T.C.
SELÇUK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRONİK TARTIM SİSTEMLERİNDE ÇEVRESEL VE MEKANİK HATA ETKİLERİNİN MİNİMİZE EDİLMESİ İÇİN YAPAY SİNİR AĞLARINA DAYALI YENİ BİR İNDİKATÖR TASARIMI

H. İbrahim ENEÇ

YÜKSEK LİSANS TEZİ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

KONYA, 2009

ÖZET

YÜKSEK LİSANS TEZİ

ELEKTRONİK TARTIM SİSTEMLERİNDE ÇEVRESEL VE MEKANİK HATA ETKİLERİNİN MİNİMİZE EDİLMESİ İÇİN YAPAY SİNİR AĞLARINA DAYALI YENİ BİR İNDİKATÖR TASARIMI

H. İbrahim ENEÇ

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Elektrik Elektronik Mühendisliği Anabilim Dalı

Danışman : Yrd. Doç. Dr. Sinan GÜLTEKİN

2009, 112 Sayfa

Juri : Yrd. Doç. Dr. Mehmet CUNKAŞ
Yrd. Doç. Dr. Ömer AYDOĞDU
Yrd. Doç. Dr. S. Sinan GÜLTEKİN

Bu çalışmada endüstriyel ölçüm sistemi olan kantarlar ele alınmış, çalışma mantıkları ve sistem bileşenleri incelenmiştir. Sistem yapı elemanlarının performans üzerindeki etkileri göz önünde bulundurularak yapay sinir ağlarına dayalı yeni bir tasarım yapılmıştır.

Teorik olarak her parametresi hesaplanarak tasarlanmış sistemler proje açısından kusursuz sayılsada, uygulama safhasında, üretim, kurulum, malzeme tedariği vb. gibi etkenler yüzünden sorunlarla karşılaşılaşılmaktadır. Bu sorunlardan bazıları, üretimin tekrarlanmasını gerektirebilir. Kantarlar için, özellikle mekanik sistemden kaynaklı hatalar göz önünde bulundurularak tasarlanan yeni sistemde, bu hataların etkileri minimize edilmeye çalışılmıştır.

Sistemin temel bileşeni yük hücrelerinin, standart uygulamalarda birbirlerine bağımlı çalıştıkları bilinmektedir. Uygulamada, yük hücreleri bağımsız ölçülerek, ölçüm sonuçları ile bir yapay sinir ağı modeli oluşturulmuş, tartım sonucu, oluşturulan modelden elde edilmeye çalışılmıştır. Elde edilen sonuçlar performans kriterleri açısından değerlendirilerek, tasarlanan sistemin standart uygalama ile karşılaştırması yapılmıştır.

Anahtar Kelimeler: Yapay Sinir Ağları, Elektronik Ölçüm Sistemleri, İndikatör, Yük Hücresi

ABSTRACT

Master Thesis

A NEW INDICATOR DESIGN BASED ON ARTIFICAL NEURAL NETWORKS TO MINIMIZE THE ENVIRONMENTAL AND MECHANICAL FAULT EFFECTS IN ELECTRONIC WEIGHING SYSTEMS

H. İbrahim ENEÇ

Selçuk University
Graduate School of Natural and Applied Sciences
Electric and Electronics Engineering Branch

Advisor: Assist. Prof. Dr. Sinan GÜLTEKİN

2009, 112 Page

Jury : Assist. Prof. Dr. Mehmet CUNKAŞ
Assist. Prof. Dr. Ömer AYDOĞDU
Assist. Prof. Dr. S. Sinan GÜLTEKİN

In this study, industrial weighing systems, scales are handled, and their working reasonings and system components are investigated. Taking into account system elements affects on performance, a new design is made based on artifical neural networks.

Theoreticallay, for all the faultless designs made by considering all the parameters, in application, there occures problems due to manufacturing, establisment, component procuring and so forth factors. Some of these problems may cause the system to be manufactured again. In new design, these problems studied to make minimal, esipecially problems that occur from the mechanical system.

It is known that, in standart applications, load cells, the systems base components, are working related to each other. In this application, load cells are measured seperately, a neural network is established with the conversion results and weigh result is tried to be obtained from the network. Lastly, the network's results are considerd according to performance criterias, and the new desing is compared to standart applications.

Key Words: Artifical Neural Networks, Electronic Weighing Systems, Indicator, Load Cell.

TEŞEKKÜR

Tez konumun belirlenmesinde ve çalışmalarımın her aşamasında yardımlarını esirgemeyen, değerli fikir ve katkıları ile bu çalışmaya ışık tutan danışman hocam, Yrd. Doç. Dr. S. Sinan Gültekin'e teşekkürü bir borç bilirim.

H. İbrahim ENEÇ

İÇİNDEKİLER

ÖZET	I
ABSTRACT	II
TEŞEKKÜR	III
İÇİNDEKİLER	IV
SİMGELER	VII
ŞEKİLLER	VIII
TABLOLAR	X
1. GİRİŞ	1
2. GERİLMEYE DAYALI AĞIRLIK ÖLÇÜMÜ	4
2.1. Wheatstone Köprüsü	9
2.2. Wheatstone Köprüsü Çeşitleri	11
2.2.1. Çeyrek wheatstone köprüsü	11
2.2.2. Yarım wheatstone köprüsü	13
2.2.3. Tam wheatstone köprüsü	15
2.3. Kantar Kurulum Prosesi ve Kalibrasyon	16
3. YAPAY SİNİR AĞLARI	23
3.1. Yapay Sinir Ağlarına Giriş	23
3.1.1. Yapay sinir ağlarının genel özellikleri	23
3.1.2. Yapay sinir ağlarının yetenekleri	24
3.1.3. Yapay sinir ağlarının kullanıldığı alanlar	26
3.1.4. Yapay sinir ağlarının avantajları	27
3.1.5. Yapay sinir ağlarının dezavantajları	28
3.1.6. Yapay sinir ağlarının tarihi	30
3.1.6.1. Yapay sinir ağlarının tarihi – 1970 öncesi çalışmalar	30
3.1.6.2. XOR problemini bu kadar özel kılan nedir?	32
3.1.6.3.Yapay sinir ağlarının tarihi – 1970 sonları çalışmalar	34
3.1.7. Yapay Sinir Ağlarının Sınıflandırılması	36

3.1.7.1. Öğretmenli algoritmalar 39
3.1.7.1.a. İleribeslemeli modeler 39
3.1.7.1.b. Geribeslemeli modeler 40
3.1.7.1.c. Yarışa dayalı modeler 40
3.1.7.2. Öğretmensiz algoritmalar 41
3.1.7.2.a. Yarışa dayalı modeler 41
3.1.7.2.b. Boyut azaltmaya dayalı modeler 42
3.1.7.2.c. Çağrışımlı öğrenmeye dayalı modeler 42
3.2. Yapay Sinir Ağlarının Yapısı 42
3.2.1. Teknoloji ve biyolojik yapılar 42
3.2.2. Biyolojik sinir ağlarının yapısı 44
3.2.3. Yapay sinir hücresinin yapısı 46
3.2.4. Yapay sinir ağının yapısı 50
3.2.5. Farklı modellerin ortaya çıkmasını sağlayan unsurlar 52
3.3. Tek Katmanlı Algılayıcılar 53
3.3.1. Tek katmanlı algılayıcıların ortak özellikleri 53
3.3.2. En basit tek katmanlı algılayıcı 55
3.3.3. ADALINE modeli 57
3.3.4. Perceptron ve ADALINE modellerinin karşılaştırılması 60
3.4. Çok Katmanlı Yapay Sinir Ağları 66
3.4.1. Çok ktamanlı yapay sinir ağlarına olan gereksinim 66
3.4.2. Çok katmanlı yapay sinir ağlarının yapısı 67
3.4.3. Çok katmanlı yapay sinir ağlarının öğrenme kuralı 68
3.4.4 Tasarımda karşılaşılabilecek problemler 78
4. UYGULAMA 80
4.1. Klasik Sistemlerin Dezavantajları 80
4.2. Tasarım 82
4.2.1. Donanım 83
4.2.2. Donanım yazılımı 88
4.2.3. Yazılım 88

4.3. Simulasyon ve Sonuçlar 91
5. SONUÇ VE ÖNERİLER 109
6. KAYNAKLAR 111
EK – A Arayüz kaynak kodu

SİMGELER

σ : Basınç

F : Kuvvet

A : Alan

L : Boy

D : Yatay kesit

ε : Birim şekil değiştirme

v : Poison Oranı

k : Yay sabiti

x : Uzama miktarı

E : Elastiklik modülü

GF : Gauge Faktörü

R : İç direnç

V : Gerilim

a : Eğim

NET : Ağırlıklı toplam fonksiyonu

x : Giriş

w : Ağırlık

y : Çıkış

η : Öğrenme katsayısı

ϕ : Eşik değer

ŞEKİLLER

Şekil 1 **:** Sistem blok diagramı
Şekil 2.1 **:** Çekme gerilmesi
Şekil 2.2 **:** Sıkıştırma gerilmesi
Şekil 2.3 **:** Wheatstone köprüsü
Şekil 2.4 **:** Çeyrek wheatstone köprüsü
Şekil 2.5 **:** Yarım Wheatstone Köprüsü
Şekil 2.6 **:** Yarım wheatstone köprüsünde ki gerilme ölçerlerin benzetimi
Şekil 2.7 **:** Tam wheatstone köprüsü
Şekil 2.8 **:** Kantar kurulumu için ön hazırlık çalışması
Şekil 2.9 **:** Kantar iskeleti ve yük hücresi yatakları
Şekil 2.10 **:** "Junction box" Birleştirme Kutusu
Şekil 2.11 **:** Kalibrasyon için yük dağılımı
Şekil 3.1 **:** İki boyutlu öklid uzayında (χ_1, χ_2) ikililerine karşılık gelen sembollerin dağılımı
Şekil.3.2 **:** Lojik VE fonksiyonuna ait doğruluk tablosu ve geometrik yorumu
Şekil.3.3 **:** Lojik ÖZEL VEYA fonksiyonuna ait doğruluk tablosu ve geometrik yorumu
Şekil.3.4 **:** 2-girişli 2-çıkışlı çok katmanlı ileribeslemeli ağ yapısı
Şekil.3.5 **:** 2-girişli 2-çıkışlı Elman-tipi geribeslemeli ağ yapısı
Şekil 3.6 **:** Sinir ağlarının temel birimi olan sinir hücresinin (nöronun) şematik yapısı
Şekil 3.7 **:** Vücudumuzdaki sinir ağının küçük bir kısmı
Şekil 3.8 **:** Yapay sinir hücresinin yapısı
Şekil 3.9 **:** Yapay sinir ağının yapısı
Şekil 3.10 **:** Tek katmanlı yapay sinir ağı modeli
Şekil 3.11 **:** İki-girişli bir sistem için sınıf ayracı
Şekil 3.12 **:** En basit tek katmanlı algılayıcı modeli (perceptron)
Şekil 3.13 **:** Aktivasyon fonksyonu sigmoidal olan bir ADALINE modeli

Şekil 3.14 : ADALINE modelinin ayrıntılı yapısı [7]
Şekil 3.15 : İleribeslemeli çok katmanlı yapay sinir ağı
Şekil 3.16 : İleribeslemeli çok katmanlı yapay sinir ağının ayrıntılı yapısı
Şekil 4.1 : Hatalı yapılmış montaj
Şekil 4.2 : Eğimli monte edilmiş yük hücresi
Şekil 4.3 : Ana kart görünümü
Şekil 4.4 : Modül kart görünümü
Şekil 4.5 : Arayüz programı genel görünümü
Şekil 4.6 : Arayüz programı rapor görünümü
Şekil 4.7 : Hazırlanan ölçüm düzeneği bloğu
Şekil 4.8 : Platform hataları
Şekil 4.9 : Ölçüm işlemi
Şekil 4.10 : Sistem ölü ağırlığının ölçümü
Şekil 4.11 : 20 kg'lık yük ile yapılan örnek ölçüm
Şekil 4.12a : Model ağ yapısı
Şekil 4.12b : Ağ ölçüm hatası
Şekil 4.13 : Tartım işleminin başlatılması
Şekil 4.14 : Merkez nokta deneme ölçümü
Şekil 4.15 : 1. Köşe deneme ölçümü
Şekil 4.16 : 2. Köşe deneme ölçümü
Şekil 4.17 : 3. Köşe deneme ölçümü
Şekil 4.18 : 4. Köşe deneme ölçümü
Şekil 4.19 : 1. rastlantısal ölçüm denemesi
Şekil 4.20 : 2. rastlantısal ölçüm denemesi
Şekil 4.21 : 3. rastlantısal ölçüm denemesi

TABLOLAR

Tablo 3.1 : Yapay sinir hücrelerinde kullanılan toplama fonksiyonları

Tablo 3.2 : Yapay sinir hücrelerinde kullanılan aktivasyon fonksiyonları

Tablo 3.3 : Giriş matrisi ve hedef vektörü

Tablo 3.4 : Yapay sinir ağlarında kullanılan toplama fonksyonları

Tablo 3.5 : Yapay sinir ağlarında kullanılan aktivasyon fonksiyonları

1. GİRİŞ

Eski çağlardan bu tarafa farklı ihityaçlar için ağırlık ölçümleri yapılagelmiş, zaman içerisinde bu ölçümler önemini daha da artırarak, endüstrinin birçok alanında, üretim, kalite kontrol, depolama gibi kritik öneme sahip uygulamalarda vazgeçilmez bir proses halini almışlardır. Bu amaçla, günümüze kadar bir çok tartım sistemi geliştirilmiştir. İlk kullanılan tartım sistemleri tamamiyle mekanik olup, sistem dengesi yada kullanılan yayda meydana gelen gerilme gibi esaslara dayalı ölçüm yapan sistemlerdi. Ağırlık sıfır iken dengede duran bir kaldıraç koluna dik olarak tutturulmuş bir ibrenin sapmasıyla yapılan ölçüm, bu sistemlere bir örnek olarak verilebilir. Gelişen teknoloji ve hassas tartım ihtiyacının artmasına paralel, tartım sistemleride gelişmiş, günümüzde elektronik destekli üretilen sistemler haline gelmişlerdir.

Ölçüm amaçlı kullanılan bu sistemler, özellik, hassasiyet ve kapasitelerine göre, terazi, baskül, kantar ve benzeri isimlerle anılmaktadırlar. Kantarlar, tonlarla ifade edilen yüklerin tartımında kullanılan sistemler olup, çalışmamıza konu ölçüm sistemi olacaktır. Kantarların çalışma prensipleri, metal bir cisme uygulanan baskı sonucu, cismin boyutlarında meydana gelen değişimin ölçülmesine dayanmaktadır. Bir kantar, load cell diye bilinen yük hücreleri üzerine yerleştirilmiş, ölçülecek ağırlıkların uygulandığı mekanik bir iskelet ve bu ağırlıklar vasıtasıyla yük hücrelerinde oluşan fiziksel değişimin anlamlı hale getirildiği indikatör tabir edilen elektronik bir aksamdan oluşmaktadır. İndikatör analog/dijital çevrim prosesine dayalı elektronik bir cihazdır.

Sistemin performansını etkileyen bir çok parametre sayılabilir. Çözünürlük, ölçüm kararlılığı, uygulanan yüklerin eş dağılım oranlarına bağlı sistem cevabı, sıcaklık, nem ve fiziksel yapıdaki mekanik hatalar gibi çevre koşullarına karşı bağımlılık bu

parametrelerden bazılarıdır. Sistem parametrelerinin ve ölçüm sonuçlarının iyileştirilmesine yönelik yapılan çalışmalar olmasına rağmen, bu çalışmaların çoğu, yük hücresi sinyalleri üzerinde yoğunlaşmıştır. Andrew Gilman ve Donald G. Bailey, yük hücresi sinyalleri üzerine yaptıkları çalışmada, Dirac delta fonksiyonu ile darbe cevabının kestirimini elde etmeye, böylece yük hücresinin tepkisini sinyal kararlı hale gelmeden belirleyerek hızlı ölçüm almaya çalışmışlardır. Diğer bir çalışma ise Hernadez W. tarafından, ölçüm sinyali ile gürültü sinyalinin birbirinden ayrılabilmesi için, adaptif filtre tasarımı üzerine yapılmıştır. Çalışmada, RLS (recursive least-squares – özyineli en düşük kareler) algoritması temel alınarak adaptif bir filtre tasarlanmıştır. Yapılan deneyler sonucu, sinyal – gürültü oranında, 27 dB bir kazanç elde edilmiştir. Bu projede yapılacak çalışma ise sistemin, yük hücreleri parametrelerine, mekanik hatalara ve yükün sistem üzerinde ki dağılımına bağımlılığını azaltma yönünde olacaktır.

