**ANKARA ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

**YÜKSEK LİSANS TEZİ**

# DOĞRUSAL OLMAYAN SÜREÇ DENETİM TASARIMINA YAPAY SİNİR AĞLARININ UYGULANMASI

**Umut AKINCIOĞLU**

**ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI**

**ANKARA**
**2006**

**Her hakkı saklıdır**

Prof. Dr. Ahmet DENKER danışmanlığında, Umut AKINCIOĞLU tarafından hazırlanan "Doğrusal Olmayan Süreç Tasarımına Yapay Sinir Ağlarının Uygulanması" adlı tez çalışması 07/11/2006 tarihinde aşağıdaki jüri tarafından oybirliği ile Ankara Üniversitesi Fen Bilimleri Enstitüsü Elektronik Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Başkan: Doç. Dr. Mehmet Önder EFE
TOBB Ekonomi ve Teknoloji Üniversitesi
Elektrik Elektronik Mühendisliği Bölümü

Üye : Prof. Dr. Ahmet DENKER
Ankara Üniversitesi
Elektronik Mühendisliği Bölümü

Üye : Yrd. Doç. Dr. Ziya TELATAR
Ankara Üniversitesi
Elektronik Mühendisliği Bölümü

**Yukarıdaki sonucu onaylarım**

**Prof. Dr. Ülkü MEHMETOĞLU**
**Enstitü Müdürü**

# ÖZET

Yüksek Lisans Tezi

## DOĞRUSAL OLMAYAN SÜREÇ DENETİM TASARIMINA YAPAY SİNİR AĞLARININ UYGULANMASI

Umut AKINCIOĞLU

Ankara Üniversitesi
Fen Bilimleri Enstitüsü
Elektronik Mühendisliği Ana bilim Dalı

Danışman: Prof. Dr. Ahmet DENKER

Gerçek teknolojik süreçler çok değişkenli ve karmaşık olmalarının yanı sıra genellikle doğrusal değildirler. Ayrıca gürültü ve çevresel etkenler nedeniyle süreç değişkenleri zamana göre değişmektedir. Klasik süreç denetim yöntemleri bu tür sistemlerin denetiminde yetersiz kalabilmektedir. Bu yetersizliğe cevap olarak, bu tür süreçlerin denetiminde yapay sinir ağlarından yararlanılması literatürde sıkça karşılaşılan bir yöntemdir.

Bu tez çalışmasında nöral uyarlanır denetçi oluşturulmuş ve robot denetiminde uygulanmıştır. Direk ters denetim ve öngörülü denetim uygulamalarından elde edilen verilerden yararlanarak iki yöntemin de olumlu yanlarından yararlanmak amacıyla bu iki yöntem arasında anahtarlama yapan bir yöntemin uygulama sonuçları da sunulmuştur. Amaç, yapay sinir ağlarıyla süreç denetiminde, özellikle de direk ters modelleme ve öngörülü denetim uygulamalarındaki kazanımları yansıtmaktır. Yapılan simülasyonlar ile denetçinin etkinliği gösterilmiştir. Uygulamalar bilgisayar ortamında simülasyon şeklinde yapılmıştır. Denetim için seçilen robot doğrudan sürümlü iki eksenli SCARA tipi bir robottur. Elde edilen test sonuçları, bu denetçi ile erişilen performansın sadece direk ters modelleme ile ya da sadece öngörülü denetim uygulamaları ile elde edilen performanstan daha iyi olduğunu göstermiştir. Ek olarak, elde edilen metodun direk ters modelleme ile öngörülü denetimin avantajlarını birleştiren faydalı bir metot olduğu görülmüştür.

**2006, 58 sayfa**

**Anahtar Kelimeler:** Yapay sinir ağları, robotbilim, direk ters modelleme, öngörülü denetim,

## ABSTRACT

Master Thesis

APPLICATION OF ARTIFICIAL NEURAL NETWORKS TO DESIGN OF NONLINEAR PROCESS CONTROL

Umut AKINCIOĞLU

Ankara University
Graduate School of Natural and Applied Sciences
Department of Electronical Engineering

Supervisor: Prof.Dr. Ahmet DENKER

Technological processes are multivariable and alongside their complexity they are nonlinear. Process variables may be time varying because of the environmental disturbances and noise. Classical control algorithms may not be adequate to control such processes. As a remedy, artificial neural networks have frequently appeared in literature to control such processes.

In this thesis, a neural adaptive control method has been developed and applied to robot control. A switching technique between direct inverse control and predictive control is proposed to exploit the useful parts of these two approaches. In this thesis applications are carried out as computer simulations. The robot used in the applications is the direct drive SCARA type robot with two joints. The process is assumed to be deterministic. Simulation results are presented to verify the effectiveness of the controller. These results show that the performance by using this controller is better than those using either direct inverse control or predictive control. In addition, they show that the resulting method is a useful method which combines the advantages of both direct inverse control and predictive control.



**2006, 58 pages**

**Key Words:** Neural networks, robotics, direct inverse control, predictive control

**TEŞEKKÜR**

Çalışmalarımı yönlendiren, araştırmalarımın her aşamasında bilgi, öneri ve yardımlarını esirgemeyen danışman hocam sayın Prof. Dr. Ahmet DENKER'e, daha önceden danışmanlığımı yapmış olan sayın Prof Dr. Abdulrıza ABİLOV ve sayın Prof. Dr. Önder TÜZİNALP'e, bu çalışmalarım süresince bana destek olan ASELSAN AŞ'ye, mesai arkadaşlarıma, desteklerinden dolayı aileme, çalışmalarım süresince birçok fedakarlıklar göstererek beni destekleyen sevgili eşim Egemen'e en derin duygularla teşekkür ederim.

Umut AKINCOĞLU
Ankara, Kasım 2006

# İÇİNDEKİLER


**ÖZET** ..... **i**
**ABSTRACT** ..... **ii**
**TEŞEKKÜR** ..... **iii**
**İÇİNDEKİLER** ..... **iv**
**ŞEKİLLER DİZİNİ** ..... **v**
**ÇİZELGE DİZİNİ** ..... **vi**
**1. GİRİŞ** ..... **1**
**2. YAPAY SİNİR AĞLARI** ..... **2**
**2.1 Biyolojik Altyapı** ..... **2**
**2.2 Temel Nöron Modeli** ..... **3**
**2.3 Ağ Yapıları** ..... **4**
**2.3.1 İleri sürümlü (Çok katmanlı) yapay sinir ağları** ..... **5**
**2.3.2 Ağ eğitimi** ..... **7**
**2.3.2.1 Hata geriye yayma eğitim algoritması** ..... **8**
**3. YAPAY SİNİR AĞLARININ SÜREÇ DENETİMİ UYGULAMALARI** ..... **12**
**3.1 Tanılama** ..... **13**
**3.1.1 ARX model yapısı** ..... **14**
**3.2 Süreç Denetimi** ..... **16**
**3.2.1 Direk ters denetim** ..... **16**
**3.2.2 Model dayanaklı denetim** ..... **17**
**3.2.3 İçsel model denetimi** ..... **18**
**3.2.4 Öngörülü denetim** ..... **19**
**4. SÜREÇ DENETİM UYGULAMALARI** ..... **24**
**4.1 İki Eksenli Doğrudan Sürümlü SCARA Tipi Robot** ..... **24**
**4.2 Direk Ters Modelleme ile Süreç Denetim Uygulaması** ..... **26**
**4.2.1 Sinüzoidal referans hız ile süreç denetim uygulaması** ..... **27**
**4.2.2 Adım referans hız ile süreç denetim uygulaması** ..... **29**
**4.3 Öngörülü Denetim ile Süreç Denetim Uygulaması** ..... **31**
**4.3.1 Sinüzoidal referans hız ile süreç denetim uygulaması** ..... **32**
**4.3.2 Adım referans hız ile süreç denetim uygulaması** ..... **35**
**4.4 Uyarlamalı Anahtarlama Denetim Uygulaması** ..... **39**
**4.4.1 RY=0.02 için yapılan uygulamalar** ..... **41**
**4.4.2 RY=0.05 için yapılan uygulamalar** ..... **44**
**KAYNAKLAR** ..... **50**
**EK 1 Kullanılan Matlab Fonksiyonları** ..... **52**
**ÖZGEÇMİŞ** ..... **58**

# ŞEKİLLER DİZİNİ


**Şekil 2.1 Nöron Modeli ..... 3**
**Şekil 2.2 Üç Katmanlı Ağ Modeli ..... 5**
**Şekil 3.1 Model Dayanaklı Kontrol Yapısı ..... 18**
**Şekil 3.2 İçsel Model Denetim Yapısı ..... 19**
**Şekil 4.1 Uygulamalarda Verilerinden Faydalanılan SCARA Tipi İki Eksenli Robot ..... 24**
**Şekil 4.2 Direk Ters Model Uygulaması Blok Şeması ..... 26**
**Şekil 4.3 Direk Ters Modelleme İle Sistemin Çıkış ve Referans Değerleri ..... 27**
**Şekil 4.4 Direk Ters Modelleme İle Sistemin Çıkış Hataları ..... 28**
**Şekil 4.5 Direk Ters Modelleme İle Tork-Zaman Grafikleri ..... 28**
**Şekil 4.6 Direk Ters Modelleme İle Sistemin Çıkış ve Referans Değerleri ..... 29**
**Şekil 4.7 Direk Ters Modelleme İle Sistemin Çıkış Hataları ..... 29**
**Şekil 4.8 Direk Ters Modelleme İle Tork-Zaman Grafikleri ..... 30**
**Şekil 4.9 Öngörülü Denetim Uygulaması Blok Şeması ..... 31**
**Şekil 4.10 Öngörülü Denetim ile Sistemin Çıkış ve Referans Değerleri ..... 32**
**Şekil 4.11 Öngörülü Denetim ile Sistemin Çıkış Hataları ..... 33**
**Şekil 4.12 Öngörülü Denetim İle Tork-Zaman Grafikleri ..... 33**
**Şekil 4.13 Öngörülü Denetim Yapılan Denetim Sonucunda Modelin Çıkışları ..... 34**
**Şekil 4.14 Öngörülü Denetim Yapılan Denetim Sonucunda Model Hatası ..... 34**
**Şekil 4.15 Öngörülü Denetim ile Sistemin Çıkış ve Referans Değerleri ..... 35**
**Şekil 4.16 Öngörülü Denetim ile Sistemin Çıkış Hataları ..... 35**
**Şekil 4.17 Öngörülü Denetim İle Tork-Zaman Grafikleri ..... 36**
**Şekil 4.18 Öngörülü Denetim Yapılan Denetim Sonucunda Modelin Çıkışları ..... 36**
**Şekil 4.19 Öngörülü Denetim Yapılan Denetim Sonucunda Model Hatası ..... 37**
**Şekil 4.20 Mükemmel Model ile Yapılan Öngörülü Denetim Sonucunda Sistemin Çıkış ve Referans Değerleri ..... 38**
**Şekil 4.21 Mükemmel Model ile Yapılan Öngörülü Denetim Sonucunda Sistemin Çıkış Hataları . 38**
**Şekil 4.22 Mükemmel Model ile Yapılan Öngörülü Denetim Sonucunda Tork-Zaman Grafikleri. 39**
**Şekil 4.23 Uyarlamalı Anahtarlama Yöntemi Blok Şeması ..... 40**
**Şekil 4.24 Anahtarlamalı Model İle Sistemin Çıkış ve Referans Değerleri ..... 41**
**Şekil 4.25 Anahtarlamalı Model İle Sistemin Çıkış Hataları ..... 42**
**Şekil 4.26 Anahtarlamalı Model ile Tork-Zaman Grafikleri ..... 42**
**Şekil 4.27 Anahtarlamalı Model İle Yapılan Denetim Sonucunda Modelin Çıkışları ..... 43**
**Şekil 4.28 Anahtarlamalı Model İle Yapılan Denetim Sonucunda Model Hatası ..... 43**
**Şekil 4.29 Anahtarlamalı İle Sistemin Çıkış ve Referans Değerleri ..... 44**
**Şekil 4.30 Anahtarlamalı Model İle Sistemin Çıkış Hataları ..... 44**
**Şekil 4.31 Anahtarlamalı Model ile Tork-Zaman Grafikleri ..... 45**
**Şekil 4.32 Anahtarlamalı Model İle Yapılan Denetim Sonucunda Modelin Çıkışları ..... 45**
**Şekil 4.33 Anahtarlamalı Model İle Yapılan Denetim Sonucunda Model Hatası ..... 46**

# ÇİZELGE DİZİNİ

**Çizelge 4.1 Robot Fiziksel Parametreleri ..... 25**

# 1. GİRİŞ

Süreç denetim uygulamalarında amaç dinamik süreçlerin performansını istenen dinamik davranışları gerçekleştirecek şekilde uyarlamaktır. Klasik kontrol tekniklerinde ilgilenilen sistemin ve ortamın hakkında tam bilgiye sahip olunduğu varsayılır. Bu durumdaki süreçler için geliştirilmiş gözlenebilirlik, kontrol edilebilirlik ve kararlılık özelliklerini incelemek ve aynı zamanda denetleyici tasarlamak için teorik bir alt yapı vardır. Fakat robot sistemlerinin tam olarak matematiksel modelleri elde mevcut değildir. Yaygın robot denetim tekniği modele dayanılarak hesaplanan tork kontrolüdür, bu yöntem model belirsizliğinden kaynaklanan performans düşüşlerine açıktır. Sanayiden gelen yüksek derecede otomasyon, hızlı işlem ve yüksek performans istekleri sonucunda yapay sinir ağları ile robot denetimi üzerine birçok araştırma yapılmış ve belirsizlik problemi ile ilgili birçok önemli noktada tatmin edici sonuçlar elde edilmiştir. Daha iyi endüstriyel performans için daha karmaşık yapay sinir ağı uygulamaları geliştirilmiştir (Cembrano *et al* 1992, Noirega *et al* 1998, Chen *et al* 2001).

Yapay sinir ağları doğaları gereği sahip oldukları özellikleri sonucunda modelden kaynaklanan eksikliklere ve dışarıdan kaynaklanan gürültü ve bozan etmenlere karşı gürbüzlük ve hata toleransı için büyük potansiyel gösterirler. Yeterince büyük ağ oluşturulduğu sürece bu özellikleri sayesinde yapay sinir ağları ile herhangi bir doğrusalsız fonksiyona daha önceden belirlenmiş doğrulukla yaklaşılabilinir.

Yapay sinir ağlarının kabiliyetleri kullanılarak onların, sistemin ve denetçinin bütün karakteristiklerini sınırlama olmadan öğrenmesi sağlanabilir.
Fakat bu sınırsız yapay sinir ağı kullanımı ağların çok fazla büyümelerine yol açacaktır. Bu durum az hesaplama yükü, az enerji ve az maliyet gerektiren gerçek bir uygulama için tabi ki kabul edilmez.

Bu nedenle literatürde yer alan yapay sinir ağları ile süreç denetimi uygulamalarında yapay sinir ağları ya doğrudan denetçide ya da dolaylı olarak sürecin modelinde kullanılmaktadır. Yapay sinir ağlarının kullanımında karşılaşılan sorunları kaldırmak amacıyla bu çalışmada ağların uyarlamalı ve anahtarlamalı yapıda kullanılması önerilmiştir.

# 2. YAPAY SİNİR AĞLARI

## 2.1 Biyolojik Altyapı

Nöronu oluşturan temel parçalar aşağıdaki gibidir:

i. Soma ya da hücre gövdesi: Nöronun mantıksal fonksiyonlarının neredeyse tamamının gerçekleştiği geniş yuvarlak merkezi gövdedir. Hücre gövdesi nöronun canlı kalması için gerekli olan genetik ve metabolik mekanizmayı içerir. Nöron somasında protein sentez mekanizması ve çekirdek vardır.

ii. Akson (çıktı): Somaya bağlanmış sinir lifidir ve nöronun en son çıktı kanalı olarak davranır. Akson genelde çok dallıdır. Aksonun ilk baştaki bölümünde işaretler sinir puls serilerine dönüşerek aksondan hedef hücrelere (diğer nöronlara, reseptörlere ve kaslara vs.) zayıflamadan akson boyunca ilerler.

iii. Dendritler (girdiler) çok dallı lif ağacı yapısındadırlar. Bu uzun ve düzensiz şekilli sinir lifleri gövdeye bağlanmıştır. Nöron başına $10^3 - 10^4$ dendrit düşer. Dendritler nöronu diğer nöronlara bağlar. Dendritler nöronun sinaps denilen özel kontak noktaları vasıtasıyla diğer nöronlardan girdi almasını sağlar ya da diğer dendritleri sinaptik çıkışlara bağlar. Genelde dendritler girdi işaretleri için alıcı yüzeyler oluştururlar ve işaretleri azaltarak hücre gövdesine ya da aksona iletirler.

iv. Sinapslar aksonların diğer nöronlardan ayıran sınır noktalarını oluşturan özelleşmiş kontaklarıdır. Sinapslar bazı hücrelerin aksonları ile girdi dendritlerinin omurgalarını birleştiren ara yüz görevini üstlenir. Sinapslar dendritlerin lokal potansiyelini pozitif ya da negatif yönde değiştirebilirler. Nöronun aktifleşme seviyesini arttırıcı ya da söndürücü olabilmelerine göre sinapslar aktifleştirici ya da söndürücü yapıda olabilir. Nöronda bilgi depolanmasının sinaps bağlantılarında, özellikle bu bağlantıların şekillerinde ve sinaptik bağlantıların gücünde (ağırlıklarında) olduğu düşünülür.

Basitleştirilmiş nöron modeline göre hücre gövdesi (soma), diğer nöronlardan girdileri ayarlanabilir sinaptik dendiritlere olan bağlantılar üzerinden alır. Hücrenin çıkış işareti (sinir darbelerinden oluşan) diğer nöronların sinapsislerine dallanmış akson üzerinden

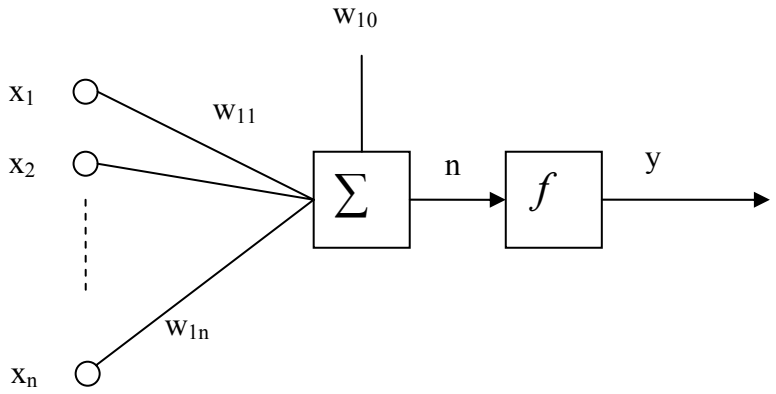
iletilir. Nöron aktifleştiğinde diğer nöronların sinaptik bağlantılarına akson üzerinden iletilen sinir darbeleri (darbe katarı) oluşturur. Çıktı puls oranı (darbe yoğunluğu) girdi işaretinin gücüne ve ilgili sinaptik bağlantıların gücüne ya da ağırlığına bağlıdır (Cichoki *et al* 1993).

## 2.2 Temel Nöron Modeli

Yapay sinir ağlarının temel yapı taşı nörondur (işleme elemanı). Biyolojik nöronu tam olarak tanımlayamadığı için yapay nöron olarak da adlandırılır.

Şekil 2.1.'deki n tane $x_j$ girdisi olan nöron modelini ele alalım. j burada 1'den n'ye kadar olan girdi işaretinin indisidir. Her $x_j$ girdi işareti ana gövdeye ulaşıncaya kadar bağlantı gücü $w_j$ tarafından ağırlıklandırılır. Ek olarak eşik değeri $w_{i0}$ vardır. Üretilen $R_i$ işaretine etki eden doğrusalsız f aktivasyon fonksiyonu ve bunların sonucunda oluşan $y_i$ çıkış işaretidir. $y_i$ diğer nöronlar için giriş işareti olarak kullanılabilir. m tane nöron bulunan bir ağda nöronları belirlemek için bir indis daha (i) kullanılmalıdır.

Şekil 2.1'de temel nöron modeli verilmiştir.




Şekil 2.1 Nöron Modeli

Şekilde verilen nöronun transfer fonksiyonu:

$$y_i = f\left(\sum_{j=1}^{n}(x_{ij}w_{ij}) - w_{i0}\right) \qquad (2.1)$$

Bu denklemde i indisi nöronu tanımlarken j indisi ise diğer nöronlardan gelen girdileri tanımlamaktadır.

