1.ابتدا متغیر ها و بازه آن هارا تعریف میکنیم

دمای آب : سرد [20,30] ولرم [30,50] گرم[70-50]

وزن ظرف : سبک [0,2] متوسط[2,4] سنگین[4,5]

چربی ظرف : کمی چرب [0,15] چرب[15,35] خیلی چرب[35,50]

سرعت موتور :خیلی پایین [0,15]پایین[15,30] متوسط[30,45] تند[45,60]

زمان شستشو : کوتاه[10,30] متوسط[30,60] طولانی[60,100]

قانون های فازی را هم صورت سوال مشخص کرده.

ورودی :

- وزن ظروف: 4 کیلوگرم (زیاد)

- درجه چربی: 45 (خیلی‌کثیف)

- دمای آب: 20 درجه (سرد)

پس به قانون - اگر ظروف خیلی‌کثیف و وزن زیاد و آب سرد باشد، سرعت موتور خیلی زیاد و زمان شستشو خیلی طولانی باشد.

میرسیم.

بنابراین، برای این وضعیت:

- سرعت موتور: خیلی زیاد (45 تا 60 دور در دقیقه)

- زمان شستشو: خیلی طولانی (60 تا 100 دقیقه)

.2

محاسبه عضویت‌ها:

عضویت چاق بودن:

برای نفر اول:

$\mu fat(55) = 1 - (-95/100)2 = 0.0975$

برای نفر دوم

$\mu fat(95) = 1- (-55/100)^2 = 0.6975$

عضویت لاغر بودن:

برای نفر اول

$\mu thin(55) = 1 - \mu fat(55) = 0.9025$

برای نفر دوم

$\mu thin(95) = 1 - \mu fat(95) = 0.3025$

عضویت خیلی لاغر بودن:

فرض کنیم که عضویت خیلی لاغر بودن از طریق درجه دو از عضویت لاغر بودن محاسبه می‌شود :

برای نفر اول:

$\mu verythin(55) = \mu thin(55) \wedge (1/2) = 0.8145$

عضویت جوان بودن:

برای نفر اول

$\mu young(45) = 0.0588$

برای نفر دوم

$\mu young(60) = 0.02$

محاسبه عضویت تقریباً جوان بودن:

برای نفر اول

$\mu almostyoung(45) = \mu young(45) \wedge (1/2) = 0.2425$

برای نفر دوم

$\mu almostyoung(60) = \mu young(60) \wedge (1/2) = 0.1414$

محاسبه تفاوت:

قدر مطلق تفاوت عضویت تقریباً جوان بودن بین دو نفر :

$|\mu almostyoung(60) - \mu almostyoung(45)| = 0.101$

تحلیل نهایی با استفاده از ممدانی:

بر اساس روش ممدانی، نتیجه نهایی با استفاده از عملگر "مینیمم" بین عضویت‌های خیلی لاغر بودن نفر اول و تفاوت تقریباً جوان بودن محاسبه می‌شود :

$\min(|\mu almostyoung(60) - \mu almostyoung(45)| , \mu verythin(55)) = 0.101$

پس میزان درستی 0.101 است.

3. الف) در ابتدای کد باید خروجی و ورودی های کد را مشخص کنیم که x1 , x2, y هستند. تعاریف زیر به این معنی هستند که x1 یک مقدار فازی بین 0 و 1 است که قادیر آن 0.01 تغییر میکند. همین مورد برای x2 است. و برای y هم که همانطور که داخل سوال گفته مقادیر آن بین 0.004 و 0.01 است که با مقادیر 0.0001 تغییر میکند.

```
x1 = ctrl.Antecedent(np.arange(0, 1.1, 0.01), 'x1')
x2 = ctrl.Antecedent(np.arange(0, 1.1, 0.01), 'x2')
y = ctrl.Consequent(np.arange(0.004, 0.01, 0.0001), 'y')
```

در ادامه توابع عضویتی که در داخل سوال به ما داده شده است را تعریف میکنیم. اینکار را به ازای همه ی توابع انجام داده ولی فقط برای یکی از آن ها توضیح خواهیم داد.

