UNIVERSITY OF WATERLOO

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ECE 457B - Assignment 3

Prepared by:

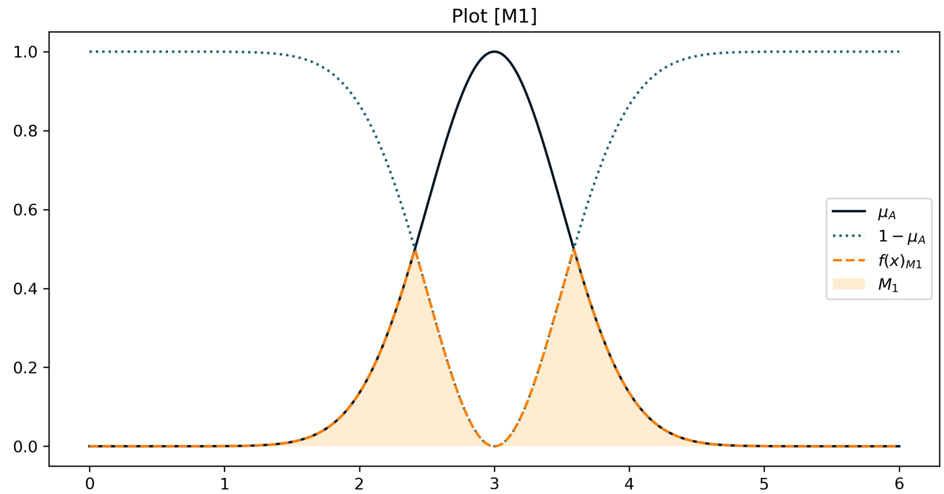
Jianxiang (Jack) Xu [20658861]

12 April 2021

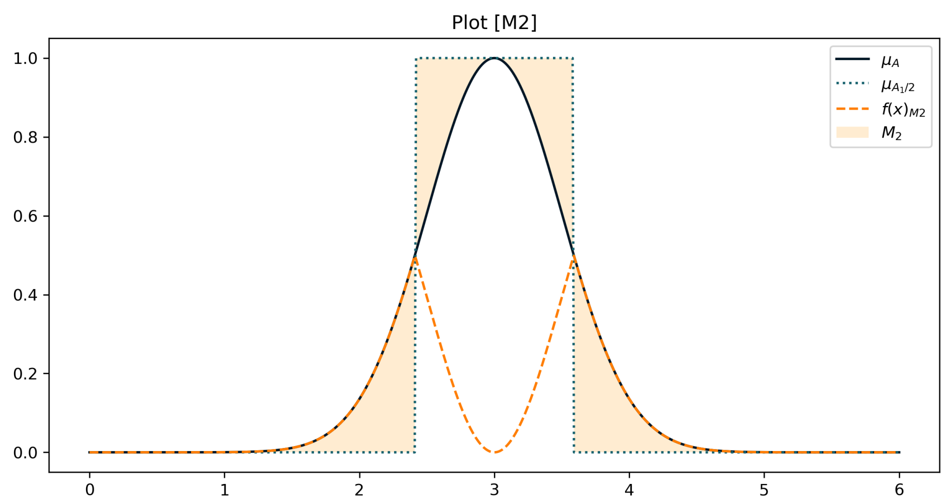
Table of Contents

1 Problem 1 (20 marks): Three Measures of Fuzziness [See Implementation in Code 1] 2
1.1 (i) Relationship between M_1, M_2, and M_3 3
1.2 (ii) Comments: 4
2 Problem 2 (20 marks): Higher Dimension Projection 5
2.1 (a) Total number of Projections: 5
2.2 (b) Numerical Examples: 6
3 Problem 3 (20 marks): Fuzzy Logic 7
3.1 (a) Comments on linguistic representations 7
3.2 (b) Discrete Example: [See Implementation in Code 2] 7
4 Problem 4 (5 marks): 9
5 Problem 5 (10 marks): [See implementation in Code 3] 11
6 Problem 6 (20 marks): [See implementation in Code 4] 14
6.1 (a) Four rules in membership diagram: 14
6.2 (b) Process Measurements of $ANG = 5^\circ$ and $VEL = 15^\circ/s$: 15
Appendix A Problem 1 17
Appendix B Problem 3 18
Appendix C Problem 5 19
Appendix D Problem 6 20

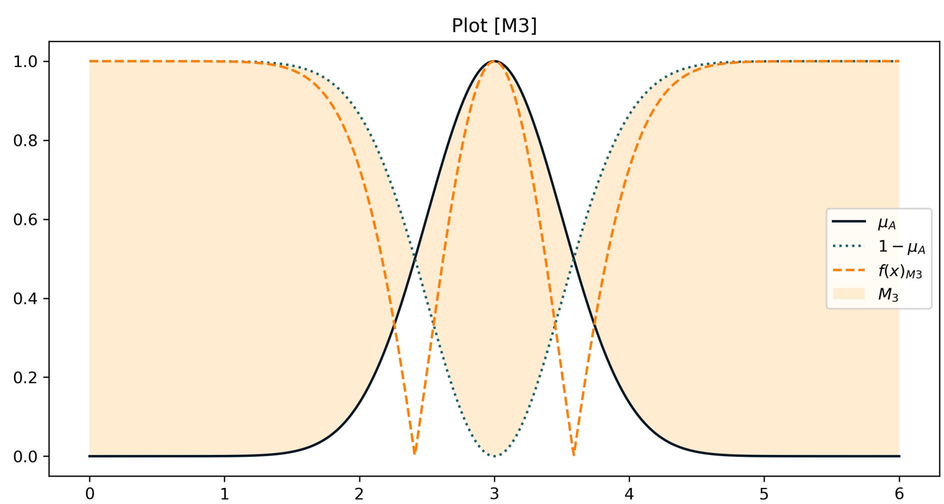
1 Problem 1 (20 marks): Three Measures of Fuzziness [See Implementation in Code 1]



(a) M_1: Closeness to grade 0.5



(b) M_2: Distance from 0.5-cut



(c) M_3: Inverse of distance from the complement

Figure 1-1. Three Measures of Fuzziness

1.1 (i) Relationship between M_1, M_2, and M_3

The idea is to find the relationship of the function it (M_i) integrates from, and then, we can apply the integration to find the final relationship.

Let's define:

$$M_1 = \int_S f_{M_1}(x)dx, \quad M_2 = \int_S f_{M_2}(x)dx, \quad M_3 = \int_S f_{M_3}(x)dx \tag{1}$$

Hence, we may define following functions ($f_{M_1}(x)$, $f_{M_2}(x)$, and $f_{M_3}(x)$):

$$f_{M_1}(x) = \begin{cases} \mu_A(x) & \text{, if } \mu_A(x) \leq 0.5 \\ 1 - \mu_A(x) & \text{, otherwise} \end{cases} \tag{2}$$

$$f_{M_2}(x) = |\mu_A(x) - \mu_{A_{1/2}}(x)| \tag{3}$$

$$f_{M_2}(x) = |\mu_A(x) - \mu_{\bar{A}}(x)| \tag{4}$$

$$\tag{5}$$

Firstly, let's derive the relationship between $f_{M_1}(x)$ and $f_{M_2}(x)$:

$$\begin{aligned} f_{M_2}(x) &= |\mu_A(x) - \mu_{A_{1/2}}(x)| && (6)\\ &= \begin{cases} \mu_A(x) - 1 & \text{, if } \mu_A(x) > \alpha = 0.5 \\ \mu_A(x) & \text{, otherwise} \end{cases} && (7)\\ &= \begin{cases} \mu_A(x) & \text{, if } \mu_A(x) \leq 0.5 \\ 1 - \mu_A(x) & \text{, otherwise} \end{cases} && (8)\\ &\equiv f_{M_1}(x) && (9) \end{aligned}$$

