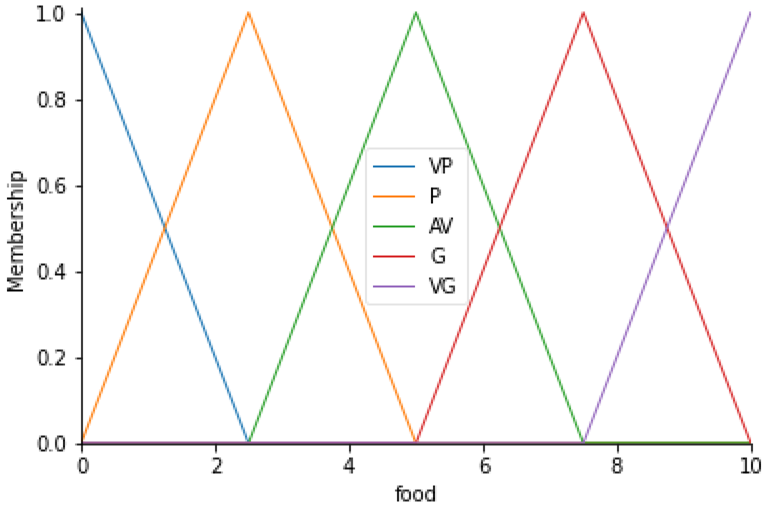
ARTIFICIAL INTELLIGENCE
FUZZY LOGIC

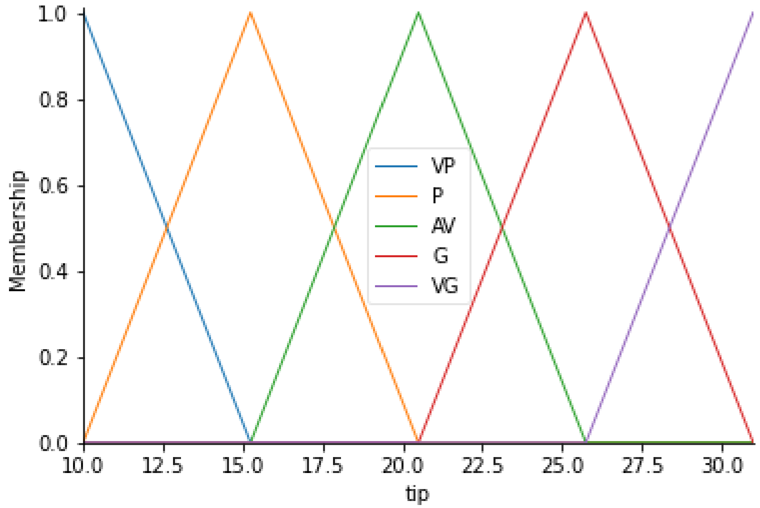
Prof. Nguyen Truong Thinh

Tipping in Restaurant

```
import numpy as np
import skfuzzy.control as ctrl

#Không gian phổ quát giúp tính toán nhanh hơn mà không phải quan tâm độ chính xác.
#Chỉ có những điểm quan trọng được bao gồm ở đây;
#làm cho nó có độ phân giải cao hơn là không cần thiết.
universe = np.linspace(0, 10, 61)
universe1 = np.linspace(10, 31, 61)

# Tạo ra 3 biến mờ: 2 vào 1 ra
food = ctrl.Antecedent(universe, 'food')
service = ctrl.Antecedent(universe, 'service')
tip = ctrl.Consequent(universe1, 'tip')

# Chúng ta đặt VP: Very Poor, P: Poor, AV; Average, G: Goog, VG: very Goog
names = ['VP', 'P', 'AV', 'G', 'VG']
food.automf(names=names)
service.automf(names=names)
tip.automf(names=names)
```

Tipping in Restaurant





Tipping in Restaurant

```
rule0 = ctrl.Rule(antecedent= ((food['VP'] & service['VP']) |
                               (food['P'] & service['VP']) |
                               (food['VP'] & service['P'])),
                  consequent=tip['VP'], label='rule VP')

rule1 = ctrl.Rule(antecedent=((food['VP'] & service['AV']) |
                               (food['VP'] & service['G']) |
                               (food['P'] & service['P']) |
                               (food['P'] & service['AV']) |
                               (food['AV'] & service['P']) |
                               (food['AV'] & service['VP']) |
                               (food['G'] & service['VP'])),
                  consequent=tip['P'], label='rule P')
```

Tipping in Restaurant

```
rule2 = ctrl.Rule(antecedent=((food['VP'] & service['VG']) |
                              (food['P'] & service['G']) |
                              (food['AV'] & service['AV']) |
                              (food['G'] & service['P']) |
                              (food['VG'] & service['VP'])),
                  consequent=tip['AV'], label='rule AV')

rule3 = ctrl.Rule(antecedent=((food['P'] & service['VG']) |
                              (food['AV'] & service['VG']) |
                              (food['AV'] & service['G']) |
                              (food['G'] & service['G']) |
                              (food['G'] & service['AV']) |
                              (food['VG'] & service['AV']) |
                              (food['VG'] & service['P'])),
                  consequent=tip['G'], label='rule G')

rule4 = ctrl.Rule(antecedent=((food['G'] & service['VG']) |
                              (food['VG'] & service['VG']) |
                              (food['VG'] & service['G'])),
                  consequent=tip['VG'], label='rule VG')
```

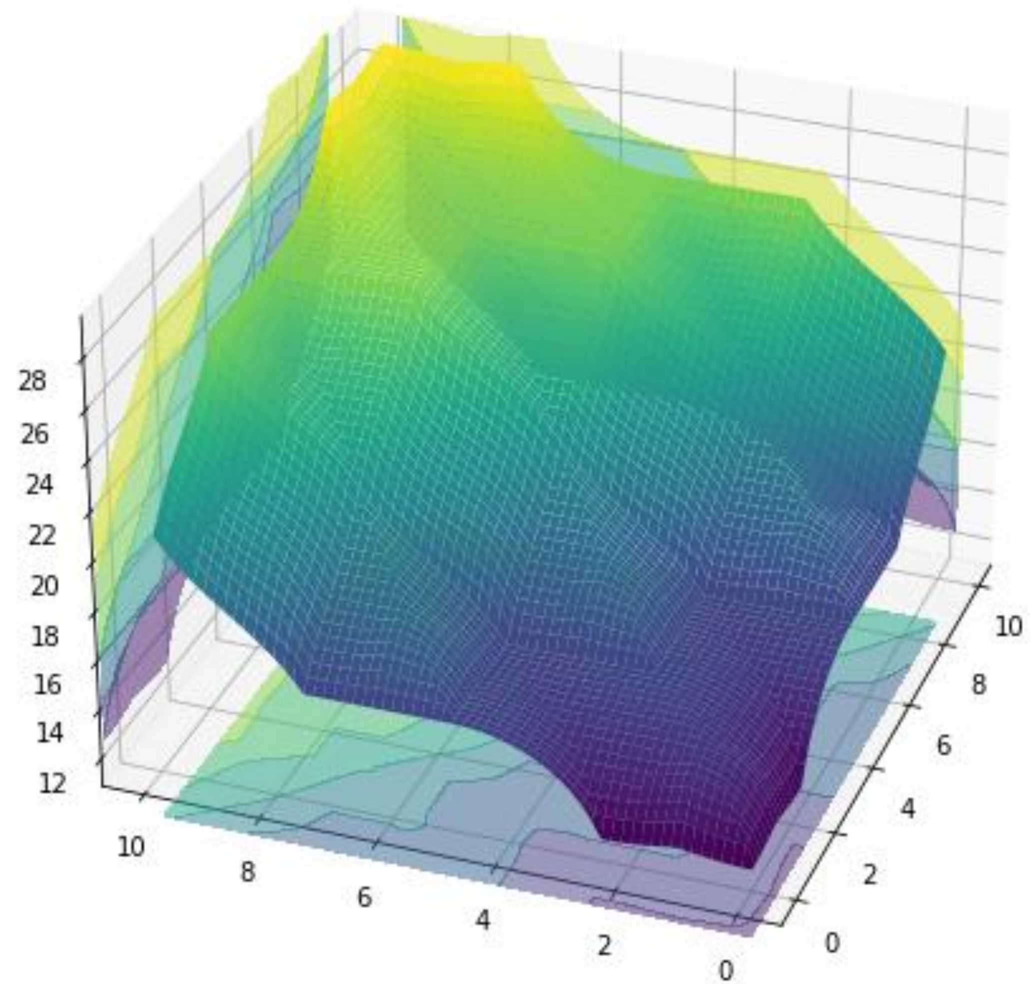
Tipping in Restaurant

```
system = ctrl.ControlSystem(rules=[rule0, rule1, rule2, rule3, rule4])
```

```
sim = ctrl.ControlSystemSimulation(system, flush_after_run=61 * 61 + 1)
```

```python
upsampled = np.linspace(0, 10, 61)
x, y = np.meshgrid(upsampled, upsampled)
z = np.zeros_like(x)

for i in range(61):
    for j in range(61):
        sim.input['food'] = x[i, j]
        sim.input['service'] = y[i, j]
        sim.compute()
        z[i, j] = sim.output['tip']

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')

surf = ax.plot_surface(x, y, z, rstride=1, cstride=1, cmap='viridis',
                       linewidth=0.4, antialiased=True)

cset = ax.contourf(x, y, z, zdir='z', offset=9, cmap='viridis', alpha=0.5)
cset = ax.contourf(x, y, z, zdir='x', offset=11, cmap='viridis', alpha=0.5)
cset = ax.contourf(x, y, z, zdir='y', offset=11, cmap='viridis', alpha=0.5)

ax.view_init(30, 200)
```

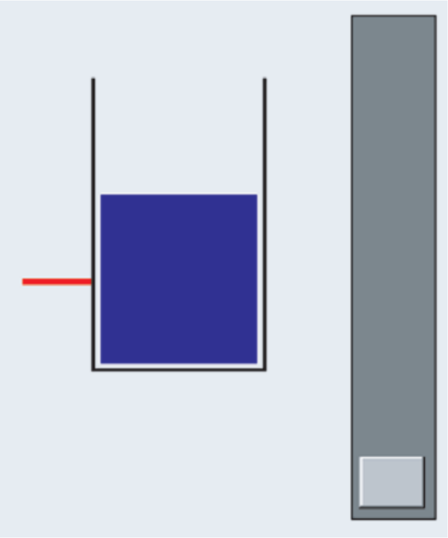
Tipping in Restaurant

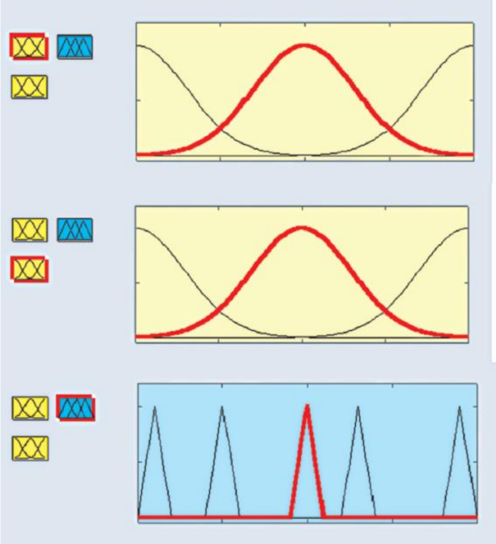


Ex 1: *water level control*

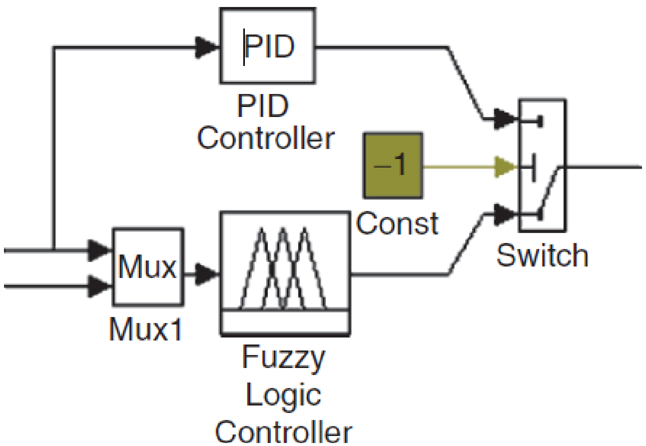
- Rules should be as follows:

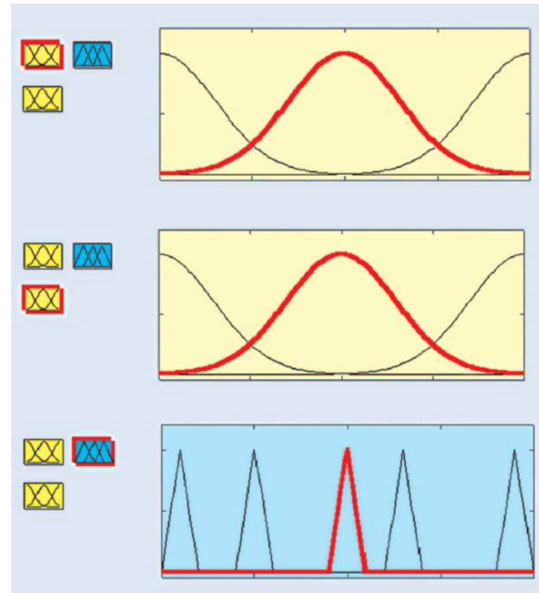
  1. IF (level is okay) THEN (valve is no change) (1)
  2. IF (level is low) THEN (valve is open fast) (1)
  3. IF (level is high)THEN (valve is close fast) (1)
- 4. IF (level is okay) and (rate is positive) THEN (valve is close slow) (1)
- 5. IF (level is okay) and (rate is negative) THEN (valve is open slow) (1)

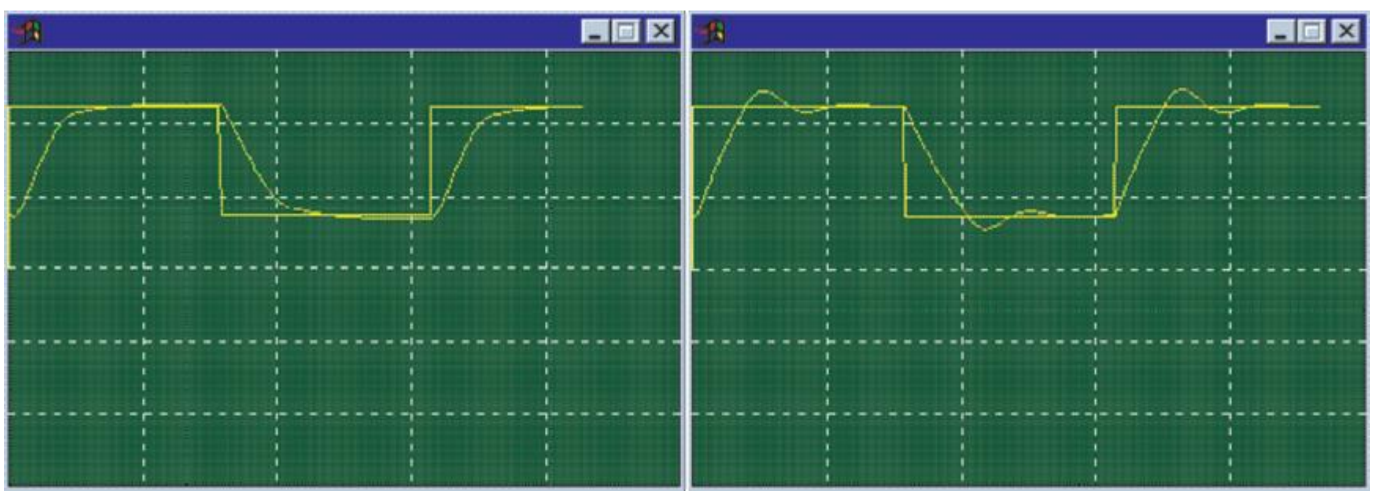


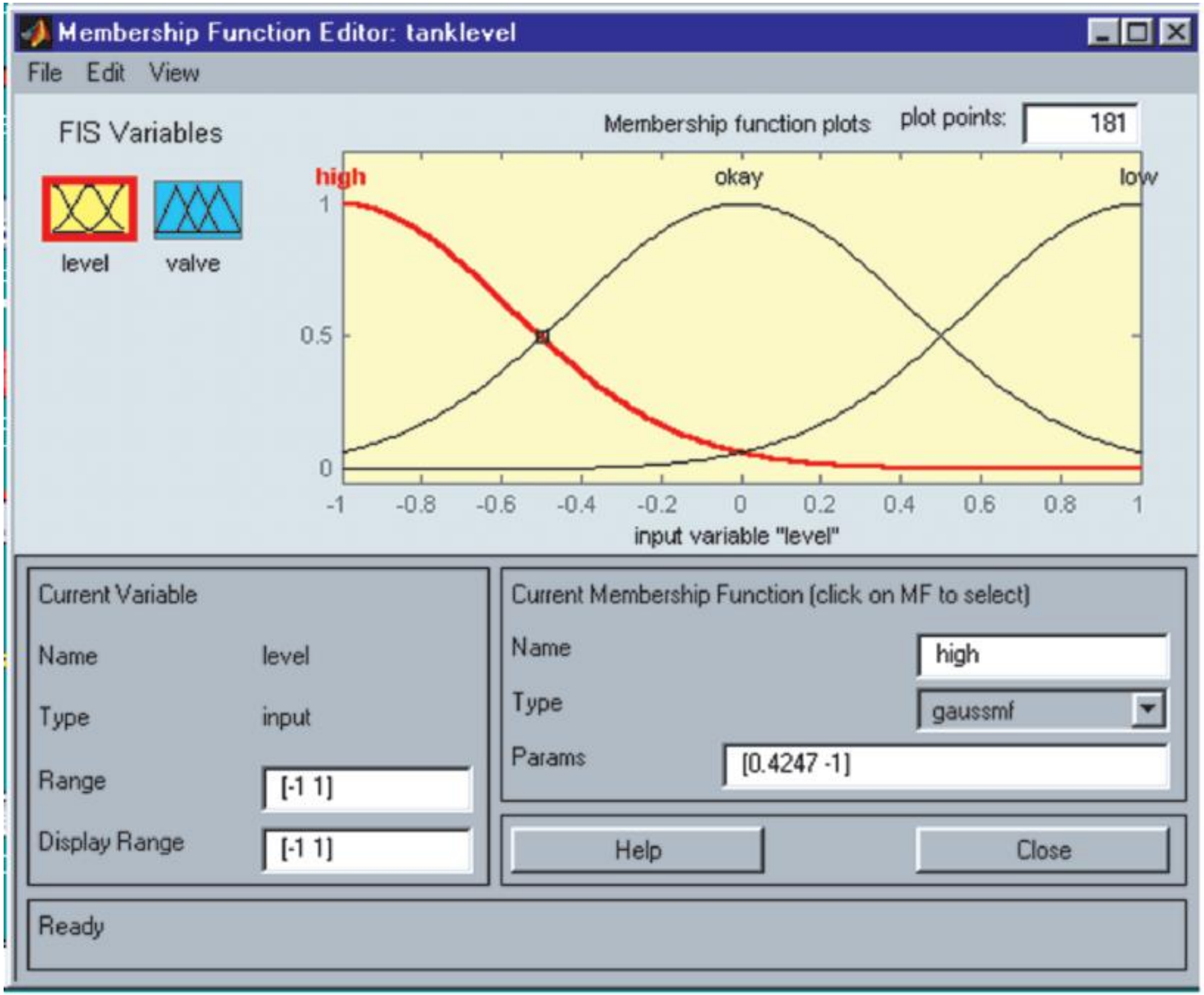


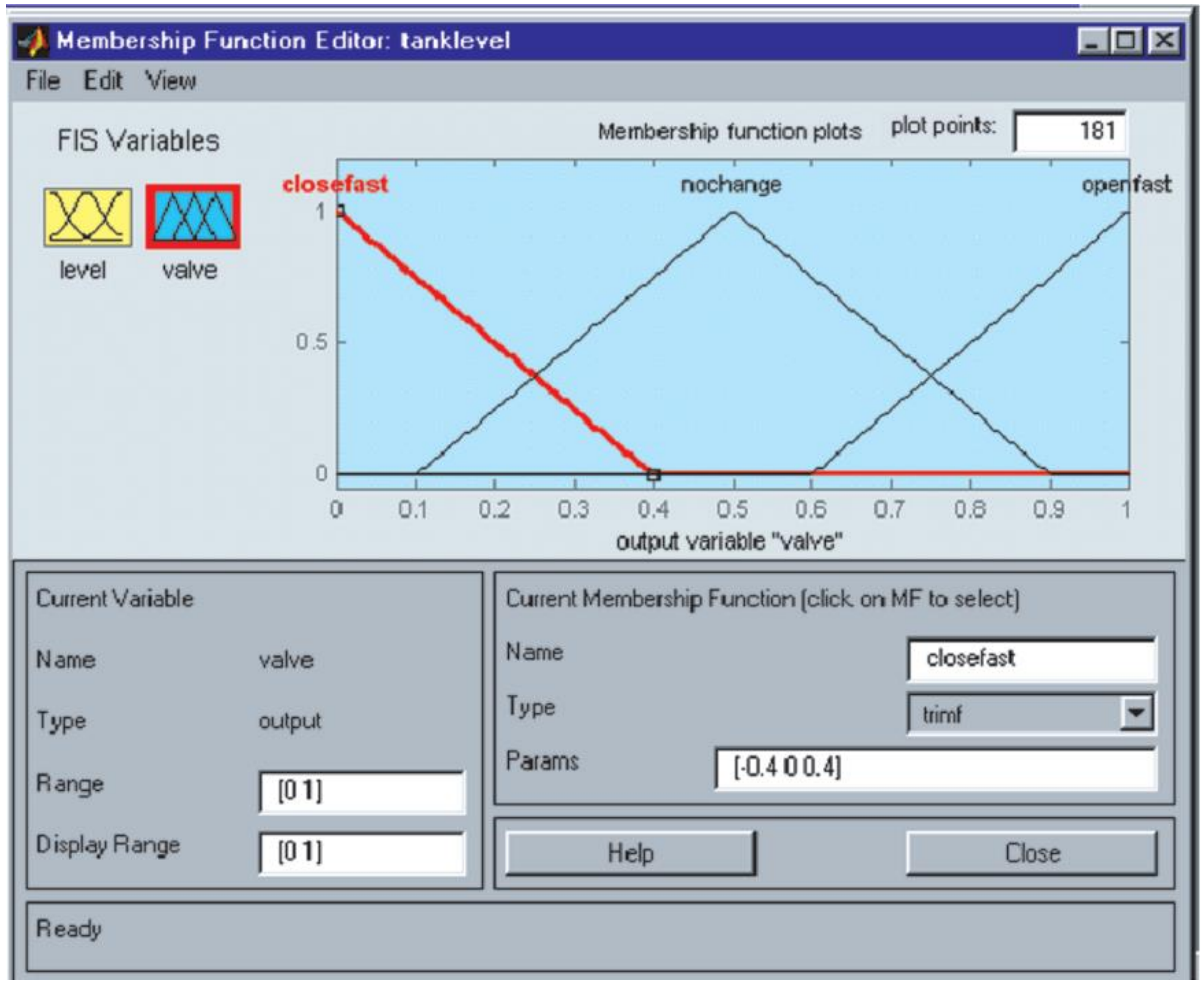
Fuzzy Controllers: *water level control*

