Bulanık Mantık ile İleri Kinematik: End-Effektörün Hedef Konuma Ulaşması

May 21, 2025

1 Giriş

Bu çalışmada, iki mafsallı bir robot kolunun uç efektörünün (*end-effector*), belirli bir hedef konuma ($x = 0, y = 2$) ulaşabilmesi için gerekli mafsal açılarını (q_1, q_2), bulanık mantık (*fuzzy logic*) yaklaşımı kullanarak hesaplamak amaçlanmıştır. Robot kolunun her iki bağlantısının uzunlukları sırasıyla $a_1 = 3$ ve $a_2 = 1$ birimdir.

2 Problemin Tanımı

Verilenler:

- Bağlantı 1 uzunluğu: $a_1 = 3$
- Bağlantı 2 uzunluğu: $a_2 = 1$
- Hedef konum: $(x, y) = (0, 2)$

Amaç, uç efektörün bu hedef konuma ulaşması için gereken eklem açılarını bulanık mantık yaklaşımıyla bulmaktır.

3 Yöntem

Çalışmada, iki serbestlik dereceli bir robot kolunun ileri ve ters kinematiği ele alınmıştır. Ters kinematik çözümü için klasik cebirsel yöntemler yerine bulanık mantık sistemi kullanılmıştır. Aşağıda, problemin çözümüne ilişkin temel adımlar ve kullanılan Python kodları sunulmuştur.

4 Kodlar ve Açıklamalar

4.1 Gerekli Kütüphanelerin Eklenmesi

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

Listing 1: Kütüphanelerin Eklenmesi

Bu bölümde, matematiksel işlemler için `numpy` ve bulanık mantık işlemleri için `skfuzzy` kütüphaneleri import edilmiştir.

4.2 Giriş ve Çıkış Değişkenlerinin Tanımlanması

```
x = ctrl.Antecedent(np.arange(-4, 4.1, 0.1), 'x')
y = ctrl.Antecedent(np.arange(-4, 4.1, 0.1), 'y')
q1 = ctrl.Consequent(np.arange(-180, 181, 1), 'q1')
q2 = ctrl.Consequent(np.arange(-180, 181, 1), 'q2')
```

Listing 2: Bulanık Değişkenlerin Tanımlanması

Bu kısımda, robotun uç efektörünün ulaşacağı x ve y değerleri ile, eklem açıları olan q_1 ve q_2 bulanık değişkenler olarak tanımlanmıştır.

4.3 Üyelik Fonksiyonlarının Oluşturulması

```
x['neg'] = fuzz.trimf(x.universe, [-4, -4, 0])
x['zero'] = fuzz.trimf(x.universe, [-1, 0, 1])
x['pos'] = fuzz.trimf(x.universe, [0, 4, 4])

y['neg'] = fuzz.trimf(y.universe, [-4, -4, 0])
y['zero'] = fuzz.trimf(y.universe, [-1, 0, 1])
y['pos'] = fuzz.trimf(y.universe, [0, 4, 4])

q1['neg'] = fuzz.trimf(q1.universe, [-180, -180, 0])
q1['zero'] = fuzz.trimf(q1.universe, [-30, 0, 30])
q1['pos'] = fuzz.trimf(q1.universe, [0, 180, 180])

q2['neg'] = fuzz.trimf(q2.universe, [-180, -180, 0])
q2['zero'] = fuzz.trimf(q2.universe, [-30, 0, 30])
q2['pos'] = fuzz.trimf(q2.universe, [0, 180, 180])
```

Listing 3: Üyelik Fonksiyonları

Her değişken için negatif, sıfır ve pozitif bölgeler olmak üzere üçgen üyelik fonksiyonları tanımlanmıştır.

