

Please be ready to return myRIO, tools
& components during the Semester 2
break, in the first week of September
Compulsory

Advanced Mechatronics System Design – MANU2451

All Students

Optional Workshop day could be after
Semester 2 finishes, at the end of October

Non Compulsory

Advanced Mechatronics System Design – MANU2451

All Students

Week 12 – Artificial Intelligence II: Fuzzy Control

Advanced Mechatronics System Design – MANU2451

Dr Chow Yin LAI

Edited by Dr Milan Simic

School of Engineering

RMIT University, Victoria, Australia

Email: milan.simic@rmit.edu.au

New Teaching Schedule

Week		Class Activity Before	Lecture	Team Activity During or After
1			Introduction to the Course / Introduction to LabVIEW	LabVIEW Programming
2			Introduction to LabVIEW / Data Acquisition	LabVIEW Programming
3			Gripper / Introduction to Solidworks / Safety	Gripper Design
4			Sensors I	myRIO Programming for Sensor Signal Reading / Gripper Design
5			Sensors II	myRIO Programming for Sensor Signal Reading
6			Actuators I	LabVIEW Tutorial
7		LabVIEW Assessment.	DC Motors I	Matlab Simulink Simulation
8		Design report submission	DC Motors II	Matlab Simulink Simulation / myRIO Programming for Control
9			Actuators II	Matlab Simulink Simulation
10			Modeling and System Identification	Matlab Simulink Simulation LabVIEW Simulation
11			Artificial Intelligence I	Matlab Simulation LabVIEW Simulation
12			Artificial Intelligent II	Revision

Contents

- Fuzzy Logic Control
 - Introduction to Fuzzy Logic Control
 - Basic Idea
 - Terminology
 - 2 Fuzzy Variables Case
 - MATLAB Simulink Example

Contents

- Fuzzy Logic Control
 - Introduction to Fuzzy Logic Control
 - Basic Idea
 - Terminology
 - 2 Fuzzy Variables Case
 - MATLAB Simulink Example

Introduction

- Assume you went to a restaurant, and at the end of the visit, you decided to give some tips.
- The amount of tips would depend on the quality of food and service.
- Your mind works as follows:

		Food				
		Very bad	Bad	Neutral	Good	Very good
Service	Very bad	No	No	No	A little	Some
	Bad	No	No	A little	Some	More
	Neutral	No	A little	Some	More	A lot
	Good	A little	Some	More	A lot	A lot
	Very good	Some	More	A lot	A lot	A lot

- This is fuzzy logic – Descriptions are just fuzzy and qualitative, but you can make final decisions based on these descriptions.

Introduction – Variables

- Input variables **are fuzzy**:
 - Food quality
 - Service quality
- Output function ???

		Food				
		Very bad	Bad	Neutral	Good	Very good
Service	Very bad	No	No	No	A little	Some
	Bad	No	No	A little	Some	More
	Neutral	No	A little	Some	More	A lot
	Good	A little	Some	More	A lot	A lot
	Very good	Some	More	A lot	A lot	A lot

Introduction – Variables

- Input variables **are fuzzy**:
 - Food quality
 - Service quality
- Output function *tip* is also fuzzy

<i>Fuzzy</i>	<i>Crisp</i>
No	\$0
A little	2.5%
Some	5%
More	7.5%
A lot	10%

Why Fuzzy Logic Controller?

- Previously, we learned about the PID controller.
- We **had to tune the parameters** of the controller to get satisfactory response.
 - Depending on the complexity or nonlinearities of the plant, this may not be easy.
- On the other hand, some **experienced human operators** can easily control the system based on linguistic terms. E.g.
 - If the motor is **too fast**, give a **big negative voltage** to slow it down rapidly.
 - If the motor is a **little fast**, give a **small negative voltage** to slow it down.
 - If the motor is **about right**, give a **voltage close to 0**.
 - If the motor is a **little slow**, give a **small positive voltage** to speed it up.
 - If the motor is **too slow**, give a **big positive voltage** to speed it up rapidly.

Why Fuzzy Logic Controller?

- The words “too slow”, “a little slow”, “about right”, “a little fast”, “too fast” are all quite **vague / fuzzy**.
- Even the words “big negative”, “small negative”, “small positive” and “big positive” are vague / fuzzy as well.
- However, using all these very vague / fuzzy terms, we can actually calculate a definite voltage value (e.g. 4.2V) to send out to the motor to change the speed.
 - This is what the Fuzzy Logic Controller would do.

For the Gripper Motor Driver we have



One direction Stop The other direction

Why Fuzzy Logic Controller?

- Fuzzy Logic Controller (FLC) **emulates human deductive thinking**, i.e. how human make / infer conclusions about what they know.
- FLC allows us to capture the operator's experience into the control rule.
- It incorporates **ambiguous human logic** into the computer program.
- It is especially useful for the control of systems where:
 - The model is hard to obtain
 - The parameters may vary with time
 - The system is too complex
 - A PID is too simple to provide good response
 - You have a good qualitative (NOT quantitative) understanding of the system.



This is one branch of **Intelligent Control**!

- It uses **Artificial Intelligence** computing approach.

History and Applications

- Fuzzy Logic was proposed by Lotfi A. Zadeh of UC Berkeley in 1965.
- First industrial application in Denmark in 1973: Cement kiln.
- Popularity increased in Japan:
 - Sendai Railway for control of acceleration, braking and stopping in 1987.
 - Inverted Pendulum 1987.
 - Matsushita vacuum cleaners for **adjustment of suction power** according to dust.
 - Hitachi washing machines to choose the **best use of power, water and detergent** based on load weight, fabric mix, dirt sensors.
 - Canon autofocus camera to control speed of lens movement.
 - Mitsubishi air conditioner with 25 heating and 25 cooling rules.
- Research started to increase in US and Europe thereafter.

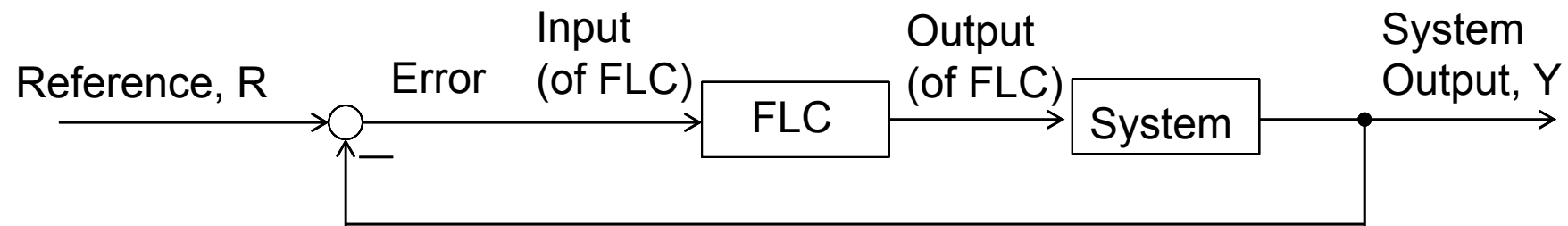
Contents

- Fuzzy Logic Control
 - Introduction to Fuzzy Logic Control
 - Basic Idea
 - Terminology
 - 2 Fuzzy Variables Case
 - MATLAB Simulink Example

Fuzzy Logic Controller

Controlled variable is *speed*

- The control scheme:

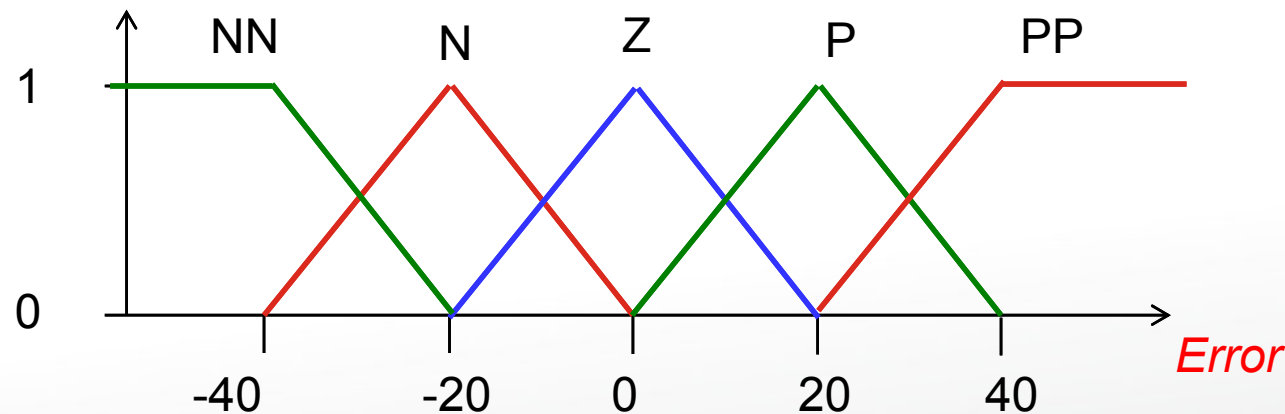


- First of all, we transform the words “Too fast”, “Too slow” etc. into the context of “Error” which is $(R - Y)$.
 - Too fast $\rightarrow Y \gg R \rightarrow$ “Error very negative” \rightarrow Give very negative voltage.
 - A little fast $\rightarrow Y > R \rightarrow$ “Error negative” \rightarrow Give negative voltage.
 - About right $\rightarrow Y \approx R \rightarrow$ “Error almost zero” \rightarrow Give almost zero voltage.
 - A little slow $\rightarrow Y < R \rightarrow$ “Error positive” \rightarrow Give positive voltage
 - Too slow $\rightarrow Y \ll R \rightarrow$ “Error very positive” \rightarrow Give very positive voltage.

Input Fuzzy Sets

- Let's see how to make this work in an actual computer program.
- First, we define **5 input fuzzy sets** for input Variable: NN, N, Z, P, PP ("very negative error" etc)

Degree of Membership (DOM)

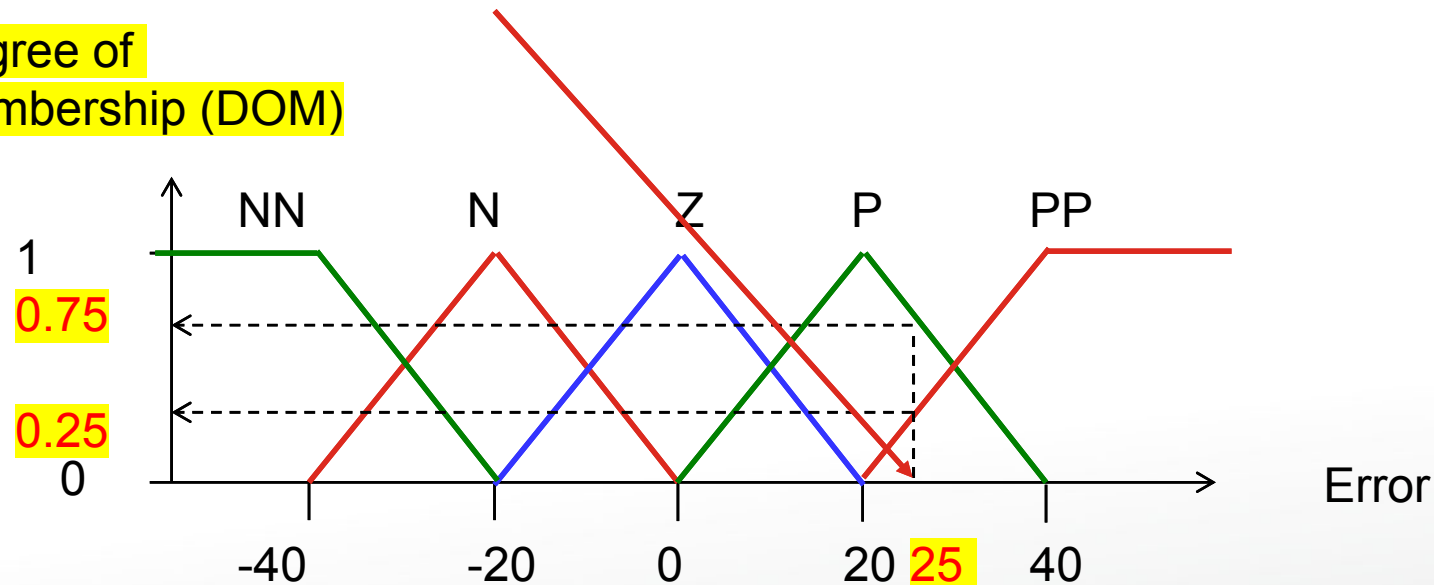


- Note that what values of error are considered too positive etc. are **based on operator's experience**.

Input Fuzzy Sets

- We will use numerical example for illustration of the concept.
- Assume that the error is 25. This is a “crisp” (clearly defined) value.

Degree of
Membership (DOM)

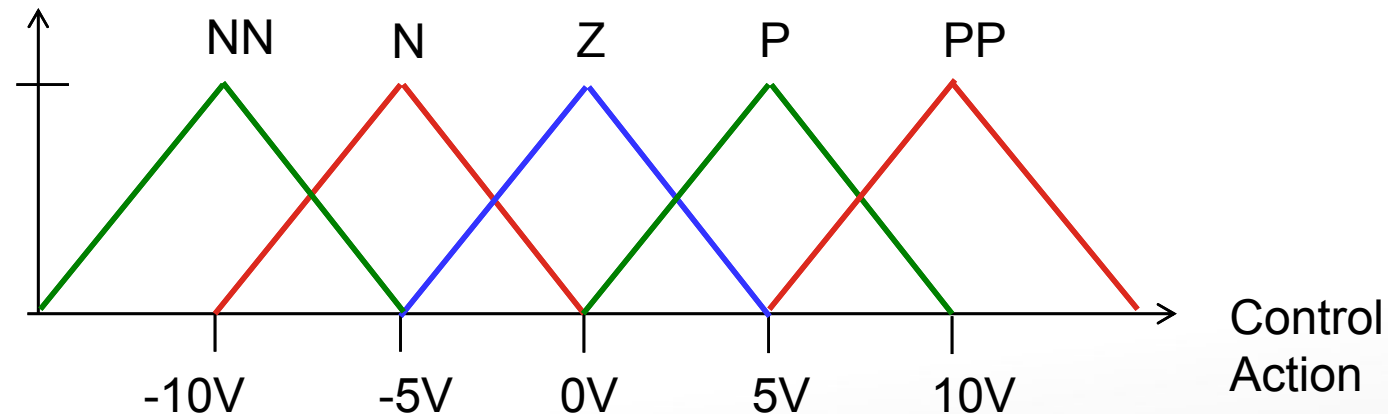


- Then it's **DOM** for the P-set is 0.75;
- And the **DOM** for PP-set is 0.25.
- The DOM shows how much percentage the number belongs to a certain set.

Fuzzy Sets for *Output*

Variable called Control Voltage

- Next, we also define 5 **Output Fuzzy Sets**. (“very negative voltage” etc.)
- We then assign output set-values to the 5 fuzzy sets:



- Note that what values of control actions are also based on operator's experience.
- Note also that the fuzzy set for NN and PP do not go to saturation. The reason will be explained later.

Output Set-Values

- Using the numerical example just now, we can calculate the final FLC output to be sent to the motor:

Set	Input DOM	x	Output Set Value	=	Output Value
NN	0	X	-10	=	0
N	0	X	-5	=	0
ZO	0	X	0	=	0
P	0.75	X	5	=	3.75
PP	0.25	x	10	=	2.5
			SUM	=	6.25

- Thus, if **error is 25**, the **FLC sends 6.25V** to the motor.

