İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

SİNCAP KAFESLİ ASENKRON MAKİNADA YAPAY SİNİR AĞLARI İLE ROTOR AKISI YÖNLENDİRİLMİŞ VEKTÖR DENETİMİ

YÜKSEK LİSANS TEZİ

Müh. Sabri Volkan KEMAL
504981021

Tezin Enstitüye Verildiği Tarih :24 Ocak 2001
Tezin Savunulduğu Tarih : 16 Şubat 2001

Tez Danışmanı : Prof. Dr. M. Emin TACER
Diğer Jüri Üyeleri Prof. Dr. Nejat TUNCAY
Doç. Dr. Ece Olcay GÜNEŞ

ŞUBAT 2001

ÖNSÖZ

Bilim ve teknolojinin hızla geliştiği günümüzde, gelişen teknolojileri izlemek ve uygulamak daha kaliteli ve verimli üretimin önkoşulu olmuştur. Özellikle son yıllarda önemi farkedilen ve uygulanabilirlik açısından önemli bir yol alan yapay sinir ağlarını, asenkron makinanın vektör denetimi uygulamasında kullanarak, geliştirdiğim algoritmaları ve simülasyon sonuçlarını ayrıntılarıyla vermeye çalıştım.

Çalışmalarım sırasında bana yardımcı olan tez danışmanım Sayın Prof.Dr. Emin Tacer' e ve araştırma görevlisi Sayın Yük.Müh. Saffet Altay'a teşekkür ederim.

Şubat 2001 Sabri Volkan Kemal

İÇİNDEKİLER Sayfa No

ŞEKİL LİSTESİ v
SEMBOL LİSTESİ vi
ÖZET viii
SUMMARY ix

1. GİRİŞ 1

1.1 Yapay Sinir Ağları ile Vektörel Kontrol 1

2. ASENKRON MAKİNA MODELLERİ 3

2.1 Asenkron Makine Modeli 3

2.2 Asenkron Makine Üç Faz Modeli 3

2.3 Dik Eksenli (Quadrature - Phase) Bilezikli Model 5

2.4 Dik Eksenli (Quadrature - Phase) Kollektörlü Model 7

2.5 Genel Eksen Takımında Uzay Fazörleri ile Modelleme 9

2.6 Sincap Kafesli Asenkron Motorun D-Q Modeli 11

3. VEKTÖR DENETİMİ 14

3.1 Asenkron Makinada Vektör Denetimi 14

3.2 Gerilim Aradevreli Eviriciden Beslenen Asenkron Makinada Rotor Akısı Yönlendirilmiş Vektör Denetimi 15

3.2.1 Rotor akısı yönlendirilmiş eksen takımında stator gerilimi eşitlikleri 15
3.2.2 Dekuplaj (ayrıştırma) devreleri 16
3.2.3 Rotor akı modeli 17

3.3 Gerilim Aradevreli Eviriciden Beslenen Asenkron Makinada Rotor Akısı Yönlendirilmiş Vektör Denetimi Simülasyon Modeli 20

4. YAPAY SİNİR AĞLARI 26

4.1 Yapay Sinir Ağları Üzerine Temel Bilgiler 26

4.1.1 YSA tanımı 26
4.1.2 Sinaps tanımı 26
4.1.3 YSA modelleri 26
4.1.4 Öğrenme açısından sınıflama 27
4.1.5 Neural network uygulaması 27
4.1.6 Nöron tipleri ve katmanlı ağlar 28
4.1.7 Yapay sinir ağları 30
4.1.8 Vektör ve matris notasyonu 32
4.1.9 Doğrusal birleştirici yapay sinir ağı 34
4.1.10 Widrow' un adaline modeli 35
4.1.11 Madaline 37

4.2 Geriye Yayılım Eğitim Algoritmaları 37

4.2.1 Geriye yayılım ile eğitim 37
4.2.2 Widrow-Hoff delta öğrenme kuralı 40
4.2.3 Çok katmanlı yapay sinir ağlarında geriye yayılım ile eğitim 43
4.2.4 Çıkış katmanındaki hücreler için ağırlıkların hesaplanması 45
4.2.5 Orata katmandaki hücreler için ağırlıkların hesaplanması 47

4.3 Yapay Sinir Ağları ile Sistem Tanıma 51

4.3.1 Referans model gösterilimi 51
4.3.2 Sistem tanıma 51
4.3.3 Rotor akısı yönlendirilmiş vektör denetimi için neural tabanlı akı modeli algılayıcıları 54
4.3.4 Rotor akısı yönlendirilmiş vektör denetimi için neural tabanlı rotor hızı algılayıcısı 56
4.3.5 Sincap kafesli asenkron makinada neural tabanlı akı ve hız algılayıcıları ile vektör denetimi 57

5. ADAPTİV KONTROL 59

5.1 Rotor Parametresi Değişimine Adapte Vektör Denetimi 59

5.1.1 Rotor zaman sabitinin belirlenmesi 59
5.1.2 Neural tabanlı algılayıcılar ile parametre adaptasyonlu vektör denetimi 60
5.1.3 Neural tabanlı algılayıcılar ile parametre adaptasyonlu vektör denetimi simülasyon modeli 62

6. SİMÜLASYON SONUÇLARI 63

6.1 Makine Modeline İlişkin Simülasyon Sonuçları 64

6.2 Rotor Akısı Yönlendirilmiş Vektör Kontrolüne İlişkin Simülasyon Sonuçları 66

6.3 Rotor Mıknatıslanma Akımı Genliğini Tahmin Eden Yapay Sinir Ağının Off-line Eğitimi ve Simülasyon Sonuçları 69

6.4 Rotor Mıknatıslanma Akımı Açısını Tahmin Eden Yapay Sinir Ağının Off-line Eğitimi ve Simülasyon Sonuçları 71

6.5 Rotor Hızını Tahmin Eden Yapay Sinir Ağının Off-line Eğitimi ve Simülasyon Sonuçları 72

6.6 Sincap Kafesli Asenkron Makinada Neural Tabanlı Akı Algılayıcısı ile Rotor Akısı Yönlendirilmiş Vektör Denetimi Simülasyon Sonuçları 73

6.7 Sincap Kafesli Asenkron Makinada Neural Tabanlı Hız Algılayıcısı ile Rotor Akısı Yönlendirilmiş Vektör Denetimi Simülasyon Sonuçları 75

6.8 Rotor Parametre Adaptasyonlu Neural Tabanlı Rotor Akısı Yönlendirilmiş Vektör Denetimi Simülasyon Sonuçları 76

7. SONUÇLAR VE ÖNERİLER 80

KAYNAKLAR 83

EKLER 85

ÖZGEÇMİŞ 111

ŞEKİL LİSTESİ

		Sayfa No
Şekil 2.1	: Simetrik üç fazlı asenkron makinanın enine kesiti	4
Şekil 2.2	: Quadrature-phase bilezikli modelin şematik gösterilimi	6
Şekil 2.3	: Dik eksenli kollektörlü modelin şematik gösterilimi	8
Şekil 2.4	: Genel eksen takımında tanımlı stator akımı uzay fazörü	9
Şekil 2.5	: Genel eksen takımında tanımlı rotor akımı uzay fazörü	10
Şekil 3.1	: Rotor akısı yönlendirilmiş eksen takımında akı modeli	18
Şekil 3.2	: Rotor akısı yönlendirilmiş vektör denetimi blok şeması	21
Şekil 4.1	: Doğrusal olmayan sinir hücresi modeli	28
Şekil 4.2	: Sinir hücreleri için tanımlanan bazı aktivasyon fonksiyonları	29
Şekil 4.3	: İleri yönde beslemeli temel sinir ağı yapısı	31
Şekil 4.4	: Basit ileri yönde beslemeli sinir ağı	32
Şekil 4.5	: Orta katman üzerinden kesilmiş olan ağın alt kısmı	33
Şekil 4.6	: Orta katmanın üzerinden kesilmiş ağın üst kısmı	34
Şekil 4.7	: Adaline işlem elemanının diyagramı	35
Şekil 4.8	: Madaline yapısına ilişkin diyagram	37
Şekil 4.9	: Sinir Hücresi	38
Şekil 4.10	: Geriye yayılım için aktivasyon fonksiyonları	40
Şekil 4.11	: Aktivasyon fonksiyonu içermeyen sinir hücresi	40
Şekil 4.12	: Widrow-Hoff eğitimi boyunca e^2 nin minimizasyonu	42
Şekil 4.13	: Delta kuralının geometrik ifadesi	43
Şekil 4.14	: Çok tabakalı sinir ağına uygulanan geriye yayılım algoritması	44
Şekil 4.15	: Çıkış katmanı için geriye yayılım ile ağırlık hesabı diyagramı	45
Şekil 4.16	: Orta katman için geriye yayılım ile ağırlık hesabı diyagramı	48
Şekil 4.17	: Paralel tanımlama modeli	52
Şekil 4.18	: Seri-Paralel Tanımlama Modeli	53
Şekil 4.19	: Neural tabanlı gözleyicinin eğitimi amacıyla kullanılan blok şema	55
Şekil 4.20	: Neural tabanlı hız algılayıcısının eğitimi kullanılan blok şema	57
Şekil 5.1	: Neural tabanlı algılayıcılar ile adaptif vektör denetimi	61

SEMBOL LİSTESİ

u_{sA}, u_{sB}, u_{sC} : Stator A, B, C faz gerilimleri
i_{sA}, i_{sB}, i_{sC} : Stator A, B, C faz akımları
ψ_{sA}, ψ_{sB}, ψ_{sC} : Stator A, B, C fazı halkalanma akıları
u_{ra}, u_{rb}, u_{rc} : Rotor a, b, c faz gerilimleri
i_{ra}, i_{rb}, i_{rc} : Rotor a, b, c faz akımları
ψ_{ra}, ψ_{rb}, ψ_{rc} : Rotor a, b, c fazı halkalanma akıları
t : Zaman
R_s : Stator direnci
R_r : Rotor direnci
θ_r : Rotor açısı
$\overline{L}_s$: Bir fazın özindüktansı
L_{sl} : Kaçak indüktans
L_{sm} : Mıknatıslanma indüktansı
$\overline{M}_s$: Ortak indüktans
L_s : Toplam stator indüktansı
M_e, M_l : İç döndürme ve yük momentleri
J : Eylemsizlik momenti
B : Sürtünme katsayısı
ω_r : Rotor açısal hızı
ω_{sl} : Kayma açısal hızı
$\overline{M}_{sr}$: Stator ve rotor sargıları arasındaki ortak indüktans
u_{sD}, u_{sQ} : Durağan eksen takımında tanımlı stator gerilimi bileşenleri
i_{sD}, i_{sQ} : Durağan eksen takımında tanımlı stator akımı bileşenleri
$u_{r\alpha}$, $u_{r\beta}$: Rotor eksen takımında tanımlı rotor gerilimi bileşenleri
$i_{r\alpha}$, $i_{r\beta}$: Rotor eksen takımında tanımlı rotor akımı bileşenleri
u_{rd}, u_{rq} : Durağan eksen takımında tanımlı rotor gerilimi bileşenleri
i_{rd}, i_{rq} : Durağan eksen takımında tanımlı rotor akımı bileşenleri
L_m : Mıknatıslanma indüktansı
N_{se}, N_{re} : Stator ve rotor etkin sarım sayıları
a : Uzaysal öteleme operatörü ($+120^\circ$)
$\bar{i}_s$, $\bar{i}_{sg}$: Durağan ve genel eksen takımlarında tanımlı stator akımı uzay fazörü
$\bar{i}_r$, $\bar{i}_{rg}$: Rotor ve genel eksen takımlarında tanımlı rotor akımı uzay fazörü
i_{sx}, i_{sy}, i_{rx}, i_{ry} : Genel eksen takımında tanımlı stator ve rotor akımı dik bileşenleri
θ_g : Durağan ve genel eksen takımları arasındaki açı
$\overline{U}_{sg}$, $\overline{U}_{rg}$: Genel eksen takımında tanımlı stator ve rotor gerilimi uzay fazörleri
ω_g : Genel eksen takımının açısal hızı
P : Çift kutup sayısı
$\bar{i}_{mr}$: Rotor mıknatıslanma akımı uzay fazörü

σ_{rl} : Rotor kaçak faktörü

L_{rl} : Rotor kaçak indüktansı

L_s' : Stator geçici indüktansı

T_r : Rotor zaman sabiti

T_r' : Rotor geçici zaman sabiti

i_{mrD}, i_{mrQ} : Durağan eksen takımında tanımlı mıknatıslanma akımı bileşenleri

σ : Kaçak faktörü

ρ_r : Rotor mıknatıslanma akımının açısı

$\bar{i}_{s\psi r}$, $\bar{i}_{r\psi r}$, $\bar{u}_{s\psi r}$: Rotor akısı yönlendirilmiş eksen takımında tanımlı stator, rotor akımları ve stator gerilimi

u_{dx}, u_{dy} : Ayrıştırma gerilimi bileşenleri

u_{sxref}, u_{syref} : Genel eksen takımında tanımlı referans stator gerilimi bileşenleri

ω_{mr} : Rotor halkalanma akısının açısal hızı

$\bar{\psi}_{r\psi r}$: Kendi eksen takımında tanımlı rotor halkalanma akısı uzay fazörü

kp1...kp5 : PI kontrolörlerin oransal katsayıları

kı1...kı5 : PI kontrolörlerin integral alıcı katsayıları

$|\bar{i}_{mref}|$: Referans mıknatıslanma akımının genliği

i_{sxref}, i_{syref} : Genel eksen takımında tanımlı referans stator akımı bileşenleri

ω_{ref} : Referans rotor açısal hızı

T_s' : Stator geçici zaman sabiti

w_{kj} : j. girişle k. nöron arasındaki ağırlık

w_{ji} : i. girişle j. nöron arasındaki ağırlık

x_j : j. giriş

u_k : Girişleri ağırlıklı toplamı

T : Aktivasyon fonksiyonlarının eşik değeri

$y_k(n)$: n. adımdaki çıkış

$d_k(n)$: n. adımdaki istenen çıkış

$e_k(n)$: n. adımdaki çıkış hatası

$\varepsilon_k(n)$: n. adımdaki karesel hata

η : Öğrenme sabiti

$\delta_j(n)$: n. adımdaki j. nöronun yerel gradyeni

y_p : Sistemin tahmin edilen çıkışı

α_0, α_1 : Gecikme elemanlarının katsayıları

u(k) : k. adımdaki giriş

dt : Adım aralığı

Nimr : Mıknatıslanma akımının normalize değeri

Nisx : $İ_{sx}$ akımının normalize değeri

Nisy : $İ_{sy}$ akımının normalize değeri

Nwsl : ω_{sl} kayma açısal hızının normalize değeri

Nwr : ω_r açısal rotor hızının normalize değeri

Nme : Elektriksel döndürme momentini normalize değeri

SİNCAP KAFESLİ ASENKRON MAKİNADA YAPAY SİNİR AĞLARI İLE ROTOR AKISI YÖNLENDİRİLMİŞ VEKTÖR DENETİMİ

ÖZET

Bu tez kapsamında, asenkron makinada yapay sinir ağları ile alan yönlendirmeli vektör denetimi amaçlanmıştır. Ayrıca gerilim aradevreli bir eviriciden beslenen asenkron makinada rotor hızı YSA tabanlı bir algılayıcı ile tahmin edilmeye çalışılmıştır.

Vektör denetiminde kullanılan kontrol bloğu içindeki akı modelinin çıkışları için iki adet çok katmanlı neural algılayıcı kullanılarak durum kestirimi yapılmaya çalışılmış; algılayıcıları eğitimi sürecinde geriye yayılım algoritmaları kullanılmıştır. Eğitim işlemine başlarken, sinir hücreleri arasındaki ağırlıklar başlangıçta rastgele olarak seçilmiş ve gerçek çıkış ile tahmin edilen çıkış arasındaki hata fonksiyonu tolere edilebilir bir değere çekilinceye kadar ağırlıklar değiştirilerek eğitim tamamlanmıştır. Bu tezde akı modeli yerine kullanılan iki adet yapay sinir ağının hücre sayısı konfigürasyonu 2-10-10-1 şeklindedir.

Üç faz asenkron makinanın YSA tabanlı akı ve hız algılayıcıları kullanılarak gerçekleştirilen kontrol çalışmasında, makinanın hem geçici hem de sürekli haldeki dinamik sistem cevabına ilişkin eğrileri simülasyon sonuçlarında sunulmuştur.

Tezin son kısmında, makine yükte iken sıcaklığa bağlı olarak rotor direncinin değişimini gözlemleyen bir algoritma geliştirilmiş ve zaman sabitinin değişimi de simülasyon sonuçlarında verilmiştir. Değişen rotor zaman sabiti bilgisine göre MRAC teorisi çerçevesinde yapay sinir ağlarının ağırlık matrisleri, dinamik sistem simülasyonu ile birlikte eş zamanlı olarak güncellenmiş ve rotor zaman sabitindeki değişimin sistemin dinamik cevabı üzerindeki bozucu etkisi kompanze edilmeye çalışılmıştır.

ARTIFICIAL NEURAL NETWORKS BASED IDENTIFICATION AND CONTROL APPROACH FOR THE FIELD ORIENTED INDUCTION MOTOR DRIVE

SUMMARY

In this thesis a high performance control of field oriented induction motor using artificial neural networks (ANNs) have been studied. An ANN-based speed observer in a vector controlled voltage source induction motor drive employing rotor flux oriented control has also been considered.

The mathematical model of the field oriented induction motor has been simulated by two aotuassociative neural networks. The training technique which was used in the thesis is the error back-propagation technique. A trained ANN has the ability to learn and effectiveness is well established. Initially the weights in the neural network have been randomly implemented, and therefore the output signals of the neural network would not be equal to the desired output. However, during the training, the actual output signals heve been compared to the desired outputs signals, and the weights have been adjusted until the output error becomes smaller than a preset threshold value. In the thesis two ANNs have been trained with a configuration of 2-10-10-1.

In conclusion, the ANN-based transient and steady state performance of three phase speed sensorless induction motor have been proposed by using back-propagation training method. Simulation results obtained by the ANN for the transient speed of the loaded machine have been presented.

At the last part of this thesis, an algorithm which identifies the secondary resistance, consequently the rotor time constant online is developed. The motor operating condition for secondary resistance identification, the stable identifier organization and the simulation results are presented. The algorithm is based on the theory of model reference adaptive control (MRAC). The proposed algorithm stably identifies the secondary resistance under load which is proportional rotor speed when a sinusoidal signal is injected into the primary windings of the induction motor. The vector controller adopting this algorithm controls motor torque and the rotor magnetizing current accurately under any load and any speed.

1. GİRİŞ

1.1 Yapay Sinir Ağları İle Vektörel Kontrol

Bilindiği gibi doğru akım makinası özellikle de serbest uyarmalı doğru akım makinası kontrolü en kolay makinadır. Çünkü akım ve moment, besleme ve uyarma sargıları tarafından sağlanır. Aralarında hiçbir elektriksel bağlantı sözkonusu değildir. A.C. makinada uygulanmaya çalışılan açı ve genlik kontrolünün amacı da, serbest uyarmalı doğru akım makinasının bu üstün kontrol özelliklerini sağlayabilmektir. Bu tip bir kontrol, stator akımını akı ve moment indükleyen birbirinden bağımsız iki bileşene ayrıştırılmasıyla sağlanabilir. Ayrıştırma işlemi için gerekli olan rotor mıknatıslanma akımının genliği ve açısını veren bir akı modeli oluşturulmalıdır. Bu çalışma kapsamında, akı modelinin çıkışları olan mıknatıslanma akımı genliği ve açısını tahmin eden 2 adet ileri yönlü ve çok katmanlı YSA modellenmiştir. Benzer şekilde rotor hızı da çok katmanlı bir algılayıcı tipinde modellenmiştir. Dolayısıyla proses içinde akı modelinin kullanımına ve rotordan hız geribeslemesi yapılmasına ihtiyaç duyulmayacaktır. Yapay sinir ağları ile bir sistemin denetlenmesi veya bir kısmının tanıtılması mümkündür; ancak yapay sinir ağları uzun zaman alan bir eğitim sürecinden sonra kontrol sistemine dahil edilebilmiştir.

Rotor akısı yönlendirmeli kontrolun dezavantajlarından biri de rotor parametrelerinin değişimine duyarlı olmasıdır. Makinanın yükte çalışması sırasında zamanla sıcaklığa bağlı olarak rotor direncinde artış gözlenir. Bu değişimi ihmal ederek yapılacak bir kontrolün sağlıklı bir yaklaşım olduğu söylenemez. Dolayısıyla bu çalışmanın ilerleyen bölümlerinde rotor parametresindeki(R_r) değişiminin, sistemin dinamik davranışı üzerindeki bozucu etkisini kompanze etmek amacıyla , rotor zaman sabiti belirli periyodik aralıklarda güncellenerek, sistemin içerdiği neural tabanlı algılayıcılar on-line eğitime tabi tutulmuştur.

Kullanılan asenkron makine sincap kafeslidir ve gerilim aradevreli eviriciden beslenmektedir. Matematiksel model literatürdeki klasik asenkron makine modelidir ve non-lineer diferansiyel denklemlerin çözümünde nümerik integrasyon yöntemi kullanılmıştır.

2. ASENKRON MAKİNA MODELLERİ

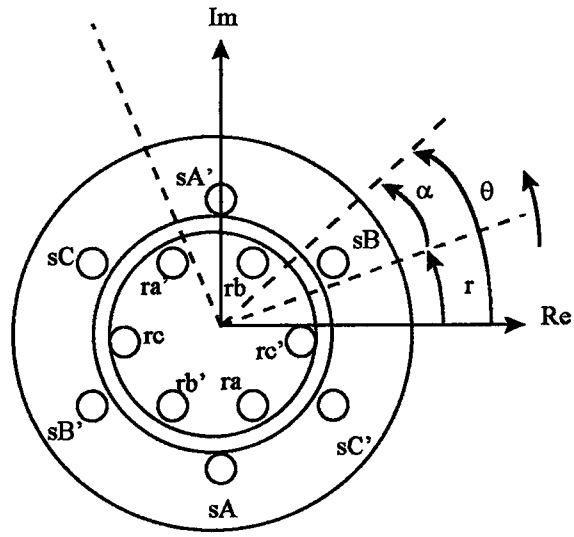
2.1 Asenkron Makine Modeli

Elektrik makinaları içinde asenkron makinanın özel bir yeri vardır. Bu çalışmada makinanın elektriksel ve mekanik yanına ilişkin diferansiyel denklemlerle dinamik sistemin matematiksel modeli oluşturulmuştur. Model kurulurken, tasarım aşamasında çok fazla tolerans gösterilmeyen veya gözardı edilmeyen bazı hususlar, bu çalışmada hem modeli basitleştirmek hem de mikroişlemcinin hesaplama süresini kısaltmak amacıyla bazı varsayımlarda bulunularak ihmal edilmiştir. Kontrol amaçlı bu varsayımlar şu şekilde özetlenebilir :

- Stator ve rotor sargı ya da çubuklarının aynı nitelikte iletkenlerden yapılmış olması ve sargılar arasında 120^0 lik elektriksel açı ile simetrinin bulunması.
- Sinüsoidal bir alan dağılımı sağlayacak şekilde düzgün bir hava aralığının olması
- Rotor ile statordaki diş ve olukarın elektriksel etkilerinin ihmal edilmesi.
- Rotor ve statorda magnetik malzemelerin magnetik geçirgenliklerinin sonsuz büyük olduğu varsayımı ile doyma etkisinin ihmal edilmesi.
- Demir kayıplarının olmadığı varsayılmıştır.[1]

2.2 Asenkron Makina Üç Faz Modeli

Asenkron makinanın üç faz modelinde yukarıdaki varsayımları da gözönüne alarak ilgili rotor ve stator değişkenleri kendi eksen takımlarında tanımlanmıştır.[1] Aşağıda üç fazlı bir asenkron makinanın rotor ve stator sargılarıyla birlikte enine kesiti verilmiştir.



Şekil 2.1 Simetrik üç fazlı asenkron makinanın enine kesiti

L_s, L_r, M_s ve $M_{s,r}$ sırasıyla stator, rotor indüktansı, stator sargıları arasındaki kuplaj ve stator ile rotor sargıları arasındaki ortak indüktanslar olmak üzere, durağan eksen takımındaki gerilim denklemleri ile rotor gerilim denklemleri aşağıdaki gibi ifade edilebilir.

$$U_{sA}(t) = R_s.i_{sA}(t) + d\Psi_{sA}(t)/dt \quad (2.1)$$

$$U_{sB}(t) = R_s.i_{sB}(t) + d\Psi_{sB}(t)/dt \quad (2.2)$$

$$U_{sC}(t) = R_s.i_{sC}(t) + d\Psi_{sC}(t)/dt \quad (2.3)$$

$$U_{rA}(t) = R_r.i_{rA}(t) + d\Psi_{rA}(t)/dt \quad (2.4)$$

$$U_{rB}(t) = R_r.i_{rB}(t) + d\Psi_{rB}(t)/dt \quad (2.5)$$

$$U_{rC}(t) = R_r.i_{rC}(t) + d\Psi_{rC}(t)/dt \quad (2.6)$$

Ψ_{sA}, Ψ_{sB}, Ψ_{sC} stator halkalanma akılarıdır. Hem rotor hem de stator halkalanma akıları dikkate alınarak yukarıdaki diferansiyel denklemler aşağıdaki matrissel formda yazılabilir. p=d/dt türev operatörü, $\theta=\theta_r$, $\theta_1=\theta_r+2\pi/3$ ve $\theta_2=\theta_r+4\pi/3$ olarak tanımlanmıştır.

$$\begin{bmatrix} U_{sA} \\ U_{sB} \\ U_{sC} \\ U_{rA} \\ U_{rB} \\ U_{rC} \end{bmatrix} = \begin{bmatrix} R_s+p.\overline{L}_s & p.\overline{M}_s & p.\overline{M}_s & p.\overline{M}_{s,r}\cos\theta & p.\overline{M}_{s,r}\cos\theta_1 & p.\overline{M}_{s,r}\cos\theta_2 \\ p.\overline{M}_s & R_s+p.\overline{L}_s & p.\overline{M}_s & p.\overline{M}_{s,r}\cos\theta_2 & p.\overline{M}_{s,r}\cos\theta & p.\overline{M}_{s,r}\cos\theta_1 \\ p.\overline{M}_s & p.\overline{M}_s & Rs+p.\overline{L}_s & p.\overline{M}_{s,r}\cos\theta_1 & p.\overline{M}_{s,r}\cos\theta_2 & p.\overline{M}_{s,r}\cos\theta \\ p.\overline{M}_{s,r}\cos\theta & p.\overline{M}_{s,r}\cos\theta_2 & p.\overline{M}_{s,r}\cos\theta_1 & R_r+p.\overline{L}_r & p.\overline{M}_r & p.\overline{M}_r \\ p.\overline{M}_{s,r}\cos\theta_1 & p.\overline{M}_{s,r}\cos\theta & p.\overline{M}_{s,r}\cos\theta_2 & p.\overline{M}_r & R_r+p.\overline{L}_r & p.\overline{M}_r \\ p.\overline{M}_{s,r}\cos\theta_2 & p.\overline{M}_{s,r}\cos\theta_1 & p.\overline{M}_{s,r}\cos\theta & p.\overline{M}_r & p.\overline{M}_r & R_r+p.\overline{L}_r \end{bmatrix} * \begin{bmatrix} i_{sA} \\ i_{sB} \\ i_{sC} \\ i_{rA} \\ i_{rB} \\ i_{rC} \end{bmatrix}$$

(2.7)

Bir fazın stator indüktansı $\overline{L}_s$ kaçak indüktans, L_{sl} ile mıknatıslanma indüktansı L_{sm} nin toplamına eşittir. Stator sargıları arasındaki kuplaj indüktans $\overline{M}_s$ ise mıknatıslanma indüktansı ile orantılıdır.

$$\overline{M}_s = L_{sm}\cos(2\pi/3) = -L_{sm}/2 \tag{2.8}$$

$$L_s = \overline{L}_s - \overline{M}_{s,r} = L_{sl} + \frac{3}{2}L_{sm} \tag{2.9}$$

Asenkron makinanın mekanik kısmına ilişkin denklemler ise (2.10) ve (2.11) de ki gibi ifade edilebilir.

$$M_e - M_l = J\frac{d\omega_r}{dt} + B.\omega_r \tag{2.10}$$

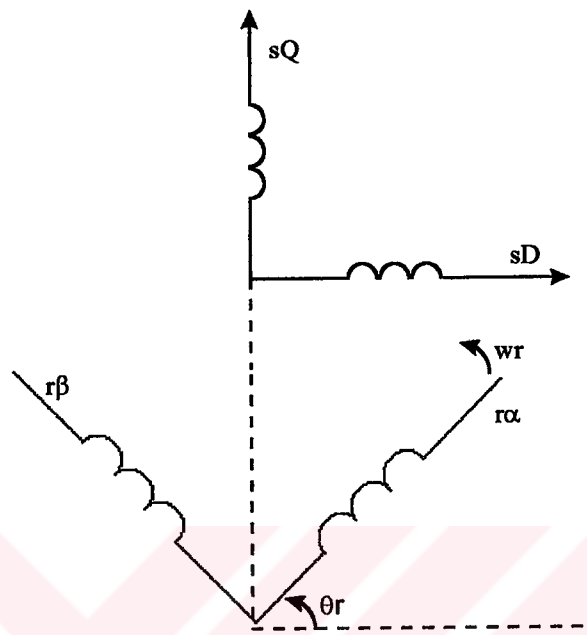
$$M_e = \frac{1}{2}\begin{bmatrix} I_s^T & I_r^T \end{bmatrix} * \begin{bmatrix} 0 & \dfrac{\partial\overline{M}_{s,r}}{\partial\theta} \\ \dfrac{\partial\overline{M}_{s,r}}{\partial\theta} & 0 \end{bmatrix} * \begin{bmatrix} I_s \\ I_r \end{bmatrix} \tag{2.11}$$

Hareket denkleminin iç döndürme momenti ifadesi elektriksel yanın hareket denklemine etkisini belirler.

2.3 Dik Eksenli (Quadrature Phase) Bilezikli Model

Düzgün hava aralıklı bir üç-faz asenkron makinanın bu modelini elde etmek için, ilk olarak kendi eksenlerinde tanımlı faz değişkenlerinden statora ilişkin olanlar sD, sQ durağan eksen takımına, rotora ilişkin olanlar ise ω_r hızıyla dönen rα, rβ eksen

takımına indirgenir. Birbirine dik iki eksen takımında tanımlama yapabilmek için sıfır bileşenlerinin olmaması gerekir; ki bu da nötr noktasının yalıtılmış olması anlamına gelir. [1]



Şekil 2.2 Quadrature-phase bilezikli modelin şematik gösterilimi

Dik eksenler üzerindeki stator gerilimleri ve akım bileşenleri aşağıdaki gibi ifade edilebilir. $a=e^{j.2\pi/3}$ uzaysal operatördür

$$U_{sD} = Re\left\{\frac{2}{3}\left[U_{sA}(t)+a.U_{sB}(t)+a^2.U_{sC}(t)\right]\right\} = \frac{2}{3}\left(U_{sA}-\frac{1}{2}U_{sB}-\frac{1}{2}U_{sC}\right) \qquad (2.12)$$

$$U_{sQ} = Im\left\{\frac{2}{3}\left[U_{sA}(t)+a.U_{sB}(t)+a^2.U_{sC}(t)\right]\right\} = \frac{1}{\sqrt{3}}\left(U_{sB}-U_{sC}\right) \qquad (2.13)$$

Benzer şekilde stator akımları ve rotor eksen takımındaki rotor gerilim bileşenleri de aşağıdaki gibi çıkarılabilir.

$$i_{sD} = \frac{2}{3}\left(i_{sA}-\frac{1}{2}i_{sB}-\frac{1}{2}i_{sC}\right) \qquad (2.14)$$

$$i_{sQ} = \frac{1}{\sqrt{3}}\left(i_{sB}-i_{sC}\right) \qquad (2.15)$$

$$U_{r\alpha} = \frac{2}{3}\left(U_{rA} - \frac{1}{2}U_{rB} - \frac{1}{2}U_{rC}\right) \quad (2.16)$$

$$U_{r\beta} = \frac{1}{\sqrt{3}}\left(U_{rB} - U_{rC}\right) \quad (2.17)$$

$$i_{r\alpha} = \frac{2}{3}\left(i_{rA} - \frac{1}{2}i_{rB} - \frac{1}{2}i_{rC}\right) \quad (2.18)$$

$$i_{r\beta} = \frac{1}{\sqrt{3}}\left(i_{rB} - i_{rC}\right) \quad (2.19)$$

Yukarıda eksen dönüşümleri ile yapılan indirgeme sonrası $L_m=3/2*M_{s,r}$ olmak üzere (2.7) de çıkarılmış olan matrissel bağıntı, aşağıdaki forma gelir. Bu ifadede rotor ve satator değişkenleri kendi referans eksen takımlarında ifade edilmiştir. Eğer indüktanslar sabit ise matris içindeki p türev operatörleri nedeniyle diagonal bir matris elde edilir; ancak eksen takımları arasındaki θ_r açısının zamana bağlı olması dolayısıyla diferansiyel denklemleri zamandan bağımsız kılmak mümkün değildir. Denklemlerin çözümünü kolaylaştırmak için trigonometrik terimlerden kurtulmak amacıyla yapılması gereken eksen takımı dönüşümleri ilerleyen bölümde ele alınmıştır.

$$\begin{bmatrix} U_{sD} \\ U_{sQ} \\ U_{r\alpha} \\ U_{r\beta} \end{bmatrix} = \begin{bmatrix} R_s + pL_s & 0 & pL_m\cos\theta_r & -pL_m\sin\theta_r \\ 0 & R_s + pL_s & pL_m\sin\theta_r & pL_m\cos\theta_r \\ pL_m\cos\theta_r & pL_m\sin\theta_r & R_r + pL_r & 0 \\ -pL_m\sin\theta_r & pL_m\cos\theta_r & 0 & R_r + pL_r \end{bmatrix} * \begin{bmatrix} i_{sD} \\ i_{sQ} \\ i_{r\alpha} \\ i_{r\beta} \end{bmatrix}$$

(2.20)

2.4 Dik Eksenli (Quadrature - Phase) Kollektörlü Model

Yukarıda elde edilen zamana bağlı diferansiyel denklemlerdeki parametreleri zamandan bağımsız kılmak amacıyla simetrili bileşenler, iki fazlı gerçek bileşenler veya Park dönüşümleri uygulanabilir. Park dönüşümleri ile dönüştürülen ve gerçek değişkenler arasında bağıl hız farkı söz konusudur. Park dönüşümlerinin tersi uygulanarak rotora ilişkin değişken büyüklükler durağan referans eksen takımına indirgenir. Rotor gerilimlerinin dönüşümü,

$$U_{r\alpha} = \cos\theta_r . U_{rd} + \sin\theta_r . U_{rq} \tag{2.21}$$

$$U_{r\beta} = -\sin\theta_r . U_{rd} + \cos\theta_r . U_{rq} \tag{2.22}$$

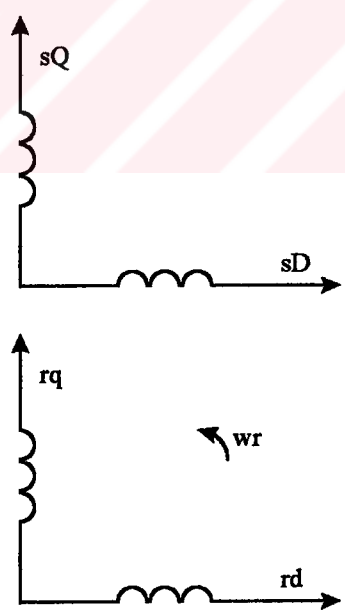
şeklinde bulunur. Benzer şekilde akımların dönüşümü de yazılabilir.

$$i_{r\alpha} = \cos\theta_r . i_{rd} + \sin\theta_r . i_{rq} \tag{2.23}$$

$$i_{r\beta} = -\sin\theta_r . i_{rd} + \cos\theta_r . i_{rq} \tag{2.24}$$

$$\begin{bmatrix} U_{sD} \\ U_{sQ} \\ U_{rd} \\ U_{rq} \end{bmatrix} = \begin{bmatrix} R_s + pL_s & 0 & pL_m & 0 \\ 0 & R_s + pL_s & 0 & pL_m \\ pL_m & \omega_r L_m & R_r + pL_r & \omega_r L_r \\ -\omega_r L_m & pL_m & -\omega_r L_r & R_r + pL_r \end{bmatrix} * \begin{bmatrix} i_{sD} \\ i_{sQ} \\ i_{rd} \\ i_{rq} \end{bmatrix} \tag{2.25}$$

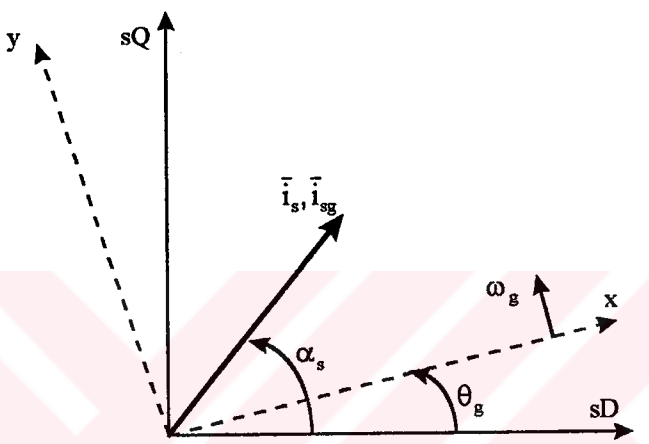
Denklemler yerine konulduğunda, kollektörlü modelin elde edilen gerilim eşitlikleri (2.25) de verilmiştir.



