



T.C.
SELÇUK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DOĞRUSAL VE DOĞRUSAL OLMAYAN SİSTEMLERİN HİYERARŞİK BULANIK KONTROLÜ

Serhat SOYLU

YÜKSEK LİSANS TEZİ

Elektrik-Elektronik Mühendisliği
Anabilim Dalı

Temmuz-2013
KONYA
Her Hakkı Saklıdır

ÖZET

YÜKSEK LİSANS TEZİ

DOĞRUSAL VE DOĞRUSAL OLMAYAN SİSTEMLERİN HİYERARŞİK BULANIK KONTROLÜ

Serhat SOYLU

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Yrd. Doç. Dr. Ömer AYDOĞDU

2013, 77 Sayfa

Jüri
Yrd. Doç. Dr. Ömer AYDOĞDU
Doç.Dr. Ramazan AKKAYA
Doç.Dr. Mehmet ÇUNKAŞ

Bu çalışmada, Hiyerarşik Bulanık Kontrol yöntemleri detaylı olarak incelenmiş, doğrusal ve doğrusal olmayan sistemlerin hiyerarşik bulanık kontrolü simülasyon olarak gerçekleştirilmiştir. Bu yöntemle çok girişli ve kural sayısı fazla olan bulanık mantık denetleyicilerin, az girişli bulanık mantık denetleyiciler halinde hiyerarşik olarak tasarlanmasıyla kural sayılarının düşürülmesi sağlanmıştır.

Çalışmada ilk olarak kontrol edilecek olan doğrusal ve doğrusal olmayan sistemlerin simulink modelleri oluşturulmuş ve modellenen sistemlerin klasik bulanık mantık denetleyiciler ile kontrolü simülasyon olarak yapılmıştır. Daha sonra aynı sistemlerin hiyerarşik bulanık mantık denetleyiciler ile kontrolünün simülasyonu gerçekleştirilmiştir. Çalışmada klasik bulanık mantık denetleyicinin ve hiyerarşik bulanık mantık denetleyicinin ölçeklendirme katsayıları Parçacık Sürü Optimizasyonu (PSO) ve Gerçek kodlu Genetik Algoritma (GA) ile ayrı ayrı belirlenmiştir. Bulanık mantık denetleyiciler ile hiyerarşik bulanık mantık denetleyicilerin optimizasyon sonuçları hem PSO için hem de gerçek kodlu GA için çeşitli amaç ölçüt kriterleri kullanılarak birbirleri ile karşılaştırılmıştır. Çalışma sonuçlarından hiyerarşik bulanık mantık denetleyicinin doğrusal olmayan sistemlerin optimal kontrolünde daha başarılı sonuçlar verdiği, doğrusal sistemlerin optimal kontrolünde de klasik bulanık mantık denetleyicilere yakın sonuçlar verdiği ve iyi bir alternatif olabileceği görülmüştür.

Anahtar Kelimeler: Bulanık mantık denetleyiciler, Gerçek kodlu genetik algoritmalar, Hiyerarşik bulanık kontrol, Parçacık sürü optimizasyonu.

ABSTRACT

MS THESIS

HIERARCHICAL FUZZY CONTROL OF LINEAR AND NONLINEAR SYSTEMS

Serhat SOYLU

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF SELÇUK UNIVERSITY
THE DEGREE OF MASTER OF SCIENCE
IN ELECTRICAL & ELECTRONICS ENGINEERING

Advisor: Asst. Prof. Dr. Ömer AYDOĞDU

2013, 77 Pages

Jury
Asst. Prof. Dr. Ömer AYDOĞDU
Assoc. Prof. Dr. Ramazan AKKAYA
Assoc. Prof. Dr. Mehmet ÇUNKAŞ

In this study, hierarchical fuzzy control methods are examined in detail, the simulation of the control of linear and non-linear systems was performed using the method of hierarchical fuzzy control. With this method, the number of rules was decreased by hierarchically designing multiple-input and multiple-rules fuzzy logic controllers as less-input controllers.

In the study, first of all, we performed the simulation of the control of linear and nonlinear control systems with fuzzy logic controllers. Later, we performed the same simulation with hierarchical fuzzy logic controllers. Primarily in the study, the Simulink model of the systems to be controlled were created, then the scaling factors of fuzzy logic controller and hierarchical fuzzy logic controller were determined separately with PSO, and real-coded GA. In this study, optimization results of fuzzy logic controllers and hierarchical fuzzy logic controllers were compared with each other using PSO and real-coded GA on a variety of objective criteria. At the end of the study we found that hierarchical fuzzy logic controller gives more successful results in the optimization of non-linear systems and it has similar results in linear systems so it can be a good alternative for classic fuzzy controllers.

Keywords: Fuzzy logic controllers, Hierarchical fuzzy control, Particle Swarm Optimization, Real-coded genetic algorithms

ÖNSÖZ

Yüksek lisans çalışmalarım süresince bilgi ve tecrübeleriyle çalışmalarımı yönlendiren, her konuda destek olan, benden hiçbir zaman yardımlarını esirgemeyen danışman hocam Yrd.Doç.Dr. Ömer AYDOĞDU'ya, çalışmalarımda gece gündüz demeden bana destek veren, ihtiyaç duyduğum her anda yardımıma koşan değerli çalışma arkadaşım Öğr.Gör. Fazıl Saray'a ve bana manevi desteğiyle güç veren değerli eşime, babama ve anneme teşekkürlerimi sunarım.

Serhat SOYLU

KONYA-2013

İÇİNDEKİLER

ÖZET ..... **iv**

ABSTRACT ..... **v**

ÖNSÖZ ..... **vi**

İÇİNDEKİLER ..... **vii**

KISALTMALAR ..... **ix**

1. GİRİŞ VE KAYNAK ARAŞTIRMASI **1**

1.1. Giriş 1
1.2. Kaynak Araştırması 3
1.3. Tez Organizasyonu 5

2. MATERYAL VE YÖNTEM **6**

2.1. Bulanık Mantık 6
2.1.1. Bulanık mantık kümeleri 7
2.1.2. Üyelik fonksiyonları 10
2.2. Bulanık Mantık Denetleyiciler (BMD) 13
2.3. Hiyerarşik Bulanık Mantık Denetleyiciler (HBMD) 20
2.4. Parçacık Sürü Optimizasyonu (PSO) 29
2.5. Gerçek Kodlu Genetik Algoritmalar (GA) 32
2.5.1. Parametrelerin kodlanması ve amaç fonksiyonu 35
2.5.2. Başlangıç popülasyonu 35
2.5.3. Uygunluk değerinin hesaplanması 36
2.5.4. Değerlendirme ve doğal seçim 36
2.5.5. Eşleştirme 37
2.5.6. Çaprazlama 39
2.5.7. Mutasyon 40

3. DOĞRUSAL VE DOĞRUSAL OLMAYAN SİSTEMLERİN HİYERARŞİK BULANIK KONTROLÜ **42**

3.1. Doğrusal Sistemlerin Bulanık Mantık Kontrolü 42
3.2. Doğrusal Sistemlerin Hiyerarşik Bulanık Mantık Kontrolü 44
3.3. Doğrusal Olmayan Sistemlerin Bulanık Mantık Kontrolü 46
3.4. Doğrusal Olmayan Sistemlerin Hiyerarşik Bulanık Mantık Kontrolü 47

4. ARAŞTIRMA BULGULARI VE TARTIŞMA **49**

4.1. Doğrusal Sistemlerde PSO Optimizasyonlu Simülasyon Sonuçları 49
4.2. Doğrusal Sistemlerde GA Optimizasyonlu Simülasyon Sonuçları 53
4.3. Doğrusal Olmayan Sistemlerde PSO Optimizasyonlu Simülasyon Sonuçları 57
4.4. Doğrusal Olmayan Sistemlerde GA Optimizasyonlu Simülasyon Sonuçları 59
4.5. Tartışma 61

5. SONUÇLAR VE ÖNERİLER **64**

5.1. Sonuçlar 64
5.2. Öneriler 65

KAYNAKLAR **66**

EKLER **69**

ÖZGEÇMİŞ **77**

KISALTMALAR

ABS	Otomatik Frenleme Sistemi
BMD	Bulanık Mantık Denetleyici
CoA	Alanın Merkezi
DC	Doğru Akım
FIS	Bulanık Çıkarım Sistemi
GA	Genetik Algoritma
HBMD	Hiyerarşik Bulanık Mantık Denetleyici
IAE	Mutlak Hata Toplamı
ISE	Karesel Hata Toplamı
ISTSE	Karesel Zaman Ağırlıklı Karesel Hata Toplamı
ITAE	Zaman Ağırlıklı Mutlak Hata Toplamı
ITSE	Zaman Ağırlıklı Karesel Hata Toplamı
LIFE	Endüstriyel Bulanık Mühendislik Laboratuvarı
PD	Oransal Türevsel
PI	Oransal İntegral
PID	Oransal İntegral Türevsel
PSO	Parçacık Sürü Optimizasyonu
TSK	Takagi-Sugeno-Kang
YSA	Yapay Sinir Ağları

1. GİRİŞ VE KAYNAK ARAŞTIRMASI

1.1. Giriş

Günlük hayatta insanlar sıkça, kesin olarak bilinmeyen, sanki kesinmiş gibi düşünülen ancak kesinlik arz etmeyen birçok durumla karşılaşırlar. Hayatımızın neredeyse tüm evresi belirsizlikler, kesin olmayan düşünceler ve karar veremediğimiz birçok durum nedeniyle karmaşıklaşır. Bu karmaşıklık birçok sosyal, iktisadi ve teknik konularda insan düşüncesinin tam anlamı ile olgunlaşmamasından kaynaklanmaktadır. İnsan zihnindeki düşünce dünyasının çok renkli, değişik hatta karmaşık birçok motiften oluştuğu veya çok belirgin olmayan bir desene sahip olduğu söylenebilir. Bu karmaşık durumların sistematik bir şekilde sayısal öngörülerinin yapılması ise, ancak bir takım ön kabul ve varsayımlardan sonra mümkün olabilmektedir. İşte bu karmaşıklığı bulanıklık (fuzzy) olarak nitelendirmek mümkündür. İnsanın düşünce tarzında doğal olarak bulunan bulanıklık Bulanık Mantık Teorisi olarak ortaya konmuş (Zadeh, 1965) ve bulanık sistemlerin temelini oluşturmuştur. Verilere uygun karar verme süreci içeren bulanık sistemler, karar süreci gerektiren birçok denetim sisteminde önemli yer edinmiştir (Alkan, 2011).

Klasik kontrol yöntemlerinde kontrol edilecek sistemin matematiksel modeli elde edildikten sonra kullanılacak kontrolörün modeli belirlenir ve böylece sisteme uygun bir kontrolör tasarlanmış olur. Oysa gerçek dünyada çoğu sistemin doğrusal olmayan karakteristiklerinin ve karmaşık yapılarının dışında bilinmeyen parametrelerinin de olması modelleme anlamında zorluk oluşturmaktadır. Parametre belirleme aşamasında ise modelleme başlı başına zaman alıcı ve maliyetli bir işlemdir. Bu zorluğu yenmek için uyarlamalı kontrol yöntemleri olmasına karşın uyarlamalı kontrolün karmaşık yapısı uygulama alanında birtakım sıkıntılar yaratmıştır. İlk kez 1965 yılında ortaya çıkan bulanık mantık kavramı ise modele dayalı kontrol sistemlerine iyi bir alternatif oluşturmuştur. Bu sayede kontrol, matematiksel olarak modellenmesi zor olan karmaşık sistemler için daha basit bir şekilde yaklaşık akıl yürütme ile gerçekleştirilir (Sağlam, 2007).

Bulanık mantıkta kesin yargılar yoktur. Bir durum bütünüyle doğru veya bütünüyle yanlış olarak ifade edilmeyip, bir doğruluk derecesiyle ifade edilir. Bir başka deyişle, klasik mantık {0, 1} olmak üzere iki değerli olup herhangi bir belirsizliğe yer vermezken, bulanık mantıkta üyelik derecesi yani bir elemanın kümeye ait olma

derecesi [0, 1] aralığındaki her değeri alabilmektedir. İnsan düşünce yapısı da olayları var ya da yok gibi keskin ifadeler yerine az, daha az gibi yaklaşık ifadelerle değerlendirir. Dolayısıyla bulanık mantık bu yönüyle gerçek dünyayı ve insan düşünce yapısını iyi bir şekilde temsil etmektedir.

Bulanık mantıkta, bilgi "küçük", "büyük", "az", "çok" gibi dilsel ifadelerle verilir. Denetim uygulanacak sistemle ilgili kontrol kuralları, sistemle ilgili bilgisi olan uzmanlar yardımıyla bu dilsel ifadeler kullanılarak oluşturulur. Bunun yanında, bulanık mantık yaklaşımı, sadece bir çözüm bölgesinde değil, tüm işlem aralığında etkindir ve tam olarak bilinmeyen ya da eksik girilen bilgilere göre de işlem yapma yeteneğine sahiptir. Ayrıca kolay, hazlı ve ekonomik bir şekilde uygulamaya geçirilebilir (Elmas, 2003).

Tüm bu özellikleriyle bulanık mantık, hem klasik yöntemlerin yetersiz kaldığı doğrusal olmayan sistemlerin, hem de klasik yöntemlerin sıkça kullanıldığı doğrusal sistemlerin kontrolünde daha basit ve etkili bir kontrol imkânı sağlayarak ciddi bir alternatif oluşturur.

Bulanık mantık kullanılan sistemlerde en önemli noktalardan birisi de kural tabanında bulunan kural sayısını azaltmaktır. Klasik bulanık sistemlerde, giriş değişkeni sayısı arttıkça kural sayısı da üstel olarak artar. Genel olarak bulanık sistemlerde "n" adet giriş değişkeni ve her bir değişken için de "m" adet üyelik fonksiyonu kullanılmışsa kural tabanında m^n adet kural bulunması gerekir. Üstel olarak artan kural sayısı da hafızayı büyük ölçüde yükleyerek kontrolörü zorlar. Bu "boyut sorunu" ile baş etmek amacıyla hiyerarşik yapıda bulanık sistemler önerilmiştir. Bu yapı, düşük boyutlu bulanık sistemlerin hiyerarşik yapıda birleştirilmesi ile oluşur. Böylece değişken sayısı artsa bile kural sayısı üstel olarak değil, $(n-1)\cdot m^2$ şeklinde, doğrusal olarak artar. Bu sayede, hiyerarşik yapı yüksek ölçekli sistemlere rahatlıkla uygulanabilir.

Hiyerarşik bulanık sistemlerde, ara katmanlardaki çıkışlar fiziksel bir anlam ifade etmez ve çoğu zaman yapay olarak değerlendirilir. Dolayısıyla bu çıkışlar bir sonraki katmana giriş olarak kullanıldıklarında tasarımı güçleştirir ve bu durum katman sayısı arttıkça daha önemli bir sorun haline gelir. Bu nedenle ara seviyelere anlam katacak "ara değişken"ler tanımlanır. Böylece bu güçlüğün üstesinden gelinebilir (Sağlam, 2007).

Bulanık kontrolöre ait giriş ve çıkış değişkenleri, bulanık dilsel kümeleri, üyelik fonksiyonları, bulanık kuralları, bulanık çıkarım mekanizması ve durulama mekanizması kontrolörün yapısal parametreleridir. Genel olarak çevrimdışı ayarlanıp

kontrol boyunca değiştirilmeyen yapısal parametreler ile kontrolörün temel yapısı oluşturulmuş olur. Ayarlama parametreleri, giriş ve çıkış ölçekleme çarpanları ile üyelik fonksiyonu parametrelerini içerir. Ayarlama parametrelerinin değerleri ise performans artırımı amacıyla kontrol boyunca değiştirilerek çevrimiçi hesaplanabilir. Bununla beraber, sadece giriş/çıkış ölçekleme çarpanlarının ayarlanması ile de kontrolörün gereksinimleri karşılanabilir (Woo ve ark, 2000).

1.2. Kaynak Araştırması

Bulanık mantık kontrol, hiyerarşik bulanık mantık kontrol, PSO ve GA bugüne kadar birçok araştırmacının ilgisini çekmiştir. Literatüre her gün bu konularla ilgili yeni çalışmalar eklenmektedir. Kaynak araştırması olarak çeşitli makale, kitap ve tezlerden istifade edilmiş olup, kısaca içerik ve özetleri aşağıdaki gibidir.

Baykal ve Beyan (2004); kaynak kitapta bulanık mantık kavramları kısaca özetlenmiş ve uygulamaya yönelik basit örneklerle açıklanmaya çalışılmıştır. Sistem modelleme yaklaşımları ve bunlara bulanık mantık tarafından sağlanan açılımlar değerlendirilmiştir. Denetleyiciler, denetim sistemleri ve bulanık mantıkla denetleyici tasarımı hakkında bilgilere genişçe yer verilmiştir.

Yılmaz (2007); kaynak kitapta sırasıyla DC motor kontrolü, hidrolik servo sistem kontrolü, çamaşır makinelerinde devir ve yıkama süresi kontrolü, boru içinde top dengeleme kontrolü ve DC servo motor konum kontrolü gibi mühendislik uygulamalarının bulanık mantıkla simülasyonları gerçekleştirilmiştir. Bu uygulamalar Matlab'in benzetim amaçlı Simulink ve bulanık modelleme amaçlı FIS (Fuzzy Inference System) araçları kullanılarak hazırlanmıştır. Bulanık sistemlerin yapısı ve bununla ilgili kavramlar detaylı olarak anlatılmıştır.

Sağlam (2007); yüksek lisans tezinde bulanık mantığın temel kontrolörlerinden olan bulanık PID kontrolörüne hiyerarşik özelliğin kazandırılması ele alınmıştır. İki girişli ve üç girişli PID kontrolör yapılarının yanı sıra üç girişli PID kontrolörden faydalanarak oluşturulan hiyerarşik PID kontrolör, temel üç doğrusal sistem ve bir doğrusal olmayan sistem üzerinde uygulanmıştır. Ele alınan sistemler üzerinde hangi kontrolör yapısının etkili sonuçlar verdiğine karar verilmeye çalışılmıştır.

Gaing (2004); kaynak makalede generatör uyartım sisteminde reaktif güç akışının kontrolü için kullanılan otomatik gerilim regülatörünün PID denetleyicisinin parametrelerini, parçacık sürü optimizasyonu ile ayarlamıştır. Çalışmasında, aşım,

yükselme zamanı, yerleşme zamanı ve sürekli hal hatasını kullanarak geliştirdiği yeni bir hedef fonksiyonu denemiş ve sonuçlarını genetik algoritma ile elde ettiği sonuçlarla karşılaştırmıştır.

Ou ve Lin (2006); kaynak makalede parçacık sürü optimizasyonu ile PID parametrelerini ayarlamak için kendi hedef fonksiyonlarını geliştirmişler ve sonuçlarını genetik algoritma ile karşılaştırmışlardır.

Lin ve ark. (2008); kaynak makalede lineer olmayan dinamik filtreler için parametre kestirimine çözüm bulmada, parçacık sürü optimizasyonu kullanılmış ve genetik algoritmayla karşılaştırmalarına yer verilmiştir. Böylece, lineer olmayan dinamik filtrelerde parçacık sürü optimizasyonunun daha etkili olduğunu ortaya koymuşlardır.

Chang ve Shih (2010); kaynak makalede lineer olmayan sistemlerde, geliştirilen bir parçacık sürü optimizasyonu yaklaşımı ile PID parametrelerini optimize etmeye çalışmışlardır. Bu yaklaşımı ters sarkaç izleme kontrolünde kullanarak, etkili olduğunu göstermişlerdir.

Uçuk (2009); yüksek lisans tezinde klasik (PD, PID) denetleyiciler ve bulanık denetleyici, bir köprülü vincin minimum salınımlı konum kontrolünü sağlamak amacıyla tasarlanmıştır. Denetim sistemleri, klasik denetleyiciler ve bulanık denetleyiciler açıklanmış ve daha sonra sistemin hareket denklemeleri elde edilmiştir. Vincin bulanık mantık kontrolü için sistem Matlab Simulink yazılımı yardımıyla modellenmiştir. Doğrusal olmayan hareket denklemleri, doğrusallaştırıldıktan sonra klasik kontrol (PD ve PID) tasarımında kullanılması için transfer fonksiyonu bulunmuştur. Bu denetleyiciler sisteme uygulanarak konum-zaman ve salınım-zaman grafikleri elde edilmiş ve denetleyiciler karşılaştırılmıştır.

Teker (2008); yüksek lisans tezinde sürekli mıknatıslı senkron motorun hız kontrolü için bulanık mantık denetleyici kullanarak hata ve hatadaki değişimin kullanılması anlatılmaktadır. Bu tezde bulanık kontrol ile PID kontrol, simülasyon ve deneysel olarak karşılaştırılmaktadır. Bu şekilde her iki yöntemin de üstün ve üstün olmayan yönleri irdelenerek hangisinin ne şekilde seçilebileceği gösterilmektedir.

Ömür (2009); yüksek lisans tezinde iki seviyeli su tankının doğrusal olmayan kontrolünü PID ile, durum geri besleme yöntemi ile, bulanık ve öz-adaptif bulanık mantık (fuzzy logic, self-orginizing adaptive fuzzy logic) denetleyici ile ve ortogonal polinomlu YSA (yapay sinir ağı) kullanarak gerçekleştirmiştir. Bulanık mantık

kullanılarak gerçekleştirilen kontrolün sistemlerinin diğer kontrol yöntemlerine göre daha iyi sonuç verdiğini deneysel sonuçlar ile ortaya koymuştur.

Yüksel (2006); kaynak kitapta otomatik kontrole giriş, matematiksel model ve sistem dinamiği, dinamik sistemlerin modellenmesi ve analizi, sistemlerin geçici ve kalıcı durum davranışlarının analizi, temel denetim etkileri, endüstriyel denetim sistemleri ve çalışma yapısı gibi konulara yer verilmiştir. Dinamik sistemlerin modellenmesi ve çözümünde sağladığı kolaylık dolayısıyla Matlab ve Simulink hakkında da temel bilgiler verilmiştir.

1.3. Tez Organizasyonu

Bu çalışma 5 ana bölümden oluşmaktadır.

1. bölüm, **"Giriş ve Kaynak Araştırması"** bölümü olup öncelikle tezin konusu hakkında genel bir değerlendirme yapılmıştır. Yapılan literatür taraması kaynak araştırması adı altında verilmiş olup ayrıca tezin ana bölüm başlıkları hakkında da kısa bilgiler verilmiştir.

2. bölümde **"Materyal ve Yöntem"** bölümü olup, öncelikle bulanık mantık denetleyiciler için bulanık mantık kavramı ve bulanık kontrol kavramı hakkında geniş bilgilere yer verilmiştir. Daha sonra hiyerarşik bulanık mantık denetleyiciler hakkında bilgiler verilmiştir. Son olarak da optimizasyon yöntemlerinden parçacık sürü optimizasyonu yöntemi ve gerçek kodlu genetik algoritma yöntemi anlatılmıştır.

3. bölümde **"Doğrusal ve Doğrusal Olmayan Sistemlerin Bulanık Mantık Kontrolü"** simülasyon olarak gerçekleştirilmiştir. Öncelikle doğrusal ve doğrusal olmayan sistemlerin yapısı anlatılmış daha sonra da bulanık mantık denetleyici ve hiyerarşik bulanık mantık denetleyici ile sistemlerin kontrol simülasyonları gerçekleştirilmiştir.

4. bölüm, **"Araştırma Bulguları ve Tartışma"** bölümü, elde edilen bütün simülasyon sonuçlarını içermektedir. Simülasyon sonuçları doğrusal ve doğrusal olmayan sistemler için, bulanık mantık denetleyici ve hiyerarşik bulanık mantık denetleyicilerle elde edilen değerler ve PSO ile gerçek kodlu GA optimizasyon cevapları ayrı ayrı yorumlanmış ve birbirleri ile karşılaştırılmıştır.

5. bölüm **"Sonuçlar ve Öneriler"** bölümü olup, elde edilen sonuçların genel bir değerlendirilmesi özet olarak yapılmış ve "Hiyerarşik Bulanık Kontrol" konusuna nasıl katkıda bulunulabileceği anlatılmıştır.

2. MATERYAL VE YÖNTEM

2.1. Bulanık Mantık

Günlük hayatımızda karşılaştığımız durumları çoğu zaman kesin olarak ifade edemeyiz. Bu durumları zihnimizde yorumlarken, kesin ifadeler kadar ara ifadeler de kullanırız. Örneğin; hızla ilerleyen bir araba için, "hızlı" ifadesinin yanı sıra "çok hızlı", "çok çok hızlı", "oldukça hızlı" veya bir kış günü hava durumunu ifade etmek için sadece "soğuk" değil, "epey soğuk", "biraz soğuk", "çok soğuk" hatta "buz gibi soğuk" şeklinde ara ifadelere yöneliriz. Bu ifadeler kesin değil, belirsizdir. Kesin bir ifade olan "soğuk", "epey soğuk" veya "azıcık soğuk" şeklinde ifade edildiğinde belirsizleşir, ara değerlere dönüşmüş olur.

Bu duruma şu şekilde başka bir örnek verebiliriz. Günlük hayatta insanları yaşlarına göre genç ve yaşlı olarak sınıflandırırız. Bu sınıflandırmayı yaparken, 40 yaşının üzerindeki kişileri yaşlı olarak belirleyelim. Bu durumda; klasik mantığa göre, kesin olarak, 40 yaşının altındaki veya 40 yaşındaki kişiler genç; 40 yaşının üzerindeki kişiler ise yaşlıdır. Yani 39 yaşındaki bir kişi genç iken, 41 yaşındaki bir kişi yaşlıdır ki bu da aslında çok doğru değildir. Zaten insan mantığında bu durum böyle kesin bir çizgiyle ayrılamaz. 39 yaşındaki bir kişi çoğu zaman "genç sayılır" şeklinde, bazen de "biraz yaşlı" olarak ifade edilir (Yılmaz, 2007).

Bulanık mantık yapısı insan düşünce yapısına paralel bir yapıya sahiptir. Yukarıda bahsettiğimiz örneği bulanık mantıkla ele alırsak, 39 yaşındaki bir kişi %70 gençken, %30 yaşlı olabilir. 41 yaşındaki bir kişi de %20 gençken, %80 yaşlı olabilir. Görüldüğü gibi bulanık mantıkta %100 genç ya da %100 yaşlı gibi kesin ifadelerin yanında ara ifadeler de bulunabilmektedir.

Bulanık mantık yaklaşımı ilk kez 1965 yılında L.A. Zadeh tarafından ortaya konmuştur. Bu tarihten günümüze kadar da başta kontrol sistemleri olmak üzere birçok alanda bulanık mantık uygulamaları ortaya çıkmıştır. Bulanık mantık kontrol sistemlerinde uzman kişilerin deneyimlerine, bilgi ve tecrübelerine başvurularak EĞER "Durum" ise O HALDE "Sonuç" şeklinde bir kural tabanı oluşturulur. Oluşturulan bu kural tabanı kesin ifadelerden ziyade bulanık ifadeler içerir.