Remarks

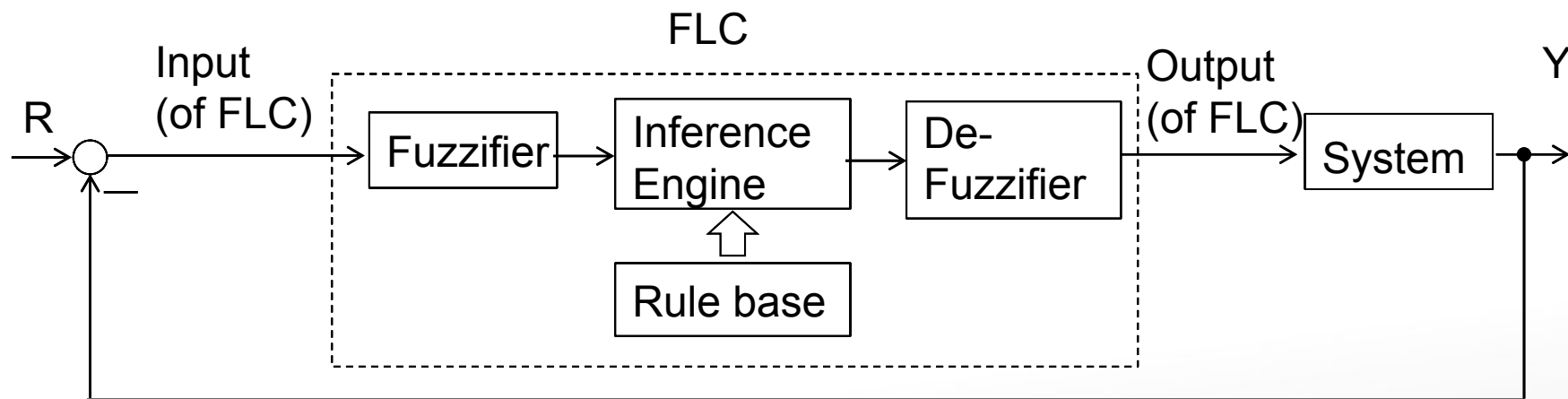
- In this simple example, it may not yet be clear what the advantages of FLC are.
- The use of FLC will become clearer when we have **more fuzzy variables**.

Contents

- Fuzzy Logic Control
 - Introduction to Fuzzy Logic Control
 - Basic Idea
 - Terminology
 - 2 Fuzzy Variables Case
 - MATLAB Simulink Example

Terminology

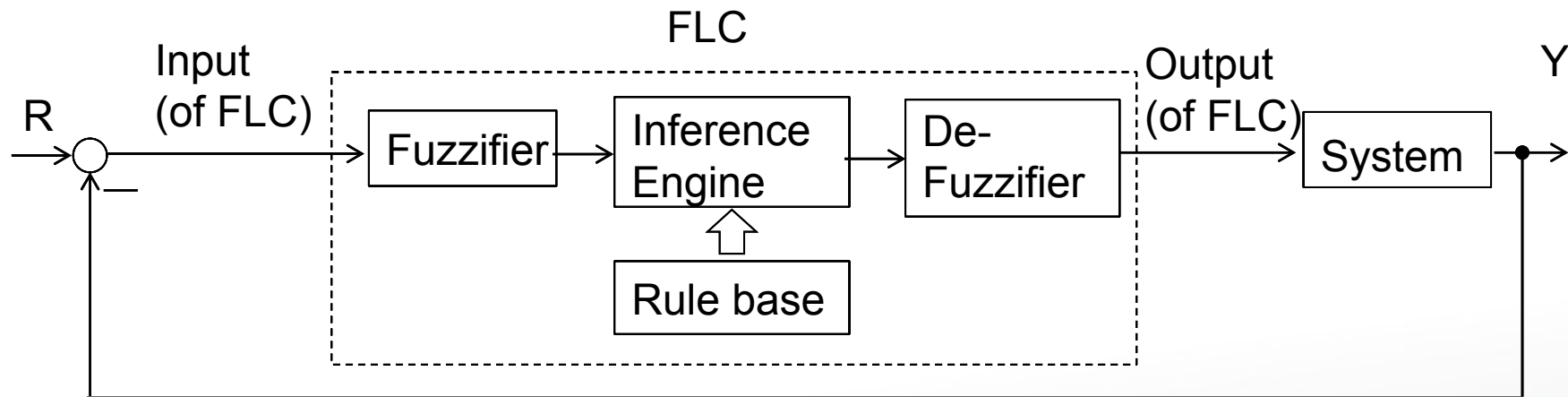
- Using the example, some terminologies of Fuzzy Logic Controller (FLC) will be introduced:



- The input "error = 25" is a "crisp" value.
- The output "6.25V" is also a "crisp" value.

Terminology

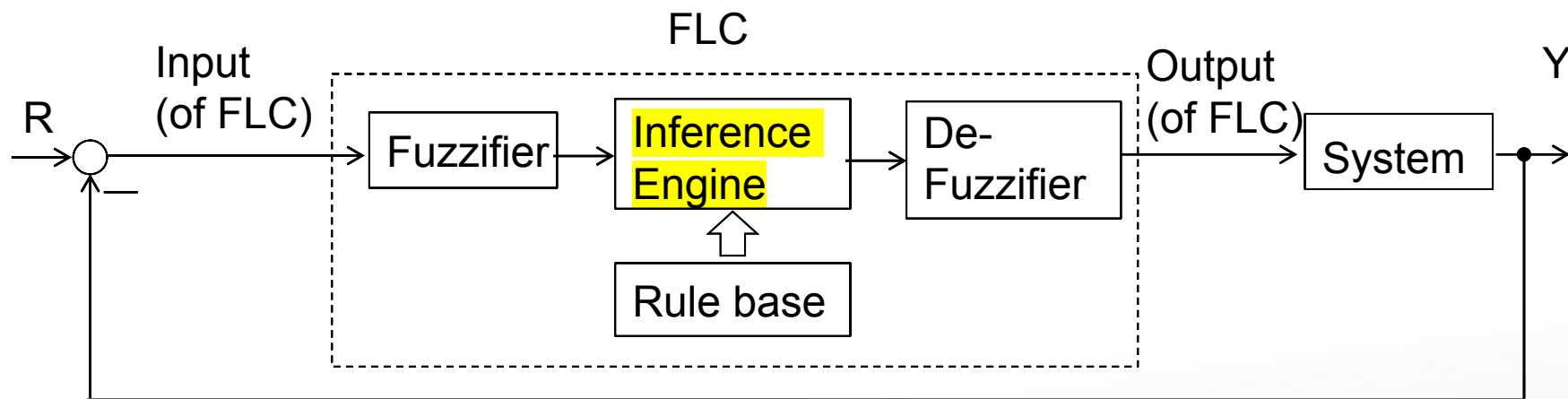
- Using the example, some terminologies of Fuzzy Logic Controller will be introduced:



- The **fuzzifier** turns the crisp input to a “fuzzy” value in terms of DOM.
- The **rule base** is where ALL the rules are stored, e.g. “if error is positive, give positive voltage”.

Terminology

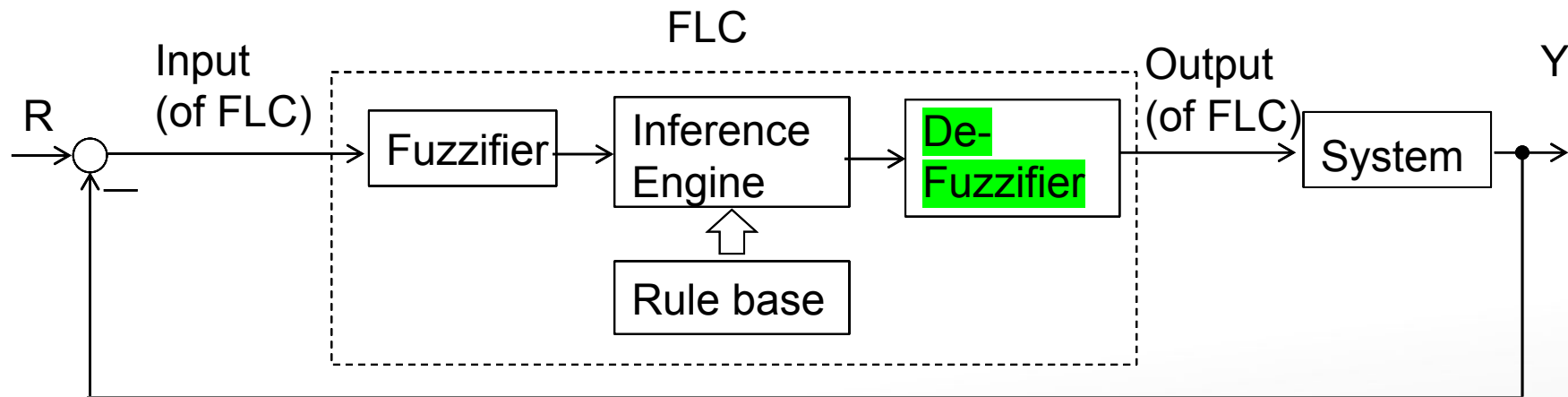
- Using the example, some terminologies of Fuzzy Logic Controller will be introduced:



- The **inference engine** calculates the **certainty of the recommendations** from the rule base.
 - In this example, the **certainty that you should use P voltage is 0.75**, and the certainty that you should use the PP is 0.25.

Terminology

- Using the example, some terminologies of Fuzzy Logic Controller will be introduced:



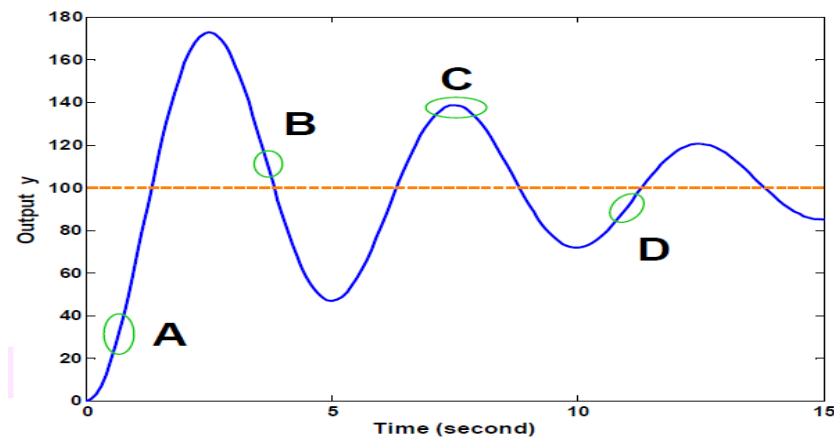
- The **De-Fuzzifier** combines the recommendation from the inference engine into a “crisp” value of *voltage which can be sent to the motor*.

Contents

- Fuzzy Logic Control
 - Introduction to Fuzzy Logic Control
 - Basic Idea
 - Terminology
 - 2 Fuzzy Variables Case
 - MATLAB Simulink Example

2 Fuzzy Variables

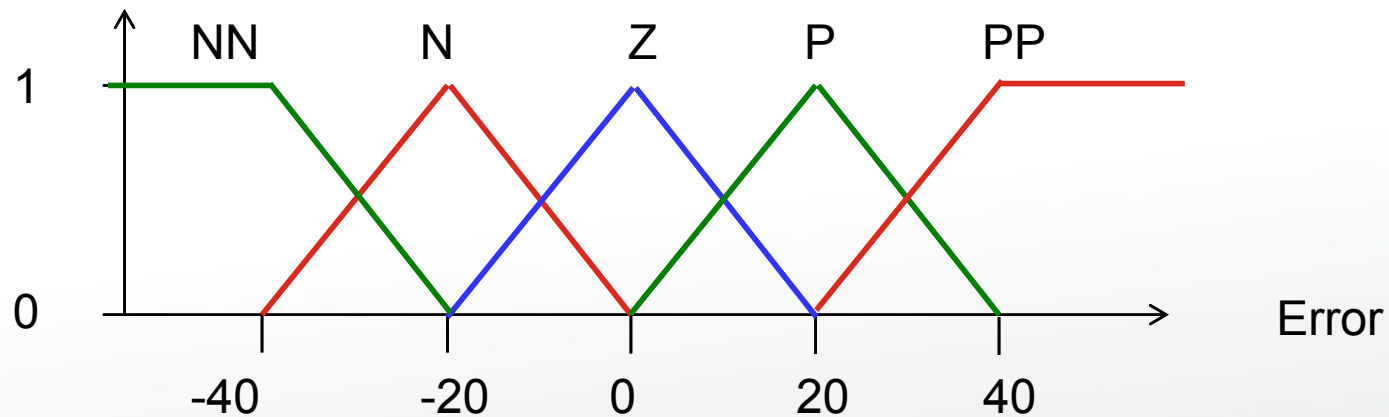
- Let's now move on to using 2-Fuzzy Variables: Error and ΔError
- E.g.
- A: Error PP, ΔError NN
- B: Error N, ΔError PP
- C: Error N, ΔError Z
- D: Error P, ΔError N



2 Fuzzy Variables - Input Fuzzy Sets

- Let's create the input fuzzy sets. We need two sets, one for Error and one for ΔError .
- First, we define the input fuzzy sets for *input variable* Error:

Degree of
Membership (DOM)



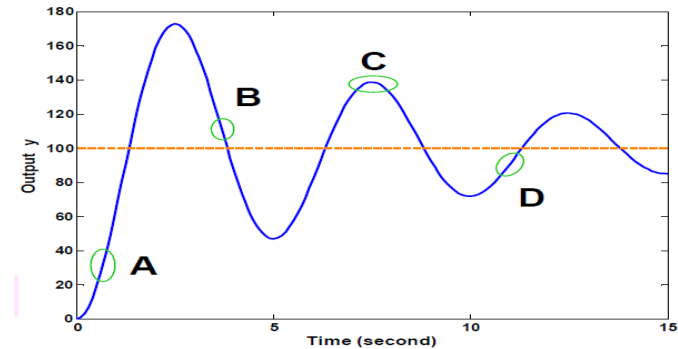
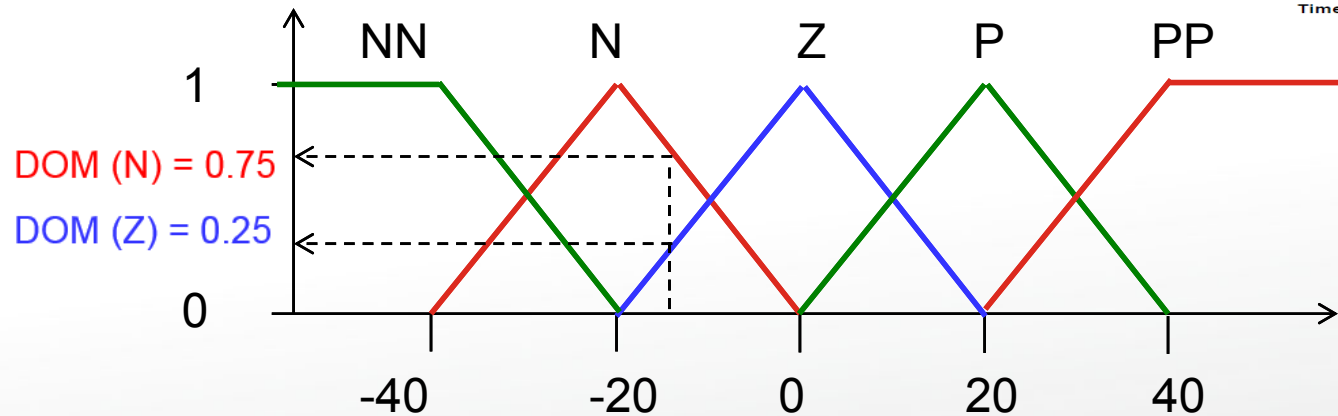
2 Fuzzy Variables - Input Fuzzy Sets

