T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK MOTORLARINDA FİZİKSEL BÜYÜKLÜKLERİN TAHMİNİNDE BULANIK MANTIK KULLANIMI

Mehmet KARAKÖSE

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ELAZIĞ
2001

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK MOTORLARINDA FİZİKSEL BÜYÜKLÜKLERİN TAHMİNİNDE BULANIK MANTIK KULLANIMI

Mehmet KARAKÖSE

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Bu Tez …./…./2001 Tarihinde, Aşağıda Belirtilen Jüri Tarafından Oybirliği/ Oyçokluğu ile Başarılı/ Başarısız Olarak Değerlendirilmiştir.

(İmza)	(İmza)	(İmza)
Danışman	Üye	Üye

Bu tezin kabulü, Fen Bilimleri Enstitüsü Kurulu'nun …./…./2001 tarih ve ……sayılı kararıyla onaylanmıştır.

ÖZET

Yüksek Lisans Tezi

ELEKTRİK MOTORLARINDA FİZİKSEL BÜYÜKLÜKLERİN TAHMİNİNDE BULANIK MANTIK KULLANIMI

Mehmet KARAKÖSE

Fırat Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

2001, Sayfa:53

Alternatif akım motorlarına uygulanan vektör kontrolün performansı büyük ölçüde akı vektörünün fazının ve genliğinin doğru olarak tahmin edilmesine bağlıdır. Vektör kontrol için kullanılan gerilim modelinde akı hesabı integrasyon içerdiğinden dolayı problemler oluşmaktadır. Bu çalışmada integratörden kaynaklanan faz ve genlik problemlerini azaltmak için geri besleme yoluna bulanık kontrollü bir düzeltim algoritması geliştirilmiştir. Bu algoritma, değişik güçteki motorlarda farklı yük koşullarında test edilmiş ve geri besleme yolunda PI kontrolör kullanılan bir sistemden daha iyi sonuç verdiği gösterilmiştir.

Bir asenkron motorun stator akısını tahmin etmek için kullanılan bu yöntemin MATLAB-SIMULINK programı kullanılarak benzetimi yapılmış ve TMS320C31 sayısal işaret işlemci kullanılarak deneysel olarak gerçekleştirilmiştir.

ABSTRACT

Master Thesis

THE USE OF FUZZY LOGIC IN THE ESTIMATION OF PHYSICAL PARAMETERS OF ELECTRIC MOTORS

Mehmet KARAKÖSE

Firat University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

2001, Page:53

Performance of the field orientation in induction motors depends on the accurate estimation of the flux vector. The voltage model used for field orientation has in the flux calculation process an open integration problem, generally solved with a feedback loop. In this study, a fuzzy controller is developed for the feedback loop of the integrator. The proposed controller is tested on various motors of different power ratings, and it has been demonstrated that the proposed algorithm provides better than performance the PI controller nominally used in the feedback loop.

The algorithm proposed to estimate the stator flux of an induction motor is simulated by using MATLAB-SIMULINK, and implemented on an experimental system using a TMS320C31 digital signal processor.

TEŞEKKÜR

Bu çalışma boyunca maddi ve manevi hiçbir yardımını esirgemeyen ve daima çalışmaya motive eden başta saygıdeğer hocam Yrd.Doç.Dr. Erhan AKIN 'a, Hayrettin CAN 'a, A.Bedri ÖZER 'e ve emeği geçen tüm hocalarıma teşekkürü bir borç bilirim.

İÇİNDEKİLER

ŞEKİLLER VE TABLOLAR LİSTESİ

SİMGELER

ce : Hatadaki değişim

e : Hata

GE : Oransal bulanık kontrolörde giriş kazancı

GU : Bulanık kontrolörde çıkış kazancı

GCE : Oransal-türevsel bulanık kontrolörde türev giriş kazancı

GCU : Artımsal bulanık kontrolörde çıkış kazancı

GIE : Oransal-artımsal-türevsel bulanık kontrolörde giriş kazancı

$i_{s\alpha}$: Akımın α bileşeni

$i_{s\beta}$: Akımın β bileşeni

J : Atalet momenti

K_P : Kontrolörün oransal katsayısı

K_I : Kontrolörün integral katsayısı

K_1 : Bulanık kontrolör için giriş kazancı

K_2 : Bulanık kontrolör için giriş kazancı

K_3 : Bulanık kontrolör için çıkış kazancı

L_{ls} : Stator kaçak indüktansı

L_{lr} : Rotor kaçak indüktansı

n : Anlık zaman

P : Aktif güç

θ_s : Sabit eksen takımı ile dönen eksen takımı arasındaki konum açısı

R_s : Stator direnci

R_r : Rotor direnci

T_{em} : Elektromagnetik moment

T_{load} : Yük momenti

T_i : İntegral zamanı

T_d : Türev zamanı

u : Kontrolör çıkışı

U_n : Kontrol sinyali

$U_{r\alpha}$: Rotor geriliminin α bileşeni

$U_{r\beta}$: Rotor geriliminin β bileşeni

$U_{s\alpha}$: Stator geriliminin α bileşeni

$U_{s\beta}$: Stator geriliminin β bileşeni

x : Temel değişkenin tanımlı olduğu söylem evreni

ω_r : Rotor açısal frekansı

y : Sistem çıkışı

$\psi_{s\alpha}$: Stator akısının α bileşeni

$\psi_{s\beta}$: Stator akısının β bileşeni

$\psi_{m\alpha}$: Hava aralığı akısının α bileşeni

$\psi_{m\beta}$: Hava aralığı akısının β bileşeni

$\psi_{r\alpha}$: Rotor akısının α bileşeni

$\psi_{r\beta}$: Rotor akısının β bileşeni

δ : İntegratörde geri besleme katsayısı

μ : Üyelik derecesi

$\mu(x)$: Üyelik fonksiyonu

1. GİRİŞ

Vektör kontrollü asenkron motor iyi bir geçici moment cevabına sahiptir. Asenkron motorlar DC motorlara göre maliyet, güvenilirlik ve sağlamlık avantajlarına sahiptir. Modern kontrol sistemlerinde hız duyargasız vektör kontrol çok önemlidir. Vektör kontrolün performansı, asenkron motorun akısının faz ve genlik bilgisi ile doğrudan ilişkilidir. Akıların direkt olarak ölçümü zor olduğundan endüstride akı tahmin tekniklerinin kullanıldığı standart asenkron motorlar tercih edilirler.

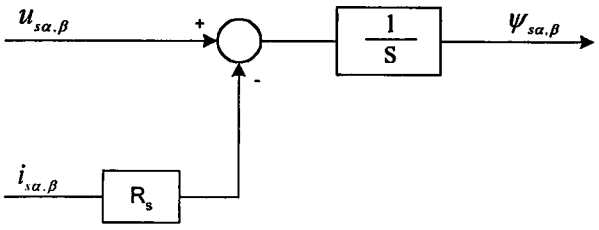
Asenkron motorun stator ve rotor akısını tahmin etmek için genel olarak kullanılan iki model bulunmaktadır. Bunlar akım modeli ve gerilim modelidir. Akım modeli akı hesaplamasında motor hızına ve rotor parametrelerine ihtiyaç duymaktadır. Ancak bu yöntem açık integrasyon içermemektedir. Bu son özellikten dolayı sürekli durumda bile akıyı hesaplamak mümkündür. Diğer taraftan gerilim modeli gerilim ve akım ölçümlerine ve ölçümü daha kolay olan stator parametrelerine ihtiyaç duymaktadır. Ayrıca bu modelde hız ölçümüne gerek yoktur. Duyargasız vektör kontrolde gerilim modeli daha çok tercih edilmektedir. Gerilim modeli kullanılarak stator akısı tahmininde değişik metotlar vardır. Rotor akısı, stator akısı ve motor parametreleri kullanılarak da tahmin edilebilir. Gerilim modeli olarak bilinen stator referans çatıdaki stator akı denklemleri aşağıdaki gibidir.

$$\psi_{s\alpha} = \int (u_{s\alpha} - i_{s\alpha} R_s)\, dt \qquad (1.1a)$$

$$\psi_{s\beta} = \int (u_{s\beta} - i_{s\beta} R_s)\, dt \qquad (1.1b)$$

Bu denklemler kullanılarak $\psi_{s\alpha}$ ve $\psi_{s\beta}$ bileşenleri kolaylıkla hesaplanabilir. Daha sonra bu bileşenlerden faydalanılarak vektör kontrol için gereken dönüştürme açısı θ_s aşağıdaki denklem kullanılarak hesaplanabilir. Bu açı $\alpha\beta$ ile dq arasında dönüştürme için kullanılır.

$$\theta_s = arctg \frac{\psi_{s\beta}}{\psi_{s\alpha}} \qquad (1.2)$$



Şekil 1.1 Gerilim modeli blok diyagramı

Şekil-1 de verilen blok diyagramı denklem 1a ve 1b 'de tanımlanan akı hesaplama işlemini göstermektedir. Şekilden de görüldüğü gibi bu açık integrasyon işlemi geri besleme içermemektedir ve belirsizliklere neden olmaktadır. Böylece sürüklenme olarak adlandırılan durum oluşmaktadır. Bu integrasyon işlemi analog veya digital olarak gerçekleştirilebilir. Digital akı integrasyonundaki hatalar aşağıdaki nedenlerden kaynaklanır.

- İntegrasyon metodu ve integrasyon adım büyüklüğü,
- Akım ve gerilim ölçümlerindeki hatalar ve ofsetler,
- Motor sıcaklığının değişimi ve motora uygulanan akımın frekansından dolayı motor parametrelerindeki değişmeler,
- İşlemcinin sınırlı sayıdaki bit uzunluğu ve komutların çalışma süresi.

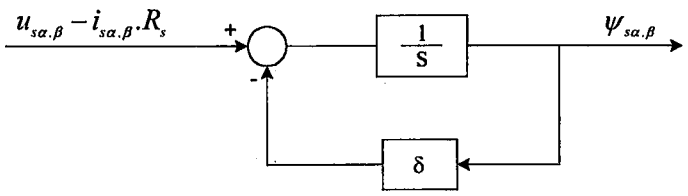
Yukarıdaki tartışmadan da anlaşılacağı üzere gerilim modelinin performansı stator direncine bağımlıdır. Bu stator direnci düşük hızlarda daha fazla önem kazanmaktadır. Çünkü üzerindeki gerilim düşümü daha önemli hale gelmektedir. Bu nedenle sıfır hızda bu algoritma kullanılarak stator akısını tahmin etmek mümkün değildir.

İntegrasyon problemi üzerine literatürde özellikle duyargasız direkt vektör kontrollü sürücülerin açık integrasyon problemiyle ilgili birçok araştırmaya rastlanmaktadır. Örneğin, integrasyon algoritmasındaki problemi çözebilmek için algoritmaya akı genliği geri beslemesi eklenmesi önerilmiştir (Ohtani, 1992). Bu çalışmada vektör kontrollü sürücüde hız sensörü kullanılmadan, hem moment hem de

hız kontrolü geniş bir hız ve yük bandında gerçekleştirilmiştir. Bu kontrol metodu vektör kontrole bağımlı olup, bu metotla rotor akı hızı, moment üreten akım ile stator gerilim ve akımı kullanılarak hesaplanan rotor akısı tarafından kontrol edilmektedir. Sonuç olarak önerilen bu çalışmada rotor-akı tahmincisi geniş bir hız ve yük bandında iyi performans göstermektedir. Diğer bir yaklaşım ise iki faz akım ve geriliminin ölçümüne ve dinamik koşullar altında yapılan kararlılık analizine bağımlıdır (Burgt, 1996). Bu çalışmada geri besleme olarak akı ve akının türevi alınarak bunlar belirli katsayılarla çarpılmış ve böylece ofset ve faz hataları düzeltilmeye çalışılmıştır. Bu yöntem oldukça iyi sonuç vermesine karşın geri besleme katsayılarının belirlenmesi oldukça zor ve karmaşıktır. Patel tarafından yapılan bir çalışmada ise integratör yerine kaskad olarak bağlanmış ve otomatik olarak ayarlanabilen alçak geçiren filtreler kullanılmıştır (Patel, 1997). Akın tarafından önerilen bir çalışmada da integrator çıkışının dc bileşeni elimine edilmiştir fakat bu sadece sürekli durumda çalışan bir algoritmadır (Akın, 1994).

Bu problemi çözmek için yapılmış olan algoritmalar, integratörün kararlılık problemini çözmek için kullanılan δ geri beslemeli integrasyon algoritması ve PI geri beslemeli integrasyon algoritmasıdır.

δ geri beslemeli integratörün blok diyagramı ve transfer fonksiyonu sırasıyla şekil 1.2 ve denklem 1.3 'de verilmiştir.



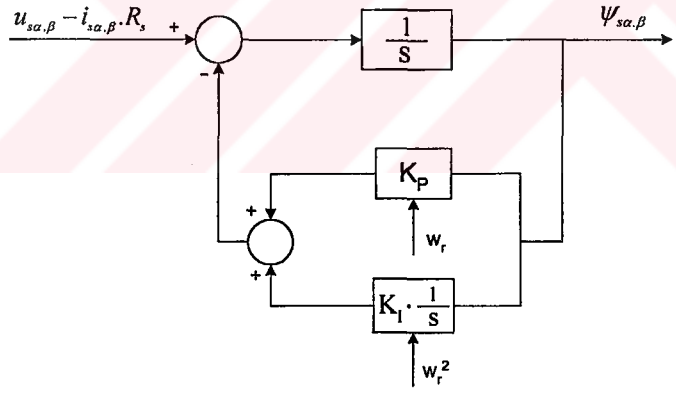
Şekil 1.2 δ geri beslemeli integratör

$$F(s) = \frac{1}{s+\delta} \tag{1.3}$$

Bu integratörde, δ geri besleme yolu integratörün kararlılığı için kullanılmıştır. Denklemden de açıkça görülmektedir ki frekans değeri jw>>δ olduğu zaman integratör fonksiyonu iyi olmaktadır. Hem geçici durumlarda hem de sürekli durumlarda jw değeri δ değeri ile karşılaştırılabilecek kadar küçük olduğu zaman faz ve genlik hatalarına neden olmakta ve bu da performansı etkilemektedir. Şekil 2' de gösterilen δ geri beslemeli integratör ofseti elimine eder. Bununla birlikte integrasyon çıkışı hala faz kayması ve genlik hatası içerir (Akın ve diğ., 1998).

İntegrasyon probleminden kaçınmak için önerilen diğer bir metot ise analog kontrollü PI integratör içermektedir (Drieseitl, 1980). Bu integratörün blok diyagramı şekil 1.3' de, transfer fonksiyonu ise denklem 1.4' de verilmiştir. Burada w_r rotor açısal frekansını göstermektedir.

$$F(s) = \frac{s}{s^2 + K_p.w_r.s + K_i.w_r^2} \tag{1.4}$$



Şekil 1.3 PI kontrollü integratörün blok diyagramı

Açık integrasyon işleminin hatalarını ortadan kaldırmak için kullanılan farklı bir yöntem Şekil 1.3 'de görülmektedir. Burada görüldüğü gibi integratörün geri besleme yolu üzerine bir PI kontrolör konulmuştur. PI kontrolörün integratör çıkışındaki dc bileşeni yok ederek istenmeyen etkilerini ortadan kaldırması ve asıl işaretin aynen kalması amaçlanmıştır. Burada kontrolörün sadece dc bileşeni yok etmesi için uygun P

ve I değerleri seçilmelidir. Fakat bu seçim tüm frekans aralığında doğru sonucu vermelidir. Dolayısıyla buradan P ve I katsayılarının hıza bağımlı olma zorunlulukları ortaya çıkar. Hatta P katsayısı devir sayısı ile lineer olarak, integral katsayısı ise devir sayısının karesi ile orantılı olarak değişmelidir. Şekil 1.3' de görüldüğü gibi integratörün geri besleme yolu üzerindeki PI kontrolör katsayıları hız ω_r ve hızın karesine ω_r^2 bağımlıdır. Bu integrasyon algoritması sayısal olarak gerçekleştirilebilir fakat hız ölçümü gerektirir. Bundan dolayı sistem daha fazla karmaşık hale gelir ve maliyet artar. PI kontrolör katsayılarının (K_P, K_I) seçimi ise diğer bir kritik noktadır.

2. BULANIK MANTIK

Endüstriyel bir sürecin denetimi için tasarım yapılırken her şeyden önce o sürecin bir dinamik modeline gereksinim vardır. Ancak pratikte bu her zaman mümkün değildir. Süreç içindeki olaylar matematiksel modellemeye el verecek ölçüde açıkça bilinmeyebilir veya bir model kurulabilse bile bu modelin parametreleri zamanla büyük değişiklikler gösterebilir. Bazı durumlarda ise doğru bir model kurulabilse bile bunun denetleyici tasarımında kullanılması karmaşık problemlere yol açabilir. Bu gibi sorunlarla karşılaşıldığı zaman genellikle bir uzman kişinin bilgi ve deneyimlerinden yararlanılma yoluna gidilir. Uzman operatör dilsel niteleyiciler (linguistic variables) olarak tanımlanabilecek; uygun, çok uygun değil, yüksek, biraz yüksek, fazla, çok fazla gibi günlük yaşantımızda sıkça kullandığımız kelimeler doğrultusunda esnek bir denetim mekanizması geliştirir. İşte bulanık denetim de bu tür mantıksal ilişkiler üzerine kurulmuştur.

Bulanık küme kuramının (fuzzy sets theory) günümüze kadar geçirdiği önemli aşamalar Tablo 1'de bir kronolojik sıra ile verilmiştir. Kuram ilk kez 1965 yılında University of California, Berkeley öğretim üyelerinden, aslen Azerbeycanlı Prof. Lotfi A. Zadeh tarafında ortaya atılmış ve hızla gelişerek modern denetim alanında birçok bilim adamının ilgisini çeken araştırmaya açık yeni bir dal oluşmuştur. Örneğin Londra Üniversitesi'nden Prof. Mamdani kuramı bir buhar türbininin hızının denetlemesine uygulamayı düşünmüş ve bu amaçla, bir insanın davranışlarını simgeleyen "eğer türbinin hızı çok hızlı artıyorsa ve basınç da çok düşükse, buhar vanasını biraz aç" türünden kurallardan oluşan bir uzman sistem geliştirmiştir. Prof. Mamdani bulanık mantık temelli bu tür bir uzman sistemle türbin hızının ve performansının çok başarılı bir şekilde denetlenebileceğini göstermiştir.