Örneğin; EĞER Giriş "Var" ise O HALDE Çıkış "Yok" klasik bir yaklaşımdır. Kullanılan iki ifade de, "Var" ve "Yok", kesin ifadelerdir. Oysa bulanık yaklaşımda kurallar;

- EĞER Giriş “Az” ise O HALDE Çıkış “Orta”
- EĞER Giriş “Orta” ise O HALDE Çıkış “Az”
- EĞER Giriş “Çok” ise O HALDE Çıkış “Çok Az”

şeklinde olmaktadır. Böylece “Az”, “Çok”, “Orta” … gibi kesin olmayan dilsel bilgilerin kullanılması, kontrol sisteminin bir insan gibi düşünebilmesini sağlamaktadır (Baykal ve Beyan, 2004). Bulanık mantık kontrolün insan düşünce yapısına paralel bir yapıda olması ve bu paralellikte sistemleri tanıyıp karar verebilmesi de onun sosyal bilimlerden mühendislik uygulamalarına kadar hemen her alanda geniş uygulama sahaları bulmasına neden olmuştur (Aslan, 2010).

Ortaya atılmasından bu yana bulanık mantık kavramının en önemli kontrol uygulamaları, 1974 yılında Mamdani tarafından gerçekleştirilen bir buhar makinesi kontrolü, 1980 yılında Danimarka’da Smith&Co. tarafından oluşturulan bir çimento fırını kontrolü ve 1987 yılında Japonya’da Hitachi şirketi tarafından tasarlanan Sendai metrosunun kontrolüdür (Alkan, 2011). Ayrıca 1989 yılında Canon, Fujitsu, Hitachi, Honda, IBM, Mazda, Mitsubishi, Nissan, Sharp, Sony, Toshiba, Toyota, Volkswagen gibi dünya devlerinin bulunduğu endüstriyel sponsorlarla oluşturulmuş, bir Japon hükümeti-endüstri ortak faaliyeti olan, yaklaşık 50 üyeli bir konsorsiyum olarak düzenlenen LIFE (Laboratory for Industrial Fuzzy Engineering – Endüstriyel Bulanık Mühendislik Laboratuvarı) adıyla bir araştırma laboratuarı kurulmuştur (Chaturvedi, 2008).

Artık günümüzde, asansörlerden klima sistemlerine, elektrikli ev aletlerinden otomobillere, video kameralardan insansız uçakların kontrolüne kadar hemen her alanda karşımıza bulanık mantık kontrol uygulamaları çıkmaktadır.

2.1.1. Bulanık mantık kümeleri

Klasik mantığa göre, bir eleman, bir kümeye ya tamamen aittir, yani o kümenin elemanıdır; ya da hiçbir şekilde o kümeye ait değildir, yani o kümenin elemanı değildir. Buna göre; o elemanın üyelik derecesi eğer 1 ise, eleman kümenin elemanıdır, kümeye aittir; eğer 0 ise, eleman kümenin elemanı değildir, yani kümeye ait değildir. Bu yüzden klasik mantıkta üyelik derecesi denklem (2.1)’deki gibi sadece 0 ve 1 değerini alır.

$$A=\{0,1\} \qquad (2.1)$$

Bulanık mantığa göre ise, bir eleman, bir kümeye tamamen ait olabilir, hiçbir şekilde ait olmayabilir, bunun yanında kümeye belirli bir derecede ait olabilir veya olmayabilir. Şöyle ki; bir elemanın kümeye ait olma derecesi 0 olabilir, elemanın kümeye ait olmadığını gösterir; 1 olabilir, elemanın kümeye tamamen ait olduğunu gösterir; 0-1 arasında ara değerler olabilir, elemanın kümeye kısmen ait olduğunu veya olmadığını gösterir. Bu nedenle bulanık mantıkta üyelik derecesi denklem (2.2)'deki gibi 0 ve 1 aralığında tüm değerleri alabilir.

$$B=[0,1] \tag{2.2}$$

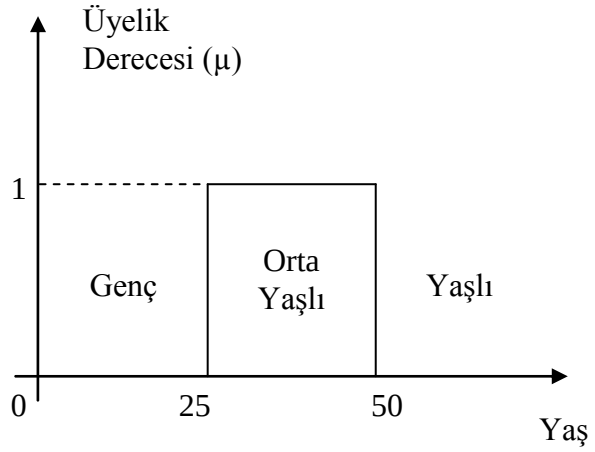
Klasik ve bulanık mantıkta üyelik derecelerinin gösterimi Şekil 2.1'de görülmektedir.



Şekil 2.1. Klasik ve bulanık küme farkı (Uçuk, 2009)

Bulanık mantık sistemlerinde bir x elemanının A kümesine ne kadar ait olduğu üyelik derecesi $\mu_A(x)$ ile ifade edilir ve [0,1] değer aralığına sahiptir. Bunu bir örnekle açıklayalım;

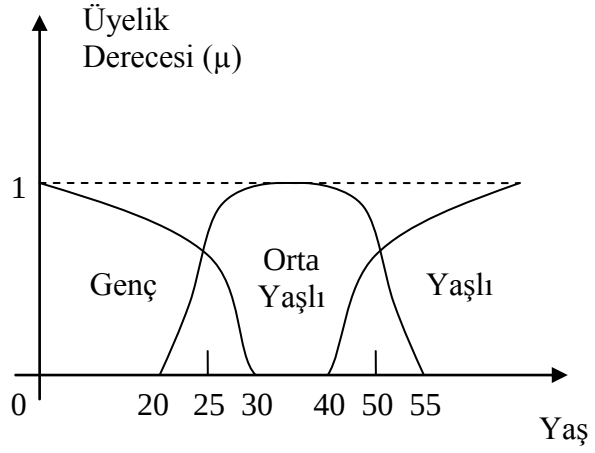
Önümüzde Şekil 2.2'deki gibi 0-25 yaş arası kişilerin genç, 25-50 yaş arası kişilerin orta yaşlı ve 50 yaşının üzerindeki kişilerin yaşlı olarak tanımlandığı kümeler olsun.



Şekil 2.2. Klasik kümede yaş örneği

Burada 24.5 yaşındaki bir kişi genç sayılırken, 25.5 yaşındaki bir kişi orta yaşlı sayılmaktadır. Klasik mantıkla oluşturduğumuz bu kümelerde hiçbir esneklik yoktur.

Bulanık mantıkta aynı örnek için oluşturulacak kümelerde ise Şekil 2.3'de de görülebileceği gibi geçişler klasik mantıkla oluşturduğumuz kümelerdeki kadar keskin değildir. Geçişlerde bir derecelendirme, haliyle bir esneklik söz konusudur. Bulanık mantık kümelemede 25 yaşındaki bir kişi orta yaşlı sayılamayacağı gibi genç de sayılamaz. Bu kişi belli bir oranda (belli bir derecede) genç, belli bir oranda da orta yaşlıdır.



Şekil 2.3. Bulanık kümede yaş örneği

Diğer bir bakış açısıyla; 20 yaşının altındaki kişilerin "orta yaşlı" kümesindeki üyelik dereceleri 0 iken, 20 yaşının hemen üzerindeki kişilerin "orta yaşlı" kümesindeki üyelik dereceleri 0'ın biraz üzerindedir. Benzer şekilde; yaşı 45 olan kişilerin "yaşlı"

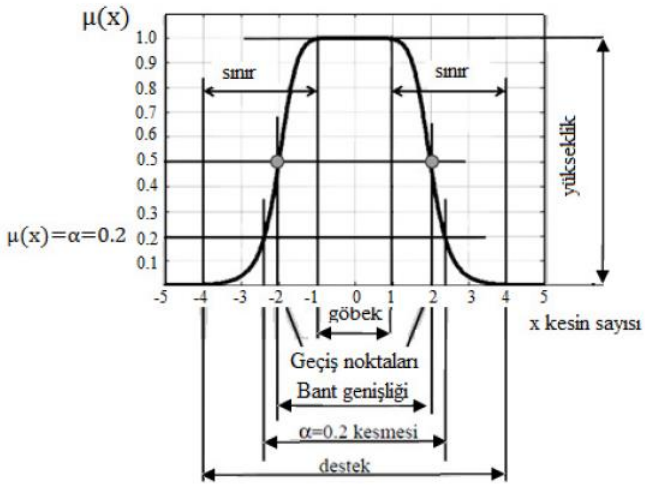
kümesindeki üyelik dereceleri 0'a yakınken (oldukça düşükken), yaşı 60 olan kişilerin üyelik dereceleri 1'e yakındır (oldukça yüksektir). Yine dikkat edersek; 25 yaşındaki bir kişi belli bir üyelik derecesiyle "genç" kümesinin elemanıyken aynı zamanda belli bir üyelik derecesiyle de "orta yaşlı" kümesinin elemanıdır.

2.1.2. Üyelik fonksiyonları

Üyelik fonksiyonları, bir elemanın ait olduğu kümelerdeki üyelik derecelerini gösteren eğrilerdir. Bu eğrilerin oluşturduğu grafikte X ekseni kümenin elemanlarını gösterirken, Y ekseni elemanın o kümedeki üyelik derecesini gösterir. Bulanık bir kümedeki her eleman belli bir derecede üyeliğe sahiptir.

Şekil 2.4'te örnek bir üyelik fonksiyonuna ait tüm unsurlar gösterilmiştir. Burada;

Göbek: Üyelik derecesi 1 olan kısımlardır.
Geçiş Noktaları: Üyelik derecesi 0.5 olan noktalardır.
Sınır: Üyeliği 0 ile 1 arasında olan kısımlardır.
Destek: Üyeliği 0'dan büyük olan noktalar kümesidir.
Alfa Kesmesi: Üyelik derecesi α'dan büyük ya da eşit olan noktalar kümesidir.
Etkin Alfa Kesmesi: Üyelik derecesi α'dan büyük olan noktalar kümesidir.
Bant Genişliği: α'nın 0.5 olduğu değerlerin arasında kalan kısımlardır.
Yükseklik: Kümenin maksimum üyelik derecesidir.



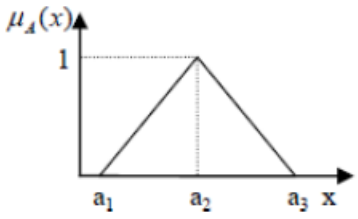
Şekil 2.4. Üyelik fonksiyonu ve kısımları (Altaş, 2008)

Üyelik fonksiyonları için standart, tek bir şekil yoktur. Kullanıcının ihtiyacına göre farklı şekillerde üyelik fonksiyonları belirlenebilir. Örneğin bir sistemde hassasiyet önemli ise üçgen üyelik fonksiyonu tercih edilirken, değişimin çok hızlı olması istenilen durumda yamuk üyelik fonksiyonu tercih edilebilir (Teker, 2008).

Üyelik fonksiyonu olarak kullanılan üçgen, yamuk, gaussian, çan eğrisi, s, z, pi, sigmoid, cauchy, trapez gibi birçok şekil vardır. Bunların arasında üçgen, yamuk ve gaussian en çok kullanılan üç üyelik fonksiyonudur (Passino ve Yurkovich, 1998).

Üçgen Üyelik Fonksiyonu

Bu fonksiyon Şekil 2.5'de görüldüğü gibi a_1, a_2 ve a_3 olmak üzere üç parametre ile tanımlanır.



Şekil 2.5. Üçgen üyelik fonksiyonu (Uçuk, 2009)

Fonksiyondaki a_2 parametresi göbeği oluştururken, a_1 ve a_3 parametreleri arasında kalan değerler de desteği oluşturur.

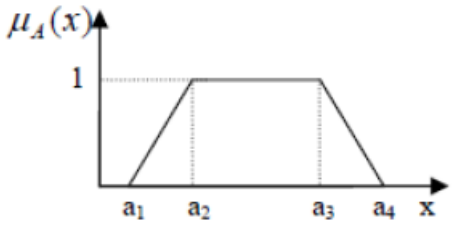
Üçgen üyelik fonksiyonu denklem (2.3)'de görüldüğü gibi;

$$\mu_A(x; a_1, a_2, a_3) = \begin{cases} a_1 \leq x \leq a_2 \Rightarrow (x - a_1)/(a_2 - a_1) \\ a_2 \leq x \leq a_3 \Rightarrow (a_3 - x)/(a_3 - a_2) \\ x > a_3 \; ve \; x < a_1 \Rightarrow 0 \end{cases} \quad (2.3)$$

şeklinde tanımlanır (Uçuk, 2009).

Yamuk Üyelik Fonksiyonu

Bu fonksiyon Şekil 2.6'da görüldüğü gibi a_1, a_2, a_3 ve a_4 olmak üzere dört parametre ile tanımlanır.



Şekil 2.6. Yamuk üyelik fonksiyonu (Uçuk, 2009)

Fonksiyondaki a_2 ve a_3 parametreleri arasında kalan değerler göbeği oluştururken, a_1 ve a_2 ile a_3 ve a_4 parametreleri arasında kalan değerler de sınırları oluşturur.

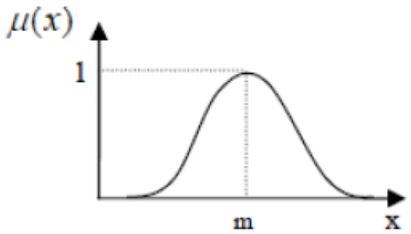
Yamuk üyelik fonksiyonu denklem (2.4)'de görüldüğü gibi;

$$\mu_A(x; a_1, a_2, a_3, a_4) = \begin{cases} a_1 \leq x \leq a_2 \Rightarrow (x - a_1)/(a_2 - a_1) \\ a_2 \leq x \leq a_3 \Rightarrow 1 \\ a_3 \leq x \leq a_4 \Rightarrow (a_4 - x)/(a_4 - a_3) \\ x > a_4 \; ve \; x < a_1 \Rightarrow 0 \end{cases} \tag{2.4}$$

şeklinde tanımlanır (Uçuk, 2009).

Gaussian Üyelik Fonksiyonu

Bu fonksiyon Şekil 2.7'de görüldüğü gibi m ve s parametreleri ile tanımlanır.



Şekil 2.7. Gaussian üyelik fonksiyonu (Uçuk, 2009)

Fonksiyondaki m parametresi göbeği oluştururken, s parametresi genişliği ifade eder. s değeri küçükken fonksiyon ince, s değeri büyükken fonksiyon yayvandır.

Gaussian üyelik fonksiyonu denklem (2.5)'de görüldüğü gibi;

$$\mu(x; m, s) = e^{\frac{-(x-m)^2}{2s^2}} \quad (2.5)$$

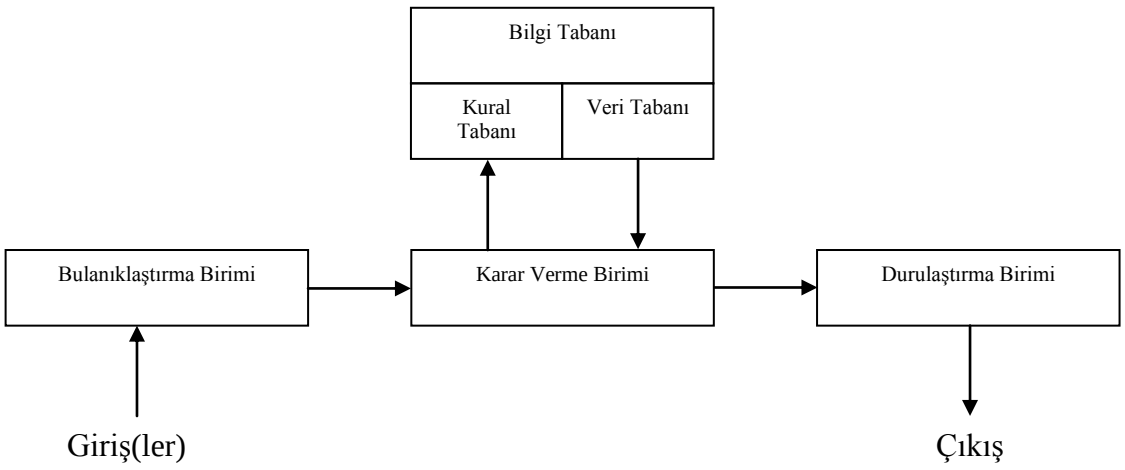
şeklinde tanımlanır (Mahmood, 2010).

2.2. Bulanık Mantık Denetleyiciler (BMD)

Klasik kontrolde, kontrol edilecek sistemin transfer fonksiyonunun, diğer bir deyişle, matematiksel modelinin elde edilmesi gerekir. Ayrıca, sağlıklı bir kontrol sağlamak için sistem parametrelerinin zamanla değişmemesi istenir. Ancak gerçek hayatta çoğu sistemin doğrusal olmayan karakteristikleri ve karmaşık yapıları haricinde bilinmeyen parametreleri de vardır ve bunlar modellemede zorluk oluşturmaktadır.

Bu nedenlerden dolayı, matematiksel modele ihtiyaç duymadan dilsel tanımlamalara dayanan ve lineer olmayan sistemlerde de iyi sonuçlar veren bulanık kontrol, son yıllarda birçok uygulamada karşımıza çıkmaktadır (Aydoğdu, 2006). İnsan bilgisini ve edinilen tecrübeleri, hatta insan düşünce yeteneğini kontrol sisteminin içine adapte edebilecek bir kontrol yöntemi olan bulanık kontrol, "Bulanık Mantık Denetleyiciler" ile makinelerin insanlar gibi karar vermelerini sağlayabilmektedir.

Bulanık mantık denetleyiciler (BMD) Şekil 2.8'de görüldüğü gibi dört temel arabirimden oluşur.



Şekil 2.8. Bulanık kontrolörün temel yapısı

Bunlar;

1- Bulanıklaştırma Birimi

2- Bilgi Tabanı

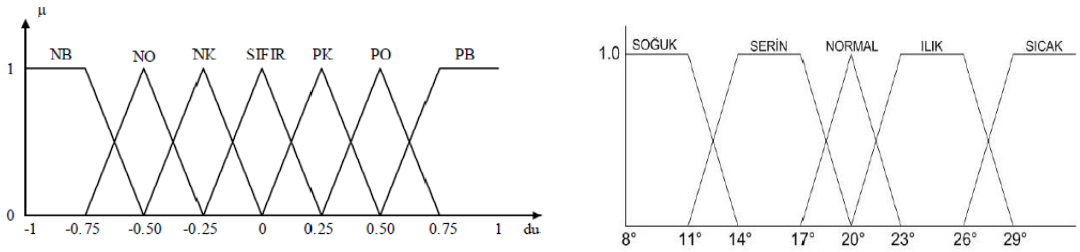
3- Karar Verme Birimi

4- Durulaştırma Birimi

Bulanıklaştırma Birimi

Bulanıklaştırma biriminin görevi girişten alınan kesin, keskin veya reel olarak ifade edebileceğimiz giriş değerlerini bulanık değerlere dönüştürmektir. Bu birimde; giriş değişkenlerinin gerçek zamanda ölçümlerinin alınması, giriş değişkenlerinin sabit bir sayıyla çarpılması ya da bölünmesi gibi işlemlerle ölçeklendirme yapılması, seçilen dilsel uzayın kesikli hale dönüştürülmesi, ölçümü yapılan reel giriş değişkenlerinin her birisinin o değişkene ait söylem uzayına göre dilsel değişkenlere dönüştürülmesi ve bunlara ait üyelik değerlerinin bulunması gibi işlemler gerçekleştirilmektedir (Ömür, 2009).

Bulanıklaştırma biriminde giriş değişkenlerinin bulanık değerlere dönüştürülmesi, uygun bir üyelik fonksiyonu kullanılarak gerçekleştirilir. Üyelik fonksiyonları denetlenecek sistemin durumuna göre uygulayıcı tarafından seçilir. Bulanıklaştırma biriminde her bir giriş/çıkış değişkeni için genel olarak 3, 5, 7 … gibi tek sayılardan oluşan adetlerde bulanık küme ve üyelik fonksiyonu bulunur. Bu giriş/çıkış değişkenleri tek bir çeşit üyelik fonksiyonundan oluşmanın yanı sıra, birkaç çeşit üyelik fonksiyonunun karışımından da oluşabilir. Şekil 2.9'da sadece üçgen üyelik fonksiyonlarından oluşan ve üçgen ile yamuk üyelik fonksiyonlarının karışımından oluşan iki farklı üyelik fonksiyonu gösterilmiştir. Şekilde görülebileceği gibi bunlardan birisi 7 üyelik fonksiyonundan, diğeri ise 5 üyelik fonksiyonundan meydana gelmiştir.



(a). İvme (b). Sıcaklık

Şekil 2.9. a) 7 elemanlı üçgen üyelik fonksiyonu (Uçuk, 2009) b) 5 elemanlı üçgen ve yamuk üyelik fonksiyonu (Ömür, 2009)

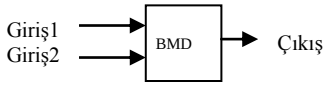
Bilgi Tabanı

Bilgi tabanı, üyelik fonksiyonlarıyla ilgili bilgileri barındıran bir "veri tabanı" ve değişik giriş değerleri için tespit edilmiş olan kontrol çıkış bilgilerini içeren "kural tabanı"ndan oluşur. Kural tabanında EĞER "Durum" ise O HALDE "Sonuç" şeklinde cümleler yer alır. Kurallar olarak adlandırılacak bu cümleler uzman bilgileri ile elde edilir. Kural tabanına; diğer kurallardan bağımsız olarak yeni kurallar eklenebilir ve eski kurallar da diğer kurallardan bağımsız olarak değiştirilebilir (Alkan, 2011). Örnek bir kural tabanında kurallar aşağıdaki şekilde yer alır;

1. EĞER Giriş "Küçük" ise O HALDE Çıkış "Küçük"
2. EĞER Giriş "Normal" ise O HALDE Çıkış "Normal"
3. EĞER Giriş "Büyük" ise O HALDE Çıkış "Büyük"

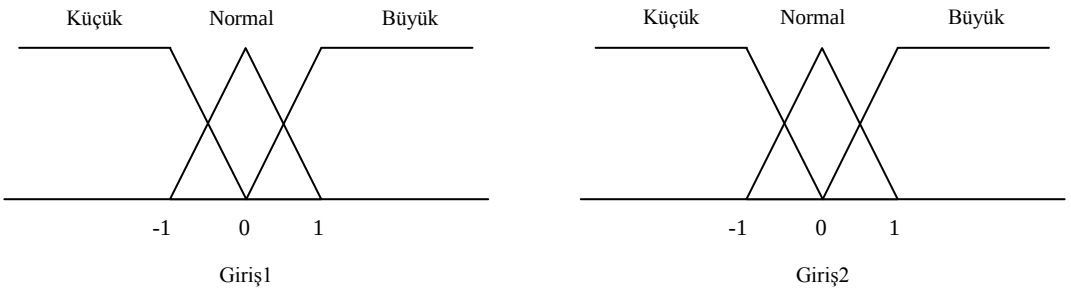
Yukarıda verilen kurallar tek girişli ve tek çıkışlı bir sistem içindir. Çok girişli sistemlerde, kurallar oluşturulurken durumlar birbirleri ile VE ya da VEYA işlemi ile birleştirilirler. Şekil 2.10'daki gibi iki girişli ve tek çıkışlı bir sistemin kural tabanında kurallar aşağıdaki şekilde yer alır;

1. EĞER Giriş1 "Küçük" VE Giriş2 "Küçük" ise O HALDE Çıkış "Küçük"
2. EĞER Giriş1 "Küçük" VE Giriş2 "Normal" ise O HALDE Çıkış "Küçük"
3. EĞER Giriş1 "Küçük" VE Giriş2 "Büyük" ise O HALDE Çıkış "Normal"
4. EĞER Giriş1 "Normal" VE Giriş2 "Küçük" ise O HALDE Çıkış "Küçük"
5. EĞER Giriş1 "Normal" VE Giriş2 "Normal" ise O HALDE Çıkış "Normal"
6. EĞER Giriş1 "Normal" VE Giriş2 "Büyük" ise O HALDE Çıkış "Büyük"
7. EĞER Giriş1 "Büyük" VE Giriş2 "Küçük" ise O HALDE Çıkış "Normal"
8. EĞER Giriş1 "Büyük" VE Giriş2 "Normal" ise O HALDE Çıkış "Büyük"
9. EĞER Giriş1 "Büyük" VE Giriş2 "Büyük" ise O HALDE Çıkış "Büyük"



Şekil 2.10. İki girişli ve tek çıkışlı BMD

Yukarıdaki kurallar Şekil 2.11'den görüldüğü gibi her bir giriş için üç bulanık kümenin (üç üyelik fonksiyonunun) kartezyen çarpımları ile elde edilmiştir.



Şekil 2.11. Giriş1 ve Giriş2 için bulanık kümeler (üyelik fonksiyonları)

Kurallar, özellikle sayıları arttığında, daha hızlı bir şekilde gözden geçirilebilmek amacıyla, Çizelge 2.1'deki gibi de gösterilebilir.

Çizelge 2.1. Kural tablosu

		Giriş1		
		Küçük	**Normal**	**Büyük**
Giriş2	**Küçük**	Küçük	Küçük	Normal
	Normal	Küçük	Normal	Büyük
	Büyük	Normal	Büyük	Büyük

Karar Verme Birimi

Karar verme birimi, girişten uygulanan ve bulanıklaştırma biriminden geçerek gelen bulanık verileri, bilgi tabanında bulunan kural tabanını ve veri tabanını kullanarak işleyen ve elde edilen bulanık verileri durulaştırma birimine durulaştırılmak üzere sunan birimdir.

İşleyişte öncelikle veri tabanı yardımıyla gelen verilerin hangi kümelere ne oranda ait oldukları belirlenir. Sonrasında Çizelge 2.1'deki gibi bir kural tabanından ilgili kurallar seçilir ve çıkarımlar yapılır. Daha sonra da yine veri tabanı yardımıyla kurallar sonucu elde edilen sonuçların hangi kümelere ne oranda ait olduğu belirlenerek bulanık veriler elde edilir ve durulaştırma birimine sunulur. Karar verme biriminde bu işleyiş için çeşitli yöntemler kullanılmaktadır. Bu yöntemlere örnek olarak Mamdani,

Larsen ve Takagi-Sugeno-Kang (TSK) verilebilir. Mamdani yöntemi bu yöntemler arasında en yaygın kullanılan yöntemdir.

Mamdani Yöntemi

Min-Max (Minimum-Maksimum) çıkarım olarak da bilinen Mamdani yönteminde VE (AND) ile bağlanmış değişkenlerin minimumları (en küçükleri) alınırken, VEYA (OR) ile bağlanmış değişkenlerin ise maksimumları (en büyükleri) alınır (Alkan, 2011).