- Numerical example for point B: Error = -15. $100 - 115 = -15$

- DOM (N) = 0.75

- DOM (Z) = 0.25

Degree of
Membership (DOM)

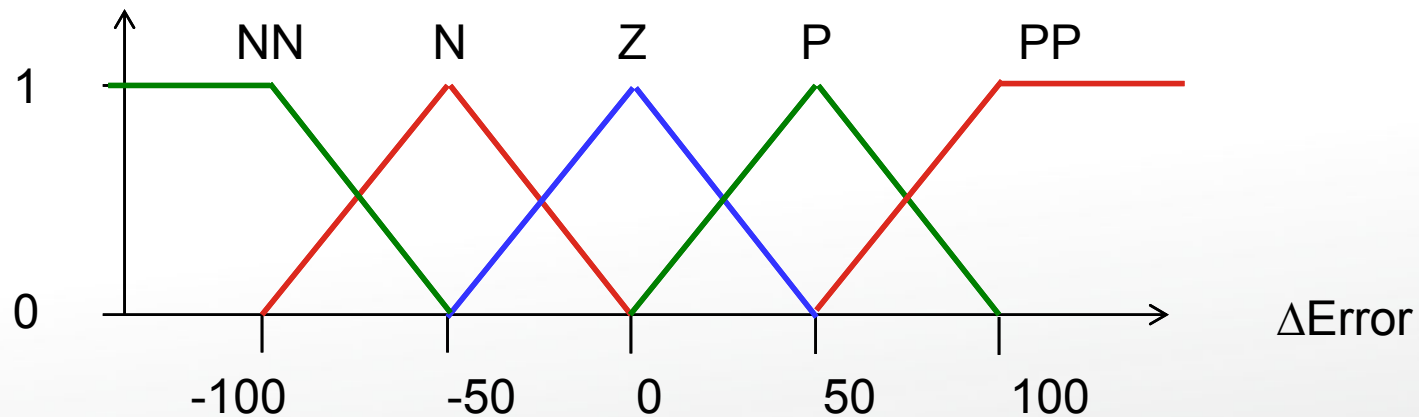


Error

2 Fuzzy Variables - Input Fuzzy Sets

- Next, we define the input fuzzy sets for *input variable* ΔError .
- Note that the **x-axis** should cover the range of **expected values of ΔError** . It will **not be the same** as the range of Error.

Degree of
Membership (DOM)

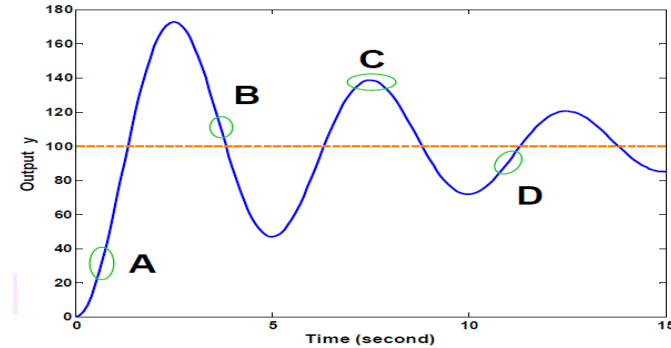
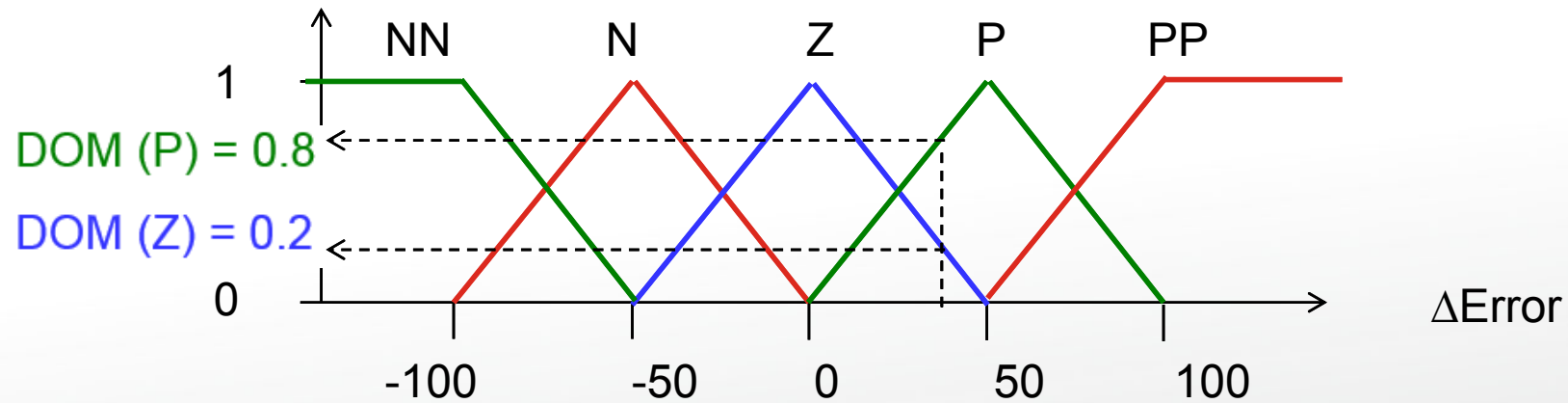


2 Fuzzy Variables - Input Fuzzy Sets

- Numerical example for point B: $\Delta\text{Error} = 40$.

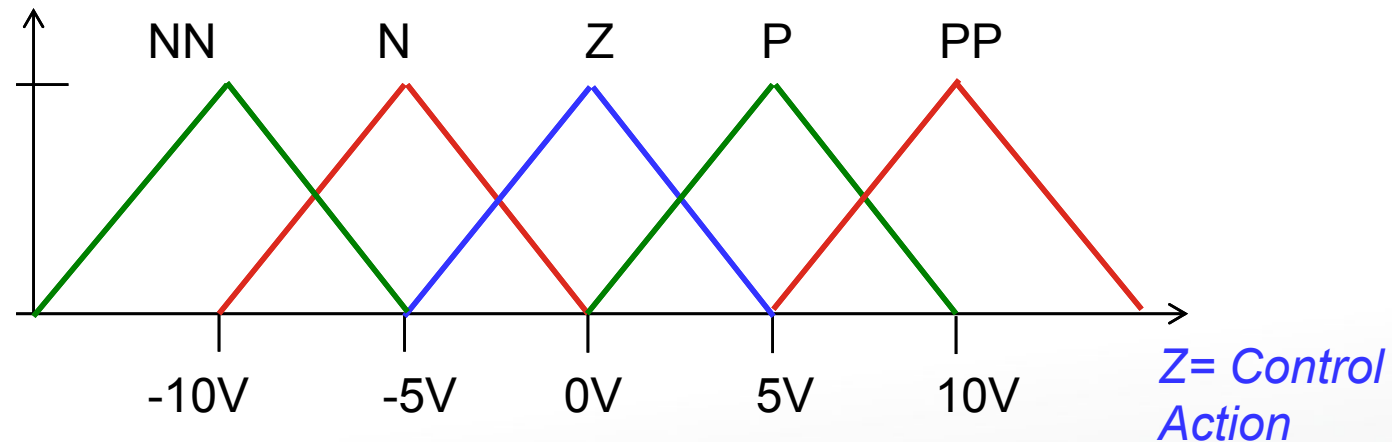
- $\text{DOM}(P) = 0.8$
- $\text{DOM}(Z) = 0.2$

Degree of
Membership (DOM)



2 Fuzzy Variables – Output Fuzzy Sets

- For the 5 possible control action in rule base, we create the following output fuzzy sets:



- Note that we **only need one** of this, because there is only one output.
- The question is, how to **combine the effect of two inputs into one output**.

2 Fuzzy Variables – Rules Base

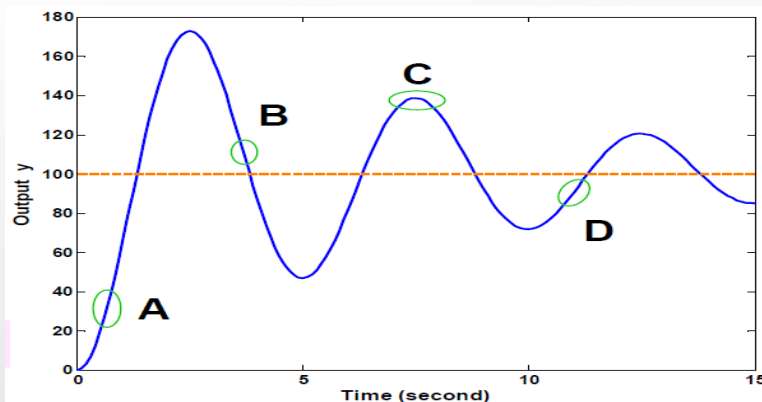
- Based on the **operator's experience**, we can create the following **rule base**:

X = Error

Y = Δ Error

	NN	N	Z	P	PP
PP	Z	P	PP	PP	PP
P	N	Z	P	PP	PP
Z	NN	N	Z	P	PP
N	NN	NN	N	Z	P
NN	NN	NN	NN	N	Z

**Z =
Control
Actions**



2 Fuzzy Variables – Rules Base

A: Error PP, Δ Error NN

B: Error N, Δ Error PP

C: Error N, Δ Error Z

D: Error P, Δ Error N

- Based on the **operator's experience**, we can create the following **rule base**:

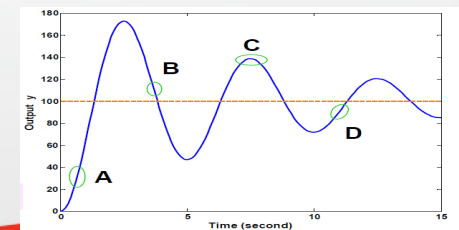
X = Error

Y = Δ Error

	NN	N	Z	P	PP
PP	Z	P	PP	PP	PP
P	N	Z	P	PP	PP
Z	NN	N	Z	P	PP
N	NN	NN	N	Z	P
NN	NN	NN	NN	N	Z

Z =
Control
Actions

- A: x (PP), y (NN) \rightarrow z (Z)
 - Zero voltage** and let inertia brings the motor to desired speed.
- B: x (N), y (PP) \rightarrow z (P)
 - Positive voltage** to prevent slowing down too fast.



2 Fuzzy Variables – Rules Base

- You should now interpret C & D on your own.

A: Error PP, Δ Error NN

B: Error N, Δ Error PP

C: Error N, Δ Error Z

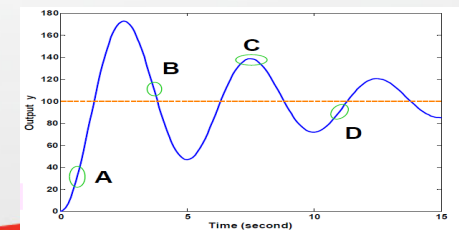
D: Error P, Δ Error N

$X = \text{Error}$

$Y = \Delta \text{Error}$

	NN	N	Z	P	PP
PP	Z	P	PP	PP	PP
P	N	Z	P	PP	PP
Z	NN	N	Z	P	PP
N	NN	NN	N	Z	P
NN	NN	NN	NN	N	Z

$Z =$
Control
Actions



2 Fuzzy Variables – Rules Base

A: Error PP, Δ Error NN

B: Error N, Δ Error PP

C: Error N, Δ Error Z

D: Error P, Δ Error N

- Based on the **operator's experience**, we can create the following **rule base**:

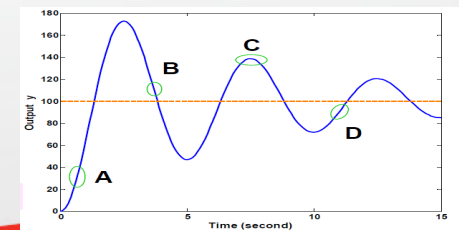
X = Error

Y = Δ Error

	NN	N	Z	P	PP
PP	Z	P	PP	PP	PP
P	N	Z	P	PP	PP
Z	NN	N	Z	P	PP
N	NN	NN	N	Z	P
NN	NN	NN	NN	N	Z

Z =
Control
Actions

- C: x (N), y (Z) \rightarrow z (N)
 - Negative voltage
- D: x (P), y (N) \rightarrow z (Z)
 - Zero voltage



Inference

$Y = \Delta \text{Error}$

$X = \text{Error}$

	<u>NN</u>	<u>N</u>	<u>Z</u>	<u>P</u>	<u>PP</u>
<u>PP</u>	Z	P	PP	PP	PP
<u>P</u>	N	Z	P	PP	PP
<u>Z</u>	<u>NN</u>	<u>N</u>	<u>Z</u>	<u>P</u>	<u>PP</u>
<u>N</u>	NN	NN	N	Z	P
<u>NN</u>	NN	NN	NN	N	Z

Z =
Control
Actions

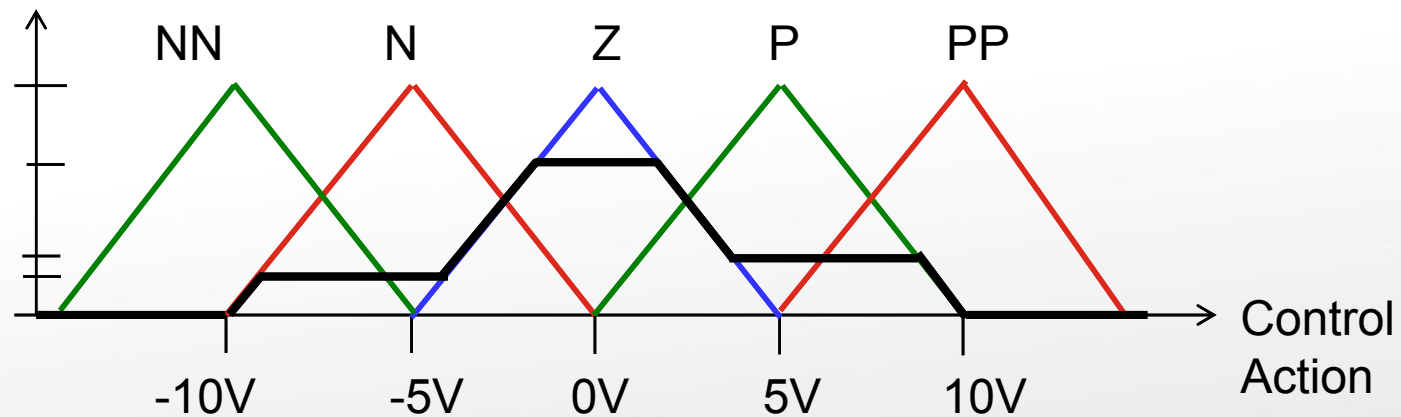
Inference engine calculates the **certainty** of the **recommendations** from the rule base.

