YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

PARAMETRELERİ BULANIK MANTIK KULLANILARAK AYARLANAN ORAN VE İNTEGRAL KONTROLÖRÜ İLE DAİMİ MIKNATISLI DOĞRU AKIM MOTORUNUN KALKIŞ ANINDAKİ PERFORMANSININ ARTTIRILMASI

Elek. Yük. Müh. K. Burak DALCI

F.B.E. Elektrik Mühendisliği Anabilim Dalı Programında
Hazırlanan

DOKTORA TEZİ

Tez Savunma Tarihi	:	7 Nisan 2000
Tez Danışmanı	:	Prof. Dr. Halit PASTACI (YTÜ)
Jüri Üyeleri	:	Prof. Dr. Atıf URAL (İÜ)
	:	Prof. Remzi GÜLGÜN (YTÜ)
	:	Prof. Dr. Burhanettin CAN (MÜ)
	:	Yrd. Doç. Dr. Selahattin DİNLER (YTÜ)

İSTANBUL, 2000

İÇİNDEKİLER

KISALTMA LİSTESİ

AC	Alternative Current
DC	Direct Current
HVDC	High Voltage Direct Current
IGBT	Insulated Gate Bipolar Transistor
ISE	Integrated Squarre Error
ITAE	Integrated Time Multiply Absolute Error
LFC	Load Frequency Control
OTP	One Time Programmable
PWM	Pulse Width Modulation
PI	Proportional and Integrale
RISC	Reduced Instructions Set Complex
SSP	Synchronous Serial Port
SCI	Serial Communications Interface
T	Örnekleme Zamanı

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

ÖNSÖZ

Bu tezin hazırlanması sırasında ilgisini ve desteğini esirgemeyen, çalışmalarım boyunca bana sabırla yön veren tez danışmanım Sayın Hocam Prof. Dr. Halit PASTACI'ya teşekkürü bir borç olarak addediyorum.

Çalışmalarımın belli bir noktaya getirilmesinde, gerek bilgi birikimi ve gerekse deneyimleriyle büyük pay sahibi olan değerli hocam Yrd.Doç.Dr. Selahattin DİNLER'e teşekkürlerimi sunuyorum.

Doktora programına başladığımdan beri desteğini hep yanımda hissetiğim değerli yardımlarını esirgemeyen kıymetli arkadaşım Hüseyin GÜNDOĞDU'ya ve bu uzun çalışma boyunca maddi manevi destekleriyle beni her zaman teşvik eden aileme de şükranlarımı sunuyorum.

K. Burak DALCI

ÖZET

Günümüzde, düşük maliyeti ve dayanıklı performansı sebebiyle endüstride en çok kullanılan kontrolörlerden biri oran-integral kontrolörüdür. Tek katlı yapıda bir kontrolör olan oran-integral kontrolörü, sabit parametreleri sebebiyle kontrol işleminin performansı sınırlandırılmıştır. Performansı arttırmak amacıyla, tek katlı kontrolör yapısından iki katlı bir kontrolör yapısına geçilmiştir. Bu iki katlı kontrolör yapısında, birinci katı oran-integral kontrolörü oluştururken, ikinci katı ise bu kontrolörün parametrelerini elde edebilmek için hız hatası ve onun birinci dereceden türevinin değerini gözönünde bulundurarak bulanık işlemleri gerçekleştiren bir bulanık kontrolör oluşturmaktadır.

PI kontrolör, RISC yapıda 8 bit işlem yapan bir kontrolör ile 2kHz örnekleme frekansında gerçekleştirilmiştir. Bulanık işlemler ise fuzzyTECH adı verilen editörde tasarlandıktan sonra assembler diline çevrilerek oran ve integral katsayıları için ayrı ayrı olmak üzere farklı iki RİSC mikrokontrolöre yüklenmiştir. Kontrolörün geri besleme işareti motorun hızı iken, çıkış işareti ise darbe genlik modülasyonudur(PWM). PWM işareti 20kHz darbe frekansında bir doğru akım kıyıcısına uygulanmaktadır. Kontrol edilen sistem ise kalıcı mıknatısiyete sahip bir doğru akım motoru ile mekanik olarak bu motora bağlı bir doğru akım jeneratöründen meydana gelmiştir. Jeneratöre değişik rezistif yükler bağlayarak kontrolörlerin performansı incelenmiştir. Aynı zamanda Borland-C dilinde yazılan program vasıtasıyla hız bilgisi seri haberleşme ile mikrokontrolör üzerinden bilgisayara gönderilmiş ve grafik ekranda izlenmesi mümkün olmuştur.

Mikrokontrolörle geliştirilen sistem, hem Bulanık-PI hem de PI kontrolör olarak çalışacak şekilde tasarlanmıştır. Değişik yükler altında yapılan denemeler sonucunda; Bulanık-PI kontrolörün kontrol kriterleri ve performans açısından PI kontrolörden daha iyi olduğu gözlenmiştir.

Anahtar Kelimeler: Bulanık-PI kontrolör, RISC, mikrokontrolör, PWM

ABSTRACT

Today, the most used controller which has a robust performance and low cost, is proportional and integrale controller. Proportional and integrale controller that has a single level structure, does control operation with limited performance by reason of fixed parameters. To improve the performance, single level control structure must be converted into multi level control structure. In this multi level structure, first level consists of proportional and integrale controller and the second level also includes fuzzy controller that realizes fuzzy operations according to the velocity error and it's first difference to provide the control parameters of the PI controller.

PI controller is designed by a 8 bit RISC microcontroller, operates under 2kHz sampling frequency. All the fuzzy operations is designed by using the editor is called as fuzzyTECH. After the design stage, this fuzzy project was converted to the assembler language of mikrocontroller for proportional and integrale parameters. And this programs were loaded to two separate RISC microcontrollers. While the feedback signal of the controller is velocity of the motor, output signal is the pulse widht modulation (PWM). PWM signal that operates under 20kHz pulse frequency, is sent to the input of the DC chopper. Controlled system consists of permanent magnet DC motor and permanent magnet DC generator is connected to the motor mechanically. Controller performance was analized by loading various loads to the generator. At the same time, velocity data is sent to the computer by microcontroller in serial communication with a Borland-C program which runs in computer to provide hand shaking procedure in computer between microcontroller. And the velocity can be observed in the graphic mode in monitor.

The system is developed by microcontroller, is designed to operate PI controller and Fuzzy-PI controller too. As the results of searchs by using different loads, Fuzzy-PI controller is realized to perform better than PI controller for the performance and control criterias.

Keywords: Fuzzy-PI controller, RISC, microcontroller, PWM

1. GİRİŞ

1.1 Tipografi

Bu tezde farklı amaçlar için aşağıdaki fontlar kullanılmıştır;

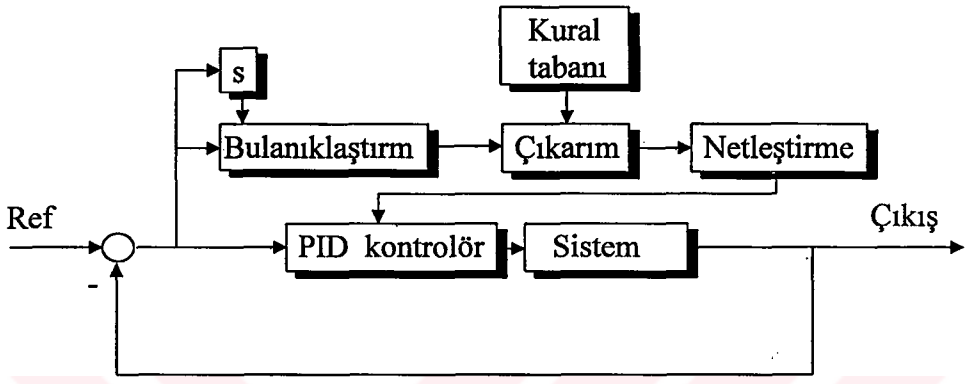
1. Tez metni: "Times New Roman"
2. Vurgulanmak istenen yerler ve başlıklar **kalın harflerle**
3. Konu içindeki alt konuların belirtilmesi gerektiğinde *"Times New Roman (italik)"*.

1.2 Önceki Çalışmalar

Bulanık kontrol, kullanım alanlarının artmasıyla ve mühendislik başta olmak üzere birçok bilimsel alanda, karmaşık ve belirsiz veriler altında üretmiş olduğu çözümler ile günlük hayatımızın bir parçası haline gelmiştir. Bu teknoloji insan beyninin düşünme sistemini taklit etme esası üzerine kurulmuş olup, bilgisayar dünyasına yeni bir bilgi işleme yaklaşımı getirmiştir. Daha birkaç sene önce araştırmalar sadece akademik boyutlarda iken bugün piyasada bulanık mantık sistemleri ve ürünleri görülmektedir.

İlk olarak Pierre Guillemin bir mutfak robotunun ana parçası olan üniversal motorun hızını bulanık kontrol kullanarak kontrol etmiştir(Guillemin, 1996). Bu çalışmada üniversal motor, besinleri parçalama işlemi sırasında değişik yüklere maruz kalmaktadır. Bu değişik yükler altında motorun hızını kontrol edebilmek için bulanık kontrol algoritması kullanılmıştır. Motorun hızı bir tako-generatör üzerinden alınarak SGS-THOMPSON firmasının ST6265 mikroişlemcisine gönderilmiştir. Bu mikroişlemci gelen analog hız bilgisini dijital bir forma dönüştürerek motorun hızını sayısal olarak işleyebilmektedir. Bu hız bilgisi ve birinci dereceden türevi kullanılarak, bulanık kontrol işlemleri yapılmıştır. Daha sonra elde edilen sayısal değer darbe genişlik modülasyonuna (PWM) dönüştürülerek 10A/500V'luk bir IGBT'ye uygulanmaktadır. Motor yüksüz iken kararlı hal hızına 1.3 saniyede ulaşmaktadır. Kararlı hal hızına ulaşılana kadar 40 bulanık çevrim işlemi yapılmaktadır. Motorun hız-zaman eğrileri göz önünde bulundurulduğunda, motorun hızında aşımın istenilen sınırlar içinde kaldığı (<%5) görülmüştür. Motor yüklü iken istenilen referans hıza ulaşması yaklaşık olarak 1 saniye sürmekte olup, aşım meydana gelmemektedir. Yük olarak ise 200W, 110W ve 40W'lık yükler kullanılmıştır. Guilleman'ın yapmış olduğu bu çalışma gerçekleştirilen Bulanık-PI kontrolör çalışmasının yapısının oluşması esnasında

oldukça yararlı birtakım fikirler edinilmesini sağladı. Yapılan çalışmanın anafikri ise, Tomizuka, Zhao ve Isaka'nın yapmış olduğu **Bulanık kontrolör ile PID kontrolörü birleştiren çalışması olmuştur**(Tomizuka, Zhao ve Isaka, 1993). Bu çalışmada iki katlı kontrol yapısı kullanılmıştır. İlk kat sistemin doğrudan kontrolünü yapan PID kontrolördür. İkinci kat ise, PID kontrolörün oran, türev ve integral katsayılarını, hata ve hatanın değişiminin değerine göre değiştiren bulanık kontrolördür. Şekil 1.1 'de bu yapının blok diyagramı verilmiştir.



Şekil 1.1 Bulanık-PID kontrolörün blok diyagramı

Sistem olarak ikinci, üçüncü ve dördüncü mertebeden sistemler ele alınmıştır. Simülasyon çalışmaları bu sistemler üzerinde sistemin vermiş olduğu çıkış işaretinin performansının yükseltilmesi üzerine yapılmıştır. Bu çalışmada bulanık kontrol için giriş bilgisi olan hata işareti ve hata işaretinin türevi, üyelik fonksiyonunda yedi terim ile tanımlanmıştır. Çıkış işareti olan oran ve integral katsayıları *büyük* ve *küçük* olmak üzere iki terimden meydana gelmiştir. Bulanık kontrolün karar mekanizması olan kural tabanı ise 49 kuraldan meydana getirilmiştir. Kural tabanındaki kurallar yazılırken ikinci dereceden bir sistemin cevabı gözönüne alınmıştır. Oran, integral ve türev katsayıları sistemin davranışının incelenmesi ile elde edilen deneyimlerle ayarlanmıştır. Gerçekleştirilen Bulanık-PID kontrolör Ziegler-Nichols ve Kitamori'nin PID kontrolörleri ile karşılaştırılmıştır. Sonuç aşağıdaki Çizelge 1.1 (Zhao ve Tomizuka, 1993)'de verilmiştir.

Çizelge 1.1 Kontrolörlerin performanslarının karşılaştırılması

Kriterler	Ziegler-Nichols PID Kontrolörü	Kitamori'nin PID Kontrolörü	Bulanık-PID Kontrolör
Aşım	%32	%6.8	%6
Yerleşme zamanı	4.16 sn	2.37 sn	3.09 sn
ITAE	1.37	1.04	1.18
ISE	0.871	0.805	0.772

Bu karşılaştırmanın sonucunda Bulanık-PID'de aşımın küçük olması dolayısıyla ISE kriterinin de küçük olmasına sebep olmuştur. Oturma zamanı ise Kitamori'nin PID'sine göre daha uzun sürmüştür. Bu sebeple ITAE kriteri daha büyüktür. Fakat Kitamori'nin PID kontrolörü ile Bulanık-PID kontrolörünün, Ziegler-Nichols'ün PID kontrolöründen daha iyi performans gösterdiği ortaya çıkmıştır.

Tomizuka, Zhao ve Isaka'nın yapmış olduğu bu çalışma birçok çalışmaya öncülük etmiştir. Bunlar daha çok **simülasyon seviyesinde** yapılmış çalışmalardır. Bulanık-PI kontrolörün MATLAB ortamında uygulama imkanı bulan bir çalışma da **iki bölgeli bir güç sisteminde yük-frekans kontrolüdür**(Akalın ve Kocaarslan, 1998). Güç sistemlerinin kalitesinin belirlenmesinde sabit frekans önemli bir faktördür. Sistem frekansındaki dalgalanmalar belirli sınırlar içerisinde tutulması gerekir. Tüketici talebindeki öngörülmeyen değişimler sistem frekansında bozulmalara sebep olacaktır. Yük-frekans kontrol sistemi bu değişimleri belirlemeli ve mümkün olduğunca hızlı ve etkin bir biçimde bu bozulmaları yoketmelidir. Sistem frekansındaki değişiklikleri ve bağlantı hattı yükünü, arzu edilen sistem frekansı ve komşu şebekelerle önceden planlanan güç alışverişi değerine ayarlamak üzere jeneratörlerin aktif güç çıkışının kontrolü problemi, otomatik üretim kontrolü veya yük-frekans kontrolü (LFC) olarak tanımlanır. Frekans bilgisi, geribesleme bilgisi olarak alınır. Bulanık-PI kontrolör doğrudan jeneratörün hızını ayarlayan hız-regülatörüne uygulanmıştır. Elde edilen sonuçlar aşağıda verilmiştir.

Çizelge 1.2 Yük-frekans kontrolünde Bulanık-PI ve PI kontrolörün performansının karşılaştırılması

	Oturma zamanı	Max. kayma
Bulanık-PI	14sn	0.020Hz
PI	18sn	0.022Hz

Çizelge 1.2(Akalın ve Kocaarslan, 1998)'de görüldüğü üzere amaçlanan PI kontrolör ile gerçekleştirilen yük-frekans kontrolü, gerek aşırı yükselmeler ve gerekse oturma zamanı açısından daha uygun ve hızlı cevap vermektedir.

Yapılan diğer bir çalışma ise **yüksek gerilimde doğru akım güç iletimidir**(HVDC). Bu çalışmada alternatif gerilimi DC yüksek gerilime çeviren tristörlerden oluşan güç elektroniği sisteminin ateşleme açılarını ve sistemin çekmiş olduğu akımı ayarlayan kontrol sistemi, Bulanık-PI kontrolör ile tasarlanmıştır (Gole ve Chapman, 1997). Bu yaklaşım elektromagnetik geçici hal

simülasyon programında simüle edilerek, PI kontrolör ile karşılaştırılmıştır. Kullanılan yöntem ile geçici hal cevabında iyileştirmeler sağlanmıştır. Sonuçlarda, bulanık yaklaşım sayesinde daha az aşım meydana geldiği görülmüştür.

Bulanık-PI kontrolör aynı zamanda Dong Yun Kim ve Hyun Seong tarafından **reaktör kontrolünde** kullanılmıştır (Dong Yun Kim ve Hyun Seong, 1997). Hazırlanan algoritmada, adım adım değişen bir deneme yöntemi kullanılarak öğrenme özelliğine sahip bulanık bir kural tabanı oluşturulmuştur. Kural tabanı amaca ait objektif fonksiyonun minimize edilmesi ile elde edilmiştir. Burada sözkonusu olan objektif fonksiyon maliyet-performans fonksiyonudur. Akıllı, öğrenebilen algoritma ile kazançlar ayarlanarak sistemin hata değerinin minimize edilmesi amaçlanmıştır. Hazırlanan algoritma, akıllı öğrenme fonksiyonundan elde edilen bulanık kurallar ile performansı arttırırken, zamandan da tasarruf edilmesini sağlamaktadır. Hazırlanan kontrolör nükleer bir güç sistemine uygulanmıştır. Elde edilen sonuçlar kazancı sabit bir PI kontrolör ile karşılaştırılmıştır. Karşılaştırma işlemi için sistem bir simülasyon programında daha önceden elde edilen matematiksel model vasıtasıyla simüle edilmiştir.

Sistem şartları değiştirildiğinde hazırlanan kontrol algoritmasının oldukça verimli olduğu görülmüştür. Bu kontrol algoritması, sistem şartlarının sistemi bozucu yönde etkilemesine rağmen hatayı minimize edecek şekilde PI kontrolörün kazançlarını ayarlamaktadır. Gerçekleştirilen Bulanık-PI kontrolörün, normal PI kontrolöre oranla daha iyi bir performans sergilediği görülmüştür.

Bulanık kontrolör, PID kontrolör ile birleştirilerek endüstriyel bir fırının kontrolünde kullanılmıştır(Ketata ve Degeest, 1995). Bu çalışmada kapalı çevrim performansı arttırılmak istenmiştir.

1.3. Amaç

Daha önce yapılan çalışmalar üzerine gerçekleştirilen literatür taraması sonucunda ortaya konan çalışmaların tamamının **simülasyon aşamasında** yapılmış olan çalışmalar olduğu görülmüştür. Yapılan çalışmada iki amaç öngörülmüştür. Bunlar sırasıyla;

a) Yapılan çalışmanın endüstriyel alanda kullanılabilecek bir tasarım olmasıdır. Yani maliyeti düşük, rahatlıkla bulunabilecek malzemeler kullanılması ve hızının diğer endüstriyel sistemlerde kullanılmaya uygun kapasiteye sahip olmasıdır.

b) Aynı zamanda normal PI kontrolöre oranla performans bakımından çok daha gelişmiş olan bir kontrolör elde edilmesidir.

İki kontrolörü performans açısından karşılaştırırken sırasıyla, oturma zamanı (t_s), yükselme zamanı (t_r), aşım ve performans kriterleri bakımından test edilecektir. Sözü edilen bu kriterler ISE (Integrated Square Error) ve ITAE (Integrated Time Area Multiply Error) olarak adlandırılır. Bu Bulanık-PI kontrolörü ve PI kontrolörü karşılaştırmak için kullanacağımız sistem, bir motor, yani tahrik sistemi ile ona mekanik olarak bağlı olan bir jeneratörden meydana gelmektedir.

1.4. Yöntem

Geleneksel bir kontrolör olan PI kontrolörün sistemi kontrol etmedeki başarısı sistemin transfer fonksiyonuna göre belirlenecek olan kontrol parametrelerinin seçimine bağlıdır. Seçilen sabit parametreler sistemin performansını sınırlandırmaktadır. Performansı arttırabilmek için sabit kontrolör parametreleri yerine sistemi kararsızlığa sürüklemeyecek, değişken parametreler kullanılmalıdır. Böylece daha başarılı bir kontrolör elde edilmiş olur. Böyle bir kontrolör iki kattan meydana gelmektedir. Bu katlardan birincisi normal PI kontrolör işlevini yerine getirirken, ikinci kat ise hata ve hatanın birinci dereceden türevini izleyerek kontrolör parametrelerini değiştirmektedir. Yapılan çalışmada ikinci kat, bulanık bir kontrolörden meydana gelmektedir. PI kontrolör ve bulanık kontrolör RISC (Reduced Instructions Set Complex) mimariye sahip bir mikrokontrolör ile gerçekleştirilmiştir. Bu mikrokontrolörün seçilmesi komutların sayısının azlığı ve basitliği yüzünden programın yazılmasını güçleştirirken, hızının yüksek olması sebebiyle işlemleri hızlandırmıştır. Sistemin çıkış cevabı olan hız bilgisi, mikrokontrolör tarafından seri port üzerinden bilgisayara aktarılmıştır. Borland-C dilinde yazılan bir program ile bu bilgiler ekranda grafik modunda çizilmiştir. Bu haberleşme işlemi 38400 baud rate ile gerçekleştirilmiştir. Fakat normal derleyiciler bu hıza çıkılmasına izin vermediği için bilgisayarın seri haberleşme çipi 8250'ye Assembler dilinde yazılan bir altprogram ile müdahele edilerek hızı arttırılmıştır.

Kontrolörün performansını izleyebilmek için, hızı kontrol edilen motor kendine mekanik olarak bağlı olan jeneratör vasıtasıyla yüklenerek, değişik yükler altında kaldırılmıştır. Bu kalkış anında elde edilen hız bilgileri sürekli olarak bir text dosyasına yazılarak Excel'de gerekli işlemlere tabi tutulmuştur.

1.5 Tezin Bölümleri

Tezin bölümlere dağılışı ana hatlarıyla aşağıdaki gibidir.

1.5.1 Giriş

Daha önce yapılan çalışmalar ve bu çalışmaların sonuçları ile tez çalışmasının tanıtımı yapılmıştır.

1.5.2. Doğru akım makinaları

Bu bölümde kontrol edilen sistemi meydana getiren doğru akım makinaları tanıtılmıştır. Doğru akım motorunun çalışma prensipleri verilmiş ve formüllerle desteklenmiştir. Aynı zamanda motorun kalkış anında maruz kaldığı durumlar ve yolverme yöntemleri anlatılmıştır. Motorun frenlenmesi sırasında meydana gelen olaylar ve frenleme metotlarından bahsedilmiştir. Bu bölümün son kısmında ise motorun elektriksel analizi yapılarak sistemin matematiksel modeli elde edilmiş ve blok diyagramı bu modele göre çizilmiştir.

1.5.3 Dijital-PI kontrolör

Bu bölümde çalışmamızdaki ana unsurlardan biri olan dijital kontrolörlerin genel bir tanıtımı yapılmıştır. Dijital kontrolörler ile analog kontrolörler karşılaştırılmış ve dijital kontrolörlerin kullanılmasına sebep olan üstünlükler belirtilmiştir. Çalışmamızda kullanılan PI kontrolörün yapısı incelenmiş ve transfer fonksiyonu verilmiştir. PI kontrolörde integral işleminin nasıl yapıldığını açıklamak üzere nümerik integrasyon teknikleri açıklanmıştır. Son kısmında ise PI kontrolör parametrelerinin, kutup iptal yöntemi ile nasıl seçilmesi gerektiği anlatılmış ve bununla ilgili MAT_LAB programı verilmiştir. Bütün bunlara ek olarak, kontrolörün parametreleri olan oran ve integral katsayılarının değişik değerleri altında sistemin cevabı incelenmiş olup, MAT_LAB ortamında simüle edilmiştir.

1.5.4 Bulanık mantık

Bu kısımda bulanık mantığın felsefesi tanıtılmıştır. Bulanık kümeler, bu kümelerle yapılan işlemler ve özellikleri anlatılmıştır. Sayısal değerlerin bulanıklaştırılması anlatılmış ve

örneklerle pekiştirilmiştir. Bulanıklaştırılan bu değerler üzerinde yapılan çıkarım işlemi ve kural tabanı konuları açıklanmıştır. Aynı zamanda çıkarım olayının grafiksel çözüm tekniği verilmiştir. Bölümün sonunda ise en çok kullanılan netleştirme yöntemlerine değinilmiştir.

1.5.5 PIC mikrokontrolörü

Bu bölümde yapılan çalışmada kullanılan; bulanık ve PI kontrol algoritmalarının çalıştırıldığı PIC mikrokontrolörü tanıtılmıştır. Sırasıyla, PIC mikrokontrolörünün genel ve mimari yapıları anlatılmış ve çalışmada da kullanılan bir takım özel modlar açıklanmıştır.

1.5.6 Bulanık-PI kontrolör ile DC motorun kalkış anındaki performansının arttırılması

Bu bölümde Bulanık-PI kontrolörü gerçekleştirmek için kullandığımız PIC mikrokontrolörü ve kullanılan sistemin genel yapısı anlatılmıştır. Tezimizi gerçekleştirirken kullanılan algoritmalar hakkında bilgi verilmiştir.

İlerleyen kısımlarda PI kontrolörün mikrokontrolör ile nasıl gerçekleştirildiği ve ilgili programın akış şemaları verilmiştir. Daha sonra Bulanık-PI kontrolörün oluşturulması, üyelik fonksiyonları, kural tabanı verilmiş olup kullanılan yöntemler belirtilmiştir.

1.5.7 Sonuçlar ve tartışma

Bu bölümde çalışmadan elde edilen sonuçların tartışılması yapılmış ve ileriye dönük öneriler verilmiştir.

2. DOĞRU AKIM MAKİNALARI

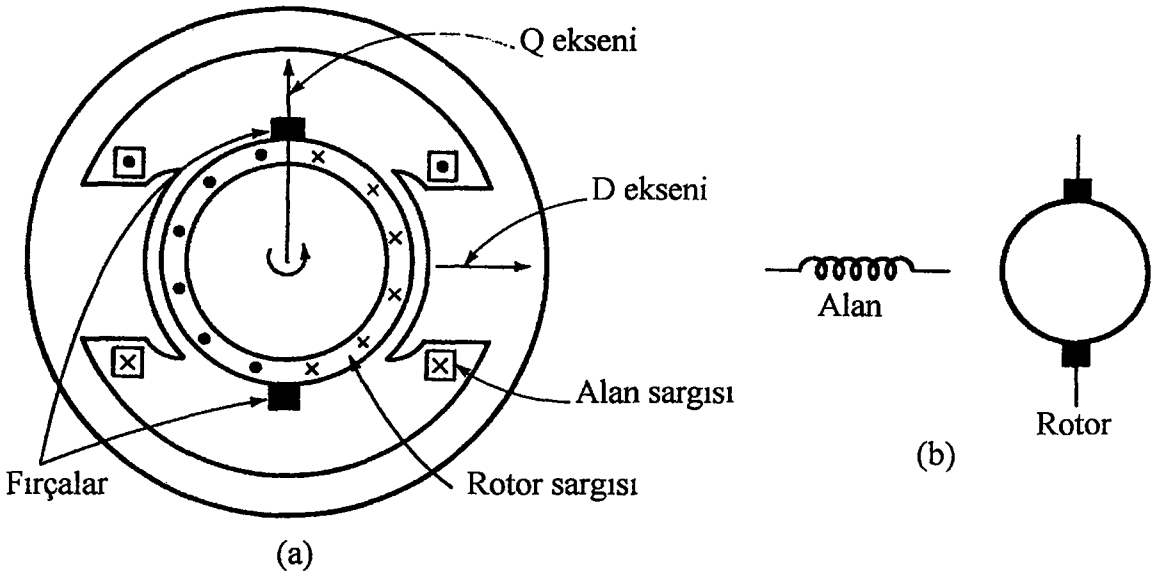
Doğru akım makinaları çok yönlü kullanım alanları ile tanınırlar. Sözü edilen çok yönlü kullanım, dinamik ve kararlı hallerin her ikisi için de, motorun geniş bir alanda değişen **hız-moment** veya **akım-gerilim** karakteristiklerini göstermesidir. Bu karakteristiklerin bu kadar geniş bir aralıkta değişmesinin sebebi; şönt, seri veya serbest gibi çeşitli uyarma sargılarının yapısında bulunmasıdır. Aynı zamanda doğru akım motorlarının kontrolunun da kolay olmasından dolayı, motor hızının geniş bir değişim aralığı göstermesi istenen yerlerde, sık sık doğru akım makinaları kullanılır. Son yıllarda yarıiletken asenkron motor sürücü sistemlerindeki teknolojilerin oldukça gelişmesi sonucunda, uygulamalarda daha önce doğru akım makinalarıyla birlikte kullanılırken, artık alternatif akım makinaları hemen hemen yalnız başına uygulama alanı bulmaya başlamıştır. Bununla birlikte yine de AC makinalarla karşılaştırıldığında esnekliği ve bununla bağlantılı olan sürme sisteminin basitliği yüzünden doğru akım makinaları uygulama alanı bulmaya devam edecektir.

2.1 Giriş

Doğru akım makinasının temel yapısı Şekil 2.1'de şematik olarak gösterilmiştir. Çıkık kutuplara sahip stator, bir çift veya daha fazla alan sargısından meydana gelmektedir. Stator ile rotor arasında yer alan hava aralığı akısı, statorda bulunan uyarma sargıları tarafından meydana getirilir. Bu uyarma kutupları arasındaki eksene alan ekseni veya D ekseni adı verilir.

Dönen her rotor sargısında endüklenen alternatif gerilim, rotor uçlarına bağlı sabit fırçalar ile hareket eden komütatörden oluşan rotorun dış uçları vasıtasıyla doğru gerilime çevrilir. Komütator ve fırçadan oluşan kombinasyon, mekanik bir doğrultucu gibi çalışarak rotor üzerinde magnetomotor kuvvetinin endüklemiş olduğu rotor geriliminin elde edilmesini sağlar. Fırçalar D eksenine dik olarak komütasyon meydana getirecek şekilde yerleştirilmiştir. Bu eksene Q ekseni adı verilir. Q ekseninde, rotor magnetomotor kuvvet dalgasının ekseni, elektriki olarak alan kutuplarının ekseninden 90 derece geridedir. Şekil 2.1a'da şematik gösterimde fırçalar Q ekseninde gösterilmiştir.

Manyetik moment ve hıza bağlı oluşan gerilim, fırçaların uçlarında görünmesine rağmen bunlar akı dağılımının uzaydaki dalga formuna oldukça bağlıdır.



Şekil 2.1 Doğru akım motorunun (a) şematik (b) sembolik gösterimi

Rotor magnetomotor kuvvetinin uzaydaki temel bileşeni F_{a1} ve D ekseninde kutup başına düşen hava aralığı akısı ϕ_d'nin kesişmesi, moment olarak tanımlanır. Moment 2.1'deki (Fitzgerald ve Kingsley, 1992) eşitlik ile ifade edilir. Q eksenindeki fırçalar alan sargılarının meydana getirdiği alanla 90 derecelik bir açı meydana getirir. P kutuplu makine için moment ifadesi yazılırsa;

$$T = \frac{\pi}{2}\frac{(P)^2}{2}\phi_d F_{a1} \tag{2.1}$$

şeklinde elde edilir. 2.2 (Fitzgerald ve Kingsley, 1992) eşitliğinde üçgen dalga formunda olan rotor magnetomotor kuvvetinin tepe değeri 2.1 eşitliğinde yerine konulmuştur. F_{a1} bu tepe değerin $8/\pi^2$ katı bir değere sahiptir. 2.1 eşitliğinde F_{a1} yerine konacak olursa

$$T = \frac{PC_a}{2\pi\, m}\phi_d i_a = K_a \phi_d i_a \tag{2.2}$$

olarak bulunur. Burada;

i_a = dış rotor devresindeki akımı

C_a = rotor sargısındaki toplam iletken sayısı

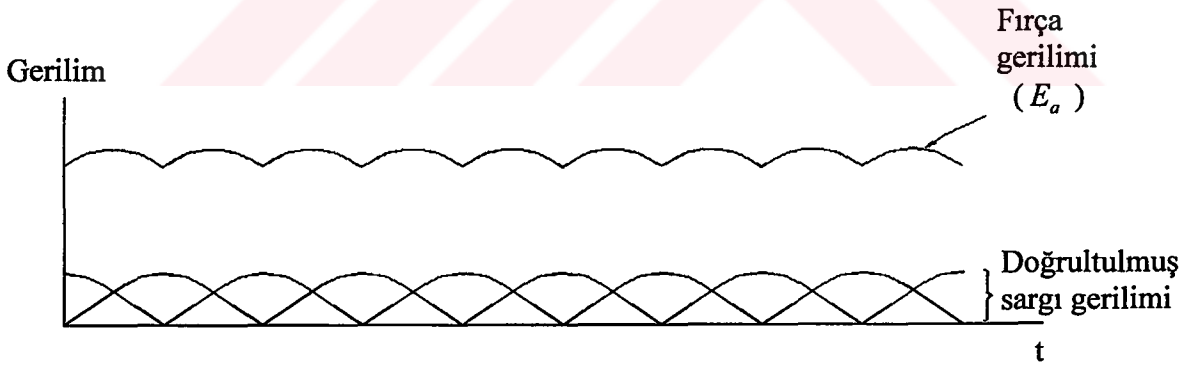
m = bobindeki sarım sayısını göstermektedir.

$$K_a = \frac{PC_a}{2\pi\ m} \tag{2.3}$$

eşitliği ile ifade edilen K_a, sargının tasarımına göre belirlenmiş bir sabittir. Tek sargılı basit bir rotor için üretilen doğrultulmuş gerilim, tam dalga doğrultulmuş bir AC gerilim dalgasına benzer. Şekil 2.2'de görülen çok sayıda sargının, rotor üzerindeki yarıklara dağıtılması ile sargı yanları nötral bölgeye geldiği zaman, her sargıdaki üretilen gerilim komütasyon sayesinde doğrultulmuş sinüs dalga formunu alır. Üretilen gerilim, fırçalardan bütün sargılardaki doğrultulmuş gerilimlerin toplamı olarak izlenir ve Şekil 2.2'de (Fitzgerald ve Kingsley, 1992) dalgalanan çizgi E_a olarak adlandırılır. Kutup başına düşen komütator dilimleri arttırılacak olursa dalgalanma oldukça küçülecek ve üretilen ortalama gerilim fırçalardan bakıldığında doğrultulmuş sargı gerilimlerinin ortalamasına eşit olacaktır. Fırçalardaki E_a doğrultulmuş gerilimi aynı zamanda hız gerilimi olarak bilinir ve formüle edilirse,

$$E_a = \frac{PC_a}{2\pi\ m}\phi_d\omega_m = K_a\phi_d\omega_m \tag{2.4}$$

elde edilir. Burada K_a sargı sabiti olarak adlandırılır ve eşitlik 2.3'te tanımlanmıştır.

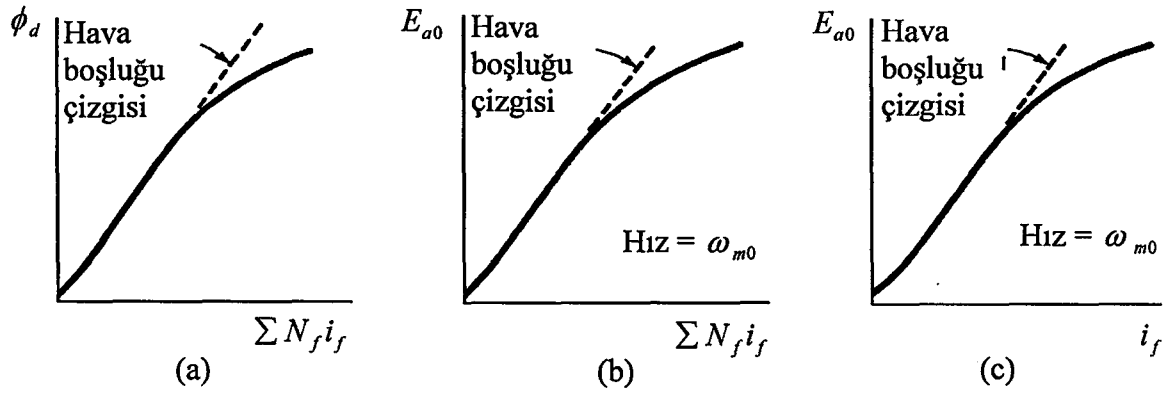


Şekil 2.2 Doğrultulmuş sargı gerilimi ve doğru akım makinasındaki fırçalar arasındaki çıkış gerilimi

Dağıtılmış sargılardaki ortalama doğrultulmuş gerilim, yoğunlaştırılmış sargılardaki gerilime eşit olmasına rağmen gerilim dalgalanması oldukça azaltılmıştır. 2.2 ve 2.4 eşitliklerinden bütün değişkenler SI biriminde kabul edilecek olursa,

$$E_a i_a = T\omega_m \tag{2.5}$$

olarak elde edilir. Bu eşitlik hız gerilimi ile bağlantılı olan ani elektriksel gücün, manyetik moment ile orantılı olan ani mekanik güce eşit olduğunu gösterir. D ekseni hava aralığı akısı, alan sargılarının birleştirilmiş magnetomotor kuvveti $N_f i_f$ ile üretilir. Makinanın demirden oluşan özel geometrik yapısına bağlı olan mıknatıslanma eğrisi, akı-magnetomotor kuvvet karakteristiğini oluşturur. Bu eğri Şekil 2.3a,b ve c'de (Fitzgerald ve Kingsley, 1992) verilmiştir.



Şekil 2.3 Doğru akım makinasının mıknatıslanma eğrileri

D eksenindeki akı ile magnetomotor kuvvet arasındaki ilişki Şekil 2.3a'da gösterilmiştir. Bu sabit hızda ω_{m0} rotor elektromotor kuvveti E_{a0}'a dayanarak mıknatıslanma eğrisini ifade etmek daha uygundur. Şekil 2.3b'de bu ilişki verilmiştir. Herhangi bir ω_m hızındaki akı için verilen E_a gerilimi 2.6 eşitliğinde görüldüğü üzere hızla orantılıdır.

$$\frac{E_a}{\omega_m} = K_a \phi_d = \frac{E_{a0}}{\omega_{m0}} \tag{2.6}$$

veya

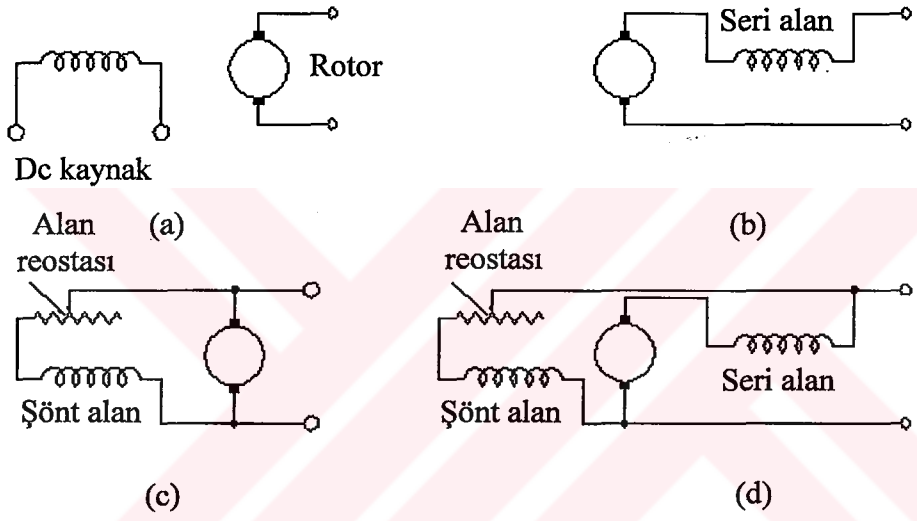
$$E_a = \frac{\omega_m}{\omega_{m0}} E_{a0} \tag{2.7}$$

Şekil 2.3c'de görülen mıknatıslanma eğrisi sadece tek bir alan uyarı sargısı için çizilmiştir. Bu eğri test metodları ile hiçbir tasarım bilgisine gerek olmaksızın kolaylıkla elde edilebilir. Hava aralığı ile karşılaştırıldığında, demirin geniş bir uyarı aralığında relüktansını ihmal edebiliriz. Bu bölgedeki akı, alan sargılarının toplam magnetomotor kuvvetiyle lineer bir şekilde

orantılıdır. Bu oranın sabit değeri P_d, D ekseni hava aralığı magnetik iletkenliğini verir. Böylece,

$$\phi_d = \mathrm{P}_d \Sigma N_f i_f \tag{2.8}$$

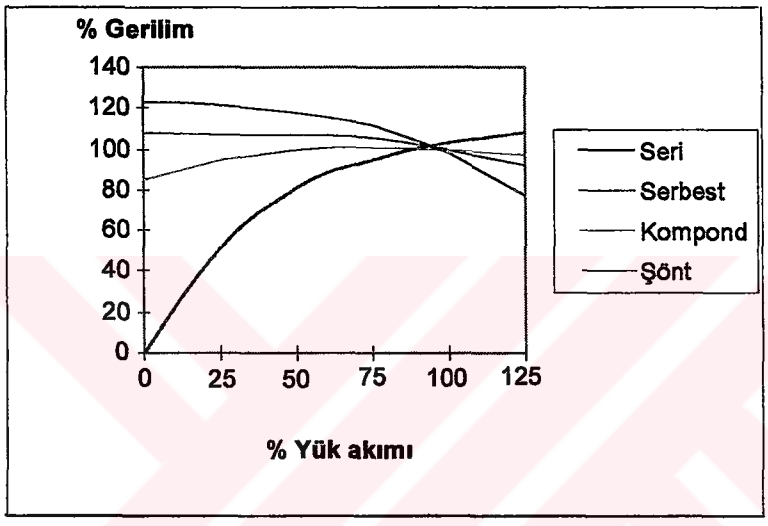
elde edilir. Bu orjinden geçen kesikli çizgiden oluşan hava boşluğu çizgisi, düz çizgiden oluşan mıknatıslanma eğrisinin doğrusal kısmı ile çakışmaktadır. Şekil 2.3'te görülen bu çizgi hava aralığı hattı olarak adlandırılır.



