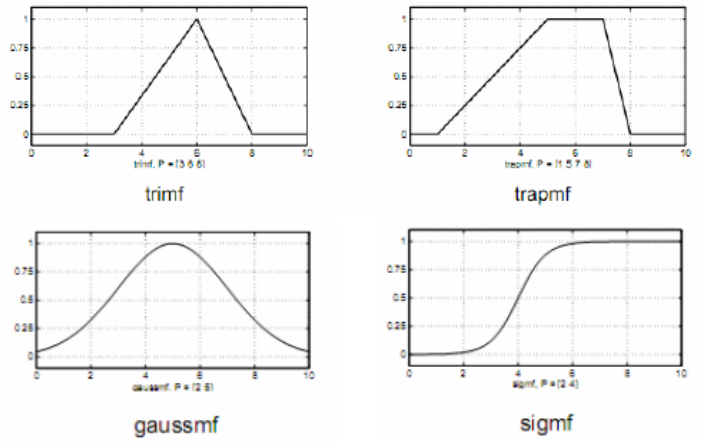
REALIZIRANJE REGULATORA TEMPERATURE POMOĆU NEIZRAZITIH SUSTAVA

1. Osnove neizrazitih sustava

Neizrazita logika (engl. Fuzzy logic) omogućuje donošenje odluka na temelju nepreciznih informacija prisutnih u stvarnom svijetu. Neizraziti sustav zaključivanja moguće je izgraditi na temelju podataka dobivenih od eksperta. Razvoj im započinje Lotfi Zadeh 1965., a usavršava Ebrahim H. Mamdani 1975. i Michio Sugeno 1985. Za razliku od neuronske mreže, proces zaključivanja je transparentan. Također je omogućeno kombiniranje s klasičnim sustavima upravljanja. Zasnovana je na prirodnom jeziku, što znači da je neizraziti sustav odlučivanja temeljen na pravilima razumljivim i jasnim čovjeku (npr. "Ako je temperatura visoka, smanji grijanje"). I pored svih prednosti neizrazite logike, nije ju preporučljivo koristiti tamo gdje postoje dobra i provjerena "klasična" rješenja.
Bitan dio neizrazite logike jeste neizraziti skup, koji se razlikuje od klasičnih skupova. Tako kod klasičnih skupova određeni element pripada nekom skupu, ili mu ne pripada (crno - bijelo), dok je kod neizrazitih skupova definirana pripadnost elementa skupu. Drugim riječima u neizrazitoj logici istinitost neke tvrdnje postaje stvar mjere! Ta mjera odnosno pripadnost definirana je kao broj, najčešće u intervalu [0, 1]. Neizrazit skup se može prikazati izrazom:

$$A = \{(\mu_A(x), x) \mid x \in X\},$$

gdje je μ_A funkcija pripadnosti, a X neki skup.
Funkcija pripadnosti (engl. membership function) definira pripadnost skupu. Postoji mnogo oblika funkcije pripadnosti (npr. trokut, trapez, gauss, zvono, itd.). Najčešće je jedini uvjet za funkciju pripadnosti da poprima vrijednosti u intervalu [0, 1].



Slika 1. Najčešće funkcije pripadnosti

Kao i kod klasične logike i u neizrazitoj logici postoje operatori I, ILI, NE. Međutim, u odnosu na klasičnu logiku ovi operatori nisu jednoznačno određeni, već su prošireni na cijeli interval [0, 1], time da na granicama prelaze u operatore klasične logike odnosno u tzv. Boolovu logiku. U neizrazitoj logici se za pojedine operatore koriste razne funkcije.
Tako se za I operator koriste funkcije:

- y = min(a, b)

- y = prod(a, b) = a * b

Za operator ILI se koriste funkcije:

- y = max(a, b)
- y = probor(a, b) = a + b – a * b

Za operator NE se koristi funkcija:

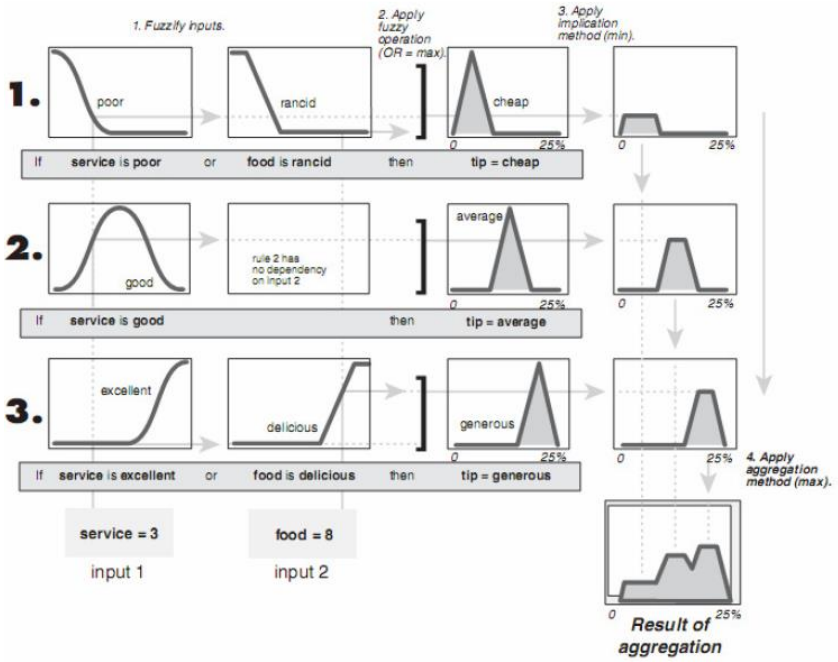
- y = 1 – a

Baza neizrazitih pravila (engl. fuzzy rule base) predstavlja središnji dio neizrazitog sustava. Neizrazito pravilo ima standardni *if…then* oblik:

***ako** je temperatura visoka **onda** smanji grijanje,*

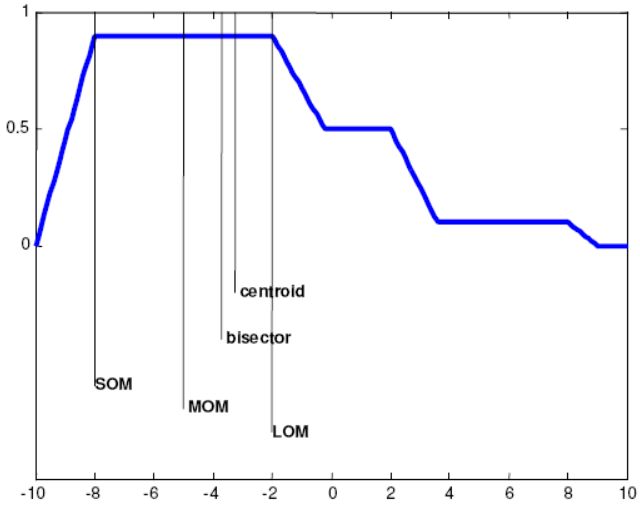
gdje je „***ako** je temperatura visoka“* polazni dio pravila, a „***onda** smanji grijanje“* posljedični dio pravila. Pravila povezuju područja vrijednosti ulaznih i izlaznih varijabli.
Neizrazito zaključivanje se sastoji od 5 koraka kao što se vidi na slici 2, a ti koraci su slijedeći:

1. Fuzifikacija ulaza
2. Primjena neizrazitih operatora (pretpostavka pravila)
3. Implikacija (zaključak pravila)
4. Agregacija izlaza
5. Defuzifikacija