Bulanık mantık kuramının ilk önemli endüstriyel uygulaması çimento sanayisinde olmuştur. Bu sanayide değirmen içerisindeki sıcaklık ve oksijen oranı ürün kalitesi açısından çok önemlidir. Kısıtlı ve hassas olmayan, ısı ve karbonmonoksit oranı gibi bilgilerle iyi bir çalışma düzeni elde edilebilmesi bir sanat olup operatörlerin bu konuda yeterli bir uzmanlık kazanabilmeleri için yıllar geçmesi gerekmektedir. Fakat kişiler ve uzmanlık düzeyleri arasında kaçınılmaz farklılıklar olacağından, üretilen

çimento da vardiyadan vardiyaya değişecek, tutarlı kalitede çimento üretimi çok zor olacaktır. İşte bir Danimarka firması bu nedenlerden dolayı lineer bir model üzerine kurulu geleneksel denetleyici yerine bir bulanık mantık denetleyici (fuzzy logic controller-FLC) kullanmayı düşünmüş ve çok başarılı sonuçlar veren bir uzman sistem geliştirmiştir. Bu veya benzeri sistemler bugün bile, Japonya ve Amerika dahil olmak üzere birçok ülkede, kullanılmaktadır.

Tablo 2.1. Bulanık mantığın kronolojik gelişimi

1965	Bulanık küme kuramı (Prof. Lotfi A. Zadeh, University of California, Berkeley)
1966	Bulanık mantık (Dr. Peter N. Marinos; Bell Laboratuvarı)
1972	Buhar türbini denetiminde bulanık mantık uygulaması (Prof. E. H. Mamadani, Queen Mary College, University of London)
1980	Çimento sanayisinde uygulama (F. L. Smidth, Danimarka)
1987	Sendal metrosunda otomatik fren denetimi (Hitachi)
1988	Hisse senedi portföyün için uzman sistem (Yamaichi Sequrity)
1989	Japonya'da LIFE (Laboratory for International Fuzzy Engineering) laboratuvarının kurulması

Kronolojik sıra içerisinde bundan sonraki en önemli aşama Japonya'da, 1987 yılında görülmüştür. Hitachi firması ilk olarak 1987 yılında Ulaştırma Bakanlığı'na başvurmuş ve Sedai Metro sisteminde çalışan trenlerin otomatik olarak denetimi için bulanık mantık kullanımını önermiştir. Bakanlık öneriye olumlu baktığını belirtmiş, fakat bulanık mantık denetleyicinin kullanılmakta olan sisteme göre belirgin üstünlükleri olacağı konusunda kanıt istemiştir. Hitachi firması 9 yıl içerisinde 300000 benzetim çalışması ve 3000 insansız operasyon gerçekleştirmiş ve sonunda, 1986 yılının sonlarına doğru Ulaştırma Bakanlığı'ndan kullanım izni almıştır. Geliştirilen sistemde, daha önce tren operatörü tarafından bir PID temelli denetleyici aracılığı ile yapılan ve yolcuların sarsıntılı bir yolculuk geçirmelerine neden olabilen hızlanma ve yavaşlama gibi işlemler otomatik olarak yapılmakta ve tren operatörünün yapması gereken işler, kapıları kapatmak ve başlatma düğmesine basmak gibi birkaç işlemle sınırlı

kalmaktadır. Böylece yolcuların, demirlere tutunma gereksinimi duymadan rahat bir yolculuk yapabilmeleri sağlanmış, daha önce kullanılan sisteme göre trenin istenilen konumda durması 3 kat iyileşmiş ve kullanılan enerji %10 azalmıştır. Sağlanan bu başarının Hitachi firmasına getirdiği mükafat, Tokyo metrosunda da böyle bir sistemin kullanılması için alınan kontrat olmuştur.

Yukarıda açıklanan başarılı uygulamadan sonra bulanık denetim konusundaki çalışmalar yeni bir ivme kazanmış ve endüstriyel uygulama alanları hızla artırmıştır. Çalışmaların uluslar arası alanda koordinasyonu amacı ile Japonya'da 1989 yılında, LIFE (Laboratory for International Fuzzy Engineering) adlı bir laboratuar kurulmuştur. Bu laboratuarda yapılan araştırma çalışmalarına, aralarında Hitachi, Toshiba, Omron, Matsushita gibi ünlü Japon firmalarının yanısıra IBM, NCR ve Thomson gibi Japonya dışı firmaların da bulunduğu toplam 51 firma katılmakta olup 6 yıllık bütçesi 70 milyon dolardır.

Bulanık mantık denetleyiciler konusundaki kuramsal çalışmaların hala sürüyor olmasına rağmen artık bu konu endüstride kendisine önemli bir yer edinmiş durumdadır. Uygulama alanları arasında çeşitli beyaz eşya, tren, asansör, trafik kontrolü ve otomotiv sanayii sayılabilir. Bugün Japonya'da bulanık denetim kullanan beyaz eşyalar ve elektronik aletler, örneğin fotoğraf ve çamaşır makinaları, güncel yaşamın birer parçasıdırlar. Tüm dünyada ise bulanık mantık içeren ürünlerin satış hacmi 1990 yılında 1.5 milyara dolara ulaşmıştır. Bu meblağın 2000 yılına kadar 13 milyar dolara çıkması beklenmektedir.

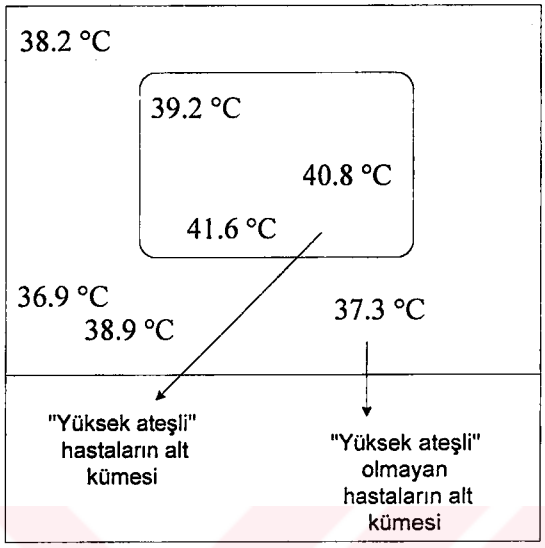
Günümüzde 30'dan fazla ülkede bulanık mantık konusunda araştırmalar yapılmakta olup bunlar arasında ABD, Japonya, Çin ve Batı Avrupa ülkeleri ile hemen arkasında Japonya yer almaktadır. Uygulama açısından ise Japonya belirgin bir şekilde önde gözükmektedir. Bu durum belki de uzak doğu insanının düşünüş şeklinin bulanık mantığa daha uygun oluşundan kaynaklanmaktadır. Üzerinde bulanık sözcüğü yer alan bir fotoğraf makinası, bir batı ülkesinin toplumu tarafından tepki ile karşılanabilir. Japonya'da ise ev kadınları bile bu sözcükle haşır neşir olmuşlar ve bulanık mantık kullanan her türlü ev aletini özellikle arar duruma gelmişlerdir. Bir başka neden de ABD'de yapılan çalışmaların genellikle askeri amaçlara ve uzay uygulamalara yönelik olması ve bu nedenle projelerin ve sonuçlarının herkese açık literatürde yayımlanması olabilir. NASA bünyesinde bulanık denetim konusunda çalışan çok kuvvetli bir grup

vardır. Bu grup uzay mekiği için, pilotların yükünü azaltmak, sistemin güvenilirliğini arttırmak ve yakıt tüketimini azaltmak amacı ile bir bulanık mantık temelli sistem geliştirmiş ve böylece konuşlandırma ve bu durumda tutma sırasında harcanan yakıt 3 misli, yaklaşım sırasında tüketilen yakıt 1.5 misli azaltılmıştır.

2.1. Bulanık Mantığa Genel Bakış

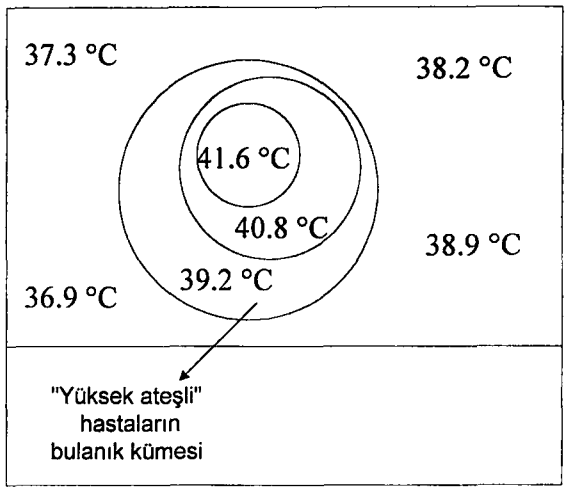
Bulanık mantık, model temelli sistem tasarımını mühendislik deneyimlerine dayandıran bir teknolojidir. Çünkü, bulanık mantık sistem davranışını tanımlamak için dilsel değişkenler kullanır ve bu sayede tam bir matematiksel modellemeye olan ihtiyaç ortadan kalkar. Bulanık mantık verilen bir sistem için tam bir matematiksel modelin kolayca bulunamadığı durumlarda, nonlineerliklerin, zaman kısıtlamalarının ve çok parametrenin olması durumlarında, problem hakkında tasarım bilgisi varsa veya tasarım sırasında elde edilebildiğinde bulanık mantık kullanılması kolaylıklar sağlar. Mühendislik tecrübe ve deneyimlerini, bilinen modelleme anlayışı ile birleştirmenin sonucunda sistem performansının artması, varolan tasarımlara yeni fonksiyonlar kazandırması, geliştirme sürecini ve pazara sunma süresini, ileri geliştirme programları kullanarak kısaltması bulanık mantığın kazandırdığı avantajlardır.

Burada klasik kümeler ile bulanık kümeler arasındaki farka bir örnek ile bakalım. Hastalar için "yüksek ateş" ölçüsü olarak $T \geq 39\,°C$ alalım.



Şekil 2.1 Klasik kümelerin gösterimi

Şekil 2.1 'de görüldüğü gibi yüksek ateş sınırı çok kesin bir ifade ile belirtilmiştir. Buradan anlaşılacağı üzere hastanın sıcaklığı 39°C 'nin çok az üstünde bile olsa bu yüksek ateşli sayılmaktadır. Yani 39.2°C ile 41.6°C 'nin ikisi de yüksek ateş sayılmakta diğer yandan 38.9°C yüksek olmayan bir sıcaklık kabul edilmektedir. Halbuki aynı olayı bulanık küme ile gösterirsek ;



Şekil 2.2 Bulanık kümelerin gösterimi

Şekil 2.2 'ye dikkat edilirse burada yüksek ateşli hastalar farklı kümeler içerisinde gösterilmiştir. Böylece 39.2°C ile 41.6°C ayrı seviyelerde göz önüne alınmıştır.

Şekildeki bulanık kümenin matematiksel tanımı şu şekilde yapılabilir :

⇒ Bulanık küme "Yüksek ateş" YA

⇒ X, tüm sıcaklıların kümesi

⇒ Her $x \in X$ ve de üyelik fonksiyonları $\mu YA(x)$

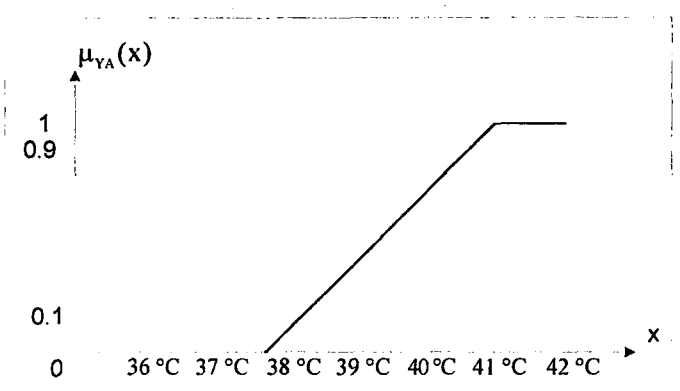
⇒ μ, üyelik derecesi, $\mu \in [0,1]$

Örnek olarak

$$\mu_{YA}(35°C) = 0 \qquad \mu_{YA}(38°C) = 0.1$$
$$\mu_{YA}(41°C) = 0.9$$

$$\mu_{YA}(36°C) = 0 \qquad \mu_{YA}(39°C) = 0.35$$
$$\mu_{YA}(42°C) = 1$$

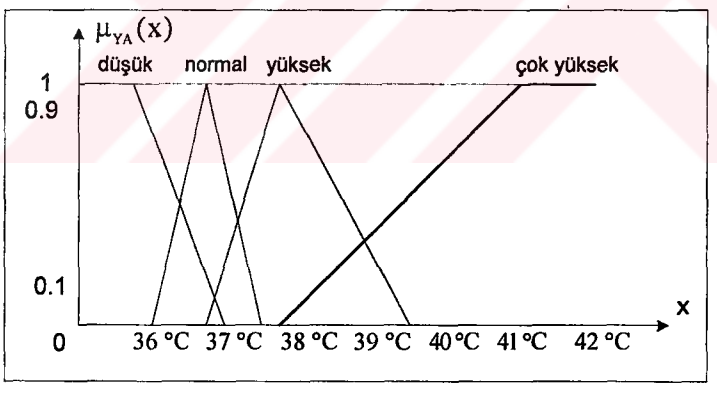
$$\mu_{YA}(37°C) = 0 \qquad \mu_{YA}(40°C) = 0.65$$
$$\mu_{YA}(43°C) = 1$$



Şekil 2.3 Bulanık kümenin sürekli fonksiyon olarak tanımı

Yukarıdaki şekildeki değerlerden yararlanarak bulanık kümeyi sürekli bir fonksiyon şeklinde tanımlamak mümkün olmuştur.

Bu durumda, dilsel değişkenlerin bulanık kümelerle ifadesi aşağıdaki şekildeki gibi yapılabilir.



Şekil 2.4 Dilsel değişkenlerin bulanık kümelerle ifadesi

2.2. Dilsel Değişkenlerin Tanımlanması

Dilsel terimlerin sayısı ve isimlerin belirlenmesi yapılır. Genelde 3, 5 veya 7 terim kullanılabilir. Dilsel değişkenlerin seçiminde şu hususlar göz önünde bulundurulabilir.

a- Kontrol stratejisinde kullanılacak kurallardan bazılarının tespiti,

b- Değişken aralığının dilsel olarak bulunması,

c- Deneyim ; birçok bulanık mantık sisteminin tasarımından sonra seçim kolaylaşır.

Üyelik fonksiyonunun tanımlanması,

a- Terim ismine en uygun temel değişkenin belirlenmesi,

b- Bu değer için $\mu=1$ koyulur,

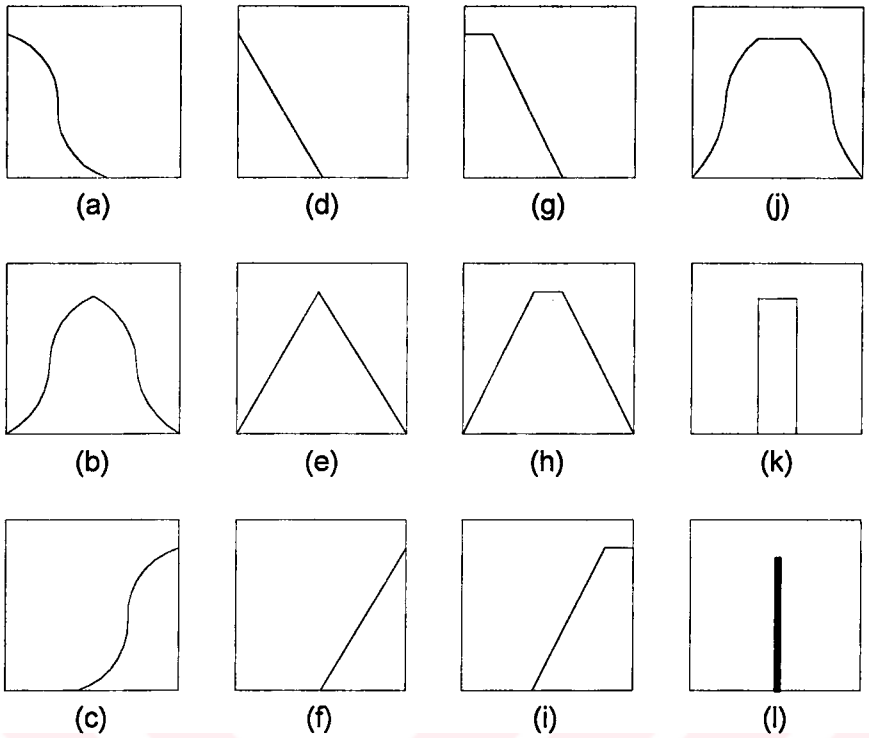
c- Komşu terimler için $\mu=0$ olacak yerden başlanır,

d- Bu noktalar doğrular ile birleştirilir,

e- Her terim için aynı işlemler tekrarlanır.

2.3. Üyelik Fonksiyonları

Üyelik fonksiyonları, insanların gerçek değerleri yorumlama şekillerini sadece yaklaşık olarak ifade edebilen matematiksel fonksiyonlardır. Aşağıdaki şekilde üyelik fonksiyonu olarak kullanılabilecek bazı fonksiyonlar gösterilmiştir. Bunlardan en çok kullanılanı üçgen fonksiyondur. Çünkü bu fonksiyonu matematiksel olarak ifade etmek ve sistemlere uygulamak daha kolaydır. Diğer yandan üyelik fonksiyonları sadece bunlarla sınırlı olmayıp çok çeşitli fonksiyonlar kullanılabilir. Kullanılan bu fonksiyonların şekli sistemin çalışmasını iyi veya kötü yönde etkileyebilecektir.



Şekil 2.5 Üyelik fonksiyonları **(a)** s-fonksiyonu **(b)** π-fonksiyonu **(c)** z-fonksiyonu **(d-f)** üçgen fonksiyonlar **(g-i)** trapezoidal fonksiyonlar **(j)** düz π-fonksiyonu **(k)** dikdörtgen **(l)** tek fonksiyon

Gerçekte, çalışmalar üyelik fonksiyonlarının şu özelliklere sahip olmaları gerektiğini ortaya çıkarmıştır.

