T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

JAVA ORTAMINDA BULANIK MANTIK KONTROL: KAMYON YÜKLEME-BOŞALTMA UYGULAMASI

Ömer KARAL

Yüksek Lisans Tezi

DENİZLİ-2004

JAVA ORTAMINDA
BULANIK MANTIK KONTROL:
KAMYON YÜKLEME-BOŞALTMA UYGULAMASI

Pamukkale Üniversitesi
Fen Bilimleri Enstitüsü
Tarafından Kabul Edilen
Elektrik-Elektronik Mühendisliği
Yüksek Lisans Tezi

Ömer KARAL

Tez Savunma Tarihi:

DENİZLİ-2004

TEŞEKKÜR

"Java Ortamında Bulanık Mantık Kontrol: Kamyon Yükleme-Boşaltma Uygulaması " konulu tez çalışmasının bu aşamaya gelmesinde yardımlarını esirgemeyen, değerli hocam Yrd. Doç. Dr. Ahmet ÖZEK'e; manevi destekleri ile çalışma azmimi arttıran aileme, akademik hayata devam etmemi özendiren değerli hocam, Prof. Dr. Mustafa TEMİZ'e, tezimi büyük titizlikle ve sabırla inceleyip düzeltmelerini gösteren değerli hocam Yrd. Doç. Dr. Abdulkadir YALTIR'a tezimin hazırlanmasında gerek kaynak gerekse de bilgi desteğini esirgemeyen değerli hocam Yrd. Doç. Dr. Sezai TOKAT'a, tezimde kararlılık analizinin yapılmasına katkı sağlayan değerli hocam Yrd. Doç. Dr. Serdar İPLİKÇİ'ye program konusunda tecrübelerini bana aktaran değerli arkadaşım Bilgisayar Mühendisi Dursun GÜNDOĞAN'a tez çalışmam boyunca yardımlarını gördüğüm ev ve yüksek lisans arkadaşlarıma teşekkür ederim.

Ömer KARAL

ÖZET

Bulanık Mantık Kontrol; anlaşılması ve geliştirilmesi kolay olan ve endüstride çalışan mühendislerin tercih edebilecekleri bir kontrol yöntemidir. Nitekim günümüzde birçok tüketici ürünlerinde ve endüstriyel uygulamalarda Bulanık Mantık Kontrol yöntemleri uygulanmaktadır.

Son yıllarda bilgisayarların, özellikle kişisel bilgisayarların yaygınlaşması ile kontrol sistemlerinin analizi önemli bir derecede gelişmiştir. Bilgisayar sistemlerinin fabrika ortamında çalışmasını sağlamak için kontrol mühendisliği ile birleştirilmesi gerekmektedir. Bu, ancak güçlü ve güvenilir bir yazılımla gerçekleştirilir. Bunu sağlayacak olan yazılım kaynaklarından biri de Java programlama dilidir

Yazılım programında hedeflenen, kullanıcı tarafından belirlenecek olan Bulanık Mantık Kontrol parametrelerinin rahatça girilebilmesini sağlamak, zaman içerisinde bunlar üzerinde değişiklik yapabilmek ve hızlı bir şekilde kontrolü gerçekleştirip kontrol sonuçlarının simulasyon ortamanına taşınabilmesini sağlamaktır.

Java hem yazılım hem de donanım boyutu olan, teknolojisi ile ticari bilgisayar dünyası ve endüstriyel kontrol uygulamalarının ihtiyaçlarını karşılayabilecek duruma gelmiştir.

Bu çalışmada, matematiksel modelinin oluşturulmasının zor ve karmaşık olduğu geriye hareket eden kamyon yükleme-boşaltma sisteminin Bulanık Mantık ile Kontrolü incelenmiştir. Yapılan kontrol sonuçlarını incelemek için, nesne tabanlı Java2 programlama dilinde sisteme ait toolbox oluşturulup, simülasyon gerçekleştirilmiştir.

Simulasyon olarak gerçekleştirilen kamyon yükleme-boşaltma sisteminde, iki adet giriş ve bir adet çıkış kontrol değişkeni vardır. Giriş kontrol değişkenlerine bağlı olarak, çıkışın bütün durumları gözönüne alınmış ve işlemlerin gözlenebilmesi için yazılan programa arayüz eklenmiştir. Ayrıca, sistemde farklı bulanık içermeler kullanılarak sistem performansına etkisini grafik konum hatası olarak çizen bir arayüz ile gösterilmiştir.

Ömer KARAL

ABSTRACT

Fuzzy Logic Control is easy to understand and develop, thus it is preferable by engineers who work on industrial areas. As a matter of fact; today we can see Fuzzy Logic Control in industrial applications and customer products.

Nowadays with the increasing usage of computers and especially personel computers, the analysis of control systems is positively effected. In order to work with computer systems in industrial areas one must connect with control engineering. This can be done only by a powerful and safe software one of which is JAVA programming language.

The purpose of the proposed software is to provide the user to adjust the fuzzy control parameters easily, to control the system rapidly and to move the controlling results to simulation.

JAVA has both software and hardware parts, with this technology commercial computer world is enough to cover all needs of industrial control applications.

In this study; fuzzy logic control of a back-driving truck system which has a difficult and complex mathematical model is examined. In order to examine the results of control process; an object oriented JAVA2 programming language is used to construct a toolbox of the system and then simulation is done.

Back-driving a truck system is a kind of simulation which has two inputs and one outuput. Depending on input values, all values of the output is taken into account and to observe the process an interface is added to program. However; different fuzzy implications are used in the control system and the effect of this to the system performance is observed and an interface which shows position error graphically is added.

Ömer KARAL

İÇİNDEKİLER

Birinci Bölüm

GİRİŞ

İkinci Bölüm

BULANIK MANTIK TEORİSİ ve BULANIK MANTIK KONTROL

Üçüncü Bölüm

NESNE TABANLI PROGRAMLAMA ve JAVA

Dördüncü Bölüm

BULANIK MANTIK KONTROL İLE KAMYON YÜKLEME-BOŞALTMA TASARIMI ve JAVA ORTAMINDA UYGULAMASI

Beşinci Bölüm

SONUÇLAR ve ÖNERİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

BİRİNCİ BÖLÜM
GİRİŞ

1.1 Otomatik Kontrol Sistemleri ve Bilgisayar Dünyası

Günlük hayatta belirli amaçlara erişmek için kontrol içeren çeşitli faaliyetler ve hareketler yapılır. Belirli hedeflere ulaşmak için kontrol stratejilerini gözeterek herhangi bir dış etki olmaksızın kendi kendine ayarlanan kontrol sistemlerine otomatik kontrol sistemleri denir. Kontrol mühendisliğinin zengin kavramları ile tümleşik entegre teknolojisinin gücü birleşince mühendisler için yeni tasarım fırsatları ortaya çıkmaktadır.

Hızla gelişen sanayi ile birlikte giderek artan verimlilik, güvenlik ve güvenilirlik gereksinimleri otomatik kontrol için geniş uygulama alanları sağlamıştır. Böylece otomatik kontrol uygarlığın gelişme ve ilerlemesinde önemli rol oynayan bir bilimdalı haline gelmiştir. Endüstride, modern araç ve gereçlerde otomatik kontrol sistemlerinin sayısız uygulamaları vardır. Örneğin ürün kalite kontrolü, güdümlü araçlar, görüntü işleme, trafik sinyalizasyonu ve sanayi robotları bunlardan sadece birkaçı olarak sayılabilir (Asherton, 1998).

Son yıllarda bilgisayarların, özellikle kişisel bilgisayarların yaygınlaşması ile kontrol sistemlerinin analizi önemli bir derecede etkilenmiştir. Kişisel bilgisayarlar o kadar güçlü ve gelişmiş hale gelmiştir ki, anlaşılması zor ve karmaşık kontrol sistemlerinin çözümünde bile kolaylıkla kullanılabilmektedir. Bugün üniversite ve şirketlerin çoğu kolay erişilebilir kişisel bilgisayarlarla donatılmıştır (Kuo,1999).

Bilgisayar, yazılım ve donanım bileşenlerinden oluşmaktadır. Yazılım ve donanım arasında ise karşılıklı bir ilişki vardır. Endüstriyel donanımlar hızlı ve pahalı iken, yazılım ile elde edilen çözümler ucuz fakat daha yavaş olmaktadır. Genel olarak sistem tasarımında, her

geçen gün yazılımın öneniminin donanıma göre arttığı maliyet oranlarındaki değişime bakılarak söylenebilir (McConnel, 1999).

Güvenilirlik ve güvenlik, yazılım mühendisliğinin ana hedefi olmaya yeni başlamışken, kontrol mühendisliğinin sürekli olarak kalbinde yer almıştır. Tarihsel olarak, kontrol mühendisliği, üretim işlemleri ile birlikte gelişmiştir. Kontrol edilemeyen bir yöntemle üretim yapılamaz. Basınç, sıcaklık, akış, malzeme özellikleri, ve diğer değişkenler, güvenli ve güvenilir üretim için hayati kriterlerdir. Etkinlik ve en iyileme ancak bu gibi kriterler sağlandıktan sonra dikkate alınabilir (Asherton, 1998).

1950'lerden önce, üretim güvenliği, elektrik, mekanik ve hidrolik kontrol sistemlerine dayalıydı. Genel amaçlı ticari bilgisayarların kontrol için iyi bir araç olup olmadığı sorusunun cevabı, on yıllar boyunca, "Hayır" oldu. Kontrol mühendisleri, küçük gruplara yönelik, özelleşmiş, ve genelde pahalı sistemler tasarladılar. Bilgisayar sistemlerinin fabrika ortamında çalışmasının güç olması da buna eklenince, bilgisayar dünyası ile kontrol mühendisliğinin yolları ayrılmış oldu. Bu yolların tekrar birleşebilmesi için güvenilir bir yazılım kaynağına ihtiyaç duyulmaktadır. Bunu sağlayacak olan yazılım kaynaklarından biri de Java programlama dilidir (Asherton, 1988).

1.2 Java

Java, hem yazılım hem de donanım boyutu olan teknolojisi ile ticari bilgisayar dünyası ve endüstriyel kontrol uygulamalarının ihtiyaçlarını karşılayabilecek duruma gelmektedir. Daha üç yıl önce, Sun Microsystems tarafından tanıtılan Java teknolojisi, kurumsal birleşmenin ve endüstriyel kontrolün ihtiyaçlarını karşılayabilecek bir Bilişim Teknoloji mimarisi sunmaktadır. Dolayısıyla, bütün iş ortamlarına (hem ofislere, hem de fabrikalara) bir bütün olarak hitap etmektedir (Aptech, 2003).

Java teknolojisinin ana elemanları, Java programlama dili, Java işletim ortamı ve Java temelli işlemcilerdir. Bunlar, Internet bütünleşmesinin yanı sıra platform bağımsızlığını da sağlamaktadır. Bu özellikler, Java'yı endüstriyel kontrol ve kontrol sistemleri geliştirme uygulamaları için güçlü bir platform haline getirmektedir.

Java, Web üzerinden çalıştırılan uygulamalar geliştirmede kullanılabilen platformdan bağımsız bir programlama dilidir. Diğer nesne tabanlı programlama (Object Oriented Programming-OOP) yöntembilimlerinin eksikliklerine sahip olmayan bir nesne tabanlı programlama yapısıdır. Java'yı yaratan ekip, C++ ile elde edilen tecrübelerine dayanarak, sorun çıkartabilecek olan yapıları, işaretçileri (pointers), belleğe doğrudan erişimi, çoklu miras almayı (multiple inheritance), genişletilmiş yapıcıları (extended constructors) kaldırmışlardır (Schildt, 2003).

Dile başka hayati özellikler eklenerek, bu yapıların kaldırılmasından kaynaklanan zayıf yönlerden arındırılmış olur. Java, istisna durumlarının ele alınışı (exception handling) yönünden daha güçlüdür. Böylece, önemli ya da beklenmedik durumların ortaya çıkması halinde program akışını yönlendirmek kolaylaşmıştır. Ayrıca, "çöplük toplama" (garbage collection) adı verilen bir sistemi de otomatikleştirerek programcının işini kolaylaştırmaktadır. C ya da C++ ile program yazılırken, dinamik olarak yerleştirilen bellek, "free" ya da "delete" gibi bir komutla serbest bırakılmak zorundayken, Java'da çöplük toplayıcı bunu kendisi yapmaktadır.

Java ayrıca, çok parçalı programların çalıştırılması için daha iyi destek vermektedir. Büyük bir uygulama oluşturan küçük parçalar, birbirlerinden bağımsız olarak çalışabilmekte ve aynı anda birden fazla parçacıkla haberleşebilmektedir. Java'nın ayrıca sağlam bir güvenlik modeli ve Web bağlantısı için mevcut desteği bulunması da diğer artılarıdır. (Schildt, 2003).

Java'nın evrensel uygulama alanı bulmasının anahtarı, Java Sanal Makinesi (Java Virtual Machine JVM) ve Java sınıfları ve uygulama programlama ara yüzleri (Application Programming Interface - API) olan iki yönlü mimarisidir. Sanal Makine, kullanıcı platformunda yer almakta ve Java hizmetçisi ile kullanıcının işletim sistemi arasında kalmaktadır. Sınıflar ve uygulama programlama ara yüzleri veritabanı bağlanabilirliğinden güvenlik ve hizmetçi yönetimine uzanan bir çok işlev ve yetenek sunmakta ve uygulamalar ve Java platformu arasında durmaktadır. Bu elemanların çalışma biçimi, Java ile hazırlanan uygulamaların, diğer işlemci platformlarına kolaylıkla taşınabilmesine olanak sağlamaktadır. Java kaynak kodu öncelikle bir ara seviye olan Java Bayt Koduna (JBC) çevrilir. Bayt Kodu, her hangi bir platforma özel yöntemler içermez ve sadece Java'yı destekleyen bir platformda,

Java Sanal Makinesi tarafından o platforma özgü bir şekilde çalıştırılır. Kısaca, uygulamanın çalıştırılırken derlendiğini söylemek mümkündür (Aptech, 2003).

Bayt Kodları, bir Java Sanal Makinesinin bulunduğu her platformda çalışabilir. Sanal Makine, kullanıcı ara yüzü, dosya sistemi işlemleri, ağ kullanımı gibi tüm platforma bağımlı hizmetleri gizler. Bu özellikleri ile Java hem bir nesne tabanlı dil, hem de bir çalışma zamanı ortamı (run-time environment) olarak değerlendirilmelidir.

Java teknolojisi mevcut müşteri/hizmetçi uygulamalarına göre bir çok üstünlük içermektedir. Java uygulamaları herhangi bir platformda çalışabilir. Bu durum yazılım hazırlanmasını kolaylaştırmaktadır ve sistemler yukarıdan aşağıya ya da aşağıdan yukarıya ölçeklenebilmektedir. Bu ise sistem tasarımını her kuruluşun ihtiyacına göre yapabilmesini sağlamaktadır. Java'nın bu özellikleri, heterojen bilgisayar sistemleri ve çok farklı platformların kullanıldığı fabrika ortamı için avantajlar sunmaktadır. Nesnelerden oluştuğu için var olan Java yazılımı yeni bir uygulama için kullanılabilmektedir. Bu, sadece işgücü ve zamandan tasarruf etmek dışında, endüstriyel uygulamaların gerektirdiği güvenlik ve güvenilirlik koşullarını da sağlamaktadır (GrupJava, 2002).

Tümleşik kontrol sistemleri daha büyük bir sistemin özel amaçlı parçalarıdır. Bir çok açıdan programlanması en zor olan parçalardır. Çünkü, hem durumları uygulama geliştiricinin önüne bir çok kısıtlama koyar, hem de yüksek performans beklenir. Örnek olarak, yazıcılardaki mikro işlemciler, kamera ve oyuncaklardaki uygulamaya özel entegre devreler verilebilir. Java teknolojisinin sunduğu platform bağımsızlığı çeşitli mikroişlemciler tarafından tümleşik sistemler de kullandıkları için, çok faydalı olmaktadır. Ayrıca Java'nın bellek kullanımı yöntemi, düzeltilmesi çok zor olabilecek bazı sorunları önlemektedir. Ayrıca Java programları genellikle C++ programlarından daha az yer kaplamakta ve daha az bellek kullanmaktadır. Bu da belleğin kısıtlı olduğu kontrol uygulamalarında bir avantajdır.

Java programlama teknolojisi gibi, silikonun içindeki Java da gerçek endüstriyel kontrol sistemlerine doğru bir adım daha olarak değerlendirilmelidir (Asherton,1998).

Cedra Rapids, Iowa'da bulunan Rockwell Collins Inc. ticari ve askeri hava taşıtları için uçuş gösterimi, oto pilot ve benzeri yönlendirme sistemleri sağlamaktadır. Bu tür ürünlerin elektronik alt sistemleri küçük ve hafif olmasının yanı sıra son derece güvenilir olmalıdır.

Dolayısıyla, bu sistemler mikro işlemcilere dayalıdır (Asherton, 1988). Firma, uzun zamandır nesne tabanlı teknolojileri kullanmaktadır. Yıllar boyunca askeri ve ticari havacılık projelerinde nesne tabanlı Ada dilini kullanmaktadır; böylece mühendisleri hem Ada dilinde uzmanlaşmış hem de Ada nesnelerinde oluşan büyük bir kütüphaneye sahip olmuşlardır. Java tanıtıldığında, mühendisleri, bu yeni teknolojinin kendi alanlarındaki sistem oluşturma yöntembilimindeki etkisini kavramış ve firma da ürünlerini Java temelli olarak hazırlamaya karar vermiştir. İlk yapılan şey, Ada'dan Java'ya çeviri yapabilecek bir derleyici için kontrat imzalayarak hazırdaki kütüphanelerini kullanabilir hale getirmişlerdir. Daha sonra da kendi ürettikleri mikroişlemcilerin çekirdeğini, Java teknolojisine göre değiştirerek, Java Bayt Kodlarını kendi komut takımı olarak çalıştırabilecek şekilde düzenlediler. Böylece denenmiş ve test edilmiş bir platformu yenilikler içeren bir platforma çevirmişlerdir.

Bu yeniliklerin sonucu olarak, yazılım geliştirme hızlanmış ve ürünlerini piyasaya daha çabuk sunabilir olmuştur. Ayrıca programları da artık daha çağdaş ve evrensel bir platforma hitap etmektedir (Asherton, 1988).

Rockwell Collins'in çalışmaları ayrıca JEM-1 mikro işlemcisinin ortaya çıkmasını sağlamıştır. JEM-1, 50 MHz'de çalışan 0.5 m CMOS işlem teknolojisi ile üretilmiş bir işlemcidir. JEM-1, Java Bayt Kodlarını, kendi komut takımı olarak kullanmaktadır. Bellek kullanımı da bir yığın (stack) kullanımı yerine bir yığıt (heap) kullanımı üzerine temellendirilmiştir. Bu seçimdeki sebep, bir yığının boyutunun derleme zamanında (compile time) belirlenmesi, bir yığıtın boyutunun ise, çalışma zamanında belirlenmesidir ki bu Java'nın özelliklerine daha uygundur.

Bir işlemciyi tamamlamak için, JEM-1 çekirdeği, bir kesme denetleyicisi (interrupt controller), iki programlanabilir zamanlayıcı, dışsal bir veri yolu için destek, güç yönetimi teknolojisi ve bir birleşik test işlemi grubu (Joint Test Action Group - JTAG) ara yüzü ile desteklenmiştir. Böylelikle JEM-1 gerçek zamanlı uygulamalar için gerekli performansı sağlayabilmektedir.

1.3 Bulanık Mantık

Endüstriyel bir süreç denetiminde; sistemin güvenliği ve kararlılığını sağlaması, kolay ve anlaşılır olması, tamir edilebilir ve değiştirilebilir olması, sistem performansını istenen seviyeye çıkarması, yatırım ve işletme açısından ucuz olması gibi kriterler istenmektedir. Bu koşulların gerçekleştirilmesi için kontrol edilecek sistemin yapısının ve dinamik özelliklerinin çok iyi bilinip matematiksel modellenmesi gerekir (Elmas,2003).

Gerçek dünya olaylarının çok karmaşık olması dolayısı ile bu olayların belirgin denklemlerle tanımlanarak, kesin bir şekilde kontrol altına alınması mümkün olmaz. Bunun doğal sonucu olarak, kesin olmasa bile yaklaşık fakat çözülebilirliği olan yöntemlere başvurulur (Şen, 2001). Mühendislikte bütün teori ve denklemler gerçek dünyayı yaklaşık bir şekilde ifade eder. Einstein'ın dediği gibi, gerçek olaylar matematik denklemlerle kesinlikle ifade edilebiliyor denirse, ya denklemlerin kesinliğinden ya da matematik denklemler gerçeği kesin olarak tasvir edebiliyor sonucuna varılırsa, bu seferde, gerçek dünya olaylarından söz edilemez. O halde, yapılan bütün çalışmalarda çözümler bir dereceye kadar yaklaşıktır. Aksi takdirde, çok sayıda doğrusal olmayan denklemin aynı zamanlı çözülmesi gerekir ki, bunun günümüz bilgilerine göre belirgin olmayan "kaotik" çözümlere yol açacağı bilinmektedir (Lorenz, 1963).

Günümüzde bilgi ve bunun getirdiği sözel verilere önem verilmektedir. Bunun sebebi, insanların bir cihaz gibi sayısal değil de yaklaşık sözel verilerle konuşarak anlaşmasıdır. Sözel insan verilerini, bir sistem içinde formüle ederek cihazların verdiği sayısal bilgilerle beraber mühendislik sistemlerinde gözönünde tutmak gerekmektedir.

Bazı sistemlerde modelleme doğru şekilde yapılsa bile elde edilen modelin denetleyici tasarımında kullanımı karmaşık problemlere ve oldukça yüksek maliyete neden olabilir. Bu nedenle, bazı denetim algoritmalarının belirsiz, doğru olmayan, iyi tanımlanmış, zamanla değişen ve karmaşık sistemlere uygulanması mümkün olmayabilir. Bu durumda, ya hiç çözüm üretilememekte ya da elde edilen kontrolörün performansı yeterince iyi olmamaktadır (Sinecen, 2002).

Bu gibi durumlarda genellikle bir uzman kişinin bilgi ve deneyimlerinden yararlanılma durumuna gidilir. Uzman kişi *az, çok, pek az, pek çok, biraz az, biraz çok,* gibi günlük hayatta sıkça kullanılan dilsel niteleyiciler doğrultusunda bir denetim gerçekleştirir. Bu dilsel ifadeler doğru bir şekilde bilgisayara aktarılırsa hem uzman kişiye ihtiyaç kalmamakta hem de uzman kişiler arasındaki kontrol farkı ortadan kalkmaktadır. Böylece kontrol mekanizması esnek bir

yapıya kavuşmaktadır. Bulanık mantığın temeli insanın herhangi bir sistemi denetlemedeki düşünce ve sezgilerine bağlı davranışının benzetimine dayanmaktadır. Dolayısıyla bir insan, bir sistemi bulunduğu gerçek durumdan istenilen duruma götürmek için sezgilerine ve deneyimlerine bağlı olarak bir denetim stratejisi uygulayarak amaca ulaşmaktadır (Elmas, 2003). Bulanık mantık bu tür mantık ilişkileri üzerine kurulmuştur. Bulanık mantık için, matematiğin gerçek dünyaya uygulanması denilebilir. Çünkü gerçek dünyada her an değişen durumlarda değişik sonuçlar çıkabilir.

Bulanık mantık yaklaşımı makinelere insanların özel verilerini işleyebilme ve onların deneyimlerinden ve önsezilerinden yararlanarak çalışabilme yeteneği verir. Bu yeteneği kazandırırken sayısal ifadeler yerine sembolik ifadeler kullanılır. İşte bu sembolik ifadelerin makinelere aktarılması matematiksel bir temele dayanır. 1965 yılında Zadeh yayınlanan bir makalesinde bu konudan ilk kez söz etmiştir. Zadeh bu çalışmasında insan düşüncesinin büyük çoğunluğunun bulanık olduğunu, kesin olmadığını belirtmiştir. Bu nedenle 0 ve 1 ile temsil edilen boolean mantık bu düşünce işlemini yeterli bir şekilde ifade edememektir.

Bu matematiksel temel, “Bulanık Mantık Kümeler”, “Kümeler Kuramı” ve buna dayanan “Bulanık Mantık” tır.

Son yıllarda çağdaş konular arasında ilk sırayı tutan bulanık küme, mantık ve sistemler hemen her mühendislik dalında uygulanır hale gelmektedir. Gün geçtikçe ülkemizdede bu konuya ilgi duyanların sayıları gittikçe artmaktadır. Bulanık Mantık Kontrol uygulanırken sistemin matematiksel modellenmesi şart değildir. Zadeh insanların kontrol alanında, mevcut makinelerden daha iyi olduğunu ve kesin olmayan sözel bilgilere bağlı olarak etkili kararlar alabildiklerini savunmuştur. Klasik kontrol uygulamalarında karşılaşılan zorluklar nedeniyle, Bulanık Mantık Kontrolü alternatif yöntem olarak çok hızlı gelişmiş ve modern kontrol alanında geniş uygulama alanı bulmuştur. Hatta, şimdiye kadar otomatik düzenlenemeyen süreçler bile “Bulanık Mantık Kontrol” ile düzenlenebilir duruma gelmiştir. Örnek olarak Japonya’daki Sendai metrosunun serbest hareket edip otomatik olarak durması verilebilir (Sinecen, 2002).

Son birkaç yıl boyunca Bulanık Mantık Kontrol, en yüksek aktivitelerde ve araştırmalar için karlı bir yol olarak bulanık küme teorisinde kendisini göstermeye başlamıştır. Buradaki en önemli etken, endüstriyel gelişimin hegomanyasında, giriş/çıkış ilişkilerine ait nicel

bilgilerin eksikliğidir. Bulanık Mantık Kontrol, bir mantıksal sistemin (geleneksel sistemlerden) insan düşüncesi ruhu ve dilinin sentezlenmesinden ortaya çıkar. Bulanık mantığın temeli Bulanık Mantık Kontrol'e (BMK) dayalıdır, bir sözel kontrol stratejisinin dönüşümü temel alınarak bilginin otomatik kontrol stratejisinde açıklanmasını sağlanır. Bir BMK'nın incelenmesi; genel metodolojinin öncesi için ve performans değerlendirmesi öncelikle tanıtılacak unsurlardır, ayrıca problemler çıkış noktasında yoğun olarak göründüğünden daha fazla araştırmaya ihtiyaç duyulur (Elmas, 2003).

Bulanık Mantık Kontrol üç ana yapı içerir.

- Bulanıklaştırma (Fuzzification)
- Çıkarım (Inference)
- Durulaştırma (Defuzzification)

Bulanıklaştırma ile bir giriş büyüklüğü, tam bir değeri veya keskin bir kümeyi bulanık bir büyüklük formuna getirilmektedir. Çıkarım da ise bulanık kümeler birleştirilir. Durulaştırma kısmında ise sonuç tekrar tam değere dönüştürülür.

1.4 Simulasyon

Bir sistem modellenmeden önce, üzerinde yapılması düşünülen değişikliklerin doğrudan gerçek sistem üzerinde denenip denenemeyeceğine karar verilmesi gerekir. Eğer denenebiliyorsa gerçek sistem üzerindeki uygulama, en emin ve en güvenli sonucu verecektir. Bir sistemin modellenebilmesi için o sistemin var olması da şart değildir. İmalat tesisleri veya nükleer silah sistemleri gibi. Dolayısıyla, sistemin bir göstergesi olarak modelinin oluşturulması ve gerçek sistem yerine bu modelin kullanılması gerekmektedir. Ancak bir model kullanıldığında da, modelin, verilecek kararların gerektirdiği doğrultuda, sistemi tam doğru olarak yansıtıp yansıtmadığı sorusu daima mevcuttur. Bu da modelin, uygunluk testlerinden geçirilmesi gerekli kılmaktadır. Ancak geçerleme, doğrulama ve onaylama

sürecini tamamlayan bir model üzerinde deneyler yapılmak suretiyle, sistem hakkında çıkarımlar ve yorumlar yapılabilir (Kuş, 2003).

Simulasyon aşamalarından birisi de matematiksel modelin oluşturulmasıdır. Eğer matematiksel model karmaşık, çok değişkenli değilse ve analitik çözümü de mümkünse, modelin analitik çözümü simulasyon tekniğine tercih edilmelidir.

1.5 Tez taslağı

Bu tez 6 bölümden oluşmaktadır: Birinci bölüm tez içerisinde kullanılan kavramların biribiriyle ilişkisini literatür bilgisiyle açıklamaktadır. İkinci bölüm'de bulanık mantık küme kavramları ve buna bağlı olarak günümüzde yaygın şekilde kullanılan kontrol tekniklerinden biri olan Bulanık Mantık Kontrol'ün temellerinden bahsedilmiştir. Üçüncü bölüm'de Bulanık Mantık Kontrol'ün bilgisayar ortamında gerçekleştirilmesi için gerekli yazılım kaynaklarından biri olan nesne tabanlı programlama dili Java hakkında bilgiler verilmektedir. Dördüncü bölümde bu tezin ana konusunu oluşturan kamyon yükleme-boşaltma sisteminin bulanık modellenmesi anlatılmaktadır. Beşinci bölümde ise elde edilen bulanık modelin bilgisayar ortamına taşınmasını sağlayan Java simulasyon programı işlenmiştir. Son bölümde ise tüm bölümlerde anlatılanlar ışığında bu çalışmadan çıkarılan sonuçlardan bahsedilmektedir.

İKİNCİ BÖLÜM
BULANIK MANTIK TEORİSİ ve BULANIK MANTIK KONTROL

2.1 Giriş

Mühendislikte ve diğer bilim dallarında sistemler, kesin matematiksel işlemleri kullanmak suretiyle modellenir. Günlük hayatta karşılaşılan problemlerin büyük çoğunluğu kesin olmama durumu veya bir tam tanımlanmama durumu içerir. Bilim adamları bu tür problemlerin daha etkin çözülebilmesi için insan gibi algılayabilen ve karar verebilen sistemler yapmaya yönelmiştir. Bu yöneliş sonucunda, kelime anlamı "Bulanık Mantık" (Fuzzy Logic) olan ve klasik mantığı da özel bir hal olarak içine alan iki değerli seviye yerine çok değerli seviyeyi benimseyen bir mantık geliştirilmiştir.

Bulanık mantık yaklaşımı, makinelere insanların özel verilerini işleyebilme ve onların deneyimlerinden ve önsezilerinden yararlanarak çalışabilme yeteneği verir. Bu yeteneği kazandırırken sayısal ifadeler yerine sembolik ifadeler kullanır. Bu sembolik ifadelerin makinelere aktarılması matematiksel bir temele dayanır. Bu matematiksel temel bulanık mantık kümeler kuramı ve buna dayanan Bulanık Mantık Kontrol'dür.

Bu bölümde, klasik kümelerin esaslarına değinildikten sonra bulanık kümelerle hesaplamaların nasıl yapıldığı anlatılacak, böylece kıyaslamalı olarak, daha geniş kapsamlı olan bulanık kümelerin simgesel tanımları özellikleri ve işlem yapma yöntemlerine değinilecektir. Daha sonra bulanık küme hesaplamaları ile Bulanık Mantık Kontrol arasında geçerli ilişkiyi sağlayacak bulanık küme kural yapıları ve bunlardan elde edilecek çıkarımlar hakkında temel bilgiler verilecektir.

2.2 Bulanık Mantığın Tarihi

1965 yılında Azerbeycan'lı bilim adamı Prof. Dr. Lütfü ZADEH tarafından ortaya atılan Bulanık küme, mantık ve sistem kavramları bu araştırıcının uzun yıllar boyunca kontrol alanında çalışması amacıyla istediği kontrolü elde edebilmesi için fazlaca doğrusal olmayan denklemlerin işin içine girmesi; yöntemin karmaşıklaşması ve çözümün zorlaşması sonucunda ortaya çıkmıştır (Zadeh,1965,1968,1971). İlk ortaya atıldığı zamanlarda, bulanık sistemlerin doğrudan uygulaması olmadığından, yapılan araştırmalar daha çok felsefi seviyede kalmış ve bunun sonucunda daha kuvvetli felsefi ve teorik temelleri olan olasılık teorisi ve istatistik yöntemleri ağır basmıştır (ŞEN 2001). Bulanık kavram ve sistemlerin dünyanın değişik araştırma merkezlerinde dikkat çekmesi 1975 yılında Mamdani ve Assilian tarafından yapılan bir kontrol uygulaması ile olmuştur. Bu araştırmacılar ilk defa bir buhar makinesinin bulanık mantık ile kontrolünü başarmıştır. Bu ön çalışmada bulanık sistemlerle çalışmanın ne kadar kolay ama sonuçlarının ne kadar etkili olduğu anlaşılmıştır. (Mamdani, 1975, 1976, 1977).

Bulanık Mantık Kontrolün son araştırmaları, su kalite kontrolü (Yagishita, 1985), otomatik tren işletim sistemleri (Yasunobu ve Miyamoto,1983,1985, 1987), otomatik vinç taşıma sistemleri (Yasunobu ve Miyamoto, 1986, 1987), asansör kontrol (Fujitech, 1988), nükleer reaktör kontrolü (Bernard, 1988), otomobil vites kontrolü (Kasai ve Morimoto, 1988), donanım sistemlerinin Bulanık Mantık Kontrolü (Yamawaka, 1986, 1987), bulanık bellek aygıtları (Togai ve Watanabe, 1986) ve bulanık bilgisayarlar (Yamawaka, 1987) teşhisi zor hastalıkların tanımlanmasında Bulanık Mantık Kontrol'ün yararlı etkilerine, bir insan operatörü tarafından dinamik bilgileri olmaksızın kontrol edilebildiğine işaret etmiştir.

Tarihsel olarak Bulanık Mantık Kontrol'ün gelişmeleri Çizelge2.1 de özetlenmiştir.

Çizelge 2.1: Tarihsel Olarak Bulanık Mantık Kontrol'ün Gelişmeleri

1972	Zadeh	Bir Bulanık Mantık Kontrol için sabitler
1973	Zadeh	Bilimsel yaklaşım
1974	Mamdani & Assilian	Makine takım kontrolü

1976	Rutherford *et al.*	Kontrol algoritmalarının analizi
1977	Ostergaard	Isı değişimi ve çimento ocaklarının kontrolü
1977	Willaeys *ve diğerleri.*	Optimum bulanık kontrol
1979	Komolov *ve diğerleri.*	Sınırlı otomasyon
1980	Tong *ve diğerleri.*	Atık su muamelesi projesi
1980	Fukami,Mizumoto ve Tanaka	Bulanık şartların sonuçları
1983	Hirota ve Pedryez	Bulanık ihtimal setleri
1983	Takagi ve Sugeno	Bulanık Mantık Kontrol kurallarının kökeni
1983	Yasunobu, Miyamoto *ve diğ.*	Bulanık Mantık Kontrol tahminleri
1984	Sugeno ve Murakami	Bir model arabanın park kontrolü
1985	Kiszka, Gupta *ve diğerleri.*	Bulanık sistem kararlılığı
1985	Togai ve Watanabe	Bulanık çip
1986	Yamakawa	Bulanık Mantık Kontrol donanım sistemi
1989	Omron	RISC bilgisayara dayalı çalışma
1990	Sharp	Uygulama ürünleri(Çamaşır mak.)
1993	Sony	Palm Top
1993	IEEE	"Transactions on Fuzzy Systems" adlı dergi

Klasik Aristo mantığı ele alındığında bir şeyin aynı anda hem var hemde yok olmayacağı, bir nesnenin bir kümeye aynı anda hem ait hemde o kümenin dışında olamayacağı kısaca bu mantığın orta terimin yokluğu düşüncesi üzerine kurulduğu görülür. Bu mantıkta kümeler arasında keskin sınırlar vardır. Bu yüzden doğal dile pek uygun değildir. Örneğin klasik kümelerde uzun insan denildiğinde bir insanın tam olarak ne kadar uzun olduğunu veya nekadar uzun olmadığını tam olarak belirleyemeyiz. Çünkü uzun kelimesi sübjektiftir ve keskin sınırlarla ifade edilemez. Kontrol ve otomasyon teknolojisinin yanısıra günlük hayatta bu türden bulanık kelimelere sıkça rastlanır. Bulanık kavramların olduğu durumlarda bu kavramların ifade edilmesi ve değerlendirilmesinde klasik mantık ve teoriler yetersiz kalmaktadır. Bulanık mantık belirsizlik üzerine kurulmuştur (Zadeh,1965). Bulanık mantık olayların gerçekleşme ihtimalinden ziyade gerçekleşme dereceleriyle ilgilenir. Artık bir elemanın bir kümeye üye olmaması değil ne derecede üye olduğu sözkonusudur. Bulanık kelimler arasında keskin sınırlar konulamaz, fakat bunları bulanık kümelerde göstermek mümkündür.

Bulanık mantık, insan gibi düşünen, karar verebilen, duruma göre seçim yapabilen mantıksal bir sistemdir. Bulanık Mantık Kontrolü de bulanık mantık teorisi üzerine kurulur. Analizi geleneksel nicel teknikler yardımıyla çok karmaşık olan ya da sistem bilgilerinin nitel, doğru olmayan, şüpheli olduğu durumlarda Bulanık Mantık Kontrol çok faydalıdır.