Slika 2. Koraci rada neizrazitog sustava

Kao što se vidi na slici, za svako pravilo određuje se pripadnost ulazne varijable odgovarajućem skupu pomoću funkcija pripadnosti. Taj postupak se naziva fuzifikacija (engl. fuzzification) ulaza. Zatim se primjenjuje neizrazita pravila i izlaz se prenosi na odgovarajuću izlaznu funkciju pripadnosti (primjena neizrazitih operatora). Implikacijom se određuje utjecaj pravila na izlaznu varijablu odnosno izlaznu funkciju pripadnosti. Izlaz neizrazitog sustava je određen svim pravilima, tako da je potrebno združiti rezultate svih pravila odnosno pojedinačnih neizrazitih skupova u jedan jedinstveni izlazni neizraziti skup. Operator združivanja (agregacije) mora biti komutativan, tj. izlaz ne smije ovisiti o redoslijedu evaluacije pravila (tako se najčešće koriste operatori max, probor i sum). **Izlaz neizrazitog sustava je neizraziti skup**. U većini primjena neizrazitih sustava (automatsko upravljanje, ekspertni sustavi, itd.) izlaz treba biti jedna numerička vrijednost. Stoga se provodi defuzifikacija. Postoji više metoda defuzifikacije, dok najčešće korištene vidimo na slici 3, a to su: Centroid, Bisector, SOM (najmanja vrijednost za koju izlazna funkcija ima maksimum, engl. Smallest Of Maximum), MOM (srednja vrijednost za koju izlazna funkcija ima maksimum, engl. Middle Of Maximum), i LOM (najveća vrijednosti za koju izlazna vrijednost ima maksimum, engl. Largest Of Maximum).



Slika 3. Metode defuzifikacije neizrazitog sustava

Navedene metode se koriste kod Mamdani neizrazitih sustava.

Postoje dva tipa neizrazitih sustava: Takagi-Sugeno-Kang (Sugeno) i Mamdani. Sve prethodno rečeno odnosilo se na Mamdani neizraziti sustav. Razlika između Sugeno i Mamdani sustava je u izlaznim funkcijama pripadnosti, koje su kod Sugeno sustava polinomi (u pravilu nultog ili prvog reda). Tako primjerice pravila imaju sljedeći oblik:
If *Input1 = x* ***and*** *Input2 = y,* ***then*** *Output* ***is*** $z = ax + by + c$ (za Sugeno sustav nultog reda vrijedi a = b = 0).
Prema tome vrijednost izlazne varijable je uvijek broj, a ne neizraziti skup (kao što je slučaj kod Mamdani sustava). Izlaz se određuje kao težinska srednja vrijednost izlaza pravila:

$$Izlaz = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i},$$

gdje je w_i izlaz i-tog pravila, a z_i vrijednost i-te funkcije pripadnosti. Kod Sugeno sustava prvog reda izlazi pravila određuju položaj i visinu izlazne vrijednosti, dok kod sustava nultog reda samo visinu (položaj je konstantan). Sugeno sustav ima nekoliko prednosti nad Mamdani sustavom: zbog odsutnost

defuzifikacije računski je manje zahtjevan, prikladan je za sustave automatskog upravljanja, omogućuje primjenu optimizacijskih i adaptivnih metoda (genetski algoritmi, neuronske mreže, itd.), jamči neprekidnost izlazne plohe, jednostavniji je za analizu.

Uporaba neizrazitih sustava danas ima implementacije u svim granama tehničkih znanosti, počevši sa jednostavnim upravljačkim sustavima pa do ekspertnih sustava i sofisticiranih umjetnih inteligencija.

2. Implementacija neizrazitih sustava u scikit-fuzzy biblioteci

Scikit-fuzzy biblioteka služi za stvaranje i rad s neizrazitim sustavima. Definira standardne elemente potrebne za rad s neizrazitim sustavima kao što su ulazne i izlazne varijable s pripadajućim neizrazitim skupovima, razne funkcije pripadnosti, te pravila koje povezuju ulazne varijable s izlaznim.
Za rad s ovom bibliotekom se tipično na početku programa uključuje sljedeće:

```
import skfuzzy as fuzz
from skfuzzy import control as ctrl
```

Također pošto se biblioteka znatno oslanja na rad ***numpy*** biblioteke, te prikaz pomoću ***matplotlib*** biblioteke, preporučeno je uključivanje i njih u skriptu.
Na početku rada se uvijek definiraju ulazne i izlazne varijable, npr.:

```
#Ulazna varijabla regulatora
pogreska = ctrl.Antecedent(np.linspace(0, 30, 100), 'pogreska')
#Izlazna varijabla regulatora
grijac = ctrl.Consequent(np.linspace(0, 10, 100), 'grijac', defuzzify_method='centroid')
```