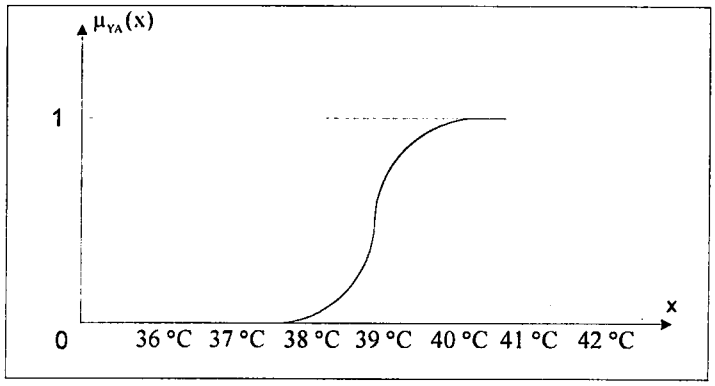
1- $\mu(x)$ X'de süreklidir, yani temel değişkenlerdeki küçük bir değişim, çıkışın bir adım fonksiyonu şeklinde değerlendirilmesine neden olmamalıdır.
2- $d\mu(x)/dx$ X'de süreklidir, aynı kural bu sefer değerlendirilme oranı için mümkündür.
3- $d^2\mu(x)/dx^2$ X'de süreklidir.
4- $\mu(x)$: min $\mu(\max_x(d^2\mu(x)/dx^2))$ eğim değişimi minimum olmalıdır.

μ : üyelik derecesi

$\mu(x)$: üyelik fonksiyonu

x : temel değişkenin tanımlı olduğu söylem evreni

matematiksel olarak, bu dört özelliği aynı anda sağlayabilen fonksiyon s-fonksiyonu (cubic spline) 'dur.



Şekil 2.6 s-fonksiyonu

Uygulamalar s-tipindeki üyelik fonksiyonlarının karar verme ve data analiz sistemlerinde iyi sonuçlar verdiklerini göstermiştir. Ancak kontrol uygulamalarında standart üyelik fonksiyonları yeterli olmaktadır.

2.4. Giriş Değişkenlerinin Bulanıklaştırılması

Giriş değişkenlerinin bulanıklaştırılması için üyelik fonksiyonları kullanılır. Giriş değerlerinin seçilen üyelik fonksiyonlarındaki karşılığı bize bulanık değeri verecektir. Örneğin;

Eğer hastanın ateşi 38°C ise şekil 2.4 'den bakılırsa,

çok yüksek 0.1
yüksek 0.9

olacaktır.

2.5. Kural Kümesinin Tanımı

Bulanık mantık sistemlerinde, kontrol stratejisi "EĞER... O HALDE" kurallarıyla tanımlanır.

EĞER <ön şart> O HALDE <sonuç>

<ön şart> belirli bir durumu tanımlar, <sonuç> ise sistemin bu durum için vermesi gereken tepkiyi kapsar. <ön şart > birden fazla şartı içinde tutabilir. Bu şartlar ve, veya deyimleri ile birleştirilir.

2.6. Bulanık Çıkarım

Bulanık çıkarım iki adımda yapılır :

1- Toplama : Toplama adımında giriş değişkenlerinin bulanıklaştırılmış değerleri ön şart kuralı altında işlenir. Bu sonuç her kuralın o andaki durum için uygulanabilme derecesini belirtir. Bu işlemin "Boole" cebrindeki karşılığı "VE" işlemidir.

2- Derleme : Bu adımda o anki duruma göre yapılacak işlev belirlenir. İşlev, kuralların <sonuç> 'ları olarak tanımlıdır ve kuralı o anki duruma uygulanabilirliği ne kadar fazla ise sonuç da o kadar fazla etkilenecektir.

2.7. Durulama

Bulanık çıkarımın sonucu kural sonuçlarının doğruluk dereceleridir. Bulanık mantık sisteminin çıkışının bir gerçek değer olması gerektiğinden durulama (defuzzification) gereklidir. Böyle bir dönüşüm dilsel terimlerin temel değişken ile aralarındaki ilişkiyi göz önüne almalıdır.

Tüm kuralların sonuçları, karşılıklı terimlerin üyelik fonksiyonlarıyla birebir eşlenirler. Sonuçta elde edilen üyelik fonksiyonları tek bir üyelik fonksiyonlarıyla birebir eşlenirler. Elde edilen tek üyelik fonksiyonu, istenen çıkışı tanımlayan bulanık bir kümedir. Ancak, çıkışta tek bir değer istendiğinden, bu üyelik fonksiyonu tek bir değere

sıkıştırılmalıdır. Durulama işlemi, ağırlık merkezi, maksimumların ortalaması, soldakilerin maksimumları ve sağdakilerin maksimumları kullanılarak yapılabilir.

Sonuç olarak ;

- Bulanık mantık, lineer kontrol modelinin bir üst kümesi gibidir. Bu yüzden, bulanık mantık geleneksel kontrolün bir genelleştirilmiş halidir.
- Bulanık mantığın gücü basit şeyleri basit olarak tutmaktır; zaten birçok uygulama karmaşık model temelli çözüm gerektirmemektedir. Bulanık mantık çözümü, daha az bir çaba ile daha iyi iş yapabilir.
- Bulanık mantık, mühendislik deneyimini etkin bir çözüme aktarmanın en kısa yoludur.

Bulanık mantık kullanarak fiziksel sistemlerin kontrolünü yapmak istersek önümüze dört seçenek çıkar. Bunlar ;

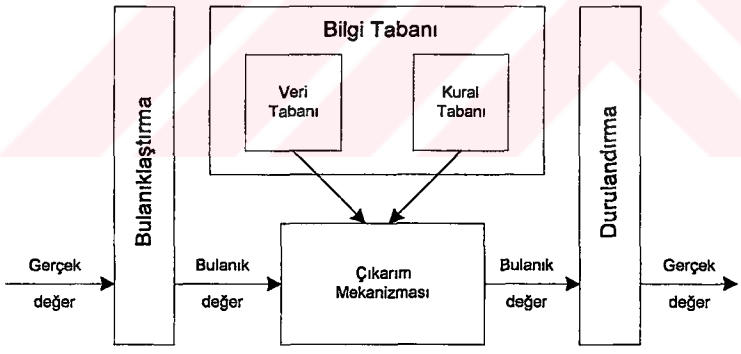
1- Bir mikrokontrolör ile bir çıkarım işlemcisini kaskad bağlayıp beraber çalıştırmak,
2- DSP ve yazılım destekli kontrolör kullanmak,
3- Bilgisayar temelli uygulamalarda ise kural tabanı, veri tabanı, bulandırıcı, çıkarım motoru ve durulayıcı olarak yazılım kullanmak ve paralel iletişimle kontrol sistemi tasarlamak,
4- İçinde RAM, EPROM, I/O birimlerinin yanı sıra bulandırıcı, çıkarım motoru durulayıcı bölümleri de bulunduran tek entegre şeklinde bulanık işlemciler kullanarak fiziksel sistemlerin pratikte kontrolünü sağlamak mümkündür.

3. BULANIK DENETLEYİCİLER

Bulanık denetleyiciler, klasik denetleyicilere göre çok daha yeni olmasına karşın giderek artan sayıda uygulamada kullanılmaktadır. Bulanık denetleyiciler klasik denetleyicilerden esinlenerek bulanık PD, bulanık PI, bulanık PID ve karma denetleyiciler şeklinde farklı yapılarda tasarlanmaktadır (Hu ve diğ., 1999).

3.1. Bulanık Denetleyici Yapısı

Bulanık denetleyici ve klasik denetleyici tasarlanırken sistemin kararlılığı, performansı gibi aynı kontrol problemleri çözülmeye çalışılır. Burada temelde şu farklılık vardır. Klasik denetleyiciler sürecin matematiksel modeline göre tasarlanır. Bulanık denetleyiciler ise uzman bilgisinden yararlanarak elde edilen EĞER-İSE kuralları sentez edilerek tasarlanır.



Şekil 3.1. Bulanık kontrol blok diyagramı

a- Bulandırıcı : Bu bölüm giriş değişkenlerini (gerçek sayılan) ölçer, onlar üzerinde bir ölçek değişikliği yapar ve bulanık kümelere dönüştürür. Yani onlara birer etiket vererek dilsel nicelik kazandırır.

b- Çıkarım Motoru : Burada kurallar üzerinde bulanık mantık yürütülür yani insan beyninin düşünüş şeklinin bir benzetimi yapılmaya çalışılır.

c- Veri Tabanı : Çıkarım motoru, kural tabanında kullanılan bulanık kümelerini bu bölümden alır.

d- Kural Tabanı : Kontrol amaçlarına uygun dilsel denetim kuralları burada bulunur ve çıkarım motoruna buradan verilir.

e- Durulayıcı : Çıkarım motorunun bulanık küme çıkışları üzerinde gerekli ölçek değişikliklerini yapar ve bunları gerçek sayılara dönüştürür.

Bir bulanık denetleyicinin gerçekleşmesinde, denetlenecek sistemin bir matematiksel modelinden daha çok o sistemi çalıştıran operatörün sistem davranışı konusunda sahip olduğu bilgiler daha önemlidir. Tasarım sırasında genellikle bu tür bilgilerden yararlanılır. Böyle bir yaklaşım, uzun yıllar boyunca kazanılan deneyimlerin denetleyici içerisine, yorumlanmış halde kolaylıkla yerleştirilmesine olanak sağlar. Bu yararın yanında getirdiği sakınca denetleyici tasarımında belirli bir otomasyon elde edilememesidir. Buna rağmen, bulanık denetleyicinin en önemli kısmını oluşturan kural tabanının oluşturulması için kullanılabilecek çeşitli yaklaşımlar şunlardır:

- Bir uzmanın bilgi ve/veya deneyimlerinden yararlanmak
- Sürecin bir bulanık modelinin kullanılması
- Operatörün süreç üzerinde yaptığı işlemleri kullanmak
- Öğrenen algoritmaları uygulamak

Bulanık mantık tabanlı kontrolörler geleneksel kontrolörlere göre birçok avantajlara sahip olmasına rağmen, aşağıdaki gibi bazı sınırlamalara ve dezavantajlara da sahiptir.

1- Değişkenlerin fuzzy kümelerini, üyelik fonksiyonlarının şekil ve eğimlerinin ve kural tabanının seçimi için tanımlı bir kriter yoktur.
2- Fuzzy kontrolörler için sistematik bir dizayn prosedürü yoktur.
3- Sistem analizi zordur.
4- Bir fuzzy lojik PI kontrolör gerçekleştirmek için kontrol yazılımı digital PI kontrolöre göre daha geniştir.
5- Bulanık kontrolde kullanılan kurallar deneyime çok bağlıdır.

6- Üyelik işlevlerinin seçiminde belirli bir yöntem yoktur, en uygun işlev deneme ile bulunur. Bu da oldukça uzun bir zaman bulunabilir.
7- Denetlenen sistemin bir kararlılık analizi yapılamaz, sistemin nasıl cevap vereceği önceden kestirilemez. Yapılacak tek şey simülasyondur.

Bir bulanık kontrolörü dizayn etmekte takip edilmesi gereken aşamalar ise şöyle sıralanabilir:
Analizi yaparken bunu iki bölüme ayırmak daha yararlı olmaktadır.

1- Statik özellikler :
   a- Kuralların tamam olması
   b- Aralarındaki bağlantıların tam olması
   c- Kararlılıkları
   d- Duyarsızlıkları
2- Dinamik özellikler :
   a- Adaptif yapılar

Bulanık kontrolörler hastalıklı tanımlanmış yani matematiksel modeli tam olarak elde edilemeyen işlemlere çok başarılı bir şekilde uygulanabilir. Bulanık kontrolörler ayarlanan değere çabuk ve az aşımlı bir şekilde ulaşılması gerektiğinde kullanılır. Bu cevabı PID kontrolörler ile elde etmek başka noktalardan feragat etmek suretiyle mümkün olabilir. Ancak, sistem ayarlanan değere yakın iken PID, uzak iken bulanık kontrolör daha iyi cevap verir. Çünkü, bulanık kontrolör bir PD gibi görev yaparken, PID 'nin I terimi sürekli hal hatasını sıfırlar. Bu yüzden karma yapıdaki bulanık PID kontrolörler üretilmektedir. Ancak görev değişiminin yapılacağı noktanın iyi belirlenmesi gerekmektedir. Bunu yapmanın bir yolu kontrolöre ek bir giriş olarak hata toplamlarını girmektir. Bulanık kontrolörün birçok parametresinin optimize edilmesi gereklidir. Bunlar ;

- Giriş ve çıkış değişkenlerinin ölçeklenme faktörleri
- Kontrol kurallarının belirlenmesi
- Değişkenlerin ölçeklenme faktörleri

- Kontrol kurallarının belirlenmesi
- Değişkenlerin üyelik fonksiyonları

Bulanık denetleyici tasarlarken uygulanan yöntemler temelde iki kategoriye ayrılır. İlk yöntem deneme yanılma yöntemi, ikincisi ise teorik yöntemlerdir. Deneme yanılma ile veya eldeki uzman bilgilerini kullanarak sistem hakkında çeşitli EĞER-İSE kuralları oluşturulur ve sonra gerçek sistem test edilir. İstenen performans sağlanmıyorsa kurallar üzerinde değişikler yapılır, ince ayar kuralları oluşturulur ve sistem istenen performansı sağlayıncaya kadar test edilir. Teorik yaklaşımda ise bulanık kontrol yapısı ve parametreleri istenen performansı garanti edecek şekilde tasarlanır. Pratikte en iyi performansı sağlayan kontrolü tasarlamak için iki yöntemi beraber kullanmak mümkündür (Passino ve Yurkovich, 1998).

3.2. Bulanık Kontrolörlerin Ayarlanması

Bir kontrol problemi bir ayar noktası etrafında çıkış ayarlaması gerektiriyorsa, genelde sisteme giriş olarak hata düşünülür. Hatta hatanın integrali veya hatanın türevi de iyi bir giriş olarak göz önüne alınabilir. Bulanıklaştırılmış PID kontrolörde, yükselme zamanı, overshoot ve yerleşme zamanı üzerinde her bir kazanç faktörünün etkisini anlatmak zordur.

PID kontrolörler değişik yollarla ayarlanabilir. Bunlarda bazıları hand-tunning, Ziegler-Nichols tunning, analitik metotlar, optimizasyon yoluyla, kutup yerleştirme veya auto-tunningdir (Smith, 1979, Astrom ve Hagglud, 1995). Ayrıca fuzzy kontrolörler belirli koşullar altında PID kontrolörler ile benzerlikler gösterir (Siler ve Ying, 1989, Mizomoto, 1992, Qiao ve Mizomoto, 1996, Tso ve Fung, 1997). Fakat tabi ki PID tipi fuzzy kontrolörler ile PID ayarlama metotları arasında ayrılıklar vardır.

Bilinen PID dizayn teknikleri kullanılarak fuzzy kontrolörler gerçekleştirilebilir. Bunlar şöyle sıralanabilir.

1- Bir PID kontrolör ayarı,
2- Bunun bir eşdeğer lineer fuzzy kontrolör ile yer değiştirilmesi,
3- Nonlineer fuzzy kontrolör yapmak,
4- Bu kontrolörü iyi ayarlamak.

3.3. PID Kontrolör Ayarlaması

Dizayn stratejisinde ilk adım bir PID kontrolör ayarlamak ve yerleştirmektir. İdeal sürekli bir PID kontrolör,

$$u = K_P\left(e + \frac{1}{T_i}\int_0 e * d\tau + T_d\frac{de}{dt}\right) \tag{3.1}$$

burada u kontrolör çıkışı, K_P oransal kazanç sabiti, T_i integral zamanı, T_d türev zamanı ve e referans ve sistem çıkışı y arasındaki farktır $(e = \mathrm{Re}\,f - y)$. Biz digital kontrol ile ilgili olduğumuzdan dolayı yukarıdaki denklemin integral kısmını toplam, türev kısmını ise fark denklemleri ile ifade edersek karşımıza şu ifade çıkacaktır.

$$u_n = K_P\left(e_n + \frac{1}{T_i}\sum_{j=1}^{n} e_j + T_d\frac{e_n - e_{n-1}}{T_s}\right) \tag{3.2}$$

n indeksi anlık zamanı gösterir. Buradaki K_P, T_i ve T_d parametreleri ayarlama yapmak için kullanılabilir.Farklı ayarlama ifadeleri statik hususlar ile gösterilebilir (Astrom ve Hagglund, 1995). Örneğin sadece oransal kontrol için T_d=0 ve $1/T_i$=0 olursa,

$$u_n = K_P\, e_n \tag{3.3}$$

3.4. Lineer Bulanık Kontrolörler

Dizayn prosedüründeki ikinci adım PID kontrol kurallarını lineer bir fuzzy kontrolör ile değiştirmektir. Bir fuzzy kontrolörde nonlineerliğin üç kaynağı vardır.

*Kural tabanı*ndaki kurallar genellikle nonlineer bir kontrol startejisi olarak ifade edilirler. *Çıkarım makinası* ve, veya ilişkilerini min, max gibi gerçekleştiriyorsa bunlar nonlineerdir. *Durulama* için kullanılan çoğu metot nonlineerdir.

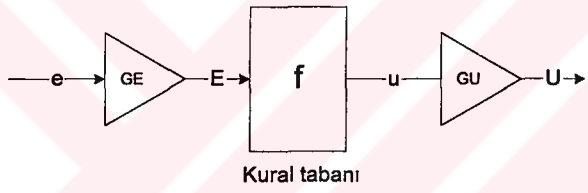
Lineer bir giriş-çıkış eşlemesi ile bir kural tablosunun yapılması mümkündür (Jantzen, 1998).

3.5. PID 'den Fuzzy 'e Kazançların Dönüşümü

Dizayn prosedüründeki üçüncü adım PID kazançlarını lineer fuzzy kontrolöre dönüştürmektir. Bu dönüştürme işlemi her bir kontrolör için aşağıdaki gibi incelenebilir.

3.5.1. P Kontrolör

Aşağıdaki şekilde gösterildiği gibi bir fuzzy oransal kontrolöre giriş *error* ve çıkış kontrol sinyali cf 'dir. Bu en basit fuzzy kontrolör yapısıdır. Bilinen keskin P kontrol ile fuzzy P kontrol karşılaştırıldığında biri diğerine göre GE ve GU olmak üzere iki kazanca sahiptir. Kazançların kullanılmasının ana nedeni cevabı ayarlamaktır fakat iki kazanç olduğundan dolayı onlar giriş sinyalini ölçeklendirmek içinde kullanılabilir.