Similarly, let's derive the relationship between $f_{M_1}(x)$ and $f_{M_3}(x)$:

$$\begin{aligned} \because \quad f_{M_3}(x) &= |\mu_A(x) - \mu_{\bar{A}}(x)| && (10)\\ &= |2\mu_A(x) - 1| && (11)\\ &= \begin{cases} 2\mu_A(x) - 1 & \text{, if } (2\mu_A(x) - 1) > 0 \\ 1 - 2\mu_A(x) & \text{, otherwise} \end{cases} && (12)\\ &= \begin{cases} 2\mu_A(x) - 1 & \text{, if } \mu_A(x) > 0.5 \\ 1 - 2\mu_A(x) & \text{, otherwise} \end{cases} && (13)\\ 1 - f_{M_3}(x) &= \begin{cases} 2 - 2\mu_A(x) & \text{, if } \mu_A(x) > 0.5 \\ 2\mu_A(x) & \text{, otherwise} \end{cases} && (14)\\ \frac{1}{2}(1 - f_{M_3}(x)) &= \begin{cases} 1 - 1\mu_A(x) & \text{, if } \mu_A(x) > 0.5 \\ 1\mu_A(x) & \text{, otherwise} \end{cases} && (15)\\ \frac{1}{2}(1 - f_{M_3}(x)) &= \begin{cases} \mu_A(x) & \text{, if } \mu_A(x) \leq 0.5 \\ 1 - \mu_A(x) & \text{, otherwise} \end{cases} && (16)\\ &\equiv f_{M_1}(x) && (17)\\ \therefore \quad f_{M_1}(x) &= \frac{1}{2}(1 - f_{M_3}(x)) && (18) \end{aligned}$$

From Equation (9) and Equation (18) we can conclude the following relationships:

$$f_{M_1}(x) = f_{M_2}(x) = \frac{1}{2}(1 - f_{M_3}(x)) \tag{19}$$

Finally, based on Equation (1), we can derive the final integral relationship between M_1, M_2, and M_3:

$$M_1 = M_2 = \frac{1}{2}(1 - M_3) \tag{20}$$

1.2 (ii) Comments:

- If the membership grade of an element is close to unity, the element is almost definitely a member of set. In contrast, if the membership grade is close to zero, the element is nearly outside the set. Hence, a membership grade of $\mu_A(x) = 0.5$ would be the most fuzzy point of an element x in a set A, since it has 50-50 chances without preference. As a result, to measure the degree of fuzziness of a membership function, it is simple to integrate or compute the area between the function and $\mu_A(x) = 0.5$.
- M_1 is the closeness of its membership function μ_A to the most fuzzy grade (0.5) as shown in Figure 1-1a, hence, the larger the value, the more fuzzy it is.
- M_2 is the distance from 1/2 cut line, which measures the distance of membership function from 1/2 cut as seen in Figure 1-1b. Similarly, the larger it is, the fuzzier it becomes.
- M_3 is the inverse distance from its complement $\mu_{\bar{A}}$, which computes the distance between the membership function $\mu_A(x)$ and its complement $\mu_{\bar{A}}(x)$ as highlighted in Figure 1-1c. Hence, in contrast, the larger it is, the less fuzzy it is.

2 Problem 2 (20 marks): Higher Dimension Projection

2.1 (a) Total number of Projections:

- Projecting from nD to a lower dimension $(n-1)$D requires n times of projections (e.g.: $3D \rightarrow 2D:\ k=3$).
- Projecting from nD to a lower dimension $(n-2)$D requires $n(n-1)$ times of projections (e.g.: $3D \rightarrow 1D:\ k=6$).
- By induction, we can derive number of projections from nD to $(n-r)$D:

$$k = n(n-1)\dots(n-(r-1)) = \prod_{j=0}^{r-1}(n-j) \tag{21}$$

- But, in fact, there are repetition due to different orders, hence, number of unique projects:

$$\begin{aligned} k_{unique}(r=n-m|n\rightarrow m) &= \frac{n}{1} \times \frac{n-1}{2} \times \frac{n-2}{3} \times \frac{n-(r-1)}{r!} && (22)\\ &= \frac{n\times(n-1)\times\dots(n-(r-2))\times\dots(n-(r-1))}{1\times 2\times\dots\times(r-1)\times r} && (23)\\ &= \frac{\prod_{j=0}^{r-1}(n-j)}{r!} && (24)\\ &\equiv \prod_{j=0}^{r-1}\frac{(n-j)}{(1+j)} && (25)\end{aligned}$$

(For example: $3D \rightarrow 1D: k = \frac{n(n-1)}{2!} = 3$, with $r=2, n=3$, specifically: unique projection to set $\{X,Y,Z\}$)
- As a result, the total number of distinct projection is:

$$\begin{aligned} K_{total,unique}(n) &= \sum_{m=1}^{n-1} k_{unique}(r=n-m|n\rightarrow m) && (26)\\ &= \sum_{r=1}^{n-1}\prod_{j=0}^{r-1}\frac{(n-j)}{(1+j)} && (27)\\ &\equiv n + \frac{n(n-1)}{2!} + \frac{n(n-1)(n-2)}{3!} + \dots + \frac{n(n-1)\times\dots\times 2}{(n-1)!} && (28)\end{aligned}$$

(For example: $K(3) = k(3\rightarrow 2) + k(3\rightarrow 1) = 3 + \frac{3(3-1)}{2!} = 3+3 = 6$ unique projections)

2.2 (b) Numerical Examples:

For simplicity, we will use maximum element-wise operation:

$$\underset{x_i,y_j,z_k\to x_i,y_j}{\text{Proj}} R(x_i,y_j,z_k) = \max_{z_k} R(x_i,y_j,z_k) = \begin{bmatrix} 0.6 & 0.5 & 0.3 \\ 0.4 & 1.0 & 0.6 \\ 0.2 & 0.6 & 0.8 \end{bmatrix} \tag{29}$$

$$\underset{x_i,y_j,z_k\to y_j,z_k}{\text{Proj}} R(x_i,y_j,z_k) = \max_{x_i} R(x_i,y_j,z_k) = \begin{bmatrix} 0.5 & 0.8 & 0.6 \\ 0.6 & 1.0 & 0.8 \\ 0.4 & 0.7 & 0.5 \end{bmatrix} \tag{30}$$

$$\underset{x_i,y_j,z_k\to z_k,x_i}{\text{Proj}} R(x_i,y_j,z_k) = \max_{y_j} R(x_i,y_j,z_k) = \begin{bmatrix} 0.5 & 0.6 & 0.4 \\ 0.8 & 1.0 & 0.7 \\ 0.6 & 0.8 & 0.5 \end{bmatrix} \tag{31}$$

$$\underset{x_i,y_j,z_k\to x_i}{\text{Proj}} R(x_i,y_j,z_k) = \max_{x_i} \underset{x_i,y_j,z_k\to x_i,y_j}{\text{Proj}} R(x_i,y_j,z_k) = \begin{bmatrix} 0.6 \\ 1.0 \\ 0.8 \end{bmatrix} \tag{32}$$

$$\underset{x_i,y_j,z_k\to y_j}{\text{Proj}} R(x_i,y_j,z_k) = \max_{y_j} \underset{x_i,y_j,z_k\to x_i,y_j}{\text{Proj}} R(x_i,y_j,z_k) = \begin{bmatrix} 0.6 & 1.0 & 0.8 \end{bmatrix} \tag{33}$$

$$\underset{x_i,y_j,z_k\to z_k}{\text{Proj}} R(x_i,y_j,z_k) = \max_{z_k} \underset{x_i,y_j,z_k\to y_j,z_k}{\text{Proj}} R(x_i,y_j,z_k) = \begin{bmatrix} 0.8 \\ 1.0 \\ 0.7 \end{bmatrix} \tag{34}$$

Note, there are totally 6 unique projections, 3 from $3D \to 2D$ and 3 from $3D \to 1D$. There are two sets of identical projections from $3D \to 2D$, as listed below:

$$\max_{x_i} \underset{x_i,y_j,z_k\to x_i,y_j}{\text{Proj}} \equiv \max_{x_i} \underset{x_i,y_j,z_k\to z_k,x_i}{\text{Proj}} \tag{35}$$

$$\max_{y_j} \underset{x_i,y_j,z_k\to x_i,y_j}{\text{Proj}} \equiv \max_{y_j} \underset{x_i,y_j,z_k\to y_j,z_k}{\text{Proj}} \tag{36}$$

$$\max_{z_k} \underset{x_i,y_j,z_k\to z_k,x_i}{\text{Proj}} \equiv \max_{z_k} \underset{x_i,y_j,z_k\to y_j,z_k}{\text{Proj}} \tag{37}$$

Hence, total $3+3=6$ unique projections.

