Attached Lecture : Robot Programming and Control Algorithms for Labs

1 Introduction

2 The programming environment

3 Real-time languages for robots

4 Computer control architectures

5 Three basic control principles

6 PID control algorithm

7 Fuzzy control algorithm

8 Summary

1 Introduction - Robot Programming & Control

1 Introduction - Performance Specifications

Motion commands

- Motion commands are sent to individual controllers.
- Each controller has a feedback control loop.

Concurrency

- There are different time requirements in the task hierarchy.
- Control program is divided into many independent processes.
- 1) parallel processing (multi-processors)
  2) multitasking Operating System
  3) concurrent language (time slicing)

Inter-process Communication

- Concurrent processes in a robot can not run in isolation.
- Three ways: shared memory, remote procedure call, and message passing

Synchronisation

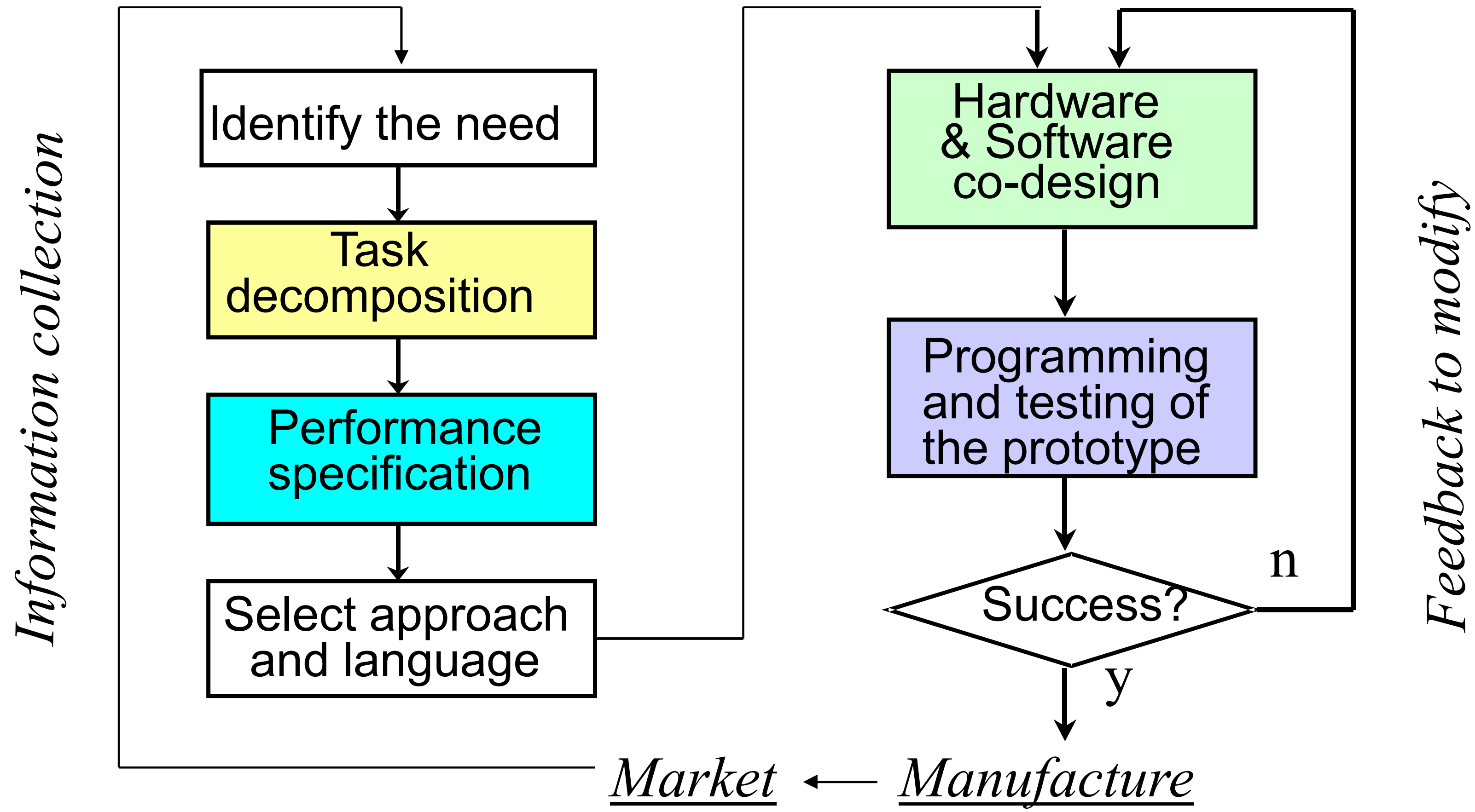
- Robots need to be synchronised with external events, e.g. initiation events, termination events, error events.

1 Introduction - Performance Specifications

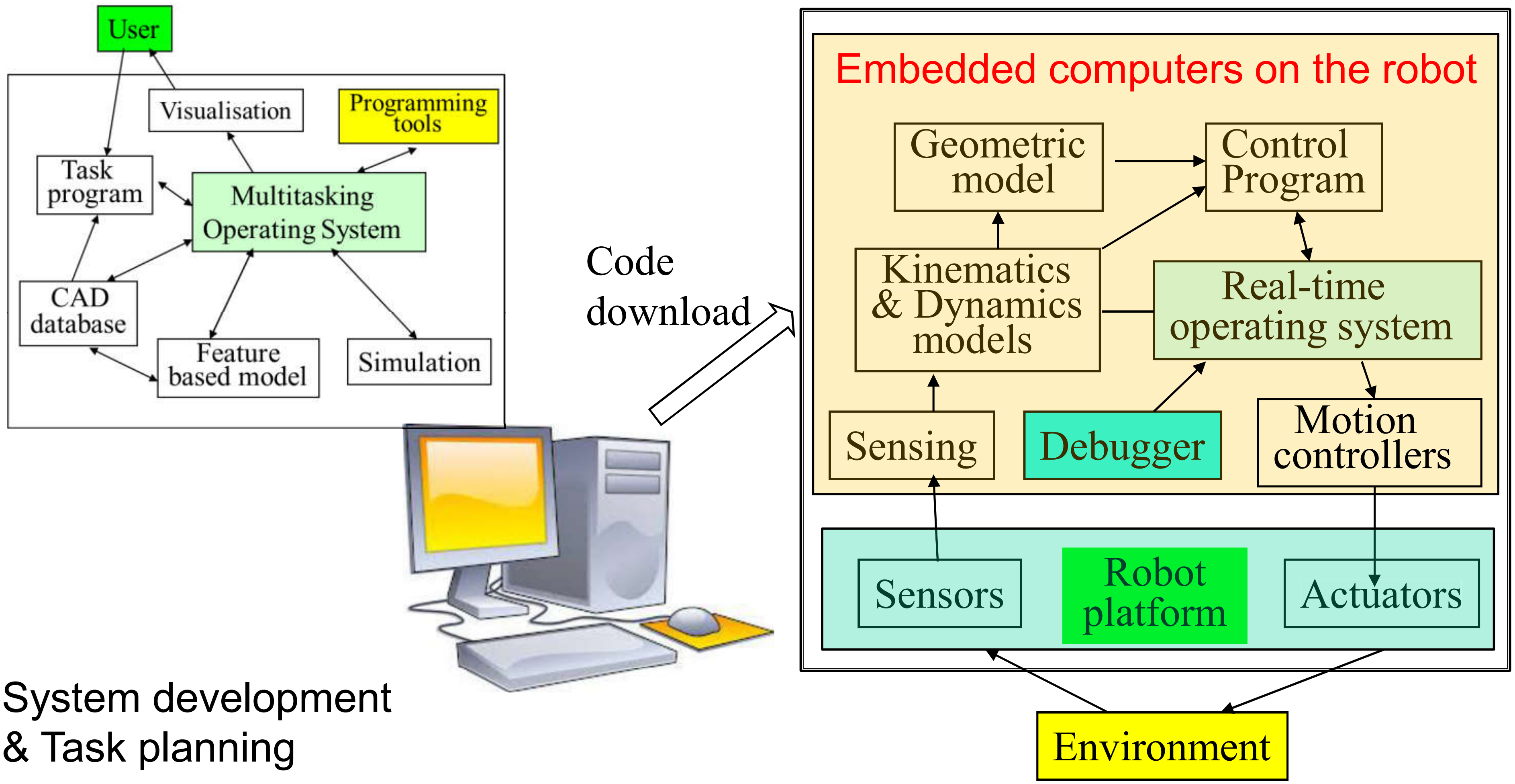
Polling & Interrupt	• The robot needs to be synchronised with external events, e.g. initiation events, termination events, error events. • Two event detection methods: ○ Polling processes continually check the sensor input. ○ Interrupt routines set a flag & handle the trigger events.
Sensor variables	• They are used to store the sensor data. • Sensor data may different data structures: a single bit, a binary number or an array.
Initialisation	• To place a robot in a known home position and the sensors such as incremental encoders should be zeroed.
Termination	• Control programs should detect possible failure situations and abort gracefully. • Watchdog timers are used to detect failures for graceful abort.

2 The Programming Environment for Robots

The development process



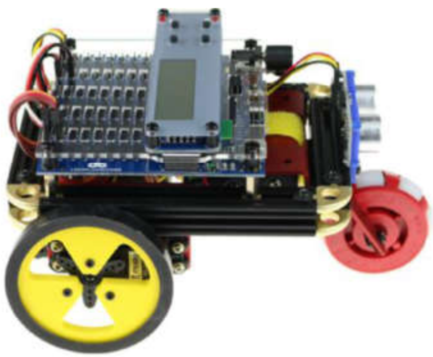
2 The Programming Environment for Robots

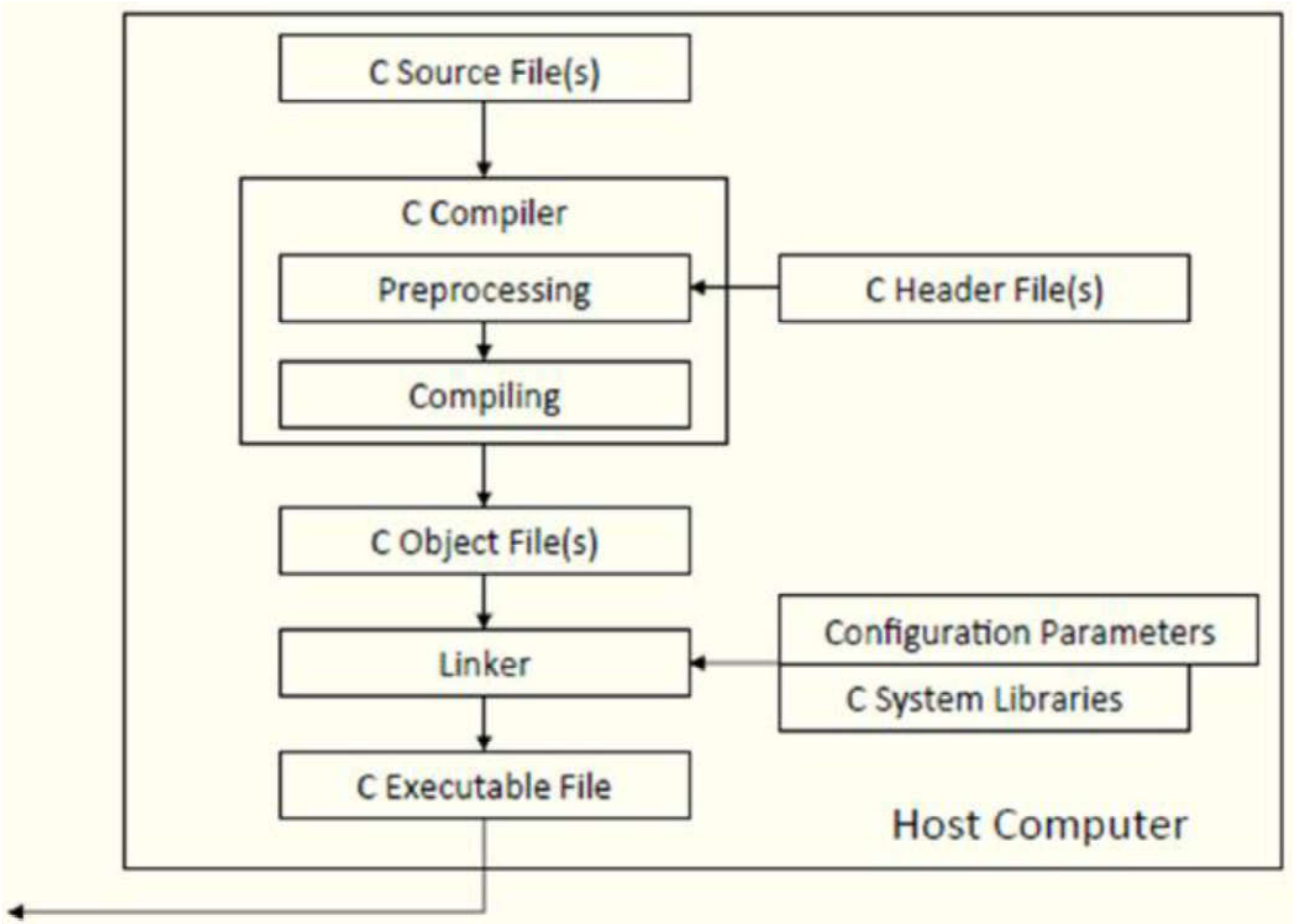


System development
& Task planning

2 The Programming Environment for Robots

- C Language is widely used for robot programming.





2 The Programming Environment for Robots

<u>Programming industrial robot</u>

- Online
- Offline

3 Real-time Languages

Scientific language features

- Variables & constants initialisation
- Data type -- points & structure
- Control structures, program layout and syntax
- Scope and visibility rules
- Modularity & compilation methods
- Exception handling

Real-time language features

- All features in scientific languages
- Construction of modules
- Creation and management of tasks
- Handling of interrupts & devices
- Inter-task communication
- Mutual exclusion

3 Real-time Languages

High-level scientific languages

- CORAL - (Woodward, *et al.*, 1970)
- RTL/2 (Barnes 1976)
- Ada (Yong 1972)
- ARGUS (liskov & Scheifler, 1983)
- KONIC (Kramer, *et al.*, 1983)
- CSP (Hare, 1978)
- CUTLASS (Bennett&Linkens,1984)
- FORTH (Brodie, 1986)
- Modula-2 (Wirth, 1982)
- OCCAM (Burns, 1988)
- PEARL/SR (Andrews, 1981/1982)
- C/C++ (Kernighan&Ritchie, 1985)
- Python (Guido van Rossum, 1991)
- Java/Real-time Java (1997)

Assembly languages for real time

- Assembly languages are processor dependent
- They are widely used for small real-time systems.
- Advantages are:
  - efficient use of CPU
  - easy access interface
  - support interrupts
- Disadvantages:
  - time consuming
  - difficult debugging
  - poor portability

3 Real-time Languages

Example: C and Assembly Languages

C		Assembly

x = y + 133;	⟺	MOVE R1, (y) ; Get the value y into R1
		ADD R1, 133 ; Add 133
		MOVE (x), R1 ; Save the result in x

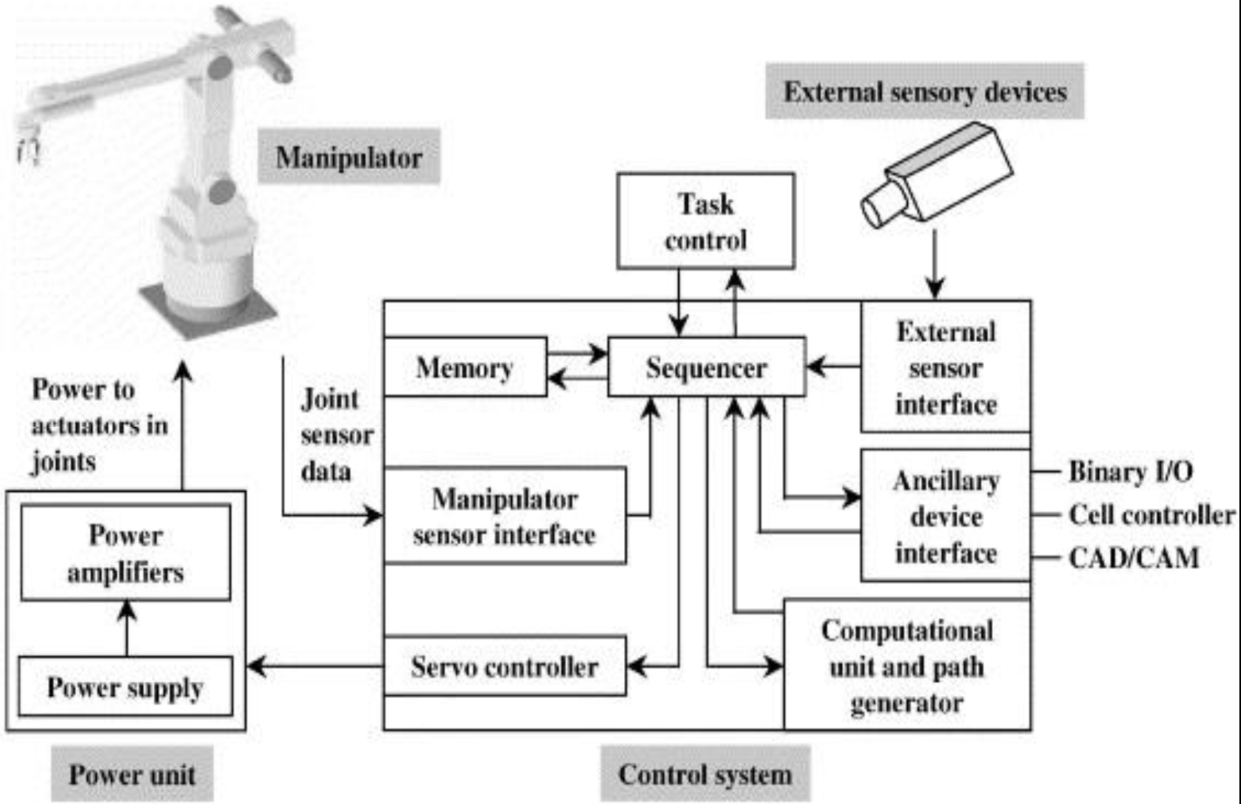
If (x >= z)	⟺	MOVE R2, (z) ; Get the value z SUTRACT R1, R2; Subtract z from x JCOND NEG, L101; Skip if the result is negative

W = sqrt (z)	⟺	MOVE R1, (z) ; Get the value z into R1
		PUSH R1 ; Put the parameter on the stack
		CALL SQRT ; Call the sqrt function
		MOVE (w), R1 ; The result return to R1
		POP R1 ; Throw away the parameter

4 Computer Control Architectures

1. Centralised control

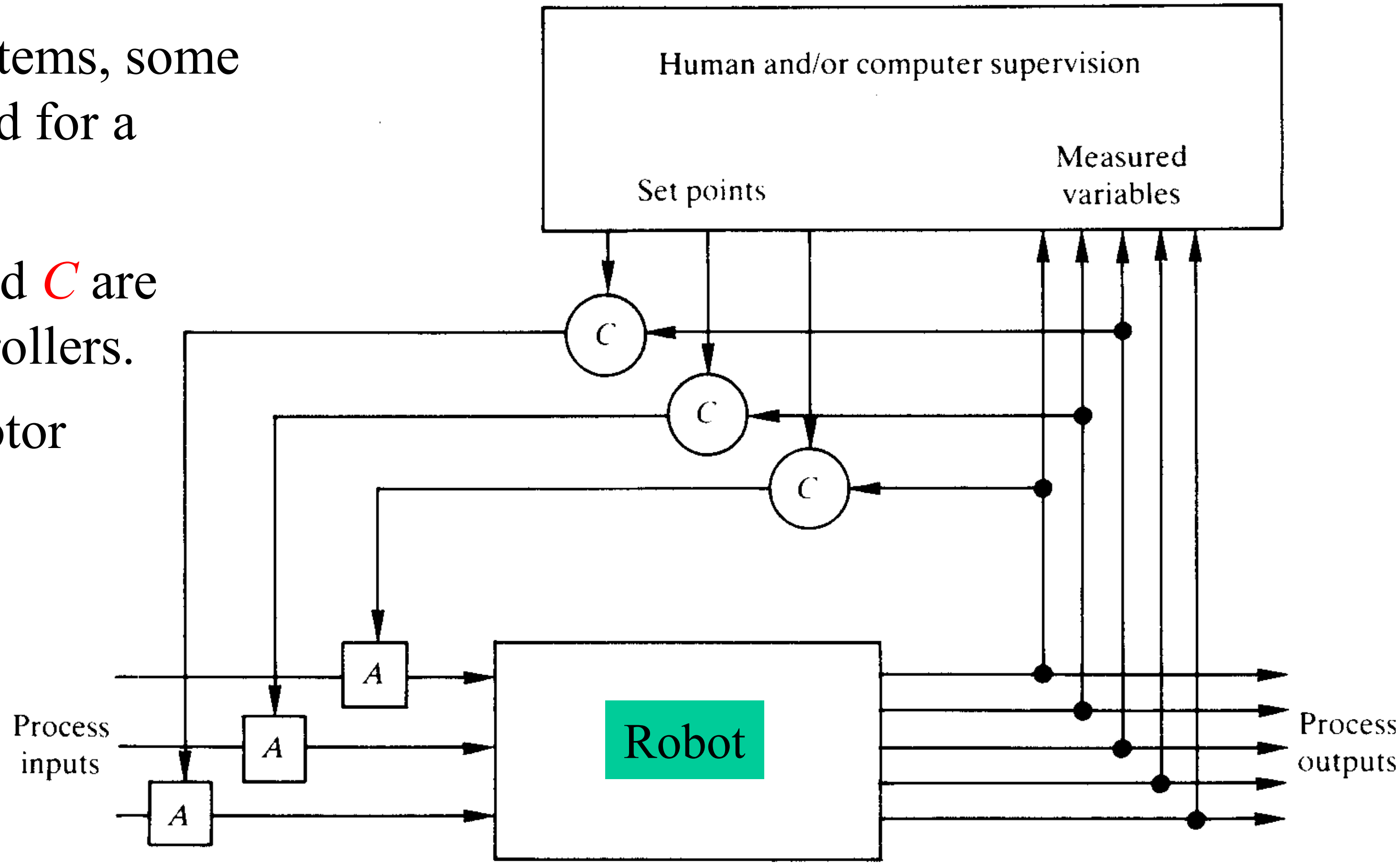
- Centralised computer control is very popular in 1960-1980, as computers were very expensive.
- A centralised computer has to deal with many different types of tasks and operations.
- Difficult programming, testing and debug.



