SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

REALIZIRANJE REGULATORA TEMPERATURE POMOĆU NEIZRAZITIH SUSTAVA

Meko računarstvo

Laboratorijska vježba 7

Ivan Gudelj

Diplomski studij računarstva, DRB

Osijek, 2022.

1. UVOD

Cilj sedme laboratorijske vježbe je upoznati se s neizrazitom logikom te izraditi neizrazite regulatore temperature toplinskog procesa. Taj proces sadrži senzor temperature kao i grijač koji je moguće kontinuirano regulirati.

Za izradu modela regulatora koristili smo Python programski jezik i ***scikit-fuzzy*** biblioteku koja služi za stvaranje i rad s neizrazitim sustavima. Ona 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 koja povezuju ulazne varijable s izlaznim.

2. OPIS PROBLEMA I RJEŠENJA

2.1. 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].

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.ž

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, a ti koraci su slijedeći:

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

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).

2.2. Problem regulatora

Potrebno je izraditi neizrazite regulatore temperature toplinskog procesa. Proces sadrži senzor temperature i grijač koji je moguće kontinuirano regulirati. Koristeći Python programski jezik i scikitfuzzy 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] oC, neizraziti skupovi = Niska, Srednja, Visoka

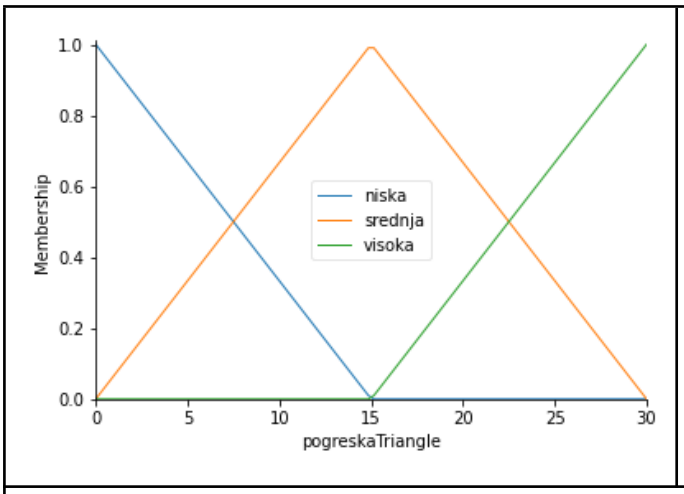
Izlazna varijabla: Grijac - snaga grijača u intervalu [0,10], neizraziti skupovi = Slabo, Srednje, Jako

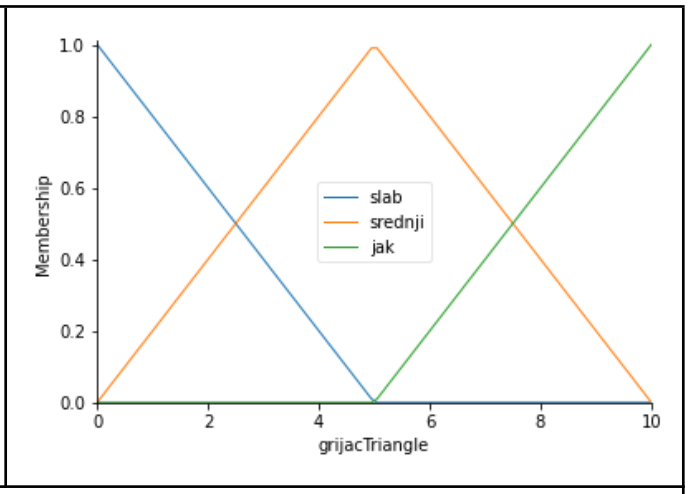
Pravila: If Greska_temp is Niska then Grijac is Slabo If Greska_temp is Srednja then Grijac is Srednje If Greska_temp is Visoka then Grijac is Jako

