



UNIVERZITET U NIŠU
ELEKTRONSKI
FAKULTET



FUZZYLITE LIBRARIES FOR FUZZY LOGIC CONTROL

SEMINARSKI RAD

Predmet: Fazi logika

Student:

Tina Radenković , br. ind. 1128

Mentor:

Prof. dr Miloš Radmanović

Niš, jun 2021. god.

SADRŽAJ

1. UVOD.....	3
2. KONTROLERI FAZI LOGIKE	4
2.1. Pravila zaključivanja.....	5
2.2. Operatori fazi logike	5
2.3. Konfiguracija kontrolera.....	5
2.3.1. Razlike između Mamdani i Takagi-Sugeno kontrolera.....	6
2.4. Operacije kontrolera fazi logike.....	7
3. IMPELEMENTACIJA KONTROLERA FAZI LOGIKE	8
4. FUZZYLITE BIBLIOTEKE	10
4.1. Klasni dijagram FuzzyLite biblioteke.....	10
4.2. Lingvističke promenljive	11
4.3. Lingvistički termini.....	12
4.4. Blokovi pravila.....	13
4.5. Defuzzifier	14
4.6. Engine	15
4.7. Uvoz i izvoz	15
4.8. FuzzyLite jezik.....	15
4.9. Factories.....	17
4.10. Usporedna analiza FuzzyLite biblioteke sa drugim bibliotekama za implementaciju kontrolera fazi logike	17
4.11. Instalacija FuzzyLite biblioteke	20
5. PRIMER IMPLEMENTACIJE KONTROLERA FAZI LOGIKE KORIŠĆENJEM PYFUZZYLITE BIBLIOTEKE.....	22
5.1. Sistem zaključivanja korišćenjem fazi logike (FIS)	22
5.2. Primer implementacije Mamdani kontrolera fazi logike	22
5.3. Primer implementacije Takagi-Sugeno kontrolera fazi logike	25
6. ZAKLJUČAK	29
7. REFERENCE.....	30

1. Uvod

Fuzzy Logic Controller (*FLC*) je softverska komponenta koja uređuje izlazne promenljive sistema na osnovu ulaznih promenljivih i skupa unapred definisanih pravila. Na primer, korišćenjem ovih kontrolera moguće je postaviti osvetljenost lampe na osnovu osvetljenosti prostorije. Ukoliko je niska osvetljenost prostorije jačina svetla biće visoka.

FLC se baziraju na fazi logici (eng. *Fuzzy logic*). Fazi logika ili rasplinuta logika predstavlja uopštenje klasične logike, razvijeno nad teorijom rasplnutih skupova. Za razliku od klasične logike gde iskazi imaju vrednosti tačno ili netačno, fazi logika koristi analitički aparat koji omogućava modeliranje iskaza čija istinitosna vrednost može pripadati kontinualnom prelazu od tačnog ka netačnom.

Fazi skupovi su osnovni elementi koji služe za opisivanje nepreciznosti. Naime, diskretni skup sadrži elemente sa istim svojstvima, dok fazi skupovi sadrže elemente sa sličnim svojstvima. U diskretnim skupovima element ili pripada ili ne pripada određenom skupu. Ako se ovo svojstvo predstavi matematički tada je stepen pripadnosti skupu 1 (ako pripada) ili 0 (ako ne pripada). Sa druge strane elementi u fazi skupovima mogu delimično da pripadaju, matematički predstavljeno na sledeći način: 1 (100% pripada), 0 (uopšte ne pripada skupu), 0.7 (70% pripada skupu).

Implementacija kontrolera fazi logike omogućena je korišćenjem različitih biblioteka (eng. *Fuzzy logic libraries*). Neke od najkorišćenijih su Matlab Fuzzy Logic Toolbox [1], Octave Fuzzy Logic Toolkit [2], jFuzzyLogic [3] i FuzzyLite [4] biblioteke.

Biblioteka Matlab Fuzzy Logic Toolbox je kreirana za Matlab okruženje [5] i ima podršku za Mamdani, Takagi-Sugeno kontrolere, preko 11 lingvističkih pojmova, četiri operatora fazi logike i sedam tzv. defuzzifier-a. Kontroleri se mogu izvoziti i uvoziti korišćenjem Fuzzy Inference System (*FIS*) formata.

Biblioteka Octave Fuzzy Logic Toolkit je bazirana na Octave okruženju [6] i poseduje sve karakteristike biblioteke Matlab Fuzzy Logic Toolbox sa dodatkom novih osam operatora fazi logike. Ova biblioteka je besplatna i otvorenog je koda (eng. *open source*).

Biblioteka jFuzzyLogic je implementirana korišćenjem Java programskog jezika. Ima podršku za Mamdani kontrolere, 14 jezičkih pojmova, 13 operatora fazi logike i 5 tzv. defuzzifier-a. Kontroleri se mogu izvoziti i uvoziti korišćenjem Fuzzy Control Language (*FCL*) formata. U implementaciji su korišćene nezavisne biblioteke i izvorni kod predstavlja mešavinu grafičkih elemenata i logike kontrolera. Ova biblioteka je besplatna i otvorenog koda.

Opisane biblioteke poseduju izvesna ograničenja u implementaciji poput skupog i restriktivnog licenciranja (Matlab Fuzzy Logic Toolbox i Octave Fuzzy Logic Toolkit) [1][2], lošeg dizajna i kompleksne implementacije (jFuzzyLogic) [3]. Sa osnovim ciljem da se prevaziđu ova ograničenja implementirana je biblioteka FuzzyLite [4]. Ova biblioteka je otvorenog koda, besplatna, komercijalna, objektno orijentisana čime je omogućena lakša i jednostavnija implementacija FLC kontrolera.

U ovom radu biće predstavljene osnovne karakteristike kontrolera fazi logike i detaljno predstavljena FuzzyLite biblioteka sa primerima korišćenja.

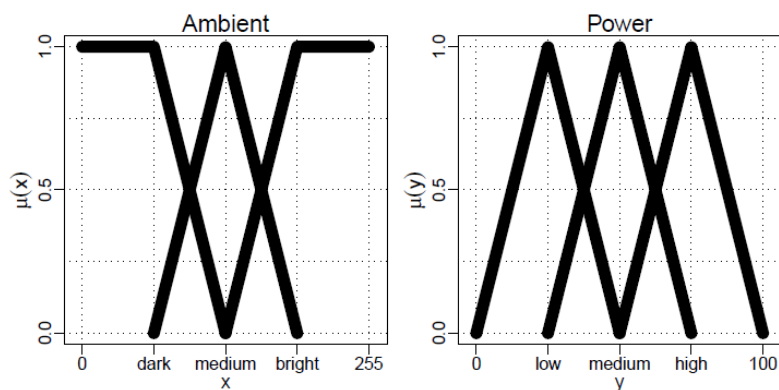
2. Kontroleri fazi logike

Kontroleri fazi logike su matematički modeli dizajnirani u cilju kontrolisanja sistema baziranih na fazi logici. Svaki kontroler se sastoji od skupa ulaznih podataka, izlaznih podataka i pravila zaključivanja relacija između podataka. Na primer, temperatura prostorije može biti kontrolisana korišćenjem kontrolera fazi logike upotrebom sledećih pravila: „ako je u prostoriji hladno, klima nije uključena“ i „ako je u prostoriji vruće, klima je uključena“. U ovim pravilima „prostorija“ označava ulaznu promenljivu koja predstavlja temperaturu kao „hladno“ i „toplo“, dok je izlazna promenljiva označena kao „klima“ koja može biti „uključena“ i „isključena“. Na ovaj način osnovne matematičke interpretacije fazi kontrolera su apstrahovane korišćenjem prirodnog jezika i zbog toga su oni pronašli veliku primenu u oblastima kao što su mašinsko učenje, kontrola dronova, autopilot, igrice i kontrola robota.

Kontroleri fazi logike modeliraju ulazne i izlazne parametre sistema kao lingvističke promenljive i kreiraju neophodna pravila kojima se reguliše tok rada sistema. Lingvističke promenljive predstavljaju skup izraza napisanih prirodnim jezikom koji se dodeljuju sistemskim podacima. Na primer, sistemska promenljiva *Ambient* je jezička promenljiva čije su vrednosti u opsegu od 0 do 255. Vrednosti ove promenljive u opsegu od 0 do 127 su definisane kao tamne, vrednosti u opsegu 64 do 191 kao srednje osvetljene i u opsegu od 128 do 255 kao svetle.

Skup pojmova jezičke promenljive je označen kao oštar (eng. *crisp*) kada svaka vrednost sistemske promenljive pripada jednom ili više linvističkih termina. Na primer, fotosenzor koji čita podatak $x=95$ je taman i srednje osvetljen jer pripada po opsegu u oba skupa vrednosti promenljive *Ambient*, dok nikako ne pripada skupu svetao.

Skup pojmova jezičke promenljive je označen kao tzv. fuzzy set kada svaka vrednost sistemske promenljive sa određenim stepenom sigurnosti pripada jednom ili više linvističkih termina određenim na osnovu njihovih funkcija članstva. Na slici 1 prikazan je primer modelovanja sistemske promenljive *Ambient* gde $\mu(x)$ predstavlja stepen sigurnosti vrednosti x , tada $x = 84$ daje fuzzy set $x = 0:68=\text{dark} + 0:32=\text{medium} + 0:0=\text{bright}$. Iz ovog razloga, oštar skup predstavlja specijalan slučaj fuzzy set-a gde je $\mu(x)$ iz opsega vrednosti 0 i 1.



Slika 1 Primer lingvističke promenljive

2.1. Pravila zaključivanja

Pravila zaključivanja su uslovni iskazi koji kontrolišu sistem. Svako pravilo se sastoji od dva dela: prethodnika i posledica. Oba dela sadrže propozicije oblika „promenljiva je pojam“. Propozicije u prethodniku mogu se povezati konjukcijom (*and*) ili disjunkcijom (*or*). Oba operatora su operatori fazi logike. Za razliku od njih, propozicije u posledicama su nezavisne jedne od drugih i mogu se odvojiti zarezom ili samo simbolom *and*.

Pojam u bilo kojoj propoziciji može biti prosleđen tzv. *hedge* komponenti koja modifikuje funkciju članstva kao neki od modalnih slučajeva poput vrlo, pomalo, retko i ne. Dodatno, važnost pravila može biti određena korišćenjem težine w čije vrednosti mogu biti $[0;0; 1;0]$.