برای تابع عضویت μlow(x1) داریم :

این تابع را به ازای تمام مقادیری که x1 میتواند داشته باشد که در قسمت بالا مشخص کردیم، تعریف میکنیم. اگر مقادیر x1 از 0.4 کمتر بود مقدار تابع 1 بوده و اگر بین 0.4 و 0.7 بود، آن را از 7/3 + x 10/3- محاسبه میکنیم.

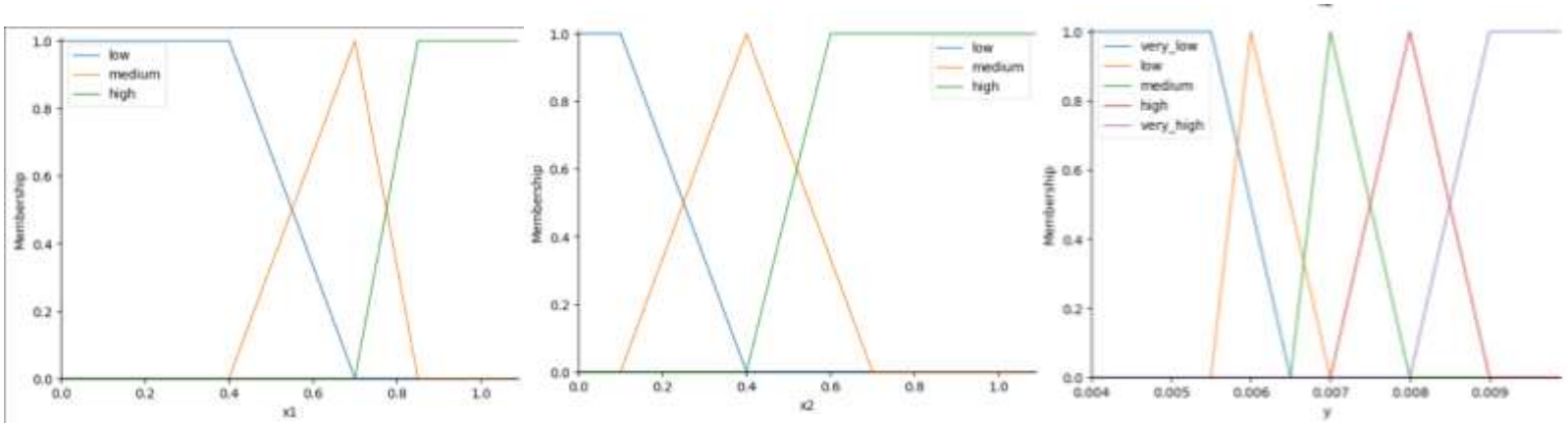
```
x1_low = np.zeros_like(x1.universe)
for i, val in enumerate(x1.universe):
    if val <= 0.4:
        x1_low[i] = 1
    elif 0.4 < val <= 0.7:
        x1_low[i] = -(10/3) * val + (7/3)
x1['low'] = x1_low
```

پس از آنکه همه ی توابع ساخته شد، نوبت به توابع عضویت y میرسد. باتوجه به شکلی که در صورت سوال داریم تابع هارا میسازیم. مثلا برای y خیلی پایین داریم :

چون به صورت ذوزنقه ای است از تابع trampf استفاده میشود. هر 4 مختصات x نقاط ذوزنقه را به عنوان یه لیست در ورودی داده و برای توابعی که مثلثی هستند، مختصات x 3 نقطه ی مثلث را میدهیم.

```
y['very_low'] = fuzz.trapmf(y.universe, [0.004, 0.004, 0.0055, 0.0065])
y['low'] = fuzz.trimf(y.universe, [0.0055, 0.006, 0.007])
y['medium'] = fuzz.trimf(y.universe, [0.0065, 0.007, 0.008])
y['high'] = fuzz.trimf(y.universe, [0.007, 0.008, 0.009])
y['very_high'] = fuzz.trapmf(y.universe, [0.008, 0.009, 0.01, 0.01])
```

خروجی توابع عضویت ساخته شده با استفاده از تابع view() :