Şekil 2.4 Doğru akım makinasının alan devresi bağlantıları (a) serbest (b) seri, (c) şönt, (d) kompound uyarma

Doğru akım makinasının göze çarpan en büyük üstünlüğü uyarı sargısının seçimine göre çok geniş alanda değişim gösteren işletme şartlarına sahip olmasıdır. Uyarı sargıları dışardan bir kaynak tarafından beslenerek **serbest uyarmalı** olabileceği gibi, makina kendi uyarma sargısını besleyerek **kendinden uyarmalı** olarak da kullanılabilir. Şekil 2.4a'da serbest uyartımlı b,c ve d'de ise kendinden uyartımlı bağlantı diyagramları gösterilmiştir. Uyarma metodunun seçimi, kontrol sistemi içerisinde motorun kararlı hal ve dinamik davranışı üzerinde oldukça etkin bir rol oynar. Şekil 2.4a'da gösterilen kendinden uyarmalı motorda çekilen alan akımı, rotor akımı ile karşılaştırıldığında son derece küçük olduğu görülür. Bu akım nominal akımın ortalama olarak %1-3'ü arasında değişim gösterir. Alan devresindeki bu küçük güç, rotor devresindeki büyük gücü kontrol etmektedir.

Kendinden uyarmalı motorlarda alan sargıları üç değişik yolla sağlanabilir. Alan sargısının rotor sargısına seri bağlanması sonucunda Şekil 2.4b'de görülen seri motor elde edilir. Alan sargısının rotor sargısına paralel bağlanması ile Şekil 2.4c'de görülen şönt motor elde edilirken her iki sargının kullanılmasıyla hem şönt hem de seri sargıya sahip olan ve Şekil 2.4d'de görülen kompound motor elde edilir. Kendinden uyarmalı motor ve jeneratörlerde kendiliğinden uyarmanın başlatılabilmesi için kalıcı mıknatıslanmanın demir nüve içinde olması gerekmektedir. Şekil 2.3'te akı ve gerilim değerlerinin alan akımı sıfır iken sıfır değerini almamasından kalıcı mıknatislanmanın etkisi açık bir şekilde görülmektedir. Şekil 2.5(Fitzgerald ve Kingsley, 1992)'te jeneratörün kararlı hal akım-gerilim karakteristikleri sabit hızda hareket ettiği varsayılarak gösterilmiştir.



Şekil 2.5 Doğru akım jeneratörünün akım-gerilim karakteristikleri

Kararlı halde üretilen gerilim E_a elektromotor kuvveti ile uç gerilimi U arasındaki bağıntı,

$$U = E_a - I_a R_a \tag{2.9}$$

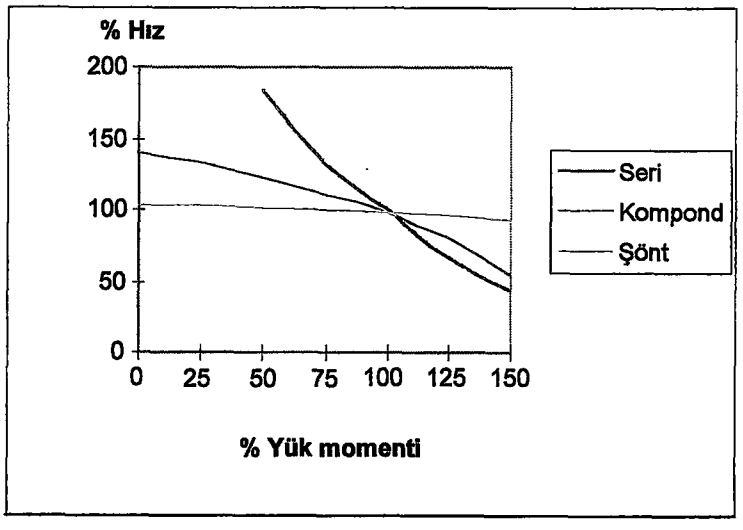
olarak tanımlanır. Bu eşitlikte I_a rotor akımını ve R_a rotor devresi direncini ifade eder. Jeneratör çalışma modunda E_a gerilimi U geriliminden daha büyüktür ve elektromagnetik moment T, dönme yönüne ters yönlü bir momenttir. Kendinden uyarmalı jeneratörde uç gerilimi, yük akımının artmasıyla rotor direnci üzerinde oluşan gerilim düşümünün artması ile azalır. Seri uyarmalı jeneratördeki alan akımı yük akımı ile aynı olduğu için hava aralığı akısı ve gerilim geniş bir alanda değişim gösterir. Sonuç olarak seri jeneratörler bu özelliklerinden dolayı çok sık kullanılmazlar. Şönt jeneratörlerde ise akımla birlikte gerilim düşümü olmakla beraber bu çok küçük bir değerde kalmaktadır. Kompound jeneratörde seri sargı kutup başına

düşen akı, yükle beraber artmaktadır ve yükün artmasına rağmen uç gerilimi yaklaşık olarak sabit bir değerde tutulur. Şönt sargı, ince kesitli ve çok sarımlı bobinlerden oluşmaktadır. Seri sargı ise makinanın yük akımını taşımak zorunda olduğu için kalın iletkenden ve az sarımlıdır. Hem şönt hem de kompound jeneratörde gerilim, şönt alan sargısına bağlı bir reosta ile belirli sınırlar içerisinde kontrol edilir. Jeneratörler için kullanılan uyarma metotları aynı zamanda motorlar içinde kullanılabilir. Motor uçlarının sabit gerilimle beslendiği kabul edilirse Şekil 2.6(Fitzgerald ve Kingsley, 1992)'da hız-moment karakteristikleri gösterilmiştir. Motorda ise rotorda üretilen elektromotor kuvveti E_a ve uç gerilimi U arasındaki bağıntı,

$$U = E_a + I_a R_a \tag{2.10}$$

$$I_a = \frac{U - E_a}{R_a} \tag{2.11}$$

olarak tanımlanır. Burada I_a rotor akımıdır. Jeneratör çalışmadakinin aksine üretilen elektromotor kuvveti, motor çalışmada uç gerilimden küçüktür ve akım ters yönde akmaktadır. Şönt ve serbest uyartımlı motorlarda alan akısı yaklaşık olarak sabittir. Sonuç olarak, artan moment yakın bir oranla akımın artmasına sebep olur. Bu nedenle akımın artması, rotor direnci üzerinde küçük bir gerilim düşümü oluşturarak zıt elektromotor kuvvetin azalmasına sebep olur. Zıt elektromotor kuvvet, akı ve hıza bağlı olduğu için bu hızın biraz azalmasına sebep olacaktır. Şönt motor, sincap kafesli indüksiyon motoru gibi yüksüz durumdan tam yük durumuna geçmesi ile hızında yaklaşık %5 kadar bir azalma meydana gelir.



Şekil 2.6 Doğru akım makinasının hız-moment karakteristikleri

Bu durum Şekil 2.6'da sarı renkli eğri ile gösterilmiştir. Kalkış momenti ve maksimum moment rotor akımıyla, bu da komutasyon olayının başarımıyla sınırlıdır. Şönt motorun en büyük avantajı hız kontrolünün kolay yapılmasıdır. Şönt alan sargısına bağlı bir reosta vasıtasıyla alan akımı ve kutup başına düşen akı ayarlanabilir. Akıdaki bu değişim zıt elektromotor kuvvetini uç gerilimine yakın tutabilmek için hızda ters yönde bir değişim olarak görülür. Maksimum hız bu methotla nominal hızın 4 veya 5 katına kadar arttırılabilinir ve sınırlar komütasyon şartları ile belirlenir. Rotor geriliminin değiştirilmesi ile hız geniş bir aralıkta ayarlanabilir (Chapman, 1985).

Seri motorda yükün artmasıyla, rotor akımı magnetomotor kuvvet ve stator alan akısı (demir nüvenin tamamen doyması engellenirse) birlikte artacaktır. Çünkü hız, akının yükle birlikte artmasıyla, uç gerilim ile zıt emk arasındaki dengeyi koruyabilmek için düşecek üstelik momentin artmasıyla akım artacaktır. Bu yüzden Şekil 2.6'da gösterildiği üzere seri motor hız-yük karakteristiğinde göze çarpan bir düşüm olmaktadır. Yüksek momente sahip aşırı yüklerin olduğu uygulamalarda bu karakteristik aşırı yüke cevap verebilmesi hız düşümüyle birleştirildiğinde oldukça makul görünmektedir. Kalkış karakteristiği akının artması ile artan rotor akımı yüzünden oldukça uygundur.

Kompound motorda seri alan gittikçe artacak şekilde bağlanarak bunun magnetomotor kuvveti şönt alanla birleştirilir veya zıt yönde bağlanabilir. Zıt yönde yapılan bağlantı son derece az kullanılır. Şekil 2.6'da görülen pembe çizgi ile çizilen eğri kompound motora aittir. İlavelerle genişleyen şekilde bağlantıları yapılan kompound motorun hız-yük karakteristikleri şönt ve seri motor arasında olup, yüke bağlı hız düşümü şönt ve seri alan sargılarındaki akım dönüş sayısı ile orantılıdır. Seri motorda çok hafif yüklerde meydana gelen hızın aşırı artması durumu (amble olması) bu tip motorda kabul edilebilir bir seviyede tutulmuştur.

Doğru akım makinasının uygulamadaki avantajı, alan sargılarının şönt, seri veya kompound bağlanması ile değişik karakteristiklerin elde edilebilmesinden ileri gelmektedir (Fitzgerald ve Kingsley, 1992).

2.2 Doğru Akım Motorunun Kalkışı

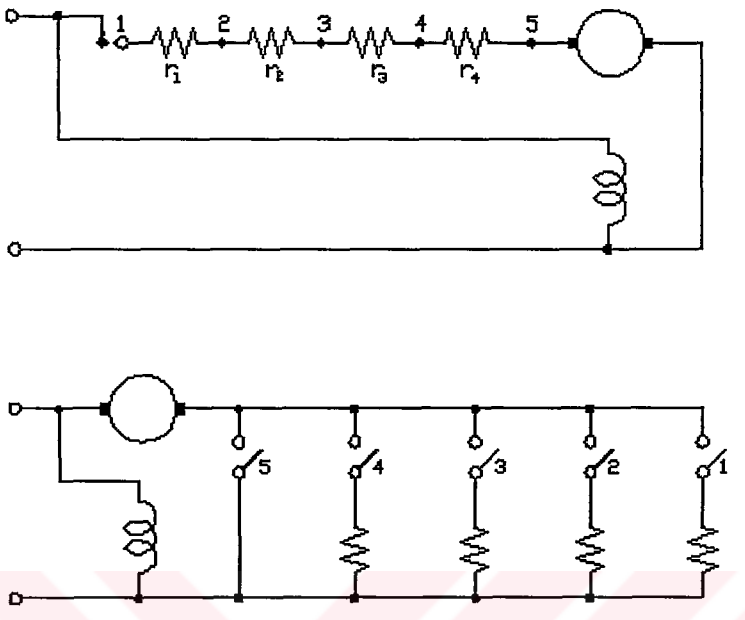
Kalkış anında motor uçlarına gerilim uygulanmasına rağmen rotor durağan haldedir. Bu şartlar altında zıt emk sıfır ve çekilen akım, rotor direnci ile sınırlandırılmıştır. Küçük motorlarda ki

bunları 1.5 kW ve daha küçük güçlerde imal edilen motorlar olarak kabul edersek; rotor sargısı direnci küçük bir değere sahiptir (1 Ω'dan daha küçük olduğu düşünülebilir). Bu da kalkış anında motorun tam yüklü akımının 10 ila 20 katı arasında bir akım çekmesi demektir.

Motorun belli bir hız değerine erişmesiyle ve zıt emk oluşmasıyla akım nominal değerine yavaş yavaş azalarak ulaşır. Dönen kısımların ataleti ve hızlanma momenti ile motorun istenen hıza ulaşması zaman alacaktır. Bu zaman süresince motor tam yük akımından daha fazla bir akım çekecektir. Mekanik kuvvetler ile bu yüksek akım bir araya gelince rotor devresi ısınacak, motor bu yüksek akımları zararlı bir takım arklar olmadan komütasyon oluşturamayacak ve sınırlandırılmış güvenli akım taşıma kapasitesini geçen bu akımdan dolayı fırçalar zarar görecektir. Aynı zamanda yüksek değere sahip kalkış akımı, kaynak üzerinde büyük gerilim düşümlerine sebep olacak ve bu da aynı kaynaktan beslenen diğer elemanları etkileyecek ve koruma devrelerini yükleyecektir. Bununla beraber kalkış akımı mekanik yüke verilecek olan moment sınırının üzerinde aşırı bir kalkış momenti oluşturacaktır. Kalkış boyunca maksimum akım rotor devresine dışardan seri olarak bağlanacak kabul edilir değerde bir direnç vasıtasıyla sınırlandırılabilir. Kalkış akımının oldukça büyük olması momentin doğrudan akımla orantılı olması sebebiyle yüksek kalkış momenti elde etmemizi sağlar (Wildi, 1997).

Genellikle kalkış akımının tam yük akımının 1.25 ile 2 katı arasında olmasına izin verilerek motorun yük altında kalkması sağlanır. Böylelikle kavrama mekanizmalarına ihtiyaç duyulmaz ve seri motorların yüke doğrudan kuplajına izin verilmiş olur. Motor uçlarının gerilim kaynağına bağlanmasıyla şönt motordaki akı şönt alan sargısının yüksek endüktansı yüzünden bir zaman sabitiyle orantılı değişim gösterir. Bu da momentin değerinin artmasını yavaşlatarak, motor ve yük arasındaki mekanik bağlantı elemanın ani bir zarar görmesini engeller. Seri motorda rotor endüktansı küçük olduğu için akı ve akımın yükselişi oldukça çabuk olur. Bununla beraber kalkış momentinin oluşumu ani olmamaktadır. Motor durağan halden hızlanmaya başladığı zaman zıt emk E_a motor uçlarına uygulanan gerilime ters yönde meydana gelir. Zıt emk'nın artmasıyla rotordan akan akımda bir azalma olur. Akım tam yük akımının değerine kadar düştüğünde kalkış direnci devre dışı bırakılır. Akımın tam yük akımından daha küçük olmasını beklemek kalkışta gereksiz bir vakit kaybı olacaktır. Bu yüzden motor hızını kazanırken kalkış direnci izin verilen maksimum akımı geçmeyecek ve tam yük akımının altına düşmeyecek şekilde adım adım azaltılmalıdır. Şekil 2.7'de kalkış akımının kontrolünü sağlayan dirençlerin seri ve paralel bağlantı şemaları verilmiştir. Seri ve

paralel düzenlemede kayan bir anahtar 1 noktasından 5 noktasına kadar ilerleyerek dirençleri devre dışı bırakmaktadır. Böylece seri düzende dış direnç sıfır olana kadar adım adım azaltılır.



Şekil 2.7 Motorun kalkışı (a) Seri dirençle kalkış (b) Paralel dirençle kalkış

Her adımda kullanılacak dirençler seri düzende n direnç adımı ve n+1 kontak olduğu varsayılarak hesaplanır. Akım değeri maksimum rotordan geçmesine izin verilen akım değeri I_H'ı aşmayacak ve minimum akım I_L akımının altına düşmeyecek şekilde seçilecektir. İstenen toplam direnç $R_1=U/I_H$, $R_1=r_1+r_2+r_3+.....+R_a$ ve r_1,r_2,r_3 ile diğerleri kalkış direncinin her adımı meydana getiren direnç değerleridir.

Devredeki endüktansın etkisi ihmal edilecek olursa I_H akımı gerilim kaynağının bağlanması ile kontağın 1 nolu konuma getirilmesi ile akmaya başlar. Hızlanma momenti meydana gelir, rotor dönmeye başlar, zıt emk hızla orantılı olarak oluşur ve akım I_L değerine doğru derece derece azalır. Bu noktada kontak 2 nolu konuma getirilir ve akım, endüktans etkisi ihmal edilir ve anahtar aniden konum değiştirdiği kabul edilirse, tekrar I_H değerine ulaşır. Anahtarlamadan biraz önce akım $I_L=(U-E_a)/R_1$ ve anahtarlamadan hemen sonra zıt emk henüz değişmemişken $I_H=(U-E_a)/R_2$ olacaktır. $R_2=r_2+r_3+....R_a$'dır.

Şayet akım oranı sabit bir k değerinde tutulacak olursa,

$$\frac{I_H}{I_L} = k = \frac{R_1}{R_2} = \frac{R_2}{R_3} = \ldots = \frac{R_n}{R_a} \tag{2.12}$$

direnç değerleri geometrik bir dizidedir ve

$$k^n = \frac{R_1}{R_2} \times \frac{R_2}{R_3} \times \cdots \times \frac{R_n}{R_a} = \frac{R_1}{R_a} \tag{2.13}$$

bu bağıntıdan n adım sayısı hesaplanabilir. Kesin bir hesaplama için adım sayısı ile akım oranı arasında bir ilişki kurmak gerekir. Her direncin değeri aşağıdaki formülden elde edilir.

$$r_1 = R_1 - R_2 = R_1\left(\frac{k-1}{k}\right) \tag{2.14}$$

$$r_2 = R_2 - R_3 = R_2\left(\frac{k-1}{k}\right) = R_1\left(\frac{k-1}{k^2}\right) \tag{2.15}$$

2.3 Doğru Akım Motorunun Frenlenmesi

Güç kaynağı motordan ayrılacak olursa motor bir süre sonra duracaktır. Frenleme etkisi sadece mekanik sürtünme ile oluşur. Rotorda ve mekanik yük üzerinde depo edilen kinetik enerji yüzünden motorun durağan hale gelmesi zaman almaktadır. Sık sık sözü edilen bu zaman çok uzun sürmektedir. Durma süresi boyunca, kullanılan motorun sargı uçlarını besleme kaynağından ayırdıktan sonra bir yük üzerine bağlanması sonucu, jeneratör çalışma moduna geçilerek rotor üzerinde biriken atalet kısa zaman içerisinde harcanır. Böylece frenleme süresi oldukça kısaltılmış olur. Jeneratör çalışmada elektromanyetik moment rotor ve yükün dönme yönüne zıt yöndedir. Mekanik sistem içerisinde saklanan enerji bir elektrik sistemini besleyerek elektrik enerjisine veya daha sık karşılaştığımız termal enerjiye çevrilir. Frenleme zamanı, izin verilen maksimum akım değeri altında kinetik enerjinin elektrik veya ısıl enerjiye transfer zamanına bağlıdır. En büyük frenleme etkisi yüksek hızlarda ve yüksek akı değerlerinde olur. Çünkü transfer edilen güç hızla orantılı olan emk'ya bağlıdır. Motor, jeneratör moda geçtiği zaman manyetik alanın yönü korunur. Çünkü saklanmış olan büyük miktardaki manyetik alan enerjisini anahtarlamak mümkün değildir. Aynı dönüş yönü için üretilmiş olan emk'nın polaritesi değişmez. Şayet elektromanyetik momentin yönü değişerek

frenleme momenti olursa rotor akımı tam ters yönde akmaya başlayacaktır. Şönt motorun besleme uçlarına bağlı alan sargıları, bağlantılarında hiçbir değişme olmaksızın motor çalışmadan jeneratör çalışmaya geçebilir. Seri makinada ise alan sargısının yönünü aynı tutmak için seri sargı ters çevrilir veya alan sargısı frenleme işlemini daha iyi kontrol edebilmek amacıyla ayrı bir kaynaktan beslenir. Kompound makinada ise frenleme esnasında seri sargı devreden ayrılır ve sadece şönt sargı uyarma yapar.

Elektriksel frenlemede üç farklı yöntem vardır. Bunlar a)ters polaritede gerilim uygulama, b)dinamik frenleme ve c)rejeneratif frenlemedir. **Zıt yönde gerilim uygulama** ya motorun çok hızlı bir şekilde durması istenen yerlerde ya da motoru ters yönde hızla yön değiştirmesinin istendiği durumlarda kullanılan bir yöntemdir. Bu yöntemde magnetik alanın aynı yönde kalması sağlanırken gerilim kaynağı uçları ters çevrilerek rotora uygulanır. Frenleme peryodu boyunca, üretilen gerilim ve kaynak gerilimi birbirlerini beslerler. Ters gerilim uygulanması anında geçen akım kalkış akımının maksimum değerinin iki katına kadar çıkar ve motor durağan hale geldiğinde son değerine doğru azalır. Durağan hal konumundayken moment hala aynı yönde kalırken şayet motor uçlarına hala gerilim uygulanmaya devam edilirse makina ters yönde hız kazanarak tekrar motor olarak çalışmaya başlar. Hızlı frenleme istendiği zaman bu metot kullanılır fakat kalkış akımını yarıya düşürebilmek için kullanılan direnç değerleri, akımı istenilen sınırlarda tutmak için iki katına çıkarılır. Rotordaki ve mekanik yük üzerindeki kinetik enerji devreye alınan bir direnç üzerinde ısıl enerjiye dönüştürülerek harcanır. Elektriki olarak frenlemede kullanılan bütün metotlarda alışılagelmiş karakteristikler hızın ve üretilen emk'nın azaltılması üzerine olmaktadır. Devreye alınan direnç değeri istenilen akımı ve momenti sağlamak üzere azaltılmalıdır.

Dinamik frenleme bazen reostatik frenleme olarak da adlandırılır ki; orjinal akı yönü sabit kalırken, rotor devresini güç kaynağından ayırıp bir frenleme direncine anahtarlanmasıyla gerçekleştirilir. Makina dönme hareketini gerçekleştirirken bir jeneratör gibi davranır. Rotor sargısından motor çalışmadakinin aksi yönde akım akmaktadır ve moment ters yönde olup frenleme momenti olarak adlandırılır. Frenleme momenti rotor sargısı akımı ve akının bir fonksiyonudur. Rotor yavaşlarken ve üretilen gerilim azalırken rotor akımı frenleme direnci adım adım azaltılmasıyla istenilen limitler arasında korunur. Bu uygulama kalkış anında yapılan uygulamayla büyük benzerlik içerisindedir. Dönen kütlenin toplam kinetik enerjisi ısı enerjisi olarak açığa çıkar. Her iki yöntemde de normalde depo edilen kinetik enerji harcanır.

Rejeneratif frenlemede ise kinetik enerji bir elektrik sistemini beslemek üzere elektrik enerjisine çevrilir. Bu yöntemin avantajı üretilen gerilimin besleme gerilimini aşması ve akımın yönünün değişmesidir. Bu durumda akı artabilir veya hız normal hızın üzerine çıkabilir veya her ikisi birden meydana gelebilir. Rejeneratif frenleme güvenli hızlarda kullanılabilir fakat yükü hareketsiz hale getiremez. Bunun için bir veya ikinci yöntemi ve gerekliyse mekanik bir frenleme de kullanılmalıdır (Ramshaw ve Heeswijk, 1990).

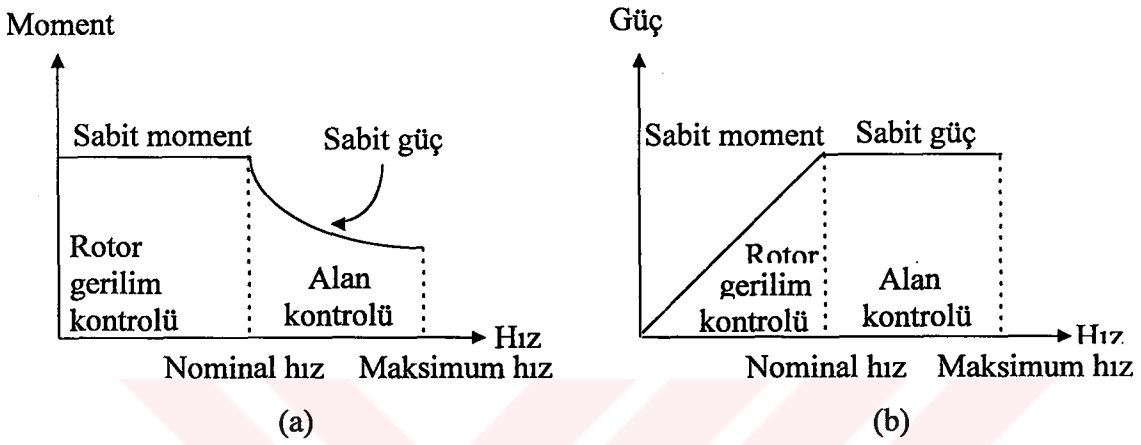
2.4 Doğru Akım Motorunun Hızının Kontrolü

Doğru akım makinaları, sabit hızla dönen alanla birarada düşünüldüğünde ayarlanabilir hız mekanizmalarına AC makinalardan daha uygundur. Hız ayarının hassas ve kolay bir şekilde yapılabilmesi doğru akım makinalarının endüstride geniş bir uygulama alanı bulmasına yol açmıştır. Hız kontrol yöntemlerinden en çok kullanılan yöntemler sırasıyla; akının ayarlanması ki; bu genellikle **alan akımının** kontrolüdür, rotor devresine bağlanan **direncin ayarı** ve rotor devresine uygulanan **gerilimin ayarı**dır.

Alan akımı kontrolü çok kullanılan bir metot olup, şönt motorların dikkat çeken bir avantajıdır. Bu yöntem aynı zamanda kompound motorlara da uygulanabilir. Alan akımının ayarlanması, bundan dolayı akının ve hızın da ayarlanması, şönt alan sargısı direncinin ayarlanması veya yarıiletken kontrolü kullanılmasıyla daha basit, ucuz ve motor kayıplarında değişim olmaksızın gerçekleştirilebilir. En düşük hız, maksimum alan akım aktığında sağlanırken, en yüksek hız ise motorun izin verdiği elektriksel olarak sınırlanmış en düşük alan akımda meydana gelir.

Rotor devresi direnç kontrolü, rotor devresine dışardan bağlanacak dirençler vasıtasıyla motorun hızını azaltmayı amaçlar. Seri, şönt ve kompound motorlarda bu yöntem kolaylıkla uygulanabilir. Dışardan bağlanacak olan direnç, şönt ve kompound motorda, şönt alan sargısı ile rotor devresi arasına bağlanmaktadır. Motor ile besleme arasına bağlanmaz. En çok seri motorlarda kullanılan bir yöntemdir. Devreye seri olarak bağlanmış sabit bir direnç için hız yükle geniş bir aralıkta değiştirilir. Hız, yüke bağlı olarak rotordan geçen akımın bu direnç üzerinde düşürdüğü gerilim düşümüne bağlıdır. Hız büyük miktarlarda azaltıldığı durumlarda bu direnç üzerindeki güç kaybı oldukça büyüktür.

Rotor gerilimi kontrolünde, rotor gerilimi ayarlanarak motorun hızı geniş bir aralıkta değiştirilir. Rotor gerilimi küçük güçlü motorların haricinde darbe genişlik modulasyonu kullanılarak ayarlanır. Bu sayede motor devri çok hassas aralıklarla istenilen hıza ayarlanabilir. Yarı iletken elemanların gelişmesi neticesinde yüksek frekanslarla yapılan anahtarlama işlemi sonucunda gerilimin ortalama değeri 0 ile %100 arasında değiştirilerek rotor uçlarına uygulanır. Çok büyük güçlerde ise tristörlü ac kıyıcılar kullanılarak bu işlem gerçekleştirilir (Nachtigal, 1990).



Şekil 2.8 Hız kontrol yöntemlerinde alan reostası ve rotor gerilimi kontrol metotlarının ayar sınırları (a) Moment-Hız (b) Güç-Hız

Doğru akım motorunda rotor uçlarına uygulanan gerilimin değiştirilmesi ile temel hız dediğimiz nominal hıza kadar olan hız ayarı rahatlıkla yapılabilir. Bu bölgede motorun momenti sabit ve çekilen güç değişkendir. Şekil 2.8(Fitzgerald ve Kingsley, 1992)'de gösterildiği üzere bu bölgeye *sabit moment bölgesi* adı verilir. Hızı daha fazla arttırmak istersek bunun için alan sargısı geriliminin kontrol edilmesi gerekir ki, bu sayede nominal hızın dört katına kadar çıkılabilmektedir. Bu bölgede moment değişmekte fakat güç sabit kalmaktadır. Bu yüzden bu bölgeye de *sabit güç bölgesi* adı verilir (Fitzgerald ve Kingsley, 1989).

2.5 Doğru Akım Motorunun Elektriksel Analizi ve Blok Diyagramı

Doğru akım motorunun rotor geriliminin kontrolü, hızı ayarlamak için en uygun ve ucuz yöntemlerden biridir. Kalıcı mıknatıslığa sahip motorlarda rotordaki uyarma sabit olacaktır. Şekil 2.9'da böyle bir motor gösterilmiştir. Rotor uçlarındaki gerilimin değeri darbe genlik modülasyonu ile ayarlanmaktadır. E_a gerilimi manyetik alan içerisinde dönen rotorda oluşan zıt emk'dır. Zıt emk, rotor hızı n (rad/sn) ve alan akısı ϕ (Wb/m^2) ile orantılıdır. Böylece,

E_a=$K_3 n \phi$ (2.16)

elde edilir. Burada K_3 motorun özel bir sabitidir. ϕ burada

$$\varphi = k_f I_f \quad (2.17)$$

şeklinde yazılır. ϕ 'nin açılımı yerine konulursa,

$$E_a = k_f I_f n = k_w n \quad (2.18)$$

bulunur. Şekil 2.9'da gösterilen şemadaki rotor devresi için eşitlikleri yazacak olursak,

$$U - k_f I_f n = R_a I_a + s L_a I_a = R_a (1 + \tau_a s) I_a \quad (2.19)$$

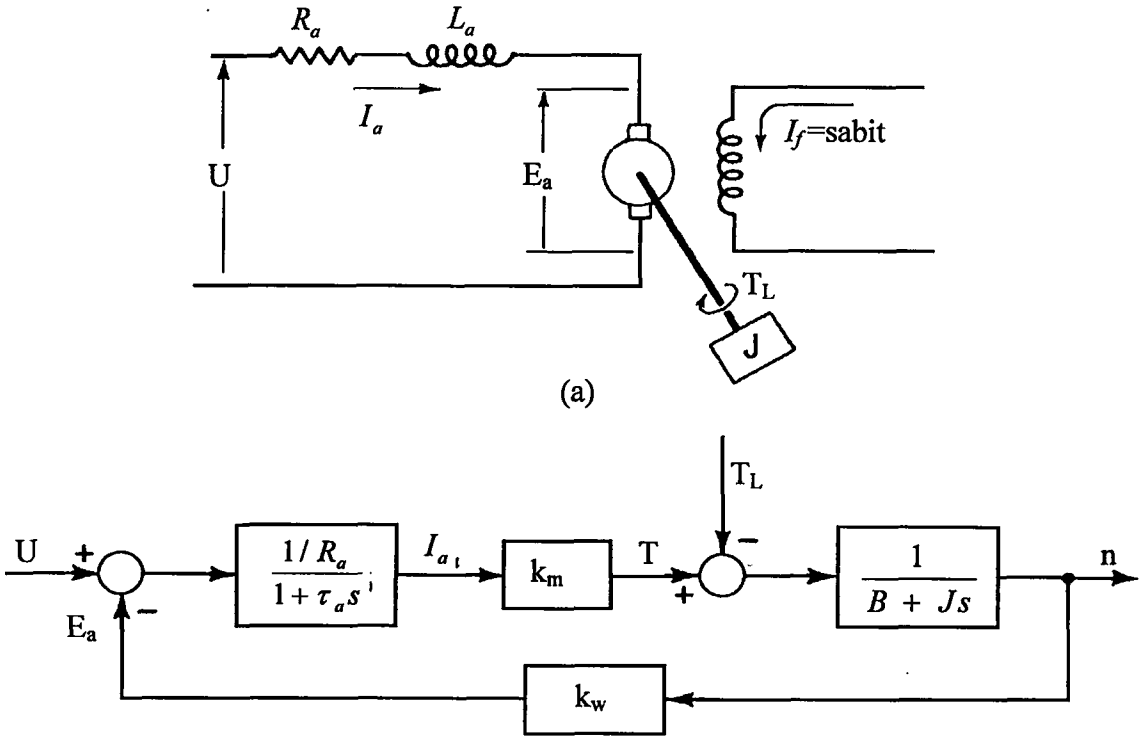
elde edilir. Burada $\tau_a = L_a / R_a$ ve $k_f I_f n$ rotorda oluşan zıt emk'dır. Motorda oluşan moment ifadesi aşağıda verilmiştir.

$T = k_S I_f I_a = k_m I_a$ (2.20)

Çıkıştaki yükün ataleti J ve sürtünme momenti katsayısı B_v hesaba katıldığında moment aynı zamanda,

$T = (B_v + Js)^{-1} n + T_L$ (2.21)

Rotor gerilimi kontrol edilen motor için 2.19, 2.20 ve 2.21 eşitliklerinden elde edilen blok diyagram Şekil 2.9'da gösterilmiştir. Burada zıt emk, geri besleme olarak gösterilmiştir (Raven, 1987).



Şekil 2.9 Rotor gerilimi kontrol edilen doğru akım motoru

Blok diyagramdaki blokların içine değerleri konulacak olursa;

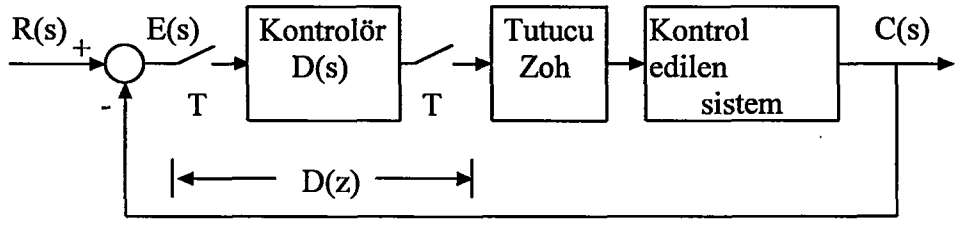
$$T_1(s) = \frac{I_a}{U - E_a} = \frac{1}{R_a + sL_a} = \frac{1}{2.9 + 4.3 \cdot 10^{-3}} \tag{2.22}$$

$$T_2(s) = \frac{w}{T} = \frac{1}{Js + B} = \frac{1}{0.12 \cdot 10^{-3} s + 0.33 \cdot 10^{-3}} \tag{2.23}$$

3. DİJİTAL-PI KONTROLÖR

3.1 Dijital Kontrolörler

Bir otomatik kontrol sistemi; geribesleme elemanı, sistem ve kontrolörler gibi çok değişik elemanların, istenilen davranışı gerçekleştirecek biçimde bir araya getirilmesinden meydana gelmiştir. Başlangıçta kontrol işlemi için analog sistemler kullanılırken, daha sonra uygunluğu ve maliyetin azaltılmasıyla işlemi daha esnek bir yapıda gerçekleştirebilen dijital kontrol sistemleri tercih edilmeye başlamıştır.



Şekil 3.1. Dijital kontrol sisteminin blok diyagramı

Şekil 3.1'de dijital kontrol sisteminin blok diyagramı gösterilmiştir. Blok diyagramda; R(s) girişe uygulanan referans işareti, C(s) sistemin uygulanan referans girişe karşı verdiği çıkış işaretini ve E(s) ise referans işareti R(s) ile çıkış işareti C(s) arasındaki farkı yani hata işaretini temsil etmektedir. Dış dünyadaki sistemlerin çoğu analog sistemlerdir. Bu sebeple analog dünya ile dijital dünya arasında bir bağlantı kurmak gerekir. Bu bağlantı analog-dijital dönüştürücülerin, analog işareti istenilen örnekleme zamanı (T) altında ayrık zamanlı işarete dönüştürmesiyle sağlanır. Ayrık zamanda gelen bu bilgi dijital kontrolör yapısı içinde, istenilen kontrol algoritmasında işlendikten sonra, analog sisteme 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 kontrolörden çıkan işaret bir tutucuya verilerek burada saklanır (Sarıoğlu, 1992).

Dijital kontrolörlerin analog olanlara göre birtakım üstünlükleri bulunmaktadır. Bu üstünlükleri sıralayacak olursak;

1. Dijital kontrolörün özelliklerinin değiştirilebilmesi sadece programının değiştirilebilmesi ile mümkünken, klasik analog kontrolörlerde karakteristiğinde yapılan en küçük değişiklik pahalı ve zahmetli olmaktadır.

2. Dijital kontrolörde bilgilerin işlenmesi daha kolaylaştırılmıştır. Karmaşık hesaplamalar hızlı ve daha uygun bir yöntemle yapılabilmektedir.

3. Aşağıda belirtilen özellikler bakımından analog kontrolörlere göre dijital yapıda olanlar daha üstündür.

a) Duyarlılık

b) Kayma etkisi

c) İç gürültüler

d) Güvenilirlik

4. Analog kontrolörlere oranla daha ucuzdur.

5. Analog kontrolörlere göre daha küçük boyutlardadır. Bununla beraber dijital kontrolörlerin analog yapıda olanlara göre bazı eksiklikleri vardır. En büyük eksiklik, hatanın değişimini sadece örnekleme frekansının izin verdiği zaman aralığında sezebilmesidir. Fakat örnekleme frekansının arttırılmasıyla bu eksiklik ortadan kaldırılabilir. (Paraskevopoulos, 1996)

3.2 PI Kontrolör

Analog kontrol sistemlerinde en geniş kullanım alanı bulan kontrolör, orantı-integral kontrolörü olan PI kontrolördür. 3.1, 3.2 ve 3.3 eşitliklerinde PI kontrolörün t, s ve z domenlerinde transfer fonksiyonları sırasıyla verilmiştir.

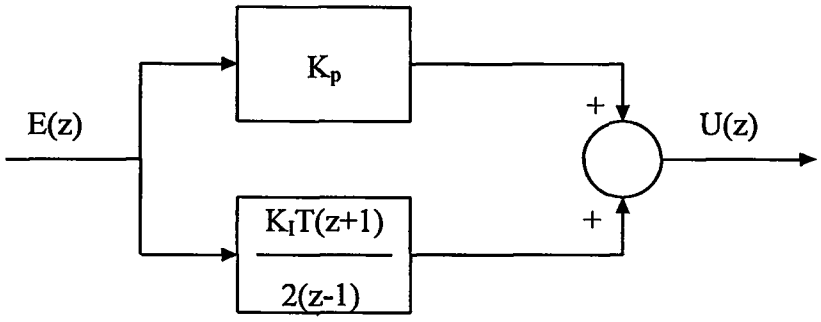
$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(t)dt \Rightarrow U(s) = K_p \cdot E(s) + \frac{K_i \cdot E(s)}{s} \tag{3.1}$$

$$G_k(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} \tag{3.2}$$

$$G_k(z) = \frac{U(z)}{E(z)} = K_p + \frac{K_i T(z+1)}{2(z-1)} \tag{3.3}$$

Burada; K_p ve K_i sırasıyla oran ve integral katsayılarıdır. Bu katsayıların seçimine bağlı olarak sistemin performansı değiştirilebilinir. Hata sinyali olan e(t), ayrık işlemlerin yapıldığı

z domeninde E(z) ile ifade edilir. Şekil 3.2'de hata sinyali E(z)'yi giriş olarak kullanan PI kontrolör gösterilmiştir. Kontrolörün çıkışındaki U(z), sisteme uygulanacak kontrol işaretini ifade etmektedir (Aström ve Wittenmark, 1989)



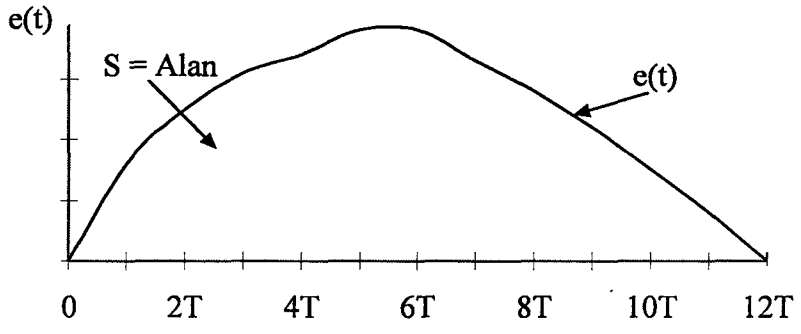
Şekil 3.2 PI kontrolörün blok diyagramı

Oransal kontrol basit olarak, hatanın K_p sabitiyle çarpılmasından elde edilen değerdir. Oransal kontrol, sistemin istenen çıkışa daha hızlı bir şekilde ulaşmasını sağlar. Böylece yükselme zamanının, sistemin dinamik özelliklerinin izin verdiği ölçüde azaltılması sağlanır.

İntegral kontrolü ise hatanın K_i sabiti ile çarpılıp, hatanın polaritesi gözönüne bulundurularak integralinin alınması ile hesaplanır. İntegral kontrolünün amacı e(t)'nin altında kalan alanı küçültmek ya da başka bir deyişle kararlı hal hatasını azaltmaktır.

3.2.1 Nümerik integrasyon

İntegral kontrolünü gerçekleştirebilmek için mikrokontrolörün içerisinde ayrık zamanda gelen e(t) hata işaretinin nümerik metotlarla integralini almamız gerekir. Bir fonksiyonun integrali, bu fonksiyonun altında kalan alana eşittir.



