دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)
دانشکدهٔ ریاضی و علوم‌کامپیوتر

گزارش ۹: پیاده‌سازی یک سیستم فازی به منظور کنترل یک مسئله با ورودی‌های زبانی

نگارش
نیما حسینی دشت بیاض

استاد
دکتر مهدی قطعی

تیر ۱۴۰۰

مقدمه

در این تمرین، برای یک مسئله با ورودی‌های زبانی و عددی یک مدل کنترلر فازی تعریف می‌کنیم. چنین مدلی با دریافت ورودی‌های زبانی و ورودی‌های عددی و فازی کردن آن‌ها، با توجه یک مجموعه قانون و با استفاده از مجموعه‌های فازی ایجاد شده تصمیم‌گیری می‌کند و در صورت نیاز تصمیم را از حالت فازی به حالت عددی تبدیل می‌کند و به عنوان خروجی به سیستم می‌دهد.

در ادامه یک مسئله و مدل آن مطرح می‌شود و در زبان پایتون پیاده سازی می‌شود.

طرح مسئله

مسئله‌ای که در این گزاش به آن پرداخته می‌شود، کنترل سرعت فن پردازنده با استفاده از سه ورودی زیر است[1].

1. دمای پردازنده: این ورودی به صورت عددی یا crisp توسط سنسورها مشخص می‌شود و به عنوان ورودی به سیستم داده می‌شود.
2. فرکانس پردازنده یا clock: فرکانس پردازنده که اصطلاحا clock نامیده می‌شود، یک متغیر عددی اعشاری در بازه‌ی ۰ تا ۴ گیگاهرتز است و سرعت پردازنده را نشان می‌دهد. این ورودی نیز به صورت Crisp به مدل داده می‌شود.
3. تنظیمات مصرف نیرو یا power profile: این ورودی توسط کاربر سیستم مشخص می‌شود که می‌تواند دو مقدار زبانی «power saver» یا «performance» را اتخاذ کند. ورودی power saver نشان دهنده‌ی تمایل کاربر به مصرف بهینه‌ی باطری است و ورودی performance نشان دهنده‌ی اولویت عملکرد بهتر سیستم است.

این سیستم باید بتواند یک مقدار عددی را به عنوان سرعت چرخش فن پردازنده (دور در دقیقه یا RPM) خروجی بدهد. این خروجی عددی بین صفر تا ۶۰۰۰ است.

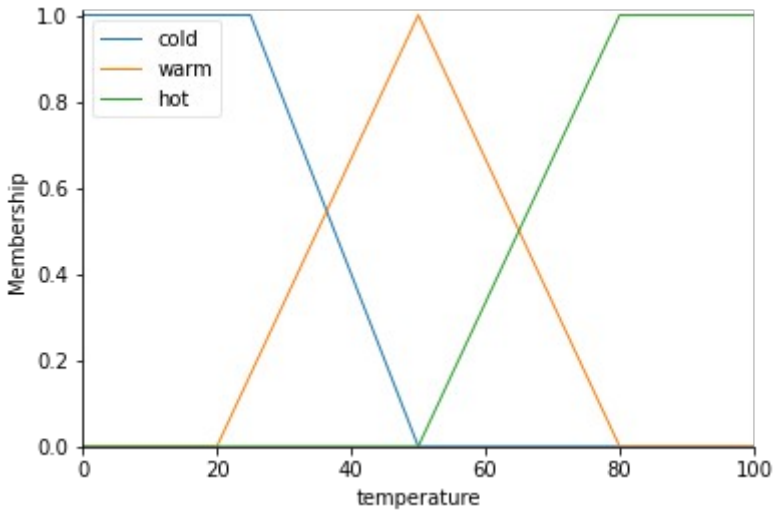
مدل فازی

برای حل مسئله با استفاده از یک سیستم فاز، لازم است که ابتدا ورودی‌های مسئله فازی شوند. بنابرین باید برای هر ورودی چند مجموعه‌ی فازی تشکیل دهیم تا مقدار عضویت ورودی‌ها در هر مجموعه‌ی فازی

مشخص شود. بنابرین مجموعه‌های فازی را برای هر ورودی به صورت زیر مشخص می‌کنیم.