Prilikom definiranja ulaznih i izlaznih varijabli potrebno je definirati tzv. „universe“ koji definira prostor vrijednosti za svaku varijablu, npr. pogreška u °C je definirana u intervalu od 0 do 30. Za izlaznu varijablu je još moguće odabrati metodu defuzifikacije ('centroid', 'bisector', 'mom', 'som' i 'lom'), pri čemu je 'centroid' predefinirana vrijednost.
Nakon toga je potrebno definirati neizrazite skupove i funkcije pripadnosti. Implementirane su funkcije pripadnosti sljedećih tipova: *dsigmf, gaussmf, gauss2mf, gbellmf, piecemf, pimf, psigmf, sigmf, smf, trapmf, trimf, zmf*. Primjer definiranja neizrazitog skupa:

```
pogreska['mala'] = fuzz.trapmf(pogreska.universe, [0, 0, 5, 15]) #Trapez, lomi se u 4 točke
grijac['slab'] = fuzz.gaussmf(grijac.universe, 0, 2.5) #Gaussova funkcija definirana s mi i sigma
```

Vizualizacija definiranih neizrazitih skupova za pojedinu varijablu je moguća pomoću *view()* metode, npr.:

```
pogreska.view()
grijac.view()
```

Na kraju je potrebno definirati pravila koja povezuje ulazne varijable (i pridružene skupove) s izlaznim varijablama (i pridruženim skupovima). Pravila se definiraju pomoću ***ctrl.Rule*** klase, npr.:

rule1 = ctrl.Rule(pogreska['mala'], grijac['slab'])

Smisao napisanog pravila je „ako je pogreška mala onda je grijač slab“. Za definiranje pravila se mogu potencijalno koristiti više neizrazitih skupova koji mogu biti povezani s operatorima '|' (ILI) i '&' (I). Kompletan kontrolni sustav s npr. 3 pravila se tada definira na slijedeći način:

reg = ctrl.ControlSystem([rule1, rule2, rule3])

Za simuliranje ovakvog kontrolnog sustava se koristi ***ControlSystemSimulation*** klasa npr.:

reg_sim = ctrl.ControlSystemSimulation(reg)

Provjera vrijednosti izlaza za neku vrijednost ulaza je moguća na sljedeći način:

```
reg_sim.input['pogreska'] = 17
reg_sim.compute()
output = reg_sim.output['grijac']
```

U *output* varijabli se potom nalazi izlaz iz neizrazitog sustava ako se na ulaz dovede navedena vrijednost.

3. Zadatak

Potrebno je izraditi neizrazite regulatore temperature toplinskog procesa. Proces sadrži senzor temperature i grijač koji je moguće kontinuirano regulirati. Koristeći Python programski jezik i scikit-fuzzy biblioteku realizirati neizraziti regulator temperature sljedećih svojstava:

Tip regulatora: Mamdani

Ulazna varijabla: Greska_temp - pogreška temperature (razlika između zadane i trenutne temperature), u intervalu [0,30] ^{o}C, neizraziti skupovi = Niska, Srednja, Visoka

Izlazna varijabla: Grijac - snaga grijača u intervalu [0,10], neizraziti skupovi = Slabo, Srednje, Jako

Pravila:

If Greska_temp is Niska then Grijac is Slabo
If Greska_temp is Srednja then Grijac is Srednje
If Greska_temp is Visoka then Grijac is Jako

Potrebno je realizirati tri različita regulatora temperature korištenjem trokutastih, trapezastih i gausovih funkcija pripadnosti. Parametre neizrazitih skupova na ulazu i izlazu podesite samostalno