3 Problem 3 (20 marks): Fuzzy Logic

3.1 (a) Comments on linguistic representations

- The use of translational operator $\mu_F(v - v_0)$, with $v_0 > 0$ indicates shifting the centre axis of membership function to right, resulting a much higher standard as the new standard. Consequently, this incorporates "very" to the state, resulting "very fast speed" state from "fast speed".
- The square operation of $\mu_F^2(v)$ behaves like a stretching on the membership function along the μ_F axis, resulting an amplification on the statement. Hence, the statement of "presumably fast speed" is contradictory to the operator. The operator is stating about "definitely fast speed" instead. To correctly represent "presumably fast speed" with power of 2, the proper operator would be the 2nd order dilation operator: $\mu_F^{1/2}(v)$

3.2 (b) Discrete Example: [See Implementation in Code 2]

Per discussion, the formal representation would be restated here:

- "very fast speed": $\mu_F(v)^{very} = \mu_F(v - v_0)$, with $v_0 > 0$
- "persumably fast speed": $\mu_F(v)^{persumably} = \mu_F^{1/2}(v)$

For the given discrete universe $V = \{0, 10, 20, \ldots, 200\}$ [rev/s] and $v_0 = 50$ [rev/s] with the Fuzzy Set F in Equation (38), we may display both membership functions in Figure 3-1 and Table 3-1 below.

$$F = \left\{ \frac{0.1}{10}, \frac{0.3}{20}, \frac{0.6}{30}, \frac{0.8}{40}, \frac{1.0}{50}, \frac{0.7}{60}, \frac{0.5}{70}, \frac{0.3}{80}, \frac{0.1}{90} \right\} \tag{38}$$

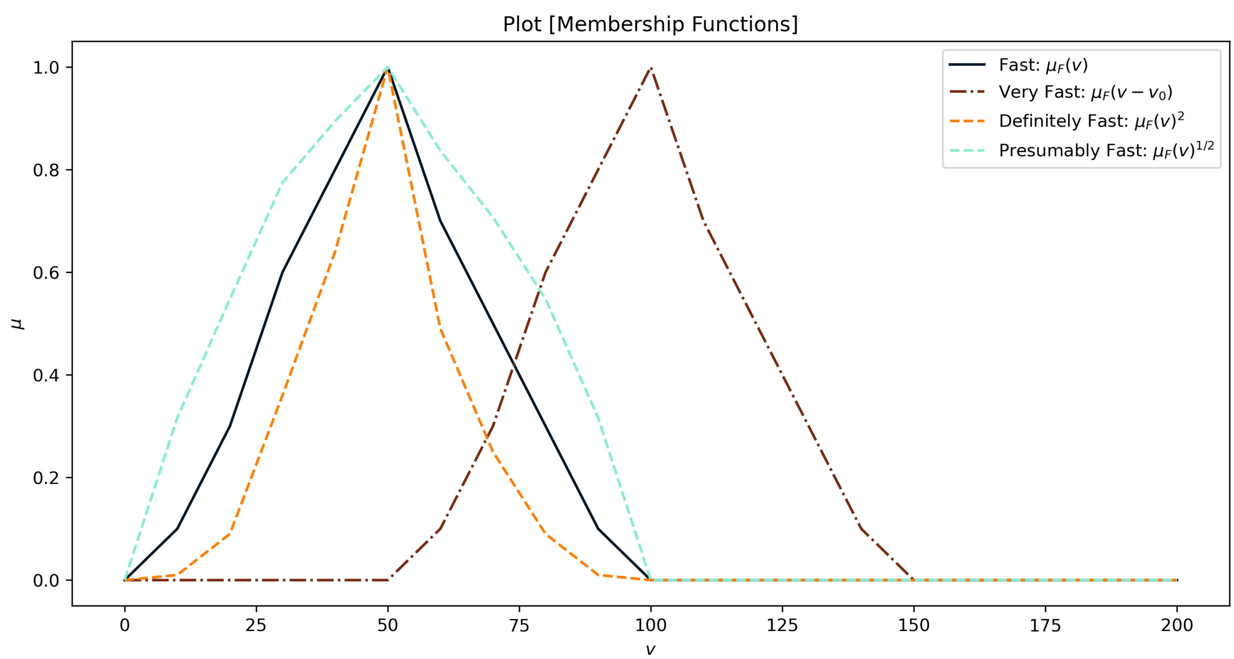


Figure 3-1. Different membership functions

Table 3-1. Membership Functions

V[rev/s]	Fast	Very Fast	Definitely Fast	Presumably Fast
0	0	0	0.0	0.0
10	0.1	0	0.01	0.32
20	0.3	0	0.09	0.55
30	0.6	0	0.36	0.77
40	0.8	0	0.64	0.89
50	1.0	0	1.0	1.0
60	0.7	0.1	0.49	0.84
70	0.5	0.3	0.25	0.71
80	0.3	0.6	0.09	0.55
90	0.1	0.8	0.01	0.32
100	0	1.0	0.0	0.0
110	0	0.7	0.0	0.0
120	0	0.5	0.0	0.0
130	0	0.3	0.0	0.0
140	0	0.1	0.0	0.0
150	0	0	0.0	0.0
160	0	0	0.0	0.0
170	0	0	0.0	0.0
180	0	0	0.0	0.0
190	0	0	0.0	0.0
200	0	0	0.0	0.0

4 Problem 4 (5 marks):

To prove that $\mathbf{T} = \max\{0, x+y-1\}$ is a **T**-norm operator, we need to prove it satisfies the **T**-norm properties:

I Non-decreasing in each argument. i.e., if $a \le b$ and $c \le d$ then $a\mathbf{T}c \le b\mathbf{T}d$.
II Commutativity: i.e., $a\mathbf{T}b = b\mathbf{T}a$
III Boundary Conditions: i.e., $a\mathbf{T}1 = a$ and $a\mathbf{T}0 = 0 \quad \Rightarrow$ take minimum
IV Associativity: i.e., $(a\mathbf{T}b)\mathbf{T}c = a\mathbf{T}(b\mathbf{T}c)$

Proof 4.1: Property I: Non-decreasing

$$\text{Let } a \le b,\ c \le d \tag{39}$$
$$a\mathbf{T}c = \max\{0, a+c-1\} \tag{40}$$
$$b\mathbf{T}d = \max\{0, b+d-1\} \tag{41}$$
$$\therefore \quad a\mathbf{T}c,\ b\mathbf{T}d\ \ge 0 \tag{42}$$
$$\because \quad a \le b,\ c \le d \tag{43}$$
$$\therefore \quad (a+c-1) \le (b+d-1) \tag{44}$$
$$\therefore \quad 0 \le \max\{0, a+c-1\} \le \max\{0, b+d-1\} \tag{45}$$
$$\therefore \quad 0 \le a\mathbf{T}c \le\le b\mathbf{T}d \tag{46}$$
$$\therefore \quad \text{Satisfy Property I} : a\mathbf{T}c \le b\mathbf{T}d \quad , \text{if } a \le b \text{ and } c \le d \tag{47}$$

d **Q.E.D.**

Proof 4.2: Property II: Commutativity

$$\textbf{LHS:} \quad = a\mathbf{T}b = \max\{0, a+b-1\} \tag{48}$$
$$\textbf{RHS:} \quad = b\mathbf{T}a = \max\{0, b+a-1\} = \max\{0, a+b-1\} \tag{49}$$
$$\therefore \quad \textbf{LHS} \equiv \textbf{RHS} \tag{50}$$
$$\therefore \quad \text{Satisfy Property II} : a\mathbf{T}b = b\mathbf{T}a \tag{51}$$

Q.E.D.