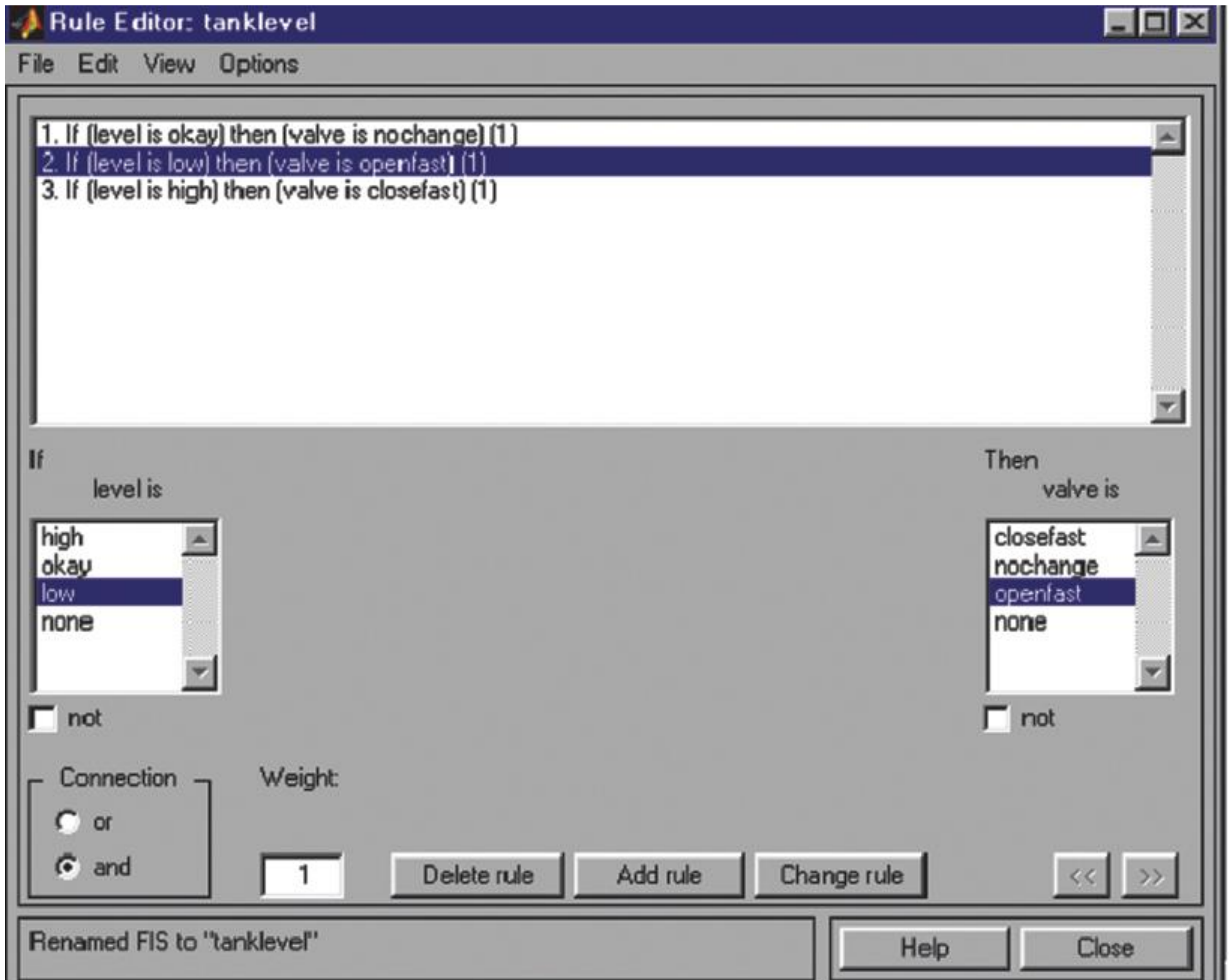


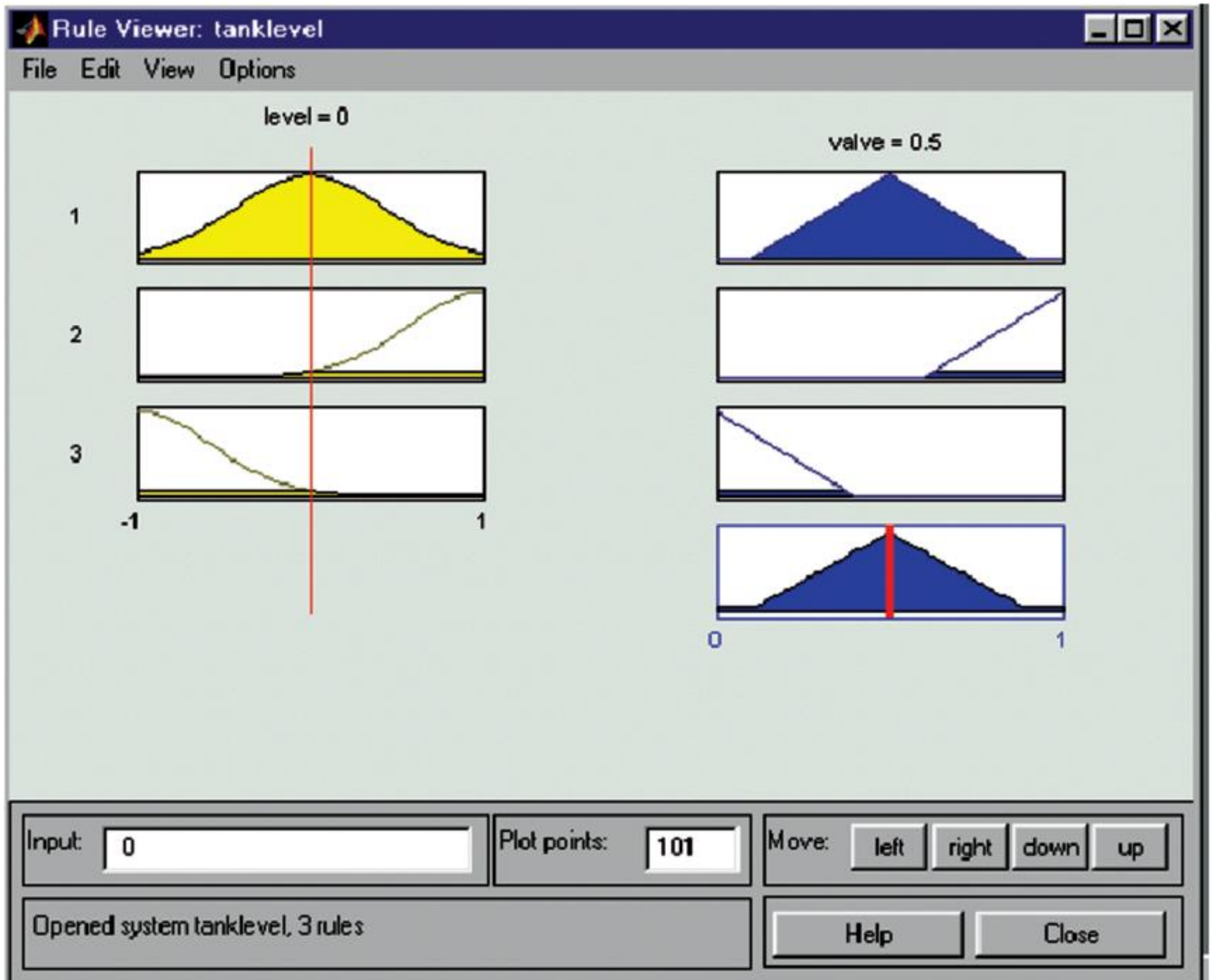


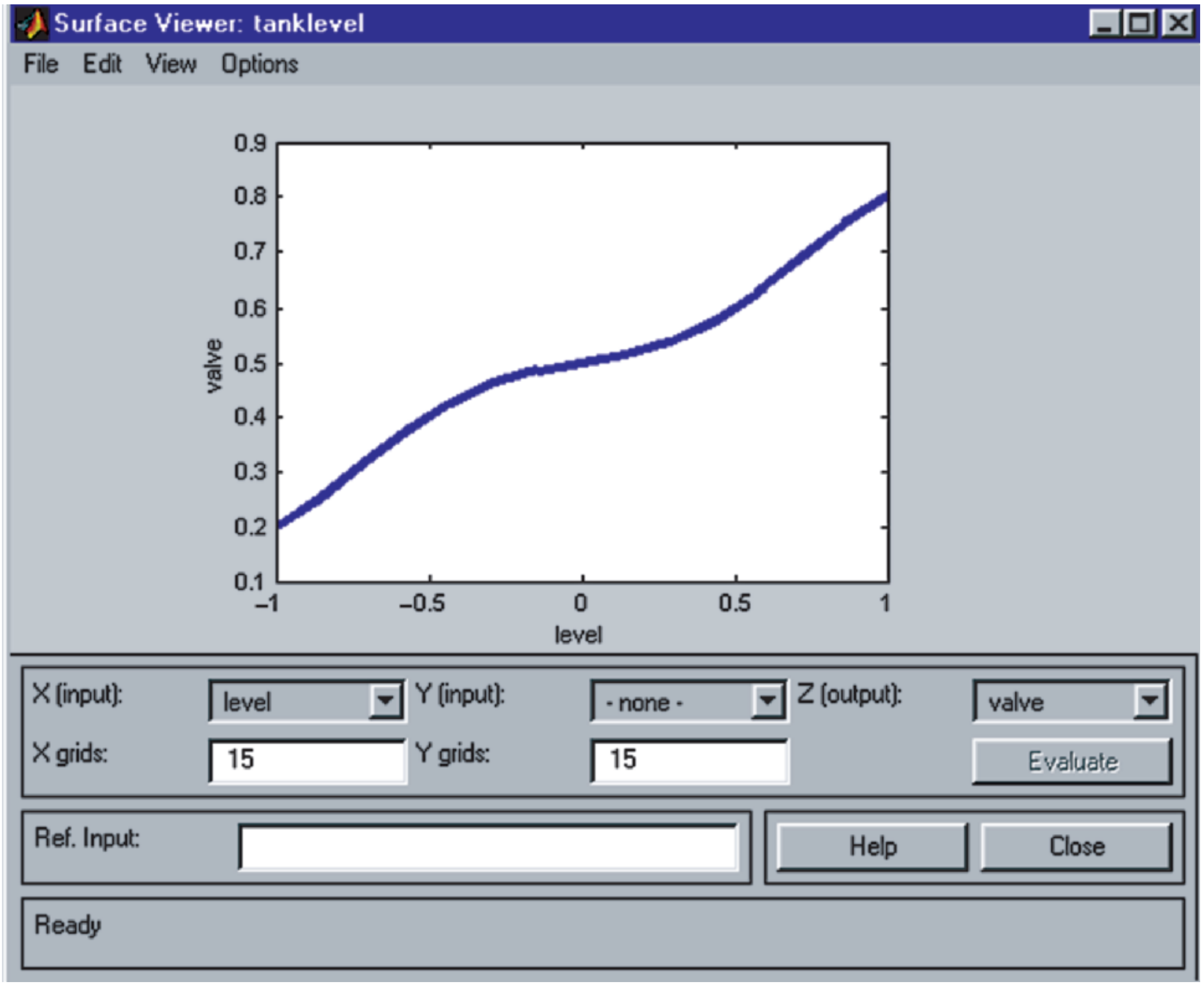



Membership Function Editor: tanklevel
File Edit View
FIS Variables
level
valve
Membership function plots
plot points:
181
high
okay
low
1
0.5
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
input variable "level"
Current Variable
Name
level
Type
input
Range
[-1 1]
Display Range
[-1 1]
Current Membership Function (click on MF to select)
Name
high
Type
gaussmf
Params
[0.4247 -1]
Help
Close
Ready

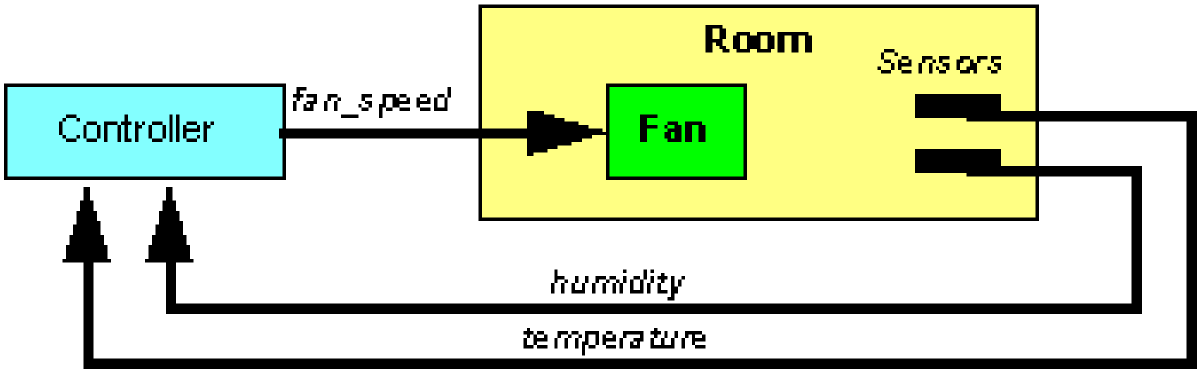

Membership Function Editor: tanklevel
File
Edit
View
FIS Variables
level
valve
Membership function plots
plot points:
181
closefast
nochange
openfast
1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
output variable "valve"
Current Variable
Name
valve
Type
output
Range
[0 1]
Display Range
[0 1]
Current Membership Function (click on MF to select)
Name
closefast
Type
trimf
Params
[-0.4 0 0.4]
Help
Close
Ready


Rule Editor: tanklevel
File Edit View Options
1. If (level is okay) then (valve is nochange) (1)
2. If (level is low) then (valve is openfast) (1)
3. If (level is high) then (valve is closefast) (1)
If
level is
high
okay
low
none
not
Then
valve is
closefast
nochange
openfast
none
not
Connection
or
and
Weight:
1
Delete rule
Add rule
Change rule
<<
>>
Renamed FIS to "tanklevel"
Help
Close


Rule Viewer: tanklevel
File
Edit
View
Options
level = 0
