crisp output
Fig. 6. Graphic expression of Center of Area (COA) Defuzzification (from lecture ppt)

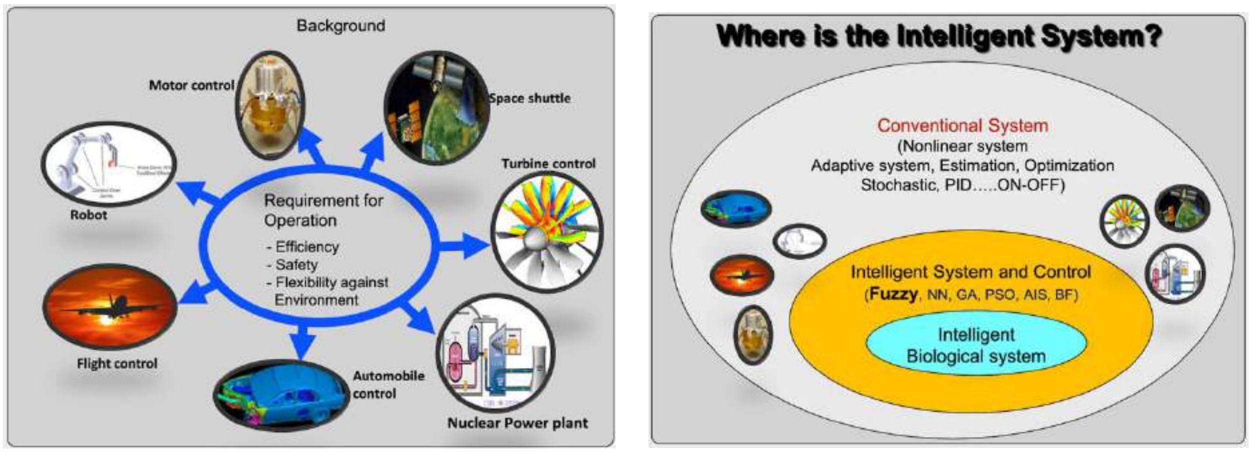
IV. LECTURE SKILL FOR FUZZY IN PYTHON

A. Lecture on The Position of Fuzzy Control as Intelligent Control

The core of a fuzzy controller is a linguistic description. That is fuzzy controller has the role of intelligent function. Therefore, lecture must introduce all control areas as introduction for student as shown in Fig. 7. Of course, lecturer must explain the purpose of control as shown in left side of Fig. 7. Also lecture explain limitation of the conventional control

IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Vol. 7, Issue 10, October 2020
DOI 10.17148/IARJSET.2020.71014

system and difficulties of modern control. In this sense, lecturer must explain that a fuzzy controller may be viewed as a real-time expert system.


Background
Motor control
Space shuttle
Turbine control
Robot
Requirement for Operation
- Efficiency
- Safety
- Flexibility against Environment
Flight control
Automobile control
Nuclear Power plant
Where is the Intelligent System?
Conventional System
(Nonlinear system
Adaptive system, Estimation, Optimization
Stochastic, PID.....ON-OFF)
Intelligent System and Control
(Fuzzy, NN, GA, PSO, AIS, BF)
Intelligent
Biological system
Fig 7. The position intelligent control of fuzzy control (from lecture ppt)

B. Lecture on Linguistic Expression of Fuzzy Set

In 1965 Lotifi A. Zadeh introduced fuzzy sets, where a more flexible sense of membership is possible. In fuzzy sets many degrees of membership are allowed. The degree of membership to a set is indicated by a number between 0 and 1, That is, a number in the interval [0,1]. The point of departure for fuzzy sets is simply generalizing the valuation set from the pair of numbers {0,1} to all numbers found in [0,1]. With this membership function fuzzy can express our linguistic expression as shown if Fig.

By expanding the valuation set we alter the nature of the characteristic function, now called membership function and denoted by $\mu_A(x)$. We no longer have crisp sets but instead have fuzzy sets. Since the interval [0,1], and we formally write this linguistic style as Fig. 8.

There are two commonly used ways of denoting fuzzy sets. If X is a universe of discourse and x is a particular element of X, then a fuzzy set A defined on X may be written as a collection of ordered pairs

$$A=\{(x,h_A(x))\}, \qquad x\in X \qquad (7)$$

where each pair $(x,h_A(x))$ is called a singleton and has x first, followed by its membership in A, $h_A(x)$. In crisp sets a singleton is simply the element x by itself. It fuzzy sets a singleton is two things: x and $\mu_A(x)$. For example, the set of small integers, A, defined (subjectively) over the universe of discourse of positive integers may be given by the collection of singletons

$$A = \{(1,1.0),(2,1.0),(3,0.75),(4,0.5),(5,0.3),(6,0.3),(7,0.1),(8,0.1)\}$$

Thus, the fourth singleton from the left tells us that 4 belongs to A to a degree of 0.5. A singleton is also written as $\mu_A(x)/x$ -that is, by putting membership first, followed by the marker "/" separating it from x^2. Singletons whose membership to a fuzzy set is zero may be omitted. The support set of a fuzzy set A is the set of its elements that have membership function other than the trivial membership of zero.

An alternative notation, used more often than equation (7.2.3), explicitly indicates a fuzzy as the union of all $\mu_A(x)/x$ singletons-that is,

$$A = \sum_{x_i\in X}\mu_A(x_i)/x_i \qquad (8)$$

For a continuous universe of discourse, we write equation (8) as

$$A = \int_X \mu_A(x)/x \qquad (9)$$

where the integral sign in equation (9) indicates the union of all $h_A(x)/x$ singletons. Consider, for example, the fuzzy set small numbers defined (subjectively) over the set of non-negative real numbers through a continuous.

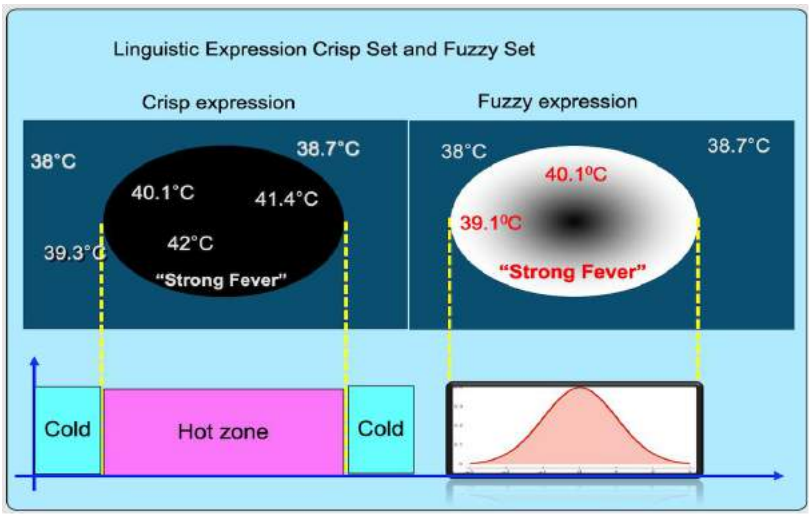


Fig 8. The graph of linguistic expression of fuzzy (from lecture ppt)

C. Lecture of Fuzzy Set Operation for Fuzzy Control

Fuzzy set can be expression more flexibly by fuzzy operation in Fig. 9. Intersection (conjunction, ^), union (disjunction) should be explained as example in Fig. 9.