Şekil 2.3 Dik eksenli kollektörlü modelin şematik gösterilimi

2.5 Genel Eksen Takımında Uzay Fazörleri İle Modelleme

Bilindiği rotor eksen takımında, stator akımları uzay fazörleri $\bar{i}_s' = \bar{i}_s e^{-j.\theta_r}$ şeklinde, durağan eksen takımındaki rotor akımı uzay fazörleri de $\bar{i}_r' = \bar{i}_r e^{j.\theta_r}$ biçiminde tanımlanır. Bu modelde gerilim uzay fazörleri w_g açısal hızı ile dönen genel eksen takımında formüle edilir. θ_g açısı stator durağan eksen takımı ile genel eksen takımı arasındaki açıdır. [1]



Şekil 2.4 Genel eksen takımında tanımlı stator akımı uzay fazörü

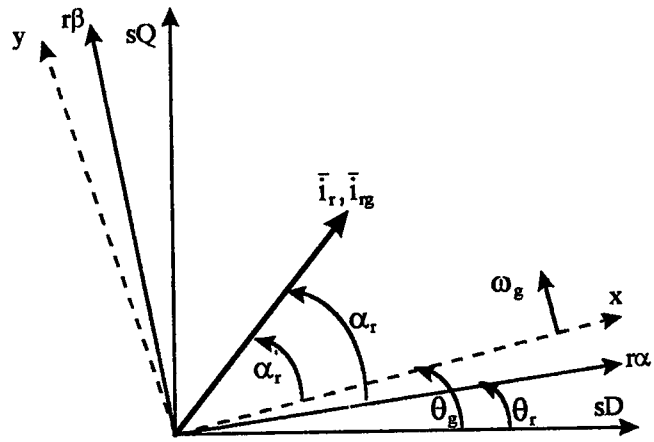
(2.26) nolu eşitlikten ve de yukarıdaki şekilden görüleceği üzere stator akımı uzay fazörünü genel eksen takımına indirgemek için uzay fazörünü $\alpha_s - \theta_g$ kadar ötelemek gerekir.

$$\bar{i}_{sg} = \bar{i}_s e^{-j.\theta_g} = |\bar{i}_s| . e^{\alpha_s} . e^{-j.\theta_g} = i_{sx} + j.i_{sy} \tag{2.26}$$

$$\bar{U}_{sg} = \bar{U}_s e^{-j.\theta_g} = u_{sx} + j.u_{sy} \tag{2.27}$$

$$\bar{\Psi}_{sg} = \bar{\Psi}_s e^{-j.\theta_g} = \psi_{sx} + j.\psi_{sy} \tag{2.28}$$

Rotor eksen takımındaki gerilim, akım ve halkalanma akısı uzay fazörlerinin, genel eksen takımına indirgenmesi için benzer şekilde fazör büyüklüklerin θ_g - θ_r kadar ötelenmesi gerekir.



Şekil 2.5 Genel eksen takımında tanımlı rotor akımı uzay fazörü

$$\bar{i}_{rg} = |\bar{i}_r| . e^{j.\alpha_r} . e^{-j.(\theta_g - \theta_r)} = \bar{i}_r . e^{-j.(\theta_g - \theta_r)} = i_{rx} + j.i_{ry} \tag{2.29}$$

$$\bar{U}_{rg} = \bar{U}_r . e^{-j.(\theta_g - \theta_r)} = u_{rx} + j.u_{ry} \tag{2.30}$$

$$\bar{\Psi}_{rg} = \bar{\Psi}_r . e^{-j.(\theta_g - \theta_r)} = \psi_{rx} + j.\psi_{ry} \tag{2.31}$$

Aşağıda denklem (2.32) ve (2.33) de genel eksen takımındaki rotor ve stator gerilimlerinin akım ve halkalanma akısını içeren bağıntıları verilmiştir.

$$\bar{U}_{sg} = R_s . \bar{i}_{sg} + \frac{d\bar{\Psi}_{sg}}{dt} + j\bar{\Psi}_{sg} . \omega_g \tag{2.32}$$

$$\bar{U}_{rg} = R_r . \bar{i}_{rg} + \frac{d\bar{\Psi}_{rg}}{dt} + j.(\omega_g - \omega_r)\bar{\Psi}_{rg} \tag{2.33}$$

Daha önce elde edilen (2.28) ve (2.31) deki akı denklemleri yukarıda yerine konursa, bağıntılar aşağıdaki forma gelir.

$$\bar{U}_{sg} = R_s \bar{i}_{sg} + \frac{d}{dt}(L_s \bar{i}_{sg}) + \frac{d}{dt}(L_m \bar{i}_{rg}) + j.\omega_g (L_s \bar{i}_{sg} + L_m \bar{i}_{rg}) \tag{2.34}$$

$$\bar{U}_{rg} = R_r \bar{i}_{rg} + \frac{d}{dt}(L_r \bar{i}_{rg}) + \frac{d}{dt}(L_m \bar{i}_{sg}) + j.(\omega_g - \omega_r)(L_r \bar{i}_{rg} + L_m \bar{i}_{sg}) \tag{2.35}$$

Görüldüğü gibi (2.35) nolu eşitliğin sağ tarafında stator gerilimi bağıntısından farklı olarak rotor eksen takımı ile genel eksen takımı arasındaki bağıl hız farkı

nedeniyle hareketin indüklediği gerilim terimi mevcuttur. Bu elektromekanik enerji dönüşümünü de yardımcı olur.

Tüm rotor ve stator uzay fazörlerinin genel eksen takımındaki reel ve imajiner bileşenleri (2.36) da matrissel bir biçimde verilmiştir. Bu matriste s kayma, w_1 senkron hız olmak üzere $w_g=w_1$ seçilmiştir. Dolayısıyla senkron hızda dönen genel eksen takımındaki rotor akım ve gerilim bileşenleri aslında sıfırdır.

$$\begin{bmatrix} U_{sx} \\ U_{sy} \\ U_{rx} \\ U_{ry} \end{bmatrix} = \begin{bmatrix} R_s + pL_s & -\omega_1 L_s & pL_m & -\omega_1 L_m \\ \omega_1 L_s & R_s + pL_s & \omega_1 L_m & pL_m \\ pL_m & -s\omega_1 L_m & R_r + pL_r & -s\omega_1 L_r \\ s\omega_1 L_m & pL_m & s\omega_1 L_r & R_r + pL_r \end{bmatrix} * \begin{bmatrix} i_{sx} \\ i_{sy} \\ i_{rx} \\ i_{ry} \end{bmatrix} \qquad (2.36)$$

Gerilim denklemleri hem geçici hem de sürekli hal için geçerlidir. Ancak geçici halde mekanik kısma ilişkin bağıntıda, elektriksel döndürme momenti ifadesinin çıkarımı gereklidir.

$$M_e - M_l = J\frac{d\omega_m}{dt} + B\omega_m \qquad (2.37)$$

P çift kutup sayısı olmak üzere, elektriksel döndürme momentinin genel eksen takımındaki ifadesi (2.38) deki gibi yazılabilir.

$$M_e = -\frac{3}{2}P.L_m.\bar{i}_{sg} \times \bar{i}_{rg} = -\frac{3}{2}P.L_m(i_{sx}i_{ry} - i_{sy}i_{rx}) \qquad (2.38)$$

2.6 Sincap Kafesli Asenkron Makinanın D-Q Modeli

Bu modelde, tüm rotor ve stator değişkenlerini stator durağan eksen takımında tanımlamak için, daha önce genel eksen takımı için çıkarılan matematiksel modelde eksen takımının açısal hızı w_g sıfır alınır. Simülasyon kontrol amaçlı gerçeklendiğinden, girişte belirtilen varsayımlar yapılmıştır. Makinanın beslendiği evirici ideal varsayılarak, eviriciden kaynaklanan gecikme sıfır kabul edilmiştir. Rotor hızı, stator durağan eksen takımındaki stator ve rotor mıknatıslanma akımları durum

değişkenleri olarak; aynı eksen takımındaki stator gerilimleri ise kontrol parametreleri olarak seçilmiştir. Rotor mıknatıslanma akımının sD ve sQ eksenlerindeki bileşenlerinin durum değişkeni olarak seçilmesiyle, mıknatıslanma akımı dolayısıyla da halkalanma akısının genliği ve açısı bulunabilmektedir.[1]

$\bar{i}_r'$ durağan eksen takımındaki rotor akımıdır. L_m ve L_s' sırasıyla mıknatıslanma indüktansı ve stator geçici indüktansı olmak üzere, stator gerilim denklemi,

$$\bar{u}_s = Rs.\bar{i}_s + L_s' \frac{d\bar{i}_s}{dt} + \frac{L_m^2}{L_r} \frac{d\bar{i}_{mr}}{dt} \tag{2.39}$$

şeklindedir. σ_{rl} rotor kaçak faktörü olmak üzere, durağan eksen takımındaki rotor mıknatıslanma akımı aşağıdaki gibi yazılabilir.

$$\bar{i}_{mr}' = \bar{i}_s + (1+\sigma_{rl}).\bar{i}_r' \tag{2.40}$$

$$\sigma_{rl} = L_{rl} / L_m \tag{2.41}$$

Rotor mıknatıslanma akımının değişimi ve stator durağan eksen takımındaki bileşenleri cinsinden aşağıdaki gibi tanımlanabilir.

$$\frac{d\bar{i}_{mr}}{dt} = \frac{\bar{i}_s}{Tr} + \left(-\frac{1}{Tr} + j.\omega_r\right).\bar{i}_{mr} \tag{2.42}$$

$$\frac{di_{mrD}}{dt} = \frac{i_{sD} - i_{mrD}}{Tr} - \omega_r.i_{mrQ} \tag{2.43}$$

$$\frac{di_{mrQ}}{dt} = \frac{i_{sQ} - i_{mrQ}}{Tr} + \omega_r.i_{mrD} \tag{2.44}$$

Yukarıda verilen (2.42) eşitliği (2.39) daki stator gerilimi bağıntısında yerine konup, durum değişkeni olarak seçilen stator akımı bileşenleri eşitliğin bir tarafında yalnız bırakılacak olursa,

$$\frac{di_{sD}}{dt} = \frac{u_{sD}}{L_s'} - \left[\frac{1}{T_s'} + \frac{1-\sigma}{T_r'}\right].i_{sD} + \frac{1-\sigma}{T_r'} i_{mrD} + \frac{1-\sigma}{\sigma} \omega_r.i_{mrQ} \tag{2.45}$$

$$\frac{di_{sQ}}{dt} = \frac{u_{sQ}}{L_s'} - \left[\frac{1}{T_s'} + \frac{1-\sigma}{T_r'}\right].i_{sQ} + \frac{1-\sigma}{T_r'} i_{mrQ} - \frac{1-\sigma}{\sigma}\omega_r . i_{mrD} \qquad (2.46)$$

Mekanik kısma ilişkin elektriksel döndürme momenti ifadesi de (2.38) dekine benzer şekilde çıkarılırsa aşağıdaki bağıntıyla birlikte model tamamlanmış olur.

$$\frac{d\omega_r}{dt} = \frac{1}{J}\left(M_e - M_L - B.\omega_r\right) \qquad (2.47)$$

$$M_e = \frac{3}{2}.P\frac{L_m^2}{L_r}\left(i_{mrD}.i_{sQ} - i_{mrQ}.i_{sD}\right) \qquad (2.48)$$

Durum değişkenlerinden i_{mrD} ve i_{mrQ} yardımı ile mıknatıslanma akımının genliği aşağıdaki gibi çıkarılabilir.

$$\left|\bar{i}_{mr}\right| = \sqrt{\left(i_{mrD}^2 + i_{mrQ}^2\right)} \qquad (2.49)$$

$$\rho_r = \arctan\left(\frac{i_{mrQ}}{i_{mrD}}\right) \qquad (2.50)$$

Yukarıda ana hatlarıyla verilen asenkron makinanın D-Q modelinin matlab 5.2 derleyicisinde yazılan algoritması Ek1 de; motor değişkenlerine ilişkin çıkış eğrileri ise simülasyon sonuçları bölümünde verilmiştir. Durum değişkenleri olarak seçilen akım matrisinde lineer olmayan diferansiyel eşitlikler sözkonusu olduğundan, çözüm için nümerik integrasyon methodu kullanılmıştır.

3. VEKTÖR DENETİMİ

3.1 Asenkron Makinada Vektör Denetimi

Asenkron makinalar, yapısının basitliği, güvenilirliği, düşük maliyeti, boyutlarının küçüklüğü ve sabit hızda verimli olması gibi avantajları dolayısıyla kullanışlı makinalardır. Aslında bunun nedeni de güçlü bir dinamik interaksiyona sahip non-lineer dinamik yapısıdır. Ayrıca son yıllarda bu makinalar hız değişimlerine çabuk yanıt vermesine olanak sağlayan sürücü düzenlerinin geliştirilmesiyle daha çok tercih edilir olmuştur. Bu kontrol düzenleri az önce değinilen güçlü non-lineer yapı nedeniyle oldukça kompleksdir; ancak ne var ki değişken hızlarda kullanılan güç çeviricileri, d.c. makinaların beslenmesi için kullanılan çeviricilere oranla maliyeti daha yüksektir. [10]

Bir asenkron makinanın dinamik modeli en genel şekilde, stator gerilimi ve frekans girişler, moment, rotor hızı, mıknatıslanma akımı veya halkalanma akılarının istenen kombinasyonu çıkışlar olmak üzere altıncı dereceden durum denklemleri ile ifade edilir. Çözümü kolaylaştırmak amacıyla yapılan eksen dönüşümleri sonrasında, stator akımlarını akı ve moment indükleyen bileşenlerine ayırmak için açı ve genlik kontrolü yapılır. Başka bir deyişle akım vektörü denetlenir. Bilindiği üzere doğru akım makinalarında alan akısı ve endüi magnetomotor kuvvetleri kollektör ve fırçalarla yönlendirilirken; asenkron makinalarda alan akısı ve endüi mmk sının uzay açıları makine dışında kontrol edilmelidir. Bu amaçla uygulanan vektör denetimi ile ani hız değişimlerine daha duyarlı ve daha güvenilir bir kontrol yapılmış olur.

Asenkron makinalarda üç temel vektör denetimi methodu vardır: Rotor akısı yönlendirilmiş, stator akısı ve mıknatıslanma akısı yönlendirilmiş. Bu çalışmada rotor akısı yönlendirilmiş vektör kontrolü prosesine ilişkin simülasyon çalışması sunulmuştur.

3.2 Gerilim Aradevreli Eviriciden Beslenen Asenkron Makinada Rotor Akısı Yönlendirilmiş Vektör Denetimi

3.2.1 Rotor akısı yönlendirilmiş eksen takımında stator gerilimi eşitlikleri

Bu bölümde stator gerilimleri, rotor halkalanma akısı uzay fazörünün doğru eksen bileşeninin olduğu eksen takımında formüle edilmiştir.[1] Rotor halkalanma akısı aşağıdaki gibi yazılacak olursa,

$$\overline{\Psi}_{r\psi r} = L_r \bar{i}_{r\psi r} + L_m \bar{i}_{s\psi r} \tag{3.1}$$

rotor mıknatıslanma akımı ve halkalanma akısı arasındaki elektriksel lineerlikten yola çıkarak,

$$\bar{i}_{mr} = \frac{\overline{\Psi}_{r\psi r}}{L_m} = \frac{L_r}{L_m}\bar{i}_{r\psi r} + \bar{i}_{s\psi r} = \bar{i}_{s\psi r} + (1+\sigma_r).\bar{i}_{r\psi r} \tag{3.2}$$

şeklinde ifade edilebilir. Mıknatıslanma indüktansı ve kaçak indüktansların sabit olduğu lineer magnetik koşullar gözönüne alınacak olursa w_{mr} hızıyla dönen eksen takımındaki stator gerilimi bağıntısı (3.4) deki gibi olur.

$$\omega_{mr} = \frac{d\rho_r}{dt} \tag{3.3}$$

$$\bar{u}_{s\psi r} = R_s \bar{i}_{s\psi r} + L_s \frac{d\bar{i}_{s\psi r}}{dt} + L_m \frac{d\bar{i}_{r\psi r}}{dt} + j\omega_{mr} L_s \bar{i}_{s\psi r} + j\omega_{mr} L_m \bar{i}_{r\psi r} \tag{3.4}$$

Rotor akımı uzay fazörü (3.2) de yeniden düzenlenirse,

$$i_{r\psi r} = \frac{|i_{mr}| - i_{s\psi r}}{1+\sigma_r} \tag{3.5}$$

bulunur. Bu eşitlik stator gerilimi denkleminde yerine konur ve eşitliğin iki tarafi stator direncine bölünürse,

$$T_s' \frac{d\bar{i}_{s\psi r}}{dt} + \bar{i}_{s\psi r} = \frac{\bar{u}_{s\psi r}}{R_s} - j.\omega_{mr} T_s' \bar{i}_{s\psi r} - (T_s - T_s')\left(j\omega_{mr} |\bar{i}_{mr}| + \frac{d\bar{i}_{mr}}{dt} \right) \qquad (3.6)$$

$\sigma=1-L_m^2/(L_s.L_r)$ toplam kaçak faktörü, $L_s'=\sigma.L_s$ stator geçici indüktansı ve $T_s'=L_s'/R_s$ stator geçici zaman sabiti olmak üzere (3.6) eşitliği, reel(x) ve imajiner(y) bileşenlerine aşağıdaki biçimde ayrıştırılır.

$$T_s' \frac{di_{sx}}{dt} + i_{sx} = \frac{u_{sx}}{R_s} + \omega_{mr} T_s' i_{sy} - (T_s - T_s') \frac{|d\bar{i}_{mr}|}{dt} \qquad (3.7)$$

$$T_s' \frac{di_{sy}}{dt} + i_{sy} = \frac{u_{sy}}{R_s} - \omega_{mr} T_s' i_{sx} - (T_s - T_s')\omega_{mr} |\bar{i}_{mr}| \qquad (3.8)$$

Asenkron makina (3.7) ve (3.8) eşitliklerindeki i_{sx} ve i_{sy} akımlarına göre, zaman sabiti stator geçici zaman sabitine, kazancı ise stator direncinin tersine eşit olan birinci dereceden time-delay eleman gibi davranır. Ayrıca aynı eksen takımındaki stator gerilimi bileşenleri ile diğer eksendeki akım bileşenleri ile arasında istenmeyen kuplajlar görülmektedir. Bu nedenle u_{sx} ve u_{sy} gerilimleri ayrıştırılmış kontrol değişkenleri olarak kullanılamaz. Rotor akısı yönlendirmeli kontrolün amacı da i_{sx} ve i_{sy} akım bileşenlerinin birbirinden bağımsız olarak denetimini sağlamaktır. Ayrıştırma amacıyla kullanılan dekuplaj devreleri ilerleyen bölümde verilmiştir.

3.2.2 Dekuplaj (ayrıştırma) devreleri

Yukarıdaki (3.7) ve (3.8) eşitliklerinde doğru eksendeki gerilimde i_{sy} nin indüklediği gerilim ve benzer şekilde dik eksen gerilim bileşeninde de i_{sx} akımının indüklediği terimler mevcuttur. Eğer sürüşün ideal yani ölü zamanın olmadığı ve yine evirici katından kaynaklanan gecikmelerin söz konusu olmadığı; ayrıca rotor akısı genliğinin sabit olduğu varsayılacak olursa i_{sx} ve i_{sy} akımları bağımsız olarak kontrol edilebilir. Bunu yapabilmek için i_{sx} ve i_{sy} akımlarını bağımsız olarak kontrol eden akım kontrolörlerinin çıkışı olan u_{sx} ve u_{sy} gerilimleri sırasıyla u_{dx} ve u_{dy} ile toplanarak doğru ve dik eksendeki gerilim bileşenleri bulunur. İlgili kontrolör çıkışları olan u_{sx} ve u_{sy} tanımları aşağıda verilmiştir.[1]

$$u_{dx} = -\omega_{mr} L_s' i_{sy} \tag{3.9}$$

$$u_{dy} = \omega_{mr} L_s' i_{sx} + (L_s - L_s')\omega_{mr} \left|\bar{i}_{mr}\right| \tag{3.10}$$

$$u_{sx} = R_s i_{sx} + L_s' \frac{di_{sx}}{dt} \tag{3.11}$$

$$u_{sy} = R_s i_{sy} + L_s' \frac{di_{sy}}{dt} \tag{3.12}$$

Böylece u_{sx} ve u_{sy} stator geçici zaman sabitiyle birlikte küçük bir gecikme ile i_{sx} ve i_{sy} akımlarını kontrol edebilmektedir.

3.2.3 Rotor akı modeli

Rotor akısı yönlendirmeli eksen takımındaki gerilim eşitlikleri kullanılarak rotor halkalanma akısı uzay fazörünün modülü ve açısı veya mıknatıslanma akımının genliği ile açısal hızı w_{mr} bulunabilir. Daha önce genel eksen takımında tanımlı rotor gerilimini veren (2.33) denkleminde genel eksen takımının açısal hızı w_g yerine w_{mr} konulursa, rotor akısı yönlendirilmiş eksen takımındaki rotor gerilimi aşağıdaki biçimde formüle edilebilir.

$$0 = R_r \bar{i}_{r\psi r} + \frac{d\overline{\psi}_{r\psi r}}{dt} + j(\omega_{mr} - \omega_r)\overline{\psi}_{r\psi r} \tag{3.13}$$

$\overline{\psi}_{r\psi r}$ rotor akısı yönlendirilmiş eksen takımındaki halkalanma akısıdır ve mıknatıslanma akımı uzay fazörünün genliğine eşit olduğu durumda aşağıdaki gibi tanımlanır.

$$\overline{\psi}_{r\psi r} = L_m \left|\bar{i}_{mr}\right| \tag{3.14}$$

Mıknatıslanma indüktansının sabit olduğu (akıda saturasyon etkisi ihmal) varsayılarak akı ifadesi (3.13) de yerine konursa rotor gerilimi ifadesi (3.15) deki gibi elde edilir.

$$0 = R_r \bar{i}_{r\psi r} + L_m \frac{d|\bar{i}_{mr}|}{dt} + j(\omega_{mr} - \omega_r) L_m |\bar{i}_{mr}| \tag{3.15}$$

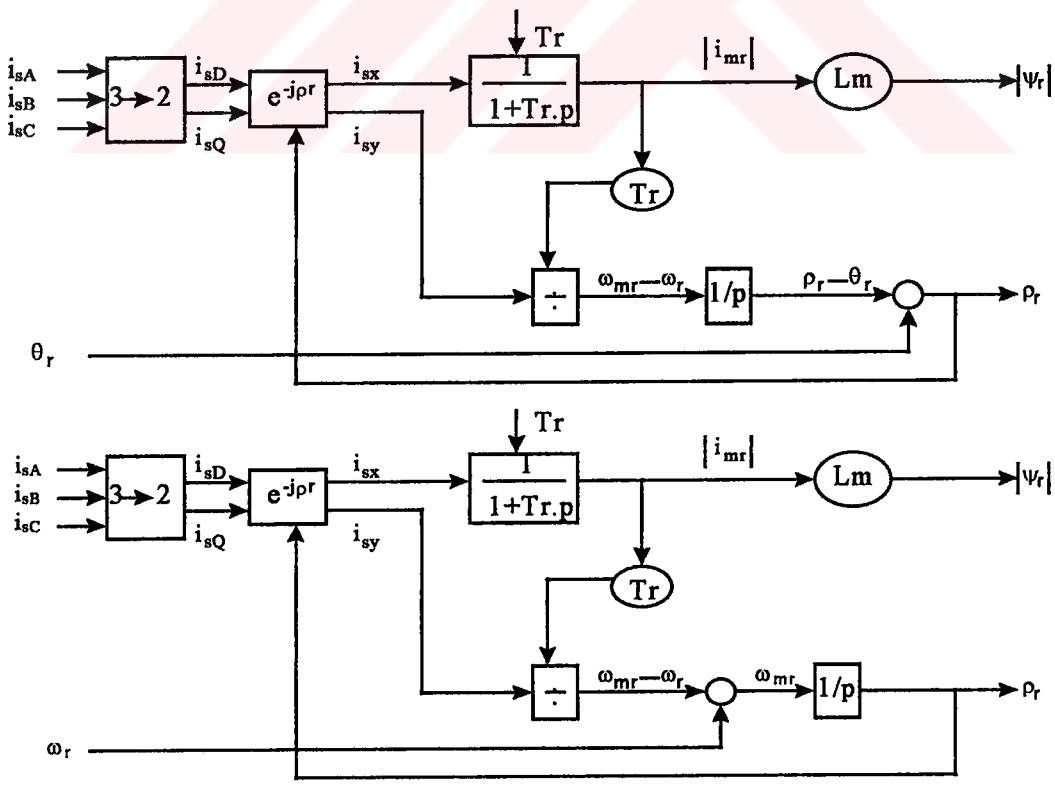
Rotor eksen takımındaki rotor akımına ilişkin (3.5) denklemi (3.15) deki rotor gerilimi bağıntısında yerine konup, eşitliğin her iki tarafı rotor direncine bölünürse,

$$T_r \frac{d|\bar{i}_{mr}|}{dt} + |\bar{i}_{mr}| = \bar{i}_{s\psi r} - j(\omega_{mr} - \omega_r) T_r |\bar{i}_{mr}| \tag{3.16}$$

Yukarıda elde edilen eşitlik reel ve imajiner bileşenlerine ayrılırsa, rotor akısı yönlendirilmiş eksen takımındaki akı modelini tanımlayan bağıntılar elde edilir.

$$T_r \frac{d|\bar{i}_{mr}|}{dt} + |\bar{i}_{mr}| = i_{sx} \tag{3.17}$$

$$\frac{d\rho_r}{dt} = \omega_{mr} = \omega_r + \omega_{sl} = \omega_r + \frac{i_{sy}}{T_r |\bar{i}_{mr}|} \tag{3.18}$$



Şekil 3.1 Rotor akısı yönlendirilmiş eksen takımında akı modeli

Yukarıdaki denklemde w_{sl} kayma açısal hızı, ρ_r rotor halkalanma akısı uzay fazörünün duran eksen takımının doğru ekseni ile yaptığı açıdır. (3.17) den de görüleceği üzere mıknatıslanma akımı, sabit olması durumunda i_{sx} akımına eşit olmaktadır ve i_{sx} bileşeni ile oynayarak mıknatıslanma akımı istenilen düzeyde tutulabilir; eğer nominal çalışma hızı altındaki koşullarda alan zayıflatma yöntemi uygulanmıyorsa elektromagnetik döndürme momentini belirleyen de dik eksendeki i_{sy} bileşenidir.

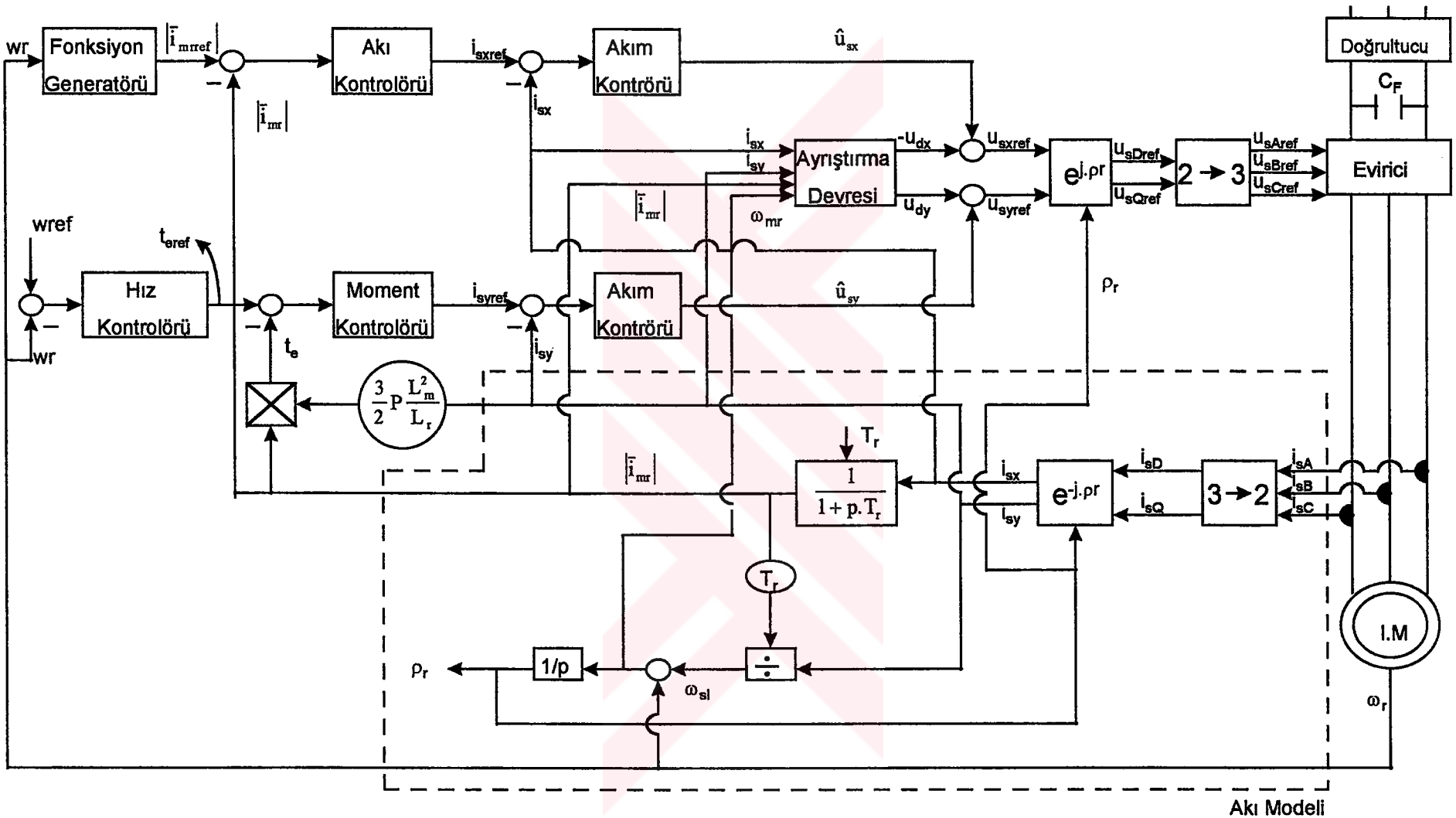
Yukarıda şekil 3.1 de, simülasyonda kullanılan akı modeline ilişkin blok şema verilmiştir. Sıcaklığa bağlı olarak değişen rotor direnci ve dolayısıyla yeni rotor zaman sabiti bilgisinin belirli periyodik aralıklarda modele girilmesi ve modelin çıkışlarının güncellenmesi gerekmektedir. Bu nedenle T_r nin değişimine duyarlı, daha sağlıklı bir kontrol yapmak için, T_r nin değişimine göre akı modelini on-line olarak güncellemek gerekir. Bu amaçla hazırlanmış neural yapılar içeren bir kontrol yapısı 5. bölümde verilmiştir..

3.3 Gerilim Aradevreli Eviriciden Beslenen Asenkron Makinada Rotor Akısı Yönlendirilmiş Vektör Denetimi Simülasyon Modeli

Asenkron makinanın anahtarlama frekansı düşük (100 Hz.- 1 kHz) darbe genişlik modülasyonlu (PWM) gerilim aradevreli evirici ile beslendiği varsayılır. Bu tür eviriciler genellikle 100kW güce kadar konum kontrollü servo sürücülerde kullanılır. Daha düşük güç seviyelerinde yüksek anahtarlama frekanslı tranzistörler, daha yüksek güçlerde ise klasik tristörler ve kapıdan kesime götürülen tristörler kullanılır. Anahtarlama frekansının düşük olmasından dolayı stator akımlarının kapalı çevrim kontrolü hızlı olarak yapılamaz. Bu sürücülerde stator akımları büyük genlikli harmonikler içerir ve bu nedenle de stator akımları yerine stator gerilimlerini kullanmak daha doğru olur. Ancak asenkron makinanın akım aradevreli eviriciden beslenmesi durumunda denklemlerde ve sürücü devrelerinde büyük kolaylıklar sağlanır. Düşük güç seviyelerinde gerilim aradevreli tranzistörlü hızlı akım kontrolü yapılan bir evirici ile beslenen asenkron makinada, stator akımları yaklaşık olarak sinüsoidal kabul edilir ve herbir faza ait stator akımları ayrı ayrı kontrol çevrimleri ile kontrol edilir. Rotor alan yönlendirmeli kontrol uygulamak için makinanın statora ait akım, gerilim ve rotora ait hız bilgilerinin alınması zaruridir. Şekil 3.2 de görüldüğü gibi bu amaçla uygulanan kontrol yapısını içeren blok şemada rotor hızı ve stator akım bilgisi birer geribesleme ile alınmaktadır. Rotor hızı stator akımlarından bağımsızdır. Sürekli çalışma koşullarında rotor direncine bağlılık artar. Stator akımları ve rotor hızı akı modeline giriş büyüklükleri olarak alınır. Benzer biçimde ayrıştırma devresine de giriş olarak stator akım bilgisi gelmektedir. Akı modeli çıkış büyüklükleri, rotor mıknatıslanma akımı uzay fazörünün modülü ve açısal hızıdır.[1,5]

Rotor akısı yönlendirilmiş kontrol, sabit moment veya sabit akı ve sabit güç bölgesinde uygulanır. Sabit akı bölgesinde i_{sx} akımı sabit tutulur. Bu bölgede moment i_{sy} akımı ile orantılı olarak değişir. Akı zayıflatma bölgesinde ise evirici en büyük gerilim değerlerinde çalışır ve makinanın doymaya girmemesi için artan hızla ters orantılı olarak rotor halkalanma akısı veya mıknatıslanma akımının kısılması yoluna gidilir.

Prosesin içinde toplam beş adet hız, akı, akım ve moment (PI) kontrolörü vardır. Fonksiyon üreteci ise akı zayıflatma bölgesini oluşturarak, rotor mıknatıslanma



Şekil 3.2 Gerilim aradevreli eviriciden beslenen asenkron makinada rotor akısı yönlendirilmiş vektör denetimi

akımının referans değerinin bulunması amacıyla kullanılır. Hız belirlenen nominal değerin üzerine çıktığında, referans akımın genliğini kısarak gerçek mıknatıslanma akımıyla arasında oluşan hata fonksiyonuna göre, referans stator akımının x ekseni bileşenini kontrol eder. Dolayısıyla fanksiyon üretecinin tasarımı makinanın doyma koşulları gözönüne alınarak yapılmalıdır.

Sırasıyla akı ve moment kontrolörlerinin çıkışları i_{sx} ve i_{sy} akımlarına karşılık gelen büyüklükler ile karşılaştırılarak elde edilen hatalar akım kontrolörlerinin girişlerini oluştururlar. Akım kontrolörlerinin çıkışları (u_{sx} ve u_{sy}) ayrıştırma devresi çıkışları ile toplanarak w_{mr} hızında dönen referans eksen takımındaki referans stator gerilim bileşenleri elde edilir. Elde edilen referans stator gerilimleri, durağan eksen takımındaki referans gerilimlere, ardından da üç faz referans gerilim değerlerine dönüştürülürler. Bu referans işaretler PWM li eviricideki yarıiletkenleri kapıdan süren anahtarlama işaretleridir.

Simülasyon için durağan eksen takımındaki stator gerilimi eşitlikleri yazılacak olursa,

$$\frac{di_{sD}}{dt} = \frac{u_{sD}}{L_s'} - \left[\frac{1}{T_s'} + \frac{1-\sigma}{T_r'}\right].i_{sD} + \frac{1-\sigma}{T_r'} i_{mrD} + \omega_r \frac{1-\sigma}{\sigma} i_{mrQ} \qquad (3.19)$$

$$\frac{di_{sQ}}{dt} = \frac{u_{sQ}}{L_s'} - \left[\frac{1}{T_s'} + \frac{1-\sigma}{T_r'}\right].i_{sQ} + \frac{1-\sigma}{T_r'} i_{mrQ} - \omega_r \frac{1-\sigma}{\sigma} i_{mrD} \qquad (3.20)$$

Rotor mıknatıslanma akımının stator durağan eksen takımındaki bileşenleri aşağıdaki gibidir.

$$\frac{di_{mrD}}{dt} = \frac{i_{sD} - i_{mrD}}{T_r} - \omega_r i_{mrQ} \qquad (3.21)$$

$$\frac{di_{mrQ}}{dt} = \frac{i_{sQ} - i_{mrQ}}{T_r} + \omega_r i_{mrD} \qquad (3.22)$$

(3.19)...(3.22) denklemleri, stator akımları ve rotor mıknatıslanma akımları durum değişkenleri olmak üzere matrissel olarak düzenlenirse,

$$A=\begin{bmatrix} -\left(\frac{1}{T_s'}+\frac{1-\sigma}{T_r'}\right) & 0 & \frac{1-\sigma}{T_r'} & 0 \\ 0 & -\left(\frac{1}{T_s'}+\frac{1-\sigma}{T_r'}\right) & 0 & \frac{1-\sigma}{T_r'} \\ \frac{1}{T_r} & 0 & -\frac{1}{T_r} & 0 \\ 0 & \frac{1}{T_r} & 0 & -\frac{1}{T_r} \end{bmatrix} \quad (3.23)$$

$$B=\begin{bmatrix} -\left(\frac{1}{T_s'}+\frac{1-\sigma}{T_r'}\right) & 0 & \frac{1-\sigma}{T_r'} & 0 \\ 0 & -\left(\frac{1}{T_s'}+\frac{1-\sigma}{T_r'}\right) & 0 & \frac{1-\sigma}{T_r'} \\ \frac{1}{T_r} & 0 & -\frac{1}{T_r} & 0 \\ 0 & \frac{1}{T_r} & 0 & -\frac{1}{T_r} \end{bmatrix} \quad (3.24)$$

$$C=\begin{bmatrix} \frac{1}{L_s'} & 0 & 0 & 0 \\ 0 & \frac{1}{L_s'} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.25)$$

Mekanik kısma ilişkin rotor hızı ve elektriksel döndürme momenti bağıntıları (3.26) ve (3.27) de olduğu gibidir.

$$M_e=\frac{3}{2}P\frac{L_m^2}{L_r}(i_{mrD}i_{sQ}-i_{mrQ}i_{sD}) \quad (3.26)$$

$$\frac{d\omega_r}{dt}=\frac{1}{J}(M_e-M_L) \quad (3.27)$$

Rotor mıknatıslanma akımı uzay fazörünün stator durağan eksen takımına göre açısal hızı,

$$\omega_{mr} = \frac{i_{mrD}\dfrac{di_{mrQ}}{dt} - i_{mrQ}\dfrac{di_{mrD}}{dt}}{i_{mrD}^2\left[1+(i_{mrQ}/i_{mrD})^2\right]} \tag{3.28}$$

ρ_r rotor mıknatıslanma akısının uzay açısı olmak üzere rotor akısı yönlendirmeli referans eksen takımında stator akımları ile durağan eksen takımındaki akım bileşenleri arasındaki dönüşüm aşağıda verilmiştir.

$$\begin{bmatrix} i_{sx} \\ i_{sy} \end{bmatrix} = \begin{bmatrix} \cos\rho_r & \sin\rho_r \\ -\sin\rho_r & \cos\rho_r \end{bmatrix} \begin{bmatrix} i_{sD} \\ i_{sQ} \end{bmatrix} \tag{3.29}$$

Akı modelinin çıkışları olan mıknatıslanma akımı genliği ve açısal hızı daha önce (3.17) ve (3.18) de verildiği gibidir. Ayrıştırma devresinin çıkışları u_{dx} ve u_{dy} bileşenleri de (3.9) ve (3.10) da verildiği gibidir.

