



**BULANIK MANTIK DENETLEYİCİNİN
YAPAY SİNİR AĞI İLE MODELLENMESİ**

MURAT SAM

YÜKSEK LİSANS TEZİ

**ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ
ANA BİLİM DALI**

DR. ÖĞR. ÜYESİ MEHMET SERHAT CAN

Temmuz - 2021

Her hakkı saklıdır

T.C.
TOKAT GAZİOSMANPAŞA ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANA BİLİM DALI

YÜKSEK LİSANS TEZİ

BULANIK MANTIK DENETLEYİCİNİN YAPAY SİNİR AĞI İLE
MODELLENMESİ

MURAT SAM

TOKAT
Temmuz - 2021

Her hakkı saklıdır

TEZ BEYANI

Tez yazım kurallarına uygun olarak hazırlanan bu tezin yazılmasında bilimsel ahlak kurallarına uyulduğunu, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezin içerdığı yenilik ve sonuçların başka bir yerden alınmadığını, kullanılan verilerde herhangi bir tahrifat yapılmadığını, tezin herhangi bir kısmının bu üniversite veya başka bir üniversitedeki başka bir tez çalışması olarak sunulmadığını beyan ederim.

MURAT SAM

13 Temmuz 2021

ÖZET

YÜKSEK LİSANS TEZİ

BULANIK MANTIK DENETLEYİCİNİN YAPAY SİNİR AĞI İLE MODELLENMESİ

MURAT SAM

TOKAT GAZİOSMANPAŞA ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANA BİLİM DALI

(TEZ DANIŞMANI: DR. ÖĞR. ÜYESİ MEHMET SERHAT CAN)

Bulanık Mantık Denetleyici (BMD) bulanık küme teorisi temelli, insan düşünme mantığına yakın olarak oluşturulabilen ve doğrusal olmayan sistemlerin kontrolünde etkili çözümler sunan bir denetleyici yapısıdır. Kontrol performansı bakımından etkili sonuçlar sunmasıyla birlikte yavaş tepki vermesine neden olan çok sayıda matematiksel işlem içermektedir. Yapay Sinir Ağı (YSA) birçok sistemin modelini elde etmede kullanılabilir. YSA'nın paralel çalışma özelliğine sahip olması onun hızlı çalışmasını mümkün kılar. Ayrıca YSA modeli, BMD'den daha basit bir yapıya sahiptir ve bu da denetleyicinin donanım ve yazılım açısından uygulanmasını kolaylaştırır. Bu çalışmada YSA kullanarak BMD'nin kısa tepki süresine sahip bir modelini elde etmek amaçlanmıştır. BMD ve BMD'nin YSA modeli MATLAB yazılımında araç kutuları yardımıyla tasarlanmıştır. Önerilen yöntemin performansı, Simulink eklentisi kullanılarak ters sarkaç sisteminin kontrolü üzerinde benzetim çalışmaları ile test edilmiştir. BMD'nin giriş ve çıkış verileri kullanılarak bir veri seti oluşturulmuş ve bu veri seti ile YSA eğitilmiştir. Bu sayede BMD'yi çok iyi derecede modelleyebilen ve uygulanan aynı giriş değerlerine BMD gibi aynı çıkış değerleri veren bir YSA modeli elde edilmiştir. Farklı bir veri seti ile BMD ve BMD'nin YSA modeli çalıştırılarak karşılaştırması yapılmıştır. Simulink eklentisinde yapılan benzetim çalışmalarında BMD ve BMD'nin YSA modelinin kontrol performanslarının birbirine çok yakın olduğu ve hedeflendiği gibi YSA modelinin tepki süresinin, BMD'nin tepki süresinden çok daha hızlı olduğu görülmüştür. Önerilen yöntem hızlı tepki ihtiyacı olan kontrol uygulamalarında kullanılabilir.

2021, 101 SAYFA

ANAHTAR KELİMELER: Bulanık Mantık Denetleyici, Modelleme, Ters Sarkaç Sistemi, Yapay Sinir Ağları.

ABSTRACT

MASTER THESIS

MODELING OF FUZZY LOGIC CONTROLLER WITH ARTIFICIAL NEURAL NETWORK

MURAT SAM

TOKAT GAZIOSMANPASA UNIVERSITY

INSTITUTE OF GRADUATE STUDIES

DEPARTMENT OF ELECTRICAL ENGINEERING

(SUPERVISOR: ASST. PROF. DR. MEHMET SERHAT CAN)

Fuzzy Logic Controller (FLC) is a fuzzy set-based controller structure that can be formed close to human thinking logic and provides effective solutions in the control of nonlinear systems. It contains many mathematical operations that cause it to react slowly while providing effective results in terms of control performance. Artificial Neural Network (ANN) can be used to obtain models of many systems. The parallel operation feature of ANN makes it possible that it works fast. In addition, the ANN model has a simpler structure than FLC, which makes it easy to implement the controller in terms of hardware and software. In this study, it is aimed to obtain a model of FLC with a short response time by using ANN. FLC and FLC's ANN model were designed with the help of toolboxes in the MATLAB software. The performance of the proposed method has been tested by simulation studies on the control of the inverted pendulum system using the Simulink add-on. A data set was created using the input and output data of FLC and ANN was trained with this data set. Thus, an ANN model has been obtained that can model the FLC very well and give the same output values as FLC to the applied input values. The comparison of FLC and FLC's ANN model were performed by running with a different data set. In the simulation studies carried out in Simulink add-on, it has been observed that the control performances of FLC and FLC's ANN model are very close to each other and the response time of the ANN model is much faster than the response time of the FLC as targeted. The proposed method can be used in control applications that need fast response.

2021, 101 PAGES

KEYWORDS: Artificial Neural Networks, Fuzzy Logic Controller, Inverted Pendulum System, Modeling.

ÖNSÖZ

Yüksek lisans tez çalışmamı yaparken her türlü konuda desteğini esirgemeyen, tezin en iyi bir şekilde hazırlanabilmesi için güzel tavsiyeler veren değerli danışmanım Dr. Öğr. Üyesi Mehmet Serhat CAN hocama ve diğer bölüm hocalarıma sonsuz şükranlarımı sunarım. Ayrıca maddi ve manevi desteklerini esirgemeyen ve beni değerli hissettiren kıymetli aileme teşekkür ederim.

MURAT SAM

13 Temmuz 2021

İÇİNDEKİLER

Sayfa

ÖZET	i
ABSTRACT.....	ii
ÖNSÖZ	iii
İÇİNDEKİLER	iv
SİMGELER ve KISALTMALAR.....	vi
ŞEKİL LİSTESİ.....	x
ÇİZELGE LİSTESİ.....	xi
1. GİRİŞ	1
2. KAYNAK ÖZETLERİ.....	6
3. KURAMSAL TEMELLER	11
3.1. Bulanık Mantık	11
3.1.1. Bulanık kümeler.....	11
3.2. Bulanık Mantık Denetleyici	14
3.2.1. Dilsel ifadeler.....	16
3.2.2. Üyelik fonksiyonları.....	18
3.2.3. Bulanıklaştırma.....	24
3.2.4. Bulanık operatörler	27
3.2.5. Kural tabanı	36
3.2.6. Bulanık çıkarım.....	37
3.2.7. Durulaştırma	46
3.3. Yapay Sinir Ağı.....	51
3.3.1. Yapısı.....	52
3.3.2. Çeşitleri	59
3.3.3. Öğrenme süreçleri.....	64
4. MATERYAL ve YÖNTEM	67
4.1. MATLAB ve Simulink.....	67
4.2. Arabalı Ters Sarkaç Modeli.....	67
4.3. Bulanık Mantık Denetleyicinin Yapay Sinir Ağı ile Modellenmesi	73
4.3.1. Bulanık mantık denetleyici tasarımı	73

4.3.2. Bulanık mantık denetleyici kullanarak veri seti oluşturma.....	77
4.3.3. Yapay sinir ağlarının yapıları.....	78
4.3.4. Yapay sinir ağlarının eğitilmesi.....	80
5. BULGULAR.....	88
6. SONUÇ	97
7. KAYNAKLAR	98



SİMGELER ve KISALTMALAR

Simgeler	Açıklamalar
∂	Kısmi türev operatörü
ΔW	Ağırlık güncelleme vektörü
$\cap$	Bulanık kesişim işlemi
$\cup$	Bulanık birleşim işlemi
$\bar{A}, \bar{B}, \dots, \bar{F}$	Giriş üyelik fonksiyonlarının tümleyenleri
$A, B, \dots, F$	Giriş üyelik fonksiyonları
$\bar{A}_1, \bar{B}_1$ veya $\bar{C}_1$	x_1 'in $\bar{A}, \bar{B}$ veya $\bar{C}$ üyelik fonksiyonundaki üyelik değeri
A_1, B_1 veya C_1	x_1 'in A, B veya C üyelik fonksiyonundaki üyelik değeri
b	Sürtünme katsayısı
$B_1, B_2, \dots, B_k$	Katmanların eşik matrisleri
b_{j1}	j. hücrenin eşik değeri
ζ	Çıkış dilsel değişkeni
ζ_B	Bulanık çıkış kümesi veya değeri
ζ_K	Keskin çıkış değeri
$\bar{D}_2, \bar{E}_2$ veya $\bar{F}_2$	x_2 'nin $\bar{D}, \bar{E}$ veya $\bar{F}$ üyelik fonksiyonundaki üyelik değeri
D_2, E_2 veya F_2	x_2 'nin D, E veya F üyelik fonksiyonundaki üyelik değeri
DD	Doğrulama durumu değeri
$DD_{maksimum}$	DD değeri için maksimum sınır
e	Euler sayısı
F, u veya U	Arabaya uygulanan giriş kuvveti sinyali
$f_1, f_2, \dots, f_k$	Katmanların aktivasyon fonksiyonları
F_s	Sürtünme kuvveti
g	Gradyan değeri
G	Gradyan vektörü
G_1 ve G_2	Giriş dilsel değişkenleri
$g_{minimum}$	g değeri için minimum sınır
H	Hessian matrisi
H, I ve J	Çıkış üyelik fonksiyonları

H_D	Doğrulama için Hata vektörü
H_E	Eğitim için Hata vektörü
I	Sarkacın atalet momenti
i	Toplam giriş sayısı
$i_{maksimum}$	İterasyon değeri için maksimum sınır
J	Jacobian matrisi
JJ	Yaklaşık Hessian matrisi
K	Bir klasik küme
k	Katman sayısı
$K_1, K_2, \dots, K_9$	Kuralların ateşleme gücü değerleri
$K_{1H}, K_{2I}, \dots, K_{9J}$	Kuralların bulanık çıkış kümeleri veya değerleri
l	Sarkacın kütle merkezinin arabaya uzaklığı
L	Sarkacın uzunluğu
M	Çıkış hücresi sayısı
m	Sarkacın kütlesi
m_a	Arabanın kütlesi
N	x eksenindeki ağırlık bileşeni
P	y eksenindeki ağırlık bileşeni
P_D	Doğrulama örneği sayısı
P_E	Eğitim örneği sayısı
U_A	BMD_{ARABA} çıkış sinyali
U_S	$BMD_{SARKAÇ}$ çıkış sinyali
W	Ağırlık vektörü
$w_1, w_2, \dots, w_9$	Kural ağırlıkları
$W_1, W_2, \dots, W_k$	Katmanların ağırlık matrisleri
W_{aday}	Ağırlık aday vektörü
W_b	En iyi ağırlık vektörü
W_i	İlk ağırlık vektörü
$w_{j1}, w_{j2}, \dots, w_{ji}$	j . hücrenin ağırlık değerleri
$\dot{x}$	Arabanın hızı
$\ddot{x}$	Arabanın ivmesi

x	Arabanın konumu
x_1 ve x_2	Keskin giriş değerleri
$x_{11}, x_{21}, \dots, x_{i1}$	1. katmanın giriş değerleri
x_G	Arabanın gerçek konumu sinyali
x_H	Arabanın konumunun hatası sinyali
x_{HD}	Arabanın konumunun hatasının değişimi sinyali
x_R	Arabanın referans konumu sinyali
$Y_0, Y_1, \dots, Y_{k-1}$	Katmanların giriş matrisleri
$Y_1, Y_2, \dots, Y_k$	Katmanların çıkış matrisleri
Y_G	Gerçek çıkış matrisi
y_{j1}	j. hücrenin çıkış değeri
Y_R	Referans çıkış matrisi
$\dot{\theta}$	Sarkacın açısal hızı
$\ddot{\theta}$	Sarkacın açısal ivmesi
θ	Sarkacın açısal konumu
θ_G	Sarkacın gerçek açısal konumu sinyali
θ_H	Sarkacın açısal konumunun hatası sinyali
θ_{HD}	Sarkacın açısal konumunun hatasının değişimi sinyali
θ_R	Sarkacın referans açısal konumu sinyali
μ	Kombinasyon katsayısı değeri
$\mu_A(x)$	Bir x elemanının A bulanık kümesindeki üyelik değeri
μ_b	μ değeri için büyütme çarpanı
μ_i	İlk μ değeri
μ_k	μ değeri için küçültme çarpanı
$\mu_{maksimum}$	μ değeri için maksimum sınır
τ	Tork
$\chi_K(x)$	Bir x elemanının K klasik kümesindeki üyelik değeri

Kısaltmalar	Açıklamalar
ANFIS	Uyarlanabilir Sinirsel Bulanık Çıkarım Sistemi
ANN	Artificial Neural Network
BMD	Bulanık Mantık Denetleyici
DA-DA	Doğru Akım-Doğru Akım
FLC	Fuzzy Logic Controller
<i>hardlim</i>	Katı Sınır fonksiyonu
<i>logsig</i>	Logaritmik Sigmoid fonksiyonu
LOM	Maksimumların Sonuncusu
<i>max</i>	Maksimum operatörü/fonksiyonu
<i>min</i>	Minimum operatörü/fonksiyonu
MOM	Maksimumların Ortalaması
MSE	Ortalama Karesel Hata
NB	Negatif Büyük
NK	Negatif Küçük
PB	Pozitif Büyük
PID	Oransal-İntegral-Türev
PID^{λ}	Kesir Dereceli PID
PK	Pozitif Küçük
<i>probor</i>	Cebirsel Toplam operatörü
<i>prod</i>	Cebirsel Çarpım operatörü
<i>purelin</i>	Doğrusal fonksiyon
<i>radbas</i>	Radyal Tabanlı fonksiyon
S	Sıfır
SOM	Maksimumların İlki
<i>sum</i>	Toplam operatörü
<i>tansig</i>	Tanjant Sigmoid fonksiyonu
<i>trainlm</i>	Levenberg-Marquardt algoritması
YSA	Yapay Sinir Ağı

ŞEKİL LİSTESİ

Şekil

Sayfa

Şekil 3.1. Klasik ve bulanık küme	13
Şekil 3.2. BMD yapısının ana hatları.....	15
Şekil 3.3. Üçgen üyelik fonksiyonu.....	19
Şekil 3.4. Yamuk üyelik fonksiyonu	20
Şekil 3.5. Gauss üyelik fonksiyonu	21
Şekil 3.6. Sigmodial üyelik fonksiyonları	23
Şekil 3.7. Çan eğrisi üyelik fonksiyonu.....	24
Şekil 3.8. Birinci giriş için örnek üyelik fonksiyonları.....	25
Şekil 3.9. İkinci giriş için örnek üyelik fonksiyonları	26
Şekil 3.10. Bulanık tümlleme operatörünün çıkışı.....	29
Şekil 3.11. Bulanık kesişim operatörlerinin çıkışları (Mamdani metodu).....	32
Şekil 3.12. Ara bulanık çıkış kümeleri (Mamdani metodu)	34
Şekil 3.13. Bulanık birleşim operatörlerinin çıkışları (Mamdani metodu).....	35
Şekil 3.14. Çıkış üyelik fonksiyonları (Mamdani metodu)	40
Şekil 3.15. Çıkış üyelik fonksiyonları (Tsukamoto metodu).....	43
Şekil 3.16. Durulaştırma sonuçları (Eksenin sol tarafı) (Mamdani metodu).....	49
Şekil 3.17. Durulaştırma sonuçları (Eksenin sağ tarafı) (Mamdani metodu)	50
Şekil 3.18. Hardlim transfer fonksiyonu.....	53
Şekil 3.19. Purelin transfer fonksiyonu	54
Şekil 3.20. Logsig transfer fonksiyonu	55
Şekil 3.21. Tansig transfer fonksiyonu	56
Şekil 3.22. Radbas transfer fonksiyonu	57
Şekil 3.23. YSA'nın ilk gizli katmanındaki j . yapay sinir hücresi.....	58
Şekil 3.24. Tek katmanlı ileri beslemeli ağ yapısı	61
Şekil 3.25. İki gizli katman içeren çok katmanlı ileri beslemeli ağ yapısı	64
Şekil 4.1. Arabalı ters sarkaç serbest gövde diyagramı	68
Şekil 4.2. Üyelik fonksiyonları (BMD_{ARABA}).....	74
Şekil 4.3. Üyelik fonksiyonları ($BMD_{SARKAÇ}$)	74
Şekil 4.4. Yüzey görünümü (BMD_{ARABA})	76
Şekil 4.5. Yüzey görünümü ($BMD_{SARKAÇ}$).....	76
Şekil 4.6. BMD'lerin eğitimi için tasarlanan benzetim sisteminin ana hatları	77
Şekil 4.7. BMD'lerin eğitimi için tasarlanan benzetim sisteminin tamamı.....	78
Şekil 5.1. YSA'ların yapısı ve içeriği	88
Şekil 5.2. Ortalama Karesel Hata (YSA_{ARABA}).....	89
Şekil 5.3. Ortalama Karesel Hata ($YSA_{SARKAÇ}$)	89
Şekil 5.4. Eğitimin regresyonu (YSA_{ARABA})	90
Şekil 5.5. Eğitimin regresyonu ($YSA_{SARKAÇ}$).....	91
Şekil 5.6. BMD'ler yerine YSA'lar konulduğunda benzetim sisteminin ana hatları	92
Şekil 5.7. BMD'ler yerine YSA'lar konulduğunda benzetim sisteminin tam hali	92
Şekil 5.8. Arabanın konumu	93
Şekil 5.9. Arabanın konumu (fark)	94
Şekil 5.10. Sarkacın açısal konumu	95
Şekil 5.11. Sarkacın açısal konumu (fark).....	96

ÇİZELGE LİSTESİ

<u>Çizelge</u>	<u>Sayfa</u>
Çizelge 3.1. Kural tabanı örneği	17
Çizelge 3.2. Bulanık operatörler	28
Çizelge 3.3. Bazı girişler için operatörlerin çıkışları	35
Çizelge 3.4. Kural tabanı	36
Çizelge 4.1. Çalışmada kullanılan semboller, açıklamalar, değerler ve birimler	69
Çizelge 4.2. Kullanılan üyelik fonksiyonlarının evrensel küme aralıkları	73
Çizelge 4.3. BMD'lerin kural tablosu.....	75
Çizelge 4.4. Tasarlanan YSA yapıları	79
Çizelge 4.5. Trainlm algoritması parametreleri	81

1. GİRİŞ

İnsanlar gün geçtikçe her türlü faaliyeti daha konforlu, hızlı ve güvenli bir şekilde yapmak isterler. Örneğin insanlar kara, hava, deniz ve raylı ulaşım sistemleri ile seyahat ederken varmak istedikleri noktaya en az sarsıntıyla, en kısa sürede ve en güvenli bir şekilde ulaşmak isterler. Bu ise ulaşım için kullanılan vasıtanın bu üç temel kriteri sağlayacak kontrol sistemi ile donatılmasına bağlıdır. Bu örnekteki konfor, hız ve güvenlik gibi gereksinimler günlük hayatımızdaki birçok ihtiyacımız için de geçerlidir.

İhtiyaçlarımızı karşılayan ulaşım, iletişim, üretim vb. faaliyetlerin daha konforlu, hızlı ve güvenli olabilmeleri için daha iyi kontrol sistemleri ile donatılması gerekmektedir. Kontrol sistemlerinin bütün bu amaçlara en iyi şekilde hizmet edebilmesi otomatik hale getirilmesiyle mümkün olabilmektedir. Kontrolü insan yardımı almadan yapabilen otomatik kontrol sistemlerinden genellikle daha hızlı ve doğru cevap üretebilmesi beklenilmektedir. Örneğin bazı hassas ayar gerektiren kontrollerde otomatik kontrol sistemi kontrol sinyalini sadece birkaç milisaniye içerisinde bile çok yüksek miktarlarda değiştirebildiğinden her an için daha doğru kontrol sinyali sisteme uygulanabilmektedir. Bu sayede kontrol performansında ciddi düzeyde iyileşme sağlanmaktadır. Hızı ve doğruluğu artırabilmek için otomatik kontrol sistemlerini daha iyi hale getirmeye çalıştıkça genellikle tasarımın karmaşıklığı artma eğilimindedir ancak bu durum istenilmemekte, tasarımın kolay olması beklenilmektedir. Tasarımların kolay olması hem maliyet hem de kolay uygulanabilirlik açısından avantajlar sağlamaktadır. Ayrıca kolay tasarıma sahip kontrol sistemlerinin öğrenilmesi ve öğretilmesi daha kolay ve bu tasarımın geliştirilme ihtimali çok daha olasıdır.

Sisteme giren kontrol sinyali ile sistemin verdiği çıkış arasında matematiksel bir model oluşturmak otomatik kontrolün neye göre ve nasıl yapılacağını bilebilmek için çok önem teşkil etmektedir. Örneğin matematik olarak kolay ifade edilebilen doğrusal sistemlerin kontrol edilmesi de otomatik kontrol sistemleri kullanılarak oldukça kolay olmaktadır. Böyle sistemlere uygulanan kontrol sinyaline sistem tarafından aynı oranda çıkış cevabı üretilmekte ve bu gibi sistemler doğrusal sistem olarak tanımlanmaktadır. Teorikte bütün değerlerde sonsuza kadar doğrusal olarak elde edilebilecek çıkış cevabı pratikte genellikle bir aralıkta doğrusal olmaktadır. Doğrusal olan aralık sistem kontrolü

için istenen aralığı kapsıyorsa sistem yine doğrusal gibi düşünülüp kontrol edilebilmektedir.

Newton'a göre evren doğrusal bir sistem gibi çalışmaktadır. Newton'un öne sürdüğü temel kavramlardan biri olan doğrusallık, karmaşıklık teorisi ile çelişmektedir. Nitekim günümüzde kontrol etmeye çalıştığımız sistemlerde doğrusallığı ve tahmin edilebilirliği reddeden bu teoriye göre çalışmakta, uygulanan kontrol sinyaline doğrusal olmayan çıkış cevabı üretmektedir (Sayğan, 2014).

Bütün sistemler için doğrusal bir formül bulmak her zaman mümkün olmamaktadır. Sistemlerde yer alan birçok faktör doğrusallığı bozmaktadır. Bu tarz sistemler için bazen doğrusal olmayan matematiksel hesaplamalar çıkarılabilmekte ise de bazı sistemler için bir matematiksel ifade bulmak bile mümkün olmamaktadır. Bu yüzden formülü bilinmeyen bilinse bile çok karmaşık olan bu tarz doğrusal olmayan sistemleri kontrol edebilmek için daha iyi kontrol sistemleri gerekmektedir.

Otomatik kontrol sistemleri genel olarak açık çevrim ve kapalı çevrim sistemler olarak sınıflandırılır. Açık çevrim sistemde kontrol edilen sistemin anlık değerinin gözlemlenmesi için bir geri besleme elemanı bulunmamaktadır. Bu ise çıkış üzerinde oluşacak bozucu etkilerin matematiksel olarak ifade edilememesine yol açmaktadır. Ayrıca matematiksel ifade elde edilirken yapılan bazı ihmaller veya olası küçük işlem hataları küçük olsalar bile, sistem bazı durumlarda düzgün çalışırken bazı durumlarda ise tahmin edilemeyecek şekilde büyük etkiler gösterebilmektedir. Bu tarz tahmin edilemeyen durumları bazen önleyebilmek bazen ise azaltabilmek gerekmektedir. Bu yüzden pek çok sistemin kontrolü için olası durumlarda geri beslemesi sayesinde sisteme gerekli ekstra işareti verebilmenin mümkün olduğu kapalı çevrim kontrol sistemleri genellikle tercih edilmekte ve böylece kontrol performansı artmaktadır. Bu sistemler içerisinde genellikle denetleyici, kontrol edilen sistem ve kontrol edilen değişkenleri ölçen sensör bulunmaktadır. Böylece sistemin maruz kaldığı bozucuların etkisinin azaltılması sağlanmaya çalışılmaktadır (Erkol, 2017).

Doğrusal sistemler için Oransal-İntegral-Türev (PID) gibi uygulanması kolay yöntemler yeterli bir kontrol sağlayabilmektedir. Ancak karmaşık ve doğrusal olmayan sistemlerin

kontrolü için PID denetleyici etkili ve yeterli olamamaktadır (Muliadi ve Kusumoputro, 2018; Khalid ve ark., 2019; Zaihidee ve ark., 2019). Doğrusal olmayan sistemlerin kontrolü için araştırmacılar tarafından çeşitli yöntemler kullanılarak farklı yaklaşımlar yapılmıştır. Bulanık mantık, genetik algoritma ve sinir ağı yöntemleri kullanılmış ve ayrıca bunların bir şekilde bir arada kullanılmasıyla hibrit yaklaşımlar da yapılmıştır. Bunlar ise bulanık-genetik, nöro-bulanık ve nöro-genetik yaklaşımlardır (Singh ve Thongam, 2017). Ayrıca kayan kipli kontrol, tahmini kontrol, geri adım kontrolü, uyarlanabilir kontrol, H_∞ kontrolü gibi yöntemler mevcuttur (Zaihidee ve ark., 2019). Bu yöntemlerden bir tanesi olan Bulanık Mantık Denetleyici (BMD) uzman tecrübesi yardımı ile oluşturulmuş bir takım kuralları bulanık küme teorisindeki bulanık operatörler ile değerlendirerek özellikle doğrusal olmayan sistem kontrollerinde etkili sonuçlar üretebilmektedir (Rahimi ve ark., 2013). BMD'ler sistem kontrolü (Abd-elaziz, 2019; Ahmed ve ark., 2018), diğer sinir ağı yöntemiyle birlikte kullanılması sonucu oluşan Uyarlanabilir Sinirsel Bulanık Çıkarım Sistemi (ANFIS) ile sistem kontrolü (Al-Mekhlafi ve ark., 2018; Farah ve ark., 2020) ve başka bir denetleyicinin katsayılarının adaptif olarak ayarlanması (Mien, 2017; Pathak ve Paliwal, 2017) gibi çeşitli kontrol uygulamalarında kullanılmaktadır. Bir diğer yöntem olan Yapay Sinir Ağı (YSA) yapısı gereği öğrenme özelliğine sahip olmasından dolayı birçok çalışmaya konu olmuştur. Ayrıca hem çok karmaşık matematiksel hesaplamalar gerektirmedikinden hem de paralel işlem yapabildiğinden dolayı hızlı çalışır. Donanımsal ve yazılımsal olarak gerçekleştirilmesi kolaydır. YSA'lar sistem kontrolü (Salim ve ark., 2019; Karunasena ve ark., 2020), sistem modelleme (Chawla ve Singla, 2019; Marie-Francoise ve ark., 2006), denetleyici katsayılarının adaptif olarak ayarlanması (Bari ve ark., 2019; Slama ve ark., 2019) gibi çeşitli kontrol uygulamalarında kullanılmaktadır.

BMD sisteminin bir yapı taşı olan bulanık kümelerin klasik kümelerden farkı kümeyi oluşturan elemanları ifade etmede kullanılan üyelik değerlerinde yatmaktadır. Klasik kümede bir elemanın üyelik değeri eğer kümenin elemanı ise 1, değilse 0 olabilirken, bulanık kümede $[0,1]$ kapalı aralığında sonsuz değerde olabilmektedir. BMD'nin diğer bir yapı taşı ise günlük hayatta çokça kullandığımız kesin olmayan dilsel ifadelerdir. Bu dilsel ifadelerin matematiksel olarak ifade edilebilmesi için bu ara değerler çok önemlidir. BMD'ler bu dilsel ifadelerin yardımı ile kodlanmış bir algoritmayı

kullanarak çıkarım yapmakta ve kontrolü gerçekleştirmektedir. Ve bu sayede daha insancıl bir kontrol yapılmaya çalışılmaktadır (Altaş, 1999).

YSA yapısı gereği öğrenme özelliğine sahiptir. Öğrenme özelliği sayesinde daha önce öğretilmeyen giriş verileri için genelleme yaparak karmaşık problemleri çözebilir. YSA'nın en küçük birimi olan hücre insan ve hayvanlardaki sinir hücresini model alan genellikle doğrusal olmayan yazılım veya donanım birimidir (Ergezer ve ark., 2003). Bu hücreler bir araya gelerek YSA katmanlarını oluşturur. YSA yapısında bir giriş katmanı, bir çıkış katmanı mutlaka olmalıdır. Sadece bu katmanlar olduğunda giriş katmanında hiçbir hesaplama yapılmadığından tek katmanlı ağ şeklinde isimlendirilir. Ayrıca bu iki katmanın arasında gizli katman olarak isimlendirilen başka katmanlarda bulunabilir. Bu durumda ise çok katmanlı ağ şeklinde isimlendirilir. Bu katmanlarda yer alan bir hücre kendi katmanı dışında başka bir katmandaki en az bir hücre ile bağlantısı bulunur. Bu bağlantıların her biri dönen verideki hataya göre devamlı kendilerini güncelleyerek en uygun yapıyı oluşturmaya çalışırlar. Bu şekilde YSA öğrenmeyi tamamlar ve yeni hiç öğretilmeyen giriş değerleri geldiğinde oluşan son ağırlık ve eşik değerlerini kullanarak işlemleri yapmakta ve sonuç üretmektedir (Haykin, 2009).

Bu tez çalışmasında, BMD'nin hem yazılım hem de donanım olarak tasarlanması kolay ayrıca kısa tepki süresine sahip bir modelini elde edebilmek için YSA kullanılmıştır. İlk olarak kontrol işlemleri için benzetim sistemi, BMD, BMD'nin YSA modeli ve yöntemin performansını test etmek için kontrolü yapılacak arabalı ters sarkaç sistemi MATLAB yazılımı kullanılarak her biri için özel araç kutuları yardımıyla tasarlanmıştır. Tasarlanan benzetim sisteminde yer alan arabalı ters sarkaç sisteminin kontrolü öncelikle BMD ile yapılmıştır. Olası referans değerlerine göre BMD sarkacı düşürmeden arabanın kontrolünü gerçekleştirmiştir. Kontrol esnasında BMD'ye gelen giriş verileri ve BMD'nin çıkışında elde edilen çıkış verileri ile YSA eğitimi için veri seti oluşturulmuştur. Bu veri seti kullanılarak YSA eğitilmiş hazır hale getirilmiştir. Bu yapılarak BMD'yi çok iyi derecede modelleyebilen ve uygulanan aynı giriş değerlerine BMD gibi aynı çıkış değerleri veren bir YSA modeli elde etmek amaçlanmıştır. YSA işlem hızının artırılmasını sağlayan paralel yapısı sayesinde, BMD'ye göre çok hızlı çalışmasından dolayı özellikle gerçek zamanlı sistemlerde oluşacak kontrol gecikmesinin önüne geçilebilecektir. Testin gerçekçiliğini artırmak için, hiç

öğretilmeyen farklı bir giriş veri seti aracılığıyla BMD ve BMD'nin YSA modeli aynı sistem için çalıştırılarak karşılaştırması yapılmıştır. İki denetleyicinin kontrol sonuçlarının birbirine çok yakın olduğu görülmüştür. BMD, bir YSA yapısı ile yüksek benzerlik oranında modellenmiştir. Sonuç olarak, hedeflendiği gibi önerilen modelin kontrol hızının, BMD'nin kontrol hızından çok daha hızlı olduğu görülmüştür. Ayrıca önerilen model BMD'den daha basit bir yapıya sahiptir ve bu da denetleyicinin donanım ve yazılım açısından kolayca uygulanmasını mümkün kılmaktadır.



2. KAYNAK ÖZETLERİ

BMD'nin ana çıkış noktası neden bazı sistemlerde insanlar basit denetleyicilerden daha iyi kontrol yapabiliyor sorusu olmuştur. Daha iyi kontrolün sebebinin insanların kullandığı dilsel kesin olmayan ifadelerin etkisi olabileceği 1965 yılında Zadeh tarafından dolaylı olarak öne sürülmüştür. Bu nedenle insan gibi dilsel ifadelerle karar verebilen bir denetleyici tasarlanabilirse, kontrol performansının artabileceği düşünülmüştür. Bu teoride kümeler artık klasik kümeler olarak isimlendirilmiş ve yeni bir küme çeşidi olarak bulanık kümeler tanımlanmıştır. Bu kümeler denetleyicinin yapı taşını oluşturmaktadır. Ancak bulanık kümeler aracılığıyla yeni bir kontrol sistemi oluşturma fikri pek ilgi görmemiştir. 1971'de Zadeh bir durum denklemleri teorisi ortaya atmıştır ve 1972'de bulanık denetimin nasıl olabileceğini ana hatlarıyla açıklamıştır. 1973'te ise eğer oda biraz sıcaksa kombiyi biraz kıs gibi dilsel ifadelerin denetleyici tasarımında nasıl kullanılabileceği açıklamıştır. Aynı zamanda Chang ve Zadeh tarafından bu kümeler yardımıyla oluşturulan BMD teorik olarak ortaya atılmıştır. Bütün bu teorik makaleler ışığında 1975 yılında ilk uygulama çalışmaları Mamdani ve Assilian tarafından gerçekleştirilmiştir. Sonraları yapılan çalışmalar olsa da en büyük canlanma noktası 1987 yılında Hitachi firmasının da BMD sistemini kullanması olmuştur. Sistem ticari olarak kullanılmış ve sıradan kontrol sistemlerine göre sağladığı daha iyi kontrol sayesinde ticari başarı getirebileceği ispatlanmıştır. 1987 yılında Yamakawa ters sarkaç sistemi oluşturup bunu BMD yardımı ile kontrol etmiştir. BMD'nin doğrusal olmayan sistemlerde etkisini daha fazla ispat edebilmek için üzerinde sıvı dolu bardak hatta rastgele hareket eden bir fare bile varken BMD'nin ne kadar kararlı bir kontrol yaptığını ispatlamıştır (Schwartz ve ark., 1994).

İnsan beyni tarafından gerçekleştirilen hata yapıldıkça öğrenmenin yapılması YSA'nın çıkış mantığı olmuştur. Belirli bir algoritma kullanılarak benzer bir sistem elektronik ortamda uygulanabilirse insan gibi hata yaptıkça öğrenen öğrendiği bilgileri kullanarak yeni daha önce yapmadığı aktiviteleri yapabilen elektronik bir sistem tasarlanabilir. Ve bu sistemin birçok alanda kullanılabileceği düşünülmektedir. Bu tarz düşünceler sürerken 1943 yılında Warren McCulloch ve Walter Pitts tarafından insan beynini modelleyebilmek için sinir ağı kullanılabileceğini gösteren çok sayıda teorem ortaya atılmıştır. Daha sonraları 1949 yılında Hebb tarafından daha gerçekçi sinir ağları

kullanabilmek için insan beynindeki sinir hücrelerinin nasıl öğrendiği ile ilgili çalışma yapılmıştır. İnsan beynindeki sinir hücrelerinin nasıl çalıştığı daha fazla netlik kazandıktan sonra 1951 yılında Minsky ve Dean Edmunds SNARC (*stochastic neural analog reinforcement calculator*) adında sinir ağını kullanarak çalışan bir makine yapmıştır. 1955 yılında Allen Newell ve Herbert Simon Mantık teorisyeni adıyla yapay zeka programı yapmışlardır (Park ve Park, 2018). 1958 yılında McCulloch ve Pitts'in önerdikleri teoremlerin yardımı ile Rosenblatt tarafından Perceptron YSA modeli önerilmiştir. 1959 yılında Arthur Samuel makine öğrenmesi terimini tanıtarak yapay zekanın gelişimini hızlandırmıştır. 1960 yılında Widrow ve Hoff, teorik çalışmalar ve pratik uygulamaları başarılı bir şekilde gerçekleştirmişlerdir. Sonraları 1970'li yıllarda birçok problemi çözmedeki uygulama ile ilgili zorluklar gelişim hızını azaltmıştır. Paul Werbos 1974 yılında yaptığı çalışma ile bazı sorunları kaldırmayı sağlayan tek katman kısıtlamasını eleme etmiştir (Tolon ve Tosunoğlu, 2008).

Bir çalışmada, YSA kullanarak süper kapasitörlerin termal ve elektriksel davranışının değerlendirilmesi için bir modelleme aracının geliştirilmesini sunulmuştur. Prensipte, kara kutu çok girişli tek çıkışlı bir modelden oluşur. Sistem girişleri sıcaklık, akım ve süper kapasitör değerleridir ve çıkış süper kapasitör voltajıdır. Giriş ve çıkış arasındaki ilişki, farklı akımlarda ve farklı sıcaklıklarda süper kapasitörlerin deneysel şarj ve deşarj döngülerinden YSA modelinin öğrenilmesi ve doğrulanmasıyla oluşturulmuştur. Eğitim parametreleri bilindikten sonra, YSA modeli süper kapasitörlerin farklı parametrelerini tahmin edebilir. YSA modelinin güncelleme parametreleri, sistemin çıktısı ile tahmin edilen çıktı arasındaki hatayı en aza indirmek için Levenberg-Marquardt algoritması kullanılarak gerçekleştirilmiştir. Bu metodoloji, YSA modeli ile, termal etkileri hesaba katarak süper kapasitörlerin geçici davranışları hakkında yararlı bilgiler sağlayabilmektedir (Marie-Francoise ve ark., 2006).

Diğer bir çalışmada, Doğru Akım-Doğru Akım (DA-DA) dönüştürücülerin, farklı sistemlere güç sağlamak için kullanılan yüksek verimli araçlar olduğu, doğrusal olmayan bir davranışları olduğu ve ana parametrelerindeki farklılıkların kararlılıklarını etkileyebildiğinden bahsedilmiştir. Çalışmada, bir Buck dönüştürücüye uygulanan farklı kontrol stratejileri, doğrusal ve doğrusal olmayan denetleyiciler incelenmiş ve karşılaştırılmıştır. Çalışmada ele alınan başlıca üç kontrol stratejisi vardır. Birincisi, QPI

(*quadratic performance index*) kullanılarak optimal kontrol tabanlı bir tasarım kullanılmış, ikincisi bilgi tabanlı bulanık bir kontrol çalışılmış ve üçüncü olarak BMD'nin dinamik bir modeli olarak YSA önerilmiştir. Bu önerilen üç denetleyiciyi dayanıklılık açısından karşılaştırmışlardır. YSA ile modellenmiş BMD'nin gerçek BMD'ye göre neredeyse yarı sürede BMD'nin sağlamlığına sahip bir şekilde kontrol performansı göstermiştir (Diaz ve Soriano, 2007).

Diğer bir çalışmada, insansız hava araçlarının uygulamalarının, model uyumsuzluğu, rüzgar bozuklukları, ölçüm gürültüsü ve değişen elektriksel değişkenlerin etkileri, örneğin pil voltajındaki kayıp gibi bozucu etkileri hafifletebilen sağlam kontrol şemaları gerektirir. Başka çalışmalarda, Kesir Dereceli PID ($PI^{\lambda}D^{\mu}$) denetleyici bir çare olarak önerilmiştir. Bu çalışmada ise, bir YSA'nın, zamanla değişen eylem katsayılarına ve farklı entegrasyon sıralarına sahip analog bir $PI^{\lambda}D^{\mu}$ denetleyicisinin yanıtına yaklaşan, Sonlu Dürtü Yanıtı (FIR) tipi yaklaşıtırcının katsayılarını sağlamak için eğitilebileceğini göstermektedir. Elde edilen sonuçlar, sinir ağı destekli FIR tipinde bir denetleyicinin, aracın önceden belirlenmiş yörüngelere doğru bir şekilde sürülmesinde çok başarılı olduğunu göstermiştir. Önerilen YSA modeli, hedef $PI^{\lambda}D^{\mu}$ denetleyicinin yanıtına oldukça benzerdir ve hesaplama yükü de oldukça düşüktür (Efe, 2011).