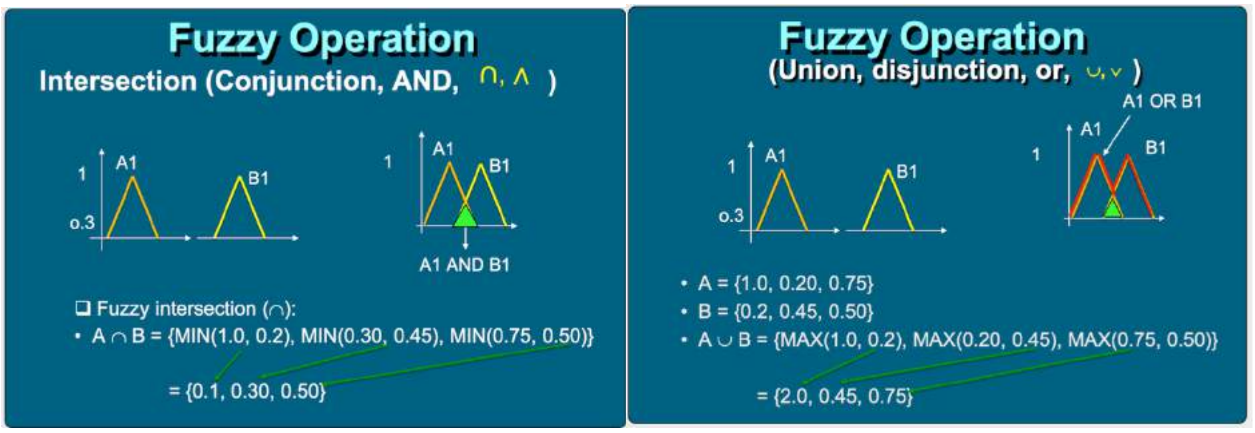

Fuzzy Operation
Intersection (Conjunction, AND, ∩, ∧)
1
0.3
A1
B1
1
A1
B1
A1 AND B1
❑ Fuzzy intersection (∩):
• A ∩ B = {MIN(1.0, 0.2), MIN(0.30, 0.45), MIN(0.75, 0.50)}
= {0.1, 0.30, 0.50}
Fuzzy Operation
(Union, disjunction, or, ∪, ∨)
A1 OR B1
A1
B1
1
1
0.3
A1
B1
• A = {1.0, 0.20, 0.75}
• B = {0.2, 0.45, 0.50}
• A ∪ B = {MAX(1.0, 0.2), MAX(0.20, 0.45), MAX(0.75, 0.50)}
= {2.0, 0.45, 0.75}

Fig 9. The graph expression of fuzzy operation (from lecture ppt)

As another example

$$\begin{aligned} A &= \mu_A(1)/1 + \mu_A(2)/2 + \mu_A(3)/3 + \mu_A(4)/4 + \mu_A(5)/5 \\ &\quad + \mu_A(6)/6 + \mu_A(7)/7 + \mu_A(8)/8 \\ &= 1.0/1 + 1.0/2 + 0.75/3 + 0.5/4 + 0.3/5 + 0.3/6 + 0.1/7 + 0.1/8 \end{aligned}$$

C. Lecture of Fuzzy Membership Function for Fuzzy Control

To design controller effectively, we must membership function as follows:

$$\text{membership function } \mu_B(x) = \frac{1}{1+\left(\frac{x}{5}\right)^3} \qquad (10)$$

Using the form of equation (9) the fuzzy set B is written as

$$B = \int_{x\geq 0} \mu_B(x)/x = \int_{x\geq 0} \left[\frac{1}{1+\left(\frac{x}{5}\right)^3} \right] /x \qquad (11)$$

The membership function of fuzzy set B is shown in Fig. 10 A graph like this is called a Zadeh diagram. The shape of membership is quite important in fuzzy controller. Therefore, lecturer should show on how we can express linguistic variations depend on the shape of membership function as shown in Fig. 10.

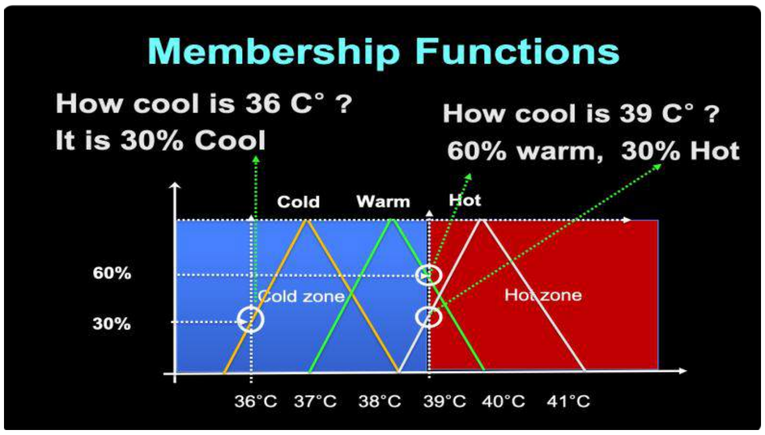


Fig 10. The graph expression of fuzzy membership function (from lecture ppt)

D. Lecture on Structure of Fuzzy Control

There are many structures of fuzzy control system in Fig. 11 (a) and Fig. 11 (b). Of course, we can find others of fuzzy control system depend on designer.

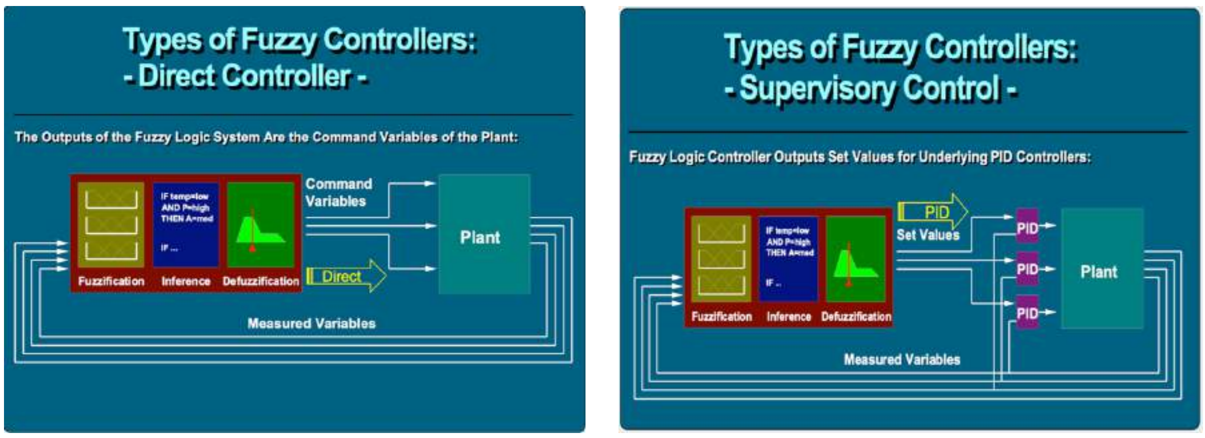


Fig 11 (a). The structure of fuzzy control system: direct and supervisory (from lecture ppt)

