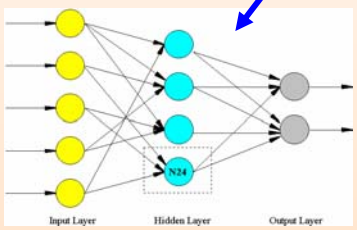
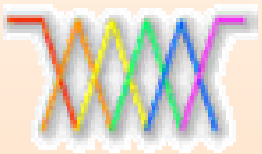
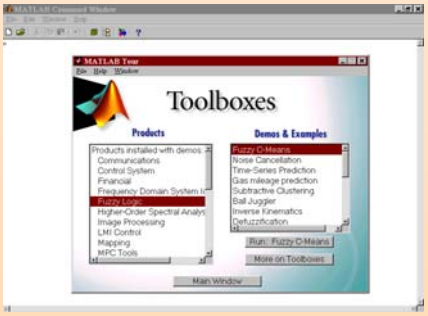
BULANIK MANTIK
ve
MATLAB UYGULAMALARI







MUSA ALCI

ENGİN KARATEPE

BULANIK MANTIK
ve
MATLAB UYGULAMALARI

BİRİNCİ BASKI

Musa Alcı

alci@bornova.ege.edu.tr
Ege Üniversitesi Mühendislik Fakültesi
Elektrik ve Elektronik Mühendisliği Bölümü
Bornova, İzmir, TÜRKİYE

Engin Karatepe

engin@bornova.ege.edu.tr
Ege Üniversitesi Mühendislik Fakültesi
Elektrik ve Elektronik Mühendisliği Bölümü
Bornova, İzmir, TÜRKİYE

Önsöz

Bulanık Mantığın kullanım alanları gerek akademik çalışmalarda gerekse uygulama alanlarında giderek artmaktadır. Bu konuda pek çok yayın/kitap yayınlanmıştır. Bu yayınlardan biri de Li-Xin Wang'ın "*A Course in Fuzzy Systems and Control*" isimli kaynak kitabıdır. Burada, bu kitabın anlaşılmasında ona yardımcı bir doküman oluşturulması hedeflenmiştir. Bu çalışmada bu kitapta verilen örneklerin ve problemlerin Matlab'deki çözümleri verilmeye çalışılmıştır. Son bölümde, Matlab Fuzzy Logic Toolbox kullanarak bazı kontrol uygulamaları için simulasyon örnekleri verilmiştir. Geleneksel anlamda oluşturulan PI, PD ve PID ile Fuzzy PI, Fuzzy PD ve Fuzzy PID gerçeklemeleri yapılıp, daha sonra bunların karşılaştırmaları yapılmıştır.

Problemlerin çözümünde şu yol izlenmiştir: Öncelikle problemin Matlab'deki çözümü ardından da ilgili çıktılar verilmiştir. Gerek duyuldukça problemle ilgili konuda teorik alt yapı verilmiştir. Verilen kodların tamamı Matlab 5.1 de test edilmiştir.

Kitap sonunda araştırmacıların çok geniş bir alanda faydalanabilecekleri internet adresleri ve bağlantılar verilmiştir.

Özellikle konuya yabancı olanların ve yeni ilgi duymaya başlayanların çalışmadan en üst düzeyde fayda sağlaması ümit edilmektedir.

Bu kitabın, Bulanık Mantık konusunda çalışacak araştırmacılara ve öğrencilere faydalı olmasını diliyorum.

Okuyucuların tavsiye, öneri ve geri beslemeleri gelecek çalışmalarda bize ışık tutacaktır. Bize alci@bornova.ege.edu.tr mail adresi üzerinden ulaşabilirsiniz.

Halen Ege Üniversitesinde vermekte olduğum "Bulanık Mantık ve Uygulamaları" isimli yüksek lisans dersinde verilen ödevler ve projeler bu çalışmaya ilham kaynağı olmuştur. Bu bağlamda yüksek lisans öğrencilerime ve özverili katkılarından dolayı Engin Karatepe'ye teşekkürü bir borç bilirim.

Dr. Musa ALCI
Eylül 2002
Bornova, İzmir

Önsöz

Günümüzde bulanık uzman sistemlerin lineer ve nonlineer kontrol, örnek tanıma, finansal sistemler, işletme araştırmaları ve veri analizleri gibi birçok alanda proje ve araştırmalarla oldukça yaygınlaştığı görülmektedir. Birçok sistemin, bulanık sistemler yardımı ile modellenebilmesi ve hatta kopyalanabilmesi, gerek bilimsel alanda gerekse hayatımızda oldukça önemli gelişmelerin kapısını aralamıştır.

Ülkemizde de gelişen teknolojinin yazılım ve donanım geliştirme ortamları üzerinde tasarımcıya sunmuş olduğu imkanlardan faydalanması kapsamında Bulanık Mantık üniversitelerin eğitim programına girmiştir. Tüm branşlarda olduğu gibi Türkçe kitapların yetersizliği ve ders kitaplarının eksikliği bu çalışmanın başlamasına ve bundan sonraki çalışmalara ışık tutacak bir kitabın oluşmasına neden olmuştur.

Kısa ve öz bir yaklaşımla yazılmış Bulanık Mantık temel konularını içeren bu ders kitabının araştırmacılara ve öğrencilere yardımcı bir rehber olacağı düşüncesindeyim.

Büyük bir zevkle hazırladığımız bu kitapta, deneyimi, birikimi ve yüreklendirici tutumu ile hocam Sayın Yard. Doç. Dr. Musa ALCI'ya en içten teşekkürlerimi sunarım.

Engin Karatepe
Eylül 2002

İÇİNDEKİLER

Önsöz **4**
Bölüm 1 ..... **8**
Giriş **8**
1.1 Neden Bulanık Mantık? 8
1.2 Bulanık Sistem Nedir? 12
1.3 Bulanık Mantığın Tarihi Gelişimi 15
INSPEC/fuzzy 15
1.4 Bulanık Mantık Uygulama Alanları 16
Bölüm 2 ..... **17**
Klasik ve Bulanık Küme Kavramı **17**
2.1 Tanımlar 17
2.2 Bulanık Kümelerle Temel İşlemler 20
2.3 Bulanık Birleşim ve *S* Normları 24
2.4 Bulanık Kesişim ve *T* Normları 25
2.5 *S* Norm ve *T* Norm Karşılaştırması 26
Bölüm 3 ..... **29**
Sözsel Değişkenler ve IF-THEN Kuralları **29**
3.1 Sözsel Değişkenlerden Sayısal Değerlere 29
Bölüm 4 ..... **31**
Bulanık Sonuç Çıkarma (Fuzzy Inference) **31**
4.1 Bulanıklaştırıcı (Fuzzifier) 32
4.2 Durulayıcı (Defuzzifier) 33
4.3 Çıkarım Metotlarına Bir Örnek 36
Bölüm 5 ..... **38**
Giriş Çıkış Bilgisinden Bulanık Sistem Tasarımı **38**
5.1 Table Look-Up Kullanarak Bulanık Küme Tasarımı 39
5.2 Table Look-Up Kullanarak Bulanık Küme Tasarımı ile Uygulamalar 41
5.2.1 Zaman Serisi Tahmini 41
5.2.2 Araç Park Uygulaması 45
5.3 Gradyen Tabanlı Eğitme (Gradient Descent Training) ile Bulanık Sistem Tasarımı 54
5.4 Gradyen Tabanlı Bulanık Sistem Tasarımı Uygulamaları 57
5.4.1 Gradyen Yöntemle Basit Fonksiyon Eğitme 57
5.4.2 Örnek 60
5.4.3 Örnek 63
5.4.4 On line parametre atama yöntemi ve adım adım eğitme ile ilgili dinamik sistem eğitme örneği 67

5.4.5 Arttırılmış Bulanık Sistemle Sistem Tanımlama Optimizasyon/Eğitme Yöntemi: BFGS 73
5.5 En Küçük Kareler (Recursive Least Squares) Kullanarak Bulanık Sistem Tasarımı....76
5.6 En Küçük Kareler Bulanık Sistem Tasarımı Uygulamakları....78
5.6.1 Örnek 1....78
5.6.2 Bulanık Taban Fonksiyonları....80
5.6.3 RLS ile FS sistem Tanımlama Örneği 81
5.7 Clustering Kullanarak Bulanık Sistem Tasarımı 85
5.8 Kümeleme Kullanarak Bulanık Sistem Tasarımı Uygulamaları 87
5.8.1 Li-Xin Wang'ın Kitabındaki Örnek 15.2....87
5.8.2 Bulanık Kümeleme ile sin(x) Fonksiyonunun Tanımlanması....90
Bölüm 6 93
MATLAB Fuzzy Logic Toolbox Kullanarak Bulanık Mantık Tabanlı Sistem Tasarımı 93
6.1 Matlab Bulanık Mantık GUI Araçları ve İşlevleri....93
6.2 MATLAB Simulink Ortamında Uygulamalar....94
6.2.1 Geleneksel PD Denetleyici 94
6.2 2. Bulanık Mantık PD Denetleyici....98
6.2.3 Geleneksel PID Denetleyici....99
6.2.4 Bulanık Mantık PID Denetleyici 102
6.2.5 Bulanık Mantık PD Denetleyicinin MATLAB Fuzzy Logic Toolbox Ortamında Gerçekleştirilmesi....103
6.2.6 Bulanık Mantık PID Denetleyicinin MATLAB Fuzzy Logic Toolbox Ortamında Gerçekleştirilmesi....108
Faydalı Linkler....113
Bulanık Mantıkla ilgili Kaynak Kitaplar ve Makaleler 116
Dizin 117

Bölüm 1

Giriş

1.1 Neden Bulanık Mantık?

Modern bilgisayarların ikili mantığı gerçek dünyanın belirsizliğini betimlemekte yetersizdir. Bulanık mantık daha iyi seçenekler sunmaktadır. Bilgisayarlar, insan beyni gibi akıl yürütmez. Bilgisayarlar, sıfır ve bir dizilerine indirgenmiş kesin gerçekler ve doğru ya da yanlış olan önermeler kullanarak akıl yürütür. İnsan beyni ise serin hava ya da yüksek hız ya da genç kız gibi belirsizlik ya da değer yargıları içeren bulanık anlatım ve önermelerin üstesinden gelebilecek biçimde akıl yürütebilir. Ayrıca insan, bilgisayardan farklı olarak, hemen her şeyin ancak kısmen doğru olduğu bir dünyada akıl yürütebilmek için sağduyusunu kullanır.

Bulanık mantık belirsiz bir dünyanın gri, sağduyulu resimlerini üretmeleri için bilgisayarlara yardımcı olan makine zekası biçimidir.

Bulanık mantık sıcak ya da hava kirli gibi kavramlar kullanır ve bu sayede, hangi hızla çalışacağına ya da programlandığı bir aşamadan diğerine ne zaman geçeceğine kendisi karar veren havalandırma, çamaşır makinesi ve benzeri aygıtları yapabilmeleri için mühendislere yardımcı olur. Mühendislerin elinde bir sistemin girdilerine yanıt verecek özel algoritmalar bulunmadığında bulanık mantık belirsiz niceliklere başvuran sağduyulu kurallar kullanarak sistemi denetleyebilir ve betimleyebilir. Bilinen hiçbir matematiksel model bir kamyonun park yerinden yükleme yerine gidişini kamyonun hareket noktası rasgele seçilebiliyorsa yönetemez. Oysa gerek insan gerekse bulanık mantık sistemleri, kamyon biraz sola dönerse, sen de biraz sağa çevir gibi pratik, ancak kesinlik taşımayan kurallar kullanarak bu doğrusal olmayan kılavuzluk işlemini gerçekleştirebilirler. Bulanık mantık sistemleri kurallarını çoğu zaman uzmanlardan öğrenir. Kuralları belirleyen bir uzman yoksa, uyum sağlama yeteneğine sahip (adaptive) bulanık mantık sistemleri işleyiş kurallarını, insanların gerçek sistemleri nasıl düzenlediklerini gözlemleyerek öğrenir.

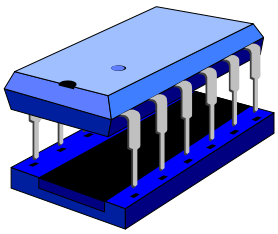
Bulanık mantık uygulama alanları kontrol sistemlerinin de ötesine uzanmaktadır. Bu konudaki gelişmeler, bulanık mantığın;

mühendislik, fizik, biyoloji ya da ekonomi gibi pek çok konuda, sürekli sistemleri modellemek üzere kullanılabileceğini göstermektedir. Çoğu alanda bulanık mantık sezgisel modellerinin standart matematik modellerinden daha yararlı ya da kesin sonuçlar verdiği görülmektedir.

Büyük bir çoğunluğunun Japonya'da üretilen bulanık mantık ürünleri bulanık mantığın yaygınlaşmasına yol açmıştır. İlk olarak 1980'de Kopenhag'de F.L. Smidth & Company isimli firma bir çimento ocağının işleyişini kontrol amacıyla bir bulanık mantık sistemi kullandı. 1988'de Hitachi firması Japonya'da metronun kontrolünü bir bulanık mantık sistemine bıraktı. Daha sonra Japonlar yüzlerce ev aygıtı ve elektronik ürünün yönlendirilmesinde bulanık mantık kullanmaya başladılar.

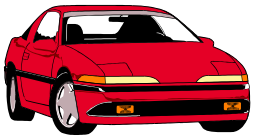
Bulanık mantıklı ürünler hem bulanık mantık yürütme algoritmaları hem de değişen girdi koşullarını ölçen algılayıcılar kullanır. Bulanık mantık entegreleri bulanık mantık kurallarını saklayacak ve işleyebilecek biçimde tasarlanmış mikroişlemcilerdir. Masaki Togai ve Hiroyuki Watanabe, AT&T'nin Bell laboratuvarlarında çalıştıkları 1985 yılında ilk dijital bulanık entegreyi yaptılar. Bu entegre saniyede 0.08 milyon bulanık mantık yürütme hızıyla, 12.5 mikrosaniyede 16 basit kuralı işleyebiliyordu.

Togal Infra Logic Inc. gibi şirketler günümüzde bulanık işlem hızlandırıcıları donanımlarıyla saniyede milyonlarca kuralı işleyebilen yongalar üretmektedirler. Günümüzde mikroişlemci firmalarının çoğu bulanık entegre üzerine araştırma projeleri başlatmışlardır.

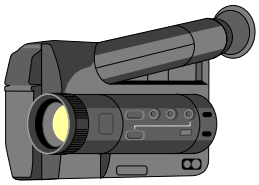


Japonya metro treninde kullanılan bulanık mantık denetleyicisi, gerek insan sürücülerden gerekse otomatik kontrol denetleyicilerden daha iyi iş çıkarmıştır. Klasik denetleyiciler, bir aracın istasyondan ne uzaklıkta olduğunu gösteren konum işaretlerine göre bir treni çalıştırır ya da durdururlar. Bu denetleyiciler çok kesin programlanmış oldukları için yolculuk sarsıntılı olabilir. Otomatik denetleyici tren

istasyondan örneğin 100 metre kadar uzakta iken yokuş yukarı ya da aşağı gittiğine bakmadan aynı fren basıncını uygular. Metro trenlerinde becerikli bir insan sürücüden daha da yumuşak biçimde hızlandırmak, yavaşlatmak ve durdurmak için bulanık kurallar kullanılmıştır. Bu kurallar trenin çalışması ile ilgili olarak her an hızının ne sıklıkta ve ne ölçüde değiştiğini, maksimum hıza ne kadar yaklaştığı gibi geniş bir değişkenler dizisini kapsar. Bulanık mantık denetleyicileribenzeşim denemelerinde yolcuların rahatı, kısalan yolculuk süresi gibi konularla ilgili ölçümlerde otomatik denetleyiciyi yenmiş, ayrıca trenin enerji tüketimini de yüzde 10 oranında azaltmayı başarmıştır.



Bulanık mantık, kameralarda görüntü verilerini çeşitli mercek düzeneklerine iletebilir. 1990'da pazara sürülen ve ilk bulanık mantık kameralardan biri olan Canon'un el tipi, 13 bulanık mantık kuralı kullanarak otomatik odaklama yapabiliyor. Algılayıcılar altı ayrı alanda görüntülerin berraklığını ölçüyor. Kurallar yaklaşık bir kilobyte kadar bellek gerektiriyor ve algılayıcı verilerini anında yeni mercek ayarlarına dönüştürüyor.



Elin hareketinden kaynaklanan görüntü titremelerini dengelemek için daha çok kural kullanılan kameralar geliştirilmiştir. Bulanık kurallar yardımı ile görüntünün ne yöne kayacağını belirliyor ve görüntüdeki bölgesel ve genel değişiklikleri göz önünde tutarak gerekli dengeleme işlemlerini yapıyor. Matematik modellere dayanan sistemlerde denetleyicileri en çok bir kaç tür görüntü titremesini dengelemektedir. Bir işin yapılması için ne kadar güç gerektiğini daha doğru hesapladıkları için bulanık sistem denetleyicileri enerjiyi çoğu zaman daha verimli kullanır.

Bulanık mantıktan helikopterlerde de yaralanılır. Bu aracın dört parçası, asansör, kanatçık, valf ve dümen – ”yukarı çık”, “yere in” ve “havada kal” gibi 13 bulanık ses komutunun istediğini yerine getirmektedir. Bulanık mantık denetleyici aracı havada tutmak gibi pilotlar için bile güç olan bir işi başarabilmektedir.

Bulanık mantık mekanik sistemlerin ötesinde bilgi de yönetebilmektedir. Bulanık sistemler hastaların sağlık durumlarını öğrenmek ve hastalıklarından korunmalarına, sağlıklı kalmalarına ve stresten kurtulmalarına yardımcı olmak üzere kişiye özgü planlar çizebilen kurallar oluşturabilirler. Japon Omron grubu bu konuda tıp veri tabanlarını bulanık mantık denetleyicileri ile denetleyen sistemler kurmuşlardır.

Bulanık mantıktan uygulamaları otomobiller üzerinde de yapılmıştır. General Motors bulanık mantıkla çalışan bir vites tasarlamıştır. Nissan, bulanık mantıkla işleyen ABS fren sistemi, bulanık bir vites sistemi, bir de bulanık yakıt enjektörü patenti almıştır. Ana karttaki bir mikroişlemcide saklı bir dizi bulanık kural yakıt akışını düzenler. Algılayıcılar valf ayarını, manifold borusu basıncını, radyatör suyu sıcaklığını ve motorun devir sayısını ölçer. İkinci bir bulanık mantık kural kümesi ise motorun bir dakikadaki dönme sayısını, su sıcaklığı ve oksijen yoğunluğuna göre makinenin ateşleme zamanını ayarlar.

Bazı şirketler ise tahvil ve hisse senedi fonları için ekonomik verilerdeki değişikliklere göre davranmak üzere bulanık mantık kurallar kullanan alışveriş programları geliştirmişlerdir.

Burada bulanık sistemlerin gittikçe artan sayıda bilgisayara, ev aletlerine ve teorik modellere kolayca girebileceği, örneklerle verilmeye çalışılmıştır. Gelecek yüzyıl düşlenildiğinden de bulanık olabilir.

1.2 Bulanık Sistem Nedir?

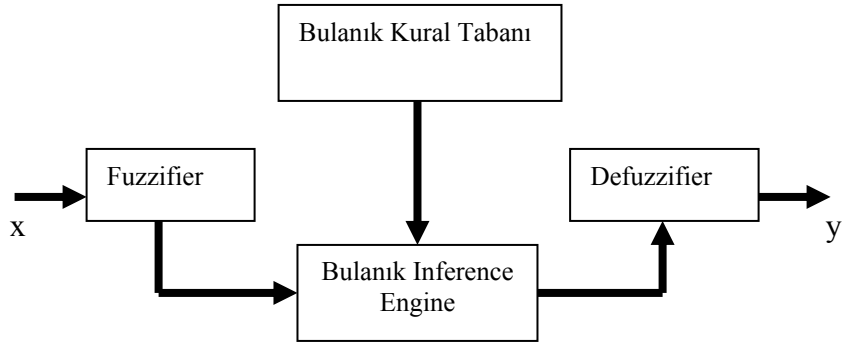
Çölde kayboldunuz, elinizde 2 şişe su var; birinin üzerinde %91 olasılıkla kirli su, diğerinin üzerinde %91'i kirli su yazıyor. Hangisini içersiniz?

Bulanık mantık ile işleyen bir sistem kurmak için bir uzmandan alınacak bir dizi bulanık mantık kuralıyla işe başlanabilir. Çeşitli bulanık girdi ve çıktı kümelerine ait üyelik dereceleri, bir takım eğri kümeleri ile tanımlanır. Daha sonra girdi ve çıktı kümeleri arasında bu ilişkinin grafiği çizilebir. "Hava serin gibiyse motorun hızını düşür" kuralı verildiğinde, girdiler (sıcaklık) grafiğin bir ekseni boyunca, çıktılar ise (motorun hızı) ikinci eksen boyunca sıralanır. Bu bulanık kümeler bulanık bir alan, yani söz konusu kuralın girdi ve çıktılar arasında oluşturduğu bütün birlikteliklerin kümesini temsil eden bulanık bir bölge oluşturacktır. Bulanık alanın büyüklüğü kuralın bulanık ya da belirsizliğini yansıtır. Bulanık küme kesinleştikçe alan da küçülür

Bulanık bir sistemin kuralları girdilerin tümünü çıktıların tümüyle ilişkilendirecek, örtüşen alanlardan oluşan bir küme tanımlar. Bu anlamda bulanık sistem matematiksel bir neden-sonuç fonksiyonu ya da denklemine yaklaşır. Bu fonksiyonlar bir mikroişlemciye bir klima cihazının gücünü ya da çamaşır makinesinin hızını, yapılan en son ölçüme uygun olarak nasıl ayarlaması gerektiğini söyeleyen kurallar olabilir.

Şekil 1.1'de basit bir bulanık sistemin konfigurasyonu verilmiştir. Burada "fuzzifier" aşamasında gerçek dünyanın değerleri bulanık kümelerde üyelik değerlerine dönüştürülürler. Daha sonra "Bulanık Inference Engine" ile IF-THEN kuralları girdi ve çıktı uzayında

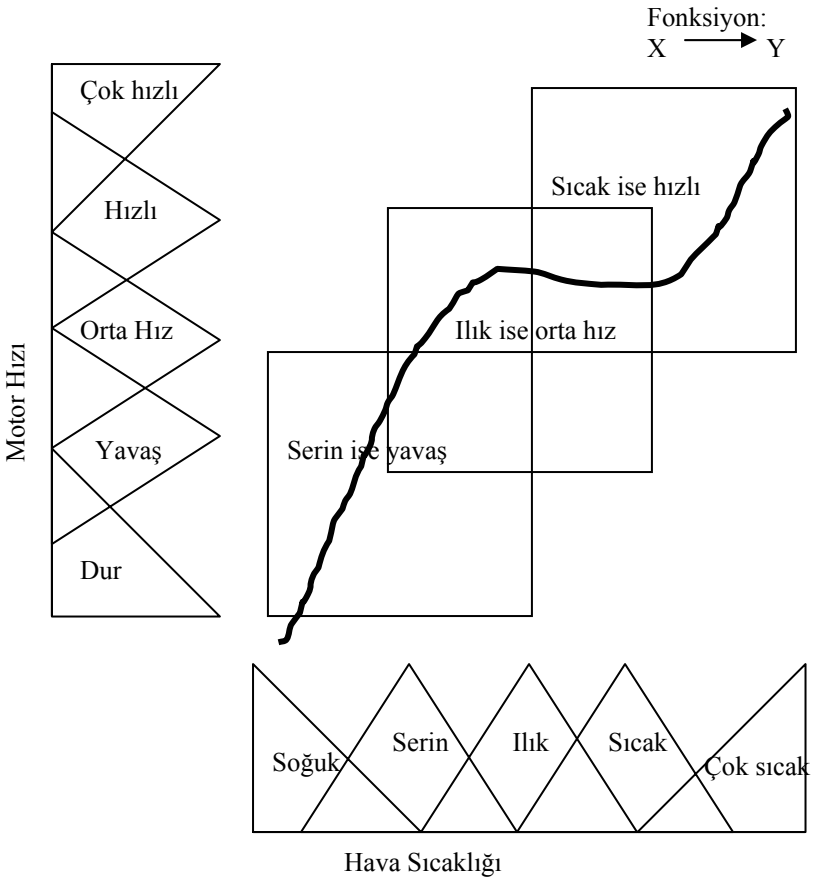
tanımlı bir bulanık ilişkiye dönüştürülür. “Defuzzifier” kısmında ise bulanık küme gerçek dünyanın değerlerine dönüştürülür.

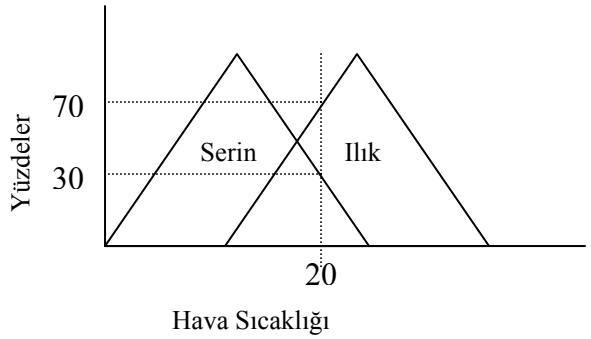


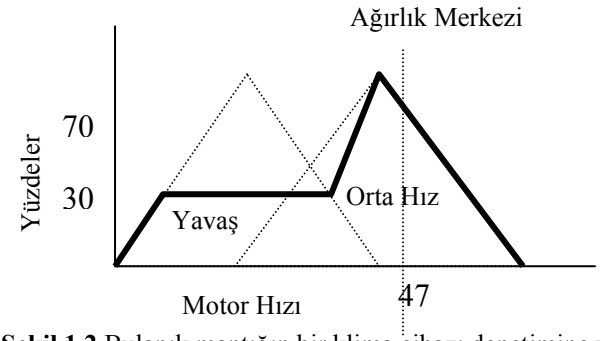
Şekil 1.1 Temel bir bulanık sistemin konfigürasyonu

Hava sıcaklığını motor hızlarıyla eşleştirmek için beş kurala, dolayısıyla beş bulanık alana dayanan bulanık bir klima cihazı düşünelim. Sıcaklık kümeleri (soğuk, serin, ılık, sıcak ve çok sıcak) olabilecek bütün bulanık girdileri kapmaktadır. Motor hızı kümeleri ise (çok yavaş, yavaş, hızlı ve çok hızlı) bütün bulanık çıktıları vermektedir. Örneğin 20 derece sıcaklık yüzde 30 serin ve yüzde 70 ılık olabilir(Şekil 1.2). Aynı anda hava da yüzde sıfır soğuk, sıcak ve çok sıcaktır. Serinse ve ılıksa kuralları hem yavaş hem de orta motor hızlarını harekete geçirip çalıştıracaktır. Her iki kural motorun son hızına orantılı olarak katkıda bulunur. Hava yüzde 20 serin olduğunda motor hızının yavaşlığını betimleyen eğri, yüksekliğinin yüzde 30’una, orta hız eğrisi de yüzde 70’e inmelidir. İndirgenmiş bu iki eğri toplanarak bulanık çıktı kümesinin son eğrisi elde edilir.