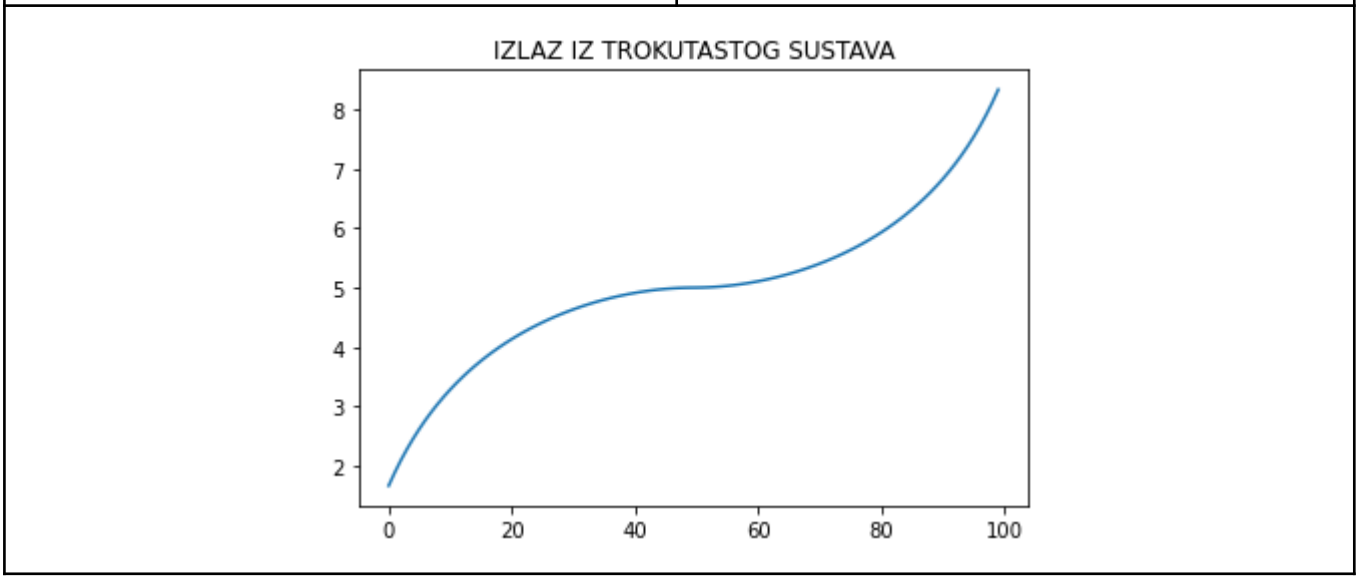
Potrebno je realizirati tri različita regulatora temperature korištenjem trokutastih, trapezastih i gausovih funkcija pripadnosti. Parametre neizrazitih skupova na ulazu i izlazu podesite samostalno.