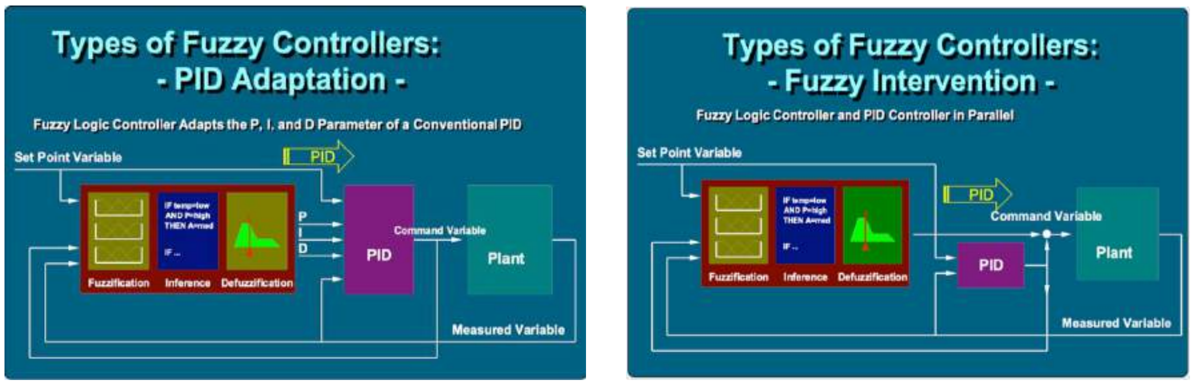


Fig 11 (b). The structure of fuzzy control system: Adaptive and intervention (from lecture ppt)

However, we can use these structures as basic teaching material of the fuzzy control system.

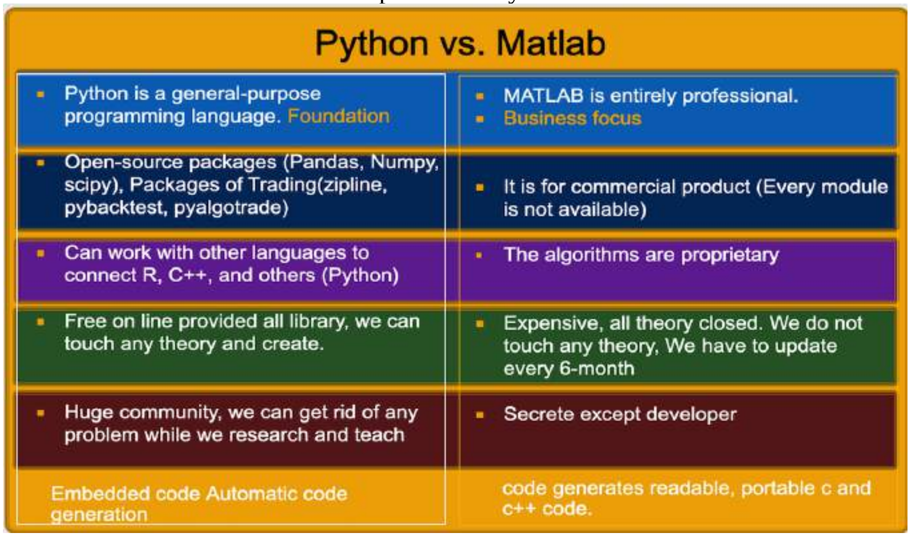
E. Lecture on Necessity Python through Comparison of Python vs. Matlab

In engineering areas, Universities have been using Matlab for lecture and learning. There was no option to teach and learn because of no alternative tool. However, we can touch a very good program language as an alternative tool like Python and Pytoch, and it is easy to learn and teach. Especially, it is free and supporting all library. Advanced countries have been supporting Matlab for teaching. However, it is not easy for under-developing countries to buy all library.
In case of Matlab, it is releasing a new version every 6 months. Each version has to have new/improved features even they did buy. The proprietary nature also makes it hard or impossible. It is serious for poor students. In Coding conception,

DOI 10.17148/IARJSET.2020.71014

there are some other issues because Matlabs uses scripts or functions as a matrix manipulation package. However, Python use disd(). That is, many students can prefer to use disp() when I want to display something. In case of programming, the python is easier to read and to program than the Matlab programming language for students because Python was developed with the goal of making a understand, while Matlab started as a Matrix based for experts. Python supports with extensive standard libraries, and has a powerful datatype such as lists, sets and dictionaries. These really help to understand student's study. Especially, 4th industrial revolution is coming now. It means public person including primary school should code to create for themselves. In that case, code should be easy and free. However, we have to pay much money and basically developing philosophy of Matlab is for experts. However, Python is for general purpose. Lecture should mention for the future. There are many advantages in Python than Matlab as follows (Table 2 and 3):

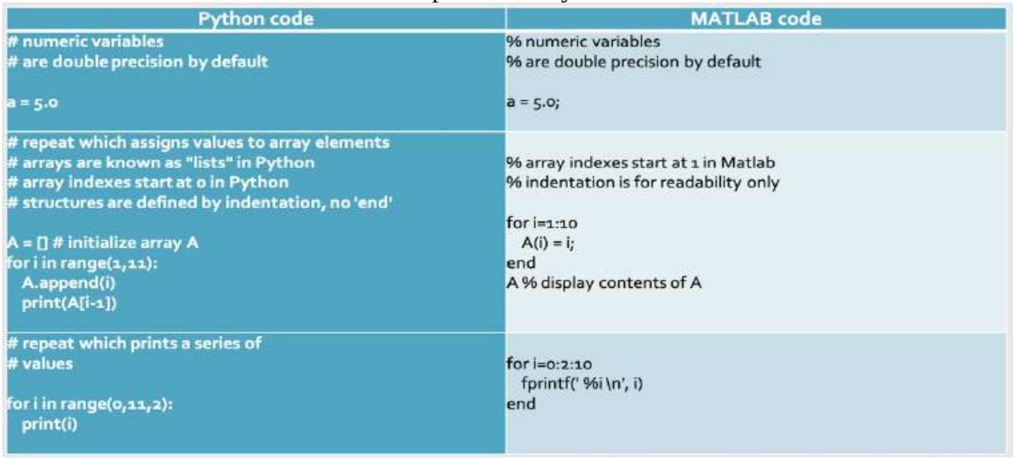
TABLE 2 Comparison of Python and Matlab



Python vs. Matlab

▪ Python is a general-purpose programming language. Foundation	▪ MATLAB is entirely professional. ▪ Business focus
▪ Open-source packages (Pandas, Numpy, scipy), Packages of Trading(zipline, pybacktest, pyalgotrade)	▪ It is for commercial product (Every module is not available)
▪ Can work with other languages to connect R, C++, and others (Python)	▪ The algorithms are proprietary
▪ Free on line provided all library, we can touch any theory and create.	▪ Expensive, all theory closed. We do not touch any theory, We have to update every 6-month
▪ Huge community, we can get rid of any problem while we research and teach	▪ Secrete except developer
Embedded code Automatic code generation	code generates readable, portable c and c++ code.