Bu yöntemin kural yapısı;

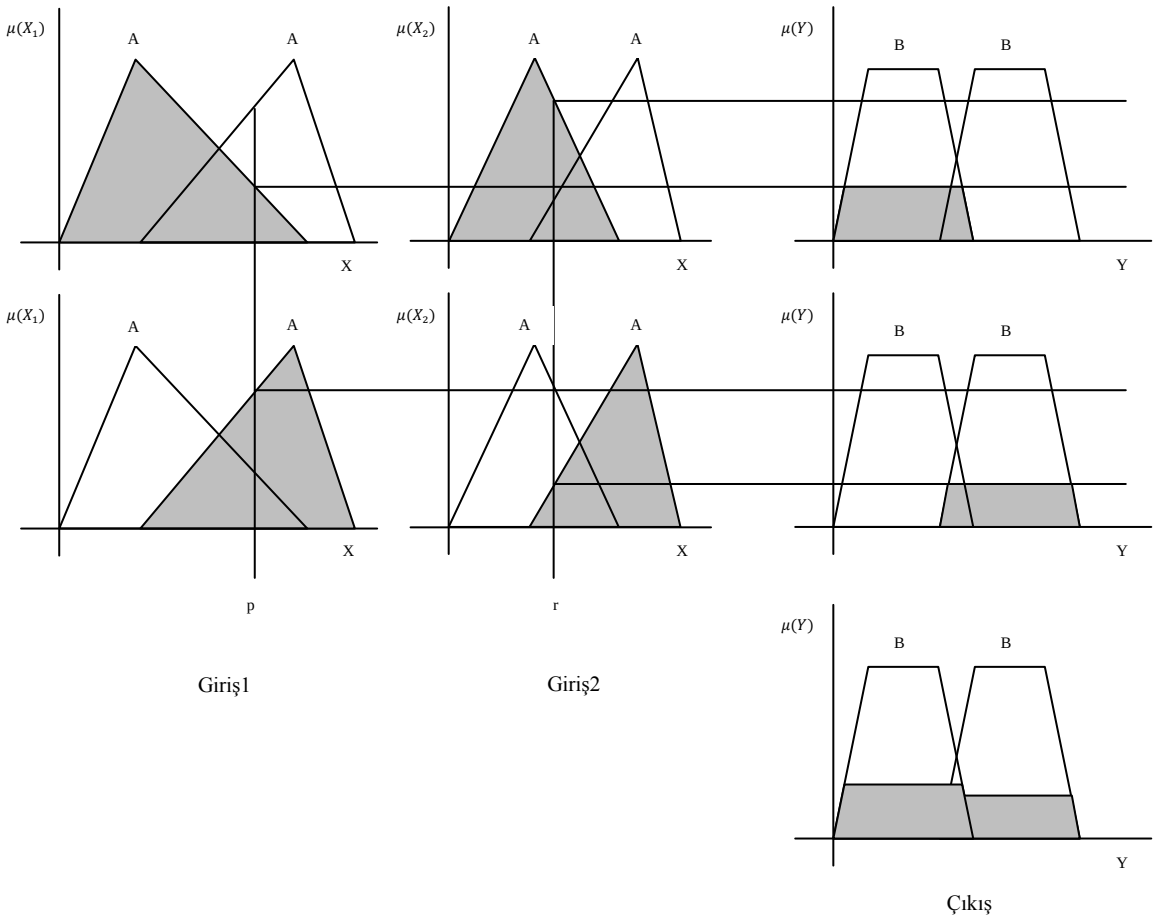
EĞER $X_1=A_{11}$ VE $X_2=A_{21}$ ise O HALDE $Y=B_1$
EĞER $X_1=A_{12}$ VE $X_2=A_{22}$ ise O HALDE $Y=B_2$

şeklindedir. Burada X_1 ve X_2 giriş değişkenlerini, Y ise çıkış değişkenini ifade eder. Ayrıca A_{11}, A_{12}, A_{21}, A_{22} giriş ve B_1, B_2 de çıkış üyelik fonksiyonlarıdır.

Verilen kurallar ışığında $X_1=p$ ve $X_2=r$ için Mamdani yöntemine göre çıkış ifadesi denklem (2.6)'daki gibi olacaktır.

$$\mu_{B^k}(y) = \max_k \left\{ min \left[\mu_{A_1^k}(p), \mu_{A_2^k}(r) \right] \right\} \; k = 1, \dots, giriş\ sayısı \qquad (2.6)$$

İki giriş ve tek çıkıştan oluşan, yukarıda tanımlanan iki kuralı içeren örnek bir sistemin karar verme birimindeki işleyiş, mamdani yöntemi ile grafiksel olarak Şekil 2.12'deki gibi gösterilebilir.



Şekil 2.12. X_1=p ve X_2=r için Mamdani yöntemi ile karar verme birimi işleyişi

Şekil 2.12'deki gösterimden de izlenebileceği üzere X_1=p ve X_2=r için, p değerinin hem A_{11}, hem de A_{12}'ye; r değerinin de hem A_{21}, hem de A_{22}'ye üye olduğu görülür ve A_{11}, A_{12}, A_{21} ve A_{22}'nin bulunduğu tüm kuralların denenmesi gerekir. Bu durumda örnek sistemimizde bulunan iki kural da denenecektir. Kurallarda değişkenler VE ile bağlandığı için ilgili üyelik derecelerinin minimum değerleri alınacaktır.

Birinci kuralda p'nin A_{11}'e olan üyeliği, r'nin A_{21}'e olan üyeliğinden daha küçük olduğu için, B_1'in; üyelik değeri, p'nin A_{11}'e olan üyelik değerine eşit veya daha küçük olan elemanları taranır. İkinci kuralda da r'nin A_{22}'ye olan üyeliği, p'nin A_{12}'ye olan üyeliğinden daha küçük olduğu için, B_2'nin; üyelik değeri, r'nin A_{22}'ye olan üyelik değerine eşit veya daha küçük olan elemanları taranır. Daha sonra elde edilen iki çıkış, bulanık kümelerde birleşim kuralına göre birleştirilerek işleyiş tamamlanır.

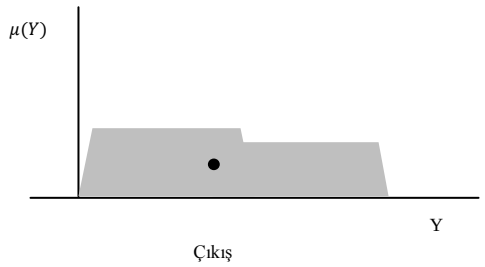
Durulaştırma Birimi

Bulanık mantık denetleyicinin en son aşaması olan durulaştırma birimi sonuç değerlerinin bulanık bir küme olarak elde edildiği Mamdani ve Larsen yöntemi gibi yöntemlerde kullanılır. Sonuç değerlerinin kesin olarak elde edildiği Takagi-Sugeno-Kang yöntemi gibi yöntemlerde bu birime ihtiyaç yoktur (Uçuk, 2009). Çünkü durulaştırma biriminde, bulanık verilerin kesin değerlere dönüştürülmesi işlemi gerçekleştirilir.

Durulaştırma işlemi için ağırlık merkezi yöntemi, ağırlıklı ortalama yöntemi, toplamların merkezi yöntemi gibi birçok durulaştırma yöntemi vardır. En çok tercih edilen durulaştırma yöntemi Ağırlık Merkezi Yöntemi'dir (Alkan, 2011).

Ağırlık Merkezi Yöntemi

Alanların merkezi yöntemi veya alanın merkezi yöntemi (Center of Area - CoA) olarak da bilinir. Bu yöntemde Şekil 2.13'de gösterilen karar verme biriminde elde edilen çıkış kümelerinin ağırlık merkezi denklem (2.7)'deki bağıntı kullanılarak hesaplanır.



Şekil 2.13. Ağırlık merkezi yöntemi ile elde edilen çıkış

Hesap sonucunda elde edilecek değer, sistemin gerçek çıkış değerini verir.

$$Çıkış = \frac{\int \mu(y) \cdot y}{\int \mu(y)} \tag{2.7}$$

Denklem (2.7)'de y değerleri karar verme biriminde elde edilen çıkıştaki bulanık kümenin elemanlarını, μ(y) değerleri ise bu elemanların üyelik derecelerini belirtir.

2.3. Hiyerarşik Bulanık Mantık Denetleyiciler (HBMD)

Bulanık mantık denetleyicilerde, kural tabanı olarak isimlendirdiğimiz birimde, uzman bilgileri ile elde ettiğimiz EĞER "Durum" ise O HALDE "Sonuç" şeklinde kurallar yer almakta idi. Bu kuralların sayısı ve doğruluğu denetlenecek sistemin performansını etkileyen en önemli faktörlerden birisidir. Diğer bir yandan, denetleyicilerde, kural tabanını olabildiğince basit yapıda tutmak hem hesap hem de kontrol kolaylığı açısından oldukça önemlidir.

Bulanık mantık denetleyicilerde, giriş değişkeni sayısı arttıkça, kural sayısı da üstel olarak artar. Özellikle giriş değişkeninin 2'den fazla olduğu durumlarda oldukça artan kural sayısı hafızayı aşırı derecede yükleyerek denetleyiciyi zorlar. Bu yüzden denetleyicinin kural tabanındaki gereksiz kuralları elimine etmek veya kural tablosunun boyutunu düşürmek gibi yaklaşımlarda bulunulur. Kural sayısını veya kural tablosunun boyutunu düşürmenin bir yöntemi de hiyerarşik yapıdır (Sağlam, 2007).

Klasik yapıdaki bir bulanık mantık denetleyicide n tane giriş değişkeni ve her bir giriş değişkeni için de m tane üyelik fonksiyonu tanımlı ise, kural tabanında gerekli olan kural sayısı m^n ile ifade edilir. Bu durumda, örneğin, 10 giriş değişkeni ve her giriş değişkeni için 7 üyelik fonksiyonu olan bir sistem için 7^{10}=282475249 adet kural ortaya çıkar. Bu uygulanamayacak kadar büyük bir rakamdır (Wang, 1999).

Giriş değişken sayısının artışı ile kural sayısının üstel olarak artışı sonucu ortaya çıkan bu durum "Boyutsallığın Laneti" (Curse of Dimensionality) olarak da adlandırılır (Wang, 1998).

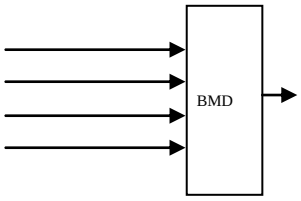
Hiyerarşik bulanık mantık denetleyiciler (HBMD), düşük boyutlu bulanık mantık denetleyicilerin hiyerarşik formda birleştirilmesi ile oluşur. Burada avantaj, giriş değişken sayısının artışıyla kural sayısının üstel olarak değil, doğrusal olarak artmasıdır. Hiyerarşik bulanık mantık denetleyicilerde n tane giriş değişkeni ve her bir giriş değişkeni için de m tane üyelik fonksiyonu tanımlı ise kural tabanlarında gerekli olan toplam kural sayısı $(n-1)\cdot m^2$'dir (Lee ve ark, 2003). Yukarıda bahsettiğimiz 10 giriş değişkenine ve her giriş değişkeni için 7 üyelik fonksiyonuna sahip sistem için HBMD ile oluşturulacak bir sistem için kural sayısı $(10-1)\cdot 7^2$=441 olacaktır. Bu durumda kural sayısı %99 azalmış olacaktır.

Yine; 4 giriş değişkeni ve her giriş değişkeni için 5 üyelik fonksiyonu olan bir sistem için Şekil 2.14'deki gibi klasik yapıdaki bir BMD'de ihtiyaç duyulacak kural sayısı 5^4=625'tir. Oysa Şekil 2.15'de verilen HBMD'lerde $(4-1)\cdot 5^2$=75 adet kural ortaya

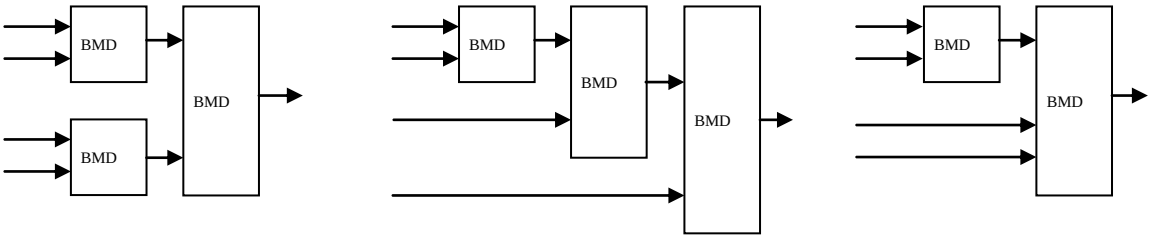
çıkar. Görüleceği gibi burada da hiyerarşik yapı sayesinde kural sayısı %88 azaltılmıştır.

Yukarıda verilen iki örneği dikkate alırsak, 4 giriş değişkeni ve 5 üyelik fonksiyonuna sahip sistemde kural sayısı %88 azalırken, 10 giriş değişkeni ve 7 üyelik fonksiyonuna sahip sistemde kural sayısı %99 azalmaktadır. Bu da gösteriyor ki; giriş değişkeni ve üyelik fonksiyonu sayısı daha çok olan denetleyicilerde HBMD kullanıldığında kural sayısı daha büyük bir yüzdeyle azalmaktadır.

Hiyerarşik bulanık mantık denetleyicilerin, düşük boyutlu bulanık mantık denetleyicilerle hiyerarşik formda birleştirilmesi ile oluşumu değişik şekillerde olabilir (Kikuchi ve ark, 1998). Bu oluşumlar için Şekil 2.15'de gösterildiği gibi örnekler verebiliriz.



Şekil 2.14. Klasik bulanık mantık denetleyici yapısı

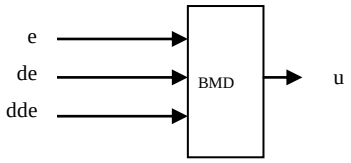


Şekil 2.15. Hiyerarşik bulanık mantık denetleyici yapısı

Hiyerarşik bulanık mantık denetleyicilerde, denetleyici çeşitli sayılarda katmanlardan oluşur. Bir katmanın çıkışı diğer bir katman için giriş halini alır. Ancak burada tasarımı zora sokan bir durum ortaya çıkar. Katmanların sahip olduğu ve bir sonraki katmanın girişi olacak ara çıkışlar yapaydır ve fiziksel bir anlama sahip değildir (Joo ve Lee, 1999). Bu yüzden bir sonraki katmanın girişi anlamsız olur. Tasarımı zorlaştıran bu anlamsızlık, özellikle katmanların sayısı arttıkça kolayca görünür hale gelir.

Bu sorunun üstesinden gelmek için, bir önceki katmanın çıkışlarının ve bir sonraki katmanın girişlerinin tanımlandığı ara değişkenleri içerecek yeni kural tabloları oluşturmak uygun olacaktır (Lee ve ark, 2003).

Bu tabloların oluşumunu ve bir hiyerarşik bulanık mantık denetleyicinin tasarımını incelemek amacıyla Şekil 2.16'daki gibi 3 giriş ve 1 çıkıştan oluşan ve Çizelge 2.2'de verilen rastgele hazırlanmış bir kural tablosuna sahip sistemi ele alalım.

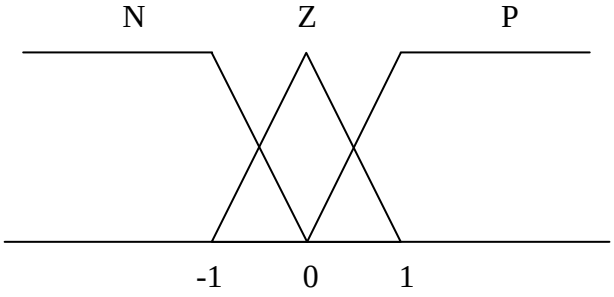


Şekil 2.16. 3 giriş 1 çıkışlı bulanık mantık denetleyici

Çizelge 2.2. Rastgele hazırlanmış kural tablosu

dde	**e**								
	P			**Z**			**N**		
	de			**de**			**de**		
	P	**Z**	**N**	**P**	**Z**	**N**	**P**	**Z**	**N**
P	P	P	P	P	P	Z	Z	N	N
Z	P	P	P	P	Z	N	N	N	N
N	P	P	Z	Z	N	N	N	N	N

Örnek sistemde bulanık mantık denetleyicinin girişleri e, de ve dde; çıkışı ise u ile isimlendirilmiştir. Giriş ve çıkış değişkenlerinin tamamı için üyelik fonksiyonları Şekil 2.17'deki gibi P (Pozitif), Z (Sıfır) ve N (Negatif) olarak tanımlanmış ve kural tablosuna bu ifadelerle işlenmiştir.

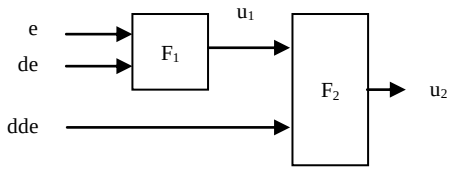


Şekil 2.17. Giriş ve çıkış üyelik fonksiyonları

Oluşturduğumuz 3 girişli ve 1 çıkışlı bu bulanık mantık denetleyiciyi hiyerarşik yapıya dönüştürürken kural tablosunun eş sütunlara veya eş satırlara göre gruplanmasıyla iki farklı yapı oluşturulabilir. Bunlara da "Sütun Hiyerarşi Yapısı" ve "Satır Hiyerarşi Yapısı" isimleri verilebilir.

Sütun Hiyerarşi Yapısında HBMD

Şekil 2.18'deki gibi bir sütun hiyerarşi yapısında hiyerarşik bulanık mantık denetleyici tasarlanırken klasik bulanık mantık denetleyici için oluşturulan kural tablosundaki kurallara bakılarak, aynı kurallardan oluşan sütunlara A, B, C … gibi bir grup ismi verilir ve böylece katmanlar arası ihtiyaç duyulan ara değişkenler tanımlanmış olur. Bu değişkenler bir önceki katmanın çıkışı ve bir sonraki katmanın girişi için ara üyelik fonksiyonlarını tanımlarlar.



Şekil 2.18. Sütun hiyerarşi yapısında HBMD

Daha önce oluşturduğumuz sistem için Çizelge 2.2'de verilen kural tablosunda sütun gruplama Çizelge 2.3'deki gibi yapılır.

Çizelge 2.3. Sütun hiyerarşi yapısında gruplanmış kural tablosu

dde	**e**								
	P			**Z**			**N**		
	de			**de**			**de**		
	P	**Z**	**N**	**P**	**Z**	**N**	**P**	**Z**	**N**
P	P	P	P	P	P	Z	Z	N	N
Z	P	P	P	P	Z	N	N	N	N
N	P	P	Z	Z	N	N	N	N	N
	A	A	B	B	C	D	D	E	E

Çizelge 2.3'de yapılan gruplama neticesinde e ve de girişlerine ve u_1 çıkışına sahip F_1 bloğu için giriş üyelik fonksiyonları P, Z ve N iken çıkış üyelik fonksiyonları A, B, C, D ve E olarak isimlendirilmiş ara üyelik fonksiyonları olmuştur. Böylece ara katmanlara anlam kazandırılmıştır. Bu ara üyelik fonksiyonları F_1'in çıkış, F_2'nin ise giriş ara üyelik fonksiyonlarıdır. F_2'nin çıkış üyelik fonksiyonları yine P, Z ve N'dir. Bu durum sonucunda elde edilen ara üyelik fonksiyonlarıyla F_1 ve F_2 için Çizelge 2.4 ve Çizelge 2.5'de verilen iki yeni kural tablosu meydana gelmiştir.

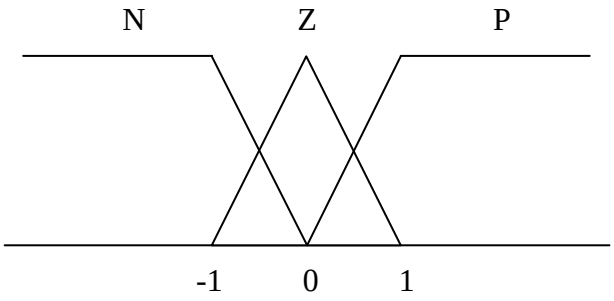
Çizelge 2.4. F_1 bloğu için kural tablosu

de	**e**		
	P	**Z**	**N**
P	A	B	D
Z	A	C	E
N	B	D	E

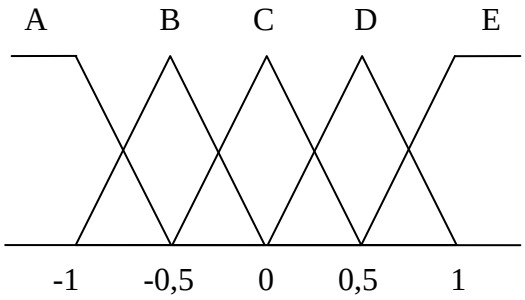
Çizelge 2.5. F_2 bloğu için kural tablosu

dde	**u_1**				
	A	**B**	**C**	**D**	**E**
P	P	P	P	Z	N
Z	P	P	Z	N	N
N	P	Z	N	N	N

Oluşturulan hiyerarşik yapı için giriş ve çıkış değişkenlerinin üyelik fonksiyonları Şekil 2.19 ve ara üyelik fonksiyonu da Şekil 2.20'deki gibidir.



Şekil 2.19. Sütun hiyerarşi yapısındaki giriş ve çıkış üyelik fonksiyonları

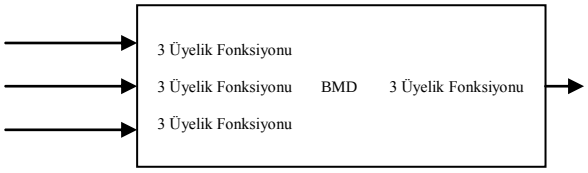


Şekil 2.20. Sütun hiyerarşi yapısındaki ara üyelik fonksiyonları

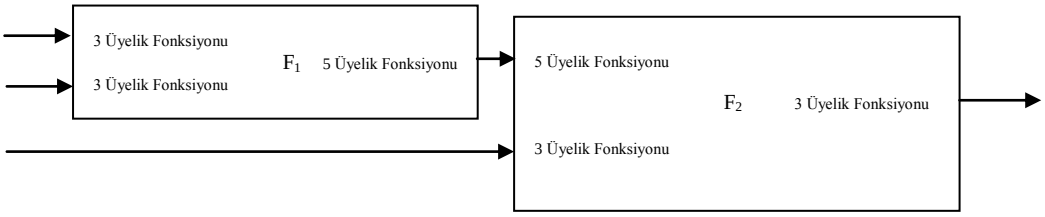
Burada giriş, ara ve çıkış değişkenlerinin üyelik fonksiyonlarının tanım aralığının aynı olmasına ve kullanılacak değişken sayısına göre eşit aralıklarla yerleştirilmiş olmasına dikkat edilmesi gerekir (Sağlam, 2007).

Tasarım işleminde tanımlanan ara üyelik fonksiyonları göz önüne alındığında tanımlanması gereken kural sayısı ara üyelik değişken sayısına göre de değişmektedir. Daha önceden belirtildiği gibi $(n-1)\cdot m^2$ olması gereken toplam kural sayısı ara üyelik değişken sayısına göre daha büyük veya daha küçük olabilmektedir.

Örneğimizde 3 girişli ve her giriş değişkeni için 3 üyelik fonksiyonuna sahip bir bulanık mantık denetleyici için Şekil 2.21'de görülebileceği gibi $3^3=27$ kural bulunmaktaydı. Ancak aynı örneği Şekil 2.22'de görülebileceği gibi sütun hiyerarşi yapıda oluşturduğumuz durumda ise, 2 girişli ve her giriş değişkeni için 3 üyelik fonksiyonuna sahip birinci katmanda (F_1 bloğu) $3^2=9$ kural; 2 giriş değişkeni ve birinci değişken için 5, ikinci değişken için de 3 üyelik fonksiyonuna sahip ikinci katmanda (F_2 bloğu) $3\cdot5=15$ kural bulunmaktadır. İki farklı yapı karşılaştırıldığında 27 kurala sahip bir BMD, sütun hiyerarşik yapıyla yeniden tasarlandığında kural sayısı 9+15=24'e düşmektedir.



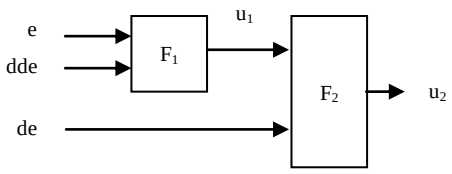
Şekil 2.21. $3\cdot3\cdot3=3^3=27$ kurallı BMD



Şekil 2.22. F_1 için $3\cdot3=9$ kurallı, F_2 için $5\cdot3=15$ kurallı, toplamda 9+15=24 kurallı HBMD

Satır Hiyerarşi Yapısında HBMD

Şekil 2.23'deki gibi bir satır hiyerarşi yapısında hiyerarşik bulanık mantık denetleyici tasarlanırken klasik bulanık mantık denetleyici için oluşturulan kural tablosundaki kurallara bakılarak, aynı kurallardan oluşan satırlara A, B, C … gibi bir grup ismi verilir ve böylece katmanlar arası ihtiyaç duyulan ara değişkenler tanımlanmış olur. Bu değişkenler bir önceki katmanın çıkışı ve bir sonraki katmanın girişi için ara üyelik fonksiyonlarını tanımlarlar.



Şekil 2.23. Satır hiyerarşi yapısında HBMD

Daha önce oluşturduğumuz sistem için Çizelge 2.2'de verilen kural tablosunda satır gruplama Çizelge 2.6'daki gibi yapılır.

Çizelge 2.6. Satır hiyerarşi yapısında gruplanmış kural tablosu

dde	**e**								
	P			**Z**			**N**		
	de			**de**			**de**		
	P	**Z**	**N**	**P**	**Z**	**N**	**P**	**Z**	**N**
P	A			B			D		
	P	P	P	P	P	Z	Z	N	N
Z	A			C			E		
	P	P	P	P	Z	N	N	N	N
N	B			D			E		
	P	P	Z	Z	N	N	N	N	N

Çizelge 2.6'da yapılan gruplama neticesinde e ve dde girişlerine ve u_1 çıkışına sahip F_1 bloğu için giriş üyelik fonksiyonları P, Z ve N iken çıkış üyelik fonksiyonları da A, B, C, D ve E olmuştur. Bahsedilen A, B, C, D ve E gruplama neticesinde elde edilmiş ara üyelik fonksiyonları olmuştur. Böylece ara katmanlara anlam kazandırılmıştır. Bu ara üyelik fonksiyonları F_1'in çıkış, F_2'nin ise giriş ara üyelik fonksiyonlarıdır. F_2'nin çıkış üyelik fonksiyonları yine P, Z ve N'dir. Bu durum sonucunda elde edilen ara üyelik fonksiyonlarıyla F_1 ve F_2 için, Çizelge 2.7 ve Çizelge 2.8'deki gibi iki yeni kural tablosu meydana gelmiştir.

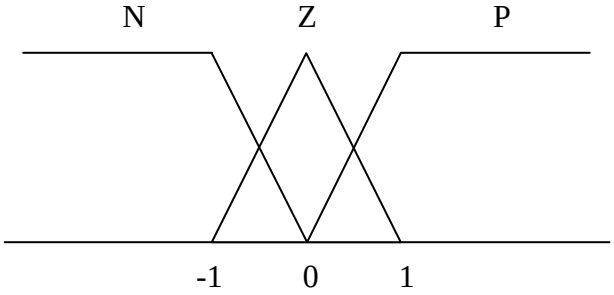
Çizelge 2.7. F_1 bloğu için kural tablosu

dde	**e**		
	P	**Z**	**N**
P	A	B	D
Z	A	C	E
N	B	D	E

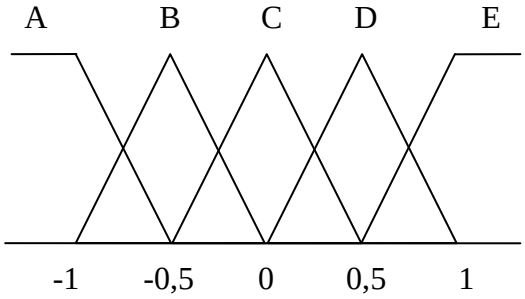
Çizelge 2.8. F_2 bloğu için kural tablosu

u_1	de		
	P	**Z**	**N**
A	P	P	P
B	P	P	Z
C	P	Z	N
D	Z	N	N
E	N	N	N

Oluşturulan hiyerarşik yapı için giriş ve çıkış değişkenlerinin üyelik fonksiyonları Şekil 2.24 ve ara üyelik fonksiyonu da Şekil 2.25'deki gibidir.