Aşağıda çıkış ifadeleri verilmiş olan PI kontrolörlerin katsayıları simülasyon sırasında deneme-yanılma yoluyla bulunmuştur. Mevcut kontrolör ile sistem yeteri kadar kararlı bir dinamik cevap vermektedir; ancak katsayıların kalıcı konum ve hız hal hatalarını sıfır yapacak biçimde sentez yoluyla belirlenmesi daha sağlıklı kontrol imkanı sağlayacaktır. Daha önce matrissel ifadesi verilen stator akımlarına ilişkin lineer olmayan diferansiyel durum denklemlerinin çözümü için nümerik integrasyon metodunun kullanılması zorunludur.

$$i_{sxref}(k) = Kp1 * (|\bar{i}_{mrref}(k)| - |\bar{i}_{mr}(k)|) + caki(k) \tag{3.30}$$

$$caki(k+1) = caki(k) + dt * \frac{Kp1}{Ki1}\left(|\bar{i}_{mrref}(k)| - |\bar{i}_{mr}(k)|\right) \tag{3.31}$$

$$u_{sx}(k) = Kp2 * (i_{sxref}(k) - i_{sx}(k)) + cisx(k) \tag{3.32}$$

$$cisx(k+1) = cisx(k) + dt * \frac{Kp2}{Ki2}\left(i_{sxref}(k) - i_{sx}(k)\right) \tag{3.33}$$

$$u_{sxref} = \hat{u}_{sx} + u_{dx} \tag{3.34}$$

$$m_{eref}(k) = Kp3 * (\omega_{ref}(k) - \omega_r(k)) + chız(k) \tag{3.35}$$

$$chız(k+1) = dt * \frac{Kp3}{Kı3} * (\omega_{ref}(k) - \omega_r(k)) \tag{3.36}$$

$$i_{syref}(k) = Kp4 * (m_{eref}(k) - m_e(k)) + cmom(k) \tag{3.37}$$

$$cmom(k+1) = cmom(k) + dt * \frac{Kp4}{Kı4} * (m_{eref}(k) - m_e(k)) \tag{3.38}$$

$$\hat{u}_{sy}(k) = Kp5 * (i_{syref}(k) - i_{sy}(k)) + cisy(k) \tag{3.39}$$

$$cisy(k+1) = cisy(k) + dt * \frac{Kp5}{Kı5} * (i_{syref}(k) - i_{sy}(k)) \tag{3.40}$$

$$u_{syref}(k) = \hat{u}_{sy}(k) + u_{dy}(k) \tag{3.41}$$

Gerilim aradevreli eviriciden beslenen sincap kafesli asenkron makinanın rotor akısı yönlendirmeli vektör kontrolü simülasyon modeli yukarıda ayrıntılı bir biçimde verilmiştir. İlgili algoritma matlab derleyicisinde yazılmış ve simülasyon sonuçları karşılaştırmalı olarak 6. bölümde verilmiştir.

4. YAPAY SİNİR AĞLARI

4.1 Yapay Sinir Ağları Üzerine Temel Bilgiler

4.1.1 Ysa tanımı

Yapay sinir ağları, birbirine yoğun bir şekilde paralel olarak bağlanmış basit (genellikle adaptif) elemanlardan oluşmuş ve gerçek dünyadaki nesnelerle biyolojik sinir dizgesinde olduğu gibi etkileşmek üzere hiyerarşik olarak düzenlenmiş bir ağdır.[2,12]

4.1.2 Sinaps tanımı

Biyolojist yaklaşımı ile, biyolojik sinir ağında bilgilerin saklandığı ve yeni bilgilerin öğrenildiği yerlere *sinaps noktaları* denir.[2,12]

4.1.3 Ysa modelleri

- Raslantısal Ağlar
  - Erol Gelenbe 1988
  - Amari
- Deterministik YSA
  - Dağılmış parametreli YSA (Hogkin Huxley)
  - Toplu parametreli YSA
    - Cebrik Ysa
      - Lineer cebrik YSA
      - Lineer olmayan cebrik YSA
    - Dinamaik YSA
      - Birinci mertebeden dinamik YSA

- Yüksek mertebeden dinamik YSA

4.1.4 Öğrenme açısından sınıflama

- Salt eylemci
- Öğrenen
  - Eğiticisiz (kendi kendine öğrenen)
  - Eğiticili
    - Tam doğru çıkışı söyleyen eğitici durumu
    - Salt ödül ceza veren

4.1.5 Neural network uygulaması

Biyolojik sinir ağlarından esinlenerek oluşturulan işlem birimi olan ilk nöron basit bir ağırlıklı toplama elemanıdır. Daha sonraki yıllarda daha karmaşık hücre ve katman tipleri geliştirilmiştir. Biyolojik sinir hücrelerinde bilgi sinapslarda saklanırken, yapay sinir hücrelerinde ise ağırlıklarda saklanmaktadır. Her ikisinde de belirli bir eğitim süreci sonunda istenilen çıkış belirli bir tolerans çerçevesinde alınabilir. Genelde nöronlar aktivasyon fonksiyonları ve net fonksiyonlarla (temel fonk.) modellenir. Biyolojik perspektifin bağlantı yapısını ve sinir fonksiyonlarını tam belirli ve detayları ile modellemek imkansızdır. Dolayısıyla bu fonksiyonların seçimi genelde sinir modeli uygulamasının içerdiği konsepte bağlıdır. Yani sinir ağı büyük yaklaşıklıkla biyolojik realitelere uygun hale getirilmeye çalışılır. Bir neural network aşağıdaki üç temel karekteristiğe sahiptir.[7]

i) *Uyumluluk (Adaptiveness) ve self organizasyon:* Adaptiv öğrenme ve self organizasyon kurallarını benimseyerek adaptiv işleme ve kumanda edebilme yeteneği sözkonusudur.

ii) *Non-Lineer Network çalışması:* Artırılmış bir network tahmini, tabakalandırma ve gürültüye karşı bağışıklık yeteneği vardır.

iii) *Paralel çalışma:* Genelde kurulan modeller, genişletilmiş ağ bağlantıları sayesinde çok sayıda ve birarada çalışan hücreler gibi davranır.

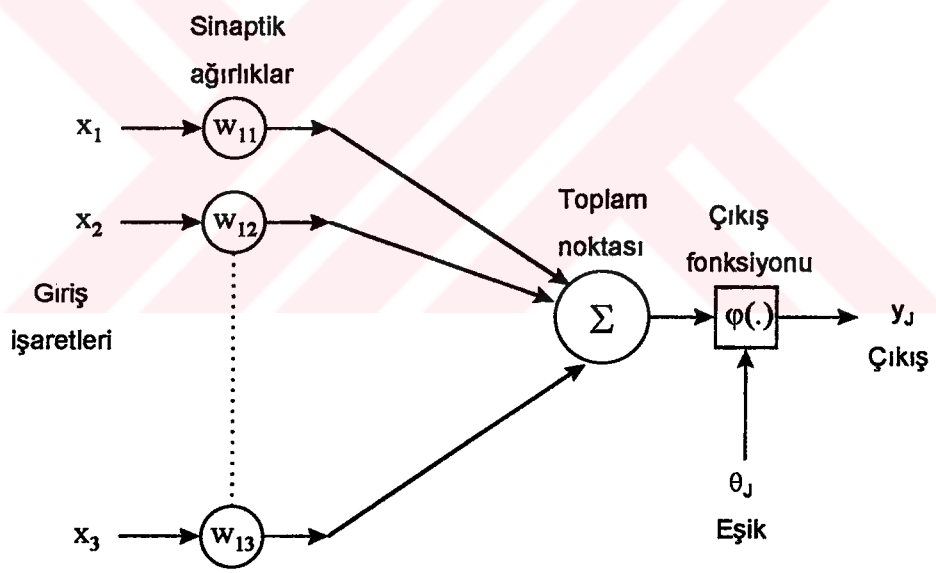
4.1.6 Nöron tipleri ve katmanlı ağlar

En basit YSA bir sinir hücresidir. Bir sinir hücresi, bağlantı elemanları (ağırlıklar), toplama elemanı ve çıkış birimi olmak üzere üç bölümden ibarettir. Ağırlıklar uygun eğitim yöntemlerinden biriyle değiştirilip istenilen değerlere getirilir. Toplama birimiyle tüm girişler ağırlıklarla çarpılıp çıkış birimine iletilir.

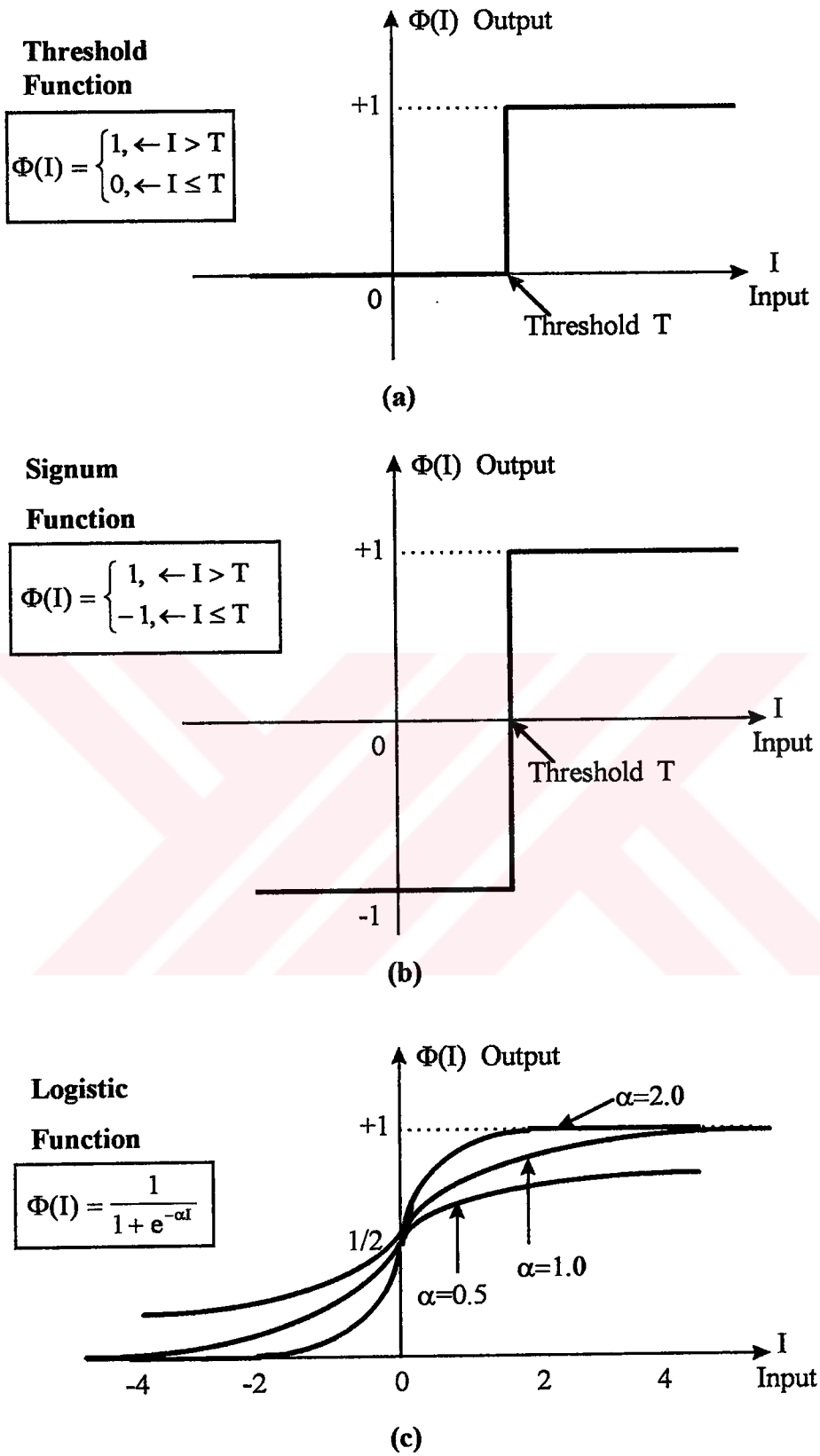
$$I_j = \sum_{i=1}^{n} w_{ij} x_i \tag{4.1}$$

$$Y_j = \phi(I_j) \tag{4.2}$$

Çıkış birimi de giriş argumanı olarak I_J ve eşik değerini alıp çıkışı, çoğu zaman iki değer arasında sınırlandırılmış olarak verilir.



Şekil 4.1 Doğrusal olmayan sinir hücresi modeli



Şekil 4.2 Sinir hücreleri için tanımlanan bazı aktivasyon fonksiyonları

Giriş işaretleri $x_1,x_2,.....x_n$ sürekli değişkenlerdir. Bu giriş işaretleri sinaptik ağırlıklarla çarpılarak hücrenin toplam ünitesinde işlem görür. Buradaki ağırlıklar, elektriksel işaret akışının ivmesi veya ket vurmasına uygun olarak pozitif veya negatif olabilir. Hücrede toplama fonksiyonu ve aktivasyon fonksiyonu olmak üzere iki temel işlem ünitesi vardır. İkinci ünite yani aktivasyon fonksionu içeren blok etkili bir lineer olmayan filtre gibi düşünülebilir.[12]

Şekil 4.2 de bazı aktivasyon fonksiyonları verilmiştir. (2.a) da eşik fonksiyonu, (2.b) de kuvantalama (signum) fonksiyonu, (2.c) de ise sigmoidal fonksiyon ve bunlara ilişkin karakteristik eğriler verilmiştir. Örneğin (2.c) dekine ilişkin çıkış ifadesi aşağıdaki biçimdedir.

$$\phi(I) = \frac{1}{1+e^{-\alpha I}} \tag{4.3}$$

Burada α, iki asimptotik değer arasında değişen fonksiyonun keskinliğini belirleyen bir sabittir.

Aktivasyon fonksiyou için daha tanımsal bir terim ise bastırma (squashing) fonksiyonudur. Bu fonksiyon, nöronun çıkış değerini iki asimptotik değer arasında sınırlar. Bu sınırlama da, işlem ünitesi çıkışının makul dinamik değerler arasında kalması için son derece kullanışlıdır. Bununla beraber aktivasyon fonksiyonu bazen lineer ilişkinin sadece sağ yarı düzlemde olduğu durumlarda kullanılır. Lineer aktivasyon fonksiyonunun kullanımı yapay sinir hücresindeki non-lineerliği ortadan kaldırır. Aktivasyon fonksiyonunda non-lineerlik söz konusu değilse yapay sinir ağı non-lineer bir fenomen modelleyemez.

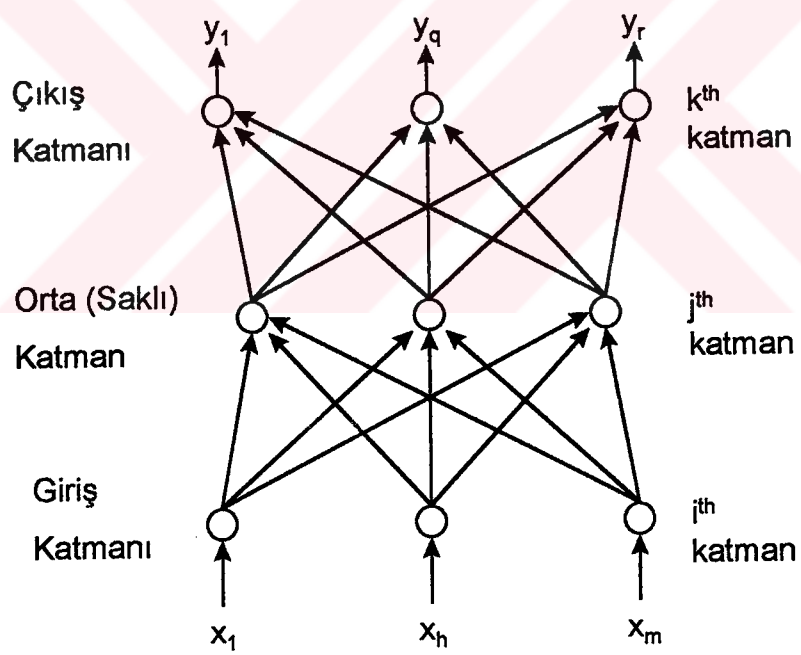
4.1.7 Yapay sinir ağları

Bir yapay sinir ağı, beynin selebral korteks yapısından esinlenerek gerçekleştirilen mimari yapı içinde, birbiriyle bağlı çok sayıda yapay sinir hücresinden oluşmuş bir ünitedir.

Bu işlem ünitesi ardışıl tabakalar veya dilimler olarak şekillenir. Bu tabakalar arasında ise ya rastgele ya da tam bağlantı sözkonusudur. Şekil 4.3 de bir sinir ağı yapısı verilmiştir. Giriş tabakası ağın giriş bilgilerini içeren bir terminaldir. Ancak

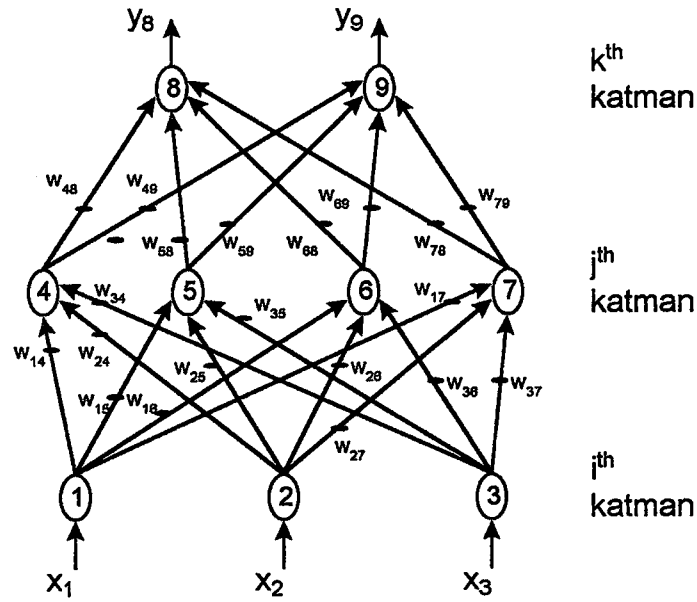
aktivasyon fonksiyonlarının uygulandığı bir yer değildir. Çünkü düğüm noktalarında giriş ağırlıkları ve aktivasyon fonksiyonları yoktur. En üstteki tabaka ise uygulanan girişe karşılık uygun bir çıkış veren çıkış tabakasıdır. Aradaki tabaka veya tabakalara da orta veya kör tabaka denir; çünkü bu tabakaların dışarıdaki prosesler ile doğrudan bir bağlantısı yoktur.

Genelde iki tip sinir ağı kullanılır. Bunlardan biri heteroassociative sinir ağı; ki burada çıkış vektörü giriş vektöründen farklıdır. Diğeri ise çıkış vektörünün, giriş vektörüyle özdeş olduğu autoassociative sinir ağıdır. Sözkonusu tabakadaki hücreler arasında yanal bağlantı veya gerideki tabakalara bağlantı yoksa buna ileri yönde beslemeli ağ denir. Fakat geri beslemeli sinir ağı yapısı daha kullanışlı ve pek çok uygulama için de daha uygun olmasına rağmen kullanım oranı düşüktür. Belirli durumlarda bir nöron, girişle çıkış arasında bir geribesleme içerir. Bu tür belirli durumlarda eğitim yoluyla belirlenmiş ağırlıklar içeren bağlantılar vardır.



Şekil 4.3 İleri yönde beslemeli temel sinir ağı yapısı

Sinir hücreleri arasındaki her bağlantı Şekil 4.4 de görüldüğü gibi düzeltilmiş bir ağırlık bilgisi içerir. Bu basit sinir ağı, giriş katmanında 3, orta katmanda 4, çıkış katmanında ise 2 hücre içeren ve tam bağlantılı ileri yönde beslemeli bir sinir ağıdır. Bağlantılar üzerindeki ağırlıklar w_{ij} ile gösterilmiştir.



Şekil 4.4 Basit ileri yönde beslemeli sinir ağı

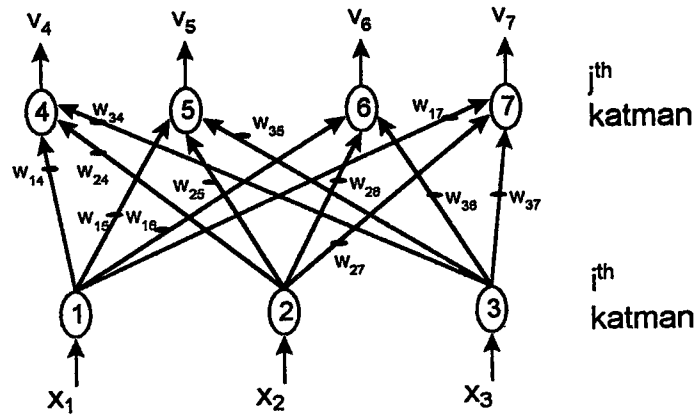
Şekilde Y çıkış vektörü bileşenleri y_8 ve y_9 ; X giriş vektörü bileşenleri x_1, x_2 ve x_3 olarak verilmiştir. x_1 işareti giriş tabakasındaki birinci nörona uygulandığında w_{14}, w_{15}, w_{16} ve w_{17} ağırlıkları ile orta tabakada ki nöronlara giriş olarak aktarılır. x_2 ve x_3 girişlerine yeni verilerin girilmesi durumunda bilgiler, ağırlıklar yardımıyla orta katmandaki hücrelere aktarılır.

4.1.8 Vektör ve matris notasyonu

Giriş, çıkış ve ağırlıklar arasında matris notasyonunun kullanılması uygundur. Sinir ağının çıkış tabakası kesilirse, Şekil 4.5 deki gibi çıkış vektörü V_J nin bileşenleri v_4, v_5, v_6, v_7 olarak ifade edilir. Eğer aktivasyon fonksiyonları lineer bir fonksiyon teşkil edecek şekilde sınırlandırılırsa, daha önce tanımlanan matematiksel fonksiyon matris formunda yazılabilir. V_J ve X_I kolon vektörleri, w_{JI} ise j*i boyutunda matris olarak yazılabilir.

$$\begin{bmatrix} v_4 \\ v_5 \\ v_6 \\ v_7 \end{bmatrix} = \begin{bmatrix} w_{14} & w_{24} & w_{34} \\ w_{15} & w_{25} & w_{35} \\ w_{16} & w_{26} & w_{36} \\ w_{17} & w_{27} & w_{37} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad (4.4)$$

$$V_j = W_{ij} * X_i \tag{4.5}$$



Şekil 4.5 Orta katman üzerinden kesilmiş olan ağın alt kısmı

Benzer şekilde orta katman çıkışları girişler olarak algılanırsa, ağın üst kısmı da Şekil 4.6 daki gibi gösterilebilir.

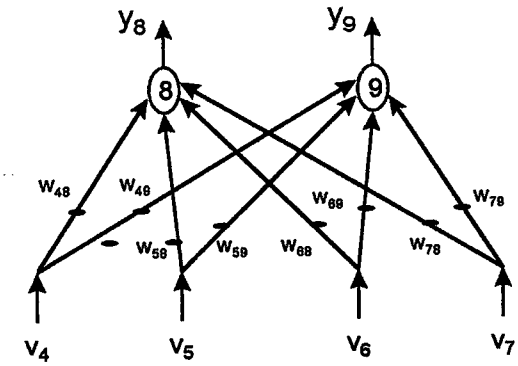
$$\begin{bmatrix} y_8 \\ y_9 \end{bmatrix} = \begin{bmatrix} w_{48} & w_{58} & w_{68} & w_{78} \\ w_{49} & w_{59} & w_{69} & w_{79} \end{bmatrix} * \begin{bmatrix} v_4 \\ v_5 \\ v_6 \\ v_7 \end{bmatrix} \tag{4.6}$$

$$Y_k = W_{jk} * V_i \tag{4.7}$$

(4.4) ve (4.6) bağıntılarından yola çıkılarak Y_k çıkış vektörü ile X_i giriş vektörü arasındaki ilişki aşağıdaki gibi yazılabilir.

$$\begin{bmatrix} y_8 \\ y_9 \end{bmatrix} = \begin{bmatrix} w_{48} & w_{58} & w_{68} & w_{78} \\ w_{49} & w_{59} & w_{69} & w_{79} \end{bmatrix} * \begin{bmatrix} w_{14} & w_{24} & w_{34} \\ w_{15} & w_{25} & w_{35} \\ w_{16} & w_{26} & w_{36} \\ w_{17} & w_{27} & w_{37} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{4.8}$$

$$Y_k = W_{ij} * W_{jk} * X_j = W_{ijk} * X_i \tag{4.9}$$



Şekil 4.6 Orta katmanın üzerinden kesilmiş ağın üst kısmı

4.1.9 Doğrusal birleştirici (linear associator) yapay sinir ağı

İlk yapay sinir ağlarının çoğu doğrusal birleştiricilerdi ve eğitim algoritmaları ile yapay sinir ağlarının yeteneklerini ve sınırlarını belirlediler. Burada temel kabul, bilgilerin model veya hücrelerin aktiviteleri olarak saklanmasıdır; ki bu bilgiler durum vektörü olarak ifade edilir. Dolayısıyla sinir ağının çıkışı bir tek hücrenin değil tüm nöronların karşılıklı etkileşimleri sonucu elde edilen bir cevaptır. Çıkış vektörünün herbir bileşeni y_i, bağlantılar üzerindeki ağırlıklar ve x_i girişleri ile hesaplanır. Matematiksel olarak toplama ünitesinin çıkışı, ağırlık ve giriş matrislerinin iç çarpımına eşittir. Doğrusal birleştiricide aktivasyon fonksiyonu lineer bir fonksiyondur. Doğrusal birleştirici istenilen çıkış modelini vermek için giriş modelini içermek zorundadır. Bunun için verilen modeldeki ağırlıklar eğitim yoluyla belirlenir. Teorik olarak ilk ağırlık parametrelerinin herhangibir değeri olmayabilir, fakat rastgele küçük ağırlık değerleriyle başlamak daha avantajlıdır.[7]

Sinir hücrelerinin aktivasyon fonksiyonu lineer ise, yapay sinir ağıda doğrusal birleştiricidir. Yapay sinir ağına uygulanan eğitimden sonra çıkış modeli ve onu üreten giriş modeliyle arasındaki matematiksel ilişki aşağıda verilmiştir.

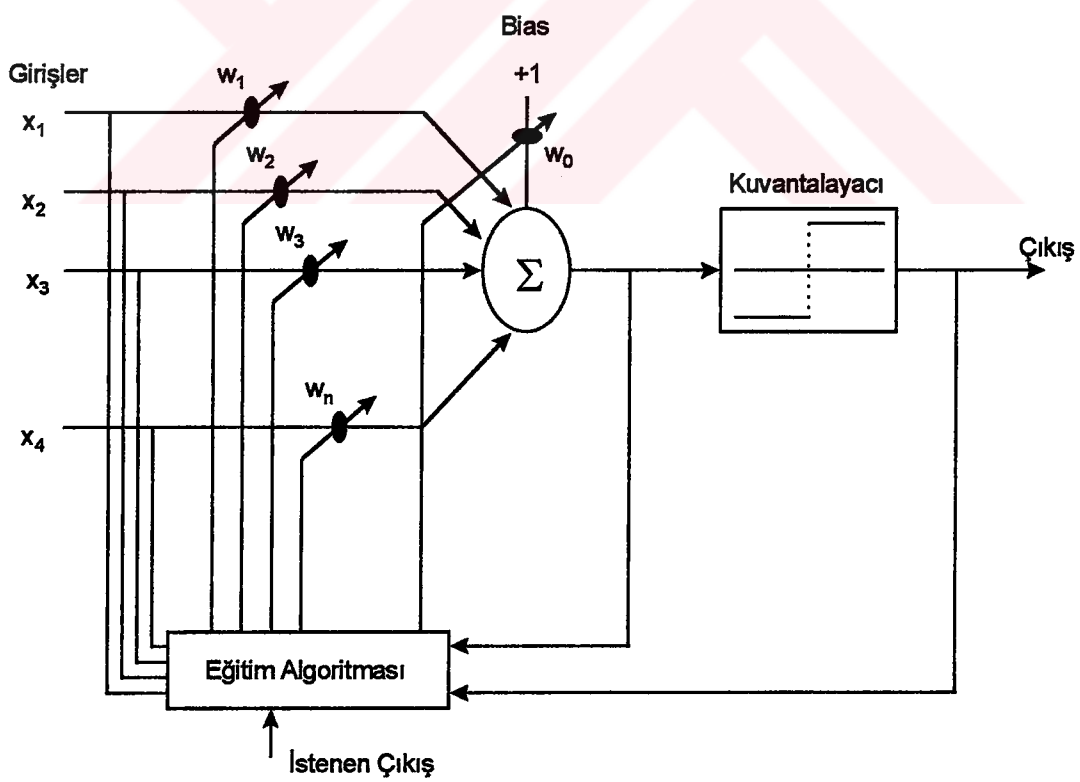
$$Y_i=W*X_i \tag{4.10}$$

Lineer bir birleştiricinin önemli avantajlarından biri de eş zamanlı olarak birden fazla matematiksel ilişkiyi saklayabilmesi; en büyük dezavantajı ise pekçok bilgiyi aynı anda saklamaya çalıştığından tam doğru yanıt verememesidir. Basit ağ yapılarında bazı

aktivasyon fonksiyonları ile hesaplama yapmayan *Hebbian learning* methodu kullanılabilir. Burada tıpkı *Widrow-Hoff learning* yönteminde olduğu gibi alınan çıkış değeri ile istenen çıkış değeri arasındaki fark (hata) gözönüne alınarak ağırlıklarda düzeltme yapılır. Neticede bu prosedür ile denetlenen bir öğrenme mekanizması geliştirilir.[12]

4.1.10 Widrow un adaline modeli

Adaline (adaptiv lineer element), eğiticili öğrenme sistemi ile hata işaretini minimize eden bir sinir ağı yapısıdır. Çeşitli giriş bilgisi modelleri için bir tür filtre rolü oynar. Basit olarak bir adaline modeli şekil 4.7 de verilmiştir. Şekildeki kuvantalayıcı [-1 1] aralığında eşik tipi non-lineer bir fonksiyondur. Eğer ağırlıklı girişlerin toplamı pozitif ise çıkış +1, negatif veya sıfır ise çıkış -1 dir. Eğitim algoritmasını kullanarak ağırlıkları düzeltmek için gerekli hata fonksiyonu üretilir. Hata fonksiyonu istenilen çıkış değeri ile çıkış değerlerinin toplamı arasındaki farktır.[9]



Şekil 4.7 Adaline işlem elemanının diyagramı

Eğitimin başında ağırlık değerleri rastgele belirlenir. Eğitim methodu delta yöntemi olarak tanımlanır ve delta vektörü hesaplanarak, elemanların çıkış hata fonksiyonlarına göre ağırlıklar düzeltilir.

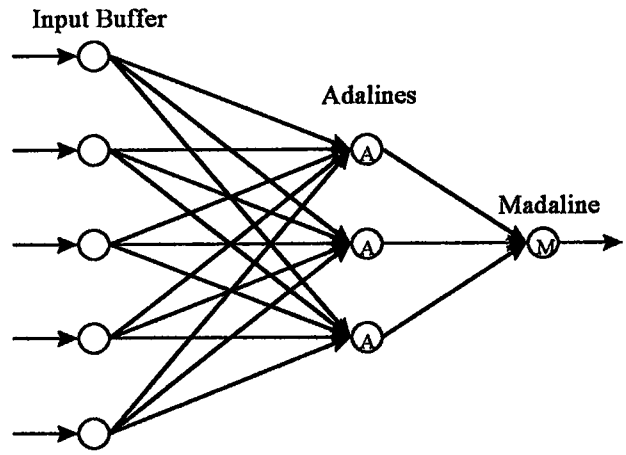
$$\Delta w_i = \frac{\eta.\varepsilon.x_i}{|X|^2} \tag{4.11}$$

Burada ε hata fonksiyonu, η öğrenme sabiti, x_i i. giriş (-1 veya +1) ve X giriş vektörüdür. Adaline için eğitim algoritması tek elemanlı veya sinir hücreli yapı için giriş uygulaması içerir. İstenilen çıkışın uygulanması ve hatanın hesaplanması, istenilen çıkış ile kuvantalayıcı öncesindeki ağırlıklar toplamı arasındaki fark olarak tanımlanır. Ağırlıklar her çevrimde düzeltilerek hata tüm ağırlık değerleri içinde yaklaşık eşit olarak dağıtılır. Biasla (polarizasyon değeri) birlikte (N+1) giriş olduğu gözönüne alınırsa öğrenme sabiti h nün yerine 1/(N+1) konulabilir. Bu da hatanın (N+1) girişe üniform olarak dağıtılması anlamına gelir.

$$\Delta w_i = \frac{\eta.\varepsilon.x_i}{|X|^2} = \frac{\varepsilon.x_i}{(N+1).|X|^2} \tag{4.12}$$

Tüm girişler +1 veya -1 olduğunda bu değerler sabitlenmiş ağırlık değerlerinden çıkarılır veya toplanır. Bu işlem hata istenilen minimum değere çekilinceye kadar sürdürülür. Hem kuvantalayıcı çıkışı hem de istenilen çıkış ikilik düzende olduğundan hatanın sıfir yapılması da mümkündür. Yani eğitim aşamasından sonra giriş işaretlerinde düzensiz değişimlere neden olan hatanın minimize edilmesi ile istenilen çıkış kuvantalayıcı çıkışına eşitlenebilir.

Adaline da eğitim işleminin yakınsaması genelde çok hızlı gerçekleşir. Tabii ki rastgele girilen ilk değerlerin de yakınsama hızına etkisi büyüktür. Çok sınırlı sayıdaki durumlarda bu yakınsama gerçekleşmez.



Şekil 4.8 Madaline yapısına ilişkin diyagram

4.1.11 Madaline (many adaline)

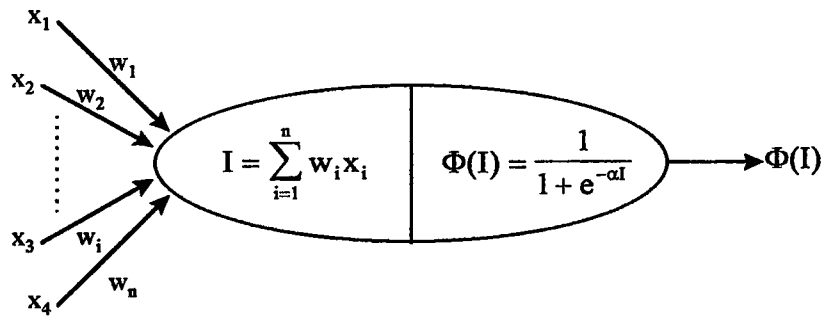
Madaline (doğrusal çağrışımlı bellek), üç tabakalı yapay sinir ağının orta tabakasında çok sayıda adaline (doğrusal uyarlanır eleman) kullanımına imkan tanıyan bir yapıdır. Şekil 4.8 de görüldüğü gibi çıkış tabakası tek hücrelidir ve bu çıkış tabakasının girişleri, orta tabakada bulunan çok sayıdaki adaline modüllerinin çıkışlarıdır. Adaline modüllerdeki tüm çıkışlar +1 ise ise madaline +1, aksi durumda -1 dir. Yine de bu tanımlamalara rağmen genelde çıkış, +1 ler yoğunluktaysa +1 olarak, -1 ler yoğunluktaysa -1 olarak tayin edilir. Madaline çıkışı ikilik düzende ifade edildiği için genelde iki tür veya sınıfı birbirinden ayırmak için kullanılır. Elbette çok sayıda tür veya sınıf arasındali farkı belirlemek sözkonusu olduğunda, o zaman herbir çift sınıf veya tür için modele bir adaline modülü ilave edilir. Madaline toplayıcısının girişlerinde bias yoktur ve dolayısıyla giriş üzerindeki ağırlıklar sabittir.[9]

4.2 Geriye Yayılım ve Eğitim Algoritmaları

4.2.1 Geriye yayılım ile eğitim

Geriye yayılım, çok tabakalı (üç veya daha fazla) yapay sinir ağı için geliştirilen ve bugün mevcut uygulamaların %80 inde kullanılan sistematik bir yöntemdir. Bazı dezavantaj teşkil edecek sınırlamalara rağmen, belki de güçlü bir matematiksel

altyapıya sahip olduğundan komplekslik derecesi yüksek problemlerin çözümünde tercih edilen bir yöntemdir.[4]



Şekil 4.9 Sinir Hücresi

x_i girişler, w_i ağırlıklar ve $\Phi(I)$ aktivasyon fonksiyonu olmak üzere bir sinir hücresi yapısı şekil 4.9 da gösterilmiştir.

$$I = \sum_{i=1}^{n} x_i w_i = x_1 w_1 + x_2 w_2 + \ldots\ldots\ldots\ldots + x_n w_n \tag{4.13}$$

$$\Phi(I) = \frac{1}{1+e^{-\alpha I}} = (1+e^{-\alpha I})^{-1} \tag{4.14}$$

Yukarıdaki aktivasyon fonksiyonu 0 veya -1 minimum değerlerinden +1 maksimum değerine ulaşan sigmoid (S eğrileri) fonksiyonudur. Bu eğriler farklı α değerlerine göre çizilmiştir. 'I' eksi veya artı sonsuza yaklaştığında fonksiyonun türevinin asimptotik olarak sıfıra yakınsadığı ; I=0 iken yani $\alpha/4$ değerinde ise türevin maksimum değerini aldığı görülür.