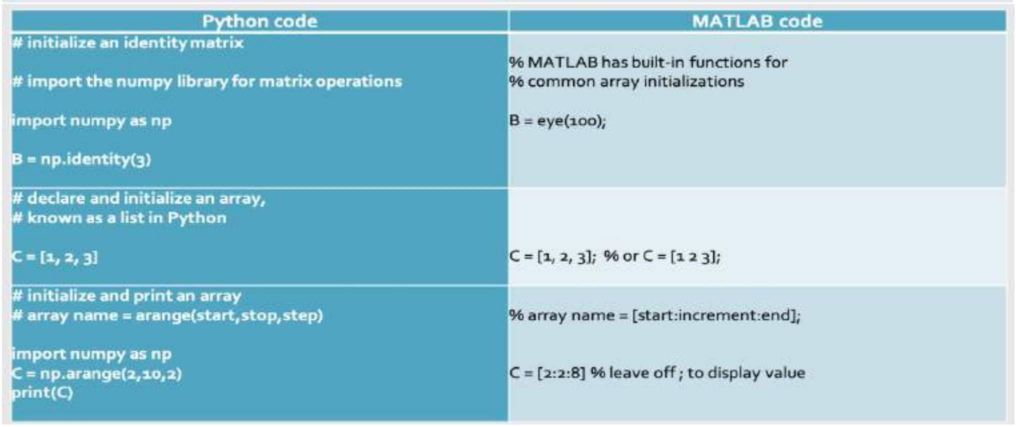
TABLE 3 Comparison of Python and Matlab



Python code	MATLAB code
# numeric variables # are double precision by default a = 5.0	% numeric variables % are double precision by default a = 5.0;

# repeat which assigns values to array elements 	
# arrays are known as "lists" in Python 	
# array indexes start at 0 in Python 	
# structures are defined by indentation, no 'end' 	
 	
A = [] # initialize array A
for i in range(1,11):
   A.append(i)
   print(A[i-1])
	% array indexes start at 1 in Matlab
	% indentation is for readability only
	
	for i=1:10
	   A(i) = i;
	end
	A % display contents of A

# repeat which prints a series of 	
# values 	
 	
for i in range(0,11,2):
   print(i)
	for i=0:2:10
	   fprintf(' %i \n', i)
	end



Python code	MATLAB code
# initialize an identity matrix # import the numpy library for matrix operations import numpy as np B = np.identity(3)	% MATLAB has built-in functions for % common array initializations B = eye(100);
# declare and initialize an array, # known as a list in Python C = [1, 2, 3]	C = [1, 2, 3]; % or C = [1 2 3];

# initialize and print an array 	
# array name = arange(start,stop,step) 	
 	
import numpy as np
C = np.arange(2,10,2)
print(C)
	% array name = [start:increment:end];
	
	C = [2:2:8] % leave off ; to display value

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology
Vol. 7, Issue 10, October 2020

DOI 10.17148/IARJSET.2020.71014

Python code	MATLAB code
# print an array element on screen # array indexes start at 0 print(C[1]) # prints 4 using C from above table cell # note square brackets C[1]	% array indexes start at 1 C(2) % prints 4 using C from above table cell % note parentheses C(2)
# declare and initialize an array # with fixed interval between values import numpy as np C = np.linspace(2,8,4) # third param is optional and = # points # between and including 1st two points # if third param left off, default # is 50 points	C = linspace(2,8,4); % third param is optional and = # points % between and including 1st two points % if third param left off, default % is 100 points

# initialize a 2D array 	
 	
D = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
	% these three examples accomplish the
	% same thing
	
	D = [1 2 3; 4 5 6; 7 8 9];
	D = [1:3; 4:6; 7:9];
	D = [1 2 3
	4 5 6
	7 8 9];

Python code	MATLAB code
# print element of 2D array # array indexes start at 0 print(D[1][1]) # row 2, column 2 # prints 5 using D from above table cell	% array indexes start at 1 D(2,2) % row 2, column 2 % prints 5 using D from above table cell
# print selected sub array of 2D array # e.g., print rows 1 to 2 of column 1 for i in range(0,2): print(D[i][0])	D(1:2,1) % rows 1 to 2 of column 1

# print all rows of column 1 of 2D 	
# array 	
 	
import numpy as np
D = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
Dsub = D[0:,0:1]
print(Dsub)
	D(:,1) % all rows, column 1

Python code	MATLAB code

# logical expression 	
 	
a = 1
b = 2
if a == 1 or b == 3:
    print('a = 2 or b = 3')
	a = 1
	b = 2;
	if a == 1 || b == 3
	    fprintf('a = 2 or b = 3 \n');
	end

# if structure 	
 	
if a == 1 and b != 3:
    print('a=1 and b not 3');
    print('OK?')
	if a == 1 && b ~= 3
	    fprintf('a=1 and b not 3 \n');
	    fprintf('OK? \n');
	end

# if, else structure 	
 	
if a != 1:
    print('a is not 1')
elif b != 3:
    print('b is not 3')
else:
    print('huh?')
	a ~= 1
	    fprintf('a is not 1 \n')
	elseif b ~= 3
	    fprintf('b is not 3 \n')
	else
	    fprintf('huh? \n')
	end

Python code	MATLAB code

# switch structure 	
 	
# Python doesn't have a switch structure 	
 	
# any switch structure can be 	
# written as an if-else structure 	
 	
# switch structures may be quicker to 	
# read and write for applications such as menus 	
	switch menuChoice
	  case 1
	    % can do any actions in a case, e.g.,
	    % call a user-defined function
	
	    myMenuFunc01();
	  case 2
	    myMenuFunc02();
	  case 3
	    myMenuFunc03();
	  otherwise
	    fprintf('invalid selection, try again')
	end