valve = 0.5
1
2
3
-1
1
0
1
Input:
0
Plot points:
101
Move:
left
right
down
up
Opened system tanklevel, 3 rules
Help
Close


Surface Viewer: tanklevel
File Edit View Options
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
valve
–1
–0.5
0
0.5
1
level
X (input):
level
Y (input):
- none -
Z (output):
valve
X grids:
15
Y grids:
15
Evaluate
Ref. Input:
Help
Close
Ready

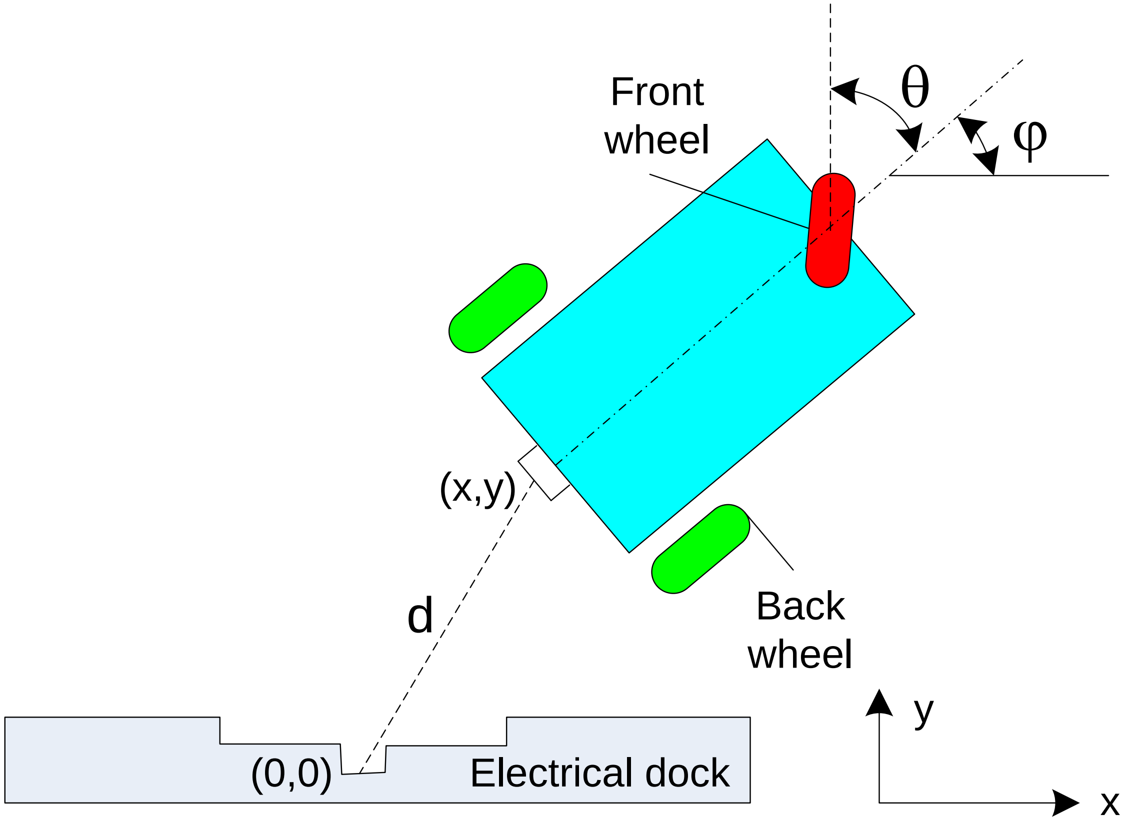
Ex 2: Temperature Controller



- IF temperature IS cold AND humidity IS high THEN fan_spd IS high
  IF temperature IS cool AND humidity IS high THEN fan_spd IS medium
  IF temperature IS warm AND humidity IS high THEN fan_spd IS low
  IF temperature IS hot AND humidity IS high THEN fan_spd IS zero
- IF temperature IS cold AND humidity IS med THEN fan_spd IS medium
  IF temperature IS cool AND humidity IS med THEN fan_spd IS low
  IF temperature IS war m AND humidity IS med THEN fan_spd IS zero
  IF temperature IS hot AND humidity IS med THEN fan_spd IS zero
- IF temperature IS cold AND humidity IS low THEN fan_spd IS medium
  IF temperature IS cool AND humidity IS low THEN fan_spd IS low
  IF temperature IS warm AND humidity IS low THEN fan_spd IS zero
  IF temperature IS hot AND humidity IS low THEN fan_spd IS zero

Ex 3: Mobile Robot

- Design a Fuzzy Logic Controller (FLC) able to back up a mobile robot into a docking stration from any initial position that has enough clearance from the docking station.