Geriye yayılımda bu türev fonksiyonu kullanılacağından daha basit bir forma getirilecek olursa

$$\frac{\partial \Phi(I)}{\partial I} = (-1)(1+e^{-\alpha I})^{-2}.e^{-\alpha I}.(-\alpha) \tag{4.15}$$

$$= \alpha.e^{-\alpha I}(1+e^{-\alpha I})^{-2} = \alpha.e^{-\alpha I}.\Phi^2(I) \tag{4.16}$$

(4.14) eşitliği $e^{-\alpha I}$ için çözülürse (4.15) eşitliği aşağıdaki formu alır.

$$\frac{\partial\Phi(I)}{\partial I} = \alpha\frac{1-\Phi(I)}{\Phi(I)}\Phi^2(I) = \{\alpha.[1-\Phi(I)].\Phi(I)\} = \alpha.(1-\Phi).\Phi \qquad (4.17)$$

Geriye yayılım algoritmasında, her noktada türevi olan ve ağırlıklı girişlerin toplamı I ile monoton artan tüm non-lineer fonksiyonlar kullanılabilir. Sigmoidal, logistic, hiperbolik tanjant ve arctanjant fonksiyonları bu şartları sağlar. Arctanjant fonksiyonu için,

$$\Phi(I) = \frac{2}{\pi}\tan^{-1}(\alpha.I) \qquad (4.18)$$

$2/\pi$, fonksiyonun genliğini azaltmak için kullanılan bir katsayı, α ise fonksiyonların değer aralığının -1, +1 sınırları arasındaki değişim oranını belirleyen bir sabittir. Yukarıdaki fonksiyonun orjindeki eğimi $2\alpha/\pi$ dir. Arctanjant fonksiyonu Şekil 4.10 da görüldüğü üzere sigmoidal (S eğrisi) formdadır ve türevi aşağıdaki gibidir.

$$\frac{\partial\Phi(I)}{\partial I} = \frac{2}{\pi}\left[\frac{\alpha}{1+\alpha^2.I^2}\right] \qquad (4.19)$$

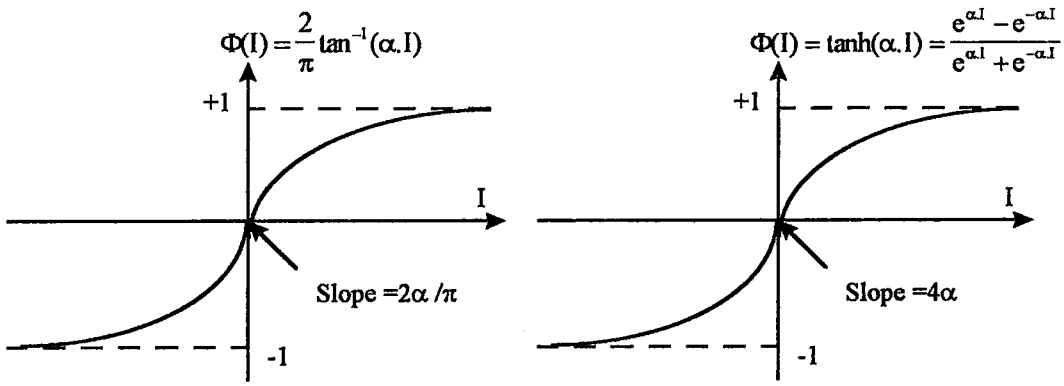
Aktivasyon fonksiyonu olarak kullanılabilecek bir diğer non-lineer fonksiyon olan hiperbolik tanjant ve türevi ifadesi aşağıda verilmiştir.

$$\Phi(I) = \tanh(\alpha.I) = \frac{e^{\alpha.I} - e^{-\alpha.I}}{e^{\alpha.I} + e^{-\alpha.I}} \qquad (4.20)$$

$$\frac{\partial\Phi(I)}{\partial I} = \alpha.\sec h^2(\alpha.I) \qquad (4.21)$$

$\Phi(I)$ nın orjindeki eğimi 4α dır ve a fonksiyonun -1, +1 değerleri arasındaki değişim oranını belirler.

Sigmoit fonksiyonu sıfıra yakın çok küçük değerler için, türevi çan eğrisi şeklinde olduğundan yüksek kazanç kontrolü sağlar. Ağırlıklı girişler toplamı I, pozitif veya negatif yönde artarsa kazanç azalır.



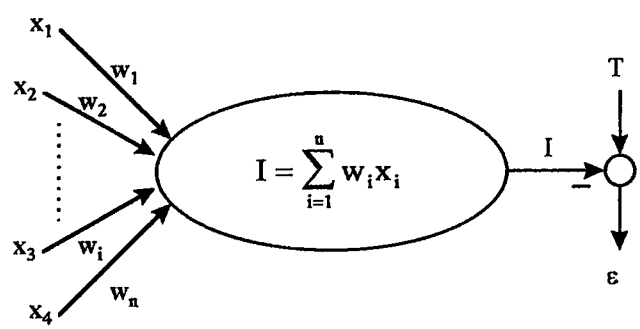
Şekil 4.10 Geriye yayılım için (a) Arctanjant (b) Hiperbolik tanjant fonksiyonu

4.2.2 Widrow-hoff delta öğrenme kuralı

Widrow Delta öğrenme kuralına ilişkin hücre yapısı, şekil 4.12 de verilmiştir ve görüldüğü üzere bir aktivasyon fonksiyonu veya kuvantalayıcıya ihtiyaç yoktur.

$$I = \sum_{i=1}^{n} x_i w_i \qquad (4.22)$$

Elde edilen sonuçlar, non-lineer elemanlar kullanıldığında alınan sonuçlarla aynı derecede geçerlidir. ε hata fonksiyonu olup Şekil 4.11 deki gibi tanımlıdır.[2]



Şekil 4.11 Aktivasyon fonksiyonu içermeyen sinir hücresi

Hata fonksiyonunun karesinin gradyanı, herbir i. ağırlığa göre kısmi türevidir.

$$\frac{\partial \varepsilon^2}{\partial w_i} = -2(T-I)\frac{\partial I}{\partial w_i} = -2(T-I).x_i \tag{4.23}$$

Yalnız x_1 ve x_2 gibi iki giriş için yukarıdaki eşitlikler tekrar yazılacak olursa,

$$\begin{aligned}\varepsilon^2 &= \left[T - w_1x_1 - w_2x_2\right]^2 = T^2 + w_1^2x_1^2 + w_2^2x_2^2 - 2Tw_1x_1 - 2Tw_2x_2 + 2w_1x_1w_2x_2 \\ &= w_1^2\left[x_1^2\right] + w_1\left[-2x_1(T - w_2x_2)\right] + \left[\left(T - w_2x_2\right)^2\right] \\ &= w_2^2\left[x_2^2\right] + w_2\left[-2x_2(T - w_1x_1)\right] + \left[\left(T - w_1x_1\right)^2\right]\end{aligned} \tag{4.24}$$

$$\frac{\partial \varepsilon^2}{\partial w_1} = -2\left[T - w_1x_1 - w_2x_2\right].x_1 = 0 \tag{4.25}$$

$$\frac{\partial \varepsilon^2}{\partial w_2} = -2\left[T - w_1x_1 - w_2x_2\right].x_2 = 0 \tag{4.26}$$

(4.25) ve (4.26) eşitliklerinde giriş vektörleri sıfır olamayacağından parantez içinin sıfır olması gerektiği aşikardır.

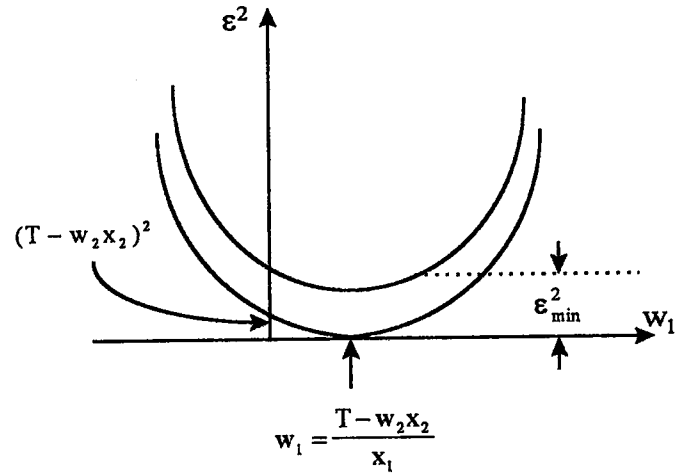
$$T - w_1x_1 - w_2x_2 = 0 \tag{4.27}$$

$$w_1 = \frac{T - w_2x_2}{x_1} \tag{4.28}$$

$$w_2 = \frac{T - w_1x_1}{x_2} \tag{4.29}$$

Bu değerler (4.24) eşitliğinde yerine konursa minimum ε^2 nin sıfır olması gerektiği görülür. Teorik olarak bu doğruda olsa gerçek dünyada gürültü, non-lineerlik, veri hatası gibi sebepler dolayısıyla minimum ε^2 asla sıfır olmaz.

Yukarıdaki (4.24) eşitliğinde de görüldüğü gibi w_1 veya w_2 ye göre ε^2 parabolik bir eğridir. Şekil 4.12 de minimum ε^2 nin sıfır olduğu ve olmadığı durumlar için geçerli eğriler verilmiştir. Her iki durum için minimum ε^2 (4.28) ile verilen w_1 değerinde oluşur. Aynı sonuç (4.29) ile verilen w_2 noktası için de elde edilebilir. Bu yüzden minimum ε^2 her iki ağırlık için parabolik bir eğri verir.



Şekil 4.12 Widrow-Hoff eğitimi boyunca e^2 nin minimizasyonu

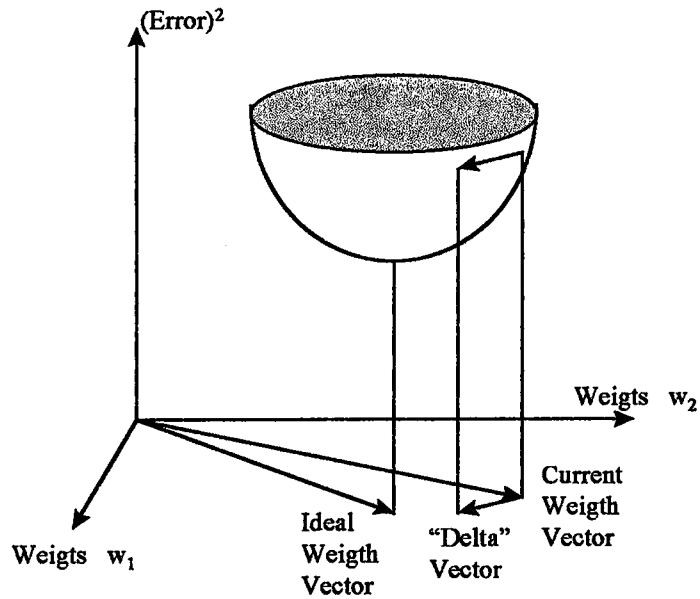
Delta kuralının geometrik açıklaması ε^2 nin minimize edilmesi için gradyen izdüşüm algoritmasının yürütülmesidir. ε^2 üç boyutta (w_1, w_2, ε^2) düşünüldüğünde ε^2 alanı, ağırlık vektörünün parabol yüzeyi üzerindeki bir gradyen vektörü boyunca aldığı minimum değere izdüşüm ile oluşan parabolik yörünge hareketinin ortaya koyduğu alandır. w_1 - w_2 düzlemindeki gradyen vektörünün izdüşümü olan 'delta vektörü' şekil 4.13 de gösterilmiştir. Delta kuralı ağırlık vektörünü, eğri yüzeyinin negatif gradyeninden ideal ağırlık vektörüne kadar aynı yörünge boyunca hareket ettirir. Bu vektör gradyeni izlediğinden, gradyen izdüşüm veya dik izdüşüm algoritması olarak tanımlanır. Eğri yüzeyinin en alt noktasına giden en iyi yol gradyen olduğu için de ε^2 nin minimize edilmesinde en iyi yol delta kuralının kullanılmasıdır.

Widrow-Hoff Delta eğitimi, herbiri kendi negatif gradyen ile orantalı olan ağırlık vektörü bileşenlerindeki değişimi belirler.

$$\Delta w_i = -K\frac{\partial \varepsilon^2}{\partial w_i} = 2K.(T-I).x_i = 2K\varepsilon.x_i \qquad (4.30)$$

K orantı sabiti ve $\eta=2.K.|X|^2$ olmak üzere norm tanımı kullanılırsa, aşağıdaki ifade elde edilir.

$$\Delta w_i = \left[2.K.|X|^2\right]\frac{\varepsilon.x_i}{|X|^2} = \frac{\eta.\varepsilon.x_i}{|X|^2} \qquad (4.31)$$



Şekil 4.13 Delta kuralının geometrik ifadesi

4.2.3 Çok katmanlı yapay sinir ağlarında geriye yayılım ile eğitim

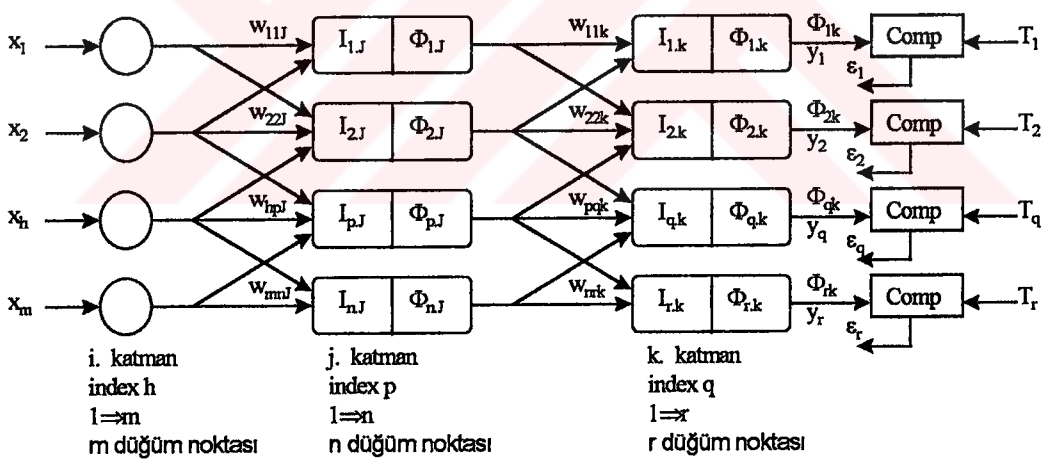
İki katmanlı bir sinir ağında giriş tabakasındaki bilgi veya sunum ne ise çıkıştan da özdeş sonuçlar alınır. Eğer girişler süreksiz veya lineer değilse bu durumda çıkış değerleri tutarsızlaşır ve algoritma çerçevesinde optimum değerlere yakınsayacak bir eğitim gerçekleşemez. Bu nedenledir ki sinir ağlarında orta katmanların önemi büyüktür. Sinir ağında bir veya daha fazla orta katmanın yer alması, lineer olmayan girişlerin mevcut olduğu durumlarda da hemen her tür eğitim algoritmasını yürütebilecek sunum kapasitesi sağlar kullanıcıya.[7]

Bir ağın orta katmanındaki sinir hücresi sayısı Kolmogorov teoremi ile belirlenir. Bir yapay sinir ağı yapılandırılırken amaç, m boyutlu bir reel vektörü bir diğer n boyutlu reel vektöre dönüştürecek uygun hücre ve ağırlık konfigürasyonunu oluşturabilmektir. Üç tabakalı bir sistemde giriş vektörlerinin 0 ve 1 değerleri arasında değiştiğini, çıkış değerlerinde ise bir sınırlama olmadığını varsayalım. Kolmogorov teoremine göre giriş tabakasında m tane, çıkış tabakasında n tane hücre bulunduran bir ağın orta katmanında 2m+1 tane hücre olmalıdır. Yine aynı teoreme göre üç tabakalı bir ağ non-lineer olarak ayrıştırılabilir tüm problemleri çözebilir. Ancak bu, bir yapay sinir ağının yapılandırılmasındaki en uygun yöntem, problemi çözmek için gerekli en küçük veya en basit yapılanmayı sağlayacağı anlamına gelmez.

Tüm aktivasyon fonksiyonlarının logistic fonksiyonlar olduğu üç tabakalı bir ağ şekil 4.14 de gösterilmiştir. Bilindiği üzere orta katmandaki tabaka sayısı ne olursa olsun geriye yayılım methodu uygulanabilir. Eğitimin amacı ağırlıkları düzelterek optimum değerleri yakalamak ve istenilen çıkışı elde etmektir. Bunu gerçekleştirmek için bazen çok sayıda giriş-çıkış çiftleri kullanılması gerekebilir.

Eğitim prosedürü:

1. Ağırlıkların büyük değerler alması durumunda ortaya çıkabilecek doyma etkisini ortadan kaldırmak için başlangıçta rastgele küçük değerler girilir.
2. Eğitim için bir giriş çıkış çifti seçilir.
3. Giriş vektörü ağ girişine uygulanır.
4. Ağ çıkışı hesaplanır.
5. Ağ çıkışı ve istenilen çıkış arasındakihata hesaplanır.
6. Hatayı minimize etmek için ağırlıklar düzeltilir.
7. 2.-6. Adımlar arasındaki işlemler, hata kabul edilebilir küçük bir değere getirilinceye kadar herbir giriş-çıkış çifti için tekrarlanır.



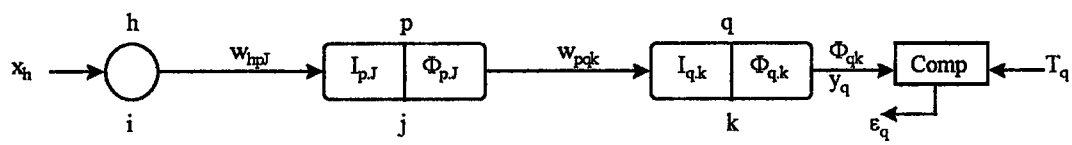
Şekil 4.14 Çok tabakalı sinir ağına uygulanan geriye yayılım eğitim algoritması

Bir yapay sinir ağının eğitiminde izlenecek iki yol vardır. İleri yol: Giriş işaretlerinin ağ girişinden çıkışa doğru ileri yönde yayılımı; geri yol: hesaplanan hatanın ağ boyunca geriye doğru yayılımı. Hesaplanan çıkış ileri yönde bir tabakadan diğerine taşınır. Bir tabakanın çıkışı diğerinin girişidir. İstenilen çıkışı elde etmek için ağırlıkların düzeltilmesi gerekir ve bunun için delta kuralı çerçevesinde geri yöndeki yol kullanılır. orta katmandaki ağırlıklar düzeltilirken asıl sorun bu ağırlıklar için referans olabilecek

istenilen bir değerin verilmemesidir. Bunun için de eğitim aşaması non-lineer fonksiyonların da etkisiyle çok karmaşık bir hal alır.

4.2.4 Çıkış katmanındaki hücreler için ağırlıkların hesaplanması

Şekil 4.15 deki yapay sinir ağından aşağıdaki kesit alınıp k. tabakadaki hücrenin çıkış değeri istenilen çıkış değerinden çıkarılacak olursa aşağıdaki hata işareti ve karesi elde edilir.[4]



Şekil 4.15 Çıkış katmanı için geriye yayılım ile ağırlık hesabı diyagramı

$$\varepsilon = \varepsilon_q = \left[T_q - \Phi_{q,k}\right] \tag{4.32}$$

$$\varepsilon^2 = \varepsilon_q^2 = \left[T_q - \Phi_{q,k}\right]^2 \tag{4.33}$$

Delta kuralı eğitim sırasında ağırlık değerindeki değişimin, ağırlığa bağlı olarak ε^2 deki değişim oranı ile orantılı olduğunu gösterir.

$$\Delta w_{pqk} = -\eta_{pq} \frac{\partial \varepsilon_q^2}{\partial w_{pqk}} \tag{4.34}$$

η_{pq} orantı sabiti olup öğrenme oranı olarak da tanımlanabilir. Yukarıdaki kısmi türev için zincir kuralı uygulanırsa,

$$\frac{\partial \varepsilon_q^2}{\partial w_{pqk}} = \frac{\partial \varepsilon_q^2}{\partial \Phi_{qk}} \frac{\partial \Phi_{qk}}{\partial I_{qk}} \frac{\partial I_{qk}}{\partial w_{pqk}} \tag{4.35}$$

Yukarıdaki ε_q^2 eşitliğinin Φ_{qk} ya göre kısmi türevi alınırsa,

$$\frac{\partial \varepsilon_q^2}{\partial \Phi_{qk}} = -2\left[T_q - \Phi_{qk}\right] \tag{4.36}$$

Daha önce elde edilmiş olan (4.17) eşitliği hatırlanacak olursa,

$$\frac{\partial \Phi_{qk}}{\partial I_{qk}} = \alpha.\Phi_{qk}\left[1 - \Phi_{qk}\right] \tag{4.37}$$

Şekil 4.15 den I_{qk} nın orta katmandaki ağırlık girişleri toplamı olduğu gözükmektedir.

$$I_{qk} = \sum_{p=1}^{n} w_{pqk}\Phi_{pj} \tag{4.38}$$

I_{qk} nın w_{pqk} ya göre kısmi türevi,

$$\frac{\partial I_{qk}}{\partial w_{pqk}} = \Phi_{pj} \tag{4.39}$$

Yukarıdaki bağıntılar (4.35) deki zincir diferansiyel denkleminde yerine konursa,

$$\frac{\partial \varepsilon_q^2}{\partial w_{pqk}} = -2.\alpha\left[T_q - \Phi_{qk}\right]\Phi_{qk}\left[1 - \Phi_{qk}\right]\Phi_{pj} = -\delta_{pqk}\Phi_{pj} \tag{4.40}$$

δ_{pqk} aşağıdaki gibi tanımlanabilir.

$$\delta_{pqk} = -2.\alpha.\left[T_q - \Phi_{qk}\right]\Phi_{qk}\left[1 - \Phi_{qk}\right] = 2\varepsilon_q \frac{\partial \Phi_{qk}}{\partial I_{qk}} \tag{4.41}$$

(4.40) eşitliği, (4.34) de yerine konursa (4.42) ve N iterasyon sayısı olmak üzere (4.43) denklemleri elde edilir.

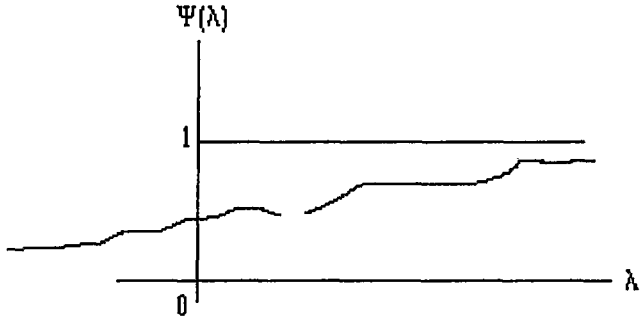
$$\Delta w_{pqk} = -\eta_{pq} \frac{\partial \varepsilon_q^2}{\partial w_{pqk}} = -\eta_{pq} \delta_{pqk} \Phi_{pj} \tag{4.42}$$

$$w_{pqk}(N+1) = w_{pqk}(N) - \eta_{pq} \delta_{pqk} \Phi_{pj} \tag{4.43}$$

Düzeltilmiş ağırlık değerlerini bulmak için, aynı işlemler çıkış tabakasındaki tüm hücreler için tekrarlanır. (4.41) deki hata terimi δ_{pqk} çıkış tabakasındaki ağırlıkların düzeltilmesinde kullanılır. Bu eşitlik ile ağ içinde geriye yayılım algoritması çerçevesinde hata hesaplanır; çünkü hatalı ağırlık değerleri ve orta katmandaki hücrelerin hatalı çıkış işareti üretmeleri nedeniyle ağ çıkışında istenilen değerden sapmış değerler sözkonusudur.

Giriş ve çıkış tabakasındaki iki nöron arasındaki ağırlık büyük olduğunda ve hücrenin çıkışı büyük değerler aldığında, orta tabakadaki ağırlık değerlerinde büyük hatalar gözlenir; hatta orta tabakadaki hücrelerin çıkışları küçük olsa ve ağ çıkışındaki hataya fazla bir katkısı olmasa dahi. Bu nedenle hatayı azaltmak ve ağırlık değerlerindeki değişimleri yumuşatmak için bastırma (squashing) fonksiyonunun türevi uygulanır.

Bastırma(Squashing) Fonksiyonu: $\Psi(.) : \Re \rightarrow (0,1)$, azalmayan ve $\lim_{\lambda \to \infty} \Psi(\lambda) = 1$, $\lim_{\lambda \to -\infty} \Psi(\lambda) = 0$ dır. Fonksiyon süreksiz olabilir.

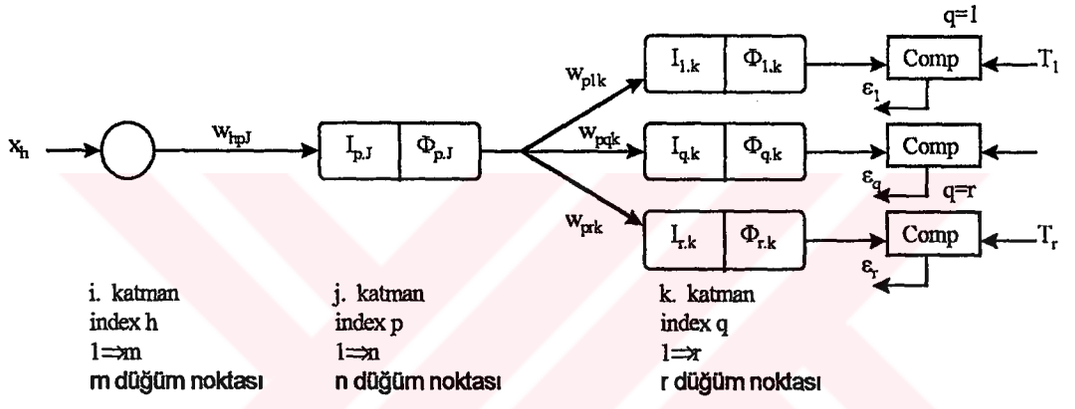


4.2.5 Orta katmandaki hücreler için ağırlıkların hesaplanması

Orta katmandaki hücre çıkışlarında ağırlıkların düzeltilebilmesi için gerekli referans değer sözkonusu olmadığından, geriye yayılım algoritmasının geliştirilmesine

dek değerlerin düzeltilmesi önemli bir problemdi. Geriye yayılım methodu ile orta tabakadaki tüm ağırlıklar eğitim süreci sonunda optimum değerlere çekilir. İstenilen sözkonusu olmaksızın üretilen δ_{hpj} hata terimi hariç tüm eşitlikler, çıkış tabakası için ağırlıkların belirlenmesinde kullanılan eşitliklerin aynısıdır. [4]

Şekil 4.16 de w_{hpj} ağırlığındaki değişimi belirlemek için geriye yayılım ile üretilen hata görülmektedir. Çıkış tabakasındaki tüm hata terimlerinin varlığından dolayı r tane çıkışa ait tüm kısmi türev ifadeleri mevcuttur. δ_{hpj} nin hesaplanması için izlenen prosedür, δ_{pqk} nın hesaplanmasında izlenen yol ile aynıdır. (4.34) eşitliğine benzer bir analoji ile delta kuralı eğitimi aşağıdaki bağıntılarla gerçeklenebilir.



Şekil 4.16 Orta katman için geriye yayılım ile ağırlık hesabı diyagramı

$$\Delta w_{hpj} = -\eta_{hp} \frac{\partial \varepsilon^2}{\partial w_{hpj}} = -\eta_{hp} \sum_{q=1}^{r} \frac{\partial \varepsilon_q^2}{\partial w_{hpj}} \tag{4.44}$$

Çok sayıda çıkış hatası sözkonusu olabileceğinden ε^2 aşağıdaki gibi tanımlanacak olursa,

$$\varepsilon^2 = \sum_{q=1}^{r} \varepsilon_q^2 = \sum_{q=1}^{r} \left[T_q - \Phi_{qk} \right]^2 \tag{4.45}$$

Öğrenme sabiti η_{hp} genelde η_{pq} ya eşit alınır; ancak her zaman eşit olması gerekmez. (4.44) de zincir kuralı uygulanacak olursa,

$$\frac{\partial \varepsilon^2}{\partial w_{hpj}} = \sum_{q=1}^{r} \frac{\partial \varepsilon_q^2}{\partial \Phi_{qk}} \frac{\partial \Phi_{qk}}{\partial I_{qk}} \frac{\partial I_{qk}}{\partial \Phi_{pj}} \frac{\partial \Phi_{pj}}{\partial I_{pj}} \frac{\partial I_{pj}}{\partial w_{hpj}} \tag{4.46}$$

$$\frac{\partial \varepsilon_q^2}{\partial \Phi_{qk}} = -2(T_q - \Phi_{qk}) = -2\varepsilon_q \tag{4.47}$$

$$\frac{\partial \Phi_{qk}}{\partial I_{qk}} = \alpha \Phi_{qk}(1 - \Phi_{qk}) \tag{4.48}$$

Daha önce verilmiş olan (4.38) eşitliğinin Φ_{pj} ye göre kısmi türevi alınacak olursa,

$$\frac{\partial I_{qk}}{\partial \Phi_{pj}} = w_{pqk} \tag{4.49}$$

(4.37) eşitliğindeki terimlerin indisleri orta tabakaya uyarlanırsa,

$$\frac{\partial \Phi_{pj}}{\partial I_{pj}} = \alpha \Phi_{pj}\left[1 - \Phi_{pj}\right] \tag{4.50}$$

Benzer şekilde yine (4.38) deki indisler değiştirilerek j. tabaka girişi Φ_{pj} yerine i. tabaka girişi x_h konursa,

$$I_{pj} = \sum_{h=1}^{m} w_{hpj} x_h \tag{4.51}$$

Yukarıdaki ifadenin kısmi türevi,

$$\frac{\partial I_{pj}}{\partial w_{hpj}} = x_h \tag{4.52}$$

Yukarıdaki eşitlikler yardımıyla (4.46) bağıntısı aşağıdaki gibi yeniden düzenlenir.

$$\frac{\partial \varepsilon^2}{\partial w_{hpj}} = \sum_{q=1}^{r} (-2).\alpha.(T_q - \Phi_{qk})\left[\Phi_{qk}(1 - \Phi_{qk})\right].w_{pqk}.\alpha.\left[\Phi_{pj}(1 - \Phi_{pj})\right].x_h$$

$$= -\sum_{q=1}^{r} \delta_{pqk} w_{pqk} \frac{\partial \Phi_{pj}}{\partial I_{pj}} x_h \qquad (4.53)$$

Şayet δ_{hpj} aşağıdaki gibi tanımlanacak olursa,

$$\delta_{hpj} = \delta_{pqk} w_{pqk} \frac{\partial \Phi_{pj}}{\partial I_{pj}} \qquad (4.54)$$

daha önce tanımlanan (4.53) eşitliği aşağıdaki forma gelir.

$$\frac{\partial \varepsilon^2}{\partial w_{hpj}} = -\sum_{q=1}^{r} \delta_{hpj} x_h \qquad (4.55)$$

(4.44) eşitliğinde de görüldüğü üzere ağırlıklardaki değişim, ε^2 nin ağırlığa göre değişiminin ters işaretlisiyle orantılı olduğundan (4.53) ve (4.54) bağıntıları bu denklemde yerine konursa,

$$\Delta w_{hpj} = -\eta_{hp} \frac{\partial \varepsilon^2}{\partial w_{hpj}} = \eta_{hp} \sum_{q=1}^{r} \delta_{pqk} w_{pqk} \frac{\partial \Phi_{pj}}{\partial I_{pj}} x_h$$

$$= \eta_{hp} x_h \sum_{q=1}^{r} \delta_{hpj} \qquad (4.56)$$

ve dolayısıyla,

$$w_{hpj}(N+1) = w_{hpj}(N) + \eta_{hp} x_h \sum_{q=1}^{r} \delta_{hpj} \qquad (4.57)$$

Yapay sinir ağı içinde birden çok orta tabaka varsa ağırlıkları düzeltmek için aynı işlemler giriş tabakasına kadar sürdürülür. İşlem tamamlandığında eğitim için yeni bir giriş vektörü ile proses tekrar işletilir ve kabul edilebilir bir hata değeri elde edilinceye kadar sürdürülür. Farklı girişler için bu süreç tamamlandığında eğitim aşaması artık sona ermiştir.

4.3 Yapay Sinir Ağları İle Sistem Tanıma

4.3.1 Referans model gösterilimi

Yapay sinir ağlarının eğitiminde, estimate edilen çıkış veya çıkışlardan hata fonksiyonunun belirlenerek geriye yayılım algoritmasının yürütülebilmesi için ayrık zamanda bir referans model oluşturulmalıdır. Aşağıda çok tabakalı ve çok girişli yapay sinir ağlarının eğitiminde kullanılabilecek referans model gösterilimlerine ilişkin dört farklı lineer olmayan fark eşitliği sunulmuştur.[7]

$$\text{i)}\quad y_p(k+1) = \sum_{i=1}^{n} \alpha_i y_p(k-i) + g\left[u(k), u(k-1), \ldots\ldots u(k-m+1)\right]$$

$$\text{ii)}\quad y_p(k+1) = f\left[y_p(k), y_p(k-1), \ldots\ldots y_p(k-n+1)\right] + \sum_{i=0}^{m-1} \beta_i u(k-i)$$

$$\text{iii)}\quad y_p(k+1) = f\left[y_p(k), y_p(k-1), \ldots\ldots y_p(k-n+1)\right] + g\left[u(k), u(k-1), \ldots\ldots u(k-m+1)\right]$$

$$\text{iv)}\quad y_p(k+1) = f\left[y_p(k), y_p(k-1), \ldots\ldots y_p(k-n+1); u(k), u(k-1), \ldots\ldots u(k-m+1)\right]$$

$[u(k),y_p(k)]$, sistem tanımlama referans modelinin k. andaki giriş çıkış çiftleridir ve $m \leq n$ dir. İkinci ve üçüncü model gösteriliminde $f: \Re^n \rightarrow \Re$; dördüncü modelde $f: \Re^{n+m} \rightarrow \Re$ ve $g: \Re^m \rightarrow \Re$ olmak üzere tanımlanan fonksiyonların her noktada türevi mevcuttur. Her dört modelde (k+1). andaki model çıkışı, daha önceki n adet çıkışa bağlıdır. Benzer şekilde k. andaki giriş fonksiyonu da daha önce modele uygulanmış olan m adet girişin bir fonksiyonudur.

4.3.2 Sistem tanıma

Neural ağ ile dinamik sistemi tanımlayan model çıkışları arasındaki hata fonksiyonu ile çıkış optimize edilmeye çalışıldığından dolayı tanımlama modelini uygun parametreler ile yapılandırmak önemlidir. Neural ağa ilişkin ağırlık matrislerinin mevcut olduğu varsayımı ile aynı başlangıç koşullarında belirli herhangibir giriş uygulandığında referans model ve yapay sinir ağı aynı çıkışı vermelidir ideal koşullarda. Bu da ancak geriye yayılım ile doğru ağırlık konfigürasyonunu oluşturacak

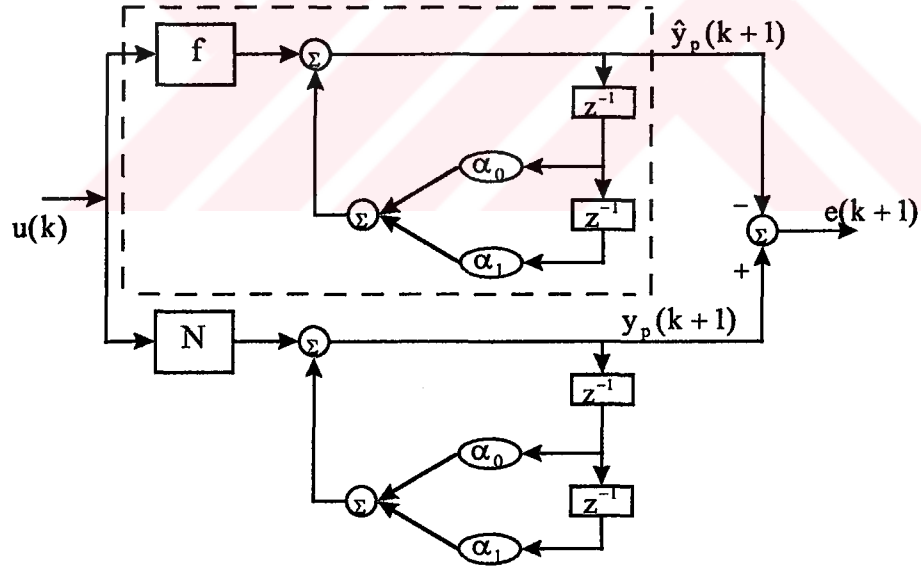
olan hata fonksiyonunun referans modele ilişkin çıkış bileşenini doğru bir biçimde verebilecek tanımlama modeli oluşturmakla mümkündür.