Yapılacak çalışmada, tartım sonucu doğruluğunun yük hücreleri farklılıkları, çevresel mekanik faktörler ve olası montaj hatalarından bağımsız kılınabilmesi amaçlanmaktadır. Bu nedenle, her bir yük hücresinin ağırlaştırılmış bir çarpım katsayısı ile lineer olmayan ayrık ölçümlerinin yapılması düşünülmüştür. Yapılacak ölçümler için yük hücreleri çarpım katsayıları, Yapay Sinir Ağı algoritmaları ile tespit edilecektir. Bu katsayılar, her bir yük hücresi için farklı değerlerde olacak, toplam sistem cevabı üzerinde yük hücrelerinin efektif değerleri, çevre faktörleri ve kurulum hataları göz önünde bulundurularak belirlenecektir. Bu sayede de tartım sonucunun uygun olmayan çevresel koşullarda dahi lineer olması ve sistemin kurulum işlemlerinin basitleştirilmesine çalışılacaktır.

Uygulamada, sistem ölçümleri için her bir yük hücresinin ayrı ölçümlerini sağlayacak yeni bir elektronik indikatör tasarlanacaktır. Daha sonra bu indikatör

sayesinde, rasgele kurulu bir sistem için bilinen ağırlık değerlerinin ölçümleri bilgisayar ortamına aktarılacaktır. Bilgisayarda depolanan ağırlık değerlerine karşın alınan çevrim sonuçları, Yapay Sinir Ağı Algoritmaları ile irdelenip, yük hücreleri için gerekli çarpım katsayıları elde edilmeye çalışılacaktır. Bu parametreler doğrultusunda bilinmeyen yükler için sistem ölçüm sonuçları elde edilerek değerlendirilecektir. Yapılacak çalışmaya ilişkin blok diagram Şekil 1'de görülmektedir.



Şekil 1 Sistem blok diagramı

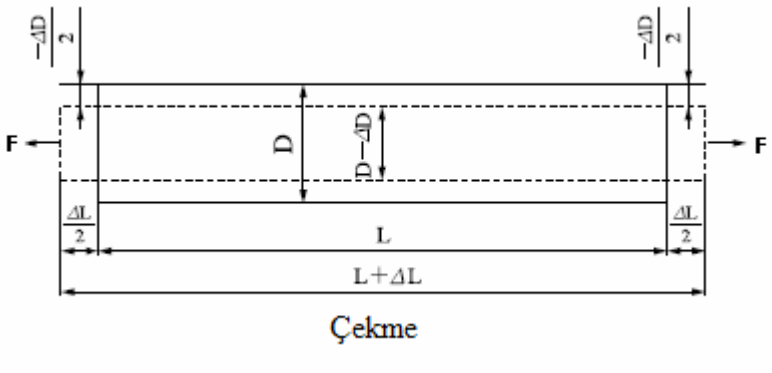
Elektronik tartım sistemlerinde, ölçüm için sistemin temel bileşenlerinden olan elektronik indikatör (çevirici katman), sistem hassasiyetinde temel parametre olarak görülmekte, yalnız çoğu sistem, mekanik ve çevresel sorunlar nedeni ile gerekli performansı sağlayamamaktadır. Bu nedenle ürtilen her bir sistem için çevre koşullarına uyum açısından, kurulum yeri alt yapı çalışmaları yapılmakta, mekanik düzenin terazisi ve denge moment ayarı nedeniyle sistemden gerekli performansın elde edilmesi için sistem üzerine müdahaleler de bulunulmaktadır. Bu da, sistemin kurulumunu maliyetli ve zahmetli kılmakta, sistem komponentleri ne kadar hassas olursa olsun, tartım sonuçlarının belirli bir marjda yaklaşım yapılarak elde edilmesine sebep olmaktadır. Bu çalışma ile, bu tür olası hataların elektroniksel olarak giderilmeye çalışılması, hem sistem kurulum maliyeti, hemde sistem tartım sonucunun iyileştirilmesi açısından büyük önem kazanmaktadır.

2. GERİLMEYE DAYALI AĞIRLIK ÖLÇÜMÜ

Hareketsiz duran bir cisme, F kuvveti uygulanacak olursa cisim üzerinde basınç ve gerilme meydana gelir. Basınç, cisme uygulanan F kuvvetine, cismin karşı koyan iç kuvvetleri sonucu oluşmaktadır. Gerilme ise cismin şekil değiştirmesi olarak tanımlanır. Dış etkilere karsı koyan iç kuvvetlerin uniform dağılıma sahip olduğu düşünülürse, basınç birim alana uygulanan kuvvet olarak tanımlanabilir (Inan 1988).

$$\sigma = F / A \tag{2.1}$$

Çekme kuvvetinin yarattığı basınçla orantılı olarak cismin yatay kesiti ΔD kadar küçülmekte ve boyu çekme kuvveti uygulanmadan önceki boyuna göre ΔL kadar uzamaktadır (Beer ve Johnston 1996).

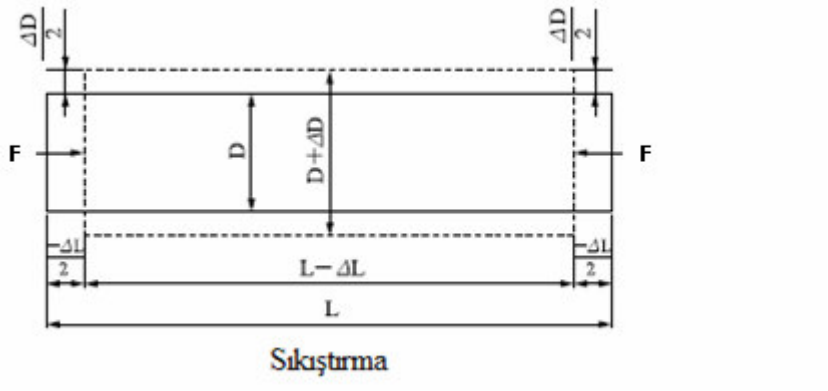


Şekil 2.1 Çekme gerilmesi

Uzama miktarının ilk boya oranına çekme gerilmesi adı verilir.

$$\varepsilon = \Delta L / L \tag{2.2}$$

Eğer cisim bir sıkıştırma kuvveti altında kalacak olursa, cismin yatay kesiti ΔD kadar büyümekte ve boyu ilk boya göre ΔL kadar kısalmaktadır.



Şekil 2.2 Sıkıştırma gerilmesi

Bu durumda ise cismin kısalma miktarının ilk boya oranına sıkıştırma gerilmesi denilmektedir (Özbek 1978).

$$\varepsilon = -\Delta L / L \tag{2.3}$$

Ayrıca kuvvetin uygulandığı eksen yönündeki gerilme, boyuna gerilme (2.4), bu ekseni kesen eksendeki gerilme, enine gerilme (2.5), enine gerilme ve boyuna gerilme arasındaki oranın mutlak değeri ise Poisson Oranı (2.6) olarak bilinmektedir.

$$\varepsilon_1 = \pm \frac{\Delta L}{L} \tag{2.4}$$

$$\varepsilon_2 = \pm \frac{\Delta D}{D} \tag{2.5}$$

$$v = \left| \frac{\varepsilon_2}{\varepsilon_1} \right| \tag{2.6}$$

Maddelerin Poisson Oranları, ağırlık ölçme uygulamalarında ve inşaat sektöründe kritik öneme sahip birer parametre olarak dikkate alınıp kullanılmaktadır (Bakioglu 2001).

Elastiklik sınırları aşılmadığı takdirde, cisme uygulanan kuvvetle cismin şekil değiştirmesi arasında doğrusal bir orantı olacaktır. Eğer elastiklik sınırı aşılmaz ve cisim üzerine uygulanan kuvvet kaldırılırsa, ilk boyutlara geri dönülür. Bu yasa İngiliz Fizikçi Robert Hooke tarafından 1676 yılında keşfedilmiştir. Örneğin düşey eksende asılı duran bir yaya bir ağırlık iliştirilerek kuvvet uygulanacak olursa, yayda meydana gelen fiziksel değişme uygulanan kuvvetle doğru orantılı olacaktır. Ağırlık kaldırıldığında yay ilk haline geri dönüyorsa, yay o kuvvet altında elastiktir denilir. Elastiklik sınırı, bir cismin elastikliğini kaybetmeksizin üzerine uygulanabilecek en büyük kuvvet olarak tanımlanır. Hooke yasası (2.7)'de ki matematiksel formülle ifade edilebilir.

$$F = kx \tag{2.7}$$

F: Cisim üzerine uygulanan kuvvet
k: Yay sabiti
***x*:** Uzama miktarı

Diğer bir deyişle Hooke Yasası gerilme ile şekil değiştirme arasındaki ilişki olarak bilinmekte ve şu şekilde de ifade edilmektedir (Hibbeler 1998).

$$\sigma = E\varepsilon \tag{2.8}$$

σ **:** Basınç kuvveti

E: Elastiklik modülü

ε: Birim şekil değiştirme

Formülde görülen elastiklik modülü, maddenin mukavemetinin bir ölçüsüdür ve birimi N/m²'dir.

İlk defa 1856'da Lord Kelvin mekanik bir gerilmeye maruz bırakılan metal iletkenlerin (yaptığı deneylerde bakır ve çelik tellere gerilme uyguladığında) elektriksel dirençlerinde bir değişim olduğunu saptamıştır. Diğer bir saptaması, aynı gerilme altında demirin direncinin bakırın direncine göre daha fazla değiştiğidir. Ancak bu bilginin pratiğe dönüştürülebilmesi 1930'ların sonlarını bulmuştur. 1938 yılında Edward E. Simmons ve Arthur C. Ruge, birbirlerinden habersiz olarak gerilme ölçer rozetlerini bulmuşlardır.

Günümüzde kullanılan birçok gerilime dayalı ölçme sistemlerinde, bu mantıkla oluşturulmuş ölçüm mekanizmaları kullanılmakta ve bu mekanizmalar yük hücresi olarak adlandırılmaktadır.

Temelde tüm yük hücreleri mekanik hareketi, elektriksel işarete çevirmeye yönelik dizayn edilmiş, belirli esneme değerlerine sahip olacak şekilde üretilmiş metal alaşımlardır. Metalin iç kapasitans ve endüktansı veya direnci, burkulmayla orantılı bir büyüklük üretir. Gerilmeye maruz bırakılan bir telin, çapının daralmasıyla beraber boyunda uzama meydana gelmekte, bu da telin iç direncini değiştirerek gerilme algılamasını sağlamaktadır. Tüm gerilmeler için Gauge Faktör denilen bir gerilim hassasiyeti faktöründen bahsedilmekte ve aşağıdaki eşitlikle hesaplanmaktadır.

$$GF = \left(\frac{\Delta R}{R}\right) / \left(\frac{\Delta L}{L}\right) \tag{2.9}$$

İdeal tip yük hücresi, yalnızca dış etkiyle meydana gelen zorlama sonucu hücredeki şekil değişikliğini elektriksel dirence çevirmektedir. Bununla beraber

uygulamalarda çevre sıcaklığı, malzemenin imalat kusurları, yük hücresinin yüzeye olan yapışma miktarı ve metalin esnekliğe karşı fiziksel stabilitesi, elektriksel direncinin değişimini etkilemektedir. Ayrıca farklı materyallerin de aynı yöndeki gerilimlere karşı tepkileri de farklı olacaktır. Bu nedenle saf metaller yerine yük hücresi yapımında uygun esneklik ve lineer gerilme özelliklerine sahip materyaller elde etmek için çeşitli alaşımlar kullanılmaktadır. Yük hücresini oluşturan bu metal alaşımlardan bazıları Bakır-Nikel, Nikel-Krom, Platinyum-Nikel ve Nikel-Demir alaşımlarıdır.

Yük hücrelerinin kuvvet altındaki deformasyonu mekanik, optik, akustik, pnomatik ve elektriksel olarak ölçülebilmektedir. İlk imal edilen yük hücreleri gerilmenin boyda yarattığı uzamayı, orjinal boyu ile karşılaştırarak ölçen mekanik dizaynlı aletlerdi. Örneğin hala kullanılan uzunluk ölçüm metresi gerilmeyi ölçmede kullanılabilmektedir. Ancak bu tip aletlerin hassasiyetleri kaba ve kullanımları zordur.

Optik tabanlı ölçüm cihazları çok doğru ve hassas ölçüm yapabilirler. Fakat çok sık arızalanmaları ve devamlı bakım zorunluluğu, onlarında endüstride kullanımlarını popüler kılmamaktadır. Çalışma mantıkları, bir optik ışın kaynağında üretilen ışın demetlerinin, cisimdeki gerilmenin yarattığı uzamaya bağlı olarak bir ayna üzerinden açısal yansımalarının yoğunluk değişiminin algılanmasına dayanmaktadır. Optik cihazların ideal olarak kullanım yerleri laboratuarlardır.

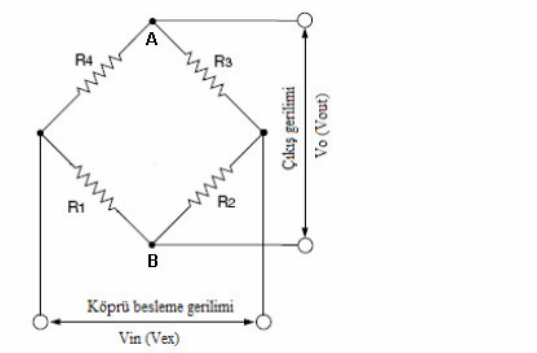
Kapasitif ve endüktif tip yük hücreleri ise imal edilip kullanılmaktaysa da, vibrasyona karşı zayıf kalmaları ve bu nedenden kaynaklanan hataların azaltılması için kompleks montaj elemanlarının kullanılma zorunluluğu, bu tip cihazların kullanımını engellemektedir.

Materyal gerilmesinin parametre olarak en uygun ölçülebildiği metot ise uygulanan yüke karşı elektriksel direnç değişimleri ile iletkenin boyundaki mikroskobik değişimin ölçülmesidir. Bu tip hücreler şu an en yaygın kullanım alanına sahip hücreler olup, yapısal olarak direnç bölücü bir sistem olan wheatstone köprülerinin türevlerinden oluşmaktadır.

Günümüz ölçüm sistemlerinde kullanılan yük hücrelerinin gerilim ölçüm metodları neredeyse tamamen bu prensibe dayanmakta, metal yüzey üzerine yerleştirilmiş Wheatstone Köprüsü ile imal edilmiş yük hücrelerinden oluşmaktadırlar. Bu noktada Wheatstone Köprüsü, yük hücreleri ile yapılan ağırlık ve benzeri ölçümlerin temelini oluşturan yapı elemanlarının ilk ve en kritik adımı olarak karşımıza çıkmaktadır.

2.1. Wheatstone Köprüsü

Gerilmeye bağlı direnç değişimleri çok küçük mertebelerde olduğu için, ölçümlerin yüksek doğrulukla yapılabilmesi amacıyla bazı devreler geliştirilmiştir. Bunlardan en çok kullanılanı 1843 yılında İngiliz fizikçi Charles Wheatstone tarafından önerilen ve kendi adıyla anılan Wheatstone köprüsüdür. Wheatstone köprüsü, iki gerilim bölücünün paralel bağlanmasından oluşmaktadır.(Şekil 2.3)



Şekil 2.3 Wheatstone köprüsü

Görüldüğü üzere kare şeklinde bağlanmış dirençlerin karşılıklı iki köşesine besleme gerilimi uygulanır, diğer karşılıklı iki köşenin arasındaki gerilim farkı ise çıkış gerilimidir. Çıkış gerilimi, A ve B noktalarının gerilimleri arasındaki farka eşit olduğuna göre A noktasının gerilimi (2.10),

$$V_A = \frac{R_3}{R_3 + R_4} V_{IN} \tag{2.10}$$

B noktasının gerilimi ise (2.11)

$$V_B = \frac{R_2}{R_2 + R_1} V_{IN} \tag{2.11}$$

formülleri ile verilebilir. Köprünün çıkış gerilimi ise (2.12)'deki formülle gösterilmektedir.