Diğer bir çalışmada, $PI^{\lambda}D^{\mu}$ denetleyicilerinin bir YSA yapısı ile gerçekleştirilmesi incelenmiştir. Çalışmanın motivasyonu, $PI^{\lambda}D^{\mu}$ denetleyicinin sahip olduğu kesirli sıra operatörlerini gerçek zamanlı olarak gerçekleştirmenin zorluğudur. İyi bir yaklaşım, frekans spektrumunda uygun şekilde programlanmış onlarca kutup ve sıfır gerektirir ve daha sonra böyle yüksek dereceli bir sistemin çözümüne ihtiyaç vardır. Bu çalışma, $PI^{\lambda}D^{\mu}$ denetleyiciyi taklit edebilen bir YSA sunarak bu hesaplama sorununu çözmektedir. $PI^{\lambda}D^{\mu}$ denetleyicilerinin YSA modelinin kullanılmasıyla elde edilebilecek performansı değerlendirmek için birkaç örnek durum sunulmuş ve YSA modelinin kullanımının $PI^{\lambda}D^{\mu}$ denetleyicinin gerçekleştirilmesinde pratik bir alternatif olduğu görülmüştür. Ayrıca, geliştirilen YSA modelinde yer alan bu parametreler ağı sürekli girdiler olarak sağlandığından denetleyici parametrelerinin çevrimiçi olarak değiştirilmesi mümkündür (Efe, 2012).

Diğer bir çalışmada, Kardiyotokografide tüm parametrelerin bağlı olduğu fetal kalp hızı parametresi için doğru sınıflandırma yapmanın son derece önemli olduğuna değinilmiştir. Hekimler tarafından verilen değerlendirmedeki doğal belirsizliğin, muhtemelen bulanık mantık kullanılarak modellenebileceğinden bahsedilmiştir. İnsanın dilsel ifadesinde özünde var olan belirsizliği ele almak için en güvenilir araçlardan biri olarak bulanık mantık çalışmada kullanılmıştır. Bulanık mantık tabanlı bir sistem tasarlamadaki ana zorluğun, üyelik fonksiyonu tasarlamak olduğundan bahsedilmiştir. Çalışmada, fetal kalp hızının bulanık üyelik fonksiyonunun tasarımı için YSA tabanlı bir teknik sunulmuştur ve Kardiyotokografiyi sınıflandırmak için Bulanık Sırasız Kural İndüksiyon Algoritması (*Fuzzy Unordered Rule Induction Algorithm*) kullanılmıştır. Elde edilen sonuçlar, bulanık üyelik fonksiyonu temelli olmayan tekniğe göre sınıflandırmada önemli gelişme göstermiştir (Das ve ark., 2015).

Diğer bir çalışmada, Şebekeye Bağlı Fotovoltaik Sistem, yenilenebilir enerjide daha çok kullanılan bir sistem haline geldiğinden bahsedilmiştir. Bu sistemin verimliliği artırmak ve enerji kayıplarını azaltmak için çeşitli araştırmalar yapılmıştır. Verimliliği artırmak için kullanılan önemli bileşenlerden biri DA-DA güçlendirme dönüştürücüdür. Çalışmada, DA-DA dönüştürücüyü kontrol etmek için BMD ve YSA üzerine inşa edilmiş yeni bir hibrit model önerilmiştir. Modeli oluşturmak için izlenen yol üç adımda sunulmuştur. İlk adımda BMD'ye dayalı bir veri oluşturulmuş, bir sonraki adımda BMD'yi modellemek için bir YSA yapısı seçilmiş ve son adımda, elde edilen modelin testi ve doğrulaması yapılmıştır. YSA eğitme aşaması, geri yayılma algoritmasına dayalı denetimli öğrenme ile gerçekleştirilmiştir. Bu algoritma, BMD çıkışı ile YSA modeli arasında minimum Ortalama Kare Hatanın Karekökü olan en uygun ağırlık ve eşik değerlerini arayarak YSA modelini eğitmek için kullanılmıştır. Doğrulama testi, hem akıllı denetleyiciler hem de klasik Değiştir ve Gözle (P&O) denetleyici arasında aynı anda çeşitli ışınlama değerleri ile gerçekleştirilmiştir. Bu karşılaştırmalardan sonra, YSA modelinin performansının, P&O denetleyiciden daha hızlı tepki süresi ile daha iyi sonuç verdiği sonucuna varmışlardır (Benkercha ve ark., 2017).

Diğer bir çalışmanın amacı, doğrusal olmayan tek girişli çok çıkışlı bir sistemi tanımlamada ANFIS etkinliğini göstermektir. Çalışmada kullanılan sistem son derece doğrusal olmayan, kararsız ve az harekete geçirilmiş doğası ile iyi bilinen arabalı ters

sarkaktır. Arabalı ters sarkacın ANFIS modeli, doğrusal olmayan matematiksel modelden elde edilen giriş ve çıkış verileri kullanılarak MATLAB yazılımında Simulink eklentisi ile tasarlanmıştır. Farklı başlangıç koşulları için benzetimin cevapları, sistemin matematiksel modeliyle karşılaştırılan ANFIS modelinden elde edilmiştir. Eğitimli çalışma aralığı içinde, ANFIS modelinin sistemin doğrusal olmayan matematiksel modelini tam olarak kopyaladığı, eğitilen çalışma aralığı dışında küçük bir sapma gözlemlendiği görülmüştür. Bu nedenle yazarlar, sistem parametreleri bilinmediğinde veya belirsiz olduğunda deneysel giriş ve çıkış verilerinden sistem tanımlaması için ANFIS kullanmayı önermiştir (Chawla ve Singla, 2019).

Hem Türkçe hem de İngilizce anahtar kelimeler aratılarak konuyla ilgili yapılan literatür taraması sonucunda BMD'nin YSA ile modellenmesi üzerine bulunan çalışmalar yukarıda sunulmuştur. Ayrıca, YSA ile başka denetleyiciler de modellenmiştir. BMD'yi modelleyerek elde ettiğimiz YSA, BMD ile aynı sonuçları daha hızlı bir biçimde üretebilecektir. Böylece daha etkili bir kontrol sonuçları elde edilebilecektir. Ayrıca, BMD'yi modelleyerek elde ettiğimiz YSA'nın çalışma hızı ve hata değerleri BMD ile karşılaştırılacaktır.

3. KURAMSAL TEMELLER

3.1. Bulanık Mantık

Bulanık mantık günlük hayatta kullandığımız sesi biraz aç, kombiyi biraz kıs, çok hızlı gel gibi kesin olmayan dilsel ifadelerin kullanımının sisteme oturtulduğu bir yapıdır. Bulanık mantık sayesinde günlük hayattaki birçok sistemin kontrolü insan zekasına dayalı bir şekilde yapılabilmektedir. Buradaki amaç insan akıl yürütme gücünü kullanarak sistem parametrelerindeki belirsiz durumlar yüzünden oluşabilecek problemlerin önüne geçilerek sistemlerin daha doğru bir karar mekanizması ile yürütülmesini sağlamaktır. Bu sayede insanların daha doğru karar verebilme yeteneği ve sistemlerin daha hızlı çalışabilme özelliği birleştirilerek, sistemlerin daha iyi kontrol performansları verebilmesi amaçlanmaktadır. Bu amaç, kesin olmayan dilsel ifadelerin bulanık mantık yardımıyla kodlama yapılması ile gerçekleştirilmektedir.

3.1.1. Bulanık kümeler

Klasik küme teorisinde, X evrensel kümesinde tanımlanan bir K kümesi için, X evrensel kümesinde yer alan herhangi bir x elemanı K kümesinde bulunmuyor ise $x \notin K$, bulunuyor ise $x \in K$ şeklinde gösterilmektedir. Bulunma durumunu sayısal olarak ifade edebilmek için χ kullanılarak bir fonksiyon gösterimi yapılabilir. Bu durumda; bulunmuyor ise $\chi_K(x) = 0$, bulunuyor ise $\chi_K(x) = 1$ olarak gösterilmektedir (Jezewski ve ark., 2017).

Bulanık mantık teorisinde ise bir küme artık *klasik küme* veya *keskin küme* olarak isimlendirilmektedir. Çünkü klasik kümeden farklı bir küme çeşidi olarak *bulanık küme* tanımlaması yapılmıştır. Benzer şekilde, X evrensel kümesinde tanımlanan bir A bulanık kümesi için, X evrensel kümesinde yer alan herhangi bir x elemanı A bulanık kümesinde tamamen bulunmuyor ise $\mu_A(x) = 0$, tamamen bulunuyor ise $\mu_A(x) = 1$ şeklinde gösterilmektedir. Klasik kümelerde olmayan durum ise herhangi bir x elemanı bir bulanık kümede kısmen bulunabilir. Kısmen bulunma durumunu sayısal olarak ifade edebilmek için yine μ kullanılarak bir fonksiyon gösterimi yapılabilir. Bu durumda X evrensel kümesinde yer alan bir x elemanın A bulanık kümesindeki kısmen bulunma

durumu $\mu_A(x)$ üyelik değeri ile ifade edilebilir ve bu üyelik değeri $[0,1]$ kapalı aralığında yer alan bütün reel sayılar olabilir ($0 \leq \mu_A(x) \leq 1$, $\mu_A(x) \in \mathbb{R}$). Aslında, teorikte 1 değerinden büyük olabilmesine rağmen pratikte neredeyse hiç kullanılmamaktadır (Dernoncourt, 2013).

Bir bulanık küme ayrıca bu şekilde de tanımlanabilir. X evrensel kümesinde tanımlı A bulanık kümesindeki bir x elemanı ve bu elemanın A bulanık kümesindeki $\mu_A(x)$ üyelik değeri sıralı olarak $A = \{(x, \mu_A(x)) | x \in X\}$ şeklinde gösterilebilir (Jezewski ve ark., 2017).

$$A = \sum_{x \in X} \mu_A(x)/x \quad (3.1)$$

Ayrıca ayrık zamanlı bulanık kümeler için Zadeh tarafından kullanılan tanımlama Denklem 3.1’de verilmiştir. Bu tanımlamada yer alan $/$ işareti bir ayırıcıdır. $\sum$ sembolü ise ayrık zamanlı olduğunu belirtmektedir (Jezewski ve ark., 2017).

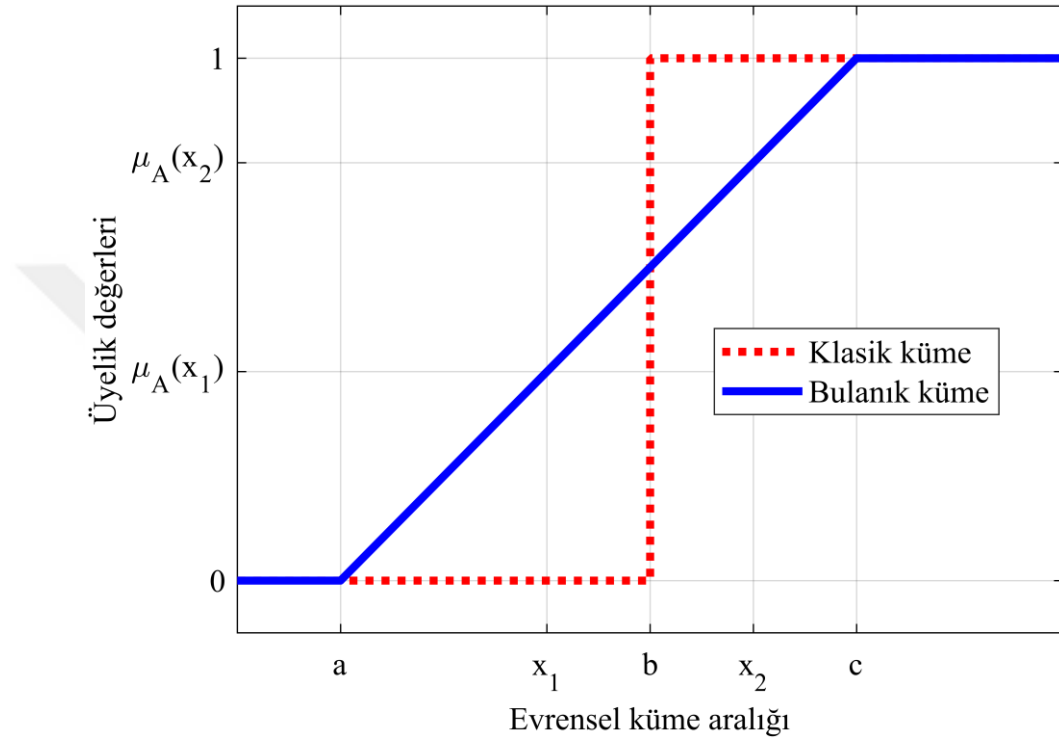
$$A = \mu_A(x_1)/x_1 + \mu_A(x_2)/x_2 + \mu_A(x_3)/x_3 + \dots + \mu_A(x_p)/x_p \quad (3.2)$$

Bu tanımlamanın açık versiyonu ise Denklem 3.2’de verilmiştir. Burada $/$ ve $+$ işaretleri yine asıl kullanımı dışında ayırıcı olarak kullanılmaktadır. Yani A bulanık kümesinde yer alan p tane elemanın kendileri ve üyelik değerleri gösterilmektedir. Bir nevi minik tablo olarak düşünülebilir (Jezewski ve ark., 2017).

$$A = \int_x \mu_A(x)/x \quad (3.3)$$

Bunların yanısıra, sürekli zamanlı bulanık kümeler için Zadeh tarafından kullanılan tanımlama ise Denklem 3.3’te verilmiştir. Yine, bu tanımlamada yer alan $/$ işareti bir ayırıcıdır. $\int$ sembolü ise sürekli zamanlı olduğunu belirtmektedir (Jezewski ve ark., 2017).

Şekil 3.1’de örnek olarak K klasik kümesi ile A bulanık kümesinin grafiği gösterilmiştir. Grafikte x_1 ve x_2 elemanlarının klasik kümedeki karşılığı sırasıyla 0 ve 1 üyelik değerleridir. Ancak bulanık kümedeki karşılığı ise bunların arasında üyelik değerleridir ve bunlar grafikte sırasıyla $\mu_A(x_1)$ ve $\mu_A(x_2)$ olarak gösterilmiştir.



Şekil 3.1. Klasik ve bulanık küme

Şekil 3.1’de örnek olarak verilen K klasik kümesinde evrensel küme aralığında yer alan herhangi bir x elemanı b değerinden küçük olduğunda 0 üyelik değerini, b değeri ve daha büyük olduğunda ise 1 üyelik değerini almaktadır. Bu durum aşağıdaki gibi formülize edilebilir (Jezewski ve ark., 2017).

$$y = \chi_K(x; b) = \begin{cases} 0, & x < b \\ 1, & x \geq b \end{cases} \quad (3.4)$$

Denklem 3.4’te verilen formülde x elemanı kullanılarak y üyelik değeri bulunmaktadır. Yatay eksenindeki evrensel küme aralığının koordinatları kullanılarak oluşturulacak

klasik kümenin şekli belirlenmektedir. Bu şekli oluşturmak için kullanılan tek parametre b değeri sınırı belirlemektedir (Jezewski ve ark., 2017).

Şekil 3.1’de örnek olarak verilen doğrusal artan A bulanık kümesi ise a değeri ve daha küçük değerler için 0 üyelik değerini, c değeri ve daha büyük değerler için 1 üyelik değerini almaktadır. Buna karşın klasik kümelerden farklı olarak a ve c arası değerler için 0 ile 1 arasında üyelik değerleri de alabilmektedir. Ayrıca, bu şekilde olduğu gibi evrensel kümenin en azından bir elemanının üyelik değeri 1 olursa *normal bulanık küme* olur. Denetleyici tasarımlarında genellikle normal bulanık kümeler kullanılmaktadır (Klir ve Yuan, 1995).

$$y = \mu_A(x; a, b, c) = \begin{cases} 0, & x \leq a \\ \frac{x - a}{c - a}, & a < x < c \\ 1, & x \geq c \end{cases} \quad (3.5)$$

Denklem 3.5’te verilen formülde x elemanı kullanılarak y üyelik değeri bulunmaktadır. Yatay eksenindeki evrensel küme aralığının koordinatları kullanılarak oluşturulacak bulanık kümenin şekli belirlenmektedir. Bu şekli oluşturmak için kullanılan parametrelerden a değeri artışın en solunu ve c değeri artışın en sağını belirlemektedir (Klir ve Yuan, 1995).

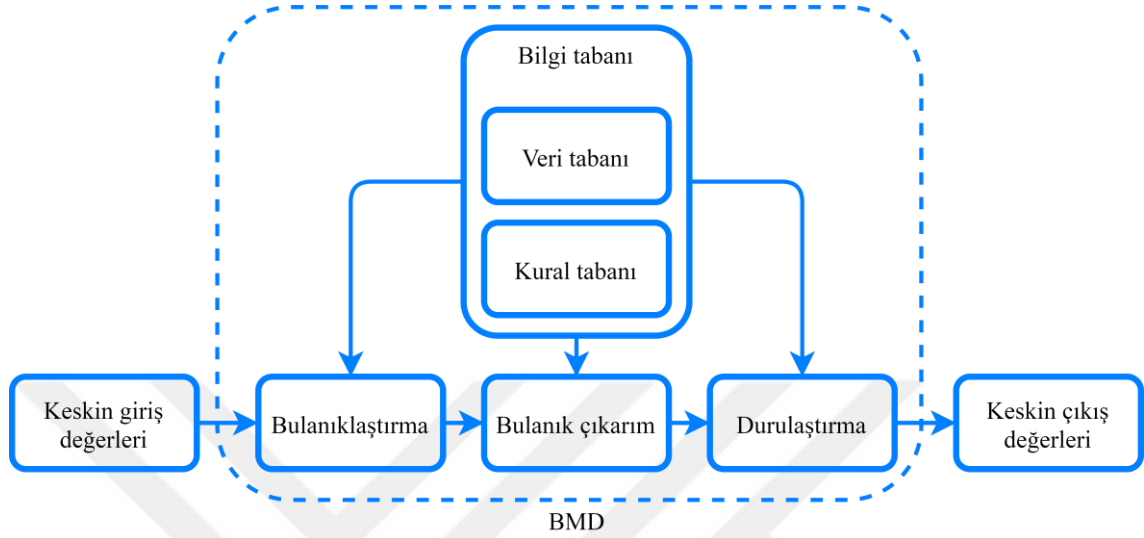
$$\mu_A(x_1) = \frac{x_1 - a}{c - a} \quad (3.6)$$

Şekil 3.1’deki x_1 elemanı için $\mu_A(x_1)$ üyelik değerinin nasıl hesaplandığı Denklem 3.6’da gösterilmiştir. Hesaplama $a < x_1 < c$ şartı sağlandığından bu şarta göre yapılmıştır. Benzer şekilde x_2 elemanının bulanık kümedeki karşılığı $\mu_A(x_2)$ üyelik değeri hesaplanabilir.

3.2. Bulanık Mantık Denetleyici

Bulanık mantık teoremleri kullanılarak bir Bulanık Mantık Denetleyici (BMD) geliştirilebilir. BMD’ler kesin olmayan dilsel ifadeler ile kodlanmış bir algoritma

kullanarak sistemlerin kontrolünü gerçekleştirmektedir. BMD yapısının ana hatları Şekil 3.2’de gösterilmiştir.



Şekil 3.2. BMD yapısının ana hatları

BMD’nin çalışırken kullandığı *bilgi tabanı* iki kısımdan oluşur. Bunlardan birisi *veri tabanı*dır ve içerisinde üyelik fonksiyonları, dilsel ifadeler, seçilen bulanık operatörler, seçilen durulaştırma metodu gibi veriler bulunur. Diğeri ise *kural tabanı*dır ve içerisinde BMD’nin bulanık çıkarım aşamasında karar vermesini sağlayan bulanık kurallar bulunur (Baran ve ark., 2010).

Bulanıklaştırma BMD’nin ilk aşamasıdır ve bu aşamada keskin giriş değerleri üyelik fonksiyonları yardımıyla üyelik değerlerine dönüştürülür. *Bulanık çıkarım* BMD içinde bütün önemli işlemlerin yapıldığı ikinci aşamadır. Bu yüzden bulanık çıkarım aşaması anlatımı üç adıma bölünmüştür. *Ateşleme gücü hesabı* bulanık çıkarım aşamasının ilk adımıdır. Elde edilen bütün üyelik değerleri bu adıma özel ve kurala uygun operatörler kullanılarak her kural için birer tane giriş olabilmesi için birleştirilir ve her bir kuralın ateşleme gücü değeri elde edilir. *Implication* bulanık çıkarım aşamasının ikinci adımıdır. Elde edilen kuralın ateşleme gücü değeri ile ilgili kuralın gösterdiği çıkış üyelik fonksiyonu bu adıma özel ve kurala uygun operatörler ile işlenerek bu sefer her bir kuralın bulanık çıkış kümesi veya değeri elde edilir. *Aggregation* bulanık çıkarım aşamasının üçüncü ve son adımıdır. Elde edilen bütün kuralların bulanık çıkış kümeleri

veya deęerleri bu adıma özel operatörler ile işlenir ve tek bir tane bulanık çıkış kümesi veya deęeri elde edilir. *Durulaştırma* BMD'nin üçüncü ve son aşamasıdır. Elde edilen bulanık çıkış kümesi veya deęeri kullanılarak bir takım yöntemler ile tek bir sayısal deęer olan keskin çıkış deęeri elde edilir. BMD'nin çalışma mantığı takip eden başlıklarda ayrıntılı olarak sunulmuştur (Tóth-Laufer ve ark., 2014).

3.2.1. Dilsel ifadeler

Bir *denetleyici* veya dięer adıyla *kontrolör* tasarlandıktan sonra kontrolün yapılabilmesi için genellikle bir *benzetim sistemi* tasarlanır. Bu benzetim sisteminde denetleyicinin yanı sıra *kontrol edilecek sistem* ve bir takım ek *Simulink blokları* uygun olarak birbirlerine baęlı şekilde bulunur. Genellikle bu benzetim sisteminde denetleyicinin girişine bir *sinyal* verilir. Bu sinyal bir dizi *sayısal deęerlerin* ardı ardına birleştirilmiş halidir. Sayısal deęerler ise $[0,10]$ gibi bir aralıkta bir mesafe deęeri veya $[-20,20]$ gibi bir aralıkta bir sıcaklık deęeri olabilir. Bir BMD için bu tarz sayısal deęerler, *keskin* olarak kabul edilir. Bu durumda bu tarz sayısal deęerler kısaca *keskin deęerler* olarak, denetleyicinin girişine gelen keskin deęerler *keskin giriş deęerleri* olarak, denetleyicinin en son çıkışında ürettięi keskin deęerler ise *keskin çıkış deęerleri* olarak isimlendirilebilir. Bulanıklaştırma işlemi bu keskin giriş deęerlerinin bulanıklaştırılmış karşılıklarını elde edebilmek için yapılır. Bunun için bulanık küme teorisi yardımıyla *dilsel ifadelerle* sayısal işlemler yapmayı mümkün kılan *üyelik fonksiyonları* oluşturulur.

Dilsel ifadeler dięer adıyla *sözel ifadeler* atanan veya atanılan olmak üzere iki çeşittir. Bir girişi temsil etmeye yarayan *dilsel deęişkenler* dięer adıyla *sözel deęişkenler* atanılan, *dilsel terimler* dięer adıyla *sözel terimler* atanan durumdadır. Örneğin *yavaş*, *normal* ve *hızlı* gibi *dilsel terimlerden* herhangi biri *arabanın hızı* ifadesine atanabilir ve bu durumda bu ifade bir *dilsel deęişken* olur (Doęan, 2016). Bir dilsel deęişkene atanabilecek dilsel terimlerin matematiksel olarak bir üyelik deęeri üretebilmeleri için üyelik fonksiyonu ile temsil edilmeleri gerekmektedir. Örneğin *arabanın hızı* dilsel deęişkenine atanabilecek olan *yavaş*, *normal* ve *hızlı* gibi dilsel terimlerin her birinin birer üyelik fonksiyonu ile temsil edilmesi gerekmektedir. Bir üyelik fonksiyonu sensörler aracılığı ile okunan herhangi bir andaki keskin giriş deęeri için çıkış olarak

üyelik değeri üretmektedir. Sade genelleştirilmiş bir anlatım olması açısından birinci girişi temsil etmek için *arabanın hızı* gibi bir *dilsel değişken* yerine kısaca G_1 ifadesi tercih edilmiştir. *Yavaş, normal ve hızlı* gibi *dilsel terimler* yerine ise kısaca A, B ve C ifadeleri tercih edilmiştir. Diğer G_2 girişi için aynı mantık ile D, E ve F ifadeleri tercih edilmiştir. Ayrıca ters bir bakış açısı ile *yavaş değilse, normal değilse ve hızlı değilse* gibi *dilsel terimler* de olabilir. Bunlar ise *bulanık tümleme* işlemi kullanılarak elde edilebilir ve gösterim için kısaca $\bar{A}, \bar{B}$ ve $\bar{C}$ ifadeleri tercih edilmiştir. Diğer giriş için aynı mantık ile $\bar{D}, \bar{E}$ ve $\bar{F}$ ifadeleri tercih edilmiştir.

Gerçek hayat kadar detaylı olmasa da dilsel ifadelerle *bulanık kural* oluşturma mantığını anlatmak için bir örnek olarak Çizelge 3.1 oluşturulmuştur. Burada bir sürücünün trafik ışıklarında ceza yememesi için yapması gereken fren miktarı gösterilmiştir. Burada yer alan ifadelerin tablo gibi verilmesinin sebebi denetleyici tasarımında *kural tabanının* genellikle tablo şeklinde gösterilmesidir.

Çizelge 3.1. Kural tabanı örneği

EĞER ışık kırmızıysa	VE hızım yüksekse	O HALDE fren sert olsun
EĞER ışık kırmızıysa	VE hızım normalse	O HALDE fren normal olsun
EĞER ışık kırmızıysa	VE hızım düşükse	O HALDE fren yumuşak olsun
EĞER ışık yeşilse	VE hızım yüksekse	O HALDE fren yumuşak olsun
EĞER ışık yeşilse	VE hızım yüksek DEĞİLse	O HALDE fren sıfır olsun

Kural tabanında EĞER şart ifadesi ile birbirine bağlanmış iki tane giriş vardır. Bu girişler *ışık* ve *hızım* dilsel değişkenleridir. Birinci giriş için *ışık* dilsel değişkenine *kırmızı* ve *yeşil* dilsel terimlerinden hangisinin atandığı, ikinci giriş için ise *hızım* dilsel değişkenine *düşük, normal ve yüksek* dilsel terimlerinden hangisinin atandığı sorgulanmaktadır. Bazen DEĞİL ifadesi ile atanmadığı da sorgulanabilir. İki giriş VE ile birbirine bağlandığından *bulanık kesişim* işlemi ile birleştirme gerekmektedir. Bazen iki giriş VEYA ile birbirine bağlanırsa *bulanık birleşim* işlemi gerekebilir. Girişler birleştirildiğinde uygun görülen kurallar için sonuç O HALDE kısmı çalıştırılır. Bu durumda çıkış için ayarlanan *fren* dilsel değişkene *sıfır, yumuşak, normal ve sert* dilsel terimlerinden biri atanmış ve uygun cevap üretilmiş olur. *Sözel* kısmı bu şekilde iken

nihayetinde kural tabanını işletecek denetleyici *sayısal* işlemleri dilsel terimleri temsil eden *üyelik fonksiyonları* ile yapmaktadır.

3.2.2. Üyelik fonksiyonları

Bulanık küme teorisinde, bulanık kümedeki bir elemanın bulanıklaştırılmış karşılığı *üyelik fonksiyonları* aracılığıyla bulunmaktadır. Üyelik fonksiyonları matematiksel bir fonksiyon olarak gösterilirler. Gerçek hayattaki keskin giriş değerleri üyelik fonksiyonlarından geçirilerek *üyelik değerleri* bir başka ismiyle *üyelik dereceleri* hesaplanır bu sayede giriş değerlerinin bulanık karşılıkları elde edilmiş olur (Thaker ve Nagori, 2018).

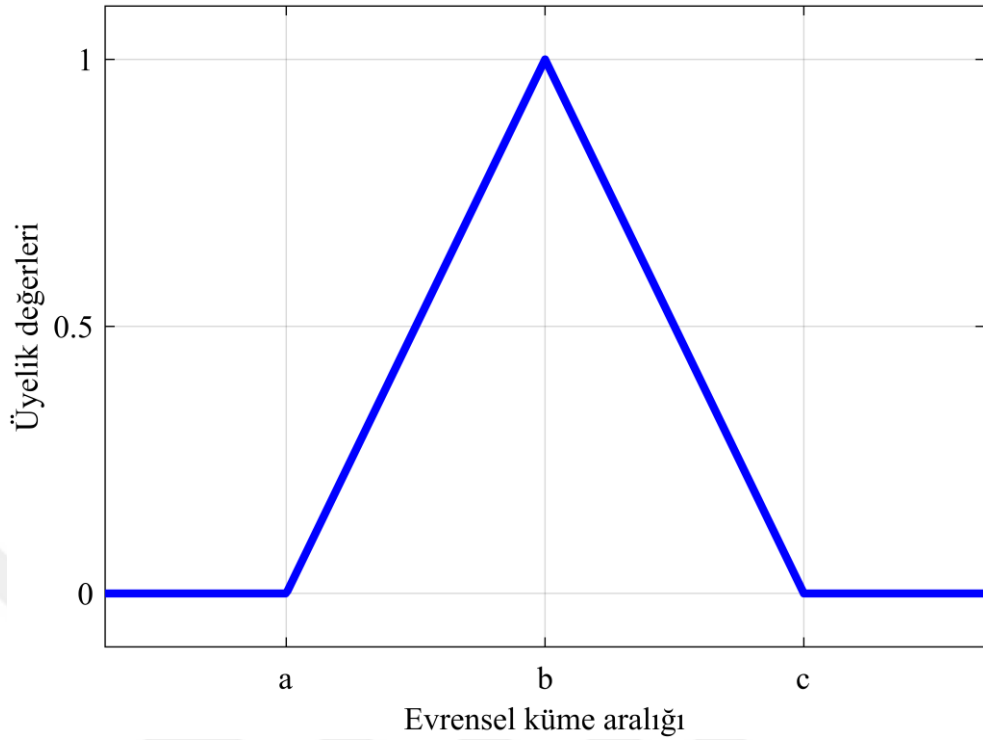
Klasik kümelerde üyelik fonksiyonu sadece dikdörtgen şekli ile oluşturulabilmesine karşın bulanık küme teorisinde üyelik fonksiyonu çeşitli şekillerde oluşturulabilir. BMD tasarımında üyelik fonksiyonlarının seçimi kontrol performansını etkilemektedir (Zhao ve Bose, 2002). En çok kullanılan üyelik fonksiyonlarından birkaç tanesi aşağıda verilmiştir.

Üçgen üyelik fonksiyonu

Üçgen üyelik fonksiyonu diğerlerine göre daha basit bir yapıya sahiptir. Çünkü sadece basit doğrusal matematiksel formüller ile üyelik değerleri hesaplanabilmektedir. MATLAB yazılımında *trimf* olarak kullanıma sunulmuştur.

$$y = \mu(x; a, b, c) = \max\left(0, \min\left(\frac{x - a}{b - a}, \frac{c - x}{c - b}\right)\right) \quad (3.7)$$

Denklem 3.7’de üçgen üyelik fonksiyonu formülü verilmiştir. Burada x keskin giriş değeri kullanılarak y üyelik değeri bulunmaktadır. Yatay eksenindeki evrensel küme aralığının koordinatları kullanılarak oluşturulacak fonksiyonun şekli belirlenmektedir. Bu şekli oluşturmak için kullanılan parametrelerden a değeri üçgenin en solunu, b değeri üçgenin tepe noktasını, c değeri üçgenin en sağını belirlemektedir. Şekil 3.3’te bu parametrelerin yerlerinin gözüktüğü örnek bir üçgen üyelik fonksiyonu gösterilmiştir (MatWorks Help Center, 2021a).



Şekil 3.3. Üçgen üyelik fonksiyonu

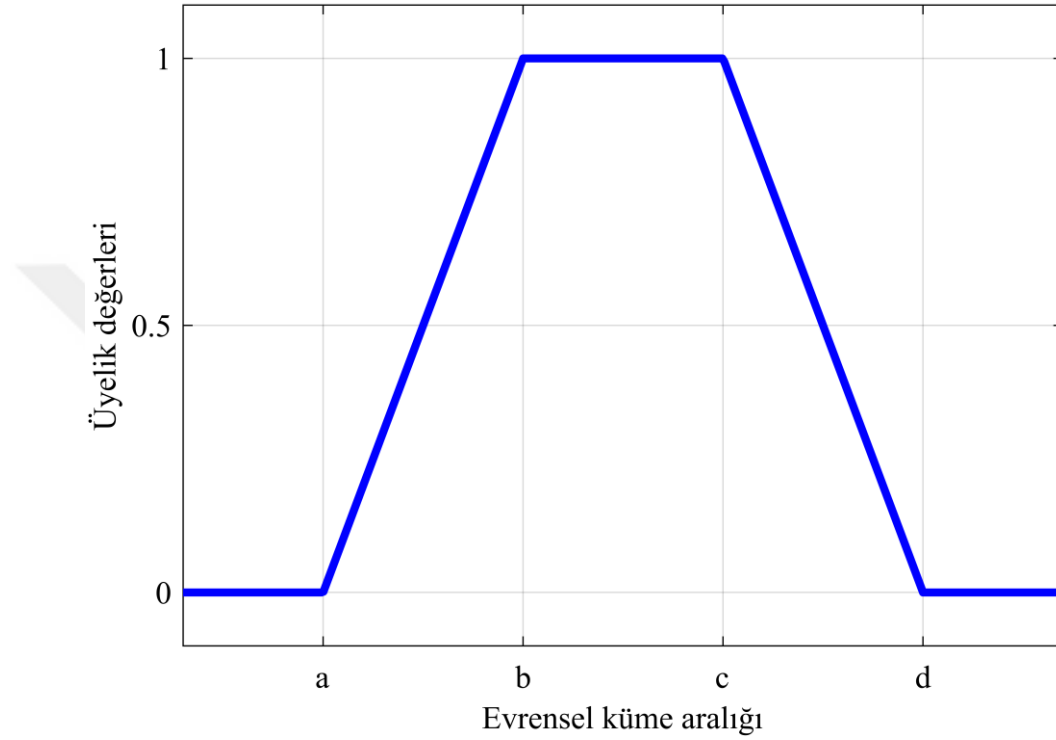
Yamuk üyelik fonksiyonu

Yamuk üyelik fonksiyonu, üçgen üyelik fonksiyonu gibi basit bir yapıya sahiptir. Çünkü sadece doğrusal matematiksel formüllerle üyelik değerleri hesaplanabilmektedir. Bu fonksiyonun orta bölgesi klasik küme gibi davranmakta ve bu bölgedeki bütün üyelik değerleri 1 olmaktadır. Çalışılan aralığın ortasında belirli bir bölgede klasik küme ihtiyacı durumunda tercih edilebilmektedir. MATLAB yazılımında *trapmf* olarak kullanıma sunulmuştur.

$$y = \mu(x; a, b, c, d) = \max\left(0, \min\left(1, \frac{x-a}{b-a}, \frac{d-x}{d-c}\right)\right) \quad (3.8)$$

Denklem 3.8’de yamuk üyelik fonksiyonu formülü verilmiştir. Burada x keskin giriş değeri kullanılarak y üyelik değeri bulunmaktadır. Yatay eksendeki evrensel küme aralığının koordinatları kullanılarak oluşturulacak fonksiyonun şekli belirlenmektedir. Bu şekli oluşturmak için kullanılan parametrelerden a değeri yamuğun en solunu, b

değeri yamuğun tepe noktasının en solunu, c değeri yamuğun tepe noktasının en sağını, d değeri yamuğun en sağını belirlemektedir. Şekil 3.4'te bu parametrelerin yerlerinin gözüktüğü örnek bir yamuk üyelik fonksiyonu gösterilmiştir (MatWorks Help Center, 2021b).



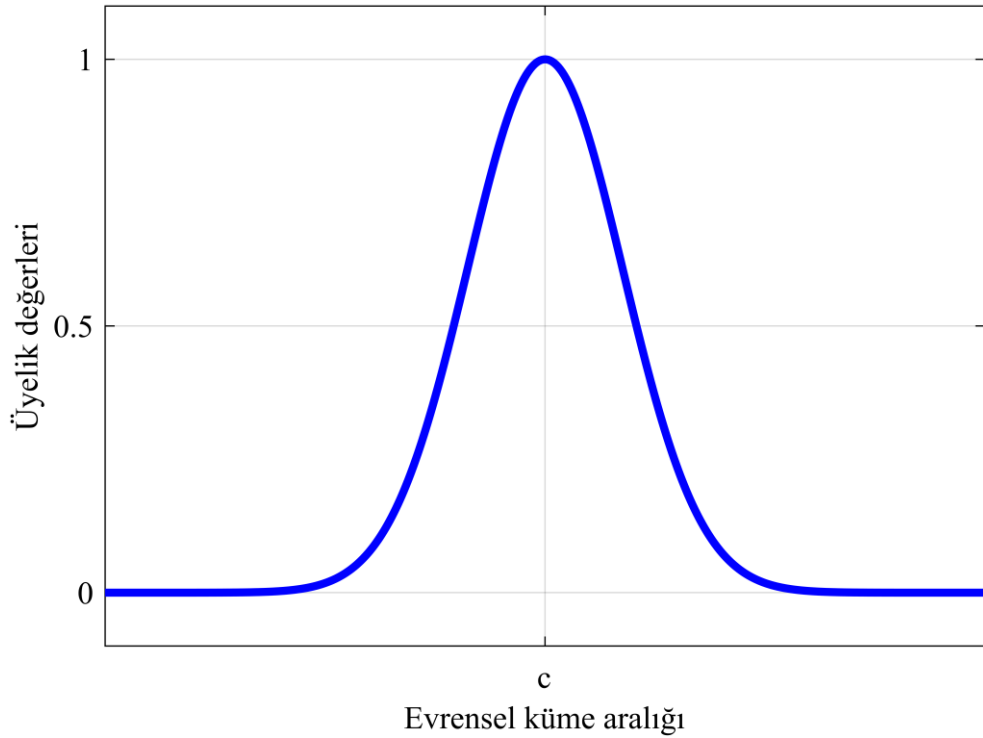
Şekil 3.4. Yamuk üyelik fonksiyonu

Gauss üyelik fonksiyonu

Gauss üyelik fonksiyonunda üstel matematiksel formüller kullanılarak üyelik değerleri hesaplanabilmektedir. Bu fonksiyonun tepe noktası klasik küme ile üçgen üyelik fonksiyonu arasında davranarak üyelik değerleri oluşturmaktadır. Herhangi bir tarafın ortalarında yani üyelik değerinin 0.5 olduğu yerlerde üçgen üyelik fonksiyonuna yakın davranmaktadır. Buna karşın herhangi bir tarafın en uç noktalarında ise üçgen üyelik fonksiyonu gibi ani bir şekilde bitmek yerine kıvrılmaktadır. Bunun sonucu olarak bütün evrensel küme elemanları için çok küçük olsa bile mutlaka bir üyelik değeri hesaplanabilmektedir. MATLAB yazılımında *gaussmf* olarak kullanıma sunulmuştur.

$$y = \mu(x; \sigma, c) = e^{\frac{-(x-c)^2}{2\sigma^2}} \quad (3.9)$$

Denklem 3.9’da Gauss üyelik fonksiyonu formülü verilmiştir. Burada x keskin giriş değeri kullanılarak y üyelik değeri bulunmaktadır. Yatay eksendeki evrensel küme aralığının koordinatları kullanılarak oluşturulacak fonksiyonun şekli belirlenmektedir. Bu şekli oluşturmak için kullanılan parametrelerden c değeri fonksiyonun tepe noktasını belirlemektedir. Diğer σ değeri ise genişliği belirlemektedir bu değer büyüdükçe fonksiyonun genişliği artmaktadır. Şekil 3.5’te bu parametrelerden c değerinin gösterildiği örnek bir Gauss üyelik fonksiyonu gösterilmiştir (MatWorks Help Center, 2021c).