2.3 Bulanık Mantığın Temel Kavramları

Bulanık mantık dört ana teoriye dayanır. Bunlar;

(1) *Bulanık kümeler*: kesin olmayan sınırlarla kurulmuş kümeler,
(2) *Sözel değişkenler*: nitel ve nicel olarak bulanık kümede tanımlanmış değerlerler,
(3) *Olasılık dağılımları*: bir bulanık kümede belirlenerek ifade edilmiş olan sözel değişkenin değer sınırları,
(4) *Bulanık EĞER-İSE kuralları*: iki değerli mantık gösterilimi olarak genelleme yapan fonksiyonel haritalama veya mantıksal formül tanımlaması için bilgi gösterim tasarımıdır.

İlk üç kavram, bulanık mantığın tüm alt alanları için temel kavramlardır. Dördüncü kavram ise birçok Bulanık Mantık Kontrol sistemini içeren ve bulanık mantığın günümüze kadar geliştirmiş olduğu birçok endüstriyel uygulamalar için temel ifade eder.

2.3.1 Klasik Küme Teorisi

Bulanık küme teorisine geçmeden önce klasik kümelerden bahsetmek yararlı olacaktır.Klasik bir küme, verilen alandaki nesneler topluluğudur. Bir nesne ya bir kümeye aittir yada değildir. Bu yüzden, kümedeki üyeler arasında ve kümede olmayanlar arasında keskin bir sınır vardır.

Bir küme ya elemanlarını birer birer sayma yoluyla; yada, elemanlarının ortak özelliklerini tarif etme yoluyla tanımlanabilir. Birincisine uzatılmış tanım; ikincisine kuvvetlendirilmiş tanım denir. Bir kuvvetlendirilmiş tanım, genellikle temel üç büyük nedenden dolayı uzatılmış tanım yerine tercih edilir. Birinci neden, bir kümenin kuvvetlendirilmiş tanımı uzatılmış tanımından daha özlü olur. Aslında bir kümenin tüm öğelerinin tek tek sayılması bazen

imkansızdır. (Örneğin Denizlide bulunan arabaların sayısı) ikinci neden, bir kuvvetlendirilmiş tanım bir kümenin anlamını çok açık şekilde ifade eder. Üçüncüsü bir kuvvetlendirilmiş tanım öğenin özellikleri değişince yada özelliklerin tanımı değişince, bir kümenin yeni elemanlarını seçmek için kullanılabilir.

U gibi bir evrensel küme(söylem evreni) içinde sonlu elemana sahip bir A kümesini ele alalım. *u* gibi bir eleman ile bu A kümesi arasındaki ilişki, *u*'nun A'ya ait olup olmaması ile ilgilidir. *u*, A kümesi içindeyse bu durum, $u \in A$ ile, içinde değilse, $u \notin A$ ile ifade edilir. A gibi bir kümenin tüm elemanları aynı zamanda B gibi bir kümenin tüm elemanlarıysa A, B'nin alt kümesidir ve bu, $A \subset B$ ile gösterilir. Her küme kendisinin ve evrensel kümenin bir alt kümesidir, Eğer A ve B gibi iki küme birbirinin alt kümeleri iseler bu iki küme eşittir ve A=B ile gösterilir. Hiç bir elemana sahip olmayan küme boş kümedir ve Ø ile gösterilir. Sadece A kümesine ait olmayan tüm elemanlara sahip kümeye, A'nın tümleyeni denir ve $\bar{A}$ ile gösterilir.

Klasik bir A kümesi, karakteristik bir fonksiyonla aşağıdaki gibi gösterilir;

$$\varphi(u) = \begin{cases} 1, \text{ sadece ve sadece } u \in A \text{ için} \\ 0, \text{ sadece ve sadece } u \notin A \text{ için} \end{cases}$$

Klasik küme operatörleri:

Birleşim: $A \cup B = \{x \mid x \in A \text{ veya } x \in B\}$ [2.1]

Kesişim: $A \cap B = \{x \mid x \in A \text{ ve } x \in B\}$ [2.2]

Klasik küme operatörlerinin özellikleri:

1. $\overline{(\bar{A})} = A$ [2.3]
2. $A \cup B = B \cup A$ [2.4]
   $A \cap B = B \cap A$ [2.5]
3. $(A \cup B) \cup C = A \cup (B \cup C)$ [2.6]
   $(A \cap B) \cap C = A \cap (B \cap C)$ [2.7]

4. $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ [2.8]

   $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ [2.9]

5. $A \cup A = A$ [2.10]

   $A \cap A = A$ [2.11]

6. $A \cup (A \cap B) = A$ [2.12]

   $A \cap (A \cup B) = A$ [2.13]

7. $A \cup (\overline{A} \cap B) = A \cup B$ [2.14]

   $A \cap (\overline{A} \cup B) = A \cap B$ [2.15]

8. $A \cup U = U$ [2.16]

   $A \cap \emptyset = \emptyset$ [2.17]

9. $A \cup \emptyset = A$ [2.18]

   $A \cap U = A$ [2.19]

10. $A \cap \overline{A} = \emptyset$ [2.20]

    $A \cup \overline{A} = U$ [2.21]

11. $\overline{(A \cap B)} = \overline{A} \cup \overline{B}$ [2.22]

    $\overline{(A \cup B)} = \overline{A} \cap \overline{B}$ [2.23]

Küme işlemcileri ile lojik işlemciler arasındaki ilişki şu şekildedir: Eğer A, B ve C bir durumun doğruluk değerlerini gösteriyorsa, $\cap, \cup$ ve $\bar{}$ sırasıyla AND(VE), OR(VEYA) ve NOT(DEĞİL)'i gösterir. $\emptyset$ FALSE(YANLIŞ)'ı ve U TRUE(DOĞRU)'yu gösterir, tüm bu özellikler, mantığın bir kolunun özellikleri olur ki buna Bolean Mantık denir.

2.3.2 Bulanık Küme Teorisi

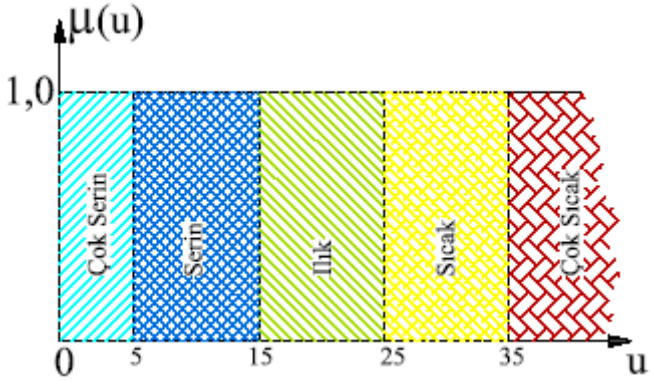
Bulanık küme kesin olmayan sınırlarla kurulmuş bir kümedir. Klasik küme teorisine göre bir küme daima kesin sınırlara sahiptir. Bulanık küme teorisi klasik küme teorilerinin kısmi üyelikleri içermesine izin verilmesini genelleştirir. Bulanık Mantık, bulanık küme teorisine dayanır.

Bazı küme ifadelerinin kesin sınırlara sahip olması gerekirken (örnek olarak, evli insanlar kümesi), bazı kümeler kesin sınırlara sahip olmak zorunda değildir (örneğin; mutlu evli çiftler kümesi, başarılı mezunlar kümesi...vb). Bulanık küme teorisinde, üyelerin küme içerisinde

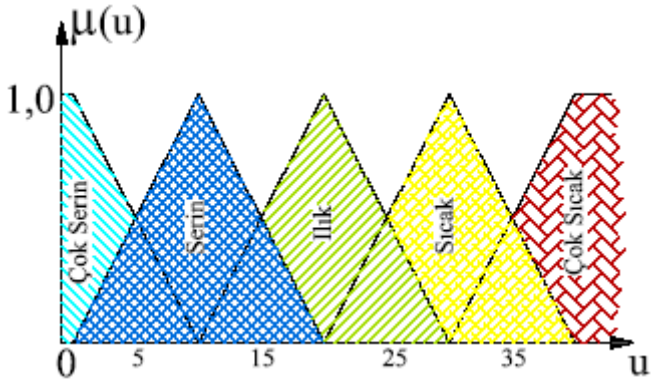
derecelendirilmeleri sağlanarak sınırlamaları doğrudan adreslenir. Küme içerisindeki üyelik derecesi 0-1 arasında bir sayı ile ifade edilir; "0" küme içinde yer almayışı, "1" tamamıyla kümenin içinde oluşu ve bu aradaki sayılar da kısmi olarak küme içerisinde oluşu ifade eder. Bu yolla, keskin olmayan ve küme dışındaki bölgelerden küme içindeki bölgelere geçişler aşamalı olarak tanımlanmış olur. Yani, bir kümenin tam üyeliği ile o kümenin üyesi olmama durumları arasında kademe kademe geçişe izin verilir. Verilen bir elemanın bir kümede kısmi üyeliğinin başlaması demek, aynı zamanda bu elemanın bu kümenin üyesi olmama durumunun da kısmen başlaması demektir. Çünkü, bulanık küme teorisi hem tam üyeliğe hemde hiç üye olmamaya izin verir. Bundan dolayı, bulanık küme teorisi klasik küme teorisinin genelleştirilmiş bir halidir denilebilir. Bulanık bir küme bir fonksiyon olarak da tanımlanabilir ve bu durumda nesneler tasarım içerisindeki yerlerinin küme içerisindeki üyelik değerlerine göre listelendirilirler. Böyle bir fonksiyon ***üyelik fonksiyonu*** olarak adlandırılır ve μ sembolü ile gösterilir (Bellman, 1973).

2.3.2.1 Bulanık Kümelerin Gösterimi

Bir bulanık kümede yatay eksendeki gerçek sayıların her biri, düşey eksende 0 ile 1 arasında değişen üyelik derecelerine dönüştürülür. Böylece, yatay eksendeki bir gerçek sayı u ile gösterilirse bunun üyelik derecesi bundan sonra $\mu_A(u)$ ile gösterilecektir. Bu söylenenlerden açıkça $0 \leq \mu_A(u) \leq 1$ olduğu anlaşılır. Klasik ve bulanık kümelerin gösterilişi Şekil 2.1'de görülmektedir.



a) Klasik Küme



b) Bulanık Küme

Şekil 2.1: Klasik ve Bulanık Küme Gösterimi

Genelde, bir klasik U kümesinin elemanları

$$U=\{u_1, u_2, u_3,.....\} \quad [2.24]$$

şeklinde gösterilirken, bunun bulanık hali

$$U=\{ \mu_A(u_1) /u_1 + \mu_A(u_2)/u_2 +..... \} = \{\Sigma \mu_A(u_i)/u_i\} \quad [2.25]$$

şeklinde gösterilir. Bulanık kümenin sürekli olması durumunda ise

$$U=\{ \int \mu_A(u)/u \} \quad [2.26]$$

olur. Her iki notasyonda da bölüm işareti asla bölmeyi göstermez, sadece alttaki gerçek sayıya yani küme öğelerine üstteki üyelik derecesinin karşı geldiğini belirtir. Yukarıdaki denklemlerin ilkinde toplam işareti de alışa geldiğimiz toplamı değil, artı işareti ile küme öğelerinin topluluğunu ifade etmek içindir. İkinci notasyonda entegral işareti de asla bildiğimiz integral anlamına gelmez yine topluluğu gösteren bir işaret olarak algılanmalıdır. Sıcaklık kelimesinin Afyon için bulanık küme olarak gösterilmesi ‘sıcaklık’={0.1/18+0.3/20+0.5/24+0.7/26+0.9/28+1.0/30} şeklinde olabilir.

2.3.2.2 Üyelik Fonksiyonu ve Üyelik Derecesi

U bir sözel evrende tanımlı, [0,1] aralığında değerler alan, sürekli yada ayrık olabilen bu evrene ait bir *u* elemanının *A* gibi bir alt kümeye ne derecede üye olduğunu veren fonksiyondur. μ_A ile gösterilir. *u* elemanının *A* alt kümesine ne kadar ait olduğunu üyelik

derecesi verir ve $\mu_A(u)$ ile gösterilir. Üyelik fonksiyonu sözel evrenden [0,1] aralığına bir dönüşüm fonksiyonudur.

$$\mu_A(u):U\rightarrow[0,1] \qquad [2.27]$$

Bulanık kümelerin gerek üyelik derecelerinin gerekse bunların tümünü temsil edebilecek üyelik fonksiyonlarının belirlenmesinde kullanılan yöntemlerin başlıcaları şunlardır;

a) Sezgi

b) Yapay sinir ağları

c) Çıkarım

d) Genetik algoritmalar

e) Mertebelendirme f) Açılı bulanık kümeler

g) Çıkarımcı muhakeme gibi değişik yaklaşımlardır.

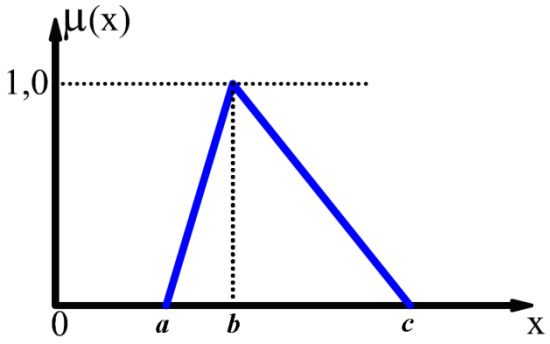
2.3.2.3 Üyelik Fonksiyonlarının Çeşitleri

Üyelik fonksiyonları, bulanık problemlerin daha kolay çözülebilmesi ve kolay hesaplanabilmesi için kullanılan matematiksel modellerdir. Bunların başlıcaları *üçgen, yamuk, gaussian, çan eğrisi, sigmoidal* üyelik fonksiyonlarıdır.

Üçgen Üyelik Fonksiyonu:Üç parametre ile açıkça belirtilir; (*a,b,c*)

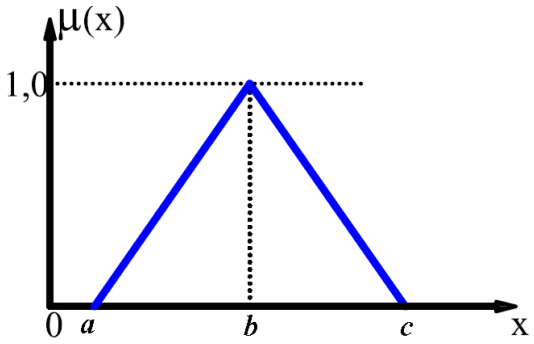
$$üçgen(x:a,b,c)=\begin{cases}0 & x<a\\ (x-a)/(b-a) & a\le x\le b\\ (c-x)/(c-b) & b\le x\le c\\ 0 & x>c\end{cases} \qquad [2.28]$$

Fonksiyonun tam görünümü *a,b,c* parametrelerinin seçimi ile tanımlanır (Şekil 2.2).



Şekil 2.2: Üçgen Üyelik Fonksiyonu

Simetrik üçgen için iki parametre (c=b-a) yeterlidir (Şekil 2.3).

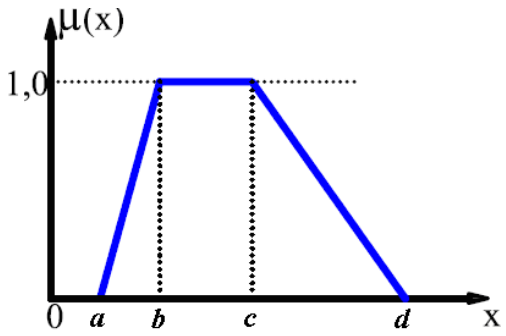


Şekil 2.3:Simetrik Üçgen Üyelik Fonksiyonu

Yamuk Üyelik Fonksiyonu: Dört parametre ile belirtilir;(a,b,c,d)

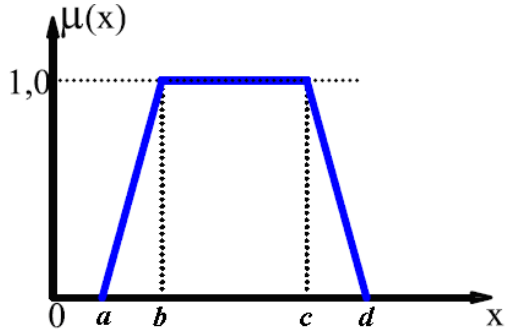
$$Yamuk(x:a,b,c,d)=\begin{cases}0 & x<a\\ (x-a)/(b-a) & a\le x \le b\\ 1 & b\le x<c\\ (d-x)/(d-c) & c\le x<d\\ 0 & x\ge d\end{cases} \qquad [2.29]$$

Fonksiyonun tam görünümü a,b,c,d parametrelerinin seçimi ile tanımlanır(Şekil 2.4).



Şekil 2.4:Yamuk Üyelik Fonksiyonu

Üçgen üyelik fonksiyonu yamuk üyelik fonksiyonunun özel bir durumudur ($b=c$). Simetrik yamuk için üç parametre ($d=b-a$) yeterlidir(Şekil 2.5).

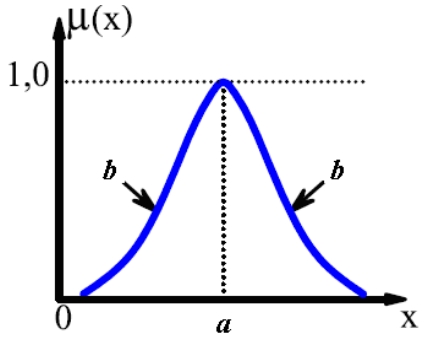


Şekil 2.5: Simetrik Yamuk Üyelik Fonksiyonu

Gaussian Üyelik Fonksiyonu: İki parametre ile tanımlanır;(a,b,c)

$$Gaussian\ (x:a,b) = \exp(-(((x-a)/b)^2)); \qquad [2.30]$$

Fonksiyonun tam görünümü şekil 2.6'daki gibidir.



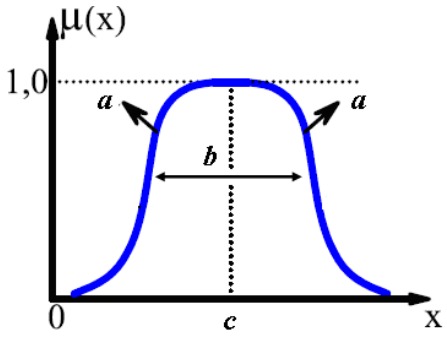
Şekil 2.6:Gaussian Üyelik Fonksiyonu

Burada *a*'yı kullanarak fonksiyonun genişliğini ayarlayabiliriz. *a*'yi eğrinin kırılma noktalarında kullanırız.

Çan Eğrisi Üyelik Fonksiyonu: Üç parametreyle tanımlanır;(*a,b,c*)

$$Çan\ eğrisi\ (x:a,b,c) = 1/1+|(x-c)/a|^{2b} \qquad [2.31]$$

Fonksiyonun tam görünümü Şekil 2.7'deki gibidir.



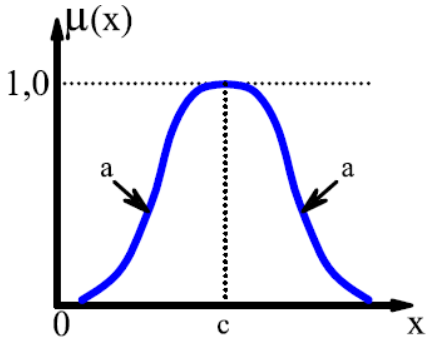
Şekil 2.7: Çan Eğrisi Üyelik Fonksiyonu

Şekil 2.7'den de görüleceği gibi *b* daima pozitiftir. Bu fonksiyon aynı zamanda olasılık teorisinde kullanılan Cauchy dizisinin doğrudan genelleştirilmiş bir biçimidir. Yine burada da *c* ve *a*'yı düzelterek fonksiyon genişliğini değiştirebiliriz. Yukarıda olduğu gibi *b*'yi kırılma noktalarının kontrolünde kullanırız.

Sigmoidal Üyelik Fonksiyonu: İki parametre ile tanımlanır;(*a,c*).

$$Sigmoidal\ (x:a,c) = 1/1+e^{-a(x-c)} \qquad [2.32]$$

Fonksiyonun tam görünümü Şekil 2.8'deki gibidir.



Şekil 2.8: Sigmoidal Üyelik Fonksiyonu

Parametreler artarken 0'dan 1'e geçiş keskin olur. Fonksiyon bu şekilde pozitif sonsuza yaklaşırken, adım fonksiyonuna yaklaşır.

Pratik uygulamalarda en fazla üçgen yamuk ve çan kullanılır. Gaussian üyelik fonksiyonları, neuro fuzzy sistemlerinde tercih edilir.

2.3.2.4 Bulanık Kümelerde Temel Küme Operasyonları

A ve B, U evreninde üyelik fonksiyonları sırasıyla μ_A ve μ_B olan iki bulanık küme olsun. Birleşim, kesişim ve tümleyen gibi küme operatörleri, bulanık kümeler için üyelik fonksiyonları ile ifade edilirler (Klir ve Folger,1988).

Küme teorisi için gerekli olan özellikler:

De Morgan Kanunu:

$$\overline{(A \cap B)} = \overline{A} \cup \overline{B} \text{ ve } \overline{(A \cup B)} = \overline{A} \cap \overline{B} \qquad [2.33]$$

Dağılma özelliği:

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \text{ ve } A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \qquad [2.34]$$

Soğurma ve gereksiz tekrar kuralı:

$$A \cup (A \cap B) = A, \quad A \cap (A \cup B) = A \qquad [2.35]$$

ve

$$A \cup A = A, \quad A \cap A = A \qquad [2.36]$$

Bulanık kümeler bu özellikleri sağlamasına karşılık, orta terimin yokluğu ve zıtlık kanunu sağlamaz (Pedrycz,1993), yani:

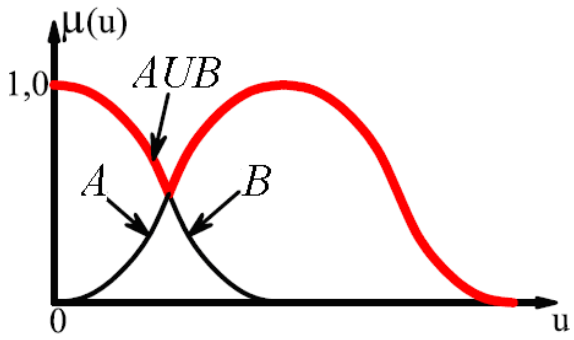
$$A \cap \overline{A} \neq \emptyset \text{ ve } A \cup \overline{A} \neq U \qquad [2.37]$$

Bundan dolayı bulanık küme ve onun tümleyeni arasında bir örtüşme meydana gelir.

Birleşim:

A∪B birleşiminin üyelik fonksiyonu olan μ_{AUB} tüm *u*'lar için;

$\mu_{AUB}(u) = max\{ \mu_A(u), \mu_B(u) \}$ ile tanımlanır ve Şekil 2.9'daki gibidir (Kosko,1992).

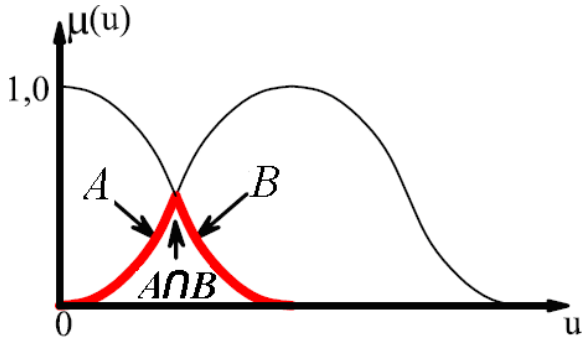


Şekil 2.9: Üyelik Fonksiyonlarının Birleşimi

Kesişim:

A∩B kesişiminin üyelik fonksiyonu olan $\mu_{A \cap B}$ tüm *u*'lar için;

$\mu_{A \cap B}(u) = \min\{ \mu_A(u), \mu_B(u) \}$ tanımlanır ve Şekil 2.10'daki gibidir (Fuller,1995).

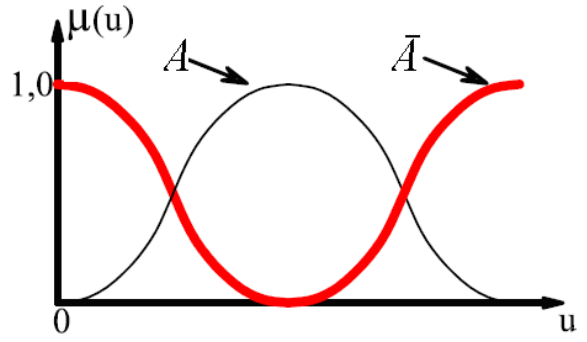


Şekil 2.10:Üyelik Fonksiyonlarının Kesişimi

Tümleyen:

Bir A bulanık kümesinin tümleyeninin üyelik fonksiyonu olan $\mu\bar{A}(u)$, tüm u'lar için;

$\mu\,\bar{A}(u) = 1 - \mu_A(u)$ ile tanımlanır ve Şekil 2.11'deki gibidir (Kosko,1992).



Şekil 2.11:Üyelik Fonksiyonlarının Tümleyeni

t-normu:

t norm operatör **t**(x,y) ile gösterilir.[0,1]*[0,1]→[0,1]'e haritalanan bir fonksiyondur. Bu koşullar bazı $w,x,y,z \in [0,1]$ ile sağlanır.

(i) her bir argüman için azalmayan fonksiyondur;

$x \le y, w \le z$ için $x\mathbf{t}w \le y\mathbf{t}z$ [2.38]

(ii) Değişme özelliği vardır;

$x\mathbf{t}y \le y\mathbf{t}x$ [2.39]

(iii) Birleşme özelliği vardır;

$(x\mathbf{t}y)\mathbf{t}z = x\mathbf{t}(y\mathbf{t}x)$ [2.40]

(iv) Sınır koşullarını sağlar;

$x\mathbf{t}0=0,\ x\mathbf{t}1=x$ [2.41]

$\mathbf{t}$ -normuna örnek olarak;

kesişim: $z \wedge y = \min\{x, y\}$, [2.42]

cebrik çarpım: $z \cdot y = x.y$, [2.43]

sınırlı çarpım: $x\Theta y = \max\{0, x+y-1\}$, [2.44]

ani çarpım: $x \bigcap y = \begin{cases} x & y=1 \\ y & x=1 \\ 0 & x,y<1 \end{cases}$ [2.45]

ifadeleri verilebilir.

$\mathbf{s}$ - normu:

$\mathbf{s}$ -norm operatör $\mathbf{s}(x,y)$ ile gösterilir.[0,1]*[0,1]→[0,1]'in haritalanmış bir fonksiyondur. Bu koşullar bazı $w,x,y,z \in [0,1]$ ile sağlanır.

(i) her bir argüman için azalmayan fonksiyondur, yani;

$x \le y, w \le z$ için $x\mathbf{s}w \le y\mathbf{s}z$ [2.46]

(ii) Değişme özelliği vardır;

$x\mathbf{s}y \le y\mathbf{s}x$ [2.47]

(iii) Birleşme özelliği vardır;

$(x\mathbf{s}y)\mathbf{s}z=x\mathbf{s}(y\mathbf{s}x)$ [2.48]

(iv) Sınır koşullarını sağlar;

$x\mathbf{s}0=x,\ x\mathbf{s}1=1$ [2.49]

s -normuna örnek olarak;

birleşim: $x \vee y = \max\{x, y\}$, [2.50]

cebrik toplam: $x \overline{+} y = x + y - x.y$, [2.51]

sınırlı toplam: $x \oplus y = \min\{1, x + y\}$, [2.52]

ani toplam: $x \bigcup y = \begin{cases} x & y=0 \\ y & x=0 \\ 1 & x, y>0 \end{cases}$ [2.53]

ayrık toplam : $x \Delta y = \max\{\min(x,\ 1 - y),\ \min(1 - x,\ y)\}$ [2.54]

ifadeleri verilebilir (Lee,1990).

Kartezyen Çarpım:

Eğer A_1, A_2,.....,A_n sırasıyla U_1,U_2,.....,U_n evrenlerinde bulanık kümeler ise A_1, A_2,.....,A_n kümelerinin kartezyen çarpımı U_1xU_2.....xU_n uzayında bir bulanık kümedir. Üyelik fonksiyonu;

μ_{A1} x .....x $\mu_{An}(u_1,u_2,.....,u_n)$ = min{ $\mu_A(u_1)$,....., $\mu_A(u_n)$ } veya [2.55]

μ_{A1} x .....x $\mu_{An}(u_1,u_2,.....,u_n)$ = $\mu_{A1}(u_1)$x $\mu_{A2}(u_2)$..... x $\mu_{An}(u_n)$ [2.56]

ile tanımlanır.

Bulanık İlişki:

Bir U bulanık kümesinin bir V bulanık kümesine olan bir R bulanık ilişkisi UxV kartezyen çarpımının bir alt kümesidir. R, $\mu_R(u,v)$ iki değişkenli üyelik fonksiyonu ile karakterize edilir. Bulanık ilişki;

R= $\int \mu_R(u,v)/(u,v)$ [2.57]

ile ifade edilir. Daha genel olarak, n-tabanında bir bulanık ilişki için $U_1xU_2.....xU_n$ uzayında bir bulanık kümedir bu durum;

$$Ru_1xu_2x...xu_n = \{((u_1,u_2,...,u_n),\ \mu_R(u_1,u_2,...,u_n)) \mid (u_1,u_2,...,u_n)) \in U_1xU_2...xU_n\} \quad [2.58]$$

ile gösterilir (Zadeh,1973).

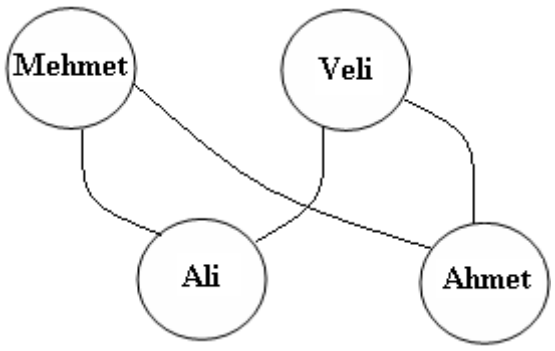
Bulanık ilişkiye örnek olarak;
U={Ali, Ahmet} ve V={Veli, Mehmet} gibi iki küme alalım. Bu iki kümenin elemanlarının benzerliği;

Benzerlik= 0.8/(Ali, Veli) + 0.4/(Ali, Mehmet) + 0.5/(Ahmet, Veli) + 0.7/(Ahmet, Mehmet) şeklinde gösterilebilir.

Bu ilişki matris biçiminde gösterilmek istenir ise;

	Veli	Mehmet
Ali	0.8	0.4
Ahmet	0.5	0.7

yazılabilir. Bu ilişkiler ilişkisel bir tablo (graf) üzerinde Şekil 2.12 ile gösterilmiştir.



Şekil 2.12 Bulanık ilişki grafı

Bulanık ilişki, matris biçiminde verilebileceği gibi bir fonksiyon şeklinde de verilebilir. Örnek olarak; verilen bir U ve V reel sayı kümelerinin elemanları için "u, v'den daha büyüktür" ilişkisi;

$$\mu_R(u,v) = \begin{cases} 0 & u \geq v \\ 1/\left(1+(10/(v-u)^2)\right) & u < v \end{cases} \qquad [2.59]$$

biçiminde bir fonksiyonla verilebilir.

Bulanık ilişkilerin kompozisyonları:

Sup-Star Kompozisyonu

Eğer R ve S sırasıyla UxV ve VxW'de bulanık ilişkilerse, R ve S'nin RoS ile gösterilen Sup-Star kompozisyonu;

RoS=Sup{ $\mu_R(u,v) * \mu_R(v,w)$ } veya [2.60]

RoS={$((u,w), \text{Sup } \mu_R(u,v) * \mu_R(v,w))), u \in U, v \in V, w \in W$} ile tanımlanır. [2.61]

Burada * operatörü herhangi bir t-normu olabilir. En yaygın kullanılanları;

Sup{ $\mu_R(u,v)\ \mu_R(u,v)$} ile gösterilen Sup-Min ve [2.62]

Sup{ $\mu_R(u,v).\ \mu_R(u,v)$} ile gösterilen Sup-Çarpım kompozisyonlarıdır. [2.63]

Max-Star Kompozisyonu

Eğer R ve S sırasıylaUxV ve VxW'de bulanık ilişkilerse, R ve S'nin R*S ile gösterilen Max-Star Kompozisyonu;

R*S ={ $\mu_R(u,v) * \mu_R(u,v)$} veya [2.64]

R*S ={$((u,v), \text{Max}\{ \mu_R(u,v) * \mu_R(u,v)\}, u \in U, v \in V, w \in W$} ile tanımlanır. [2.65]

Burada * operatörü herhangi bir s-normu olabilir. En yaygın kullanım alanları;

Max{ $\mu_R(u,v)$ $\mu_R(u,v)$} ile gösterilen Max-Min ve [2.66]

Max{ $\mu_R(u,v)$. $\mu_R(u,v)$} ile gösterilen Max-Product kompozisyonlarıdır. [2.67]

(Terano ve diğ.,1991)

2.3.2.5 Bulanık Kümelerin Özellikleri

Bulanık eleman sayısı (cardinality):

Bir kümenin kardinelitesi kümedeki elemanların toplam sayısıdır. Bir eleman bir bulanık kümeye kısmen ait olduğu için klasik kardinal fikrinin doğal genellemesi herbir elemanın onun üyelik derecesine ağırlığıdır. Bir bulanık kümenin kardinelitesini hesaplamak için aşağıdaki formül kullanılır.

$$Card(A) = \sum \mu A(x_i) \quad [2.68]$$

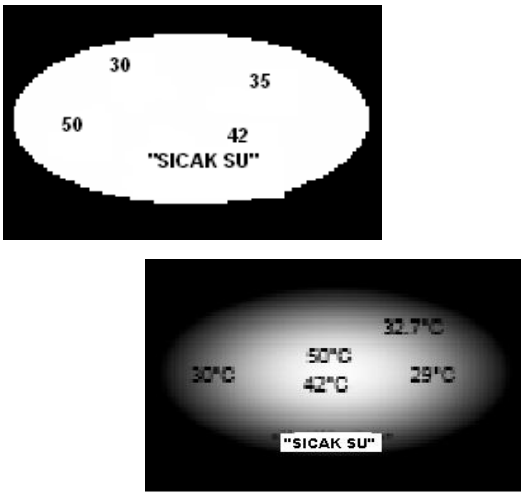
Bir bulanık kümenin kardinelitesi aynı zamanda normalizasyon faktörü gibi diğer özelliklerin tanımında da kullanılır.

Yükseklik:

Bir bulanık kümenin yüksekliği onun üyelik fonksiyonunun üyelik değerinin en yükseğidir.

Yükseklik(A)=max $\mu_A(x_i)$ [2.69]

Bir bulanık kümenin yüksekliği 1 ise normal bulanık küme olarak isimlendirilir, eğer bulanık kümenin yüksekliği 1'den daha küçük ise subnormal bulanık küme denir. Klasik küme teorisinde bir küme ya boştur yada boş değildir. Bulanık kümelerin subnormal kavramı siyah ve beyaz gibi iki sınır arasında bir gri alan ortaya çıkarır. Subnormal bulanık küme sadece kısmi üyeleri içermez, aynı zamanda tam üyesi olmayan bir bulanık kümedir. Bundan dolayı bazı durumlarda boş ve boş olmayan iki küme arasındadır (Şekil 2.13).



(a) Klasik mantık (b) Bulanık mantık

Şekil 2.13 Klasik ve Bulanık Mantığın Karşılaştırılması

Bu kavramların çoğu insan mantığını normal bulanık mantığa uydurmak için kullanılır. Subnormal bulanık kümeler genellikle bulanık kural tabanı oluşturma sırasında oluşturulur.

Destekleme ve alfa seviye kesimleri:

U evreninde A gibi bir bulanık küme olsun. Bir bulanık kümenin desteği A'daki üyelik derecesi 0'dan büyük elemanların kümesidir. Bu durum;

$$\text{Spt}(A)=\{x\ U|\mu A\ (x) >0\} \qquad [2.70]$$

şeklinde ifade edilir.

α-kesimi kavramı (yada alfa seviyesi) destekten daha genel bir ifadedir. α_0 0 ile 1 arasında bir numara olsun. α_0'da A bulanık kümesinin α-kesimi $A\alpha_0$ ile gösterilsin, kümenin elemanlarının üyelik derecesi α_0'dan daha küçük değildir. Matematiksel olarak, U evreninde A bulanık kümesinin α-kesimi;

$$A\alpha_0 = \{x \in U |\ \mu_A(x) \geq \alpha_0\} \qquad [2.71]$$

ile tanımlanır.