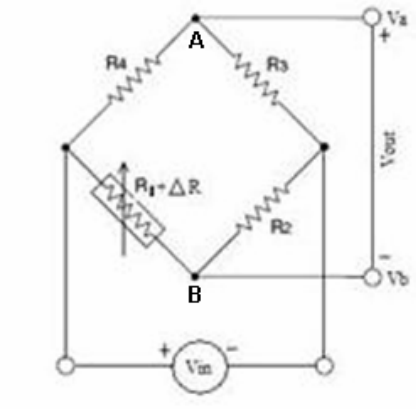
$$V_{OUT} = V_A - V_B = \left[\frac{R_1 R_3 - R_2 R_4}{(R_1 + R_2)(R_3 + R_4)} \right] V_{IN} \tag{2.12}$$

2.2. Wheatstone Köprüsü Çeşitleri

Wheatstone köprüsü, içinde bulundurduğu gerilme ölçer sayısına göre 3 çeşide ayrılmaktadır.

2.2.1. Çeyrek wheatstone köprüsü

Bu tip köprü, 1 adet gerilme ölçer ve 3 adet köprü tamamlayıcı dirençten oluşmaktadır.(Şekil 2.4)



Şekil 2.4 Çeyrek Wheatstone köprüsü

Anlaşılması açısından, yalnızca çeyrek köprünün giriş-çıkış bağıntısı hesaplanacaktır. R1 direncinin değerinin ΔR kadar değiştiğini (arttığını) varsayarsak ve R1+ΔR sonucu denklemde yerine yazılacak olursa (2.13)'deki sonuç elde edilmektedir.

$$V_{OUT} = \left[\frac{(R_1 + \Delta R)R_3 - R_2 R_4}{(R_1 + \Delta R + R_2)(R_3 + R_4)} \right] V_{IN} \tag{2.13}$$

Eğer tüm dirençler birbirine eşit seçilecek olursa, (R1, R2, R3, R4 = R), denklem (2.14)'teki gibi olur.

$$V_{OUT} = \left[\frac{R^2 + R\Delta R - R^2}{(2R + \Delta R)2R} \right] V_{IN} \tag{2.14}$$

(2R+ΔR) ifadesinde, ΔR değeri 2R değerine göre çok küçük olduğundan, (ΔR << 2R), ΔR ihmal edilebilir. Bu durumda gerekli sadeleştirmeler yapılırsa

$$V_{OUT} = \frac{1}{4} \frac{\Delta R}{R} V_{IN} \tag{2.15}$$

(2.15)'te verilen çıkış geriliminin, gerilme ölçer üzerinde meydana gelen direnç değişimiyle doğru orantılı olduğu görülmüş olur. Bu değişim de gerilmeyle doğru orantılı olduğundan, çıkış geriliminin de gerilmeyle orantılı olması beklenmektedir.

Böylece (2.15) eşitliği düzenlenebilir.

$$\frac{\Delta R}{R} = GF \frac{\Delta L}{L} = GF \cdot \varepsilon \tag{2.16}$$

(2.16), (2.15)'te yerine yazılırsa, giriş-çıkış bağıntısı daha anlaşılır hale getirilebilir.

$$V_{OUT} = \left(\frac{1}{4} GF \cdot \varepsilon\right) V_{IN} \tag{2.17}$$

Çeyrek köprü devresinde çıkış gerilimi gerilmeyle orantılı olarak değişmemektedir. Yapılan deneyler sonucunda ;

$$\frac{1}{1 + GF \cdot \varepsilon/2} \tag{2.18}$$

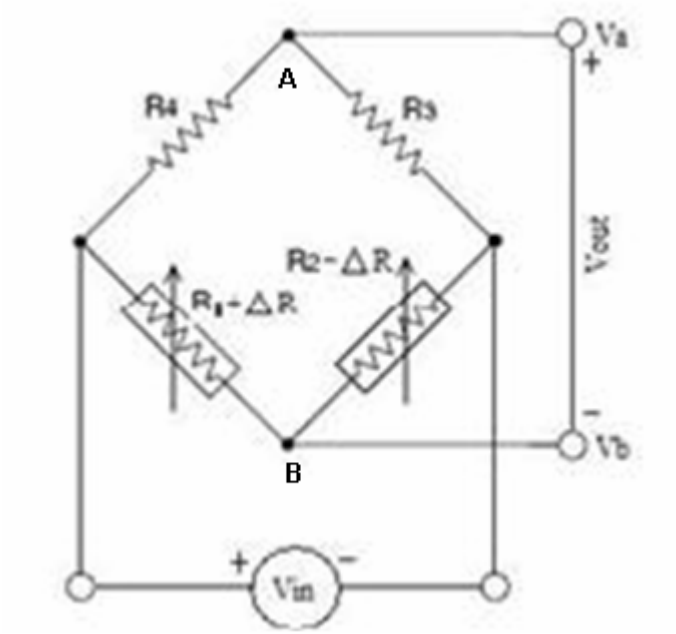
(2.18) ifadesinin, (2.17) denklemine çarpan olarak eklenmesi gerektiği belirlenmiştir. Pratikteki sonuç

$$V_{OUT} = \left(\frac{1}{4} GF \cdot \varepsilon\right)\left(\frac{1}{1 + GF \cdot \varepsilon/2}\right) V_{IN} \tag{2.19}$$

şeklindedir.

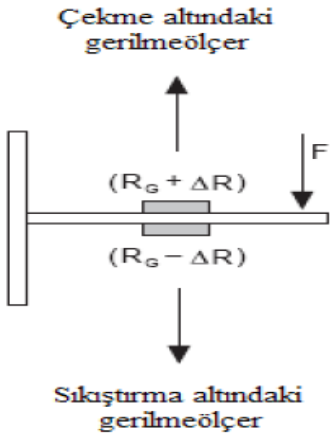
2.2.2. Yarım wheatstone köprüsü

Yarım Wheatstone köprüsü, 2 adet gerilme ölçer ve 2 adet köprü tamamlayıcı dirençten oluşmaktadır.(Şekil 2.5)



Şekil 2.5 Yarım Wheatstone köprüsü

Gerilme ölçerlerden birinin sıkıştırma etkisiyle direnci artarken, diğerinin çekme etkisiyle direnci azalmaktadır. Bu durumu Şekil 2.6'da ki gibi basitleştirerek göstermek mümkündür.



Şekil 2.6 Yarım Wheatstone köprüsünde ki gerilme ölçerlerin benzetimi

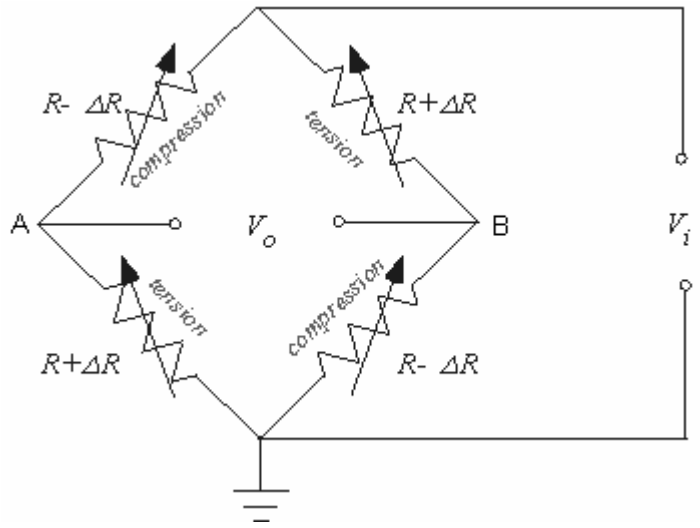
Yarım Wheatstone köprüsünün giriş-çıkış bağıntısı (2.20)'de verilmiştir.

$$V_{OUT} = \frac{GF \cdot \varepsilon}{2} V_{IN} \tag{2.20}$$

Yarım köprü, çeyrek köprüye göre çok daha doğrusal bir giriş-çıkış karakteristiği gösterir ve çıkış voltaj değeri aynı yük altında çeyrek köprünün yaklaşık 2 katı civarındadır.

2.2.3. Tam wheatstone köprüsü

Son olarak, tüm kollara gerilme ölçerlerin yerleştirilmesiyle tam köprü devresi elde edilmektedir. (Şekil 2.7) Tam köprü devresinin duyarlılığı çeyrek ve yarım köprülere göre çok daha iyidir. Bu yapıda bulunan gerilme ölçerlerden 2'si sıkıştırma, 2'si de çekme kuvveti altındadır.



Şekil 2.7 Tam Wheatstone köprüsü

Tam Wheatstone köprüsünün giriş-çıkış bağıntısı ise (2.21) ile ifade edilmektedir.

$$V_{OUT} = (GF \cdot \varepsilon) V_{IN} \tag{2.21}$$

2.3. Kantar Kurulum İşlemi ve Kalibrasyon

Kantarlar, üzerlerine uygulanan yüklerin meydana getirdigi gerilmeye dayalı ölçüm yapan sistemlerdir. Standart tip kantarlar, belirli bir projeye göre ayarlanmiş zemin üzerinde, yük hücresine uygun baskı gelecek şekilde kurulurlar.

Proses, öncelikle zeminin ayarlanması ve iskeletin montajı ile başlamaktadır. Olası zemin deformasyonu, arazi engebeleri ve ileride oluşabilecek toprak kayma ve çökmelerine karşı genelde kurulacak alan betonla doldurularak düz bir yüzey elde edilmeye çalışılır. Şekil 2.8'de kurulum için zemin projesi çalışması görülmektedir.



Şekil 2.8 Kantar kurulumu için ön hazırlık çalışması

Oluşturulan düz zemin üzerine kantar iskeleti yerleştirilir. İskelet ayakları, yük hücrelerinin yerleştirilebilmesi için özel olarak projelendirilip imal edilmişlerdir ve "yük hücresi yatağı" olarak adlandırılırlar.



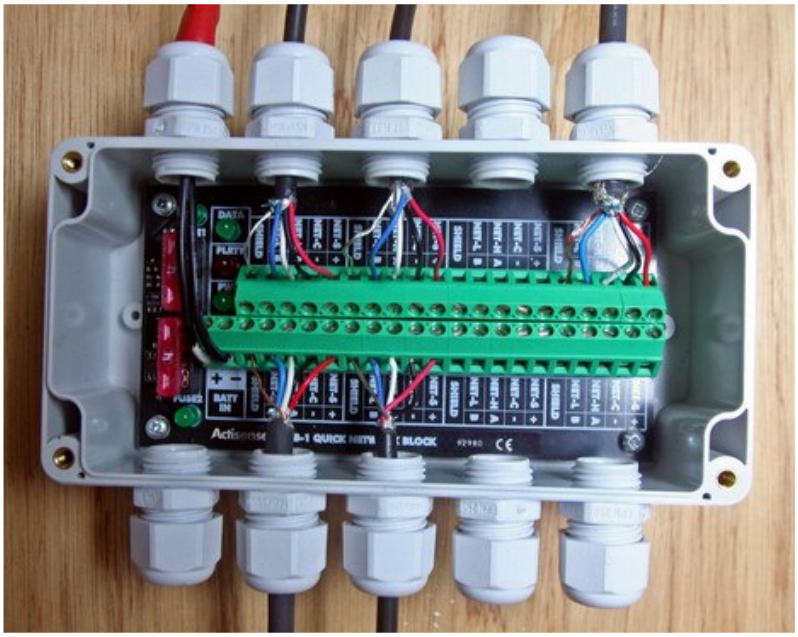
Şekil 2.9 Kantar iskeleti ve yük hücresi yatakları

Kapasite ve tonajına göre, belirli sayıda yük hücresi, montaj projesine uygun şekilde bahsedilen yataklara yerleştirilir. Kantar ebatına göre ayarlanmış bir yada birkaç platform, iskelet üzerine yerleştirilerek mekanik kurulum tamamlanmış olur. Bu sayede, gelen yüklerin uygulamış oldugu baskı, yük hücrelerinde gerilme meydana getirecektir.

Elektronik anlamda ölçüm, yük hücrelerinin gerilme oranına paralel elde edilen çıkış voltaj seviyelerinin uygun şekilde dönüştürülmesiyle gerçekleşmektedir. Yük hücreleri, tiplerine göre 5 – 15 Volt arası bir gerilim kaynağı ile beslenirler. Bu besleme gerilimi, yük hücreleri üzerine yerleştirilmiş gerilim bölücü sistem olan wheatstone köprülerini beslemektedir. Yük hücresi gerilime maruz kaldığında, lineer

oranda wheatstone köprüleride bir şekil değişikliğine maruz kalacak, boylarındaki değişime paralel direnç değerleri değişeceği için, çıkış voltaj seviyeleri de değişecektir.

Kurulan kantarın, kapasitesine göre birden fazla yük hücresi kullanıldığı için, tek bir çıkış değil, bir çok çıkış olacaktır. Çıkışlar yaygın uygulamada "junction box" olarak adlandırılan birleştirme kutusu içerisinde, trimpotlarla uygun seviyede direnç değerleri ayarlanarak birleştirilirler.



Şekil 2.10 "Junction box" Birleştirme Kutusu

Bu sayede analog/dijital çeviriciye, yani indikatöre, sistemden tek bir giriş elde edilmiş olur. Sistem üzerine uygulanan yük, tüm yük hücrelerinin ortalama lineer bir değeri olarak sisteme sunulmuş olacaktır.

Montajı yapılmış kantarın çıkış voltaj değerinin kaç kiloya denk düştüğünü anlamak ve ölçülen ağırlığın gerçek değerinin indikatörde gösterimini sağlamak için, kurulan sistemin kalibre edilmesi gerekmektedir. Sonuçta sistem, üzerine baskı yapan iskelet ve platformlar nedeni ile yük altında, ve çıkış değeri sıfırdan farklı bir seviyededir. Bu baskı sistem üzerinde olmasa bile, yük hücrelerinin tam anlamıyla sıfır voltaj degeri vermeleri beklenmez. Küçük değerlerde de olsa belirli sapmalar mutlaka olacaktır.

"Junction box"'ta trimpotlar sayesinde yük hücreleri çıkışları birleştirilerek, ölçüm için tek bir çıkış elde edilmiştir. Bu çıkış, toplam çıkışların bir ortalaması şeklinde düşünülebilir. Kantarın yapısı ve üzerine gelen yüklerin dağılımı nedeniyle, yük hücreleri üzerine, birbirine yakın değerlerde baskı uygulanmaktadır. Sistemde kullanılan yük hücrelerinin besleme gerilimleri, çıkış omaj değerleri ve birim voltaj değerine karşı üretmiş oldukları çıkış voltaj seviyeleri aynı olmalıdır. Bu değerler farklı yük hücrelerinde aynı olsa bile, hassasiyetlerinde farklılık gösterebilecekleri için sistemde kullanılan tüm yük hücreleri aynı tip ve marka olmak zorundadır. Bu sayede eşdeğere yakın uygulanan baskı ile hemen hemen yük hücreleri birbirine yakın değerlerde çıkış verecek, montaj ve mekanik yapısı nedeniyle meydana gelen farklılıklar da, junction box üzerindeki trimpotlarla hassas ayar yapılarak giderilebilecektir.

Kalibrasyon işleminin ilk adımı, sistemin sıfır değerini belirlemek olur. Bunun için kantar üzerinde herhangi bir yük yok iken kararlı konumda ölçüm yapılır ve ölçülen değer kaydedilir. Bu değer aynı zamanda kantarın "ölü ağırlığı" olarak adlandırılmaktadır.