دما

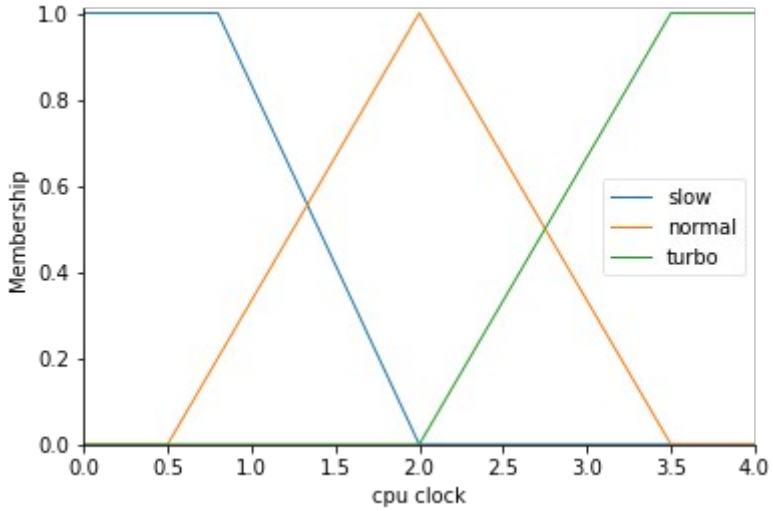
برای این ورودی، متغیر Temperature و سه مجموعه‌ی Cold, Warm و Hot را در نظر می‌گیریم. دامنه‌ی این متغیر دمای صفر تا ۱۰۰ درجه‌ی سانتی گراد است. مجموعه‌ی cold یک مجوعه‌ی فازی ذوذنقه‌ای است که بازه‌ی صفر تا ۲۵ درجه در آن دارای درجه عضویت یک است و دمای پنجاه درجه نیز دارای کمترین درجه‌ی عضویت این مجموعه است. به طور مشابه مجموعه‌ی Hot نیز ذوذنقه‌ای است از دمای ۵۰ درجه شروع می‌شود و از دمای ۸۰ تا ۱۰۰ درجه دارای درجه عضویت بیشینه است. مجموعه فازی warm در میان این دو مجموعه قرار می‌گیرد و مثلثی است. این مجموعه بازه‌ی ۲۰ تا ۸۰ درجه را شامل می‌شود و در دمای ۵۰ درجه دارای بیشترین عضویت است.



شکل ۱: مجموعه‌های فازی دما

فرکانس پردازنده

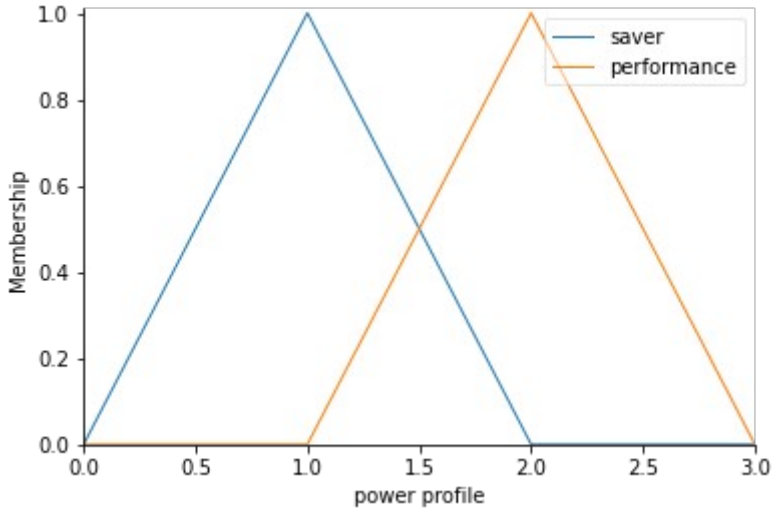
برای فرکانس پردازنده نیز یک متغیر CPU clock در نظر گرفته می‌شود که در بازه‌ی صفر تا ۴ گیگاهرتز تغییر می‌کند و سه مجموعه‌ی فازی slow, normal و turbo برای آن تعریف می‌شوند. مجموعه‌ی slow ذوذنقه‌ای در بازه‌ی صفر تا ۲ گیگاهرتز است و در بازه‌ی صفر تا ۸۰۰ مگاهرتز دارای بیشترین عضویت است. مجموعه‌ی normal مثلثی در بازه‌ی ۰.۵ تا ۳.۵ گیگاهرتز است و در نقطه‌ی ۲ گیگاهرتز دارای بیشترین عضویت است. مجموعه‌ی turbo نیز ذوذنقه‌ای در بازه‌ی ۲ تا ۴ گیگاهرتز است و از ۳.۵ تا ۴ گیگاهرتز بیشترین مقدار عضویت را دارد.