Çözüm kimliği:

α-kesimleri fikri üzerinde, bir bulanık küme α değerleri kullanarak birçok dilimli kümeye ayrılır. Herbir α-seviyesi, orijinal üyelik fonksiyonunu katmanlanarak (piling up) tekrar kurabilir. Bir üyelik fonksiyonunun α-kesimlerinden yeniden yapılandırılmasının temeli bulanık küme teorisindeki çözüm kimliğidir. A ayrık bir bulanık küme alındığında destek kümesindeki elemanların sıfır olmayan üyelik değerliklerini artan bir sıra şeklinde sıralanabilir: (α_0 , α_1 , α_2 ,..... α_n) bulanık küme teorisinde çözüm kimliği prensibi;

$$A= \alpha_0 * A\alpha_0 + \alpha_1 * A\alpha_1 + ... \alpha_n * A\alpha_n \qquad [2.72]$$

olarak ifade edilir. $\alpha_i * A\alpha_i$

$$\mu\alpha_i * A\alpha_i = \begin{cases} \alpha_i & \text{eğer } \mu A(x) \geq \alpha_i \\ 0 & \text{diğer durumlarda} \end{cases} \qquad [2.73]$$

bir bulanık kümeyi ifade eder. Burada, (+) disjunction işlemcisini gösterir.

Sürekli bulanık kümenin çözüm kimliği;

$$A=U\alpha * A\alpha \qquad [2.74]$$

ile ifade edilir. α 0'dan 1'e değişen sürekli değişken, U "sup" işlemini gösterir.

Dışbükey bulanık kümeler:

Eğer üyelik fonksiyonu bir vadi şeklinde değilse bu bulanık küme dışbükeydir.
U evreninde A adında bulanık bir alt küme olsun. A dışbükeyse ,

$$\mu_A(\lambda a + (1-\lambda)b) \geq min\{ \mu_A(a), \mu_A(b)\} \qquad [2.75]$$

$\forall$ $a,b \in$ U ve $0 \leq \lambda \leq 1$ bu koşullarda [a,b] aralığında verilen elemanların üyelik değerleri her iki noktanın üyelik değerlerinden az olamaz (Kosko, 1992).

2.3.2.6 Klasik Mantık ile Bulanık Mantığın Karşılaştırılması

- Klasik mantıkla kontrol süreç değişkenlerinin ölçümleri doğru ve kesin olmalıdır.

Bulanık Mantık Kontrol'de kesin olmayan bilgiler kullanılabilir.

- Klasik mantıkla kontrol sürecinin matematiksel modeline ihtiyaç vardır. Bulanık Mantıkta böyle bir matematiksel modele ihtiyaç yoktur.
- Klasik mantıkla kontrol, karmaşık olan sistemlerde, kontrolcü sistemini de karmaşık yapacağından uygulamaya geçirilişi ekonomik olmayabilir. Açık sistemlerde de durum aynıdır. Ancak Bulanık Mantık Kontrol'de ucuz algılayıcılar sayesinde sürecin ölçümünde esneklik kazanılır. Böylelikle uygulamaların hızlı olması, ucuza mal olması ve kolaylaşması sağlanır.

2.4 Sözel Değişkenler

Üyelik fonksiyonuna ek olarak, bulanık kümelerde anlamlı sözel ifadelerde incelenir. Bir bulanık kümenin sözel ifadelerle (linguistic) gösterilmesi iki önemli yarar sağlar. İlk olarak, bu tür gösterim şekli ile uzmanlar bilgi ve deneyimlerini daha kolay açıklayabilmektedir. İkinci olarak, bilgi ve deneyimler sözel ifadeler kullanılarak daha kolay anlaşılır duruma gelmektedir. Bu yarar sayesinde bulanık mantık sistemlerinin tasarlanmasında, yenilenmesinde, ve bakımının yapılıp sistemin çalıştırılmasında önemli kazançlar sağlanmaktadır. Bulanık mantıkta iki yönden verimliliği arttıran bu önemli kavram *sözel değişken* (linguistic variable)'dir.

Bulanık küme ifadelerinde yer alan sözel ifadelerin genel durum değişikliklerine göre anlam değişikliğine uğraması dikkat edilmesi gereken önemli bir noktadır. Örnek olarak; kısa boylu insan ifadesinin anlamı Amerika ve Tayvan'a göre değişik boy ifadesini çağrıştırır. Yine bu ifade; Amerika'da da, halkın genel boy ortalamasından farlılık gösteren değişik gruplar içinde tanımlanmak istense kısa boyluluk kavramı değişik uzunluk değerlerine karşılık gelir. Örneğin; "NBA'deki kısa boylu insan" ifadesi "kısa boylu Amerikalı" ifadesinden daha farklı anlam ifade etmektedir. Bir terim genel durum içinde ifade edilişine bağlı olarak anlam kazandığından, genel durum kesin ve net bir şekilde açıklanmamış olsa bile bir bulanık kümenin, bir genel durum ifadesi ile tanımlanması unutulmaması gereken önemli bir noktadır. Sözel terimlerle ifade edilen genel durum, kullanılan uygulama içinde genellikle dolaylı olarak belirtilir ve bu yüzden nadiren de olsa yanlış anlaşılmalar oluşur.

Bulanık kümelerdeki temel kavramlar incelediğinde, bunların nasıl kullanıldığı da görülür. Bir değişken değeri tanımlamak için herhangi bir küme gibi bulanık kümeler de kullanılabilir. Örnek olarak; "Satışın tutarı az" cümlesindeki 'az' bulanık kümesi, stok marketteki bir günlük satış miktarını belirlemekte kullanılır. Daha biçimsel bir ifade ile şu şekilde açıklanabilir; "Satış miktarı az".

Bu örnekteki *satış miktarı* değişkeni, bulanık mantıkta sözel değişkenlerin önemli bir kavram olduğunu gösterir. Bir sözel değişken kendi değerini nitel ve nicel olarak ifade edebilir. Nitel özelliğini, sözel terim olarak (örneğin, bulanık kümenin ismini sembolize ederek); nicel özelliğini üyelik fonksiyonuna bağlı olarak (bulanık kümenin anlamını açıklayarak) tanımlar. Üyelik fonksiyonundan, sayısal giriş bilgilerinin işlenmesinde faydalanılırken; sözel terimler, insan iletişiminde yer alan bilgi ve kavramların ifadelendirilmesinde kullanılır.

Sözel değişken, sembolik değişkenin (değeri bir sembolle ifadelendirilmiş değişken) ve sayısal değişkenin (değeri bir sayı ile ifadelendirilmiş değişken) birleşiminden oluşur. Sembolik değişkene bir örnek verilirse;

Şekil = Silindir

Burada *Şekil* değişkeni, bir nesnenin şeklini gösterir. Sayısal değişkene de bir örnek verilirse;

Yükseklik = 4

Sembolik değişkenler yapay zeka ve bilimsel kararlarda önemli bir yere sahipken, sayısal değişkenler bilim, mühendislik, matematik, sağlık ve diğer bilim dallarında kullanılır. Sözel değişken kavramı kullanılarak, bu iki değişken bir çatı altında birleştirilebilir. Böylece elde edilen en önemli sonuçlardan biri;bulanık mantığın mühendislik ve daha bir çok alandaki sürekli problem çözümlerinde ortaya koyduğu zekice yaklaşımların başarılı olmasıdır.

Borsa satış faaliyeti örneğinde satış miktarı ifadesini "normalin altında" , "orta" , "biraz düşük" , "çok düşük" gibi tanımlarla gösterebiliriz. Bütün bu farklı tanımları tek tek sıralamak yerine niteleyiciler (örnek olarak; "çok", "daha fazla ya da daha az") ve bağlaçlar (örnek olarak; "ve", "veya") yardımıyla sözel terimlerden oluşmuş bir genelleme (terim kümesi olarak adlandırılan) yapılabilir ve bu genelleme bulanık mantığın önemli

kavramlarından biridir. Bulanık mantıkta kullanılan niteleyiciler “karşılık” olarak adlandırılır (Zadeh, 1975).

2.5 Bulanık Mantık ve Olasılık Teorisi

Bulanık mantık ile ilgili değerlendirmelerin çoğu olasılık teorisi gurubundan gelmektedir. Bu genellikle olasılık teorisi ile bulanık mantık arasındaki fark ile ilgili bazı karışıklıklardan kaynaklanmaktadır. Bu karışıklık olasılık ve bulanık mantık arasındaki karmaşık bir ilişkiden ileri gelir. Bir çok açıdan bu ikisi benzerlik gösterirken bazı açılardan da farklılık gösterirler. Bu ikisi arasındaki ilişki çok iyi ayırdedilmelidir. Yani sadece “bulanık mantık olasılık teorisinin akıllı bir maskesidir” ifadesini netleştirmeye değil bunların yararlarını maksimize etmek için iki teknolojiyi öğrenmemiz de bizim için büyük önem taşır.

Bulanık mantık ve olasılık teorisi tornavida ve çekiç gibi iki farklı araçtır. Bu iki araç farklı görevler için tasarlanmışlardır. Bir vidayı yerine oturmak için çekiç kullanılabilir fakat çekiç yerine tornavida kullanmak çok daha etkili olur. Ancak bu araçların hangi işler için hazırlandığını anladıktan sonra bu aletleri uygun bir şekilde kullanabiliriz. Basit olarak, aşina olduğumuz eski bir aracı kullandığımız için, maalesef çok sık olarak yeni bir aracı kullanmayı öğrenmekten kaçınma eğiliminde oluruz. Profesör Zadeh şöyle söylemiştir “Bildiğiniz tek araç çekiç ise her şey çivi gibi gözükür” (Zadeh, 1968).

2.6 Bulanık Kurallar

Bulanık kümeleri kullanarak geliştirilen tekniklerin içinde, bulanık eğer-o halde kuralları, başarılı uygulamalarından dolayı en geçerli kurallardır. Bulanık eğer-o halde kuralları; kontrol sistemleri, karar verme, örnek tanıma ve sistem tanımlama gibi çoğu disipline uygulanabilir. Bu kurallar, finansal ticarette, sanayi ürünlerinde, robotikte, üretimde ve proses kontrolünde kritik rol oynar.

Bir bulanık eğer-o halde kuralı, bir şart ile bir sonuç arasında, bulanık kümeleri ve sözel değişkenleri kullanarak ilişki kurar. Bilinen bir bakış açısına göre, bir eğer-o halde kuralı,

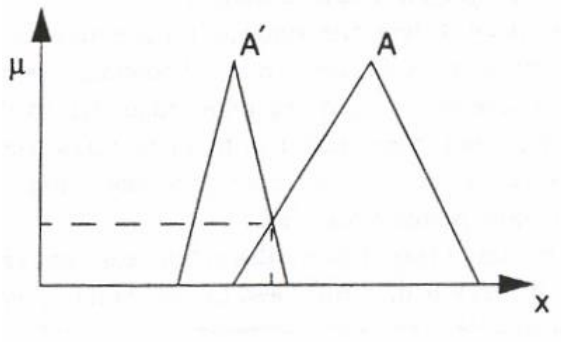
kesin olmayan bilgileri ihtiva eden bir şemadır. Bu kurallar kullanılarak yapılan yargının ana özelliği (bulanık kural tabanlı yargı), onun kısmi eşleme yeteneğidir. Bu da, kuralın şartı sadece kısmi yeterlilik ise bir bulanık kural çiftinden sonuç elde edilmesini (çıkarım) sağlar.

Bulanık eğer-o halde kurallarının öğrenilmesi hem kolay hem de zordur. Birincisi, bulanık kuralların birbirinden farklı iki türü vardır; bulanık haritalama kuralları ve bulanık kapsama kuralları. Bunların başarılı olarak uygulanması için, temel farklılıkların anlaşılmış olması gerekir. İkincisi, çok çeşitli bulanık kural tabanlı anlam çıkarma teknikleri vardır. Bu tekniklerin teorik kurulumlarının anlaşılması ve uygun uygulamalarla ilişkilendirilmesi zor olabilir.

Bir bulanık eğer-o halde kuralı, tabiatıyla kesin ve tam doğru olmayan elde edilmiş bilgi için (tipik olarak insan bilgisi) bilgi temsil şemasıdır. Bu şema, bulanık kuralların "eğer" bölümündeki elastik şartların tanımlanması için sözel değişkenler kullanılarak oluşturulur.

Bulanık kural tabanlı sonuç çıkarma işleminin ana özelliği, onun kısmi eşleme hali altındaki sonuç (anlam) çıkarabilme yeteneğidir. Bu özellik, bir kural şartıyla giriş verisinin eşleme derecesini hesaplar. Şekil 2.14'de, A' bulanık girişi ile A bulanık şartı arasındaki eşleme derecesinin hesaplama yöntemlerinden birisi görülmektedir.

$$\text{eşleşme\_derecesi}\left(A,A'\right)=\sup_x \min\left(\mu_A(x),\mu_{A'}(x)\right) \qquad [2.76]$$



Şekil 2.14: A' Bulanık Girişi ile A Bulanık Şartının Eşlenmesi

Bu eşleme derecesi, bulanık kural tarafından, sonuç çıkarmak için, kurala bağlı olan bölüm (o halde) ile birleştirilir. Daha yüksek eşleme derecesi, daha yakın sonuç çıkarılmasını sağlar.

Kelimeler, bulanık kural tabanlı sistemlerde önemli rol oynar. Elastik şart ve bir bulanık kuralın neticesi, sıklıkla kesin olmayan anlamdaki kelimelerce (sözel etiketler) ifade edilir. Bu kelimeler, özellikle tabiatıyla kesin olmayan bilgilerin kesin ve kolaylıkla anlaşılabilir formda ortaya çıkarılmasını sağlar. Bulanık kurallardaki kelimeler, yapay zekadaki (AI-Artificial Intelligence) klasik kuralların sembollerinden farklıdır. Bir nümerik değişken için sembollerin anlamı, sıklıkla aralıklar kullanılarak tanımlanır. Halbuki bir bulanık kuraldaki kelimelerin anlamı, üyelik fonksiyonlarıyla karakterize edilir. Aralıklar, düzgün ve keskin sınırlara sahiptir. Bir bulanık kuraldaki kelimelerin öneminden dolayı, bulanık kuralları ve daha genel olarak bulanık mantık, "kelimelerle hesaplama" şeklinde adlandırılır (Lee, 1990).

2.6.1 Bulanık kuralların Tipleri

Bulanık kuralların iki tipi vardır:

1) Bulanık haritalama kuralları,
2) Bulanık kapsama kuralları.

Bir bulanık kapsama kuralı, sözel değişkenleri ve kesin olmayan sözel terimleri içeren iki lojik formül arasındaki genelleştirilmiş lojik kapsama ilişkisini tanımlarken, bir bulanık haritalama kuralı, sözel terimleri kullanan girişler ve bir çıkış arasındaki fonksiyonel haritalama ilişkisini tanımlar.

Bulanık haritalama kuralının temeli bir bulanık grafik iken, bulanık kapsama kuralının temeli bulanık mantığın sınırlı hassasiyetidir. Ayrıca bulanık kuralların bu iki tipi, farklı disiplinlere bağlıdır. Bulanık haritalama kuralları, sistemin kimliklendirilmesindeki ve yapay sinir ağlarındaki diğer fonksiyon yaklaşım tekniklerine bağlıdır. Halbuki bulanık kapsama kuralları, klasik iki değerli ve çoklu değerli mantığa bağlıdır.

i) ***Bulanık Haritalama Kuralları:***

Çoğu gerçek problemde, görünür parametreler kümesi ile değerlerini bilmediğimiz tekli yada çoklu parametreler arasındaki fonksiyonel ilişkinin bulunması üzerine yoğunlaşılır. Örnek olarak bir finansal ticaret komisyoncusu (broker), çeşitli finansal göstergeler ve gelecekteki mali değerleri arasındaki ilişkiyle ilgilenir. Bir kontrol mühendisi, görünür durum

değişkenleri ve kontrol edilebilir parametreler arasındaki fonksiyonel ilişkiyi araştırır. Bulanık mantığın çoğu endüstriyel uygulamaları, bulanık haritalama kurallarını kullanır.

Bir yada daha fazla nedenden dolayı, ilgilenilen fonksiyonun yaklaşıklığı ele alınmalıdır. Birincisi, bir fonksiyonun matematik yapısı, tam olarak bilinemeyebilir. Yapı bilindiğinde, parametreleri bulmak için çeşitli parametre tanımlama teknikleri kullanılır. İkincisi, oldukça kompleks bir fonksiyonun tam matematik formunun bulunması, onun yüksek değerinden ötürü imkansız ya da pratik olarak yapılmaktan uzak olabilir. Üçüncüsü, yapılabilir olsa bile bu sefer de maliyeti çok yüksek olur. Bu noktada, düşük maliyetli çok sayıda üretilebilir ürünler ön plana çıkar (örneğin otomobiller, kameralar ve diğer tüketici ürünleri).

Her bir bulanık kural, fonksiyonun küçük bir parçasıdır. Tüm fonksiyon, bulanık haritalama kuralları kümeleri tarafından yaklaşık olarak oluşturulur. Bulanık haritalama kuralları topluluğu, bulanık kural tabanlı modeller veya basit bulanık modeller şeklinde adlandırılır.

Fonksiyon yaklaşım teknikleri, üç kategoride ele alınır; *global teknikler, süper yükleme teknikleri* ve *bölme tabanlı teknikler*. Global teknikler, evrensel bir matematiksel yapıyı kullanarak bir fonksiyonu yaklaştırır (örneğin lineer, ikinci dereceden polinomsal). Bu tekniğin en önemli sonucu, verilen problem için uygun model yapısının bulunmasını sağlamasıdır. Süper yükleme teknikleri, verilen formun süper yükleme fonksiyonlarını kullanarak bir fonksiyonu yaklaştırır (örnek Taylor açılımı). Bölme tabanlı yaklaşım teknikleri, fonksiyonun giriş uzayını bölümlemek ve her bir bölümlenmiş alanı ayrı ayrı yaklaştırmak suretiyle fonksiyonu yaklaşıma tabi tutarlar (lineer yaklaşım).

Bulanık kural tabanlı fonksiyon yaklaşımı, bölme tabanlı bir tekniktir. Bu teknik, komşu alt bölgelerle kısmen aynı olan bir alt bölge genellemesi yapar (bulanık bölümleme). Alt bölgelerin kısmen aynı olduğu giriş uzayında bir çeşit interpolasyon tekniği kullanılarak fonksiyon yaklaştırılır.

ii)*Bulanık Kapsama Kuralları*

Bulanık kapsama kuralları, iki değerli lojikteki kapsamanın genelleştirilmiş halidir. Bu kuralların amacı, insanın yeteneklerini, doğa tarafından oluşturulan kesin olmayan olaylar karşısındaki fikirlerini ve durumlarını taklit edebilmektir. Bulanık kapsama kuralları için, istenilen nitelikleri sağlayan tek bir küme yoktur. Çok çeşitli kümeler geliştirilmiştir. Bunlar,

bulanık kapsama kurallarını kullanarak farklı nedensel şemaların karşılaştırılmasında oldukça kullanışlıdır.

Bulanık kapsamalardan çıkarılan anlam, klasik mantıktaki kapsamalar kullanılarak iki lojik kapsamasının genelleştirilmesini sağlar; *MODUS PONENS* ve *MODUS TOLENS*. Bu lojik kapsamalar, kısaca şu örnekle açıklanabilir:

EĞER bir kişinin IQ'su yüksekse, O HALDE kişi akıllıdır.

Verilen kapsama ve gerçek, yani örneğin Ali'nin IQ'sunun yüksek oluşu, bir *modus ponens*' tir ve Ali'nin akıllı olduğu çıkarımını yapmamızı sağlar. Diğer taraftan, eğer "Ali akıllı değildir" kapsaması ve gerçeği verilirse, *modus tollens* durumu oluşur ve buradan "Ali'nin IQ'su yüksek değil" çıkarımı elde edilir.

Bu kapsamalar, kusursuz eşleme üzerinde durur. Eğer Ali'nin IQ'su daha çok ya da daha az yüksekse, örnek için, yukarıdaki kapsamadan *modus ponens* durumu çıkarılamaz. Çünkü "daha çok ya da daha az akıllı", "akıllı" için bir tanımlama değildir. Bununla birlikte ortak hislerimiz, sıklıkla "Ali daha çok ya da az akıllı" şeklinde bir sonuç çıkarır. Ayrıca bu anlam çıkarmalar, kesin olarak kullanılamaz. Örnek için, eğer Ali bize IQ'sunun yüksek olduğunu söyler fakat bunu destekleyemezse, mantık, kesin olmayan bir olgudan sonuç çıkaramaz.

Bu sınırlamalar, L. A. Zadeh'i bir sorgulama şeması geliştirmeye yöneltmiştir. Bu şema; kısmi eşleme şartı altında genel his sorgulaması ve durumun kesinlik derecesi tayinini yapabilmesidir. Özellikle, mantık kapsamları kısmi eşlemeyi sağlar hale getirilmiştir. Örneğimizde, "Ali'nin IQ'su biraz yüksektir" den "Ali biraz akıllıdır" sonucunu çıkarabiliriz. Böylelikle bulanık kapsama, "eğer Ali'nin IQ'su yüksekse, o halde Ali akıllıdır" ibaresinden aşağıdaki hale gelir (Çizelge 2.2).

Çizelge 2.2: Mantık Kapsamları

Verilen	**Çıkarım**
Ali'nin IQ'su yüksek	Ali akıllıdır

Ali'nin IQ'su biraz yüksek	Ali biraz akıllı

İkinci mantık sınırlaması, klasik mantığa başka bir ekleme yapmıştır; çoklu değerli mantık. Çoklu değerli mantık, mantıktaki doğruluk değerlerini binary' den (doğru ya da yanlış) çoklu (ikiden fazla) yapıya dönüştürür. Burada kesinliğin farklı dereceleri, farklı doğruluk derecelerince tanımlanır. Doğruluk derecelerinin klasik mantıktaki doğru ve yanlışın ötesinde geliştirilmesinden beri, bulanık mantık, çoklu değerli mantıkla anılır olmuştur. Bununla birlikte bulanık mantık, çoklu değerli mantıktan mantığın ilk sınırlama problemine hitap etmesi noktasında ayrılır. Bir bulanık kapsama, ilk probleme hitap eder. Ki bu da, kendisinden önceki sözel değişkenleri kullanır. Netice itibariyle, kendisinden önceki bildirim, kısmen kabul edilen esnek bir durumu tanımlar. Örneğimizde "bir kişinin IQ' su", sözel bir değişkendir. "Bir kişinin IQ' su yüksektir" ifadesi, kişinin IQ' suna dair esnek bir durumu tanımlar. Bu, biraz yüksek IQ' lu kişi ile, durumun kısmi eşlenmesini sağlar. Durumun derecesi, bulanık kapsamadan çıkarılır.

Bulanık kapsama kuralları ve bulanık haritalama kuralları arasındaki ayırımlar, kolayca belirlenemez. Prof. Zadeh, bulanık mantığın 4 temel yaklaşımını ortaya koymuştur:

1) Mantıksal yaklaşım,
2) Küme-Kuram yaklaşımı,
3) Bağlamsal yaklaşım ve
4) Deneysel (Epistemic) yaklaşım.

Bulanık kapsama kuralları, *mantıksal* bir yaklaşımdır. Oysa bulanık haritalama kuralları, *bağlamsal* bir yaklaşımdır. Çünkü fonksiyonel haritalama, bir çeşit ilişkidir. Bu iki tip bulanık kural arasındaki önemli farklılıklar, Çizelge 2.3'de verilmiştir.

Çizelge 2.3: İki Tip Bulanık Kuralın Karşılaştırılması

	Bulanık Kapsama Kuralları	**Bulanık Haritalama Kuralları**

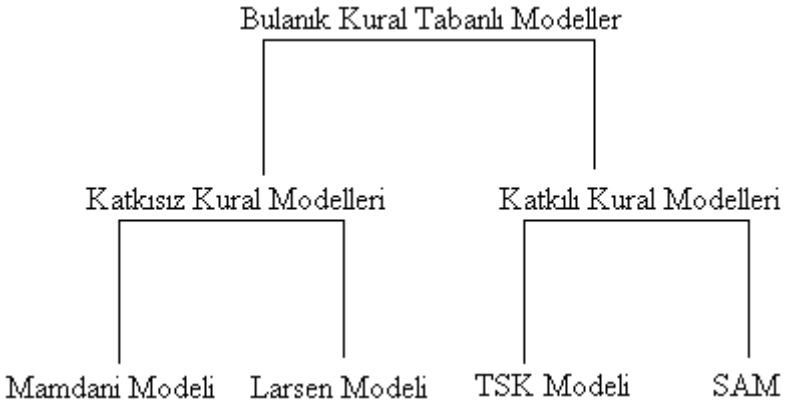
Amaç	Kesin olmayan yönetim için genel kapsamlar	Yaklaşık fonksiyonel haritalama
Arzulanan Çıkarım	Genelleştirilmiş modus ponens ve modus tollens	Sadece ilerisi
Uygulama	Teşhis, yüksek seviyeli karar verme	Kontrol, sistem modelleme ve işaret işleme
Bağlı Disiplinler	Klasik mantık, çoklu değerli mantık	Sistem tanımlama, lineer interpolasyon, neural networkler
Tipik Tasarım Yaklaşımı	Bireysel tasarım	Bir kural kümesi şeklinde tasarım
Uygun Problem Alanları	Sürekli ve Ayrık değişkenli alanlar	Sürekli nonlineer alanlar

2.6.2 Bulanık Kural Tabanlı Model Tipleri

Fonksiyon yaklaşımı için, 4 tip bulanık kural tabanlı model tipi vardır;

1) Mamdani modeli
2) Larsen modeli
3) Takagi-Sugeno-Kang (TSK) modeli
4) Kosko'nun katkı modeli (SAM)

SAM'ın sonuç şeması ile TSK modeli benzerdir. İkisi de çıkarım sonucunda çoklu kural yaklaşımını kullanır. Mamdani modeli, çıkarım sonuçlarını kullanır. Katkısız kural modelidir. Şekil 2.15'de bulanık kural tabanlı model sınıflandırması görülmektedir.



Şekil 2.15: Fonksiyon Yaklaşımı için Bulanık Kural Tabanlı Modellerin Sınıflandırılması.

Geliştirilen ilk kural tabanlı model, Mamdani modelidir. Çoğu Bulanık Mantık Kontrol sistemleri, Mamdani modeli kullanılarak 80'lerde geliştirilmiştir. Buna örnek, 1980 yılında P.M. Larsen, tarafından gerçekleştirilen Larsen modelidir. TSK modeli ilk olarak T. Takagi ve Prof. M. Sugeno tarafından 1985'te sunulmuştur. Sugeno'nun başka bir öğrencisi olan K. T. Kang, model üzerinde çeşitli uygulamalar yapmıştır. TSK modeli, daha çok 90'larda endüstriye uygulanmaya başlanmıştır. TSK modelinin ana avantajlarından biri, bir fonksiyonu daha az kural kullanarak yaklaşıma tabi tutmasıdır.

Mamdani modeli ve SAM, bir bulanık kümenin bölümü olan kuralları kullanır.

R_i : Eğer x_1, A_{i1} ve x_2, A_{i2} $ve \ldots ve$ $x_\mathrm{s}, \mathrm{A}_{is}$

O halde $y = C_i$ dir. $i = 1,2,\ldots,M$

Burada M, bulanık kuralların sayısıdır. $x_j \in \mathrm{U}_j (j = 1,2,\ldots,\mathrm{s})$ değerleri giriş değişkenleridir. $y \in \mathrm{V}$ çıkış değişkenidir. A_{ij} ve C_i, $\mu_{A_{ij}}(x_j)$ ve $\mu_{C_i}(y)$ üyelik fonksiyonlarınca karakterize edilen bulanık kümelerdir.

Bu iki bulanık kural tabanlı model, çıkarım şemaları bakımından ayrılır. TSK modeli, lineer bir modelin bölümü olan bir kuralı kullanır.

R_i : Eğer x_1, A_{i1} ve x_2, A_{i2} ve...ve x_s, A_{i5}

O halde $y = f_i(x_1, x_2, \ldots, x_s),\ i = 1,2,\ldots,M$

Burada f_i , lineer bir fonksiyondur.

2.6.2.1 Mamdani Modeli

Pratikte en geniş kullanım alanına sahip model, Mamdani modelidir. Sözel kurallardan oluşur ve $U_1 \times U_2 \times \ldots \times U_r$'den W'ya bir haritalama tanımlar.

R_i : Eğer $x_1, A_{i1}\ ve \ldots ve\ x_r, A_{ir}$

O halde $y = C_i$

Burada $x_j (j = 1,2,\ldots,r)$ giriş değişkenleri, y çıkış değişkenleri, A_{ij} ve C_i ise, x_j ve y için bulanık kümelerdir. Bu form için verilen girişler,

$x_1, A_1^{'}\ x_2, A_2^{'}, \ldots, x_r\ A_r^{'}$

şeklindedir. Burada $A_1^{'}, A_2^{'}, \ldots, A_r^{'}$; $U_1, U_2, \ldots, U_r$ (bulanık numaralar)'ın bulanık alt kümeleridir. R_i kuralının Mamdani modelinin çıkışına katkısı;

$$\mu_{C_i^{'}(y)} = (\alpha_{i1} \wedge \alpha_{i2} \wedge \ldots \wedge \alpha_{in}) \wedge \mu_{C_i}(y) \qquad [2.77]$$

şeklinde hesaplanan bir bulanık kümedir.

Burada α_i, R_i kuralının eşleme derecesidir. α_{ij} ise, x_j ve R_i arasındaki eşleme derecesidir.

$$\alpha_{ij} = \sup_{x_j} \left(\mu_{A_i^{'}}(x_j) \wedge \mu_{A_{ij}}(x_j) \right) \qquad [2.78]$$

Burada $\wedge$, *min* operatörüdür.

Modelin sonuç çıkışı, *max* operatörünü kullanan tüm kurallardan elde edilen çıkışların kümesidir.

$$\mu_C(y) = max\left\{ \mu_{C_i^{'}}(y), \mu_{C_2^{'}}(y), \ldots, \mu_{C_L^{'}}(y) \right\} \qquad [2.79]$$

Burada C çıkışı, bir bulanık kümedir. Bu bulanık çıkış, belirtilen durulaştırma tekniklerinden biri kullanılarak keskin bir çıkış halinde durulaştırılabilir. Mamdani modelinden elde edilen sonuç şemasını tanımlamak mümkündür. Aşağıdaki operatörler kullanılarak model oluşturulabilir.

- Sup-min düzenlemesi
- Kartezyen çarpım için *min*
- Kurallar dahilinde birleşme şartları için *min*
- Kümelenmiş çoklu kurallar için *max*

$*$, monoton azalmayan bir işlem olsun (örneğin; eğer *a*, *b*, *c*, *d* reel sayılarsa ve $a \geq b$, $c \geq d \rightarrow a * b \geq c * d$ $*$ operatörünü $\wedge$(min) ve $\vee$(max) üzerine dağıtırsak;

$$a * (b \wedge c) = (a * b) \wedge (a * c) \qquad [2.80]$$

$$a * (b \vee c) = (a * b) \wedge (a * c) \qquad [2.81]$$

elde edilir (Mamdani, 1974).

2.6.2.4 Larsen Modeli

Pratikte geniş kullanım alanına sahip diğer bir model, Larsen modelidir. Mamdani modelinde olduğu gibi Larsen modeli de sözel kurallardan oluşur ve $U_1 \times U_2 \times \ldots \times U_r$'den W'ya bir haritalama tanımlar.

R_i: Eğer x_1, A_{i1} *ve*…*ve* x_r, A_{ir}

O halde $y = D_i$

Burada $x_j (j = 1,2,\ldots,r)$ giriş değişkenleri, *y* çıkış değişkenleri, A_{ij} ve D_i ise, x_j ve *y* için bulanık kümelerdir. Bu form için verilen girişler,

x_1, A_1' $x_2, A_2', \ldots, x_r$ A_r'

şeklindedir. Burada $A_1^{'},A_2^{'},\ldots,A_r^{'}$; $U_1,U_2,\ldots,U_r$ (bulanık numaralar)'ın bulanık alt kümeleridir. R_i kuralının Larsen modelinin çıkışına katkısı;

$$\mu_{C_i^{'}(y)} = (\alpha_{i1} \wedge \alpha_{i2} \wedge \ldots \wedge \alpha_{in}).\mu_{C_i}(y) \qquad [2.82]$$

şeklinde hesaplanan bir bulanık kümedir.

Burada α_i, R_i kuralının eşleme derecesidir. α_{ij} ise, x_j ve R_i arasındaki eşleme derecesidir (Denklem 2.78).

$$\alpha_{ij} = \sup_{x_j}\left(\mu_{A_i^{'}}(x_j) \wedge \mu_{A_{ij}}(x_j)\right)$$

Burada $\wedge$, *min* operatörüdür.

Modelin sonuç çıkışı, çarpım operatörünü kullanan tüm kurallardan elde edilen çıkışların kümesidir.

$$\mu_C(y) = \left\{\mu_{C_i^{'}}(y).\mu_{C_2^{'}}(y).\ldots.\mu_{C_L^{'}}(y)\right\} \qquad [2.83]$$

Burada D çıkışı, bir bulanık kümedir. Bu bulanık çıkış, belirtilen durulaştırma tekniklerinden biri kullanılarak keskin bir çıkış halinde durulaştırılabilir.

2.6.2.2 TSK Modeli

Takagi-Sugeno-Kang (TSK) modeli 1984 yılında, T. Takagi ve M. Sugeno tarafından ortaya atılmıştır. Daha sonra M. Sugeno ve K. T. Kang bu modeli geliştirmiştir.
Bu modelin geliştirilmesinin ana nedeni, Mamdani modelince talep edilen, özellikle kompleks ve yüksek boyutlu problemler için kuralların sayısını azaltma isteğidir. Bu hedefi başarmak için TSK modeli, lineer giriş değişkenleri denklemi ile Mamdani kuralının o-halde bölümüne bağlı bulanık kümelerin yerine konulmuştur. Örneğin iki girişli ve bir çıkışlı TSK modeli, aşağıda belirtilen formdaki kurallardan oluşur:

Eğer x, A_i ve y, B_j

O halde $z = ax + by + c$

Burada a, b ve c, nümerik sabitlerdir.

Genelde TSK modelindeki kurallar, şu forma sahiptir:

Eğer $x_1, A_{i1}\ ve \ldots ve\ x_r, A_{ir}$

O halde $y = f_i(x_1, x_{2,\ldots,} x_r) = b_{i0} + b_{i1}x_1 + \ldots + b_{ir}x_r$

Burada f_i, lineer modeldir ve b_{ij} (j = 0,1,.....,r) ise gerçek değerli parametrelerdir.

TSK modelinden sonuç çıkarma, tüm ilişkili lineer modellerin interpolasyonudur. Bir lineer modelin ilişki derecesi, lineer modelle birleşen bulanık alt uzaya ait giriş verisinin derecesince tanımlanır. Bu ilişki dereceleri, interpolasyon sürecinde önemli hale gelir.

Modelin toplam çıkışı, aşağıdaki denklemde verilmiştir. Burada α_i, R_i kuralının eşleme derecesidir. Bu, Mamdani modelinin denklem 2.82'de hesaplanan eşleme derecesiyle benzerdir.

$$y = \frac{\sum_{i=1}^{L} \alpha_i f_i(x_1, x_2, \ldots, x_r)}{\sum_{i=1}^{L} \alpha_i} = \frac{\sum_{i=1}^{L} \alpha_i (b_{i0} + b_{i1}x + \ldots + b_{ir}x_r)}{\sum_{i=1}^{L} \alpha_i} \qquad [2.84]$$

TSK modelinin girişleri, keskin (non-fuzzy) sayılardır. Bu yüzden, i'inci kuralla eşlenen $x_1 = a_1, x_2 = a_2, \ldots, x_r = a_{1r}$, giriş derecesi tipik olarak *min* operatörü kullanılarak hesaplanır.

$$\alpha_i = min\left(\mu_{A_{i1}}(a_1), \mu_{A_{i2}}(a_2), \ldots, \mu_{A_{ir}}(a_r)\right) \qquad [2.85]$$

Bununla birlikte çarpım operatörü de kullanılabilir.

$$\alpha_i = \mu_{A_{i1}}(a_1) \times \mu_{A_{i2}}(a_2) \times \ldots \times \mu_{A_{ir}}(a_r) \qquad [2.86]$$

TSK modelinin, aşağıdaki 3 kuraldan ibaret olduğunu göz önüne alalım.

Eğer *x* küçük ise $y = L_1(x)$

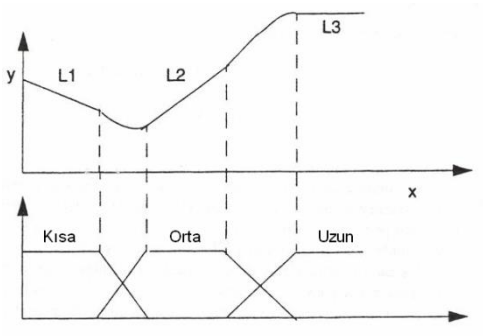
Eğer *x* orta ise $y = L_2(x)$

Eğer *x* büyük ise $y = L_3(x)$

Model çıkışı;

$$y = \frac{\mu_{small}(x) \times L_1(x) + \mu_{medium}(x) \times L_{2(1)}(x) + \mu_{large}(x) \times L_3(x)}{\mu_{small}(x) + \mu_{medium}(x) + \mu_{large}(x)} \qquad [2.87]$$

olur. Bu durum, şekil 2.16 ile ifade edilir.



Şekil 2.16: TSK Modeline bir Örnek

TSK modeli, kompleks sistemlerin modellenmesi için güçlü bir oluşumdur. Az sayıda kural kullanarak yüksek non-lineer fonksiyon ilişkileri çözülebilir (Takagi ve Sugeno,1985).