İkinci adım ise değeri bilinen bir yükün, kantar üzerinde ölçülmesinden oluşmaktadır. Bu noktada dikkat edilmesi gereken husus, yük hücrelerinin birbirine olan etkileri göz önünde bulundurularak, uygulanan yükün, platformlar üzerine eş dağılımlarının sağlanması gerektiğidir. Bu işlem Şekil 2.11'de görülmektedir.



Şekil 2.11 Kalibrasyon için yük dağılımı

Yük hücresinin yapısı gereği sistem lineerdir. Bu sayede çözüm, iki noktası bilinen bir eğrinin eğiminin bulunması problemine indirgenmiş olur. Eğrinin en az hata ile belirlenebilmesi için, ikinci nokta kantarın kapasitesinin en azından 70%'i oranında bir yükle belirlenmelidir. Kalibrasyon, daha küçük bir değerle yapılacak olursa, oluşan eğride dikkate alınmayacak küçüklüklte ki sapma, uygulanacak yükle birlikte oransal olarak artacağından, kapasiteye yakın değerlerde yapılan ölçümlerde yanlış sonuçlar elde edilir.

Belirlenen noktalar ile,

AD0 = Sıfır çevrim sonucu

ADx = X kilo degerinin çevrim sonucu

olmak üzere eğim a,

$$a = (ADx - AD0) / X \tag{2.22}$$

formülü ile belirlenir. Böylece tartılan herhangi bir yük için sonuç;

ADy : Y kilo değerinin çevrim sonucu

olmak üzere,

$$Y = (Ady - AD0)/a \tag{2.23}$$

formülü kullanılarak belirlenir.

Sistem eğrisinin belirlenmesiyle, yaklaşık olarak ölçümler yapılabilecektir. Yalnız hassas ayar yapılmadığından tam değer henüz ölçülememektedir. Yani, kalibrasyon için kullanılan değer sistem üzerinde farklı noktalarda ölçüldüğü zaman sonuç sapmakta, ölçüm yaklaşık olarak yapılabilmektedir. Karşımıza çıkan değerde ki sapma, köşe ayarı problemi olarak bilinir. Kabaca kalibre edilmiş kantar için bu problem junction box'ta bulunan trimpotlar ile sapmayı oluşturan yük hücrelerinin ağırlık değerleri değiştirilerek giderilmeye çalışılır.

Kamyon ve tır kantarları için sıkça başvurulan ayar metodlarından birisi, sistem üzerinde değeri bilinen bir kamyonun, farklı noktalardaki ölçümlerinin eşitlenmesi işlemidir. Sapmanın en az olacağı ölçüm noktası, platforma yükün en dengeli dağıldığı

orta noktadır. Bir kamyon orta noktada ölçülerek, platform üzerinde farklı noktalarda yapılan ölçüm sonuçları bu değere yakınsamaya çalışılır. Yakınsama işlemi zahmetli bir iştir. Çünkü, her trimpot değerinin değişimi, diğer trimpotları, yani yük hücrelerini etkileyeceği için, ortak noktanın bulunması için belirli sayıda iterasyon yapmak gerekir.

Bu işlem tamamlandıktan sonra ikinci aşama kamyonun, ön ve arka dingil ağırlıkları farklı olduğu için kantar üzerine ters yönden çıkarılması olacaktır. Böylece baskıyı oluşturan yükün ağırlık merkezinin değiştirlmesi sağlanır. Aynı ölçümler bu durum içinde yapılarak sonuçlar eşitlenmeye çalışılır. Nihai durumda sistemin ortalama bir değer vermesi sağlanarak kalibrasyon işlemi tamamlanmış olur.

3. YAPAY SİNİR AĞLARI

3.1. Yapay Sinir Ağlarına Giriş

Bu bölümde, ağ yapılarını ayrıntılı bir şekilde incelemeye başlamadan önce yapay sinir ağları hakkında genel bir fikir vermek amaçlanmaktadır. Kısım 3.1.1 de yapay sinir ağlarının tanımı, eğitilmesi ve bilgilerin saklanması gibi genel özelliklere değinilecektir. Kısım 3.1.2 de yapay sinir ağlarının uygulamalarda hangi amaçlarla kullanıldığı anlatılacaktır. Kısım 3.1.3 de ağların hangi alanlarda kullanıldığı açıklanacak, teorik ve pratik uygulamalara örnekler verilecektir. Sırasıyla kısım 3.1.4 ve kısım 3.1.5 de yapay sinir ağlarının avantajlarına ve dezavantajlarına yer verilecektir. Kısım 3.1.6 da yapay sinir ağlarının kısa bir tarihi anlatılacak, kısım 3.1.7 de ise en çok bilinen ağ yapılarına ilişkin bir sınıflandırma verilecektir.

3.1.1. Yapay sinir ağlarının genel özellikleri

Yapay sinir ağları, örnekler aracılığı ile olayları öğrenebilen, öğrendiği bilgileri kullanarak yeni bilgiler üretebilen ve keşfedebilen böylelikle kazandığı bilgi, birikim ve tecrübeler ile çevreden gelen etkilere insanlarınkine benzer tepkiler verebilen sistemlerdir. Örneklerden elde ettikleri bilgiler ile kendi deneyimlerini oluşturur ve benzer konularda kararlar verebilirler. Bu özellikleri sayesinde ilişkilendirme, sınıflandırma ve optimizasyon konularında başarılı bir şekilde uygulanmaktadır (Haykin 1994).

Yapay sinir ağlarında yapılması gerekli ilk işlem ağın eğitilmesidir. Ağın eğitilmesi, eğitim verilerinin yapay sinir ağına tanıtılması ve bu verilerin temsil ettiği olaylar arasındaki ilişkilerin yapay sinir ağı tarafından belirlenmesidir. Eğitilen ağın, doğruluğunu ve güvenirliğini ölçebilmek için test verileri kullanılır. Test verileri, eğitim verilerinin bir bölümü olabileceği gibi yapay sinir ağının eğitiminde kullanılmayan verilerde olabilir. Test amacı ile farklı verilerin kullanılması yapay sinir ağının doğruluğunun daha iyi ölçülmesini sağlayacaktır.

Yapay sinir ağı eğitildikçe hücreler arasındaki bağlantı değerleri, bunlara ağırlık değeri denilmektedir, değişir. Yapay sinir ağlarında bilgi, bu bağlantılardaki ağırlık değerleridir. Yapay sinir ağlarında bilgi, belleğe dağılmış durumdadır. Ağ üzerindeki ağırlık değerleri, yani bilgiler tek başlarına bir anlam taşımazlar, çünkü öğrenilen olay tüm ağ tarafından karakterize edilir (ÖZTEMEL 2006).

3.1.2. Yapay sinir ağlarının yetenekleri

Yapay sinir ağları uygulamalarda genellikle şu fonksiyonları yerine getirmek üzere kullanılırlar;

Tahmin : Yapay sinir ağına tanıtılan örneklerden yararlanılarak, verilen giriş değerleri için çıkış değerleri tahmin edilir. Hava tahmini, döviz kurlarının tahmini gibi örnekler vermek mümkündür.

Veri Filtreleme : Bu amaçla eğitilen ağlar, birçok veri arasından uygun olanları belirleme görevini yerine getirirler. Telefon konuşmalarındaki gürültüleri asıl konuşmalardan ayıran ağlar bu konudaki uygulamalara örnek olarak verilebilir.

Arıza Tespiti ve Teşhisi : Bu amaçla geliştirilen ağlar, makinalarda, sistemlerde yada en genel halde süreçlerde meydana gelen olumsuzlukların ve problemlerin ortaya çıkartılmasında, arızaların tespit edilmesinde kullanılmaktadır.

Sınıflandırma : Yapay sinir ağları, sınıflandırma uygulamalarında başarılı bir şekilde kullanılmaktadır. Bu uygulamalarda örnekler, belirli sınıflara ayrıştırılarak kümelendirilmekte ve daha sonra gelen bir örneğin hangi sınıfa ait olduğu belirlenmektedir. Birçok problem, sınıflandırma problemi haline getirilmekte ve yapay sinir ağları kullanılarak çözülebilmektedir. Bir makine üzerinde görülen hataların sınıflandırılması bu konudaki uygulamalara örnek olarak verilebilir.

Eksik Verilerin Tamamlanması : Bu amaçla eğitilen ağlar, verilerin hatalı ve eksik olup olmadığını belirleyerek, eksik olan bilgileri tamamlarlar. Yırtık bir fotoğrafın tamamlanması bu konudaki uygulamalara örnek olarak verilebilir.

Tanıma ve Eşleştirme : Bu amaçla eğitilen ağlar, değişik şekil ve örüntülerin tanınmasında, eksik, karışık, belirsiz bilgilerin işlenerek eşleştirilmesinde kullanılmaktadır. Yapay sinir ağları günümüzde geliştirilmiş en mükemmel şekil tanıyıcı ve sınıflandırıcı olarak kabul edilmektedir. Yapay sinir ağlarının, rakiplerine kıyasla bu kadar üstün olmasını sağlayan eksik bilgiler ile çalışabilme ve normal olmayan verileri işleyebilme yetenekleridir (ÖZTEMEL 2006).

3.1.3. Yapay sinir ağlarının kullanıldığı alanlar

Yapay sinir ağlarının kullanılabileceği birçok alan vardır. Dolayısıyla yapay sinir ağlarının kullanıldığı ve başarılı sonuçlar elde edilen yüzlerce uygulama sayılabilir. Örneğin, 1997 yılında Caere firması tarafından üretilen optik karakter okuma sistemi yılda 3 milyon $'dan fazla gelir getirmiştir. Aynı yıl HNC firması tarafından pazarlanan ve kredi kartlarının haksız yere kullanılmasını ortaya çıkartan Falcon isimli yapay sinir ağı sistemi yılda 23 milyon $ kar sağlamıştır. 1998 yılında Sensory firması tarafından geliştirilen ses tanıma sistemindeki yonganın 5$'a mal olduğu ve bir milyondan fazla sattığı bilinmektedir.

Bu örneklerin çoğaltılması mümkündür. Fakat herhangi bir problemin çözümü için yeterli etkinlikte ve verimlilikte bir yöntem varsa, bu problemi yapay sinir ağı ile çözmek mantıklı olmayabilir. Başarılı uygulamalar incelendiğinde, yapay sinir ağlarının, doğrusal olmayan, çok boyutlu, gürültülü ve hata olasılığı yüksek sensör verilerinin bulunduğu, kesin olmayan, karmaşık, verilerin eksik olduğu, ortaya çıkan çözümde matematiksel modele ve algoritmaya ihtiyaç duyulmayan hallerde yaygın olarak kullanıldıkları görülmektedir (SARAÇ 2004).

a) Yapay Sinir Ağlarının Kullanıldığı Teorik Uygulamalar

- Doğrusal olmayan sistem modelleme
- Akıllı kontrol
- Sinyal filtreleme ve doğrusal olmayan işleme
- Optimizasyon
- Probabilistik fonksyon kestirimleri
- Sınıflandırma
- Örüntü tanıma, ilişkilendirme ve eşleştirme

b) Yapay Sinir Ağlarının Kullanıldığı Pratik Uygulamalar

- K proseslerin modellenmesi
- Akıllı araçlar ve robotlar için optimum rota belirleme
- Robotlarda görme sistemlerinin ve hareket mekanizmalarının kontrol
- Makina, sistem ve süreçler için arıza tespiti
- İletişim kanallarındaki ekoların filtrelenmesi, anahtarlama ve yoğunluğun kontrolü
- Hedef tanıma ve takip sistemleri
- Radar ve sonar sinyallerinin sınıflandırılması
- Radar ve görüntü sinyalleri işleme
- Güvenlik sistemlerinde konuşma ve parmak izi tanıma
- Endüstriyel ürünlerin görsel kalite kontrolü ve imalatta meydana gelen hataların tespiti
- Kredi kartı hilelerinin tespiti
- Döviz kuru tahminleri, risk analizleri

3.1.4. Yapay sinir ağlarının avantajları

Gerçek dünyada olaylar birçok parametreye bağlı olabilir. Ayrıca bu parametrelerin birbirleri üzerinde, açıkça görülemeyen ilişkileri ve etkileri olabilir. Geleneksel yöntemler kullanıldığında bu ilişkileri belirlemek, eğer belirlenemiyorsa yok saymak gerekmektedir. Oysaki yapay sinir ağları kullanıldığında, kullanıcının bu ilişkileri bilmesi ve ağa söylemesi beklenmemektedir. Örneklerin dışında herhangi bir ön bilgiye ihtiyaç yoktur. Yapay sinir ağı, bu ilişkileri, örnekler yardımıyla kendisi öğrenir.

Olayların bağlı olduğu parametrelerin tümü bilinse bile, parametreler arasındaki ilişkiler doğrusal olmayabilir. Geleneksel yöntemler kullanıldığında, varsayımlarla çalışma noktaları civarında lineerleştirmeler yapılır. Bu durum, oluşturulan modelin, gerçek sisteme olan uygunluğunu azaltmakta ve gerçek sistemin kontrolünü

zorlaştırmaktadır. Yapay sinir ağlarında ise ilişkilerin doğrusal olup olmaması problem teşkil etmez. Yapay sinir ağları, yeni bilgilerin ortaya çıkması ve sistemde değişikliklerin olması durumunda yeniden eğitilebilirler, hatta bazı ağların eğitilmesine bile gerek yoktur.

Bilgilerin eksik olduğu durumlarda, yapay sinir ağları etkin çözümler üretebilmektedir. Ayrıca yapay sinir ağlarının belirsiz bilgileri işleyebilme yetenekleri vardır.

Yapay sinir ağları hata toleransına sahiptir. Ağın bazı hücrelerinin bozulması ve çalışamaz duruma gelmesi halinde bile ağ çalışmaya devam eder. Fakat ağın bozuk olan hücrelerinin önemine göre performansında düşmeler görülebilir.

Yapay sinir ağlarının paralel çalışabilmeleri, gerçek zamanlı kullanımlarını kolaylaştırmaktadır (Karna ve Breen 1989).

3.1.5. Yapay sinir ağlarının dezavantajları

Biyolojik sinir ağlarının taklit edilme sebeplerinden en önemlisi paralel çalışabilme yetenekleridir. Dolayısıyla paralel çalışma, yapay sinir ağının da en önemli özelliğidir. Yapay sinir ağlarının bu özelliğinin etkin bir biçimde kullanılabilmesi paralel çalışabilen işlemcilere bağlıdır. Oysaki günümüzde yaygın olarak kullanılan bilgisayar sistemleri seri şekilde çalışmakta ve aynı anda sadece tek bir bilgiyi işleyebilmektedir. Bu durum yapay sinir ağının tam performansta çalışamamasına sebep olmakta ve zaman kaybına

yol açmaktadır. Paralel şekilde çalışabilen işlemciler kullanılsa bile, bu işlemcilerin birbiri ile paralel çalışabilmesi mümkün olamayabilir.

Probleme uygun ağ yapısının belirlenmesi genellikle deneme yanılma yolu ile yapılmaktadır. Uygun ağın oluşturulamaması, düşük performanslı çözümlere neden olabilir. Uygun ağ oluşturulduğunda ise iyi bir çözüm bulunabilir fakat yapay sinir ağı bu çözümün en iyi (optimum) çözüm olduğunu garanti etmez. Oysaki geleneksel yöntemler ile optimum çözümler üretilebilir.

Bazı ağlardaki öğrenme katsayısını, gizli katman sayısını ve gizli katmandaki nöron sayılarını belirlemek için genel geçer bir kural yoktur. Bu durum iyi çözümler bulmayı güçleştirir. Yapay sinir ağı parametreleri, her problem için ayrı faktörler dikkate alınarak tasarlayıcının tecrübesine bağlı olarak belirlenir.

Yapay sinir ağları sadece sayısal veriler ile çalışırlar. Sembolik ifadelerin nümerik gösterime çevrilmesi gerekmektedir.

Yapay sinir ağının ne kadar eğitileceğine keyfi olarak karar verilir. Ağın yaptığı hatanın belirli bir değerin altına düşmesi, eğitimin tamamlanması için yeterli görülmektedir. Fakat en iyi (optimum) öğrenmenin gerçekleştirildiği söylenmemektedir. Optimum sonuçlar veren bir mekanizma henüz geliştirilememiştir. “Yapay sinir ağlarında en iyi çözüm” ucu açık ve araştırılan bir konudur.