Proof 4.3: Property III: Boundary Conditions

$$a\mathbf{T}1 = \max\{0, a+1-1\} = \max\{0, a\} = a,\ \forall a \in [0,1] \tag{52}$$
$$a\mathbf{T}0 = \max\{0, a+0-1\} = \max\{0, a-1\} = 0,\ \forall a \in [0,1] \tag{53}$$
$$\therefore \quad \text{Satisfy Property III} : a\mathbf{T}1 = a,\ a\mathbf{T}0 = 0 \tag{54}$$

Q.E.D.

Proof 4.4: Property IV: Associativity

$$
\begin{aligned}
\textbf{LHS:} \quad & = (a\mathbf{T}b)\mathbf{T}c = \max\{0, \max\{0, a+b-1\} + c - 1\} && (55)\\
& = \max\{0, \max\{c-1, a+b-1+c-1\}\} && (56)\\
\because \quad & c \in [0,1] \Rightarrow \ (c-1) \leq 0 && (57)\\
& = \begin{cases} 0 & \text{, if } (a+b+c-2) \leq 0 \\ (a+b+c-2) & \text{, otherwise} \end{cases} && (58)\\
\textbf{RHS:} \quad & = a\mathbf{T}(b\mathbf{T}c) = \max\{0, a + \max\{0, b+c-1\} - 1\} && (59)\\
& = \max\{0, \max\{a-1, a+b+c-1-1\}\} && (60)\\
\because \quad & a \in [0,1] \Rightarrow \ (a-1) \leq 0 && (61)\\
& = \begin{cases} 0 & \text{, if } (a+b+c-2) \leq 0 \\ (a+b+c-2) & \text{, otherwise} \end{cases} && (62)\\
\therefore \quad & \textbf{LHS} \equiv \textbf{RHS} && (63)\\
\therefore \quad & \text{Satisfy Property IV : } (a\mathbf{T}b)\mathbf{T}c = a\mathbf{T}(b\mathbf{T}c) && (64)
\end{aligned}
$$

Q.E.D.

Hence, as proved above in Proof 4.1, Proof 4.2, Proof 4.3, and Proof 4.4, it is confirmed that $\mathbf{T} = \max\{0, x+y-1\}$ is a **T**-norm operator.

Now, let's derive for the **T**-conorm (aka. **S**-norm) from DeMorgan's Law:

$$
\begin{aligned}
a\mathbf{S}b &= \overline{\bar{a}\mathbf{T}\bar{b}} && (65)\\
&= 1 - [\max\{0, (1-a) + (1-b) - 1\}] && (66)\\
&= 1 - [\max\{0, 1-a-b\}] && (67)\\
&= 1 + [\min\{0, a+b-1\}] && (68)\\
&= \min\{1, a+b\} && (69)\\
\therefore \quad \mathbf{S} &= \min\{1, x+y\} && (70)
\end{aligned}
$$

5 Problem 5 (10 marks): [See implementation in Code 3]

As implemented in Code 3, we have plotted the membership function $\mu_A(x) = \exp\{-\lambda|x-a|^n\}$ below, by varying its parameters.

Varying a: As we may see from Figure 5-1, the a changes the center of the membership function but it does not affect its shape. As a increases, the membership function translates towards to the right side, this leads towards an amplification of "very" from the fuzzy modifier. Conversely, as a decreases, the membership function translates towards the left side, attenuate the fuzzy state, and the state become "less". The fuzziness remains unchanged $\forall a$.

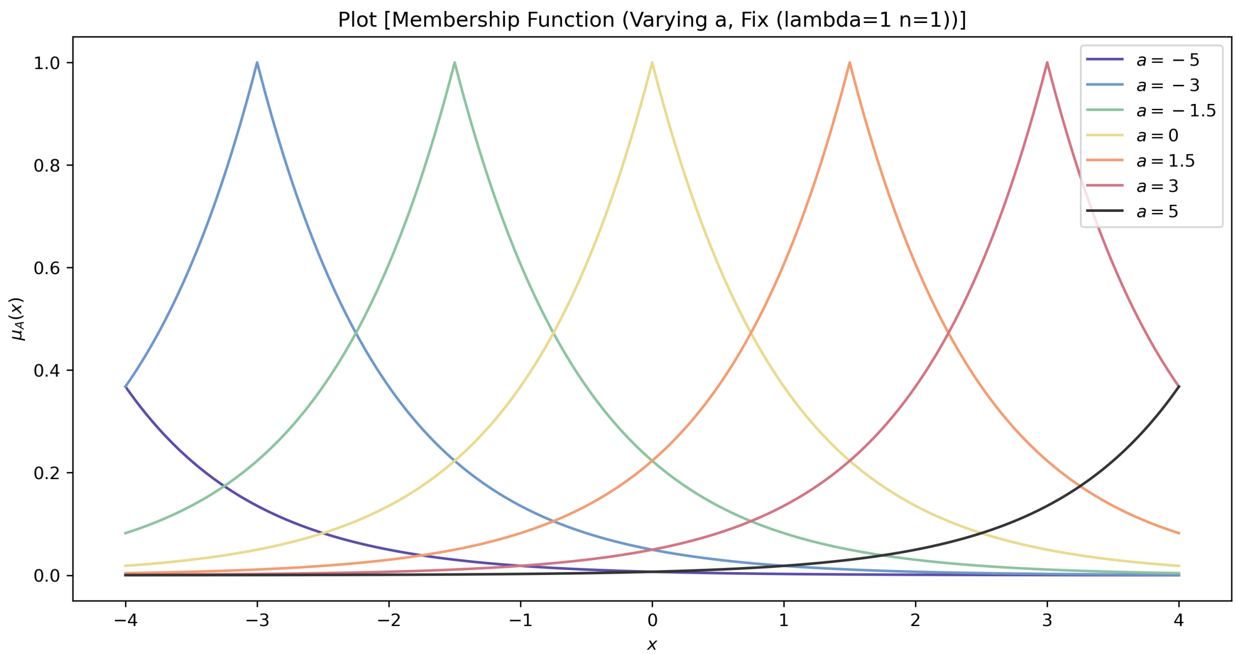


Figure 5-1. Varying a

Varying λ: As we may see from Figure 5-2, the λ changes the shape of the membership function. An increasing λ leads to a narrower graph, resulting a contraction effect to the fuzzy state. Hence, an increasing λ would make the fuzzy state more firm (or less fuzziness). In contrast, when reducing the λ towards 0, it makes the fuzzy state more "presumable" and uncertain. At the $\lambda \leq 0$, the whole fuzzy state is completely useless, since $\mu_A(x) = 1 \; \forall x \in \mathbb{R}$.