Daha açık bir tarifle yapay sinir ağının modeli tanımadaki başarısı, verilen lineer sistem modellerindeki katsayı matrislerinin bulunmasına veya lineer olmayan sistem modellerindeki f(.) fonksiyonuna yakın bir fonksiyonu bulmadaki yaklaşıklığına bağlıdır. Bu amaçla sistem tanımlama için kullanılan iki temel yapı olan "paralel tanımlama" ve "seri-paralel tanımlama" aşağıda verilmiştir. [7]

i) Paralel Tanımlama Modeli :

Daha önceki bölümde referans model gösterilimleri ile ilgili tanımlamalardan birincisi alınarak n=2, m=1 olarak seçilirse, çıkış aşağıdaki gibi olur. Bu tanımlamaya ilişkin blok şema 4.18 de verilmiştir.

$$\hat{y}_p(k+1) = \hat{\alpha}_o \hat{y}_p(k) + \hat{\alpha}_1 \hat{y}_p(k-1) + N[u(k)]$$



Şekil 4.17 Paralel tanımlama modeli

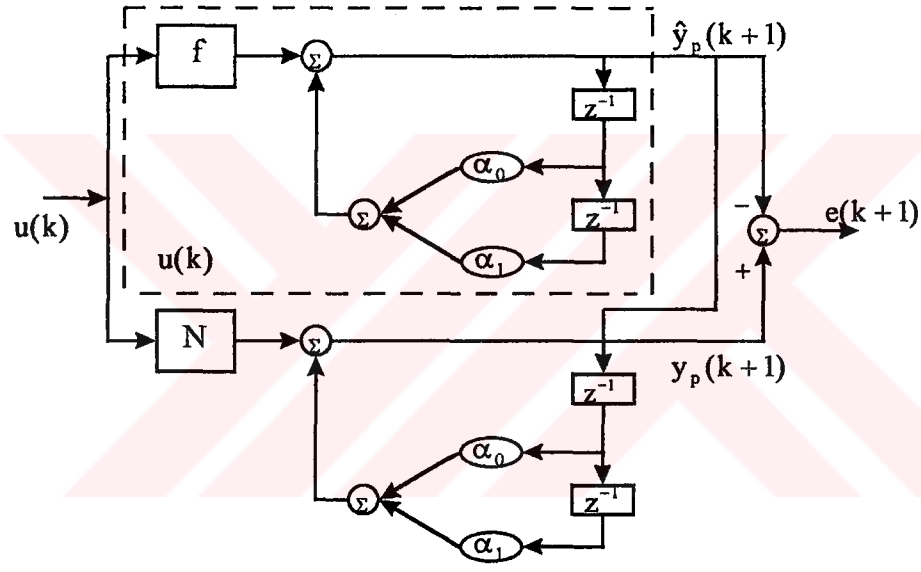
Modele girilen giriş parametreleri sınırlı bir şekilde üniform olarak dağıtıldığından modelin sınırlı giriş ve çıkışlar ile kararlı çıkış vermesi beklenir. Buna karşın yine de neural yapı için kararlılığı sağlayacak uygun modelin her zaman gerçeklenebileceği söylenemez. Dolayısıyla paralel tanımlama kullanıldığında

parametrelerin kararlı çıkış verecek şekilde estimate edileceği veya ağırlıkların, hata fonksiyonunu sıfıra götürecek biçimde güncellenebileceği garanti edilemez. Bunun içindir ki sistem tanımlamada daha ziyade seri-paralel tanımlama kullanılır. [7,8]

ii) Seri-Paralel Tanımlama Modeli :

Yukarıda tanımlanan modelin aksine seri-paralel modelde, tanımlama modelinin geçmiş anlardaki değerleri geribesleme ile Şekil 4.18 daki gibi ağa aktarılır.

$$\hat{y}_p(k+1) = \alpha_o y_p(k) + \alpha_1 y_p(k-1) + N[u(k)]$$



Şekil 4.18 Seri-Paralel Tanımlama Modeli

Bu nedenle paralel modele oranla pek çok avantaj sunar. Refarans modelden elde edilen gecikmeli çıkış değerleri geribesleme ile ağa verildiğinden estimate edilen çıkış bileşenleri sınırlanır; dolayısıyla kararlılığın sağlanması daha kolaydır. Çıkış hatası fonksiyonunun asimptotik olarak sıfıra yaklaştığı varsayılırsa $\hat{y}_p(k) \approx y_p(k)$ olduğu görülür. [7,8]

4.3.3 Rotor akısı yönlendirilmiş vektör denetimi için neural tabanlı akı modeli algılayıcıları

Daha önce bölüm 3 de açıklandığı gibi, rotor akısı yönlendirmeli vektör denetimi prosesi içinde rotor mıknatıslanma akısı genliğinin ve uzaysal konumunun bilinmesi gerekir. Normal şartlarda bu çıkışları elde etmek için akı gözleyicisi kullanılır. Bu bölümde ise akı modeli yerine kullanılacak bir neural yapı ile rotor mıknatıslanma akımı genliği ve açısal hızı büyük yaklaşıklıkla tahmin edilebilmektedir. [5]

Rotor akısı yönlendirilmiş eksen takımında akı modelini tanımlayan denklemler aşağıda tekrar verilmiştir.

$$T_r \frac{d|\bar{i}_{mr}|}{dt} + |\bar{i}_{mr}| = i_{sx} \tag{4.58}$$

$$\frac{d\rho_r}{dt} = \omega_{mr} = \omega_r + \omega_{sl} = \omega_r + \frac{i_{sy}}{T_r|\bar{i}_{mr}|} \tag{4.59}$$

Neural tabanlı gözleyici tasarlamanın amacı, mıknatıslanma akımı genliği ve açısal hızını belirleyecek bir yapay sinir ağı yapılandırarak, algılayıcı kullanmaktan kurtulmaktır. Bu sebeple önce ağın eğitimi için gerekli bir referans model oluşturmak amacıyla (4.58) ve (4.59) de verilen akı modeline ilişkin eşitlikler ayrık zamanda fark denklemleri olarak yazılır.

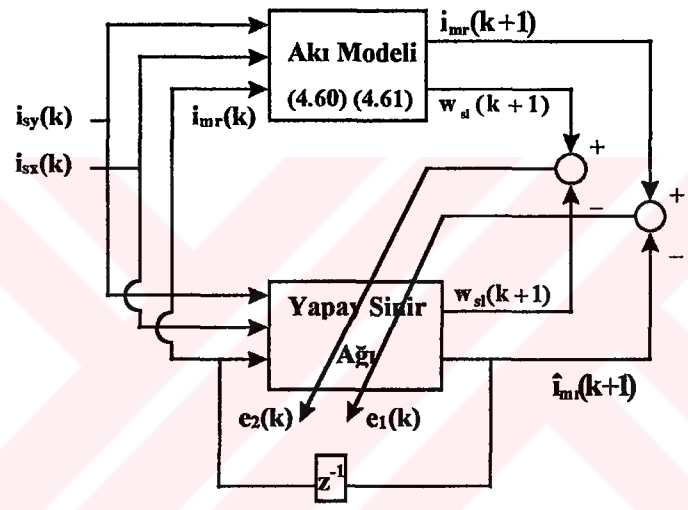
$$T_r \frac{I_{mr}(k+1) - I_{mr}(k)}{\Delta t} + I_{mr}(k) = I_{sx}(k)$$

$$\Rightarrow \quad I_{mr}(k+1) = \left(1 - \frac{\Delta t}{T_r}\right).I_{mr}(k) + \frac{N_{isx}}{N_{imr}}\frac{\Delta t}{T_r}.I_{sx}(k) \tag{4.60}$$

$$\omega_{mr}(k) = \omega_r(k) + \omega_{sl}(k) = \omega_r(k) + \frac{i_{sy}(k)}{T_r.I_{mr}(k)}$$

$$\Rightarrow \quad \omega_{sl}(k+1) = \frac{N_{isy}}{N_{wsl}.N_{imr}}\frac{i_{sy}(k+1)}{T_r.I_{mr}(k+1)} \tag{4.61}$$

Akı modelinin girişleri olan i_{sx}, i_{sy} rotor akısı yönlendirilmiş eksen takımındaki stator akımı x ve y ekseni bileşenleri, w_r ise rotor hızıdır. Akı modeli yerine akı modelinin çıkışlarını tahmin eden iki ayrı yapay sinir ağı düşünülmüştür. Bunlardan genlik estimasyonu yapan ağın off-line eğitimi esnasında kullanılan referans modelin (k+1). ayrık zamandaki genlik çıkışı, (k). andaki çıkışının ve stator akımı x ekseni bileşeninin fonksiyonudur. Mıknatıslanma akımının açısının estimasyonu ise dolaylı olarak açısal kayma hızı w_{sl} nin tahmin edilmesi şeklindedir. Tahmin edilen kayma hızı kutup sayısını da bağlı olarak rotor hızına eklenerek mıknatıslanma akımının açısal hızı hesaplanmıştır.



Şekil 4.19 Neural tabanlı gözleyicinin eğitimi amacıyla kullanılan blok şema

Akı genliğinin ve açısının tahmini için kullanılan her iki yapay sinir ağı da ileri yönlü dört katmanlı algılayıcılardır. Ağlar için hücre sayısı konfigürasyonu 2-10-10-1 şeklindedir. Yine her iki yapay sinir ağı da ağırlıklar ve eşik değerlerin hata fonksiyonundaki değişime göre ayarlandığı geriye yayılım algoritması ile eğitilmiştir. Ağların kararlı çıkışlar vermesi için gerek mıknatıslanma akımı ve gerekse açısal kayma hızı için belirlenen modellerde giriş vektörü örnekleme sayısı 250 seçilmiştir. Lineer olmayan bu sistemde hatanın kabul edilebilir bir toleransın altına çekilebilmesi için iterayon sayısı 50000 (1000 yaklaşım) olarak seçilmiştir. Off-line eğitimin sonunda görülmüştür ki; rotor akı modelinin çıkışları yapay sinir ağları ile oldukça iyi bir yaklaşıklıkla tahmin edilebilmiştir. Eğitim sonunda belirlenen ağırlık konfigürasyonu

eğitim programından alınarak sabit parametreli matrisler biçiminde kontrol prosesinin içine sokulacaktır.

Off-line eğitim algoritmasında giriş değişkenlerini sınırlamak amacıyla bir normalizasyon işlemi uygulanmıştır. Bilindiği gibi orta ve çıkış katmanlarındaki aktivasyon fonksiyonlarının değer aralığı -1,+1 aralığının dışında değildir. Dolayısıyla uygulanan giriş değişkenlerinin normalizasyonu için bu değişkenlerin negatif veya pozitif alternansta alabileceği tepe değerlere en az eşit veya daha büyük normalizasyon sabitleri seçilerek giriş vektörleri -1,+1 aralığına sokulmalıdır. Pik değerlere bakılırken daha önce yapılan vektör denetimi simülasyonundaki çıkış eğrilerinden yararlanılmıştır.

4.3.4 Rotor akısı yönlendirilmiş vektör denetimi için neural tabanlı rotor hızı algılayıcısı

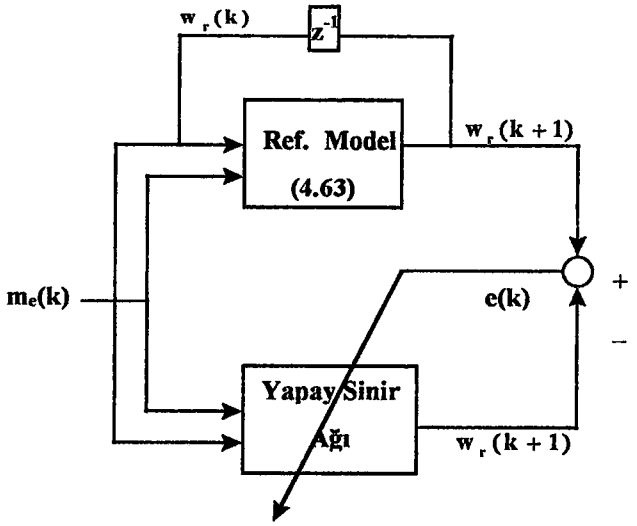
Vektörel kontrol için kullanılan şekil 2.7 deki blok diyagramında hız geribeslemesi mevcuttur. Çalışmanın bu bölümünde makinadan rotor hızı bilgisini almak amacıyla kullanılan encoder veya Hall-effect sensör yerine çok katmanlı ileri yönde bir algılayıcı modellenmiştir. Sinir hücrelerinin sayısı, akı modelinde olduğu gibi 2-10-10-1 şeklindedir. Ağın off-line eğitiminde kullanılan referans model (2.37) de verilmiş olan dinamik kısma ilişkin denklemin ayrık zamanda yeniden düzenlenmesiyle aşağıdaki gibi elde edilmiştir.

$$w_r(k+1) = w_r(k) + \frac{\Delta t}{J}\left[m_e(k) - m_l(k)\right] \qquad (4.62)$$

Yük momentinin λ sabit olmak üzere, $\underline{m_e(k) = \lambda . w_r(k)}$ biçiminde rotor hızı ile doğru orantılı olarak değiştiği gözönüne alınırsa rotor hızı referans modeli,

$$w_r(k+1) = \left(1 - \frac{\lambda . \Delta t}{J}\right).w_r(k) + \frac{\Delta t}{J}.\frac{N_{me}}{N_{wr}}.m_e(k) \qquad (4.63)$$

şeklinde tanımlanır. Eşitlikteki Δt örnekleme zamanı, N_{me} ve N_{wr} ise moment ile hız eğirilerinin tepe değerlerine en az eşit veya tepe değerlerden daha büyük seçilen normalizasyon sabitleridir.



Şekil 4.20 Neural tabanlı hız algılayıcısının eğitimi amacıyla kullanılan blok şema

(k+1). andaki rotor hızını tahmin eden neural yapının girişleri k. ayrık zamandaki moment ve hız değerine bağlıdır. Referans modelin giriş vektöründe ki gecikmeli bu hız bileşeni, sağlıklı bir estimasyon yapılmasını güçleştirmektedir. Bu nedenle neural yapının orta katmanındaki sinir hücrelerinin aktivasyon fonksiyonları akı modelinden farklı olarak sigmoital fonksiyon olarak seçilmiş ve ayrıca referans modelinin giriş vektörü için örnekleme sayısını artırma yoluna gidilmiştir. Neticede akı modelinde olduğu gibi 1000 yaklaşımı tamamlaması beklenmeden, 100. yaklaşımda hatanın $3.5*10^{-3}$ ün altına düştüğü görülerek eğitim kesilmiştir.

4.3.5 Sincap kafesli asenkron makinada neural tabanlı akı ve hız algılayıcıları ile vektör denetimi

Rotor akısı yönlendirmeli vektör denetiminde, akı modeli kullanılmasının amacı rotor halkalanma akısı veya mıknatıslanma akımının genliğini uzaysal konumunu belirlemektir. Bu amaçla akı modeli yerine kullanılacak akı gözleyicisi ve rotor hızını algılayan yapay sinir ağlarının off-line eğitimi bir önceki bölümde tamamlanmıştı. Eğitimin sonunda istenilen çıkışlara oldukça yakın çıkışlar veren ağlardan gerekli olan ağırlık matrisleri alınarak vektör denetimi algoritmasına konulmuştur.

Aşağıda akı modeli yerine kullanılan iki yapay sinir ağı içeren ve sadece neural tabanlı bir adet hız algılayıcısı içeren iki ayrı prosese ilişkin algoritmalar ve simülasyon sonuçları verilmiştir. Birinci sistemde mıknatıslanma akımını uzay fazörünün genliğini

ve açısal hızını estimate eden yapay sinir ağlarının büyük bir yaklaşıklıkla çıkışları tahmin edebildiği ve dinamik sistemin kararlığını bozmadan iyi bir performans gösterdiği görülmüştür. İkinci sisteme ilişkin algoritmada kullanılan neural tabanlı hız algılayıcısının ise ani hız değişimlerine çok daha süratli yanıt veren, yükselme zamanı oldukça kısa ve aşımı hemen hemen sıfır yapan çok daha verimli bir kontrol imkanı sağladığı görülmüştür. [10]

5. ADAPTİV KONTROL

5.1 Rotor Parametresi Değişimine Adapte Vektör Denetimi

Asenkron makinada rotor akısı yönlendirilmiş vektör kontrolünün başlıca dezavantajlarından biri de rotor zaman sabitinin değişimine duyarlı olmasıdır. Rotor zaman sabiti, akı modeli çıkışlarını doğrudan etkilediğinden bu parametre değişiminin gözlenmemesi veya kötü tahmin edilmesi kontrolün performansını olumsuz etkiler. Maalesef rotor sıcaklığı nedeniyle de rotor direncinin değişim aralığı oldukça geniştir. Bu problemi çözmek için geliştirilen yöntemler [5], [6] ve [10] nolu kaynaklarda mevcuttur. Bu yaklaşımlardan biri de MRAC (model reference adaptiv control) dır.

5.1.1 Rotor zaman sabitinin belirlenmesi

Rotor zaman sabitinin değişimini belirleyen matematiksel model (5.1) eşitliğinde verildiği gibi açısal kayma hızı, mıknatıslanma akımının genliği ve stator akımının dik eksen bileşeninin fonksiyonudur. Çalışmada bu parametrenin yeniden belirlendiği periyodik aralık 200 ms. olarak seçilmiştir. Başlangıçta tüm rotor parametrelerinin sabit olduğu varsayımı ile başlayan simülasyon algoritmasında her 200 ms de yeniden belirlenen zaman sabitine göre, akı modeli yerine kullanılan neural tabanlı gözleyicilerin on-line eğitilmesi ve uygun ağırlık konfigürasyonunun yeniden belirlenmesi gerekmektedir. On-line eğitim sürecinde açı ve genlik tahmini yapan ağların ağırlık matrisleri kendi hata fonksiyonları ile belirlenir. Ancak daha önce off-line olarak başarılı bir şekilde eğitimi yapılmış ağların ağırlıkları kararlı bir şekilde belirlenmiş olduğundan başka bir deyişle ilk koşullardaki ağırlıklar rastgele belirlenmeyeceğinden, online eğitim için gereken iterasyon sayısının daha küçük bir öğrenme sabitiyle daha az olması yeterlidir.[5]

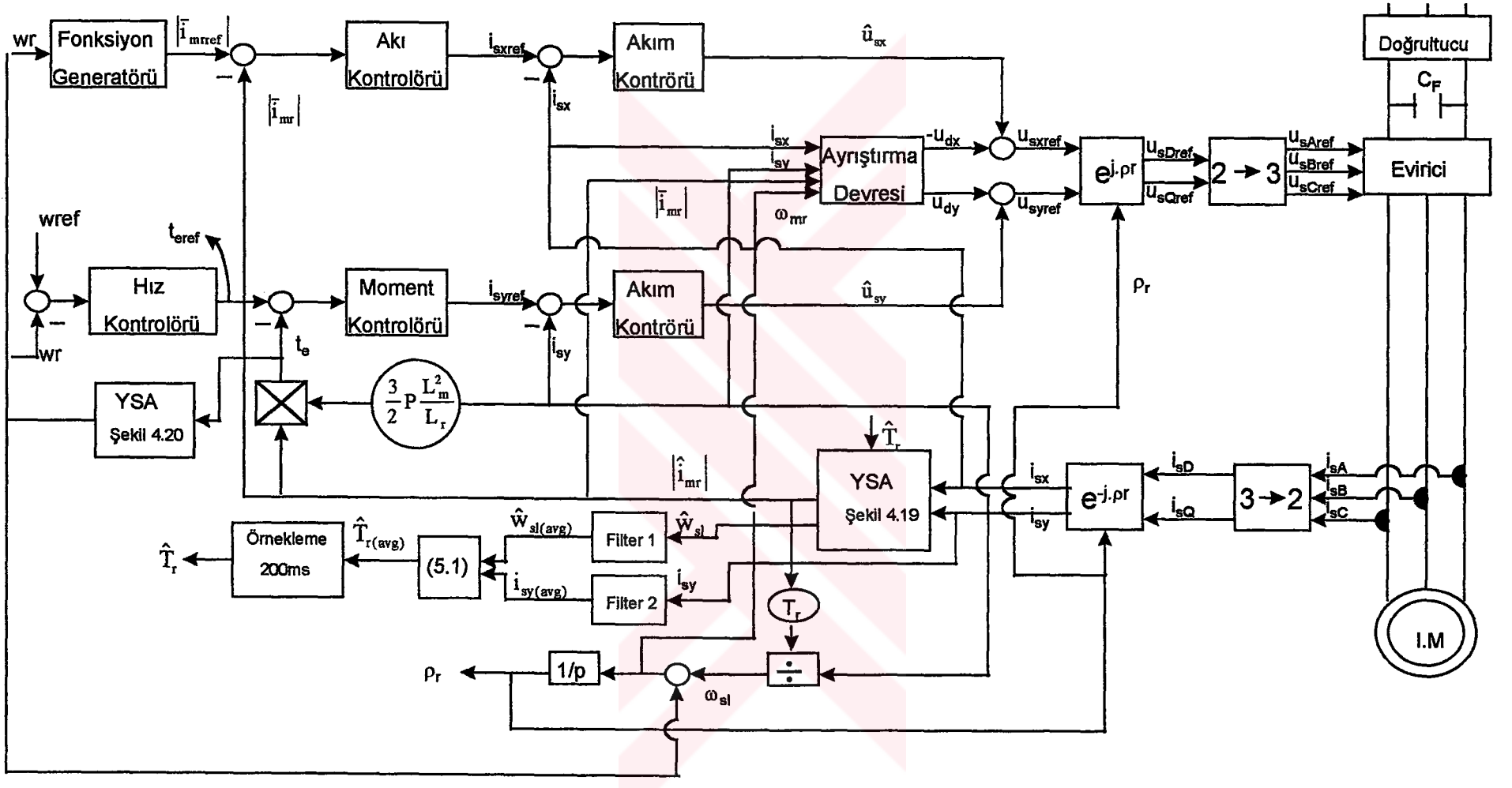
$$T_{r(avg)} = \frac{i_{sy(avg)}[k]}{w_{sl(avg)}[k].|i_{mr}[k]|} \qquad (5.1)$$

Bu bölümde rotor zaman sabitinin değişimi yukarıdaki matematiksel modele göre incelendiğinde nominal değerini 2/3 üne kadar düştüğü gözlenmiştir. Her 200 ms lik periyodun sonunda neural tabanlı algılayıcıların eğitimi ile daha önce yapılmış simülasyon sonuçlarına çok yakın neticeler alınmıştır.

5.1.2 Neural tabanlı algılayıcılar ile parametre adaptasyonlu vektör denetimi

Şekil (5.1) de tüm kontrol yapısına ilişkin detaylı bir blok diyagram verilmiştir. Buradaki rotor zaman sabitinin değişimini gözlemleyen blok her periyodun sonunda daha önceki i_{sy} ve w_{sl} parametrelerinin almış olduğu değerlerin ortalamasını hesaplayarak yeni zaman sabiti bilgisini akı modelindeki referans modellere verir. Neural tabanlı algılayıcılar da referans modellerin çıkışlarındaki değişime adapte olmak için on line döngüye girer. Burada pratik olarak tek sakınca bu denli hassas ve eviriciden kaynaklanan gecikmelerin bile ihmal edildiği bir durumda eğitim prosedürlerinin uzun sürmesidir. Kuşkusuz işlem hızı daha yüksek işlemcilerle daha hızlı ve daha verimli sonuçlar alınır.

Aşağıdaki T_r-tanımlama ünitesine göre yapılan simülasyonda rotor zaman sabitinin değişimi periyodik aralıklarla gözlenmektedir. T_r nin değişimini gösteren eğri simülasyon sonuçları bölümünde verilmiştir.



Şekil 5.1 Parametre adaptasyonlu neural tabanlı algılayıcılar ile rotor akısı yönlendirilmiş vektör denetimi blok şeması

5.1.3 Neural tabanlı algılayıcılar ile parametre adaptasyonlu vektör denetimi simülasyon model

Bu sistemde 3. bölümdekinden farklı olarak akı modeline yerine kullanılan neural tabanlı algılayıcılar ve bu algılayıcıların T_r değişimine göre belirli periyodik aralıklarda on-line eğitimi sözkonusudur. Sözü edilen periyodik aralıklarda rotor zaman sabitinin belirlenmesinde (5.1) eşitliğinden yararlanılmıştır. Yeni rotor zaman sabitine göre her 200 ms. lik periyodik aralığın sonunda akı genliği ve açısının tahmin eden ağlar on-line döngüye girmekte ve rotor zaman sabitindeki değişimi kompanze etmek amacıyla dolaylı olarak üretilmesi gereken yeni stator gerilimi bileşenleri hesaplanmaktadır. Bu prosese ilişkin algoritma şekil 5.1 den yola çıkılarak geliştirilmiş ve simülasyon sonuçlarıyla birlikte ilerleyen bölümde karşılaştırmalı olarak verilmiştir. [1,5]

6. SİMÜLASYON SONUÇLARI

Buraya kadar olan bölümlerde önce asenkron makinanın D-Q modeli kurulmuş, aynı makinanın rotor akısı yönlendirilmiş vektör kontrolü için kullanılan kontrol modülü yapılandırılmıştı. Daha sonra akı modeli ve hız sensörü yerine kulllanılacak neural tabanlı çok katmanlı algılayıcıların off-line eğitimi için oluşturulan referans model ve ağ yapısı verilmişti. Eğitimi yapılan neural ağlar vektör denetimi içine katılarak kontrol yapılmıştı. 5. Bölümde ise rotor zaman sabiti değişiminin dinamik sistem cevabı üzerindeki etkisini kompanze etmek amacıyla kullanılan neural ağların on-line eğitimi prosedürünü içeren vektörel kontrol için gerekli model kurulmuştu. Aşağıda verilen simülasyon sonuçları, yukarıda sözü edilen modeller için yazılmış olan algoritmaların, matlab 5.2 de koşturulmasıyla elde edilmiştir. İlgili algoritmalar ise çalışmanın sonunda ekler biçiminde verilmiştir

Simülasyonlarda kullanılan sincap kafesli asenkron makinanın plaka değerleri aşağıdaki gibidir. [13]

P_N (Nominal güç)	*3.7 kW.*
U_N (Faz arası nominal gerilim)	*380 V*
n (Nominal hız)	*1720 d./dak.*
J (Eylemsizlik momenti)	*$8*10^{-3}$ kg.m/s^2*
p (Kutup sayısı)	*2 ; (2p=4)*
R_s (Stator direnci)	*1.5 Ω*
R_r (Rotor direnci)	*1.6 Ω*
L_s (Stator indüktansı)	*109 mH*
L_r (Rotor indüktansı)	*115 mH*
M (Ortak indüktans)	*98 mH*

6.1 Makine Modeline İlişkin Simülasyon Sonuçları

Yukarıda plaka değerleri verilen üç fazlı sincap kafesli asenkron makinanın modeline ilişkin simülasyon sonuçları verilmiştir. Makinaya hiç bir kontrol uygulanmamıştır. Simülasyon sonuçları göstermiştir ki; geçici halde, makinaya ilişkin mıknatıslanma akımı ve elektriksel döndürme momenti eğrilerinde sıçramalar sözkonusudur. Hız-zaman eğrisinden de görüleceği üzere, motor yaklaşık 0.4 saniyede karalı çalışma noktasına gelmektedir.

Matlab editöründe derlenen ilgili algoritma Ek 1 de verilmiştir. Aşağıda simülasyon sonuçları verilmiş olan makinaya daha sonra vektör kontrolü uygulandığında, ani hız değişimlerine daha duyarlı bir dinamik sistem cevabı elde edildiği görülecektir.

Simülasyon Sonuçları :

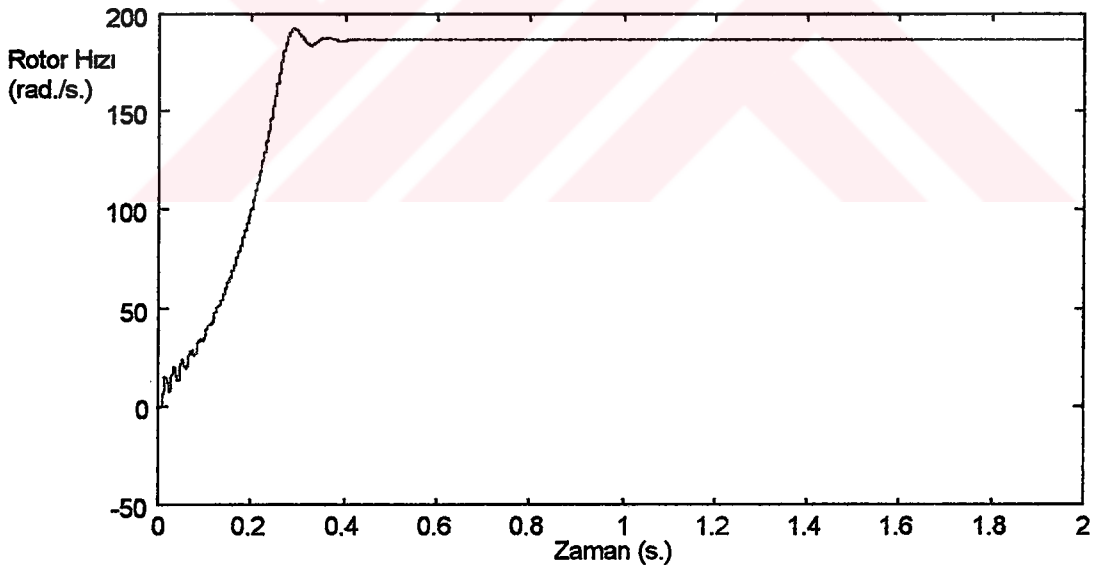


Fig 6.1.1 Hız - Zaman Eğrisi

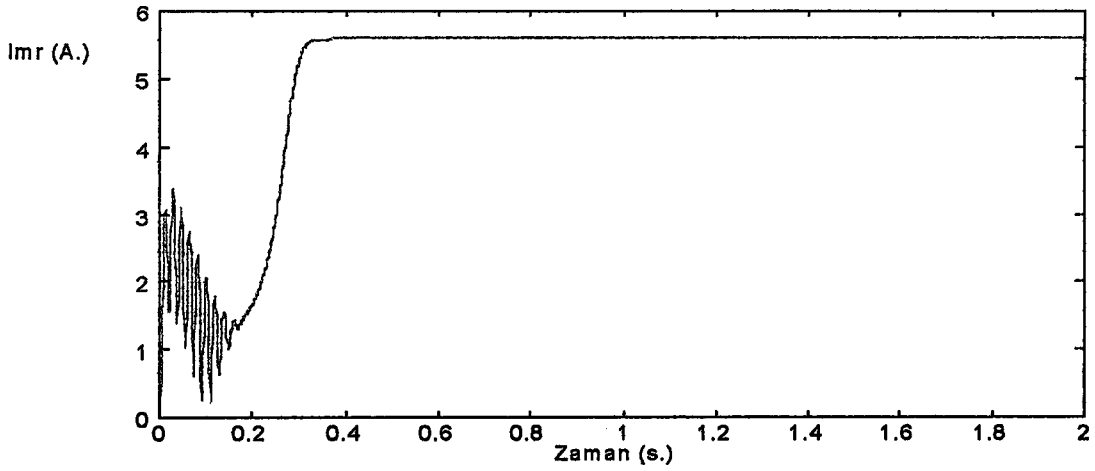


Fig 6.1.2 Mıknatıslanma akımının zamanla değişimi

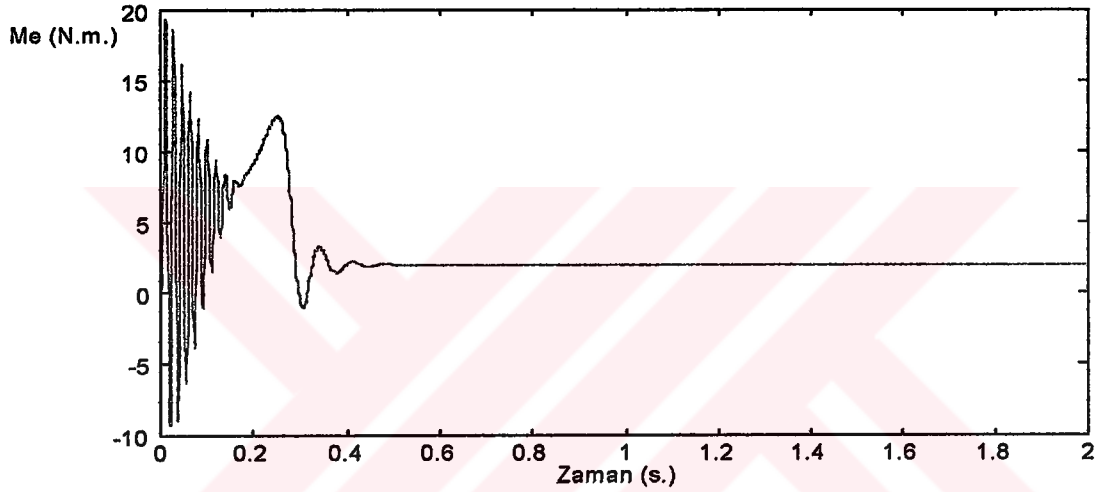


Fig 6.1.3 Rotor mıknatıslanma akımının zamanla değişimi

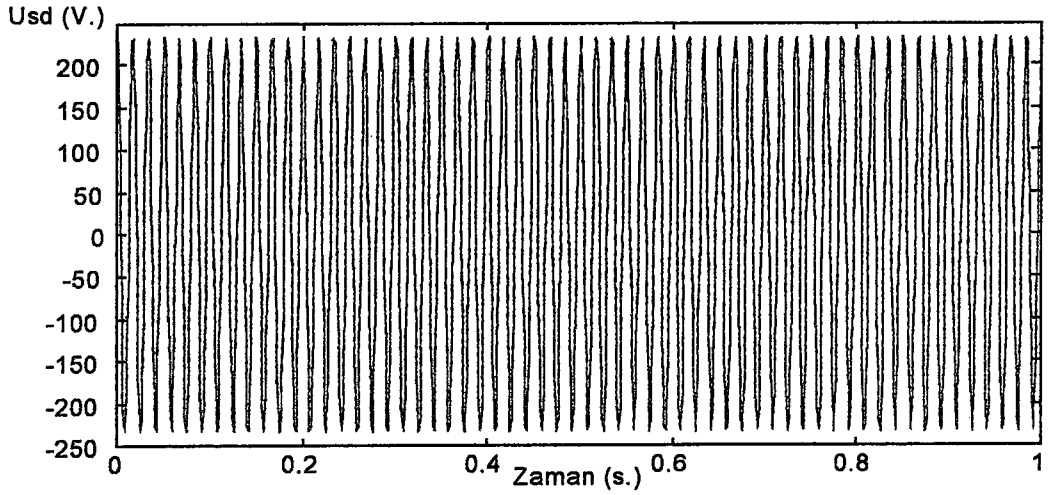


Fig 6.1.4 Stator gerilimi sD bileşeninin zamanla değişimi

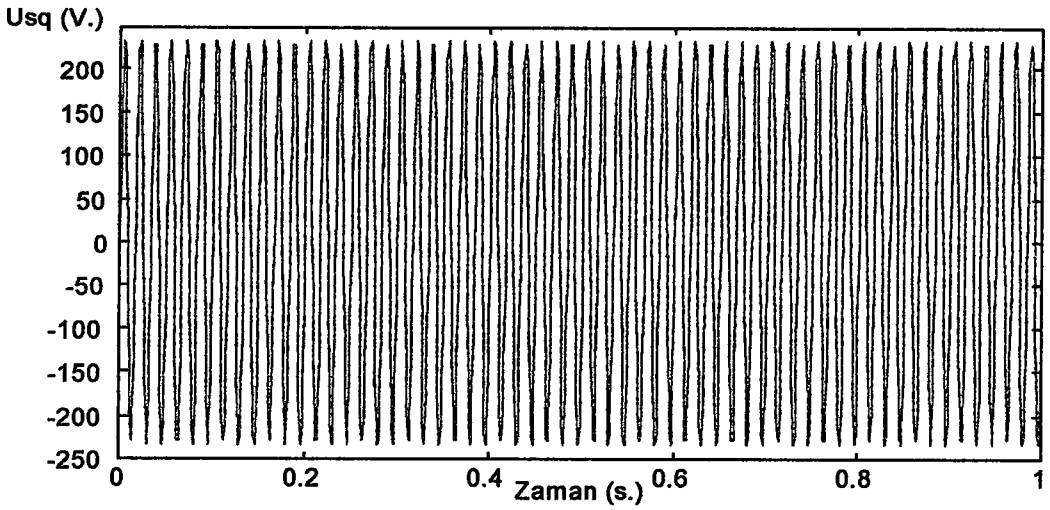


Fig 6.1.5 Stator gerilimi sQ bileşeninin zamanla değişimi

6.2 Rotor Akısı Yönlendirilmiş Vektör Kontrolüne İlişkin Simülasyon Sonuçları

Bu bölümde daha önce simülasyon sonuçları verilen makinaya vektör denetimi uygulanmıştır. Amaç ani değişimlere daha duyarlı ve kararlı bir dinamik sistem cevabı elde etmektedir. Ek 2 de verilen algoritmanın matlab derleyicisinde koşturulmasıyla aşağıdaki simülasyon sonuçlar elde edilmiştir. Fig 6.2.1 den de görüleceği üzere vektör kontrolü uygulanan motor, yaklaşık %25 lik bir aşımla çok daha kısa sürede stabil çalışma noktasını yakalamaktadır.