Şekil 3.5. Gauss üyelik fonksiyonu

Sigmoidal üyelik fonksiyonu

Sigmoidal üyelik fonksiyonunda yine üstel matematiksel formüller kullanılarak üyelik değerleri hesaplanabilmektedir. Bu fonksiyonun tepe noktası Gauss üyelik fonksiyonuna göre daha yüksek üyelik değerleri oluşturmaktadır. Üyelik değerinin 0.5

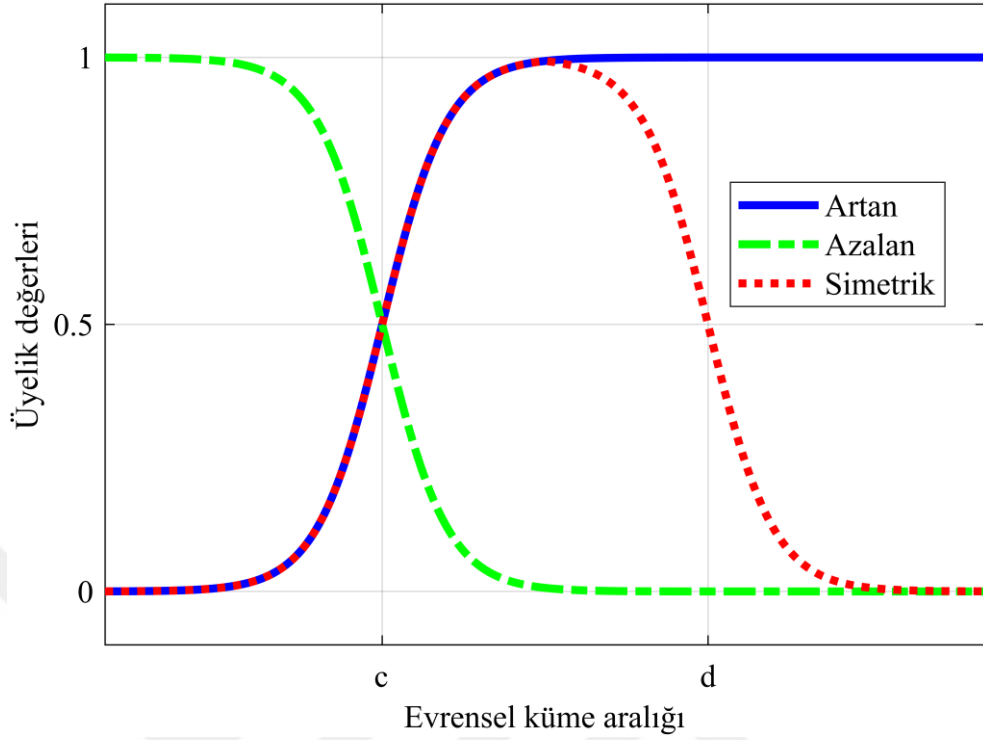
olduğu yerde Gauss üyelik fonksiyonu gibi davranmaktadır. Buna karşın en uç noktalarında ise Gauss üyelik fonksiyonundan daha çok kıvrılmaktadır. Gauss gibi bütün evrensel küme elemanları için çok küçük olsa bile mutlaka bir üyelik değeri hesaplanabilmektedir. MATLAB yazılımında artan tipi *sigmf* olarak kullanıma sunulmuştur.

$$y = \mu(x; a, c) = \frac{1}{1 + e^{-a(x-c)}} \quad (3.10)$$

$$y = \mu(x; a, c) = \frac{1}{1 + e^{-a(c-x)}} \quad (3.11)$$

$$y = \mu(x; a, c, d) = \max\left(0, \min\left(\frac{1}{1 + e^{-a(x-c)}}, \frac{1}{1 + e^{-a(d-x)}}\right)\right) \quad (3.12)$$

Denklem 3.10’da sigmodial üyelik fonksiyonunun genellikle tasarımlarda en sağ üyelik fonksiyonu için tercih edilebilen artan tipinin formülü verilmiştir. Denklem 3.11’de genellikle tasarımlarda en sol üyelik fonksiyonu için tercih edilebilen azalan tipinin formülü verilmiştir. Bu formül $x - c$ yerine $c - x$ kullanarak çok ufak bir değişiklik ile elde edilebilmektedir. Denklem 3.12’de ise genellikle orta üyelik fonksiyonu için tercih edilebilen simetrik tipinin formülü verilmiştir. Bütün formüllerde x keskin giriş değeri kullanılarak y üyelik değeri bulunmaktadır. Yatay eksenindeki evrensel küme aralığının koordinatları kullanılarak oluşturulacak fonksiyonun şekli belirlenmektedir. Bu şekli oluşturmak için kullanılan parametrelerden c değeri fonksiyonun üyelik değerinin 0.5 olduğu tam orta noktasını belirlemektedir. Diğer a değeri ise dikliği belirlemektedir bu değer büyüdükçe fonksiyonun dikliği artmakta çok arttığında ise neredeyse bir klasik küme olmaktadır. Denklem 3.12’de farklı olarak yer alan parametrelerden d değerinin yer aldığı bölüm azalan kısmı temsil ettiğinden orta noktaların çakışmaması için artan kısmın orta noktasını temsil eden c değerinden bir miktar büyük olmalıdır. Şekil 3.6’da artan, azalan ve simetrik tip için aynı c değeri kullanılmıştır. Diğer d değeri ise sadece simetrik tip için kullanılmıştır. Bunlara göre oluşturulan 3 örnek sigmodial üyelik fonksiyonu gösterilmiştir (MatWorks Help Center, 2021d).



Şekil 3.6. Sigmodial üyelik fonksiyonları

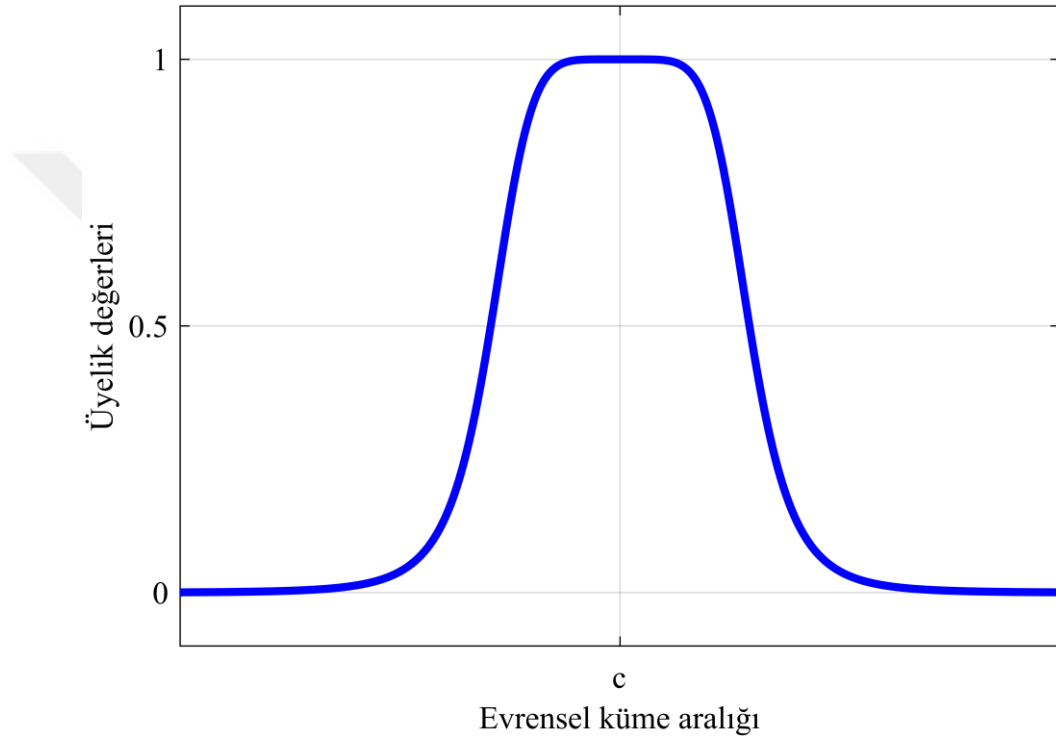
Çan eğrisi üyelik fonksiyonu

Çan eğrisi üyelik fonksiyonunda yine üstel matematiksel formüller kullanılarak üyelik değerleri hesaplanabilmektedir. Bu fonksiyonun tepe noktası neredeyse klasik küme gibi davranmaktadır. Üyelik değerinin 0.5 olduğu yerde Gauss üyelik fonksiyonu gibi davranmaktadır. Buna karşın en uç noktalarında ise Gauss üyelik fonksiyonundan daha çok kıvrılmaktadır. Gauss üyelik fonksiyonu gibi bütün evrensel küme elemanları için çok küçük olsa bile mutlaka bir üyelik değeri hesaplanabilmektedir. MATLAB yazılımında *gbellmf* olarak kullanıma sunulmuştur.

$$y = \mu(x; a, b, c) = \frac{1}{1 + \left| \frac{x-c}{a} \right|^{2b}} \quad (3.13)$$

Denklem 3.13'te Çan eğrisi üyelik fonksiyonu formülü verilmiştir. Burada x keskin giriş değeri kullanılarak y üyelik değeri bulunmaktadır. Yatay eksendeki evrensel küme aralığının koordinatları kullanılarak oluşturulacak fonksiyonun şekli belirlenmektedir.

Bu şekli oluşturmak için kullanılan parametrelerden c değeri çan eğrisinin tepe noktasını belirlemektedir. Diğer a değeri ise genişliği belirlemektedir bu parametre büyüdükçe fonksiyonun genişliği artmaktadır. Son b değeri ise dikliği belirlemektedir bu değer büyüdükçe fonksiyonun dikliği artmakta çok arttığında ise neredeyse bir klasik küme olmaktadır. Şekil 3.7’de bu parametrelerden c değerinin gözüktüğü örnek bir çan eğrisi üyelik fonksiyonu gösterilmiştir (MatWorks Help Center, 2021e).



Şekil 3.7. Çan eğrisi üyelik fonksiyonu

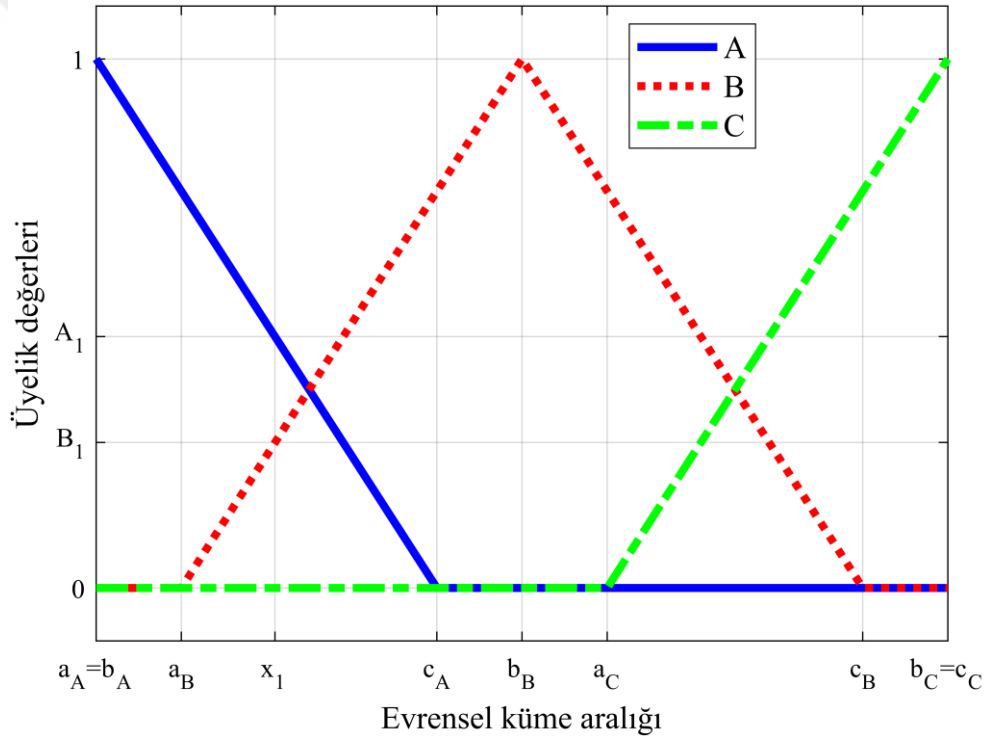
Bütün bu üyelik fonksiyonlarının yanı sıra başka çeşit üyelik fonksiyonları da kullanılabilmektedir.

3.2.3. Bulanıklaştırma

BMD tasarımlarında her giriş için değişen sayıda üyelik fonksiyonu kullanılmaktadır. Genellikle tek sayıda üyelik fonksiyonu tercih edilmektedir. Bir BMD tasarımında 2 giriş değişkeni olduğu durumu örnek olarak incelediğimizde, birinci giriş değişkeni için 3, ikinci giriş değişkeni içinse 7 üyelik fonksiyonu seçilsin. Ve çıkış için 5 üyelik

fonksiyonu seçilmesi durumunda toplamda 15 üyelik fonksiyonunun tanımlanması gerekecektir. Bu üyelik fonksiyonlarının çeşitleri aynı veya farklı olabilmektedir. Üyelik fonksiyonu sayıları da aynı veya farklı olabilmektedir. Uygulamalarda farklı giriş değişkenleri ve çıkış değişkenleri için aynı çeşit aynı sayıda üyelik fonksiyonlarının kullanımı mümkün olabilmektedir. Bu sayede BMD tasarımı kısmen kolaylaşmaktadır.

Üyelik fonksiyonları Şekil 3.8’de görüldüğü gibi giriş ve çıkış için bu şekilde seçilebilir. Bu grafikte birinci giriş için seçilebilecek örnek 3 üyelik fonksiyonu gösterilmiştir.



Şekil 3.8. Birinci giriş için örnek üyelik fonksiyonları

Örnek olarak alınan bu birinci girişi G_1 olarak, bu G_1 girişi için oluşturulan üyelik fonksiyonlarını A , B ve C olarak isimlendirmiştik (bkz. s.16). G_1 girişinden gelen evrensel küme aralığında yer alan x_1 keskin giriş değeri, farklı üyelik fonksiyonlarında farklı üyelik değerleri alacaktır.

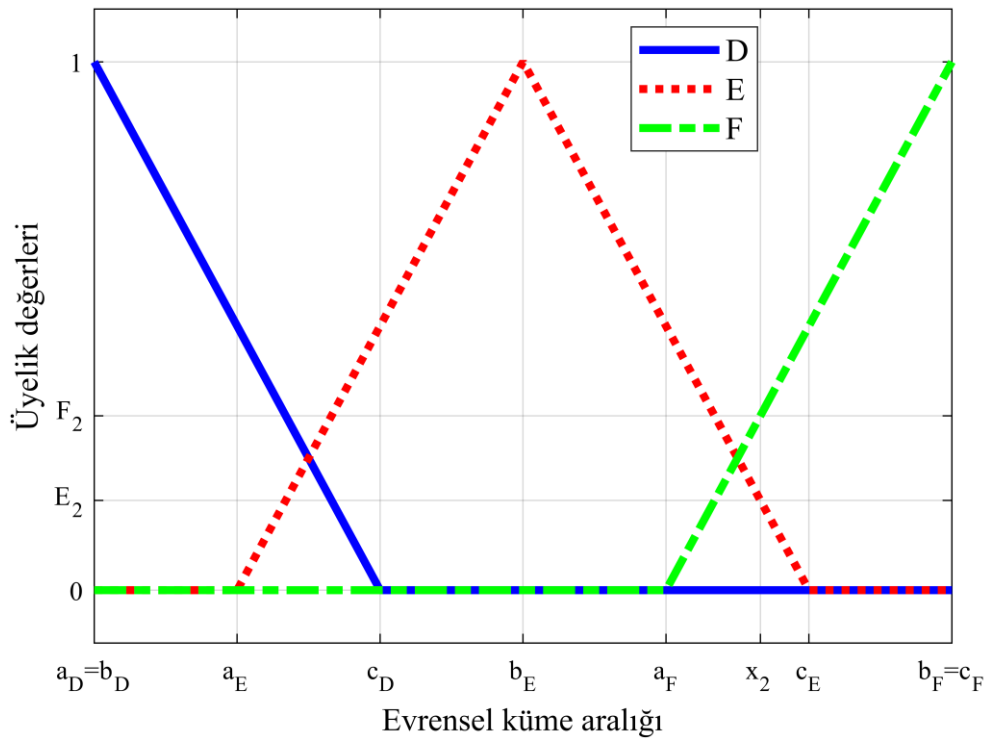
$$A_1 = \mu_A(x_1) = \max\left(0, \min\left(1, \frac{c_A - x_1}{c_A - b_A}\right)\right) \quad (3.14)$$

Denklem 3.14'te x_1 keskin giriş değeri ve A üyelik fonksiyonunu oluşturan a_A , b_A ve c_A değerleri kullanılarak $A_1 = \mu_A(x_1)$ üyelik değerinin nasıl hesaplandığı gösterilmiştir. Buradaki üçgen üyelik fonksiyonu azalan olduğundan $a_A = b_A$ olur.

$$B_1 = \mu_B(x_1) = \max\left(0, \min\left(\frac{x_1 - a_B}{b_B - a_B}, \frac{c_B - x_1}{c_B - b_B}\right)\right) \quad (3.15)$$

Benzer yöntem ile x_1 keskin giriş değeri B üyelik fonksiyonunda $B_1 = \mu_B(x_1)$ üyelik değerini almaktadır. Benzer yöntem ile x_1 keskin giriş değeri C üyelik fonksiyonunda $C_1 = \mu_C(x_1) = 0$ üyelik değerini almaktadır. Ayrıca bu keskin giriş değerinin C üyelik fonksiyonu içinde olmadığı için aldığı 0 üyelik değeri şekilde de görülmektedir.

Şekil 3.9'da ikinci giriş için seçilebilecek örnek 3 üyelik fonksiyonu gösterilmiştir. Aynı tarz üyelik fonksiyonu seçildiği için bu şekiller birbirlerini andırmaktadır.



Şekil 3.9. İkinci giriş için örnek üyelik fonksiyonları

Daha fazla giriş olduğu durumda ise bunun gibi veya daha farklı üyelik fonksiyonlarından oluşan üyelik fonksiyonları grupları gereklidir. Örnek olarak alınan bu ikinci girişi G_2 olarak, bu G_2 girişi için oluşturulan üyelik fonksiyonlarını D , E ve F olarak isimlendirmiştik (bkz. s.16). Örnek olarak sadece bu 2 girişin olduğunu varsayalım. Bu durumda bu girişlerden sensörler aracılığı ile okunan x_1 ve x_2 keskin giriş değerlerinin bulanıklaştırılmış karşılıkları 6 tane olur. Diğer G_1 girişinde olduğu gibi G_2 girişinden gelen evrensel küme aralığında yer alan x_2 keskin giriş değeri, farklı üyelik fonksiyonlarında farklı üyelik değerleri alacaktır.

Grafikte diğer G_2 girişinden gelen x_2 keskin giriş değerinin D üyelik fonksiyonunda $D_2 = \mu_D(x_2) = 0$ üyelik değerini aldığı görülmektedir.

$$E_2 = \mu_E(x_2) = \max\left(0, \min\left(\frac{x_2 - a_E}{b_E - a_E}, \frac{c_E - x_2}{c_E - b_E}\right)\right) \quad (3.16)$$

Denklem 3.16'da x_2 keskin giriş değeri ve E üyelik fonksiyonunu oluşturan a_E , b_E ve c_E değerleri kullanılarak $E_2 = \mu_E(x_2)$ üyelik değerinin nasıl hesaplandığı gösterilmiştir.

$$F_2 = \mu_F(x_2) = \max\left(0, \min\left(\frac{x_2 - a_F}{b_F - a_F}, 1\right)\right) \quad (3.17)$$

Benzer şekilde ile x_2 keskin giriş değeri F üyelik fonksiyonunda $F_2 = \mu_F(x_2)$ üyelik değerini alacaktır. Buradaki üçgen üyelik fonksiyonu artan olduğundan $b_F = c_F$ olur.

Böylelikle bütün *keskin giriş değerleri* için karşılık gelen *üyelik değerleri* elde edilmiş olur (A_1 , B_1 , C_1 , D_2 , E_2 ve F_2).

3.2.4. Bulanık operatörler

Bulanık kümelerde aynı klasik kümelerde olduğu gibi bir takım işlemler yapılmaktadır. Bu işlemler için neredeyse her zaman $[0,1]$ kapalı aralığındaki bulanık kümeler kullanılmaktadır. Klasik küme teorisinin operatörlerini bulanık mantığın belirli üyelik değeri bulma işlemlerine uyacak şekilde yeniden tanımlayabiliriz. Bulanık kümelerin

özellikleri genellikle aynı olsa da bunların sonuç üretebilmesi için işlenmesi klasik kümelerden daha farklı operatörler kullanılarak yapılmaktadır.

Klasik kümelerde olan *kesişim*, *birleşim* ve *tümleme* işlemleri bulanık kümelere göre uyarlanmak istenildiğinde bu işlemleri yapmak için farklı operatörler önerilmiştir. Bunlar genellikle *bulanık operatörler* olarak isimlendirilmektedir.

Bulanık operatörlerin *bulanık kesişim*, *bulanık birleşim* ve *bulanık tümleme* işlemleri için en yaygın kullanılanları Çizelge 3.2’de verilmiştir. Sensörler ile okunan x_1 ve x_2 keskin giriş değerlerinin A ve E bulanık kümelerindeki karşılıkları sırasıyla A_1 ve E_2 üyelik değerleri olsun ($A_1 = \mu_A(x_1)$ ve $E_2 = \mu_E(x_2)$). Bu durumda sonuçta bir üyelik değeridir ancak sadece x_1 ve x_2 keskin giriş değerleri kullanmak yerine bütün değerler için işlem yapılırsa sonuçta bulanık küme olacaktır.

Çizelge 3.2. Bulanık operatörler

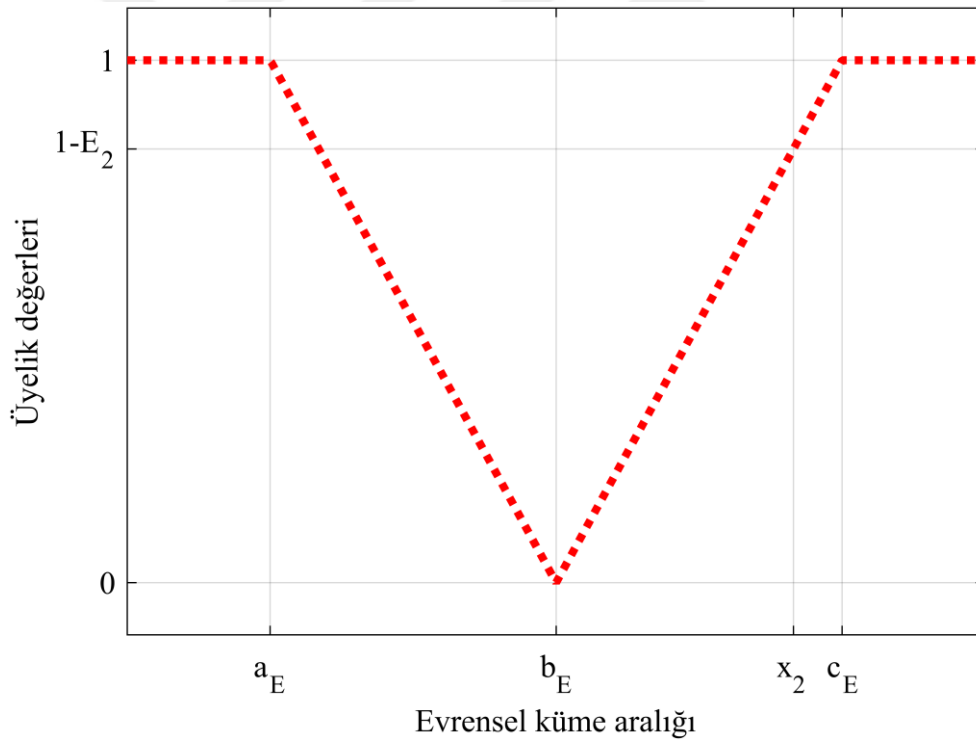
İsim	Bulanık kesişim $\mu_{A \cap E}$	Bulanık birleşim $\mu_{A \cup E}$	Bulanık tümleme $\mu_{\bar{E}}$
Zadeh operatörleri <i>min, max</i>	$\min(A_1, E_2)$	$\max(A_1, E_2)$	$1 - E_2$
Olasılıksal operatörler <i>prod, probor</i>	$A_1 \times E_2$	$A_1 + E_2 - A_1 \times E_2$	$1 - E_2$

Bulanık operatörlerin alışılmış tanımları ile her zaman *değişme*, *birleşme* ve *dağılma* özelliklerini buluruz. Bununla birlikte, iki önemli istisna vardır. Birincisi bir bulanık kümenin kendi tümleyeni ile yaptığı birleşim kümesi her zaman evrensel kümeye eşit değildir ($A \cup \bar{A} \neq X$). Dolayısıyla bir x elemanının bu birleşim kümesindeki üyelik değeri her zaman 1 değildir ($\mu_{A \cup \bar{A}}(x) \neq 1$). Benzer olarak bir bulanık kümenin kendi tümleyeni ile yaptığı kesişim kümesi her zaman boş kümeye eşit değildir ($A \cap \bar{A} \neq \emptyset$). Dolayısıyla bir x elemanının bu kesişim kümesindeki üyelik değeri her zaman 0 değildir ($\mu_{A \cap \bar{A}}(x) \neq 0$). Bu eşitliklerin sağlanmamasının sebebi aynı anda hem A hem de $\bar{A}$ bulanık kümelerine kısmen ait bir eleman olabilir (Dernoncourt, 2013).

Bulanık çıkarım aşamasında *bulanık operatörler* kullanılarak belirlenen kurallara göre çıkarım yapılmaktadır. Denetleyici tasarlanırken bulanık çıkarımın hangi adımında hangi bulanık operatörün kullanılacağı belirlenmektedir.

Bulanık tümlleme operatörleri

Girişler her zaman pozitif ifadelerden oluşmayabilirler. İhtiyaç halinde negatif ifadeleri elde edebilmek için Zadeh, Sugeno ve Yager'in *bulanık tümlleme* operatörleri kullanılmaktadır. Dilsel ifadeler bölümünde örnek olarak gösterilen $\bar{A}$, $\bar{B}$ ve $\bar{C}$ bulanık küme tümlleyenleri *bulanık tümlleme* operatörleri ile elde edilmektedir. Aynı şekilde ikinci giriş için yine değil ifadeleri kullanılabilir ve kısaca $\bar{D}$, $\bar{E}$ ve $\bar{F}$ olarak gösterilebilir (bkz. s.16).



Şekil 3.10. Bulanık tümlleme operatörünün çıkışı

Şekil 3.10'da örnek olarak E üyelik fonksiyonu kullanılarak Zadeh'in *bulanık tümlleme* operatörü ile elde edilmiş $\bar{E}$ üyelik fonksiyonu gösterilmiştir. Her zaman olduğu gibi G_2 girişinden gelen x_2 keskin giriş değerinin, $\bar{E}$ üyelik fonksiyonundaki bulanık karşılığı

$\bar{E}_2$ üyelik değeri olur ($\bar{E}_2 = \mu_{\bar{E}}(x_2) = 1 - E_2$). Zadeh'in *bulanık tümleme* operatörü kullanıldığında E_2 üyelik değerini 1 değerinden çıkartmak, $\bar{E}_2$ üyelik değerini bulmak için yeterlidir. Buradaki E_2 üyelik değerinin nasıl hesaplandığı ise önceden gösterilmişti (bkz. Denklem 3.16). Bazı örnek giriş değerleri için Zadeh'in *bulanık tümleme* operatörü ile hesaplanan çıkış değerleri Çizelge 3.3'te gösterilmiştir.

Bulanık kesişim operatörleri

Bulanık kesişim işlemi için genellikle Minimum (*min*) veya Cebirsel Çarpım (*prod*) operatörleri kullanılmaktadır. Bu operatörler bulanıklaştırma aşamasında ve *Implication* adımıyla kullanılmaktadır. Bir tasarımda karşımıza çıkabilecek örnek bulanık küme veya değerler bu operatörlerin girişi olarak kullanılmış ve anlatımlar adımlara özel olarak yapılmıştır.

Min operatörü

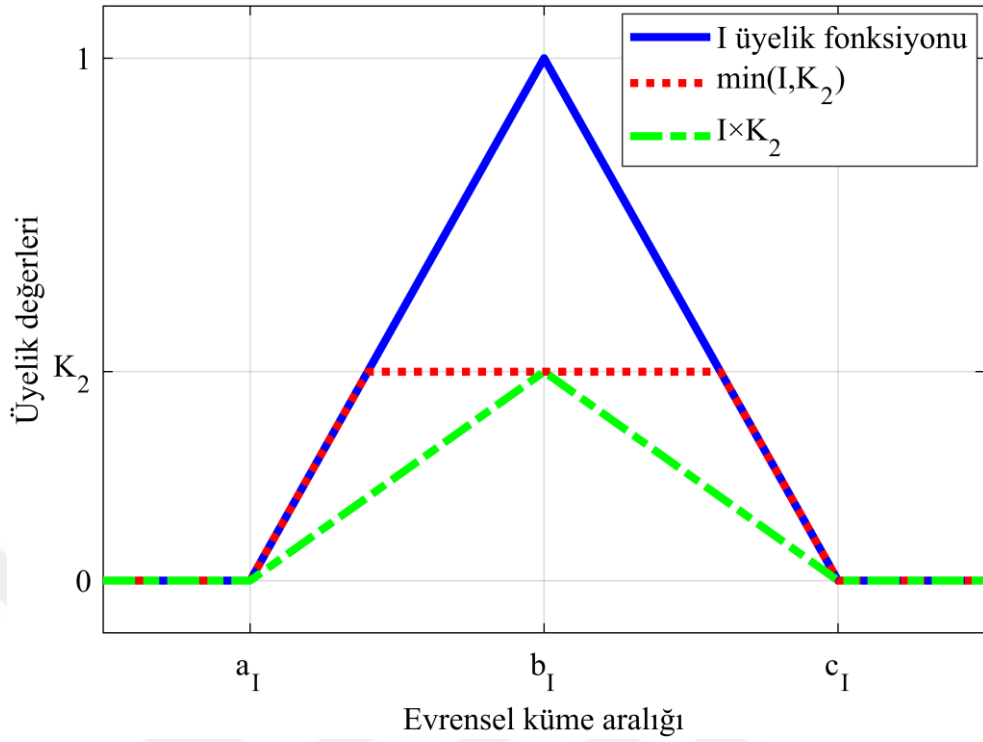
Bütün metotlarda *ateşleme gücü hesabı* adımıyla birer sayıdan ibaret olan A_1 ve E_2 üyelik değerleri için *min* operatörü ile işlem yapılması gerektiği zaman ilgili kuralın *ateşleme gücü* değeri $\min(A_1, E_2)$ ile hesaplanabilir. Hesaplama iki sayısal değer ile yapıldığı için basitçe bu sayıların en küçüğü sonuç olarak kabul edilir. Bazı örnek giriş değerleri için *min* operatörü ile hesaplanan çıkış değerleri Çizelge 3.3'te gösterilmiştir.

Mamdani metodunda *Implication* adımıyla bir sayıdan ibaret olan 2. kuralın *ateşleme gücü* (K_2) değeri ile ilgili kuralın gösterdiği I çıkış üyelik fonksiyonu için *min* operatörü ile işlem gerektiği zaman ilgili kuralın bulanık çıkış kümesi $\min(I, K_2)$ ile hesaplanabilir. Hesaplama için I çıkış üyelik fonksiyonunun evrensel küme aralığındaki her bir değer kullanılır. Her bir değer ve K_2 değeri kıyaslanır ve en küçük olan sonuç olarak kabul edilir. Yani başka bir deyişle K_2 değerini evrensel kümenin bütün noktalarında aynı olan yatay bir çizgi olarak kabul edip bütün noktalarda en küçük değeri sonuç olarak kabul eder. Bu girişler için *min* operatörü seçilmesi durumunda hesaplanacak 2. kuralın bulanık çıkış kümesi Şekil 3.11'de gösterilmiştir.

Prod operatörü

Bütün metotlarda *ateşleme gücü hesabı* adımı birer sayıdan ibaret olan A_1 ve E_2 üyelik değerleri için *prod* operatörü ile işlem yapılması gerektiği zaman ilgili kuralın ateşleme gücü değeri $A_1 \times E_2$ ile hesaplanabilir. *Sugeno* ve *Tsukamoto* metotlarında *Implication* adımı birer sayıdan ibaret olan I ve K_2 değerleri için *prod* operatörü ile işlem yapılması gerektiği zaman ilgili kuralın bulanık çıkış değeri $I \times K_2$ ile hesaplanabilir. İki hesaplamada iki sayısal değer ile yapıldığı için basitçe bu sayıların çarpımı sonuç olarak kabul edilir. Bazı örnek giriş değerleri için *prod* operatörü ile hesaplanan çıkış değerleri Çizelge 3.3'te gösterilmiştir.

Mamdani metodunda *Implication* adımı K_2 değeri ile o kuralın gösterdiği I çıkış üyelik fonksiyonu için *prod* operatörü ile işlem gerektiği zaman ilgili kuralın bulanık çıkış kümesi $I \times K_2$ ile hesaplanabilir. Hesaplama için I üyelik fonksiyonunun evrensel küme aralığındaki her bir değer, K_2 değeri ile çarpılır. Bu girişler için *prod* operatörü seçilmesi durumunda hesaplanacak 2. kuralın bulanık çıkış kümesi Şekil 3.11'de gösterilmiştir.



Şekil 3.11. Bulanık kesişim operatörlerinin çıkışları (Mamdani metodu)

Bulanık birleşim operatörleri

Bulanık birleşim işlemi için genellikle Maksimum (*max*), Cebirsel Toplam (*probor*) veya Toplam (*sum*) operatörleri kullanılmaktadır. Bu operatörler bulanıklaştırma aşamasında ve *Aggregation* adımıyla kullanılmaktadır. Bir tasarımda karşımıza çıkabilecek örnek bulanık küme veya değerler bu operatörlerin girişi olarak kullanılmış ve anlatımlar adımlara özel olarak yapılmıştır.

Max operatörü

Bütün metotlarda *ateşleme gücü hesabı* adımıyla birer sayıdan ibaret olan A_1 ve E_2 üyelik değerleri için *max* operatörü ile işlem yapılması gerektiği zaman ilgili kuralın ateşleme gücü değeri $\max(A_1, E_2)$ ile hesaplanabilir. Hesaplama iki sayısal değer ile yapıldığı için basitçe bu sayıların en büyüğü sonuç olarak kabul edilir. Bazı örnek giriş değerleri için *max* operatörü ile hesaplanan çıkış değerleri Çizelge 3.3'te gösterilmiştir.

Mamdani metodunda *Aggregation* adımında ζ_H , ζ_I ve ζ_J bulanık kümeleri için *max* operatörü ile işlem gerektiği zaman bulanık çıkış kümesi $\max(\zeta_H, \zeta_I, \zeta_J)$ ile hesaplanabilir. Bu bulanık kümelerin evrensel küme aralığındaki her bir değer için en büyük olanlar sonuç olarak kabul edilir. Örnek bir *max* operatörü bulanık çıkış kümesi Şekil 3.13'te gösterilmiştir.

Probor operatörü

Bütün metotlarda *ateşleme gücü hesabı* adımında birer sayıdan ibaret olan A_1 ve E_2 üyelik değerleri için *probor* operatörü ile işlem yapılması gerektiği zaman ilgili kuralın ateşleme gücü değeri $A_1 + E_2 - A_1 \times E_2$ ile hesaplanabilir. Hesaplama iki sayısal değer ile yapıldığı için basitçe bu sayıların çarpımını toplamından çıkartıp sonuç olarak kabul edilir. Bazı örnek giriş değerleri için *probor* operatörü ile hesaplanan çıkış değerleri Çizelge 3.3'te gösterilmiştir.

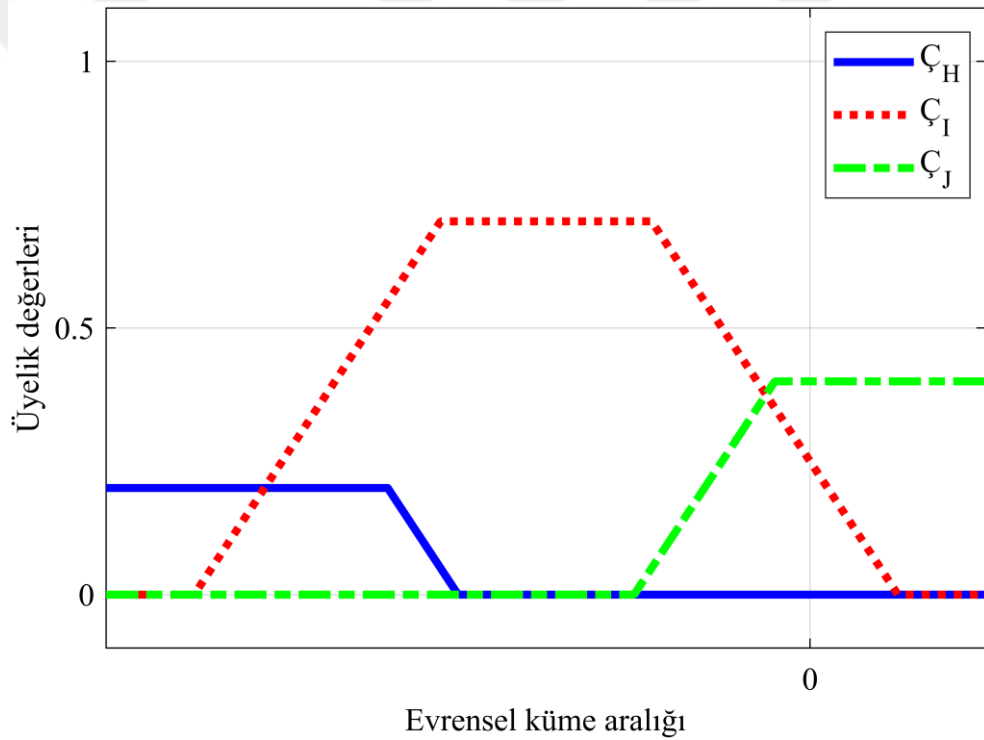
Mamdani metodunda *Aggregation* adımında ζ_H , ζ_I ve ζ_J bulanık kümeleri için *probor* operatörü ile işlem gerektiği zaman bulanık çıkış kümesi $\text{probor}(\zeta_H, \zeta_I, \zeta_J)$ ile hesaplanabilir. İlk olarak ζ_H ve ζ_I bulanık kümelerinin evrensel küme aralığındaki bütün değerleri için çarpımı toplamdan çıkartıp sonuç olarak kabul eder. Bu yeni bulanık kümeyi ζ_{HI} olarak isimlendirelim. Bu durumda ikinci gerekli işlem $\text{probor}(\zeta_{HI}, \zeta_J)$ yapılması gerekmektedir ve buradan sonuç bulunur. Bu aslında genel bir kuraldır. Ancak normalde yapılan bu ikiyeşerli işlemler atlanmıştır. Çünkü *max* ve *sum* operatörleri için $\max(\zeta_H, \zeta_I, \zeta_J) = \max(\max(\zeta_H, \zeta_I), \zeta_J)$ ve $\zeta_H + \zeta_I + \zeta_J = (\zeta_H + \zeta_I) + \zeta_J$ eşitlikleri sağlanmaktadır. *Probor* operatörü için ise bu özellik sağlanmadığı için işlemlerin ikiyeşerli olarak yapılması gerekmektedir (MatWorks Help Center, 2021f). Örnek bir *probor* operatörü bulanık çıkış kümesi Şekil 3.13'te gösterilmiştir.