4 Computer Control Architectures

2. Supervisory control

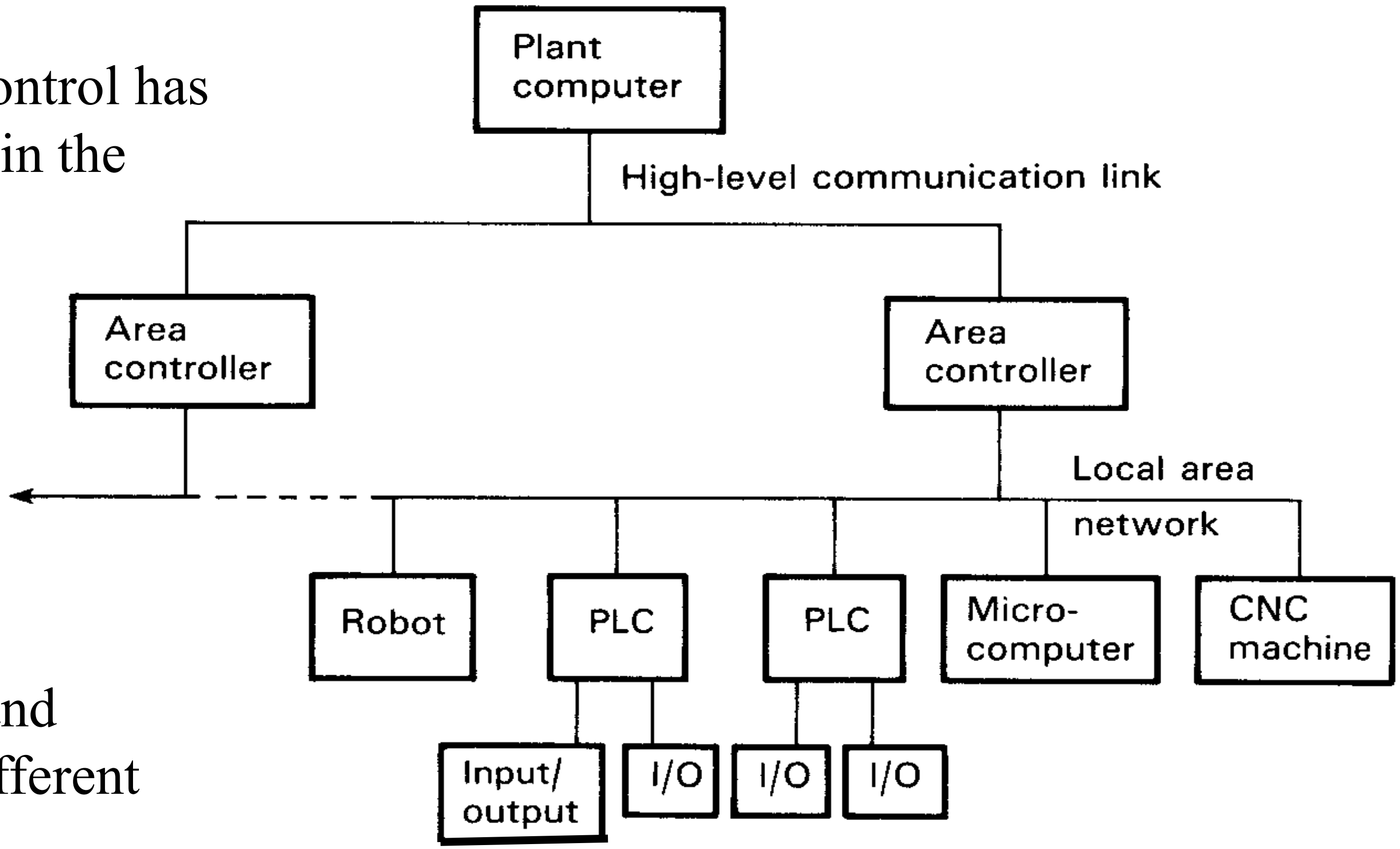
- In many robot systems, some computers are used for a supervisory role.
- The circles labelled *C* are the feedback controllers.
- The A means a motor or a actuator.
- The supervisory computer adjusts set points to the feedback controllers.



4 Computer Control Architectures

3. Hierarchical control

➢ The hierarchical control has been very popular in the computer control.



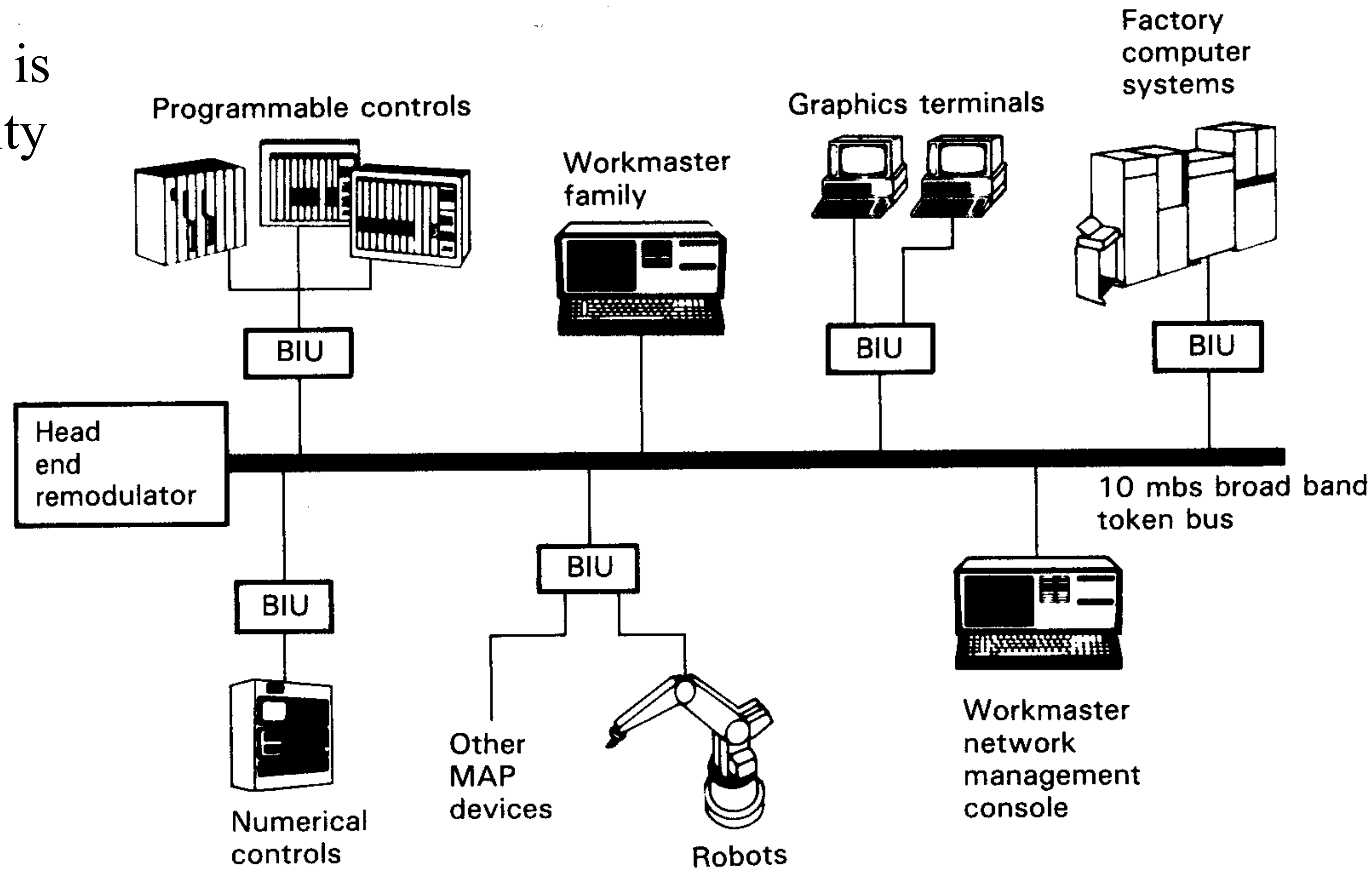
➢ Tasks are divided and implemented by different computers.

4 Computer Control Architectures

4. Distributed control

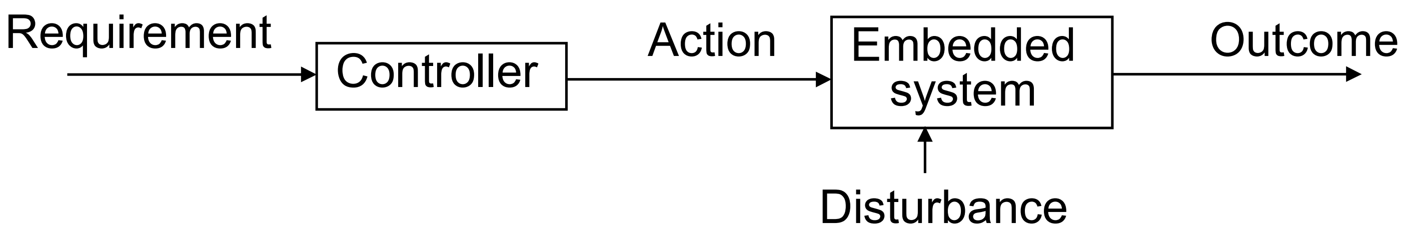
Many fieldbuses: CAN & LonWorks

- Distributed control is to achieve scalability and reliability.
- Many computers perform similar tasks in parallel.
- High-bandwidth data highway is necessary.

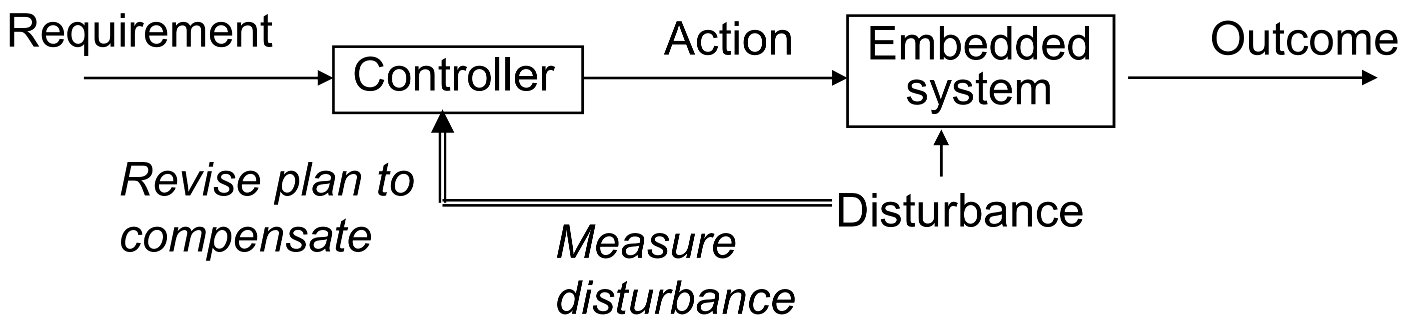


5 Three Basic Control Principles

1. **Open-loop control** -- a simplest form & predefined strategy



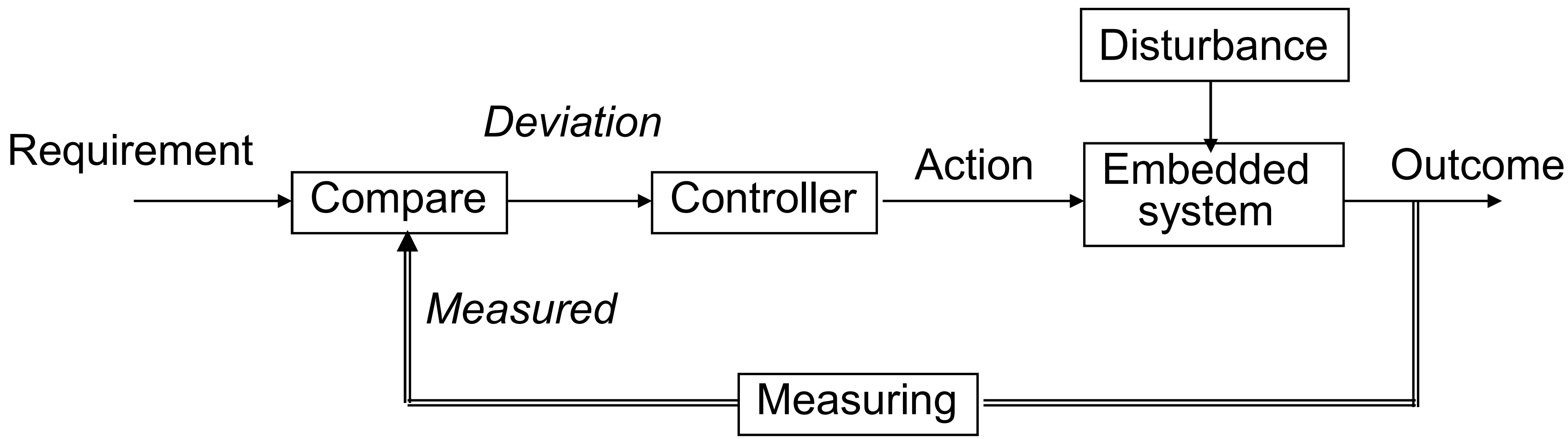
2. **Feedforward control** -- monitor & compensate disturbance



5 Three Basic Control Principles

3. <u>Close-loop or Feedback control</u>

- measuring disturbance effect
- calculating a correct action to minimise deviation.



Deviation = requirement - measured value

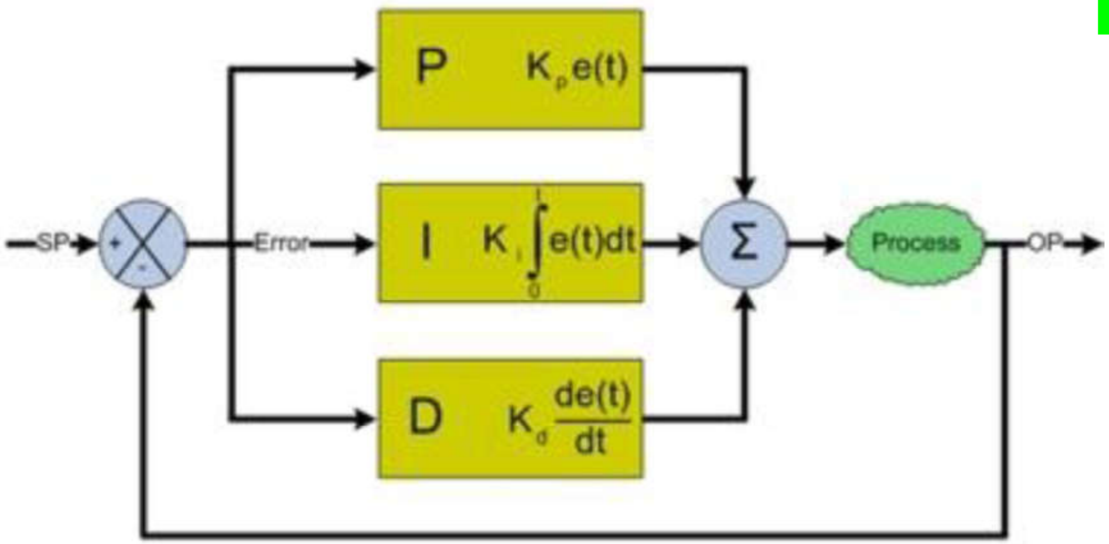
6 PID Control Algorithm

Control law: PID (Proportional+Integral+Derivative) control algorithm

- A well-established analogy control algorithm

e(t)=r(t)-f(t) -- error or deviation;

r(t) -- set points; *f(t)* -- measured value



Differential equation:

$$V_o(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt}$$

The discrete PID:
(difference equation)

V o

$$V_o = k_p e_n + K_i \sum_{i=1}^{n} e_i + K_d(e_n - e_{n-1})$$

where *n* is the present sample
n-1 is the previous sample

6 PID Control Algorithm

- ***Proportional action*** is the simplest form of control and produces an output signal directly proportional (by an amplifying factor $\boldsymbol{K_p}$) to the error input.

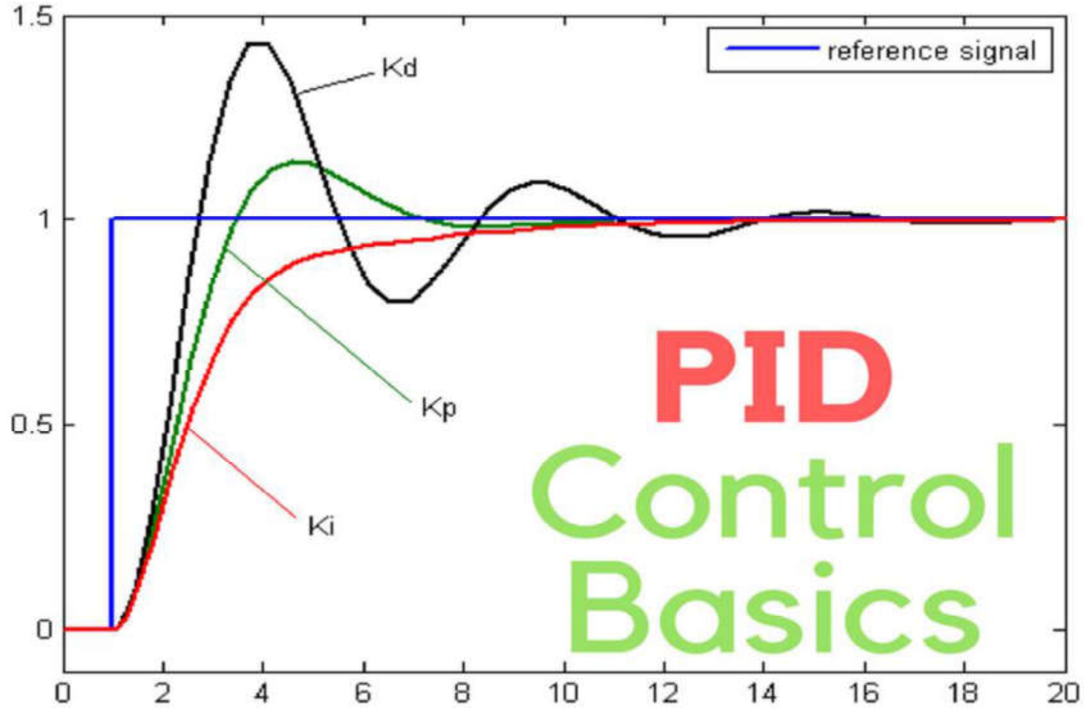
Note that:

Choosing the value of K involves a compromise:

- A high value of K gives a small steady-state error and a fast response, but the response may be oscillatory and the system is unstable.
- A low value gives a slow response and a large steady-state error.

6 PID Control Algorithm

- ***Integration action*** generates an output signal proportional to the mathematical integral of the error, K_i, i.e. the summed history of the error. By adding integral action, the system will have zero steady-state error.



- ***Deviation action*** provides an output signal proportional, K_d, to the change rate of the errors. It anticipates the error and reduce it quickly.