Şekil 3.2. Fuzzy P kontrolörün blok diyagramı

Kontrolör çıkışı U_n kontrol sinyalidir ki e_n 'nin nonlineer bir fonksiyonudur.

$$U_n = f(GE * e_n) * GU \tag{3.4}$$

f fonksiyonu fuzzy kontrolörün fuzzy giriş-çıkış eşlemesidir. $f(GE * e_n) = GE * e_n$ lineer yaklaşımı kullanılarak,

$$U_n = GE * e_n * GU = GE * GU * e_n \tag{3.5}$$

Kazanç faktörlerinin çarpımı bilinen oransal kontrolörün ifadesi $u_n = K_P\, e_n$ ile karşılaştırılırsa oransal kazancın eşdeğeri,

$$GE * GU = K_P \tag{3.6}$$

olarak bulunur. Yaklaşımın doğruluğu çoğunlukla üyelik fonksiyonları ve kurallara bağımlıdır. En iyi yaklaşım, eğer her iki giriş ve çıkış tarafını aynı seçersek elde ederiz. Örneğin [-100, 100]. Bu durumda kural tabanı,

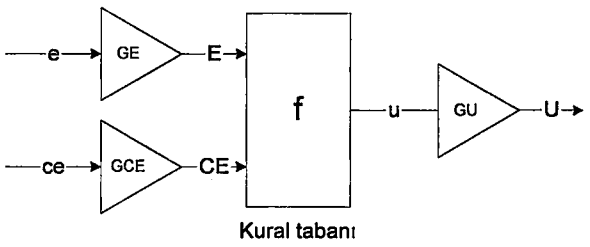
1- If E is Pos then u is 100
2- If E is Neg then u is –100

3.5.2. PD Kontrolör

Türevsel etki hatayı önceden belirlemeye yardımcı olur ve oransal-türevsel (PD) kontrolör kapalı çevrim kararlılığı geliştirmek için türevsel etkiyi kullanır. PD kontrolörün temel yapısı,

$$u_n = K_P\left(e_n + T_d \frac{e_n - e_{n-1}}{T_s}\right) \tag{3.7}$$

Aşağıdaki şekilde gösterildiği gibi fuzzy PD kontrolörün girişi hata ve hatanın türevidir. Fuzzy kontrolde genellikle ikinci terim hatadaki değişim olarak adlandırılır.



Şekil 3.3. Fuzzy PD kontrolörün blok diyagramı

$$ce_n = \frac{e_n - e_{n-1}}{T_s} \tag{3.8}$$

Bu geriye yönelik fark kullanılarak diferansiyel bölüme ayrık bir yaklaşımdır.

Kontrolör çıkışı hata ve hatadaki değişimin nonlineer bir fonksiyonudur.

$$U_n = f(GE * e_n, GCE * ce_n) * GU \tag{3.9}$$

f fonksiyonu fuzzy kontrolörün giriş-çıkış eşlemesidir. $GE * e_n + GCE * ce_n$ lineer yaklaşımı kullanılarak,

$$U_n = f(GE * e_n + GCE * ce_n) * GU \tag{3.10}$$

$$= GE * GU * (e_n + \frac{GCE}{GE} ce_n \tag{3.11}$$

Yukarıdaki denklemle daha önce yazılan PD kontrolörün ifadesi karşılaştırılırsa,

$$GE * GU = K_p \tag{3.12}$$

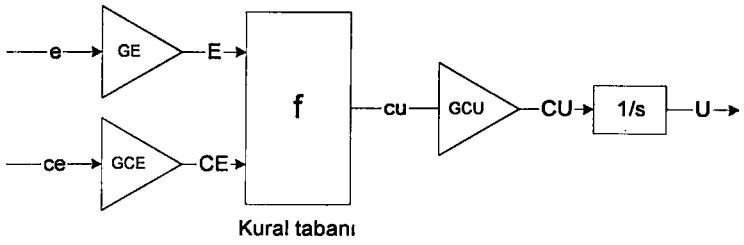
$$\frac{GCE}{GE} = T_d \tag{3.13}$$

olduğu görülebilir.

3.5.3. PI Kontrolör

Eğer sürekli durumda devamlı bir hata varsa integral etkisi zorunludur. Eğer küçük bir pozitif hata varsa integral etkisi artacaktır, hatanın küçük olması problem değildir. Eğer hata negatif ise integral etkisi azalacaktır. İntegral etkisine sahip bir kontrolör daima sürekli durumda sıfıra gidecektir.

Aşağıdaki şekilde görülen fuzzy PI kontrolör fuzzy PD kontrolör gibi aynı konfigürasyonlara sahiptir.



Şekil 3.4. Fuzzy PI kontrolörün blok diyagramı

Kontrol sinyali U_n önceki bütün artımların toplamıdır.

$$U_n = \sum_i (cu_i * GCU * T_s) \tag{3.14}$$

Bu kontrolöre lineer yaklaşım,

$$\begin{aligned} U_n &= \sum_{i=1}^{n} (E_i + CE_i) * GCU * T_s \\ &= GCU * \sum_{i=1}^{n} \left[GE * e_i + GCE * \frac{e_i - e_{i-1}}{T_s} \right] * T_s \\ &= GCU * \left[GE * \sum_{i=1}^{n} e_i T_s + GCE * \sum_{i=1}^{n} (e_i - e_{i-1}) \right] \\ &= GCE * GCU * \left[\frac{GE}{GCE} \sum_{i=1}^{n} e_i T_s + e_n \right] \end{aligned} \tag{3.15}$$

Bilinen PI kontrolörün ifadesi ile yukarıdaki denklem karşılaştırıldığında,

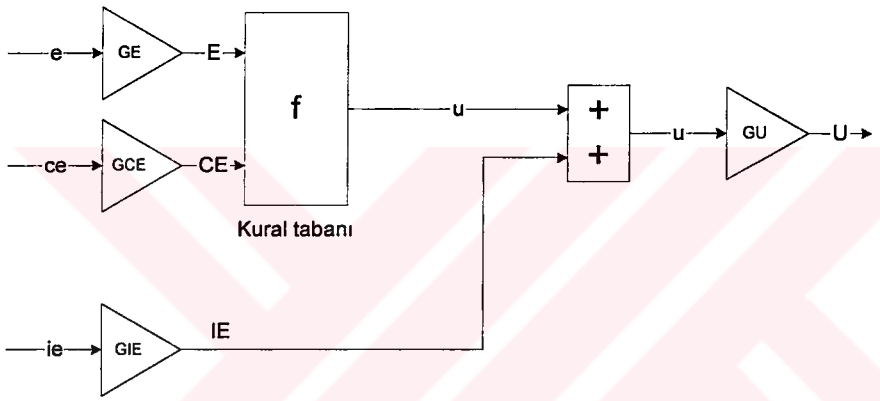
$$\begin{aligned} GCE * GCU &= K_P \\ \frac{GE}{GCE} &= \frac{1}{T_i} \end{aligned} \tag{3.16}$$

olduğu görülebilir.

3.5.4. PID Kontrolör

Fuzzy PID kontrolör hata, hatanın integrali ve hatanın türevi olmak üzere üç girişe sahiptir. Üç girişli bir kural tabanı oldukça büyük olacaktır. Bundan dolayı genellikle fuzzy PD+I olarak integral etkisi ayrı değerlendirilir. Bu aşağıdaki şekilde gösterilmiştir. İntegral hatası şöyle hesaplanabilir.

$$ie_n = \sum_i (e_i * T_s) \tag{3.17}$$



Şekil 3.5. Fuzzy PID kontrolörün blok diyagramı

Kontrolör üç girişli bir fonksiyona sahiptir.

$$U_n = \left[f(GE * e_n, GCE * ce_n) + GIE * ie_n \right] * GU \tag{3.18}$$

Bunun lineer yaklaşımı ise,

$$\begin{aligned} U_n &= \left[GE * e_n + GCE * ce_n + GIE * ie_n \right] * GU \\ &= GE * GU * \left[e_n + \frac{GCE}{GE} * ce_n + \frac{GIE}{GE} ie_n \right] \end{aligned} \tag{3.19}$$

Burada GE 'nin sıfır olmadığı farz edilmelidir. Bu denklem ile bilinen PID kontrolörün fonksiyonu karşılaştırıldığında aradaki ilişki şöyle çıkar.

$$GE * GU = K_p$$
$$\frac{GCE}{GE} = T_d \qquad (3.20)$$
$$\frac{GIE}{GE} = \frac{1}{T_i}$$

Bu kontrolör PID kontrolörün bütün faydalarına sahiptir.

Tablo 3.1 Fuzzy kontrolörlerin karşılaştırılması

Kontrolörler	Avantajları	Dezavantajları
Fuzzy P	Basit	Çok basit olması
Fuzzy PD	Aşmasız.	Gürültüye duyarlı
Fuzzy PI	Sürekli durum hatalarını kaldırır, düzgün kontrol sinyali üretir.	Yavaş
Fuzzy PID	Bütün kontrolörleri içerir.	İntegratördeki hatalara duyarlı

Yukarıdaki tabloda dört fuzzy kontrolörün avantaj ve dezavantajları listelenmiştir. Fuzzy P kontrolör durum uzay modellerinde veya pratik için kullanılabilir. Yerleşme zamanı ve overshoot azaltmayı geliştirmek için fuzzy PD kontrolör seçilebilir. Eğer sürekli durum hatası problemi varsa o zaman da fuzzy PI kontrolör veya fuzzy PID kontrolör tercih edilmelidir. Aşağıdaki tabloda ise PID kazançları ile fuzzy kazançları arasındaki ilişki özetlenmiştir.

Tablo 3.2 Fuzzy ve PID kazançları arasındaki ilişki

Kontrolör	K_P	$1/T_i$	T_d
Fuzzy P	*GE*GU*		
Fuzzy PI	*GCE*GCU*	*GE/GCE*	*GCE/GE*
Fuzzy PD	*GE*GU*		*GCE/GE*
Fuzzy PID	*GE*GU*	*GIE/GE*	

3.6. Fuzzy Kontrolörü Nonlineer Yapmak

Dizayn prosedüründeki dördüncü adım lineer fuzzy kontrolörü nonlineer yapmaktır. Bu pratikte fuzzy kümelerin etiketlerini gösteren Pos, Neg ve Zero gibi terimlerden kural tabanının oluşturulması ile yapılır. Örneğin üç terimden oluşan ki girişli bir sistem için 3*3=9 kural tanımlamak mümkündür. Kümelerin şekilleri ve kuralların seçimi kapalı çevrim sistemin dinamiklerini ve kontrol stratejisini etkileyecektir.

3.7. Nonlineer Fuzzy Kontrolörü İyi Ayarlamak

Dizayn prosedüründeki beşinci adım ise kazançların iyi ayarlanmasıdır. İyi ayarlama fazındaki kazançların seçimi genel olarak sezgi ve deneyimlerin sonucudur.

4. STATOR AKI TAHMİNİNİN GERÇEKLEŞTİRİLMESİ

Modern sürücü sistemleri büyük oranda akı veya moment tahmin yöntemleri kullanmaktadır. Özellikle akı tahmini alternatif akım motorlarında vektör kontrol için önemlidir. Ayrıca dolaylı olarak bu akı moment ve hız büyüklüklerinin hesaplanmasında da kullanılır. Alternatif akım makinalarında akı bilgisi ya doğrudan ölçme ya da diğer elektriksel büyüklüklerden hesaplama yoluyla elde edilebilir. Ancak çoğu endüstri uygulamalarında sensör kullanılması istenmemektedir. Bu model üzerinden akı tahminini ön plana çıkarmaktadır.

4.1. Geliştirilen Yöntemin Bilgisayar Benzetimi

Duran referans çatıdaki bir üç fazlı asenkron motorun bilgisayar benzetimini yapmak için aşağıdaki denklemleri kullanabiliriz. Üç fazlı sincap kafesli asenkron motorların faz değişkenleri dönüşümler kullanılarak iki faza dönüştürülür. Burada inverter anahtarlama elemanları ideal kabul edilmektedir. Asenkron motorun gerilim denklemleri s türev operatörü olmak üzere aşağıdaki gibidir.

$$s\psi_{s\alpha} = u_{s\alpha} + \frac{R_s}{L_{ls}}(\psi_{m\alpha} - \psi_{s\alpha}) \tag{4.1}$$

$$s\psi_{s\beta} = u_{s\beta} + \frac{R_s}{L_{ls}}(\psi_{m\beta} - \psi_{s\beta}) \tag{4.2}$$

$$s\psi_{r\alpha} = u_{r\alpha} + \omega_r\psi_{r\beta} + \frac{R_r}{L_{lr}}(\psi_{m\alpha} - \psi_{r\alpha}) \tag{4.3}$$

$$s\psi_{r\beta} = u_{r\beta} + \omega_r\psi_{r\alpha} + \frac{R_r}{L_{lr}}(\psi_{m\beta} - \psi_{r\beta}) \tag{4.4}$$

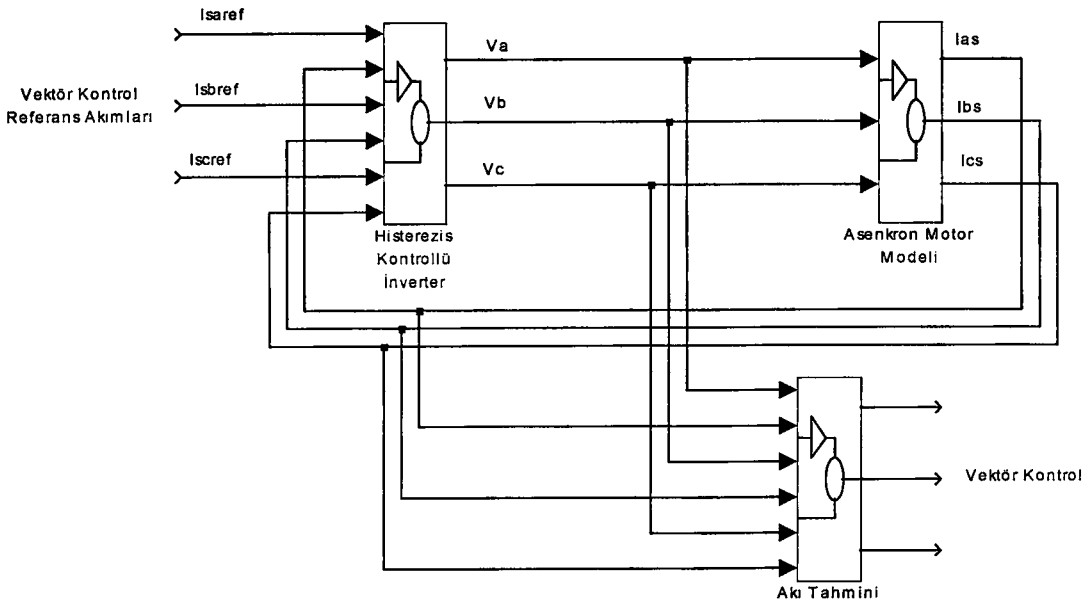
elektromagnetik moment,

$$T_{em} = \frac{3}{2}\frac{P}{2}(\psi_{s\alpha}i_{s\beta} - \psi_{s\beta}i_{s\alpha}) \tag{4.5}$$

ve hareket denklemi sürtünme kayıpları ihmal edilerek aşağıdaki gibi yazılabilir.

$$s\omega_r = \frac{1}{J}(T_{em} - T_{load}) \tag{4.6}$$

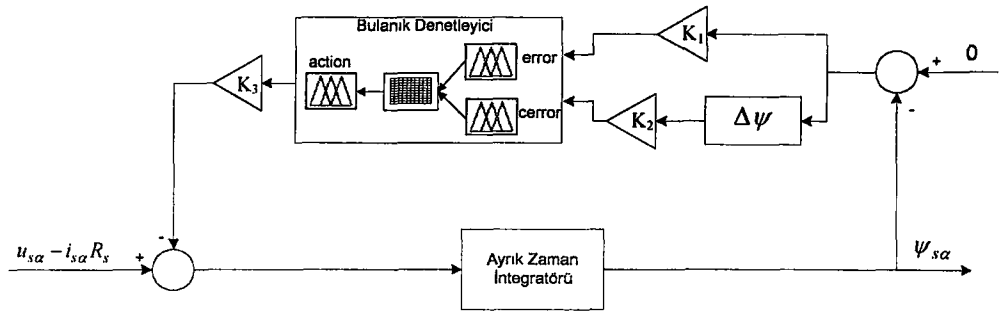
Yukarıdaki denklemler kullanılarak asenkron motorun bilgisayar benzetimi yapılabilir. Bilgisayar benzetiminde temel olarak üç blok vardır. Bunlardan biri motor modeli, ikinci histerezis kontrollü inverter diğeri ise stator akısı tahmin bloğudur. Stator akısını tahmin etmek için kullanılan gerilim modelindeki integratörün geri besleme yolundaki bulanık kontrolörün performansını değerlendirmek için bilgisayar benzetimleri MATLAB / SIMULINK programı kullanılarak yapılmıştır. Bu modelde ilk olarak asenkron makinanın modeli oluşturulmuştur. Sonra modele geri besleme yolunda PI kontrolör ve bulanık kontrolör kullanılan stator akı tahmin bloğu eklenerek model genişletilmiştir. Şekil 3.5' de gösterildiği gibi gerçek akımlar ve referans akımlar karşılaştırılarak histerezis kontrollü inverter sinyalleri üretilmiştir. Histerezis kontrollü inverter bloğundan elde edilen üç faz gerilimler α–β referans çatıya dönüştürülmüştür. Sonra bu gerilimler asenkron motor modeline uygulanarak gerçek akımlar elde edilir. Akıyı tahmin etmek için kullanılan bulanık kontrolör MATLAB Fuzzy Toolbox ile oluşturulmuştur. Simülasyonlar düşünülen akı tahmin edicinin performansını geçici durumda ve sürekli durumda araştırmak için gösterilmiştir.



Şekil 4.1. Sistemin SIMULINK modeli

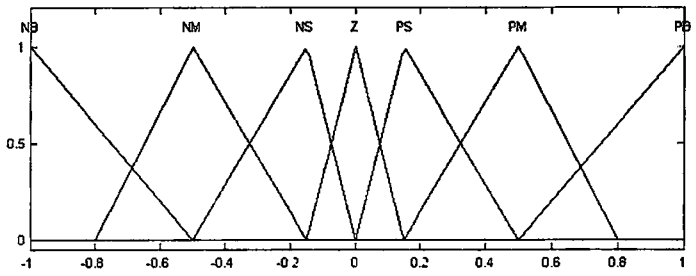
4.2. Stator Akısı Tahmini için Geliştirilen Bulanık Kontrolör

Yukarıdaki bilgisayar benzetiminde kullanılan akı tahmin bloğunun iç yapısı şekil 4.2 'de gösterildiği gibi bir bulanık kontrolör gerilim modelinde integratörün geri besleme yolunda kullanılmıştır. Burada düşünülen bulanık kontrolör algoritması hız ölçümü gerektirmemektedir. Geleneksel kontrolörler ile bir başka problem geri besleme yolundaki kontrolör parametrelerinin dizayn zorluğudur. Burada önerilen bulanık kontrolör kurallara dayanır ve farklı makinalara kolaylıkla adapte edilebilir. Geleneksel kontrolörlerden farklı olarak bulanık kontrolörler sensörlerin hatalarına ve parametrelerdeki küçük değişimlere karşı duyarlı değildir.