Sum operatörü

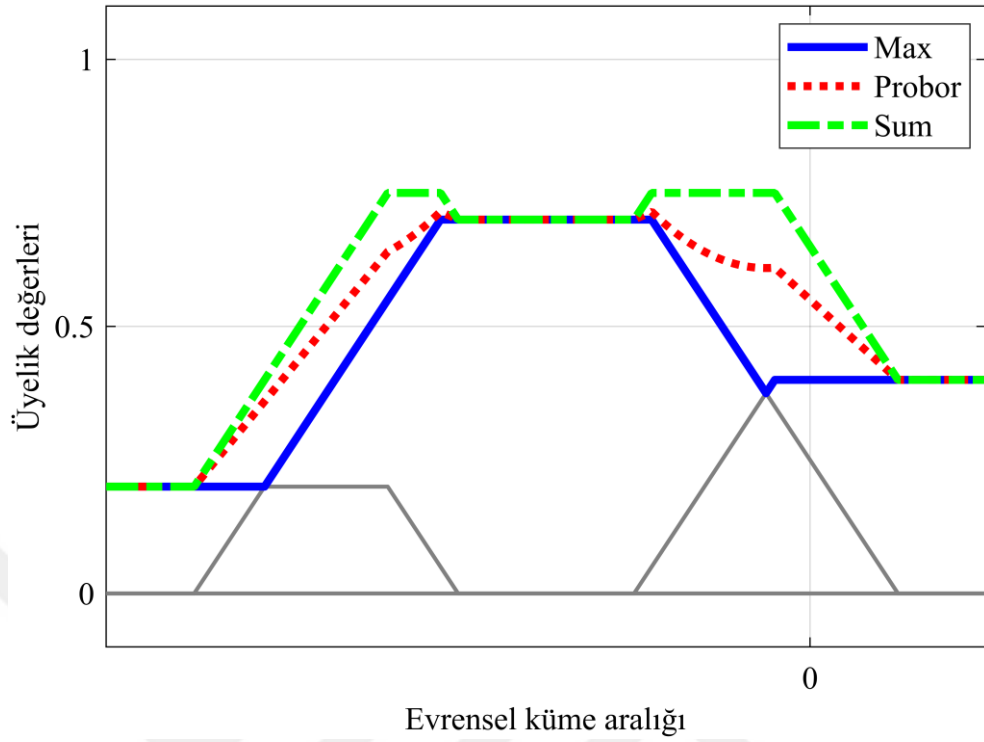
Sugeno ve *Tsukamoto* metotlarında *Aggregation* adımında birer sayıdan ibaret olan K_{1H} , K_{2I} , K_{3H} , K_{4I} , K_{5I} , K_{6H} , K_{7H} , K_{8I} ve K_{9J} değerleri için *sum* operatörü ile işlem

yapılması gerektiği zaman bulanık çıkış değeri $K_{1H} + K_{2I} + K_{3H} + K_{4I} + K_{5I} + K_{6H} + K_{7H} + K_{8I} + K_{9J}$ ile hesaplanabilir. Hesaplama sayısal değerler ile yapıldığı için basitçe bu sayıların toplamı sonuç olarak kabul edilir.

Mamdani metodunda *Aggregation* adımında ζ_H , ζ_I ve ζ_J bulanık kümeleri için *sum* operatörü ile işlem gerektiği zaman bulanık çıkış kümesi $\zeta_H + \zeta_I + \zeta_J$ ile hesaplanabilir. Bu bulanık kümelerin evrensel küme aralığındaki her bir değer için toplamlar sonuç olarak kabul edilir. Örnek bir *sum* operatörü bulanık çıkış kümesi Şekil 3.13'te gösterilmiştir.



Şekil 3.12. Ara bulanık çıkış kümeleri (Mamdani metodu)



Şekil 3.13. Bulanık birleşim operatörlerinin çıkışları (Mamdani metodu)

Çizelge 3.3. Bazı girişler için operatörlerin çıkışları

	Operatör girişleri		Bulanık tümlleme	Bulanık kesişim		Bulanık birleşim	
	A_1	E_2	$\bar{E}_2$ (Zadeh)	Min	$Prod$	Max	$Probor$
Klasik küme	0	0	1	0	0	0	0
	0	1	0	0	0	1	1
	1	1	0	1	1	1	1
Bulanık küme	0.2	0.2	0.8	0.2	0.04	0.2	0.36
	0.2	0.4	0.6	0.2	0.08	0.4	0.52
	0.2	0.6	0.4	0.2	0.12	0.6	0.68
	0.2	0.8	0.2	0.2	0.16	0.8	0.84
	0.4	0.4	0.6	0.4	0.16	0.4	0.64
	0.4	0.6	0.4	0.4	0.24	0.6	0.76
	0.4	0.8	0.2	0.4	0.32	0.8	0.88
	0.6	0.6	0.4	0.6	0.36	0.6	0.84
	0.6	0.8	0.2	0.6	0.48	0.8	0.92
	0.8	0.8	0.2	0.8	0.64	0.8	0.96

3.2.5. Kural tabanı

Oluşturulan bütün üyelik fonksiyonlarının çıkışında elde edilmiş üyelik değerleri *bulanık kurallar* aracılığı ile işlenecektir. Üyelik fonksiyonlarının sayısı ile ilişkili olan kural sayısının artması ise çoğu durumda daha iyi kontrol sonuçlarının elde edilmesini sağlamaktadır. Ancak kural sayısının artması bulanık işlem sayısını artırdığından BMD'nin yazılımsal ve donanımsal olarak gerçekleştirilmesi zorlaşmaktadır. Kurallar genellikle tablo halinde gösterilir ve *kural tabanı* olarak isimlendirilir. Kurallar oluşturulurken bir mantık çerçevesinde oluşturulması gerekmektedir (bkz. Çizelge 3.1). Bu yüzden uzman tecrübesi yardımı ile kurallar oluşturulmaktadır. Ancak buradaki örnek kural tablosu anlatımda bütün olası girişlerdeki hesaplamaları görebilmek açısından uygun olacak bir şekilde oluşturulmuştur.

Bulanık çıkarım aşamasında kural tabanında işlem yapılırken bulanık operatörler tercihe göre kullanılmaktadır. Genellikle her bir bulanık işlem için uygun bir bulanık operatör tercih edilmekte bu da BMD tasarımını kısmen kolaylaştırmaktadır. Çizelge 3.4'te örnek bir kural tabanı gösterilmiştir. Burada *bulanık kesişim* işlemi için *prod* operatörü, *bulanık birleşim* işlemi için ise *max* operatörü tercih edilmiştir.

Çizelge 3.4. Kural tabanı

Kurallar	Girişler		Çıkış	Kural ağırlığı	Bulanık operatör
	G_1	G_2	ζ		
Kural 1	A	D	H	w_1	<i>Prod</i>
Kural 2	A	E	I	w_2	<i>Prod</i>
Kural 3	A	F	H	w_3	<i>Prod</i>
Kural 4	B	D	I	w_4	<i>Prod</i>
Kural 5	B	$\bar{E}$	I	w_5	<i>Max</i>
Kural 6	$\bar{B}$	F	H	w_6	<i>Prod</i>
Kural 7	C	D	H	w_7	<i>Prod</i>
Kural 8	$\bar{C}$	$\bar{E}$	I	w_8	<i>Max</i>
Kural 9	C	F	J	w_9	<i>Max</i>

Çizelge 3.4'te gösterilen kural tabanında 2 farklı giriş bulunmaktadır. Girişler Bulanıklaştırma aşamasında anlatılan örnek üyelik fonksiyonları ile aynı seçilmiştir. O grafikte örnek bir G_1 girişi için seçilebilecek 3 üyelik fonksiyonu gösterilmiş ve bunlar

A , B ve C olarak isimlendirilmişti (bkz. Şekil 3.8). Ayrıca diğer G_2 girişi için seçilebilecek 3 üyelik fonksiyonu gösterilmiş ve bunlar da D , E ve F olarak isimlendirilmişti (bkz. Şekil 3.9). Çizelgede yeni olarak $\checkmark$ çıkışı için ise yine 3 farklı üyelik fonksiyonu oluşturulmuş ve bunlar H , I ve J olarak isimlendirilmiştir. Bunlar girişlerde olduğu gibi $\checkmark$ çıkışını nitelendirmek için kullanılan dilsel terimlerdir. Her bir *kuralın ateşleme gücü* değeri için her kurala özel olmak üzere genellikle $[0,1]$ arasında olmak üzere *kural ağırlıkları* ($w_1, w_2, \dots, w_9$) belirlenebilir. Ancak ağırlık istenmediği durumlarda çarpım durumunda olan bütün bu ağırlıklar 1 yapılır ve bu işlem adımı atlanmış olur. Ayrıca her kural için kullanılacak bulanık operatör de kural tabanında gösterilmiştir. Böyle girişler ve böyle üyelik fonksiyonları olması durumunda genellikle G_1 girişindeki üyelik fonksiyonları ile G_2 girişindeki üyelik fonksiyonlarının sayısının çarpımı kadar kural oluşturulmaktadır. Böylece her türlü giriş olasılığı için BMD'nin istenen çıkış cevabını her zaman verebilmesi hedeflenmektedir.

3.2.6. Bulanık çıkarım

Bu kurallar ile bulanık çıkarım işlemleri başlıca üç adımda yapılmaktadır. Bulanık çıkarım işlemleri esnasında tek giriş olması durumunda bulanık çıkarımın ilk adımı gerekli değildir. Ancak 2 veya daha fazla giriş olması durumunda bulanıklaştırma aşamasında elde edilmiş üyelik değerleri giriş olarak kullanılarak oluşturulan her kural değerlendirilir ve çıkışta kuralların ateşleme gücü değerleri elde edilir. Bu değerler kural sayısı kadardır. Bulanık çıkarımın ikinci adımı bütün kurallar için birer adet oluşturulan bu değerler kullanılarak yapılmaktadır. Bunun sonucunda kuralların bulanık çıkış kümeleri veya değerleri elde edilmektedir. Üçüncü ve son adımda ise bu kümeler veya değerler kullanılarak tek bir tane bulanık çıkış kümesi veya değeri bulunacaktır.

Ateşleme gücü hesabı

Ateşleme gücü hesabı bulanık çıkarım aşamasının ilk adımıdır. Çizelge 3.4'te verilen kural tablosu ile işlemler yapılacaktır. *Kural 1* için detaylı bir şekilde verilirken diğer kurallar için ise gerektiği kadar detay verilecektir. Tabloda olmayan bazı harf atamaları Bulanıklaştırma aşamasında olduğu gibi yapılacaktır (bkz. Şekil 3.8 ve Şekil 3.9).

Kural 1: EĞER (G_1, A ise) VE (G_2, D ise) O HALDE ($\checkmark, H$ olsun).

EĞER ile şart konulmuştur. Bağlaç VE olduğundan *bulanık kesişim* işlemi yapılacaktır. Sonuç O HALDE ile belirtilmiştir.

Kural 1: Bu kuralda G_1 girişinden gelen x_1 keskin giriş değeri kullanılarak A üyelik fonksiyonu aracılığı ile bulanıklaştırma aşamasında elde edilmiş $A_1 = \mu_A(x_1)$ üyelik değeri ile G_2 girişinden gelen x_2 keskin giriş değeri kullanılarak D üyelik fonksiyonu aracılığı ile elde edilmiş $D_2 = \mu_D(x_2)$ üyelik değeri, kuralın operatörü *prod* olduğu için çarpılacaktır (bkz. s.31). Buradan çıkan sonuç ise operatörden bağımsız olarak o kuralın etkisini belirleyebilmek için w_1 kural ağırlığı ile çarpılır ($K_1 = A_1 \times D_2 \times w_1$) ve 1. *kuralın ateşleme gücü (K_1) değeri* elde edilir. Bu değer kuralın indisi ile uyumlu olacak şekilde isimlendirilmiştir.

Sıradaki kurallar için işlemler aynı mantık ile yapıldığında her bir kuralın ateşleme gücü değeri bulunur ($K_2 = A_1 \times E_2 \times w_2$, $K_3 = A_1 \times F_2 \times w_3$ ve $K_4 = B_1 \times D_2 \times w_4$).

Kural 5: EĞER (G_1, B ise) VEYA (G_2, E DEĞİLse) O HALDE ($\checkmark, I$ olsun).

EĞER ile şart konulmuştur. G_2 girişine DEĞİL olduğu için *bulanık tümleme* işlemi yapılacaktır. Bağlaç VEYA olduğundan *bulanık birleşim* işlemi yapılacaktır. Sonuç O HALDE ile belirtilmiştir.

Kural 5: B_1 üyelik değeri ile $\bar{E}_2$ üyelik değeri, kuralın operatörü *max* olduğu için maksimumu alınacaktır (bkz. s.32). Çıkan sonuç ise operatörden bağımsız olarak o kuralın etkisini belirleyebilmek için w_5 kural ağırlığı ile çarpılır ($K_5 = \max(B_1, \bar{E}_2) \times w_5$) ve 5. *kuralın ateşleme gücü (K_5) değeri* elde edilir.

Geriye kalan kuralların ateşleme gücü değerleri de bulunur ($K_6 = \bar{B}_1 \times F_2 \times w_6$, $K_7 = C_1 \times D_2 \times w_7$, $K_8 = \max(\bar{C}_1 \times \bar{E}_2) \times w_8$ ve $K_9 = \max(C_1 \times F_2) \times w_9$).

Böylelikle kuralların gösterdiği *üyelik değerleri* kullanılarak *kuralların ateşleme gücü değerleri* elde edilmiş olur ($K_1, K_2, \dots, K_9$).

Kural ağırlığı belirleme işlemi bazen bir sonraki adım içinde ilk işlem olarak gösterilmektedir. Ancak bu adımda son işlem veya bir sonraki adımda ilk işlem olmasının matematiksel hesaplama sonuçlarına bir etkisi yoktur.

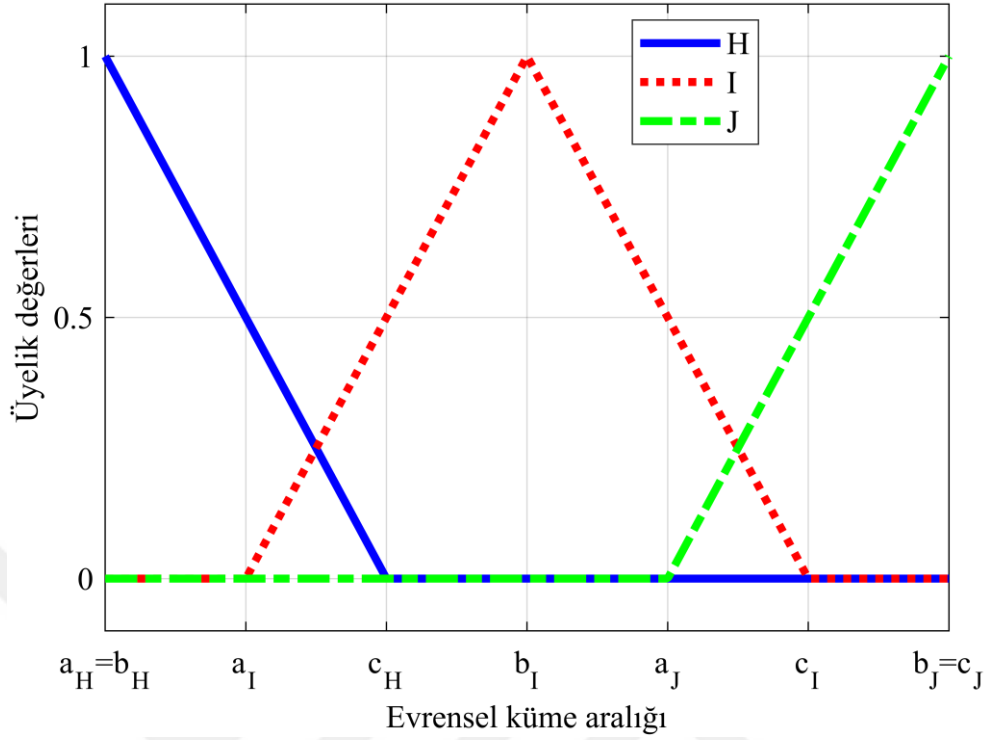
Implication

Implication bulanık çıkarım aşamasının ikinci adımıdır. Şimdiye kadar anlatılanların genellikle birçok bulanık mantık sisteminde aynı olmasına rağmen bu adımda farklı metotlar önerilmiştir. Bunlardan en önemli olanlar anlatılacaktır.

Mamdani metodu

En çok kullanılan metotlardan birisidir. Bu metodun *Implication* adımı yapılan işlemleri anlatılacaktır. Bu metot bulanık kümeleri kullanmaya devam etmektedir. Kurallar aracılığı ile elde edilmiş kuralların ateşleme gücü değerleri ve çıkış üyelik fonksiyonları bu adıma özel bulanık operatörler ile işlendiği zaman bu sefer kuralların bulanık çıkış kümeleri veya değerleri elde edilecektir.

Şekil 3.14'te örnek bir Ç çıkışı için seçilebilecek 3 üyelik fonksiyonu gösterilmiştir. Örnek olarak aynı tarz ve aralıkta üyelik fonksiyonu seçildiği için girişlere benzemektedir. İstenirse farklı üyelik fonksiyonu seçilebilmektedir.



Şekil 3.14. Çıkış üyelik fonksiyonları (Mamdani metodu)

Kural 1: EĞER (G_1, A ise) VE (G_2, D ise) O HALDE (ζ, H 'dır)

Kural 2: EĞER (G_1, A ise) VE (G_2, E ise) O HALDE (ζ, I 'dır)

⋮

İlk adımda kuralların EĞER kısımları birleştirilmiş ve her bir kuralın ateşleme gücü değeri bulunmuştu. Bu adımda ise kuralın O HALDE kısımlarını çalıştıracaktır.

İlk kuralda elde edilmiş olan K_1 değeri ve kuralın gösterdiği H üyelik fonksiyonu bir bulanık operatör ile işleme tabi tutulacaktır. Aynı şekilde K_2 değeri ve I üyelik fonksiyonu bir bulanık operatör ile işleme tabi tutulacaktır. *Ateşleme gücü hesabında* o adıma özel olarak seçilen *prod* operatörü ile işlemler yapılmıştı. *Implication* adımımda ise yine adıma özel bir bulanık operatör seçilecektir. Bu adım için yine *bulanık kesişim* operatörlerinden birisi seçilmektedir. Bulanık kesişim işlemi yapabilmek için seçilebilecek *min* ve *prod* operatörlerinin çıkışları gösterilmişti (bkz. Şekil 3.11). *Min*

operatörü ile üçgenin K_2 değeri üstünde kalan kısmı kesilip atılmaktadır. Bu durumda *kuralın bulanık çıkış kümesi* $K_{2I} = \min(I, K_2)$ bu şekilde elde edilir (bkz. s.30). Diğer *prod* operatörü ile üçgen yeni maksimum değeri 1 yerine K_2 değeri olacak şekilde daraltılmaktadır. Bu durumda ise *kuralın bulanık çıkış kümesi* $K_{2I} = I \times K_2$ bu şekilde elde edilir (bkz. s.31) ve yine aynı isimlendirmeyi yapabiliriz. Sonucun hangi bulanık operatör ile bulunduğu ileride yapılacak işlemleri etkilemediği için daha sade olması açısından aynı isimlendirme tercih edilmiştir. Bütün kurallar için aynı işlemler kuralın bulanık operatörüne göre yapıldıktan sonra bütün *kuralların bulanık çıkış kümeleri* elde edilmiş olur ($K_{1H}, K_{2I}, K_{3H}, K_{4I}, K_{5I}, K_{6H}, K_{7H}, K_{8I}$ ve K_{9J}).

Sugeno metodu

En çok kullanılan metotlardan birisidir. *Mamdani* metodundan farklı olarak bu metot bu adımda artık bulanık kümeleri kullanmaya son vermiştir. Bundan sonra yapılacak hesaplamalarda *doğrusal* ve *sabit* gibi farklı polinom yaklaşımları yapılmıştır. Hangi yaklaşım seçilirse seçilsin bu adımda her bir *kuralın ateşleme gücü değeri* ile o kuralın gösterdiği çıkış üyelik fonksiyonunun sonucu genellikle *prod* operatörü kullanılarak işleme tabi tutulur. İşlem sonucunda her bir *kuralın bulanık çıkış değeri* elde edilecektir.

Ç çıkışında yer alan H , I ve J üyelik fonksiyonları tercih edilen yaklaşıma göre değişmektedir. Bu yaklaşımlardan birisi *doğrusal polinom* kullanmaktır. Bu durumda keskin giriş değerleri bir kez daha hem de bulanıklaştırılmadan kullanılmaktadır.

$$I = a_I \times x_1 + b_I \times x_2 + c_I \quad (3.18)$$

$$I = c_I \quad (3.19)$$

Denklem 3.18'de doğrusal polinom fonksiyonu gösterilmiştir. Burada a_I ve b_I değerleri, x_1 ve x_2 keskin giriş değerleri için birer katsayıdır ve c_I eklenen sabit bir değerdir. Denklem 3.19'da ise sabit katsayı çıkışları kullanılmaktadır. Yani I çıkış üyelik fonksiyonu için *sabit polinom* yaklaşımı seçildiğinde c_I sabit değeri tek başına kullanılmaktadır. Bütün bu parametreler her çıkış üyelik fonksiyonu için ayrı olarak

belirlenmektedir. Yani örneğimizde yer alan diğer H ve J çıkışları yine parametrelerin değiştiği çıkış üyelik fonksiyonlarından oluşmaktadır.

H çıkış üyelik fonksiyonunun sonucu aynı isimle H değeri olsun. Bu durumda *Kural 1*'de elde edilmiş olan K_1 değeri ve H değeri kuralın gösterdiği operatör ile işleme tabi tutulacaktır. Genellikle *prod* operatörü ile yapılan bu işlemde 1. *kuralın bulanık çıkış değeri* $K_{1H} = H \times K_1$ bu şekilde elde edilir ve yine aynı isimlendirmeyi yapabiliriz. Aynı şekilde 2. *kuralın bulanık çıkış değeri* $K_{2I} = I \times K_2$ olarak hesaplanır (bkz. s.31). Sonucun hangi operatör veya metot ile bulunduğu ileride yapılacak işlemleri etkilemediği için daha sade olması açısından aynı isimlendirme tercih edilmiştir.

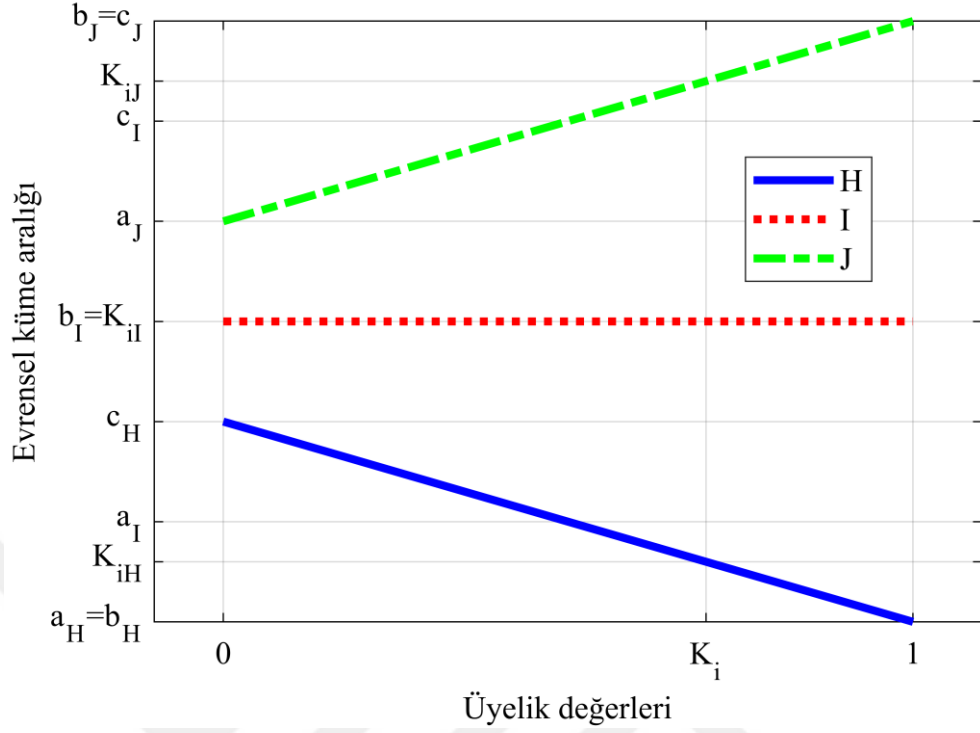
Bütün kurallar için aynı işlemler yapıldıktan sonra *kuralların bulanık çıkış değerleri* elde edilmiş olur ($K_{1H}, K_{2I}, K_{3H}, K_{4I}, K_{5I}, K_{6H}, K_{7H}, K_{8I}$ ve K_{9J}).

Tsukamoto metodu

Bu metotta, diğer metotlardan farklı olarak *kuralların ateşleme gücü değerleri* bu adımda iki kere kullanılmaktadır. Bu adımda her bir *kuralın ateşleme gücü değeri* ile o kuralın gösterdiği çıkış üyelik fonksiyonunda *kuralın ateşleme gücü değerine* karşılık gelen *keskin giriş değeri* karşılığı kullanılarak genellikle *prod* operatörü ile işleme tabi tutulur. İşlem sonucunda bu sefer *kuralların bulanık çıkış değerleri* elde edilecektir.

Çıkış üyelik fonksiyonları tercih edilen üyelik fonksiyonuna göre değişmektedir. Bu fonksiyonlardan birisi olan üçgen için oluşacak çıkış üyelik fonksiyonları her hangi bir kuralın ateşleme gücü değeri için iki farklı sonuç vermektedir. Bu sonuçlar ortalama olarak kabul edilse ortalama her zaman aynı olacaktır. O yüzden çıkışlar için genellikle *sürekli azalan* veya *artan* şekilde üyelik fonksiyonları seçildiği zaman tek *keskin giriş değer* karşılığı olacaktır.

Şekil 3.15'te azalan, simetrik ve artan bir üçgen fonksiyonu için kullanılabilecek çıkış üyelik fonksiyonu gösterilmiştir. Bir standart olarak giriş ekseni yatayda olacak şekilde gösterilmiştir. Ayrıca ortadaki simetrik üçgen olduğu için girişler değiştiğinde ortalama olarak tek çıkış vermekte bu yüzden çubuk olarak gözükmemektedir.



Şekil 3.15. Çıkış üyelik fonksiyonları (Tsukamoto metodu)

Bu metotta, $i = \{1, 2, \dots, 9\}$ olmak üzere *Kural i* aracılığı ile bulunan K_i değeri kuralın gösterdiği çıkış üyelik fonksiyonu ile asıl işlemde önce bir kez daha işlenmektedir. Bu değer kullanılarak her bir i değeri için sadece bir tane keskin giriş değer karşılığı (H_i , I_i veya J_i) bulunacaktır.

$$H_i = K_i \times (b_H - c_H) + c_H \quad (3.20)$$

$$I_i = b_I \quad (3.21)$$

$$J_i = K_i \times (b_J - a_J) + a_J \quad (3.22)$$

Denklem 3.20’de azalan bir üçgen fonksiyonu için kullanılabilecek çıkış üyelik fonksiyonu gösterilmiştir. Burada i değerinin aralığı kural sayısı kadardır ancak her denklem için değer hesaplanmaz. *Kural i* hangi çıkış üyelik fonksiyonunu gösteriyorsa o denkleme göre sonuç hesaplanır ve sonuçlarda kural sayısı kadar olur. Örneğin, $i = 1$ durumu için *Kural 1* ile bulunan K_1 değeri ve *Kural 1*’in gösterdiği H çıkış üyelik

fonksiyonu ile asıl işlem den önce bir kez daha işlenmektedir. Bu değ er kullanılarak buradan keskin giriş karş ılığ ı H_1 değ eri bulunacaktır. Burada b_H ve c_H değ erleri azalan üç genin parametreleridir. Üç gen azaldığı için artan kısmı belirleyen a_H değ eri b_H değ eri ile aynıdır ve kullanmaya gerek yoktur. Diğ er kurallar için ise denk gelen çıkış üyelik fonksiyonları kullanılacaktır. Denklem 3.21’de simetrik tip bir üyelik fonksiyonu için oluş akan çıkış üyelik fonksiyonu gösterilmiştir. Simetrik bir üç gen üyelik fonksiyonu seçildiğ i için $i = 2$ durumunda *Kural 2* kullanılarak elde edilmiş olan K_2 değ eri ne olursa olsun kullanılan fonksiyon için keskin giriş karş ılığ ı iki tane olacak ve ortalaması daima b_I olacaktır ve bunu isimlendirme düzenini bozmamak için I_2 olarak isimlendirebiliriz. Denklem 3.22’de ise artan bir üç gen üyelik fonksiyonu için kullanılabilecek çıkış üyelik fonksiyonu gösterilmiştir. Burada $i = 9$ durumu için *Kural 9* ile bulunan K_9 değ eri ve *Kural 9*’un gösterdiğ i J çıkış üyelik fonksiyonu ile asıl işlem den önce bir kez daha işlenmektedir. Bu değ er kullanılarak buradan keskin giriş karş ılığ ı J_9 değ eri bulunacaktır. Burada a_J ve b_J değ erleri artan üç genin parametreleridir. Üç gen arttığı için azalan kısmı belirleyen c_J değ eri b_J ile aynıdır ve kullanmaya gerek yoktur.

Bütün bu parametreler her çıkış üyelik fonksiyonu için ayrı olarak belirlenmektedir. Yani örneğ imizde yer alan H , I ve J çıkış üyelik fonksiyonlarının parametreleri farklıdır. Ancak *Sugeno* metodundan farklı olarak çıkış üyelik fonksiyonu sayısı kadar değ il kural sayısı kadar farklı sonuç işleme alınacaktır. Yani *Sugeno* metodunda bu adımda operatör girişleri H , I ve J değ erleri olarak sadece üç tane olurken bu metotta operatör girişleri $H_1, I_2, H_3, I_4, I_5, H_6, H_7, I_8$ ve J_9 değ erleri olmak üzere kural sayısı kadardır.

Kural 1’de elde edilmiş olan K_1 değ eri ve kuralın gösterdiğ i H çıkış üyelik fonksiyonu kullanılarak elde edilen H_1 keskin giriş değ eri karş ılığ ı bir operatör ile işleme tabi tutulacaktır. Genellikle *prod* operatörü ile yapılan bu işlemde 1. *kuralın bulanık çıkış değ eri* $K_{1H} = H_1 \times K_1$ bu şekilde elde edilir ve yine aynı isimlendirmeyi yapabiliriz. Aynı şekilde 2. *kuralın bulanık çıkış değ eri* $K_{2I} = I_2 \times K_2$ olarak hesaplanır (bkz. s.31). Sonucun hangi operatör veya metot ile bulunduğu ileride yapılacak işlemleri etkilemediğ i için daha sade olması açısından aynı isimlendirme tercih edilmiştir.

Bütün kurallar için aynı işlemler yapıldıktan sonra *kuralların bulanık çıkış değerleri* elde edilmiş olur ($K_{1H}, K_{2I}, K_{3H}, K_{4I}, K_{5I}, K_{6H}, K_{7H}, K_{8I}$ ve K_{9J}).

Aggregation

Bulanık çıkarımın üçüncü ve son adımında bütün kuralların bulanık çıkış kümeleri veya değerlerinin bir operatör aracılığı ile birleştirilmesi gerekmektedir. Çünkü artık kuralların işletilmesi tamamlanmıştır. Ayrıca tek kural olursa zaten bir tane kuralın bulanık çıkış kümesi veya değeri olacağından bu işleme gerek duyulmayacaktır. Ancak tek kurallı bir BMD tasarımı mantıklı olmadığından bu adım her BMD tasarımında yapılmaktadır. *Implication* adımında mevcut olan metotlar bu adımda da farklı çalışmaktadır.

Mamdani metodu

Elde edilmiş olan *kuralların bulanık çıkış kümeleri* bu metotta bu adıma özel bulanık operatörlerle işleme tabi tutularak tek bir tane *bulanık çıkış kümesi* genellikle bir işlemle elde edilecektir. Örneğin *max* operatörü seçildiğinde tek bir tane *bulanık çıkış kümesi* $\zeta_B = \max(K_{1H}, K_{2I}, K_{3H}, K_{4I}, K_{5I}, K_{6H}, K_{7H}, K_{8I}, K_{9J})$ şeklinde elde edilir. Bu 9 kümenin nasıl işlendiğini göstermek grafik üzerinde karmaşa oluşturmaktadır. Ancak ζ çıkışında 3 tane üyelik fonksiyonuna denk gelenler aralarında işlenirse sonuç aynı olur. Yani $\zeta_B = \max(\max(K_{1H}, K_{3H}, K_{6H}, K_{7H}), \max(K_{2I}, K_{4I}, K_{5I}, K_{8I}), K_{9J})$ şeklinde de sonuç bulunabilir. İlk olarak *H* üyelik fonksiyonuna denk gelenler $\zeta_H = \max(K_{1H}, K_{3H}, K_{6H}, K_{7H})$ şeklinde isimlendirilebilir. İkinci olarak *I* üyelik fonksiyonuna denk gelenler $\zeta_I = \max(K_{2I}, K_{4I}, K_{5I}, K_{8I})$ şeklinde isimlendirilebilir. Son olarak ise *J* üyelik fonksiyonuna denk gelen $\zeta_J = K_{9J}$ hiçbir işlem yapılmadan yeni ismini verebiliriz. Bu durumda $\zeta_B = \max(\zeta_H, \zeta_I, \zeta_J)$ işlemi yapılarak aynı *bulanık çıkış kümesi* bulunacaktır. Buradaki *max* operatörünün girişleri olan ara bulanık çıkış kümeleri gösterilmişti (bkz. Şekil 3.12). Bütün bu işlemler sonucunda *max*, *sum* veya *probor* operatörü kullanılarak bu adımda elde edilecek *bulanık çıkış kümeleri* de gösterilmişti (bkz. Şekil 3.13). Kısaca bulanık operatörlerden bahsederek *max* operatöründe üçgenlerin tepe noktaları birleştirilir. *Sum* operatöründe bütün noktalarda

toplama yapıldığı için en yüksek üyelik değerleri oluşur. *Probor* operatöründe ise bu toplamdan çarpım çıkarıldığı için çarpımın sıfır olmadığı bölgelerde biraz daha az üyelik değerleri oluşur. Herhangi bir bulanık operatör seçilerek 9 kuralın bulanık çıkış kümesi ile veya sadeleştirilmiş 3 ara bulanık çıkış kümesi ile yapılan işlemde tek bir tane *bulanık çıkış kümesi* elde edilmiş olur (ζ_B).

Sugeno ve Tsukamoto metotları

Bu metotlarda *kuralların bulanık çıkış değerleri* genellikle *sum* operatörü ile işleme tabi tutulur (bkz. s.33). Bu sefer tek bir tane *bulanık çıkış değeri* elde edilmiş olur (ζ_B).

3.2.7. Durulaştırma

Bulanık çıkarım işlemlerinden sonra elde edilmiş *bulanık çıkış kümesinin* denetleyicinin anlayabileceği bir şekle dönüştürülmesi gerekmektedir. Bu işleme *durulaştırma* denir. İşlem sonucunda çıkan *keskin çıkış değeri*dir ve BMD'ye yeni keskin giriş değeri geldikçe BMD yeni keskin çıkış değeri üretir ve döngü devam eder. Böylece BMD kontrol sinyalini sağlar ve kontrol işlemi bu sinyal ile gerçekleştirilir. Ayrıca bu sinyal başka denetleyiciler için giriş olarak kullanılabilir.

Durulaştırma işlemi için kullanılan farklı yöntemler vardır. Aslında durulaştırma aşaması adı gereği her zaman *bulanık kümeler* için yapılıyor gibi düşünülse de farklı yaklaşımlar ile bu durum değişmiştir. Bazen durulaştırma bile yapılmamaktadır. Devam eden başlıklarda durulaştırma için kullanılabilecek yöntemler metotlara özel bir şekilde ayrı olarak anlatılmıştır.

Mamdani metodunda kullanılan yöntemler

Elde edilmiş olan *bulanık çıkış kümesi* kullanılarak durulaştırma yapılacaktır. İşlem sonucunda *keskin çıkış değeri* elde edilecektir (ζ_K). Bulanık kümeler için kullanılan yöntemlerden MATLAB yazılımında kullanıma sunulanlar anlatılmıştır.

Ağırlık merkezi yöntemi

Durulaştırma aşamasında yaygın kullanılan yöntemlerden biri *ağırlık merkezi yöntemi* olmuştur. Çünkü bu yöntemde *bulanık çıkış kümesinin* tamamı kullanılarak durulaştırma sonucu bulunmaktadır.

$$\zeta_K = \sum_i (\zeta_B(x_i) \times x_i) / \sum_i \zeta_B(x_i) \quad (3.23)$$

Denklem 3.23'te gösterilen ζ_K ağırlık merkezi yöntemi ile bulunan keskin çıkış değerini temsil etmektedir. Denklemdeki x_i evrensel küme aralığında yer alan bütün x değerlerinin işleneceğini göstermektedir. $\zeta_B(x_i)$ ise ζ_B üyelik fonksiyonunun evrensel küme aralığında yer alan bütün x değerleri için karşılık gelen bütün üyelik değerlerini temsil etmektedir. Şekil 3.16 ve Şekil 3.17'de örnek bir bulanık çıkış kümesi üzerinde ağırlık merkezi yöntemi ile bulunan keskin çıkış değeri MATLAB yazılımında geçen adıyla *centroid* olarak gösterilmiştir.

İki bölge yöntemi

İki bölge yönteminde bulanık çıkış kümesi eşit iki alana bölünerek durulaştırma sonucu bulunur. İki bölge yöntemi ile ağırlık merkezi yöntemi benzer sonuçlar üretmektedir. Ancak formülün bilgisayar ortamında uygulanması biraz daha karmaşıktır.

$$\zeta_K = x_j, \quad (j | \sum_{i=1}^j \zeta_B(x_i) \geq \sum_{i=j+1}^p \zeta_B(x_i)) \quad (3.24)$$

Denklem 3.24'te gösterilen ζ_K iki bölge yöntemi ile bulunan değeri temsil etmektedir. Denklemdeki x_i evrensel küme aralığında yer alan p tane değer bütün x değerlerinin işleneceğini göstermektedir. $\zeta_B(x_i)$ ise ζ_B üyelik fonksiyonunun evrensel küme aralığında yer alan bütün x değerleri için karşılık gelen bütün üyelik değerlerini temsil etmektedir. Burada bulunan j bu eşitsizliği sağlayacak en küçük değer olmalıdır. Yani diğer bir deyişle j öyle bir değerdir ki j yerine konulacak $j - 1$ değeri bile bu eşitsizliği sağlamayacaktır. Bu eşitsizliği sağlayan j değeri x_i ifadesinde yerine konulduğunda bulunan x_j değeri durulaştırma sonucudur. Bunu ζ_K olarak düzene uygun bir şekilde

yeniden isimlendirdik. Şekil 3.16 ve Şekil 3.17’de örnek bir bulanık çıkış kümesi üzerinde iki bölge yöntemi ile bulunan keskin çıkış değeri MATLAB yazılımında geçen adıyla *bisector* olarak gösterilmiştir.