## 2.3 Ağ Yapıları

Nöronları büyük ağlara bağlamanın çok değişik yolları vardır. Bu değişik bağlama yöntemlerine mimari ya da devre yapıları denilmektedir. Böylece elde edilecek ağlarla tek bir nöron için imkânsız olan karmaşık işler gerçekleştirilebilir.

Kabaca yapay sinir ağlarında iki önemli mimari vardır:

i. İleri sürümlü ağlar
ii. Geri beslemeli ağlar

İleri sürümlü ağ yapısında yapay nöronlar ileri sürümlü tavırda yapılandırılmıştır, yani her nöron girdisini ya dışarıdan ya da diğer nöronlardan alır ama bir geribesleme yapısı yoktur. Standart bir ileri sürümlü ağ bir giriş kümesi için çıktı kümesi oluşturur ve eğitim bittiği zaman (bağlantı ağırlıkları sabitlendiğinde) daha önceki ağ aktivitesine bakmaksızın aynı girdi için vereceği çıktı her zaman aynı olacaktır. Bu demek oluyor ki ileri sürümlü ağlar dinamik ağ aktivitesi sergilemezler ve bu ağlarda kararlılık problemi yoktur. İleri sürümlü ağlar için dinamiklik genellikle doğrusalsız fonksiyon olarak basitleştirilmiştir.

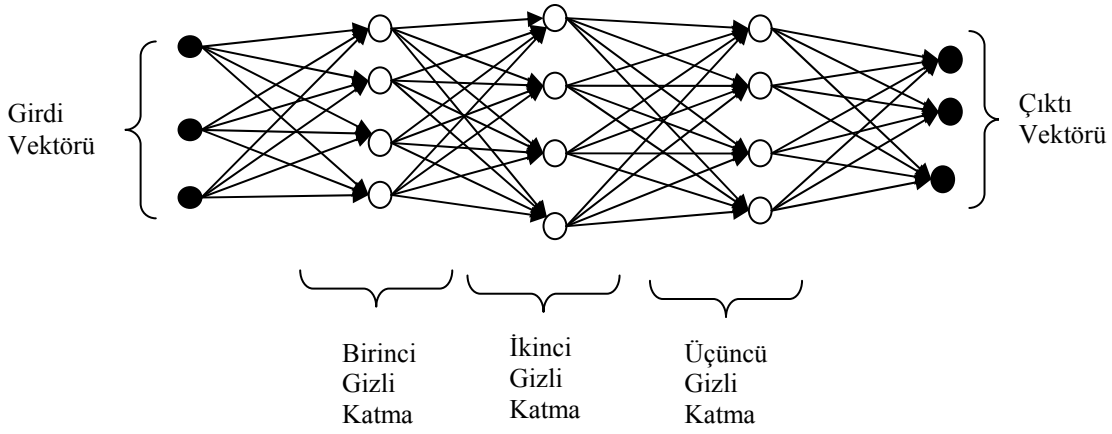
Diğer yönden geribeslemeli ağlarda dinamiklik önemsiz değildir çünkü yapılarında dinamik yapı blokları bulunan işleme elemanları içerirler (integratörler veya gecikme elemanları) geribesleme biçiminde çalışırlar. Bu gibi ağların dinamik özellikleri alışılmış diferansiyel ve fark denklemeleri ile ifade edilen sistemlerle tanımlanır. İleri sürümlü ağlar statik doğrusalsız eşleşmeler ile tanımlanırken geribeslemeli sinir ağları dinamik doğrusal olmayan sistemlerle ifade edilmektedirler.

Ağ yapıları sadece bu belirtilen yapılar değildir. Fakat bütün ağ yapıları dinamik sistemlerin modellenmesinde ve kontrolüne uygun yapıda değildir (Norgaard *et al* 2000).

### 2.3.1 İleri sürümlü (Çok katmanlı) yapay sinir ağları

İleri sürümlü yapısında nöronlar katmanlar halinde yerleştirilir. Bu bağlantı yapısında bir katmandaki nöronlar bitişik katmandaki nöronlarla tek yönlü olarak bağlantılıdır. Şekil 2.2.'de üç katmanlı bir yapı verilmiştir. Nöronlar sıralanmış katmanlar içinde gruplandırılmıştır, katmanlar 0, 1, 2, 3 numaraları ile numaralandırılmıştır. 0. katmanda (giriş katmanı) bulunan nöronlar işlem yapmazlar sadece gelen girdileri 1 numaralı katmanın nöronlarına iletirler. 1 numaralı katman birinci gizli katman olarak adlandırılır. Ağın çıktısının alındığı katman yani en son katman olan 3. katman çıktı katmanı olarak adlandırılır. Girdi ve çıktı katmanı arasında yer alan katmanlara gizli katmanlar adı verilir.

Rosenblatt'ın katmanların sayımı sırasında girdi katmanını da sayılması geleneğine göre yazan yazarlar şekil 2.2.'deki yapıyı 4 katmanlı olarak değerlendirmektedirler (Cichoki *et al* 1993).




Şekil 2.2 Üç Katmanlı Ağ Modeli

Giriş ve çıkış katmanlarında nöron sayıları ele alınan problemin gereklerine göre belirlenir, fakat gizli katman (ya da katmanlardaki) nöron sayısının optimallik anlamında doğru sayısını veren herhangi bir analitik yöntem şu ana kadar geliştirilememiştir. Dolayısıyla gizli katman sayısındaki ve bu katmanlardaki nöron sayılarındaki belirsizlikleri aşabilmenin tek yolu deneme yanılma yöntemidir. (Efe vd. 2004)

İleri sürümlü bağlantı yapısında aynı katmanda bulunan nöronlar arasında ve bir üst katmanda bulunan nörondan bir alt katmana bağlantı yapılmamaktadır. Genellikle her katmanda farklı sayıda nöron ve farklı sinaptik ağırlık bulunur. Her ağdaki her nöron bir çıktı ve bir önceki katmanın çıktıları olan bir dizi girdi ile karakterize edilir.
$R_i$ s katmanında yer alan j nci nöronun net toplam çıktısı ise

$$R_i^{[s]} = \sum_{j=1}^{n_{s-1}} w_{ij}^{[s]} y_j^{[s-1]} + b_i^{[s]} \quad (s=1, 2, 3\ i=1,2,\ldots n_s) \quad (2.2)$$

Eşitlik 2.2.'de $w_{ij}^{[s]}$ j nci nöronun girdilerini ağırlıklandıran sinaptik ağırlıklar ve $n_s$ s inci katmandaki nöron sayısıdır.

Bir önceki katmanın çıktısı sonraki katmanın girdisidir, yani;
$x_i^{[s]} = y_i^{[s-1]}$, $y_i^{[0]} = x_i$, $y_i^{[3]} = y_i$, dir.

Nöronun çıktısı ağırlıklandırılmış çıktının doğrusal olmayan aktivasyon fonksiyonundan geçirilmesi ile elde edilir (Norgaard *et al* 2000).

$$y_i^{[s]} = f_i^{[s]}(R_i^{[s]}) = f_i^{[s]}\left[\sum_{j=1}^{n_{s-1}} w_{ij}^{[s]} y_i^{[s-1]} + b_i^{[s]}\right] \quad (2.3.)$$

Bu eşitliklerde $b_i^{[s]}$ eşik değeridir. Bazı kaynaklarda eşik değeri nörona sürekli 1 girdisini bağlayan sinaptik ağırlık olarak ele alınmaktadır yani $w_{i0}^{[s]} = b_i^{[s]}$ ( Freeman *et al* 1991,Kartalopoulos 1996). Yani denklem 2.3. denklem 2.4 olarak da ifade edilmektedir.

$$y_i^{[s]} = f_i^{[s]}(R_i^{[s]}) = f_i^{[s]}\left[\sum_{j=0}^{n_{s-1}} w_{ij}^{[s]} y_i^{[s-1]}\right] \quad (2.4.)$$

Sistemin çıktısını matris formunda ele aldığımız zaman eşitlik (Hagan *et al* 1996):

$$\mathbf{y}^3 = \mathbf{f}^3(\mathbf{W}^3\,\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{x} + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3) \quad (2.5.)$$

formunu almaktadır.
$\mathbf{y}^3$ : Ağın çıktı vektörü
**x :** Ağın girdi vektörü
$\mathbf{f}^s$ : s katmanının diagonal doğrusal olmayan operatörü

$\mathbf{b}^s$: s katmanın eşik değer vektörü

$\mathbf{W}^s$: s katmanının ağırlık matrisi

Süreç sırasında sinaptik ağırlıkların sabit olduğu unutulmamalıdır ve bu ağırlıkların değerleri ağın davranışını ve istenilen çıktının elde edilmesindeki yeteneğini belirler. İstenilen ağ davranışını elde etmek için ağırlıkların doğru şekilde hesaplanması gerekir. Bu demek oluyor ki ağ eğitilmelidir.

### 2.3.2 Ağ eğitimi

Yapay sinir ağları için eğitim, genel olarak yapay sinir ağlarının kendini girdiye göre uyarlayarak yani kendinde gerekli parametre ayarlamalarını yaparak sonunda istenilen çıkışı elde etme sürecidir. Eğitim ayrıca girdilerin sürekli sınıflandırma sürecidir yani bir girdi geldiğinde ya ağ onu tanır ya da yeni bir sınıflandırma oluşturur. Gerçek durumda, eğitim sürecinde ağ parametrelerini girdiden elde edilen çıktıya göre ayarlar ve sonuçta gerçek çıktı beklenen çıktıya yaklaşır. İstenen çıktı beklenen çıktı ile aynı olduğu zaman, eğitim süreci tamamlanır.

Eğitim algoritmaları genel olarak öğreticili öğrenme ve öğreticisiz öğrenme olarak iki sınıfta incelenebilir. Öğreticili öğrenmede eğitim sırasında kullanılacak girdiler ve bu girdilerden elde edilmesi beklenen çıktılar elde mevcuttur. Böylece gerekli parametre düzeltmeleri yapılarak beklenen çıktı ile gerçek çıktının aynı olması sağlanır.
Öğreticisiz öğrenmede ise beklenen çıktı değerleri mevcut değildir. Bu yöntemde ise ağı yönlendirecek çeşitli kılavuzlar kullanılır. (Kartalopoulos 1996)

Efe (2004) "Yapay Sinir Ağları ve Uygulamaları" çalışmasında öğreticili öğrenmeye öğretmenin ders anlatmasını öğreticisiz öğrenmeye ise bisiklet sürme örneklerini vermiştir.

#### 2.3.2.1 Hata geriye yayma eğitim algoritması

Hata geriye yayma algoritması Paul Werbos tarafından bulunmuş olup ve Rumelhart ve Parker tarafından bağımsız olarak tekrar bulunmuştur (Cichoki *et al* 1993,Kartalopoulos 1996).

Ağın eğitim sürecinde, iki numune $X_k$ girdi vektörü ve bu girdi vektöründen elde edilecek $T_k$ istenen çıktılar kullanılır. $X_k$ girdisi her katmandaki nöronda ve sonunda çıktı katmanında $Y_k$ gerçek çıktıyı oluşturur. Çıktı katmanında gerçek çıktı ve istenen çıktı arasındaki fark hata işaretini oluşturur. Bu hata işareti her katmandaki nöronun ağırlık değerlerine bağlıdır. Bu hata en aza indirilir ve bu süreç boyunca ağırlıkların yeni değerleri elde edilir. Eğitim sürecinin hızı ve doğruluğu aynı zamanda "öğrenme katsayısı" parametresine dayanır.

Hata geriye yayma algoritması ile eğitim sürecine başlamadan önce bunlara ihtiyacımız vardır:

- Eğitim numunelerine, girdilerine ve hedef değerlerine
- Öğrenme katsayısı değerine
- Algoritmayı durduracak kritere
- Ağırlıkları güncellemek için bir metoda
- Aktivasyon fonksiyonuna
- İlk ağırlık değerlerine

Süreç ilk girdi ve beklenen çıktı çiftinin ele alınmasıyla başlar. Bu girdiden çıktı katmanında gerçek çıktı elde edilir. Elde edilen gerçek çıktı beklenen çıktı ile karşılaştırılır ve fark (hata işareti) hesaplanır. Çıkış nöronlarındaki hata farkından, algoritma nöronun aktivasyon düzeyindeki değişimin hatayı hangi oranda değiştirdiğini hesaplar. Algoritma bu aşamaya kadar ileri yönde ilerlemiştir bu aşamadan sonra geriye doğru ilerleyerek bir önceki katmana yani en son gizli katmana geçer ve çıkış katmanı ile en son gizli katman arasındaki ağırlıklarını tekrar hesaplar. Algoritma en son gizli katmanın hata çıktısını hesaplar ve en son gizli katman ile bir önceki katman arasındaki ağırlıkları günceller. Algoritma bu şekilde hata çıktısını hesaplayarak ve ağırlıkları güncelleyerek giriş katmanına doğru ilerler. Giriş katmanına ulaştığı zaman ağırlıklar

değişmez ve yeni girdi ve hedef çıktısı ele alınarak süreç tekrarlanır. (Kartalopoulos 1996)

Hata geriye yayma yönteminin temel felsefesi eğim düşümü yöntemine, yani 2.6.'da verilen tek parametreli ($\Phi$) bir maliyet fonksiyonunun en küçük değerini aldığı noktanın 2.7. bağıntısıyla verilen kural ile iteratif olarak bulunabilmesine dayandırılır. Her bir iterasyonda fonksiyonun minimum noktasına daha çok yaklaşır.

$$J_r = \frac{1}{2}\Phi^2 \tag{2.6.}$$

$$\Delta\Phi = -\eta \frac{\partial J_r}{\partial \Phi} \tag{2.7.}$$

Öğrenme kat sayısı ya da adım büyüklüğü olarak da tanımlanan $\eta$ değeri önemlidir. $\eta$ değişkeni eğer çok büyük ise orijin etrafında salınımlara hatta ıraksamaya sebep olurken, küçük bir değer ise hata uzunca bir sürede orijine yakınsayabilir. Birçok uygulamada adım büyüklüğü deneme yanılma yöntemi ile bulunurken bazı çalışmalarda bu parametre de uyarlanır olarak değerlendirilir.

Hata geriye yayma yönteminde denklem 2.8'de verilen $J_r$ maliyet fonksiyonu minimize edilmelidir. 2.9'da verilen ağırlık güncelleme formülü benimsenecektir. Yöntemin türetimi çıkış katmanı için farklı gizli katmanlar için farklı formülasyon ortaya çıkarır. Çıkış katmanın türetimi için s+1 inci katman olan ve $n_{s+1}$ sayıda nöron içeren çıkış katmanı ele alınsın. Değişkenler aşağıdaki gibi olsun:

$J_r$ : Maliyet fonksiyonu

$t_i$: Ağın i'inci çıkışı için istenen çıkış değerleri

$y_i^{s+1}$ : s+1 inci katmandaki i nöronundan elde edilen gerçek çıkış

$w_{ij}^{s}$ : s+1 inci katmandaki i'inci nöron ile s'inci katmandaki j'inci nöronu birleştiren ağırlık

$R_i^{k+1}$ : s+1 inci katmandaki nöronun girişinde oluşan net toplam

$$J_r = \frac{1}{2}\sum (t_i - y_i)^2 \tag{2.8.}$$

$$\Delta w = -\eta \nabla_w J_r \tag{2.9.}$$

$\nabla_w$ sembolü w parametresine göre kısmi türevi göstermektedir. 2.9 denkleminde verilen türev zincir kuralı yardımıyla 2.10. denkleminde verildiği gibi açılabilir. Bu çarpanların açılımı 2.11 – 2.13 denklemlerinde verilmiştir.

$$\frac{\partial J_r}{\partial w_{ij}^s} = \frac{\partial J_r}{\partial y_i^{s+1}} \frac{\partial y_i^{s+1}}{\partial R_i^{s+1}} \frac{\partial R_i^{s+1}}{\partial w_{ij}^s} \tag{2.10.}$$

$$\frac{\partial J_r}{\partial y_i^{s+1}} = -(t_i - y_i^{s+1}) \tag{2.11.}$$

$$\frac{\partial y_i^{s+1}}{\partial R_i^{s+1}} = \frac{df(R_i^{s+1})}{dR_i^{s+1}} = f'(R_i^{s+1}) \tag{2.12.}$$

$$\frac{\partial R_i^{s+1}}{\partial w_{ij}^s} = \frac{\partial}{\partial w_{ij}^s}\left[\sum_{j=1}^{n_s} w_{ij}^s y_j^s\right] = y_j^s \tag{2.13.}$$

Delta değeri 2.14'teki gibi tanımlanırsa çıkış katmanındaki nöronlar için delta değerinin genel hali 2.15'te verilen biçimde ve ağırlıklardaki değişim miktarı ise 2.16'da verilen biçimde olacaktır.

$$\delta_i^{s+1} = -\frac{\partial J_r}{\partial R_i^{s+1}} \tag{2.14.}$$

$$\delta_i^{s+1} = (t_i - y_i^{s+1}) f'(R_i^{s+1}) \tag{2.15.}$$

$$\Delta w_{ij}^s = \eta \delta_i^{s+1} y_j^s \tag{2.16.}$$

Gizli katmanlardaki nöronların parametrelerini güncellenmesinde 2.10 denkleminde yer alan kısmi türevi oluşturan terimler değişik yollardan gelebilirler. Bu durum 2.20 denklemindeki zincir kuralının ilk terimi olan 2.17 denkleminde de görülmektedir. Aynı terimin daha açık ifadeleri 2.18 ve 2.19'de de verilmiştir.

$$\frac{\partial J_r}{\partial w_{ij}^s} = \frac{\partial J_r}{\partial y_i^{s+1}} \frac{\partial y_i^{s+1}}{\partial R_i^{s+1}} \frac{\partial R_i^{s+1}}{\partial w_{ij}^s} \tag{2.10.}$$

$$\frac{\partial J_r}{\partial y_i^{s+1}} = \sum_{h=1}^{n_{s+2}} \frac{\partial J_r}{\partial R_h^{s+2}} \frac{\partial R_h^{s+2}}{\partial y_i^{s+1}} \tag{2.17.}$$

$$\frac{\partial J_r}{\partial y_i^{s+1}} = \sum_{h=1}^{n_{s+2}} \left[ \frac{\partial J_r}{\partial R_h^{s+2}} \frac{\partial}{\partial y_i^{s+1}} \left( \sum_{i=1}^{n_{s+1}} w_{hi}^{s+1} y^{s+1} \right) \right] \tag{2.18.}$$

$$\frac{\partial J_r}{\partial y_i^{s+1}} = \sum_{h=1}^{n_{s+2}} \frac{\partial J_r}{\partial R_h^{s+2}} w_{hi}^{s+1} \tag{2.19.}$$

Gizli katman için delta değeri 1.18'deki gibi tanımlanabilir.

$$\delta_i^{s+1} = -\frac{\partial J_r}{\partial R_i^{s+1}} \tag{2.20.}$$

Delta tanımının kullanılması ile 2.10. denkleminin ilk terimi 2.21'de gösterilen biçimde yazılabilir. 2.10 denkleminin ikinci ve üçüncü terimleri 2.22 ve 2.23'te daha açık ifade edilmiştir. Elde edilen terimler kullanılarak 2.24'deki delta değerine ve 2.25'teki parametre güncelleme kuralına ulaşılır.

$$\frac{\partial J_r}{\partial y_i^{s+1}} = -\sum_{h=1}^{n_{s+2}} \delta_h^{s+2} w_{hi}^{s+1} \tag{2.21.}$$

$$\frac{\partial y_i^{s+1}}{\partial R_i^{s+1}} = \frac{df(R_i^{s+1})}{dR_i^{s+1}} = f'(R_i^{s+1}) \tag{2.22.}$$

$$\frac{\partial R_i^{s+1}}{\partial w_{ij}^s} = y_j^s \tag{2.23.}$$

$$\delta_i^{s+1} = \left( -\sum_{h=1}^{n_{s+2}} \delta_h^{s+2} w_{hi}^{s+1} \right) f'(R_i^{s+1}) \tag{2.24.}$$

$$\Delta w_{ij}^s = \eta \delta_i^{s+1} y_j^s \tag{2.25.}$$

# 3. YAPAY SİNİR AĞLARININ SÜREÇ DENETİMİ UYGULAMALARI

Yapay sinir ağlarıyla yapılan süreç denetimi çalışmaları son yıllarda dikkat çekmektedir. Bunun sebebini yapay sinir ağlarının sahip oldukları özelliklere bağlamak mümkündür. Yapay sinir ağlarının temel özellikleri aşağıdaki gibidir:

- Doğrusalsız sistemler: Kavramsal olarak sahip oldukları doğrusalsız eşleşmelere yakınlaşabilme yetenekleri, doğrusalsız denetim problemlerinde umut verici çözüm olarak ele alınmalarını sağlar (Hunt *et al* 1992, Efe vd. 2004, www.statsoftinc.com 2006).
- Yapay sinir ağları bir özelliği paralelliğidir, toplamsal işlevin yapısal dağılmışlığıdır. Yani yapay sinir ağlarında bulunan nöronlar eş zamanlı olarak paralel çalışırlar. Böyle bir yapının diğer geleneksel yöntemlere göre daha yüksek derecede hata toleransı gerçekleştirebileceği umudunu doğurur. Yapay sinir ağındaki temel işleme elamanı yapı olarak çok basittir fakat paralel yapı ile bir araya geldiğinde çok hızlı bir işleme doğurur (Freeman *et al* 1991, Hunt *et al* 1992, Kartalopoulos *et al* 1996, Efe vd. 2004).
- Yapay sinir ağlarının her hangi bir doğrusalsız fonksiyona daha önceden belirlenmiş doğrulukla yaklaşabildikleri görülmüştür (Noirega *et al* 1998).
- Yapay sinir ağları örneklerle öğrenirler. Sistemi tanımlayacak uygun girdi çıktı çiftleri yardımıyla öğrenirler (Norgaard *et al* 2000, www.statsoftinc.com 2006).
- Genelleme yetenekleri vardır (Efe vd. 2004). Eğitim esnasında girdiler verilerek uygun çıktılar elde edilmesi sağlanır. Eğitim sırasında verilmeyen bir girdi geldiği zaman anlamlı çıktı üretebilirler.
- Kendilerini değişen ortama uyarlayabilirler. Bu özellik onların, zamanla değişen davranışları olan doğrusal olmayan sistemlerle ve bu sistemleri verimli bir şekilde denetlemek için kullanılan uyarlanır denetleyicilerle rekabet edebilmesini sağlar.
- Ek olarak, taklit edilerek tasarlandıkları biyolojik emsallerinde bulunan özellikler gibi gürültüye karşı sağlamlık ve hata toleransı için büyük potansiyel gösterirler.