Şekil 3.3 İntegratörün giriş ve çıkış ilişkisi

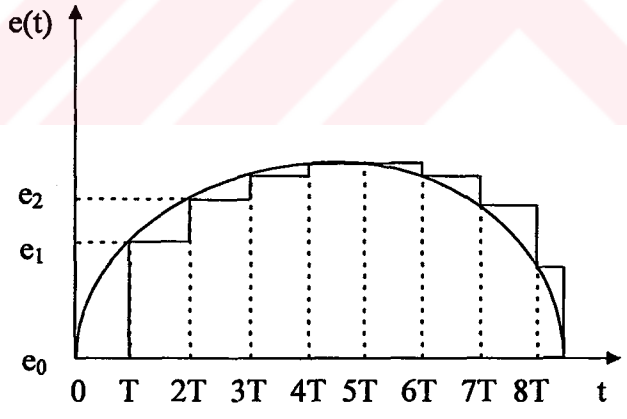
Şekil 3.3'de integratör elemanının girişi olan hata ve çıkışı olan alan arasındaki ilişki gösterilmiştir. Bu ilişki 3.4 eşitliğinde gösterildiği gibi yazılabilir.

$$S = \int_0^t e(t)\ d(t) \tag{3.4}$$

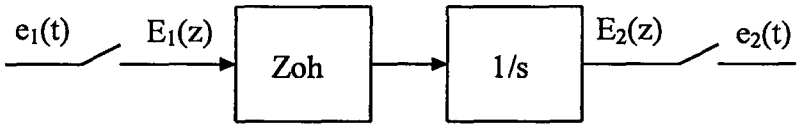
Bu yaklaşımın irdelenmesiyle fonksiyonun altında kalan alanın hesaplanması için üç değişik nümerik integrasyon yaklaşımı yapılmaktadır. Bunlar sırasıyla **ileri yol**, **geri yol** ve **trapezoidal** integrasyon metotlarıdır (Auslander ve Takashi, 1974).

3.2.1.1 İleri yol integrasyonu

Şekil 3.4'te gösterildiği üzere ilk örnekleme anı olan 0 anında alınan değer tutucuda T zamanına kadar aynı değerde tutulur. Bu kısmın alanı 0'dır. Bu yöntemde sadece bu alan ihmal edilmektedir. Bu yöntemde alan hesabı örnekleme peryodunun başlangıcında yapılır. T anında tutucuya yeni örnek ulaşır. Bu andan 2T anına kadar bu örnek tutularak bu dikdörtgensel bölgenin alanı hesaplanır. Böylece her örnekleme frekansında gelen bilgiye göre bu dörtgen şekle sahip yaklaşık alanlar toplanarak ileri integrasyon işlemi yapılır.



(a)



(b)

Şekil 3.4 İleri yol integrasyonunun a) grafik b) blok diyagramı ile gösterilmesi

Gelen bilgilerin e_0, e_1, e_2,..... olduğunu kabul edersek;

$$S = \frac{e_0 \cdot (2T - T) + e_1 \cdot (3T - 2T) + e_2 \cdot (4T - 3T) + \ldots\ldots e_n \cdot \left((n+2)T - (n+1)T\right)}{nT} \tag{3.5}$$

olur. Burada S grafiğin altında kalan alanı ifade etmektedir. Bu integrasyon işlemini z domeninde transfer fonksiyonu verilecek olursa aşağıdaki eşitlik elde edilir.

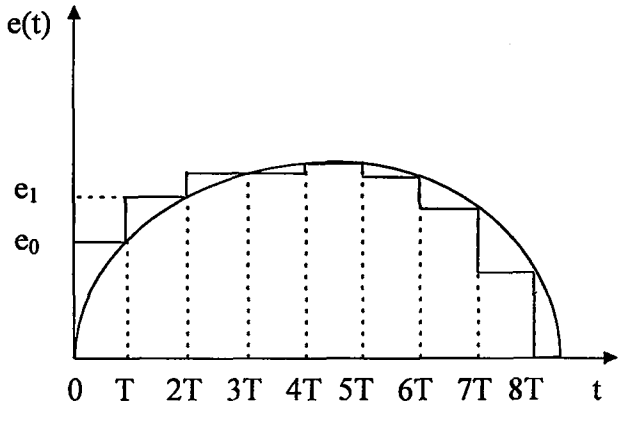
$$\frac{E_2(z)}{E_1(z)} = Z\left\{\frac{1 - e^{-Ts}}{s^2}\right\} = \left(1 - z^{-1}\right)\frac{Tz}{(z-1)^2} = \frac{T}{z-1} \tag{3.6}$$

3.2.1.2 Geri yol integrasyonu

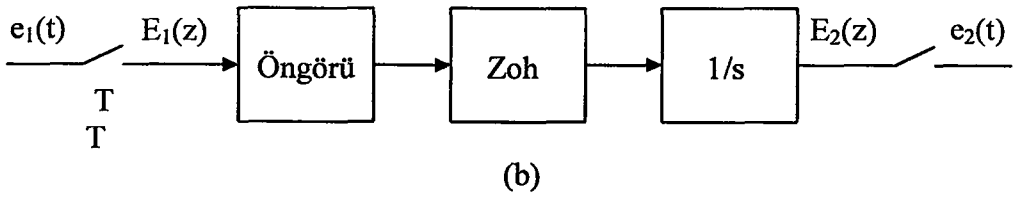
İleri yol integrasyonunda bahsettiğimiz gibi, örnekleme işleminin başlaması ile ilk alınan bilgi sıfır değerini taşımaktadır. Fakat bu bilginin ikinci örnekleme işlemine kadar değişmesine rağmen tutucunun çıkışında sıfır bilgisi tutulmaya devam etmektedir. Bu işlem bir hata oluşturmaktadır. Bu hatayı gidermek için geri yol integrasyon yöntemi kullanılır. Bu yöntemde alan hesabı örnekleme peryodunun sonunda yapılır. Öngörü, T anında gelecek olan bilginin bilinmesi ve bunu sıfır anındaki ilk bilgi olarak kabul edilmesi ile gerçekleştirilir. Bu yöntem Şekil 3.5'te gösterilmiştir. Transfer fonksiyonunu z domeninde ifade etmek gerekirse;

$$\frac{E_2(z)}{E_1(z)} = Z\left\{e^{Ts}\left(\frac{1 - e^{Ts}}{s}\right)\frac{1}{s}\right\} = (z-1)\frac{Tz}{(z-1)^2} = \frac{Tz}{z-1} \tag{3.7}$$

şeklinde bulunur.



(a)



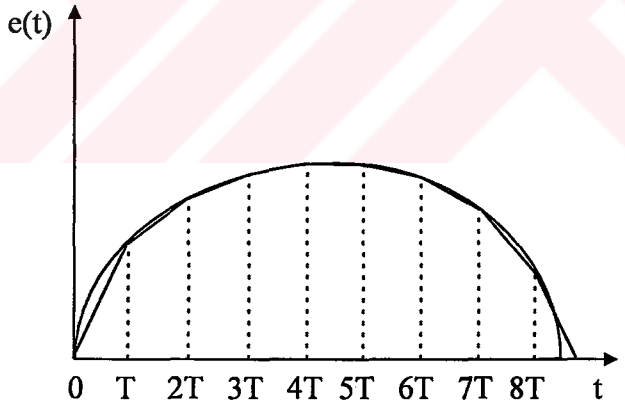
(b)

Şekil 3.5 Geri yol integrasyonunun (a) grafik (b) blok diyagram ile gösterilişi

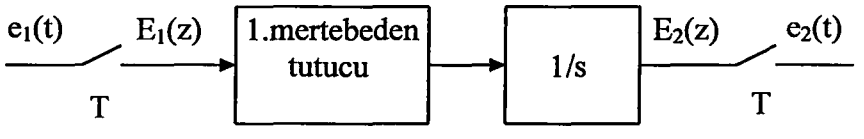
3.2.1.3 Trapezoidal integrasyon

Bu yöntem Şekil 3.6'da gösterilmiştir. Hem ileri hem de geri yol integrasyonunda şekillerde de görüldüğü üzere kullanılan basamak şeklindeki alan işleminde hatalar oluşmaktadır. Çok hassas integrasyonun gerektiği durumlarda trapezoidal integrasyon yöntemi kullanılır. Bu yöntem, ileri yol integrasyonu ile geri yol integrasyonu arasındaki farkın yarısının, ileri yol integrasyonuna ilave edilmesiyle hesaplanır. Yapılan işlemin hassasiyetinden dolayı genellikle uygulamalarda bu yöntem tercih edilir. z domeninde aşağıdaki gibi ifade edilir (Kuo, 1992).

$$\frac{E_2(z)}{E_1(z)} = \frac{T}{z-1} + \left(\frac{Tz}{z-1} - \frac{T}{z-1}\right)\cdot\frac{1}{2} = \frac{T(z+1)}{2(z-1)} \tag{3.8}$$



(a)



(b)

Şekil 3.6 Trapezoidal integrasyonun (a) grafik (b) blok diyagramı ile gösterilişi

3.3 PI Kontrolör Parametrelerinin Seçimi

PI kontrolör parametrelerinin belirlenmesinde pek çok yöntem kullanılmaktadır. Bu yöntemlerden bir diğeri de kutup-sıfır yerleştirme metodudur. s veya z domenindeki kontrol sistemlerinin tasarımında en alışılagelmiş yöntem; sistemi kararsızlığa götürebilecek istenmeyen kutup ve sıfırların, kontrolörün kutup ve sıfırlarını kullanarak transfer fonksiyonundan çıkarılmasıdır. Daha sonra aşımın minimum olmasını sağlamak amacıyla yerleştirilecek sıfırlar, s domenindeki eksen takımında kontrol kutuplarının beş katı kadar uzağına yerleştirilir.

Kontrolör parametrelerini bulmak için öncelikle kontrol edilecek sistemin transfer fonksyonunun bulunması gerekir. Kontrol edilecek sistem, kalıcı mıknatıslı uyarma sargısına sahip doğru akım motoru olduğuna göre matematiksel modelini çıkaracak olursak,

$$E_a = k_w \cdot w \tag{3.9}$$

$$U = R_a \cdot I_a + L_a \cdot \frac{dI_a}{dt} + k_w \cdot w \tag{3.10}$$

$$T = k_m \cdot I_a = T_L + J \cdot \frac{dw}{dt} + B \cdot w \tag{3.11}$$

Burada, T_L yük momentini, J atalet momentini ve B sürtünme katsayısını göstermektedir.

$L_a \cong 0$ alınarak ihmal edilirse; $G_m(s) = \frac{w(s)}{U(s)} = \frac{b}{s+a}$ şeklinde elde edilir. Buradan;

$b = \frac{k_m}{J \cdot R_a}$ ve $a = \frac{1}{J}\left[B + \frac{k_w \cdot k_m}{R_a}\right]$ olarak bulunur. Motorların etiket değerleri verilen formüllerde yerine konulacak olursa kontrol edilecek sistemin transfer fonksiyonu elde edilmiş olur. Fakat yapılan çalışmada aynı etiket değerlerine sahip üç motor sırasıyla motor, jeneratör ve tako jeneratör olarak kullanıldığı için atalet momenti J ve sürtünme katsayısı B'nin değerlerini üçle çarptıktan sonra ilgili eşitliklerde kullanacağız.

Yukarıda verilen a ve b eşitliklerini hesaplayabilmek için çalışmamızda kullanmış olduğumuz kalıcı mıknatıslı doğru akım motorunun etiket değerleri İnternet üzerinden Engels firmasına bağlanarak elde edilmiştir. Bu etiket değerleri aşağıda verilmiştir.

$J = 0.0001263 kgm^2$

B = 0.000334224Nm*sn/rad

km = 0.14Nm/A

kw = 0.13941973V*sn/rad

Ra = 2.9 ohm

Bu durumda transfer fonksiyonunu elde etmek için motorun etiket değerleri yerine konursa ;

$$G_m(s) = \frac{w(s)}{U(s)} = \frac{382.317}{s + 55.9369} \qquad (3.12)$$

şeklinde elde edilir. Kutup ve sıfır ilave etme işlemlerimizi s domeninde yapıldıktan sonra elde edilen K_p ve K_i katsayılarını Tustin yöntemiyle z domenindeki katsayılara dönüştürülür. Bu işlemleri gerçekleştirebilmek ve simüle edebilmek için MAT_LAB kontrol toolbox'ı kullanılmıştır. Programın çalıştırılmasıyla Şekil 3.7'de görülen sistemin kök-yer eğrileri otomatik olarak çizildi. Bu kök-yer eğrileri üzerinde imajiner eksenin solunda kalacak şekilde kontrol kutuplarını belirlenmiştir. Kontrol kutuplarının belirlenmesiyle, kontrolör katsayıları, yazılan program sayesinde hesaplandı. Kontrolörün çalışmasını simüle edebilmek için Şekil 3.9'da görülen kontrol sisteminin blok diyagramı kontrol toolbox'ı içinde oluşturularak sistem çalıştırıldı. Kontrolörün performansını test edebilmek için, kontrol edilen çıkış işareti olan motorun hızı, istenilen referansa oturduktan sonra 52 W'lık yük devreye alınıp çıkarıldı. Şekil 3.8'de bu çıkış işaretinin oturması ve yükün devreye alınıp çıkarılması anındaki değişim gösterilmiştir (Kuo, 1994).

PI Kontrol Katsayılarının Elde Edilmesi İçin Hazırlanmış Olan MATLAB Programı

```
J = 0.0001263kgm^2;
B = 0.000334224Nm*sn/rad;
km = 0.14Nm/A;
kw = 0.13941973V*sn/rad;
Ra = 2.9 ohm;
a = (1/J)*(B+(kw*km/Ra))
b = km/(J*Ra)
nums = b;
dens = [1 a];
```

```
s1 =5a;
s2 =6a;
numpid = [1 s1+s2 s1*s2];
denpid = [1 0];
num = b*numpid;
den = [1 a 0];
rlocus(num,den);
[K,poles] = rlocfind(num,den);
K
Kd = K*1
Kp = K*(s1+s2)
Ki = K*s1*s2
fi = angle(poles(1,1));
zeta = -cos(fi);
Mp = exp(-zeta*pi/sqrt(1-zeta^2))
T = 0.5e-3; örnekleme zamanı
k1 = Kp+(2*Kd/T)+(Ki*T/2); k2=(Ki*T)-(4*Kd/T); k3=-Kp+(Ki*T/2)+(2*Kd/T)
numdpid = [k1 k2 k3]; dendpid=[1 0 -1]
```

Mat-lab Programının Çalıştırılması ile Elde Edilen Sonuçlar

```
a =      55.9369
b =      382.2317
Select a point in the graphics window
selected_point = -79.4278+ 80.5491i
K =      4.2837e-004
Kd =     4.2837e-004
Kp =     0.3636
Ki =     40.21
Mp =     0.1032
k1 =     1.9871
k2 =     -3.4069
k3 =     1.4600
numdpid =   1.9871   -3.4069    1.4600
denpid   =  1    0   -1
```

Buradan elde edilen sonuçlara göre PI kontrolörün ayrık zamanda transfer fonksiyonu;

$$G_{PI}(z) = \frac{1.9871z^2 - 3.4069z + 1.46}{z^2 - 1} \tag{3.13}$$

olarak elde edilir.

3.4 PI Kontrolör Parametrelerinin Analitik Metotlarla Elde Edilmesi

Daha önce elde edilen ve 3.12 eşitliğinde verilen sistemin transfer fonksiyonu aşağıdaki şekildeki gibi ifade edilebilir.

$$G_m(s) = \frac{w(s)}{U(s)} = \frac{382.317}{s + 55.9369} = \frac{K}{\tau s + 1} = \frac{6.84}{17.85 \cdot 10^{-3} s + 1} \tag{3.14}$$

3.14 eşitliğinden görüleceği üzere payda da bulunan τ sistemin zaman sabitini göstermektedir. Genellikle sistem 5τ süresi sonunda kararlı hale ulaşmalıdır. Buradan hareketle sistemin kapalı çevrim kontrol kutbunun reel kısmını $1/\tau$ seçilirse s'in reel kısmı 56'ya eşit olacaktır.

Sistemin aşım yapmadan kararlı hal cevabına ulaşabilmesi için sönüm faktörü $\xi = 0.707$ olarak seçilir.

$$tg\beta = \frac{\sqrt{1 - \xi^2}}{\xi} \tag{3.15}$$

olduğu gözönüne alınırsa, kontrol kutbunun reel eksenle yapmış olduğu açı olan $\beta = 135^\circ$ değerini alacaktır. Bu durumda kontrol kutbu s = -56+j56 şeklinde elde edilir. Bu kontrol kutbu transfer fonksiyonunda yerine konacak olursa;

$$\frac{383}{s + 56}\Big|_{s=-56+j56} = -7.2j \tag{3.16}$$

modülü 7.2 ve açısı $\psi = 90°$ olarak elde edilir. PI kontrolör katsayıları kök-yer eğrilerinin analizi ile elde edilir (Phillips ve Harbor, 1988).

$$K_p = \frac{-\sin(\beta+\psi)}{|G(s)H(s)|\sin\beta} - \frac{2K_i\cos\beta}{|s|} \tag{3.17}$$

olarak elde edilmiştir. Oran ve integral katsayılarını elde etmek için bu parametrelerden biri seçilerek diğeri elde edilmesi yolu izlenmiştir. Buna göre $K_i = 40$ olarak seçilmiştir. K_i 3.17 eşitliğinde yerine konulacak olursa $K_p = 0.576$ olarak elde edilir. Bu elde edilen sonuçlar MAT_LAB programı ile elde edilen sonuçlara oldukça yakındır.

3.5. PI Kontrolör Parametrelerinin Değişimlerinin Sisteme Etkilerinin İncelenmesi

Daha önce MAT_LAB'de sistemi kararlı bir şekilde istenilen referans değere ulaştıracak olan kontrolör parametreleri K_p ve K_i hazırlanan program sayesinde hesaplanmıştır. Bu parametrelerin sabit olması kontrolörün performansını sınırlandırmaktadır. Kontrolörün performansını arttırabilmek için bu parametrelerin zamanla değişen değerlere sahip olması gerekmektedir. Bu parametrelerin sınırlarını belirleyebilmek ve sisteme olan etkisini inceleyebilmek amacıyla daha önce z domeninde hazırlanmış olan sistemin blok diyagramı analizi kolaylaştırmak amacıyla s domeninde tekrar oluşturulmuştur. Şekil 3.10'da sistemin blok diyagramı verilmiştir. Bu blok diyagram içerisindeki kontrolörün parametreleri değiştirilerek sistemin çıkış cevabı simüle edilmiştir.

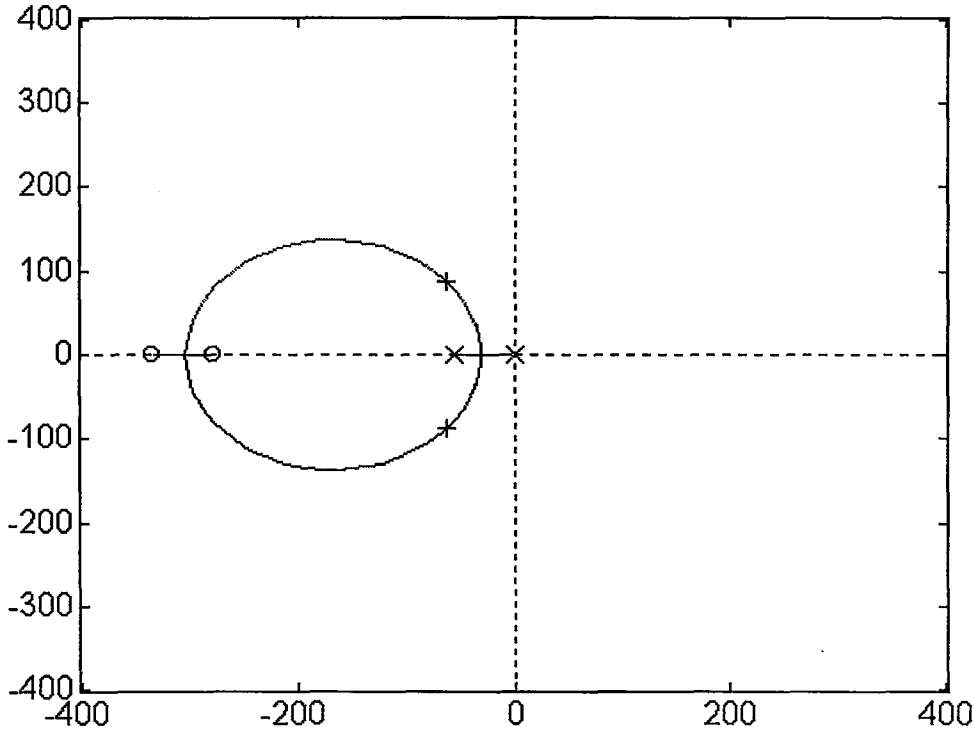
Sistemin %5 aşımla oturmasını sağlayan değer K_p ve K_i için sırasıyla 0.4 ve 40 olarak kabul edilirse bu katsayılar için sistemin çıkış cevabı Şekil 3.11'de elde edilmiştir.

$$u(nT) = K_p \cdot e_n(nT) + K_i \cdot \sum_{n=0}^{t} e_n(nT) \tag{3.18}$$

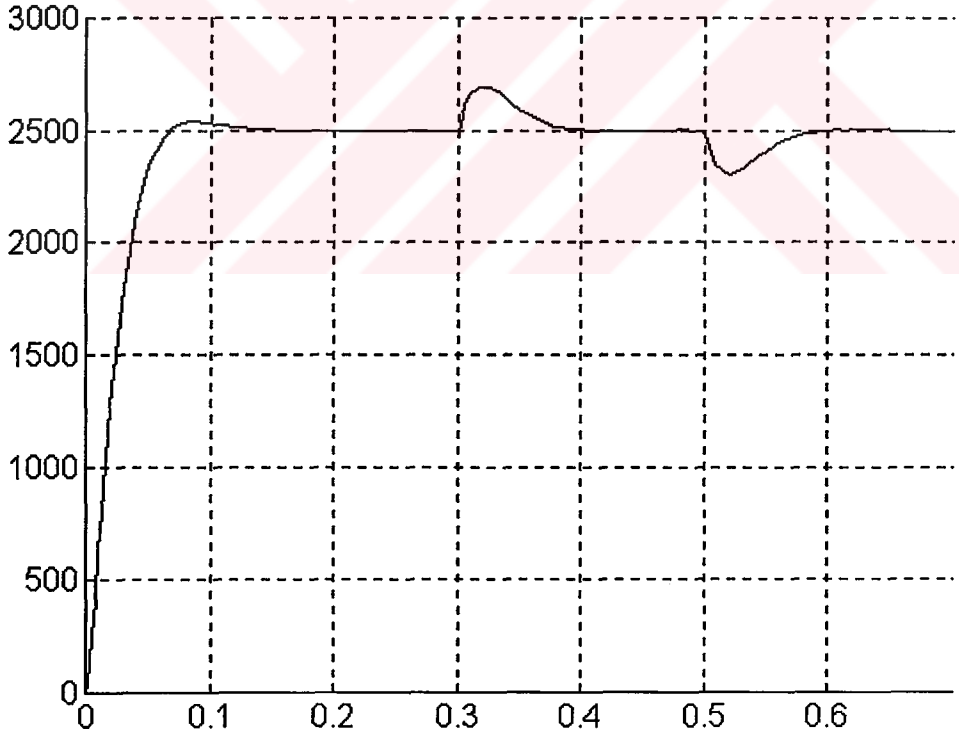
Bu eşitlikte u(t) kontrolörün çıkış işaretini göstermektedir. K_p değeri sabit tutulurken K_i değeri arttırıldığı zaman, 3.18'de verilen formulden görüldüğü üzere başlangıç anından beri toplanan hatanın integrali ile K_i değerinin çarpılması sonucunda kontrolörün çıkışındaki u(t) işareti eskisine oranla oldukça büyümektedir. Bu büyümenin sonucunda daha büyük bir kontrol işareti elde edilmektedir. Bu kontrol işaretinin sisteme uygulanması ile eskisine göre

oldukça artan bir aşım meydana gelmektedir. K_i parametresinin arttırılması, istenmeyen bu aşımı meydana getirirken t_r yükselme zamanının azalmasına da sebep olmaktadır. Şekil 3.12'de bu aşımlı çıkış işareti gösterilmiştir. K_i parametresinin değerinin 80 yapılarak çok arttırılması ise oturma zamanı içerisinde sönümlü osilasyonları meydana getirecektir. Bu sönümlü osilasyonlu çıkış işareti Şekil 3.13'de gösterilmiştir. Aşırı arttırılarak 100 yapılması ise sistemi kararsızlığa sürükleyecektir. K_i değerinin 20 yapılarak azaltılması ise aşımın yokolmasına sebep olmakla beraber yükselme t_r ve oturma t_s zamanlarının artmasına sebep olmaktadır. Şekil 3.14'de bu çıkış işareti verilmiştir.

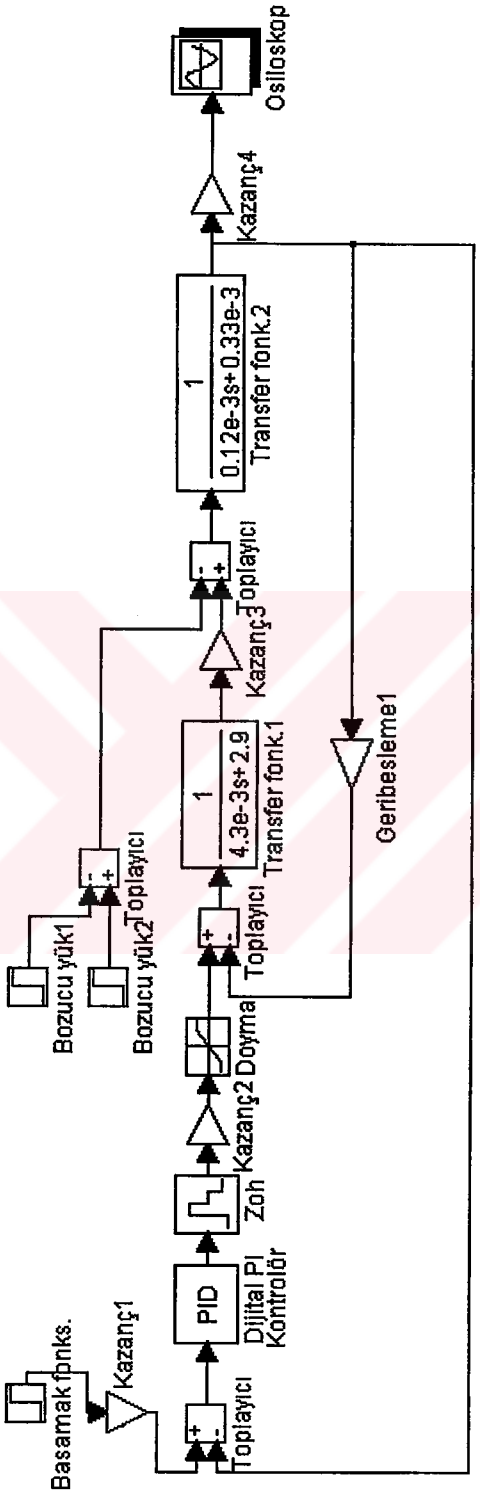
K_i değeri sabit tutulurken K_p değeri 10 değerine kadar arttırıldığı zaman, 3.18'de verilen formülden görüldüğü üzere hatanın büyük olduğu başlangıç anında kontrolörün çıkışında oldukça büyük bir kontrol işareti elde edilir. Bu kontrol işaretinin sisteme uygulanması ile yükselme zamanı oldukça küçülür. Fakat hatanın küçülmesi ile 3.18 formulündende anlaşılacağı üzere bu bileşen etkisini kaybetmektedir. K_p parametresinin hatayı hızlı bir yükselme zamanı sonunda daha çabuk küçültmesi sonucunda hatanın integrali küçüleceği için K_p 'nin etkisini yitirdiği zaman kontrol işareti istenen referans değere hemen ulaşamaz. Bu sebeple oturma zamanı artacaktır. Bu durum Şekil 3.15'te gösterilmiştir. K_p 'nin değerinin 100 yapılarak çok arttırılması sonucunda ise kullanılan sistemde motorun kazanmış olduğu ataleti hemen harcayamaması yüzünden sistemin osilasyon yapmasına sebep olacaktır. Bu durum da Şekil 3.16'da verilmiştir.



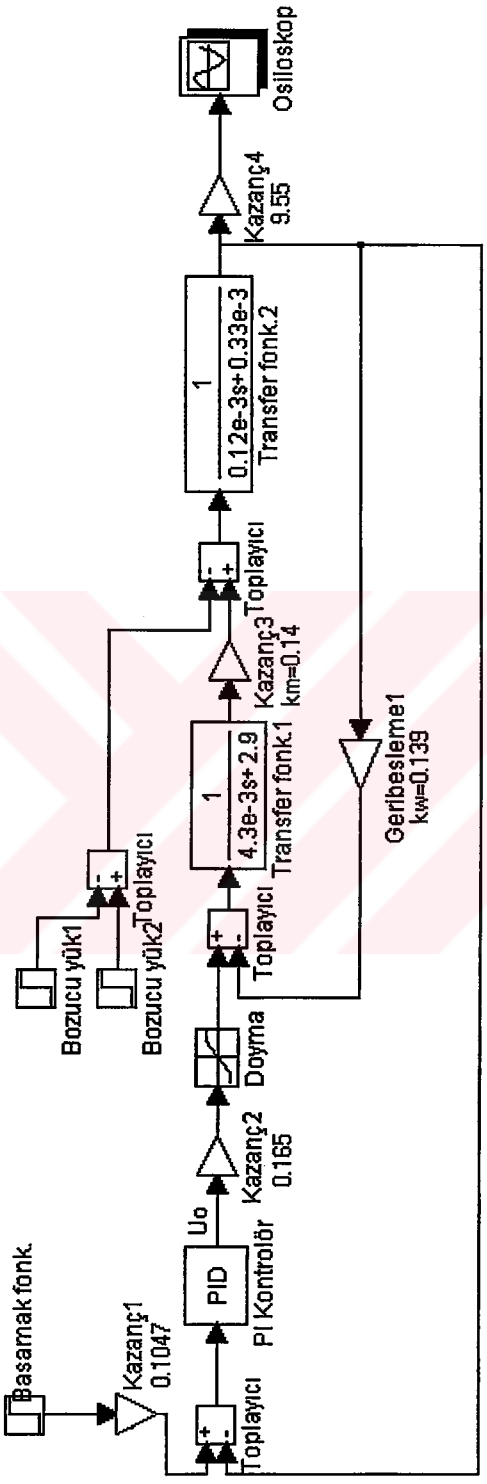
Şekil 3.7 Sistemin kök-yer eğrilerinin MAT_LAB programında çizdirilmesi



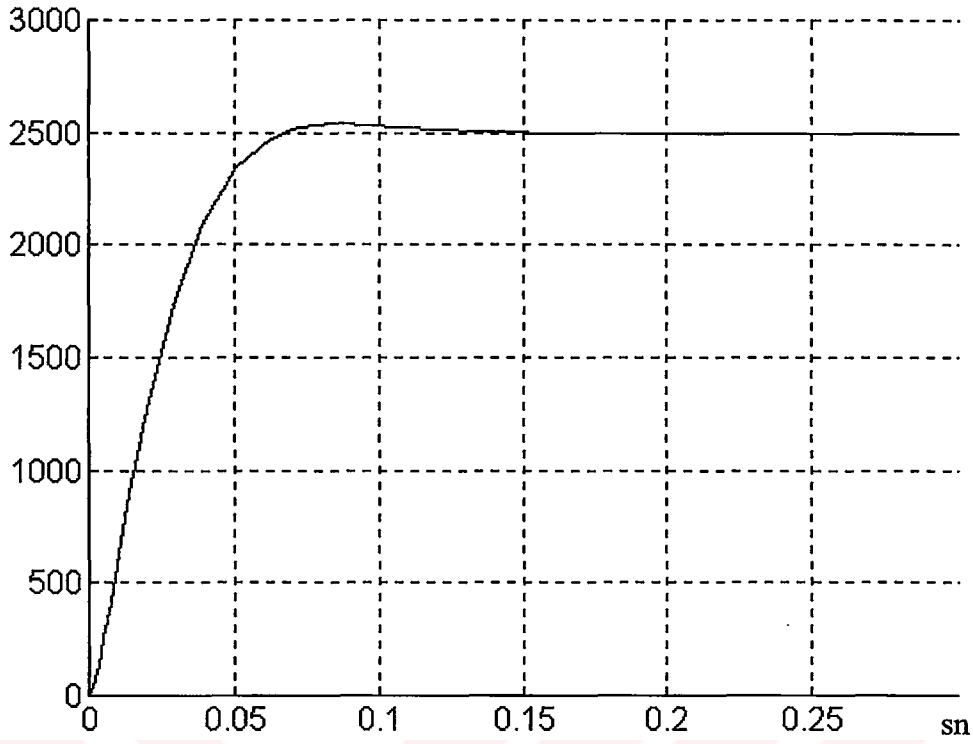
Şekil 3.8 Sistemin yükün devreye alınması ve çıkarılması anlarındaki çıkış işareti



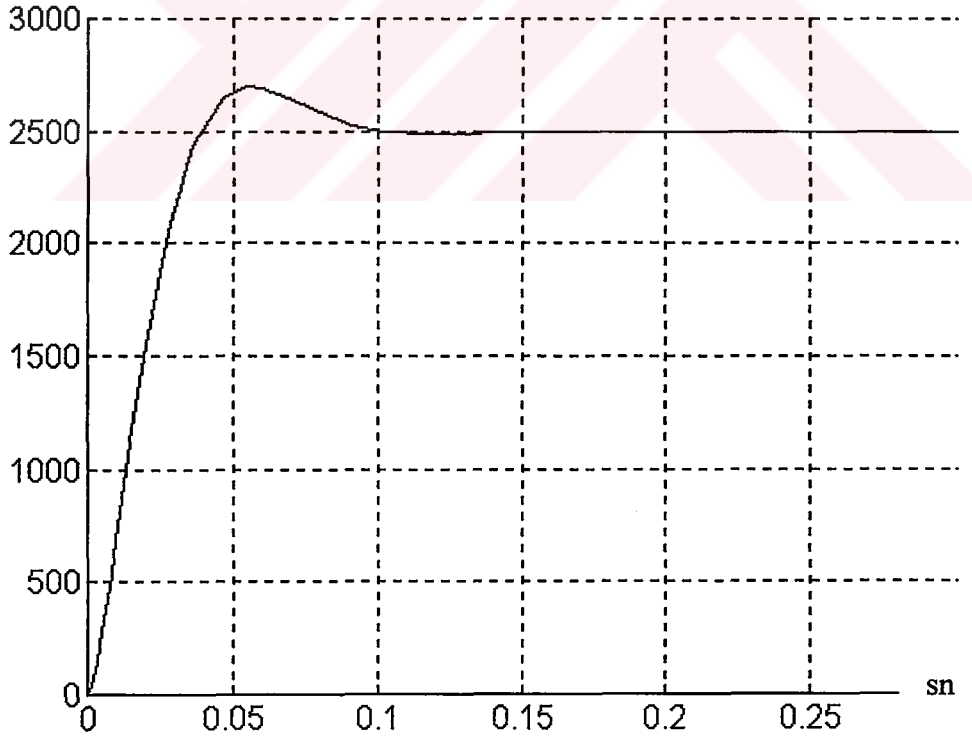
Şekil 3.9 Kontrol sisteminin z domenindeki blok diyagramı



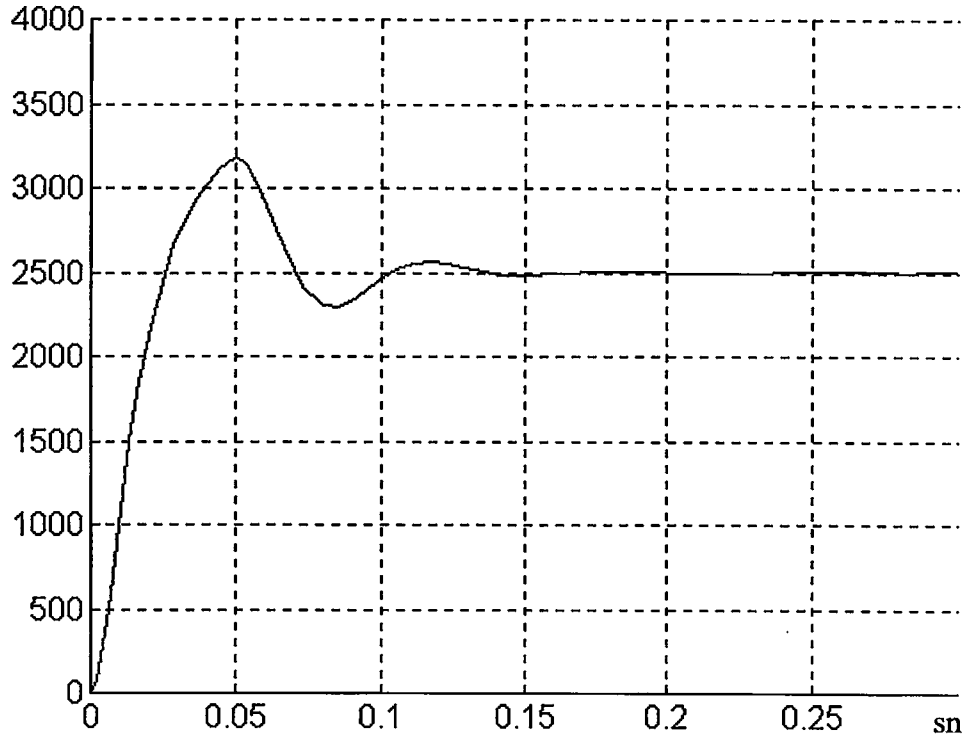
Şekil 3.10 Kontrol sisteminin s domeninde blok diyagramı



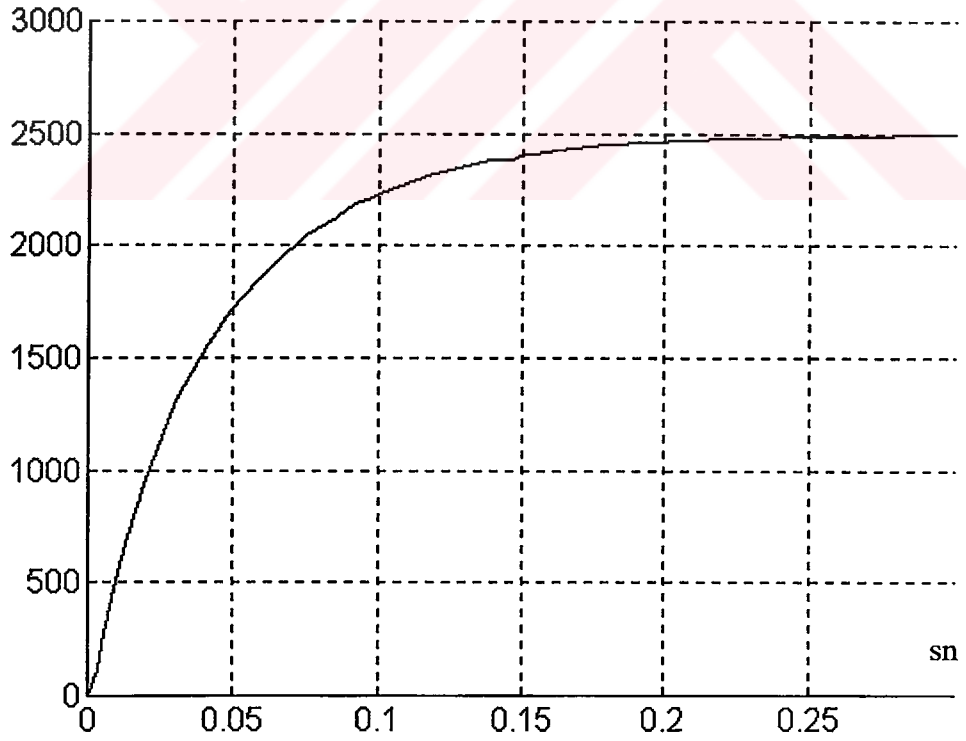
Şekil 3.11 K_p=0.4 ve K_i=40 için sistemin çıkış işareti



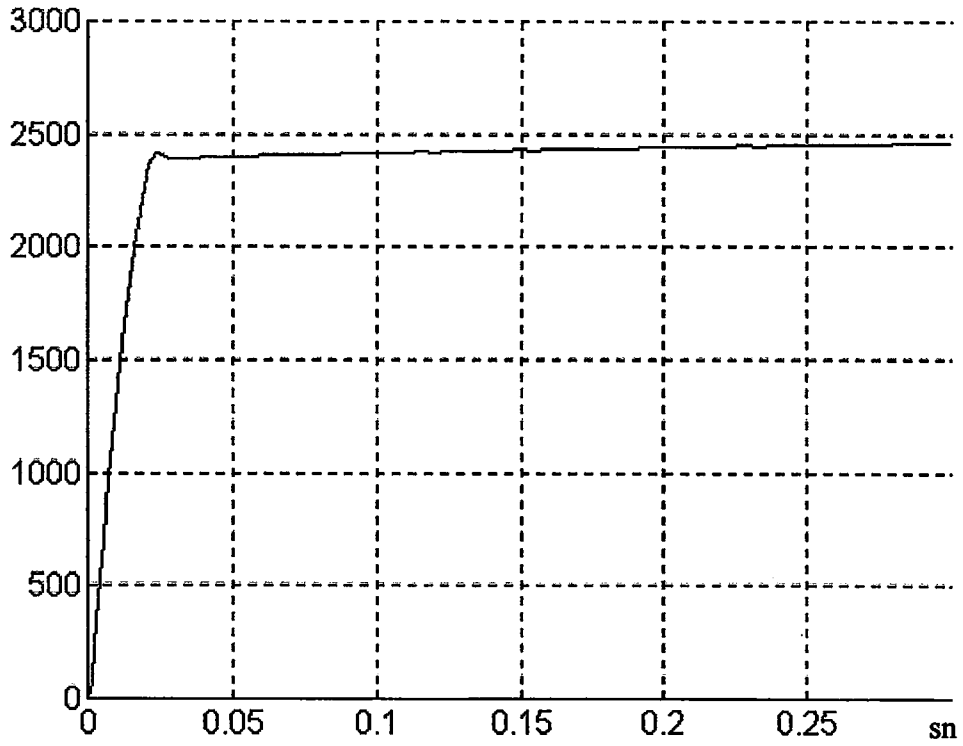
Şekil 3.12 K_p=0.4 ve K_i=80 için sistemin çıkış işareti