Simülasyon sonuçları :

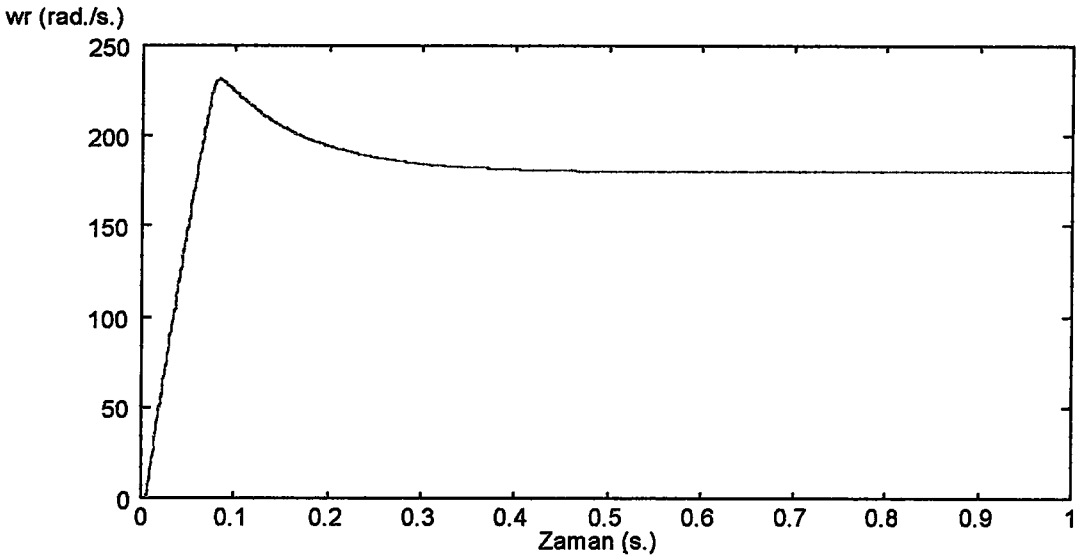


Fig 6.2.1 Vektör denetimi uygulanan asenkron makinada hız-zaman eğrisi

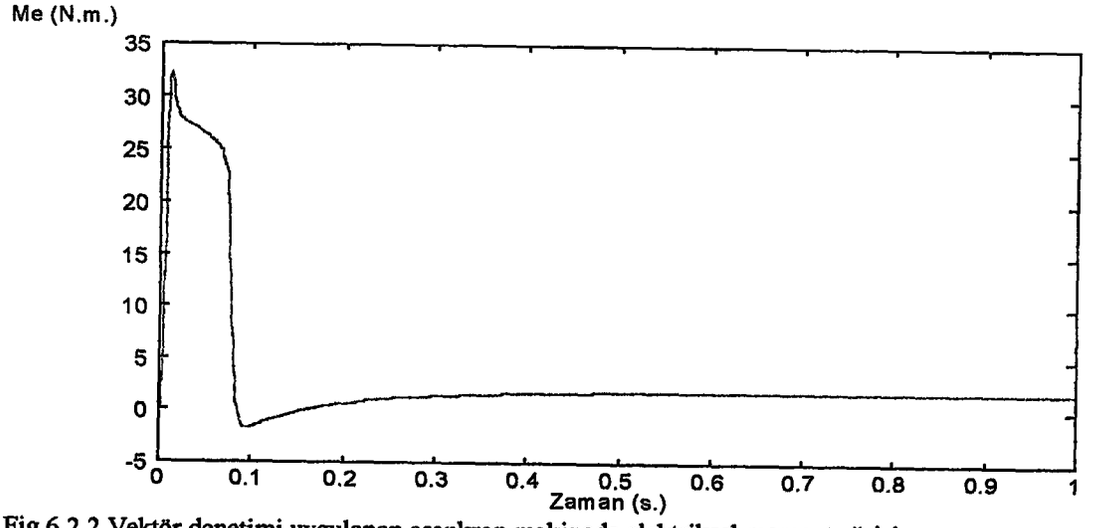


Fig 6.2.2 Vektör denetimi uygulanan asenkron makinada elektriksel moment eğrisi

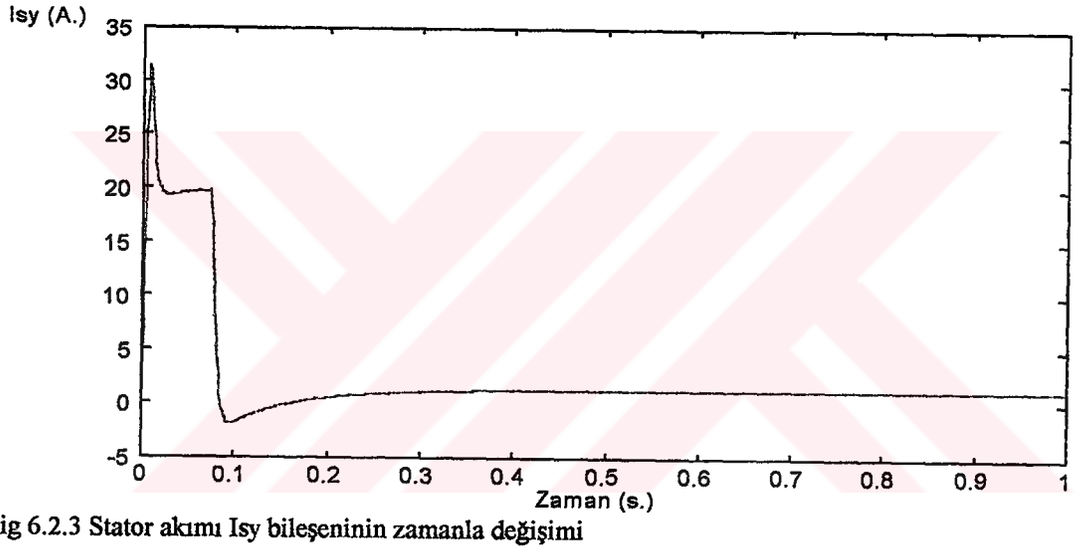


Fig 6.2.3 Stator akımı Isy bileşeninin zamanla değişimi

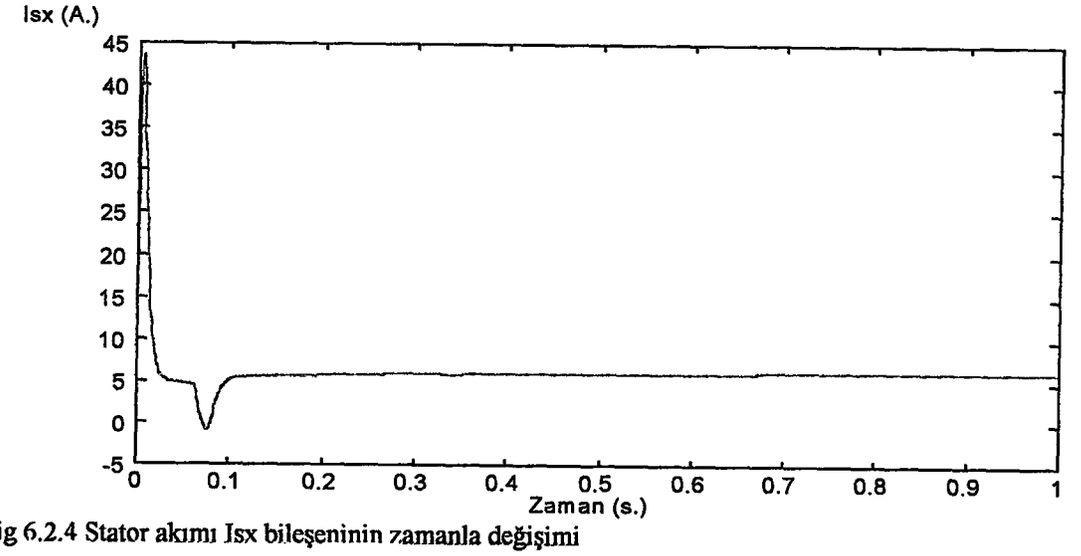


Fig 6.2.4 Stator akımı Isx bileşeninin zamanla değişimi

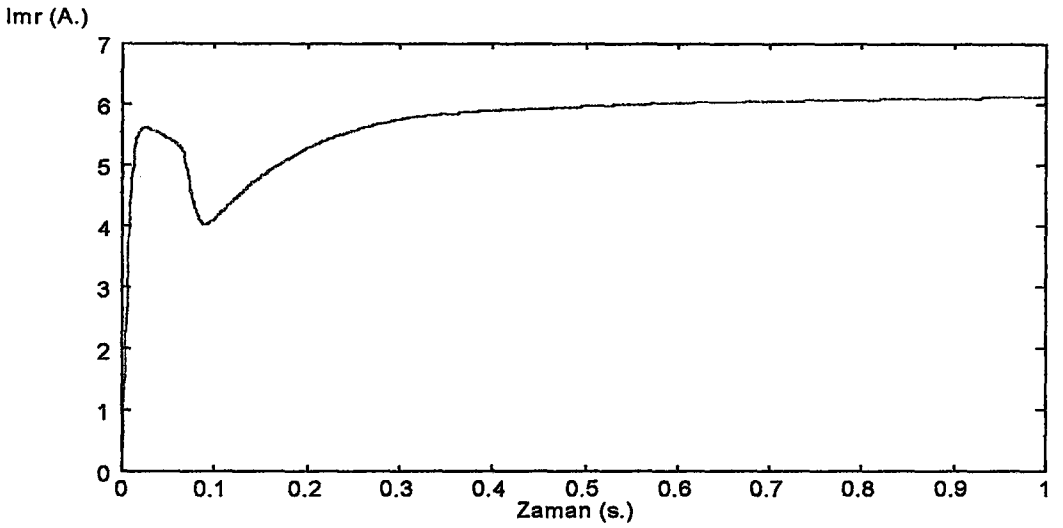


Fig 6.2.5 Rotor mıknatıslanma akımı genliğinin zamanla değişimi

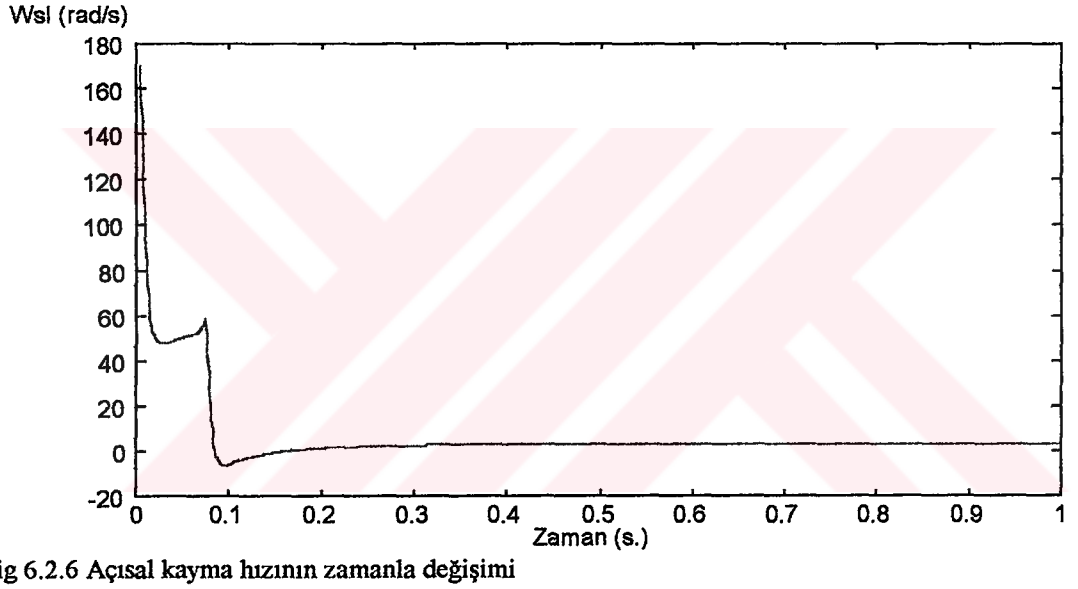


Fig 6.2.6 Açısal kayma hızının zamanla değişimi

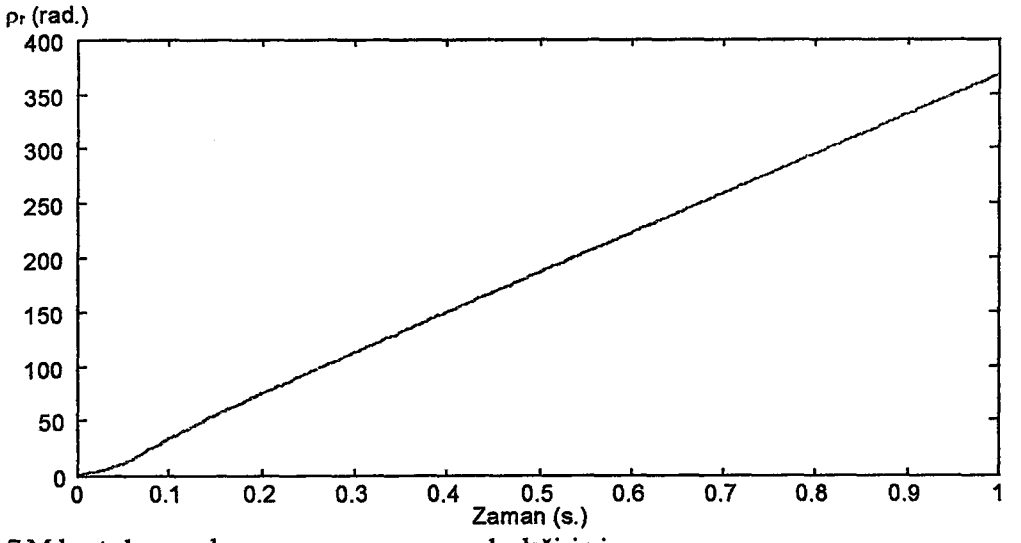


Fig 6.2.7 Mıknatıslanma akımı uzay açısının zamanla değişimi

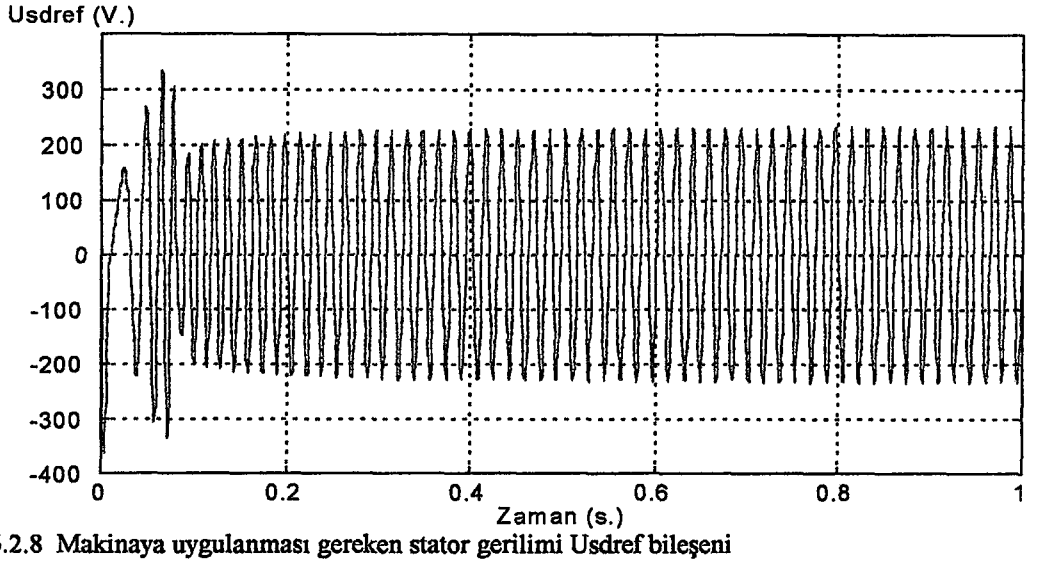


Fig 6.2.8 Makinaya uygulanması gereken stator gerilimi Usdref bileşeni

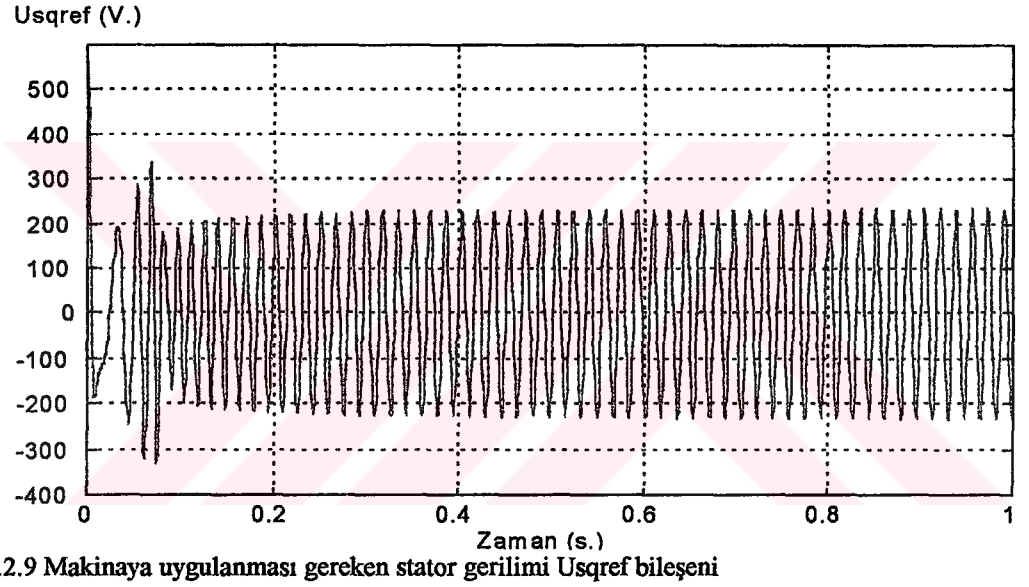


Fig 6.2.9 Makinaya uygulanması gereken stator gerilimi Usqref bileşeni

6.3 Rotor Mıknatıslanma Akımı Genliğini Estimate Eden Yapay Sinir Ağının Off-line Eğitimi ve Simülasyon Sonuçları

Bir önceki bölümde simülasyon sonuçları verilen vektör denetimine ilişkin şekil 3.2 deki blok şemada, akı modeli ve hız sensörü yerine neural tabanlı çok katmanlı algılayıcılar kullanılabilir. Bu amaçla dördüncü bölümde verilen yapay sinir ağı modellerinden, mıknatıslanma akımının genliğini tahmin eden ağın off-line eğitimi Ek 3 deki algoritma ile yapılmış ve yapay sinir ağının çıkışı ne kadarlık bir hata ile kestirebildiği Fig 6.3.1 de verilmiştir.

Çıkışı yeteri kadar iyi yaklaşıklıkla tahmin edebilmek için 50000 iterasyon (1000

yaklaşım) yapılmıştır. Simülasyon süresi 1.5 saattir. Bu süre sonunda hata 10^{-5} mertebesine kadar çekilmiştir. Arzu edilirse modellenen ağın, çıkışı 10^{-3} mertebesinde bir hata ile verebileceği yarım saatlik bir sürenin sonunda da kesilebilir.

Simülasyon sonuçları :

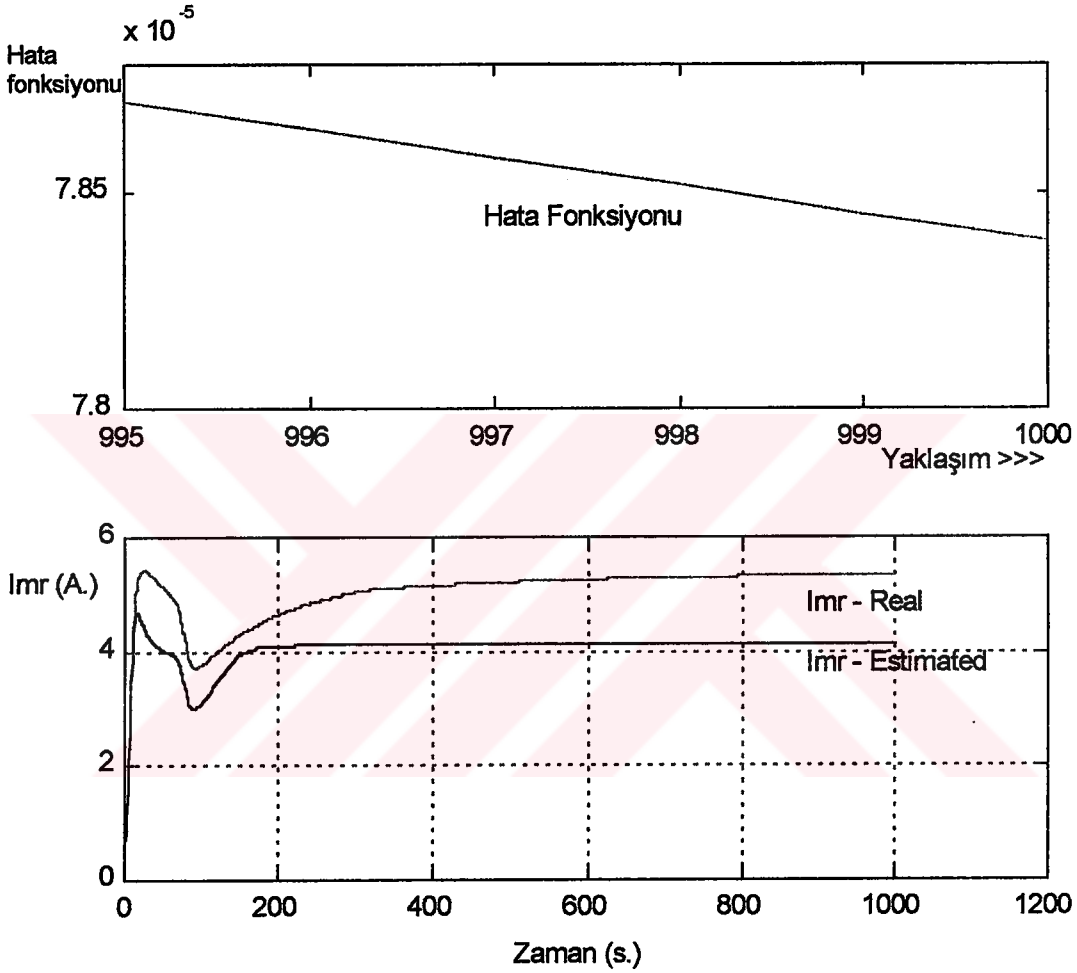


Fig 6.3.1 (a) Rotor mıknatıslanma akımı tahminindeki hata oranı

(b) Rotor mıknatıslanma akımının gerçek ve tahmin edilen eğrileri

Yukarıdaki tahmin edilen mıknatıslanma akımı eğrisi ve tahmindeki hata oranı göstermiştir ;ki yukarıda off-line eğitimi yapılan yapay sinir ağı bu performansı ile vektör denetimi prosesi içinde akı modeli yerine kullanılabilir. Elbette tek başına genlik tahmini yeterli değildir. Rotor mıknatıslanma akımının uzaysal konumunu veya açısal dönme hızını hesaplayan bir algılayıcıya daha ihtiyaç vardır ve buna ilişkin algoritma bir sonraki kısımda verilmiştir.

6.4 Rotor Mıknatıslanma Akımı Açısını Estimate Eden Yapay Sinir Ağının Off-line Eğitimi ve Simülasyon Sonuçları

Daha önce akı modeli algılayıcılarından genlik tahmini yapan yapay sinir ağının off-line eğitimine ilişkin simülasyon sonuçları verilmişti. Bu kısımda da, Bölüm 4 de modellendiği gibi mıknatıslanma akımı uzay fazörünün açısal hızını tahmin eden bir off-line eğitim algoritması ve simülasyon sonuçları verilmiştir. Aslında algoritma ile kayma açısal hızı tahmin edilerek, mıknatıslanma akımı uzay fazörünün açısal hızı dolaylı olarak hesaplanmaktadır.

Programın, çıkışları büyük yaklaşıklıklarla tahmin etmesi için 50000 iterasyon (1000 yaklaşım) yapılmıştır. Toplam simülasyon süresi 1.5 saattir. Bu süre sonunda hata $5*10^{-3}$ mertebesine kadar çekilmiştir. Arzu edilirse modellenen ağın, çıkışı 10^{-2} mertebesinde bir hata ile verebileceği yarım saatlik bir simülasyon süresi sonunda kesilebilir ve eğitimi yapılan ağ, vektör denetimi blok diyagramındaki akı modeli içine sokulabilir.

Simülasyon sonuçları :

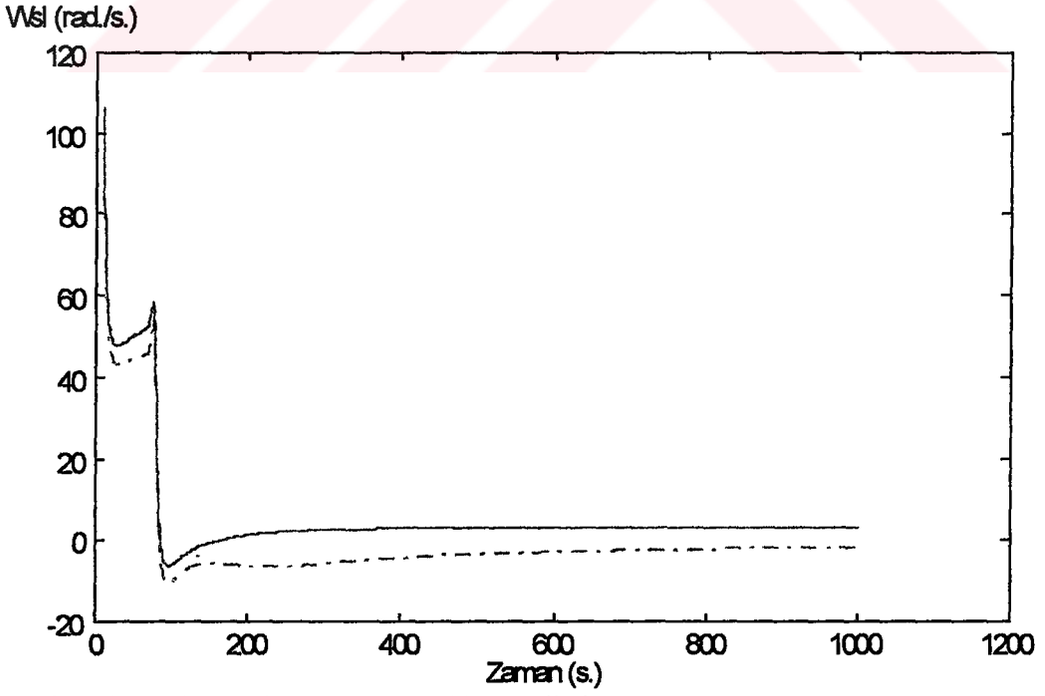


Fig 6.4.1 Gerçek ve tahmin edilen kayma açısal hızı eğrileri

Yukarıda gösterilen eğrilerden sürekli olanı kayma açısal hızının gerçek eğrisi, kesikli çizgilerle gösterileni ise tahmin edilen kayma açısal hızı eğrisidir. Burada tahmin edilen kayma açısal hızı, rotor hızına eklenerek rotor akısı yönlendirilmiş eksen takımındaki mıknatıslanma akımı uzay fazörünün açısal hızı hesaplatılacaktır.

6.5 Rotor Hızını Estimate Eden Yapay Sinir Ağının Off-line Eğitimi ve Simülasyon Sonuçları

Bölüm 3 de verildiği gibi, vektör denetimi blok şemasında en dışta hız geribeslemesi (kapalı çevrim) vardır. Bu kısımda rotor hızı geribeslemesi almak için kullanılan donanımların yerine rotor hızını tahmin eden ve 4. Bölümde modeli verilmiş olan neural tabanlı çok katmanlı algılayıcının off-line eğitimi yapılmış ve simülasyon sonuçları verilmiştir. İlgili algoritma Ek 5 de verildiği gibidir.

Başlangıçta 50000 iterasyon (1000 yaklaşım) seçilmiş ancak; simülasyon 125. yaklaşımda kesilmiştir. Simülasyon süresi 15 dakikadır. Bu süre sonunda hata 10^{-3} mertebesine kadar çekilmiştir. Simülasyon sonuçları, eğitilen ağın rotor hızını daha küçük bir aşımla tahmin edebildiğini ve kontrol bloğuna yerleştirildiğinde daha iyi bir dinamik cevap verdiğini göstermiştir.

Simülasyon sonuçları:

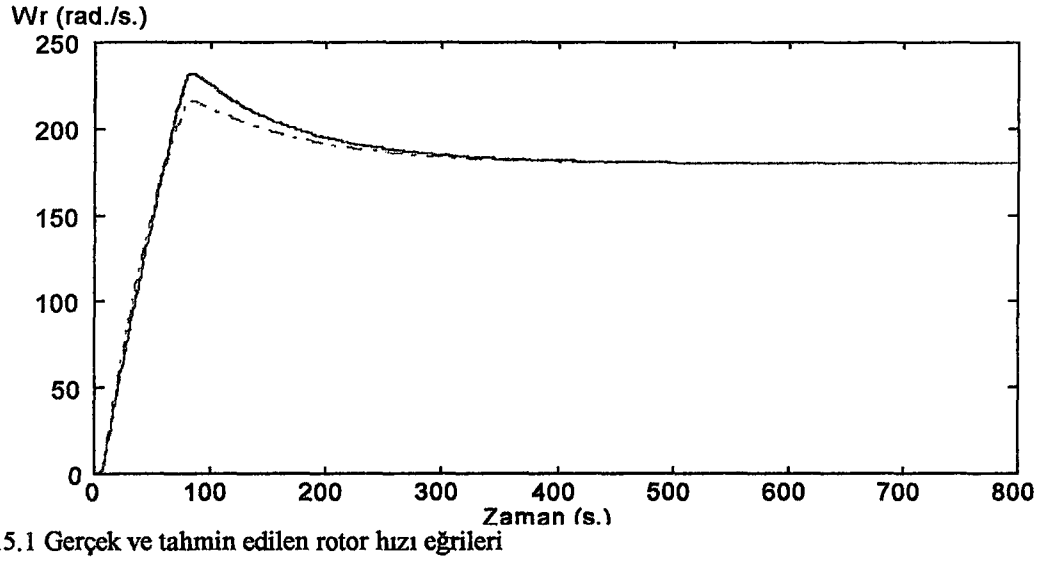


Fig 6.5.1 Gerçek ve tahmin edilen rotor hızı eğrileri

Not: Sürekli eğri gerçek rotor hızı, alttaki kesik çizgili eğri ise tahmin edilen rotor hızıdır.

6.6 Sincap Kafesli Asenkron Makinada Neural Tabanlı Akı Algılayıcısı ile Rotor Akısı Yönlendirilmiş Vektör Denetimi Simülasyon Sonuçları

Bu bölümde rotor akısı yönlendirilmiş vektör denetimine ilişkin simülasyon sonuçları verilmiştir. Burada akı modeli yerine 6.3 ve 6.4 de off-line eğitimleri yapılmış olan çok katmanlı neural algılayıcılar kullanılmıştır. Bu ağların kullanımıyla sistemin kararlılığının bozulmadığı simülasyon sonuçlarından görülmektedir. İlgili algoritma Ek 6 da verilmiştir. Simülasyon süresi yaklaşık yarım dakika civarındadır. Simülasyon programında yer alan neural tabanlı 2 adet algılayıcı, off-line eğitimi yapılmış yapay sinir ağlarıdır. Algoritma içinde on-line eğitim prosedürü yoktur.

Simülasyon sonuçları :

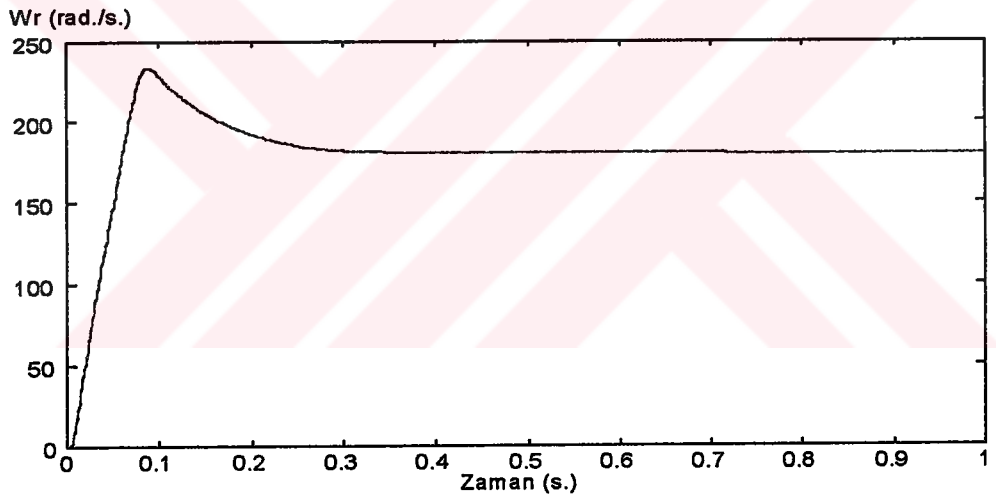


Fig 6.6.1 Hız - zaman eğrisi

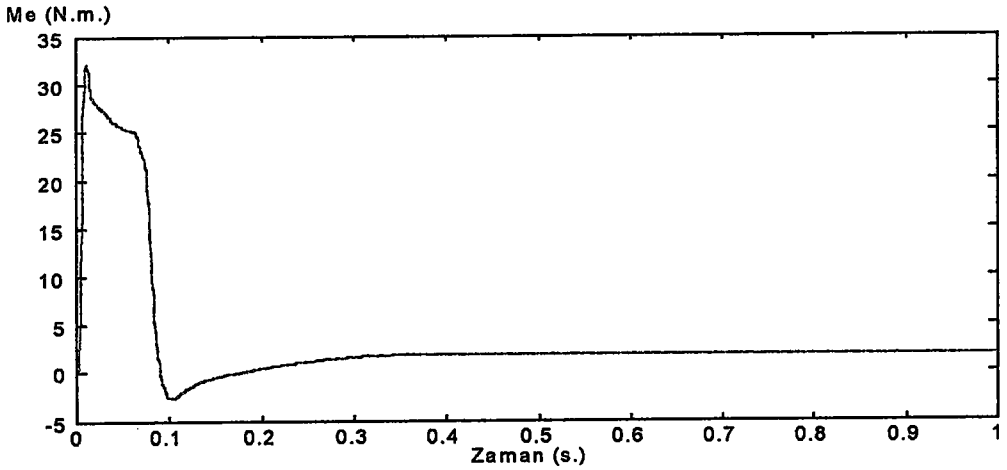


Fig 6.6.2 Elektriksel Döndürme Momenti Eğrisi

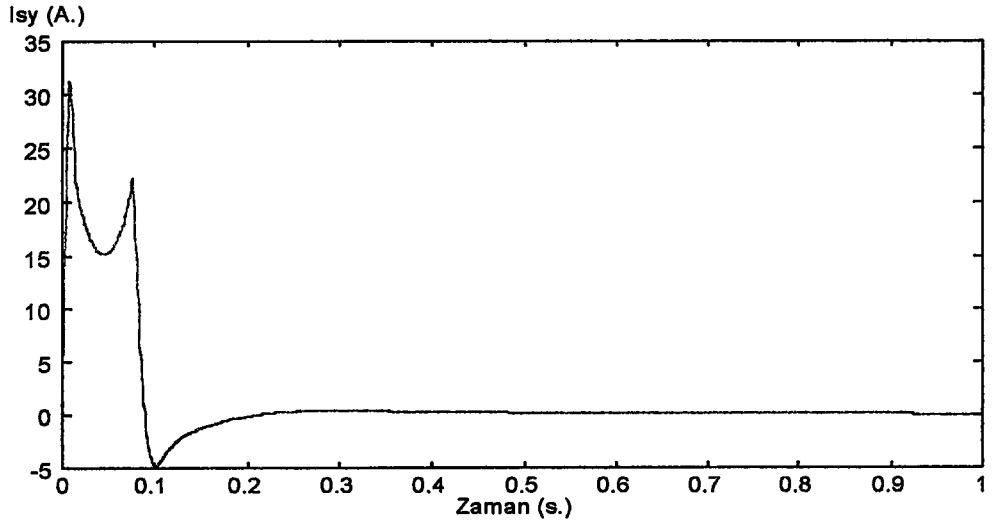


Fig 6.6.3 Stator akımı Isy bileşeni

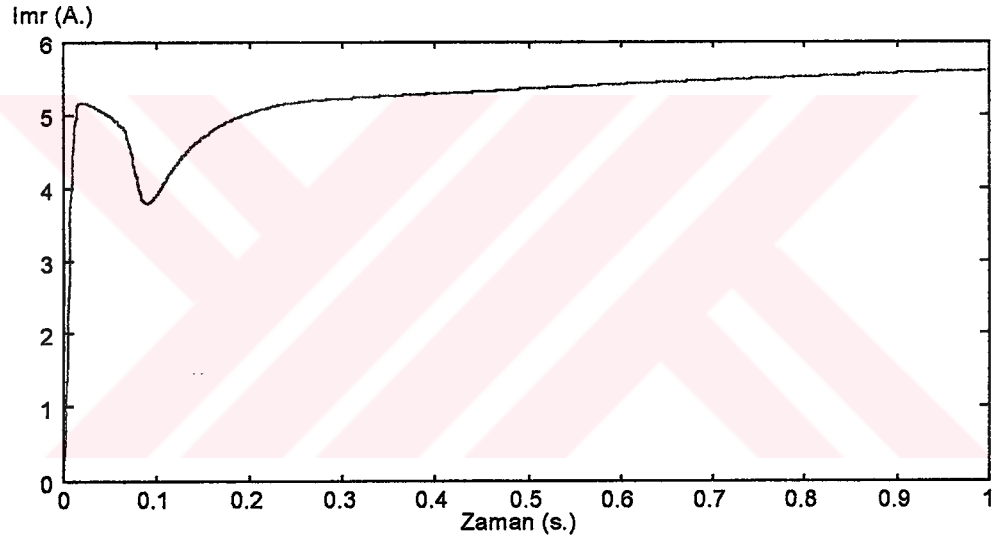


Fig 6.6.4 Tahmin edilen rotor mıknatıslanma akımı eğrisi

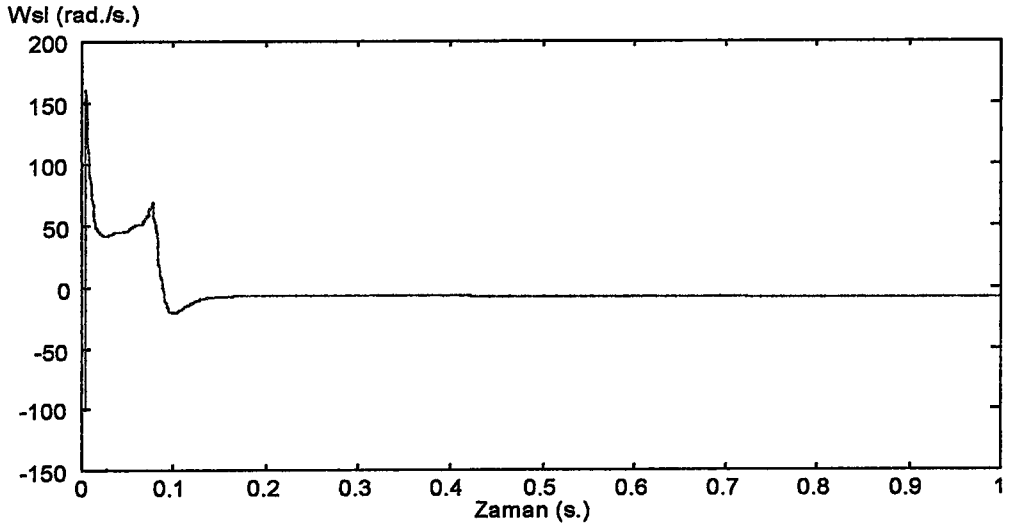


Fig 6.6.5 Tahmin edilen açısal kayma hızının değişimi

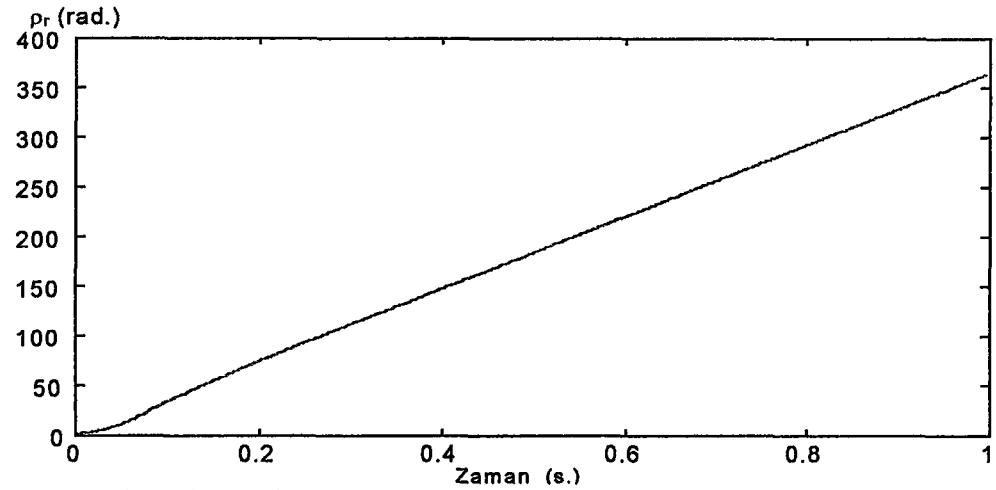


Fig 6.6.6 Rotor mıknatıslanma akımı uzay fazörünün konumu

6.7 Sincap Kafesli Asenkron Makinada Neural Tabanlı Hız Algılayıcısı ile Rotor Akısı Yönlendirilmiş Vektör Denetimi Simülasyon Sonuçları

Bu kısımda sadece neural tabanlı hız algılayıcısı içeren vektör denetimi algoritması ve simülasyon sonuçları verilmiştir. Kullanılan neural tabanlı çok katmanlı algılayıcı off-line eğitimi **6.5** de yapılmış olan yapay sinir ağıdır. İlgili algoritma Ek 7 de verilmiştir. Simülasyon süresi yaklaşık yarım dakikadır. Algoritma içinde on-line eğitim prosedürü yoktur. Simülasyon sonuçlarından da görüleceği üzere rotor hızı algılayıcısının kullanımı ile sistemin dinamik cevabında aşımın önemli ölçüde azalmış, hatta motorun hiçbir salınım yapmadan çok kısa zamanda stabil çalışma noktasına ulaştığı görülmüştür.