Bu tür bir çıktı eğrisi bulanık mantık biçimindeyken binary sistemine göre çalışan denetleyicilere yardımcı olamaz. Bu nedenle son adım bulanık çıktı eğrisinin sayısal tek bir değere dönüştüğü bir “defuzzifier” sürecidir. En yaygın defuzzifier tekniği eğri alanın ağırlık merkezini hesaplamaktır. Bu örnekte bulanık mantık çıktı eğrisinin ağırlık merkezi dakikada 47 dönüşlü bir motor hızına denk olabilir. Böylece elektronik denetleyici niceliksel bir sıcaklık girdisiyle bulanık sıcaklık ve motor hızı kümeleri yardımıyla akıl yürüterek uygun ve kesin bir hız çıktısı elde edilebilir.







Şekil 1.2 Bulanık mantığın bir klima cihazı denetimine uygulanması

1.3 Bulanık Mantığın Tarihi Gelişimi

1960'lar : Bulanık mantığın başlangıcı

1970'ler : Teoremlerin gelişimi ve ilk uygulamalar

1980'ler : Önemli uygulamaların başlangıcı

1990'lar : Bulanık mantık uygulamalarının olgunlaşması

1993 : IEEE Transaction Fuzzy Systems dergisinin çıkışı

Bulanık mantık uygulamalarının akademik çevrede de çarpıcı gelişmesini yıllara göre gösteren makale sayıları aşağıdaki gibidir.

INSPEC/fuzzy

1970-1980 : 566

1980-1990 : 2,361

1990-2000 : 23,753

Toplam : 26,680

Math.Sci.Net/fuzzy

1970-1980 : 453

1980-1990 : 2,476

1990-2000 : 8,428

Toplam : 11,357

INSPEC/soft computing

1990-2000 : 1,994

1.4 Bulanık Mantık Uygulama Alanları

Lineer ve Nonlineer Kontrol Sistemleri

Sinyal İşleme

Entegre Devre Üretimi

Tıp, Psikoloji

Ekonomik Sistemler

Kuantum Fiziği

Veri Analizi

Veri Tabanı Uygulamaları

Toplum Bilimi

Proses Planlama

Karar Verme

Bölüm 2

Klasik ve Bulanık Küme Kavramı

2.1 Tanımlar

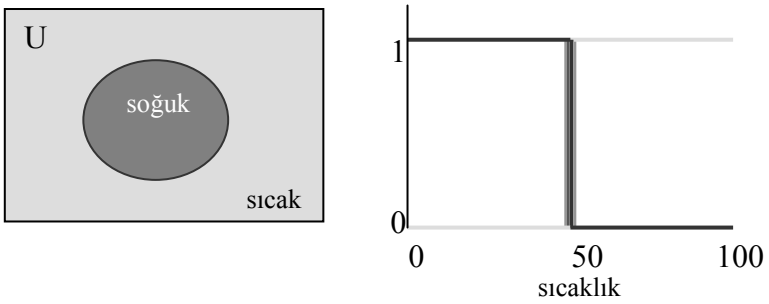
Standart küme kuramında herhangi bir nesne bir kümeye ya aittir ya da değildir. Bunun ortası yoktur. Ayrıca bu iki kümeden hiçbirine ait olmaması söz konusu değildir. Bu ilke bir nesnenin aynı anda hem bir şey olması hem de o şey olmaması çelişkisini olanaksız kılmaktadır.

Bulanık mantıkta, elemanlar bulanık bir kümeye ancak kısmen aittir. Ayrıca, aynı anda birden çok kümeye de ait olabilirler. Hava, tek bir kişiye bile değişik derecelerde serin, ılık ya da sıcak gelebilir. Standart kümelerin sınırları ise eğridir yada giderek yok olur ve bu eğrilik kısmı bulanıklık oluşturur. Hava yüzde 20 serin olabilir aynı anda yüzde 80 serin değil olabilir.

Bulanık mantık olasılık yüzdeleri ile aynı şeyi ifade etmemektedir. Olasılıklar bir şeyin olup olamayacağını ölçer. Bulanık mantık ise bir olayın ne dereceye kadar olduğunu, bir koşulun ne dereceye kadar var olduğunu ölçer. Yüzde 30 olasılıkla hava serin olacak önermesi serin hava olasılığını dile getirir. Fakat sabah hava yüzde 30 serin gibi geliyor önermesi hava bir dereceye kadar serin aynı zamanda da değişen derecelerde ılık ve sıcak demektir.

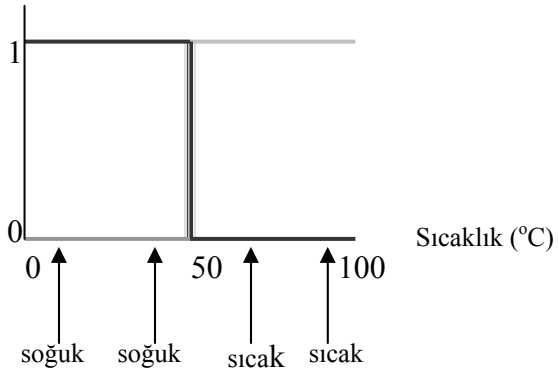
İnsan zekası gerektiren problemlerin matematiksel modellenmesinde klasik küme teorisi yetersiz kalmaktadır.

Örneğin, Şekil 2.1'de bir odanın sıcaklığı klasik küme teorisi kullanılarak karakterize edilmiştir. Açıkça görüldüğü üzere klasik teoride 50 derecenin üstündeki sıcaklıklarda havanın sıcak, altında ise havanın soğuk olduğu sonucu çıkmaktadır. Üyelik değeri 1 ya da 0'dır.



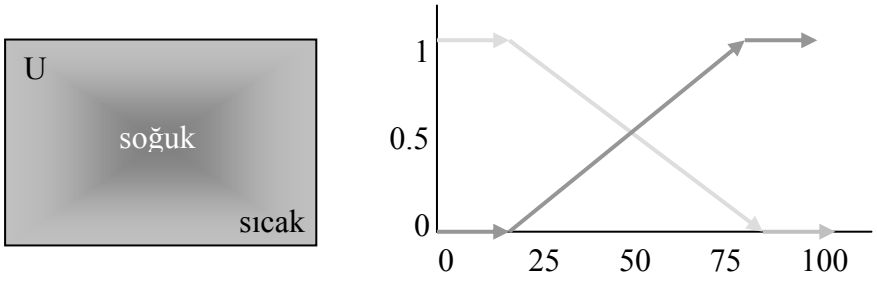
Şekil 2.1 Herhangi bir sıcaklık soğuk kümesinin ya üyesidir ya da değildir.

Aynı zamanda Şekil 2.2'de görüldüğü gibi 100 derecedeki bilgi de 51 derecedeki bilgi de aynı derecedeki sıcak kavramı ile tanımlanmaktadır.



Şekil 2.2 Klasik küme mantığındaki karşılaşılan problem.

Günlük hayatta ise çok soğuk, soğuk, ılık, çok sıcak gibi kavramlar kullanılmaktadır. Bu kavramlar arasındaki geçiş klasik mantıktaki gibi bu kadar keskin değildir. Bu doğal olayları bulanık kümelerle daha doğru karakterize etmek mümkün olmaktadır. Şekil 2.3'de aynı bilgi bulanık kümesi ile karakterize edilmiştir.



Şekil 2.3 Herhangi bir sıcaklık değeri hem soğuk kümesine hem de sıcak kümesine ait olabilir. Üyelik değeri 1 ile 0 arasındadır.

Bir bulanık A kümesi U uzayında tanımlanan ve [0, 1] aralığında değerler alabilen $\mu_A(x)$ üyelik fonksiyonu ile karakterize edilen kümedir.

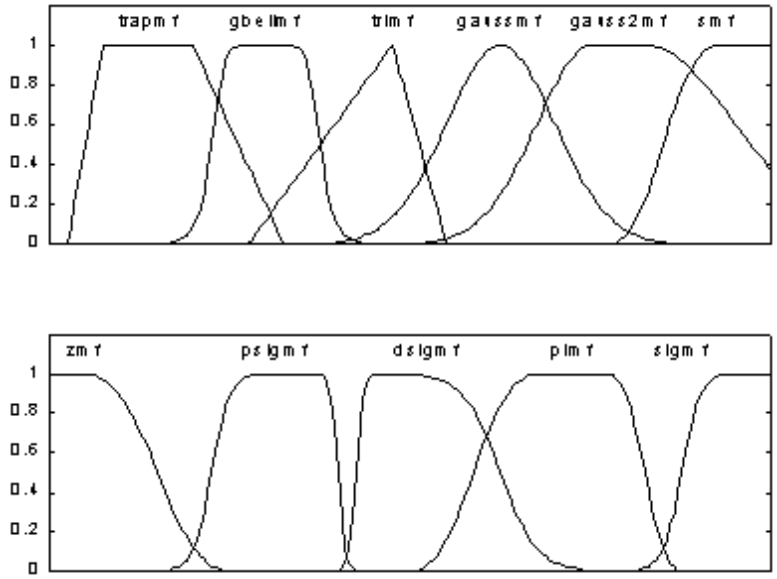
$A = \{ (x, \mu_A(x)) / x \in U\}$ şeklinde gösterilir.

Üyelik fonksiyonunun aldığı değerlere üyelik değerleri denir. $\mu_A(x)$, *x*'in bu uzaya ne kadar üye olduğunun ölçüsüdür.

$$\mu_A(x) = \begin{cases} 1 \text{ ise, x tamamen A kümesinin üyesidir.} \\ (0, 1) \text{ ise, kısmen A kümesinin üyesidir.} \\ 0 \text{ ise, A kümesinin üyesi değildir.} \end{cases}$$

Üyelik fonksiyonlarının şekli tecrübeye göre veya data toplandıktan sonra ilgili parametre ayarları yapılıp seçilir. Matlab Bulanık Toolbox'da kullanabileceğimiz başlıca üyelik fonksiyonları Şekil 2.4'de verilmiştir.

trimf(x, [1.8 3.2 4.3]);
zmf(x, [2 8]);
trapmf(x, [0.5 1 1.5 2]);
gbellmf(x, [1.5 5 2.5]);
smf(x, [8 10]);
dsigmf(x,[5 2 5 7]);
gaussmf(x, [4 6]);
sigmf(x, [2 3]);
psigmf(x, [2 3 -5 8]);
pimf(x, [2 5 8 9]);
dsigmf(x,[5 2 5 7]);
gauss2mf(x, [2 8 1 4]);



Şekil 2.4 Matlab Fuzzy Toolbox'daki hazır üyelik fonksiyonları

2.2 Bulanık Kümelerle Temel İşlemler

Bir örnekle bulanık küme kavramını açıklamaya çalışalım.

"İzmir'deki yerli arabalar" kümesi D bulanık kümesi olsun.

$\mu_D(x) = p(x)$
$p(x)$ arabasının parçalarının yerlilik oranını göstersin.

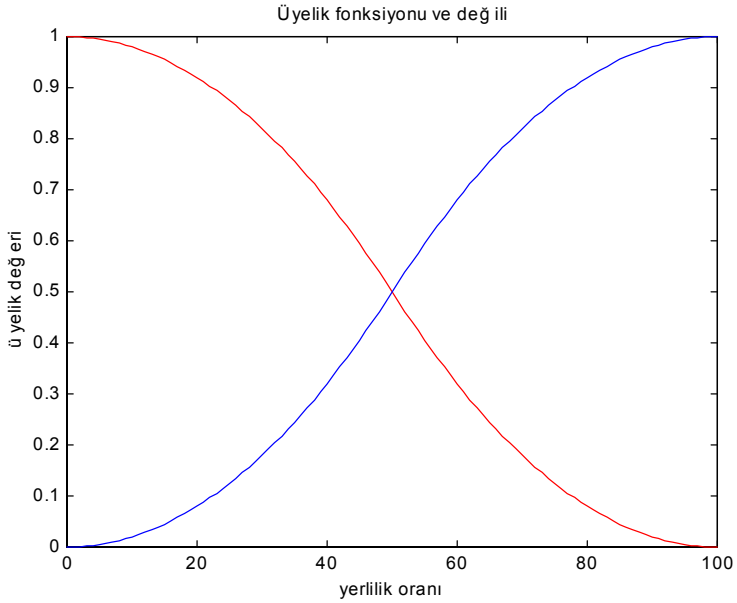
Örneğin x_1 arabası %60 yerli ise x_1 D kümesine 0.6 derecesinde üyedir.

"İzmir'deki yerli olmayan arabalar" kümesi F kümesi ile gösterilsin.

$$\mu_F(x) = 1 - \mu_D(x) \qquad (2.1)$$

olacaktır.

```
x=0:1:100;
mD=zmf(x, [0 100]);
mF=1-mD;
plot(x,mD,'r',x,mF,'b')
title('Üyelik fonksiyonu ve değili')
xlabel('yerlilik oranı')
ylabel('üyelik değeri')
```



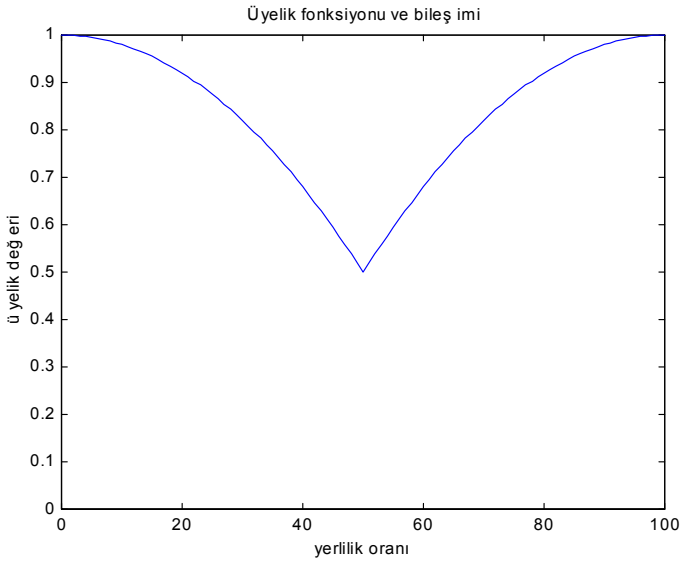
Şekil 2.5 D üyelik fonksiyonu ve değili F üyelik fonksiyonu

D ve F üyelik fonksiyonlarının birleşimi;

$$\mu_{D\cup F}(x) = \max\left[\mu_D(x), \mu_F(x)\right] \qquad (2.2)$$

şeklinde tanımlanır.

```
x=0:1:100;
mD=zmf(x, [0 100]);
mF=1-mD;
plot(x,max(mD,mF))
title('Üyelik fonksiyonu ve bileşimi')
xlabel('yerlilik oranı')
ylabel('üyelik değeri')
```



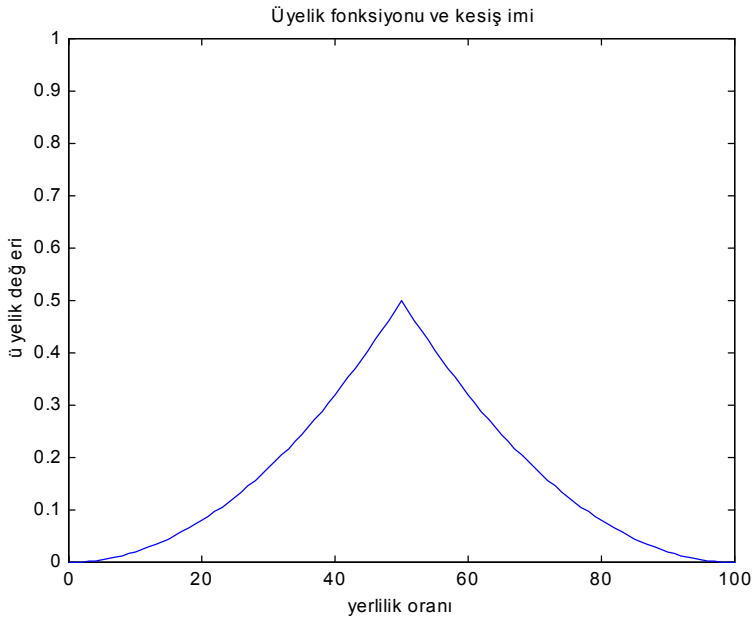
Şekil 2.6 D ve F üyelik fonksiyonlarının birleşimi

D ve F üyelik fonksiyonlarının kesişimi;

$$\mu_{D\cap F}(x) = \min\left[\mu_D(x), \mu_F(x)\right] \qquad (2.3)$$

şeklinde tanımlanır.

```
x=0:1:100;
mD=zmf(x, [0 100]);
mF=1-mD;
plot(x,min(mD,mF))
title('Üyelik fonksiyonu ve kesişimi')
xlabel('yerlilik oranı')
ylabel('üyelik değeri')
```



Şekil 2.7 D ve F üyelik fonksiyonlarının kesişimi

2.3 Bulanık Birleşim ve *S* Normları

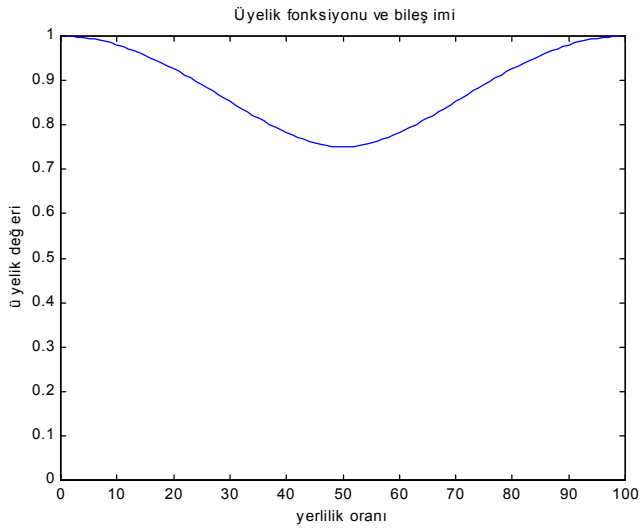
Üyelik fonksiyonlarının maksimumunun, bulanık küme birleşimi için kullanıldığını bir önceki bölümde göstermiştik. Literatürde bulanık küme birleşimi için, farklı uygulamalarda kullanılan *s* normları vardır. Farklı *s* normlarının olmasının nedeni bazı *s* normlarının bazı uygulamalarda daha anlamlı olmasıdır.

D ve F üyelik fonksiyonlarının birleşimini aritmetik toplam *s* normu kullanılarak bulunmak istenirse;

$$\mu_{D\cup F}(x) = s\left[\mu_D(x), \mu_F(x)\right]$$
$$= \mu_D(x) + \mu_F(x) - \mu_D(x)\mu_F(x) \qquad (2.4)$$

şeklinde olacaktır.

```
x=0:1:100;
mD=zmf(x, [0 100]);
mF=1-mD;
plot(x,(mD+mF-mD.*mF))
title('Üyelik fonksiyonu ve bileşimi')
xlabel('yerlilik oranı')
ylabel('üyelik değeri')
```



Şekil 2.8 Bulanık birleşim ve s normu

2.4 Bulanık Kesişim ve *T* Normları

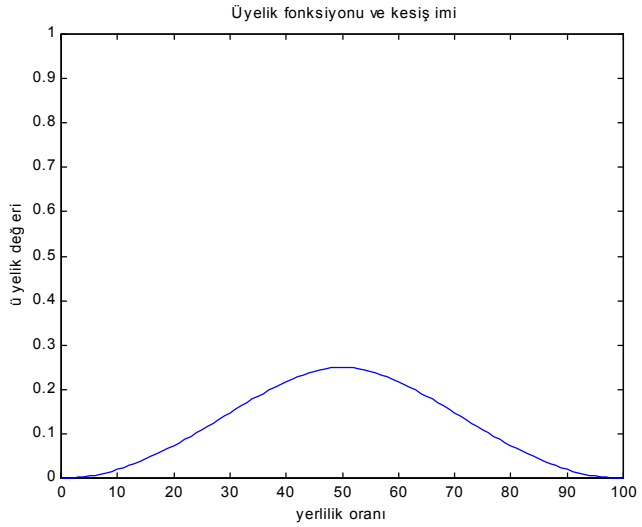
A ve B bulanık kümelerinin kesişimi için daha önce minimum fonksiyonunu kullandık. *s* normlarında olduğu gibi *t* normları da birden fazladır.

D ve F üyelik fonksiyonlarının kesişimi aritmetik toplam *t* normu kullanılarak bulunmak istenirse;

$$\mu_{D\cap F}(x) = t\left[\mu_D(x), \mu_F(x)\right] = \mu_D(x)\mu_F(x) \qquad (2.5)$$

şeklinde olacaktır.

```
x=0:1:100;
mD=zmf(x, [0 100]);
mF=1-mD;
plot(x,(mD.*mF))
title('Üyelik fonksiyonu ve kesişimi')
xlabel('yerlilik oranı')
ylabel('üyelik değeri')
```



Şekil 2.9 Bulanık kesişim ve t normu

2.5 *S* Norm ve *T* Norm Karşılaştırması

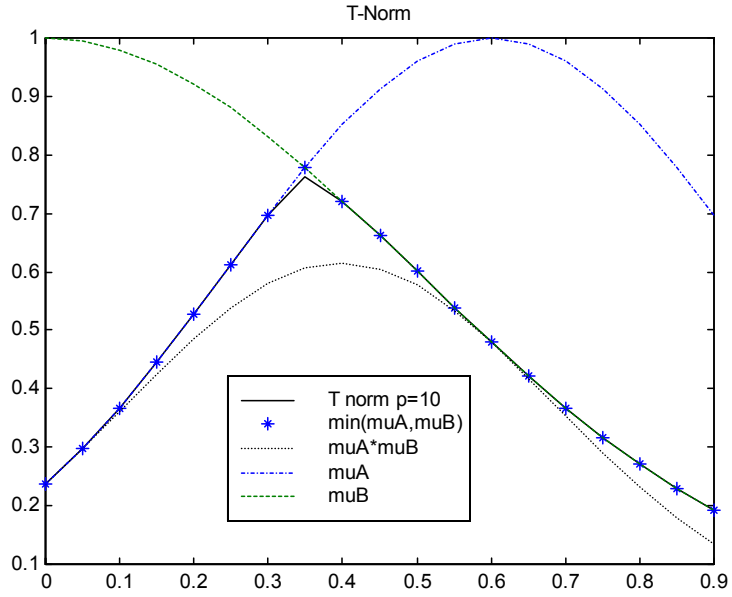
Bu bölümde *s* ve *t* normları,

s normu : [a^p+b.^p-(a.*b).^p].^(1/p)
t normu : [1-[(1-a).^p+(1-b).^p-((1-a).^p).*(1-b).^p].^(1/p)]

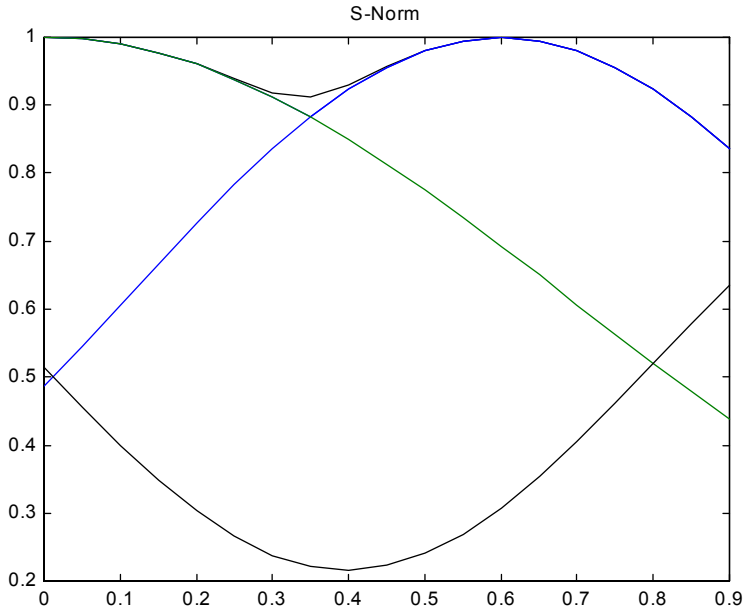
şeklinde tanımlandığında *p* parametresine göre *s* ve *t* normlarının nasıl değiştiği incelenmiştir.

```
clear
x=0:0.05:.9;
a=0:0.1:.9;
b=1-a;

mua=gaussn(x,0.6,0.5);a=mua;
mub=gaussn(x,0,0.7);b=mub;

plot(x,a,x,b),title('mA mB')
pause

for lam=0.1:0.2:4
s=(1+((1./a-1).^(-lam)+(1./b-1).^(-lam)).^(-1/lam)).^(-1);
plot(x,s,x,a,x,b),axis([0,1,0,1]),
title('S-Norm'),pause(0.1)
end

pause
close

w=3;
sw=min(1,(a.^w+b.^w).^(1/w));
plot(x,sw),title('S-Norm'),
axis([0,1,0,1])
pause

%S norm

for p=0.01:0.25:20
S=[a.^p+b.^p-(a.*b).^p].^(1/p);
plot(x,S,'k',x,max(a,b),'b',x,a,x,b,x,1-a.*b,':k'),
title('S-Norm'),pause(0.3)
end
```

```
pause

% p=1 iken S norm=1-a*b
% p=sonsuz iken S norm max(a,b) ye eşit
% S norm aUb:Bulanık birleşim


%T norm

for p=0.01:0.25:10,p
T=1-[(1-a).^p+(1-b).^p-((1-a).^p).*(1-b).^p].^(1/p);
plot(x,T,'k',x,min(a,b),'*b',x,a.*b,':k',x,a,'-.',x,b,'--'),
title('T-Norm'),pause(0.4)
legend('T norm p=10','min(muA,muB)','muA*muB','muA','muB');
end

% p=1 iken T normu a*b ye eşit
% p=sonsuz iken T normu min(a,b) ye eşit
% T norm a kesişim b :Bulanık kesişim
```



Şekil 2.10 *T normu* ve üyelik fonksiyonları



Şekil 2.24 *S normu* ve üyelik fonksiyonları

Bölüm 3

Sözsel Değişkenler ve IF-THEN Kuralları

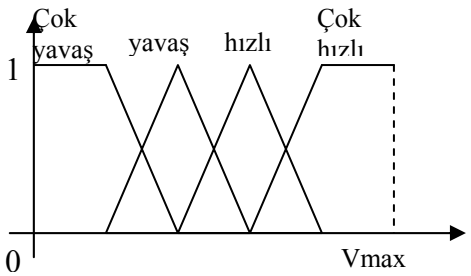
3.1 Sözsel Değişkenlerden Sayısal Değerlere

Günlük hayatta kullandığımız değişkenleri çoğunlukla kelimeler karşılar. Örneğin, bugün hava çok soğuk, dendiğinde,

sözsel değişken ; bugünkü havanın sıcaklığı
değeri ; çok soğuk

olduğu anlaşılmaktadır.

Bulanık sistemlerde kelimeler tanımlı oldukları uzayda bulanık kümelerle temsil edilirler. Bir arabanın hızının bulanık modeli çıkartılmak istendiğinde, ilk önce tanımlı olduğu uzayı belirlememiz gerekir. Bu aralık bir arabanın ulaşabileceği maksimum hızı ile minimum hızı arasındaki değerlerdir. Bu uzayı belirledikten sonra, çok yavaş, yavaş, hızlı, çok hızlı gibi sözsel değişkenleri temsil edecek bulanık kümeler belirlenir. Şekil 3.1' e bakınız.