TSK modelinin en büyük avantajı, onun güçlü anlatım özelliğidir. Daha da fazlası, açık fonksiyonel anlatım formu, bazı öğrenme algoritmalarını kullanarak parametre tanımlaması yapabilmesini sağlar. ANFIS(Adaptive Neuro-Fuzzy Inference System) gibi çeşitli neuro-fuzzy sistemler, TSK modeli temel alınarak yapılandırılmıştır.

2.6.2.3 Kosko Standart Katkı Modeli (SAM)

SAM modeli, 1996'da B. Kosko tarafından geliştirilmiştir. SAM'daki bulanık kuralların yapısı, Mamdani modeliyle benzerdir. Ancak bu iki model arasında sonuç çıkarma şeması açısından 4 farklılık vardır:

1) SAM, girişleri keskin olarak tanımlarken Mamdani modeli keskin ve bulanık olarak ele alır.

2) SAM, ölçekli çıkarım metodunu kullanırken Mamdani modeli kesme metodunu kullanır. Bu fark, iki modelce seçilen düzenleme operatörlerinden kaynaklanır.

3) SAM, bulanık kural sonuçlarını birleştirmek için ek (toplama) yaparken Mamdani modeli *max*'ı kullanır.

4) SAM, centroid durulaştırma tekniğini içerirken Mamdani modeli, özel bir durulaştırma metodu üzerinde durmaz.

İki giriş bir çıkış modeli üzerinde üzerinde duralım.

Eğer x, A_i ve y, B_i

O halde z, C_i dir.

Verilen keskin girişler, $x = x_0$, $y = y_0$ olduğuna göre model çıkışı;

$$z = Centroid\left(\sum_i \mu_{A_i}(x_0) \times \mu_{B_i}(y_0) \times \mu_{C_i}(z) \right) \qquad [2.88]$$

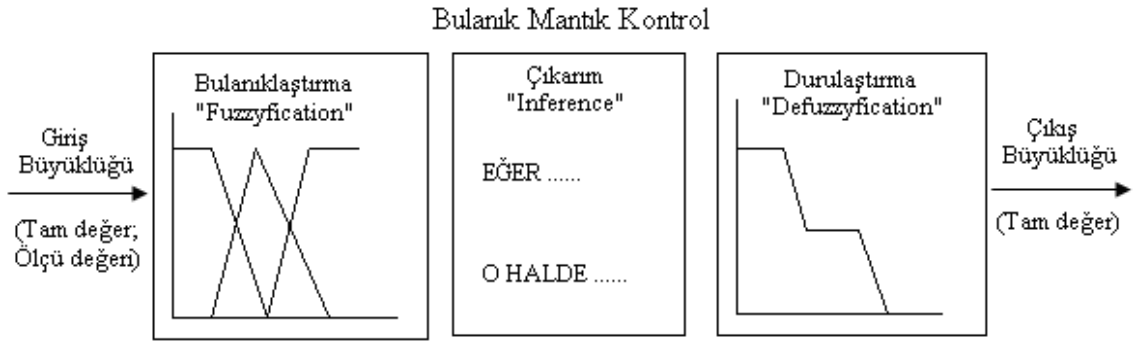
olur. Centroid bir fonksiyondur. Denklem 2.88, yeterli bir hesaplama yapmak için başka bir forma dönüştürülebilir.

Standart katkı modelinin ana avantajı, hesaplama verimliliğidir. Örneğin kurallar bir kez tanımlandığında, bunlar sabit halini alırlar. Standart katkı modelinin temeli bir bulanık grafiktir (Kosko, 1997).

2.7 Bulanık Mantık Kontrol

Bulanık mantığın esas alanı, son bulma hüküm bulmanın yanında kontrol tekniğidir. Hızlı süreçteki hatalı değerler kolaylıkla düzenlenebilir. Bunun yanında sürecin tam olmayan matematiksel modeli tanınır. Anlam bakımından bulanık mantığın kontrol tekniğinde kullanımı olan Bulanık Mantık Kontrol daha fazla görülür. Bulanık Mantık Kontrol bir matematiksel süreç modeli değildir. Esas itibari ile giriş çıkış büyüklüklerinin gramer yapıya ait büyüklüklerin formülasyonu temeli üzerine işlenmesi kuralıdır. Örnek olarak "sıcaklık yüksek ve eğer basınç çok yüksek ise vana tamamen kapanır" zor veya kısmen anlaşılamayan süreç, parametreli süreçte düzenlenebilir.
Bulanık Mantık Kontrol üç ana yapı içerir (Şekil 2.17).

1) Bulanıklaştırma
2) Çıkarım
3) Durulaştırma



Şekil 2.17: Bulanık Mantık Kontrol Sistemi

Bulanıklaştırma ile bir giriş büyüklüğü, tam bir değeri veya keskin bir kümeyi bulanık bir büyüklük formuna getirilmektedir. Çıkarımda, bulanık kümeler bir arada birleştirilir. Durulaştırma kısmında ise sonuç tekrar tam değere transpoze edilir.

2.7.1 Bulanıklaştırma

Bulanıklaştırma, sistemden alınan kontrol giriş bilgilerini dilsel niteliyiciler olan sembolik değerlere dönüştürme işlemidir.

Uygulama Yöntemi:

1) Bulanık kümenin tespiti
2) Üyelik fonksiyonunun tespiti
3) Üyelik derecesinin toplamı

Örnek: Banyo suyu sıcaklığının bulanıklaştırılması incelenecektir. Özellikler "Ilık", "Sıcak" ve "Çok Sıcak" tır.

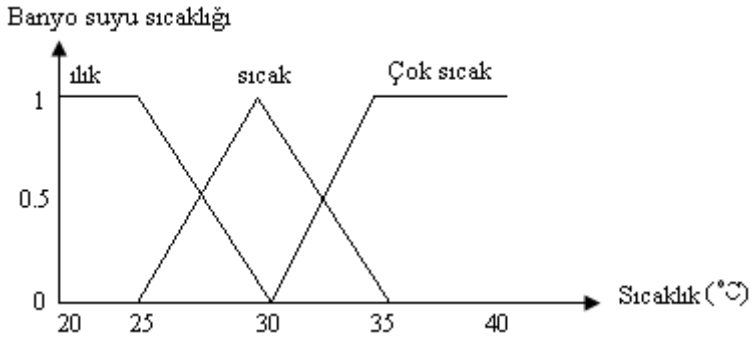
1- Ayrı ayrı bulanık kümelerin tespiti:

- Ilık
- Sıcak
- Çok sıcak

2- Üyelik fonksiyonunun Tespiti:

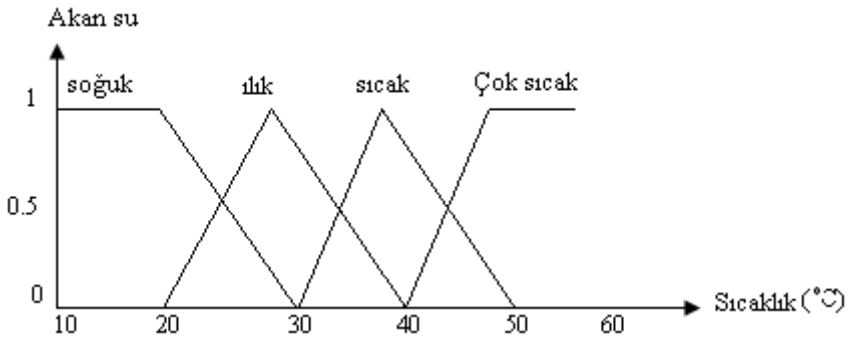
Üyelik fonksiyonun tespitinde geçiş problemi tespit edenin keyfi tespiti olabilir. Dolayısı ile geçişlerin anlamlı olmasına dikkat edilmelidir. Şayet netice kesin değil ise verilen durum geçişi modifiye edilmelidir. Şekil 2.18'de üyelik fonksiyonu için geçiş keyfi seçilmiştir.

Grafik formda verilen üyelik fonksiyonu yerine başka formlarda (Tablo/Matris(Look up Tables), matematiksel formda üyelik fonksiyonu) düşünülebilir.



Şekil 2.18: Banyo Suyu Sıcaklığının Üyelik Fonksiyonları

Bulanıklaştırmada, çıkış büyüklükleri bulanık kümesi için üyelik fonksiyonunun tespitine dikkat edilmelidir. Banyo sıcaklığı örneğinde bulanık küme akan suyun sıcaklığıdır. Şekil 2.19'de üyelik fonksiyonu keyfi (üçgen) seçilmiştir.

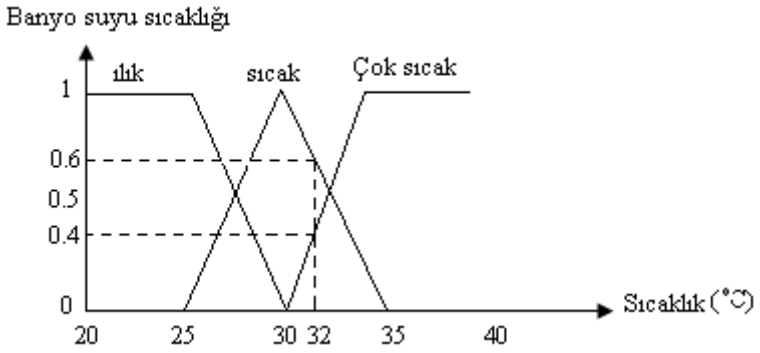


Şekil 2.19:Akan Suların Üyelik Fonksiyonu

3- Üyelik Derecesinin Toplamı

İki nokta altında tespit edilen üyelik fonksiyonu (Şekil 2.20'de) grafiksel geçişi 32ºC banyo suyu sıcaklığı için şekile göre verilebilir.

$\mu_{\text{ılık}}(32^oC) = 0,\ \mu_{\text{sıcak}}(32^oC) = 0.6,\ \mu_{\text{çok sıcak}}(32^oC)=0.4$



Şekil 2.20: Üyelik Derecesinin İncelenmesi

32C deki banyo sıcaklığı %60 sıcak, %40 çok sıcak ve %0 ılık olarak derecelendirilir. Ayrı ayrı üyelik derecelerinin toplamı yine 1(%100) neticesini verecektir. Kontrol tekniğinde özellikle pratik olarak, parça parça lineer geçiş doğruluğu sabit olarak dikkate alınır. Fakat tamamen başka geçişlerde düşünülebilir.

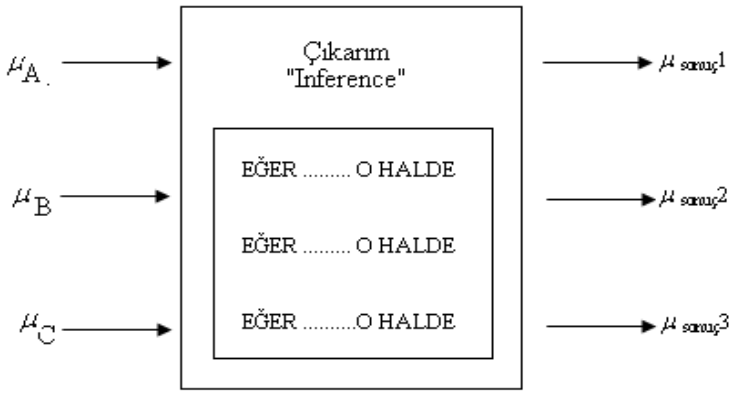
Soyut üyelik fonksiyonu matematiksel eşitlik şeklinde ise bulanıklaştırma işlemi hesap yolu ile yapılabilir. Look-Up-Tablosu olarak adlandırılan gösterimde Şekil 2.20'deki durumlar mümkündür. Look-Up-Tablosu bir tablodur. Bu tabloda düzenlenen ayrı ayrı değerler veya ilgili üyelik derecelerinin giriş değişkenleri değerleri gösterilmiştir. Burada mevcut olmayan değer için ara değer alınmalıdır. Yukarıda ifade edilen değer için çok kaba olarak Çizelge 2.4'deki gibi bir "Look-Up Tablosu" verilebilir.

Çizelge 2.4: Look-Up Tablosu

	20°C	24°C	28°C	32°C	36°C	40°C
μ ılık	1	1	0.4	0	0	0
μ sıcak	0	0	0.6	0.6	0	0
μ çok sıcak	0	0	0	0.4	1	1

2.7.2 Çıkarım

Çıkarımda tespit edilen kurallar (Operatör kuralı, bulanıklaştırmada bulunan μ_i üyelik derecesi kuralları) yorumlanır (Şekil 2.21). Bu işlem sonunda üyelik derecesi çıkış büyüklükleri ve kısmi sonuç kümesi üyelik derecesi bulunur.



Şekil 2.21: Çıkarım Mekanizması

Uygulama yöntemi:

1. Operatör kuralının Tespiti
2. VE, VEYA, v.s. için operatör tespiti
3. Kısmi sonuç kümesi üyelik derecesi hesabı

Örnek: Bir önceki bölümde verilen banyo suyu sıcaklığı üyelik derecesi çıkarımı: banyo suyu sıcaklığının ani değerine bağlı olarak akan suyun sıcaklığı sıcak/soğuk gibi tespit edilmelidir.

1.Operatör kurallarının tespiti :

Operatör kurallarının tespitinde karışık matematiksel model yerine insani düşünme yönteminden faydalanılır, eğer (ön şart) o halde (sonuç). Bu kurallar tecrübeye bağlıdır.

Bir çok önşart vardır:

EĞER (önşart_1) VE/VEYA/GAMA(önşart_2) VE/VEYA/GAMA(önşart_3).....

O HALDE (sonuç)

Banyo suyu sıcaklığı örneğine bakarsak yaklaşık değer aşağıdaki gibi görülebilir:

EĞER banyo suyu ılık ise,

O HALDE akan su çok sıcak

EĞER banyo suyu sıcak ise,

O HALDE akan su sıcak

EĞER banyo suyu çok sıcak ise,

O HALDE akan su çok ılık

Burada küvete akan suyun ölçüsü ılık, sıcak, çok sıcak olmalıdır. Bu yine üyelik derecesidir. Bir tablo üzerinde bir çok çıkarım kuralı ön şart için gösterilebilir (Çizelge 2.5).

Çizelge 2.5: Çıkarım Kuralı Ön Şart Tablosu

Ön şart2 / Ön şart1			
	VE/VEYA Son durum	.....	.....
	VE/VEYA Son durum	.....	.....

Daha önceden belirtilmemiş, tarif edilmemiş durumların ortaya çıkabilmesi son derece önemlidir. Ön şart kombinasyonlarının hepsi kapatılmalıdır. Her giriş büyüklüğü kümesi için bir çıkarım kuralı mevcut olmalıdır. Burada VEYA bağıntısı bütün satır ve sütunları teşkil eder. Bununla beraber VE bağıntısı sadece kullanım için bir durum oluşturur.

2-VE/VEYA için operatör tesbiti:

Şayet işlem kurallarında VE/VEYA/GAMA gibi bağıntılar ortaya çıkmışsa bunlar için uygun operatörler seçilmelidir (*min max* operatörleri). En iyi operatörün seçimi için herhangi bir kural yoktur. Seçilen operatör kural/tabanı verilen problemi kolaylıkla çözebilmelidir. Eğer böyle olmazsa başaka bir operatöre karşı aynı bağıntı değiştirilmelidir.

Pratikte VE bağıntısı için *min*-operatörü, VEYA-bağıntısı için *max*-operatörü kullanılmıştır.

3.Kısmi Sonuç Kümesi üyelik derecesinin hesabı:

Sadece bir önşartta son durum üyelik derecesi için, ön şarttan üyelik derecesi değeri kabul edilir. Bir çok önşartta Bulanık Mantık Kontrol kurallarına göre ayrı ayrı üyelik derecelerinin değerleri (VE, VEYA, GAMA) bağlantılandırılır. (örnek: *min*-operatörü, *çarpım* operatörü, *gama* operatörü) ilgili operatörlerin seçimi keyfidir. Sadece verilen problem tatminkar çözümlü olmalıdır.

Örnek:

EĞER banyo suyu orta ise,

O HALDE akan su orta

μ_{orta} (B.suyu) = 0.6, μ_{orta} (Akan su) = 0.6 idi.

Birden fazla önşartta :

EĞER banyo suyu sıcak VEYA banyo yolu çok sıcak ise,

O HALDE akan su ılık

μ_{orta} (B.suyu) = 0.6, μ_{orta} (Akan su) = 0.4 idi.

VEYA bağıntısı için *max* operatörü seçilir:

μ_{orta} (Akan su) = max{ μ_{orta} (B.suyu); $\mu_{\text{çok sıcak}}$ (B.suyu) }

= max{0.6;0.4}

=0.6'dır.

Bir çok kural söylenen eşitlikle ispatlanırsa birçok μ üyelik derecesi bir ve aynı bulanık kümeye verilir. Bu μ için bir bağıntı stratejisi seçilir. Mümkün olan diğer stratejiler aşağıdadır.

- Üyelik derecesinin maksimumunu seçmek
- Üyelik derecesinin aritmetik ortalama değerini seçmek
- İki bulanık kümenin VEYA-bağıntısı toplamındaki gibi üyelik derecesi

Her stratejinin kendine özgü bir karekteristiği vardır.

i. Maksimum üyelik derecesi:

VEYA bağıntısı için uygun *max*-operatörü sadece büyük üyelik derecesinde kullanılabilir. Bu seçenekler hızlıdır fakat olumsuz tarafıda vardır. Küçük değerlerin değişimi süreçte değişmeyi ifade eder. büyük üyelik derecesi değişmediği müddetçe bu durum dikkate alınmamalıdır.

ii. Aritmetik ortalama değer:

Çeşitli üyelik derecelerinin ortalama değeri kolaylıkla oluşturulabilir. Burada olayın tamamen maksimumunu kullanmaya göre dengeli karekteristiği vardır. Ayrı ayrı üyelik dereceleri büyüklüğüne göre her durumdaki küçüklük verilen üyelik derecesi karekteristiğidir.

iii. İki bulanık küme toplamı:

Bu strateji oluşturulan iki bulanık küme toplamı için bağıntı kuralı olarak kullanılır. VEYA-operatörü ile gösterilir.

$$\mu_{A+B} = \mu_A + \mu_B - \mu_A.\mu_B \qquad [2.89]$$

$$\mu_{A+A} = \mu_A + \mu_A - \mu_A.\mu_A \qquad [2.90]$$

$$\mu_{A+A} = 2\mu_A - \mu_A^2 \qquad [2.91]$$

Toplam üyelik derecesi, ayrı ayrı üyelik derecesi büyüklüğüne göre daha büyüktür. Bu strateji olayı kuvvetlendirme meyillidir.

Hangi stratejinin kullanılacağı süreç (işlem) kuralına bağlıdır. Ara sıra kuvvetlendirme ara sıra zayıflama arzulanır.

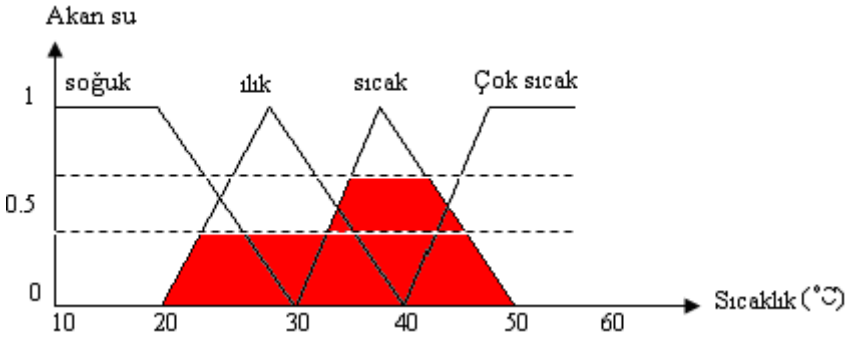
Aşağıda gösterildiği gibi birçok seçenek mevcuttur.

$\mu_{soğuk}$(akan su) = 0, $\mu_{sıcak}$(akan su) = 0.6

$\mu_{\text{ılık}}$(akan su) = 0.4, $\mu_{\text{çok sıcak}}$(akan su) = 0

1. *max*/*min* Metodu:

max/*min* metodunda çıkışa ait ayrı ayrı bulanık kümelerin üyelik fonksiyonu, sonuç büyüklüğü, ilgili üyelik derecesi yüksekliğinde kesilir. Bu durum "clipping" diye isimlendirilir. Elde edilen yüzey toplam yüzey olarak düşünülür. Yukarıda verilen üyelik derecelerinden faydalanarak Şekil 2.22' deki gibi bir durum elde edilir.



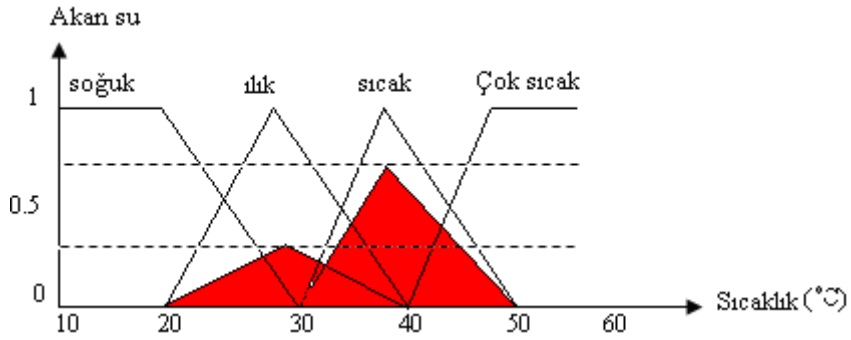
Şekil 2.22: Sonuç Kısmi Yüzey Oluşumu

Elde edilen taralı yüzey toplam yüzeyi içerir ve böylece bulanık sonuç kümesi elde edilir. Akan suyun sıcaklığı durulaştırma esnasında elde edilir.

2. Max/Çarpım Metodu:

Max/Çarpım metodunda çıkışa ait ayrı ayrı bulanık kümelerin üyelik fonksiyonu sonuç büyüklüğü, ilgili üyelik derecesi ile çarpılır. Böylece korunan yüzeyler toplam yüzey olarak düşünülür.

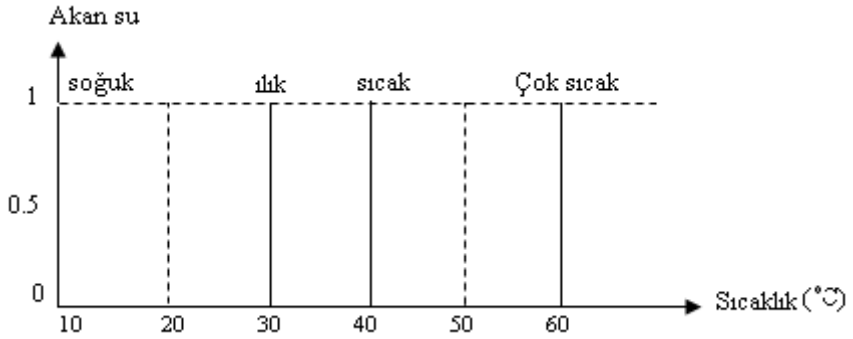
Daha önceden verilen üyelik derecelerinden faydalanılarak Şekil 2.23 elde edilir. Elde edilen taralı yüzey sonuç kümesini meydana getirir.



Şekil 2.23: Sonuç Kısmi Yüzey Oluşumu

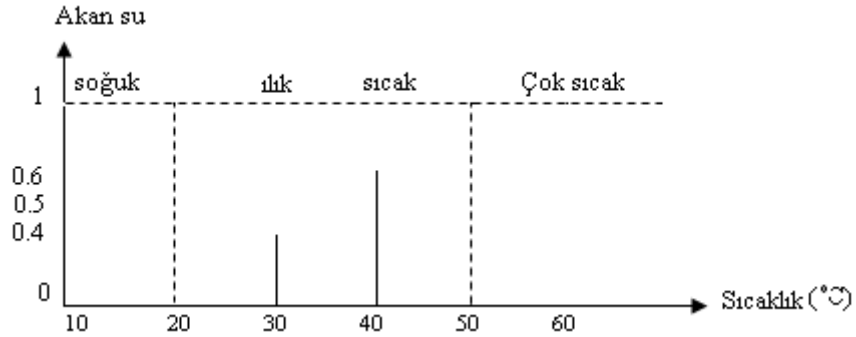
3. Singletons:

Singletons kullanımında bulanık kümenin çıkış değişkenleri sadece ayrı ayrı çizgi ile düzenlenir (Şekil 2.24).



Şekil 2.24: Singletons

Daha önce verilen üyelik derecelerinden faydalanılarak Şekil 2.25'deki durum elde edilir.



Şekil 2.25: Değerlendirilen Singletons

Çözüm kümesi olarak çizgi spektrumu oluşturulur. Akan suyun somut sıcaklığı durulaştırma esnasında elde edilir.

2.7.3 Durulaştırma

Bulanık çıkarımın sonucu bulanık bir kümedir. Bu sonucun tekrar sisteme uygulanabilmesi için giriş değeri gibi sayısal değere dönüştürülmesi gerekir. Bu işlem durulaştırma olarak adlandırılır. Durulaştırma birimi karar verme biriminden gelen bulanık bir bilgiden bulanık olmayan ve uygulamada kullanılacak gerçek değerlerin elde edilmesini sağlar.

Durulaştırma işleminde değişik yöntemler kullanılmaktadır. Önce her kural için üyelik derecelerinden oluşan değer ve sonuç kural tespit edilir. Daha sonra en uygun yöntem seçilerek durulaştırma yapılır. En çok kullanılan yöntemler şunlardır;

- Maksimum üyelik yöntemi,
- Ağırlık merkezi yöntemi,
- Ağırlık ortalaması yöntemi,
- Maksimumların ortalaması yöntemi,

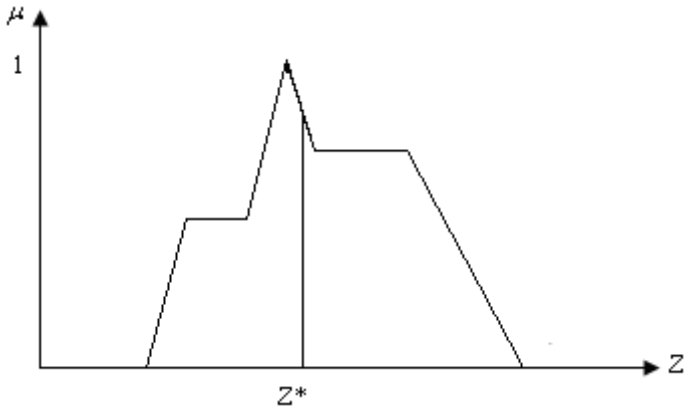
2.7.3.1 Maksimum Üyelik Yöntemi

Yükseklik yöntemi olarak da adlandırılmaktadır. Bütün üyelik dereceleri içinde en büyük olana eşittir ve aşağıdaki gibi ifade edilir.

$$\mu_C(z^*) \geq \mu_C(z) z \in Z \qquad [2.92]$$

Z* çıkış değerinin elde edilişi Şekil 2.26'da görülmektedir.

Burada *c* çıkış üyelik işlevlerinin birleşimini, *z* ise üyelik değerlerini ifade eder.



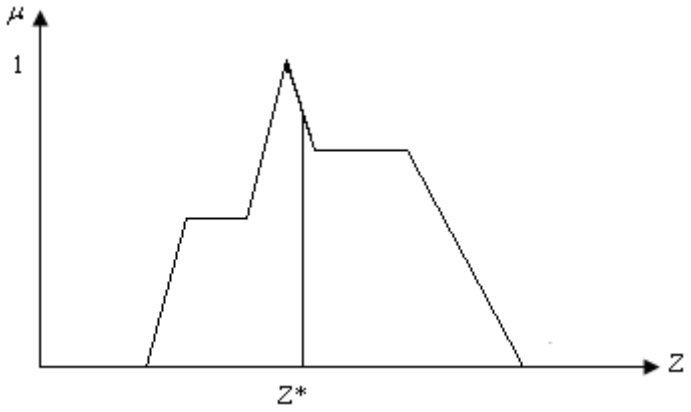
Şekil 2.26: Maksimum Üyelik Yöntemi

2.7.3.2 Ağırlık Merkezi Yöntemi

Ağırlık merkezi veya alan merkezi olarak da bilinen bu yöntem en yaygın kullanılan durulaştırma yöntemidir. Şu formülle ifade edilir (Sugeno1985).

$$Z^*(A) = \frac{\int \mu_C(z)\, z dz}{\int \mu_C(z)\, dz} \qquad [2.93]$$

Z* değerinin elde edilişi Şekil 2.27'de görülmektedir.



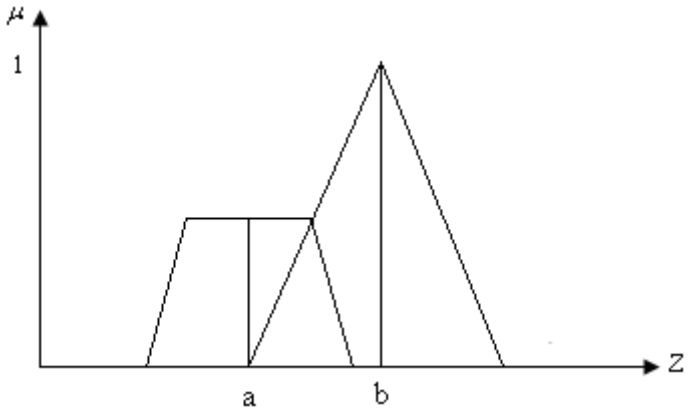
Şekil 2.27:Ağılık Merkezi Yöntemi

2.7.3.3 Ağırlık Ortalaması Yöntemi

Bu yöntem de girişlerden elde edilen bütün bulanık değerler ile üyelik değeri kullanılarak durulaştırma yapılmaktadır.

$$Z^* = \frac{\sum \mu_C(\bar{z})\,\bar{z}}{\sum \mu_C(\bar{z})} \qquad [2.94]$$

Burada $\sum$ sembolü cebirsel toplamayı ifade eder. Z* değerinin elde edilişi Şekil 2.28'de görülmektedir.



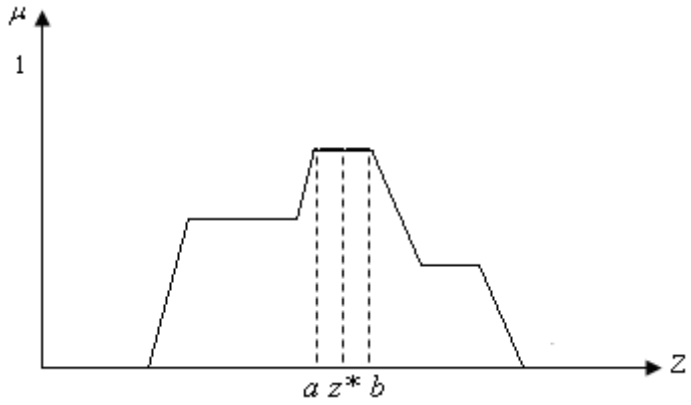
Şekil 2.28: Ağırlık Ortalaması Yöntemi

2.7.4.3 Maksimumların Ortalaması Yöntemi

Maksimum üyelik işlevi yöntemiyle ilişkilidir. Bu işlev maksimum üyelik derecesi tek bir nokta olmayıp düz olabilen sistemler içinde kullanılabilmektedir. Şu şekilde ifade edilir.

$$Z^* = \frac{a+b}{2} \qquad [2.95]$$

Z* değerinin elde edilişi Şekil 2.29'da görülmektedir.



Şekil 2.29: Maksimumların Ortalaması Yöntemi

Bulanık Mantık Kontrol'ün performansı için durulaştırma metodlarının seçimi çok önemlidir. Durulaştırma için bir çok yöntem mevcut olmasına karşın bunlardan en çok kullanılanı ağırlık merkezi yöntemidir.

2.8 Sonuç

Yukarıda açıklanan bulanık yöntemlerde, öznel bilgilerin işlenmesiyle basit ve yumuşak biçimde sistem modellemenin yolu açılmaktadır. Bulanıklık sonucunda varılan bütün harmanlaşmış çıkarımların durulaştırılması ile sayısal değerlerin elde edilmesi, yine insan tasarım, kontrol ve planlarında kullanılması için gereklidir. Şimdiye kadar, birçok klasik ve belirgin sistem modellenmesinde kullanılamayan bilgilerin EĞER O HALDE kuralları ile modellemeye dahil edilmesi sonucunda, insan zekasındaki bazı işlevlerin hesaplamalara sokulması bulanık küme, mantık ve sistem kuralları ile mümkün olmaktadır. Böylece, insanların deneyim, tecrübe, sezgi, görüş, vb yeteneklerinden kaynaklanan kişisel bilgilerin de modellemelerde işin içine nesnel bulanık kurallarla katılması mümkün olmaktadır.

Günümüzde birçok tüketici ürünlerinde ve endüstriyel uygulamalarda Bulanık Mantık Kontrol yöntemleri uygulanmaktadır. Son yıllarda bilgisayarların, özellikle kişisel bilgisayarların yaygınlaşması ile kontrol sistemlerinin analizi önemli bir derecede gelişmiştir. Bilgisayar sistemlerinin fabrika ortamında çalışmasını sağlamak için kontrol mühendisliği ile birleştirilmesi gerekmektedir. Bu, ancak güçlü ve güvenilir bir yazılımla gerçekleştirilir. Bunu sağlayacak olan yazılım kaynaklarından biri de nesne tabanlı Java programlama dilidir

Bundan sonraki bölümde; Nesne tabanlı programlamanın ne anlama geldiği, yapısı ve özellikleri ve buna bağlı olarak Java programlama dili tanıtılıp bazı özelliklerinden bahsedilecektir.

ÜÇÜNCÜ BÖLÜM

NESNE TABANLI PROGRAMLAMA

ve

JAVA

3.1 Giriş

Bundan önceki bölümde, bulanık küme, mantık ve sistem kuralları ile insan zekasındaki bazı işlevlerin birçok klasik ve belirgin sistem modellenmesinde hesaplamalara sokulmasından bahsedilmiştir. Günümüzde bilgisayarların yaygınlaşması ile kontrol sistemlerinin analizi önemli derecede gelişmiştir. Bu bölümde Bulanık Mantık Kontrol'ü bilgisayar ortamına aktarmak için gerekli olan yazılım kaynaklarından nesne tabanlı Java programlama dilinden bahsedilecektir.

Nesne tabanlı programlama (Object-Oriented Programming-OOP), bir yazılım tasarlama ve uygulama yöntemidir. Nesne tabanlı programlama dilleri, bir dil sözdizimi ve derleyicisinin yanında, tam bir geliştirme ortamı sunar. Bu geliştirme ortamı iyi tasarlanmış kullanımı kolay nesnelerden oluşan önemli bir kütüphane içerir.

Nesne tabanlı programlama hiçbir programlama dili ile ilişkilendirilmemelidir. Ama nesne tabanlı programlamayı destekleyen bir programlama dilinde nesne tabanlı teknikleri uygulamak daha kolaydır. Nesne tabanlı teknoloji, yazılım sistemlerinin geliştirilmesi sürecini iyileştirir. İşlevselliğin ve ilgili verinin ortaklığını geliştirir.

Nesne tabanlı programlama, bilgisayarlarda problem çözümüne yeni bir bakış açısıdır. Nesne tabanlı yaklaşım, bir probleme %100 çözüm aramayı hedefler. Programcılar, her zaman gerçek yaşamdaki durumlara yakın senaryolarla çalışır. Bu yöndeki ilk adım bilgisayarın yaşadığımız dünyadan nesnelerle ilişki kurmasını sağlamaktır. Hepimizin bildiği

gibi, bilgisayar sadece bir makinedir. Ona yapmasını programladığımız şeyleri yapar. Bu nedenle, bilgisayarın bizim bildiğimiz varlıkları tanımasını sağlayacak bilgiyi vermek bizim sorumluluğumuzdadır.

Bu noktada nesne-tabanlı programlama devreye girer. Nesne-tabanlı teknolojiyi kullanarak gerçek problemlerde karşılaştığımız varlıkları, bilgisayarda benzer varlıklar olarak modelleyebiliriz.

Nesne tabanlı geliştirme, gerçek sistem ile uygulama alanı arasındaki uyuşmazlığı azaltmayı hedefler.

Nesne tabanlı programlama, veriyi ve veri üzerinde uygulanacak olan işlemleri tek bir nesne olarak görür. Veriyi kendine has kimliği ve özellikleri olan bir birim veya varlık olarak görür. Aşağıda, ‘nesne’, ‘veri’ ve ‘yöntem’ ayrı ayrı incelenmektedir.

Nesne kavramı, gerçek yaşamın her noktasına ve programlama dünyasına uygulanabilir. Herhangi bir uygulama, insanın düşünce sürecini en iyi şekilde taklit etmek için, varlıklar ve nesneler cinsinden tanımlanabilir.

Üstten-alta yaklaşım, yapısal programlama olarakta bilinir. Bir programın temel işlevlerini belirleme işleminden oluşur ve daha sonra görevleri, en alt seviyeye kadar küçük birimlere böler. Bu teknik ile programlar modül hiyerarşisi olarak organize edilir. Her modülün tek bir giriş ve tek bir çıkış noktası vardır. Her modül içinde kontrol, yapı boyunca aşağı doğru akar; yapıda, daha yukarıya hiçbir dallanma olmaz.

