YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DOĞRU AKIM MOTORUNUN GENETİK ALGORİTMALAR YARDIMIYLA BİLGİSAYAR TEMELLİ PI-TİP BULANIK MANTIK KONTROLU

Elek-Elkn Yük. Müh. Mehmet BULUT

F.B.E Elektrik Mühendisliği Anabilim Dalında
Hazırlanan

DOKTORA TEZİ

Tez Savunma Tarihi	: 23 Ekim 2001
Tez Danışmanı	: Prof. Dr. Galip CANSEVER
Jüri Üyeleri	: Prof.Dr. Halit PASTACI
	: Prof. Dr. Burhanettin CAN
	: Prof. Dr. Cingiz HACIYEV
	: Yrd. Doç. Dr. Selahattin DİNLER

İSTANBUL, 2001

İÇİNDEKİLER

KISALTMA LİSTESİ

AC	Alternatif Akım (Alternative Current)
DC	Doğru Akım (Direct Current)
GA	Genetik Algoritmalar (Genetic Algorithms)
PI	Oransal + Integral (Proportional and Integral)
FLC	Bulanık Mantık Kontrol (Fuzzy Logic Control)
MF	Üyelik Fonksiyonu (Membership Function)
FMRLC	Bulanık Model Referans Öğrenen Kontrol (Fuzzy Model Reference Learning Control)
GBS	Genetik Bulanık Sistem (Genetic Fuzzy Systems)
ISE	Hatanın Karesinin İntegrali (Integral-Square Error)
ITAE	Mutlak Hatanın Zaman Çarpımlı İntegrali (Integral -Time-Absolute Error)
DAC	Sayısal-Analog Dönüştürücü
ADC	Analog-Sayısal Dönüştürücü

ŞEKİL LİSTESİ Sayfa

ÇİZELGE LİSTESİ Sayfa

ÖNSÖZ

Bu tezin hazırlanması süresince yakın ilgisini ve desteğini esirgemeyen, çalışmalarımda bana sabırla yön veren tez danışmanım Sayın Prof.Dr. Galip CANSEVER'e teşekkürü bir borç olarak kabul ediyorum ve saygılarımı sunuyorum.

Çalışmalarım boyunca bana manevi destek ve moral kaynağı olan çocuklarım Abdülkadir, Zehra Nur ve Fatma Gül'e ve teşvikleriyle beni destekleyen ve sabrıyla bana tahammül eden Eşime ve Aileme içten teşekkürlerimi sunarım.

ÖZET

Günümüzde yaygın olarak endüstride kullanılmaya başlanılan bulanık kontrol sistemlerinde en önemli kısım, sisteme ait kural tabanın oluşturulmasıdır. Çünkü iyi sonuçlar alınabilecek kural taban, ancak sistemi tanıyan ve sistem hakkında tecrübeye sahip bir uzman tarafından tanımlanabilir. Bu da sistem için gerekli kontrol yapısını oluşturmada, ancak uzun zaman alan denemeler sonucu başarılabilir. Son yıllarda bu ve benzeri problemler nedeniyle bulanık kontrol sistem için gerekli kural tabanı otomatik olarak öğrenme veya örneklerden çıkaracak araştırıcı yöntemler kullanma yoluna gidilmektedir. Bu çalışmada, bir DC motorun bulanık kontrolünde gerekli kural tabanın belirlenmesi için öğrenme metodları araştırılmış ve Bulanık Mantık kontrolör için Genetik Algoritmalar temelli yeni bir öğrenme algoritması geliştirilmiştir. Geliştirilen metod ile elde edilen PI-tip bulanık kontrolör, Bulanık Model Temelli Öğrenme algoritması (Layne J.R., Passino K.M., 1993) ile elde edilen PD-tip Bulanık kontrolör ve manuel olarak yapılandırılan bir bulanık kontrolör sonuçları ile karşılaştırılmış ve geliştirilen öğrenme algoritmasının DC motorun bulanık kontrolünde başarılı sonuçlar verdiği gözlenmiştir.

Bu tez çalışmasında, Bulanık Model Referans Öğrenen Control Algoritması (FMRLC- Fuzzy Model Reference Learning Control) ilk olarak DC Motor hız kontrolü için simulasyon bazda kullanılmış ve elde edilen sonuçlar "International Conference on Signal Process Applications And Technologies , Dallas, TX, October 2000, USA" konferansında sunulmuştur. Bu algoritma öğrenmeye dayalı olduğundan DC motor PD-tip bulanık kontrolör dizaynı için kullanılmıştır. FMRLC algoritması doğrudan bulanık kontrolör bilgi tabanını tasarlamak ve yeni durumlara karşı düzenlemek için otomatik öğrenmeye bağlı bir metod sağlamaktadır. PD-tip Bulanık Kontrolöre ait elde edilen kural tabanın tezde kullanılan DC motor transfer fonksiyonuna uygulanmasıyla elde edilen sonuçlar, bu tezde geliştirilen Genetik Bulanık Sisteme ait sonuçların değerlendirilmesinde kullanılmıştır.

Bu tezde kullanılmak üzere, PC parallel portu üzerinde 8-bit olarak bilgi transferi olanağı sağlayan ve üzerinde motor sürme devresi ve hız geri bilgisi sağlayabilen bir dijital I/O (giriş-çıkış) arabirim oluşturulmuştur. Kontroller PC üzerinde oluşturularak on-line olarak motor kontrolü gerçekleştirilmiştir. Motor hız bilgisi arabirim üzerinden bilgisayara gönderilerek grafik üzerinde izlenmesi sağlanmıştır.

DC motor hız kontrolü için Genetik temelli Bulanık Kontrol Kural Tabanı Öğrenme algoritması (GBS) geliştirilerek en uygun kurallar araştırılmıştır. Genetik Bulanık Kontrol Algoritması Turbo C programlama dili ile yazılmıştır. Kontrol edilecek sistemin parametreleri kullanılarak matematiksel modeli elde edilmiş ve ilk olarak bu model üzerinde uygulanmıştır. Off-line olarak elde edilen kural taban bulanık kontrol programına yerleştirilerek PC üzerinden tasarımı yapılan arabirim devresi üzerinden motora başarıyla uygulanmıştır. Alınan sonuçlar göz önüne alındığında GBS algoritması elde tasarlanan PI-tip Bulanık kontrolörün FMRLC öğrenme algoritması ile tasarımı yapılan PD-tip bulanık kontrol sistemine göre daha iyi performansa sahip olduğu görülmüştür. Genetik algoritmanın bir çözüm uzayında en iyiyi araştırma yapısından dolayı algoritmanın her çalıştırılışında farklı kural taban elde edilmektedir. Verimli sonuç alınması, algoritma çözüm havuzun geniş olarak seçilmesiyle orantılı olmakta fakat öğrenme zamanı açısından süreyi uzatmaktadır.

Anahtar Kelimeler : Bulanık Kontrol, Genetik Algoritmalar, DC motor, Kural Taban Öğrenilmesi

ABSTRACT

Today, Fuzzy Control are now considered as one of the most important applications of the fuzzy-rule-based systems which is widely being used in industry. The experience of skilled operators and the knowledge of control engineers are expressed qualitatively by a set of fuzzy control rules. The construction of fuzzy rules has been mainly based on the operator's control experience or actions. Genetic algorithms (GAs) are search algorithms based on the mechanics of natural selection and natural genetics. GAs have the properties that make them a powerful technique for selecting high-performance parameters for FLCs.

In a few years many different approaches have been presented using the genetic algorithms (Gas) as a base of the learning process. Gas have demonstrated to be a powerful tool for automating the definition of the fuzzy control rule knowledge base. These approaches called the general name of genetic fuzzy systems (GFSs).We propose a GFS methodology based on three stages. In this thesis, genetic algorithms are used to learn PI-typefuzzy control rules for direct current motor speed control. Comparisons are made between systems utilizing learning rules and human expert based rules to verify the performance of the work.

Recently GFS algorithm has been searched in developing well-performing fuzzy rule-base without help of human expertise. Learning the rule-base requires experience human with long time. Genetic Fuzzy System provides a method to design automatically the knowledge base a direct fuzzy controller.

In this work, we are interested in automatically learning rule-base for a fuzzy logic based dc motor controller. At the same time, we provide an analysis of Genetic Fuzzy System for dc motor control system. It seems that we designed GFS is powerful method for control of dc motor and gives well performance. In this study, I have been used Pentium 120 mHz computer and Turbo C program. Hence, it provides an approach to computer aided design about automation for dc motor control system.

As a result, this approach can provide a low-cost and robust means of design of the fuzzy rule-based controller. We achieved a result, the proposed genetic algorithm model has demonstrated its capabilities in term so high robustness, flexibility and reliability by consistently improving the performance of the fuzzy logic controller.

Keywords: Fuzzy control, Genetic Algorithms, Rule Base, DC motor, Learning Control

1. GİRİŞ

1.1 Tarihçe

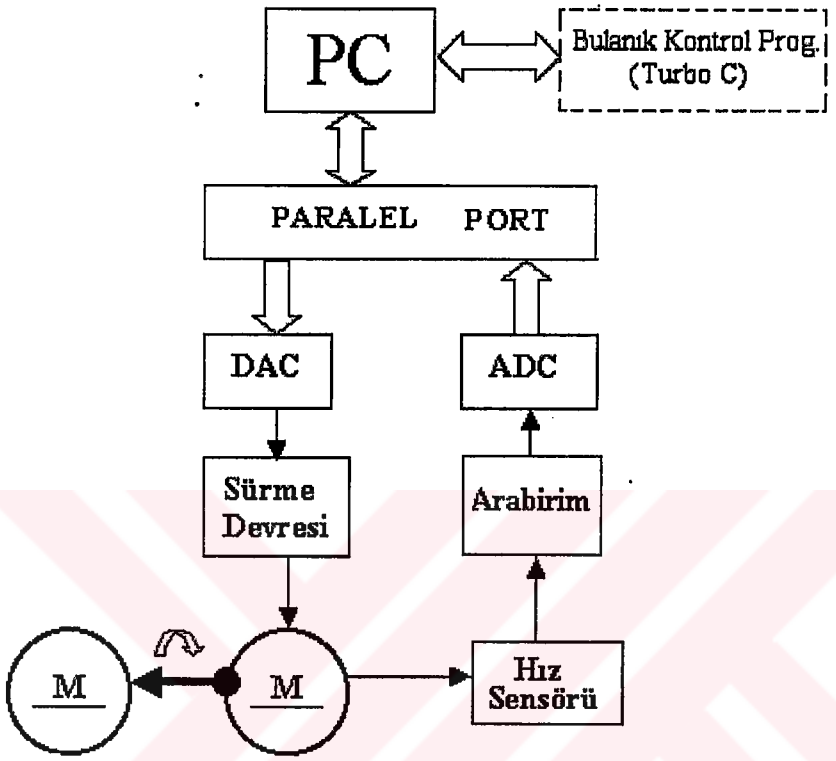
Bulanık mantık temelli kontrol sistemlerinin bilinen avantajlarına rağmen, analitik tasarım yönteminin eksikliği yüzünden bunların tasarımı hala çoğunlukla deneme ve yanılma yoluyla olmaktadır. Bununla birlikte genetik algoritmalar ve benzeri yöntemler ile kendinden öğrenen algoritmalar mevcut olup böyle bir engelin üstesinden gelmek mümkün olmaktadır. Ayrıca bulanık kurallarını otomatik olarak düzenlemek için literatürde kullanılan farklı yöntemler bulunmaktadır (Ross, Timothy J., 1995).

Genetik algoritmalar, Bulanık Kontrol sistemleri için yüksek performanslı parametre seçiminde kullanılabilecek güçlü tekniklere sahip özellik taşırlar. Genetik algoritmaların bulanık kontrol sistemlerinin kural tabanı tasarımında kullanılması ilk olarak pH kontrolü, inverted pendulum gibi uygulamalarda ortaya çıkarılmıştır. Örneğin Lee ve Takagi inverted pendulum için genetik algoritmalar kullanarak bulanık kontrol tasarım etmişlerdir (Lee and Takagi, 1993). Karr ve Gentry, Genetik Algoritmalar yardımıyla düzenlenebilen bulanık sistemin üyelik fonksiyonlarıyla bir asidin pH değerini kontrol etmeye çalışmışlardır (Karr and Gentry, 1993). Park ve arkadaşları doğru akım seri motoru kontrol etmek için genetik algoritma bulanık yaklaşım modelini optimize etmeye çalışmışlardır (Park D., Kandel A. and Langholz, G., 1994). Chin T.C. ve Qi X.M. ise kural kümesi belli olan yani önceden belirlenen bir bulanık kontrol sistemine ait kural tabandaki kural sayısını minimize ederek FLC (Bulanık Logic Controller) 'nın performansını artırmak için genetik algoritmaları kullanmışlardır (Chin T.C. ve Qi X.M, 1998).

Son yıllarda bu konuda yapılan son çalışmalardan birisi de Gürocak H.B. tarafından geliştirilmiştir. Bulanık mantık kontrol sisteminin kural tabanın ayarlanması için genetik algoritma temelli bir metod kullanmıştır. Geliştirilen metod bir inverted pendulum üzerine simule edilmiştir. Çalışmanın sonuç bölümünde geliştirilen metodun ancak, uygun üyelik fonksiyonları ve iyi tanımlı kurallar belirlenmesinden sonra kural tabanın ayarlanması (tunning) için en son tasarım işlem basamağı olarak kullanılabileceği belirtilmektedir. Ayrıca bu metodu sınırlayan başka bir unsurda başlangıçta belirlenen kural tabana dayanmasıdır (Gürocak, H.B., 1999).

Önceden yapılan bu çalışmalar ve benzerleri daha çok Bulanık Kontrol kural sayısını azaltmak üzerinde ve bulanık kontrol sisteminin parametrelerini optimize etmeye çalışılmıştır. Ayrıca yapılan çalışmaların çoğu simulasyondan öteye geçmemiştir. Buna karşın bu

çalışmada, literatürdeki bunlar gibi benzer çalışmalar baz alınarak yüksek performanslı kural tabanın bulanık kontrol sistemine öğretilmesi yani hiçbir ön bilgi ve tecrübe bilgisi kullanmaksızın çıkarılması ve gerçek sistem üzerine uygulanması amaçlanmıştır.



Şekil 1.1 Yapılan çalışma için tasarlanan deney seti blok diyagramı

Yukarda değinildiği gibi bir sistemi kontrol edecek normal bulanık kontrolcuda kullanılacak kuralların belirlenmesi için bir uzman bilgisi gerekmektedir. Bu çalışmada ise DC motoru kontrol edecek PI-tip Bulanık kontrol kurallarının Genetik Algoritmalar (GA) kullanılarak çıkarılması, yani PI-tip Bulanık kontrol sistemine öğretilmesi hedeflenmektedir. Yani bir uzman bilgisine ihtiyaç duyulmadan Genetik Algoritmalar kullanılarak bir Bulanık Kontrolcu tasarımı yapılarak DC motor modeli üzerinde off-line olarak çalıştırılmaktadır. Model üzerinde alınacak sonuçlardan sonra elde edilecek parametreler, PI-tip Bulanık Kontrol algoritmasına yerleştirilerek PC üzerinden bir DC motorun kontrolüne uygulanacaktır. Böylece bir uzman bilgisi kullanılmaksızın DC motor bulanık kontrolcu tasarımı gerçekleştirilmiş olunacaktır. Performans ölçümü yapılarak, DC motor kontrolü için FMRLC ile elde edilen PD-tip Bulanık kuralları ve klasik yöntemler ile karşılaştırılması yapılacaktır.

1.2 Amaç

Bu çalışmada doğru akım motor hız kontrolü için insan tecrübesini kullanmaksızın Genetik Algoritmalar kullanılarak PI-tip Bulanık kontrolör tasarlanması amaçlanmıştır. Genetik algoritmalar yardımıyla tasarlanan Bulanık kontrol sistemi kullanılarak PC üzerinden DC motor devir kontrolü yapılmaktadır. Motor devir bilgisi sensör yardımıyla frekans bilgisine ve sonradan gerilime dönüştürülerek bir arabirim vasıtasıyla PC'ye aktarılmaktadır. Burada C Programlama dili yazılan kontrol algoritmasında kullanılarak sistemi verilen referansa değerde tutmak için gerekli kontrol sinyali bulunarak tekrar arabirim üzerinden motora uygulanmaktadır.

1.3 Yöntem

Bu çalışmada doğru akım motor hız kontrolü için insan tecrübesini kullanmaksızın Bulanık PI kontrol sistemi tasarlanması amacıyla Genetik Algoritmalar temelli bir öğrenme yöntemi geliştirilmiştir. Bu metod üç aşamada gerçekleştirilmiştir; ilk olarak bulanık kuralların elde edilmesinde kullanılan ve iteratif olarak çalışan genetik algoritma temelli bir öğrenme algoritması geliştirilmiştir. İkinci olarak genetik bulanık öğrenme algoritmasının her nesilde elde edilen tüm kural dizileri motor matematiksel modeli üzerinde simule edilerek en uyumlu kural taban araştırması yapılmaktadır. İstenen performans değerli kural taban bulununcaya kadar veya maksimum nesil sayısına ulaşıncaya kadar algoritma çalıştırılmaktadır. Üçüncü aşamada istenen uyumluluk değerine sahip kural tabanı elde edildikten sonra bu kural tabanı PI-tip Bulanık Kontrolcu yapısına yerleştirilerek direk olarak motora uygulanmaktadır.

1.4 Tezin Bölümleri

Tezin bölümlere dağılışı ana hatlarıyla aşağıdaki gibidir.

1.4.1 Giriş

Daha önce yapılan çalışmalar ve bu tez çalışmasının tanıtımı ve içerdiği konular tanıtılmıştır.

1.4.2 Doğru Akım Makinaları

Bu bölümde kontrol edilen sistemi meydana getiren doğru akım (DC) makinaları tanıtılmıştır. Elektronik olarak DC motorun kontrol anlatılarak, sürücü teknikleri ile ilgili bilgiler verilmiştir. Serbest uyarmalı doğru akım motorunun matematiksel modeli verilerek çeşitli hız kontrol yöntemleri anlatılmıştır.

Doğru akım (DC) makinaları uzun süredir hız kontrolünde kullanılmaktadır. Endüstride yaygın olarak kullanılmalarında, çok yönlü kontrol karakteristikleri rol oynamaktadır. DC motorları büyük kalkış torku sağlayabilmektedir. Nominal hızın altında ve üstünde olmak üzere geniş bir aralıkta hız kontrolü kolayca yapılabilmektedir. Kontrol yöntemleri de alternatif akım (AC) motor kontrolcularına göre daha basit ve ucuzdur.

Günümüzde DC motorlar AC motorlara göre çok yaygın olmamalarına karşın, bazı kesin uygulamalar için daha uygun olmaktadırlar. DC motorlar değişken karakteristiklere sahiptirler ve değişken yapıdaki sürücülerde geniş bir şekilde kullanılmaktadırlar. Bu motorlar yüksek bir kalkış torku sağlayabilmekte ve geniş bir aralıkta hız kontrolüne de imkan tanımaktadır. Hız kontrol yöntemleri, AC sürücülerine göre daha basit ve ucuz olmaktadır. DC motorlar, kollektörlü olmaları sebebiyle yüksek hızlı uygulamalar için uygun olmamakta ve AC motorlara göre daha fazla bakım gerektirmektedir. Armatür ve alan sargılarının ayrı kaynaklardan beslendiği serbest uyarmalı DC motoru vasıtasıyla çok esnek kontrol yapılabilmektedir. Bu düzenleme motorun hız tork karakteristiğini hemen hemen ideal bir karakteristiğe yaklaştırmaktadır. Armatür gerilimi kontrollü bir doğrultucu veya bir kıyıcıdan kontrol edilebilmektedir. Eğer uyarma akımı kontrol edilmek istenirse, benzer düzenleme geçerli olabilmektedir (Dubey G.K., ve Kasarabada, C. R., 1993).

1.4.3 Dijital Kontrol Sistemleri

Otomatik Kontrol, genel olarak otomatik kontrol doğrudan insan müdahalesi olmaksızın çalışan aygıtların sistemlerin ve fabrikaların gelişmeleri ile ilgilenen bir bilim ve teknoloji dalı olarak tanımlanır. Kısaca bir sistemde faaliyetlerin insan müdahalesi olmaksızın önceden belirlenen bir amaca göre denetimi ve yönetilmesi olarak ifade edilir.

Bilgisayar teknolojisindeki gelişme ile birlikte, analog kontrol sistemlerin yerini dijital kontrol sistemlerine bırakmışlardır. Uygun olmaları, esnek bir yapıda olmaları, parametre değiştirilme olanakları sağlamaları ve maliyeti azaltmaları nedeniyle tercih sebebi olmuşlardır. Dijital kontrol sistemleri, kullandıkları işaretler bakımından sürekli sistemlere göre farklılık gösterir. Dijital kontrol sistemlerinde, kontrol sisteminin bir kısmında yada tümünde zamanda süreksiz işaretler vardır. Pratikte kullanılan dijital kontrol sistemlerindeki işaretler, sürekli işaretler, örneklenmiş işaretler ile bilgisayar yada mikroişlemciler ve kodlayıcıların ürettiği dijital işaretlerden oluşur . Bu sebeple dijital kontrol sistemlerinde, sürekli (analog) işaretlerin dijitale çevrilmesi ve gerekli işlemlerden sonra dijital işaretlerin tekrar sürekli işaretlere çevrilmeleri gerekmektedir. Bu işlemler Analog-Dijital Çevirici

(ADC) ve Dijital-Analog Çevirici (DAC) gibi temel ara elemanlarla gerçekleştirilmektedir (Sarıoğlu, M. K., 1991). Tezin bu bölümünde dijital kontrol ve PI kontrol sistemlerinin yapıları anlatılmakta ve bu çalışmada kontrol edilecek DC motor sisteminin matematik modeli, transfer fonksiyonu çıkarılmıştır.

1.4.4 Bulanık Mantık Kontrol Algoritması

Bulanık mantık, insan düşünme ve algılamasını modellemek için kullanılan güçlü bir araçtır. Klasik iki değerli mantığın doğru ve yanlış (0 veya 1) olan doğruluk değerleri daha esnek hale getirilmektedir. Bunun sonucunda bulanık mantık ortaya çıkmıştır. Bulanık küme konusu ilk defa Lotfi A. Zadeh tarafından 1965 yılında ortaya atılmıştır .

Zadeh'in bu çalışması : İnsanların bazı sistemleri makinelerden daha iyi kontrol edebilmelerinin bir sebebi olarak insanların belirsiz, yani kesinlik ifade etmeyen bir takım bilgileri kullanarak karar verebilme özelliğine sahip olmalarını olduğunu göstermiştir. Dolayısıyla eğer bu özellik sistemlerin modellenmesinde kullanılırsa tasarım edilen kontrol sistemlerinin performansının arttırılması mümkün olacaktır.

Pratik Uygulamalarda Kullanılması, 1970'li yıllarda gerçekleşmiştir. Bulanık Mantık Kontrolcu (FLC-Fuzzy Lojik Controller) 'nin dünyaya tanıtılmasında önemli olaylardan biri FLC 'nin Sendai (Japonya) metrosunun otomatik kontrolü için kullanılması olmuştur (1987). Özellikle Japonya'da 90'lı yıllarda FLC kullanılan tüketici ürünleri pazarda sıkça görülmeye başlanmıştır Bunun yanında birçok endüstriyel uygulamalarda da FLC başarıyla kullanılmıştır. Tezin bu bölümünde bulanık mantık ve onu oluşturan yapılar ayrıntılarıyla incelenmiştir. Ayrıca bir örnek üzerinde bulanık kuralların nasıl çalıştığı pratik olarak hesaplamalar yapılarak gösterilmiştir.

1.4.5 Model Temelli Bulanık Kontrol Bilgi Tabanı Öğrenilmesi ve DC Motora Uygulanması

Burada sistemdeki parametre değişimlerine göre kontrollerin katsayılarını ayarlayan adaptasyon mekanizması gibi geleneksel bir algoritma yerine bir bulanık adaptif metodu kullanılabilir. Yani bu kazanç katsayıların hesaplanması için bulanık mantık kullanılabilir. Böyle bir bulanık prosedür bulanık kuralları üzerine dayanacaktır. Bu bulanık kuralları sistemin iç yapısından ve sistem davranışlarından çıkarılır. Yalnız bütün sistemler için bulanık kuralların çıkarılması mümkün değildir. Çok değişken davranış gösteren ve nonlineer olan sistemlerde bulanık kurallarının çıkarılması için literatürde çeşitli yöntemler

kullanılmaktadır. Çalışmanın bu bölümünde Bulanık Model Referans Öğrenme Algoritması (FMRLC) (Layne J.R., Passino K.M., 1993) öğrenmeye dayalı olduğundan bu yöntem kullanılarak PD-tip Bulanık kontrolör tasarlanarak kontrol etmek istediğimiz DC motor sisteminin matematiksel modeli üzerinde kullanılmış elde edilen sonuçlar verilmiştir. Elde edilen sonuçlar bu tezde tasarlanan Genetik Bulanık sistem (GBS) öğrenme algoritması ile elde edilen sonuçların değerlendirilmesinde kullanılmaktadır.

1.4.6 Sistemde Kullanılan Genetik Algoritmalar

Genetik algoritmalar, doğal seçme ve doğal genetik kurallara dayanan bir arama türüdür. Doğal seçme, doğa koşullarına en fazla uyum sağlamış olan canlının neslini devam ettirmesi, uyum sağlayamamış olan türlerin ise elenmesidir. Canlılar nesilden nesile genlerini aktarırken, bu genler de doğal genetik kurallara göre başka genlerle çaprazlanır, değişime uğrar ve yeni genleri oluştururlar. Genetik algoritmalar da tabiattaki bu iki oluşumu birleştirerek, en iyi (optimal nokta)'yı arar. Bir önceki neslin en uyumlu fertleri kullanılarak yeni neslin üyeleri oluşturulur. Bu yöntem sayesinde, klasik yöntemlerle çözülmesi çok zor kimi zaman da imkansız olan problemler çözülebilmektedir (Goldberg, D.E., 1989), (Tang K.S., Man K.F., Kwong S. and He Q., 1996).

Genetik algoritmaları diğer algoritmalardan ayıran en önemli özelliklerden biri de seçmedir. Genetik algoritmalarda çözümün uygunluğu onun seçilme şansını arttırır ancak bunu garanti etmez. Seçim de ilk grubun oluşturulması gibi rasgeledir ancak bu rasgele seçimde seçilme olasılıklarını çözümlerin uygunluğu belirler. Bu çalışmada genetik algoritmaların tanımları yapılmış bu algoritmalarda kullanılan terimleri açıklayan bir sözlük verilmiştir. Temel genetik algoritmaların çalışma prensibi anlatılmış, akış diyagramı verilerek her bir basamakta yapılan işlemler ayrıntısıyla anlatılmıştır.

1.4.7 Çalışmada Kullanmış Olunan Bilgisayar Temelli DC Motor Hız Kontrolünün Gerçekleştirilmesi

Bu çalışmada genetik algoritmalar yardımıyla tasarlanan PI-tip Bulanık kontrolör kullanılarak bilgisayar üzerinden DC motor devir kontrolü yapılmaktadır. Motor devir bilgisi sensör yardımıyla frekans bilgisine ve sonradan gerilime dönüştürülerek bir arabirim vasıtasıyla PC'ye aktarılmaktadır. Burada C Programlama dili yazılan kontrol algoritmasında kullanılarak sistemi verilen referansa değerde tutmak için gerekli kontrol sinyali bulunarak tekrar arabirim üzerinden motora uygulanmaktadır. Tezin bu bölümünde bu çalışmalar için tasarlanmış arabirim devrelerinin şemaları verilerek çalışmaları anlatılmıştır.

1.4.8 Çalışmada Yapılmış Olan DC Motor İçin Genetik Algoritmalar Yardımıyla PI-tip Bulanık Kontrol Sistemi Tasarımı

Bu bölümde geliştirilen öğrenme algoritmasının yapısı, çalışma şekli ve öğrenme sonrası elde edilen kural tabanların gerçek DC motor sistemine uygulanmasıyla alınan sonuçlar verilmiştir. Genetik bulanık öğrenme algoritmasının her nesilde elde edilen tüm kural dizileri motor matematiksel modeli üzerinde simule edilerek en uyumlu kural taban araştırması yapılmaktadır. İstenen performans değerli kural taban bulununcaya kadar veya maksimum nesil sayısına ulaşıncaya kadar algoritma çalıştırılmaktadır. İstenen uyumluluk değerine sahip kural tabanı elde edildikten sonra bu kural tabanı ile Bulanık PI Kontrol yapısına yerleştirilerek direk olarak motor uygulanmaktadır. Bu bölümde yukarıdaki konuların açıklanması yapılmış bu çalışmadan elde edilen sonuçlar verilmiştir.

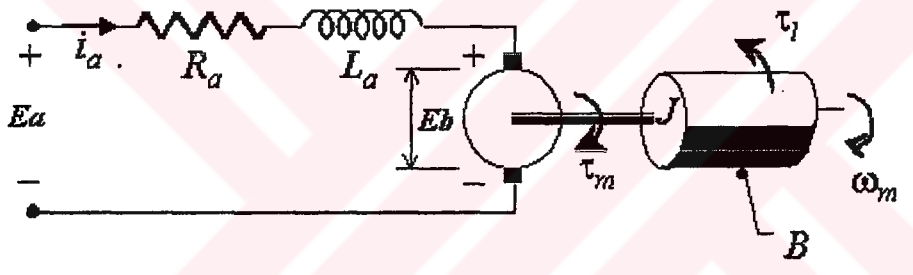
1.4.9 Sonuç ve Öneriler

Bu bölümde bu tez çalışmasından elde edilen sonuçların tartışılması yapılmış ve ileriye yönelik öneriler ve araştırma konuları verilmiştir.

2. DOĞRU AKIM MAKİNALARI

2.1 Giriş

Doğru akım (DC) makinaları uzun süredir hız kontrolünde kullanılmaktadır. Endüstride yaygın olarak kullanılmalarında, çok yönlü kontrol karakteristikleri rol oynamaktadır. DC motorlar büyük kalkış torku sağlayabilmektedir. Nominal hızın altında ve üstünde olmak üzere geniş bir aralıkta hız kontrolü kolayca yapılabilmektedir. Kontrol yöntemleri de alternatif akım (AC) motorlarındakine göre daha basit ve ucuzdur. Kollektörlü olmaları, yüksek hızlarda ve zor şartlarda çalışmalarında engel teşkil etmesine rağmen, DC motorlar endüstride önemli rol oynamaktadır. Şekil 2.1'de miline yük bağlı doğru akım motoruna ait devre şeması verilmektedir (Dubey, Gopal K., ve Kasarabada, C. Rao , 1993)



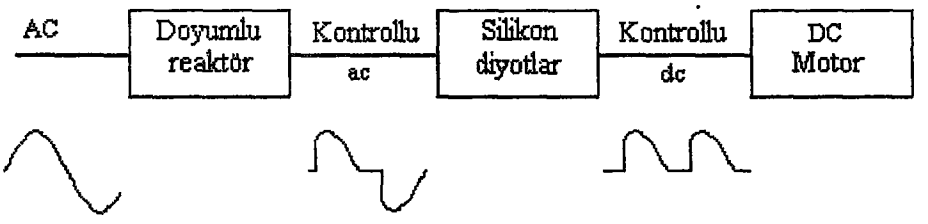
Şekil 2.1 Miline yük bağlı doğru akım motoruna ait devre şeması

2.2 Elektronik Olarak DC Motorun Kontrolü

1940 'ların sonlarında ve 1950 'lerin başlarında, elektronik kontrol DC sürücü sistemlerinde önemli bir gelişmeye neden olmuştur. İlk aşamada, endüstri tipi gazlı doğrultucu ve kontrollü doğrultucu (thyratron) tüpler, motor-jeneratör (M-G) gurubundaki uyarma sargılarında ve regülatörlerde kullanılmıştır. Bu sistem daha iyi cevap ve daha fazla doğruluk sağlamış, otomatik kapalı çevrim kontrolün daha önceki manuel veya elektromekanik kontrol yöntemlerinin yerine kullanılmasına da imkan tanımıştır. Daha sonra, bu tüpler akım kapasitelerinin artmasıyla DC sürücü motorların hız kontrolünde alternatif akımı doğru akıma çevirmek amacıyla doğrultucu devrelerde kullanılmıştır. Tüplü doğrultucuları kullanarak yapılan tek fazlı doğrultucu devreler düşük güçlü sürücülerde geçerli olmuştur. 10 HP (Horse Power) 'nin üstündeki değerlerde, tüplü doğrultucu sistemler güvenilir olmamıştır.

1950 'lerin sonlarında yarıiletken elemanlar, silikon diyotlar ve silikon kontrollü doğrultucular

(SCRs) endüstride kullanılmaya başlamıştır. Bu elemanlar öncelikle düşük güçlü sistemlerde geçerli olmuştur ve bunun neticesinde tüplü regülatörlerin yerine jeneratör alan regülatörlerinde kullanılmıştır. Yariletken regülatörleri uzun ömür, yüksek güvenirlik, daha küçük bir yapı ve performans olarak da önemli bir gelişme ortaya koymuştur.



Şekil 2.2 Doyumlu reaktörlü, hız ayarlı, DC sürücü sistemin blok şeması ve dalga şekilleri.

Daha sonraları silikon diyotların ve tristörlerin (SCRs) yüksek güçlerde üretilmesiyle birlikte bu elemanlar, sürücü motorların direkt kontrolünde AC 'ı DC 'a çevirmek için kullanılmıştır. İlk olarak yüksek güçlü diyotlar geliştirilmiş ve Şekil 2.2'de görüldüğü gibi, hız kontollü DC sürücülerde doyumlu reaktörlerle birlikte kullanılmıştır. M-G grubuna kıyasla, doyumlu reaktörlü sürücü sistemler sağlam, aynı zamanda daha güvenilir olmuşlar ve daha iyi performans sağlamışlardır.

1960 'ların başında, yüksek güçlü tristörlerin kullanılması endüstriyel kontrol donanımında ve sürücü sistem performansında küçük bir devrime yol açmıştır. 1960 'lı yıllar boyunca, yüksek kapasiteli değişken gerilimli DC kaynak sistemlerinin tasarımını yapan mühendislerin dikkatleri bir M-G grubu kullanarak güç üretmekten tristörleri kullanarak güç çevirmeye yöneltilmiştir (Dubey, Gopal K., ve Kasarabada, C. Rao, 1993). Şekil 2.3'da bir tristör kontrollü DC sürücünün blok şemasını göstermektedir. Uzun süredir hız kontrollü DC sürücülerde kullanılan M-G grubu geniş ölçüde tristörlü çeviricilerle yer değiştirmiştir.

Tristörlü sürücünün avantajları;

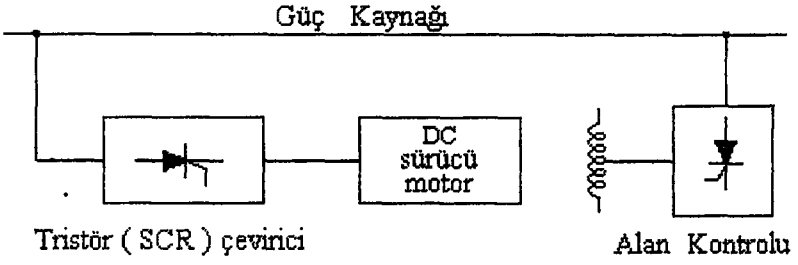
1. Tristörlü güç modülü, jeneratör alan ve armatürün elektriksel zaman gecikmesini ortadan kaldırır.

1. Temel çalışması basit ve güvenilirdir.

2. Minimum bakım gerektirir. Çalışma verimi yüksektir (%95 'in üstünde).

3. Küçük boyuta, hafifliğe ve paketleme esnekliğine sahip olması, daha az kurulma ve işletme

maliyetini, daha az yeri gerektirir.



Şekil 2.3 Tristör kontrollü DC sürücünün blok şeması dalga şekilleri.

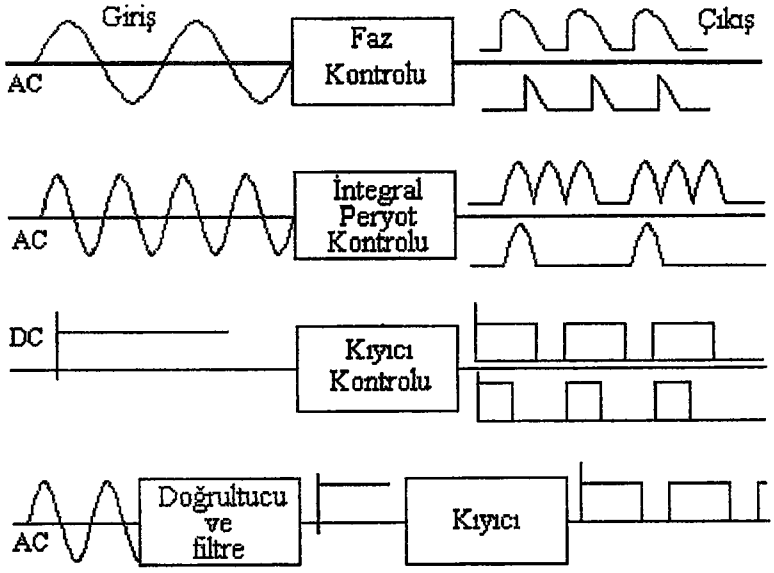
Tristörlü sürücünün dezavantajları;

1. Çevirici çıkışının yüksek dalgalılık içermesi, motorda ısınma ve komutasyon problemleri oluşturur. Armatür devresine bir bobin ilave ederek akımdaki dalgalılığı düzeltmek mümkündür.
2. Aşırı yüklenme kapasitesi M-G grubuna göre daha düşüktür.
3. Tristörlerin anahtarlanması sebebiyle, AC kaynak geriliminde bozulma ve telefon parazitleri oluşabilir.

2.3 DC Motor Sürücü Teknik Özellikleri

Günümüzde tristörlü çeviricilerle kontrol edilen serbest uyarmalı DC motorları, endüstride çok yaygın kullanıma sahip sürücü sistemlerdir. Bu sistemler çok geniş bir aralıkta hız kontrolü sağlar. Nominal hızdan aşağı doğru hız ayarı armatür geriliminin kontrol edilmesiyle yapılır. Nominal hız üstündeki hız kontrolü ise alanı zayıflatmak suretiyle elde edilir. Güç ve hız sınırları DC motorları tarafından belirlenir, yarıiletken elemanların bir etkisi yoktur. Tristörler (SCRs) yüksek gerilim ve akımlarda seri ve paralel bağlanabilirler. Armatür akımı ve tork sınırları motor sıcaklık artışı tarafından belirlenir.

Şekil 2.3 'deki blok diyagram ile gösterilen tristörlü çevirici, sürücü motora değişken armatür gerilimi sağlamaktadır. Sabit gerilim kaynağından değişken bir DC çıkış gerilimi elde etmek için kullanılan üç temel yöntem (faz kontrolü, integral peryot kontrolü ve kıyıcı kontrolü) Şekil 2.4 'da gösterilmiştir. Bütün bu yöntemlerde, tristörler kaynak ile motor arasındaki elektriksel bağlantıyı sağlar ve keser. Anahtarlama frekansı hızlıdır. Bunun neticesinde motor, gerilim darbelerine değil ortalama çıkış gerilim seviyesine cevap verir .



Şekil 2.4 Sabit bir gerilim kaynağından değişken dc çıkış gerilimi elde etmek için kullanılan kontrol teknikleri.

Faz ve integral peryot kontrol uygulamalarında, ac 'yı dc'ye dönüştürme işlemi doğrultmak suretiyle yapılır. Faz kontrolünde, tristör peryodun belirli bir süresinde kaynağı motora bağlar ve peryodun geri kalan kısmında kaynağı motordan ayırır. İntegral peryot kontrolünde ise, kaynak birkaç yarım peryot süresince motora bağlanır ve birkaç yarım peryot süresince de kaynak motordan ayrılır. Kıyıcı ile kontrol uygulamasında, tristör hızlı bir şekilde anahtarlanır ve sürücü motora kıyılmış bir gerilim sağlanır. Kıyılmış gerilimin ortalama değeri tristörün iletimde ve kesimde kalma sürelerini ayarlamak suretiyle kontrol edilir. Eğer kaynak gerilimi AC ise bir doğrultucu kıyıcı grubu da kullanılır.

Faz ve integral peryot kontrolünde, AC kaynak gerilimi tristörleri kesime götürür, yani doğal komutasyon vardır ve özel bir komutasyona gerek yoktur. Bu uygulamalar bu nedenle basit ve ucuzdurlar. Faz kontrolü yaygın olarak kullanılmaktadır. Bu kontrol yönteminde, çıkış gerilimi geniş bir aralıkta düzgünce ayarlanabilmektedir. Buna karşın daha düşük çıkış gerilimlerinde AC tarafında giriş güç faktörü düşer. Kaynak frekansı yüksekse integral peryot kontrolü başarılı olmaktadır. Aksi takdirde motor ortalama hızı civarında osilasyon yapacaktır. Bundan dolayı motor hız kontrolünde bu yöntem tercih edilmemektedir. Eğer kaynak gerilimi DC ise kıyıcı devreler kullanılır. Kıyıcılar, tristörleri kesime götürmek için yardımcı devrelere ihtiyaç duyarlar ve motor akımındaki dalgalılığı azaltmak içinde yüksek anahtarlama frekansında çalışırlar. Yüksek hızlı anahtarlama da özel tristörler kullanmayı gerektirir. Bu nedenle kıyıcı

kontrolü duruma göre karmaşık olabilmektedir, fakat buna karşın yaygın olarak kullanılmaktadır. Doğrultucu ve kıyıcı grubundan oluşan kontrol yöntemi pahalıdır, fakat girişte diyot köprü kullanılması sebebiyle yüksek giriş güç faktörü avantajına sahiptir (Roach,H. P. ve Anderson, P. H., 1996).

Hız sürücü sistemlerinde frenleme çoğu uygulamalarda önemlidir. DC motoru durdurmak için genellikle sürtünme frenlemesi kullanılmaktadır. Sık sık frenleme yapmak gerekiyorsa, peryodik bakım ve fren ayaklarının değiştirilmesi sürtünme frenlemesini ekonomik ve tercih edilebilir olmaktan çıkarmaktadır. Tristör kontrolünün tüm özelliklerini kullanmak amacıyla, dinamik ve generatör olmak üzere her iki frenleme türü çoğu tristör kontrollü DC sürücülerde günümüzde yaygın olarak kullanılmaktadır. Generatör frenlemede, sürücü sistemin kinetik enerjisi elektrik enerjisine dönüştürülür. Dinamik frenlemede elektrik enerjisi bir gurup frenleme direncinde tüketilir, oysa generatör frenlemede elektrik enerjisi kaynağa geri verilir. Bundan dolayıdır ki, bu tip sürücü sistemler daha verimlidir. Çoğu taşıma araçlarında ve akü ile çalışan elektrikli otomobillerde itici güç bir tristörlü çeviriciden sağlanmaktadır ve generatör frenleme kullanılır (Dubey, Gopal K., ve Kasarabada, C. Rao , 1993).

Tristörlü DC sürücüler sık sık karmaşık kontrol sistemlerine ihtiyaç duyarlar. Analog ve sayısal (dijital) geri beslemeli kontrol olmak üzere her iki sistem de kullanılmaktadır. Faz-kilitlemeli kontrol teknikleri, temel olarak sıfır hız ayarı ve tam hız kontrolü sağlamak için bazı DC sürücülerde kullanılır.

DC sürücülerin tristör kontrol teknikleri son yıllarda hızlı bir şekilde gelişmiştir. Günümüzde, kontrol amaçlı ve hızlı çalışan mikrokontrolörlerin gelişmesi ve bunların sürücü sistemlerde kullanılmasıyla büyük bir çalışma esnekliği elde edilmiştir .

Son yıllarda mikroişlemci teknolojisindeki hızlı gelişmelerle birlikte yüksek hızlı, çok fonksiyonlu, daha güçlü ve kontrol amaçlı üretilen mikrokontrolörler ön plana çıkmış durumdadır. Mikrokontrolörlerin daha fazla fonksiyon sağlaması, sistem maliyetlerini düşürmesi ve daha küçük sistem yapısına imkan tanıması sebebiyle, tasarım mühendisleri de sayısal (dijital) kontrol sistemlerinde mikrokontrolörleri oldukça fazla kullanmaktadırlar.

Analog kontrol yöntemlerinin bazı dezavantajları vardır. Bunlar analog işaretlerin iletiminde ortaya çıkan güçlük, sıcaklık nedeniyle oluşan hatalar, elemanların yıpranması, analog elemanlarda oluşan kayma ve sapmalar, dağılmış bozucu etkiler gibi özelliklerdir. Bir sayısal (mikrokontrolörlü) kontrol sisteminde ise bu dezavantajlar yoktur. Sayısal bir kontrol

uygulamasında hız bilgisi, frekansı motor hızıyla orantılı olarak değişen darbe dizilerini üreten bir sayısal takometreden elde edilebilmektedir. Darbe dizileri bir sayıcıya verilmek suretiyle sayısal bilgiye dönüştürülerek bu bilgi hız bilgisi olarak mikrokontrolöre verilmektedir. Sistem gerilimi ve akımıyla ilgili analog işaretler, A/D (analog-dijital) çeviriciler kullanılarak sayısal bilgi şekline getirilerek bu bilgiler mikrokontrolör tarafından alınmaktadır. Elde edilen bilgiler bir kontrol algoritmasından geçirilerek sürücü devredeki anahtarlama elemanları için gerekli kontrol işaretleri mikrokontrolör tarafından üretilmektedir. Aynı zamanda sayısal kontrol sistemlerinde ölçülemeyen büyüklükler, ölçülen büyüklükleri kullanarak yapılan hesaplamalar sonucunda elde edilebilmektedir.

Bazı modern kontrol yöntemlerini, analog devrelerle gerçekleştirmek çok zor olmakta veya mümkün olmamaktadır. Bu gibi durumlarda mikrokontrolörlerin kullanılması büyük kolaylık ve çalışma esnekliği sağlamaktadır. Bir tasarımcı için donanım yerine yazılımı değiştirmek suretiyle kontrol sisteminde değişiklik yapmak önemli bir özelliktir. Bu şekilde aynı sistem için değişik kontrol algoritmalarını uygulamak mümkün olabilmektedir (Maiocchi G., 1995).

2.4 Serbest Uyarmalı Doğru Akım Motoru

Serbest uyarmalı DC motorunda temelde birbirinden bağımsız iki sargı bulunmaktadır. Bunlardan birisi manyetik alanın oluşmasını sağlayan ve statorda bulunan uyarma sargısıdır. Diğeri ise rotora yerleştirilmiş, kollektör ve fırçalar yardımıyla beslenerek magnetik-motor-kuvvetini oluşturan sargıdır. Burada iki sargının birbirinden bağımsız beslenmesi kontrol açısından kolaylık sağlamaktadır.

Günümüzde DC motorları AC motorlarına göre çok yaygın olmamalarına karşın, bazı kesin uygulamalar için daha uygun olmaktadırlar. DC motorları değişken karakteristiklere sahiptirler ve değişken yapıdaki sürücülerde geniş bir şekilde kullanılmaktadırlar. Bu motorlar yüksek bir kalkış torku sağlayabilmekte ve geniş bir aralıkta hız kontrolüne de imkan tanımaktadır. Hız kontrol yöntemleri, AC sürücülerine göre daha basit ve ucuz olmaktadır. DC motorları, kollektörlü olmaları sebebiyle yüksek hızlı uygulamalar için uygun olmamakta ve AC motorlarına göre daha fazla bakım gerektirmektedir. Buna karşın elektronik cihazlarda soğutma amaçlı fanları döndüren küçük ünitelerden haddehanelerde tamburları süren 10000 HP 'lik motorlara kadar, çatallı kaldırıcılar, delgi ve zımba mengeneleri, akü ile çalışan elektrikli araçlar gibi birçok uygulamalarda kullanılmaktadırlar.

Armatür ve alan sargılarının ayrı kaynaklardan beslendiği serbest uyarmalı DC motoru

vasıtasıyla çok esnek kontrol yapılabilmektedir. Bu düzenleme motorun hız tork karakteristiğini hemen hemen ideal bir karakteristiğe yaklaştırmaktadır. Armatür gerilimi kontrollü bir doğrultucu veya bir kıyıcıdan kontrol edilebilmektedir. Eğer uyarma akımı kontrol edilmek istenirse, benzer düzenleme geçerli olabilmektedir (Zhang W., 1992).

2.5 Serbest Uyarmalı Doğru Akım Motorun Matematiksel Modeli

Karmaşık sistemleri anlamak ve kontrol etmek için, bu sistemleri tanımlayan matematiksel modellerin oluşturulması gerekmektedir. Bundan dolayı sistem değişkenleri arasındaki ilişkilerin analiz edilmesi ve matematiksel modelinin de bulunması gerekmektedir. Gerçekte incelenen sistemler dinamik yapılardır ve bunları tanımlayan denklemlerde genellikle diferansiyel denklemlerdir.

Değişkenler bazı sınırlar içinde olduklarında fiziksel sistemlerin büyük bir çoğunluğu lineerdir. Buna karşın değişkenler sınırsız arttırıldığı için bütün sistemler eninde sonunda lineer olmamaktadır. Pratikte, sistemlerin karmaşıklığı ve ilgili faktörlerin ihmal edilmesi, sistemin çalışmasını ilgilendiren varsayımların yapılmasını zorunlu kılmaktadır. Bundan dolayı bilim adamları da, fiziksel sistemi incelemek ve onu lineerleştirmek için bazı gerekli varsayımların yapılmasını faydalı bulmaktadırlar. Bunun neticesinde, lineer eşdeğer sistemi tanımlayan fiziksel kuralları kullanarak bir grup lineer diferansiyel denklem elde edilebilmektedir. Sonuçta, bazı matematiksel işlemlerden faydalanarak sistemin çalışmasını tanımlayan bir çözüm bulunabilir. Burada bir DC motorunun matematiksel modelini elde etmek için, fırçalardaki gerilim düşümleri ve histerizis gibi ikinci dereceden etkilerin ihmal edilmesi faydalı olmaktadır. Sistemi lineerleştirmek ve sistemin karmaşıklığından kaçınmak için bu varsayımların yapılması gerekmektedir.

Doğru akım motorları kontrol sistemlerinde yaygın olarak kullanıldığı için matematiksel modelinin çıkartılması analitik çözümleri kolaylaştırmaktadır. Aşağıda serbest uyarmalı DC motorunun sembolik gösterimi ve matematiksel modeli verilmiştir. Bu lineerleştirilmiş modelde alan sargısı tarafından üretilen magnetik akı yoğunluğunun alan sargısı akımı ile lineer değiştiği kabulü yapılmıştır (Dubey, Gopal K., ve Kasarabada, C. Rao , 1993).

DC motorunu tanımlayan matematiksel ifadeler aşağıdaki gibidir:

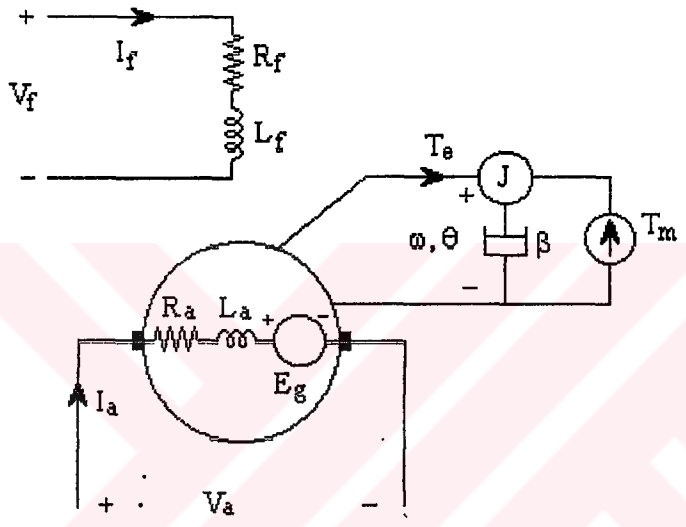
$$v_f(t) = R_f i_f(t) + L_f \frac{di_f(t)}{dt} \qquad (2.1)$$

$$v_a(t) = R_a i_a(t) + L_a \frac{di_a(t)}{dt} + E_g \tag{2.2}$$

$$T_e(t) = J\frac{d\omega(t)}{dt} + \beta\omega(t) + T_L \tag{2.3}$$

$$E_g = G_f i_f \omega(t) \tag{2.4}$$

$$T_e = G_f i_f i_a(t)$$



Şekil 2.5 Serbest uyarmalı DC motorunun sembolik gösterimi.

E_g : Armatürde (armatürde) endüklenen gerilim.

T_e : Motorda endüklenen tork.

T_L : Yük torku.

J : Motorun atalet torku

β : Sürtünme katsayısı.

R_a : Armatür sargısının direnci.

R_f : Uyarma (Alan) sargısının direnci.

L_a : Armatür sargısının özendüktansı.

L_f : Uyarma sargısının özendüktansı.

G_f : Hız gerilim katsayısı.

$w_m(t)$: Motorun açısal hızı.

I_a : Armatür akımı.

I_f : Uyarma akımı

Serbest uyarmalı DC motorunun Şekil 2.5'deki eşdeğer devresi kullanılarak motoru tanımlayan matematiksel eşitlikler yukarıda verilmiştir. Bu tanım bağıntılarının uygun şekilde düzenlenmesiyle DC motorunun durum denklemleri elde edilir. Genel durum ve kontrol vektörleri, durum denklemleri aşağıdaki gibidir .

$$X_T = \left[i_a(t) \, , \, i_f(t) \, , \, \omega(t) \right] \tag{2.6}$$

$$U = \left[v_a(t) \, , \, v_f(t) \, , \, T_L(t) \right] \tag{2.7}$$

$$\frac{d}{dt}\begin{bmatrix} i_a(t) \\ i_f(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & 0 & -\frac{G_f i_f(t)}{L_a} \\ 0 & -\frac{R_f}{L_f} & 0 \\ -\frac{G_f i_f(t)}{J} & 0 & -\frac{\beta}{J} \end{bmatrix} \begin{bmatrix} i_a(t) \\ i_f(t) \\ \omega(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} & 0 & 0 \\ 0 & \frac{1}{L_f} & 0 \\ 0 & 0 & -\frac{1}{J} \end{bmatrix} \begin{bmatrix} v_a(t) \\ v_f(t) \\ T_L \end{bmatrix} \tag{2.8}$$

Yukarıdaki denklemler,

$$\frac{dX}{dt} = AX + BU \tag{2.9}$$

şeklinde de gösterilebilir. Burada

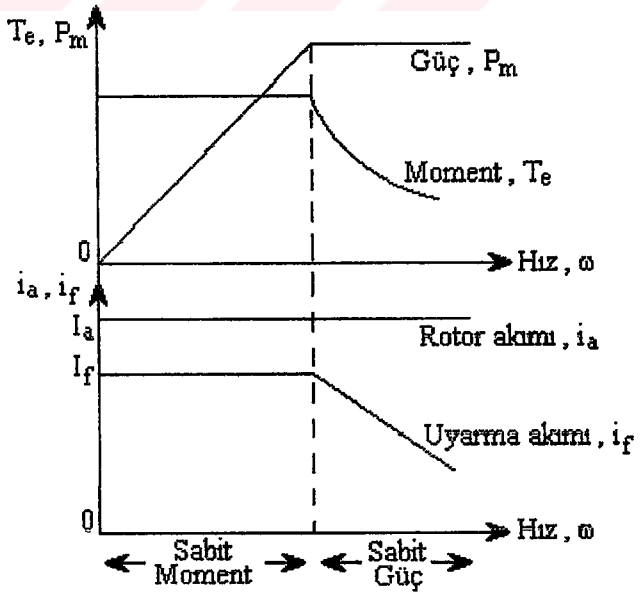
$$A = \begin{bmatrix} -\frac{R_a}{L_a} & 0 & -\frac{G_f i_f(t)}{L_a} \\ 0 & -\frac{R_f}{L_f} & 0 \\ \frac{G_f i_f(t)}{J} & 0 & -\frac{\beta}{J} \end{bmatrix} \, , \quad B = \begin{bmatrix} \frac{1}{L_a} & 0 & 0 \\ 0 & \frac{1}{L_f} & 0 \\ 0 & 0 & -\frac{1}{J} \end{bmatrix} \tag{2.10}$$

olmaktadır. Görüldüğü gibi A matrisi durum değişkenlerine bağlı elemanlardan oluşmaktadır.

2.6 Doğru Akım Motorunun Kontrolü ve Blok Diyagramı

Otomatik kontrol sistemlerinde çok yaygın bir biçimde kullanılmaları nedeniyle doğru akım motorlarının lineer çalışma halleri için transfer fonksiyonlarının bilinmesi yararlı olmaktadır. Kontrol sistemlerinde çoğunlukla serbest uyarmalı DC motorları kullanılır. Bu motorlar ya rotor (armatür) gerilimleri kontrol edilerek ve uyarma gerilimleri sabit tutularak kullanılır, ya da sabit rotor geriliminde uyarma gerilimi kontrol edilerek kullanılır. Bu sonuncu durumda lineer çalışma sağlamak için rotor akımı da sabit bir akım kaynağından elde edilebilir (Sarıoğlu, M. K., 1991).

Pratikte, nominal hızın altındaki hızlar için rotor akımı ve uyarma akımı sabit tutulur, hızı kontrol etmek için de rotor gerilimi değiştirilir. Nominal hızın üstündeki hızlar için ise, rotor gerilimi nominal değerinde sabit tutulur ve hızı kontrol etmek için uyarma akımı yani gerilimi değiştirilir. Rotor gerilimi ile kontrolde tork, uyarma gerilimi ile kontrolde de güç sabit kalır. Şekil 2.6'de hız ile tork, güç, rotor akımı ve uyarma akımının nasıl değiştiğini göstermektedir.



Şekil 2.6 Serbest uyarmalı doğru akım motorunun karakteristiği.

Uyarma gerilimi ile kontrolün en büyük avantajı, hızın sürekli ve ekonomik bir şekilde

ayarlanabilmesidir. Çünkü kontrol elemanlarının fiyatlarını ve boyutlarını etkileyen küçük bir akım uyarma devresinden akar. Kontrol için gerekli olan uygun kontrollü bir gerilim tristörlü veya transistörlü çeviricilerden elde edilebilmektedir. Kontrollü doğrultucular sabit bir AC gerilimden değişken bir DC gerilim sağlar. Oysa kıyıcılar sabit bir DC gerilimden değişken bir DC gerilim sağlar. Genellikle DC motorlarının hız kontrolleri için kontrollü doğrultucular kullanılmaktadır.

2.6.1 Rotor Gerilimi İle Kontrol

Kontrol sistemlerinde, rotor geriliminin değiştirilmesi ile DC motorunun hız kontrolü çok kullanılmaktadır. Motor için gerekli olan kaynak gerilimi çok kullanılan ve güç kazançları çok büyük olan tristörler (yarıiletken kontrollü diyotlar) kullanılarak elde edilmektedir. Bu elemanlar, kapılarına uygulanan işaretlerle alternatif gerilimden ayarlı doğru gerilim sağlarlar. Motorun zaman domeni dinamik denklemleri V_f =sabit olduğundan

$$v_f(t) = V_f = I_f R_f = \text{sabit} \tag{2.11}$$

$$v_a(t) = R_a i_a(t) + L_a \frac{di_a(t)}{dt} + E_b \tag{2.12}$$

$$E_b = K_b i_f \omega_m(t) \ , \ \omega_m = \frac{d\theta(t)}{dt} \ , \ T_e = K_i \cdot i_a(t) \tag{2.13}$$

$$T_e(t) = J \frac{d^2\theta(t)}{dt} + \beta \frac{d\theta(t)}{dt} + T_L \tag{2.14}$$

olur.

$w_m(t)$: motor hızı, rad/s

$I_a(t)$: Armatür akımı, Amper

v_a : Motor giriş gerilimi, volt

Ra: Armatür direnci

La: Armatür İndüktansı

Kb: Geri Besleme Sabiti

Ki: Tork Sabiti

J: Başlangıç (Atalet) Torku

B: Sürtünme Sabiti

T_L : Yük torku

Durum uzayı denklemleri yukardaki eşitsizlikler kullanılarak çıkarılacak olursa,

$$x = [w_m(t) \quad I_a(t)]^T$$
$$y = [w_m(t)] \tag{2.15}$$
$$u = [v_a \quad T_L]^T$$

olarak tanımlanırsa,

x, y, u parametreleri durum denklemlerinde yarine konulursa

$$\mathbf{x} = \mathbf{A}\,\mathbf{x} + \mathbf{B}\,\mathbf{u} \tag{2.16}$$

$$\mathbf{y} = \mathbf{C}\,\mathbf{x} + \mathbf{D}\,\mathbf{u}$$

$$A=\begin{bmatrix} -\frac{B}{J} & \frac{K_i}{J} \\ \frac{K_b}{L_a} & -\frac{R_a}{I_a} \end{bmatrix}, \qquad B=\begin{bmatrix} 0 & \frac{-1}{J} \\ \frac{1}{L_a} & 0 \end{bmatrix}, \qquad C=[1 \quad 0], \qquad D=[0 \quad 0] \tag{2.17}$$

parametreleri bulunur.

Motor dinamik denklemlerinde bütün başlangıç koşulları sıfır alınır ve Laplace dönüşümleri yazılırsa,

$$V_a(s) = R_a I_a(s) + sL_a I_a(s) + K_b I_f W(s) \tag{2.18}$$

$$K_i I_a(s) = (Js + \beta)W(s) + T_L(s) \tag{2.19}$$

bulunur. $T_L(s)$ yük torku Laplace dönüşümü ve $V_a(s)$ rotor gerilimi giriş büyüklüğü olarak alınırsa (2.18) ve (2.19) denklemlerini Şekil 2.7 de verilen blok diyagramı ile göstermek mümkün olur.

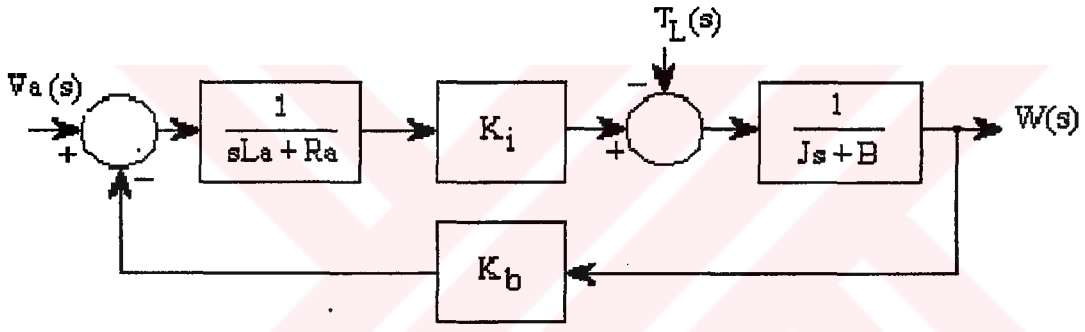
Yük torkunun (T_L) sıfır olması durumunda $V_a(s)$ giriş ve $W(s)$ çıkış alınmak üzere motorun transfer fonksiyonu aşağıdaki gibi olur.

$$G(s)=\frac{w_m(s)}{v_a(s)}=\frac{K_i}{s^2J\cdot L_a+s(B\cdot L_a+J\cdot R_a)+B\cdot R_a+K_b\cdot K_i} \tag{2.20}$$

Eğer yük torku sıfırdan farklı olarak ele alınırsa, bu durumda motor hızı aşağıdaki gibi olur.

$$w_m(s)=\frac{K_i}{s^2J\cdot L_a+s(B\cdot L_a+J\cdot R_a)+B\cdot R_a+K_b\cdot K_i}v_a(s)+\frac{R_a+sL_a}{(R_a+sL_a)\cdot(J+sB)+K_b\cdot K_i}T_L \tag{2.21}$$

Armatür kontrollu DC motorun blok diyagramı, motora ait transfer fonksiyonu denklemi denk.(2.21) 'e göre Şekil 2.7 deki gibi olur.



Şekil 2.7 Mil üzerinde J, β, T_L elemanları bulunan, uyarma akımı sabit olan ve V_a rotor gerilimi ile kontrol edilen serbest uyarmalı doğru akım motorunun blok diyagramı

2.6.2 Uyarma Akımı İle Kontrol

Birçok kontrol uygulamasında sadece uyarma akımı değiştirilerek hız kontrolü yapmak yeterli olabilmektedir. Ancak, bu kontrol biçiminde w, i_f ve i_a değişken olduğundan (2.18) denkleminden görüldüğü gibi, (i_f , w) ve (i_a , i_f) çarpımları denklemlerin lineer olmayan denklemler biçimine dönüşmesine yol açar. Bu halde transfer fonksiyonu tanımlanamaz ve denklemler ancak bilgisayar yardımı ile çözülebilir. Buna karşılık yalnız lineerlik sağlamak amacı ile pratikte doğru akım motorunun armatürünün (rotorunun) sabit akım kaynağı ile beslendiği kabul edilir. O halde I_a= sabit olur ve rotor için gerilim denklemi yazma zorunluluğu ortadan kalkar. Motorun uyarma ve mekanik kapısına ilişkin denklemler,

$$v_f(t)=R_fi_f(t)+L_f\frac{di_f(t)}{dt}\quad,\ (T_L=0) \tag{2.22}$$

$$G_f I_a i_f(t) = J\frac{d^2\theta(t)}{dt} + \beta\frac{d\theta(t)}{dt} \quad , \quad i_a(t) = I_a = \text{sabit} \tag{2.23}$$

olarak yazılır. w çıkış ve v_a giriş olarak alınırsa, ilk koşullar sıfır olmak üzere transfer fonksiyonu için

$$G(s) = \frac{W(s)}{V_f(s)} = \frac{G_f I_a}{(sJ+\beta)(sL_f + R_f)} \tag{2.24}$$

olarak bulunur. Bu transfer fonksiyonu normalize edilmiş biçimde

$$G(s) = \frac{K_m}{(T_{m1}s+1)(T_{m2}s+1)} \tag{2.25}$$

$$K_m = \frac{G_f I_a}{\beta R_f} \quad , \quad T_{m1} = \frac{J}{\beta} \quad , \quad T_{m2} = \frac{L_f}{R_f} \tag{2.26}$$

olarak yazılır. Eğer θ çıkış olarak alınırsa transfer fonksiyonu

$$G(s) = \frac{\theta(s)}{V_f(s)} = \frac{K_m}{s(T_{m1}s+1)(T_{m2}s+1)} \tag{3.29}$$

olur.