Şekil 3.1 Sözsel bir değişken olan bir arabanın hızının bulanık kümelerle temsili

İnsan bilgi ve tecrübesini IF-THEN kuralları kullanarak sistemlerin kontrolü gerçeklenebilir. Bulanık IF-THEN kuralı;

IF < bulanık önerme >, THEN < bulanık önerme >

şeklinde ifade edilir.

U ve V gerçek uzaylarında tanımlanan *x* ve *y* sözsel değişkenlerini, A ve B bulanık kümeleri temsil etmek üzere;

"*x* A ise ve *y* B ise" önermesi A∩B şeklindeki bulanık ilişkiye ve üyelik fonksiyonları ile herhangi bir *t* normuna karşılık gelir.

$$\mu_{D\cap F}(x) = t\left[\mu_D(x), \mu_F(x)\right] \qquad (3.1)$$

"*x* A ise ya da *y* B ise" önermesi A∪B şeklindeki bulanık ilişkiye ve üyelik fonksiyonları ile herhangi bir *s* normuna karşılık gelir.

$$\mu_{DF}(x) = t\left[\mu_D(x), \mu_F(x)\right] \qquad (3.2)$$

" *x* A değil ise" önermesi A'nın değilidir. A'nın değilini E bulanık kümesi ile gösterirsek,

$$\mu_E(x) = c\ (\mu_A(x)\ ) = 1 - \mu_A(x) \qquad (3.3)$$

Örnek: " (x_1 A ise ve x_2 B değil ise) ya da x_3 C ise önermesini bulanık sistemde üyelik fonksiyonları ile ifade etmek istersek;

$$s\ [\ t\ [\ \mu_A(x_1),\ c(\mu_B(x_2))],\ \mu_C(x_3)] \qquad (3.4)$$

şeklinde olacaktır.

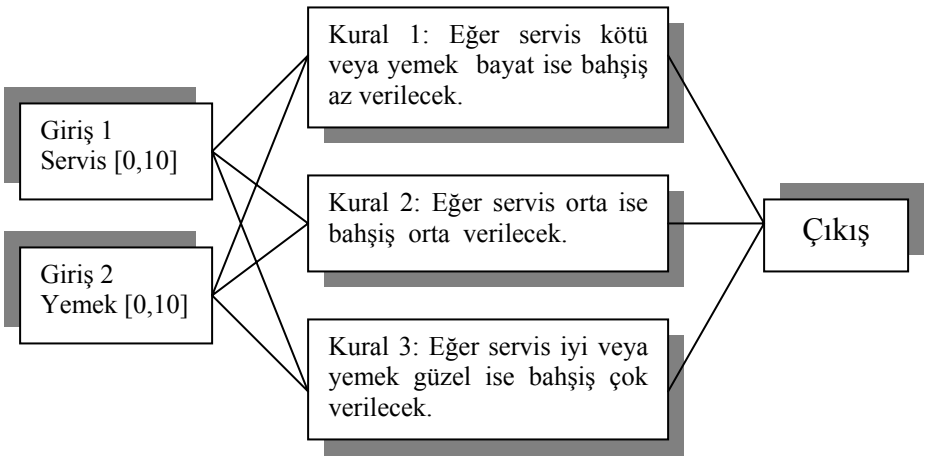
Bölüm 4

Bulanık Sonuç Çıkarma (Fuzzy Inference)

Bulanık çıkarım, bulanık mantık kullanarak girdilerden çıktı elde etme işlemidir. Bulanık çıkarım daha önceki bölümlerde anlatılan bulanık üyelik fonksiyonlarının, üyelik değerlerinin, IF-THEN kurallarının, bulanık mantık operatörlerinin kullanıldığı bir işlemdir. Bir çok çıkarım metodu vardır. Aşağıda Mamdani'nin Minimum çıkarım metodu verilmiştir.

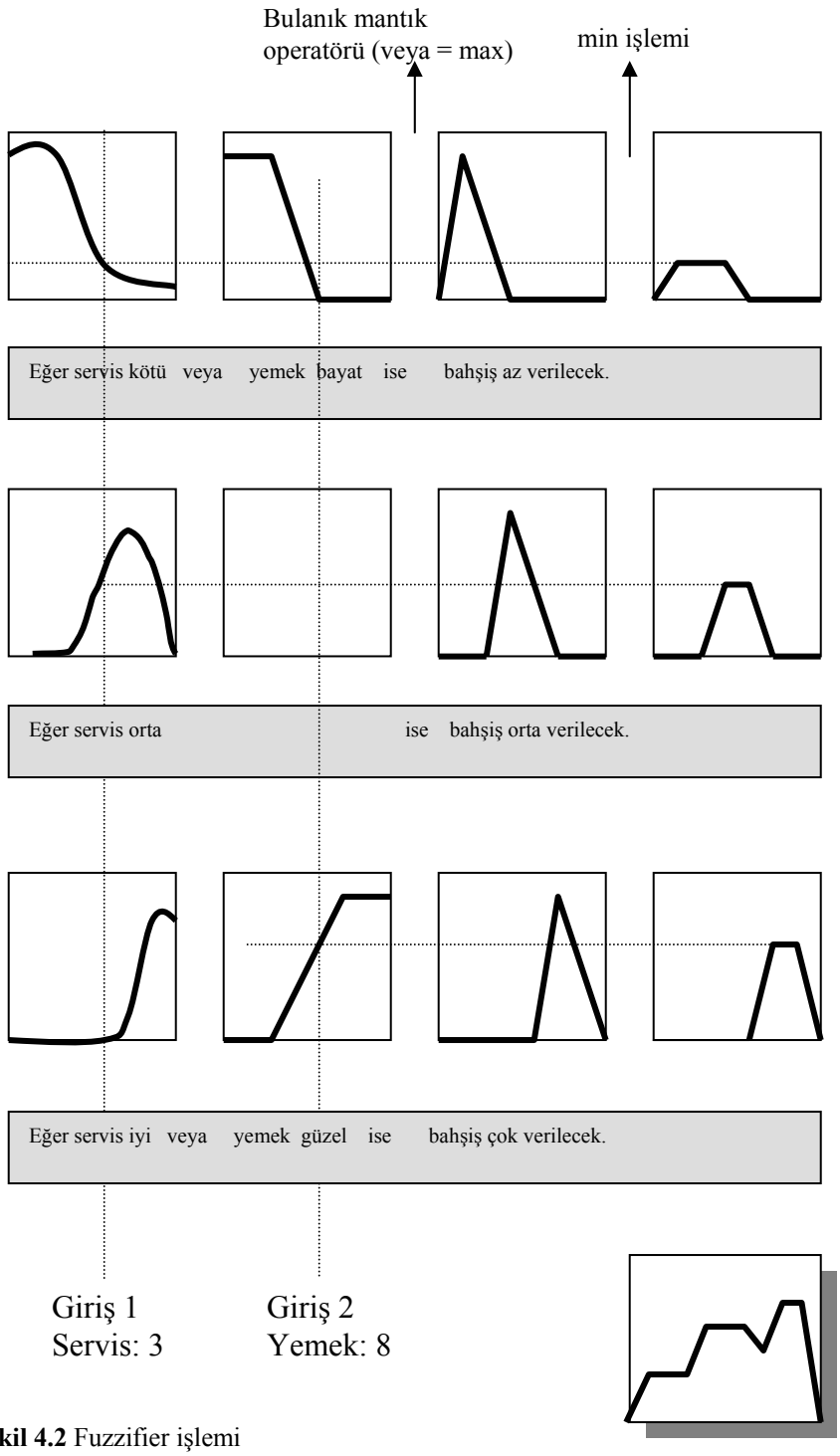
$$\mu_{B'} = \max_{I=1}^{M} \left[\min\left(\mu_{A_1'}(x), \mu_{A_2'}(x), ..., \mu_{A_n'}(x), \mu_{B'}(x) \right) \right] \quad (4.1)$$

İki girişli, bir çıkışlı ve üç kurallı bir sistem üzerinde izlenmesi gereken adımları açıklamaya çalışalım.



Şekil 4.1 İki girişli tek çıkışlı bulanık kural tabanı

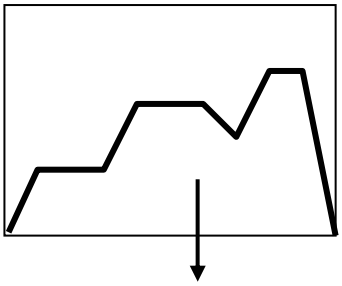
4.1 Bulanıklaştırıcı (Fuzzifier)



Şekil 4.2 Fuzzifier işlemi

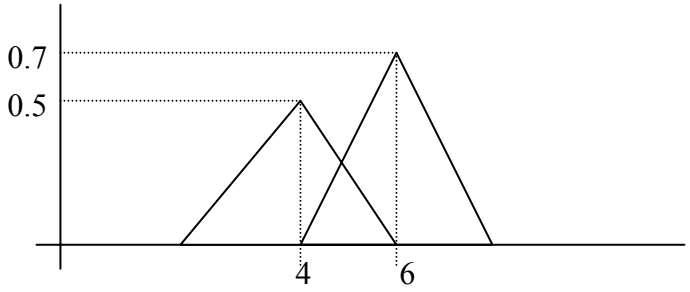
4.2 Durulayıcı (Defuzzifier)

Durulama işlemi için en çok kullanılan yöntem eğriler altında kalan alanın ağırlık merkezinin bulunmasıdır.



Şekil 4.3 Defuzzifier işlemi

Örnekler üzerinde Durulama kavramını daha iyi anlamaya çalışalım. Burada sadece Center of Gravity ve Center of Average Defuzzifier yöntemlerinden bahsedilecektir.



Şekil 4.4 Center of Average Defuzzifier işleminin grafiksel gösterimi

Center of Average defuzzifier işleminde, durulanmış y^* değeri

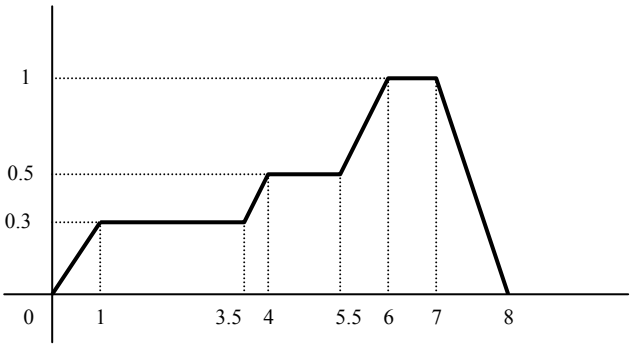
$$y^* = \frac{\sum_{l=1}^{M} \bar{y}^{l} w_l}{\sum_{l=1}^{M} w_l} \qquad (4.2)$$

şeklinde hesaplanır.

Şekil 4.4'deki verilen problem için M=2'dir. w_l üyelik fonksiyonlarının maksimum değerlerini, $\bar{y}^{l}$ ise üyelik fonksiyonlarının merkezini göstermektedir. Buna göre;

$$y^* = \frac{4*0.5. + 6*0.7}{0.5 + 0.7} = 5.166 \qquad (4.3)$$

şeklinde durulanmış değer elde edilir.



Şekil 4.5 Center of Gravity Defuzzufier işleminin grafiksel gösterimi

Center of Gravity Defuzzifier işleminde, durulanmış y^* değeri,

$$y^* = \frac{\int_v y \mu_{B'}(y) dy}{\int_v \mu_{B'}(y) dy} \qquad (4.4)$$

şeklinde hesaplanır. Buna göre gerekli integraller alınıp durulanmış y^* değeri hesaplanır. Şekil 4.5'de verilen çıkış üyelik fonksiyonu center of gravity metodu kullanarak durulamak istersek;

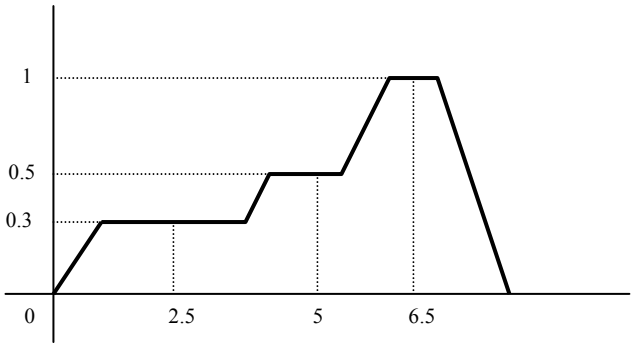
$$y^* = \frac{\int_v y\mu_{B'}(y)dy}{\int_v \mu_{B'}(y)dy} = \frac{\int_0^1 y(0.3y)dy + \int_1^{3.5} y(0.3)dy + \int_{3.5}^4 y((y-3)/2)dy + \dots}{\int_0^1 (0.3y)dy + \int_1^{3.5} (0.3)dy + \int_{3.5}^4 ((y-3)/2)dy + \dots}$$

$$\frac{\int_4^{5.5} y(0.5y)dy + \int_{5.5}^6 y(y-5)dy + \int_6^7 ydy + \int_7^8 y(8-y)dy}{\int_4^{5.5} (0.5y)dy + \int_{5.5}^6 (y-5)dy + \int_6^7 dy + \int_7^8 (8-y)dy} = 4.9$$

(4.5)

şeklinde bulunur.

Şekil 4.5'deki çıkış üyelik fonksiyonunu durulama işlemi için center average defuzzifier metodunu kullanırsak;

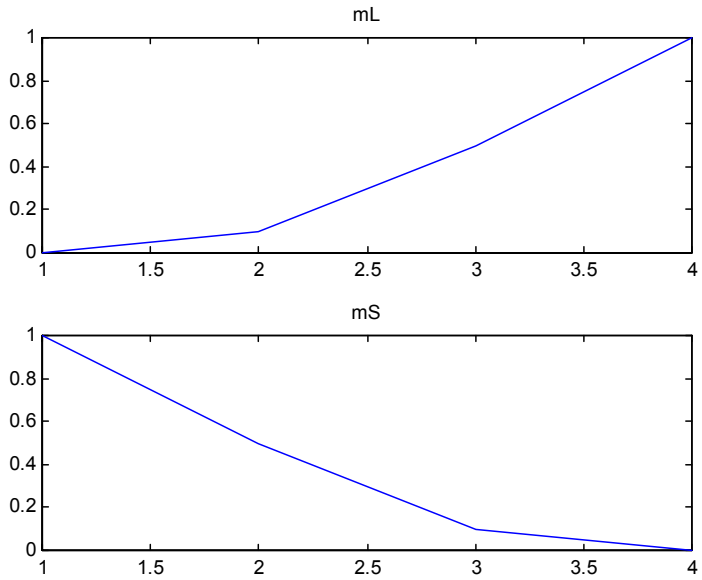


Şekil 4.6 Center average defuzzifier

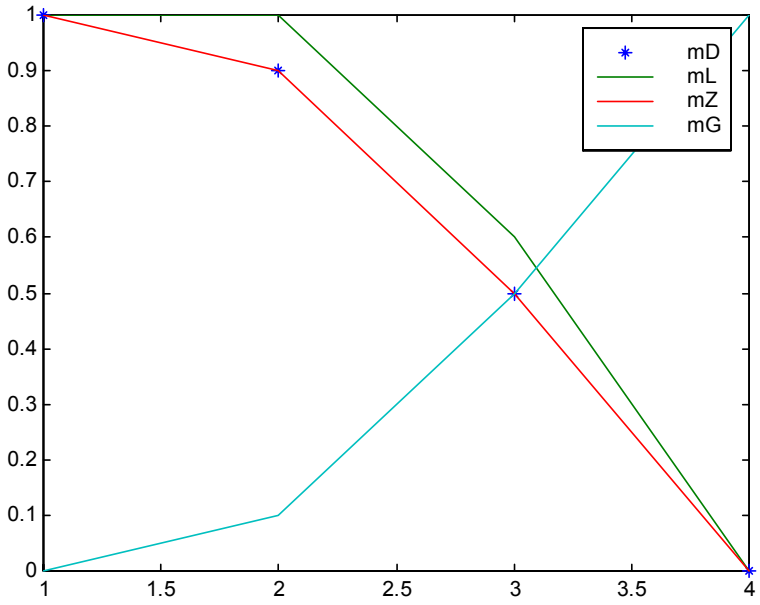
$$y^* = \frac{\sum_{l=1}^{M} \bar{y}^l w_l}{\sum_{l=1}^{M} w_l} = \frac{2.5x0.3 + 5x0.5 + 6.5x1}{0.3 + 0.5 + 1} = 5.41 \qquad (4.6)$$

4.3 Çıkarım Metotlarına Bir Örnek

```
U=1:4;
V=1:4;

L=[0 0.1 0.5 1];
S=[1 0.5 0.1 0];

subplot(2,1,1);plot(U,L);title('mL')
subplot(2,1,2);plot(V,S);title('mS')
pause

mD=max(1-L,S);
mL=min(1,1-L+S);
mZ=max(min(L,S),1-L);
mG=L;mg=L<S;
mMP=L'*S;
close
plot(U,mD,'*',U,mL,U,mZ,U,mG);
legend('mD','mL','mZ','mG')
```



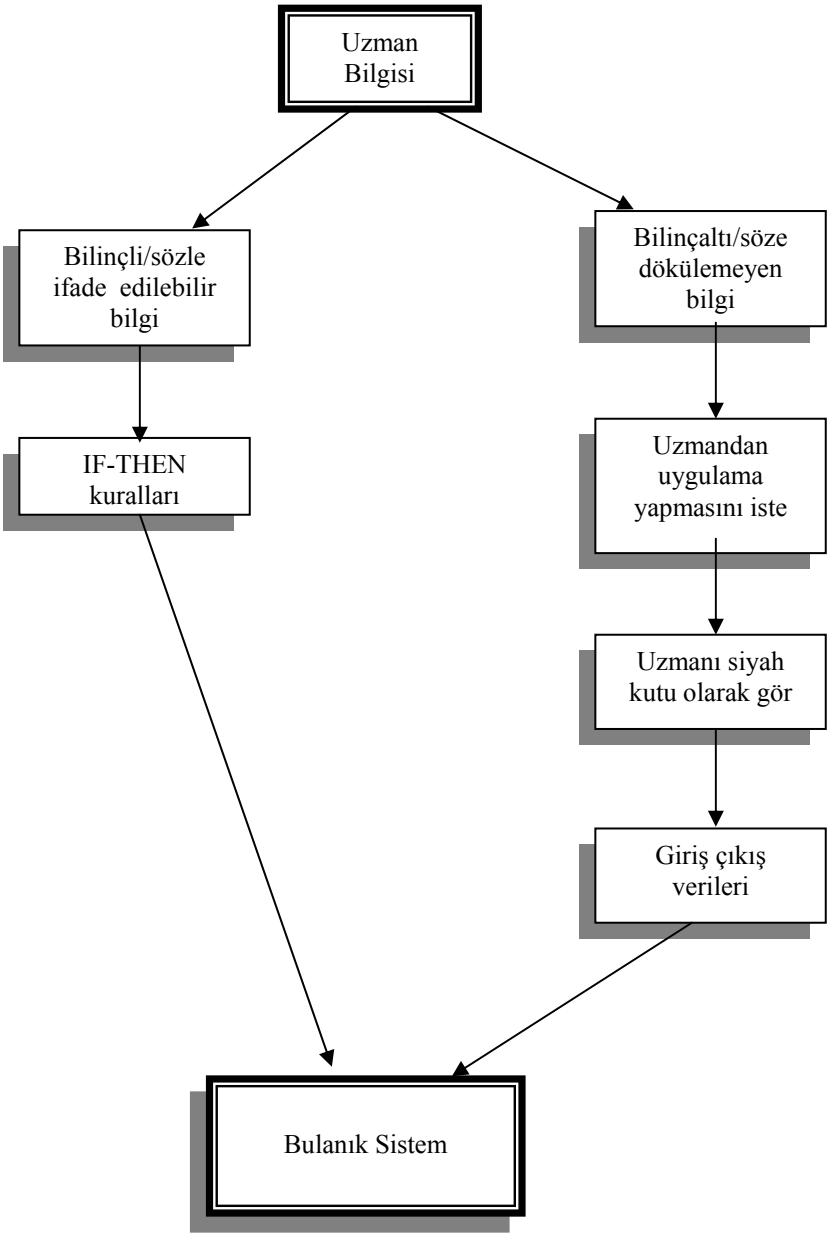
Şekil 4.7 *mL* ve *mS* üyelik fonksiyonları



Şekil 4.8 Çıkarım örnekleri (mL : Larsen, mZ : Zadeh)

Bölüm 5

Giriş Çıkış Bilgisinden Bulanık Sistem Tasarımı



5.1 Table Look-Up Kullanarak Bulanık Küme Tasarımı

Girdileri ve çıktıları bilinen bir sistem için bulanık sistem tasarlanabilir. Oluşturulan bulanık sistem tanımlanan sistemin yerini alabilir ve yeni girdilerin çıktılarını bulmada kullanılabilir.

$(x_0^p ; y_0^p)$ p =1, 2, ……. N veriliyor ya da ölçülüyor.

$$x_0^P \in U = [\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times ..... \times [\alpha_n, \beta_n] \qquad (5.1)$$

$$y_0^p \in V = [\alpha_y, \beta_y] \qquad (5.2)$$

Table look-up metodunda izlenmesi gereken kurallar:

Adım 1

Herbir $[\alpha_i , \beta_i]$ i =1, 2, 3, .. n için A_I^j (j = 1, 2, 3, …N_i) bulanık kümeleri ve benzer şekilde B^j (j = 1, 2, 3, …. N_y) bulanık kümesi [α_y, β_y] 'de tanımlanmalıdır.

Örnek:
n = 2, N_1 = 5, N_2 = 7 ve N_y = 5 gi bi bir sistem tanımlandığında;

Giriş sayısı 2, birinci ve ikinci giriş için sırasıyla 5 ve 7, çıkış için 5 bulanık üyelik fonksiyonu/kümesi olduğu anlaşılmaktadır.

Adım 2

Her $(x_{o1}^p, x_{o2}^p, ...x_{on}^p ; y_o^p)$ giriş / çıkış çifti için;
x_{oi}^p 'lerin A_l^j (j = 1, 2, 3, .. N_i) bulanık kümelerdeki ve benzer şekilde y_o^p 'lerin B^l (l = 1, 2, 3, …. N_y)'deki üyelik değerleri belirlenmelidir.

x ve y gerçek değerleri fuzzifier edilirken aynı anda farklı bulanık kümelerde değer alabilirler. Bu durumda en büyük üyelik değerine sahip bulanık küme seçilir ve daha sonra IF-THEN kuralları oluşturulur.

Adım 3

Aynı IF kısmına fakat farklı THEN kısmına sahip kuralları elemek ve kural sayısını azaltmak için kural derecesini belirleyip en büyük dereceye sahip kurallar bulanık sistemi oluşturmak üzere seçilir.

$(x_0^p ; y_0^p)$ giriş çıkış çifti için üyelik değeri:

$$D(kural) = \prod \mu_{A_i^{j*}}(x_{0i}^p)\mu_{B^{l*}}(y_0^p) \qquad (5.3)$$

şeklinde hesaplanır.

Adım 4

Elde edilen kurallara uzman bilgisi de var ise eklenmelidir.

İki girişli tek çıkışlı bir bulanık sistem düşünelim. Her iki giriş ve çıkış için 4 üyelik fonksiyonu tanımlansın. Bu adıma kadar elde edilen kuralları tabloya taşıyabilir ve kuralları daha açık görebiliriz.

S1, S2, S3, S4 birinci giriş (x1) için üyelik fonksiyonları, T1, T2, T3, T4 ikinci giriş (x2) için üyelik fonksiyonları, C1, C2, C3, C4 çıkış (y) için üyelik fonksiyonları olsun.

T4	C3			
T3			C2	
T2		C1		
T1				C4
	S1	S2	S3	S4

Şekil 5.1 Bulanık kural tabanı için table look-up

Şekil 5.1'deki tabloya bakarak bu sistem için aşağıdaki kuralı yazabiliriz.
IF x1 is S1 and x2 is T4 THEN y is C3
vb.

Adım 5

Elde edilen bulanık kurallar kullanılarak bulanık sistem Denklem (5.9) deki gibi tanımlanır.

5.2 Table Look-Up Kullanarak Bulanık Küme Tasarımı ile Uygulamalar

5.2.1 Zaman Serisi Tahmini

Mackey-Glass kaotik zaman serisi $\tau > 17$ iken,

$$d\,x(t) / dt = ((0.2\,x(t - \tau)) / (1 + x^{10}(t - \tau))) - 0.1\,x(t) \qquad (5.4)$$

diferansiyel denklemi ile üretilir.

1 sn'lik aralıklarla 600 nokta için yukarıdaki diferansiyel denklem çözülüp zaman serini üretildikten sonra, ilk 300 nokta bulanık sistemi tasarlamak için; giriş çıkış bilgisi olarak tanımlanır. Bu örnekte 7 bulanık üyelik fonksiyonlu, 4 girişli 1 çıkışlı bulanık sistem tasarlanmıştır. Bulanık sistem tasarlandıktan sonra geriye kalan 300 nokta bulanık sistem ile hesaplanır.