- For 2 fuzzy variables case, there are a lot of different rules proposed in the literature. Here we show **Mandani's Min-Max method**:
- Use the example of point B:
 - For Error: $\text{DOM}(\text{N}) = 0.75$ & $\text{DOM}(\text{Z}) = 0.25$
 - For ΔError : $\text{DOM}(\text{P}) = 0.8$ & $\text{DOM}(\text{Z}) = 0.2$
 - There are **4 combinations** of the DOM's:

DOM Error	DOM ΔError	DOM <u>Min</u>	Control Action from Rule Base
N = 0.75	P = 0.8	0.75	$\text{N} \ \& \ \text{P} \rightarrow \text{Z}$
N = 0.75	Z = 0.2	0.2	$\text{N} \ \& \ \text{Z} \rightarrow \text{N}$
Z = 0.25	P = 0.8	0.25	$\text{Z} \ \& \ \text{P} \rightarrow \text{P}$
Z = 0.25	Z = 0.2	0.2	$\text{Z} \ \& \ \text{Z} \rightarrow \text{Z}$

2 Fuzzy Variables – Defuzzification

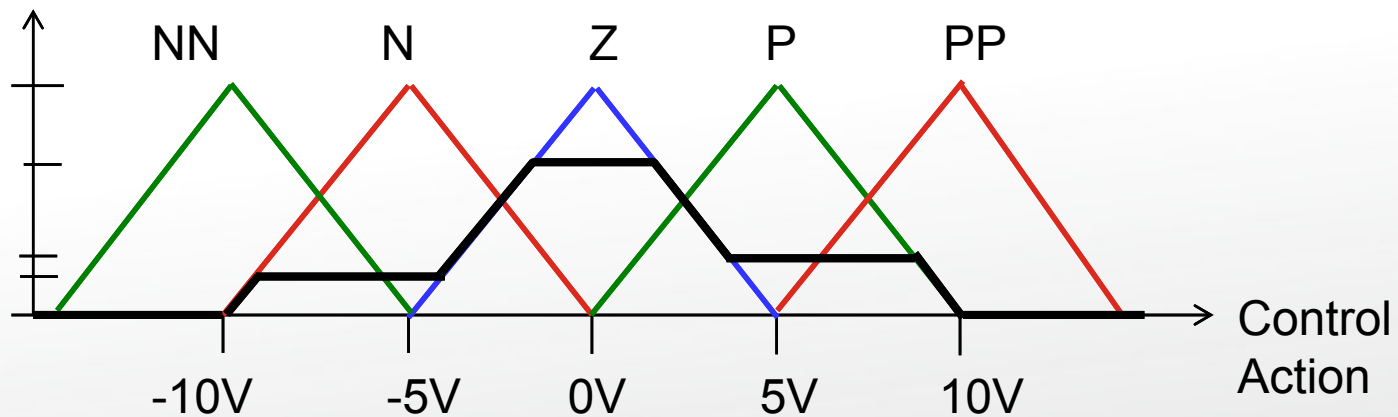
- Finally, we use **maximum function** and **center of gravity (CoG)** to perform defuzzification.
 - Max of N** control action is 0.2 (*looking at columns 3 and 4 previous page*)
 - Max of Z** control action is 0.75
 - Max of P** control action is 0.25
 - We draw lines in the fuzzy sets to indicate this:



2 Fuzzy Variables – Defuzzification

- To calculate the center of gravity for the area under the black curve, use the following method:
 - Let $\mu(z)$ be the equation of each “straight line” segment.
 - Then the **center of gravity** is

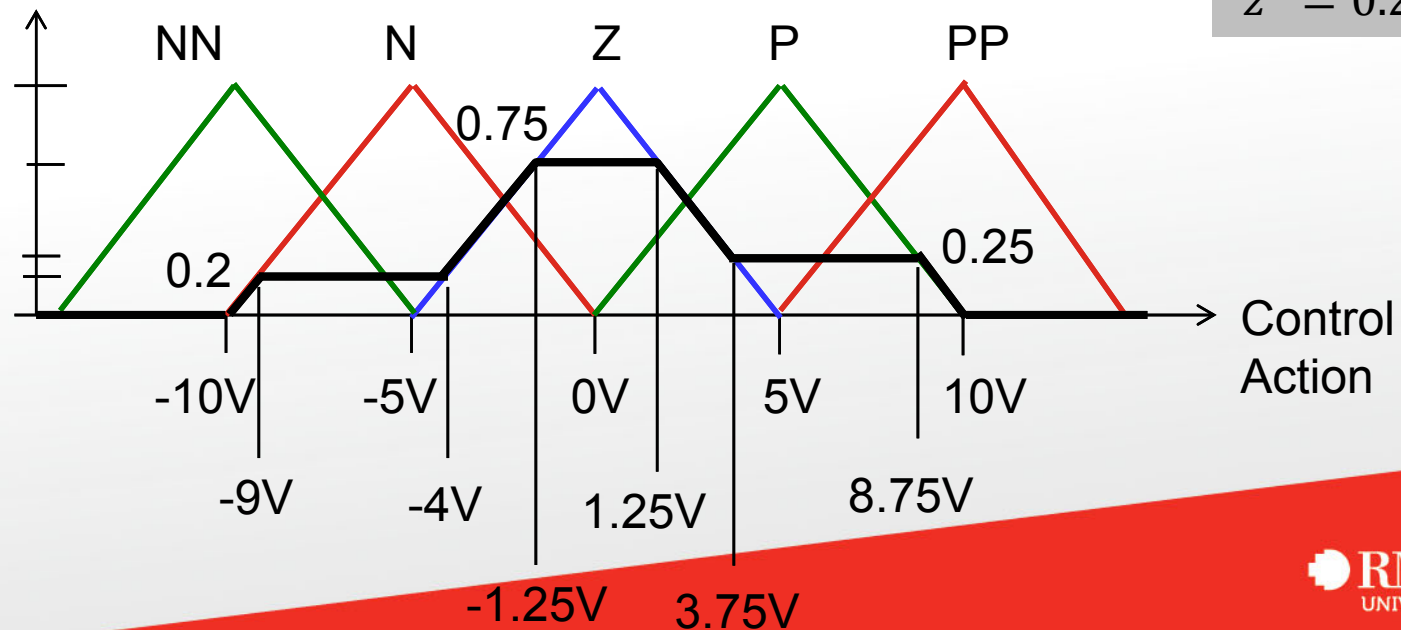
$$z^* = \frac{\int \mu(z) \cdot z \, dz}{\int \mu(z) \, dz}$$



2 Fuzzy Variables – Defuzzification

- In the following example:

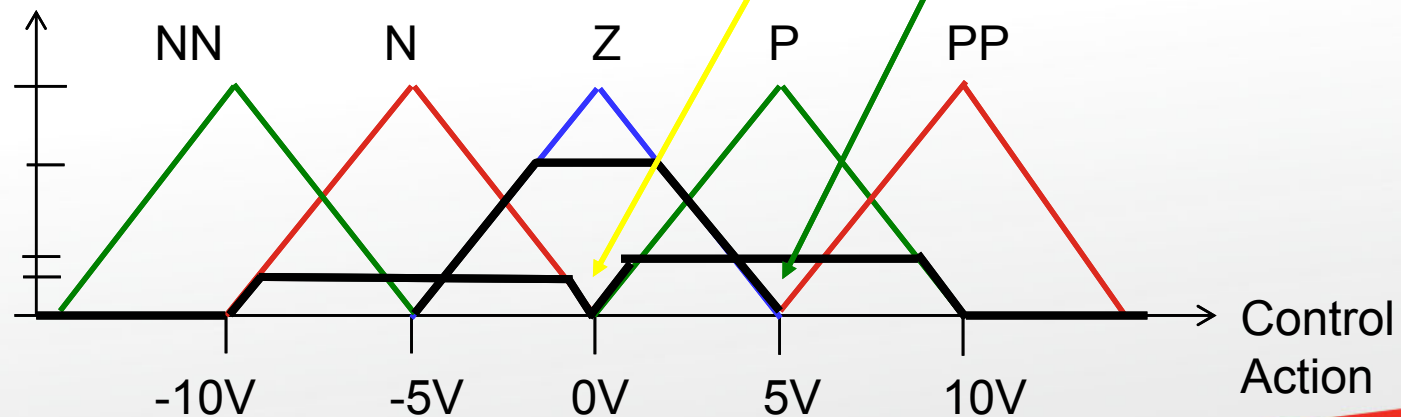
$$z^* = \frac{\left[\int_{-10}^{-9} (0.2z + 2)z dz + \int_{-9}^{-4} (0.2)z dz + \int_{-4}^{-1.25} (0.2z + 1)z dz + \int_{-1.25}^{1.25} (0.75)z dz \right. \\ \left. + \int_{1.25}^{3.75} (-0.2z + 1)z dz + \int_{3.75}^{8.75} (0.25)z dz + \int_{8.75}^{10} (-0.2z + 2)z dz \right]}{\left[\int_{-10}^{-9} (0.2z + 2) dz + \int_{-9}^{-4} (0.2) dz + \int_{-4}^{-1.25} (0.2z + 1) dz + \int_{-1.25}^{1.25} (0.75) dz \right. \\ \left. + \int_{1.25}^{3.75} (-0.2z + 1) dz + \int_{3.75}^{8.75} (0.25) dz + \int_{8.75}^{10} (-0.2z + 2) dz \right]}$$





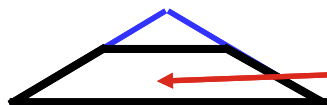
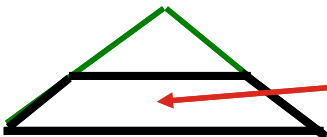

2 Fuzzy Variables – Defuzzification

- The **Center of Gravity** method can be hard to implement.
- Another method which provides reasonable output, yet easy to implement, is called the **Center of Sum (CoS)** method.
 - Let b_i be the **center of the fuzzy sets**. For e.g. $b_Z = 0$, $b_P = 5$.
 - Let $\int \mu_i$ be the **area under each membership function μ_i**
 - Then the **center of gravity** is

$$\frac{\sum_i b_i \int \mu_i}{\sum_i \int \mu_i}$$



2 Fuzzy Variables – Defuzzification

Output Fuzzy Set	B_i	Graph	Area <small>Under black line</small>	$B_i * \text{Area}$
NN	-10		0	0
N	-5		1.8	$-5 * 1.8 = -9$
Z	0		4.6875	0
P	5		2.1875	$5 * 2.1875 = 10.9375$
PP	10		0	0

$$\frac{\sum_i b_i \int \mu_i}{\sum_i \int \mu_i} = \frac{-9 + 10.9375}{1.8 + 4.6875 + 2.1875} = 0.2233$$

Close to CoG method.

Some areas included twice.

2 Fuzzy Variables – Defuzzification

- So the final conclusion is, **send out 0.2297V (CoG) or 0.2233V (CoS)** to the motor **when it is at point B**.
- Remark: Earlier on, it was mentioned that the NN and PP of output fuzzy set do not go to saturation.
- One reason is that we need to have fair calculation of area under each membership function.
 - If NN and PP are not triangular like the others, their area could become infinite.
- For 2 fuzzy variable case, the output voltage is also automatically limited to acceptable range, e.g. **+/-10V**.
 - If all DOM points towards NN, then $(b_i * \text{area}) / \text{area} = b_i = -10V$.
 - Or if all DOM points towards PP, then $(b_i * \text{area}) / \text{area} = b_i = 10V$.

Further Advantages of FLC

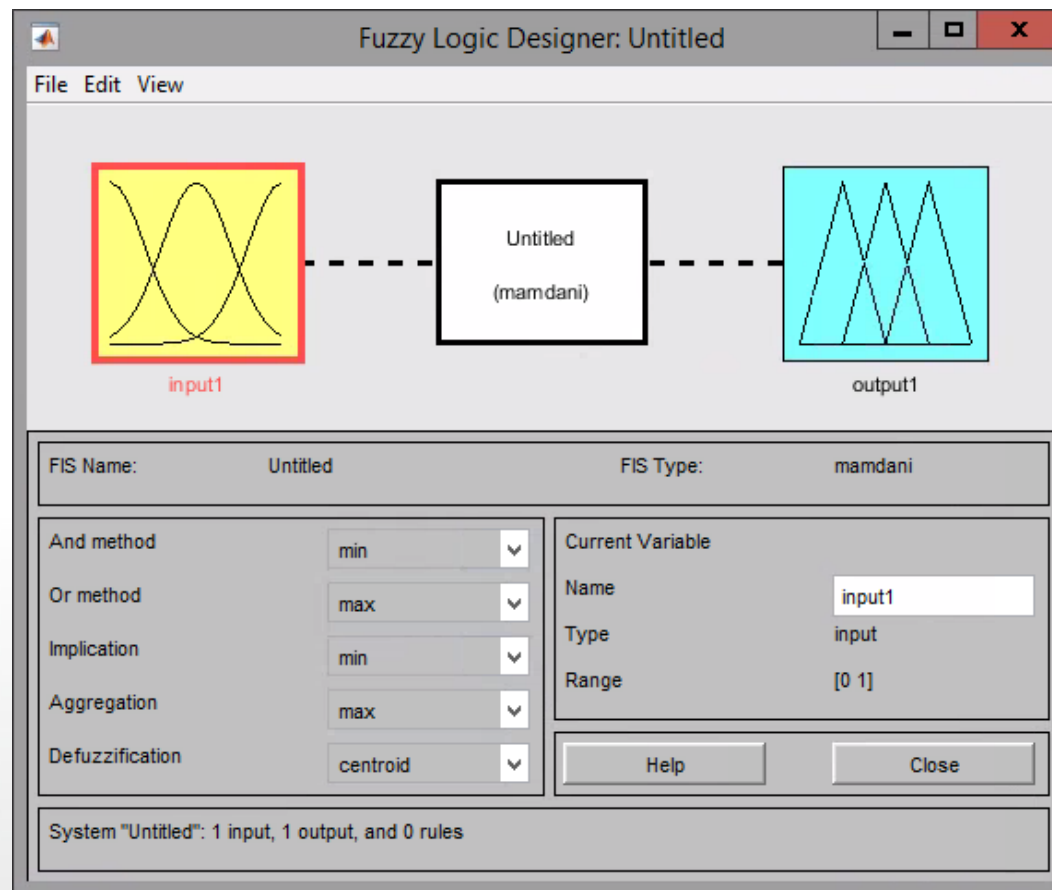
- Tremendous **Degrees-of-Freedom** for fine tuning:
 - **Number of membership functions**:
 - Shown in examples is 5, but any number is possible.
 - Different input-variables can have different numbers.
 - The **shape of membership functions** can be arbitrary:
 - Triangular, trapezoidal, Gaussian etc.
 - Symmetrical or asymmetrical.
 - The **support for each label** can be different:
 - The center of the membership functions can be 'anywhere', as long as all the membership functions cover the whole range of expected values.
 - The rule base can be **highly nonlinear**.