Ağırlık merkezi yöntemi veya iki bölge yöntemi ile bulunan durulaştırma sonuçları bulanık çıkış kümesinin eksenin ne tarafında olduğundan etkilenmez, sadece şeklinden etkilenir. Şekillerde görüldüğü üzere bulanık çıkış kümesinin sağ taraflarında daha yüksek üyelik değerleri olduğu için durulaştırma sonuçları ortadan sağa doğru kaymıştır.

Maksimumların ilki, sonuncusu ve ortalaması yöntemleri

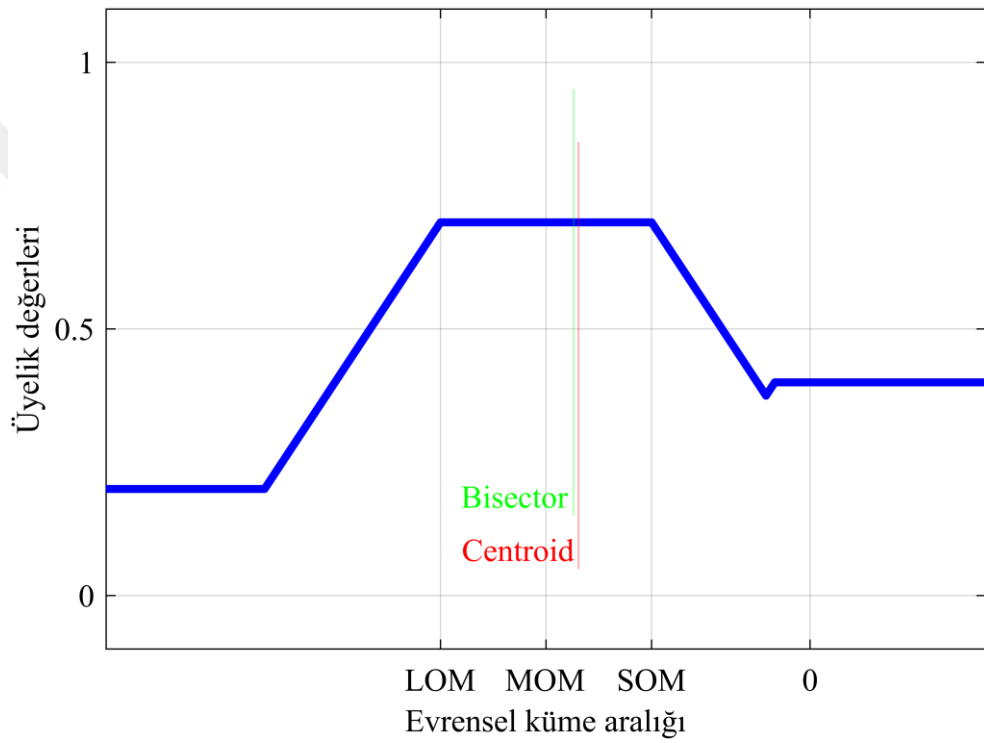
Çıkışın en büyük olduğu üyelik değerlerine denk gelen bölgedeki keskin giriş değerleri kullanılarak hesaplanmaktadır. Bu yöntemlerde çıkışın en büyük olduğu yerler dışındaki *bulanık çıkış kümesinin şekli sonuca hiçbir şekilde etki etmemektedir.*

Çıkışın en büyük olduğu üyelik değerlerine denk gelen bölgedeki keskin giriş değerlerinin ortalaması alınarak Maksimumların Ortalaması (MOM) değeri bulunmaktadır. Şekil 3.16 ve Şekil 3.17’de örnek bir bulanık çıkış kümesi üzerinde MOM değeri gösterilmiştir. Çıkışın en büyük olduğu üyelik değerlerine denk gelen bölgedeki keskin giriş değerlerinin 0 noktasına en yakın olanı alınarak Maksimumların İlki (SOM) değeri bulunmaktadır. Şekil 3.16 ve Şekil 3.17’de örnek bir bulanık çıkış kümesi üzerinde SOM değeri gösterilmiştir. Çıkışın en büyük olduğu üyelik değerlerine denk gelen bölgedeki keskin giriş değerlerinin 0 noktasına en uzak olanı alınarak Maksimumların Sonuncusu (LOM) değeri bulunmaktadır. Şekil 3.16 ve Şekil 3.17’de örnek bir bulanık çıkış kümesi üzerinde LOM değeri gösterilmiştir.

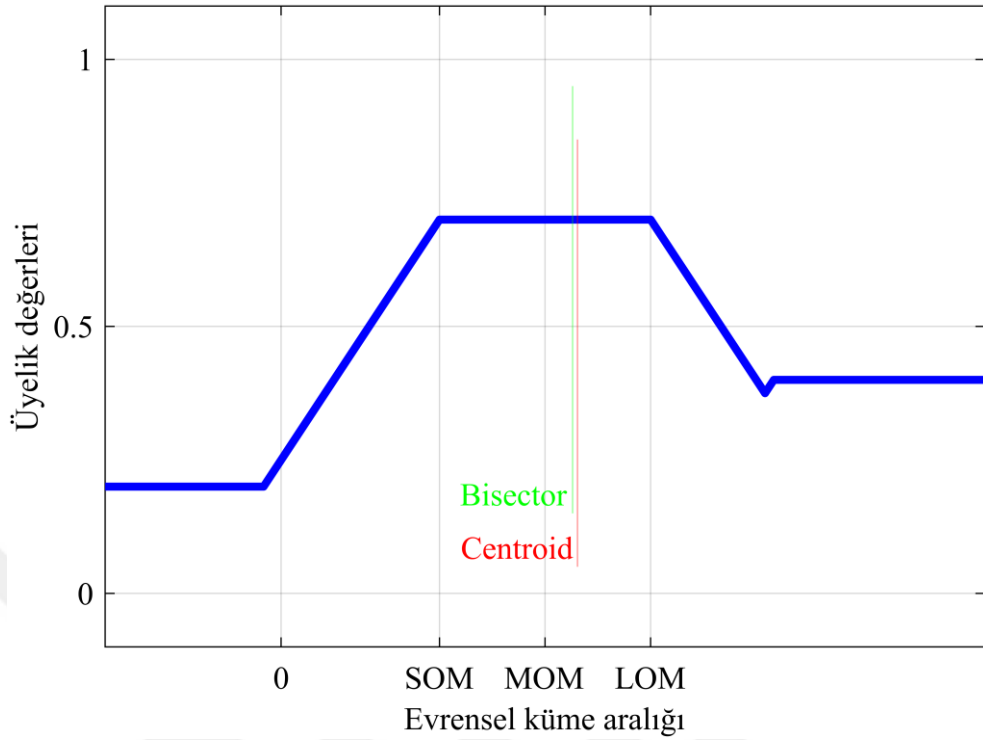
Maksimumların ilki, sonuncusu ve ortalaması yöntemleri ile bulunan durulaştırma sonuçları bulanık çıkış kümesinin maksimum üyelik değeri olmayan yerlerinden etkilenmez. Ancak maksimumların ilki ve sonuncusu yöntemleri eksenin ne tarafında olduğundan etkilenirler. Şekillerde görüldüğü üzere bulanık çıkış kümesinin eksenin ne tarafında olduğu durulaştırma sonuçlarının yerlerini değiştirmiştir. Çünkü bu yöntemlerde en küçüklük ve en büyüklük kıstası mutlak değerli bir biçimde

değerlendirilmektedir. Yani 0 noktasına yakınlık ve uzaklık açısından değerlendirme yapılmaktadır.

Şekil 3.16 ve Şekil 3.17’de *max* operatörü kullanılarak bulunan bulanık çıkış kümesi gösterilmiştir. Ayrıca bu küme kullanılarak bulunabilecek durulaştırma sonuçları gösterilmiştir. Yatay ekseninde neredeyse aynı noktaya rastlayan bazı değerler üst üste binmeyi önlemek için eksenin yukarısında gösterilmiştir.



Şekil 3.16. Durulaştırma sonuçları (Eksenin sol tarafı) (Mamdani metodu)



Şekil 3.17. Durulaştırma sonuçları (Eksenin sağ tarafı) (Mamdani metodu)

Sugeno ve Tsukamoto metotlarında kullanılan yöntemler

Elde edilmiş olan *bulanık çıkış değeri* kullanılarak durulaştırma yapılacaktır. İşlem sonucunda *keskin çıkış değeri* elde edilecektir (ζ_K). Sugeno metotlarında kullanılan yöntemlerden MATLAB yazılımında kullanıma sunulanlar anlatılmıştır. Tsukamoto metodunda da aynı yöntemler kullanılabilir.

Ağırlıklı ortalama yöntemi

Sum operatörü kullanılarak elde edilmiş *bulanık çıkış değeri* zaten bir toplam ifadesidir. Bu ifade direk olarak daha önceki aşamalarda elde edilmiş *Kuralların ateşleme gücü değerlerinin* toplamına bölünerek *ağırlıklı ortalama yöntemi* ile sonuç bulunmaktadır.

$$\zeta_K = \zeta_B / \sum_i K_i \quad (3.25)$$

Denklem 3.25'te gösterilen ζ_K *ağırlıklı ortalama yöntemi* ile bulunan değeri temsil etmektedir. Denklemdeki K_i her bir *kuralın ateşleme gücü değeri*ni işleneceğini göstermektedir. ζ_B ise önceki aşamada elde edilmiş *bulanık çıkış değeri*dir.

Ağırlıklı toplam yöntemi

Sum operatörü kullanılarak bulunan *bulanık çıkış değeri* bir toplam ifadesidir. Bu ifade zaten önceden toplanmış tek bir değerdir ve tekrar toplanılmak istenilse bile toplamı da aynı değerdir. Yani kısaca bu durulaştırma yönteminde hiçbir işleme gerek yoktur. Sonuç aynen kullanılmaktadır.

$$\zeta_K = \zeta_B \quad (3.26)$$

Denklem 3.26'da gösterilen ζ_K *ağırlıklı toplam yöntemi* ile bulunan değeri temsil etmektedir. ζ_B önceki aşamada elde edilmiş *bulanık çıkış değeri*dir ve aynen kullanılmaktadır. Yani başka bir deyişle bu tek *bulanık çıkış değeri*ni durulaştırma ihtiyacı yoktur.

3.3. Yapay Sinir Ağı

Yapay Sinir Ağı (YSA) son derece karmaşık fonksiyonları modelleyebilen doğrusal olmayan bir hesaplama aracıdır. Giriş ve çıkış arasında herhangi bir fonksiyonel ilişki bulunmasının mümkün olmadığı veya çok zor olduğu durumlarda tercih edilebilir. Fonksiyon ihtiyacı duymaksızın uygun şekilde seçilmiş bir mimari ve yapılacak başarılı bir öğrenme sonucu elde edilen YSA sayesinde sistemler temsil edilebilir (Marini ve ark., 2008). YSA'nın temel işlem elemanı olan hücreler doğrusal değildir. Bu sayede doğrusal olmayan karmaşık problemlerin çözümünde en önemli araçlardan biri olmuştur ve mimarilerinde genellikle paralel dağılmış hücreler bulunur. Bu paralel dağılmış yapısı sayesinde kolay öğrenibilme yapabilmektedir. Genelleme özelliği sayesinde karşılaşmadığı girişler için uygun tepkiler üretebilir. Bu sayede fonksiyonu bilinmeyen karmaşık problemleri bile çözebilmektedir (Ergezer ve ark., 2003). YSA'lar askeri, elektronik, enerji, finans, otomotiv ve uzay bilimleri gibi birçok değişik alanda kullanılmaktadır. Bu alanlarda YSA'lar kullanılarak veri madenciliği, istatistiksel

tahmin yöntemleri, meteorolojik yağış verilerinin tahmini, endüstriyel problemlerin çözümü, güç sistemlerinde yük akış analizi ve insansız robotların geliştirilmesi gibi uygulamalar yapılmaktadır (Öztürk ve Şahin, 2018). YSA'lar ayrıca keşif analizi yapmak, regresyon yapmak, sınıflandırma yapmak, sınıf modellemek ve karmaşık sistemleri kontrol etmek gibi birçok farklı amaç için kullanılmaktadır (Marini ve ark., 2008).

3.3.1. Yapısı

YSA girişlerin bulunduğu bir *giriş katmanı*, yapay sinir hücrelerinin bulunduğu *gizli katmanlar* ve çıkışların bulunduğu bir *çıkış katmanı*ndan oluşmaktadır. Bu içeriklerden büyük bir çoğunluğu oluşturan yapay sinir hücresi gizli katmanlar ve çıkış katmanında bulunmaktadır. Bu hücreler önceki katmandan gelen bir giriş değeri için, *ağırlık* ve *eşik değerlerini* kullanılarak *çarpım*, *toplam* ve *aktivasyon fonksiyonları* aracılığı ile bir çıkış değeri üretmektedir. *Ağırlık* ve *eşik değerleri* sadece sayılardan ibarettir. *Çarpım fonksiyonu* kendi girişine gelen bütün değerleri çarparak bir çıkış üretir. *Toplam fonksiyonu* da aynı şekilde kendi girişine gelen bütün değerleri toplayarak bir çıkış üretir. *Aktivasyon fonksiyonu* ise kendi girişine gelen bütün değerleri seçilen bir fonksiyondan geçirerek bir çıkış üretir.

Aktivasyon fonksiyonları

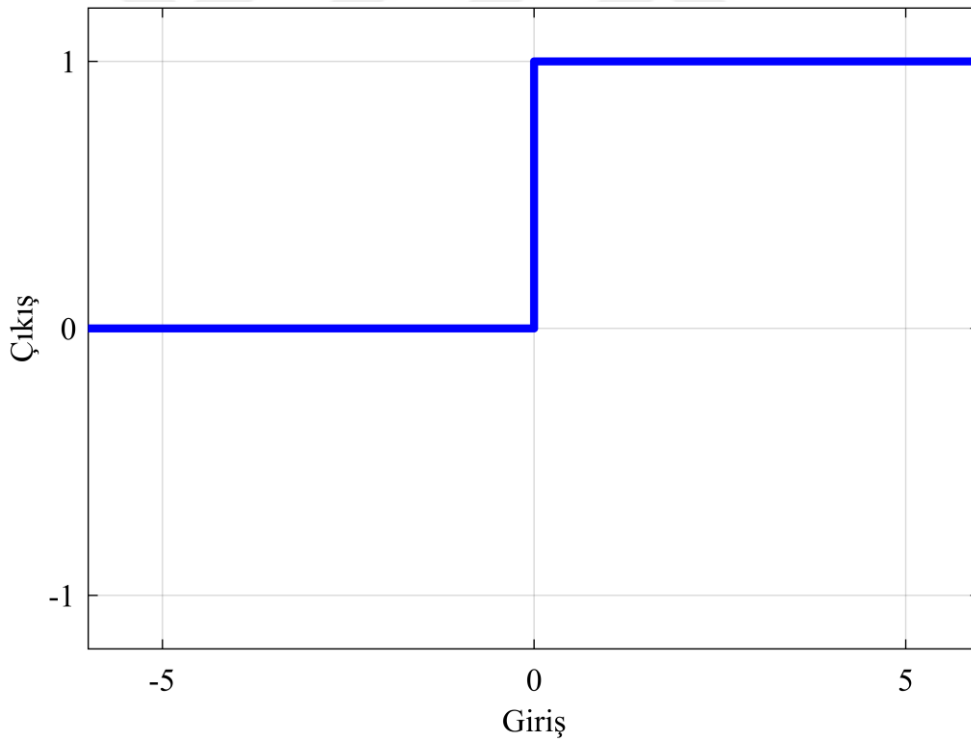
YSA tasarımında kullanılacak olan *aktivasyon fonksiyonları* veya diğer adıyla *transfer fonksiyonları* için çok farklı sayıda seçenek bulunmaktadır. Aktivasyon fonksiyonu bir yapay sinir hücresinin çıkışını ağırlık ve eşik değerlerini daha uygun bir şekilde güncelleyebilmek için gerekli olan bir biçimde işlemek için kullanılmaktadır. İhtiyaca göre *doğrusal* veya *doğrusal olmayan* fonksiyonlar kullanılabilir ve bu sayede doğrusal olmayan YSA yapıları oluşturulabilir. Doğrusal olmayan bir YSA ise doğrusal olmayan sistemlerin kontrolünde etkili sonuçlar verebilmektedir. Ayrıca aktivasyon fonksiyonlarının çıkışı bazen belirli bir sayı aralığında sonuç üretecek şekilde sınırlandırılmaktadır. MATLAB yazılımında diğer adıyla kullanıma sunulan *transfer fonksiyonlarından* önemli olanları anlatılmıştır.

Eşik aktivasyon fonksiyonu

Eşik aktivasyon fonksiyonu veya *Heaviside fonksiyonu* olarak isimlendirilmektedir. MATLAB yazılımında diğer bir ismiyle *Katı Sınır (hardlim) transfer fonksiyonu* olarak kullanıma sunulmuştur. Bir eşiği andıran bu fonksiyon girişine gelen 0 ve üstü değerler için 1 değerini, altı için ise 0 değerini sonuç olarak üretir (Haykin, 2009).

$$y(v) = \begin{cases} 1, & v \geq 0 \\ 0, & v < 0 \end{cases} \quad (3.27)$$

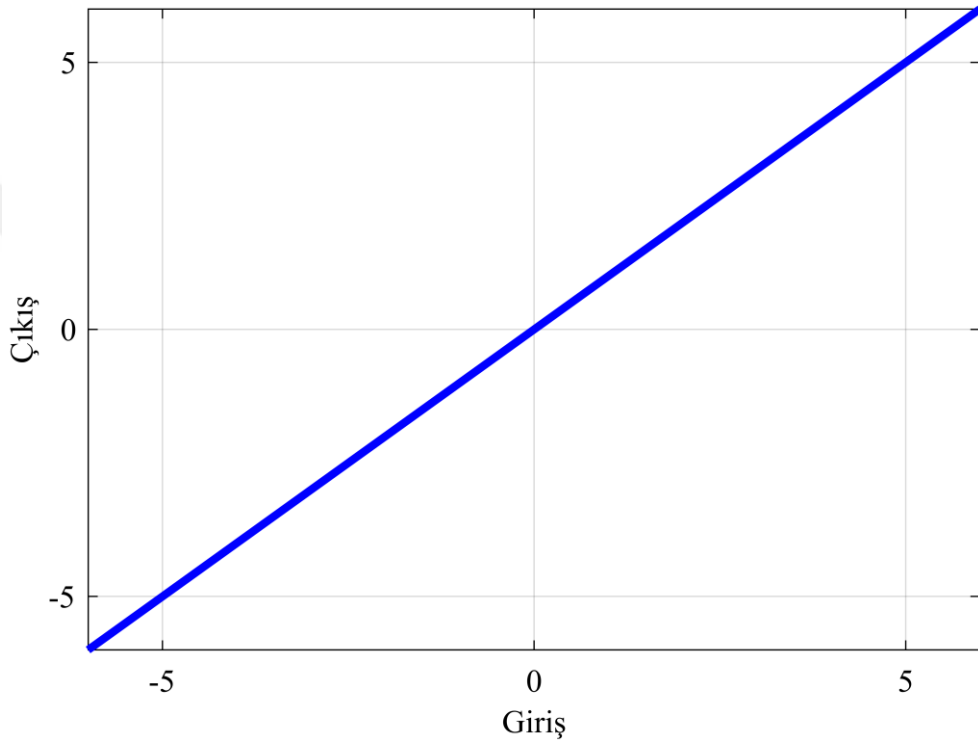
Denklem 3.27’de gösterildiği gibi formülize edilmektedir. Burada v aktivasyon fonksiyonuna giren değer, y ise çıkan sonuçtur. Şekil 3.18’de *hardlim transfer fonksiyonu* grafiği gösterilmiştir.



Şekil 3.18. Hardlim transfer fonksiyonu

Doğrusal aktivasyon fonksiyonu

Aslında diğer fonksiyonlar gibi girişte gelen değere bir işlem yapmaz. Giriş olduğu gibi çıkışa aktarılır ve $y(v) = v$ ile gösterilebilir. MATLAB yazılımında *Doğrusal (purelin) transfer fonksiyonu* olarak kullanıma sunulmuştur. Şekil 3.19'da *purelin transfer fonksiyonu* grafiği gösterilmiştir.



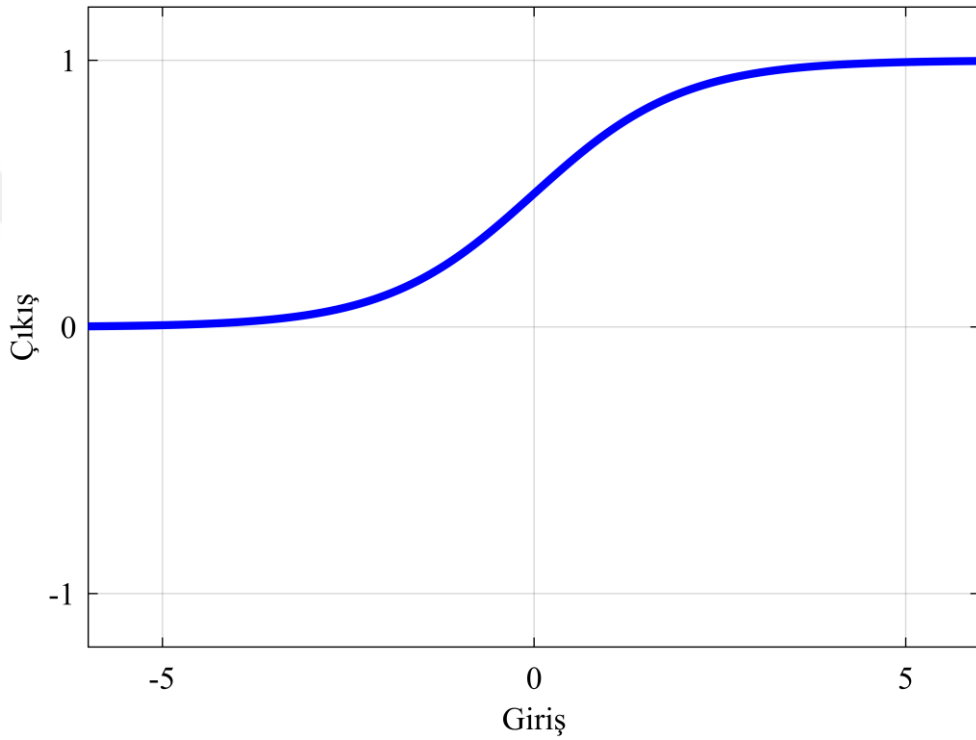
Şekil 3.19. Purelin transfer fonksiyonu

Sigmoid aktivasyon fonksiyonu

Sigmoid aktivasyon fonksiyonu yaygın olarak kullanılmaktadır. Şekli *S* harfini andıran bu fonksiyon doğrusal ve doğrusal olmayan davranış arasında bir denge sergiler (Haykin, 2009).

$$y(v) = \frac{1}{1 + e^{-a*v}} \quad (3.28)$$

Denklem 3.28’de gösterildiği gibi formülize edilmektedir. Burada v aktivasyon fonksiyonuna giren değer, y ise çıkan sonuçtur. Ayrıca burada karşımıza yeni çıkan a parametresi bu fonksiyonun eğimini belirlemekte ve arttıkça eğim dikleşmektedir. Ayrıca $a = 1$ seçildiğinde oluşan özel *sigmoid fonksiyonu* MATLAB yazılımında *Logaritmik Sigmoid (logsig) transfer fonksiyonu* olarak geçmektedir. Şekil 3.20’de *logsig transfer fonksiyonu* gösterilmiştir.



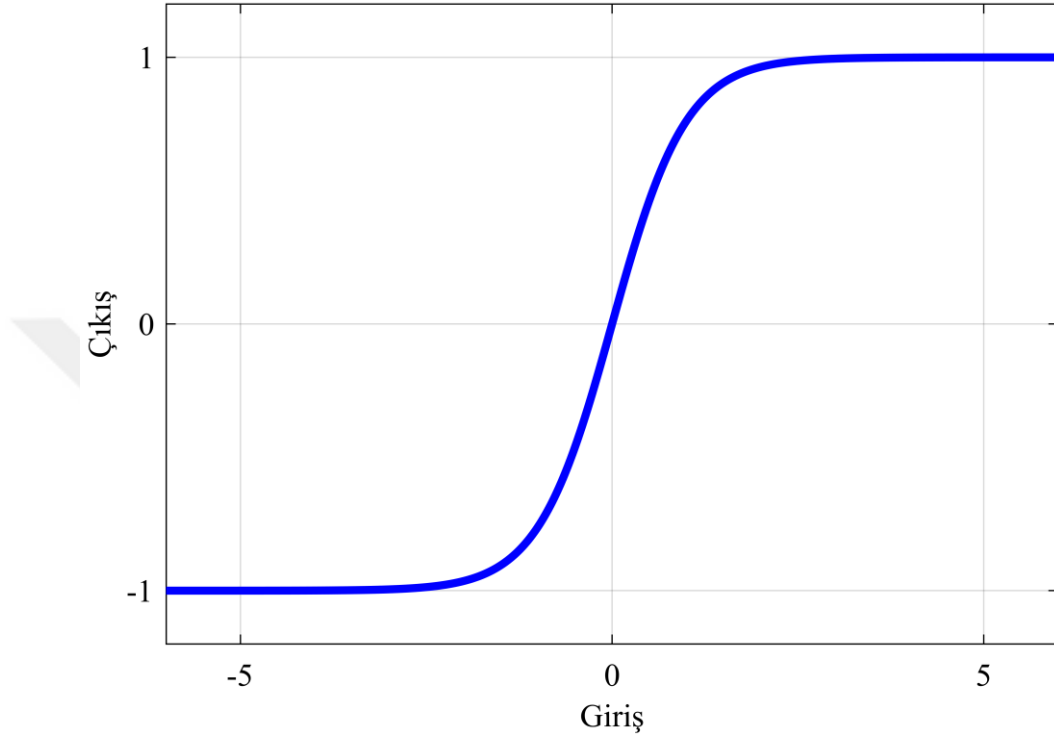
Şekil 3.20. Logsig transfer fonksiyonu

Tanjant sigmoid aktivasyon fonksiyonu

Tanjant Sigmoid (tansig) aktivasyon fonksiyonu, yaygın olarak kullanılmaktadır. *Logsig aktivasyon fonksiyonu* ile farkı negatif çıkışta verebilmesidir (Haykin, 2009).

$$y(v) = \frac{2}{1 + e^{-2*v}} - 1 \quad (3.29)$$

Denklem 3.29’da gösterildiği gibi formülize edilmektedir. Burada v aktivasyon fonksiyonuna giren değer, y ise çıkan sonuçtur. MATLAB yazılımında kullanılabilen *tansig transfer fonksiyonu* Şekil 3.21’de gösterilmiştir.



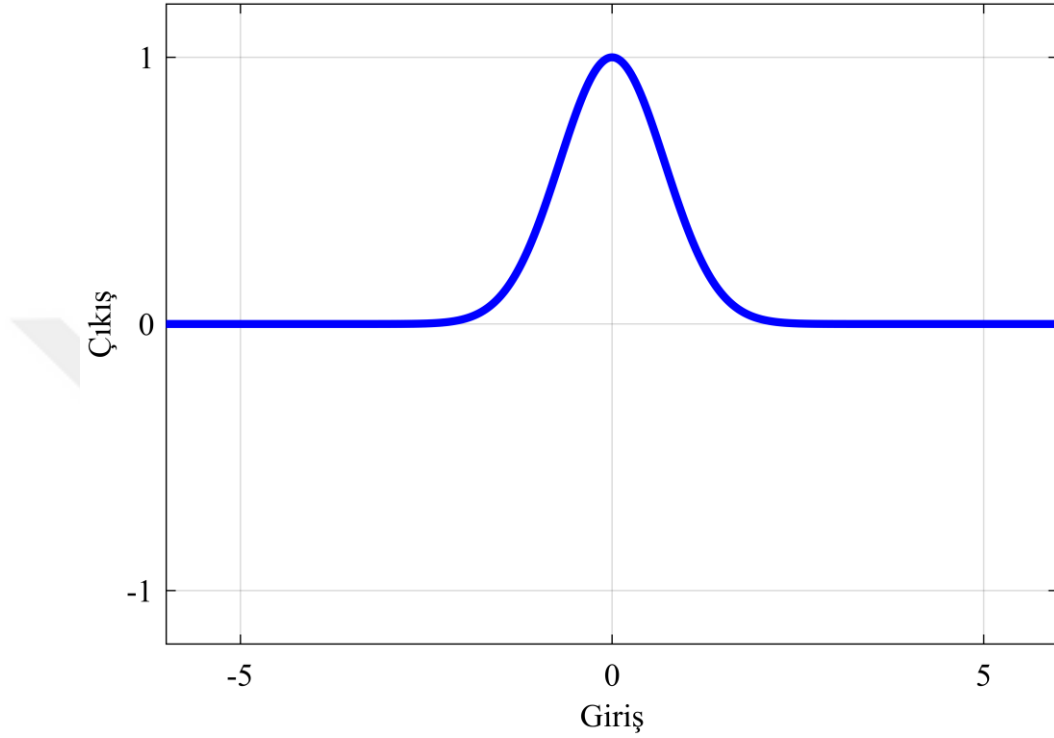
Şekil 3.21. Tansig transfer fonksiyonu

Radyal tabanlı aktivasyon fonksiyonu

Radyal Tabanlı (radbas) aktivasyon fonksiyonu sinir ağlarının yapımında kullanılan simetrik bir aktivasyon fonksiyonu şeklindedir. Girişlerin karesi kullanıldığı için negatif girişlerde pozitif gibi muamele görür yani girişin mutlak değeri alınmış gibi olur. Örneğin giriş bir mesafe değeri ise bu durumda bu fonksiyonun kullanılması iyi bir seçim olabilir.

$$y(v) = e^{-v^2} \quad (3.30)$$

Denklem 3.30’da gösterildiği gibi formülize edilmektedir. Burada v aktivasyon fonksiyonuna giren değer, y ise aktivasyon değerinden çıkan sonuçtur. MATLAB yazılımında kullanılabilen *radbas transfer fonksiyonu* Şekil 3.22’de gösterilmiştir.



Şekil 3.22. Radbas transfer fonksiyonu

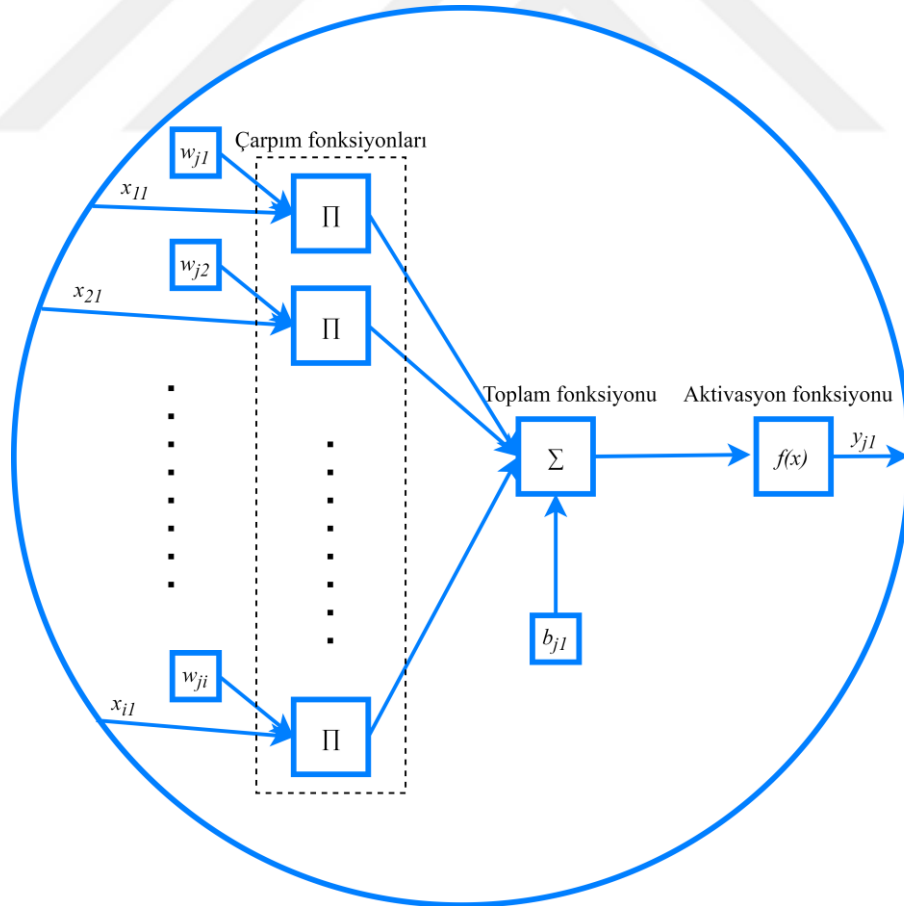
Bunlardan başka çok sayıda aktivasyon fonksiyonu çeşidi bulunmaktadır. Bu aktivasyon fonksiyonları her katman için farklı seçilebilir. Ayrıca katmanlardaki her bir hücre içinde farklı seçilebilir.

Yapay sinir hücresi

Yapay sinir hücresi diğer adıyla *yapay nöron* YSA içerisinde yer alan temel bilgi işleme birimidir. YSA içerisinde çok fazla sayıda hücre olabilir ve her bir hücrede bir takım işlemler yapılmaktadır. Bu hücreler sadece gizli katmanlar ve çıkış katmanında yer alır. Diğer giriş katmanında yer alan hücrelerde ise bu hücrelerdeki gibi işlemler yapılmaz (Haykin, 2009).

$$y_{j1} = f(w_{j1} \times x_{11} + w_{j2} \times x_{21} + \dots + w_{ji} \times x_{i1} + b_{j1}) \quad (3.31)$$

Giriş katmanı tek katmanlı ağlarda çıkış katmanına bağlanırken, çok katmanlı ağlarda ilk gizli katmana bağlanmaktadır. İki durumda da girişe bağlı katmanda bulunan j . yapay sinir hücresinde yapılan işlemler Denklem 3.31’de verilmiştir. Burada i toplam giriş sayısıdır. Sadece j . hücreye değil girişe bağlı 1. katmandaki her bir hücreye giriş olarak kullanıldığından 1. katmanın giriş $(x_{11}, x_{21}, \dots, x_{i1})$ değerleri herhangi bir hücreye özel değildir. Ancak $x_{11}, x_{21}, \dots, x_{i1}$ değerlerinin çarpıldığı j . hücrenin ağırlık $(w_{j1}, w_{j2}, \dots, w_{ji})$ değerleri ve j . hücrenin eşik (b_{j1}) değeri ise j . hücreye özeldir. Bu b_{j1} değeri pozitif veya negatif olabilir. İçerideki matematiksel hesaplar yapıldıktan sonra bulunan değer bir aktivasyon fonksiyonundan geçirilerek j . hücrenin çıkış (y_{j1}) değeri elde edilir. Denklemi verilen j . hücre Şekil 3.23’te gösterilmiştir, j yerine konacak değerler ile bütün hücreler elde edilebilecektir.



Şekil 3.23. YSA’nın ilk gizli katmanındaki j . yapay sinir hücresi

Bütün x , w , b ve y ifadeleri için kullanılan indislerden birincisi ilerde matris ifadesinde satır numarasına ikincisi ise sütun numarasına uygun olması için seçilmiştir. Bu durumda, ikinci indislerde sabit 1 olan değerler, ifadelerin sütun vektörü olarak kullanılacağını vurgulamak içindir. Farklı hücre modelleri de bulunmasına rağmen genellikle yapay sinir hücreleri oluşturmak için kullanılan mantık bu şekildedir.

3.3.2. Çeşitleri

Yapay sinir hücreleri bir araya getirilerek çeşitli ağ mimarileri oluşturulmaktadır. Bir sinir ağının hücrelerinin yapılandırılma şekli, ağı eğitmek için kullanılan öğrenme algoritması ile de yakından ilişkilidir. Dolayısıyla, sinir ağlarının tasarımında kullanılan öğrenme algoritmalarının etkisi söz konusu olabilir. Öğrenme algoritmaları daha aşağılarda ele alınmıştır. Ağ mimarileri genel olarak, temelde üç farklı ağ mimarisi sınıfı olarak tanımlanabilir. Bunlar *tek katmanlı ileri beslemeli ağlar*, *çok katmanlı ileri beslemeli ağlar* ve *tekrarlayan sinir ağlarıdır* (Haykin, 2009). Bunlardan çalışmamızla daha alakalı olan tek ve çok katmanlı ileri beslemeli ağlar başlıklar halinde anlatılmıştır.

Girişlerin bulunduğu katman *giriş katmanı* olarak isimlendirilir ve bu katmanda hücre bulunmaz ancak çok fazla sayıda giriş bulunabilir. Giriş katmanı tek katmanlı ağlarda *çıkış katmanına* bağlanmaktadır. Çok katmanlı ağlarda ise ayrıca *gizli katman* olarak isimlendirilen katmanlar bulunmaktadır. Bu ağlarda ise giriş ilk gizli katmana bağlanmaktadır. Ancak her iki durumda da giriş katmanına bağlı katmanda yapılan hesaplamalar değişmeyecektir. Tek katmanlı ağlarda çıkış katmanı hesaplamaları ve çok katmanlı ağlarda ise ilk gizli katman hesaplamaları aynı olduğundan Tek katmanlı ileri beslemeli ağlar başlığında ortak olarak anlatılmıştır.

Tek katmanlı ileri beslemeli ağlar

Tek katmanlı ileri beslemeli ağlarda hesaplama yapılan hücreler sadece çıkış katmanında yer almaktadır. Katmanlı bir ağın en basit biçimidir ve giriş katmanı doğrudan çıkış katmanına bağlanmaktadır (Haykin, 2009). Bir YSA'da yer alan çıkış veya gizli katmanlar çok fazla sayıda hücre içerebilir. Hem çok giriş hem çok hücre

olması yüzünden gereken bütün bu hesaplamalar bütün katmanlar için matris yöntemiyle genelleştirilebilir.

$$f\left(\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1i} \\ w_{21} & w_{22} & \cdots & w_{2i} \\ \vdots & \vdots & \ddots & \vdots \\ w_{j1} & w_{j2} & \cdots & w_{ji} \end{bmatrix} \times \begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{i1} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{21} \\ \vdots \\ b_{j1} \end{bmatrix}\right) = \begin{bmatrix} y_{11} \\ y_{21} \\ \vdots \\ y_{j1} \end{bmatrix} \quad (3.32)$$

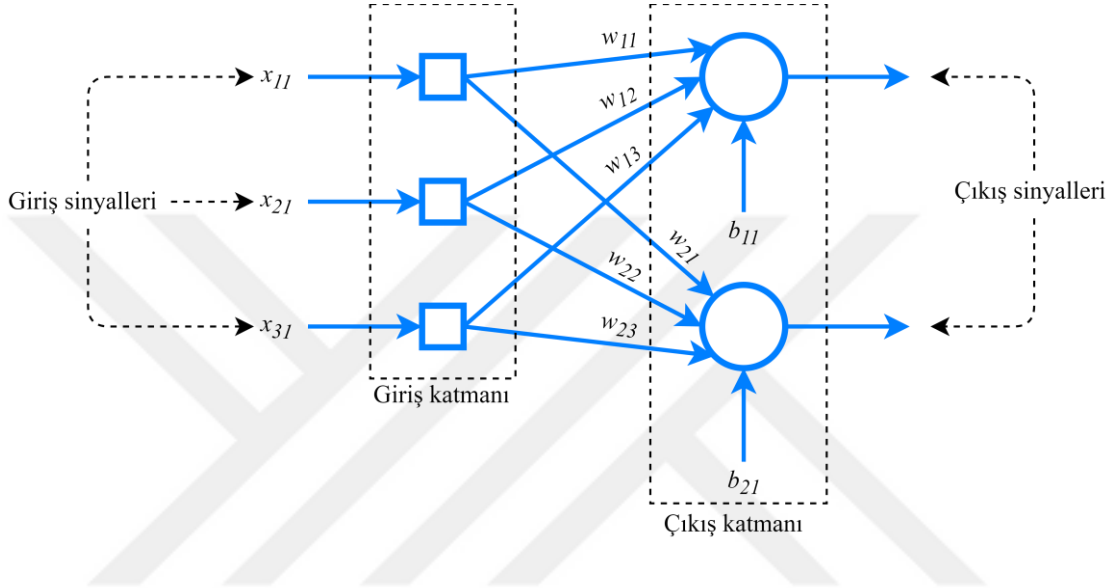
Denklem 3.32’de önceden verilen tüm eşitliklerin çıkış veya ilk gizli katman için oluşturulan matris ifadesi verilmiştir. Burada yer alan tüm harf ifadeleri Yapay sinir hücresi bölümünde anlatılmıştı (bkz. s.57). O ifadeler sadece burada matris formatına göre satır ve sütun numarasına dikkat edilerek bu katmandaki bütün hücreler için yerleştirilmiştir.

$$f_1(W_1 \times Y_0 + B_1) = Y_1 \quad (3.33)$$

Denklem 3.33’te, Denklem 3.32’de yer alan matris ifadeleri için çıkış veya ilk gizli katmanla uyumlu olacak şekilde kısa karşılık olarak 1 alt indisi kullanılarak hesaplama formülü verilmiştir. Ayrıca matris olduğunu vurgulamak için büyük harfler tercih edilmiştir. Burada $j \times i$ boyutlu 1. katmanın ağırlık (W_1) matrisi ile $i \times 1$ boyutlu 1. katmanın giriş (Y_0) matrisi çarpımı ile elde edilen $j \times 1$ boyutlu matris aynı boyutta 1. katmanın eşik (B_1) matrisi ile toplanarak matrisin bütün elemanları tekerli olarak fonksiyondan geçirilerek 1. katmanın çıkış (Y_1) matrisi elde edilmektedir. Sadece tek katman olduğu durum için çıkış katmanı zaten son katman olacağından çıkış hücresi sayısı M ile son katman hücre sayısı j eşit olur ($M = j$). Ayrıca çok katmanlı ağlarda yapılacak alt indis numaralandırmaya uygun olması için YSA’nın ilk girişleri olan x_{11} , x_{21} , $\dots$, x_{i1} değerlerinin bulunacağı matris bir istisna olarak X yerine Y_0 olarak isimlendirilmiştir.