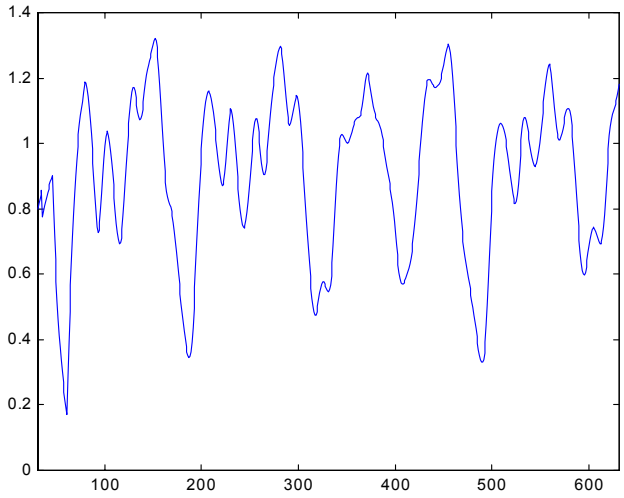
k = 1, 2, 3, ..600 örnekleme sayısı
n = 4 giriş sayısı olmak üzere zaman serisi tahminleme problemini aşağıdaki gibi formülize edebiliriz.

giriş : x(k-n+1, x(k-n+2), ..., x(k))
çıkış : x(k+1) çıkış

Mackey-Glass kaotik zaman serisini üreten diferansiyel denkleminin τ=30 için çözümü.

```
h=1;
x(1:31)=zeros(1,31);
x(1:15)=ones(1,15);
x(5)=100000;
x(23)=12220;
x(31)=0.8;
t(1:31)=0:h:30;
t(31)=31;
i=31;
j=1;

for n=1:1:600;

k1(j)=h*(((0.2*x(i-30))/(1+x(i-30)^10))-0.1*x(i));
```

```
xk1i=x(i)+0.5*k1(j);
xk13=x(i-30)+0.5*k1(j);
k2(j)=h*((0.2*xk13/(1+xk13^10))-0.1*xk1i);
xk2i=x(i)+0.5*k2(j);
xk23=x(i-30)+0.5*k2(j);
k3(j)=h*((0.2*xk23/(1+xk23^10))-0.1*xk2i);
xk3i=x(i)+k3(j);
xk33=x(i-30)+k3(j);
k4(j)=h*((0.2*xk33/(1+xk33^10))-0.1*xk3i);
x(i+1)=x(i)+(1/6)*(k1(j)+2*k2(j)+2*k3(j)+k4(j));
t(i+1)=t(i)+h;
i=i+1;
j=j+1;

end

plot(t(31:631),x(31:631))
axis([31 631 0 1.4])
```



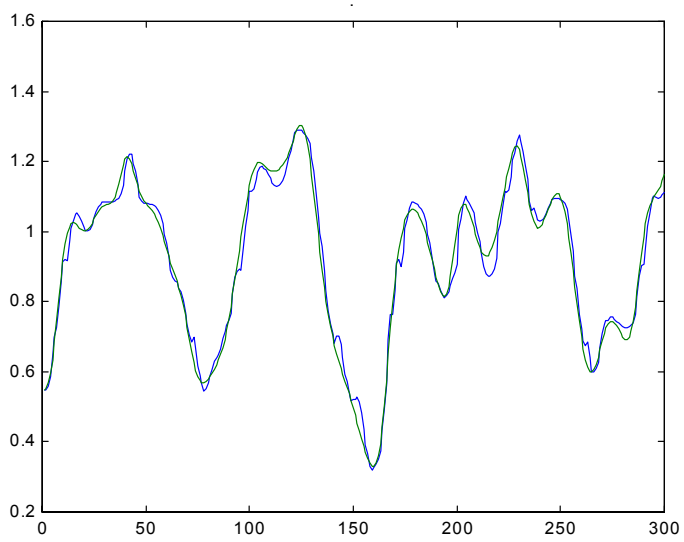
Şekil 5.2 Mackey-Glass kaotik zaman serisinden bir kesit

Elde edilen ilk 300 nokta ile oluşturulan bulanık sistem ile geriye kalan 300 nokta hesaplayacak Matlab kodu aşağıda verilmiştir.

```
x=0:0.1:1.6;
y1=trimf(x, [0 0.2 0.4]);
y2=trimf(x, [0.2 0.4 0.6]);
y3=trimf(x, [0.4 0.6 0.8]);
y4=trimf(x, [0.6 0.8 1]);
y5=trimf(x, [0.8 1 1.2]);
y6=trimf(x, [1 1.2 1.4]);
y7=trimf(x, [1.2 1.4 1.6]);
ymrk=[0.2 0.4 0.6 0.8 1 1.2 1.4];

y=[y1;y2;y3;y4;y5;y6;y7];

load xdata
n=1;

for i=1:1:296;

data(n,:)=[xdata(i) xdata(i+1) xdata(i+2) xdata(i+3) xdata(i+4)];
n=n+1;
end

for n=1:296;

for i=1:5;

for k=1:7;
mu(k)=interp1(x,y(k,:),data(n,i));
end

mux(n,i)=max(mu);
kmux(n,i)=find(mux(n,i)==mu);

end

end

for i=1:296;
D(i)=mux(i,1)*mux(i,2)*mux(i,3)*mux(i,4)*mux(i,5);
end

for n=1:295;
T=kmux(n,:)*ones(5,1);
```

```
if T = 0;
i=n+1;
p=1;

for s=i:296;
fark=kmux(n,1:4)-kmux(s,1:4);

if fark==0;
aynk(p)=s;
p=p+1;
end

end
[a r]=size(aynk);

for q=1:r;
aynkD(q,:)=[aynk(q) D(aynk(q))];
end

aynkD(r+1,:)=[n D(n)];
makDkr=max(aynkD(:,2));

for q=1:r+1;
if aynkD(q,2) < makDkr;
kmux(aynkD(q,1),:)=zeros(1,5);
end
end
end
aynk=[];
aynkD=[];
end
kurallar=find(kmux(:,1)~=0);
kn=kurallar;
[r c]=size(kn);
n=1;

for i=297:1:596;
prdata(n,:)=[xdata(i) xdata(i+1) xdata(i+2) xdata(i+3) xdata(i+4)];
n=n+1;
end
A(1)=0;
B(1)=0;

for s=1:300;
q=2;
for i=1:r;
```

```
a=kmux(kn(i),1);
b=kmux(kn(i),2);
c=kmux(kn(i),3);
d=kmux(kn(i),4);
mrk=kmux(kn(i),5);
pmux1=interp1(x,y(a,:),prdata(s,1));
pmux2=interp1(x,y(b,:),prdata(s,2));
pmux3=interp1(x,y(c,:),prdata(s,3));
pmux4=interp1(x,y(d,:),prdata(s,4));
uy=[pmux1 pmux2 pmux3 pmux4];
A(q)=(ymrk(mrk)*min(uy)) + A(q-1);
B(q)=min(uy)+B(q-1);
q=q+1;
end

f(s)=A(r+1)/B(r+1);
end

p=1:300;
plot(p,f,':',p,xdata(301:600))
```

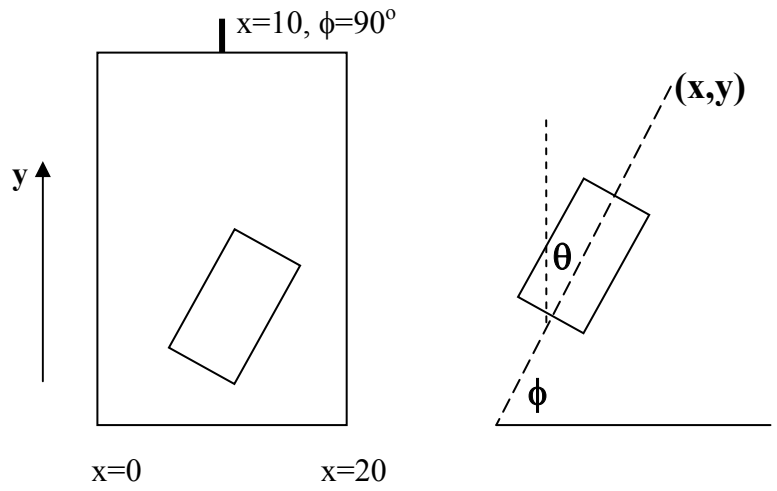


Şekil 5.3 Zaman serisinin gerçek ve bulanık sistem ile hesaplanan değerleri

5.2.2 Araç Park Uygulaması

Bir otomobilin geri geri park etmesi doğrusal olmayan bir kontrol problemidir. Geleneksel kontrol uygulamalarında ilk adım olarak sistemin matematiksel modeli çıkartılır, daha sonra da kontrol edici tasarlanır. Diğer bir yaklaşımda ise bir sürücünün davranışını yansıtacak kontrol edici söz konusudur. Burada ikinci yaklaşım bulanık mantık kullanılarak ele alınacaktır. Bu metotta, bir sürücünün aracı park etmesi sırasında aracın pozisyonuna göre davranması ve kontrol etmesi giriş çıkış verisi olarak toplanır ve bu veri ile bulanık mantık sistemi tasarlanır. Sürücünün yapması gerekenler bulanık mantık sistemine öğretilir.

Simüle edilecek araç ve park yeri Şekil 5.4'de verilmiştir.



Şekil 5.4 Simüle edilen araç ve park yeri

Aracın pozisyonu ϕ, x ve y üç durum değişkeni ile belirlenmiştir. ϕ aracın yatay ile yaptığı açıdır. Aracın kontrolü direksiyon açısı θ ile sağlanmaktadır. Aracı istenilen yere götürme esnasında sadece geri gitmeğe izin verilmektedir. Sistemi basitleştirmek için araç ile park yeri arasındaki mesafenin yeterli olduğu düşünülerek y değişkeni durum değişkeni olarak alınmamıştır. Bulanık mantık sistemi tasarlanırken giriş değişkenleri olarak (x, ϕ), çıkış değişkeni olarak da θ alınmıştır.

Son durum değişkeni Şekil 5.4'den görüldüğü üzere (x_f, ϕ_f) = (10, 90) değerlerini alacaktır. Simülasyon için durum değişkenlerinin değişim aralıkları $x \in [0,20]$, $\phi \in [-90,270]$ ve $\theta \in [-40,40]$ şeklindedir.

Bulanık mantık sistemini tasarlarken aşağıdaki adımlar takip edilmiştir.

Adım 1

İlk adımda (x, ϕ^p; θ^p) giriş-çıkış çiftlerini oluşturmak gerekir. Bu veriler uzman bilgisi ile oluşturulabilir. Çıkış değişkeni sürücü tecrübesine bağlı olarak belirlenir. Bu uygulamada simülasyon için Wang ve Mendel'in [1992b] matematiksel modeli kullanılacaktır.

$$x(t+1) = x(t) + \cos[\phi(t) + \theta(t)] + \sin[\theta(t)]\sin[\phi(t)] \qquad (5.5)$$

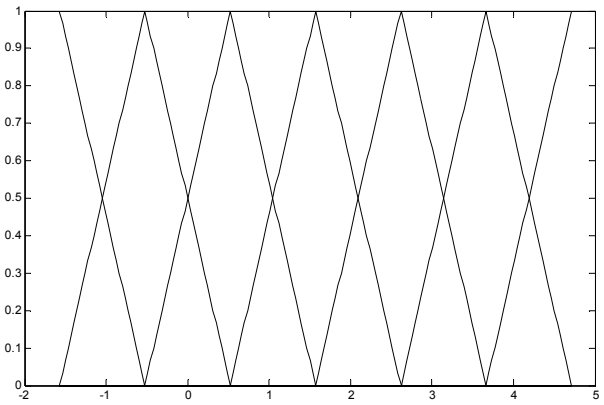
$$y(t+1) = y(t) + \sin[\phi(t) + \theta(t)] - \sin[\theta(t)]\cos[\phi(t)] \qquad (5.6)$$

$$\phi(t+1) = \phi(t) - \sin^{-1}\left[\frac{2\sin[\theta(t)]}{b}\right] \qquad (5.7)$$

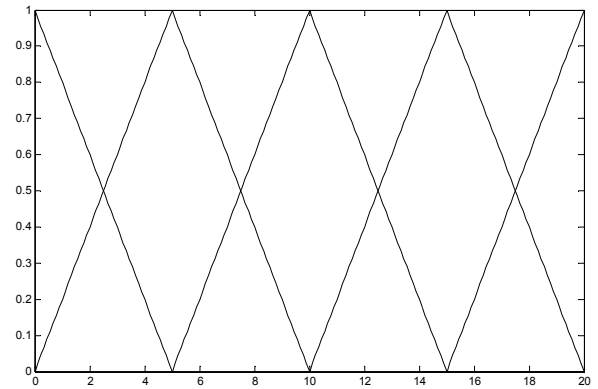
burada b aracın uzunluğu ve b = 4 inç alınmıştır. Aracın harekete başlangıç yeri rastgele seçilecektir.

Adım 2

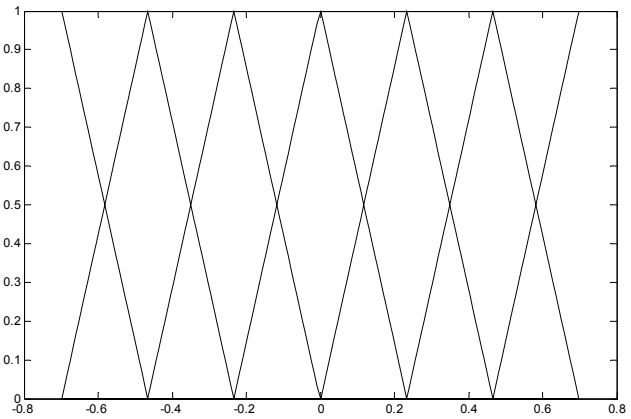
Giriş ve çıkış değişkenlerine ait üyelik fonksiyonları belirlenir. Bu problemde ϕ için 7, x için 5 ve θ için 7 üyelik fonksiyonu belirlenmiştir. Şekil 5.5'e bakınız. ϕ ve θ rad cinsinden ifade edilmiştir. (ϕ (rad) = ϕ (derece) * pi/180).



a) ϕ girişi için üyelik fonksiyonları



b) x girişi için üyelik fonksiyonları



c) θ çıkışı için üyelik fonksiyonları

Şekil 5.5 Araç park probleminde giriş ve çıkış için üyelik fonksiyonları

Adım 3

Üyelik fonksiyonlarını belirledikten sonra uzman bilgisi kullanarak Şekil 5.6'daki giriş-çıkış kural tablosu oluşturulur.

ϕ	**S3**	*S2*	*S3*			
	S2	*S2*	*S3*	*S3*	*S3*	
	S1	*B1*	*S1*	*S2*	*S3*	*S2*
	CE	*B2*	*B2*	*CE*	*S2*	*S2*
	B1	*B2*	*B3*	*B2*	*B1*	*S1*
	B2		*B3*	*B3*	*B3*	*B2*
	B3				*B3*	*B2*
		S2	**S1**	**CE**	**B1**	**B2**
	X					

Şekil 5.6 Araç park problemi için kural tablosu

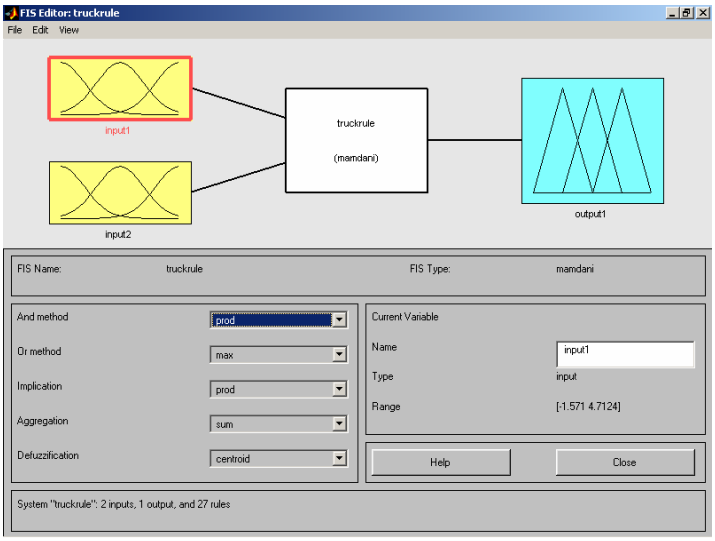
Adım 4

Adım 3'de kural tablosunu oluşturduktan sonra, kullanılacak bulanık mantık sistemini seçmek gerekir. Denklem (5.8)'de verilen form ve Şekil 5.6'daki kural tablosu kullanılarak bulanık mantık sistemi tasarımı tamamlanır.

$$f(x) = \frac{\sum_{l=1}^{M} y^{-l} \left[\prod_{i=1}^{n} \mu_{A_i^l}(x_i) \right]}{\sum_{l=1}^{M} \left[\prod_{i=1}^{n} \mu_{A_i^l}(x_i) \right]} \qquad (5.8)$$

Bu problem için de 5.2.1'de zaman serisi problemi için verilen matlab koduna benzer kod yazılabilir. Biz burada hem kolaylık olması hem de Matlab Fuzzy Toolbox'ını tanıtma amacıyla FIS editörünü kullanacağız. Şekil 5.7'ye bakınız.

Bunun için Matlab komut satırında *fuzzy* komutunu yazıp FIS editör penceresini açıyoruz.



Şekil 5.7 FIS editörü

Giriş değişkenimiz x ve ϕ olmak üzere iki tane giriş değişkeni olduğundan edit menüsünden bir tane daha giriş değişkeni eklenir. Daha sonra her iki değişkenin üyelik fonksiyonlarının belirlenmesi için *input1* ve *input2* kutularına çift tıklanır. Giriş aralıkları dikkate alınarak kaç adet üyelik fonksiyonu kullanılacağına karar verilir ve atanır. Çıkış değişkeni için de üyelik fonksiyonları atandıktan sonra edit menüsünden *rule* menüsüne girilerek Şekil 5.6'da verilen kural tablosuna göre kurallar oluşturulur. Bulanık mantık sistemi için uygulanacak durulama yöntemi seçildikten sonra FIS editörü kapatılır ve bir isim verilir. Burada *truckrule* ismi verilmiştir. Aşağıda *truckrule* ismindeki FIS editöründe tasarlanan sistemin özellikleri görülmektedir.

[System]
Name='truckrule'
Type='mamdani'
Version=2.0
NumInputs=2
NumOutputs=1
NumRules=27
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='input1'
Range=[-1.571 4.7124]
NumMFs=7
MF1='s3':'trimf',[-2.618 -1.571 -0.524]
MF2='s2':'trimf',[-1.571 -0.524 0.5239]
MF3='s1':'trimf',[-0.524 0.5239 1.57]
MF4='ce':'trimf',[0.5239 1.57 2.618]
MF5='b1':'trimf',[1.57 2.618 3.664]
MF6='b2':'trimf',[2.618 3.664 4.712]
MF7='b3':'trimf',[3.664 4.712 5.761]

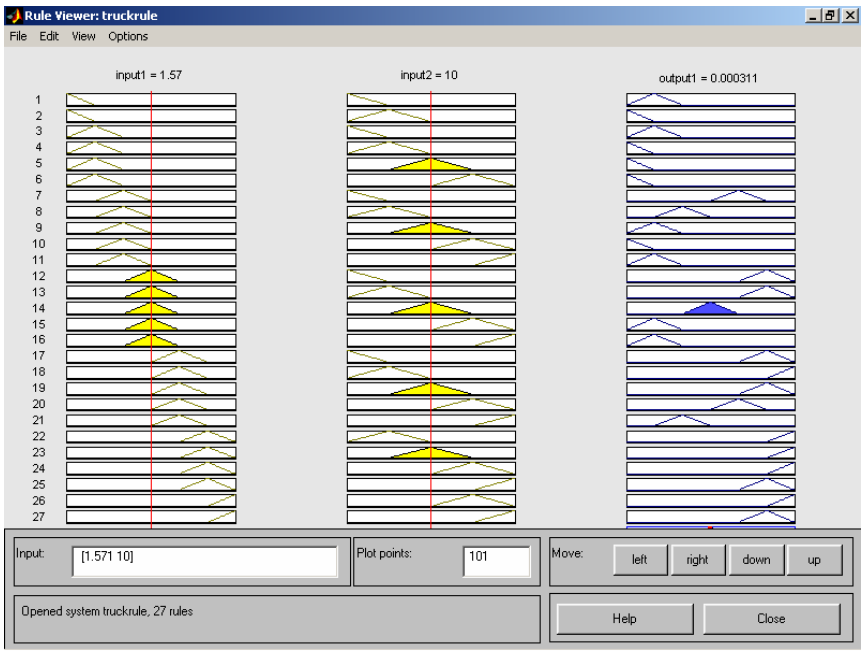
[Input2]
Name='input2'
Range=[0 20]
NumMFs=5
MF1='s2':'trimf',[-5 -5.551e-017 5]
MF2='s1':'trimf',[0 5 10]
MF3='ce':'trimf',[5 10 15]
MF4='b1':'trimf',[10 15 20]
MF5='b2':'trimf',[15 20 25]

[Output1]
Name='output1'
Range=[-0.6981 0.6981]
NumMFs=7
MF1='s3':'trimf',[-0.9307 -0.6981 -0.4655]
MF2='s2':'trimf',[-0.6981 -0.4655 -0.2326]
MF3='s1':'trimf',[-0.4655 -0.2326 0]
MF4='ce':'trimf',[-0.2326 0 0.2326]
MF5='b1':'trimf',[0 0.2326 0.4655]
MF6='b2':'trimf',[0.2326 0.4655 0.6981]
MF7='b3':'trimf',[0.4655 0.6981 0.9307]

[Rules]
1 1, 2 (1) : 1
1 2, 1 (1) : 1
2 1, 2 (1) : 1
2 2, 1 (1) : 1
2 3, 1 (1) : 1
2 4, 1 (1) : 1
3 1, 5 (1) : 1
3 2, 3 (1) : 1
3 3, 2 (1) : 1
3 4, 1 (1) : 1
3 5, 2 (1) : 1

4 1, 6 (1) : 1
4 2, 6 (1) : 1
4 3, 4 (1) : 1
4 4, 2 (1) : 1
4 5, 2 (1) : 1
5 1, 6 (1) : 1
5 2, 7 (1) : 1
5 3, 6 (1) : 1
5 4, 5 (1) : 1
5 5, 3 (1) : 1
6 2, 7 (1) : 1
6 3, 7 (1) : 1
6 4, 7 (1) : 1
6 5, 6 (1) : 1
7 4, 7 (1) : 1
7 5, 6 (1) : 1

Şekil 5.8'de giriş ve çıkış değişkenlerine ait üyelik fonksiyonları ve kural ilişkileri görülmektedir.

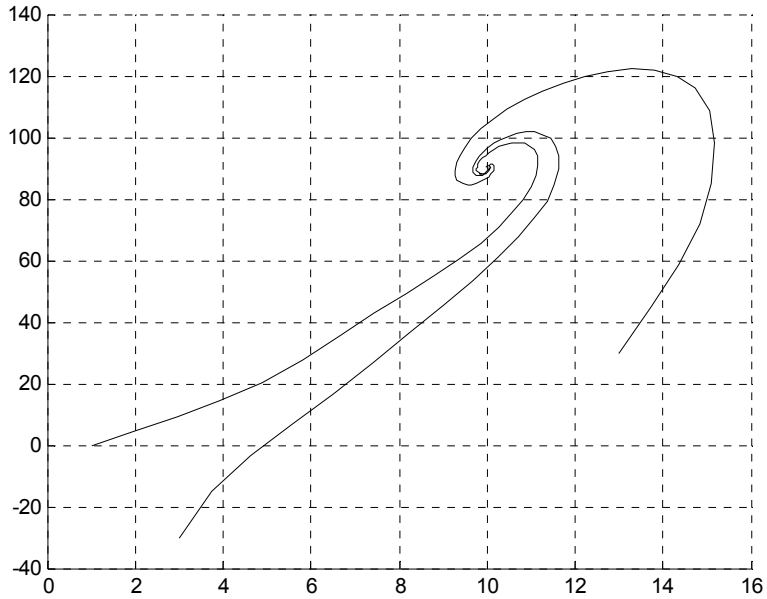


Şekil 5.8 FIS editörü ve kurallar

Bu adımlardan sonra aşağıdaki Matlab kodu ile sistem tasarımı tamamlanır.

```
% aracın başlangıç yeri
x(1)=1;
phi(1)=30*pi/180;
%
b=4;
fis1=readfis('truckrule'); % FIS editörde tasarlanan sistemin tanıtılması

for n=1:50;
out1=evalfis([phi(n) x(n)],fis1);
theta(n)=out1;

x(n+1)=x(n)+cos(phi(n)+theta(n))+sin(theta(n))*sin(phi(n));
phi(n+1)=phi(n)-asin((2/b)*sin(theta(n)));
end
```

Başlangıç yeri (1,0), (13,30) ve (3,-30) seçildiğinde *y* değişkeni ihmal edildiğinden aracın ϕ açısının x değişkenine göre değişimi Şekil 5.9'de verilmiştir.



Şekil 5.9 Araç park ederken ϕ değişkeninin x değişkenine göre değişimi

5.3 Gradyen Tabanlı Eğitme (Gradient Descent Training) ile Bulanık Sistem Tasarımı

"İnekler gradyen hesaplayamazlar, ama yine de vadinin dibini bulabiliyorlar."

Table – look up metodunda üyelik fonksiyonları giriş çıkış çiftlerine bağlı değildir. İlk başlangıç olarak seçilen üyelik fonksiyonları bütün giriş çıkış çiftleri için aynı kalmaktadır. Gradient Descent Training metodunda ilk önce bulanık sistem yapısı belirlenir ve daha sonra giriş çıkış çiftlerine göre bulanık sistem yapısındaki bazı parametreleri değiştirilip optimum üyelik fonksiyonları belirlenebilmektedir.

İlk adım olarak bulanık sistem yapısı oluşturulmalıdır. En çok kullanılan bulanık sistem yapısında, product inference engine, single tone fuzzifier, merkezi ortalama defuzzifier ve gaussian üyelik fonksiyonu kullanılarak elde edilmektedir.

$$f(x) = \frac{\sum_{l=1}^{M} y^{-l} \left[\prod_{i=1}^{n} \exp(-(\frac{x_1 - x_i^{-l}}{\sigma_i^l})^2 \right]}{\sum_{l=1}^{M} \left[\prod_{i=1}^{n} \exp(-(\frac{x_1 - x_i^{-l}}{\sigma_i^l})^2 \right]} \qquad (5.9)$$

burada M sabit kural sayısı, n sabit giriş sayısı ve [y^{-l} , x_i^{-l} *ve* σ_i^{-l}] serbest parametrelerdir.

Gradient Descent metodunda izlenmesi gereken adımları özetlemek istersek:

Adım 1

Bulanık sistem yapısı seçildikten sonra, M belirlenir. M büyük seçilirse doğruluk nispeti artacaktır. $y^{-l}(0)$, $x_i^{-l}(0)$ *ve* $\sigma_i^{-l}(0)$ ilk değerleri atanır.

Bu ilk değerler uzman bilgisinden faydalanılarak belirlenebilir veya üyelik fonksiyonlarının giriş ve çıkış değerlerini örtecek şekilde seçilmelidir.

Adım 2

Verilen $(x_0^p; y_0^p)$ giriş çıkış çiftleri için, sistem eğitilmeli ve bulanık sistem çıkışı hesaplanmalıdır.

$$z^l = \prod_{i=1}^{n} \exp(-(\frac{x_{0i}^p - x_i^{-l}(q)}{\sigma_i^l(q)})^2) \qquad (5.10)$$

$$b = \sum_{l=1}^{M} z^l \qquad (5.11)$$

$$a = \sum_{l=1}^{M} y^{-l}(q) z^l \qquad (5.12)$$

p = 0, 1, 2, 3, ..
q = 0, 1, 2,3, ...
i = 1, 2, 3, .. n
l = 1, 2, 3, .. M

olmak üzere

$$f = \frac{a}{b} \qquad (5.13)$$

hesaplanır. Burada toplam (2n+1)M tane serbest parametre söz konusudur.

Adım 3

Eğitme algoritması kullanılarak parametreler güncellenmelidir. Bu adımdaki amaç parametreler ayar edilerek optimum f değeri hesaplanmaya çalışılmasıdır. Belirlenen e^p hata oranına ulaşılana kadar her giriş çıkış çifti için eğitime devam edilir.

$$e^p = \frac{1}{2}\left[f(x_o^p) - y_o^p\right]^2 \tag{5.14}$$

Gradyan tabanlı eğitme ile bulanık sistem tasarımında, e^p hatasını serbest parametrelere göre minimize etmek için, y^{-l} , x_i^{-l} *ve* σ_i^{-l} parametreleri aşağıdaki şekilde güncellenir.

$$y^{-l}(q+1) = y^{-l}(q) - \alpha \left.\frac{\partial e}{\partial y^{-l}}\right|_q \tag{5.15}$$

$$x_i^{-l}(q+1) = x_i^{-l}(q) - \alpha \left.\frac{\partial e}{\partial x_i^{-l}}\right|_q \tag{5.16}$$

$$\sigma_i^{-l}(q+1) = \sigma_i^{-l}(q) - \alpha \left.\frac{\partial e}{\partial \sigma_i^{-l}}\right|_q \tag{5.17}$$

Adım 4

Belirlenen hataya ulaşılıncaya kadar adım 2 ve adım 3 tekrarlanmalıdır.