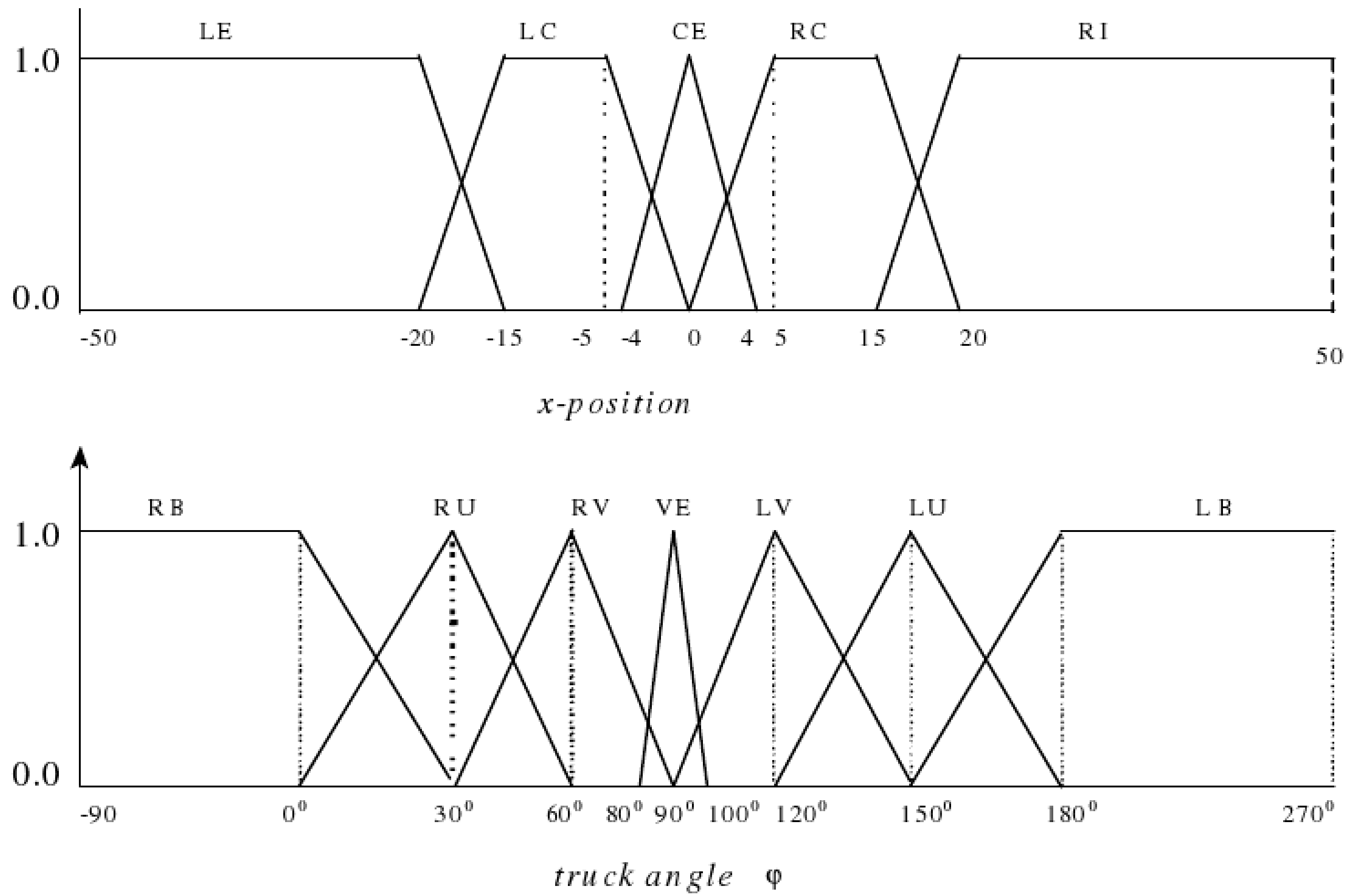


$$\dot{x} = -v\cos(\varphi)$$

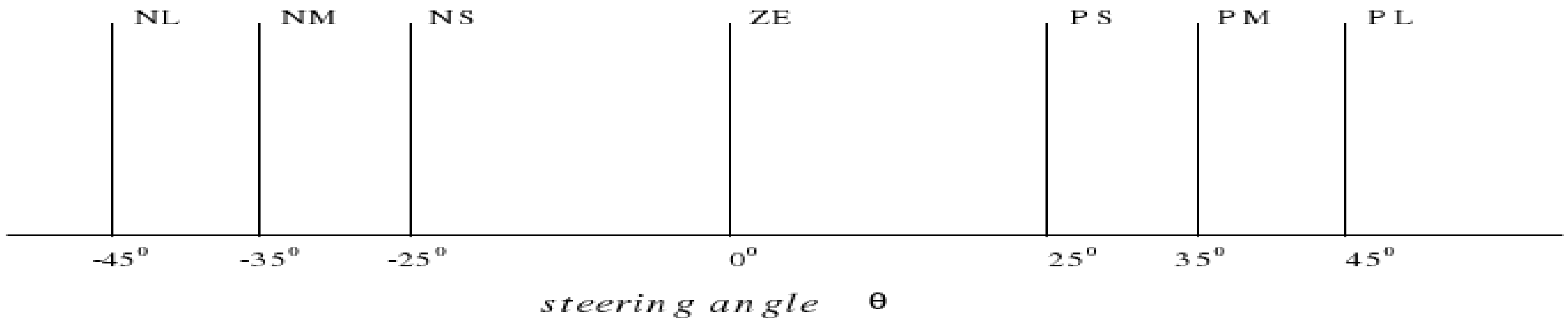
$$\dot{y} = -v\cos(\varphi)$$

$$\dot{\varphi} = -\frac{v}{l}\sin\theta$$

Mobile robot



Mobile robot



φ \ x	LE	LC	CE	RC	RI
RL	NL [1]	NL [2]	NM [3]	NM [4]	NS [5]
RU	NL [6]	NL [7]	NM	NS	PS
RV	NL	NM	NS	PS	PM
VE	NM	NM	ZE [18]	PM	PM
LV	NM	NS	PS	PM	PL
LU	NS	PS	PM	PL	PL [30]
LL	PS [31]	PM [32]	PM [33]	PL [34]	PL [35]

Ex 4: A fuzzy vacuum cleaner

- Let us try to develop the rules table for the fuzzy controller of a vacuum cleaner. This controller should regulate the force of sucking dust from a surface being cleaned. This force can be described as a linguistic variable with values: *very strong, strong, ordinary, weak, very weak. The input of this controller should* obviously consider an amount of dust on the surface. The surface can be *very dirty, dirty, rather dirty, almost clean, clean. The* controller can change the force depending on how dirty the surface is. One can propose the following set of rules to describe the controller operation:
  - **if surface is *very dirty then force is very strong,***
  - **if surface is *dirty then force is strong,***
  - **if surface is *rather dirty then force is ordinary,***
  - **if surface is *almost clean then force is weak,***
  - **if surface is *clean then force is very weak.***

A fuzzy vacuum cleaner

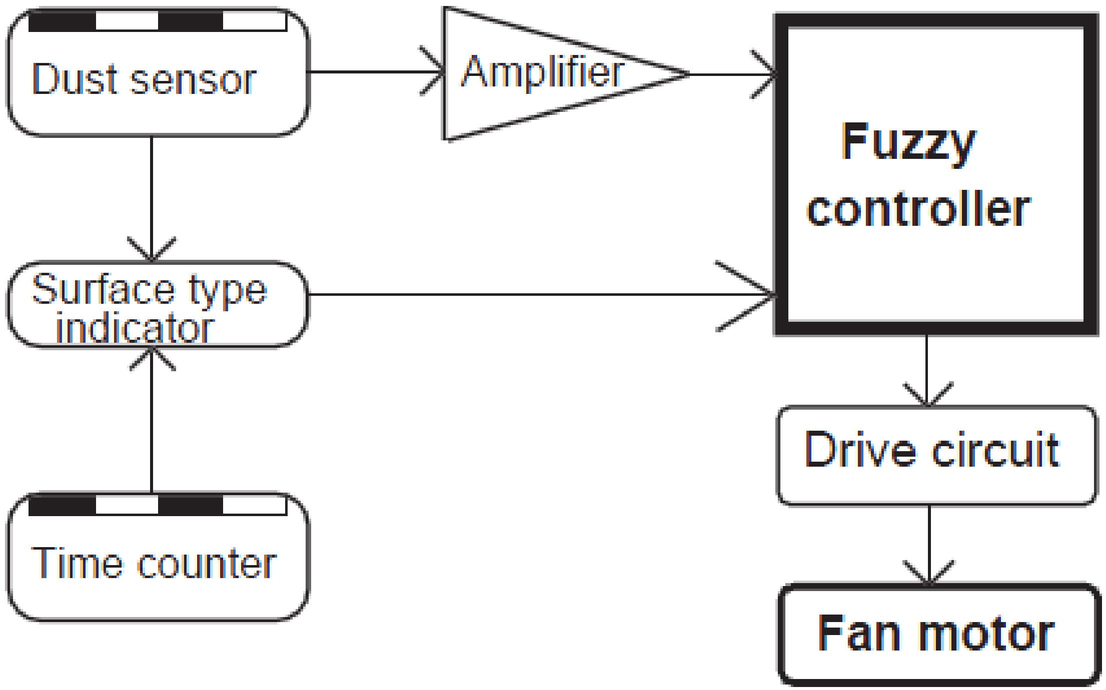
Table 3.3 Rules table for a fuzzy vacuum cleaner

Surface	*Force*
very dirty	very strong
dirty	strong
rather dirty	ordinary
almost clean	weak
clean	very weak

Table 3.4 Rules table for surface type and dust amount

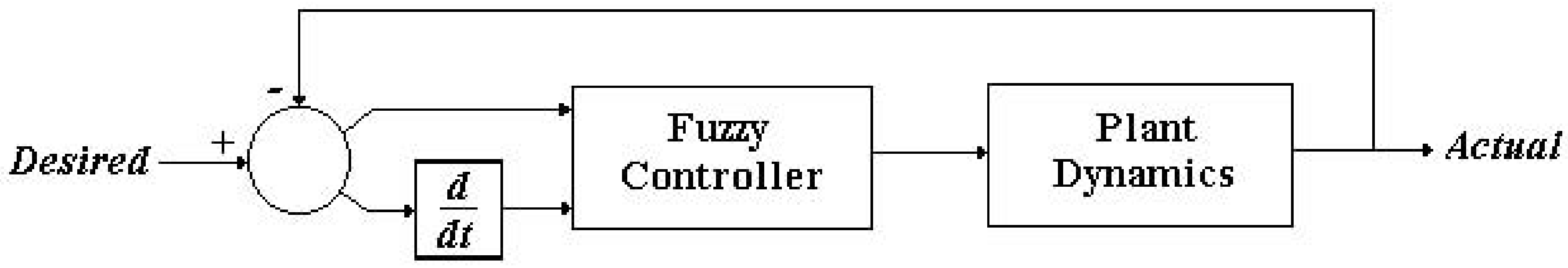
	clean	almost clean	rather dirty	dirty	very dirty
wood	*very weak*	*very weak*	*weak*	*ordinary*	*strong*
tatami	*very weak*	*weak*	*ordinary*	*strong*	*very strong*
carpet	*weak*	*ordinary*	*ordinary*	*strong*	*very strong*

A fuzzy vacuum cleaner



Ex 5: Automobile cruise control

- we discussed the design of an automobile cruise control system using the standard approach. Here we solve the same problem using the Mamdani method, a fuzzy approach. It should become clear that this fuzzy approach, which provides a model-free approach to developing a controller, is simpler.

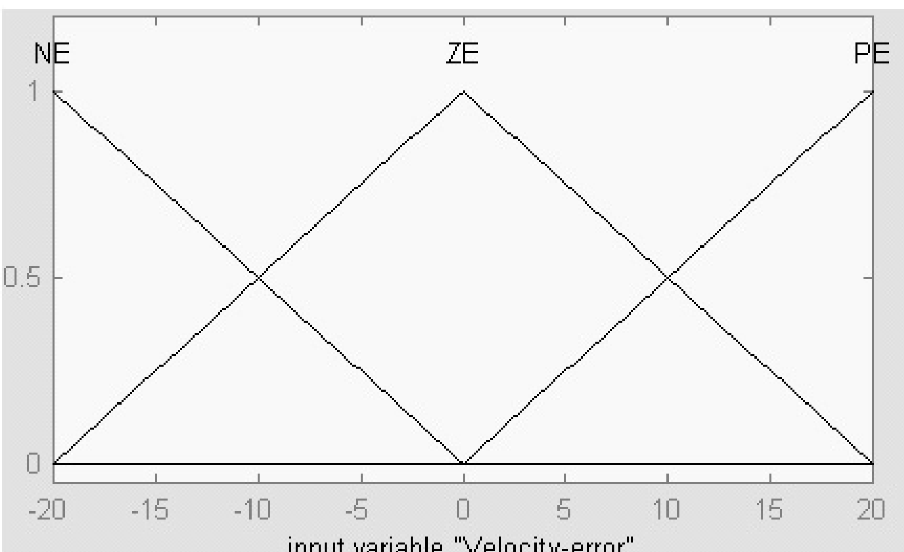


Fuzzy variables: velocity error, acceleration, and engine force.