Şekil 3.24’te tek katmanlı ileri beslemeli ağ yapısı için örnek gösterilmiştir. En solda yer alan $i = 3$ tane giriş kare ile ve en sağda yer alan $M = 2$ tane çıkış hücresi ise yuvarlak ile gösterilmiştir. Girişler ve hücreler gösterilen oklarla birbirine bağlanmış ve hesaplamalar yapılırken bu oklar üzerinde yer alan W_1 matrisinin elemanları olan w_{11} , w_{12} , $\dots$, w_{23} ağırlık değerleri güncellenerek en uygun hale getirilip en doğru çıkış

bulunmaya çalışılmaktadır. Çıkış hücreleri bulunan son ağırlık ve eşik değerleri yardımı ile giriş hücrelerinden gelen işaretleri işleyerek çıkış işaretlerini oluşturacaktır. Burada gösterilen ağ yapısı tamamen bağlı olarak gösterilmiştir. Yani, çıkış katmanındaki her bir hücrenin giriş katmandaki tüm hücelere bağlı olduğu anlamına gelir. Tamamen bağlı ağ yapıları matrisli ağ hesaplamalarında kolaylık sağlamaktadır.



Şekil 3.24. Tek katmanlı ileri beslemeli ağ yapısı

Çok katmanlı ileri beslemeli ağlar

Çok katmanlı ileri beslemeli ağlar ardı ardına bağlanmış *tek katmanlı ileri beslemeli ağların* uygun bir şekilde birleştirilmesiyle meydana gelir. Ağın yapısında bir *giriş katmanı* bulunmaktadır ve bu katmanda hücre yoktur. Ayrıca bir *çıkış katmanı* mutlaka bulunmaktadır. Bu katmanların arasında bir veya daha fazla katman bulunmaktadır. Bu katmanlar *gizli katman* veya *ara katman* olarak isimlendirilmektedir. Gizli katmanlardaki hücrelerin yaptıkları işlemler çıkış katmanı ile aynıdır. Bir veya daha fazla gizli katman eklemek, ağın girişinden daha üst düzey istatistikler çıkarılmasına yardım edebilmektedir (Haykin, 2009). Çok katmanlı ağ yapıları en az üç katman içerdiğinden doğrusal olmayan problemlerin çözümünü kolaylaştırabilmektedir. Bağlantılar genellikle farklı katmanlar arasındaki hücreler arasında bulunmaktadır.

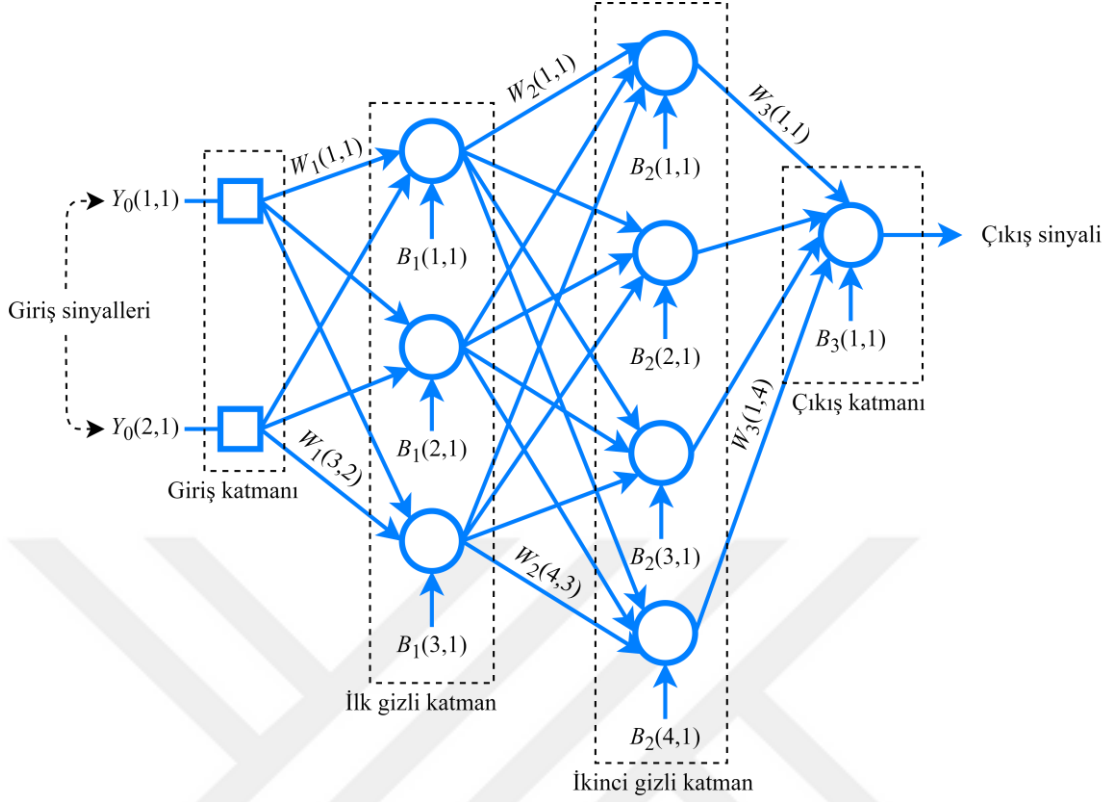
Çok katmanlı ileri beslemeli ağ hesaplamalarında ise yine genelleştirme yapabiliriz. Tek katmanlı ileri beslemeli ağlar bölümünde çıkış veya ilk gizli katman için yapılan hesaplamalar verilmişti (bkz. s.59). O hesaplamalar tek katmanlı ağ olması durumunda çıkış katmanı hesaplamaları olacaktır. Ancak çok katmanlı ağ olması durumunda ise ilk gizli katman hesaplamaları olacaktır. Çok katmanlı ağın yapısında fazladan gizli katman olması durumunda ise ilk gizli katmanın çıkışı olan Y_1 matrisi diğer ikinci gizli katmanın girişi olarak kullanılacaktır. Matrislerin boyutları yine ilk katmanda olduğu gibi çarpım ve toplam için uyumlu olacaktır.

$$\begin{aligned}
 f_1(W_1 \times Y_0 + B_1) &= Y_1 \\
 f_2(W_2 \times Y_1 + B_2) &= Y_2 \\
 &\vdots \\
 f_k(W_k \times Y_{k-1} + B_k) &= Y_k
 \end{aligned} \tag{3.34}$$

Denklem 3.34'te k tane katman olması durumu için genelleştirilmiş matris formülü verilmiştir. Katmanlardan $k - 1$ tanesi gizli katman ve 1 tanesi çıkış katmanıdır. Giriş katmanında hesaplama yapılmadığından sayıya dahil değildir. Bu durumda 1. katman ilk gizli katmandır ve bu tarz devam eder. Sondan bir önceki $(k - 1)$. katman $(k - 1)$. gizli katmandır. En son k . katman ise çıkış katmanıdır. Her katmanda benzer hesaplamalar yapılır, $j = \{1, 2, \dots, k\}$ olmak üzere herhangi bir j . katmanında yapılacak olan hesaplamalar şöyle gerçekleşir. İlk olarak j . katmanın ağırlık (W_j) matrisi ile j . katmanın giriş (Y_{j-1}) matrisi çarpılır ve çarpıma j . katmanın eşik (B_j) matrisi eklenir. Sonra matrisin tüm elemanları j . katmanın aktivasyon (f_j) fonksiyonundan geçirilir ve j . katmanın çıkış (Y_j) matrisi elde edilir. Buradaki Y_{j-1} matrisi aynı zamanda bir önceki katmanda hesaplanan $(j - 1)$. katmanın çıkış (Y_{j-1}) matrisidir. Son katman hariç her zaman bir katmanın çıkışı bir sonraki katmanın girişi olur. En son satırda yer alan k numarası ile gösterilen katman ise çıkış katmanıdır. Çıkış katmanında M tane çıkış hücresi olduğunu düşünelim. Bu durumda son hesaplama ile elde edilen $M \times 1$ boyutlu k . katmanın çıkış (Y_k) matrisi YSA'nın gerçek çıkış (Y_G) matrisi olacaktır. Bu matrisin her elemanı ilgili indisine göre bir çıkışın işaretidir. Ek olarak, burada YSA'nın sistem için kontrol işareti üretmesi sırasında yapılan hesaplamalar verilmiştir. Bu yüzden burada eğitim aşamasında bahsedilecek P_E örnek sayısı burada 1 tanedir. YSA eğitimi

sırasında ise bu P_E değeri örnek sayısı kadar olacaktır. Bu durumda çıkış $M \times P_E$ boyutunda olacaktır. Ancak bu matris hesaplamalarını değiştirmemektedir. Sistemlerde çok fazla hücre, giriş veya örnek sayısı olması durumunda matris yöntemiyle genelleştirme büyük kolaylık sağlamaktadır.

Rastgele olarak dizilmiş örnek bir YSA yapısı Şekil 3.25'te gösterilmiştir. En solda yer alan $i = 2$ tane giriş kare ile gösterilmiştir. Girişe bağlı $k = 3$ tane katman bulunmaktadır. En sağdaki çıkış katmanında yer alan $M = 1$ tane çıkış hücresi ve ortadaki 2 gizli katmanda yer alan hücreler yuvarlak olarak gösterilmiştir. İlk gizli katmanda 3 hücre ikinci gizli katmanda ise 4 hücre bulunmaktadır. Tüm bu hücreler gösterilen oklarla birbirine bağlanmıştır. Bütün oklar üzerinde ağırlık değerleri bulunmakta ve hesaplamalar yapılırken bu oklar üzerinde yer alan ağırlık değerleri güncellenerek en uygun hale getirilip en düşük hata çıkış bulunmaya çalışılmaktadır. Bütün ağırlık değerleri şekil üzerinde karmaşa oluşturmamak için gösterilmemiştir. Gösterimdeki $W_2(4,3)$ ifadesi, W_2 matrisindeki 4. satır, 3. sütun elemanıdır ve ikinci gizli katmandaki 4. hücreye, ilk gizli katmandaki 3. hücreden giden ağırlıktır. Diğer ifadelerde bu tarzda gösterilmiştir. Burada gösterilen ağ tamamen bağlıdır. Yani, ağın herhangi bir katmanındaki bir hücrenin önceki katmandaki tüm hücrelere bağlıdır. Bu durumda oklar ile nereye gittiği gösterilen bütün sinyaller ilgili hücre içerisine girdiğinde hücre içerisinde bir ağırlık değeri ile işleme tabi tutulacaktır.



Şekil 3.25. İki gizli katman içeren çok katmanlı ileri beslemeli ağ yapısı

3.3.3. Öğrenme süreçleri

Yapay sinir hücreleri öğrenmeyi yapabilmek ve öğrenilmiş bilgiyle işlem yapabilmek için gerçek hayattaki sinir hücrelerinden esinlenilerek tasarlanmıştır. Gerçek hayatta herhangi bir şeyi öğrenme süreci değişik şekillerde gerçekleşebildiği gibi tasarlanan bu yapay sinir hücreleri ile oluşan YSA'da gerçekleşen öğrenme süreci değişik şekillerde gerçekleşebilir. Öğrenme süreçleri başlıca iki kategoriye ayrılarak aşağıda anlatılmıştır. Bu süreçlerden ilki *öğretmenli öğrenme*dir. İkincisi ise *öğretmensiz öğrenme*dir. YSA tarafından gerçekleştirilen bu farklı öğrenme biçimleri, insan öğrenmesine benzemektedir (Haykin, 2009).

Bu yöntemlerden ilki olan *öğretmenli öğrenme* aynı zamanda *denetimli öğrenme* olarak da isimlendirilebilmektedir. Bu öğrenme sürecinde öğretmenin çevre hakkında bilgi sahibi olduğunu ve bu bilginin bir giriş çıkış örnek veri seti ile temsil edildiğini düşünebiliriz. Bununla birlikte, çevre sinir ağı tarafından bilinmemektedir. Öğretmenin

ve sinir ağının her ikisinin de aynı ortamdan çekilmiş bir eğitim vektörüne maruz kaldığı durumda yerleşik bilgi sayesinde, öğretmen sinir ağına o eğitim vektörü için istenen cevabı sağlayabilir. Dolaylı olarak istenen cevaba göre eğitim yapıldığından bu cevap sinir ağının ulaşabileceği en iyi cevap olarak düşünülebilir. Çünkü ağın parametreleri, eğitim vektörünün ve hata sinyalinin birleşik etkisi altında ayarlanmaktadır. Hata sinyali, istenen cevap ile ağın gerçek cevabı arasındaki fark olarak tanımlanmaktadır. Ağın parametrelerinin ayarlanması, sonunda sinir ağının öğretmeni taklit edebilmesini sağlamak için iterasyonlu olarak yani sürekli tekrarlayan bir döngü yardımıyla gerçekleştirilir. Taklidin istatistiksel bakış açısı ile en iyi şekilde yapıldığı varsayılır. Bu şekilde, öğretmen için mevcut olan çevre bilgisi, sinir ağına eğitim yoluyla aktarılır ve uzun süreli hafızayı temsil eden sabit ağırlık ve eşik değerleri olarak depolanır. Bu durumda elde edilen son ağırlık ve eşik değerleri ile sinir ağı öğretmeni olmadan kendi cevaplarını üretebilmektedir (Haykin, 2009). Bu yapı oluştuktan sonra eğer çok fazla tekrarlama yapılmamışsa sinir ağı öğretmenin bilgilerini tamamen ezberlemez ve yeni gelen verileri düzgün bir şekilde kontrol edebilir. Ters durumda ise beklenen çıkış oluşmayabilir bu yüzden tekrarlama sayısı oldukça önemlidir.

Öğretmensiz öğrenme bu yöntemlerden ikincisidir ve adından anlaşılacağı gibi öğrenme sürecini denetleyecek bir öğretmen yoktur. Yani, ağ tarafından öğrenilecek fonksiyonun etiketli örnekleri yoktur. Bu ikinci yöntem için iki alt kategori tanımlanabilir bunlar *pekiştirmeli öğrenme* ve *özörgütlemeli öğrenme*dir (Haykin, 2009).

Öğretmensiz öğrenmenin bir çeşidi *pekiştirmeli öğrenme* algoritmasıdır. Bu algoritmada bir hedefe ulaşabilmek için ödüllendirilme veya cezalandırılma sistemi oluşturularak eğitim gerçekleştirilmektedir. Bu işlem için genellikle tablo kullanımı tercih edilmektedir. Her bir kombinasyon için istenen ödüllendirme değerleri belirlenir. En yüksek ödül değerine sahip durumlar içerisinde istenilen davranışı ortaya çıkaran kontrol işareti seçilmektedir. Bu öğrenme yöntemi modele bağımlı veya bağımsız olarak gerçekleştirilebilmektedir (İmamoğlu ve ark., 2009).

Öğretmensiz öğrenmenin diğer bir çeşidi *özörgütlemeli öğrenme*dir. Özörgütlemeli öğrenmede, öğrenme sürecini denetleyecek dışarıdan bir öğretmen yoktur. Bunun yerine

ağın parametreleri sonuçlardan elde edilen bir ölçüye göre optimize edilir. Özörgütlemeli öğrenmeyi gerçekleştirmek için rekabetçi bir öğrenme kuralı kullanılabilir. Örneğin, bir giriş katmanı ve bir rekabetçi katman olmak üzere iki katmandan oluşan bir sinir ağı kullanılabilir. Giriş katmanı mevcut verileri alır. Rekabetçi katman, girdi verilerinde bulunan özelliklere yanıt verme "fırsatı" için birbirleriyle bir öğrenme kuralına uygun olarak rekabet eden yapay sinir hücrelerinden oluşur. Ağ, en basit şekliyle "kazanan her şeyi alır" stratejisine göre çalışır. Böyle bir stratejide, en büyük toplam girdiye sahip yapay sinir hücresi rekabeti “kazanır” ve açılır; sinir ağındaki diğer tüm yapay sinir hücreleri kapanır (Haykin, 2009).



4. MATERYAL ve YÖNTEM

4.1. MATLAB ve Simulink

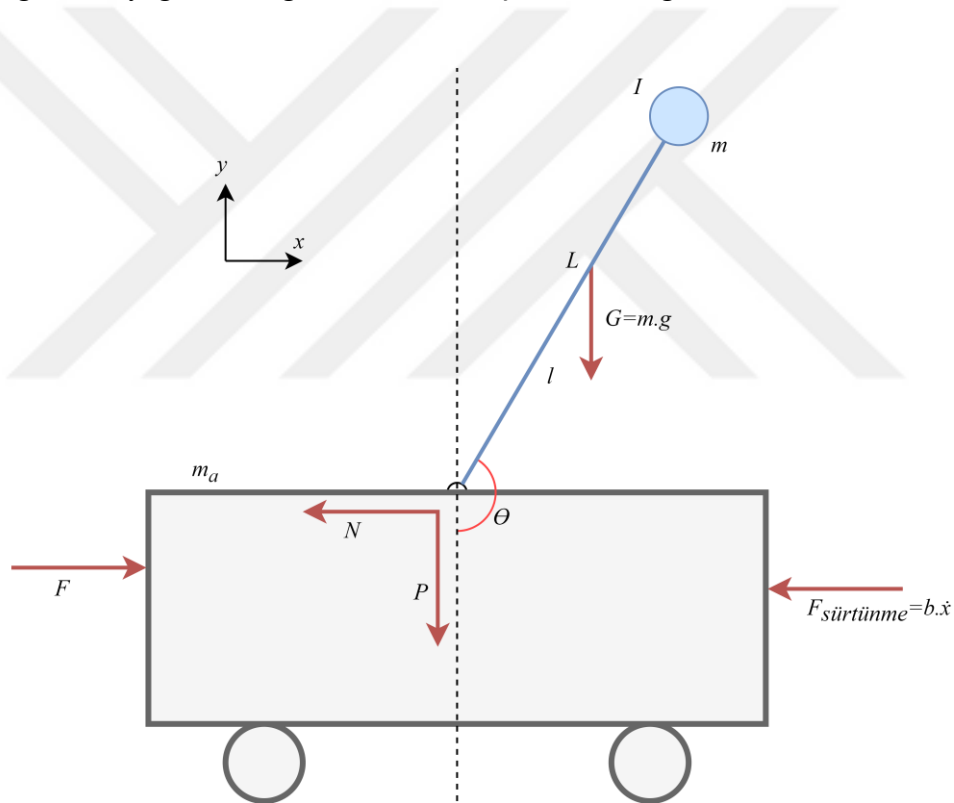
Bu deneysel çalışmada yer alan benzetim sistemi tasarımını ve denetleyicilerin tasarımlarını gerçekleştirebilmek için Intel® Core™ i5-7200U model 2,50 GHz hızında işlemci ve 8 GB ram içeren bir bilgisayar kullanılmıştır. Bu bilgisayarda Windows 10 pro işletim sistemi üzerine kurulu MATLAB R2017b yazılımı Simulink eklentisi ile birlikte kullanılmıştır.

Tez çalışmasında önerdiğimiz yöntemin performansını göstermek amacıyla MATLAB yazılımında yer alan örneklerden Simulink eklentisinde görselli bir şekilde tasarlanmış olan ters sarkaç sistemi modeli kullanılmıştır. Bu model üzerinde ters sarkaç sisteminin kontrolü için öncelikle MATLAB yazılımında yer alan *Bulanık Mantık Araç Kutusu* (*Fuzzy Logic Toolbox*) yardımı ile BMD'ler tasarlanmıştır. Hem sarkacın dik olarak ayakta durması hem de arabanın istenen konuma gitmesi hedeflenmiş ve buna göre Simulink eklentisinde bir benzetim sistemi tasarımı yapılmıştır. MATLAB yazılımında yer alan *Derin Öğrenme Araç Kutusu* (*Deep Learning Toolbox*) yardımı ile YSA tasarımları yapılmıştır. Ayrıca benzetim sistemi çalıştırıldığında denetleyiciler arasında oluşacak hız farkını gösterebilmek için bilgisayarın benzetimi hesaplama süresini ölçecek bir kronometre tasarlanmıştır.

4.2. Arabalı Ters Sarkaç Modeli

Günümüzde insan vücudunu taklit eden mekanizmalar yaygın bir şekilde tercih edilebilmektedir. Bunun sebebi insan vücudunun mükemmelliği olmuştur, bir robotun bir insan vücudunu tam olarak temsil edebilmesi çok uzun sürecek gibi görünmektedir. Robotikteki konulardan biri, ters bir sarkacı dengeleme hareketidir. Bu denge sorununun çözümü yapıldığında bu tecrübe ve kazanım birçok gerçek hayat probleminde benzer sorunların çözümüne yardımcı olacaktır. Ters sarkacın dengede durabilmesini sağlamak için bir kontrol sistemine ihtiyaç vardır. Basit matematiksel modellemeye uygun olmayan denge kontrolü gibi uygulamalarda sayısız ve çeşitli kontrol sorunları vardır. Bilindiği gibi, temel sistemin matematiksel modeline dayanan

Ters sarkaç, tasarlanmış bir denetleyicinin kontrol performansını test etmek için birçok çalışmaya konu olan klasik bir kontrol problemidir. Şekil 4.1’de görüldüğü gibi, x eksenini boyunca doğrusal hareket edebilen bir araba ve menteşe ile x ve y eksenlerinde açısal olarak hareket edebilen bir çubuktan oluşmaktadır. Bu sistem, avuç içinde dik tutulan bir çubuğa benzetilebilir. Ters sarkaç, doğrusal olmayan bir transfer fonksiyonuna sahiptir ve bu nedenle kolayca kararsız hale gelebilir. *Arabalı ters sarkaç* serbest gövde diyagramının genel bir temsili Şekil 4.1’de gösterilmektedir.



Şekil 4.1’de ve aşağıdaki denklemlerde kullanılan semboller ve açıklamaları ile benzetim çalışmalarında kullanılan değerler ve birimleri Çizelge 4.1’de verilmiştir. Değerler sütununda bilinen değerlerin yanı sıra bilinmeyen değerlerin bağlı olduğu etkenler de verilmiştir.

Çizelge 4.1. Çalışmada kullanılan semboller, açıklamalar, değerler ve birimler

Sembol	Açıklama	Değer	Birim
g	Yerçekimi ivmesi	9.8	m/s^2
L	Sarkacın uzunluğu	0.61	m
l	Sarkacın kütle merkezinin arabaya uzaklığı	0.305	m
m_a	Arabanın kütlesi	0.455	kg
m	Sarkacın kütlesi	0.21	kg
G	Sarkacın ağırlığı	$m \times g$ ile bulunuyor	N
I	Sarkacın atalet momenti	$1/3 \times m \times l^2$ ile bulunuyor	$kg \times m^2$
F, u ve U	Arabaya uygulanan giriş kuvveti	Denetleyici sistemin durumuna göre uyguluyor	N
τ	Tork	Sistemin durumuna göre oluşuyor	$N.m$
F_s	Sürtünme kuvveti	İhmal edilmiştir	N
b	Sürtünme katsayısı	İhmal edilmiştir	$N \times s/m$
N	x eksenindeki ağırlık bileşeni	θ açısal konumu ile değişiyor	N
P	y eksenindeki ağırlık bileşeni	θ açısal konumu ile değişiyor	N
$\ddot{\theta}$	Sarkacın açısal ivmesi	Denklem yardımı ile bulunuyor	rad/s^2
$\dot{\theta}$	Sarkacın açısal hızı	Sarkacın açısal ivmesinin integrali ile bulunuyor	rad/s
θ	Sarkacın açısal konumu	Sarkacın açısal hızının integrali ile bulunuyor	rad
$\ddot{x}$	Arabanın ivmesi	Denklem yardımı ile bulunuyor	m/s^2
$\dot{x}$	Arabanın hızı	Arabanın ivmesinin integrali ile bulunuyor	m/s
x	Arabanın konumu	Arabanın hızının integrali ile bulunuyor	m

Şekil 4.1’de gösterilen arabalı ters sarkaç serbest gövde diyagramında yer alan semboller kullanılarak Newton’un ikinci yasasına göre aşağıdaki formüller çıkarılmıştır.

$$\ddot{x} = \frac{1}{m_a} \sum_{araba} F_x = \frac{1}{m_a} (F - N - b\dot{x}) \quad (4.1)$$

$$\ddot{\theta} = \frac{1}{I} \sum_{sarkaç} \tau = \frac{1}{I} (-Pl \sin \theta - Nl \cos \theta) \quad (4.2)$$

Sarkacın x ve y yönlerindeki dinamik denklemleri aşağıdaki gibidir.

$$N = \sum_{sarkaç} F_x = m\ddot{x}_s \quad (4.3)$$

$$P = mg + \sum_{sarkaç} F_y = mg + m\ddot{y}_s \quad (4.4)$$

Ancak, sarkacın x ve y açısal konum koordinatları aşağıdaki gibi denklemlerle gösterilebilir. Buradan türevleri yardımı ile sarkaca özel açısal konum koordinatından kurtulabiliriz.

$$x_s = x + l \sin \theta \quad (4.5)$$

$$\ddot{x}_s = \ddot{x} + l\ddot{\theta} \cos \theta - l\dot{\theta}^2 \sin \theta \quad (4.6)$$

$$y_s = -l \cos \theta \quad (4.7)$$

$$\ddot{y}_s = l\ddot{\theta} \sin \theta + l\dot{\theta}^2 \cos \theta \quad (4.8)$$

Denklem 4.3 ve Denklem 4.4 içerisinde yer alan $\ddot{x}_s$ ve $\ddot{y}_s$ değerleri yerine Denklem 4.6 ve Denklem 4.8'de bulunan karşılıkları kullanılarak aşağıdaki denklemler elde edilmiştir.

$$N = m\ddot{x} + ml\ddot{\theta} \cos \theta - ml\dot{\theta}^2 \sin \theta \quad (4.9)$$

$$P = mg + ml\ddot{\theta} \sin \theta + ml\dot{\theta}^2 \cos \theta \quad (4.10)$$

Bulunan bu denklemler ile arabalı ters sarkaç sistemi tasarlanabilir (Michigan ve ark., 2021a). Ayrıca modele bağımlı bazı kontrol uygulamalarında bu sistemin kontrolü için mecburi ekstra işlemler gerekebilir. Ancak çalışmamız modelden tamamen bağımsız olduğu için buradan sonra yapılacak sadeleştirme, doğrusallaştırma ve transfer

fonksiyonu bulma işlemlerine gerek duyulmadığından, hesaplama yükü de ortadan kalkmaktadır.

Denklem 4.1 ve Denklem 4.2 içerisinde yer alan N ve P değerleri yerine Denklem 4.9 ve Denklem 4.10'da bulunan karşılıkları kullanılarak gerekli sadeleştirmeler yapılmış ve aşağıdaki denklemler elde edilmiştir.

$$(m_a + m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F \quad (4.11)$$

$$(I + ml^2)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta \quad (4.12)$$

Yukarıdaki denklemlerde $\sin$ ve $\cos$ fonksiyonlarının neden olduğu doğrusal olmayan durumdan kurtulmak için sarkacın yukarı denge durumunda ($\theta = \pi \approx \pi + \phi$) küçük bir ϕ aralığında değiştiği kabul edilerek aşağıdaki yaklaşımlar ile doğrusallaştırma yapılmıştır.

$$\cos\theta = \cos(\pi + \phi) \approx -1 \quad (4.13)$$

$$\sin\theta = \sin(\pi + \phi) \approx -\phi \quad (4.14)$$

$$\dot{\theta}^2 = \dot{\phi}^2 \approx 0 \quad (4.15)$$

Denklem 4.11 ve Denklem 4.12 içerisinde yer alan $\cos\theta$, $\sin\theta$ ve $\dot{\theta}^2$ değerleri yerine Denklem 4.13, Denklem 4.14 ve Denklem 4.15'te bulunan varsayımlara göre elde edilen karşılıkları kullanılarak aşağıdaki denklemler elde edilmiştir. Ayrıca F yerine u kullanılmıştır. Böylece aşağıdaki denklemler elde edilmiştir.

$$(m_a + m)\ddot{x} + b\dot{x} - ml\ddot{\phi} = u \quad (4.16)$$

$$(I + ml^2)\ddot{\phi} - mgl\phi = ml\ddot{x} \quad (4.17)$$

Denklem 4.16 ve Denklem 4.17'nin Laplace dönüşümü sıfır başlangıç koşullarında aşağıda gösterilmiştir.

$$(m_a + m)X(s)s^2 + bX(s)s - ml\Phi(s)s^2 = U(s) \quad (4.18)$$

$$(I + ml^2)\Phi(s)s^2 - mgl\Phi(s) = mlX(s)s^2 \quad (4.19)$$

Sarkacın ve arabanın Denklem 4.18 ve Denklem 4.19 kullanılarak elde edilen transfer fonksiyonları aşağıda gösterildiği gibi elde edilmiştir (Michigan ve ark., 2021b).

$$TF_{araba}(s) = \frac{X(s)}{U(s)} = \frac{\frac{(I+ml^2)s^2 - mgl}{q}}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(m_a+m)mgl}{q}s^2 - \frac{bmgl}{q}s} \left[\frac{m}{N} \right] \quad (4.20)$$

$$TF_{sarkaç}(s) = \frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s}{s^3 + \frac{b(I+ml^2)}{q}s^2 - \frac{(m_a+m)mgl}{q}s - \frac{bmgl}{q}} \left[\frac{rad}{N} \right] \quad (4.21)$$

Bu denklemlerde q değeri şöyledir.

$$q = (m_a + m)(I + ml^2) - (ml)^2 \quad (4.22)$$

Ters sarkaç sistemine ait yukarıda verilen matematiksel ifadeler ve bunlar sonucunda elde edilen araba ve sarkaca ait transfer fonksiyonları modele bağlı klasik denetleyici tasarımlarında etkili bir kontrol performansı elde etmede çok önemlidir. Benzetim çalışmaları yapmak için elde edilen doğrusal veya doğrusal olmayan denklemler gerekliliğe göre seçilip kullanılarak arabalı ters sarkaç modeli oluşturulabilir. MATLAB yazılımı ile çok entegre çalışabilen Simulink eklentisi ile bu modeli oluşturmak nispeten daha kolaydır. Oluşturulan bu model yine Simulink eklentisinde kullanılarak bir benzetim sistemi oluşturulabilir. Bu tez çalışmasında BMD tasarımları modelden bağımsız olarak gerçekleştirilmiştir. Bu sebeple yukarıda verilen arabalı ters sarkaç modeli sadece Simulink eklentisinde benzetim sonuçlarının elde edilmesi için kullanılmıştır. Bu modeldeki giriş ve çıkışlar gerekli tek ihtiyacımızdır.

4.3. Bulanık Mantık Denetleyicinin Yapay Sinir Ağı ile Modellenmesi

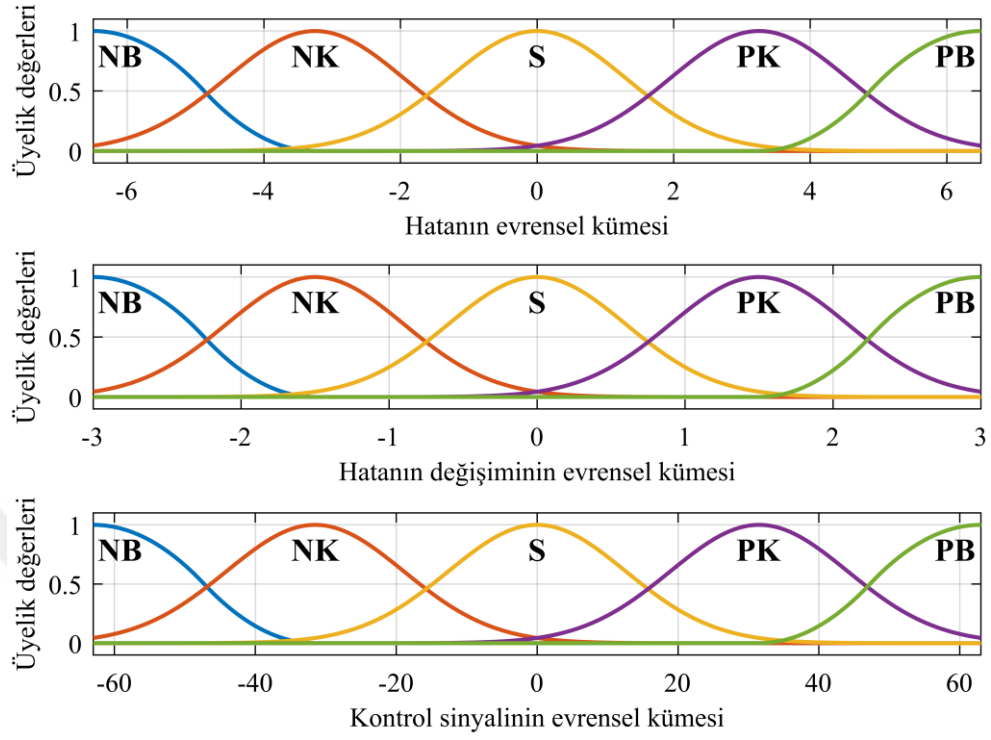
Uzman tecrübesi yardımı ile BMD tasarımları yapıldıktan sonra bu tasarım kullanılmak istenecektir. Ancak kullanılmak istenen BMD tasarımları içerisinde karmaşık ve çok fazla matematiksel hesaplamalar içereceğinden, tasarlanan BMD'nin gerçekleşmesi hem yazılımsal olarak, hem de donanımsal olarak bir takım güçlükler getirecektir. Aynı zamanda denetleyicinin yavaş tepki vermesine yol açacaktır. Bu durum, bilgisayarda MATLAB yazılımında Simulink eklentisi ile yapılan benzetim deneyleri yardımı ile gözlemlenmiştir. Önce BMD'lerin nasıl tasarlandığı anlatılmıştır. İlerleyen başlıklarda ise bu BMD'lerin yaptığı kontrolü yapabilecek YSA'ların nasıl tasarlandığı anlatılmıştır.

4.3.1. Bulanık mantık denetleyici tasarımı

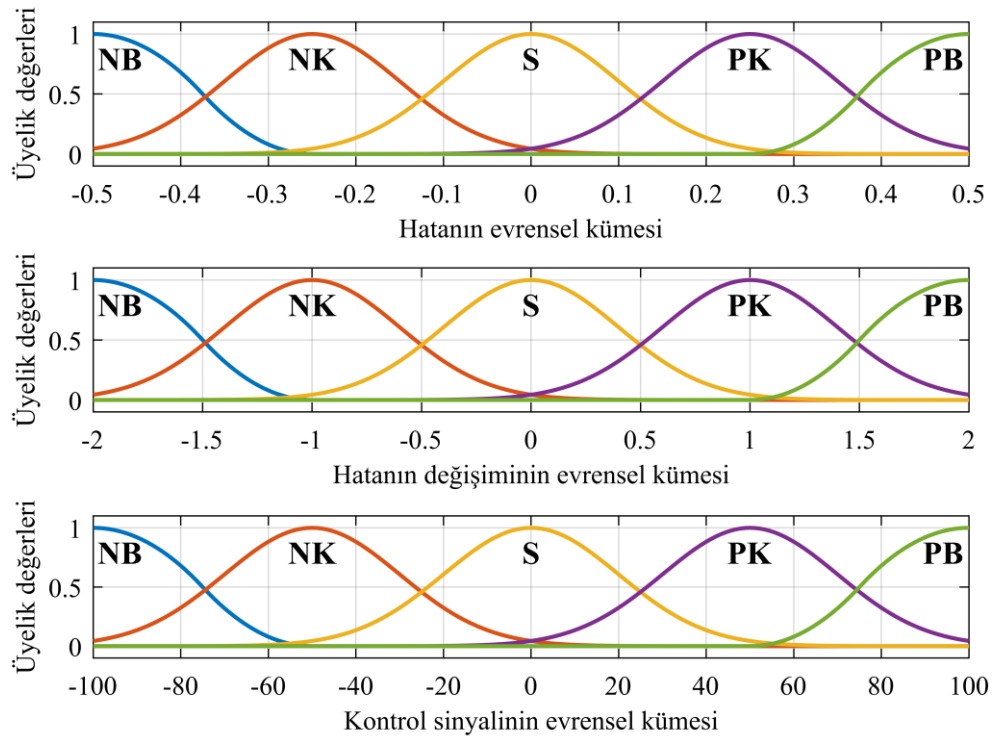
Arabalı ters sarkacı kontrol etmek için BMD_{ARABA} ve $BMD_{SARKAÇ}$ olarak isimlendirdiğimiz iki ayrı BMD kullanılmıştır. Her iki BMD için birbirlerinin genişletilip daraltılmış hali olan üyelik fonksiyonları kullanılmıştır. Yani BMD'lerin önemli olan farkı evrensel küme aralıklarıdır. BMD'lerin tasarımı için kullanılan üyelik fonksiyonlarının evrensel küme aralıkları Çizelge 4.2'de, grafikleri Şekil 4.2 ve Şekil 4.3'te gösterilmiştir. Bu değerler uzman tecrübesi yardımı ile belirlenmiştir. Toplam 5 üyelik fonksiyonu kullanılmıştır. Bunlar: Negatif Büyük (NB), Negatif Küçük (NK), Sıfır (S), Pozitif Küçük (PK) ve Pozitif Büyük (PB) olarak isimlendirilmiştir. NB için Z üyelik fonksiyonu, PB için S üyelik fonksiyonu ve diğerleri için $Gauss$ üyelik fonksiyonu tercih edilmiştir.