- Donanım uygulaması: Bir sürü üretici yakın zamanda VLSI uygulaması üretmiştir. Bu da fazladan hızı ve yapılabilecek ağların büyüklüğünü arttırmaktadır.
- Veri tümleştirme: Yapay sinir ağları eş zamanlı olarak hem nicel hem de nitel verilerlerle işlem yapabilirler. Bu açıdan bakıldığında yapay sinir ağları geleneksel mühendislik sistemleri (nicel veri) ile yapay zeka alanındaki işleme teknikleri (sembolik veri) arasında bir yerdedir (Hunt *et al* 1992).
- Çok verili sistemler. Yapay sinir ağları doğal olarak çok girdi işlerler ve çok çıktıları vardır; çok değişkenli sistemlerde uygulanmaya hazırdırlar. (Hunt *et al* 1992).

## 3.1 Tanılama

Tanılama, bir sistemin hangi şartlar altında nasıl davranacağını belirleyen modelleri kurmaktır. Efe (2004) sistem tanılamasını, "Girişleri ve çıkışları gözlenebilen bir sistemin hangi şatlar altında nasıl davranacağını belirleyen modelleri kurmak ve bu modellerin değişken şartlar altında da geçerli modeller olduğunu göstermek" olarak tanımlanmaktadır.

Tanılama, sistemin giriş ve çıkışların gözlenerek gerçekleştirilir. Yapay sinir ağları ile sistem tanılaması eş zamanlı yapılacağı gibi zamandan bağımsız olarak da gerçekleştirilebilir. Zamandan bağımsız tanılamada sistemin bazı girdi ve çıktıları elde mevcuttur ve bu girdi çıktı çiftleri kullanılarak yapay sinir ağı sistemi modelleyecek şekilde eğitilir. Eş zamanlı tanılama yapısında sistem ve model aynı girdiyi almaktadır ve sistemin ve modelin ürettiği çıktılarının farkından elde edilen hata işareti minimize edilmeye çalışılır.

Doğrusalsız sistemlerin tanılamasında ya geri beslemeli yapay sinir ağları kullanılır ya da gecikme ile geri besleme uygulanmış standart çok katmanlı ağlar ile gerçekleştirilebilir (Hunt *et al* 1992).
Denklem (3.1)'de zamanda ayrık fark denklemi ile ifade edilen doğrusalsız sistemi ele alalım:

$$y(t+1) = f[y(t),...,y(t-n+1);u(t),...,u(t-m+1)] \quad (3.1.)$$

Bu denklemde belirtildiği gibi sistemin t+1 zamanındaki çıktısı doğrusalsız f fonksiyonu bağlantısı ile n tane geçmişteki y çıkış değerine ve m tane geçmişteki u giriş değerine bağlıdır. Sistem tanılamadaki en belli yöntem giriş çıkış yapısı sistem ile aynı olan yapay sinir ağı seçmektir. Yapay sinir ağı modelinin çıktısını $y^m$ ile ifade edilirse

$$y^m(t+1) = f^m[y(t),...,y(t-n+1);u(t),...,u(t-m+1)] \quad (3.2.)$$

Burada $f^m$ yapay sinir ağının doğrusalsız fonksiyonudur yani f'e yaklaşımdır. Bu denklemde yapay sinir ağının girdileri gerçek sistemin çıktılarına bağlıdır. Gerekli eğitim periyodundan sonra $y^m \approx y$ yaklaşımını kullanabiliriz. Bu varsayımdan sonra model sistemden bağımsız hale gelir ve denklem:

$$y^m(t+1) = f^m[y^m(t),...,y^m(t-n+1);u(t),...,u(t-m+1)] \quad (3.3.)$$

Yapay sinir ağları ile kullanılan çeşitli model yapıları bulunmaktadır. Örneğin, NNFIR(Finite Input Response),NNARX(AutoRegressive eXternal input) NNARMAX(AutoRegressive Moving Average eXternal input), NNOE(Output Error), NNSIF(State Space Innovations Form) gibi. İsimlerin başlarında yer alan "NN" harfleri yapay sinir ağları kullanılarak uygulandığını göstermektedir. Öngörülü denetim uygulaması sırasında kullanılan model yapay sinir ağıyla uygulanmış ARX yapısı olduğu için aşağıda yapay sinir ağları için ARX yapısı ele alınmıştır.

### 3.1.1 ARX model yapısı

Gerçek sistem (2.4.) denklemi ile ifade edilsin. (Norgaard et al 2000)

$$y(t) = G_0(q^{-1}) + H_0(q^{-1})e_0(t) \quad (3.4.)$$

$e_0(t)$, u(t) giriş işaretinden bağımsız beyaz gürültü. G ve H $q^{-1}$ zaman gecikme operatöründe transfer fonksiyonu. $q^{-1}$ zaman gecikme operatörü aşağıdaki gibi çalışır.

$$q^{-d}x(t) = x(t-d) \tag{3.5.}$$

Model yapısı:

$$M : \left\{G(q^{-1},\vartheta), H(q^{-1},\vartheta) \middle| \vartheta \in D_m \right\}$$
$$y(t) = G(q^{-1},\vartheta)u(t) + H(q^{-1},\vartheta)e(t) \tag{3.6.}$$

$\theta$ ayarlanabilir parametreleri belirtmektedir. Kestirimci model yapısı aşağıdaki gibidir.

$$\hat{y}(t|t-1,\vartheta) = H^{-1}(q^{-1},\vartheta)G(q^{-1},\vartheta)u(t) + \left[1 - H^{-1}(q^{-1},\vartheta)\right]y(t) \tag{3.7.}$$

Yukarıda bir adım ileri kestirimler ele alınmıştır. Model yapısı genellikle (3.8.)'deki gibi alternatif yapıda yazılır.

$$\hat{y}(t|\vartheta) = \varphi^T(t)\vartheta \tag{3.8.}$$

$\theta$ parametre vektörü $\varphi$ regresyon vektörü.
ARX(AutoRegresive, eXternal input) yapısında ise:

$$G(q^{-1},\vartheta) = q^{-d}\frac{B(q^{-1})}{A(q^{-1})} \qquad H(q^{-1},\vartheta) = \frac{1}{A(q^{-1})} \tag{3.9.}$$

kestirimci ise

$$\begin{aligned}\hat{y}(t|\vartheta) &= q^{-d}B(q^{-1})u(t) + \left(1 - A(q^{-1})\right)y(t) \\ &= \varphi^T(t)\vartheta\end{aligned} \tag{3.10.}$$

ve

$$\begin{aligned}\varphi(t) &= [y(t-1)...y(t-n), u(t-d)...u(t-d-m)]^T \\ \vartheta &= \left[-a_1,...-a_n, b_0,...,b_m\right]\end{aligned} \tag{3.11.}$$

## 3.2 Süreç Denetimi

Süreç denetimi bir sürecin istenildiği gibi davranılmasını sağlamaktır. Yapay sinir ağları süreç denetim uygulamalarında dolaylı uygulamalar ve direk uygulamalar olmak üzere iki türlü kullanılmaktadır. Dolaylı uygulamalarda sistemin tanılayıcı modelinin oluşturulmasında ve direk uygulamalarda ise denetleyicilerde kullanılmaktadır. Süreç denetimi uygulamalarında kullanılan örnek birkaç yapı aşağıda özetlenmiştir.

### 3.2.1 Direk ters denetim

Direk ters denetim yapısında yapay sinir ağı sistemde denetleyici olarak kullanılmaktadır.

Direk ters denetim uygulamalarında sistemin ters modelinden faydalanılır. Beklenen cevap(Yapay sinir ağı girdisi) ile denetlenen sistemin cevabının aynı olması için ters model ile sistem arka arkaya bağlanır. Yapay sinir ağı böyle yapılarda denetçi olarak davranır(Hunt *et al* 1992).

Denetlenecek sistem aşağıdaki eşitlik ile ifade edilsin

$$y(t+1) = g\left[y(t),...,y(t-n+1),u(t),...,u(t-m)\right] \tag{3.14}$$

Yapay sinir ağının oluşturacağı denetim işareti

$$\hat{u}(t) = \hat{g}^{-1}\left[y(t+1),y(t),...,y(t-n+1),u(t),...,u(t-m)\right] \tag{3.15.}$$

Sistemin denkleminde t+1'deki istenen çıktı yani r(t+1)'i denkleme sokulabilir. Yapay sinir ağı gerçek ters model ise t+1'deki çıkış işareti r(t+1) olacaktır.

Efe vd. (2004) ve Freeman (1991) bu yaklaşımın yararına iki kısıtlama olarak; bu teknik çok girdili çıktılı genelde çoktan teke eşleşmelerin olduğu durumlarda sistemlerin tersinin bulunmasının mümkün olmadığını kanıtlayabilir. Diğer yandan bu metot sadece

ters modellerin eğitiminde kullanılabilir. Ve beklenen ve gerçek değerin arasındaki hatadan ziyade çıkışı değerlendirmeyi kapsayan eğitimler için genel formülasyona izin vermez ( Lazar *et al* 2002, Efe vd. 2004).

Denetlenecek sistem bire bir değil ise tek ters model yoktur Yani

$$g[y(t),...,y(t-n+1),u_1(t),...,u(t-m)] = g[y(t),...,y(t-n+1),u_2(t),...,u(t-m)] \quad (3.17.)$$

Yukarıdaki eşitlikteki durum iki farklı denetim işareti için oluşabilir.$u_1(t) \neq u_2(t)$. Bu benzersizlik olmayışı eğitimde dikkate alınmaz ise prensipte sistemin denetimi için yeterli olmayan bir ters model elde edilebilir (Norgaard *et al* 2000).

Referanstan sistemin gerçek çıktısına kadar olan kapalı devre transfer fonksiyonu:

$$T(q^{-1}) = q^{-1} \quad (3.18.)$$

Denetleyici, sistemi doğrusallaştırır elde edilen sonuç sistemin çıkışının referansı bir örnekleme periyodu gecikme ile takip etmesidir. Sonuçta sistemin transfer fonksiyonu: $T(q^{-1}) = q^{-d}$ olur.

Deneyle ya da eşzamanlı olarak elde edilen girdi çıktı çiftleri kullanılarak yapay sinir ağı eğitilir. Yapay sinir ağı aşağıda verilen gibi bir maliyet fonksiyonunu minimize etmek üzere eğitilir.

$$J(\theta, Z^N) = \frac{1}{2N}\sum_{i=1}^{N}[u(t) - \hat{u}(t|\theta)]^2 \quad (3.16)$$

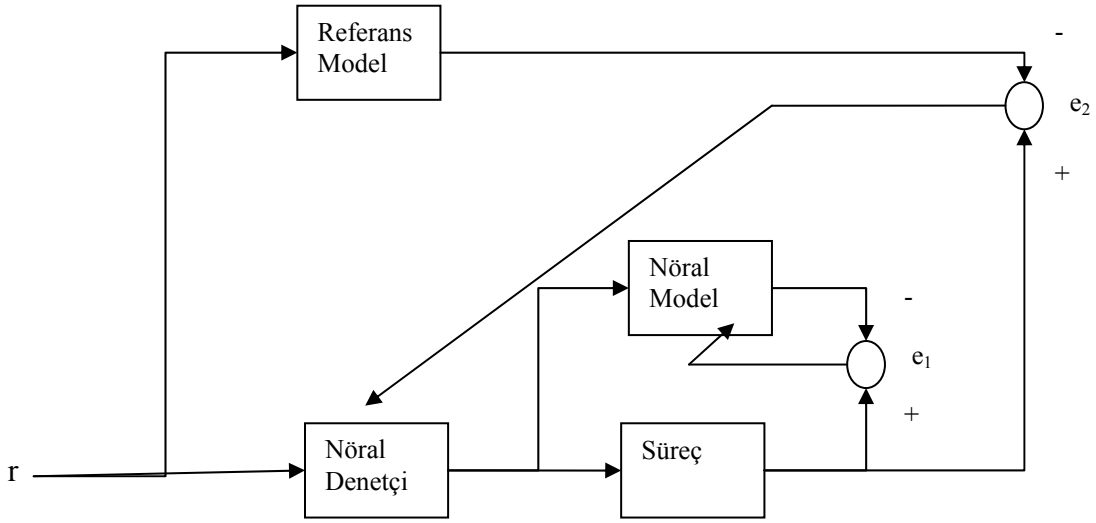
Direk ters modelleme uygulamasında nöral denetçi uyarlanır olabileceği gibi uyarlanmaz yapıda da olabilmektedir. Uyarlamalı olmadığı durumlarda performansının kalitesi denetçinin tasarımında kullanılan modelin doğruluğuna dayanır. Model çok doğru ve sistemdeki bozan etken az ise kullanılabilir (Rivals *et al* 2000).

### 3.2.2 Model dayanaklı denetim

Bu yapıda kapalı devre sistemin performansı girdi çıktı çifti {r(t), $y^r(t)$} tarafından tanımlanan kararlı bir model üzerinden yapılmaktadır. Kontrol sistemi, süreç çıktısı $y_p$'yi sonuşur bir biçimde referans modelin çıktısı ile eşleştirmeye çalışır. Yani

$$\lim_{t\to\infty}\left\|y^r(t)-y^p(t)\right\|\le\varepsilon \qquad \varepsilon\ge 0 \text{ sabiti için} \qquad (3.19.)$$

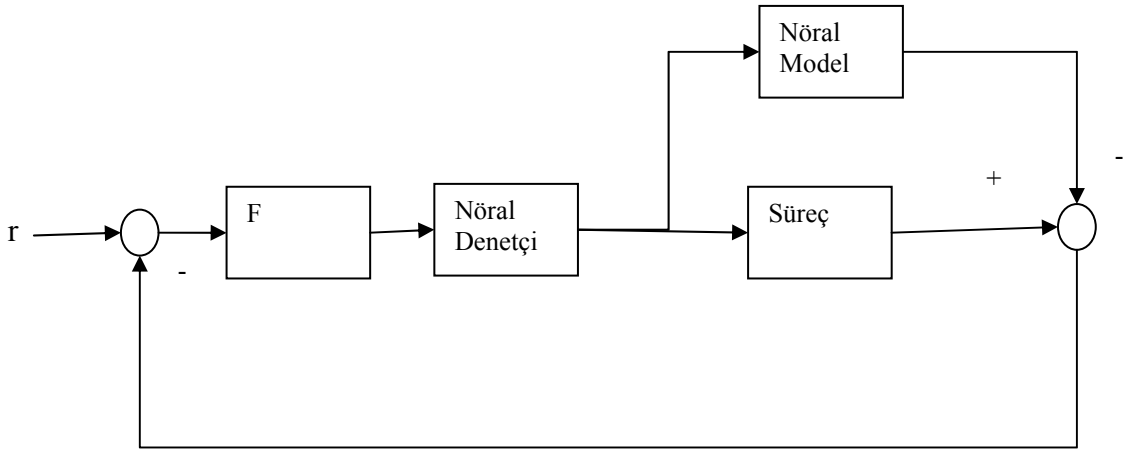
Model referans kontrol yapısı şekil 3.1'de verilmiştir. Sürecin istenilen davranışı harici olarak referans model tarafından üretilir. Bu referans model seçilirken, kontrol edilen sürecin davranışını kabul edilebilir seviyede göstermesine dikkat edilmelidir. Şekildeki yapıda iki adet yapay sinir ağı bloğu bulunur, nöral model ve nöral denetçi. Nöral modeldeki yapay sinir ağları, çıkış değerinin öngörülen ve gerçek değeri arasındaki hatayı ($e_1$) geriye yayarak ağırlıklarını ayarlarlar. Denetçideki yapay sinir ağları ise referans modelin çıkışı ile gerçek çıkış arasındaki hatayı ($e_2$) geriye yayarak ağırlıklarını ayarlarlar. Nöral model ile nöral denetçi arasındaki çapraz çizgi nöral modeldeki $e_2$ hatasının geriye yayılımını temsil eder (Cembrano *et al* 1992).




Şekil 3.1 Model Dayanaklı Kontrol Yapısı

### 3.2.3 İçsel model denetimi

İçsel model denetim yapısı şekil 3.2.'de verilmiştir.




Şekil 3.2 İçsel Model Denetim Yapısı

İçsel model denetim yapısında sistemin ileri ve ters modelleri geri besleme devresinde direk eleman olarak kullanılmaktadır. Yapıda model sisteme paralel olarak yerleştirilmekte ve model ve sistem arasındaki fark geri beslemede kullanılmaktadır. Bu geri besleme ileri yoldaki alt sistemde bulunan ve İçsel model denetimde sistemin tersine bağlı olduğu dikte edilen denetçide işlenir. F alt sistemi ise istenilen güçlü ve kapalı devre sisteminin cevabını izlenmesini sağlamak için tasarlanan genellikle doğrusal süzgeçtir. İçsel denetim sisteminin uygulamaları açık devre kararlı sistemlere uygulanabilir. Bununla birlikte bu teknik süreç denetiminde geniş uygulama alanı bulmuştur (Hunt *et al* 1992).

Eğer kontrol sistemi kararlıysa, sabit girişler için ofsetsiz denetim garanti edilmişse ve yüksek frekanslarda içsel modelin olası uyumsuzluğuna karşı sağlamlık elde edilmişse içsel model denetim uygulamalarında yaygın tasarım stratejisi, birim kazançlı alçak geçirgen süzgeç ile iç modelin tersini art arda takmaktır (Rivals *et al* 1992)

### 3.2.4 Öngörülü denetim

Öngörülü denetim uygulamalarında sadece anlık kontrol işaretiyle ilgilenilmek ile kalınmaz gelecekte olacak değerler de göz önüne alınır. Öngörülü denetimin başarısı ilk etapta modele yani sistemin gelecekteki cevabının doğru tahmin edilmesine dayanmaktadır. Yapay sinir ağları ile uygulamalarında denetçi formülasyonunda

sistemin yapay sinir ağları modelini öngördüğü çıkış işareti, çıkış işareti ve referans değeri kullanılır.

Öngörülü denetimin temel prensibi sürecin modeli ile gelecek ufkunda denetlenecek değişkenleri tahmin etmek, daha sonra maliyet fonksiyonunu minimize edecek şekilde denetçi çıkışlarını hesaplamak ve sonunda ilk denetim hareketini sürece uygulamaktır. Bu prosedür her örnekleme zamanında sürecin güncellenmiş bilgileri ile tekrarlanır (Wang *et al* 2001).