شکل ۲: مجموعه‌های فازی فرکانس پردازنده یا Clock

تنظیمات مصرف نیرو

سومین ورودی مدل، تنظیمات مربوط به مصرف باطری است که توسط کاربر مشخص می‌شود. از آن‌جایی‌که این ورودی به صورت یک متغیر زبانی دریافت می‌شود، می‌توان همان مقادیر ورودی را به عنوان مجموعه‌های فازی در نظر گرفت. بنابرین دو مجموعه‌ی فازی saver و performance برای این ورودی در نظر گرفته می‌شود که هر دو مثلثی‌اند.

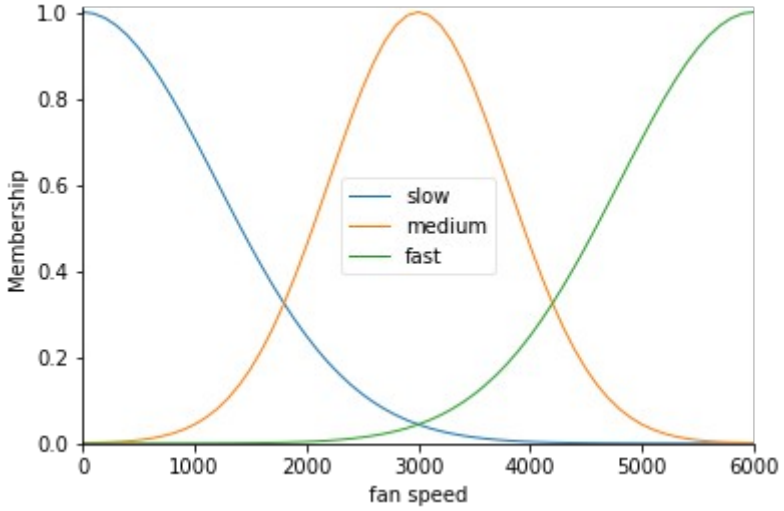


شکل ۳: مجموعه‌های فازی مصرف نیرو

علاوه بر ورودی‌ها، باید برای خروجی مسئله نیز مجموعه‌های فازی مناسب را تعریف کرد.

سرعت فن

بنابرین یک متغیر fan speed برای خروجی در نظر گرفته می‌شود که دامنه‌ی آن صفر تا ۶۰۰۰ است. سه مجموعه‌ی فازی slow, medium و fast تعریف می‌شود که هر سه از نوع گاوسی هستند. مجموعه‌ی slow تابع گاوسی با میانگین صفر و انحراف معیار ۱۲۰۰ است. به طور مشابه مجموعه‌های medium و fast هم توابع گاوسی با میانگین ۳۰۰۰ و ۶۰۰۰ و انحراف معیار ۸۰۰ و ۱۲۰۰ هستند.