Simülasyon sonuçları :

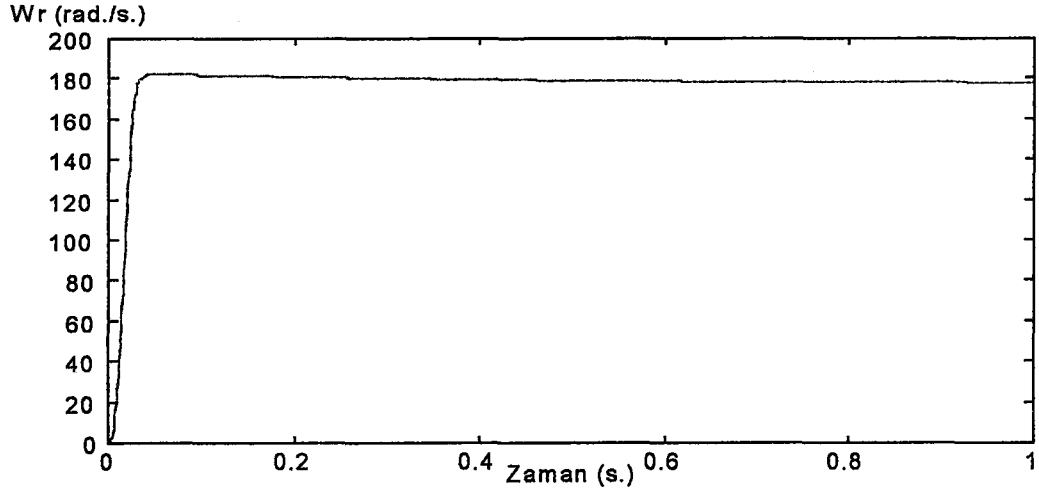


Fig 6.7.1 Hız - Zaman Eğrisi

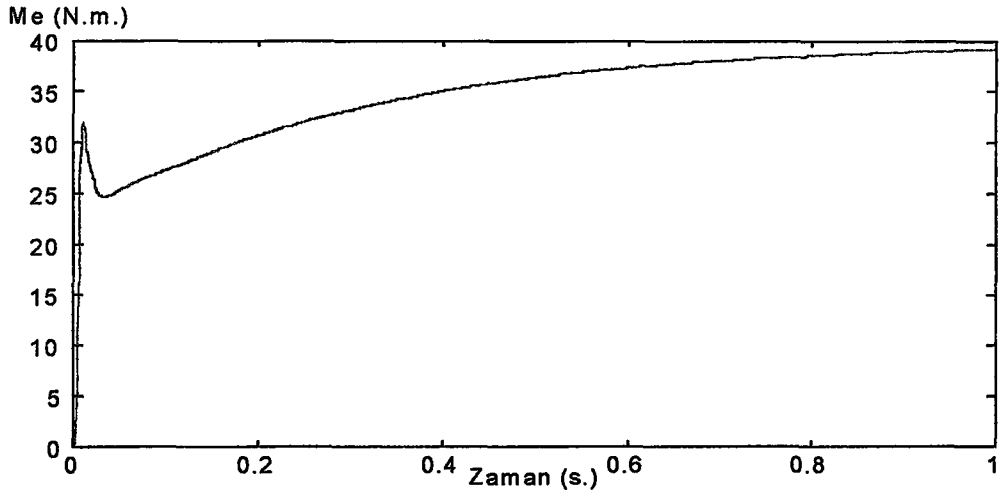


Fig 6.7.2 Elektriksel döndürme momenti eğrisi

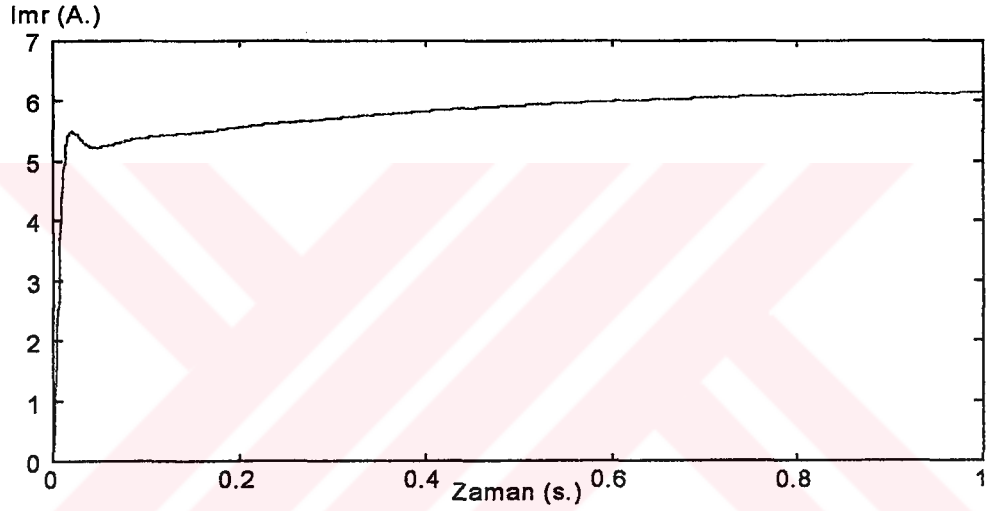


Fig 6.7.3 Rotor mıknatıslanma akımı eğrisi

6.8 Rotor Parametre Adaptasyonlu Neural Tabanlı Rotor Akısı Yönlendirilmiş Vektör Denetimi Simülasyon Sonuçları

Çalışmanın bu bölümünde rotor akısı yönlendirilmiş vektör kontrolünde yer alan akı modelinin rotor parametrelerinin değişimine duyarlı olması sebebiyle, adaptif kontrol yapılmıştır. Bu amaçla daha önce off-line eğitimi yapılmış olan neural tabanlı çok katmanlı algılayıcılar, değişen motor parametresinin sürekli olarak güncellendiği periyodik aralıklarda on-line eğitilerek daha iyi bir dinamik cevap elde edilmeye çalışılmıştır. Her online çevrim için iterasyon sayısı 20000 (400 yaklaşım) seçilmiştir. Toplam simülasyon süresi yaklaşık 3.5 saattir.

Aşağıdaki algoritma 5. Bölümde verilen şekil 5.1 deki blok diyagrama göre yazılmış ve ilgili değişkenlerin çıkış eğrileri simülasyon sonuçları bölümünde, ilgili algoritma ise Ek 8 de verilmiştir.

Simülasyon sonuçları :

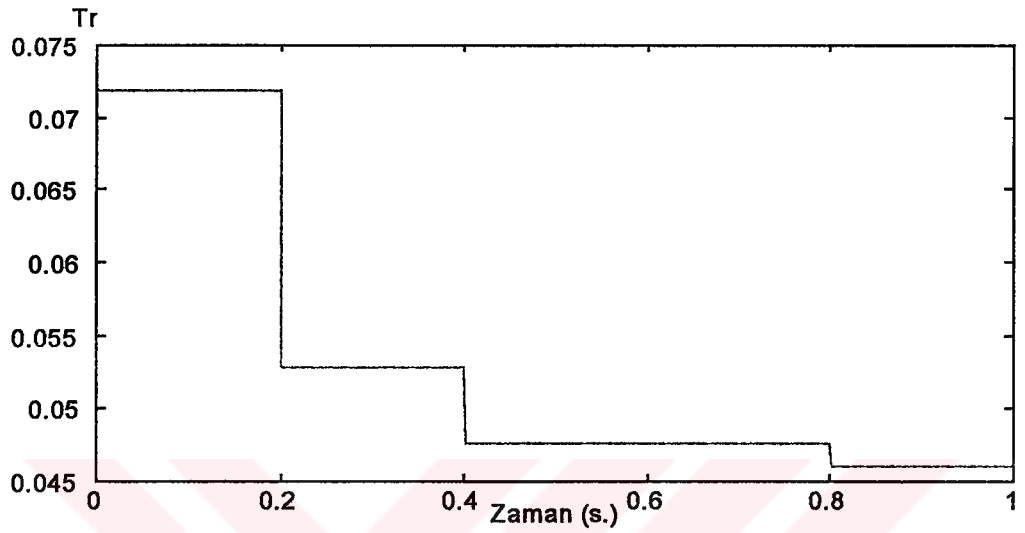


Fig 6.8.1 Rotor zaman sabitinin değişimi

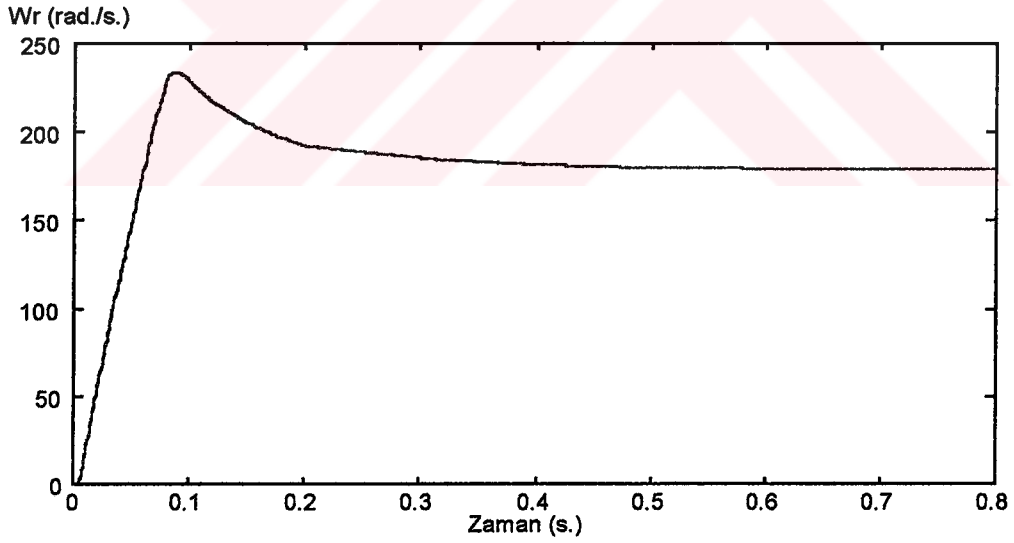


Fig 6.8.2 Hız-zaman eğrisi

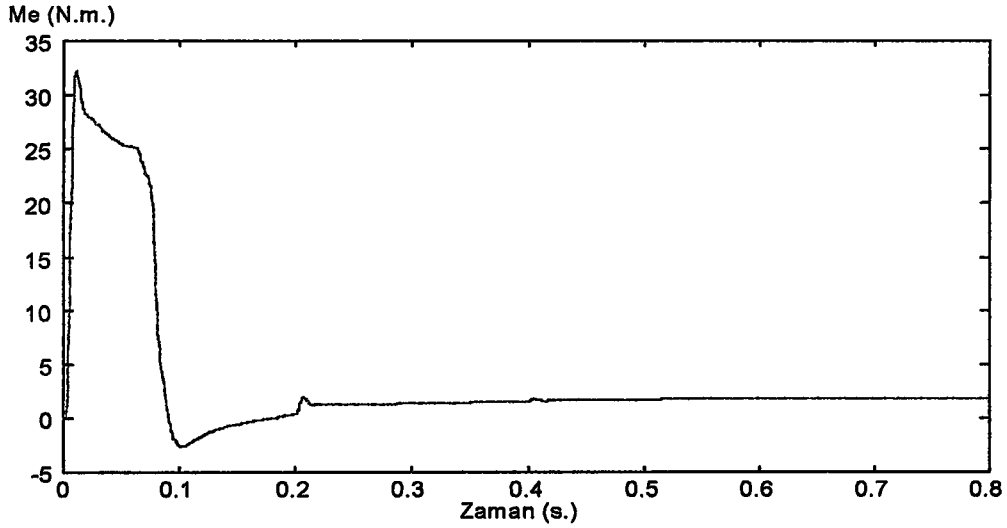


Fig 6.8.3 Elektriksel döndürme momentinin zamanla değişimi

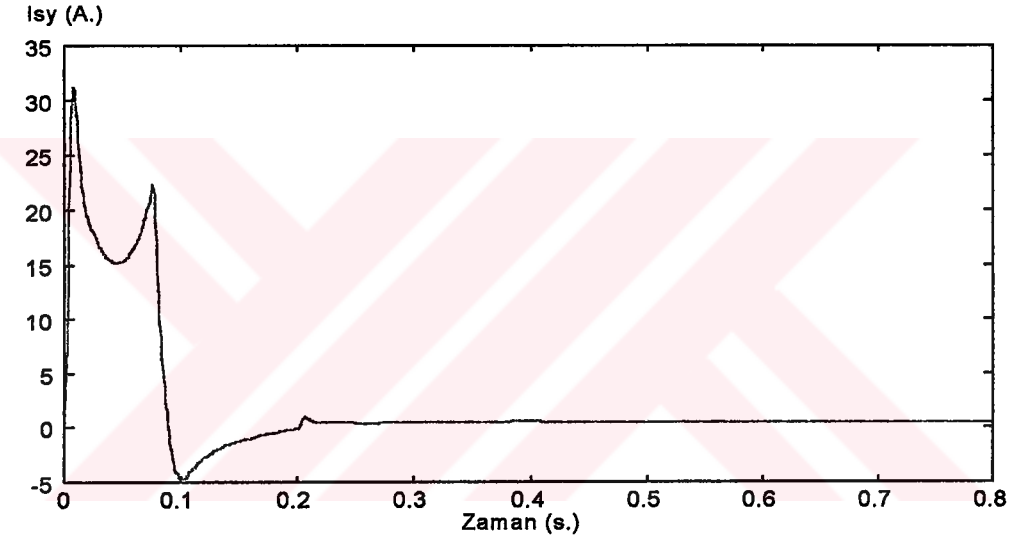


Fig 6.8.4 Stator akımı İsy bileşeninin zamanla değişimi

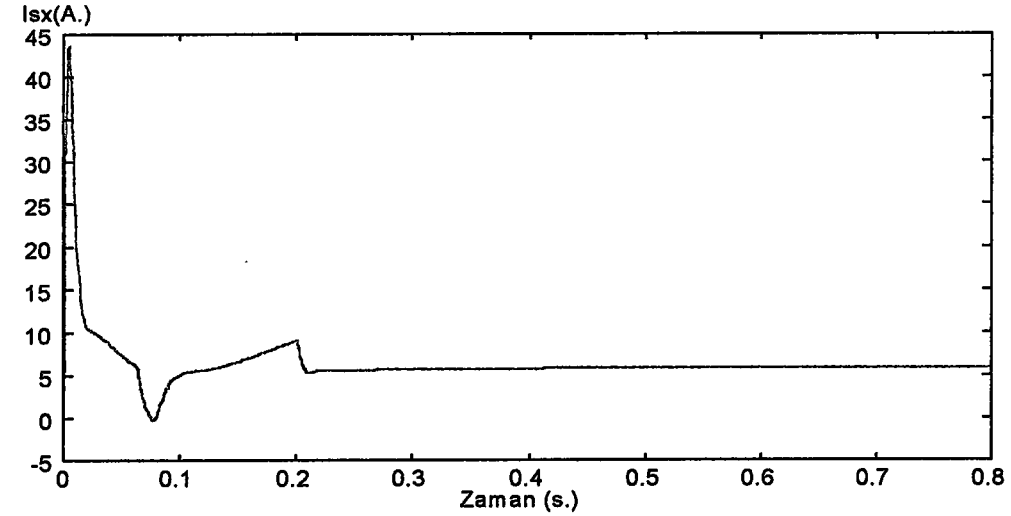


Fig 6.8.5 Stator akımı İsx bileşeninin zamanla değişimi

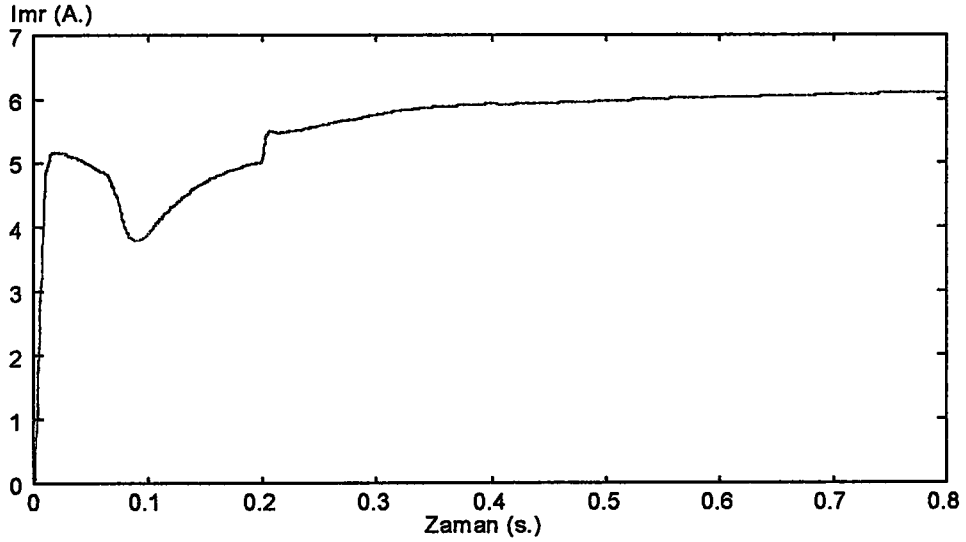


Fig 6.8.6 Rotor mıknatıslanma akımının zamanla değişimi

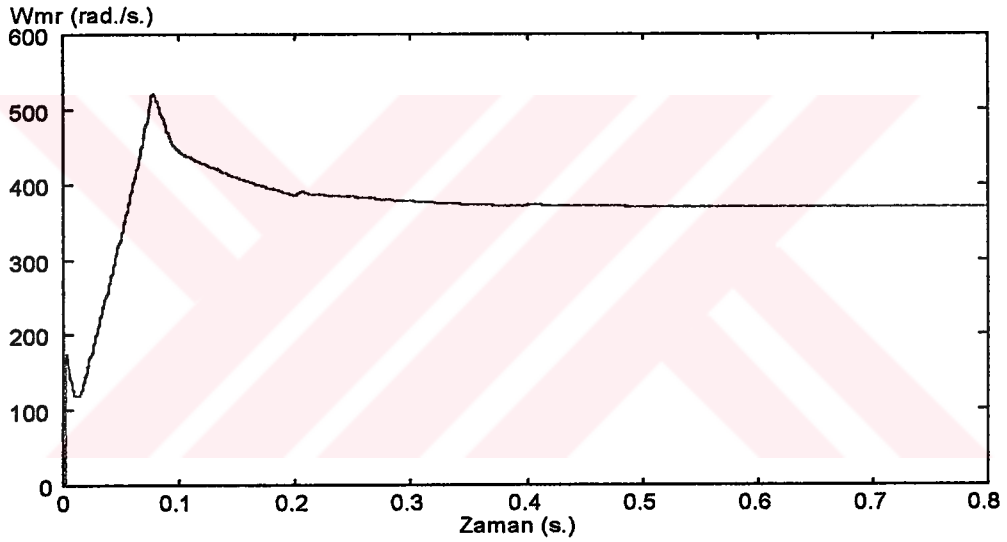


Fig 6.8.7 Rotor mıknatıslanma akımı uzay fazörünün açısal hızı

7. SONUÇLAR VE ÖNERİLER

Bu çalışmada sincap kafesli asenkron makine, rotor akısı yönlendirmeli vektörel kontrol uygulanarak doğru akım makinasına benzer bir biçimde kontrol edilmeye çalışılmıştır. Rotor akısı yönlendirilmiş eksen takımındaki akı ve moment indükleyen stator akımı bileşenlerinin birbirinden bağımsız olarak kontrol edilebilmesi amacıyla bir dekuplaj devresi ve sincap kafes nedeniyle ölçülemeyen rotor mıknatıslanma akımını gözlemleyen bir akı modeli kullanılmıştır. Kontrol yapısına ilişkin algoritma matlab editöründe derlenmiş ve simülasyon sonuçları 6. bölümde ayrıntılarıyla verilmiştir.

Sistem içinde yer alan akı modelinin çıkışları olan mıknatıslanma akımı uzay fazörünün modülü ve açısını tahmin eden yapay sinir ağlarının öngörülen referans model ile off-line eğitimi başarılı bir şekilde tamamlanmış ve yine hız geribeslemesi için kullanılan donanımların yerine neural tabanlı çok katmanlı bir algılayıcı modellenerek eğitimi yapılmıştır. Eğitim programları eklerde, simülasyon sonuçları ise 6. bölümde verilmiştir.

Eğitim prosedürünün tamamlanmasından sonra akı modeli ve hız algılayıcısı olarak modellenen neural yapılar, ağırlık konfigürasyonları sabit değerli ağırlık matrisleri olarak kontrol algoritmasının içine dahil edilmiştir. Yapılan simülasyonlar neticesinde akı modeli yerine kullanılan ağların büyük yaklaşıklıklarla çıkışları tahmin ettiği ve yine neural tabanlı hız algılayıcısının da sistemin dinamik cevabını büyük oranda iyileştirdiği: yükselme ve oturma zamanını hissedilir oranda düşürdüğü, aşımı hemen hemen sıfır yaptığı görülmüştür. Elde edilen sonuçlar, neural yapıların vektörel kontrolün uygulama alanları içerisinde dinamik sistemin stabilitesini bozmadan randımanlı bir biçimde kullanılabileceğini ortaya koymuştur.

Çalışmanın son bölümünde değişen rotor parametresi (R_r) uyumlu bir kontrol algoritması yapılandırılmaya çalışılmıştır. Simülasyon sonuçları göstermiştir ki rotor zaman sabiti önemli oranda değişmektedir. Önceki bölümlerde de ifade edildiği gibi rotor akısı yönlendirilmiş kontrol yapılırken akı modeli, rotor zaman sabitinin değişiminden doğrudan etkilenir. Rotor parametresindeki bu değişimi gözardı etmeden yapılacak bir kontrol dinamik sistemin davranışını iyileştirecektir. Bu amaçla yazılan

bir algoritma ile rotor zaman sabiti 0.2 saniyelik periyodik aralıklarla güncellenmiş ve belirlenen yeni rotor parametresine bağlı olarak, kullanılan neural ağların on-line eğitimi yapılmıştır. Elde edilen simülasyon sonuçları, bu şekilde yapılacak bir kontrolün verimli olabileceğini ortaya koymuştur. Ancak gözardı edilmemesi gereken önemli bir sorun, pratikte bu tip bir kontrolün gerçeklenebilirliğini zorlaştırmaktadır. Bu sorun adaptif kontrol algoritmasındaki neural tabanlı algılayıcıların on-line eğitiminin hayli uzun sürmesidir. Örneğin her örnekleme zamanı sonunda güncellenen yeni parametrelere göre on-line eğitimi yapılan neural tabanlı algılayıcıların eğitimi herbir online çevrim için yaklaşık 50 dakika kadardır. Algoritma 133 MHz (7.5 nano saniye) işlemcili bir makinada yürütülmüştür. Bugün kullanılan daha yüksek işlem gücüne ve daha gelişmiş donanım mimarisine sahip makinalarla veya sadece neural ağların eğitimine adanmış DSP'ler ile bu süre %95 oranında düşürülebilir. Ancak bu da yeterli değildir. Çünkü 0.2 saniye olarak seçilen örnekleme zamanından daha kısa bir sürede kabul edilebilir bir gecikme ile online eğitim prosedürü tamamlanabilirse ancak o zaman kullanışlı bir kontrol methodu olarak kullanılabilir.

Elbette bu proses gerçeklenirken gözardı edilmemesi gereken önemli bir nokta da şudur. Kontrol bloğu içerisindeki PI kontrolörlerin katsayıları başlangıçta motor parametrelerinin sabit olduğu varsayımı ile belirlenmiştir. Ancak rotor zaman sabitinin değişimi gözlemlenerek adaptif kontrol yapılmaya çalışıldığında, bu değişimden mıknatıslanma akımı modülünün, gerilimlerin ve stator akımı bileşenlerinin etkilendiği görülür. Bu değişkenlerin, kontrolörlerin girişine verilen hata fonksiyonlarının birer bileşeni olduğu gözönüne alınacak olursa, makine parametrelerinin değişken olması dolayısıyla K_p ve K_i katsayılarını sürekli güncelleyen bir sentez algoritması ile, daha iyi bir dinamik cevap elde edilecektir. Bu çalışmada kontrolörlerin katsayılarını parametre değişimine göre sürekli güncelleyen bir sentez algoritması olmadığı için elde edilen çıkışların, parametre adaptasyonu olmadan yapılan simülasyondaki sonuçlara oldukça yakın ama belirgin biçimde daha iyi olmadığı görülmüştür. Kuşkusuz makina parametrelerinin değişimine bağlı olarak kontrolör katsayılarının sentez ilkeleri çerçevesinde sürekli güncellenmesiyle, tam sağlıklı bir kontrol yapılmış olacaktır. Bu arada alan zayıflatma için modellenen fonsiyon üretecinin de değişen mıknatıslanma akımı sınırları ile güncellenmesi gerekmektedir. Buraya kadar bahsedilen eğitim prosedürlerinin uzun sürmesi problemi ilerleyen zamanda çözülebilir ve yine bu model

içindeki dijital kontrolörler periyodik aralıklarla güncellenirse, daha iyi bir dinamik cevap elde edileceği aşikardır.

Yapay sinir ağları, lineer olmayan algılayıcıların modellenmesi ve elde edilen modellerin oldukça etkin yaklaşımlar yapabilmesi dolayısıyla önü açık bir tasarım yöntemidir. Daha hızlı işlem yapabilme potansiyeline sahip gelişmiş donanımların kullanımıyla birlikte, ilerisi için ümit vericidir.

KAYNAKLAR

[1] **VAS, P.** Vector Control of AC Machines, Clarendon Press Oxford, 1990

[2] **GÜZELİŞ, Cüneyt** Yapay Sinir Ağları Ders Notları 1998

[3] **SARIOĞLU, M.K.** Dynamics of Electrical Machines Ders Notları

[4] **MathWorks Inc.** Matlab Supplement to Fuzzy and Neural Approachs in Engineering

[5] **THEOCHARIS J., PETRIDIS V.** Neural Network Observer for Induction Motor Control, IEEE T-CS, 0272-170894/94, April, 1996

[6] **MATSUO T. LIPO T.A** A Rotor Parameter Identification Scheme for Vector-Controlled Induction Motor Drives, IEEE T-IA, Vol IA-21, NO.4, May/June, 1985

[7] **NARENDRA K.** Identification and Control of Dynamical Systems Using Neural Networks, IEEE T-NN, Vol 1, No.1, March 1990

[8] **Wishart M.T. Harley R. G.** Identification and Control of Induction Machines Using Artificial Neural Networks, IEEE T-IA, Vol 31, No.3, May/June, 1995

[9] **Simoes M. G. BOSE B. K.** Neural Network Based Estimation Of Feedback Signals for A Vector Controlled Induction Machine, IEEE T-IA, Vol 31, No.3, May/June, 1995

[10] **YANG H. T., HUANG K. Y., HUANG C. L.** An Artificial Neural Network Based Identification and Control Approach for the Field-Oriented Induction Motor, Electrical Power System Resarch 30 (1994) 35-45

[11] **CHTOUROU M., DEBEL N., KAMOUN M. B. A.** Control of A Loaded Induction Machine Using A Feedforward Neural Network, International Journal of System Science, Vol 27, No.12, Pages 1287-1295, 1996

[12] **JOHAN A.K. SUYKENS** Artificial Neural Networks for Modelling and Control of Non-lineer Systems, Boston: Kluwer Academic Publishers, c1996

[13] **GAMAK**, 3 FAZLI TAM KAPALI (IP 55) STANDART ASENKRON MOTORLAR, Broşür No. 022T/94

EK 1.

```
% Sincap kafesli asenkron makinanın D-Q simülasyonu
clear
t0=clock;
p=2;
J=8e-3;
m=98e-3;
ls=109e-3;
lr=115e-3;
rs=1.5;
rr=1.6;
s=1-m^2/(ls*lr);
lsg=s*ls;
lrg=s*lr;
Tr=lr/rr;
Tsg=lsg/rs;
Trg=lrg/rr;
ml=2; Ml(1)=ml;
time=2;
dt=0.001;
v=165;
freq=60;
wr=0;
i=zeros(4,1);
A=[-(1/Tsg+(1-s)/Trg) 0 (1-s)/Trg 0;0 -(1/Tsg+(1-s)/Trg) 0 (1-s)/Trg
  ;1/Tr 0 -1/Tr 0;0 1/Tr 0 -1/Tr];
B=[0 0 0 (1-s)/s;0 0 -(1-s)/s 0;0 0 0 -1;0 0 1 0];
C=[1/lsg 0 0 0;0 1/lsg 0 0;0 0 0 0;0 0 0 0];
for t=0:dt:time-dt
me=3/2*p*m^2/lr*(i(3)*i(2)-i(4)*i(1));
imramin=sqrt(i(3)^2+i(4)^2);
if i(3)==0 & i(4)==0
imranin=NaN;
else
imranin=atan2(i(4),i(3));
end
urd=0; urq=0;
usd=sqrt(2)*v*cos(120*pi*t);
usq=sqrt(2)*v*sin(120*pi*t);
U=[usd;usq;urd;urq];
ki1=(A*i+B*wr*p*i+C*U)*dt;
ki2=(A*(i+0.5*ki1)+B*wr*p*(i+0.5*ki1)+C*U)*dt;
```

```
ki3=(A*(i+0.5*ki2)+B*wr*p*(i+0.5*ki2)+C*U)*dt;
ki4=(A*(i+ki3)+B*wr*p*(i+ki3)+C*U)*dt;
i=i+1/6*(ki1+2*ki2+2*ki3+ki4);
imram=sqrt(i(3)^2+i(4)^2);
imran=atan2(i(4),i(3));
wr=wr+dt/J*(me-ml);
Usd((t+dt)/dt)=usd;
Usq((t+dt)/dt)=usq;
I(:,(t+dt)/dt)=i;
Imram((t+dt)/dt)=imram;
Imran((t+dt)/dt)=imran;
Me((t+dt)/dt)=me;
Wr((t+dt)/dt)=wr;
end
mes=3/2*(i(3)*i(2)-i(4)*i(1))*p*m^2/lr;
ame=[Me mes];
aimram=[imramin Imram];
aimran=[imranin Imran];
tt=0:dt:time;
ai(:,1)=zeros(4,1);
ai(:,2:(time+dt)/dt)=I;
awr=[0 Wr];
ausd=[0 Usd];
ausq=[0 Usq];
plot(awr,ate);grid;
plot(tt,awr);grid;
plot(tt,aimram);grid;
plot(tt,aimran);
end
```

EK 2.

```
% Sincap kafesli asenkron makinada rotor akısı yönlendirilmiş vektör denetimi
clear
t0=clock;
time=1.0;
dt=0.001;
lamda=0.01;
p=2;
J=8e-3;
rs=1.5;
rr=1.6;
ls=109e-3;
lr=115e-3;
m=98e-3;
s=1-m^2/(ls*lr);
lsg=s*ls;
Tsg=lsg/rs;
lrg=s*lr;
Tr=lr/rr;
Trg=lrg/rr;
wr=0;
i=zeros(4,1);
U=[180*ones(2,1);zeros(2,1)];
wsl=0;
imran=0;
imram=0;
wref=180;
cakı=0;
cisx=0;
chız=0;
cmom=0;
cisy=0;
kp1=13.45;
kı1=0.4;
kp2=7.7;
kı2=0.8;
kp3=2.7;
kı3=0.1;
kp4=2.7;
kı4=0.6;
kp5=1.9;
kı5=0.19;
```

```
A=[-(1/Tsg+(1-s)/Trg) 0 (1-s)/Trg 0;0 -(1/Tsg+(1-s)/Trg) 0 (1-s)/Trg;1/Tr 0 -1/Tr
0;0 1/Tr 0 -1/Tr];
B=[0 0 0 (1-s)/s;0 0 -(1-s)/s 0;0 0 0 -1;0 0 1 0];
C=[1/lsg 0 0 0;0 1/lsg 0 0;0 0 0 0;0 0 0 0];
for t=0:dt:time-dt
me=3/2*p*(m^2)/lr*(i(3)*i(2)-i(4)*i(1));
if t~=0
U=[usdref;usqref;0;0];
end
ki1=(A*i+B*wr*p*i+C*U)*dt;
ki2=(A*(i+0.5*ki1)+B*wr*p*(i+0.5*ki1)+C*U)*dt;
ki3=(A*(i+0.5*ki2)+B*wr*p*(i+0.5*ki2)+C*U)*dt;
ki4=(A*(i+ki3)+B*wr*p*(i+ki3)+C*U)*dt;
i=i+1/6*(ki1+2*ki2+2*ki3+ki4);
isx=i(1)*cos(imran)+i(2)*sin(imran);
isy=i(2)*cos(imran)-i(1)*sin(imran);
k1g=(-imram/Tr+isx/Tr)*dt;
k2g=(-(imram+0.5*k1g)/Tr+isx/Tr)*dt;
k3g=(-(imram+0.5*k2g)/Tr+isx/Tr)*dt;
k4g=(-(imram+k3g)/Tr+isx/Tr)*dt;
imram=imram+1/6*(k1g+2*k2g+2*k3g+k4g);
if imram==NaN
imram=0;
end
wr=(1-lamda*dt/J)*wr+dt/J*me;
ml=lamda*wr;
wsl=isy/(Tr*imram);
Wsl((t+dt)/dt)=wsl;
imran=imran+dt*(wr*p+isy/(Tr*imram));
if imran==NaN
imran=0;
end
wmr=(i(3)*((i(2)-i(4))/Tr+wr*p*i(3))-i(4)*((i(1)-i(3))/Tr-
wr*p*i(4)))/(i(3)^2*(1+(i(4)/i(3))^2));
udx=-wmr*lsg*isy;
udy=wmr*lsg*isx+(ls-lsg)*wmr*imram;
if abs(wr)>184 & abs(wr)<265
imref=6.1-abs(wr)/30.16+6.1;
elseif abs(wr)<=184
imref=6.1;
else
imref=0;
end
cakı=cakı+dt*kp1/kı1*(imref-imram);
isxref=kp1*(imref-imram)+cakı;
cisx=cisx+dt*kp2/kı2*(isxref-isx);
usxes=kp2*(isxref-isx)+cisx;
usxref=usxes+udx;
chız=chız+dt*kp3/kı3*(wref-wr);
```

```
meref=kp3*(wref-wr)+chız;
if meref>40
meref=40;
end
if meref<-40
meref=-40;
end
cmom=cmom+dt*kp4/kı4*(meref-me);
isyref=kp4*(meref-me)+cmom;
cisy=cisy+dt*kp5/kı5*(isyref-isy);
usyes=kp5*(isyref-isy)+cisy;
usyref=udy+usyes;
usdref=usxref*cos(imran)-usyref*sin(imran);
usqref=usyref*cos(imran)+usxref*sin(imran);
if usdref>800
usdref=800;
elseif usdref<-800
usdref=-800;
end
if usqref>800
usqref=800;
elseif usqref<-800
usqref=-800;
end
Me((t+dt)/dt)=me;
Ml((t+dt)/dt)=ml;
I(:,(t+dt)/dt)=i;
Wr((t+dt)/dt)=wr;
Isx((t+dt)/dt)=isx;
Isy((t+dt)/dt)=isy;
Imram((t+dt)/dt)=imram;
Imran((t+dt)/dt)=imran;
Us(:,(t+dt)/dt)=[usdref;usqref];
Chız((t+dt)/dt)=chız;
Cakı((t+dt)/dt)=cakı;
Cisx((t+dt)/dt)=cisx;
Cmom((t+dt)/dt)=cmom;
Cisy((t+dt)/dt)=cisy;
Wmr((t+dt)/dt)=wmr;
Imref((t+dt)/dt)=imref;
Isxref((t+dt)/dt)=isxref;
Meref((t+dt)/dt)=meref;
Isyref((t+dt)/dt)=isyref;
Usxes((t+dt)/dt)=usxes;
Usyes((t+dt)/dt)=usyes;
Usxref((t+dt)/dt)=usxref;
Usyref((t+dt)/dt)=usyref;
end
mes=3/2*p*(m^2)/lr*(i(3)*i(2)-i(4)*i(1));
```

```
ame=[Me mes];
aimram=[0 Imram];
aimran=[0 Imran];
tt=0:dt:time;
ai(:,1)=[0;0;0;0];
ai(:,2:(time+dt)/dt)=I;
awr=[0 Wr];
aml=[0 Ml];
aisx=[0 Isx];
aisy=[0 Isy];
ausxes=[0 Usxes];
ausyes=[0 Usyes];
ausxref=[0 Usxref];
ausyref=[0 Usyref];
aus(:,1)=[0;0];
aus(:,2:(time+dt)/dt)=Us;
acakı=[0 Cakı];
acisx=[0 Cisx];
acmom=[0 Cmom];
acisy=[0 Cisy];
achız=[0 Chız];
awsl=[0 Wsl];
awmr=[0 Wmr];
aimref=[6.1 Imref];
aisxref=[0 Isxref];
aisyref=[0 Isyref];
ameref=[0 Meref];
sure=etime(clock,t0);
plot(tt,ame);grid
plot(tt,awr);grid;
compass(ai(1,:),ai(2,:))
plot(tt,aimram);grid;
plot(tt,aimran);grid;
plot(tt,aus(1,:));
plot(tt,wmr);grid;
plot(tt,awsl);grid;
end
```