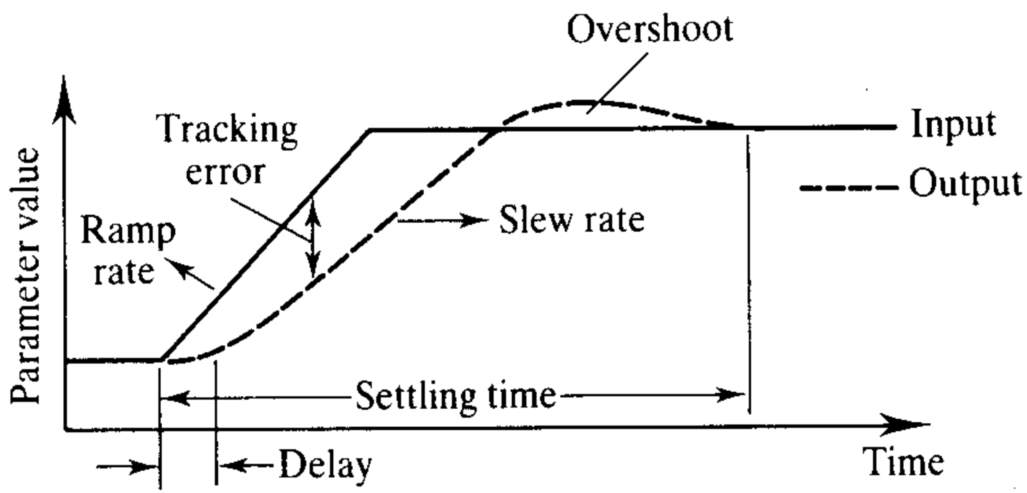
6 PID Control Algorithm

- ***Response delay***

It is time between the input change and the output starts to change, as shown in the following figure.

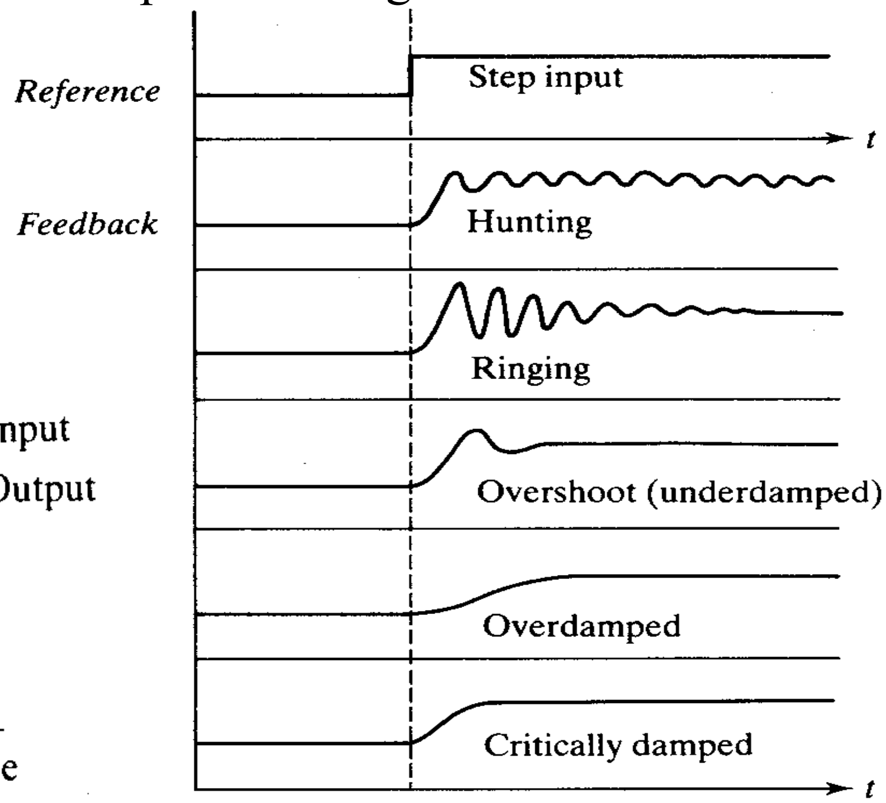
- ***Response time***

It is to measure the maximum slew rate of the system between 10-90%



- ***System stability***

A stable system should be critically damped. Hunting is unstable...



6 PID Control Algorithm

Example: A PID controller is used to control the speed of a car, which has the parameters: $K_p = 0.8$, $K_i = 0.5$ and $K_d = 0.4$.

$$V_o = k_p e_n + K_i \sum_{i=1}^{n} e_i + K_d(e_n - e_{n-1})$$

The speed of the car is set to be 50 Km/h for running 5 minutes. Its measured speed is 5 Km/h at the end of the 1st minute, 12 Km/h at the 2nd minute, 22 Km/h at the 3rd minute, 35 Km/h at the 4th minute, and 50 Km/h at the 5th minute. Calculate the controller output at the end of the 5th minute.

At the end of the 1st minute: e1=50-5=45; At the end of the 2nd minute: e2=50-12=38

At the end of the 3rd minute: e3=50-22=28; At the end of the 4th minute: e4=50-35=15

At the end of the 5th minute: e5=50-50=0

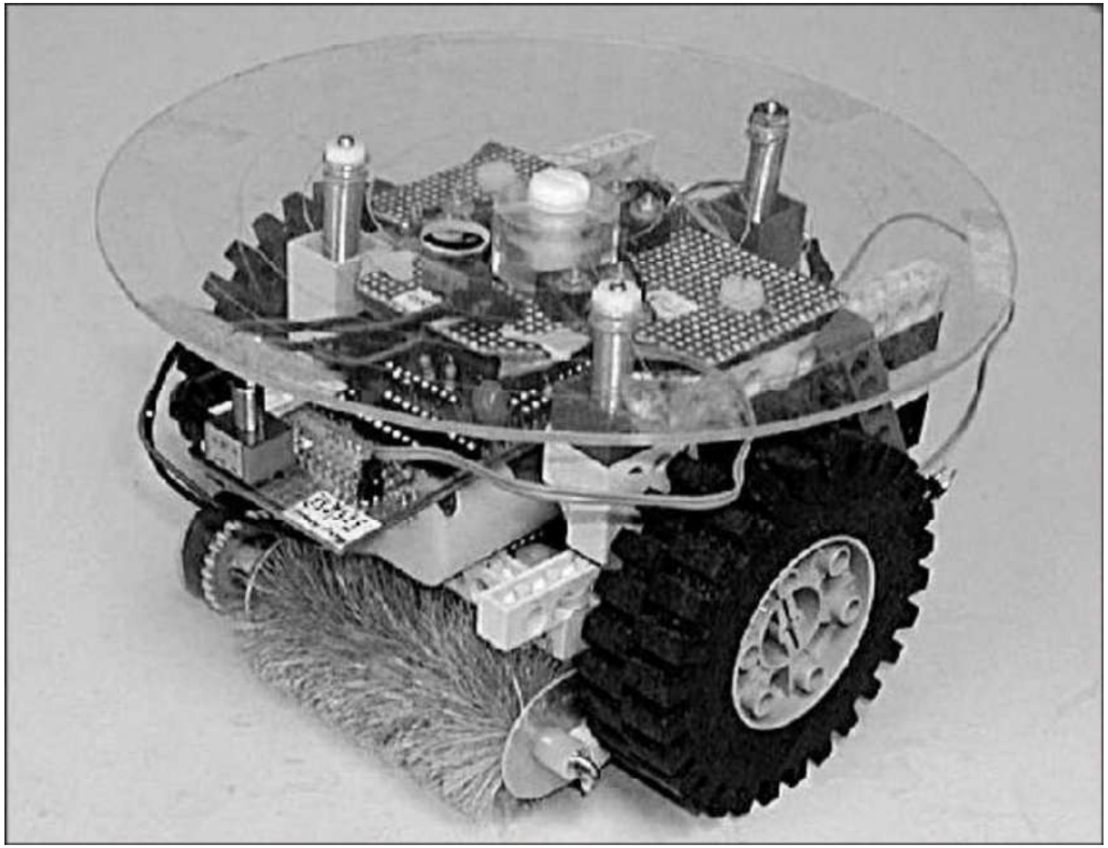
Controller Output =0.8*0+0.5*(45+38+28+15+0)+0.4*(0-15)= 0+63-6=57

6 PID Control Algorithm-Application

6 PID Control Algorithm-Application

Simple PI controllers for a differential-driving robot, Rug Warrior

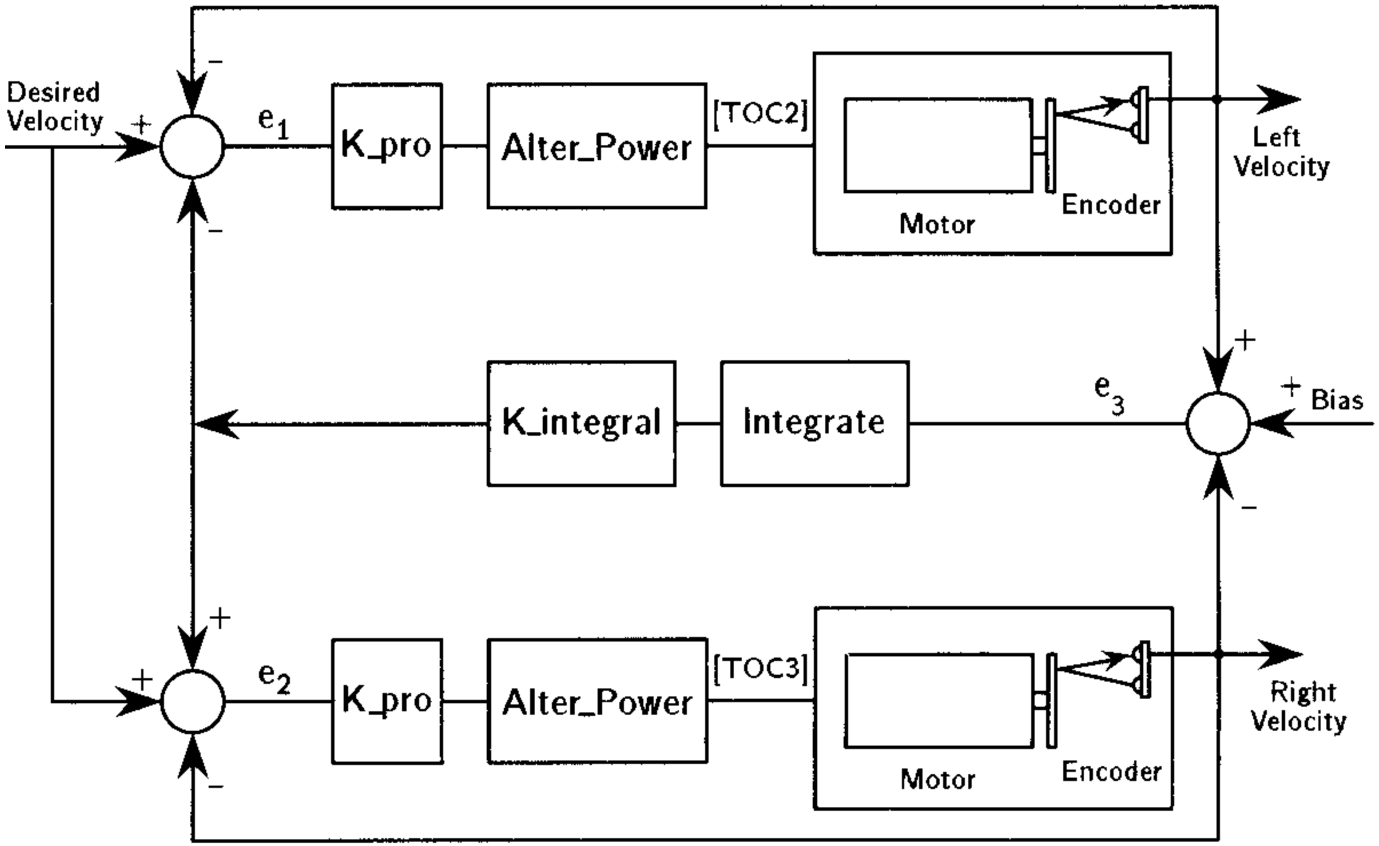
- There are two optic encoders used to measure the motor speed.
- Two proportional controllers are used to control two motors.
- An integration control, is used to synchronise two motors.



6 PID Control Algorithm-Application

Simple PI controllers for a differential-driving robot, Rug Warrior

- There are 3 control loops in this figure.
- Two proportional controllers are used to control 2 motors.
- An integration control, is used to synchronise two motors.
- Bias signal balances the difference of two motor speeds to achieve straight line motion.



6 PID Control Algorithm-Application

A simple Interactive C routines: *("Mobile Robots" by JL Jones & AM Flynn at MIT)*

```
float control_interval=0.25;   /* Run servo loop at this rate */
float des_vel_clicks = 0.0;    /* Desired velocity, clicks/interval */
float des_bias_clicks = 0.0;   /* Desired bias, clicks/interval */
float power[2] = {0.0,0.0};    /* Power command to motor */
float integral = 0.0;          /* Integral of velocity difference */
float k_integral = 0.10;       /* Integral error gain */
float k_pro = 1.0;             /* Proportional gain */
float k_clicks = 8.0 / 100.0;

/* set power level */
void alter_power(float error, int motor_index)
{ power[motor_index]=limit_range(power[motor_index]+error, 0.0, 100.0);
  pwm_motor(power[motor_index], motor_index); }

float integrate(float left_vel, float right_vel, float bias)
{ integral=integral+left_vel - right_vel + bias;
  return integral;}
```

6 PID Control Algorithm-Application

```
void speed_control ( )
{  float left_vel, right_vel, integral_vel, left_error, right_error;
   while (1)
    { left_vel = get_left_vel ( );              /* get current velocity using encoder readings */
      right_vel = get_right_vel ( );
      integral_error = k_integral * integrate(left_vel, right_vel, des_bias_clicks);
      left_error = k_pro * (des_vel_clicks - left_vel - integral_error);
      right_error = k_pro * (des_vel_clicks - right_vel + integral_error);
      alter_power(left_error, 0);
      alter_power(right_error, 1);
      sleep (control_interval);                 /* Run speed_control periodically */
    }
}

void start_speed_control ( )
{ init_velocity ( );
   init_pwm ( );
   start_process (speed_control ( ) );
}
```