شکل ۴: مجموعه‌های فازی سرعت فن

با تعریف مجموعه‌های فازی ورودی و خروجی، حال می‌توانیم قوانین استنتاج را برای مدل تعریف کنیم.

قوانین

برای تعریف قوانین باید توجه داشته باشیم که به طور منطقی با افزایش دمای پردازنده، لازم است تا سرعت فن افزایش یابد. علاوه‌بر آن، افزایش فرکانس پردازنده به‌طور طبیعی می‌تواند باعث افزایش دمای پردازنده شود. بنابرین با افزایش فرکانس، سرعت فن نیز افزایش می‌یابد (حتی اگر دما همچنان پایین باشد؛ افزایش سرعت فن از افزایش دما جلوگیری می‌کند). در کنار این دو ورودی، با اعمال تنظیمات مصرف نیرو، سعی می‌شود که سرعت فن را در حالت‌های مختلف کاهش داد که البته باعث کار کردن سیستم در دمای بالاتر می‌شود[1].

۱ هرچند که در این پیاده‌سازی این موضوع درنظر گرفته نشده، اما تنظیم سیستم برروی حالت power saver باید مانع بالا رفتن فرکانس پردازنده شود.

بنابرین با استفاده از نکات بالا، می‌توان قوانین را به صورت عبارات منطقی بین مجموعه‌های فازی مختلف بیان کرد. این قوانین در جدول زیر آمده است.

جدول ۱: قوانین مدل فازی

Temperature		Clock Speed		Power Profile		Fan Speed
cold	AND	slow	AND	S / P	THEN	slow
cold	AND	normal	AND	S / P	THEN	slow
cold	AND	turbo	AND	saver	THEN	slow
cold	AND	turbo	AND	performance	THEN	medium
warm	AND	slow	AND	S / P	THEN	slow
warm	AND	normal	AND	performance	THEN	medium
warm	AND	normal	AND	saver	THEN	slow
warm	AND	turbo	AND	saver	THEN	medium
warm	AND	turbo	AND	performance	THEN	fast
hot	AND	slow	AND	saver	THEN	medium
hot	AND	slow	AND	performance	THEN	fast
hot	AND	normal	AND	saver	THEN	medium
hot	AND	normal	AND	performance	THEN	fast
hot	AND	turbo	AND	S / P	THEN	fast