Bütün bu dezavantajlara rağmen, yapay sinir ağları ile değişik şekilde çözümler üretilmekte ve uygulamalarda başarılı şekilde kullanılabilmektedir (Haykin 1994).

3.1.6. Yapay sinir ağlarının tarihi

Yapay sinir ağlarının ilk olarak çıkışı, bilim adamlarının nörobiyoloji konusuna ilgi duyması ve elde ettikleri bilgileri bilgisayar sistemleri ile birleştirmeleri sayesinde olmuştur. Yapay sinir ağlarının yadsınamaz katkıları ile aritmetik işlemleri yapan bilgisayarlar, bugün olayları öğrenebilen ve çevre şartlarına göre karar verebilen sistemler haline dönüşmüşlerdir. Bu konuda yapılan çalışmalar, 1970 yılında yaşanan dönüm noktasından ötürü iki kısımda incelenebilir (Haykin 1994).

3.1.6.1. Yapay sinir ağlarının tarihi – 1970 öncesi çalışmalar

1890 yılında beyin fonksiyonları hakkında bilgi veren ilk eser yayınlanmıştır. 1940'dan önceki yıllarda Helmholtz, Pavlov ve Poincare'in yapay sinir ağı kavramı üzerinde çalıştıkları bilinmektedir. Fakat bu çalışmalar mühendislik disiplinlerine dayanmamaktadır. 1940'lı yıllardan sonra Hebb, McCulloch ve Pitts yapılan araştırmaları mühendislik alanlarına kaydırmış ve günümüzdeki yapay sinir ağlarının temellerini atarak ilk yapay sinir hücresini oluşturmuşlardır. Yapay sinir hücreleri ile her türlü mantıksal ifadeyi formülüze etmenin mümkün olduğunu göstermişlerdir. 1949 yılında Donald Hebb, yapay sinir ağının değerlerini değiştiren bir öğrenme kuralı geliştirmiştir. Hebbian öğrenme diye bilinen bu kural, günümüzde de birçok öğrenme kuralının temelini oluşturmaktadır. 1951 yılında ilk nörobilgisayar üretilmiştir. 1954 yılında Farley ve Clark tarafından adaptif tepki üretme kavramı ortaya atılmıştır. Bu kavram 1958 yılında Rosenblatt ve 1961 yılında Caianiello tarafından geliştirilmiştir. Özellikle Rosenblatt tarafından geliştirilen algılayıcı model (perceptron), yapay sinir ağlarının gelişim sürecinde bir dönüm noktası sayılan çok katmanlı algılayıcıların temelini oluşturmuştur. 1962 yılında Widrof ve Hoff, ADALINE (ADAptive LInear NEuron) modelini ortaya atmışlardır. Bu model, Rosenblatt'ın perceptron modeline

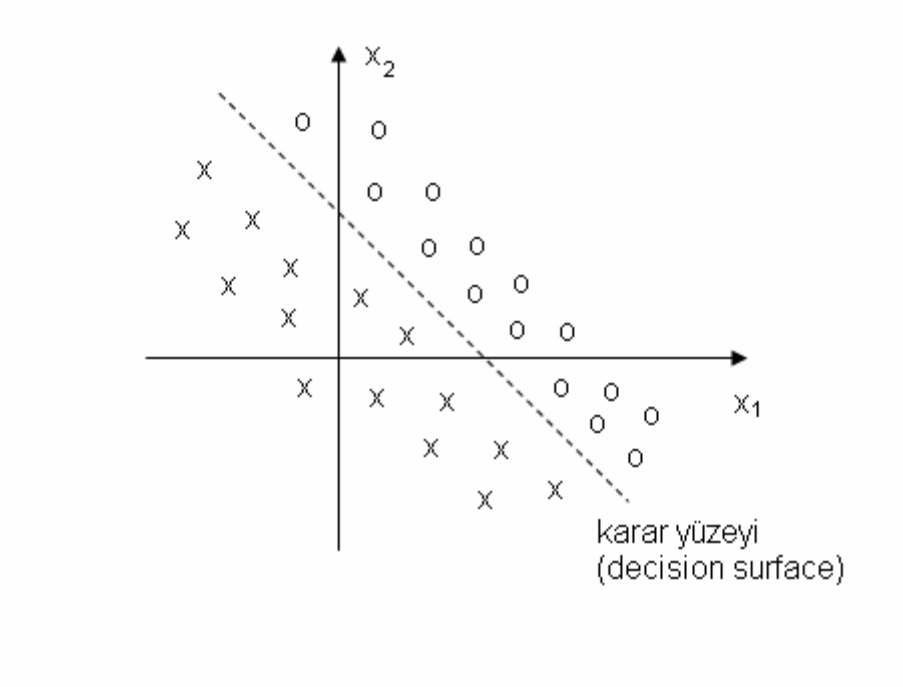
benzemekle birlikte, daha gelişmiş bir öğrenme algoritmasına sahiptir. Adaptif öğrenmenin de temelini oluşturan MADALINE modeli, bu çalışmanın neticesinde ortaya çıkmıştır.

Bu arada bilim dünyasında başka gelişmelerde olmuştur. 1956 yılında Yapay Zeka kavramı ortaya atılmış ve bilim çevreleri tarafından kabul görmüştür. İlk yapay zeka çalışmalarında, sinir ağlarına değinilmemiştir. Nöro-bilgisayarlar ve yapay sinir ağları güncelliğini kaybetmiş, tüm ilgi yapay zeka ve çalışmalarına odaklanmıştır. Yapay sinir ağı savunucularından Grosberg, Kohonen, Rosenblatt, Widrow, Nilssons ve Fukushima'nın yaptığı çalışmalar, özellikle 1965 yılında Nilssons tarafından kaleme alınan ve o tarihe kadar yapılan teorik çalışmaların bir araya getirildiği "Öğrenen Makinalar" isimli kaynak, yapay zekanın yükselmesine ve yapay sinir ağlarının önüne geçmesine engel olamamıştır.

1969 yılında, yapay sinir ağı çalışmaları, yapay zeka biliminin önde gelen isimlerinden Misnky ve Pappert tarafından yazılan Algılayıcılar (perceptrons) isimli kitap yüzünden duraklama devrine girmiştir. Bu kitapta araştırmacılar, yapay sinir ağlarına dayalı algılayıcıların bilimsel bir değer taşımadığını çünkü doğrusal olmayan problemlere çözüm üretemediğini iddia etmişlerdir. Bu tezlerini desteklemek amacı ile XOR probleminin yapay sinir ağı algılayıcıları tarafından çözümlememesini örnek olarak göstermişlerdir. Bu örnek ve ortaya atılan iddia, bilim çevreleri tarafından kabul görmüş ve yapay sinir ağı çalışmaları durma noktasına gelmiştir. Araştırma geliştirme çalışmaları yürüten ve eşgüdümü sağlayan bir organizasyon olan DARPA, yapay sinir ağları ile ilgili çalışmaları desteklemeyi durdurmuştur.

3.1.6.2. XOR problemini bu kadar özel kılan nedir?

İki boyutlu bir öklid uzayı düşünülsün. Bu öklid uzayındaki (χ_1, χ_2) noktalarının her birisine "χ" yada "o" sembollerinin karşılık geldiği farz edilsin. Bu semboller iki boyutlu öklid uzayında Şekil.3.1'de gösterildiği gibi dağılmış olsun.



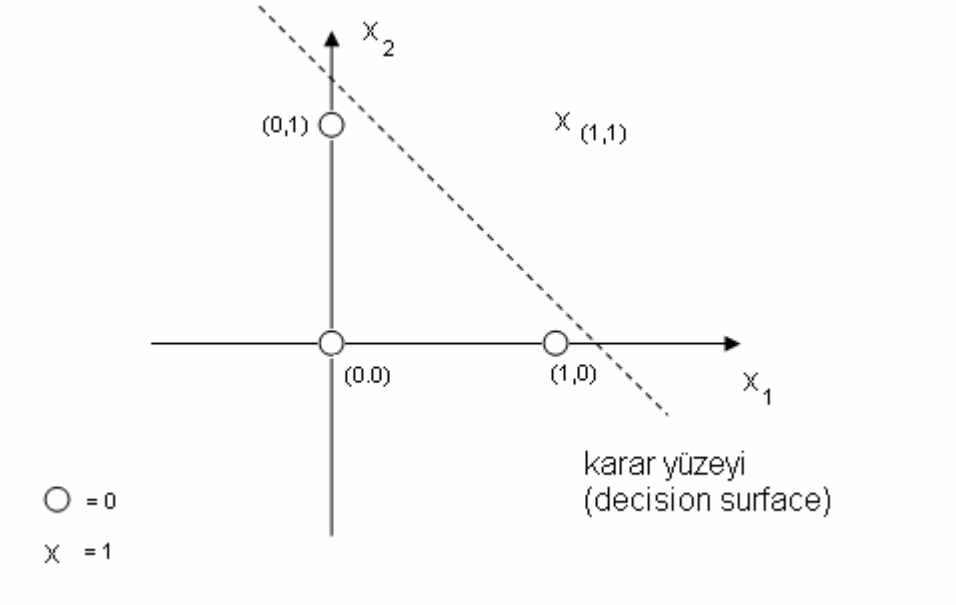
Şekil 3.1 İki boyutlu öklid uzayında (χ_1, χ_2) ikililerine karşılık gelen sembollerin dağılımı

Şekil.3.1'deki "χ" ve "o" sembolleri, tek bir doğru ile izole edilebilmiştir. Bu sebeple, "χ" ve "o" sembolleri lineer ayrıştırılabilirdir (linearly separable) (Haykin 1994).

Örneğin lojik-VE fonksiyonu lineer ayrıştırılabilirdir. Çünkü tek bir doğru ile 0 ve 1 çıkışları izole edilebilmektedir. Detaylı açıklamaları ileriki bölümlere bırakarak, bu

geometrik yorumu, yapay sinir ağları ile ilişkilendirmek istersek, Şekil.3.2'de gösterilen doğrunun, tek katmanlı bir algılayıcıya tekabül ettiğini söyleyebiliriz.

X1	X2	VE
0	0	0
0	1	0
1	0	0
1	1	1

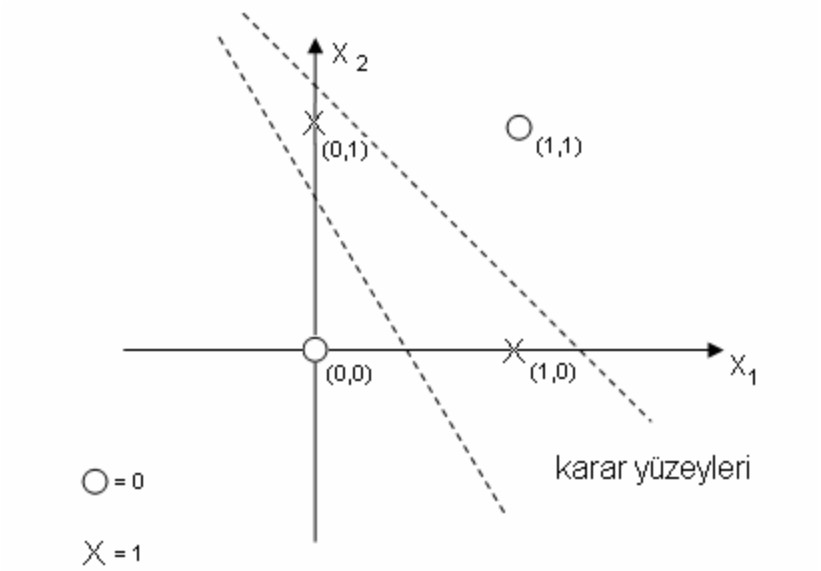


Şekil.3.2 Lojik VE fonksiyonuna ait doğruluk tablosu ve geometrik yorumu

Fakat lojik-ÖZEL VEYA (XOR) fonksiyonu lineer ayrıştırılamamaktadır. 0 ve 1 çıkışlarını izole edebilmek için tek bir doğru yeterli değildir.

X1	X2	ÖZEL VEYA
0	0	0

0	1	1
1	0	1
1	1	0



Şekil.3.3 Lojik ÖZEL VEYA fonksiyonuna ait doğruluk tablosu ve geometrik yorumu

XOR problemini bu kadar özel kılan, en basit non-lineer problem oluşudur. XOR probleminin, lineer yaklaşımlar ile çözülmesi mümkün değildir. (Haykin 1994).

3.1.6.3. Yapay sinir ağlarının tarihi – 1970 sonları çalışmalar

1972 yılında elektrik mühendisi Kohonen ve nöropsikolojist Anderson, çağrışımlı bellek (associative memory) konusunda benzer çalışmalar yayınladılar. Bu çalışmalar öğretmensiz öğrenme kurallarının temelini oluşturdu. Bu yıllarda Grosberg, yapay sinir ağlarının psikolojik mantıksallığını ve mühendislik konularına kolaylıkla uygulanabileceğini gösterdi ve 1978 yılında Carpenter'la birlikte yürüttüğü ortak

çalışma ile Adaptif Rezonans Teorisini (ART) geliştirildi. ART, öğretmensiz öğrenme konusunda zamanında en karmaşık yapay sinir ağı olma özelliğini taşıyordu.

1982 ve 1984 yıllarında Hopfield tarafından yayınlanan çalışmalar ile yapay sinir ağlarının genelleştirilebileceği ve özellikle geleneksel programlama teknikleri ile çözülmesi zor olan problemlere çözüm üretilebileceği gösterildi. Gezgin satıcı probleminin çözülmesi bunun en güzel örneğiydi. Hopfield'in çalışmaları, Hinton ve arkadaşlarının geliştirdikleri Boltzman makinasının temellerini oluşturdu.

1982 yılında, Rumelhart ve arkadaşları paralel programlama ve çok katmanlı algılayıcılar (multilayer perceptron) konularındaki çalışmalarını iki ciltlik bir eserde topladılar. Çok katmanlı algılayıcıların bulunması yapay sinir ağlarının tarihsel gelişimi bakımından çok önemli bir adım oldu. Çünkü tek katmanlı algılayıcıların çözemediği XOR problemi çok katmanlı algılayıcıların geliştirilmesi ile çözüme ulaştı. Yeni bir yapay sinir ağı dalgası, tüm disiplinlerde kendini göstermeye başladı. Çok katmanlı algılayıcılar sadece XOR problemini çözmekle kalmadı, aynı zamanda Hopfield ağlarının ve Boltzman makinasının kısıtlarının da aşılmasını sağladı.

1988 yılında Broomhead ve Lowe, radyal tabanlı fonksiyonlar modelini (radial basis functions) geliştirdiler. Çok katmanlı algılayıcılar modeline alternatif olarak geliştirilen bu model, özellikle filtreleme problemlerinde çok başarılı oldu.

1990 ve 1991 yıllarında Specht, temelini radyal tabanlı fonksiyonlar modelinin oluşturduğu, probabilistik ağları (probabilistic neural networks) ve genel regresyon ağlarını (general regression neural networks) geliştirdi.

Yapay sinir ağlarının gelişim süreci halen devam etmektedir. Yapılan konferanslarda ve sempozyumlarda sinir ağları tartışılmakta ve yeni modeller, yeni öğrenme teknikleri ortaya atılmaktadır. Bununla birlikte, donanım teknolojisindeki gelişmeler sayesinde, sadece kağıt üzerindeki çalışmalar olmaktan çıkmış, günlük hayatta kullanılan pratik uygulamalar haline gelmiştir.