4.4 Kural Tabanının Oluşturulması

```
rule1 = ctrl.Rule(x['zero'] & y['pos'], (q1['zero'], q2['zero']))
rule2 = ctrl.Rule(x['neg'] & y['pos'], (q1['pos'], q2['zero']))
rule3 = ctrl.Rule(x['pos'] & y['pos'], (q1['neg'], q2['zero']))
rule4 = ctrl.Rule(x['zero'] & y['neg'], (q1['zero'], q2['pos']))
rule5 = ctrl.Rule(x['neg'] & y['neg'], (q1['pos'], q2['pos']))
rule6 = ctrl.Rule(x['pos'] & y['neg'], (q1['neg'], q2['pos']))
rule7 = ctrl.Rule(x['zero'] & y['zero'], (q1['zero'], q2['zero']))
```

Listing 4: Bulanık Kural Tabanı

Bu bölümde, uç efektörün farklı konumlarına göre uygun eklem açılarını belirleyen 7 adet temel bulanık kural tanımlanmıştır.

4.5 Kontrol Sisteminin Tanımlanması ve Hesaplama

```
system = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6,
    rule7])
sim = ctrl.ControlSystemSimulation(system)

```

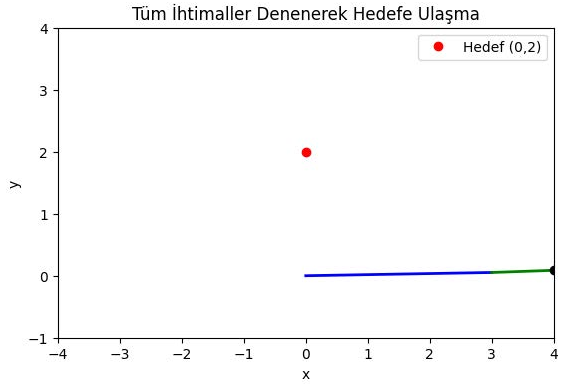
```
# Hedef pozisyonu giriyoruz:
sim.input['x'] = 0
sim.input['y'] = 2

# Hesaplama yap l r
sim.compute()
print('q1:', sim.output['q1'])
print('q2:', sim.output['q2'])
```

Listing 5: Kontrol Sisteminin Tanımlanması ve Hesaplama

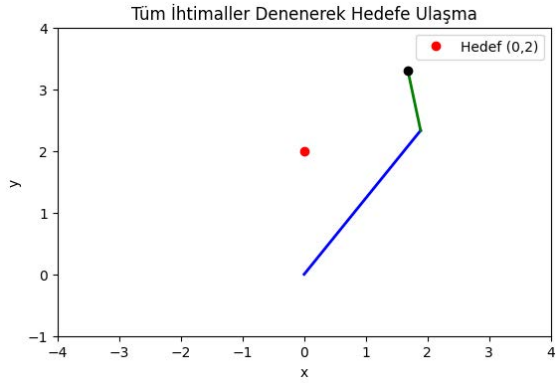
Bu bölümde kontrol sistemi oluşturulmuş, hedef konum atanmış ve simülasyon sonucunda elde edilen eklem açıları (q_1 ve q_2) ekrana yazdırılmıştır.

5 Çıktılar ve Sonuçlar

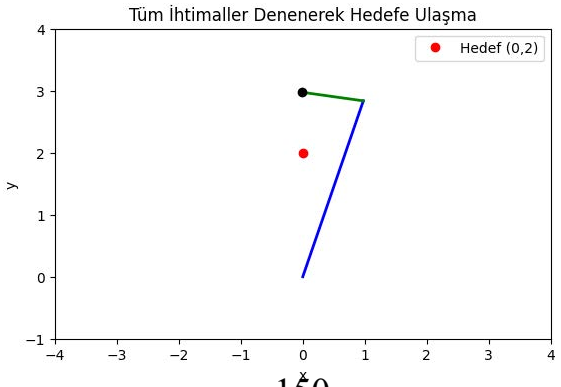
Not: Hesaplanan q_1 ve q_2 açılarını buraya ekleyiniz.