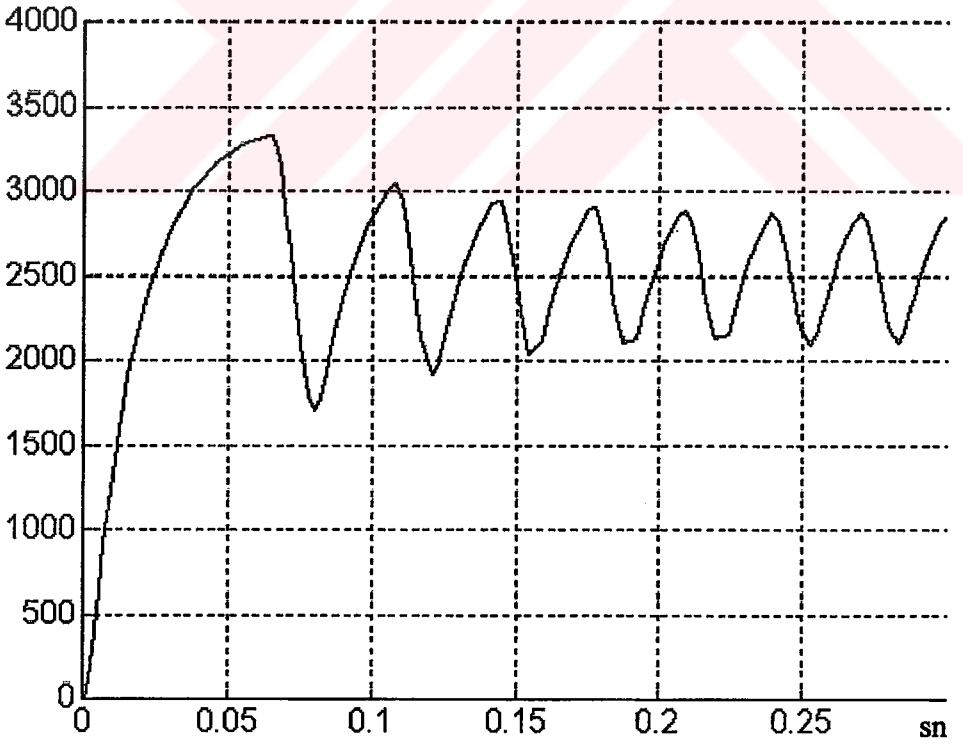
Şekil 3.13 K_p=0.4 ve K_i=100 için sistemin çıkış işareti



Şekil 3.14 K_p=0.4 ve K_i=20 için sistemin çıkış işareti



Şekil 3.15 K_p=10 ve K_i=40 için sistemin çıkış işareti



Şekil 3.16 K_p=100 ve K_i=40 için sistemin çıkış işareti

4. BULANIK MANTIK

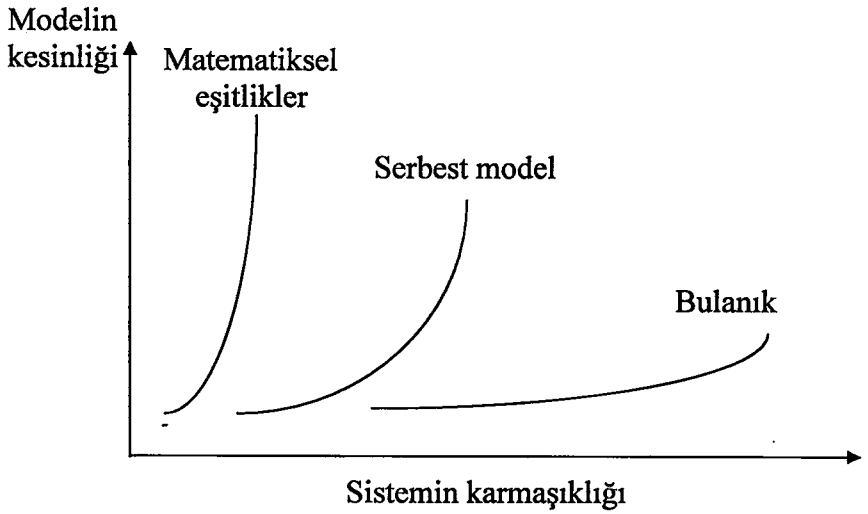
4.1 Giriş

Gerçek hayatta sınırlar, her zaman kesin cümlelerle ifade edilemez. Bu sınırları kesin bir şekilde ifade etmeye çalışmak karmaşaya sebep olur. Dünyadaki karmaşa, belirsizliğin iyi tarif edilmemesinden dolayı artmaktadır. Problemlerin genel özelliklerinden olan karmaşa ve belirsizlik, insanoğlu düşünebildiğinden beri bilinçsiz bir şekilde ifade edilmektedir. Bu birçok yerde mevcut olan özellikler, insanlığın karşılaştığı ekonomik, teknik ve çoğu sosyal konularda görülen problemlerde de karşımıza çıkar. Niçin bilgisayarlar bütün bu gelişmelerden sonra karışık ve belirsiz problemleri çözecek kapasitede tasarlanmıyorlar? İnsanoğlu gerçek sistemlerden nasıl sonuç çıkarabiliyor. Gerçek bir sistemin tamamıyla tanımlanabilmesi için gereken fazla sayıdaki veriyi insanoğlu nasıl aynı zamanda hem anlayıp hem de bir çözüm üretebiliyor? Bütün bunların cevabı bilgisayarlarda olmayan bir düşünme şeklinin insanoğlunda olmasından kaynaklanmaktadır. Bu genellik ve belirsizlik içerisinde, insanoğlunun düşünme sistemi sayesinde, karmaşık sistemleri anlaması, çözümlemesi oldukça hızlı ve kolay olmaktadır.

Yukarıda anlatılanlar ile karmaşa ve belirsizlik arasında ilişki kurulması gerekirse Zadeh'in şu şözü bu karmaşaya bir çözüm sunar; "Gerçek dünyadaki problemlere yakından bakılacak olursa, bulanıklık çözüm haline gelir" (Zadeh, 1973).

Sistem hakkında ne kadar çok bilgimiz olursa sistemin karmaşıklığı o derecede azalırken, sistemin anlaşılması o derecede kolaylaşacaktır. Karmaşıklığın azalmasıyla, sistemin modellenmesinde kesin hesaplama yöntemleri daha kullanışlı hale gelecektir. Şekil 4.1'de sistemin karmaşıklığına göre kullanılacak sistem yapıları verilmiştir. Karmaşanın az olduğu sistemlerde, matematiksel ifadelerle sistemin tanımlanması yeterli olmaktadır. Daha karmaşık ama gerekli tanımlayıcı bilgilerin olduğu sistemlerde ise serbest-model yöntemleri kullanılabilinir. Bu modele örnek vermek gerekirse yapay sinir ağlarını sayabiliriz. Bu ve benzeri yöntemlerde, öğrenme algoritmaları vasıtasıyla daha önce elde edilen sistemi tanımlayıcı bilgiler baz alınarak, belirsizliği bir miktar azaltmayı başarmaktadırlar. Sonuçta az sayıda veri ile belirsiz ve kesin olmayan bir takım bilgilerin olduğu çok karmaşık sistemlerde ise bulanık mantık, gözlenen giriş ve çıkış arasındaki ilişkiyi tanımlamamıza izin vererek sistemin davranışını anlamamızı sağlar. Bulanık sistemlerde belirsizlik oldukça yüksektir.

Buna rağmen bulanık sistemler kesin giriş ve çıkışlara sahiptir. Şekil 4.1(Ross, 1995)'de gösterilen sistemler lineer olmayan eşitlikler, bulanık modelller veya sinir ağları olabilir. Bütün bu modeller gerçek fiziksel dünyanın matematiksel birer özetidir. Önemli nokta ise problemde verilen belirsizliğin karakterine en uygun modelin seçilmesidir.



Şekil 4.1 Sistemin modelindeki kesinlik ile sistemin karmaşıklığı arasındaki ilişki

Bulanık mantık doğal ve insanın düşünme tarzına yakındır. Teorik matematiksel bilim dallarındaki değişkenlerin var veya yok gibi sabit değişimlerine bulanık mantık karşı çıkar. Geleneksel mantıktaki gibi değerin sadece doğru ve yanlış olmasını kabul etmez. Bulanık mantık kısmi doğru ve ara değerli doğrulara sahiptir. **Bu bilim dalı, özellikle sistemi tanımlayan verilerin olmadığı veya çok karışık olduğu için matematiksel modeli olmayan problemlerin çözümünde avantaj sağlar.** Gerçek dünyada konuşulan dilin kullanıldığı bulanık mantık, bilgisayarların düşünme tarzının insan mantığı ile birleştirilmesi neticesinde, mühendislerin karmaşa içerisinde olan sistemleri çözmesine yardımcı olur. Matematiksel model kullanmak yerine sistemin özelliklerinin dilsel terimlerle ifade edildiği bulanık mantığın kullanılması ile sistemin tasarımı ve üzerinde yapılacak değişiklikler basitleşir. Bu da sistemi kontrol edecek programın geliştirme zamanından tasarruf etmeyi, kolay programlamayı ve doğruluğu yüksek bir kontrolün gerçekleştirilmesini sağlar.

Bilim, matematik ve bilgisayarların temeli bool veya diğer adıyla klasik mantıktır. Doğruluğunun yüksek olmasına rağmen bool mantığın bir eksikliği vardır; bu da sınırlarının son derece dar olmasıdır. Bu sebeple insanın düşünme şeklini ifade edemez. Bunu açıklamak için *hız* isimli bir küme tanımlayalım. Bu kümeyi klasik mantıkta tanımlayabilmek için sınırlarını belirlememiz gerekir. Böylece eşik değerin altı için yanlış, üstü içinse doğru

terimini kullanabiliriz. Örnek vermek gerekirse; 100km/h hızlı olmak için eşik değer kabul edilirse, 101km/h hızlı olurken, 99km/h hızlı değildir gibi mantıksız bir sonuç elde edilir. Görüldüğü üzere klasik mantık böyle basit bir durumda oldukça büyük bir hata yapmaktadır. Buna benzer durumlarda insanoğlu kurulan eşik değer ile örnek veri arasındaki ilişkiyi kuvvetlendirerek veya zayıflatarak bir sonuç elde etmeye çalışır. Bulanık mantıkta bu üyelik fonksiyonun derecesi olarak belirtilir (Mikrochip,1994).

Dr. Zadeh'in bulanık kümeler hakkında yeni ufuklar açan (Zadeh, 1965) makalesinden itibaren on sene içerisinde Birleşik Devletler, Avrupa ve Japonyada pek çok kuramsal gelişmeler elde edildi. 1970'lerin ortasından bugüne kadar, teorinin pratiğe geçirilmesinde Japon araştırmacılar en büyük birinci güç oldular. Bu alanda ikibin patentli ürünle, bu teknolojiyi ticari kazanç haline dönüştüren Japonlar mükemmel bir iş başardılar.
Bulanık mantık, birçok teknik alanı ve bilim dalını etkilemektedir. Örnek vermek gerekirse videografiyi sayabiliriz. Fisher, Sanyo ve diğerleri bulanık mantık kullanarak ürettikleri kamera ile bulanık odaklama ve görüntü kararlılığı özelliklerini geliştirdiler. Mitsubishi kişiye göre değişen sıcaklık derecelerini ayarlayan bulanık bir klima gerçekleştirdi. Matsushita akıllı sensörler ile bulanık mantığı birleştiren bir çamaşır makinası üretti. Üretilen bu makinadaki akıllı sensörler; çamaşırın rengini, tipini ve kirlilik miktarını belirlemektedir. Bulanık işlemci suyun sıcaklığını, deterjan miktarını, yıkama ve sıkma zamanlarını altıyüz kombinasyon içerisinden uygun kombinasyonu seçerek gerçekleştirir. Bir Japon şehri olan Sendai'de bulanık bir bilgisayar tarafından kontrol edilen onaltı istasyonlu bir metro sistemi bulunmaktadır. Bulanık mantığın kullanılmasıyla, sürüş konforu arttırılarak sarsıntısız bir sürüş sağlandı. Hızlanma ve yavaşlama esnasında kontrolör, insan operatörlere göre %70 daha az hata yaparak daha verimli olduğunu kanıtladı. Nissan son zamanlarda çıkardığı üst sınıf otomobillerde bulanık otomatik vites ile bulanık anti-patinaj sistemini kullanmaya başladı (Ross, 1995).

4.2 Bulanık Kümeler

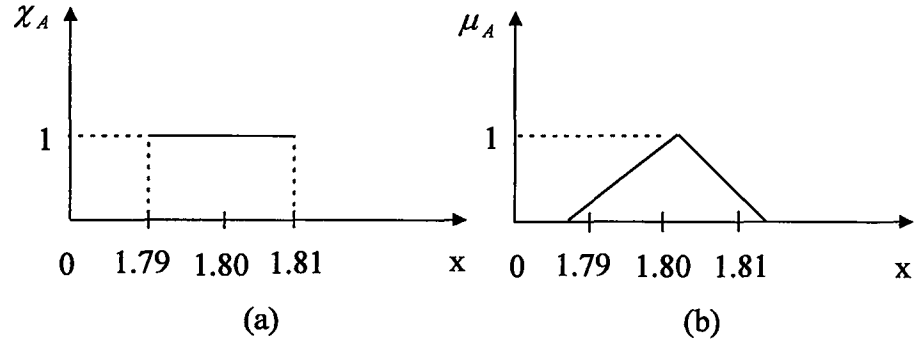
Klasik mantıkta sınırlar çok keskindir. Bu mantıkta bir önermenin olabilirlik derecesi var (1) veya yok (0) olarak tanımlanır. Bu giriş bölümünde de değindiğimiz üzere, gerçek dünyada büyük hatalara sebep olmaktadır. Bir örnek ile açıklayacak olursak, 1.80m.'den uzun olan bir insan düşünelim. Klasik mantıkta ölçmenin sonucuna bağlı olarak insan uzundur veya değildir. Bunun ara bir değeri yoktur. Şayet 1.80m. ve daha yukarısı uzun kümesi olarak

tanımlanırsa, bilgisayar 1.79m. uzunluğundaki bir insanı uzun kümesine dahil etmeyecektir. Bu da bir belirsizlik ve karmaşa yaratacaktır.

Klasik kümede, kümenin bütün elemanlarının üyeliklerinin kesin bir değeri vardır. Bulanık kümede ise küme elemanlarının kesin olmayan, bulanık değerleri bulunmaktadır. Buna örnek verecek olursak; uzunluk kümesinin elemanları 1.79, 1.80 ve 1.81m olsun. 1.80m bölgesi civarında uzunluk kümesi bulanıktır. Ayrıntılı bir şekilde ele alırsak, herbiri kendine özgü x elemanlarından oluşan geniş bir küme olduğunu varsayalım. Bu elemanların bir X uzayı tarafından kapsandıklarını kabul edeceğiz. Bu herbiri ayrı olan elemanların değişik kombinasyonları farklı kümeleri meydana getirecektir. Uzaydaki bu kümelerden biri A kümesi olsun. X uzayındaki gerçel değerlerden oluşan x elemanları A kümesinin üyesidirler veya üyesi olmayabilirler. Üyelik işleminin klasik mantıkta matematiksel gösterilişi işaret fonksiyonu ile ifade edilir.

$$\chi_A(x)=\begin{cases}1, & x \in A \\ 0, & x \notin A\end{cases} \tag{4.1}$$

Burada $\chi_A(x)$ sembolü x elemanının A kümesindeki üyeliğinin kesin değerinin bir işaretidir. $\in$ ve $\notin$ sembolleri sırasıyla elemanıdır, elemanı değildir demektir. Bizim örneğimizde bulunan uzun insanlar kümesi için 1.79m ile 1.81m arasındaki bütün insanlar A kümesinin elemanıdır. A kümesi Şekil 4.2a'da gösterilmiştir. Buradan bir örnek olarak x_1 1.80m. uzunluğundaki bir insandır. Bu ferdi elemanın A kümesindeki üyelik derecesi 1'dir. Yani bu eleman A kümesinin bir elemanıdır. Sembolik olarak ifade edilmesi gerekirse; $\chi_A(x_1)=1$ olarak yazılır. Diğer bir birey x_2 olsun. Bu elemanın ifade ettiği gerçek değer 1.78m uzunluğundaki insan olsun. A kümesinde böyle bir eleman olmadığı için üyelik derecesi 0'dır. Bunu da sembolik olarak ifade edersek; $\chi_A(x_2)=0$ olarak Şekil 4.2a'da görülür. Klasik küme teorisinin kuralları geçerli olduğu bu durumda eleman ya kümenin elemanıdır ya da değildir.



Şekil 4.2 Uzunluk için üyelik fonksiyonları (a) kesin A kümesi (b) bulanık A kümesi

Zadeh klasik küme teorisi fikrini daha da geliştirerek, 0 ile 1 aralığında üyeliğin derecelere sahip olabileceğini gösterdi. Bu derecelikte 1 tam üyeliği ifade ederken, 0 ise bu elemanın bu kümeye üye olmadığını göstermektedir. Klasik küme anlayışında olduğu gibi burada da bir işaret fonksiyonu bulunmaktadır. Bulanık kümelerde uzayda bulunan kümeler içerisindeki bir x elemanı için üyelik derecesi 0 ile 1 arasında çok yüksek çözünürlükte bir değer almaktadır. X uzayındaki kümelerin üyelik derecesi ile birleştirilmesi sonucunda bu kümelere Zadeh tarafından bulanık kümeler adı verilmiştir. Uzunluk üzerine verilmiş örnekle devam edecek olursak; A kümesi uzunluğu **1.80m'ye yakın olanlar** diye bir küme olduğunu kabul edelim. 1.80m'ye yakın olma özelliği bulanıktır. A kümesi için tek bir üyelik fonksiyonu yoktur. A kümesi üyelik fonksiyonu μ_A ile gösterilecektir. Bu kümede 1.80m uzunluğundaki insan için üyelik fonksiyonu $\mu_A(1.80)=1$ olacaktır. 1.80m'nin yakınındaki simetrik sayılar aynı üyelik fonksiyonuna sahiptir. (Bezdek,1993)

Üyelik fonksiyonu Şekil 4.2b'de gösterilmiştir. Kesin ve bulanık kümeler arasındaki en büyük fark kesin kümenin tek bir üyelik fonksiyonu varken, bulanık kümenin çok sayıda üyelik fonksiyonu vardır (Zimmermann, 1991).

Üyelik fonksiyonu küme içindeki üyeliğin matematiksel gösteriminin somutlaştırılmış halidir. Bu tezde bulanık kümeler $\tilde{A}$ ile gösterilecektir. Haritalama fonksiyonu ise;

$$\mu_A(x) \in [0,1] \tag{4.2}$$

olarak verilir. Burada $\mu_A(x)$, A bulanık kümesi içerisindeki x elemanının üyelik derecesini göstermektedir. $\mu_A(x)$'nın değeri birim aralıkta değişen x elemanının A bulanık kümesine % olarak ne kadar ait olduğunu gösterir.

Bulanık küme için kabul edilen notasyon, X uzayında ayrık ve sonlu olursa A bulanık kümesi;

$$A = \left\{ \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \cdots \right\} = \left\{ \sum_i \frac{\mu_A(x_i)}{x_i} \right\} \tag{4.3}$$

şeklinde gösterilir. X uzayı sonsuz ve sürekli olursa A bulanık kümesi;

$$A = \left\{ \int \frac{\mu_A(x)}{x} \right\} \tag{4.4}$$

olur. Her iki notasyonda da yatay çizgiler bölme işlemi değildir. Bu çizgiler, eleman ile onun üyelik fonksiyonu arasındaki ilişkiyi göstermek üzere kullanılmıştır. Eşitliklerde verilen her terimin payı A bulanık kümesindeki üyelik fonksiyonu olup, paydada ise üyelik fonksiyonu ile bağlantılı olan eleman bulunmaktadır. 4.3 eşitliğinde veya ilk notasyonda, toplama sembolü cebirsel toplama sembolü değildir. Bu sembol daha çok her elemanın biraraya gelmesini, toplanmasını ifade etmektedir. İkinci notasyonda ise integral sembolü cebirsel integralden çok sürekli değişkenler için kuramsal sürekli bir fonksiyonu göstermektedir.

4.2.1 Bulanık küme işlemleri

X uzayında sırasıyla A, B ve C olarak üç bulanık küme tanımlayalım. Uzayda verilen bir x elemanı için aşağıda birleşme, kesişme ve evrik işlemleri yine X uzayındaki A, B ve C kümeleri vasıtasıyla tanımlanacak olursa;

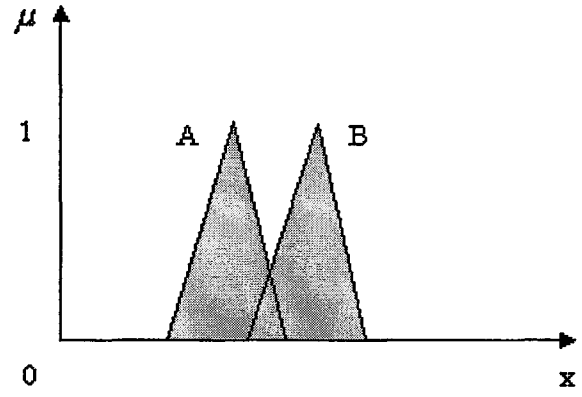
Birleşme $$\mu_{A\cup B}(x) = \mu_A(x) \cup \mu_B(x) \tag{4.5}$$

Kesişme $$\mu_{A\cap B}(x) = \mu_A(x) \cap \mu_B(x) \tag{4.6}$$

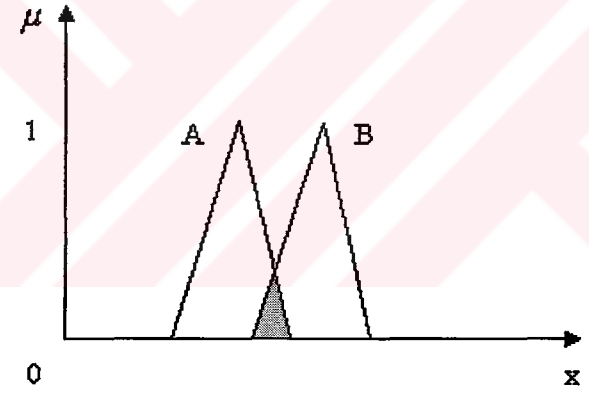
Evrik

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{4.7}$$

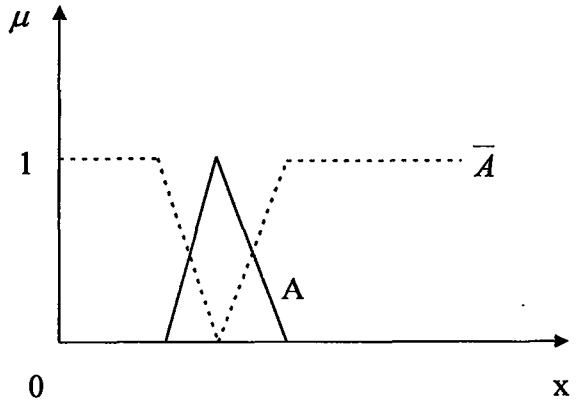
elde edilir. Bu işlemlerle ilgili Venn diyagramları ile işlemlerin sonucunda oluşan yeni bulanık kümeler Şekil 4.3 - 4.5'de gösterilmiştir. X uzayında tanımlı herhangi bir bulanık A kümesi aynı zamanda uzayın bir alt kümesidir. Bu tanımlama klasik küme içinde geçerlidir.



Şekil 4.3 Bulanık A ve B kümelerinin birleşimi



Şekil 4.4 Bulanık A ve B kümelerinin kesişimi



Şekil 4.5 Bulanık A kümesinin evriği

Boş kümedeki herhangi bir x elemanının üyelik değeri 0 iken bütün küme X'teki değeri 1'dir. Boş ve tam kümeler bulanık kümeler değildir. Bu anlatılanlar için yaklaşık notasyonlar aşağıda verilmiştir.

$$A \subseteq X \Rightarrow \mu_A(x) \leq \mu_X(x) \tag{4.8}$$

$$\text{Bütün } x \in X, \mu_{\varnothing}(x) = 0 \tag{4.9}$$

$$\text{Bütün } x \in X, \mu_X(x) = 1 \tag{4.10}$$

Bütün bulanık kümelerin ve X'in bulanık alt kümelerinin toplamı bulanık güç kümesi P(X)'i verir. De Morgan kuralları bulanık kümeler için de klasik kümelerde olduğu gibidir. Bulanık kümeler için bu kurallar aşağıdaki gibi gösterilir.

$$\overline{A \cap B} = \overline{A} \cup \overline{B} \tag{4.11}$$

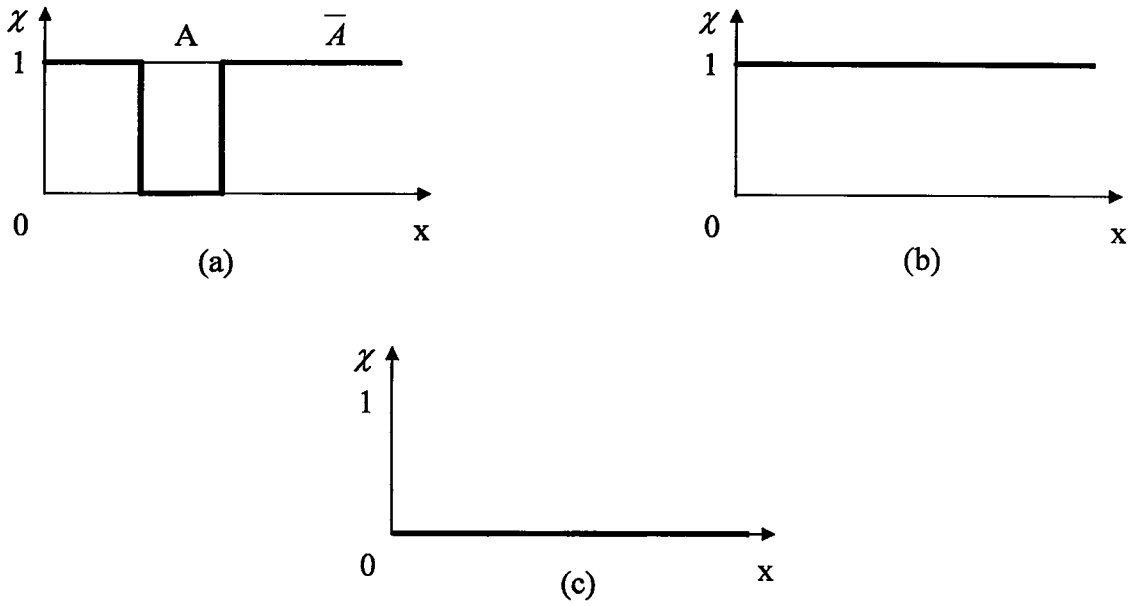
$$\overline{A \cup B} = \overline{A} \cap \overline{B} \tag{4.12}$$

Daha önce de belirttiğimiz gibi, bulanık kümelerde orta kanunlar haricindeki diğer işlemler klasik kümelerde olduğu gibidir. İki bulanık küme üstüste geldiği ve evriği ile üstüste geldiği zaman bu kurallar klasik kümede geçerli olan kurallardan farklı hale gelir. Orta kanunlar bulanık küme için ifade edilecek olursa;

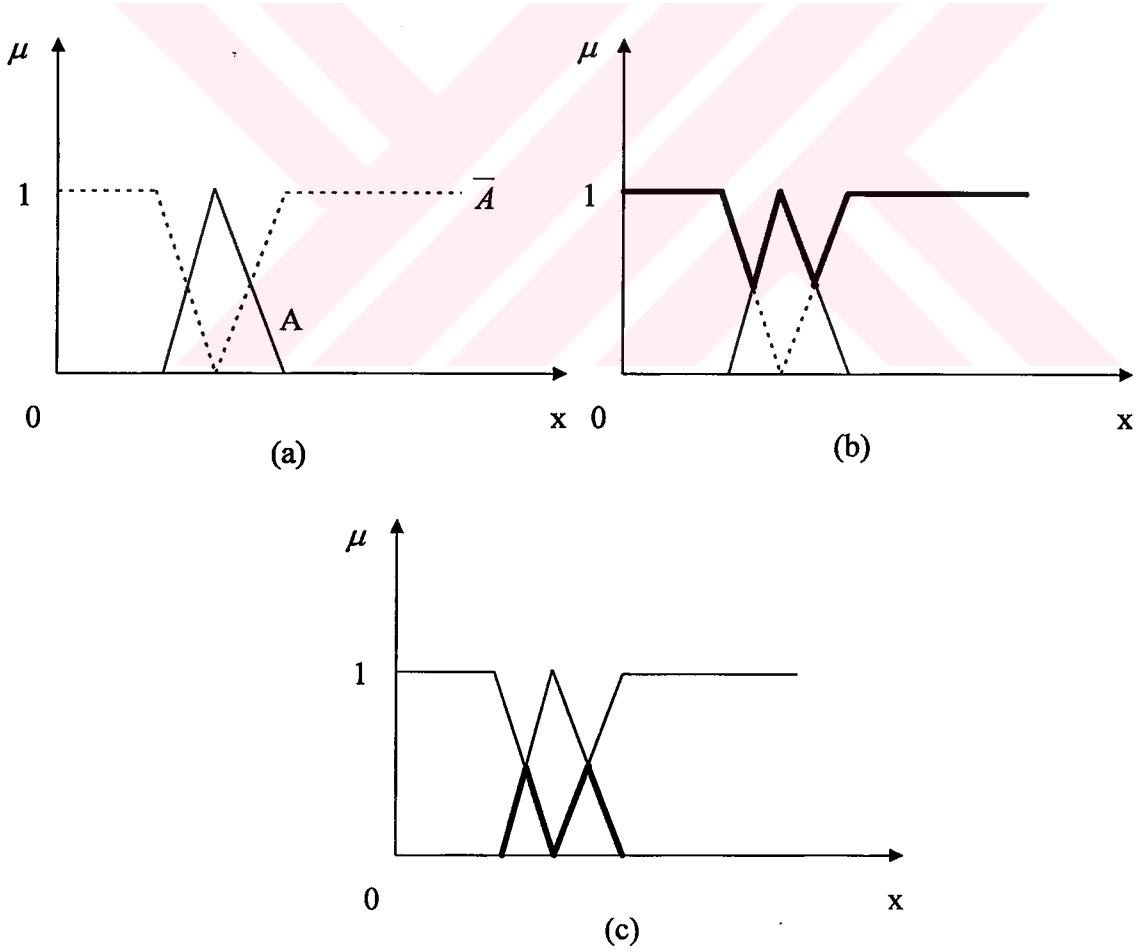
$$A \cup \overline{A} \neq X \tag{4.13}$$

$$A \cap \overline{A} \neq \varnothing \tag{4.14}$$

elde edilir. Klasik ve bulanık kümeler için orta kuralların gösterildiği genişletilmiş Venn diyagramları Şekil 4.6 - 4.7'de sırasıyla gösterilmiştir.



Şekil 4.6 Kesin kümeler için orta kanunlar (a) A kesin kümesi ve evriği (b) $A \cup \overline{A} = X$ (c) $A \cap \overline{A} = \varnothing$



Şekil 4.7 Bulanık kümeler için orta kanunlar (a) Bulanık A kümesi ve evriği (b) $A \cup \overline{A} \neq X$

4.2.2 Bulanık kümelerin özellikleri

Bulanık kümeler klasik küme anlayışında yer etmiş olan özelliklere sahiptir. Bunun sebebi; üyelik değerlerinin 0 ile 1 aralığında değişmesinden kaynaklanmaktadır. Klasik kümeler bulanık kümelerin özel bir şekli olarak düşünülebilinir. Bulanık kümelerde çoğunlukla kullanılan özellikler aşağıda verilmiştir.

Yer değiştirme

$$A \cup B = B \cup A \tag{4.15}$$

$$A \cap B = B \cap A \tag{4.16}$$

Birleşme

$$A \cup (B \cup C) = (A \cup B) \cup C \tag{4.17}$$

$$A \cap (B \cap C) = (A \cap B) \cap C \tag{4.18}$$

Dağılma

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \tag{4.19}$$

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \tag{4.20}$$

$$A \cup A = A \text{ ve } A \cap A = A \tag{4.21}$$

Benzerlik

$$A \cup \varnothing = A \text{ ve } A \cap X = A \tag{4.22}$$

$$A \cap \varnothing = \varnothing \text{ ve } A \cup X = X \tag{4.23}$$

Kapsama

$$A \subseteq B \subseteq C \Rightarrow A \subseteq C \tag{4.24}$$

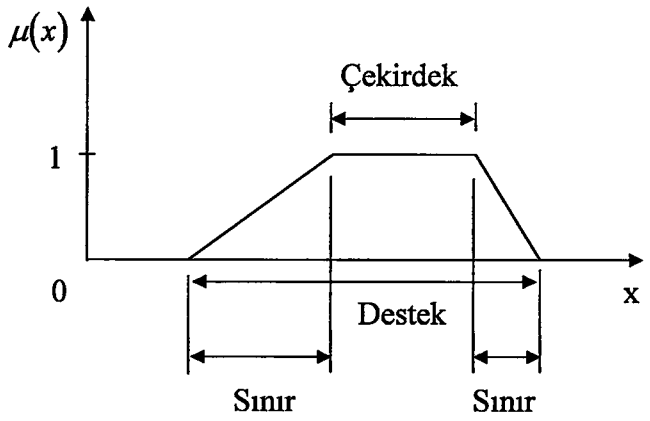
Evrik

$$\overline{\overline{A}} = A \tag{4.25}$$

4.3 Üyelik Fonksiyonları

Evrensel kümenin bir alt kümesi olan bulanık kümenin, üyelerine karşılık gelen ağırlık değerleri bir eğri şeklinde gösterilebilir. Bu eğriler üyelik fonksiyonu olarak adlandırılır.

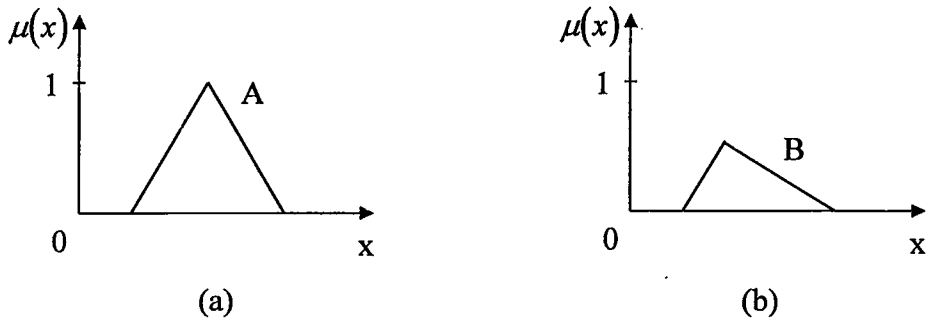
Üyelik fonksiyonunu tanımlayabilmek için bir takım terimlerin açıklanması gerekmektedir. Açıklanan bu terimler Şekil 4.8'de gösterilmiştir.



Şekil 4.8 Bulanık kümenin sınırları, çekirdek ve desteği

A bulanık kümesinde evrensel bölge olarak tanımlanan ve tam üyeliğe yani $\mu_A(x) = 1$ olduğu bölgeye **çekirdek** adı verilir. A bulanık kümesinde üyeliği sıfırdan farklı olan yani $\mu_A(x) = 0$ olduğu bölgeye **destek** adı verilir. A bulanık kümesinde üyelik değeri sıfır ve birden farklı olduğu yani $0 < \mu_A(x) < 1$ şarlarının sağlandığı bölgeye **sınır** adı verilir. Evrensel küme içerisinde bulunan, bulanık kümeyi oluşturan elemanların kısmi üyeliği veya tam üyeliği bulunmaktadır (Dubois ve Prade, 1980).

Üyelik fonksiyonunda, evrensel kümede bulunan ve üyelik değeri bir olan en az bir elemanın bulunduğu bulanık kümelere normal bulanık küme olarak adlandırılır.Üyelik değeri bir olan bu elemana kümenin **prototipi** veya **prototopik eleman** adı verilir. Şekil 4.9'da **normal** ve **normal altı** bulanık kümeleri gösterilmiştir.

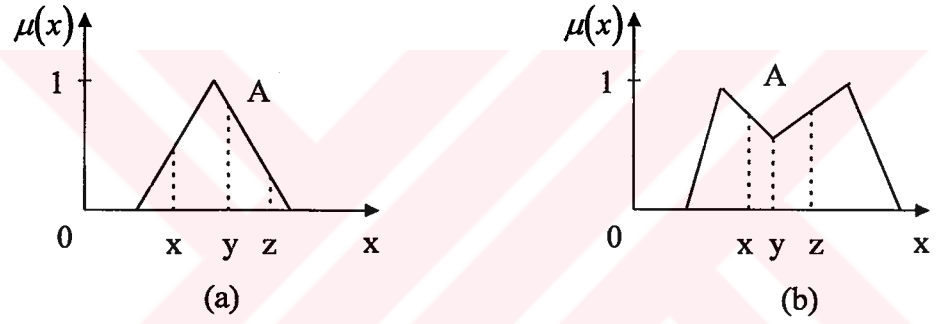


Şekil 4.9 (a) Normal bulanık küme, (b) Normal altı bulanık küme

Normal altı bulanık kümede ise normal bulanık kümelerin aksine üyelik değeri bir olan hiçbir elemana sahip değildir. Üyelik değerlerinin tam olarak monoton arttığı ve monoton azaldığı üyelik fonksiyonlarına sahip bulanık kümelere dışbükey bulanık kümeler adı verilir. Bunu başka bir şekilde ifade edecek olursak; x, y ve z bulanık kümenin elemanları olsun. Aralarındaki ilişki x<y<z ise,

$$\mu_A(y) \geq \min\left[\mu_A(x), \mu_A(z)\right] \tag{4.25}$$

şeklinde elde edilir. Burada min bir bulanık mantık operatörüdür. İki üyelik fonksiyonu olan $\mu_A(x)$ ve $\mu_A(z)$'den değeri küçük olanı çıkış değeri olarak verir. Eğer üyelik fonksiyonu yukardaki şartı sağlıyorsa dışbükey bulanık bir kümedir (Zadeh,1965). Şekil 4.10'da dışbükey olan ve olmayan bulanık kümeler gösterilmiştir.



Şekil 4.10 (a) Dışbükey normal bulanık küme (b) Dışbükey olmayan normal bulanık küme

A ve B kümeleri farklı özelliklere sahip dışbükey iki bulanık küme olsun. Bu iki kümenin kesişmesi yine dışbükey bir bulanık küme olacaktır. Üyelik fonksiyonunda üyelik değeri 0.5 olan elemanın bulunduğu noktaya **geçiş noktası** adı verilir. A bulanık kümesinin en yüksek noktası, üyelik fonksiyonunun maksimum değeri olarak tanımlanır. Bu değer $\max\{\mu_A(x)\}$ ile belirtilir. Şayet bu kümenin maksimum üyelik değeri birden küçükse, daha önce de belirttiğimiz üzere bu tür bulanık kümelere normal altı kümeler adını veriyoruz.

4.4 Bulanıklaştırma

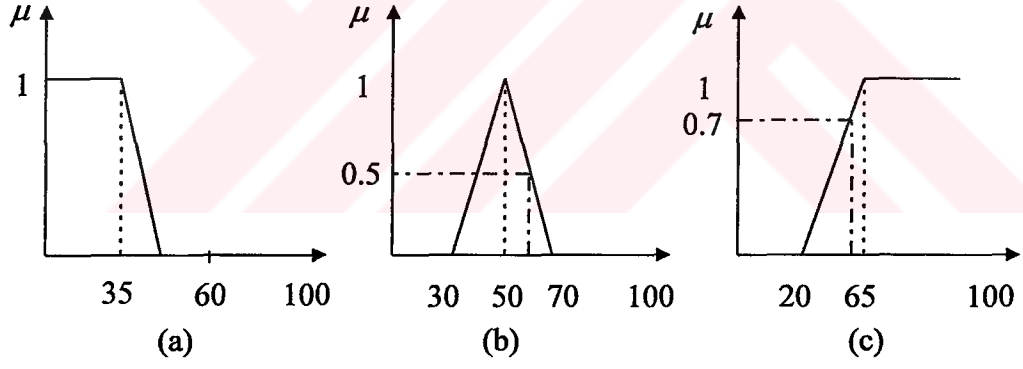
Bulanıklaştırma işlemi; kesin değerleri bulanık değerler haline getirme işlemidir. Biz bu olaya basitçe, kesin ve kararlı olarak bildiğimiz bazı değerlerin gerçekten kararlı olmaması ve bu kararsızlığın ortaya konulmasına bulanıklaştırma diyebiliriz.