Struktura prethodnika u pravilima je oblika “**if** promenljiva **is** [hedge]* pojam [**and/or** promenljiva **is** [hedge]* pojam]*”. Struktura posledice u pravilima je oblika „**then** promenljiva **is** [hedge]* pojam [**and** promenljiva **is** [hedge]* term] [**with** w]?”“. U obe strukture naglašeni pojmovi predstavljaju ključne reči, dok su delovi obuhvaćeni uglastim zagradaama opcioni, pomovi označeni * mogu se ponavljati jednom ili više puta, dok se pojmovi označeni ? mogu pojaviti jednom ili se ne pojaviti. Na primeru promenljive *Ambient* moguće je definisati skup pravila kao:

- **if** *Ambient* **is** mračan **then** *Power* **is** velika
- **if** *Ambient* **is** srednji **then** *Power* **is** srednja
- **if** *Ambient* **is** svetao **then** *Power* **is** mala

2.2. Operatori fazi logike

Operatori fazi logike definišu operacije između vrednosti sistema. Ukoliko se modeluje disjunkcija definiše se T-norma, dok u je u slučaju konjukcije definisana S-norma. Primeri T-normi su minimum i algebarski proizvod između dve vrednosti, dok su primeri S-normi maksimum i algebarski zbir dve vrednosti. Za operator *and* koji označava S-normu važe sledeći uslovi:

- komutativnost: $x*y = y*x$;
- monotonost: $\text{if } x \leq y \Rightarrow x*z \leq y*z$;
- asocijativnost: $x*(y*z) = (x*y)*z$;
- neutralnost nule: $0*x = x$ za svako x iz opsega $[0,1]$.

Osim toga, *and* je T-norma ukoliko ispunjava sledeće uslove:

- komutativnost: $x*y = y*x$;
- monotonost: $\text{if } x \leq y \Rightarrow x*z \leq y*z$;
- asocijativnost: $x*(y*z) = (x*y)*z$;
- neutralnost jedinice: $1*x = x$ za svako x iz opsega $[0,1]$.

2.3. Konfiguracija kontrolera

Svakodnevno se povećava potreba za sistemima koji koriste fazi logiku u sve većem broju inženjerskih aplikacija. Ove aplikacije su izazvale niz aktivnosti u analizi i dizajnu kontrolera fazi logike, pa je vremenom porastao i broj njihovih implementacija. Dva najčešće korišćena kontrolera fazi logike su: Mamdani kontroler i Takagi-Sugeno kontroler [7].

Konfiguracija Mamdani kontrolera podrazumeva četiri operatora fazi logike i tzv. defuzzier. Ključne reči *and* i *or* koje povezuju propozicije u delu pravila koji prethodi posledici definisane su T-normom, odnosno S-normom. Posledice pravila su određene *and* operatorom, odnosno T-Normom. Defuzzier se definiše kao metoda za pretvaranje svakog skupa ulaznih podataka u tzv. crisp skup. Tipičan primer defuzzier-a je centroid koji vrši izračunavanje tzv. crisp vrednosti iz centra mase fazi skupa.

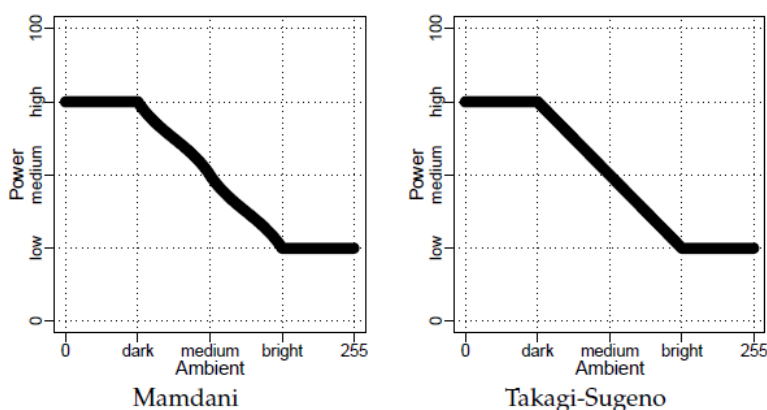
Glavna ideja u konfiguraciji Takagi-Sugeno kontrolera je izvođenje svakog kontrolnog pravila tako da se kompenzuje svako pravilo fazi logike. Postupak dizajniranja konceptualno je jednostavan i prirodan. Ovaj kontroler modeluje izlazne promenljive ugrađnim skupom funkcija. Termini su definisani polinomnim funkcijama oblika $f(x,y)=ax+by+c$, gde su x i y ulazne promenljive sistema, b je zadati koeficijent, dok c predstavlja konstantu. Raspored kontrolera je određen stepenom polinoma. Na primer, promenljiva sistema *Snaga* definisana kao slaba = 25, srednja = 50 i jaka = 75, ima koeficijente a i b postavljene na nulu, pa je stoga Takagi-Sugeno kontroler tzv. zero-order.

Konfiguracija Takagi-Sugeno kontrolera se sastoji od tri operatora fazi logike i različitih tipova defuzzier-a. Operatori su definisani na isti način kao i kod Mamdani kontrolera, T-norma i S-norma za određivanje operatora *and* i *or*, kao i dodatna T-norma za aktivacioni operator. Defuzzier se razlikuje od defuzzier-a Mamdani kontrolera po tome što izlazne promenljive ne pripadaju skupu fazi logike već su vrednosti i težine podataka izražene odgovarajućim funkcijama i stepenima aktivacije. Tipičan primer defuzzier-a je težinski prosek.

Kontroler zasnovan na Mamdani kontroleru, ali konfigurisan kao Takagi-Sugeno kontroler naziva se Tsukamoto [8].

2.3.1. Razlike između Mamdani i Takagi-Sugeno kontrolera

Glavna razlika između Mamdani i Takagi-Sugeno kontrolera ogleda se u pogledu interoperabilnosti, performansi i informacija. Interoperabilnost Mamdani kontrolera je bolja u odnosu na Takagi-Sugeno kontrolere zahvaljujući jednostavnijoj implementaciji i mogućnosti nadogradnje u kasnijem radu. U pogledu performansi, Mamdani kontroleri su računski skuplji i zahtevniji. Defuzzier-e koje oni koriste je potrebno integrisati u rezultujući fazi set, dok se defuzzier-i Takagi-Sugeno kontrolera sastoje od nekoliko aritmetičkih operacija. Pri implementaciji Takagi-Sugeno kontrolera potrebno je više informacija o samom sistemu kako bi se na adekvatan način definisale izlazne vrednosti sistema.



Slika 2 Razlika između Mamdani i Takagi-Sugeno kontrolera

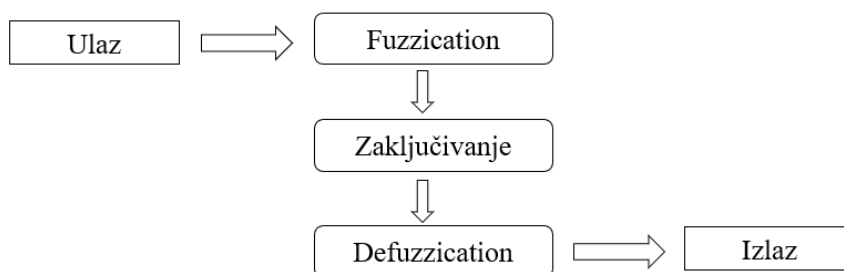
Razlika između opisanih kontrolera predstavljena je korišćenjem primera ulaznih i izlaznih vrednosti ovih kontrolera za promenljive *Ambient* i *Power* (Slika 2). Prikazane vrednosti podataka su slične, međutim postoje mala odstupanja na prikazanoj krivi koja je u slučaju Takagi-Sugeno kontrolera oštra, a u slučaju Mamdani kontrolera zakrivljena.

2.4. Operacije kontrolera fazi logike

Svaki kontroler fazi logike obavlja tri osnovne operacije: tzv. fuzzification, zaključivanje i tzv. defuzzification (Slika 3). U procesu tzv. fuzzification vrši se prevođenje ulaznih podataka koji su predstavljeni numerički u lingvističke termine. Skup lingvističkih termina ulazne promenljive i je definisan kao funkcija članstva $\mu_{ij}(x_i)$ za svaki termin j ulaznog podatka. Na primer, ukoliko fotosenzor očita podatak $x=168$, skup vrednosti fazi logike za komponentu *Ambient* će biti $x=0.0/\text{taman} + 0.36/\text{srednji} + 0.64/\text{svetao}$.

Sledeća faza, zaključivanje, podrazumeva primenu operacija fazi logike nad pripremljenim podacima, kao i izvođenje zaključaka. Za svako pravilo, propozicije prethodnika ocenjuju funkcije članstva njihovih termina. Ukoliko prethodnik u pravilu ima više od jedne propozicije, koriste se T-norma i S-norma odgovarajućih konektora za izračunavanje pojedinačne vrednosti iz funkcija članstva. Svaka dobijena vrednost se množi sa stepenom pravila i kreira se stepen aktivacije. Stepene aktivacije se koriste zajedno sa operatorom aktivacije kako bi se modifikovale funkcije članstva pojmova koji se javljaju u posledicama pravila.

Poslednja faza, tzv. defuzzification omogućava prevođenje izlaznih podataka predstavljenih lingvističkim terminima u podatke razumljive ljudima. Ova faza koristi defuzzier komponentu sistema i to najčešće zasnovanu na centroidama i na maksimumu za Mamdani kontrolere, odnosno težinskom proseku i sumi za Takagi-Sugeno ili Tsukamoto kontrolere. Defuzzier zasnovan na centroidama izračunava x vrednost centra mase fazi skupa. Maksimalni defuzzier određuje maksimalnu, srednju ili minimalnu vrednost maksimalne funkcije članstva, dok se težinski prosek i suma određuju nad modifikovanim funkcijama koristeći njihove stepene aktivacije kao težine. U slučaju Tsukamoto kontrolera, defuzzier koristi stepene aktivacije kao težine, dok funkcije članstva stepena aktivacije koristi kao vrednosti.



Slika 3 Operacije kontrolera fazi logike

3. Impelementacija kontrolera fazi logike

Razvoj fazi logike započet je 1965. godine zahvaljujući radu Loftija Zadeha (Lofti Zadeh) [9], dok danas ima više od 20 softverskih biblioteka koje implementiraju kontrolere fazi logike. Trenutno najkorišćenije biblioteke fazi logike su: Matlab Fuzzy Logic Toolbox, Octave Fuzzy Logic Toolkit, jFuzzyLogic i FuzzyLite biblioteke. Sve biblioteke imaju veliki broj atributa, otvorenog su koda, dobro su dizajnirane i dokumentovane.

Matlab Fuzzy Logic Toolbox je komponenta Matlab okruženja zadužena za razvoj kontrolera fazi logike. Ova biblioteka podržava biblioteku zasnovanu na C programskom jeziku čime je omogućeno kompajliranje kontrolera napisanih u Matlabu kao zasebnih aplikacija. Matlab okruženje i toolbox se prodaju odvojeno pod vlasničkim licencama. Izvorni kod Matlab Fuzzy Logic Toolbox-a je privatno dostupan sa dodeljenom licencom.