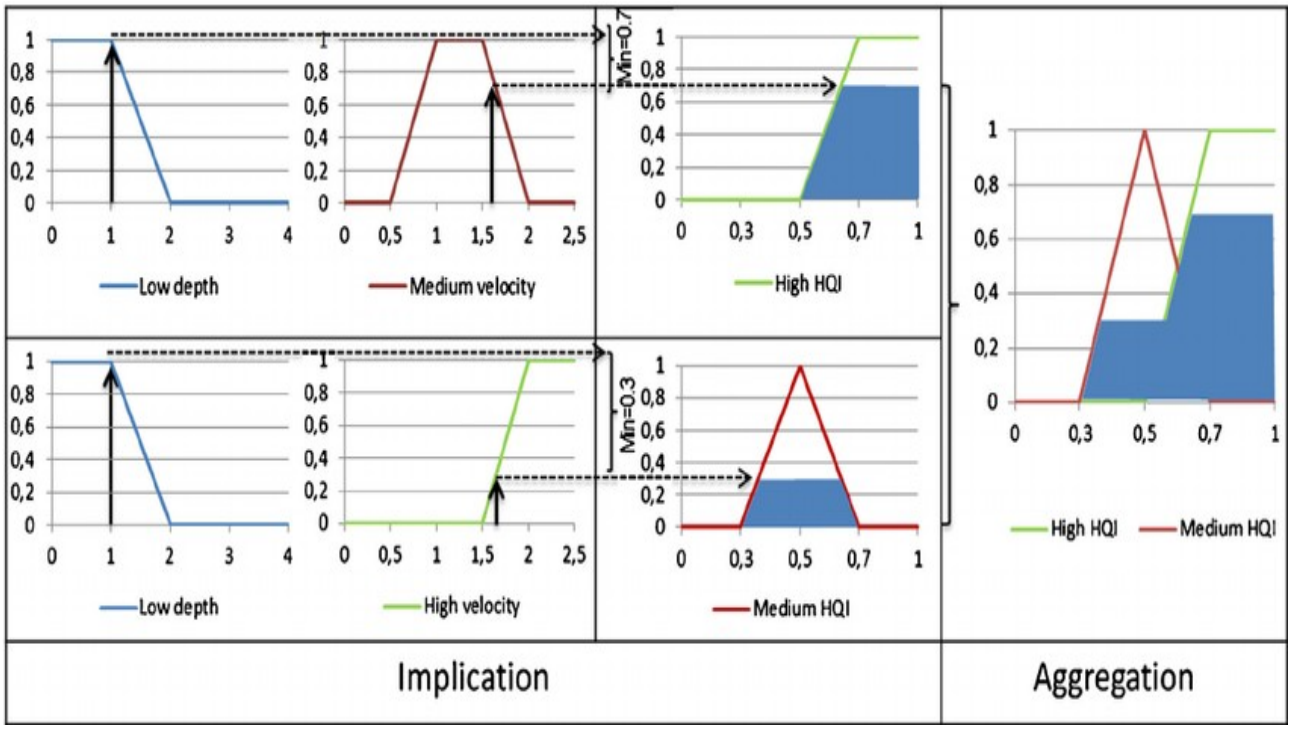
با داشتن همه‌ی این بخش‌ها، می‌توان به استنتاج و تصمیم‌گیری پرداخت.

استنتاج

برای استنتاج و تصمیم‌گیری از روش Mamdani استفاده می‌کنیم. در این روش مقدار درجه عضویت ورودی‌ها در مجموعه‌های سمت چپ هر شرط (Antecedentها) محاسبه می‌شود و مقدار عضویت متناظر آن در مجموعه‌ی سمت راست شرط (Consequent) محاسبه می‌شود؛ مثلا اگر عضویت در یک antecedent برابر ۰.۵ است، در Consequent یک خط روی مقدار عضویت ۰.۵ کشیده می‌شود تا بخشی از زیر نمودار مجموعه‌ی فازی اصلی انتخاب شود. سپس اگر بین Antecedent های یک شرط AND برقرار باشد، مینیموم خط‌های کشیده شده برای Consequent انتخاب می‌شود. این کار برای هر کدام از شرط‌ها انجام می‌شود و درنهایت نمودار حاصل از اجتماع آن‌ها رسم می‌شود. با داشتن این نمودارها، مقدار عضویت خروجی در هر کدام از مجموعه‌های فازی خروجی یا consequentها مشخص می‌شود[2].

برای به‌دست آوردن مقدار عددی یا crisp خروجی براساس نمودارهای حاصل، می‌توان از روش‌های مختلف

defuzzification مانند centroid استفاده کرد. در این روش با محاسبه‌ی مساحت زیر نمودارهای انتخاب شده، گرانی‌گاه آن محاسبه می‌شود و به عنوان مقدار عددی خروجی انتخاب می‌شود.



شکل ۵: نمونه‌ای از روش Mamdani برای دو شرط با دو Antecedent

با کامل شدن بخش‌های مختلف مدل، می‌توان مدل را در پایتون پیاده سازی کرد.