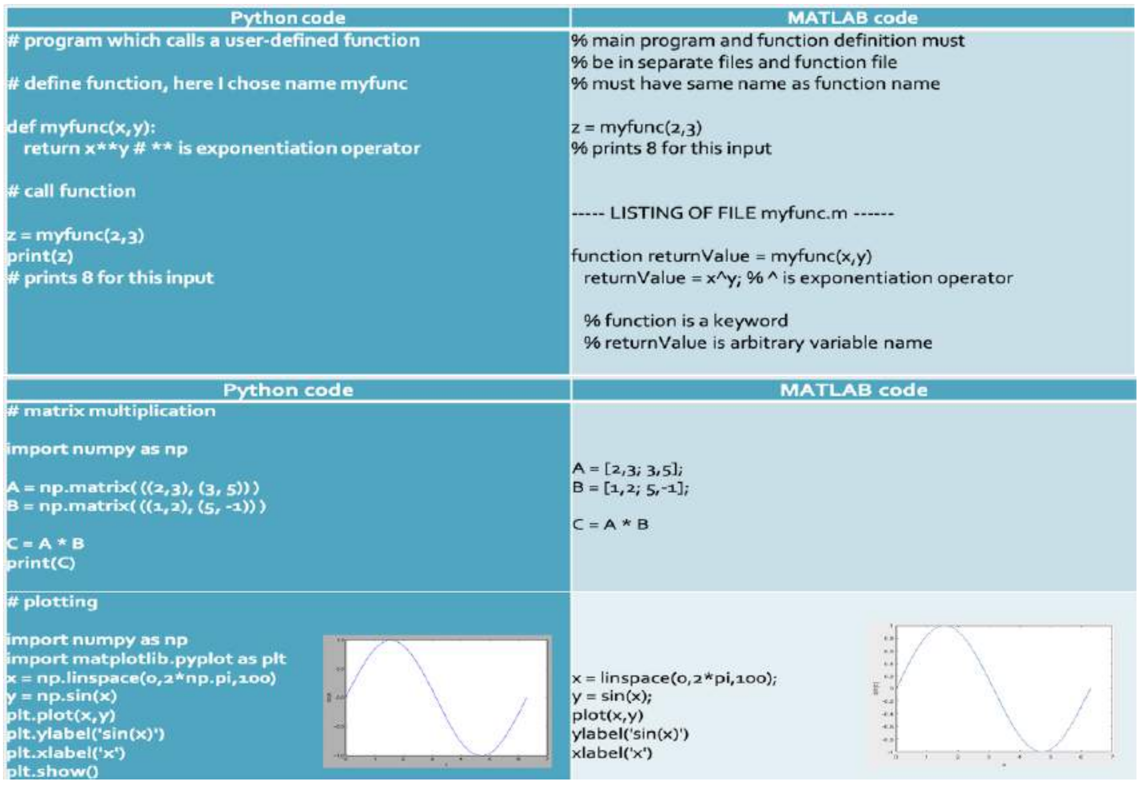
IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 7, Issue 10, October 2020

DOI 10.17148/IARJSET.2020.71014



Python code	MATLAB code

# program which calls a user-defined function 	
 	
# define function, here I chose name myfunc 	
 	
def myfunc(x,y):
  return x**y # ** is exponentiation operator
 	
# call function 	
 	
z = myfunc(2,3)
print(z)
# prints 8 for this input 	
	% main program and function definition must
	% be in separate files and function file
	% must have same name as function name
	
	z = myfunc(2,3)
	% prints 8 for this input
	
	----- LISTING OF FILE myfunc.m ------
	
	function returnValue = myfunc(x,y)
	  returnValue = x^y; % ^ is exponentiation operator
	
	  % function is a keyword
	  % returnValue is arbitrary variable name

Python code	**MATLAB code**

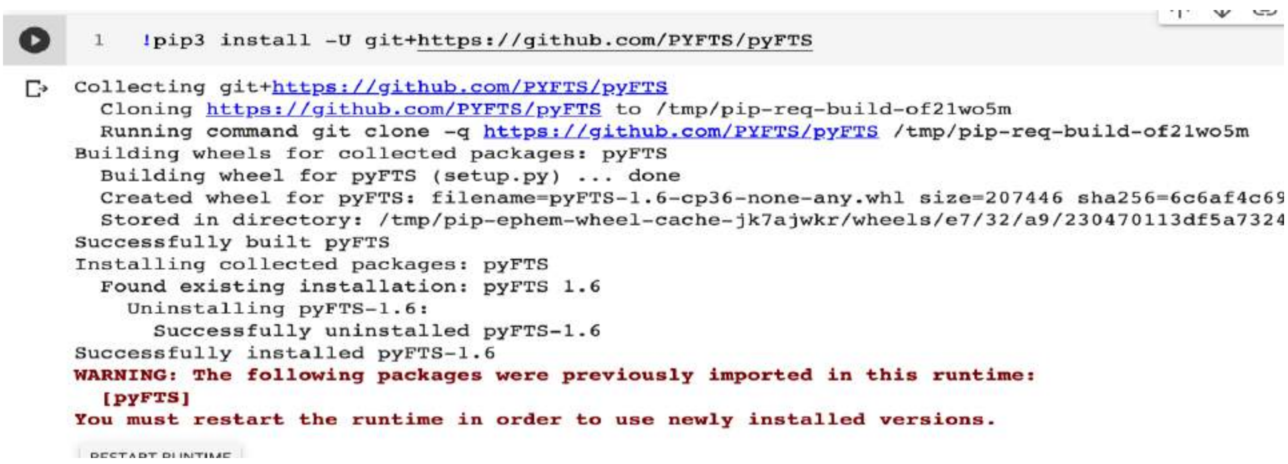
# matrix multiplication 	
 	
import numpy as np
 	
A = np.matrix(((2,3), (3, 5)))
B = np.matrix(((1,2), (5, -1)))
 	
C = A * B
print(C)
	A = [2,3; 3,5];
	B = [1,2; 5,-1];
	
	C = A * B

# plotting 	
 	
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0,2*np.pi,100)
y = np.sin(x)
plt.plot(x,y)
plt.ylabel('sin(x)')
plt.xlabel('x')
plt.show()
	x = linspace(0,2*pi,100);
	y = sin(x);
	plot(x,y)
	ylabel('sin(x)')
	xlabel('x')

F. Lecture on Implementation in Python as Example

There are many codes for implementation with Python [1, 2, 3, 4]. However, I like to candidate implementation for beginner students in the Google Colab because it is ease and it is almost done in library[1]