3. REZULTATI ZA TRAŽENI PROBLEM - Izrada regulatora temperature toplinskog procesa

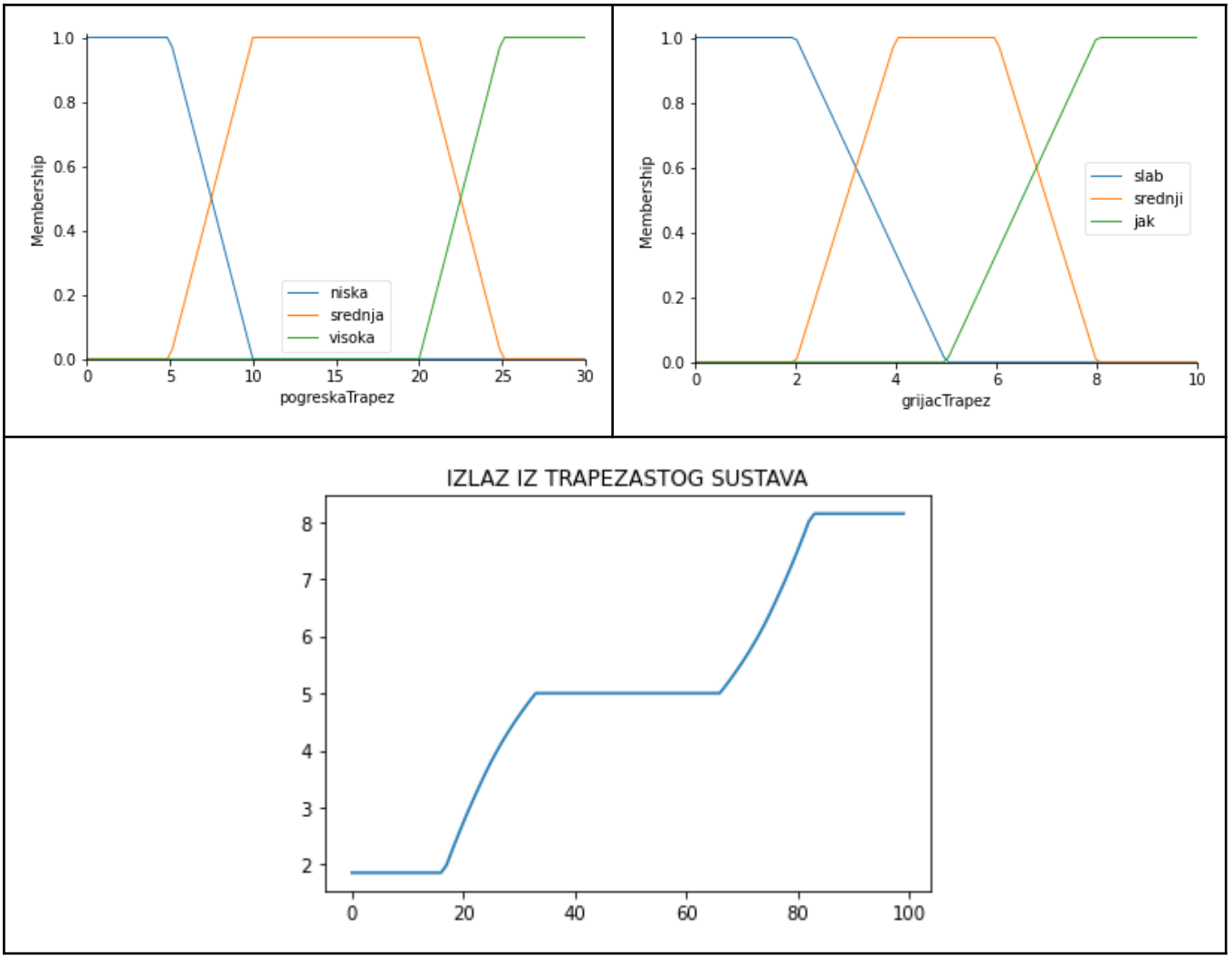
3.1. Trokutasta funkcija pripadnosti



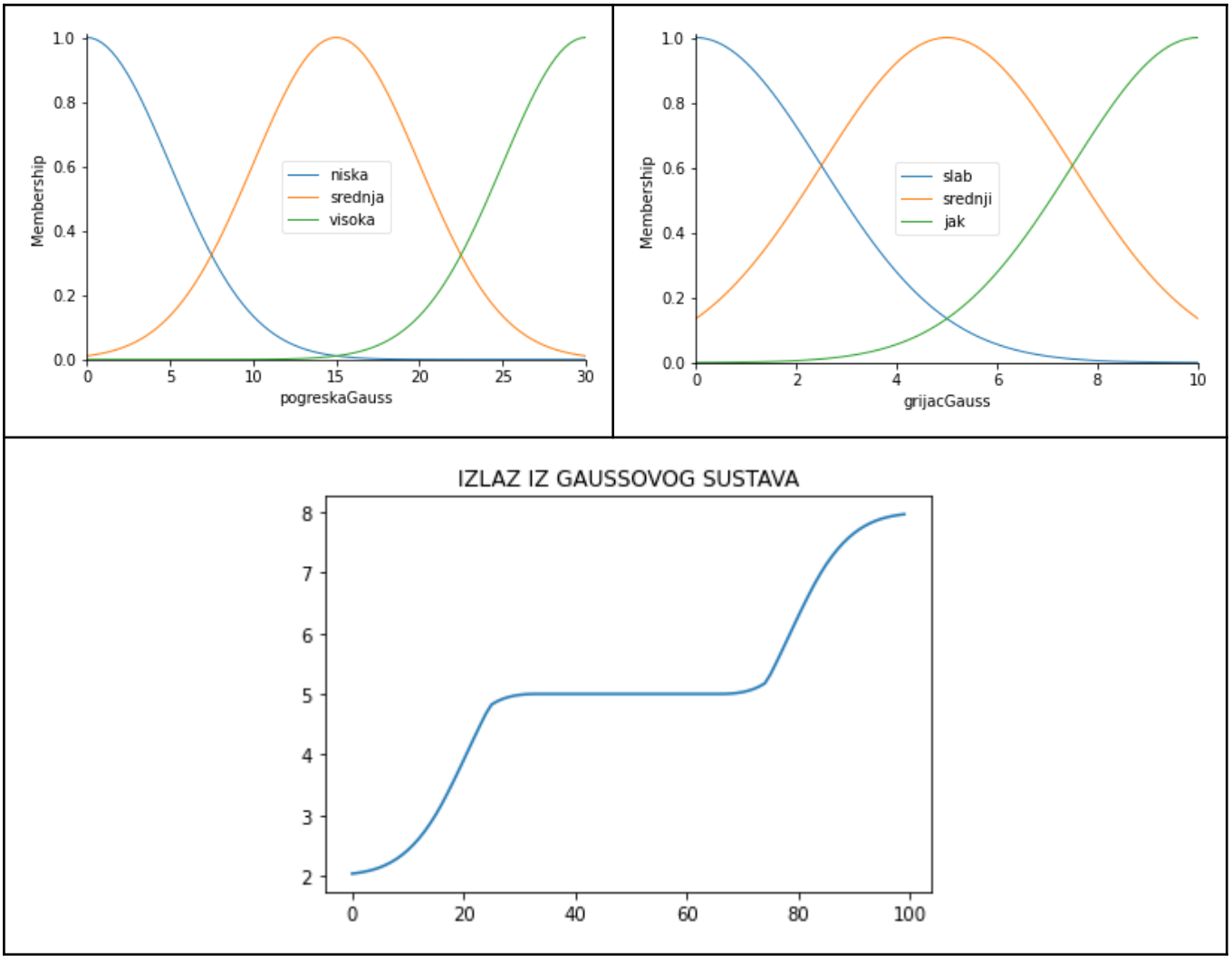




3.2. Trapezasta funkcija pripadnosti



3.3. Gaussova funkcija pripadnosti4



3.4. Programska podrška

```
pogreskaTriangle = ctrl.Antecedent(np.linspace(0, 30, 100), 'pogreskaTriangle')
pogreskaTriangle['niska'] = fuzz.trimf(pogreskaTriangle.universe, [0,0,15])
pogreskaTriangle['srednja'] = fuzz.trimf(pogreskaTriangle.universe, [0,15,30])
pogreskaTriangle['visoka'] = fuzz.trimf(pogreskaTriangle.universe, [15,30,30])
pogreskaTriangle.view()
grijacTriangle = ctrl.Consequent(np.linspace(0, 10, 100), 'grijacTriangle', defuzzify_method='centroid')
grijacTriangle['slab'] = fuzz.trimf(grijacTriangle.universe, [0,0,5])
grijacTriangle['srednji'] = fuzz.trimf(grijacTriangle.universe, [0,5,10])
grijacTriangle['jak'] = fuzz.trimf(grijacTriangle.universe, [5,10,10])
grijacTriangle.view()
triangleRule1 = ctrl.Rule(pogreskaTriangle['niska'],grijacTriangle['slab'])
triangleRule2 = ctrl.Rule(pogreskaTriangle['srednja'],grijacTriangle['srednji'])
triangleRule3 = ctrl.Rule(pogreskaTriangle['visoka'],grijacTriangle['jak'])
trianReg = ctrl.ControlSystem([triangleRule1,triangleRule2,triangleRule3])
######
trianReg_sim = ctrl.ControlSystemSimulation(trianReg)
trianReg_sim.input['pogreskaTriangle'] = np.linspace(0,30,100)
trianReg_sim.compute()
outputTriangle = trianReg_sim.output['grijacTriangle']
plt.figure()
plt.plot(outputTriangle)
plt.title("IZLAZ IZ TROKUTASTOG SUSTAVA")
plt.show()
```

```
pogreskaTrapez = ctrl.Antecedent(np.linspace(0, 30, 100), 'pogreskaTrapez')