3. DİJİTAL KONTROL SİSTEMLERİ

3.1 Otomatik Kontrol Tarihçesi ve Günümüzdeki Durum

Çok eski çağlardan beri insan oğlunun çok basitte olsa "kendi kendine çalışma sistemine" göre hareket eden "otomatik makinalar" yaptığı bilinmekteydi. Bu aletler daha çok hayvan taklitleri yapan oyuncak biçimi aygıtlar (M.Ö 430 yılında Taretumlu Achytasın yaptığı otomatik güvercin) veya zamanı saptamaya sulama sistemini düzenlemeye yarayan aygıtlar (İskenderiyeli Heronun (Ktesbios) 285-247 M.Ö yaptığı su otomatları) şeklinde özellikle saray ve hükümdarlar için kullanışlılık arz den aygıtlardır. Bu konudaki bilimsel çalışmanın kesin olarak ne zaman başladığı bilinmemektedir. Fransızlar bu çalışmayı 17. y.y da yaşamış Descardes ve Raseal ile başlatmakta; Almanlar aynı y.y de yaşamış Leibnitz üzerinde durmaktadır. İngilizler ise daha eski tarihlerde yaşamış olan Roger Baconun bu sistemlerin ileri sürmüş olduğunu söylemektedirler. Bizde ise günümüzden sekiz y.y önce 1205 lerde yaşamış Cizreli Eb-ül-iz adında bir Türk bilginin Diyarbakır'da "otomatik makinalar" yapmış olduğu ve bu makinaların saraylarda kullanmış olduğu söylenmektedir. Batı dünyasında "al Carazi" olarak ta bilinen Cizrali Eb-ül-iz İbni İsmail İbni Razzaz adındaki Türk bilgini su saatleri , içki meclislerinde kullanılan kaplar ve oyunlar, ibrikdarlık yapan ve kan toplamaya yarayan düzenler fiskiyeler ve müzik otomatları kuyu yada akarsulardan su çıkaran tulumbalar ve kaldırma düzenleri üzerinde çizim ve tasarımlar yapmış ve bunların imalatını gerçekleştirdiği ileri sürülmektedir. Bugün için otomatik kontrol kuramı hakkında bir bilgi veya çalışma olmadığından bu aygıt tarihte bir teknolojik uygulama olarak yer almıştır. Otomatik kontrol kuramı üzerindeki ilk çalışmalar 1920'li yıllarda Minorsky, Kazan ve Nyquist tarafından gerçekleştirilmiştir.

Otomatik Kontrol, genel olarak otomatik kontrol doğrudan insan müdahalesi olmaksızın çalışan aygıtların sistemlerin ve fabrikaların gelişmeleri ile ilgilenen bir bilim ve teknoloji dalı olarak tanımlanır. Kısaca bir sistemde faaliyetlerin insan müdahalesi olmaksızın önceden belirlenen bir amaca göre denetimi ve yönetilmesi olarak ifade edilir. Günümüzde amacına uygun bir enerji kullanımı istendiğinde bir takım kontrol birimleri gerekmektedir. İlk zamanlarda uzay-araçları, güdümlü-füzeleri, otomatik uçuş ve seyir sistemleri ve askeri alanlarında aşırı öneme sahip olan otomatik kontrol sistemleri günümüzde çağdaş imalat endüstrilerinin (makina-takım tezgahları) ve endüstriyel işlemlerin çok önemli ve tamamlayıcı bir parçası haline gelmişlerdir. Örneğin Proses endüstrilerinde sıcaklığın, basıncın, debinin v.s. denetiminde ve imalat endüstrilerinde takımla işleme, test, ayırma ve mekaniksel parçaların montajında otomatik kontrol şart olmaktadır.

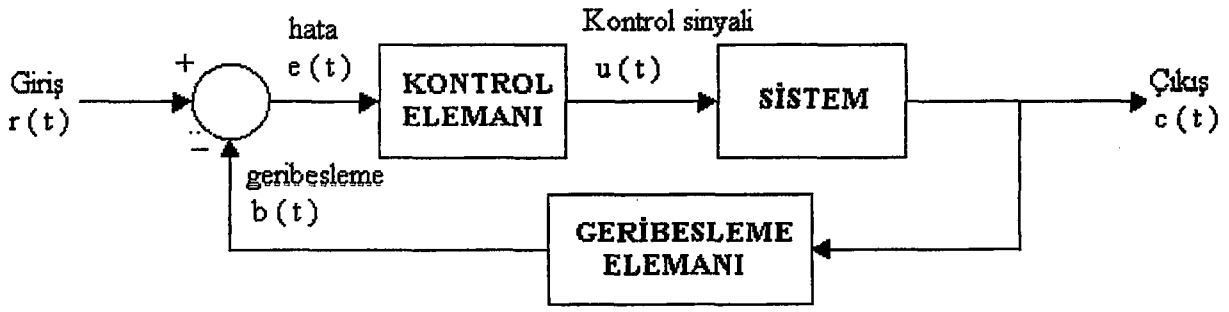
Otomatik kontrol sistemleri hemen hemen tüm çağdaş endüstrilerin tamamlayıcısı ve vazgeçilmez bir parçası haline geldiğinde farklı üretimlerde bulunan işletmelerde çalışacak olan makina, kimya, elektrik, elektronik, tekstil v.s. olmak üzere tüm mühendisleri az çok, doğrudan veya dolaylı olarak ilgilendirmektedir. Otomatik kontrol sistemlerinde kullanılan teknikler temelde daha çok doğrudan doğruya elektrik, elektronik ve makina mühendisliğini ilgilendirmektedir. Bir kontrol organının ilk kademelerinde elektriksel aygıtlar en iyi çözümü sağlamakta beraber çalışma hareketi sağlayan en son kademelerinde daha çok güç yoğunlukları yüksek olan mekaniksel, pnömatik ve hidrolik aygıtlar kullanılmaktadır. Diğer taraftan endüstriyel işlem kontrol sistemlerinde örneğin damıtma tesisleri v.s. sıcaklığın, PH, kimyasal konsantrasyon denetiminde işlemde meydana gelen reaksiyonlar açısından kimya mühendisi konu ile doğrudan ilgili olup kontrol organları açısından da konudan dolaylı bilgisi olması gerekir .

3.2 Açık Çevrim ve Kapalı Çevrim Kontrol Sistemleri

Bir kontrol sistemi, arzu edilen sistem cevabını sağlayacak bir yapıyı oluşturmak üzere elemanların bir araya getirilmesiyle meydana gelmektedir. Temel amaç, sistemde istenen bir dinamik değişkenin önceden belirlenmiş bir yörüngeye veya değere yakın kalmasını sağlamaktır. Bu amaç etrafında oluşturulan bir kontrol sistemi iki ana blok içerir. Bunlardan birisi kontrol edilen sistem, diğeri bu sistemin düzgün çalışmasını sağlamak üzere giriş (referans) işaretini işleyen kontrolcu bloğudur. Çıkış kontrol edilen değişkendir. Şekil 3.1 'deki sisteme, bir çevrim bulunmadığı için açık-çevrim kontrol sistemi denir. Burada girişten çıkışa direkt bir iletim vardır ve regülasyon yapılmamaktadır.

Şekil 3.1. Açık-çevrim kontrol sistemi.

Açık çevrim kontrol sisteminden farklı olarak, referans (arzu edilen çıkış) işareti ile gerçek çıkış işaretini karşılaştırmak amacıyla kapalı-çevrim bir kontrol sisteminde çıkışı ölçmek için ilave bir ölçme düzeni bulunmaktadır. Basit bir kapalı-çevrim geribeslemeli kontrol sistemi Şekil 3.2.'de görülmektedir. Genel olarak iki önemli nokta vardır. Birincisi çevrim kapalıdır ve regülasyonu sağlamak için geribesleme bilgisi kullanılmaktadır. İkincisi, kontrolcunun gerekli ayarlamaları yapıldıktan sonra araya girmeksizin sistemin kendisi çalışabilmektedir.



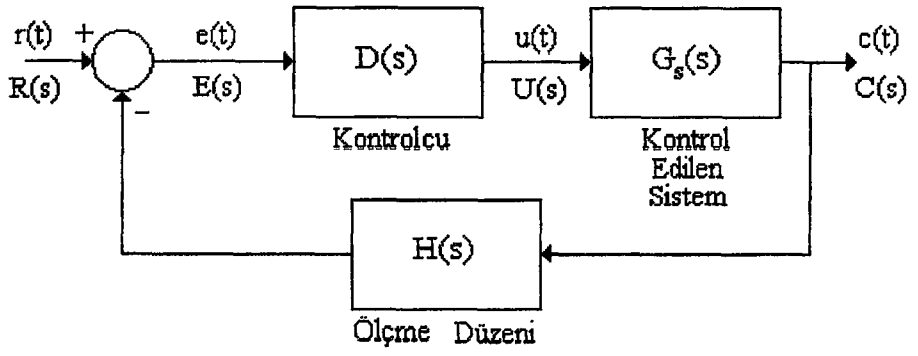
Şekil 3.2 Kapalı-çevrim geribeslemeli kontrol sistemi blok diyagramı

Bir geribeslemeli kontrol sistemi, referans giriş ile çıkış arasında önceden belirlenmiş bir ilişkiyi tanımlayan bir fonksiyon altında çalışmaktadır. Kontrol edilen sistemin çıkışı ile referans giriş arasındaki fark çoğunlukla kuvvetlendirilerek sistemi kontrol etmek için kullanılmakta ve neticesinde bu fark azalmaktadır. Geribesleme kavramı kontrol sistem analizi ve tasarımı için en temel kavramdır (Sarıoğlu, M. K., 1991).

Geleneksel kontrol teorisinde, alışılagelmiş kontrol sistemlerinin analiz ve sentezinde sistemin matematik modelinin kurulması esastır. Sürekli kontrol sistemlerinde sistem parametreleri lineer ve zamanla değişmiyorsa, bu sistemler zaman tanım bölgesinde normal diferansiyel denklemlerle yada durum denklemleriyle, s domeninde Laplace dönüşümü ve transfer fonksiyonlarıyla , w domeninde Fourier dönüşümü ve transfer fonksiyonları yardımıyla ifade edilirler. Transfer fonksiyon yaklaşımının temel avantajı, kontrol sistemlerinin analiz ve sentezinin kolayca yapılabilmesine imkan tanımasıdır.

Sürekli bir kontrol sisteminin elemanları transfer fonksiyonları ile ifade edilerek Şekil 2.3. 'de blok diyagramı ile gösterilmiştir. Şekil 3.3'de verilen blok diyagramında r(t) referans değerini, c(t) sistem çıkışını, e(t)=r(t) - c(t) hata değerini ve u(t) kontrol işaretini ifade eder. Blok diyagramı ile verilen Şekil 2.3'deki sistemin kapalı çevrim transfer fonksiyonu,

$$T(s)=\frac{C(s)}{R(s)}=\frac{D(s)G_s(s)}{1+D(s)G_s(s)H(s)} \qquad (3.1)$$



Şekil 3.3 Sürekli kapalı çevrim bir kontrol sistemi.

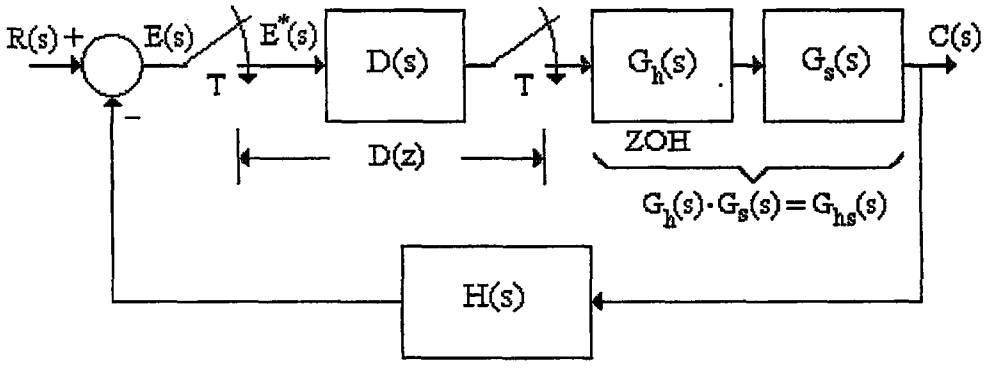
olarak bulunur. Bu denklem $G(s)=D(s)\,G_s(s)$ tanımı ile ,

$$T(s)=\frac{C(s)}{R(s)}=\frac{G(s)}{1+G(s)H(s)} \tag{3.2}$$

şeklinde en basit bir kontrol sisteminin yapısı elde edilmiş olur.

Dijital kontrol sistemlerinin matematik modelinin elde edilmesinde, fark denklemleri kullanılır. Alışılmış kontrol sistemlerinin s domeni analizine benzer şekilde, dijital kontrol sistemlerinin matematik modeli z domeninde z dönüşümü transfer fonksiyonları ile verilir. Ayrıca, giriş işaretlerinin sürekli sinüsoidal işaretlerden eşit aralıklarla alınmış darbe yada impuls dizisi olması halinde, genlik ve fazlarını veren frekans domeni davranışı ile de incelenebilir.

Analog sistemlere benzer şekilde, bir dijital kontrol sistemi kullanılan elemanların transfer fonksiyonları ile Şekil 3.4. 'de blok diyagramlarıyla gösterilmektedir. Şekilde örneklenmiş işaretler * ile gösterilmektedir.



Şekil 3.4 Geri beslemeli basit bir dijital kontrol sistemi.

Blok diyagramıyla verilen Şekil 3.4'deki sistemin transfer fonksiyonu,

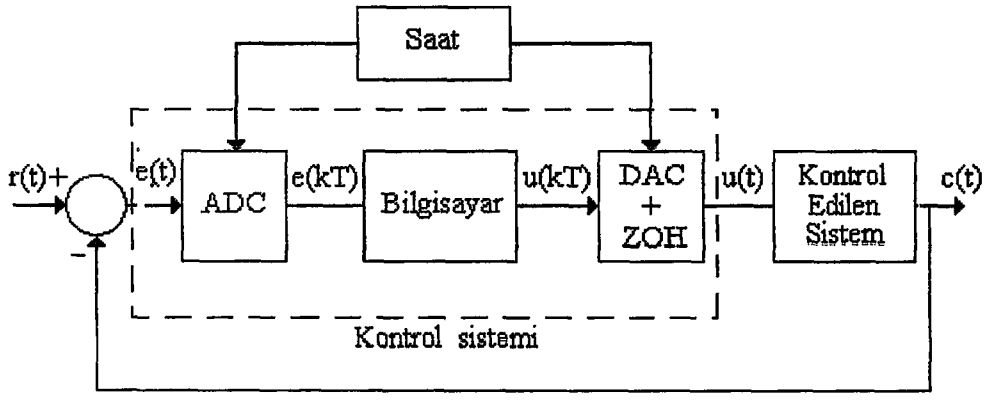
$$T(z)=\frac{C(z)}{R(z)}=\frac{D(z)G_{hs}(z)}{1+D(z)G_{hs}H(z)} \tag{3.3}$$

şeklinde olur. Bu denklemde $G_{hs}(s)=G_h(s)\ G_s(s)$ tanımı ile verilmektedir.

3.3 Dijital Kontrol Sistemi Temel Yapısı ve Birimleri

Fiziksel dünyada sistemler analog olduklarından bunları dijital kontrol sistemleri ile kullanmak için bunlar arasında bir bağlantı kurulması gerekir. Bu bağlantı analog-dijital dönüştürücülerin, analog işareti istenilen örnekleme zamanı (Ts) altında kesikli zamanlı işarete dönüştürmesiyle sağlanır. Kesikli zamanda gelen bilgi, dijital kontrolcu vasıtasıyla istenilen kontrol algoritmasında işlendikten sonra, analog sinyale uygulayabilmek amacıyla tekrar dijital analog dönüştürücü vasıtasıyla analog işarete dönüştürülür. Bu dönüştürme işlemleri istenilen örnekleme frekansında devam ederken, sisteme uygulanan kontrol işaretinin sürekli olabilmesi için kontrolcudan çıkan işaret bir tutucuya verilerek burada saklanır.

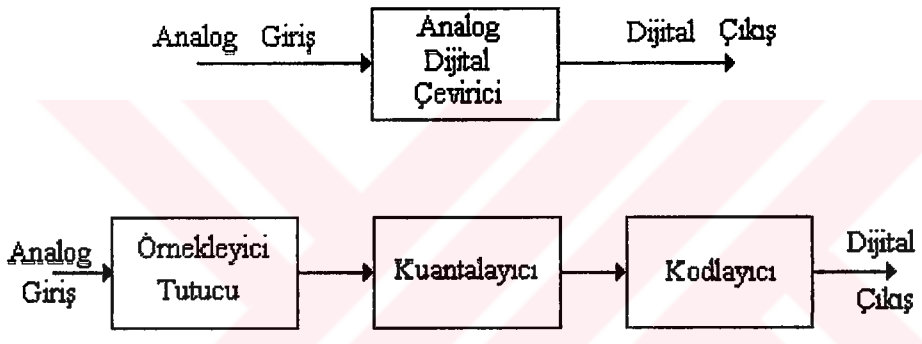
Dijital işaret, sürekli işaretin belirli bir örnekleme frekansı ile örneklenmesi, tutulması, kuantalanması (sayısal değerlendirilmesi) ve özel bir şekilde kodlanmasıyla elde edilir. Buna göre belirli bir aralıkla değişen sürekli bir işaret 0 ve 1 'lerden oluşmuş bir darbe katarı şeklinde dijital olarak elde edilir. Bu dijital işaret bir bilgisayar (mikroişlemci, mikrokontrolör veya dijital işaret işleyici) vasıtasıyla değerlendirilerek sistemi kontrol eden bir işaret üretilir. Şekil 3.5 'de bir dijital kontrol sisteminin blok diyagramı verilmiştir.



Şekil 3.5 Kapalı çevrim dijital kontrol sisteminin blok diyagramı.

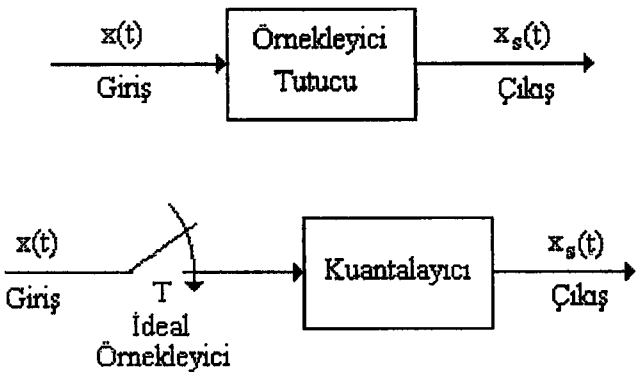
Burada r(t) referans değeriyle c(t) sistem çıkışı arasındaki e(t) hatası kT örnekleme zamanlarında ADC vasıtasıyla e(kT) şeklinde dijital veriye çevrilir. Bilgisayar (veya mikroişlemci, mikrokontrolör veya dijital işaret işleyici) e(kT) işaretini, kontrol algoritmasını kullanarak yorumlar ve u(kT) şeklinde yeni bir kontrol işareti üretilir. DAC vasıtasıyla bu kontrol işareti sıfırıncı mertebeden tutucu (ZOH) tarafından örnekleme anlarında sabit tutularak bir sürekli işarete çevrilir. Yukarıda adı geçen ADC, DAC ve ZOH elemanlarının yapıları ve çalışma şekilleri aşağıda açıklanmaktadır.

Sürekli işaretten dijital işaret dizisini elde etmede kullanılan elemana Analog-Dijital Çevirici adı verilir. Bu elemanın yapısı ve çalışması Şekil 3.6 'da verilmiştir.



Şekil 3.6 Analog-Dijital Çevirici (ADC) elemanının yapısı.

Şekil 3.6'de gösterilen ADC yapısı içinde bulunan ilk eleman örnekleyici ve tutucu elemandır. Bu elemanın görevi, belirli aralıklarla sürekli işaretten örnek alıp bu değerleri belirli bir süre tutmaktır. Elemanın prensip şeması Şekil 3.7 'de verilmiştir.



Şekil 3.7 Örnekleyici-tutucu elemanın blok yapısı.

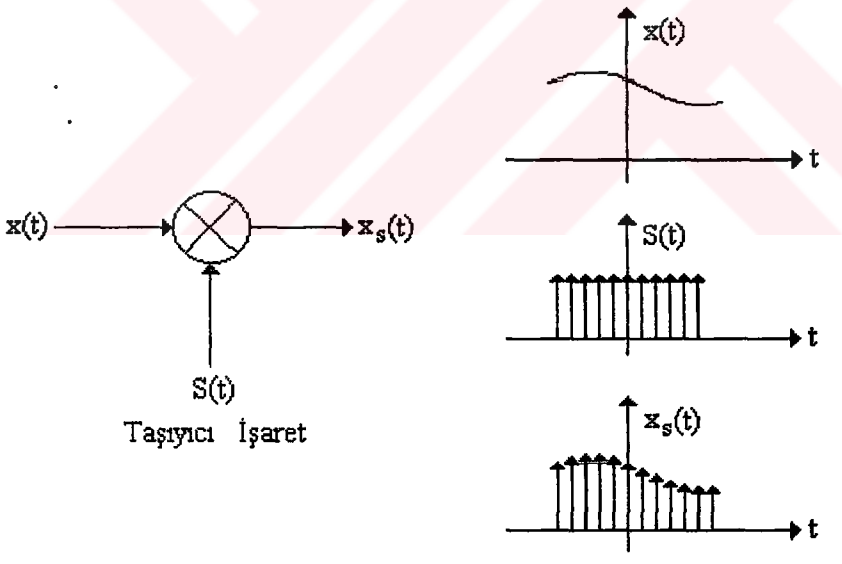
Şekil 3.7'deki yapıda T ideal örnekleyicinin örnekleme peryodunu göstermektedir. Örneklemede işlem süresi, örnekleme peryodundan çok küçük olduğu için, sıfır yani herhangi bir gecikmenin olmadığı kabul edilmiştir. Örnekleme elemanın çalışma prensibi Shannon teoremi veya Nyquist örnekleme teoremi olarak da bilinen örnekleme teoremine dayanmaktadır. Bu teoremin tanımı verilmeden önce band sınırlı işaret kavramı açıklanmalıdır.

$$X(\omega) = 0 \quad , \quad |\omega| \geq \omega_m \tag{3.4}$$

koşulunu sağlayan x(t) işareti için "ω_m : rad/s ile band sınırlıdır" denir. Bu durumda örnekleme teoremi için; ω_m rad/s ile band sınırlı bir x(t) işaretinin, bu işaretten eşit aralıklarla (T) alınan örnek değerlerinden tek ve bozulmasız olarak yeniden elde edilmesi için gerek ve yeter koşul

$$\omega_s \geq 2\omega_m \tag{3.5}$$

olmasıdır (ω_s : örnekleme frekansı). Şekil 2.8'de ideal örnekleme işlemleri gösterilmiştir.



Şekil 3.8 İdeal örnekleyicide örnekleme işlemleri.

İdeal örnekleyicide taşıyıcı işaret birim impuls fonksiyonu olarak,

$$S(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT) \tag{3.6}$$

şeklinde gösterilir. Buna göre örneklenmiş işaret,

$$x_s(t) = x(t)S(t) \tag{3.7}$$

$x_s(t) = x(kT)\delta(t - kT)$ olmaktadır.

Örnekleme frekansının teoremde belirtilen değerden küçük seçilmesi örneklenen işaretin yeniden elde edilmesini imkansızlaştırır, bu olaya örtüşme etkisi denir. Kapalı çevrim kontrol sistemlerinde bu durum sistem kararlılığını bozar. Bu nedenle dijital kontrol sistemlerinde örnekleme frekansının seçimi önem taşımaktadır ve sistem frekansının birkaç katı olacak şekilde seçilmelidir. Örnekleme frekansının çok büyük alınması sistem performansı üzerinde görünür bir iyileştirme yapamayacağı gibi daha karmaşık ve pahalı dijital kontrolcular gerektirecektir.

Endüstride kullanılmakta olan bazı sistemler için örnekleme peryodları aşağıda verilmiştir.

Servo mekanizmalarda: T = 0,001-1 s.

Seviye kontrolünde: T = 5-10s

Basınç kontrolündeT = 1-5 s.

Sıcaklık kontrolündeT = 10-45 s.

Damıtma kontrolündeT = 10-180 s.

Kimyasal sistemlerdeT = 10-45 s

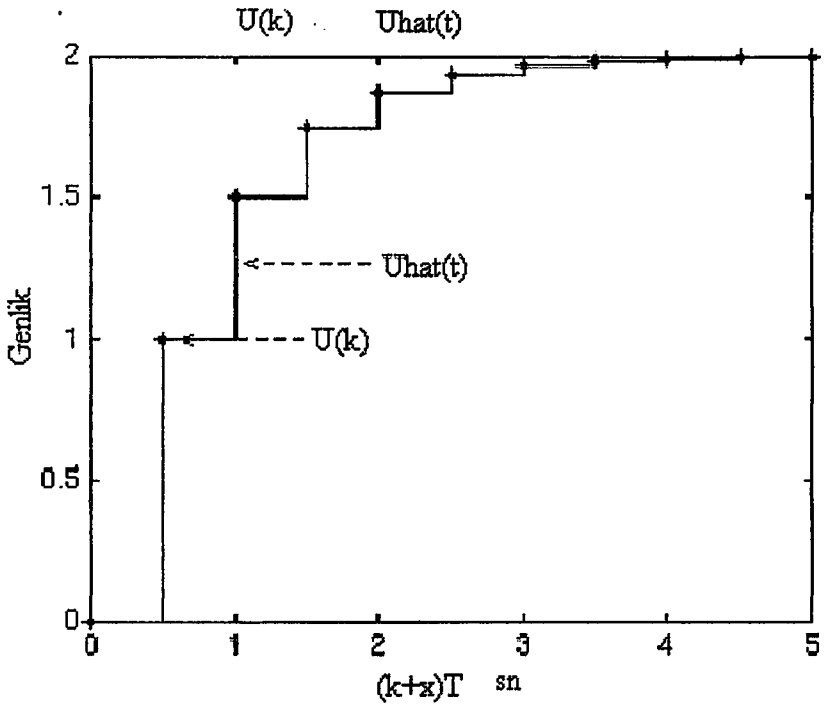
Çimento sanayii ve kurutma sistemlerinde: T = 20-45 s.

Tutma elemanı olarak doğrudan sıfırıncı mertebeden tutucu-ZOH kullanılır. Amaç bir sonraki örneklenmiş değer elde edilinceye kadar eski işaretin sabit tutulmasıdır. Şekil 3.9 'da tutma işlemi gösterilmiştir.

Lineer bir eleman olan sıfırıncı mertebeden tutucunun transfer fonksiyonunu bulmak için impuls cevabını birim basamak cinsinden ifade etmek ve Laplace dönüşümünü almak yeterlidir. Bu durumda giriş işareti birim impuls olduğunda, çıkış işareti,

$$e_o(t) = u(t) - u(t - T) \tag{3.8}$$

$$E_o(s) = \frac{1}{s} - \frac{e^{-sT}}{s} \tag{3.9}$$



Şekil 3.9 Tutucuda giriş ve çıkış işaretleri.

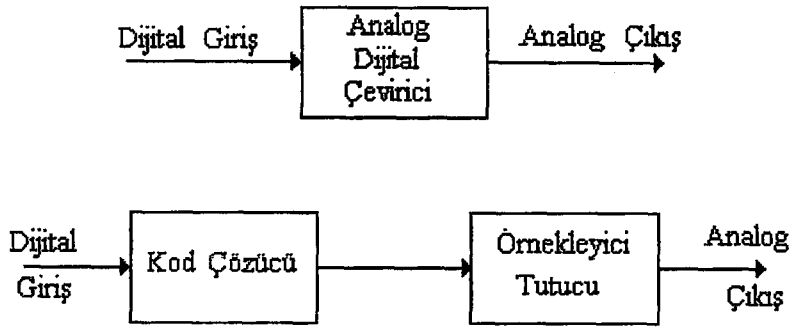
olur. $E_i(s)=1$ olduğu için, tutucunun transfer fonksiyonu,

$$G_h(s)=\frac{E_o(s)}{E_i(s)}=\frac{1-e^{-sT}}{s} \tag{3.10}$$

şeklinde elde edilir.

ADC yapısında, örnekleyici ve tutucu elemandan sonra kuantalayıcı adı verilen birim yer almaktadır. Örneklenip tutulan her değer önceden belirlenmiş düzeylere karşı düşürülür ve bu düzeylerden gerçek değere yaklaştırma yapılır. Bu işleme kuantalama, bu işlemi gerçekleştiren birime de kuantalayıcı adı verilir. Son olarak her kuanta düzeyi özel bir sözcükle kodlanır. Bu son birim kodlayıcı olarak bilinir. Bu şekilde sürekli işaret, darbe katarı şeklinde dijital işarete dönüştürülmüş olur.

Kontrolcu vasıtasıyla sisteme uygulanacak dijital kontrol işareti belirlendikten sonra bu işaret DAC vasıtasıyla sürekli işarete çevrilir. Burada elde edilen işaret istenen değerde olmayabilir. Bunun sebebi, ADC dönüşümü sırasında oluşan kuantalama hatasıdır. DAC elemanının yapısı Şekil 3.10'da verilmiştir. Burada gerçeklenen işlem önce dijital işaretin kodunun çözülmesi, DAC 'ye uygulanan referans gerilimiyle orantılı bir analog gerilim değeri belirlenmesi ve bu gerilim değerinin çıkışta bir sonraki değer gelene kadar sabit tutulmasıdır.



Şekil 3.10 Dijital-Analog Çevirici (DAC) yapısı.

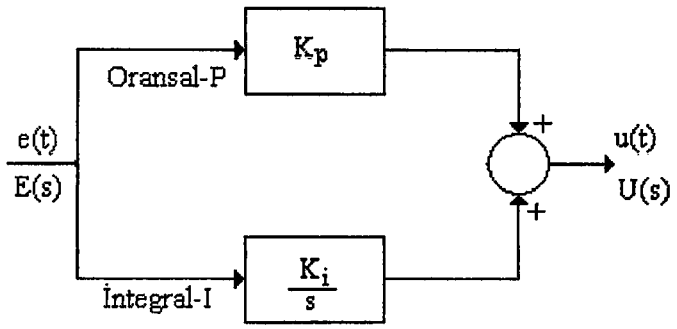
Bu şekilde gerçeklenen dijital kontrol sisteminin sürekli kontrol sistemine göre çeşitli avantajları bulunmaktadır. Dijital kontrol sistemleri gürültü ve bozucu etkilerden daha az etkilendiklerinden, sistem parametrelerinin değişimlerine daha az duyarlı olduklarından daha güvenilir kontrol sistemleri oluştururlar. Dijital elemanların kullanılması da sistemin güvenilirliğini arttırmaktadır. Ayrıca programlama işleminin çok basit ve esnek olmasından dolayı analog sistemlerde aşılması güç olan tasarım değişikliklerini dijital sistemlerde gerçekleştirmek mümkün olmaktadır. Kontrol edilen sisteme göre, dijital kontrolcularla kuantalama aralığının uygun seçilmiş olması halinde, yeterli derecede hassasiyete ulaşılmaktadır. Diğer bir önemli avantaj ise bu sistemlerin giderek ucuzlayan fiyatlarıdır. Bilgisayar mimarisindeki gelişmeye elverişli yapıdaki yeni elemanlar üretildikçe, daha önce üretilen ürünlerin fiyatlarının düşmesi de kaçınılmaz olmaktadır.

3.4 Dijital PI Kontrol Genel Yapısı ve Gerçeklenmesi

Kontrol sistemlerinde yaygın olarak kullanılan kontrolculardan birisi PI (proportional+integral) kontrolcudur. Genel bir PI kontrolcunun zaman tanım bölgesinde,

$$u(t) = K_p e(t) + K_i \int_o^t e(t)dt \qquad (3.11)$$

denklemiyle ifade edilebilir. Bu yapı Şekil 3.11 'de gösterilmiştir.



Şekil 3.11 PI kontrolcunun zaman tanım bölgesindeki blok diyagramı.

PI kontrolcudaki integral fonksiyonu sürekli hal hatasını azaltma işlemini gerçekleştirir. PI kontrolcunun z-tanım bölgesi transfer fonksiyonunu yaklaşık olarak elde edebilmek için; integral işlemi trapezoidal (Tustin yaklaşımı olarak da bilinmektedir) yaklaşım ile,

$$X(z) = Z\left\{K_i \int_0^t e(t)\,dt\right\} = \frac{K_i T(z-1)}{z(z-1)} E(z) \tag{3.12}$$

şeklinde ifade edilir. Buna göre PI kontrolcunun z-tanım bölgesindeki transfer fonksiyonu,

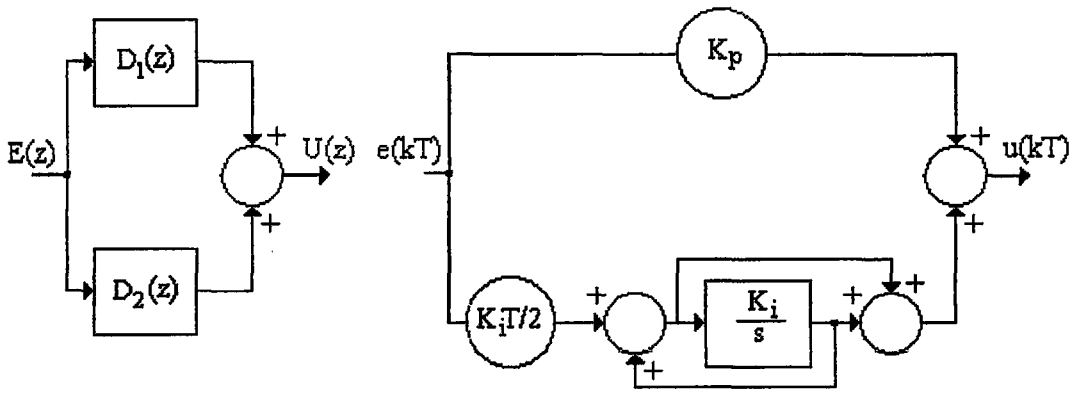
$$\frac{U(z)}{E(z)} = K_p + K_i \frac{T(z+1)}{2(z-1)} \tag{3.13}$$

olarak elde edilir. Buradan,

$$D(z) = \frac{U(z)}{E(z)} = \frac{(2K_pT + K_iT^2)z^2 + (K_iT^2 - 2K_pT)z}{2Tz(z-1)} \tag{3.14}$$

ifadesi türetilebilir.

Dijital kontrol sistemlerinde dijital kontrolcuları gerçekleştirmek için bir dijital bilgisayar, mikroişlemci veya mikrokontrolörler kullanılır. Dijital kontrolcuların fark denklemlerinin elde edilip bir dijital programın yazılmasıyla dijital kontrolcular gerçekleştirilebilmektedir. Ancak dijital kontrolcuların gerçekleştirilebilmesi için, z-tanım bölgesindeki transfer fonksiyonlarında payın derecesi paydanın derecesine eşit veya küçük yani $m \leq n$ olmalıdır. Eğer $m > n$ olursa kontrol vektörü dinamik sistemin zaman içinde önüne geçer ki bunun da fiziksel bir anlamı yoktur. Şekil 3.12'de PI kontrolcunun paralel programlama ile gerçekleştirilen yapısı gösterilmiştir.



Şekil 3.12 PI kontrolcunun paralel programlama ile gerçekleştirilmesi.

Denklem (3.14) aşağıdaki şekilde

$$\frac{U(z)}{E(z)} = D_1(z) + D_2(z) \tag{3.15}$$

olarak basitleştirilebilir. Burada

$$D_1(z) = \frac{U_1(z)}{E(z)} = K_p \tag{3.16}$$

$$D_2(z) = \frac{U_2(z)}{E(z)} = \frac{K_i T}{2}\frac{1+z^{-1}}{1-z^{-1}} \tag{3.17}$$

şeklinde ifade edilir.

$E(z) = e_1$: (kT) anındaki örneklemeye ilişkin hata işareti,

$z^{-1}E(z) = e_2$: (kT-T) anındaki örneklemeye ilişkin hata işareti,

$U(z) = u$: (kT) anındaki örneklemeye ilişkin kontrol işareti,

olmak üzere fark denklemleri sırasıyla,

$$u_1 = K_p e_1 \tag{3.18}$$

$$u_2 = \frac{K_i T}{2} e_1 + K_i T e_2 \tag{3.19}$$

bulunur. Genel kontrol işareti $u=u_1 + u_2$ ve

$$p_1 = K_p + \frac{K_i T}{2} \tag{3.20}$$

$$p_2 = \frac{K_i T}{2} \tag{3.21}$$

tanımları ile,

$$u = p_1 e_1 + p_2 e_2 \tag{3.22}$$

şeklinde, her T örnekleme anı için PI kontrolcu algoritmasının oluşturduğu kontrol işareti bulunur (Wang Y.G. and Shao H. H., 2000).

3.5 Kontrol Sistemi Karakteristik Değerleri ve Performans Ölçümü

Kapalı çevrim bir kontrol sisteminde referans büyüklüğünde bir birim basamak değişmesi durumunda, cevap ve diğer bir deyimle çıkışın zamana bağlı olarak değişmesi Şekil 3.13' te gösterilmiştir. Kararlı bir kontrol sistemi için elde edilmiş bulunan böyle bir cevap fonksiyonunun veya davranışının uygunluğunu belirtmek üzere genellikle aşağıdaki karakteristik değerler kullanılmaktadır (Sarıoğlu, K., 1991),

1. Aşma (Overshoot): Çıkış büyüklüğünün ilk yükselmede son (kalıcı hal) değeri aşma miktarıdır.

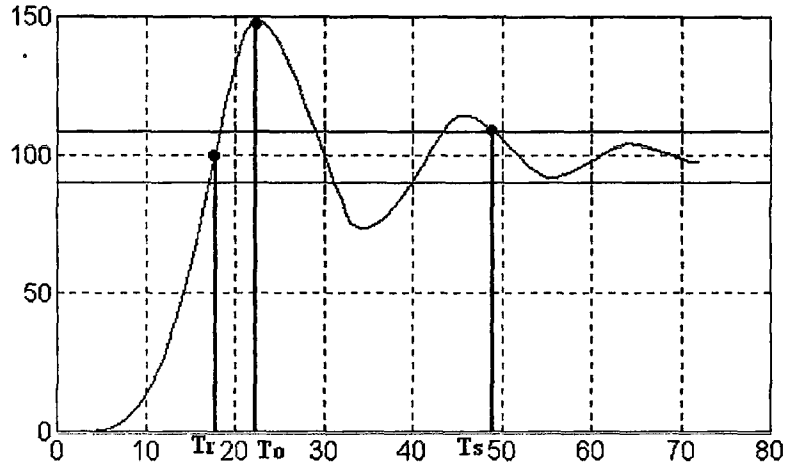
$$C_{max}\text{-}C \tag{3.23}$$

2. Aşma yüzdesi : Aşma miktarının son değere oranıdır ve yüzde olarak tanımlanır.

$$\frac{C_{max} - C}{C} \cdot 100\% \tag{3.24}$$

3. Maksimum değere ulaşma zamanı (To_{max}) : Çıkışın C_{max} değerine ulaşması için geçen zamandır.

4. Yükselme zamanı (Rise time, Tr) : Cevabın referans değerinin 10%ından 90% kadar çıkması veya cevabın başlangıçtan referans değerine kadar erişmesi için geçen zamandır. verilen her iki tanım da belirtilmek şartıyla kullanılmaktadır.



Şekil 3.13 Kapalı çevrim kontrol sistem cevabı performans parametreleri

5. Yerleşme zamanı (Ts) : Cevap eğrisinde titreşimlerin son değerin belirli bir yüzdesine kadar küçülmesi için gereken zamandır. Genellikle Ts hatanın 2% veya %5 i geçmemesini sağlayacak şekilde tanımlanır ve değişmenim hata için kabul edilen yüzde ile sınırlanmış bir tolerans bölgesi içinde kalmaya başlamasına kadar geçen süreden ibarettir.

6. Genlik oranı : Cevap eğrisinin referans eğrisine nazaran birbirini takip eden titreşimlerinin genliklerinin oranıdır.

7. Sönümlü titreşim peryodu veya frekansı wl: Kararlı bir sistemde cevap eğrisinin sönümlü titreşimlerinin peryodu veya frekansıdır.

Bilindiği gibi, sistemlerin başlangıç şartlarına bağlı kalarak çıkış fonksiyonu, aynı giriş için değişir. Dolayısıyla sistemlerin davranışlarını karşılaştırmak için sükunetteki sistemlerin girişine bir birim basamak girişi uygulamak yoluyla yükselme zamanı, aşma yüzdesi, yerleşme zamanı v.s gibi karakteristik değerler karşılaştırılabilir. Söz konusu karakteristik değerler bir veya birkaçı birlikte kullanılarak Optimum performans tanımlanması gereklidir.

Herhangi bir sistemin referans girişine bir birim basamak uygulandığında en mükemmel çıkış şüphesiz ki aynı birim basamak fonksiyonu elde ederek her an hatasız çalışma sağlayabilmesidir. Fiziksel sistemlerde böyle bir çıkışın imkan dahilinde olmamasından dolayı Şekil 3.13'deki çıkışa ait karakteristik değerler tek tek veya birlikte göz önüne alınarak en uygun sistem davranışı bulmaya çalışılmaktadır. En uygun sistem cevabını belirlemede yaygınca kullanılan metodlar;

1. Hatanın Karesinin İntegrali (ISE) Kriteri,

$$J = \int_0^\infty e^2(t)\, dt \tag{3.25}$$

ISE kriterine göre sistem performansının kalitesi yükseltilmesi, hatanın karesinin integralinin minimize edilmesiyle mümkün olmaktadır. Bu kriter ile tasarlanan sistemlerde büyük başlangıç hatası hızla azalacak şekilde bir davranış görülmektedir. Buna göre sistemin cevabı hızlı fakat osilasyonlu olmaktadır.

2. Mutlak Hatanın Zaman Çarpımlı İntegrali (ITAE) Kriteri,

$$J = \int_0^\infty t \cdot |e(t)|\, dt \tag{3.26}$$

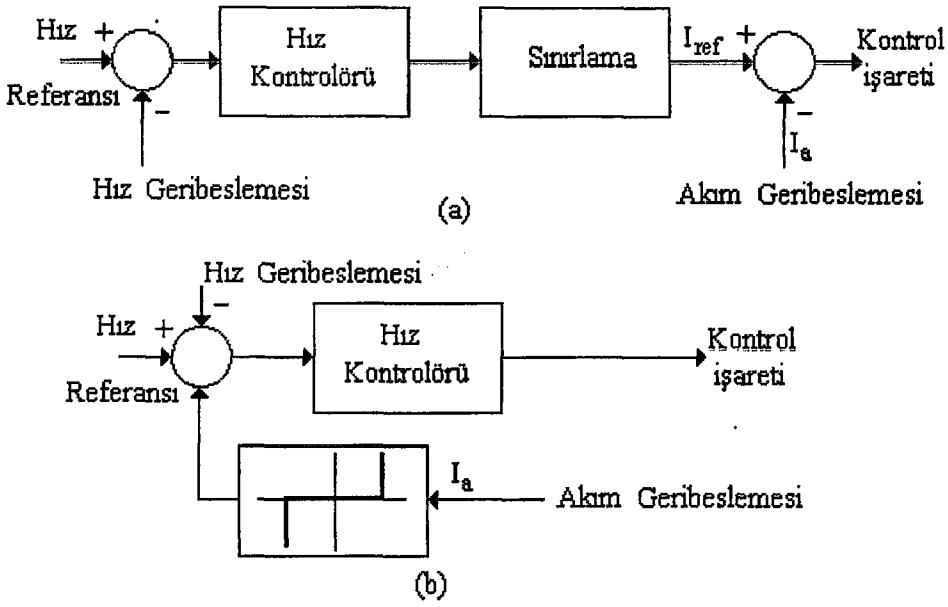
Bu kritere göre optimum sistem, yukardaki J kriterini minimize eden sistemdir. ITAE'ye göre tasarlanan sistemlerde geçici cevapta aşım miktarı küçük ve osilasyonlar bastırılmıştır.

3. Bu çalışmada ise hatanın karesinin zaman çarpımlı integralini esas alan bir performans (ITSE) kriterini içeren bir uyumluluk fonksiyonu kullanılmıştır.

$$J = \int_0^\infty t \cdot e^2(t)\, dt \tag{3.27}$$

3.6 DC Motorun Kapalı Çevrim Hız Kontrolü

Birçok hız kontrol sürücüsünde kapalı çevrim kontrol bulunmaktadır. DC motorların açık çevrim kontrolü birçok uygulamada iyi bir performans sergilemiyor. Örneğin, tetikleme açısı sabit iken motora bir yüklenme olursa hız değişecektir. Buna karşın sabit hızda çalışma için tetikleme açısı değiştirilmek zorundadır. Bu durum ancak kapalı çevrim kontrol sistemlerinde gerçekleştirilmektedir. Genel olarak, yüksek doğruluk, hızlı dinamik cevap ve yükleme gibi bozucu durumların etkisinin azaltılması şeklinde kapalı çevrim kontrol sisteminin avantajları vardır.



Şekil 3.14 Akım sınırlama yöntemi a) İç akım kontrol çevrimi b) Dış akım kontrol çevrimi.

Çoğu endüstriyel sürücü sistemler de kapalı çevrim geribeslemeler içerir. Kapalı çevrim kontrol vasıtasıyla etkin koruma da sağlanabilmektedir. Tam bir kontrol sisteminin oluşturulmasında değişik kontrol yöntemleri uygulanmaktadır. Hız kontrol sisteminde akım sınırlama özelliğinin de bulunması önemlidir. İki akım sınırlama yöntemi vardır. Şekil 3.14 'de her iki akım sınırlama yöntemi gösterilmiştir.

1. İç Akım Kontrol Çevrimi : İç akım kontrol çevrimi uygulamasında, hız hata işareti akım referansı olarak kullanılır. Hız hata işaretinde bir sınırlama akım referansında bir sınırlama sağlar. Bu uygulamada ana veya dış döngü hız çevrimidir. Akım çevrimi her zaman işlem altındadır. Bu yöntem hızlı cevap sağlar, tristörlü doğrultucunun lineer olmayan kazancını lineerleştirir ve kaynağın bozucu etkilerinin üstesinden gelir.

2. Dış Akım Kontrol Çevrimi : Dış akım kontrol çevrimi uygulamasında, bir akım sınırlayıcı kuvvetlendiricisinde önceden belirlenmiş bir referans eşik değeri ile akım geribesleme işareti karşılaştırılır ve neticeye göre hız kontrolcusuna işaret gönderilir. Akım önceden belirlenen değer altında kaldığı sürece, akım sınırlayıcı kuvvetlendiricinin çıkışı sıfırdır. Buna karşın, eğer akım eşik değerini aşarsa, akım sınırlayıcı kuvvetlendirici hız kontrol işaretinin üzerine binen büyük bir çıkış üretir ve sabit akım modunda çalışmayı sağlar. Burada yalnızca hız döngüsünün olduğu tek bir çevrim vardır. Bundan dolayı tasarım basit olmaktadır.

3.6.1 DC Motor PI Kontrol Sistemine Ait Sonuçlar

PI kontrolcu için parametre belirlenmesinde kullanılan birçok metod bulunmaktadır. Litaretürde kullanılan yaygın yöntemlerden birisi kutup-sıfir yerleştirme yöntemidir. Laplace s veya z domenindeki kontrol sistemleri tasarımında kullanılan bu yöntemde; sistemi kararsızlığa götürebilecek kutup ve sıfırların kontrolcunun transfer fonksiyonundan çıkarılmaktadır. Daha sonra aşımın minimum olmasını sağlamak amacıyla yerleştirilecek sıfırlar, s domenindeki eksen takımında kontrol kutuplarının beş katı kadar uzağına yerleştirilir. Kontrolcu parametrelerini bulmak için öncelikle kontrol edilecek sistemin transfer fonksiyonunun çıkarılması gerekir. Bu tezde kontrol edilecek sistem, kalıcı mıknatıslı uyarma sargısına sahip doğru akım motorudur.

Bu tezde kullanılan DC motora ait parametreler motorun üretici firması olan Sinano Electric Co. (Japonya) firmasıyla yapılan iletişim yoluyla elde edilmiş olup aşağıdaki verilmiştir.

Atalet momenti (J) : 0.2225 g-cm-s^2

Sürtünme katsayısı (B) : 1.75 Kg-cm

Armatür direnci (Ra) : 12.6 ohm

Armatür Endğktansı (La) : 9 mH

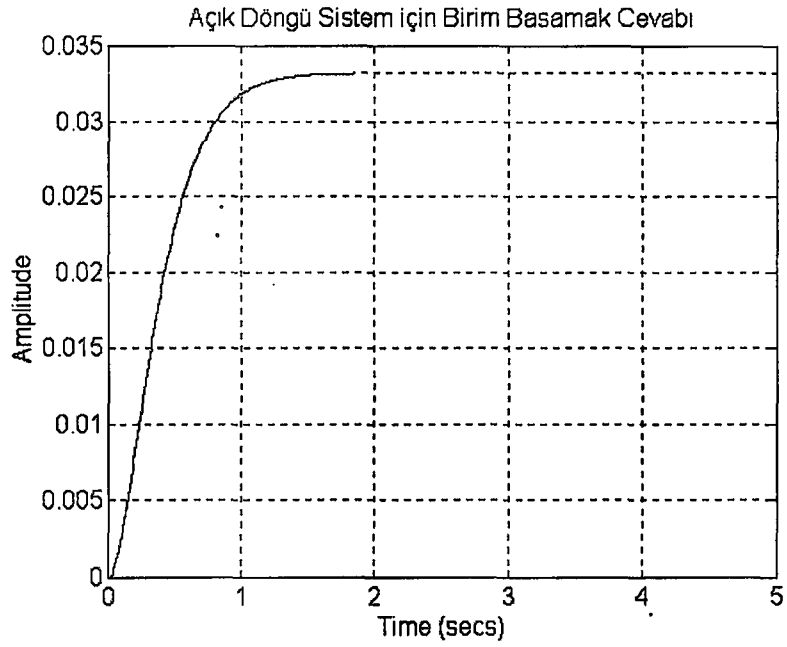
Tork sabiti (Ki) : 1.5 Kg-cm/A

Geri besleme sabiti (Kb) : 15.4 mV/rpm

Motora ait parametreler dc motor transfer fonksiyonunda yerine yazılırsa, kontrol edilecek sisteme ait transfer fonksiyonu elde edilmiş olur. Buna göre tezde kullanılan motorun transfer fonksiyonu

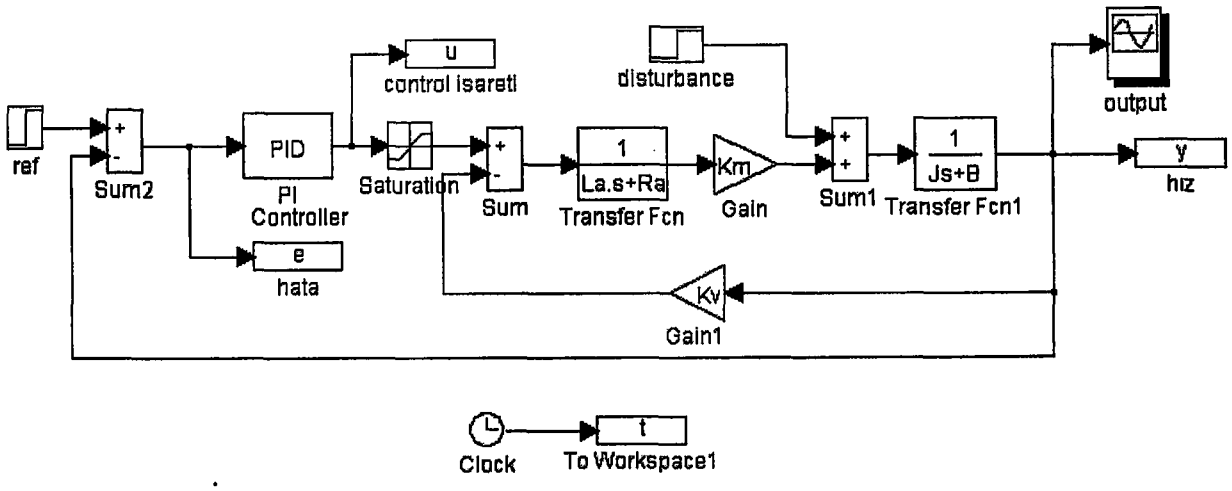
$$G(s) = \frac{0.7407}{s^2 + 9.178\,s + 22.3} \tag{3.28}$$

şeklindedir. Kontrol edilecek sistem olan DC motorun birim basamak cevabı Şekil 3.15' da verilmiştir.

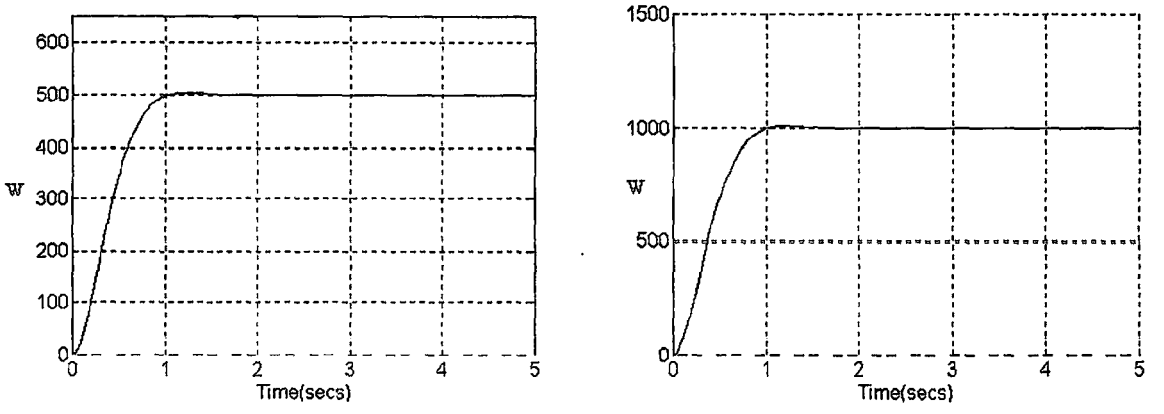


Şekil 3.15 DC motor kontrol sisteminin açık-çevrim birim basamak cevabı

Kontrol edilecek DC motor sistemine ait PI parametreleri, Matlab-Simulink ile tasarlanan kontrol sisteminin modeli üzerinde çeşitli değerler denenerek sisteme uygun PI parametreleri Kp=21 ve Ki=76 olarak tespit edilmiştir. Şekil 3.16'de DC Motor PI-Kontrol Sistemine ait blok diyagramı verilmiştir. Kontrol sisteminin blok diyagramı üzerinde alınan sonuçlar Şekil 3.17'de gösterilmektedir.

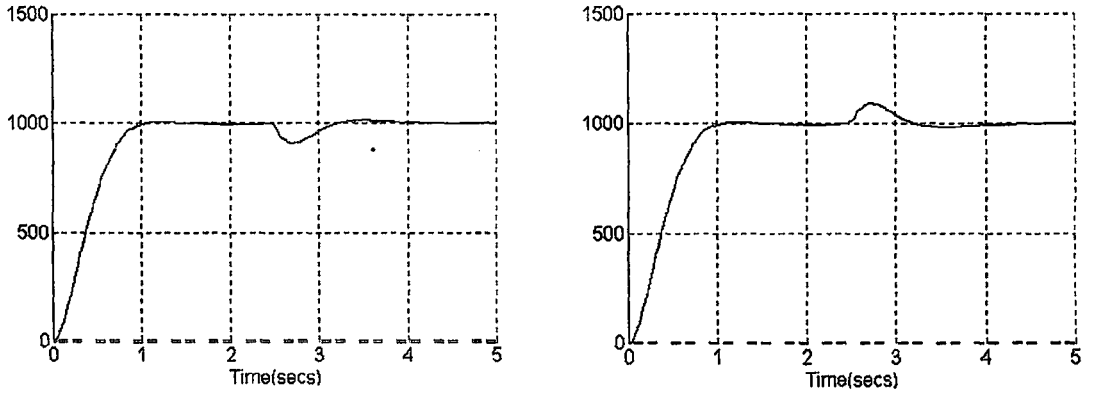


Şekil 3.16 Matlab-Simulink ile çizilen DC Motor PI-Kontrol Sistemine ait blok diyagramı



Şekil 3.17 Motor hızı olarak 500 rad/s ve 1000 rad/s referans hızları için sistem cevabı

Kontrolcunun performansını ölçmek için, kontrol edilen çıktı işareti olan motor hızı, istenilen referans hızına oturduktan sonra 250 N.m'lık yük devreye konulup ve çıkarılarak kontrol sisteminin tepkisi ölçülmüştür. Şekil 3.18'de yükün devreye alınıp çıkarılmasına ait sistem cevabındaki değişimler gösterilmektedir.



(a) (b)

Şekil 3.18 W=1000 rad/s referans hızı için t=2.5 sn'de T_L=250 Nm'lık yükün (a) devreye alınması (b) devreden çıkarılması

4. BULANIK MANTIK KONTROL

4.1 Bulanık Mantık Tarihçe ve Gelişmesi

Bulanık mantık, insan düşünme ve algılamasını modellemek için kullanılan güçlü bir araçtır. İki değerli önerme yerine bulanık sistemler, çok-değerli kümelerle sonuç verir. Bulanık sistemler kuralları saklayıp dilsel girişten dilsel çıkışa örneklenmiş fonksiyonları belirler. Klasik mantığın dayandığı temel varsayım "her önerme doğru veya yanlıştır". Bu Aristo'dan beri tartışma konusu olmuştur. Aristo "Temel varsayım" adlı tezinde gelecek şartlara bağlı olarak olayların şüpheli doğruluk durumlarından bahseder. Bahsettiği gelecek olaylar hakkındaki önermeler aslında ne doğru ve ne de yanlıştır. Fakat iki durumun da olması imkan dahilindedir. Bulanık kümeleri konusu ilk defa Lotfi A. Zadeh tarafından 1965 yılında ortaya atılmıştır. Zadeh'in bu çalışması şu gerçeği kuvvetlendirmiştir ki, insanların bazı sistemleri, makinelerden daha iyi kontrol edebilmelerinin bir sebebi, insanların kesinlik ifade etmeyen bir takım bilgileri kullanarak karar verebilme özelliğine sahip olmalarındandır. Dolayısıyla eğer bu özellik sistemlerin modellenmesinde kullanılırsa dizayn edilen kontrolörlerin performansının arttırılması mümkün olacaktır. Bulanık kümelerinin Bulanık Mantık Kontrol (Fuzzy Logic Control-FLC) olarak pratik uygulamalarda kullanılması ise 70'li yıllarda gerçekleşmiştir. Bunlar bazı küçük çaplı endüstriyel uygulamalardı. İlk bulanık çipi ise 80'lerin başında AT&T Bell laboratuarlarında Togai ve Watanabe tarafından geliştirilmiştir (Atmaca H. ve Bulut M., 1995).

Temel olarak geleneksel veya konvansiyonel bir PID kontrolör sistemi, matematiksel olarak bir dizi diferansiyel eşitlik ile sistemin modellenmesi ve bunların çözümü ile sistem davranışının kontrol edilmesine dayanır. Fakat bir FLC sistemi ise bir insanın bir sistemi kendisi kontrol ediyorken yapacağı düşünce işlemlerinin mantıksal olarak modellenmesine dayanır. Bu mantık yapay sinir ağları (neural networks) ve genetik algoritmalar (Genetic Algorithms) ile beraber 'akıllı' kontrol sistemleri kuşağının ilk adımlarını oluşturmakta olduğu bugün kabul edilmektedir (Kruse,R. ve Nauck,D., 1995).

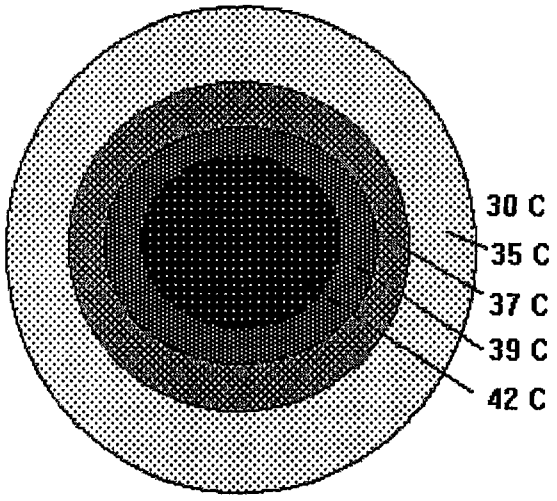
FLC'nin dünyaya tanıtılmasında önemli olaylardan biri FLC 'nin Sendai (Japonya) metrosunun otomatik kontrolü için kullanılması olmuştur (1987). Bu uygulamada FLC'nin bir çok parametre açısından konvansiyonel bir PID kontrolörden daha üstün olduğu gösterilmiştir (Örneğin, istasyonda yavaşlayıp durma, yolcu konforu ve yakıt tüketimi gibi..). Bu uygulamanın başarısı sebebiyle FLC kontrollü bu sistem yeni Tokyo metrosunda da kullanılmaktadır.

Özellikle Japonya'da 90'lı yıllarda FLC kullanılan tüketici ürünleri pazarda sıkça görülmeye başlanmıştır. Genellikle bu tüketici ürünlerinde (FLC çiplerinin yüksek hızlarına gerek olmadığından dolayı) FLC çipleri yerine, standart dijital çipler üzerinde tablo göz atmaları (table look-up) vasıtasıyla simulasyonlar kullanılmıştır. Tüketici ürünleri örneklerini şöyle sıralayabiliriz: Çamaşır makineleri, elektrik süpürgeleri, klimalar, fanlar, ısıtıcılar, mikrodalga fırınları, çamaşır kurutucuları. Bunun yanında birçok endüstriyel uygulamalarda da FLC başarıyla kullanılmıştır (King P.J. and Mamdani E.H., 1977).

4.2 Bulanık Mantık Temel Kavramları

Bulanık mantık, bulanık küme teorisine dayanmaktadır. Bulanık küme teorisi genel bir matematiksel yaklaşımdır. Çözülmesi güç olan problemler genel bir yapıya kavuşturularak daha kolay bir sonuca gidilir. Bulanık küme teorisi kısmi üyeliğe izin veren bir mantık sistemidir. Yani bir kümenin tam üyeliği ile o kümenin üyesi olmama durumları arasında derece derece geçişe izin verir. Verilen bir elemanın bir kümede kısmi üyeliğinin bulunması demek, aynı zamanda bu elemanın bu kümenin üyesi olmama durumunun da kısmen başlaması demektir. Çünkü bulanık küme teorisi, hem tam üyeliğe ve hem de hiç üye olmamaya izin verir.

Yukardaki duruma uygun bir örnek üzerinde duralım. Bir dilsel belirsizliğin uygun olarak nasıl modeleneceğine bakalım. Eğer bir doktor bir hastasının "yüksek ateş" ten acı çekip çekmediğini belirlemek için kafasında tam bir eşik değere (threshold) sahip değilse nasıl çalışacak? Psikodilsel araştırma doktorun iki hasta "örnekleri" ni karşılaştırarak bulacağını göstermektedir. Bir tarafta soluk tenli, terleyen ve titreyen "tam" bir yüksek ateşli bir hasta. Bir diğer tarafta "tam " normal ateşli ve bu belirtileri göstermeyen bir hasta. Doktor iki hasta durumunu karşılaştırarak "yüksek ateş" hakkındaki değerlendirmeyi yapar. Bu durum matematiksel olarak nasıl modellenebilir. İlk olarak yüksek ateşli bütün hastaların kümesini belirlemek için küme teorisi ele alalım. Sonra her bir hastanın bu kümenin üyesi olup olmadığını gösterecek bir matematiksel fonksiyon tanımlayalım. Geleneksel küme teorisinde tanımlanacak gösterge fonksiyonu her bir hastanın "yüksek ateş" kümenin üyesi olduğu veya bu kümenin üyesi olmadığı gibi bir kesin tanıma sahip olacaktır. Halbuki bir doktor bu kümeye girmeyen bir hastasını yüksek ateşli bir hastanın örnekleriyle karşılaştırıp bunu da bu sınıf içinde değerlendirebilecektir (Ross, Timothy J., 1995).



Şekil 4.1 Yüksek ateşli hastaların bulanık kümesi (Ross, Timothy J., 1995).

Yukardaki şekilde bir küme örneği verilmektedir. Bu kümenin siyah bölgesi kesin elemanları göstermekte diğer bölgeler de bu kümeye yakın elemanları göstermektedir. Bu bir bulanık kümesidir.