پیاده‌سازی

برای پیاده‌سازی از کتابخانه‌ی skfuzzy در زبان پایتون استفاده شده است[3][4]. ابتدا لازم است که ورودی‌های مدل را مشخص کنیم. برای این کار از کلاس Antecedent استفاده کنیم که نام و دامنه‌ی ورودی را می‌گیرد. سپس باید مجموعه‌های فازی برای هر ورودی را تعریف کرد. مجموعه‌های فازی مثلثی با استفاده از تابع trimf و مجموعه‌های ذوذنقه‌ای با تابع trampf تعریف می‌شوند.

```
temp = ctrl.Antecedent(np.arange(0, 101, 1), 'temperature')
clock = ctrl.Antecedent(np.arange(0, 4.1, 0.1), 'cpu clock')
power = ctrl.Antecedent(np.arange(0, 4, 1), 'power profile')
temp['cold'] = fuzzy.trapmf(temp.universe, [0, 0, 25, 50])
temp['warm'] = fuzzy.trimf(temp.universe, [20, 50, 80])
temp['hot'] = fuzzy.trapmf(temp.universe, [50, 80, 100, 100])
clock['slow'] = fuzzy.trapmf(clock.universe, [0, 0, 0.8, 2])
clock['normal'] = fuzzy.trimf(clock.universe, [0.5, 2, 3.5])
clock['turbo'] = fuzzy.trapmf(clock.universe, [2, 3.5, 4, 4])
power['saver'] = fuzzy.trimf(power.universe, [0,1,2])
power['performance'] = fuzzy.trimf(power.universe, [1, 2, 3])
```

متغیر خروجی یعنی fan speed نیز به شکل مشابه تعریف می‌شود، با این تفاوت که از کلاس Consequent استفاده می‌شود و برای تعریف مجموعه‌های فازی گاوسی نیز از تابع gaussmf استفاده می‌شود که میانگین و انحراف معیار تابع گاوسی را دریافت می‌کند.

```
fan_speed = ctrl.Consequent(np.arange(0, 6001, 100), label='fan speed')
fan_speed['slow'] = fuzzy.gaussmf(fan_speed.universe, 0, 1200)
fan_speed['medium'] = fuzzy.gaussmf(fan_speed.universe, 3000, 800)
fan_speed['fast'] = fuzzy.gaussmf(fan_speed.universe, 6000, 1200)
```

با تعریف شدن ورودی‌ها و خروجی‌ها، می‌توان قوانین را نیز پیاده‌سازی کرد. برای اضافه کردن قوانین از کلاس Rule در پکیج control استفاده می‌کنیم. برای هر قانون باید عبارت منطقی متناظر با آن را پیاده کرد. به طور مثال یک قانون به صورت زیر تعریف می‌شود.

```
ctrl.Rule(temp['cold'] & clock ['turbo'] & power['performance'], fan_speed['medium'])
```

در نهایت لیستی از تمام قوانین ایجاد می‌شود و به عنوان ورودی به کلاس ControlSystem داده می‌شود تا یک کنترلر با استفاده از آن قوانین ایجاد شود. برای استفاده از این کنترلر باید یک شی از کلاس ControlSystemSimulation ساخته شود. این شی با دریافت مقدار ورودی‌ها، استنتاج را انجام می‌دهد و مقدار عددی خروجی را به ما می‌دهد.

```
controller = ctrl.ControlSystem(rules)
speed = ctrl.ControlSystemSimulation(controller)
speed.inputs({'temperature':int(input_temp), 'cpu clock':float(input_clock), 'power profile':
int(input_power)})
speed.compute()
print(speed.output['fan speed'])
```

نمونه خروجی

برای مثال، در حالتی که دمای پردازنده ۵۰ درجه، فرکانس آن ۳ گیگاهرتز و حالت power saver فعال باشد، مدل مقدار 2514 دور در دقیقه را برای سرعت فن خروجی می‌دهد.