pogreskaTrapez['niska'] = fuzz.trapmf(pogreskaTrapez.universe, [0,0,5,10])
pogreskaTrapez['srednja'] = fuzz.trapmf(pogreskaTrapez.universe, [5,10,20,25])
pogreskaTrapez['visoka'] = fuzz.trapmf(pogreskaTrapez.universe, [20,25,30,30])
pogreskaTrapez.view()
grijacTrapez = ctrl.Consequent(np.linspace(0, 10, 100), 'grijacTrapez', defuzzify_method='centroid')
grijacTrapez['slab'] = fuzz.trapmf(grijacTrapez.universe,[0,0,2,5])
grijacTrapez['srednji'] = fuzz.trapmf(grijacTrapez.universe,[2,4,6,8])
grijacTrapez['jak'] = fuzz.trapmf(grijacTrapez.universe,[5,8,10,10])
grijacTrapez.view()
trapezRule1 = ctrl.Rule(pogreskaTrapez['niska'],grijacTrapez['slab'])
trapezRule2 = ctrl.Rule(pogreskaTrapez['srednja'],grijacTrapez['srednji'])
trapezRule3 = ctrl.Rule(pogreskaTrapez['visoka'],grijacTrapez['jak'])
trapezReg = ctrl.ControlSystem([trapezRule1,trapezRule2,trapezRule3])
#######
trapezReg_sim = ctrl.ControlSystemSimulation(trapezReg)
trapezReg_sim.input['pogreskaTrapez'] = np.linspace(0,30,100)
trapezReg_sim.compute()
outputTrapez = trapezReg_sim.output['grijacTrapez']
plt.figure()
plt.plot(outputTrapez)
plt.title("IZLAZ IZ TRAPEZASTOG SUSTAVA")
plt.show()
```

```
pogreskaGauss = ctrl.Antecedent(np.linspace(0, 30, 100), 'pogreskaGauss')
pogreskaGauss['niska'] = fuzz.gaussmf(pogreskaGauss.universe, 0,5)
pogreskaGauss['srednja'] = fuzz.gaussmf(pogreskaGauss.universe, 15,5)
pogreskaGauss['visoka'] = fuzz.gaussmf(pogreskaGauss.universe, 30,5)
pogreskaGauss.view()
grijacGauss = ctrl.Consequent(np.linspace(0, 10, 100), 'grijacGauss', defuzzify_method='centroid')
grijacGauss['slab'] = fuzz.gaussmf(grijacGauss.universe, 0,2.5)