Şekilde, her vücut sıcaklığı "yüksek ateş" örneklerine yakınlığına göre bir dereceyle belirtilmiştir. Bu derece " üyelik derecesi" olarak adlandırılır ve "yüksek ateş" YA kümesine X elemanının kümeye ait olma derecesini verir. Burada vücut sıcaklığı evrensel küme içinde bir x "temel değişken" olarak adlandırılır. Böylece

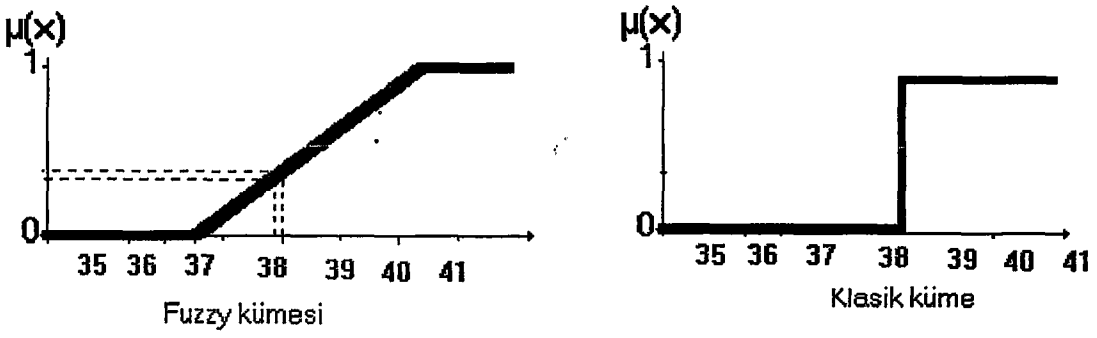
$$\mu_{YA}(x) \rightarrow \text{YA:}[0,1] \qquad (4.1)$$

ifadesi, 0-1 aralığındaki değerleri kapsayan YA kümesindeki x elemanının üyelik derecesini gösterir. Buna göre çeşitli vücut sıcaklıklarının "Yüksek Ateş" kümesindeki üyelik dereceleri şu şekilde verilebilir.

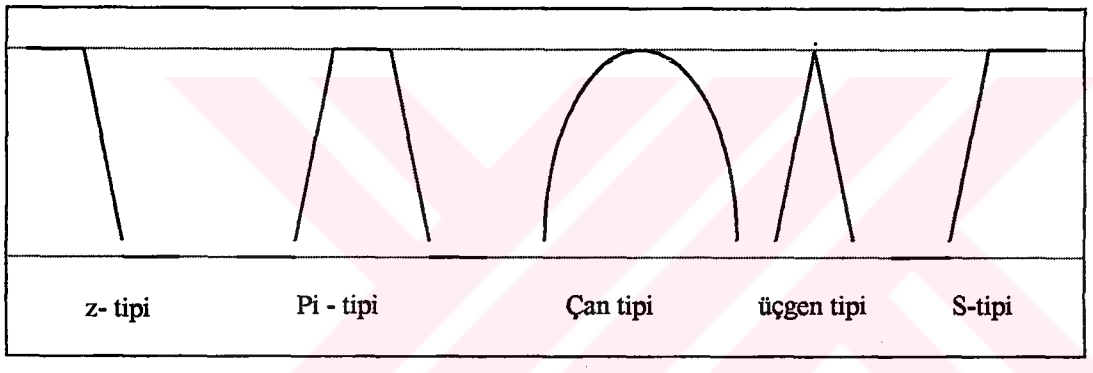
$\mu SF(33°C) = 0$	$\mu SF(37°C) = 0.1$	$\mu SF(40°C) = 0.9$
$\mu SF(34°C) = 0$	$\mu SF(38°C) = 0.35$	$\mu SF(41°C) = 1$
$\mu SF(35°C) = 0$	$\mu SF(39°C) = 0.65$	$\mu SF(42°C) = 1$

Bulanık mantık ile yapılan bir çözümün ilk aşamasında sistemin giriş ve çıkışlarına, üyelik fonksiyonları tayin edilir. Tipik olarak, bir üyelik fonksiyonu x-y düzleminde bir eğri olarak

gösterilir. Bu düzlemde, x-ekseni, giriş ve çıkış değişkenlerinin değer aralığını ve y-ekseni ise 0-1 arasında olmak üzere değişkenin üyelik derecesini gösterir.

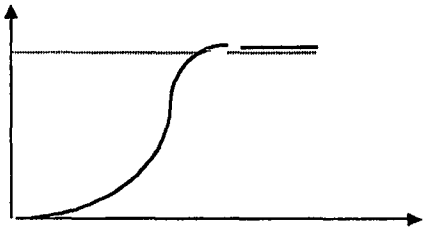


Şekil 4.2 Bulanık küme ile klasik kümenin karşılaştırılması



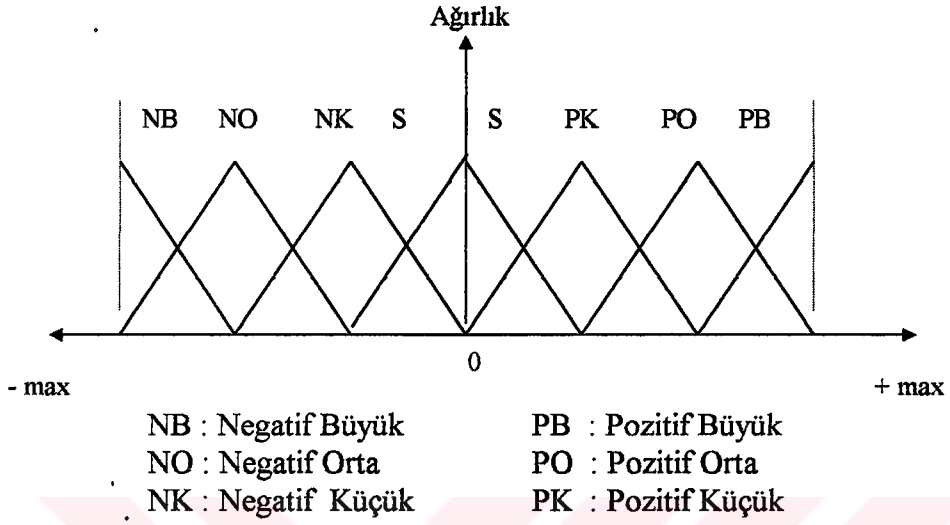
Şekil 4.3 Çeşitli Üyelik Fonksiyonları

Üyelik fonksiyonları, denetlenen sürecin özelliklerine göre çok değişik biçimlerde şekillenebilirler. Çeşitli üyelik fonksiyonları Şekil 4.3'te gösterilmiştir. Tabii ki üyelik fonksiyonları, insanların gerçek değerleri yorumlama şekillerini sadece yaklaşık olarak ifade edebilirler. Gerçekte, psikodilsel çalışmalar, üyelik fonksiyonlarının bazı özelliklere sahip olmaları gerektiğini ortaya çıkarmıştır. İşte, bu özellikleri aynı anda sağlayabilen fonksiyon " cubic spline" dır ki Şekil 4.4'te gösterilmiştir.



Şekil 4.4 "Cubic spline" üyelik fonksiyonu

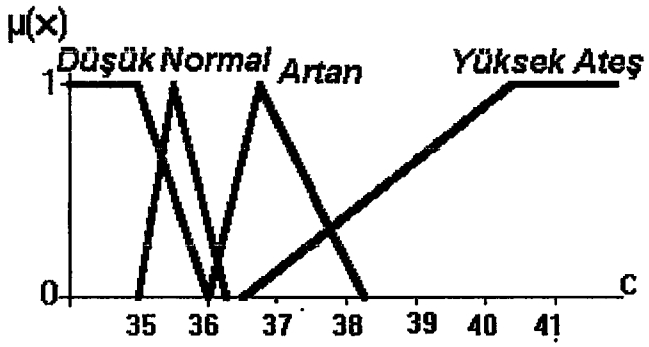
Uygulamalar, S-tipindeki cubic spline üyelik fonksiyonlarının karar verme ve data analiz sistemlerinde iyi sonuçlar verdiklerini göstermiştir. Ancak, uygulamalar için diğer üyelik fonksiyonları da yeterlidir.



Şekil 4.5 Yedi etiketli üyelik fonksiyonuna tanım uzayı

Üyelik fonksiyonlarında kullanılacak etiket sayısı 3,5,7 olabilir ve kullanıcı tarafından en uygun olanı seçilir. En yaygın kullanılanı yedi etiket olup Şekil 4.5'te gösterilmiştir.

Bir Bulanık mantık sistemin ilk yapı bloğu "dilsel değişken" olarak adlandırılır. Aynı kavramı tanımlayan kategoriler birleştirilir. Ateş konusunda, yalnız "yüksek ateş" değil "yükselen ateş", "normal ateş" ve "düşük ateş" kümeleri de oluşturulur. Aşağıdaki şekilde aynı grafik üzerinde birkaç dilsel değişkene ait üyelik fonksiyon gösterilmektedir. Bir dilsel değişken gerçek değerleri dilsel değerlere dönüştürür. Bunu gösteren üyelik fonksiyonları Şekil 4.6'da gösterilmektedir.



Şekil 4.6 Ateşli bir hastanın ateş durumunun üyelik fonksiyonlarına dağılımı

4.2.1 Kural Tabanın Oluşturulması

Bulanık mantık, makinalara operatörünün şahsi düşüncelerini işleyebilme ve deneyimlerinden faydalanarak çalışabilme imkanı sağlar. Bu, bir sistemin bulanık mantık ile yapılan tasarımında sistemin çalışma mekanizmasını belirleyecek olan "Kural Kümesi" ile gerçekleşir. Bu kurallar sayesinde, insana ait çıkarım şeklinin veya karar verme tarzının sistemlere uygulanması mümkün olur. Kuralların belirlenmesinde kullanılacak belli bir yöntem yoktur. Ama, bir uzmanın bilgi ve deneyimlerine dayanarak , sistemin bir bulanık modelinin kurulmasına göre ya da çeşitli öğrenen algoritmaları kullanarak kuralları oluşturabiliriz. Bir kural yazımı şu şekilde gerçekleşir (Sugeno M., 1985).

IF < ön şart > THEN < sonuç > (4.2)

< ön şart >, belirli bir durumu tanımlar. < sonuç > ise sistemin bu durum için vermesi gereken tepkisini kapsar. < ön şart > birden fazla şartı içinde bulundurabilir. Bu şartlar aşağıda belirtilen operasyonlar ile birbirlerine bağlanırlar.

x AND y = min (x'in üyelik ağırlığı, y'nın üyelik ağırlığı) (4.3)

x OR y = max (x'in üyelik ağırlığı, y'nın üyelik ağırlığı) (4.4)

NOT x = 1 - x'in üyelik ağırlığı (4.5)

Şimdi, oluşturulan bu kural kümesinin işlenmesini ve bulanık sistemin çalışmasını adım-adım inceleyerek açıklayalım;

Toplama : Bu adımda, giriş değişkenlerinin bulanıklaştırılmış değerleri ön şart kuralı altında işlenir. Bu sonuç, her kuralın, o andaki durum için uygulanabilme derecesini belirtir. Bu adım "AND" işlemi ile gerçekleştirilir.

Derleme : Bu adımda, o anki duruma göre yapılacak işlev belirlenir. İşlev, kuralların <sonuç>'ları olarak tanımlıdır ve kuralın o anki duruma uygulanabilirliği ne kadar fazla ise sonuç'da o kadar fazla etkilenecektir. Bu adım " (ise)" işlemi ile gerçekleştirilir.

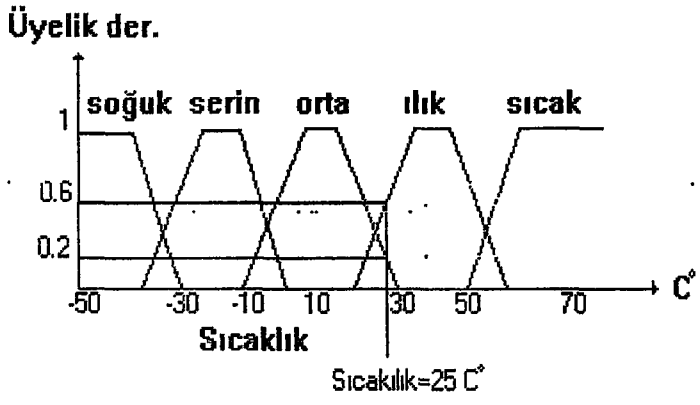
Durulama : Derleme adımının sonunda "bulanık çıkarım" elde edilmiş olur. Elde edilen bu bulanık çıkarım, kural sonuçlarının doğruluk derecelerini gösterir. Bulanık mantık sisteminin çıkışının bir gerçek değer olması gerektiğinden durulama (defuzzification) gereklidir. Böyle bir dönüşümde dilsel terimlerin, temel değişken ile aralarındaki ilişkisi göz önüne alınmalıdır.

Durulama İşlemi : Tüm kuralların sonuçları, karşılıklı terimlerin üyelik fonksiyonlarıyla birebir eşlenirler. Sonuçta elde edilen üyelik fonksiyonları tek bir üyelik fonksiyonuna indirgenir. Elde edilen tek üyelik fonksiyonu, istenen çıkışı tanımlayan bulanık bir kümedir. Ancak, çıkışta tek bir değer istendiğinden , bu üyelik fonksiyonu tek bir değere sıkıştırılmalıdır. Durulama işlemi, ağırlık merkezi, maksimumların ortalaması, soldakilerin maksimumları ve sağdakilerin maksimumları gibi birçok yöntem kullanılarak yapılabilir.

Bir bulanık mantık sisteminin kuralları sistem bilgisini gösterir. Örnek için bir Bulanık Mantık kontrol sisteminin kontrol stratejisini ifade etmek için vocabulary (kendisine ait sözcükler) olarak dilsel değişkenler kullanılır. Şekilde giriş sıcaklığı atanan beş üyelik fonksiyonunu göstermektedir. Bu şekilde y-ekseni her giriş değeri için ona karşılık gelen bütün üyelik fonksiyonlarındaki üyelik derecesini vermektedir. Mesela giriş sıcaklığı 25 C° olsun. y-eksenine bakıldığında bu giriş değeri için ılık ve orta fonksiyonlarındaki üyelik derecesi sırasıyla 0.2 ve 0.6'dır (Şekil 4.7). Bu giriş değeri için diğer bulanık kümelerindeki üyelik derecesi sıfırdır. Giriş ve çıkış üyelik fonksiyonları atandıktan sonra yapılacak iş, sistemin davranışını tanımlamak için bulanık kuralları (rule) üretmektir. Bulanık kurallar tipik olarak IF-THEN şeklindedir. Mesela tipik bir bulanık kural olarak

IF sıcaklık **Düşük** AND basınç **Orta** ise THEN güç **Hızlı**'dır. (4.6)

Bu kural iki varsayım ve bir karar içermektedir. Bulanık mantıkta varsayımlar AND işlemi ile kurallar da OR işlemi ile ayrılır.



Şekil 4.7 Sıcaklık değişkeni için üyelik fonksiyonları

4.3 Bulanık Mantık Kontrol Sisteminin Yapısı

FLC'ye bir örnek olması açısından Mamdani ve Assilian tarafından geliştirilen buhar türbini (steam engine) kontrolörü önemli bir yer tutar. Bu çalışmalarıyla Mamdani ve Assilian bugün FLC'ye Mamdani yaklaşımı olarak adlandırılan ve bugün kullanılan birçok pratik yaklaşımın temeli olan metodu ortaya koymuşlardır (King P.J. and Mamdani E.H., 1977). Bu kontrolör sensörlerden alınan kazan (boiler) basıncı ve türbin (engine) hızı verilerine göre kazan ısı kaynağı ve buhar kısma valfinda (throttle) ayarlamalar yapar. Eğer bu kontrolörü PD yapısında temellendirmek istersek basınç ve hız değişkenlerinden herbiri için ikişer değişken düşünmek zorundayız. Bunlar basınç için basınç hatası (istenen seviyenin altında ve üstündeki miktar) ve basınç değişimi (en son ölçümden beri basınçtaki değişikliğin derece ve yönü), hız için ise aynı şekilde hız hatası ve hız değişimidir. Mamdani ve Assilian'ın bu uygulama için geliştirdikleri kontrolör aşağıdakine benzer şekilde 24 kural (rule) içermektedir:

(Yani: EĞER Basınç Hatası Negatif Büyük bir değer,
veya Negatif Orta bir değer, veya Negatif Küçük bir değer;
ve Basınç Değişikliği Pozitif Büyük bir değer Değil;
ve Hız Hatası Negatif Büyük bir değer;
ve Hız Değişikliği Pozitif Büyük bir değer Değil
İSE Isı Ayarlaması Pozitif Orta bir değer olmalı.)

Yukarıdaki notasyon elbette pek kullanışlı değildir. Bunun yerine değişkenler ve değerler için uygun kısaltmalar kullanmak daha uygun olacaktır.

Değişkenler için;

HH = Hız Hatası
BH = Basınç Hatası
HD = Hız Değişikliği
BD = Basınç Değişikliği
ID = Sıcaklık Değişikliği (Isı seviyesindeki ayarlama)
VD = Buhar Kısma Valfi Değişikliği

Değerler için;

PB = Pozitif Büyük
PO = Pozitif Orta

PK = Pozitif Küçük

OC = Sıfır Civarında

NK = Negatif Küçük

NO = Negatif Orta

NB = Negatif Büyük

Bu kısaltmaları kullanarak yukarıdaki kuralı tekrar yazarsak:

IF (BH=NB or BH=NO or BH=NK) and not (BD=PB) and HH=NB and not (HD=PB)
THEN ID=PO (4.7)

elde ederiz. Bütün kuralları bir tabloda toplamak görünüş ve algılama açısından faydalı olabilir. Fakat bu durumda ölçülen değişkenlerin sadece iki tane olması ve sonucun bir tane olması gerekirdi. Örneğin, örneğimizde ısıtıcı ayarlaması sadece basınç hatası ve basınç değişikliğine göre olsa idi aşağıdaki gibi bir tabloda bütün kurallarımızı gösterebilecektir (Çizelge 4.1). Bu tablolama işlemi bilgisayara giriş yaptığımızda çok faydalı olacaktır.

Çizelge 4.1 Isı Kaynağı Kontrol Kuralları

		BASINÇ HATASI						
		NB	NO	NK	OC	PK	PO	PB
BASINÇ DEĞİŞİKLİĞİ	NB	PB	PB	PB	PO	PO	PK	OC
	NO	PB	PB	PO	PO	PK	OC	NK
	NK	PB	PO	PO	PK	OC	NK	NO
	OC	PO	PO	PK	OC	NK	NO	NO
	PK	PO	PK	OC	NK	NO	NO	NB
	PO	PK	OC	NK	NO	NO	NB	NB
	PB	OC	NK	NO	NO	NB	NB	NB

Notasyondan daha önemli olan bir şey ise "Basınç Hatası Negatif Büyük bir değerdir" gibi bir cümle ile ne anlatılmak istendiğidir. Bu cümledeki Basınç Hatası ve Negatif Büyük terimlerin hepsi dile ait birer ifadedirler. Burada üzerinde durulması gereken nokta bir buhar türbini kontrolünü bir çok defalar tecrübe etmiş bir insanın türbin kontrolü hakkındaki bilgisini bu tip cümlelerle ifade edebilmesidir. Bu tip cümle dizileri kural temelli bilgiye dayalı uzman (expert) sistemlere de benzemektedir.

Yani, eğer biz bir insan kontrolörünün yaptığı gibi bu bilgileri otomatik kontrolde makinalara kullandırabilirsek, inceden inceye bir matematiksel modelleme yapmamıza gerek kalmayacaktır.

Ayrıca bilgilerin illaki tecrübeye dayanması da gerekmemektedir. Uzman bir kişinin yeni geliştirilen bir sistem hakkındaki sezgisel bilgileri de bu gibi cümlelerle ifade edilebilmektedir. Buda sistem dizaynının en zor tarafı olan modelleme, linearizasyon gibi matematiksel kesinlik isteyen çalışmaları ortadan kaldırmaktadır. Bu yönleriyle FLC'nin endüstriyel uygulamalarda hızla yaygınlaşacağını söylemek pek yanlış olmasa gerek.

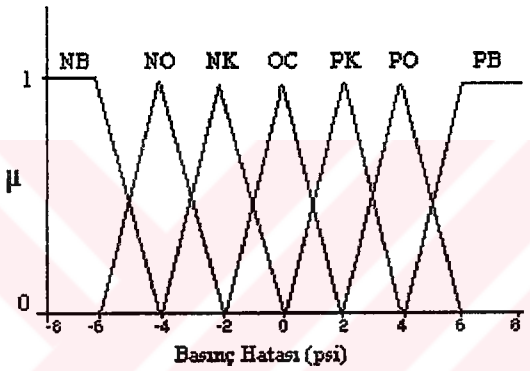
4.3.1 Bulanık Hesaplama (Computation)

Bir çok uzman sistemler sadece ayrık (discrete) olarak çalışabilirler. Bu da onlara bazı sınırlamalar ve belirsiz bilgiyi (uncertain info) başka metotlarla ifade etme zorunluluğunu getirmiştir. Fakat sezgisel bilgiye dayalı bir otomatik kontrol sistemi çok daha değişiktir. Kontrolördeki herhangi bir kural sürekli değişkenler ile ifade edilebilir. Normal olarak, iki veya daha çok değişken arasındaki ilişkiyi içeren pratik bir problem ele alındığında, bu değişkenler arasındaki ilişkiyi ifade etmek için fonksiyonları kullanmak akla gelir. Fakat $y = f(x)$ şeklinde bir fonksiyon ile sezgisel veya tecrübeye kullanmak pek doğal olmasa gerek. Örnek olarak, bir mühendis bile, araba kullanırken virajı belirli bir hızda almak için direksiyonun ne kadar çevrilmesi gerektiğini bir fonksiyonla ifade etmez. Bu bakımdan araba kullanmak gibi sezgisel ve tecrübeye dayanan bir bilgiyi ifade etmek için bahsedilen şekilde bir anlatım daha doğal olacaktır. Fakat biz, "Basınç Hatası Negatif Büyük bir değer" ifadesiyle ne anlatmak istediğimizi kesin olarak tanımlamalıyız.

Mümkün olan bütün "Basınç Hatası" değerleri sürekli aralığını sabit aralıklı bölümlere parçalamak mantıksal görünebilir. Örneğin, -5 psi veya daha aşağısına "Negatif Büyük", -3 psi'den -5 psi'ye kadar olan aralığa "Negatif Orta" ve bunun gibi benzer değerler verilebilir. Fakat bu aralıklardaki sabit değerler, iki sebep yüzünden tam olarak istenilen formu vermezler. Bu durumda "Basınç Hatası" değerlerini bir takım aralıklarda sabit olan değerler olarak düşünmek pek doğal bir şey olmayacaktır. İkinci sebep ise kontrol sisteminin performansı açısındandır. Böyle bir düşünce ile temellendirilen bir kontrol sistemi bazı durumlarda kararsızlığa sürüklenebilir. Bunun sebebi, böyle bir sistemde kontrolörün girişlerindeki çok küçük bir değişikliğin kontrol işleminde çok büyük değişikliğe sebep olmasıdır. Bu problemlere çözüm sunabilecek en güzel yol bu aşamada bulanık küme teorisi olarak görülmektedir. Bu da sezgisel bilgiye dayalı kontrol sistemlerinin neden bizi bulanık kontrol denilen bir çözüme götürdüğünün açıklığa kavuşması demektir. Burada NB, NO, NK, OC, PK, PO, PB gibi BH

değerleri bulanık kümeleri olarak tanımlanabilir. Bu bulanık kümelerinin üyelik fonksiyonları Şekil 4.8' de gösterilmiştir.

BH değişkeni bulanık kümeleri ile veya bulanık sayıları ile gösterilen değerler aldığı için bu değişken, bulanık değişkeni veya dile ait değişken olarak tanımlanabilir. BD, HH, HD, ID, VD gibi bütün tanımlanan veya tanımlanacak değişkenler aynı şekilde bulanık kümelerini kullanan değerler alabilirler ve bu sebeple bunların hepsine bulanık değişkenleri veya dile ait değişkenler denir. Nasıl ki kurallar sezgisel varsayımlara dayanır aynı şekilde bütün değişkenler için de üyelik fonksiyonları doğal olarak sezgisel bilgilere dayanmaktadır.

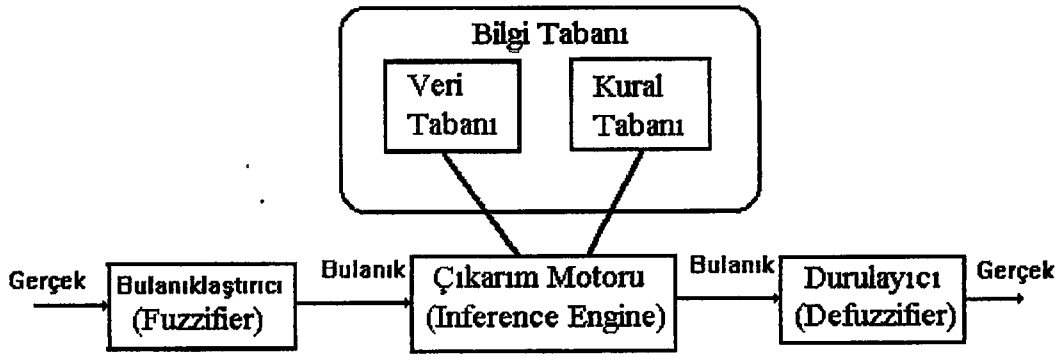


Şekil 4.8. Bulanık Üyelik Fonksiyonu

Biz "Eğer Basınç Hatası, Negatif Büyük bir değerse ....." dediğimiz zaman ne anlatmak istediğimizi iyice bilmeliyiz. Dolayısıyla, pratikte değişkenlerin değerlerinin tanımlanması kuralların ortaya konulmasından önce gelir. Üyelik fonksiyonları bazen lineer olmayan şekillerde alabilirler, fakat Şekil 4.8'de olduğu gibi genellikle üçgen ve yarım yamuk şekiller kullanılmaktadır. Bu üçgen ve yarım yamuk üyelik fonksiyonlarının en büyük özelliği belirli aralıklarda lineer olduklarından dolayı pratikte gerçeklenmelerinin kolay olmalarıdır.

Buraya kadar bahsedilen FLC kurallarının dile ait değişkenler ile ifade edilebileceği idi. Bundan sonra dile ait değişkenlerle temellendirilmiş kurallardan bulanık çıkarsamalarının (bulanık inference) nasıl yapılacağı olacaktır. Bu kısmı dört bölüme ayırabiliriz;

1. Giriş değerlerinin bulanıklaştırılması (fuzzification)
2. Birleşim, kesişim ve değillemelerle oluşturulan bileşimlerin hesaplanması
3. Bütün kurallardan toplam bulanık çıkışlarının bulunması
4. Bu bulanık çıkışlarının durulanması (defuzzification) ile son çıkış değerlerinin bulunması



Şekil 4.9 Bir Bulanık Mantık Denetleyicinin yapısı.

Bulanıklaştırıcı (Fuzzifier) : Bu bölüm giriş değişkenlerini (gerçek sayıları) ölçer, onlar üzerinde bir ölçek değişikliği yapar ve bulanık kümelere dönüştürür, yani onlara birer etiket vererek dilsel nicelik kazandırır.

Çıkarım Motoru (Inference Engine) : Burada, sistem için yazılan kurallar üzerinde bulanık mantık yürütülür ve tekrar-tekrar üyelik ağırlıkları hesaplanarak çıkarım yapılır, yani insan beyninin düşünüş şeklinin bir benzetimi yapılmaya çalışılır.

Veri Tabanı (Data Base) : Burada, oluşturulan bulanık kümelerin üyelik fonksiyonları yer alır. Ve bunlar , dilsel denetimi sağlayacak olan kuralların oluşturulmasında kullanılır. Ayrıca, bu kısım çıkarım motoruna da verilir.

Kural Tabanı (Rule Base) : Kontrol amaçlarına uygun dilsel kontrol kuralları burada bulunur ve çıkarım motoruna buradan verilir.

Durulayıcı (Defuzzifier) : Çıkarım motorunun herbir kural için oluşturacağı bulanık çıkışları toparlayarak, bunlar üzerinde gerekli ölçekleme değişiklerini yapar ve bunları gerçek sayılara dönüştürür.

Kuralların Oluşturulması : Bir sistemin Bulanık Mantık ile kontrolünde, kontrol edilecek sistemin matematik modelinden çok o sistemi çalıştıran operatörün *sistem davranışı hakkında sahip olduğu bilgiler* daha önemlidir. Tasarım sırasında genellikle bu tür bilgilerden yararlanılır. Bu yaklaşım uzun yıllar boyunca kazanılan deneyimlerin kontrolcu içerisine yorumlanmış halde kolaylıkla yerleştirilmesine olanak sağlar. Şimdi bu bölümleri teker teker ele alalım.

4.3.2 Bulanıklaştırma (Fuzzification)

Kontrolcu girişleri basit olarak belli bir kesinliğe kadar reel sayılardır. Bunlar;

v_{BH} = basınç hatası = ölçülen basınç - istenilen basınç

v_{BD} = basınç değişikliği = ölçülen basınç - önceki ölçülen basınç

v_{HH} = hız hatası = ölçülen hız - istenilen hız

v_{HD} = hız değişikliği = ölçülen hız - önceki ölçülen hız

Bu girişlerin bulanıklaştırılması üyelik fonksiyonları ile olacaktır. Bu çalışmada en temel tanımları kullandığımız için mantıksal 'and' (ve) için 'min' (minimum), 'or' (veya) için 'max' (maximum), 'not' için ise '1'den çıkarma' operatörlerini bulanıklaştırma işleminde kullanacağız. Bunları kullanarak,

(BH=NB or BH=NO or BH=NK) and not(BD=PB) and HH=NB and not(HD=PB)

gibi bir ifadeyi

$$p = \text{Min} \{ \text{Max} [\mu_{BH=NB}(v_{BH}), \mu_{BH=NO}(v_{BH}), \mu_{BH=NK}(v_{BH})], [1 - \mu_{BD=PB}(v_{BD})], [\mu_{HH=NB}(v_{HH})], [1 - \mu_{HD=PB}(v_{HD})] \} \quad (4.8)$$

şeklinde yazabiliriz.

Fakat aynı çıkış değeri için kurallar kümemizde birden çok kural bulunabilir. Örneğimizde olduğu gibi ID (Isı Değişikliği Ayarlaması) değişkeni ile sonlanan birden çok kural kontrol sisteminde bulunabilir.

Bu yüzden, toplam çıkışı elde etmek için herbir kural için çıkış hesaplanır ve o çıkış değerinde çıkışa ait bulanık kümesi sınırlandırılır ('truncate') ve sonra da bütün ilgili kurallar için bulanık kümelerinin birleşimi alınır. İlgili kuralları 1'den n'e kadar indeksleyelim ve herbir hesaplanan i kuralı için elde edilen değere p_i diyelim. i'inci kuralın çıkışında çıkış değişkeni A_i olsun. B_i'yi de A_i grafiğinin p_i değerinde sınırlandırılması ile elde edilen bulanık kümesi olarak tanımlarsak son çıkış değeri B^* aşağıdaki gibi bir bulanık set olarak tanımlanacaktır:

$$B^* = \bigcup_{i=1}^{n} B_i \quad (4.9)$$

Bulanık kümeleri için en orijinal ve doğal küme birleşimi elbette ki herbir eleman için üyelik fonksiyonlarının maksimumunu almak olacaktır. Dolayısıyla çıkış değişkeninin reel aralığındaki bütün y'ler için

$$\mu_{B^*}(y) = \text{Max}_{i=1}^{n}[\mu_{B_i}(y)] \quad (4.10)$$

olacaktır.

4.3.3 Durulaştırma (Defuzzification)

Bu son çıktı değerini durulaştırmak gerekmektedir. Çünkü son çıktı değerimiz halen bulanık bir değerdir ve bu değerle kontrol sistemimize geri besleme yapmamız mümkün değildir. Bu değeri durulaştırmak için ilk bulanık kontrolörlerde maksimumların ortalamasını (mean of maxima) almak suretiyle sonuca gidilmiştir. Bu metotta en son bulanık kümesinin üyelik fonksiyonun maksimum değerine ulaştığı değer en son çıktı olarak alınır. Eğer birden çok yerde maksimum oluşuyorsa her bir maksimuma karşı gelen değerlerin ortalaması alınır. Bu metod pratik uygulamalarda gerçeklenme yönünden çok kolaydır. Fakat sonraki FLC uygulamalarında daha çok 'ağırlık merkezi' (center of gravity) metodu kullanılmıştır.

4.4 Bulanık Mantık Sistem Dizaynı Geliştirme Metodolojisi

Herhangi bir uygulama için Bulanık Mantık sistemi nasıl geliştirilir? Birçok başarılı uygulamalarda kullanılan *Geliştirme Metodolojisi* aşağıdaki işlemler ile gösterilmektedir:

a. Sistem Dizaynı : Sistem dizaynı aşağıdaki basamakları kapsar.

1- Sistemin Dilsel Değişkenlerin Tanımlanması

2- Üyelik Fonksiyonlarının Tipi

3- Sistem için Kural Taban (Rule Base) oluşturulması

4- Uygun Durulama Yönteminin Seçilmesi

b. Off-Line Optimizasyon : İkinci gelişme basamağında, ilk basamakta dizayn edilen protipin test ve simule edilmesi gelir. Burada kullanılan teknolojiler uygulama tipine daha çok bağlıdır. Off-line optimizasyon basamağı tamamen software geliştirme aletleri (yazılım) tarafından desteklenir.

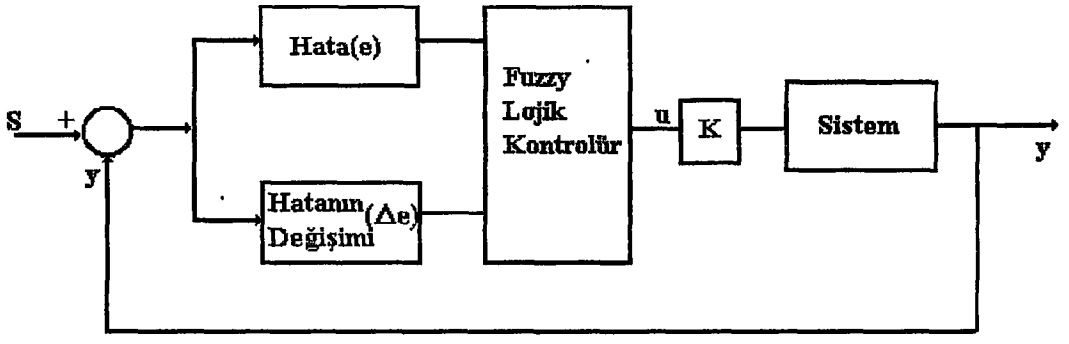
1. Prosesten önceden kaydedilen veri kullanılır
2. Bir programlama dilinde proses simulasyonu yazılır.

Bu basamakta gerçek zamanda proses bağlantısı yapmadan PC üzerinde çalışılmaktadır.

c. On-Line Optimizasyon :Çoğu kapalı döngü sistemler için simulasyon teknikleri kullanılmayabilir. Çünkü prosesi oluşturacak iyi bir matematiksel model tanımlanamaz. Bu yüzden sistem ile on-line çalışan teknikler kullanılır.

d. Implementasyon (Yürürlüğe Koyma) : Dizayn bitirildikten sonra donanım (hardware) üzerinde Bulanık Mantık sistemi çalıştırılabilir. Donanıma bağlı olarak, farklı çalıştırma (gerçekleştirme) teknikleri vardır.

4.5 Genel Bir Sistemin Bulanık Mantık Kontrolü



Şekil 4.10 Bulanık Mantık kontrol sistemine blok şema

Şekil 4.10'da bir servo sistemin Bulanık Mantık Kontrol ile kontrolünü göstermektedir. Burada kontrolun amacı izleme hatasını azaltmaktır. Hata ne kadar büyük ise kontrol girişi o kadar büyük olması gerekmektedir.

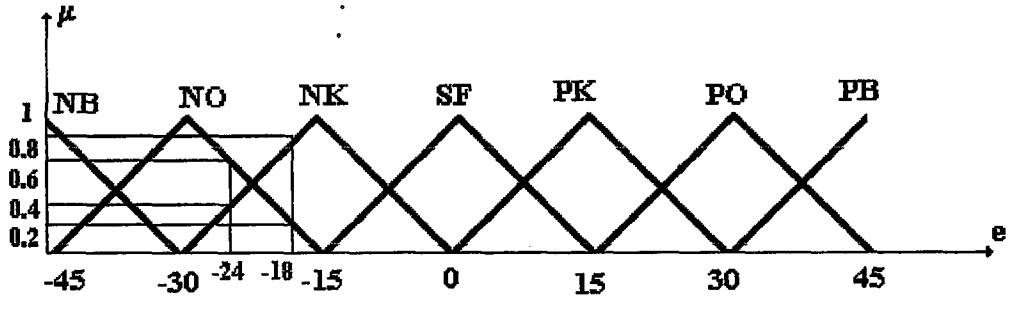
Burada FLC iki hata (e) ve hata değişimi (Δ) gibi iki girişe ve y konum bilgisi olarak bir çıkışa sahiptir. Çıkıştan alınan geri besleme ile ;

$$u = f(e, \Delta e) \qquad (4.11)$$

Hata(e) = İstenen Konum (S) - Çıkış Konumu (y) (4.12)

Hata Değişimi (Δe) = Şimdiki Hata(e_k) - Önceki Hata (e_{k-1}) (4.13)

gibi iki ifade elde edilir. Bunlar FLC'nın giriş değerleridir. Hata (e) ve Hata değişimi (Δe) değerleri iki ayrı evrensel kümeyi oluştururlar. Bu iki kümeye ait bulanık alt kümeler ve bunların üyelik fonksiyonları aşağıda gösterilmektedir. Bu üyelik fonksiyonlarında kullanılan dilsel değişkenler şu şekildedir.



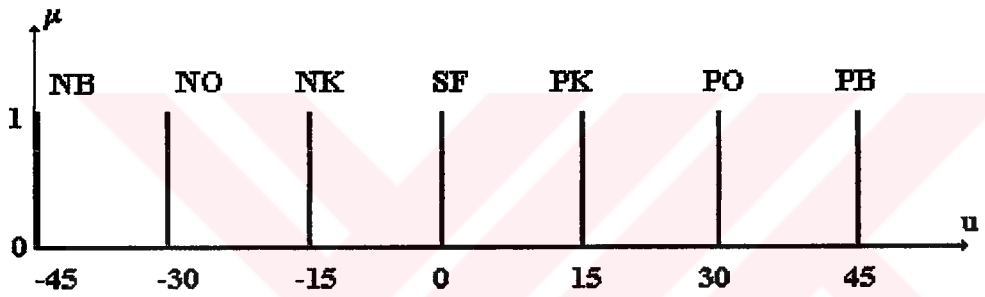
Şekil 4.11 İki ayrı hata değeri içim karşılık gelen üyelik fonksiyonları

PB : Pozitif Büyük
PO : Pozitif Orta
PK : Pozitif Küçük
SF : Sıfır
NK : Negatif Küçük
NO : Negatif Orta
NB : Negatif Büyük

Üyelik fonksiyonları belirli bir anda Hata (e) ve Hata değişimi (Δe) değerine karşılık düşen iki ayrı üyelik derecesini belirler. Ayrıca bu fonksiyonlar IF-THEN kurallarındaki şart bölümünü oluştururlar. Kurallar şu şekilde yazılır

IF e=Negatif Büyük AND Δe=Pozitif Büyük THEN u=Negatif Büyük

IF e=Sıfır AND Δe=Sıfır THEN u=Negatif Büyük



Şekil 4.12 Çıkışa ait üyelik singleton fonksiyonları

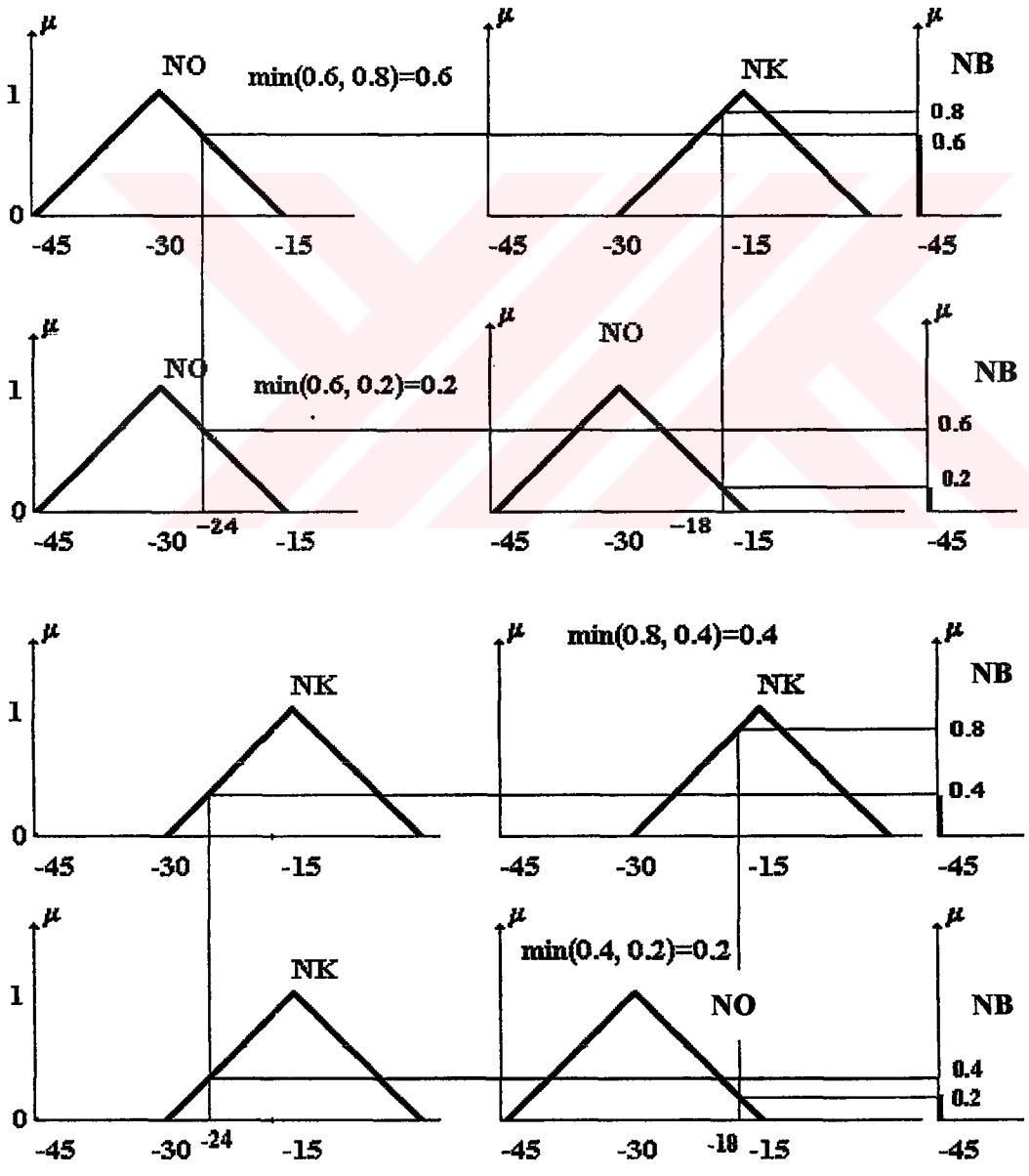
Hata (e) ve Hata değişimi (Δe) ayrı ayrı yedi tane üyelik fonksiyonu olduğu için 49 tane kural yazılabilir. Burada önemli bir husus bu kuralların sistemi bilen ve kontrolünü yapmış olan bir kişinin kuralları istenilen amaca göre çıkarmış olmasıdır. Çizelge 4.2'de bir servomotor için kurallar yazılmıştır.

Çizelge 4.2 Servo kontrol için bulanık kural tablosu

		Hata (e)						
		NB	**NO**	**NK**	**SF**	**PK**	**PO**	**PB**
Hata Değişimi	**PB**	NB	PO	PB	PB	PB	PB	PB
	PO	NB	*	PB	PB	PB	PB	PB
	PK	NB	*	*	PK	PB	PB	PB
	SF	NB	NB	NK	SF	PB	PB	PB
	NK	NB	NB	NB	NK	PK	*	PB
	NO	NB	NB	NB	NB	*	*	PB
	NB	NB	NB	NB	NB	NB	NB	PB

Yukardaki tabloya göre e= -24 ve Δe = -18 değerleri için FLC 'nın üreteceği değerin nasıl hesaplandığına bakalım. Hata (e) değeri için NO ve NB üyelik fonksiyonlarında değer üretilmektedir. Hata değişimi değeri da iki üyelik fonksiyonunda değer üretmektedir. Buna göre e= -24 ve Δe = -18 değerleri için FLC mümkün olan 49 kuraldan sadece 4 kural çalıştırılmaktadır. Bu kurallar şunlardır:

IF	e=NO	AND	Δe=NK	THEN	u=NB
IF	e=NO	AND	Δe=NO	THEN	u=NB
IF	e=NK	AND	Δe=NK	THEN	u=NB
IF	e=NK	AND	Δe=NO	THEN	u=NB



Şekil 4.13 Girişlere göre bulanık kuralların gerçekleştirilmesi

Sisteme uygulanacak değer bu kurallar sonucuna göre üretilecektir. İşlem sonunda elde edilecek u kontrol girişi için 4 değer üretilecektir. Kural tabanda üretilen bu 4 değer sonra durulama ve gerçek değerlere dönüş işlemleri yapılarak sisteme uygulanacak tek bir değere dönüştürülecektir. Bu son çıktı değerini durulama yani gerçek değere dönüştürmek gerekmektedir (Şekil 4.13).

Çünkü son çıktı değerimiz halen bulanık bir değerdir. Durulama için ağırlık merkezi (center of gravity-COG) bulma yöntemi kullanılırsa; u kontrol işareti değerini bulmak için aşağıdaki formül kullanılır.

$$u = \frac{\sum_{i=1}^{n} \mu_i(u_i)\, u_i}{\sum_{i=1}^{n} \mu_i(u_i)} = \frac{0.6 \cdot (-45) + 0.2 \cdot (-45) + 0.4 \cdot (-45) + 0.2 \cdot (-45)}{(0.6 + 0.2 + 0.2 + 0.4) = -45} = -45$$

Burada u : sisteme uygulanacak crisp değeri
n : Kural sayısını
ui . : i. kural sonucu elde edilen değer
μ(ui) : bulanık kümesindeki üyelik derecesi

4.6 Bulanık Kontrol ve Klasik Kontrolün Karşılaştırılması

Bugün kullanılan konvansiyonel kontrol sistemlerinin çoğu geri besleme (feedback) üzerine kurulmuştur. Geri besleme, çıkıştan alınan sinyalin istenen sinyal ile karşılaştırılıp aradaki fark sinyalini (error signal) çıkış sinyalindeki hatayı düzeltecek şekilde sistem girişine vermek olarak tanımlanabilir. Geri besleme, bu tanımı ile insana algısal olarak uygulaması kolay gibi görünüyor olsa da, uygulamada tamamıyla otomatik bir kontrol sisteminin geri besleme ile dizayn edilmesi bir mühendis için gerçekten zor bir olaydır. Bunun en büyük sebebi ise kontrol mekanizmalarındaki (sensörler, aktivatörler gibi...) zaman gecikmeleridir.

Bir geri beslemeli otomatik kontrol sisteminde etkin bir kontrol için bir çok sistem parametresinin bilinmesi lazımdır. Fakat bu bahsedilen zaman gecikmesi yüzünden bu parametrelerin aynı anda bilinmesi her zaman mümkün olmayabilir. Eğer sistemde, dışarıdan gelen olağandışı bir gürültü yüzünden ani bir değişiklik olursa, kontrol sistemi çıkışları mümkün olduğu kadar çabuk düzeltmeye çalışacaktır. Fakat eğer sistem çok kuvvetli olarak bu ani değişikliğe cevap verirse, sistem davranışının kararsız olma ihtimali yükselir. Bu gibi problemler

yüzünden sadece basit bir geri besleme ile etkin bir tam otomatik kontrol sistemi gerçekleştirmek mümkün olmayabilir. Fakat, günümüz kontrol teorisi bu problemlere bir takım çareler bulmuştur. Birisi konvansiyonel kontrolde dizayn ve ayarlama metodlarının tamamen gelişmiş matematiksel modellemelere dayandırılmasıdır. İkinci olarak, PID kontrol sistemlerinin geliştirilmesidir (Tremaine, B. P., 1992).

Bulanık kontrole baktığımızda da tamamına yakın örneklerde geri besleme yapısı üzerinde kurulduğunu görmekteyiz. Ayrıca bulanık kontrolde geleneksel kontrol kuramındaki PID geri besleme yapısı da kullanılabilmektedir. Bu özellikler bulanık kontrol teorisine geleneksel kontrolün bazı güzel ve gelişmiş taraflarını alma imkanını vermektedir. Geleneksel kontrolde doğrusal olmayan kontrol sistemleri ileri bir konu olarak düşünülmektedir. Fakat bu düşünce bulanık kontrol için geçerli değildir. Bulanık kontrol değişkenler arasında karmaşık ve doğrusal olmayan ilişkilerde bile aynı mantığa sahiptir ve pratik olarak uygulaması nispeten daha kolaydır.

FLC'nin normal PID kontrolörlere karşı bazı avantajları vardır. FLC kontrolörler ilgili sensör verilerinin zayıflamasına ve bozunumuna karşı daha kuvvetlidirler ve sistem parametrelerinin değişmesine karşı tekrar dizayn etmeyi gerektirmezler. Doğrusal olmayan kontrolde de normal PID'e göre performansları daha yüksektir. FLC kontrolörlerin dizaynı PID kontrol sistemlerine göre daha kolaydır ve az elektronik eleman gerektirir. Bu yüzden maliyetleri de düşüktür. Bu sebeple de PID kontrolünün rahatça kullanıldığı yerlerde bile FLC kullanılması yaygınlaşmıştır. Bulanık Mantığın en iyi uygulama alanları Doğrusal olmayan (non-lineer), Tam olarak bir matematiksel model ile tanımlanamayan Zamanla değişen sistemlerdir. Bulanık mantık kontrolün bahsedilen avantajlarına karşın bazı dezavantajlara da sahiptir;

1. Bulanık kontrolde kullanılan kurallar deneyime çok bağlıdır.
2. Bu durum sistemi yeterince tanımayan birinin yazacağı kurallar ile FLC'nın sistemi istenen şekilde kontrol etmesini zorlaştırır.
3. Üyelik fonksiyonlarının seçiminde kesin bir yöntem yoktur. En uygun fonksiyon deneme ile elde edilebilir.
4. Bu da sağlam bir kontrolör gerçekleştirilmesini geciktirir.
5. Yazılan kurallar ile sistemin bir kararlılık analizi yapılamaz. Sistemin nasıl cevap vereceği önceden kestirilemez.

Çizelge 4.3 Bulanık Mantık Kontrol ile Klasik kontrolün karşılaştırılması

Klasik Kontrol	Bulanık Kontrol
Proses için matematiksel modele ihtiyaç duyar	Kontrol için uzman deneyim ve tecrübelerine ihtiyaç duyar.
Proses değişkenlerinin ölçümleri doğru ve kesin olmalıdır.	Kesin olmayan bilgileri kullanabilir
Karmaşık sistemlerde denetleyici de karmaşık olacağından uygulamaya koymak ekonomik olmayabilir.	Ucuz sensörler sayesinde prosesin ölçümünde esneklik kazandırır.
	Hızlıdır
	Uygulamaya geçirilişi kolaydır.
	Prosesin matematiksel modeline gereksinimi yoktur.

Bu nedenle FLC ilk önce sistemin simulasyonu üzerinde uygulanmakta ve alınacak olumlu sonuçlardan sonra gerçek sisteme uygulanmaya başlanılmaktadır.

5. MODEL TEMELLİ BULANIK KONTROL KURAL TABANI ÖĞRENİLMESİ ve DC MOTORA UYGULANMASI

5.1 Bulanık Model Referans Öğrenme Algoritması

Sistemdeki parametre değişimlerine göre kontrollerin katsayılarını ayarlayan adaptasyon mekanizması gibi geleneksel bir algoritma yerine bir bulanık adaptif metodu kullanılabilir. Yani bu kazanç katsayıların hesaplanması için bulanık mantık kullanılabilir. Sisteme ait bulanık kuralları sistemin iç yapısından ve sistem davranışlarından çıkarılır. Yalnız bütün sistemler için bulanık kuralların çıkarılması mümkün değildir. Çok değişken davranış gösteren ve nonlineer olan sistemlerde bulanık kurallarının çıkarılması için literatürde çeşitli yöntemler kullanılmaktadır. Bu çalışmada Bulanık Model Referans Öğrenme Algoritması (FMRLC) (Layne J.R., Passino K.M., 1993) öğrenmeye dayalı olduğundan kullanılmaktadır. FMRLC algoritması, çalışmamızda kullandığımız dc motor için PD-tip bulanık kontroller dizaynı için kullanılacaktır. Bunun motor transfer fonksiyonu üzerindeki sonuçları, tasarlanacak Genetik Algoritma temelli PI-tip bulanık kontroller ile karşılaştırılmasında kullanılacaktır.

Öğrenme kontrol algoritması burada direk olarak bir bulanık kontrolcu üzerine dayandırılmıştır. Genelde, bir "bulanık kontrolcu" bilgisayar algoritmasında kullanmak üzere bir uzman insanın prosesi nasıl kontrol ettiği hakkındaki bilgisini kullanmak için bulanık sistem kullanır. Uzmanın sistem hakkındaki bilgi ve tecrübesi bir bulanık kontroller için bir *ön bilgi* olarak kullanılır (Kwong W.A., Passino K.M., 1996).

FMRLC algoritması sistemin nasıl davrandığını gösteren bir referans model kullanır bu da bulanık kontrolörün temel bilgisini sağlar ve temel bilgi değişimlerini sağlayan kapalı döngü geri besleme performansını artırır. Sonuç olarak, bu algoritma bir bulanık model referans öğrenen kontroller (FMRC) olarak adlandırılabilir. Bu algoritma, davranışı hakkında yeterince bilgi sahibi olunmayan veya çok değişiklik gösteren sistemlerde kullanılabilir.

FMRLC öğrenme mekanizması aşağıdaki amaçlar için kullanılır.

a) Bir bulanık kontrol sisteminden $y_r(kt)$ ve $y(kt)$ değerlerini gözetler.

b) Bulanık kontrol sisteminin performans karakteristiğini gösterir.

c) Otomatik olarak bulanık kontrolleri kural tabanı düzenleyerek ayarlar ve/veya sentez eder. Öyleki önceden belirtilen performans hedeflerini karşılaştırır. Bu performans hedefleri, referans model yoluyla karakterize edilir.

5.2 FMRC Sistemini Oluşturan Kısımlar

5.2.1 Bulanık Kontrol Sistemi

Prosesin r-boyutlu vektör ile gösterilen $u(kT)=[u_1(kT),.....,u_r(kT)]^T$ girişlerine ve s-boyutlu vektör ile gösterilen $y(kT)=[y_1(kT),.....,y_s(kT)]^T$ çıkışlarına sahip olduğu kabul edilsin. Bu çalışmada ise proses girişi u(kT) ve çıkışı da y(kT) dır. Burada T örnekleme peryodunu temsil etmektedir.

Şekil 5.1 Tek giriş ve tek çıkışlı proses blok diyagramı

Bulanık kontrollerin girişleri ise, proses çıkışı y(kT) ve referans giriş $y_r(kT)$'nın fonksiyonu olarak tanımlanmaktadır. Bulanık kontroller girişleri $e(kT)=[e_1(kT),.....,e_s(kT)]^T$ ve hatadaki değişim $c(kT)=[c_1(kT),....., c_s(kT)]^T$ olarak tanımlanırsa

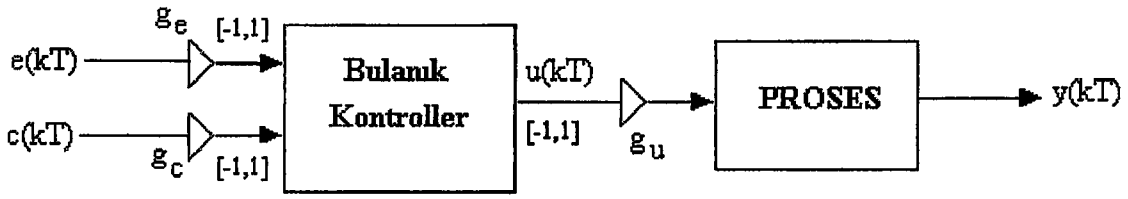
$$e(kT) = y_r(kT) - y(kT) \tag{5.1}$$

$$c(kT) = \frac{e(kT) - e(kT-T)}{T} \tag{5.2}$$

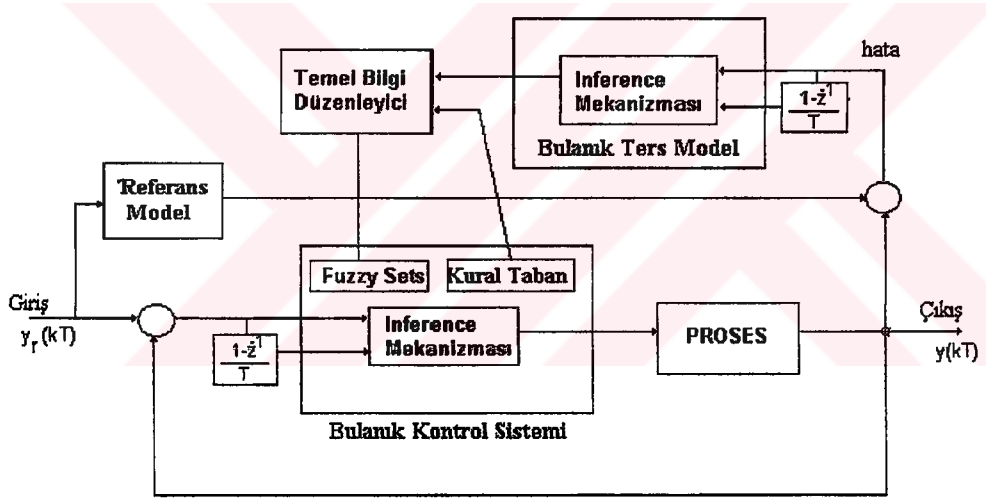
olur. Burada $y_r(kT) = [y_{r1}(kT),....., y_{rs}(kT)]^T$ istenen proses çıkışını gösterir. Bu yapısıyla kontroller PD-tip bir bulanık kontrollerdir.

Bulanık kontrol teorisinde, kontrolcunun giriş ve çıkışının alabileceği değerler aralığı, parametre "tanım uzayı" olarak adlandırılır. Daha esnek kontroller gerçekleştirmek için herbir proses girişi sabit ölçekleme katsayıları kullanılarak [-1,1] tanım uzayına kaydırılarak normalize edilir. Burada Şekil 5.3'de görülen FMRLC algoritması blok şemasındaki bulanık kontrol sistemi dizaynında g_e, g_c ve g_u ölçekleme katsayıları sırasıyla, e(kT) hata değeri, c(kT) hatadaki değişim ve u(kT) kontroller çıkışının çalışma aralığına normalizasyonu için kullanılmaktadır. Bulanık kontrollerin bir girişi bu durumda $g_e * e(kT)$ dır ve g_e öyle seçilmelidir ki $g_e * e(kT)$'in alacağı değerler [-1, 1] tanım uzayı içinde olmalıdır. Aynı şekilde g_u da proses girişi için izin verilen değerler aralığı kullanılarak seçilir. Yani bir anlamda g_u

kontroller çıkışı u(kT) yı prosesin kullanabileceği tanım uzayına taşımalıdır. Hata değişimi için ise g_c katsayısı, c(kT) nın alacağı normal değerleri belirlemek için sisteme değişik girişler verilerek deneysel olarak belirlenir, sonra bu değerleri [-1,1] aralığına taşıyacak şekilde belirlenir. Şekil 5.2' de bu katsayıların blok şema üzerindeki yerleri gösterilmektedir.



Şekil 5.2 Bulanık kontrol giriş ve çıkışlarında kullanılan ölçekleme katsayıları



Şekil 5.3 Bulanık Model Referans Kontrol (FMRLC) öğrenme algoritması blok şeması

Burada FMRC algoritmasında çok girişli - tek çıkışlı (MISO) Bulanık kontroller kullanılmıştır. Bulanık kontroller için bilgi tabanı (knowledge base) aşağıdaki şekildeki kontrol kurallarından üretilen n.ci proses girişiyle birleştirilir.

IF e_1 is E_1^j **and**....**and** e_s is E_s^k **and** ....**and** c_1 is C_1^l **and**....**and** c_s is C_s^m (5.3)
THEN u_n is $U_n^{j..k,l..m}$

Burada e_i ve c_i bulanık kontrollerin girişlerini ifade eden "dilsel değişkenleri" göstermektedir. u_n kontroller çıkışı ifade eden dilsel değişkeni gösterir. E_i^k ve C_i^k ise sırasıyla

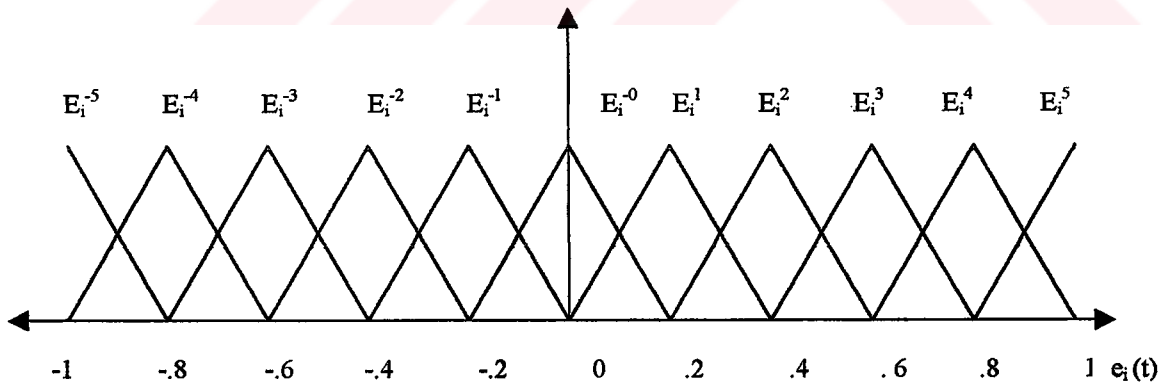
e_i ve c_i ile ilişkilendiren dilsel değerleri göstermektedir. Böylece bir örnek olarak bir bulanık kontrol kuralı şöyle yazılabilir:

IF hata **is** positive-large **and** hata değişimi **is** negative-small
THEN proses-girişi **is** positive-big (5.4)

Burada e_1 = hata ve E_1^4 ="positive-large v.s . bu şekildeki kurallar bir kümesi, bir dinamik sistemin nasıl kontrol edildiğini karakterize eden "kural taban"ı oluşturur. Yukardaki kontrol kuralı, bulanık küme teorisi kullanılarak bir bulanık çıkarım formu oluşturulabilir.

IF E_1^j **and....and** E_s^k **and** ....**and** C_1^l **and....and** C_s^m **THEN** $U_n^{j..k,l..m}$ (5.5)

Burada E^j, C^l ve $U^{j..k,l..m}$ sırasıyla "e is E^j", "c is C^l" ve "u is $U^{j..k,l..m}$" dilsel durumlarını tanımlayan bulanık kümeleri gösterir. Yukarıdaki örnek kural için Şekil 5.4'deki tanım uzayındaki üyelik fonksiyonları e(t) için kullanılabilir. Aynı bulanık kümelerinin normalize edilmiş c(t) içinde kullanıldığı kabul edilsin. Bulanık kontroller çıkışı u(t) için tanım uzayı üzerindeki çıkış üyelik fonksiyonları bilinmediği kabul . edilmektedir; bu üyelik fonksiyonlarının nasıl olunması gerektiği FMRLC tarafından otomatik olarak belirlenmektedir.



Şekil 5.4 Bulanık kontroller değişkenleri için tanım uzayı üzerindeki bulanık kümeler

Böylece sistemi kontrol eden bulanık kontroller öğrenilmiş olacaktır. Başlangıçta bulanık kontroller prosesi nasıl kontrol edeceği hakkında birşey bilmemektedir. Örnek olarak eğer s=1 alınırsa bulanık kontrollerin bütün kuralları aşağıdaki formda olacaktır.

IF E_1^j **and** C_1^1 **THEN** $U_n^{j,1}$ (5.6)

Burada E_1^j ve C_1^1 üçgen tip giriş üyelik fonksiyonlarıdır. Aynı şekilde $U_n^{j,1}$ ise orijin merkezli ve taban genişliği 0.4 olan bir üçgen tip üyelik fonksiyonudur, (Şekil 5.7). Giriş ve çıkış değerlerinin üyelik derecesi hesaplanmasında aşağıdaki üyelik fonksiyonu kullanılmaktadır.

$$\mu_{E^i}(x) = \begin{cases} \max\left(\left[0,1+\frac{x-c_{E^i}}{w}\right]\right), & x <= c_{E^i} \quad i = 1...11 \\ \max\left(\left[0,1+\frac{c_{E^i}-x}{w}\right]\right), & x > c_{E^i} \quad i = 1...11 \end{cases} \tag{5.7}$$

Burada; C_E^i , üçgen tip E^i üyelik fonksiyonun merkezi ve w ise üyelik fonksiyonun taban genişliğinin yarısını göstermektedir.

Direkt olarak dizayn edilen geleneksel bulanık kontroller için kontrol kuralları başlangıçta belirtilir. bu kurallardaki $U_n^{j,1}$ bilgisi prosesi tanıyan bir uzmanın ortaya koyduğu *ön_bilgi* olarak ifade edilir. Fakat FMRLC'de ise sistem bu bilgiyi otomatik olarak öğrenir ve gerek gördüğü durumlarda $U_n^{j,1}$ bulanık kümelerini, performansı sürdürmek veya geliştirmek için değiştirir. Burada durulama işleminde standart ağırlık merkezi yöntemi (COG-center of gravity) kullanılmaktadır.