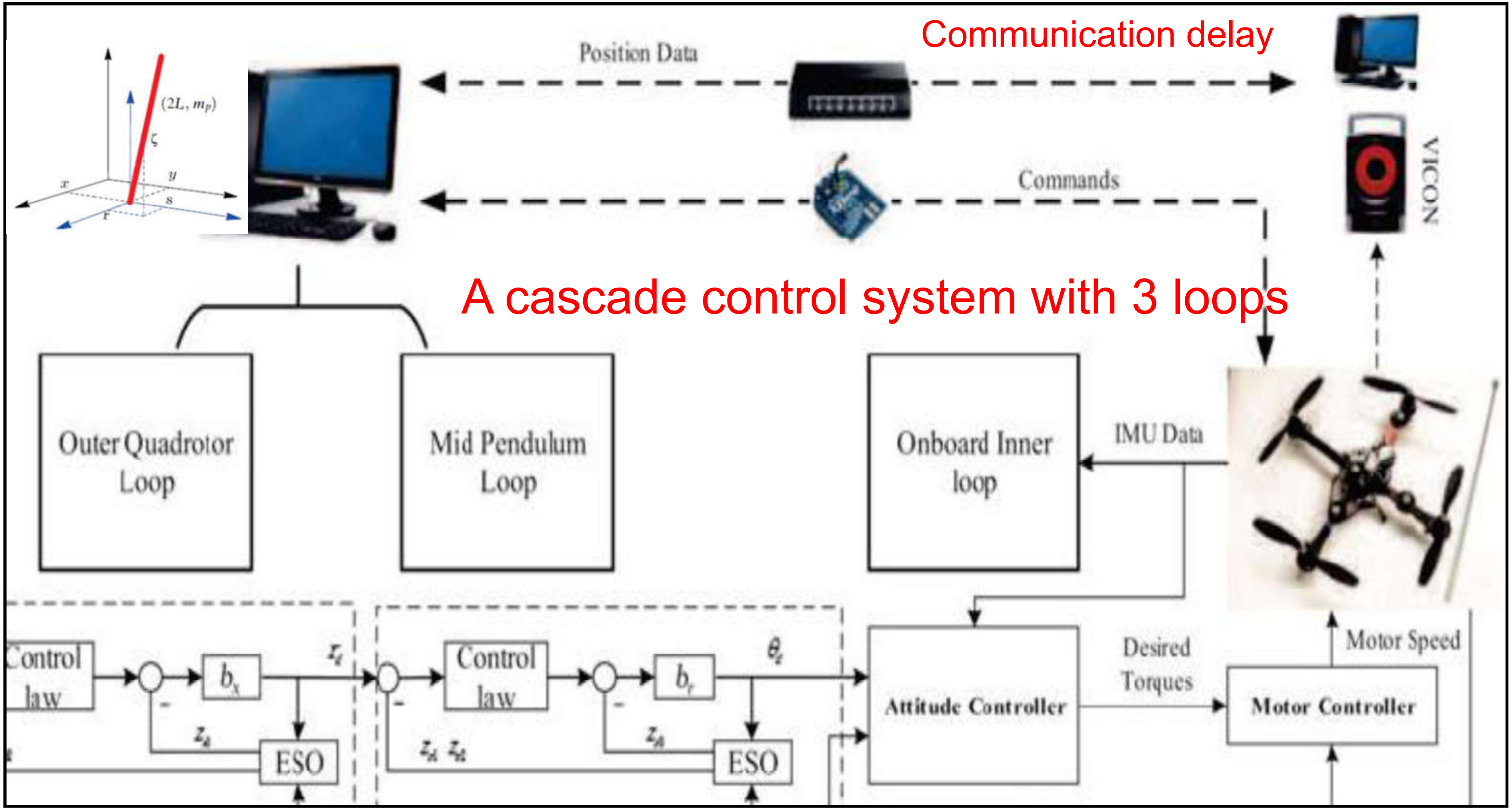
6 PID Control Algorithm-Application

Rug Warrior robot follows a wall by using its onboard sonars

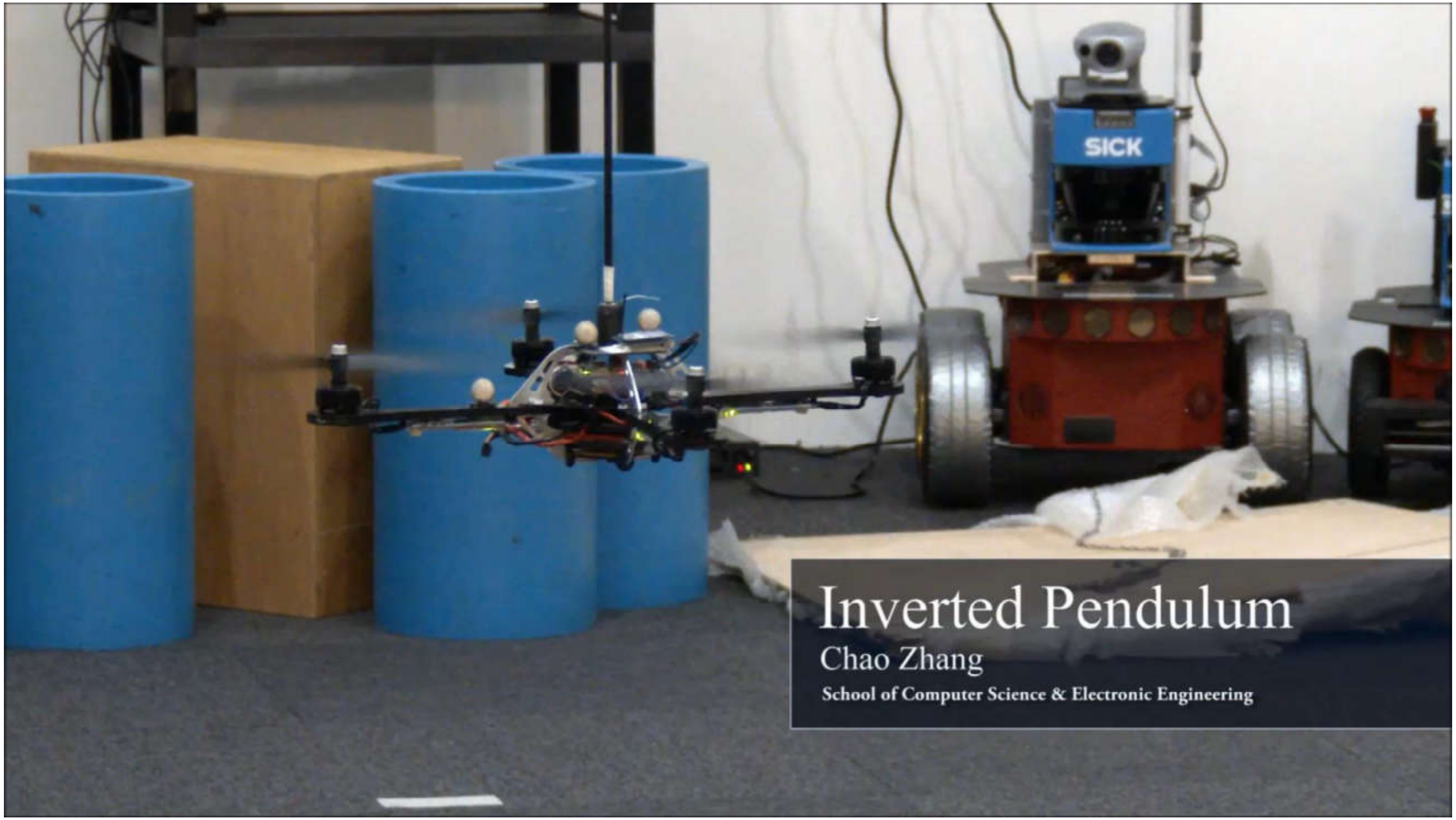
6 PID Control Algorithm-Application

PID control of a car

6 PID Control Algorithm-Application



6 PID Control Algorithm-Application



7 Fuzzy Control Algorithm

7.1 Introduction

7.2 Set theory: Classic vs Fuzzy

7.3 Fuzzy membership functions

7.4 Linguistic variables

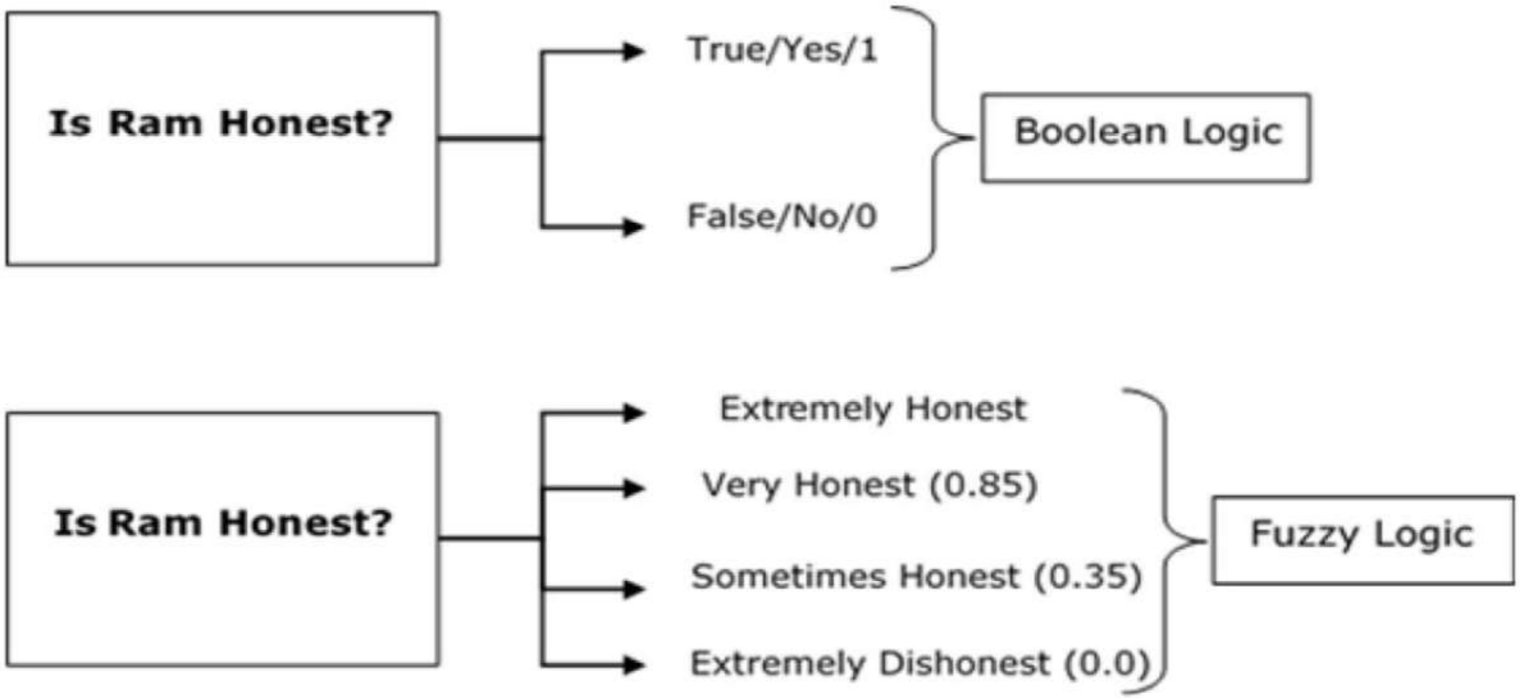
7.5 Fuzzification & Defuzzification

7.6 Fuzzy inference system

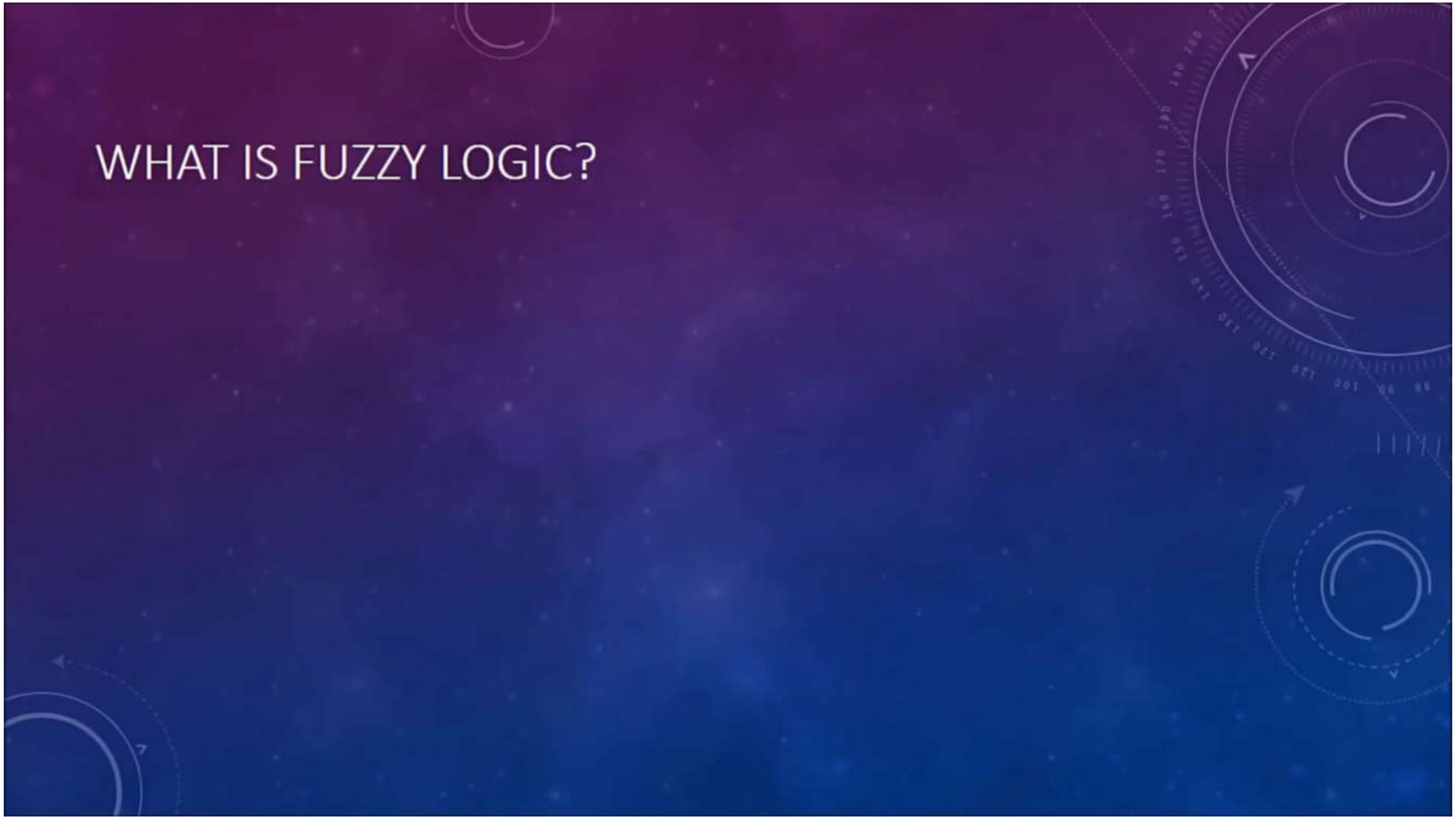
7.7 Fuzzy controller & applications

7.1 Introduction

- Fuzzy Logic was introduced by Professor Lofti A. Zadeh in 1965, i.e. his research paper "Fuzzy Sets".
- Fuzzy Logic resembles the human decision-making methodology. It deals with vague and imprecise information.



7.1 Introduction



7.2 Set Theory: Classic vs Fuzzy

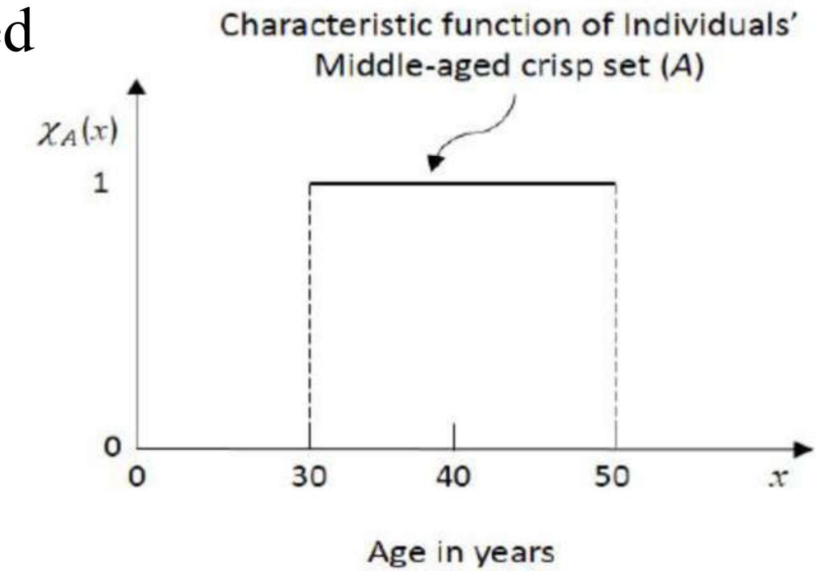
- A classic set is an unordered collection of different elements.
- It can be written explicitly by listing its elements using the set bracket.
- If the order of the elements is changed or any element of a set is repeated, it does not make any changes in the set.

Example: A set of individuals' middle aged

$$A = \{ x \mid 30 \leq x \leq 50 \}$$

Its full membership function is

$$\chi_A(x) = \begin{cases} 1, & \text{for} \quad 30 \leq x \leq 50 \\ 0, & \text{for} \quad otherwise \end{cases}$$



7.2 Set Theory: Classic vs Fuzzy

1. Mathematical Representation:

Roster or Tabular Form

- A classic set is represented by listing all the elements comprising it.
- The elements are enclosed within braces and separated by commas.

  e.g. Set of odd numbers less than 10, $B=\{1,3,5,7,9\}$

Set Builder Notation

- The classic set is described as A={x:p(x)}

  e.g. The set {1,3,5,7,9} is written as $A=\{x{:}1\leq x<10\ and\ (x\%2)\neq 0\}$
- If an element x is a member of any classic set S, it is denoted by $x \in S$
- If an element y is not a member of classic set S, it is denoted by $y \notin S$.

  e.g. If $S = \{1,1.2,1.7,2\}, 1 \in S, but\ 1.5 \notin S$

7.2 Set Theory: Classic vs Fuzzy

2. Types of Classical Sets

- **Finite Set** contains a definite number of elements is called a finite set.

  e.g. $S=\{x|x\in N \text{ and } 70>x>50\}$

- **Infinite Set** contains infinite number of elements is called an infinite set.

  e.g. $S=\{x|x\in N \text{ and } x>10\}$

- **Subset** - Y is a subset of set X if its elements are the elements of set X.

  e.g. Let Y={1,2,3,4,5,6} and X={1,2}. Hence, we can write $Y\subseteq X$.

- **Universal Set** is a collection of all elements in a particular context and represented as U. Example: if U is the set of all animals on earth, then a set of all fishes is a subset of U

- **Empty Set or Null Set** is finite set without elements, denoted by Ø.

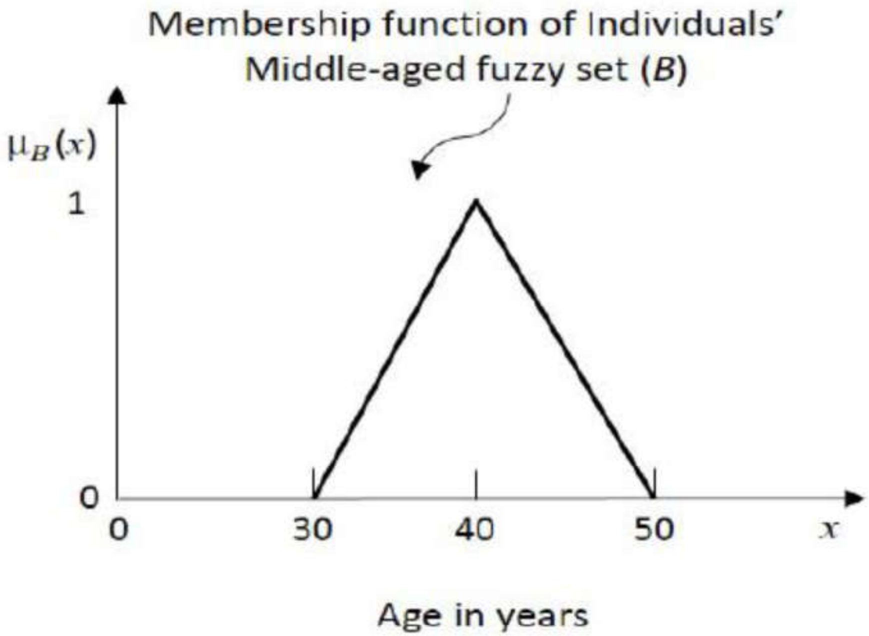
7.2 Set Theory: Classic vs Fuzzy

Concept of Fuzzy Sets

Fuzzy sets extend the concept of classical sets by providing a means of calculating intermediate values between absolute false and absolute true with values (0.0 ~1.0). Example: we can define the individuals' middle-aged set into a fuzzy set

$$B = \{ \ (x, \mu_B(x)) \mid 30 \le x \le 50 \ \}$$

Its graphical representation is shown →



Its membership degrees between 0 and 1 can be mathematically expressed by

$$\mu_B(x) \ : \ X \ \rightarrow \ [0, 1]$$

7.2 Set Theory: Classic vs Fuzzy

Basic Properties of Fuzzy Sets

Suppose *B* is a fuzzy set of elements, denoted by *x*, defined on the universe of discourse *X*, and represented by a membership function $\mu_B(x)$, then:

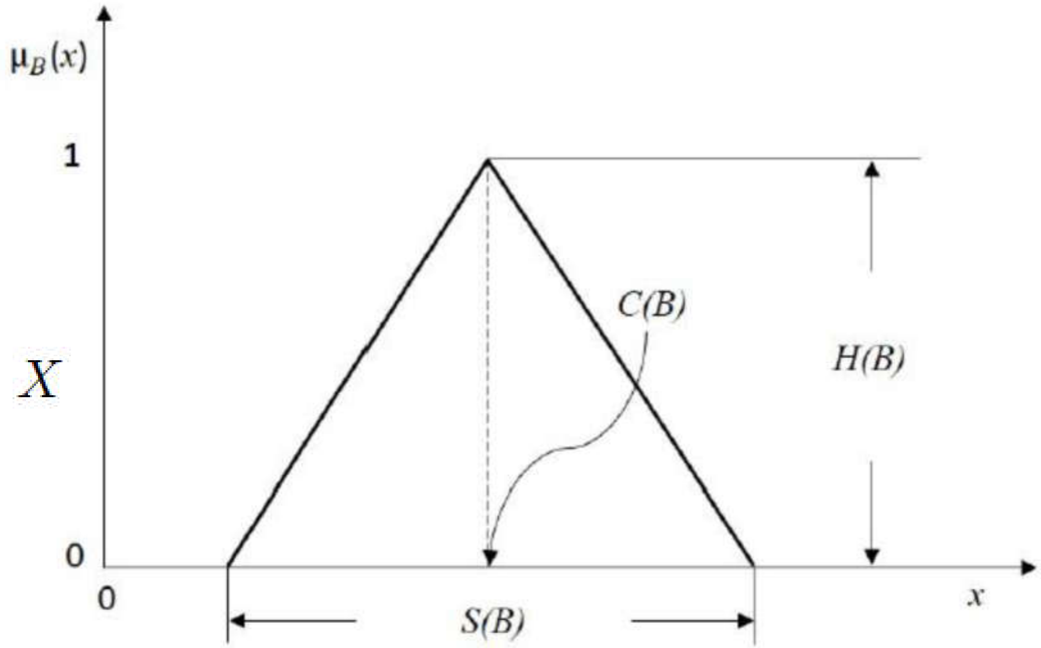
1) Support of a Fuzzy Set: *S(B)* is the crisp set of all elements $x \in X$ such that $\mu_B(x) > 0$, and is written as

$$S(B) = \{\ x \in X \mid \mu_B(x) > 0\ \}$$

2) Core of a Fuzzy Set: *C(B)*

It is the crisp set of all elements $x \in X$ such that $\mu_B(x) = 1$, and is written as

$$C(B) = \{\ x \in X \mid \mu_B(x) = 1\ \}$$



7.2 Set Theory: Classic vs Fuzzy

3) Height of a Fuzzy Set: *H(B)* is the largest membership degree corresponding to any element in the set, and is written as

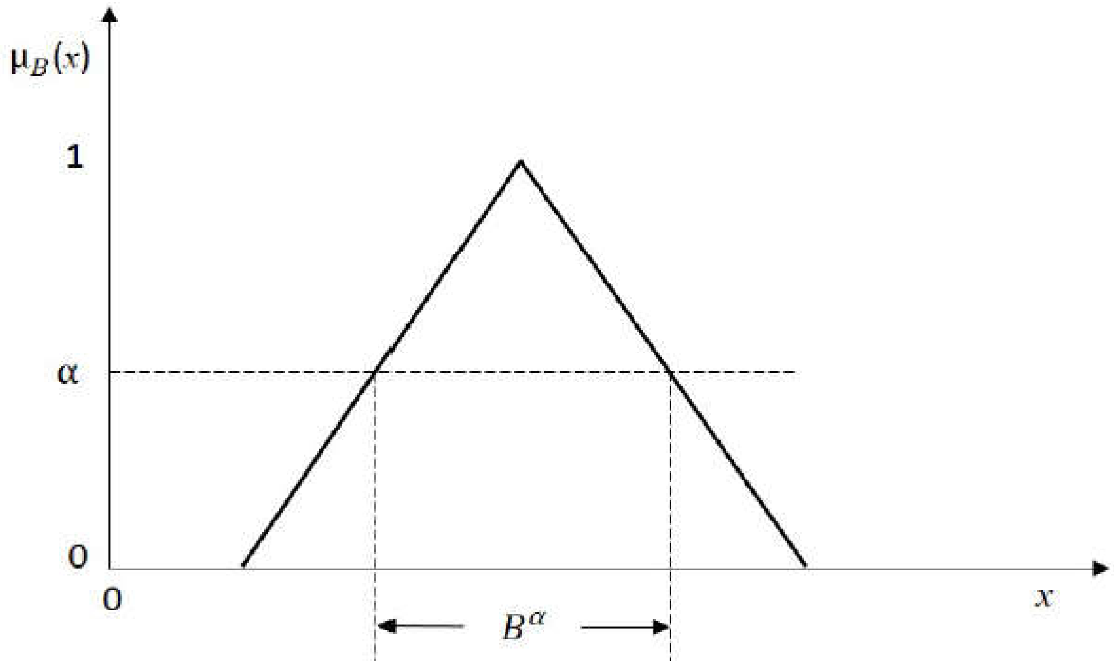
$$H(B) = \sup_{x \in X} \mu_B(x)$$

4) Normal & subnormal Fuzzy Set: A fuzzy set is called normal when *H(B)=1*, and subnormal when *H(B)<1*.