Pored skupa alata dostupnih za kreiranje i modeliranje kontrolera fazi logike ova biblioteka ima podršku za tzv. neuro-fuzzy modeliranje i klasterovanje podataka fazi setova. Trenutna verzija biblioteke je R2018b. Matlab Fuzzy Logic Toolbox sadrži sledeće komponente:

- kontrolere: Mamdani i Takagi-Sugeno,
- funkcije članstva: triangle, trapezoid, bell, gaussian, gaussian product, pi-shape, sigmoid difference, sigmoid product, sigmoid, s-shape, z-shape, constant, linear i proizvoljno implementirane funkcije,
- T-norme: minimum, proizvod i proizvoljno implementirane funkcije,
- S-norme: maksimum, algebarska suma i proizvoljno implementirane funkcije,
- tzv. defuzzifier: baziran na centru, simetriji, najmanji maksimum, težinski prosek, težinska suma i proizvoljno implementirane funkcije,
- uvoz kontrolera: FIS,
- izvoz kontrolera: FIS.

Octave Fuzzy Logic Toolkit je komponenta Octave okruženja zadužena za razvoj kontrolera fazi logike. Biblioteka je besplatna, licencirana od strane GNU General Public License. Dodatno uz mogućnost implementiranja kontrolera fazi logike ova biblioteka omogućava klasterovanje podataka fazi logike. Trenutna verzija alata je 0.4.5. Komplet alata pruža sledeće karakteristike:

- kontrolere: Mamdani i Takagi-Sugeno,
- funkcije članstva: triangle, trapezoid, bell, gaussian, gaussian product, pi-shape, sigmoid difference, sigmoid product, sigmoid, s-shape, z-shape, constant, linear i proizvoljno implementirane funkcije,
- T-norme: minimum, proizvod, ograničenu razliku, drastic proizvod, einstein proizvod, hamacher proizvod i proizvoljno implementirane funkcije,
- S-norme: maksimum, algebarska suma, ograničena suma, drastic suma, einstein suma, hamacher suma i proizvoljno implementirane funkcije,
- tzv. defuzzifier: baziran na centru, simetriji, najmanji maksimum, srednja vrednost maksimuma, najveći maksimum, težinski prosek, težinska suma i proizvoljno implementirane funkcije,
- uvoz kontrolera: FIS,
- izvoz kontrolera: FIS.

Biblioteka jFuzzyLogic je besplatna biblioteka otvorenog koda programirana za Java programski jezik, zadužena za razvoj kontrolera fazi logike. Biblioteka je licencirana kao GNU

Lesser General Public License 3 i Apache License 2.0. Pored osnovne namene biblioteke u dužijanju i modelovanju kontrolera fazi logike, ona ima podršku za optimizaciju parametara. Trenutna verzija biblioteke je 3.3. Biblioteka jFuzzyLogic sadrži sledeće komponente:

- kontrolere: Mamdani i Takagi-Sugeno,
- funkcije članstva: triangle, trapezoid, discrete, bell, cosine, gaussian, gaussian product, sigmoid difference, sigmoid, s-shape, z-shape, constant, linear i proizvoljno implementirane funkcije,
- T-norme: minimum, proizvod, ograničena razlika, drastic proizvod, hamacher proizvod i nilpotent maksimum,
- S-norme: maksimum, algebarska suma, ograničena suma, drastic suma, hamacher suma, nilpotent minimum i proizvoljno implementirane funkcije,
- tzv. defuzzifier: baziran na centru, simetriji, najmanji maksimum, srednja vrednost maksimuma, najveći maksimum, težinski prosek,
- uvoz kontrolera: FCL,
- izvoz kontrolera: FCL.

Biblioteka FuzzyLite je besplatna biblioteka otvorenog koda programirana za C++, Java i Python programske jezike. Kao dodatna mogućnost ove biblioteke izdava se korišćenje aplikativnog interfejsa qtFuzzyLite koji omogućava lakšu vizuelizaciju kreiranih kontrolera fazi logike. Biblioteka FuzzyLite sadrži sledeće komponente:

- kontrolere: Mamdani, Takagi-Sugeno i Tsukamoto,
- lingvističke termine: triangle, gaussian, sigmoid, bell, cosine, gaussian, gaussian product, sigmoid difference, sigmoid, s-shape, z-shape, constant, linear i proizvoljno implementirane funkcije,
- T-norme: minimum, proizvod, ograničenu razliku, drastic proizvod, einstein proizvod, hamacher proizvod i proizvoljno implementirane funkcije,
- S-norme: maksimum, algebarska suma, ograničena suma, normalizovana suma, drastic suma, einstein suma, hamacher suma i proizvoljno implementirane funkcije,
- tzv. defuzzifier: baziran na centru, simetriji, najmanji maksimum, srednja vrednost maksimuma, najveći maksimum, težinski prosek, težinska suma i proizvoljno implementirane funkcije,
- uvoz kontrolera: FIS i FCL,
- izvoz kontrolera: FIS i FCL,
- tzv. hedge: any, not, extremely, seldom, somewhat, very.

4. FuzzyLite biblioteke

FuzzyLite biblioteka je dostupna kao fuzzylite za C++, kao jfuzzylite za Java, kao pyfuzzylite za Python programski jezik i kao QtFuzzyLite za Qt aplikativni interfejs (Tabela 1). Osnovni ciljevi korišćenja ove biblioteke su olakšana implementacija i rad sa kontrolerima fazi logike oslanjajući se na objektno-orijentisano programiranje bez korišćenja dodatnih biblioteka. Biblioteka je kreirana 2010 godine.

Biblioteke FuzzyLite su programirane da budu objektno orijentisane, bez zavisnosti i kompatibilne za više platformi poput Windows, Linux i macOS operativnih sistema. Objektno orijentisan programski model apstrahuje sastvane komponente kontrolera fazi logike kao i njihove funkcionalnosti u klase i metode. Ove biblioteke omogućavaju implemetiranje kontrolera fazi logike bez primene drugih biblioteka pa su iz tog razloga označene kao biblioteke bez zavisnosti.

Programski jezik	FuzzyLite biblioteka
C++	fuzzylite
Java	jfuzzylite
Python	pyfuzzylite
Aplikativni interfejs	FuzzyLite biblioteka
Qt	QtFuzzyLite

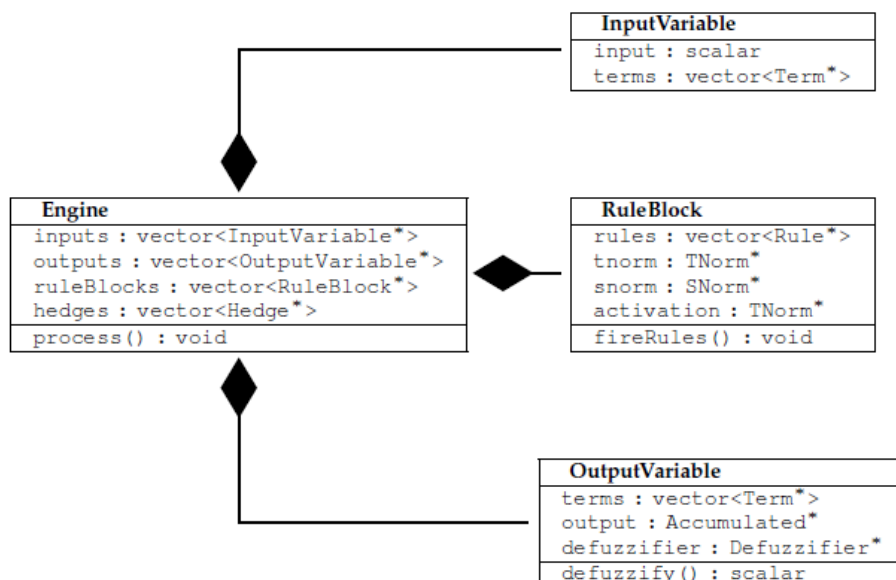
Tabela 1 FuzzyLite biblioteke

Razvoj ovih biblioteka je započet 2010. godine za fuzzylite i 2012. godine za jfuzzylite. Na osnovu COCOMO modela trenutna verzija biblioteke ima procenjene troškove razvoja od 888 155 američkih dolara na osnovu 14 949 fizičkih izvornih linija koda procenjeni potrebni napor od 41,10 meseci, i predviđeno trajanje od 10,26 meseci za četiri programera. Isto tako, procenjeni troškovi razvoja biblioteke jfuzzylite su 747 496 američkih dolara na osnovu 12 692 fizičkih izvornih linija koda, procenjeni potreban napor od 34,59 osoba-meseci, dok je predviđeno vreme trajanja 9,61 meseci za rad četiri programera [10].

4.1. Klasni dijagram FuzzyLite biblioteke

Klasni dijagram fuzzyLite biblioteke prikazan je na slici 4. Klase koje su korišćene su implemetaciji fazi kontrolera su:

- Engine: klasa predtsvalja kontrolere fazi logike, sadrži ulazne, izlazne podatke, skup pravila i tzv. hadges;
- InputVariable: klasa predstavlja ulazne podatke, sadrži skalar i vektor termina;
- OutputVariable: klasa predstavlja izlazne podatke, sadrži vektor termina, izlazni podatak i tzv. defuzzier;
- RuleBlock: klasa predstavlja pravila zaključivanja, sadrži vektor pravila, T-normu, S-normu i aktivacionu funkciju.



Slika 4 Klasni dijagram fuzzyLite biblioteke

4.2. Lingvističke promenljive

Lingvističke promenljive predstavljaju apstraktne koncepte koji označavaju promenljive kontrolnog sistema. U implementaciji fuzzyLite biblioteke one su predstavljene klasom *Variable*. Ulazne promenljive su predstavljene klasom *InputVariable*, dok su izlazne promenljive označene klasom *OutputVariable*. Obe klase su izvede iz klase *Variable*. Klasa *InputVariable* ima dodatnu metodu koja omogućava tekstualnu reprezentaciju ulaznih podataka, dok klasa *OutputVariable* ima dodatnu izlaznu promenljivu predstavljenu klasom *Aggregated*, implementiran tzv. defuzzier za prevođenje izlaznih podataka sistema u čitljive rezultate. Algoritam 1 predstavlja metodu određivanja izlazne promenljive.