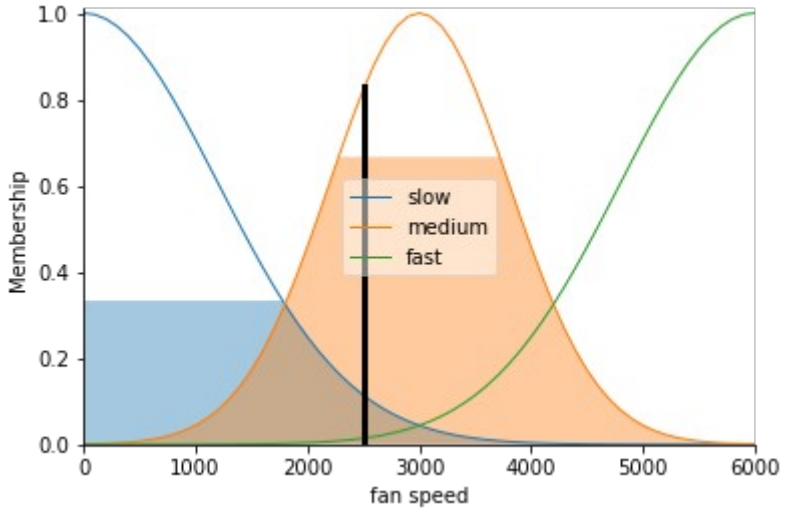
```
speed = ctrl.ControlSystemSimulation(controller)
input_temp = 50
input_clock = 3
input_power = 1  # 1 for power saving and 2 for performance
speed.inputs({'temperature':int(input_temp), 'cpu clock':float(input_clock), 'power profile': int(input_power)})
speed.compute()
print(speed.output['fan speed'])
```

```
2514.105973677407
```

شکل ۶: نمونه خروجی مدل

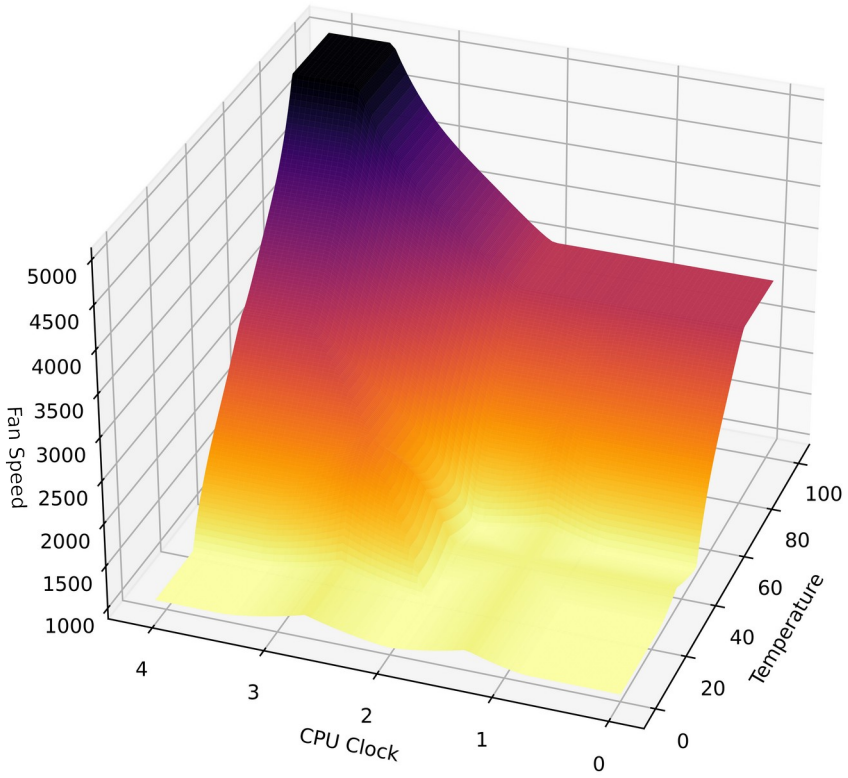
این مقدار بیشتر در محدوده‌ی سرعت Medium فن قرار می‌گیرد. از طرفی ورودی‌ها بیشترین عضویت را در مجموعه‌های Turbo و دمای warm دارند. همچنین Power Saver نیز فعال است. بنابرین طبق جدول شرط‌ها، خروجی باید در محدوده‌ی سرعت Medium باشد که همینطور است.

نمودار حاصل از روش Mamdani برای این ورودی به شکل زیر است.