EK 3.

```
% Mıknatıslanma akımı genliğini estimate eden yapay sinir ağının eğitim programı
clear
nntwarn off
dt=0.001;
Nimr=8;
Nisx=50;
Tr=0.0719;
b=(Tr-dt)/Tr;
c=(dt*Nisx)/(Tr*Nimr);
load vektör_denetimi
aisxn=aisx/Nisx;
y=1;
mc=0.95;lr=0.02;
Layer2=10;Layer3=8;
[Weigth1,Bias1]=rands(Layer2,2);
[Weigth2,Bias2]=rands(Layer3,Layer2);
[Weigth3,Bias3]=rands(1,Layer3);
dWeigth1=zeros(size(Weigth1));
dWeigth2=zeros(size(Weigth2));
dWeigth3=zeros(size(Weigth3));
dBias1=zeros(size(Bias1));
dBias2=zeros(size(Bias2));
dBias3=zeros(size(Bias3));
K=250;
imramn(1)=0;
for k=1:K;
isxn(k)=rand*2-1;
imramn(k+1)=b*imramn(k)+c*isxn(k);
end
for n=1:50000;
Hidden1=tansig(Weigth1*[imramn(1:K);isxn],Bias1);
Hidden2=tansig(Weigth2*Hidden1,Bias2);
Output=purelin(Weigth3*Hidden2,Bias3);
e1=imramn(2:K+1)-Output;
Delta3=deltalin(Output,e1);
Delta2=deltatan(Hidden2,Delta3,Weigth3);
Delta1=deltatan(Hidden1,Delta2,Weigth2);
[dWeigth1,dBias1]=learnbpm([imramn(1:K);isxn],Delta1,lr,mc,dWeigth1,dBias1);
[dWeigth2,dBias2]=learnbpm(Hidden1,Delta2,lr,mc,dWeigth2,dBias2);
[dWeigth3,dBias3]=learnbpm(Hidden2,Delta3,lr,mc,dWeigth3,dBias3);
Weigth1=Weigth1+dWeigth1; Bias1=Bias1+dBias1;
Weigth2=Weigth2+dWeigth2; Bias2=Bias2+dBias2;
```

```
Weigth3=Weigth3+dWeigth3; Bias3=Bias3+dBias3;
if n==50*y
n;
if y>5
yy=y-5
else
yy=1;
end;
err1(y)=sum(e1.^2);
subplot(2,1,1);
plot(yy:y,err1(yy:y))
xlabel('Yaklaşım >>>');
ylabel('Hata Fonksiyonu');
title('Hata fonksiyonun Değişimi');
drawnow
y=y+1;
imram_n(1)=0.1;
imram_est(1)=0.1;
for kk=1:1000
Hidden1=tansig(Weigth1*[imram_est(kk);aisxn(kk)],Bias1);
Hidden2=tansig(Weigth2*Hidden1,Bias2);
Output=purelin(Weigth3*Hidden2,Bias3);
imram_est(kk+1)=Output;
imram_n(kk+1)=b*imram_n(kk)+c*aisxn(kk);
end
z=1:1001;
subplot(2,1,2)
plot(z,7*imram_n,z,7*imram_est,'k');grid
title('Gerçek ve Estimate Edilen Mıknatıslanma Akımı Eğrileri');
xlabel('Zaman (s.)');
ylabel('Imr_amplitude - Imr_estimated')
end;
end
```

EK 4.

```
% Açısal kayma  hızını estimate eden yapay sinir ağının eğitim programı
clear
nntwarn off
Nwsl=1200;
Nimr=8;
Nisy=35;
dt=0.001;
Tr=0.0719;
a=Nisy/(Nwsl*Nimr*Tr);
load vektör_denetimi
aisyn=aisy/Nisy;
aimramn=aimram/Nimr;
y=1;
lr=0.02;
mc=0.95;
layer2=10;
layer3=10;
[weigth1,bias1]=rands(layer2,2);
[weigth2,bias2]=rands(layer3,layer2);
[weigth3,bias3]=rands(1,layer3);
dweigth1=zeros(size(weigth1));
dweigth2=zeros(size(weigth2));
dweigth3=zeros(size(weigth3));
dbias1=zeros(size(bias1));
dbias2=zeros(size(bias2));
dbias3=zeros(size(bias3));
K=250;
wsln(1)=NaN;
for k=1:K;
isyn(k+1)=rand*2-1;
imramn(k+1)=rand*2-1;
wsln(k+1)=a*isyn(k+1)/imramn(k+1);
end
for n=1:50000;
hidden1=tansig(weigth1*[imramn(2:K+1);isyn(2:K+1)],bias1);
hidden2=tansig(weigth2*hidden1,bias2);
output=purelin(weigth3*hidden2,bias3);
e=wsln(2:K+1)-output;
delta3=deltalin(output,e);
delta2=deltatan(hidden2,delta3,weigth3);
delta1=deltatan(hidden1,delta2,weigth2);
```

```
[dweigth1,dbias1]=learnbpm([imramn(2:K+1);isyn(2:K+1)],delta1,lr,mc,dweigth1,dbi
as1);
[dweigth2,dbias2]=learnbpm(hidden1,delta2,lr,mc,dweigth2,dbias2);
[dweigth3,dbias3]=learnbpm(hidden2,delta3,lr,mc,dweigth3,dbias3);
weigth1=weigth1+dweigth1; bias1=bias1+dbias1;
weigth2=weigth2+dweigth2; bias2=bias2+dbias2;
weigth3=weigth3+dweigth3; bias3=bias3+dbias3;
if n==50*y
n;
if y>5
yy=y-5
else
yy=1;
end;
err(y)=sum(e.^2);
subplot(2,1,1);
plot(yy:y,err(yy:y))
drawnow
y=y+1;
wsl_n(1)=0.1;
wsl_est(1)=0.1;
for kk=1:1000
hidden1=tansig(weigth1*[aimramn(kk+1);aisyn(kk+1)],bias1);
hidden2=tansig(weigth2*hidden1,bias2);
output=purelin(weigth3*hidden2,bias3);
wsl_est(kk+1)=output;
wsl_n(kk+1)=a*aisyn(kk+1)/aimramn(kk+1);
end
z=1:1001;
subplot(2,1,2)
plot(z,Nwsl*wsl_n,z,Nwsl*wsl_est,'k')
xlabel('Zaman (s.)');
ylabel('wsl_reel - wsl_estimated');
title('Açısal Kayma Hızı');
end
end
```

EK 5.

```
% Rotor hızını algılayan yapay sinir ağının off-line eğitim algoritması
clear
nntwarn off
lamda=0.01;
J=8e-3; dt=0.001; p=1;
Nme=50;Nwr=400;
a=(1-lamda*dt/J);b=dt*Nme/(J*Nwr);
load vektör_denetimi
amen=ame/Nme;
awrn=awr/Nwr;
lr=0.02; mc=0.95;
layer2=10;layer3=10;
[weigth1,bias1]=rands(layer2,2);
[weigth2,bias2]=rands(layer3,layer2);
[weigth3,bias3]=rands(1,layer3);
dweigth1=zeros(size(weigth1));
dweigth2=zeros(size(weigth2));
dweigth3=zeros(size(weigth3));
dbias1=zeros(size(bias1));
dbias2=zeros(size(bias2));
dbias3=zeros(size(bias3));
K=400; wrn(1)=0;
for k=1:K;
men(k)=rand*2-1;
wrn(k+1)=a*wrn(k)+b*men(k);
end
for n=1:50000;
hidden1=logsig(weigth1*[wrn(1:K);men(1:K)],bias1);
hidden2=logsig(weigth2*hidden1,bias2);
output=purelin(weigth3*hidden2,bias3);
ee=wrn(2:K+1)-output;
delta3=deltalin(output,ee);
delta2=deltatan(hidden2,delta3,weigth3);
delta1=deltatan(hidden1,delta2,weigth2);
[dweigth1,dbias1]=learnbpm([wrn(1:K);men(1:K)],delta1,lr,mc,dweigth1,dbias1);
[dweigth2,dbias2]=learnbpm(hidden1,delta2,lr,mc,dweigth2,dbias2);
[dweigth3,dbias3]=learnbpm(hidden2,delta3,lr,mc,dweigth3,dbias3);
weigth1=weigth1+dweigth1; bias1=bias1+dbias1;
weigth2=weigth2+dweigth2; bias2=bias2+dbias2;
weigth3=weigth3+dweigth3; bias3=bias3+dbias3;
if n==50*p
p;
```

```
if p>5
q=p-5
else
q=1;
end;
Err(p)=sum(ee.^2);
subplot(2,1,1);
plot(q:p,Err(q:p))
drawnow
p=p+1;
wr_est(1)=0;wr_n(1)=0;
for kk=1:1000
hidden1=logsig(weigth1*[awrn(kk);amen(kk)],bias1);
hidden2=logsig(weigth2*hidden1,bias2);
output=purelin(weigth3*hidden2,bias3);
wr_est(kk+1)=output;
wr_n(kk+1)=a*wr_n(kk)+b*amen(kk);
end
z=1:1001;
subplot(2,1,2)
plot(z,Nwr*wr_n,z,Nwr*wr_est,'-.');grid;
end;
end
```

EK 6.

```
% Neural tabanlı akı gözleyicileri içeren vektör denetimi algoritması
clear
nntwarn off
t0=clock;
time=1.0; dt=0.001;
Nimr=8; Nisx=50; Nisy=35; Nwsl=1200;
lamda=0.01;
p=2; J=8e-3;
rs=1.5; rr=1.6;
ls=109e-3; lr=115e-3; m=98e-3;
s=1-m^2/(ls*lr);
lsg=s*ls;
 Tsg=lsg/rs;
lrg=s*lr;
Tr=lr/rr;
Trg=lrg/rr;
wr=0;
i=zeros(4,1);
U=[180*ones(2,1);zeros(2,1)];
wsl=0;
imran=0;
imram=0;
wref=180;
cakı=0;
cisx=0;
chız=0;
cmom=0;
cisy=0;
kp1=13.45;
kı1=0.4;
kp2=7.7;
kı2=0.8;
kp3=2.7;
kı3=0.1;
kp4=2.7;
kı4=0.6;
kp5=1.9;
kı5=0.19;
A=[-(1/Tsg+(1-s)/Trg) 0 (1-s)/Trg 0;0 -(1/Tsg+(1-s)/Trg) 0 (1-s)/Trg;1/Tr 0 -1/Tr
0;0 1/Tr 0 -1/Tr];
B=[0 0 0 (1-s)/s;0 0 -(1-s)/s 0;0 0 0 -1;0 0 1 0];
```

```
C=[1/lsg 0 0 0;0 1/lsg 0 0;0 0 0 0;0 0 0 0];
load imram_weights
load wsl_weights
for t=0:dt:time-dt
me=3/2*p*(m^2)/lr*(i(3)*i(2)-i(4)*i(1));
if t~=0
U=[usdref;usqref;0;0];
end
ki1=(A*i+B*wr*p*i+C*U)*dt;
ki2=(A*(i+0.5*ki1)+B*wr*p*(i+0.5*ki1)+C*U)*dt;
ki3=(A*(i+0.5*ki2)+B*wr*p*(i+0.5*ki2)+C*U)*dt;
ki4=(A*(i+ki3)+B*wr*p*(i+ki3)+C*U)*dt;
i=i+1/6*(ki1+2*ki2+2*ki3+ki4);
isx=i(1)*cos(imran)+i(2)*sin(imran);
isy=i(2)*cos(imran)-i(1)*sin(imran);
isxn=isx/Nisx;
isyn=isy/Nisy;
imramn=imram/Nimr;
Hidden1=tansig(Weigth1*[imramn;isxn],Bias1);
Hidden2=tansig(Weigth2*Hidden1,Bias2);
Output=purelin(Weigth3*Hidden2,Bias3);
imramn=Output;
imram=imramn*Nimr;
if imram==NaN
imram=0;
end
wr=(1-lamda*dt/J)*wr+dt/J*me;
hidden1=tansig(weigth1*[imramn;isyn],bias1);
hidden2=tansig(weigth2*hidden1,bias2);
output=purelin(weigth3*hidden2,bias3);
wsln=output;
wsl=wsln*Nwsl;
ml=lamda*wr;
Wsl((t+dt)/dt)=wsl;
imran=imran+dt*(wr*p+isy/(Tr*imram));
if imran==NaN
imran=0;
end
wmr=(i(3)*((i(2)-i(4))/Tr+wr*p*i(3))-i(4)*((i(1)-i(3))/Tr-
wr*p*i(4)))/(i(3)^2*(1+(i(4)/i(3))^2));
udx=-wmr*lsg*isy;
udy=wmr*lsg*isx+(ls-lsg)*wmr*imram;
if abs(wr)>184 & abs(wr)<265
imref=6.1-abs(wr)/30.16+6.1;
elseif abs(wr)<=184
imref=6.1;
else
imref=0;
end
```

```
cakı=cakı+dt*kp1/kı1*(imref-imram);
isxref=kp1*(imref-imram)+cakı;
cisx=cisx+dt*kp2/kı2*(isxref-isx);
usxes=kp2*(isxref-isx)+cisx;
usxref=usxes+udx;
chız=chız+dt*kp3/kı3*(wref-wr);
meref=kp3*(wref-wr)+chız;
if meref>40
meref=40;
end
if meref<-40
meref=-40;
end
cmom=cmom+dt*kp4/kı4*(meref-me);
isyref=kp4*(meref-me)+cmom;
cisy=cisy+dt*kp5/kı5*(isyref-isy);
usyes=kp5*(isyref-isy)+cisy;
usyref=udy+usyes;
usdref=usxref*cos(imran)-usyref*sin(imran);
usqref=usyref*cos(imran)+usxref*sin(imran);
if usdref>800
usdref=800;
elseif usdref<-800
usdref=-800;
end
if usqref>800
usqref=800;
elseif usqref<-800
usqref=-800;
end
Me((t+dt)/dt)=me;
Ml((t+dt)/dt)=ml;
I(:,(t+dt)/dt)=i;
Wr((t+dt)/dt)=wr;
Isx((t+dt)/dt)=isx;
Isy((t+dt)/dt)=isy;
Imram((t+dt)/dt)=imram;
Imran((t+dt)/dt)=imran;
Us(:,(t+dt)/dt)=[usdref;usqref];
Chız((t+dt)/dt)=chız;
Cakı((t+dt)/dt)=cakı;
Cisx((t+dt)/dt)=cisx;
Cmom((t+dt)/dt)=cmom;
Cisy((t+dt)/dt)=cisy;
Wmr((t+dt)/dt)=wmr;
Imref((t+dt)/dt)=imref;
Isxref((t+dt)/dt)=isxref;
Meref((t+dt)/dt)=meref;
Isyref((t+dt)/dt)=isyref;
```

```
Usxes((t+dt)/dt)=usxes;
Usyes((t+dt)/dt)=usyes;
Usxref((t+dt)/dt)=usxref;
Usyref((t+dt)/dt)=usyref;
end
mes=3/2*p*(m^2)/lr*(i(3)*i(2)-i(4)*i(1));
ame=[Me mes];
aimram=[0 Imram];
aimran=[0 Imran];
tt=0:dt:time;
ai(:,1)=[0;0;0;0];
ai(:,2:(time+dt)/dt)=I;
awr=[0 Wr];
aml=[0 Ml];
aisx=[0 Isx];
aisy=[0 Isy];
ausxes=[0 Usxes];
ausyes=[0 Usyes];
ausxref=[0 Usxref];
ausyref=[0 Usyref];
aus(:,1)=[0;0];
aus(:,2:(time+dt)/dt)=Us;
acakı=[0 Cakı];
acisx=[0 Cisx];
acmom=[0 Cmom];
acisy=[0 Cisy];
achız=[0 Chız];
awsl=[0 Wsl];
awmr=[0 Wmr];
aimref=[6.1 Imref];
aisxref=[0 Isxref];
aisyref=[0 Isyref];
ameref=[0 Meref];
sure=etime(clock,t0);
 plot(tt,ame);grid
plot(tt,awr);grid;
compass(ai(1,:),ai(2,:))
plot(tt,aimram);grid;
plot(tt,aimran);grid;
plot(tt,aus(1,:));
plot(tt,wmr);grid;
plot(tt,awsl);grid;
end
```

EK 7.

```
% Neural tabanlı rotor hızı algılayıcısı içeren vektör denetimi algoritması
clear
nntwarn off
t0=clock;
Nme=50;
Nwr=400;
time=1.0; dt=0.001;
lamda=0.01;
p=2; J=8e-3;
rs=1.5;
rr=1.6;
ls=109e-3;
lr=115e-3;
m=98e-3;
s=1-m^2/(ls*lr);
lsg=s*ls;
Tsg=lsg/rs;
lrg=s*lr;
Tr=lr/rr;
Trg=lrg/rr;
wr=0;
i=zeros(4,1);
U=[180*ones(2,1);zeros(2,1)];
wsl=0;
imran=0;
imram=0;
wref=180;
cakı=0;
cisx=0;
chız=0;
cmom=0;
cisy=0;
kp1=13.45;
kı1=0.4;
kp2=7.7;
kı2=0.8;
kp3=2.7;
kı3=0.1;
kp4=2.7;
kı4=0.6;
kp5=1.9;
kı5=0.19;
```

```
A=[-(1/Tsg+(1-s)/Trg) 0 (1-s)/Trg 0;0 -(1/Tsg+(1-s)/Trg) 0 (1-s)/Trg;1/Tr 0 -1/Tr
0;0 1/Tr 0 -1/Tr];
B=[0 0 0 (1-s)/s;0 0 -(1-s)/s 0;0 0 0 -1;0 0 1 0];
C=[1/lsg 0 0 0;0 1/lsg 0 0;0 0 0 0;0 0 0 0];
load wr_weights
for t=0:dt:time-dt
me=3/2*p*(m^2)/lr*(i(3)*i(2)-i(4)*i(1));
if t~=0
U=[usdref;usqref;0;0];
end
ki1=(A*i+B*wr*p*i+C*U)*dt;
ki2=(A*(i+0.5*ki1)+B*wr*p*(i+0.5*ki1)+C*U)*dt;
ki3=(A*(i+0.5*ki2)+B*wr*p*(i+0.5*ki2)+C*U)*dt;
ki4=(A*(i+ki3)+B*wr*p*(i+ki3)+C*U)*dt;
i=i+1/6*(ki1+2*ki2+2*ki3+ki4);
isx=i(1)*cos(imran)+i(2)*sin(imran);
isy=i(2)*cos(imran)-i(1)*sin(imran);
k1g=(-imram/Tr+isx/Tr)*dt;
k2g=(-(imram+0.5*k1g)/Tr+isx/Tr)*dt;
k3g=(-(imram+0.5*k2g)/Tr+isx/Tr)*dt;
k4g=(-(imram+k3g)/Tr+isx/Tr)*dt;
imram=imram+1/6*(k1g+2*k2g+2*k3g+k4g);
if imram==NaN
imram=0;
end
men=me/Nme;
wrn=wr/Nwr;
hidden1=logsig(weigth1*[wrn;men],bias1);
hidden2=logsig(weigth2*hidden1,bias2);
output=purelin(weigth3*hidden2,bias3);
wrn=output;
wr=wrn*Nwr;
ml=lamda*wr;
wsl=isy/(Tr*imram);
Wsl((t+dt)/dt)=wsl;
imran=imran+dt*(wr*p+isy/(Tr*imram));
if imran==NaN
imran=0;
end
wmr=(i(3)*((i(2)-i(4))/Tr+wr*p*i(3))-i(4)*((i(1)-i(3))/Tr-
wr*p*i(4)))/(i(3)^2*(1+(i(4)/i(3))^2));
udx=-wmr*lsg*isy;
udy=wmr*lsg*isx+(ls-lsg)*wmr*imram;
if abs(wr)>184 & abs(wr)<265
imref=6.1-abs(wr)/30.16+6.1;
elseif abs(wr)<=184
imref=6.1;
else
imref=0;
```

```
end
cakı=cakı+dt*kp1/kı1*(imref-imram);
isxref=kp1*(imref-imram)+cakı;
cisx=cisx+dt*kp2/kı2*(isxref-isx);
usxes=kp2*(isxref-isx)+cisx;
usxref=usxes+udx;
chız=chız+dt*kp3/kı3*(wref-wr);
meref=kp3*(wref-wr)+chız;
if meref>40
meref=40;
end
if meref<-40
meref=-40;
end
cmom=cmom+dt*kp4/kı4*(meref-me);
isyref=kp4*(meref-me)+cmom;
cisy=cisy+dt*kp5/kı5*(isyref-isy);
usyes=kp5*(isyref-isy)+cisy;
usyref=udy+usyes;
usdref=usxref*cos(imran)-usyref*sin(imran);
usqref=usyref*cos(imran)+usxref*sin(imran);
if usdref>800
usdref=800;
elseif usdref<-800
usdref=-800;
end
if usqref>800
usqref=800;
elseif usqref<-800
usqref=-800;
end
Me((t+dt)/dt)=me;
Ml((t+dt)/dt)=ml;
I(:,(t+dt)/dt)=i;
Wr((t+dt)/dt)=wr;
Isx((t+dt)/dt)=isx;
Isy((t+dt)/dt)=isy;
Imram((t+dt)/dt)=imram;
Imran((t+dt)/dt)=imran;
Us(:,(t+dt)/dt)=[usdref;usqref];
Chız((t+dt)/dt)=chız;
Cakı((t+dt)/dt)=cakı;
Cisx((t+dt)/dt)=cisx;
Cmom((t+dt)/dt)=cmom;
Cisy((t+dt)/dt)=cisy;
Wmr((t+dt)/dt)=wmr;
Imref((t+dt)/dt)=imref;
Isxref((t+dt)/dt)=isxref;
Meref((t+dt)/dt)=meref;
```

```
Isyref((t+dt)/dt)=isyref;
Usxes((t+dt)/dt)=usxes;
Usyes((t+dt)/dt)=usyes;
Usxref((t+dt)/dt)=usxref;
Usyref((t+dt)/dt)=usyref;
end
mes=3/2*p*(m^2)/lr*(i(3)*i(2)-i(4)*i(1));
ame=[Me mes];
aimram=[0 Imram];
aimran=[0 Imran];
tt=0:dt:time;
ai(:,1)=[0;0;0;0];
ai(:,2:(time+dt)/dt)=I;
awr=[0 Wr];
aml=[0 Ml];
aisx=[0 Isx];
aisy=[0 Isy];
ausxes=[0 Usxes];
ausyes=[0 Usyes];
ausxref=[0 Usxref];
ausyref=[0 Usyref];
aus(:,1)=[0;0];
aus(:,2:(time+dt)/dt)=Us;
acakı=[0 Cakı];
acisx=[0 Cisx];
acmom=[0 Cmom];
acisy=[0 Cisy];
achız=[0 Chız];
awsl=[0 Wsl];
awmr=[0 Wmr];
aimref=[6.1 Imref];
aisxref=[0 Isxref];
aisyref=[0 Isyref];
ameref=[0 Meref];
sure=etime(clock,t0);
plot(tt,ame);grid
plot(tt,awr);grid;
compass(ai(1,:),ai(2,:))
plot(tt,aimram);grid;
plot(tt,aimran);grid;
plot(tt,aus(1,:));
plot(tt,wmr);grid;
plot(tt,awsl);grid;
end
```

EK 8.

```
% Parametre adaptasyonlu neural tabanlı vektör denetimi simülasyonu
clear
nntwarn off
t0=clock;
time=0.8;
dt=0.001;
Nimr=8;
Nisx=50;
Nisy=35;
Nwsl=1200;
lamda=0.01;
Lr=0.01;
mc=0.95;
p=2;
J=8e-3;
rs=1.5;
rr=1.6;
ls=109e-3;
lr=115e-3;
m=98e-3;
s=1-m^2/(ls*lr);
lsg=s*ls;
Tsg=lsg/rs;
lrg=s*lr;
Tr=lr/rr;
Trg=lrg/rr;
wr=0;
i=zeros(4,1);
U=[180*ones(2,1);zeros(2,1)];
wsl=0;
imran=0;
imram=0;
wref=180;
cakı=0;
cisx=0;
chız=0;
cmom=0;
cisy=0;
kp1=13.45;
kı1=0.4;
kp2=7.7;
```

```
ki2=0.8;
kp3=2.7;
ki3=0.1;
kp4=2.7;
ki4=0.6;
kp5=1.9;
ki5=0.19;
load imram_weights
load wsl_weights
for t=0:dt:time-dt
A=[-(1/Tsg+(1-s)/Trg) 0 (1-s)/Trg 0;0 -(1/Tsg+(1-s)/Trg) 0 (1-s)/Trg;1/Tr 0 -1/Tr
0;0 1/Tr 0 -1/Tr];
B=[0 0 0 (1-s)/s;0 0 -(1-s)/s 0;0 0 0 -1;0 0 1 0];
C=[1/lsg 0 0 0;0 1/lsg 0 0;0 0 0 0;0 0 0 0];
me=3/2*p*(m^2)/lr*(i(3)*i(2)-i(4)*i(1));
if t~=0
U=[usdref;usqref;0;0];
end
ki1=(A*i+B*wr*p*i+C*U)*dt;
ki2=(A*(i+0.5*ki1)+B*wr*p*(i+0.5*ki1)+C*U)*dt;
ki3=(A*(i+0.5*ki2)+B*wr*p*(i+0.5*ki2)+C*U)*dt;
ki4=(A*(i+ki3)+B*wr*p*(i+ki3)+C*U)*dt;
i=i+1/6*(ki1+2*ki2+2*ki3+ki4);
isx=i(1)*cos(imran)+i(2)*sin(imran);
isy=i(2)*cos(imran)-i(1)*sin(imran);
isxn=isx/Nisx;
isyn=isy/Nisy;
imramn=imram/Nimr;
if t>0 & (t/0.2-round(t/0.2))==0
isyavg=sum(abs(Isy.^1))*dt/t;
wslavg=sum(abs(Wsl.^1))*dt/t;
Tr=isyavg/(imram*wslavg);
Trg=Tr*lrg/lr;
K=250;
a=Nisy/(Nwsl*Nimr*Tr);
b=(Tr-dt)/Tr;
c=(dt*Nisx)/(Tr*Nimr);
imrgen(1)=0;
for k=1:K;
isxnn(k)=rand*2-1;
imrgen(k+1)=b*imrgen(k)+c*isxnn(k);
end
dWeigth1=zeros(size(Weigth1));
dWeigth2=zeros(size(Weigth2));
dWeigth3=zeros(size(Weigth3));
dBias1=zeros(size(Bias1));
dBias2=zeros(size(Bias2));
dBias3=zeros(size(Bias3));
for n=1:20000;
```

```
Hidden1=tansig(Weigth1*[imrgen(1:K);isxnn],Bias1);
Hidden2=tansig(Weigth2*Hidden1,Bias2);
Output=purelin(Weigth3*Hidden2,Bias3);
e=imrgen(2:K+1)-Output;
Delta3=deltalin(Output,e);
Delta2=deltatan(Hidden2,Delta3,Weigth3);
Delta1=deltatan(Hidden1,Delta2,Weigth2);
[dWeigth1,dBias1]=learnbpm([imrgen(1:K);isxnn],Delta1,Lr,mc,dWeigth1,dBias1);
[dWeigth2,dBias2]=learnbpm(Hidden1,Delta2,Lr,mc,dWeigth2,dBias2);
[dWeigth3,dBias3]=learnbpm(Hidden2,Delta3,Lr,mc,dWeigth3,dBias3);
Weigth1=Weigth1+dWeigth1; Bias1=Bias1+dBias1;
Weigth2=Weigth2+dWeigth2; Bias2=Bias2+dBias2;
Weigth3=Weigth3+dWeigth3; Bias3=Bias3+dBias3;
end
Q=250;
wslnn(1)=NaN;
for q=1:Q;
isynn(q+1)=rand*2-1;
imrgenn(q+1)=rand*2-1;
wslnn(q+1)=a*isynn(q+1)/imrgenn(q+1);
end
dweigth1=zeros(size(weigth1));
dweigth2=zeros(size(weigth2));
dweigth3=zeros(size(weigth3));
dbias1=zeros(size(bias1));
dbias2=zeros(size(bias2));
dbias3=zeros(size(bias3));
for r=1:20000;
hidden1=tansig(weigth1*[imrgenn(2:Q+1);isynn(2:Q+1)],bias1);
hidden2=tansig(weigth2*hidden1,bias2);
output=purelin(weigth3*hidden2,bias3);
ee=wslnn(2:Q+1)-output;
delta3=deltalin(output,ee);
delta2=deltatan(hidden2,delta3,weigth3);
delta1=deltatan(hidden1,delta2,weigth2);
[dweigth1,dbias1]=learnbpm([imrgenn(2:Q+1);isynn(2:Q+1)],delta1,Lr,mc,dweigth1,
dbias1);
[dweigth2,dbias2]=learnbpm(hidden1,delta2,Lr,mc,dweigth2,dbias2);
[dweigth3,dbias3]=learnbpm(hidden2,delta3,Lr,mc,dweigth3,dbias3);
weigth1=weigth1+dweigth1; bias1=bias1+dbias1;
weigth2=weigth2+dweigth2; bias2=bias2+dbias2;
weigth3=weigth3+dweigth3; bias3=bias3+dbias3;
end
end
Hidden1=tansig(Weigth1*[imramn;isxn],Bias1);
Hidden2=tansig(Weigth2*Hidden1,Bias2);
Output=purelin(Weigth3*Hidden2,Bias3);
imramn=Output;
imram=imramn*Nimr;
```

```
if imram==NaN
imram=0;
end
wr=(1-lamda*dt/J)*wr+dt/J*me;
hidden1=tansig(weigth1*[imramn;isyn],bias1);
hidden2=tansig(weigth2*hidden1,bias2);
output=purelin(weigth3*hidden2,bias3);
wsln=output;
wsl=wsln*Nwsl;
ml=lamda*wr;
Wsl((t+dt)/dt)=wsl;
imran=imran+dt*(wr*p+isy/(Tr*imram));
if imran==NaN
imran=0;
end
wmr=(i(3)*((i(2)-i(4))/Tr+wr*p*i(3))-i(4)*((i(1)-i(3))/Tr-
wr*p*i(4)))/(i(3)^2*(1+(i(4)/i(3))^2));
udx=-wmr*lsg*isy;
udy=wmr*lsg*isx+(ls-lsg)*wmr*imram;
if abs(wr)>184 & abs(wr)<265
imref=6.1-abs(wr)/30.16+6.1;
elseif abs(wr)<=184
imref=6.1;
else
imref=0;
end
cakı=cakı+dt*kp1/kı1*(imref-imram);
isxref=kp1*(imref-imram)+cakı;
cisx=cisx+dt*kp2/kı2*(isxref-isx);
usxes=kp2*(isxref-isx)+cisx;
usxref=usxes+udx;
chız=chız+dt*kp3/kı3*(wref-wr);
meref=kp3*(wref-wr)+chız;
if meref>40
meref=40;
end
if meref<-40
meref=-40;
end
cmom=cmom+dt*kp4/kı4*(meref-me);
isyref=kp4*(meref-me)+cmom;
cisy=cisy+dt*kp5/kı5*(isyref-isy);
usyes=kp5*(isyref-isy)+cisy;
usyref=udy+usyes;
usdref=usxref*cos(imran)-usyref*sin(imran);
usqref=usyref*cos(imran)+usxref*sin(imran);
if usdref>800
usdref=800;
elseif usdref<-800
```

```
usdref=-800;
end
if usqref>800
usqref=800;
elseif usqref<-800
usqref=-800;
end
Me((t+dt)/dt)=me;
Ml((t+dt)/dt)=ml;
I(:,(t+dt)/dt)=i;
Wr((t+dt)/dt)=wr;
Isx((t+dt)/dt)=isx;
Isy((t+dt)/dt)=isy;
Imram((t+dt)/dt)=imram;
Imran((t+dt)/dt)=imran;
Us(:,(t+dt)/dt)=[usdref;usqref];
Chız((t+dt)/dt)=chız;
Cakı((t+dt)/dt)=cakı;
Cisx((t+dt)/dt)=cisx;
Cmom((t+dt)/dt)=cmom;
Cisy((t+dt)/dt)=cisy;
Wmr((t+dt)/dt)=wmr;
Imref((t+dt)/dt)=imref;
Isxref((t+dt)/dt)=isxref;
Meref((t+dt)/dt)=meref;
Isyref((t+dt)/dt)=isyref;
Usxes((t+dt)/dt)=usxes;
Usyes((t+dt)/dt)=usyes;
Usxref((t+dt)/dt)=usxref;
Usyref((t+dt)/dt)=usyref;
end
mes=3/2*p*(m^2)/lr*(i(3)*i(2)-i(4)*i(1));
ame=[Me mes];
aimram=[0 Imram];
aimran=[0 Imran];
tt=0:dt:time;
ai(:,1)=[0;0;0;0];
ai(:,2:(time+dt)/dt)=I;
awr=[0 Wr];
aml=[0 Ml];
aisx=[0 Isx];
aisy=[0 Isy];
ausxes=[0 Usxes];
ausyes=[0 Usyes];
ausxref=[0 Usxref];
ausyref=[0 Usyref];
aus(:,1)=[0;0];
aus(:,2:(time+dt)/dt)=Us;
acakı=[0 Cakı];
```

```
acisx=[0 Cisx];
acmom=[0 Cmom];
acisy=[0 Cisy];
achız=[0 Chız];
awsl=[0 Wsl];
awmr=[0 Wmr];
aimref=[6.1 Imref];
aisxref=[0 Isxref];
aisyref=[0 Isyref];
ameref=[0 Meref];
sure=etime(clock,t0);
plot(tt,ame);
grid
plot(tt,awr);
grid;
compass(ai(1,:),ai(2,:))
plot(tt,aimram);
grid;
plot(tt,aimran);
grid;
plot(tt,aus(1,:));
plot(tt,wmr);
grid;
plot(tt,awsl);
grid;
end
```

ÖZGEÇMİŞ

8 Kasım 1976 da İstanbul'da doğdu. İlk ve orta öğrenimimi Fikret Yüzatlı İlkokulu ve Bahçelievler Ortaokulunda tamamladı. 1993 de Bahçelievler lisesinden mezun oldu ve aynı sene İstanbul Teknik Üniversitesi Elektrik-Elektronik fakültesinin Elektrik Mühendisliği bölümüne girmeye hak kazandı. 1994 de aynı üniversitenin hazırlık sınıfını tamamladıktan sonra 1994 de lisans programına başladı. Haziran 1998 de Elektrik Mühendisliği bölümünden mezun oldu ve ona müteakip dönemde yine aynı üniversitede Elektrik Mühendisliği anabilim dalı, Elektrik Mühendisliği programında yüksek lisansa başladı ve Mart 2001 de mezun oldu.