Algoritam 1: Određivanje izlazne promenljive

```

if  $y^{t-1}$  ne pripada  $\{-\infty, \infty, \text{NaN}\}$ 
     $y^t = y^{t-1}$  // ažuriranje prethodne vrednosti u trenutku t
if  $y^t$  različito od 0
     $y^t = \text{defuzzify}(y^t, a, b)$  // tzv. defuzzify trenutne vrednosti
else
    if  $l^y = T$ 
         $y^t = y^t$  // koristi prethodnu vrednost u trenutku t
    else
         $y^t = k$  // koristi podrazumevanu vrednost
if  $l_v = T$ 
     $y^t = a$ , ako je  $y^t < a$ 
     $y^t = b$ , ako je  $y^t < b$ 
     $y^t = y^t$ , u ostalim slučajevima
  
```

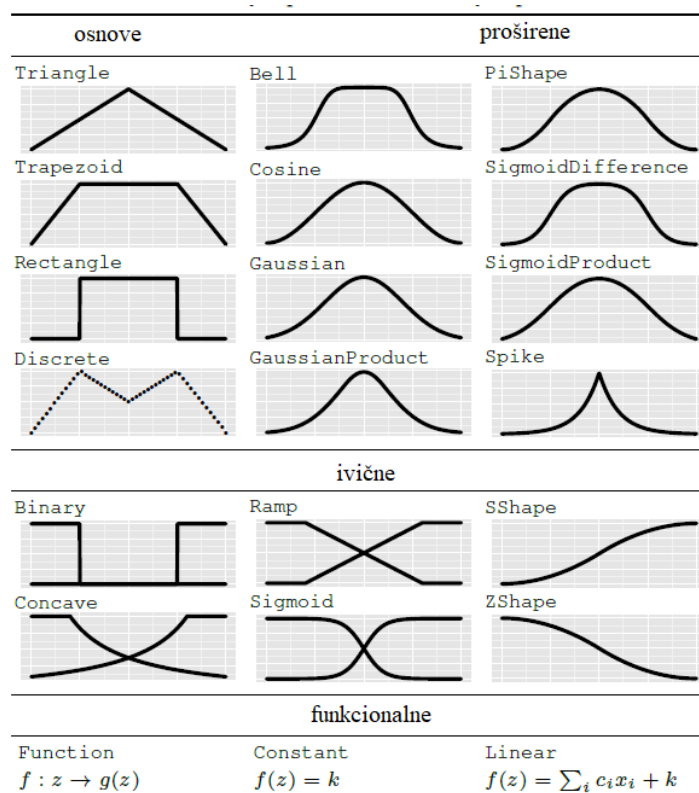
4.3. Lingvistički termini

Lingvistički termini ili pojmovi su apstraktna stanja jezičkih promenljivih. Predstavljeni su klasom *Term* koja sadrži imena, visinu h_2 koja množi vrednost funkcije članstva i metodu *membership* za određivanje vrednosti funkcije članstva.

Ova klasa ima 21 različitu implementaciju lingvističkih termina gde svaka ima drugačiju funkciju članstva. Funkcije su podeljene na: osnovne, proširene, ivične i funkcionalne. Grupa osnovnih funkcija koristi samo aritmetičke operacije. Grupa proširenih funkcija čine funkcije koje pored aritmetičkih operacija koriste i trigonometrijske funkcije. Ivične funkcije obuhvataju monotone funkcije, dok funkcionalna grupa sadrži linearne i prilagođene funkcije (Slika 5).

Pored predstavljenih lingvističkih termina, fuzzyLite biblioteka poseduje i *Activated* i *Aggregated* termine koji označavaju apstrakciju faza zaključivanja i tzv. defuzzification u modelovanju kontrolera.

Activated termini označavaju modulaciju lingvističkih termina iz određenih uslova pravila. Ovi termini su označeni sa p i sadrže aktivacioni stepen, operator kao i aktivacioni termin. *Aggregated* termini označavaju agregaciju svih aktivnih termina koji se pojavlju u izlaznim podacima. Označavaju se sa q i sadrže n aktivnih termina i skup operatora agregacije. U slučaju Takagi-Sugeno i Tsukamoto kontrolera implikacija i operatori agregacije njihovih lingvističkih pojmova mogu se odrediti korišćenjem množenja ili sabiranja.



Slika 5 Lingvistički termini

4.4. Blokovi pravila

Blokovi pravila sadrže pojedinačna pravila koja kontrolišu ceo sistem. Predstavljeni su klasom *RuleBlock* koja sadrži definicije imena i opisa, implementaciju skupa pravila, aktivacione metode, kao i fazi operatore: konjunkcija, disjunkcija i implikacija. Kako fazi kontroleri ne poseduju operator agregacije u blokovima pravila, već u agregiranim terminima izlaznih podataka, oni se mogu konfigurisati korišćenjem višestrukih blokova pravila primenjenih nad istim skupom izlaznih podataka.

Pravila su označena klasom *Rule* koja ima instance klasa *Antecedent* i *Consequent* koje predstavljaju prethodnike i posledice jednog pravila, aktivacioni stepen i težinu. Prethodnik u pravilu nastaje parsiranjem ulaznog teksta uz pomoć Shunting-Yard i Finite State Machines algoritma. Primenjivanjem pravila ovih algoritama određuje se binarno stablo propozicija. Korišćenjem ovakve strukture podataka, prethodnik može imati različit broj predloga sa različitim brojem pridruženih operatora koji su dodeljeni poštujući konvencionalni redosled operatora. Ocena propozicije „variable is term“ rezultira nulom ukoliko je promenljiva označana za nekorišćenje ili kao vrednost funkcije članstva. Ako propozicija sadrži tzv. hedges, hedge koji se nalazi na krajnjoj desnoj strani se prvi primenjuje na funkciju članstva, dok se ostali redom primenjuju na dobijeni rezultat prethodnog hedge-a. Na primer, propozicija „service is seldom not very good“ evoluirala u $\text{seldom}(\text{not}(\text{very}(\mu_{\text{good}}(x_{\text{service}}))))$. Dodatno, prethodnik u pravilu može se pored ulaznih parametara primeniti i na izlaze parametre. U tom slučaju evoluiranje propozicije zavisi od aktivacionog stepena termina u izlaznoj promenljivoj.

Posledice pravila se parsiraju iz teksta korišćenjem FSM kako bi se sastavio spisak predloga koji određuje termine koji će se aktivirati u izlaznim promenljivim. Kada je pravilo aktivirano, termini u propozicijama se dodeljuju novim aktivacionim terminima, poštujući odgovarajuće aktivacione stepene kao i operatore implikacije (u slučaju Mamdani kontrolera fazi logike). Ovako aktivirani uslovi se zatim dodeljuju agregiranim terminima odgovarajućih izlaznih podataka.

Operatori fazi logike koji se koriste u blokovima pravila su konjunkcija, disjunkcija i implikacija. Blok pravila zahteva operator konjunkcije ukoliko postoji bar jedno pravilo koje koristi „i“ vezu. Na sličan način, koristi operator disjunkcije ukoliko bar jedno pravilo koristi vezu „ili“. U slučaju Mamdani kontrolera operatori implikacije i agregacije su nepohodni pri kreiranju kontrolera fazi logike, dok to nije slučaj pri modelovanju kontrolera tipa Takagi-Sugeno i Tsukamoto.

Operatori implikacije i konjunkcije su predstavljeni klasom *TNorm*, dok su operatori disjunkcije i agregacije označeni klasom *SNorm*. Na slici 6 su prikazane implementacije T-normi i S-normi koje su dostupne u FuzzyLite biblioteci. Pored ovih normi moguća je proizvoljna implementacija normi korišćenjem metoda *TnormFunction* i *SnormFunction*.

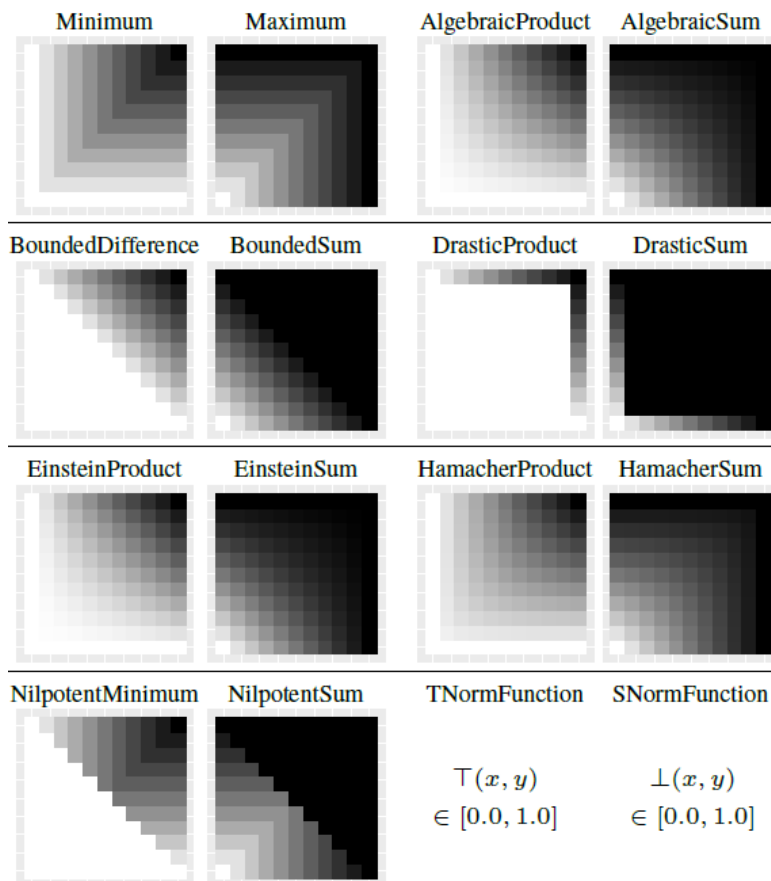
Metode aktivacije u pravilima se odnose na strategije za aktiviranje pravila u bloku pravila. Ukupno postoji sedam implementacija metoda aktivacije u okviru FuzzyLite biblioteke i to su: opšte, najviše, najniže, prve, poslednje, proporcijalne i prazne. Opšti način aktivacije aktivira pravila u redosledu koji je dodeljen bloku pravila. Najviše i najniže metode aktivacije sortiraju pravila na osnovu stepena aktivacije (uzlazno i silazno, respektivno) i aktiviraju samo prvih n pravila. Prva i poslednja metoda aktivacije aktiviraju prvih odnosno poslednjih n pravila čiji su stepeni aktivacije veći ili jednaki zadatom pragu. Metoda proporcionalne

aktivacije izračunava i normalizuje stepene aktivacije pravila tako da se iz skupa od n pravila, svako pravilo i aktivira kao

$$\sum_{i=1}^n \alpha_i^* = 1.0, \text{ umesto}$$

$$\alpha_i^* = \frac{\alpha_i}{\sum_{j=1}^n \alpha_j}.$$

Pražne metode aktiviraju pravila čiji aktivacioni stepeni zadovoljavaju $\alpha \otimes \theta$ gde \otimes označava realacioni operator iz skupa $=, \neq, <, >, \geq, \leq$, a θ realan broj koji predstavlja prag aktivacije.