Şekil 2.24. Satır hiyerarşi yapısındaki giriş ve çıkış üyelik fonksiyonları



Şekil 2.25. Satır hiyerarşi yapısındaki ara üyelik fonksiyonları

Görüldüğü gibi, örneğimizi satır hiyerarşik yapıda oluşturduğumuz zaman da, 2 girişli ve her giriş değişkeni için 3 üyelik fonksiyonuna sahip birinci katmanda $3^2=9$ kural; 2 giriş değişkeni ve birinci değişken için 5, ikinci değişken için de 3 üyelik fonksiyonuna sahip ikinci katmanda $3 \cdot 5=15$ kural bulunmaktadır.

Örneğimizdeki bulanık mantık denetleyici ile satır hiyerarşik bulanık mantık denetleyici yapıları karşılaştırıldığında 27 kurala sahip BMD, satır hiyerarşik yapıyla yeniden tasarlandığında kural sayısı 9+15=24'e düşmektedir.

Simetrik kural tablolarında kurallar, satır tarafından bakıldığında da, sütun tarafından bakıldığında da aynı sıralamaya sahiptir. İster satırlarda isterse sütunlarda gruplama yapılsın, gruplama değişmez. Bu nedenle satır gruplamayla sütun gruplamada elde edilecek ara değişken sayısı aynı olur. Dolayısıyla kural sayıları da eşittir (Sağlam, 2007).

Simetrik kural tabloları daha çok doğrusal sistemlerde etkili olmaktadır. Doğrusal olmayan sistemlerde ise asimetrik yapıdaki kural tabloları daha iyi sonuçlar verir. Asimetrik kural tablolarında gruplamanın satır veya sütunda yapılmasına göre ara üyelik fonksiyonlarının sayısı da değişir. Bu da sistemdeki kural sayısının değişmesine neden olur (Sağlam, 2007).

Örnek bir asimetrik kural tablosu Çizelge 2.9'daki gibi olabilir.

Çizelge 2.9. Asimetrik kural tablosu

dde	**e**								
	P			**Z**			**N**		
	de			**de**			**de**		
	P	**Z**	**N**	**P**	**Z**	**N**	**P**	**Z**	**N**
P	P	P	P	P	P	Z	Z	Z	Z
Z	P	P	P	P	P	N	N	N	N
N	P	Z	Z	Z	Z	N	N	N	N

Çizelge 2.9'da verilen asimetrik kural tablosunda yapılacak satır ve sütun gruplaması Çizelge 2.10 ve Çizelge 2.11'deki gibi olacaktır.

Çizelge 2.10. Sütun şeklinde gruplanmış asimetrik kural tablosu

dde	**e**								
	P			**Z**			**N**		
	de			**de**			**de**		
	P	**Z**	**N**	**P**	**Z**	**N**	**P**	**Z**	**N**
P	P	P	P	P	P	Z	Z	Z	Z
Z	P	P	P	P	P	N	N	N	N
N	P	Z	Z	Z	Z	N	N	N	N
	A	B	B	B	B	C	C	C	C

Çizelge 2.11. Satır şeklinde gruplanmış asimetrik kural tablosu

dde	**e**								
	P			**Z**			**N**		
	de			**de**			**de**		
	P	**Z**	**N**	**P**	**Z**	**N**	**P**	**Z**	**N**
P	A			C			F		
	P	P	P	P	P	Z	Z	Z	Z
Z	A			D			G		
	P	P	P	P	P	N	N	N	N
N	B			E			G		
	P	Z	Z	Z	Z	N	N	N	N

Yapılan gruplamada görüleceği gibi sütun hiyerarşi yapı sonucunda A, B ve C gibi üç ara değişken ortaya çıkarken, satır hiyerarşi yapı sonucunda ise A, B, C, D, E, F ve G gibi 7 ara değişken ortaya çıkmıştır. Bunun neticesi olarak da sütun hiyerarşi ile oluşturulacak HBMD'nin $3 \cdot 3+3 \cdot 3=18$ kuralı olacakken, satır hiyerarşi ile oluşturulacak HBMD'nin $3 \cdot 3+7 \cdot 3=30$ kuralı olacaktır.

2.4. Parçacık Sürü Optimizasyonu (PSO)

Parçacık sürü optimizasyonu (PSO), kuş sürülerinin davranışlarından esinlenerek geliştirilmiş bir optimizasyon yöntemidir (Ying ve ark, 2007). 1995 yılında R. Eberhart ve J. Kennedy tarafından geliştirilen bu yöntem doğrusal olmayan problemlere ve çok parametreli, çok değişkenli optimizasyon problemlerine çözüm bulmak amaçlı kullanılmaktadır (Chang ve Shih, 2010).

Yöntem, kuş sürülerinin yiyecek arayışları sırasında sergiledikleri toplu hareketlerden esinlenerek geliştirilmiştir. Kuş sürüsünün bir bölgedeki yiyecek arayışı, kuşların o bölgeye rastgele dağılımı ile gerçekleşir. Kuşlar yiyeceğin nerede olduğunu, eş zamanlı olarak, ani hareketlerle yön değiştirip arama bölgesine yayılarak ararlar. Daha sonra, yine eş zamanlı olarak, tekrar bir araya gelirler ve yiyeceğin nerede olduğu konusunda bilgi paylaşımı yaparlar. Bu paylaşımla yiyeceğe ne kadar mesafede olduklarını ve sürüdeki yiyeceğe en yakın kuşun pozisyonunu belirlerler. Böylece elde ettikleri pozisyon bilgilerini kullanarak yiyeceğe ulaşırlar (Lin ve ark, 2008).

PSO, arama uzayında optimum noktayı ararken, optimize edilecek olan fonksiyonun olası çözümlerini barındıran bir popülasyon kullanması açısından, genetik algoritma gibi hesaplama teknikleriyle benzerlik gösterir (Aygün, 2011). Ancak

PSO'nun bazı özellikleriyle diğer optimizasyon tekniklerinden daha üstün olduğu söylenebilir (Ou ve Lin, 2006). Örneğin, ayarlanması gereken parametre sayısının az olması, PSO'yu diğer optimizasyon yöntemlerine nazaran uygulaması daha kolay hale getirir. Bunun dışında PSO'da parçacıkların hem kendi en iyi pozisyon değerlerini hem de sürüdeki diğer komşularının en iyi pozisyon değerlerini hatırlamaları PSO'nun iyi bir hafıza yeteneğine sahip olduğunu gösterir. Ayrıca PSO'da çözüm uzayında arama yapılırken en iyi pozisyona sahip parçacığın değerinden yararlanılır ve arama uzayında herhangi bir değişiklik olmaz. Ancak, örneğin genetik algoritmalarda kötü sonuçlar devre dışı bırakılır ve arama uzayı daraltılır. PSO'da arama uzayında herhangi bir değişikliğe ya da kısıtlamaya gidilmemesi PSO'nun yerel optimumlara takılmasını engeller (Ortakcı, 2011).

PSO'da parçacıklar, boyutu, optimize edilmesi istenen parametrelerin sayısı ile belirlenen, d boyutlu arama uzayında dolaşan birer noktadır. Denklem (2.8)'de görüleceği gibi m adet parçacığın her birinin iterasyondaki konumu (X_i), d boyutlu birer diziyle;

$$X_i=(X_{i1}, X_{i2}, X_{i3} \dots X_{id}),\ i=1, 2, 3 \dots m \qquad (2.8)$$

şeklinde gösterilir.

Parçacıkların konum değiştirme ivmeleri ise hız (V_i) olarak tanımlanır ve denklem (2.9)'daki gibi d boyutlu birer diziyle;

$$V_i=(V_{i1}, V_{i2}, V_{i3} \dots V_{id}),\ i=1, 2, 3 \dots m \qquad (2.9)$$

şeklinde gösterilir.

Her bir parçacığın en iyi konum bilgisi, parçacığın kişisel en iyi konumu (p_i) olarak tanımlanır ve denklem (2.10)'da verildiği gibi d boyutlu bir dizi ile;

$$p_i=(p_{i1}, p_{i2}, p_{i3} \dots p_{id}),\ i=1, 2, 3 \dots m \qquad (2.10)$$

şeklinde gösterilir (Aygün, 2011).

Global en iyi konum (g_i) ise elde edilen tüm p_i'ler arasından elde edilecek en iyi konum olarak tanımlanır.

Bu tanımlamalara göre her bir parçacığın hızı ve konumu denklem (2.11) ve denklem (2.12) ile güncellenir.

$$V_{id}^{t+1}=V_{id}^{t}+c_1 \cdot r_1 \cdot (p_{id}^{t}-X_{id}^{t})+c_2 \cdot r_2 \cdot (g_i^{t}-X_{id}^{t}) \quad (2.11)$$

$$X_{id}^{t+1}=X_{id}^{t}+V_{id}^{t+1} \quad (2.12)$$

Bu eşitliklerde c_1 ve c_2, parçacığı kişisel en iyi konuma (p_{eniyi}) ve global en iyi konuma (g_{eniyi}) doğru çeken, hızlandırma katsayılarıdır. [0,4] aralığında seçilen bu değerler genelde 2 olarak alınır ve birbirine eşittir (Gaing, 2004). Ancak bu bir zorunluluk değildir. r_1 ve r_2 ise [0,1] aralığında rastgele değerler alan ve PSO'nun rastgeleliğini sağlayan parametrelerdir. V_{id}^{t+1} olarak gösterdiğimiz yeni hız bilgisini kullanarak elde edilen ve X_{id}^{t+1} olarak gösterdiğimiz değer ise yeni konum bilgisi olacaktır.

PSO'da sonlandırma kriteri, kullanıcı tarafından belirtilmiş sabit bir iterasyon sayısı ya da belirlenmiş bir çalışma zamanı veya son iki iterasyonun g_{eniyi} değerleri arasındaki farkın belirli bir değerin altına düşmesi gibi durumlar olabilir (Ortakcı, 2011).

1995 yılında Eberhart ve Kennedy tarafından geliştirilen PSO, 1998 yılında Shi ve Eberhart tarafından bir atalet ağırlığı "w" terimi eklenerek modifiye edilmiştir. Bu eklenti neticesinde hız denklemi denklem (2.13)'deki gibi güncellenmiştir.

$$V_{id}^{t+1}=w \cdot V_{id}^{t}+c_1 \cdot r_1 \cdot (p_{id}^{t}-X_{id}^{t})+c_2 \cdot r_2 \cdot (g_i^{t}-X_{id}^{t}) \quad (2.13)$$

Atalet ağırlığının büyük değerler alması parçacığın çözüm uzayında daha genel (global) aramalar yapmasını, küçük değerler alması ise daha bölgesel (kişisel) aramalar yapmasını sağlamaktadır. Atalet ağırlığı, bütün arama işlemi boyunca sabit bir değer olarak kullanılabileceği gibi, iterasyonlar ilerledikçe azalacak şekilde dinamik olarak da kullanılabilir (Xu-zhou ve ark, 2007).

w değerinin her iterasyonda doğrusal olarak veya doğrusal olmayarak azalması, iterasyonun başlangıcında global arama yeteneğinin, sonlarına doğru ise kişisel arama yeteneğinin daha yüksek olmasını sağlayabilmektedir. Bu sayede optimizasyon performansında da artış sağlanabilmektedir (Aygün, 2011).

Uygulamalarda, genel olarak, atalet ağırlığının en büyük değeri w_{mak}=0.9 ve en küçük değeri w_{min}=0.4 olarak alınır (Hu ve ark, 2005).

Atalet ağırlığının dinamik değeri denklem (2.14)'de verilen;

$$w = w_{mak} - t \cdot \frac{w_{mak} - w_{min}}{t_{mak}} \quad (2.14)$$

bağıntısı ile bulunur. Burada t iterasyon sayısını göstermektedir (Aygün, 2011).

1999 yılında ise Clerc, w atalet ağırlığı yerine optimizasyon yakınsama performansını arttırmak amacıyla bir "K" faktörü kullanmayı önermiştir. Denklem (2.15)'de görüleceği gibi K faktörü ile hız denklemi;

$$V_{id}^{t+1}=K\cdot[V_{id}^{t}+c_1\cdot r_1\cdot(p_{id}^{t}-X_{id}^{t})+c_2\cdot r_2\cdot(g_i^{t}-X_{id}^{t})] \quad (2.15)$$

şeklini almıştır. Burada K denklem (2.16)'dan görüleceği gibi;

$$K = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|} \quad (2.16)$$

şeklindedir. Burada $\varphi = c_1 + c_2 > 4$ olarak tanımlanır (Pillay ve Govender, 2007).

2.5. Gerçek Kodlu Genetik Algoritmalar (GA)

Genetik algoritmalar (GA), biyolojik bir süreci modelleyerek, bir fonksiyonu ya da bir sistemi optimize eden evrim algoritmalarıdır. Biyolojideki genleri genetik algoritmada parametreler temsil etmektedir. Parametrelerin toplu kümesi ise kromozomu oluşturmaktadır. Genetik algoritmaların her bir ferdi kromozomlar şeklinde temsil edilen popülasyonlardan oluşur. Genetik algoritmada yeni nesiller, rastgele bilgi değişimi ile oluşturulan diziler içinde hayatta kalanların birleştirilmesi ile elde edilmektedir (Angeline, 1995).

İlk olarak 1975 yılında Michigan Üniversitesi'nde John Holland ve arkadaşları tarafından tanıtılan genetik algoritmaların ilk uygulaması, boru hatları ile ilgili bir problemin çözümü amacıyla David Goldberg tarafından yapılmıştır (Goldberg, 1989).

Genetik algoritmalar bireylerin popülasyonu ile çalışarak bir probleme olası bir çözüm sunarlar. Genetik algoritmaların işleyişinde her bir bireye, verilen problemi ne kadar iyi çözdüğüne göre bir uygunluk puanı verilir. Uygunluk puanı, bir sistem için performans indeksi veya bir tasarım için kuvvet/ağırlık oranı olabilir. Bunun neticesinde

uygunluk puanı yüksek bireyler oldukça büyük sayıda ürerler, uygunluk puanı düşük bireyler ise çok az üreyebilirler veya hiç üreyemezler. Böylece yeni bireyler, ortalaması giderek artan bir uygunluk sağlarlar (Aydoğdu, 2006).

Olası çözümlerin yeni bir popülasyonu, mevcut nesillerin en iyi bireylerinin seçilerek, yeni birey kümesi oluşturmak için çiftleştirilmesiyle oluşturulur. Optimizasyon açısından bakıldığında, rastgele üretilen bir başlangıç popülasyonuyla arama uzayında geniş bir alan araştırılır ve daha sonra uygun bireylerin birbirleri ile çiftleştirilmesiyle arama uzayının en uygun bölgesi belirlenir. Sonuçta genetik algoritma muhtemel global optimum noktasını bulabilir. Eğer genetik algoritma iyi tasarlanmış ise popülasyon optimal çözüm noktasına yaklaşacaktır (Çunkaş, 2004).

Genetik algoritmalar, global optimal çözümü elde etmede daima garanti veremez. Ancak genellikle bu sağlanır. Ayrıca her problemin çözümünde genetik algoritmalar kullanmak iyi bir yol değildir. Örneğin birkaç parametreden oluşan bir analitik fonksiyonun çözümünde klasik metotlar daha hızlıdır.

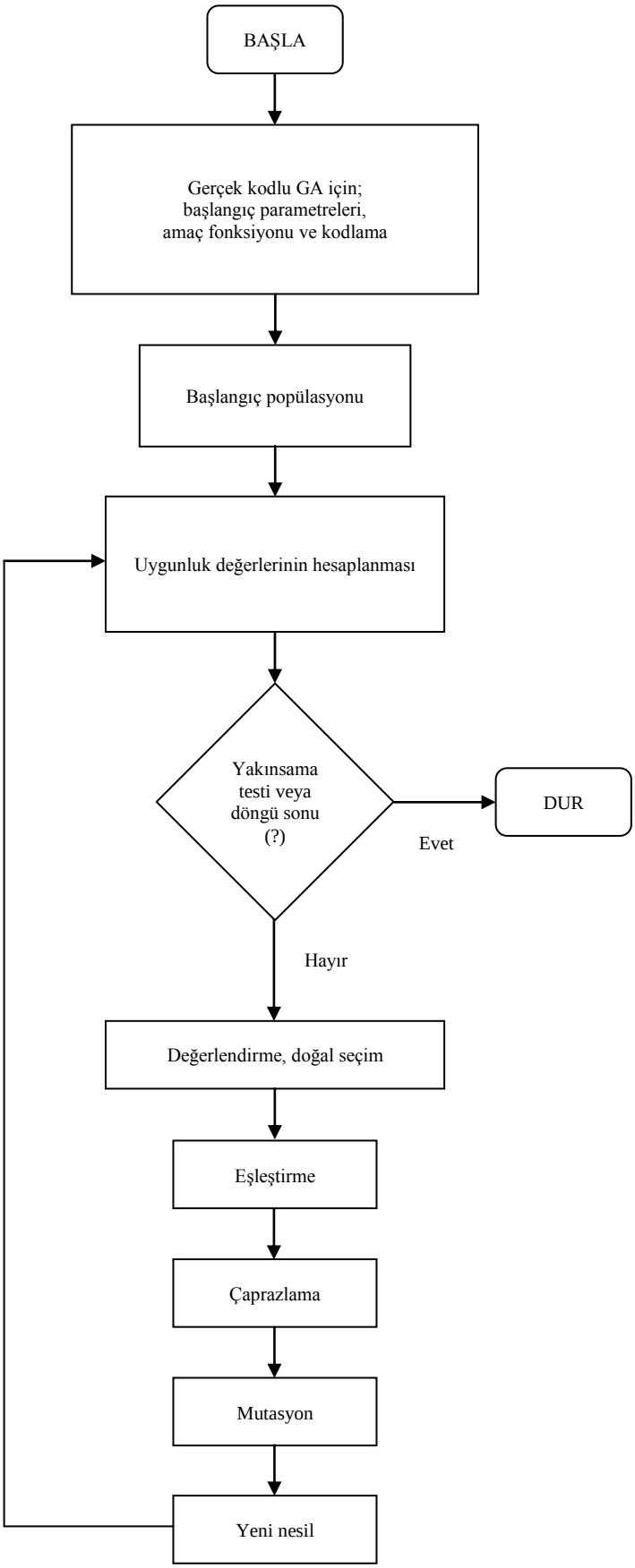
Genetik algoritmalarla optimizasyon yapılırken, optimize edilecek parametreler için ya rastgele bir değer kümesi üretilir ya da bir başlangıç çözüm kümesi tanımlanır. Optimizasyon yapabilmek için bu parametrelerin tanımlanması ve uygun biçimde kodlanması gerekir. Genetik algoritmalar, parametrelerin kodlama biçimine göre ikili kodlu genetik algoritmalar (Binary Coded GA) ve gerçek kodlu genetik algoritmalar (Real Coded GA) olarak ikiye ayrılır. Kodlama biçimi, algoritmada yürütülen matematiksel işlemler açısından oldukça önemlidir.

Bu çalışmada gerçek kodlu genetik algoritma kullanılmış ve kısaca incelenmiştir.

Problemlerin çözümünde rakamların hassasiyetinin önemli bir yer tutması ve ikili kodlu genetik algoritmalarda, parametrelerin “1” ve “0” ile ifade edilmesi neticesinde kromozom boyutlarının oldukça artması, gerçek rakamlarla kodlama yapabilen gerçek kodlu genetik algoritmaları kullanmayı avantajlı hale getirmektedir. “1” ve “0”larla oluşturulacak büyük boyutlu kromozomların yerine gerçek rakamlar kullanılarak oluşturulacak daha küçük boyutlu kromozomların kullanılması, gerçek kodlu genetik algoritmaların ikili kodlu genetik algoritmalardan daha hassas olmasını ve PC belleğinde daha az yer kaplamasını sağlamaktadır (Aydoğdu, 2006).

Literatürde, gerçek kodlu genetik algoritmaların, ikili kodlu genetik algoritmalardan daha hızlı çalıştığı ve daha kısa sürede global optimumu bulduğu belirtilmektedir (Çunkaş ve Akkaya, 2002).

Gerçek kodlu genetik algoritmaların akış diyagramı Şekil 2.26'daki gibidir.



Şekil 2.26. Gerçek kodlu genetik algoritmaların akış diyagramı

2.5.1. Parametrelerin kodlanması ve amaç fonksiyonu

Genetik algoritmalarda optimize edilecek parametrelerin her biri bir gen (gen_n) ile ifade edilmektedir. Burada her bir gen, on tabanlı bir sayı dizisiyle temsil edilir. Parametreleri temsil eden bu genlerin topluluğuna kromozom denir. N_{par} adet gene sahip bir kromozom $1 \cdot N_{par}$ boyutlu bir matrisle denklem (2.17)'deki gibi yazılır (Aydoğdu, 2006).

$$Kromozom = [gen_1, gen_2, gen_3, \dots, gen_{N_{par}}] \tag{2.17}$$

Gerçek kodlu genetik algoritmalarda kromozomların her birinin bir uygunluk değeri (C) vardır. Tanımlanan amaç fonksiyonuna (f) göre uygunluk değerleri denklem (2.18)'deki gibi;

$$C = f(kromozom) = f(gen_1, gen_2, gen_3, \dots, gen_{N_{par}}) \tag{2.18}$$

şeklinde ifade edilir.

2.5.2. Başlangıç popülasyonu

Genetik algoritmalarda olası çözüm kümesi tek bir kromozom ile ifade edilmez. Algoritmanın yürütüleceği ortam kapasitesi ve parametre karakteristikleri de dikkate alınarak popülasyon olarak tanımlanan bir kromozom kümesi oluşturulur. N_{ipop} kromozomdan oluşan başlangıç popülasyonu, $N_{par} \cdot N_{ipop}$'luk bir matris ile temsil edilir.

Başlangıç popülasyonu iki şekilde oluşturulur. Birincisinde; eğer, geniş arama yüzeyinde lokal bir bölgede sonuç aranacaksa, o bölgeye yakın noktalardan oluşan bir başlangıç popülasyonu tanımlanabilir. Böylece, algoritmanın daha kısa sürede sonuca ulaşması sağlanabilir. İkincisinde; kromozomları oluşturacak genler, denklem (2.19) ile verilen rastgele sayı üreteci ile elde edilir.

$$\text{İPOP} = (P_{üst} - P_{alt}) \cdot \text{rasgele}\ \{N_{ipop}, N_{par}\} + P_{alt} \tag{2.19}$$

Burada; $P_{üst}$, parametrenin üst sınır değeri; P_{alt}, parametrenin alt sınır değeri ve rastgele $\{N_{ipop}, N_{par}\}$ ise $N_{ipop} \cdot N_{par}$ olacak şekilde, matris formunda, 0-1 arasında üretilen rastgele sayıdır.

2.5.3. Uygunluk değerinin hesaplanması

Parametrelerin değerleri belirlendikten sonra, amaç fonksiyonundan yararlanılarak uygunluk değerleri (C) hesaplanır. Burada; amaç fonksiyonu karmaşık bir yapıya sahip ise, popülasyon sayısının yüksek tutulması iyi sonuç vermektedir. Başlangıç popülasyonunun büyüklüğü, araştırma uzayının daha geniş seçilmesini sağlamaktadır. Genetik algoritmaların döngüsünün sona ermesi için, algoritmada bir yakınsama testi veya döngü sayısı tanımlanmalıdır.

2.5.4. Değerlendirme ve doğal seçim

Genetik algoritmanın her döngüsünde, her bir kromozomun problemi ne kadar iyi çözebileceğini belirlemek ilk adımdır. Bu adım değerlendirme olarak bilinir. Değerlendirme, bir sonraki adımda ne kadar yeni çiftin üretilmesi gerektiğini belirlemede yardımcı olmaktadır.

Genetik algoritmalarda gelecek nesilde hangi kromozomun yer alacağının belirlenmesi gerekir. Gerçek kodlu genetik algoritmalarda, toplam N_{ipop} kadar kromozom olduğu düşünülürse, gelecek iterasyonda kullanılmak üzere N_{pop} kadar kromozom tutulur ve geri kalanı dikkate alınmaz ($N_{pop}<N_{ipop}$). N_{pop} kromozomları da, kendi aralarında iyilerden kötülere doğru sıralamaya tabi tutularak N_{iyi} ve $N_{kötü}$ olarak iki kısma ayrılır. N_{iyi} eşleştirme havuzuna atılırken, $N_{kötü}$ hariçte tutulur (Aydoğdu, 2006).

Doğal seçim, algoritmanın her bir iterasyonunda gerçekleşir. Elde tutulacak kromozom sayısı biraz keyfidir. Popülasyon içerisinde bütün kromozomların seçilmesinin gelecek nesle çok fazla katkısı olmamaktadır.

Genetik algoritmalarda ağırlıklı seçim metodu, eşik değer seçim metodu, rulet tekerleği seçim metodu gibi değişik seçim metotları vardır.

Ağırlıklı Seçim Metodu

Bu metotta ilk olarak kromozomların uygunluk değerleri hesaplanır. Hesaplanan değerler, iyiden kötüye doğru sıralanır ve N_{ipop} kadar kromozomdan N_{pop} kadarı tutulur ve geriye kalanlar atılır. Burada N_{pop} değeri N_{ipop}'a kadar olabilir ancak genelde $N_{pop}=N_{ipop}/2$ uygun seçenektir. Seçilen bu N_{pop} kadar kromozomun da yarısı N_{iyi}, yarısı da $N_{kötü}$ olarak ayrılır ve N_{iyi}'ler eşleştirme havuzuna konulurken, $N_{kötü}$ olanlar eşleştirme havuzundan atılır.

Eşik Değer Seçim Metodu

Bu metotta bir eşik uygunluk değeri tanımlanır ve bu tanımlanan değerle, her bir bireyin uygunluk değeri karşılaştırılır. Eşik değerden daha iyi olan kromozomlar yaşamaya devam ederken, eşik değerinden daha kötü olan kromozomlar ölür. Bu metotta popülasyonun sıralanmasına gerek yoktur.

Rulet Tekerleği Seçim Metodu

Bu metotta ilk olarak kromozomların uygunluk değerleri ve uygunluk değerlerinin toplamı alınarak toplam uygunluk değeri hesaplanır. Kromozomlar, toplam uygunluk değerine bölünerek her bir kromozom için seçim ihtimalleri bulunur. Daha sonra seçim ihtimalleri birbirlerine eklenerek kümülatif ihtimaller hesaplanır. Son olarak popülasyon sayısı kadar 0-1 arasında rastgele sayılar üretilir ve bu sayılar birinci kromozomun kümülatif seçim ihtimalinden küçük ise birinci kromozom; değilse ikinci kromozom veya diğerlerinin kümülatif ihtimalleriyle karşılaştırılarak hangisinden küçükse o kromozom seçilir ve rulet seçim metodu gerçekleştirilmiş olur.

2.5.5. Eşleştirme

Eşleştirmede doğal seçim yoluyla elde edilen N_{iyi} kromozomların bulunduğu eşleştirme havuzundan iki tane kromozom seçilir. Seçilen bu iki kromozom arasında gerçekleştirilecek eşleştirme için değişik yöntemler vardır.