Çizelge 4.2. Kullanılan üyelik fonksiyonlarının evrensel küme aralıkları

BMD_{ARABA}	NB	NK	S	PK	PB
x_H	$[-6.5, -3.25]$	$[1.3, -3.25]$	$[1.3, 0]$	$[1.3, 3.25]$	$[3.25, 6.5]$
x_{HD}	$[-3, -1.5]$	$[0.6, -1.5]$	$[0.6, 0]$	$[0.6, 1.5]$	$[1.5, 3]$
U_A	$[-63, -31.5]$	$[12.6, -31.5]$	$[12.6, 0]$	$[12.6, 31.5]$	$[31.5, 63]$
$BMD_{SARKAÇ}$	NB	NK	S	PK	PB
θ_H	$[-0.5, -0.25]$	$[0.1, -0.25]$	$[0.1, 0]$	$[0.1, 0.25]$	$[0.25, 0.5]$
θ_{HD}	$[-2, -1]$	$[0.4, -1]$	$[0.4, 0]$	$[0.4, 1]$	$[1, 2]$
U_S	$[-100, -50]$	$[20, -50]$	$[20, 0]$	$[20, 50]$	$[50, 100]$



Şekil 4.2. Üyelik fonksiyonları (BMD_{ARABA})



Şekil 4.3. Üyelik fonksiyonları (BMD_{SARKAÇ})

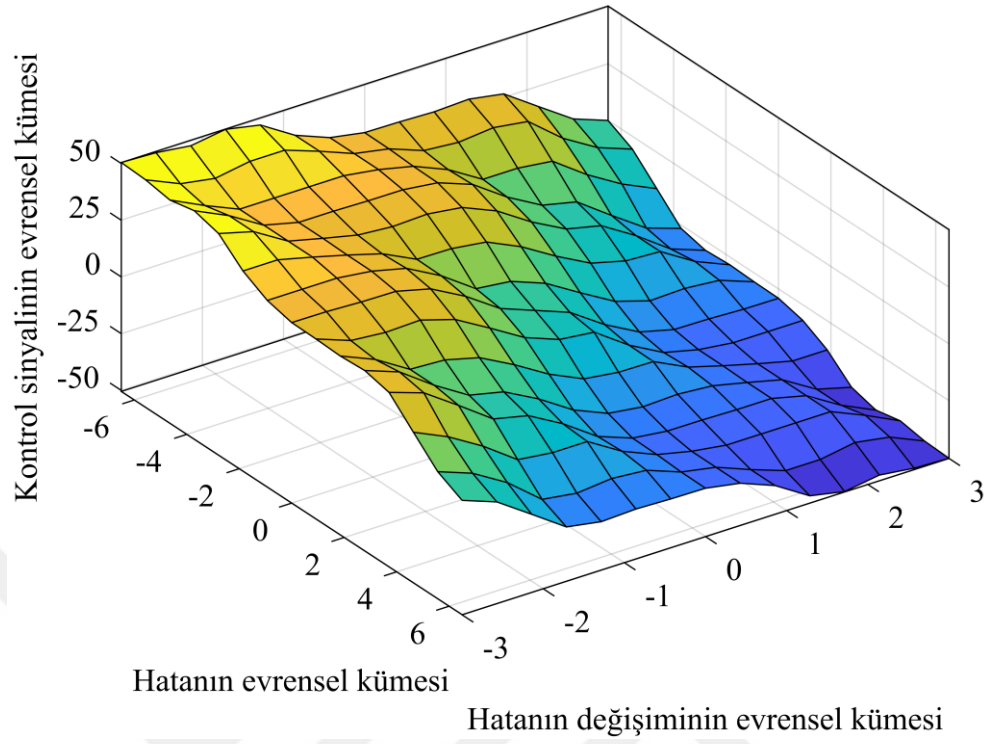
BMD'ler Mamdani metoduna göre tasarlanmıştır. Bu metotta girişlerin yanı sıra çıkış için de üyelik fonksiyonları kullanılır ve BMD'nin çıkışı yani kontrol sinyali için kullanılan üyelik fonksiyonları da Şekil 4.2 ve Şekil 4.3'te gösterilmiştir.

Kullanılan her iki BMD için aynı kural tabanı kullanılmıştır. Verilen üyelik fonksiyonlarına göre oluşturulan BMD'lerin kullandığı kurallar Çizelge 4.3'te gösterilmiştir. Her farklı çıkış için çıkış üyelik fonksiyonunda yer alan girişler ile aynı isimlendirmeye sahip 5 üyelik fonksiyonundan hangisine denk geldiği gösterilmiştir. Burada hata ve hatanın değişiminin durumlarına göre bulunan 25 sonuç gösterilmiştir. Her bir kuralın EĞER şartı bölümünde yer alan hata ve hatanın değişimi girişleri *bulanık kesişim* işlemi ile birleştirilmiştir. *Bulanık kesişim* işlemi için *min* operatörü kullanılmıştır. Her bir kuralın O HALDE bölümünün çalıştırıldığı *Implication* adımında *min* operatörü kullanılmıştır. Bütün kuralların birleştirildiği *Aggregation* adımında *max* operatörü kullanılmıştır. Durulaştırma için ise *centroid* yöntemi kullanılmıştır.

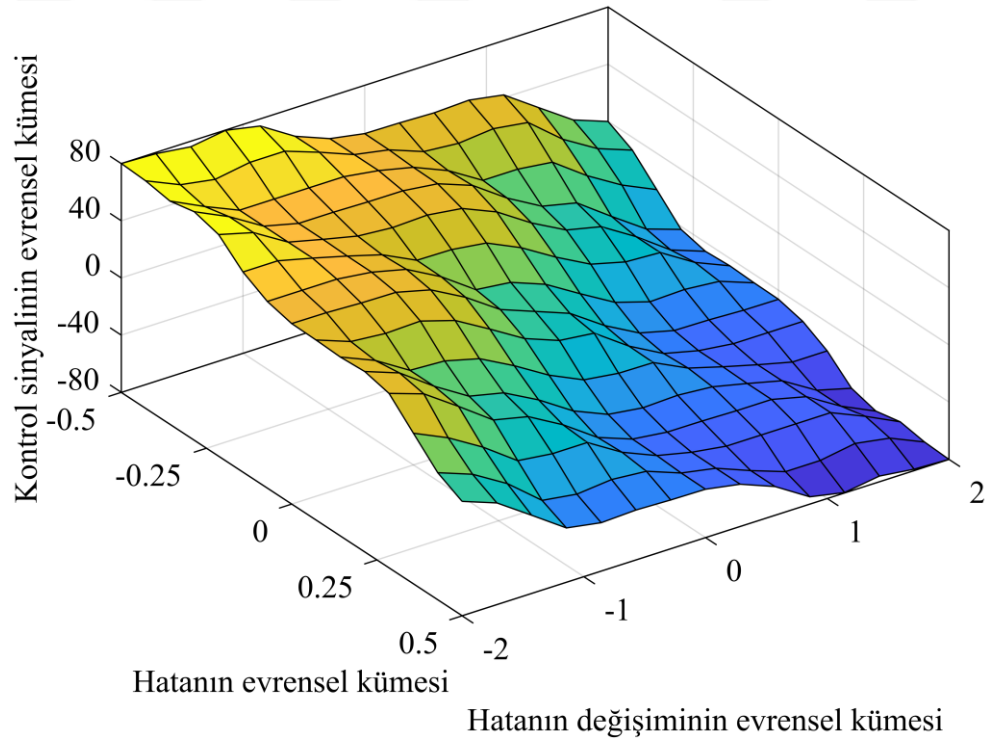
Çizelge 4.3. BMD'lerin kural tablosu

Üyelik fonksiyonları		x_H, θ_H				
		NB	NK	S	PK	PB
x_{HD}, θ_{HD}	NB	PB	PB	PK	PK	S
	NK	PB	PK	PK	S	NK
	S	PK	PK	S	NK	NK
	PK	PK	S	NK	NK	NB
	PB	S	NK	NK	NB	NB

Verilen üyelik fonksiyonları ve kurallara göre giriş değişkenleri ile çıkış değişkenleri arasındaki ilişkiyi gösteren *yüzey görünümü* (surface view) grafikleri, Şekil 4.4 ve Şekil 4.5'te gösterilmiştir.



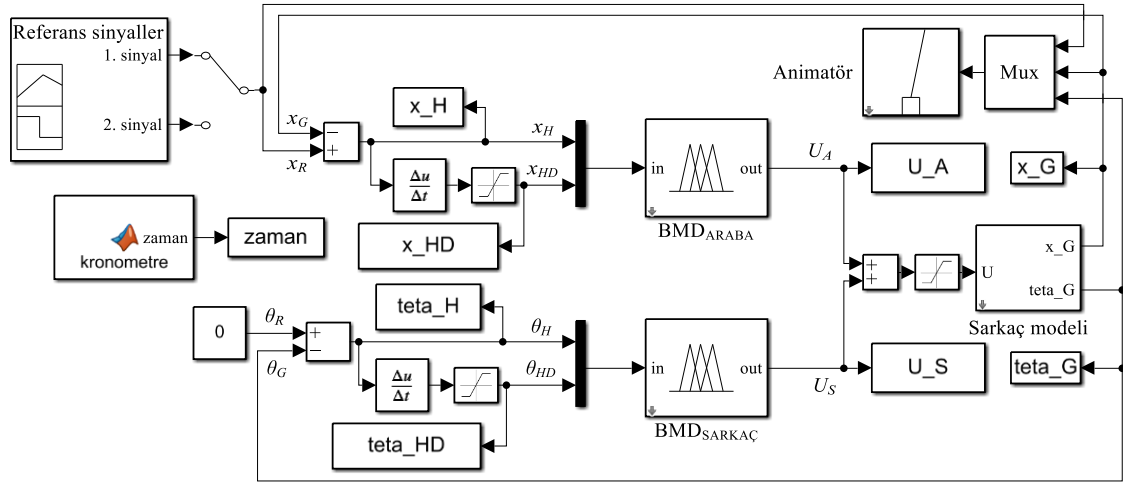
Şekil 4.4. Yüzey görünümü (BMD_{ARABA})



Şekil 4.5. Yüzey görünümü (BMD_{SARKAÇ})

İki BMD'nin ikişer tane girişi bulunmaktadır. BMD_{ARABA} için ilk giriş *arabanın konumunun hatası* (x_H) sinyalini bulabilmek için *arabanın referans konumu* (x_R) sinyalinden ters sarkaç sisteminin çıkışında elde edilen *arabanın gerçek konumu* (x_G) sinyali çıkartılır ($x_H = x_R - x_G$). İkinci giriş *arabanın konumunun hatasının değişimi* (x_{HD}) sinyali ise x_H sinyalinin türevi alınarak bulunur. $BMD_{SARKAÇ}$ için ise ilk giriş *sarkacın açısal konumunun hatası* (θ_H) sinyalini bulabilmek için *sarkacın referans açısal konumu* (θ_R) sinyalinden ters sarkaç sisteminin çıkışında elde edilen *sarkacın gerçek açısal konumu* (θ_G) sinyali çıkartılır ($\theta_H = \theta_R - \theta_G$). İkinci giriş *sarkacın açısal konumunun hatasının değişimi* (θ_{HD}) sinyali ise θ_H sinyalinin türevi alınarak bulunur.

İki BMD'nin birer tane çıkışı bulunmaktadır. BMD_{ARABA} çıkış (U_A) sinyali ve $BMD_{SARKAÇ}$ çıkış (U_S) sinyali son olarak toplanmış ve bu son bulunan toplanmış kontrol sinyali sayesinde *arabaya uygulanan giriş kuvveti* (U) sinyali belirlenmiştir. Bu kuvvetin uygulaması ile sarkaç düşürülmeden arabanın istenen konuma hareket ettirilmesi hedeflenmiştir.



Şekil 4.7. BMD'lerin eğitimi için tasarlanan benzetim sisteminin tamamı

4.3.3. Yapay sinir ağlarının yapıları

Kontrol için aynı mimariye sahip iki YSA kullanılmıştır. Ancak eğitim verileri farklı olduğu için *ağırlık* ve *eşik değerleri* farklıdır. Çok katmanlı ileri beslemeli ağ yapısı

kullanılmıştır. Her bir YSA için iki *giriş*, on *hücre*, bir *çıkış* kullanılmıştır. Toplamda dört *katmanın* içerisinde birçok *aktivasyon fonksiyonu*, *toplam fonksiyonu*, *çarpım fonksiyonu*, *ağırlık* ve *eşik değerleri* vardır. *Giriş katmanında* iki *değişken giriş* bulunmaktadır. On *hücre* ise *gizli katmanlarda* ilk *katmanda* beş ve *ikinci katmanda* beş olmak üzere dağıtılmıştır. Bir *değişken çıkış* ise nihai çıkışın üretildiği *çıkış katmanından* elde edilmektedir. *Gizli katmanların* ve *çıkış katmanının* ağırlık ve eşik değerleri vardır. *Gizli katmanlarda* aktivasyon fonksiyonu olarak *tansig transfer fonksiyonu*, *çıkış katmanında* ise *purelin transfer fonksiyonu* kullanılmıştır. *Normalizasyon* işlemi için *Varsayılan (Default) Ayar* olan *mapminmax* fonksiyonu kullanılmıştır. Bu fonksiyon giriş ve hedef verilerini en küçük değerin -1 ve en büyük değerin 1 olmasını sağlayacak şekilde ölçeklendirmektedir. Eğitimin performansı ise Ortalama Karesel Hata (MSE) değerine göre ölçülmüştür. *Doğrulama* ve *test* işlemleri için *Varsayılan Ayar* olan *dividerand* fonksiyonu kullanılmıştır. Bu fonksiyon eldeki verinin rastgele %70'ini *eğitim* için, rastgele %15'ini *doğrulama* yapabilmek için kullanmaktadır. Geriye kalan %15'in ise eğitime bir katkısı olmaz sadece ağırlık istenildiği gibi öğretilmeyen verilere de cevap verip veremediği *test* edilir. Eğitim için *Levenberg-Marquardt (Trainlm) algoritması* kullanılmıştır. Tasarlanan bu YSA yapısı Çizelge 4.4'te verilmiştir.

Çizelge 4.4. Tasarlanan YSA yapısı

Ağın adı	YSA _{ARABA} ve YSA _{SARKAÇ}
Ağ yapısı	İleri beslemeli
Giriş katmanı hücre sayısı	2
Çıkış katmanı hücre sayısı	1
İlk gizli katman hücre sayısı	5
İkinci gizli katman hücre sayısı	5
Gizli katmanların transfer fonksiyonları	<i>tansig</i>
Çıkış katmanının transfer fonksiyonu	<i>purelin</i>
Veri ön/son işleme (Normalizasyon)	<i>mapminmax</i> $[-1,1]$
Performans fonksiyonu	MSE
Veri bölme oranı (%)	<i>dividerand</i> (70: 15: 15)
Eğitim algoritması	<i>trainlm</i>

4.3.4. Yapay sinir ağlarının eğitilmesi

MATLAB yazılımında arabanın konumu için farklı *referans* değerleri verilmiş ve BMD'ler ile kontrol yapılmıştır. Kontrol esnasında elde edilen BMD'lerin girişleri olan hata ve hatanın değişimi sinyalleri ($x_H, x_{HD}, \theta_H, \theta_{HD}$) ve BMD'lerin ürettiği kontrol sinyalleri (U_A, U_S) *To Workspace* blokları ile Simulink eklentisinden *Çalışma Alanı* içerisine aktarılmıştır. Böylece YSA'ların *veri setleri* oluşturulmuştur. BMD_{ARABA}'nın girişleri olan hata ve hatanın değişimi değerleri (x_H, x_{HD}) YSA_{ARABA}'nın veri setinin giriş değişkenleri olarak ve BMD_{ARABA}'nın ürettiği kontrol sinyali (U_A) ise veri setinin çıkış değişkeni olarak alınmıştır. Benzer şekilde BMD_{SARKAÇ}'ın girişleri (θ_H, θ_{HD}) ve çıkışı (U_S) YSA_{SARKAÇ} için veri seti olmuştur. Hem BMD_{ARABA} hem de BMD_{SARKAÇ}'ı modellemek için aynı mimariye sahip YSA blokları kullanılmıştır.

İlk ağırlık vektörü

YSA tasarımı yapıldıktan sonra ağırlık eğitilmesi gerekir. Eğitimin gidişatı birçok etkene bağlıdır. İçerisinde tüm katmanlardaki ağırlık ve eşik değerlerinin bulunduğu *ilk ağırlık* (W_i) *vektörünü* seçme bu etkenlerden biridir ve bu vektörün her bir elemanı rastgele $[-1,1]$ gibi bir aralıkta seçilebilir. Ayrıca farklı W_i *vektörü* belirleme yöntemleri vardır. Daha uygun bir W_i *vektörü* ile iterasyonun başlatılması daha uygun bir *en iyi ağırlık* (W_b) *vektörüne* yaklaştıracak veya yaklaştırma olasılığını artıracaktır (Teo ve ark., 2001). Rastgele W_i *vektörü* seçildiğinde şansa göre değişmekte birlikte ortalama olarak normalden fazla sayıda deneme yapılarak daha uygun bir W_b *vektörüne* yaklaşılabılır. Çok sayıda algoritma W_i *vektöründen* W_b *vektörüne* ulaşabilmek için yapılan ağ eğitimi için kullanılmaktadır. Bu algoritmalar en uygun W_b *vektörünü* bulabilmek için her iterasyonda sürekli ağırlık ve eşik değerlerini güncellemektedir. Eğitim için MATLAB yazılımında deneme yanılma yöntemi ile amacımıza daha iyi oranda yaklaştırdığını tespit ettiğimiz bir algoritma kullanılmış ve çalışma şekli ilerleyen başlıkta anlatılmıştır.

Levenberg-Marquardt algoritması

Newton algoritmasında kullanılan bazı formüller bir takım ihmaller yapılarak güncellendiğinde, Newton algoritmasında kullanılan *Hessian* (H) *matrisi*, Gauss-

Newton algoritmasında kullanılan *Jacobian (J) matrisi* yardımı ile yaklaşık olarak ifade edilebilir. Bunun faydası, *J matrisini* kullanarak *H matrisi* hesabı için gereken ikinci dereceden türev hesabından kurtulmaktır. Ancak Gauss-Newton algoritması, Newton algoritması ile aynı yakınsama problemine sahiptir. Bu problem yaklaşım ile elde edilen *Yaklaşık Hessian (JJ) matrisinin* de *H matrisi* gibi tersini almanın her zaman mümkün olmayacağıdır. *Levenberg-Marquardt (trainlm) algoritmasında* ise her zaman pozitif olan küçük bir *kombinasyon katsayısı (μ) değeri* ile uygun boyutta bir *birim (I) matris* çarpılarak *JJ matrisine* eklendiğinde oluşan yeni matrisin tersini almak her zaman mümkün olur (Yu ve Wilamowski, 2011).

Trainlm algoritması eğitim için kullanılmıştır. Bu algoritmanın son ağırlık ve eşik değerleri için uygun değeri bulabilmesi birçok farklı parametreye bağlıdır. Çizelge 4.5'te bu parametreler verilmiştir. Parametrelerin MATLAB yazılımındaki adları *Araç Kutusu (Toolbox) parametreleri* sütununda verilmiştir. Diğer sütunlar da çalışmamızda kullanılan simgeler, değerler ve parametrelerin açıklamaları verilmiştir.

Çizelge 4.5. Trainlm algoritması parametreleri

<i>Araç Kutusu parametreleri</i>	<i>Simgeler</i>	<i>Değerler</i>	<i>Açıklamalar</i>
<i>epochs</i>	$i_{maksimum}$	500	İterasyon değeri için maksimum sınır
<i>goal</i>	-	0	MSE_E değeri için minimum sınır
<i>max_fail</i>	$DD_{maksimum}$	50	DD değeri için maksimum sınır
<i>min_grad</i>	$g_{minimum}$	10^{-7}	g değeri için minimum sınır
<i>mu</i>	μ_i	0.001	İlk μ değeri
<i>mu_dec</i>	μ_k	0.1	μ değeri için küçültme çarpanı
<i>mu_inc</i>	μ_b	10	μ değeri için büyütme çarpanı
<i>mu_max</i>	$\mu_{maksimum}$	10^{10}	μ değeri için maksimum sınır
<i>show</i>	-	<i>false</i>	Komut satırı çıktısı oluşturulması
<i>showCommandLine</i>	-	25	<i>show=true</i> ise kaç iterasyonda bir komut satırı çıktısı oluşturulsun
<i>showWindow</i>	-	<i>true</i>	Eğitim arayüz penceresinin gösterimi
<i>time</i>	-	<i>inf</i>	Süre için maksimum saniye sınırı

İleri hesaplama işlemi

İleri hesaplama işleminde ağın ilk veya güncellenmiş hali kullanılarak hata ve performans hesaplanmaktadır. İleri hesaplama işlemi için *ağırlık (W) vektörü* kullanılarak YSA çalıştırılacaktır. YSA *k* tane katmana sahip olsun. Bu durumda *k*. katman çıkış katmanı olur. Ve çıkış katmanında *M* tane *çıkış hücresi* ve *P_E* tane *eğitim örneği* olsun. YSA için bütün katman hesaplamaları yapılıncaya *k*. katmanın çıkışında $M \times P_E$ boyutlu Y_k *matrisi* elde edilecektir (bkz. s.61). YSA'nın *gerçek çıkış (Y_G) matrisi* bu değere eşit olacaktır. Bu matris, *referans çıkış (Y_R) matrisi* ile aynı boyuttadır. İstenen değerden mevcut değer çıkartılarak $M \times P_E$ boyutlu *Eğitim için Hata (H_E) matrisi* Denklem 4.23 ile elde edilmektedir. Ancak hesaplamalarda sütun vektörü gerektiği için bu matris değerlere dokunulmaksızın sütun vektörüne çevrilmekte (Denklem 4.24) ve $(M \times P_E) \times 1$ boyutlu yeni H_E *matrisi* elde edilmektedir. Artık toplam sütun sayısı 1 olduğu için bu durumda H_E *vektörü* olur. Denklem 4.25'te H_E *vektörünün* açık hali gösterilmiştir. Performansın ölçümü için kullanılan *Eğitim için Ortalama Karesel Hata (MSE_E) değeri* Denklem 4.26 ile yardımıyla elde edilmektedir. Hesaplamalarda gereken doğrulama performansı ise aynı yöntemle *eğitim seti* yerine *P_D* tane *doğrulama örneği* bulunan *doğrulama seti* kullanılarak elde edilmektedir. Denklem 4.27 ile bulunan hata matrisinin vektöre dönüşmüş hali (Denklem 4.28) *Doğrulama için Hata (H_D) vektörü* kullanılarak *Doğrulama için Ortalama Karesel Hata (MSE_D) değeri* Denklem 4.29 ile elde edilmektedir. Bu denklemlerde ^T ilgili vektör veya matrisin transpozunu gösterir.

$$H_E = (Y_R - Y_G) \quad (4.23)$$

$$H_E = H_E(:) \quad (4.24)$$

$$H_E = \begin{bmatrix} e_{11_1} \\ e_{11_2} \\ \vdots \\ e_{11_{P_E}} \\ e_{21_1} \\ e_{21_2} \\ \vdots \\ e_{21_{P_E}} \\ e_{M1_1} \\ e_{M1_2} \\ \vdots \\ e_{M1_{P_E}} \end{bmatrix} \quad (4.25)$$

$$MSE_E = \frac{H_E^T \times H_E}{M \times P_E} \quad (4.26)$$

$$H_D = (Y_R - Y_G) \quad (4.27)$$

$$H_D = H_D(:) \quad (4.28)$$

$$MSE_D = \frac{H_D^T \times H_D}{M \times P_D} \quad (4.29)$$

İleri hesaplama işlemi algoritmada her bir iterasyon için bir veya daha fazla sayıda yapılmaktadır. Bu yüzden her bir iterasyon bittiğinde genellikle çoğu değer bir veya daha fazla sayıda güncellenmektedir. Bu çok sayıda değişimi belirtmek için iterasyona özel alt indisler karmaşayı önlemek için gösterilmemiştir. Ancak bu algoritma numaralı adımlarla değişik bir bakış açısı ile anlatılmıştır.

Adım 1: İlk olarak W_i vektörü hesaplanır (bkz. 80) ve bu vektör ile W vektörü oluşturulur ($W = W_i$). Denklem 4.30'da barındırdığı *ağırlık* ve *eşik değerleri* gösterilen bu W vektörü kullanılarak *ileri hesaplama işlemi* yapılır (bkz. s.82) ve Denklem 4.25'te içeriği gösterilen H_E vektörü hesaplanır. Benzer şekilde H_D vektörü hesaplanır. Sonra bu vektörler ile MSE_E ve MSE_D değerleri hesaplanır.

$$W = \begin{bmatrix} b_{11} \\ b_{21} \\ \vdots \\ b_{j1} \\ w_{11} \\ w_{21} \\ \vdots \\ w_{j1} \\ w_{12} \\ w_{22} \\ \vdots \\ w_{j2} \\ w_{1i} \\ w_{2i} \\ \vdots \\ w_{ji} \\ \vdots \\ \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \quad (4.30)$$

Denklem 4.30'da W vektörü içeriği gösterilmiştir. Burada bütün katmanlardaki b eşik değerleri ve w ağırlık değerleri bulunmaktadır. Bütün katmanları temsil ettiği için indisi olmayan W harfi kullanılmıştır. Gösterimde sadece ilk katmandakiler sıralanmıştır ve diğer katmandakiler de aynı mantık ile devam eder. Sırası önemsiz olmakla beraber İleri hesaplama işlemi yapabilmek için ayrılması gerektiğinden bir mantığa göre vektörün birleştirilmesi hesap kolaylığı sağlar. *Jacobian (J) matrisini* sade gösterebilmek için bu vektörde yer alan toplamda N tane ağırlık ve eşik değeri x ifadeli hale çevrilmiştir.

Adım 2: Denklem 4.31'de açık hali verilen J matrisi hesaplanır. Sonra Denklem 4.32, Denklem 4.33 ve Denklem 4.34 yardımı ile yaklaşık *Hessian (JJ) matrisi*, *Gradyan (G) vektörü* ve *gradyan (g) değeri* hesaplanır. Hesaplanan g değeri Çizelge 4.5'te yer alan g_{minimum} değerinden büyükse *Adım 3*, küçükse *Adım 7*'den devam edilir.

$$J = \begin{bmatrix} \frac{\partial e_{11_1}}{\partial x_1} & \frac{\partial e_{11_1}}{\partial x_2} & \dots & \frac{\partial e_{11_1}}{\partial x_N} \\ \frac{\partial e_{11_2}}{\partial x_1} & \frac{\partial e_{11_2}}{\partial x_2} & \dots & \frac{\partial e_{11_2}}{\partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{11_{P_E}}}{\partial x_1} & \frac{\partial e_{11_{P_E}}}{\partial x_2} & \dots & \frac{\partial e_{11_{P_E}}}{\partial x_N} \\ \frac{\partial e_{21_1}}{\partial x_1} & \frac{\partial e_{21_1}}{\partial x_2} & \dots & \frac{\partial e_{21_1}}{\partial x_N} \\ \frac{\partial e_{21_2}}{\partial x_1} & \frac{\partial e_{21_2}}{\partial x_2} & \dots & \frac{\partial e_{21_2}}{\partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{21_{P_E}}}{\partial x_1} & \frac{\partial e_{21_{P_E}}}{\partial x_2} & \dots & \frac{\partial e_{21_{P_E}}}{\partial x_N} \\ \frac{\partial e_{M1_1}}{\partial x_1} & \frac{\partial e_{M1_1}}{\partial x_2} & \dots & \frac{\partial e_{M1_1}}{\partial x_N} \\ \frac{\partial e_{M1_2}}{\partial x_1} & \frac{\partial e_{M1_2}}{\partial x_2} & \dots & \frac{\partial e_{M1_2}}{\partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{M1_{P_E}}}{\partial x_1} & \frac{\partial e_{M1_{P_E}}}{\partial x_2} & \dots & \frac{\partial e_{M1_{P_E}}}{\partial x_N} \end{bmatrix} \quad (4.31)$$

Denklem 4.31’de J matrisinin açık hali verilmiştir. Bu matris H_E vektörünün kısmi türevlerinden oluşmaktadır. İlk sütunu x_1 ağırlık veya eşik değerine göre H_E vektörünün kısmi türevi iken ikinci sütun x_2 ağırlık veya eşik değerine göre H_E vektörünün kısmi türevidir. Ve bu böylece devam eder sonunda, J matrisi hesaplanmış olur (Hagan ve Menhaj, 1994; Yu ve Wilamowski, 2011).

$$JJ = \frac{J^T \times J}{M \times P_E} \quad (4.32)$$

$$G = \frac{J^T \times H_E}{M \times P_E} \quad (4.33)$$

$$g = 2 \times \sqrt{G^T \times G} \quad (4.34)$$

Denklem 4.32’de $N \times N$ boyutunda JJ matrisinin formülü verilmiştir. Denklem 4.33’te ise $N \times 1$ boyutunda G vektörünün nasıl hesaplandığı gösterilmiştir. Burada yer alan H_E

vektörünün Denklem 4.25'te açık hali verilmiştir. Denklem 4.34'te ise g değerinin nasıl hesaplandığı gösterilmiştir. Buralarda P_E eğitim örneklerinin sayısı, M çıkışların sayısı, N ağırlık ve eşik değerlerinin sayısıdır. Ve ^{T} ise ilgili vektör veya matrisin transpozunu gösterir.

Adım 3: Denklem 4.35 yardımı ile *ağırlık güncelleme* (ΔW) vektörü hesaplanıp W vektörüne eklenerek *ağırlık aday* (W_{aday}) vektörü hesaplanır ($W_{aday} = W + \Delta W$). Ve W_{aday} vektörü kullanılarak *adayın Ortalama Karesel Hata* (MSE_{aday}) değeri hesaplanır. Eğer bu değer en iyisi ise ($MSE_{aday} < MSE_E$) *Adım 4*, değilse *Adım 5*'ten devam edilir.

$$\Delta W = -(JJ + \mu \times I)^{-1} \times G \quad (4.35)$$

Denklem 4.32'de JJ matrisinin nasıl hesaplandığı gösterilmiştir. Buradaki *kombinasyon katsayısı* (μ) değeri algoritma boyunca devamlı güncellenecektir ve ilk μ (μ_i) değeri Çizelge 4.5'te verilmiştir. Bu katsayı JJ matrisi ile aynı boyutta *birim* (I) matrisiyle çarpılmaktadır. Denklem 4.33'te ise G vektörünün nasıl hesaplandığı gösterilmiştir. Bütün bu ifadeler yardımı ΔW vektörünün nasıl hesaplandığı Denklem 4.35'te verilmiştir. Denklemdaki $^{-1}$ ilgili matrisin tersini gösterir.

Adım 4: W_{aday} vektörü artık yeni W vektörü olarak kaydedilir ($W = W_{aday}$). Sonra μ değeri Çizelge 4.5'te yer alan μ_k değeri ile çarpılarak küçültülür ($\mu = \mu \times \mu_k$). *Adım 6*'dan devam edilir.

Adım 5: Yeni bir deneme daha yapabilmek için μ değeri Çizelge 4.5'te yer alan μ_b değeri ile çarpılarak büyütülür ($\mu = \mu \times \mu_b$). Eğer μ değeri Çizelge 4.5'te yer alan $\mu_{maksimum}$ değerinden büyükse veya $i_{maksimum}$ değerine ulaşılmışsa *Adım 7*'den devam edilir. Değilse *Adım 3*'ten devam edilir.

Bu iki adımda yer alan μ değerinin ayarlanabilir bir parametre olduğu görülmektedir. Bu adımlarda yapılan çarpma işlemleri ile bu parametre iterasyon sayısından daha fazla sayıda değişmektedir. Bu parametre çok büyükse, bu algoritma en dik iniş yöntemi gibi

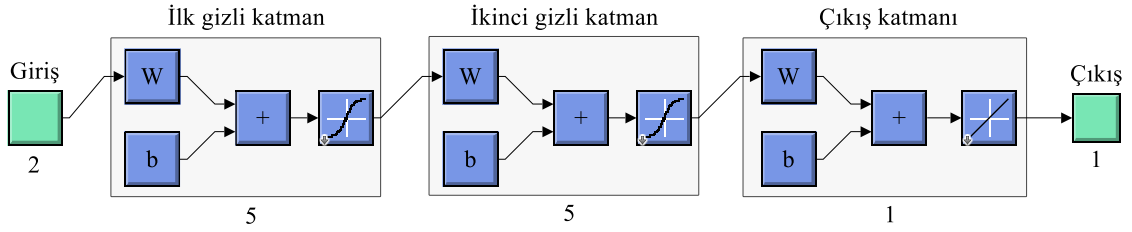
davranır, çok küçükse bir Gauss-Newton yöntemi gibi davranır (Hagan ve Menhaj, 1994).

Adım 6: W vektörü kullanılarak ileri hesaplama işlemi yapılır (bkz. s.82). Bu sayede H_E ve H_D vektörleri sonrasında bu vektörler ile güncel MSE_E ve MSE_D değerleri hesaplanır. Hesaplanan MSE_D değeri döngü boyunca elde edilen doğrulama performansları arasında en iyisi ise W vektörü eldeki en iyi ağırlık (W_b) vektörü olarak kaydedilir ($W_b = W$). Doğrulama Durumu (DD) değeri 0 olarak tazelenir ($DD = 0$). Değilse DD değeri 1 artırılır ($DD = DD + 1$). Ve Adım 2'ye geri dönülür böylece yeni iterasyon başlar. Ancak bu durumda DD değeri Çizelge 4.5'te yer alan $DD_{maksimum}$ değerine eşit olmuşsa Adım 7'den devam edilir.

Adım 7: En uygun ağırlık ve eşik değerleri olarak kaydedilen W_b vektörünün artık en iyi olduğu kabul edilir ve bütün döngü biter.

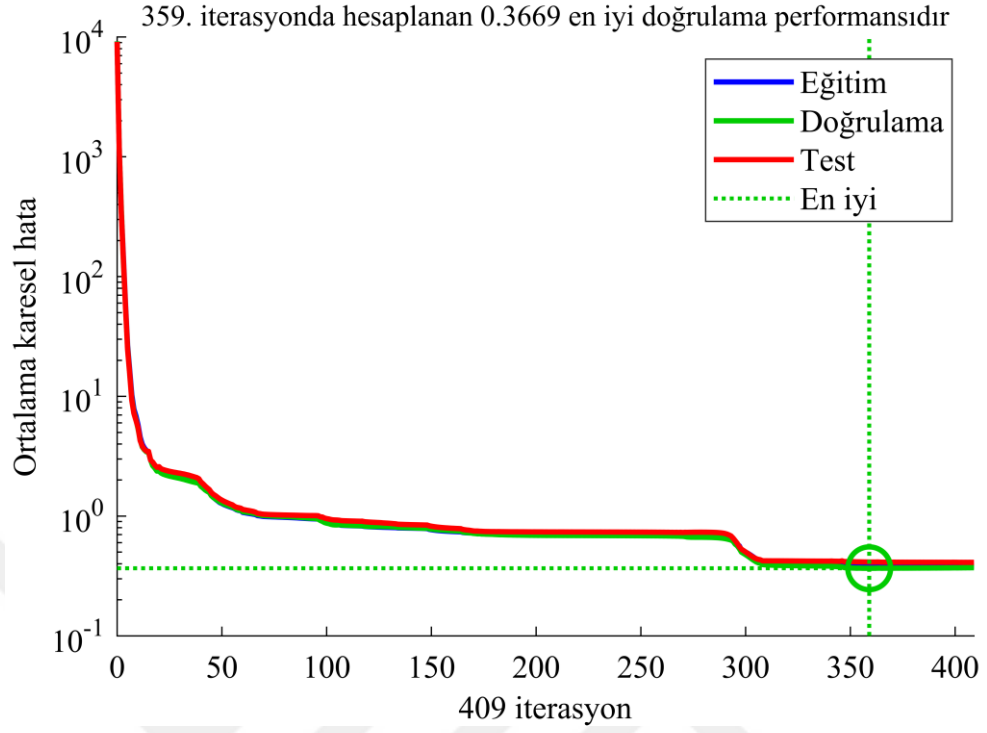
5. BULGULAR

YSA'nın en uygun ağırlık ve eşik değerleri eğitim esnasında MATLAB yazılımı tarafından hesaplanmış ve YSA yapısı oluşturulmuştur. Daha sonra bu yapı *Simulink bloğu oluşturma (gensim) fonksiyonu* ile Simulink bloğuna dönüştürülmüştür. BMD bloğu yerine yerleştirilen YSA bloğunun içeriği Şekil 5.1'de gösterilmiştir. Diğer BMD bloğu yerine yerleştirilen YSA bloğunun da içeriği aynı ancak *ağırlık ve eşik eğerleri* farklıdır.

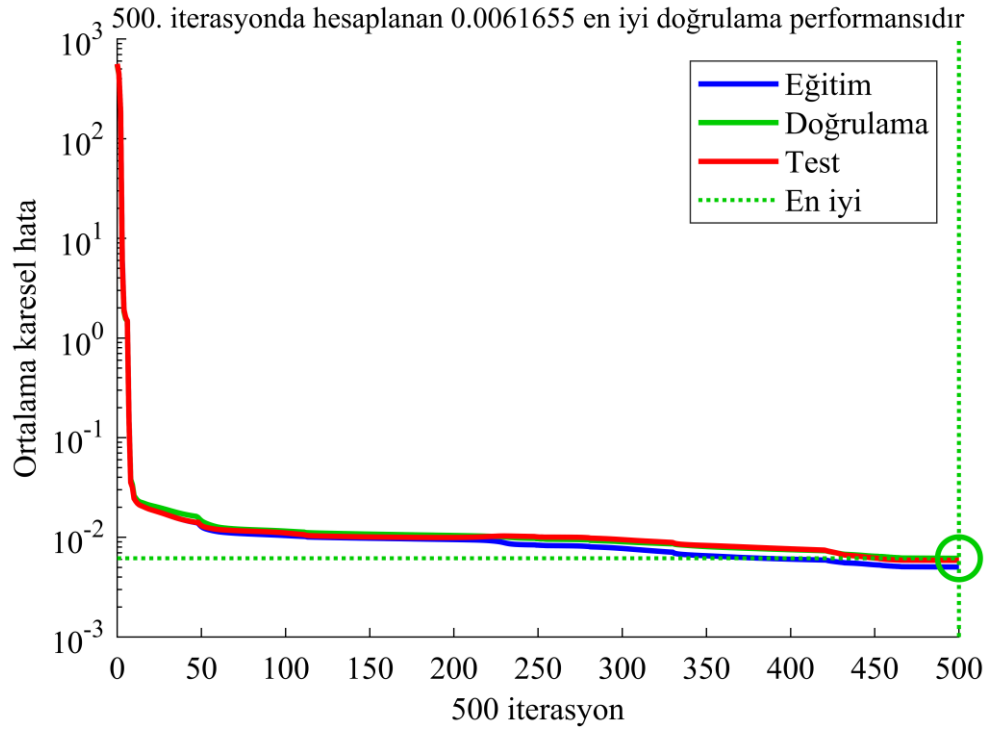


Şekil 5.1. YSA'ların yapısı ve içeriği

Şekil 5.2 ve Şekil 5.3 YSA'ların eğitimi sırasında elde edilen grafikleri göstermektedir. YSA_{ARABA} için yapılan eğitim sırasında MSE_D değeri 359 iterasyonda 0.3669'a düşmüştür. Eğitimin 409. iterasyonuna gelindiğinde bir hata düşüşü olmadığından son 50 iterasyon yapılmadan önceki ağırlık ve eşik değerleri kullanılmıştır. YSA_{SARKAÇ} için yapılan eğitim sırasında ise MSE_D değeri 500 iterasyonda 0.0078433'e düşmüştür. Son iterasyonlara gelindiğinde ciddi bir hata düşüşü olmadığından çok fazla ezber yapılmasının önüne geçebilmek için 500 tekrar yeterli görülmüş ve YSA_{SARKAÇ} için yapılan eğitim tamamlanmıştır.