Adım 5

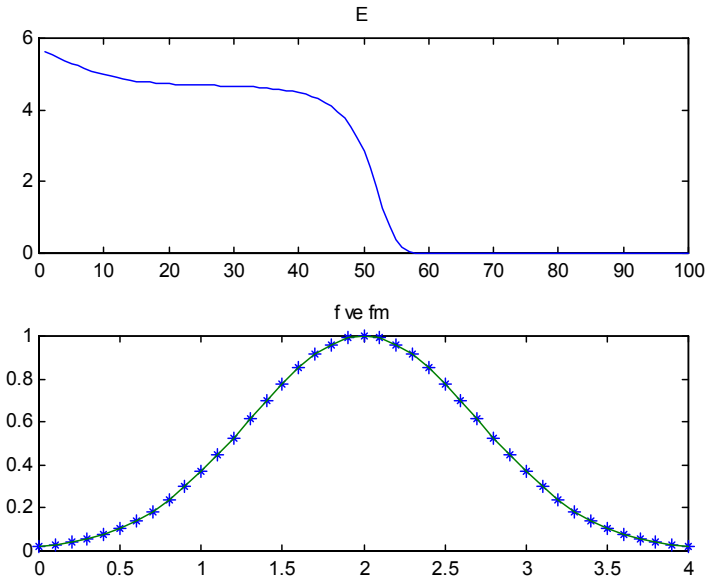
Bir sonraki giriş çıkış çifti için adım 2, 3 ve 4 tekrarlanır.

5.4 Gradyen Tabanlı Bulanık Sistem Tasarımı Uygulamaları

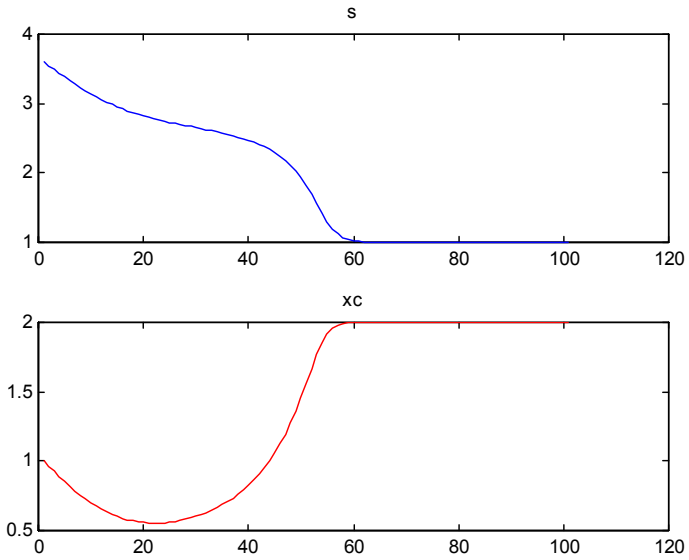
5.4.1 Gradyen Yöntemle Basit Fonksiyon Eğitme

```
%Gradyen yöntemle y=exp(-(x-xc)/s)^2 fonk.parametrtelerini ayarlama
clear all
x=0:0.1:4;
alfa=0.05;

xco=1;so=3.6;
p=[xco;so];
xc=p(1);s=p(2);

sd=[s];
xcd=[xc];
Ed=[];

%aşağıdan xc ve s geliyor...

fm=gauss(x,2,1);%Model

for i=1:100,

f=gauss(x,xc,s);%Process
e=(f-fm);
E=0.5*e*e';
Ed=[Ed E];

po=p;
dp=[e*[(x-xc)/s^2.*f;(x-xc).^2/(s^3).*f]'];
d1=[d1 dp];
p=po-2*alfa*dp;

tt=0;
if tt==1
if p(1)>0
xc=p(1);
else
xc=0;
end

if p(1)>4
xc=4;
end
```

```
if p(2)>2
s=2;
else
s=p(2);
end

if p(2)<0
s=0.1;
end
end

xc=p(1);
s=p(2);
xcd=[xcd xc];
sd=[sd s];

end

subplot(2,1,1),plot(sd,'b')
title('s');
subplot(2,1,2),plot(xcd,'r')
title('xc');
pause

f=gauss(x,xc,s);%Process

subplot(2,1,1),plot(Ed,'b')
title('E')
subplot(2,1,2),plot(x,f,'*',x,fm)
title('f ve fm')
pause,close
plot(d1'),title('Parametrelerin (xc ve s) türevlerinin iterasyonla
değişimi');
```



Şekil 5.10 a) Hatanın değişimi **b)** Gerçek ve bulanık sistem çıkışları



Şekil 5.11 **a)** s parametresinin eğitme ile değişimi
b) x_c parametresinin eğitme ile değişimi

5.4.2 Örnek

Li-Xin Wang'ın Kitabındaki Örnek 13.1'in çözümü aşağıda verilmiştir.

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + g[u(k)] \quad (5.18)$$

şeklinde verilen fark denkleminde bilinmeyen fonksiyon

$$g(u) = 0.6\sin(\pi u) + 0.3\sin(3\pi u) + 0.1\sin(5\pi u) \quad (5.19)$$

ve

$$u(k)=\sin(2\pi k7200) \quad (5.20)$$

olsun.

M=10 ve α=5 seçelim.

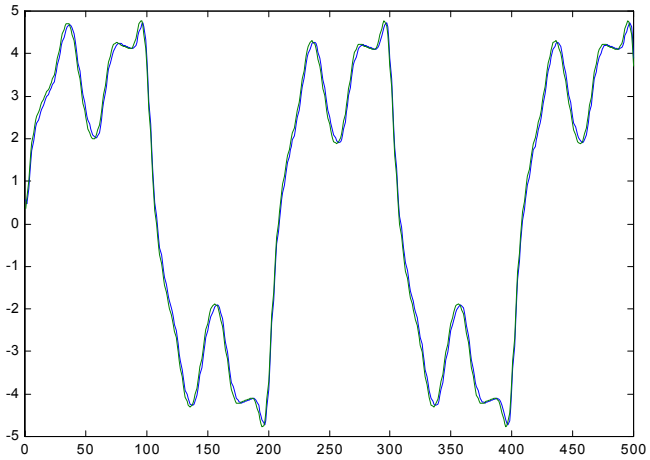
Bu fark denkleminin bulanık sistem eşdeğer modeli;

$$\hat{y}(k+1) = \hat{f}(0.3y(k) + 0.6y(k-1) + g[u(k)]) \quad (5.21)$$

şeklinde olacaktır. Burada $\hat{f}(x)$ ilk adımda seçilen bulanık sistemi göstermektedir.

```
y(1)=0;
y(2)=0;
u(1)=0;
p=1;

for k=2:699;
u(k)=sin(2*pi*k/200);

y(k+1)=0.3*y(k)+0.6*y(k-1)+0.6*sin(pi*u(k))+0.3*sin(3*pi*u(k))...
  +0.1*sin(5*pi*u(k));

xvy(p,:)=[y(k) y(k-1) u(k) y(k+1)];
p=p+1;
end
```

```
xm=ones(10,3);
sgm=0.1.*ones(10,3);
ym=ones(10,1);
a=0;
b=0;
alfa=5;

d=10;


for p=1:500;
d=10;
a=0;
b=0;

while abs(d) > 0.1;

for k=1:10
z(k)=exp(-((xvy(p,1)-xm(k,1))/sgm(k,1))^2)*exp(-((xvy(p,2)-
xm(k,2))/sgm(k,2))^2)*...
exp(-((xvy(p,3)-xm(k,3))/sgm(k,3))^2);

b=b+z(k);

a=a+ym(k,1)*z(k);

end
f(p)=a/b;

for k=1:10;

for g=1:3;

xm(k,g)=xm(k,g)-(alfa/b)*(f(p)-xvy(p,4))*(ym(k,1)-
f(p))*z(k)*2*(xvy(p,g)-xm(k,g))/(sgm(k,g)^2);

sgm(k,g)=sgm(k,g)-(alfa/b)*(f(p)-xvy(p,4))*(ym(k,1)-f(p))*z(k)*2*...
((xvy(p,g)-xm(k,g))^2)/(sgm(k,g)^3);

end
ym(k,1)=ym(k,1)-(alfa/b)*z(k)*(f(p)-xvy(p,4));
end
d=0.5*(f(p)-xvy(p,4));
end

end
```



Şekil 5.12 Bulanık eşdeğer model ve gerçek değerler

5.4.3 Örnek

Li-Xin Wang'ın Kitabındaki Örnek 13.2'nin çözümü aşağıda verilmiştir.

Gradient Descent metodunu çok girişli ve çok çıkışlı bir sistem için düşünelim.

$$y1(k+1) = y1(k)/(1+y2(k)^2)+u1(k) \quad (5.22)$$
$$y2(k+1) = y1(k)*y2(k)/(1+y2(k)^2)+u2(k) \quad (5.23)$$

$$y2^{(k+1)= f2^{(y1(k)*y2(k)/(1+y2(k)^2))}+u2(k) \quad (5.24)$$
$$y1^{(k+1)= f1^{(y1(k)/(1+y2(k)^2))}+u1(k) \quad (5.25)$$

$$u1(k)=\sin(2\pi k/25) \quad (5.26)$$

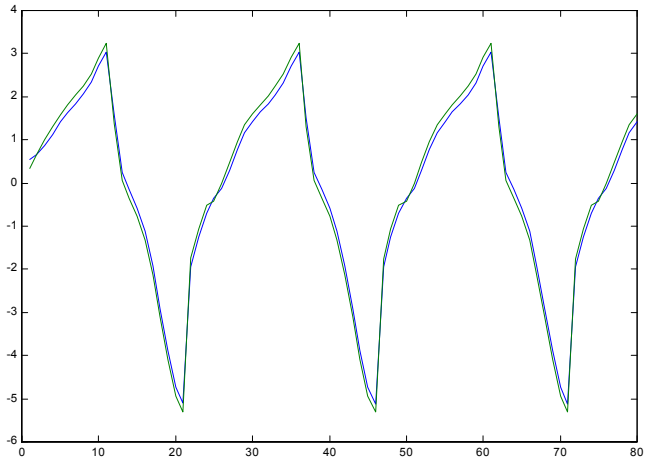
$$u2(k)=\cos(2\pi k/25) \quad (5.27)$$

M=121 seçelim.

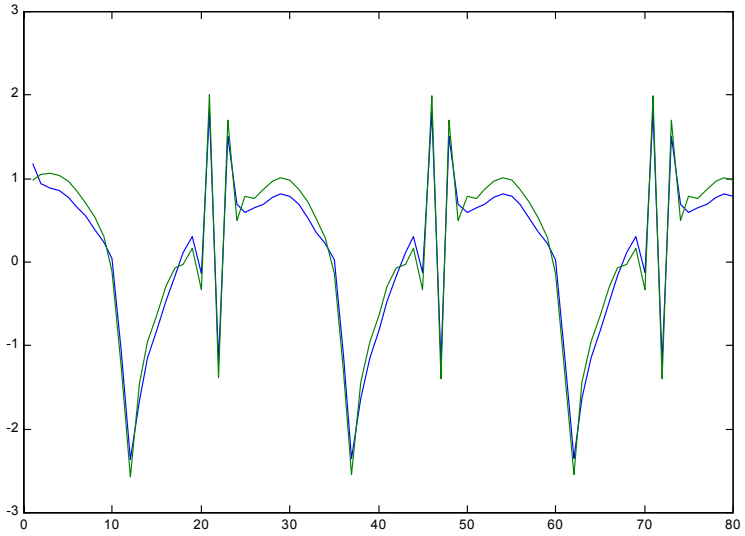
```
y1(1)=0.1;
y2(1)=0.1;
p=1;

for k=1:80;
u1(k)=sin(2*pi*k/25);
u2(k)=cos(2*pi*k/25);
y1(k+1)=y1(k)/(1+(y2(k))^2)+u1(k);
xvy1(p,:)=[y1(k) y2(k) u1(k) y1(k+1)];
y2(k+1)=y1(k)*y2(k)/(1+(y2(k))^2)+u2(k);
xvy2(p,:)=[y2(k) y1(k) u2(k) y2(k+1)];
p=p+1;
end


xm1=ones(121,2);
sgm1=0.1.*ones(121,2);
ym1=ones(121,1);

xm2=ones(121,2);
```

```
sgm2=0.1.*ones(121,2);
ym2=ones(121,1);

alfa=50;
d1=10;

for p=1:80;
d1=10;
a1=0;
b1=0;

while abs(d1) > 0.1;

for k=1:121;
z1(k)=exp(-((xvy1(p,1)-xm1(k,1))/sgm1(k,1))^2)*exp(-((xvy1(p,2)-
xm1(k,2))/sgm1(k,2))^2);

b1=b1+z1(k);

a1=a1+ym1(k,1)*z1(k);

end

f1(p)=(a1/b1)+xvy1(p,3);

for k=1:121;

for g=1:2;
xm1(k,g)=xm1(k,g)-(alfa/b1)*(f1(p)-xvy1(p,4))*...
(ym1(k,1)-f1(p))*z1(k)*2*(xvy1(p,g)-xm1(k,g))/(sgm1(k,g)^2);

sgm1(k,g)=sgm1(k,g)-(alfa/b1)*(f1(p)-xvy1(p,4))*(ym1(k,1)-
f1(p))*z1(k)*2*...
((xvy1(p,g)-xm1(k,g))^2)/(sgm1(k,g)^3);

end

ym1(k,1)=ym1(k,1)-(alfa/b1)*z1(k)*(f1(p)-xvy1(p,4));

end

d1=0.5*(f1(p)-xvy1(p,4));

end

end
```

```
for p=1:80;
d2=10;
a2=0;
b2=0;

while abs(d2) > 0.1;

for k=1:121;
z2(k)=exp(-((xvy2(p,1)-xm2(k,1))/sgm2(k,1))^2)*exp(-((xvy2(p,2)-
xm2(k,2))/sgm2(k,2))^2);

b2=b2+z2(k);

a2=a2+ym2(k,1)*z2(k);

end

f2(p)=(a2/b2)+xvy2(p,3);

for k=1:121;

for g=1:2;
xm2(k,g)=xm2(k,g)-(alfa/b2)*(f2(p)-xvy2(p,4))*...
(ym2(k,1)-f2(p))*z2(k)*2*(xvy2(p,g)-xm2(k,g))/(sgm2(k,g)^2);

sgm2(k,g)=sgm2(k,g)-(alfa/b2)*(f2(p)-xvy2(p,4))*(ym2(k,1)-
f2(p))*z2(k)*2*...
((xvy2(p,g)-xm2(k,g))^2)/(sgm2(k,g)^3);

end

ym2(k,1)=ym2(k,1)-(alfa/b2)*z2(k)*(f2(p)-xvy2(p,4));

end

d2=0.5*(f2(p)-xvy2(p,4));
end
end
plot(t,f1,t,xvy1(:,4))
plot(t,f2,t,xvy2(:,4))
```



Şekil 5.13 y1(k) için bulanık eşdeğer model çıkışı ve gerçek değerler



Şekil 5.14 y2(k) için bulanık eşdeğer model çıkışı ve gerçek değerler

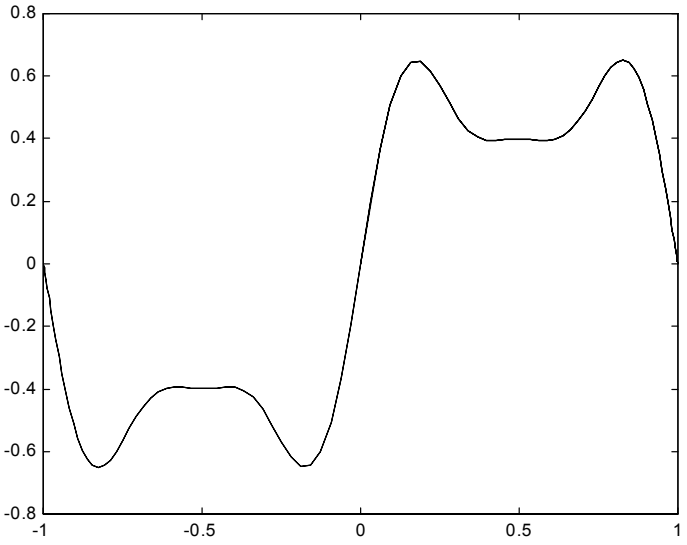
5.4.4 On line parametre atama yöntemi ve adım adım eğitme ile ilgili dinamik sistem eğitme örneği

```
% giriş x,çıkış yf
% 5 kural, basit gradyen yöntemi, on-line parametre ataması
% adım adım eğitme.

clear
yp(1)=0; yp(2)=0;
u(1)=0; d(1)=1;

for k=2:250-1;
d(k)=k;
if k<500
u(k)=sin(2*pi*k/200);
else
u(k)=0.5*sin(2*pi*k/250)+0.5*sin(2*pi*k/25);
end
g(k)=0.6*sin(pi*u(k))+0.3*sin(3*pi*u(k))+0.1*sin(5*pi*u(k));
yp(k+1)=0.3*yp(k)+0.6*yp(k-1)+g(k);
end


plot(u,g,'k')
%title('girişe(u) dinamik sistemin verdiği çıkış ')
pause

%alfa=0.0075; beta=0.0015; %M=15,9,7
%alfa=0.0025; beta=0.0015; %M=5

beta=0.08; %
%beta=0; %72.16, beta=0.08; 54.07
alfa=0.5;
M=5;

% On line parametre ataması.
xco=u(:,1:M)';xi=xco;
so=((max(xco)-min(xco))/M)*ones(1,M)';
%so=ones(1,M)';
si=so;

yco=yp(:,1:M)';yc=yco;yi=yc;
deo=ones(1,M)';
```

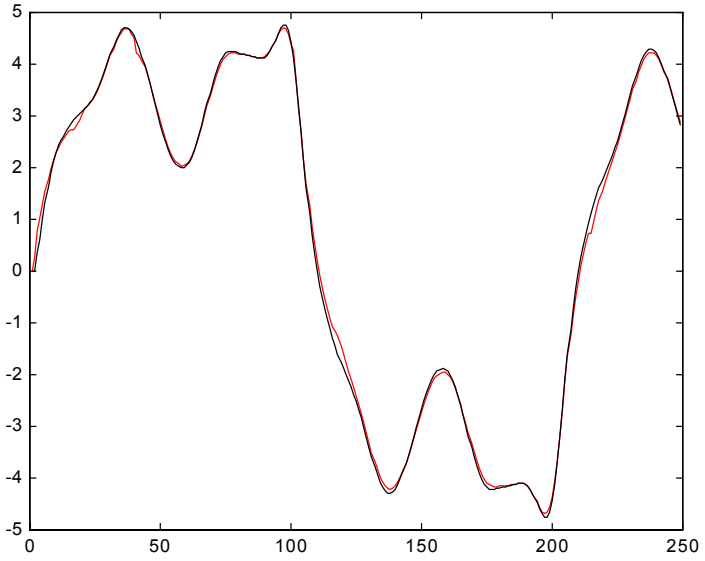
```
po=[xco so yco deo];
dp=zeros(5,4);

%kuralciz([-0.1:0.02:0.3],xco,so,yco,[-1:0.02:1]),pause

Tope=1.0;
tt=[];GG=[];

[aa,bb]=size(u);
p=0;

while Tope>0.5 & p<1
Tope=0.0;
p=p+1;

for q=1:bb,
y=yp(q); %Tanımlanacak fonksiyon
z=gaussn(u(q),xco,so);
z=z./deo;

b=sum(z);
a=yc'*z;
f=a/b;

ys=0.3*yp(q+1)+0.6*yp(q)+f;
fd(q)=f;
yf(q)=ys;

e=0.5*(ys-y)^2;
Tope=Tope+e;

%mom=ones(size(1,M))*beta*(ys-y);
mom=beta*dp;
x(q)=u(q);

eo=(ys-y);
yc=yco-alfa*eo*z/b+mom(:,3);
xc=xco-alfa*eo*2*(yco-ys).*z.*(x(q)-xco)./(b*so.^2)+mom(:,1);
s=so-alfa*eo*2*(yco-ys).*z.*((x(q)-xco).^2)./(b*so.^3)+mom(:,2);
de=deo-alfa*eo*(yco-ys).*(-z./deo)/sum(z);

p=[xc s yc de];
dp=p-po;
po=p;
```

```
%po=pupdate(po,[-1:1],[-10:10]);
xco=po(:,1);
so=po(:,2);
yco=po(:,3);
deo=po(:,4);

end
tt=[tt Tope];
end
plot(d,yf,'r',d,yp(1:249),'k');
%title('Eğitme fazı'),
%legend('Bulanık sistem çıkışı','gerçek sistem çıkışı')
pause;

%semilogy(tt),pause

[tn,tm]=size(xi);

for j=1:tn,
GG=[GG;gaussn(x,xc(j),s(j))];
end
bGG=GG;
if min(size(GG))>1
bGG=sum(GG);
end

fson=yc'*[GG]./bGG;

% plot(d,yp(1:249),'r',d,yf,'b'),pause
% plot(u,g,u,fd),pause

% Deneme fazı.

tson=700;

for k=250:tson-1;
d(k)=k;
if k<500
u(k)=sin(2*pi*k/200);
else
u(k)=0.5*sin(2*pi*k/250)+0.5*sin(2*pi*k/25);
end
g(k)=0.6*sin(pi*u(k))+0.3*sin(3*pi*u(k))+0.1*sin(5*pi*u(k));
yp(k+1)=0.3*yp(k)+0.6*yp(k-1)+g(k);
end
```

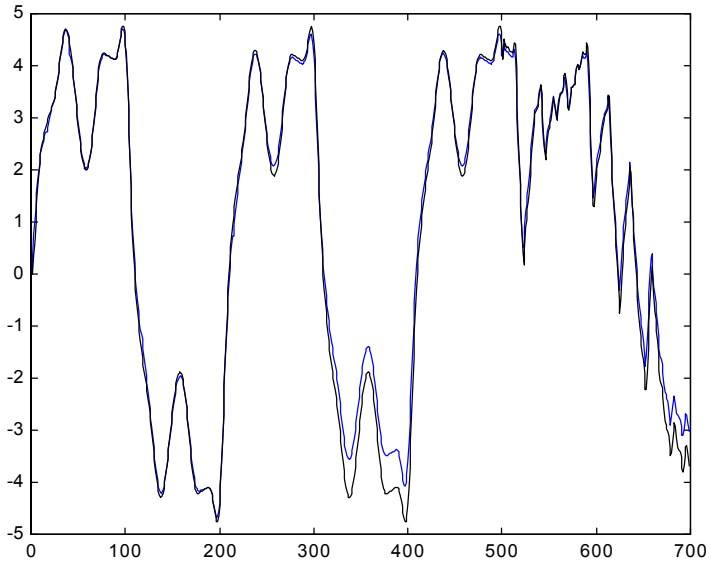
```
E=0;
for q=250:tson-1,
y=yp(q); %Tanımlanacak fonksiyon

z=gaussn(u(q),xc,s);
z=z./deo;

b=sum(z);
a=yc'*z;
f=a/b;
fdt(q)=f;
ys=0.3*yp(q+1)+0.6*yp(q)+f;
E=E+(ys-y)^2;

yf(q)=ys;
end
plot(d,yf,'b',d,yp(1:tson-1),'k');
%title('Eğitme fazı (k<=250) ve Test fazı (k>250)'),
%legend('Bulanık sistem çıkışı','gerçek sistem çıkışı')
pause;
%kuralciz([-1:0.02:1],xco,so,yco,[-1:0.02:1])
E
plot(d,yf-yp(1:tson-1),'k'),grid
axis([0 700 -5 5])
%title('Hata'),pause

plot(d,yf,'b',d,yp(1:tson-1),'k',d,yf-yp(1:tson-1))
```



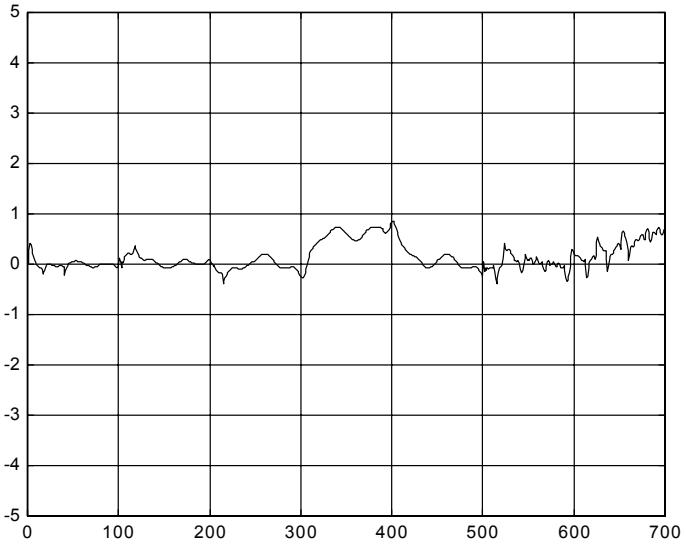
Şekil 5.15 Dinamik sistemin verdiği çıkış



Şekil 5.16 Eğitme fazı



Şekil 5.17 Eğitme fazı (k=250) ve Test fazı (k > 250) iken bulanık sistem ve gerçek sistem çıkışı



Şekil 5.18 Gerçek sistem ve bulanık sistem arasındaki hata

5.4.5 Arttırılmış Bulanık Sistemle Sistem Tanımlama Optimizasyon/Eğitme Yöntemi: BFGS

Çıkış üyelik fonksiyonlarının merkezini de dikkate alan bulanık sistem modeli aşağıda verilmiştir.

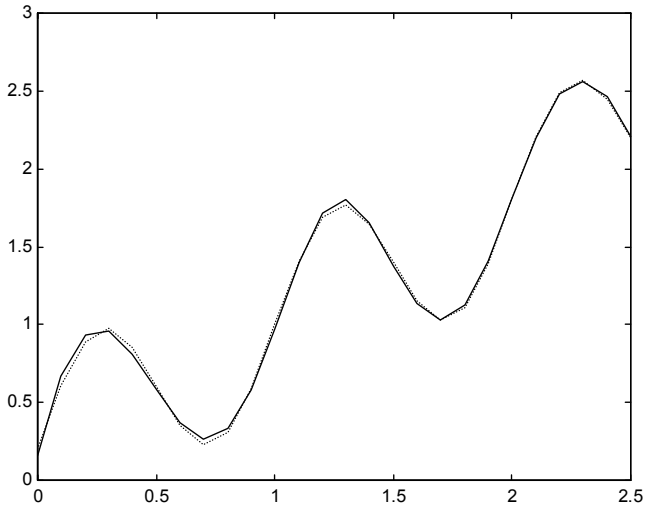
$$f(x,\bar{x}_i^l,\sigma_i^l,\bar{y}^l,\delta^l)=\frac{\sum_{l=1}^{5}\bar{y}^l\prod_{i=1}^{2}\mu_{A_i^l}(x_i)/\delta^l}{\sum_{l=1}^{5}\prod_{i=1}^{2}\mu_{A_i^l}(x_i)/\delta^l} \qquad (5.28)$$

Bu sisteme, Arttırılmış Bulanık Sistem denmektedir. Bu örnekte bu sistem kullanılarak gradyen tabanlı yöntemlerden biri olan BFGS yöntemi ile eğitme yapılacaktır. Bu yöntem doğrusal arama (line search) kullanır. Bir başka deyişle, öğrenme adımı her iterasyon adımında hesapanır.