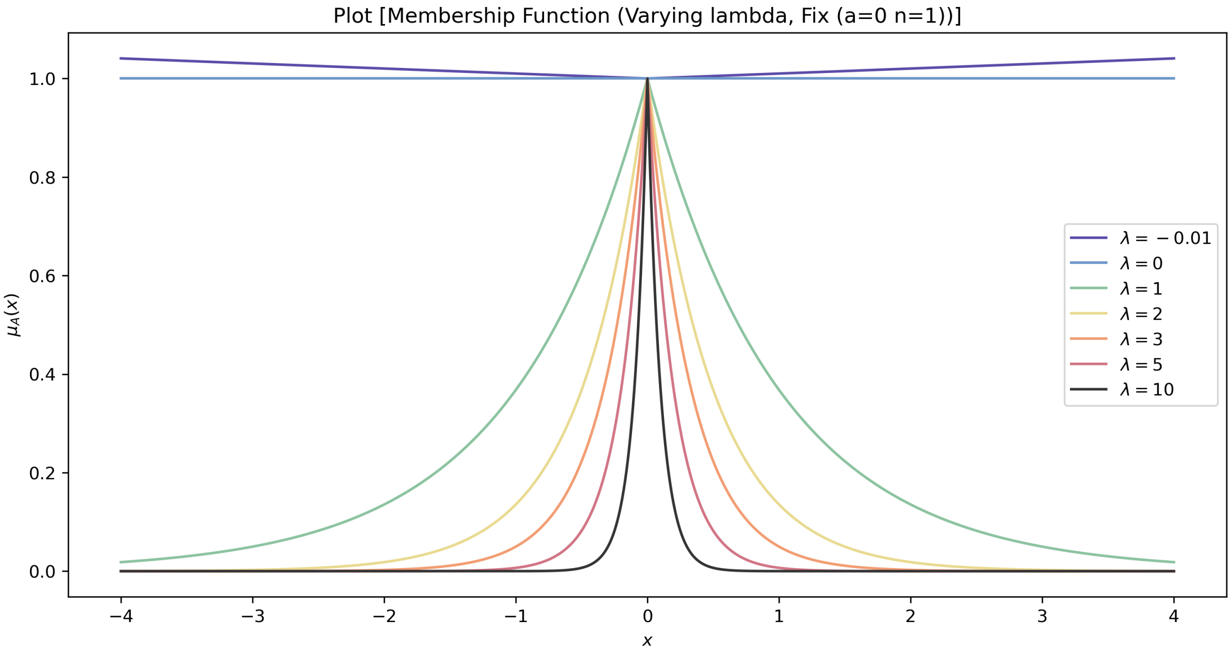


Figure 5-2. Varying λ

Varying n As we may see from Figure 6-3, the n changes the shape of the membership function. When $n > 0$, an increasing n results a sharper pulse curve, and it tends toward to form a shape of a unit pulse as $n \to \infty$. As a result, an increasing n decreases the fuzziness of the membership function, resulting a much more firm state. In contrast, as the n decreases, the state becomes fuzzier and more uncertain (”somehow”). However, when $n < 0$, the membership function flips, resulting a flipped effects: the smaller $n < 0$ is, it becomes less fuzzy, besides the fact that the entire state is flipped.

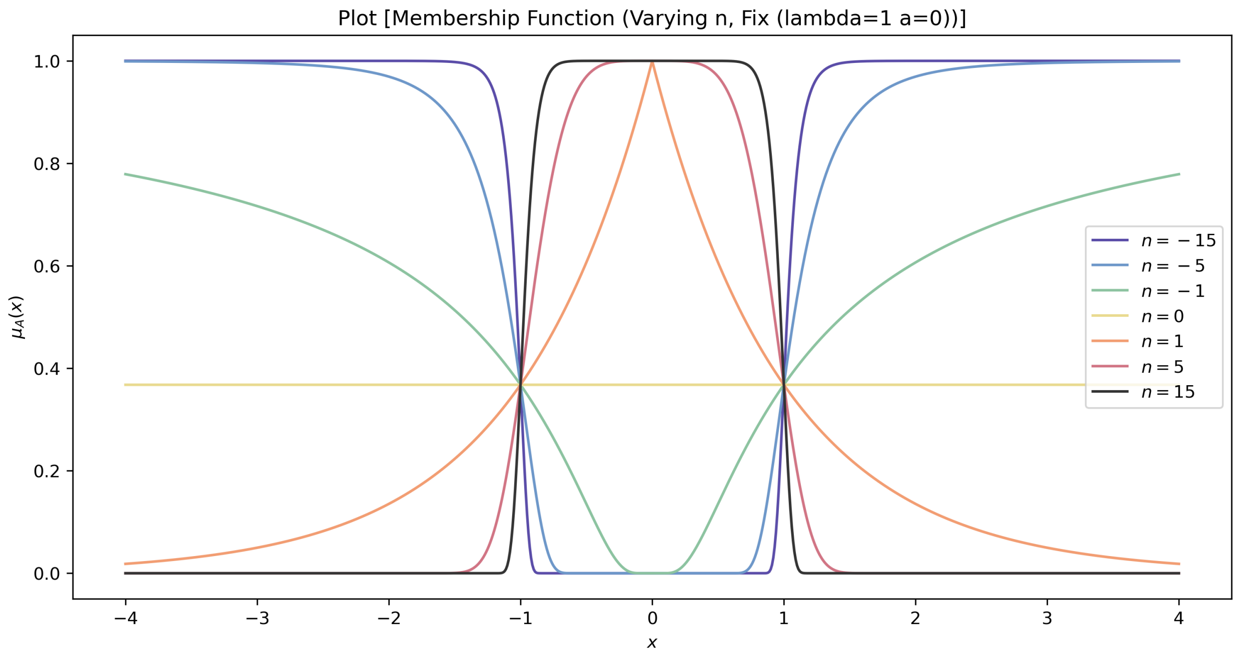


Figure 5-3. Varying n

6 Problem 6 (20 marks): [See implementation in Code 4]

6.1 (a) Four rules in membership diagram:

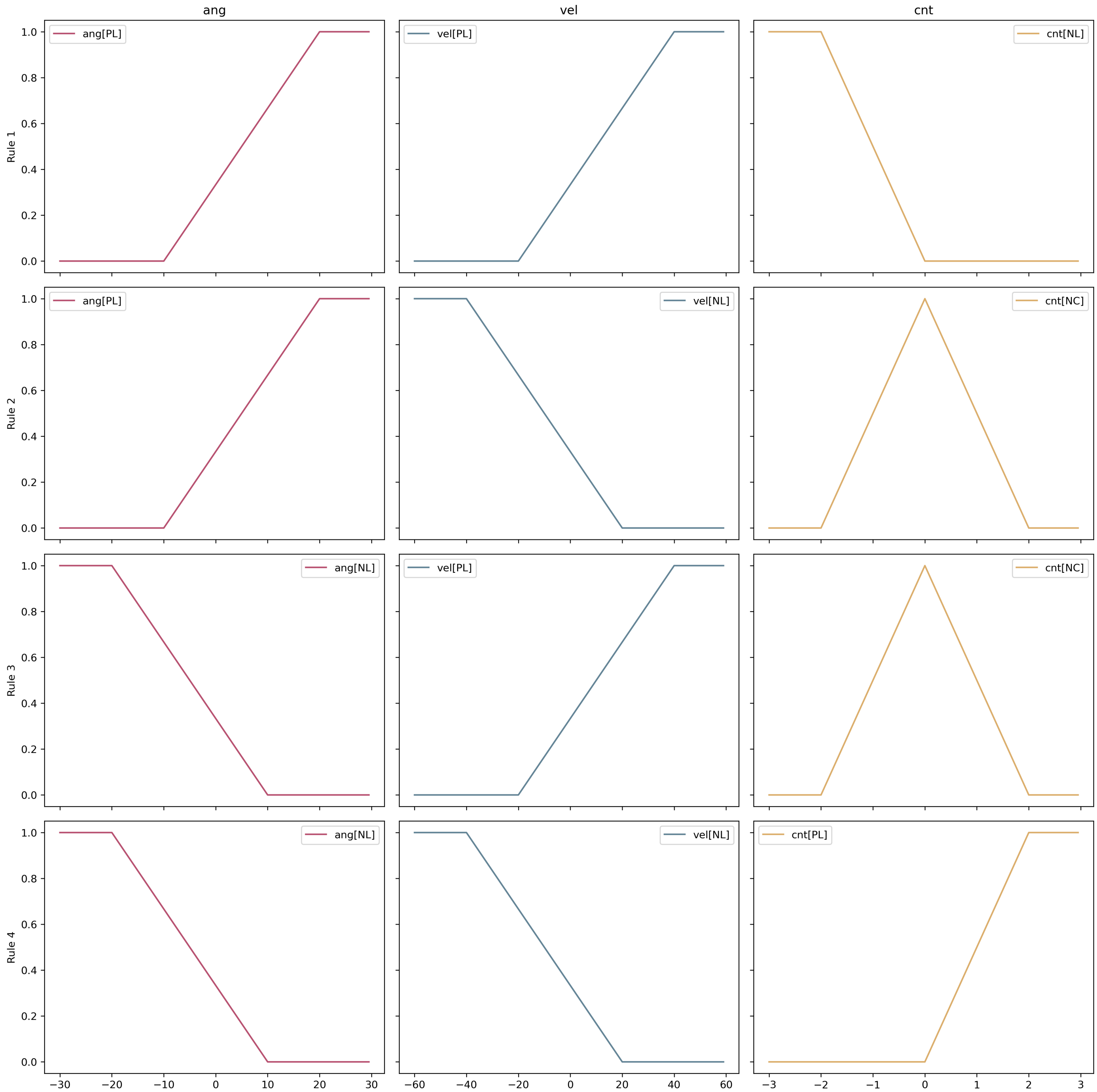


Figure 6-1. Sketch of the four rules in a membership diagram for the purpose of making control inferences using individual rule-based inference

6.2 (b) Process Measurements of $ANG = 5^\circ$ and $VEL = 15^\circ/s$:

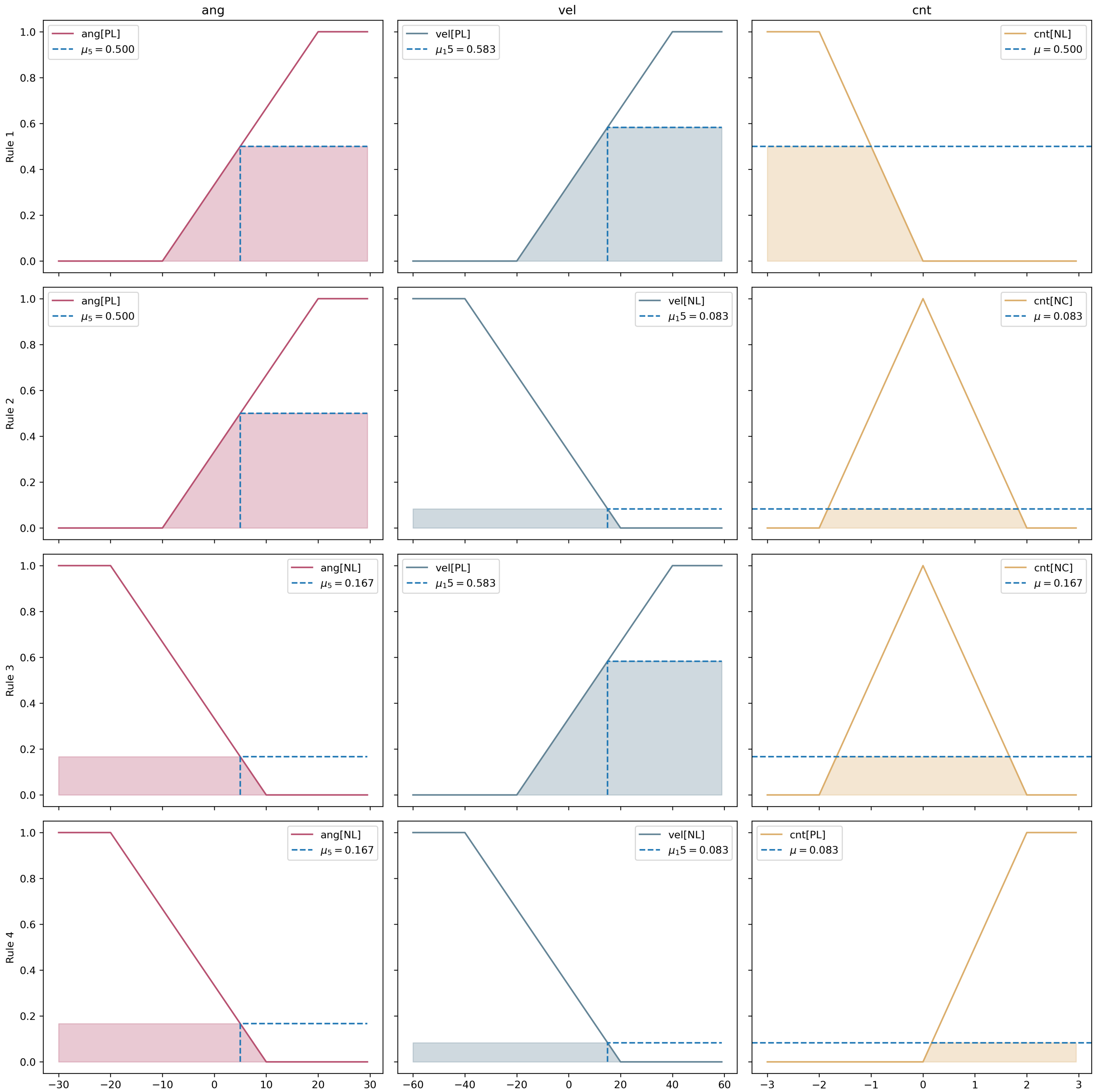


Figure 6-2. Sketch with corresponding control inference

As a result, the final control value was determined to be $-1.02A$ from the final control inference plot shown below:

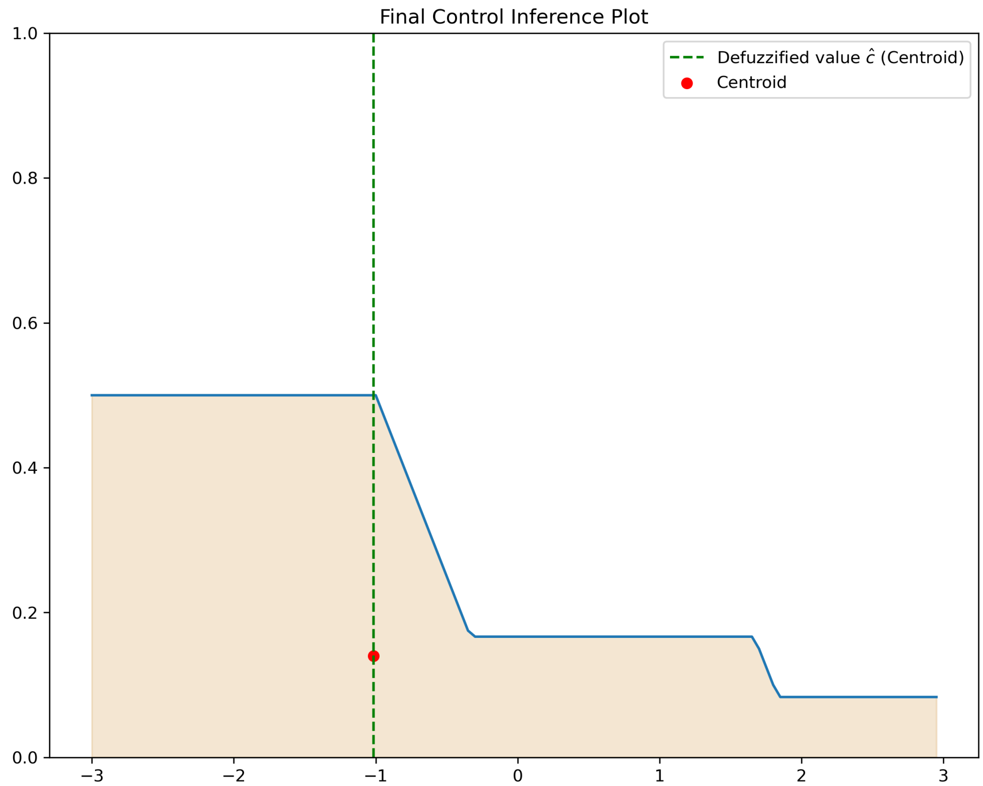


Figure 6-3. Sketch with corresponding control inference

Appendix A Problem 1

```python
# %% Lib:
import numpy as np
import matplotlib.pyplot as plt

# custom lib:
import jx_lib

#%% P1
OUT_DIR_P1="output/P1"
jx_lib.create_all_folders(DIR=OUT_DIR_P1)


# %% ---- USER PARAMS:
S = [0, 6]
STEP = 0.01
PARAMS = {
    "lambda":   2,
    "n":        2,
    "a":        3,
}
def f_mu_A(x, params):
    return np.exp(- params["lambda"] * ((x - params["a"]) ** (params["n"])))

def A_alpha_cut(mu_A, alpha):
    return mu_A >= alpha

# Pre-Compute:
s = np.linspace(S[0], S[1], int((S[1]-S[0])/STEP + 1))
mu_A = f_mu_A(x=s, params=PARAMS)


# %%
def output_plot(
    data_dict,
    highlight_dict = None,
    Ylabel  = "",
    Xlabel  = "",
    figsize = (10,5),
    OUT_DIR = "",
    tag     = ""
):
    COLOR_TABLE = ["#001524","#15616d","#ff7d00","#ffecd1","#78290f"]
    fig = plt.figure(figsize=figsize)
    ax = plt.gca()
    i = 0
    for name_, data_ in data_dict.items():
        linestyle="-"
        if "linestyle" in data_:
            linestyle = data_["linestyle"]
        plt.plot(data_["x"], data_["y"],
            label=name_, color=COLOR_TABLE[i], linestyle=linestyle)
        i += 1;
    # highlight content:
    if highlight_dict is not None:
        highlight_x = highlight_dict["highlight_x"]
        highlight_lb = highlight_dict["highlight_lb"]
        highlight_ub = highlight_dict["highlight_ub"]
        label = highlight_dict["label"]
        ax.fill_between(
            highlight_x, highlight_lb, highlight_ub,
            label=label, facecolor=COLOR_TABLE[i])
    plt.ylabel(Ylabel)
    plt.xlabel(Xlabel)
    plt.legend()
    plt.title("Plot [{}]".format(tag))
    fig.savefig("{}/plot_{}.png".format(OUT_DIR, tag), bbox_inches = 'tight', dpi=300)
    plt.close(fig)
    return fig
```

```
# %% ---- Draw Membership Functions (a):
fx_1 = lambda mu_A_: np.array([mu_a_ if mu_a_ <= 0.5 else (1-mu_a_) for mu_a_ in mu_A_])

# Plot M1:
output_plot(
    data_dict= {
        "$\mu_A$" : {"x": s, "y": mu_A},
        "$1 - \mu_A$" : {"x": s, "y": (1 - mu_A), "linestyle":':'},
        "$f(x)_{M1}$": {"x": s, "y": fx_1(mu_A_=mu_A), "linestyle":'--'},
    },
    highlight_dict = {
        "highlight_x"   : s,
        "highlight_lb"  : 0,
        "highlight_ub"  : fx_1(mu_A_=mu_A),
        "label" : "$M_1$"
    },
    OUT_DIR = OUT_DIR_P1,
    tag = "M1",
)

# %% ---- Draw Membership Functions (b):
fx_2 = lambda mu_A_: np.abs(mu_A - A_alpha_cut(mu_A, alpha=0.5))

# Plot M2:
output_plot(
    data_dict= {
        "$\mu_A$" : {"x": s, "y": mu_A},
        "$\mu_{A_1/2}$" : {"x": s, "y": A_alpha_cut(mu_A, alpha=0.5), "linestyle":':'},
        "$f(x)_{M2}$": {"x": s, "y": fx_2(mu_A_=mu_A), "linestyle":'--'},
    },
    highlight_dict = {
        "highlight_x"   : s,
        "highlight_lb"  : mu_A,
        "highlight_ub"  : A_alpha_cut(mu_A, alpha=0.5),
        "label" : "$M_2$"
    },
    OUT_DIR = OUT_DIR_P1,
    tag = "M2",
)

# %% ---- Draw Membership Functions (b):
fx_3 = lambda mu_A_: np.abs(mu_A - (1 - mu_A))

# Plot M2:
output_plot(
    data_dict= {
        "$\mu_A$" : {"x": s, "y": mu_A},
        "$1 - \mu_A$" : {"x": s, "y": (1 - mu_A), "linestyle":':'},
        "$f(x)_{M3}$": {"x": s, "y": fx_3(mu_A_=mu_A), "linestyle":'--'},
    },
    highlight_dict = {
        "highlight_x"   : s,
        "highlight_lb"  : mu_A,
        "highlight_ub"  : (1 - mu_A),
        "label" : "$M_3$"
    },
    OUT_DIR = OUT_DIR_P1,
    tag = "M3",
)

# %%
```

Code 1: Main Code for P1

Appendix B Problem 3

```
# %% Lib:
import numpy as np

# custom lib:
import jx_lib
```

```python

#%% P1
OUT_DIR_P3="output/P3"
jx_lib.create_all_folders(DIR=OUT_DIR_P3)

# %%
V = np.linspace(0, 200, 21) # rev/s
v0 = 50 # rev/s
n = len(V)

F_LUT = {
    10: 0.1,
    20: 0.3,
    30: 0.6,
    40: 0.8,
    50: 1.0,
    60: 0.7,
    70: 0.5,
    80: 0.3,
    90: 0.1
}
F = [F_LUT[v] if v in F_LUT else 0 for v in V]
F_very = [F_LUT[v] if v in F_LUT else 0 for v in (V-v0)]
F_def = np.array(F) ** 2
F_pre = np.array(F) ** 0.5

# %%
jx_lib.output_plot(
    data_dict= {
        "Fast: $\mu_F(v)$" : {"x": V, "y": F},
        "Very Fast: $\mu_F(v-v_0)$" : {"x": V, "y": F_very, "linestyle":'-.'},
        "Definitely Fast: $\mu_F(v)^2$": {"x": V, "y": F_def, "linestyle":'--'},
        "Presumably Fast: $\mu_F(v)^{1/2}$": {"x": V, "y": F_pre, "linestyle":'--'},
    },
    Xlabel  = "$v$",
    Ylabel  = "$\mu$",
    OUT_DIR = OUT_DIR_P3,
    tag     = "Membership Functions",
)
```

Code 2: Main Code for P3

Appendix C Problem 5

```python
# %% Lib:
import numpy as np

# custom lib:
import jx_lib

#%% P1
OUT_DIR_P5="output/P5"
jx_lib.create_all_folders(DIR=OUT_DIR_P5)

# %%
X = [-4, 4]
step = 0.01
xs = np.linspace(X[0], X[1], int((X[1]-X[0])/step)+1)
n = len(xs)

f_mu_A = lambda x, params: np.exp(- params["lambda"] * (np.abs(x - params["a"]) ** params["n"]))


# %%
TESTS = {
    "Varying lambda, Fix (a=0 n=1)": {
        "default": {"lambda":1, "a":0, "n":1},
        "test-subject-name": "$\lambda={}$",
        "test-subject": "lambda",
        "test-values": [-0.01, 0, 1, 2, 3, 5, 10],
    },
```

```
    "Varying a, Fix (lambda=1 n=1)": {
        "default": {"lambda":1, "a":0, "n":1},
        "test-subject-name": "$a={}$",
        "test-subject": "a",
        "test-values": [-5, -3, -1.5, 0, 1.5, 3, 5],
    },
    "Varying n, Fix (lambda=1 a=0)": {
        "default": {"lambda":1, "a":0, "n":1},
        "test-subject-name": "$n={}$",
        "test-subject": "n",
        "test-values": [-15, -5, -1, 0, 1, 5, 15],
    },
}
for TAG, test_set in TESTS.items():
    data_dict = {}
    for val in test_set["test-values"]:
        params = test_set["default"]
        params[test_set["test-subject"]] = val
        data_dict[test_set["test-subject-name"].format(val)] = \
            {"x": xs, "y": f_mu_A(x=xs, params=params), "linestyle":'-'}
    jx_lib.output_plot(
        data_dict= data_dict,
        Xlabel  = "$x$",
        Ylabel  = "$\mu_A(x)$",
        OUT_DIR = OUT_DIR_P5,
        tag     = "Membership Function ({})".format(TAG),
        COLOR_TABLE = ["#5A4CA8", "#6D97C9", "#8CC3A0", "#E9DA90", "#F29D72", "#D17484", "#333333"],
    )

# %%
```