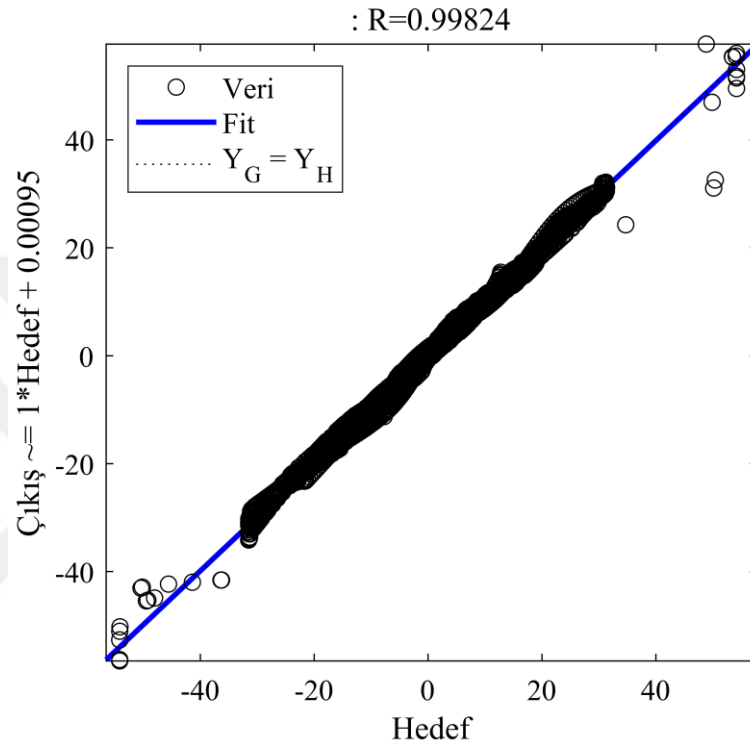


Şekil 5.2. Ortalama Karesel Hata (YSA_{ARABA})

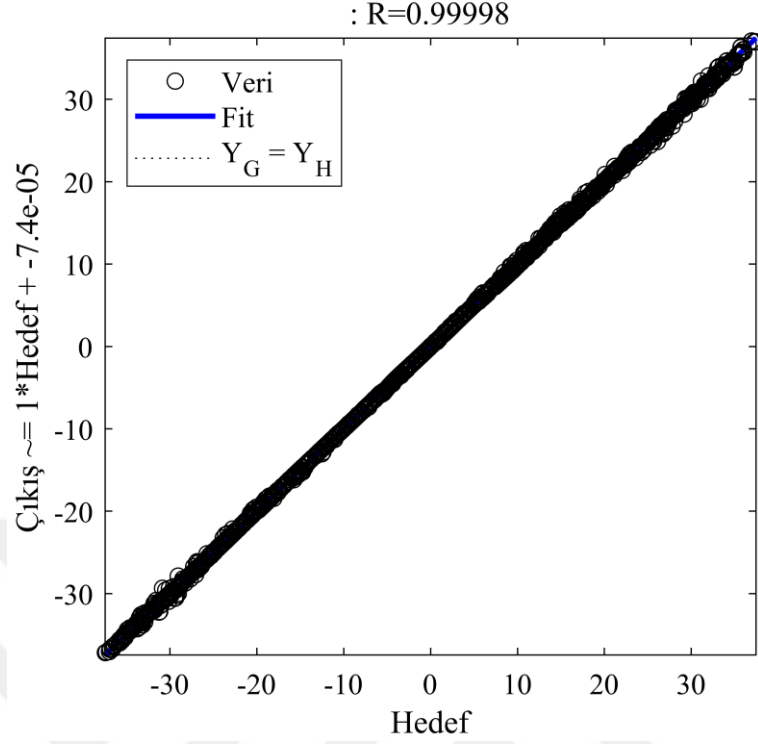


Şekil 5.3. Ortalama Karesel Hata (YSA_{SARKAÇ})

YSA'ların eğitimi sonunda elde edilen regresyon grafiği Şekil 5.4 ve Şekil 5.5'te gösterilmiştir. Burada, regresyon eğrisi üzerine toplanan noktalar iyi bir eğitimin bir göstergesidir. Yani YSA'ların BMD'lerin sonuçlarının neredeyse hatasız oranda elde edebildiği gözükmemektedir.

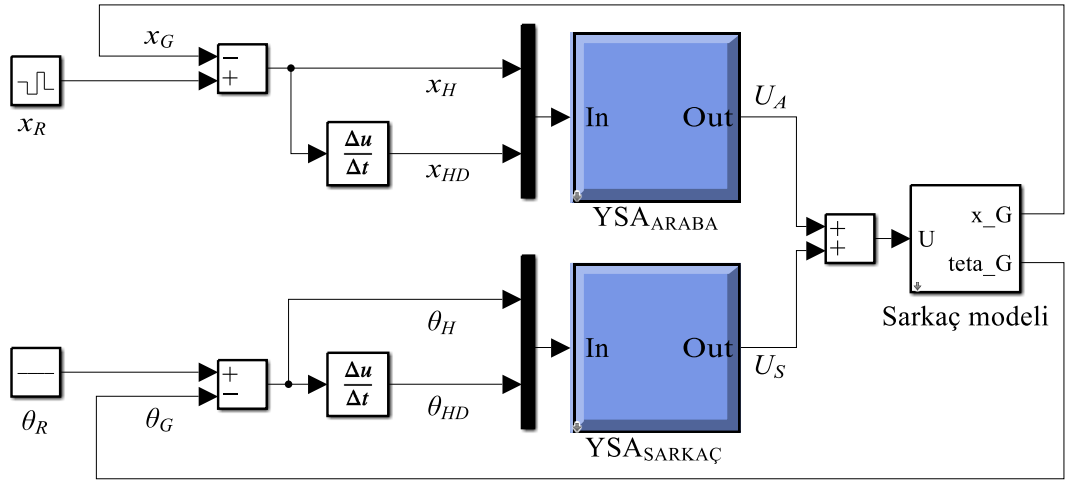


Şekil 5.4. Eğitimin regresyonu (YSA_{ARABA})



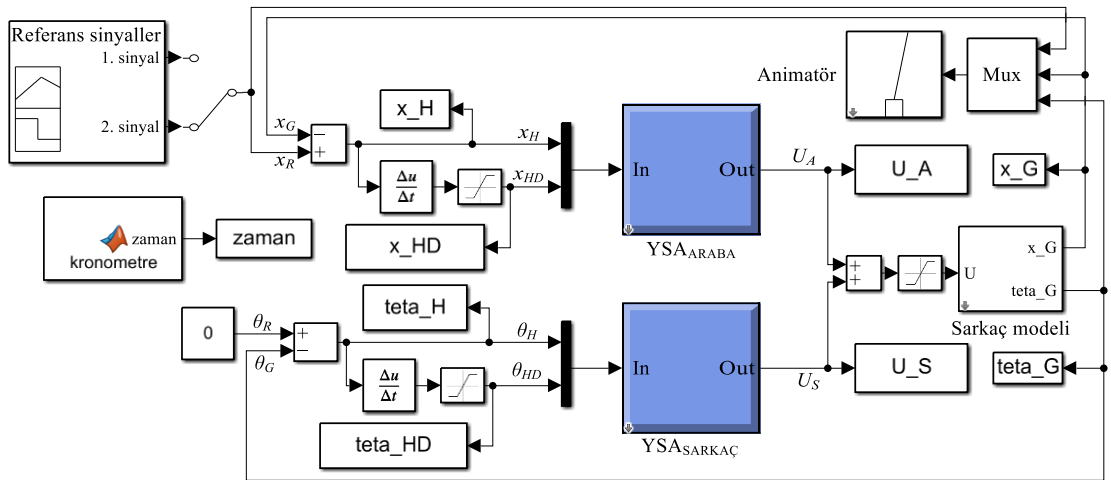
Şekil 5.5. Eğitimin regresyonu (YSA_{SARKAÇ})

Eğitim sonucu elde edilen iki YSA modeli Simulink eklentisinde tasarlanan benzetim sisteminde denenmeye hazır hale gelmiştir. İki BMD yerine bu iki BMD modellenerek elde edilen iki YSA modeli yerleştirilmiştir. Böylece kullanılan benzetim sisteminde BMD'lerin girişleri YSA'ların girişleri, BMD'lerin çıkışları YSA'ların çıkışları olmuş benzetim sistemindeki kalan Simulink blokları korunmuştur. Bu benzetim sisteminin ana hatları blok diyagramı olarak Şekil 5.6'da gösterilmiştir.



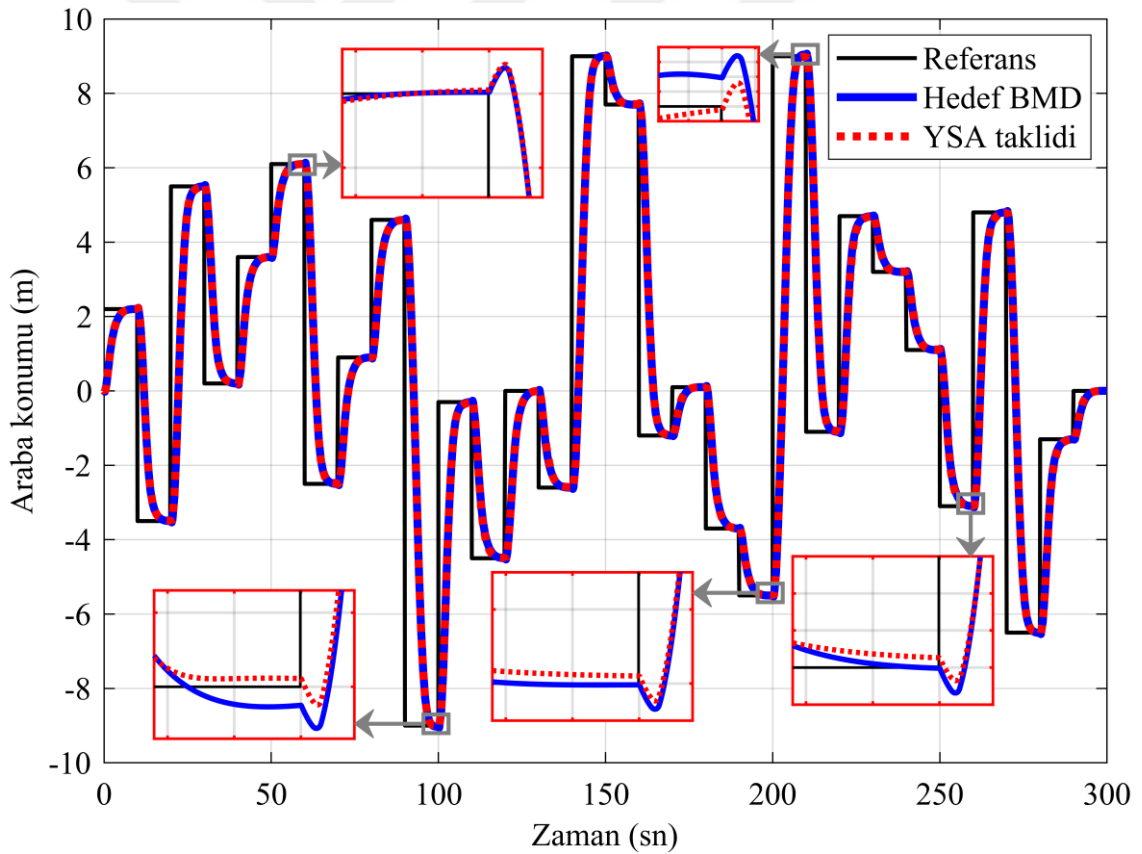
Şekil 5.6. BMD'ler yerine YSA'lar konulduğunda benzetim sisteminin ana hatları

Modellenen YSA'ların daha gerçekçi denenebilmesi için BMD'lerin çalıştırılıp veri setinin oluşturulduğu 1. *referans sinyali* yerine tamamen farklı 2. *referans sinyali* kullanılmıştır. İçerisinde birçok farklı bloğun olduğu benzetim sisteminin blok diyagramının tam hali Şekil 5.7'de gösterilmiştir. Simulink eklentisinde tasarlanan bu benzetim sisteminde 2. *referans sinyali* ile BMD'ler ve YSA'lar sırasıyla çalıştırılmıştır. Elde edilen sonuçlar aşağıdaki gibidir.



Şekil 5.7. BMD'ler yerine YSA'lar konulduğunda benzetim sisteminin tam hali

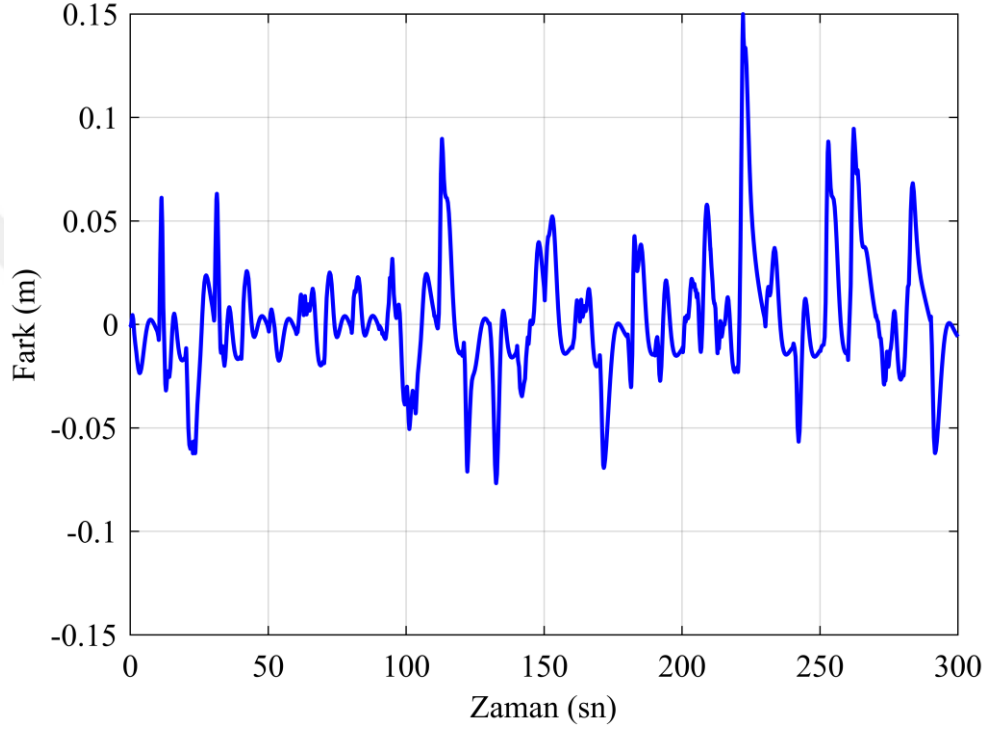
BMD'ler tarafından kontrol edilen arabanın konumu ile referans konum arasındaki farka dayalı olarak hesaplanan MSE değeri 10.1312'dir. Benzer şekilde, YSA'lar tarafından kontrol edilen arabanın konumu ile referans konum arasındaki farka dayalı olarak hesaplanan MSE değeri 10.1177'dir. Görüldüğü gibi her iki denetleyicinin MSE değerleri birbirine çok yakındır. Bu bize BMD'nin YSA ile yüksek oranda modellenebileceğini göstermektedir. Ayrıca, bu iki denetleyici tarafından yapılan kontrol esnasında elde edilen arabanın konumlarının arasındaki farka dayalı olarak hesaplanan MSE değeri 0.000806 yani çok küçük bir değer çıkmıştır. İki denetleyicinin sırasıyla arabayı kontrol etmesi esnasında arabanın konumu kaydedilmiş ve kontrol sonuçları Şekil 5.8'de gösterilmiştir. Ayrıca daha iyi karşılaştırma yapabilmek için grafikteki bazı yerler büyütülerek gösterilmiştir.



Şekil 5.8. Arabanın konumu

Şekil 5.8 arabanın konumunun grafiğidir ve iki denetleyici tarafından sırasıyla kontrol yapıldıktan sonra elde edilen sonuçlar beraber gösterilmiştir. Bu farkı daha net hale

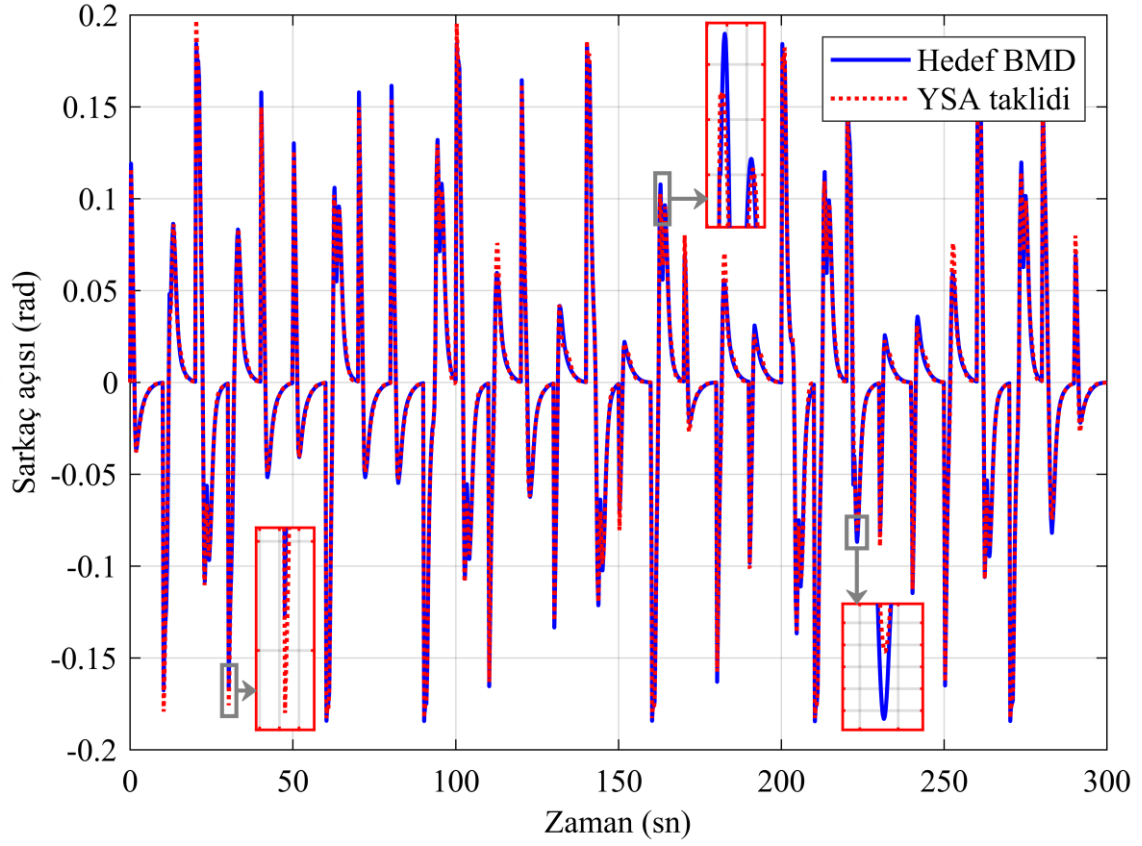
getirmek için, YSA'lar tarafından yapılan kontrol esnasında elde edilen arabanın konumundan BMD'ler tarafından yapılan kontrol esnasında elde edilen arabanın konumunun çıkarılmasıyla elde edilen fark grafiği Şekil 5.9'da gösterilmiştir. Grafik incelendiğinde iki denetleyicinin kontrol sonuçları arasındaki en büyük farkın 15 cm olduğu ve bu farkın kısa sürede kapandığı görülmektedir.



Şekil 5.9. Arabanın konumu (fark)

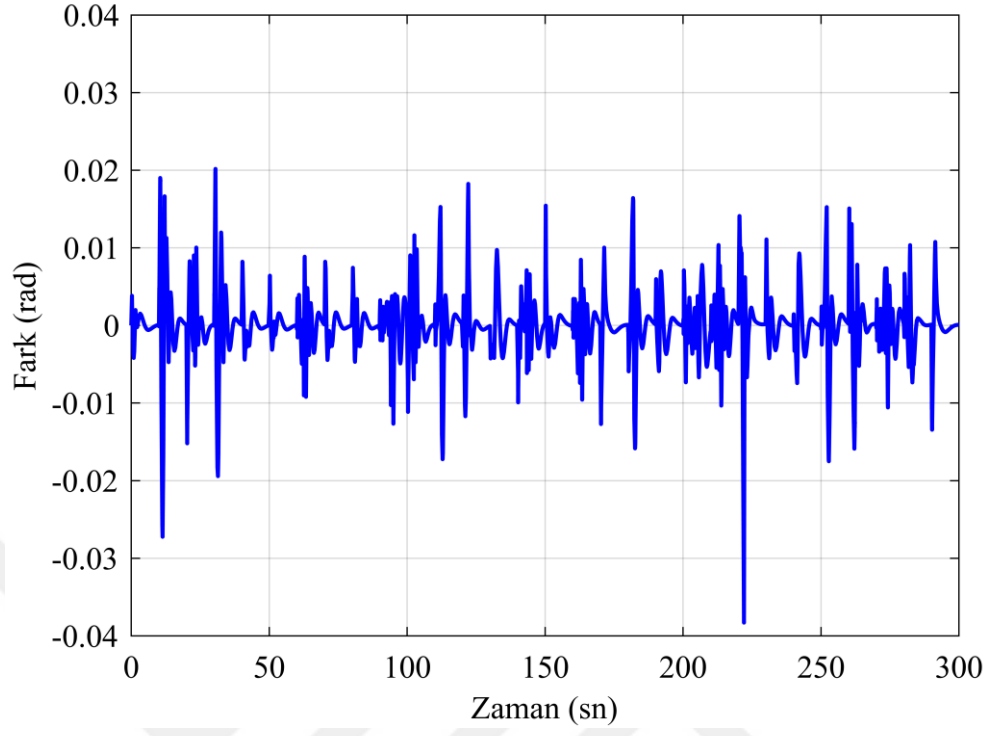
BMD'ler tarafından yapılan kontrol esnasında elde edilen sarkacın açısal konumu ile referans açısal konum arasındaki farka göre hesaplanan MSE değeri 0.0032098'dir. Benzer şekilde YSA'lar tarafından yapılan kontrol esnasında elde edilen sarkacın açısal konumu ile referans açısal konum arasındaki farka göre hesaplanan MSE değeri 0.0031916'dır. Görülebileceği gibi, her iki denetleyicinin sarkacın açısal konumuna göre hesaplanan MSE değerleri birbirine çok yakındır. Bu bize BMD'nin YSA ile yüksek oranda modellenebileceğini göstermektedir. Ayrıca, bu iki denetleyici tarafından yapılan kontrol esnasında elde edilen sarkacın açısal konumlarının arasındaki farka dayalı olarak hesaplanan MSE değeri 1.595×10^{-5} yani çok küçük bir değer çıkmıştır. İki denetleyicinin sırasıyla arabayı kontrol etmesi esnasında sarkacın açısal konumu

kaydedilmiş ve kontrol sonuçları Şekil 5.10'da gösterilmiştir. Ayrıca daha iyi karşılaştırma yapabilmek için grafikteki bazı yerler büyütülerek gösterilmiştir.



Şekil 5.10. Sarkacın açısal konumu

Şekil 5.10 sarkacın açısal konumunun grafiğidir ve iki denetleyici tarafından sırasıyla kontrol yapıldıktan sonra elde edilen sonuçlar beraber gösterilmiştir. Bu farkı daha net hale getirmek için, YSA'lar tarafından yapılan kontrol esnasında elde edilen sarkacın açısal konumundan BMD'ler tarafından yapılan kontrol esnasında elde edilen sarkacın açısal konumunun çıkarılmasıyla elde edilen fark grafiği Şekil 5.11'de gösterilmiştir. Grafik incelendiğinde iki denetleyicinin kontrol sonuçları arasındaki en büyük farkın 0.038 radyan olduğu ve bu farkın kısa sürede kapandığı görülmektedir.



Şekil 5.11. Sarkacın açısal konumu (fark)

6. SONUÇ

YSA birçok sistemin modelini elde etmede kullanılabilir, BMD ise bunlardan biridir. YSA'nın paralel çalışma özelliği vardır ve bu nedenle BMD'den çok daha hızlı çalışır. Ayrıca, hem yazılım hem de donanımdaki YSA tasarımı BMD tasarımından daha kolaydır.

BMD'nin giriş ve çıkış verilerinin kullanılmasıyla eğitimi yapıp elde edilen YSA ile BMD başarılı bir şekilde modellenmiştir. Bu model kontrol sürecini eğitimde yer alan veya almayan farklı referans değerleri altında gerçekleştirmeyi başarmıştır. Yapılan benzetim çalışmasında arabanın konumuna göre BMD'ler ile YSA'lar arasındaki farkın MSE değeri 0.000806'dır. Sarkacın açılma konumuna göre BMD'ler ile YSA'lar arasındaki farkın MSE değeri 1.595×10^{-5} olarak elde edilmiştir.

YSA tarafından kullanılan işlemler, BMD tarafından kullanılan işlemlerden daha kolaydır. Bu yüzden 300 benzetim saniyesi boyunca çalıştırılan BMD benzetimi 178.799 gerçek saniye ile gerçekleştirilirken, aynı benzetim YSA ile 2.059 gerçek saniyede gerçekleştirilmiştir. YSA ile tasarlanan sistem, daha az karmaşık matematiksel işlemler içerdiğinden, çeşitli kontrol uygulamalarında kullanımı ve tasarımı daha kolay olacaktır. Bu çalışma, YSA eğitim sürecinde daha fazla eğitim verisi, daha fazla hücre sayısı, daha fazla yinleme veya farklı eğitim algoritmaları kullanılarak geliştirilebilir.

Doğrusal olmayan sistemler için uzman tecrübeli BMD gibi veya öğrenme yetisine sahip olan YSA gibi yöntemler etkili çözümler olmuştur. BMD'nin etkili kontrol performansını YSA ile hızlı bir biçimde elde etmek, kontrol sistemlerinde birçok faydalar sağlayacaktır. Endüstrinin farklı kollarında üretim aşamasında kontrol sonuçlarının iyileşmesi, üretim verimliliğini ve ürün kalitesini artıracaktır. Bu ise enerji israfını azaltacak ve böylece ekonomik getiriye de katkı sağlayabilecektir. Bunun yanında, askeri ve gündelik ihtiyaçlarımızda güvenlik gerektiren uygulamalarda da etkili kontrol sonuçlarının birçok faydası olacaktır.

7. KAYNAKLAR

- Abd-elaziz, D. M., 2019. Conventional Fuzzy Logic Controller for Balancing Two-Wheel Inverted Pendulum. *Journal of Advanced Engineering Trends (JAET)*, 38 (2), 107-119. doi:10.21608/jaet.2020.73045
- Ahmed, A. H., Kotb, A. E. S. B. ve Ali, A. M., 2018. Comparison Between Fuzzy Logic and PI Control for the Speed of BLDC Motor. *International Journal of Power Electronics and Drive Systems (IJPEDS)*, 9 (3), 1116-1123. doi:10.11591/ijpeds.v9.i3.pp1116-1123
- Al-Mekhlafi, M. A. A., Wahid, H. ve Aziz, A. A., 2018. Adaptive Neuro-Fuzzy Control Approach for a Single Inverted Pendulum System. *International Journal of Electrical and Computer Engineering (IJECE)*, 8 (5), 3657-3665. doi:10.11591/ijece.v8i5.pp3657-3665
- Altaş, İ. H., 1999. Bulanık Mantık: Bulanıklılık Kavramı. *Enerji, Elektrik, Elektromekanik-3e*, (62), 80-85.
- Baran, A., Kılağız, Y. ve Keleş, A., 2010. Bulanık Uzman Sistem Yaklaşımı ile Yeşil Kart Başvuru Değerlendirme Sistemi. *Atatürk Üniversitesi İktisadi ve İdari Bilimler Dergisi*, 24 (1), 63-76.
- Bari, S., Hamdani, S. S. Z., Khan, H. U., Rehman, M. ur ve Khan, H., 2019. Artificial Neural Network Based Self-Tuned PID Controller for Flight Control of Quadcopter. 2019 International Conference on Engineering and Emerging Technologies (ICEET), (ss. 1-5). doi:10.1109/CEET1.2019.8711864
- Benkercha, R., Moulahoum, S. ve Colak, I., 2017. Modelling of Fuzzy Logic Controller of a Maximum Power Point Tracker Based on Artificial Neural Network. 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), (ss. 485-492). Cancun, Mexico: IEEE. doi:10.1109/ICMLA.2017.0-114
- Chawla, I. ve Singla, A., 2019. System Identification of an Inverted Pendulum Using Adaptive Neural Fuzzy Inference System. *Harmony Search and Nature Inspired Optimization Algorithms*, Editörler: N. Yadav, A. Yadav, J. C. Bansal, K. Deep ve J. H. Kim, (ss. 809-817). Singapore. doi:10.1007/978-981-13-0761-4_77
- Das, S., Roy, K. ve Saha, C. K., 2015. Fuzzy Membership Estimation Using ANN: A Case Study in CTG Analysis. *Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014*, Editörler: S. C. Satapathy, B. N. Biswal, S. K. Udgata ve J. K. Mandal, (ss. 221-228). Cham. doi:10.1007/978-3-319-11933-5_25
- Dernoncourt, F., 2013. Introduction to Fuzzy Logic. Massachusetts Institute of Technology.
- Diaz, N. L. ve Soriano, J. J., 2007. Study of Two Control Strategies Based in Fuzzy Logic and Artificial Neural Network Compared with an Optimal Control Strategy Applied to a Buck Converter. *NAFIPS 2007—2007 Annual Meeting of the North American Fuzzy Information Processing Society*, (ss. 313-318). San Diego, CA, USA: IEEE. doi:10.1109/NAFIPS.2007.383857
- Doğan, O., 2016. Uyarlamalı Sinirsel Bulanık Çıkarım Sisteminin (ANFIS) Talep Tahmini İçin Kullanımı ve Bir Uygulama. *Dokuz Eylül Üniversitesi İktisadi İdari Bilimler Fakültesi Dergisi*, 31 (1), 257-288. doi:10.24988/deuiibf.2016311513

- Efe, M. Ö., 2011. Neural Network Assisted Computationally Simple $PI^{\lambda}D^{\mu}$ Control of a Quadrotor UAV. *IEEE Transactions on Industrial Informatics*, 7 (2), 354-361. doi:10.1109/TII.2011.2123906
- Efe, M. Ö., 2012. Neural Network-Assisted $PI^{\lambda}D^{\mu}$ Control. *Fractional Dynamics and Control*, Editörler: D. Baleanu, J. A. T. Machado ve A. C. J. Luo, (ss. 19-31). New York, NY. doi:10.1007/978-1-4614-0457-6_2
- Ergezer, H., Dikmen, M. ve Özdemir, E., 2003. Yapay Sinir Ağları ve Tanıma Sistemleri. *PiVOLKA*, 2 (6), 14-17.
- Erkol, H. O., 2017. GA ve PSO ile Kontrol Parametrelerinin Optimizasyonu. *Karaelmas Fen ve Mühendislik Dergisi*, 7 (1), 179-185.
- Farah, L., Haddouche, A. ve Haddouche, A., 2020. Comparison Between Proposed Fuzzy Logic and ANFIS for MPPT Control for Photovoltaic System. *International Journal of Power Electronics and Drive Systems (IJPEDS)*, 11 (2), 1065-1073. doi:10.11591/ijpeds.v11.i2.pp1065-1073
- Hagan, M. T. ve Menhaj, M. B., 1994. Training Feedforward Networks with the Marquardt Algorithm. *IEEE Transactions on Neural Networks*, 5 (6), 989-993. doi:10.1109/72.329697
- Haykin, S. S., 2009. *Neural Networks and Learning Machines* (3rd ed.). New York: Prentice Hall.
- İmamoğlu, N., Eresen, A. ve Efe, M. Ö., 2009. Döner Ters Sarkaç Sisteminin Pekiştirmeli Öğrenme Algoritmaları ile Kontrolü. *Otomatik Kontrol Türk Milli Komitesi, Otomatik Kontrol Ulusal Toplantısı TOK'09*.
- Jezewski, M., Czabanski, R. ve Leski, J., 2017. Introduction to Fuzzy Sets. *Theory and Applications of Ordered Fuzzy Numbers*, Editörler: P. Prokopowicz, J. Czerniak, D. Mikołajewski, Ł. Apiecionek ve D. Ślęzak, (C. 356, ss. 3-22). Cham. doi:10.1007/978-3-319-59614-3_1
- Karunasena, G. M. K. B., Priyankara, H. D. N. S. ve Madhusanka, B. G. D. A., 2020. Artificial Neural Network vs PID Controller for Magnetic Levitation System. *International Journal of Innovative Science and Research Technology*, 5 (7), 505-511. doi:10.38124/IJISRT20JUL432
- Khalid, A., Zeb, K. ve Haider, A., 2019. Conventional PID, Adaptive PID, and Sliding Mode Controllers Design for Aircraft Pitch Control. *2019 International Conference on Engineering and Emerging Technologies (ICEET)*, (ss. 1-6). Lahore, Pakistan: IEEE. doi:10.1109/CEET1.2019.8711871
- Klir, G. J. ve Yuan, B., 1995. *Fuzzy Sets and Fuzzy Logic: Theory and Applications* (C. 4). Upper Saddle River, N.J: Prentice Hall PTR.
- Marie-Francoise, J.-N., Gualous, H. ve Berthon, A., 2006. Supercapacitor Thermal- and Electrical-Behaviour Modelling Using ANN. *IEE Proceedings—Electric Power Applications*, 153 (2), 255-262. doi:10.1049/ip-epa:20050096
- Marini, F., Bucci, R., Magrì, A. L. ve Magrì, A. D., 2008. Artificial Neural Networks in Chemometrics: History, Examples and Perspectives. *Microchemical Journal*, 88 (2), 178-185. doi:10.1016/j.microc.2007.11.008
- MatWorks Help Center, M., 2021a. Triangular membership function—MATLAB trimf. <https://www.mathworks.com/help/fuzzy/trimf.html> (15 Nisan 2021).
- MatWorks Help Center, M., 2021b. Trapezoidal membership function—MATLAB trapmf. <https://www.mathworks.com/help/fuzzy/trapmf.html> (13 Mayıs 2021).

- MatWorks Help Center, M., 2021c. Gaussian membership function—MATLAB gaussmf. <https://www.mathworks.com/help/fuzzy/gaussmf.html> (13 Mayıs 2021).
- MatWorks Help Center, M., 2021d. Sigmoidal membership function—MATLAB sigmf. <https://www.mathworks.com/help/fuzzy/sigmf.html> (13 Mayıs 2021).
- MatWorks Help Center, M., 2021e. Generalized bell-shaped membership function—MATLAB gbellmf. <https://www.mathworks.com/help/fuzzy/gbellmf.html> (13 Mayıs 2021).
- MatWorks Help Center, M., 2021f. Probabilistic OR - MATLAB probor. <https://www.mathworks.com/help/fuzzy/probor.html> (22 Ocak 2021).
- Michigan, U., Carnegie Mellon, U. ve Detroit Mercy, U., 2021a. Control Tutorials for MATLAB and Simulink—Inverted Pendulum: Simulink Modeling. <https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum§ion=SimulinkModeling> (13 Haziran 2021).
- Michigan, U., Carnegie Mellon, U. ve Detroit Mercy, U., 2021b. Control Tutorials for MATLAB and Simulink—Inverted Pendulum: System Modeling. <https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum§ion=SystemModeling> (13 Haziran 2021).
- Mien, T. L., 2017. Liquid Level Control of Coupled-Tank System Using Fuzzy-PID Controller. *International Journal of Engineering Research & Technology (IJERT)*, 6 (11), 459-464.
- Muliadi, J. ve Kusumoputro, B., 2018. Neural Network Control System of UAV Altitude Dynamics and Its Comparison with the PID Control System. *Journal of Advanced Transportation*, 2018, 1-18. doi:10.1155/2018/3823201
- Öztürk, K. ve Şahin, M. E., 2018. Yapay Sinir Ağları ve Yapay Zekâ'ya Genel Bir Bakış. *Takvim-i Vekayi*, 6 (2), 25-36.
- Park, W. J. ve Park, J.-B., 2018. History and Application of Artificial Neural Networks in Dentistry. *European Journal of Dentistry*, 12 (4), 594-601. doi:10.4103/ejd.ejd_325_18
- Pathak, V. K. ve Paliwal, S., 2017. Stabilization of Inverted Pendulum System Using Fuzzy Logic Controller. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 6 (4), 3071-3079.
- Rahimi, M. R., Hajighasemi, S. ve Sanaei, D., 2013. Designing and Simulation for Vertical Moving Control of UAV System Using PID, LQR and Fuzzy Logic. *International Journal of Electrical and Computer Engineering (IJECE)*, 3 (5), 651-659.
- Salim, H., Sultan, Kh. F. ve Jawad, R., 2019. Comparison Between PID and Artificial Neural Networks to Control of Boiler for Steam Power Plant. *Journal of Engineering Sciences*, 6 (1), E 10-E 15. doi:10.21272/jes.2019.6(1).e2
- Sayğan, S., 2014. Örgüt Biliminde Karmaşıklık Teorisi. *Ege Akademik Bakış*, 14 (3), 413-423.
- Schwartz, D. G., Klir, G. J., Lewis, H. W. ve Ezawa, Y., 1994. Applications of Fuzzy Sets and Approximate Reasoning. *Proceedings of the IEEE*, 82 (4), 482-498. doi:10.1109/5.282229
- Singh, N. H. ve Thongam, K., 2017. Fuzzy Logic-Genetic Algorithm-Neural Network for Mobile Robot Navigation: A Survey. *International Research Journal of Engineering and Technology (IRJET)*, 4 (8), 24-35.

- Slama, S., Errachdi, A. ve Benrejeb, M., 2019. Neural Adaptive PID and Neural Indirect Adaptive Control Switch Controller for Nonlinear MIMO Systems. *Mathematical Problems in Engineering*, 2019, 1-11. doi:10.1155/2019/7340392
- Teo, K. K., Wang, L. ve Lin, Z., 2001. Wavelet Packet Multi-Layer Perceptron for Chaotic Time Series Prediction: Effects of Weight Initialization. *Computational Science—ICCS 2001*, Editörler: V. N. Alexandrov, J. J. Dongarra, B. A. Juliano, R. S. Renner ve C. J. K. Tan, (C. 2074, ss. 310-317). Berlin, Heidelberg.
- Thaker, S. ve Nagori, V., 2018. Analysis of Fuzzification Process in Fuzzy Expert System. *Procedia Computer Science, International Conference on Computational Intelligence and Data Science*, 132, 1308-1316. doi:10.1016/j.procs.2018.05.047
- Tolon, M. ve Tosunoğlu, N. G., 2008. Tüketici Tatmini Verilerinin Analizi: Yapay Sinir Ağları ve Regresyon Analizi Karşılaştırması. *Gazi Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*, 10 (2), 247-259.
- Tóth-Laufer, E., Rudas, I. J. ve Takács, M., 2014. Operator Dependent Variations of the Mamdani-Type Inference System Model to Reduce the Computational Needs in Real-Time Evaluation. *International Journal of Fuzzy Systems*, 16 (1), 57-72.
- Yu, H. ve Wilamowski, B. M., 2011. Levenberg-Marquardt Training. *Intelligent Systems*, 12-1-12-16.
- Zaihidee, F. M., Mekhilef, S. ve Mubin, M., 2019. Robust Speed Control of PMSM Using Sliding Mode Control (SMC)—A Review. *Energies*, 12 (9), 1669. doi:10.3390/en12091669
- Zhao, J. ve Bose, B. K., 2002. Evaluation of Membership Functions for Fuzzy Logic Controlled Induction Motor Drive. *IEEE 2002 28th Annual Conference of the Industrial Electronics Society. IECON 02*, (C. 1, ss. 229-234). Sevilla, Spain: IEEE. doi:10.1109/IECON.2002.1187512