Nesne tabanlı programlama yaklaşımı, gerçek problemlerde var olan karmaşıklığın üstesinden gelmeye çalışır. Bunu yapmak için, nesnelerin içindeki bazı bilgileri saklar. Nesne tabanlı programlama, önce veri üzerine yoğunlaşır. Daha sonra veriyi işleyen ve üzerinde çalışan fonksiyonları (yöntemleri), verinin bir parçası olarak oluşturur.

3.2 Veri Soyutlama

Bir programcı program geliştireceği zaman, program kodunu yazmaya hemen başlamaz. Önce uygulamayı inceler ve bu uygulamayı oluşturan parçaları belirler. Daha sonra her parça hakkında gerekli bilgileri belirler.

Veri soyutlama, bir varlığa ilişkin elimizdeki ugyulama ile ilgili özellikleri ve işlemleri belirleme ve gruplama sürecidir.

Soyutlamanın Avantajları:

Veri soyutlamanın faydaları şunlardır:

- Problem üzerine odaklanır
- Gerekli işlemleri ve özellikleri belirler
- Gereksiz detaylardan kurtulmaya yardım eder

Veri soyutlama gereklidir, çünkü bir nesnenin tüm özelliklerini ve işlemlerini kullanmak pratik değildir. Gerekli noktaların üzerinde odaklanmak ve onları uygulamalarda gerçekleştirmek çok önemlidir.

3.3 Sınıf

Sınıf, bir varlığı ortak özellikler ve işlemler cinsinden tanımlar, veya bir varlığın ortak özellikleri ve işlemleri, sınıf dediğimiz tek bir mantıksal birimde toplanır, veya bir sınıf, benzer varlıkların paylaşılan özellikleridir. Bir sınıf, bir varlığın kavramsal bir modelidir. Evrensel veya özel değildir.

Bir sınıf tanımlanırken, sınıfın içindeki herşeyin belirli özellikleri ve işlemleri olacağı belirtilir.

3.4 Nesne

Sınıf bir varlığın muhtemel özelliklerini ve işlemlerini belirten bir prototipidir. Sınıfın tanımlandığı varlığı kullanabilmek için, bu sınıftan bir "nesne" oluşturmalıdır. Sınıf bir kavramdır; nesne de sınfın tanımladığı asıl numunelerden biridir. Bir nesne, çoğunlukla dokunabilen, görülebilen ve hissedilebilen somut bir varlıktır. Nesne var olduğu için, özellikleri belirli değerler alabilir ve nesne için tanımlanmış işlemler üzerinde çalıştırılabilir. Aşağıdaki hususlar her nesne için belirgin bir şekilde tanımlanmıştır:

- Durum
- Davranış
- Kimlik

Her nesnenin, ne olduğunu veya ne yaptığını belirten kendi özellikleri ve nitelikleri vardır. Programlama nesnelerini ve gerçek yaşamdaki nesneleri birbirine eşleştirmek için, bir nesnenin özelliklerini ve işlemlerini bir araya getirmek gerekir.

3.4.1 Öznitelik

Bir sınıf tanımlamak için özellikleri ve işlemleri belirtilir. Özellikler, değer atandığı zaman, bir nesne tanımlarlar. Öznitelik; bir nesneyi tanımlayan bir niteliktir. Başka bir deyişle, öznitelik, bir nesne içinde, belirli bir nesneyi tanımlayan değerleri tutar.

Sınıf bir prototip olduğundan, sınıftaki özellikler değer alamaz. Bir öznitelik, sadece sınıftan bir nesne oluşturulduktan sonra değer alabilir.

3.4.2 İşlem

Bir sınıfta tanımlandığı şekliyle izin verilen işlemler, 'eylemler' olarak isimlendirilir. Eylem; bir nesneden istenebilecek bir servistir. İşlemler, bir sınıftan oluşturulmuş bir nesneden isteyebileceğimiz eylemleri belirtir.
Sınıf sadece bir prototiptir. Bu nedenle, bir işlem sadece sınıfın içinde tanımlanabilir. İşlemler, sadece nesneler tarafından ve sadece nesnelere uygulanabilir.

3.4.3 Yöntem

İşlemler, bir nesneden istenebilecek işlemleri tanımlar. Yöntemler, bir nesnenin verilerinin nasıl işleneceğini belirtir. Tanım olarak yöntem, istenilen bir eylemin nasıl yerine getirileceğini belirtir. Yöntem, bir işlemin asıl uygulaması olduğu için, bir nesneye uygulanabilir. Yöntem, bir işlem istenildiği zaman, ne yapılacağını adım adım belirten bir algoritmadır.

3.4.4 İleti

Bir eylemin yerine getirilmesini istemek için, işlemin tanımlı olduğu nesneye bir ileti gönderilir. Tanım olarak ileti, bir işlemin yerine getirilmesi için istektir. Bir nesne ileti aldığı zaman, istenilen yöntemi çalıştırır.

3.4.5 Olay

Olay, bir nesneden diğerine gönderilen uyarıdır. Örneğin, bir buton üzerine sol fare tuşu ile tıklamak, bir olayı oluşturur.

Olay, bir nesnenin herhangi bir anda başına gelebilir. Olaya cevap vermek için bir nesne, bir veya daha fazla yöntem çalıştırabilir.

3.5 Sınıf ve Nesne

Sınıf ve nesne arasında belirgin bir fark vardır. Sınıf bir varlık tanımlar. Nesne ise varlığın kendisidir. Nesne gerçektir; sınıf ise nesne için gerekli tüm özellikleri ve eylemleri tanımlayan kavramsal bir modeldir. Aynı sınıfa ait tüm nesneler aynı özelliklere ve olası eylemlere sahiptir. Sınıf, bir nesnenin prototipidir. Belirli bir grup nesne için gerekli özellikleri ve eylemleri belirtir.

3.6 Oluşturma ve Yok Etme

3.6.1 Oluşturma

Sınıf, sadece özniteliklerin ve olası işlemlerin tanımlarını sağlar. Öznitelikler ve işlemler, sadece sınıftan bir nesne oluşturulduğu zaman erişilebilir. Kısaca, bir nesne oluşturulduğu

zaman, öznitelikleri hayat bulur ve değerler alabilir. Benzer şekilde, tanımlanmış çeşitli işlemlerde kullanılabilir.

Tanım olarak oluşturma; bir nesneyi var etme sürecidir. Oluşturucu, sınıf içindeki diğer yöntemler kullanılmadan önce çalıştırılması gereken özel bir yöntemdir. Oluşturucu, özniteliklerin başlangıç değerlerini atar ve gerekli ise, bellekte yer ayırır. Her sınıfın bir oluşturucusu vardır.

3.6.2 Yok Etme

Bir nesne, artık ihtiyaç duyulmuyorsa silinmeli veya yok edilmelidir. Bir nesnenin yaşamına devam etmesine ihtiyaç duyulmuyor olmasına rağmen varlığının sürmesine izin verilirse bellek gibi kaynaklar israf edilmiş olur. Tanım olarak yokedici; bir nesneyi yok etmek için kullanılan özel bir yöntemdir. Yok etme süreci, bir nesneyi siler ve oluşturucunun ayırmış olduğu bellek alanlarını serbest bırakır. Yok etme, ayrıca, nesnenin erişilebilirliğinide kaldırır. Bir nesne yok edildikten sonra, özniteliklerine erişilemez ve üzerinde hiçbir işlem gerçekleştirilemez.

3.7 Kalıcılık

Tanım olarak kalıcılık, bir nesnenin yaşamından sonra veri tutabilme yeteneğidir.

3.8 Veri Kapsülleme

Veri soyutlama süreci, gerekli öznitelikleri ve yöntemleri belirlemede yardımcı olur. Nesneler, çoğu kez nesneyi kullananlar tarafından görülmeyen öznitelikleri ve yöntemleride kullanırlar. Tanım olarak veri kapsülleme; bir nesnenin çalışma detaylarını kullanıcısından saklama işlemidir. Kapsülleme, diğerlerinin erişebileceklerini, nesnenin içinde kullanılanlardan ayırır.

Kapsüllemenin avantajı; bir sınıf oluşturulduğu zaman, görevin yerine getirilmesi için gerekli olan tüm özelliklerin ve yöntemlerin oluşturulmasıdır. Ama, sadece diğerlerinin erişebileceği özellikler ve yöntemler erişilebilir yapılır.

3.9 Kalıtım

Kalıtım; bir sınıfa, diğer sınıflardan tanımlanmış öznitelikleri ve işlemleri kendi içinde kullanma şansı verir. Bir başka sınıftan miras alan sınıf, alt-sınıftır. Üst-sınıf, kalıtımla diğer sınıflara davranışlarını aktaran sınıftır.

Çoklu Kalıtım:

Çoklu kalıtımda alt-sınıf iki veya daha fazla sınıftan kalıtım yoluyla oluşur. Çoklu kalıtım, birden fazla sınıfın işlevselliğini kullanan bir sınıf oluşturmak için kullanılabilir. Yeni sınıfın kullanıcısına, bu yeni sınıf tüm işlevselliğini sunar. Bu nedenle, kullanıcı birden fazla nesne ile uğraşmak zorunda değildir. Kalıtımın en önemli avantajı, kodun yeniden kullanılabilmesini sağlar.

3.10 Çok Biçimlilik

Çok biçimlilik, bir işlemin farklı nesnelerde farklı biçimde davranmasını sağlar. Çok biçimlilik ile; farklı sınıflara ait nesnelere aynı işlem uygulanırsa, işlem farklı sonuçlar üretir. Aynı işlem olması, sadece aynı sayıda parametre aldığı anlamına gelir. Çok biçimlilik, nesne tabanlı sistemin en önemli özelliklerinden biridir. Çok biçimlilik, kapsüllemeyi destekler.

3.11 Nesne Tabanlı Yaklaşımın Avantajları

Nesne-tabanlı programlama, programcılar için büyük bir yaklaşım değişimi gerektirir. Bu yeni yaklaşım, programların geliştirilmesini hızlandırır ve iyi kullanılırsa bakıma, tekrar kullanılabilirliğe katkı verir.

Nesne-tabanlı yaklaşımın bazı avantajları:

- Bir problemin inceleme, tasarım ve gerçekleştirme aşamalarına, uygulama alanı terimleri ve kavramları ile yaklaşır. Bunun sonunda, uygulama ve gerçek problem arasında çok yakın bir benzerlik oluşur.

- Uygulama içinde paylaşımı destekler.

- Yeni uygulamalar geliştirileceği zaman, varolan nesnelerin yeniden kullanılmasını olanıklı kılar. Bu çok önemli avantajdır çünkü uzun vadede maliyetlerde tasarruf sağlar.

- Hataları ve bakım sorunlarını azaltır. Çünkü eski nesnelerin yeniden kullanılması mümkündür.

- Yine nesnelerin yeniden kullanılmasının bir sonucu olarak; tasarım ve geliştirme süreçlerini hızlandırır.

3.12 Java'nın Soyağacı

Java 1991'de Sun Microsystems'de James Gosling, Patric Naughton, Chris Warth, Ed Frank ve Mike Sheridan tarafından düşünülmüştür. İlk çalışan versiyonunu geliştirmek 18 aylarını almıştır. Bu dil ilk başlarda "Oak" olarak anılmış, fakat sonra 1995'te "Java" olarak değiştirilmiştir. Oak'ın ilk 1992 güzünde tamamlanmasından, 1995 baharında Java adıyla ilan edilmesine kadar geçen sürede birçok kişi dilin tasarlanması ve gelişim sürecinde bulunmuştur. Bill Joy, Arthur Van Hoff, Jonathan Payne, Frank Yellin ve Tim Lindholm orjinal prototipin olgunlaşmasında etkin rol oynayan kilit katılımcılardandır.

Sürpriz bir şekilde, Java'nın üretimi için güdü internet değildi. Aslında ilk motivasyon, mikrodalga fırın yada uzaktan kumanda gibi, elektronik tüketim eşyalarının içerisine yerleştirmek amacıyla yazılım yaratmak için, platformdan bağımsız (mimari nötr) bir dil ihtiyacıydı. Tahmin edilebileceği gibi pek çok değişik mikroişlemci olarak kullanılır. C yada C++'ın (diğer birçok dilinde) asıl problemi, bunların özel bir hedef için derlenmiş olarak tasarlanmalarıdır. Her ne kadar bir C++ programını herhangi bir tip mikroişlemci için derlemek mümkünsede bunun için o mikroişlemcinin hedeflendiği tam bir C++ derleyicisi gerekmektedir. Problem, derleyicilerin yaratılmasının zaman alıcı ve pahalı olmasıdır. Daha kolay, maliyet açısından verimli bir çözüm gerekir. Çözüm için bir denemede, Sun Microsystems'de bu konuda çalışmalarını yürüten Gosling, Naughton, Warth, Frank ve Sheridan; değişik ortamlarda değişik mikroişlemciler üzerinde çalışabilecek kodlar için kullanılabilecek, taşınabilir, platformdan-bağımsız bir dil üzerinde çalışmaya başlamışlardır. Bu çabalar en sonunda; Java'nın doğmasına yol açmıştır.

Java'nın ayrıntıları üzerinde çalışıldığı zaman civarlarında, Java'nın geleceğinde önemli bir rol oynayacak olan ikinci, sonuç olarak daha önemli bir faktör doğmaktaydı. Bu ikinci güç tabii webdi. Web, Java tamamlandığında henüz şekle girmemişti; Java, tüketici elektronik eşyaları programlamada kullanılan, yararlı ama tanınmamış bir dil olarak kalabilirdi. Ancak webin doğmasıyla Java, bilgisayar dili tasarımında ön sıraya ilerlemiştir, çünkü web de taşınır programlara ihtiyaç duymaktaydı. İnternet çeşitli, yaygın ve çok değişik bilgisayar tipleri, işletim sistemleri, mikroişlemcilerden oluşan bir evrendir. Birçok değişik platform internete bağlı olsa bile kullanıcılar hepsininde aynı programı çalıştırabiliyor olmasını isteyeceklerdir.

1993'te Java tasarım ekibi açıkça fark etti ki, kontrolörlerin içerisine gömecekleri kodları oluştururken sıkça karşılarına çıkan taşınabilirlik problemleri, internet için kod oluştururken de karşılarına çıkmaktaydı. Aslında aynı problemi ilk olarak küçük bir ölçekte çözmek için tasarlanan Java geniş bir ölçeğe, internete uyarlanırken de uygulanabilirdi. Bu farkına varma, Java'nın odağının tüketici elektronik eşyalarından Internet programcılığına dönmesine neden olmuştur. Bu yüzden, mimariden-bağımsız bir programlama diline olan ihtiyaç ilk kıvılcımını sağlarken Internet, sonuç olarak Java'nın büyük başarı elde etmesini sağlamıştır.

Daha önceden de bahsedildiği gibi Java, karakterinin büyük bir kısmını C ve C++'dan türetmiştir. Bu ise bilinçli yapılmıştır. Java tasarımcıları, C'nin tanıdık sözdizimini kullanır ve C++'ın nesne yönelimli özelliklerini tekrarlarsa, bu dilin C/C++'ın tecrübeli programcı

toplulukları için cezbedici olacağını biliyorlardı. Yüzeysel benzerliklere ek olarak Java, C ve C++'ı başarılı kılan değişik özellikleri de paylaşır. Birincisi, Java, gerçek ve çalışan programcılar tarfından tasarlandı, test edildi, iyileştirildi. Onu tasarlayanların, ihtiyaçlarına, tecrübelerine dayanan bir dildir. Böylelikle, aynı zamanda bir programcının dilidir. İkincisi, uyumlu ve mantıksal olarak tutarlı bir dildir. Üçüncüsü, Internet ortamının dayattığı kısıtlar dışında, Java programcıya tam bir kontrol verir. Eğer iyi programlamlanırsa programlarınız bunu yansıtır. Eğer kötü programlarsanız programlarınız bunu da yansıtır. Bir başka deyişle Java, profesyonel programcılar için bir dildir.

Bilgisayar dilleri iki nedenden dolayı güncellenirler: Ortamdaki değişikliklere uyum için ve programlama sanatındaki gelişmeleri gerçekleştirmek için. Java'yı dayatan ortamsal değişim, internet üzerinde dağıtımın zorladığı platformdan-bağımsız programlara olan ihtiyaçtır.

3.13 Java Appletleri ve Uygulamaları

Java iki tip program oluşturmak için kullanılabilir: uygulamalar ve appletler. Uygulama, bilgisayarda, o bilgisayarın işletim sistemiyle çalışan bir programdır. Yani Java'yla yazılmış bir uygulama C yada C++'la yazılmış bir uygulamayla aşağı yukarı benzerdir. Java'yı önemli yapan, daha çok, appletler oluşturma yeteneğidir. Applet, internet üzerinden nakledilebilecek ve Java uyumlu bir web tarayıcıda çalışabilecek bir uygulamadır. Aslında applet network(ağ) üzerinde tıpkı bir resim, ses yada video klibi gibi dinamik olarak indirilen küçük bir Java programıdır. Önemli olan fark, bu programın "akıllı" olması; sadece bir animasyon yada medya dosyası olmadığıdır. Başka bir deyişle applet kullanıcı girdisine göre tepki veren, dinamik olarak değişen bir programdır. Sürekli aynı animasyonu yada sesi oynatan bir program değildir.

3.14 Java Dilinin Özellikleri

Basit:

Java profosyonel programcıların kolayca öğrenip etkili bir şekilde kullanabilmesi için tasarlanmıştır. Nesne-yönelimli programlamanın temeli anlaşıldığı zaman Java'yı öğrenmek daha da kolaylaşır. Java C/C++'ın dil yapısını ve C++'ın nesne yönelimli özelliklerini devraldığı için C yada C++ bilen bir kişi için Java'ya geçiş çok az çaba gerektirecektir.

Nesne-Tabanlı:

Javada herşey nesnedir. Bu yüzden, uygulamadaki veri ve veri üzerinde çalışan metodlar odak noktasıdır. Java sadece yordamlar üzerinde yoğunlaşmaz. Veri ve metodlar bir nesnenin durumunu ve tavrını birlikte tanımlar. Java'da, metod terimi ile çok karşılaşılır. Bu terim, fonksiyon anlamında kullanılmıştır.

Java geçmiş yılların nesne yazılım ortamlarından ödünç almıştır. Java'daki nesne modeli, tamsayılar gibi basit tipler yüksek performanslı nesne-olmayanlar olarak kalınca, basit ve genişletmesi kolay bir modeldir.

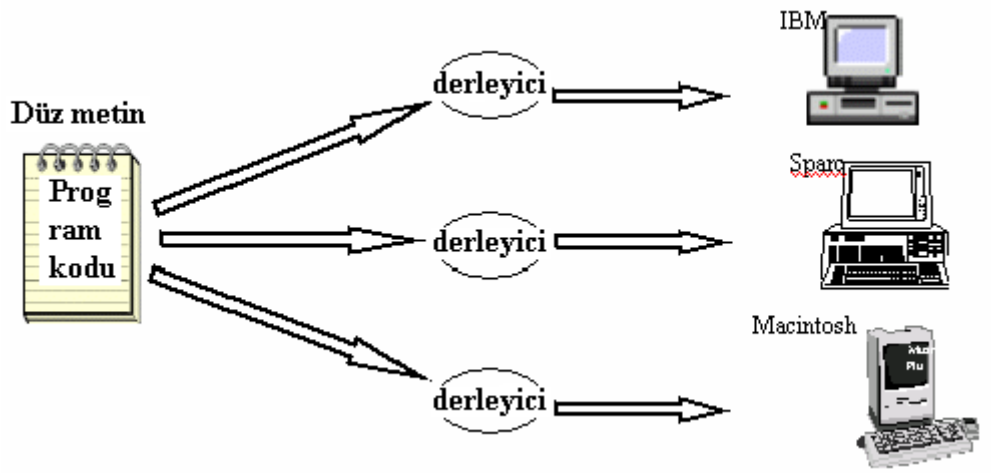
Platform-Bağımsız:

Platform-bağımsızlığı, bir programın hiçbir zorlukla karşılaşmadan, başka bir bilgisayara aktarabilme yeteneğidir. Java, kaynak düzeyinde ve ikili düzeyinde platform bağımsızdır.

Java, tür tabanlı(strongly typed) bir dildir. Bu demektir ki, her değişkenin türü belirtilmek zorundadır. Java veri türleri, tüm platformlarda aynıdır. Java, kendi temel sınıf kütüphanesine (foundation class library) sahiptir. Bu, programcının bir makineden diğerine tekrar yazılmadan aktarılabilen programlar yazabilmesini sağlar. Kısaca, kaynak düzeyindeki platform-bağımsızlık, kaynak kodunuzu bir sistemden diğerine aktarabilmenizi, derleyebilmenizi ve rahatça çalıştırabilmenizi sağlar.

İkili düzeydeki platform-bağımsızlık, derlenmiş ikili amaç dosyayı, tekrar derlemeden farklı platformlarda çalıştırabilmesini sağlar. Ama, burada, yorumlayıcı olarak çalışacak bir Java Sanal Makinesine ihtiyaç olacaktır.

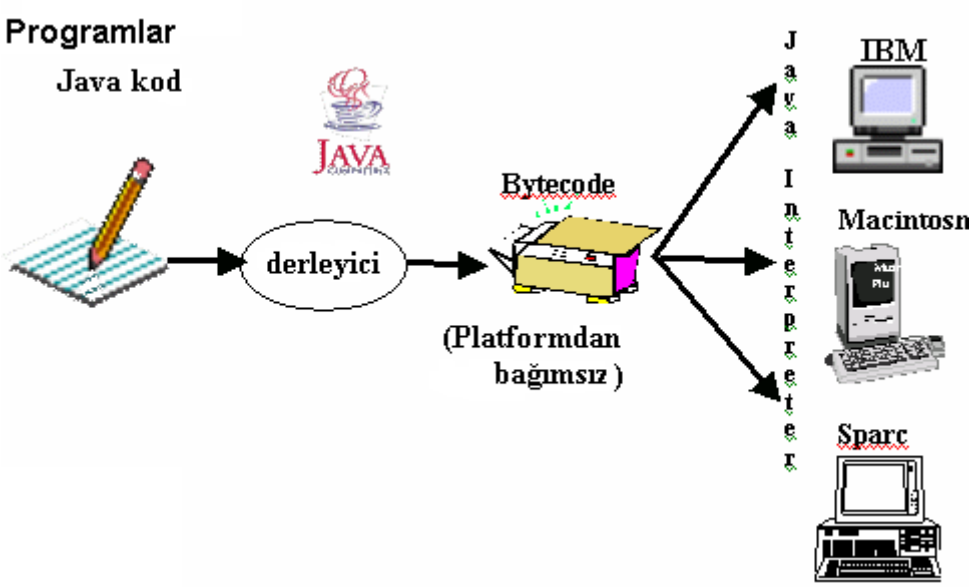
Diğer programlama dillerinde yazılmış programlar için derleyici, komutları makine koduna, yani, işlemci komutlarına çevirir. Bu komutlar, işlemciye özeldir. Bu nedenle, bu kod başka bir sistemde kullanacaksa, o sistemin derleyicisi edinilmelidir. Sonra, o makineye özel makine kodunu elde etmek için kod yeniden derlenmelidir. Şekil 3.1'de diğer programlama dillerinde yazılan programların farklı makinelerde çalıştırılmasını göstermektedir.



Şekil 3.1: Diğer Programlama Dillerinde Program Derleme

Java geliştirme ortamı iki kısımdan oluşur: Java derleyicisi ve Java yorumlayıcısı. Java derleyicisi kaynak kodu bayt koda çevirir; ki bu kod makine-bağımsızdır.baytkodları, baytlara bölünmüş Java komutlarından oluşmuş yığındır ve her işlemci tarafından anlaşılabilir.

Java yorumlayıcısı, Java Virtual Machine (JVM, Java Sanal Makinesi) veya Java Çalışma-Anı Yorumlayıcısı (Java Runtime Interpreter) olarakta bilinir. Java baytkodlarını çalıştırır. (Şekil 3.2)



Şekil 3.2: Java'da Program Derleme

Bytecode(Bayt Kodu):

Java'nın taşınabilirlik ve güvenlik problemlerini çözmesine izin veren anahtar sözcük bayt kodudur. Java derleyicisinin çıktısı yürütülebilir bir kod değil, bir bayt kodudur. Bayt kod,

Java Virtual Machine adlı, Java'nın yürütme-zamanı (run-time) sistemi tarafından yürütülmek üzere çok iyi optimize edilmiş bir dizi talimattır. Yani JVM standart haliyle bayt kodu için bir yorumlayıcıdır.

Java programını bayt koduna çevirmek, bir programı birçok değişik ortamda çalıştırabilme olanağı verir. Sebep oldukça açıktır: Sadece JVM her platformda gerçekleştirilme ihtiyacı duyar. Bir sistem için bir defa run-time paketi var oldu mu, üzerinde herhangi bir Java programı çalışabilir. Her ne kadar JVM'in ayrıntıları platformdan platforma değişsede hepsi aynı Java bayt kodunu yorumlar. Bu yüzden, bayt kodunun yorumlanması gerçek anlamda taşınabilir programlar oluşturabilmek için en kolay yoldur.

Java programının yorumlanıyor olduğu gerçeği aynı zamanda güvenli olmasına da yardımcı olur. Bir java programının yürütülmesi JVM'in kontrolü altında olduğu için, JVM o programı kapsayabilir ve sistemin dışında yan etkiler üretmesini engelleyebilir. Dolayısıyla; güvenlik, Java dilinde bulunan bazı kısıtlamalardan dolayı da artar.

Güvenlik:

Java, programın çalışması için kontrollü bir ortam sağlar. Hiçbir zaman, kodun çalıştırmak için güvenli olduğunu varsaymaz ve birkaç güvenlik kontrolü aşaması sağlar.

İlk katmanda, veri ve metodlar sınıf içinde kapsüllenir. Bunlar, sadece sınıfın sağladığı arayüzler aracılığı ile erişilebilir. Java, işaretçi aritmetiği sunmaz. Bu nedenle, belleğe doğrudan erişime izin vermez. Dizi taşmasına izin vermez; belleği sınırlar, dışarıdan okumayı önler ve çöp toplayıcı sunar. Tüm bu özellikler, güvenlik ve taşınırlık sorunlarını azaltmaya yardım eder.

İkinci katmanda, derleyici kodun güvenli olduğunu teyit eder ve kodu derlemeden önce, Java tarafından konulmuş protokolleri uygular.

Üçüncü katman yorumlayıcı tarafından sağlanan güvenliktir. Çalıştırmadan önce, tüm baytkodlar üzerinden geçer ve kurallara uyup uymadıklarını kontrol eder.

Dördüncü katman sınıfları yükleme işi ile ilgilenir. Sınıf yükleyici, sınıfı sisteme yüklemeden önce, sınıfın erişim kısıtlamalarına uyup uymadığından emin olur.

Taşınabilirlik:

İnternete bağlı birçok değişik platformlarda dinamik olarak indirelecek programların taşınabilir, yürütülebilir kodlar üretecek araçlara ihtiyaçları vardır. Yukarıda güvenliği sağlayan mekanizma aynı zamanda taşınabilirliğide sağlamaktadır.

Sağlam(Güçlü):

Java, bir tür tabanlı dildir. Bu yüzden açık metod bildirimi gerktirir. Java, kodu derleme aşamasında ve ayrıca yorumlama aşamasında kontrol eder. Böylece bazı programlama hatalarını ortadan kaldırır.

Javada işaretçiler (pointers) ve işaretçi aritmetiği (pointer arithmetic) yoktur. Dizi ve dizgelere tüm erişimleri, sınırlar içinde olup olmadıklarını teyit etmek için, çalışma zamanında denetler. Ayrıca, nesnelerin bir türden diğerine çevrilmesinide çalışma zamanında kontrol eder. Geleneksel programlama dillerinde programcı, gerekli bellek alanlarını kendisi ayırır. Programın sonunda da, programcı açık bir şekilde bu aldığı bellek alanlarını geri verir. Programcı bellek alanlarını geri vermeyi unuttuğu zaman sorunlar çıkar. Java'da, programcı bellek ayırma ile ilgilenmek zorunda değildir. Bu işlemler, kullanılmayan nesneler için Java'nın sağladığı çöp toplayıcısı (garbage collection) sayesinde otomatik olarak yapılır.
Java'nın istisna işleme (exception handling) özelliği, hatalarla uğraşma ve onlardan kurtulma işlemini kolaylaştırır.

Çok Kanallı:

Java, aynı anda pek çok işi yapabilecek olan programlar yazma imkanı veren, çok kanallı programcılığı destekler. Java'nın çok kanallılığa kullanımı kolay yaklaşımı, çok amaçlı alt sistemler yerine, programın özel davranışları üzerinde düşünebilmesine imkan sağlar.

Mimari Nötr:

Java tasarlanırken temel konulardan bir tanesi de kodun uzun solukluluğu ve taşınabilirliğidir. Bugün yazılan bir programın yarında çalışacağına dair hiçbir garantinin olmaması programcıların karşılaştığı ana problemlerden birisidir. İşletim sistemlerinin yükseltilmesi (upgrade), işlemcilerin yükseltilmesi, asıl sistem kaynaklarındaki değişimler, bir program hatası verecek biçimde etki edebilirler. Java tasarımcıları Java dilinde ve Java sanal makinesinde bu sorunu gidermek için "bir kere yaz, her yerde, her zaman, sonsuza kadar çalıştır" amacını gerçekleştirmişlerdir.

Dağıtık:

Java internetin dağıtık ortamı içinde tasarlanmıştır, çünkü internet TCP/IP protokollerini kullanır.

Yorumlama Ve Yüksek Performans:

Java, farklı platformlara, işletim sistemlerine ve grafik arayüzlerine aktarılabilen uygulamalar geliştirmek için kullanılabilir. Java, ağ uygulamalarını destekleyecek şekilde tasarlanmıştır. Bu yüzden, Java, Internet gibi farklı platformları olan bir ortamda geliştirme aracı olarak kullanılmaktadır.

Dinamik:

Java, değişen ortamlara ayak uydurabilmesi için tasarlanmış dinamik bir dildir. Java programları çalışma, zamanında nesneleri teyit edebilmek ve onlara erişebilmek için çok miktarda çalışma-zamanı bilgisi tutar. Bu, kodu dinamik bir şekilde bağlamayı mümkün kılar.

3.15 Java Programlarının Türleri

Javada yazılan programlardan, aşağıdaki gibi farklı kategoriler oluşur:

3.15.1 Apletler

Apletler, Internette çalışmaları için yazılmış özel Java programlarıdır. Netscape veya Internet Explorer gibi Java destekli tarayıcılar aracılığıyla çalışırlar. Herhangi bir Java geliştirme-düzenleme aracı kullanarak aplet oluşturulur. Aplet, bir HTML dosyasına veya Web sayfasına gömülmelidir. HTML dosyası veya Web sayfası bir tarayıcıda görüntülendiği zaman, aplet yüklenir ve çalıştırılır.

Apletler, diğer ugyulamalrın yanında, oyunlar ve (oldukça popüler göl ve kar efekti gibi) görsel efektler geliştirmek için kullanılır.

3.15.2 Komut Satırı Uygulamaları

Bu Java programları komut satırından çalıştırılır. Grafik arayüz tabanlı hiçbir pencere göstermezler. Böyle programlar, konsol-tabanlı çıktı verirler.

3.15.3 GUI (Grafik Kullanıcı Arayüzü)Uygulamaları

Bu Java programları bağımsız çalışır ve GUI-tabanlı bir ekran aracılığı ile kullanıcının girdisini alır.

3.15.4 Servletler

Java, web-tabanlı n-katmanlı uygulama geliştirmek için uygundur. Aplet, GUI tabanlı, istemci tarafında herhangi bir tarayıcıda çalışabilen bir programdır. Web-tabanlı bir uygulamada, istemci sunucuya bir istek gönderir. Sunucu, isteği işler ve istemciye cevabını gönderir. Sunucu-tarafı Java API (API-Application Programming Interface-Uygulama Programlama Arayüzü), sunucu-tarafı program işlemleri ve istemcinin isteğinin cevapları ile ilgilenir. Bu sunucu-tarafı API, standart Java API içinde bulunan yetenekleri artırır. Bunlar, Java servletleri olarak da bilinir. Ayrıca, sunucu-tarafı apletleri olarak da isimlendirilirler. HTML formu işlenmesi, servletlerin en basit uygulamalarından bir tanesidir. Veritabanları ile çalışabilirler ve sunucu-tarafı işlemleri için kullanılabilirler. Çoğunlukla, web sunucu aracılığı ile çalıştırılırlar.

3.15.5 Veri Tabanı Uygulamalrı

Bu programlar, veritabanı bağlantısı için JDBC (Java DataBase Connectivity-Java Veritabanı Bağlantısı) API kullanır. Aplet ve uygulama olabilirler. Ama, apletler veritabanı ile çalışırken güvenlik sorunları yaratabilirler.

3.16 Java Sanal Makinesi

Sanal makine, hayali bir bilgisayar düşüncesi üzerine kurulu yazılım kavramıdır. Mantıksal komutlar kümesi vardır. Bu komutlar bu bilgisayarın işlemlerini tanımlar. Ufak bir işletim sistemi gibi düşünülebilir. Alt kademede bulunan işletim sistemi, donanım platformu ve derlenmiş kod için soyutlama katmanını oluşturur.

Derleyici, kaynak kodu, herhangi bir işlemciye özel olmayan hayali bir bilgisayarın komutları ile yazılmış koda dönüştürür. Yorumlayıcı, bu komut dizisini anlayan uygulamadır. Yorumlayıcı, bu komutları hedeflenen işleyici makine komutlarına dönüştürür. JVM, kodun çalışmasına yardım eden çalışma-zamanı sistemini şunlar ile oluşturur:

- .class dosyalarının yüklenmesi
- Bellek yönetimi
- Çöp toplama

Donanım platformlarının uyumsuzluğu sanal makinenin yığıt makinesi(stack machine) kavramını kullanmasına neden olur ve burada sanal makine şunları tutar:

- Bir metodun durumlarını tutan çerçeveler
- Baytkodların işlenenleri (operands)
- Metodların argümanları
- Yerel değişkenler

JVM kodu çalıştırırken ‘program sayacı’ isimli bir yazmaç (register) kullanır. Bu yazmaç, o anda çalışan komutları gösterir. Gerektiğinde, komutlar program sayacını değiştirerek çalışmanın yönünü değiştirebilir. Yoksa çalışmanın akışı sıralıdır; yani, bir komuttan hemen sonraki komuta geçilir.

Java’da bir diğer ilginç kavram ‘tam zamanında’ derleyici kullanımıdır. Netscape 4.0 ve üzeri Internet Explorer 4.0 ve üzeri gibi tarayıcılar, Java kodunun çalışmasını hızlandıran JIT(Just-in-time-tam zamanında) derleyiclerini içerir. JIT’ın ana amacı, baytkod komut kümesini, özel bir işlemci için tasarlanmış makine komutlarına çevirmektir. Bu komutlar saklanır ve ilgili metoda çağrı geldiği zaman kullanılır.

3.17 Çöp Toplama ve Bellek Yönetimi

C, C++ veya Pascal'da, programcılar ilkel bir şekilde yığın olarak bellek blokları alırlar ve kullandıktan sonra geri verirler. Yığın, tüm iş parçacıkları (thread) tarafından paylaşılan büyük bellek kümesidir.

Bir yığını yönetebilmek için, bellek şu listeler ile izlenir:

- Boş bloklar listesi
- Ayrılmış bloklar listesi

Boş bloklar listesi, bir bellek bloğu için istek geldiği zaman taranır. Kullanılan ayırma mekanizması, 'ilk-uyan blok'tur; ki bu mekanizmada, isteğe bağlı olarak, ilk en küçük bellek bloğu ayrılır. Yığından değişik boyutlarda küçük bellek kümeleri alma ve geri verme stratejisi, yığının parçalanmasını en alt tutmaya yardımcı olur.

Birleştirme, yığın parçalanmasını en alt düzeyde tutan diğer bir mekanizmadır. Burada, bellek yığına geri verildiği zaman, boş listesine yeni bir blok eklenir. Bu yeni blok, yığında daha önceden varolan boş bloktan hemen önce veya hemen sonra gelebilir. Eğer gelirse, iki blok, daha büyük boş bir blok oluşturmak için birleştirilir. Bu süreç, birleştirme olarak adlandırılır.

Yığın yöneticisi, bellek isteği elinde olandan daha büyük bir bellek bloğu içinse veya boş listesindekinden daha fazla ise, daha fazla bellek oluşturulmalıdır. Bu teknik, bitiştirme olarak bilinir. Burada, alınmış bellek yığınının sonuna taşınır ve tüm boş bellek blokları birleştirilir. Bu, tek ve büyük bir bellek bloğu oluşturur.

Java Sanal Makinesi, durağan ve dinamik bellek ayırma için iki ayrı yığın kullanır. Bir yığın, sınıf tanımlarının, sabit havuzunun ve metod tablolarının tutulduğu, çöp toplamaya gitmeyen yığındır.

İkinci yığın, gerektiğinde ve gerektiği gibi zıt yönlerde genişletilebilen iki bölümden oluşur. Bir bölüm nesnelerin örneklemelerini ve diğeri de bu örneklemelerin adreslerini tutar (handles).