grijacGauss['srednji'] = fuzz.gaussmf(grijacGauss.universe, 5,2.5)
grijacGauss['jak'] = fuzz.gaussmf(grijacGauss.universe, 10,2.5)
grijacGauss.view()
gaussRule1 = ctrl.Rule(pogreskaGauss['niska'],grijacGauss['slab'])
gaussRule2 = ctrl.Rule(pogreskaGauss['srednja'],grijacGauss['srednji'])
gaussRule3 = ctrl.Rule(pogreskaGauss['visoka'],grijacGauss['jak'])
gaussReg = ctrl.ControlSystem([gaussRule1,gaussRule2,gaussRule3])
######
gaussReg_sim = ctrl.ControlSystemSimulation(gaussReg)
gaussReg_sim.input['pogreskaGauss'] = np.linspace(0,30,100)
gaussReg_sim.compute()
outputGauss = gaussReg_sim.output['grijacGauss']
plt.figure()
plt.plot(outputGauss)
plt.title("IZLAZ IZ GAUSSOVOG SUSTAVA")
plt.show()
```

4. ZAKLJUČAK

Za razliku od neuronske mreže, proces zaključivanja kod neizrazite logike je transparentan. Kod klasičnih skupova određeni element pripada skupu ili mu uopće ne pripada. **Primjer: Da li je petak dan vikenda?** Kod neizrazitih skupova definiramo pripadnost elementa skupu, koja je broj u intervalu [0,1]. **Primjeri: Skup visokih ljudi; Osjet temperature.**

Moguće je proširenje neizrazitog sustava s neuronskom mrežom budući je jedan od nedostataka nemogućnost automatskog podešavanja na temelju raspoloživih podataka. ANFIS (Adaptive Neuro-Fuzzy Inference System) kombinira transparentnost zaključivanja neizrazitog sustava s mogućnošću učenja neuronske mreže. Prema ulaznim podacima ANFIS podešava parametre funkcija pripadnosti kako bi izlaz sustava za podatke za učenje imao što manju grešku. Početni neizraziti sustav moguće je stvoriti automatski, ili je moguće podesiti postojeći sustav u kojem su ugrađene raspoložive informacije o problemu.