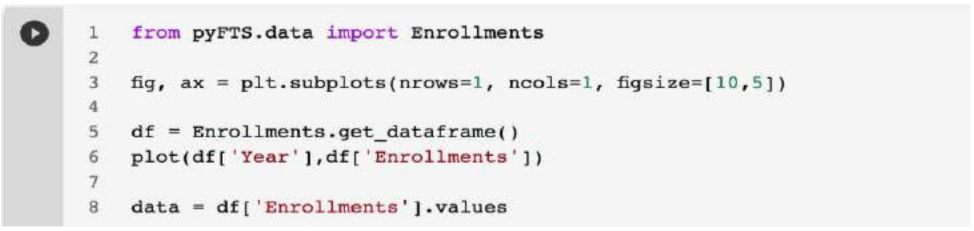
- **Install pyFTS library**



```
!pip3 install -U git+https://github.com/PYFTS/pyFTS
```

```
Collecting git+https://github.com/PYFTS/pyFTS
  Cloning https://github.com/PYFTS/pyFTS to /tmp/pip-req-build-of21wo5m
  Running command git clone -q https://github.com/PYFTS/pyFTS /tmp/pip-req-build-of21wo5m
Building wheels for collected packages: pyFTS
  Building wheel for pyFTS (setup.py) ... done
  Created wheel for pyFTS: filename=pyFTS-1.6-cp36-none-any.whl size=207446 sha256=6c6af4c69
  Stored in directory: /tmp/pip-ephem-wheel-cache-jk7ajwkr/wheels/e7/32/a9/230470113df5a7324
Successfully built pyFTS
Installing collected packages: pyFTS
  Found existing installation: pyFTS 1.6
    Uninstalling pyFTS-1.6:
      Successfully uninstalled pyFTS-1.6
Successfully installed pyFTS-1.6
WARNING: The following packages were previously imported in this runtime:
  [pyFTS]
You must restart the runtime in order to use newly installed versions.
```

- **Data Loading**



```
from pyFTS.data import Enrollments

fig, ax = plt.subplots(nrows=1, ncols=1, figsize=[10,5])

df = Enrollments.get_dataframe()
plot(df['Year'],df['Enrollments'])

data = df['Enrollments'].values
```

[1] (https://colab.research.google.com/drive/1S1QSZfO3YPVr022nwqJC5bEJvrXbqS_A#scrollTo=l6Lw2PZrwETk

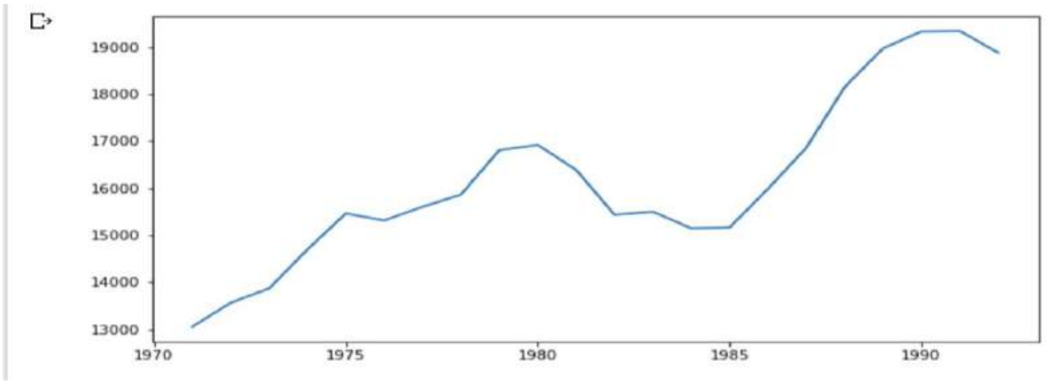
IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

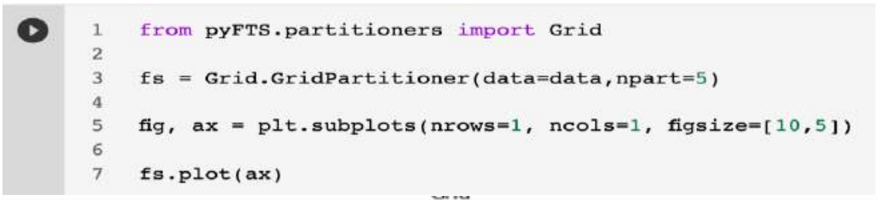
International Advanced Research Journal in Science, Engineering and Technology

Vol. 7, Issue 10, October 2020

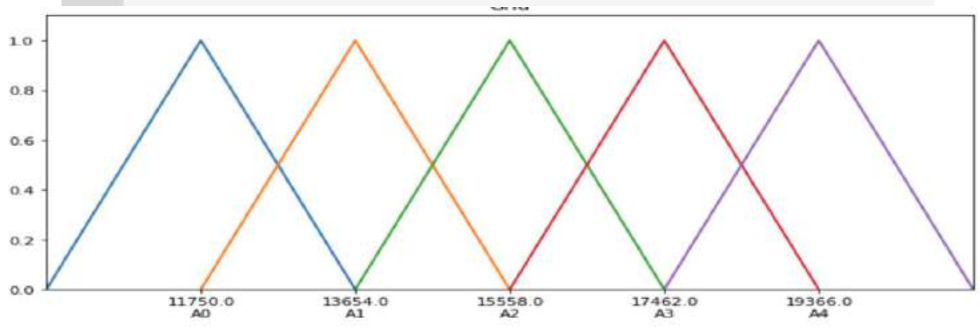
DOI 10.17148/IARJSET.2020.71014



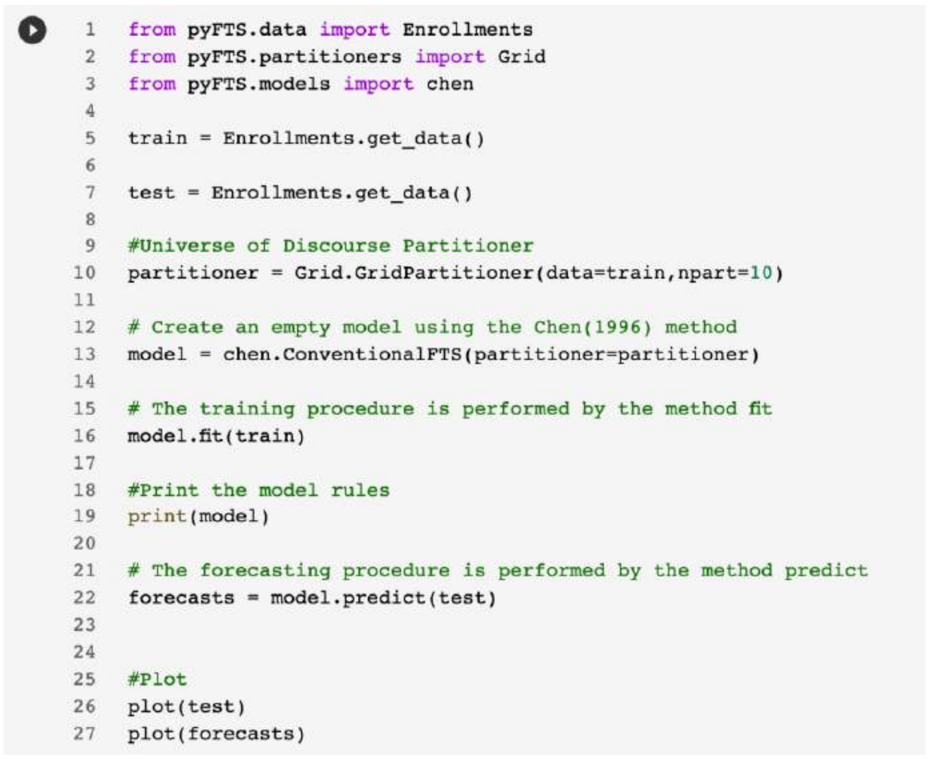
- **Fuzzy membership Function Generation**



```
from pyFTS.partitioners import Grid

fs = Grid.GridPartitioner(data=data,npart=5)

fig, ax = plt.subplots(nrows=1, ncols=1, figsize=[10,5])

fs.plot(ax)
```