Bulanıklaştırma işlemi tanımlanan kesin verilere karşı düşen yaklaşık üyelik fonksiyonlarının bulunması olarak da tanımlanabilir. Örnek vermek gerekirse; dilsel değişken olarak hız değişkenini ele alalım. Hız değişkeninin üç terimden oluştuğunu kabul edersek;

T (hız) = {yavaş, orta, yüksek} (4.26)

olarak sınıflandırabiliriz. Skalayı 0 ile 100 arasında aldığımızda, yavaş hız 35 in üstünde bir değere kadar, orta hız 30-70 arasında ve yüksek hız 65-100 arasında olduğu varsayılır. Bu üç bulanık değişken için bir çok üyelik fonksiyonu tanımlanabilir. Şekil 4.11'de bu üç bulanık değişken için ayrı ayrı üyelik fonksiyonları çizilmiştir.

Bu üyelik fonksiyonlarını daha iyi açıklayabilmek için bir örnek olarak 60 km/saat hızını ele alalım. Bu hızın, yavaş hızdaki üyeliği sıfır, orta hızdaki üyeliği 0.5 ve yüksek hızdaki üyeliği ise 0.7'dir. Burdan çıkarılacak sonuç; bu hızın %50 orta hız ve %70 yüksek hız kümesine üyeliği olduğudur (Paraskevopoulos,1996).



Şekil 4.11 Hız değişkeni için (a) yavaş (b) orta (c) yüksek terimlerinin üyelik fonksiyonları

4.5. Çıkarım İşlemi

Artık kesin değerler, bulanık üyelik fonksiyonları olarak ifade edildiğine göre bunlara bulanık işlemleri uygulamak artık mümkün hale gelmiştir. Bu işlemler; sistemimizin yapısına uygun ve sistemin istenilen davranışları yapması için yönlendirecek kuralların yazılması olacaktır.

4.5.1 Kural tabanlı sistemler

Yapay zeka alanında bilgiyi ifade edebilmek için çok çeşitli yollar bulunmaktadır. İnsan bilgisini sunabilmek için en alışılagelmiş yöntem, doğal konuşma formu ile yapılan ifade şeklidir. Bunun yapısını açıklamak gerekirse;

Eğer önerme (varsayım), **İse** sonuç (4.27)

şeklinde yazılabilir. 4.27'de verilen yapı **Eğer-İse** kural tabanı olarak bilinir. Bu yapı çıkarım işlemini ifade eder. Eğer gerçeği (hipotezi, varsayımı) bilirsek, bundan çıkarım yapabilir veya türetebilir ve sonuca ulaşabiliriz. Bu yapıda gerçekleştirilen bilgiyi ifade etme **yüzeysel bilgi** olarak adlandırılır ve dilsel ifade yapısına oldukça uygundur. Bunun sebebi, insanın iletişim kurmak için kullanmış olduğu dil yapısında, insan tecrübesini, olayları anlama ve kavrama yeteneğini ifade edebilmesidir. Çevremizdeki objelerin davranışları, fonksiyonları, yapıları ve sezgileri ile birleştirildiğinde, bilginin çok derin formları elde edilmez. Çünkü bu tür yapıdaki bilgiler elde edilse bile kolaylıkla dilsel terimlere indirgenemez. Dilsel kontrol kurallarının elde edilmesindeki en genel ve alışılagelmiş kaynak insan deneyimleridir. Bu kurallar sistem hakkında, sistemi kontrol edebilecek bütün bilgileri ihtiva etmektedir. Böylece sistemin o anki bulanık girişine uygun kural kullanılarak sisteme uygun bir kontrol işareti elde edilir (Paraskevopulos,1996).

4.5.1.1 Kanonik kural formları

Genelde, dilsel değişkenler üç çeşit yapıdan oluşmaktadır (Vadiee,1993). Bu yapılar sırasıyla, (i) atama yapıları, (ii) şart yapıları ve (iii) şartsız yapılardır. Bunların herbiri ile ilgili örnekler aşağıda verilmiştir.

Atama yapıları

x = büyük

x büyük değildir ve çok küçük değildir

Mevsim = kış

Dış sıcaklık = sıcak

Şart yapıları

Eğer x çok sıcak **İse** dur

Eğer x büyük **İse** y küçüktür **Değil ise** y küçük değildir

Şartsız yapılar

9'a git

Dur

x'e böl

Basıncı yükseğe ayarla

Atama yapıları değişkenin özel bir değerde olmasını sınırlar. Şartsız yapılar, şartlı yapılardaki giriş koşulunun hep doğru olarak kabul edildiği ve sınırlandırılmış şartlı yapılardaki özel bir yapı olarak kabul edilir. Şartsız yapı olan "çıkış düşük seviyede olsun" aşağıdaki şekilde yazılabilir. **Eğer** şart ne olursa olsun **İse** çıkış düşük seviyede olsun

Bu sebepten, kural tabanı incelendiğinde sınırlı şart yapılarının toplamı olarak tanımlanabilir. Bu yapı aynı zamanda bulanık şart yapıları olarak modellenebilir. Örnek verilecek olursa;

Eğer şart C^1 İse sınırlayan kural R^1

şeklinde yazılır. Şartsız sınırlamalar aşağıdaki yapıda olurlar.

R^1 : Çıkış B^1'dir

ve

R^2 : Çıkış B^2'dir

ve

Burada B^1, B^2,..... bulanık sonuçlardır.

4.5.1.2 Birleşik kuralların ayrıştırılması

Bazı dilsel yapılar, birleşik kural yapıları şeklinde ifade edilirler. Örnek olarak aşağıdaki kuralları içeren, iki girişli bir çıkışlı bir vinç sistemini ele alalım. Bu vinç sistemindeki girişler vincin taşıdığı yükün ulaşılmak istenen noktaya olan **uzaklığı** ve yükün dikey eksenle yapmış olduğu **açı** olsun. Sisteme uygulanacak işaret veya kontrolörün çıkışıda sisteme verilen güç olsun. Buna göre aşağıdaki kuralları türetecek olursak;

Kural 1: **Eğer** "Mesafe" orta **Ve** "Açı" pozitif-küçük **İse** "Güç" pozitif-orta

Kural 2: **Eğer** "Mesafe" uzak **Ve** "Açı" sıfır **İse** "Güç" pozitif-orta

Kural 3: **Eğer** "Mesafe" orta **Veya** "Açı" sıfır **İse** "Güç" sıfır

şeklinde birleşik kurallar elde edilir. Kuralları birleştirmek için **Ve-Veya** bağlaçları kullanılır. Böylece daha etkili kurallar dilsel terimlerle ifade edilebilir.

4.5.1.2.1 Çok katlı birleşik varsayımlar

Kurallar yukarıda anlattığımız gibi etki sahalarını genişletmek için birleşik kural yapılarını kullanırlar. Bu birleştirme işlemi için **Ve** bağlacı kullanılır.

Eğer x A^1 **Ve** A^2....**Ve** A^L **İse** y B^S (4.28)

şeklinde **Ve** bağlacı kullanarak birleşik kural yapısı yazılır. Yeni bir bulanık altküme olan A^S,

$$A^S = A^1 \cap A^2 \cap .... \cap A^L \tag{4.29}$$

olarak ifade edilir. Ve bağlacı üyelik fonksiyonunda min operatörü olarak işlem görür. Min operatörü bulanık kümeler arasında üyelik değeri en küçük değere sahip olan kümenin üyelik değerini sonuç olarak verir.

$$\mu_A(x) = \min\left[\mu_{A^1}(x), \mu_{A^2}(x), ... \mu_{A^L}(x)\right] \tag{4.30}$$

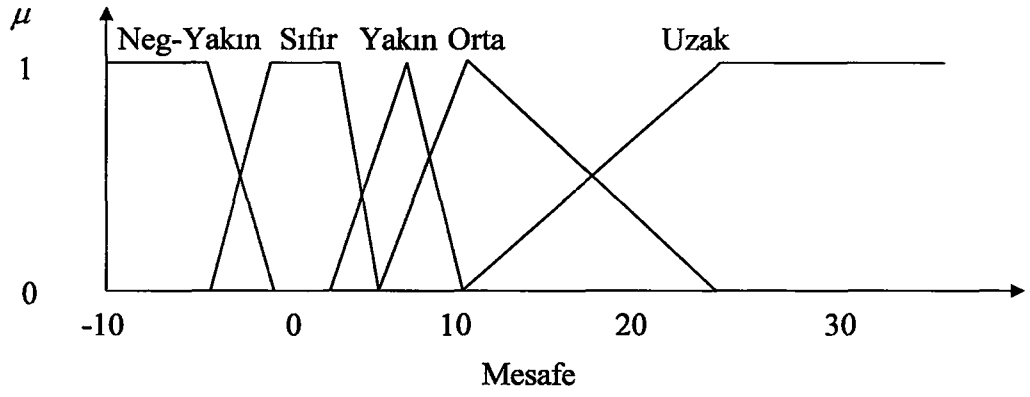
Bunu bir örnekle pekiştirecek olursak bir önceki konuda kullanmış olduğumuz birinci kuralı yeniden ele alalım.

Kural 1: **Eğer** "Mesafe" uzak **Ve** "Açı" positif-küçük **İse** "Güç" pozitif-orta

Sistemimizde iki bulanık giriş değişkeni bulunmaktaydı. Bunlar mesafe ve açı bulanık değişkenleriydi. Mesafe ve açı bulanık değişkenleri Şekil 4.12 ve 4.13'de gösterilmiştir. Vincimizin 12 metre mesafede ve yükün açısının 4 derece olduğunu farzedelim. Tablo 4.1'deki üyelik fonksiyonundan bu mesafenin 0.9 üyelik değeri ile orta mesafe olduğunu elde ederiz. Tablo 4.2'den ise 4 derece için açı üyelik fonksiyonuna bakıldığında 0.8 üyelik fonksiyonu ile positif-küçük olduğu anlaşılır. Bu iki değeri minimum işlemine tabi tutacak olursak;

$$\min(0.9,0.8)=0.8 \tag{4.31}$$

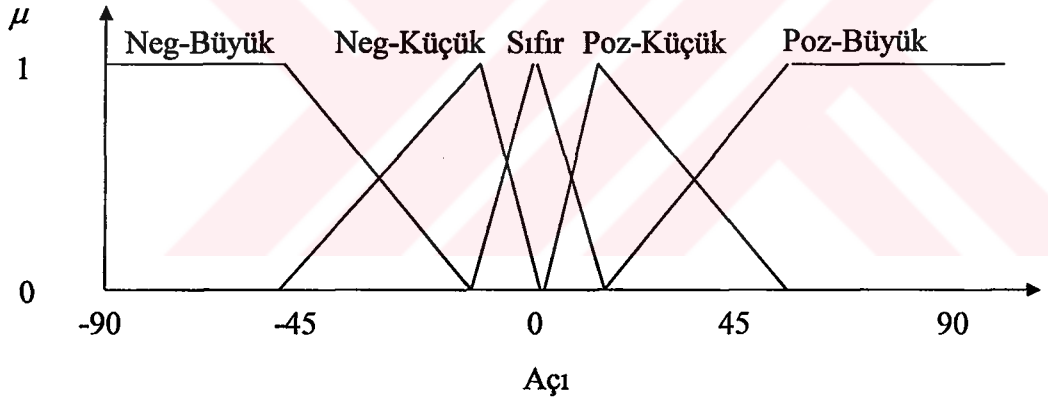
olarak elde edilir.



Şekil 4.12 Mesafe için üyelik fonksiyonu

Çizelge 4.1 12metre mesafe için üyelik fonksiyonu değerleri

Uzak	0.1
Orta	0.9
Yakın	0
Sıfır	0
Neg-Yakın	0



Şekil 4.13 Açı için üyelik fonksiyonu

Çizelge 4.2 4 derece için üyelik fonksiyon değerleri

Neg-Büyük	0
Neg-Küçük	0
Sıfır	0.2
Poz-Küçük	0.8
Poz-Büyük	0

4.5.1.2.2 Çok katlı ayrık varsayımlar

Kuralların farklı birleştirme şekillerinden biri de **Veya** işlemidir.

Eğer x A^1 **Veya** A^2....**Veya** A^L **İse** y B^S (4.32)

şeklinde **Veya** bağlacı kullanarak birleşik kural yapısı yazılır. Yeni bir bulanık altküme olan A^S,

$$A^S = A^1 \cup A^2 \cup .....\cup A^L \tag{4.33}$$

olarak ifade edilir. Veya bağlacı üyelik fonksiyonunda max operatörü olarak işlem görür. Max operatörü bulanık kümeler arasında üyelik değeri en büyük değere sahip olan kümenin üyelik değerini sonuç olarak verir.

$$\mu_A(x) = \max\left[\mu_{A^1}(x), \mu_{A^2}(x), ... \mu_{A^L}(x)\right] \tag{4.34}$$

Bir önceki konuda verilen örneğin aynısı kullanılacak olursa;

max(0.9,0.8)=0.9 (4.35)

olarak elde edilir. (Mikrochip,1994)

4.5.2 Çıkarım işleminin grafik teknikleriyle gerçekleştirilmesi

Daha önceki kısımlarda çıkarım işlemi için kullanılan **Eğer-İse** kurallarından oluşan kural tabanını gördük. Çıkarım işlemi genellikle bilgisayarlarda bu iş için özel hazırlanmış yazılımlar ile gerçekleştirilir. Bazen çıkarım işlemini kontrol edebilmek için veya bazı kuralların sonuçlarından emin olabilmek için bu işlemler manuel olarak da gerçekleştirilebilinir. Bu yöntemi gösterebilmek için iki basit kurala sahip bir sistemi ele alalım. Bu sistemdeki her kuralın iki varsayım ve bir sonuca sahip olduğunu kabul edeceğiz. Bu sistem iki girişli bir çıkışlı bulanık sisteme benzemektedir. Burada gösterilen grafiksel prosedür kolaylıkla geliştirilebilir ve istenilen sayıda giriş (varsayım) ve çıkışa (sonuç) sahip

kural tabanı tarafından kullanılabilinir. Bulanık sistem birbirini etkilemeyen x_1 ve x_2 girişleri (varsayımları) ile tekil y çıkışıyla (sonuç) r sayıda dilsel Eğer-İse kurallarından oluşmuştur.

Eğer x_1 A_1^k VE x_2 A_2^k İse y^k B^k olur. $k = 1,2,....,r$ (4.36)

Burada, A_1^k ve A_2^k k tane varsayıma sahip küme olup B^k ise k tane sonuca sahip bulanık kümelerdir.

Aşağıda iki girişli sistemler için en çok kullanılan iki yöntemi ele alacağız. Bunlar sırasıyla açıklanmıştır.

1) Sistemin girişleri kesin değerlerdir ve **max-min** çıkarım yöntemini kullanmaktadır.

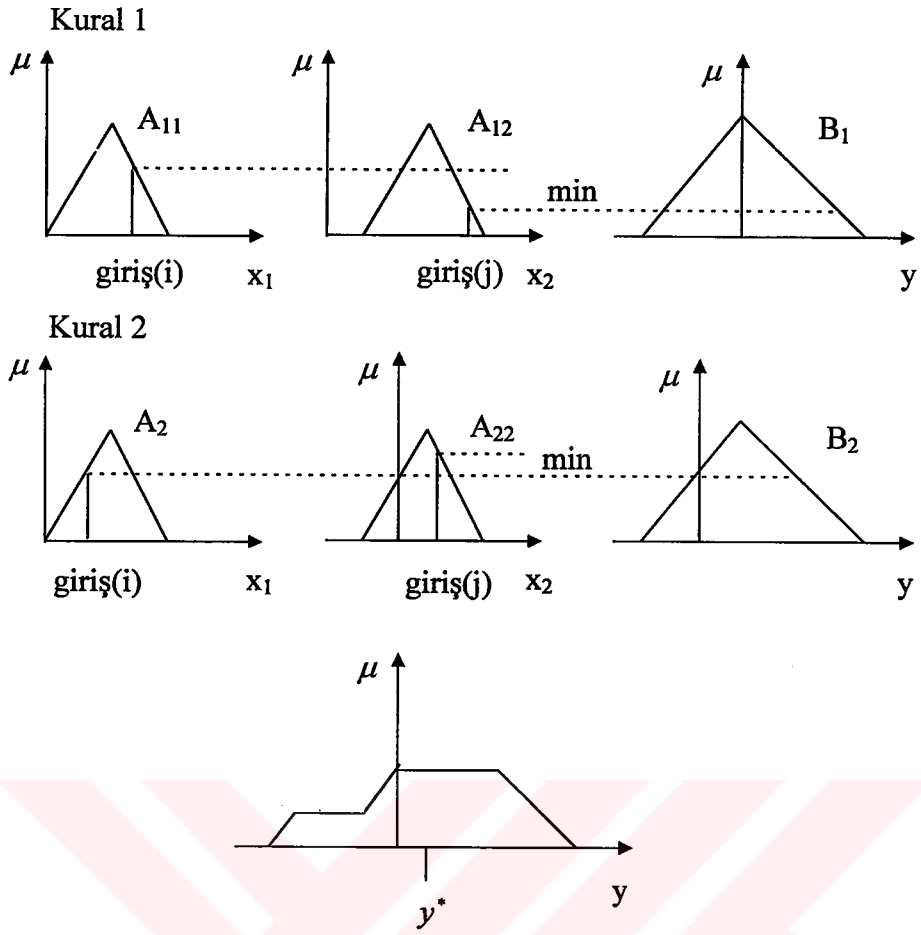
2) Sistemin girişleri yine kesin değerlerdir. Fakat çıkarım yöntemi olarak **max-prod** metodu kullanılmaktadır.

Durum1. x_1 ve x_2 girişleri üçgen (delta) fonksiyonu şeklinde kesin değerlerdir. Kural tabanı 4.30 eşitliğinde tanımlanmıştır. r sayıda kural için birleşik kurallar için elde edilecek olan çıkış aşağıdaki eşitlik ile verilir.

$$\mu_{B^k}(y) = \max_k \left[\min \left[\mu_{A_1^k}(input(i)), \mu_{A_2^k}(input(j)) \right] \right] \qquad k = 1,2,....,r \tag{4.37}$$

4.37 eşitliğinin grafiksel olarak Şekil 4.14'de görüldüğü üzere çok basit bir yorumu vardır. Şekil 4.14'de iki kuralın grafiksel analizi görülmektedir. Burada A_{11} ve A_{12} sırasıyla ilk kuralın birinci ve ikinci bulanık varsayımları ifade etmektedir. B_1 sembolü ise birinci kuralın bulanık sonucunu göstermektedir. A_{21} ve A_{22} sırasıyla ikinci kuralın birinci ve ikinci bulanık varsayımlarını ifade etmektedir. B_2 sembolü ise ikinci kuralın bulanık sonucunu göstermektedir. 4.37'deki minimum fonksiyonu Şekil 4.14'de ortaya çıkmış her kural için ayrı ayrı uygulanmıştır. Sistem için genel kural yapısı içerisinde verilen varsayım çifti lojik VE yapısı ile birbirine bağlanmıştır.

Seçilen kural için varsayımlardan en küçük üyelik değerine sahip olan girişin işaret ettiği sonuç alınır. Her kural için bu işlem yapılır. Daha sonra her kural için elde edilen çıkışların maksimum değerleri toplanarak çıkış grafiği elde edilir. Bütün bunlardan sonra bu grafik yardımıyla netleştirme yöntemlerinden biri kullanılarak kesin çıkış elde edilir.

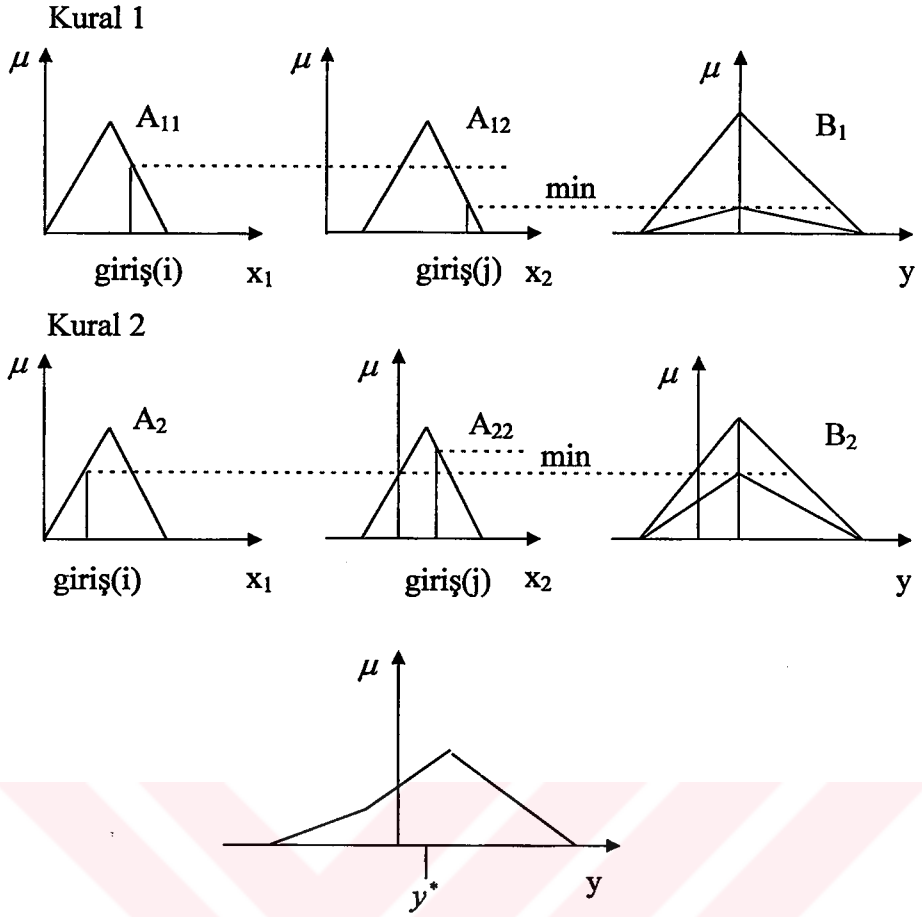


Şekil 4.14 Kesin değerlerle max-min çıkarım metodunun grafiksel gösterimi

Durum2. Önceki örnekte eğer max-prod yöntemini kullansaydık r kurallı bulanık sistem için çıkış;

$$\mu_{B^k}(y) = \max_k \left[\left[\mu_{A_1^k}(input(i)) \cdot \mu_{A_2^k}(input(j))\right]\right] \qquad k = 1,2,....,r \qquad (4.38)$$

şekilde elde edilir. Bunun grafiksel olarak Şekil 4.15'de verilmiştir. Burada ilgili kuralda girişlere min fonksiyonu uygulanır. Küçük üyelik değerine sahip olan girişin üyelik değerine denk düşen çıkış üyelik değerini tepe noktası kabul eden bir üçgen çizilir. Her kural için bu yöntem uygulanır. Daha sonra her kural için elde edilen çıkışların maksimum değerleri toplanarak çıkış grafiği elde edilir. Bütün bunlardan sonra bu grafik yardımıyla netleştirme yöntemlerinden biri kullanılarak kesin çıkış elde edilir (Jamshidi, 1993).



Şekil 4.15 Kesin değerlerle max-prod çıkarım metodunun grafiksel gösterimi

4.6 Netleştirme Yöntemleri

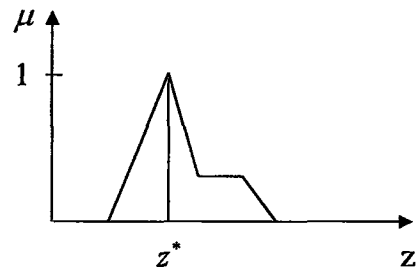
Çıkarım işlemlerinin gerçekleştirilmesi sonucunda hala elimizde bulanık bir değer bulunmaktadır. Oysa fiziksel sistemler kesin değerlere ihtiyaç duyarlar. Bu sebeple bu bulanık değerleri kesin değerlere dönüştürmemiz gerekir. Bu dönüştürme işlemine **netleştirme** adı verilir. Netleştirme işleminde yedi yöntem bulunmaktadır. Aşağıda sırasıyla bunları açıklayacağız. (Hellendom ve Thomas, 1993)

4.6.1 Maksimum üyelik yöntemi

Yükseklik yöntemi olarak da bilinir. Bu yöntemde üyelik fonksiyonundaki en büyük üyelik değeri kesin sonuç olarak alınır. Aşağıdaki şekilde ifade edilir.

$$\mu_C(z^*) \geq \mu_C(z) \qquad \forall z \in Z \tag{4.39}$$

Şekil 4.16'de bu yöntem grafiksel olarak gösterilmiştir.



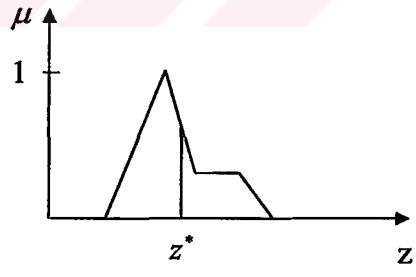
Şekil 4.16 Maksimum üyelik fonksiyonu netleştirme yöntemi

4.6.2 Merkezi yöntem

Bu yöntem ağırlık merkezini bulma veya alan merkezini bulma yöntemi olarak adlandırılır. En çok tercih edilen netleştirme yöntemlerinden biridir. Aşağıdaki şekilde ifade edilebilir.

$$z^* = \frac{\int \mu_C(z) \cdot z \; dz}{\int \mu_C(z) dz} \tag{4.40}$$

Şekil 4.17'de bu yöntem grafiksel olarak gösterilmiştir.



Şekil 4.17 Merkezi netleştirme yöntemi

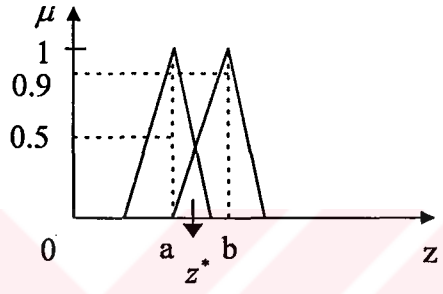
4.6.3 Ağırlık ortalaması yöntemi

Bu yöntem sadece çıkış grafiğinin simetrik olması durumunda uygulanır. Aşağıdaki şekilde ifade edilebilir.

$$z^* = \frac{\sum \mu_C(\bar{z}) \cdot \bar{z}}{\sum \mu_C(\bar{z})} \tag{4.41}$$

Burada $\sum$ cebirsel toplamayı göstermektedir.Bu yöntem Şekil 4.18'de gösterilmiştir. Her üyelik fonksiyonunu en büyük üyelik değerleri alınarak aşağıdaki işlem yapılarak sonuç elde edilir.

$$z^* = \frac{a \cdot 0.5 + b \cdot 0.9}{0.5 + 0.9} \tag{4.42}$$

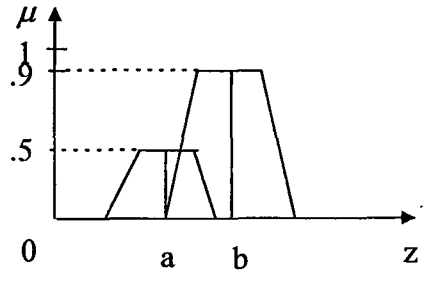


Şekil 4.18 Ağırlık ortalaması yöntemi

4.6.4 Maksimum üyelikleri ortalama yöntemi

Bu yöntem ilk yönteme oldukça yakındır. Fakat bu yöntemin uygulanacağı grafiklerde birden fazla maksimum nokta vardır. Bu yöntemde maksimum noktaların evrensel kümedeki karşılıklarının aritmetik ortalaması alınır. Aşağıdaki şekilde ifade edilebilir.

$$z^* = \frac{a+b}{2} \tag{4.43}$$

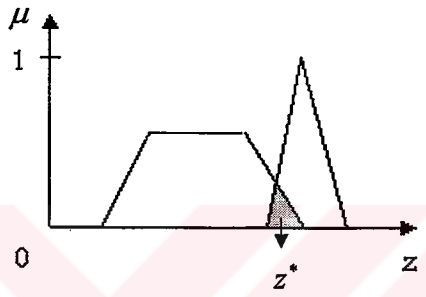


Şekil 4.19 Maksimum üyelikleri ortalama yöntemi

4.6.5 Toplamların merkezini bulma yöntemi

Kullanılan pek çok netleştirme yönteminden daha hızlı bir yöntemdir. Şekil 4.20'de gösterildiği gibi bulanık çıkış kümesi birden fazla bulanık kümenin kesişim bölgesinin ağırlık merkezi şeklindedir. Aşağıdaki eşitlik ile hesaplanır.

$$z^* = \frac{\int_z z \sum_{k=1}^{n} \mu_{C_k}(z) dz}{\int_z \sum_{k=1}^{n} \mu_{C_k}(z) dz} \tag{4.44}$$



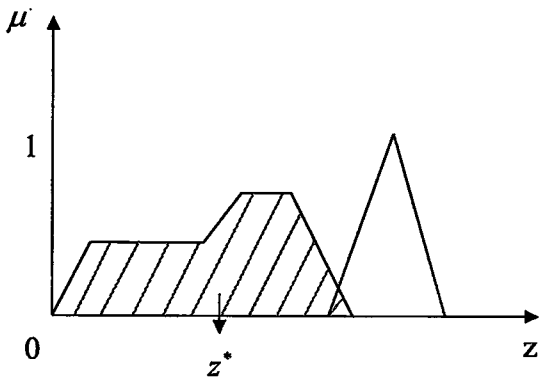
Şekil 4.20 Toplamların merkezini bulma yöntemi

4.6.6 En büyük alan merkezini bulma yöntemi

Eğer çıkışta en az iki dışbükey bölge varsa en büyük alana sahip alanın ağırlık merkezi, merkezi yöntem kullanılarak hesaplanır. Cebirsel olarak aşağıdaki gibi hesaplanır (Özgüven,1996).

$$z^* = \frac{\int \mu_{C_m}(z) z dz}{\int \mu_{C_m}(z)} \tag{4.45}$$

Burada C_m, C_k'daki en büyük dışbükey alanı temsil eder.



Şekil 4.21 En büyük alan merkezini bulma yöntemi

4.6.7 İlk veya son maksimum değeri bulma yöntemi

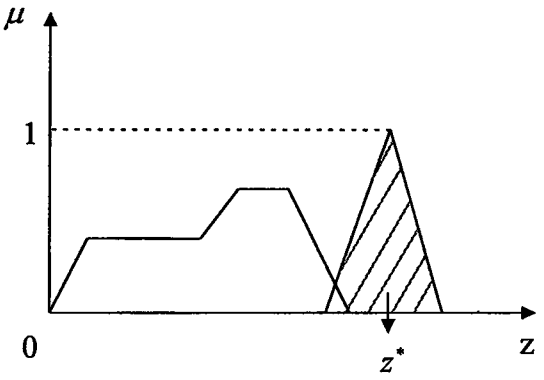
Bu yöntemde çeşitli üyelik fonksiyonlarının bileşimi olan C_k'nın evrensel çıkış kümesinde üyelik değeri en büyük olan ilk veya son değer alınır.Aşağıdaki şekilde ifade edilir. İlk maksimum değeri alma yöntemi;

$$z^* = \inf_{z \in Z} \left\{ z \in Z \middle| \mu_{C_k}(z) = hgt(C_k) \right\} \tag{4.46}$$

şeklinde olur. Son maksimum değeri alma ise;

$$z^* = \underset{z \in Z}{sub} \left\{ z \in Z \middle| \mu_{C_k}(z) = hgt(C_k) \right\} \tag{4.47}$$

şeklinde gösterilir. Burada inf evrensel kümede üyelik değeri en büyük olan ilk değeri ve sub üyelik değeri en büyük olan son değeri göstermektedir (Lee, 1990).



Şekil 4.22 İlk veya son maksimum değeri bulma yöntemi

5. PIC MİKROKONTROLÖRÜ

5.1 PIC Mikrokontrolörü

Bu çalışmada kullanılan mikrokontrolör, PI kontrol ve bulanık kontrol işlemlerini gerçekleştirmek amacıyla kullanılmıştır. PI kontrolör ve bulanık kontrolörler için assembler dilinde programlar gerekli olan kontrol işlemini gerçekleştirmek için üç ayrı mikrokontrolöre bilgisayar üzerinden yüklenmiştir. Yapılan uygulamada bu mikrokontrolörlerin seçilmesinin sebebi; gerekli olan birçok çevre birimlerinin bu mikrokontrolörlerin yapısında bulunması, kolay programlanması ve azaltılmış komut seti (RISC) yapısında üretilmiş olmasıdır. RISC yapı sayesinde, komut setindeki komut sayısı azalıp, komutların işlevlerinin basitleşmesi tasarımcı için bir dezavantaj oluştururken, çevrim zamanından büyük ölçüde kazanç elde edilmektedir. Böylece yazmış olduğumuz algoritmalar 200ns gibi mevcut mikrokontrolörler için yüksek bir çevrim hızında işlem görebilmektedir. Çalışmamızda Mikrochip firmasının **16C71** mikrokontrolörü **bulanık kontrol** algoritmalarını yürütürken, **16C65** mikrokontrolörü de **PI kontrolör** algoritmasını yürütmektedir. **16C65** mikrokontrolörü, motordan almış olduğu hız bilgisini RS232 üzerinden seri haberleşme protokolü ile bilgisayara iletirken, aynı zamanda motora uygulanacak gerilimin ortalama değerini değiştirmek için gerekli PWM darbelerini IGBT sürme devresine uygulamaktadır.

5.1.1 PIC mikrokontrolörünün genel yapısı

16C65 düşük maliyetli, yüksek performanslı, CMOS, tamamen statik sekiz bitlik bir mikrokontrolördür. 16C65 mikrokontrolörlerü geliştirilmiş RISC (Reduced Instructions Set Complex) yapısını kullanmaktadır. PIC16C65 sekiz seviyeli yığını, içerden ve dışardan çok katlı kesme kaynakları ile geliştirilmiş bir yapıya sahiptir. Veri yolu ile komut yolu ayrı olan Harvard mimarisi; ondört bit genişliğinde komut ve sekiz bit genişliğinde veri yoluna izin verir. İki kademeli komut yolu, dallanma komutları dışındaki bütün komutların tek çevrim zamanında işlenmesini sağlar. Dallanma komutları iki çevrim zamanında işlem görmektedir. 16C65 toplam otuzbeş komuttan meydana gelen azaltılmış komut kümesine sahiptir. Bütün bunlara ek olarak, yüksek performans elde edebilmek için mimari bir takım yenilikler yapılarak büyük sayıda kaydedici kümeleri kullanılmıştır. PIC16C65 mikrokontrolörü 2:1 oranında kod sıkıştırma ve 4:1 oranında hızının arttırılması ile kendi sınıfındaki sekiz bitlik diğer mikrokontrolörlerden daha gelişmiş bir mimariye sahiptir. Çalışmamızda kullanmış

olduğumuz 16C65, 192 byte rastgele erişimli belleğe (RAM) ve 33 adet giriş ve çıkış bacağına sahiptir. Ek olarak bir çok çevresel birimleri bulunmaktadır. Bu birimler; üç sayıcı-zamanlayıcı modulü, darbe genişlik modülasyonu için PWM modülü ve iki değişik seri haberleşme portudur. Senkron seri haberleşme portu üç yollu seri dış iletişim portu (SPI) veya iki yollu bağlantı ile entegre devrelerle (EEPROM, PROM, RAM,...) haberleşme birimi olarak ilgili kaydedici bloğundan biçimlendirilir. Seri haberleşme birimi (SCI) isteğe bağlı olarak senkron veya asenkron (USART) haberleşme modunu kullanabilir. Aynı zamanda dış birimleri gerektiğinde adresleyebilmek için sekiz bitlik paralel yardımcı bir portta 16C65'in içinde yer almaktadır.

PIC16C65 dışardan bağlanan elemanları azaltacak yönde bir takım özelliklere sahiptir. Böylece maliyet azalmakta, sistemin güvenilirliği artmakta ve tüketilen güç miktarı düşürülmektedir. Dört çeşit osilatör seçeneği bulunmaktadır. Bunlar düşük maliyetli ve tek bacak kullanan RC osilatör, güç tüketimini minimize eden LP osilatör, standart kristal osilatör (XT), yüksek frekanslara çıkmaya izin veren yüksek hızlı kristal HS osilatör olarak, tasarıma göre geniş bir tercih alanı yaratır. Uyku (kapanma) modu özelliği ise güç tasarrufu gibi bir seçenek sunar. Kullanıcı, çipi uyku modundan iç ve dış bir çok kesme ile uyandırabileceği gibi aynı zamanda reset modu da bu işlevi görmektedir.

PIC16C71 36 byte rastgele erişimli belleğe (RAM) ve 13 adet giriş ve çıkış birimine sahiptir. Ek olarak bir adet zamanlayıcı-sayıcı mevcuttur. Bununla beraber yüksek hızlı, dört kanallı ve sekiz bit çözünürlükte bir analog-dijital dönüştürücüyü bünyesinde barındırmaktadır. Güvenilirliği yüksek tutmak amacıyla, herhangi bir program kilitlenmesine karşın çipin üzerinde olan RC osilatör vasıtasıyla koruma sağlanmıştır.

16C71'de 16C65'in aksine 4 kanallı 8 bit bir analog-dijital dönüştürücü bulunmasına rağmen program hafızası 16C65'in dörtte biri büyüklüğündedir. Ayrıca 16C71'de biri 4, diğeri 8 bitlik iki port bulunurken 16C65'te üç adet 8 bitlik, 4bitlik 1 adet ve 3 bitlik 1 adet olmak üzere geniş bir giriş çıkış birimi bulunmaktadır. Aynı zamanda 16C65'te PWM ve seri haberleşme portları bulunmaktadır.

Program geliştirmek için ultraviyole silinebilir modeli kullanılırken, program randımanlı olarak çalıştığında düşük maliyetli ve bir-kez programlanabilen (OTP) modeli kullanılabilir.

Düşük maliyet, düşük güç tüketimi, yüksek performans, kullanım kolaylığı giriş ve çıkıştaki esneklik, daha önce mikrokontrolör kullanılması düşünülmeyen yerlerde bile kullanılabilmesine imkan tanımıştır.

5.1.2 PIC mikrokontrolörünün mimari yapısı

PIC16Cxx ailesinin yüksek performansı, genellikle RISC mikroişlemcilerde bulunan bir takım mimari özelliklerden kaynaklanmaktadır. PIC16Cxx Harvard mimarisi ile tasarlanmıştır. Bu mimarinin genel özelliği; program ile verilere ulaşmak için ayrı hafıza ve veri yolları kullanılmasıdır. Geleneksel Von Neumann yapısının kullanılmaması band genişliğini arttırmıştır. Bu yapıda veri ve program aynı hafızadan aynı veri yolu ile alınır. Bu da çevrim zamanında gecikmeye yol açar. Program ve veri yollarının ayrılması, komutların sekiz bitlik veriden farklı boyutlarda olabilmesine izin verir. Komutların opkodlarının ondört bit genişliğinde olması, bütün komutların tek kelimeden oluşmasına olanak verir. Ondört bit genişliğinde olan program hafıza ulaşım yolu, ondört bitlik komutun tek bir çevrimde işlem görmesini sağlar. İki kademeli iş hattında işin hatta girmesi ile bir önceki işin sonuç vermesi aynı anda olur. Sonuç olarak, bütün komutlar tek bir çevrimde seçtiğimiz 20mhz'lik kristal osilatör frekansı için 200ns'de işlem görür. Şekil 5.1(Mikrochip, 1995)'de 16C65'in mimari yapısı gösterilmiştir.

PIC16C65 4kx14 bitlik, PIC16C71 ise 1kx14 bitlik bir program hafızasını adresleyebilir. Bütün program hafızası çipin içindedir. PIC16Cxx doğrudan veya dolaylı olarak kaydedici dosyalarını veya veri hafızasını adresleyebilir. Bütün özel fonksiyon kaydedicileri veri hafızasında işaretlenmiş bir program sayıcısı içerirler. PIC16Cxx ortagonal (simetrik) komut kümesine sahiptir. Bu sayede herhangi bir işlem, istenilen adresleme modu kulanılarak istenilen kaydedici üzerinde gerçekleştirilebilir. Doğal simetrinin olması ile özel optimal durumların olmaması PIC16Cxx ile programlamayı oldukça basit ve verimli kılmaktadır.

PIC16Cxx mikrokontrolörü sekiz bitlik aritmetik lojık birimi ile çalışan kaydedicileri içerir. Aritmetik lojik birim, ALU olarak adlandırılır. Bu birim çalışılan kaydedicideki data ile bir kaydedici arasındaki aritmetik ve Bool fonksiyonlarını işleme sokar. ALU sekiz bit genişliğindeki verileri toplayabilme, çıkarabilme, kaydırabilme ve mantıksal işlemleri yapabilme kapasitesine sahiptir. İşleme giren iki verinin olduğu komutlarda, verilerden biri

çalışan kaydedicide (W kaydedicisi) iken, diğer veri ya dosya kaydedicisindedir ya da sabit bir sayıdır. Tek veriye sahip komutlarda ise, veri W kaydedicisindedir.

W kaydedicisi ALU işlemlerinde kullanılan ve sekiz bit çalışan bir kaydedici olup, adreslenme özelliği yoktur. Komutun sonucunun elde edilmesine bağlı olarak, ALU *status kaydedicisinde* bulunan Carry (C), Digit Carry (DC) ve Zero (Z) bitlerinin değerini değiştirir.