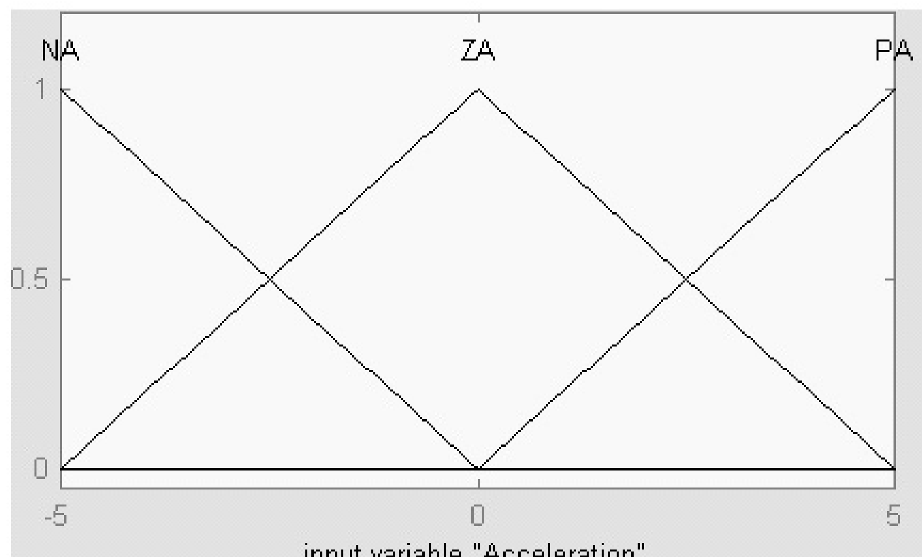
Automobile cruise control

Following are some example rules:

- If velocity error is positive and acceleration is negative then apply maximum force.
- If velocity error is negative and acceleration is positive then apply minimum force.
- If velocity error is zero and acceleration is zero then apply normal force.

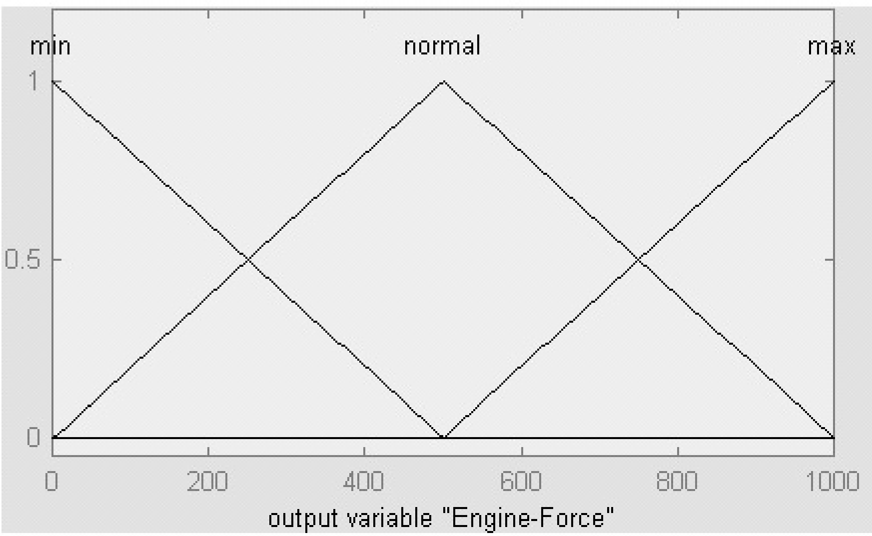


Velocity error



Acceleration

Membership functions

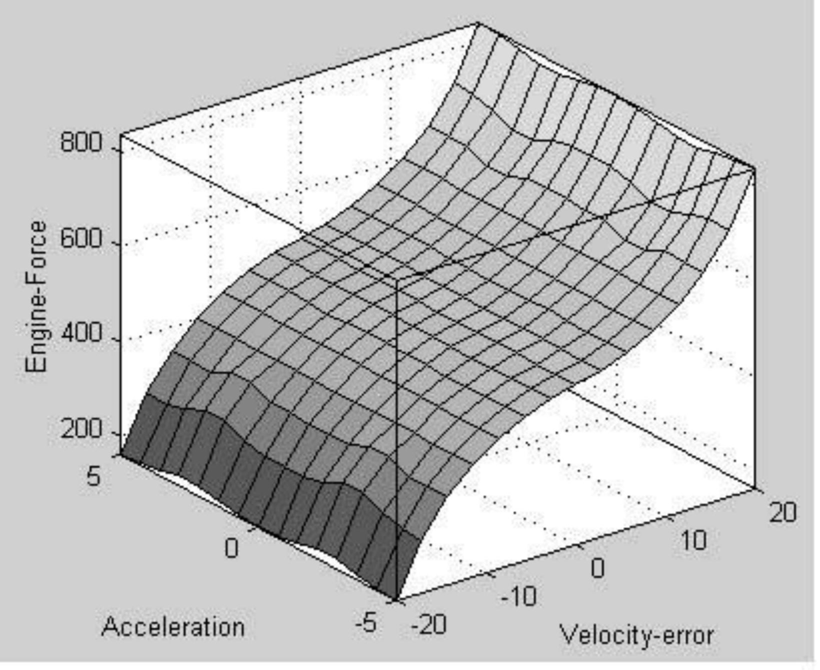


Engine force

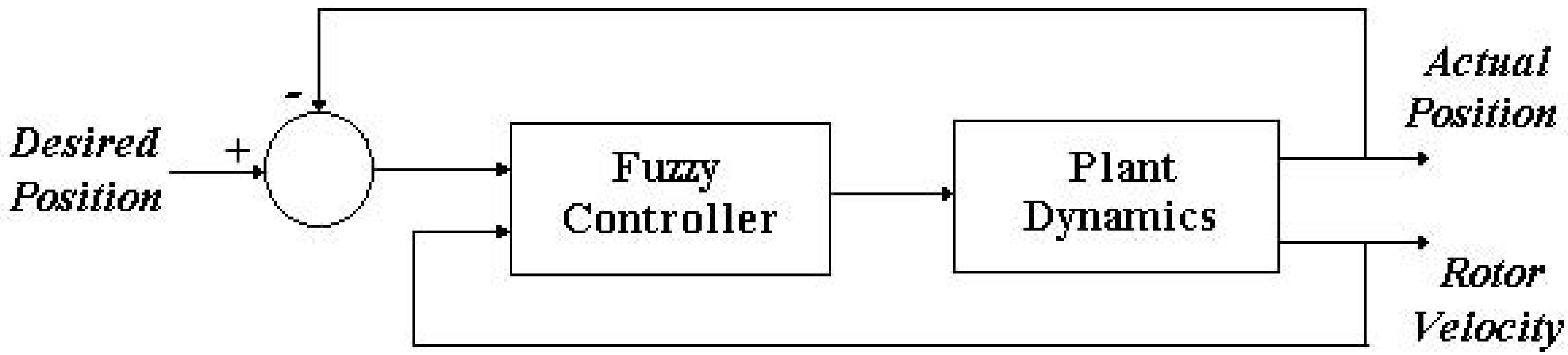
Velocity-Error \ Acceleration	NA	ZA	PA
NE	Min	Min	Min
ZE	Normal	Normal	Normal
PE	Max	Max	Max

Control Rules

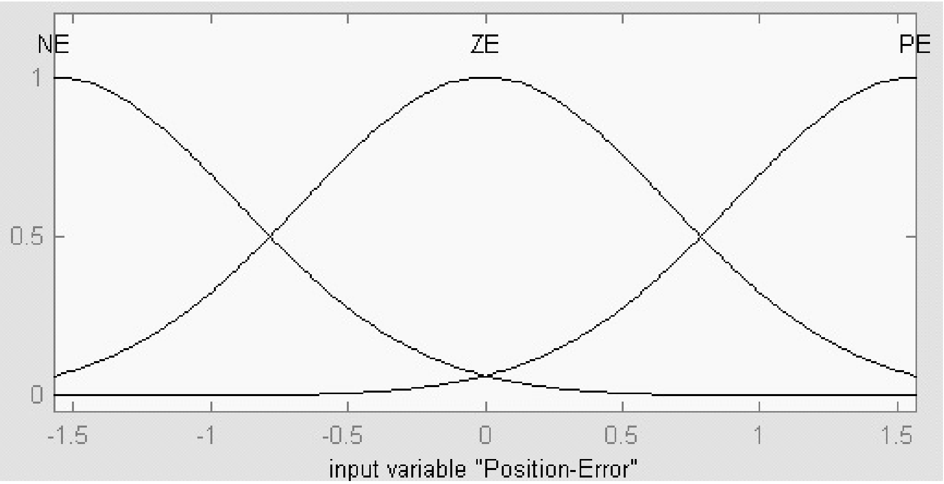
1. If velocity error is NE and acceleration is NA, then engine force is Min.
2. If velocity error is NE and acceleration is ZE, then engine force is Min.
3. If velocity error is NE and acceleration is PA, then engine force is Min.
4. If velocity error is ZE and acceleration is NA, then engine force is Normal.
5. If velocity error is ZE and acceleration is ZA, then engine force is Normal.
6. If velocity error is ZE and acceleration is PA, then engine force is Normal.
7. If velocity error is PE and acceleration is NA, then engine force is Max.
8. If velocity error is PE and acceleration is ZA, then engine force is Max.
9. If velocity error is PE and acceleration is PA, then engine force is Max.