Yukarıdan Aşağıya Doğru Eşleştirme

Bu yöntemde listenin en üstünden başlanır ve eşleştirme için seçilen N_{iyi} kromozomların sayısına ulaşana kadar sırayla ikişer kromozom eşleştirilerek devam edilir. Böylece $kromozom_{2i-1}$ ile $kromozom_{2i}$ (i=1,2,3,… için) eşleştirilmiş olur.

Rastgele Eşleştirme

Bu yöntemde kromozomların eşleştirilmesi için rastgele sayı üreteci kullanılır. Kromozomlar birinciden başlayarak N_{iyi}'nciye kadar sıralanır. Birinci eşleştirmeyi bulmak için denklem (2.20) kullanılarak iki adet rastgele sayı üretilir.

$$\text{Kromozom} = \text{Roundup}\ \{N_{iyi} \cdot \text{rastgele sayı}\} \qquad (2.20)$$

Burada "roundup" fonksiyonu, sayıları bir üst tam sayıya yuvarlar ve bulunan tam sayıların ifade ettiği kromozomlar eşleştirilir. İkinci ve sonraki eşleştirmeleri bulmak için aynı işlemler tekrarlanır.

Ağırlıklı Rastgele Eşleştirme

Bu yöntemde; eşleştirme havuzundaki kromozomların, amaç fonksiyonunda hesaplanan uygunluk değerlerine göre seçilme ihtimalleri dikkate alınır. En iyi uygunluk değerine sahip kromozom, eşleştirme ihtimali en yüksek kromozom olurken, en kötü uygunluk değerine sahip kromozom eşleştirme ihtimali en düşük olan olarak düşünülür. Üretilen rastgele sayı, hangi kromozomun seçileceğini belirler. Bu tip uygulamaya "Rulet Tekerleği Ağırlık Yöntemi" de denir. Bu yöntem kendi arasında ikiye ayrılır:

1) Sıralama (Rank-Based) Ağırlık: Bu yöntemde kromozomlar uygunluk değerlerine göre en iyiden en kötüye doğru sıralanır ve denklem (2.21) kullanılarak sıralamadaki yerlerine göre kromozomların olasılıkları (P_n) hesaplanır.

$$P_n = \frac{N_{iyi}-n+1}{\sum_{n=1}^{N_{iyi}} n} \qquad (2.21)$$

Burada n, sıra sayısını göstermektedir. Bu yöntemle uygunluk değerlerinden bağımsız olarak, sadece sıralamayı dikkate alan P_n normalize değerleri elde edilmektedir. Bu yöntemde kromozomların seçilmesinde P_n'lerin kümülatif toplamı (kümülatif ihtimaller) kullanılır. 0-1 arasında rastgele bir sayı üretilir ve bu sayı kümülatif ihtimallerin en başından başlanarak sırayla karşılaştırılır. Kümülatif ihtimal değeri, üretilen rastgele sayıdan büyükse, eşleştirme havuzu için kromozom seçilir. Burada, eğer kromozomlar kendisi ile eşleşirse, bu kromozomun gelecek nesilde daha etkin olarak yer aldığı söylenebilir.

2) Değer Ağırlık: Bu yöntemde, eşleştirme havuzunda N_{iyi+1}'inci sırada bulunan kromozomun amaç fonksiyonunda hesaplanan uygunluk değeri, bütün eşleştirme havuzundaki kromozomların uygunluk değerlerinden çıkartılır. Sonrasında eşleştirme havuzundaki her bir kromozomun normalize uygunluk değeri (C_n) ve seçilme ihtimali (P_n) denklem (2.22) ve denklem (2.23) ile hesaplanır.

$$C_n = f(kromozom_n) - f(kromozom_{N_{iyi+1}}) \tag{2.22}$$

$$P_n = \left| \frac{C_n}{\sum_{p=1}^{N_{iyi}} C_p} \right| \tag{2.23}$$

Hesaplanan bu değerlerden sonra kromozomların seçilmesinde yine P_n'lerin kümülatif toplamı kullanılır ve işlemin bundan sonraki kısmı sıralama ağırlıkta olduğu şekliyle aynen devam eder.

2.5.6. Çaprazlama

Çaprazlama popülasyondaki çeşitliliği arttırmak ve daha iyi nitelikli kromozomlar üretmek için kullanılan bir operatördür (Şahman, 2008). Çaprazlama işlemi için, gerçek kodlu genetik algoritmada tek noktalı çaprazlama, lineer çaprazlama, heuristic çaprazlama, karma çaprazlama, kuadratik çaprazlama gibi farklı yöntemler bulunmaktadır.

Gerçek kodlu genetik algoritmalarda en çok kullanılan çaprazlama metodu, heuristic çaprazlama metoduyla extrapolasyon çaprazlama metodunun kombinasyonundan oluşan bir metottur (Haupt ve Haupt, 1998).

Bu metotta ilk olarak denklem (2.24) ile rastgele bir çaprazlama noktası seçilir.

α = roundup {rasgele · N_{par}} (2.24)

Seçilen bu çaprazlama noktası denklem (2.25) ve denklem (2.26)'da α ile gösterilmiştir.

$$Kromozom_1 = [P_{a1}, P_{a2}, \dots, P_{a\alpha}, \dots, P_{aN_{par}}] \quad (2.25)$$

$$Kromozom_2 = [P_{b1}, P_{b2}, \dots, P_{b\alpha}, \dots, P_{bN_{par}}] \quad (2.26)$$

Burada a ve b indisleri sırasıyla anne ve baba kromozomlarını tanımlamak amacıyla kullanılmaktadır. Karşı karşıya getirilen ve rastgele seçilen bir noktasından çaprazlanacak anne ve baba kromozomlardaki değerler ile denklem (2.27) ve denklem (2.28) kullanılarak iki yeni değer elde edilir.

$$P_{yeni1} = P_{a\alpha} - \beta[P_{a\alpha} - P_{b\alpha}] \quad (2.27)$$

$$P_{yeni2} = P_{b\alpha} - \beta[P_{a\alpha} - P_{b\alpha}] \quad (2.28)$$

Burada β, 0-1 arasında rastgele üretilmiş bir değerdir. Metotta, elde edilen bu iki değer kromozomlarda yerlerine konulur ve son olarak; eğer, kromozomun en son parametresi için işlem yapılıyorsa, o parametrenin öncesindeki parametreler; eğer diğer parametrelerden birisi için işlem yapılıyorsa, o parametrenin sonrasındaki parametreler iki kromozomda karşılıklı olarak yer değiştirilerek işlem tamamlanır.

Üretilen yeni nesiller denklem (2.29) ve denklem (2.30)'daki gibi son halini almış olur.

$$Nesil_1 = [P_{a1}, P_{a2}, \dots, P_{yeni1}, \dots, P_{bN_{par}}] \quad (2.29)$$

$$Nesil_2 = [P_{b1}, P_{b2}, \dots, P_{yeni2}, \dots, P_{aN_{par}}] \quad (2.30)$$

2.5.7. Mutasyon

Genetik algoritmalar, bazen çok hızlı yakınsarlar. Yakınsama sonucunda global noktalar bulunursa sonuç iyi, lokal noktalar bulunursa beklenen sonuç elde edilememiş demektir. Hızlı yakınsamadan kurtulmanın yolu, araştırma uzayında, mutasyon aracılığıyla yeni çözümler elde etmektir. İkili kodlu genetik algoritmada mutasyon oranı

%1 - %5 arasında değiştirilerek iyi sonuçlar elde edilse de gerçek kodlu genetik algoritmada mutasyon oranı daha yüksektir. Parametrelerin toplam sayısı ile mutasyon oranı çarpılarak mutasyona girecek parametre sayısı tespit edilir. Bir matriste mutasyon, satır ve sütunu ifade edecek şekilde rastgele rakamlar üretilerek gerçekleştirilir. Mutasyona uğrayacak parametreler belirlendikten sonra eski parametreler silinir ve yerine rastgele sayılar üretilerek konulur.

3. DOĞRUSAL VE DOĞRUSAL OLMAYAN SİSTEMLERİN HİYERARŞİK BULANIK KONTROLÜ

Bu bölümde doğrusal ve doğrusal olmayan sistemlerin PSO ve GA optimizasyonlu bulanık mantık denetleyiciler ve hiyerarşik bulanık mantık denetleyiciler ile kontrolü gerçekleştirilmiştir. Kontrol simülasyonları için MATLAB Simulink benzetim programı kullanılmıştır. Çalışmada kontrolörlerin ayarlama parametrelerinden olan giriş ve çıkış ölçekleme çarpanları tüm sistemler için hem PSO hem de GA ile ayrı ayrı belirlenmiş ve karşılaştırma yapılmıştır.

Ayrıca parametrelerin aratılması sırasında amaç ölçütü olarak tüm sistemlerde sırasıyla denklem (3.1), denklem (3.2), denklem (3.3), denklem (3.4) ve denklem (3.5)'de verilen IAE (mutlak hata toplamı), ISE (karesel hata toplamı), ITAE (zaman ağırlıklı mutlak hata toplamı), ITSE (zaman ağırlıklı karesel hata toplamı) ve ISTSE (karesel zaman ağırlıklı karesel hata toplamı) ölçütleri kullanılmıştır.

$$IAE = \int |e(t)| \cdot dt \tag{3.1}$$

$$ISE = \int e(t)^2 \cdot dt \tag{3.2}$$

$$ITAE = \int t \cdot |e(t)| \cdot dt \tag{3.3}$$

$$ITSE = \int t \cdot e(t)^2 \cdot dt \tag{3.4}$$

$$ISTSE = \int t^2 \cdot e(t)^2 \cdot dt \tag{3.5}$$

3.1. Doğrusal Sistemlerin Bulanık Mantık Kontrolü

Bu çalışmada doğrusal sistem olarak denklem (3.6) ve denklem (3.7)'de verilen iki farklı sistem kullanılmıştır. Sistemlerden birisi dördüncü dereceden bir sistem, diğeri ise sağ yarı düzlemde sıfırı olan bir sistemdir.

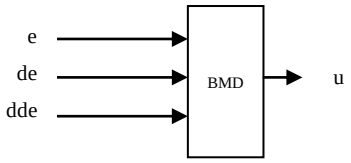
$$G(s) = \frac{1}{(s+1)^4} \tag{3.6}$$

$$G(s) = \frac{1-0{,}1s}{(s+1)^3} \tag{3.7}$$

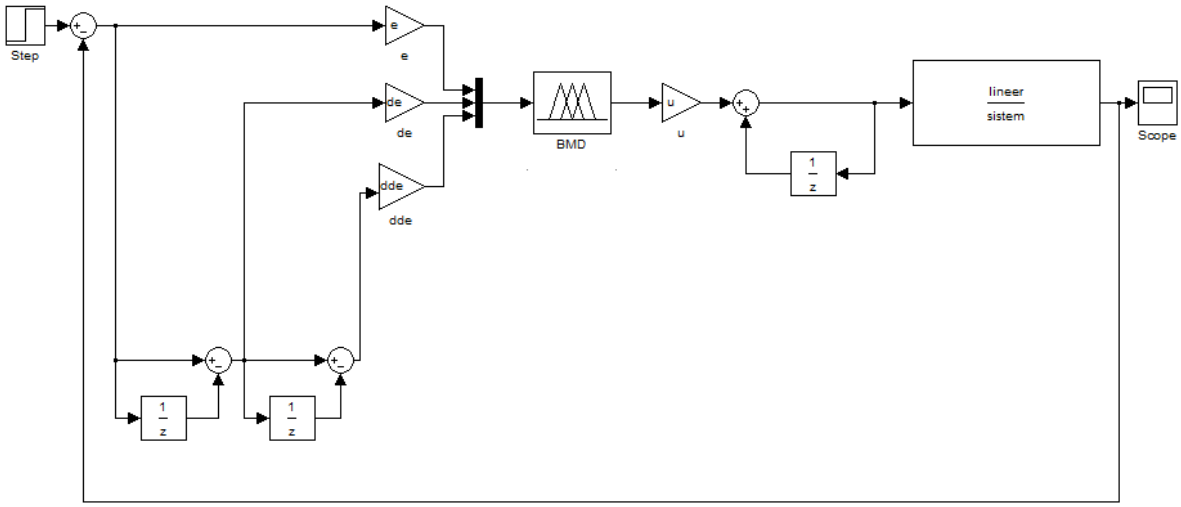
Bu iki sistemin kontrolünde, Şekil 3.1'deki gibi, 3 giriş ve 1 çıkışlı bulanık mantık denetleyiciler tasarlanmıştır. Denetleyici girişleri hata (e), hatanın türevi (de) ve

hatanın ikinci türevi (dde) olarak alınmıştır. Denetleyici çıkışı ise "u" olarak isimlendirilmiştir.

Çalışmada kullanılacak MATLAB Simulink devre modeli Şekil 3.2'deki gibidir.



Şekil 3.1. 3 giriş 1 çıkışlı bulanık mantık denetleyici



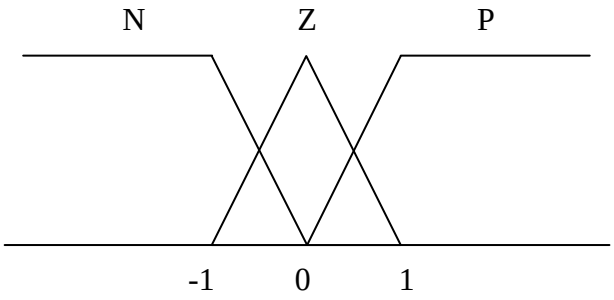
Şekil 3.2. Doğrusal sistemler için bulanık mantık denetleyicinin MATLAB Simulink modeli

Sistemlerin kontrolü için Çizelge 3.1'de gösterildiği gibi simetrik yapıda bir kural tablosu seçilmiştir.

Çizelge 3.1. Simetrik yapıda kural tablosu

dde	**e**								
	P			**Z**			**N**		
	de			**de**			**de**		
	P	**Z**	**N**	**P**	**Z**	**N**	**P**	**Z**	**N**
P	P	P	P	P	P	Z	P	Z	N
Z	P	P	Z	P	Z	N	Z	N	N
N	P	Z	N	Z	N	N	N	N	N

Ayrıca kontrolörde giriş ve çıkış değişkenleri için üyelik fonksiyonları Şekil 3.3'deki gibi oluşturulmuştur.

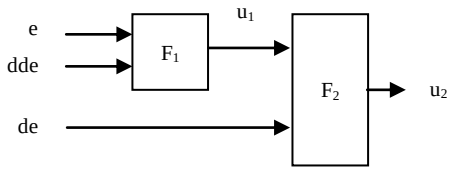


Şekil 3.3. BMD için giriş ve çıkış üyelik fonksiyonları

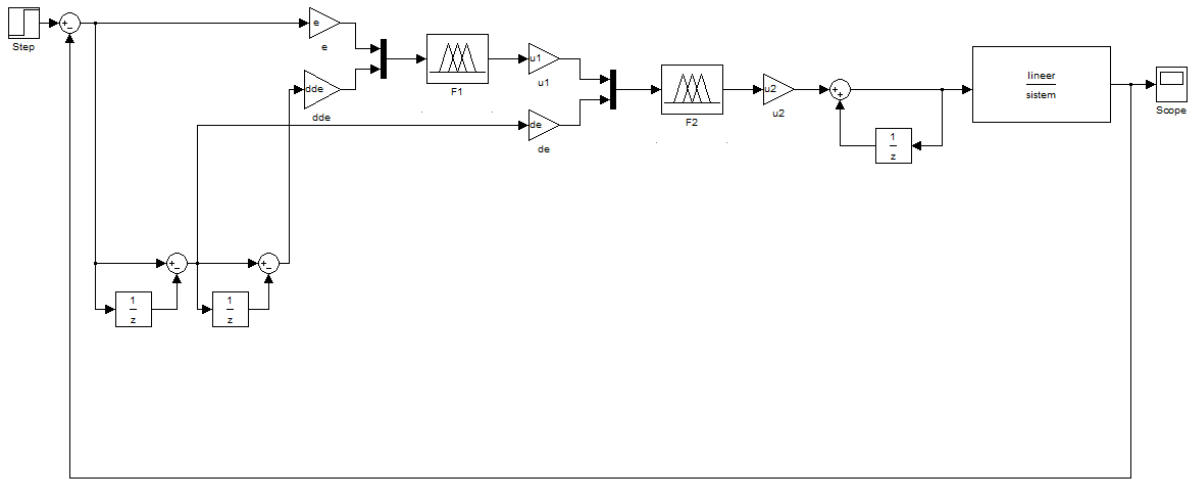
3.2. Doğrusal Sistemlerin Hiyerarşik Bulanık Mantık Kontrolü

Doğrusal sistem olarak denklem (3.6) ve denklem (3.7)'de verilen iki sistemin kontrolünde de, Şekil 3.4'deki gibi, 3 giriş ve 1 çıkışlı hiyerarşik bulanık mantık denetleyici tasarlanmıştır. Denetleyici girişleri hata (e), hatanın türevi (de) ve hatanın ikinci türevi (dde) olarak alınmıştır. Bu girişler hiyerarşik olarak; "e" ve "dde" birinci bloğa ve birinci bloğun çıkışı "u_1" ve "de" ikinci bloğa giriş olarak uygulanmıştır. İkinci bloğun çıkışı da "u_2" olarak isimlendirilmiştir.

Çalışmada kullanılacak MATLAB Simulink devre modeli Şekil 3.5'deki gibidir.



Şekil 3.4. Satır hiyerarşi yapısında HBMD



Şekil 3.5. Doğrusal sistemler için satır hiyerarşi yapısındaki MATLAB Simulink modeli

Sistemlerin kontrolü için Çizelge 3.2'de gösterildiği gibi simetrik yapıda bir kural tablosu seçilmiş ve bu tabloda satır gruplama yapılarak Çizelge 3.3 ve Çizelge 3.4'deki gibi satır hiyerarşi uygulanmış tablolar kullanılmıştır.

Çizelge 3.2. Satır hiyerarşi yapısında gruplanmış kural tablosu

dde	**e**								
	P			**Z**			**N**		
	de			**de**			**de**		
	P	**Z**	**N**	**P**	**Z**	**N**	**P**	**Z**	**N**
P	A			B			C		
	P	P	P	P	P	Z	P	Z	N
Z	B			C			D		
	P	P	Z	P	Z	N	Z	N	N
N	C			D			E		
	P	Z	N	Z	N	N	N	N	N

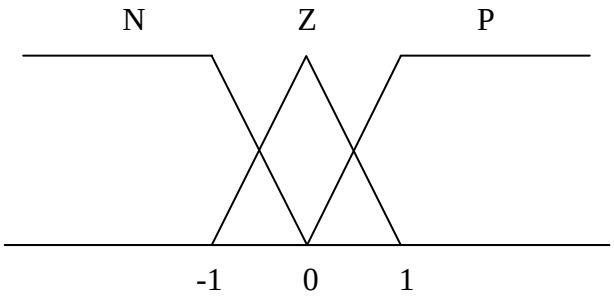
Çizelge 3.3. F_1 bloğu için kural tablosu

dde	**e**		
	P	**Z**	**N**
P	A	B	C
Z	B	C	D
N	C	D	E

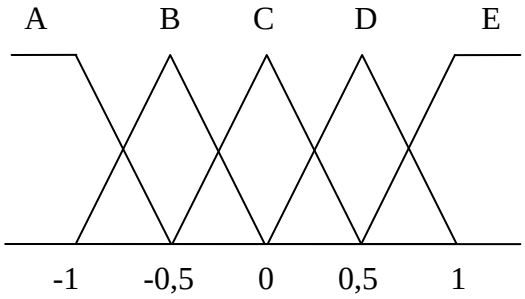
Çizelge 3.4. F_2 bloğu için kural tablosu

u_1	**de**		
	P	**Z**	**N**
A	P	P	P
B	P	P	Z
C	P	Z	N
D	Z	N	N
E	N	N	N

Ayrıca kontrolörde giriş, ara ve çıkış değişkenleri için üyelik fonksiyonları Şekil 3.6 ve Şekil 3.7'deki gibi oluşturulmuştur.



Şekil 3.6. Satır hiyerarşi yapısındaki giriş ve çıkış üyelik fonksiyonları



Şekil 3.7. Satır hiyerarşi yapısındaki ara üyelik fonksiyonları

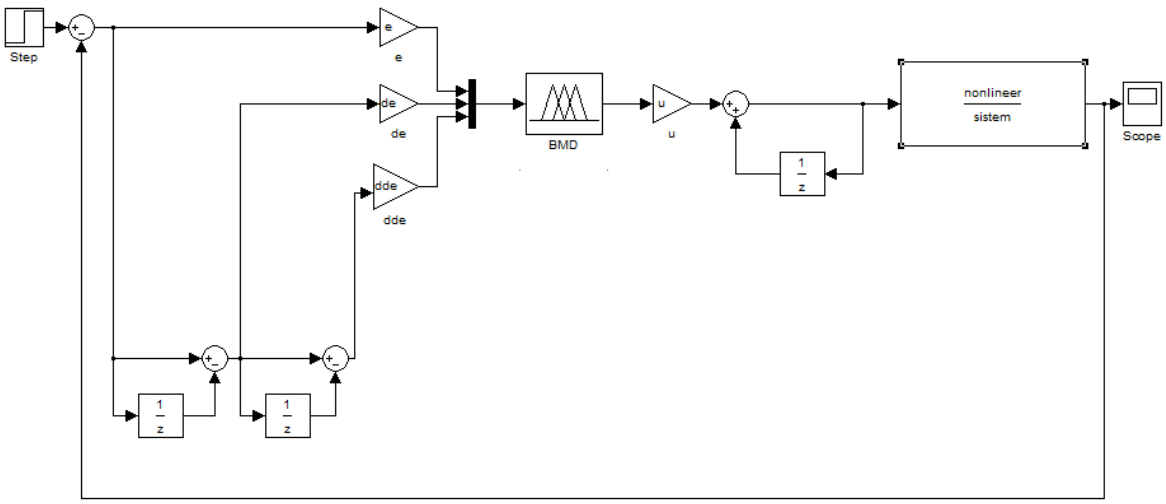
3.3. Doğrusal Olmayan Sistemlerin Bulanık Mantık Kontrolü

Yapılan çalışmada doğrusal olmayan sistem olarak denklem (3.8)'de verilen sistem kullanılmıştır.

$$\dot{y} = -0{,}2 \cdot [Sin(y) + Cos(y)] - \frac{y}{(1+y^2)} + [0{,}4 \cdot Sin(y) \cdot Cos(y^2)] + 0{,}8 \cdot u \qquad (3.8)$$

Sistemin kontrolünde Şekil 3.1'deki gibi, 3 giriş ve 1 çıkışlı bir bulanık mantık denetleyici tasarlanmıştır. Denetleyici girişleri hata (e), hatanın türevi (de) ve hatanın ikinci türevi (dde) olarak alınmıştır. Denetleyici çıkışı ise "u" olarak isimlendirilmiştir.

Çalışmada kullanılacak MATLAB Simulink devre modeli Şekil 3.8'deki gibidir.



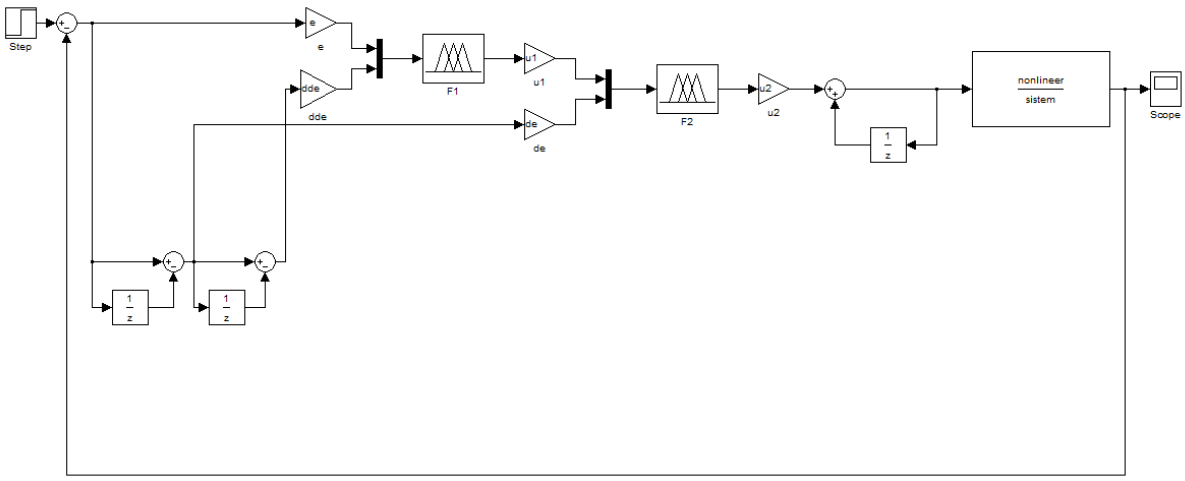
Şekil 3.8. Doğrusal olmayan sistemler için bulanık mantık denetleyicinin MATLAB Simulink modeli

Sistemlerin kontrolü için yine Çizelge 3.1'de gösterildiği gibi simetrik yapıda bir kural tablosu seçilmiştir. Ayrıca kontrolörde giriş ve çıkış değişkenleri için üyelik fonksiyonları Şekil 3.3'deki gibi oluşturulmuştur.

3.4. Doğrusal Olmayan Sistemlerin Hiyerarşik Bulanık Mantık Kontrolü

Doğrusal olmayan sistem olarak denklem (3.8)'de verilen sistemin kontrolünde, Şekil 3.4'deki gibi, 3 giriş ve 1 çıkışlı bir hiyerarşik bulanık mantık denetleyici tasarlanmıştır. Denetleyici girişleri hata (e), hatanın türevi (de) ve hatanın ikinci türevi (dde) olarak alınmıştır. Bu girişler hiyerarşik olarak; "e" ve "dde" birinci bloğa ve birinci bloğun çıkışı "u_1" ve "de" ikinci bloğa giriş olarak uygulanmıştır. İkinci bloğun çıkışı da "u_2" olarak isimlendirilmiştir.

Çalışmada kullanılacak MATLAB Simulink devre modeli Şekil 3.9'daki gibidir.



Şekil 3.9. Doğrusal olmayan sistemler için satır hiyerarşi yapısındaki MATLAB Simulink modeli

Bu sistemin kontrolü için de, doğrusal sistemlerin kontrolü için kullandığımız, satır gruplama yapılarak satır hiyerarşi uygulanan Çizelge 3.2'deki simetrik yapıdaki kural tablosu seçilmiş ve gruplama sonucu elde edilen ve Çizelge 3.3 ve Çizelge 3.4'de verilen tablolar kullanılmıştır.

Giriş, ara ve çıkış üyelik fonksiyonları, yine, doğrusal sistem için kullanılan ve Şekil 3.6 ve Şekil 3.7'de verilen üyelik fonksiyonlarıyla aynıdır.

4. ARAŞTIRMA BULGULARI VE TARTIŞMA

Bu bölümde, denklem (3.6) ve denklem (3.7) ile verilen doğrusal iki farklı sistem ile denklem (3.8)'de verilen doğrusal olmayan sistemin, PSO ve GA optimizasyonlu bulanık ve hiyerarşik bulanık denetleyiciler ile kontrolünün, MATLAB-Simulink ile gerçekleştirilen simülasyon çıkış grafiklerine, performans ölçüt değerlerine ve bunların karşılaştırmalarına yer verilmiştir.