5.1.3 Giriş ve çıkış portları

PIC16C65'in port-a'dan port-e'ye kadar beş, PIC16C71'in ise port-a ve port-b olmak üzere iki portu bulunmaktadır. Bu portlardaki bacaklar, mikrokontrolörün özel kaydedicileri ile beraber kullanılabilinir. Port-a 16C65'te altı bit, 16C71'de ise beş bit genişliğinde gönderilen bilgileri tutabilme özelliğine sahip bir porttur. 16C71'de a portu dört kanallı bir analog-dijital dönüştürücü içermektedir. RA4 bacağı giriş olarak koşullandırıldığında, schmitt tetikleyici bir giriş, çıkış olarak koşullandırıldığında ise açık kollektörlü bir çıkış olabilmektedir. Port-b hem 16C65'te hem de 16C71'de sekiz bit genişliğinde gönderilen bilgileri tutabilme özelliğine sahip bir porttur. Port-b aynı zamanda tuştakımlarının kolay sürülebilmesi için içerden zayıf pull-up devresine sahiptir. İstenildiğinde pull-up devre dışı bırakılabilinir. Bununla beraber port-b'de bazı bacaklar dış kesmeyi sezebilme özelliğine sahiptir. Port-c sekiz bit genişliğinde gönderilen bilgileri tutabilme özelliğine sahip bir porttur. Bununla beraber bir takım özel kaydediciler bu portu kullanabilmektedir. Bu port; zamanlayıcı girişi olarak, PWM çıkışı olarak, senkron ve asenkron haberleşme hattı olarak şartlanabilinir. Port-d sekiz bitlik, port-e ise üç bitlik normal portlardır. Portların bütün bacakları ayrı ayrı giriş veya çıkış olarak koşullanabilir. Bütün portlarda çıkıştan çekilen akım 25mA ile sınırlanmıştır. Böylece kısadevre durumlarında mikrokontrolörün zarar görmesi engellenmiştir.

5.1.4 Özel modlar

PIC mikrokontrolörünü diğer mikrokontrolörlerden ayıran en büyük özellik, bir çok işlevi yerine getirebilen özel kaydedicilerinin olmasıdır. Çalışmamızda da kullanmış olduğumuz özel kaydediciler aşağıda verilmiştir.

- PWM modu
- Seri haberleşme modu
- Analog-dijital dönüştürücü modu

5.1.4.1 PWM modu

Bu özellik sadece 16C65 mikrokontrolöründe bulunmaktadır. Darbe genişlik modülasyonu modunda c portunun iki nolu bacağı on bit çözünürlüğe kadar PWM çıkışı verebilmektedir. Bu bacağın mutlaka çıkış bacağı olarak koşullandırılması gerekmektedir. Bu bacak aynı zamanda data aktarmada da kullanılabilinir. PWM modunda kullanıcı sekiz bitlik darbenin peryoda oranı CCPR1 kaydedicisinin alt byte'ına yazılır. Bu kaydedici CCPR1L olarak adlandırılır. Yüksek byte CCPR1H on bit çözünürlük istendiğinde alt byte'la beraber kullanılır. Darbe genişlik modülasyonunun peryodu, çip içerisinde bulunan ikinci zamanlayıcının kaydedicisine (PR2) istenen değerin yazılması ile elde edilir.

$$\text{PWM peryodu} = \{(PR2)+1\} \times 4\, T_{osc} \times (TMR2PRV) \tag{5.1}$$

$$\text{PWM darbe süresi} = (DC1) \times T_{osc} \times (TMR2PRV) \tag{5.2}$$

Burada T_{osc} mikrokontrolörün osilatör peryodudur. TMR2PRV ise ikinci zamanlayıcının ön skala değeridir. Ön skala değeri seçilen değere göre zamanlayıcıya gelen osilatör frekansını bölerek yapılacak sayma veya zamanlama işinin süresini uzatır. DC1 darbe genişlik modülasyonunda darbe zamanının yazıldığı on bit genişliğinde bir kaydedicidir (Mikrochip Embedded, 1995).

5.1.4.2 Seri haberleşme modu

Seri haberleşme arabirimi (SCI) modülü, 16C65 içerisinde bulunan iki seri haberleşme modulünden sadece birisidir. SCI, CRT terminalleri, kişisel bilgisayarlar gibi dışardan bağlanabilecek cihazlarla full duplex asenkron seri haberleşme yapabildiği gibi, aynı zamanda analog-dijital, dijital-analog dönüştürücüler ile seri EEPROM'larla half duplex senkron seri haberleşme yapabilecek şekilde tasarlanmıştır. SCI modülü aşağıdaki modlarda şartlanabilir.

- Asenkron (full dupleks)
- Senkron - ana (half dupleks)
- Senkron - yedek (half dupleks)

Bu modül içerisinde bir baud-rate jeneratörü (BRG) bulunmaktadır. BRG hem asenkron hem de senkron modları desteklemektedir. Baud-rate jeneratörü sekiz bit çalışmaktadır. Bununla ilgili olan ve gerekli ayarlamaların yapıldığı kaydedici SPBRG kaydedicisidir. Bu kaydedici

serbest sayan sekiz bitlik zamanlayıcının peryodunu kontrol eder. İstenilen baud-rate verilen çalışma frekansında aşağıdaki eşitlik ile hesaplanır.

$$\text{İstenen baud-rate} = f_{osc} / (64(X+1)) \quad (5.3)$$

Yüksek baud-rate'leri kullanmak düşük değerde olanları kullanmaktan daha avantajlıdır. Böylece bazı durumlarda istenen ile hesaplanan baud-rate arasındaki hata küçülmüş olur.

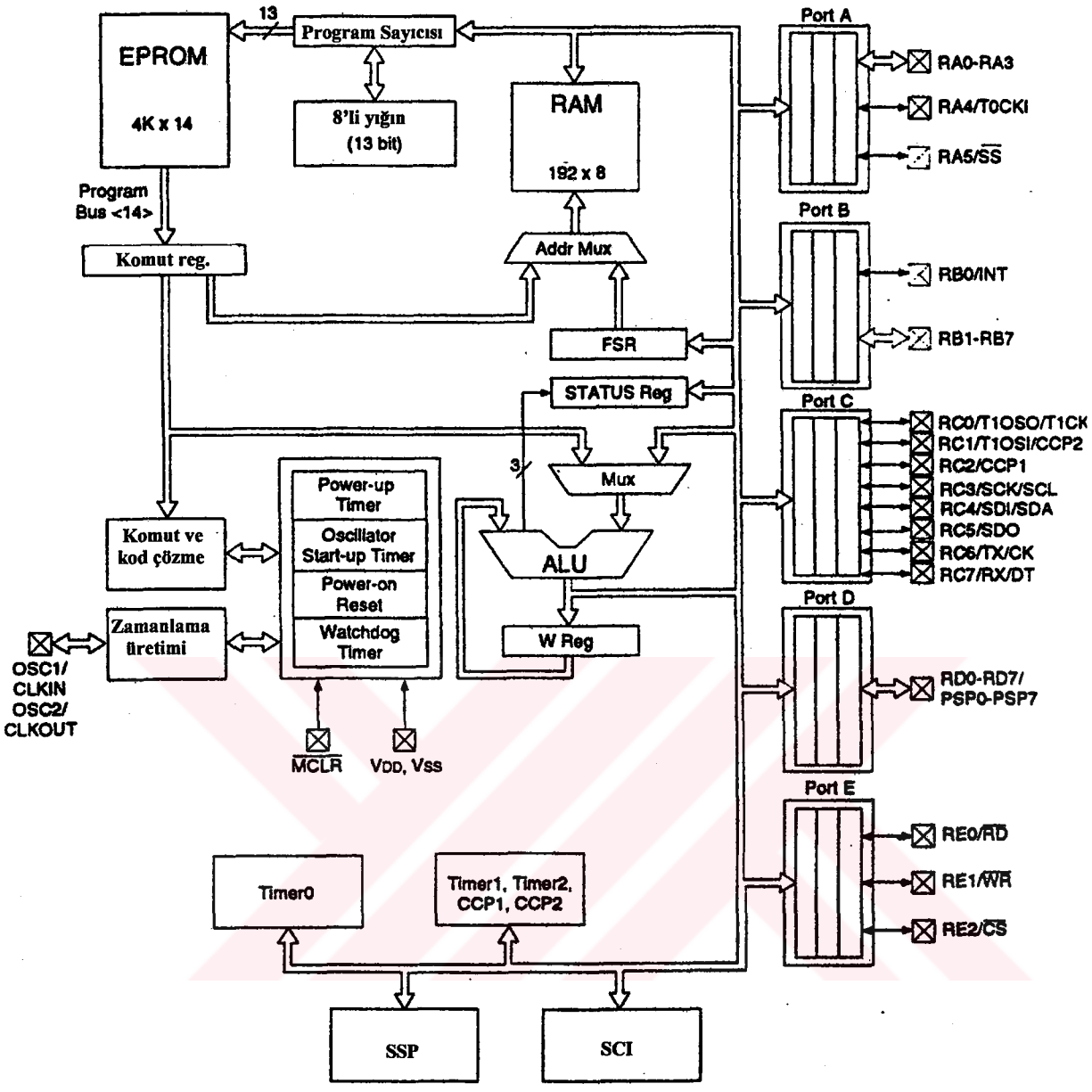
5.1.4.3 Analog-dijital dönüştürücü modu

Bu özellik sadece 16C71 mikrokontrolöründe bulunmaktadır. Analog-dijital (A/D) dönüştürücü, analog giriş sinyalinin sekiz bitlik dijital veriye çevrilmesini sağlar. A/D modül 16C71 mikrokontrolöründe bu çevrimi gerçekleştirebilecek dört analog-dijital dönüştürme kanalı bulunmaktadır. Analog referans gerilimi, çipin içinden program ile besleme gerilimi seçilebileceği gibi, uygun şekilde şartlandırılırsa a portunun üç nolu bacağına uygulanan gerilim referans gerilim olacaktır. Mikrokontrolör uyku modunda iken bile A/D dönüştürücü çalışmaya devam eder. A/D modülü üç ayrı kaydediciye sahiptir. Bu kaydediciler;

- A/D sonuç kaydedicisi (ADRES)
- A/D kontrol kaydedicisi (ADCON0)
- A/D kontrol kaydedicisi (ADCON1)

olarak sayılablir. ADCON0 kaydedicisi A/D modülün işlemlerini kontrol eder. Bu işlemler, çevrim işlemini başlatma, bitirme, kanal seçimi ve örnekleme zamanının belirlenmesi gibi işlemlerdir. ADCON1 kaydedicisi ise port üzerindeki bacakların fonksiyonlarını belirler. ADRES kaydedicisi ise A/D çevirme işinin sonucunu saklayan kaydedicidir. Analog-dijital çevirme işleminde izlenecek adımlar şöyle sıralanabilinir;

1. A/D modülün biçimlendirilmesi
    - Analog, dijital ve gerilim referans bacaklarını koşullandırılması (ADCON1)
    - A/D giriş kanalının seçilmesi (ADCON0)
    - A/D çevrim frekansının belirlenmesi (ADCON0)
    - A/D çevrimi başlatma
2. Örnekleme zamanını bekleme
3. Çevrime başlama
4. A/D çevrimin tamamlanmasının beklenmesi
5. A/D sonucunun yazıldığı kaydedicinin (ADRES) okunması



Şekil 5.1 Mikrokontrolörün mimari yapısı

6. BULANIK-PI KONTROLÖR ile DC MOTORUN KALKIŞ ANINDAKİ PERFORMANSININ ARTTIRILMASI

6.1 Giriş

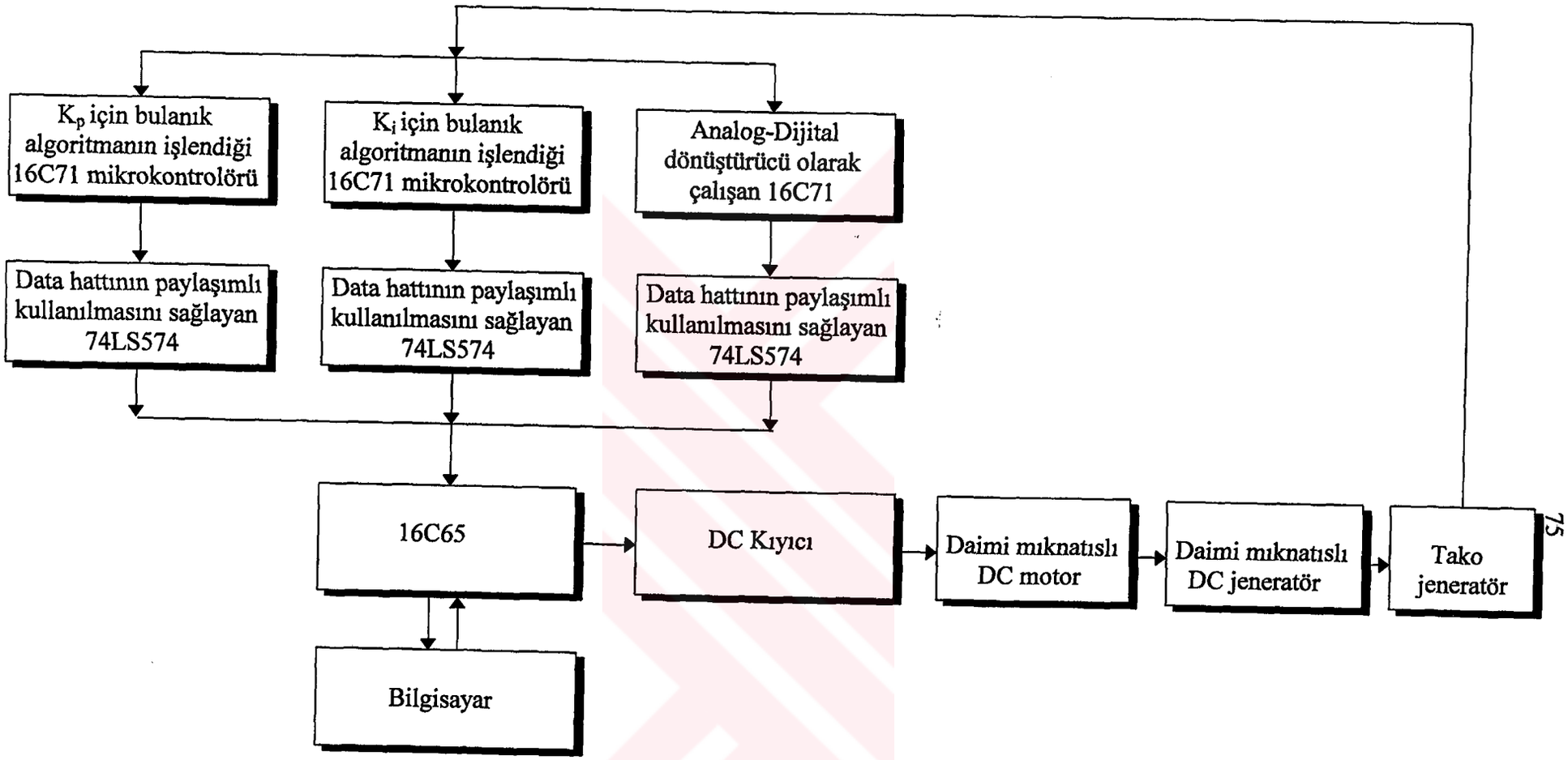
Yapılan çalışmada, sabit parametrelere sahip PI kontrolörün performansını arttırabilmek amacıyla PI kontrolörün parametreleri zamanın bir fonksiyonu haline getirilmiştir. Bu fonksiyonun sınırları, hata ve hatanın türevinin değişimini giriş değerleri kabul eden bulanık kontrol algoritması ile belirlenmiştir.

PI kontrolör ve Bulanık kontrolör RISC bir mikrokontrolör olan PIC mikrokontrolörleri ile gerçekleştirilmiştir. PI kontrolör algoritması assembler dilinde yazılmıştır. Bulanık kontrol algoritması ise Inform firmasının *fuzzytech* adlı editöründe bulanıklaştırma, kural tabanı ve netleştirme işlemlerine tabi tutulmuştur. Daha sonra bu editör elde edilen programı PIC16C71 için derlemiştir.

6.2 Sistem Hakkında Genel Tanıtım

1) Kontrol edilen sistem, daimi mıknatıslı doğru akım motorlarından oluşan üç adet motorun mekanik olarak birbirine bağlanmasından oluşmuştur. Bu mekanik bağlantı motorların millerinin aynı eksen üzerinden, mekanik bir bağlantı elemanı olan ve anti-burulma özelliğine sahip kaplinlerin kullanılması ile gerçekleştirilmiştir. Motorların kontrolü sırasında titreşim problemi ile karşılaşılmaması için, sistem aliminyum bir profilin üzerine doksan derece açı ile yerleştirilen aliminyum profillerle desteklenmiştir.

2) Motorların hızını kontrol edebilmek için DC kıyıcı kullanılmıştır. DC kıyıcının çalışma frekansı 20kHz olarak belirlenmiştir. DC kıyıcıda yarıiletken eleman olarak IGBT (İzole kapılı bipolar transistör) kullanılmıştır. IGBT'yi sürebilmek için özel bir sürme devresi tasarlanmıştır. IGBT'yi iletime sokmak için mikrokontrolörden gelen sinyal işareti 6N136 opto-kuplörü ile izole edilmiştir. Daha sonra izole edilen bu sinyal kuvvetlendirilerek IGBT'nin kapısına uygulanmıştır. IGBT'yi iletimden hızlı çıkarabilmek için, sürme sinyali kesildiğinde IGBT'nin kapısına -12V uygulanmıştır. IGBT kesimde iken motor üzerinde akan akımın kesilmesi aşırı gerilimlere sebep olacağı için motora paralel olarak hızlı ters bir diyot bağlanmıştır. Böylece motor üzerinden geçen akım ani kesilmelere maruz kalmamıştır. Bu



Şekil 6.1 Kontrol sisteminin genel blok diyagramı

diyot aynı zamanda motora uygulanan gerilim kesildiğinde motora rotor direnci üzerinden frenleme yaptırmaktadır. DC kıyıcı gücünü iki adet 120W'lık paralel bağlanmış trafodan almaktadır. Trafoların çıkışı 35A/1000V'luk bir diyot köprüsü ile doğrultulduktan sonra 6800 μF 'lık iki kapasite tarafından filtre edilmiştir.

3) Kontrol kartı üzerinde birbiriyle birlikte çalışan dört adet mikrokontrolör bulunmaktadır. Bu mikrokontrolörlerden 16C65 PI kontrolör algoritmasını yürütüp aynı zamanda bilgisayara RS232 üzerinden hız bilgisini 38400 baud-rate ve asenkron olarak göndermektedir. Bu seri haberleşme sırasında hatalı bilgi gitmemesi için bütün veriler kodlu olarak gönderilmektedir (Schweber, 1988). Aynı zamanda 16C65 DC kıyıcıya gidecek olan darbe genişlik modülasyonu sinyallerini üretir.

16C71 mikrokontrolörlerinden üç adet kullanılmıştır. Bunlardan birincisi tako-jeneratör üzerinden gelen hız bilgisini analogdan dijitale çevirdikten sonra, alçak geçiren filtre programı sayesinde gelen veriyi 74LS574 ile gerçekleştirilen veri yolu üzerinden 16C65'e ulaştırır. İkinci ve üçüncü 16C71'ler ise hata ve hatanın türevine göre bulanık işlemleri yaparak yine 74LS574'ler vasıtasıyla oran ve integral katsayılarını sekiz bit üzerinden 16C65'e ulaştırır.

Yapmış olduğumuz kart reset düğmesine basıldıktan sonra çalıştırıldığında Bulanık-PI modunda çalışırken, reset düğmesine basarken aynı anda başla düğmesine basıldığında PI kontrolör modunda çalışır. Böylece aynı kart ile her iki çalışma modu da kullanılabilinir. Kontrol ve güç kartlarının devre çizimleri Ek-4'de verilmiştir.

6.3 PI Kontrolörün Mikrokontrolör İle Tasarımı

Dijital bir kontrolör yapılacağı için öncelikle sistemin transfer fonksiyonunun bir parametresi olan örnekleme zamanını belirledik. Örnekleme zamanı servomekanizmalarda 10Hz ile 1kHz arasında seçilir. Yapılan uygulamada örnekleme zamanı değişik uygulamalar gözönünde bulundurularak 2kHz olarak belirlendi. Bu değerin çok büyük olması, sistemi analog bir sisteme yaklaştırmasına rağmen, yüksek örnekleme frekanslarında gürültü problemleriyle karşılaşılması ve çok fazla işlem yapılması, yüksek frekansta örneklemeyi gereksiz kılmaktadır. Geri besleme bilgisi, motor ve jeneratöre anti-burulma özelliğine sahip kaplinle bağlı olan bir tako-jeneratör vasıtasıyla gelmektedir. Tako jeneratör motorun maksimum devri olan 3000dev/dak 52 V'luk bir gerilim değeri üretmektedir. Gelen bilgi bir arabirim

aracılığıyla analog-dijital dönüştürücüye uygun bir gerilim seviyesine getirildikten sonra 16C71 'in içindeki analog-dijital dönüştürücü ile dijital veri haline getirilerek bir filtreden geçirilmektedir. Analog-dijital çevrim işlemi 1MHz saat frekansında yapılırken, bir alçak geçiren filtre ile yüksek frekansta oluşan gürültüler yok edilmektedir. Bu işlem için 16C71'in OTP modeli kullanılmıştır. Daha sonra bu bilgiler 16C71'in b portundan 16C65'in b portuna 74LS574 ile verilerek hız bilgisi PI kontrol işlemlerinin yapılacağı mikrokontrolöre gönderilmiş olur.

PI işlemlerini yapabilmek için istenen referans hız bilgisi seri haberleşme yoluyla bilgisayarın RS232 hattı üzerinden 16C65'e gönderilir. Aynı zamanda bu haberleşme hattı üzerinden oran parametresi olan K_p ve integral parametresi K_i'de 16C65'e seri olarak iletilir. Hata bilgisi ise sekiz bitlik referans hız bilgisinden geri besleme bilgisi çıkartılarak hesaplanmıştır. Buraya kadar olan işlemlerimiz sekiz bit olmasına rağmen, sekiz bit ile yüksek çözünürlük elde edemeyeceğimiz için bu kontrolör ile onaltı bit üzerinden işlem yapıldı. Toplama, çıkarma ve çarpma işlemleri için yazılan alt programlar ekte gösterilmiştir. Hata bilgisi yapılan çıkarma işlemi sonucu elde edildikten sonra, seri port üzerinden gelen K_p katsayısı ile çarpılarak sonuç bir hafıza biriminde saklanır. Daha sonra yine aynı hata bilgisi K_i ile çarpılarak elde edilen sonuç bir hafıza birimine yüklenir. Daha önce PI kontrolör bölümünde anlatmış olduğumuz ileri yol integrasyonunu kullanarak integral işlemini mikrokontrolör içerisinde hesaplarız. Bu iki çarpım işleminin sonucu sekiz bitlik iki sayının çarpılması ile onaltı bit olmuştur. Bu sebeple her iki çarpım sonucu toplam dört byte hafıza birimi işgal eder. Yapılan integral işlemi sonucu her çevrimde $K_i \cdot \Sigma_n$ (Σ_n n. Örneklemedeki hatayı göstermektedir) hesaplanmıştır. Elde edilen bu değer bir kaydedici içerisinde daha önce hesaplanmış olan $\sum_{k=1}^{n} K_i \cdot \Sigma_n$ ile hatanın işareti pozitif ise toplama, negatif ise çıkarma işlemine tabi tutulmuştur. Aynı zamanda her çevrimde $K_p \cdot \Sigma_n$ bu integral toplamı ile hatanın işareti pozitif ise toplama, negatif ise bu çıkarma işlemine tabi tutulmuştur. Bu işlemleri bir sıraya koyacak olursak;

$$Y_p = K_p \cdot \Sigma_0 \tag{5.4}$$

$$Y_i = \sum_{k=1}^{n} K_i \cdot \Sigma_n \tag{5.5}$$

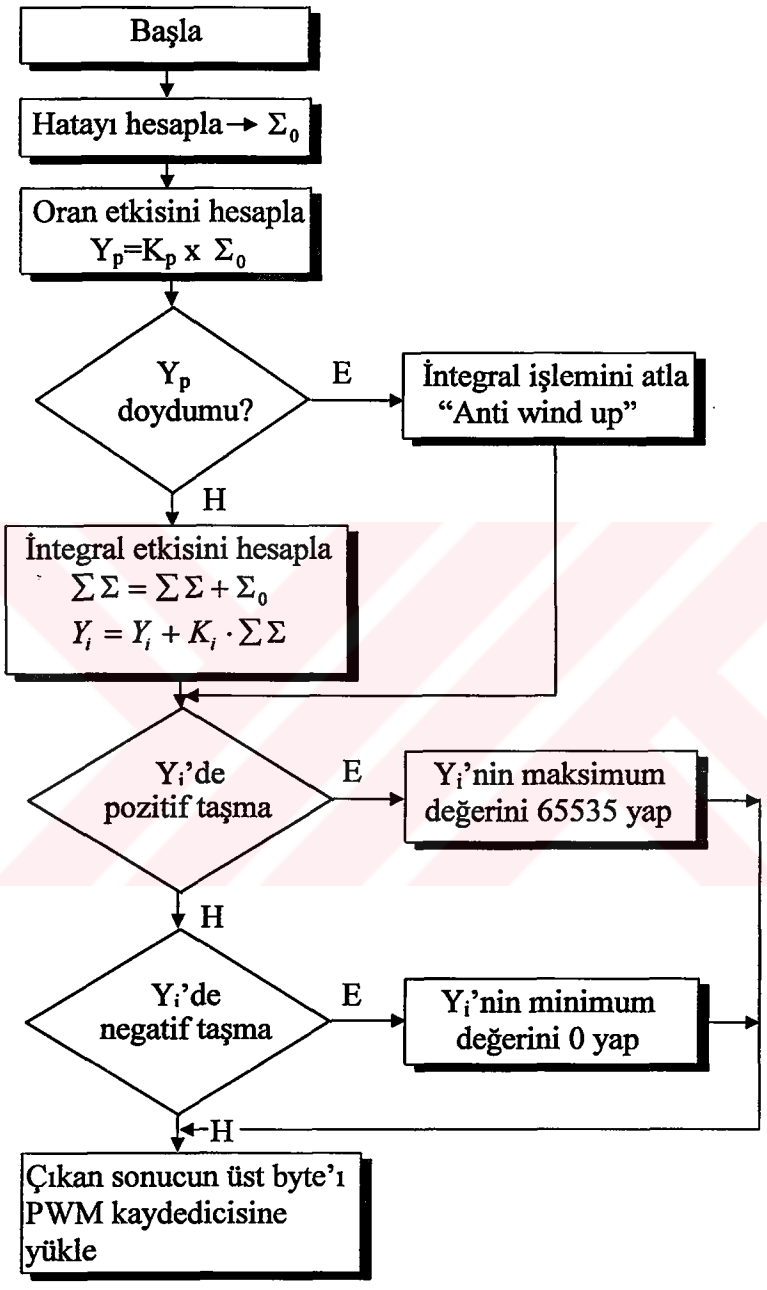
$$U = Y_p + Y_i \tag{5.6}$$

şeklinde elde edilir. Burada Y_p oran kontrolü ile elde edilen çıkış, Y_i integral kontrolü ile elde edilen çıkış, U ise her ikisinin birleştirilmesi ile elde edilen çıkışı simgelemektedir. Bu işlemlerin sonucunda elde edilen kontrol işareti onaltı bitlik bir bilgidir. Bu onaltı bit uzunluğundaki sayısal veriyi analog dünyaya aktarabilmek için daha önce tanımlamış olduğumuz özel fonksiyonlu kaydedicilerden PWM kaydedicisini kullandık. Burada elde ettiğimiz sayıyı üst sekiz bitini alarak PWM kaydedicisinde darbe oranı olarak c portunun iki nolu bacağından DC motor sürme devresine verildi. Kontrolörümüzde $K_p \cdot \Sigma_0$ değeri $2^{16} = 65535$'i geçecek olursa integral işlemi atlanır. Böylece çıkış işaretinin doymaya ulaşması engellenir. Bu işlem anti-wind up olarak adlandırılır.

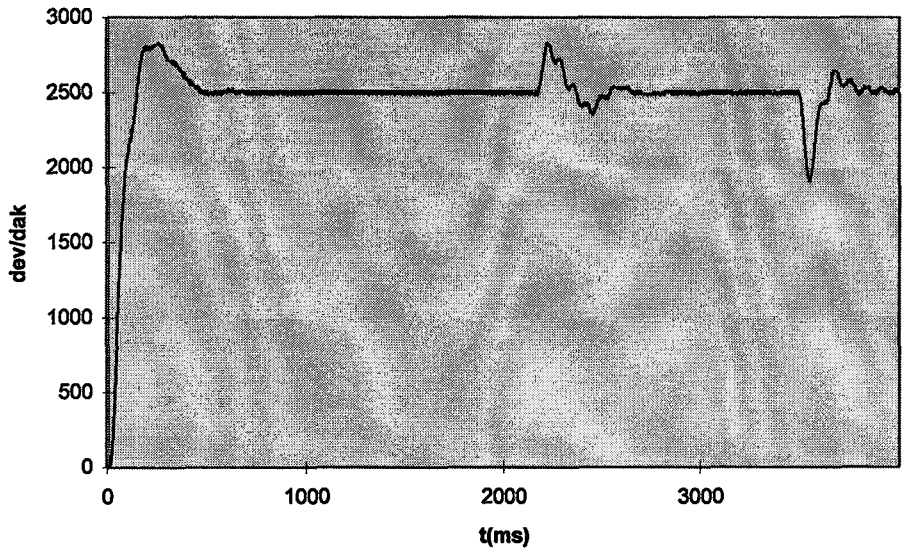
PI kontrolörde integral işlemi hata pozitif iken sürekli bir toplama işlemi yapmaktadır. Bu işlem sonucunda onaltı bitlik kaydedicinin dolması durumunda bir sonraki çevrimde toplama işlemi sonucunda hatalı bir toplama işlemi meydana gelecektir. Böyle bir olasılığı engellemek için integral işleminin sonucunun yazıldığı kaydedicide bulunan sayının değeri $2^{16} = 65535$ olunca hata pozitif ise bu sayı artık toplama işlemine sokulmayarak yapılacak bir hata engellenmiştir. Aynı işlem çıkarma işlemi içinde geçerlidir. Çıkarma işlemi sonucunda sıfırdan daha küçük bir sayı olamayacağı için çıkarma işlemi sıfırda sınırlandırılmıştır. Assembler dilinde yazılan PI kontrol programı Ek-2'de verilmiştir.

Kontrol işlemi yapılırken aynı zamanda 16C65, seri port üzerinden bilgisayara hız bilgisini göndererek hızın zamanla değişiminin ekranda görülmesini sağlamaktadır. Aynı zamanda bu programda elde edilen veriler bir text dosyasına işlenerek daha sonra bu verilerin EXCEL üzerinde incelenmesi mümkün olmuştur. Bu işlem için iki program yazılmıştır. Bu programlardan biri sadece motorun kalkış anını gözlemlerken diğeri sürekli olarak motorun hız değişimini göstermektedir. Bu programların yazılmasında Borland-C dili kullanılmıştır. Sözü edilen bu programlar Ek-1'de verilmiştir.

Yazılan PI kontrol programın akış diyagramı Şekil 6.2'de gösterilmiştir. Aynı zamanda gerçekleştirilen PI kontrolörün 52W'lık yükün devreye alınıp çıkarılması anında kontrol sisteminin davranışı Şekil 6.3'de gösterilmiştir.



Şekil 6.2 PI kontrol algoritmasının akış diyagramı



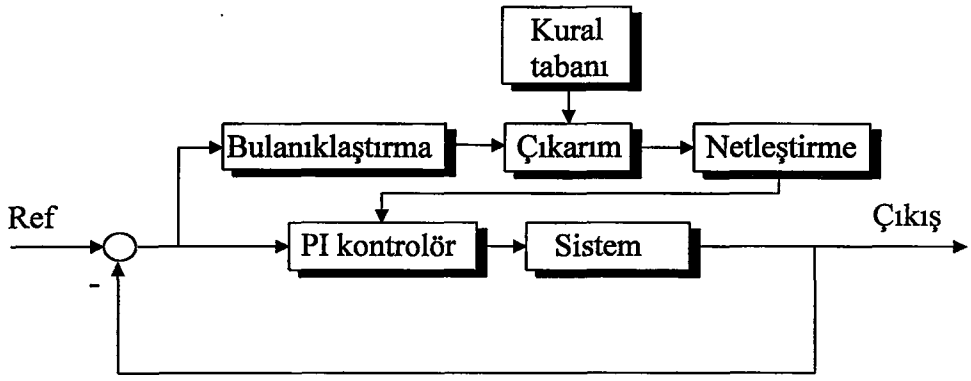
Şekil 6.3 PI kontrolörün kontrol ettiği sistemde yükün devreye alınması ve çıkarılması

6.4 PI Kontrolörün Katsayılarının Bulanık Kontrolör İle Ayarlanması

PI kontrolör tek katlı bir kontrolör yapısına sahiptir. Sabit katsayılara sahip PI kontrolörün verimini arttırabilmek için kontrol katsayıları K_p ve K_i'nin sistemin cevabına uyumlu bir şekilde değiştirilmesi gerekmektedir. Bundan dolayı tek katlı kontrol yapısından sıyrılarak, iki katlı kontrol yapısına geçilmesi gerekir. Yapılan çalışmada iki katlı bir kontrol yapısını kullanılarak doğru akım motorunun kalkış anındaki yükselme ve oturma zamanları ile aşım miktarını küçültmeyi hedefledik. Bu iki katlı kontrolörün birinci katını PI kontrolör, ikinci katını ise kontrolörün katsayılarını kontrol eden bulanık kontrolör oluşturmaktadır.

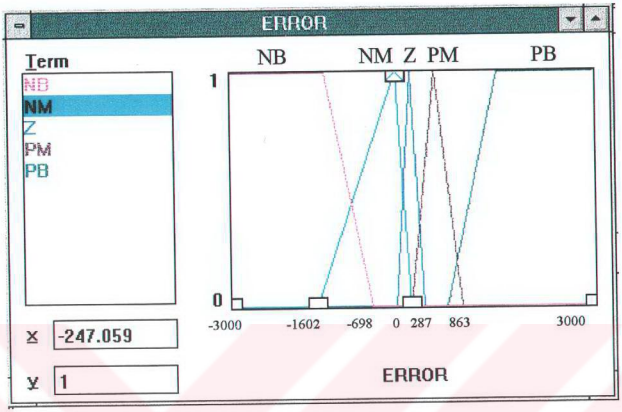
İki katlı kontrolörlerin tek katlı kontrolöre göre üstünlüğü, farklı kontrolör yapılarının farklı görevleri hedef alarak tasarlanmasıdır. Bu sayede kontrolörlerin yapıları basitleşirken, performanslarında gözle görülen bir artış meydana gelmektedir. Yapılan uygulamada kullanmış olduğumuz yapı Şekil 6.4'de gösterilmiştir.

Bulanık kontrolde giriş işareti olarak hata ve hatanın değişimi yani türevi alınmıştır. Doğru akım motorunun maksimum hızı 3000dev/dak olduğu için, hata -3000dev/dak ile 3000dev/dak arasında değişmektedir. Bu durumda bulanık işlemler için $6000/\left(2^8-1\right)=23.5$ dev/dak gibi bir çözünürlüğe ulaşılmıştır.

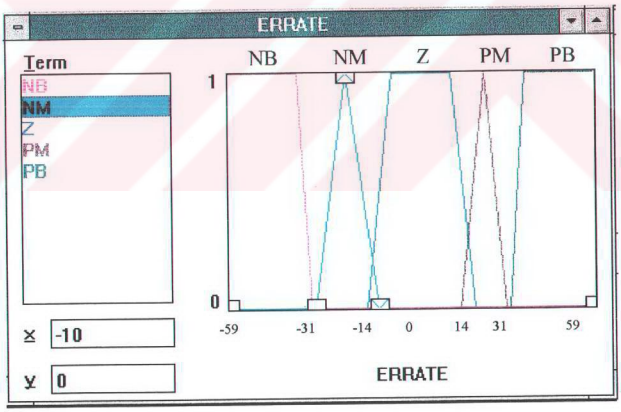


Şekil 6.4. Bulanık-PI kontrolörün blok diyagramı

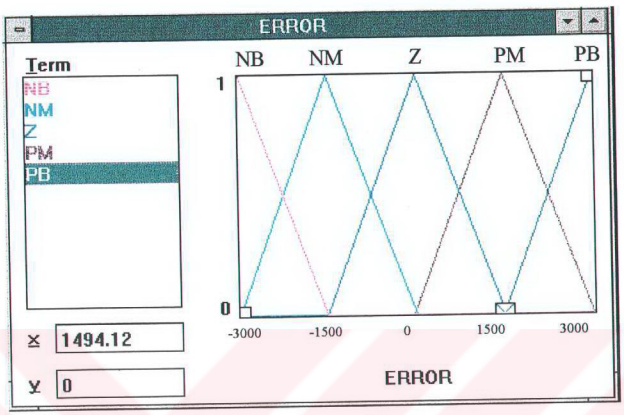
Hatanın türevi yani değişimi ise motorumuzun hız-zaman eğrisinden faydalanılarak elde edilmiştir. Burdan elde edilen verilere göre hatanın maksimum değişimi, örneklemenin 2kHz olduğu gözönüne alınacak olursa 59 dev/dak olarak bulunmuştur. Bu değerlere göre üyelik fonksiyonları bu iş için özel bir editör olan Inform firmasının geliştirmiş olduğu *fuzzy*TECH-MP'de, oran ve integral katsayıları için ayrı ayrı düzenlenmiştir. Bu çizimler sırasıyla Şekil 6.5, Şekil 6.6, Şekil 6.7 ve Şekil 6.8'de gösterilmiştir.



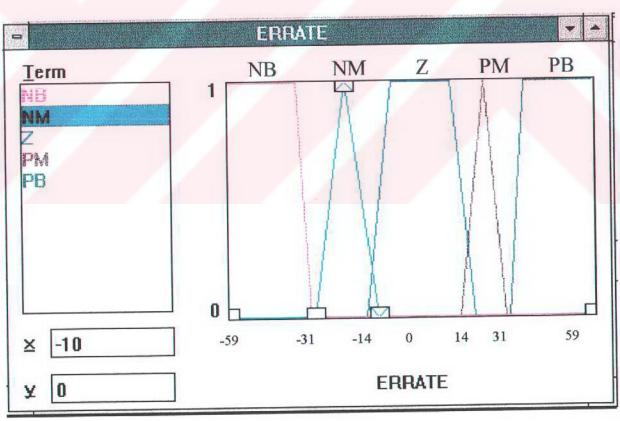
Şekil 6.5 İntegral katsayısının, hata için üyelik fonksiyonu



Şekil 6.6 İntegral katsayısının, hatanın değişimi için üyelik fonksiyonu



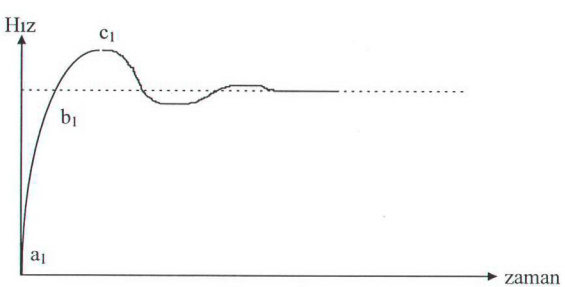
Şekil 6.7 Oran katsayısının, hata için üyelik fonksiyonu



Şekil 6.8 Oran katsayısının, hatanın değişimi için üyelik fonksiyonu

Bulanıklaştırma işleminde giriş üyelik fonksiyonları beş adet terim içermektedir. Bunlar negatif yönde büyük terimini simgeleyen NB, negatif yönde orta terimini simgeleyen NM, sıfır terimini simgeleyen Z, pozitif yönde orta terimini simgeleyen PM, pozitif yönde büyük terimini simgeleyen PB terimleridir. İntegral katsayısının çıkış büyüklükleri de beş adet terimle gösterilmiştir. Bunlar sırasıyla çok küçük, küçük, normal, orta ve büyük terimleridir. Oran katsayısı ise büyük ve küçük olmak üzere iki terime sahiptir. Yapılan bulanıklaştırma işlemi sonucunda hata ve hatanın değişimi için oran ve integral katsayılarının üyelik fonksiyonları denemeler sonucunda Şekil 6.5, Şekil 6.6, Şekil 6.7 ve Şekil 6.8'de gösterildiği üzere elde edilmiştir.

Kural tabanını elde etmek için DC motor sisteminin girişine birim basamak fonksiyonu uygulandığında sistemden elde edilen çıkış incelenmiştir. Bu çıkış üzerinde bir takım yorumlar yapılarak kural tabanı oluşturulmuştur. Şekil 6.9'da DC motor sisteminin birim basamak fonksiyonuna verdiği cevap gösterilmiştir. Başlangıçta Şekil 6.9'da verilen grafikteki a_1 noktası civarında hata maksimum değerine sahiptir. Motorun hızlı bir yükseliş zamanı ile istenen referans hıza ulaşabilmesi için büyük bir kontrol işaretine ihtiyaç duyulur. Büyük bir kontrol işareti elde etmek için bu bölgede özellikle oransal ve integral sabitlerinin büyük olması gerekir. Şekil 6.9'da verilen grafikteki b1 noktasında artık hata işareti sıfıra yakın bir değer almıştır. Bu bölgede büyük bir aşımdan kaçınmak gerekmektedir. Bu sebeble küçük bir kontrol işareti uygulanması gerekir. Şekil 6.9'da verilen grafikteki c1 noktasında ise sistem bir miktar aşım yapmıştır. Bu aşımı kısa zamanda söndürebilmek için orta değerde oran ve integral katsayıları kullanılmalıdır. Böylece sistem daha kısa zamanda referans değere oturmuş olur. Hata için beş ve hatanın değişimi için de beş dilsel terim olduğuna göre oran ve integral katsayıları için ayrı ayrı olmak üzere 25'er kural bulunmaktadır. Bu kurallar Çizelge 6.1 ve Çizelge 6.2'de integral ve oran katsayıları için verilmiştir.