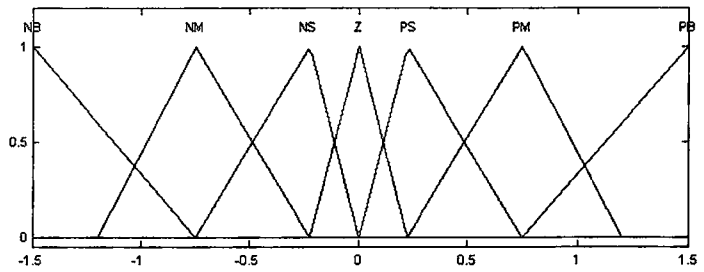


Şekil 4.2. Stator akı tahmini için kullanılan bulanık kontrolörün blok diyagramı

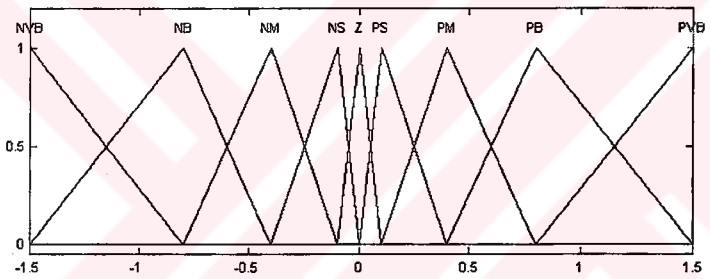
Şekil 4.2 'de gösterildiği gibi stator akı tahminin temeli $(u_{s\alpha} - i_{s\alpha}.R_s)$ ve geri besleme sinyali arasındaki farkın ayrık integraline dayanır. R_s simülasyon boyunca sabit kabul edilmiştir. Bu şekilde, ilk olarak akı geri besleme yolundaki sıfır referansla karşılaştırılmıştır. Sonra, bu fark ve farkın türevi K_1 ve K_2 katsayıları ile çarpılarak bulanık kontrolöre girilmiştir. Geri besleme yolundaki K_1 ve K_2 katsayıları sırasıyla hata ve hatanın türevini ölçeklemek için kullanılmıştır. Uygun K_1 ve K_2 katsayıları tarafından yapılan bu ölçekleme üyelik fonksiyonlarının kullanımı için önemlidir. Şöyle ki bulanık kontrolöre giriş olarak gelen değişkenlerin değerleri üyelik fonksiyonlarının taban değerleri dışında olursa tanımsız kural oluşacaktır. Bu da kontrolörün yanlış davranmasına neden olacaktır. K_3 katsayısı ise bulanık kontrolörün çıkışını yükseltmek için kullanılır. Böylece sistemin cevap zamanı geliştirilebilir. Bu şekilde sistemin yerleşme zamanı K_3 ile ayarlanabilir.



(a)



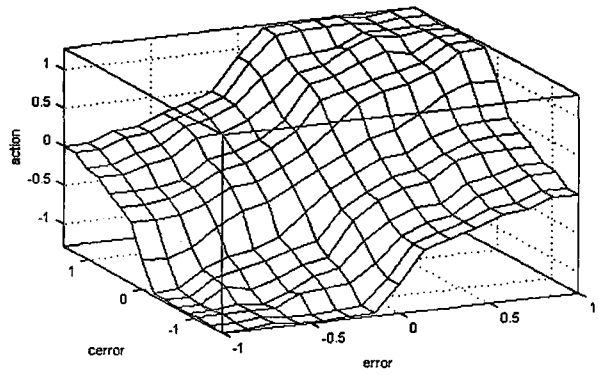
(b)



(c)

		error						
		NB	**NM**	**NS**	**Z**	**PS**	**PM**	**PB**
cerror	**NB**	NVB	NVB	NVB	NB	NM	NS	Z
	NM	NVB	NVB	NB	NM	NS	Z	PS
	NS	NVB	NB	NS	NS	Z	PS	PM
	Z	NB	NM	NS	Z	PS	PM	PB
	PS	NM	NS	Z	PS	PM	PB	PVB
	PM	NS	Z	PS	PM	PB	PVB	PVB
	PB	Z	PS	PM	PB	PVB	PVB	PVB

(d)



(e)

Şekil 4.3. Bulanık kontrolörün üyelik fonksiyonları, kural tablosu ve giriş-çıkış değişimi **(a)** Giriş değişkeni "*error*" 'ün üyelik fonksiyonları **(b)** Giriş değişkeni "*cerror*" 'ün üyelik fonksiyonları **(c)** Çıkış değişkeni "*action*" 'ün üyelik fonksiyonları **(d)** Bulanık kontrolörün kural tablosu **(e)** Bulanık kontrolörün giriş-çıkış değişimi

Giriş değişkenlerini üyelik fonksiyonlarının taban değerleri içerisinde tutan K_1 ve K_2 katsayıları aşağıdaki gibi ayarlanabilir.

$$K_1 = \frac{1}{\psi_{s\alpha_{max}}} \tag{4.7}$$

$$K_2 = \frac{1.5}{\Delta\psi_{s\alpha}} \tag{4.8}$$

K_3 katsayısının değeri ise aşağıdaki gibi türetilebilir.

$$K_3 = \frac{u}{K_1.\Delta\psi_{s\alpha} + K_2.\psi_{s\alpha}} \tag{4.9}$$

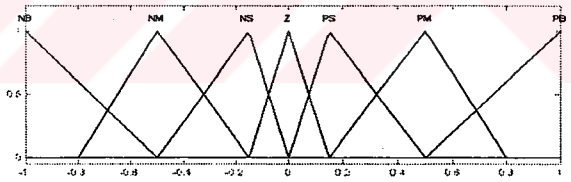
Burada u bulanık kontrolörün çıkışı $\Delta\psi_{s\alpha}$ *error* 'deki maksimum değişmedir.

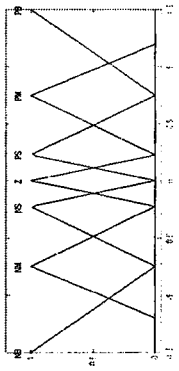
4.3. Bulanık Kontrolörlerde Kararlılık Analizi

Bilindiği gibi, ayrık zamanlı integratöre eklenen geri besleme yolu sistemi kararlı yapmak için kullanılmaktadır. Bulanık kontrolör ile oluşturulan bir sistemin kararlılık analizini yapmak için durum uzay yaklaşımından faydalanılabilir (Hu ve diğ., 1999). Bunun için kurallar ve kontrol altındaki dinamik sistemin durum uzayı arasındaki karakteristik ilişkisine ihtiyaç duyulur. Bu ilişki bulanık kontrolör tarafından üretilen kontrol hareketi üzerine kural tabanının her bir kuralının bağıl etkisine dayanır.

Kurallardan meydana gelmiş kural tabanımızı ele alalım. İki giriş değişkenine sahip sistemimizde ; error ve cerror kontrolör giriş değişkenlerini temsil etmektedir. Kuralın sonuç kısmı tek kontrol çıkışı action ı içermektedir. Sistem giriş değişkenleri error ve cerror 7 tane değere sahiptir. Bu şartlar altında kuralların max. sayısı kural tabanında 7*7=49 'dur. Durum uzayındaki bulanık kontrolörün analizi için çalışma bölgesi normal olarak bazı sonlu değerler; x_{min} , x_{max} ile sınırlanır. Durum uzayının keskin elemanları (error, cerror) j. kural ile birleştirilmiş alt uzay kısmına aittir.

$$\forall_j \neq k \; \forall(x_1, x_2) : \mu_{Rj}(x_1,x_2) => \quad \mu_{Rk}(x_1,x_2); \tag{4.10}$$





	NB	NM	NS	Z	PS	PM	PB
PB	Z 43	PS 44	PM 45	PB 46	PVB 47	PVB 48	PVB 49
PM	NS 36	Z 37	PS 38	PM 39	PB 40	PVB 41	PVB 42
PS	NM 29	NS 30	Z 31	PS 32	PM 33	PB 34	PVB 35
Z	NB 22	NM 23	NS 24	Z 25	PS 26	PM 27	PB 28
NS	NVB 15	NB 16	NS 17	NS 18	Z 19	PS 20	PM 21
NM	NVB 8	NVB 9	NB 10	NM 11	NS 12	Z 13	PS 14
NB	NVB 1	NVB 2	NVB 3	NB 4	NM 5	NS 6	Z 7

Şekil 4.4. Bulanık kontrolörün yörünge yolu ile kararlılık analizi

Burada $\mu_{Rj}(x_1,x_2)$ ve $\mu_{Rk}(x_1,x_2)$ sırasıyla j. ve k. kuralların kural varsayım kısımlarını anlamlarını belirten bulanık ilişkilerdir. "Bölmelendirme sınırları" (4.10) denklemi vasıtasıyla hesaplanır. Bir kapalı-çevrim sistem yörüngesi şekil 4.4 'de yapıldığı gibi bölmeli uzay üzerinde haritalanabilir. Şekilden bölmeli uzayın hangi kesin bölgeleri sistem yörüngesi ile ilişkili olduğunu görülebilir. Ardışık kurallar ateşleme emirlerine göre bulunur ve formlar dilsel yörünge olarak isimlendirilir, öyle ki kesin bir sistem yörüngesine karşılık gelir. Daha sonra dilsel yörünge sistem yörüngesi ile birleştirilir. Bu durum aşağıda verilmiştir ;

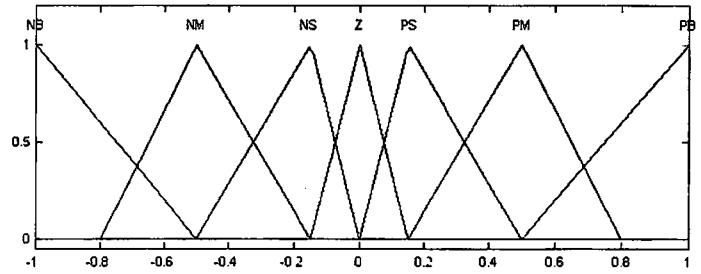
Dilsel yörünge = (kural 17, kural 18, kural 19, kural 20, kural 22, kural 24, kural 25, kural 27, kural 29, kural30, kural31, kural32, kural33).

Tasarım açısından bu metot bulanık kontrolörün analizi için meselenin ilgi çekici ana noktalarını gösterir. Sistemin kararlılığı için bu sonuçlar niteldir fakat geri beslemeli sistemin dinamik davranışını karakterize etmeye yeterlidir ve kuralların değiştirilmesi veya uygun seçimi için kullanılır.

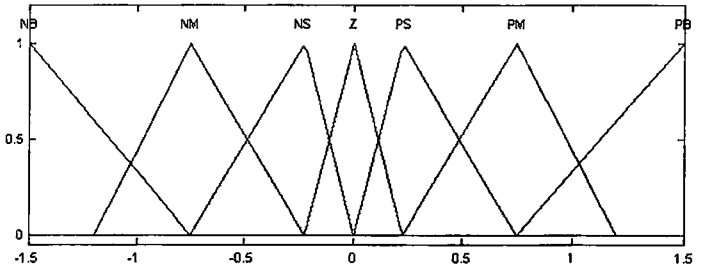
4.4. Bulanık Mantığın Gerçekleştirilmesi

Bulanık mantık nispeten yeni bir teoridir. Bulanık sistemler üzerine yapılan çalışmaların çoğu yazılım simülasyonlarına veya büyük gerçek sistemlere sahiptir. Bununla birlikte mikroişlemciler ile birlikte bulanık mantığı araştırmak ve uygulamak için basit, gerçek zamanlı ve gerçek sistem platformları olanağı doğmuştur.

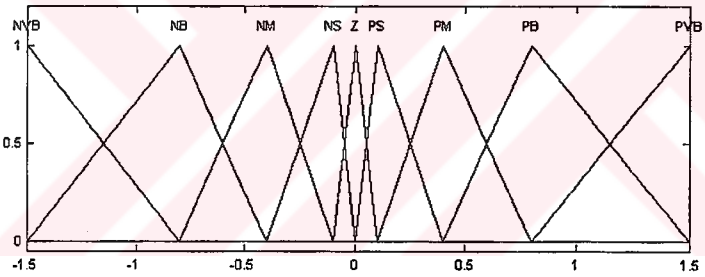
Bulanık mantığın gerçekleştirilmesi için üç aşama vardır. Bunlar bulanıklaştırma, çıkarım ve durulaştırmadır. Sistemde kullandığımız üyelik fonksiyonları basit olarak üçgen seçilmiş olup iki giriş için yedi ve çıkış için ise dokuz üyelik fonksiyonu tanımlanmıştır. Bunlar aşağıda görülmektedir.



(a)



(b)

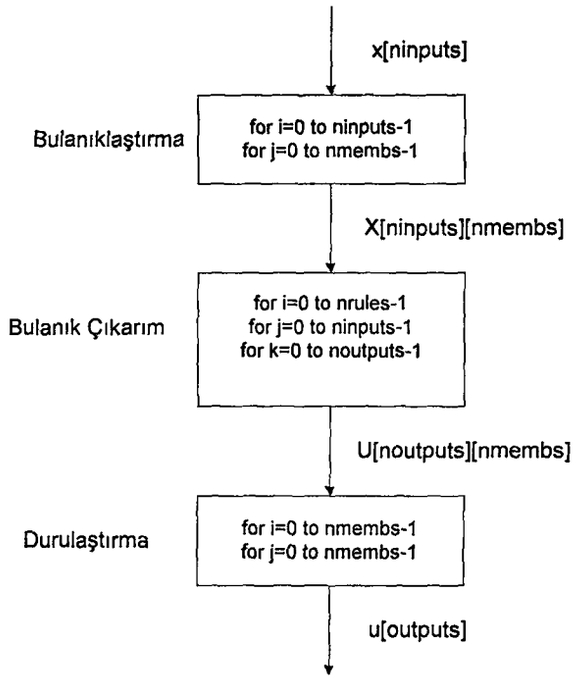


(c)

Şekil 4.5. Gerçekleştirmede kullanılan üyelik fonksiyonları

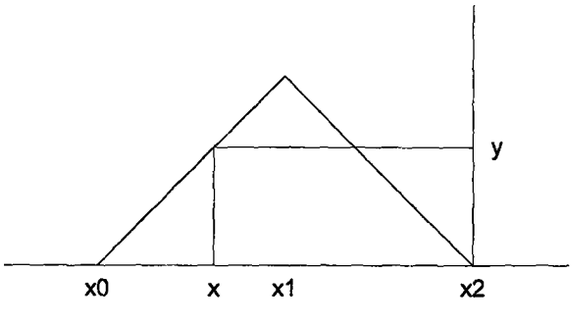
Giriş değişkenleri hata *"error"* ve hatadaki değişim *"cerror"* 'dir. Çıkış değişkeni ise *"action"* dir. Sistemden istenen cevap bilgisine göre oluşturulan kurallar aşağıdaki tabloda görülmektedir.

Bulanık mantık hesaplamalarında diziler kullanılır ve bunlar gerçekleştirme için gerekli değişik döngüler içerisinde değiştirilirler. Aşağıdaki şekilde bu dizilerin her aşamada nasıl kullanıldığı görülmektedir.



Şekil 4.6. Gerçekleştirmede kullanılan dizilerin durumu

Bulanıklaştırma döngüsünde giriş üyelik fonksiyonuna karşılık gelen her girişin üyelik derecesi değerlendirilir ve X[ninputs][nmembs] dizisinde yazılır. Dilsel değişkenlerin değerleri giriş değerinin üçgenlerde karşılık geldiği değerdir. Bu değer üçgenlerdeki doğruların denklemlerinden yararlanılarak bulunur. Denklem geometrik olarak aşağıdaki şekil için şöyle ifade edilebilir.



Şekil 4.7. Dilsel bir değişkenin analizi

$$If\ \{x : x0 < x < x1\}\ then\ y = (x - x0)/(x1 - x0)\ else$$
$$If\ \{x : x1 < x < x2\}\ then\ y = (x2 - x)/(x2 - x1) \qquad (4.11)$$

Tablo 4.1. İndekslenmiş kural tablosu

Kural tablosu			Kuralların indekslenmiş hali			Kuralların tekrar indekslenmesi		
error	*cerror*	*action*	*error*	*cerror*	*action*	*error*	*cerror*	*action*
NB	NB	NVB	-3	-3	-4	0	0	0
NB	NM	NVB	-3	-2	-4	0	1	0
NB	NS	NVB	-3	-1	-4	0	2	0
NB	Z	NB	-3	0	-3	0	3	1
NB	PS	NM	-3	1	-2	0	4	2
NB	PM	NS	-3	2	-1	0	5	3
NB	PB	Z	-3	3	0	0	6	4
NM	NB	NVB	-2	-3	-4	1	0	0
NM	NM	NVB	-2	-2	-4	1	1	0
NM	NS	NB	-2	-1	-3	1	2	1
NM	Z	NM	-2	0	-2	1	3	2
NM	PS	NS	-2	1	-1	1	4	3
NM	PM	Z	-2	2	0	1	5	4
NM	PB	PS	-2	3	1	1	6	5
NS	NB	NVB	-1	-3	-4	2	0	0
NS	NM	NB	-1	-2	-3	2	1	1
NS	NS	NS	-1	-1	-1	2	2	3
NS	Z	NS	-1	0	-1	2	3	3
NS	PS	Z	-1	1	0	2	4	4
NS	PM	PS	-1	2	1	2	5	5
NS	PB	PM	-1	3	2	3	6	6
Z	NB	NB	0	-3	-3	3	0	1
Z	NM	NM	0	-2	-2	3	1	2
Z	NS	NS	0	-1	-1	3	2	3
Z	Z	Z	0	0	0	3	3	4
Z	PS	PS	0	1	1	3	4	5
Z	PM	PM	0	2	2	3	5	6
PS	PB	PB	1	3	3	4	6	7
PS	NB	NM	1	-3	-2	4	0	2
PS	NM	NS	1	-2	-1	4	1	3
PS	NS	Z	1	-1	0	4	2	4
PS	Z	PS	1	0	1	4	3	5
PS	PS	PM	1	1	2	4	4	6
PS	PM	PB	1	2	3	4	5	7
PM	PB	PVB	2	3	4	5	6	8
PM	NB	NS	2	-3	-1	5	0	3
PM	NM	Z	2	-2	0	5	1	4
PM	NS	PS	2	-1	1	5	2	5
PM	Z	PM	2	0	2	5	3	6
PM	PS	PB	2	1	3	5	4	7
PM	PM	PVB	2	2	4	5	5	8
PB	PB	PVB	3	3	4	6	6	8
PB	NB	Z	3	-3	0	6	0	4
PB	NM	PS	3	-2	1	6	1	5
PB	NS	PM	3	-1	2	6	2	6
PB	Z	PB	3	0	3	6	3	7
PB	PS	PVB	3	1	4	6	4	8
PB	PM	PVB	3	2	4	6	5	8
PB	PB	PVB	3	3	4	6	6	8

Sonraki adım bulanık çıkarım sistemini içerir. Bu döngü içerisinde maksimum ve minimum fonksiyonlar kullanılır. Burada kuralların birbirleriyle karşılaştırılmaları için indekslerin negatif olmaması gereklidir. Bu yüzden bütün kurallar sıfırdan başlatılarak tekrar indekslenir. Bu indeksleme daha önceki tabloda verilmiştir. Bu tabloda RULE_TABLE[rule][input+output] dizisindeki indeksler görülmektedir.