Yapay sinir ağlarından oluşturulan öngörücü ile uygulanan öngörülü denetim stratejisinin iki katı vardır i) sürecin gelecek çıkışını tahmin etmek ve ii) referans yörünge ve tahmin edilen süreç çıkışı arasındaki farktan oluşan maliyet fonksiyonunu minimize etmektir (Lazar *et al* 2002).

Öngörülü denetim, arkasındaki eşitlik J maliyet fonksiyonunu minimize etmekle başlar. Maliyet fonksiyonu süreçten sürece değişmektedir. Eşitlik 3.20'deki matris formundaki maliyet fonksiyonunu ele alalım.

$$\begin{aligned} J(t,U(t)) &= \left[R(t)-\hat{Y}(t)\right]^T\left[R(t)-\hat{Y}(t)\right]+\rho\tilde{U}(t)\tilde{U}(t) \\ &= E^T(t)E(t)+\rho\tilde{U}(t)\tilde{U}(t) \end{aligned} \qquad (3.20.)$$

Burada,

$$\begin{aligned} R(t) &= \left[r(t+N_1)....r(t+N_2)\right]^T \\ \hat{Y}(t) &= \left[\hat{y}(t+N_1|t)...\hat{y}(t+N_2|t)\right]^T \\ E(t) &= \left[e(t+N_1|t)...e(t+N_2|t)\right]^T \\ \tilde{U}(t) &= \left[\Delta u(t)...\Delta u(t+N_u-1)\right] \end{aligned} \qquad (3.21.)$$

Ve

$$e_k(t) = r(t+k) - y(t+k|t) \qquad \text{k=N}_1,\ldots\text{N}_2 \qquad (3.22.)$$

$$\Delta u(k+i-1) = 0 \qquad 1 \le \text{N}_\text{u} < \text{i} \le \text{N}_2 \qquad (3.23.)$$

Bu denklemlerde

$N_1$: En az öngörüm

$N_2$: öngörüm ufku

$N_u$: denetim ufku. $N_u$ değerinden sonra kontrol işareti sabit kalır.

$\rho$: kontroldeki değişiklikleri gösteren ağırlık faktörü

Denetlenecek sürecin gerekirci olması koşuluyla bir adım sonraki öngörü (3.24.)'de verilmiştir.

$$\hat{y}(t) = \hat{y}(t|t-1) = \hat{g}_1\left[y(t-1),...,y(t-n),u(t-d),...,u(t-d-m)\right] \tag{3.24.}$$

g yapay sinir ağı tarafından gerçekleştirilen fonksiyon, d gecikme zamanıdır. Sistemin k adım sonraki çıkışını bulmak için ifade zamanda ileri kaydırılır ve olmayan gerçek çıkışlar için öngörümler konulur. Bu bağlamda tekrar (3.25.)'deki gibi yazılabilir.

$$\begin{aligned}&\hat{y}(t+k) = \hat{y}(t+k|t)\\&= g[\hat{y}(t+k-1),...\hat{y}(t+k-\min[k,n]), y(t-1),..., y(t-\max[n-k,0]),\\&u(t+k-d),...u(t+k-d-m)]\end{aligned} \tag{3.25}$$

Bu eşitlikte çıkış değerinin t-1 zamanına kadar elde olduğu var sayılmaktadır bu nedenle $\hat{y}$ öngörü işareti eşitliğe girmektedir.

Kullanılan g fonksiyonun yapay sinir ağı tarafından gerçekleştirildiği ve bu ağın aktivasyon fonksiyonun bütün katmanlarda aynı f fonksiyonu olduğunu varsayalım. Bu anlamda g' yi açarsak eşitlik (3.26) ve (3.27) elde edilir.

$$\hat{y}(t+k) = \sum_{j=1}^{n_h} W_j f\left(\tilde{h}(h,j)\right) + W_0 \tag{3.26}$$

$$\begin{aligned}\tilde{h}(k,j) = &\sum_{i=1}^{\min(k,n)} w_{j,i}\hat{y}(t+k-i) + \sum_{i=\min(k,n)+1}^{n} w_{j,i}y(t+k-i)\\&+ \sum_{i=0}^{m} w_{j,n+i+1}u(t+k-d-i) + w_{j,0}\end{aligned} \tag{3.27}$$

Kontrol girdilerindeki öngörüler doğrusalsız olduğu zaman genel öngörülü kontrol maliyet fonksiyonunu minimize etmek bir optimizasyon problemidir. Yapay sinir ağları eğitiminde kullanılan özyineli süreç gibi bir süreç uygulanır.

$$U^{(i+1)} = U^{(i)} + \mu^{(i)} f^{(i)} \qquad (3.28.)$$

$U^{(i)}$ : gelecek kontrol girişleri serisindeki şimdiki iterasyon
$\mu^{(i)}$ : adım büyüklüğü
$f^{(i)}$: arama yönü
(Norgaard *et al* 2000)
Tasarım parametrelerinin seçimi aşağıdaki gibi yapılır (Norgaard *et al* 2000):
En az öngörüm $N_1$: Bu parametre genellikle modelin gecikme zamanı d olarak seçilir. Daha küçük seçmek için bir neden yoktur çünkü d-1 ilk öngörümler sadece kontrol işaretinin geçmiş değerlerine dayanır ve bu nedenle etkilenmez. Diğer yönden daha büyük seçmek tavsiye edilmez çünkü bu öngörülmeyen sonuçlara yol açabilir. (Soeterboek, 1992)

Öngörüm ufku $N_2$: Kararlı olmayan bir ters ile sistemlerin kararlılığını sağlamak için en az ağ modeline uygulanan girdi kadar zaman adımı olmalıdır. Genellikle biraz daha uzun seçilir. Kural olarak öngörüm ufku yaklaşık sistemin yükselme süresi (kararlı ise) olarak seçilir. Yine de bu kadar uzun seçmek mümkün değildir çünkü optimizasyon seçilen örnekleme zamanına göre çok çaba gerektirir.

Denetim ufku $N_u$: Doğrusal durumlarda karasız ya da zayıf sönümlenmiş kutup sayısına eşit ya da daha fazla seçilmesi önerilir. Fakat sistemin sadece yapay sinir ağı modelinin bulunduğu gibi sürecin dinamiği hakkında çok az bilgi bulunan durumlarda bu bilgiyi elde etmek çok güçtür. Genel olarak kabul edilebilir kural olarak Soeterboek (1992) basitçe $N_u$= n olarak kabul edilmesini önermektedir. Bununla birlikte, hesaplama yükü $N_u$ 'nun artmasıyla birlikte çok büyümesine rağmen günümüzdeki uygulamalarda daha geniş ufuklar seçilmektedir. Örnekleme periyodu çok büyük değilse uygulanabilir en küçük değerin seçilmesi tavsiye edilir.

Kontrol ağırlık faktörü ρ: Sayısal sağlamlık açısından $\rho > 0$ olarak seçilmelidir. Fakat temel olarak kontrol işaretinin genliğini ve düzlüğünün denetiminde kullanıldığı için uygulamada simülasyon çalışmalarından elde edilebilir.

# 4. SÜREÇ DENETİM UYGULAMALARI

## 4.1 İki Eksenli Doğrudan Sürümlü SCARA Tipi Robot

Uygulamalar sırasında kullanılan şekil 4.1' de gösterilen SCARA tipi iki eksenli robot bilgileri, Denker (1996), Efe (2004) ve Cılız (2005)'ten alınmıştır.

Şekil 4.1 Uygulamalarda Verilerinden Faydalanılan SCARA Tipi İki Eksenli Robot

Kullanılan Robot parametreleri Çizelge 4.1' de verilmiştir.

Çizelge 4.1 Robot Fiziksel Parametreleri

| **Parametre** | **Değer** |
|---|---|
| Motor 1 Rotor Eylemsizliği | 0.267 |
| Kol 1 Eylemsizliği | 0.334 |
| Motor 2 Rotor Eylemsizliği | 0.0075 |
| Motor 2 Stator Eylemsizliği | 0.04 |
| Kol 2 Eylemsizliği | 0.063 |
| Motor 1 Kütlesi | 73 |
| Kol 1 Kütlesi | 9.78 |
| Motor 2 Kütlesi | 14.0 |
| Kol 2 Kütlesi | 4.45 |
| Yük kütlesi | 2 |
| Kol 1 Uzunluğu | 0.359 |
| Kol 2 Uzunluğu | 0.24 |
| Kol 1 Ağırlık Merkezi | 0.136 |
| Kol 2 Ağırlık Merkezi | 0.102 |
| Eksen 1 Sürtünme Sınırı | 5.3 |
| Eksen 2 Sürtünme Sınırı | 1.1 |
| Tork 1 Sınırı | 245 |
| Tork 2 Sınırı | 39.2 |

Robot dinamiği 4.1. denklemi ile ifade edilmektedir. (Efe vd 2004)

$$M(Q)\ddot{Q} + V(Q,\dot{Q}) = \tau - f_c \qquad (4.1.)$$

Bu denklemde

M(Q): eylemsizlik matrisi

V(Q, Q̇): Koriolis terimleri

$\tau$: tork

$f_c$: sürtünme kuvveti

4.1. denkleminde açısal pozisyon ve açısal hızlar durum değişkeni olarak tanımlanırsa sistemin davranışı dört adet diferansiyel denklem ile modellenebilir.

$$M(Q)=\begin{bmatrix} p_1+2p_3\cos(Q_2) & p_2+p_3\cos(Q_2) \\ p_2+p_3\cos(Q_2) & p_2 \end{bmatrix} \tag{4.2.}$$

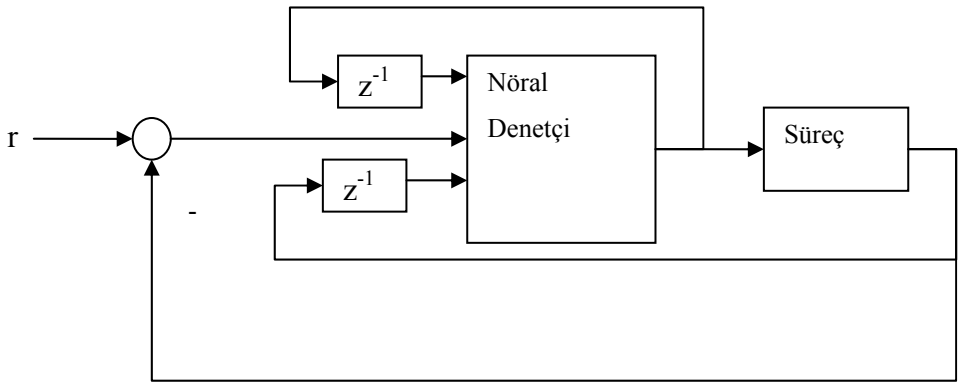
$$V(Q,\dot{Q})=\begin{bmatrix} -Q_2(2\dot{Q}_1+\dot{Q}_2)p_3\sin(Q_2) \\ \dot{Q}_1^2 p_3\sin(Q_2) \end{bmatrix} \tag{4.3.}$$

3.3 denkleminde verilen p parametreleri

$$\begin{aligned} p_1 &= 2.0857+0.0576M_p \\ p_2 &= 0.1168+0.0576M_p \\ p_3 &= 0.1630+0.0862M_p \end{aligned} \tag{4.4.}$$

### 4.2 Direk Ters Modelleme ile Süreç Denetim Uygulaması

Bu uygulamalarda yapay sinir ağı denetçide kullanılmaktadır. Direkt ters modelleme uygulamasında sistemin ters modeli denetçide kullanılmaktadır. Yapay sinir ağının eğitiminde kullanılan robotun giriş tork ve bu torklara karşılık elde edilen açısal hız ve konum verileri Efe (2004)'nin simülasyon çalışmalarından elde edilmiştir. Yapay sinir ağı eğitilirken robotun girdileri yapay sinir ağının çıktıları robotun çıktıları ise yapay sinir ağının girdileri olmuştur. Eğitim zamandan bağımsız olarak eldeki verilerle yapılmıştır fakat yapay sinir ağı uyarlamalı olarak tasarlanmıştır yani uygulama süresince önceki giriş çıkış değerleri kullanılarak yapay sinir ağı uyarlanmıştır.. Uygulamada kullanılan blok şeması şekil 4.2'de verilmiştir.



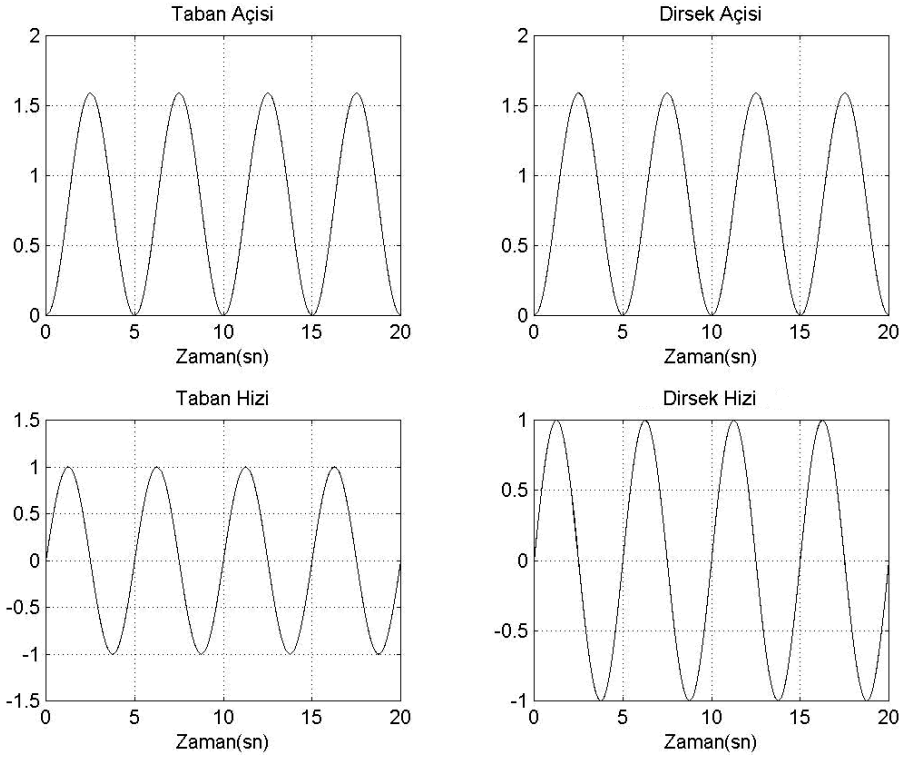

Şekil 4.2 Direk Ters Model Uygulaması Blok Şeması

Bu yapıda denetçinin girdileri hata işareti (yani referans konum ve hız değerleri ile gerçek konum ve hız değerleri arasındaki fark), geciktirilmiş tork değerleri ve geciktirilmiş gerçek konum ve hız değerleridir. Çıktısı ise tork değerlerdir.

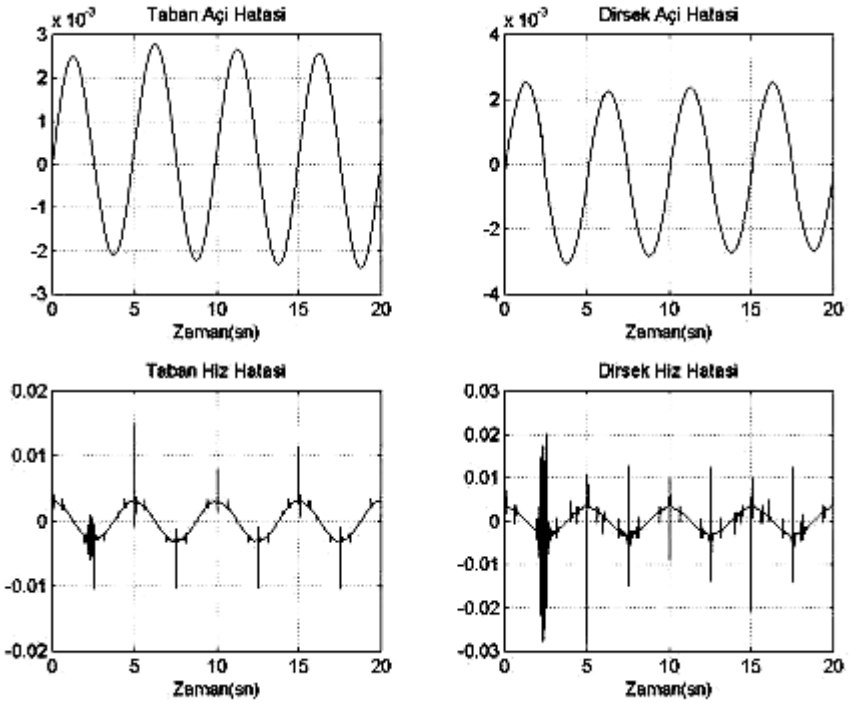
Yapay sinir ağının girdilerinden olan hata işareti tork değerlerinin hesaplamasında kullanılmaktadır. Geciktirilmiş tork ve bu tork değerine karşılık robotun ürettiği konum ile hız değerleri kullanılarak denetçi sürekli uyarlanmaktadır.

### 4.2.1 Sinüzoidal referans hız ile süreç denetim uygulaması

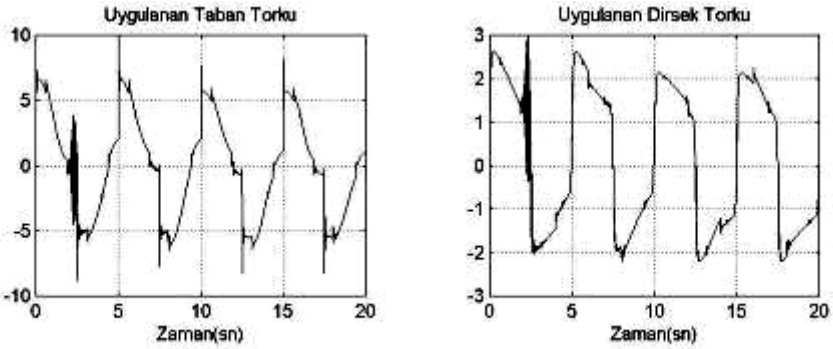
Simülasyon Sonucunda Aşağıdaki grafikler elde edilmiştir:




Şekil 4.3 Direk Ters Modelleme İle Sistemin Çıkış ve Referans Değerleri

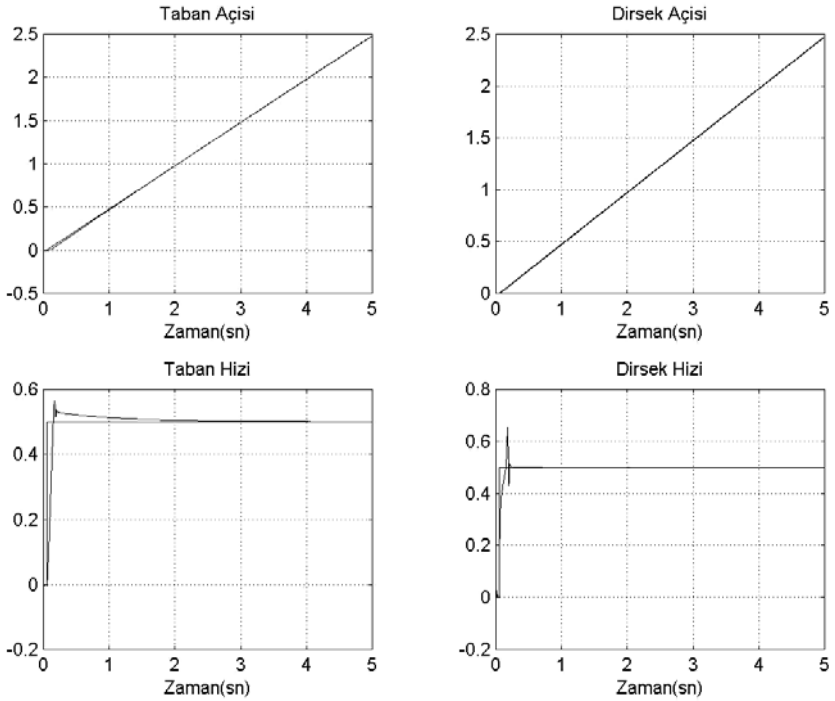



Şekil 4.4 Direk Ters Modelleme İle Sistemin Çıkış Hataları

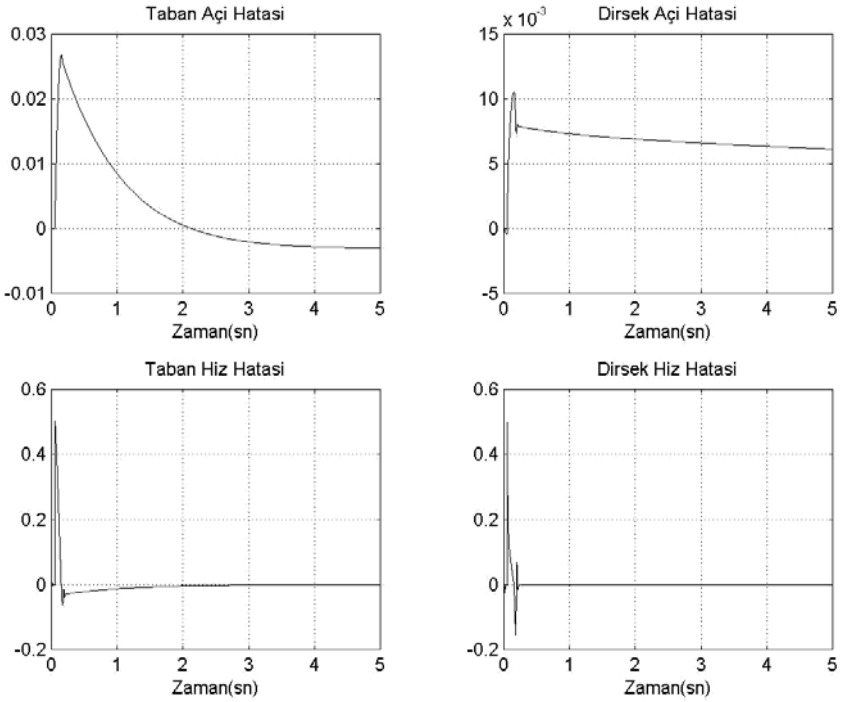



Şekil 4.5 Direk Ters Modelleme İle Tork-Zaman Grafikleri

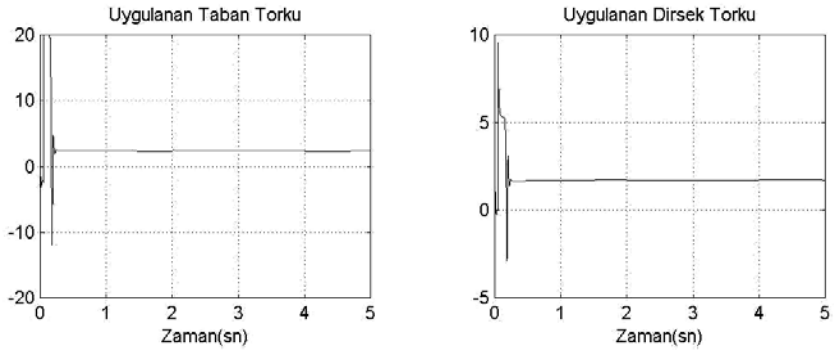
### 4.2.2 Adım referans hız ile süreç denetim uygulaması




Şekil 4.6 Direk Ters Modelleme İle Sistemin Çıkış ve Referans Değerleri




Şekil 4.7 Direk Ters Modelleme İle Sistemin Çıkış Hataları



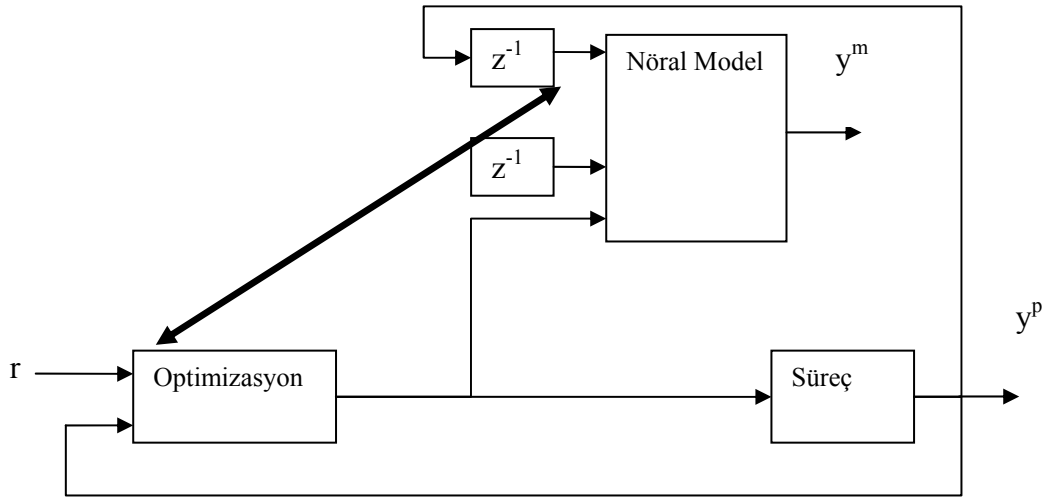

Şekil 4.8 Direk Ters Modelleme İle Tork-Zaman Grafikleri

Sinüzoidal giriş ile yapılan uygulamada robotun ± 0.0028 rad açısal konum hatası ve ± 0.02 rad/sn açısal hız hatası ile kontrol edilebildiği görülmektedir. Adım referans değişme ile yapılan deneyde ise sistemin referans değere ulaşmasının yaklaşık 0.2 saniye aldığı fakat aşma değerinin 0.5 rad/sn'ye kadar çıktığı görülmektedir. Hata grafikleri incelendiğinde deney ilerledikçe denetçinin uyarlanır olmasına rağmen bir gözle görülü iyileşme görülmemiş denetçi daha fazla iyileştirilememiştir. Bunun nedeni denetçide kullanılan yapay sinir ağının deney öncesi eğitilmiş olmasıdır.

Bu deneyler sonucunda direk ters modelleme ile robotun kontrol edilebildiği, referanstaki ani değişimlere hızlı cevap verdiği fakat aşma değerinin yüksek olduğu görülmektedir.

### 4.3 Öngörülü Denetim ile Süreç Denetim Uygulaması

Öngörülü denetim uygulamasında kullanılan blok şeması şekil 4.9'da verilmiştir.




Şekil 4.9 Öngörülü Denetim Uygulaması Blok Şeması

Öngörülü denetim uygulamasında robotun modeli yapay sinir ağları kullanılarak oluşturulmuştur. Bu modelin girdileri sistemin önceki durumu ve uygulanan torktur. Elde edilen çıktı ivme değeridir. İvmenin entegrali alınarak hız değerine, hızın entegrali alınarak konum değerine ulaşılmaktadır. Modelde kullanılan yapay sinir ağı bir örnekleme zamanı önceki giriş ve çıkış verileri kullanılarak uyarlanmaktadır.

Denetçi olarak MATLAB içinde yer alan hazır optimizasyon fonksiyonu kullanılmıştır. Yapılan denemelerde optimizasyon ile maliyet fonksiyonu minimize edilmeye çalışılmaktadır.

Maliyet fonksiyonu:

$$J = \sum_{k=N_1}^{N_2} (R(t+k) - Y(t+k)) + \rho \sum_{k=1}^{N_u} \Delta u(t+k-1) \qquad (4.5)$$

$$\Delta u(t) = u(t) - u(t-1) \qquad (4.6)$$

$N_1$=1

$N_2$=4

$N_u$=2

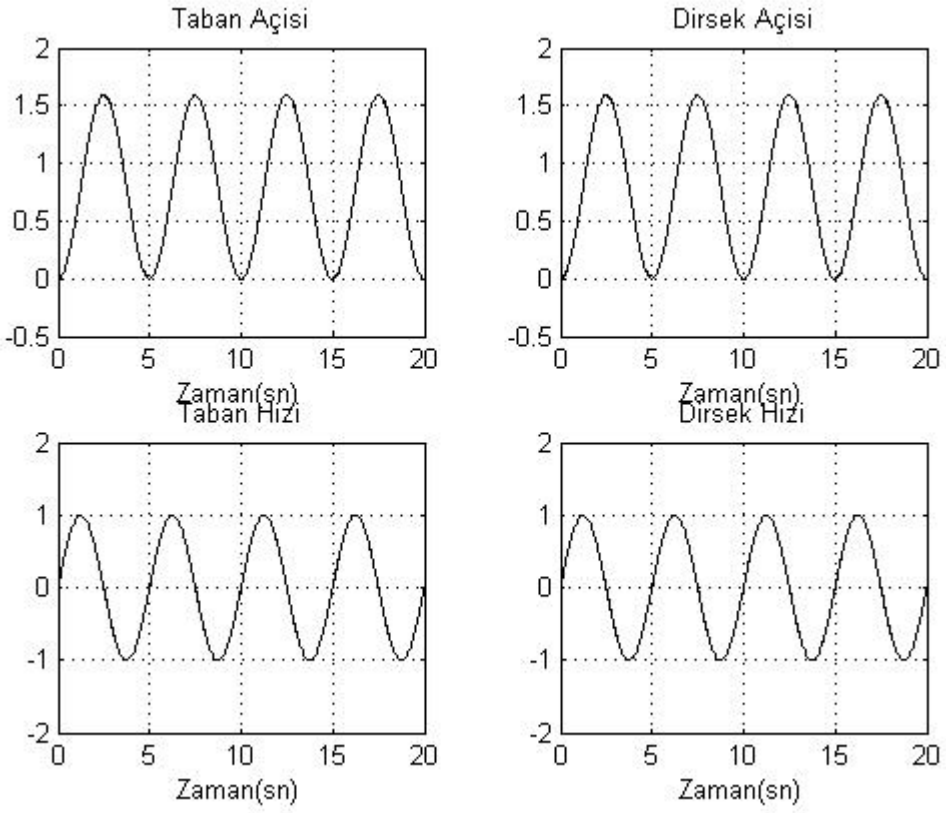
$T_s$: Örnekleme periyodu

$\rho$= 0,008

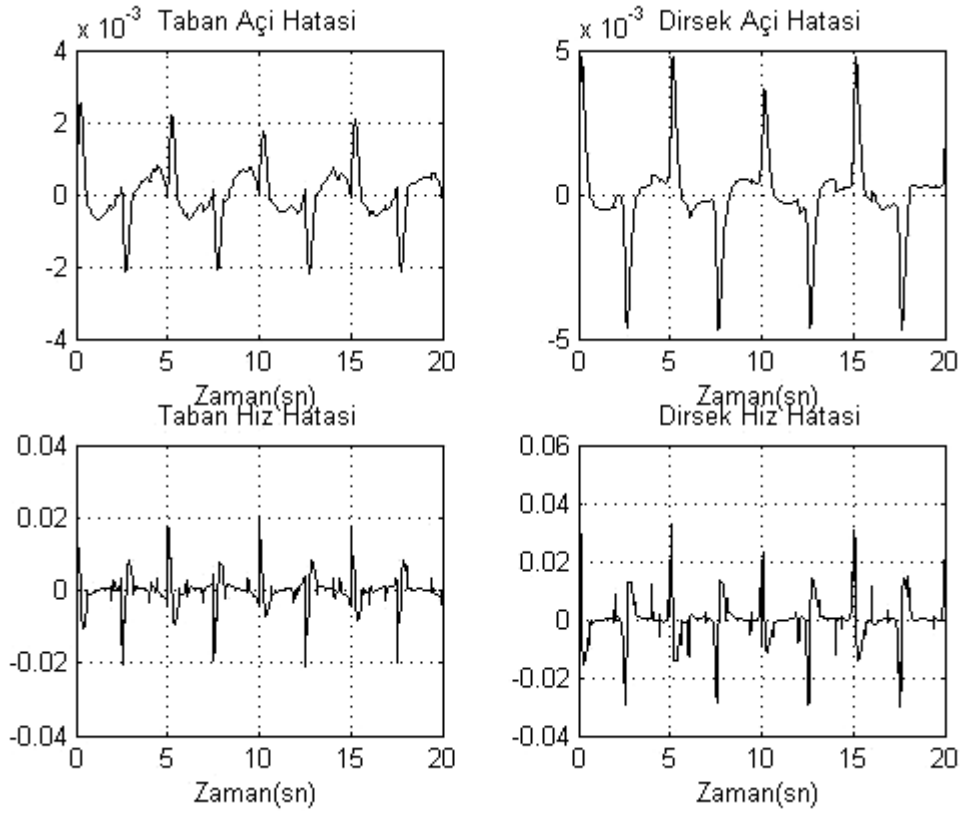
Model ile optimizasyon arasındaki çapraz iki yönlü bağlantı ise özyineli optimizasyon sırasında optimizasyon algoritmasının defalarca robotun modelini çağırarak o anki hesaplanan tork değerine karşılık elde edilecek konum ve hız değerleri hesaplanmasını ve bu değerlerin tekrar optimizasyon algoritmasında kullanılmasını sembolize etmektedir.

Sinüzoidal referans hız ile yapılan deneme sonuçları şekil 4.10-4-14'te ve adım referans hız ile yapılan deneme sonuçları şekil 4.15-4-19'te verilmiştir.

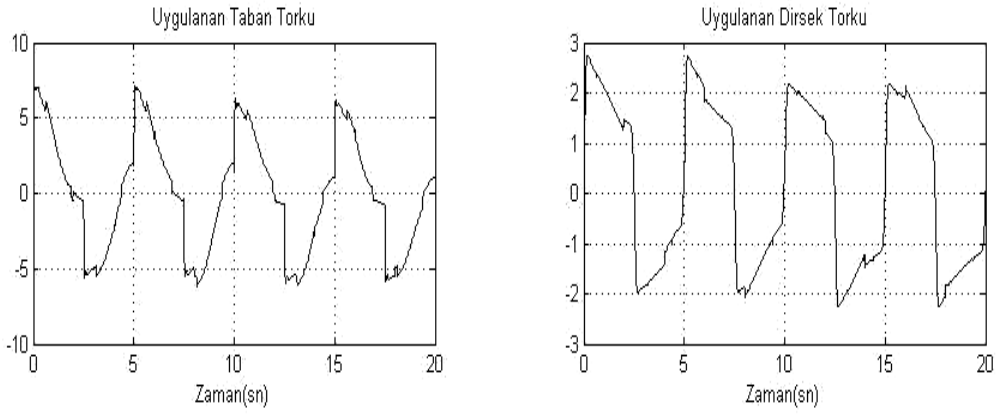
### 4.3.1 Sinüzoidal referans hız ile süreç denetim uygulaması



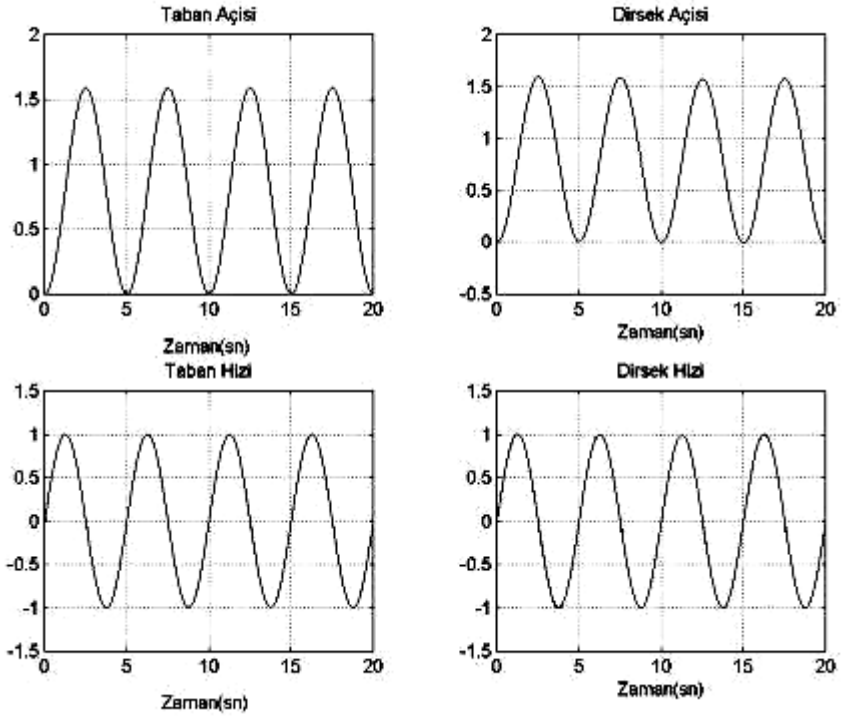

Şekil 4.10 Öngörülü Denetim ile Sistemin Çıkış ve Referans Değerleri



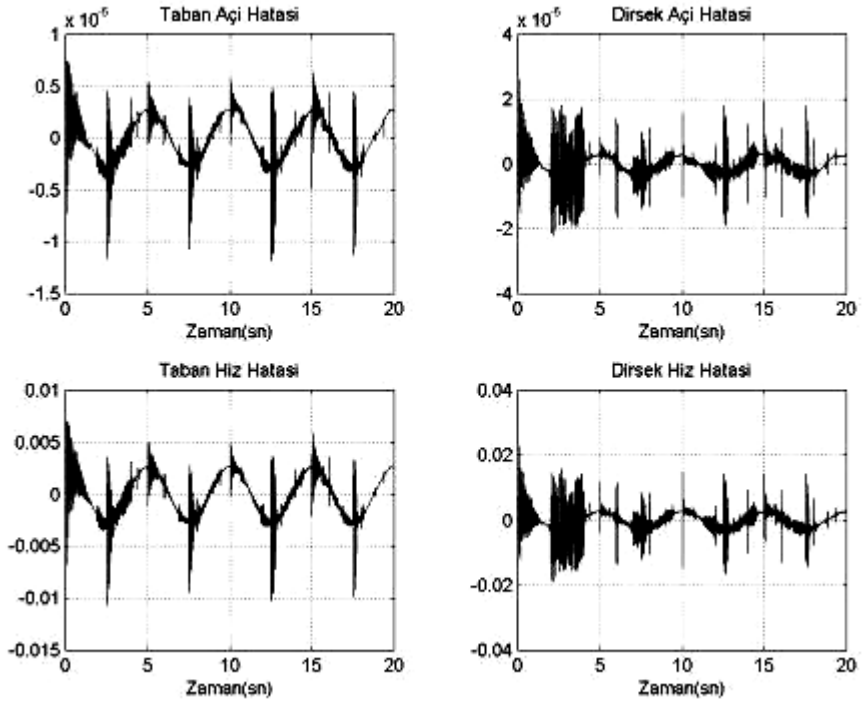

Şekil 4.11 Öngörülü Denetim ile Sistemin Çıkış Hataları




Şekil 4.12 Öngörülü Denetim İle Tork-Zaman Grafikleri

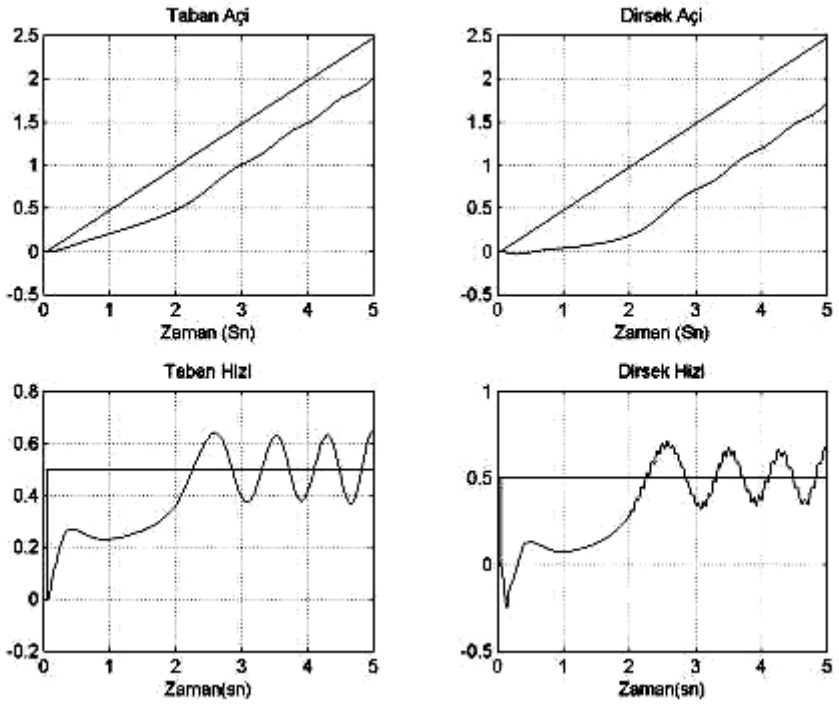



Şekil 4.13 Öngörülü Denetim Yapılan Denetim Sonucunda Modelin Çıkışları

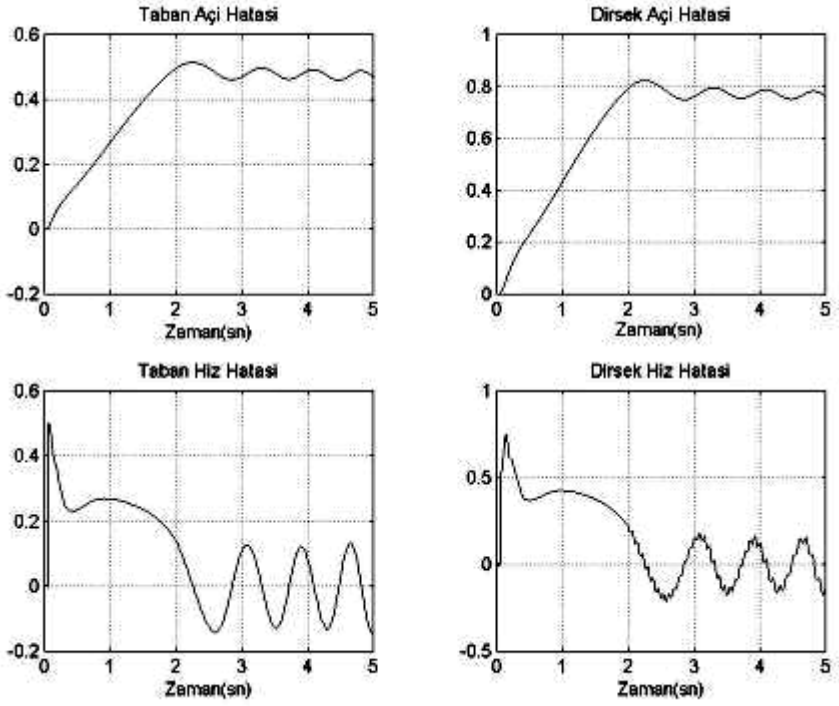



Şekil 4.14 Öngörülü Denetim Yapılan Denetim Sonucunda Model Hatası

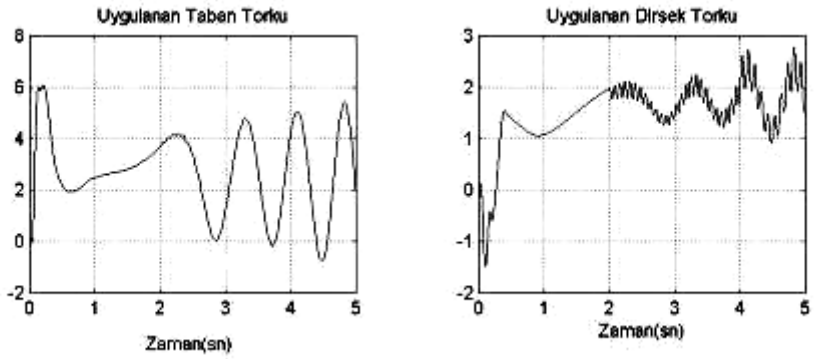
### 4.3.2 Adım referans hız ile süreç denetim uygulaması