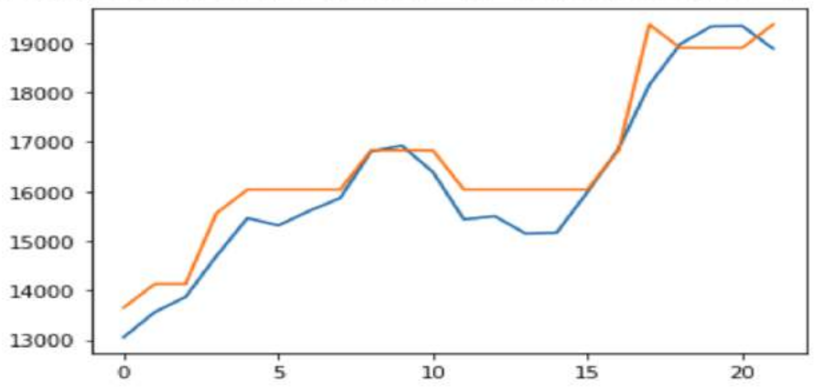


- **General Process**



```
from pyFTS.data import Enrollments
from pyFTS.partitioners import Grid
from pyFTS.models import chen

train = Enrollments.get_data()

test = Enrollments.get_data()

#Universe of Discourse Partitioner
partitioner = Grid.GridPartitioner(data=train,npart=10)

# Create an empty model using the Chen(1996) method
model = chen.ConventionalFTS(partitioner=partitioner)

# The training procedure is performed by the method fit
model.fit(train)

#Print the model rules
print(model)

# The forecasting procedure is performed by the method predict
forecasts = model.predict(test)


#Plot
plot(test)
plot(forecasts)
```

```
[<matplotlib.lines.Line2D at 0x7fa24beaa2e8>]
```



V. CONCLUSION

This paper deals with lecture skill of advanced fuzzy control in Python. Fuzzy set commonly is useful for complex and uncertain plants that cannot express through well-known linear approaches.
In almost control system, complex controllers are needed to attain expected stability, safety, energy saving, and robustness. Fuzzy logic control is one the powerful tool as an intelligent technique because it allows the translation from crisp logic statements to linguistic expression. Although it has been proven to effectively deal with complex plants, many recent studies have already proved from the basic premise of linguistic interpretability.
This is because fuzzy systems modeling, analysis, and control incorporate a certain amount of human knowledge into its components (fuzzy sets, fuzzy logic, and fuzzy rule base).
Because in the earlier 1970s, fuzzy systems and fuzzy control theories was introduced as an emerging technology targeting industrial applications and these lectures have added a promising new dimension to the existing domain of conventional control systems engineering for students with Matlab.
It is now a common technology that provide a set of complex system or reasonably accurate mathematical model, certain human experience in linguistic terms, fuzzy systems and fuzzy control theories. Additionally, fuzzy control methods and algorithms, including many specialized software and hardware is also available on the market today. It means that it can be designed and applied as one type of intelligent control using Python. Therefore, teaching well should be given for merits over many areas with other language like Python.
We are standing on the line of 4th industrial revolution. Its core technology is artificial intelligence and we should implement with code. In that case, it is difficult for public person like primary students to understand because Matlab is developed for expert. That is, algorithm is difficult to understand for them and they cannot obtain code and library because of expensive price. Serious one is that they do not open all algorithm for public person because their basic philosophy is business.
That is why we should share lecture skill intelligent control and its application using Python, which is free and open library. Teaching skill is not high technology. However, it needs lots of experiences. This is why this paper deal with teaching skill and these considerations give an influence on advancing 4th industrial revolution in your country.

REFERENCES

[1]. https://github.com/jsoma/fuzzy_pandas
[2]. https://github.com/carmelgafa/ml_from_scratch/find/master
[3]. https://towardsdatascience.com/a-very-brief-introduction-to-fuzzy-logic-and-fuzzy-systems-d68d14b3a3b8
[4]. https://towardsdatascience.com/fuzzy-inference-system-implementation-in-python-8af88d1f0a6e
[5]. The scikit-fuzzy Documentation Release 0.2 (2016)
[6]. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, IEEE Std. 802.11, 1997.
[7]. Bernard, J. A., Use of a Rule-Based System for Process Control, IEEE Control Systems Magazine, pp. 3-13, October 1988.
[8]. Cox, E., Adaptive Fuzzy Systems, IEEE Spectrum, pp. 27-31, February 1993.
[9]. Driankov, D., Hellendoorn, H., and Reinfrank, M., An Introduction to Fuzzy Control, Springer-verlag, Berlin, 1993.
[10]. Dubois, D., and Prade, H., Fuzzy Sets and Systems: Theory and Applications, Academic Press, Boston, 1980.
[11]. Efstathiou, J., Rule-Based Process Control Using Fuzzy Logic, in Approximate Reasoning in Intelligent Systems, Decision and Control, E. Sanchez and L. A. Zadeh, soning in Intelligent Systems, Decision and Control, E. Sanchez and L. A. Zadeh, eds., Pergamon press, Oxford, 1987, pp. 145-158.
[12]. Fuller, R., and Zimmermann, H.-J., On Computation of the Compositional Rule of Inference under Triangular Norms, Fuzzy Sets and Systems, Vol. 51, pp. 267-275, 1992.
[13]. Graham, B. P., and Newell, R. B., Fuzzy Identification and Control of a Liquid Level Rig, Fuzzy Sets and Systems, Vol. 26, pp. 255-273, 1988.
[14]. Harris, C. J., Moore, C. G., and Brown, M., Intelligent Control-Aspects of Fuzzy Logic an d Neural Nets, World Scientific, Singapore, 1993.
[15]. He, S-Z., Tan, S., and Xu, F-L., Fuzzy Self-Tuning of PID Controller, Fuzzy Sets and Systems, Vol. 56, pp. 37-46, 1993.