حال در ادامه قوانین فازی را تعریف میکنیم با توجه به جدول داده شده . در این مرحله، 9 قانون فازی تعریف می‌شود که به ترکیب مقادیر x1و x2و تعیین مقدار y است.

```
rule1 = ctrl.Rule(x1['low'] & x2['low'], y['low'])
rule2 = ctrl.Rule(x1['low'] & x2['medium'], y['medium'])
rule3 = ctrl.Rule(x1['low'] & x2['high'], y['very_high'])
rule4 = ctrl.Rule(x1['medium'] & x2['low'], y['low'])
rule5 = ctrl.Rule(x1['medium'] & x2['medium'], y['medium'])
rule6 = ctrl.Rule(x1['medium'] & x2['high'], y['high'])
rule7 = ctrl.Rule(x1['high'] & x2['low'], y['very_low'])
rule8 = ctrl.Rule(x1['high'] & x2['medium'], y['low'])
rule9 = ctrl.Rule(x1['high'] & x2['high'], y['medium'])
```

در ادامه سیستم کنترلی را با قوانین ساخته شده تعریف میکنیم.

```
dosing_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8, rule9])
dosing = ctrl.ControlSystemSimulation(dosing_ctrl)
```

ب) ورودی هارا به سیستم فازی داده و خروجی :

```
dosing.input['x1'] = 0.5
dosing.input['x2'] = 0.6

dosing.compute()

print(dosing.output['y'])
```

```
0.008524764595103589
```

```
dosing.input['x1'] = 0.65
dosing.input['x2'] = 0.5

dosing.compute()

print(dosing.output['y'])
```

```
0.007857516339869286
```

4. ابتدا در این سوال باید مقادیر فازی ای که سرعت، موقعیت ماشین و اکشنی که ما میتوانیم انجام دهیم را مشخص کنیم. سرعت میتواند همانطور که در داک گفته شد مقادیری بین 0.07- تا 0.07 داشته باشد و تعداد مقادیر ممکن آن 100 تا است. برای موقعیت هم مقادیر بین 1.2- تا 0.6 است که میتواند 100 حالت داشته باشد. اکشن ها هم بین -1 تا 1 است که 100 حالت دارد. البته این 100حالت میتواند متفاوت باشد و بیشتر یا کمتر باشد ولی با آزمون و خطا این عدد بنظر مناسب می آید.

```
position_range = np.linspace(-1.2, 0.6, 100)
velocity_range = np.linspace(-0.07, 0.07, 100)
action_range = np.linspace(-1, 1, 100)
```

در ادامه باید توابع عضویت هر سه تا مورد را تعریف کنیم.

برای سرعت ماشین هم در نظر میگیریم که ماشین در حال حرکت به راست یا چپ یا سرعت خیلی کم دارد. و برای حالت حرکت به راست مقادیری که بالای 0 هستند تا 0.7 برای حرکت به چپ از 0 تا -0.07 و برای سرعت کم هم یک بازه ی کوچک مثلا 0.005 تا -0.005 را در نظر میگیریم البته این مقادیر با آزمون و خطا به دست آمد.

```
velocity_negative = fuzz.trimf(velocity_range, [-0.07, -0.07, -0.001])
velocity_neutral = fuzz.trimf(velocity_range, [-0.005, 0, 0.005])
velocity_positive = fuzz.trimf(velocity_range, [0.001, 0.07, 0.07])
```

```
velocity['negative'] = velocity_negative
velocity['neutral'] = velocity_neutral
velocity['positive'] = velocity_positive
```

برای موقعیت ماشین چیزی که برای ما اهمیت دارد قرار گرفتن ماشین در سربالایی یا سرپایینی است. بنابراین مقادیری که در این دو مورد قرار میگیرد را باتوجه به شکل داخل سوال بازه بندی میکنیم.

```
position_low = fuzz.trapmf(position_range, [-1.2, -1.2, -0.52, -0.45])
position_high = fuzz.trapmf(position_range, [-0.52, -0.45, 0.6, 0.6])
```

```
position['low'] = position_low
position['high'] = position_high
```

اکشن های ما میتواند نیرو به سمت راست و چپ و یا نیرویی وارد نکردن باشد.از آنجایی که بازه بین -1 تا 1 است پس از 1 تا 0 را نیرو به راست و -1 تا 0 را نیرو به سمت چپ و از -0.005 تا 0.005 را وارد نکردن نیرو در نظر میگیریم.