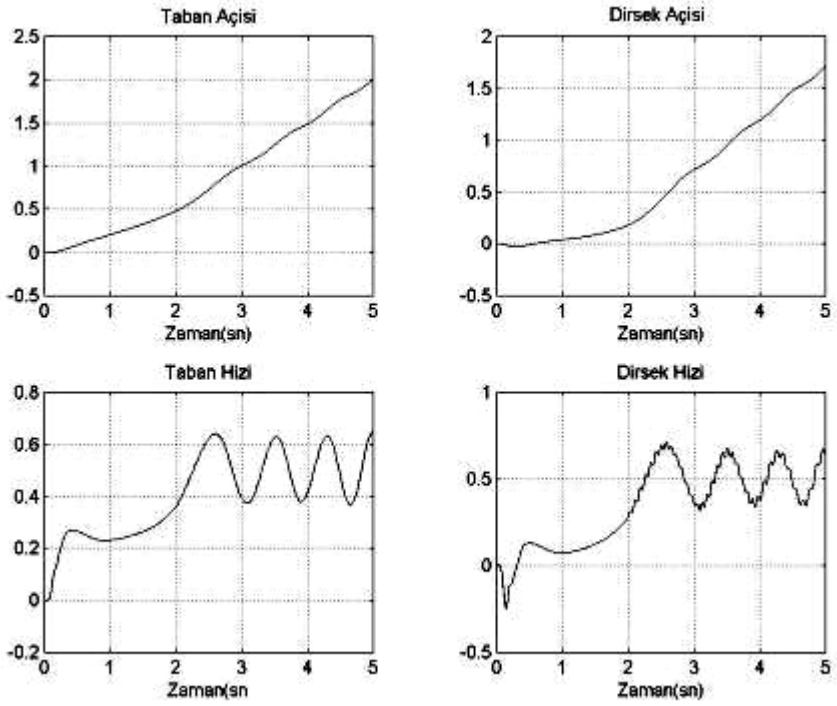

Şekil 4.15 Öngörülü Denetim ile Sistemin Çıkış ve Referans Değerleri



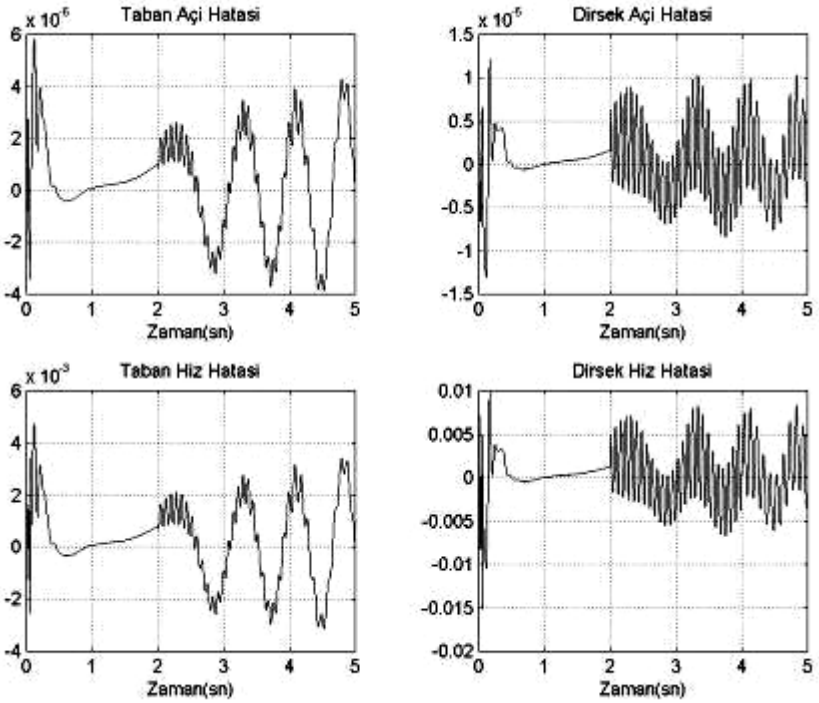

Şekil 4.16 Öngörülü Denetim ile Sistemin Çıkış Hataları




Şekil 4.17 Öngörülü Denetim İle Tork-Zaman Grafikleri




Şekil 4.18 Öngörülü Denetim Yapılan Denetim Sonucunda Modelin Çıkışları

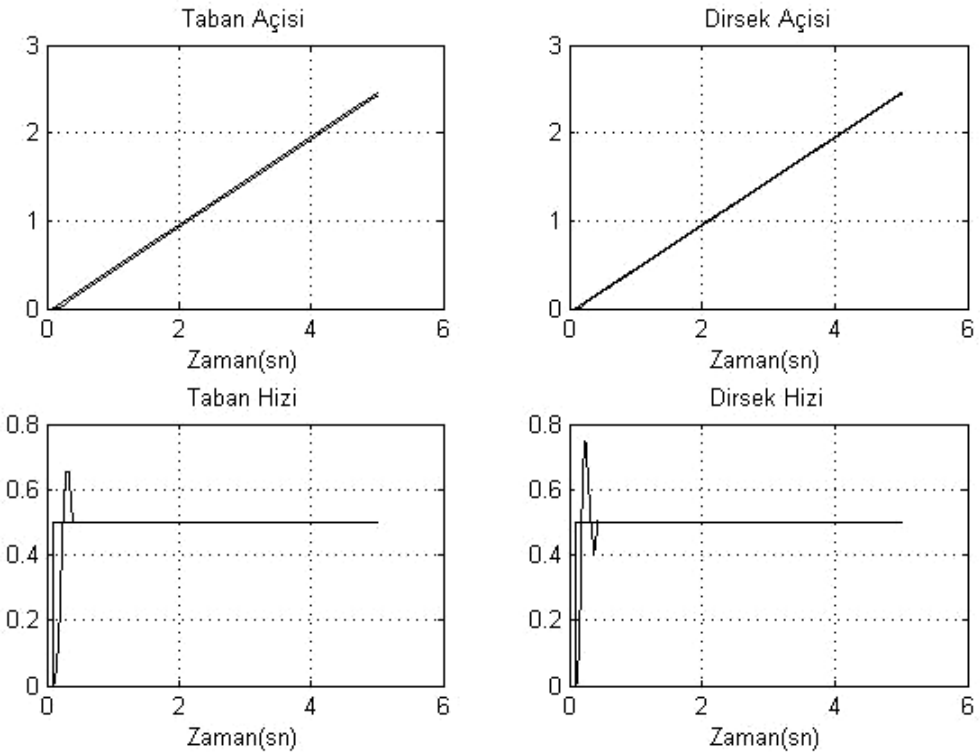



Şekil 4.19 Öngörülü Denetim Yapılan Denetim Sonucunda Model Hatası

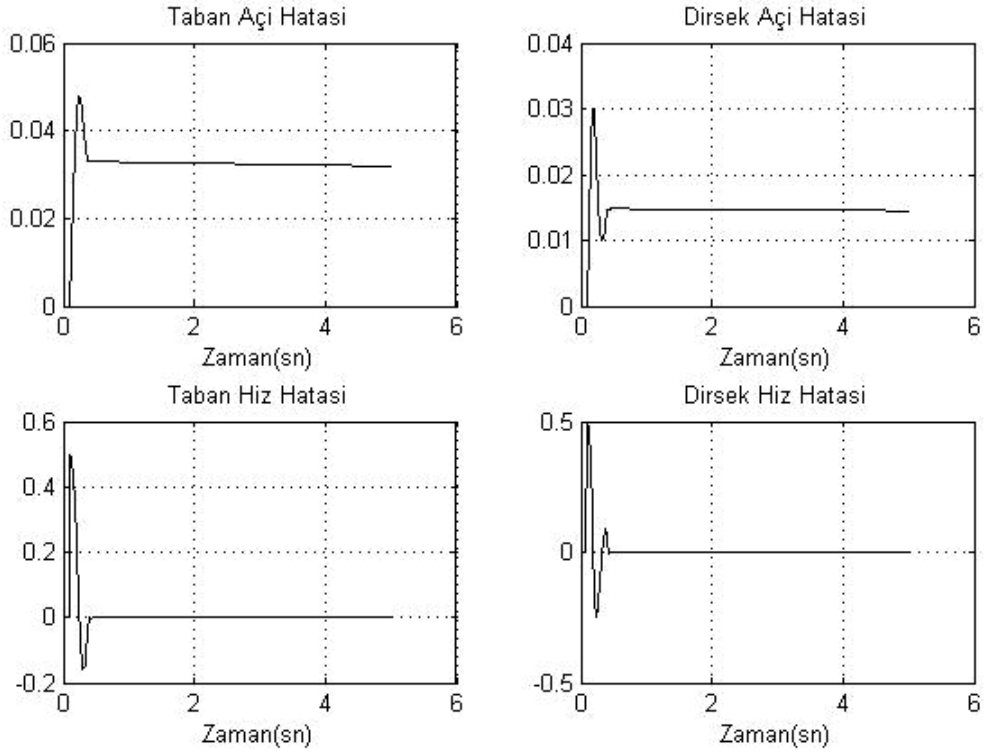
Sinüzoidal giriş ile yapılan uygulamada robotun ± 0.02 radyan açısal konum hatası ve ± 0.04 rad/sn açısal hız hatası ile kontrol edilebildiği görülmektedir. Adım referans değişme ile yapılan deneyde ise sistemin aşma değerinin 0.2 rad/sn fakat referans değere ulaşmasının yaklaşık 2 saniye aldığı görülmektedir.

Öngörülü denetimin başarısı öngörülerin yapılacağı model ve optimizasyon algoritmasının başarısına dayanmaktadır. Öngörülü denetim deneylerinde görülen hataların bir kısmı modelden bir kısmı da optimizasyon algoritmasından kaynaklanmaktadır. Bu yapılan deneyler de model hız hataları karşılaştırıldığında model ile süreç arasındaki hız hatası 0.01 radyan/sn iken süreç ve referans arasındaki hata 0.02 radyan/sn seviyelerine çıkmaktadır. Bunun sebebi deneyler sırasında basit bir optimizasyon algoritması kullanılmasıdır. Ayrıca denetimler sırasında kullanılan maliyet fonksiyonunda her çıkış değişkenine verilen ağırlık aynıdır, bu ağırlıklar değiştirilerek istenilen çıkış değişkenlerinin maliyet fonksiyonundaki ağırlığı arttırılarak o değişkenin referans hatası daha düşük seviyelere çekilebilir.

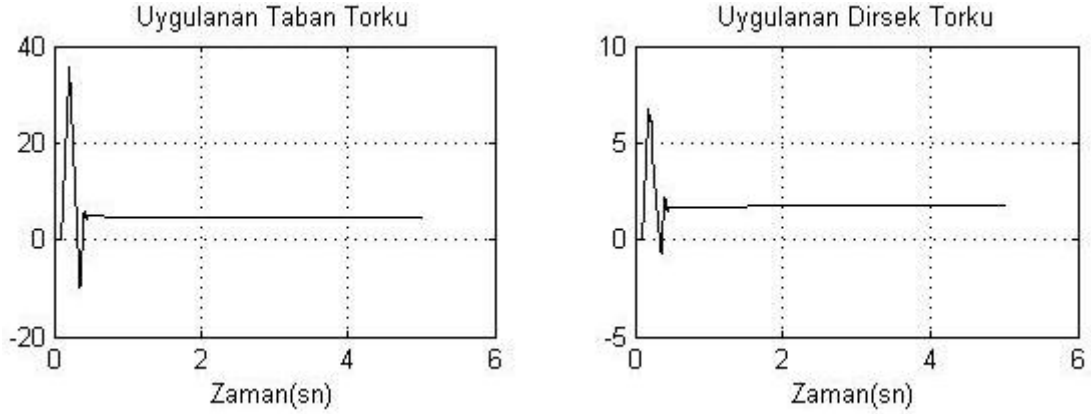
Mükemmel bir model kullanılması durumunda ise öngörülü denetim ile elde edilecek çıkışlar şekil 4.20-4.22'de olduğu gibi olacaktır.




Şekil 4.20 Mükemmel Model ile Yapılan Öngörülü Denetim Sonucunda Sistemin Çıkış ve Referans Değerleri




Şekil 4.21 Mükemmel Model ile Yapılan Öngörülü Denetim Sonucunda Sistemin Çıkış Hataları

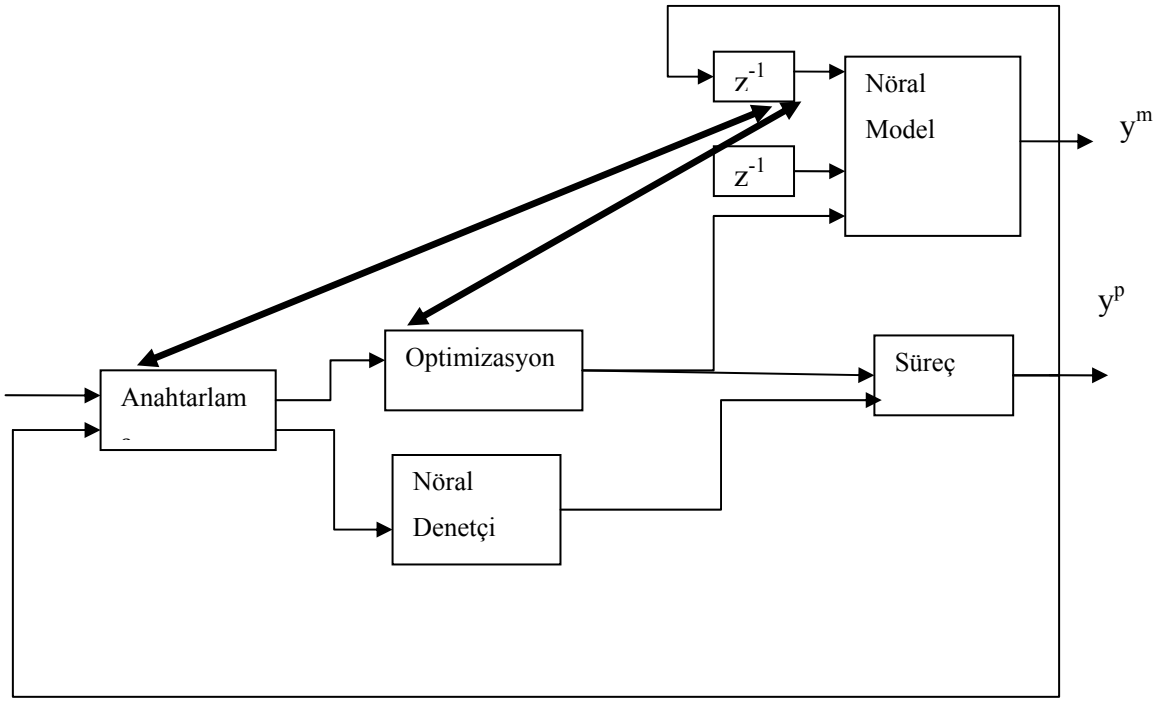



Şekil 4.22 Mükemmel Model ile Yapılan Öngörülü Denetim Sonucunda Tork-Zaman Grafikleri

Bu deneyler sonucunda öngörülü denetim ile robotun kontrol edilebildiği, referanstaki ani değişimlerde direk ters modellemeye göre düşük olduğu fakat referanstaki değişimlere yavaş cevap verdiği görülmektedir.

### 4.4 Uyarlamalı Anahtarlama Denetim Uygulaması

Bu tez çalışmasında daha iyi bir denetim için direk ters modelleme ve öngörülü denetim algoritmaları arasında anahtarlama yaparak denetim uygulanmıştır. Uygulamalarda öngörülü denetimin referanstaki ani değişimlere yavaş cevap verdiği, direk ters modellemenin hızlı cevap verdiği fakat direk ters modelleme ile aşma değerinin yüksek olduğu görülmüştü. Anahtarlama ile öngörülü denetimden hızlı cevap veren ve de direk ters modellemeden daha düşük aşma değerine sahip bir yöntem hedeflenmiştir. Uygulanan yöntemin blok şeması şekil 4.23'te verilmiştir.




Şekil 4.23 Uyarlamalı Anahtarlama Yöntemi Blok Şeması

Anahtarlama için RY kriteri kullanıldı

$$RY = (R(t) - Y(t-1))^T (R(t) - Y(t-1)) \qquad (4.7)$$

RY kriterinin belirlenen değerden küçük olduğu bölgelerde öngörülü denetim büyük olduğu yerlerde ise direk ters modelleme kullanılmıştır.