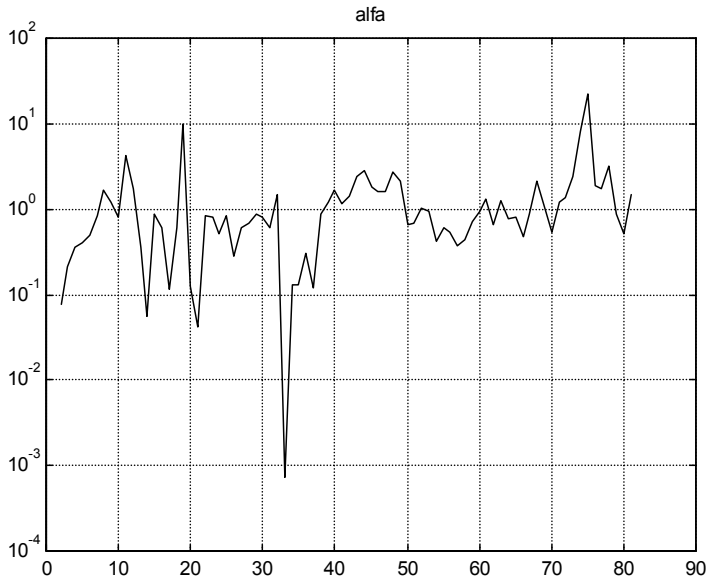
```
% Golden section search.
% Aynı başlangıç şartlarıyla Artırılmış sistem,
% daha az adımda istenen hatanın altına iniyor.

clear
global xk yp
xk=0:0.1:2.5;

xco=[0;1;1;2];
so=[1;1;1;1];
yco=[-1;0;1;1];
do=[1;1;1;1];
M=size(yco,1); %Kural sayısı

po=[xco;so;yco;do];
pd=po';

olcut='folcut5';
gradyen='fgrad5';

hata=feval(olcut,po);
```

```
ed=hata;
adiz=0;

k=1;
Ho=eye(4*M);

co=feval(gradyen,po);
con=norm(co);

d0=-inv(Ho)*co;

while  hata>0.0080 & con>0.0010
k=k+1;

alfa=goldens1(po,olcut,-d0);
%alfa=arac(po,olcut,-d0,0.001,0)

p=po+alfa*d0;
%p=pupdate(p,xk,yp);

hata=feval(olcut,p);

s0=alfa*d0;
c1=feval(gradyen,p);
yo=c1-co;

Do=yo*yo'/(yo'*s0);
Eo=co*co'/(co'*d0);

H1=Ho+Do+Eo;

d1=-inv(H1)*c1;
d0=d1;
Ho=H1;
co=c1;
con=norm(co);

po=p;

pd=[pd;p'];
ed=[ed hata];
adiz=[adiz alfa];
[k alfa hata con]
fcikis5(po,xk),pause(.2);
end
fcikis5(po,xk),pause
```

```
p=reshape(p,M,4)
renk='brkc';

for jj=1:size(pd,2);
plot(pd(:,jj),renk(rem(jj,4)+1))
hold on
end
pause
hold off

semilogy(ed,'k'),grid,title('Hata'),pause
semilogy(adiz,'k'),grid,title('alfa'),pause
```



Şekil 5.19 Gerçek sistem ve Bulanık Sistem çıktıları



Şekil 5.20 alfa değişimi

5.5 En Küçük Kareler (Recursive Least Squares) Kullanarak Bulanık Sistem Tasarımı

Gradient Descent metodunda adım, adım eğitmede hatanın en aza indirilmesi her bir veri çifti için yapılır. Eğitme algoritması bir kerede sadece bir tane giriş çıkış çifti için parametreleri güncellemektedir. Recursive Least Squares metodunda bütün giriş çıkış çiftleri için hesaplanan toplam hata en aza indirmeye çalışılır.

Performans indeksi aşağıda verilmiştir.

$$J_p = \sum_{j=1}^{p} \left[f(x_o^j) - y_o^j \right]^2 \qquad (5.29)$$

Recursive Least Squares metodu kullanarak bulanık sistem tasarımında izlenecek adımlar şöyledir:

Adım 1

$$U = [\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times ..... \times [\alpha_n, \beta_n] \tag{5.30}$$

uzayını örtecek şekilde N_i tane bulanık küme oluşturulur.

Adım 2

$\prod_{i=1}^{n} N_i$ tane bulanık IF – THEN kuralları oluşturulur. Daha sonra örneğin, product inference engine, single tone fuzzifier ve merkezi ortalama defuzzifier seçilerek tanımlanan fuzzy sistem tasarlanır.

$$f(x) = \frac{\sum_{l_1=1}^{N_1} ..... \sum_{l_n=1}^{N_n} y^{-l_1 ... l_n} \left[\prod_{i=1}^{n} \mu_{A_i^{l_i}}(x_i) \right]}{\sum_{l_1=1}^{N_1} ..... \sum_{l_n=1}^{N_n} \left[\prod_{i=1}^{n} \mu_{A_i^{l_i}}(x_i) \right]} \tag{5.31}$$

Adım 3

İlk değer parametreleri atanır.

Adım 4

Adım 2'de tanımlanan *f(x)* bulanık sistem aşağıdaki gibi düzenlenir.

$$\theta = (y^{-1...1}, y^{-N_1 1...1}, ..., y^{-N_1 N_2 N_3})^T \tag{5.32}$$

$$f(x) = b^T(x)\theta \tag{5.33}$$

$$b(x) = (b^{1..1}(x), ..., b^{N_1 1...1}, ..., b^{N_1 N_2 ... N_n}(x))^T \tag{5.34}$$

$$b^{l_1 ... l_n}(x) = \frac{\left[\prod_{i=1}^{n} \mu_{A_i^{l_i}}(x_i) \right]}{\sum_{l_1=1}^{N_1} ..... \sum_{l_n=1}^{N_n} \left[\prod_{i=1}^{n} \mu_{A_i^{l_i}}(x_i) \right]} \tag{5.35}$$

J_p minimize edecek parametre güncelleme algoritması (en küçük kareler) aşağıda verilmiştir

$$\theta(p) = \theta)p-1) + K(p)\left[y_0^p - b^T(x_0^p)\theta(p-1)\right] \quad (5.36)$$

$$K(p) = P(p-1)b(x_0^p)\left[b^T(x_0^p)P(p-1)b(x_0^p)+1\right]^{-1} \quad (5.37)$$

$$P(p) = P(p-1) - K(p)b^T(x_0^p)P(p-1) \quad (5.38)$$

$\theta(0)$ üçüncü adımda seçilmişti. $P(0) = \sigma I$, I birim matris olmak üzere σ büyük bir sabit seçilir. Adım 2'de tanımlanan bulanık sistemdeki $y^{-l_1....l_n}$ parametreleri $\theta(p)$ elemanlarına karşılık gelmektedir.

5.6 En Küçük Kareler Bulanık Sistem Tasarımı Uygulamakları

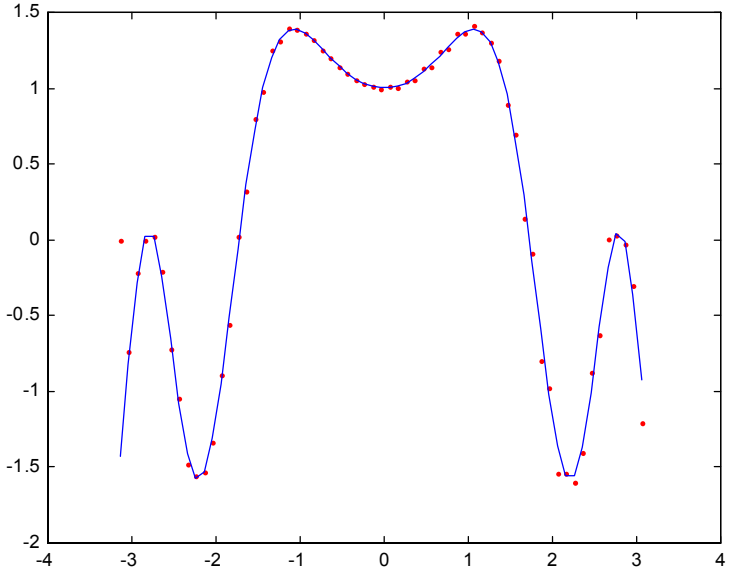
5.6.1 Örnek 1

$$y = \sin(x_1 * x_1) + \cos(x_1) \quad (5.39)$$

$-\pi < x_1 < \pi$ arasında tanımlanan sistemi Recursive Least Squares metodu kullarak bulanık sistem tasarımı yapalım.

```
x1=-pi:0.1:pi;
y=sin(x1.*x1)+cos(x1);

n1=100;

c1(1,:)=linspace(-pi,pi,n1);
sigma1(1,:)=2*ones(1,n1);

for j=1:n1
mu1(j,1)=exp(-0.5*((x1(1)-c1(1,j))/sigma1(1,j))^2);
end
```

```
payda(1)=sum(mu1(:,1));

for j=1:n1
bi(j,1)=mu1(j,1)/payda(1);
end

sgm=500;

P(:,:,1)=sgm*eye(n1);
Q(:,1)=zeros(n1,1);

yest(1)=bi(:,1)'*Q(:,1);


for k=2:63;


for j=1:n1
mu1(j,k)=exp(-0.5*((x1(k)-c1(1,j))/sigma1(1,j))^2);
end

payda(k)=sum(mu1(:,k));

for j=1:n1
bi(j,k)=mu1(j,k)/payda(k);
end

K(:,k)=P(:,:,k-1)*bi(:,k)/(bi(:,k)'*P(:,:,k-1)*bi(:,k)+1);
Q(:,k)=Q(:,k-1)+K(:,k)*(y(k)-bi(:,k)'*Q(:,k-1));
P(:,:,k)=eye(size(P(:,:,k-1)))-K(:,k)*bi(:,k)'*P(:,:,k-1);

yest(k)=bi(:,k)'*Q(:,k);

end

plot(x1,yest,'r.',x1,y,'b');
title('Bulanık Systems Using Recursive Least Squares')
```

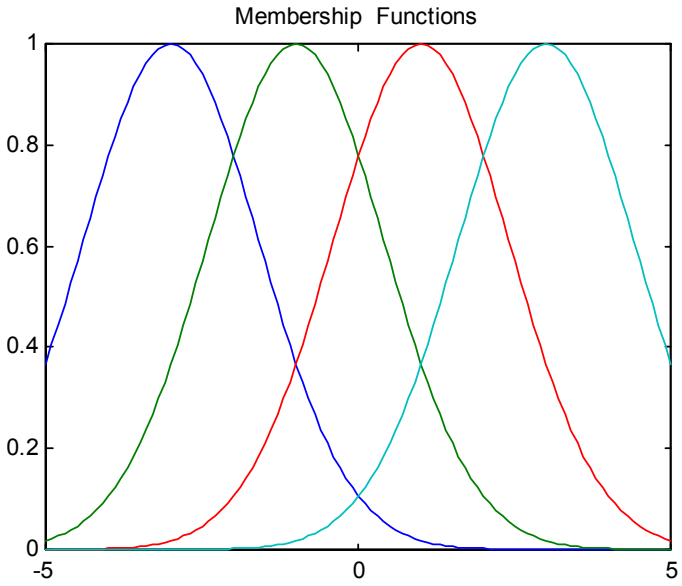


Şekil 5.21 Gerçek değerler ve bulanık sistem çıkışı

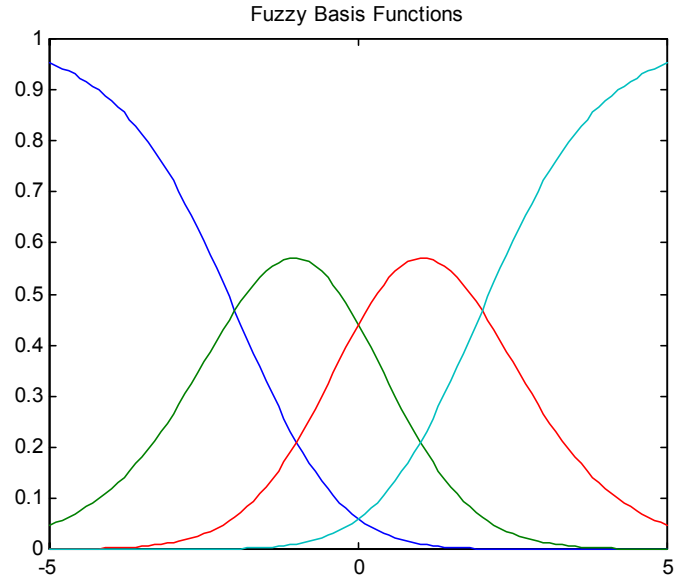
5.6.2 Bulanık Taban Fonksiyonları

[-3;3] uzayında 4 adet gauss üyelik fonksiyonu tanımlayıp, bulanık taban fonksiyonlarını hesaplatıp çizdiren Matlab kodları aşağıdaki gibidir.

```
clear all;
x=-5:0.1:5;
xc=[-3 -1 1 3];
s=sqrt(2)*ones(size(xc));
muf=gaussn(x,xc,s);
plot(x,muf'),,title('Membership  Functions');
pause
close

for j=1:max(size(xc))
p(j,:)=muf(j,:)./sum(muf);
end

plot(x,p),title('Bulanık Basis Functions')
```



Şekil 5.22 Üyelik fonksiyonları



Şekil 5.23 Bulanık taban fonksiyonları

5.6.3 RLS ile FS sistem Tanımlama Örneği

Burada kullanılan *gaussn* fonksiyonu ***n*** adet gauss üyelik fonksiyonunu tanımlar...

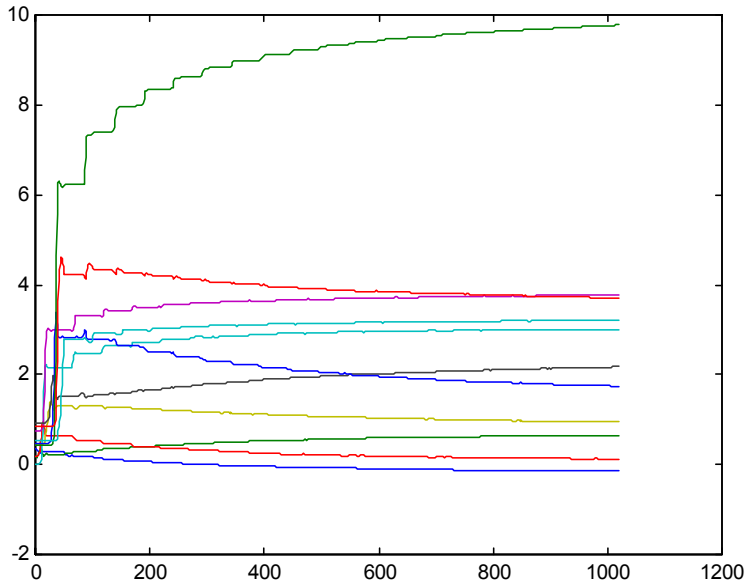
```
function y=gaussn(x,c,s)
sc=max(size(c));

for i=1:sc
y(i,:)=exp(-0.5*((x-c(i))./s(i)).^2);
end
```

RLSE ile yo=2*xo/(sin(3*xo)+1+tanh(xo)) (5.40)

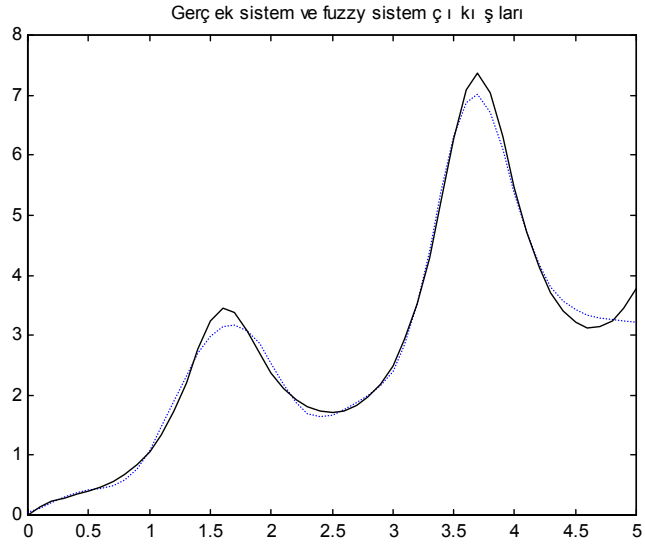
fonksiyonunu xa=[0:0.1:5] de tanımlanıyor.

```
clear all
%RLSE  yo=2*xo/(sin(3*xo)+1+tanh(xo));
%fonksiyonunu xa=[0:0.1:5] de tanımlıyor
%
td=[];
bx=11;
xar=[0:5/bx:5-5/bx];

t=rand(1,bx);
Po=eye(bx);
hata=1;
p=0;
adim=0;
hatad=[];
while hata>0.10  & adim<20
hata=0;
for xo=0:0.1:5;
yo=(2*xo)/(sin(3*xo)+1+tanh(xo));
bff=regf(xo,xar);
AA=inv(bff*Po*bff'+1);
P=Po-Po*bff'*AA*bff*Po;
K=Po*bff'*AA;
t=t+K'*(yo-bff*t');
td=[td;t];
Po=P;
hata=hata+(yo-bff*t')^2;
end
hatad=[hatad hata];
adim=adim+1;
[adim hata]
```

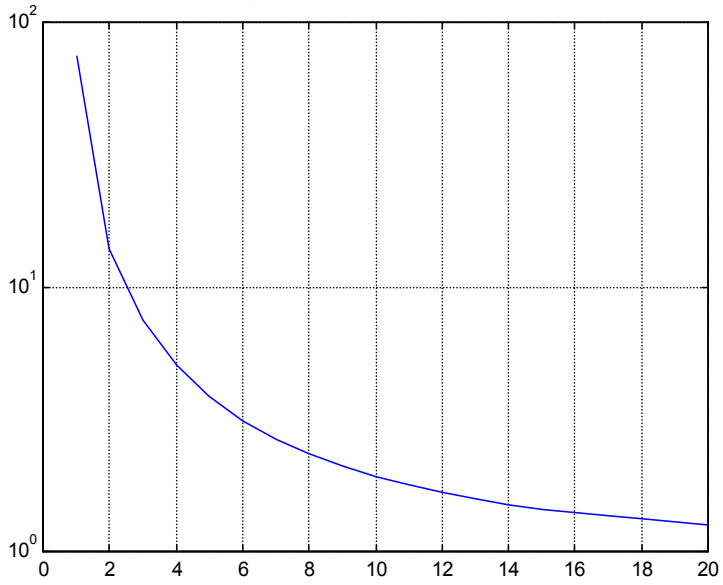
```
end
plot(td),title('parametrelerin çıkış üyelik fonksiyonu
merkezleri)değişimi');
pause

xa=[0:0.1:5];

r1=regf(xa,xar);
r=r1';
yson=t*r;
yg=(2*xa)./(sin(3*xa)+1+tanh(xa));
plot(xa,yson,':b',xa,yg,'k-'),
title('Gerçek sistem ve bulanık sistem çıkışları');
pause
semilogy(hatad),grid
title('Hatanın iterasyonla değişimi');
```



Şekil 5.24 Parametrelerin çıkış üyelik fonksiyonu merkezleri değişimi



Şekil 5.25 Gerçek sistem ve bulanık sistem çıkışları



Şekil 5.26 Hatanın iterasyonla değişimi

5.7 Clustering Kullanarak Bulanık Sistem Tasarımı

Bu metotta giriş çıkış çiftlerini kullanarak, kümeleme işlemi ile bulanık sistemin oluşturulması hedeflenmektedir. Çok kompleks sistemlerde kural sayısının çokluğu istenmeyen bir problem olarak karşımıza çıkmaktadır.

Bu metotta izlenecek kurallar sırasıyla şöyledir:

Adım 1

İlk giriş çıkış çifti için cluster merkezleri ve *r* yarıçapı belirlenir.

$A^1(1) = y_o^1, \quad B^1(1) = 1$ ilk değerleri atanır.

$A^l(k)$ k giriş/çıkış için l'ninci kümedeki çıkışların toplamı, $B^l(k)$ ise k giriş/çıkış için l'ninci kümedeki giriş çıkış sayısıdır.

Adım 2

k'ıncı giriş çıkış çifti için x_o^k'nın cluster merkezine olan uzaklığı hesaplanır.

Eğer $\left|x_o^k - x_c^{l_k}\right| > r$ ise x_o^k yeni cluster merkezi olarak atanır.

$$x_c^{M+1} = x_o^k \tag{5.41}$$

$$A^{M+1}(k) = y_o^k, \quad B^{M+1}(k) = 1 \tag{5.42}$$

$$A^l(k) = A^l(k-1), \quad B^l(k) = B^l(k-1) \quad l = 1...M \tag{5.43}$$

Eğer $\left|x_o^k - x_c^{l_k}\right| < r$ ise

$$A^{l_k}(k) = A^{l_k}(k-1) + y_o^k \tag{5.44}$$

$$B^{l_k}(k) = B^{l_k}(k-1) + 1 \tag{5.45}$$

$$A^l(k) = A^l(k-1), \quad B^l(k) = B^l(k-1) \quad l = 1...M \quad l \neq l_k \tag{5.46}$$

Adım 3

Eğer $x_o(k)$ yeni bir küme oluşturmuyor ise k giriş/çıkış çifti $(x_0^j ; y_0^j)$, j = 1, 2, 3, ...k için aşağıdaki bulanık sistem tasarlanır.

$$f(x) = \frac{\sum_{l=1}^{M} A^l(k) \exp(-(\frac{x - x_c^{-l}}{\sigma})^2}{\sum_{l=1}^{M} B^l(k) \exp(-(\frac{x - x_c^{-l}}{\sigma})^2} \qquad (5.47)$$

Adım 4

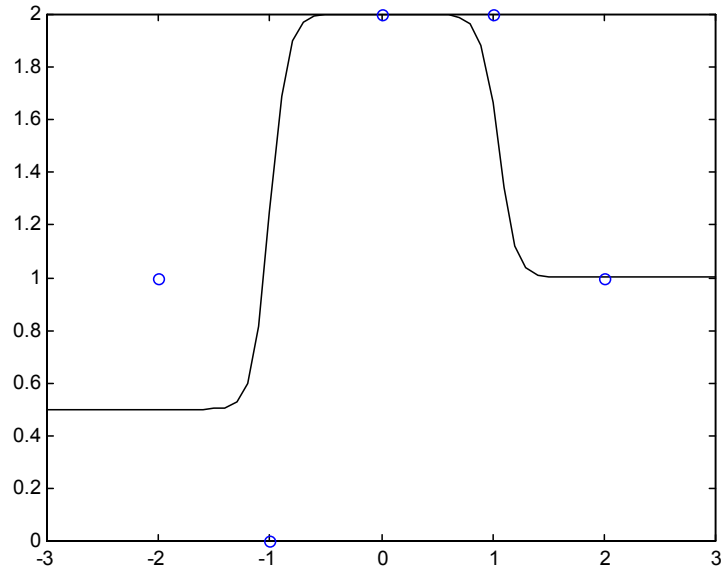
k=k+1 için 2. adıma geri dönüp, işlemler tekrarlanır.

5.8 Kümeleme Kullanarak Bulanık Sistem Tasarımı Uygulamaları

5.8.1 Li-Xin Wang'ın Kitabındaki Örnek 15.2

```
clear
format compact
xo=[-2 -1 0 1 2];
yo=[ 1  0 2 2 1];

r=1.5;

xc(1)=xo(1);
A(1,1)=1;
B(1,1)=1;
M=1

for k=2:max(size(xo)),
k
uz=abs(xo(k)-xc);
muz=min(uz);

imuz=find(uz==muz);


if abs(xo(k)-xc(imuz))>r
disp('yeni cluster')
M=M+1
xc(M)=xo(k)
A(M,k)=yo(k);
B(M,k)=1;

for l=1:(M-1)
A(l,k)=A(l,k-1);
B(l,k)=B(l,k-1);
end
end

if abs(xo(k)-xc(imuz))<r
disp('eski cluster')
A(imuz,k)=A(imuz,k-1)+yo(k);
B(imuz,k)=B(imuz,k-1)+1;

for l=1:M
if l~=imuz
```

```
A(l,k)=A(l,k-1);
B(l,k)=B(l,k-1);
end
end
end
AA=A([1:M],k)'
BB=B([1:M],k)'

end

% A(l,k) ; l. clusterdaki çkış(yo) ların toplamı.
% B(l,k) ; l. clusterdaki giriş/çıkış sayısı

% A(l,k) ve B(l,k) tamam,şimdi bulanık sistem...
s=0.3;
x=-3:0.1:3;

fu=0;
fa=0;

for l=1:M;
fu=fu+AA(l)*exp(-(x-xc(l)).^2./s);
fa=fa+BB(l)*exp(-(x-xc(l)).^2./s);
end

f=fu./fa;
plot(x,f,'k',xo,yo,'ob')
pause
l=1:M;
si=s*ones(1,M);
kuralciz(x,xc,si,BB,[-3:0.1:3]);

fuo=0;
fao=0;
for l=1:M;
fuo=fuo+AA(l)*exp(-(xo-xc(l)).^2./s);
fao=fao+BB(l)*exp(-(xo-xc(l)).^2./s);
end
%sadece verilen xo noktalarında hesap.
fo=fuo./fao;
E=0.5*(fo-yo)*(fo-yo)'
```



Şekil 5.27 Gerçek sistem ve bulanık sistem çıkışları

Ekran çıktısı

```
» M =
     1
k =
     2
eski cluster
AA =
     1
BB =
     2
k =
     3
yeni cluster
M =
     2
xc =
    -2     0
AA =
     1     2
BB =
     2     1
k =
```

```
    4
eski cluster
AA =
    1    4
BB =
    2    2
k =
    5
yeni cluster
M =
    3
xc =
   -2    0    2
AA =
    1    4    1
BB =
    2    2    1
```

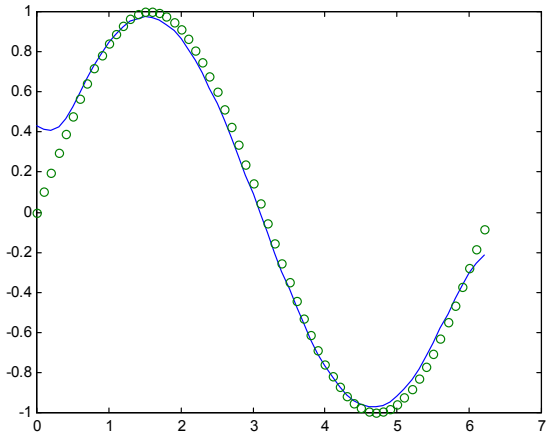
5.8.2 Bulanık Kümeleme ile sin(x) Fonksiyonunun Tanımlanması

r = 0.11 alınmıştır.

```
clear
format compact
xo=0:0.1:2*pi;
yo=sin(xo);

r=0.11;

xc(1)=xo(1);
A(1,1)=1;
B(1,1)=1;
M=1

for k=2:max(size(xo)),
k
uz=abs(xo(k)-xc);
muz=min(uz);

imuz=find(uz==muz);
```

```
if abs(xo(k)-xc(imuz))>r
disp('yeni cluster')
M=M+1
xc(M)=xo(k)
A(M,k)=yo(k);
B(M,k)=1;

for l=1:(M-1)
A(l,k)=A(l,k-1);
B(l,k)=B(l,k-1);
end
end

if abs(xo(k)-xc(imuz))<r
disp('eski cluster')
A(imuz,k)=A(imuz,k-1)+yo(k);
B(imuz,k)=B(imuz,k-1)+1;

for l=1:M
if l~=imuz
A(l,k)=A(l,k-1);
B(l,k)=B(l,k-1);
end
end
end
AA=A([1:M],k)'
BB=B([1:M],k)'

end


% A(l,k) ve B(l,k) tamam,şimdi fuzzy sistem...
s=0.1;

x=xo;
fu=0;
fa=0;
for l=1:M;
fu=fu+A(l,k)*exp(-(x-xc(l)).^2./s);
fa=fa+B(l,k)*exp(-(x-xc(l)).^2./s);
end
f=fu./fa;
whitebg
plot(x,f,xo,yo,'o')
pause
l=1:M;
```

```
si=s*ones(1,M);
kuralciz(x,xc,si,B(l),[-2:0.1:2]);
pause
close

fuo=0;
fao=0;
for l=1:M;
fuo=fuo+A(l,k)*exp(-(xo-xc(l)).^2./s);
fao=fao+B(l,k)*exp(-(xo-xc(l)).^2./s);
end
fo=fuo./fao;
E=(fo-yo)*(fo-yo)'
M
```



Şekil 5.28 Bulanık mantık çıkış ve gerçek çıkış

"E" toplam karesel hata = 0.4503
"M" küme sayısı = 32

Bölüm 6

MATLAB Fuzzy Logic Toolbox Kullanarak Bulanık Mantık Tabanlı Sistem Tasarımı

6.1 Matlab Bulanık Mantık GUI Araçları ve İşlevleri

Matlab ortamında GUI (graphical user interface - grafiksel kullanıcı arabirimi) kullanarak grafiksel ortamda sistemimizi kolaylıkla kurabiliriz. Fuzzy Logic Toolbox'a komut satırından da ulaşılabilir. Fuzzy Logic Toolbox'ta bir sistemi kurmak, kontrol etmek ve gözlemlemek için beş temel GUI aracı vardır:

Bulanık Karar Sistemi Editörü (Fuzzy inference System - FIS),
Üyelik Fonksiyonu Editörü (Membership Function Editor),
Kural Editörü (Rule Editor),
Kural İzleyici (Rule Viewer) ve
Yüzey İzleyici (Surface Editor).