5.2.2 Referans Model

Bir sistemin davranışını incelemek için en basit ve yararlı metod, sistemin matematiksel modelinin çıkarılmasıdır. Referans model, kontroller için gerekli performansı sağlamak için kullanılır. Genelde referans model herhangi bir dinamik sistem tipinde olmaktadır (Lineer veya nonlineer, zamanla-değişen veya zamanla değişmeyen, kesikli veya sürekli v.s). Tüm sistemin performansı referans modele göre üretilen hata sinyali ile hesaplanır.

$$y_e(kT) = [\, y_{e1},.....,y_{es}]^T \tag{5.8}$$

Burada $y_e(kT) = y_m(kT) - y(kT)$dir. Verilen referans model, rise time ve overshoot gibi dizayn kriterlerinin karakteristiklerini ifade eder ve referans modelin girişi, $y_r(kT)$ referans girişidir. Eğer öğrenme mekanizması $y_e(kT)$'yı her zaman küçük olmaya zorlarsa kontrol edilen prosesin istenen performansı sağlanır. Eğer istenen performans sağlanırsa (yani $y_e(kT) \cong 0$) o zaman öğrenme mekanizması bulanık kontrollerde önemli değişiklikler

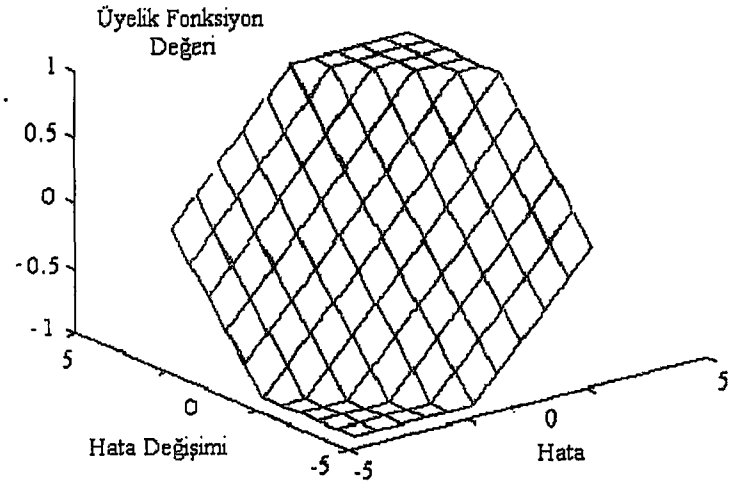
yaptırmayacaktır. Bir başka deyişle eğer $y_e(kT)$ büyükse istenen performans sağlanamaz ve öğrenme mekanizması bulanık kontrolleri ayarlamalıdır.

FMRLC nın amacı ele alınan kapalı çevrim kontrol sisteminin davranışını verilen referans modele benzetmektir. Burada referans model olarak sistemin nasıl davranmasını gösteren birinci mertebeden bir sistem alınmıştır. Yapılan çalışmalara dayanılarak dc motor için referans model olarak $G(s) = \frac{1}{s+1}$ birim basamak transfer fonksiyonu alınmıştır.

5.2.3 Öğrenme Mekanizması

Daha önce belirtildiği gibi, öğrenme mekanizması, kapalı döngü sistemin referans model gibi davranmasını sağlamak için bulanık kontrollerin kural tabanı değişim fonksiyonunu düzenler. Bu temel bilgi değişimi, kontrol edilen proses, referans model ve bulanık kontrollerden gelen verileri gözlenmesiyle gerçekleştirilir. Öğrenme mekanizması iki kısımdan oluşur:

a) Bulanık Ters Model : Bulanık ters model, bulanık kontrolör temelini oluşturan, istenen davranıştan değişime karşı kapalı çevrim sistem davranışındaki sapmayı gösteren $y_e(kT)$ değerini azaltmaya çalışan bir uyarlama döngüsünde çalışan bir kontrolör olarak çalışır. Esassen bulanık ters model, temel bilgi düzenleyici vasıtasıyla referans modelin davranışı gibi davranmasını isteyen kapalı-çevrim sistemden on-line veriler kullanarak bir bulanık kontrolör yapısı inşa eder. FMRLC, geleneksel sabit direkt bulanık kontrolörün off-line dizayn işleminde kullanılan bulanık kontrolün nasıl davranacağına karar veren önbilgi ile aynı türden bir bilgi kullanır. Bulanık ters modelde kullanılan kural-tabanı Şekil 5.5'te gösterilmiştir.



Şekil 5.5 Bulanık Ters Modelin için kullanılan Kural Tabanı 3D gösterimi

Sapma miktarı olan $y_e(kT)$ yı sıfır olmaya zorlamak için proses girişi p' de değişikler yapmak gerekmektedir.

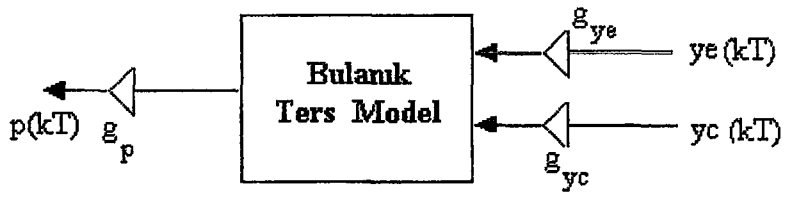
b) Temel Bilgi Düzenleyici: Temel bilgi düzenleyici, proses girişinde ihtiyaç duyulan değişiklikleri sağlamak için bulanık kontrollorun kural-tababnını düzenleme fonksiyonu olarak çalışır. Bulanık kontrolörün daha iyi performansa ulaşmasını sağlar. $y_e \approx 0$ yapmak için ihtiyaç duyulan girişteki gerekli değişikler hakkında bilgi (ters modelden) sağlar. Temel bilgi düzenleyici bulanık kontrolörün bilgi tabanını, önceden uygulanan kontrol hareketinin ters model çıkışları p(kT) ile belirtilen miktar kadar düzenlenmesini sağlayacak şekilde değiştirir.

Bulanık ters model, istenen proses davranışından sapmayı gösteren $y_e(kT)$'nın haritalama (mapping) fonksiyonunu düzenler. Bu fonksiyon ile $y_e(kT)$ 'yi sıfırlamaya zorlamak için gerekli olan $p=[p_1.....p_r]^t$ proses girişlerindeki değişikler sağlanır. Temel bilgi düzenleyici proses girişlerinde ihtiyaç duyulan değişiklikleri etkilemek için bulanık kontrollerin temel bilgi değişim fonksiyonunu düzenler. Bir kontrol mühendisi, bir prosesin ters modelini kabaca nasıl karakterize edeceğini bilir. Proses girişi p(kT)'de gerekli değişikliklerin yapılması için $y_e(kT)$ fonksiyonlarını haritalamak için bir bulanık kontrol sistem fikri ortaya atılmıştır(Layne J.R., Passino K.M., Yurkovich S., 1993).

Bu harita yapısında proses ters dinamikleri hakkındaki bilgi kullanıldığından, bulanık ters model olarak adlandırılır. Esasında bir bulanık kontrollere benzemektedir. Yapısında bulanık kontrollorde olduğu gibi bir kural taban bulunmakta ve giriş-çıkış değerlerini normalize etmek için g_{ye}, g_{yc} ve g_p katsayılarını içermektedir. Bulanık ters model girişleri $g_{ye} * ye$ ve $g_{yc} * yc$ dir (Şekil 5.6). Bulanık ters modelin kural tabanı proses girişi ile ilişkilendirildiğinde şekilsel olarak,

If Y^j_{el} and .....and Y^k_{es} and Y^l_{cl} and.....and Y^m_{cs} **Then** $P_n^{j..k,l..m}$ (5.10)

Burada Y^b_a ve Y^b_c , sırasıyla hata ye ve hata değişimi yc için b.ci bulanık kümeleri göstermektedir.



Şekil 5.6 Bulanık ters model blok yapısı

Çizelge 5.1 Bulanık ters kontrolcu kural tabanı

		e										
		-5	-4	-3	-2	-1	0	1	2	3	4	5
de	-5	-1	-1	-1	-1	-1	-1	-0.8	-0.6	-0.4	-0.2	0
	-4	-1	-1	-1	-1	-1	-0.8	-0.6	-0.4	-0.2	0	0.2
	-3	-1	-1	-1	-1	-0.8	-0.6	-0.4	-0.2	0	0.2	0.4
	-2	-1	-1	-1	-0.8	-0.6	-0.4	-0.2	0	0.2	0.4	0.6
	-1	-1	-1	-0.8	-0.6	-0.4	-0.2	0	0.2	0.4	0.6	0.8
	0	-1	-0.8	-0.6	-0.4	-0.2	0	0.2	0.4	0.6	0.8	1
	1	-0.8	-0.6	-0.4	-0.2	0	0.2	0.4	0.6	0.8	1	1
	2	-0.6	-0.4	-0.2	0	0.2	0.4	0.6	0.8	1	1	1
	3	-0.4	-0.2	0	0.2	0.4	0.6	0.8	1	1	1	1
	4	-0.2	0	0.2	0.4	0.6	0.8	1	1	1	1	1
	5	0	0.2	0.4	0.6	0.8	1	1	1	1	1	1

Bulanık ters model de 11x11 'lık kural taban kullanılmaktadır. Bu çalışmada bulanık ters model için Çizelge 5.1'de görülen kural taban kullanılmıştır (Layne J.R., Passino K.M., Yurkovich S., 1993). Giriş çıkış değişkenleri için de üçgen tip üyelik fonksiyonları ve durulama işleminde ağırlık merkezi yöntemi kullanılmaktadır.

Giriş değerindeki gerekli değişiklikleri belirten bilgi p(kT) vektörü tarafından ifade edilmektedir. Temel bilgi düzenleyici, (kT-T) anında uygulanan kontrol işaretini p(kT) miktarı kadar değiştirecektir. Bu andaki kontrol işareti u(kT-T) olsun. Sistemin şu anki performansının iyi veya kötü olduğunu belirtmek için e(kT-T) ve c(kT-T) aynı andaki hata ve hatanın değişimi değerleri olsun. İstenen u(kT-T) + p(kT) çıkış değerini üretmek bulanık kontrollerin kural tabanın değiştirilmesi gerekmektedir. Bulanık kontroller çıkışı için simetrik $U_n^{j,\ldots k,l,\ldots m}$ üyelik fonksiyonları tanımlanmış ve bulanık kümeler ile bağlantılı üyelik fonksiyonlarının merkezleri $c_n^{j,\ldots k,l,\ldots m}$ olsun. Öğrenme işleminin başlangıcında $C_n^{j,\ldots k,l,\ldots m}(0)=0$ dır. . Temel bilgi düzenleyici bir önceki kontrol işareti u(kT-T) iyileştirmek için bulanık çıkarımda ilgili $U_n^{j,\ldots k,l,\ldots m}$ bulanık kümelerinin üyelik fonksiyonlarının merkezlerini kaydırarak düzenler. Başlangıçta tümünün sıfır merkez noktasında olduğu kabul edilmektedir. Bu düzenleme p(kT) ile belirtilen miktar kadar üyelik fonksiyonlarının merkezleri kaydırılarak yapılır. Buna göre

$$C_n^{j,\ldots k,l,\ldots m}(kT) = C_n^{j,\ldots k,l,\ldots m}(kT\text{-}T) + p(kT) \qquad (5.11)$$

olur.

Belirli bir bulanık çıkarımdaki bir kuralın katkı derecesi bunun aktivasyon seviyesi tarafından belirlenir.

$$\delta_n^{j....k,l...m}(t) = \min\left(\mu_E^j(e_1(t)), \ldots\ldots \mu_E^k(e_s(t)), \mu_E^l(C_1(t)), \ldots\ldots \mu_E^m(c_s(t))\right) \qquad (5.12)$$

Burada μ_A A bulanık kümesinin üyelik fonksiyonunu göstermektedir. Yalnız aktivasyon seviyesi $\delta_n^{j....k,l...m}$ (kT-T) > 0 olan kurallar düzenlenmeye tabi tutulur, geri kalanlar aynı bırakılır. Buradaki kural taban düzenleme prosedürü lokal öğrenme şeklinde çalışır ve bir hafiza kullanır. Yani, kural tabanın farklı bölümleri, sistem için farklı çalışma şartlarına bağlı olarak doldurulur ve kural tabanı güncellenirken diğer kurallar etkilenmez.

Örneğin, direkt bulanık kontroller ve bulanık ters modelin her ikisi için normalizasyon katsayılarının 1 olduğunu kabul edelim. Bulanık ters model bir p_n (kT) = 0.5 çıkışını üretmiş olsun. Bu sistemin (kT-T) anında prosesin çıkış değerinin, performansını geliştirmek yani hata girişini $y_{e1} \approx 0$ olmaya zorlamak için u(kT-T) + 0.5 olması gerektiğini göstermektedir. Bu anda e_1(kT-T)=0.75 ve c_1(kT-T)=-0.2 olsun. O zaman yalnız

If E_1^3 and C_1^{-1} **Then** $U_n^{3,-1}$

If E_1^4 and C_1^{-1} **Then** $U_n^{4,-1}$

Kuralların aktivasyon değeri sıfırdan büyük olur ($\delta_n^{3,-1} = 0.25$ ve $\delta_n^{4,-1} = 0.75$). Böylece yalnız kuralların sonuç bulanık kümeleri olan $U_n^{3,-1}$, $U_n^{4,-1}$ düzenlenir. Bu bulanık kümeleri düzenlemek için p(kT)=0.5 olduğundan merkezleri bu değer kadar 1'e doğru kaydırılır.

5.3 FMRLC İle DC Motor İçin Bulanık Kontroller Tasarımı

Bu çalışmada yukarda ayrıntılarıyla anlatılan Bulanık Model Referans Öğrenme Algoritması (FMRLC) öğrenmeye dayalı olduğundan DC motorun hız kontrolörünü sağlayacak PD-tip Bulanık kontrolörün kural-tabanı bu metod kullanılarak, insan tecrübesine gereksinim duyulmaksızın öğretilmiştir (Bulut M., Cansever G., 2000).

Doğru akım (DC) motoru iki temel bileşene sahiptir; alan sargıları ve rotor üzerinde yerleştirilmiş değişmeyen armatür sargıları. Doğru akımın alan sargıları arasından geçirilmesi motor hava aralığı akısının alanını ayarlayan bir uyartım akımı doğurur. Motor üzerindeki yük torkunun (T_L) sıfırdan farklı olması durumunda kontrol edilecek sistem olan motorun transfer fonksiyonu, $v_a(s)$ giriş ve $W(s)$ çıkış olmak üzere aşağıda çıkarılmıştır.

$$G(s)=\frac{w_m(s)}{v_a(s)}=\frac{K_i}{s^2J\cdot L_a+s(B\cdot L_a+J\cdot R_a)+B\cdot R_a+K_b\cdot K_i} \qquad (5.13)$$

Bu tezde kullanılan DC motora ait parametreler motorun üretici firması olan Sinano Electric Co. (Japonya) firmasıyla yapılan iletişim yoluyla elde edilmiş olup aşağıdaki gibidir:

Atalet Torku (J) : 0.225 g-cm-s^2

Sürtünme katsayısı (B) : 1.75 Kg-cm

Armatür direnci (Ra) : 12.6 ohm

Armatür Endğktansı (La) : 9 mH

Tork sabiti (Ki) : 1.5 Kg-cm/A

Geri besleme sabiti (Kb) : 15.4 mV/rpm

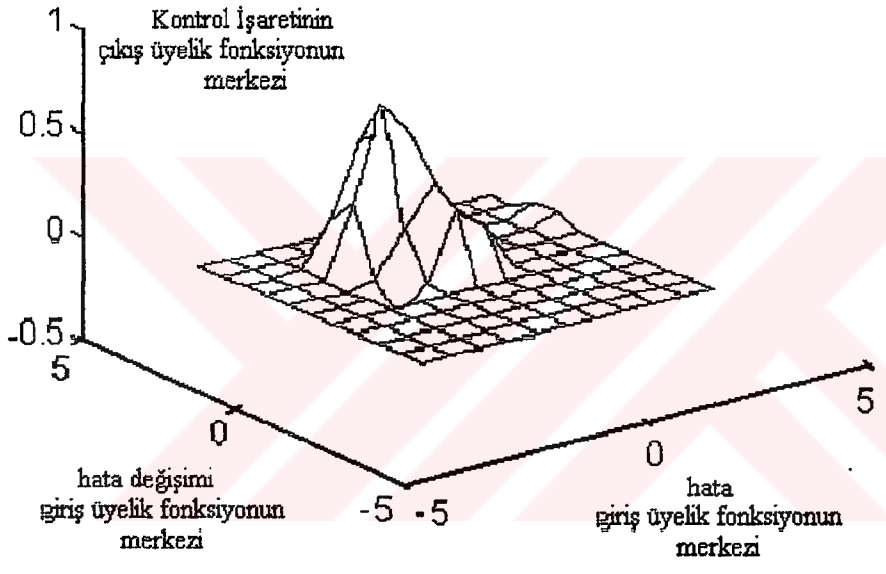
FMRLC ile dizayn edilen PD-tip bulanık kontrolörü performans testi için dijital simulasyon çalışmaları üzerinde model temelli yapı düzenlenmiştir. Motor, birim referans hızında yüksüz olarak start almaktadır. Burada öğrenme işlemi, bir fonksiyonun bilinmeyen büyüklüğünün ardışıl yaklaşımı veya tahmin etme olarak görülebilir. FMRLC algoritması çalışırken, bulanık çıkış kümeleri öğrenme mekanizması tarafından doldurulur. Öğrenme işleminden sonra, bulanık kontrolör üyelik fonksiyonlarının taban merkezini gösteren değerler ile doldurulur. Bu değerler herbir kural için üyelik fonksiyonunu göstermektedi. Öğrenilen kural tabana ait değerler Çizelge 5.2' de veilmiştir.

Çizelge 5.2 FMRLC algoritması ile öğrenilen PD-tip Bulanık Kontrolcuya Kural Tabanı FAM tablosu

		E										
		-5	-4	-3	-2	-1	0	1	2	3	4	5
de	-5	0	0	0	0	0	0	0	0	0	0	0
	-4	0	0	0	0	0	0	0	0	0	0	0
	-3	0	0	0	0	0	0	0	0	0	0	0
	-2	0	0	0	0	0	0	0	0	0	0	0
	-1	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	-0.1097	0.0065	0.3393	0.2231	0	0	0	0
	1	0	0	-0.0160	0.0190	0.2328	0.4209	0.2231	0	0	0	0
	2	0	0	-0..0043	0.3468	0.7954	0.6516	0	0	0	0	0.0542
	3	0	0	0.0117	0.2181	0.5691	0.5701	0	0	0	0	0.0542
	4	0	0	0	0	0	0	0	0	0	0	0
	5	0	0	0	0	0	0	0	0	0	0	0

Sistem parametrelerinde değişmeler olması durumunda, öğrenme işlemini bitirip uyuma moduna geçen FMRLC tekrar bu yeni durumu öğrenmeye başlar ve kural tabanda gerekli kuralları güncelleyerek yeni bir kural taban oluşturur. Böylece sistemdeki değişikleri gözleyerek yeni duruma uygun kural taban oluşturmakta ve bununla sistemi kontrol etmektedir.

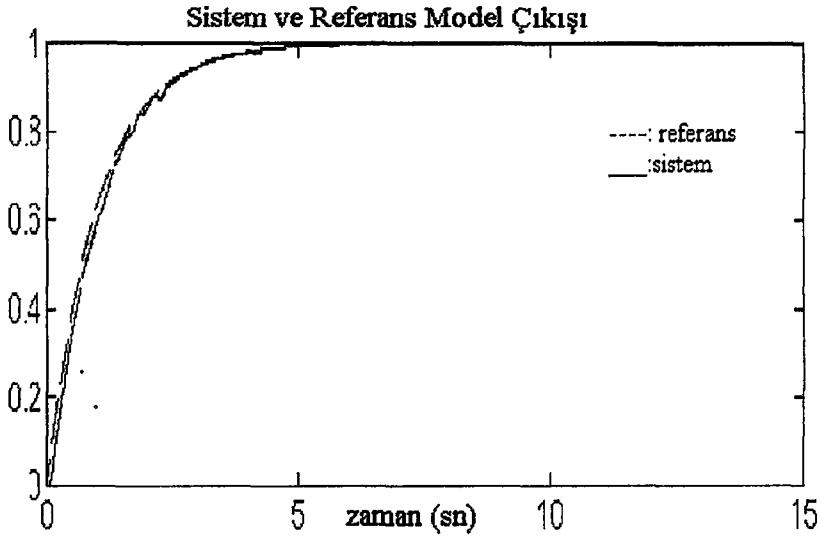
Doğru akım motoru içim model sistem yoluyla öğrenme boyunca, model çıkışı ve yapılandırılan bulanık kontrolör çıkışı Şekil 5.8-a,b'de gösterilmektedir. Öğrenmeden sonra Bulanık kontrolör kural tabanın 3 boyutlu haritası da Şekil 5.7'de gösterilmiştir.



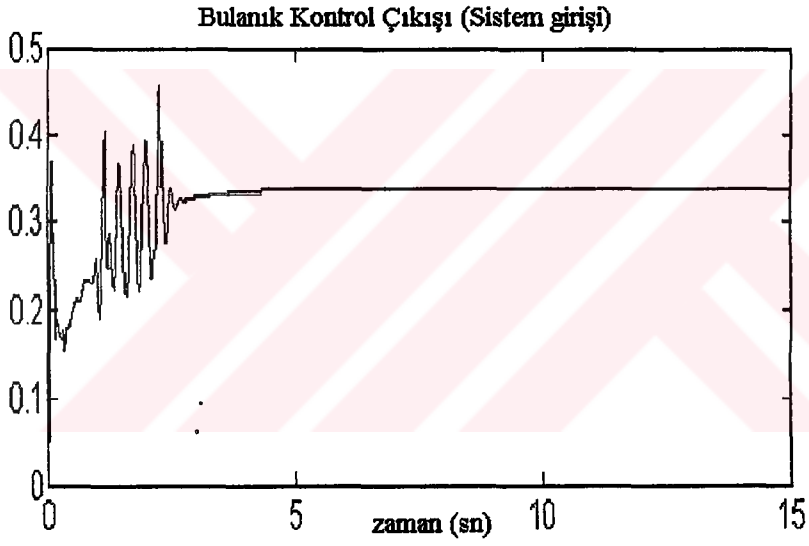
Şekil 5.7 Bulanık kontrolcu için öğrenme işleminden sonra yapılandırılan kural tabanın 3D görünümü

5.4 Elde Edilen Sonuçlar

Son yıllarda insan uzman bilgisi yardımı olmaksızın uygun düzenlenmiş bulanık kural tabanı geliştirmede birçok araştırma vardır. Ele alınan bir sistem için uygun bulanık kontrolör kural tabanı geliştirmesi uzun zaman alan tecrübelerden geçmektedir. Halbuki FMRLC algoritması doğrudan bulanık kontrolör bilgi tabanını dizayn etmek ve yeni durumlara karı düzenlemek için otomatik öğrenmeye bağlı bir metod sağlamaktadır. Bir bulanık kontrolör herhangi bir sistemin kontrolünde iyi bir performans gösterebilir, fakat tahmin edilmeyen sistem parametre değişimleri veya herhangi yük değişimleri durumunda yeterli olmamaktadır. Çünkü dizayn edilen kural tabanın başarılı olabilmesi için tüm olası durumları içermesi gerekmektedir.



(a)

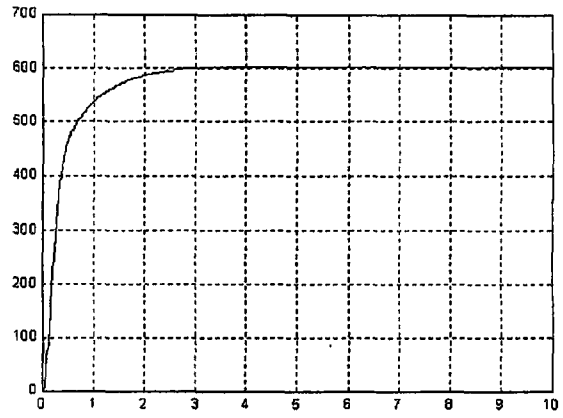


(b)

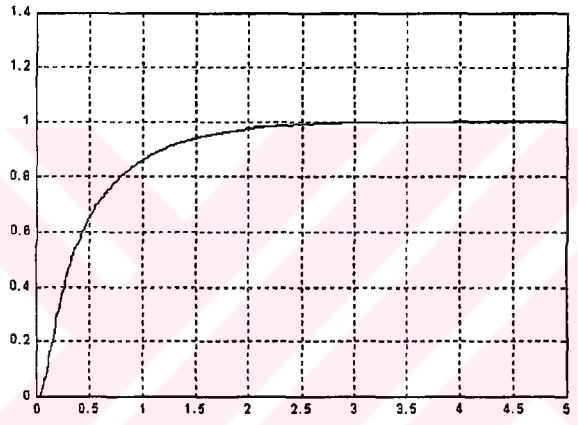
Şekil 5.8 Öğrenme aşaması süresince a) sistem ve model çıkışı b) Bulanık kontrolör çıkışı

Burada bulanık mantık temelli doğru akım motor kontrolü için PD-tip bulanık kontrolcuya kural tabanın otomatik olarak öğrenilmesi ile üzerine çalışılmıştır. Bu nedenle bu amaç için FMRLC algoritması DC motor kontrol sistemi için analiz edilmiş ve kullanılmıştır. Şekil 5.8'teki sonuçlardan da görüleceği şekilde bu metod kullanılarak dizayn edilen kural tabana sahip bulanık kontrol ile sistem çıkışı yaklaşık 3-4 sn'de referans hıza ulaşmıştır.

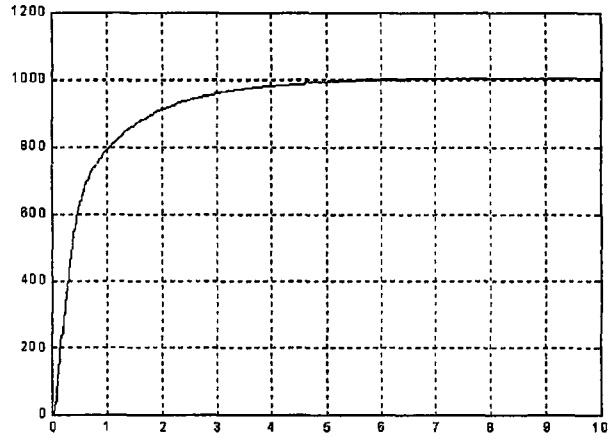
FMRLC algoritması ile öğrenme aşamasından sonra elde edilen PD-tip bulanık kontrolörün çeşitli referans değerlerinde DC motora uygulanması ile edilen sonuçlar Şekil 5.9a,b,c'de verilmektedir.



(a)



(b)



(c)

Şekil 5.9 DC motora uygulanan PD-tip bulanık kontrolcunun a) birim basamak cevabı b) 600 dev/dak referans girişe cevabı c) 1000 dev/dak referans girişine cevabı

6. SİSTEMDE KULLANILAN GENETİK ALGORİTMALAR

6.1 Genetik Algoritmaların Tarihçe ve Tanımı

Michigan Üniversitesinde psikoloji ve bilgisayar bilimi uzmanı olan John Holland bu konuda ilk çalışmaları yapan kişidir. Mekanik öğrenme (machine learning) konusunda çalışan Holland, Darwin'in evrim kuramından etkilenerek canlılarda yaşanan genetik süreci bilgisayar ortamında gerçekleştirmeyi düşündü. Tek bir mekanik yapının öğrenme yeteneğini geliştirmek yerine böyle yapılarda oluşan bir topluluğun çoğalma, çiftleşme, mutasyon, vb. genetik süreçlerden geçerek başarılı (öğrenebilen) yeni bireyler oluşturabildiğini gördü. Çalışmalarının sonucunu açıkladığını kitabının 1975'te yayınlanmasından sonra geliştirdiği yöntemin adı Genetik Algoritmalar (ya da kısaca GA) olarak yerleşti. Ancak 1985 yılında Holland'ın öğrencisi olarak doktorasını veren David E. Goldberg adlı inşaat mühendisi 1989 da konusunda bir klasik sayılan kitabını yayınlayana dek genetik algoritmaların pek pratik yararı olmayan bir araştırma konusu olduğu düşünülüyordu. Halbuki Goldberg'in gaz boru hatlarının denetimi üzerine yaptığı doktora tezi ona sadece 1985 National Science Foundation Genç Araştırmacı ödülünü kazandırmakla kalmadı, genetik algoritmaların pratik kullanımının da olabilirliğini kanıtladı. Ayrıca kitabında genetik algoritmalara dayalı tam 83 uygulamaya yer vererek GA'nın dünyanın her yerinde çeşitli konularda kullanılmakta olduğunu gösterdi (Goldberg, D.E., 1989).

Genetik algoritmalar, doğal seçme ve doğal genetik kurallara dayanan bir arama türüdür. Doğal seçme, doğa koşullarına en fazla uyum sağlamış olan canlının neslini devam ettirmesi, uyum sağlayamamış olan türlerin ise elenmesidir. Canlılar nesilden nesile genlerini aktarırken, bu genler de doğal genetik kurallara göre başka genlerle çaprazlanır, değişime uğrar ve yeni genleri oluştururlar. Genetik algoritmalar da tabiattaki bu iki oluşumu birleştirerek , en iyi (optimal nokta)'yı arar. Bir önceki neslin en uyumlu fertleri kullanılarak yeni neslin üyeleri oluşturulur. Bu yöntem sayesinde, klasik yöntemlerle çözülmesi çok zor kimi zaman da imkansız olan problemler çözülebilmektedir.

Genetik algoritma, doğadaki evrim mekanizmasını örnek alan bir arama metodudur ve bir veri grubundan özel bir veriyi bulmak için kullanılır. Genetik algoritmalar 1970'lerin başında John Holland tarafından ortaya atılmıştır. Genetik algoritmalar doğada geçerli olan en iyinin yaşaması kuralına dayanarak sürekli iyileşen çözümler üretir. Bunun için "iyi"nin ne olduğunu belirleyen bir uygunluk (fitness) fonksiyonu ve yeni çözümler üretmek için yeniden kopyalama (recombination), mutasyon (mutation) gibi operatörleri kullanır. Genetik

algoritmaların bir diğer önemli özelliği de bir grup çözümle uğraşmasıdır. Bu sayede çok sayıda çözümün içinden iyileri seçilip kötüleri elenebilir. Genetik algoritmaları diğer algoritmalardan ayıran en önemli özelliklerden biri de seçmedir. Genetik algoritmalarda çözümün uygunluğu onun seçilme şansını arttırır ancak bunu garanti etmez. Seçim de ilk grubun oluşturulması gibi rasgeledir ancak bu rasgele seçimde seçilme olasılıklarını çözümlerin uygunluğu belirler (Man K.F. , Tang K.S. , Kwong S. and Halay W.A., 1997).

6.2 Genetik Algoritmaların Çalışma Prensibi

Genetik algoritmanın çalışmasını aşağıdaki gibi özetleyebiliriz;

Adım 1: Olası çözümlerin kodlandığı bir çözüm grubu oluşturulur (çözüm grubu, biyolojideki benzerliği nedeniyle, toplum (population), çözümlerin kodları (string) da kromozom olarak adlandırılır).

Adım 2: Her kromozomun ne kadar iyi olduğu bulunur (fitness function).

Adım 3: Bu kromozomlar eşlenerek (matching), yeniden kopyalama (recombination) ve çaprazlama (crossover) operatörleri uygulanır. Bu sayede yeni bir toplum oluşturulur.

Adım 4: Yeni kromozomlara yer açmak için eski kromozomlar ortadan kaldırılır.

Adım 5: Tüm kromozomların uygunlukları tekrar hesaplanır.

Adım 6: Eğer jenerasyon süresi dolmamışsa 3. adıma gidilir.

Adım 7: O ana kadar bulunmuş en iyi kromozom sonuçtur.

Genetik Algoritma Prosedürü aşağıda verilmektedir;

Başla(1)

Parametrelerin Kodlanması

t =0; zaman sayacının sıfırlanması

P(t) Başlangıç Neslini Oluştur;

P(t) Neslinin Uyumluluk Değerlendirmesini Yap;

Devam (Bitiş Şartı sağlanmadığı sürece) **Et**

Başla (2)

t =t + 1; zaman sayacını artır

Selection P(t - 1) neslinden P(t) neslinin Seçilmesi

P(t) Neslinin Üremesi;

P(t) nesli içinde Çaprazlama

P(t) neslinin Mutasyona maruz kalması;

P(t) Neslinin Uyumluluk Değerlendir;

Son (2)

Parametreleri Gerçek Değerlerine Çevir

Son(1)

Prosedür işlemlerini daha ayrıntılı olarak açıklamak gerekirse ;

Adım-1. Bu adıma toplumda bulunacak birey sayısını belirleyerek başlanmaktadır. Kullanılacak sayı için bir standart yoktur. Genel olarak önerilen 100-300 aralığında bir büyüklüktür. Büyüklük seçiminde yapılan işlemlerin karmaşıklığı ve aramanın derinliği önemlidir. Toplum bu işlemden sonra rasgele oluşturulur.

Adım-2. Kromozomların ne kadar iyi olduğunu bulan fonksiyona uygunluk fonksiyonu denir. Bu fonksiyon işletilerek kromozomların uygunluklarının bulunmasına ise değerlendirme (evaluation) adı verilir. Bu fonksiyon genetik algoritmanın beynini oluşturmaktadır. Genetik algoritmada probleme özel çalışan tek kısım bu fonksiyondur. Uygunluk fonksiyonu kromozomları problemin parametreleri haline getirerek onların bir bakıma şifresini çözmektedir (decoding), sonra bu parametrelere göre hesaplamayı yaparak kromozomların uygunluğunu bulur. Çoğu zaman genetik algoritmanın başarısı bu fonksiyonun verimli ve hassas olmasına bağlı olmaktadır.

Adım-3. Kromozomların eşlenmesi kromozomların uygunluk değerlerine göre yapılır. Bu seçimi yapmak için rulet tekerleği seçimi (roulette wheel selection) , turnuva seçimi (Tournament Selection) gibi seçme yöntemleri vardır. Örnek olarak bu çalışmada kullanılan rulet tekerleği seçimi aşağıda açıklanmıştır.

1. Tüm bireylerin uygunluk değerleri bir tabloya yazılır.

2. Bu değerler toplanır.

3. Tüm bireylerin uygunluk değerleri toplama bölünerek [0,1] aralığında sayılar elde edilir.

4. Bu sayılar bireylerin seçilme olasılıklarıdır. Sayıların hepsi bir tabloda tutulur.

5. Seçilme olasılıklarını tuttuğumuz tablodaki sayılar birbirine eklenerek rasgele bir sayıya kadar ilerlenir. Bu sayıya ulaşıldığında yada geçildiğinde son eklenen sayının ait olduğu çözüm seçilmiş olur.

6. Bu yönteme rulet tekerleği seçimi ismi, bir daireyi, çözümlerin uygunluklarına göre dilimleyip çevirdiğimizde olacakların benzeşimi olduğu için verilmiştir. Rulet tekerleği seçimi çözümlerin uygunluk değerlerinin negatif olmamasını gerektirir. Çünkü olasılıklar negatif olursa bu çözümlerin seçilme şansı yoktur. Çoğunluğunun uygunluk değeri negatif olan bir toplumda yeni nesiller belli noktalara takılıp kalabilir. şeklinde verilebilir.

7. Çaprazlama (crossover) genetik algoritmanın motoru kabul edilir. Basitçe olay iki ebeveyn kromozomun arasında belirlenen parçaların takasıdır. Genetik algoritmalar bu olayın benzeşimini temelde; Tek noktalı (Single (one) point crossover), çok noktalı (Multi point crossover) Çaprazlama adı verilen iki yolla yapar. Genetik algoritmalarda ikilik dizi (binary string) çok kullanılır. Doğadaki genlerin benzeşimi 'bit'lerdir.

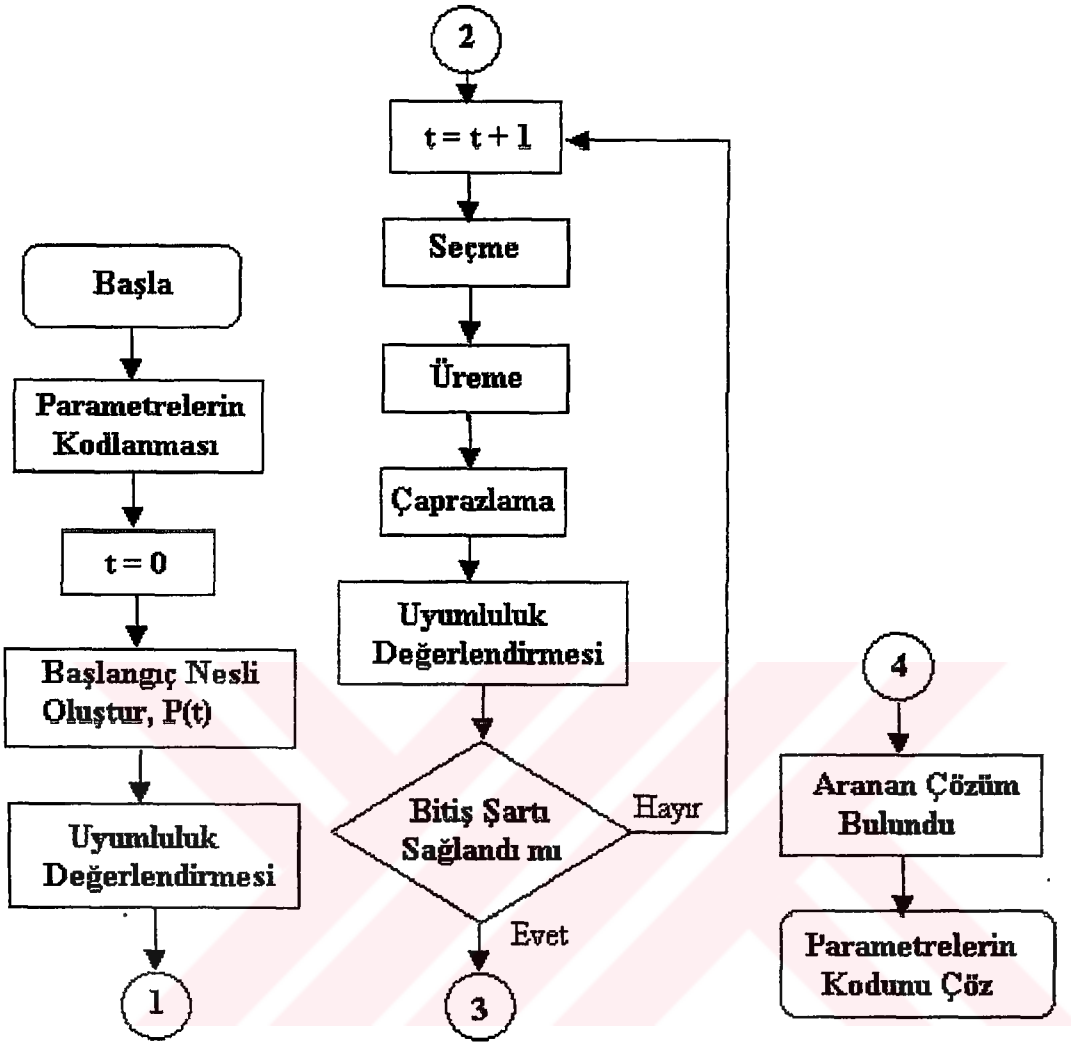
Çaprazlama toplumda çeşitliliği sağlar. İyi özelliklerin bir araya gelmesini kolaylaştırarak en iyiye yaklaşmayı sağlar. Değiştirme kromozomun bir parçasının dışarıdan değiştirilmesi şeklinde tanımlanır. Değiştirme görünüşte genetik algoritmanın dayanak noktasıdır, ancak etkisi bir çözüm üzerindedir. Bu da yalnız başına başarılı olmasını zorlaştırır. İkilik dizilerde değiştirme rasgele bir bit'in değiştirilmesiyle sağlanabilir. Çok düşük bir değiştirme olasılığı toplumda bazı özelliklerin kaybolmasına neden olabilir. Bu da en iyi sonuçların bulunmasına engeldir. Ancak yüksek bir değiştirme olasılığı da eldeki çözümleri bozarak sonuca ulaşmayı zorlaştırır. Çaprazlama ve mutasyon olasılıkları için kesin bir sayı yoktur. Mutasyon olasılığı 0.01-0.001, Çaprazlama (cross-over) olasılığı 0.5-1.0 aralığında tavsiye edilir.

Adım-4. Eski kromozomlar çıkartılarak sabit büyüklükte bir toplum sağlanır.

Adım-5. Tüm kromozomlar yeniden hesaplanarak yeni toplumunun başarısı bulunur.

Adım-6. Genetik algoritma defalarca çalıştırılarak çok sayıda toplum oluşturulup hesaplanır.

Adım-7. Toplumların hesaplanması sırasında en iyi bireyler saklandığı için o ana kadar bulunmuş en iyi çözüm çözümdür. Genetik algoritmanın yaptığı işleri temelini akış diyagramı olarak Şekil 6.1'de görebiliriz.



Şekil 6.1 Genetik Algoritmalar için çalışma prensibini gösteren akış diyagramı

6.3 Genetik Algoritmaların Kullanılma Nedenleri ve Uygulama Alanları

Öncelikle niye diğer yöntemlerin kullanılmadığı belirtilmelidir. Denklem en iyilemesinde (optimization);

1. Türev-İntegral temelli olanlar,

2. Numaralama (numeration) temelli olanlar,

3. Rastgele aramalar (random searches) olmak üzere üç tip çözümden bahsedilir:

Türev-İntegral hesaplamalarına dayanan hesaplama yöntemleri çok derinlemesine çalışılmıştır. Bu yöntemler fonksiyonun türevinin köklerinin fonksiyonun en küçük ve en büyük değer veren noktaları olmasından yararlanır. Gerçek problemler için sıfır veren

noktaları bulmak da ayrı bir problemdir. Diğer bir yöntem ise alınan bir noktadan sadece yukarı ilerleyerek en iyi sonucu bulmayı hedefler. Tepe tırmanma (hill climbing) denen bu yöntem fonksiyon grafiğinin tepelerini tırmanır. Ancak çok sayıda dönme noktası içeren bir fonksiyonda çok sayıda tepe oluşur. Hangi tepenin en iyi çözüm olduğunu bilenemez. Numaralama yöntemleri ise oldukça alışılagelmiştir. Sürekli olan gerçel sayı aralıkları belli sayıda parçaya ayrılarak parçalar denenir. Ancak problemler böyle çözmek için büyük olabilir. Bu yöntemin biraz daha geliştirilmiş şekli Dinamik Programlamayla (dynamic programming) oluşturulur. Parçalar arasından iyi görünenler seçilir. Bu parçalar parçalara ayrılarak işlem tekrarlanır. Bu yöntem de tepe tırmanma yöntemi gibi yanlış tepeleri araştırabilir. Dinamik Programlama tepelerin olmadığı aralıklarda başarılı ve hızlıdır .

En iyilemenin bir işin daha iyi yapılması, en doğru şekilde yapılması olmak üzere iki amacı vardır. Günümüzde rasgele aramaların kullanımı artmaktadır. Bu tip aramalar en iyilemenin daha iyi yapma amacını sağlamakta daha başarılıdırlar. İnsanların bilgisayarlardan genel beklentisi mükemmellik olduğu için bu tip aramalar başarısız görünebilir. Genetik algoritmalar klasik yöntemlerin çok uzun zamanda yapacakları işlemleri kısa bir zamanda çok net olmasa da yeterli bir doğrulukla yapabilir.

Genetik Algoritmaların diğer metodlardan farklarını birkaç maddede toplanabilir;

1. Genetik algoritma parametrelerin kodlarıyla uğraşır. Parametreler kodlanabildiği sürece algoritma için parametrelerin neyi temsil ettiği fark etmez.
2. Genetik algoritma bir tek yerden değil, bir grup çözüm içinden arama yapar..
3. Genetik algoritma ne yaptığı konusunda bilgi içermez, nasıl yaptığını bilir. Bu nedenle bir kör arama (blind search) metodudur.
4. Genetik algoritmalar olasılık kurallarına göre çalışır. Programın ne kadar iyi çalışacağı önceden kesin olarak belirlenemez. Ama olasılıkla hesaplanabilir.

Genetik algoritmaların en uygun olduğu problemler geleneksel yöntemler ile çözümü mümkün olamayan ya da çözüm süresi problemin büyüklüğü ile üstel orantılı olarak artanlardır. Bugüne kadar GA ile çözümüne çalışılan uygulama konulardan bazıları şunlardır.

Optimizasyon: GA, sayısal optimizasyon ve kombinetoral optimizasyon problemleri olan devre tasarımı, doğrusal olmayan denklem sistemlerinin çözümünde ve fabrika-üretim planlamasında kullanılır.

Otomatik Programlama (automatic programming): GA, bilgisayar programları yardımıyla network sıralamasında (sorting),ders programı hazırlanmasında kullanılır.

Makine öğrenmesi (machine learning): GA, robot sensorlerinde, neural networklerde,VLSI yonga tasarımı ve protein yapısal analizinde kullanır.

Ekonomi (economics): GA, ekonomik modellerin geliştirilmesinde ve işlemesinde kullanılır.

İmmün sistemler (Immune systems): GA, çok-gen'li ailelerin evrimi esnasında ve doğal immün sistem modellerinde kullanılır.

Popülasyon genetiği (population genetics): GA, evrim ile ilgili sorulara cevap bulmada kullanılır.

Evrim ve öğrenme (evolution and learning): GA, fertlerin öğrenmesini ve türlerin evrilmesinde kullanılır.

Sosyal sistemler (social systems): GA, sosyal sistemlerin analizinde kullanılır.

6.4 Genetik Algoritmaların Performansını Etkileyen Nedenler

Kromozom sayısı: Kromozom sayısını arttırmak çalışma zamanını arttırırken azaltmak da kromozom çeşitliliğini yok eder.

Mutasyon Oranı: Kromozomlar birbirine benzemeye başladığında hala çözüm noktalarının uzağında bulunuyorsa mutasyon işlemi GA'nın sıkıştığı yerden kurtulmak için tek yoludur. Ancak yüksek bir değer vermek GA'yı kararlı bir noktaya ulaşmaktan alıkoyacaktır.

Kaç Noktalı Çaprazlama Yapılacağı: Normal olarak çaprazlama tek noktada gerçekleştirilmekle beraber yapılan araştırmalar bazı problemlerde çok noktalı çaprazlamanın çok yararlı olduğunu göstermiştir.

Çaprazlamanın sonucu elde edilen bireylerin nasıl değerlendirileceği: Elde edilen iki bireyin birden kullanılıp kullanılamayacağı bazen önemli olmaktadır.

Nesillerin birbirinden ayrık olup olmadığı: Normal olarak her nesil tümüyle bir önceki nesle bağlı olarak yaratılır. Bazı durumlarda yeni nesli eski nesille birlikte yeni neslin o ana kadar elde edilen bireyleri ile yaratmak yararlı olabilir.

Parametre kodlanmasının nasıl yapıldığı: Kodlananın nasıl yapıldığı en önemli noktalardan biridir. Örnek vermek gerekirse kimi zaman bir parametrenin doğrusal yada logaritmik kodlanması GA'nın performansında önemli bir farka yol açabilir.

Kodlama gösteriminin nasıl yapıldığı: Bu da nasıl olduğu yeterince açık olmamakla beraber GA'nın performansını etkileyen bir noktadır. İkilik düzen, kayan nokta aritmetiği ya da gray kodu ile gösterim en yaygın yöntemlerdir.

Başarı değerlendirmesinin nasıl yapıldığı: akıllıca yazılmamış bir değerlendirme işlevi çalışma zamanını uzatabileceği gibi çözüme hiçbir zaman ulaşmamasına neden olabilir.

6.5 Genetik Algoritmalar Sözlüğü

Algoritma Kavramı: Verilen bir problemin çözümü için izlediğimiz sistemli yönteme algoritma denir. Verilebilecek en basit örnek Eukliedes (öklitin iki sayının ortak bölenini bulma) algoritmasıdır.

Biyolojik Temeller: Biyolojinin kalıtımla ilgilenen dalına genetik denir. Kalıtım , bazı genetik özelliklerin bir kuşaktan diğer kuşağa aktarılma sürecidir. Modern genetik bilimin temelleri Gregor Mendel (1822-1884) tarafından gerçekleştirilen deneylere dayanmaktadır. Bu deneyler sonucunda, bir sonraki kuşağa aktarılacak özelliklerin bir takım kurallara uyduğu ortaya çıkmıştır. Genetik biliminde kullanılan ve tezde sözü edilen bazı kavram ve terimler aşağıda açıklanmıştır.

Allel (Allele): Bir özelliği temsil eden bir genin alabileceği değişik değerlerdir.
Çaprazlama (Cross-over): İki kromozomun bir araya gelerek genetik bilgi değişimi yapması Örnek: 100011101 ve 010110001 kromozomları üzerinde 4. locusdan başlayarak tek noktalı çaprazlama yapıldığında 100110001 ve 010011101 kromozomları elde edilir.
Evrimsel Algoritma (Evolutionary Algoritm EA): Genetik algoritmalarında içine alan bir algoritmik yöntem.

Evrilmek (Evolve): Bir evrim sürecinden geçmek.

Evrim (Evolution): Genetik bilgi taşıyan bir topluluk üzerinde genetik işlemlerin uygulanması süreci.

Gen (Gene): Kendi başına anlamlı genetik bilgi taşıyan en küçük genetik yapı Örnek: 101 bit dizisi bir noktanın x- koordinatının ikilik düzende kodlandığı bir gen olabilir.
Locus : Kromozom üzerindeki her bitin yerine verilen isimdir.

Genetik Programlama (Genetic Programming GP): Genlerinde program parçacıklarının kodlandığı kromozomlar üzerinde çalışan bir genetik algoritma yolu ile istenilen işi yapan bir programın oluşturulması.

Tersinme (Inversion) : Bir kromozomu oluşturan genlerden ardışık bir grubun kendi içerisinde birbirleri ile yer değiştirerek ters dizilmeleri.

Örnek: 011110101 kromozomu (her genin bir konum olduğu varsayımı ile) 5. Ve 8. Gen konumları arasında tersindiğinde ortaya 011101011 kromozomu çıkıyor.
Eşleme (Matching) : İki kromozom çaprazlanma amacı ile seçilmesi.
Mutasyon (Mutation) : Bir kromozomum taşıdığı genetik bilgide bir nedene bağlı olmaksızın (rasgele) değişme olması.

> Örnek : 100110000 kromozomunun 3. Konumunun değişmeye uğraması sonuncunda 101110000 kromozomu oluşur.

Çoğalma,Kopyalama (reproduction): Bir kromozomun kendisi ile aynı genetik bilgiyi taşıyan bir kopyasının oluşturulması

> Örnek : 100111011 kromozomu çoğaltıldığında 100111011 ve 100111011 kromozomları elde edilir.

Seçme (Selection): Bir kromozom havuzundaki kromozomlardan hangilerini yeni yaratılacak havuza aktarılacağının (kromozomların başarı değerlerine bağlı olarak belirlenmesi)

En iyileme (Optimizasyon): Matematiksel bir terim olarak optimizasyon, en iyinin tanımlanması ve çözülmesi anlamına gelir. Optimizasyon teorisi, optimanın, yani en iyinin, sayısal olarak elde edilmesi için kullanılan tüm yöntemleri içerir. Matematiksel olarak ifade etmek gerekirse, optimizasyon bilinen bir fonksiyonun (amacın) belirli kısıtlar altında ya da kısıtlar olmaksızın optimumunun bulunması demektir.

Optimizasyon problemlerinin pek çok çeşidi vardır; bunlar arasında tam sayılı programlama, doğrusal programlama, dinamik programlama, birleşim (combinatorial) optimizasyon sayılabilir. Optimizasyon, yöneticilerin karar verme aşamasında kullanacakları bir araç olmasının yanı sıra, diğer pek çok alanda da kullanılmaktadır. Kimya, bilgisayar bilimleri, elektronik, üretim planlama, vb. gibi. 20. yüzyıl ile birlikte bilgisayarların süratle yayılması ve ucuzlaması sayesinde pek çok yeni araştırmacı, birleşik optimizasyon problemleri başta olmak üzere bu konuda çalışmaya başlamıştır. Bilgisayarlar birçok konuda olduğu gibi matematik konusunda da hayatımızı kolaylaştırdılar. Fakat yine de onların hızının bile yeterli olamayacağı ve pratik hayatta çözüm bekleyen problemler çok sayıdadır.

6.6 Genetik algoritmaların Kuramsal Temelleri (Şema Teorisi)

Genetik algoritmaların çalışması açıklayan en önemli kuramlardan birisi de John Holland tarafından ortaya atılan şema (*schema*) kuramıdır. Bu kurama göre genetik algoritmalar, iyi yapı taşlarını ortaya çıkarır, çoğaltır ve birleştirir. Bu işlemler oldukça paralel bir şekilde yapılabilir. Ana fikir iyi çözümlerin iyi yapıtaşlarından oluştuğu şeklindedir (Tang K.S., Man K.F., Kwong S. and He Q., 1996).

Bu kuram, temel bir genetik algoritmanın çalışmasını açıklamaya çalışır. Bu genetik algoritma tek noktalı 'crossover' (çaprazlama) işlemini, ikilik dizi kodlamasını, nokta mutasyonu (bitin değerinin değiştirilmesi şeklinde) ve rulet tekerleği seçimini içerir.

Rulet tekerleği seçimine, her toplum üyesine negatif olmayan bir başarım puanı verilerek başlanır. Bu puanlar toplanarak her üyenin başarımı toplama bölünür. Elde edilen değerler bir diziye yerleştirilir. [0,1] aralığında bir rasgele sayı seçilir. Bu dizi üzerinde toplam değer rasgele sayıyı geçene kadar her üyenin değeri toplanarak ilerlenir. O an bulunulan değer seçilmiş olur. Bu olay bir rulet tekerleğinin çevrilmesinin benzeşimidir. Farklı olarak parçalar üyelerin başarımlarına göre ayrılmıştır. Bu yöntem, büyük toplumlarda, üyelerin ortalama olarak başarımlarının ortalamaya oranı kadar yeni üye meydana getirmesini sağlar.

Genetik algoritmalarda oluşan başarılı bireyler incelenirse, bu bireyler arasındaki benzerlikler bulunabilir. Bu benzerliklerden yola çıkarak şemalar oluşturulabilir. İkilik dizi kodlaması için aşağıdaki yöntem önerilebilir.

0,1 ve # ('#' o konumda 0 veya 1 olmasının önemsiz olduğunu gösterir) Örnek olarak ikinci ve dördüncü bitleri 1, altıncı biti 0 olan çözümlerin başarılı olduğu bir toplumda şu şema oluşturulabilir:

#1#1#0 (6.1)

Bu şemaya uygun aşağıdaki ikilik diziler yazılabilir:

010100, 010110, 011100, 011110, 110100, 110110, 111100, 111110. (6.2)

Görüldüğü gibi şemaların katılması ikilik dizilerle gösterilen arama aralığını büyütmektedir. Arama aralığının büyümesinin sonucun bulunmasını zorlaştırması beklenir ancak durum böyle değildir. Seçilim ve yeniden kopyalama ile iyi özellikler daha çok bir araya gelerek daha iyi değerlere sahip şemalara uygun çözümler elde edilir.

Genetik algoritma kendi içinde sanal olarak şemaları oluşturur. Toplumun bireyleri incelenerek bu şemalar ortaya çıkarılabilir. Genetik algoritmalar şemaları oluşturmak için toplum üyelerinin kodları dışında bir bilgi tutmaz. Genetik algoritmaların bu özelliğine içsel paralellik (implicit parallelism) denir. Her nesilde, iyiyi belirleyen şemalardaki belirsiz yada önemsiz elemanlar azalır. Böylece genetik algoritmalar sonuca doğru belli kalıplar içinde ilerler. Genetik algoritmaların problem çözmedeki başarısı, pek çok ampirik deneyle gösterilmiştir. Yöntemin çalıştığını ispatlamak için birçok çalışma yapılmıştır. Bu çalışmaların büyük bir bölümü özel durumları içerir. Genetik algoritma uygulamalarının çeşitliliği ve probleme özel kullanılan yöntemler bu incelemeyi daha da zorlaştırmaktadır.

Holland, yapı taşlarını göstermek amacıyla şema gösterimini oluşturmuştur. Şemaların gösteriminde ikilik dizi gösterimde kullanılan alfabe *{0,1}* bir karakter genişletilir *{0,1, #}*. Diyez *{#}* karakteri şema içinde o noktada bulunan değerin önemsiz durum (don't care condition) olduğunu belirtir. Örneğin *H=1#0* şemasına *{100}*ve *{110}* dizileri uyar.

l uzunlukta diziler kullanılırsa, herhangi bir dizi 2^l değişik şemaya uyar (şemaların her pozisyonunda dizinin o pozisyonundaki değer veya diyez sembolü bulunabilir). *n* elemanlı bir toplumda en az 2^l en çok $n*2^l$ tane değişik şemanın örneği bulunur. Bu durumda genetik algoritma çalışırken dizileri hesaplamanın yanında içsel olarak çok daha fazla sayıda şemanın başarım değerlerini de yaklaşık olarak hesaplamaktadır. Örneğin *n* elemanlı bir toplum rasgele oluşturulduğunda *n/2* eleman *1###...####*, *n/2* eleman da *0###...#####* şemasına uyar. İlk hesaplama sonucunda bu iki şemaya uyan elemanların kendi aralarındaki ortalama başarımları o şemaların başarımıdır. Genetik algoritmalar çalışmaları sırasında şemaları ayrıca saklamadıkları gibi bu ortalama başarım değerlerini de saklamazlar.

Bu modelde bir şemanın bir sonraki zaman diliminde kaç örneğinin bulunacağı yaklaşık olarak hesaplanabilir. *H*, toplumda en az bir örneği bulunan bir şema olsun. *m(H,t)* o şemanın *t* anında toplumda bulunan örneklerinin sayısı olsun. *û(H,t)* şemanın *t* anında gözlenen başarım değeri (şemanın örneklerinin başarım değerlerinin ortalaması) olsun. Burada amaç *H* şemasının *t+1* anındaki örneklerinin yaklaşık sayısını *E(m(H,t+1))* bulmaktır. Seçme yöntemi bir elemanın yeni toplumda oluşturacağı yeni eleman sayısını $E(x)=f(x)/f_{ort}(t)$ şeklinde olmasını sağlar. Bu eşitlikte *f(x)* elemanın başarım değeri, $f_{ort}(t)$ ise *t* anındaki toplumun başarım ortalamasıdır. *x*, *t* anında toplumun bir üyesi ve aynı zamanda *H*'nin bir örneği olsun . "$x \in H$" de "*x*, *H*'nin bir örneğidir" anlamına gelsin.

. Bu durumda

$$\hat{u}(H,t) = \left(\sum_{x \in H} f(x) \right) / m(H,t) \qquad (6.3)$$

tanımından

$$E(m(H,t+1)) = \sum_{x \in H} f(x) / f_{ort}(t) = (\hat{u}(H,t) / f_{ort}(t)) m(H,t)$$

Şekil 6.2 H şemasının $t+1$ anındaki örnek sayısı.

bağıntısı yazılabilir. Genetik algoritma $\hat{u}(H,t)$ değerini hesaplamasa da şemaların örnek sayılarının değişimi bu değere bağlıdır. Şekil 5.2'deki denklem 'crossover' ve mutasyonun etkilerini içermemektedir. 'crossover' ve mutasyon hem H şemasının örneklerini oluşturabilir, hem de yok edebilir. Aşağıda 'crossover' ve mutasyonun yok edici etkileri incelenmiştir.

$S_c(H)$, H şemasının tek noktalı 'crossover' işleminden bozulmadan çıkabilme olasılığı, p_c dizi üzerinde 'crossover' uygulanma olasılığı olsun. H şemasının iki örneği arasında olacak bir 'crossover' şemayı bozmayacaktır. Bu nedenle aşağıda $S_c(H)$ için bir alt sınır hesaplanacaktır. Bu işlem için $d(H)$ H şemasının uzunluğu, l ise kullanılan dizilerin uzunluğu olarak tanımlanmıştır.

$$S_c(H) \geq 1 - p_c\left(\frac{d(H)}{l-1}\right) \qquad (6.4)$$

Bu denklemde $d(H)$ uzunluğunun herhangi bir noktasında 'crossover' gerçekleşirse H şeması bozulur. H şemasını bozan 'crossover'ların oranı ile 'crossover' olma olasılığını çarparak bozucu 'crossover'ların olasılığı bulunabilir. Kısa şemaların 'crossover'da bozulmama şansı daha yüksektir.

Her bitin mutasyonun olasılığı p_m olsun. $o(H)$ H şemasındaki belirli (# olmayan) bitlerin sayısı olsun. S_m mutasyondan bozulmadan geçme şansı ise :

$$S_m(H) = (1-p_m)^{o(H)} \qquad (6.5)$$

olur. Bu durumda az biti belirli olan şemalar mutasyondan daha az etkilenir.

Hepsini birleştirirsek:

$$E(m(H,t+1)) \geq \frac{\hat{u}(H,t)}{fort(t)} m(H,t)\left(1 - p_c \frac{d(H)}{l-1}\right)\left[(1-p_m)^{o(H)}\right] \quad (6.6)$$

eşitsizliğini elde ederiz. Bu Şema Teorisi olarak adlandırılır (Holland 1975). Bu eşitsizlik mutasyon ve 'crossover'ın sadece yok edici etkisini içerir ve bir alt sınırdır. Pek çok araştırıcı 'crossover'ın genetik algoritmanın ilerlemesindeki temel etken olduğunu düşünmektedir. Belli değerler arasında kaldığı sürece 'crossover'ın yapıcı etkisinin yıkıcı etkisinden çok daha baskın olduğu deneylerle gösterilmiştir. Mutasyon ise çok büyük olasılıklarla kullanılmadığı sürece yararlıdır ve çeşitliliğin korunmasını sağlar (Man K.F. , Tang K.S. , Kwong S. and Halay W.A. (1997)

6.7 Genetik Algoritmalar Kullanılarak Bir Problem Çözümünün Aşamaları

GA'lar yukarda değinildiği gibi genel olarak üç temel işlev içerir: Üretme (reproduction), çaprazlama ve mutasyon adı verilen bir optimizasyon (uygunlaştırma) problemi, GA ilgili parametreleri sonlu bit dizileri şeklinde kodlar ve sonra bir rastgele yöntemler ile üç işlev kullanılarak iteratif olarak çalıştırılır (Varsek A., Urbancic T. and filipic B., 1993). Programın amacına göre değişiklikler olmasına rağmen genetik algoritmalar temelde aşağıdaki yöntemi izler:

1. Bir çözüm grubu oluşturulur. Bu grubun oluşturulması tamamen rasgele olabileceği gibi probleme göre özelleşmiş de olabilir. Tamamen rasgele bir çözüm grubu genetik algoritmaya tüm problem uzayını arama şansı verir; probleme göre özelleşme ise işlemi önemli oranda hızlandırabilir. Çözüm grubunun büyüklüğü de önemli bir faktördür. Büyük çözüm grupları çok işlem gerektirir. Küçük çözüm grupları, yerel maksimum değerlerine takılabilir. Başarılı bir çözümün küçük bir grupta baskın hale geçmesi çok daha kolaydır. Bu nedenle küçük çözüm grupları çeşitliliklerini çok çabuk kaybederler.

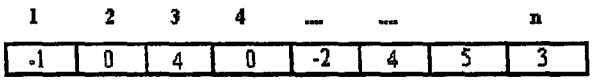
2. Eldeki çözüm grubunun içinden başarılı çözümler seçilir. Seçme işleminin temel mantığı, doğadaki gibi başarılı olan çözümlere üstel çoğalma imkanı vermekle açıklanabilir. Bilgisayar ortamında, bellek ve işlem zamanı sınırlı olduğu için, başarılı çözümleri çoğaltmak diğer bireyleri azaltmak anlamına gelir. Bu yolla, çözüm grubunun büyüklüğü sabit tutulabilir. Kullanılan seçme yöntemlerinin amacı, başarılı bireylere üstel çoğalma imkanı vermekle çeşitliliği korumayı en verimli şekilde dengelemektir. Sıklıkla kullanılan yöntemler arasında rulet tekerleği seçimi, turnuva seçimi, sigma ölçeklendirmesi, Boltzman seçimi, derece

seçimi, kararlı durum (steady state) seçimi sayılabilir. Yardımcı olarak kullanılan elitist seçim ise en başarılı bireylerin bir sonraki çözüm grubuna değiştirilmeden aktarılmasıdır. Bazı uygulamalarda başarılı olduğu görülmüştür. Başarılı bireyler kullanılarak yeni çözüm grubu oluşturulur. Yeni çözüm grubunu oluşturmak için kullanılan işlemciler 'crossover' ve mutasyondur. Bu işlemciler de yapılan işleme göre değişse de genel yöntemleri pek farklı değildir.

3. 'crossover' iki çözümün yapıtaşları kullanılarak yeni bir çözüm oluşturulması esasına dayanır. Bu işlem doğada görülen 'crossing over' olayının analoğudur. 'crossover' işlemi genel olarak ikili dizilerin parçalarının değiş tokuşu şeklinde gerçekleştirilir. Farklı uygulamalarda, farklı kodlama yöntemleri kullanıldığı için farklı 'crossover' yöntemleri kullanılır. Mutasyon bir çözümün, çoğunlukla rasgele olarak, değiştirilmesidir. Bu işlem çok değişik şekillerde kullanılır. Problemin yapısı bu aşamada çok önemlidir. Örneğin sıralama problemlerinde sıralamayı değiştirmek, ikilik dizi gösteriminde bitleri değiştirmek, çözüm ağaçlarında parçaları değiştirmek işlem olarak tanımlanır.

4. 2. ve 3. işlemler belirlenen bir şart sağlanana kadar tekrarlanır. Bu şart tekrar sayısının belirlenen bir sayıya ulaşması, belirlenen bir başarım değerine ulaşılması, genetik algoritmanın daha başarılı çözümler oluşturamaması vb. olabilir. Yukarıda belirtilen işlevleri gerçekleştirmek için çeşitli yöntemler bulunmaktadır. Bunlar;

Parametre Kodlanması : GA ile araştırılması ve optimum çözüm bulunması istenen problemin parametreleri belirlenir. Yaygın kullanılan yol olarak parametreler ikili bit dizileri içersine transfer edilir. Birkaç parametre bir uzun dizi yapısı içine kodlanabilir.