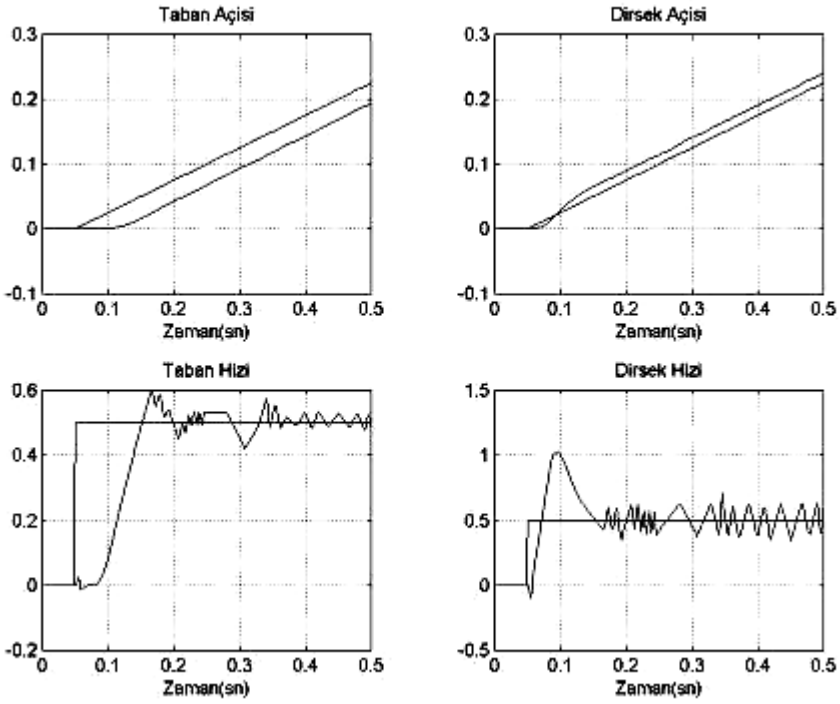
Blok diyagramda yer alan model ile anahtarlama arasındaki iki yönlü çizgi anahtarlama ile model arasındaki bağlantıyı göstermektedir. Yani sistemin gürbüzlüğünü arttırmak üzere her zaman ters modelleme ile sistem için bir tork değeri hesaplanmakta ve sistemin modeli kullanılarak bu tork değeri kullanılarak sistemin beklenen çıktısı elde edilmekte ve bu çıktıya göre yenir bir RY değeri $RY_e$ oluşturulmaktadır.

Eğer bu $RY_e$ değeri önceki RY değerinden büyükse öngörülü denetim kullanılmaktadır. RY değeri denemelerle bulunmuştur. RY değerinin çok küçük ya da büyük olması durumunda direk ters modelleme uygulamasından öngörülü denetim uygulamasına zamanında geçiş sağlanamamaktadır. Bu uygulamalarda RY değeri için uygun değer 0.05 olarak bulunmuştur.

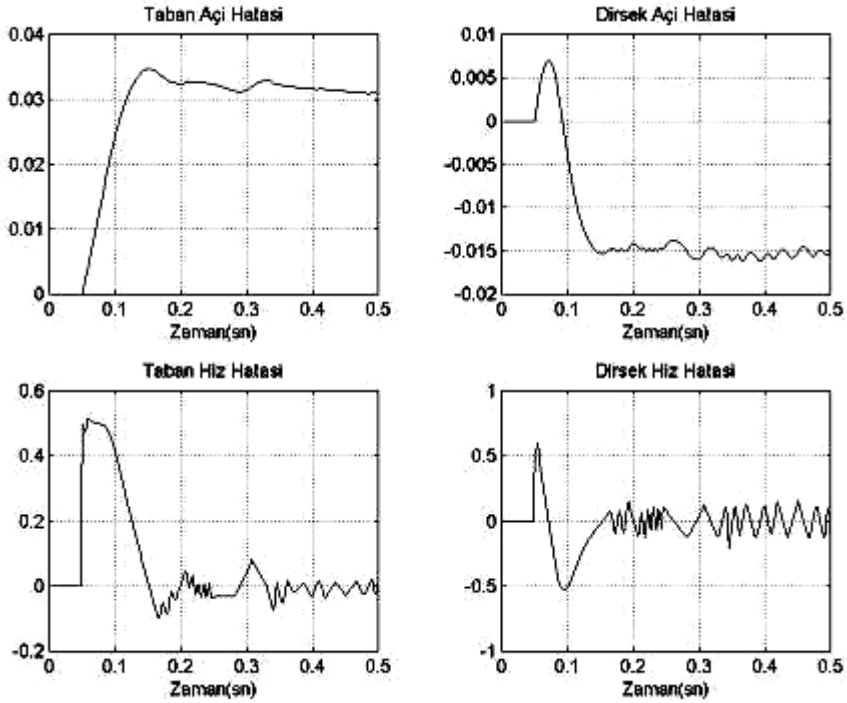
Bu yapıda öngörülü denetim uygulamasında modelde ve direk ters modellemede kullanılan yapay sinir ağları bir örnekleme zamanı önceki verilerle kendilerini uyarlamaktadırlar.

RY'nin 0.2 değeri için yapılan deneme sonuçları şekil 4.24-4-28'de ve RY'nin 0.5 değeri için yapılan deneme sonuçları şekil 4.29-4-33'te verilmiştir.

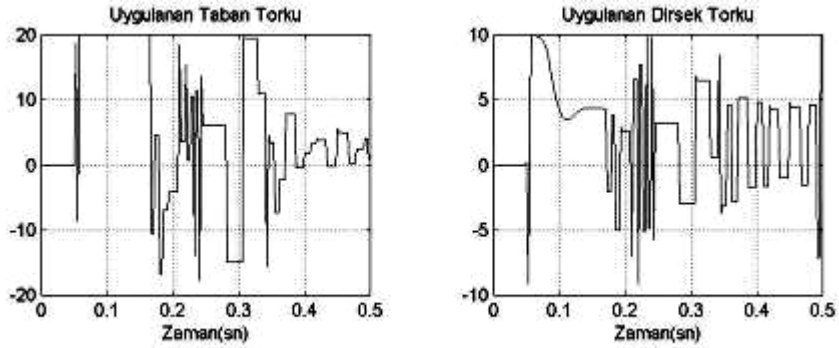
### 4.4.1 RY=0.02 için yapılan uygulamalar



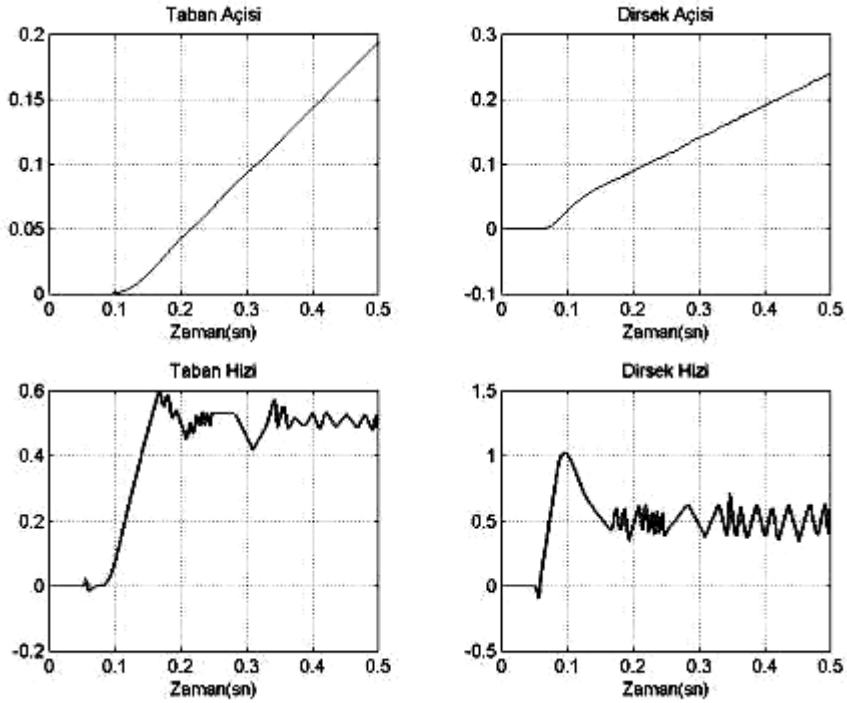

Şekil 4.24 Anahtarlamalı Model İle Sistemin Çıkış ve Referans Değerleri



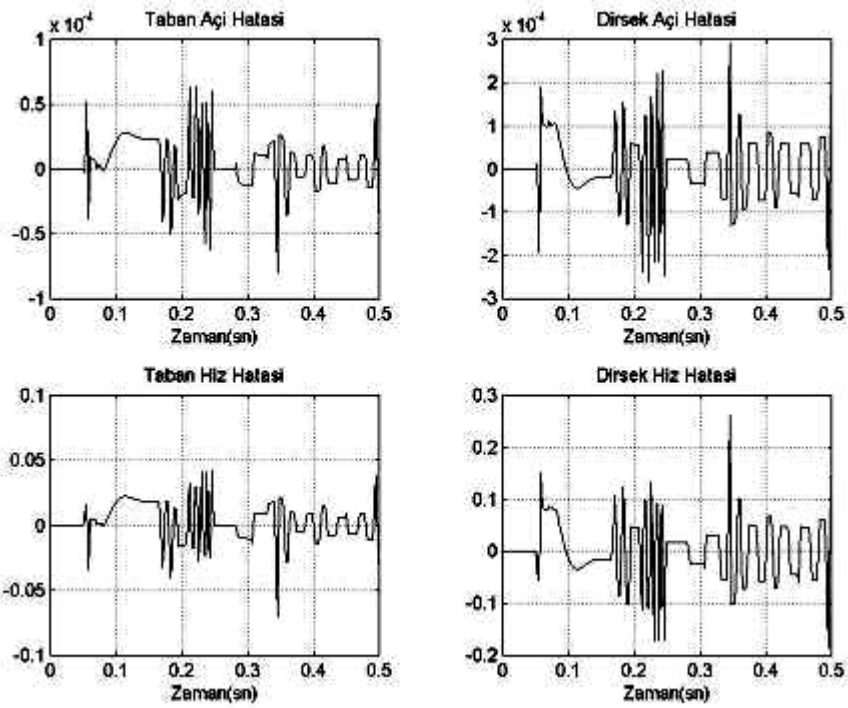

Şekil 4.25 Anahtarlamalı Model İle Sistemin Çıkış Hataları




Şekil 4.26 Anahtarlamalı Model ile Tork-Zaman Grafikleri

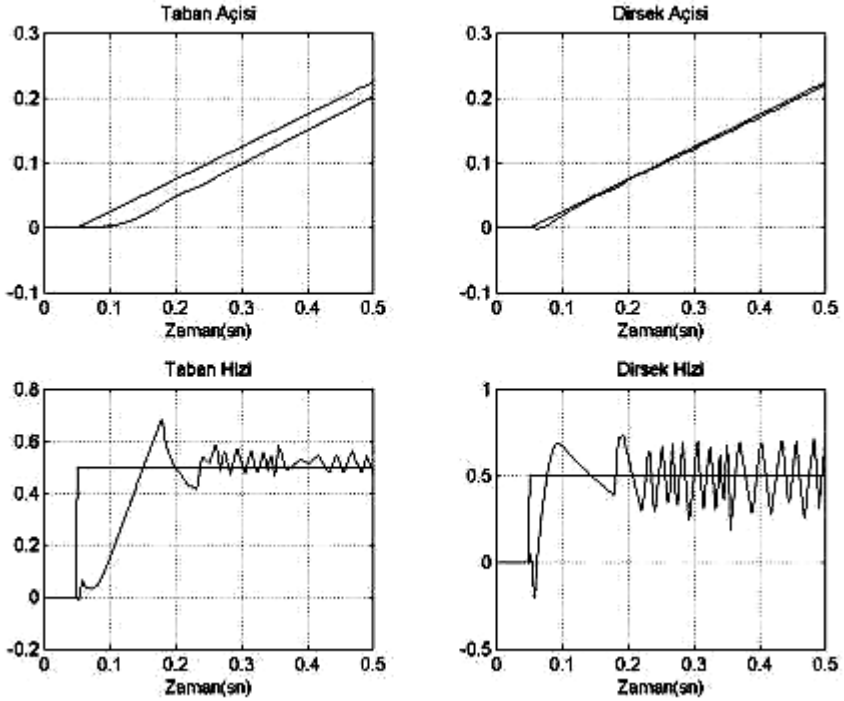



Şekil 4.27 Anahtarlamalı Model İle Yapılan Denetim Sonucunda Modelin Çıkışları

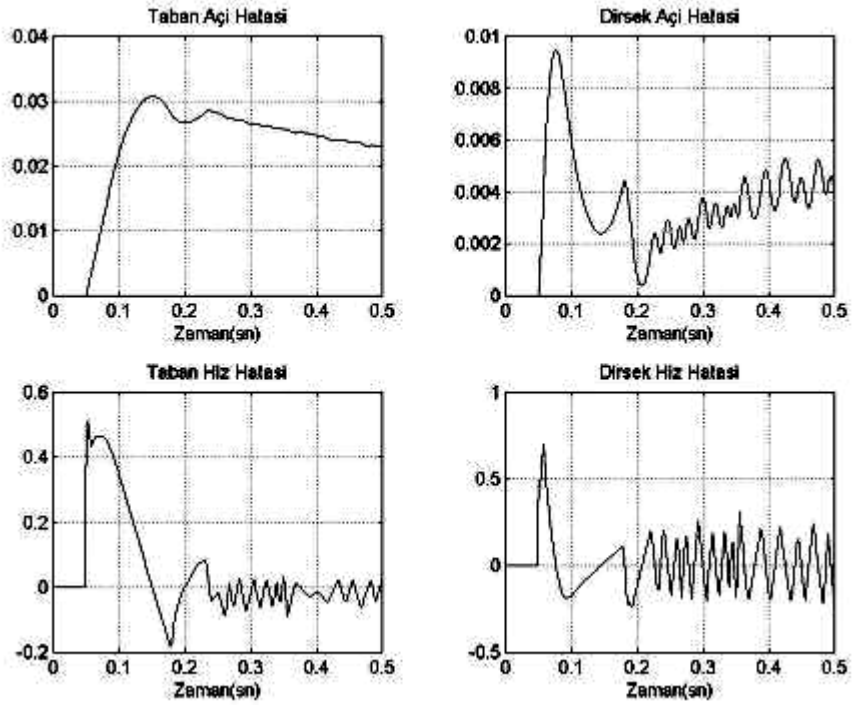



Şekil 4.28 Anahtarlamalı Model İle Yapılan Denetim Sonucunda Model Hatası

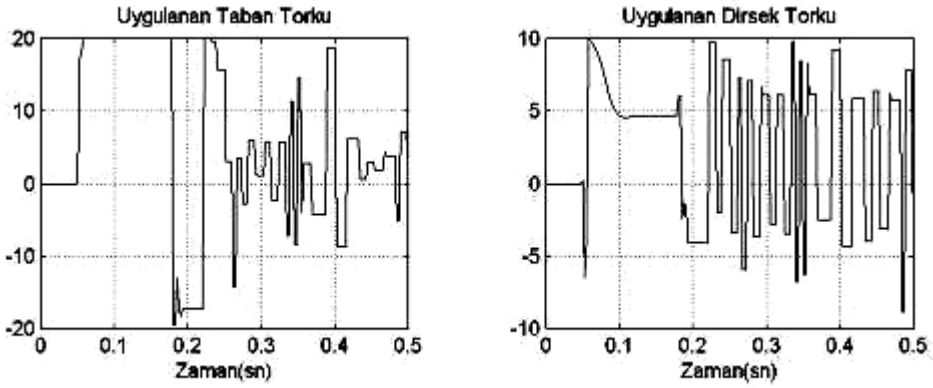
### 4.4.2 RY=0.05 için yapılan uygulamalar



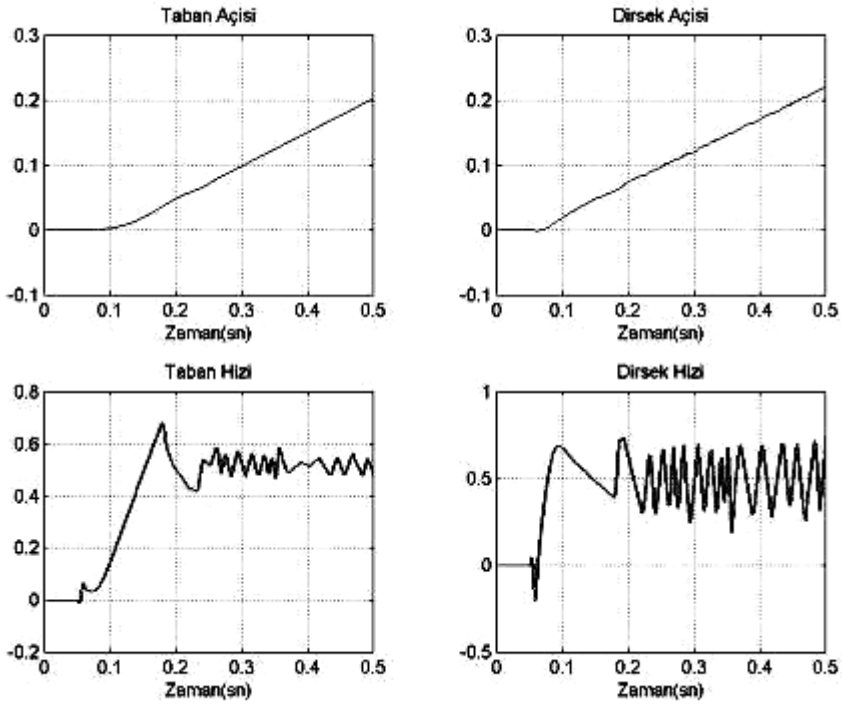

Şekil 4.29 Anahtarlamalı İle Sistemin Çıkış ve Referans Değerleri



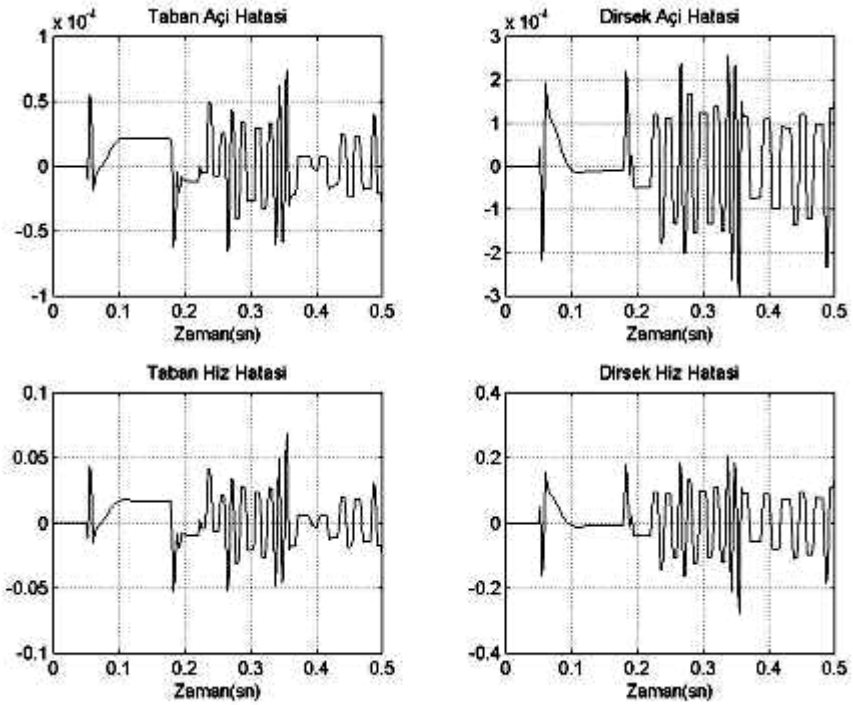

Şekil 4.30 Anahtarlamalı Model İle Sistemin Çıkış Hataları




Şekil 4.31 Anahtarlamalı Model ile Tork-Zaman Grafikleri




Şekil 4.32 Anahtarlamalı Model İle Yapılan Denetim Sonucunda Modelin Çıkışları




Şekil 4.33 Anahtarlamalı Model İle Yapılan Denetim Sonucunda Model Hatası

Direk ters modelleme ile yapılan deneyde sistemin referans değere ulaşmasının yaklaşık 0.2 saniye aldığı fakat aşma değerinin 0.5 rad/sn'ye kadar çıktığı, öngörülü denetimde ise yöntemin referanstaki değişime 2 saniyede cevap verdiği ve aşma değerinin ise 0.2 rad/sn olduğu görülmüştü. Uyarlamalı anahtarlama yönteminde ise aşma değeri 0.2 rad/sn ve cevap verme zamanı ise 0.2 saniye olmuştur. Böylece anahtarlama ile iki yöntemin de olumlu yönleri sonuca yansıtılmış oldu.

Uygulamalar sırasında kullanılan yöntemlerin üçünde de kullanılan yapay sinir ağları uyarlanır yapıda oluşturulmuştur. Denemeler sırasında bu ağların hatalarında gözle görülür bir iyileşme görülmemesi deneyler öncesi ağların eğitiminin yeterli olduğunu göstermektedir.

Hesaplama yükü yönünden karşılaştırıldığında; öngörülü denetim sürecinde hesaplamalar özyineli bir optimizasyon algoritmasıyla yapılmakta ve bu adım ilerlemeleri sırasınca defalarca model çağrılmakta ve bu ilerlemeler için model ile çıkış

tahmin edilmektedir. Direk ters model uygulamasında ise veriler ters modelden bir defa geçmektedir. Anahtarlamalı yöntemde ise bu yük iki yöntemin arasında bir yerdedir.