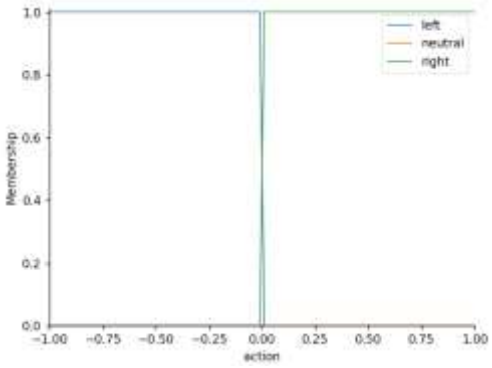
```
action_left = fuzz.trapmf(action_range, [-1, -1, -0.005, 0])
action_neutral = fuzz.trimf(action_range, [-0.005, 0, 0.005])
action_right = fuzz.trapmf(action_range, [0, 0.005, 1, 1])
```

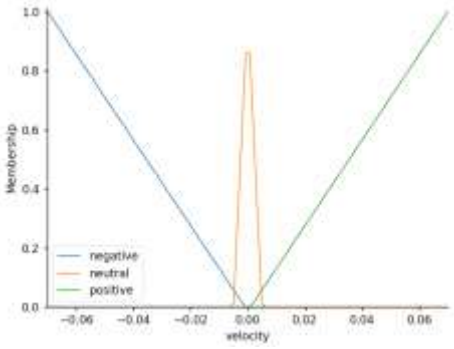
```
action['left'] = action_left
action['neutral'] = action_neutral
action['right'] = action_right
```

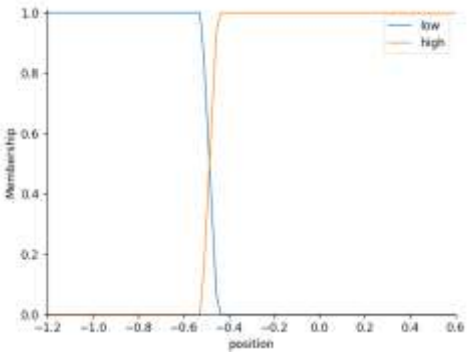
```
position = ctrl.Antecedent(position_range, 'position')
velocity = ctrl.Antecedent(velocity_range, 'velocity')
action = ctrl.Consequent(action_range, 'action')
```

در ادا مه متغیرهای فازی position, velocity به عنوان متغیرهای ورودی و actionبه عنوان متغیر خروجی را ایجاد میکنیم.

برای مشاهده ی توابع عضویت از ()view استفاده میکنیم.







قوانین را تعریف کرده مثلا اگر ماشین در سربالایی قرار داشت و سرعت آن منفی بود یعنی به سمت چپ حرکت میکرد ، نیرو به سمت چپ وارد کن. و برای بقیه حالات هم به همین شکل تعریف میکنیم.

```
rules = [
    ctrl.Rule(position['high'] & velocity['neutral'], action['neutral']),
    ctrl.Rule(position['high'] & velocity['negative'], action['left']),
    ctrl.Rule(position['high'] & velocity['positive'], action['right']),
    ctrl.Rule(position['low'] & velocity['negative'], action['left']),
    ctrl.Rule(position['low'] & velocity['positive'], action['right']),
    ctrl.Rule(position['low'] & velocity['neutral'], action['neutral']),
]
```

سیستم فازی را ساخته و هربار که باید یک اکشن انجام دهیم با توجه به سرعت و موقعیت سیستم فازی اکشن را compute میکند.
در اینجا باید اکسپشنی را هندل کنیم که مشاهده میکنید برای جلوگیری از ورودی‌های غیرمنتظره یا خارج از محدوده.

```
Error: Crisp output cannot be calculated, likely because the system is too sparse. Check to make sure this set of input
values will activate at least one connected Term in each Antecedent via the current set of Rules.
```

همچنین جهت بررسی قانون 500 قدم در سوال، هربار که یک اکشن را انجام میدهیم چک میکنیم که به 500 قدم نرسیده باشیم.و اگر موقعیت ماشین به 0.45 برسد بازی را متوقف میکنیم.