X[ninputs][nmembs] dizisinin minimum değerini bulmak için,

$$minZ = min\,(X[input]\ \ [RULE\_TABLE\,[rule][input]]) \tag{4.12}$$

kullanılır. Buradan bulunan minimum değerler minZ dizisinde depolanır. Burada kural tabanının sadece ilk iki sütunu yani giriş değerleri kullanılarak hesaplamalar yapılır. Bu durumda, her bir kural için çıkış değerleri U[nmembs] 'in maksimumu alınır. U[output][nmembs] basitleştirilmiştir çünkü kural tablosu değerlerinden ve *minZ* 'den sadece bir çıkış üretilir. Bu denklem şöyle yazılabilir.

$$\begin{aligned} U[nmembs] &= U(RULE\_TABLE[rule][ninput + output]) \\ &= max\,(U[RULE\_TABLE[rule][ninput + output]], minZ) \end{aligned} \tag{4.13}$$

Bu işlem yapılırken kural tablosunun sadece son sütunu yani çıkış değerlerinin olduğu sütun kullanılır. Denklem en son haliyle aşağıdaki gibi elde edilebilir.

$$\begin{aligned} U[nmembs] = max\,\{&U[RULE\_TABLE[rule][ninput + output]], \\ &min\,(X[input]\ \ [RULE\_TABLE\,[rule][input]])\} \end{aligned} \tag{4.14}$$

Bu işlem bütün kurallar için devam ettirilir ve *U[nmembs]* minimumların maksimumlarını içerir.

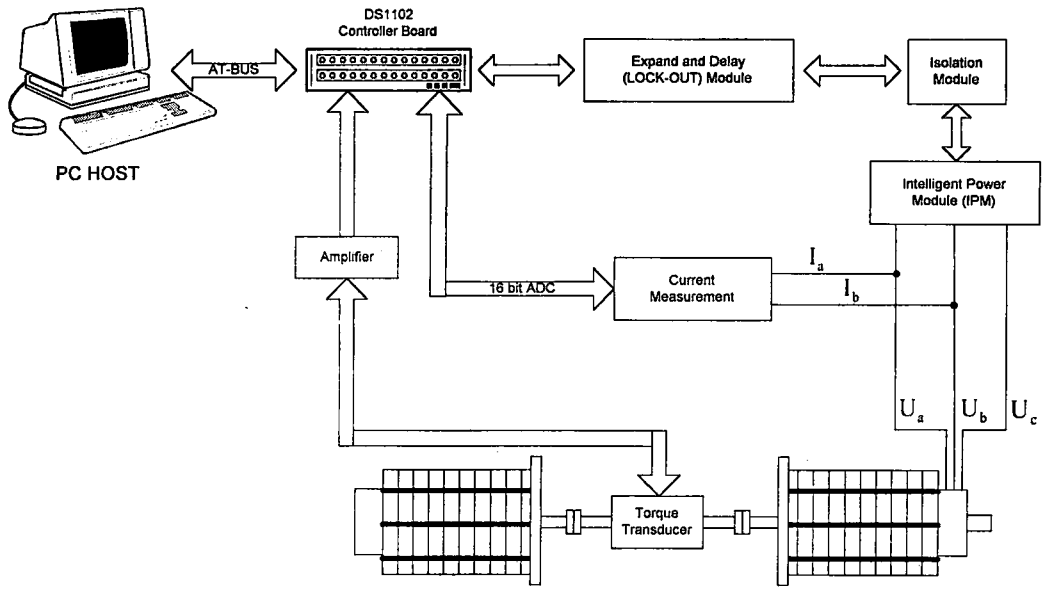
Son adım U[noutputs][nmembs] dizisinin durulandırma işlemidir. U[noutputs][nmembs] dizisi noutputs=1 olduğundan dolayı U[nmembs] olarak basitleştirilebilir. Durulandırma işlemi sonucunda U[nmembs] dizisi 7 üyelik fonksiyonuna karşılık 7 değere sahip olacaktır. Durulandırma metodu olarak kullanılan ağırlık merkezi yöntemi ise iki döngü ile gerçekleştirilebilir. İlk döngü bunun pozisyonu ile U[nmembs] 'in ağırlığı çarpılarak pay bulunur. İkinci döngüde ise pozisyonların

toplanarak payda bulunur ve bu iki değer bölünerek nihai sonuca ulaşılır. Bu çalışmada geliştirilen yöntemin TMS320C31 mikroişlemcisi ile gerçekleştirilmesi için gerekli program EK-A 'da verilmiştir.

Sonuç olarak bulanık mantığın gerçekleştirilmesindeki dezavantaj bilinen kontrol metotlarına göre çok fazla işlem zamanı tüketmesidir. Fakat son yıllarda gelişen teknoloji ile üretilen mikroişlemciler sayesinde bu problem aşılmış ve iyi bir kontrol yöntemi olarak bulanık mantık ön plana çıkmıştır. Sonuç olarak geleneksel PID tipi kontrol sistemlerine göre bulanık mantık daha üstün davranmaktadır. Bu nedenle bulanık mantık mühendislik için programlanabilir bir çözüm olarak muhteşem bir potansiyele sahiptir.

4.5. Deney Setinin Tanıtılması

Bu çalışmada önerilen yöntemin çalışmasını doğrulamak için bir asenkron motor sürücü sistemi oluşturulmuştur. Şekil-11'de sürücü sistemin blok şeması verilmiştir. Bu sisteme alan yönlendirmeli kontrol uygulanmıştır. Motoru sürme devresi olarak histerezis kontrol ile kontrol edilen IGBT inverter kullanılmıştır. Sistemde sayısal kontrol kartı olarak DS1102 dSPACE GmbH kullanılmıştır. Kontrol kartının işlemcisi TMS320C31 olup 32 bit, floating-point ve 60ns komut çevrimine sahiptir. Ayrıca DS1102 dijital kontrol kartında 4 adet analog-dijital çevirici (2 adet 12 bit, 2 adet 16 bit), 4 adet 12 bit dijital-analog çevirici ve 2 adet incremental encoder bulunmaktadır. Bu deney setinde akım ve gerilim bilgilerini ölçmek için LEM sensörler kullanılmıştır. Gerilim bilgisi ölü zaman etkilerini içeren inverterin anahtarlama pozisyonundan elde edilir. Vektör kontrol algoritması 35 μs zaman alır. Önerilen bulanık kontrolörlü yeni algoritmayı içeren alan yönlendirme kontrol algoritmasının çalışma çevrim zamanı 80μs'dir.



Şekil 4.8. Deney düzeneği

4.6. Simülasyon ve Deneysel Sonuçlar

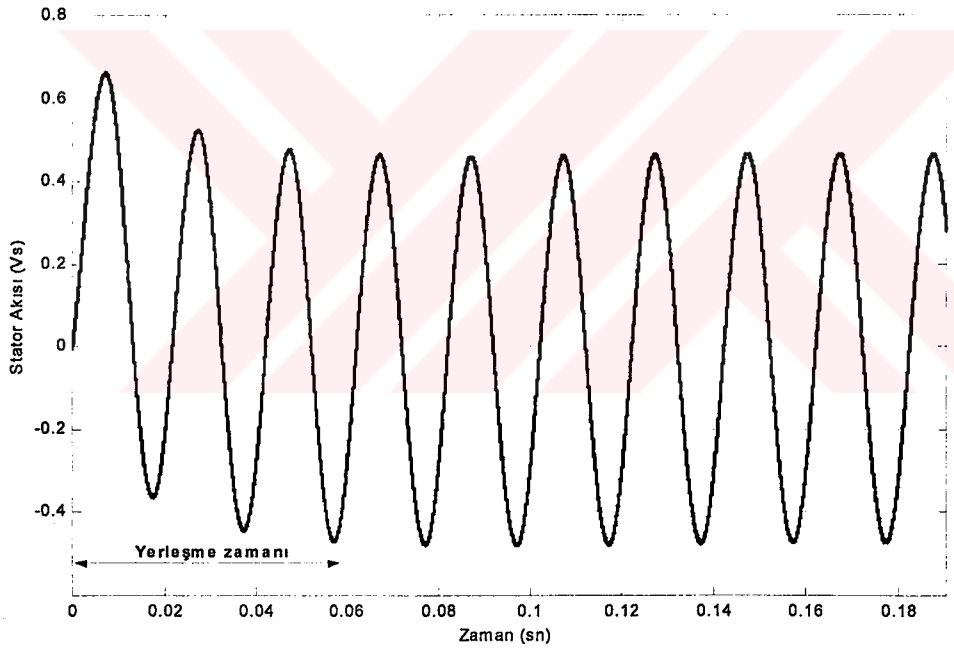
Yukarıda anlatılan benzetimler farklı güçlerdeki üç asenkron motor kullanılarak gerçekleştirilmiştir. Burada PI kontrolör ve bulanık kontrolörden elde edilen sonuçlar karşılaştırılmıştır. Deneysel olarak bir motorda gerçekleştirilmiş olan sistem sonuçları aşağıdaki gibi değerlendirilmiştir.

Şekil 4.9 ve 4.10 'de sırasıyla 1 HP 'lik motor için bulanık kontrollü ve PI kontrollü integratörden elde edilen stator akı dalga şekilleri verilmiştir. Bu şekillerde dikkat edilirse akı belirli bir yerleşme zamanı içerisinde sürekli duruma ulaşmıştır. Bulanık kontrolörlü integratörde bu zaman PI kontrolörlü integratöre göre biraz daha iyidir. Aynı simülasyonlar şekil 4.11 'de görüldüğü gibi 3 HP 'lik motor için de yapılmıştır ve sonuçların birbirine yakın oldukları fakat yine bulanık kontrolörlü integratörden elde edilen akı dalga şeklindeki yerleşme zamanı ve faz hatasının daha az olduğu görülmektedir.

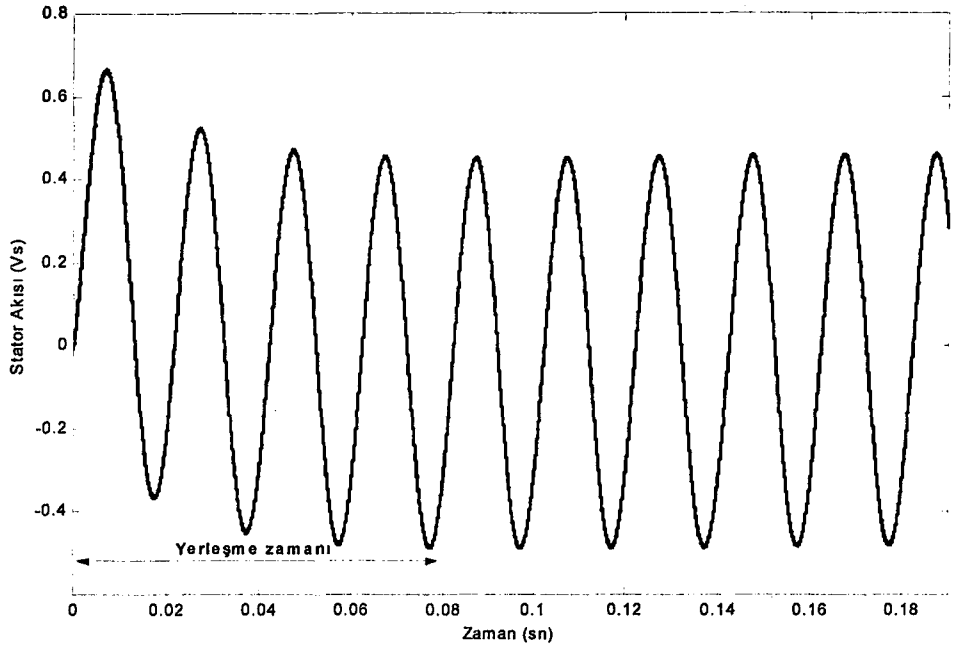
Şekil 4.12 ve 4.13 'da 3 HP 'lik motor için bulanık kontrolörlü integratörün simülasyon sonucu ve deneysel sonucu görülmektedir. Bu sonuçlar arasındaki fark ihmal edilebilecek kadar azdır. Yukarıda bahsedilen simülasyonların aynısı 10 HP 'lik

motor için de yapılmış ve şekil 4.14 ve 4.15 deki sonuçlar elde edilmiştir. Bunlar sırasıyla bulanık kontrolörlü integratör ile PI kontrolörlü integratör için alınan sonuçlardır. Şekillerin birbirine yakınlığı görülmektedir. Fakat yerleşme zamanı ve faz hatası bakımından yine azda olsa bulanık kontrolörün daha iyi cevap verdiği şekillerden görülmektedir.

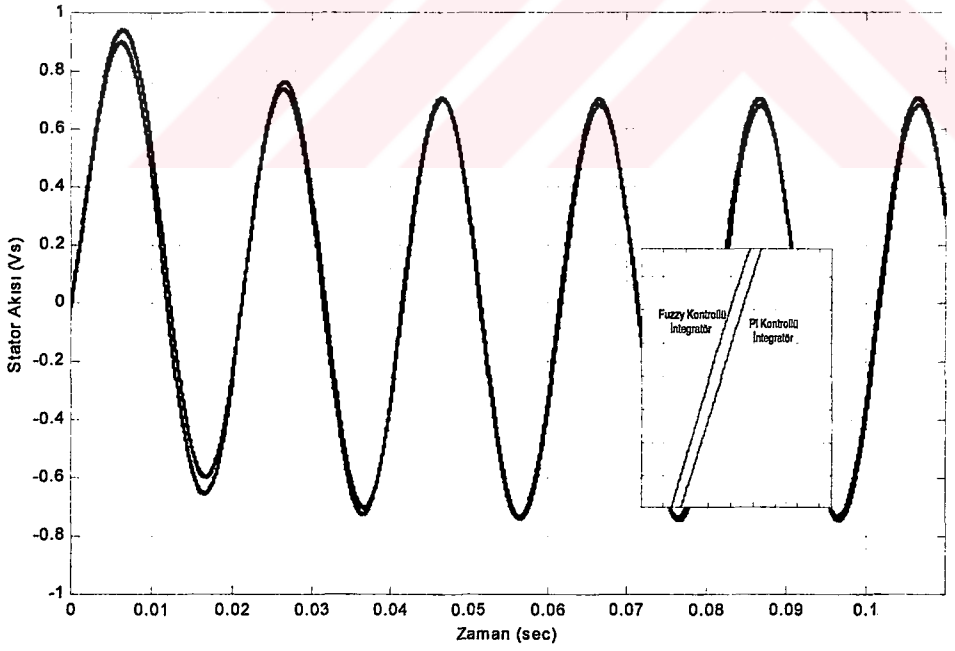
Şekil 4.16 ve 4.17 'de ise histerezis kontrollü inverter için simülasyon sonucu ve deneysel olarak elde edilen akım dalga şekillerinin sürekli durumdaki değişimleri verilmiştir. Yukarıda bahsedilen bütün deneysel sonuçlar TRACE31 yazılımı tarafından elde edilmiştir. Deneyler ve simülasyonlar kullanılan üç farklı motor için farklı moment referanslarında ve yük koşullarında denenmiştir. Bu simülasyonlarda bulanık kontrolörün PI kontrolörden daha iyi olduğu gözlenmiştir.