Slika 6 Dostupne T-norme i S-norme u FuzzyLite biblioteci

4.5. Defuzzifier

Tzv. defuzzifier-i pretvaraju fazi vrednosti izlaznih podataka u tzv. crisp vrednosti. U modelu klasa FuzzyLite biblioteke oni su predstavljeni apstraktnom klasom *Defuzzifier* koja je nadklasa dvema apstraktnim klasama za integraciju i određivanje težine, *IntegralDefuzzifier* i *WeightedDefuzzifier*, respektivno. Tzv. defuzzification metode koje implementira klasa *IntegralDefuzzifier* primenjuju integrale nad izlaznim fazi vrednostima sa unapred definisanom rezolucijom r . Ova rezolucija je najčešće postavljena na vrednost 100. Rezolucija određuje broj podintervala fazi vrednosti uravnotežujući tačnosti i performansi kontrolera fazi logike. Naime, veća rezolucija pruža tačniju defuzifikaciju na uštrb performansi dok niža rezolucija omogućava bolje performanse sa manjom tačnošću.

U okviru FuzzyLite biblioteke postoji pet implementacija *IntegralDefuzzifier-a* to su: implementacija bazirana na centru, na simetriji, najmanji maksimum, srednji maksimum i najveći maksimum. Implementacija bazirana na centru određuje x koordinatu geometrijskog

centra fazi vrednosti. Implementacija bazirana na simetriji određuje x koordinatu koja deli fazi vrednost na dva jednaka dela. Najmanji maksimum, srednji maksimum i najveći maksimum određuju najmanju, srednju i najveću vrednost x koordinate globalnog maksimuma fazi vrednosti respektivno.

Tzv. defuzzification metode koje implementira klasa *WeightedDefuzzifier* koriste vrednosti aktivacionih termina i njihovih stepena aktivacije kao težine. U slučaju Takagi-Sugeno kontrolera fazi logike vrednosti funkcije dobijaju se iz izraza koji koriste isti način članstva, dok u slučaju Tsukamoto kontrolera monotoni termini primenjuju metodu *tsukamoto* kako bi zadovoljili jednačinu. Postoji dve implementacije *WeightedDefuzzifier-a* i to su implementacije zasnovane na težinskom proseku i na težinskom zbiru fazi vrednosti izlaznih podataka.

4.6. Engine

Kontroleri fazi logike su predstavljeni klasom *Engine* koja se sastoji od imena i opisa, skupa ulaznih promenljivih, skupa izlaznih promenljivih i skupa blokova pravila. Podržana su četiri načina zaključivanja: Mamdani, Larsen, Takagi-Sugeno i Tsukamoto koji se automatski određuju na osnovu konfiguracije komponenti.

Objektno orijentisani dizajn pruža mogućnost konfigurisanja jednog kontrolera fazi logike koristeći više vrsta zaključivanja istovremeno. Na primer, moguće je projektovati hibridni kontroler sa jednim ili više blokova pravila koji primenjuju Mamdani, Takagi-Sugeno i Tsukamoto način zaključivanja na skupu promenljivih.

4.7. Uvoz i izvoz

Kontroleri fazi logike mogu biti predstavljeni koristeći programski jezik koji se razlikuje od programskog jezika koji je korišćen za njihovu implementaciju. Na ovaj način, moguće je umanjiti složenost programiranja, fokusirati se na konfiguraciju kontrolera fazi logike, pripremiti kontroler za skladištenje i izvoz. Dakle, uvoznik je komponenta koja kreira i konfiguriše kontrolere fazi logike, dok izvoznik opisuje FLC koristeći određeni tip reprezentacije kontrolera.

Uvoznik je predstavljen klasom *Importer*, dok je izvoznik označen klasom *Exporter*. U okviru FuzzyLite biblioteke dostupni su *FllImporter* i *FllExporter* koji koriste FuzzyLite jezik, *FisImporeter* i *FisExporter* koji koriste FIS format, *FclImporter* i *FclExporter* koji koriste FCL format. Klase *CppExporter* i *JavaExporter* implementiraju izvorni kod kontrolera koristeći biblioteke *fuzzyLite* i *jfuzzylite*. Koristi se i *FldExporter* koji procenjuje kontroler kako bi generisao tzv. plain-based i column-base FuzzyLite Dataset (FLD). Ovako kreiran skup podataka sadrži ulazne vrednosti, izlazne vrednosti i zaglavlja kolona koje su odvojene zadatim graničnikom. *RscriptExporter* kreira skriptu R koristeći biblioteku *ggplot2* za iscrtavanje kontrolnih površina svake izlazne promenljive za bilo koji dati par ulaznih promenljivih.

4.8. FuzzyLite jezik

FuzziLite jezik (FLL) je jezik koji se koristi u implementaciji kontrolera fazi logike kako bi se ostvario jednostavniji, fleksibilniji, sažetiji i efikasniji dizajn kontrolera u odnosu na FIS i FCL formate. Na slici 7 prikazana je struktura FuzzyLite jezika. Uvlačenje i razdvajanje

svojstava je proizvoljno, podrazumevano se koriste dva razmaka za uvlačenje. Komentari počinju sa #, ime i vrednost svojstva su tzv. case-sensitive i odvojeni su dvotačkom, parametri su odvojeni razmacima, a ključna reč *none* označava neodređenu vrednost. Svojstva *term* i *rule* se ponavljaju za svako pravilo i za svaki pojam. Uglaste zagrade oko vrednosti (npr. [parametar]) i linija između dve vrednosti (npr. a | b) nisu deo jezika već uglaste zagrade ukazuju na to da vrednost nije obavezna dok linija označava da bar jedna vrednost od zadate dve mora biti prisutna.






















```
# Comment: text
Engine: string
  description: text
InputVariable: string
  description: text
  enabled: boolean
  range: scalar scalar
  lock-range: boolean
  term: string Term [parameters]
OutputVariable: string
  description: text
  enabled: boolean
  range: scalar scalar
  lock-range: boolean
  aggregation: SNorm|none
  defuzzifier: [Defuzzifier [parameter]]|none
  default: scalar
  lock-previous: boolean
  term: string Term [parameters]
RuleBlock: string
  description: text
  enabled: boolean
  conjunction: TNorm|none
  disjunction: SNorm|none
  implication: TNorm|none
  activation: [Activation [parameter]]|none
  rule: if antecedent then consequent
```

Slika 7 Struktura FuzzyLite jezika

Dostupni tipovi podataka su:

- string: niz slova i brojeva, bez razmaka i specijalnih karaktera;
- text: proizvoljan tekst u jednom redu;
- boolean: može imati vrednosti tačno ili netačno;
- scalar: tzv. floating-point vrednost.

Pojam *Term* se odnosi na naziv klase lingvističkog termina i opciono posle njega se može postaviti parametar za konfiguraciju funkcije članstva (Slika 8). Klase *SNorm* i *Tnorm* odgovaraju S-normi i T-normi, klasa *Defuzzifier* tzv. defuzzifier komponenti. Klasa *Activation* odgovara aktivacionoj metodi koja se koristi pri implementaciji kontrolera.

term		parameters			
		first	second	third	fourth
Bell		center	width	slope	
Binary		start	direction		
Concave		inflection	end		
Constant		value			
Cosine		center	width		
Discrete		x_0	y_0	\dots	x_i y_i \dots x_n y_n
Function		$f : x \rightarrow \mathbb{R}$			
Gaussian		mean	stdev		
GaussianProduct		mean	stdev	mean	stdev
Linear		c_0	\dots	c_i \dots	c_n k
PiShape		bottom left	top left	top right	bottom right
Ramp		start	end		
Rectangle		start	end		
Sigmoid		inflection	slope		
SigmoidDifference		left	rising	falling	right
SShape		start	end		
SigmoidProduct		left	rising	falling	right
Spike		center	width		
Trapezoid		vertex a	vertex b	vertex c	vertex d
Triangle		vertex a	vertex b	vertex c	
ZShape		start	end		

Slika 8 Parametri za konfiguraciju funkcije članstva

4.9. Factories

Factories se odnose na implementaciju Factory projektnog obrasca u kreiranju komponenti biblioteke. Ovaj obrazac centralizira instanciranje objekata, i uz pomoć registra novi objekti koji su eksterni za biblioteku mogu biti definisani i instancirani. Dodatna pogodnost korišćenja Factory projektnog obrasca je olakšano dodavanje i rukovanje novim objektima.

Implementirano je sedam Factory projektnih obrazaca i to su:

- ActivationFactory: koristi se za aktivaciju metode;
- DefuzzifierFactory: koristi se za tzv. defuzzifier-e;
- FunctionFactory: koristi se za nove funkcije i operatore;
- HedgeFactory: koristi se za tzv. hedge;
- SnormFactory i TnormFactory: koristi se za S-norme i T-norme (respektivno);
- TermFactory: koristi se za jezičke pojmove i odgovarajuće funkcije članstva.





















4.10. Uporedna analiza FuzzyLite biblioteke sa drugim bibliotekama za implementaciju kontrolera fazi logike

U radu Juan Rada-Vilela „The FuzzyLite Libraries for Fuzzy Logic Control“ [11] izvršena je uporedna analiza biblioteka za implementaciju kontrolera fazi logike: fuzzylite (6.0), jfuzzylite (6.0), matlab (R2016a), cfis (R2016a), octave (4.0.2) fuzzy logic toolkit (0.4.5) i jfuzzylogic (3.3). Analiza je bazirana na performansama, tačnosti i metirkama koje određuju kvalitet kreiranih kontrolera korišćenjem različitih biblioteka.

Performanse se odnose na analizu vremena neophodnog za izvršenje zadatih aktivnosti u toku procene biblioteka. Uporednom analizom dobijenih rezultata i razlike u performansama vrši se rangiranje biblioteka. Tačnost se odnosi na analizu numeričkih razlika između biblioteka

o dobijenim rezultatima. Na kraju, sveukupni kvalitet rangiranja biblioteka od boljih do lošijih zavisi od pokazatelja performansi, tačnosti, broja karakteristika i dokumentacije izvornog koda.