Contents

- Fuzzy Logic Control
 - Introduction to Fuzzy Logic Control
 - Basic Idea
 - Terminology
 - 2 Fuzzy Variables Case
 - MATLAB Simulink Example

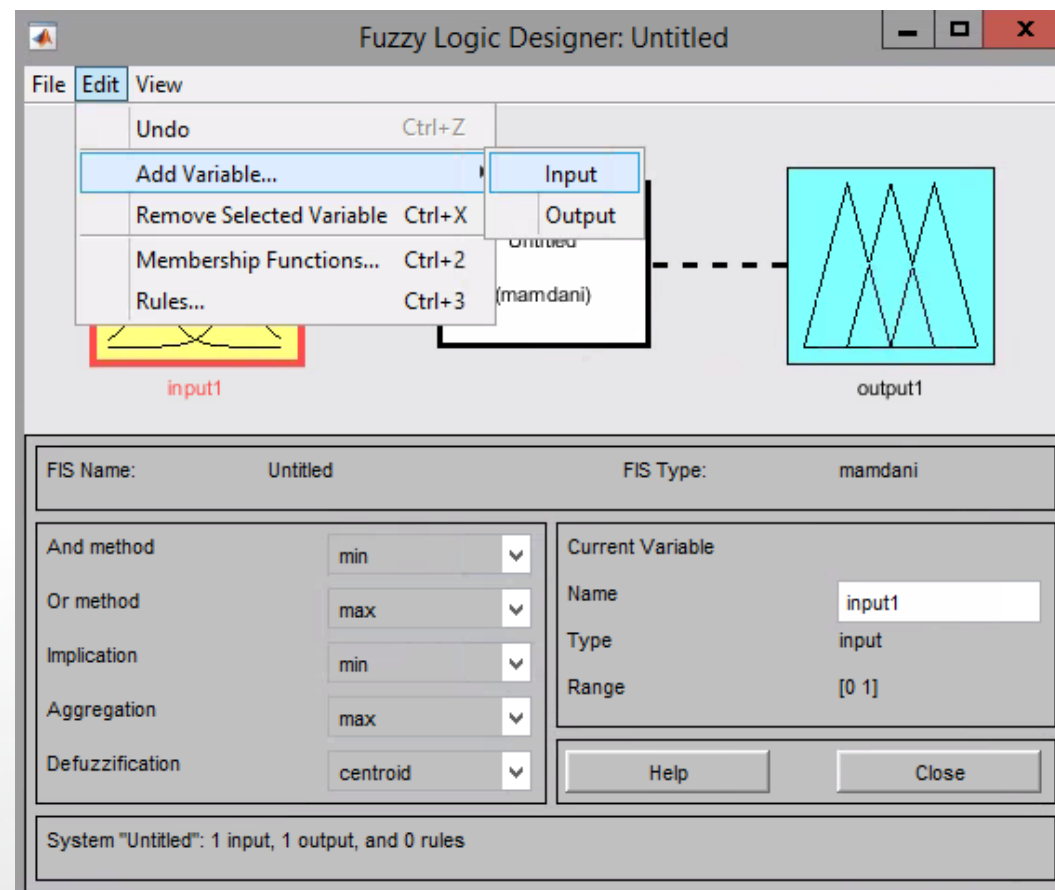
MATLAB Fuzzy Logic Toolbox

- Start MATLAB, and type `'fuzzy'` on MATLAB Window.
- The Fuzzy Logic Designer will pop up.



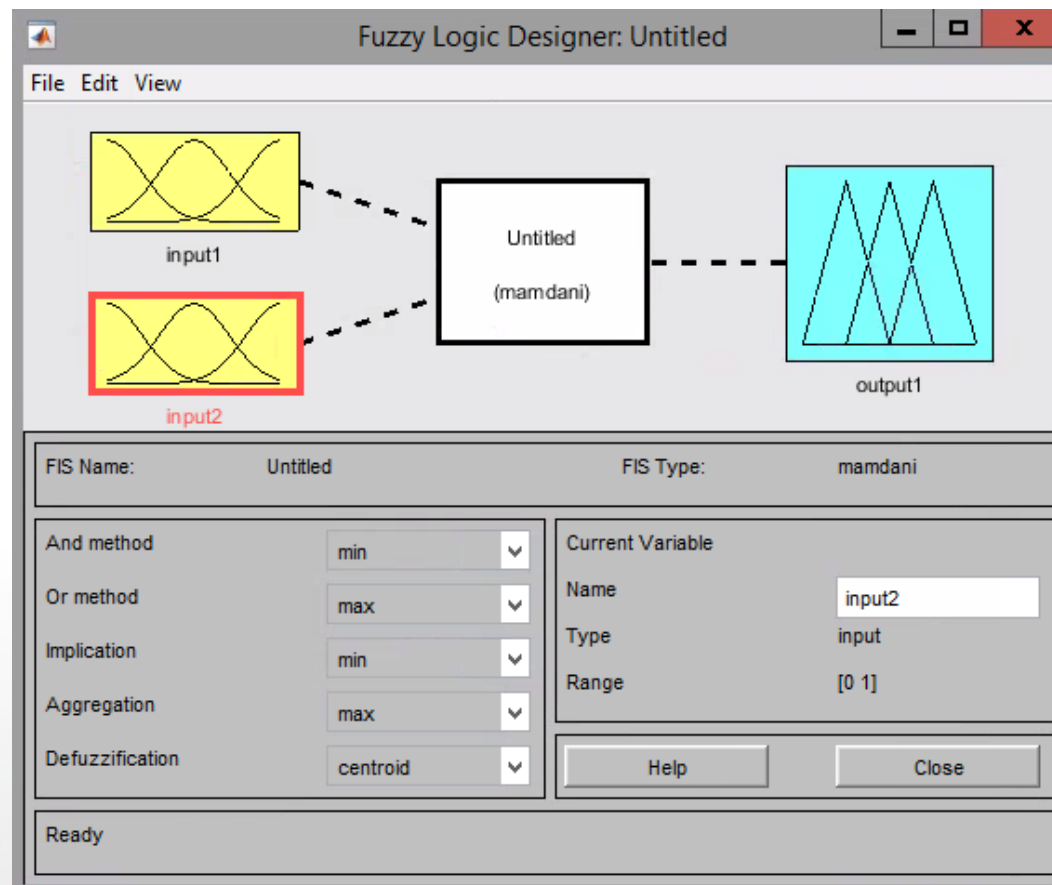
MATLAB Fuzzy Logic Toolbox

- We have two input fuzzy variables (Error and Δ error), therefore we proceed to add variables.:



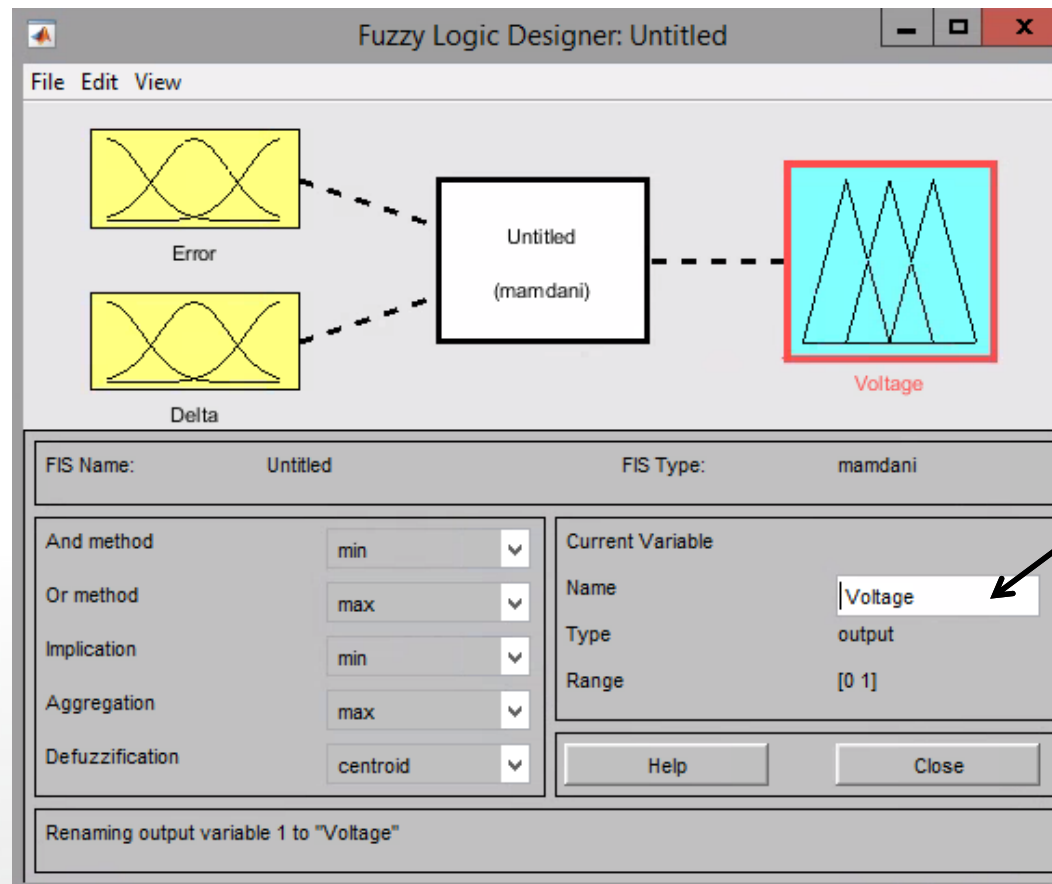
MATLAB Fuzzy Logic Toolbox

- After adding the variable, we will see two input variables in the system:



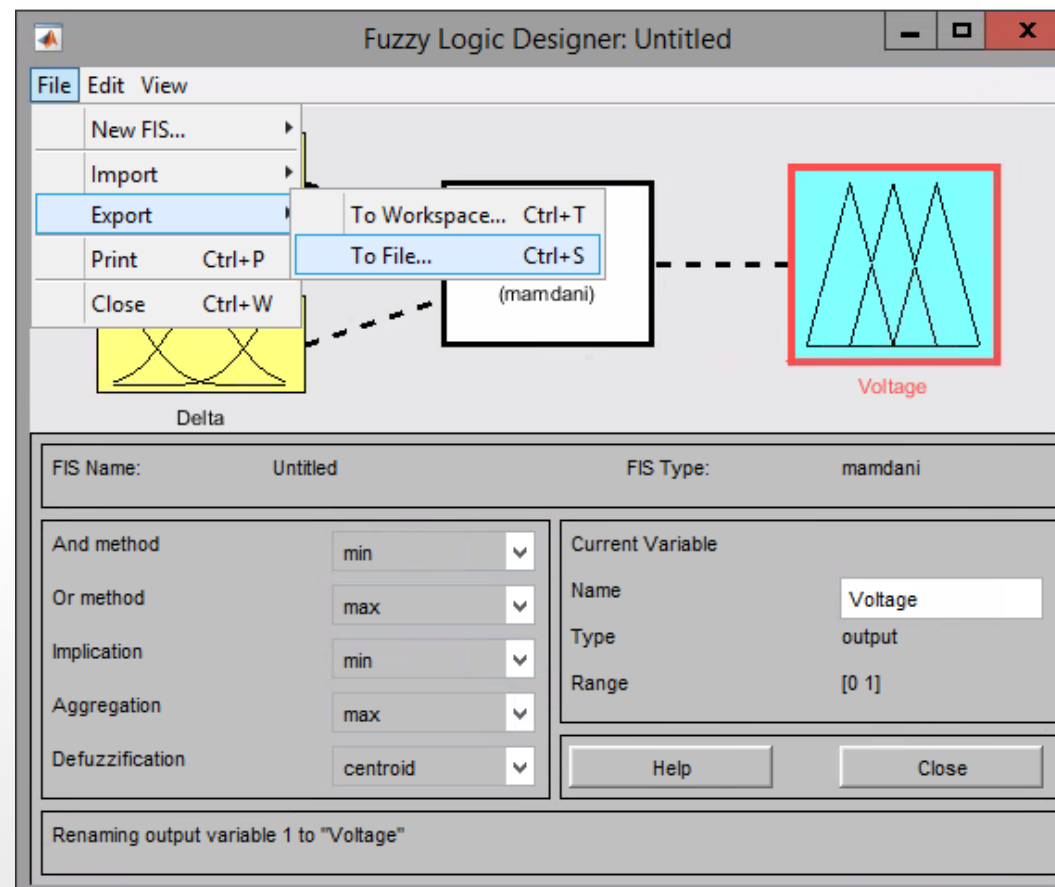
MATLAB Fuzzy Logic Toolbox

- We can also change the names of all the variables:



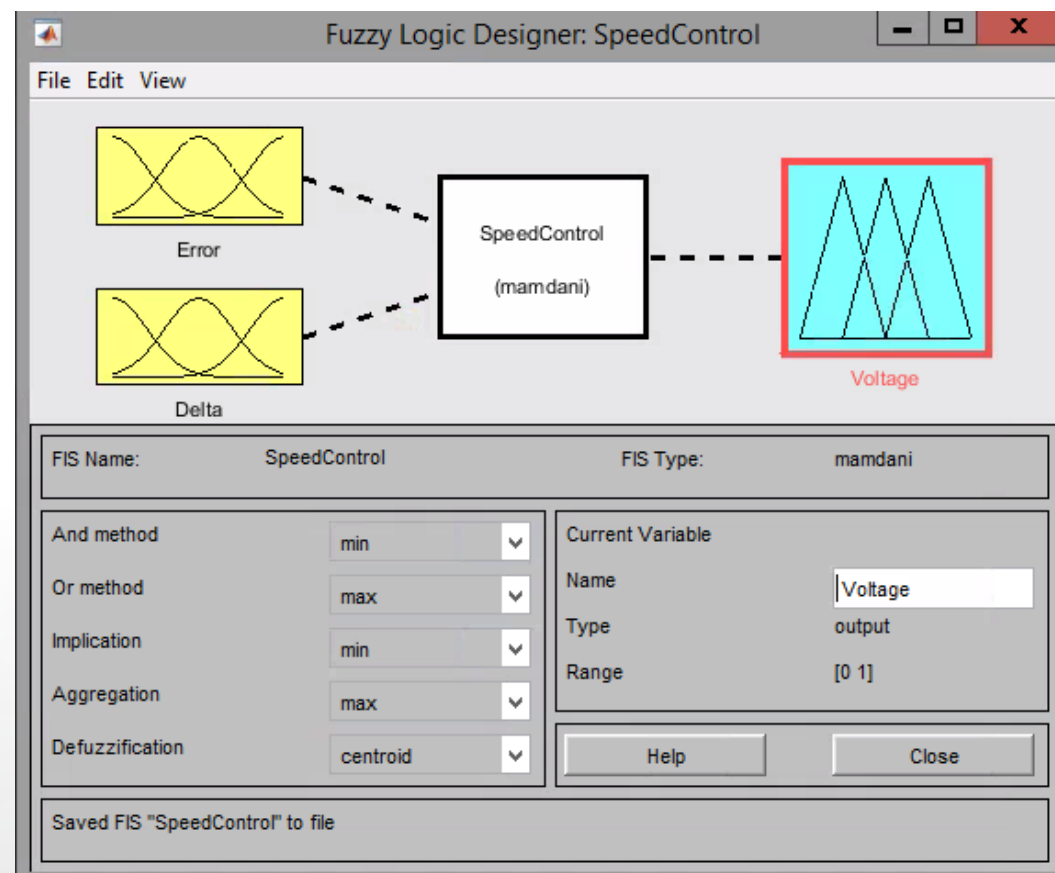
MATLAB Fuzzy Logic Toolbox

- Notice that the Fuzzy system is still untitled.
- To give it a name, export the system to file:



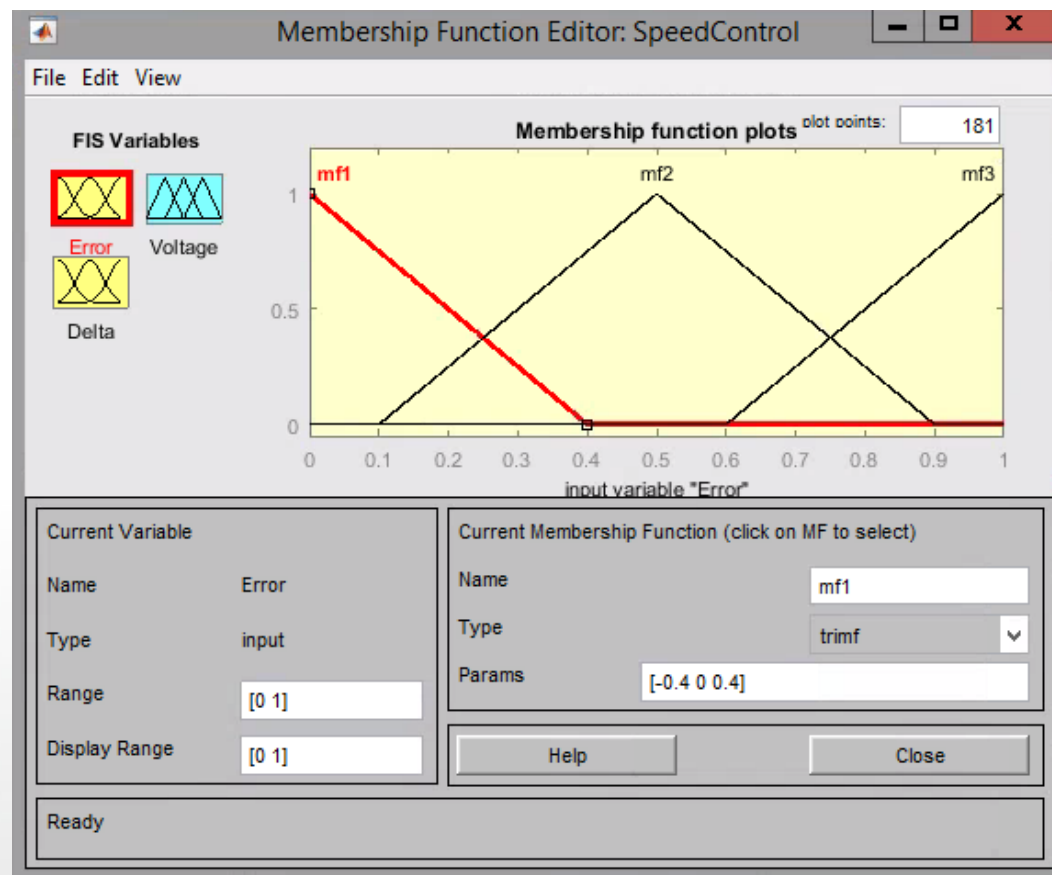
MATLAB Fuzzy Logic Toolbox

- In this example, we call it “SpeedControl.fis”.
- “*fis*” stands for Fuzzy Inference System.



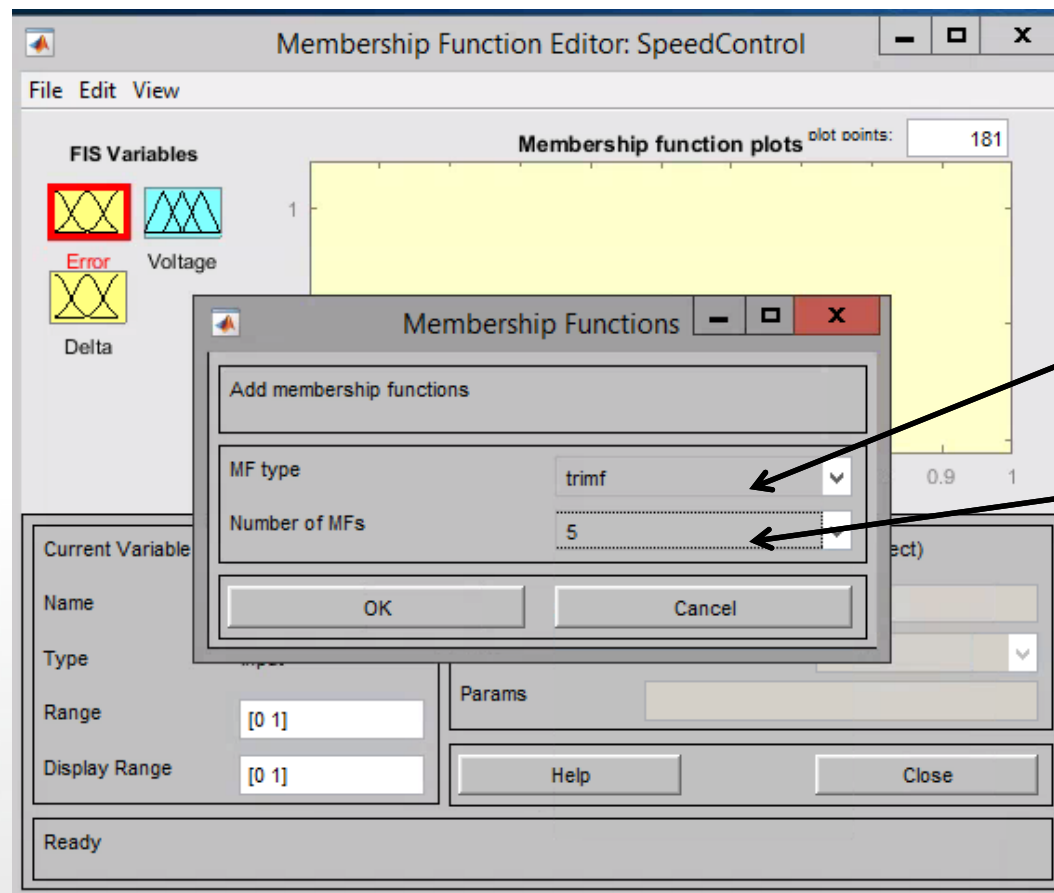
MATLAB Fuzzy Logic Toolbox

- Our next step is to edit the fuzzy sets.
- Double click on the “Error” variables, and a new window will pop up.



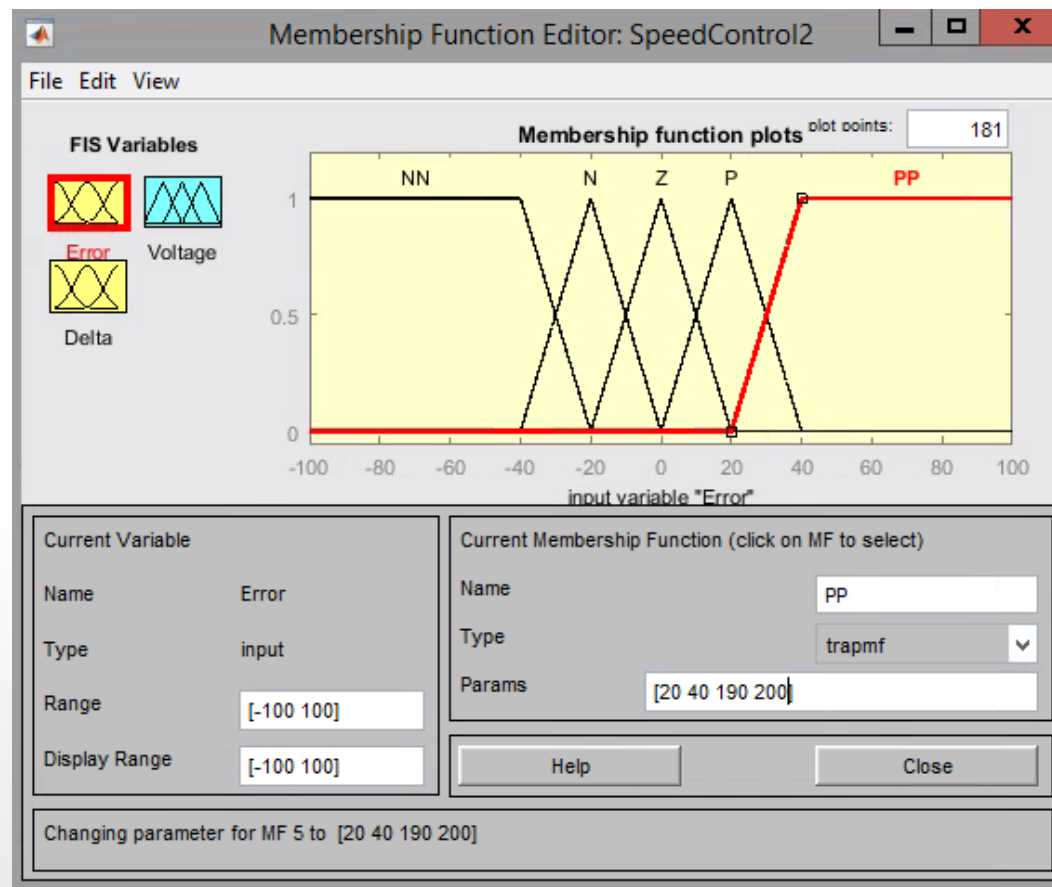
MATLAB Fuzzy Logic Toolbox

- We would like to define our own membership functions.
- Therefore, go to Edit, “Remove all MFs” and then “Add MFs”



MATLAB Fuzzy Logic Toolbox

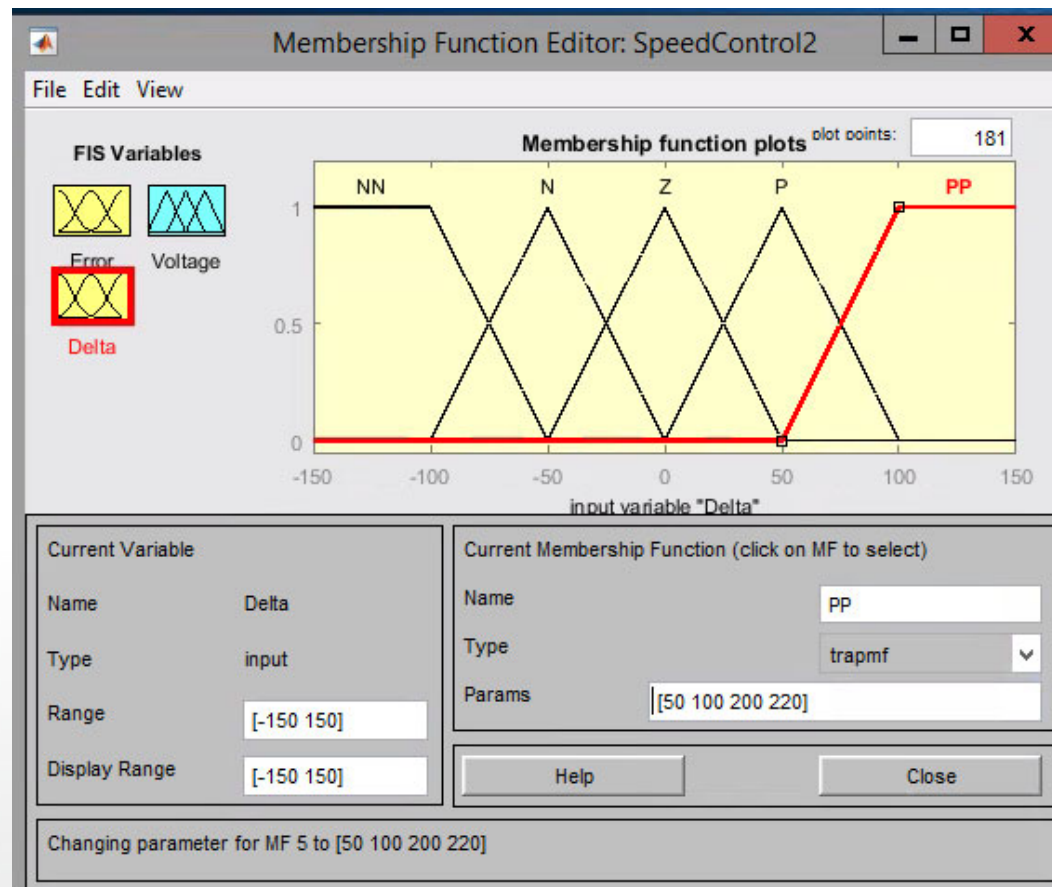
- Name all the membership functions (NN, N, Z, P, PP).
- Configure their centers and widths.



Note: NN and PP are changed to trapezoidal

MATLAB Fuzzy Logic Toolbox

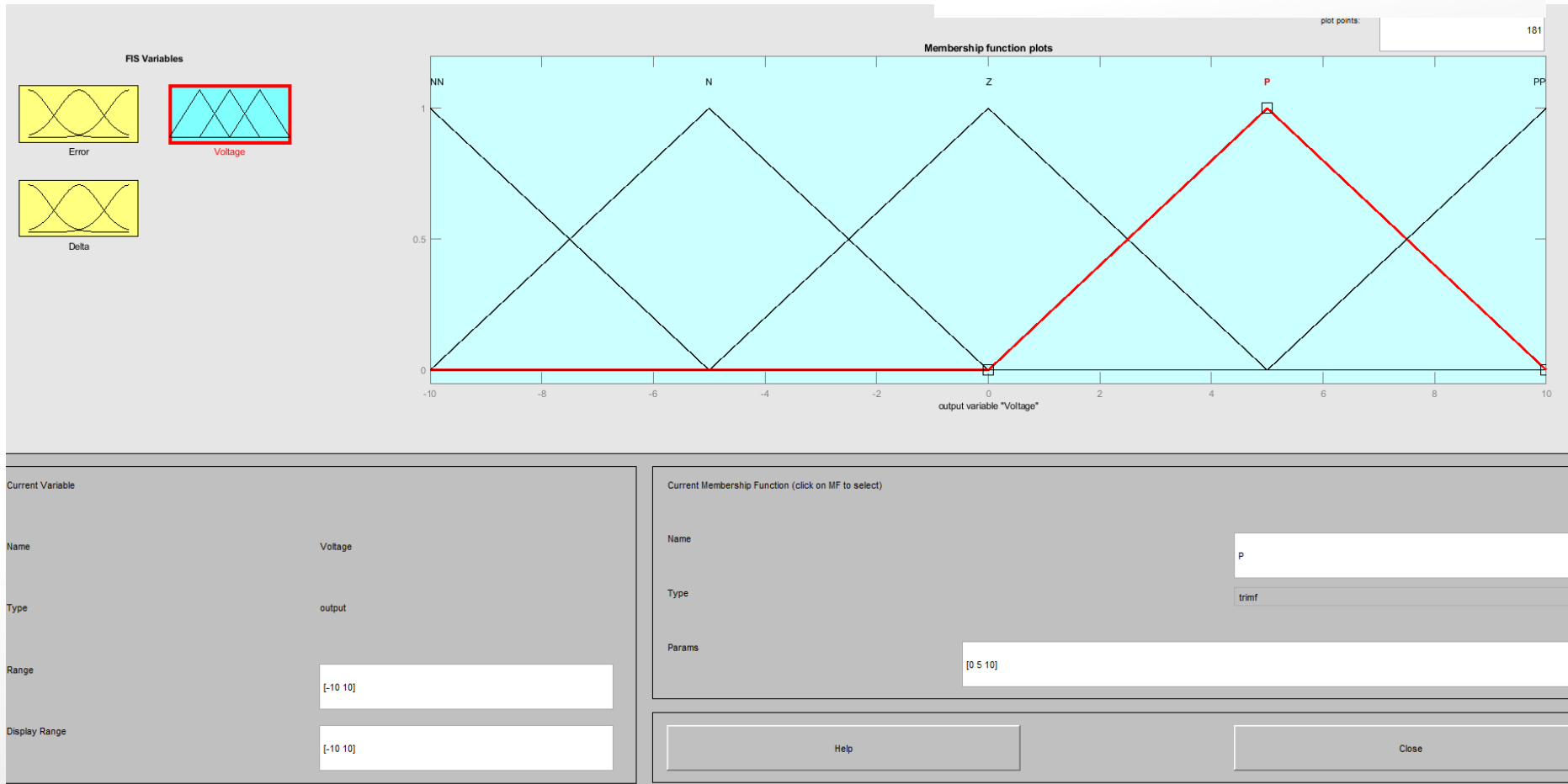
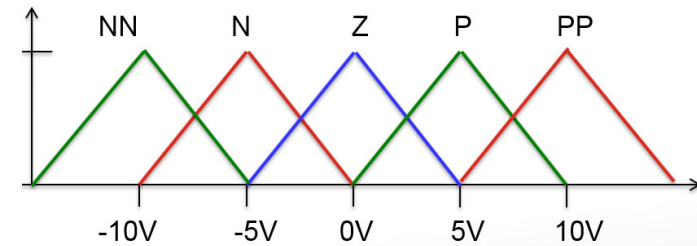
- We do the same for Delta...