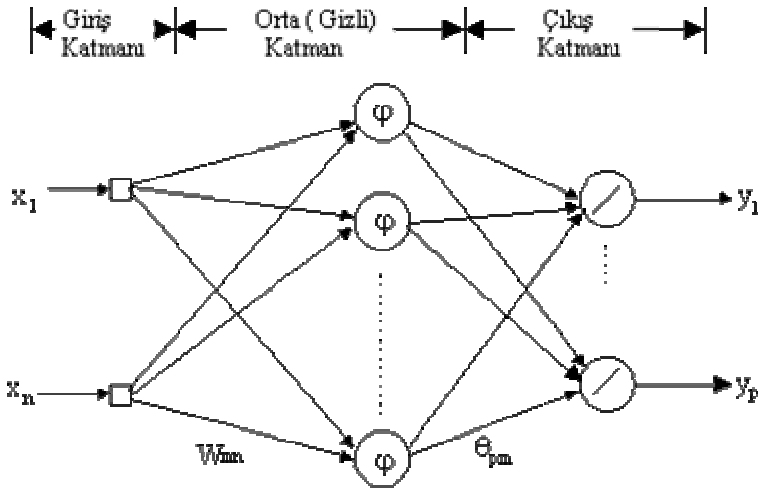
3.1.7. Yapay sinir ağlarının sınıflandırılması

Yapay sinir ağlarına olan ilgi, her geçen gün yeni bir model ve yeni bir öğrenme kuralının geliştirilmesini sağlamaktadır. İlerleyen kısımlarda sadece en çok bilinen yapay sinir ağı modellerine değinilecektir.

Öğrenme algoritmaları en genel halde, öğretmenli (supervised) ve öğretmensiz (unsupervised) olarak iki grupta incelenebilir. Öğretmenli öğrenme algoritmalarında, örnekler girdi-çıktı kümesinin birlikte kullanılması ile oluşturulur. Sistem girişleri ve sistemin bu girişlere verdiği cevaplar, yapay sinir ağına tanıtılır ve yapay sinir ağının giriş-çıkış arasındaki ilişkileri öğrenmesi istenir. Öğretmensiz öğrenme algoritmalarında, sisteme sadece girdi değerleri tanıtılır. Giriş değerleri arasındaki ilişkileri, benzerlikleri ve farklılıkları yapay sinir ağının kendi kendisine öğrenmesi istenir. Bu öğrenme biçimi daha çok sınıflandırma problemleri için kullanılır (SAĞIROĞLU ve ark. 2003).

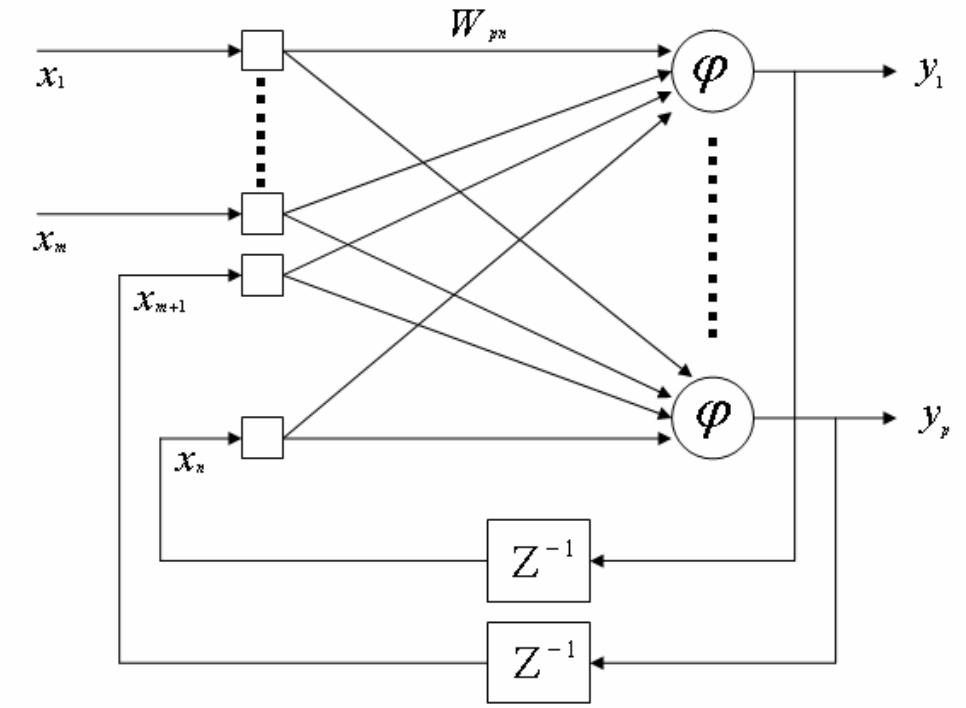
Yapay sinir ağları, ağ yapıları bakımından ileribeslemeli (feedforward) ve geribeslemeli (feedback yada reccurent) olarak gruplandırılabilir. İleribeslemeli ağlarda, bağlantılar kapalı çevrimler oluşturmaz, diğer bir ifade ile herhangi bir nöronun çıkışı, sadece ve sadece bir sonraki katmanda bulunan nöronlar için giriş olarak kullanılır, bu

çıkış önceki katmandaki veya aynı katmandaki nöronlara giriş olarak verilemez. İleribeslemeli ağlar, geleneksel nümerik yöntemler ile eğitilebilir. Şekil.3.4'te 2-girişli 2-çıkışlı çok katmanlı ileribeslemeli bir ağ yapısı gösterilmektedir.



Şekil.3.4 2-girişli 2-çıkışlı çok katmanlı ileribeslemeli ağ yapısı

Geribeslemeli ağlarda ise bağlantılar kapalı çevrimler oluşturur. Herhangi bir nörondan alınan çıkış, aynı katmandaki veya bir önceki katmandaki nöronlara giriş olarak verilebilir. Geribeslemeli ağların eğitim süreleri daha uzun, eğitim yöntemleri ise daha karmaşıktır. Şekil.3.5'te 2-girişli 2-çıkışlı Elman-tipi geribeslemeli bir ağ yapısı gösterilmektedir.



Şekil.3.5 2-girişli 2-çıkışlı Elman-tipi geribeslemeli ağ yapısı

Yapay sinir ağları, çalıştıkları veriler bakımından kategorik (categorical) ve nicel (quantitative) olmak üzere iki grupta incelenebilir. Kategorik veriler yeni, eski, kırmızı, lacivert vb. gibi nitel ifadeler olsa bile, bu sembolik ifadeler, nümerik olarak kodlanmalı ve nümerik değerler ile çalışılmalıdır. Sınıflandırma problemlerinde, öğretmenli öğrenme algoritmalarındaki hedef değerler ve öğretmensiz öğrenme algoritmalarındaki çıkış değerler kategoriktir. Nicel ifadeler ise herhangi bir parametrenin nümerik değeri olabilir. Bazı veriler ise hem kategorik hemde nicel olarak yorumlanabilir. İkilik tabandaki "0" ve "1" değerleri buna örnek olarak verilebilir.

Bu bilgiler ışığı altında en çok yapay sinir ağı modelleri aşağıdaki gibi sınıflandırılabilir (ÖZTEME 2006).

3.1.7.1. Öğretmenli algoritmalar

3.1.7.1.a İleribeslemeli modeller

A1.Lineer Algılayıcılar (Lineer)

- Hebbian (1949-Hebb)
- Perceptron (1958-Rosenblatt)
- Adaline (1960-Widrow, Hoff)
- Functional Link (1989-Pao)

A2.Çok Katmanlı Algılayıcılar (Multilayer Perceptron)

- Backpropagation (1989-Rumelhart, Hinton, Williams)
- Cascade Correlation (1990-Fahlman, Lebiere)
- Quickpropagation (1989-Fahlman)
- RPROP (1993-Riedmiller, Braun)

A3.Radyal Tabanlı Ağlar (Radial Basis Functional Networks)

- Orthogonal Least Squares (1991-Chen, Cowan, Grant)

A4.Cerebellar Model Articulation Controller (1995-Albus)

A5.Classification

- LVQ: Learning Vector Quantization (1988-Kohonen)
- PNN :Probabilistic Neural Network (1990 - Specht)

A6. Regression

- General Regression Nueral Network (1991-Specht)

3.1.7.1.b Geribeslemeli modeller

B1. Bidirectional Associative Memory (1992-Kosko)
B2. Boltzman Machine (1985-Ackley)
B3. Recurrent Time Series

- Backpropagation Through Time (1990-Werbos)
- Elman (1990-Elman)
- Finite Impuls Response (1990-Wan)
- Jordan (1986-Jordan)
- Real Time Recurrent Network (1989-Williams, Zipser)
- Recurrent Backpropagation (1989-Pineda)
- Time Delay Nueral Networks (1990-Lang, Waibel)

3.1.7.1.c. Yarışa dayalı modeller

C1.ARTMAP: Adaptive Resonance Theory with Mapping
C2.Fuzzy ARTMAP (1992-Carpenter, Grossberg)
C3.Gaussian ARTMAP (1995-Williamson)
C4.Counterpropagation (1987-Hecht Nielsen)

3.1.7.2. Öğretmensiz algoritmalar

3.1.7.2.a. Yarışa dayalı modeller

A1.Vector Quantization

- Grossberg (1976-Grossberg)
- Kohonen (1984-Kohonen)
- Conscience (1988-Desieno)

A2.SOM: Self Organizing Map

- Kohonen (1995-Kohonen)
- GTM: Generative Topographic Map (1997-Bishop)
- Local Linear (1995-Mulier, Cherkassky)

A3.ART: Adaptive Resonance Theory

- ART1 (1987-Carpenter, Grossberg)
- ART2 (1987-Carpenter, Grossberg)
- ART3 (1990-Carpenter, Grossberg)
- Fuzzy-ART (1991-Carpenter, Grossberg)

A4.DCL: Differential Competitive Learning (1992-Kosko)

3.1.7.2.b. Boyut azaltmaya dayalı modeller

B1. Hebbian (1949-Hebb)
B2.Sanger (1989-Sanger)
B3.Differential Hebbian (1992-Kosko)

3.1.7.2.c. Çağrışımlı öğrenmeye dayalı modeller

C1.Linear Autoassociator (1977-Anderson)
C2.BSB: Brain State in a Box (1977-Anderson)
C3.Hopfield (1982-Hopfield)

3.2. Yapay Sinir Ağlarının Yapısı

Bu alt bölümde yapay sinir ağlarının yapısı incelenecektir. Kısım 3.2.1'de doğadaki canlılardan esinlenerek yapılmış ve başarılı olmuş uygulamalara örnekler verilecektir. Kısım 3.2.2'de modellemeye çalıştığımız mekanizmanın biyolojik özelliklerine değinilecektir. Kısım 3.2.3'de yapay sinir ağlarının temel birim olan hücreleri incelenecektir. Kısım 3.2.4'de temel birim olan hücrelerin bir araya gelmesiyle oluşturulan topoloji tetkik edilecektir. Kısım 3.2.5'de ise farklı modellerin ortaya çıkmasını sağlayan unsurlar anlatılacaktır.

3.2.1. Teknoloji ve biyolojik yapılar

Günümüz teknolojisi, biyolojik yapılardan esinlenerek bulunmuş yada geliştirilmiş örneklerle doludur. Uçağın mucidi olarak kabul edilen Wright kardeşler, Kittyhawk

adındaki uçaklarının kanatlarını yaparken akbaba kanatlarının yapısını örnek almışlardır. Planörlerin kanat yapısı albatroslardan örnek alınarak tasarlanmıştır.

Yunusların burun çıkıntısı, modern gemilerin pruvasına örnek olmuştur. Yunus burnu şeklindeki bu tip pruvalar sayesinde %25'e ulaşan oranda yakıt tasarrufu sağlanmaktadır. Yunusların derisi ise denizaltı tasarlayan mühendislere ilham vermiştir. Yunus derisi üç katmanlı bir yapıdadır. Dıştaki katman ince ve çok esnektir, içteki katman kalındır, orta katman ise süngerimsi bir maddeden oluşur. Alman denizaltı mühendisleri, bu özelliğe sahip sentetik bir kaplama yapmayı başarmışlardır. Bu kaplamaların kullanıldığı denizaltıların hızlarında %45 oranında bir artış görülmüştür.

Yarasalar ultrason denilen çok yüksek titreşimli ses dalgaları yayarlar. Bu ses dalgaları havadaki ve yerdeki hayvanlara veya yarasanın önündeki diğer engellere çarparak yansır. Yarasa, yansıyan ve kendine gelen bu titreşimlere göre yönünü ve hareket hızını belirler. Radarın çalışma prensibi, yarasanın yön ve hız tayini yaptığı mekanizma ile aynıdır.

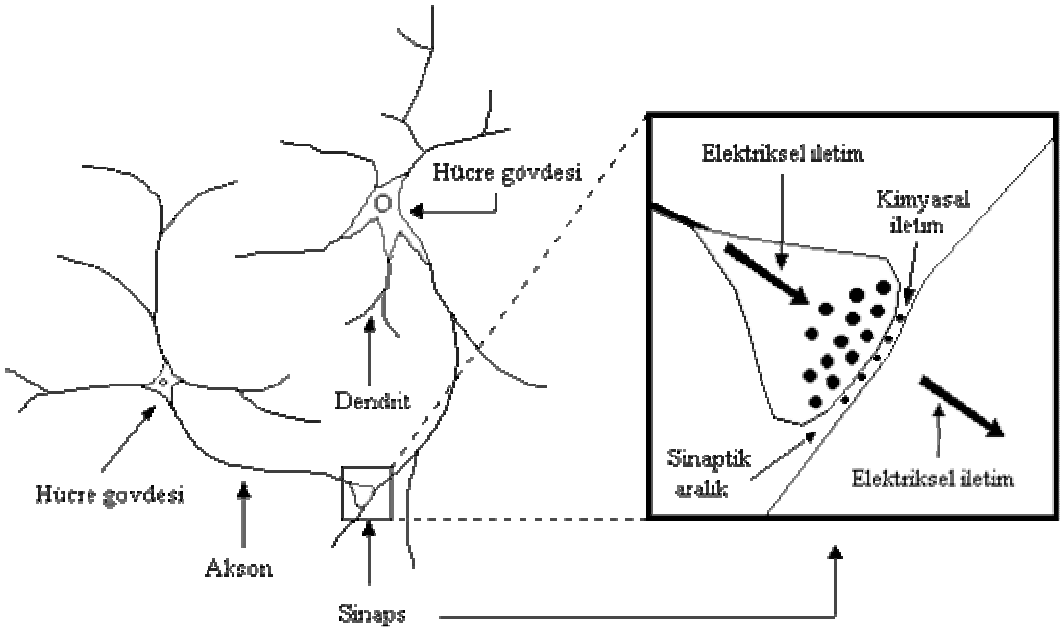
Arıların gözleri yan yana dizilmiş altıgenlerden oluşur. Gök cisimlerinden gelen X-ışınlarını toplamak için kullanılan uzay teleskoplarının merceği arıların göz yapısından ilham alınarak, yan yana dizilmiş altıgen şeklindeki aynalardan üretilmektedir. Altıgenlerden oluşan dizilim teleskopa geniş bir görüş alanı ve yüksek kalite sağlamaktadır.

Bilim adamlarının ve mühendislerin taklit etmeye çalıştığı bir diğer biyolojik yapı ise insan beyni ve sinir ağlarıdır. Beyin ve sinir ağları algılama, öğrenme, düşünme, problem çözme, hafızada tutma ve hafızada tutulan bilgileri tekrar çağırma gibi

yetenekleri olan karmaşık bir yapıya sahiptir. Ağ yapılarını incelemeye başlamadan önce modellemeye çalıştığımız mekanizmanın biyolojik özellikleri incelenecektir (Karna ve Breen 1989).