Şekil 6.3 Parametrelerin dizilere yerleştirilmesi

Genetik algoritmanın başarımını etkileyebilecek bir diğer etken de kodlama yöntemidir. Genetik algoritmalar, problemden bağımsız olarak sadece problemin koduyla uğraşırlar. Bu kodlama problemi ne kadar iyi temsil edebiliyorsa program o kadar başarılı olur. Sayıların kodlamasında temelde iki yöntem kullanılır.

İkilik dizilere haritalama (binary string mapping) yöntemi, bir gerçel sayı aralığının belli sayıda eşit parçaya bölünüp her parçaya bir numara verilmesi şeklinde tanımlanabilir. Bu

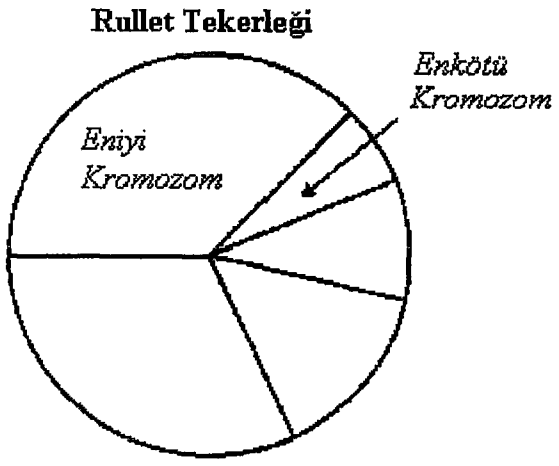
yöntemin en önemli dezavantajı arama aralığını kısıtlamasıdır. Tanımsızlık içermemesi sayesinde genetik algoritmanın çalışması sırasında bir düzeltme ve kontrole gerek kalmamıştır.

İkinci yöntem ise gerçel sayıları gerçel kısım ve üs olarak tanımlanabilen iki parçadan oluşan bir ikilik dizi ile göstermektir. Bu gösterimle çok geniş gerçel sayı aralıkları gösterilebilmektedir. Bu yöntemin dezavantajı ise iki parçası yani gerçel kısım ve üs arasındaki değer farkıdır. Büyük değerler için üs kısmının değişmesi sayıyı çok değiştirir. Bu bölgelerde bulunan alternatif değerlerin ortadan kalkma hızının yani bir değerin baskın gelme hızının yüksek olmasını getirir. Bu kodlamanın probleme özel bir kodlama olması da tercih edilmemesinin bir nedenidir. Kodlama ayrıca pek çok tanımsız durum içerir. Üssün büyük olduğu durumlarda çarpma işlemi kolaylıkla taşmalara neden olabilmektedir. Büyük ve küçük sayıların 'crossover' işleminden geçirilmesi sonucunda çok büyük sayılar elde edilebilir. Bu sayıların hesaplanması da taşmalara neden olabilir. Bu sorunları gidermek için genetik algoritmanın çalışması sırasında oluşturulan her çözümün geçerliliğinin kontrol edilmesi ve geçersiz olan çözümlerin düzeltilmesi gerekir. Bu tip bir işlem genetik algoritmanın karmaşıklaşmasına neden olur.

Başlangıç Nesil : N dizilik başlangıç populasyonu bir havuzda her zaman random olarak üretilir. Her bir dizi (kromozom) *m* bit (gen) uzunluğundadır. Populasyon (havuz) büyüklüğü N, bir uyuşma faktörüdür. Büyük N seçilmesi, ilk nesilde çözümü içerme mümkünlüğünü artırır, fakat GA'ların çalışma hızını azaltır ve çözüme gitmeyi uzatabilir. *m* dizi uzunluğu çözünürlüğü belirler. Genetik algoritmalar bir çözüm uzayında global bir araştırmayı yerine getirir, kodlu bit-dizi uzunlukları mümkün olduğu kadar kısa tutulmalıdır, çünkü arama uzayının büyüklüğü, dizi büyüklüğü ile exponansiyel olarak artar.

Uyumluluk Değerlendirme : Geçerli nesilde, dizilerin her biri ona karşılık gelen gerçek parametrelere çevrilir. Bu parametreler bir yargı makinesine gönderilir. Bu yargı makinesi, çözümün kalitesinin ölçümünü oluşturur. Çözüm kalitesi bazı nesnel fonksiyonlar ile değerlendirilir ve uyumluluk değerleri olarak atanır.

Üreme (Reproduction) : Üreme, büyük uyumluluk değerli dizilerin yeni nesilde sayısının artırılması ile daha büyük olasılıkta üretilme işlemidir. Çok yaygın metod olarak, ağırlıklı rulet seçimi kullanılır (Şekil 6.4).



Şekil 6.4 Kromozom seçiminde kullanılan Rullet Tekerleği

Seçme (Selection): Seçme işlemi için turnuva seçimi benimsenebilir. Bu yöntem sayesinde başarım ölçeklendirmesi gibi işlemlerin yapılmasına gerek kalmaz. Başarım ölçeklendirmesi fonksiyon en iyilemesinde çeşitli problemlere neden olur. Örneğin fonksiyonun değeri 10^8 mertebesine ulaştığı zaman fonksiyonun değerindeki 10^5'lik değişmeler çözümlerin başarımını önemli ölçüde etkilemez. Bu problemden kurtulmak için fonksiyon değerinin algoritmasını içeren bir başarım fonksiyonu tanımlanabilir. Logaritma işlemi negatif sayılar için tanımsız olduğundan tüm başarım değerlerine bir katsayı eklenerek bu değerlerin pozitif yapılması gerekir. Pozitif yapma işlemi sırasında bazı çözümlerin anlamlı rakamları ortadan kalkar. Örneğin bir toplumda çoğunluğu 10^{-5} mertebesinde sayılar oluştururken en küçük değer -100 olursa bu değerin pozitif yapılması için toplumun diğer elemanlarına eklenecek değerler bu sayıların 7 anlamlı basamağının kaybolmasına neden olur.

Sonuçların net olması için genetik algoritmanın çalışması sırasında seçimin baskısının (başarılı çözümlerin çoğaltılma ve başarısızlarının elenme hızı) azalmaması gerekir. Seçim baskısı azalırsa, daha başarılı çözümler çoğaltılamadığı için başarılı yapıtaşlarının çoğaltılması da mümkün olmaz. İyi yapıtaşlarının az olması iyi çözümlerin oluşmasını engeller ve dolayısıyla çözüme ulaşma hızını ve şansını azaltır. Bu genetik algoritmanın başarımını önemli ölçüde düşürür.

Turnuva seçimi yukarıda sözü geçen yöntemlere göre oldukça basittir. Toplumun rasgele iki elemanı seçilir. Bu iki eleman içinden başarılı olanı bir olasılık şartının gerçekleşmesi durumunda seçilir. Bu işleme seçilmiş elemanın rasgele seçilmiş bir başka elemanla birlikte

yukarıda belirtilen işlemden geçirilmesi ile devam edilir. Bu işlemin tekrar sayısı *t_rounds* kadardır.

Çaprazlama (Crossoverring) : İki dizi arasında sistematik bilgi değişiminin olasılıksal kararlar kullanılarak gerçekleştirildiği işlemdir. Çaprazlama işleminde, yeni olarak tekrar üretilen diziler araştırma havuzundan seçilir ve kendi aralarında rastgele olarak seçilen bölümleri karşılıklı değiştirilir. Bu işlem tercih edilen iyi diziler arasında iyi kaliteleri birleştirir ve optimal çözüme yakın diziler oluşmasını sağlar. Düzenli çaprazlamada, çaprazlama noktaları arama işlemi yapılmaz, basitçe her bir dizinini orta noktasından yerdeğiştirme yapılır. Çaprazlama birden fazla noktada da yapılabilir. 'crossover' yöntemi çok noktalı 'crossover' olarak belirlenebilir. Maksimum 'crossover' sayısı biçim dosyası ile belirlenmiştir. 'crossover' olması için bir olasılık koşulunun sağlanması gereklidir. Bu koşul sağlandığı sürece 'crossover' gerçekleştirilir. 'crossover' işlemi sadece aynı değişkenlerin kodları arasında gerçekleştirilmektedir. Bu tip bir 'crossover' işleminin tanımlanmasında farklı değişkenlerin kodlarını içeren gen havuzlarının karışmasını önlemek amaçlanmıştır.

1. dizi	100110|011011	çaprazlama sonrası :	1. dizi	100110011001
2. dizi	110001|011001		2. dizi	110001011011

↑
çaprazlama noktası

Mutasyon (Mutation): Mutasyon, genellikle rastgele olarak seçilen bir bitin (gen) durumunun değiştirilerek optimal noktaya ulaşmak için GA'nın şans işlemidir. Mutasyon işlemi, düzgün olarak önceki üreme ve çaprazlama işlemlerinin üretmeyeceği dizileri hızlıca üretebilir. Fakat mutasyon, ani olarak mevcut uygun üreme fırsatını bozabilir, bu yüzden bu işlem küçük olasılıkta yapılır ve üreme, çaprazlamanın tamamlayıcısıdır.

mutasyon bit dizileri üzerindeki bitlerin değiştirilmesiyle sağlanmaktadır. İkilik dizi gösterimi tanımsızlık içermediği için bu yöntem oldukça uygundur. Bu gösterimle her sayı arama aralığının bir üyesi olduğu için mutasyon çok verimli çalışabilmektedir. Bu yöntemle mutasyon çok küçük (arama aralığının dört milyarda biri) ve çok büyük (arama aralığının yarısı) değişikliklere neden olabilmektedir.

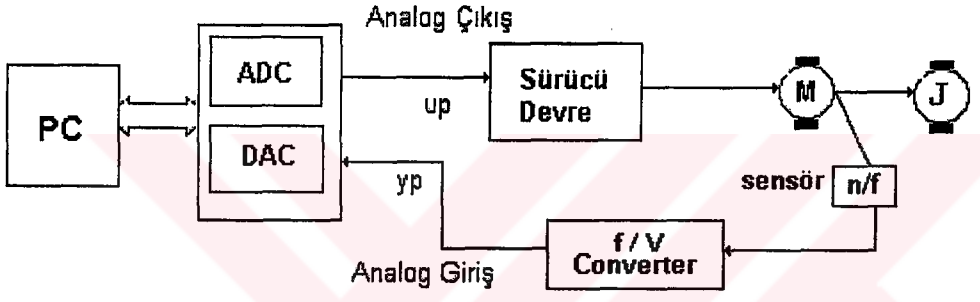
dizi 100110011001 mutasyondan sonra : 100100011001

↑
mutasyon biti

7. ÇALIŞMADA KULLANMIŞ OLUNAN BİLGİSAYAR TEMELLİ DC MOTOR HIZ KONTROLÜNÜN GERÇEKLEŞTİRİLMESİ

7.1 Giriş

Bu çalışmada Genetik algoritmalar yardımıyla PI-tip Bulanık kontrolör kullanılarak PC üzerinden DC motor devir kontrolü yapılmaktadır. Motor devir bilgisi sensör yardımıyla frekans bilgisine ve sonradan gerilime dönüştürülerek bir arabirim vasıtasıyla PC'ye aktarılmaktadır. Burada C Programlama dili yazılan kontrol algoritmasında kullanılarak sistemi verilen referansa değerde tutmak için gerekli kontrol sinyali bulunarak tekrar arabirim üzerinden motora uygulanmaktadır (Şekil 7.1).



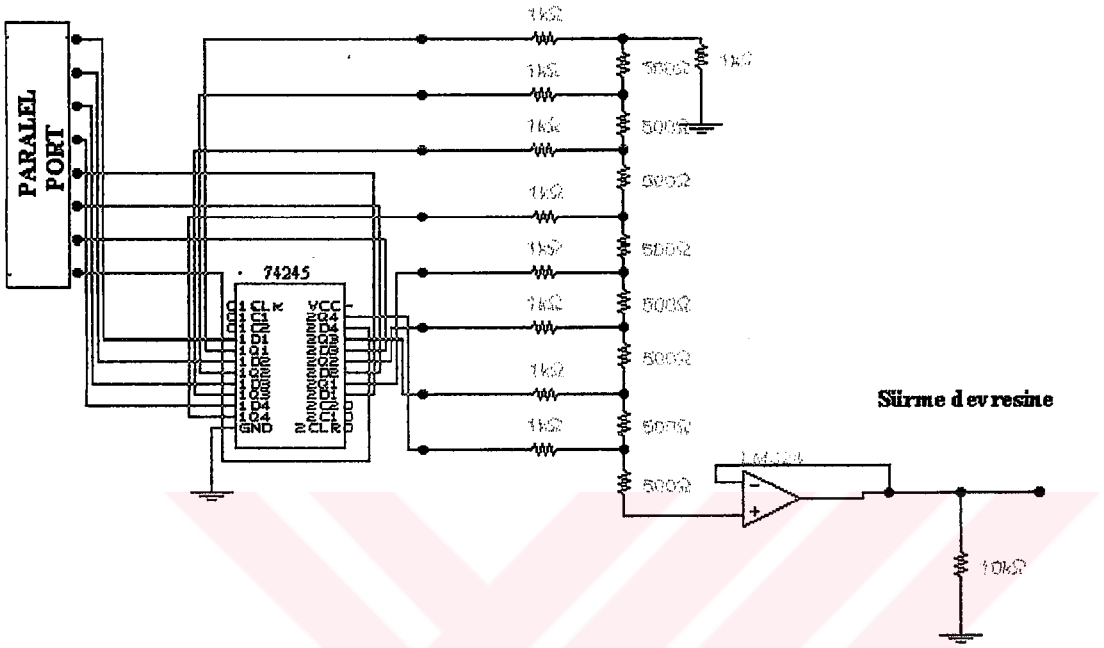
Şekil 7.1 PC Temelli DC Motor Devir Kontrolü Şematik Gösterimi

Bu çalışmada doğru akım motor hız kontrolü için insan tecrübesini kullanmaksızın Genetik Algoritmalar kullanılarak PI-tip Bulanık kontrolör tasarlanması amaçlanmıştır. Bu amaç üç aşamada gerçekleştirilmiştir: ilk olarak iteratif olarak çalışan genetik algoritma temelli bir öğrenme algoritması geliştirilerek bulanık kurallar bu algoritma ile çıkarılmıştır. İkinci olarak Genetik bulanık öğrenme algoritmasının her nesilde elde edilen tüm kural dizileri motor matematiksel modeli üzerinde simule edilerek en uyumlu kural taban araştırması yapılmaktadır. İstenen performans değerli kural taban bulununcaya kadar veya maksimum nesil sayısına ulaşıncaya kadar algoritma çalıştırılmaktadır. Üçüncü aşamada istene uyumluluk değerine sahip kural tabanı elde edildikten sonra bu kural tabanı ile PI-tip Bulanık Kontrolcu yapısına yerleştirilerek direk olarak motora uygulanmaktadır.

7.2 Bilgisayar Temelli Sistemin Özellikleri

Bu tezde istenen amaç için kullanılmak üzere, PC parallel portu üzerinde 8-bit olarak bilgi transferi olanağı sağlayan ve üzerinde motor sürme devresi ve hız geri bilgisi sağlayabilen bir dijital I/O (giriş-çıkış) arabirim oluşturulmuştur. Yani motor sürme devresi ile bilgisayar

arasında paralel port üzerinden 8-bitlik bilgi alışverişi sağlanmıştır. Kontroller PC üzerinde oluşturularak on-line olarak motor kontrolü gerçekleştirilmiştir. Motor hız bilgisi arabirim üzerinden direkt bilgisayara gönderilerek grafik üzerinde izlenmesi sağlanmıştır.



Şekil 7.2 PC paralel portu üzerinden DAC ile kontrolör sinyal çıkış devresi

Turbo C dilinde yazılmış bir program ile bilgisayarın paralel portundaki (0x378) 8-bitlik data pinleri, motora kontrolör işareti olan hız bilgisini sürmek için kullanılmaktadır. Bu pinlerin numaraları en önemli bitten (MSB) en önemsiz bite (LSB) doğru 2, 3, 4, 5, 6, 7, 8, 9 numaralı pinlerdir. Her data pininin veri yoluna bağlanan 1 Kohm'lık dirençler ile o yola 0 bilgisi geldiğinde istenmeyen parazitler toprağa aktarılmıştır. Bilgisayar üzerinde çalıştırılan Turbo C ile yazılmış programdan gelen kontrolör çıkışı olan dijital bilgiyi bir DAC üzerinden analog değere çevrilmekte ve sürme devresi üzerinden motora uygulanmaktadır. Bu çalışmada kullanılan DAC piyasada flaş DAC olarak bilinen R-2R prensibine göre çalışan basit ve maliyeti düşük olan bir çeviricidir. DAC çıkışına bir gerilim izleyici (voltage follower) bağlanmıştır. DAC çıkışındaki 10 K'lık direnç istenmeyen parazitleri toprağa aktarmak için kullanılmıştır. DAC bloğunun çıkış akımı aşağıdaki şekilde hesaplanır.

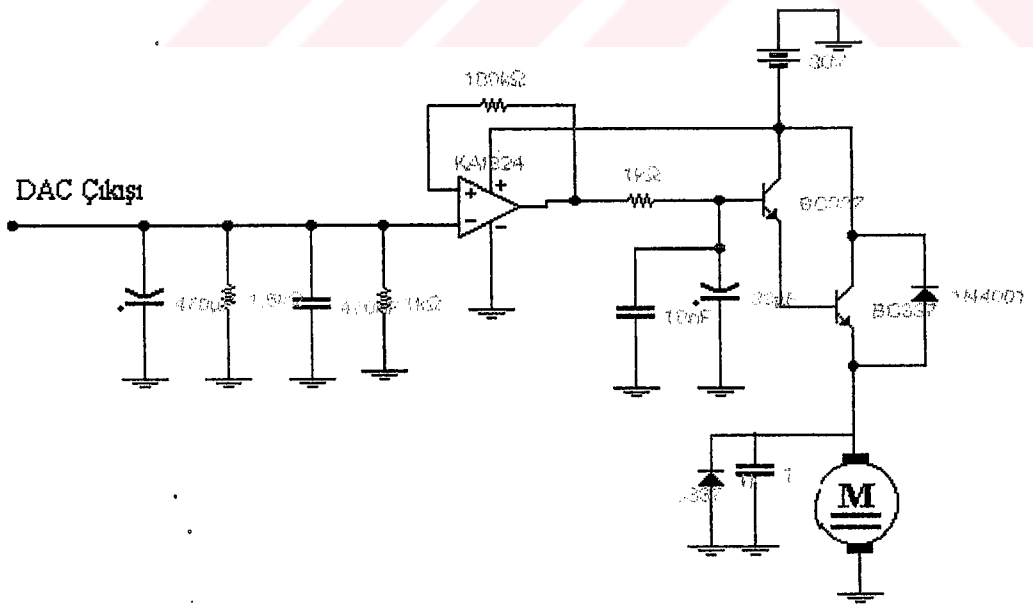
$$Io = \frac{Vref}{R_1}\left(\frac{A_1}{2}+\frac{A_2}{4}+\frac{A_3}{8}+\frac{A_4}{16}+\frac{A_5}{32}+\frac{A_6}{64}+\frac{A_7}{128}+\frac{A_8}{256}+\right) \qquad (7.1)$$

$$Io = \frac{+Vref}{R_1}\frac{N}{256} \tag{7.2}$$

şeklinde yazılabilir.

7.2.1 DC Motor Sürme Devresi

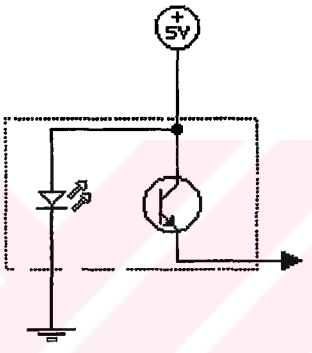
Bu çalışmada kontrol edilen sistem, daimi mıknatıslı doğru akım motorlarından oluşan iki motorun kaplin ile birleştirilmesinden oluşmuştur. Motor hız kontrolü, bilgisayar paralel portu üzerinden gelen kontrol sinyalini bir DAC ile analog sinyale dönüştürüldükten sonra 2N3055 güç transistörü ile darlington çiftini oluşturan BC337 transistörünün base ucu üzerinden bu transistörün kollektör-emittör üzerindeki gerilimi değiştirmek suretiyle sağlanır.
BC 337 nın emitter ucuna bağlı olan 2N3055 in base ucu bu gerilim değişikliği ile kendi kollektor-emmitör gerilimini ayarlar. Bu gerilimin değişimi direkt olarak DC motor armatür gerilimini değiştirdiğinden motor hızına doğrudan yansır. Motor sürme devresi Şekil 7.3 'de verilmektedir. Burda kullanılan sabit mıknatıslı DC motor 30 Voltluk besleme gerilimi ile çalıştırılmaktadır. Motorun maksimum devri 1500 dev/dak'dır. Devre yapısından dolayı motor üzerine düşen gerilim maksimum 26-27 volt olmaktadır. Yani motorun maksimum devri 1380-1400 dev/dak'dır.



Şekil 7.3 Tasarlanan sistem için DC Motor Süreme devresi

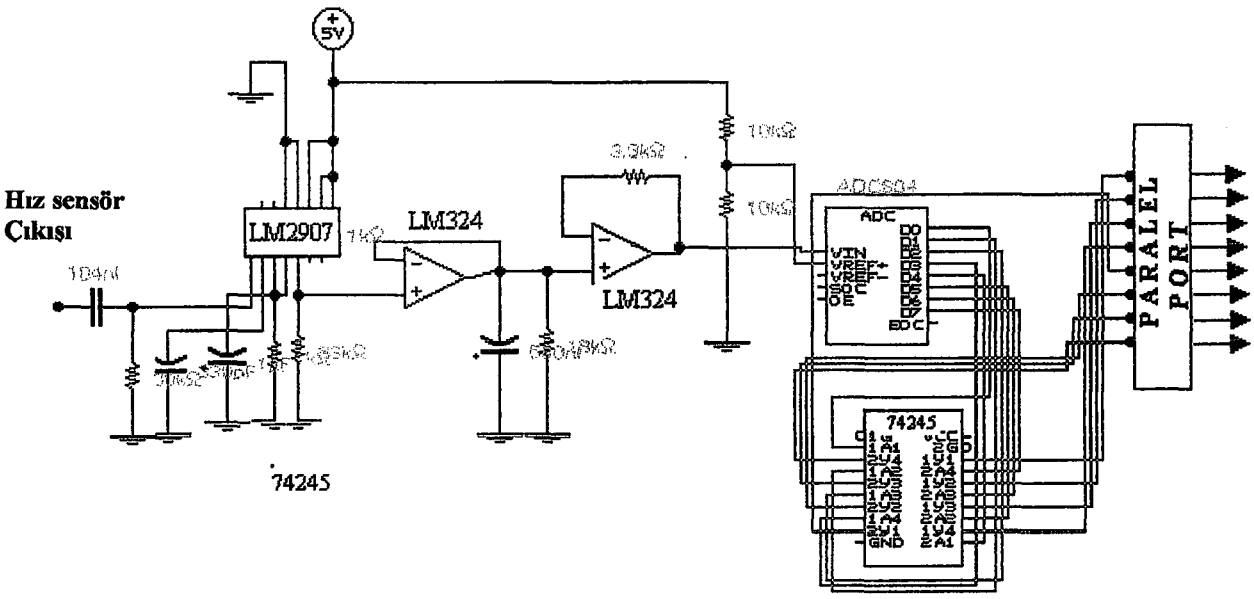
7.2.2 Hız Geri Bilgisinin Alınması

Kontrol edilen DC motor miline bağlı bulunan delikli disk ile bu diskin iki tarafında bulunan fototransistor ve fotodiod tarafından alınan motor hızı ile orantılı frekans bilgisi alınmaktadır. Delikli diskin dönerek foto transistör-diyot çiftinin arasındaki ışık iletimini kesip açmasıyla binary olarak kare dalga bir işaret üretilir. Bu işaretin frekansının büyüklüğü motor milinin hızıyla doğru orantılıdır. Bu çalışmada bu bilgi motorun hızını kontrol etmek için geribesleme sinyali olarak kullanılmaktadır. Şekil 7.4'te motor milinden kare dalga işaret üreten sensör gösterilmiştir.



Şekil 7.4 Motor milinden kare dalga işaret üreten hız sensörü

Motor miliyle doğru orantılı olan bu frekans bilgisi kullanılması için gerilime dönüştürülmektedir. Bunun için LM2907 frekans-gerilim dönüştürücü kullanılmıştır. Giriş işaretini frekansı yükseldikçe çıkış gerilimi de yükselmektedir. Entegre yüksek hassasiyette çalışmaktadır. LM2907 entegresinin çıkışına bir adet gerilim izleyici ve ile bir OPAMP kuvvetlendirici takılmıştır. Kuvvetlendirici ile motor devir bilgisi ile ölçülen gerilim arasında uyumluluk sağlanmıştır. Gerilime dönüştürülen hız bilgisi bir ADC üzerinden dijital işarete çevrilmekte ve yine paralel portun diğer pinleri kullanılarak bilgisayara aktarılmaktadır. Burada bilgisayara dijital veri girişini sağlamak için paralel portun STATUS ve CONTROL bitlerinden giriş olarak kullanılabilen 8 tanesi seçilmiştir. Hız geri bilgisinin bilgisayara aktarılması için tasarlanan devre Şekil 7.5'te gösterilmiştir.



Şekil 7.5 Hız sensörü bilgisini paralel port üzerinden bilgisayara aktaran devre şeması

7.2.3 Arabirim Devresi için Besleme Bloğu

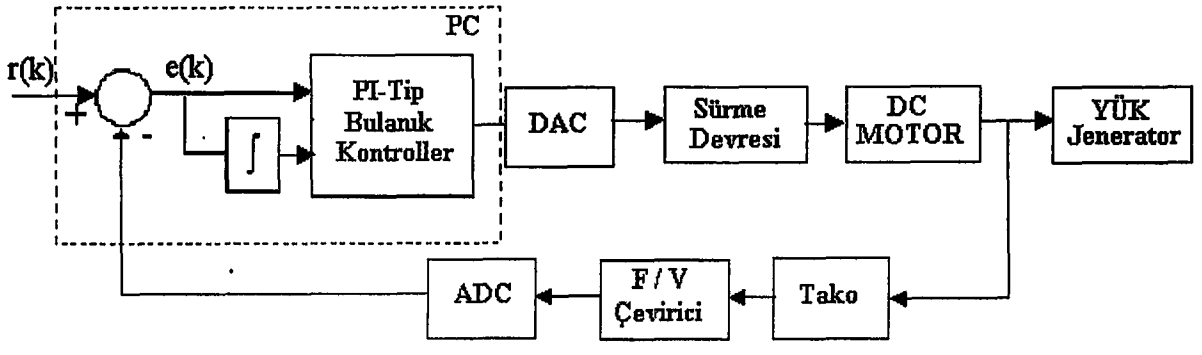
Motorun besleme gerilimi olan 30 volt dc gerilimi ve sürme-geri besleme devrelerinin ihtiyacı olan +5 volt dc gerilimi sağlamak için besleme kartı gerçekleştirilmiştir. Bu devrede +5 voltu üretmek için iki adet 7805 pozitif gerilim regülatörü ve 7905 negatif gerilim regülatörü kullanılmıştır. +30 volt için ise L200 entegresi ile sağlanmıştır. Burada 220/30 voltluk 60 watt ve 220/6 voltluk 3 watt olan iki adet trafo kullanılmıştır.

8. ÇALIŞMADA YAPILMIŞ OLAN DC MOTOR İÇİN GENETİK ALGORİTMALAR YARDIMIYLA PI-TİP BULANIK KONTROL GERÇEKLEŞTİRİLMESİ

8.1 Giriş

Günümüzde yaygın olarak endüstride kullanılmaya başlanılan bulanık kontrol yapılarında en önemli kısım, sisteme ait kural tabanın oluşturulmasıdır. Çünkü iyi sonuçlar alınabilecek kural taban, ancak sistemi tanıyan ve sistem hakkında tecrübeye sahip bir uzman tarafından tanımlanabilir. Bu da sistem için gerekli kontrol yapısını oluşturmada, ancak uzun zaman ve denemeler sonucu başarılabilir. Son yıllarda bu ve benzeri problemler nedeniyle kontrol sistem için gerekli kural tabanı otomatik olarak öğrenme veya örneklerden çıkaracak araştırıcı yöntemler kullanma yoluna gidilmektedir (Herrara, F., Lozano, M., Verdegay, J.L., 1995).

Bu çalışmada, bir DC motorun PI-tip Bulanık kontrolü için gerekli kural tabanın oluşturulmasında, araştırma yöntemleri arasında popüler olan genetik algoritmalar kullanılmıştır. Alınan sonuçlar bu yöntemin bir DC motorun bulanık kontrolör için uzman bilgisi olmaksızın kural tabanın tasarlanmasında iyi performans verdiğini ortaya çıkarmıştır. Bu amaç için bilgisayar üzerinde çalıştırılan ve Turbo C programlama dilinde yazılmış bulanık kontrol programıyla paralel port üzerinden kontrol edilen motor sistemi oluşturulmuştur. Motor kontrol sistemi; kontrol algoritmasının bulunduğu bir PC, paralel bilgi alışverişini sağlamak için ADC-DAC grubu, sürme devresi, yük olarak kullanılan jeneratör olarak çalıştırılan bir dc motor, tako ve sinyal çeviriciden oluşmaktadır. Bu çalışmada kullanılan kontrol sistemine ait genel diyagram Şekil 8.1'de verilmiştir.



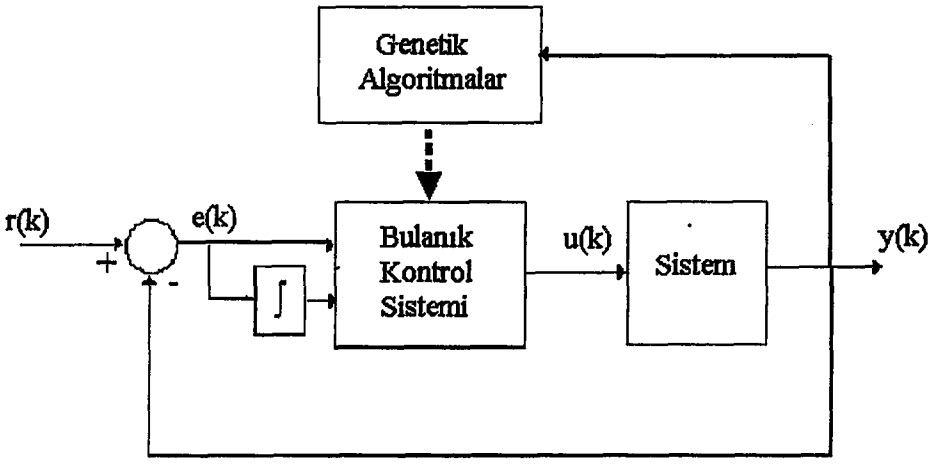
Şekil 8.1 Bu çalışmada kullanılan kontrol sistemine ait genel blok diyagram

8.2 Genetik Bulanık Sistemler

Genetik algoritmalar, doğal genetik ve doğal seçme mekanizmaları üzerine dayanan araştırma algoritmalarıdır. Son yıllarda, öğrenme işleminin temeli olarak genetik algoritmaları alan çok farklı yaklaşımlar kullanılmaktadır. Genetik algoritmalar, daha çok sistem hakkındaki deneyim ve sezgisel yaklaşımlara dayanan bulanık kontrolörün kural taban oluşturma işlemini otomatik olarak tanımlamada kullanılabilecek güçlü bir alet olduğu görülmüştür. Bu tür yaklaşımlar genellikle genetik bulanık sistemler (GBS) olarak tanımlanmaktadır.

8.2.1 Genetik Algoritma Yardımıyla Bulanık Kural Tabanın Öğrenilmesi

Bu çalışmada doğru akım motor hız kontrolü için insan tecrübesini kullanmaksızın Genetik Algoritmalar kullanılarak PI-tip Bulanık Kontrolör tasarlanması amaçlanmıştır (Şekil 8.2). Bu amaç üç aşamada gerçekleştirilmiştir: ilk olarak iteratif olarak çalışan genetik algoritma temelli bir öğrenme algoritması geliştirilerek bulanık kurallar bu algoritma ile çıkarılmıştır. İkinci olarak genetik bulanık öğrenme algoritmasının her nesilde elde edilen tüm kural dizileri motor matematiksel modeli üzerinde simule edilerek en uyumlu kural taban araştırması yapılmaktadır. İstenen performans değerli kural taban bulununcaya kadar veya maksimum nesil sayısına ulaşıncaya kadar algoritma çalıştırılmaktadır. Üçüncü aşamada istene uyumluluk değerine sahip kural tabanı elde edildikten sonra bu kural tabanı ile PI-tip Bulanık Kontrolör yapısına yerleştirilerek direk olarak motor uygulanmaktadır. Bu öğrenme algoritması uygun genetik ve bulanık formülasyon kullanılarak elde edilmiştir. Elde edilen sonuçlardan otomatik olarak GA temelli olarak tasarlanan kontrolörün performansının insan bilgi temelli kontrolörünki ile karşılaştırıldığında iyi olduğu ve nesil sayısı artırılarak daha fazla çözüm noktasının araştırılmış olacağından daha iyi sonuçlar alınabileceği görülmüştür.



Şekil 8.2 Genetik Algoritma temelli PI-tip Bulanık kontrolör öğrenme algoritma blok şeması

Bir bulanık kontrollerin bilgi-tabanı için IF-THEN kurallar tanımlandığı zaman bazı problemler ile karşılaşılır. Kontroller için bilgi-tabanı tanımlamada açık bir problem; hiç kimse kuralların doğru olarak tanımlandığından ve belli durumlarda birbiriyle bir karışıklığa sebep olamayacağından emin olamaz. Bu çalışmada ise bu durumların üstesinden gelmek için kural tabanı GA yardımıyla otomatik belirlenmekte yani sisteme özgü kontrolör öğrenilmektedir.

8.2.2 Öğrenme Algoritmasında Kullanılan PI-tip Bulanık Kontrolcu Yapısı

Öğrenme aşaması ve kontrol aşamasında aynı PI-tip Bulanık Kontrolcu kullanılmaktadır. Öğrenme aşamasında genetik algoritma ile elde edilen kural tabanlar kontrolör üzerinde çalıştırılarak uygunlukları ölçülmektedir. Öğrenme aşaması ya belli bir iteratif döngü (nesil) sayısı kadar veya bir sonlandırma kriterine ulaşıncaya kadar devam etmektedir. Burada çalışmamızda kullanılan PI-tip bulanık kontrolcunun detayları verilmektedir.

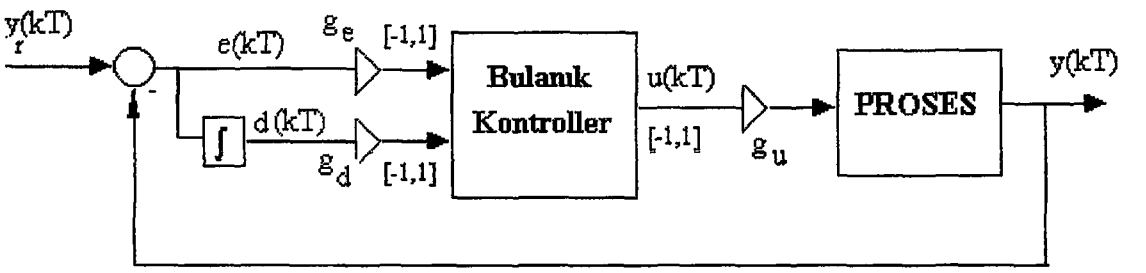
PI tip Bulanık kontrolör için, hata sinyalının integralini bualnıklaştırmada ölçekleme güçlükleri çekildiğinden literatürdeki uygulamalarda PI-like bulanık kontrolör girişleri olarak hata ve hata değişimi kullanılmakta, çıkışta ise kontrol sinyalinin değişimi elde edilmek suretiyle PI özelliği kazandırılmaktadır (Chwee Ng, Kim ve Li Yun ,1998). Bu çalışmada ise bulanık kontrolör girişleri olarak hata ve hata sinyalinin integrali kullanılmış, fakat bazı sınırlandırmalar getirilmiştir. Integral bir toplama işlemi olduğundan alt ve üst limitler belirlenerek saturasyona gidilmesi önlenmiştir.

Kontrol etmeye çalıştığımız dc motor sisteminin girişi u(kT) ve çıkışı y(kT) olarak kabul edilsin. Burada T örnekleme peryodunu temsil etmektedir. PI-tip bulanık kontrollerin girişleri ise, proses çıkışı y(kT) ve referans giriş y_r(kT)'nın fonksiyonu olarak tanımlanmaktadır. PI-tip bulanık kontroller girişleri e(kT) ve hatanın integrali d(kT) olarak tanımlanırsa

$$e(kT) = y_r(kT) - y(kT) \tag{8.1}$$

$$d(kT) = \sum e(kT) \cdot Ts \tag{8.2}$$

olur. Burada y_r(kT) istenen proses çıkışını gösterir.



Şekil 8.3 PI-tip Bulanık kontrol yapısı ve giriş ve çıkışlarında kullanılan ölçekleme katsayıları

Bulanık kontrol teorisinde, kontrolcunun giriş ve çıkışının alabileceği değerler aralığı, parametre "tanım uzayı" olarak adlandırılır. Daha esnek kontroller gerçekleştirmek için herbir proses girişi sabit ölçekleme katsayıları kullanılarak [-1,1] tanım uzayına kaydırılarak normalize edilir. Bulanık kontrol sistemi dizaynında g_e, g_c ve g_u ölçekleme katsayıları sırasıyla, e(kT) hata değeri, d(kT) hatanın integrali ve u(kT) kontroller çıkışının çalışma aralığına normalizasyonu için kullanılmaktadır (Şekil 8.3). Bulanık kontrollerin bir girişi bu durumda $g_e * e(kT)$ dır ve g_e öyle seçilmelidir ki $g_e * e(kT)$'in alacağı değerler [-1, 1] tanım uzayı içinde olmalıdır. Aynı şekilde g_u da proses girişi için izin verilen değerler aralığı kullanılarak seçilir. Yani bir anlamda g_u kontroller çıkışı u(kT) yı prosesin kullanabileceği tanım uzayına taşımalıdır. Hata integrali için ise g_d katsayısı, d(kT) nın alacağı normal değerleri belirlemek için sisteme değişik girişler verilerek deneysel olarak belirlenir, sonra bu değerleri [-1,1] aralığına taşıyacak şekilde belirlenir.

PI- tip bulanık kontroller için kural taban aşağıdaki şekildeki kontrol kurallarından üretilen n.ci proses girişiyle birleştirilir.

IF e_1 is E_1^j **and**....**and** e_s is E_s^k **and** ....**and** d_1 is D_1^l **and**....**and** d_s is D_s^m (8.3)
THEN u_n is $U_n^{j..k,l..m}$

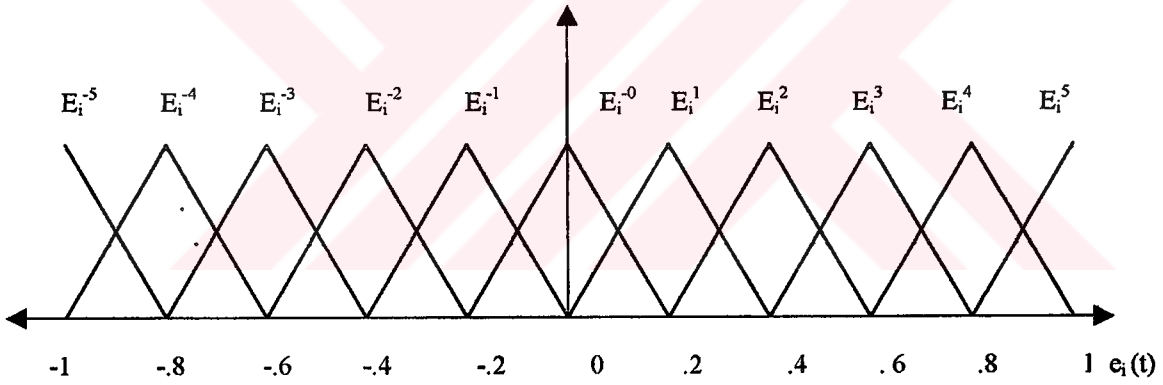
Burada e_i ve d_i bulanık kontrollerin girişlerini ifade eden "dilsel değişkenleri" göstermektedir. u_n kontroller çıkışı ifade eden dilsel değişkeni gösterir. E_i^k ve D_i^k ise sırasıyla e_i ve d_i ile ilişkilendiren dilsel değerleri göstermektedir. Böylece bir örnek olarak bir bulanık kontrol kuralı şöyle yazılabilir:

IF hata **is** positive-large **and** hatanın integrali **is** negative-small
THEN proses-girişi **is** positive-big (8.4)

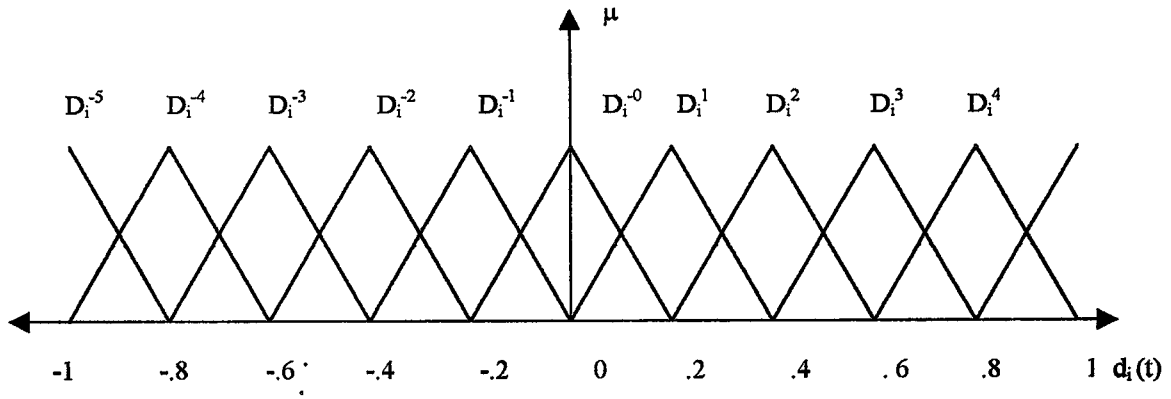
Burada e_1 = hata ve E_1^4 ="positive-large v.s . bu şekildeki kurallar bir kümesi, bir dinamik sistemin nasıl kontrol edildiğini karakterize eden "kural taban"ı oluşturur. Yukardaki kontrol kuralı, bulanık küme teorisi kullanılarak bir bulanık çıkarım formu oluşturulabilir.

IF E_1^j **and....and** E_s^k **and** ....**and** D_1^l **and....and** D_s^m **THEN** $U_n^{j..k,l..m}$ (8.5)

Burada E^j, D^l ve $U^{j..k,l..m}$ sırasıyla "e is E^j", "d is D^l" ve "u is $U^{j..k,l..m}$" dilsel durumlarını tanımlayan bulanık kümeleri gösterir. Yukarıdaki örnek kural için Şekil 5.4'deki tanım uzayındaki üyelik fonksiyonları e(t) için kullanılabilir. Aynı bulanık kümelerinin normalize edilmiş d(t) içinde kullanıldığı kabul edilsin. Bulanık kontroller çıkışı u(t) için tanım uzayı üzerindeki çıkış üyelik fonksiyonları genetik algoritmalar tarafından otomatik olarak belirlenmektedir. Böylece sistemi kontrol eden bulanık kontroller öğrenilmiş olacaktır. Başlangıçta bulanık kontroller ait başlangıç kural taban genetik öğrenme algoritmasının başlangıcında random olarak üretilmektedir.



Şekil 8.4 Bulanık kontroller hata e(t) değişkeni için tanım uzayındaki bulanık kümeler



Şekil 8.5 Bulanık kontroller hatanın integrali d(t) değişkeni için tanım uzayındaki bulanık kümeler

Sistemimizde iki giriş bir çıkış olduğundan bulanık kontrollerin bütün kuralları aşağıdaki formda olacaktır.

IF E_1^j **and** D_1^l **THEN** $U_n^{j,l}$ (8.6)

Burada E_1^j ve D_1^l üçgen tip giriş üyelik fonksiyonlarıdır. Aynı şekilde $U_n^{j,l}$ ise algoritma başlangıcında random olarak merkez değerleri üretilen ve taban genişliği 0.4 olan bir üçgen tip üyelik fonksiyonudur, (Şekil 8.4 ve 8.5). Giriş ve çıkış değerlerinin üyelik derecesi hesaplanmasında aşağıdaki üyelik fonksiyonu kullanılmaktadır.

$$\mu_{E^i}(x) = \begin{cases} \max\left(\left[0,1+\dfrac{x-c_{E^i}}{w}\right]\right), & x <= c_{E^i} \quad i = 1...11 \\ \max\left(\left[0,1+\dfrac{c_{E^i}-x}{w}\right]\right), & x > c_{E^i} \quad i = 1...11 \end{cases} \tag{8.7}$$

Burada; c_E^i , üçgen tip E^i (veya D^i) üyelik fonksiyonun merkezi ve w ise üyelik fonksiyonun taban genişliğinin yarısını göstermektedir. Direkt olarak dizayn edilen geleneksel bulanık kontroller için kontrol kurallarındaki $U_n^{j,l}$ bilgisi prosesi tanıyan bir uzmanın ortaya koyduğu *ön_bilgi* olarak ifade edilir. Burada çıkış üyelik fonksiyonlarını gösteren bu bilgi genetik algoritmalar yardımıyla öğrenilmektedir. Burada durulama işleminde standart ağırlık merkezi yöntemi (COG-center of gravity) kullanılmaktadır.

8.2.3 Bulanık Kural Tabanın Kodlanması

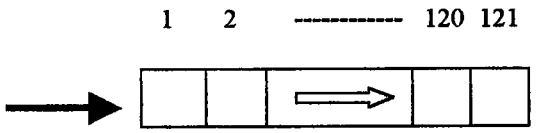
Bu çalışmada genetik algoritma havuzunu oluşturan kromozomlar bir PI-tip bulanık kontrolörün kural tabanlarından oluşturulmaktadır. GA'ları istenen şemaya yerleştirmek için, kural tabanı uzun bir dizi içersine yerleştirilir. Her bir kromozom dizisi, bir bulanık kontrollerin kural tabanını oluşturmaktadır. Herbir nesilde havuzdaki kromozom sayısı kadar kural taban araştırılmış olunacaktır. Böylece bir bulanık kural-tabanı uzayında, kontrol etmeye çalıştığımız dc motor sistemi için en uygun kural tabanı araştırılmaktadır. Havuzdaki kromozom sayısının fazla olması süreyi artırmakla beraber çözüm uzayında daha fazla vektör araştırılacağından sonuca ulaşmayı yakınlaştıracaktır.

Bu çalışmada kromozomlar ikili taban yerine onluk (0-9) tabanda kodlanmaktadır. Bir kuralı oluşturan karakter, bir gerçel sayı ile ifade edilmektedir. Karakterlerimizin alabileceği değerler [-1,1] aralığında sınırlandırılmaktadır. Kurallarımız bir üyelik fonksiyonun tanım

uzayındaki [-1,1] merkez değerini gösterdiğinden ondalık olarak gösterebilmek için kural indisi ile aynı indisli farklı bir değişken kullanılmaktadır Karakterin işaretini gösteren bit için 0 ve 9 değerleri kullanılmaktadır. İlk bit değeri 0 ise karakter negatif, eğer 9 ise karakter pozitiftir.

1.kural	2.kural	3.kural	4.kural	.....	.....	.....	.....	.....	120.kural	121.kural

		E										
		-5	-4	-3	-2	-1	0	1	2	3	4	5
Δe	-5	1	2	3	4	-	-	-	-	-	-	11
	-4	12	13	14	15	16	-	-	-	-		22
	-3	23	24	25	26	27	28			31	32	33
	-2	35									-	44
	-1	45									-	55
	0	56									-	66
	1	67									-	77
	2	78									-	88
	3	89									-	99
	4	100	101	102							110	111
	5	112	113	114	115	-	-	-	-	-	120	121



Şekil 8.6 Bulanık kontrolöre ait kural tabanın bir kromozoma yerleştirilmesi

Her bir kural taban toplam 121 kuraldan oluşmaktadır. Kural tabanı oluşturan kromozon her bir karakteri bir kuralı ifade etmektedir. Populasyon büyüklüğü N=20 ile 40 arasında seçilmiştir. Yani havuzda maksimum 40 kural taban bulunmaktadır. Herbir nesilde 40 adet kural taban dc motora uygulanarak uyumluluk değeri hesaplanmaktadır. Dizide kural sıralanış şekli ve örnek kural aşağıda gösterilmiştir.

8.2.4 Kural taban İçin Uyumluluk Fonksiyonu (Performans Kriteri)

Genetik bulanık öğrenme algoritmasının öğrenme esnasında elde edilen kromozomlar (kural taban) dc motor modeline belirli bir süre (10 sn) uygulanma ve hız-zaman eğrileri çıkarılmaktadır. Kural tabanın sistemde kullanılmasıyla elde edilen bu çıkış belirli bir uyumluluk fonksiyonuna tabi tutularak; kural tabanların performansları ölçülmekte ve sistem için uygunluk derecesi hesaplanmaktadır. Bütün nesiller içinde elde edilen en yüksek uyumluluk değerli kural taban kontrolörde kullanılmaktadır.

Bir kontrol sistemi için en iyi performans tanımlamak güçtür. ideal olarak sistemin referans değerindeki bir değişimle ilgili bir isteği hatasız olarak gerçekleştirmesi beklenir. Pratikte ise ideal olan bu duruma yaklaşma nispetinde kontrol etken olur. Genellikle benimsenen yol bir

performans kriteri veya indeksinin matematiksel olarak tanımlanması ve bu kriter veya indeksi optimum olmasını sağlayan çözümlerin aranmasıdır. Özellikle karmaşık bir kontrol sisteminin en uygun davranışını bir performans indeksi ile göstermenin mümkün olabileceğini düşünmektir. Genellikle kontrollü sistemlerin performansları incelenirken referans değerinde giriş olarak basamak değişmesi alındığı belirtilmiştir.

Diğer bir kriter Ziegler ve Nichols tarafından açıklanan genlik oranı kriteridir. Bu kritere göre genel olarak bir peryot aralıklı iki genlik arasında 1/4 oranı veren cevap fonksiyonu en uygundur ve bu durumu sağlayan kontrol parametreleri de verilmektedir. Bu kriter uzunca devam eden titreşimler ortaya getirmektedir.

Karakteristik değerlerden bir veya birkaçını birlikte kullanma yerine diğer bir yaklaşımla hatayı esas alan matematiksel kriterler geliştirilmiştir. Bu kriter, kontrol çıkışının değişimi referans giriş değeri arasında kalan hata alanını minimum olmasını gerektirmektedir. Hata alanı ve hatanın karesinin alanı şekilde gösterilmiştir. Bu çalışmada sisteme uygun kural tabanı araştırmasında, elde edilen kromozomlar ile temsil edilen kural tabanların performanslarını ölçmek için uyumluluk fonksiyonu tanımlanması yapılmıştır. Kontrolörün performansını doğru olarak değerlendirmek için bir *f(x)* uyumluluk fonksiyonu kullanılacaktır. *f(x)* uyumluluk fonksiyonu, araştırmaya yapılacak konuya ve uygulama tipine göre tanımlanması gerekmektedir.

Bu çalışmada en iyi kural tabanı bulmak için kullanılan uyumluluk fonksiyonu olarak; küçük kalıcı hal hatası, kısa bir yükselme zamanı T_r , düşük osilasyon ve overshootlar ile iyi kararlılık durumunu yansıtan aşağıdaki fonksiyon alınımştır. (Ng K.C., Li Y.,Murray D.J. and Sharman K.C (1995)

$$\textit{uyumluluk}: \qquad f(x) = \exp\left(-\frac{a}{T}\sum_{t=0}^{T} t * e^2 + t * (\Delta e)^2\right) \tag{8.7}$$

Burada T dizayn değerlendirmede kullanılmak için proses modeline uygulama süresi; a uyumluluk alt ve üst tabanlarını ayarlamada kullanılan pozitif bir sayı, *t* zaman indeksi; e, t anında ölçülen sinyal ile istenen referans sinyal arasındaki hata, Δe, t anındaki hata değişimidir. Bu tanımlama altında uyumluluk değeri 0 ile 1 arasında oluşmaktadır., daha yüksek değerlikli kural tabanlar daha iyi kontroller performansına karşılık gelmektedir.

8.2.5 Genetik-Bulanık Sistem (GBS) Öğrenme Algoritması

Bu çalışmada istenilen kural tabanı elde etmek için üç aşamalı bir genetik bulanık sistem metodolojisi oluşturulmuştur. İlk aşama iteratif bir kural öğrenme yaklaşımı üzerine dayalı bir genetik bulanık kural oluşturma işlemidir. İkincisi, her bir nesil boyunca elde edilen en uyumlu kural tabanın sisteme uygulanması , üçüncü aşama ise istenen bir uyumluluk değerini gösteren bir sona erdirme kriteri veya belli sayıdaki nesilden sonra öğrenme işlemi sona erdirilmesi ve tüm nesil boyunca en uyumlu kural tabanlar arasına en iyi uyumluluk değeri taşıyan kural tabanın sisteme kontrol amacıyla uygulanmasıdır.

Genetik üretme işlemi bir iteratif metod ile birlikte bir bulanık kural taban oluşturma metodunu içerir ve iteratif bir kural öğrenme yaklaşımına dayanmaktadır. Bulanık kural taban öğrenme metodu, genetik algoritmanın herbir kromozomunu bir bulanık kural taban olarak gerçel sayılar ile kodlanması yoluyla geliştirilmiştir. Burada kullanılan genetik algoritma her nesilde içersinde en yüksek uyumluluk değerini veren özellikleri içeren kural tabanı sisteme uygulamak suretiyle bulmaktadır.

Bu çalışmada GA, kontrollerde kullanılacak kural tabanını araştırırken aynı zamanda, bulanık kontrolörün performansını artıracak en iyi kalıcı hal hatasına sahip kural tabanını da araştırmaktadır. Böylece GA, bir bulanık Kontrollerin bir sistemi en uyumlu değere sahip kural ile maksimum performans ile kontrol etmesini sağlayacak kuralları tespit etmektedir. Şekil 8.7'de akış diyagramı görülen öğrenme algoritmasını adım adım açıklayacak olursak,

. Öğrenme Algoritması ;

Adım 1. Kural tabanın kodlanmış değerlerinden oluşan genlerin oluşturduğu kromozom yapısının boyutları belirlenir. Ayrıca maksimum nesil sayısı, havuzdaki kromozom sayısı .vb değerler belirlenir . İlk olarak random olarak bir başlangıç kural tabanı havuzu oluşturulur.

Adım 2. Havuzdaki bütün kural tabanları bulanık kontrolör vasıtasıyla modele uygulayarak sistem çıkışı ve hata eğrisini çıkar.

Adım 3. Her bir kromozomun uygulanmasıyla elde edilen sistem hata ve hata değişimi eğrilerini kullanarak kural tabanların uyumluluk değerlerini hesapla.

Adım 4. Öğrenme algoritmasındaki Maksimum nesil sayısına bak; eğer yeterli nesile bakılmışsa adım 7' ye git

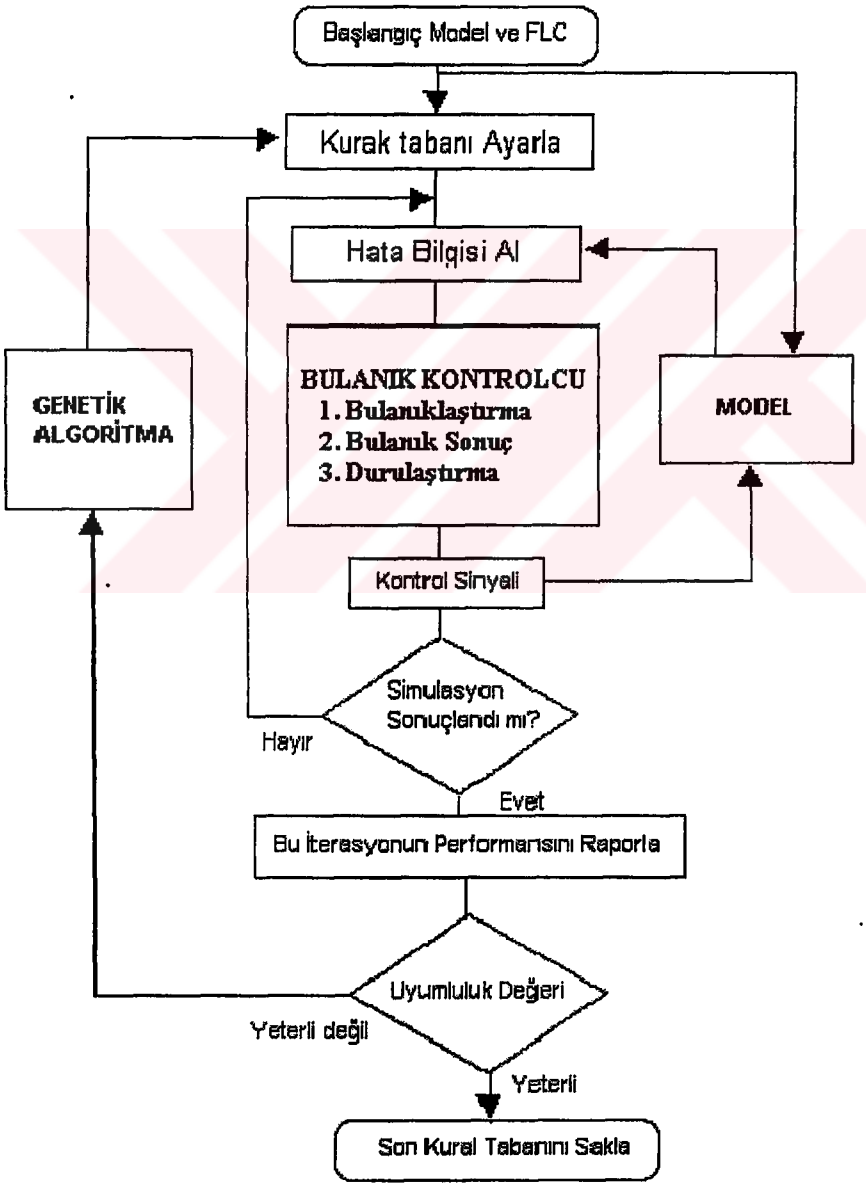
Adım 5. Havuz içinde en yüksek uyumluluğa sahip kromozomları Rulet Tekerleği kullanarak seç. Bunlar havuz içinde saklanacak diğerleri yok edilecektir.

Adım 6. Yüksek uyumluluk değerine sahip kromozomları çaprazlayarak yeni kural tabanları oluştur. Çaprazlanacak kromozom sayısı algoritma basında yüzde değerine göre yapılır.

Adım 7. Havuz içindeki kromozomları verilen yüzdeye göre mutasyona uğrat.

Adım 8. Çaprazlama ve mutasyon işlemleri sonucu yeni bir havuz oluştur. Yeni kural tabanlardan oluşan bu havuzu modele uygulamak üzere Adım 2.'ye git.

Adım 9. En yüksek uyumluluğa sahip kural tabanı sakla. Öğrenme işlemini bitir.



Şekil 8.7 Bu çalışmada geliştirilen Genetik Algoritmalar yardımıyla kural tabanı öğrenme algoritması akış şeması

Öğrenme işlemi sonunda elde edilen bu kural taban, PI-tip bulanık kontrol yapısına yerleştirilerek gerçek dc motor kontrol sisteminde kullanılmaktadır. Öğrenme algoritmasının öğrenme süresi, çeşitli faktörlere bağlı olarak değişmektedir. Havuzdaki kromozom sayısı, seçilen sonlandırma kriteri, maksimum nesil sayısı artıkça süre uzamaktadır. Ayrıca herbir nesildeki kromozomlara yerleşik kural tabanların dc motora uygulanma süresi de etkili olmaktadır. Bu çalışmada havuzdaki kromozom sayısı 20 ve nesil sayısı olarak 10 ve 30 için algoritma çalıştırılarak sonuçlar alınmıştır.

8.3 PC Temelli Bulanık Kontrol Yapısında Giriş-Çıkış Değişkenleri Tanımı

Bu çalışmada bilgisayar temelli dijital kontrol yapıldığından öncelikle sistemin transfer fonksiyonun bir parametresi olan örnekleme zamanın tanımlanması gerekmektedir. Örnekleme zamanı servo mekanizmalarda genellikle 10 Hz ile 1 kHz arasında seçilir. Buna göre saniyede alınabilecek örnek sayısı 10 ile 1000 arasında değişmektedir. Örnekleme peryodu olarak 0.1 sn ile 0.001 sn arasında olmaktadır.

Bu çalışmada çok girişli - tek çıkışlı (MISO) bulanık kontroller yapısı kullanılmıştır. Girişler olarak hız hatası (e), hedeflenen hızdan ölçülen hızın çıkarılmasıyla elde edilir.

$$e = V_{ref} - V_{takometre} \tag{8.8}$$

Hız hatasının integrali ise ileri yol integrasyonu kullanılarak elde edilir. Burada önce hız hatası hesaplanır. Bu değer örnekleme peryodu ile çarpılır. Integral işlemi hata pozitif iken sürekli bir toplama işlemi yapılmaktadır. Bu işlem sonunda toplam değer 2^8=255 değerini aşarsa, 8 bitlik DAC kullanılması sebebiyle bu değerde sabit tutulur. Aynı şlem çıkarma ilemi içinde yapılmakta eğer toplam değer sıfira ulaşması durumunda, bir sonraki değerin negatif olması durumunda ilave edilmemektedir.

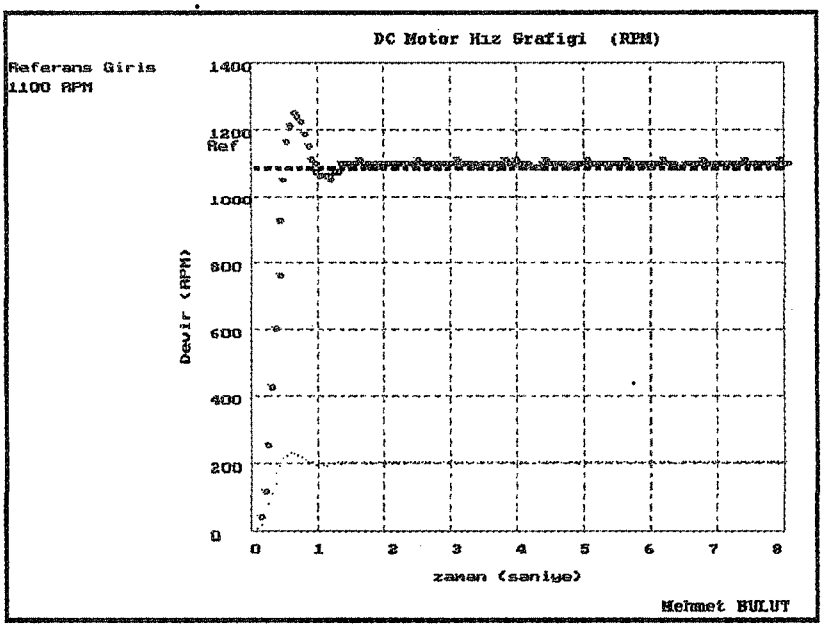
$$\int e\,dt = \sum \left[V_{ref}(n) - V_{tako}(n)\right] * Ts = \sum_{n=0}^{Bitişsüresi} e(n) \cdot Ts \tag{8.9}$$

Yukarda değinildiği gibi , PI-tip Bulanık Kontrolör girişleri, motor hız bilgisi ile referans giriş arasındaki fark olan hata (e) ve hatanın integrali (eint) alınmıştır. Giriş değişkenleri hata ve hatanın integrali olduğundan tasarlanan bulanık kural tabanı bir PI kontrolöre ait katsayı bilgisini de içerecek şekilde olmaktadır. Kontrolünü yapacağımız motorun maksimum hızı 1400 dev/dak (rpm) dir. Bu yüzden hatanın alabileceği değerler [-1400, +1400] arasındadır.

Genetik algoritmalar ile tasarım esnasında kontrolör için hatanın tanım aralığı, osilasyon etkisini azaltmak ve sisteme referansa daha kararlı oturtmak için [-1000,1000] olarak alınmıştır.

Bulanık kontrolcunun çıkışı, kontrol sistemimiz 8-bit giriş-çıkış veri iletişimine sahip olduğundan (0-255) arasında kodlanmaktadır. Yazılan bulanık programda girişlerin değer aralıkları tasarlanan kart özelliği ve paralel port kullanılmasından dolayı [-255, 255] aralığına kaydırılmıştır. Böylece hata ve hata değişimi programda, motorun maksimum hız değerini olan 1400 rpm olduğundan bu parametrelerin alabileceği değerler [-1400, 1400] arasındadır. Bu değer aralığı da 8-bit ile çalışıldığından [-255, 255] arasına kaydırılarak PI-tip bulanık kontrol programına girilmektedir. PI –tip bulanık kontrol çıkışı ise [0, 255] değerleri arasında alınmaktadır. Bu değer aralığı 0-24 volt besleme değerine karşılık gelmektedir.

Bu çalışmada kullanılan C programına ait grafik ekranı Şekil 8.8'te gösterilmektedir. Bu ekranda motordan start anından 8 sn'ye kadar alınan hız bilgisi ekranda çizdirilmektedir. Kontrol devresi 8-bit olduğundan 1400 / 255 = 5.49 devirlik çözünürlüğe sahiptir. Yani grafik üzerindeki hız en az 5.49 rpm'lık değişimler ile oluşmaktadır. Grafik ekranından motor kontrol sistemine dışardan gelecek bir bozucunun (disturbance) etkisi görülebilmekte ve bozucunun etkisi ortadan kalktığı zaman kontrolörün hemen kendisini toparladığı ve verilen referansa sorunsuz oturduğu görülmektedir.