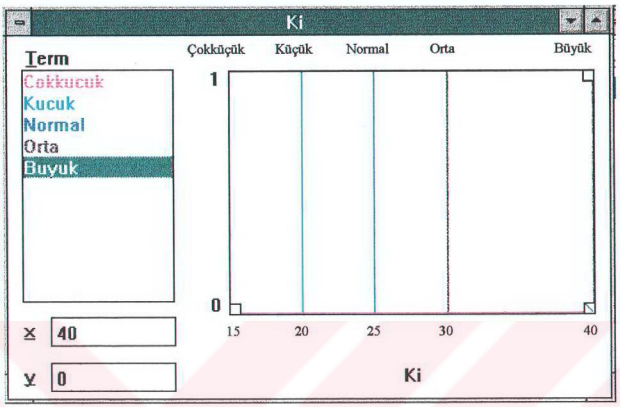
Şekil 6.9 DC motor sisteminin birim basamak fonksiyonu girişe vermiş olduğu cevap

Çizelge 6.1 K_i katsayısı için kural tabanı

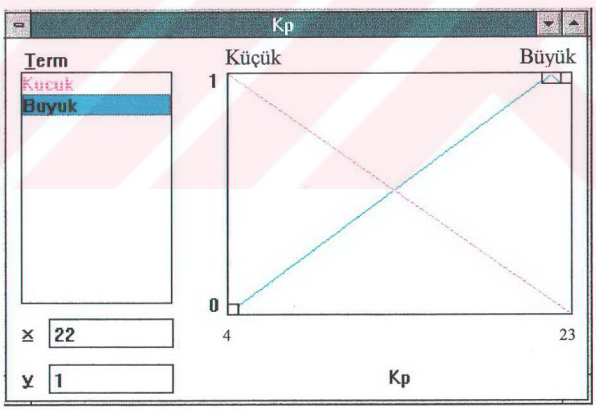
Eğer	Hatanın türevi	NB	Ve	Hata	NM	İse	Ki	Çok küçük
Eğer	Hatanın türevi	NM	Ve	Hata	NM	İse	Ki	Çok küçük
Eğer	Hatanın türevi	Z	Ve	Hata	NM	İse	Ki	Çok küçük
Eğer	Hatanın türevi	NB	Ve	Hata	Z	İse	Ki	Normal
Eğer	Hatanın türevi	NM	Ve	Hata	Z	İse	Ki	Normal
Eğer	Hatanın türevi	Z	Ve	Hata	Z	İse	Ki	Normal
Eğer	Hatanın türevi	PM	Ve	Hata	Z	İse	Ki	Normal
Eğer	Hatanın türevi	PB	Ve	Hata	Z	İse	Ki	Normal
Eğer	Hatanın türevi	PB	Ve	Hata	PM	İse	Ki	Normal
Eğer	Hatanın türevi	NB	Ve	Hata	NB	İse	Ki	Orta
Eğer	Hatanın türevi	Z	Ve	Hata	PM	İse	Ki	Orta
Eğer	Hatanın türevi	PM	Ve	Hata	PM	İse	Ki	Orta
Eğer	Hatanın türevi	NB	Ve	Hata	PM	İse	Ki	Büyük
Eğer	Hatanın türevi	NB	Ve	Hata	PB	İse	Ki	Büyük
Eğer	Hatanın türevi	NM	Ve	Hata	NB	İse	Ki	Büyük
Eğer	Hatanın türevi	NM	Ve	Hata	PM	İse	Ki	Büyük
Eğer	Hatanın türevi	NM	Ve	Hata	PB	İse	Ki	Büyük
Eğer	Hatanın türevi	Z	Ve	Hata	NB	İse	Ki	Büyük
Eğer	Hatanın türevi	Z	Ve	Hata	PB	İse	Ki	Büyük
Eğer	Hatanın türevi	PM	Ve	Hata	NB	İse	Ki	Büyük
Eğer	Hatanın türevi	PM	Ve	Hata	NM	İse	Ki	Büyük
Eğer	Hatanın türevi	PM	Ve	Hata	PB	İse	Ki	Büyük
Eğer	Hatanın türevi	PB	Ve	Hata	NB	İse	Ki	Büyük
Eğer	Hatanın türevi	PB	Ve	Hata	NM	İse	Ki	Büyük
Eğer	Hatanın türevi	PB	Ve	Hata	PB	İse	Ki	Büyük

Çizelge 6.2 K_p katsayısı için kural tabanı

Eğer	Hatanın türevi	NB	Ve	Hata	NM	İse	Kp	Büyük
Eğer	Hatanın türevi	NM	Ve	Hata	NM	İse	Kp	Büyük
Eğer	Hatanın türevi	Z	Ve	Hata	NM	İse	Kp	Büyük
Eğer	Hatanın türevi	NB	Ve	Hata	Z	İse	Kp	Küçük
Eğer	Hatanın türevi	NM	Ve	Hata	Z	İse	Kp	Küçük
Eğer	Hatanın türevi	Z	Ve	Hata	Z	İse	Kp	Küçük
Eğer	Hatanın türevi	PM	Ve	Hata	Z	İse	Kp	Küçük
Eğer	Hatanın türevi	PB	Ve	Hata	Z	İse	Kp	Küçük
Eğer	Hatanın türevi	PB	Ve	Hata	PM	İse	Kp	Büyük
Eğer	Hatanın türevi	NB	Ve	Hata	NB	İse	Kp	Büyük
Eğer	Hatanın türevi	Z	Ve	Hata	PM	İse	Kp	Büyük
Eğer	Hatanın türevi	PM	Ve	Hata	PM	İse	Kp	Büyük
Eğer	Hatanın türevi	NB	Ve	Hata	PM	İse	Kp	Büyük
Eğer	Hatanın türevi	NB	Ve	Hata	PB	İse	Kp	Büyük
Eğer	Hatanın türevi	NM	Ve	Hata	NB	İse	Kp	Büyük
Eğer	Hatanın türevi	NM	Ve	Hata	PM	İse	Kp	Büyük
Eğer	Hatanın türevi	NM	Ve	Hata	PB	İse	Kp	Büyük
Eğer	Hatanın türevi	Z	Ve	Hata	NB	İse	Kp	Büyük
Eğer	Hatanın türevi	Z	Ve	Hata	PB	İse	Kp	Büyük
Eğer	Hatanın türevi	PM	Ve	Hata	NB	İse	Kp	Büyük
Eğer	Hatanın türevi	PM	Ve	Hata	NM	İse	Kp	Büyük
Eğer	Hatanın türevi	PM	Ve	Hata	PB	İse	Kp	Büyük
Eğer	Hatanın türevi	PB	Ve	Hata	NB	İse	Kp	Büyük
Eğer	Hatanın türevi	PB	Ve	Hata	NM	İse	Kp	Büyük
Eğer	Hatanın türevi	PB	Ve	Hata	PB	İse	Kp	Büyük

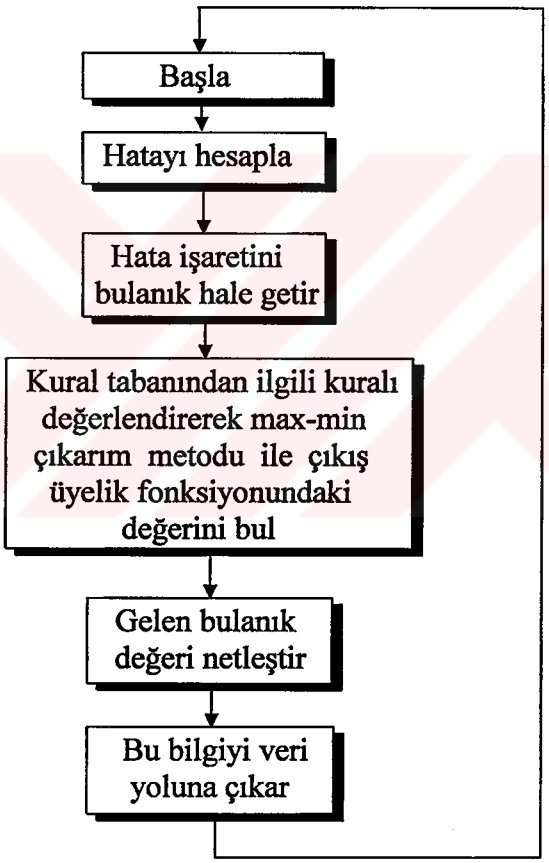


Şekil 6.10 Ki için çıkış üyelik fonksiyonu



Şekil 6.11 Kp için çıkış üyelik fonksiyonu

Çıkış üyelik fonksiyonları Şekil 6.10 ve Şekil 6.11'de oran ve integral katsayıları için verilmiştir. Bütün üyelik fonksiyonları ve kural tabanı belirlendikten sonra sıra netleştirme işlemine gelmiştir. Netleştirme işleminde K_p daha önce bulanık mantık kısmında anlatılan yöntemlerden en çok kullanılan ve mikrokontrolörde kolaylıkla gerçekleştirilebilen **ağırlık ortalaması yöntemi** kullanılmıştır. K_i için **maksimum üyelik yöntemi** kullanılmıştır. Bulanık işlemler için iki adet 16C71 mikrokontrolöründen yararlanılmıştır. Bunlardan biri K_p için bulanık işlemleri hesaplarken diğeri de K_i için bu hesapları tekrarlamaktadır. Her iki mikrokontrolörde netleştirme işleminden sonra elde ettiği kesin değerleri paralel veri yolu ile 16C65'e göndermektedir. Bulanık kontrol için yazılan assembler programı Ek-3'te verilmiştir. Programın akış diyagramı aşağıda Şekil 6.12'de verilmiştir.



Şekil 6.12 Bulanık kontrol programının akış diyagramı

6.5 Performans Kriterleri

Kontrol sistemlerinde sistemin performansı, performans kriterleri ile ifade edilir. Performans göstergesi, sistem performansının başarısının derecesini gösteren bir sayıdır.

6.5.1 Hatanın karesinin integrali (ISE) kriteri

ISE'ye göre sistem performansının kalitesi, hatanın karesinin integralinin minimize edilmesi ile mümkündür.

$$J = \int_0^\infty e^2(t)dt \tag{5.7}$$

Bu kriter ile tasarlanan sistemlerde büyük başlangıç hatası hızla azalacak şekilde bir davranış görülür. Buna göre sistemin cevabı hızlı ve osilasyonlu olmaktadır. ISE pratikte bazı sistemler için güç tüketiminin bir ölçüsüdür.

6.5.2 Mutlak hatanın zaman çarpımlı integrali (ITAE) kriteri

Bu kritere göre optimum sistem, performans göstergesini minimize eden sistemdir.

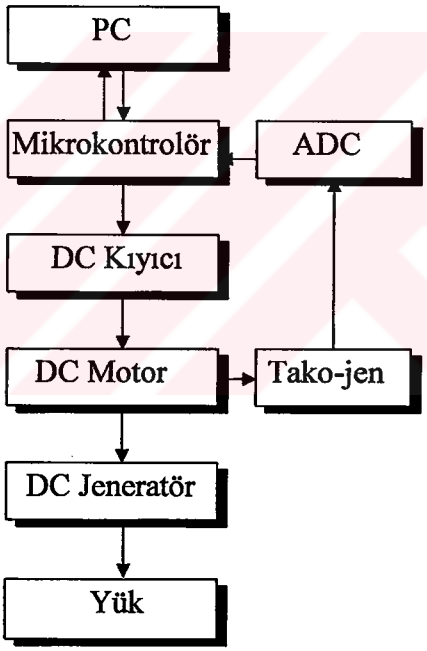
$$J = \int_0^\infty t \cdot |e(t)|dt \tag{5.8}$$

ITAE'ye göre tasarlanan sistemlerde geçici cevapta aşım miktarı küçük ve osilasyonlar bastırılmıştır (Ogata, 1970).

7. SONUÇLAR VE TARTIŞMA

7.1. Sonuçlar

Bulanık-PI kontrolör ilk defa Zhao, Tomizaku ve Isaka'nın yapmış olduğu teorik çalışmaların IEEE'de makale haline getirilmesiyle gündeme gelmiştir (Zhao,Tomizuka ve Isaka, 1993). Bu çalışma temel alınarak, bulanık bir kontrolör ile PI kontrolör birleştirilerek sabit mıknatıslı bir DC motorun hızı kontrol edilmiştir. Yapılan bu çalışmada, iki katlı kontrol yapısına sahip bir kontrolör meydana getirilmiştir. Bu kontrolörün birinci katını endüstriyel ortamlarda kullanımına son derece sık rastlanan PI kontrolör oluştururken, ikinci katı ise bulanık kontrolör meydana getirmektedir. Bu kontrol yapısının ikinci katını oluşturan bulanık kontrolör, PI kontrolörün parametrelerini, hata ve hatanın birinci dereceden türevi gözönünde bulundurarak kural tabanındaki uygun kuralın belirlenmesi ile ayarlamaktadır.



Şekil 7.1 Sistemin genel çalışmasını gösteren blok diyagram

Kontrol ettiğimiz sistem anti-burulma özelliğine sahip kaplinlerle mekanik olarak birbirine bağlanmış olan motor-jeneratör ve tako-jeneratör grubundan meydana gelmektedir. Her üç elemanda kalıcı mıknatıslı doğru akım motorudur. Etiket değerleri birbirinin aynıdır. Denemelerde kullanılan referans değer 2500dev/dak'dır. Yukardaki diyagramda sistemin genel blok diyagramı verilmiştir. Mikrokontrolör başlangıç anında referans hız bilgisi, oran ve integral katsayılarını bilgisayardan almaktadır. Çalışma başladıktan sonra ise hız bilgisini

bilgisayara göndererek, bilgisayarda Borland-C dilinde hazırlanan bir program vasıtasıyla hız-zaman grafiği, grafik ekranda çizilmektedir. Aynı zamanda EXCEL'de yapılacak hesaplama ve çizim işlemleri için hız bilgisi bir text dosyasına yazılmaktadır. Bu haberleşme işlemi seri port üzerinden 38400 baud-rate ile gerçekleştirilmektedir. Normal derleyiciler haberleşme hızını 19200 baud-rate kadar desteklediğinden, 38400 baud-rate hızına çıkabilmek için 8086 işlemcinin assembler komutları ile haberleşme çipine direk müdahele edilmiştir. Tako-jeneratörden gelen bilgi bir arabirim ile 3000dev/dak'da maksimum skala değeri olan 52 volt, analog-dijital dönüştürücünün tam skala değeri olan 5 volt gerilime indirgenmiştir. Daha sonra mikrokontrolörün içinde bulunan A/D dönüştürücü ile 8 bit üzerinden sayısal bilgi haline getirilen hız bilgisi $3000/\left(2^8-1\right)=11$ dev/dak'lık bir çözünürlüğe sahip olmuştur.

Ana işlemci 16C65 PI kontrolör algoritmasını yürütmektedir. PI kontrolörün parametreleri ise iki ayrı 16C71 kontrolörü ile sürekli olarak hata ve türevinin değerine değerlendirilip, bulanık işlemler ile hesaplanarak veri hattı üzerinden 16C65 'e iletilmektedir. 16C65 bu parametreleri 2kHz'lik örnekleme frekansında PI kontrol algoritması içinde değerlendirerek çıkış değerini elde etmektedir. Dijital bir değer olan çıkış, darbe genişlik modülasyonu olarak (PWM) 20kHz anahtarlama frekansında DC kıyıcının girişine uygulanmaktadır. DC kıyıcı girişindeki 52 voltluk DC gerilimi %0 ile %100 arasında uygun değerde kıyarak %0.4'lük bir çözünürlük ile motora uygulamaktadır.

Bulanık işlemler; bulanıklaştırma, kural tabanının elde edilmesi ve netleştirme bilgisayar üzerinde *fuzzy*TECH adlı editör ile gerçekleştirildi. Bu editör üzerinde oluşturulan bulanık sistem, PİC 16C71 için assembler koda dönüştürüldü. Editörün yapamadığı giriş değerlerinin adaptasyonu ile motor hızının analogtan dijitale çevrilmesi ve 16C65 ile paralel veri hattı üzerinden haberleşme programları ise ayrıca yazılarak bu programa ilave edildi.

Yapılan bu çalışmada; Bulanık-PI kontrolör ile PI kontrolörün kontrol ettiği motor ve jeneratörden oluşan sistemin, değişik yük grupları altında istenilen referans değere ulaşması esnasındaki performans kriterleri karşılaştırılmıştır. Her iki kontrolör karşılaştırılırken gözönüne alınan parametreler; sistemin oturma zamanı (t_s), yükselme zamanı (t_r) ve sistemin yapmış olduğu aşım miktarıdır. Bu karşılaştırma Çizelge 7.1'de ayrıntılı olarak gösterilmiştir. Aynı zamanda performans kriteri olarak kullanılan, hatanın değişimiyle orantılı iki kriter olan ISE ve ITAE, bu iki kontrolör için karşılaştırmalı olarak değişik yük değerleri için ayrı ayrı hesaplanmıştır. Bu hesaplamalar mikrokontrolörden bilgisayara aktarılan hız bilgilerinin

EXCEL ortamında değerlendirilmesi ile elde edilmiştir. Bu hesaplama sonunda yapılan Bulanık-PI kontrolörün, PI kontrolöre göre gerek yükselme ve oturma zamanları açısından gerekse aşım bakımından daha başarılı olduğu tablodan görülmektedir. Kontrolörleri değişik yükler altında deneyerek, Bulanık-PI kontrolörün performansının, PI kontrolörden daha iyi olduğu görülmüştür. Sistemi yükleyebilmek için, kontrol etmiş olduğumuz doğru akım motoruna mekanik olarak kaplinle bağlı jeneratör vasıtasıyla bağlamış olduğumuz 39W, 45W, 52W'lık rezistif yükler kullanıldı. Bu yükler için elde edilen hız-zaman grafikleri karşılaştırmalı olarak Şekil 7.2, Şekil 7.3 ve Şekil 7.4'de gösterilmiştir.

Bugüne kadar bulanık kontrolün kullanıldığı uygulamalarda maliyeti yüksek ve kullanımı daha zor olan bulanık işlemciler veya hesaplama süresi mikroişlemcilere nazaran oldukça yavaş paket programlar kullanılmaktaydı. Bu çalışmada üyelik fonksiyonları, kural tabanı ve netleştirme işlemlerinin mikrokontrolörün içine assembler dilinde yazılarak yerleştirilmesi ile Bulanık-PI kontrolörün uygulamasının gerçekleştirilmesi yapılan çalışmanın **orijinal** noktalarını oluşturmaktadır. Bulanık kontrol için kullandığımız mikrokontrolörde 20MHz'lik kristal kullanılması ile 200ns olan çevrim süresi sayesinde, bulanık kontrol algoritması toplam $450\,\mu s$ süre tutmuştur. Böylece maliyet açısından oldukça ucuz ve hızlı bir yöntem geliştirilmiştir. Elde edilen sistem, sadece üyelik fonksiyonlarının değiştirilmesi ile diğer sistemlere kolaylıkla uygulanabilmektedir.

Uygulamalı olarak hazırlanan Bulanık-PI kontrolörde kural tabanının oluşturulması esnasında, tasarımcının simülasyonlar sonucunda sistemin davranışı üzerinde elde etmiş olduğu bilgi ve deneyimlerden faydalanılmıştır. K_p ve K_i katsayılarının sınırları tamamen deneysel olarak belirlendi.

7.2 İleriye Dönük Öneriler

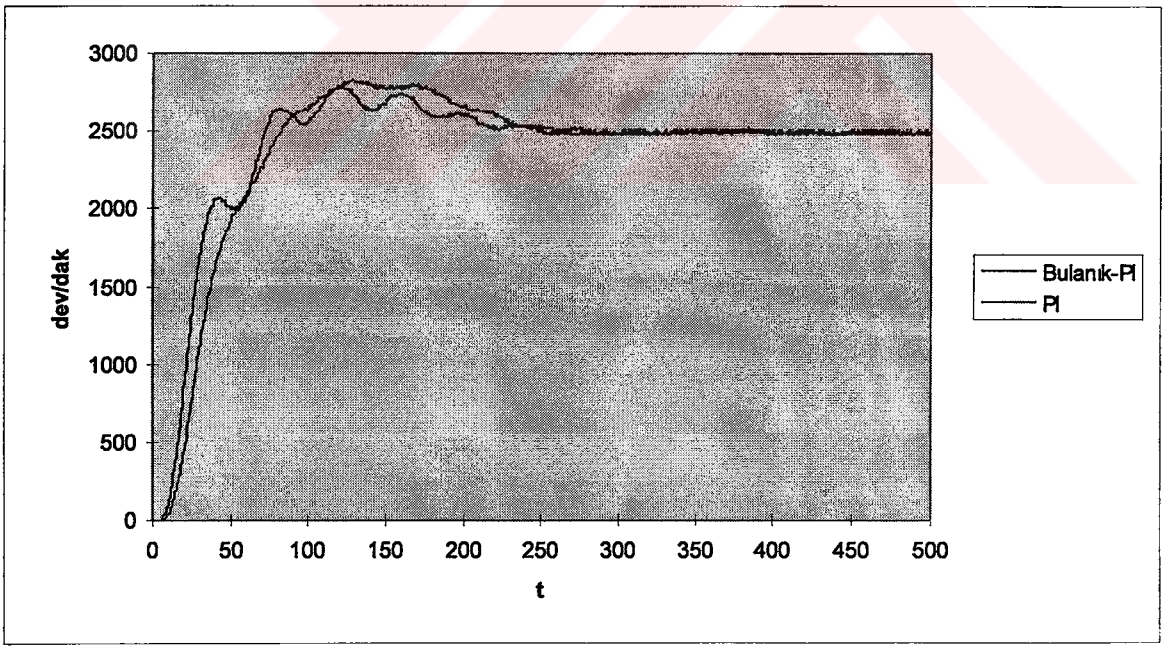
Daha ilerde kullanılmak üzere bu sisteme yeni ilaveler yapılabilir. Öncelikle motor gücü arttırılarak endüstride kullanılan sistemlere uygulanabilinir. Bu işlemin yapılması ile motor gücü artarken atalet momenti J ve sürtünme momenti katsayısı B artacaktır. Bunların artması, sistemin oturma ve yükselme zamanlarının milisaniyeler mertebesinden saniyeler mertebesine doğru artmasına sebep olacaktır. Bu da mikrokontrolörlü sistemde PI kontrolörde örnekleme zamanının düşürülmesi ve bulanık kontrolörde ise kurallar ve üyelik fonksiyonlarının değiştirilmesi ile oluşturulan yeni Bulanık-PI kontrolör kontrol işlemini başarıyla yerine

getirecektir. Kurallar ve üyelik fonksiyonlarının oluşturulabilmesi için yapay sinir ağları kullanılabilinir. Bu sayede bir öğrenme algoritması vasıtasıyla kurallar ve üyelik fonksiyonları otomatik olarak belirlenmiş olur. Böylece en büyük problem olan ve her sistem için değişen katsayıları ayarlama problemi çözülmüş olur.

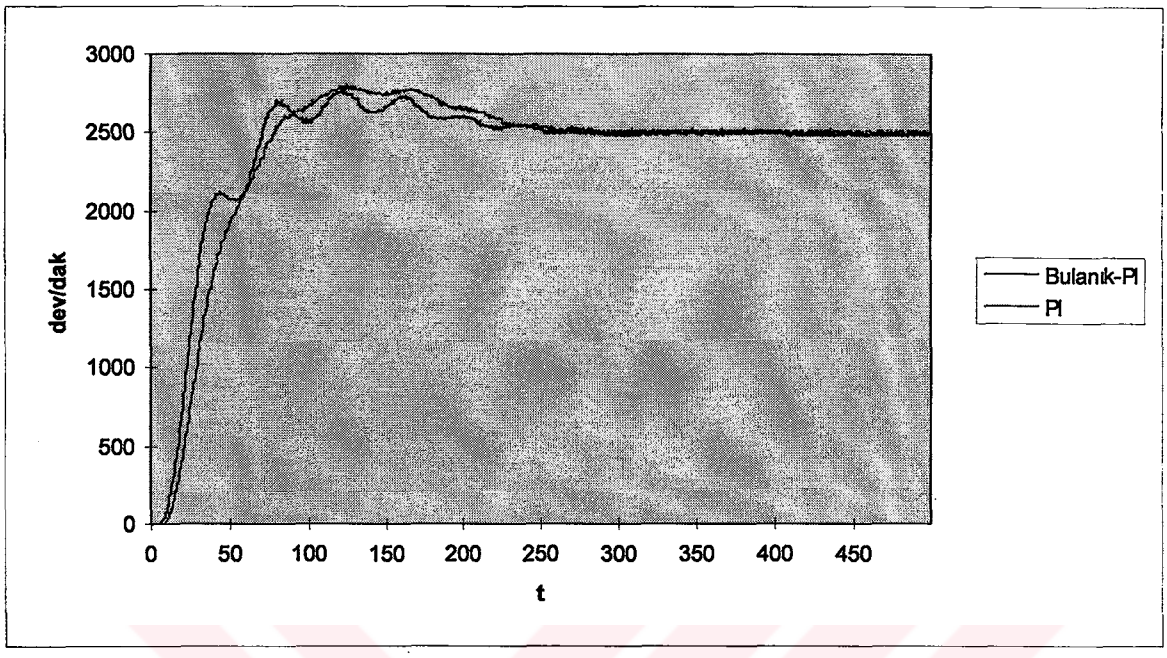
Yüksek örnekleme zamanına ihtiyaç duyan sistemlerde bu yöntem kullanılmak istenirse mikrokontrolöre nazaran daha hızlı bir işlemci gerekecektir. Bu durumda mikrokontrolör için yazılan yazılım DSP'ye uygulanabilinir.

Çizelge 7.1 Kontrolörlerin performanslarının karşılaştırılması

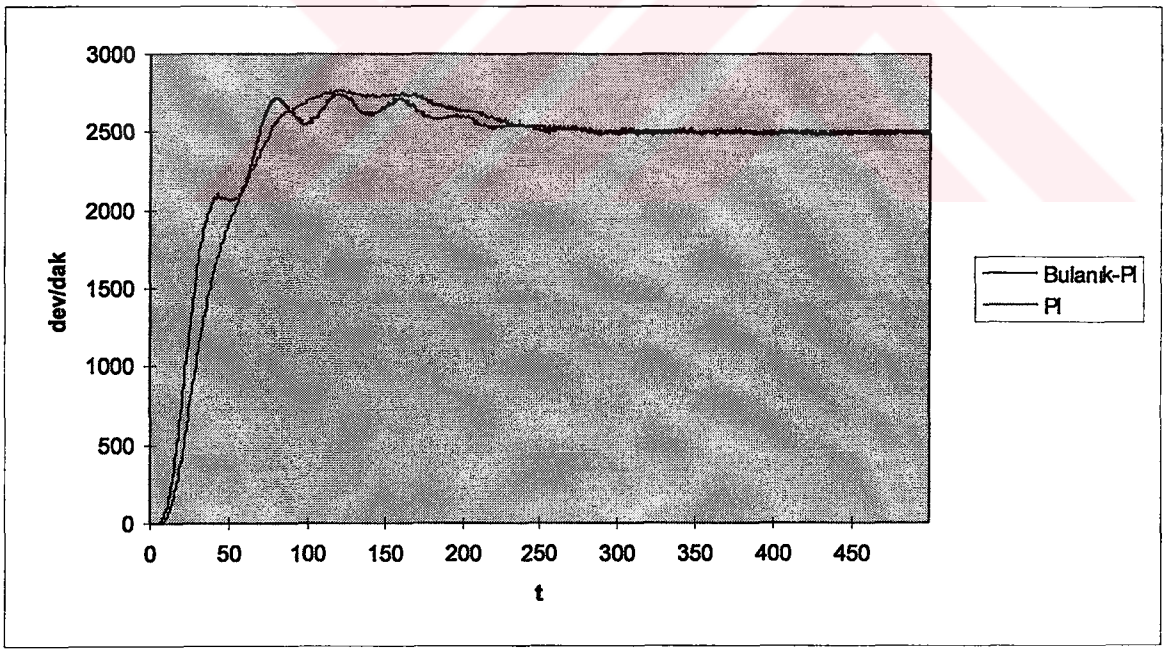
Sistem	Yük	Aşım	ts	tr	ITAE	ISE
Bulanık-PI	52W	289d/d	177ms	73ms	%4.4	%8.14
PI	52W	335d/d	217ms	84ms	%5.74	%10.4
Bulanık-PI	45W	276d/d	177.5ms	72ms	%4.44	%8.08
PI	45W	312d/d	216ms	80ms	%5.697	%10.14
Bulanık-PI	39W	241d/d	174ms	70ms	%4.4	%7.95
PI	39W	276d/d	213ms	77ms	%5.67	%9.9642



Şekil 7.2 52W yük için Bulanık-PI ile PI kontrolörlerinin hız-zaman eğrileri



Şekil 7.3 45W yük için Bulanık-PI ile PI kontrolörlerinin hız-zaman eğrileri



Şekil 7.4 39W yük için Bulanık-PI ile PI kontrolörlerinin hız-zaman eğrileri

KAYNAKLAR

Akalın, G., Kocaarslan, İ., Yörükeren, N. ve Erfidan, T., (1998), "İki bölgeli bir güç sistemi için yük-frekans kontrolünde kullanılan PI kontrolörün kazancının bulanık mantık ile programlanması", TOK98 Otomatik Kontrol Bilimsel Toplantısı, İstanbul.

Aström, K.J. ve Wittenmark, B., (1989), "Adaptive Control, Addison Wesley Publication".

Auslander, D.M., Takahashi, Y., Rabins, M.J., (1974), "Introducing Systems and Control", Mc Graw-Hill Book Company.

Bezdek, J., (1993), "Editorial: Fuzzy models- What are they and why?", IEEE Trans. Fuzzy Syst., vol. 1, pp. 1-5.

Chapman, J.S., (1985), "Electric Machinery Fundamentals", Mc Graw-Hill Book Company.

Dong, Y.K. ve Seong, P.H., (1996), "Fuzzy gain scheduling of velocity PI controller with intellegent learning algorithm for reactor control", Annual Nuclear Energy, vol. 24, No. 10, pp. 819-827.

Dubois, D. ve Prade, H., (1980), "Fuzzy sets and systems: Theory and applications", Academic, Newyork.

Erdun, H. ve Demiralp, F., (1994), "Turbo C Programlama Dili", Beta Basım Yayım Dağıtım A.Ş.

Fitzgerald, A.E., Kingsley, C. ve Umans, S.D., (1992), "Electric Machinery", Fifth Edition SI Units, Mc Graw Hill.

Gole, A.M., Chapman, D.G., Daneshpooy, A., Davies, J.B., (1997), "Fuzzy logic control for HVDC Transmission", IEEE Transactions on Power Delivery, vol. 12, Iss. 4, pp. 1690-1697.

Guillemin, P., (1994), "Universal motor control with fuzzy logic", Fuzzy Sets and Systems, vol. 63, Iss. 3, pp. 339-348

Hellendoorn, H. Ve Thomas, C., (1993), "Defuzzification in fuzzy controllers", Intelligent and Fuzzy Systems, vol. 1, pp. 109-123.

Jamshidi, M., Vadiee, N. ve Ross, T., (1993), "Fuzzy Logic and Control: Software and Hardware Applications", Prentice Hall, Englewood Cliffs, N.J.

Ketata, R., Degeest, D. ve Titli, A., (1995), "Fuzzy controller - design, evaluation, parallel and Hierarchical combination with PID controller", Fuzzy Sets and Systems, vol. 71, Iss 1, pp. 113-129.

Kuo, B.C. ve Hanselman, D.C., (1994), "Matlab Tools For Control System Analysis and Design", Prentice Hall.

Ramshaw, R. ve Heeswijk, V., (1990), "Energy Conversion Electric Motors and Generators", Mc Graw Hill.

Kuo, B.C., (1992), "Digital Control Systems", Saunders College Publishers.

Lee, C., (1990), "Fuzzy logic in control systems: fuzzy logic controller", Parts I and II, IEEE Trans. Syst., Man & Cybern., vol. 20, pp. 404-435

Mikrochip, (1995), "PIC16/17 Mikrocontroller Data Book".

Mikrochip Embedded, (1995), "Embedded Control Handbook".

Nachtigal, C.L., (1990), "Instrumentation and Control, Fundamentals and Applications", Wiley İnterscience Publication.

Norton, P. ve Wilton, R., (1985), "Programmer's Guide to IBM PC & PS/2", Microsoft Press.

Ogata, K., (1970), "Modern Control Engineering", Prentice-Hall.

OrCAD, 1989a. "OrCAD Schematic Design Tools Users Guide".

OrCAD, 1989b. "OrCAD PCB II Users Guide".

Özgüven, Ö.F., (1996), "80196 - 16 bitlik mikrodenetleyicili geliştirme seti tasarımı ve FP 3000 fuzzy işlemci kullanarak çok amaçlı mikrodenetleyicili fuzzy lojik kontrol ve uygulaması", doktora tezi.

Phillips, C., Harbor, R., (1988), "Feedback Control Systems", Prentice Hall.

Paraskevopoulos, P.N., (1996), "Digital Control Systems", Prentice Hall.

Raven, F.H., (1987), "Automatic Control Engineering", Mc Graw Hill .

Ross, T.J., (1995), "Fuzzy Logic with Engineering Applications", Mc Graw-Hill, Inc.

Sarıoğlu, K. (1992), "Dijital Kontrol Sistemleri", Sistem Yayıncılık.

Schweber, W.L., (1988), "Data Communications", Mc Graw-Hill Book Company.

Wildi, T., (1997), "Electrical Machines, Drives and Power Systems", Prentice Hall.

Zadeh, L., (1973), "Outline of a new approach to the analysis of complex systems and decision processes", IEEE Trans. Syst., Man. Cybern., vol. SMC-3, pp. 28-44.

Zadeh, L., (1965), "Fuzzy sets, Inf. Control", vol. 8, pp. 338-353.

Zhao, Z.Y., Tomizuka, M. Ve Isaka, S., (1993), "Fuzzy Gain Scheduling of PID controllers", IEEE Transactions on Systems, Man and Cybernetics, vol. 23, No. 5, pp. 1392-1398.

Zimmerman, H., (1991), "Fuzzy Set Theory and It's Applications", 2nd ed., Kluver Academic Publishers, Dordrecht, Germany.

EKLER

EK-1

Aşağıda verilen program Borland-C dilinde yazılmıştır (Erdun ve Demiralp, 1994). Bu program sayesinde mikrokontrolör, bilgisayar ile RS232 üzerinden seri porttan haberleşerek sistemin çıkış bilgisi olan hız bilgisinin grafik ekranda zamana bağlı değişimi çizilebilmektedir. Aynı zamanda gelen hız bilgisi bir text dosyasına çevrilerek daha sonra üzerinde işlem yapabilmek amacıyla saklanmaktadır. Normal derleyiciler 38400 baudrate haberleşme hızına çıkamadıkları için bu programda bilgisayarın haberleşme çipi 8250'ye doğrudan assembler dilinde yazılan program ile müdahele edilmiştir. Böylece çipin hızı arttırılmıştır (Norton ve Wilton, 1985).

```
/* Bu program 38400 Baudrate'de COM1 portunu kullanmaktadır. */

#include <dos.h>
#include <conio.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <graphics.h>
#include <stdlib.h>
#define PIXEL_COUNT 1000
#define DELAY_TIME  100  /* in milliseconds */
#define RHR  0x03f8
#define THR  0x03f8
#define COM1 0x03f8
#define MSB  0x03f9
#define LSB  0x03f8
#define IER  0x03f9
#define FCR  0x03fa

#define ISR  0x03fa
#define LCR  0x03fb
#define MCR  0x03fc
#define LSR  0x03fd
```

```
#define MSR  0x03fe
#define SPR  0x03ff
#define BPSL  03
#define BPSH  00
#pragma inline

main()
{
yeniden:
    unsigned char say;
    unsigned int Pkat,Nkat,Ikat;
    char msg[100];
    char Sa;
    int m=0;
    FILE *out;
    unsigned int A[2000], B[2000], C[10];
    int gdriver = DETECT, gmode, errorcode;
    int i, x, y, devir, color, maxx, maxy,
    maxcolor, seed;
    printf(" N katsayisi :");
    scanf("%d",&Nkat);
    printf(" P katsayisi :");
    scanf("%d",&Pkat);
    printf(" I katsayisi :");

    scanf("%d",&Ikat);
/* initialize graphics and local variables */
   initgraph(&gdriver, &gmode, "");

/* read result of initialization */
   errorcode = graphresult();

/* an error occurred */
   if (errorcode != grOk)
```

```
    {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Press any key to halt:");
    getch();
/* terminate with an error code */
    exit(1);
   }
    devir = (Nkat/11.76470588);
    setcolor(15);
    line(620,300,20,300);
    line(30,330,30,20);
    setcolor(14);
    sprintf(msg,"Devir : %d Ikat : %d Pkat : %d", Nkat, Ikat, Pkat);
    outtextxy( 100, 440, msg);
    sprintf(msg,"%d", 3000);
    outtextxy( 0, 41, msg);
    sprintf(msg,"%d", 2500);
    outtextxy( 0, 84, msg);

    sprintf(msg,"%d", 2000);
    outtextxy( 0, 127, msg);
    sprintf(msg,"%d", 1500);
    outtextxy( 0, 170, msg);
    sprintf(msg,"%d", 1000);
    outtextxy( 0, 213, msg);
    sprintf(msg,"%d", 500);
    outtextxy( 0, 256, msg);
    sprintf(msg,"%d", 0);
    outtextxy( 0, 299, msg);
/* initialization of UART 16550 & 8250 */
asm {
     mov dx,LCR  /* divisor latch enable */
     mov al,83h
     out dx,al
```

```
mov dx,LSB   /*  lsb of dlatch */
mov al,BPSL
out dx,al
mov dx,MSB   /*  msb of dlatch */
mov al,BPSH
out dx,al
mov dx,LCR   /* dlatch disable */
mov al,03h
out dx,al
mov dx,IER   /* SETS THE INTERRUPTS */
mov al,05h
out dx,al

mov dx,FCR   /* ENABLE FIFO */
mov al,01h
out dx,al
mov dx,FCR   /*  RESET TRANS. RECEIV. FIFO */
mov al,07h
out dx,al
/* to get data in receive register */
mov dx,COM1
in al,dx
mov say,al
 }
delay(1);
asm  { mov dx,COM1
mov al,'B'
out dx,al }
delay(1);
asm  {
mov dx,COM1
mov al,'H'
out dx,al }
delay(1);
```

```
    asm  {
    mov dx,COM1
    mov al,Pkat
    out dx,al }
    delay(1);
    asm  {
    mov dx,COM1

    mov al,Ikat
    out dx,al }
    delay(1);
    asm  {
    mov dx,COM1
    mov al,devir
    out dx,al }
geldi:
    asm {
    mov dx,LSR
    in al,dx
    mov cl,1
    and al,cl
    cmp al,1
    jne geldi
    mov dx,COM1
    in al,dx
    mov say,al
          }
    m++;
    C[m] = say;
    if(m >= 3) goto assa1;
    goto geldi;
assa1:  m=0;
    if( Pkat == C[1] && Ikat == C[2] && devir == C[3]) goto devam;
    setcolor(4);
```

```
    sprintf(msg,"Error in Data Verification ");
    outtextxy( 0, 0, msg);
    getch();
    goto ende;
devam:
    setcolor(4);
    sprintf(msg,"Data is Successfully verified ");
    outtextxy( 0, 0, msg);
 /*       getch(); */
/* routine to capture data from recieve buffer*/
geld:
    asm {
    mov dx,LSR
    in al,dx
    mov cl,1
    and al,cl
    cmp al,1
    jne geld
    mov dx,COM1
    in al,dx
    mov say,al
          }
    m++;
    A[m] = say;
    if(m > 630) goto assa;
    goto geld;
assa:   m=0;
     /* for(m=0;m < 10 ;m++) printf("%c ", A[m]);*/
     /*   getch(); */
    setcolor(4);
    line(30,300-devir,640,300-devir);
    /* data ploting */
    for(x=1; x < 631; x++){
    putpixel(x+29,300-B[x] , 0 );
```

```
    putpixel(x+29,300-A[x] , 15);
            }
    /* file open */
    out = fopen("C:\\THESIS\\THESIS.TXT", "wt");
    for(x=1; x < 630; x++)
    {
    B[x] = A[x] ;
    fprintf(out,"%d %d\n",x,A[x]);
         }
    fclose(out);
    getch();
    /*    goto geld; */
ende:
     closegraph();
     return 0;
         }
    END
```