Gerçekleştirilen simülasyonlarda örnekleme zamanı 0.2 sn seçilmiştir. Tüm denemelerde "e", "de", "dde", "u", "u_1" ve "u_2" için ölçeklendirme katsayıları hem PSO ile hem de gerçek kodlu GA ile [0.1 4] aralığında aratılmıştır. Çalışmada PSO için c_1 ve c_2 katsayıları c_1=c_2=1 olarak alınmıştır. Parçacık sayısı ve iterasyon sayısı, sırasıyla, 200-200, 1000-50 ve 500-100 olarak seçilmiş, gerçekleştirilen simülasyonlarda elde edilen değerler, en iyi değerler koyulaştırılmış şekilde, Çizelge 4.1, Çizelge 4.2 ve Çizelge 4.5'deki sistem yanıtları tablolarına işlenmiştir. Ayrıca PSO'da atalet ağırlığı kullanılmış ve w_{min}=0.4 ve w_{max}=0.9 olarak dikkate alınmıştır. GA için ise, bir gerçek kodlu GA tercih edilerek popülasyon sayısı ve iterasyon sayısı, sırasıyla, 200-200, 1000-50 ve 500-100 olarak seçilmiş, mutasyon oranı 0.3 olarak alınmış, gerçekleştirilen simülasyonlarda elde edilen değerler, en iyi değerler koyulaştırılmış şekilde, Çizelge 4.3, Çizelge 4.4 ve Çizelge 4.6'daki sistem yanıtları tablolarına işlenmiştir.

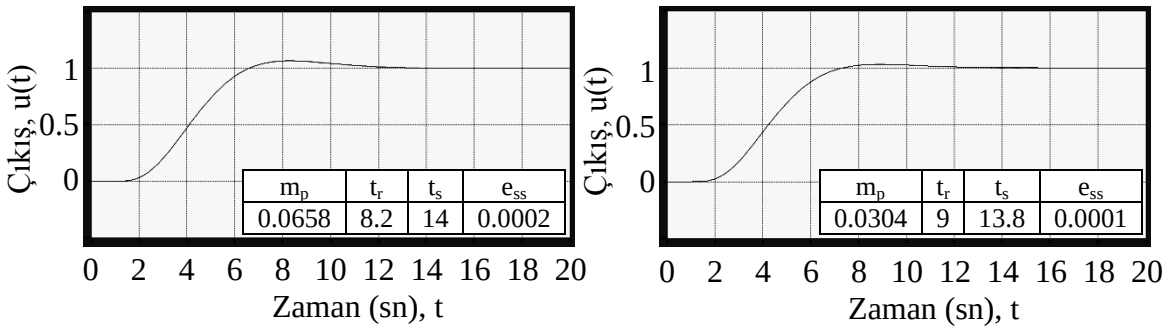
4.1. Doğrusal Sistemlerde PSO Optimizasyonlu Simülasyon Sonuçları

Denklem (3.6) ile verilen $\frac{1}{(s+1)^4}$ doğrusal sistemi için PSO kullanılarak gerçekleştirilen simülasyonlarda sistem yanıtları Çizelge 4.1'de gösterildiği gibi elde edilmiştir.

Çizelge 4.1. $\frac{1}{(s+1)^4}$ doğrusal sistemi için PSO kullanılarak gerçekleştirilen simülasyon cevapları

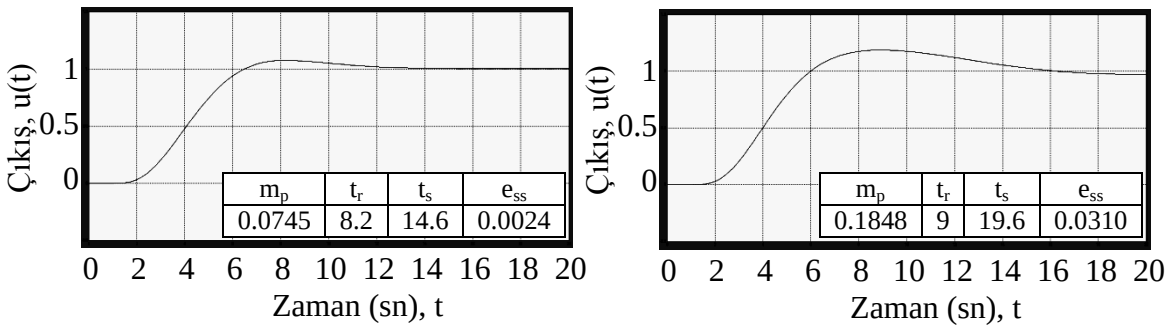
	Bulanık Mantık Denetleyici			**Hiyerarşik Bulanık Mantık Denetleyici**		
	Parçacık Sayısı / İterasyon Sayısı			**Parçacık Sayısı / İterasyon Sayısı**		
	200/200	**1000/50**	**500/100**	**200/200**	**1000/50**	**500/100**
IAE	17.362332	17.362405	19.083142	17.624694	20.763467	20.74905
ISE	12.598768	12.598768	12.602635	13.248406	13.190387	13.186646
ITAE	61.195154	60.531007	54.787947	106.278394	55.809667	57.721104
ITSE	30.320256	30.320256	30.272145	32.441935	32.33673	32.441935
ISTSE	87.779864	87.780164	274.00178	94.782917	125.249008	125.248916

Çizelge 4.1'deki tabloda koyulaştırılmış şekilde gösterilen en iyi değerler için çıkış cevapları ve cevabın aşım (m_p), tepe zamanı (t_r), yerleşme zamanı (t_s) ve kararlı hal hatası (e_{ss}) değerleri Şekil 4.1 ile Şekil 4.5 arasında verilmiştir.



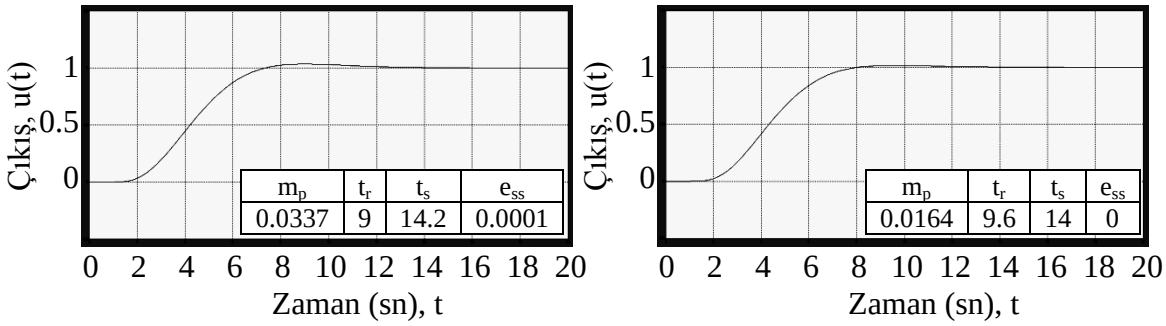
(a) (b)

Şekil 4.1. Çizelge 4.1'de verilen IAE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



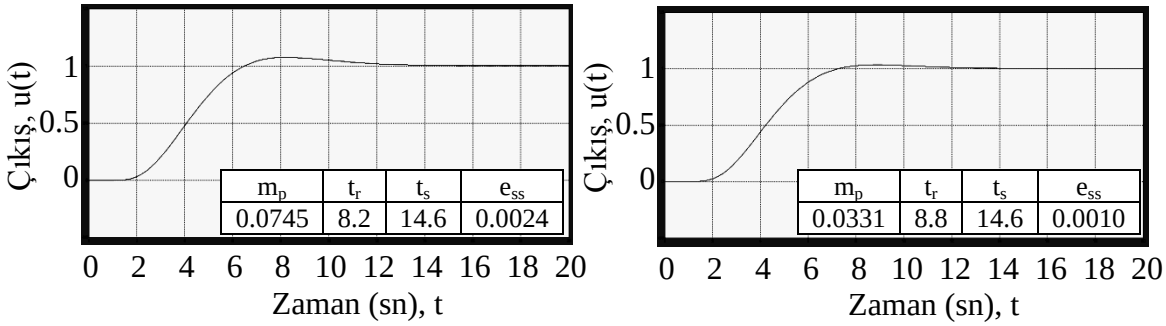
(a) (b)

Şekil 4.2. Çizelge 4.1'de verilen ISE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



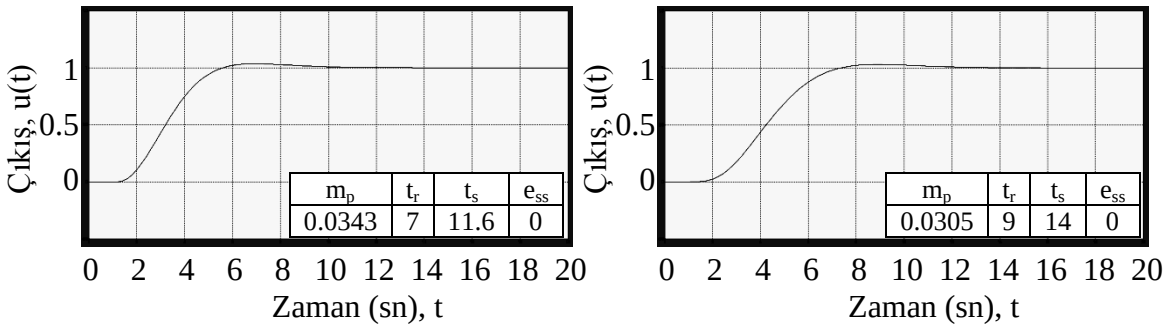
(a) (b)

Şekil 4.3. Çizelge 4.1'de verilen ITAE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



(a) (b)

Şekil 4.4. Çizelge 4.1'de verilen ITSE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



(a) (b)

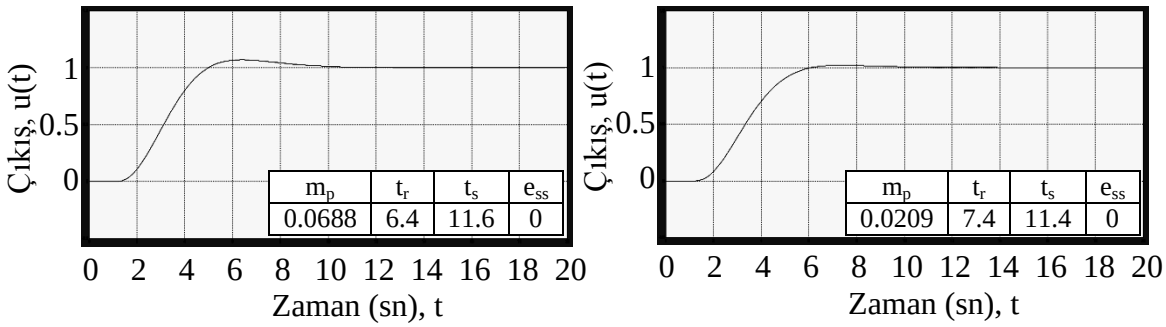
Şekil 4.5. Çizelge 4.1'de verilen ISTSE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali

Denklem (3.7) ile verilen $\frac{1-0,1s}{(s+1)^3}$ doğrusal sistemi için PSO kullanılarak gerçekleştirilen simülasyonlarda sistem yanıtları Çizelge 4.2'de gösterildiği gibi elde edilmiştir.

Çizelge 4.2. $\frac{1-0,1s}{(s+1)^3}$ doğrusal sistemi için PSO kullanılarak gerçekleştirilen simülasyon cevapları

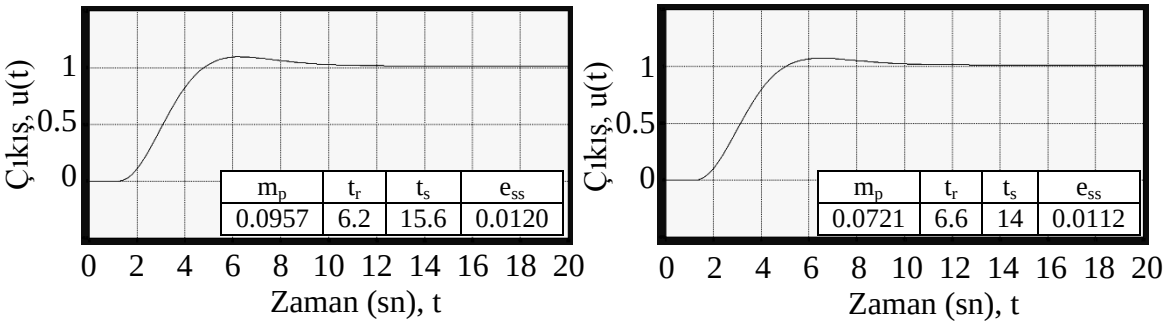
	Bulanık Mantık Denetleyici			**Hiyerarşik Bulanık Mantık Denetleyici**		
	Parçacık Sayısı / İterasyon Sayısı			**Parçacık Sayısı / İterasyon Sayısı**		
	200/200	**1000/50**	**500/100**	**200/200**	**1000/50**	**500/100**
IAE	12.356316	19.424049	12.356509	13.490652	13.001485	13.001366
ISE	8.757098	8.740623	8.740622	9.54238	8.906679	9.573408
ITAE	35.665318	68.421238	31.555076	34.164672	32.999266	32.996354
ITSE	17.083826	17.083827	17.083826	19.317528	16.668104	20.894138
ISTSE	38.972694	113.516464	39.014554	45.327937	46.350223	46.320878

Çizelge 4.2'deki tabloda koyulaştırılmış şekilde gösterilen en iyi değerler için çıkış cevapları ve cevabın aşım (m_p), tepe zamanı (t_r), yerleşme zamanı (t_s) ve kararlı hal hatası (e_{ss}) değerleri Şekil 4.6 ile Şekil 4.10 arasında verilmiştir.



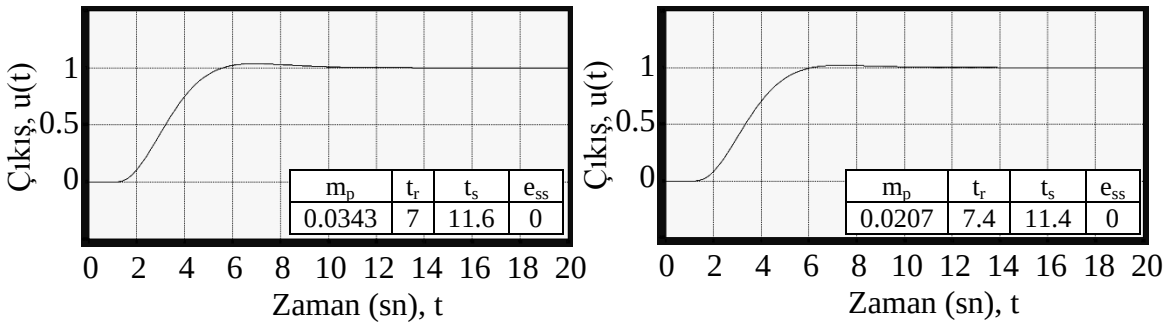
(a) (b)

Şekil 4.6. Çizelge 4.2'de verilen IAE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



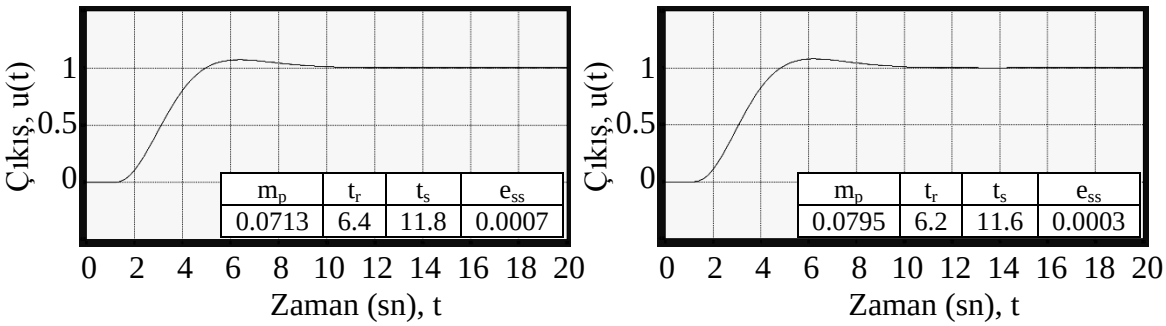
(a) (b)

Şekil 4.7. Çizelge 4.2'de verilen ISE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



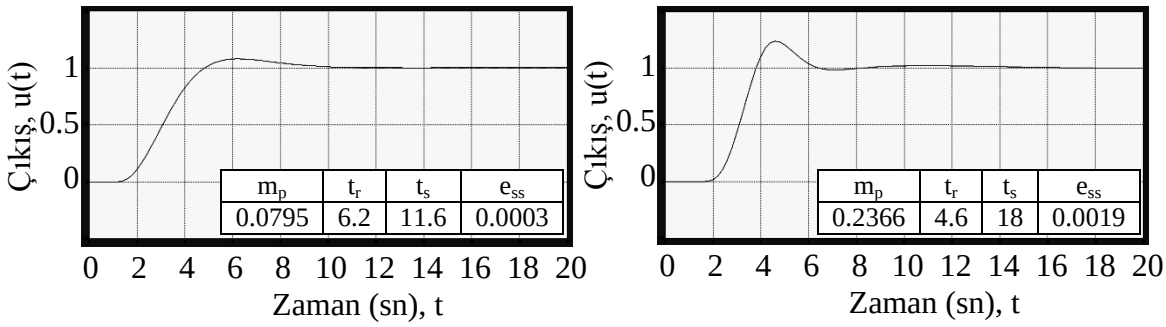
(a) (b)

Şekil 4.8. Çizelge 4.2'de verilen ITAE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



(a) (b)

Şekil 4.9. Çizelge 4.2'de verilen ITSE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



(a) (b)

Şekil 4.10. Çizelge 4.2'de verilen ISTSE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali

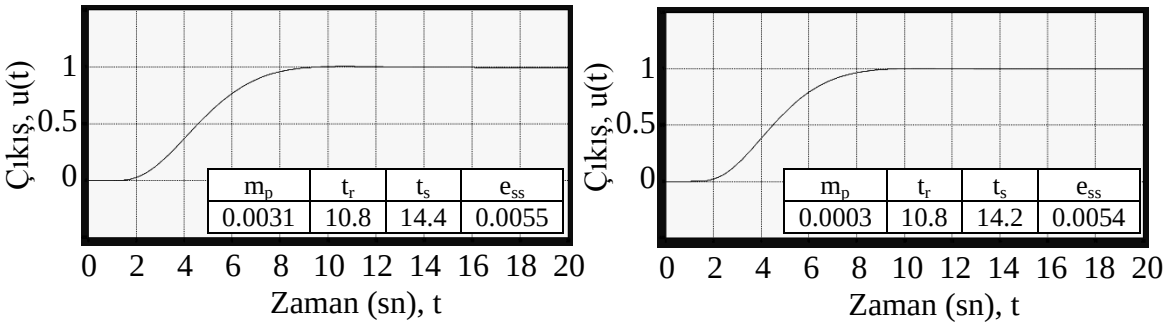
4.2. Doğrusal Sistemlerde GA Optimizasyonlu Simülasyon Sonuçları

Denklem (3.6) ile verilen $\frac{1}{(s+1)^4}$ doğrusal sistemi için GA kullanılarak gerçekleştirilen simülasyonlarda sistem yanıtları Çizelge 4.3'de gösterildiği gibi elde edilmiştir.

Çizelge 4.3. $\frac{1}{(s+1)^4}$ doğrusal sistemi için GA kullanılarak gerçekleştirilen simülasyon cevapları

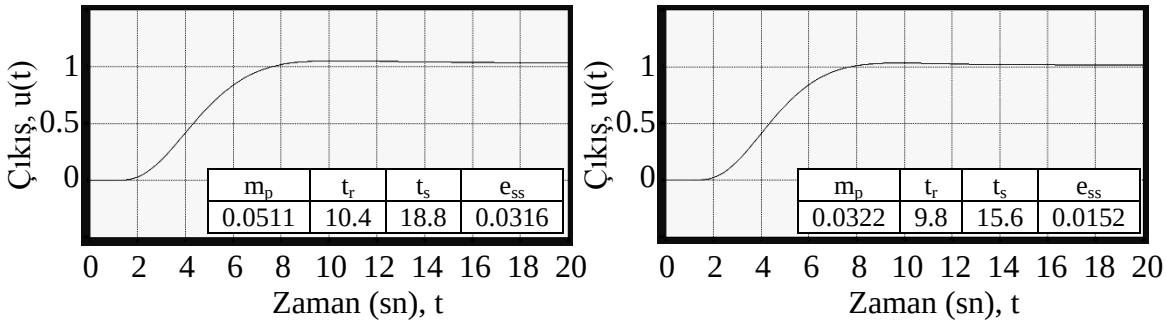
	Bulanık Mantık Denetleyici			**Hiyerarşik Bulanık Mantık Denetleyici**		
	Popülasyon Sayısı / İterasyon Sayısı			**Popülasyon Sayısı / İterasyon Sayısı**		
	200/200	**1000/50**	**500/100**	**200/200**	**1000/50**	**500/100**
IAE	21.021068	20.158756	21.058352	20.660441	20.305412	21.831597
ISE	14.101726	13.690123	13.740164	13.865713	13.798161	13.956384
ITAE	94.960208	78.170721	95.978434	76.392085	63.101010	77.976474
ITSE	35.792362	37.334463	39.285165	38.964366	35.701499	37.159560
ISTSE	175.450677	132.982118	134.892685	116.650891	133.672502	133.258168

Çizelge 4.3'deki tabloda koyulaştırılmış şekilde gösterilen en iyi değerler için çıkış cevapları ve cevabın aşım (m_p), tepe zamanı (t_r), yerleşme zamanı (t_s) ve kararlı hal hatası (e_{ss}) değerleri Şekil 4.11 ile Şekil 4.15 arasında verilmiştir.



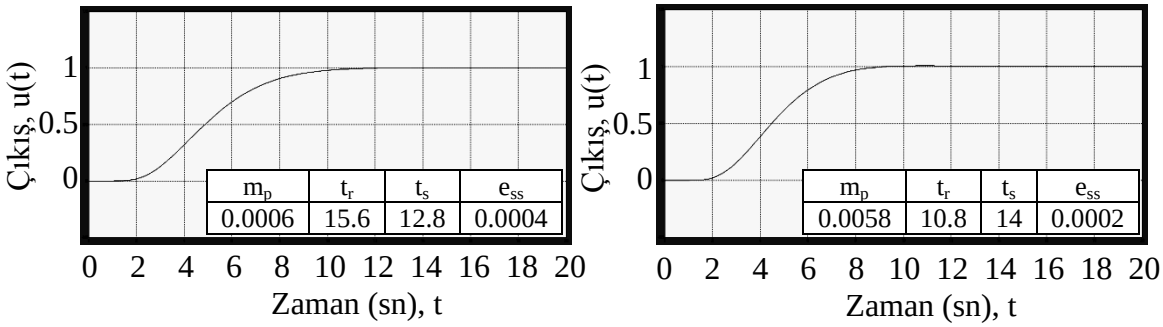
(a) (b)

Şekil 4.11. Çizelge 4.3'de verilen IAE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



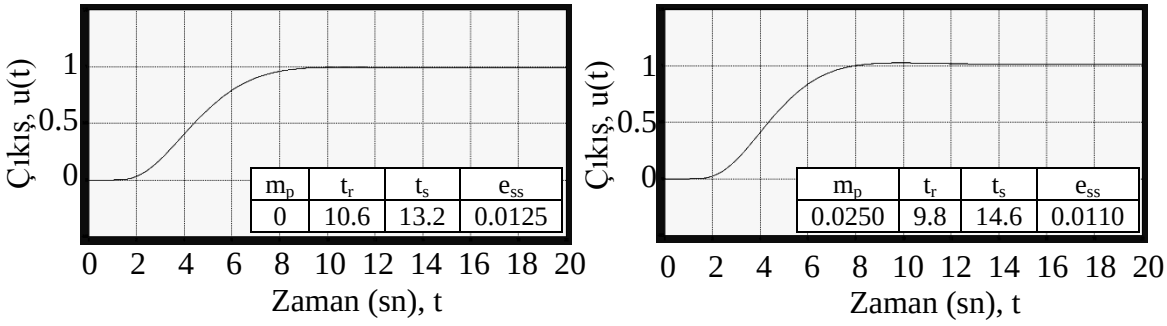
(a) (b)

Şekil 4.12. Çizelge 4.3'de verilen ISE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



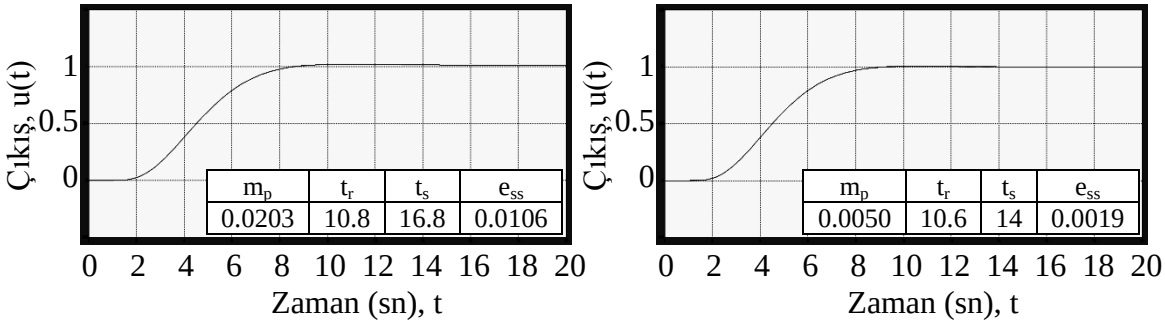
(a) (b)

Şekil 4.13. Çizelge 4.3'de verilen ITAE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



(a) (b)

Şekil 4.14. Çizelge 4.3'de verilen ITSE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



(a) (b)

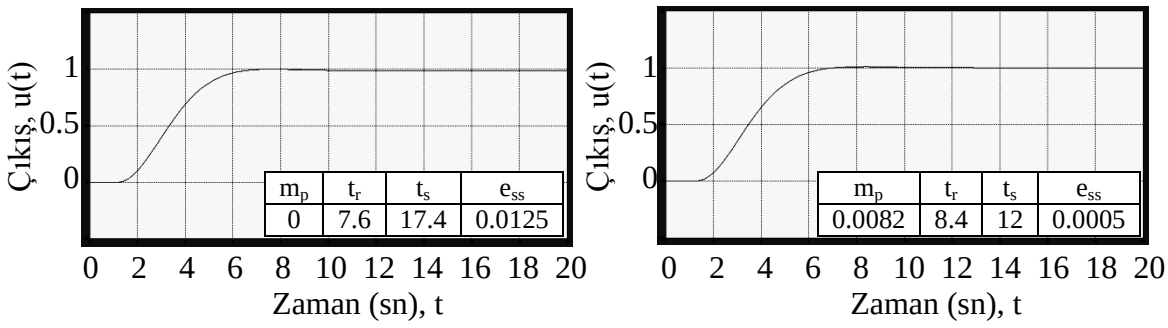
Şekil 4.15. Çizelge 4.3'de verilen ISTSE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali

Denklem (3.7) ile verilen $\frac{1-0,1s}{(s+1)^3}$ doğrusal sistemi için GA kullanılarak gerçekleştirilen simülasyonlarda sistem yanıtları Çizelge 4.4'de gösterildiği gibi elde edilmiştir.