[16]. Hirota, K., Industrial Applications of Fuzzy Control, Springer-Verlag, Tokyo, 1993.
[17]. Jager, R., Fuzzy Logic in Control, Unpublished Ph. D. Dissertation, University of Delft, The Netherlands, 1995.
[18]. Jang, J-S., and Gulley, N., Fuzzy Logic Toolbox User's Guide, Mathworks, 1995.
[19]. Jang, J-S., and Sun, C-T., Neuro-Fuzzy Modeling and Control, Proceedings of the IEEE, Vol. 83, No. 3, pp. 378-406, 1995.
[20]. Jiangin, C., and Laijiu, C., Study on Stability of Fuzzy Closed-Loop Control Systems, Fuzzy Sets and Systems, Vol. 57, pp. 159-168, 1993.
[21]. Johnson, C. D., Process Control Instrumentation Technology, John Wiley & Sons, New York, 1970.
[22]. King, P. J., and Mamdani, E. H., The Application of Fuzzy Control Systems to Industrial Processes, in Fuzzy Automata and Decision Processes, M. M. Gupta, G. N. Saridis, and B. R. Gaines, eds., North-Holand, New York, 1977, pp. 321-330.
[23]. King, R. E., and Karonis, F. C., Multi-Level Expert Control of a Large-Scale Industrial Process, in Fuzzy Computing Theory, Hardware, and Applications, M. M. Gupta and T. Yamakawa, eds., Elsevier/North-Holland, Amsterdam, 1988, pp. 323-340.
[24]. Kiszka, J. B., Gupta, M. M., and Nikiforuk, P. N., Some Properties of Expert Control Systems, in Approxiamte Reasoning in Expert Systems, M. M. Gupta, A. Kandel, W. Bandler, and J. B. Kiszka, eds., Elsevier/North-Holland, Amsterdam, 1985, pp. 283-306.
[25]. Larkin, L. I., A Fuzzy Logic Controller for Aircraft Flight Control, in Industrial Applications of Fuzzy Control, M. Sugeno, ed., North-Holland, New York, 1985, pp. 87-103.
[26]. Lee, C. C., Fuzzy Logic in Control Systems: Fuzzy Logic Controller-Part-I, IEEE Transactions on Systems, Man and Cybernetics, Vol. 20, No. 2, pp. 404-418, March/April 1990a.
[27]. Lee, C. C., Fuzzy Logic in Control Systems: Fuzzy Logic Controller-Part-II, IEEE Transactions on Systems, Man and Cybernetics, Vol. 20, No. 2, pp. 419-435, March/April 1990b.
[28]. Maeda, M., and Murakami, S., A Self-Tuning Fuzzy Controller, Fuzzy Sets and Systems, Vol. 51, pp. 29-40, 1992.
[29]. Mamdani, E. H., Application of Fuzzy Algorithms for Control of Simple Dynamic Plants, Proceedings of IEE, Vol. 121, No. 12, pp. 1585-1588, 1974.
[30]. Mamdani, E. H., Advances in the Linguistic Synthesis of Fuzzy Controllers, International Journal of Man-Machine Studies, Vol. 8, pp. 669-678, 1976.
[31]. Mamdani, E. H., Applications of Fuzzy Set Theory to Control Systems: A Survey, in Fuzzy Automata and Decision Processes, M. M. Gupta, G. N. Saridis, and B. R. Gaines, eds., North-Holland, New York, 1977, pp. 1-13.
[32]. Ostergaard, J. J., and Holmblad, L. P., Control of Cement Kiln by Fuzzy Logic, in Fuzzy Information and Decision Processes, M. M. Gupta and E. Sanchez, eds. North-Holland, Amsterdam, 1982, pp. 389-399.
[33]. Pedrycz, W., Fuzzy Control and Fuzzy Systems, second (extended) edition, John Wiley & Sons, New York, 1993.
[34]. Schwartz, D. G., Fuzzy Logic Flowers in Japan, IEEE Spectrum, pp. 32-35, July 1992.
[35]. Shoureshi, R., and Rahmani, K., Derivation and Application of an Expert Fuzzy Optimal Control System, Fuzzy Sets and Systems, Vol. 49, pp. 93-101, 1992.
[36]. Sugeno, M., An Introductory Survey of Fuzzy Control, Information Sciences, Vol. 36, pp. 59-83, 1985.
[37]. Terano, T., Asai, K., and Sugeno, M., Fuzzy Systems Theory and Its Applications, Academic Press, Boston, 1992.
[38]. Tsukamoto, Y., An Approach to Fuzzy Reasoning Method, Advances in Fuzzy Set Theory and Applications, M. M. Gupta, R. K. Ragade, and R. R. Yager, eds., North-Holland, Amsterdam, 1979, pp. 134-149.
[39]. Tzafestas, S. G., and Papanikolopoulos, N. P., Incremental Fuzzy Expert PID Control, IEEE Transactions on Industrial Electronics, Vol. 37, No. 5, pp. 365-371, 1990.
[40]. Wang, Li-Xin, Aaptive Fuzzy Systems and Control, Prentice-Hall, Englewood Cliffs, NJ, 1994.
[41]. Yager, R. R., and Filev, D. P., SLIDE: A Simple Adaptive Defuzzification Method, IEEE Transactions on Fuzzy Systems, Vol. 1, No. 1, pp. 69-78, 1993.
[42]. Yager, R. R., and Filev, D. P., Essential of Fuzzy Modeling and Control, John Wiley & Sons, New York, 1994.
[43]. Yamakawa, T., Fuzzy Hardware Systems of Tomorrow, in Approximate Reasoning in Intelligent Systems, Decision and Control, E Sanchez and L. A. Zadeh, eds., Pergamon Press, Oxford, 1987, pp. 1-20.
[44]. Yamakawa, T., Stabilization of an Inverted Pendulum by a High-Speed Fuzzy Logic Controller Hardware System, Fuzzy Sets and Systems, Vol. 32, pp. 161-180, 1989.
[45]. Yasunobu, S., and Miyamoto, S., Automatic Train Operation System by Predictive Fuzzy Control, in Industrial Application of Fuzzy Control, M. Sugeno, ed., NorthHolland, Amsterdam, 1985, pp. 1-18.
[46]. Zimmermann, H.-J., Fuzzy Set Theory and Its Applications, Kluwer-Nijhoff, Boston, 1985.

BIOGRAPHY

Dong Hwa Kim Ph.D: Dept. of Computational Intelligence and Systems Science, Interdisciplinary Graduate School of Science and Engineering (AI Application for Automatic control), TIT (Tokyo Institute of Technology), Tokyo, Japan. Hanbat National University (Dean, Prof., S. Korea), He has experience in many University, overseas as Prof. He was NCP of EU-FP7 (EU-Framework Program, ICT). He had keynote speak at several international conference and University. He has 200 papers in Journal and conferences. He was editor of IJCIR (International Journal of Computational Intelligence). He is current Prof. at Electrical Power and Control Eng. Adama Science and Tech. Uni., Ethiopia (http://worldhumancare.wixsite.kimsite).