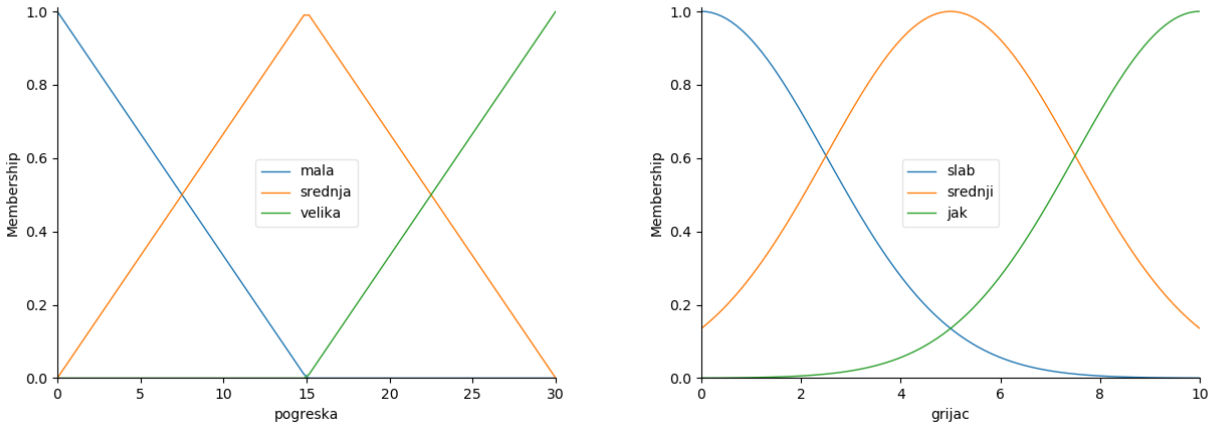
Rješenje je potrebno pokazati na vježbama, te predati kao pisano izvješće koje treba sadržavati:

- opis problema i mogućnost korištenja neizrazitog regulatora za njegovo rješavanje,

- grafički prikaz funkcija pripadnosti i prijenosne funkcije za svaki regulator,
- opisati svojstva realiziranih regulatora.

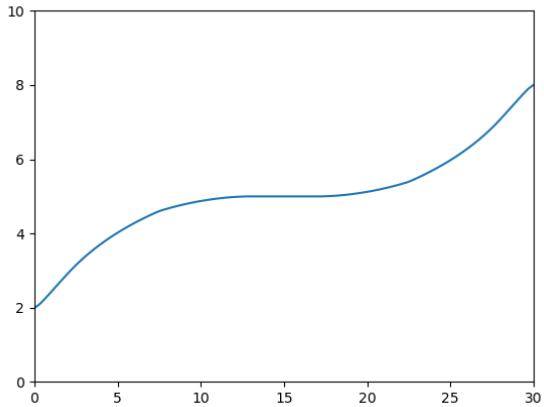
4. Upute

Pošto je traženi kod u Python-u za navedeni zadatak jednostavan potrebno je kompletnu skriptu napisati samostalno. Kao pomoć prilikom pisanja programa koristite opis ***scikit-fuzzy*** biblioteke koji se nalazi u 2. poglavlju ovog predloška. Također možete koristiti i službenu dokumentaciju koja je dostupna OVDJE. Kada realizirate regulator, izgled neizrazitih skupova možete dobiti pomoću *view()* metode kako je to opisano u poglavlju 2. Izgled ulaznih i izlaznih neizrazitih skupova pomoću navedene metode može biti sličnima prikazanima na sljedećoj slici.



Slika 4. Prikaz definiranih neizrazitih skupova ulazne i izlazne valrijable dobivene pomoću *view()* metode

Za dobivanje prijenosne funkcije realiziranog regulatora potrebno je ispitati rad regulatora koristeći ***ControlSystemSimulation*** klasu na način kako je to opisano u 2. poglavlju. Rad regulatora je potrebno ispitati nad cijelim intervalom ulazne varijable (0→30) u najmanje 100 točaka. Za dobivanje vrijednosti za ispitivanje regulatora preporučam koristiti *linspace* funkciju implementiranu u ***numpy*** biblioteci, dok za vizualizaciju preporučam korištenje ***matplotlib*** biblioteke. Sve navedene i korištene biblioteke su vam predinstalirane u dostupnom razvojnom okruženju. Primjer izgleda prijenosne funkcije za neizrazite skupove prikazane na Slici 4 se može vidjeti na sljedećoj slici.



Slika 5. Primjer izgleda prijenosne funkcije neizrazitog regulatora