Şekil 8.8 Bu çalışmada kullanılan bulanık kontrol programının grafik ekranı

8.3.1 Bulanık Kontrol Programının DC Motor Sistemi ile Veri İletişimi

Bu çalışmada kontrol algoritması Turbo C programa dilinde yazılmış ve dc motor kontrol sistemi ile program arasında bilgisayarın paralel port pinleri kullanılarak bilgi alışverişi sağlanmıştır. Orijinal IBM-PC'lerin paralel portlarında DATA, STATUS ve CONTROL olmak üzere üç adres portları mevcuttur. DATA portu bilgisayardan 8-bit veri çıkışı sağlamak için kullanılmaktadır. Bu porta 0x378 adresinden ulaşılmaktadır. Veri çıkışı ise outportb(0x378,N) komutu ile yapılmaktadır. Burada N çıkış yapmak istediğimiz veridir ve bir 0-255 arasında bir tamsayı olmak zorundadır.

DC motor sisteminden veri almak için STATUS ve CONTROL portlarının giriş olarak kullanılan pinlerinden 8 adeti kullanılmıştır. Paralel portun bu giriş pinlerine ulaşmak için 0x379 adresi kullanılır. Burada veri okunurken giriş olarak kullanılan bütün pinlere ulaşmak için Turbo C nin 16-bit okuma yapan komutu kullanılmış ve maskeleme-kaydırma yoluyla giriş olarak kullanılan 8-bit veriye ulaşılmıştır. Komut olarak

M = inport(0x379) (8.10)

M=(M & 0x07fc) >> 3 (8.11)

Bu şekilde yazılan programa 8-bit veri girişi sağlanmış olur.

8.4 Alınan Bilgilerin Kontrol Sisteme Uygulanması ve Değerlendirilmesi

8.4.1 GBS ile Motor Modeli Üzerinden Alınan Sonuçlar

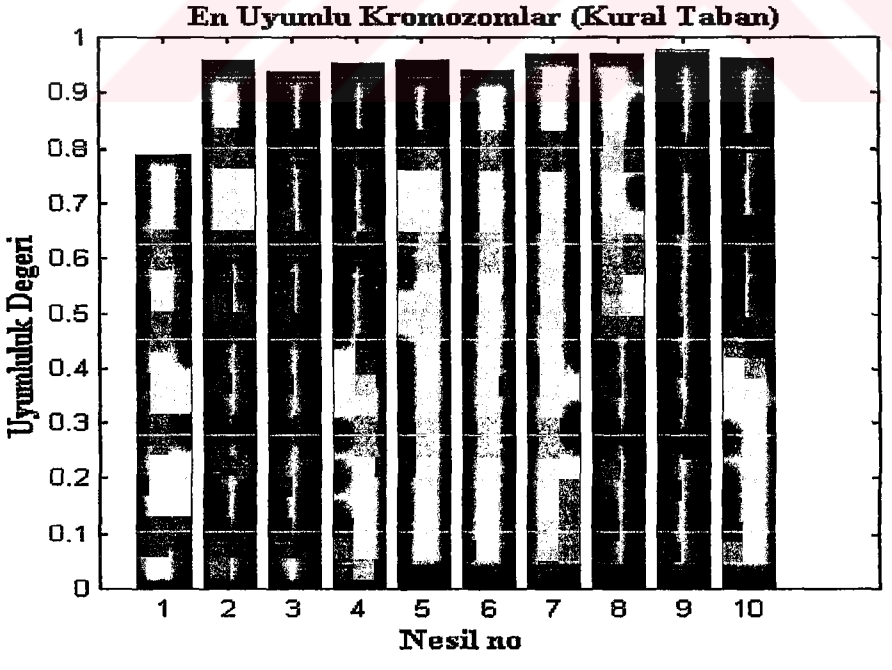
Genetik bulanık sisteminde ilk başlangıç havuzu rastgele olarak oluşturulduğundan dolayı algoritmanın her çalıştırılmasında farklı sonuçlar alınabilmektedir. Daha önceki konularda değinildiği gibi GBS sistemi elde edilen, uyumluluk değeri yüksek kural tabanlar kontrol edilecek sistemin matematiksel modeli üzerinde uygulanmakta ve sonuçları gözlenmektedir. Belirlenen bir sonlandırma kriteri veya belli sayıdaki nesilden sonra, her nesildeki en uyumlu kural tabanlar arasında performans değeri en yüksek çıkan kural taban öğrenilen kural taban olarak alınmaktadır. Aşağıda 10 nesil için çalıştırılan GBS sisteminin sonuçları verilmiştir. Şekil 8.9'de on nesil boyunca uyumluluk değeri en yüksek çıkan kural tabanların bir grafiği ve bunların uyumluluk değerleri Çizelge 8.1'de verilmiştir. Buradan görüldüğü gibi 10 nesil boyunca en yüksek değerlikli kromozom (kural taban) 9.nesilde elde edilen kural tabandır.

Elde edilen bu en yüksek uyumluluk değerli kromozoma yerleşik kural taban, PI-tip bulanık kontrol yapısına yerleştirilerek kontrol edilecek sisteme uygulanmaktadır. Yani bur off-line bir öğrenme olmaktadır.

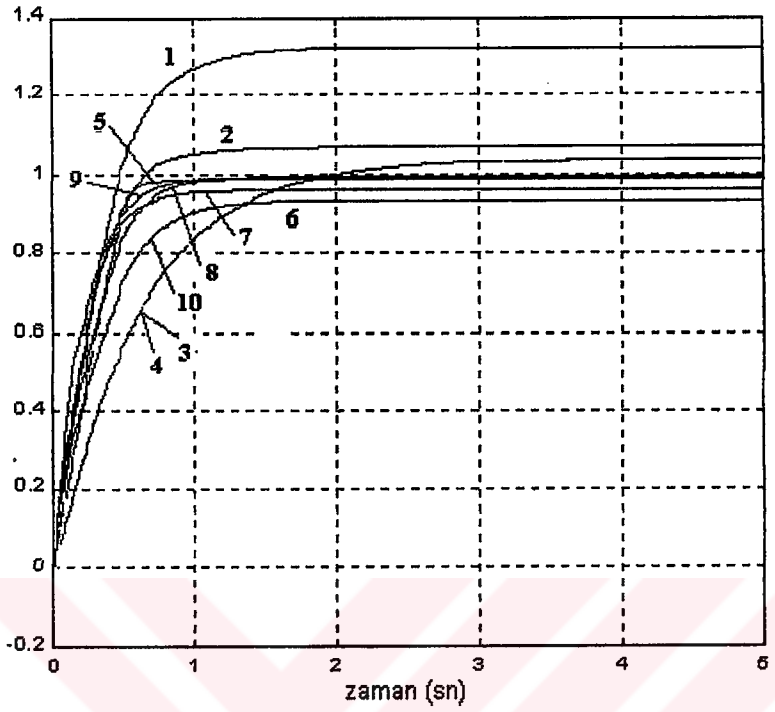
Genetik bulanık sistemin 10 nesil boyunca elde edilen en yüksek uyumluluk değerli kural tabanların aynı giriş-çıkış üyelik fonksiyonlarına sahip PI-tip Bulanık kontrol sisteminde kullanılarak DC motor sistemin matematiksel modeline uygulanmasıyla elde edilen birim basamak cevapları Şekil 8.10'de verilmektedir.

Çizelge 8.1 En uyumlu kural tabanların uyumluluk değerleri

NESİL	1	2	3	4	5	6	7	8	9	10
UYUMLULUK DEĞERİ	0.791	0.968	0.95	0.962	0.966	0.964	0.974	0.974	0.977	0.972



Şekil 8.9 Bütün nesiller boyunca en yüksek uyumluluk değerleri

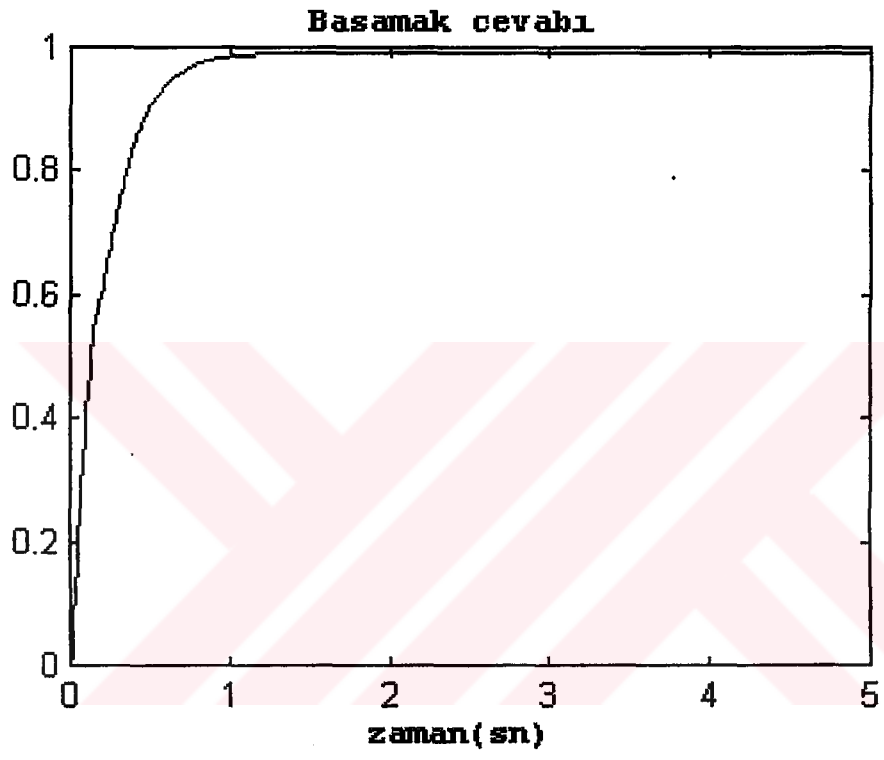


Şekil 8.10 Her bir nesilde (10 nesil) öğrenilen en uyumlu kural taban için DC motor sisteminin birim basamak cevapları

Çizelge 8.2 Bütün nesiller içinde elde edilen en iyi yüksek uyumluluk değerli kural tabanın FAM tablosunda gösterimi

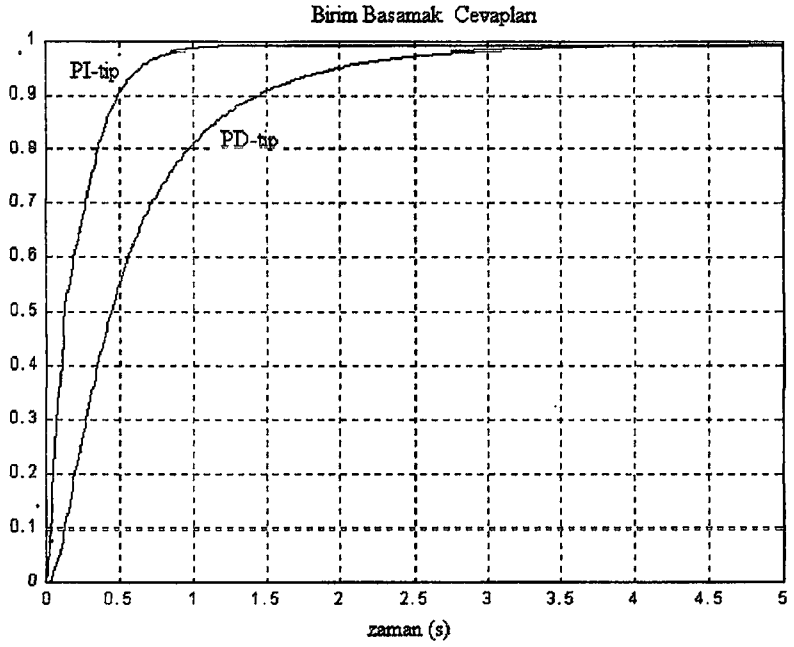
u		e										
		-5	-4	-3	-2	-1	0	1	2	3	4	5
eint	-5	1.0	0.9	0.8	-1.0	0.5	0.2	0.9	-1.0	1.0	0.4	0.2
	-4	-0.4	-0.3	0.8	-0.4	0	-0.2	1.0	1.0	0	1.0	-0.1
	-3	0.3	0.8	0.8	0.5	1.0	0.7	0.9	0	0	0.5	-0.1
	-2	1.0	-0.9	-1.0	-0.5	-0.7	-0.7	0	1.0	-1.0	-0.4	0.8
	-1	-0.9	0.5	0	0.3	0.4	-1.0	-0.1	0	-0.3	0.8	0
	0	-1.0	-0.7	-0.6	-0.4	1.0	0.2	1.0	-1.0	1.0	0	0.5
	1	-1.0	1.0	0.7	-0.9	-0.9	0.6	-1.0	0.8	-0.6	1.0	1.0
	2	-0.2	-0.1	0.3	0.6	1.0	-1.0	0.4	0.8	0.5	0.2	1.0
	3	-0.9	-1.0	0.9	-0.2	-0.9	0.3	1.0	0.6	-0.3	-1.0	-1.0
	4	1.0	-0.7	1.0	0.3	0.9	-0.3	-0.6	-1.0	-1.0	-1.0	-0.2
	5	-0.9	0.4	0.5	0.5	0.2	0.7	-1.0	1.0	1.0	0.6	0.6

Motor transfer fonksiyonu üzerinden alınan tüm cevapların aşım yapmadığı ve salınımsız ideal bir cevap niteliğinde oldukları görülmektedir. Bütün nesiller içinde en yüksek uyumluluk değerine sahip, 9. nesilde elde edilen kural taban için DC motor transfer fonksiyonun birim basamak cevabı Şekil 8.11'de verilmiştir. Elde edilen birim basamak cevabı 1 sn içinde kararlı bir şekilde referans değere oturduğu görülmektedir.

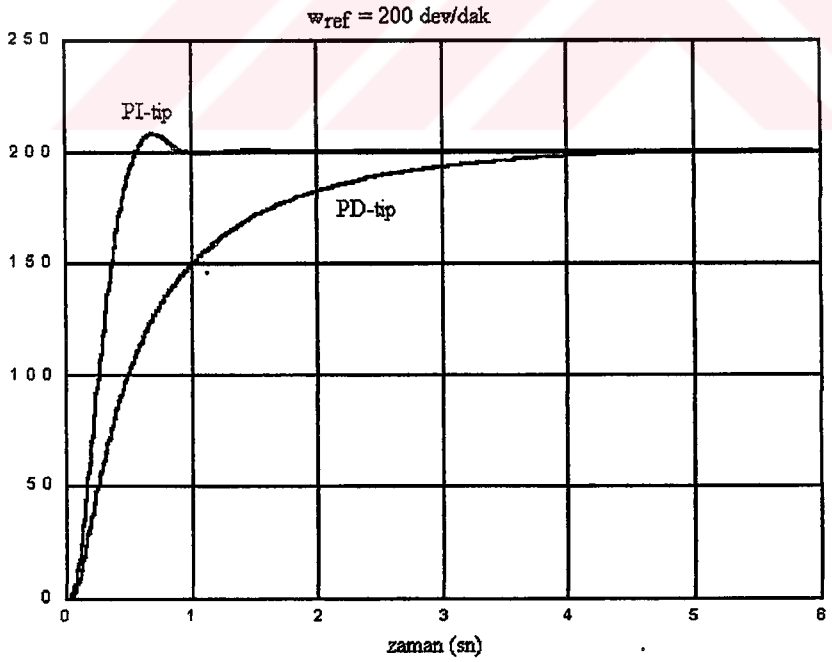


Şekil 8.11 PI-tip bulanık kontrolör için GA kullanılarak elde edilen ve bütün nesiller içinde en uyumlu kural taban için DC motor sisteminin birim basamak cevabı

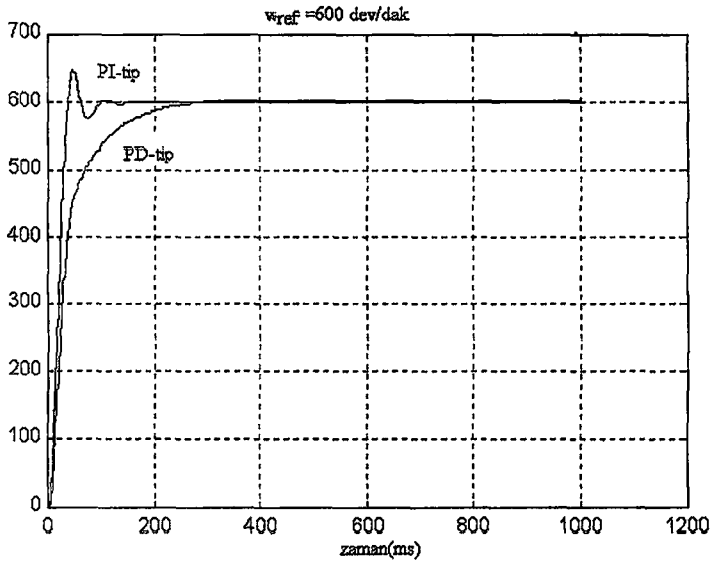
Tasarlanan öğrenme algoritması ile elde edilen PI-tip bulanık kontrolun çeşitli referans değerlerinde elde edilen sonuçları ve Bölüm 5'te anlatılan FMRLC öğrenme algoritması ile elde PD-tip bulanık kontrolcunun sonuçları ile karşılaştırmalı olarak Şekil 8.11-13'te verilmiştir. Bu şekillerden de görüleceği üzere daha verimli olduğu görülmektedir. Şekil 8.11'deki birim basamak cevaplarında PI-tip bulanık kontrolör sistemi 1. sn'de referans değerine oturturken PD-tip bulanık kontrolör ancak 3 sn'de referans değerine oturtmaktadır.



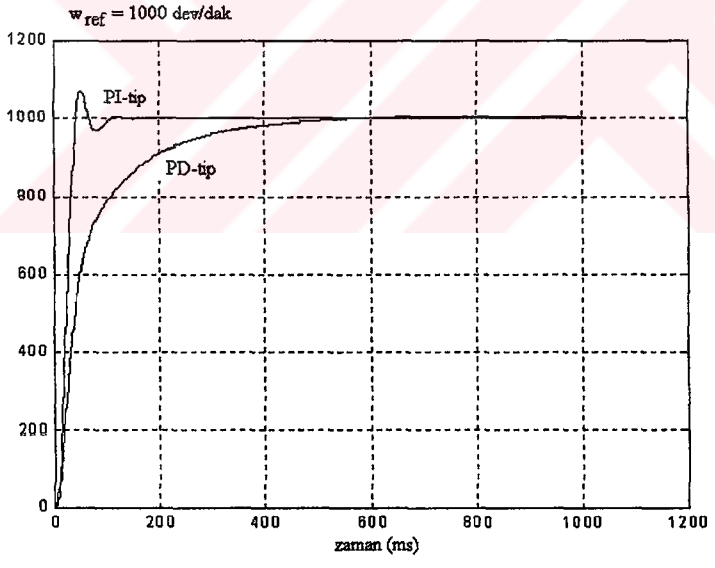
Şekil 8.12 PI-tip Bulanık kontrolcu ile PD-tip bulanık kontrolcu ile dc motor sisteminin birim basamak cevapları



Şekil 8.13 PI-tip Bulanık kontrolcu ile PD-tip bulanık kontrolcunun 200 dev/dak referans girişi için hız-zaman eğrileri



Şekil 8.14 PI-tip Bulanık kontrolcu ile PD-tip bulanık kontrolcunun 600 dev/dak referans girişi için hız-zaman eğrileri



Şekil 8.15 PI-tip Bulanık kontrolcu ile PD-tip bulanık kontrolcunun 1000 dev/dak referans girişi için hız-zaman eğrileri

Elde edilen hız-zaman eğrilerinden de görüleceği üzere genetik algoritmalar ile tasarlanan PI-tip bulanık kontrolörün daha uygun sonuçlar vermektedir. PI-tip ve PD-tip bulanık kontrolcunun sonuçlarının çeşitli performans kriterlerine ait karşılaştırmalı sonuçları Çizelge 8.7'de verilmektedir.

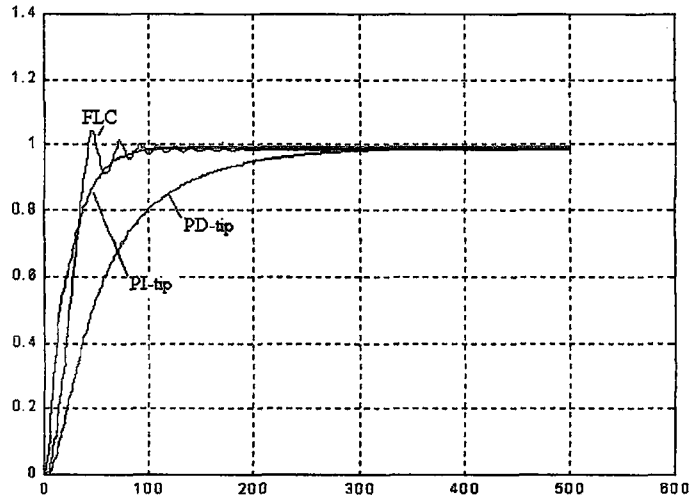
Çizelge 8.3 FMRLC ile öğrenilen PD-tip bulanık kontrol ile Tasarlanan Genetik temelli PI-tip bulanık kontrollerin performans karşılaştırılması

Performans Kriteri	600 dev / dak		1000 dev/dak	
	FMRLC ile PD-Tip Bulanık Kont	GA Temelli PI-tip Bulanık Kont	FMRLC ile PI-Tip Bulanık Kont	GA Temelli PI-tip Bulanık Kont
Tr Yükselme zamanı	2.9 sn	0.7	4.1 sn	0.8 sn
Ts, Yerleşme zamanı	3.1 sn	1.2 sn	4.3 sn	1.3 sn
ISE ($e^2(t)$)	1531000	752390	32007000	16089000
ITAE ($t\|e(t)\|$)	1209200	302560	7706600	451430

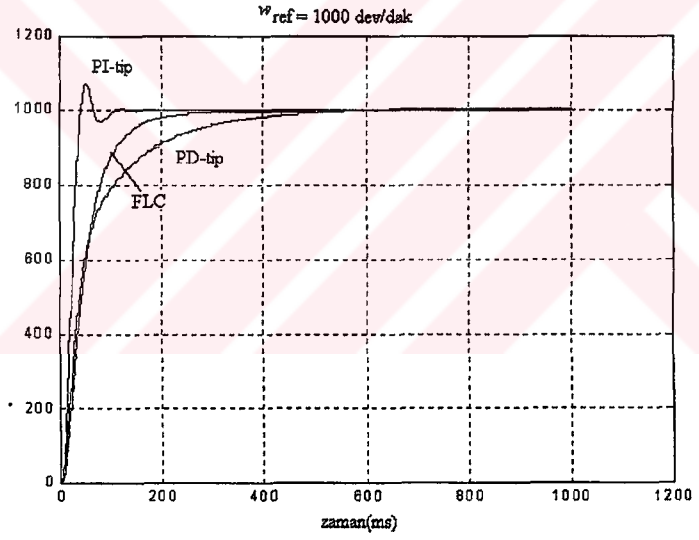
Tasarlanan kontrolcunun performansını ölçmek için dc motor sistemini kontrol edecek bir bulanık mantık kontrolör (FLC) manuel olarak geliştirilmiştir. FLC ye ait kural taban tasarlanırken çeşitli uygulamalarda kullanılan simetrik kural taban yapısı kullanılmıştır. Bu Bulanık kontrolör için giriş ve çıkış parametrelerinde bu çalışmada tasarlanan sistemde kullanılan üçgen üyelik fonksiyonları kullanılmıştır. Kural taban, dc motor sistemine uygulanmış ve cevabı iyileştirmek için bazı kurallarda değişiklik yapılmıştır. Bu değişiklikler dc motora kural tabanın uygulanması ile edilen cevaplara ve uygulama esnasında aktif olan kurallar göz önüne alınarak yapılmıştır. FLC ye geliştirilen ait kural taban matrisi Çizelge 8.8'de verilmektedir. FLC ye ait birim basamak cevabı da Şekil 8.15' da verilmektedir.

Çizelge 8.4 Manuel olarak tasarlanan FLC ye ait kural taban

U		e										
		-5	-4	-3	-2	-1	0	1	2	3	4	5
de	-5	-1	-1	-1	-1	-1	-1	-0.8	-0.6	-0.4	-0.2	0.2
	-4	-1	-1	-1	-1	-1	-0.8	-0.6	-0.4	-0.2	0.2	0.4
	-3	-1	-1	-1	-1	-0.9	-0.6	-0.4	-0.2	0.2	0.4	0.6
	-2	-1	-1	-1	-0.9	-0.8	-0.4	-0.2	0.2	0.4	0.6	0.8
	-1	-1	-1	-0.9	-0.8	-0.6	-0.2	0.2	0.4	0.6	0.8	0.9
	0	-1	-0.9	-0.8	-0.6	-0.4	0	0.4	0.6	0.8	0.9	1
	1	-0.9	-0.8	-0.6	-0.4	-0.2	0.2	0.6	0.8	0.9	1	1
	2	-0.8	-0.6	-0.4	-0.2	0.2	0.4	0.8	0.9	1	1	1
	3	-0.6	-0.4	-0.2	0.2	0.4	0.6	0.9	1	1	1	1
	4	-0.4	-0.2	0.2	0.4	0.6	0.8	1	1	1	1	1
	5	-0.2	0.2	0.4	0.6	0.8	1	1	1	1	1	1



Şekil 8.16 Manuel FLC ile PI ve PD tip bulanık kontrolörlerin birim basamak girişi için hız-zaman eğrileri



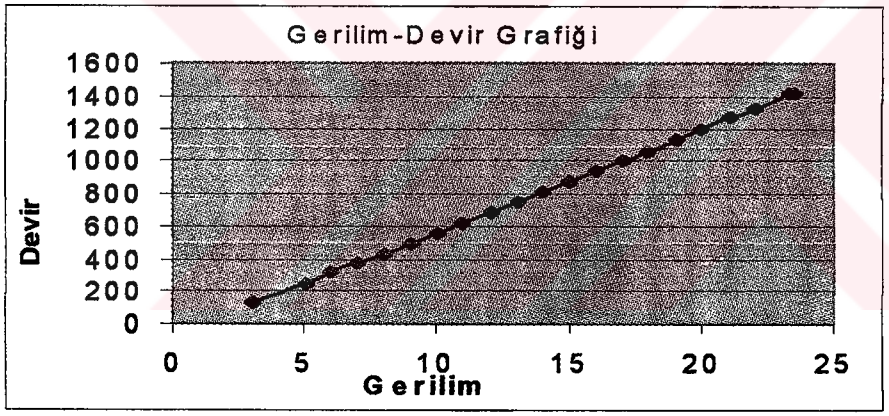
Şekil 8.17 Manuel FLC ile PI ve PD tip bulanık kontrolörlerin 1000 dev/dak girişi için hız-zaman eğrileri

Şekil 8.15'te görüldüğü gibi salınımlı ve kararlı hal hatası yüksek ve 1000 dev/dak referans girişi için yerleşme zamanı uzun olan bir cevap alınmıştır. İyi bir cevap alınması için kontrol ettiğimiz sistemin özelliklerini daha iyi temsil edecek kural taban tasarlanması gerekmektedir. Bu da ancak uzun zaman alan ve sürekli deneme-yanılma metoduna başvurularak başarılabilir. Şekil 8,16-17'de Genetik Bulanık sistem algoritması ile elde edilen PI-tip Bulanık kontrolör cevabıyla karşılaştırıldığında bu çalışmada tasarlanan sistemin iyi bir performansa sahip olduğu görülmektedir. Çünkü Genetik Bulanık Kontrol algoritması bir araştırma yöntemi olduğundan sistemi temsil edecek kural tabanı model üzerinde uygulayarak karar vermektedir.

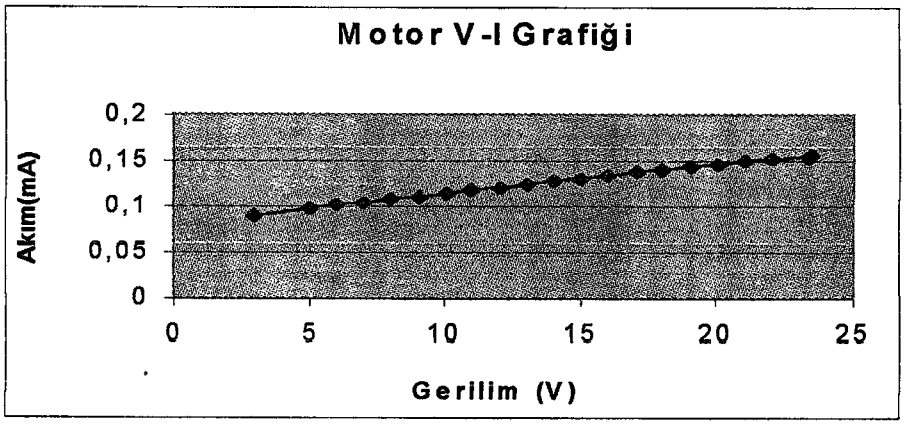
8.4.2 Öğrenilen Kural Tabanın DC Motor Kontrol Sistemine Uygulanması

Bu çalışmada tasarlanan sistemde, Pentium-120 Mhz bir bilgisayarın paralel portunu iki yönlü veri iletişiminde (8-bit) kullanacak arabirim ve sürme devresi dizayn edilerek serbest uyarmalı DC motorunun hız kontrolü gerçekleştirilmeye çalışılmıştır. Kontrol edilen sistem, mekanik olarak birbirine bağlanmış takometresi üzerinde bir dc motor ve yük olarak kullanılan dc motor-jeneratör gurubundan oluşmaktadır.

Bu tezde kontrol edilmesi istenen DC motor Sinano Electric Co. (Japonya) firması ürünü, üzerinde fotodiyot-fototransistör ikilisinden oluşan hız sensörü bulunmakta 24 V besleme gerilimiyle 1400 dev/dak ile dönmektedir. Kontrol edilecek DC motora ait gerilim-devir ilişkisi ve V-I grafiği foto-takometre kullanılarak elde edilen değerler ile çıkarılmıştır. Bunlar Şekil 8.18-a,b 'de verilmektedir.



(a)



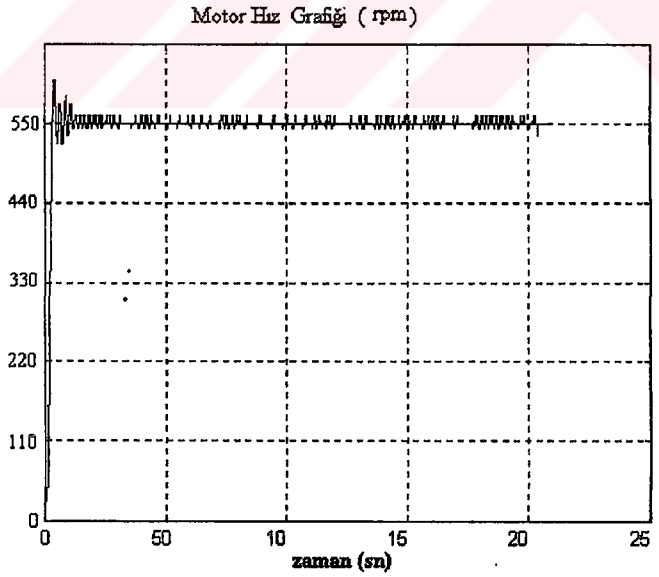
(b)

Şekil 8.18 Kontrol edilecek DC motora ait a) Gerilim-devir (V-n) ve b) V-I grafikleri

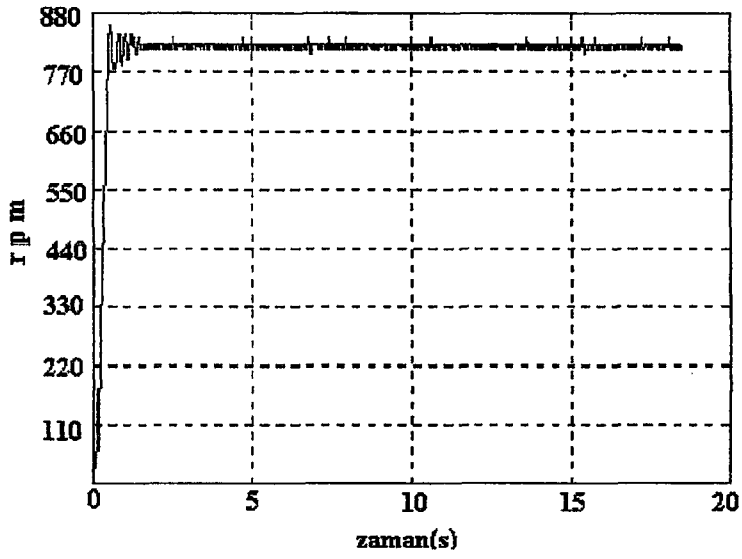
Kontrol edilen sisteme, yük bağlanılmadan önce tasarlanan GBS algoritması ile tasarımı yapılan Bulanık-PI kontrolöre referans giriş değerleri uygulanarak motor hızları grafiksel olarak elde edilmiştir. Dc motor kontrol sistemimize yük olarak 8V'lık bir dc motor kaplin yoluyla eksenel olarak eklenmiştir. Kontrol edilen DC motor sistemi yük altında çeşitli referans değerlerinde çalıştırılmış ve olumlu sonuçlar alınmıştır.

8.4.2.1 DC Motor Sisteminden Yüksüz Durumda Alınan Bilgiler

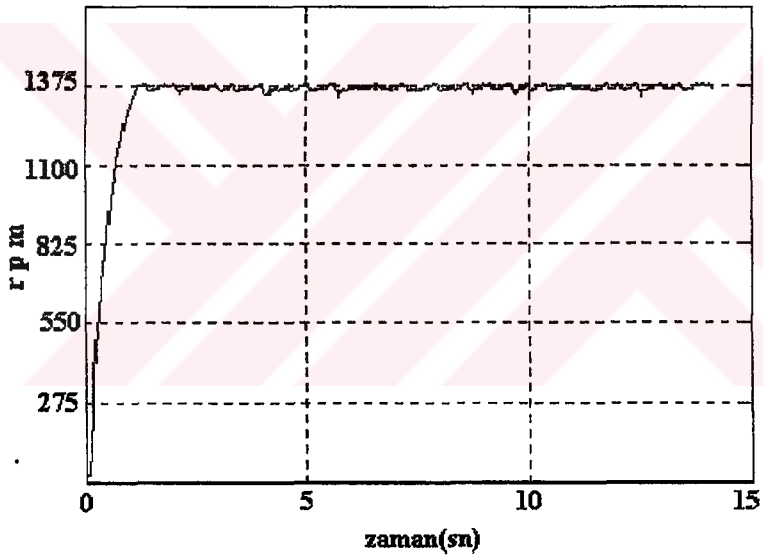
Bilgisayar üzerinde çalıştırılan PI-tip Bulanık kontrolör programıyla kontrol sistemine 8-bit DAC ile referans hız komutları verilebilmektedir. Kontrol ettiğimiz motorun hızı maksimum hızı olan 1400 dev/dak değeri sisteme verilirken 255 olarak verilmektedir. Aynı şekilde motor sisteminden de kontrol programına 8-bit üzerinden hız bilgisi alınmaktadır. Programda ilk olarak daha önce kural tabanı verilen PI-tip Bulanık kontrolör sistem üzerinde çeşitli referans değerlerinde uygulanmış ve alınan sonuçlar Matlab programıyla çizdirilmiştir. Motor sistemi yüksüz iken PI-tip Bulanık kontrolörden alınan sonuçlar Şekil 8.19 ve Şekil 8.20-b,c'de 550, 875 ve 1375 dev/dak referans değerleri için alınan sonuçlar verilmiştir. Sonra programa grafiksel izleme ekranı eklenip sonuçlar direkt online olarak gözlenmesine olanak sağlanmıştır.



Şekil 8.19 550 dev/dak referans girişi için dc motor sistem cevabı, TL=0 Nm



(a)

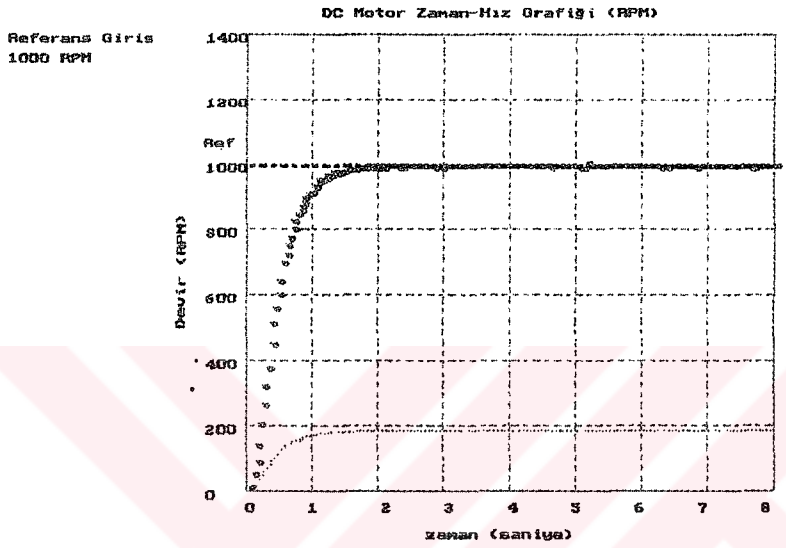


(b)

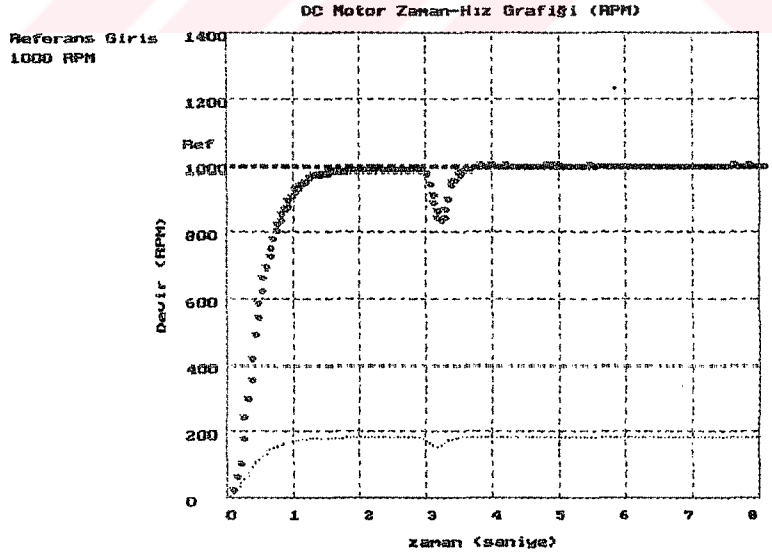
Şekil 8.20 a) Motor yüksüz durumda (T_L=0) iken 825 dev/dak referans girişi için DC motor sistem cevabı b) 1375 dev/dak referans girişi için DC motor sistem cevabı

GFS algoritmasının 10 nesil çalıştırılmasıyla elde edilen kural taban kullanılarak oluşturulan Bulanık-PI kontrolör ile DC motor sistemine yük yokken uygulanmasıyla alınan sonuçlara bakıldığında 550 dev/dak ve 825 dev/dak hız zaman eğrilerinde transient bölgesinde yüksek bir salınımın olduğu kalıcı hal durumunda bunun düşük bir şekilde devam ettiği görülmektedir. 1375 dev/dak' nın hız –zaman eğrisinde de cevabın kısman salınımlı olduğu görülmektedir. Bu nedenle 10 nesil için elde edilen kural tabanın sistemi yeterli şekilde temsil etmediği düşünülmektedir. Bu nedenle tasarlanan öğrenme algoritması 30 için çalıştırılarak

elde edilen kural taban kullanılmıştır. Elde edilen bu kural taban gerçekleştirilen bilgisayar temelli dc motor kontrol sistemi üzerinde uygulanmış ve Şekil 8.21'daki sonuçlar alınmıştır. Dc motor yüksüz durumda çalıştırılmaktadır. Bu kural taban 1000 dev/dak referans girişi için uygulanmıştır. Ayrıca bu referans girişin cevabı esnasında bozucu ilave edilerek sonucu gözlenmiştir. Bozucunun etkisi ortadan kaldırmaya çalıştığı, sistemin bozucunun etkisinden kurtulunca verilen referans değere kısa zamanda oturduğu görülmüştür görülmektedir.



(a)

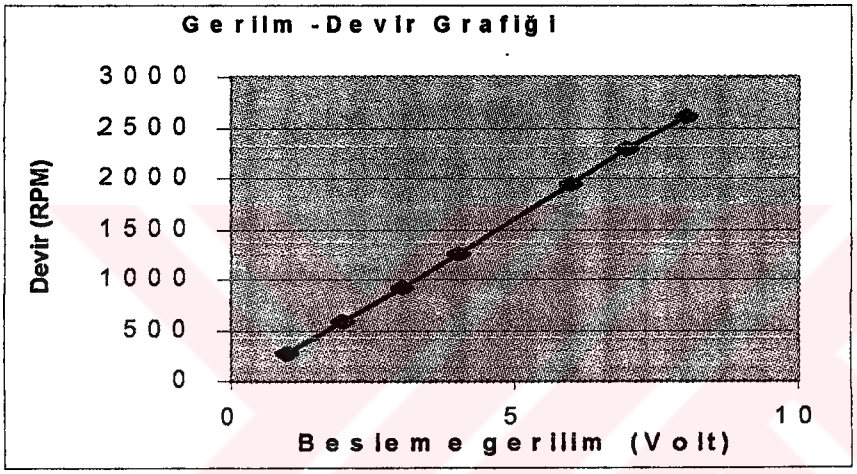


(b)

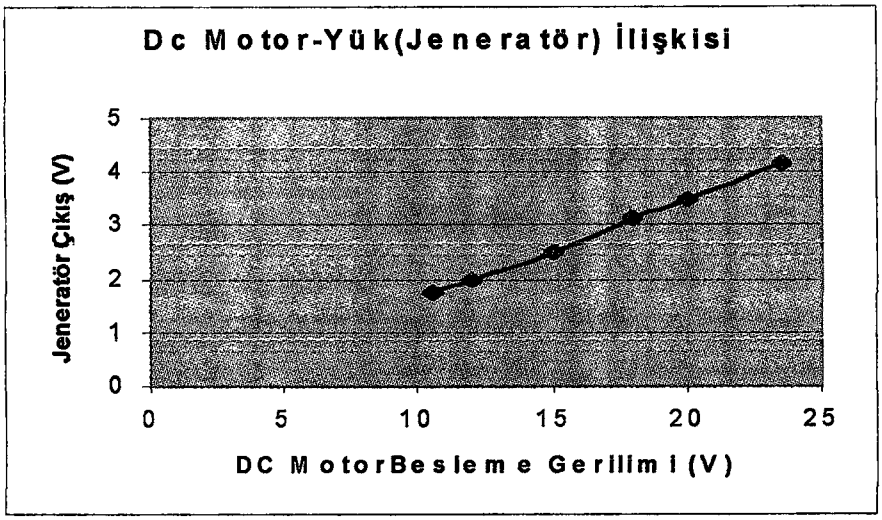
Şekil 8.21 Genetik Bulanık Sistem ile 30 nesil için öğrenilen kural tabanın yük yokken a) 1000 dev/dak referans girişe motor sistemin cevabı b) 1000 dev/dak referans giriş uygulanmışken bozucu etkinin devreye sokulup çıkarılması

8.4.2.2 DC Motor Sisteminden Yük Bağlı Durumda İken Alınan Bilgiler

Bu çalışmada yük olarak, kontrol edilmek istenen DC motora tunçtan yapılmış kaplin ile birleştirilmiş ve jeneratör modunda kullanılan DC motor kullanılmıştır. Yük olarak kullanılan motor 8 V besleme gerilimi altında 2600 dev/dak maksimum hızda dönmektedir. Bu motorun besleme gerilimi-devir ilişkisini gösteren grafik Şekil 8.22'de verilmiştir. Kontrol edilecek DC motor ve yük olarak kullanılan DC motor-jeneratör kaplin ile bağlı iken motor çeşitli gerilimler ile beslenerek buna karşılık jeneratörden gerilimler elde edilmiş ve elde edilen değerler Şekil 8.23'de grafik üzerinde gösterilmiştir.

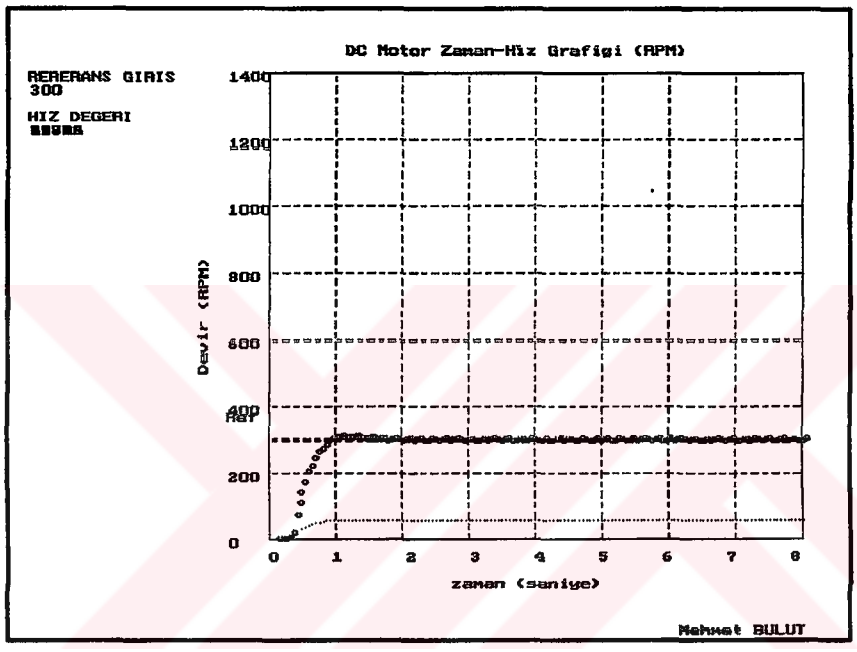


Şekil 8.22 Yük olarak kullanılan DC motorun gerilim-devir grafiği

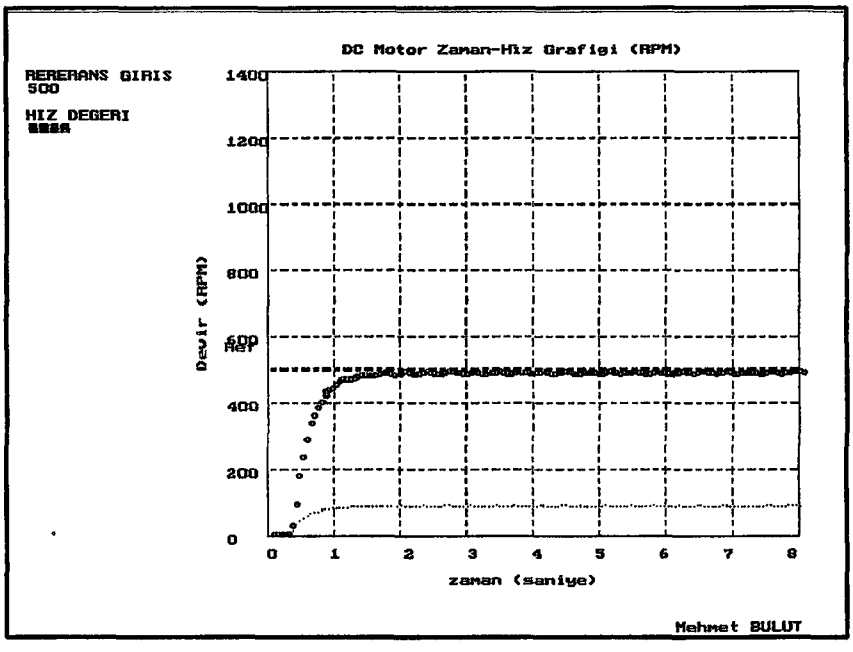


Şekil 8.23 DC Motor besleme gerilimi -jeneratör çıkışı grafiği

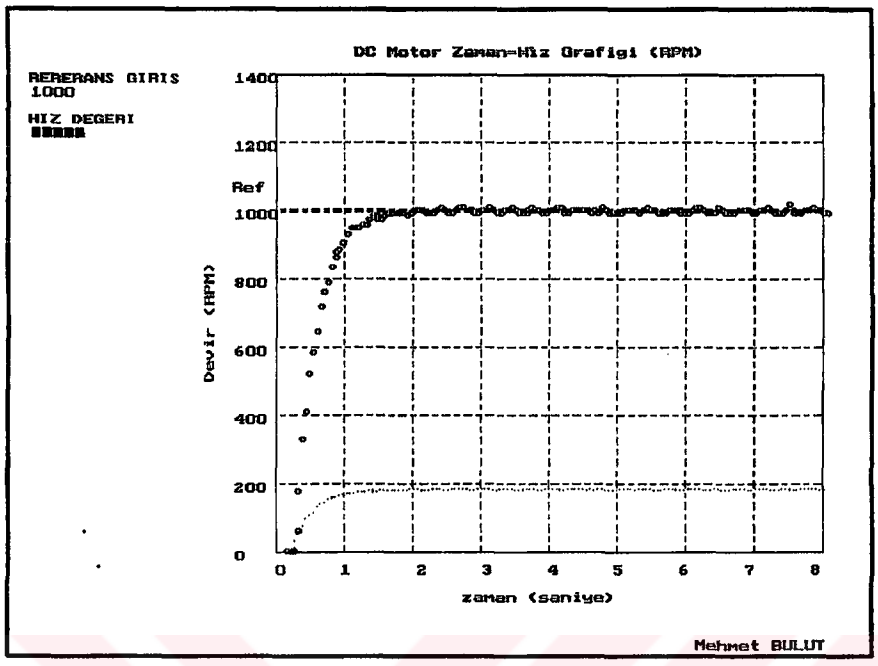
Motor yüklü durumdayken de, öğrenme işleminde yalnız 10 nesil ele alındığından elde edilen kural taban sistemi yeterli derecede temsil etmekten uzak kalmaktadır (Şekil 8.24). Kontrolör verilen referans değerine sistemi kısa zaman da oturtmasına rağmen, kalıcı hal hatası ortaya çıkmakta ve kalıcı hal durumunda salınımlı bir sonuç alınmaktadır. Bu daha yüksek hızlarda, Şekil 8.24-c'de 1000 dev/dak'da görüldüğü gibi salınımlar artmaktadır. Bu nedenle genetik algoritma tabanlı öğrenme algoritması daha yüksek sayıda (30) nesil için çalıştırılarak daha verimli kural taban öğrenilme yoluna gidilmiştir.



(a)



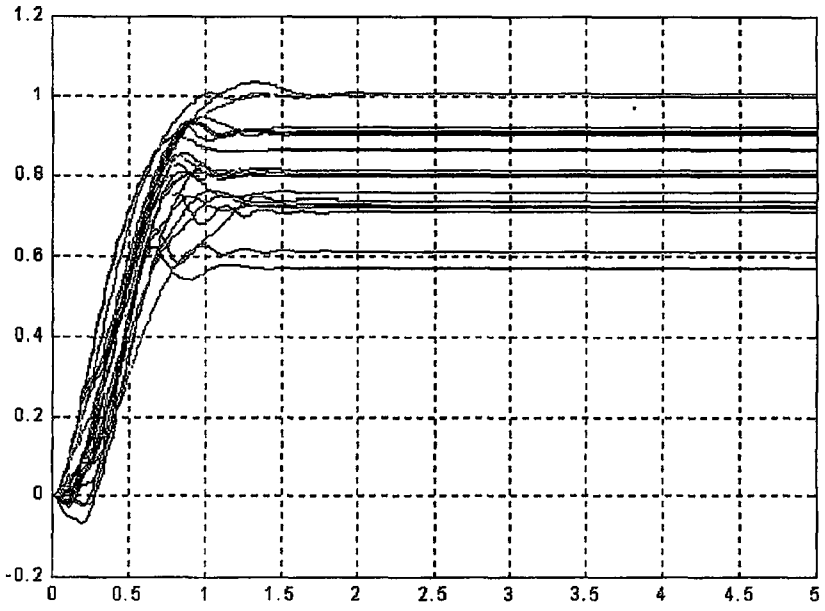
(b)



Şekil 8.24 Sistem yük altındayken 300 dev/dak ve 500 dev/dak ve 1000 dev/dak referans girişleri için elde edilen çıkış cevapları

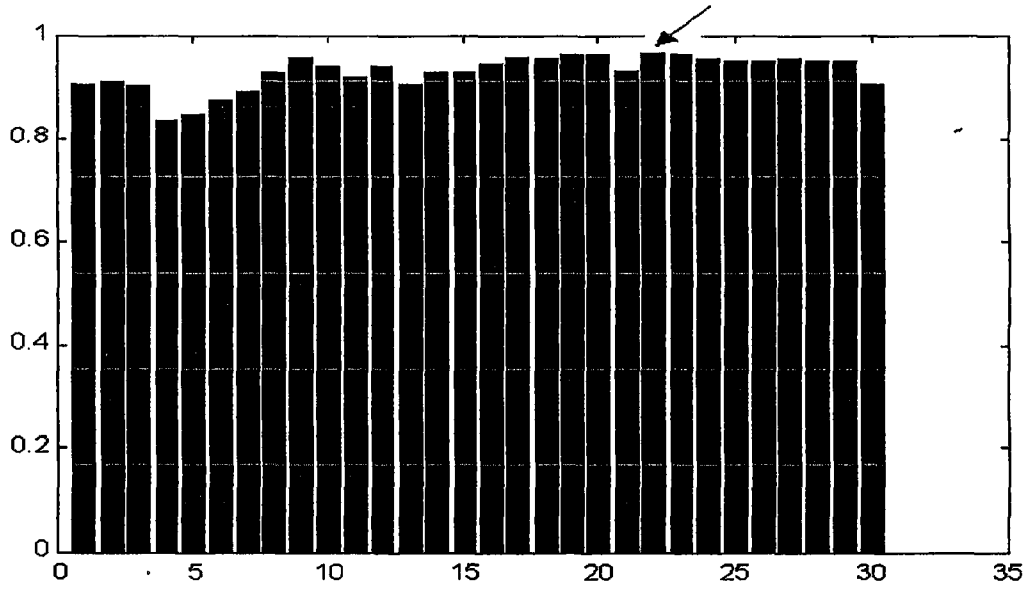
Genetik algoritmaların özelliğinden dolayı , öğrenme algoritmasını her çalıştırılışında farklı değerler elde edilmektedir. Hem başlangıç kural taban havuzunun random olarak oluşturulması, hemde seçme ve mutasyon genetik işlemlerin random yöntemlere dayanması bu sonucu doğurmaktadır. Genetik algoritmalar çözüm uzayında araştırma yapmaktadırlar. Bu yüzden algoritmanın çalıştırıldığı nesil sayısı araştırılacak çözüm noktası sayısı ile orantılı olduğundan çözüm noktasına ulaşmada etkili olmaktadırlar.

GFS algoritması 30 nesil için çalıştırılmış ve elde edilen kural tabanın 10 nesil için elde edilen kural tabana göre daha iyi sonuçlar verdiği görülmektedir. 30 nesil içinde 22 nolu nesilde en yüksek uyumluluk değerli (0.9625) kural taban elde edilmiştir. Öğrenme boyunca herbir nesil için elde edilen yüksek uyumluluk değerli kural tabanların değerleri Çizelge 8.5'da verilmiştir. En yüksek uyumluluk değerlikli kural tabanın FAM tablosu ise Çizelge 8.6'da verilmiştir.



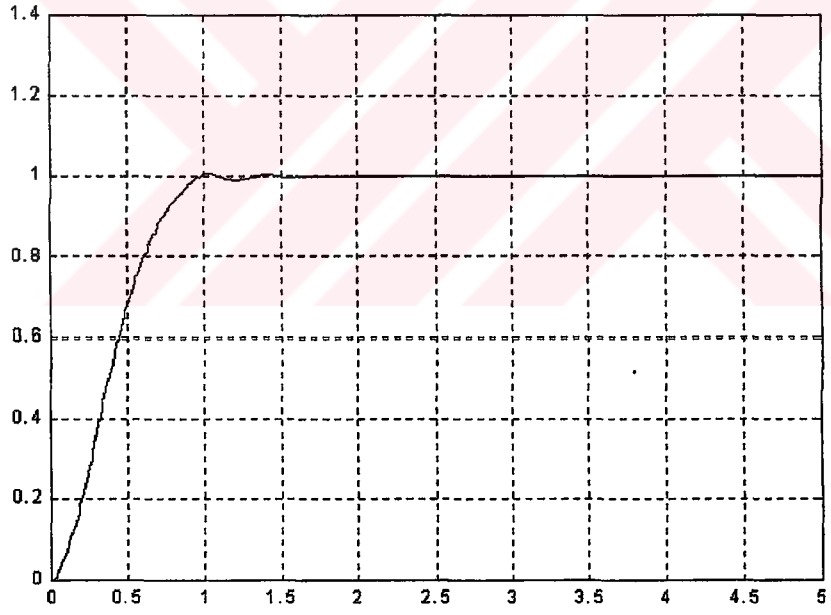
Şekil 8.25 GBS'nın 30 nesil için elde edilen tüm yüksek uyumluluklu kural tabanlar için dc motor hız-zaman eğrisi

Çizelge 8.5 30 nesil içinde en yüksek uyumluluklu kural tabanların uyumluluk değerleri



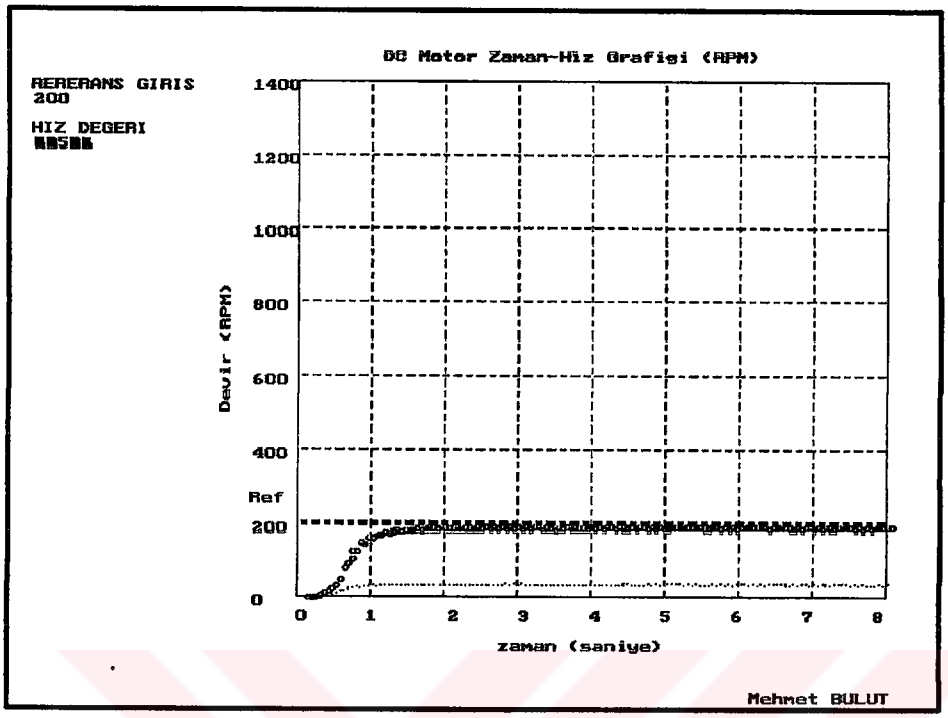
Çizelge 8.6 30 nesil için elde edilen PI-tip bulanık kontrolör kural taban karar tablosu

u		E										
		-1.0	**-0.8**	**-0.6**	**-0.4**	**-0.2**	**0**	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**
eint	**-1.0**	0.70	-0.40	-0.90	-1.00	-1.00	-1.00	1.00	-1.00	-1.00	-1.00	1.00
	-0.8	-1.00	1.00	1.00	1.00	1.00	1.00	-1.00	-1.00	-1.00	-1.00	0.60
	-0.6	-1.00	-1.00	1.00	-1.00	-1.00	1.00	-1.00	1.00	1.00	1.00	0.00
	-0.4	-1.00	-0.30	1.00	1.00	-1.00	-1.00	-1.00	1.00	1.00	1.00	1.00
	-0.2	-1.00	1.00	-1.00	1.00	1.00	1.00	-1.00	-1.00	-1.00	1.00	-1.00
	0	1.00	-1.00	0.70	1.00	1.00	1.00	-1.00	-1.00	1.00	1.00	1.00
	0.2	1.00	0.00	-1.00	-1.00	-1.00	0.20	-1.00	-1.00	1.00	-1.00	1.00
	0.4	-1.00	1.00	1.00	1.00	-1.00	-1.00	1.00	0.00	0.40	1.00	1.00
	0.6	1.00	-1.00	0.20	0.60	-1.00	1.00	1.00	1.00	1.00	1.00	1.00
	0.8	-1.00	-1.00	-1.00	1.00	-1.00	-1.00	1.00	1.00	-1.00	1.00	1.00
	1.0	1.00	-1.00	1.00	-1.00	1.00	0.80	1.00	0.30	1.00	1.00	1.00

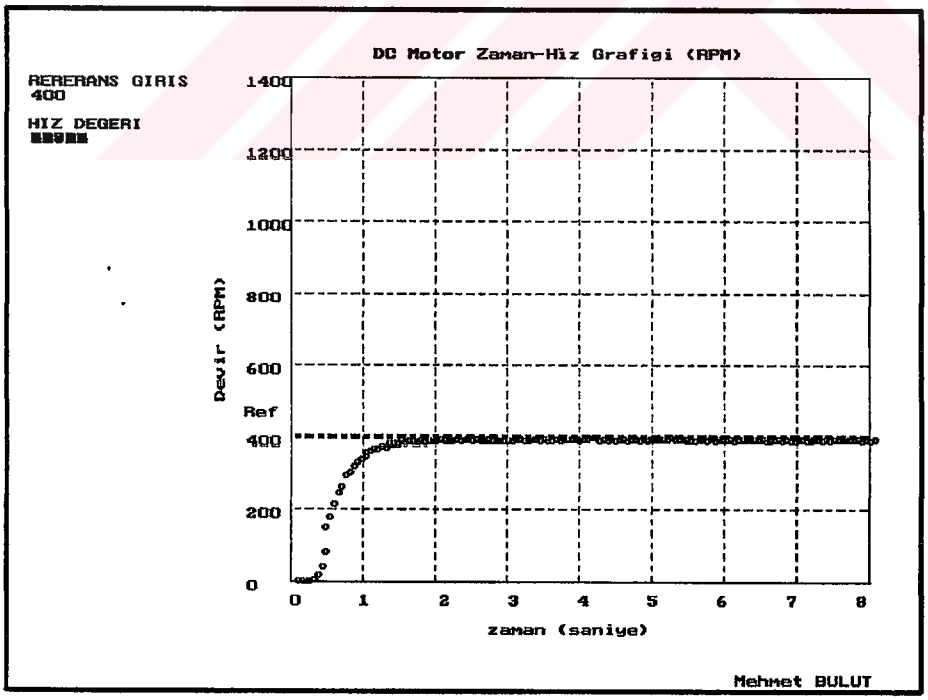


Şekil 8.26 30 nesil için elde edilen PI-tip bulanık kontrolör birim basamak cevabı

Genetik Bulanık Sistemi ile 30 nesilde elde edilen en yüksek uyumluluk değerine sahip kural tabanlı PI-tip bulanık kontrolör sistemde yük var iken çeşitli referans giriş değerleri ile çalıştırılmış elde edilen sonuçlar Şekil 8.27-29'de verilmiştir. Sistemde yük varken 300 dev/dak ve 500 dev/dak ve 1000 dev/dak referans giriş değerleri uygulandığında; Tr <1 sn'nın altında artma zamanı ile referans değere ulaştığı ve Ts=1.2 sn'de referansa oturduğu görülmektedir.