Dobijene performanse biblioteka u okviru istraživanja su prikazane na slici 9. Biblioteke sa najboljim performansama su cfis i matlab, zatim fuzzylite i jfuzzylite i na kraju jfuzzylogic i octave. Kako bi se što jednostavnije prikazale razlike u performansama ovih biblioteka u nastavku je dat uporedni prikaz karkterstika parova biblioteka.

benchmark		cfis	matlab	fuzzylite	jfuzzylite	jfuzzylogic	octave
Bell		0.116	0.162	0.595	1.114	1.628	955.379
Binary				0.170	0.193	2.501	
Concave				0.199	0.269	2.486	
Constant		0.006	0.007	0.023	0.012	0.073	388.558
Cosine				0.396	1.071	1.712	
Discrete				0.383	0.421	1.548	
Function				4.039	9.310	2.407	
Gaussian		0.116	0.163	0.329	0.841	1.174	941.648
GaussianP.		0.116	0.160	0.378	0.858	1.202	1324.361
Linear		0.005	0.007	0.025	0.013	0.067	387.853
PiShape		0.117	0.161	0.229	0.295	2.206	1094.229
Ramp				0.205	0.281	2.392	
Rectangle				0.181	0.219	2.207	
Sigmoid		0.116	0.161	0.313	0.891	1.245	942.106
SigmoidD.		0.117	0.167	0.445	1.562	2.400	1214.323
SigmoidP.		0.116	0.158	0.425	1.547	2.204	1111.997
Spike				0.321	0.836	2.198	
Trapezoid		0.117	0.166	0.208	0.261	0.294	1179.037
Triangle		0.116	0.166	0.200	0.250	0.312	1202.354
ZSShape		0.116	0.165	0.203	0.279	2.204	1074.589
common mean		0.092	0.129	0.279	0.645	0.933	948.402
overall mean		0.098	0.137	0.463	1.026	1.623	984.703
rank		1st	2nd	3rd	4th	5th	6th

Slika 9 Performanse biblioteka

Prvi par biblioteka čine fuzzylite i jfuzzylite biblioteke. Obe biblioteke imaju isti klasni dijagram, strukture podataka i tipove podataka, međutim biblioteka fuzzylite je brža u izvršavanju instrukcija u odnosu na biblioteku jfuzzylite. Većoj brzini fuzzylite biblioteke diprenelo je kompajliranje u izvršni kod, dok se u slučaju jfuzzylite biblioteke prevođenje vrši u bajt kod.

Sledeći par biblioteka čine biblioteke jfuzzylite i jfuzzylogic. Biblioteka jfuzzylite je brža u odnosu na biblioteku jfuzzylogic s obzirom na to da ona ima jednostavniji klasni dijagram, hijerarhiju i organizaciju komponenti čime je omogućena jednostavnija optimizacija performansi. Dodatno, proces tzv. defuzzication u slučaju Mamdani kontrolera u implementaciji jfuzzylogic biblioteke je složeniji, na šta je uticalo smeštanje izlaznih podataka u niz koji je inicijalizivan nulom.

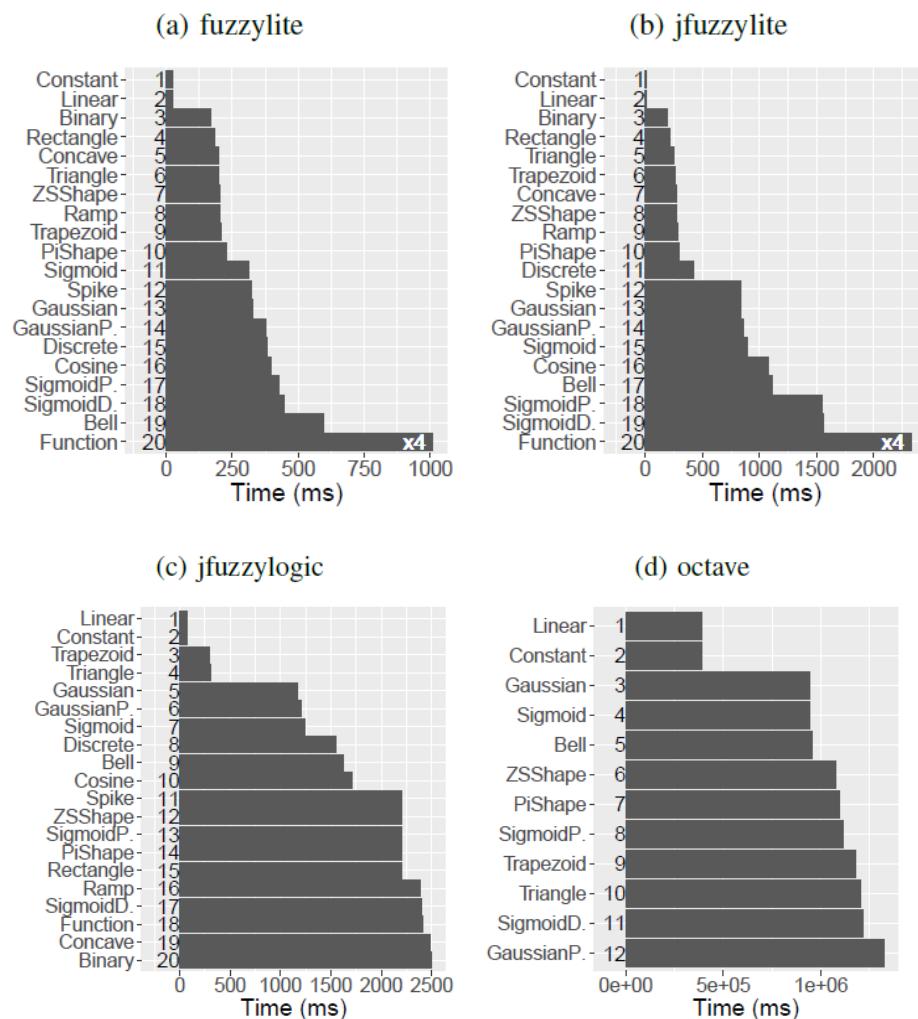
Upoređivanjem biblioteka fuzzylite i cfis ustanovljeno je da je biblioteka fuzzylite sporija u pogledu performansi. Objektno orijentisani način programiranja fuzzylite biblioteke doprineo je podeli kontrolera fazi logike na klase i strukture, za razliku od proceduralne implementacije kontrolera u okviru cfis biblioteke gde su podaci smešteni u metode. Dodatna karakteristika fuzzylite biblioteke je dinamička alokacija objekata koji odgovaraju

lingvističkim terminima, promenljivama, pravilima, operatorima i tzv. defuzzifier-ima. Ovakav način skladištenja podataka zahteva korišćenje heap memorije koja je sporija u odnosu na stek memoriju koja se koristi kod cfis biblioteke. Takođe, implementacije biblioteke fuzzylite su računski zahtevnije od implementacija biblioteke cfis.

Biblioteka fuzzylite je sporija u pogledu performansi u odnosu na biblioteku matlab. Matlab biblioteka koristi JIT kompajler koji joj je omogućio brže kompajliranje programa i dodatno ima mogućnost optimizovanog rada sa velikim matricama podataka. Ova biblioteka koristi i unapred kompajlirane binarne datoteke koje ostvaruju veliku uštedu u pogledu performansi izvršenja programa.

Sledeći par biblioteka čine biblioteke fuzzylite i octave. Upoređivanjem ovih biblioteka ustanovljeno je da je fuzzylite biblioteka brža s obzirom na to da se program odmah kompajlira u izvršni kod, dok je JIT kompajler još uvek u eksperimentnoj fazi kod octave biblioteke.

Rangiranje biblioteka prikazano je na slici 10. Izvršena je analiza vremena izvršenja operacija korišćenjem različitih funkcija članstva. Na osnovu dobijenih rezultata fuzzylite i jfuzzylite biblioteka su pokazale najbolje rezultate [12].



Slika 10 Rangiranje biblioteka

Ukupan kvalitet biblioteka određen je na osnovu performansi, tačnosti, karakteristika i dokumentacije izvornog koda biblioteka. U tabeli 2 su date sumarne statistike biblioteka cfis, matlab, fuzzylite, jfuzzylite, jfuzzylogic i octave u predstavljenim kategorijama. Na osnovu ovih rezultata biblioteka fuzzylite je na drugom mestu.

	Biblioteka	Kod	Komentari	Procentualno	Rang
1	octave	4067	5824	143.20	1
2	Fuzzylite	14949	8671	58.00	2
3	Jfuzzylite	12692	7344	57.86	3
4	Matlab	13657	3889	28.48	4
5	Jfuzzylogic	16718	4129	24.70	5
6	cfis	1992	312	15.66	6

Tabela 2 Uporedne karakteristike biblioteka za implementaciju kontrolera fazi logike

4.11. Instalacija FuzzyLite biblioteke

















Na zvaničnom sajtu FuzzyLite biblioteke [4] u odeljku Downloads dostupno je preuzimanje ove biblioteke za Qt okruženje, kao i za Windows 64-bit, Windows 32-bit, Linux 64-bit i Mac i Android operativne sisteme. Preuzimanjem i raspakivanjem odgovarajućeg paketa dobija se folder strukture prikazane na slici 11. U primeru prikazanom na slici 9 preuzeta je verzija FuzzyLite biblioteke za Windows 64-bit operativni sistem. Folder se sastoji od fajlova neophodnih za instalaciju biblioteke, kao i foldera sa dokumentacijom i primerima korišćenja biblioteke.

Name	Date modified	Type	Size
documentation	20/03/2017 11:06	File folder	
examples	20/03/2017 11:06	File folder	
fuzzylite	26/03/2017 17:00	File folder	
QtFuzzyLite	26/03/2017 17:05	File folder	
.dockerignore	20/03/2017 11:06	DOCKERIGNORE F...	2 KB
.gitignore	20/03/2017 11:06	Text Document	2 KB
.travis.yml	20/03/2017 11:06	YML File	7 KB
appveyor.yml	20/03/2017 23:01	YML File	1 KB
AUTHOR	20/03/2017 09:12	File	1 KB
CHANGELOG	26/03/2017 10:56	File	14 KB
COPYING	26/03/2017 11:42	File	32 KB
Dockerfile	20/03/2017 11:06	File	1 KB
Doxyfile	20/03/2017 11:06	File	105 KB
fuzzylite	20/03/2017 11:06	PNG File	257 KB
fuzzylite	20/03/2017 11:06	Microsoft Edge HT...	3 KB
INSTALL	20/03/2017 09:12	File	2 KB
LICENSE	26/03/2017 11:42	File	35 KB
LICENSE.FuzzyLite	26/03/2017 11:42	FUZZYLITE File	32 KB
NEWS	26/03/2017 10:56	File	3 KB
README.md	26/03/2017 10:56	MD File	19 KB
THANKS	26/03/2017 10:56	File	2 KB

Slika 11 Struktura foldera FuzzyLite biblioteke za Windows 64-bit operativni sistem

U okviru projekta ovog seminarskog rada korišćena je verzija FuzzyLite biblioteke namenjena Python programskom jeziku, pyfuzzylite. Instaliranje ove biblioteke je moguće izvršavanjem komande *pip install pyfuzzylite*. Nakon instalacije biblioteke kreiran je folder čija struktura je prikazana na slici 12. Pored direktorijuma sa primerima korišćenja pyfuzzylite

biblioteke u folderu se nalaze implementacije sledećih komponenti: activation, defuzzier, engine, exporter, factory, hedge, importer, library, norm, operation rule, term i variable.