Bu GUI 'ler dinamik olarak birbirine bağlıdır.

FIS Editör'ü sistemin en üst seviyesindeki işlemleri yapar. Kaç tane giriş ve çıkış değişkeni olduğu, isimlerinin ne olduğu burada belirlenir. Fuzzy Logic Toolbox giriş sayısını sınırlamaz. Ama yine de giriş sayısı kullandığımız bilgisayarın hafıza miktarıyla sınırlı olabilir. Eğer giriş sayısı yada üyelik fonksiyonlarının sayısı çok fazlaysa, diğer GUI araçlarıyla FIS'i analiz etmek zorlaşabilir.

Üyelik Fonksiyonu Editörü her bir değişkenle ilgili üyelik fonksiyonlarının şekillerini belirler.

Kural Editörü sistem davranışını belirleyen kuralları oluşturmak ve üzerinde düzenlemeler yapmak içindir.

Kural ve Yüzey İzleyici, düzenleme değil de FIS'i gözlemek için kullanılır. Bunlar sistem üzerinde değişiklik yapma özelliğine sahip değildir. Kural İzleyici hangi kuralların aktif olduğunu, her bir üyelik fonksiyonunun sonucu nasıl değiştirdiğini gösterir.

Yüzey İzleyici, çıkışın herhangi bir veya iki çıkış değerine bağlılığını gösterir yani sistemin çıkış yüzeyini oluşturur ve haritasını çizer.

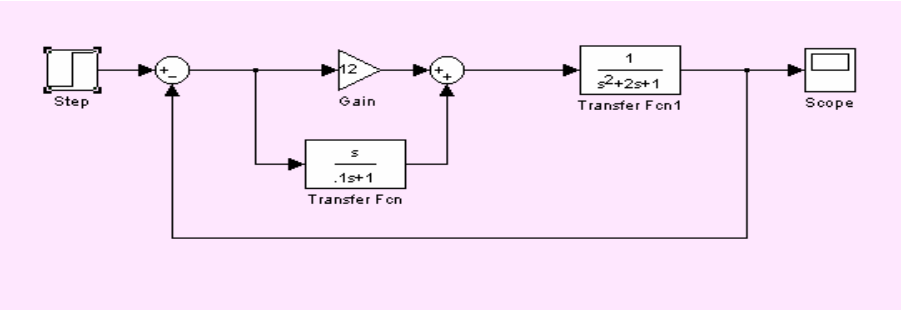
Fuzzy Logic Toolbox, Simulink (MathWorks'un sağladığı simülasyon yazılımı) ile birlikte çalışabilecek şekilde tasarlanmıştır. Bunun için tasarlanan sistem MATLAB Workspace ortamına kayıt edilir. GUI araçlarıyla yada diğer metotlarla oluşturulan bir sistem simülasyon ortamına doğrudan uygulanabilir.

6.2 MATLAB Simulink Ortamında Uygulamalar

Bu bölümde, Matlab Fuzzy Logic Toolbox kullanarak bazı kontrol uygulamaları için simulasyonlar yapılacaktır. Geleneksel anlamda oluşturulan PI, PD ve PID ile Fuzzy PI, Fuzzy PD ve Fuzzy PID gerçeklemeleri yapılıp, daha sonra karşılaştırılacaktır. Simulasyonlarda geleneksel denetleyicilerin parametrelerinin optimize edilmediği hatırdan çıkarılmamalıdır. Her iki durumda da denetleyici parametreleri deneme yanılma metodu ile seçilmiştir.

6.2.1 Geleneksel PD Denetleyici

Aşağıda görülen geleneksel PD denetleyici devresi MATLAB Simulink ortamında kurulmuştur.



Şekil 6.1 PD denetleyici blok şeması

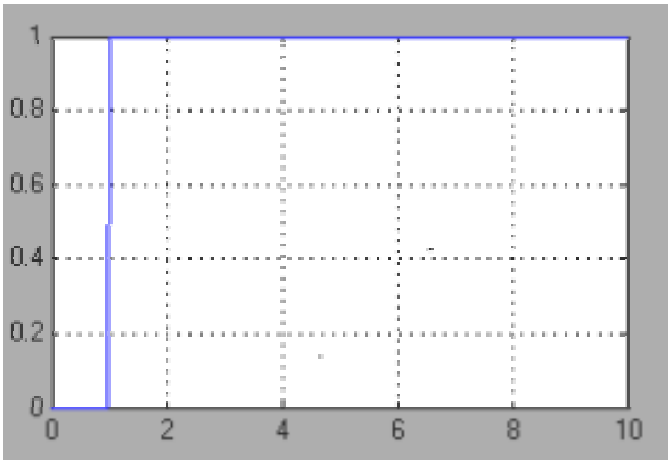
Kontrol edilen sistem, ikinci dereceden bir transfer fonksiyonuna sahiptir. Sistemin blok şeması Şekil 6.1 de verilmiştir.

Sistemde kazanç faktörü olarak K_p = 12 seçilmesinin sebebi sistem çıkışıyla giriş değerleri arasındaki farkın azaltılmasıdır, böylece kararlı-hal hatası da azalmış olur.

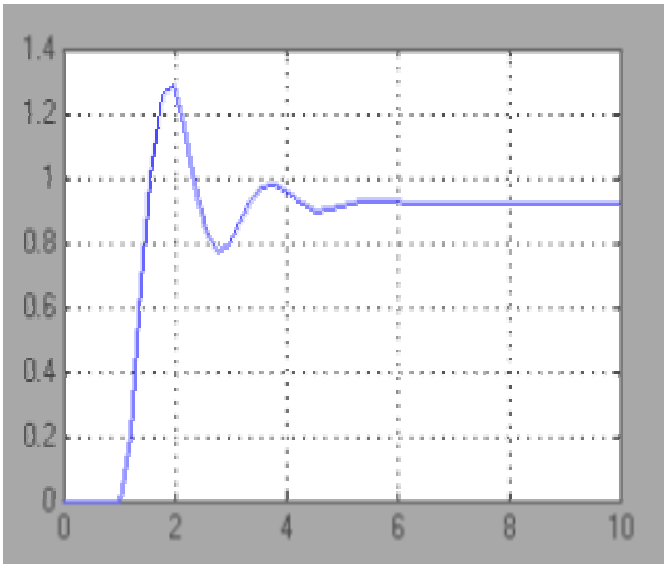
Türev zaman sabiti T_d =1/12 olarak seçilmiştir. Hatanın türevinin alındığı kısımdaki transfer fonksiyonu [s / (0.1*s +1)] ile ifade

edilmiştir. Bunun sebebi sadece [s] ile ifade edildiğinde türev sinyalinin hatanın ilk andaki ani değişimine bağlı olarak sonsuza gitmesidir.

Simulink ortamında sisteme girilen birim basamak fonksiyonu ve değişik durumlarda sistem çıkışları aşağıda verilmiştir.



Şekil 6.2 Sistemin girişi



Şekil 6.3 Sistemin sadece orantısal denetleyiciyle kontrol edildiği durumda yani T_d katsayısının sıfır olduğu durumda gözlenen çıkış sinyali:

Şekil 6.4 Sistemin orantısal ve diferansiyel kontrolü sonucunda gözlenen çıkış sinyali

Şekil 6.3 ve Şekil 6.4'de çıkış sinyalleri incelendiğinde türev kontrolünün sitemin kararlı-hale geçişindeki osilasyonları azalttığını ve bu istenen geçişe daha kısa sürede ulaştığını görüyoruz

Hata sinyalinin, hatanın türevinin ve denetleyici çıkışında gözlenen kontrol sinyalinin zamana bağlı olarak aldığı değerler Tablo 6.1'de verilmiştir.

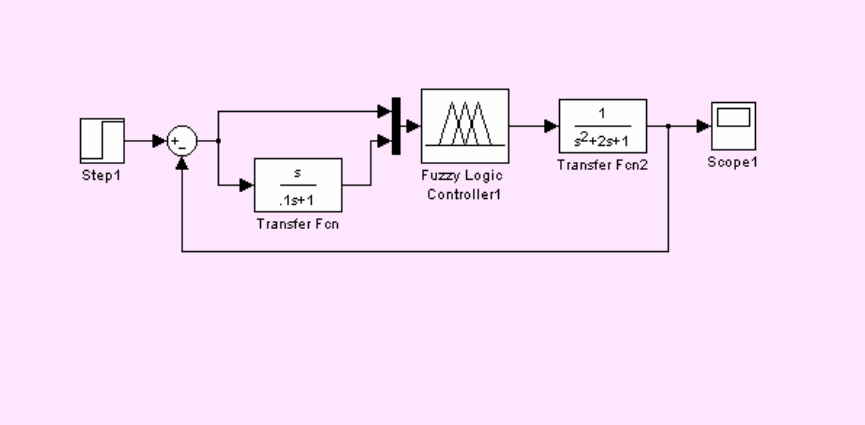
Tablo 6.1 PD Denetleyici Çıkış Değerleri

Time (sec)	Error	Error Derivative	Control Output
1.0	1.00	10.0	22.0
1.1	0.90	3.20	14.0
1.2	0.70	000	8.50
1.3	0.46	-1.45	4.00
1.4	0.25	-2.00	1.20
1.5	0.07	-1.88	-1.10
1.6	-0.07	-1.50	-2.25
1.7	-0.13	-1.05	-2.67
1.8	-0.17	-0.60	-2.63
1.9	-0.16	-0.15	-2.00
2.0	-0.12	0.20	-1.35
2.1	-0.08	0.38	-0.54
2.2	-0.02	0.50	0.30
2.3	0.03	0.50	0.87
2.4	0.08	0.46	1.40
2.5	0.10	0.36	1.65
2.6	0.13	0.25	1.80
2.7	0.14	0.14	1.74
2.8	0.13	0.04	1.65
2.9	0.12	-0.04	1.45
3.0	0.11	-0.10	1.25
3.1	0.10	-0.11	1.08
3.2	0.08	-0.12	1.06
3.2	0.07	-0.10	0.92
3.4	0.07	-0.08	0.82
3.5	0.06	-0.06	0.74
3.6	0.06	-0.03	0.72
3.7	0.06	-0.01	0.72
3.8	0.06	0.01	0.72
3.9	0.06	0.02	0.72
4.0	0.07	0.03	0.88
4.1	0.07	0.03	0.92
4.2	0.08	0.02	0.95
4.3	0.07	0.02	0.96
4.4	0.07	0.01	0.97
4.5	0.07	000	0.97

6.2 2. Bulanık Mantık PD Denetleyici

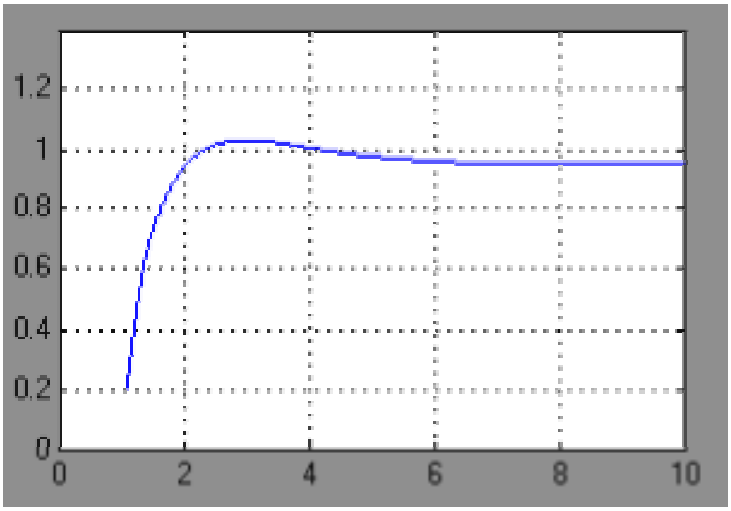
Aşağıda görülen Bulanık Mantık PD denetleyici devresi MATLAB Simulink ortamında kurulmuştur.

Fuzzy Logic Controller (Bulanık Mantık Denetleyici) bloğunun MATLAB Fuzzy Logic Toolbox ortamında tasarlanması bir sonraki alt başlıkta verilecektir.



Şekil 6.5 Bulanık Mantık PD denetleyici blok şeması

Bulanık Mantık PD denetleyici devresinin çıkışı (kontrol sinyali) ve tüm sistemin çıkışı aşağıdaki şekillerde görülmektedir.



Şekil.6.6 Sistemin bulanık mantık PD kontrollü çıkışı.

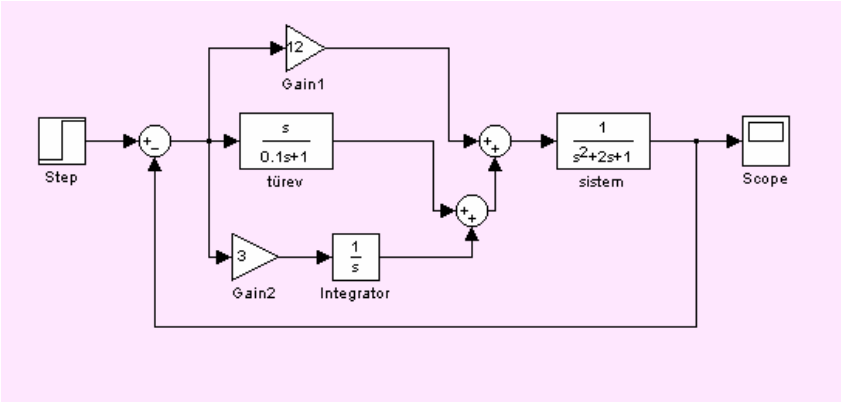
Şekil 6.6 ve Şekil 6.4' de grafiksel olarak gösterilen sırasıyla Bulanık Mantık PD denetleyici kullanılması sonucunda gözlenen sistem çıkışı ve geleneksel PD denetleyici kullanılması sonucunda gözlenen sistem çıkışı arasında farklar vardır.

Burada amaç daha iyi performansta çalışan bir sistem tasarlamak olmadığı halde, sadece bir sistemin giriş ve çıkış değerlerine yani geleneksel PD denetleyicisinin hata, hatanın türevi ve denetleyici çıkış değerlerine bakarak bu sistem MATLAB Fuzzy Logic Toolbox ortamında gerçekleştirilmiş, ve bulanık mantık PD denetleyicisi ile kontrol edilen sistem çıkışının geçiş hali osilasyonları, çıkışın aldığı maximum değer (overshoot) ve kararlı hale geçiş süresinin istenen yönde değiştirilebileceği gösterilmiştir.

Yukarıda anlatılan yöntemler kullanarak parametreleri istenilen özellikleri sağlayacak şekilde ayarlanabilen ve çok iyi performans özellikleri gösteren sistemler tasarlanabilir.

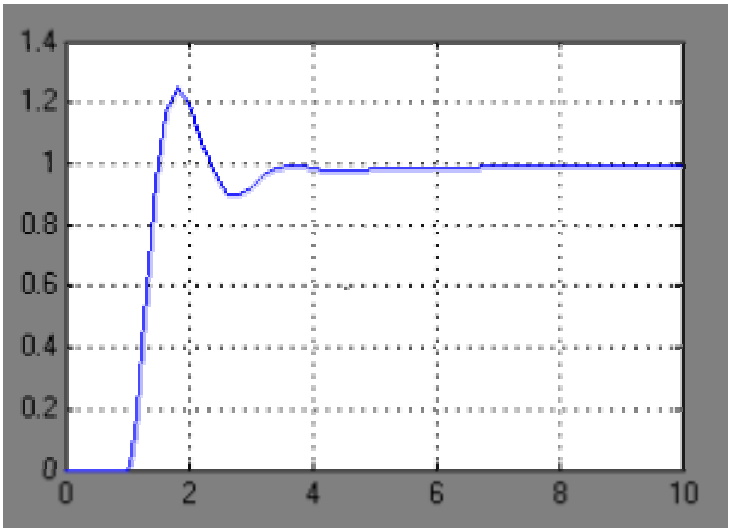
6.2.3 Geleneksel PID Denetleyici

Aşağıda görülen geleneksel PID denetleyici devresi MATLAB Simulink ortamında kurulmuştur.



Şekil 6.7. PID denetleyici blok şeması

Önceden kurulmuş olan PD denetleyicisine integral fonksiyonu ekleyerek oluşturulan PID denetleyicisinin integral zamanı $T_i = 4$ olarak seçilmiştir. Bu durumda elde edilen sistemin çıkış sinyali Şekil 6.8'de verilmiştir.



Şekil 6.8 Sistemin orantısal-integral-diferansiyel kontrollü çıkışı

Şekil 6.8'deki sistem çıkışı Şekil 6.4 ve Şekil 6.5'deki grafikler ile karşılaştırıldığında, integral fonksiyonun sistem çıkışına offset ekleyerek kararlı hal hatasını giderdiği görülmektedir.

Devrenin hata sinyalinin, hatanın türevinin, hatanın integralinin ve denetleyici çıkışında gözlenen kontrol sinyalinin zamana bağlı olarak aldığı değerler Tablo 6.2'de verilmiştir.

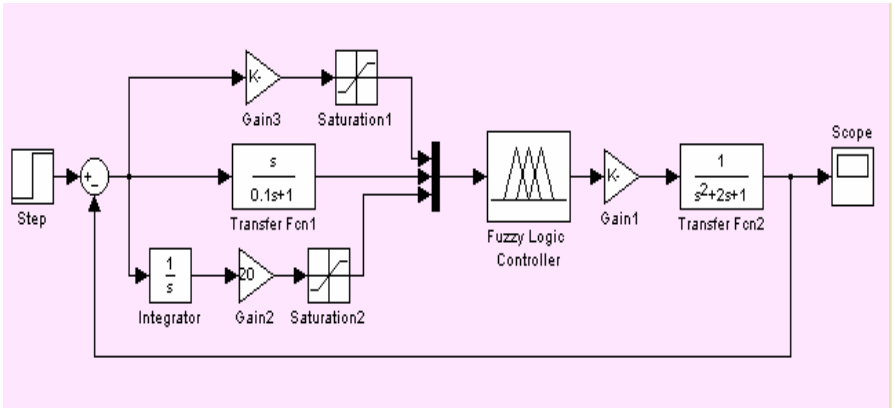
Tablo 6.2 PID denetleyici çıkış değerleri

Time (sec)	Error	Error Derivative	Error Integral	Control Output
1.0	1.00	10.0	000	22.0
1.1	0.90	3.20	0.09	14.5
1.2	0.70	000	0.17	9.00
1.3	0.46	-1.50	0.23	5.00
1.4	0.23	-2.00	0.26	1.50
1.5	0.03	-2.00	0.28	-0.80
1.6	-0.11	-1.60	0.24	-2.10
1.7	-0.19	-1.16	0.26	-2.67
1.8	-0.24	-0.64	0.25	-2.35
1.9	-0.23	-0.20	0.27	-2.33
2.0	-0.02	1.16	0.29	-1.65
2.1	-0.14	0.38	0.27	-0.80
2.2	-0.08	0.55	0.16	0.10
2.3	-0.02	0.56	0.16	0.73
2.4	0.03	0.53	0.18	1.37
2.5	0.06	0.43	0.18	1.70
2.6	0.09	0.30	0.17	1.90
2.7	0.10	0.18	0.18	1.94
2.8	0.10	0.05	0.19	1.86
2.9	0.09	-0.04	0.20	1.67
3.0	0.08	-0.11	0.21	1.48
3.1	0.06	-0.14	0.22	1.26
3.2	0.04	-0.16	0.23	1.06
3.2	0.03	-0.15	0.24	0.92
3.4	0.02	-0.12	0.23	0.80
3.5	0.01	-0.09	0.23	0.75
3.6	0.01	-0.06	0.23	0.73
3.7	0.01	-0.02	0.24	0.72
3.8	0.01	-0.01	0.25	0.80
3.9	000	000	0.25	0.85
4.0	000	000	0.26	0.90
4.1	000	000	0.27	0.95

4.2	000	000	0.27	1.00
4.3	000	000	0.28	0.99
4.4	000	000	0.29	1.00
4.5	000	000	0.30	1.00

6.2.4 Bulanık Mantık PID Denetleyici

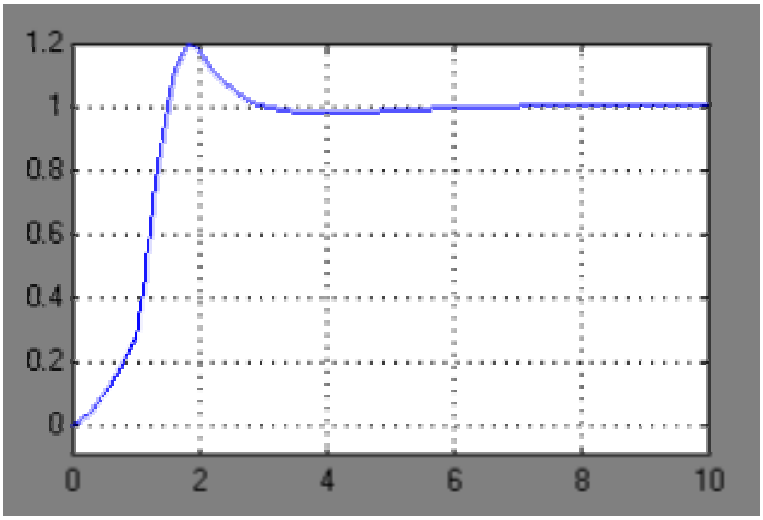
Aşağıda görülen Bulanık Mantık PID denetleyici devresi MATLAB Simulink ortamında kurulmuştur.



Şekil 6.9 Bulanık mantık PID denetleyici blok şeması

Devrede Gain1 (Kazanç1) blok değeri 0.85, Gain2 (Kazan2) blok değeri 20 ve Gain3 (Kazanç3) blok değeri 1.5 olarak seçilmiştir. Saturasyon değerleri ise Tablo 6.2'deki hata ve hatanın integrali değişkenlerinin değer aralığı göz önüne alınarak belirlenmiştir. Buna göre Saturasyon1 bloğunun değer aralığı [-0.24,1] ve Saturasyon2 bloğunun değer aralığı [0,0.3] olarak alınmıştır. Bu ilave bloklar sistemin çıkış değerlerinin sonsuza gitmesini önlemek ve daha iyi performans elde edebilmek içindir.

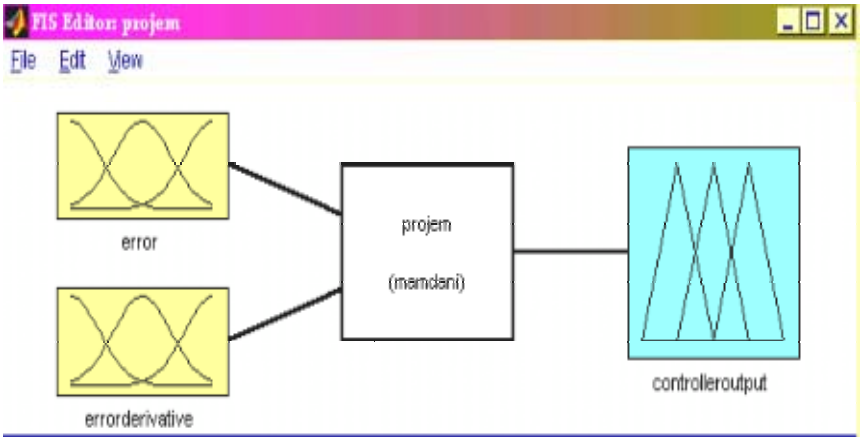
Bulanık Mantık PID denetleyici ile kontrol edilen sistemin çıkışı aşağıdaki şekilde görülmektedir.



Şekil 6.10 Sistemin Bulanık Mantık PID kontrollü çıkışı

6.2.5 Bulanık Mantık PD Denetleyicinin MATLAB Fuzzy Logic Toolbox Ortamında Gerçekleştirilmesi

Matlab ortamında komut satırına "fuzzy" yazılarak ulaşılan Fuzzy Logic Toolbox açıldıktan sonra, giriş ve çıkışı değişkenleri şekilde görüldüğü gibi belirlenip adlandırılmıştır.



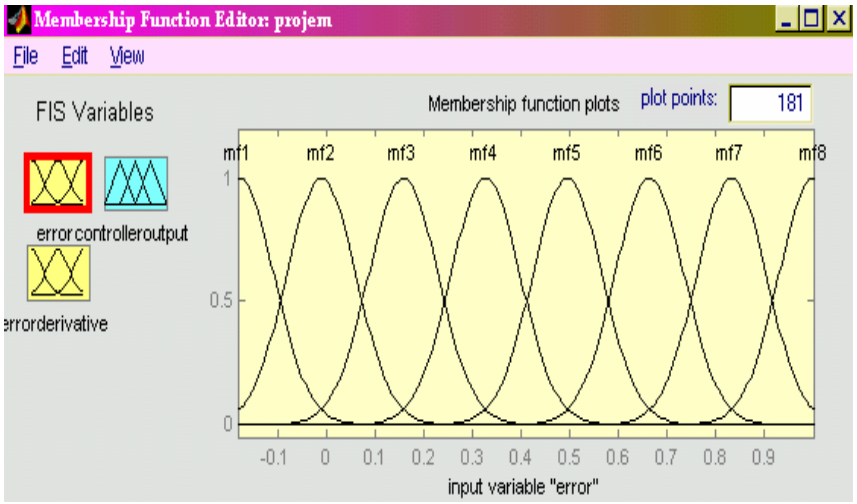
Şekil .6.11 Bulanık mantık giriş çıkış değişkenleri

Sisteme verilen her bir giriş için üyelik fonksiyonu tanımlama gereği vardır. Sistemde kullanılan üyelik fonksiyonları gauss

fonksiyonlarıdır. FIS ortamının kullanımına olanak tanıdığı başka üyelik fonksiyonlar da mevcuttur. Üyelik fonksiyonlarının FIS ortamında oluşturulması şekillerle anlatılmıştır.