3.2.2. Biyolojik sinir ağlarının yapısı

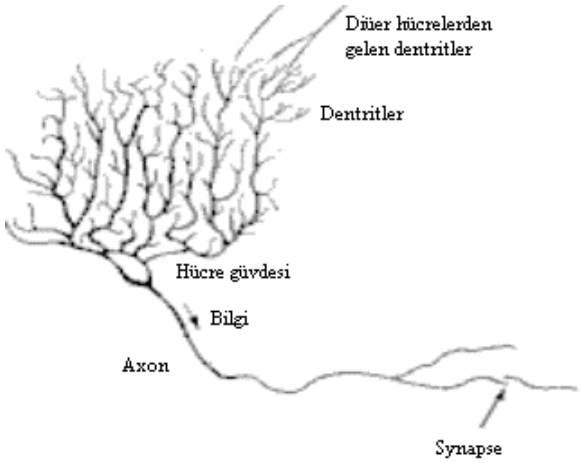
Sinir sistemi, genel olarak merkezi sinir sistemi ve çevresel sinir sistemi olmak üzere iki kısma ayrılır. Beyin ve omurilik merkezi sinir sistemini oluşturur. Çevresel sinir sistemi ise, vücudun sensörlerinden alınan bilgileri (görme, işitme, ağrı, ısı, titreşim, vb.) merkezi sinir sistemine taşıyan ve merkezden alınan emirleri ilgili birimlere (kas, salgı bezleri, vb.) ulaştıran sinir ağlarından meydana gelmiştir. Sinir ağının temel birimi sinir hücreleridir (nöron). Nöronların birbirlerini uyarması ve böylelikle bilginin bir nörondan diğer nörona geçmesiyle elektriksel sinyaller sinir ağı boyunca taşınır. Şekil 3.6'da biyolojik bir nöronun şematik yapısı gösterilmektedir.



Şekil 3.6 Sinir ağlarının temel birimi olan sinir hücresinin (nöronun) şematik yapısı

Nöronların bilgiyi aldıkları noktalara dendrit adı verilir. Eğer nörona gelen işaret yeteri kadar güçlüyse, nöron bir çıkış işareti üretir. Üretilen bu çıkış işareti bir sonraki nöronun yada nöronların dendritlerine, aksonlar vasıtasıyla iletilir. İşaretin, verici nöronun aksonundan, alıcı nöronun dendritlerine iletilmesi sinapslar (sinaptik boşluk) sayesinde gerçekleşir. Sinaptik boşluklar, nöro-transmitter sıvısı ile doludur. Bu sıvı iletilen işaretin güçlendirilmesini veya zayıflatılmasını sağlar. Verici nörondan gelen işaret, nöro-transmitter sıvısı içindeki kimyasalların yer değiştirmesine sebep olur. Kimyasalların yer değiştirmesi ile meydana gelen potansiyel sayesinde oluşan elektrik sinyali alıcı nöronun dendritlerine ulaşır. Öğrenme ve hafızaya alma olayları, sinaptik boşluğun empedansının yada kondüktansının ayarlanması ile mümkün olmaktadır (Zurada 1992).

Beynin kıvrımlı bölümü olan beyin kabuğunda (korteks), yaklaşık 100 milyar nöron bulunur. Herbir nöronun yaklaşık 1000 nöron ile bağlantısı vardır. İki nöronun bağlantısını sinaptik boşluklar sağladığından, toplam 100 trilyon sinaptik boşluğun olduğu söylenebilir. Şekil 3.7'de 100 milyar nöronun oluşturduğu sinir ağının küçük bir kısmı gösterilmektedir.



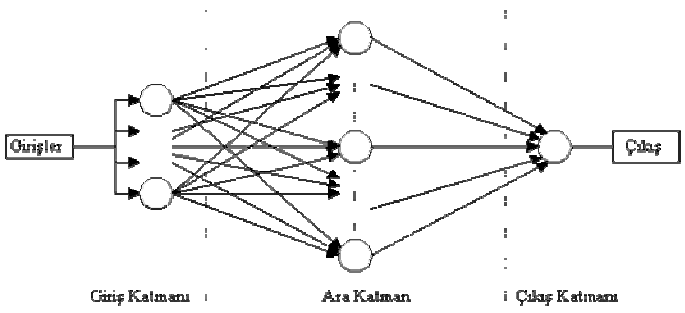
Şekil 3.7 Vücudumuzdaki sinir ağının küçük bir kısmı

Bu rakamlar en karmaşık yapay sinir ağı ile karşılaştırıldığında bile çok büyüktür. Biyolojik sinir ağlarının, yapay sinir ağlarına olan sayıca üstünlüğü açıkça görülmektedir. Fakat insan beyninin 400 – 500 [Hz] ile, işlemcilerin ise 3 – 4 [GHz] ile çalıştığı düşünülürse, bilgisayar tabanlı sistemlerin hız bakımından daha avantajlı olduğu söylenebilir. Buna rağmen, seri olarak çalışan işlemciler, paralel olarak çalışan beyin yanında bu avantajını kaybetmektedir. Bu durumu aşmak için son yıllarda paralel tarzda çalışan nöroişlemciler üretilmekte ve yapay sinir ağları için özel donanımlar geliştirilmektedir (Zurada 1992).

Bundan sonraki kısımda yapay sinir ağlarının temel birimi olan yapay sinir hücresi incelenecek ve biyolojik sinir hücreleri ile olan benzerlikleri irdelenecektir.

3.2.3 Yapay sinir hücresinin yapısı

Yapay sinir ağlarının temel birimi sinir hücreleridir. Mühendislik disiplinlerinde, yapay sinir hücrelerine, proses elemanları da denilmektedir. Her bir proses elemanı 5 temel unsura sahiptir. Bunlar, girişler, ağırlıklar, toplama fonksiyonu, aktivasyon fonksiyonu ve çıkıştır. Şekil 3.8'de görüldüğü gibi her bir hücre çok-girişli tek-çıkışlı yapıdadır (Haykin 1994).



Şekil 3.8 Yapay sinir hücresinin yapısı

Girişler (x_1, x_2,….,x_i) : Giriş katmanındaki hücreler için, kullanıcı tarafından örnekler ile oluşturulmuş veri kümesidir. Diğer katmandaki hücreler için, herhangi bir katmandaki hücrenin çıkışı olabilir.

Ağırlıklar (w_1, w_2, …., w_n) : Girişlerin, çıkışa ne oranda aktarılacağını gösterir. Örneğin w_1 ağırlığı, x_1 girişinin, çıkışa olan etkisini göstermektedir. Ağırlıkların büyük, küçük, pozitif yada negatif olması, ilgili girişin önemli yada önemsiz olduğunu göstermez. Ağırlıklar sabit yada değişken değerler olabilir.

Toplama Fonksiyonu: Bir hücrenin net girdisini hesaplamak için kullanılır. Bu amaç ile değişik fonksiyonlar kullanılmaktadır. En çok tercih edilen, ağırlıklı toplam fonksiyonudur. Bu fonksiyonda her giriş kendi ağırlığı ile çarpılır ve bu değerler toplanır. x_i girişleri, w_i ağırlıkları ve n hücrenin giriş sayısını göstermek üzere ağırlıklı toplam fonksiyonu;

$$NET = \sum_{i=1}^{n} x_i \cdot w_i \quad \text{'dir.} \tag{3.1}$$

Yapay sinir ağındaki bütün hücrelerin toplama fonksiyonlarının aynı olması gerekmez. Her hücre bağımsız olarak farklı bir toplama fonksiyonuna sahip olabilir. Kullanılan diğer toplama fonksiyonları Tablo 3.1'de gösterilmektedir.

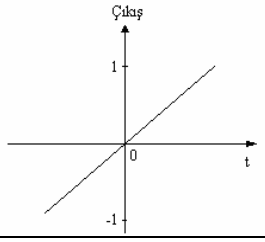
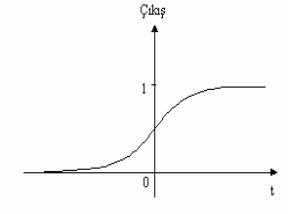
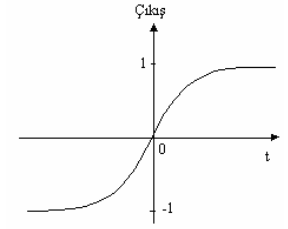
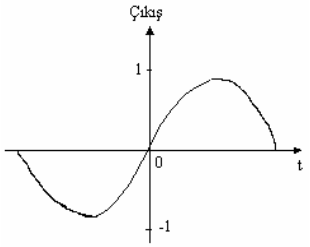
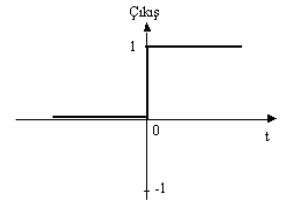
Tablo 3.1 : Yapay sinir hücrelerinde kullanılan toplama fonksiyonları

Fonksiyonun Adı	Fonksiyon *(i=1,2,…,n)*	Açıklama
Ağırlıklı Toplam	$NET = \sum x_i \cdot w_i$	Girişler ve ağırlık değerleri çarpılır. Bulunan değerler birbirleri ile toplanır.
Çarpım	$NET = \prod x_i \cdot w_i$	Girişler ve ağırlık değerleri çarpılır. Bulunan değerler birbirleri ile çarpılır.
Maksimum	$NET = Max(\mathrm{x}_i . w_i)$	Girişler ve ağırlık değerleri çarpılır. Bulunan değerlerin en büyüğü alınır.
Minimum	$NET = Min(\mathrm{x}_i . w_i)$	Girişler ve ağırlık değerleri çarpılır. Bulunan değerlerin en küçüğü alınır.
Artımlı Toplam	$NET_k = NET_{k-1} + \sum x_i \cdot w_i$	Ağırlıklı toplam bulunur. Bir önceki ağırlıklı toplam ile toplanır.

Aktivasyon Fonksiyonu: Hücrenin net girdi değerine karşılık üretilecek çıktı değerinin hesaplanmasında kullanılır. Çok katmanlı algılayıcılarda olduğu gibi, bazı sinir ağı modellerinde aktivasyon fonksiyonunun türevlenebilir olması şartı vardır.

Ayrıca aktivasyon fonksiyonunun türevinin hesaplanması, ağın eğitim süresi bakımından önem taşır. Sigmoid fonksiyonunun türevi, fonksiyonun kendisi cinsinden yazılabilmektedir. Bu durum, işlem kolaylığı sağladığından Sigmoid fonksiyonu, yaygın şekilde kullanılmaktadır. Toplama fonksiyonunda olduğu gibi, bütün hücrelerin aynı aktivasyon fonksiyonunu kullanması gerekmez. Her hücre bağımsız olarak farklı bir aktivasyon fonksiyonuna sahip olabilir. Tablo 3.2'de çeşitli aktivasyon fonksiyonları gösterilmektedir.

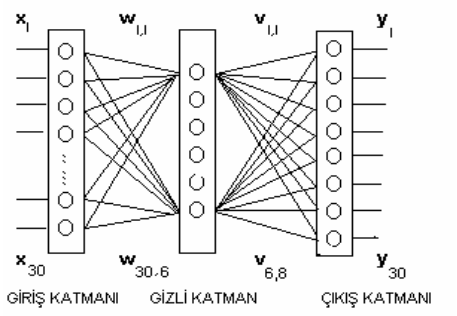
Tablo 3.2 : Yapay sinir hücrelerinde kullanılan aktivasyon fonksiyonları

Fonksiyon Adı	Fonksiyonun Şekli	Matematiksel İfadesi	Açıklama
Lineer Fonksiyon		$F(NET)=NET$	Hesaplanan net giriş değeri, hücrenin çıkışı olarak kabul edilir
Sigmoig Fonksiyon		$F(NET)=\frac{1}{1+e^{-\beta.NET}}$	β değeri keyfi şekilde değiştirilerek farklı eğimlere sahip eğriler elde edilir
Hiperbolik Tanjant Fonksiyon		$\boldsymbol{F(NET)=\frac{e^{NET}+e^{-NET}}{e^{NET}-e^{-NET}}}$	Hesaplanan net giriş değeri, tanjant fonksiyonuna uygulanır
Sinüs Fonksiyon		$F(NET)=Sin(NET)$	Öğrenilmesi istenen olayın, sinüs fonksiyonuna uygun dağılım gösterdiği durumlarda kullanılır
Basamak Fonksiyon		$F(NET)=\begin{cases}1 & NET>k\\ 0 & NET\leq k\end{cases}$	k, eşik değeri göstermektedir. Hesaplanan değerin, eşik değerden büyük yada küçük olmasına göre 1 yada 0 çıkışları üretilir

Çıkış (y): Aktivasyon fonksiyonu tarafından belirlenen değerdir. Üretilen çıktı, başka bir hücreye yada dış dünyaya gönderilebilir. Geribesleme olması durumunda, hücre kendi üzerinde geribesleme yaparak, kendi çıktı değerini, giriş olarak kullanabilir. Bununla birlikte, geribesleme başka bir hücreye de yapılabilir. Ağ şeklinde gösterildiğinde, bir hücrenin, birden fazla çıkışı varmış gibi görünmektedir. Fakat bu durum sadece gösterim amaçlıdır. Bu çıkışların hepsi aynı değere sahiptir.

3.2.4. Yapay sinir ağının yapısı

Yapay sinir hücreleri bir araya gelerek, yapay sinir ağını oluşturur. Yapay sinir ağı üç kısımdan meydana gelir. Bunlar giriş katmanı, gizli katman(lar) ve çıkış katmanıdır. Giriş ve çıkış katmanlardaki hücre sayıları, uygulama tarafından belirlenir. Örneğin 30-girişli 8-çıkışlı bir sistem için kurulacak yapay sinir ağında, giriş katmanında üç hücre, çıkış katmanında ise iki hücre bulunacaktır. Gizli katmanların sayısı ve bu katmanlardaki gizli hücrelerin sayısı ise keyfi şekilde tasarlayıcı tarafından belirlenir. Gizli katman ve gizli hücre sayıları arttıkça, yapay sinir ağının arıza toleransı artacaktır, bununla birlikte işlem karmaşıklığı ve eğitim süresi de artacaktır. İyi bir çözüm için gizli katman ve gizli hücrelerin sayısı önem teşkil etmektedir. Şekil 3.9'da 30-girişli 8-çıkışlı bir sistem için kurulan yapay sinir ağı gösterilmektedir.



Şekil 3.9 Yapay sinir ağının yapısı

Bu sistem için bir tane gizli katman ve bu gizli katmanda altı adet gizli hücre kullanılmıştır. Ağırlık değerleri wij notasyonu ile belirtilir. Bu notasyon i. hücreden j. hücreye olan bağlantıyı gösterir. Yapay sinir ağlarında bilgi, bağlantıların sahip olduğu ağırlık değerlerinde saklanır. Ağırlık değerleri bütün ağa yayılmış olduğundan, ağ belleği dağıtık hafızalıdır. Olay tüm ağ tarafından karakterize edilir, bu yüzden tek bir ağırlık değeri bir anlam ifade etmez (RUMELHART ve McClelland 1986).

Yukarıda gösterilen ağ için giriş vektörü x1, x2 … x30 değerlerinden oluşmaktadır. Bu girişler motora verilen gerilim değerleri, bir kontrol sistemi için birim basamak giriş, bir resmin gri tonlarını temsil eden sayısal değerler yada bir parmak izini gösteren nümerik değerler olabilir. Çıkış vektörü ise y1, y2.. y8'den oluşmaktadır. Öğretmenli öğrenme algoritmalarında y değerleri, hedef değerler olarak tanımlanmaktadır.

Verilen giriş vektörüne karşılık, hedef vektörün üretilebilmesi için ağırlıkların değiştirilmesine yapay sinir ağının eğitilmesi denir. Diğer bir ifade ile eğitim, ağılıkların, hedef vektörü üretecek şekilde güncellenmesidir.

Başlangıçta ağırlık değerleri keyfi şekilde belirlenir, çünkü ilk iterasyonda büyük bir hataya sebep olsa bile, bir değerinin olması gerekmektedir. Örnekler yapay sinir ağına tanıtıldıkça, ağılıklar güncellenir ve hedef değeri üretecek şekilde değişir. Güncelleme işlemi öğrenme kurallarına göre gerçekleştirilir. Öğretmenli yada öğretmensiz öğrenme stratejileri için farklı öğrenme kuralları vardır.

Hedef değerler ile yapay sinir ağının ürettiği çıkışlar arasındaki hata, belirli bir değerin altına düştüğünde, ağın eğitilmesi durdurulur ve performansını ölçmek için test

aşamasına geçilir. Test aşamasında ağırlıklar değiştirilmez. Eğitim sırasında bulunan ağırlıklar kullanarak, yapay sinir ağının çıkış üretmesi sağlanır. Üretilen bu çıkışların doğruluğu, ağın performansı hakkında bilgi verir. Eğer performans yeterli görülürse, yapay sinir ağının öğrendiği kabul edilir. Artık yapay sinir ağı, gösterilen örneklerin temsil ettiği olay hakkında genelleme yapabilecek yeteneği kazanmıştır (RUMELHART ve McClelland 1986).