Çizelge 4.4. $\frac{1-0,1s}{(s+1)^3}$ doğrusal sitemi için GA kullanılarak gerçekleştirilen simülasyon cevapları

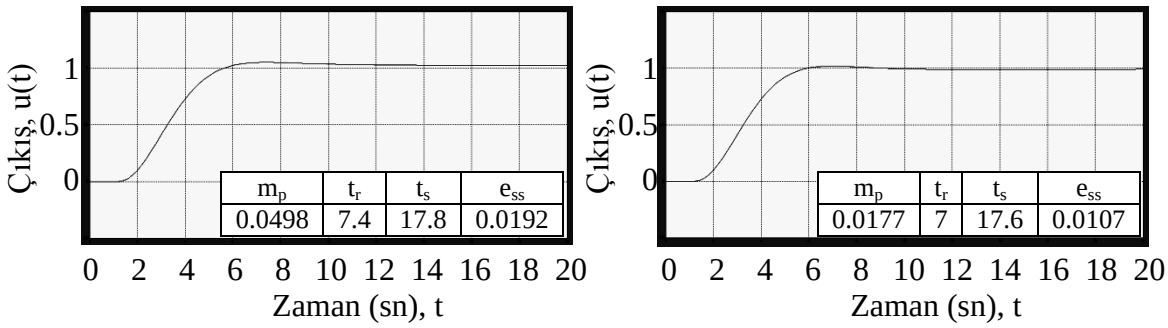
	Bulanık Mantık Denetleyici			**Hiyerarşik Bulanık Mantık Denetleyici**		
	Popülasyon Sayısı / İterasyon Sayısı			**Popülasyon Sayısı / İterasyon Sayısı**		
	200/200	**1000/50**	**500/100**	**200/200**	**1000/50**	**500/100**
IAE	16.475043	16.310962	15.069761	15.694629	13.832827	15.433519
ISE	9.328499	9.574037	9.876884	9.287697	9.386805	9.672743
ITAE	66.040261	45.707279	46.968623	56.517814	55.583106	58.401898
ITSE	24.657732	22.765624	22.637433	22.724447	20.892780	21.872622
ISTSE	72.562605	46.997735	71.385314	69.600203	66.757752	62.329000

Çizelge 4.4'deki tabloda koyulaştırılmış şekilde gösterilen en iyi değerler için çıkış cevapları ve cevabın aşım (m_p), tepe zamanı (t_r), yerleşme zamanı (t_s) ve kararlı hal hatası (e_{ss}) değerleri Şekil 4.16 ile Şekil 4.20 arasında verilmiştir.



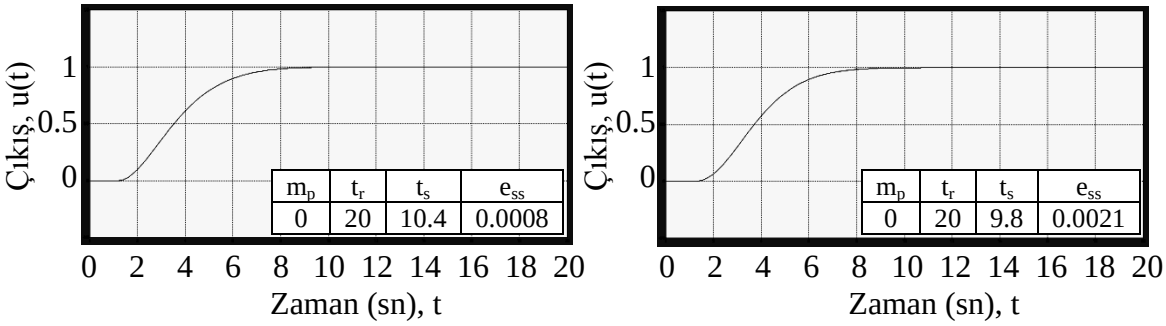
(a) (b)

Şekil 4.16. Çizelge 4.4'de verilen IAE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



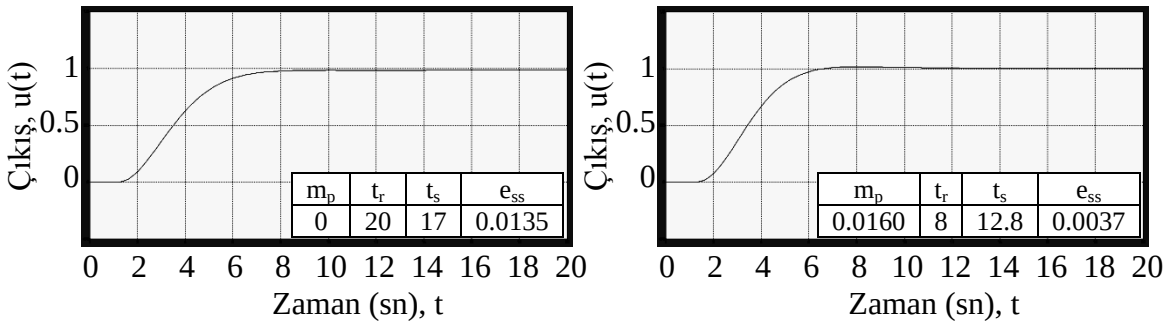
(a) (b)

Şekil 4.17. Çizelge 4.4'de verilen ISE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



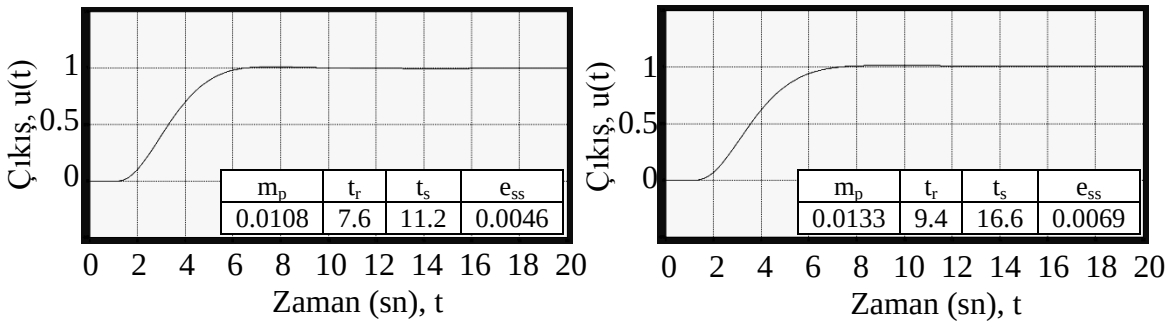
(a) (b)

Şekil 4.18. Çizelge 4.4'de verilen ITAE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



(a) (b)

Şekil 4.19. Çizelge 4.4'de verilen ITSE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



(a) (b)

Şekil 4.20. Çizelge 4.4'de verilen ISTSE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali

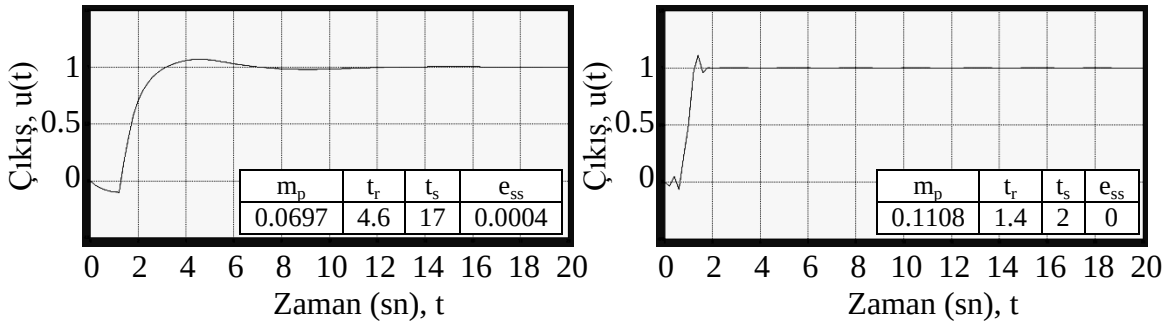
4.3. Doğrusal Olmayan Sistemlerde PSO Optimizasyonlu Simülasyon Sonuçları

Denklem (3.8) ile verilen $\dot{y} = -0{,}2 \cdot [Sin(y) + Cos(y)] - \frac{y}{(1+y^2)} + [0{,}4 \cdot Sin(y) \cdot Cos(y^2)] + 0{,}8 \cdot u$ doğrusal olmayan sistem için PSO kullanılarak gerçekleştirilen simülasyonlarda sistem yanıtları Çizelge 4.5'de gösterildiği gibi elde edilmiştir.

Çizelge 4.5. $\dot{y} = -0{,}2 \cdot [Sin(y) + Cos(y)] - \frac{y}{(1+y^2)} + [0{,}4 \cdot Sin(y) \cdot Cos(y^2)] + 0{,}8 \cdot u$ doğrusal olmayan sistem için PSO kullanılarak gerçekleştirilen simülasyon cevapları

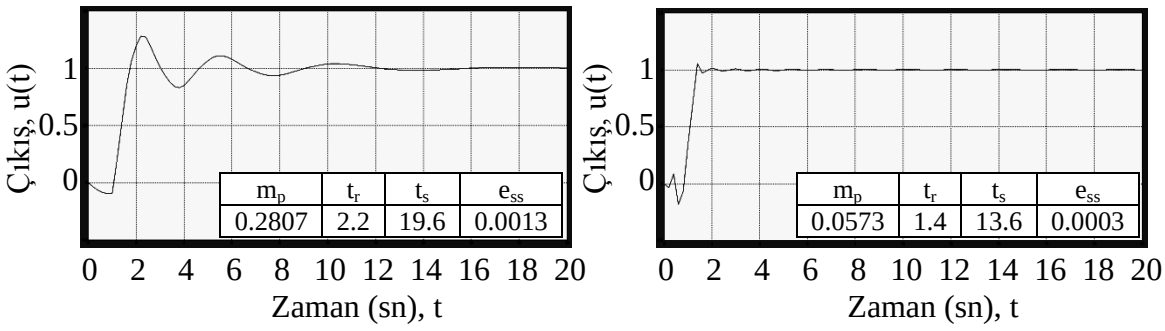
	Bulanık Mantık Denetleyici			**Hiyerarşik Bulanık Mantık Denetleyici**		
	Parçacık Sayısı / İterasyon Sayısı			**Parçacık Sayısı / İterasyon Sayısı**		
	200/200	**1000/50**	**500/100**	**200/200**	**1000/50**	**500/100**
IAE	6.331656	6.331656	6.331656	1.809194	1.690783	1.102745
ISE	2.545969	2.54597	2.545969	0.578491	1.36457	1.393379
ITAE	14.496507	14.496507	14.496507	2.161403	1.448733	2.167748
ITSE	3.984379	3.984381	3.984379	0.689214	0.620933	0.714316
ISTSE	8.049387	8.049387	8.049387	0.651553	2.091084	0.380746

Çizelge 4.5'deki tabloda koyulaştırılmış şekilde gösterilen en iyi değerler için çıkış cevapları ve cevabın aşım (m_p), tepe zamanı (t_r), yerleşme zamanı (t_s) ve kararlı hal hatası (e_{ss}) değerleri Şekil 4.21 ile Şekil 4.25 arasında verilmiştir.



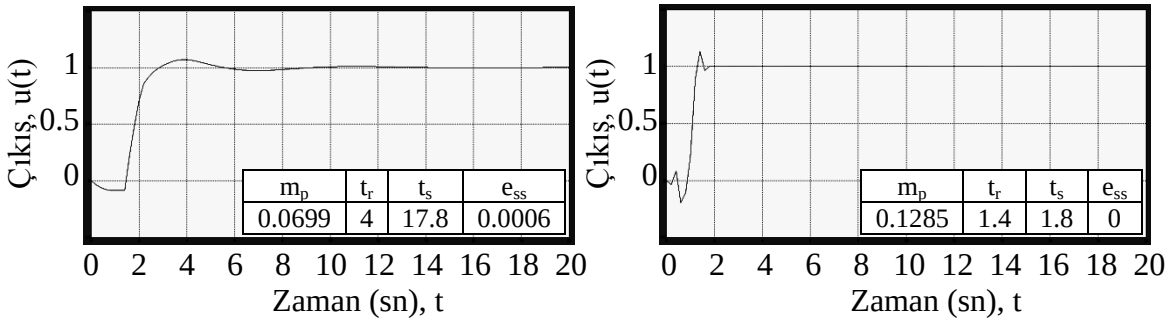
(a) (b)

Şekil 4.21. Çizelge 4.5'de verilen IAE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



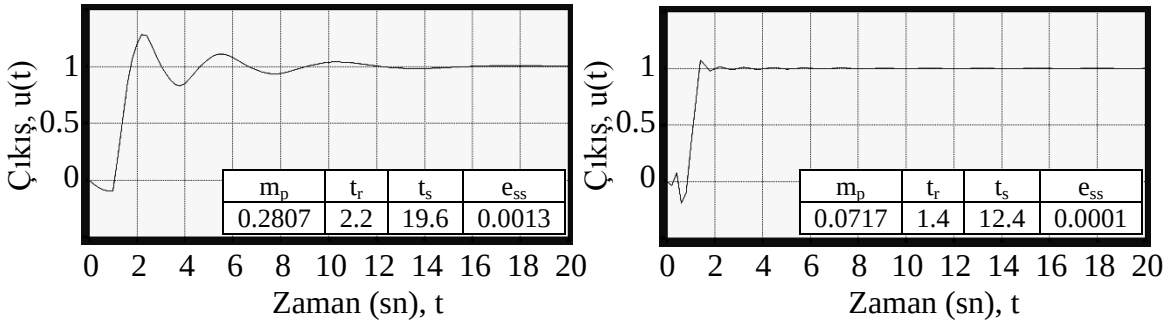
(a) (b)

Şekil 4.22. Çizelge 4.5'de verilen ISE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



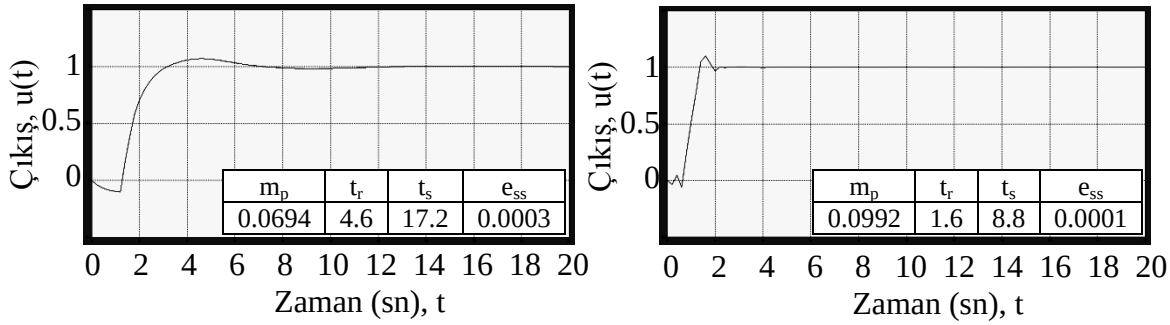
(a) (b)

Şekil 4.23. Çizelge 4.5'de verilen ITAE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



(a) (b)

Şekil 4.24. Çizelge 4.5'de verilen ITSE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



(a) (b)

Şekil 4.25. Çizelge 4.5'de verilen ISTSE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali

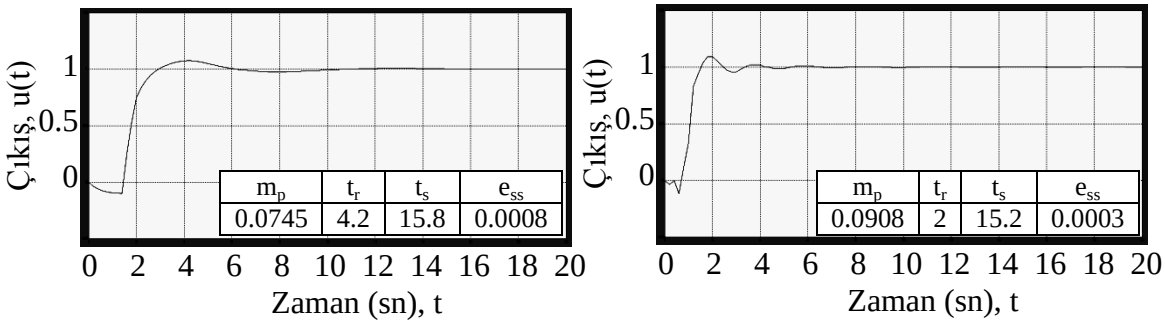
4.4. Doğrusal Olmayan Sistemlerde GA Optimizasyonlu Simülasyon Sonuçları

Denklem (3.8) ile verilen $\dot{y} = -0{,}2 \cdot [Sin(y) + Cos(y)] - \frac{y}{(1+y^2)} + [0{,}4 \cdot Sin(y) \cdot Cos(y^2)] + 0{,}8 \cdot u$ doğrusal olmayan sistem için GA kullanılarak gerçekleştirilen simülasyonlarda sistem yanıtları Çizelge 4.6'da gösterildiği gibi elde edilmiştir.

Çizelge 4.6. $\dot{y} = -0{,}2 \cdot [Sin(y) + Cos(y)] - \frac{y}{(1+y^2)} + [0{,}4 \cdot Sin(y) \cdot Cos(y^2)] + 0{,}8 \cdot u$ doğrusal olmayan sistem için GA kullanılarak gerçekleştirilen simülasyon cevapları

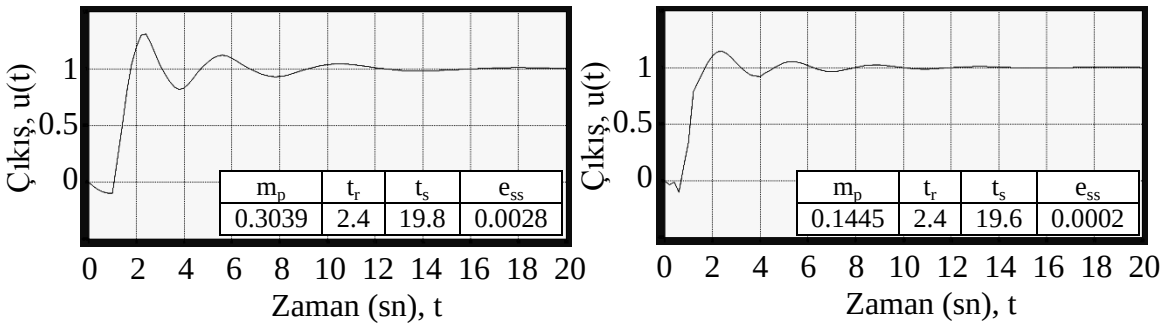
	Bulanık Mantık Denetleyici			**Hiyerarşik Bulanık Mantık Denetleyici**		
	Popülasyon Sayısı / İterasyon Sayısı			**Popülasyon Sayısı / İterasyon Sayısı**		
	200/200	**1000/50**	**500/100**	**200/200**	**1000/50**	**500/100**
IAE	6.607659	6.554135	6.642880	2.563499	2.975795	1.865765
ISE	2.703925	2.790108	2.848789	0.716484	0.675222	0.650287
ITAE	15.866980	15.726099	15.728225	3.672688	5.324780	2.979898
ITSE	4.877539	4.365466	4.972553	0.932823	0.912039	1.196070
ISTSE	9.210110	9.285063	9.342303	1.093718	1.913883	1.324797

Çizelge 4.6'daki tabloda koyulaştırılmış şekilde gösterilen en iyi değerler için çıkış cevapları ve cevabın aşım (m_p), tepe zamanı (t_r), yerleşme zamanı (t_s) ve kararlı hal hatası (e_{ss}) değerleri Şekil 4.26 ile Şekil 4.30 arasında verilmiştir.



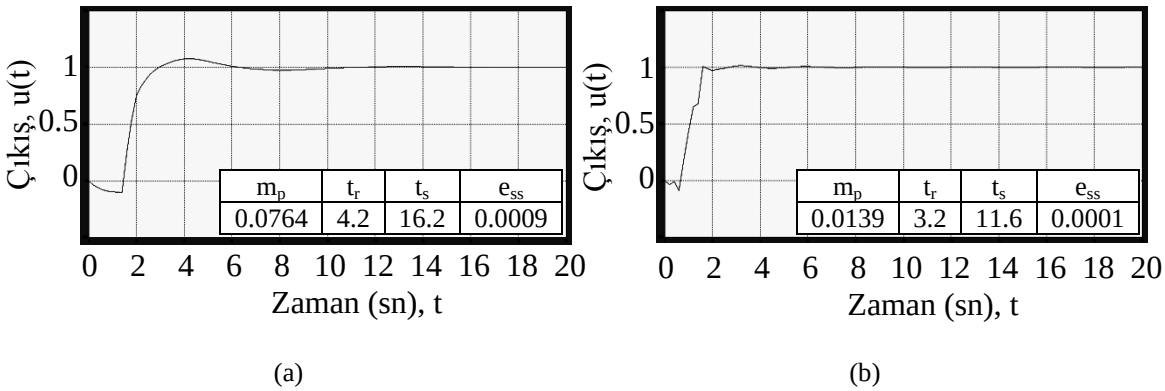
(a) (b)

Şekil 4.26. Çizelge 4.6'da verilen IAE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



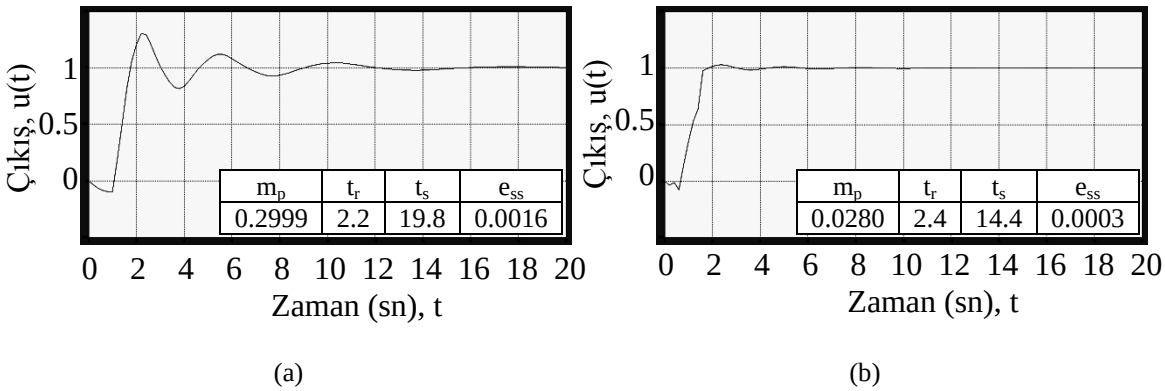
(a) (b)

Şekil 4.27. Çizelge 4.6'da verilen ISE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



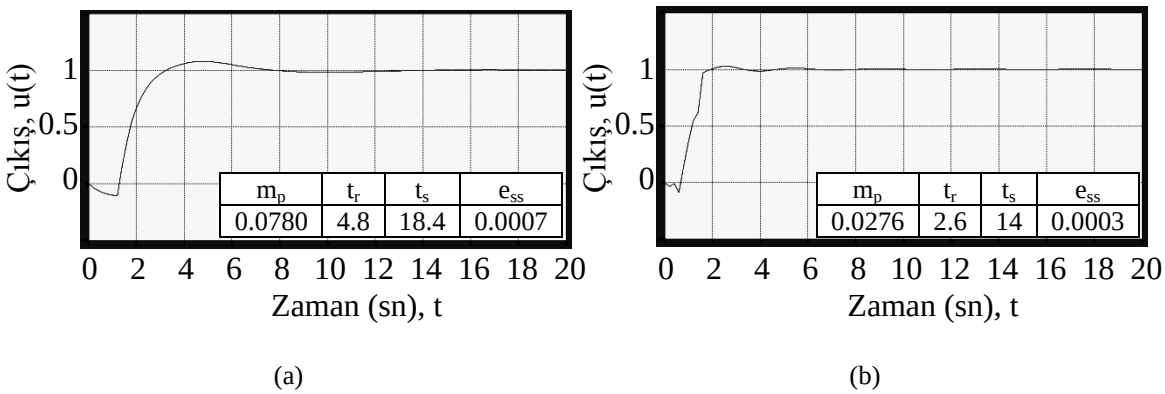
(a) (b)

Şekil 4.28. Çizelge 4.6'da verilen ITAE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



(a) (b)

Şekil 4.29. Çizelge 4.6'da verilen ITSE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali



(a) (b)

Şekil 4.30. Çizelge 4.6'da verilen ISTSE için çıkış sinyalleri, (a) BMD için çıkış sinyali, (b) HBMD için çıkış sinyali

4.5. Tartışma

Yapılan simülasyon çalışmaları neticesinde elde edilen en iyi sonuçlar Çizelge 4.7, Çizelge 4.8 ve Çizelge 4.9'da gösterilmiştir. Çizelge 4.9 incelendiğinde, PSO optimizasyonlu sonuçlarda ITAE amaç ölçüt kriterine göre, BMD için 14.496507 olarak ölçülen sonuç, HBMD için 1.448733 olarak ölçülmüş, benzer şekilde, GA

optimizasyonlu sonuçlarda ISTSE amaç ölçüt kriterine göre, BMD için 9.210110 olarak ölçülen sonuç, HBMD için 1.093718 olarak ölçülmüş ve kontrol yöntemleri içerisinde hiyerarşik bulanık mantık denetleyicilerin doğrusal olmayan sistemlerde daha etkili olduğu gözlemlenmiştir.

Çizelge 4.7 incelendiğinde, HBMD için PSO optimizasyonunda IAE için elde edilen değer 17.624694 iken GA optimizasyonunda elde edilen değer 20.305412; yine PSO optimizasyonunda ISTSE için elde edilen değer 97.782917 iken GA optimizasyonunda elde edilen değer 116.650891 olmuş ve yapılan simülasyon çalışmalarının neredeyse tamamında PSO ile yapılan optimizasyonların gerçek kodlu GA ile yapılan optimizasyonlardan daha iyi sonuçlar verdiği görülmüştür.

Çizelge 4.8'e dikkat edildiğinde de, PSO optimizasyonunda, BMD için 12.356316 olarak ölçülen IAE değeri, HBMD için 13.001366 olarak ölçülmüş; yine BMD için 8.740622 olarak ölçülen ISE değeri, HBMD için 8.906679 olarak ölçülmüş ve BMD ile HBMD'nin birbirlerine oldukça yakın sonuçlar verdiği görülmüştür.

Çizelge 4.7. $\frac{1}{(s+1)^4}$ doğrusal sistemi için elde edilen en iyi simülasyon sonuçları

Amaç Ölçüt Kriteri	Denklem (3.6) için PSO ile elde edilen sonuçlar		Denklem (3.6) için GA ile elde edilen sonuçlar	
	BMD	HBMD	BMD	HBMD
IAE	17,362332	17,624694	20,158756	20,305412
ISE	12,598768	13,186646	13,690123	13,798161
ITAE	54,787947	55,809667	78,170721	63,101010
ITSE	30,272145	32,336730	35,792362	35,701499
ISTSE	87,779864	97,782917	132,982118	116,650891

Çizelge 4.8. $\frac{1-0,1s}{(s+1)^3}$ doğrusal sistemi için elde edilen en iyi simülasyon sonuçları

Amaç Ölçüt Kriteri	Denklem (3.7) için PSO ile elde edilen sonuçlar		Denklem (3.7) için GA ile elde edilen sonuçlar	
	BMD	HBMD	BMD	HBMD
IAE	12,356316	13,001366	15,069761	13,832827
ISE	8,740622	8,906679	9,328499	9,287697
ITAE	31,555076	32,996354	45,077279	55,583106
ITSE	17,083826	16,668104	22,637433	20,892780
ISTSE	38,972694	45,327937	46,997735	62,329000

Çizelge 4.9. $\dot{y} = -0{,}2 \cdot [Sin(y) + Cos(y)] - \frac{y}{(1+y^2)} + [0{,}4 \cdot Sin(y) \cdot Cos(y^2)] + 0{,}8 \cdot u$ doğrusal olmayan sistem için elde edilen en iyi simülasyon sonuçları

Amaç Ölçüt Kriteri	Denklem (3.8) için PSO ile elde edilen sonuçlar		Denklem (3.8) için GA ile elde edilen sonuçlar	
	BMD	HBMD	BMD	HBMD
IAE	6,331656	1,102745	6,554135	1,865765
ISE	2,545969	0,578491	2,703925	0,650287
ITAE	14,496507	1,448733	15,726099	2,979898
ITSE	3,984379	0,620933	4,365466	0,912039
ISTSE	8,049387	0,380746	9,210110	1,093718

Ancak simülasyon çalışmaları sonucunda elde edilen grafiklerde yerleşme zamanı ve kararlı hal hataları dikkate alınırsa, özellikle PSO ile elde edilen sonuçlarda bulanık mantık denetleyicilerle, hiyerarşik bulanık mantık denetleyiciler için doğrusal sistem sonuçlarında bu iki değerin birbirleri ile aynı veya çok yakın olduğu belirlenmiştir.