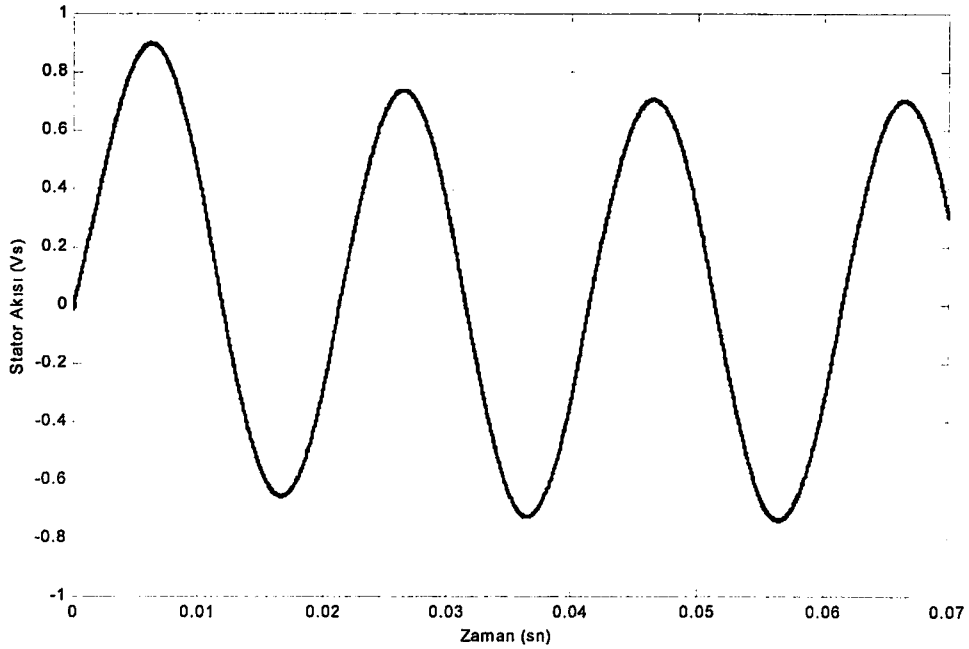
Şekil 4.9. 1 hp motor için bulanık kontrolörlü integratörün geçici zaman cevabı



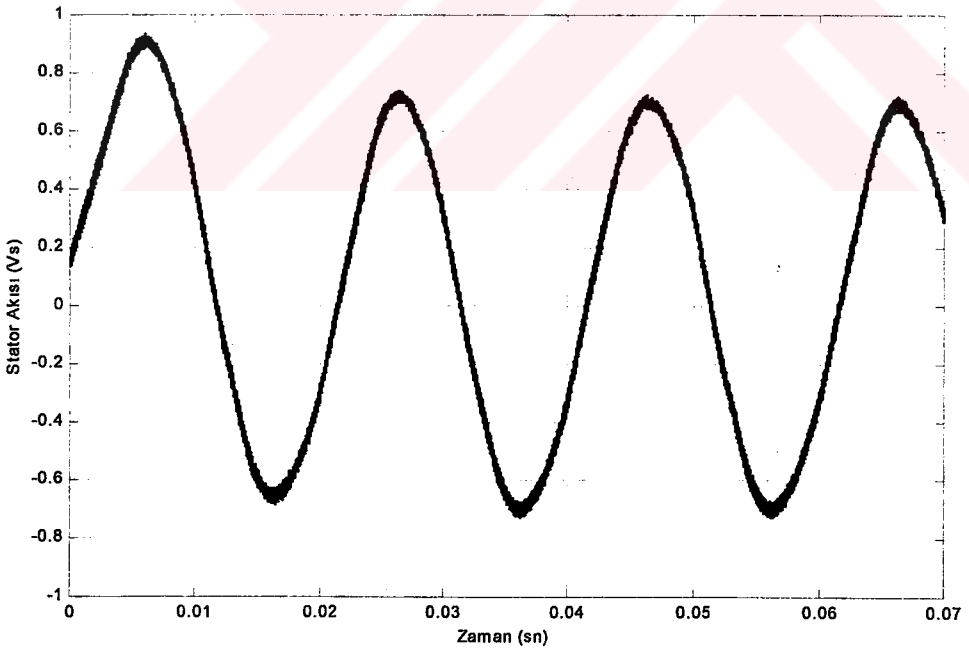
Şekil 4.10. 1 hp motor için PI kontrolörlü integratörün geçici zaman cevabı



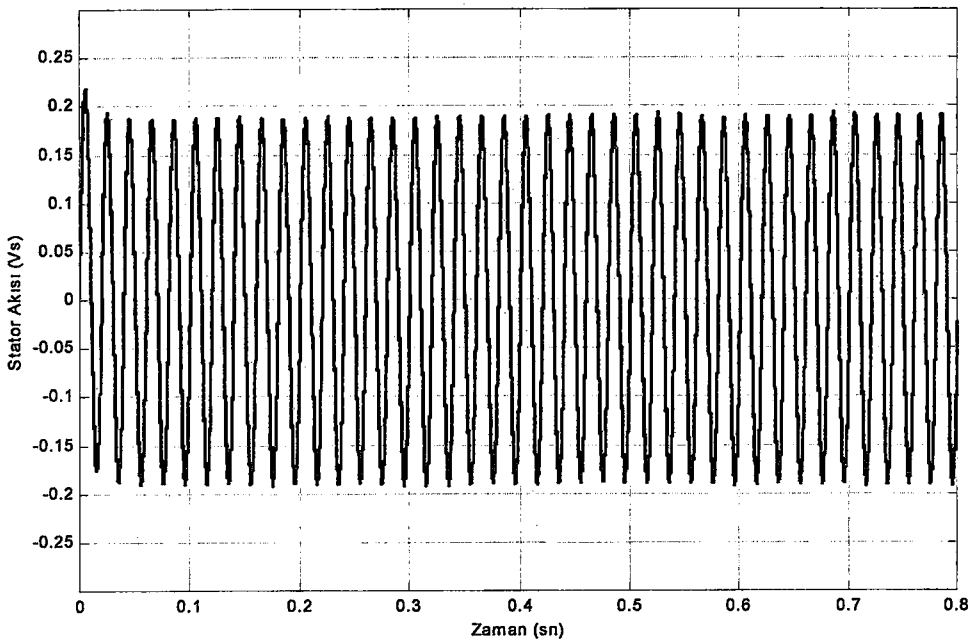
Şekil 4.11. 3 hp motor için bulanık ve PI kontrolörlü integratörün simülasyon sonuçları



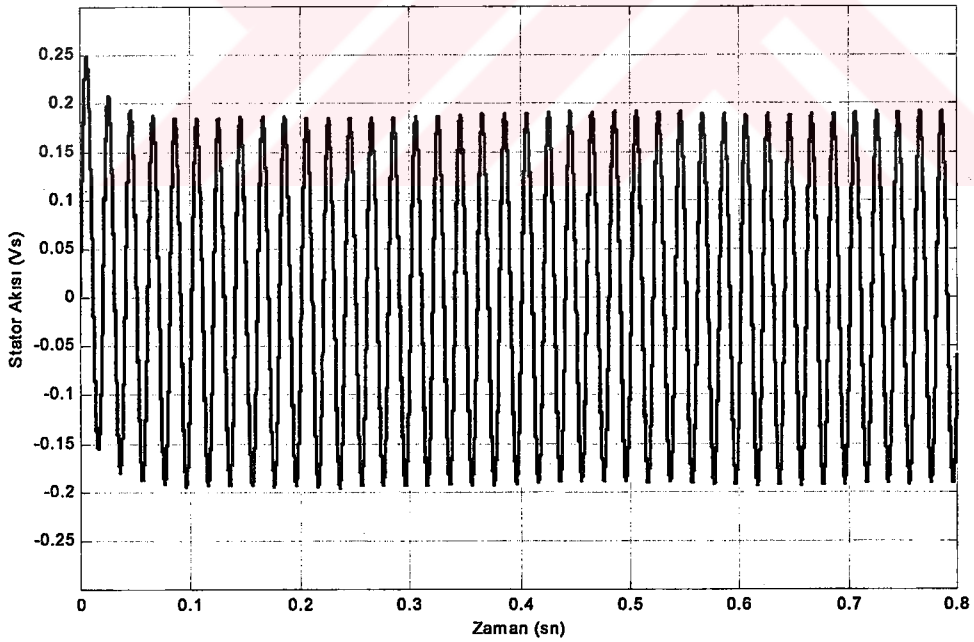
Şekil 4.12. 3 hp motor için bulanık kontrolörlü integratörün simülasyon sonuçları



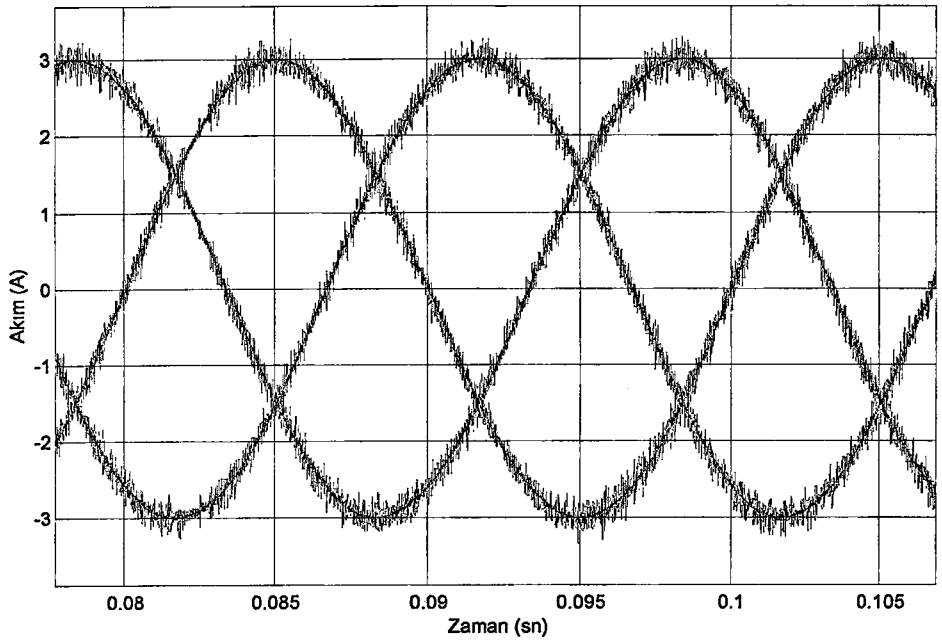
Şekil 4.13. 3 hp motor için bulanık kontrolörlü integratörün deneysel sonuçları



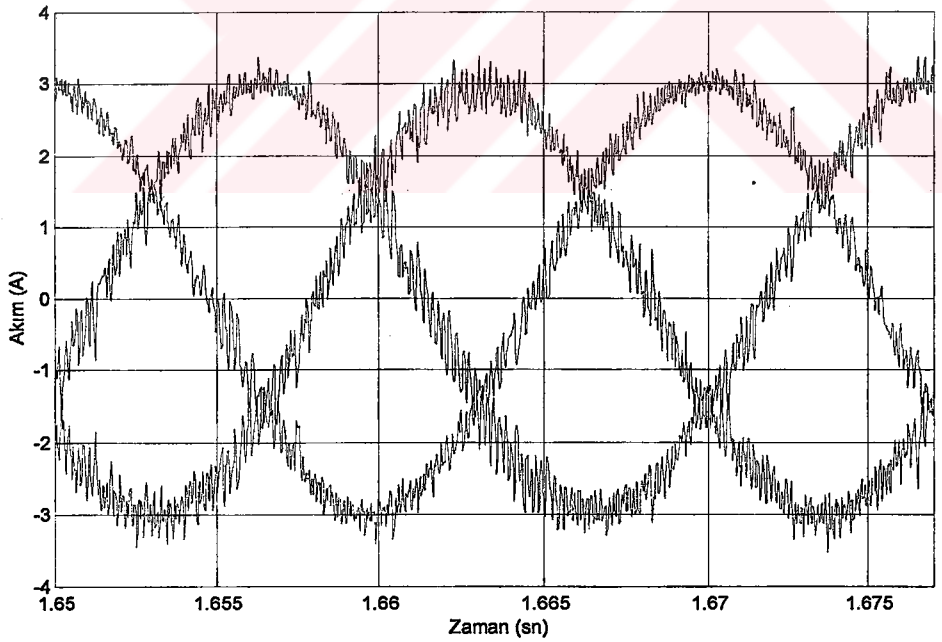
Şekil 4.14. 10 hp motor için bulanık kontrolörlü integratörün simülasyon sonuçları



Şekil 4.15. 10 hp motor için PI kontrolörlü integratörün simülasyon sonuçları



Şekil 4.16. Histerezis kontrollü inverterden elde edilen akım dalga şekillerinin simülasyon sonuçları



Şekil 4.17. Histerezis kontrollü inverterden elde edilen akım dalga şekillerinin deneysel sonuçları

Deney ve simülasyonlarda kullanılan motorların parametreleri aşağıdaki tabloda verilmiştir.

Tablo 4.2. Simülasyonlarda ve deneyde kullanılan motorların parametreleri

	Motor parametreleri		
Güç	1 hp	3 hp	10 hp
Faz gerilimi	200 V	220 V	220 V
Kutup sayısı	4	2	6
Stator direnci	3.35 Ω	0.435 Ω	0.0453 Ω
Rotor direnci	1.99 Ω	0.816 Ω	0.0222 Ω
Stator kaçak reaktansı	0.754 Ω	0.754 Ω	0.0775 Ω
Rotor kaçak reaktansı	0.754 Ω	0.754 Ω	0.0322 Ω
Stator mıknatıslanma reaktansı	27.2 Ω	26.13 Ω	2.042 Ω
Atalet momenti	0.05 kgm^2	0.089 kgm^2	0.866 kgm^2

4. SONUÇLAR VE TARTIŞMA

Bu tez çalışması, bir asenkron motorun stator akı tahmini için yapılan yeni bir algoritmayı incelemektedir. Bu çalışmada aşağıdaki sonuçlar elde edilmiştir.

Asenkron motorun stator akısının tahmininde kullanılan gerilim modelindeki integrasyonun bir sonucu olarak meydana gelen problemleri ortadan kaldırmak için bulanık kontrollü bir tahmin algoritması geliştirilmiştir. Gerilim modelinde kullanılan açık integrasyon nedeniyle oluşan sürüklenme ve faz hataları geri besleme yolu üzerine konulan bir katsayı ile kısmen çözülmesine rağmen elde edilen akı hala ofset ve faz hataları içermektedir. Bunun için akının sıfır ile karşılaştırılması ile elde edilen hata ve bu hatadaki değişim kullanılarak bulanık kontrollü bir tahmin algoritması geri besleme yolu üzerinde kullanılmıştır. Bu algoritmanın MATLAB/SIMULINK kullanılarak bilgisayar benzetimi yapılmış, TMS320C31 mikroişlemcisi kullanılarak deneysel olarak gerçekleştirilmiştir. Bilgisayar benzetimleri değişik motor tipleri ve değişik yük momentleri için yapılarak yöntemin oldukça iyi sonuç verdiği gözlemlenmiştir. Bu benzetimlerde aynı zamanda PI geri beslemeli akı tahmini kullanılarak bulanık kontrollü algoritma karşılaştırılmıştır. Bulanık kontrollü akı tahmininin daha iyi sonuç verdiği sonuçlardan görülmüştür. Aynı durum deneysel olarak bir motorda incelenmiş ve burada da bulanık kontrollü akı tahmin algoritmasının iyi sonuç verdiği görülmüştür. Burada bulanık kontrolün bir dezavantajı yöntemin gerçekleştirilirken PI kontrole göre daha fazla süre harcamasıdır. Fakat günümüzdeki mikroişlemcilerin hızları göz önüne alınınca artık bu sorun da ortadan kalkmaya başlamıştır. Çalışmada ayrıca PI kontrole göre bulanık kontrol yapısının oldukça sağlam olduğu ve basit bir dizayna sahip olduğu da görülmüştür. Elde edilen akı dalga şekillerine bakıldığında ise görülen bulanık kontrollü yöntem ile akının daha çabuk sürekli duruma ulaşmasıdır. Ayrıca akıdaki ofset de elimine edilerek tamamen sıfıra çekilmiştir.

Bu çalışmada kullanılan bulanık kontrol algoritmasındaki üyelik fonksiyonlarının taban değerleri deneme yanılma yolu ile bulunan değerlerdir. Bu çalışmanın önemli bir kısmını oluşturan üyelik fonksiyonlarının taban değerlerinin genetik algoritma ile otomatik olarak ayarlanması ileriki çalışmaların amacı olacaktır.

REFERANSLAR

Akın E., 1994, Stator Akısı Üzerinden Asenkron Motorun Rotor Akısı Alan Yönlendirmesi İçin Bir Yöntem, Elektrik-Elektronik Anabilim Dalı ;Doktora Tezi, F.Ü., Fen Bilimleri Enstitüsü, Elazığ.

Akın E., Can H., Ertan H.B., Üçtuğ Y., "Comparison of Integration Algoritms for Vector Control", *ICEM98*, Vol. 3, pp. 1626-1631, September, 2-4, 1998, İstanbul, Turkey.

Astrom, K., Hang, C., Persson. P. and Ho, W., Towards intelligent PID control, Automatica, 28(1):1-9, 1992.

Astrom, K.J. and Hagglund, T., PID Controllers-Theory, Design and Tuning, second edn, Instrument Society of America, 67 Alexander Drive, PO Box 12277, Research Triangle Park, 1995, North Carolina, 27709, USA.

B.K. Bose, N.R. Patel, "A Programmable Cascaded Low-Pass Filter-Based Flux Synthesis for a Stator Flux-Oriented Vector-Controlled Induction Motor Drive", *IEEE Transactions on Industrial Electronics*, vol. 44, no. 1, pp. 140-143, February, 1997.

Chee-Mun Ong, *Dynamic Simulation of Electric Machinery*, Prentice Hall PTR, Upper Saddle River, New Jersey, 1998.

Driankov D., Hellendoorn H., Reinfrank M., *An Introduction to Fuzzy Control*, Springer-Verlag Berlin Heidelberg, 1996.

Drieseitl W., etc. Schaltungsanordnung zur bildung eines Elektrischen Spannungssignals, das einer Flubkomponente in einer Drehfeldmaschine proportional ist. Patentschrift Nr DE 2833593. C2., 1980.

Hu B., Mann G.K.I., R.G. Gosine, "New Methodology for Analytical and Optimal Design of Fuzzy PID Controllers", *IEEE Trans. On Fuzzy Systems*, vol. 7, no. 5, 521-539, October 1999.

Jos van der Burgt, *The Voltage/Current Model in Field-Oriented AC Drives at Very Low Flux Frequencies*, PhD Thesis, 1996.

Mizomoto, M., Realization of PID controls by fuzzy controls method, IEEE First Int. Conf. On Fuzzy systems, number 92CH3073-4, The Instute of Electrical and Electronics Engineers, Inc, San Diego, pp. 709-715, 1992.

Ohtani T., Takada T., Tanaka K., "Vector Control of Induction Motor without Shaft Encoder", *IEEE Transactions on Industry Applications*, vol. 28, no. 1, pp. 157-164, January/February, 1992.

Özer A.B., 2000, Hız Duyargasız Vektör Kontrol Sisteminin Hız Kontrolünün Gerçekleştirilmesi, Bilgisayar Mühendisliği Anabilim Dalı, Yüksek Lisans Tezi, F.Ü., Fen Bilimleri Enstitüsü, Elazığ.

Passino K.M., Yurkovich S., *Fuzzy Control*, Addison-Wesley, 1998.

Qiao, W. and Mizumoto, M., PID type fuzzy contrtoller and parameters adaptive method, Fuzzy Sets and Systems, 78, 23-35, 1996.

Siler, W. and Ying, H., Fuzzy Control Theory: The linear case, Fuzzy Sets and Systems, 33, 275-290, 1989.

Smith, L.C., Fundamentals of control theory, Chemical Enigneering, 86(22):11-39, 1979.

The MathWorks Inc., *The Student Edition of MATLAB Student User Guide*, Prentice Hall, 1992.

Tso, S.K. and Fung, Y.H., Methodological development of fuzzy-logic controllers from multivariable linear control, IEEE Trans. Systems, Man & Cybernetics, 27(3), 566-572, 1997.