5) *a*-cut of a Fuzzy Set: It is denoted by

$$B^\alpha = \{\, x \in X \mid \mu_B(x) \geq \alpha \,\}$$

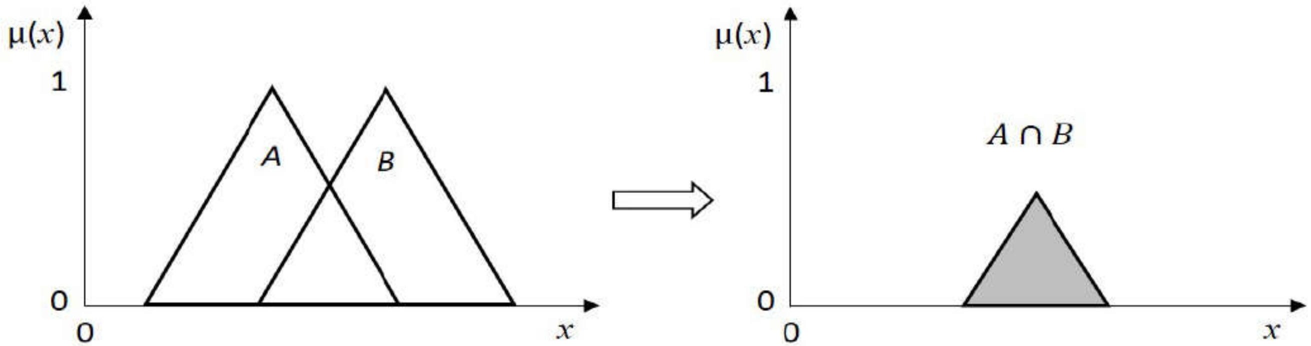
It can be graphically described at right →



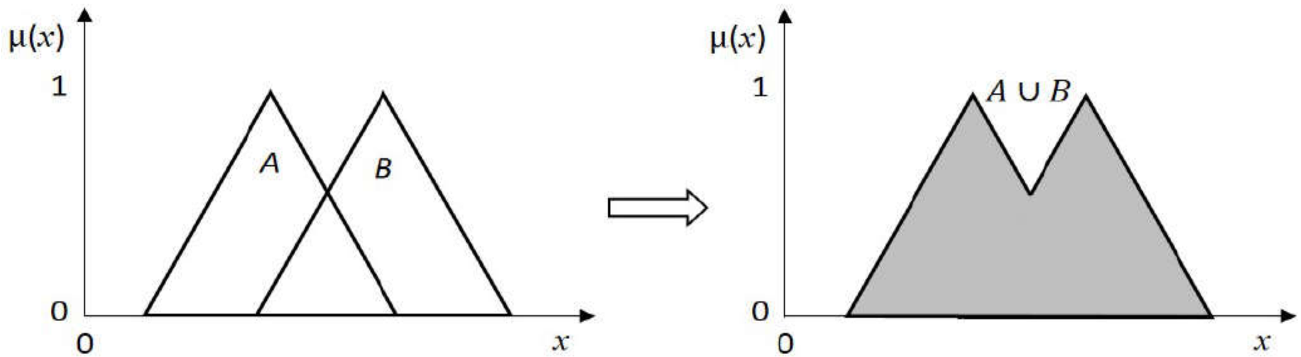
7.2 Set Theory: Classic vs Fuzzy

Basic Operations on Fuzzy Sets

- Intersection Operation: $\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$



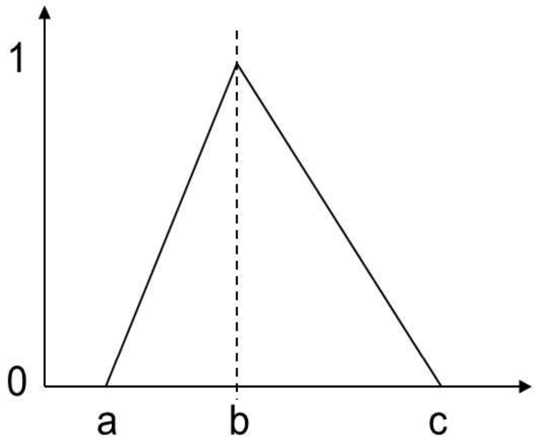
- Union Operation: $\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$



7.3 Fuzzy Membership Functions

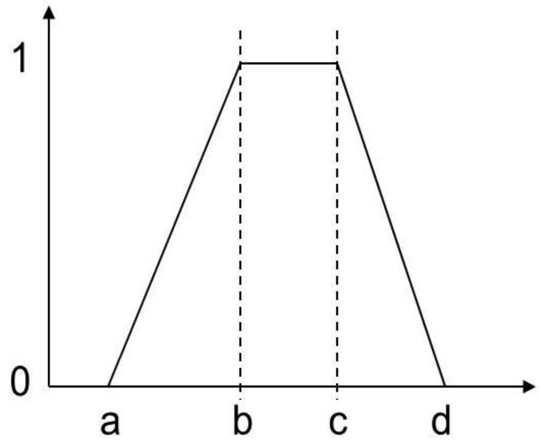
- The membership function is an important part of a fuzzy system that needs to be carefully defined.
- It maps values of input variables on a given universe of discourse into real numbers in a range of values [0,1] and can either be symmetrical or asymmetrical.
- The most common types of membership functions used in fuzzy applications are
  1) Triangular – It can be expressed mathematically using 3 parameters.
  2) Trapezoidal – It can be expressed mathematically using 4 parameters.
  3) Gaussian – It can be expressed mathematically using 2 parameters.

7.3 Fuzzy Membership Functions



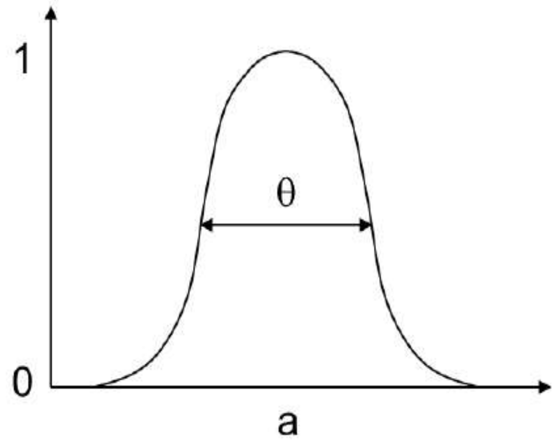
Triangle

$$\begin{array}{ll} (x-a)/(b-a) & \text{for } a \le x \le b \\ (c-x)/(c-b) & \text{for } b \le x \le c \\ 0 & \text{Otherwise} \end{array}$$



Trapezoid

$$\begin{array}{ll} (x-a)/(b-a) & \text{for } a \le x \le b \\ 1 & \text{for } b \le x \le c \\ (d-x)/(d-c) & \text{for } c \le x \le d \\ 0 & \text{Otherwise} \end{array}$$

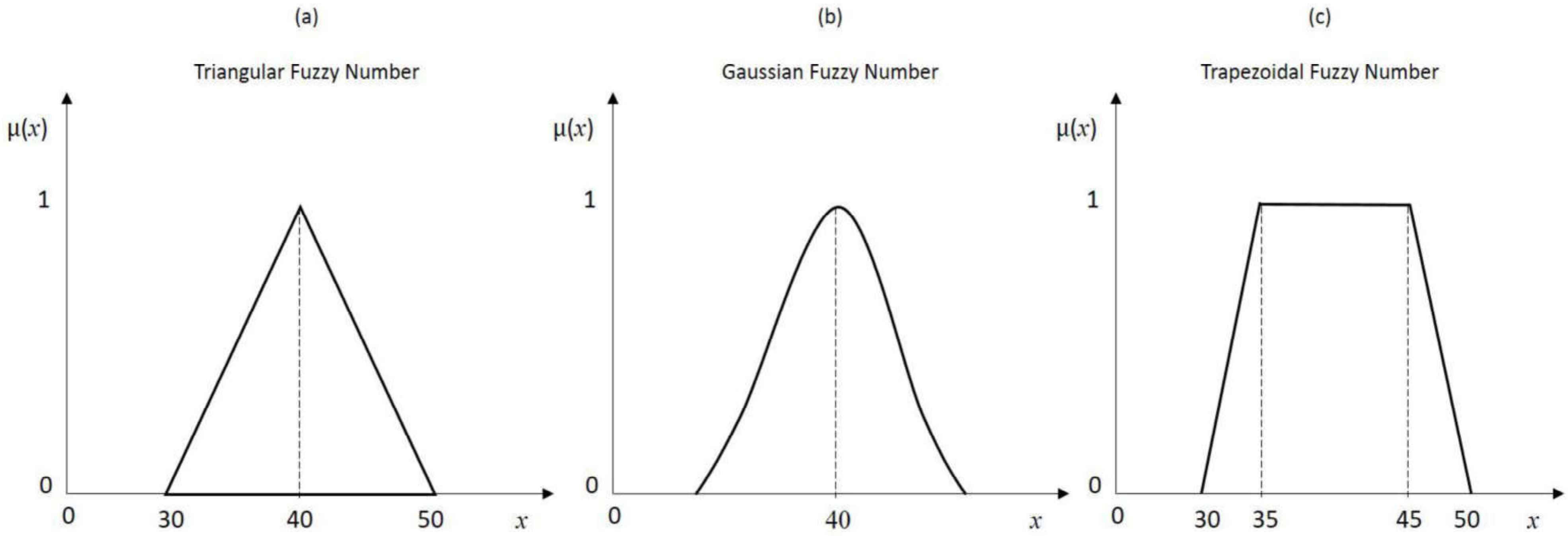


Gaussian

$$e\left(-0.5\left((x-a)/\theta\right)^2\right)$$

7.3 Fuzzy Membership Functions

- The figure below shows an example of a Fuzzy number or a fuzzy interval for individuals' middle-aged



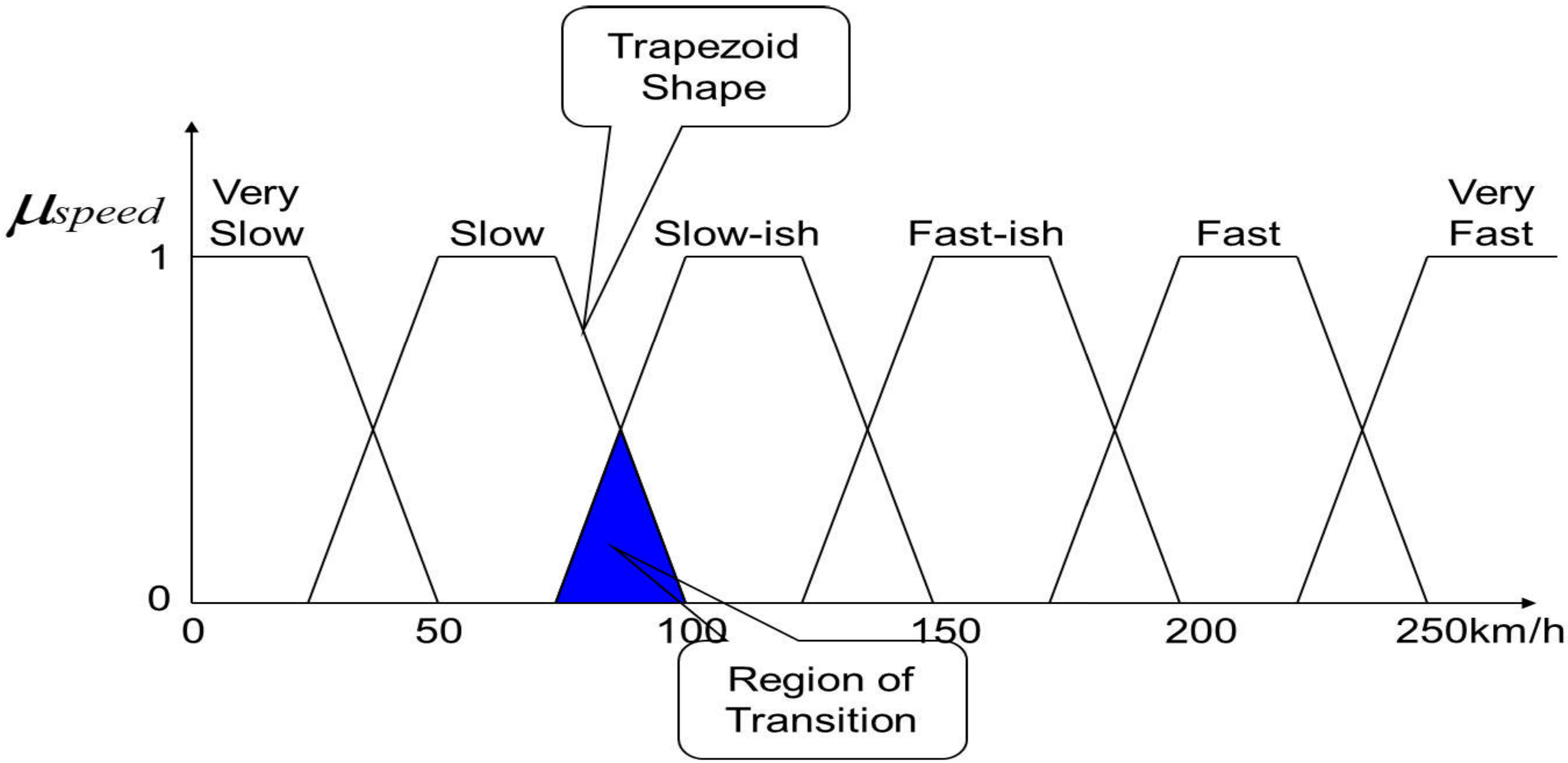
7.3 Fuzzy Membership Functions

Determine Fuzzy Membership Function Process

- The 1st step in the fuzzy logic control process is to determine input - output membership functions.
- A fuzzy algorithm categorizes the information and assigns values that represent the degree of membership within those categories.
- Input membership functions can take any form, which should ensure:
  - the shapes could accurately represent the distribution of data within the system
  - A region of transition exists between adjacent membership functions

7.3 Fuzzy Membership Functions

Example: Input Fuzzy Membership Function



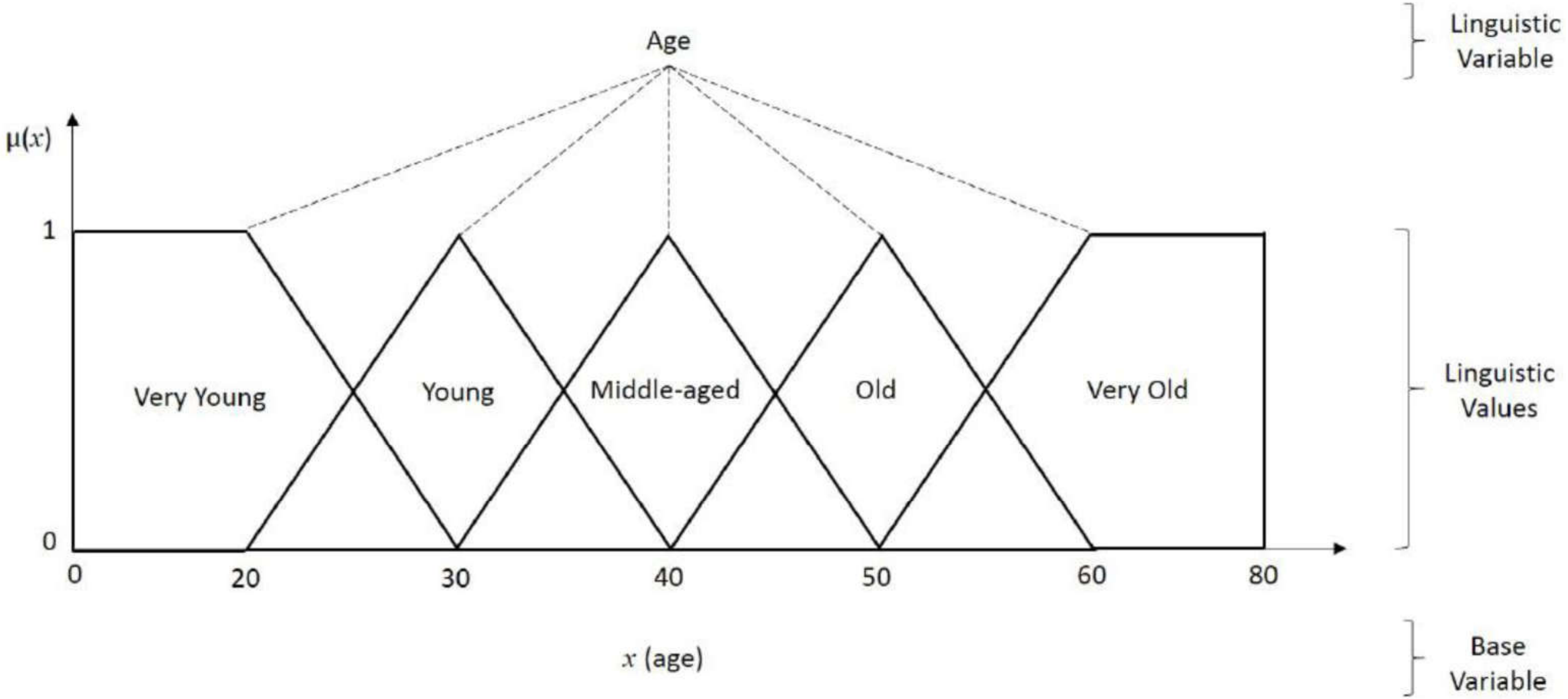
7.3 Fuzzy Membership Functions

7.4 Linguistic Variables

- The linguistic variables are inexact/uncertain variables to describe the behaviour of complex systems in terms of propositions used by humans.
- These propositions (phrases) are expressed in natural language to be converted into fuzzy sets or fuzzy numbers for fuzzy operations.
- Example 1: a linguistic variable might be Age, which is shown in the next page. The linguistic variable, Age, is defined as a base variable on a specified universe of real numbers.
- It is expressed as 5 linguistic values (also called linguistic terms or linguistic labels), *Very Young, Young, Middle-aged, Old and Very Old*. These linguistic values are represented by specific fuzzy numbers defined on the base variable universe.

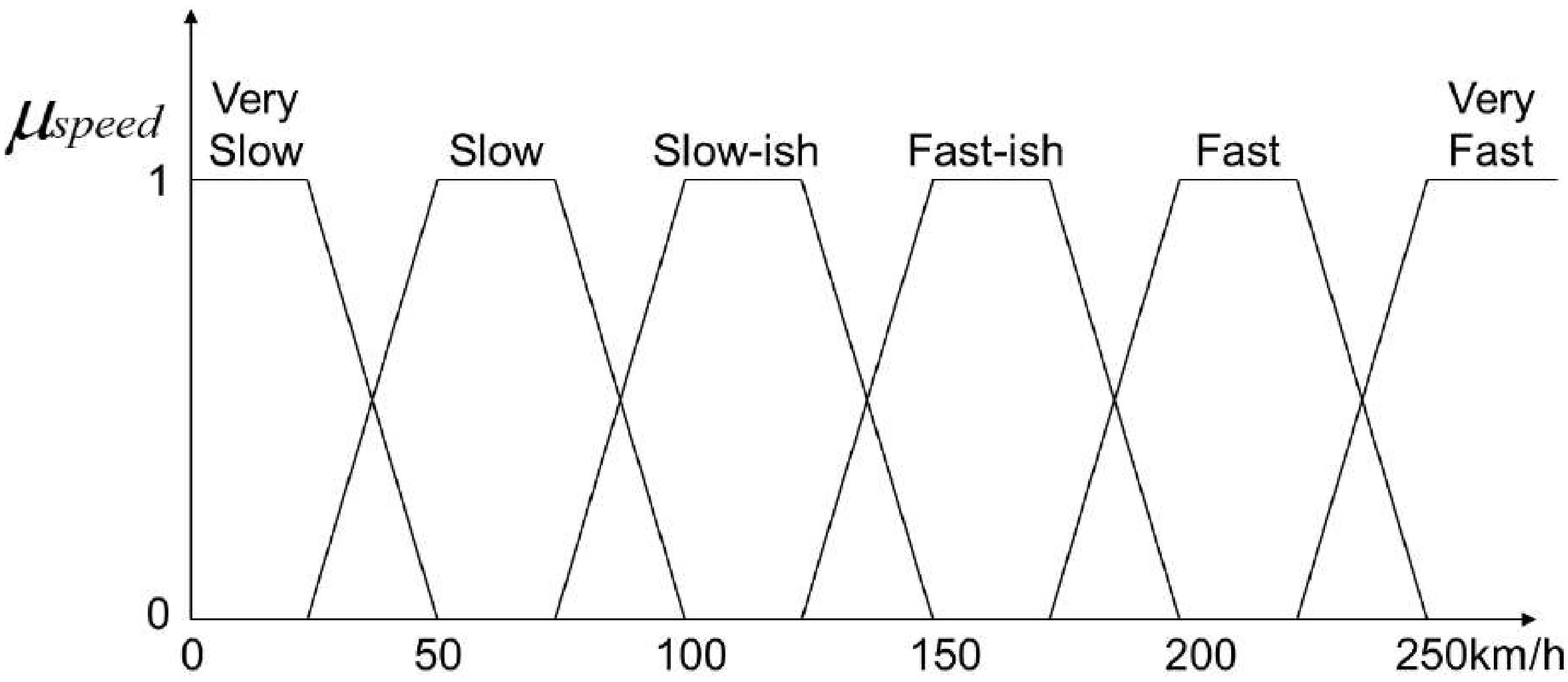
7.4 Linguistic Variables

Linguistic Variables Example 1: Age



7.4 Linguistic Variables

Linguistic Variables Example 2: Speed



7.5 Fuzzification and Defuzzification

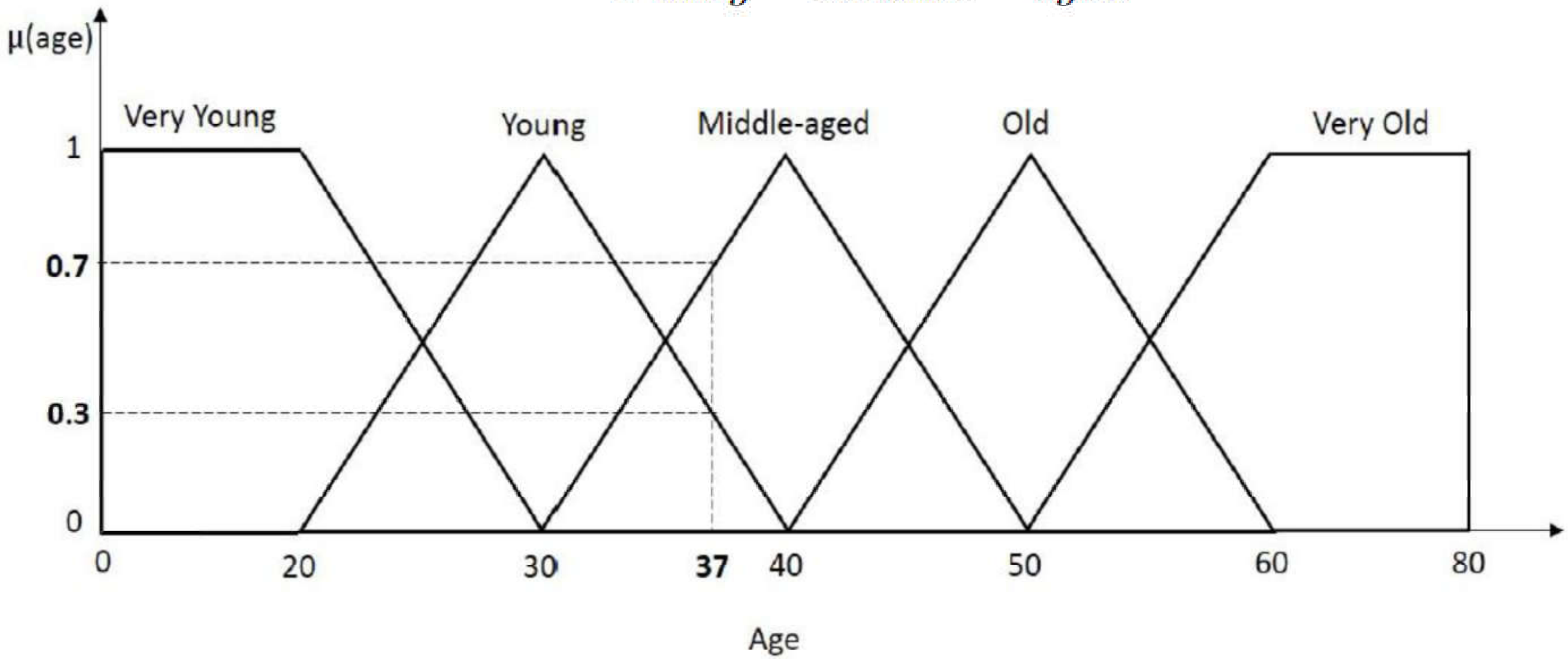
1. Fuzzification process

- This is a process of converting the input variables into a fuzzy format in order to handle the necessary uncertainty.
- That is to determine a fuzzy set representing all membership degrees of the linguistic values corresponding to the input variable data.
- When the input to the fuzzification process is a crisp value, the obtained membership degrees 0.3 and 0.7 are associated with the linguistic values "Young" and "Middle-aged", respectively,
- When the input to the fuzzification process is a fuzzy value, you might obtain many membership values of the linguistic values. In this case, the greater membership is selected to represent their linguistic value.