EK-2

Bu program 16C65 mikrokontrolörünün assembler dilinde yazılmıştır. Bu programda PI kontrolör algoritması bulunmaktadır.

```
LIST    p=16C65 ; PIC16C65 is the target processor
                ;
                ; An application of asencron
                ; comunucation
                ; Processor 16C65

PIC65       equ     00  ; Define reset vectors
pc          equ     02  ; program counter
porta       equ     05
portb       equ     06
portc       equ     07
portd       equ     08
TXREG       equ     19
RCREG       equ     1A
T1CON       equ     10
TXSTA       equ     98
RCSTA       equ     18
SPBRG       equ     99
PIE1        equ     8C
PIR1        equ     0C
TMR1L       equ     0E
CCP1CON     equ     17
CCP2CON     equ     1d
PR2         equ     92
T2CON       equ     12
DC1         equ     15
TRISA       equ     85
TRISB       equ     86
TRISC       equ     87
TRISD       equ     88
TRISE       equ     89
STATUS      equ     03  ; contains some useful flags
CARRY       equ     00
Z           equ     02
Same        equ     01
Kp          equ     20
Ki          equ     21
G1          equ     22
VELA        equ     23
VELR        equ     24
FLAGS       equ     25
ER_SGN      equ     00
EROR        equ     26
mulplr      equ     27
mulcnd      equ     28
H_byte      equ     29
L_byte      equ     2A
```

```
YHI         equ     2B
YLO         equ     2C
XHI         equ     2D
XLO         equ     2E
ACCaHI      equ     2F
ACCaLO      equ     30
ACCbHI      equ     31
ACCbLO      equ     32
G2          equ     33

        org PIC65

start

        clrf porta
        clrf portb
        clrf portc
        clrf portd
        bsf STATUS,5
        movlw b'11111111'     ;
        movwf TRISA
        movlw b'10111011'     ;
        movwf TRISC
        movlw b'11111111'     ;
        movwf TRISB
        bcf TRISE,4
        movlw b'00000000'
        movwf TRISD
        movlw b'00100100'  ;high speed
        movwf TXSTA
        movlw d'32'      ;38400 baudrate
        movwf SPBRG
        movlw b'11111111'

        movwf PIE1
        bcf STATUS,5
        bcf STATUS,6
        movlw b'10010000'  ;high speed
        movwf RCSTA

pwm
        movlw b'00001100'
        movwf CCP1CON
        movlw b'00001100'
        movwf CCP2CON
        bsf STATUS,5
        movlw d'255'
        movwf PR2
        bcf STATUS,5
        movlw b'00000100'
        movwf T2CON
```

```
seriwait
        clrf XHI
        clrf XLO
        clrf DC1
        bcf portd,0
        bsf portd,1

kod1
         btfss PIR1,5


        goto kod1
        movlw d'66'
        subwf RCREG,0
        btfss STATUS,2
        goto kod1

kod2
        btfss PIR1,5
        goto kod2
        movlw d'72'
        subwf RCREG,0
        btfss STATUS,2
        goto kod2

data1
        btfss PIR1,5
        goto data1
        movf RCREG,0
        movwf Kp

data2
        btfss PIR1,5
        goto data2
        movf RCREG,0
        movwf Ki

data3
        btfss PIR1,5

        goto data3
        movf RCREG,0
        movwf VELR
        movf Kp,0
        movwf TXREG
        bsf STATUS,5

B1
        btfss TXSTA,1
        goto B1
        bcf STATUS,5
        movf Ki,0
```

```
        movwf TXREG
        bsf STATUS,5

B2
        btfss TXSTA,1
        goto B2
        bcf STATUS,5
        movf VELR,0
        movwf TXREG
        bsf STATUS,5

B3
        btfss TXSTA,1
        goto B3
        bcf STATUS,5


bekle
        btfss porta,4    ;start butonu
        goto main
        goto bekle

main
        bsf portd,0
        bcf portd,1
        btfss porta,3      ;stop butonu
        goto seriwait
        movf portb,0       ;ADC gelen bilgiyi al
        movwf VELA
        movf portb,0       ;Hiz bilgisini seriden gonder
        movwf TXREG
; Hatanın hesaplanması
C_ERR
        movf VELA,0        ;ADC`den gelen hiz bilgisi
        subwf VELR,0       ;istenen referans hiz
        btfsc STATUS,CARRY
        goto PLS_ER
        goto MNS_ER
;Hatanın polaritesinin belirlenmesi
PLS_ER
        movwf EROR
        bcf FLAGS,ER_SGN
        goto CE_EXIT

MNS_ER

        movf VELR,0
        subwf VELA,0
        movwf EROR
        bsf FLAGS,ER_SGN

CE_EXIT
```

```
        movf EROR,0
        movwf mulplr
        movf Kp,0
        movwf mulcnd
        call carpma       ;Kp*EROR sonuc YHI ve YLO da saklanir.
        movf H_byte,0
        movwf YHI
        movf L_byte,0
        movwf YLO
        movf XHI,0
        movwf ACCbHI
        movf XLO,0
        movwf ACCbLO
        movf EROR,0       ;Ki*EROR
        movwf mulplr
        movf Ki,0
        movwf mulcnd
        call carpma
        rrf L_byte
        bcf STATUS,CARRY
        rrf L_byte

        bcf STATUS,CARRY
        rrf L_byte
        bcf STATUS,CARRY
        bcf STATUS,CARRY
        rrf H_byte
        btfsc STATUS,CARRY
        goto s1
        goto s2

s1
        bsf L_byte,7
        goto d1

s2
        bcf L_byte,7

d1
        bcf STATUS,CARRY
        rrf H_byte
        btfsc STATUS,CARRY
        goto s3
        goto s4

s3
        bsf L_byte,6
        goto d2

s4
        bcf L_byte,6
```

```
d2
        bcf STATUS,CARRY
        rrf H_byte
        btfsc STATUS,CARRY
        goto s5
        goto s6

s5
        bsf L_byte,5
        goto d3

s6
        bcf L_byte,5

d3
        movf H_byte,0
        movwf ACCaHI
        movf L_byte,0
        movwf ACCaLO
        btfsc FLAGS,ER_SGN
        goto mn
        goto pl

pl
        call toplama
        movf ACCbHI,0
        movwf XHI
        movf ACCbLO,0

        movwf XLO
        goto ex

mn
        call cikarma
        movf ACCbHI,0
        movwf XHI
        movf ACCbLO,0
        movwf XLO

ex
        movf XHI,0
        movwf ACCbHI
        movf XLO,0
        movwf ACCbLO
        movf YHI,0
        movwf ACCaHI
        movf YLO,0
        movwf ACCaLO
        btfsc FLAGS,ER_SGN
        goto cik
        goto top
```

```
top
        call toplama
        movf ACCbHI,0
        movwf DC1       ;PWM registerina yukleme yaptik.
        goto cikis

cik
        call cikarma
        movf ACCbHI,0
        movwf DC1       ;PWM registerina yukleme yaptik.

cikis
        movlw d'5'
        movwf G1
        movlw d'4'
        movwf G2

again                   ;465 usaniye gecikme
        decfsz G1       ;pi 35 usaniye tutuyor
        goto again
        decfsz G2
        goto again
        goto main

pma
mpy_F
        clrf H_byte
        clrf L_byte
        movf mulcnd,0
        bcf STATUS,CARRY

mult0
        btfsc mulplr,0
        addwf H_byte,Same

        rrf H_byte,Same
        rrf L_byte,Same

mult1
        btfsc mulplr,1
        addwf H_byte,Same
        rrf H_byte,Same
        rrf L_byte,Same

mult2
        btfsc mulplr,2
        addwf H_byte,Same
        rrf H_byte,Same
        rrf L_byte,Same

mult3
        btfsc mulplr,3
```

```
        addwf H_byte,Same
        rrf H_byte,Same
        rrf L_byte,Same

mult4
        btfsc mulplr,4
        addwf H_byte,Same
        rrf H_byte,Same
        rrf L_byte,Same

mult5

        btfsc mulplr,5
        addwf H_byte,Same
        rrf H_byte,Same
        rrf L_byte,Same

mult6
        btfsc mulplr,6
        addwf H_byte,Same
        rrf H_byte,Same
        rrf L_byte,Same

mult7
        btfsc mulplr,7
        addwf H_byte,Same
        rrf H_byte,Same
        rrf L_byte,Same
        retlw 0

toplama
        nop
        nop
        nop
        nop
        nop
        movf ACCaLO,0
        addwf ACCbLO
        btfsc STATUS,CARRY
        incf ACCbHI

        movf ACCaHI,0
        addwf ACCbHI
        btfsc STATUS,CARRY
        goto overflo
        goto noprob

overflo
        movlw d'255'
        movwf ACCaLO
        movwf ACCbLO
        movwf ACCaHI
```

```
        movwf ACCbHI

noprob
        retlw 0

cikarma
        comf ACCaLO
        incf ACCaLO
        btfsc STATUS,Z
        decf ACCaHI
        comf ACCaHI
        movf ACCaLO,0
        addwf ACCbLO
        btfsc STATUS,CARRY
        incf ACCbHI
        movf ACCaHI,0
        addwf ACCbHI

        btfsc STATUS,CARRY
        goto nopr
        goto over

over
        movlw d'00'
        movwf ACCaLO
        movwf ACCbLO
        movwf ACCaHI
        movwf ACCbHI

nopr
        retlw 0
        END
```

EK-3

Aşağıda oran katsayısı için bulanık işlemlerin assembler dilinde yazılmış programı verilmiştir.

```
LIST P=16C71
;-----------------------------------------------------------------
;- USER MAIN FILE                                                -
;-----------------------------------------------------------------
   CODE_START          EQU    0x100  ;code startadr for 16C71
   RESET_ADR           EQU    0x000  ;reset vector
   FUZZY_RAM_START EQU    0x00C  ;first free RAM location for 16C71
   ADCON0              EQU    0x008
   ADCON1              EQU    0x088
   ADRES               EQU    0x009
   VELR                EQU    0x02E
   VELA                EQU    0x02F
   porta          EQU    0x005
   portb          EQU    0x006
   TRISA               EQU    0x085
   TRISB               EQU    0x086
   status              EQU    0x003
   carry               EQU    0x000
        include "myprojp.var"   ;include preassembler variables
        CBLOCK                  ;starts after fuzzy ram locations
        user1           ;reserve 1 byte (example)
        ENDC
        ORG CODE_START  ;example start adress for code


start
        call  initmyprojp       ;call init once
        bsf status,5
        movlw b'11111111'
        movwf TRISA     ;a portu giris
        movlw b'00000000'
        movwf TRISB
        bcf status,5
        clrf portb

bekle1
        movlw d'212'
        movwf VELR

ADC
        bsf status,5
        movlw b'00000000'
        movwf ADCON1
        bcf status,5
        movlw b'00001001'
        movwf ADCON0

zirt
```

```
        bsf ADCON0,2

loop
        btfsc ADCON0,2
        goto loop

        movf ADRES,0
        movwf VELA

C_ERR
        movf VELA,0
        subwf VELR,0
        btfsc status,carry
        goto pls_er
        goto mns_er

pls_er
        movwf lv0_ERROR
        goto ce_exit

mns_er
        movf VELR,0
        subwf VELA,0
        movwf lv0_ERROR

ce_exit
main_loop
        call   myprojp    ;call preassembler code
        movf   invalidflags,W
        btfss  _Z                 ;test if the project is completely defined
        goto   case_no_fire

case_fire

        ;proj OK
        movf   lv1_Kp,W   ;fetch crisp output
        movwf portb
        ;user code
        goto   ADC

case_no_fire
        ;no rule defined for this input combination
        ;call default_handling_routine
        ;user code
        goto   ADC
        INCLUDE "myprojp.asm"   ;include preassembler code
;-----------------------------------------------------------------
;- RESET VECTOR                                                  -
;-----------------------------------------------------------------
        ORG    RESET_ADR
        goto   start        ;jump to program code
        END                 ;end for assembler (only here)
```

Myprojp.asm

```
;---------- eğimlerin verildiği terim tanımlamaları (x1, a_s, x3, d_s) ----------
tpts
        DW    03400H, 03400H, 03400H, 03408H
        DW    03400H, 03408H, 03440H, 03408H
        DW    03440H, 03408H, 03480H, 03408H
        DW    0347FH, 03408H, 034BFH, 03408H
        DW    034BFH, 03408H, 034FFH, 03400H

;---------- xcom table (netleştirme) ----------
xcom

        DW    0340FH, 03431H
;---------- kural tabanı ----------
rt0
        DW    03405H
        DW    03401H,03401H, 03400H, 03405H
        DW    03401H,03401H, 03401H, 03406H
        DW    03401H,03401H, 03402H, 03406H
        DW    03401H,03401H, 03403H, 03406H
        DW    03401H,03401H, 03404H, 03406H
;---------- initmyprojp ----------
initmyprojp
        movlw  fuzvals + .5
        movwf  FSR
        movlw  .2
        movwf  loopcnt
        movlw  0
initloop
        movwf  0
        incf   FSR
        decfsz loopcnt
        goto   initloop
        return
----------
myprojp
        movlw  fuzvals
        movwf  FSR
        clrf   itptr
;---------- bulanıklaştırma ----------
        movlw  .5
        movwf  itcnt
        movf   lv0_ERROR,W
        movwf  crisp
        call   flmss
 ;---------- çıkarım ----------
        LD_TBL16   rt0
        call   Min                ;min aggregation
;---------- netleştirme ----------
        clrf   invalidflags
        clrf   otoffset
        movlw  fuzvals + .5
```

```
        movwf  FSR
        movlw  .2
        movwf  otcnt
        call   com
        rlf    invalidflags
        btfsc  invalidflags,0
        movlw  0x00F
        movwf  lv1_Kp
        return
        include "flmss.asm"          ;"FLMSS.ASM"
        include "Min.asm"            ;"MIN.ASM"
        include "com.asm"            ;"COM.ASM"
        include "mpy8_8.asm"         ;"MPY8_8.ASM"
        include "div16_8.asm"        ;"DIV16_8.ASM"

;data size knowledge base (bytes):
;RAM:       10  0000AH
;ROM:       43  0002BH

;TOTAL:     53  00035H
```

Aşağıda integral katsayısı için bulanık işlemlerin assembler dilinde yazılmış programı verilmiştir.

```
PROCESSOR 16C71

;-----------------------------------------------------------
;- USER MAIN FILE                                          -
;-----------------------------------------------------------
   CODE_START          EQU   0x100  ;code startadr for 16C71
   RESET_ADR           EQU   0x000  ;reset vector
   FUZZY_RAM_START EQU   0x00C  ;first free RAM location for 16C71
   ADCON0              EQU   0x008
   ADCON1              EQU   0x088
   ADRES               EQU   0x009
   VELR                EQU   0x02E
   VELA                EQU   0x02F
   sign                EQU   0x02D
   porta               EQU   0x005
   portb               EQU   0x006
   TRISA               EQU   0x085
   TRISB               EQU   0x086
   status              EQU   0x003
   carry               EQU   0x000
        include "myprojs.var"   ;include preassembler variables
        CBLOCK                  ;starts after fuzzy ram locations
        user1               ;reserve 1 byte (example)

        ENDC
        ORG CODE_START     ;example start adress for code
start
        call   initmyprojs         ;call init once
```

```
        bsf status,5
        movlw b'11111111'
        movwf TRISA         ;a portu giris
        movlw b'00000000'
        movwf TRISB
        bcf status,5
        clrf portb
bekle1
        movlw d'212'
        movwf VELR

ADC
        bsf status,5
        movlw b'00000000'
        movwf ADCON1
        bcf status,5
        movlw b'00001001'
        movwf ADCON0

zirt
        bsf ADCON0,2

loop
        btfsc ADCON0,2
        goto loop
        movf ADRES,0
        movwf VELA

C_ERR
        movf VELA,0
        subwf VELR,0
        btfsc status,carry
        goto pls_er
        goto mns_er

pls_er
        bsf sign,1
        movwf lv0_ERROR
        goto ce_exit

mns_er
        bcf sign,1
        movf VELR,0
        subwf VELA,0
        movwf lv0_ERROR

ce_exit
        btfsc sign,1
        goto pos
        goto neg
```

```
pos
        bcf status,carry
        rrf lv0_ERROR,0
        addlw d'127'
        movwf lv0_ERROR
        goto main_loop

neg
        bcf status,carry
        rrf lv0_ERROR,0
        sublw d'127'
        movwf lv0_ERROR

main_loop
        call   myprojs    ;call preassembler code
        movf   invalidflags,W
        btfss  _Z                 ;test if the project is completely defined
        goto   case_no_fire

case_fire
        ;proj OK
        movf lv1_Ki,W    ;fetch crisp output
        movwf portb
        ;user code
        goto   ADC

case_no_fire
        ;no rule defined for this input combination
        ;call default_handling_routine
        ;user code
        goto   ADC
        INCLUDE "myprojs.asm"    ;include preassembler code
;-----------------------------------------------------------------
;- RESET VECTOR                                                  -
;-----------------------------------------------------------------
        ORG    RESET_ADR
        goto   start        ;jump to program code
        END
Myprojs.asm
;---------- eğimlerin verildiği terim tanımlamaları (x1, a_s, x3, d_s) ----------
tpts
          DW    03400H, 03400H, 0342AH, 03408H
          DW    0343CH, 0340CH, 03466H, 03414H
          DW    03470H, 03422H, 0347FH, 03422H
          DW    0347FH, 03412H, 0349CH, 0340CH
          DW    034AAH, 0340CH, 034FFH, 03400H
;---------- xcom table (netleştirme) ----------
xcom
          DW    0340FH, 03415H, 0341DH, 03428H, 03432H
;---------- kural tabanı ----------
rt0
          DW    03405H
```

```
        DW    03401H,03401H, 03400H, 03409H
        DW    03401H,03401H, 03401H, 03407H
        DW    03401H,03401H, 03402H, 03406H

        DW    03401H,03401H, 03403H, 03407H
        DW    03401H,03401H, 03404H, 03408H
;---------------------------- initmyprojs ----------------------------
initmyprojs
        movlw  fuzvals + .5
        movwf  FSR
        movlw  .5
        movwf  loopcnt
        movlw  0
initloop
        movwf  0
        incf   FSR
        decfsz loopcnt
        goto   initloop
        return
        movlw  fuzvals
        movwf  FSR
        clrf   itptr
;---------------------------- bulanıklaştırma ----------------------------
        movlw  .5
        movwf  itcnt
        movf   lv0_ERROR,W
        movwf  crisp
        call   flmss
;---------------------------- çıkarım ----------------------------
        LD_TBL16   rt0
        call   Min                ;min aggregation
;---------------------------- netleştirme ----------------------------
        clrf   invalidflags
        clrf   otoffset
        movlw  fuzvals + .5
        movwf  FSR
        movlw  .5
        movwf  otcnt
        call   com
        rlf    invalidflags
        btfsc  invalidflags,0
        movlw  0x00F
        movwf  lv1_Ki
        return
        include "flmss.asm"          ;"FLMSS.ASM"
        include "Min.asm"            ;"MIN.ASM"
        include "com.asm"            ;"COM.ASM"
        include "mpy8_8.asm"         ;"MPY8_8.ASM"
        include "div16_8.asm"        ;"DIV16_8.ASM"

;data size knowledge base (bytes):
;RAM:       13   0000DH
```

```
;ROM:     46  0002EH
;TOTAL:   59  0003BH
```

Her iki programda kullanılan alt programlar aşağıda verilmiştir.

Flmss.asm

```
  if     RESOLUTION == 8
GET_TERM   MACRO

        movwf   offset
        call    tpts_routine
        ENDM
        else
GET_TERM   MACRO  target
        movwf   offset
        bcf     _C
        rlf     offset
        call    tpts_routine
        movwf   target +0
        incf    offset
        call    tpts_routine
        movwf   target +1
        ENDM
        endif

if      FAMILY == 2
tpts_routine
        LD16PC          tpts,offset
flmss
        else
flmss_
        endif
        flmss_start
        movf    itptr,W
        if      FAMILY == 2
        if  RESOLUTION == 8
        GET_TERM
        movwf   cur_term
        subwf   crisp,W
        else
        GET_TERM   cur_term
        FSUB16F    cur_term,crisp,Temp16
        endif
        else
        call    tpts
        movwf   cur_term
        subwf   crisp,W
        endif
        btfsc   _C
        goto    cmp_flmss
        movlw   .1
```

```
        addwf   itptr,W
        if      FAMILY == 2
        if  RESOLUTION == 8
        GET_TERM
        andlw   0x0FF
        else
        GET_TERM  Temp16
        iorwf   Temp16+0,W
        endif
        else
        call    tpts
        andlw   0x0FF
        endif
        btfss   _Z
        goto    mindone
        goto    maxdone
cmp_flmss
        movlw   .2
        addwf   itptr,W
        if      FAMILY == 2
        if  RESOLUTION == 8
        GET_TERM
        movwf   cur_term2
        movf    crisp,W
        subwf   cur_term2,W
        else
        GET_TERM  cur_term2
        FSUB16W   crisp,cur_term2
        endif
        else
        call    tpts
        movwf   cur_term2
        movf    crisp,W
        subwf   cur_term2,W
        endif
        btfss   _C
        goto    ds
as
        movlw   .1
        addwf   itptr,W
        if      FAMILY == 2
        if  RESOLUTION == 8
        GET_TERM
        andlw   0x0FF
        else
        GET_TERM  Temp16
        iorwf   Temp16+0,W
        endif
        else
        call    tpts
        andlw   0x0FF
        endif
```

```
        btfsc   _Z
        goto    maxdone
        if      RESOLUTION == 8
        movwf   mulplr
        movf    cur_term,W
        subwf   crisp,W
        movwf   mulcnd
        call    mpy8_8
        bcf     _C
        rrf     res_mpy+0,W
        andlw   0x0FF
        btfss   _Z
        goto    maxdone
        rrf     res_mpy+1,W
    else
        FMOV16      Temp16,mulplr
        FSUB16F     cur_term,crisp,mulcnd
        call    mpy16_16
        bcf     _C
        rrf     res_mpy+0,W
        movwf   Temp16+0
        rrf     res_mpy+1,W
        movwf   Temp16+1
        iorwf   Temp16+0,W
        btfss   _Z
        goto    maxdone
        rrf     res_mpy+2,W
        movwf   Temp16+0
        rrf     res_mpy+3,W
        movwf   Temp16+1
        endif
        goto    store
ds
        movlw   .3
        addwf   itptr,W
        if      FAMILY == 2
        if  RESOLUTION == 8
        GET_TERM
        andlw   0x0FF
        else
        GET_TERM    Temp16
        iorwf   Temp16+0,W
        endif
        else
        call    tpts
        andlw   0x0FF
        endif
        btfsc   _Z
        goto    mindone
        if      RESOLUTION == 8
        movwf   mulplr
        movf    cur_term2,W
```

```
        subwf   crisp,W
        movwf   mulcnd
        call    mpy8_8
        bcf     _C
        rrf     res_mpy+0,W
        andlw   0x0FF
        btfss   _Z
        goto    mindone
        rrf     res_mpy+1
        comf    res_mpy+1,W
        else
        FMOV16      Temp16,mulplr
        FSUB16F  cur_term2,crisp,mulcnd
        call    mpy16_16
        bcf     _C
        rrf     res_mpy+0,W
        movwf   Temp16+0
        rrf     res_mpy+1,W
        movwf   Temp16+1
        iorwf   Temp16+0,W
        btfss   _Z
        goto    mindone
        rrf     res_mpy+2,W
        movwf   Temp16+0
        rrf     res_mpy+3,W
        movwf   Temp16+1
        FCOMF16  Temp16
        endif
        goto    store
maxdone
        if  RESOLUTION == 8
        movlw   0x0FF
        else
        movlw   0x0FF
        movwf   Temp16+0
        movwf   Temp16+1
        endif
        goto    store
mindone
        if  RESOLUTION == 8
        movlw   0x000
        else
        movlw   0x000
        movwf   Temp16+0
        movwf   Temp16+1
        endif
store
        if  RESOLUTION == 8
        if      FAMILY == 1
        SET_PAGE
        movwf   0
        incf    FSR
```

```
        bsf     FSR,4
        RES_PAGE
        else
        movwf   0
        incf    FSR
        endif
        else
        FMOV16F0  Temp16
        incf    FSR
        incf    FSR
        endif
        movlw   4
        addwf   itptr
        decfsz  itcnt
        goto    flmss_start
        if   FAMILY == 2
        return
        else
        retlw   0
        endif

; The end of module flmss

Min.asm

Min.asm


GET_TAB   MACRO
        if      FAMILY < 2
        movf    rtptr,W
        call    rt0
        else
        call    rt_routine
  endif
        incf    rtptr
          ENDM

if FAMILY ==      2
rt_routine
        LD16PTR rt_table,rtptr
        endif

Min
        if      FAMILY == 1
        IS_PAGE   fuzvals
        endif
        clrf    rtptr
        GET_TAB
        movwf   rulecount
start_rule_min
    if  RESOLUTION == 8
```

```
        movlw   0x0FF
        movwf   min_tmp
        else
        movlw   0x0FF
        movwf   min_tmp+0
        movwf   min_tmp+1
        endif
        GET_TAB
        movwf   no_i
        GET_TAB
        movwf   no_o
cmp_min
        GET_TAB
        ADD_PAGE
        if      FAMILY == 1
        SET_PAGE
        endif
        if  RESOLUTION == 8
        movf    0,W
        else
        FMOV0F16   Temp16
        endif
        if      FAMILY == 1
        RES_PAGE
        btfss   _Z
        goto    cmp_min_ff
        clrf    min_tmp
        movf    no_i,W
        addwf   rtptr
        decf    rtptr
        goto    comp_min
cmp_min_ff
        movwf   tmp_var
        endif
        if  RESOLUTION == 8
        subwf   min_tmp,W
        else
        FSUB16W    Temp16,min_tmp
        endif
        btfss   _C
        goto    next_in_min
        if    FAMILY      ==  1
        movf    tmp_var,W
        movwf   min_tmp
        else
        if  RESOLUTION == 8
        movf    0,W
        movwf   min_tmp
        else
        FMOV0F16   min_tmp
        endif
        endif
```

```
next_in_min
        decfsz    no_i
        goto      cmp_min
comp_min
        GET_TAB
        ADD_PAGE
        if        FAMILY == 1
        SET_PAGE
        endif
        if  RESOLUTION == 8
        movf      0,W
        else
        FMOV0F16   Temp16
        endif
        if        FAMILY == 1
        RES_PAGE
        endif
        if  RESOLUTION == 8
        subwf     min_tmp,W
        else
        FSUB16W    Temp16,min_tmp
        endif
        btfss     _C
        goto      next_out_min
        if  RESOLUTION == 8
        movf      min_tmp,W
        else
        FMOV16          min_tmp,Temp16
        endif
        if        FAMILY == 1
        SET_PAGE
        endif
        if  RESOLUTION == 8
        movwf   0
        else
        FMOV16F0   Temp16
        endif
        if        FAMILY == 1
        RES_PAGE
        endif
next_out_min
        decfsz    no_o
        goto      comp_min
        decfsz    rulecount
        goto      start_rule_min
        if       FAMILY == 2
        return
        endif

; The end of module inference minimum
```

Com.asm

Com.asm

```
GET_XCOM   MACRO  target
        if   RESOLUTION == 8
        movwf    offset
        call     xcom_routine
        movwf    target
        else
        movwf    offset
        bcf      _C
        rlf      offset
        call     xcom_routine
        movwf    target+0
        incf     offset
        call     xcom_routine
        movwf    target+1
        endif
          ENDM

        if FAMILY ==      2
xcom_routine
        LD16PC       xcom,offset
        endif

com
        if   RESOLUTION == 8
        FCLR24           num
        FCLR16           denom
        else
        FCLR32           num
        FCLR16           denom
        endif
start_com
        if   RESOLUTION == 8
        if         FAMILY == 1
        decf     otcnt
        SET_PAGE
        movf     0,W
        clrf     0
        bcf      FSR,4
        decf     FSR
        bsf      FSR,4
        RES_PAGE
        else
        if    FAMILY > 1
        movf     0,W
        clrf     0
        incf     FSR
        andlw    0x0FF
        else
        decf     otcnt
```

```
        movf    0,W
        clrf    0
        decf    FSR
        andlw   0x0FF
        endif
        endif
        else
        FMOV0F16   Temp16
        FCLR0F16
        incf    FSR
        incf    FSR
        bcf     _C
        FRR16   Temp16
        bcf     _C
        FRR16   Temp16
        bcf     _C
        FRR16   Temp16
        movf    Temp16+0,W
        iorwf   Temp16+1,W
        endif
        btfsc   _Z
        goto    ready_com
        if  RESOLUTION == 8
        movwf   mulplr
        FADD8TO16  mulplr, denom
        else
        FMOV16      Temp16,mulplr
        FADD16      Temp16,denom
        endif
        if      FAMILY < 2
        movf    otcnt,W
        call    xcom
        movwf   mulcnd
        call    mpy8_8
        FADD16TO24 res_mpy,num
        else
        movf    otoffset,W
        if  RESOLUTION == 8
        GET_XCOM   mulcnd
        call    mpy8_8
        FADD16TO24 res_mpy,num
        else
        GET_XCOM   mulcnd
        call    mpy16_16
        FADD32      res_mpy,num
        endif
        endif
ready_com
        if      FAMILY > 1
        incf    otoffset
        decfsz  otcnt
        else
```

```
        movf    otcnt
        btfss   _Z
        endif
        goto    start_com
        if  RESOLUTION == 8
ishift_com
        movf    numEXT,W
        btfss   _Z
        goto    ishift_com2
        movf    denom+0,W
        btfsc   _Z
        goto    compute_com
ishift_com2
        bcf     _C
        rrf     num+0
        rrf     num+1
        rrf     num+2
        bcf     _C
        rrf     denom+0
        rrf     denom+1
        goto    ishift_com
        endif
compute_com
        if  RESOLUTION == 8
        movf    denom+1,W
        btfss   _Z
        else
        FTSTFSZ16  denom+0
        endif
        goto    rout_div_com
        bsf     _C
        if  FAMILY ==    2
        return
        else
        goto    end_com
        endif
rout_div_com
        if      FAMILY == 2
        if  RESOLUTION == 8
        call    div16_8
        else
        call    div32_16
        endif
        else
        include  "div16_8.asm"
        endif
        bcf     _C
        if  RESOLUTION == 8
        movf    res_div+1,W
        else
        FMOV16      res_div+2,Temp16
        endif
```

```
        if      FAMILY == 2
        return
        else
        end_com
        endif

; The end of module com
```

Mpy 8-8

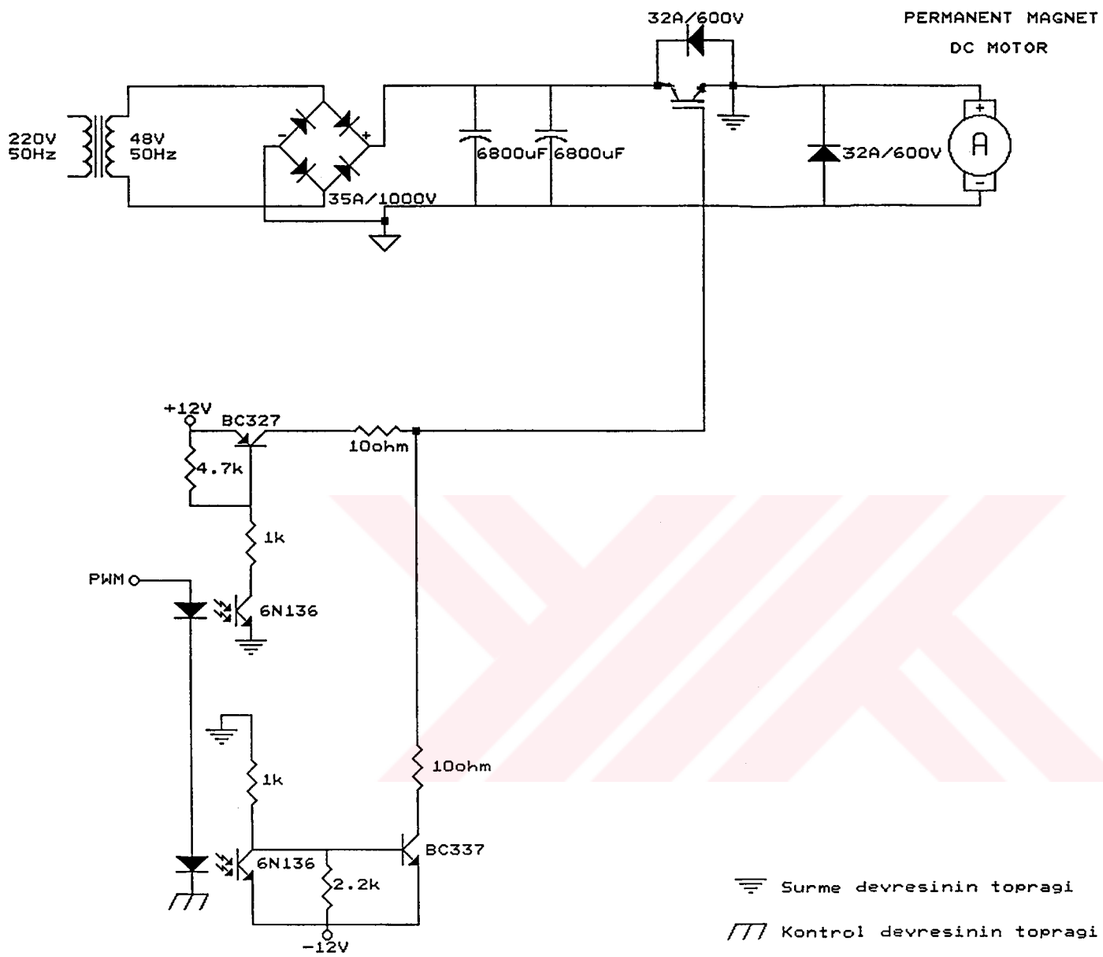
```
;*************************************************************************
*
;*                                                                       *
;* Input:   8 bit unsigned fixed point multiplicand in mulcnd                  *
;*          8 bit unsigned fixed point multiplier   in mulplr            *
;*                                                                       *
;* Output: 16 bit unsigned fixed point result      in res_mpy+0, res_mpy+1 *
;*                                                                       *
;* Result:  res_mpy <-- mulcnd * mulplr                                  *
;*                                                                       *
;*************************************************************************
*


        if      FAMILY < 2
        mpy8_8_
        else
        mpy8_8
        endif

        clrf    FTcount
        bsf     FTcount,3
        clrf    res_mpy+0

if FAMILY < 3
        movf    mulcnd,W
        rrf     mulplr
mpyLoop
        btfsc   _C
        addwf   res_mpy+0
        rrf     res_mpy+0
        rrf     mulplr
        decfsz  FTcount
        goto    mpyLoop
        movf    mulplr,W
        movwf   res_mpy+1
else
        movfp   mulcnd,Wreg
        rrcf    mulplr
mpyLoop
        btfsc   _C
```

```
        addwf res_mpy+0
        rrcf    res_mpy+0
        rrcf    mulplr
        decfsz FTcount
        goto    mpyLoop
        movfp mulplr,Wreg
        movwfres_mpy+1
        endif

        if      FAMILY < 2
        retlw   0
        else
        return
        endif

; The end of module mpy8_8
```

Div 16-8

```
;*************************************************************************
;*                                                                       *
;*   Input:  16 bit unsigned fixed point dividend  in FAARG+0, FAARG+1   *
;*               8 bit unsigned fixed point divisor   in FBARG+0         *
;*                                                                       *
;*   Output: 16 bit unsigned fixed point quotient  in FAARG+0, FAARG+1   *
;*               8 bit unsigned fixed point remainder in FREM +0         *
;*                                                                       *
;*   Result: FAARG, FREM <-- FAARG / FBARG                               *
;*                                                                       *
;*************************************************************************


div16_8
        clrf            FREM
        clrf            FTcount
        bsf             FTcount,4

        if FAMILY < 3

        divLoop
        rlf             FAARG+1
        rlf             FAARG+0
        rlf             FREM
        movf            FBARG+1,W
        btfss           _C
        goto            no_carry
        subwf           FREM
        bsf             _C
        goto            no_sub
         no_carry
        subwf           FREM,W
        btfsc           _C
```

```
        movwf       FREM
no_sub
        decfsz      FTcount
        goto        divLoop
        rlf         FAARG+1
        rlf         FAARG+0
else

divLoop
        rlcf        FAARG+1
        rlcf        FAARG+0
        rlcf        FREM
        movfp       FBARG+1,Wreg
        btfss       _C
        goto        no_carry
        subwf       FREM
        bsf         _C
        goto        no_sub
no_carry
        subwf       FREM,W
        btfsc       _C
        movwf       FREM
no_sub
        decfsz      FTcount
        goto        divLoop
        rlcf        FAARG+1
        rlcf        FAARG+0
        endif

        if FAMILY > 1
        return
        endif

; The end of module div16_8
```

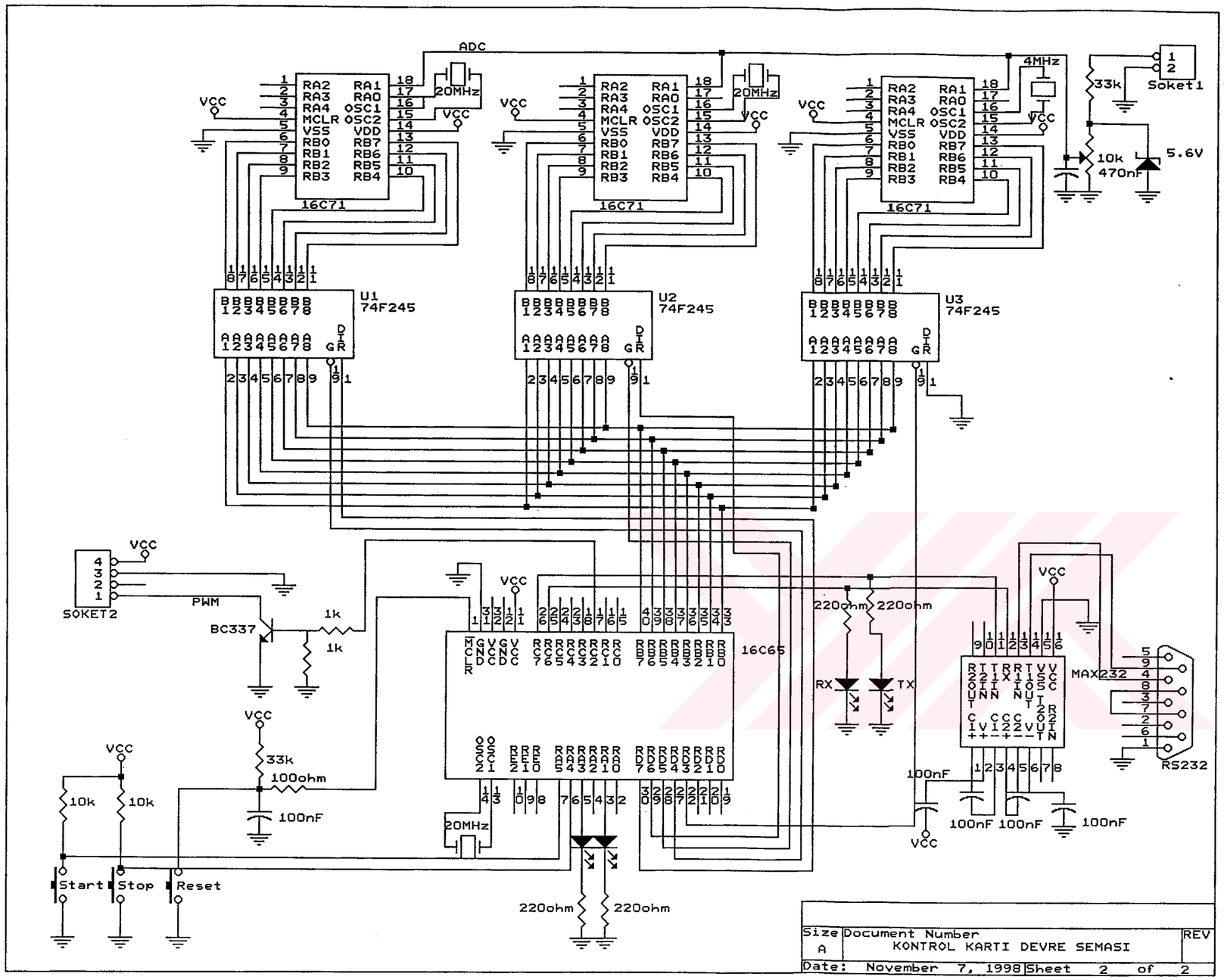


Surme devresinin topragi

Kontrol devresinin topragi

Guc devresinin topragi

Size	Document Number	REV
A	GUC VE SURME DEVRELERI	
Date: December 18, 1998	Sheet 1 of 2	


ADC
20MHz
VCC
16C71
U1
74F245
U2
74F245
U3
74F245
4MHz
33k
Soket1
10k
470nF
5.6V
SOKET2
PWM
BC337
1k
1k
16C65
220ohm
220ohm
RX
TX
MAX232
RS232
100nF
33k
100ohm
10k
10k
Start
Stop
Reset
Size
A
Document Number
KONTROL KARTI DEVRE SEMASI
REV
Date: November 7, 1998
Sheet 2 of 2

ÖZGEÇMİŞ

Doğum tarihi	05.02.1974	
Doğum yeri	Üsküdar	
Lise	1987-1990	İntaş Lisesi
Lisans	1990- 1994	Yıldız Teknik Üniversitesi Elektrik-Elektronik Fakültesi Elektrik Mühendisliği Bölümü
Yüksek Lisans	1994-1996	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektrik Mühendisliği Anabilim Dalı
Doktora	1996-1999	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektrik Mühendisliği Anabilim Dalı
Çalıştığı kurum		
	1995-.....	YTÜ Elektrik Mühendisliği Araştırma Görevlisi