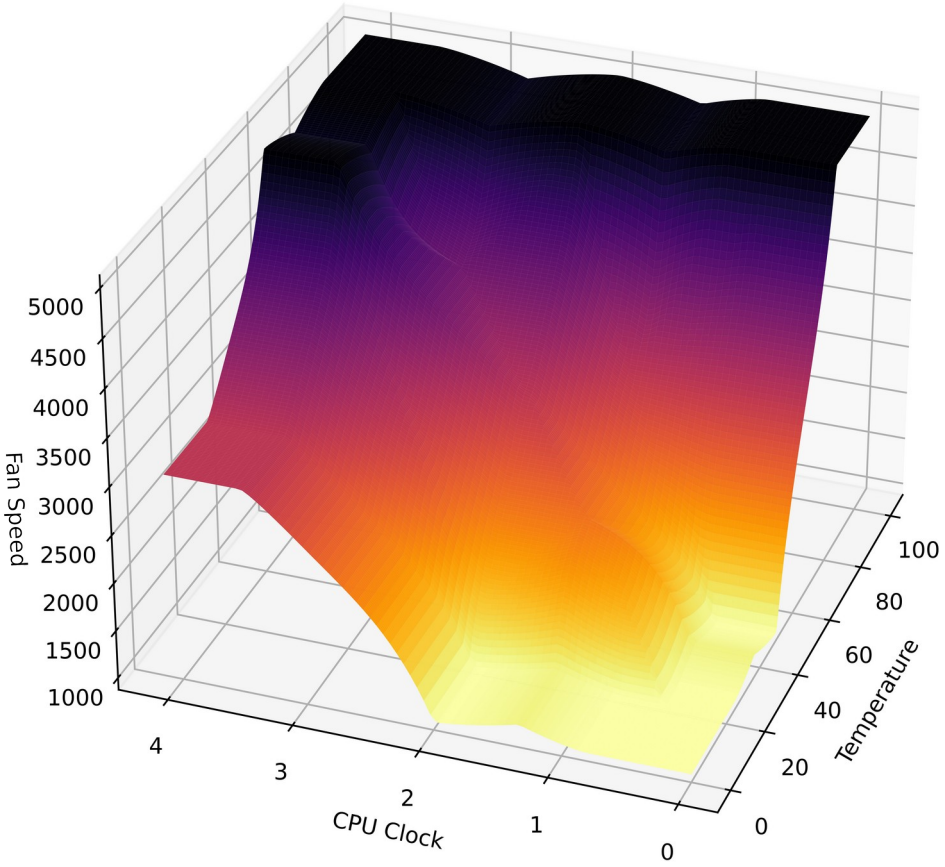


شکل ۷: نمودار حاصل از روش Mamdani برای ورودی نمونه

با در نظر گرفتن حالات مختلف ورودی در دامنه‌های مجاز آن‌ها، می‌توان نموداری برای خروجی مدل براساس ورودی‌های مختلف نیز به دست آورد. دو نمودار زیر مقدار خروجی را در دو حالت مختلف Power Saver و Performance نشان می‌دهند[5].



شکل ۸: سرعت فن برحسب ورودی‌های مختلف هنگامی که Power Saver فعال باشد.



شکل ۹: سرعت فن برحسب ورودی‌های مختلف هنگامی که Performance فعال باشد.

کد پروژه

سورس‌کد کامل پروژه برروی آدرس زیر در گیت‌هاب قرار دارد.

https://github.com/nimahsn/AI-Course-Projects/tree/main/report9-FuzzyLogic

منابع

1 https://github.com/raymelon/FanSpeed-Fuzzy-demo
2 https://www.youtube.com/watch?v=fqZvzQayx7A
3 https://pythonhosted.org/scikit-fuzzy/overview.html
4 https://github.com/makashy/FuzzyControllerExamples
5 https://scikit-fuzzy.readthedocs.io/en/latest/auto_examples/plot_control_system_advanced.html