'handle' iki işaretçiden oluşan bir yapıdır. Bir tanesi nesnenin metodlar tablosunu, diğeride o nesnenin örneklemesini gösterir. Bu yerleşim, bitiştirme işleminden sonra işaretcileri güncellerken hangi değişkenin hangi nesneye işaret ettiğinin izini tutma ihtiyacını ortadan kaldırır. Sadece işaretçilerin değerini güncellememiz yeterli olur.

Çöp toplama algoritması, dinamik yığındaki nesnelere uygulanır. Bellek bloğu için istek geldiği zaman, yığın yöneticisi önce boş listesini kontrol eder. yığın yöneticisi, boş bellek bloğu bulamazsa, sistemin yeteri kadar boş olduğu bir zamanında, çöp toplayıcı çağrılır. Uygulamalar çok etkileşimli ise ve sistemin boş olduğu zaman çok azsa, uygulama çöp toplayıcıyı açık bir şekilde çağırabilir.

Çöp toplayıcı bir nesnenin bir örneklemesini toplamadan önce, 'finalize' metodunu çağırır. 'finalize' metodu dosyalar ve akımlar (stream) gibi açık olan dış kaynakları, temizlemeye yarar. Bununla, çöp toplama yordamı, ilgilenmeyecektir. Çöp toplama metodunu (System.gc()) açık bir şekilde çağırmış olsanız bile, hemen çalışmayabilir. Sadece, daha sonra çalışmak için kurulur ve bu değiştiriliemez. Bu, çöp toplama iş parçacıklarının düşük öncelikle çalışmasından ve sık sık çalışmalarının kesilmesinden dolayıdır. Nesneniz, uygulamanız bitmeden önce atılamamış olabilir. Bu nedenle, 'finalize' metodunun kulanılışı tartışılır.

3.18 .class Dosyasının Doğrulanması

Güvenliği sağlamak için doğrulama işlemi yüklenen tüm .class dosyalarına uygulanır. Sınıf Yükleyici, sisteme ait olmayan tüm .class dosyalarını, protokollere uyup uymadıklarından emin olmak için kontrol eder. Sınıf Yükleyici, .class dosyasının kötü bir niyetle yazılmış olup

olmadığını kontrol eder; ki, kötü niyetli bir dosya belleğe, istemci dosya sistemine, ağa veya işletim sistemine zarar verebilir. Doğrulayıcı, sınıfın genel bütünlüğünü kontrol eder.

.class dosyası 3 mantıksal bölümden oluşur:

- Oluşturulmuş baytkodları
- Sınıf metodları, arayüzleri, derleme sırasında sabit havuzundan toplanan değerler gibi sınıf hakkındaki bilgiler
- Sınıfın özellikleri ve nitelikleri

Sonra, .class dosyası bağımsız bir şekilde, her alt-bölümün aşağıdaki tablolar formundaki bilgilerine bakar:

- Öznitelikleri ve nitelikleri içeren alan tablosu
- Özellikleri ve nitelikleri ile metod tablosu
- Arayüz tablosu ve elamanları ile sabit havuzu

.class dosyasının bu doğrulanma süreci 4 aşamada yapılır:

- Birinci aşama, yüklenen .class dosyasının yapısal ve sözdizimsel bütünlüğünü kontrol eden, sözdizimi kontrol aşamasıdır.
- İkinci aşama, sınıfın kuralları ihlal etmediğinden emin olmak için, .class dosyasının mantıksal tutarlılığını kontrol eder.
- Üçüncü aşama, tüm detayların uygun şekilde başlatıldığı baytkod doğrulayıcısıdır. Metodlar, doğru sayıda argüman ile çağrılır. Dizilerin, dizgilerin ve ifadelerin (expression) taşma yapıp yapmadığı kontrol edilir.
- Doğrulamanın son aşaması çalışma zamanında yapılır. Bu aşama, ilk üç aşamada yapılmayan kontrolleri yapar. Örneğin, çalışma sırasında diğer sınıflara (bir alan veya metod formunda) başvuran komutlar için bağlantı (linkage) kontrol edilir. Erişim için izinler kontrol edilir. Eğer herşey yolundaysa, sınıf örneklendirilir.

3.19 Java Geliştirme Takımı

Sun Microsystems Firması, Java dilini Java Geliştirme Takımı (Java Development Kit-JDK) isimli bir ürün olarak piyasaya sürmüştür.

Java dilinin önemli üç sürümü:

- Java 1.0-1995'te başlayan ilk sürüm.
- Java 1.1-önceki versiyon üzerine birçok geliştirme yapılan 1997 sürümü.
- Java 2 – birçok geliştirme aracı sunan son sürüm

JDK, müşterilerin Java apletlerini ve Java çekirdeklerini (JavaBeans), tarayıcının varsayılan çalışma-zamanı ortamı yerine Sun firmasının JRE ortamında çalıştırabilmeleri için Java Uyumlu-eki (plug-in) içerir. JDK'da aşağıdaki araçlar vardır:

Java Derleyicisi, 'Javac'

Javac komutu, .java uzantılı kaynak kodlarını derler. .java dosyası, baytkod olarakta isimlendirilen .class dosyasına dönüştürülür. Daha sonra, yorumlayıcı bu baytkodu çalıştırabilir.

Sözdizimi:

javac[seçenekler]kaynak dosya.java

Java Yorumlayıcısı, 'Java'

Java komutu, baytkodunda yazılmış .class dosylarını yorumlar ve çalıştırır. Java komutu, apletleri ve servletleri çalıştırmaz.

Sözdizimi:

java[seçenekler]sınıfismi

Java Tersine Çevirici, 'Javap'

Javap komutu, derlenmiş Java baytkodunu tersine çevirir. Kodu tersine çevirdikten sonra, sınıfın metodları ve üye değişkenleri hakkında bilgi verir.

Sözdizimi:

javap[seçenekler]sınıfismi

Belgeleme Aracı, 'Javadoc'

Bu araç, bir Java kaynak dosyasındaki /*...*/ türünden açıklamalarda bulunan biçim eklerini(tag) kullanaran bir HTML dosyası oluşturur. Bu HTML dosyaları, bir programdaki sınıf ve metodlar hakkında bilgi tutar. Bunlar hakındaki belgeleme, herhangi bir web tarayıcı ile erişilebilir.

Sözdizimi:

Javadoc[seçenekler]kaynakdosyaismi

Java Hata Ayıklayıcı, 'Jdb'

Bu, Java ortamı için bir hata ayıklama aracıdır. Tamamen komut satırından yönetilir. Genellikle sözdizimsel ve mantıksal hataları bulmak için kullanılır.

Sözdizimi:

*Jdb[seçenekler]kaynakdosyası //*Eğer kod yerel makinede ise

Veya

*Jdb-makine-şifre[seçenekler]kaynakdosyası //*Eğer kod başka bir makinede ise

Applet Görüntüleyici, 'Appletviewer'

Appletviewer, apletleri görüntülemek ve test etmek için kullanılır. Appleti içinde bulunduran HTML sayfasını argüman olarak alır.

Sözdizimi:

appletviewer[seçenekler]url

3.20 Java Çekirdek API

En sık kullanılan paketlerin bazıları:

Java.lang

Java.lang paketi, Java dilinin özündeki sınıflardan oluşur. Bu sınıflar karekterler, tamsayılar gibi temel veri türlerini de içerir. Ayrıca, istisnalarla ve hatalarla ilgilenen sınıfları da içerir. Standart giriş, standart çıkış ve hata çıkışı akımlarını sunan sınıflarda bu pakette tanımlanmıştır. Diğer önemli sınıflar arasında String ve StringBuffer sınıfları sayılabilir.

Java.applet

Java.applet paketi, en küçük pakettir. Tek bir sınıftan oluşur; Applet sınıfı.Applet sınıfı, bir web sayfasına gömülecek veya Java Applet Görüntüleyici ile görüntülenecek her apletin ata (parent) sınıfı olmalıdır.

Java.awt

Bu paket, Soyut Pencere Araçtakımı (Abstract Window Toolkit-AWT) olarak da bilinir. GUI-tabanlı apletler ve uygulamalar oluşturmak için kaynaklar sunar. AWT paketindeki bazı sınıflar; Button, GridBagLayout ve Graphics'dir.

Java.io

Bu paket, Java dili için standart giriş/çıkış kütüphanesi sunar. Veri akımlarını değişik şekillerde oluşturmanızı ve yönetmenizi sağlar.

Java.util

Bu paket, birkaç faydalı yardımcı sınıf sağlar. Yardımcı sınıflar, C ve C++'daki veri yapıları gibidir. Bu sınıflar, sadece farklı veri türlerinin yönetimi için kullanılır. Bu paketteki sınıflar arasında, Date, Hashtable, Stack, Vector ve StringTokenizer sayılabilir.

Java.net

Bu paket, diğer bilgisayarlar ile iletişim yeteneğini barındırır. Soketler yaratabilir, soketlere veya URL'lere bağlanabilir.

Java.awt.event

Bu paket, olay dinleyicilerini çağırmada ve olayları istediğiniz şekilde kullanmada faydalıdır. Dinleyici, bir olay gerçekleştiği zaman uyarılan nesnedir. Bu olaylar arasında, fare tıklaması, klavyede bir tuşa basılması, bir butona basılması sayılabilir. Programcılar, dinleyicileri kullanarak olayları işleyebilirler.

Java.rmi

Bu paket, uzak metodlar çağırmak için kullanılır. Uzak bir makinede nesneler oluşturmanızı ve onları bilgisayarınızda kullanabilmenizi sağlar.

Java.security

Bu paket, Java programlarınızda şifreleme yapmak için gerekli araçları sağlar. İstemci ve sunucu arasında güvenli veri alış-verişi yapmanızı mümkün kılar.

Java.sql

Bu paket, JDBC'yi, yani Java Veri Tabanı Bağlantısı'nı içerir. JDBC, Oracle ve Microsoft SQL Server gibi ilişkisel veri tabanlarına erişebilmenizi sağlar.

3.21 Java2'deki Yeni Özellikler

Java'nın eski sürümleri, ağ üzerinde büyük ticari uygulamalardan veya dağılmış sistemlerden daha çok, küçük ölçekli web uygulamaları geliştirmek için daha uygundu.
Şu anki sürümün bazı özellikleri:

Swing

'gör ve hisset' tasarımı ile ileri düzeyde GUI oluşturmak için, yeni sınıflar ve arayüzler kümesi

Sürükle ve Bırak

Bilgiyi, etkileşimli bir şekilde farklı uygulamalar arasında ve bir programın arayüzünün bir parçasından bir diğerine aktarma yeteneğidir.

Java 2D API

İleri düzey 2D (iki boyutlu) grafik ve görüntüleme için sınıflar kümesi.

Java Ses

Java'nın ses özelliklerine tamamen yeni eklemeler.

RMI

Uzaktan Metod Kullanma (Remote Method Invocation), uygulamaların uzak sitelerdeki nesnelerin metodlarını kullanabilmelerini ve onlarla iletişim kurabilmelerini sağlar.

3.22 Sonuç

Bu bölümde Bulanık Mantık Kontrol'ü bilgisayar ortamına aktarmak için yazılım kaynaklarından biri olan nesne tabanlı Java programlama dilinden bahsedilmiştir. Burada, Java'nın temelinin nesne tabanlı programlama olduğu gösterilmiştir.Ayrıca, taşınabilirlik ve güvenilirlik problemlerine Java'nın çözümü, Java derleyecisinin çalıştırılamayan çıktı (bytecode) oluşturduğundan, bu kodları çözümleyen ve her tür makinede çalışabilmesini sağlayan Java Sanal Makinesinden bahsedilmiştir.

Bundan sonraki bölümde; kontrol edilecek kamyon yükleme-boşaltma sisteminin bulanık olarak modellenmesinden bahsedilecektir. Elde edilen bulanık model Java programlama dilinde kodlanarak simulasyon arayüz kontrol'ü gerçekleştirilecektir.

DÖRDÜNCÜ BÖLÜM

BULANIK MANTIK KONTROL İLE KAMYON YÜKLEME-BOŞALTMA TASARIMI ve JAVA ORTAMINDA UYGULAMASI

4.1.Giriş

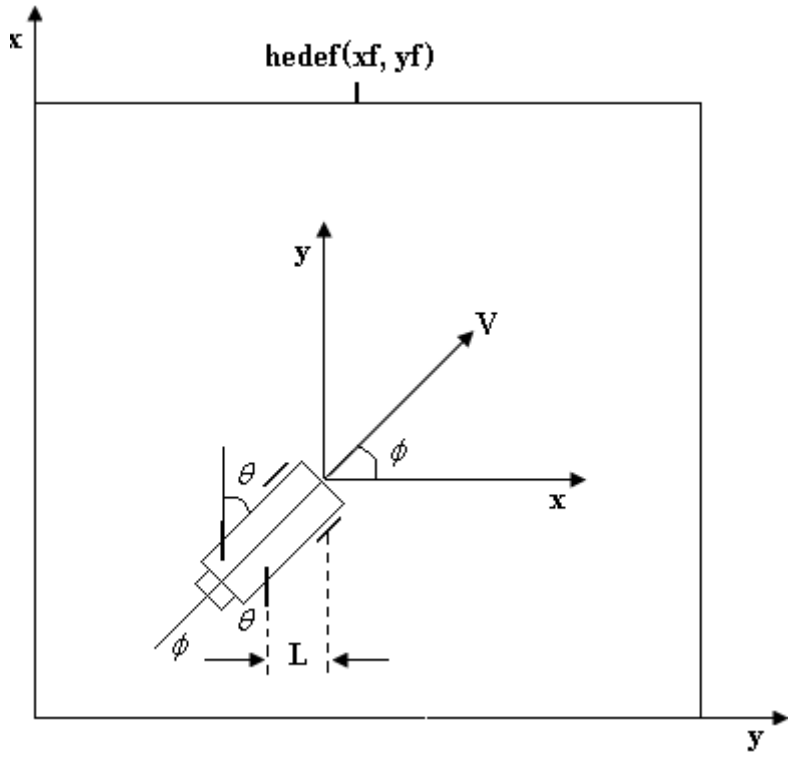
Bundan önceki bölümlerde; Endüstriyel bir süreç kontrolünde; sistemin güvenlik ve kararlılığının sağlanması, kolay ve anlaşılır olması, tamir edilebilir ve değiştirilebilir olması, sistem performansını istenen seviyeye çıkarılması, yatırım ve işletme açısından ucuz olması gibi kriterler için önerilen bulanık mantık sistem ve kontrol'ün yanında bu istenenleri en iyi şekilde bilgisayar ortamına aktarmak için gerekli yazılım kaynaklarından biri olan Java'dan da bahsedilmiştir. Bu koşulların gerçekleştirilmesi için kontrol edilecek sistemin yapısının ve dinamik özelliklerinin çok iyi bilinip matematiksel modellenmesi gerekir.

Bu çalışmada, matematiksel modelinin oluşturulması zor ve karmaşık olan kamyon yükleme-boşaltma sisteminin Bulanık Mantık Kontrol aracılığıyla Java ortamında gerçekleştirilmesi üzerinde durulmuştur. Holonomik olmayan, geriye doğru hareket eden kamyonun doğal ve gerçeğe uygun parabolik yörüngelerle belirlenmiş yollar üzerinden, hedef noktaya optimum şekilde park edebilmesi sağlanmıştır.

İncelemekte olduğumuz kontrol sistemi non-lineer'dir ve hareket denklemleri holonomik değildir. Kamyon kontrol sistemlerinde iki karmaşık problem vardır: yörünge planlaması ve hedef takibi. Herhangi bir ilave gereksinim veya kısıtlama ve herhangi bir optimallik kriteri olmadan, bu tür takip sistemleri her zaman kontrol edilebilmektedir (Laumond, 1993). Burada, öncelikle kamyon hareket alanı içerisinde bir noktaya getirilir, daha sonra kamyonun yönü ayarlanır ve son olarak kamyon harekete geçirilir. Bu nedenle, bu tür kamyon sürüş sistemlerinde kontrol edilebilirlik seviyesi, bazı ilave ve zorlaştırılmış hareket kriterlerine bağlıdır.

4.2 Sistemin Tanıtılması

İlk olarak simulasyon için gerekli olan kamyon hareket sisteminin yaklaşık matematiksel bir modeli tanımlanmıştır (Şekil 4.1).



Şekil 4.1: Kamyon ve Hareket Alanı

Kamyonun tekerleklerinde hiçbir kaymanın olmadığını farzederek, şeklin basit geometrisinden hareketle aşağıdaki dinamik denklemler elde edilir:

$$\dot{x}(t) = V\cos\phi(t)\cos\theta(t) \qquad [4.1]$$

$$\dot{y}(t) = V\sin\phi(t)\cos\theta(t) \qquad [4.2]$$

$$\dot{\phi}(t) = \frac{V}{L}\sin\theta(t) \qquad [4.3]$$

Burada,

ϕ, kamyonun ana ekseni ile, *x* ekseni arasındaki açı;

x, kamyonun arka kısmının orta noktasının yatay koordinatı;

y, kamyonun arka kısmının orta noktasının dikey koordinatı;

V, sadece geriye doğru hareket için geçerli olan, kamyonun ön tekerleklerinin hareket hızı(burada V=2m/s olarak sabit kabul edilmiştir);

L, kamyonun uzunluğu(Ön ve arka tekerleklerin merkezi baz alınarak L=4m olarak kabul edilmiştir);

θ, ön tekerlerle, kamyonun ana ekseni arasındaki açı olarak tanımlanan teker dönüş açısı (burada her iki tekerin açısını aynı olarak kabul edildi, fakat gerçekte bu açılar farklı ise θ açısını iki farklı açının ortalamasını alarak belirleriz), ayrıca kamyon saat yönünün tersine doğru dönüyorsa θ>0, saat yönünde dönüyorsa θ<0 olarak kabul edilmiştir.

θ'nın her bir adımdaki küçük değişimi için yukarıda tanımlanan denklemler yaklaşık olarak aşağıdaki gibi olur:

$$x[(k+1)\Delta t] = x(k\Delta t) + V\Delta t\cos[\phi(k\Delta t)]\cos[\theta(k\Delta t)] \qquad [4.4]$$

$$y[(k+1)\Delta t] = y(k\Delta t) + V\Delta t\sin[\phi(k\Delta t)]\cos[\theta(k\Delta t)] \qquad [4.5]$$

$$\phi[(k+1)\Delta t] = \phi(k\Delta t)\frac{V}{L}\Delta t\sin[\theta(k\Delta t)] \qquad [4.6]$$

Burada amaç kamyon hedef noktaya gidene kadar (*x*, *y*) pozisyon çiftini (x_f, y_f) noktasına, θ açısını ise 90°'ye getirmektir.. Burada sadece geri dönüş incelenecektir. Hareket alanı[0,100]x[0,100] boyutuna sahip ve hedef (50, 100) noktasıdır.

4.3 Sistemin Bulanıklaştırılması

Bulanıklaştırma işleminde öncelikle giriş ve çıkış değişkenleri belirlenir. Giriş değişkenleri kamyonun açısı ϕ ve *x* eksenindeki pozisyonu gösteren *x* değişkenidir. Çıkış değişkeni ise ön tekerleklerin yaptığı θ açısıdır. Değişkenlerin değişim aralığı ise şu şekildedir:

$0 \le x \le 100$

$-90 \le \phi \le 270$

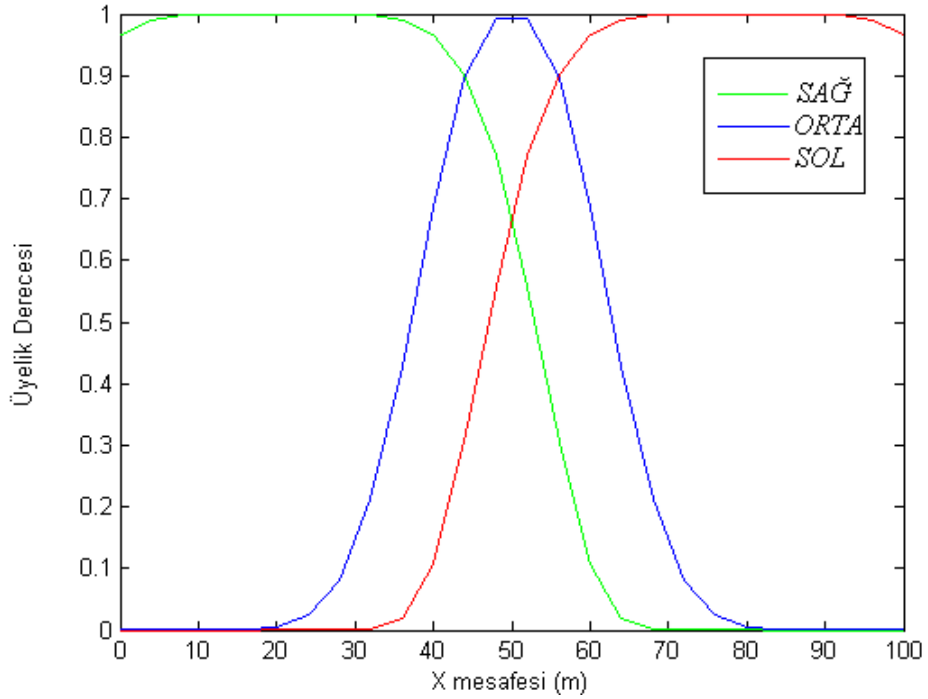
$-30 \le \theta \le 30$

Sonra giriş ve çıkış değişkenlerinin bulanık küme değerleri belirlenir. Bulanık küme, sayısal ifadeleri sözle ifade edilen terimler haline getirmek için kullanılır. Aşağıda ϕ kamyon açısı, *x* pozisyonu ve tekerlek açısı θ için belirlenen kümeler verilmiştir.

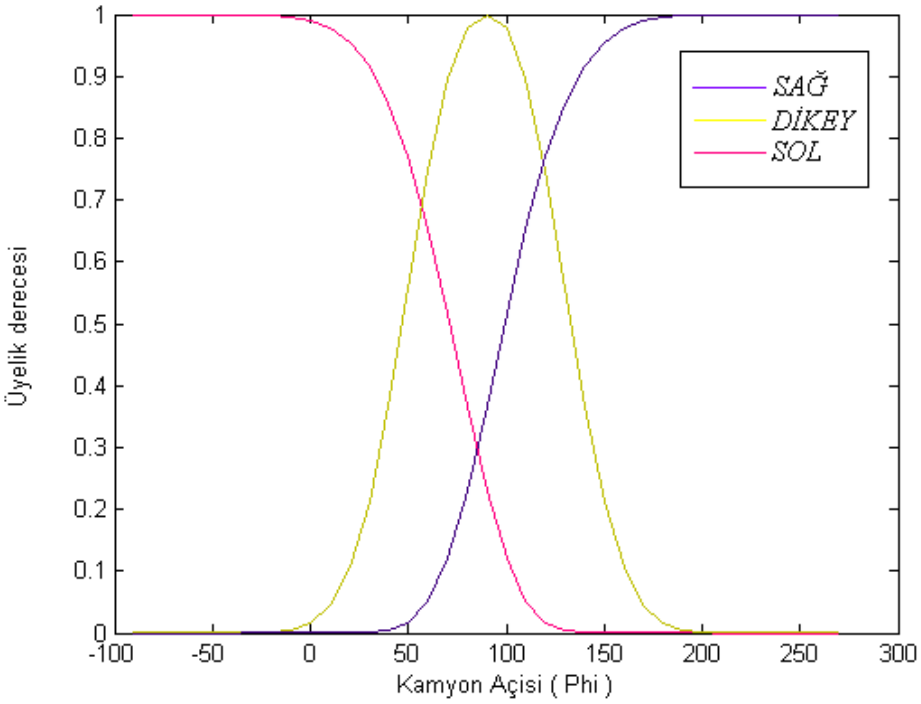
Çizelge 4.1: Bulanık Küme Değerleri

Açı(ϕ)	*x*-pozisyonu	tekerlek açısı(θ)
SAĞ:sağ	SOL:sol	NB:negatif büyük
D:dikey	O:orta	NO:negatif orta
SOL:sol	SAĞ:sağ	NK:negatif küçük
		S:sıfır
		PK:pozitif küçük
		PO:pozitif orta
		PB:pozitif büyük

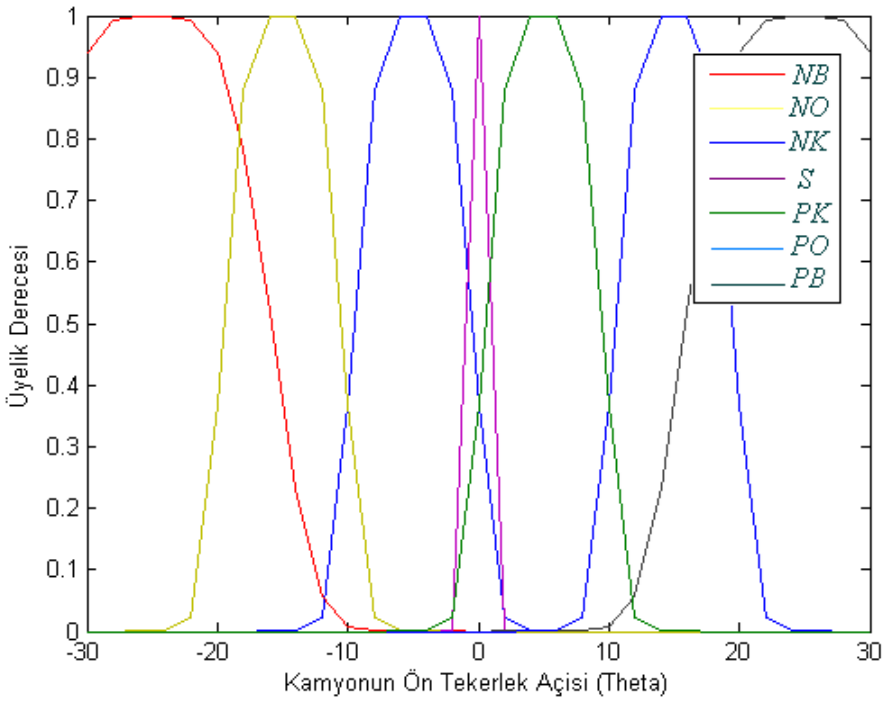
Burada verilen bulanık alt kümeler değişkenlerin üyelik derecelerini de içermektedir. Bulanık üyelik işlevi $\mu_{\tilde{A}} : Z \to [0,1]$ şeklinde gösterilir ve Z evrensel kümesinin herhangi bir elemanı olan *z* değerine 0 ve 1 aralığında gerçek bir değer atar. $\mu_{\tilde{A}}(z)$ sayısı *z* verisinin derecesini gösterir ve bulanık $\tilde{A}$ kümesinin elemanıdır. Bu eşitlik $\tilde{A}$ kümesindeki *z* elemanının bulanık değerini tanımlar. Bulanık küme üyelik işlevleri değişik şekillerde seçilebilir. En çok kullanılanlar ikinci bölümde de belirtiğimiz gibi üçgen, yamuk ve çan şekilleridir. Bu çalışmada üyelik fonksiyonları çan eğrisi şeklinde seçilmiştir. Şekil 4.2, 4.3 ve 4.4'de bulanık alt kümelerin üyelik grafik işlevleri görülmektedir.



Şekil 4.2: Giriş X mesafesinin üyelik fonksiyonlarının gösterimi



Şekil 4.3: Giriş ϕ kamyon açısının üyelik fonksiyonlarının gösterimi



Şekil 4.4: Çıkış θ tekerlek açısının üyelik fonksiyonlarının gösterimi

Bu işlevlere sistemin işleyişi sırasında x = 35, θ = 150^o olarak giriş yapıldığında öncelikle bu değerlerin üye oldukları kümeler ve üyelik dereceleri tespit edilir. x=20 için sol bulanık kümesine 0.95 derecesinde üyedir. θ=150 için sağ bulanık kümesine 0.8 derecesinde üyedir. Bu yöntemle sayısal girişlere sözlü ifadeler atanarak bulanıklaştırma gerçekleştirilir.

4.4 Sistemin Bulanık Çıkarımı

Bulanık çıkarımda da bulanıklaştırılan değerler üzerinde bulanık mantık yürütülerek (fuzzy reasoning) insan beyninin düşünme şekli taklit edilir. Bulanık kurallar üzerine bulanık mantık uygulanarak elde edilen ifadelere bulanık çıkarım denir. Bulanık Mantık Kontrol'ün kalbi bulanık çıkarım kısmıdır. Burada bilgi tabanı (knowledge base) ve karar verme mantığı (decision making logic) kullanılmaktadır. Veri tabanı (database) ve kural tabanı (rule base) bilgi tabanını oluşturur.

Veritabanı: Bulanık kümeler kullanarak giriş ve çıkış değişkenlerinin tanımlanmasını içerir. Çıkarım mekanizması kural tabanında kullanılan kümelerin üyelik işlevlerini bu bölümden alır.

Kural tabanı ise bulanık şart cümlelerinin tamamını içerir. Kontrol amaçlarına uygun dilsel kontrol kuralları burada bulunur ve çıkarım mekanizmasına buradan verilir.

Örneğin;

EĞER $a=x$ ve $b=y$ ise O HALDE $c=z$' dir.

Kuralların Oluşturulması:

Bir Bulanık Mantık Kontrol'ün gerçekleştirilişinde denetlenecek sistemin bir matemetiksel modelinden daha çok o sistemi çalıştıran operatörün sistem davranışı konusunda sahip olduğu bilgiler daha önemlidir. Tasarım sırasında genellikle bu tür bilgilerden yararlanılır. Böyle bir yaklaşım uzun yıllar boyunca kazanılan deneyimlerin kontrolör içerisine kolaylıkla yerleştirilmesini sağlar. Kural tabanının kurulması için kullanılabilecek yaklaşımlar şunlardır:

- Bir uzmanın bilgi ve/veya deneyimlerine dayanır.
- Sürecin bir bulanık modelinin kullanılmasına dayanır.
- Operatörün süreç üzerinde yaptığı işlemlere dayanır.
- Öğrenen algoritmalar kullanılır.

Sistemde kullanılan Bulanık Mantık Kontrol tasarımında, sadece kamyonun o andaki pozisyonu ve yönü bağımsız olarak kabul edilmiştir. Bu yaklaşımla Mamdani bulanık içerme sisteminde 9, Larsen bulanık içerme sisteminde ise 35 bulanık EĞER-O HALDE kuralları kontrol için yeterli görülmüştür. Böylece kontrolörün karmaşıklığı artmaz ama kontrol performansı önemli bir derecede artacaktır. (Chen ve Zhang, 1997).

Çizelge 4.2'de Mamdani bulanık içermesi için kullanılan 9 adet bulanık kural tanımlanmıştır. Örnek olarak sol üst köşede 1. kural aşağıdaki bulanık ilişkiyi temsil eder.

EĞER x=SOL ve θ=SAĞ ise
O HALDE ϕ=PB

Çizelge 4.2: Kural Tablosu

	(x)SOL	(x)ORTA	(x)SAĞ
(ϕ)SAĞ	(θ)PB	(θ)NO	(θ)PK
(ϕ)DİKEY	(θ)PO	(θ)S	(θ)PO
(ϕ)SOL	(θ)NK	(θ)NO	(θ)NB

4.5 Sistemin Durulaştırılması

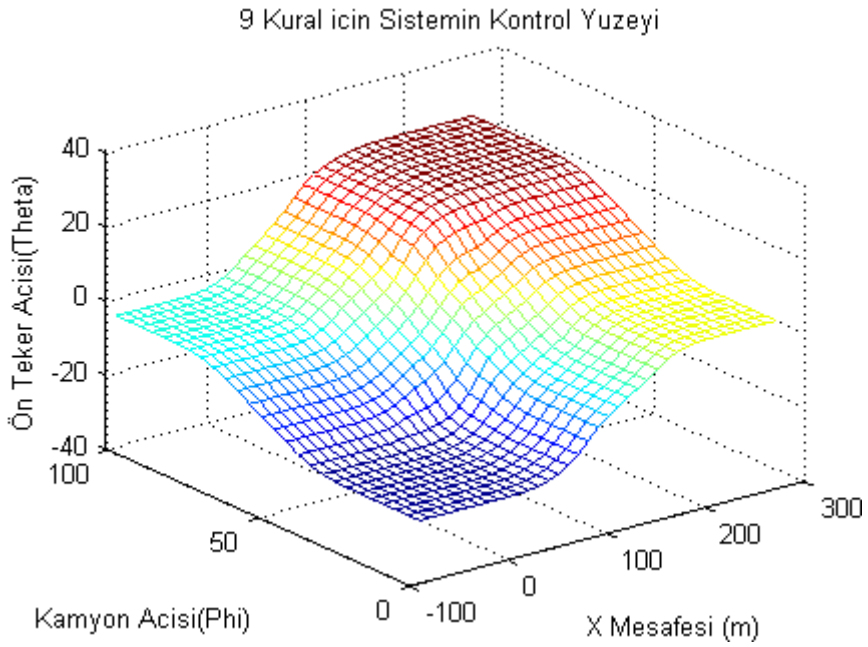
Bulanık çıkarım mekanizmasının çıkışı, çıkış evrensel kümesinde bulanık bir kümedir. Bunun için bulanık olmayan bir değere çevrilmesi gerekir. Bu çevirme işlemine durulaştıma denir.

Önce kullanılan her kural için üyelik değerlerinden oluşan her bir bulanık çıkış kümesi, çıkış evrensel kümesinde tespit edilir. Daha sonra bu kümeler tarafından oluşturulan mantıksal birleşim kümesi üzerinde durulama yöntemlerinden birisi kullanılır ve tek çıkış değeri bulunarak durulama işlemi yapılmış olur. Elde edilen değer Bulanık Mantık Kontrol'ün sisteme uygulayacağı çıkış değeridir.

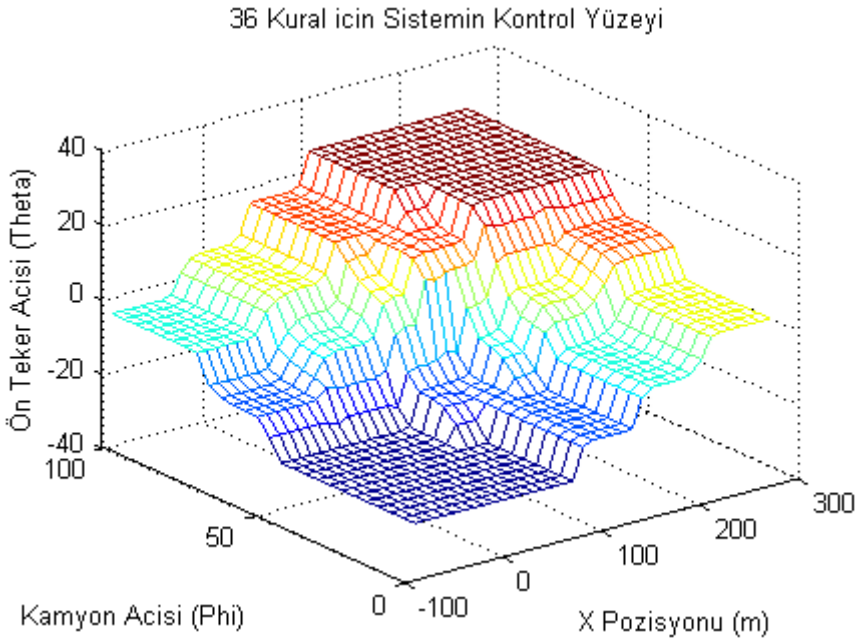
Burada durulaştırma yöntemlerinden en yaygın kullanılan yöntem olan ağırlık merkezi yöntemi kullanılmıştır. Buna göre;

$Z^*(A) = \frac{\int \mu_C(z)\, z dz}{\int \mu_C(z)\, dz}$ formülünden (denklem 2.83) yararlanarak çıkış bulanık kümesinin sayısal değere dönüştürülmesi gerçekleştirilmiş olur.