7.5 Fuzzification and Defuzzification

- Fuzzification operation 1: the input value is a crisp number.

$$Age_{(37)} = \{\frac{0.3}{Young}, \frac{0.7}{Middle-aged}\}$$

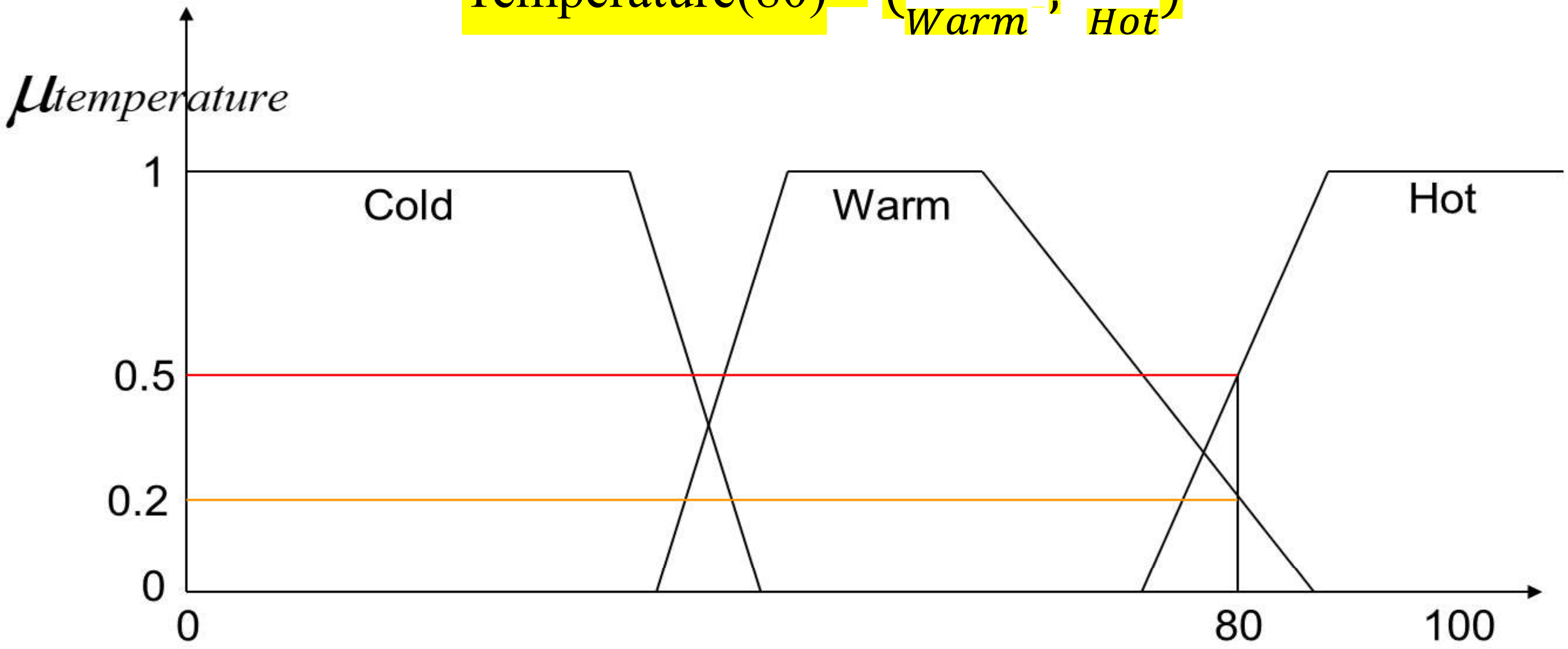


7.5 Fuzzification and Defuzzification

Another example: Fuzzification Operation 1

The input is a crisp temperature, 80. It covers Warm MF at 0.2 and Hot MF at 0.7.

$$\text{Temperature}(80) = \left(\frac{0.2}{Warm}, \frac{0.5}{Hot}\right)$$

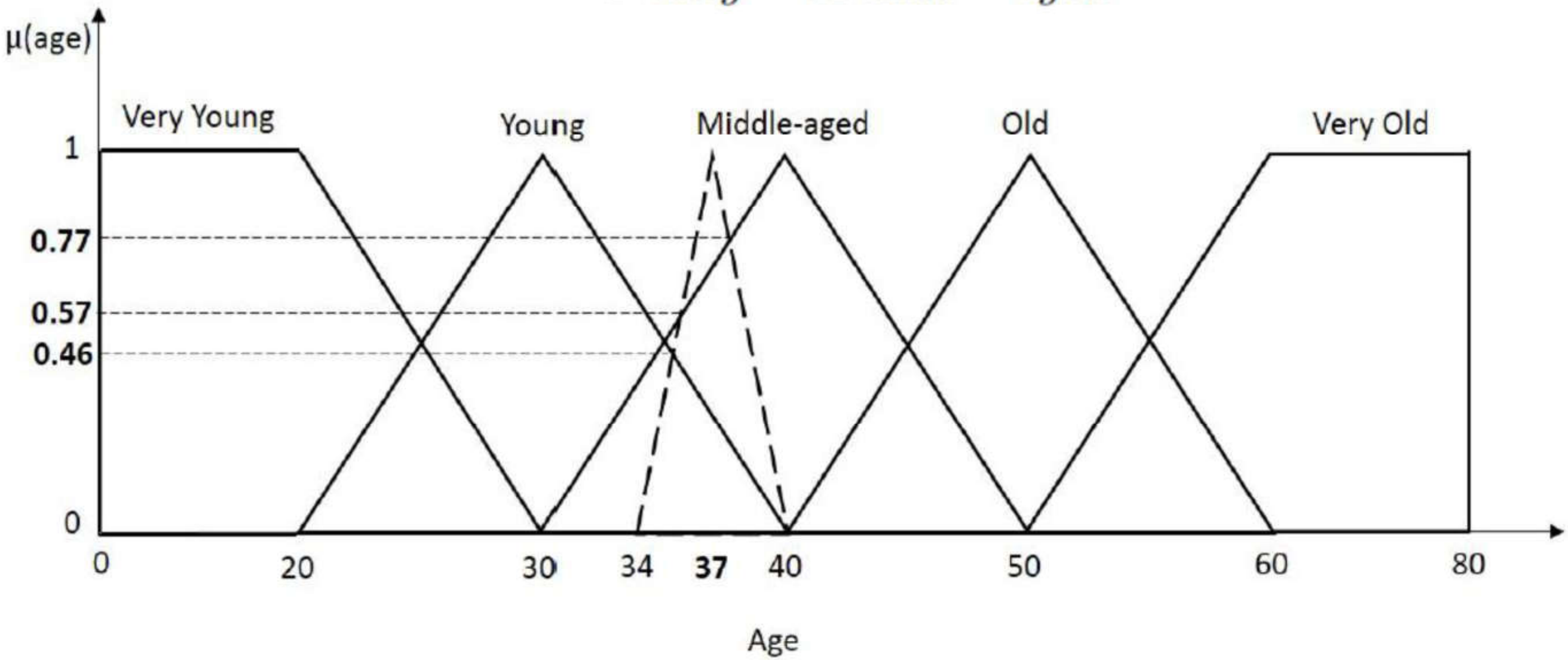


7.5 Fuzzification and Defuzzification

- Fuzzification operation 2: the input value is a fuzzy number.

$$Age_{(34,37,40)} = \{\frac{0.46}{Young}, \frac{0.77}{Middle-aged}\}$$

Note: 0.77 > 0.57



7.5 Fuzzification and Defuzzification

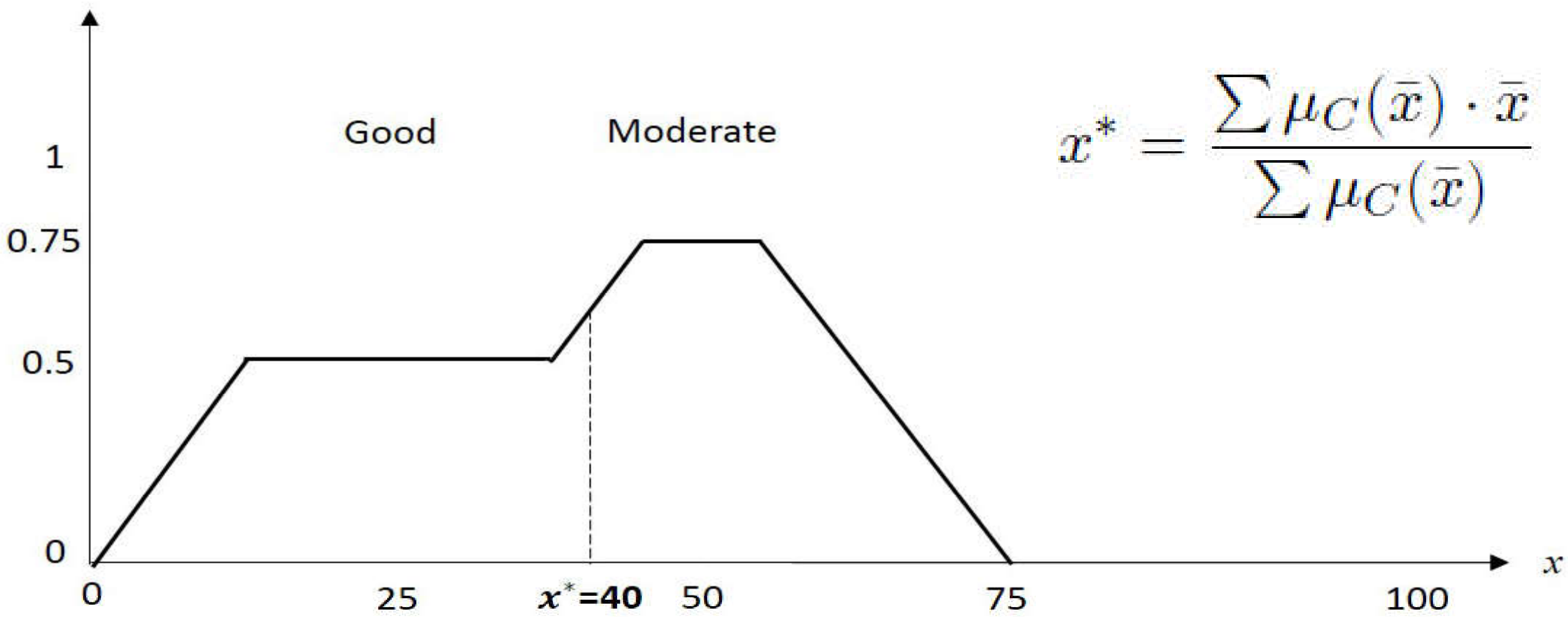
2. Defuzzification process

- It is a process of converting fuzzy output value into a single crisp value needed in many real-world applications.
- The commonly used methods are:
  - Weighted Average method
  - Centroid method
  - Mean-Max method
- Note that defuzzification is an art rather than a science! (as there is no systematic procedure existed.)

7.5 Fuzzification and Defuzzification

Weighted Average Method

- It is commonly used because of its simple computational procedure.
- Its results lie approximately in the middle of the fuzzy support.
- It is suitable to the output fuzzy set with one type membership functions.

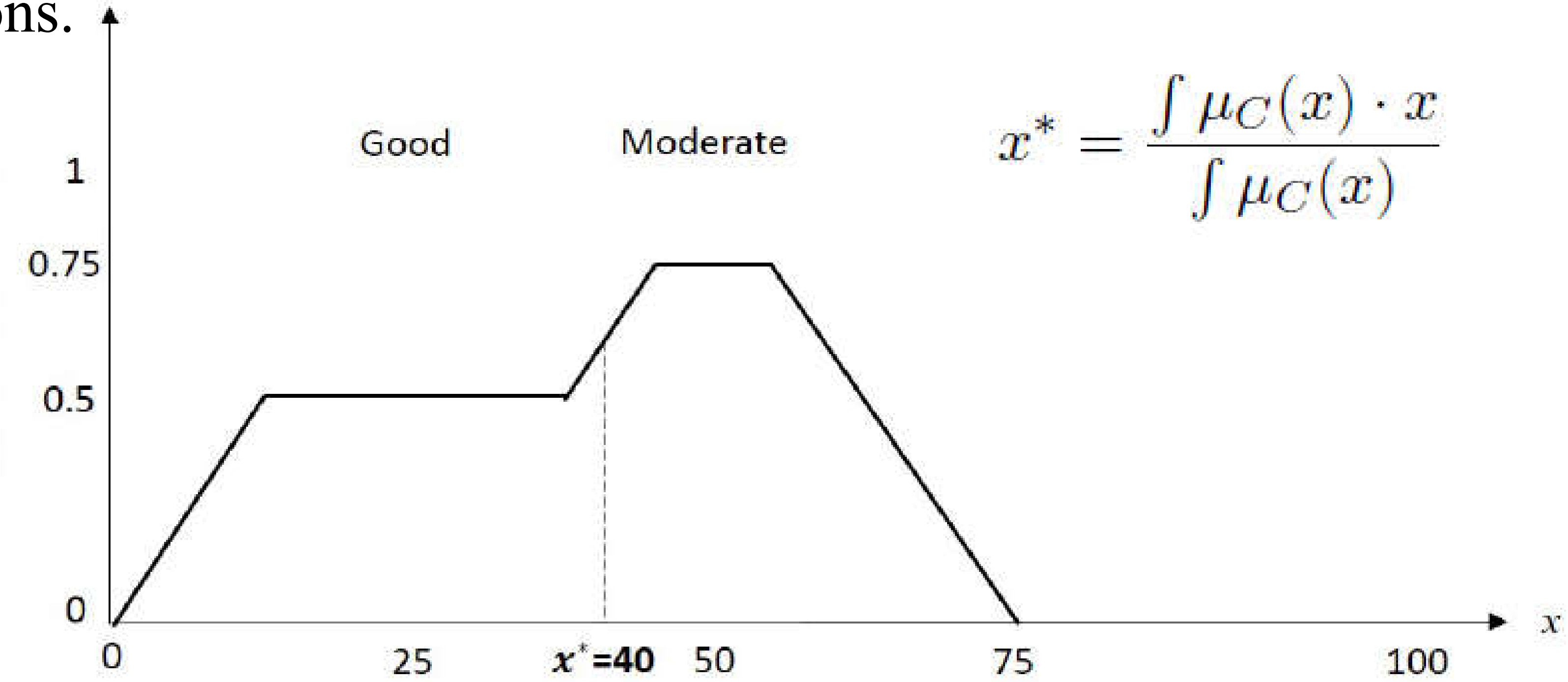


$$x^* = \frac{\sum \mu_C(\bar{x}) \cdot \bar{x}}{\sum \mu_C(\bar{x})}$$

7.5 Fuzzification and Defuzzification

Centre of Gravity

- Its computation is more complex than the weighted average method.
- It is suitable to the output fuzzy set with different types of membership functions.



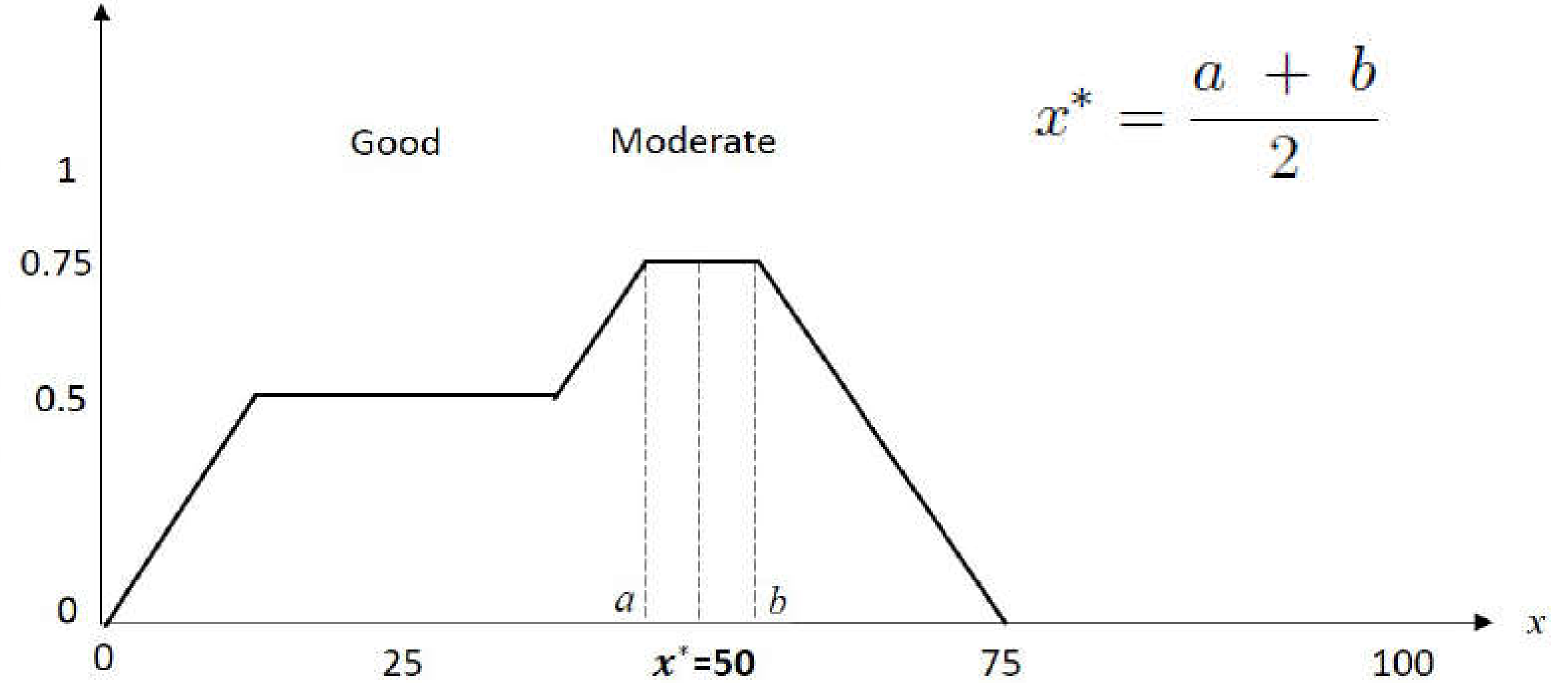
$$x^* = \frac{\int \mu_C(x) \cdot x}{\int \mu_C(x)}$$

7.5 Fuzzification and Defuzzification

Mean-Max method

- It obtains the mean value of the min-max values related to the maximum membership grade.
- It is also called the max membership or height method.

$$x^* = \frac{a + b}{2}$$



7.5 Fuzzification and Defuzzification

- For triangular membership functions the ‘cut off’ area is given by

$$w\left(h - \frac{h^2}{2}\right)$$

  where w is the triangle base width; h is the height at which the function is cut off.