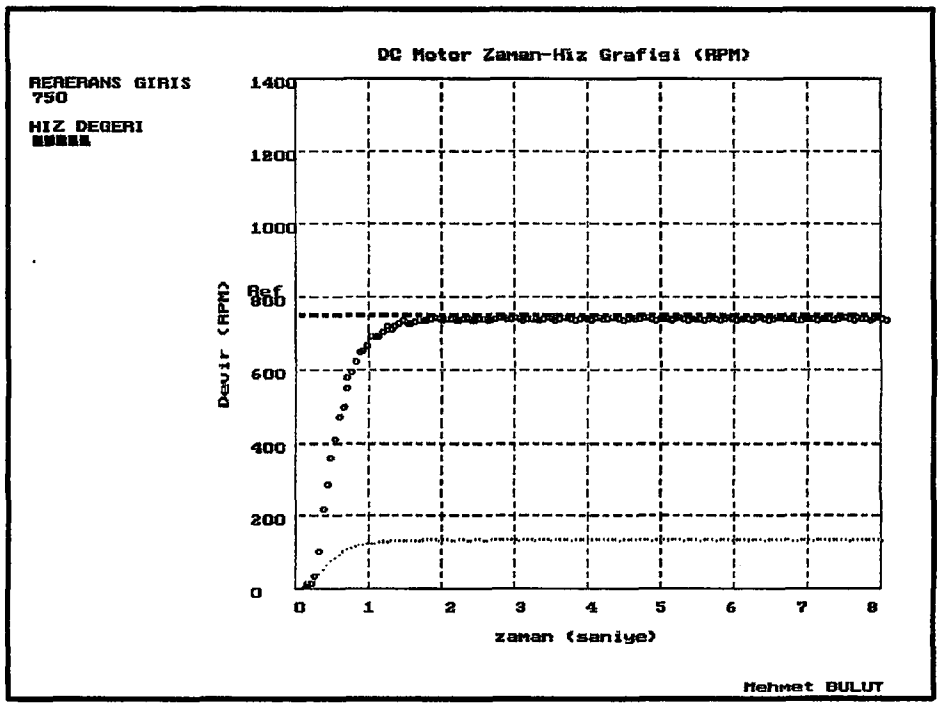


(a)

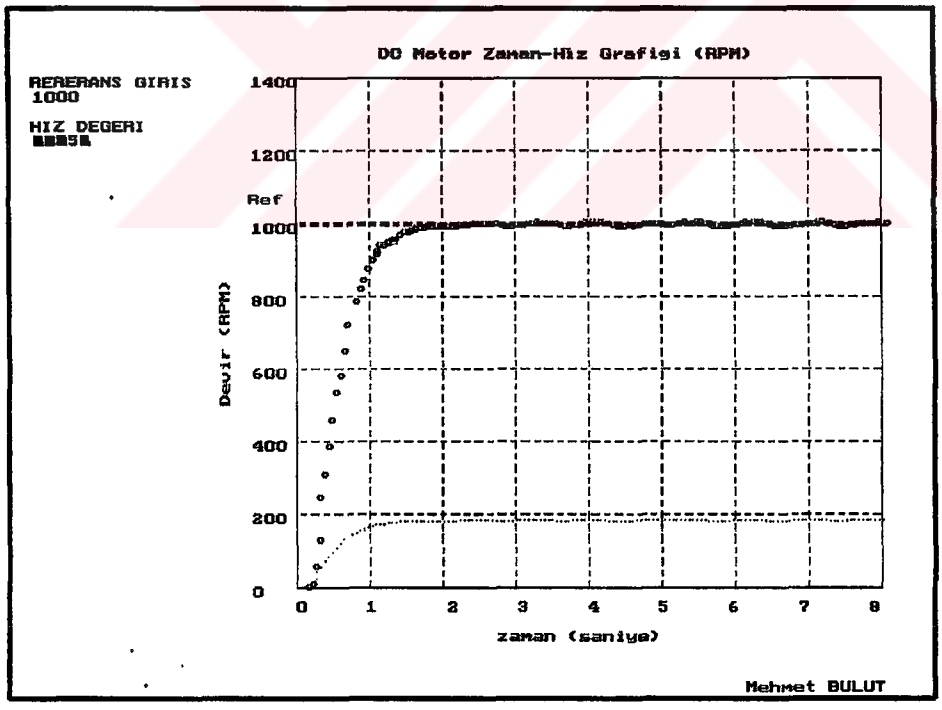


(b)

Şekil 8.27 Sistem yük altındayken a) 200 dev/dak , b) 400 dev/dak referans girişi için hız-zaman eğrisi

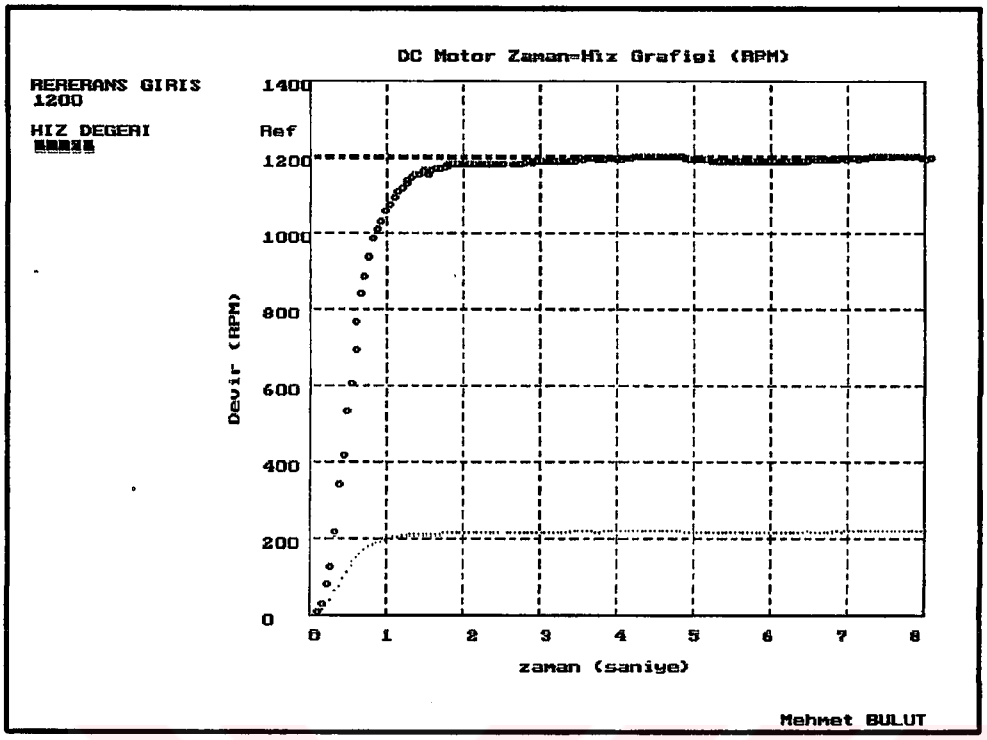


(a)

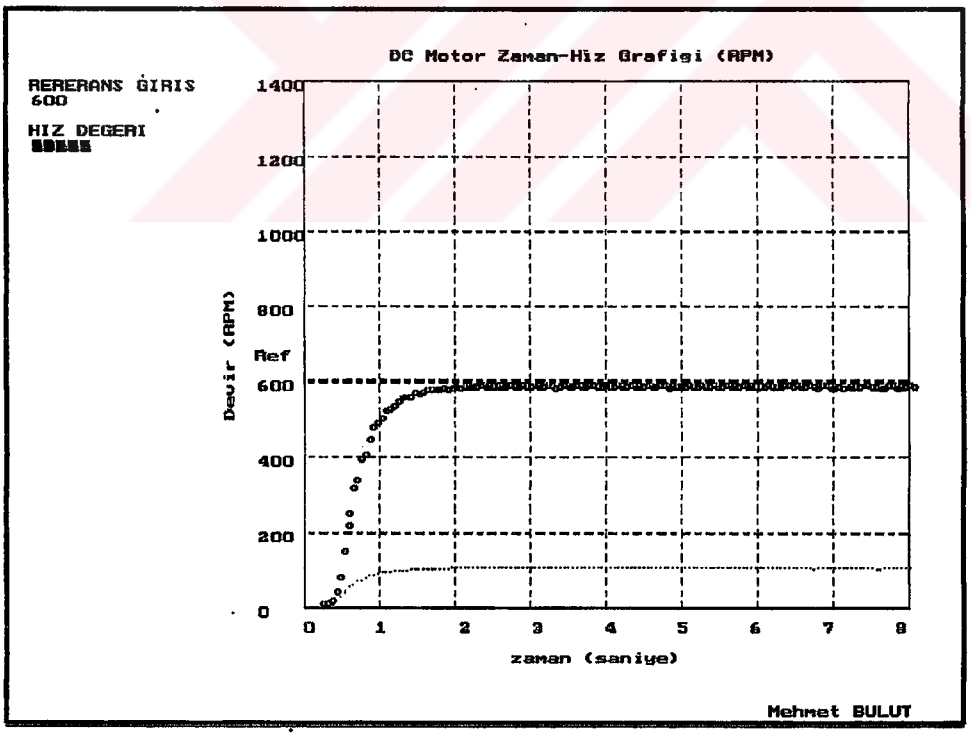


(b)

Şekil 8.28 Sistem yük altındayken a) 750 dev/dak, b) 1000 dev/dak referans girişi için hız-zaman eğrisi



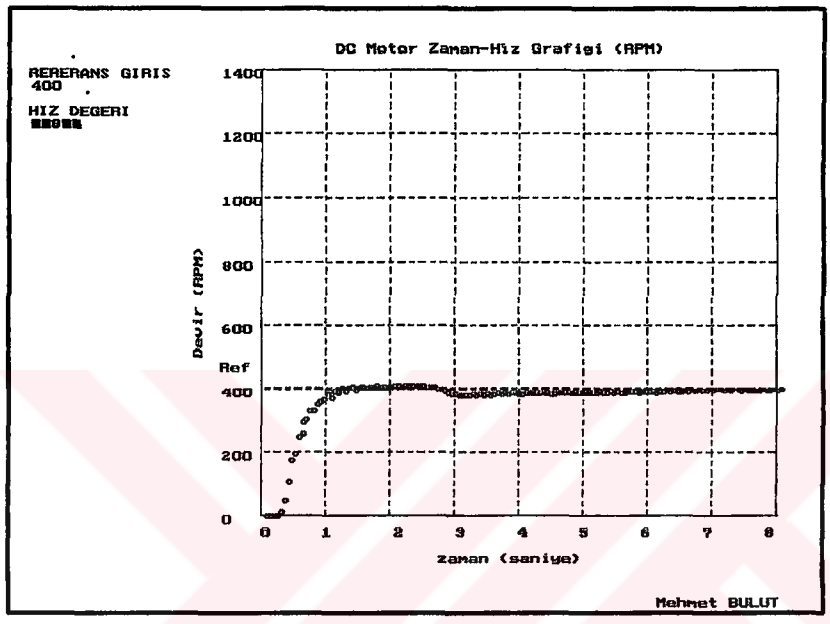
(a)



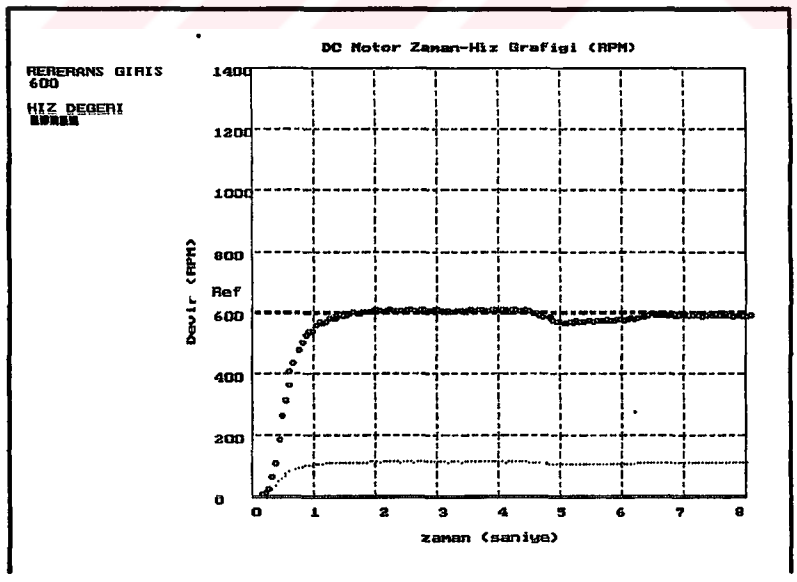
(b)

Şekil 8.29 Sistem yük altındayken 6a) 1200 dev/dak , b) 600 dev/dak referans girişi için hız-zaman eğrisi

Ayrıca sisteme yük bağlı iken çeşitli hızlarda, dönen DC motor miline bir tahta baskı yapılmak suretiyle dışardan bozucu etkisi verilerek tasarlanan kontrol sisteminin etkisi gözlenmiştir. Şekil 8.30-31'de verilen sonuçlardan görüleceği üzere bozucu etki azaltılmaya çalışılmakta ve motor gücünün kaldırabileceği büyüklükteki bozucuların etkisini kısa zamanda ortadan kaldırmaktadır.

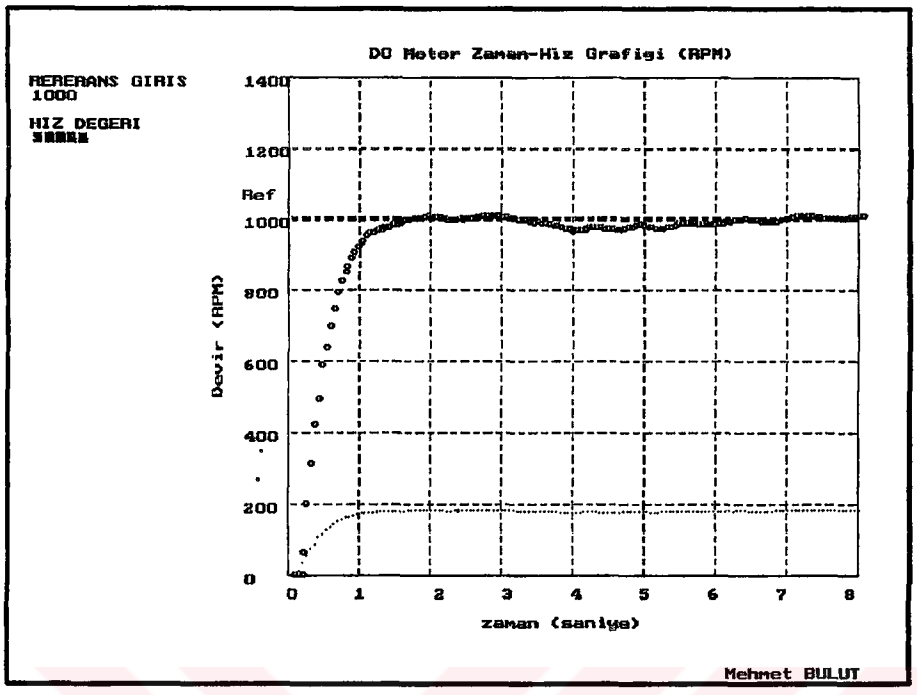


(a)

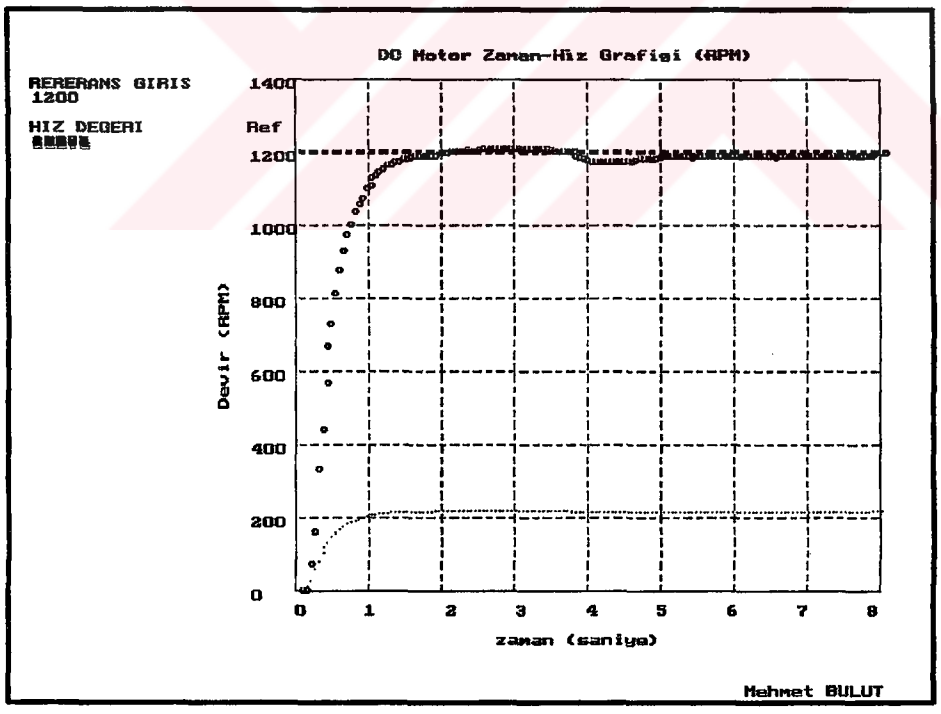


(b)

Şekil 8.30 Sistem yük altındayken a) 400 dev/dak , b) 600 dev/dak referans girişinde bozucunun hız-zaman eğrisindeki etkisi



(a)



(b)

Şekil 8.31 Sistem yük altındayken a) 1000 dev/dak , b) 1200 dev/dak referans girişinde bozucunun hız-zaman eğrisindeki etkisi

9. SONUÇ VE ÖNERİLER

9.1 Sonuçlar

Genetik algoritmaların bulanık kontrolörlerin kural tabanı tasarımında kullanılması ilk olarak pH kontrolü, inverted pendulum gibi uygulamalarda ortaya çıkarılmıştır (Lee and Takagi, 1994 ve Karr and Gentry, 1994). Park ve arkadaşları doğru akım seri motoru kontrol etmek için genetik algoritmaları ile bulanık yaklaşım modelini optimize etmeye çalışmışlardır (Park., 1996). Önceden yapılan çalışmalar daha çok bulanık kontrolörün kural sayısını azaltmak veya bulanık kontrolör sistem parametrelerini iyileştirmişlerdir. Bu çalışmada, literatürdeki bunlar gibi benzeri çalışmalar baz alınarak bulanık kontrolör ile sabit mıknatıslı DC motorun hızı kontrol edilmiştir. Bu bulanık kontrolörün kural tabanı, genetik algoritmalar temelli bir öğrenme algoritması geliştirilip kullanılmasıyla elde edilmiştir.

Bu tezde kontrol edilen sistem, mekanik olarak birbirine bağlanmış takometresi üzerinde bir dc motor ve yük olarak kullanılan dc motor-jeneratör gurubundan oluşmaktadır. Pentium-120 Mhz bir bilgisayarın paralel portunu iki yönlü veri iletişiminde (8-bit) kullanacak arabirim devre dizayn edilerek bilgisayar üzerinden serbest uyarmalı DC motorunun hız kontrolü gerçekleştirilmiştir. Bu amaçla, bir PC üzerinde bulanık kontrol programı C programlama diliyle yazılmıştır. Kontrol sisteminin başlangıç anında referans hız bilgisi bilgisayardaki program üzerinden girilmektedir. Çalışma başladıktan sonra , bu referans değer parallel porta bağlı sürme devresi üzerinden DC motora uygulanmaktadır. Hız sensörü ile DC motordan alınan hız bilgisi tekrar paralel portan okunarak bilgisayardaki kontrol programında değerlendirilerek motora uygulanacak yeni kontrol sinyali üretilmektedir. Bu işlerin yapılmasında olduğu gibi dijital kontrol sistemlerinde dönüştürücüler büyük önem taşımaktadır. Burada 8 bit 'lik bir ADC ve DAC kullanılmıştır.

İlk olarak kontrol edilecek DC motorun üretici firması olan Sinano Co. (Japan) ile temasa geçilerek motor parametreleri elde edilmiş ve bu parametreler kullanılarak matematiksel model çıkarılmıştır. DC motoru kontrolünde kullanılacak PI-tip Bulanık kontrolör için kural tabanını otomatik olarak öğretecek bir Genetik Bulanık Sistem (GBS) Algoritması tasarlanmıştır. Tasarlanan algoritmadan elde edilen veriler motor modeli üzerinde denenerek sonuçları gözlenmiş ve uygulanabilirliği olan yüksek performanslı kural taban DC motorun PI-tip Bulanık kontrolüne uygulanmıştır.

Deneme çalışmalarında 825 dev/dak ve 550 dev/dak referans değerleri kullanılmıştır. Hız bilgisi program çalıştırıldıktan sonra komut şeklinde girilmekte ve kontrol algoritmasının çalışması ile birlikte motordan gelen hız bilgileri Turbo-C dillinde yazılan programın grafik akranında zaman-devir eğrisi şeklinde çizilmektedir. Buradan online motor karakteristiği grafiksel olarak gözlenmesi sağlanmıştır. Aynı zamanda yapılacak hesaplama ve diğer parametre çizimleri için hız bilgisi, kontrol giriş sinyali bilgisi, hata değeri, hatanın integrali değeri bir text dosyasına kaydedilmektedir. Bu haberleşme işlemi paralel port üzerinden gerçekleştirilmektedir. Hız sensöründen gelen frekans bilgisi, bir frekans–gerilim dönüştürücü vasıtasıyla gerilime dönüştürülmekte sonra 8-bit ADC ile sayısallaştırılıp program vasıtasıyla paralel port üzerinden PC ortamına aktarılıp programda kontrol algoritmasında işlenmektedir. Motor maksimum hızı [0-255] arasında ölçeklendirilmiştir. Program üzerindeki bulanık kontrolörün ürettiği 0-255 arasındaki kontrol işareti paralel port üzerinden DAC vasıtasıyla sürme devresine verilmekte ve kontrol işaretiyle orantılı olarak motor hızı sağlanmaktadır.

Yapılan bu çalışmada, tasarlanan PI-tip bulanık kontrolör ile FMRLC algoritması ile öğretilen PD-tip bulanık kontrolör ve manuel olarak geliştirilen bir Bulanık mantık kontrolör (FLC)ün kontrol ettiği dc motor ve jeneratörden oluşan sistemin, kontrol edilen DC motor modelinin istenilen referans değerine ulaşması esnasındaki performans kriterlerine bakılmıştır. Kontrol yöntemleri karşılaştırılırken gözönüne alınan parametreler; sistemin oturma zamanı (Ts), yükselme zamanı (Tr) ve sistemin yaptığı aşım miktarıdır. Bölüm 8'de Çizelge 8.3'te verilen sonuçlardan görüleceği üzere tasarlanan yönteme ait Tr, ve Ts zamanları daha uygundur.

Geliştirilen GBS öğrenme algoritmasının 10 nesil ve 30 nesil için çalıştırılmasıyla elde edilen en yüksek uyumluluk değerli PI-tip Bulanık Kontrolöre kural tabanların PC temelli DC motor sistemine uygulanmıştır. GFS algoritmasının 10 nesil çalıştırılmasıyla elde edilen kural taban kullanılarak oluşturulan Bulanık-PI kontrolör ile DC motor sistemine yük yokken uygulanmasıyla alınan sonuçlara bakıldığında 550 dev/dak ve 825 dev/dak hız zaman eğrilerinde transient bölgesinde yüksek bir salınımın olduğu kalıcı hal durumunda bunun düşük bir şekilde devam ettiği görülmektedir. 1375 dev/dak' nın hız –zaman eğrisinde de cevabın kısmen salınımlı olduğu görülmektedir.

Motor yüklü durumdayken de, öğrenme işleminde yalnız 10 nesil ele alındığından elde edilen kural taban sistemi yeterli derecede temsil etmekten uzak kalmaktadır. Kontrolör verilen referans değerine sistemi kısa zaman da oturtmasına rağmen, kalıcı hal hatası ortaya çıkmakta

ve kalıcı hal durumunda salınımlı bir sonuç alınmaktadır. Bu daha yüksek hızlarda, Şekil 8.25-c'de 1000 dev/dak'da görüldüğü gibi salınımlar artmaktadır.Bu nedenle 10 nesil için elde edilen kural tabanın sistemi yeterli şekilde temsil etmediği düşünülmektedir. Bu nedenle tasarlanan öğrenme algoritması 30 için çalıştırılarak elde edilen kural taban kullanılmıştır. Bununla ilgili sonuçlar Bölüm 8'de verilmektedir. Elde edilen sonuçlardan, hedeflenen ve gerçekleştirilen genetik bulanık sistem algoritması, Bulanık kontrolör için dc motora uygulanabilir, performansı yüksek, esnek bir yapı ortaya çıkarmıştır. Sonuç olarak, bu tezde tasarlanan yaklaşım DC motor kontrolünde düşük kalıcı hal hatasına sahip, düşük yerleşme zamanına sahip sistem cevapları sağlayan bir Bulanık kontrolör dizaynını sağlamaktadır.

Bu çalışmada öğrenme ile elde edilen kural tabanlar, algoritmanın yapısından dolayı her çalıştırıldığında farklı değerler elde edilmektedir. Başlangıç havuzunun random olarak oluşturulmasından kaynaklanmaktadır. Burada ön bilgi olarak kural sayısı, ve giriş parametrelerine ait üyelik fonksiyonları ve tanım uzayı önceden belirlenmiş olarak verilmektedir. Kural tabanın FAM tablosu ise algoritma ile öğrenilmektedir. Bu yüzden ilk havuzun çözüm noktasına yakınlığı sonuca ulaşma zamanını etkilemektedir.

9.2 Öneriler

Bu çalışma ile tasarlanan Genetik Bulanık sistem başarıyla gerçek DC motor sistemine uygulanarak olumlu sonuçlar alınmıştır. Tasarlanan sistem algoritması Turbo C diliyle Pc üzerinde çalıştırıldığında algoritmada ilaveler değişikler kolayca yapılabilmektedir. Bu da sisteme esnek bir yapı kazandırmaktadır. İleriki aşamalarda, bu sistem kullanılarak serbest uyarmalı DC motorunun kontrolünde değişik kontrol algoritmaları da sadece yazılımı değiştirerek kolayca denenebilir.

Bunun neticesinde kontrol sisteminin yazılımında yüksek seviyeli dillerden C programlama dilinin kullanılmasıyla büyük kolaylık sağlanmıştır. Böylece gerekli olduğu hallerde donanımda küçük bir değişiklik yaparak, gerçekleştirilen kontrol sistemiyle değişik motorların veya sistemlerin kontrolü kolaylıkla gerçekleştirilebilir.

KAYNAKLAR

Atmaca H. ve Bulut M., (1995), "Bulanık Mantık Denetleyiciler Ve Uygulamaları" , Kara Harp Okulu Öğretim Başkanlığı, 1. Sistem Mühendisliği ve Savunma Uygulamaları, Ekim 1995, Ankara.

Bose, Bimal K. , (1994) "Expert system, Fuzzy Logic and Neural Networks Applications in Power Electronics and Motion Control", Proceedings of The IEEE, vol 82. No.8, 1303-1324.

Bulut M., Cansever G., Ustun S., (2000), "Fuzzy Model-Based Learning for a DC Motor Controller", Int. Conf. on signal Process. Appl. And Tech. , Dallas, TX, October 2000, USA.

Chwee N. K, Li Y., "Reduced Rule-base and Direct Implementatiton of Fuzzy Logic Control Systems", University og Glasgow, United Kingdom ,www.gla.ac.uk

Chin T.C., Qi X.M., (1998), "Genetic Algorithms for learning the rule Base of Fuzzy Logic Controller", Fuzzy Sets and Systems, vol.97, pp. 1-7.

Chung I-F., Lin C-J., Lin C-T., (2000), "A GA-based fuzzy adaptive learning control network", Fuzzy Sets and Systems 112 (2000) pp. 065-84.

Donescu V., Neacsu D.A., Griva G., (1996), "Design of a Fuzzy Logic Speed Controller for Brushless DC Motor Drivers", IEEE Control Systems App., pp.404-408.

Dubey, Gopal K., ve Kasarabada, C. Rao , (1993) "Power Electronics and Drivers", Tata McGraw-Hill Pub, 261-281, New Delhi, India.

Goldberg, D.E.,(1989), "Genetic Algorithms in Search, Optimization and Machine Learning" Addison-Wesley, Reading, MA, 1989.

Gürrocak, H.B.,(1999), "A Genetic-algorithm-based Method for Tuning Fuzzy Logic Controllers" ,Fuzzy Sets and Systems 108 (1999) 39-47.

Ha Q.P., (1996), "Proportional-Integral Controllers with Fuzzy Tuning", Electronics Letters, vol. 32, No. 11, 23rd may 1996.

Herrara, F., Lozano, M., Verdegay, J.L.,(1995) "Generating Fuzy Rules from Examples Using Genetic Algorithms", Department of Computer Science and Artificial Intelligence. University of Granada, 1995, Spain.

Homaifar A., McCormick E., (1995), "Simultaneous Design of Membership Functions and Rule sets of fuzzy Controller Using Genetic Algorithms", IEEE Trans. on Fuzzy Systems, vol. 3, no.2, 1995, pp. 129-139

Johansen T.A., (1994), "Fuzzy Model Based Control: Stability, Robustness, and, performance Issues", IEEE Trans. On Fuzzy Systems, Vol.2, No.3,pp. 221-234.

Karr, C. and Gentry, E., (1993) , Fuzzy control of pH using genetic algorithms, IEEE Trans. Fuzzy Systems 1, 46-53.

King P.J. and Mamdani E.H., (1977), "The Applications of Fuzzy Control Systems to Industrial Processes", Automatica, vol.3, pp.235-242

Kruse,R. ve Nauck,D., (1995), " Learning Methods For Fuzzy Systems", Proc. 3rd German GI-Workshop Neuro-Fuzzy Systems , Nov. 1995, Germany

Kwong W.A., Passino K.M., (1996),"Dynamically Focused Fuzzy Learning Control", IEEE Trans. on Systems, Man, and Cybernetics, Vol. 26, No. 1, pp. 53-74, Feb. 1996.

Layne J.R., Passino K.M., (1993)."Fuzzy Model Reference Learning Control for Cargo Ship Steering", IEEE Control Systems Magazine, Vol. 13, No. 6, pp. 23-34.

Layne J.R., Passino K.M., Yurkovich S., (1993), "Fuzzy Learning Control for Anti-Skid Braking Systems", IEEE Trans. on Control Systems Technology, Vol. 1, No. 2, pp. 122-129.

Leitch D. and Robert P.J., (1998), "New Techniques for Development of a Class of Fuzyy Controllers", IEEE Trans. on Syst. Man and Cyber., Vol.28, no.1, pp.112-123.

Lee, M.A. and Takagi, H., (1993), "Integrating design stages of fuzzy systems using genetic algorithms", Proc. IEEE Internat. Conf. on Fuzzy Systems, 612 617.

Man K.F. , Tang K.S. , Kwong S. and Halay W.A. (1997), "Genetic Algorithms for Control and Signal Processing", Springer-Verleg London,1997, Great Britain.

Maiocchi G., (1995), "Driving Dc Motors", Application Note, SGS-THOMSON Microelectronics.

Moudgal V.G., Kwong W.A., Passino K.M., and Yurkovich S.(1995), "Fuzzy Learning Control for a Flexible-Link Robot", IEEE Transactions on Fuzzy Systems, Vol. 3, No. 2, pp. 199-210, May 1995.

Ng K.C., Li Y., (1994), "Design of Sophisticated of Fuzzy Logic Controller Using genetic Algorithms" In Proc. 3rd Intr. Conf. on Fuzzy Syst., Orlando, FL, June, vol. 3, pp.1708-1712.

Ng K.C., Li Y.,Murray D.J. and Sharman K.C (1995), "Genetic Algorithms Applied to Fuzzy Sliding Mode Controller Design", University og Glasgow, Scotland, United Kingdom,

Park D. , Kandel A. and Langholz, G.,(1994), "Genetic-based new fuzzy reasoning models with application to fuzzy control", IEEE Trans. Systems Man Cybernetics. 24, 1994.

Roach, H. P. ve Anderson, P. H., (1996), "Control of Speed and Direction of a DC Motor Using Power FETs", Department of Electrical Engineering, Morgan State University, Baltimore.

Ross, Timothy J.,(1995), "Fuzzy Logic with Engineering Applications", McGraw-Hill Inc.,469-514, 1995.

Rubaai A., Ricketts D., Dankam M.D., (2000), "Experimental Evaluation of Fuzzy Logic Based Controller for High Performance Brushless DC Motor Drivers", IEEE, pp. 1299-1305.

Sarıoğlu, M. K., (1991), "Otomatik Kontrol I-II", İTÜ Yayınları, 1991, İstanbul.

Shieh M.Y and Li T.H.S.,(1998), "Design and Implementation of Integrated fuzzy Logic Controllerfor a Servomotor System", Mechatronics, vol. 8, pp. 217-240.

Shing J. ve Jang, R.,(1992), "Self-Learning Fuzzy Controllers Based Temporal Back Propagation", IEEE Trans. on Neural Networks, vol.3, no. 5.

Soliman, H.F., Sharaf A.M., Kandil S.A, (1994), "A Tunable Fuzzy Controller For Chopper-Fed Separately Excited DC Motor Drives", IEEE, p.821-824

Sugeno M., (1985) "Industrial Applications of Fuzzy Control ", Elsevier Science Pub., 1985, USA.

Surmann H., (1996), " Genetic Optimization of a Fuzzy System for Charging Batteries", IEEE Trans. On Indutrial Electronics, Vol.43, No.5, pp.541-548.

STMicroelectronics , "An Introduction To Sensorless Brushless Dc Motor Drive Applications With The St72141" , AN1130, Application Note, 2000.

Tang K.S., Man K.F., Kwong S. and He Q., (1996), "Genetic Algorithms and Their Applications", IEEE Signal Processing Magazine, pp.23-37, November 1997.

Tremaine, B. P.,(1992), "Comparison of Fuzzy and Classical Control: Applied to Disc Drive Spindle Servos", Seagate Technology, Scotts Valley, 1992 ,CAY.

Rahman S., Ullah Z. and Neely W. S., (1994), "DC Motor Controller Design with NeuFuz", Application Note 958, National Semiconductor, August 1994

VanDoren V.,(1998), "Ziegler-Nichols Methods Facilitate Loop Tuning", September, 1998 Control Engineering International.

VanDoren V.,(1998), "Basics of Proportional-Integral-Derivative Control", March, 1998, Control Engineering International.

Varsek A., Urbancic T. and filipic B., (1993), "Genetic Algorithms in Controller Design and Tuning", IEEE Trans. On Syst. And Cyber., vol. 23, no.5, pp.1330-1339

Velasco J., Magdelena L., (1995), "Genetic Algorithms in Fuzzy Control Systms", John Wiley, vol.30, 12-1.

Wang, Li X., (1992), "Adaptive Fuzzy Systems and Control Desing and Stability Analysis", University of California at Berkeley, Printice Hall Pub, 1992, New Jersey, USA.

Wang Y.G. and Shao H. H., (2000), "Optimal tuning for PI controller", Automatica, vol. 36, pp. 147-152.

Zhang W.,(1992), "Servo Motor Force Control" , Aptronix Inc. Pub.

EKLER

EK 1 : DC Motor İcin Fuzzy Kontrol Sistemi

```
#include <math.h>
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <graphics.h>
#define MAX(A,B)  ((A) > (B) ? (A) : (B))
#define MIN(A,B)  ((A) < (B) ? (A) : (B))
// fonksiyon tanımları
void ec_centers(double *ce, double ge, double *cc, double gc);
void Initial();
int  plotgrafik(int r, float time, float y,double ge, double gc,double Kb);
void dosyaoku( char filename[],double fuzzyrules[][11],double gu, double gf);
// Ana program
void main(void)
{
  clock_t  start, end;
  int gdriver = DETECT, gmode, errorcode;
  int port=0x378;                    // Çıkış port adresi
  double  eold=0.0, yold=0.0;        // e ve eint için başlangıç değerleri
  int     nume=11, numc=11;          // e için üyelik fonk. Sayısı
                                     // eint için üyelik fonk. sayısı
  int     deger;
  // Ölçekleme katsayıları
    double   ge=0.004, gc=0.004, gu=127;
    double Kb=0.9;
// we üçgen giriş üyelik fonksiyonun yarım taban genişliği
// wc üçgen giriş üyelik fonksiyonun yarım taban genişliği
    double   we=0.2*(1.0/ge);
    double   wc=0.2*(1.0/gc);
// Base width of output membership fuctions of the fuzzy controller
   double   base=0.4*gu;
   double   ce[11], cc[11]; /     / Üyelik fonksiyonları taban merkez noktaları
```

```
// fuzzy controller icin kullanılan  kural taban matrisi dosyası
   double   gf=1;
   double   fuzzyrules[11][11];
   char filename[12]="sn30.dat";
 // Modelde kullanılan proses değerleri
   double  t=0.0;
   float   time,timeold,adim;
   int     i,j,k,l,r;
   float   u,y,ynew,ref;
   double  e=0.0,c=0.0,ctop=0.0;
   // Kaydedilen değerler y : sistem (motor) çıkışı
   //                  r : referans giriş
   //                  u : Kontrolör çıkışı
    double  num, den, prem;
    int     e_count , c_count, e_int, c_int;
    double  mfe[11], mfc[11];
    FILE    *myfile;   //--------Değişken tanımlaması bitti----------
  clrscr();    Initial();  //menu programı
  ec_centers(ce,ge,cc,gc);                // e ve eint için mfs  merkezlerinin ayarlanması
  dosyaoku(filename,fuzzyrules,gu,gf);   // Dosyadan kural taban matrisinin okunması

// Başlangıç değerlerinin kontrolü
printf("ce=[ ");
 for (i=0; i<=10; i++)
     printf("%f  ", ce[i]);
 printf(" ]\n");
 printf("cc=[ ");
 for (i=0; i<=10; i++)
   printf("%f  ",cc[i]);
printf(" ]\n");
printf("gu=%f\n", gu);
getch();

    //------- Kural tabanı ----------------------
printf("fuzzyrules=\n");
```

```
for (i=0; i<=10; i++)
{
 for (j=0; j<=9; j++)
    printf("%f  ", fuzzyrules[i][j]);
 printf("%f\n",fuzzyrules[i][10]);
 }
getch();
 // Burada bulanık kontrolör ile sistem kontrolü yapılıyor
 myfile=fopen("sn30out.dat","w");  // Verileri saklamak için dosya açılıyor

    clrscr();
    puts("=========================================================");
    puts("       DC MOTOR SİSTEMİNİN BULANIK-PI KONTROLÜ        ");
    puts("=========================================================");
    gotoxy(10,12);cprintf("Referans Giris:    ");
    gotoxy(26,12);scanf("%d",&r);
    re f= floor( r / 5.49); // Girilen değer 0-255 arasında ölçekleniyor
    outportb(port,ref); // paralel portun çıkışına veriliyor
  // geribesleme katsayısının belirlenmesi
   if  (ref<=40 ) Kb=0.25;
   else if (ref>40 & ref<=60 ) Kb=0.39;
    else if( ref>60 & ref<=80 ) Kb=0.52;
     else if ( ref>80 & ref<=100 )  Kb=0.62;
      else if ( ref>100 & ref<=120 )  Kb=0.74;
       else if ( ref>120 & ref<=140 )  Kb=0.87;
        else if ( ref>140 & ref<=160 )  Kb=0.98;
         else if ( ref>160 & ref<=180 )  Kb=1.22;
          else if ( ref>180 & ref<=200 )  Kb=1.36;
           else if ( ref>200 & ref<=220 )  Kb=1.47;
            else if ( ref>220 & ref<=240 )  Kb=1.6;
             else Kb=1.7;



 // Kontrol döngü baslatılıyor
  start = clock(); //saat başlatılıyor
```

```
time=0.0;   timeold=0.0;
// kontrol algoritmasi dongusu basliyor
initgraph(&gdriver, &gmode, "");
errorcode = graphresult();
if (errorcode != grOk)
{
  printf("Graphics error: %s\n", grapherrormsg(errorcode));
  printf("Press any key to halt:");
  getch(); exit(1);
}
// Ana kontrol döngüsü başlangıcı
 while (!kbhit())
 {   //delay(10);  // örnekleme frekansı
    end = clock();
   //printf("Gecen Zaman: %f\n", (end - start) / CLK_TCK);
   time= (end - start) / CLK_TCK;
   // printf("zaman: %f\n",time);
     // Paralel portun 0x379 adresinden bilgi okunuyor.
     deger=(inport(0x379));  deger=(deger&0x07fc)>>3;    // Kaydırma işlemi,
      y= Kb * deger;  // Geri besleme katsayısı
      plotgrafik(r,time,y,ge,gc,Kb);  // okunan değerin grafik ekranına girilmesi
       if  (time>=8) goto son;
       // e ve eint için sıfır olmayan üyelik fonk sayısının hesaplnması
       c_count=0; e_count=0;
       e = ref - y;  // fuzzy controller için hata giriş değerinin hesaplanması
       adim=time-timeold;
      ctop = ctop + (e - eold) * adim;          //  hatanın integralinin hesaplanması
      if(ctop>(255*gc))
   c=255*gc;
   else if(ctop<-255*gc)
    c=-255*gc;
     else
      c=ctop;
// Üyelik fonksiyonlarında sol tarafta saturasyon durumu
if (e<=ce[0])
```

```
    {   mfe[0]=1.0;
        for (i=1; i<=10; i++)      mfe[i]=0.0;
        e_count=e_count+1;  e_int=0;
    }
    else
     {  if (e>=ce[nume-1])
   // Üyelik fonksiyonlarında sağ tarafta saturasyon durumu
    for (i=0; i<=9; i++)    mfe[i]=0.0;
    mfe[10]=1.0;
  e_count=e_count+1; e_int=nume-1;
}
   else
   {  for (i=0; i<=nume-1; i++)
      {  if (e<=ce[i])
         { mfe[i]=MAX(0, (1.0+(e-ce[i])/we) );
           if (mfe[i]!=0)
           {   e_count=e_count+1;
               e_int=i;
           }
      }
     else
      { mfe[i]=MAX(0,(1.0+(ce[i]-e)/we) );
        if (mfe[i]!=0)
        { e_count=e_count+1;  e_int=i; }
      }       // end of else
    }         // end of for
  }           // end of else
}             // end of else
    // İslemlerin eint için tekrarlanması
    if (c<=cc[0])
     {  mfc[0]=1.0;
        for (i=1; i<=10; i++)     mfc[i]=0.0;
        c_count=c_count+1;  c_int=0;
     }
    else
```

```
{  if (c>=cc[numc-1])
    { for (i=0; i<=9; i++)      mfc[i]=0.0;
      mfc[10]=1.0;
      c_count=c_count+1; c_int=numc-1;
    }
   else
    {  for (i=0; i<=numc-1; i++)
       {  if (c<=cc[i])
          {  mfc[i]=MAX(0, (1.0+(c-cc[i])/wc) );
             if (mfc[i]!=0)
             { c_count=c_count+1;
               c_int=i;
             }
          }
          else
         {
           mfc[i]=MAX(0,(1.0+(cc[i]-c)/wc) );
           if (mfc[i]!=0)
           { c_count=c_count+1;
             c_int=i;
           }
         }       // end of else
       }         // end of for
    }             // end of else
}                 // end of else
// Burada sisteme uygulanacak crisp değer hesaplanıyor
// Yani durulama işlemi ile fuzzy kontroller çıkışı sisteme
// uygulanabilecek hale getirildi.
  num=0.0;   den=0.0;
   for (k=(e_int-e_count+1); k<=e_int; k++)
   {
     for (l=(c_int-c_count+1); l<=c_int; l++) // Scan over e indices of mfs that are on
     {
     prem=MIN(mfe[k], mfc[l]);    // Scan over c indices of mfs that are on
  // Defuzzification islemi
```

```
        num=num+fuzzyrules[k][l]*base*(prem-(prem*prem)/2.0);
        den=den+base*(prem-(prem*prem)/2.0);
      }
    }
     u=num/den;
// Crisp output of fuzzy controller that is the input to the plant
     eold=e;
     if(u<(-127)) u=-127;
     if(u>127) u=127;
     u=u+127;
/*  gotoxy(10,14);    printf("e:   %3.2f   edot: %3.2f\n ", e, c);
     gotoxy(10,15);  printf("y:   %3.2f   u   : %3.2f\n ", y, u);   */
     time= (end - start) / CLK_TCK;
     printf("zaman: %f",time);*/
     outportb(port,u);
     fprintf(myfile, "%3.2f  %3.2f  %3.2f  %3.2f %3.2f \n",time,y,u,e,c);
     timeold=time;
     //printf("timeold %f\n",timeold);
     t=t+step;
     index=index+1;  // zaman indeksinin artırılması
     // index=0,  t=0 zamanına denke geliyor
}    // while ana döngü sonu
son:
  // motorun frenlenmesi
  u=0;  outportb(0x378,u);
  fclose(myfile);              // veri dosyasının kapatılması
     // fuzzy controllerın kural tabanın dosyaya kaydedilmesi
     myfile=fopen("fuzrules.dat","w");
     for (i=0; i<=10; i++)
     {   for (j=0; j<=9; j++)
           fprintf(myfile,"%f  ", fuzzyrules[i][j]);
           fprintf(myfile,"%f\n",fuzzyrules[i][10]);
     }
    fclose(myfile);   getch();
    closegraph();
```

```
} // Ana Program sonu
// Alt program ve fonksiyonlar
// Giriş üyelik fonksiyonlarının merkezlerinin hesaplanması
void ec_centers(double *ce, double ge, double *cc, double gc)
{  int i;
   for (i=0; i<=10; i++)    ce[i]=(-1.0+0.2*i)*(1.0/ge);
   for (i=0; i<=10; i++)   cc[i]=(-1.0+0.2*i)*(1.0/gc);
}
void Initial()          // baslangıc menusu
{ int secim;
  clrscr();  textcolor(0); extbackground(15);
 gotoxy(12,3);cprintf("-----------------------------------------------------------");
 gotoxy(12,4);cprintf("   BİLGİSAYAR DESTEKLİ DC MOTOR HIZ KONTROLÜ      ");
 gotoxy(12,5);cprintf("-----------------------------------------------------------");
  gotoxy(52,24);printf("            Mehmet BULUT                ");
  textcolor(0);  textbackground(15);
  gotoxy(26,7); cprintf("========================== ");
  gotoxy(26,8); cprintf(" MOTOR REFERANS DEVİR GİRİSİ ");
  gotoxy(26,9); cprintf(" ========================== ");
  gotoxy(26,10);cprintf("   1. Motor Hız Girişi      ");
  gotoxy(26,11);cprintf("   2. Motor Reset           ");
  gotoxy(26,12);cprintf("   3. EXIT                  ");
  gotoxy(26,13);cprintf(" ========================== ");
  gotoxy(26,14);cprintf("      SECIMINIZ      :   ");
  gotoxy(51,14);scanf("%d",&secim);
}
// Elde edilen hız bilgisinini grafik ekranında çizilmesi
int plotgrafik(int r,float time, float y, double ge, double gc,double Kb)
{  int x,d,yy,ref;
   char msg[80];
   int sig = 5;
   setcolor(4);
   //rectangle(8,48,55,68);
   sprintf(msg, " RERERANS GIRIS ");outtextxy(10,50,msg);
   gcvt(r, sig, msg);  outtextxy(20,60,msg);
```

```
sprintf(msg, " HIZ DEGERI ");outtextxy(10,80,msg);  gcvt(y, 4, msg);
outtextxy(20,90,msg);  setcolor(3); settextstyle(0, 0, 1);
sprintf(msg, " Mehmet BULUT ");outtextxy(500,465,msg);
setcolor(3); setlinestyle(0, 1, 3);  rectangle(2,2,636,476);
sprintf(msg,"PARAMETRELER");outtextxy(20,140,msg);
sprintf(msg,"ge :");outtextxy(20,150,msg);  gcvt(ge, sig, msg); outtextxy(50,150,msg);
sprintf(msg,"gc :");outtextxy(20,160,msg);  gcvt(gc, sig, msg); outtextxy(50,160,msg);
sprintf(msg,"Kb :");outtextxy(20,170,msg);  gcvt(Kb, sig, msg); outtextxy(50,170,msg);
setcolor(8);  setlinestyle(2, 1, 3);
ref= (400-r / 4);
line(200,ref,600,ref);
sprintf(msg, " Ref"); outtextxy(160,ref-20,msg);
moveto(200, 50); /* move the C.P. to location (20, 30) */
setlinestyle(0, 2, 1); setcolor(8);
lineto(200, 400); lineto(600, 400);
lineto(600, 50);  lineto(200, 50);  setlinestyle(2, 1, 1);
line(200,100,600,100);  line(200,150,600,150);  line(200,200,600,200);  line(200,250,600,250);
line(200,300,600,300);  line(200,350,600,350);  line(200,50,200,400);line(250,50,250,400);
line(300,50,300,400);line(350,50,350,400);  line(400,50,400,400); line(450,50,450,400);
line(500,50,500,400);line(550,50,550,400);
/* create and output a message at C.P. */
setbkcolor(15);  setcolor(9);  settextstyle(6, 0, 2);
sprintf(msg, " DC Motor Zaman-Hız Grafiği (RPM)");
outtextxy(250,30,msg);  settextstyle(1, 0, 4);
sprintf(msg, " zaman (saniye)");  outtextxy(330,430,msg);
sprintf(msg, "0    1    2    3    4    5    6    7    8");
outtextxy(200,410,msg);  settextstyle(1, 1, 4);
sprintf(msg, "Devir (RPM) ");  outtextxy(155,180,msg);  settextstyle(1, 0, 4);
sprintf(msg, "0"); outtextxy(170,400,msg);  sprintf(msg, "200"); outtextxy(170,350,msg);
sprintf(msg, "400"); outtextxy(170,300,msg);  sprintf(msg, "600"); outtextxy(170,250,msg);
sprintf(msg, "800"); outtextxy(170,200,msg);  sprintf(msg, "1000"); outtextxy(170,150,msg);
sprintf(msg, "1200"); outtextxy(170,100,msg);  sprintf(msg, "1400"); outtextxy(170,50,msg);

 //if (r>150 & r<200) y=1.4*y;
  x=200+floor(time*100/2);   d=400-floor(y*5.568/4);    yy=400-floor(y/4);
```

```
    if ( (yy<r+r*5/100) & (yy>r-r*5/100))  yy=r;
    //  if (yy>2*r) yy=2*r;
    if ( (d<5.568*(r+r*5/100)) & (d>5.568*(r-r*5/100)) )  d=5.568*ref;
    //      if (d>2*5.41*r) d=2*5.41*r;
    // circle(x, yy, 2);  // setfillstyle(11, 2);
    putpixel(x,d,7);    putpixel(x,yy,6);    setcolor(3);
    sprintf(msg,"ø");  outtextxy(x,d,msg);
  // getch();
  return 0;
}
void dosyaoku(char filename[],double fuzzyrules[][11], double gu, double gf)
{ FILE *dosya;
  int i,j;
  float deg;
  double rules[11][11];
  clrscr();
  dosya = fopen(filename, "r");
  if ((dosya = fopen(filename, "r")) == 0)fprintf(stderr, "Cannot file \n");
  for(i=0;i<=10;i=i+1)
  { for(j=0;j<=10;j=j+1)
    {  fscanf(dosya, "%f ", &deg);
       rules[i][j]=deg;
       printf("%2.2f ", deg);   }
      printf("\n");  }
   fclose(dosya);
   for (i=0;i<=10;i++)
  for (j=0; j<=10; j++)
    fuzzyrules[i][j]=rules[i][j]*gu*gf;
  getch();
}
```

EK 2 :Genetik Bulanık Sistemi Öğrenme Algoritması Matlab Kodu

```
clear
rand('state',0)
NUM_TRAITS=121;              % Herbir ferde ait karakter sayısı
SIG=ones(121);
HIGHTRAIT=SIG(1,:);          % Karakter üst sınır değeri
LOWTRAIT=-1*SIG(1,:);        % Karakter alt sınır değeri

%SIG_FIGS=[1.....1]';        % Herbir karakterdeki gen sayısı
SIG_FIGS=2*SIG(1,:)';
DECIMAL=SIG(1,:);            % Karakter buyukluğu
MUTAT_PROB=0.08;             % Mutasyon olasılığı ( <.1)
CROSS_PROB=0.9;              % Çaprazlama olasılığı  ( 1'e yakın)
SELF_ENTERED=0;              % "0": random başlangıç nesli.
POP_SIZE=30;                 %  populasyondaki fert sayısı
ELITISM=0;
EPSILON = 0.025;
MAX_GENERATION=30;           % Algoritmanın çalıştırılacağı Nesil sayısı
clear SIG;
ref=1;
popcount=1;

if SELF_ENTERED == 0     % Random olarak baslangıç havuzu oluşturma

for pop_member = 1:POP_SIZE
    for current_trait = 1:NUM_TRAITS,
        trait(current_trait,pop_member,popcount) = (rand-(1/2))*(HIGHTRAIT(current_trait) +..
         LOWTRAIT(current_trait)) + (1/2)*(HIGHTRAIT(current_trait)+LOWTRAIT(current_trait));
    end
    if pop_member==POP_SIZE
        trait(:,POP_SIZE,popcount)=0;
    end
end
else
 for pop_member = 1:POP_SIZE
     for current_trait = 1:NUM_TRAITS,
        trait(current_trait,pop_member,popcount)=0;
     end
end
end

  CHROM_LENGTH=sum(SIG_FIGS)+NUM_TRAITS;
  TRAIT_START(1)=1;

  for current_trait=1:NUM_TRAITS,
      TRAIT_START(current_trait+1) = TRAIT_START(current_trait)+SIG_FIGS(current_trait)+1;
  end
```

```
while popcount <= MAX_GENERATION   % ana program
   popcount
   for pop_member = 1:POP_SIZE
   for current_trait = 1:NUM_TRAITS,

   if trait(current_trait,pop_member,popcount)>HIGHTRAIT(current_trait)
         trait(current_trait,pop_member,popcount) = HIGHTRAIT(current_trait);
         elseif   trait(current_trait,pop_member,popcount)<LOWTRAIT(current_trait)
               trait(current_trait,pop_member,popcount)=LOWTRAIT(current_trait);
   end

   if  trait(current_trait,pop_member,popcount) < 0
            pop(TRAIT_START(current_trait),pop_member)=0;
       else
            pop(TRAIT_START(current_trait),pop_member)=9;
    end

temp_trait(current_trait,pop_member)= abs(trait(current_trait,pop_member,popcount));

  for counter=1:DECIMAL(current_trait)-1,
      temp_trait(current_trait,pop_member)=temp_trait(current_trait,pop_member)/10;
   end

% karakterlerin kromozom formuna sokulması
for make_gene = TRAIT_START(current_trait)+1:TRAIT_START(current_trait+1)-1,

pop(make_gene,pop_member)=temp_trait(current_trait,pop_member)-...
  rem(temp_trait(current_trait,pop_member),1);

% Next, we take temp_trait and rotate the next digit to the left so that
    temp_trait(current_trait,pop_member) = (temp_trait(current_trait,pop_member) –
        pop(make_gene,pop_member))*10;
   end
      end   % Ends "for current_trait=..." loop
   end      % Ends "for pop_member=..." loop

% Fitness fonksiyonu ile kural tabanların uyumluluk değeri
  sumfitness = 0;
   fit=zeros(POP_SIZE);
   fit_bar=fit(1,:);
   for chrom_number = 1:POP_SIZE,      % Test fitness
```

```
  kromozom=trait(:,chrom_number,popcount);
  rules = ruletable(kromozom');      % kromozomun kural taban tablosu haline getirme
  newpi; % Fuzzy PI programının kullanılması
  tt=0;indeks=0;
  %fitness_bar çıkarılması
  for n=1:step:tstop
      indeks=indeks+1;
      tt=tt+step;
      fit_bar(chrom_number)=fit_bar(chrom_number)+tt*(e(indeks)^2+tt*c(indeks)^2);
  end
end

for chrom_number = 1:POP_SIZE,       % Test fitness
  fitness(chrom_number)=exp(-10^-5/step*fit_bar(chrom_number));
   if fitness(chrom_number)<0
      fitness(chrom_number)=0;
   end
  sumfitness = sumfitness + fitness(chrom_number);  % Store this for   use below
end

[bestfitness(popcount),bestmember]=max(fitness);
bestfitness(popcount)
bestindividual(:,popcount)=trait(:,bestmember,popcount);

  best_kromozom=trait(:,bestmember,popcount);
  rules=ruletable(best_kromozom');  % en uyumlu rule base

  newpi; //bulanık kontrol programınında motora uygulanması

  plot(v)  // elde edilen hız-zaman eğrisini çizilmesi
  hold on
  vsum(popcount,:)=v;
  avefitness(popcount) = sumfitness / POP_SIZE;

% Yeni populasyonun oluşturulması
for pop_member = 1:POP_SIZE,
      if ELITISM ==1 & pop_member==bestmember  % seçkin fert seçilmesi
            parent_chrom(:,pop_member)=pop(:,pop_member);
  % seçkim fertlerin gelecek nesle aktarılması .
      else
  pointer=rand*sumfitness;    % This makes the pointer for the roulette  wheel.
  member_count=1;             % Initialization
  total=fitness(1);
while total < pointer,  % This spins the wheel to the pointer and finds the  identified by member_count
    member_count=member_count+1;
    total=total+fitness(member_count);
end
```

```
% Next, make the parent chromosome
          parent_chrom(:,pop_member)=pop(:,member_count);
     end
  end
% Reproduce section (i.e., make off-spring - "children")
for parent_number1 = 1:POP_SIZE,    % Crossover (parent_number1 is the
                                                  % individual who gets to mate
        if ELITISM ==1 & parent_number1==bestmember % If elitism on, and
                                                                  % have the elite member
                child(:,parent_number1)=parent_chrom(:,parent_number1);
        else
          parent_number2=parent_number1;  % Initialize who the mate is a mate other than yourself
      parent_number2 = rand*POP_SIZE;      % Choose parent number 2 randomly (a random mate)
      parent_number2 = parent_number2-rem(parent_number2,1)+1;
          end
   if CROSS_PROB > rand          % If true then crossover occurs
      site = rand*CHROM_LENGTH;   % Choose site for crossover number for a site

% The next two lines form the child by the swapping of genetic material between the parents

child(1:site,parent_number1)=parent_chrom(1:site,parent_number1);
child(site+1:CHROM_LENGTH,parent_number1)=...
    parent_chrom(site+1:CHROM_LENGTH,parent_number2);
    else                        % No crossover occurs
child(:,parent_number1)=parent_chrom(:,parent_number1);
         end
  end  % End the "if ELITISM..." statement
end  % End "for parent_number1=..." loop

% Mutate children.
  for pop_member= 1:POP_SIZE,
        if ELITISM ==1 & pop_member==bestmember      % If elitism on, and  have the elite member
                child(:,pop_member)=child(:,pop_member); % Do not mutate the elite member
        else
          for site = 1:CHROM_LENGTH,

      if MUTAT_PROB > rand       % If true then mutate
         rand_gene=rand*10;              % Creat a random gene

        while child(site,pop_member) == rand_gene-rem(rand_gene,1),
          rand_gene=rand*10;
        end;

        % If it is not the same one, then mutate
        child(site,pop_member)=rand_gene-rem(rand_gene,1);
        if rand_gene == 10
           site=site-1;
        end
      end  % End "if MUTAT_PROB > rand ...
```

```
            end  % End for site... loop
          end  % End "if ELITISM..."
  end  % End for pop_member loop

% Create the next generation (this completes the main part of the GA)
  pop=child;       % Create next generation (children  become parents)
  popcount=popcount+1;                    % Increment to the next generation

for pop_member = 1:POP_SIZE
        for current_trait = 1:NUM_TRAITS,
          trait(current_trait,pop_member,popcount)=0; % Initialize variables
place_pointer=1;

 for gene=TRAIT_START(current_trait)+1:TRAIT_START(current_trait+1)-1,
 place=DECIMAL(current_trait)-place_pointer;
 trait(current_trait,pop_member,popcount)=...
 trait(current_trait,pop_member,popcount)+...
   (pop(gene,pop_member))*10^place;
 place_pointer=place_pointer+1;
end

% Determine sign of the traits and fix
% trait(current_trait,pop_member,popcount) so that it has the right sign:

if pop(TRAIT_START(current_trait),pop_member) < 5
  trait(current_trait,pop_member,popcount)=...
    -trait(current_trait,pop_member,popcount);
end
        end % Ends "for current_trait=..." loop
end   % Ends "for pop_member=..." loop

% Terminate the program when the best fitness has not changed
  if  1-abs(bestfitness(popcount-1)) < EPSILON
    break;
   end
end  % End "for pop_count=..." loop - the main loop.

figure
plot(time,vsum)

% End of program
T=cputime;
```

```
%-------------------------------------------------------------------------------
% Bulanık kontrol program rutini  (newpi)
%clear
nume=11;        % No of input membership func.  e
numie=11;       % No of input membership func.  integral-e

g1=2;,g2=8;,g0=34;
rules=g0*rules;
                % Scaling gains for tuning membership functions for
                % e, integ-e and u respectively
                % These can be tuned to try to improve the performance.
we=0.2*(1/g1);  % we is half the width of the triangular input membership function bases

wie=0.2*(1/g2);  % Similar to we but for the int e universe of discourse

base=0.4*g0;    % Base width of output membership fuctions of the fuzzy controller

% Centers of input membership functions for the e universe of
% discourse of fuzzy controller (a vector of centers)
 ce=[-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1]*(1/g1);

% Centers of input membership functions for the int e universe of
% discourse of fuzzy controller (a vector of centers)
cie=[-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1]*(1/g2);

%  Next, we intialize the simulation of the closed-loop system.
%-----------------------------------------------
% Sinao DC motor  parameters are

Ra=12.6;
La=9;
J=0.225;
B=1.75;
Km=1.50;
Kv=15.4;
%            740.7407
%  G(s)= ------------------------
%       s^2 + 1407.8 s +  10890

pay=Km/(La*J);

payda1=(La*B+Ra*J)/(La*J);
payda2=(Ra*B+Kv*Km)/(La*J);

k_p=pay;
w_p=sqrt(payda2);
zeta_p=payda1/(2*w_p);

% Next, we initialize the application
t=0;            % Reset time to zero
```

```
index=2;        % This is time's index (not time, its index).
tstop=5;        % Stopping time for the simulation (in seconds)
step=0.01;      %Integration step size
x=[0;0];        % Intial condition on state       of the car


% Need a state space representation for the plant.  Since our
% plant is linear we use the standard form of xdot=Ax+Bu, y=Cx+Du
% Matrix A of state space representation of plant

A=[0 1;
  -w_p^2 -2*zeta_p*w_p];
B=[0; 1];           % Matrix B of state space representation of plant
C=[k_p 0];          % Matrix C of state space representation of plant

e(1)=0;
ctop=0;
y(1)=0;
r(1)=1;

while t <= tstop
        y(index)=C*x;    % Output of the plant

% Next, we define the reference input r as a sine wave

r(index)=1; % reference input
ref=r(1);

vd(index)=r(index); % reference desired output
v(index)=y(index);  % output

% Fuzzy controller calculations:
% First, for the given fuzzy controller inputs we determine
% the extent at which the error membership functions
% of the fuzzy controller are on (this is the fuzzification part).

ie_count=0;,e_count=0;   % These are used to count the number of
                         % non-zero mf certainities of e and int e

e(index)=vd(index)-v(index);
                         % Calculates the error input for the fuzzy controller

eold=e(index-1);
c(index)=(e(index)-eold)/step;


ctop = ctop+ e(index)*step;      % Sets the value of the integral of e
if ctop>1/g2
  ctop=1/g2;
end
```

```
b(index)= ctop;


% The following if-then structure fills the vector mfe
% with the certainty of each membership fucntion of e for the
% current input e.  We use triangular membership functions.

    if e(index)<=ce(1)                  % Takes care of saturation of the left-most
                                        % membership function
  mfe=[1 0 0 0 0 0 0 0 0 0 0]; % i.e., the only one on is the left-most one
    e_count=e_count+1;,e_int=1;     %  One mf on, it is the left-most one.
    elseif e(index)>=ce(nume)          %Takes care of saturation of the right-most mf
    mfe=[0 0 0 0 0 0 0 0 0 0 1];
    e_count=e_count+1;,e_int=nume; % One mf on, it is the right-most one
    else     % In this case the input is on the middle part of the universe of discourse for e Next,
             % we are going to cycle through the mfs to find all that are on
      for i=1:nume
            if e(index)<=ce(i)
             mfe(i)=max([0 1+(e(index)-ce(i))/we]);
                                        % In this case the input is to the
                                        % left of the center ce(i) and we compute
                                        % the value of the mf centered at ce(i)
                                        % for this input e
                 if mfe(i)~=0
                         % If the certainty is not equal to zero then say
                         % that have one mf on by incrementing our count
                  e_count=e_count+1;
                  e_int=i;          % This term holds the index last entry
                                        % with a non-zero term
                 end
            else
             mfe(i)=max([0,1+(ce(i)-e(index))/we]);
                                                    % In this case the input is to the
                                                    % right of the center ce(i)
                 if mfe(i)~=0
                  e_count=e_count+1;
                  e_int=i;  % This term holds the index of the
                                % last entry with a non-zero term
                 end
            end
            end
    end

% The following if-then structure fills the vector mfie with the
% certainty of each membership fucntion of the integral of e
% for its current value (to understand this part of the code see the above
```

```
% similar code for computing mfe

        if b(index)<=cie(1)       % Takes care of saturation of left-most mf
      mfie=[1 0 0 0 0 0 0 0 0 0 0];
        ie_count=ie_count+1;
        ie_int=1;
       elseif b(index)>=cie(numie)
                                  % Takes care of saturation of the right-most mf
        mfie=[0 0 0 0 0 0 0 0 0 0 1];
        ie_count=ie_count+1;
        ie_int=numie;
       else
              for i=1:numie
               if b(index)<=cie(i)
                mfie(i)=max([0,1+(b(index)-cie(i))/wie]);
                     if mfie(i)~=0
                      ie_count=ie_count+1;
                      ie_int=i;          % This term holds last entry
                                         % with a non-zero term
                     end
               else
                mfie(i)=max([0,1+(cie(i)-b(index))/wie]);
                     if mfie(i)~=0
                      ie_count=ie_count+1;
                      ie_int=i;          % This term holds last entry
                                         % with a non-zero term
                     end
               end
              end
       end

%  crisp çıkışın hesaplanması.

num=0;
den=0;
       for k=(e_int-e_count+1):e_int
                                         % Scan over e indices of mfs that are on
              for l=(ie_int-ie_count+1):ie_int
                                         % Scan over int e indices of mfs that are on
               prem=min([mfe(k) mfie(l)]);
               num=num+rules(k,l)*base*(prem-(prem)^2/2);
               den=den+base*(prem-(prem)^2/2);
              end
       end

  u(index)=num/den;
% Bulanık kontrollerin crisp çıkışının dc motora uygulanması

% , the Runge-Kutta equations
time(index)=t;
```

```
        time(index)=t;
        F=A*x+B*u(index);
        k1=step*F;
        xnew=x+k1/2;
        F=A*xnew+B*u(index);
        k2=step*F;
        xnew=x+k2/2;
        F=A*xnew+B*u(index);
        k3=step*F;
        xnew=x+k3;
        F=A*xnew+B*u(index);
        k4=step*F;
        x=x+(1/6)*(k1+2*k2+2*k3+k4); % Calculated next state

    t=t+step;
    index=index+1;
end % "while" sonu

%-------------------------------------------------------------------------------------------------------
% kromozomlardan kural taban oluşturma rutini
function y=ruletable(dizi);
n=size(dizi,2); %dizi uzunlugu
L=sqrt(n);
k=1;
 for i=1:L
  for j=1:L
      y(i,j)=dizi(k);
      k=k+1;
 end
end
```