Aşağıda tasarlamış olduğumuz bu sistem için, Mamdani ve Larsen bulanık içermelerinin kontrol yüzeyi görülmektedir. (Şekil 4.5 ve 4.6)



Şekil 4.5: Mamdani Bulanık İçermesi için Kamyon Hareket Sisteminin Kontrol Yüzeyi



Şekil 4.6: Larsen Bulanık İçermesi için Kamyon Hareket Sisteminin Kontrol Yüzeyi

4.6 Sisteminin Java Ortamında Uygulaması

Matematiksel modelinin oluşturulması zor ve karmaşık olan kamyon yükleme-boşaltma uygulmasının bulanık modellenmesi gerçekleştirilmiştir. Elde edilen bulanık sistemin

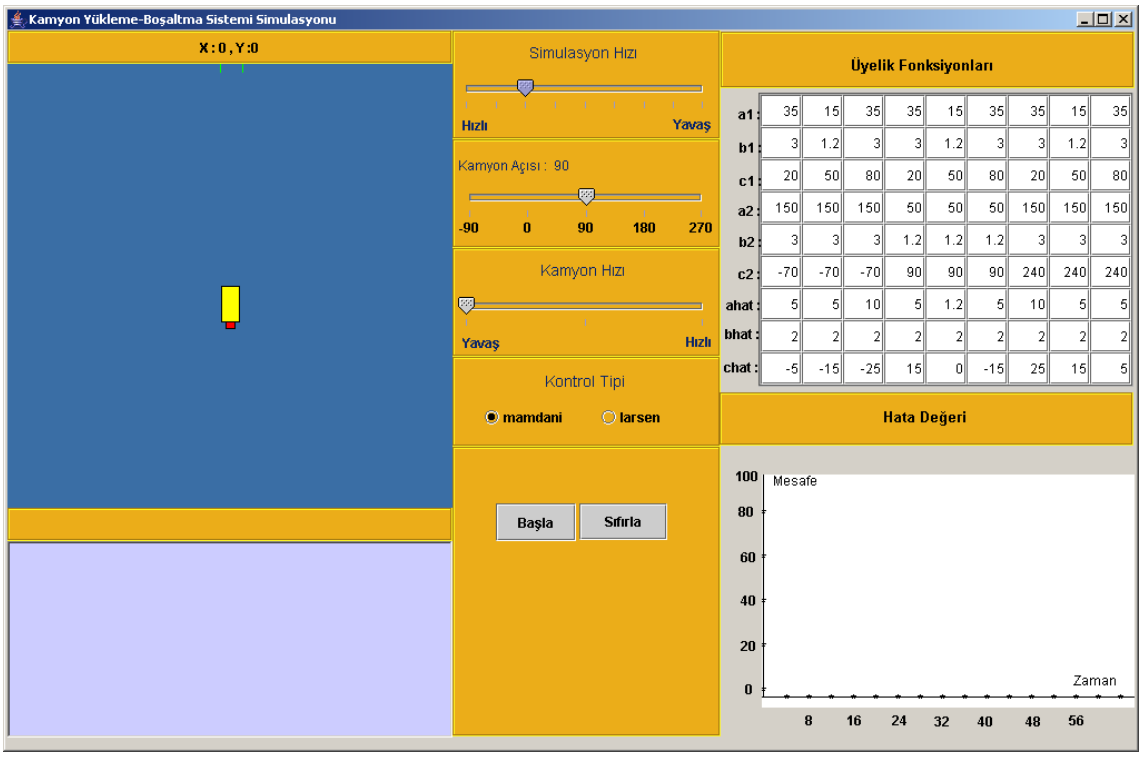
bilgisayar ortamına aktarılıp kontrol edilmesi için önceki bölümlerde de bahsedildiği gibi güçlü ve güvenilir bir yazılım kaynağına ihtiyaç vardır. Üstün özelliklerinden dolayı (Bölüm 3), simulasyon için nesne tabanlı Java2 programlama dili seçilmiştir.

Bunun için gerekli temel 10 sınıf kullanılmıştır. Bu sınıflar;

- Bulanıklaştırma sınıfı
- Durulaştırma sınıfı
- Kısmi üyelik sınıfı
- Kural tabanı sınıfı
- Kamyon tanım sınıfı
- Üyelik fonksiyonu sınfı
- Mamdani bulanık içerim sınıfı
- Larsen bulanık içerim sınıfı
- İyileştirici sınıfı
- Pencere sınıfı

Bu sınıfların dışında simulasyon için gerekli diğer sınıflar için Java'nın kütüphanesinde var olan sınıflardan yararlanılmıştır. Ek A'da bu sınıfların UML diyagramları da verilmiştir.

Bu programda amaç, bir kamyonun, bulunduğu konumdan yükleme platformuna ulaşmak için geriye doğru hareket ederek park etmesinin, Java ortamında Bulanık Mantık Kontrol ile gerçeklenmesini göstermektedir. Programda kullanıcıyla etkileşimi sağlamak amacıyla Şekil 4.7'de gösterildiği gibi bir arayüz uygulaması tasarlanmıştır.



Şekil 4.7: Sistemin Java Kullanıcı Arayüzü

Şekilde, kamyonun göründüğü sol taraf Java GUI kullanarak yapılmış hareket alanıdır. Kamyonun arka tarafı (gövdesi) sarı renkle ve ön tarafıda (sürücü bölümü) kırmızı renkle gösterilmiştir. Kamyon, bu hareket alanının herhangi bir yerine farenin sol tuşuna tıklanmasıyla kolay bir şekilde pozisyon alabilir. Bu alanının en üst kısmında, farenin hareket alanı içerisinde o anda bulunduğu yerin koordinatları (0'dan 100'e) gösterilir. Bu alanının alt tarafında, kamyonla ilgili çeşitli durum mesajları da görülür. Yükleme platformu, hareket alanının en üst kısmının orta noktasında (50-100) iki küçük düşey çizgi ile gösterilmiştir.

Bu simülasyonun amacı, kamyonun arka kısmının yükleme platformuna mümkün olan en yakın şekilde yanaştırılmasıdır. Kamyon, yukarıda değinildiği gibi, farenin, hareket alanında herhangi bir yere tıklanması ile pozisyon alabilir. Kamyonun yönlendirilmesi hareket alanının sağ tarfında bulunan sürgü vasıtasıyla (-90 derece ile +270 derece arasında) bir değere ayarlanabilir. Varsayılan kamyon açısı 90^o olarak ayarlanmıştır. Bu durumda kamyonun arka kısmı, yükleme platformu ile karşı karşıyadır. Kamyon açısının -90^0 olduğu durumda kamyonun arka kısmı aşağı doğru, 0^0 olduğu durumda ise sağa doğru yönlenir.

Kamyonun harekete geçmesi için alanın sağ kısmında bulunan *başla* butonuna tıklanır. Böylece kamyon yükleme platformuna doğru geri geri hareket edecektir. Kamyon yükleme platformunun bulunduğu üst sınırın dışına çıkacak olursa simülasyon durur. Kamyonun arka kısmı yükleme platformundaki iki çizginin orta noktasına en yakın şekilde yerleştiğinde uygulama başarı ile sonuçlanmış olur.

Uygulama, aşağıda anlatılan, hareket alanının sağında bulunan çeşitli butonlar ve sürgülerle kontrol edilebilir.

Başla Butonu: Yukarıda da belirtildiği gibi uygulama başlar ve kamyon yükleme platformuna doğru harekete geçer.
Sıfırla Butonu: Bu butona tıklanınca, kamyon en son girilen başlangıç durumuna geri gelir.
Simülasyon Hızı: Simülasyon hızı 8 farklı şekilde ayarlanıp, kamyonun hedef noktaya varış süresi değiştirilebilir.
Kamyon Hızı: Kamyon hızı üç farklı şekilde ayarlanıp, kamyonun hedef noktaya varış süresi değiştirilebilir.
Kamyon Açısı: Yukarıda açıklandığı gibi kamyonun hareket alanındaki açısı ayarlanır. Başlangıç açısı 90^o dir ve bu durumda kamyonun arka kısmı yükleme platformuna bakar. Burada dikkat çeken husus kamyonun farklı açılarda ve konumlarda tanımlanan kurallar çerçevesindeki davranışıdır.

Başla ve sıfırla butonlarının üzerinde bulunan radyo butonları vasıtasıyla istenilen kontrol algoritması seçilebilir. Varsayılan değer olarak Mamdani tip kontrol algoritması seçilmiştir.

Arayüzün en sağındaki tablo Mamdani tipi kontrol algoritması üyelik fonksiyonlarının giriş ve çıkış değerleridir. Program çalıştırılmaya başladığı anda varsayılan olarak tablo içinde yazılan değerler işlem görür. Tablo içerisindeki değerler fare tıklanarak değiştirildikten sonra başla butonuna tıklanırsa o anda girilen değerler işlem görür.

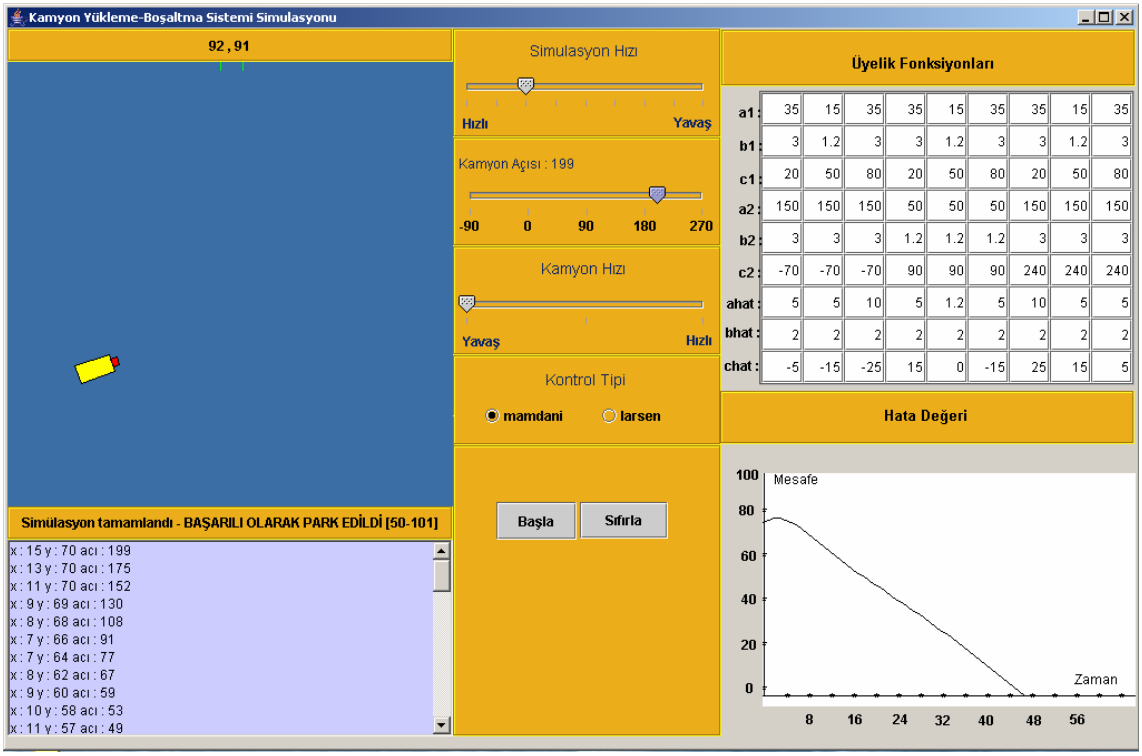
Tablonun altında bulunan grafik; kamyonun arka kısmının bulunduğu noktanın her bir iterasyon için hedef noktaya olan uzaklığının zamana bağlı olarak değişimini gösterir. Böylece sistemin hata değerini yakından görme olanağı sağlanmıştır.

Hareket alanının alt kısmında bulunan kutu; kamyonun arka kısmının bulunduğu noktanın her bir iterasyon için, X ve Y koordinat çiftlerini ve kamyonun açı değerini gösterir.

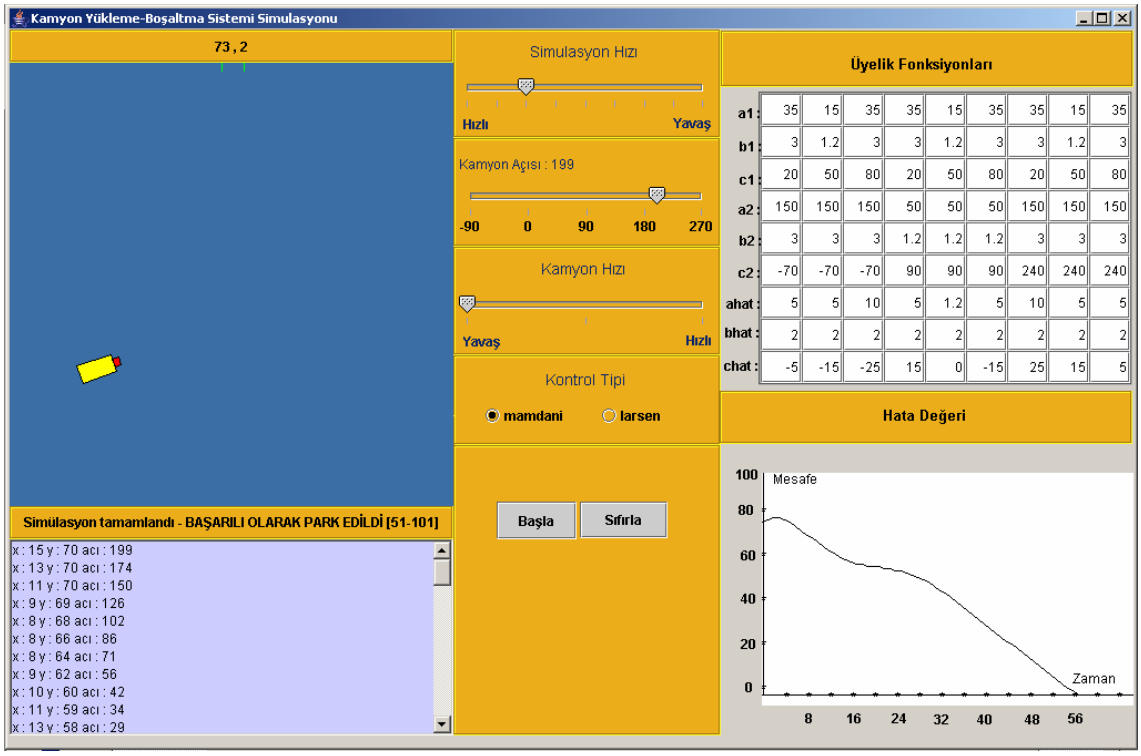
4.7 Simülasyonda Farklı Bulanık İçermelerin Karşılaştırılması

Gerçekleştirilen simulasyon programında pratik hayatta çok kullanılan 2 tip kontrol içermesi kullanılmıştır. Bunlar Mamdani ve Larsen kontrol içermeleridir. Yukarıda da değinildiği gibi Mamdani için 9, Larsen için 35 kural tanımlanmıştır. Ayrıca Mamdani kontrol içermesinin üyelik fonksiyonları değerleri kullanıcı arayüzünde gösterilmiştir. Böylece kullanıcıya Mamdani bulanık içermesinin üyelik fonksiyonlarını istediği şekilde ayarlama imkanı verilmiştir. Larsen bulanık içermesinin üyelik fonksiyonları değerleri program içinde sabit olarak ayarlanmıştır. Kullanıcı bu üyelik fonksiyonlarını değiştiremez. Bunun nedeni üyelik fonksiyonlarını tanımlayan değerlerin gösteriminin program arayüzüne sığmayacak kadar fazla olmasıdır.

Simülasyonla, Mamdani ve Larsen tip bulanık içermelerin aynı nokta ve aynı açıda karşılaştırılması Şekil 4.8(a) ve 4.8(b)'de, ayrıca, bu iki bulanık içerme ile farklı noktaların aynı üyelik fonksiyonları ile karşılaştırılması Şekil 4.9(a) ve 4.9(b) ve son olarakta aynı noktalarda farklı üyelik fonksiyonları ile karşılaştırılması Şekil 4.10(a) ve 4.10(b) gerçekleştirilerek, istenen optimum üyelik fonksiyonu gösterilmeye çalışılmıştır.

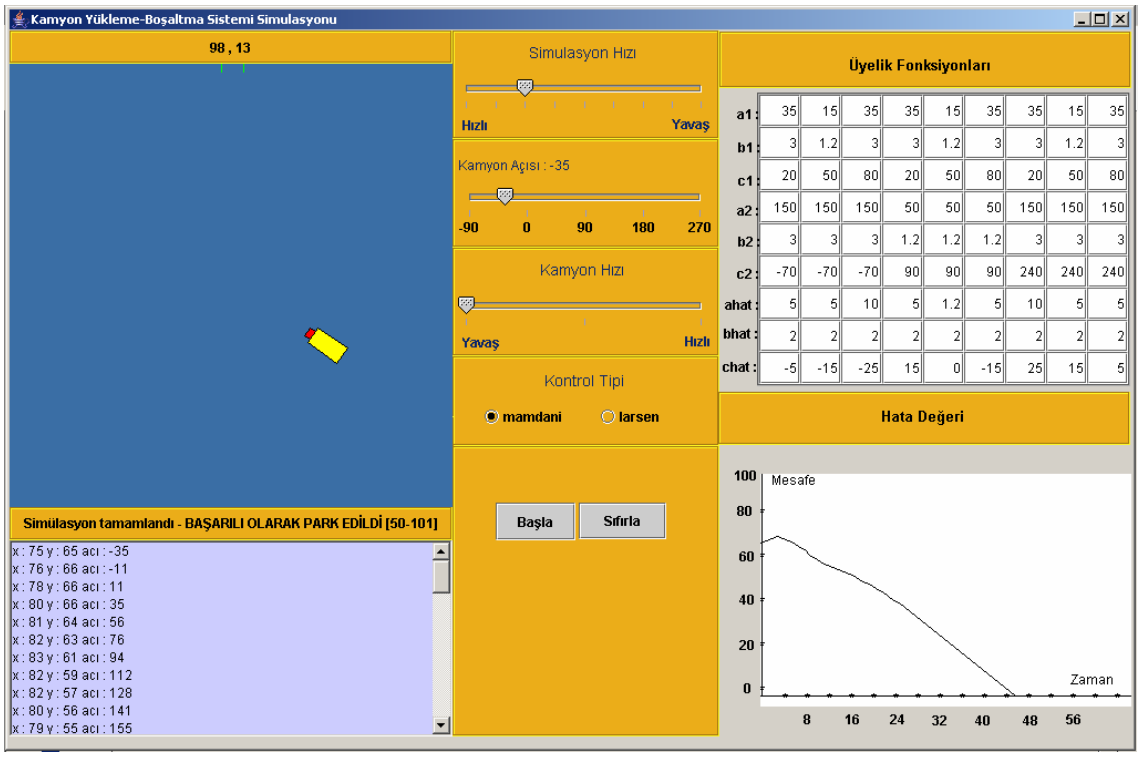


a) Mamdani Bulanık İçermesi (X=15, Y=70, ϕ=199º)

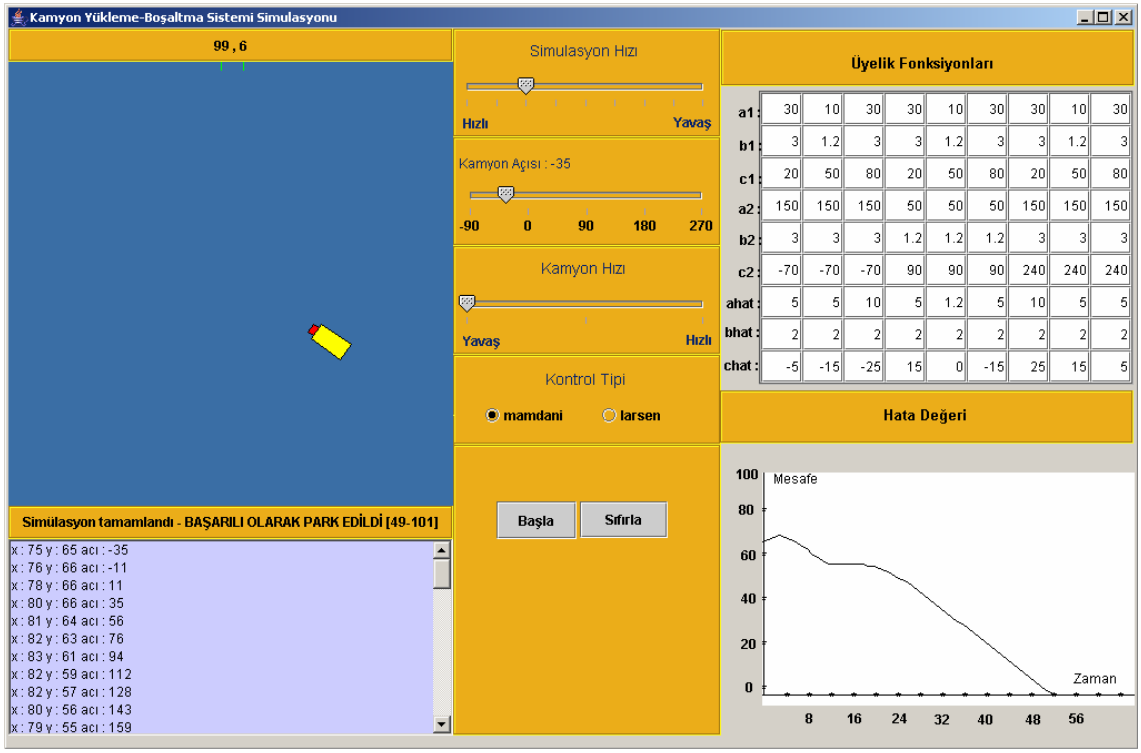


b) Larsen Bulanık İçermesi (X=15, Y=70, ϕ=199º)

Şekil 4.8: Mamdani ve Larsen Bulanık İçermelerinin Aynı Nokta ve Açıda Karşılaştırılması

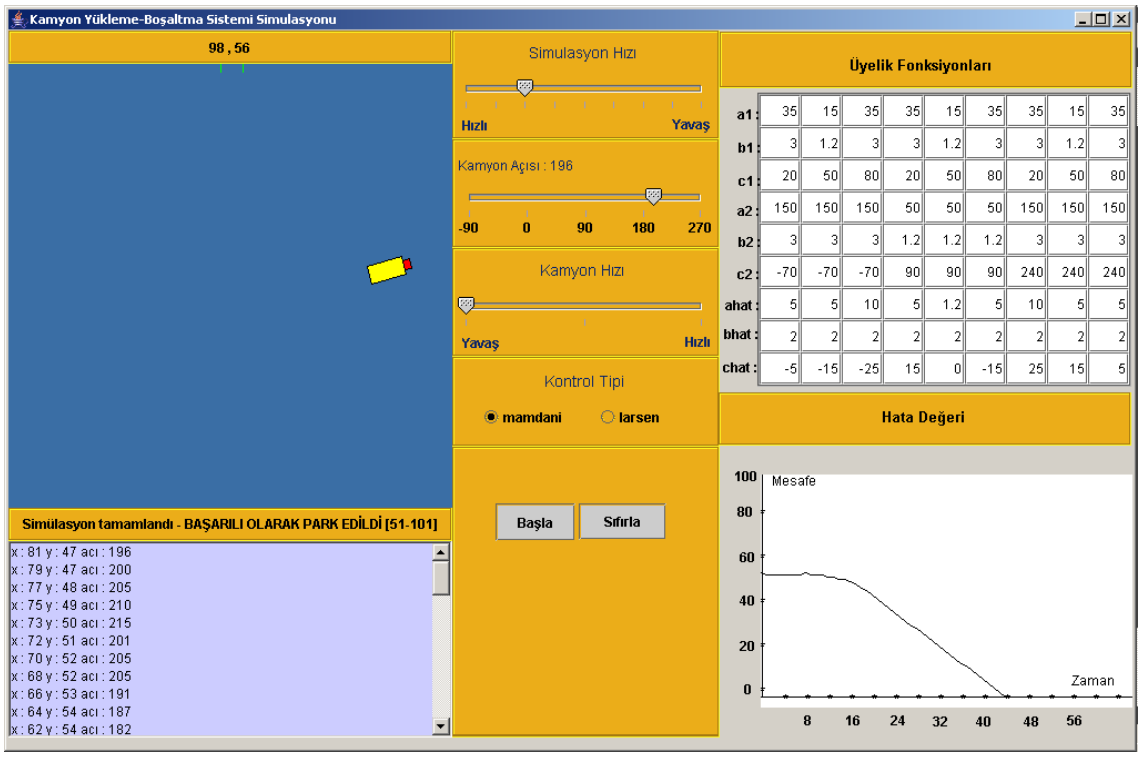


a) Mamdani Bulanık İçermesi (X=75, Y=65, ϕ =-35º)

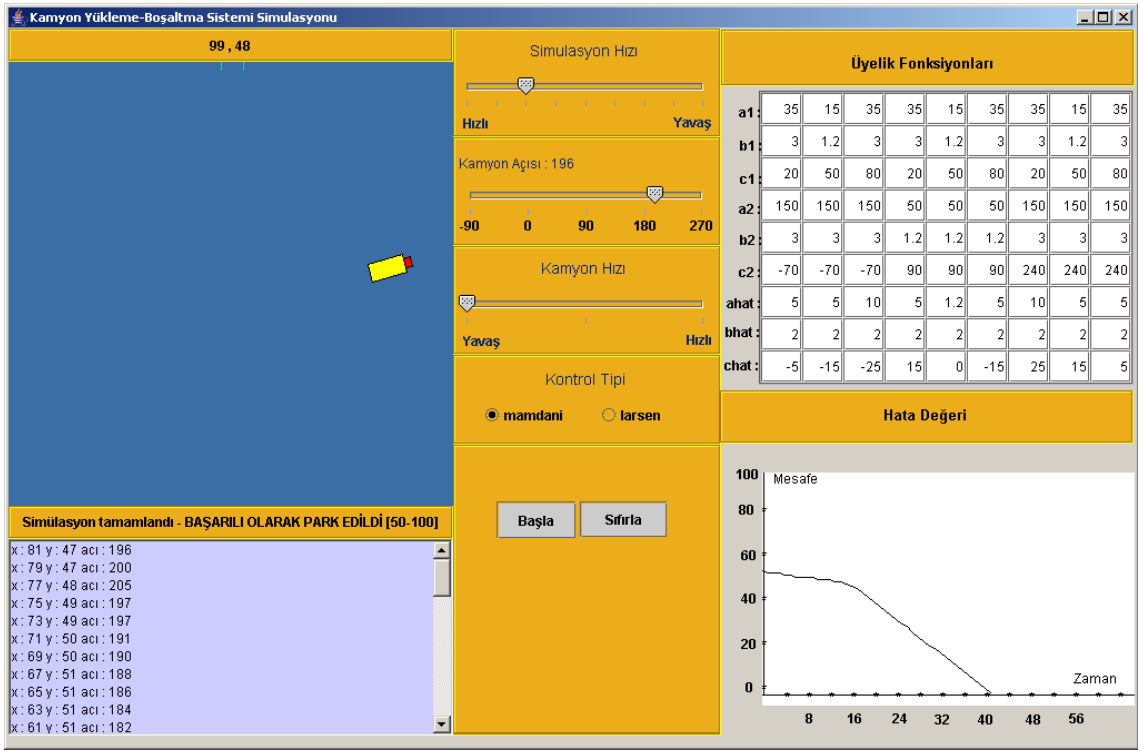


b) Mamdani Bulanık İçermesi (X=75, Y=65, ϕ =-35º)

Şekil 4.9:Mamdani Bulanık İçermesi Kullanılarak Aynı Noktada Farklı Üyelik Fonksiyonları ile Karşılaştırılması



a) Larsen Bulanık İçermesi (X=81, Y=47, ϕ =196^o)



b) Larsen Bulanık İçermesi (X=81, Y=47, ϕ =196^o)

Şekil 4.10: Larsen Bulanık İçermesi Kullanılarak Aynı Noktada Farklı Üyelik Fonksiyonları ile Karşılaştırılması

Yukarıdaki grafiklere bakarak şu yorumlar çıkarılabilir;

Mamdani ve Larsen bulanık içermelerinin aynı nokta ve açıda karşılaştırılmasında (X=15, Y=70, ϕ=199^o) hata değeri grafiğine bakarak, Mamdani bulanık içermesinin performansının daha yüksek olduğu görülür.

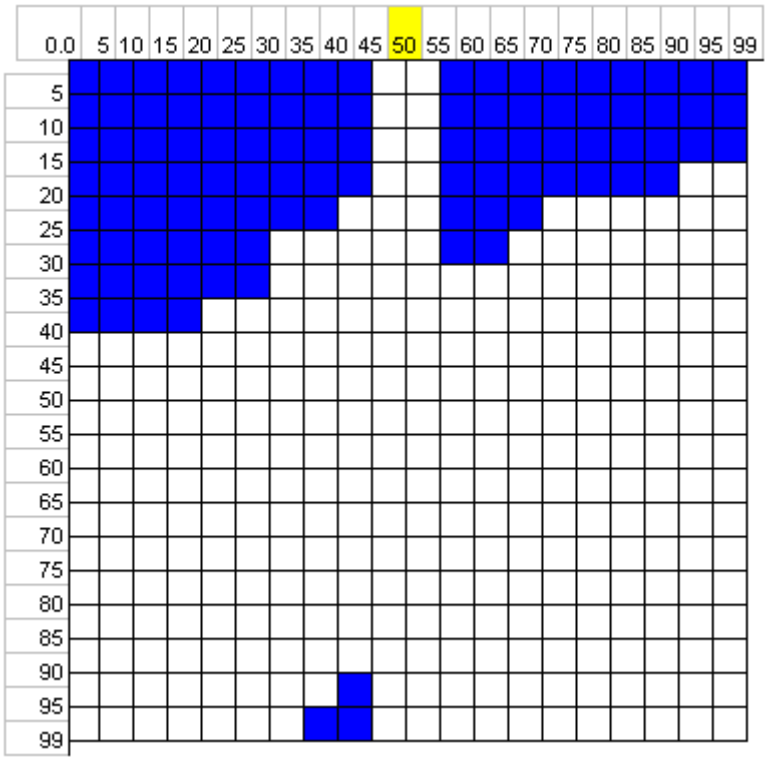
Mamdani bulanık içermesi ile aynı noktada (X=75, Y=65, ϕ=-35^o) farklı üyelik fonksiyonları kullanılarak yapılan uygulamada simulasyon arayüzünde varsayılan olarak verilen üyelik fonksiyonlarının daha performanslı olduğu görülür.

Son olarak da; Larsen bulanık içermesi kullanılarak aynı noktada (X=81, Y=47, ϕ=196^o) farklı üyelik fonksiyonları ile karşılaştırılması sonucu üyelik fonksiyonları değiştirilen bulanık içermeninin daha performanslı olduğu söylenebilir. Burada unutulmaması gereken nokta, kullanıcı simülasyon arayüzünde de görüldüğü gibi sadece Mamdani bulanık içermesinin üyelik fonksiyonlarını değiştirilebilir. Larsen bulanık içermesi için tanımlanan üyelik fonksiyon değerleri simulasyon arayüzüne sığmadığı için varsayılan üyelik fonksiyonları kullanılmaktadır.

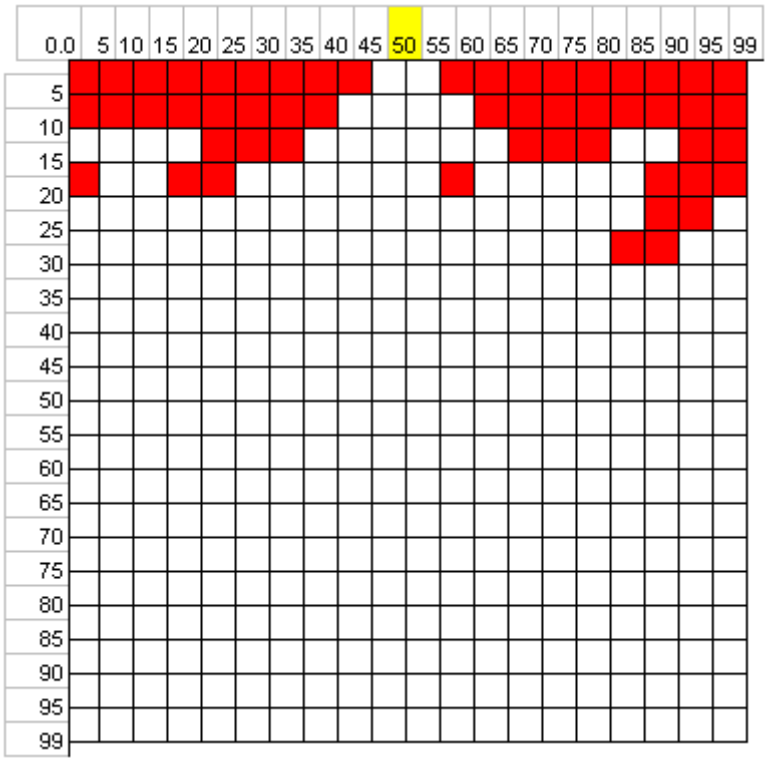
4.8 Sistemin Kararlılık Analizi

Sistemin kararlılık analizi, kamyon hareket alanınındaki belirli noktalarda sabit açı ile kamyonun hedef noktaya ulaşıp-ulaşmamasına bakarak yapılmıştır. Kamyon hedef noktaya ulaşıyorsa o nokta kararlı, ulaşmıyorsa kararsız olarak kabul edilmiştir.

Simulasyon arayüzünde de görüleceği gibi yatay eksen X, düşey eksen Y değerini göstermektedir. X=0'dan 100'e kadar 5'er 5'er artırılarak bu eksende düşey çizgiler çizilmiştir. Aynı şekilde Y =0'dan 100'e kadar 5'er 5'er artırılarak bu eksende yatay çizgiler çizilmiştir. Her iki eksende çizilen çizgilerin kesişmesiyle de elde edilen noktalarda ϕ=90^o sabit kabul edilerek kamyonun Mamdani ve Larsen bulanık içermeleri ile hedef noktaya ulaşıp-ulaşmadığına göre kararlılık noktaları Şekil 4.11 ve 4.12' de ayrı ayrı gösterilmiştir. Şekillerde beyaz alanlar kararlı noktaları gösterir.



Şekil 4.11: Mamdani Bulanık İçermesi Kararlılık Analizi



Şekil 4.12: Larsen Bulanık İçermesi Kararlılık Analizi

4.9 Sonuç

Bu bölümde, model kamyonun bulunduğu konumdan yükleme platformuna ulaşmak için geriye doğru hareket ederek park etme kontrol'ünün bulanık modellenmesi yapılmıştır. Yapılan bu modellemenin bilgisayar ortamında kontrolünü sağlamak için Java programlama dilinde simülasyon arayüzü gerçekleştirilmiştir.

Geriye hareket eden kamyon modellemelerinde, Bulanık Mantık Kontrol uygulaması her zaman başarılı olmuştur. Yine de, varolan birçok Bulanık Mantık Kontrol, hem tek eksenli hareketler için hem de parketme amacı dışında başka gereksinim ve kısıtlama olmayan hareketler için tasarlanmışlardır (Kosko, 1992; Tanaka, 1995). Sonuç olarak, bu tür metotlar eğer kamyonun hareket yörüngesi ilave bir optimallik gereksinimine veya yol kısıtlamalarına maruz kalırlarsa, görevi başarıyla tamamlayamazlar.

Verilen iki bulanık içerme (Mamdani ve Larsen) ile sistemin performans ve kararlılık analizlerinin incelenmesinde bir kıyaslama imkanı doğmaktadır. Bu suretle, hangi içermenin daha performanslı ve kararlı olduğu karşılaştırılmalı olarak gösterilmiştir.

Performans analizinde görüldüğü gibi verilen örneklerde rastgele seçilen bir noktada karşılaştırma yapılmıştır. Daha fazla noktada deneme-yanılma yöntemi kullanarak o noktalar için optimum performans karşılaştırması yapılabilir. Performans analizinin daha iyi çıkması için gerekli diğer adım üyelik fonksiyonlarının belirlenmesidir. Burada üyelik fonksiyonları elle ayarlanabilmektedir. Günümüzde optimum üyelik fonksiyonlarının belirlenmesinde kullanılan diğer yöntemlerden biri de genetik algoritmalardır. Genetik algoritmalar sayesinde sistem için istenen optimum üyelik fonksiyonu yine yazılım programı ile geliştirilebilir.

Kararlılık analizlerinde de görüldüğü gibi analiz $\phi = 90^o$ sabit alınarak belirli noktalarda elle deneme yapılarak gerçekleştirilmiştir. Burada ϕ açısı değiştirilecek olursa farklı bir kararlılık tablosu çıkacaktır. Daha keskin bir kararlılık tablosu için yazılım programı geliştirilmelidir.

BEŞİNCİ BÖLÜM
SONUÇ ve ÖNERİLER

5.1 Sonuçlar

Gerçek dünya olaylarının çok karmaşık olması nedeniyle dolayısı ile bu olayların belirgin denklemlerle tanımlanarak, kesin bir şekilde kontrol altına alınması mümkün olmaz. Mühendislikte bütün teori ve denklemler gerçek dünyayı yaklaşık bir şekilde ifade eder. Yapılan literatür araştırmalarında ve bu çalışmada gözlenen sonuçlara göre Bulanık Mantık Kontrol ile sisteme ait matematiksel modele ihtiyaç duymadan sözel değişkenlerin yardımıyla, kontrol daha esnek ve basit bir yapıya kavuşmuştur.

Üzerinde çalıştığımız Bulanık Mantık Kontrol uygulaması bir kamyonun, bulunduğu konumdan yükleme platformuna ulaşmak için geriye doğru hareketle park etmesidir. Günümüzde bu tür uygulamalar büyük limanlarda gemilere kamyonlar vasıtasıyla konteynır yükleme-boşaltma sisteminde, bir aracın yol üzerinde geri geri giderek park yapmasında ve fabrikalarda belirli alan içindeki herhangi bir noktadan hedef noktaya sürekli malzeme gönderimlerinde ve robotların hareket hareket kabiliyetleri tasarımlarında kullanılabilir.

Bu çalışmada, bir model kamyon yükleme-boşaltma sisteminin, doğal ve gerçeğe uygun parabolik yörüngelerle belirlenmiş yollar üzerinden, hedef noktaya geri geri yanaşarak optimum park edebilmesi probleminin Bulanık Mantık Kontrol kullanılarak Java ortamında simulasyonu incelenmiştir.

Bu amaç için tek bir arayüzden oluşan bir program yazılmıştır. Bu arayüzde Bulanık Mantık Kontrol'üne ait parametrelerin (üyelik fonksiyonlarının değerleri, kamyonun bulunduğu koordinatların belirlenmesi, kamyonun bulunduğu konumun açısı, kamyonun hızı ve simulayon hızı) girilebileceği giriş ekranı bulunmaktadır.