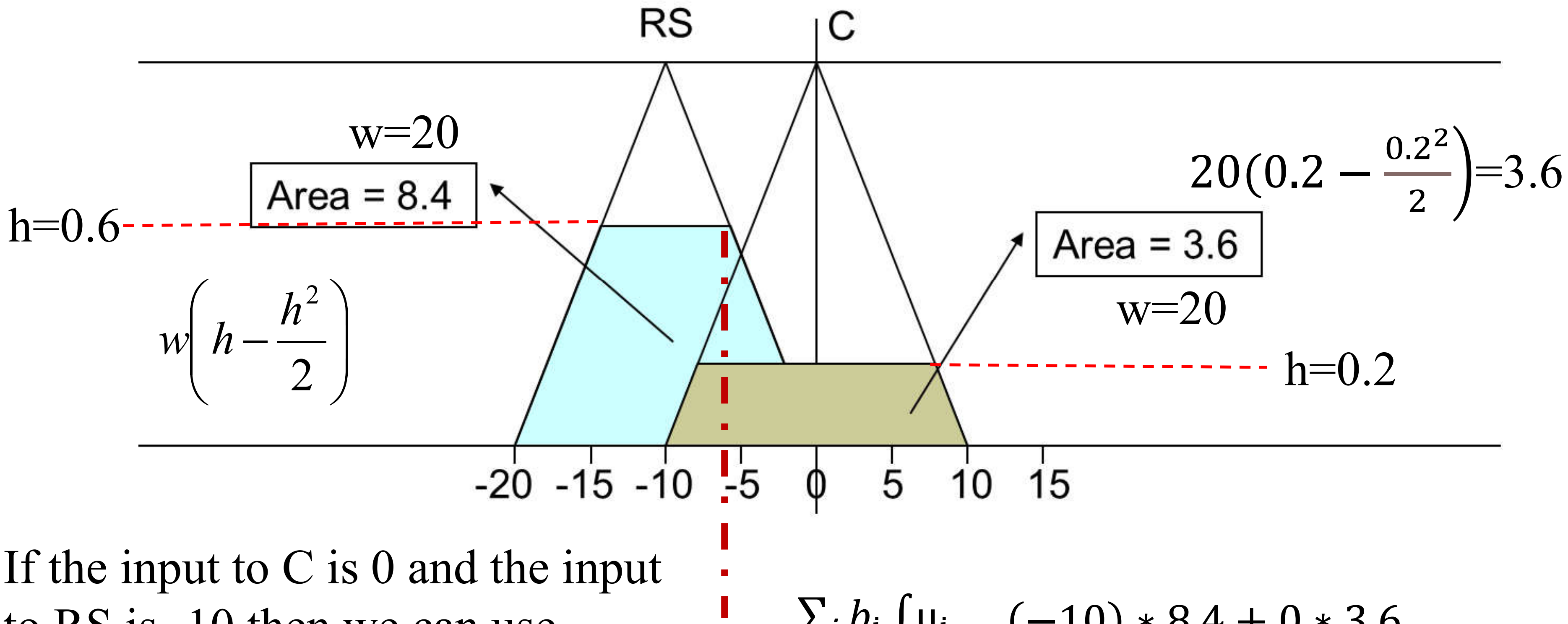
- For trapezoid membership functions the ‘cut off’ area is given by

$$h\left(\left(2w_1 + hw_2 - hw_1\right)/2\right)$$

  where w_1 is the trapezoid base width; w_2 is the trapezoid top width; h is the height at which the function is cut off.

7.5 Fuzzification and Defuzzification

- Suppose: the membership of C is 0.2 and the membership of RS is 0.6.

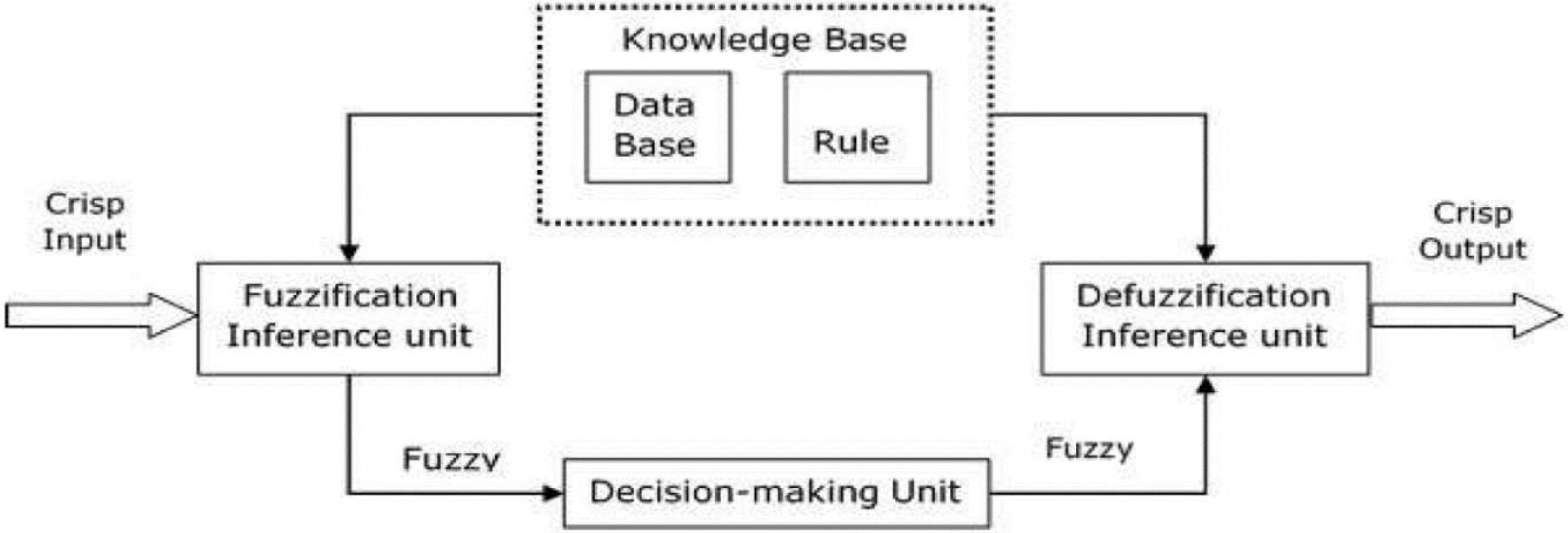


If the input to C is 0 and the input to RS is -10 then we can use "Centre of Gravity" to obtain the crisp output value.

$$u=\frac{\sum_i b_i \int \mu_i}{\sum_i \int \mu_i}=\frac{(-10)*8.4+0*3.6}{8.4+3.6}=-7$$

7.6 Fuzzy Inference System

A Fuzzy inference system has four function blocks:



1) ***Fuzzification Interface Unit*** converts the crisp quantities into fuzzy quantities.
2) ***Knowledge Base*** contains data base and fuzzy IF-THEN rules.
3) ***Decision-making Unit*** performs operation on rules.
4) ***Defuzzification Interface Unit*** converts the fuzzy quantities into crisp quantities.

7.6 Fuzzy Inference System

1. Mamdani Fuzzy Inference System

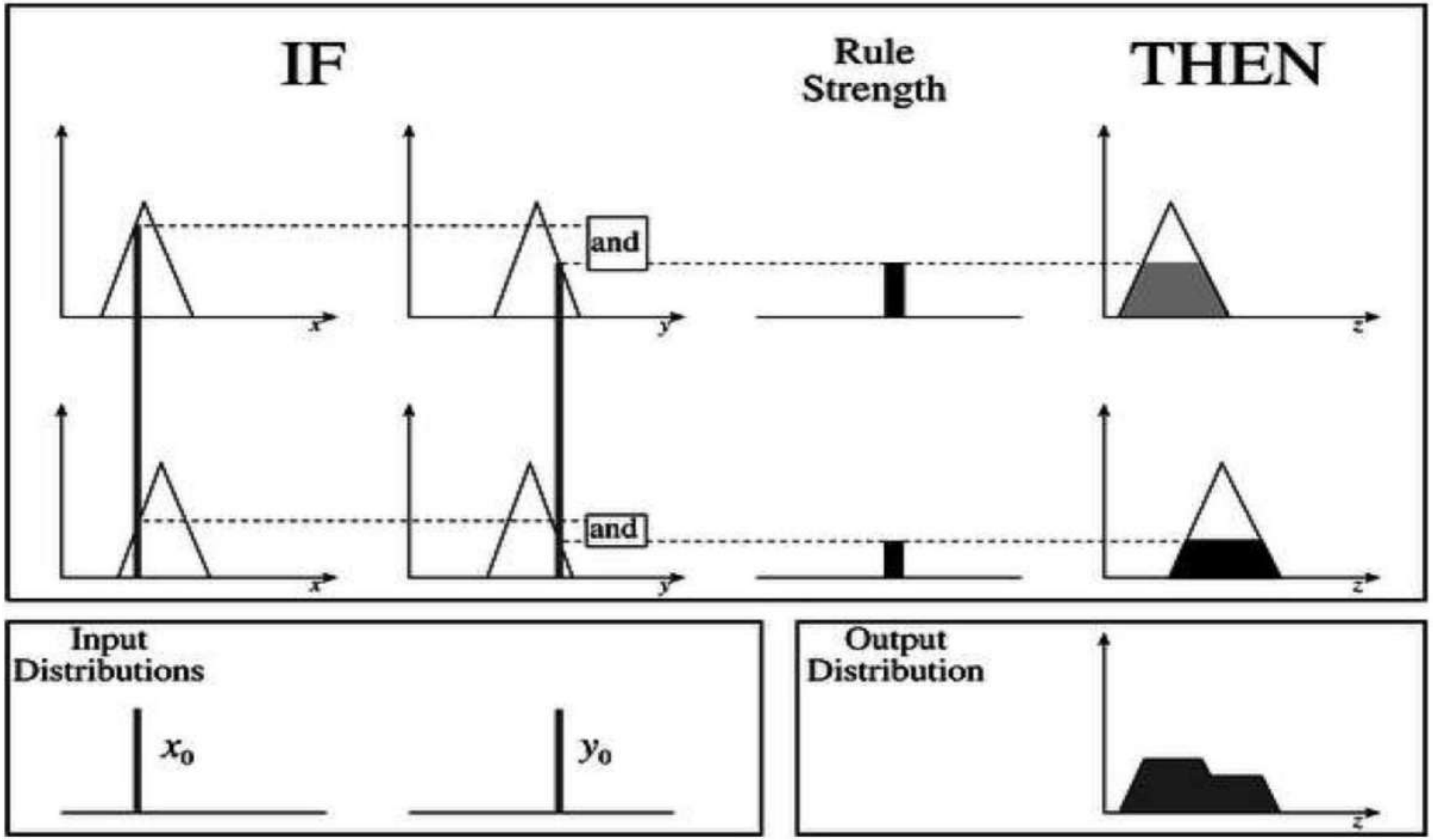
- It was proposed in 1975 by Prof. Mamdani (Queen Mary, London)
- It was used to control a combined system of a steam engine and a boiler.

Steps for Computing the Output of a Mamdani FIS

- **Step 1** − The fuzzification of the input variables.
- **Step 2** − Rule evaluation.
- **Step 3** − The aggregation of the rule outputs.
- **Step 4** − Finally, defuzzification.

7.6 Fuzzy Inference System

The block diagram of M-Fuzzy Interface System is shown below.



7.6 Fuzzy Inference System

2. TS Fuzzy Inference System

It was proposed by Takagi Sugeno & Kang in 1985 with the following format:

IF *x is A and y is B* ***THEN*** *Z = f(x,y)*

where *A* and *B* are fuzzy sets in antecedents and

z = f(x, y) is a crisp function in the consequent.

TS Fuzzy Inference Process

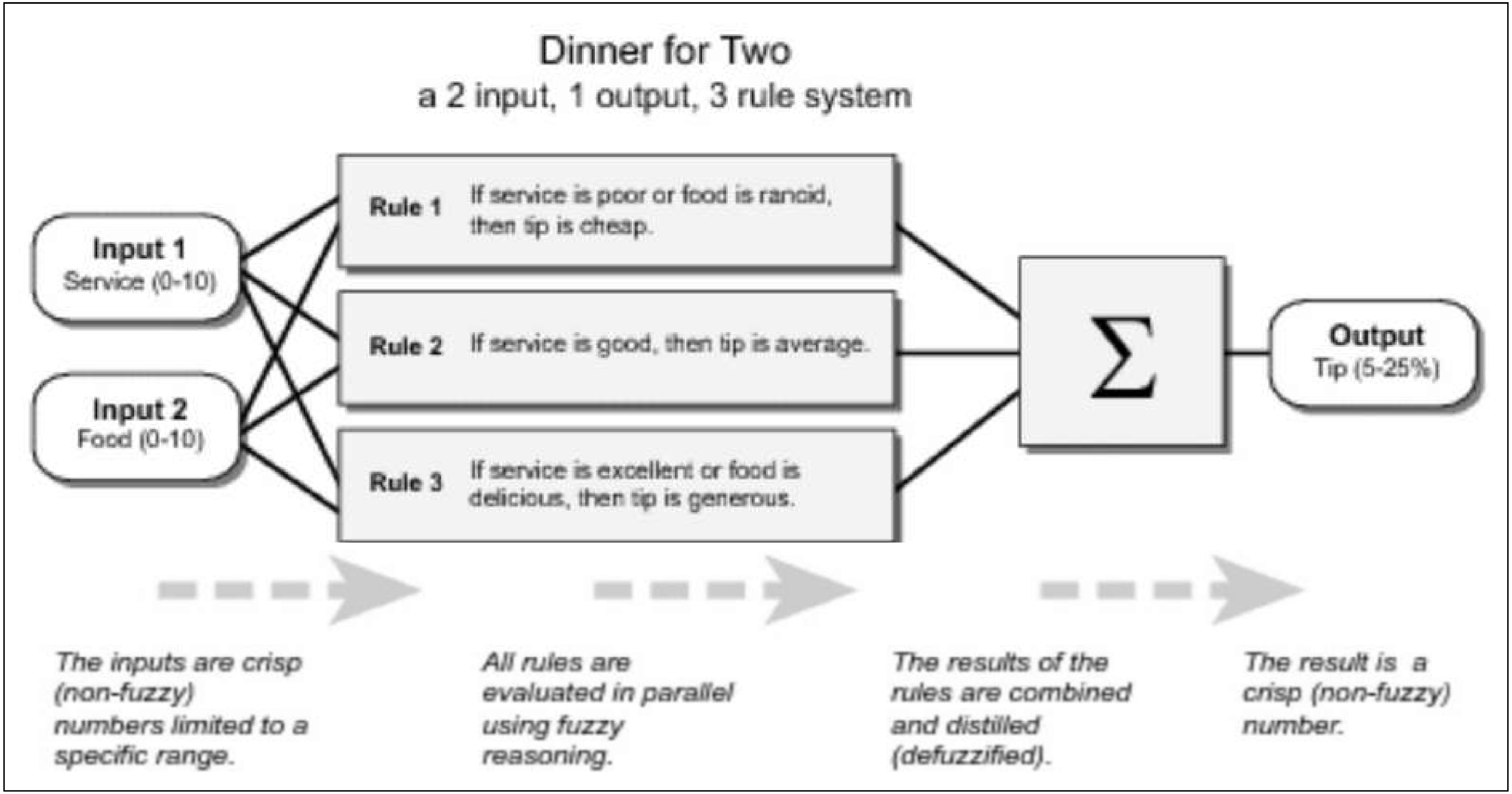
- ***Step 1: Fuzzifying the inputs*** − Here, the inputs of the system are made fuzzy.
- ***Step 2: Applying the fuzzy operator*** − In this step, the fuzzy operators must be applied to get the output.

7.6 Fuzzy Inference System

Comparison between Mamdani FIS and TS FIS

- ***Output Membership Function***
  - M-FIS has linear output membership function.
  - TS-FIS has either linear or constant output membership functions.
- ***Aggregation and Defuzzification Procedure***
  - They have different aggregation and defuzzification procedures due to their consequences of fuzzy rules.
- ***Mathematical Rules***
  - TS-FIS has more mathematical rules than M-FIS.
- ***Adjustable Parameters***
  - TS-FIS controller has more adjustable parameters than M-FIS controller.

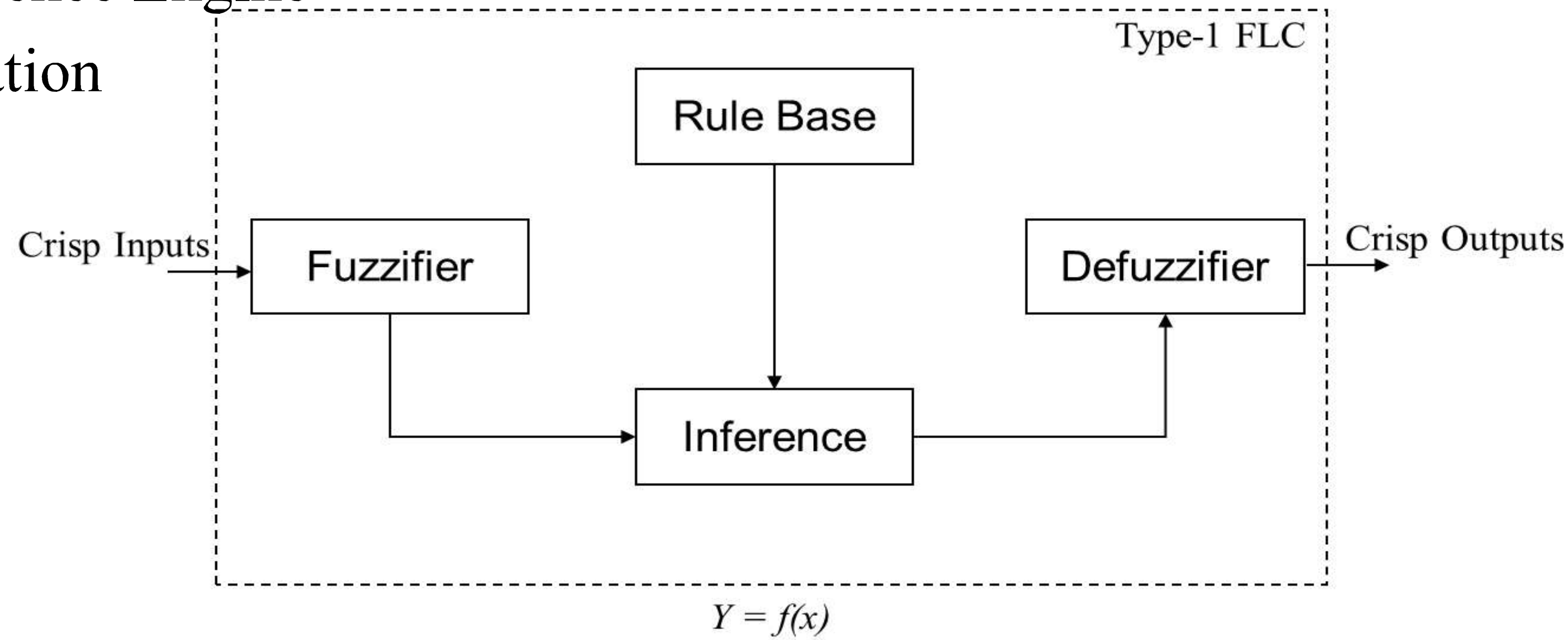
7.6 Fuzzy Inference System



7.7 Fuzzy Controller and Applications

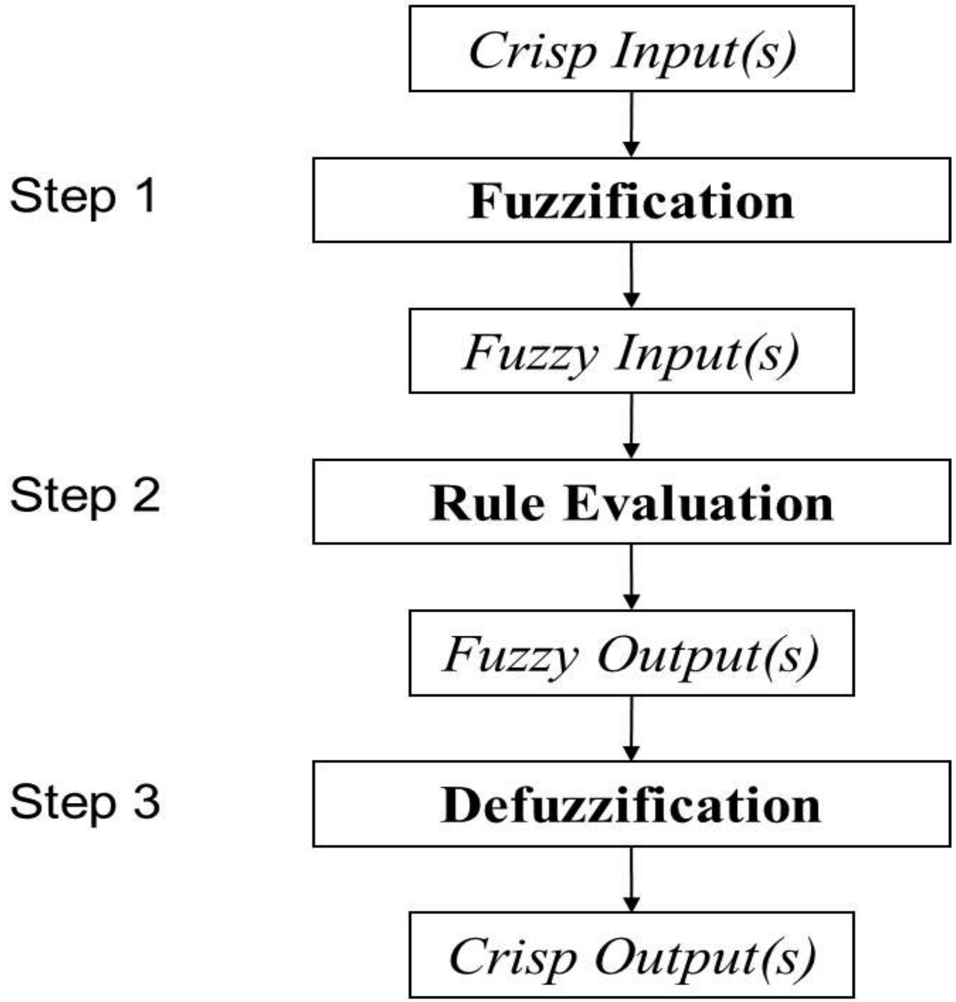
A fuzzy logic control system usually consists of four major parts:

- Fuzzification Interface
- Fuzzy Rule Base
- Fuzzy Inference Engine
- Defuzzification



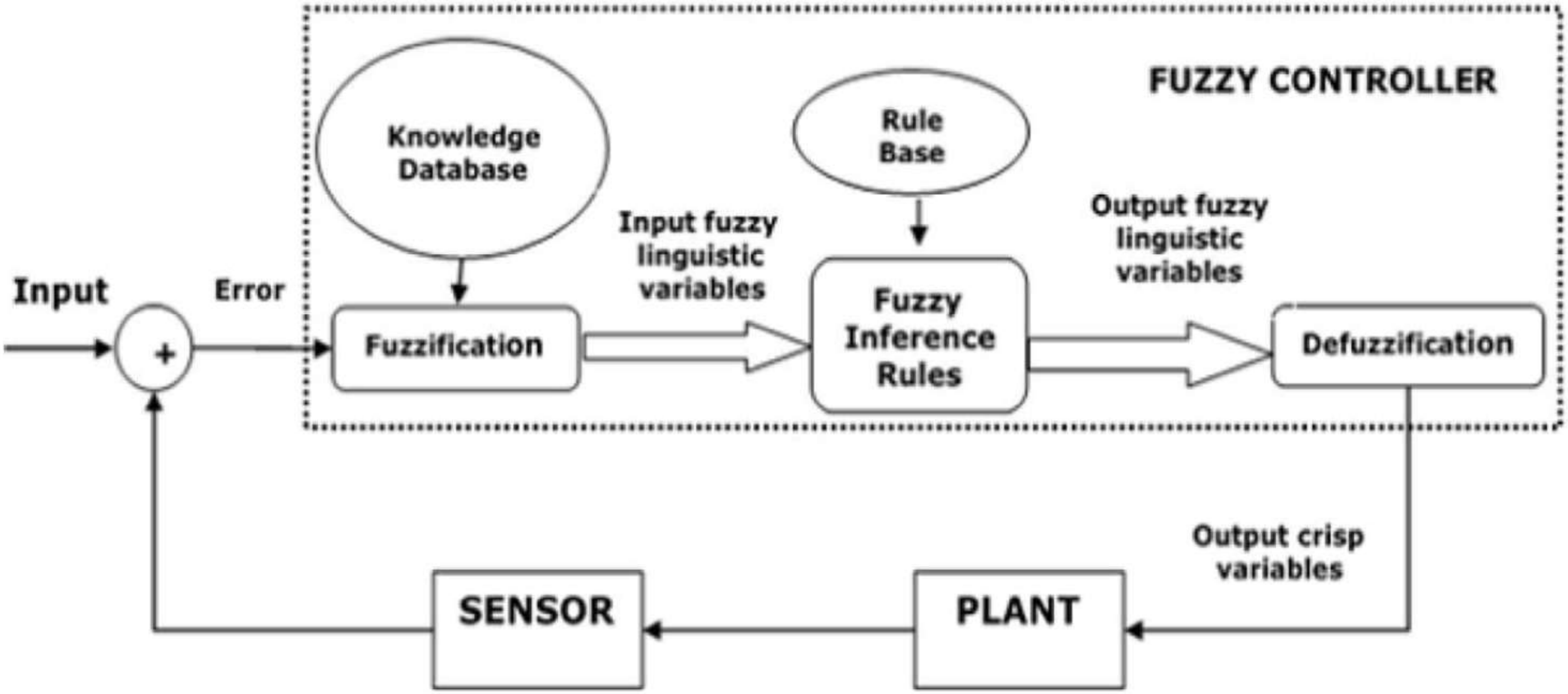
7.7 Fuzzy Controller and Applications

The steps for building a fuzzy logic controller



7.7 Fuzzy Controller and Applications

The structure of a fuzzy control system



7.7 Fuzzy Controller and Applications

The major components of the FLC as shown in the previous figure:

- **Fuzzifier** − The role of fuzzifier is to convert the crisp input values into fuzzy values.
- **Fuzzy Knowledge Base** − It stores the knowledge about all the input-output fuzzy relationships. It also has the membership function which defines the input variables to the the fuzzy rule base and the output variables to the plant under control.
- **Fuzzy Rule Base** − It stores the knowledge about the operation of the process of domain.
- **Inference Engine** − It acts as a kernel of any FLC. Basically it simulates human decisions decisions by performing approximate reasoning.
- **Defuzzifier** − It is to convert the fuzzy values into crisp values getting from fuzzy inference engine.

7.7 Fuzzy Controller and Applications

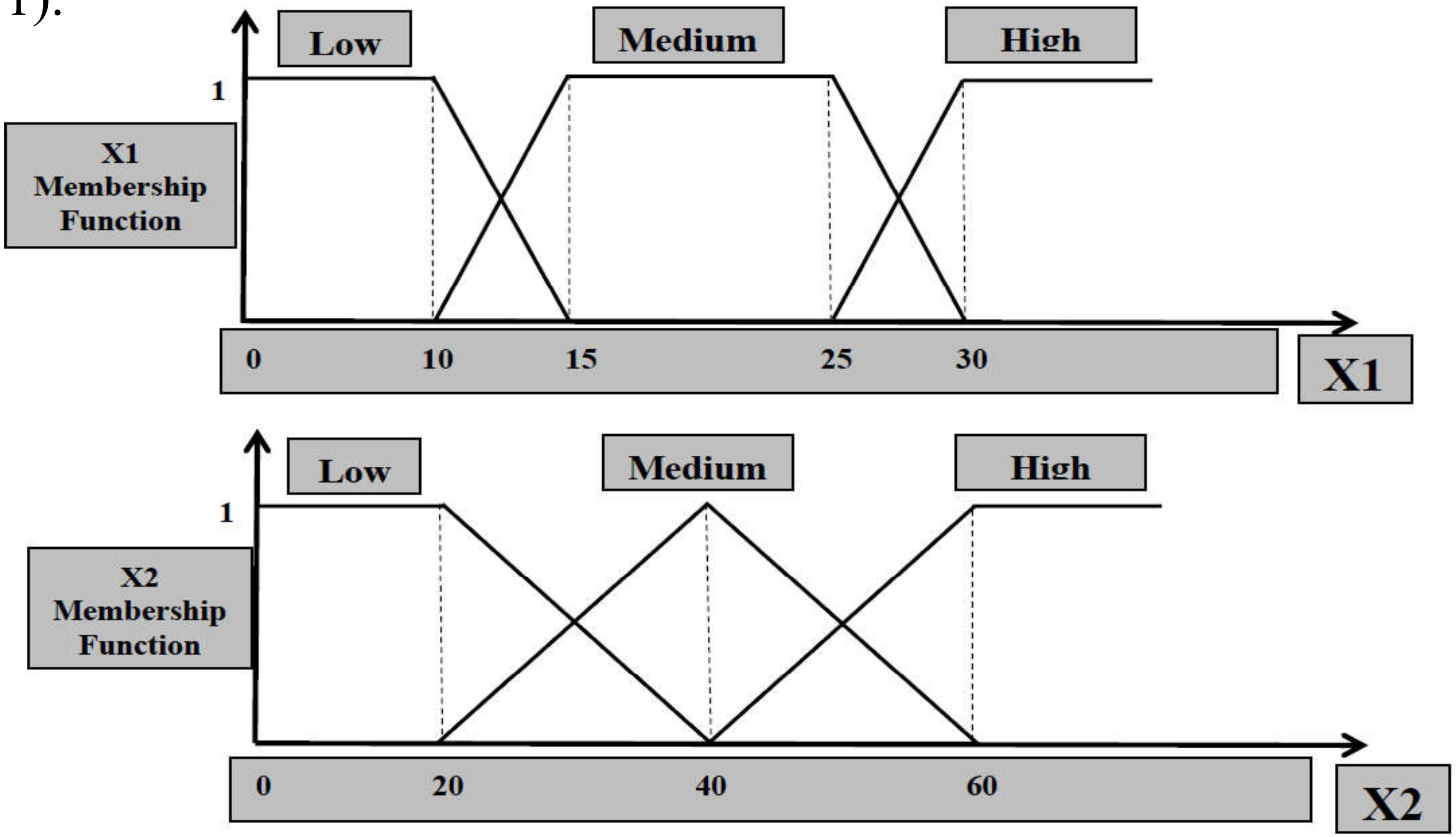
Fuzzy logic has been widely used in

- **Aero Space**: Altitude control of spacecraft, Satellite altitude control, etc.
- **Automobile**: Fuzzy systems for idle speed control, Intelligent highway systems, Traffic control, etc.
- **Defence**: Underwater target recognition, Automatic target recognition of thermal infrared images, Naval decision support aids, Control of a hypervelocity interceptor
- **Industry**: Water purification plant control, Quantitative pattern analysis, etc.
- **Medical**: Medical diagnostic support system, Radiology diagnoses, etc.
- **Electronics**: Air conditioning systems, Washing machine timing, Microwave ovens, Vacuum cleaners, etc.

7.7 Fuzzy Controller and Applications

Example: A Fuzzy Logic Controller (FLC) has two antecedents (X1 and X2) and one consequent (Y1).

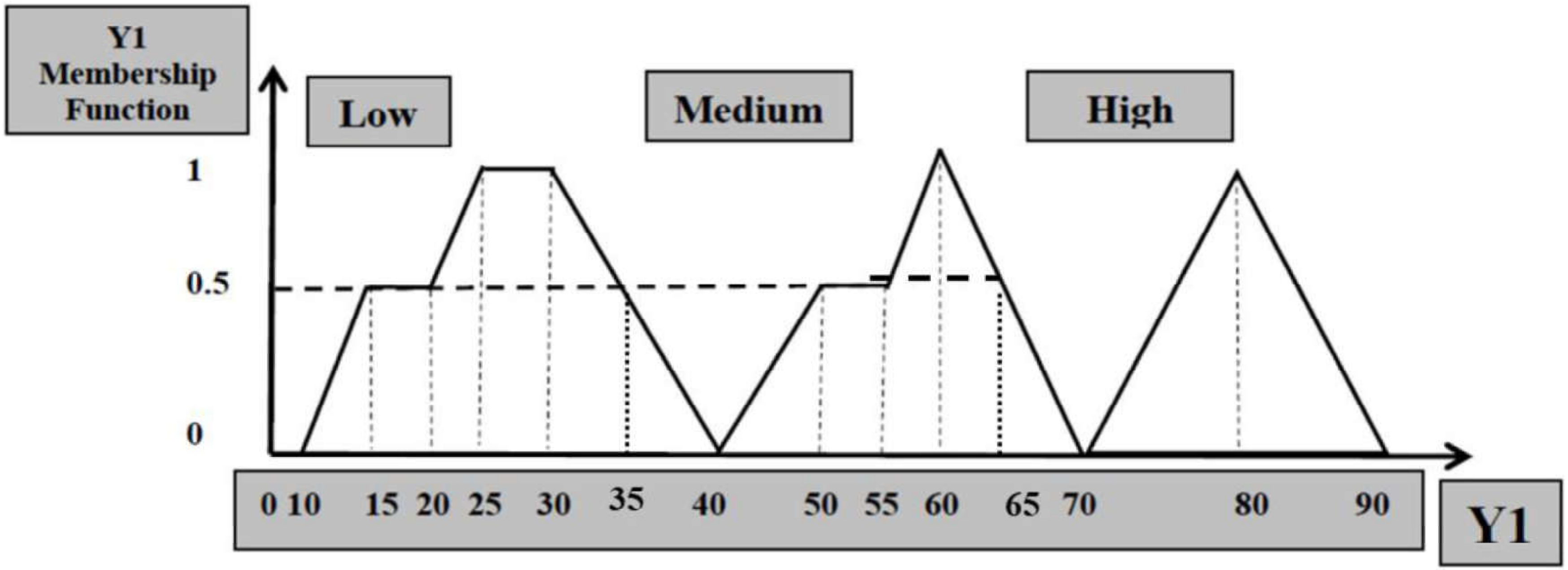
As shown in the right figure, each antecedent (X1 or X2) is represented by 3 fuzzy sets {Low, Medium, High} with trapezoid shape or triangular shape respectively.



.

7.7 Fuzzy Controller and Applications

The consequent is represented by 3 fuzzy sets{Low, Medium, High} as shown in the figure below.



7.7 Fuzzy Controller and Applications

The rule base for this FLC is as follows:

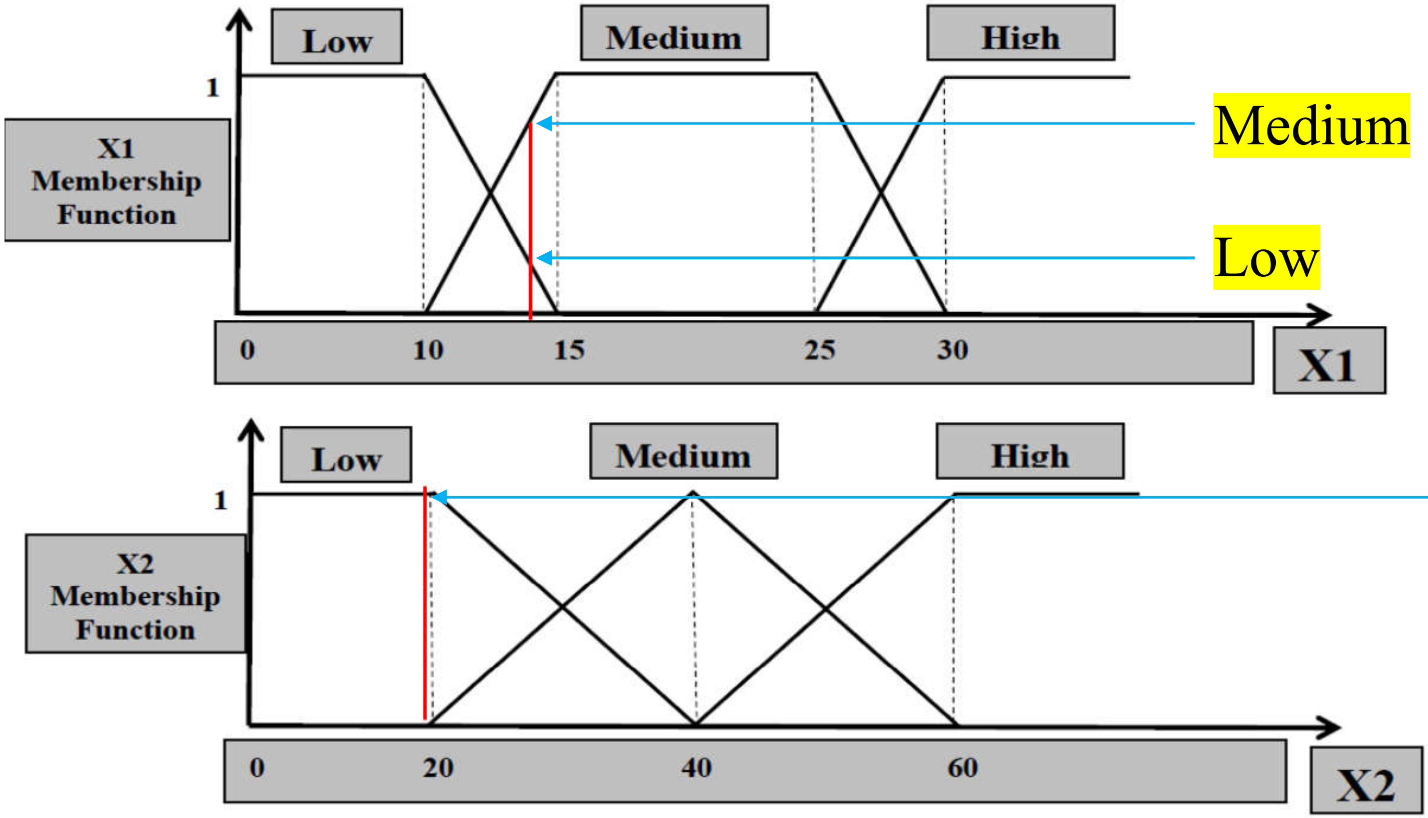
- Rule 1: IF X1 is Low and X2 is Low Then Y1 is Low
- Rule 2: IF X1 is Low and X2 is Medium Then Y1 is High
- Rule 3: IF X1 is Low and X2 is High Then Y1 is Medium
- Rule 4: IF X1 is Medium and X2 is Low Then Y1 is Medium
- Rule 5: IF X1 is Medium and X2 is Medium Then Y1 is Low
- Rule 6: IF X1 is Medium and X2 is High Then Y1 is Medium
- Rule 7: IF X1 is High and X2 is Low Then Y1 is Low
- Rule 8: IF X1 is High and X2 is Medium Then Y1 is High
- Rule 9: IF X1 is High and X2 is High Then Y1 is High

Table presentation of Rule Base

No.	X1	X2	Y1
1	Low	Low	Low
2	Low	Medium	High
3	Low	High	Medium
4	Medium	Low	Medium
5	Medium	Medium	Low
6	Medium	High	Medium
7	High	Low	Low
8	High	Medium	High
9	High	High	High

7.7 Fuzzy Controller and Applications

Assume that the current crisp inputs to the FLC are X1=13 and X2=20, as RED lines shown in the figure below.



1) Find the rules to be fired:

For X1=13, its fuzzy membership values is Low and Medium. For X2=20, its fuzzy membership value is Low. Therefore, Rule 1 and Rule 4 will be fired.

7.7 Fuzzy Controller and Applications

2) For each fired rule, calculate the firing strength using the Product operation to represent the AND logical connective.

For X1 and Low fuzzy set, b=10, c=15, the equation for a triangular membership function is:

1	$x \le b$
(c-x)/(c-b)	$b \le x \le c$

Therefore, the membership value is (15-13)/(15-10)=0.4

For X1 and the Medium fuzzy set, the equation for trapezoid membership function is shown at right. As a=10, b=15, c=25, d=30,

The membership value is (13-10)/(15-10)=0.6

(x-a)/b-a)	$a \le x \le b$
1	$b \le x \le c$
(d-x)/(d-c)	$c \le x \le d$
0 otherwise	

7.7 Fuzzy Controller and Applications

<u>For X2 and Low fuzzy set:</u>

The equation for a triangular membership function is

$$1 \qquad x \le b$$

As X2 =20, the membership value to the Low fuzzy sets is 1.0.

<u>To calculate the firing strength:</u>

Rule 1: Using the product operation to represent the AND logical connective, its firing strength is 0.4*1.0 = 0.4.

Rule 4: Using the product operation to represent the AND logical connective, its firing strength is 0.6*1.0 = 0.6.

8 Summary

- The scientific programming is mainly dealing with offline data and not interacted with the real-world in real time.
- The robot programming needs real-time performance, robustness, modularity, concurrency, and intelligence.
- Robot tasks can be decomposed into smaller tasks such as modelling, planning, localisation, perception, action, communication & learning.
- Languages for robotic systems: assembly language & high-level languages.
  - -- ***Assembly languages*** are processor dependent and widely used in small systems.
  - -- ***High-level languages*** are easy to develop, but demand high computing power.
- C/C++ is a most popular language being used in robotic systems.
- PID algorithms are widely deployed to control robot motions and manipulations.

8 Summary

- The Fuzzy logic methodology appears very useful
  - when the processes are too complex for analysis by conventional quantitative techniques or
  - when available sources of information are interpreted qualitatively, inexactly or uncertainly.
- Fuzzy logic is useful for controlling non-linear systems or systems that cannot be described mathematically.
- Fuzzy logic has been applied in numerous application areas:
  - Rice cookers, subway control, air-conditioning, braking,
  - Automatic camera focusing, colour adjustment of TVs,
  - Lifts, shower temperature control, vehicle speed and steering control