Name	Date modified	Type	Size
 __pycache__	23/06/2021 18:14	File folder	
 examples	23/06/2021 18:14	File folder	
 __init__	23/06/2021 18:14	Python File	2 KB
 activation	23/06/2021 18:14	Python File	8 KB
 defuzzifier	23/06/2021 18:14	Python File	10 KB
 engine	23/06/2021 18:14	Python File	6 KB
 exporter	23/06/2021 18:14	Python File	23 KB
 factory	23/06/2021 18:14	Python File	13 KB
 hedge	23/06/2021 18:14	Python File	3 KB
 importer	23/06/2021 18:14	Python File	11 KB
 library	23/06/2021 18:14	Python File	7 KB
 norm	23/06/2021 18:14	Python File	4 KB
 operation	23/06/2021 18:14	Python File	7 KB
 rule	23/06/2021 18:14	Python File	23 KB
 term	23/06/2021 18:14	Python File	49 KB
 variable	23/06/2021 18:14	Python File	9 KB

Slika 12 Struktura foldera pyfuzzylite biblioteke

5. Primer implementacije kontrolera fazi logike korišćenjem pyfuzzylite biblioteke

U ovom radu izvršena je implementacija kontrolera fazi logike korišćenjem pyfuzzylite biblioteke u Python programskom jeziku [13]. Rezultati projekta su predstavljeni u obliku Jupyter notebook-a [14]. Kako bi se omogućilo korišćenje pyfuzzylite biblioteke izvršeno je njeno importovanje pre kreiranje kontrolera upotrebom import naredbe *import fuzzylite as fl*.

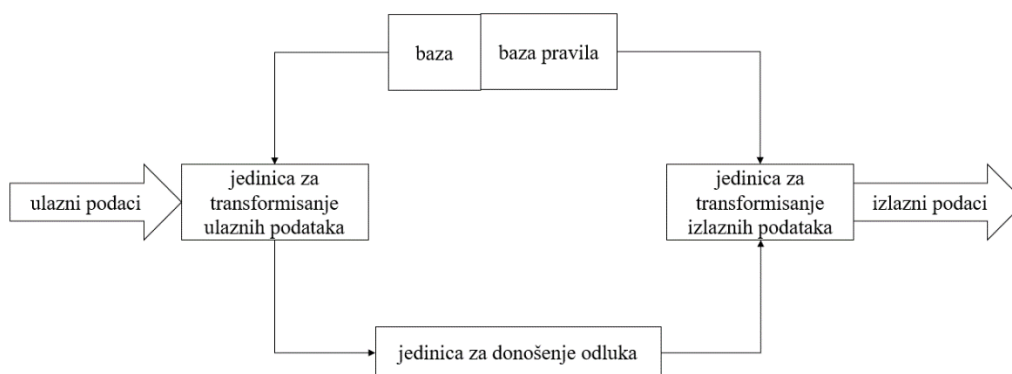
5.1. Sistem zaključivanja korišćenjem fazi logike (FIS)

Sistem zaključivanja korišćenjem fazi logike (eng. Fuzzy Inference System) je ključna jedinica sistema fazi logike koja ima ulogu u donošenju zaključaka na osnovu ulaznih podataka i skupa definisanih pravila. Sadrži pet funkcionalnih blokova:

- baza pravila: sastoji se od IF-THEN pravila;
- baza: definiše funkcije članstva fazi seta koje se koriste u pravilima;
- jedinica za donošenje odluka: realizuje operacije pravila;
- jedinica za transformisanje ulaznih podataka: transformiše ulazne podatke u fazi podatke;
- jedinica za transformisanje izlaznih podataka: transformiše dobijene rezultate.

Na slici 13 prikazan je blok dijagram FIS sistema. Dve najvažnije metode FIS-a su:

- Mamdani Fuzzy Inference System
- Takagi-Sugeno Fuzzy Model (TS Method).



Slika 13 Blok dijagram FIS sistema

5.2. Primer implementacije Mamdani kontrolera fazi logike

Pri implementaciji Mamdani kontrolera korišćen je primer sistema za preporuku. Posmatra se ukupna ocena proizvoda, ukoliko je ocena niska, proizvod se ne preporučuje kupcima, ukoliko je proizvod srednje ocenjen onda se on delimično preporučuje, dok ukoliko je proizvod ocenjen visokom ocenom uvek se preporučuje kupcima.

Definisanje kontrolera je prikazano na slici 14. Korišćenjem klase *Engine* biblioteke pyfuzzylite kreirana je instanca kontrolera *controller*, dodeljen mu je naziv „RecommenderSystem“ kao i opis „Product recommendation system“.


```

controller = fl.Engine(
    name="RecommenderSystem",
    description="Product recommendation system"
)

```

Slika 14 Definicija kontrolera

Ulazni podaci kontrolera određeni su klasom *input_variables* biblioteke *pyfuzzylite*. Naziv ulaznih podataka je „Review“. Moguće vrednosti ulaznih podataka su u opsegu od 0 do 1 (Slika 15).

```

controller.input_variables = [
    fl.InputVariable(
        name="Review",
        description="",
        enabled=True,
        minimum=0.000,
        maximum=1.000,
        lock_range=False,
        terms=[
            fl.Triangle("LOW", 0.000, 0.250, 0.500),
            fl.Triangle("AVERAGE", 0.250, 0.500, 0.750),
            fl.Triangle("HIGH", 0.500, 0.750, 1.000)
        ]
    )
]

```

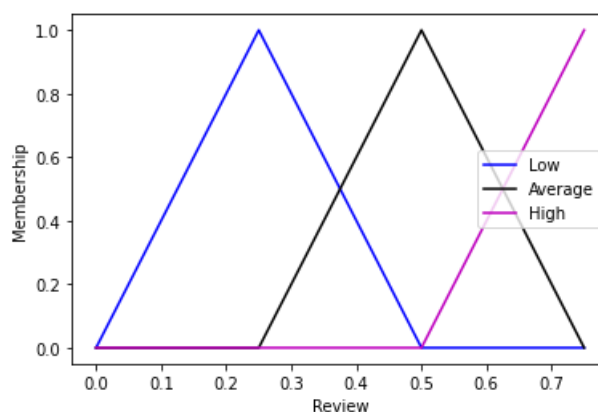
Slika 15 Definicija ulaznih podataka

Za definisanje termina iskorišćena je *triangle* funkcija članstva. Vrednosti ocene proizvoda su „LOW“, „AVERAGE“ i „HIGH“. Način izvršenja ove funkcije prikazan je na slici 16.

Vrednost „LOW“ definisana kao [0.0, 0.25, 0.5] označava da će članstvo biti 0% za 0.0 ocenu, linearno će porasti do 100% za 0.25 i opasti do 0% za ocenu 0.5.

Vrednost „AVERAGE“ definisana kao [0.25, 0.5, 0.75] označava da će članstvo biti 0% za 0.25 ocenu, linearno će porasti do 100% za 0.5 i opasti do 0% za ocenu 0.75.

Vrednost „HIGH“ definisana kao [0.5, 0.75, 1.0] označava da će članstvo biti 0% za 0.5 ocenu, linearno će porasti do 100% za 0.75 i opasti do 0% za ocenu 1.0.



Slika 16 Triangle funkcija članstva ulaznih podataka

Izlazni podaci kontrolera određeni su klasom *output_variables* biblioteke *pyfuzzylite*. Naziv ulaznih podataka je „Product“. Moguće vrednosti izlaznih podataka su u opsegu od 0 do 1 (Slika 17). Kao funkcija agregacije koristi se maksimum i tzv. defuzzier je baziran na centru.

```
controller.output_variables = [
    fl.OutputVariable(
        name="Product",
        description="",
        enabled=True,
        minimum=0.000,
        maximum=1.000,
        lock_range=False,
        aggregation=fl.Maximum(),
        defuzzifier=fl.Centroid(200),
        lock_previous=False,
        terms=[
            fl.Triangle("NOT_RECOMMENDED", 0.000, 0.250, 0.500),
            fl.Triangle("SOMEWHAT_RECOMMENDED", 0.250, 0.500, 0.750),
            fl.Triangle("RECOMMENDED", 0.500, 0.750, 1.000)
        ]
    )
]
```

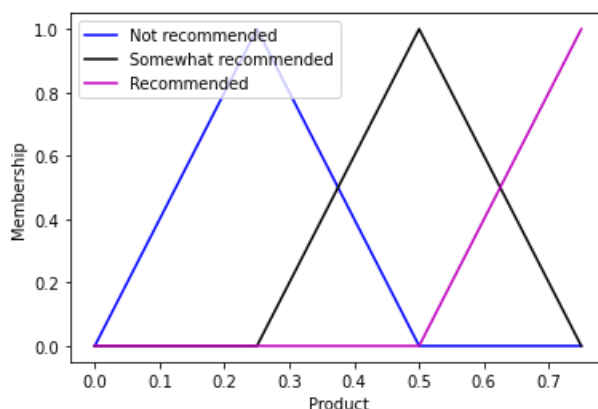
Slika 17 Definicija izlaznih podataka

Za definisanje termina iskorišćena je *triangle* funkcija članstva. Vrednosti ocene proizvoda su „NOT_RECOMMENDED“, „RECOMMENDED“ i „SOMEWHAT_RECOMMENDED“. Način izvršenja ove funkcije prikazan je na slici 18.

Vrednost „NOT_RECOMMENDED“ definisana kao [0.0, 0.25, 0.5] označava da će članstvo biti 0% za 0.0 preporuku, linearno će porasti do 100% za 0.25 i opasti do 0% za preporuku 0.5.

Vrednost „SOMEWHAT_RECOMMENDED“ definisana kao [0.25, 0.5, 0.75] označava da će članstvo biti 0% za 0.25 preporuku, linearno će porasti do 100% za 0.5 i opasti do 0% za preporuku 0.75.

Vrednost „RECOMMENDED“ definisana kao [0.5, 0.75, 1.0] označava da će članstvo biti 0% za 0.5 preporuku, linearno će porasti do 100% za 0.75 i opasti do 0% za preporuku 1.0.



Slika 18 Triangle funkcija članstva izlaznih podataka

Blok pravila se sastoji od sledećih tri pravila:

1. if Review is LOW then Product is NOT_RECOMMENDED
2. if Review is AVERAGE then Product is SOMEWHAT_RECOMMENDED
3. if Review is HIGH then Product is RECOMMENDED.