MATLAB

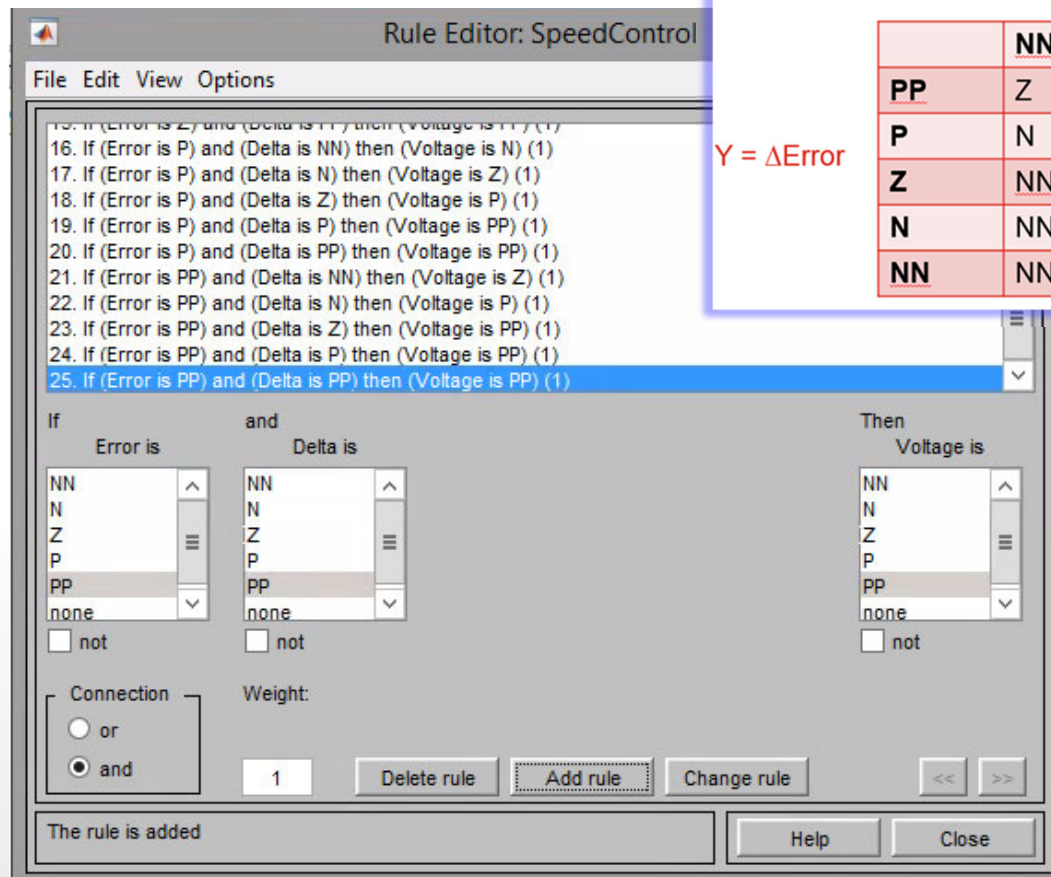
Fuzzy Logic Toolbox

- ... And also for Voltage



MATLAB Fuzzy Logic Toolbox

- The membership functions have been defined. Click close.
- Next, we should define the rules. Go to Edit → Rules.
- Change any settings and then “Add Rule”



Y = Δ Error

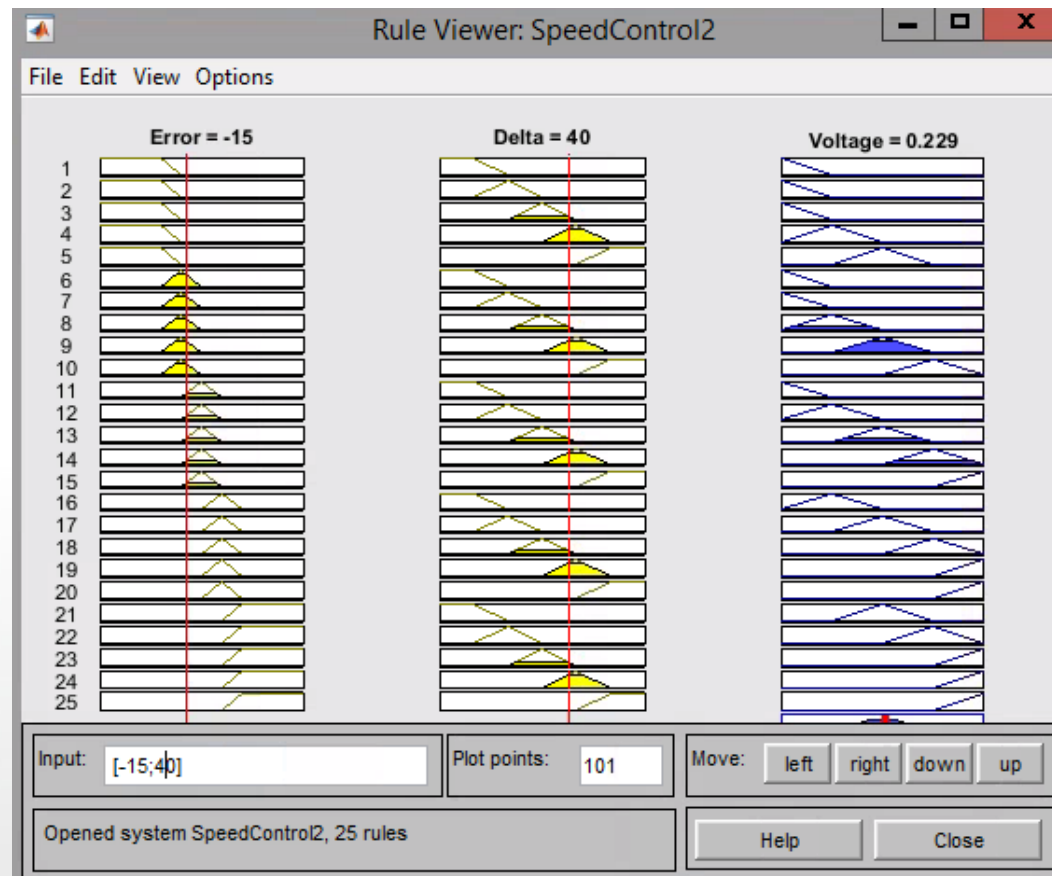
X = Error

	NN	N	Z	P	PP
PP	Z	P	PP	PP	PP
P	N	Z	P	PP	PP
Z	NN	N	Z	P	PP
N	NN	NN	N	Z	P
NN	NN	NN	NN	N	Z

Z =
Control
Actions

MATLAB Fuzzy Logic Toolbox

- When this is done, you can simulate / verify the results.
- Click View → Rule.
- Give the input values and see the output value. It is similar to our calculation.
- Finally, save and close the *fis*.



MATLAB Fuzzy Logic Toolbox

Reopen the *.fis* file

- Start MATLAB, and type 'fuzzy' on MATLAB Window.
 - The Fuzzy Logic Designer will pop up.
- Type in

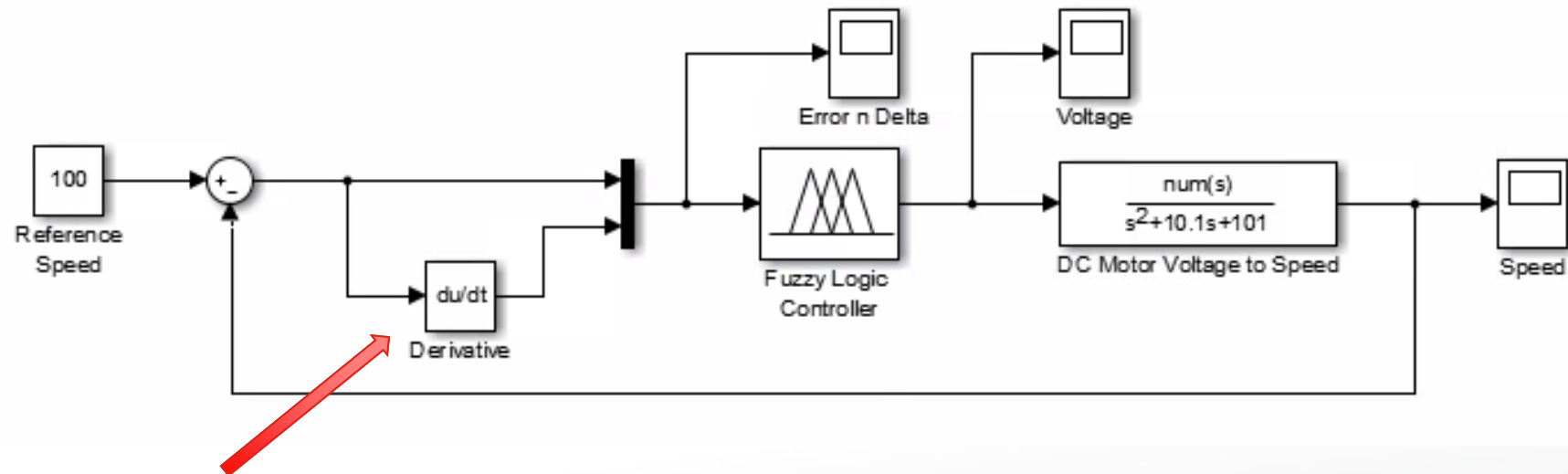
File

Import

From File

Simulink

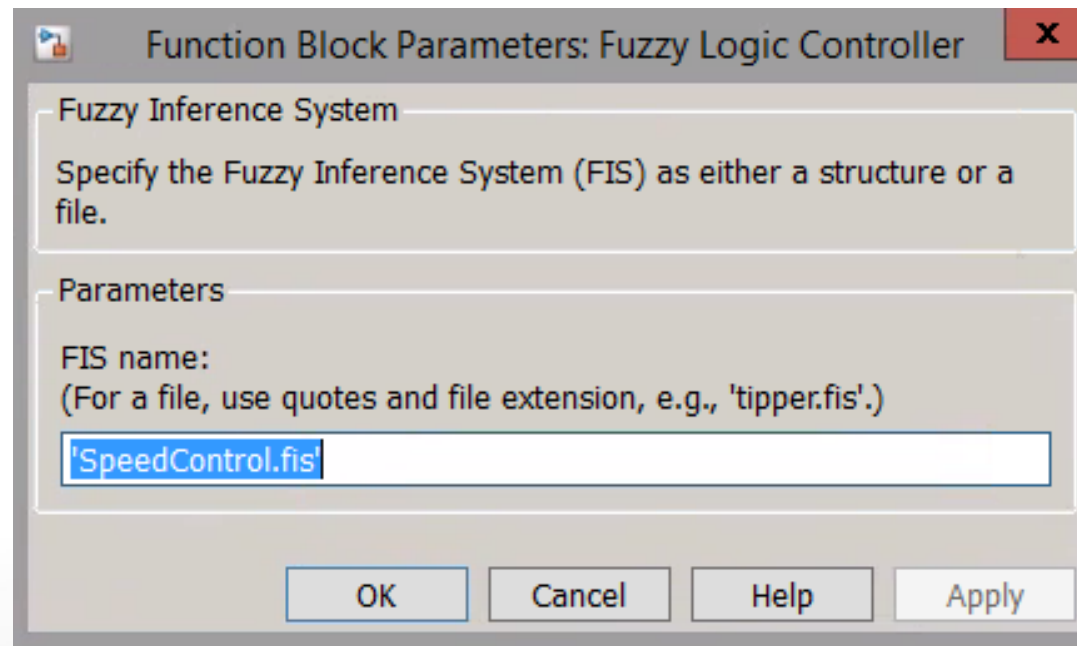
- Next, run Simulink and build the following block diagram:



- In this case is gradient of the first variable, but In other projects
- Could be other variable like for example **Force**
- The first one could be **Displacement**...or even more variables, i.e. **multiple variables, or attributes**

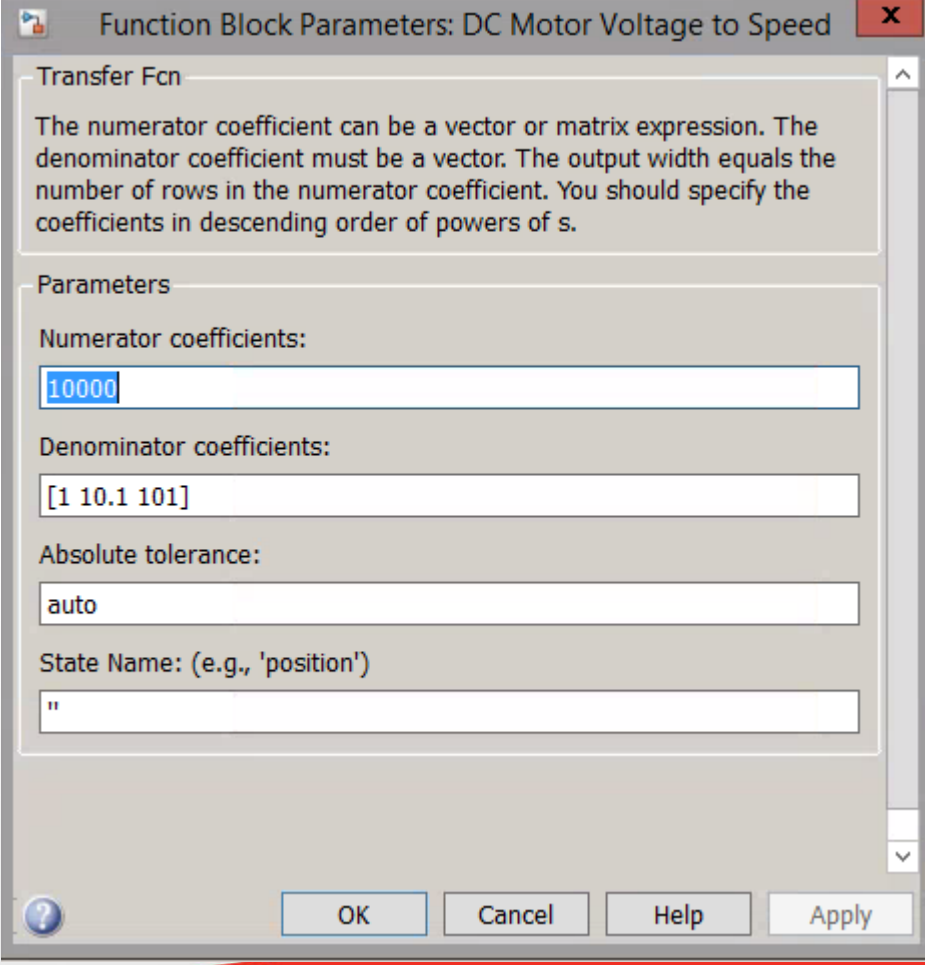
Simulink

- The Fuzzy Logic Controller is specified as follows:



Simulink

- The DC Motor is defined as follows: (Note: This is different from all our examples in the past)



Function Block Parameters: DC Motor Voltage to Speed

Transfer Fcn

The numerator coefficient can be a vector or matrix expression. The denominator coefficient must be a vector. The output width equals the number of rows in the numerator coefficient. You should specify the coefficients in descending order of powers of s.