Kamyon yükleme-boşaltma sisteminin simulasyonu ile ilgili yapılan daha önceki araştırma ve uygulamalar genelde Matlab paket programında, C ve C++ programlama dilllerinde gerçekleştirilmiştir. Matlab, C yada C++'ın (ve diğer birçok dillerde) asıl problemi, bunların özel bir hedef için derlenmiş olarak tasarlanmalarıdır. Her ne kadar bir C++ programını herhangi bir tip CPU için derlemek mümkünse de bunun için o CPU'nun hedeflendiği tam bir C++ derleyicisi gerekmektedir. Problem ise derleyicilerin yaratılmasının zaman alıcı ve pahalı olmasıdır. Daha kolay, maliyet açısından verimli bir çözüm herzaman için daha avantajlıdır. Ayrıca, C++ programları hacim olarak daha büyük ve daha çok bellek kullanmaktadır. Bu da belleğin kısıtlı olduğu kontrol uygulamalarında bir dezavantajdır.

Bu çalışmada ise sistemin platformdan bağımsız şekilde her türlü ortamda çalıştırılması sağlanmış ve bu nedenle nesne tabanlı Java programlama dili ile yazılım geliştirilmiştir.

Böylece hem Matlab, C ve C++'da olduğu gibi yüksek bellek gereksinimi ortadan kaldırılmış, hem de derleyeci sorunu giderilmiştir.

Bu çalışmada, Mamdani ve Larsen kontrol algoritmalarının karşılaştırılması yapılmış ve sonuçları, geliştirilen arayüzde kullanıcıya gösterilmiştir. Yapılan uygulamalarda, Mamdani algoritmanın Larsen algoritmasına göre daha iyi sonuç verdiği gözlemlenmiştir. Özellikle Y mesafesinin büyük olduğu yerlerde bu sonuç daha belirgin olarak göze çarpmaktadır.

5.2 Öneriler

Yazılımda kullandığımız Mamdani ve Larsen kontrol algoritmaları dışında, başka kontrol algoritmaları da (Sugeno, SAM, vb) bu tür yazılımların gelişiminde kullanılabilir.

Uygulamalarda da gözlemlenebileceği gibi, üyelik fonksiyonu değerlerinin değişimi sistemin davranışını belirleyen en önemli etkendir. Bu yüzden ileride yapılabilecek üyelik

fonksiyonlarında genetik algoritmalar gibi global arama yöntemlerinin kullanılması ile sistemin daha optimum olması sağlanabilir.

Bu sistem için kullanılan nesne tabanlı Java2 programlama dilinde, sisteme ait toolbox modüler ve gelişmeye açık olduğu için ileriki zamanlarda farklı sistemlere de uygulanabilir.

Ayrıca, bu çalışma ile elde edilen yazılım geliştirme tecrübesinin, bilimsel bir bakış açısıyla bir çok mühendislik ugyulamasına aktarılması mümkün olacaktır. Böylece hem yazılım hem de donanım boyutu olan Java teknolojisi ile, ticari bilgisayar dünyası, endüstriyel kontrol uygulamalarının ihtiyaçlarını karşılayabilecek duruma gelecektir.

Uygulamalı bulanık mantık üzerinde faaliyet gösteren firmalardan biriside INFORM GmbH (fuzzytech)'dir. Bu firma, Java teknolojisi ile ortaya çıkardığı ticari lisanslı ürünlerde daha az kod, daha basit ve modüler bir yapı kullandığı için bu ürünlere olan talebin artmasını sağlamaktadır.

KAYNAKLAR

Asherton, R., IEEE Spectrum, "Moving Java to Factory", Sun Microsystems, 1988.

Babuska,R., "Fuzzy Modeling for Contol", Kluwer Academic Publishers, Boston, 1998.

Bellman, R. E., and Giertz, M., "On the Analytic Formalism of the Theory of Fuzzy Sets", inform Sci., Vol. 5, pp. 149-157, 1973.

Bernard, J. A., "Use of Rulebased System for Process Control", IEEE Cont. Syst. Mag., Vol. 8, No.5, pp. 3-13, 1988.

Berenji, H. R., "Fuzzy Logic Controllers, Sterling Software, Artificial Intelligence Research Branch", NASA Ames Research Center.

Bezdek, J., "Fuzziness vs. Probability – Again(! ?)", IEEE Trans. on Fuzzy Syst., Vol. 2,No. 1, 1994.

Biltek Magazin, 1999.

Castro, J.L., "Fuzzy Logic Controllers Are Universal Approximators", IEEE Trans. Syst.Man Cybernet, Vol. 25, No.4, pp. 629-635, 1995.

Chen, G. and Zhang, D., "Back-Driving a Truck with Suboptimal Distance Trajectories: A Fuzzy Logic Control Approach", IEEE Trans. on Fuzzy Syst., Vol. 5, No. 3, 1997.

Çamaşır Makinasının Hız Denetiminde Bulanık Mantık Temelli Yöntemlerin Uygulanması, Boğaziçi Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi.

Dubois, D. and Prade, H., "What are Fuzzy Rules and How to Use Them", Institut de Recherche en Informatique de Toulouse (I.R.I.T). Université Paul Sabatier, France.

Dubois, D. and Prade, H., "Fuzzy Sets and Probability: Misunderstandings, Bridges and Gaps", Institut de Recherche en Informatique de Toulouse (I.R.I.T)-C.N.R.S. Université Paul Sabatier, France.

Elkan, C., "The Paradoxial Success of Fuzzy Logic", Deparment of Computer Science and Engineering, University of California, 1993.

Elmas,Ç.,Bulanık Mantık Denetleyiciler(1.Basım), 230s, Ankara, 2003.

Ergüven,Çiğdem, Bulanık Mantık Kontrolör ile Klasik PID Kontrolör Algoritmalarının Karşılaştırılması, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi.

Fujitec, F., "FLEX-8800 series elevator group control system", Fujitec Co., Ltd.,Osaka Japan, 1988.

Fuller, R. A., "Note of Neural Fuzzy Systems", 1995.

Galichet, S. and Foulloy, L., "Fuzzy Controllers: Synthesis and Equivalencies", IEEE Transactions on Fuzzy Systems, Vol.3, No. 3, pp. 140-148, 1995.

Hellondorn, H. ve Thomas, C., "Defuzzification in Fuzzy Controllers", Intelligent and Fuzzy Systems, Vol. 1, pp. 109-123, 1993.

Isokangas, A. ve Juuso, E., "Fuzzy Modeling with Linguistic Equations", Control Engineering Laboratory Department of Process Engineering, University of Oulu, Report A No. 11, 2000.

Java Kullanım Kılavuzu, Grup Java, (3.basım), İstanbul, 2002.

Kasai, Y. and Morimoto, Y., "Electronically Controlled Continuously Variable Transmission", in Proj.Int. Congress on Transportation Electronics,Dearborn., MI, 1988.

Klir, G. J., and Folger T. A., “Fuzzy Sets, Uncertainty and Information”, Prentice Hall, Englewood Clifts, N. J., 1988.

Kosko, B., “Fuzzy engineering”, Prentice Hall, 1997.

Kosko, B., “Fuzzy Systems as Universal Approximators”, in: Proc. IEEE Int Conference Fuzzy Systems, San Diego, pp. 1153-1162, 1992.

Kosko, B., “Neural Networks and Fuzzy System”, Englewood Clifts, NJ: Prentice Hall, 1992

Kuo, B, C., Otomatik Kontrol Sistemleri (Çev.:Atilla Bir),7.basım, 933s, İstanbul, 1999.

Kuş, P., Simulasyon Uygulamaları, Kara Harp Okulu Öğretim Başkanlığı Öğretim Elemanı.

Laumond, J. P., “Controllability of a Multibody Mobile Robot”, IEEE Trans. Robot., Automat., Vol. 9, pp. 755-763, 1993.

Lee,C. C., “Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Parts I and II”, IEEE Trans. Syst., Man&Cybern., Vol. 20, pp. 404-435, 1990.

Mamdani, E. H., “Applications of Fuzzy Algorithms for Simple Dynamic Plant”, Proc. IEEE, Vol. 121, No. 12, pp. 1585-1588, 1974.

Mamdani, E. H. and Assilian, S., “An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller”, Int. J. Man. Mach. Studies, Vol.7, No. 1, pp. 1-13, 1975.

Mamdani, E. H., “Advances in the Linguistic Synthesis of Fuzzy Controllers”, Int. J. Man. Mach. Studies, Vol.8, No. 6, pp. 669-678, 1976.

Mamdani, E. H., “Application of Fuzzy Logic to Approximate Reasining Using Linguistic Synthesis”, IEEE Trans. Computer. Vol. C-26, No..12, pp 1182-1191,

1977.

Matlab 6.5 R13 for Windows, Mathworks Inc., 2003.

McConnel, S., “Software Engineering Principles”, IEEE Software, Vol. 16, No. 2, March/April 1999.

Neogita, V. C., “On Fuzzy Systems: PartI, Kybernetes”, Vol. 28, No.5, pp. 597-610, 1999.

Neogita, V. C., “On Fuzzy Systems: PartI, Kybernetes”, Vol. 28, No.8, pp. 945-961, 1999.

Okuyucu, Sadık, Bulanık Kontrollü DC Motor Eğitim Setinin Oluşturulması, Marmara Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi.

Pedrycz, W., “Fuzzy Control and Fuzzy System, Research Studies Press LTD.”, 2nd , Somerset, England, 1993.

Peker, Hıdır, Bulanık Mantık Uygulamaları, Osmangazi Üniversitesi Fen Bilimleri Enstitüsü Yüksek lisans Tezi.

Schildt, H., Java2 (Çev.: Üzeyir Yazıcı), (2.Basım), İstanbul, 2003.

Sinecen, Mahmut, Klima Sistem Kontrolünün Bulanık Mantık ile Modellenmesi, Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Yüksek lisans Tezi.

Sugeno, M. and Kang, K. T., “Structure Idenditification of Fuzzy Model.”, Fuzzy Sets and Systems, Vol. 28, 1988.

Sugeno, M., “An Introductory Survey of Fuzzy Control”, Information of Science, Vol. 36, No. 1, 1985.

Şaka,Saime, Bulanık Kontrol ve Uygulamaları, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi.

Şen,Z., Bulanık Mantık ve Modelleme İlkeleri(1.Basım), İstanbul, 2001.

Takagi, T. ve Sugeno, M., "Fuzzy Identification of Systems and Its Application to Modelling and Conrol", IEEE Trans. Syst. Man Cybernet, pp. 116-132, 1985.

Tanaka, K., "Design of Model Based Fuzzy Controller Using Lyapunov's Stability Approach and Its Application to Trajectory Stabilization of a Model Car", in Teoretical Aspects of Fuzzy Control, H. T. Nguyen, M. Sugeno, R.Tong and R. R Yager, Eds. New York : Wiley, pp. 31-50, 1995.

Terano, T., Asai, K. Ve Sugeno, M., "Fuzzy Systems Theory and Its Apllications", Academic Press Inc., London, 1991.

Togai, M., and Watanabe, H., "Expert System on a Chip: An Engine for a Real-Time Approximate Reasoning", IEEE Expert Syst. Mag., Vol.1, pp. 55-62, 1986.

Ünal, Serkan,Çağlar, Bulanık Mantıkla Proses Kontrolü için Genel Amaçlı Bir Arabirim ve Arayüz Tasarımı, Gebze Yüksek Teknoloji Enstitüsü Mühendislik ve Fen Bilimleri Enstitüsü.

www.fuzzytech.com

Yamakawa, T., "Fuzzy Controller Hardware System", in Proc. 2nd IFSA Congress, Tokyo, Japan, 1987.

Yamakawa, T., "High Speed Fuzzy Controller Hardware System", in Proc. 2nd Fuzzy System Symp., Japan, pp. 122-130, 1986.

Yasunobu, S. and Hasegawa, "Predictive Fuzzy Control and its Application for Automatic Container Crane Operation System", in Proj. 2nd IFSA Congress, Tokyo, Japan, pp. 349-352, 1987.

Yagishita, O., Itoh, O. and Sugeno, M., "Application of Fuzzy Reasoning to the Water Purification Process", in Industrial Applications of Fuzzy Control, M. Sugeno, Ed.

Amsterdam: North-Holland, pp.19-40,1985.

Yasunobu, S. and Miyamoto, S., “Automatic Train Operation by Predictive Fuzzy Control”, in Industrial Application of Fuzzy Control., M. Sugeno, Ed. Amsterdam: North-Holland, pp.1-18, 1985.

Yasunobu, S., Miyamoto, S. and Ihara, H., “Fuzzy Control for Automatic Train Operation System”, in Proj. 4th IFAC/IFIP/IFORS Int. Congress on Control in Transportation Systems, Baden-Baden, 1983.

Yen, J., “Fuzzy Logic – A Modern Perspective”, Senior Member, IEEE , 1998.

Zadeh, L.,A., “Fuzzy Sets”, Information, Control, Vol. 8, pp. 338-353, 1965.

Zadeh, L.,A., “Fuzzy Algorithm”, Information, Control, Vol. 12, pp. 94-102, 1968.

Zadeh, L.,A., “Toward a theory of fuzzy systems.”, in Aspects of Network and System Theory. R. E. Kalman and Winston., pp. 469-490,1971.

Zadeh, L.,A., “Analysis of Complex Systems and Decision Processes”, IEEE Trans. Syst., Man&Cybern., Vol. 20, No.1, 1973.

Zadeh, L.,A., “Consept of a Linguistic Variable and Its Application to Approximate Reasoning I, II, III, Information Sciences, .8, pp. 199-249, 301-357, 9, 43-80, 1975.

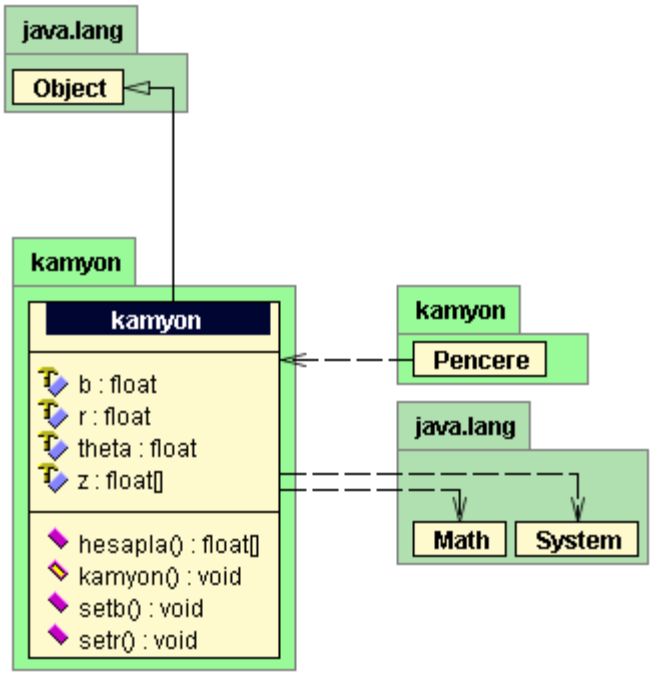
Zadeh, L.,A., “Fuzzy Logic=Computing with Words”, IEEE Trans. on Fuzzy Syst., Vol.4, No. 2, 1996.

Zimmerman, H., “Fuzzy Set Theory and Its Applications”, Kluwer Academic Publisher, Dordrecht, Germany, 1991.

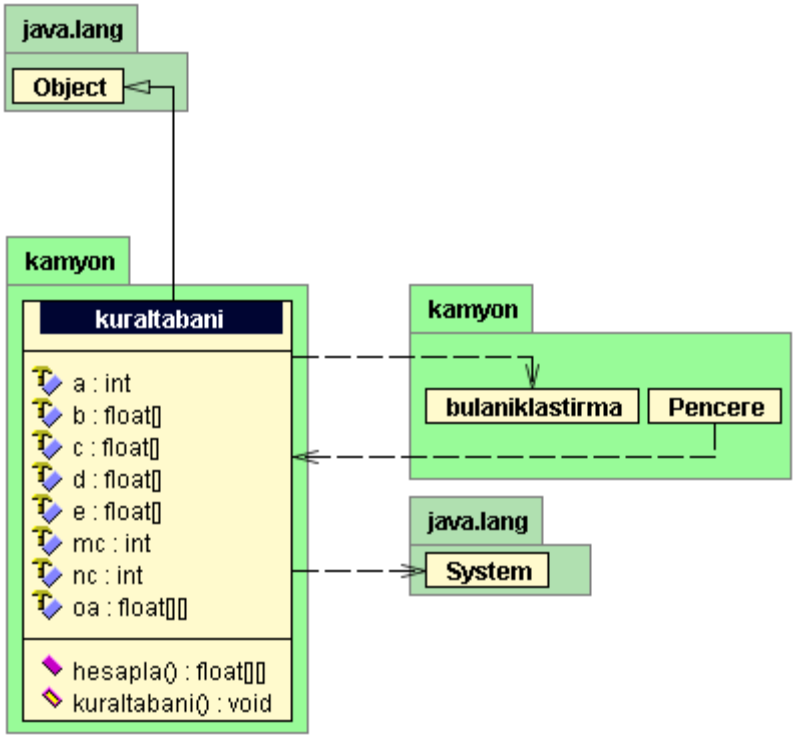
Yen, J., Langari, R. and Zadeh, L.A., “Industrial Applications of fuzzy Logic and Intelligent Systems”, IEEE press, 1995.

EK-A

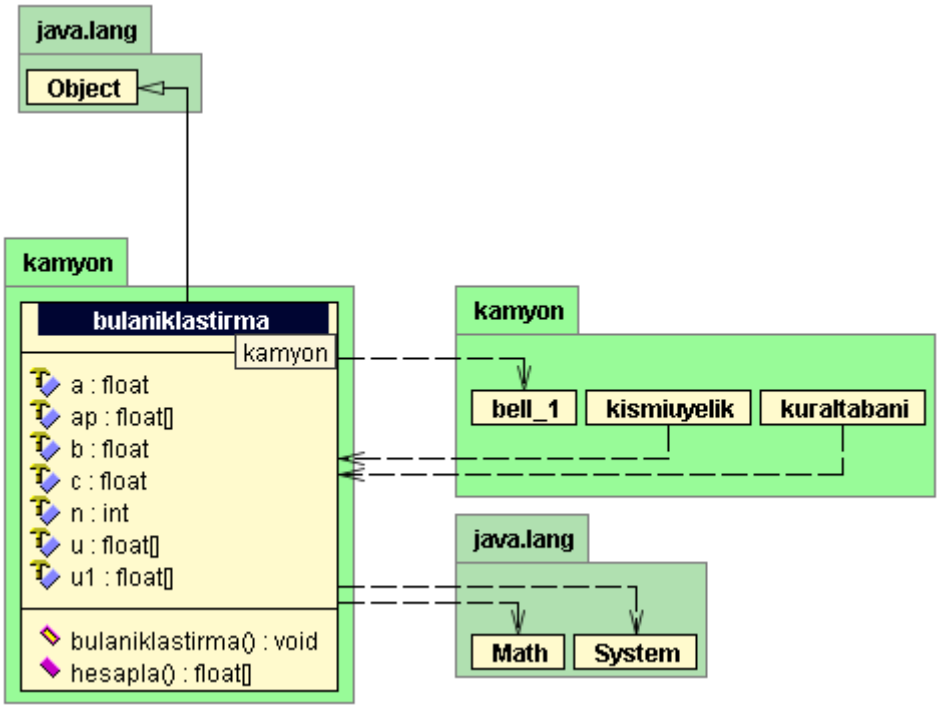
Kamyon Tanım Sınıfı



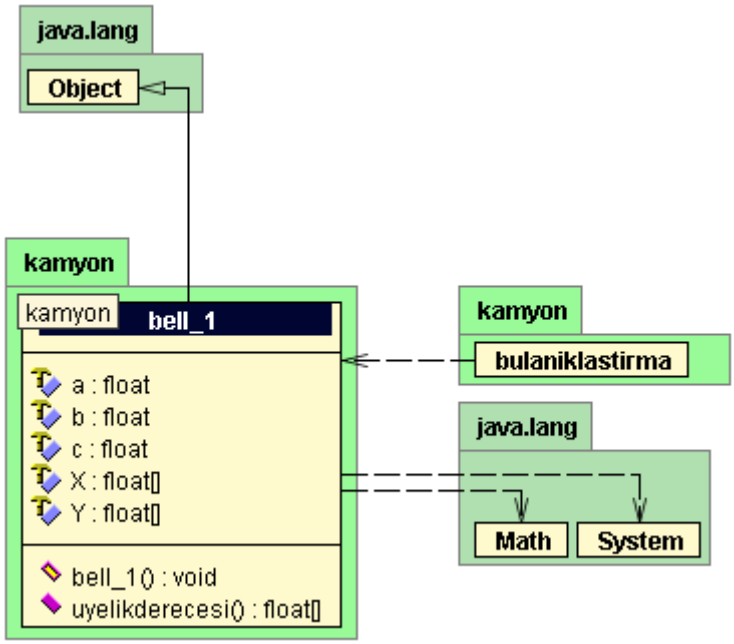
Kural Tabanı Sınıfı



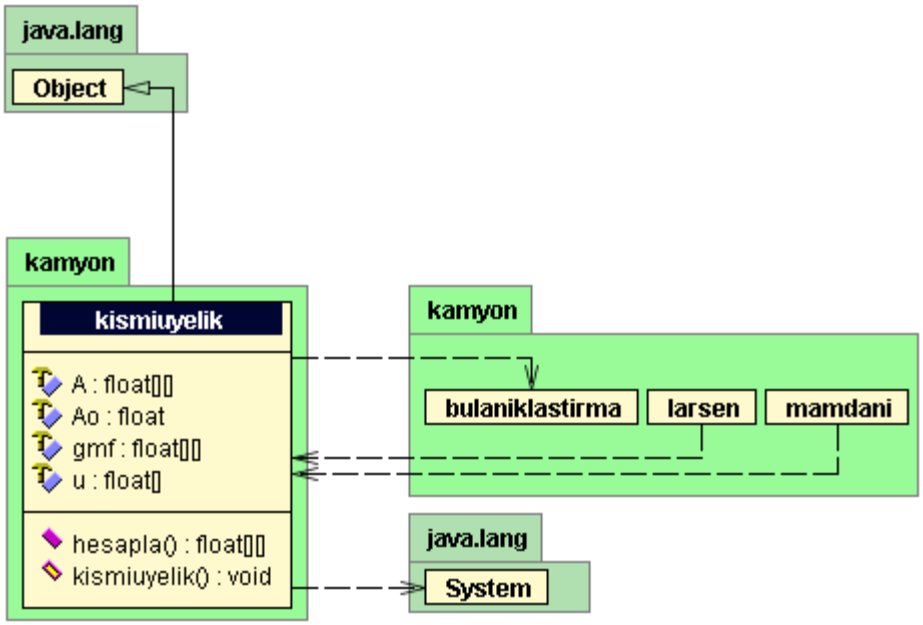
Bulanıklaştırma Sınıfı



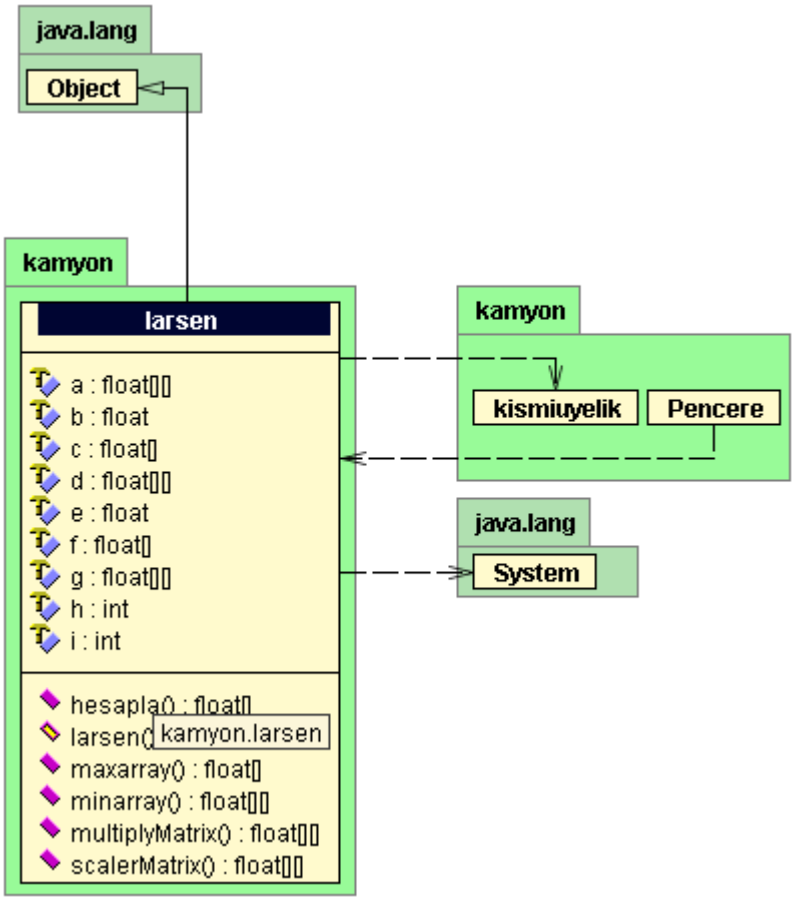
Bell Üyelik Fonksiyon Sınıfı



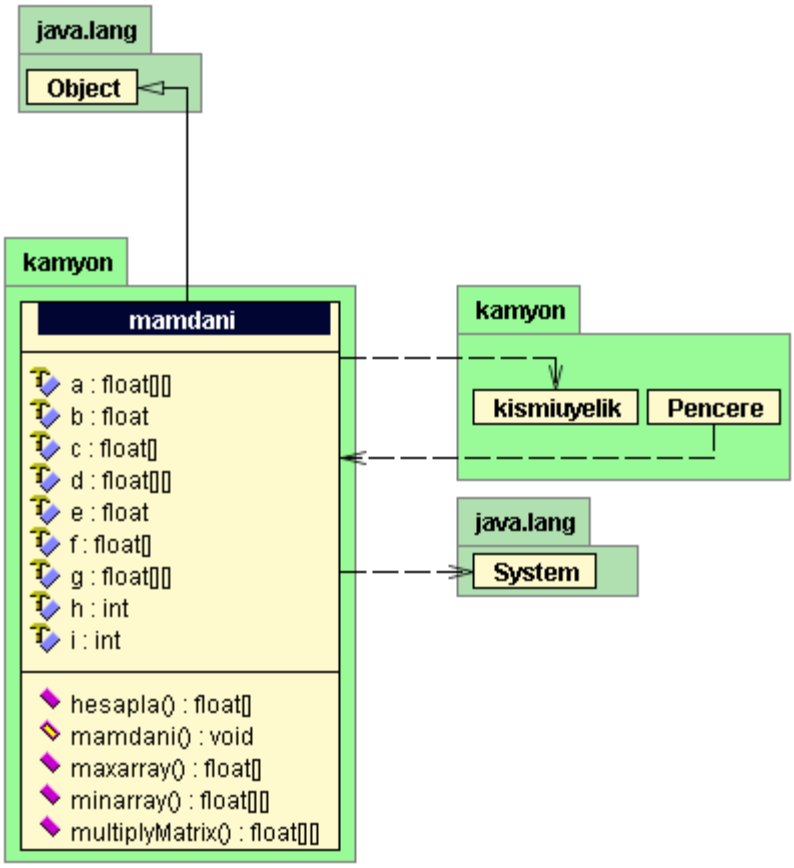
Kısmı Üyelik Sınıfı



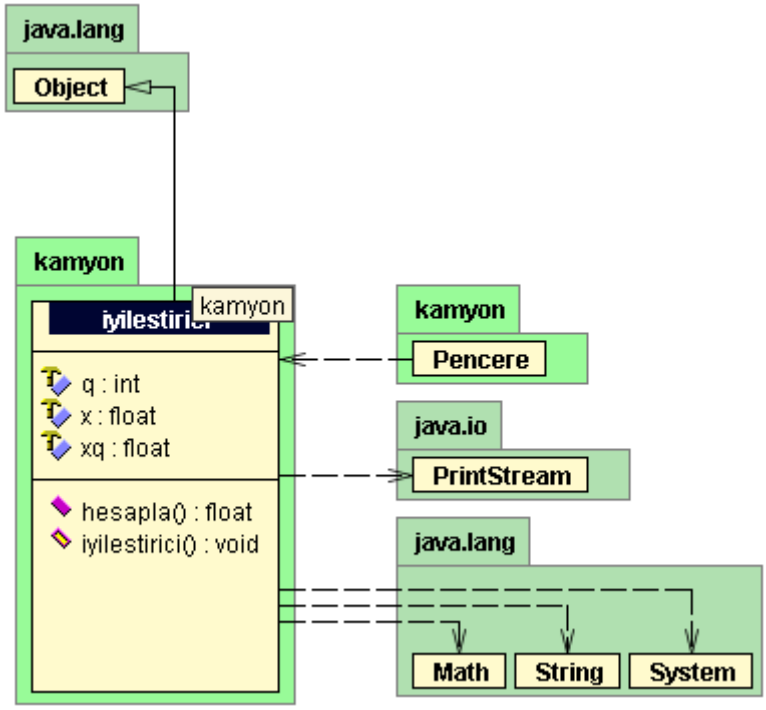
Larsen Kontrol Sınıfı



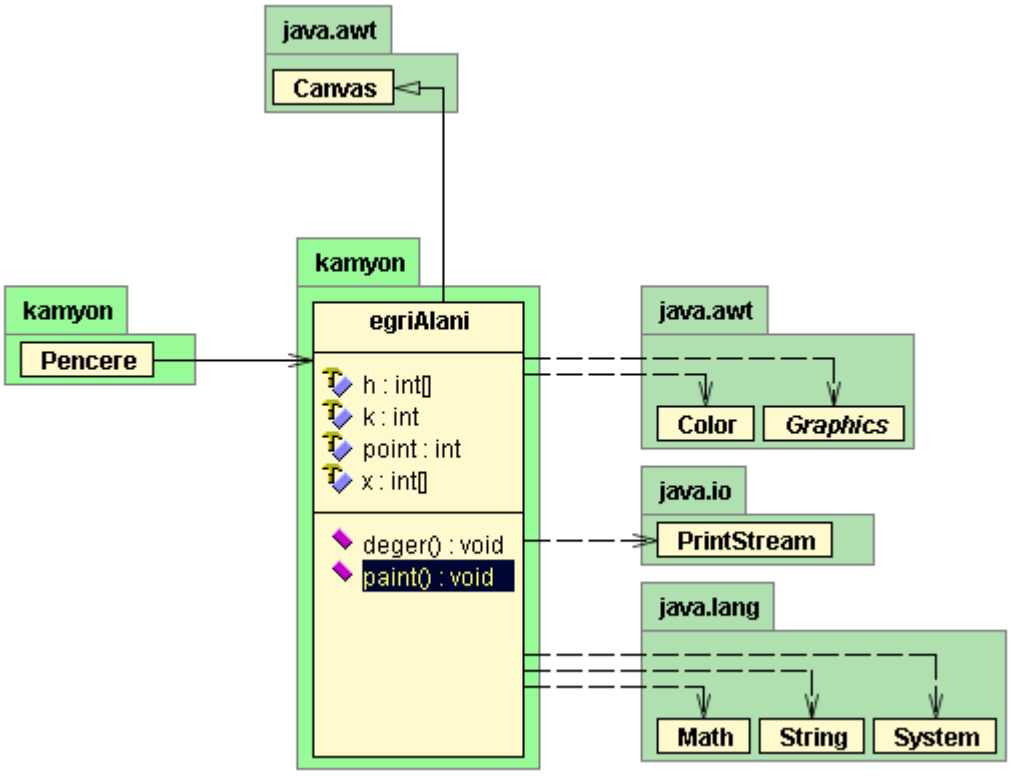
Mamdani Kontrol Sınıfı



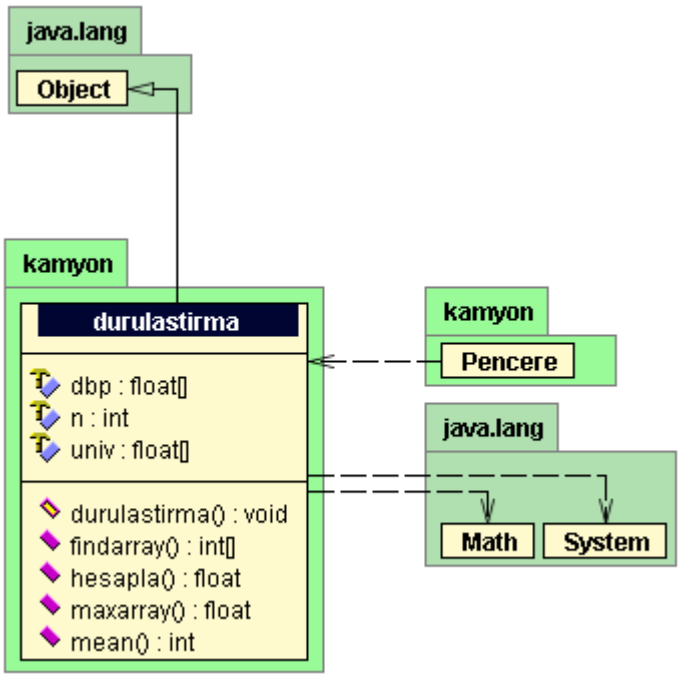
İyileştirici Sınıfı



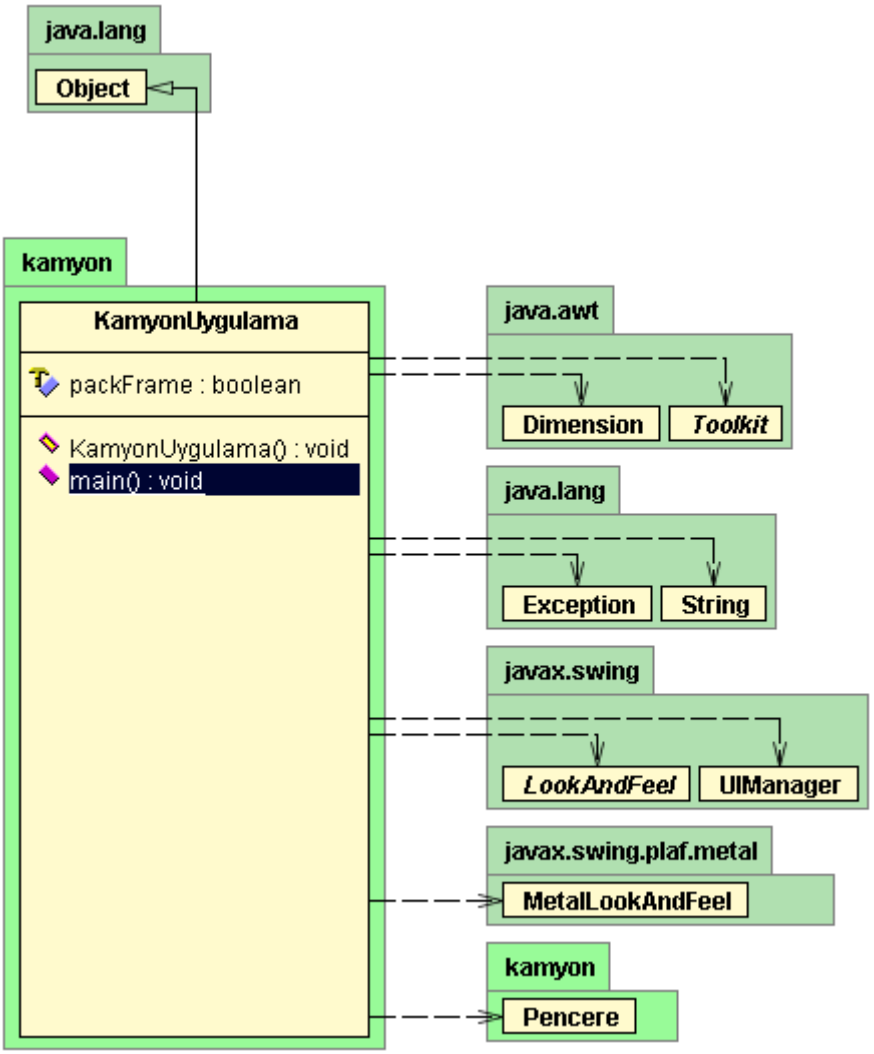
Hata Değeri Sınıfı



Durulaştırma Sınıfı



Kamyon Uygulama Sınıfı


java.lang
Object
kamyon
KamyonUygulama
packFrame : boolean
KamyonUygulama() : void
main() : void
java.awt
Dimension
Toolkit
java.lang
Exception
String
javax.swing
LookAndFeel
UIManager
javax.swing.plaf.metal
MetalLookAndFeel
kamyon
Pencere

ÖZGEÇMİŞ

Adı Soyadı	**:** Ömer KARAL
Ana Adı	**:** Göksal
Baba Adı	**:** Ahmet
Doğum Yeri ve Tarihi	**:** Afyon, 1977
Lisans Eğitimi ve Mezuniyet Tarihi	**:** Pamukkale Üniversitesi, 2000
Yabancı Dil	**:** İngilizce