Operatori konjunktije, disjunktije nisu dozvoljeni, dok za implikaciju važi minimum. Aktivaciona funkcija je definisana sa General (Slika 19).

```
controller.rule_blocks = [  
    fl.RuleBlock(  
        name="",  
        description="",  
        enabled=True,  
        conjunction=None,  
        disjunction=None,  
        implication=fl.Minimum(),  
        activation=fl.General(),  
        rules=[  
            fl.Rule.create("if Review is LOW then Product is NOT_RECOMMENDED", engine),  
            fl.Rule.create("if Review is AVERAGE then Product is SOMEWHAT_RECOMMENDED", engine),  
            fl.Rule.create("if Review is HIGH then Product is RECOMMENDED", engine)  
        ]  
    )  
]
```

Slika 19 Blok pravila

Ukoliko se kao ulaz kontrolera navede ocena proizvoda 0.9, vrši se njena transformacija u adekvatnu fazi vrednost oznčenu kao „high“. Primenom skupa definisanih pravila, određuje se preporuka proizvoda. Vrednost „high“ odgovara pravilu broj 3: if Review is HIGH then Product is RECOMMENDED. Na osnovu ovog pravila određuje se izlaz kontrolera, a to je vrednost „recommended“ (Slika 20).

```
controller.input_variable("Review").fuzzy_value = 0.9  
  
controller.process()  
  
print(controller.output_variable("Product").fuzzy_value)  
RECOMMENDED
```

Slika 20 Primer korišćenja implementiranog kontrolera

5.3. Primer implementacije Takagi-Sugeno kontrolera fazi logike

Pri implementaciji Takagi-Sugeno kontrolera korišćen je primer sistema za navodnjavanje zemlje. Procenjuje se vlažnost zemlje i ukoliko je ona niska, jačina motora za navodnjavanje je visoka, ukoliko je vlažnost zemlje umerena onda je i jačina motora umerena, dok ukoliko je vlažnost zemlje visoka, jačina motora je slaba.

Definisanje kontrolera je prikazano na slici 21. Korišćenjem klase *Engine* biblioteke *pyfuzzylite* kreirana je instanca kontrolera *controllerT*, dodeljen mu je naziv „IrrigationSystem“.

```
controllerT = fl.Engine(
    name="IrrigationSystem",
    description=""
)
```

Slika 21 Definicija kontrolera

Ulazni podaci kontrolera određeni su klasom *input_variables* biblioteke *pyfuzzylite*. Naziv ulaznih podataka je „SoilMoisture“. Moguće vrednosti ulaznih podataka su u opsegu od 0 do 1 (Slika 22).

```
controllerT.input_variables = [
    fl.InputVariable(
        name="SoilMoisture",
        description="",
        enabled=True,
        minimum=0.000,
        maximum=1.000,
        lock_range=False,
        terms=[
            fl.Triangle("LOW", 0.000, 0.250, 0.500),
            fl.Triangle("MEDIUM", 0.250, 0.500, 0.750),
            fl.Triangle("HIGH", 0.500, 0.750, 1.000)
        ]
    )
]
```

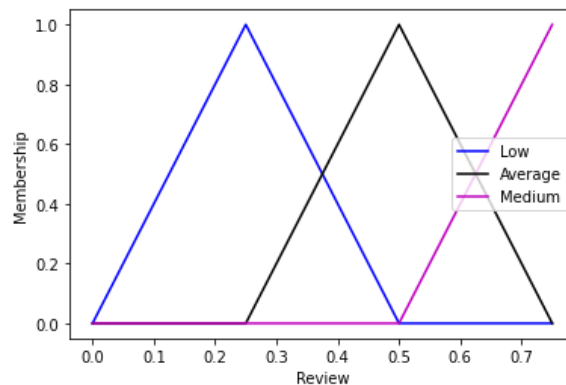
Slika 22 Definicija ulaznih podataka

Za definisanje termina iskorišćena je *triangle* funkcija članstva. Vrednosti vlažnosti zemlje su „LOW“, „MEDIUM“ i „HIGH“. Način izvršenja ove funkcije prikazan je na slici 23.

Vrednost „LOW“ definisana kao [0.0, 0.25, 0.5] označava da će članstvo biti 0% za 0.0 ocenu, linearno će porasti do 100% za 0.25 i opasti do 0% za ocenu 0.5.

Vrednost „MEDIUM“ definisana kao [0.25, 0.5, 0.75] označava da će članstvo biti 0% za 0.25 ocenu, linearno će porasti do 100% za 0.5 i opasti do 0% za ocenu 0.75.

Vrednost „HIGH“ definisana kao [0.5, 0.75, 1.0] označava da će članstvo biti 0% za 0.5 ocenu, linearno će porasti do 100% za 0.75 i opasti do 0% za ocenu 1.0.



Slika 23 Triangle funkcija članstva ulaznih podataka

Izlazni podaci kontrolera određeni su klasom *output_variables* biblioteke *pyfuzzylite*. Naziv ulaznih podataka je „MotorPower“. Moguće vrednosti izlaznih podataka su u opsegu od 0 do 1 (Slika 24). Kao funkcija članstva koristi se težinski prosek.

```
controllerT.output_variables = [  
    fl.OutputVariable(  
        name="MotorPower",  
        description="",  
        enabled=True,  
        minimum=0.000,  
        maximum=1.000,  
        lock_range=False,  
        aggregation=None,  
        defuzzifier=fl.WeightedAverage("TakagiSugeno"),  
        lock_previous=False,  
        terms=[  
            fl.Constant("LOW", 0.250),  
            fl.Constant("MEDIUM", 0.500),  
            fl.Constant("HIGH", 0.750)  
        ]  
    )  
]
```

Slika 24 Definicija izlaznih podataka

Za definisanje termina iskorišćena je *constant* funkcija članstva. Vrednosti jačine motora su „LOW“, „MEDIUM“ i „HIGH“. Vrednost „LOW“ definisana je sa 0.25, vrednost „MEDIUM“ sa 0.5 i vrednost „HIGH“ kao 0.75.

Blok pravila se sastoji od sledećih pravila:

1. if SoilMoisture is LOW then MotorPower is HIGH,
2. if SoilMoisture is MEDIUM then MotorPower is MEDIUM,
3. if SoilMoisture is HIGH then MotorPower is LOW.

Operatori konjukcije, disjunkcije i implikacije nisu dozvoljeni. Aktivaciona funkcija je definisana sa General (Slika 25).

```
controllerT.rule_blocks = [  
    fl.RuleBlock(  
        name="",  
        description="",  
        enabled=True,  
        conjunction=None,  
        disjunction=None,  
        implication=None,  
        activation=fl.General(),  
        rules=[  
            fl.Rule.create("if SoilMoisture is LOW then MotorPower is HIGH", controllerT),  
            fl.Rule.create("if SoilMoisture is MEDIUM then MotorPower is MEDIUM", controllerT),  
            fl.Rule.create("if SoilMoisture is HIGH then MotorPower is LOW", controllerT)  
        ]  
    )  
]
```

Slika 25 Blok pravila

Ukoliko se kao ulaz kontrolera navede vlažnost zemlje 0.1, vrši se transformacija ulazne vrednosti u adekvatnu fazi vrednost oznčenu kao „low“. Primenom skupa definisanih pravila, određuje se jačina motora za navodnjavanje. Vrednost „low“ odgovara pravilu broj 1: if SoilMoisture is LOW then MotorPower is HIGH. Na osnovu ovog pravila određuje se izlaz kontrolera, a to je vrednost „high“ (Slika 26).

```
controllerT.input_variable("SoilMoisture").fuzzy_value = 0.1
```

```
controllerT.process()
```

```
print(controllerT.output_variable("MotorPower").fuzzy_value)
```

HIGH

Slika 26 Primer korišćenja implementiranog kontrolera

6. Zaključak

Kontroleri fazi logike (FLC) su matematički modeli dizajnirani da vrše kontrolu sistema baziranih na fazi logici. Njihovom jednostavnom implementacijom, fleksibilnošću, interoperabilnošću u radu kao i lakom prilagođenošću novim problemima omogućen je širok spektar primene. Samo neke od primena su: su mašinsko učenje, kontrola dronova, autopilot, igrice i kontrola robota. Osnovne ideje o razvoju fazi kontrolera su se javile 1965. godine, dok danas ima više od 20 različitih softverskih biblioteka koje implementiraju način rada ovih kontrolera. Uzimajući u obzir različite primene kontrolera fazi logike svaka od biblioteka je korišćenjem drugačijih metoda implementacije izvršila modelovanje funkcija za kreiranje kontrolera čime je dostignut određeni stepen performansi, tačnosti i jednostavnosti korišćenja. Neke od najčešće korišćenih biblioteka su FuzzyLite biblioteke, Matlab, Octave i jFuzzyLogic gde biblioteka FuzzyLite ima najviše dostupnih karakteristika kontrolera fazi logike, dobro dokumentovan kod, visok nivo performansi izvršenja programa kao i veliku tačnost i preciznost u modelovanju.

7. Reference

- [1] Fuzzy Logic Toolbox. Design and simulate fuzzy logic systems. Dostupno na: <https://uk.mathworks.com/products/fuzzy-logic.html> [pristupljeno: 29.6.2021.]
- [2] Octave Fuzzy Logic Toolkit . Dostupno na: <https://octave.sourceforge.io/fuzzy-logic-toolkit/> [pristupljeno: 29.6.2021.]
- [3] jFuzzyLogic. Dostupno na: <http://jfuzzylogic.sourceforge.net/html/index.html> [pristupljeno: 29.6.2021.]
- [4] FuzzyLite. The FuzzyLite Libraries for Fuzzy Logic Control. Dostupno na: <https://www.fuzzylite.com/> [pristupljeno: 29.6.2021.]
- [5] MatLab. Dostupno na: <https://uk.mathworks.com/products/matlab.html> [pristupljeno: 29.6.2021.]
- [6] GNU Octave. Dostupno na: <https://www.gnu.org/software/octave/index> [pristupljeno: 29.6.2021.]
- [7] Juan Rada-Vilela, “The FuzzyLite Libraries for Fuzzy Logic Control”
- [8] O. N. Jensen, P. Mortensen, O. Vorm, and M. Matthias, “Automation of matrix-assisted laser desorption/ionization mass spectrometry using fuzzy logic feedback control.” *Analytical Chemistry*, vol. 69, no. 9, pp.1706–1714, 1997.
- [9] B. Center and B. P. Verma, “Fuzzy logic for biological and agricultural systems,” *Artificial Intelligence Review*, vol. 12, no. 1, pp. 213–225,1998.
- [10] J. Alcal´a-Fdez and J. M. Alonso, “A survey of fuzzy systems software: Taxonomy, current research trends, and prospects,” *IEEE Transactions on Fuzzy Systems*, vol. 24, no.1, pp. 40–56, 2016.
- [11] P. Cingolani and J. Alcal´a-Fdez, “jFuzzyLogic: a robust and flexible fuzzy-logic inference system language implementation,” in *Proceedings of the IEEE International Conference on Fuzzy Systems*, 2012, pp. 1–8.
- [12] “jFuzzyLogic: a java library to design fuzzy logic controllers according to the standard for fuzzy control programming,” *International Journal of Computational Intelligence Systems*, pp. 61–75, 2013.
- [13] Python. Dostupno na: <https://www.python.org/> [pristupljeno: 29.6.2021.]
- [14] Jupyter. Dostupno na: <https://jupyter.org/> [pristupljeno: 29.6.2021.]