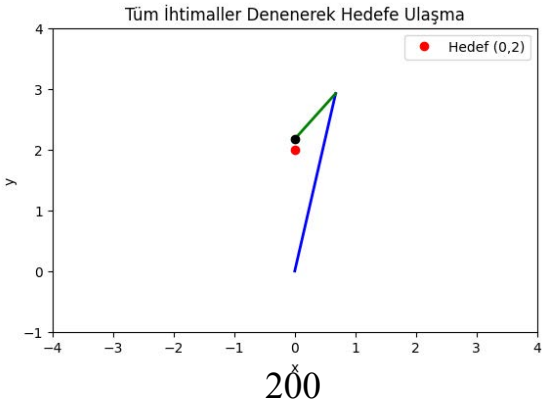
50



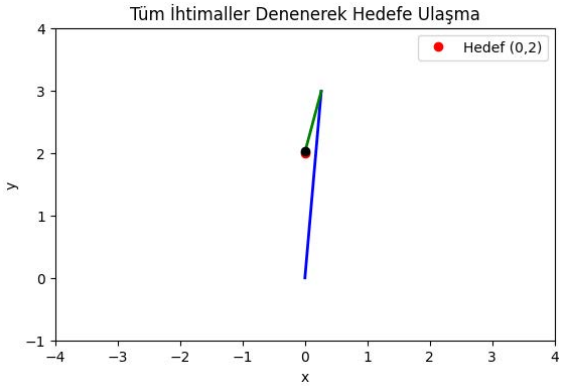
100



150



200

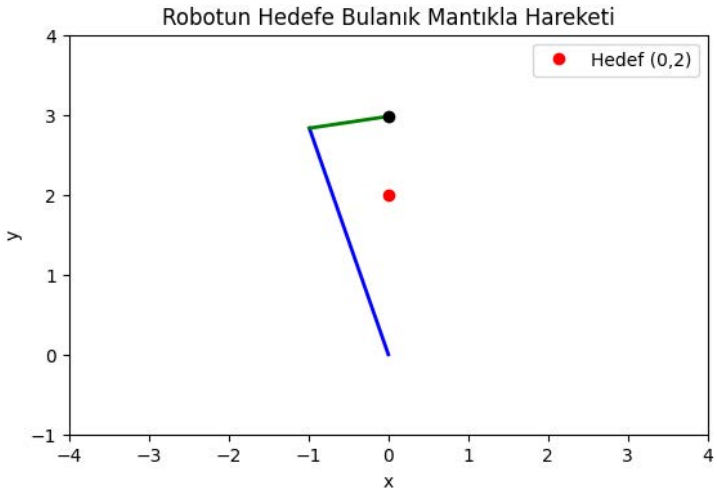


250 ve başarılı sonuç!

- $q1 = 85.00]$
- $q2 = [170.00]$

(Sonuç öncesi bilgilendirme)

Sağ tarafta da github üzerinde paylaştığım .ipynb dosyamda da görebileceğiniz ilk kurguladığım yanlış fuzzy algoritmasının "1500" adımda dahi ulaşabildiği en iyi konum gözükmektedir. Daha da iyi bir sonuca ulaşılamamış robot kolu sabitlenmiştir. Bu da fuzzy kurallarının belirlenmesinin önemini anlatmaktadır. "Doğru kurallarla 250 aşamada daha iyi sonuç aldık"



6 Sonuç

Bu çalışmada, iki eklemli bir robot kolunun uç efektörünün belirli bir konuma ulaşması için gereken eklem açıları bulanık mantık kullanılarak hesaplanmıştır. Elde edilen sonuçlar, bulanık mantık yönteminin ters kinematik problemlerinde alternatif bir yaklaşım sunduğunu göstermektedir.

OSTİM TEKNİK ÜNİVERSİTESİ
YZM304 ROBOTİK
FUZZY ALGORİTMASI ÖDEVİ

Dr. Öğr.Üyesi EROL DUYMAZ

Hasan Fatih Öztürk
220212002
Yapay Zeka Mühendisliği