Parameters

Numerator coefficients:

10000

Denominator coefficients:

[1 10.1 101]

Absolute tolerance:

auto

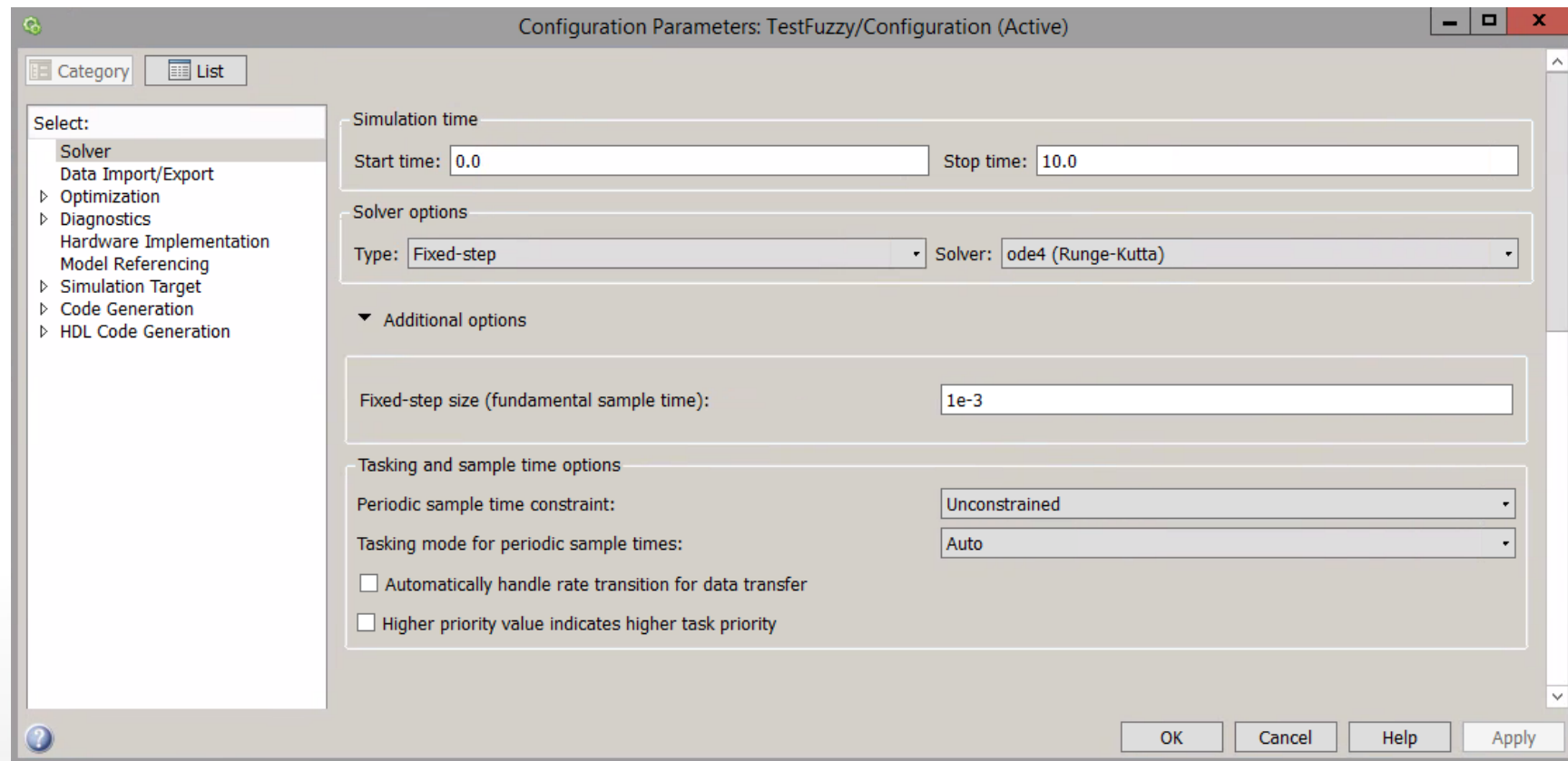
State Name: (e.g., 'position')

"

OK Cancel Help Apply

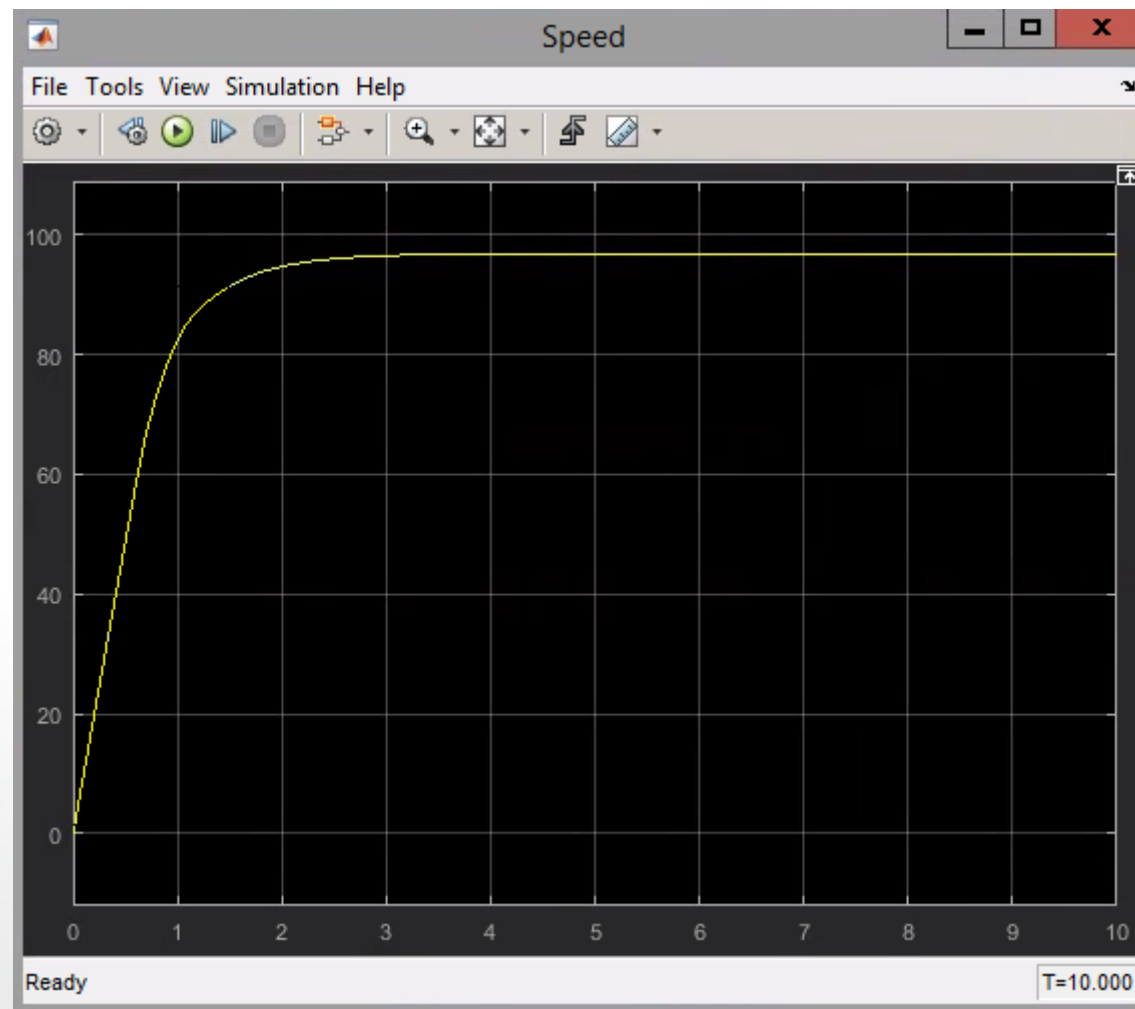
Simulink

- Finally, we configure the simulation parameters as follows:



Simulink

- Run the simulation and we will get the following response:



Transition to Electrical Vehicles Based on Multi-Attribute Decision Making

Publisher: IEEE[Cite This](#) [PDF](#)**2 Author(s)**Milan Todorovic ; Milan Simic [All Authors](#)**2**
Paper
Citations**25**
Full
Text Views

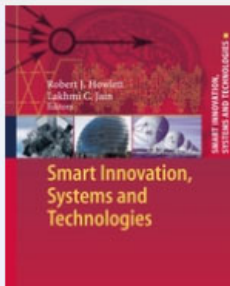
Abstract

Document Sections

- I. Introduction
- II. DECISION MAKING MODEL
- III. CHOICES
- IV. GOALS
- V. CASE STUDY

Abstract:

This paper presents application of multi-attribute decision making applied in a large public institution, to make strategic choice about new mobility technology. Electrical vehicles are becoming important parts of all vehicles product portfolios. Based on their advantages, many users are making transitions and replacing internal combustion engine vehicles with electrical vehicles. Considering a large number of vehicles, with various features, which are currently available on the market, and specific users' requirements, it is often required to apply a comprehensive decision making model in order to select the best solution. In the project presented here, decision making process and methodology used by the Institution, is highlighted. Final decision is based on the knowledge of the previous solution, already executed by one organization with similar operational requirements, and on inputs from internal departments which placed their preferences on various features of several models of electrical vehicles.

[KES International](#)[Journals](#)[Conferences](#)[KES Groups](#)[KES Portal](#)[Welcome](#)[Conference Scope](#)[General & Invited Sessions](#)[List of Invited Sessions](#)[Information for Authors](#)[-- Call for Papers](#)[-- Submission of Papers](#)[Keynote Speakers](#)[Dates and Deadlines](#)[International Programme
Committee](#)[Organisation](#)[KES Virtual Conference Centre](#)[Registration and Charges](#)[Registration FAQs](#)[Contact](#)[Terms and Conditions](#)[PROSE Paper Submission &
Review System Login](#)

INNOVATION IN MEDICINE & HEALTHCARE

**KES Virtual Conference
Centre
17-19 June 2020**

8th International KES Conference

Welcome!

The KES International Conference on Innovation in Medicine and Healthcare (KES-InMed-20) will gather a multi-disciplinary group consisting of researchers and engineers, managers, students and practitioners from the medical arena, to discuss the ways that innovation, knowledge exchange and enterprise can be applied to issues relating to medicine, surgery, healthcare and the issues of an ageing population.

A central theme of the conference will be Smart Medical and Healthcare Systems which will cover the ways in which modern intelligent systems contribute to the solution of problems faced by healthcare and medical practitioners today, addressing the application of these systems through all of the strands of the event.

Proceedings

KES InMed 2020

**SMART DIGITAL
FUTURES 2020**

[in](#) [Follow](#)[Follow](#)

About KES Journal

Welcome

Journal Topics

Call for Papers

Editorial Team

Editorial Review Board

Subscribe to the Journal -
Free to Silver Members

Frequently Asked Questions

Contact Details

IOS KES Journal Page

[Online Paper Upload,
Authors, Reviewers & Editors
Login](#)



KES Journal

International Journal of Knowledge-Based and Intelligent Engineering Systems

The **KES Journal** provides a medium for publishing the results of recent research into the applications, tools and techniques of Intelligent Systems, including a wide range of [Topics](#).

The content of the Journal is the responsibility of the [KES International](#) Research Organisation, which also presents a portfolio of well-attended annual conferences. The Journal is published by the major international publisher [IOS Press](#) which maintains its own [IOS Press KES-Journal Web Page](#).

Fewer than 25% of the papers submitted to the KES Journal are published - see the FAQs page for more detail.

Please Note: We are experiencing a very high number of submissions for the journal. We have recruited a number of new Associate Editors to help us clear the backlog that has built up. However, authors of new submissions can expect a long delay before their papers will be processed. Please do not submit a paper at the moment if you need fast publication.

Please do not email us to ask about the progress of a paper you have submitted as we are unable to respond to such requests. Instead log in to the PROSE review system to see the status of the paper. We are unable to provide more information about your paper than this.

KES Journal



Editors-in-Chief:
R.J.Howlett, UK
and
B.Gabrys, UK

General Editor:
M. Simic, Australia

Founder Editor:
L.C.Jain, Australia

Published by:
IOS Press, Netherlands
ISSN:1327-2314



About KES Journal

Welcome

Journal Topics

Call for Papers

Editorial Team

Editorial Review Board

Subscribe to the Journal -
Free to Silver Members

Frequently Asked Questions

Contact Details

IOS KES Journal Page

[Online Paper Upload,
Authors, Reviewers & Editors
Login](#)



Journal Topics

The aim of research into Knowledge-Based Intelligent Engineering is to develop systems that replicate the analytical, problem solving and learning capabilities of the brain. These systems bring the benefits of knowledge and intelligence to the solution of complex problems. Authors are invited to submit original unpublished work that is not currently under consideration for publication elsewhere. The topics covered by the Journal include the following:-

Generic Intelligent Tools, Techniques and Algorithms

Knowledge-Based Systems, Expert Systems, Neural Networks, Fuzzy Techniques and Systems, Genetic Algorithms and Evolutionary Computing, Hybrid Intelligent Systems, Intelligent Agents and Multi-Agent Systems, Knowledge Discovery and Data Mining, Machine Learning, Cognitive Modelling, Knowledge Representation and Management, Planning, Spatial & Temporal Reasoning, Knowledge Acquisition.

Applications using Intelligent Techniques

Industrial Control and Monitoring, Fault Diagnosis, Robotics, Image Processing, Machine & Computer Vision, Medical & Diagnostic Systems, Financial & Stock Market Monitoring and Prediction, Speech Processing and Synthesis, Natural Language Processing, Environmental Monitoring, Power Electronics & Drives, High Voltage Systems, Engine Control and Vehicle Applications, Intelligent Signal Processing and Wavelets.

Emerging Intelligent Technologies:

Artificial Intelligence and the Internet, Information Agents on the Internet, E-commerce/E-business and E-learning, Intelligent Information Retrieval, Intelligent Web Mining & Applications, Intelligent User Interfaces, Bioinformatics using Intelligent & Machine Learning Techniques, Intelligent Tutoring Systems, Virtual Reality & Multi-Media Intelligent Information Systems, Artificial Life, Ant and Swarm Algorithms.

Course Experience Survey

Advanced Mechatronics System Design – MANU2451

All Students

Thank you for your attention and cooperation over the past semester!

I wish you all the best for your final quiz
and other assessments and
have a good semester break

Revision and Quiz Discussion

Advanced Mechatronics System Design – MANU2451

Dr Chow Yin LAI

Edited by Dr Milan simic

School of Engineering

RMIT University, Victoria, Australia

Email: milan.simic@rmit.edu.au