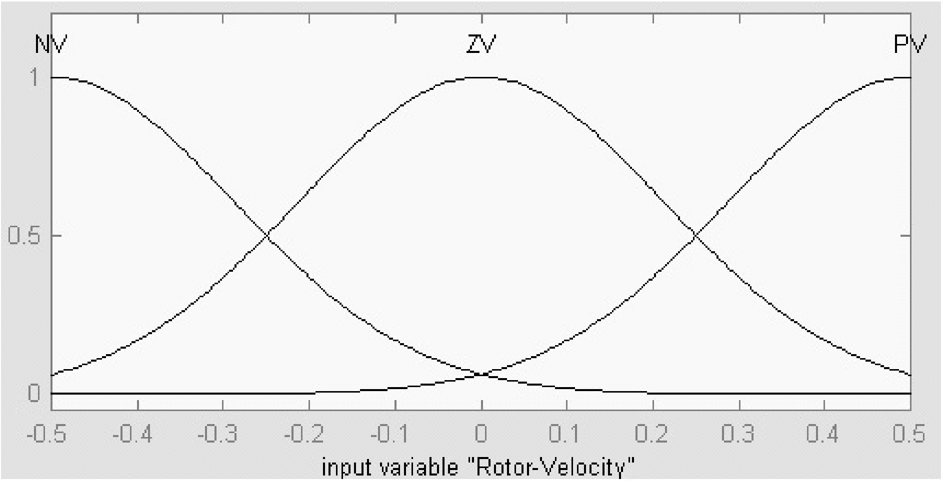


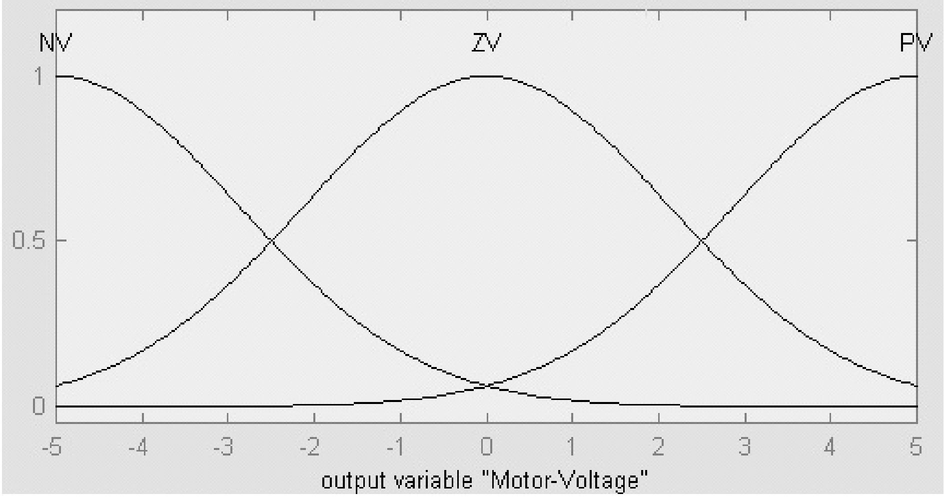
Ex 5: Controlling dynamics of a servomotor



Controlling dynamics of a servomotor







Controlling dynamics of a servomotor

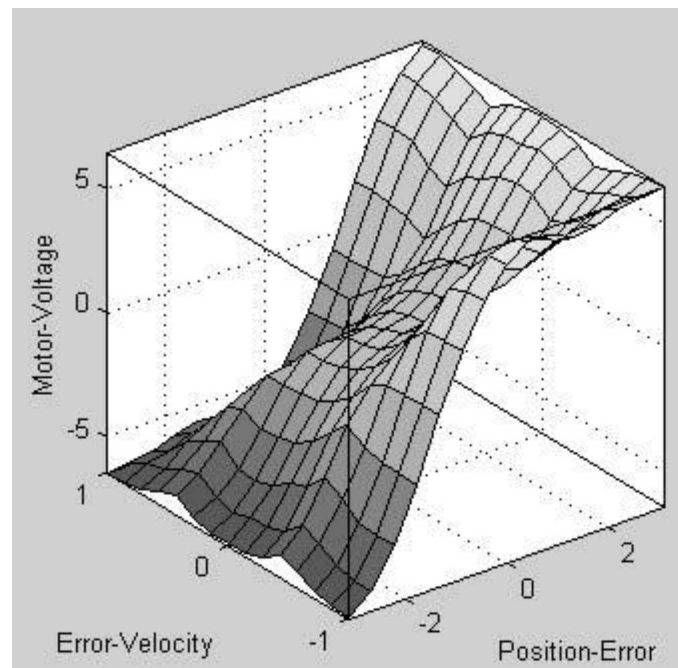
- This knowledge of the system behavior allows us to formulate a set of general rules as, for example,

1. If position error is positive and velocity is negative, then apply positive voltage.
2. If position error is negative and velocity is positive, then apply negative voltage.
3. If position error is zero and velocity is zero, then apply zero voltage.

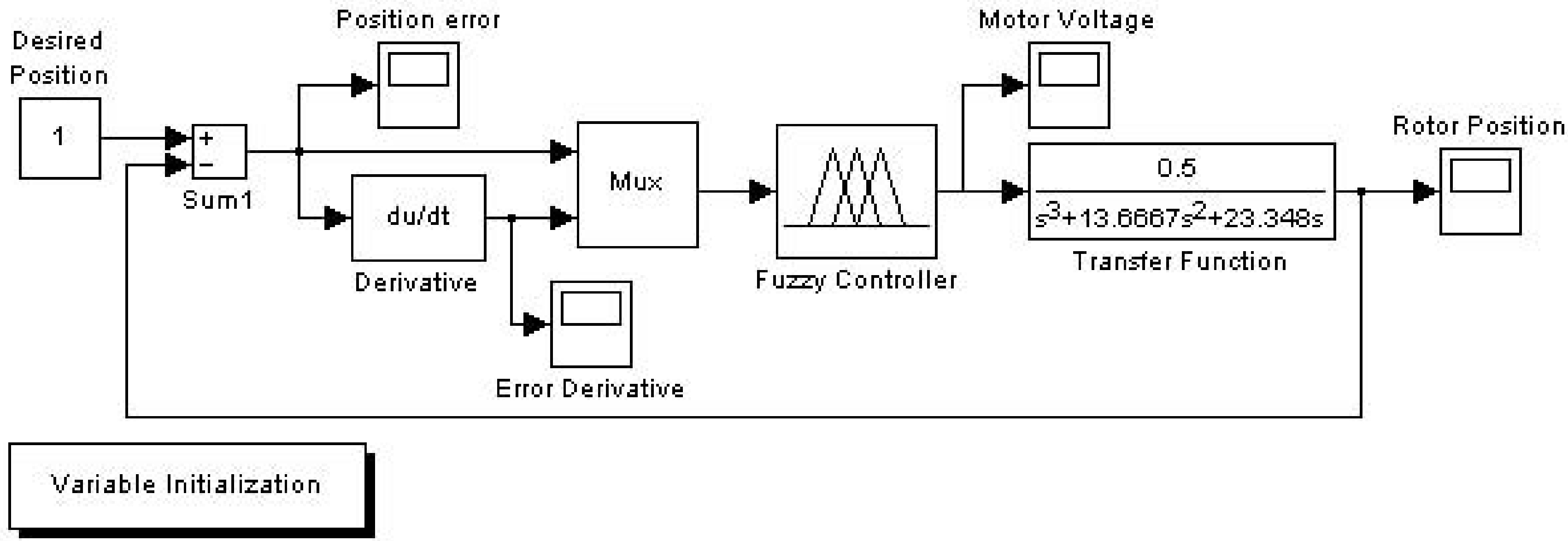
Position-Error \ Error-Velocity	*NV*	*ZV*	*PV*
NE	*Negative Voltage*	*Negative Voltage*	*Negative Voltage*
ZE	*Negative Voltage*	*Zero Voltage*	*Positive Voltage*
PE	*Positive Voltage*	*Positive Voltage*	*Positive Voltage*

Controlling dynamics of a servomotor

1. If position error is NE and velocity is NA, then motor voltage is Negative.
2. If position error is NE and velocity is ZE, then motor voltage is Negative.
3. If position error is NE and velocity is PA, then motor voltage is Negative.
4. If position error is ZE and velocity is NA, then motor voltage is Zero.
5. If position error is ZE and velocity is ZA, then motor voltage is Zero.
6. If position error is ZE and velocity is PA, then motor voltage is Zero.
7. If position error is PE and velocity is NA, then motor voltage is Positive.
8. If position error is PE and velocity is ZA, then motor voltage is Positive.
9. If position error is PE and velocity is PA, then motor voltage is Positive.



Controlling dynamics of a servomotor



Ex 6:

Example 4.4. Using your own intuition and definitions of the universe of discourse, plot fuzzy membership functions for "weight of people."

Solution. The universe of discourse is the weight of people. Let the weights be in "kg" – kilogram.

Let the linguistic variables are:

Very light	–	$w \leq 30$
Light	–	$30 < w \leq 45$
Average	–	$45 < w \leq 60$
Heavy	–	$60 < w \leq 75$
Very heavy	–	$w > 75$

Representing this using triangular membership function, as shown in Fig. 4.12.

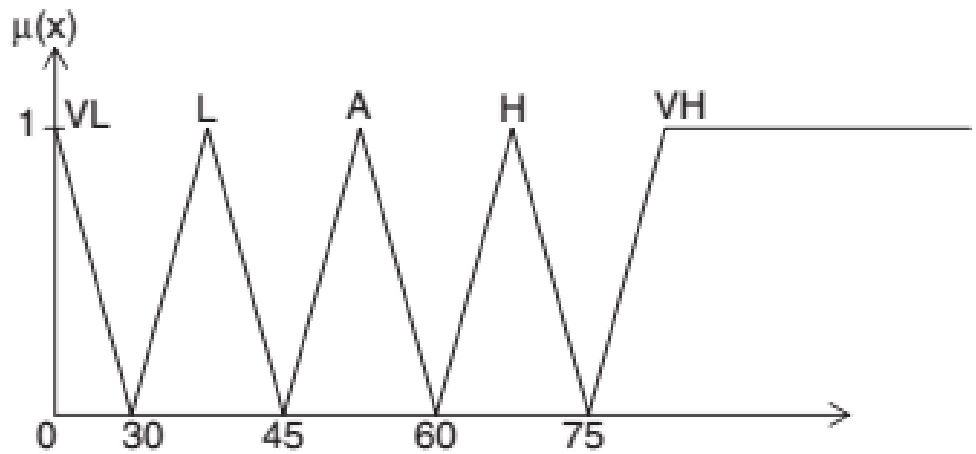


Fig. 4.12. Membership function of weight of people

Ex 7:

Example 4.5. Using your own intuition, plot the fuzzy membership function for the age of people.

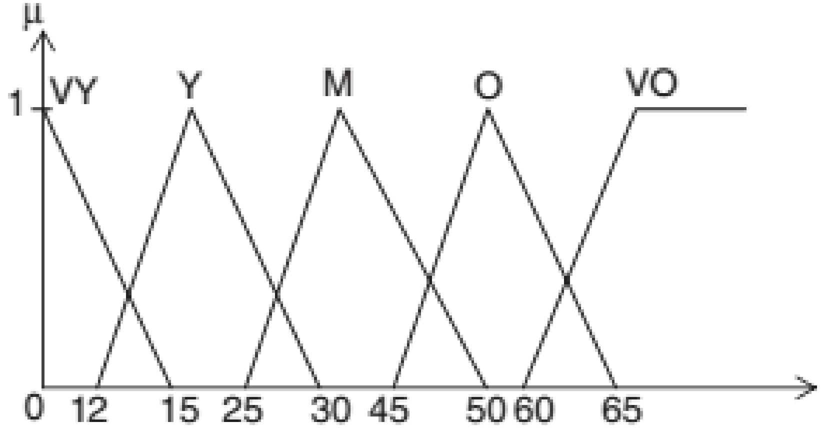


Fig. 4.13. Membership function for age of profile

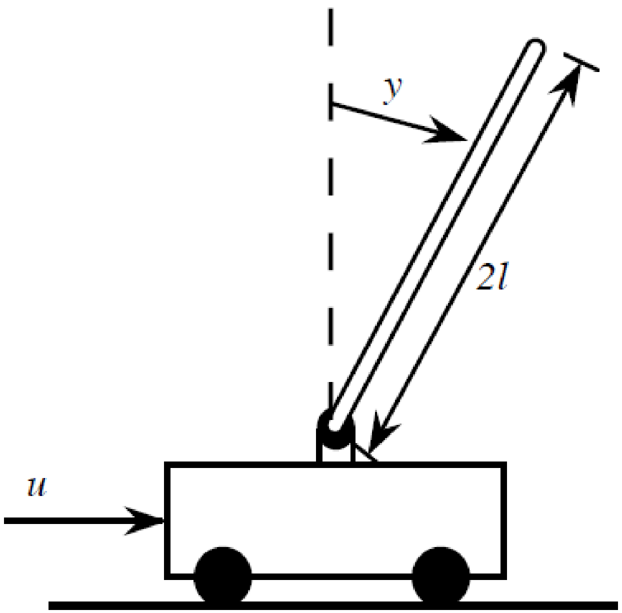
Solution.