Code 3: Main Code for P3

Appendix D Problem 6

```
# %%
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
import skfuzzy as fuzz
from skfuzzy import control as ctrl
from skfuzzy import defuzzify
import copy

import jx_lib

#%% P6
OUT_DIR_P6="output/P6"
jx_lib.create_all_folders(DIR=OUT_DIR_P6)

# %% ---- Define:
class FuzzySys():
    members = {}

    def __init__(self):
        self.init_fuzzy()

    def init_fuzzy(self):
        # Antecedent: Input
        # Consequent: Output
        ang_universe = np.arange(-30,30,0.5)
        vel_universe = np.arange(-60,60,1.0)
        cnt_universe = np.arange(-3.0, 3.0, 0.05)

        # New Antecedent/Consequent objects hold universe variables and membership
        ang = {}
        vel = {}
        cnt = {}

        # Assign membership functions
        # Membership function: angle
```

```
        ang['PL'] = fuzz.trapmf(ang_universe, [-10, 20, 30, 30])
        ang['NL'] = fuzz.trapmf(ang_universe, [-30, -30, -20, 10])

        # Membership function: velocity
        vel['PL'] = fuzz.trapmf(vel_universe, [-20,40,60,60])
        vel['NL'] = fuzz.trapmf(vel_universe, [-60,-60,-40,20])

        # Membership function: control action
        cnt['PL'] = fuzz.trapmf(cnt_universe, [0,  2, 3, 3])
        cnt['NL'] = fuzz.trapmf(cnt_universe, [-3, -3, -2, 0])
        cnt['NC'] = fuzz.trimf(cnt_universe, [-2, 0, 2])

        self.members["ang"] = ang
        self.members["vel"] = vel
        self.members["cnt"] = cnt
        self.members["ang_universe"] = ang_universe
        self.members["vel_universe"] = vel_universe
        self.members["cnt_universe"] = cnt_universe


    def custom_visualize(
        self,
        ang = None,
        vel = None,
        tag = "",
        figsize = (15,15),
        OUT_DIR = OUT_DIR_P6,
        COLOR_TABLE = ["#b74f6fff","#628395ff","#dbad6aff","#dfd5a5ff","#cf995fff"],
    ):
        HEADER_COL = ["ang", "vel", "cnt"]
        HEADER_ROW = ['Rule 1', 'Rule 2', 'Rule 3', 'Rule 4' ]
        MEM_TABLE = [
            [ 'PL', 'PL', 'NL' ],
            [ 'PL', 'NL', 'NC' ],
            [ 'NL', 'PL', 'NC' ],
            [ 'NL', 'NL', 'PL' ],
        ]
        m,n = np.shape(MEM_TABLE)
        fig = plt.figure(figsize=figsize)

        where_ = lambda X, val, tol: next(i for i, _ in enumerate(X) if np.isclose(_, val, tol))

        cnt_universe, cnt_vals, cx_hat_centroid = None, None, None
        for j,row in enumerate(MEM_TABLE):
            mu_a_ang = None
            mu_a_vel = None
            # pie : prediction percentage
            for i,mem_func in enumerate(row):
                member_type = HEADER_COL[i]
                topic = MEM_TABLE[j][i]
                S = self.members["{}_universe".format(member_type)]
                mu_A = self.members["{}".format(member_type)][topic]
                ax = plt.subplot(m, n, j * n + i + 1)
                ax.plot(
                    S,
                    mu_A,
                    label="{}[{}]".format(member_type, topic),
                    color=COLOR_TABLE[i]
                )

                if member_type == "ang" and ang is not None:
                    mu_a_ang = mu_A[where_(X=S, val=ang, tol=0.01)]
                    mu_A_ang = np.minimum(mu_A, mu_a_ang)
                    ax.plot([ang, ang, S[-1]], [0, mu_a_ang, mu_a_ang],
                        linestyle='dashed', label="$\mu_{}={:.3f}$".format(ang,mu_a_ang))
                    ax.fill_between(S, mu_A_ang, color=COLOR_TABLE[i], alpha=0.3)

                if member_type == "vel" and vel is not None:
                    mu_a_vel = mu_A[where_(X=S, val=vel, tol=0.01)]
                    mu_A_vel = np.minimum(mu_A, mu_a_vel)
                    ax.plot([vel, vel, S[-1]], [0, mu_a_vel, mu_a_vel],
                        linestyle='dashed', label="$\mu_{}={:.3f}$".format(vel, mu_a_vel))
```

```
                    ax.fill_between(S, mu_A_vel, color=COLOR_TABLE[i], alpha=0.3)

                if member_type == "cnt" and vel is not None and ang is not None:
                    mu_a_cnt = min(mu_a_ang, mu_a_vel)
                    mu_A_cnt = np.minimum(mu_A, mu_a_cnt)
                    ax.axhline(y=mu_a_cnt,
                        linestyle='dashed', label="$\mu={:.3f}$".format(mu_a_cnt))
                    ax.fill_between(S, mu_A_cnt, color=COLOR_TABLE[i], alpha=0.3)
                    if cnt_vals is None:
                        cnt_vals = mu_A_cnt
                    else:
                        cnt_vals = np.maximum(cnt_vals,mu_A_cnt)

                ax.legend()
                if j == 0:
                    ax.set_title(HEADER_COL[i])
                if i == 0:
                    ax.set_ylabel(HEADER_ROW[j])

                ax.label_outer()


        # save:
        fig.tight_layout()
        fig.savefig("{}/plot_{}.png".format(OUT_DIR, tag), bbox_inches = 'tight', dpi=300)

        if cnt_vals is not None:
            # plot final aggregated control plot:
            fig = plt.figure(figsize=(10,8))
            ax = plt.subplot(1,1,1)
            # compute centroid:
            cnt_universe = self.members["cnt_universe"]
            cx_hat_centroid = (np.sum([u * c for u, c in zip(cnt_universe, cnt_vals)]) / np.sum(cnt_vals))
            cy_hat_centroid = np.sum(cnt_vals)/2/len(cnt_vals)
            # plot:
            ax.plot(cnt_universe, cnt_vals)
            ax.set_ylim(0, 1)
            ax.fill_between(cnt_universe, cnt_vals, color=COLOR_TABLE[2], alpha=0.3)
            ax.axvline(x=cx_hat_centroid, ymin=0, ymax=1, color="green", linestyle='dashed', label="Defuzzified value $\hat{c}$ (Centroid)")
            ax.scatter(cx_hat_centroid, cy_hat_centroid, color="red", label="Centroid")
            ax.legend()
            plt.title("Final Control Inference Plot")
            fig.savefig("{}/plot_final_control_{}.png".format(OUT_DIR, tag), bbox_inches = 'tight', dpi=300)

        return cnt_universe, cnt_vals, cx_hat_centroid




# %% ---- Run:
fuzzsys = FuzzySys()

# %% ---- Vis (a):
fuzzsys.custom_visualize(tag="Rule-Based Inference")

# %% ---- Vis (b):
cnt_universe, cnt_vals, cx_hat_centroid = fuzzsys.custom_visualize(tag="Control Inference", ang=5, vel=15)
print("Final Control Value: ", cx_hat_centroid)

# %%
```

Code 4: Main Code for P3