EK A. Deneyi gerçekleştirmek için gerekli program

```
/* Bu program bulanık kontrolör kullanarak akı tahmini yapan programdır.*/

#include "brtenv.h"
#include "math.h"
#define TS 800.0e-6
#define TS1 40.0e-3

/* Değişkenlerin tanımı */

float a,b,c,d;
float rs=2.8;
float xss=0.23;
float sig=0.126;
float Lm=0.2152;
float isdref,isqref=0,isdsr,isqsr,isar,isbr,iscr,isa,isb,isc;
float sintet=0;
float costet=1;
float t=0;
float ibb=0;
float fia,fib,fialfa,fibeta,fi;
float A1,B1;
float e1=0;
float b2=0;
float etop=0;
float btop=0;
float pi=3.141592654;
float kd=14;
float fialfa1,fibeta1,hiz,hiz0,hiz1;
int   err = 0;

float udc=556;
float usa,usb;
float akie1=0;
float akib2=0;
float akietop=0;
float akibtop=0;
float akiA1,akiB1,akifia,akifib;
float hata=0;
float n=0;
float max=0,min=0;
float wk,say,rpm=0;
int s=0;
float rpmref=500,er=0,rpmki=0.003,rpmkp=0.0025;
int i,j,k,minz[48],pp,qq,rr;
float error,cerror,action,x[1][6];
float value1,value2,value3,value4,value5,value6;
float input[1][48],output[48],actiont[8],u[8],actions,pay,payda;
```

```
unsigned long count0;
volatile int *error = (int *) (DP_MEM_BASE + DP_MEM_SIZE - 1);
void isr_t0()
{
   ds1102_ad_start();
   service_trace();
   isdref=3;
   if (t>=1) rpmref=500;
   if (t>=3) rpmref=75;


   isdsr=(-isqref*sintet)+(isdref*costet);
   isqsr=(isqref*costet)+(isdref*sintet);
   isar=isdsr;
   isbr=(-0.5*isdsr)+(0.866025403*isqsr);
   iscr=(-0.5*isdsr)+(-0.866025403*isqsr);

   isa=5.830*(ds1102_ad(1));
   isb=5.458*(ds1102_ad(2));
   isc=-(isa+isb);

   if (isa<(isar-ibb)) a=1;
   if (isa>(isar+ibb)) a=0;
   if (isb<(isbr-ibb)) b=1;
   if (isb>(isbr+ibb)) b=0;
   if (isc<(iscr-ibb)) c=1;
   if (isc>(iscr+ibb)) c=0;
   d=(4*a+2*b+c)*256;
   ds1102_p14_pin_io_write(d);

      usa=udc*(0.6666666*a-0.3333333*b-0.3333333*c);
      usb=udc*(0.6666666*b-0.3333333*a-0.3333333*c);

/*Akı tahmin algoritması */

      akiA1=(usa-isa*rs)-actionsa*akietop;
      akietop=akie1+akiA1*TS;
      akie1=akietop;
      akifia=akietop-sig*xss*isa;

/*Alfa bileşeni için bulanık kontrol algoritması*/

         error1=0-akifia;
         cerror=error1-error;

/*error giriş değişkeni için üyelik fonksiyonlarının denklemleri*/

if (error>-1 && error<-0.5){
value1=(-100*error-50)/50;      // 1.üyelik fonk.
value2=(100*error+80)/30;       // 2.üyelik fonk
```

```
x[0][0]=value1;
x[0][1]=value2;
x[0][2]=0;
x[0][3]=0;
x[0][4]=0;
x[0][5]=0;
x[0][6]=0;
}
else if (error>-0.5 && error<-0.15){
value1=(-100*error-15)/35;
value2=(100*error+50)/35;
x[0][0]=0;
x[0][1]=value1;
x[0][2]=value2;
x[0][3]=0;
x[0][4]=0;
x[0][5]=0;
x[0][6]=0;
}
else if (error>-0.15 && error<0){
value1=(-100*error)/15;
value2=(100*error+15)/15;
x[0][0]=0;
x[0][1]=0;
x[0][2]=value1;
x[0][3]=value2;
x[0][4]=0;
x[0][5]=0;
x[0][6]=0;
}
else if (error>0 && error<0.15){
value1=(-100*error+15)/15;
value2=(100*error)/15;
x[0][0]=0;
x[0][1]=0;
x[0][2]=0;
x[0][3]=value1;
x[0][4]=value2;
x[0][5]=0;
x[0][6]=0;
}
else if (error>0.15 && error<0.5){
value1=(-100*error+50)/35;
value2=(100*error-15)/35;
x[0][0]=0;
x[0][1]=0;
x[0][2]=0;
x[0][3]=0;
x[0][4]=value1;
x[0][5]=value2;
```

```
x[0][6]=0;
}
else if (error>0.5 && error<1){
value1=(-100*error+80)/30;
value2=(100*error-50)/50;
x[0][0]=0;
x[0][1]=0;
x[0][2]=0;
x[0][3]=0;
x[0][4]=0;
x[0][5]=value1;
x[0][6]=value2;
}
else printf("error giriş değerine karşılık kural bulunamadı.");

//cerror girişi için üyelik fonksiyonları denklemleri

if (cerror>-1.5 && cerror<-0.75){
value3=(-100*cerror-75)/75;
value4=(100*cerror+120)/45;
x[1][0]=value3;
x[1][1]=value4;
x[1][2]=0;
x[1][3]=0;
x[1][4]=0;
x[1][5]=0;
x[1][6]=0;
}
else if (cerror>-0.75 && cerror<-0.225){
value3=(-100*cerror-22.5)/52.5;
value4=(100*cerror+75)/52.5;
x[1][0]=0;
x[1][1]=value3;
x[1][2]=value4;
x[1][3]=0;
x[1][4]=0;
x[1][5]=0;
x[1][6]=0;
}
else if (cerror>-0.225 && cerror<0){
value3=(-100*cerror)/22.5;
value4=(100*cerror+22.5)/22.5;
x[1][0]=0;
x[1][1]=0;
x[1][2]=value3;
x[1][3]=value4;
x[1][4]=0;
x[1][5]=0;
x[1][6]=0;
}
```

```
else if (cerror>0 && cerror<0.225){
value3=(-100*cerror+22.5)/22.5;
value4=(100*cerror)/22.5;
x[1][0]=0;
x[1][1]=0;
x[1][2]=0;
x[1][3]=value3;
x[1][4]=value4;
x[1][5]=0;
x[1][6]=0;
}
else if (cerror>0.225 && cerror<0.75){
value3=(-100*cerror+75)/52.5;
value4=(100*cerror-22.5)/52.5;
x[1][0]=0;
x[1][1]=0;
x[1][2]=0;
x[1][3]=0;
x[1][4]=value3;
x[1][5]=value4;
x[1][6]=0;
}
else if (cerror>0.75 && cerror<1.5){
value3=(-100*cerror+120)/45;
value4=(100*cerror-75)/75;
x[1][0]=0;
x[1][1]=0;
x[1][2]=0;
x[1][3]=0;
x[1][4]=0;
x[1][5]=value3;
x[1][6]=value4;
}
else printf("cerror giriş değerine karşılık kural bulunamadı.");


//minZ dizisinin hesaplanması

for(k=0;k<49;k++){
pp=input[0][k];
qq=input[1][k];
minz[k]=min(x[0][pp],x[1][qq]);
}

for(i=0;i<49;i++){
rr=output[2+i];
u[i]=max(rr,minz[i]);
}

//action ‡□k□Ÿ□ i‡in □yelik fonksiyonlar□ denklemleri
```

```
if (action>-1.5 && action<-0.8){
value5=(-100*action+80)/70;
value6=(100*action+150)/70;
}
if (action>-0.8 && action<-0.4){
value5=(-100*action-40)/40;
value6=(100*action+80)/40;
}
if (action>-0.4 && action<-0.1){
value5=(-100*action-10)/30;
value6=(100*action+40)/30;
}
if (action>-0.1 && action<0){
value5=(-100*action);
value6=(100*action+10)/10;
}
if (action>0 && action<0.1){
value5=(-100*action+10)/10;
value6=(100*action);
}
if (action>0.1 && action<0.4){
value5=(-100*action+40)/30;
value6=(100*action-10)/30;
}
if (action>0.4 && action<0.8){
value5=(-100*action+80)/40;
value6=(100*action-40)/40;
}
if (action>0.8 && action<1.5){
value5=(-100*action+150)/70;
value6=(100*action+80)/70;
}

else printf("action çıkış değerine karşılık kural bulunamadı.");

//Center of gravity metodu

for(j=0;j<9;j++){
pay=u[j]*action[j];
payda=payda+u[j];
}
actionsa=pay/payda;

/*Beta bileşeni için bulanık kontrol algoritması*/

      akiB1=(usb-isb*rs)-actionsb*akibtop;
      akibtop=akib2+akiB1*TS;
      akib2=akibtop;
      akifib=akibtop-sig*xss*isb;
```

```
        error1=0-akifib;
        cerror=error1-error;

/*error giriş değişkeni için üyelik fonksiyonlarının denklemleri*/

if (error>-1 && error<-0.5){
value1=(-100*error-50)/50;      // 1.üyelik fonk.
value2=(100*error+80)/30;       // 2.üyelik fonk
x[0][0]=value1;
x[0][1]=value2;
x[0][2]=0;
x[0][3]=0;
x[0][4]=0;
x[0][5]=0;
x[0][6]=0;
}
else if (error>-0.5 && error<-0.15){
value1=(-100*error-15)/35;
value2=(100*error+50)/35;
x[0][0]=0;
x[0][1]=value1;
x[0][2]=value2;
x[0][3]=0;
x[0][4]=0;
x[0][5]=0;
x[0][6]=0;
}
else if (error>-0.15 && error<0){
value1=(-100*error)/15;
value2=(100*error+15)/15;
x[0][0]=0;
x[0][1]=0;
x[0][2]=value1;
x[0][3]=value2;
x[0][4]=0;
x[0][5]=0;
x[0][6]=0;
}
else if (error>0 && error<0.15){
value1=(-100*error+15)/15;
value2=(100*error)/15;
x[0][0]=0;
x[0][1]=0;
x[0][2]=0;
x[0][3]=value1;
x[0][4]=value2;
x[0][5]=0;
x[0][6]=0;
```

```
}
else if (error>0.15 && error<0.5){
value1=(-100*error+50)/35;
value2=(100*error-15)/35;
x[0][0]=0;
x[0][1]=0;
x[0][2]=0;
x[0][3]=0;
x[0][4]=value1;
x[0][5]=value2;
x[0][6]=0;
}
else if (error>0.5 && error<1){
value1=(-100*error+80)/30;
value2=(100*error-50)/50;
x[0][0]=0;
x[0][1]=0;
x[0][2]=0;
x[0][3]=0;
x[0][4]=0;
x[0][5]=value1;
x[0][6]=value2;
}
else printf("error giriş değerine karşılık kural bulunamadı.");

//cerror girişi için üyelik fonksiyonları denklemleri

if (cerror>-1.5 && cerror<-0.75){
value3=(-100*cerror-75)/75;
value4=(100*cerror+120)/45;
x[1][0]=value3;
x[1][1]=value4;
x[1][2]=0;
x[1][3]=0;
x[1][4]=0;
x[1][5]=0;
x[1][6]=0;
}
else if (cerror>-0.75 && cerror<-0.225){
value3=(-100*cerror-22.5)/52.5;
value4=(100*cerror+75)/52.5;
x[1][0]=0;
x[1][1]=value3;
x[1][2]=value4;
x[1][3]=0;
x[1][4]=0;
x[1][5]=0;
x[1][6]=0;
}
else if (cerror>-0.225 && cerror<0){
```

```
value3=(-100*cerror)/22.5;
value4=(100*cerror+22.5)/22.5;
x[1][0]=0;
x[1][1]=0;
x[1][2]=value3;
x[1][3]=value4;
x[1][4]=0;
x[1][5]=0;
x[1][6]=0;
}
else if (cerror>0 && cerror<0.225){
value3=(-100*cerror+22.5)/22.5;
value4=(100*cerror)/22.5;
x[1][0]=0;
x[1][1]=0;
x[1][2]=0;
x[1][3]=value3;
x[1][4]=value4;
x[1][5]=0;
x[1][6]=0;
}
else if (cerror>0.225 && cerror<0.75){
value3=(-100*cerror+75)/52.5;
value4=(100*cerror-22.5)/52.5;
x[1][0]=0;
x[1][1]=0;
x[1][2]=0;
x[1][3]=0;
x[1][4]=value3;
x[1][5]=value4;
x[1][6]=0;
}
else if (cerror>0.75 && cerror<1.5){
value3=(-100*cerror+120)/45;
value4=(100*cerror-75)/75;
x[1][0]=0;
x[1][1]=0;
x[1][2]=0;
x[1][3]=0;
x[1][4]=0;
x[1][5]=value3;
x[1][6]=value4;
}
else printf("cerror giriş değerine karşılık kural bulunamadı.");


//minZ dizisinin hesaplanması

for(k=0;k<49;k++){
pp=input[0][k];
```

```
qq=input[1][k];
minz[k]=min(x[0][pp],x[1][qq]);
}

for(i=0;i<49;i++){
rr=output[2+i];
u[i]=max(rr,minz[i]);
}

//action çıkışı için üyelik fonksiyonları denklemleri

if (action>-1.5 && action<-0.8){
value5=(-100*action+80)/70;
value6=(100*action+150)/70;
}
if (action>-0.8 && action<-0.4){
value5=(-100*action-40)/40;
value6=(100*action+80)/40;
}
if (action>-0.4 && action<-0.1){
value5=(-100*action-10)/30;
value6=(100*action+40)/30;
}
if (action>-0.1 && action<0){
value5=(-100*action);
value6=(100*action+10)/10;
}
if (action>0 && action<0.1){
value5=(-100*action+10)/10;
value6=(100*action);
}
if (action>0.1 && action<0.4){
value5=(-100*action+40)/30;
value6=(100*action-10)/30;
}
if (action>0.4 && action<0.8){
value5=(-100*action+80)/40;
value6=(100*action-40)/40;
}
if (action>0.8 && action<1.5){
value5=(-100*action+150)/70;
value6=(100*action+80)/70;
}

else printf("action çıkış değerine karşılık kural bulunamadı.");

//Center of gravity metodu

for(j=0;j<9;j++){
pay=u[j]*actiont[j];
```

```
payda=payda+u[j];
}
actionsb=pay/payda;


        fi=sqrt(fialfa*fialfa+fibeta*fibeta);
        costet=fialfa/fi;
        sintet=fibeta/fi;
        hiz=(((fialfa1*fibeta)-(fibeta1*fialfa))/(fi*fi*TS))-10*hiz0;
        hiz0=(hiz0+hiz*TS);
        hiz1=48*(hiz0+wk);


t=t+TS;
s=s+1;
        if (s>25 & t>1)
        {
        s=0;
        er=rpmref-hiz1;
        hata=hata+er*TS*25;
        isqref=hata*rpmki+er*rpmkp;
        if (isqref>3) isqref=3;
        if (isqref<-3) isqref=-3;
        }




}

        void isr_t1()
        {
        begin_isr_t1(*error);
        say=ds1102_inc(2);
        rpm=6578947*say;
        ds1102_inc_clear_counter(2);
        end_isr_t1();
        }


//minimum alan fonksiyon

        int min(int value1, int value2){
                return ( (value1 < value2) ? value1 : value2);
                }

//maksimum alan fonksiyon

        int max(int value1, int value2){
                return ( (value1 > value2) ? value1 : value2);
```

```
}

void main()

{

//Kural tablosu

input[0][0]=0;
input[0][1]=0;
input[0][2]=0;
input[0][3]=0;
input[0][4]=0;
input[0][5]=0;
input[0][6]=0;
input[0][7]=1;
input[0][8]=1;
input[0][9]=1;
input[0][10]=1;
input[0][11]=1;
input[0][12]=1;
input[0][13]=1;
input[0][14]=2;
input[0][15]=2;
input[0][16]=2;
input[0][17]=2;
input[0][18]=2;
input[0][19]=2;
input[0][20]=2;
input[0][21]=3;
input[0][22]=3;
input[0][23]=3;
input[0][24]=3;
input[0][25]=3;
input[0][26]=3;
input[0][27]=3;
input[0][28]=4;
input[0][29]=4;
input[0][30]=4;
input[0][31]=4;
input[0][32]=4;
input[0][33]=4;
input[0][34]=4;
input[0][35]=5;
input[0][36]=5;
input[0][37]=5;
input[0][38]=5;
input[0][39]=5;
```

```
input[0][40]=5;
input[0][41]=5;
input[0][42]=6;
input[0][43]=6;
input[0][44]=6;
input[0][45]=6;
input[0][46]=6;
input[0][47]=6;
input[0][48]=6;

input[1][0]=0;
input[1][1]=1;
input[1][2]=2;
input[1][3]=3;
input[1][4]=4;
input[1][5]=5;
input[1][6]=6;
input[1][7]=0;
input[1][8]=1;
input[1][9]=2;
input[1][10]=3;
input[1][11]=4;
input[1][12]=5;
input[1][13]=6;
input[1][14]=0;
input[1][15]=1;
input[1][16]=2;
input[1][17]=3;
input[1][18]=4;
input[1][19]=5;
input[1][20]=6;
input[1][21]=0;
input[1][22]=1;
input[1][23]=2;
input[1][24]=3;
input[1][25]=4;
input[1][26]=5;
input[1][27]=6;
input[1][28]=0;
input[1][29]=1;
input[1][30]=2;
input[1][31]=3;
input[1][32]=4;
input[1][33]=5;
input[1][34]=6;
input[1][35]=0;
input[1][36]=1;
input[1][37]=2;
input[1][38]=3;
input[1][39]=4;
```

```
input[1][40]=5;
input[1][41]=6;
input[1][42]=0;
input[1][43]=1;
input[1][44]=2;
input[1][45]=3;
input[1][46]=4;
input[1][47]=5;
input[1][48]=6;


output[0]=0;
output[1]=0;
output[2]=0;
output[3]=1;
output[4]=2;
output[5]=3;
output[6]=4;
output[7]=0;
output[8]=0;
output[9]=1;
output[10]=2;
output[11]=3;
output[12]=4;
output[13]=5;
output[14]=0;
output[15]=1;
output[16]=3;
output[17]=3;
output[18]=4;
output[19]=5;
output[20]=6;
output[21]=1;
output[22]=2;
output[23]=3;
output[24]=4;
output[25]=5;
output[26]=6;
output[27]=7;
output[28]=2;
output[29]=3;
output[30]=4;
output[31]=5;
output[32]=6;
output[33]=7;
output[34]=8;
output[35]=3;
output[36]=4;
output[37]=5;
output[38]=6;
```

```
output[39]=7;
output[40]=8;
output[41]=8;
output[42]=4;
output[43]=5;
output[44]=6;
output[45]=7;
output[46]=8;
output[47]=8;
output[48]=8;

actiont[0]=-1.2;
actiont[1]=-0.8;
actiont[2]=-0.4;
actiont[3]=-0.1;
actiont[4]=0;
actiont[5]=0.1;
actiont[6]=0.4;
actiont[7]=0.8;
actiont[8]=1.2;


  *error = NO_ERROR;
  init();
  start_isr_t0(TS);
  start_isr_t1(TS1);
  ds1102_inc_clear_counter(2);
  ds1102_p14_pin_io_init(11100000000);
  isqref=0;
  while(1)
  {
  }
}
```