The linguistic variables are defined as, let A denotes age in years.

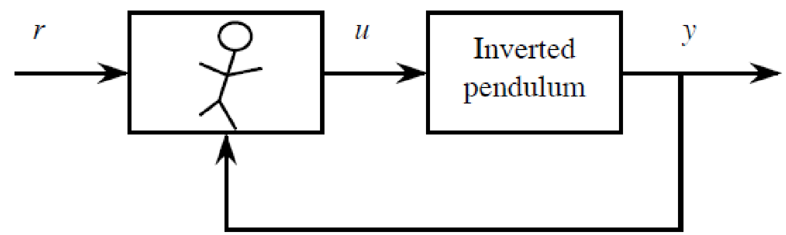
(1)	Very young (vy)	–	$A < 15$
(2)	Young (y)	–	$12 \leq A < 30$
(3)	Middle aged (m)	–	$25 \leq A < 50$
(4)	Old (o)	–	$45 \leq A < 65$
(5)	Very old (vo)	–	$60 < A$

This is represented using triangular membership, as shown in Fig. 4.13.

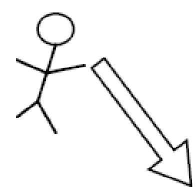
Ex 8: Inverted pendulum on a cart

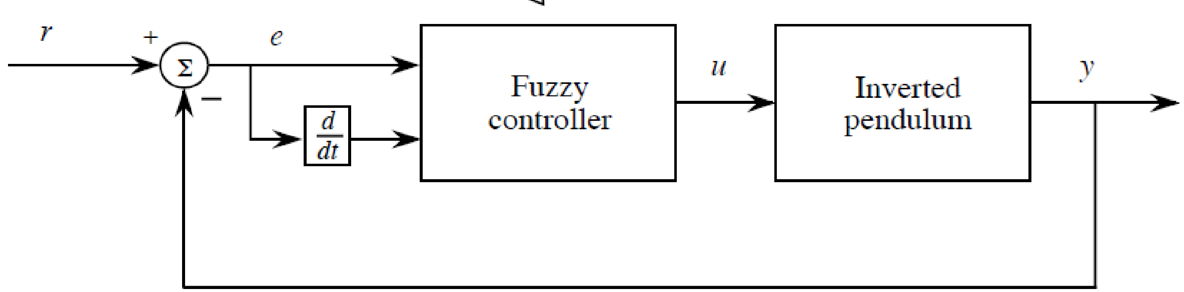


Suppose that for the inverted pendulum, the expert says that she or he will use *e(t) = r(t) − y(t)* and *d/dt(e(t))*



Human controlling an inverted pendulum on a cart.





Fuzzy controller for an inverted pendulum on a cart

Linguistic Descriptions

- There will be "linguistic variables" that describe each of the time varying fuzzy controller inputs and outputs.
- For the inverted pendulum,

  * "error" describes *e(t)*

  * "change-in-error" describes *d/dt(e(t))*

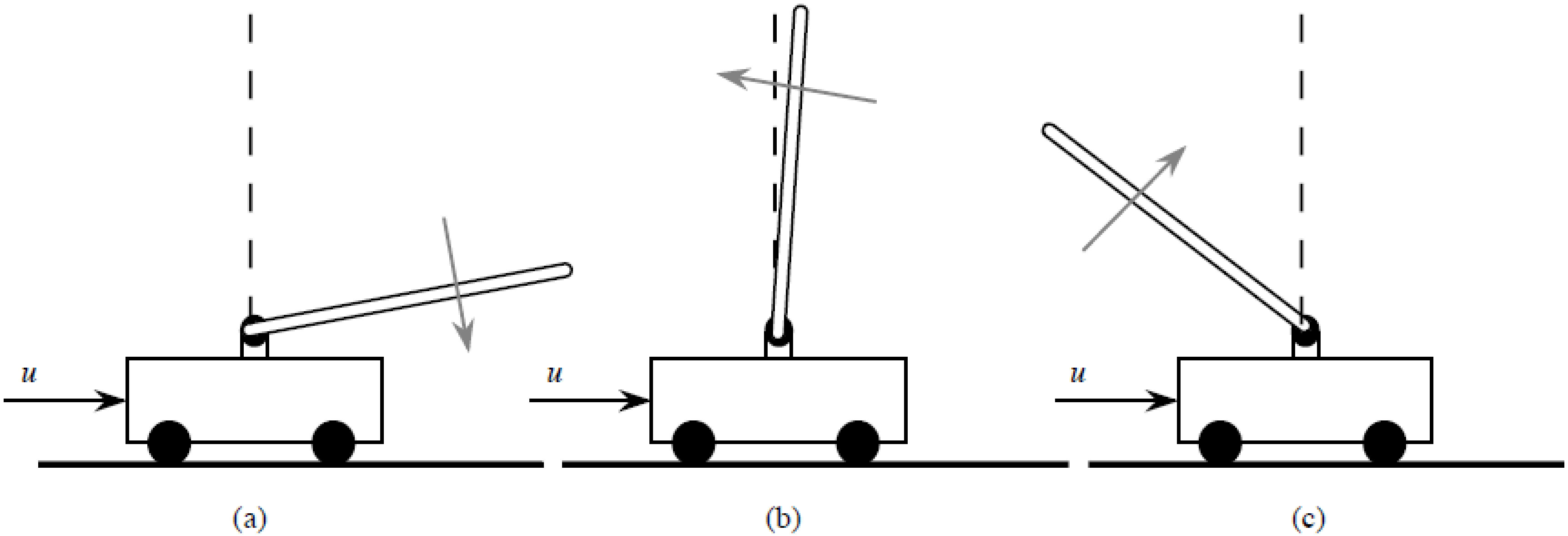
  * "force" describes *u(t)*

Linguistic Descriptions

- Suppose for the pendulum example that "error," "change-in-error," and "force" take on the following values:

    * "neglarge"
    * "negsmall"
    * "zero"
    * "possmall"
    * "poslarge"

Rules



Inverted pendulum in various positions.

If error is neglarge and change-in-error is neglarge Then force is poslarge

If error is zero and change-in-error is possmall Then force is negsmall

If error is poslarge and change-in-error is negsmall Then force is negsmall

Linguistic Descriptions

- For an even shorter description we could use integers:

  “−2” to represent “neglarge”

  “−1” to represent “negsmall”

  “0” to represent “zero”

  “1” to represent “possmall”

  “2” to represent “poslarge”

Rule-Bases

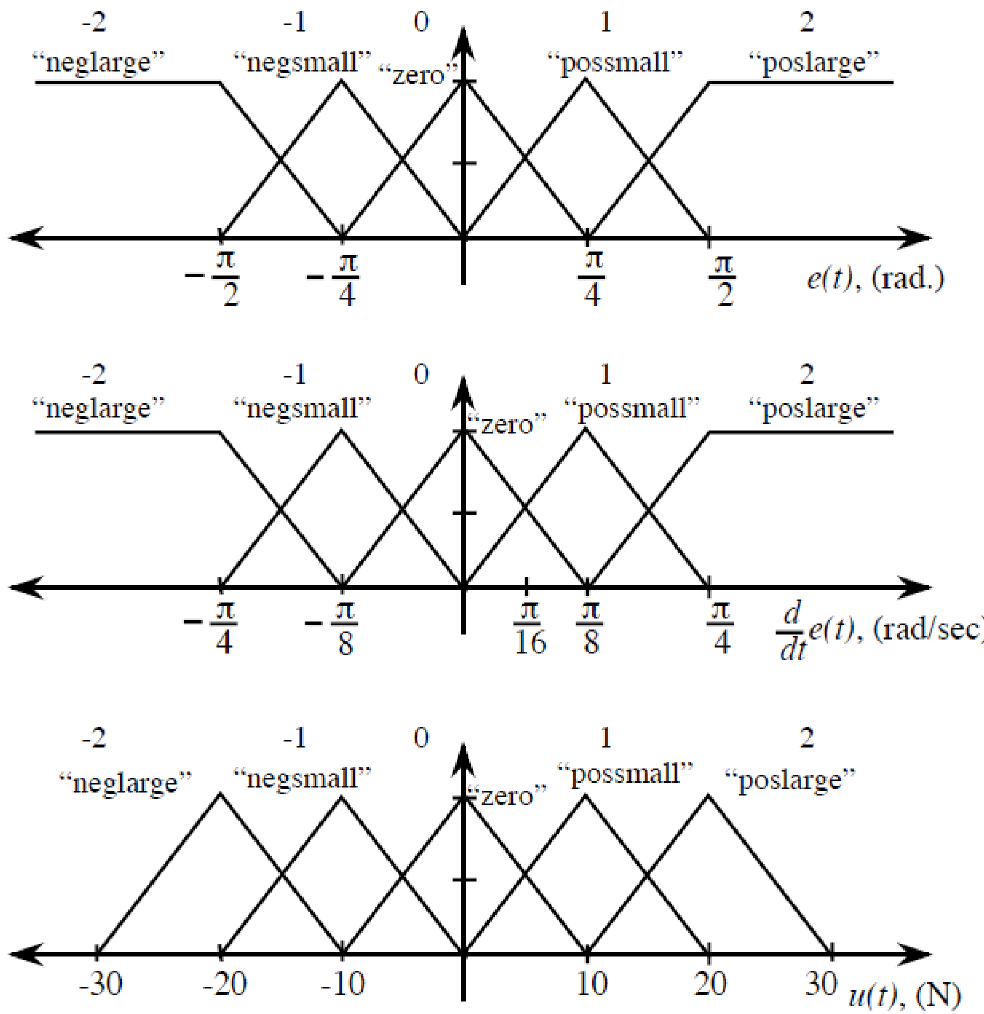
- For the pendulum problem, with two inputs and five linguistic values for each of these, there are at most $5^2 = 25$ possible rules

"force" u		"change-in-error" $\dot{e}$				
		-2	-1	0	1	2
"error" e	-2	2	2	2	1	0
	-1	2	2	1	0	-1
	0	2	1	0	-1	-2
	1	1	0	-1	-2	-2
	2	0	-1	-2	-2	-2

Rule Table for the Inverted Pendulum

if (input 1 is membership function 1) and/or (input 2 is membership *function 2)* ***and/or. . . then (outputn is output membership functionn).***

Membership Functions



Mamdani’s Fuzzy Inference Method