Uygulama yapılan her üç yöntemle de doğrusalsız olan robot süreci başarı ile denetlenmiştir.

Bu üç uygulamada da klasik yöntemlerde ihtiyaç duyulan denetlenilecek sistem hakkında bilgiye ihtiyaç duyulmamaktadır. Klasik yöntemlerde denetlenilecek sürecin matematiksel modeline ya da sistem hakkında parametre tahminine ihtiyaç duyulmaktadır. Oysa bu örnek uygulamalarda da görüldüğü gibi yapay sinir ağlarıyla yapılan süreç denetimlerinde bu bilgilere ihtiyaç duyulmamaktadır, çünkü yapay sinir ağları örneklerle öğrenirler. Denetlenecek sürecin elde olan ya da eş zamanlı elde edilen giriş çıkış çiftleri ile sürecin yapay sinir ağı modeli ya da denetçi oluşturulabilmektedir. Bu tür süreç hakkında bilginin bulunmadığı süreçlerde yapay sinir ağları ile yapılacak uygulamalar faydalı olacaktır.

Klasik doğrusalsız süreç denetimi yöntemlerinde doğrusalsız süreçler bir denge noktası etrafında doğrusal olarak kabul edilir ve bu şekilde denetlenir. Fakat yapay sinir ağları ile yapılan süreç denetimi uygulamalarında bu şekilde doğrusal bir yaklaşıma gerek duyulmamaktadır. Bu uygulamalarda da görüldüğü gibi hem robot modelinde hem de denetçide kullanılan yapay sinir ağlarının robot sürecinin sahip olduğu doğrusalsızlıklara başarıyla yaklaşabilmektedir. Bu yetenek de yapay sinir ağlarını doğrusalsız süreçlerin denetimi için önemli bir aday yapmaktadır.

# 5. SONUÇ

Bu tezdeki uygulamalar yapay sinir ağlarının kullanımı yönünden kavram olarak iki değişik uygulama yöntemi üzerinde yapılmıştır. Direk ters modelleme uygulamasında yapay sinir ağı denetçi olarak kullanılırken öngörülü denetim algoritmasında ise sürecin modeli olarak kullanılmıştır.

İki denetim modeli incelendiğinde:
Direk ters modelleme uygulamasının girişteki değişime hızlı cevap verdiği fakat aşma değerinin yüksek olduğu görülmektedir. Ayrıca uygulamalar sırasında gürültü uygulanmamıştır sistemde gürültü veya diğer bozucu etkenler olması durumunda direk ters modellemenin girdisi referans değer ile gerçek değerin farkı olduğu için gürültü de çıkışa yansıtılacaktır.

Öngörülü denetim uygulamasında ise girişteki değişime yavaş yanıt verdiği ama aşma değerinin küçük olduğu görülmüştür. Ayarlanan değer etrafındaki salınımlar özellikle optimizasyon algoritmasından kaynaklanmaktadır. Uygulamalar sırasında MATLAB içinde yer alan hazır optimizasyon algoritması kullanılmıştır ileriki çalışmalarda bu salınımların önlenmesi için daha uygun bir optimizasyon algoritması geliştirilebilir. Ayrıca seçilen maliyet fonksiyonda bütün durum değişkenleri eşit ağırlıklandırılmıştır. Değişkenlerin ağırlıkları değiştirilerek istenen değişken yada değişkenler daha fazla ağırlıklandırılabilir.

Direk ters modelleme ile öngörülü denetim uygulaması arasında yapılan anahtarlama ile iki modelin faydalı yönleri yansıtılmaya çalışılmıştır. Referans değerde değişiklik olduğu zaman direk ters modelleme ile referans değere hızlı yaklaşmak ve referans değere yakın olduğu yerlerde ise direk ters modelleme ile oluşan yüksek aşma öngörülü denetime anahtarlanarak düzeltilmiştir. Ayrıca sistemin dış etmenlere dayanıklılığını arttırmak üzere denetim uygulaması esnasında direk ters modelleme ile üretilen tork değerleri sistemin modeli ile denenerek çıkış değerinde görülebilecek sapma durumunda öngörülü denetim uygulamasına geçilmektedir.

Yapay sinir ağları ile oluşturulan bu tür denetleyiciler özellikle lineer denetleyiciler ile denetlenemeyen birçok süreçte uygulanabilir.

İleriki çalışmalarda yapay sinir ağlarının yapısını belirlemek için bulunabilecek bir yöntem bulunması durumunda yapay sinir ağlarıyla süreç denetimi uygulamaları kolaylaşacaktır.

Genellikle gerçek süreçler çok değişkenli ve karmaşık olduğundan temel olarak doğrusal değildirler. Ayrıca gürültü ve çevresel etkenler nedeniyle süreç değişkenleri zamana göre değiştiğinden çevreye ilişkin bilgiler kesinlikten uzaktır. Klasik kontrol algoritmalarının yetersiz kaldığı robot bilim gibi uygulama alanlarında yapay sinir ağları denetim için önemli araçtırlar.

**KAYNAKLAR**


Cembrano, G. and Wells, G. 1992. Neural Networks for Control, Boulberg, L. Krıjgsman, A. and Vingehoods, R. A. Application of Artificial Intelligence in Process Control. Pergoman Pres, 388 – 402.

Noriega, J. R. and Wang, H. 1998. A Direct Adaptive Neural-Netwok Control for Unknown Nonlinear Systems and Its Application. IEEE Transactions on Neural Networks. 9(1). Pp 27-34

Chen, L. and Narendra K. S. 2001. Nonlinear Adaptive Control Using Neural Netwoks and Multiple Models. Automatica, 1245-1255

Cichocki, A. Unbehauen, R. 1993. Neural Networks for Optimization and Signal Processing. WILEY. Chichester

Norgaard, M. Ravn, O. Poulsen, N. K. and Hansen, L.K. 2000. Neural Networks for Modelling and Control of Dynamic Systems, Springer, 246 p. London

Efe, M. Ö. ve Kaynak O. 2004. Yapay Sinir Ağları ve Uygulamaları. Boğaziçi Üniversitesi, 148s., İstanbul

Freeman, L. A. Skapura, D. M. 1991.Neural Networks Algorithms Applications and Programing Techniques Addison-Wesley

Kartalopoulos, S. 1996. Understanding Neural Networks and Fuzzy Logic: Basic Concepts and Applications IEEE Press, 232p New York

Hagan, M. T. Demuth, H. B. and Beale, M. H. 1996. Neural Network Design, University of Colorado, Colorado

Hunt, K. J. Sbarbaro, D. Zbikowski , R. and Gawthrop, P. J. 1992. Neural Networks for Control Systems – A Survey. Automatica, 28(6) 1083-1112

Anonymous. 2006. www.statsoftinc.com Erişim Tarihi: 01.02.2006

Lazar, M. and Pastavanu, O. 2002. A neural predictive controller for non-linear systems, Mathematics and Computers in Simulation, 60 315-324

Rivals, I. Personnaz, L. 2000. Nonlinear Internal Model Control sing Neural Networks: Application to Process with Delay and Design Issues, IEEE Transections on Neural Networks, 11(1) pp 80-90

Wang, L. Wan, F. 2001. Structured Neural Networks for Constrained Model Predictive Control. Automatica, 1235-1243

Denker, A. and Ohnishi, K. 1996. Robust Tracking Control of Mechatronic Arms. IEEE/Asme Transactıons on Mechatronıcs. 1(2), 181-188

Cılız, M. K. 2005. Adaptive Control of Robot Manipulators with Neural Network Based Compensation Of Frictional Uncertainties. Robotica, 23, 159-167

**EK 1 Kullanılan Matlab Fonksiyonları**

**DICBASLA.M**

```
global DIC P

%Bu fonksiyonun amacı denetim için gerekli yapay sinir  ağını oluşturmaktır.
sfark;%sfark fonksiyonu ile veri çalışma alanında yer alan durum ve tork
%değişkenleriden yapay sinir ağının eğitimiiçin gerekli değişkenleri oluşturur
DIC=newff([-1 1;-1 1;-1 1;-1 1],[4 8 2],{'purelin','purelin','tansig'},'traingd');%yapay
sinir ağının yapısı
%belirlenmektedir. bu örnekte 3 katmanlı katmanlarda sırasıyla 4,8 ve 2 nöron bulunan
yapay sinir ağı oluşturulmuştur.
DIC.trainParam.show = 100;%Eğitim sırasında her 100 iterasyonda sonucu göster
DIC.trainParam.lr = 0.05;%iterasyon katsayısı 0.05
DIC.trainParam.epochs = 50000;
DIC.trainParam.goal = 1e-5;%eğitim amacı
g=sstate';%yapay sinir ağının eğitimi için gerekli giriş değerleri
c=suapp';%yapay sinir ağının eğitimi için gerekli hedef değerleri
[DIC,tr]=train(DIC,g,c);
DIC.adaptParam.passes=1;%adaptasyon sırasında ağdan 1 defa geçirillecektir
DIC.trainParam.lr = 0.01; % adaptasyon iterrasyonu
```

**SFARK.M**

```
global farkstate sstate suapp

%direk ters modellemede kullanılan yapay sinir ağının eğitimi için gereki giriş ve çıkış
değerleri oluşturulmaktadır.

farkstate(1,1)=state(1,1);
farkstate(1,2)=state(1,2);
farkstate(1,3)=state(1,3);
farkstate(1,4)=state(1,4);

for i =2:8000,
  for j=1:4,
  farkstate(i,j)=state(i,j)-state((i-1),j);
   end;
end;
for m=1:250,
  for n=1:4,
     sstate(m,n)= farkstate(10*m,n);
  end;
  suapp(m,1)= 0.05*uapp(10*m,1);
  suapp(m,2)= 0.1*uapp(10*m,2);
end;
```

**DIC.M**

```
function Y1 = FDIC(invector)

global DIC Y1 P T

%Bu fonksiyon ile yapay sinir ağının çıkışları oluşturulmakta ve bir adım önceki
verilere göre yapay
% sinir ağı adapte edilmektedir.
IN=invector(1:4);%Yapay sinir ağının giriş vektörü
P=invector(5:8);%bir adım önceki hata değerleri
T=[0.05*invector(9);0.1*invector(10)];;%bir adım önceki tork değerleri
[DIC,Y,E] = ADAPT(DIC,P,T);% adaptasyon sırasında bir adım önceki tork değerleri
ve hata değerleri kullanılmaktadır
A=sim(DIC,IN);%yapay sinir ağının çıkış değerleri
Y1=[20*A(1,1);10*A(2,1)];%yapay sinir ağının çıkış değerleri ağırlıklandırılarak tork
değerleri elde edilir
```

**GPCBASLA.M**

```
global MODEL g1 c1 MODEL1
global OPTU R1 FR
global R2 R3 MODEL1 MODEL2
global R4 R5 R6 R7 R8 R9 R10 U DU
global MY1 MY2 MY3 DU0 YESKI
global MY4 MY5 MY6 MY7 MY8 MY9 MY10 KSAYI U  options
global IW11 LW11 LW21 LW31 LW41 IW12 LW12 LW22 LW32
global COST YESKI FR
global KSAYI
global cikti MODEL1
global MODEL YM22 eskistate eskiIN

for m=1:800,
  for n=1:4,
     gir(m,n)=0.25*state((5*m-1),n);
  end
cik(m,1)=0.25*acc(5*m,1);
cik(m,2)=0.25*acc(5*m,2);
gir(m,5)=0.05*uapp((5*m),1);
gir(m,6)=0.1*uapp((5*m),2);
end;
%yukarıdaki döngülerle ağın eğitimi için gerekli giriş ve çıkış verileri
oluşturulmaktadır.

MODEL=newff([-500 500;-500 500;-15 15;-15 15;-50 50;-20 20;],[6 6
2],{'purelin','tansig','tansig'},'traingd');
%Yapay sinir ağı 3 katmandan oluşmaktadır. Katmanlarda sırasıyla 6,6 ve 2 nöron
bulunmaktadır
MODEL.trainParam.show = 100;%Eğitim sırasında her 100 iterasyonda sonucu göster
```

```
MODEL.trainParam.lr = 0.05;%iterasyon katsayısı 0.05
MODEL.trainParam.epochs = 50000;
MODEL.trainParam.goal = 1e-5;%eğitim amacı
c1=cik';%yapay sinir ağının eğitimi için gerekli giriş değerleri
g1=gir';%yapay sinir ağının eğitimi için gerekli hedef değerleri
[MODEL,tr1]=train(MODEL,g1,c1);
MODEL.adaptParam.passes=1;%adaptasyon sırasında ağdan 1 defa geçirillecektir
MODEL1=MODEL;
save MODELgp;
```

**FOPT.M**

```
function OPTU=FOPT(softinput)

global OPTU R1 FR
global R2 R3 MODEL MODEL2
global R4 R5 R6 R7 R8 R9 R10 U DU
global MY1 MY2 MY3 DU0 YESKI
global MY4 MY5 MY6 MY7 MY8 MY9 MY10 KSAYI U  options
global IW LW1 LW2 LW3 b1 b2 b3 b4

%Bu fonksiyon ile maliyet fonksiyonunu minmize edecek tork değerleri
hesaplanmaktadır

%Ri t+i zamanındaki referans değerdir
R1=softinput(1:4);
R2=softinput(5:8);
R3=softinput(9:12);
R4=softinput(13:16);

t=softinput(21);
%İterasyon hesaplamalarını kolaylaştırmak için MATLAB'ta hazır bulunan sim
fonksiyonun yerine modelout ve
%aşağıdaki matrisler oluşturulmuştur
IW=MODEL.IW{1,1};
LW1=MODEL.LW{2,1};
LW2=MODEL.LW{3,2};
b1=MODEL.b{1,1};
b2=MODEL.b{2,1};
b3=MODEL.b{3,1};


if t<0.003
  KSAYI=0.008;%maliyet fonksiyonunda yer alan DU değerlerinin katsayısıdır
    options=optimset(options,'TolX',1.0e-10);
    options=optimset(options,'Display','none');
    options=optimset(options,'Tolfun',1.0e-10);
    options=optimset(options,'Maxiter',1000);
```

```
    options=optimset(options,'MaxFunEvals',1000);
  U=[0;0];%Başlangıç tork değeri
  YESKI = [0;0;0;0];%Önceki çıkış değeri
  DU0=[0;0;0;0;0;0];%İlk U torklarından değişim değerleri
end
exitflag=0;
while exitflag == 0
  [DU,fval,exitflag]= fmincon('COSTF',DU0,[],[],[],[],[-1;-1;-1;-
1],[1;1;1;1],[],options);%Matlab'ta hazır bulunan
  %fmincon fonksiyonu ile optimizasyon yapılmaktadır
  U=U+DU(1:2);
end
YESKI=softinput(17:20);
OPTU = U;
```

**FMATRIS.M**

```
function YM22 = FMATRIS(invector)

global MODEL MODEL1 YM22 eskistate eskiIN ONCEKIIN dT

%Fonksiyonun amacı model olan yapay sinir ağının çıkış değerlerini oluşturmak ve
yapay sinir ağını adapte etmektir
IN=invector;%Yapay sinir ağının giriş değerleridir
YM22=modelout3(IN);%YM22 Yapay sinir ağının çıkışıdır

 zaman=invector(7);
 if zaman>0.003
  %aşağıdaki katsayılarla yapay sinir ağının giriş ve çıkış değerleri
ağırlıklandırılmaktadır
  P(1,1)=0.25*ONCEKIIN(1,1);
    P(2,1)=0.25*ONCEKIIN(2,1);
    P(3,1)=0.25*ONCEKIIN(3,1);
    P(4,1)=0.25*ONCEKIIN(4,1);
    P(5,1)=0.05*ONCEKIIN(5,1);
  P(6,1)=0.1*ONCEKIIN(6,1);
  T(1,1)=(invector(3,1)-ONCEKIIN(3,1))/dT*0.25;
  T(2,1)=(invector(4,1)-ONCEKIIN(4,1))/dT*0.25;
  [MODEL,Y,E] = adapt(MODEL,P,T);%adaptasyon
end
ONCEKIIN=IN;%yapay sinir ağının adaptasyonu için kullanılan bir adım önceki giriş
değeri
```

**COSTF.M**

```
function COST = COSTF(softinput)
%Maliyet fonksiyonu
%DUi değerleri t+i zamanındaki U tork değerlerindeki değişimi göstermektedir
```

```
%Ri t+i zamanındaki referans değerdir
%MYi t+i zamanındaki sistemin model tarafından tahmin edilen değeridir

global COST R1 R2 R3 R4
global MY1 MY2 MY3 U YESKI FR
global MY4 KSAYI

DU1=softinput(1:2);
DU2=softinput(3:4);
MY1 = modelout3([YESKI;(U+DU1)]);
MY2 = modelout3([MY1;(U+DU1+DU2)]);
MY3 = modelout3([MY2;(U+DU1+DU2)]);
MY4 = modelout3([MY3;(U+DU1+DU2)]);
COST =([R1-MY1]'*[R1-MY1]+[R2-MY2]'*[R2-MY2]+[R3-MY3]'*[R3-MY3]+[R4-
MY4]'*[R4-MY4])+KSAYI*((DU1)'*(DU1)+(DU2)'*(DU2));
```

**FBIR.M**

```
function BU = FBIR(softinput)

global DIC Y1 P T
global OPTU R1 FR
global R2 R3 MODEL MODEL2
global R4 R5 R6 R7 R8 R9 R10 U DU
global MY1 MY2 MY3 DU0 YESKI BU
global MY4 MY5 MY6 MY7 MY8 MY9 MY10 KSAYI U  options
global IW LW1 LW2 LW3 b1 b2 b3 b4

%Bu fonksiyon ile öngörülü denetim ile direk ters modelleme arasında anahtarlama
yapılarak tork değerleri bulunmaktadır

IN=softinput(22:25);
P=softinput(26:29);
T=[0.05*softinput(30);0.1*softinput(31)];
[DIC,Y,E] = ADAPT(DIC,P,T);

YESKI=softinput(17:20);

%Ri t+i zamanındaki referans değerdir
R1=softinput(1:4);
R2=softinput(5:8);
R3=softinput(9:12);
R4=softinput(13:16);

t=softinput(21);

%İterasyon hesaplamalarını kolaylaştırmak için MATLAB'ta hazır bulunan sim
fonksiyonun yerine modelout ve
```

```
%aşağıdaki matrisler oluşturulmuştur
IW=MODEL.IW{1,1};
LW1=MODEL.LW{2,1};
LW2=MODEL.LW{3,2};
b1=MODEL.b{1,1};
b2=MODEL.b{2,1};
b3=MODEL.b{3,1};

%iki denetim yöntemi arasındaki anahtarlama için dfark değeri kullanılmaktadır
dfark=IN'*IN;

A=sim(DIC,IN)
BU1=[20*A(1,1);10*A(2,1)];
tcik=modelout3([YESKI;BU1]);
dfark1=(YESKI-tcik)'*(YESKI-tcik);%dfark1 değeri ise direk ters modelleme ile elde
edileceği tahmin edilen çıkış ile
%önceki durum vektörü arasındaki farktır
if t<0.003
  KSAYI=0.008;%maliyet fonksiyonunda yer alan DU değerlerinin katsayısıdır
    options=optimset(options,'TolX',1.0e-10);
    options=optimset(options,'Display','none');
    options=optimset(options,'Tolfun',1.0e-10);
```

## ÖZGEÇMİŞ

Adı Soyadı: Umut AKINCIOĞLU

Doğum Yeri: Kırklareli

Doğum Tarihi: 02.04.1977

Medeni Hali: Evli

Yabancı Dili: İngilizce

Eğitim Durumu (Kurum ve Yıl)

Lise : Kırklareli Anadolu Lisesi 1991-1995

Lisans : ODTÜ- Elektrik-Elektronik Mühendisliği 1995–1999

Çalıştığı Kurum/Kurumlar ve Yıl: ASELSAN AŞ. 1999---

Denker, A. Akıncıoğlu, U. 2006. Neural Adaptive Switching Control of Robotic Systems, World Enformatica Society, 14, 314-317

Denker, A. Akıncıoğlu, U. 2006. Robotik Sistemlerin Yapay Sinir Ağları ile Anahtarlamalı Uyarlamalı Denetimi, Otomatik Kontrol Ulusal Toplantısı 6-8 Kasım 2006 TOBB ETÜ, Ankara. 340-345