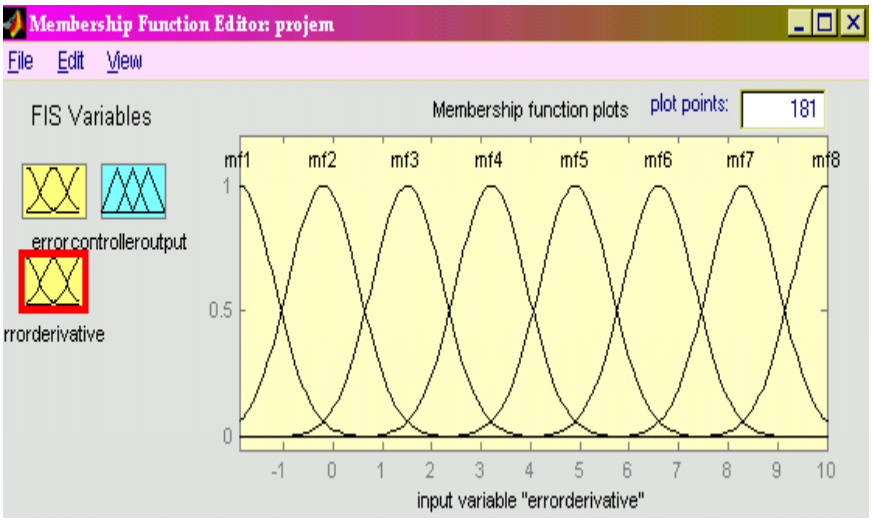
Bu üyelik fonksiyonları oluşturulurken Tablo 6.1'deki değerler göz önüne alınmıştır.

FIS ortamında error (hata) değişkenine çift tıklayarak ulaşılan Üyelik Fonksiyonu Editörü (Membership Function Editor) aşağıdaki şekilde gösterilmiştir.



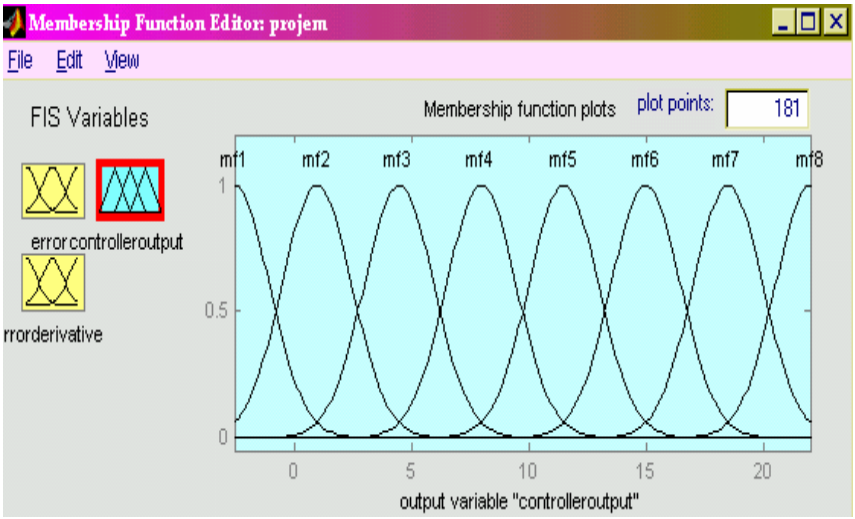
Şekil 6.12 FIS editörü ve hata

FIS ortamında error derivative (hata türevi) değişkenine çift tıklayarak ulaşılan Üyelik Fonksiyonu Editörü (Membership Function Editor) aşağıdaki şekilde gösterilmiştir;



Şekil 6.13 FIS editörü ve hatanın türevi

FIS ortamında controller output (kontrolör çıkışı) değişkenine çift tıklayarak ulaşılan Üyelik Fonksiyonu Editörü (Membership Function Editor) aşağıdaki şekilde gösterilmiştir;

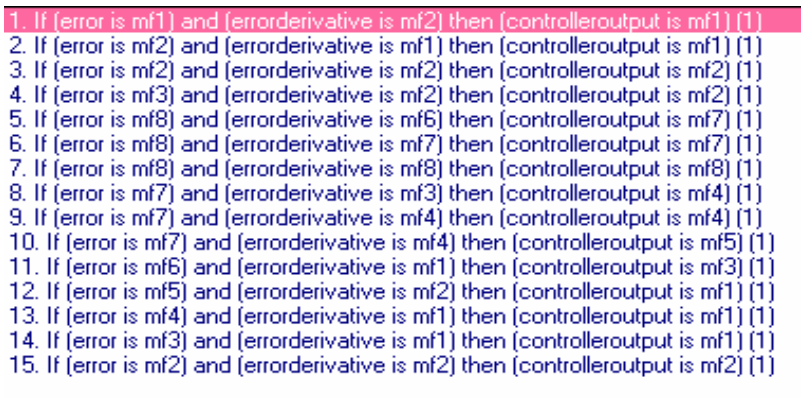


Şekil 6.14 FIS editörü ve kontrolör çıkışı

Girişimizin istediğimiz çıkış değerleriyle ilişkisi, belirlediğimiz kurallarla olur. Bulanık mantık kuralları geleneksel PD kontrolör yapısına göre hazırlanır ve sadece dizayn için gereklidir.

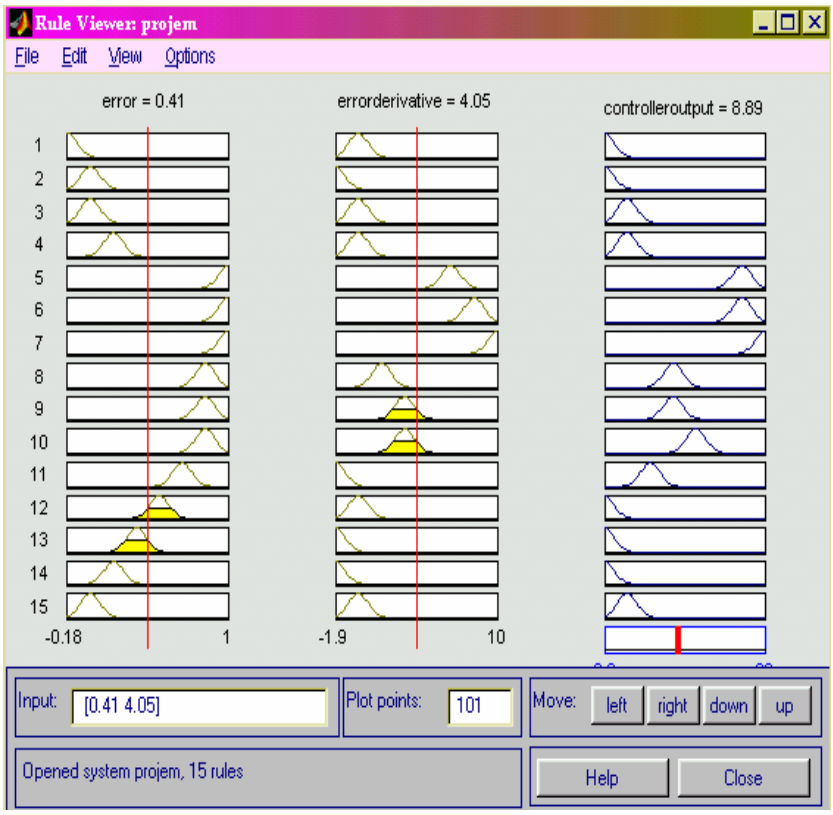
Tablo 6.1'deki değerler belirli zaman aralıklarına göre eşleştirilir ve bu değerlerin ait olduğu üyelik fonksiyonlarına göre eşleştirme FIS ortamına kurallar aracılığıyla taşınır.

FIS ortamında FIS editörü üzerindeki Edit menüsü açılıp Kural Editörü (Rule Editor)'ün seçilmesiyle ulaşılan Rule Editor aşağıdaki şekilde gösterilmiştir;



Şekil 6.15 Kural Editörü

Kurallar yazılırken AND (ve) bulanık operatörü kullanılmıştır. Bu operatörün kullanılması sonucunda kuralların yorumlanmasını Kural İzleyici (Rule Viewer) yardımıyla görebiliriz. FIS editörü üzerindeki View menüsü açılıp Kural İzleyici (Rule Viewer)'nin seçilmesiyle ulaşılan Rule Viewer aşağıdaki şekilde gösterilmiştir.

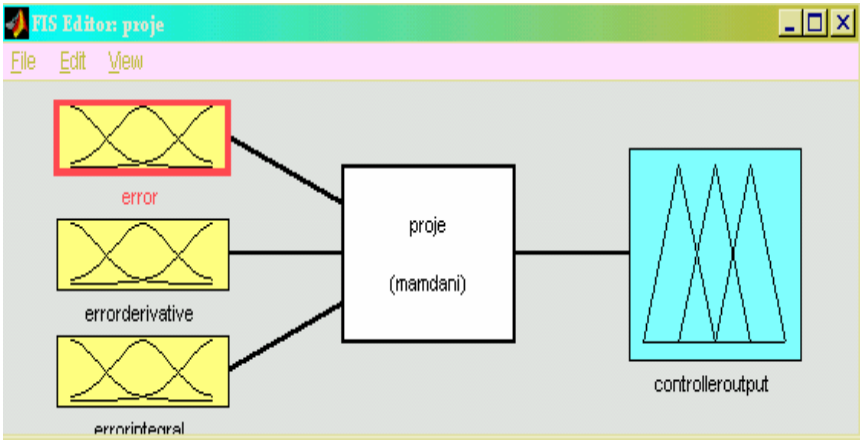


Şekil 6.16 Kural izleyici

6.2.6 Bulanık Mantık PID Denetleyicinin MATLAB Fuzzy Logic Toolbox Ortamında Gerçekleştirilmesi

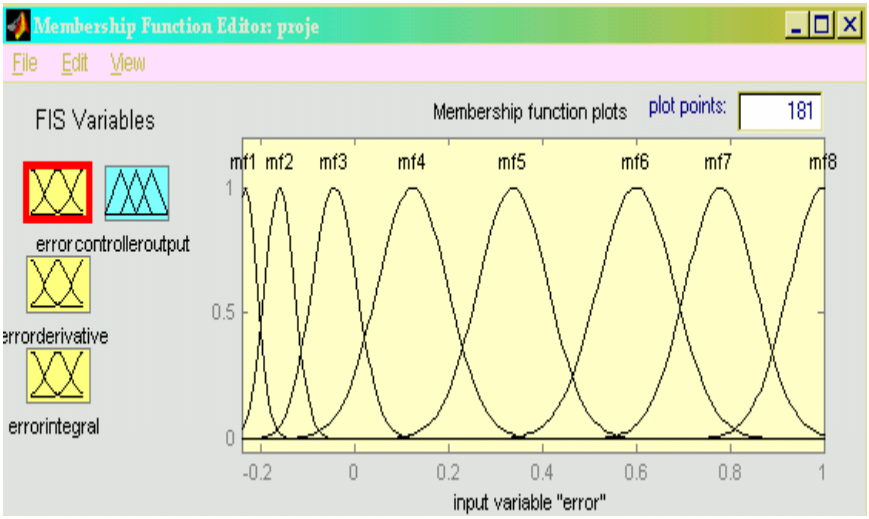
Bulanık Mantık PID denetleyici oluşturulurken yapılan işlemler, Bulanık Mantık PD denetleyici ile aynıdır aralarındaki tek fark Tablo 6.2'den yararlanılmış olmasıdır.

FIS ortamında oluşturulan yapı aşağıda verilmiştir.

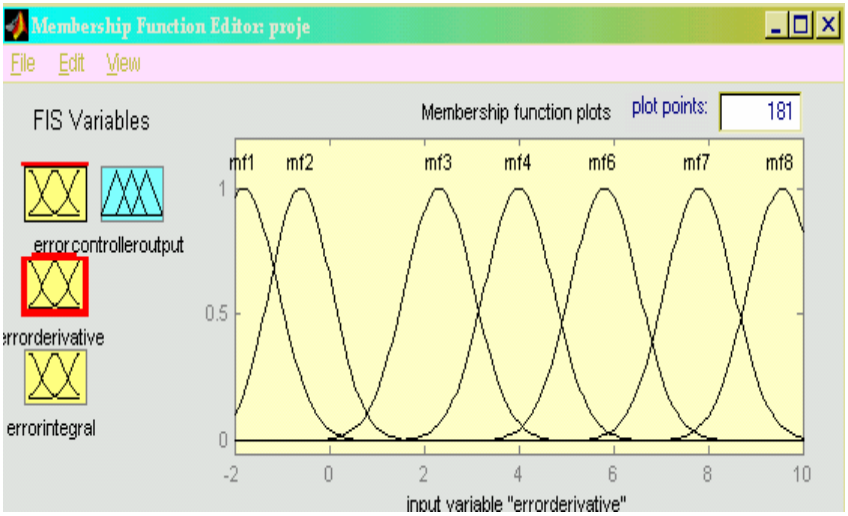


6.17 FIS ortamında oluşturulan yapı

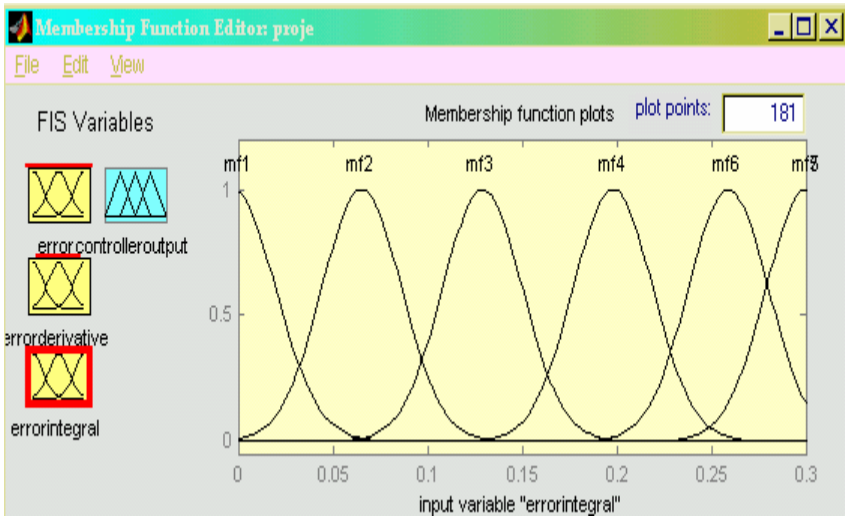
Sistemin hata sinyalinin, hatanın türevinin, hatanın entegralinin ve denetleyici çıkışında gözlenen kontrol sinyalinin üyelik fonksiyonları sırasıyla Şekil 6.18, Şekil 6.19, Şekil 6.20, Şekil 6.21 de verilmiştir. Bulanık Mantık PD denetleyiciden farklı olarak daha iyi sonuçlar elde edebilmek için üyelik fonksiyonlarının şekillerinde değişiklikler yapılmıştır. FIS ortamında bu değişiklikler kolaylıkla yapılabilmektedir.



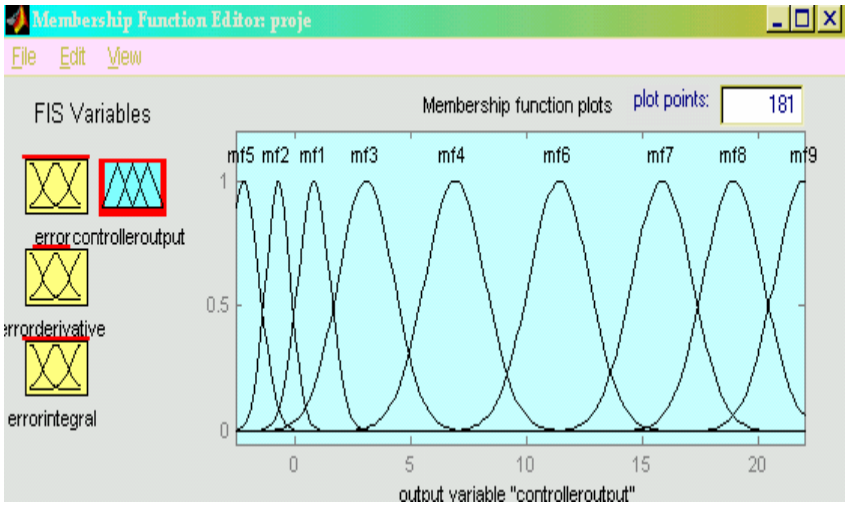
6.18 FIS editörü ve hata



6.19 FIS editörü ve hatanın türevi



6.20 FIS editörü ve hatanın integrali

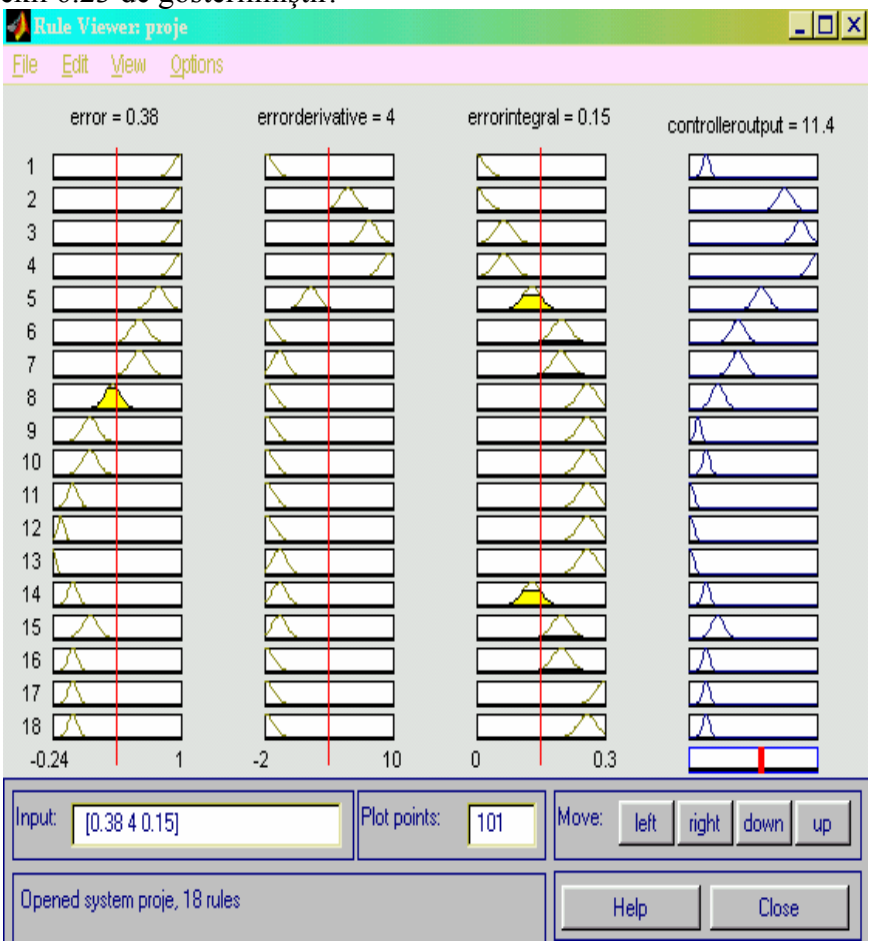


Şekil 6.21 FIS editörü ve kontrolör çıkışı

Kurallar aşağıdaki listede belirtilmiştir:

Şekil 6.22 Kural editörü

Kuralların FIS ortamında yorumlanması ise Rule Viewer yardımıyla Şekil 6.23 de gösterilmiştir.



Şekil 6.23 Kural izleyici

Faydalı Linkler

About Fuzzy Logic Inc. - website services:
http://www.fuzzylogic.ca/about.html

Adaptive Model Predictive Control Using Neural Networks: http://www.xdiv.lanl.gov/XCM/neural/chemical_proc/baum_talk/baum_talk.html

Artificial Intelligence, Knowledge Management and more - a CompInfo Directory: http://www.compinfo-center.com/tpai-t.htm

Bart Kosko : http://sipi.usc.edu/~kosko/

Benchmark Group on Data Modeling
http://neural.cs.nthu.edu.tw/jang/benchmark/

Bruno Di Stefano's Homepage @ UofT - Bruno Di Stefano's Fuzzy Logic Links: http://www.ecf.utoronto.ca/~bruno/fzlnk.html

comp.ai.fuzzy
http://groups.google.com/groups?hl=tr&lr=&ie=UTF8&oe=UTF8&group=comp.ai.fuzzy

Course Notes and Learning Resources - Process Control:
http://lorien.ncl.ac.uk/ming/Dept/Swot/connotes.htm

Course Outline - Neuro-fuzzy and Soft Computing:
http://www.uoguelph.ca/~syang/ENGG4430/outline.htm

Dr. M. Kudra's Fuzzy Page: http://www.uni-leipzig.de/~kudra/fuzzy/

FAQ Fuzzy Logic and Fuzzy Expert Systems:
http://www.cs.cmu.edu/Groups/AI/html/faqs/ai/fuzzy/part1/faq.html

FLLL - A brief course in Fuzzy Logic:
http://www.flll.uni -linz.ac.at/pdw/fuzzy/fuzzy.html

From Semi-Heuristic Fuzzy Techniques To Optimal Fuzzy Methods Mathematical Foundations and Applications (Research Index): http://citeseer.nj.nec.com/34960.html

Fuzzy archive By Thread:
http://www.dbai.tuwien.ac.at/marchives/fuzzy-mail/index.html

Fuzzy Logic Archive: http://www.austinlinks.com/fuzzy/

Fuzzy Logic, partial truth:
http://www.eg3.com/WebID/elect/fuzzy/blank/book/1-a-f.htm

Fuzzy Logic General: http://www.cms.dmu.ac.uk/~rij/general.html

Fuzzy Logic What is Fuzzy Logic:
http://www.emsl.pnl.gov:2080/proj/neuron/fuzzy/what.html

Fuzzy Systems Engineering: http://www.fuzzysys.com/

Fuzzy Systems Resources: http://www.nwcil.pdx.edu/fuzzy/

Fuzzy TECH Home Page : http://www.fuzzytech.com/

IFSA: http://www.abo.fi/~rfuller/ifsa.html

Internet's Resources for Neuro-fuzzy and Soft Computing :
http://www.cs.nthu.edu.tw/~jang/nfsc.htm

Jerry M. Mendel: http://sipi.usc.edu/~mendel/

Li-Xin WANG: http://www.ee.ust.hk/~eewang/

Neural Fuzzy Systems: http://www.abo.fi/~rfuller/nfs.html

Neuro-Fuzzy and Soft Computing:
http://neural.cs.nthu.edu.tw/jang/book/

Neuro-Fuzzy Computing in Automation - Course Material:
http://www.control.hut.fi/Kurssit/AS-74.115/Material/

Open Directory - Computers Artificial Intelligence Fuzzy:
http://dmoz.org/Computers/Artificial_Intelligence/fuzzy/

Ortech Engineering's Fuzzy Logic Reservoir:
http://www.ortech-engr.com/fuzzy/reservoir.html

Roger Jang's Home Page: http://neural.cs.nthu.edu.tw/jang/

The Control Engineering Virtual Library Conferences Page:
http://www-control.eng.cam.ac.uk/extras/conferences/conferences.html

WOB List of Examples: http://www.bifurcation.de/Examples.html

Bulanık Mantık Ve Sistemler Forumu
http://farabi.selcuk.edu.tr/egitim/bulanik/forums/

İTÜ Bulanık Mantık ve Teknolojileri kulübü
http://www.bumat.itu.edu.tr/

Matlab official web site: http://www.mathworks.com

Useful Matlab M-Files and Toolboxes: http://www.mathtools.net

Bulanık Mantıkla ilgili Kaynak Kitaplar ve Makaleler

A Course in Fuzzy Systems and Control, Li-Xin Wang, 1997, Prentice Hall.

Neuro-Fuzzy and Soft Computing, J.-S.R. Jang, C.-T. Sun, E. Mizutani, 1997, Prentice Hall.

Adaptive Fuzzy Systems and Control, Li-Xin Wang, 1994, Prentice Hall.

Fuzzy Engineering, Bart Kosko, 1997, Prentice Hall.

Matlab Supplement To Fuzzy and Neural Approaches in Engineering, J.W. Hines, 1997, John Wiley & Sons Inc.

L. A. Zadeh, Fuzzy Sets, Info S. Ctl Vol. 8, 1965, sayfa 338 – 353

L. A. Zadeh, Fuzzy Algorithms, Info S. Ctl. Vol.12, 1968, sayfa 94-102

L. A. Zadeh, Making computers think like people, IEEE Spectrum, Vol.8 1984, sayfa 26 – 32

Neural Networks and Fuzzy Systems, Bart Kasko, Prentice Hall, 1991

Fuzzy Sets and Applications: Selected Papers by L. A. Zadeh, R. R. Yager vd., Wiley-Interscience, 1987

Dizin

A

ABS fren sistemi, 11
adaptive, 8
arttırılmış bulanık sistem, 73

B

BFGS, 73
bulanık inference engine, 12
bulanık küme, 17, 20
bulanık küme birleşimi, 24
bulanık küme kesişimi, 25
bulanık kural tabanı, 31
bulanık mantık PD denetleyici, 98
bulanık mantık PID denetleyici, 102
bulanık sistem, 12
bulanıklaştırıcı, 32

C

center average defuzzifier, 35
center of average, 33
center of gravity, 34
clustering, 85

D

defuzzifier, 13, 33
dinamik sistem, 67
doğrusal arama, 73
durulayıcı, 33

E

editör, 93
en küçük kareler, 76

F

FIS, 93
fonksiyon eğitme, 57
fonksiyon tanımlanması, 90
fuzzifier, 12, 32
fuzzy logic toolbox, 94, 103

G

gaussian üyelik fonksiyonu, 54
gradient descent, 54, 76
gradyen tabanlı eğitme, 54
grafiksel kullanıcı arabirimi, 93
graphical user interface, 93
GUI, 93

H

hata oranı, 56

I

IF-THEN, 29, 39

K

kararlı hal, 99
klasik küme, 17, 18
kümeleme, 87
kural izleyici, 112

L

Larsen, 37
Li-Xin Wang, 60, 87

M

Mackey-Glass, 41
Mamdani, 31

O

olasılık, 17
on line, 67
önerme, 30
optimizasyon, 73

P

parametre atama, 67
parametre güncelleme, 78
PD denetleyici, 94
performans indeksi, 76
PID denetleyici, 99, 108

R

recursive least squares, 76
RLS, 81

S

s norm, 26
s normları, 24
serbest parametre, 56
sistem tanımlama, 73
sözsel değişken, 29

T

t norm, 26
t normları, 25
table look-up, 39

U

üyelik değeri, 17
üyelik fonksiyonu, 19

Z

Zadeh, 37
zaman serisi, 41