Bu nedenle doğrusal olmayan sistemler için hiyerarşik bulanık mantık denetleyicilerin kullanılmasının iyi bir yöntem olacağı ve özellikle çok girişli doğrusal sistemler için de hiyerarşik bulanık mantık denetleyicilerin, bulanık mantık denetleyicilere iyi bir alternatif olabileceği düşünülmektedir.

5. SONUÇLAR VE ÖNERİLER

5.1. Sonuçlar

Klasik bulanık mantık denetleyici yapılarında, giriş değişkeni sayısının ya da bu değişkenler için kullanılan üyelik fonksiyonu sayısının artması sonucu üstel olarak artan kural sayısı kural tabanında önemli bir boyut sorunu oluşturmakta ve bu da uzman tanımlamasında zorluklara neden olmaktadır. Bu sorun, giriş değişkeni sayısının ikiden fazla olduğu denetleyicilerde daha da belirgin hale gelmektedir. Bunu ortadan kaldırabilmek amacıyla literatürde bulanık mantık denetleyicilerin hiyerarşik yapıda kullanılması önerilmektedir. Bu sayede klasik bulanık denetleyicilerde giriş sayısına bağlı olarak oluşan üstel kural artışı, hiyerarşik yapıyla birlikte doğrusal olarak artmaktadır.

Bu tez çalışmasında, giriş sayısı ikiden fazla ve üyelik fonksiyonu sayısı yüksek olan bulanık sistemler için, kural sayısının düşürülmesine imkan sağlayan Hiyerarşik Bulanık Mantık Denetleyiciler literatürde incelenmiş ve doğrusal/doğrusal olmayan sistemlerin kontrolünde kullanılmıştır. Bunun için, 3 girişli klasik bir bulanık mantık denetleyici ile 2 girişli iki bulanık denetleyicinin hiyerarşik olarak bağlanmasıyla elde edilmiş 3 girişli bir hiyerarşik bulanık mantık denetleyici optimal olarak tasarlanmış ve kontrol sonuçları karşılaştırılmıştır. Üç üyelik fonksiyonuna sahip girişler kullanılmış ve rastgele oluşturulmuş, simetrik bir kural tablosundan faydalanılmıştır.

Simülasyon çalışmalarında, denetleyicilerin giriş ve çıkış ölçeklendirme katsayıları hem PSO hem de gerçek kodlu GA ile ayrı ayrı optimize edilmiş ve sistem cevapları gözlenmiştir.

Sonuç olarak çalışmalarda, hiyerarşik bulanık mantık denetleyicilerin, doğrusal olmayan sistemlerde, klasik bulanık mantık denetleyicilerden çok daha iyi sonuçlar verdiği, doğrusal sistemlerde de, klasik bulanık denetleyicilere çok iyi bir alternatif olabileceği; ayrıca tüm simülasyon çalışmalarında PSO ile gerçekleştirilen optimizasyonlarda elde edilen cevapların, gerçek kodlu GA'ya göre daha iyi cevaplar olduğu amaç ölçüt kriterlerinden açıkça görülmüştür.

5.2. Öneriler

Hiyerarşik bulanık mantık denetleyiciler, ABS fren denetimi, gemi dümen denetimi, uçak kanat denetimi, roket hız denetimi ve uydu konum denetimi gibi çok girişli tüm klasik bulanık mantık denetim sistemlerine uygulanabilir. Ayrıca hiyerarşik bulanık mantık denetleyicinin optimal tasarımı için farklı optimizasyon yöntemleri uygulanabilir. Ayrıca hiyerarşik bulanık mantık denetleyicinin kontrol hızı, başta klasik bulanık mantık denetleyici olmak üzere diğer kontrol sistemlerinin kontrol hızlarıyla karşılaştırılarak, pratik uygulanabilirliği araştırabilinir.

KAYNAKLAR

Alkan Ö., 2011, Zamanla Değişen Sistemlerin Bulanık Model Referans Adaptif Kontrolü, Yüksek Lisans Tezi, Selçuk Üniversitesi Fen Bilimleri Enstitüsü, Konya

Altaş H.İ., 2008, Neural Fuzzy Systems Yüksek Lisans Ders Notları, KTÜ

Angeline, P.J., 1995, Evolution revolution: An introduction to the special track on genetic and evolutionary programming, IEEE Expert Intelligent Systems and their Applications, 10, 6-10

Aslan K., 2010, Doğrusal Hareketli Asenkron Motorun Bulanık Mantıkla Denetimi, Yüksek Lisans Tezi, Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü, Trabzon

Aydoğdu Ö., 2006, Fırçasız Doğru Akım Motorlarının Genetik Tabanlı Bulanık Denetleyici ile Sensörsüz Kontrolü, Doktora Tezi, Selçuk Üniversitesi Fen Bilimleri Enstitüsü, Konya

Aygün H., 2011, Akışkan Yataklı Buhar Kazanının Yatak Sıcaklığının Parçacık Sürüsü Optimizasyonu Tabanlı PID Kontrolör (PSO-PID) ile Kontrolü, Yüksek Lisans Tezi, Karabük Üniversitesi Fen Bilimleri Enstitüsü, Karabük

Baykal N. and Beyan T., 2004, Bulanık Mantık Uzman Sistemler ve Denetleyiciler, Bıçaklar Kitabevi, Ankara

Chang W-D. and Shih S-P., 2010, PID Controller Desing of Nonlinear Systems Using an Improved Particle Swarm Optimization Approach, Communications in Nonlinear Science and Numerical Simulation, 15 (11), 3632-3639

Chaturvedi D.K., 2008, Soft Computing: Techniques and its Applications in Electrical Engineering, Springer, Germany

Clerc M., 1999, The Swarm and The Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization, *Proc. 1999 ICEC*, 1951-1957

Çunkaş M. and Akkaya R., 2002, İkili ve gerçek kodlu genetik algoritmaların karşılaştırılması, S.Ü. Müh. Mim. Fak. Dergisi, 7.2, 11-17

Çunkaş M., 2004, Elektrik motorlarında genetik algoritma ile tasarım optimizasyonu, Doktora Tezi, S.Ü. Fen Bilimleri Enstitüsü, Konya

Elmas Ç., 2003, Bulanık Mantık Denetleyiciler, Seçkin Yayımcılık, Ankara, 90-104

Gaing Z-L., 2004, A Particle Swarm Optimization Approach for Optimum Design of PID Controller in AVR System, IEEE Transactions on Energy Conversion, 19 (2), 384-391

Goldberg D.E., 1989, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley, New York

Haupt R.L. and Haupt S.E., 1998, Practical Genetic Algorithms, A Willey – Interscience Publication, USA

Hu H., Hu Q., Lu Z. and Xu D., 2005, Optimal PID Controller Design in PMSM Servo System via Particle Swarm Optimization, 31st Annual Conference of IEEE Industrial Electronics Society, Raleigh, 79-83

Joo M. G. and Lee J. S., 1999, Hierarchical Fuzzy Control Scheme using Structured Takagi-Sugeno Type Fuzzy Inference, IEEE International Fuzzy Systems Conference Proceedings, Seoul, 78-83

Kikuchi H., Otake A. and Nakanishi S., 1998, Functional Completeness of Hierarchical Fuzzy Modeling, Information Sciences, 110, 51-60

Lee M-L., Chung H-Y. and Yu F-M., 2003, Modelling of Hierarchical Fuzzy Systems, Fuzzy Sets and Systems, 138, 343-361

Lin Y-L., Chang W-D. and Hsieh J-G., 2008, A Particle Swarm Optimization Approach to Nonlinear Rational Filter Modeling, Expert Systems with Applications, 34 (2), 1194-1199

Mahmood M. S., 2010, Bulanık Mantık Kullanılarak Trafik Kontrolünün Tasarımı ve Uygulaması, Yüksek Lisans Tezi, Ankara Üniversitesi Fen Bilimleri Enstitüsü, Ankara

Ortakcı Y., 2011, Parçacık Sürü Optimizasyonu Yöntemlerinin Uygulamalarla Karşılaştırılması, Yüksek Lisans Tezi, Karabük Üniversitesi Fen Bilimleri Enstitüsü, Karabük

Ou C. and Lin W., 2006, Comparison between PSO and GA for Parameters Optimization of PID Controller, Proceedings of the IEEE International Conference on Mechatronics and Automation, Luoyang, 2471-2475

Ömür Y. G., 2009, Adaptif Bulanık Denetleyici ile Doğrusal Olmayan Sistem Kontrolü, Yüksek Lisans Tezi, Ege Üniversitesi Fen Bilimleri Enstitüsü, İzmir

Passino K. and Yurkovich S., 1998, Fuzzy Control, Addison Wesley Longman Inc.

Pillay N. and Govender P., 2007, A Particle Swarm Optimization Approach for Model Independent Tuning of PID Control Loops, The 8th IEEE Africon Conference, Windhoek, 1-7

Sağlam G., 2007, Hiyerarşik Bulanık Mantık PID Kontrolörleri, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul

Shi Y. and Eberhrt R., 1998, A Modified Particle Swarm Optimizer, Proceedings of IEEE International Conference on Evolutionary Computation, USA, 69-73

Şahman M.A., 2008, Karma Yemlerin Genetik Algoritmayla Maliyet Optimizasyonu, Yüksek Lisans Tezi, S.Ü. Fen Bilimleri Enstitüsü, Konya

Teker A., 2008, Sürekli Mıknatıslı Senkron Motor'un Bulanık Mantık ile Hız Kontrolü, Yüksek Lisans Tezi, Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Kocaeli

Uçuk S., 2009, Bir Vinçteki Yük Salınımının Bulanık Mantık Tabanlı Kontrolü, Yüksek Lisans Tezi, Selçuk Üniversitesi Fen Bilimleri Enstitüsü, Konya

Wang L-X., 1998, Universal Approximation by Hierarchical Fuzzy Systems, Fuzzy Sets and Systems, 93, 223-230

Wang L-X., 1999, Analysis and Desing of Hierarchical Fuzzy Systems, IEEE Transactions on Fuzzy Systems, 7 (5), 617-624

Woo Z. W., Chung H. Y., Lin J. J., 2000, A PID-Type Fuzzy Controller with Self-Tuning Scaling Factors, Fuzzy Sets and Systems, 115, 321-326

Xu-zhou L.,Fei Y. and You-bo W., 2007, PSO Algorithm based Online Self-Tuning of PID Controller, International Conference on Computational Intelligence and Security, Harbin, 128-132

Yılmaz S., 2007, Bulanık Mantık ve Mühendislik Uygulamaları, Kocaeli Üniversitesi Yayınları, No:289, Kocaeli

Ying S., Zengqiang C. and Zhuzhi Y., 2007, Adaptive Constrained Predictive PID Controller via Particle Swarm Optimization, Proceedings of the 26th Chinese Control Conference, Hunan, 729-733

Yüksel İ., 2006, Otomatik Kontrol Sistem Dinamiği ve Denetim Sistemleri, Vipaş, Bursa

Zadeh L. A., 1965, Fuzzy Sets Information Control, Vol: 8, 338-353

EKLER

EK-1 BMD ile Gerçekleştirilen Simülasyonlarda Kullanılan Matlab Programları

Program 1: BMD ve PSO programı

```
c1=1;
c2=1;
psay=500;
parcaciklar=zeros(psay,14);
t=0:0.2:100;
h_ust=4;
h_alt=0.1;
ht_ust=4;
ht_alt=0.1;
htt_ust=4;
htt_alt=0.1;
o1_ust=4;
o1_alt=0.1;
it_say=0;
it_say_max=100;

for k=1:psay

   x=(rand*h_ust);
   if(x<h_alt)
       x=h_alt;
   end
   parcaciklar(k,1)=x;

   x=(rand*ht_ust);
   if(x<ht_alt)
       x=ht_alt;
   end
   parcaciklar(k,2)=x;

   x=(rand*htt_ust);
   if(x<htt_alt)
       x=htt_alt;
   end
   parcaciklar(k,3)=x;

   x=(rand*o1_ust);
   if(x<o1_alt)
       x=o1_alt;
   end
   parcaciklar(k,4)=x;

   h=parcaciklar(k,1);
   ht=parcaciklar(k,2);
   htt=parcaciklar(k,3);
   o1=parcaciklar(k,4);

   sim('satir1devre');

        for j=1:501
            error(j)=(simout(j)*simout(j))*t(j);
        end
        newerror=sum(error);
```

```
        simout=newerror;

   parcaciklar(k,5)=simout;

end

for it_say=0:it_say_max

    for i=1:psay
         if(parcaciklar(i,5)>parcaciklar(i,14)&&parcaciklar(i,14)~=0)
                   parcaciklar(i,1)=parcaciklar(i,10);
                   parcaciklar(i,2)=parcaciklar(i,11);
                   parcaciklar(i,3)=parcaciklar(i,12);
                   parcaciklar(i,4)=parcaciklar(i,13);
                   parcaciklar(i,5)=parcaciklar(i,14);
         end
    end

    [m,n]=min(parcaciklar(:,5));

    gbest_h=parcaciklar(n,1);
    gbest_ht=parcaciklar(n,2);
    gbest_htt=parcaciklar(n,3);
    gbest_o1=parcaciklar(n,4);

    fprintf('%d. İterasyon |
min.Hata=%f\t\n',it_say,parcaciklar(n,5));

    hiz=0.9-((0.5*it_say)/it_say_max);

         for i=1:psay

                pbest_h=parcaciklar(i,1);
                pbest_ht=parcaciklar(i,2);
                pbest_htt=parcaciklar(i,3);
                pbest_o1=parcaciklar(i,4);

            parcaciklar(i,6)=hiz*parcaciklar(i,6)+c1*rand*(pbest_h-
parcaciklar(i,1))+c2*rand*(gbest_h-parcaciklar(i,1));
            parcaciklar(i,7)=hiz*parcaciklar(i,7)+c1*rand*(pbest_ht-
parcaciklar(i,2))+c2*rand*(gbest_ht-parcaciklar(i,2));
            parcaciklar(i,8)=hiz*parcaciklar(i,8)+c1*rand*(pbest_htt-
parcaciklar(i,3))+c2*rand*(gbest_htt-parcaciklar(i,3));
            parcaciklar(i,9)=hiz*parcaciklar(i,9)+c1*rand*(pbest_o1-
parcaciklar(i,4))+c2*rand*(gbest_o1-parcaciklar(i,4));
         end

      .
      .
      .
      .
      .
      .
      .
      .
      .
```

Program 2: **BMD ve GA programı**

```
Npop_ilk=1000;
Npop=500;
it_say=100;
populasyon_ilk=zeros(Npop_ilk,5);
populasyon=zeros(Npop,5);
mutasyon_orani=0.3;
m_say=ceil(Npop*4*mutasyon_orani);
t=0:0.2:100;
h_ust=4;
h_alt=0.1;
ht_ust=4;
ht_alt=0.1;
htt_ust=4;
htt_alt=0.1;
o1_ust=4;
o1_alt=0.1;

for k=1:Npop_ilk

   x=(h_ust-h_alt)*rand+h_alt;
   populasyon_ilk(k,1)=x;

   x=(ht_ust-ht_alt)*rand+ht_alt;
   populasyon_ilk(k,2)=x;

   x=(htt_ust-htt_alt)*rand+htt_alt;
   populasyon_ilk(k,3)=x;

   x=(o1_ust-o1_alt)*rand+o1_alt;
   populasyon_ilk(k,4)=x;


   h=populasyon_ilk(k,1);
   ht=populasyon_ilk(k,2);
   htt=populasyon_ilk(k,3);
   o1=populasyon_ilk(k,4);

   sim('satir1devre');

   for j=1:501
       error(j)=(simout(j)*simout(j))*t(j);
   end
       newerror=sum(error);
       simout=newerror;

   populasyon_ilk(k,5)=simout;
end

populasyon_ilk=sortrows(populasyon_ilk,5);
populasyon=populasyon_ilk(1:Npop,:);

eniyi_s=1;

for s=1:it_say
    if(s==1)
        eniyipopulasyon(eniyi_s,:)=populasyon(1,:);
```

```
        eniyi_s=eniyi_s+1;
        fprintf('%d. İterasyon | min.Hata=%f\t\n',s-
1,populasyon(1,5));
    end

    rulet_mat=zeros((Npop),2);
    toplamuygunluk=0;
    farkmat=zeros(Npop,1);
    for i=1:Npop
        fark=populasyon(i,5)-populasyon(Npop,5);
        toplamuygunluk=toplamuygunluk+fark;
        farkmat(i,1)=fark;
    end

        uygunluk=farkmat(1,1);
        rulet_degeri=uygunluk/toplamuygunluk;
        rulet_mat(1,1)=rulet_degeri;
        rulet_mat(1,2)=rulet_degeri;

    for i=2:Npop
        uygunluk=farkmat(i,1);
        rulet_degeri=uygunluk/toplamuygunluk;
        rulet_mat(i,1)=rulet_degeri;
        rulet_mat(i,2)=rulet_mat(i-1,2)+rulet_degeri;
    end

    RandomNum = rand(Npop,1);

    for i=1:Npop
       index = find(RandomNum < rulet_mat(i,2));
       RandomNum(index) = i*ones(size(index));
    end
    populasyon = populasyon(RandomNum,:);

    for i=1:Npop/2
        anne=populasyon(i,:);
        baba=populasyon((Npop/2+i),:);
        c_noktasi=ceil(rand*4);

        nesil1=anne;
        nesil2=baba;

        if(c_noktasi==4)
            for k=1:3;
                temp=nesil1(k);
                nesil1(k)=nesil2(k);
                nesil2(k)=temp;
            end
        else
            for k=c_noktasi+1:4;
                temp=nesil1(k);
                nesil1(k)=nesil2(k);
                nesil2(k)=temp;
            end
        end
      .
      .
      .
      .
```

EK-2 HBMD ile Gerçekleştirilen Simülasyonlarda Kullanılan Matlab Programları

Program 1: HBMD ve PSO programı

```
c1=1;
c2=1;
psay=500;
parcaciklar=zeros(psay,17);
t=0:0.2:100;
h_ust=4;
h_alt=0.1;
ht_ust=4;
ht_alt=0.1;
htt_ust=4;
htt_alt=0.1;
o1_ust=4;
o1_alt=0.1;
o2_ust=4;
o2_alt=0.1;
it_say=0;
it_say_max=100;

for k=1:psay

   x=(rand*h_ust);
   if(x<h_alt)
       x=h_alt;
   end
   parcaciklar(k,1)=x;

   x=(rand*ht_ust);
   if(x<ht_alt)
       x=ht_alt;
   end
   parcaciklar(k,2)=x;

   x=(rand*htt_ust);
   if(x<htt_alt)
       x=htt_alt;
   end
   parcaciklar(k,3)=x;

   x=(rand*o1_ust);
   if(x<o1_alt)
       x=o1_alt;
   end
   parcaciklar(k,4)=x;

   x=(rand*o2_ust);
   if(x<o2_alt)
       x=o2_alt;
   end
   parcaciklar(k,5)=x;

   h=parcaciklar(k,1);
   ht=parcaciklar(k,2);
   htt=parcaciklar(k,3);
   o1=parcaciklar(k,4);
   o2=parcaciklar(k,5);
```

```
   sim('satirhiyerarsi1devre');

   simout=simout'*simout;

   parcaciklar(k,6)=simout;

end

for it_say=0:it_say_max

    for i=1:psay
         if(parcaciklar(i,6)>parcaciklar(i,17)&&parcaciklar(i,17)~=0)
                   parcaciklar(i,1)=parcaciklar(i,12);
                   parcaciklar(i,2)=parcaciklar(i,13);
                   parcaciklar(i,3)=parcaciklar(i,14);
                   parcaciklar(i,4)=parcaciklar(i,15);
                   parcaciklar(i,5)=parcaciklar(i,16);
                   parcaciklar(i,6)=parcaciklar(i,17);

         end
    end

    [m,n]=min(parcaciklar(:,6));

    gbest_h=parcaciklar(n,1);
    gbest_ht=parcaciklar(n,2);
    gbest_htt=parcaciklar(n,3);
    gbest_o1=parcaciklar(n,4);
    gbest_o2=parcaciklar(n,5);

    fprintf('%d. İterasyon |
min.Hata=%f\t\n',it_say,parcaciklar(n,6));

    hiz=0.9-((0.5*it_say)/it_say_max);

       for i=1:psay

                pbest_h=parcaciklar(i,1);
                pbest_ht=parcaciklar(i,2);
                pbest_htt=parcaciklar(i,3);
                pbest_o1=parcaciklar(i,4);
                pbest_o2=parcaciklar(i,5);

            parcaciklar(i,7)=hiz*parcaciklar(i,7)+c1*rand*(pbest_h-
parcaciklar(i,1))+c2*rand*(gbest_h-parcaciklar(i,1));
            parcaciklar(i,8)=hiz*parcaciklar(i,8)+c1*rand*(pbest_ht-
parcaciklar(i,2))+c2*rand*(gbest_ht-parcaciklar(i,2));
            parcaciklar(i,9)=hiz*parcaciklar(i,9)+c1*rand*(pbest_htt-
parcaciklar(i,3))+c2*rand*(gbest_htt-parcaciklar(i,3));
            parcaciklar(i,10)=hiz*parcaciklar(i,10)+c1*rand*(pbest_o1-
parcaciklar(i,4))+c2*rand*(gbest_o1-parcaciklar(i,4));
            parcaciklar(i,11)=hiz*parcaciklar(i,11)+c1*rand*(pbest_o2-
parcaciklar(i,5))+c2*rand*(gbest_o2-parcaciklar(i,5));
         end
      .
      .
      .
      .
```

Program 2: **HBMD ve GA programı**

```
Npop_ilk=1000;
Npop=500;
it_say=100;
populasyon_ilk=zeros(Npop_ilk,6);
populasyon=zeros(Npop,6);
mutasyon_orani=0.3;
m_say=ceil(Npop*5*mutasyon_orani);
t=0:0.2:100;
h_ust=4;
h_alt=0.1;
ht_ust=4;
ht_alt=0.1;
htt_ust=4;
htt_alt=0.1;
o1_ust=4;
o1_alt=0.1;
o2_ust=4;
o2_alt=0.1;

for k=1:Npop_ilk

   x=(h_ust-h_alt)*rand+h_alt;
   populasyon_ilk(k,1)=x;

   x=(ht_ust-ht_alt)*rand+ht_alt;
   populasyon_ilk(k,2)=x;

   x=(htt_ust-htt_alt)*rand+htt_alt;
   populasyon_ilk(k,3)=x;

   x=(o1_ust-o1_alt)*rand+o1_alt;
   populasyon_ilk(k,4)=x;

   x=(o2_ust-o2_alt)*rand+o2_alt;
   populasyon_ilk(k,5)=x;

   h=populasyon_ilk(k,1);
   ht=populasyon_ilk(k,2);
   htt=populasyon_ilk(k,3);
   o1=populasyon_ilk(k,4);
   o2=populasyon_ilk(k,5);

   sim('satirhiyerarsi1devre');

   for j=1:501
       error(j)=(simout(j)*simout(j))*(t(j)*t(j));
   end
       newerror=sum(error);
       simout=newerror;

   populasyon_ilk(k,6)=simout;

end

populasyon_ilk=sortrows(populasyon_ilk,6);
populasyon=populasyon_ilk(1:Npop,:);
```

```
 eniyi_s=1;

for s=1:it_say
    if(s==1)
        eniyipopulasyon(eniyi_s,:)=populasyon(1,:);
        eniyi_s=eniyi_s+1;
        fprintf('%d. İterasyon | min.Hata=%f\t\n',s-
1,populasyon(1,6));
    end

    rulet_mat=zeros((Npop),2);
    toplamuygunluk=0;
    farkmat=zeros(Npop,1);
    for i=1:Npop
        fark=populasyon(i,6)-populasyon(Npop,6);
        toplamuygunluk=toplamuygunluk+fark;
        farkmat(i,1)=fark;
    end

        uygunluk=farkmat(1,1);
        rulet_degeri=uygunluk/toplamuygunluk;
        rulet_mat(1,1)=rulet_degeri;
        rulet_mat(1,2)=rulet_degeri;

    for i=2:Npop
        uygunluk=farkmat(i,1);
        rulet_degeri=uygunluk/toplamuygunluk;
        rulet_mat(i,1)=rulet_degeri;
        rulet_mat(i,2)=rulet_mat(i-1,2)+rulet_degeri;
    end

    RandomNum = rand(Npop,1);

    for i=1:Npop
       index = find(RandomNum < rulet_mat(i,2));
       RandomNum(index) = i*ones(size(index));
    end
    populasyon = populasyon(RandomNum,:);

    for i=1:Npop/2
        anne=populasyon(i,:);
        baba=populasyon((Npop/2+i),:);
        c_noktasi=ceil(rand*5);

        nesil1=anne;
        nesil2=baba;

        if(c_noktasi==5)
            for k=1:4;
                temp=nesil1(k);
                nesil1(k)=nesil2(k);
                nesil2(k)=temp;
            end
      .
      .
      .
      .
      .
```

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Serhat SOYLU
Uyruğu : T.C.
Doğum Yeri ve Tarihi : Konya 24.05.1983
Telefon : 332-4612890
Faks : 332-4612892
e-mail : serhatsoylu@selcuk.edu.tr

EĞİTİM

Derece	Adı, İlçe, İl	Bitirme Yılı
Lise	: Konya Meram Anadolu Lisesi – KONYA	2001
Üniversite	: Konya Selçuk Üniversitesi Elektrik-Elektronik Mühendisliği	2005
Yüksek Lisans	: Selçuk Üniversitesi FBE Elektrik-Elektronik Müh. A.B.D. (Tezsiz Yüksek Lisans)	2007
Yüksek Lisans	: Selçuk Üniversitesi FBE Elektrik-Elektronik Müh. A.B.D.	2013
Doktora	: ---	

İŞ DENEYİMLERİ

Yıl	Kurum		Görevi
2007 – …	S.Ü. Akören Ali Rıza Ercan MYO	KONYA	Öğretim Görevlisi
2005 – 2007	S.Ü. Akören Ali Rıza Ercan MYO	KONYA	Öğretim Elemanı

UZMANLIK ALANI

Bulanık Mantık, Bulanık Denetleyiciler ve Hiyerarşik Bulanık Mantık Kontrol

YABANCI DİLLER

İNGİLİZCE : Orta seviye
ÜDS (58 Puan)