EK 3

SN10.mat dosyasına ait tüm eniyi kural tabanlar

1	0.0835	-0.9863	-0.0973	-0.6087	0.5743	0.2371	-0.9690	0.7817	0.5234	0.8141	0.5171
	-0.2385	-0.3378	0.0082	0.1291	0.5344	0.5597	-0.0318	0.6044	-0.0580	-0.5945	0.1592
	0.3330	0.3535	0.8850	0.5403	0.4748	0.7325	0.9819	0.0079	0.2582	0.5852	-0.1027
	0.0487	-0.6571	-0.7387	-0.5624	-0.7890	-0.7171	-0.0861	0.5763	-0.4379	-0.5504	0.8177
	-0.9853	0.1775	0.0842	0.3070	-0.3731	-0.5377	-0.1679	-0.4024	0.3449	0.8765	-0.3137
	0.1259	-0.7622	-0.6620	-0.4422	0.1136	-0.0288	0.9044	-0.5362	-0.0427	0.0530	0.5854
	-0.6140	0.8192	0.8444	-0.9735	0.5351	0.8947	0.6266	0.8477	-0.6020	0.3485	0.8542
	-0.3124	0.1890	0.2310	-0.9933	0.9640	0.7990	0.3855	-0.1207	0.4020	0.2194	-0.4002
	0.7121	-0.7759	-0.4169	-0.8051	-0.2051	-0.3334	0.8885	0.6771	-0.4831	-0.9142	-0.9882
	0.1488	0.4878	0.6137	0.2751	-0.4975	-0.7114	0.3031	0.8922	0.6318	0.8605	-0.3801
	-0.4624	0.0729	-0.6734	-0.5780	-0.5664	0.3036	-0.8944	-0.5414	0.3349	-0.3781	-0.3867
2	0	-0.9000	0	-0.9000	0.5000	0.2000	-0.9000	0.7000	0.5000	0.8000	0.5000
	-0.2000	-1.0000	0	0.1000	0.5000	0.5000	0	0.6000	0	-0.5000	-0.1000
	0.3000	0.3000	0.8000	0.5000	0.4000	0.7000	0.9000	0	0.2000	0.5000	-0.1000
	0	-0.6000	-0.7000	-0.5000	-0.7000	-1.0000	0	0.5000	-0.4000	-0.5000	0.8000
	-0.9000	0.1000	0	0.3000	0	-1.0000	-0.1000	-0.4000	0.3000	0.8000	-0.3000
	0.1000	-0.7000	-0.6000	-0.4000	0.1000	0	0.9000	-0.5000	0	0	0.5000
	-0.6000	0.8000	0.8000	-0.9000	0.5000	0.8000	-0.6000	0.8000	-0.6000	1.0000	0.4000
	0.2000	-0.1000	-0.6000	0.6000	0.9000	-0.7000	-0.4000	0.8000	0.3000	0.2000	0
	-0.9000	-0.5000	0.9000	-0.7000	-0.9000	0	-0.1000	0.5000	0	0	0.5000
	0.1000	-0.7000	-0.7000	0.3000	0.9000	-0.3000	-0.6000	-0.8000	-0.3000	-0.3000	-0.2000
	0.9000	0.4000	0.5000	0.2000	0.2000	0.5000	-0.7000	0.1000	0.7000	-0.6000	0.6000
3	0.6000	-0.5000	0.5000	0.7000	0.6000	0.5000	0.8000	-0.1000	0.3000	-0.9000	0.9000
	-0.1000	0.3000	0.3000	-0.1000	0.6000	0	0.4000	-1.0000	1.0000	0.9000	0.9000
	0.4000	-0.5000	0	-0.9000	-1.0000	1.0000	-0.5000	-0.1000	0.9000	-0.4000	-0.8000
	-1.0000	-0.6000	-0.1000	0	-0.9000	-0.2000	0	0.4000	-0.9000	0.9000	0
	1.0000	0.6000	0.1000	0	-0.3000	-0.5000	-0.1000	0.4000	0.3000	0.8000	-0.3000
	0.1000	-0.7000	-0.6000	-1.0000	0.1000	0	0.9000	-0.5000	0	1.0000	0.5000
	-0.6000	0.8000	0.8000	-0.9000	0.5000	-0.8000	0.6000	1.0000	-0.6000	0.3000	0.8000
	-0.3000	1.0000	-0.2000	-0.9000	0.9000	0.7000	0.7000	-0.1000	0.4000	0.2000	-0.4000
	0.7000	-0.7000	-0.6000	-0.3000	-0.2000	-0.3000	0.7000	0.6000	-0.4000	-0.6000	-0.9000
	0.1000	1.0000	0.3000	1.0000	-1.0000	-0.7000	0.3000	-0.8000	0.6000	0.8000	-0.3000
	-0.4000	0	-0.6000	-0.5000	-0.5000	0.3000	-0.9000	-0.5000	-0.3000	-0.3000	-0.3000

4	0.6000 -0.5000 0.5000 0.7000 0.6000 0.5000 0.8000 -0.1000 0.3000 0.9000 0.9000 -0.1000 0.3000 0.3000 -0.1000 0.6000 0 0.4000 -1.0000 1.0000 -0.9000 0.9000 0.4000 -0.5000 0 -0.9000 -1.0000 1.0000 -0.5000 -0.1000 0.9000 -0.4000 -0.8000 -1.0000 -0.6000 -0.1000 0 -0.4000 -0.2000 0 0.4000 -0.1000 0.9000 0 1.0000 0.6000 0.1000 0 -0.3000 -1.0000 -0.1000 0.4000 1.0000 0.7000 -0.1000 0.1000 -0.7000 -0.6000 -1.0000 0.1000 0 0.9000 -0.5000 0 1.0000 0.5000 -0.6000 0.8000 0.8000 -0.9000 1.0000 -0.8000 0.6000 1.0000 -0.6000 0.3000 0.8000 -0.3000 1.0000 -1.0000 -0.9000 0.9000 0.7000 1.0000 -0.1000 0.4000 -0.2000 1.0000 0.7000 -0.7000 -0.4000 0.9000 -1.0000 -0.3000 0.3000 1.0000 -0.4000 -0.9000 -1.0000 1.0000 -0.7000 -1.0000 0.3000 0.9000 -0.3000 -0.6000 -0.8000 -0.3000 -1.0000 -0.2000 0.6000 0 -0.6000 -1.0000 -0.5000 0.3000 -1.0000 -0.5000 0.3000 -1.0000 -0.3000
5	0 -0.6000 0 -0.9000 -0.5000 1.0000 -1.0000 -0.7000 0.5000 0.8000 -0.5000 -0.4000 -1.0000 0 -0.1000 0.5000 0.5000 0 0.6000 0 -0.8000 -0.2000 0.3000 -0.3000 -1.0000 0.5000 0.4000 0.7000 0.9000 0 0.2000 0.5000 -1.0000 0 -0.6000 -1.0000 -0.5000 0 -1.0000 1.0000 0.5000 -0.4000 -0.5000 -0.8000 -0.9000 0.1000 0 1.0000 0 -1.0000 -0.9000 0.8000 0.3000 0.8000 -0.3000 0.1000 -0.7000 -0.6000 -0.4000 0.1000 0 1.0000 -0.5000 0 0 0.5000 -0.6000 1.0000 0.8000 -0.9000 0.5000 0.3000 0.2000 -1.0000 -0.6000 0.3000 0.9000 0.3000 0.4000 0 -1.0000 0.9000 0.7000 1.0000 -0.1000 0.4000 -1.0000 -1.0000 0.7000 -0.8000 -0.4000 0.8000 -1.0000 -0.3000 0.3000 0.6000 -0.4000 -0.9000 -1.0000 1.0000 -0.7000 -1.0000 -0.3000 0.9000 -0.3000 -0.6000 -0.8000 -0.3000 -1.0000 -0.2000 0.9000 0.9000 0.5000 0.2000 0.2000 -0.5000 -0.7000 1.0000 0.1000 -0.6000 0.6000
6	0.3000 -0.9000 0.8000 0.2000 0.5000 -0.2000 -0.9000 0.7000 0.5000 0.8000 1.0000 -0.9000 -0.3000 0 -0.4000 0.5000 0.5000 0.6000 0.6000 0 -0.5000 0.1000 0.3000 0.3000 0.9000 0.5000 1.0000 1.0000 0.9000 0 0 0.5000 -0.1000 1.0000 -0.9000 0 -0.5000 -0.7000 0.7000 0 0.5000 -0.4000 -0.5000 0.8000 -0.9000 -0.1000 0 -1.0000 -0.3000 -1.0000 -0.3000 -0.4000 1.0000 1.0000 -0.2000 0.1000 -0.7000 -0.3000 -0.4000 0.1000 0.6000 -0.9000 -0.3000 0 0 0.2000 -0.6000 1.0000 0.8000 -0.9000 0.5000 0.3000 0.2000 -1.0000 0.6000 0.3000 0.7000 0.3000 0.2000 0 -1.0000 1.0000 0.7000 1.0000 -0.2000 1.0000 -0.1000 -1.0000 0.7000 -0.8000 -0.4000 0.8000 -1.0000 -0.3000 1.0000 0.6000 -0.4000 -0.9000 -1.0000 0 0.7000 -1.0000 -0.3000 0.9000 -0.3000 -0.6000 -0.8000 -0.3000 -1.0000 -0.5000 0.9000 0.4000 0.5000 0.2000 0.2000 0.5000 -0.5000 1.0000 0.1000 -0.6000 0.6000

7	1.0000	-1.0000	0.8000	-1.0000	0.5000	0.2000	0.9000	1.0000	0.5000	0.4000	0
	-0.4000	-0.3000	0.8000	-0.5000	0.5000	0.5000	0.6000	0.6000	0	-0.7000	-0.1000
	0.3000	0.8000	0.8000	0.5000	1.0000	0.7000	0.9000	0	0	0.5000	-0.1000
	1.0000	-0.9000	-1.0000	-0.5000	-0.7000	-0.7000	0	0.5000	-1.0000	-0.5000	0.8000
	-0.9000	0.5000	0	-0.3000	0.4000	-1.0000	-0.1000	-0.3000	-0.3000	0.8000	-0.3000
	-1.0000	-0.7000	-0.6000	-0.4000	1.0000	0	1.0000	-1.0000	0	0	0.5000
	-0.6000	1.0000	0.7000	-0.9000	-1.0000	0.8000	-1.0000	0.8000	-1.0000	1.0000	0.4000
	0.2000	-0.1000	-0.6000	0.6000	0.6000	0.9000	0.3000	0.8000	1.0000	0.2000	1.0000
	-0.8000	0.4000	0.9000	1.0000	-0.9000	0.6000	-0.1000	-0.5000	0	1.0000	0.3000
	0.1000	-0.7000	-0.7000	0.2000	0.9000	-0.3000	-0.6000	-1.0000	-0.3000	-0.3000	-0.5000
	0.9000	0.4000	0.5000	0.2000	0.2000	0.5000	-1.0000	1.0000	0.1000	0.6000	0.6000
8	1.0000	0.9000	0.8000	-1.0000	0.5000	0.2000	0.9000	-1.0000	0.5000	0.4000	0
	-0.4000	-0.3000	0.8000	-0.4000	0	0.2000	0.6000	1.0000	0	1.0000	-0.1000
	0.3000	0.8000	0.8000	0.5000	1.0000	0.7000	0.9000	0	0	0.5000	-0.1000
	1.0000	-0.9000	-1.0000	-0.5000	-0.7000	-0.7000	0	1.0000	-1.0000	-0.5000	0.8000
	-0.9000	0.5000	0	0.3000	0.4000	-1.0000	-0.1000	-0.3000	-0.3000	0.8000	0
	-1.0000	-0.7000	-0.6000	-0.4000	1.0000	0.2000	1.0000	-1.0000	1.0000	0	0.5000
	-1.0000	1.0000	0.7000	-0.9000	-0.9000	0.6000	-1.0000	0.8000	-0.6000	1.0000	1.0000
	-0.2000	-0.1000	0	0.6000	1.0000	-1.0000	0.4000	0.8000	0.5000	0.2000	1.0000
	-0.9000	-1.0000	0.9000	-0.2000	-0.9000	0.3000	1.0000	0.6000	-0.3000	-1.0000	-1.0000
	1.0000	-0.7000	0	-0.3000	0.9000	-0.3000	-0.6000	-0.8000	-1.0000	-1.0000	-0.2000
	-0.9000	0.4000	0.5000	0.2000	0.2000	0.5000	-1.0000	1.0000	1.0000	0.6000	0.6000
9	1.0000	0.9000	0.8000	-1.0000	0.5000	0.2000	0.9000	-1.0000	1.0000	0.4000	0.2000
	-0.4000	-0.3000	0.8000	-0.4000	0	-0.2000	1.0000	1.0000	0	1.0000	-0.1000
	0.3000	0.8000	0.8000	0.5000	1.0000	0.7000	0.9000	0	0	0.5000	-0.1000
	1.0000	-0.9000	-1.0000	-0.5000	-0.7000	-0.7000	0	1.0000	-1.0000	-0.4000	0.8000
	-0.9000	0.5000	0	0.3000	0.4000	-1.0000	-0.1000	0	-0.3000	0.8000	0
	-1.0000	-0.7000	-0.6000	-0.4000	1.0000	0.2000	1.0000	-1.0000	1.0000	0	0.5000
	-1.0000	1.0000	0.7000	-0.9000	-0.9000	0.6000	-1.0000	0.8000	-0.6000	1.0000	1.0000
	-0.2000	-0.1000	0.3000	0.6000	1.0000	-1.0000	0.4000	0.8000	0.5000	0.2000	1.0000
	-0.9000	-1.0000	0.9000	-0.2000	-0.9000	0.3000	1.0000	0.6000	-0.3000	-1.0000	-1.0000
	1.0000	-0.7000	1.0000	0.3000	0.9000	-0.3000	-0.6000	-1.0000	-1.0000	-1.0000	-0.2000
	-0.9000	0.4000	0.5000	0.5000	0.2000	0.7000	-1.0000	1.0000	1.0000	0.6000	0.6000

EK 4 : GBS algoritmasının 30 nesil için elde edilen kural tabanlar

Eniyi fitness = 0.9670

Rule base indis no = 22

rule base i = 1

0.4414	0.9088	-0.7377	-0.8634	-0.7495	-0.6677	0.8228	-0.7275	0.2340	-0.4620	-0.5587
0.4258	0.0980	0.8827	-0.3403	0.4090	0.8869	0.1632	0.7603	0.4992	-0.2408	0.4511
-0.6744	0.9124	-0.6075	0.5524	0.2266	-0.6754	-0.9379	-0.4227	0.9421	0.9010	-0.5439
0.9171	0.3597	-0.8901	0.1997	-0.2137	-0.5693	-0.6352	-0.8465	-0.9852	0.5776	-0.9644
0.7559	-0.2949	0.4443	0.9369	-0.6887	-0.6741	-0.3732	-0.9412	-0.2847	-0.9456	0.5873
0.9985	-0.7795	0.2452	-0.7349	-0.3799	-0.7304	-0.5533	-0.2069	-0.7297	-0.5179	0.8550
-0.2178	0.0225	-0.8142	-0.9566	-0.6809	0.6890	0.7583	-0.6260	0.9826	0.4241	0.7427
-0.0407	-0.0080	-0.4249	-0.8781	-0.4751	-0.6275	0.8342	-0.7534	-0.9731	-0.2606	0.3973
0.7787	0.1875	-0.6866	-0.3666	-0.5332	-0.9832	-0.2062	0.2997	-0.8300	0.5376	0.9394
0.4296	0.5639	-0.5249	-0.6085	-0.4736	0.4276	0.9552	0.2742	0.0918	0.6961	0.6042
0.3366	0.3420	0.6413	0.9409	-0.0262	0.6349	0.2831	-0.3873	0.3219	-0.2840	0.8764

rule base i = 2

-0.7000	0.8000	0	-0.5000	-0.1000	0.6000	-0.7000	-0.3000	0.6000	-0.4000	-0.4000
0.8000	-0.6000	-0.2000	-1.0000	0.6000	-0.1000	0.5000	-0.2000	0.9000	0.1000	0.9000
-0.4000	0	-0.3000	-0.8000	-0.4000	0.3000	0.9000	-0.9000	0.6000	-0.7000	0.9000
-0.4000	-0.5000	0	-0.1000	0.6000	-0.4000	-1.0000	-0.5000	0.1000	-0.5000	0.5000
-0.3000	0	0	0.3000	-0.6000	-0.4000	0.8000	-0.5000	-0.7000	-1.0000	0.2000
-0.6000	-0.1000	0.2000	-1.0000	-0.3000	-0.7000	0.5000	-0.2000	-0.7000	-0.5000	0.8000
-0.2000	0	-0.8000	-0.9000	-0.8000	0.6000	0.7000	-1.0000	0.9000	0.4000	0.7000
0	0	-0.4000	0.8000	0.4000	-0.6000	0.8000	-0.7000	-0.9000	-0.2000	0.3000
0.7000	0.1000	-0.6000	-0.3000	-0.5000	-0.9000	-0.2000	1.0000	-0.8000	0.5000	0.900C
0.4000	0.5000	-0.5000	-0.6000	-0.4000	0.4000	0.9000	0.2000	0.5000	0.6000	0.600C
0.3000	0.3000	0.6000	0.6000	0	0.6000	-0.2000	1.0000	0.3000	-0.2000	0.800C

rule base i = 3

0.9000	0	0.2000	0	0.7000	0.5000	0	-0.9000	0.6000	-0.1000	0.2000
0.5000	0.8000	0.4000	0	0.6000	-0.1000	0.1000	0.1000	0.5000	-0.2000	0.4000
-0.6000	0.9000	-0.6000	0.5000	0.3000	-0.6000	-0.9000	-1.0000	1.0000	0.9000	0.5000
1.0000	0.3000	-0.8000	-0.1000	-0.1000	-0.6000	-1.0000	-0.8000	-0.6000	-0.5000	0
0.7000	-0.2000	-0.4000	0.9000	-0.3000	-0.6000	0.3000	-0.9000	-0.2000	-0.9000	0.5000
0.9000	-1.0000	1.0000	-0.7000	-0.5000	-0.7000	-0.8000	-0.2000	-0.2000	1.0000	-0.8000
-0.2000	0	0	-0.9000	-0.6000	0.6000	0.7000	-0.6000	0.9000	0.4000	0.7000
0	1.0000	-0.4000	-0.8000	-0.4000	-0.6000	0.8000	-1.0000	-0.9000	-0.2000	0.3000
-0.7000	0.1000	-1.0000	-1.0000	-0.5000	-0.9000	-0.2000	0.2000	-0.6000	0.5000	0.9000
0.4000	0.5000	0.3000	-0.6000	-0.7000	0.4000	0.9000	0.2000	0	0.6000	0.6000
-0.9000	0.2000	0.7000	1.0000	0.8000	-0.7000	0.5000	0.7000	0.4000	-0.8000	0.7000

rule base i = 4

-0.3000	-0.9000	0	-0.9000	0.5000	0.2000	-0.9000	0.7000	0.5000	1.0000	0.3000
-0.2000	-0.3000	1.0000	-0.1000	1.0000	0.7000	0	0.6000	0	-0.6000	0.1000
1.0000	0.5000	0.6000	-0.5000	0.4000	0.7000	0.9000	0	0.8000	1.0000	-1.0000
0	-0.6000	-0.7000	-1.0000	-0.7000	-0.7000	-1.0000	0.2000	-0.4000	-0.5000	0.8000
1.0000	1.0000	-0.7000	1.0000	-0.4000	0.9000	-0.8000	-1.0000	-0.9000	0.7000	0.2000
-0.3000	0.3000	-0.5000	-0.5000	-0.1000	-0.2000	-0.4000	0.5000	-1.0000	1.0000	0
0.7000	-0.4000	1.0000	-0.3000	-0.3000	-0.2000	0.7000	0	0.7000	-0.3000	0.2000
1.0000	-0.3000	1.0000	-0.7000	0.5000	1.0000	-0.7000	0.7000	0	-0.4000	-0.8000
1.0000	0.2000	0.4000	-1.0000	-0.2000	-0.6000	0.7000	-0.4000	1.0000	0.5000	-1.0000
1.0000	-0.9000	-1.0000	0.3000	-0.7000	0.8000	1.0000	0.5000	-0.2000	0.6000	0.3000
1.0000	-0.4000	0.1000	0.3000	-0.2000	-0.6000	1.0000	0	0.6000	-0.5000	1.0000

rule base i =5

0.4000	-0.8000	-0.2000	0.6000	0	0.8000	1.0000	-1.0000	0.7000	-0.5000	0.3000
0.9000	-0.1000	0.4000	-0.6000	-0.3000	0	-0.2000	0.1000	0	-1.0000	0.3000
0.2000	-1.0000	0.8000	-1.0000	-1.0000	0.6000	-0.9000	-1.0000	0.9000	-1.0000	-0.4000
0.6000	0.4000	-0.8000	1.0000	0.2000	-1.0000	-0.6000	-1.0000	-0.5000	0.5000	-0.9000
0	-0.2000	0.9000	0.5000	-0.6000	-0.8000	0	-0.4000	-0.2000	-1.0000	0.5000
0.9000	0	1.0000	-0.7000	-0.3000	-0.7000	-0.6000	-0.2000	-0.7000	-1.0000	0.8000
-0.9000	0.6000	-0.8000	-0.9000	-0.6000	0.6000	0.7000	-0.2000	0.9000	0.4000	1.0000
0	0	-0.4000	-1.0000	0	0.6000	1.0000	-0.8000	-1.0000	-1.0000	0.3000
0.7000	0.1000	-0.6000	-0.7000	-0.5000	-0.9000	-0.3000	1.0000	-0.8000	0.5000	-1.0000
0.7000	0.5000	-0.5000	-0.6000	0	1.0000	1.0000	0.4000	-1.0000	1.0000	1.0000
0.5000	-1.0000	-0.7000	-0.7000	-0.6000	-0.9000	0	0.6000	-1.0000	1.0000	1.0000

rule base i =6

-1.0000	-0.9000	0.2000	-0.9000	1.0000	-0.2000	-1.0000	-1.0000	1.0000	1.0000	0.5000
0.2000	-0.3000	1.0000	-0.5000	1.0000	1.0000	0	-0.6000	1.0000	-0.6000	0.1000
1.0000	1.0000	0.8000	-0.5000	0.4000	0.7000	0.9000	0	-0.8000	-1.0000	-1.0000
1.0000	-0.6000	-0.2000	-1.0000	-0.7000	-0.7000	-1.0000	0.2000	-1.0000	-0.3000	0.8000
-1.0000	1.0000	-0.7000	1.0000	-0.4000	0.9000	-0.8000	-1.0000	-0.2000	0.7000	-0.2000
-0.2000	0.3000	-0.5000	-0.5000	-1.0000	-0.2000	-1.0000	1.0000	-1.0000	1.0000	0
0.7000	-0.1000	1.0000	-1.0000	-0.3000	-0.2000	0.7000	0	0.7000	-1.0000	1.0000
1.0000	-0.3000	1.0000	-0.7000	0.5000	1.0000	-0.7000	0.7000	0	-0.4000	-1.0000
-1.0000	0.2000	0	-1.0000	-0.2000	-0.7000	0.7000	1.0000	-1.0000	0.8000	1.0000
-0.1000	-0.7000	-0.1000	-0.1000	0	0	0.5000	0.8000	0.7000	1.0000	1.0000
-1.0000	-0.8000	-0.6000	-0.3000	0.8000	-1.0000	1.0000	-1.0000	-1.0000	-0.1000	1.0000

rule base i =7

-0.8000	0.1000	-1.0000	1.0000	0	0	0.5000	0	1.0000	0	0
0.8000	1.0000	1.0000	0.2000	0.2000	0	1.0000	1.0000	0	-0.3000	1.0000
0	0	1.0000	0.7000	1.0000	1.0000	0	0.8000	1.0000	0	-1.0000
1.0000	1.0000	1.0000	1.0000	0	0	0	0	1.0000	1.0000	0
0	1.0000	1.0000	0.8000	0	1.0000	-0.8000	-0.2000	-0.9000	0	0
-0.9000	1.0000	-0.7000	-1.0000	1.0000	0.4000	-0.5000	-1.0000	0.5000	0	-0.5000
1.0000	-1.0000	1.0000	-1.0000	0	-0.3000	0.2000	-0.7000	1.0000	-0.2000	-1.0000
-1.0000	1.0000	1.0000	-0.2000	-0.7000	0.4000	-1.0000	1.0000	-1.0000	0	0.5000
0.3000	0	1.0000	-0.7000	0.4000	0.7000	0.5000	1.0000	-1.0000	1.0000	1.0000
0.1000	-0.7000	-0.1000	-0.1000	0	0	1.0000	1.0000	-0.7000	1.0000	1.0000
-1.0000	-0.8000	-1.0000	-0.3000	0.8000	-1.0000	1.0000	-1.0000	-1.0000	-0.1000	1.0000

rule base i =8

-0.8000	-0.1000	-0.7000	-1.0000	0.6000	0.1000	-1.0000	0.6000	1.0000	-1.0000	0.1000
0.4000	-0.1000	1.0000	-1.0000	1.0000	1.0000	1.0000	0.3000	1.0000	-0.6000	-1.0000
1.0000	0.9000	1.0000	1.0000	1.0000	0.8000	-0.9000	0	0.8000	1.0000	-1.0000
1.0000	-0.6000	-0.7000	-1.0000	0.7000	-0.1000	0.1000	-1.0000	1.0000	0.8000	1.0000
-1.0000	-1.0000	0.8000	1.0000	0.7000	0.2000	0.4000	-1.0000	0.9000	0	0.8000
0	0.6000	1.0000	-0.6000	-1.0000	1.0000	-0.4000	1.0000	0.3000	0.1000	-1.0000
0.7000	-1.0000	1.0000	0.4000	-1.0000	1.0000	-1.0000	1.0000	-1.0000	0	0.3000
-0.5000	1.0000	1.0000	-1.0000	1.0000	-0.1000	-0.5000	0.4000	0.2000	-1.0000	0.9000
-0.3000	0.3000	0.6000	1.0000	-0.2000	1.0000	0.7000	-0.2000	-1.0000	0.6000	1.0000
-1.0000	-0.7000	-1.0000	-0.3000	-1.0000	-0.3000	0.5000	1.0000	-0.3000	-0.2000	1.0000
1.0000	0.4000	-0.5000	0.2000	0.6000	0.4000	-1.0000	1.0000	1.0000	-0.1000	1.0000

rule base i =9

-0.4000	1.0000	-0.2000	-1.0000	0.6000	1.0000	1.0000	0.6000	1.0000	1.0000	0.1000
1.0000	-1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	0.6000	1.0000	-0.6000	-0.1000
1.0000	0.9000	1.0000	-0.6000	-1.0000	1.0000	1.0000	1.0000	0.5000	0	1.0000
1.0000	-1.0000	-0.3000	-1.0000	0.7000	-0.7000	-1.0000	-0.1000	-1.0000	-0.5000	0.2000
-1.0000	1.0000	-0.7000	1.0000	-0.4000	0.9000	1.0000	1.0000	-0.2000	0.7000	-0.2000
-0.1000	1.0000	0	-0.5000	-1.0000	0.2000	-0.4000	0.5000	-1.0000	-1.0000	0.5000
1.0000	-1.0000	1.0000	0	-0.3000	-0.2000	0.8000	1.0000	1.0000	-1.0000	1.0000
-1.0000	-0.3000	1.0000	-1.0000	0.5000	1.0000	0.9000	1.0000	0.2000	-0.4000	1.0000
-1.0000	1.0000	0	-1.0000	-0.2000	-0.4000	0.7000	-0.4000	1.0000	0.5000	1.0000
1.0000	-1.0000	-1.0000	0.3000	-0.7000	-0.4000	1.0000	1.0000	0.2000	1.0000	1.0000
1.0000	0.4000	0.6000	1.0000	-0.2000	1.0000	1.0000	0	0.6000	0.5000	1.0000

rule base i =10

-1.0000	-0.1000	-0.7000	-1.0000	0.1000	0.1000	1.0000	0.6000	1.0000	-1.0000	1.0000
0.4000	-0.1000	-1.0000	-1.0000	1.0000	1.0000	1.0000	0.3000	1.0000	-0.6000	1.0000
1.0000	0.9000	-1.0000	1.0000	1.0000	0.8000	-0.9000	0	1.0000	1.0000	-1.0000
1.0000	-0.6000	-0.7000	-1.0000	-0.7000	-0.1000	0.6000	-1.0000	1.0000	0.8000	1.0000
-1.0000	-1.0000	0.8000	1.0000	0.7000	0.2000	0.4000	-1.0000	0.9000	0	0.8000
0	0.6000	1.0000	-0.6000	-1.0000	1.0000	-0.4000	-1.0000	0.3000	0.8000	-1.0000
0.7000	-1.0000	1.0000	0.4000	-1.0000	1.0000	-1.0000	1.0000	-1.0000	0	0.2000
-0.5000	1.0000	1.0000	-1.0000	1.0000	-0.1000	-0.5000	0.4000	0.2000	-1.0000	0.9000
-0.3000	0.3000	0	1.0000	-0.2000	1.0000	1.0000	-0.2000	-1.0000	0.6000	1.0000
-1.0000	-1.0000	-1.0000	-0.3000	-1.0000	-0.3000	0.6000	1.0000	-0.3000	-0.2000	1.0000
1.0000	0.4000	0.6000	0.3000	0	1.0000	1.0000	0	0.6000	0.5000	1.0000

rule base i =11

0.8000	-1.0000	0.2000	-1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.1000
-1.0000	-1.0000	1.0000	-0.5000	1.0000	1.0000	1.0000	0.6000	1.0000	-0.6000	-1.0000
1.0000	0.2000	0.1000	-0.6000	1.0000	1.0000	1.0000	-1.0000	0.7000	1.0000	1.0000
1.0000	-1.0000	-0.9000	-1.0000	1.0000	-1.0000	-0.6000	-1.0000	-1.0000	-0.5000	-1.0000
1.0000	1.0000	-1.0000	1.0000	0.9000	-0.1000	-0.4000	-1.0000	1.0000	-0.8000	1.0000
0.9000	-1.0000	-1.0000	-1.0000	0	1.0000	1.0000	-1.0000	1.0000	-0.6000	-1.0000
-1.0000	-1.0000	1.0000	-0.4000	1.0000	0.8000	-0.1000	1.0000	1.0000	1.0000	-1.0000
0.9000	0.3000	1.0000	-1.0000	1.0000	1.0000	-1.0000	1.0000	0.7000	0.1000	1.0000
-1.0000	-1.0000	1.0000	1.0000	-0.7000	0.1000	0.8000	-0.7000	1.0000	1.0000	0.8000
0.1000	-1.0000	-0.3000	0	0.2000	0	0.2000	1.0000	-1.0000	1.0000	1.0000
-1.0000	-0.8000	0.6000	-0.4000	-0.8000	-1.0000	1.0000	-1.0000	0.1000	0	0.9000

rule base i =12

0.8000	-1.0000	0.2000	-1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.1000
-1.0000	-1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	0.6000	1.0000	-0.6000	-1.0000
1.0000	0.2000	1.0000	-0.6000	1.0000	1.0000	1.0000	-1.0000	0.7000	1.0000	1.0000
1.0000	-1.0000	-0.9000	-1.0000	1.0000	-1.0000	-0.6000	-1.0000	-1.0000	-0.5000	-1.0000
1.0000	1.0000	-1.0000	1.0000	0.9000	-0.1000	-0.6000	-1.0000	1.0000	-0.8000	1.0000
0.9000	-1.0000	-1.0000	-1.0000	0	1.0000	1.0000	-1.0000	1.0000	-0.6000	-1.0000
-1.0000	-1.0000	1.0000	0.4000	1.0000	-0.8000	-0.1000	1.0000	1.0000	1.0000	-1.0000
-0.9000	0.3000	1.0000	-1.0000	1.0000	1.0000	-1.0000	1.0000	-0.7000	0.1000	1.0000
-1.0000	-1.0000	1.0000	1.0000	-0.7000	0.1000	0.8000	-0.7000	1.0000	1.0000	0.8000
0.1000	-1.0000	-0.3000	0.3000	-1.0000	0.4000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	0.4000	0.6000	-0.3000	0.2000	1.0000	1.0000	1.0000	1.0000	-0.5000	1.0000

rule base i =13

0.4000	0.3000	-0.4000	-0.9000	0.5000	-1.0000	1.0000	0.8000	1.0000	-1.0000	-1.0000
1.0000	0	-0.5000	0	1.0000	1.0000	0.3000	0.6000	1.0000	1.0000	-0.1000
1.0000	-1.0000	1.0000	-0.7000	1.0000	0.4000	0.2000	0	-0.8000	1.0000	-1.0000
-1.0000	1.0000	0.7000	-1.0000	0.6000	-0.8000	-1.0000	0.5000	1.0000	0.8000	0.2000
-1.0000	0	-0.4000	0.8000	0.4000	-0.1000	1.0000	1.0000	1.0000	0.5000	0.9000
1.0000	-1.0000	0.2000	0.7000	-1.0000	0	1.0000	-1.0000	0.8000	-1.0000	-1.0000
-1.0000	1.0000	1.0000	-0.8000	1.0000	1.0000	1.0000	1.0000	-1.0000	1.0000	0.2000
0.1000	-0.1000	-0.6000	-0.6000	-1.0000	-1.0000	-0.4000	1.0000	-1.0000	1.0000	0
1.0000	-1.0000	-1.0000	0.7000	-0.9000	1.0000	-1.0000	1.0000	0.8000	0.4000	1.0000
-1.0000	-0.7000	-0.1000	0.1000	0.9000	1.0000	-1.0000	1.0000	1.0000	1.0000	0.4000
1.0000	0.8000	0.9000	-1.0000	1.0000	1.0000	-1.0000	1.0000	1.0000	-0.6000	1.0000

rule base i =14

-1.0000	-0.9000	-0.2000	-1.0000	-1.0000	-0.2000	-1.0000	-1.0000	1.0000	1.0000	1.0000
1.0000	-1.0000	0	-1.0000	1.0000	1.0000	0	0.8000	1.0000	-1.0000	1.0000
1.0000	1.0000	1.0000	-1.0000	1.0000	-1.0000	0	-1.0000	-1.0000	-1.0000	-1.0000
1.0000	-0.6000	-1.0000	1.0000	1.0000	-1.0000	-1.0000	-0.1000	-1.0000	1.0000	1.0000
-1.0000	1.0000	-0.7000	1.0000	-0.4000	0.9000	1.0000	-1.0000	-0.6000	0.7000	-0.2000
0.9000	1.0000	0	-0.5000	0	0	0.6000	1.0000	-1.0000	1.0000	1.0000
1.0000	-1.0000	1.0000	-1.0000	-1.0000	0.2000	0.7000	-1.0000	0.7000	-1.0000	0
1.0000	-1.0000	-1.0000	-1.0000	-1.0000	1.0000	-1.0000	1.0000	1.0000	-1.0000	0.8000
1.0000	-1.0000	-0.9000	0.6000	-1.0000	0	1.0000	1.0000	0.4000	0.4000	0.7000
-1.0000	-0.4000	-0.1000	0.7000	0	1.0000	1.0000	0.6000	1.0000	1.0000	1.0000
1.0000	0.7000	0.9000	1.0000	1.0000	0.7000	-1.0000	1.0000	1.0000	-0.6000	1.0000

rule base i =15

-1.0000	0.3000	1.0000	-1.0000	0.5000	-1.0000	1.0000	-0.8000	0	-1.0000	-1.0000
1.0000	0	1.0000	-1.0000	1.0000	1.0000	1.0000	0.1000	-1.0000	-1.0000	0.4000
1.0000	1.0000	1.0000	-0.5000	-0.1000	1.0000	0.1000	1.0000	1.0000	1.0000	-1.0000
1.0000	-0.6000	-1.0000	1.0000	1.0000	-1.0000	-1.0000	-0.1000	-1.0000	1.0000	1.0000
-1.0000	1.0000	-0.7000	1.0000	-0.2000	0.9000	1.0000	-1.0000	-0.6000	0.7000	-0.2000
0.9000	1.0000	1.0000	0.5000	0	0	1.0000	1.0000	-1.0000	1.0000	1.0000
1.0000	-1.0000	1.0000	-1.0000	-1.0000	0.2000	0.7000	-1.0000	0.7000	-1.0000	0
-1.0000	-1.0000	-1.0000	-1.0000	1.0000	1.0000	-1.0000	1.0000	1.0000	-1.0000	0.8000
-1.0000	-1.0000	-0.9000	0.6000	-1.0000	0	1.0000	1.0000	0.4000	0.4000	0.7000
-1.0000	-0.4000	-0.1000	0.7000	0	1.0000	1.0000	0.6000	1.0000	1.0000	1.0000
1.0000	0.7000	-0.9000	1.0000	1.0000	0.7000	-1.0000	1.0000	1.0000	-0.6000	1.0000

rule base i =16

0	1.0000	-1.0000	1.0000	1.0000	1.0000	-0.5000	1.0000	-1.0000	0.6000	1.0000
1.0000	1.0000	1.0000	-1.0000	-1.0000	1.0000	-1.0000	-1.0000	1.0000	-1.0000	0.4000
1.0000	1.0000	1.0000	-0.5000	-0.1000	1.0000	0.1000	1.0000	1.0000	1.0000	-1.0000
1.0000	-0.6000	-1.0000	1.0000	0	-1.0000	-1.0000	-1.0000	-1.0000	1.0000	1.0000
-1.0000	1.0000	-0.7000	1.0000	-0.2000	0.9000	1.0000	-1.0000	-0.6000	0.7000	0.2000
1.0000	1.0000	1.0000	0.5000	0	0	1.0000	1.0000	-1.0000	1.0000	1.0000
1.0000	-1.0000	1.0000	-1.0000	-1.0000	0.2000	0.8000	-1.0000	0.7000	-1.0000	0
-1.0000	-1.0000	-1.0000	-1.0000	1.0000	1.0000	0	-1.0000	1.0000	-1.0000	0.8000
-1.0000	-1.0000	-0.9000	0.6000	-1.0000	1.0000	1.0000	1.0000	0.6000	0.4000	0.7000
-1.0000	-0.4000	-0.1000	0.7000	0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	0.7000	-0.9000	1.0000	-1.0000	0.7000	1.0000	1.0000	1.0000	-0.6000	1.0000

rule base i =17

-0.4000	-1.0000	-1.0000	1.0000	1.0000	1.0000	-0.2000	1.0000	-1.0000	0.8000	1.0000
-1.0000	1.0000	-1.0000	-1.0000	-1.0000	0	-1.0000	1.0000	-1.0000	1.0000	1.0000
0.4000	1.0000	1.0000	1.0000	-1.0000	1.0000	-1.0000	-1.0000	-0.5000	0.4000	-1.0000
1.0000	-1.0000	-0.9000	-1.0000	1.0000	-1.0000	0.6000	-1.0000	-1.0000	0.9000	-1.0000
1.0000	-1.0000	-1.0000	0.8000	-1.0000	0.9000	-1.0000	1.0000	0	-0.8000	-1.0000
0.8000	-1.0000	1.0000	1.0000	0	1.0000	1.0000	-1.0000	1.0000	-0.6000	-1.0000
-1.0000	-0.7000	-0.1000	1.0000	-1.0000	1.0000	0	1.0000	-1.0000	1.0000	1.0000
1.0000	-0.1000	-1.0000	1.0000	1.0000	0.2000	-0.3000	-1.0000	-1.0000	0	1.0000
1.0000	-0.7000	0.9000	0.8000	1.0000	0.1000	1.0000	-1.0000	1.0000	-1.0000	1.0000
-0.8000	-1.0000	-1.0000	1.0000	1.0000	1.0000	-1.0000	-1.0000	0.3000	1.0000	1.0000
1.0000	1.0000	1.0000	-1.0000	1.0000	-0.3000	1.0000	1.0000	1.0000	0.1000	1.0000

rule base i =18

-1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	0.5000	1.0000	-1.0000	0.8000	1.0000
-1.0000	1.0000	0	1.0000	-1.0000	1.0000	-1.0000	1.0000	-1.0000	1.0000	1.0000
-0.4000	1.0000	1.0000	0	-1.0000	1.0000	-1.0000	-0.8000	0.3000	1.0000	0
1.0000	-1.0000	0	1.0000	0	-1.0000	1.0000	-1.0000	-1.0000	-0.4000	1.0000
1.0000	1.0000	-1.0000	1.0000	-1.0000	1.0000	1.0000	-1.0000	-0.2000	1.0000	-0.2000
-0.2000	0	-0.6000	0.5000	-1.0000	0	-0.7000	1.0000	-1.0000	0.6000	1.0000
1.0000	-1.0000	1.0000	-1.0000	-1.0000	1.0000	1.0000	-1.0000	0	-1.0000	0.9000
1.0000	-1.0000	-1.0000	0	-1.0000	1.0000	-1.0000	1.0000	0.3000	-1.0000	1.0000
1.0000	-1.0000	0.1000	0.8000	-1.0000	0.1000	1.0000	-1.0000	1.0000	-1.0000	1.0000
0.1000	-1.0000	-1.0000	1.0000	1.0000	1.0000	-1.0000	-1.0000	-0.3000	1.0000	1.0000
1.0000	-1.0000	1.0000	-1.0000	1.0000	-0.8000	1.0000	-1.0000	1.0000	1.0000	1.0000

rule base i =19

1.0000	-0.5000	-1.0000	1.0000	-0.7000	-1.0000	1.0000	0.6000	1.0000	-1.0000	-1.0000
1.0000	1.0000	1.0000	-1.0000	1.0000	1.0000	-1.0000	-0.6000	1.0000	-1.0000	0
-1.0000	0.4000	1.0000	-0.5000	1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	0.9000	1.0000	1.0000	0	0.7000	1.0000	0.2000	-1.0000	1.0000	-1.0000
-1.0000	1.0000	1.0000	1.0000	-1.0000	0	1.0000	-1.0000	1.0000	1.0000	-1.0000
-1.0000	1.0000	0.8000	0.5000	1.0000	0	-1.0000	-1.0000	-1.0000	1.0000	1.0000
1.0000	-1.0000	1.0000	-1.0000	-1.0000	1.0000	1.0000	-1.0000	0	-1.0000	0.7000
1.0000	-1.0000	-1.0000	0	-1.0000	1.0000	-1.0000	1.0000	0.3000	-1.0000	1.0000
1.0000	-1.0000	0.5000	0.8000	-0.7000	0.1000	1.0000	-1.0000	1.0000	-1.0000	1.0000
0.1000	-1.0000	-1.0000	1.0000	1.0000	1.0000	-1.0000	-1.0000	0.4000	1.0000	0.8000
1.0000	-1.0000	1.0000	0	1.0000	0.8000	1.0000	1.0000	1.0000	1.0000	1.0000

rule base i =20

1.0000	-0.5000	-1.0000	-1.0000	-0.7000	-1.0000	1.0000	0.6000	1.0000	-1.0000	-1.0000
1.0000	1.0000	1.0000	-1.0000	-1.0000	1.0000	-1.0000	-0.6000	1.0000	-1.0000	0.9000
-1.0000	0.3000	1.0000	-0.5000	1.0000	-1.0000	-1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	0.9000	1.0000	1.0000	1.0000	0.7000	1.0000	0.2000	-1.0000	1.0000	-1.0000
-1.0000	1.0000	1.0000	1.0000	-1.0000	0	1.0000	-1.0000	1.0000	1.0000	-1.0000
-1.0000	1.0000	0.8000	0.5000	1.0000	0	-1.0000	-1.0000	-1.0000	1.0000	1.0000
1.0000	-1.0000	1.0000	-1.0000	-1.0000	1.0000	1.0000	-1.0000	1.0000	-1.0000	0.7000
1.0000	-1.0000	-1.0000	0	-1.0000	1.0000	-1.0000	1.0000	0.3000	-1.0000	1.0000
1.0000	0	0.5000	0.8000	-1.0000	0.1000	1.0000	1.0000	1.0000	-1.0000	1.0000
0.7000	-1.0000	-1.0000	1.0000	1.0000	1.0000	-1.0000	1.0000	0.4000	1.0000	0.8000
1.0000	-1.0000	-1.0000	0.4000	1.0000	0.8000	1.0000	1.0000	1.0000	1.0000	1.0000

rule base i =21

0.7000	-1.0000	-0.4000	-1.0000	-1.0000	1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000
1.0000	1.0000	0	1.0000	1.0000	1.0000	-1.0000	1.0000	-1.0000	-1.0000	0.4000
1.0000	0.8000	1.0000	-1.0000	-0.9000	-1.0000	-1.0000	1.0000	1.0000	1.0000	1.0000
-1.0000	1.0000	1.0000	1.0000	1.0000	-1.0000	1.0000	1.0000	-1.0000	1.0000	1.0000
1.0000	1.0000	-0.4000	1.0000	1.0000	-0.9000	1.0000	-1.0000	-1.0000	1.0000	1.0000
1.0000	1.0000	-1.0000	-0.5000	1.0000	0	-1.0000	0	-1.0000	1.0000	1.0000
1.0000	0	1.0000	-1.0000	-1.0000	-0.2000	-1.0000	-1.0000	1.0000	1.0000	0
-1.0000	-1.0000	-1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	0.3000	-1.0000	0.9000
1.0000	-1.0000	0.2000	0	-1.0000	0.3000	1.0000	1.0000	0.4000	1.0000	1.0000
-1.0000	-1.0000	-0.5000	0.5000	-1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-0.7000	1.0000	1.0000	1.0000	-1.0000	1.0000

rule base i =22

0.7000	-0.4000	-0.9000	-1.0000	-1.0000	-1.0000	1.0000	-1.0000	-1.0000	-1.0000	1.0000
-1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	-1.0000	-1.0000	-1.0000	-1.0000	0.6000
-1.0000	-1.0000	1.0000	-1.0000	-1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	0
-1.0000	-0.3000	1.0000	1.0000	-1.0000	-1.0000	-1.0000	1.0000	1.0000	1.0000	1.0000
-1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	-1.0000	-1.0000	-1.0000	1.0000	-1.0000
1.0000	-1.0000	0.7000	1.0000	1.0000	1.0000	-1.0000	-1.0000	1.0000	1.0000	1.0000
1.0000	0	-1.0000	-1.0000	-1.0000	0.2000	-1.0000	-1.0000	1.0000	-1.0000	1.0000
-1.0000	1.0000	1.0000	1.0000	-1.0000	-1.0000	1.0000	0	0.4000	1.0000	1.0000
1.0000	-1.0000	0.2000	0.6000	-1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
-1.0000	-1.0000	-1.0000	1.0000	-1.0000	-1.0000	1.0000	1.0000	-1.0000	1.0000	1.0000
1.0000	-1.0000	1.0000	-1.0000	1.0000	0.8000	1.0000	0.3000	1.0000	1.0000	1.0000

rule base i =23

1.0000	-0.5000	-1.0000	-1.0000	0.7000	-0.5000	1.0000	0.6000	1.0000	-1.0000	-1.0000
1.0000	1.0000	-1.0000	0	-1.0000	-1.0000	1.0000	-1.0000	-1.0000	-1.0000	1.0000
1.0000	-1.0000	1.0000	-0.2000	1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	1.0000
-1.0000	1.0000	1.0000	1.0000	-1.0000	1.0000	1.0000	-1.0000	-1.0000	1.0000	1.0000
-1.0000	1.0000	-1.0000	1.0000	1.0000	-0.9000	1.0000	-1.0000	-1.0000	1.0000	1.0000
1.0000	1.0000	-1.0000	-0.5000	1.0000	0	-1.0000	1.0000	-1.0000	1.0000	-1.0000
1.0000	0.1000	1.0000	1.0000	-1.0000	0	1.0000	-1.0000	1.0000	-1.0000	0.7000
-1.0000	-1.0000	-1.0000	0	-1.0000	1.0000	-1.0000	1.0000	0.3000	-1.0000	1.0000
1.0000	1.0000	0.5000	0.8000	-1.0000	0.1000	1.0000	0	1.0000	-1.0000	1.0000
0.7000	-1.0000	1.0000	1.0000	-1.0000	-1.0000	-1.0000	0	1.0000	1.0000	0.8000
-1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

rule base i =24

1.0000	-0.4000	-1.0000	1.0000	1.0000	-1.0000	1.0000	0.6000	-1.0000	-1.0000	-1.0000
1.0000	1.0000	1.0000	-1.0000	-1.0000	1.0000	1.0000	1.0000	-1.0000	1.0000	1.0000
1.0000	1.0000	1.0000	-0.2000	-1.0000	1.0000	1.0000	1.0000	-1.0000	-1.0000	1.0000
1.0000	1.0000	1.0000	1.0000	0.2000	1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000
1.0000	1.0000	-0.3000	1.0000	1.0000	-0.9000	1.0000	-1.0000	-1.0000	0.4000	1.0000
1.0000	-1.0000	-1.0000	-1.0000	-1.0000	1.0000	-1.0000	1.0000	-1.0000	1.0000	1.0000
0	-1.0000	-1.0000	-1.0000	1.0000	1.0000	-0.8000	1.0000	-1.0000	1.0000	1.0000
-1.0000	1.0000	-1.0000	1.0000	-1.0000	1.0000	1.0000	-1.0000	-0.3000	-1.0000	1.0000
1.0000	-1.0000	-1.0000	1.0000	-1.0000	0.3000	1.0000	-1.0000	1.0000	1.0000	1.0000
-1.0000	-1.0000	-0.5000	0.5000	-1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
-1.0000	-1.0000	0	-1.0000	-1.0000	-0.7000	0	1.0000	1.0000	0	1.0000

rule base i =25

1.0000	-1.0000	-1.0000	1.0000	1.0000	-1.0000	1.0000	1.0000	-1.0000	1.0000	1.0000
1.0000	1.0000	-1.0000	1.0000	-1.0000	1.0000	1.0000	-1.0000	1.0000	1.0000	0.3000
-1.0000	1.0000	1.0000	-1.0000	-1.0000	-1.0000	-1.0000	1.0000	1.0000	-1.0000	1.0000
1.0000	-1.0000	0	-1.0000	-1.0000	0	1.0000	1.0000	1.0000	0	0
1.0000	1.0000	1.0000	1.0000	-1.0000	1.0000	-1.0000	0.9000	1.0000	1.0000	0.4000
1.0000	0.6000	0.9000	-1.0000	0	1.0000	-1.0000	1.0000	-1.0000	1.0000	0
0	-1.0000	-1.0000	-1.0000	1.0000	1.0000	-0.8000	1.0000	-1.0000	-1.0000	1.0000
-1.0000	1.0000	-1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	-0.3000	-1.0000	1.0000
1.0000	-1.0000	-1.0000	1.0000	-1.0000	0.4000	-1.0000	1.0000	1.0000	1.0000	1.0000
-1.0000	-1.0000	-0.5000	-0.5000	-1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
-1.0000	-1.0000	0	-1.0000	-1.0000	-0.7000	0	1.0000	1.0000	0	1.0000

rule base i =26

0.5000	-0.7000	-0.4000	1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000
-1.0000	-1.0000	0	1.0000	1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	0.3000
-1.0000	1.0000	1.0000	-1.0000	-1.0000	-1.0000	-1.0000	1.0000	1.0000	-1.0000	1.0000
1.0000	-1.0000	0	-1.0000	-1.0000	0	1.0000	1.0000	1.0000	0	0
1.0000	1.0000	1.0000	1.0000	-1.0000	1.0000	-1.0000	0.9000	1.0000	1.0000	0.4000
1.0000	0.6000	0.9000	-1.0000	0	1.0000	-1.0000	1.0000	-1.0000	1.0000	0
0.4000	-1.0000	-1.0000	-1.0000	1.0000	1.0000	-0.8000	-1.0000	-1.0000	-1.0000	1.0000
-1.0000	1.0000	-1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	-0.3000	-1.0000	1.0000
1.0000	-1.0000	-1.0000	1.0000	-1.0000	0.4000	-1.0000	1.0000	1.0000	1.0000	1.0000
-1.0000	-1.0000	-0.5000	-0.5000	-1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	1.0000
-1.0000	-1.0000	0	-1.0000	-1.0000	-0.9000	0	1.0000	1.0000	0	1.0000

rule base i =27

1.0000	-1.0000	-1.0000	1.0000	1.0000	-1.0000	1.0000	1.0000	-1.0000	1.0000	1.0000
1.0000	-1.0000	-1.0000	1.0000	-1.0000	1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000
-1.0000	1.0000	1.0000	-1.0000	1.0000	-1.0000	-1.0000	1.0000	-1.0000	-1.0000	1.0000
1.0000	-1.0000	0	-1.0000	-1.0000	0	1.0000	1.0000	1.0000	0	1.0000
1.0000	1.0000	1.0000	1.0000	-1.0000	1.0000	-1.0000	0.9000	1.0000	1.0000	-0.4000
1.0000	0.6000	0.9000	-1.0000	0	1.0000	-1.0000	1.0000	-1.0000	1.0000	0.2000
0.4000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-0.8000	-1.0000	-1.0000	-1.0000	1.0000
1.0000	1.0000	-1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	-0.3000	-1.0000	1.0000
1.0000	-1.0000	-1.0000	1.0000	-1.0000	0.4000	-1.0000	1.0000	1.0000	1.0000	1.0000
-1.0000	-1.0000	-0.5000	-0.5000	-1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	1.0000
-1.0000	-1.0000	0.7000	-1.0000	-1.0000	-0.9000	0	1.0000	1.0000	0.7000	1.0000

rule base i =28

-1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	-1.0000	1.0000	-1.0000	1.0000	-1.0000
1.0000	1.0000	-1.0000	-0.9000	1.0000	1.0000	1.0000	-1.0000	-1.0000	-1.0000	0.9000
-1.0000	0.7000	1.0000	1.0000	0.1000	-1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
-1.0000	0	-1.0000	1.0000	-0.1000	-1.0000	-1.0000	-1.0000	1.0000	1.0000	0
1.0000	1.0000	-1.0000	0	0	-0.9000	1.0000	1.0000	-0.1000	-1.0000	1.0000
1.0000	-1.0000	-1.0000	-0.5000	1.0000	0	1.0000	1.0000	-1.0000	1.0000	1.0000
-1.0000	0.7000	1.0000	-1.0000	1.0000	0	-1.0000	-1.0000	1.0000	1.0000	1.0000
1.0000	-1.0000	1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	0	-1.0000	1.0000
1.0000	-1.0000	0.1000	1.0000	-1.0000	-0.3000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	0	-0.7000	0.8000	-1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
-1.0000	1.0000	1.0000	-1.0000	-1.0000	0	1.0000	1.0000	1.0000	-1.0000	1.0000

rule base i =29

1.0000	1.0000	1.0000	1.0000	-1.0000	-1.0000	-1.0000	1.0000	-1.0000	1.0000	1.0000
1.0000	1.0000	-1.0000	1.0000	1.0000	-1.0000	-1.0000	1.0000	1.0000	-1.0000	0.3000
1.0000	0	-1.0000	0	1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000
-1.0000	0	-0.6000	-1.0000	-1.0000	-1.0000	-1.0000	1.0000	-1.0000	-1.0000	-1.0000
1.0000	1.0000	-1.0000	1.0000	-1.0000	-0.9000	0.8000	-1.0000	1.0000	-1.0000	0.9000
-1.0000	1.0000	0	-1.0000	-1.0000	-1.0000	1.0000	-1.0000	1.0000	-1.0000	0
0.4000	-1.0000	-1.0000	0	1.0000	1.0000	-0.8000	-1.0000	-1.0000	1.0000	1.0000
-1.0000	1.0000	-1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	-0.3000	-1.0000	1.0000
-1.0000	-1.0000	-1.0000	1.0000	-1.0000	0.4000	-1.0000	-1.0000	1.0000	1.0000	1.0000
-1.0000	-1.0000	0.7000	-0.5000	-1.0000	1.0000	-1.0000	1.0000	1.0000	1.0000	1.0000
-1.0000	0	0	1.0000	-1.0000	-0.9000	0	1.0000	1.0000	0.4000	1.0000

rule base i =30

1.0000	-1.0000	0.4000	1.0000	1.0000	-1.0000	1.0000	1.0000	-1.0000	-1.0000	1.0000
1.0000	1.0000	0.6000	0	-1.0000	1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000
1.0000	-0.8000	1.0000	1.0000	-0.9000	-1.0000	-1.0000	1.0000	1.0000	0.8000	1.0000
-1.0000	1.0000	1.0000	1.0000	1.0000	-1.0000	1.0000	1.0000	-1.0000	1.0000	1.0000
-1.0000	-1.0000	-1.0000	1.0000	-0.2000	1.0000	1.0000	-1.0000	-0.2000	1.0000	1.0000
1.0000	-1.0000	-1.0000	1.0000	-1.0000	0.3000	-1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	0	-1.0000	1.0000	-1.0000	0	0	-1.0000	1.0000	1.0000	1.0000
1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	1.0000	0	0.8000	-1.0000	0.5000
0	1.0000	-0.5000	0	-1.0000	0.6000	1.0000	1.0000	1.0000	1.0000	1.0000
0	-1.0000	1.0000	-1.0000	1.0000	1.0000	-0.1000	1.0000	-1.0000	1.0000	1.0000
1.0000	1.0000	-1.0000	1.0000	1.0000	-0.5000	-1.0000	1.0000	-1.0000	1.0000	1.0000

Sinano Co. Firmasına Ait DC Motor Parametreleri

2001年 1月18日 18時08分 シナノデンキ(KK) ポンシャ 3946→2686

FP323-3CEのデータNo. 1911 改訂する.

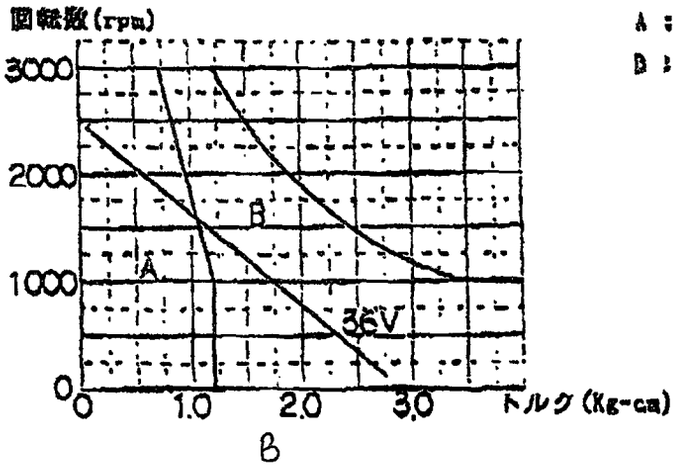
2/2

モータ特性表

平沢 01.1.18 花

項目	単位	FP323-3CE
瞬時最大トルク	kg-cm	6.0
定格トルク	kg-cm	1.2
トルク定数	kg-cm/A	1.50
電機子抵抗	Ω	12.6
電機子インダクタンス	mH	9.0
瞬時最大電流	A	4.3
誘起電圧定数	mV/rpm	15.4
粘性制動係数	g-cm-/rpm	0.05
摩擦トルク	kg-cm	0.16
起動トルク	kg-cm	0.20
コギングトルク	kg-cm	±0.035
イナーシャ(エンコーダ込)	g-cm-sec²	0.225
機械的時定数	msec	12.9
電気的時定数	msec	0.71
パワーレイト	kW/sec	0.63
角加速度	rad/sec²	5330
熱抵抗	deg C/W	3.8
熱時定数	min	—
巻線許容温度	℃	155
定格回転数	rpm	1000
瞬時最大回転数	rpm	3000

定格出力	定格トルク	定格回転数	定格電圧	定格電流
12.3 (W)	1.2 (kg·cm)	1000 (rpm)	30 (V)	1.1 (A)



B

A : 連続使用範囲(全域)

B : 瞬時使用範囲

○ 使用周囲条件

周囲温度 25(℃)

湿 度 80 (%)

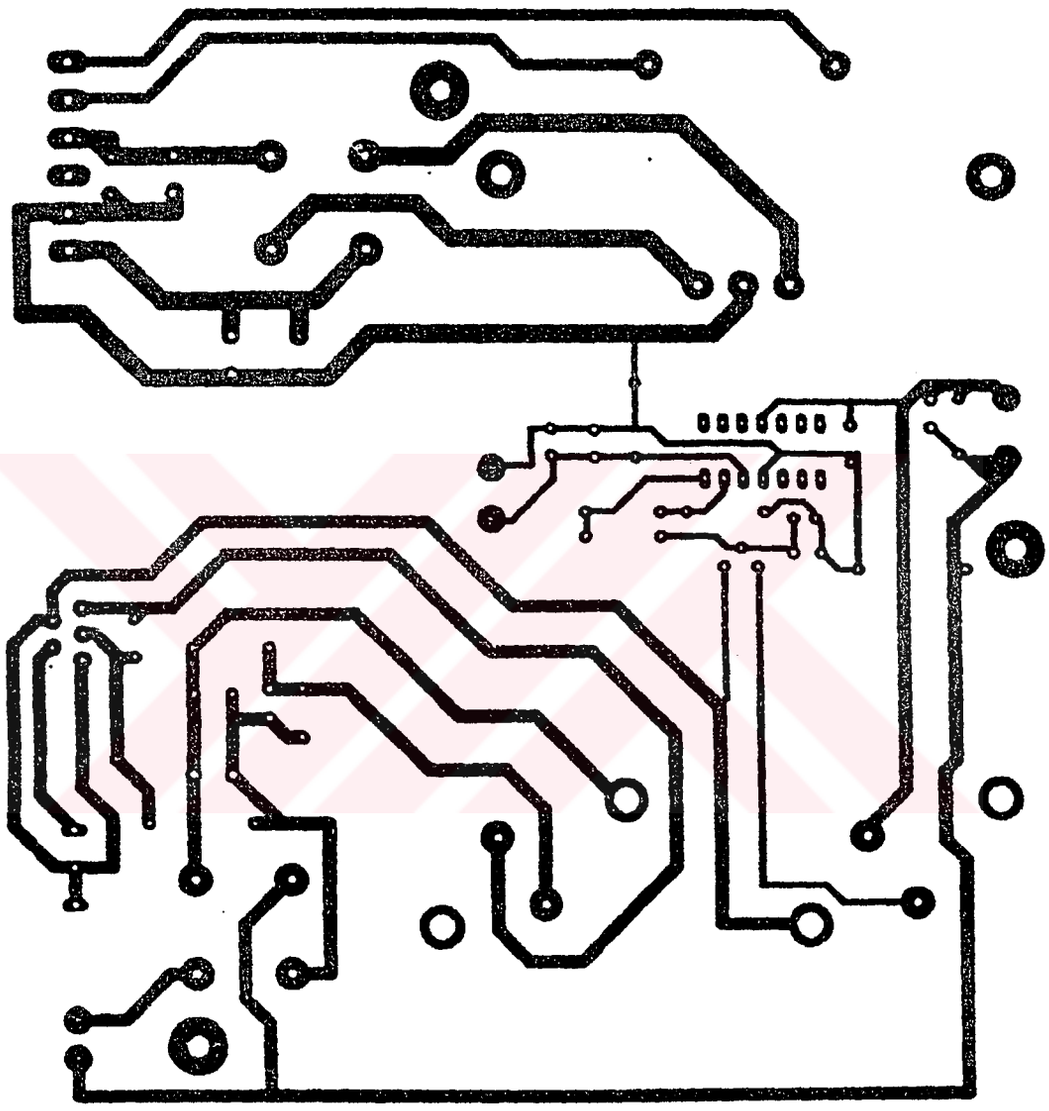
○ 冷却条件

取付フィン 250×250×6 (mm)

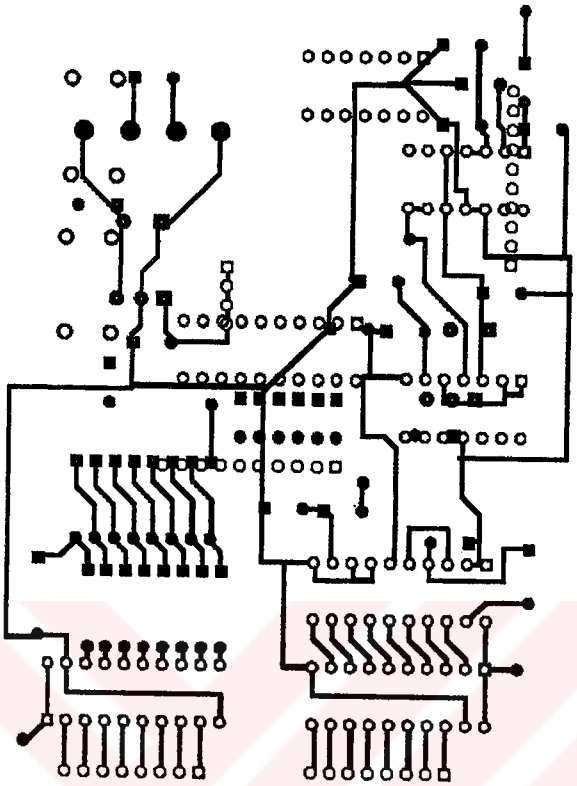
○ 定格値 および トルク一定領域は電機子巻線温度

125℃ その他は 25℃ の時の値です.

820A924



BESLEME KARTI BASKI DEVRESİ



DC MOTOR SÜRME ve HIZ GERİBESLEME DEVRESİ

ÖZGEÇMİŞ

Doğum tarihi	20.09.1971	
Doğum yeri	Batman	
Lise	1986-1989	Batman Lisesi
Lisans	1989-1993	Anadolu Üniversitesi Mühendislik-Mimarlık Fak. (Osmangazi Üniversitesi) Elektrik-Elektronik Mühendisliği Bölümü
Yüksek Lisans	1994-1996	Dumlupınar Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Müh. Anabilim Dalı
Doktora	1997-2001	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektrik Müh. Anabilim Dalı

Çalıştığı Kurumlar

1993-1998	Dumlupınar Üniversitesi Mühendislik Fakültesi Elektrik-Elektronik Müh. Böl. Araştırma Görevlisi
1998-Devam	Eti Holding, Eti Gümüş A.Ş Kütahya Enerji Baş Mühendisliği, Otomasyon Servisi Müh.