3.2.5. Farklı modellerin ortaya çıkmasını sağlayan unsurlar

Öğrenme stratejileri (öğretmenli, destekleyici, öğretmensiz), kullanılan öğrenme kuralı (Hebb, Hopfield, Delta, Kohonen), yapay sinir hücrelerinin bağlanması sonucu oluşan topoloji (ileribeslemeli veya statik, geribeslemeli veya dinamik), hücrelerin sahip oldukları toplama ve aktivasyon fonksiyonları, yapay sinir ağının modelini belirlemektedir.

Bu modellerden en çok kullanılanlar; Tek Katmanlı Algılayıcılar, Çok Katmanlı Algılayıcılar, Vektör Kuantizasyon Modelleri, Olasılığa Dayalı Ağlar, Radyal Tabanlı Ağlar, Boltzman makinası, Elman Ağı, Adaptif Rezonans Teorisi Modelleri'dir (Haykin 1994).

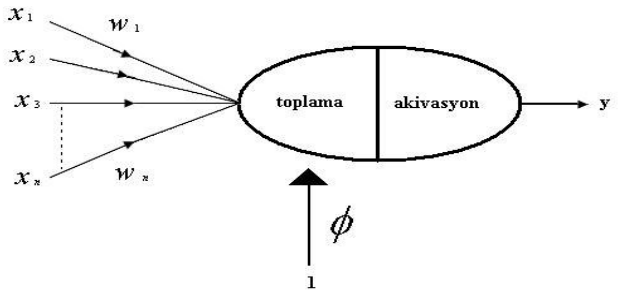
Çalışmanın bundan sonraki kısmında sırayla tek katmanlı algılayıcılara ve çok katmanlı algılayıcılara değinilecektir. Tek katmanlı algılayıcılar, tüm modellerin temelini oluşturmaktadır. Fakat tek katmanlı algılayıcılar ile sadece lineer problemler çözülebilmektedir. Lineer olmayan sistemleride modelleyebilmek için çok katmanlı algılayıcılara da değinilecektir.

3.3. Tek Katmanlı Algılayıcılar

Bu bölümde, çok katmanlı algılayıcıların temeli olan katmanlı algılayıcılar incelenecektir. Kısım 3.3.1'de tek katmanlı algılayıcıların ortak özelliklerine değinilecektir. Yüzey ayırıcının tanımı yapılacak ve özel bir durum için yüzey ayırıcının geometrik yorumu irdelenecektir. Kısım 3.3.2'de Rosenblatt tarafından geliştirilen en basit tek katmanlı algılayıcı olan '' perceptron '',kısım 3.3.3'de ise diğer bir algılayıcı modeli olan ''ADALINE '' incelenecektir. Perceptron ve ADALINE arasındaki farklar ortaya konulacaktır. Perceptron modelinde kullanılan Hebb öğrenme kuralı ve ADALINE modelinde kullanılan Delta öğrenme kuralı tanıtılacaktır. Kısım 3.3.4'de bu iki tek katmanlı algılayıcı modelinin, sınıflandırma ve lineer ilişlileri belirleme performansları karşılaştırılacak ve aktivasyon fonksiyonları açısından birbirine olan üstünlükleri ortaya konulacaktır.

3.3.1. Tek katmanlı algılayıcıların ortak özellikleri

Tek katmanlı algılayıcılar girişler, toplama-aktivasyon görevlerini yerine getiren bir gövde ve çıkıştan oluşur. Bu bakımdan çok-girişli tek çıkışlı yapıdadır. Şekil 3.10'da tek katmanlı bir yapay sinir modeli gösterilmektedir.

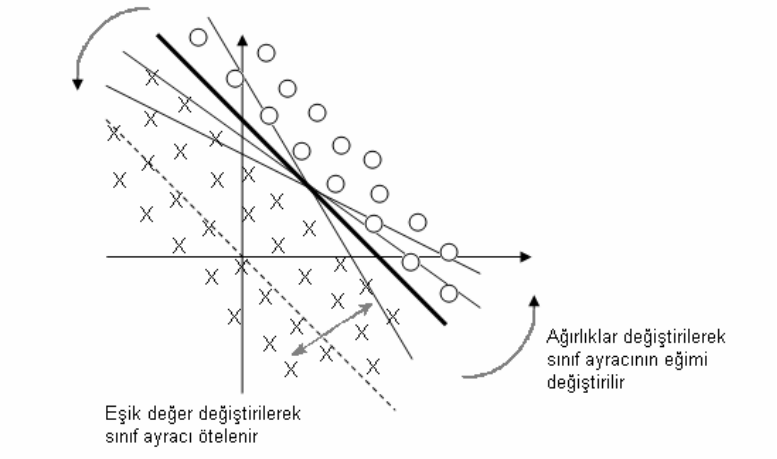


Şekil 3.10 Tek katmanlı yapay sinir ağı modeli

Tek katmanlı ağlarda, girişlerin toplamının ve dolayısıyla çıkışın sıfır olmasını önleyecek bir eşik değer (Ø) kullanılabilir. Kullanılmasını zorunlu kılan durumların dışında, eşik değerin olması yada olmaması keyfidir. Eşik değer iterasyonlar sırasında, ağırlık değerleri gibi değiştirilebilir. Eşik değerin girişi her zaman 1'dir (Simpson 1990).

Tek katmanlı algılayıcılarda, giriş değerleri iki farklı sınıfa ayrılarak, kümelendirmeye çalışılır. Dolayısıyla problem, iki sınıfı birbirinden ayıran bir yüzeyin bulunmasıdır. Bu yüzey, uzayı iki farklı bölgeye ayırır ve farklı çıkış değerlerini alan giriş parametreleri, bu yüzey tarafından kümelendirilmiş olur.

İki sınıfı birbirinden ayırmak için kullanılan yüzey, sınıf ayracı olarak tanımlanır. Bu kavramı daha kolay anlayabilmek için, tek katmanlı yapay sinir ağının 2-girişli olduğunu farz edilsin. Bu durumda, sınıf ayracı iki boyutlu öklid uzayında bir doğru belirtir. İki girişli sistem için sınıf ayracı Şekil 3.11'de gösterilmektedir.

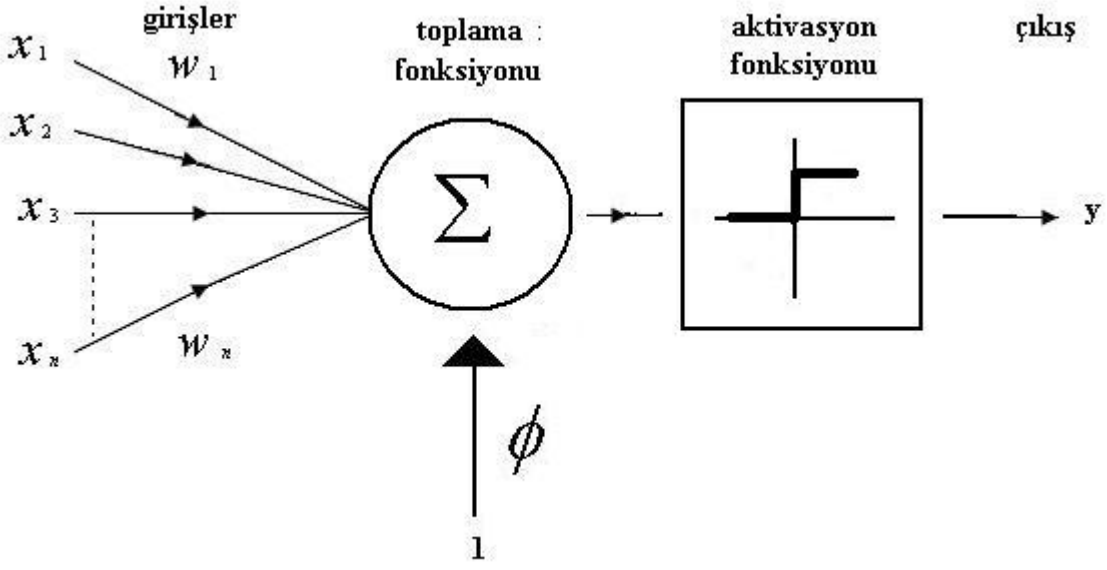


Şekil 3.11 İki-girişli bir sistem için sınıf ayracı

Eşik değerin değiştirilmesi, sınıf ayracını ötelerken, ağırlıkların değiştirilmesi eğimini etkiler. Eğitim sırasında eşik değer ve ağırlıklar değiştirilerek, doğrunun, sınıfları ayıracak şekilde konumlandırılması sağlanır (Simpson 1990).

3.3.2. En basit tek katmanlı algılayıcı

Perceptron, 1958 yılında Rosenblatt tarafından şekil sınıflandırma amacı ile geliştirilmiştir. Şekil 3.12'de en basit tek katmanlı algılayıcılar modeli gösterilmektedir (ÖZTEMEL 2006).



Şekil 3.12 En basit tek katmanlı algılayıcı modeli (perceptron)

Giriş ve hedef vektörleri : Giriş değerlerinin sayısı için bir kısıtlama yoktur. Giriş vektörü, negatif yada pozitif herhangi bir değeri içerebilir. Giriş değerlerinin 1'den büyük yada küçük olması performansı etkilemez. Perceptronun üretmesini istediğimiz

çıkışlara hedef değer denir. Giriş vektörü için söylenenler, hedef vektör için geçerli değildir. Kullanılan aktivasyon fonksiyonu sebebi ile hedef vektör, sadece 0 yada 1 değerlerini içerebilir.

Toplama fonksiyonu: Ağırlıklı toplam fonksiyonu kullanılır. Yüzey ayracınım ötelenmesi gereken durumlarda eşik değer ilave edilebilir.

$$NET = \left(\sum_{i=1}^{r} w_i \cdot x_i \right) + \text{Ø} \tag{3.2}$$

Aktivasyon fonksiyonu : Perceptron, sınıflandırma amacı ile geliştirildiğinden, farklı sınıfları temsil edecek değerler üretilmelidir. Sınıflandırma işleminde, klasik küme kuramındaki ait olma yada ait olmama durumları incelenir. Bulanık mantıkda olduğu gibi bir aitlik derecesi vermek mümkün olmamaktadır. Bu amaçla aktivasyon fonksiyonu olarak basamak fonksiyon kullanılır.

NET girdinin eşik değerden büyük yada küçük olmasına göre çıkış değeri 0 yada 1 olarak üretilir.

$$y = f(NET) = \begin{cases} 1 & Eger \quad NET > \phi \\ 0 & aksi \quad takdirde \end{cases} \tag{3.3}$$

Öğrenme kuralı (Hebb kuralı) : Eğer perceptronun ürettiği çıkış ile hedef değer aynı olursa, ağırlıklarda herhangi bir değişme olmaz. Bu durum sınıf ayracının doğru pozisyonda olduğunu gösterir. Fakat ağ, hedef değerden farklı bir çıkış üretmiş ise ki bu

durumda ağırlıklar değiştirilerek sınıf ayracının pozisyonu değiştirilmelidir, iki durum söz konusu olabilir.

i) Hedef değer 0 ve gerçek çıkış 1 ise $\Rightarrow w_i(k+1) = w_i(k) - \eta \cdot x_i$

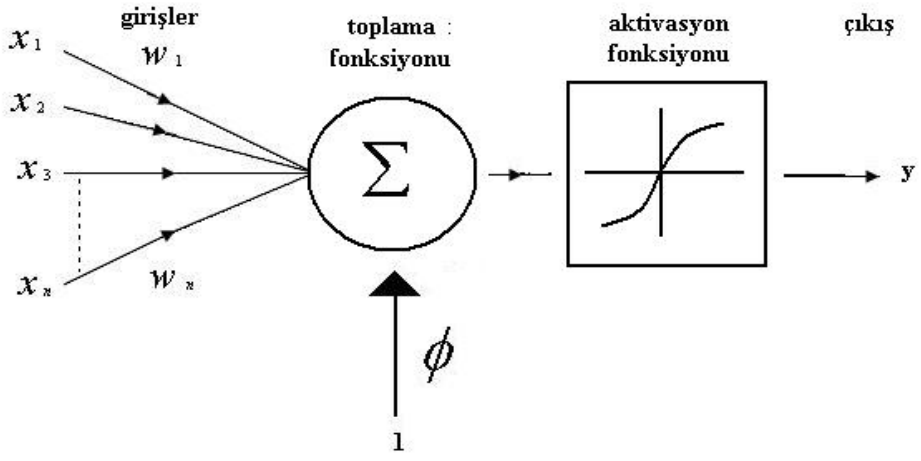
ii) Hedef değer 1 ve gerçek çıkış 0 ise $\Rightarrow w_i(k+1) = w_i(k) + \eta \cdot x_i$

η , öğrenme katsayısı olup [0 1] aralığında herhangi bir değer alabilir. Eşik değer benzer şekilde güncellenir.

$$\phi(k+1) = \phi(k) \pm \eta \tag{3.4}$$

3.3.3. ADALINE modeli

Widrow ve Hoff tarafından 1959 yılında geliştirilmiştir. ‘‘ADAptive LINear Element’’ kelimelerinin kısaltılmış şeklidir. Şekil 3.13’de ADALNE modeli gösterilmektedir.



Şekil 3.13 Aktivasyon fonksyonu sigmoidal olan bir ADALINE modeli

ADALINE modelinde, perceptronda olduğu gibi basamak aktivasyon fonksyonu kullanılabilir. Fakat genellikle hiperbolik tanjant fonksyonu yada sigmoidal fonksyon kullanılır. Bunun dışında öğrenme kuralı, perceptronda kullanılan Hebb kuralından farklıdır.

Giriş ve hedef vektörleri : Perceptronlarda olduğu gibi, giriş değerlerinin sayısı için bir kısıtlama yoktur. Giriş vektörü, negatif yada pozitif herhangi bir değeri içerebilir. Giriş değerlerinin 1'den büyük yada küçük olması performansı etkilemez. Hedef vektörüne getirilecek kısıtlamalar ise, aktivasyon fonksyonları ile ilişkilidir. Eğer perceptronda olduğu gibi basamak aktivasyon fonksyonu kullanılırsa, hedef değerler sadece 0 veya 1 olabilir. Fakat hiperbolik tanjant yada sigmoidal aktivasyon fonksyonları kullanılırsa, bu durumda hedef vektör, 0 ve 1 aralığındaki değerleride içerebilir.

Toplama fonksyonu : Ağırlıklı toplam fonksiyonu kullanılır. Yüzey ayracının ötelenmesi gereken durumlarda eşik değer ilave edilebilir.

$$NET = \left(\sum_{i=1}^{n} w_i \cdot x_i \right) + \phi \tag{3.5}$$

Aktivasyon fonksyonu : ADALINE modelinde, genellikle hiperbolik tanjant fonksyonu yada sigmoidal fonksyon kullanılır. Sigmoidal fonksyonda β değerini değiştirerek, farklı eğimlerde fonksyonlar elde etmek mümkündür.

$$y = f(NET) = \frac{e^{NET} - e^{-NET}}{e^{NET} + e^{-NET}} \quad \text{yada} \quad y = f(NET) = \frac{1}{1 + e^{-\beta NET}} \tag{3.6}$$

Öğrenme kuralı (Delta kuralı) : y* hedef değeri ve y gerçek çıkışı göstermek üzere, delta öğrenme kuralını elde etmek amacı ile hata,

$$e = y^* - y \tag{3.7}$$

ve hatanın negatif değerinden kurtulmak için karesel hata tanımlansın.

$$kareselhata = \left(y^* - y\right)^2 \tag{3.8}$$

Hata yüzeyinin minimum noktasını bulabilmek için gradyanın tersi yönde ilerlenirse,

$$\frac{\partial(kareselhata)}{\partial w_i} = -2\left(y^* - \sum_i w_i x_i\right)\cdot x_i \Rightarrow \eta \cdot e \cdot x_i \tag{3.9}$$

denklemi elde edilir.

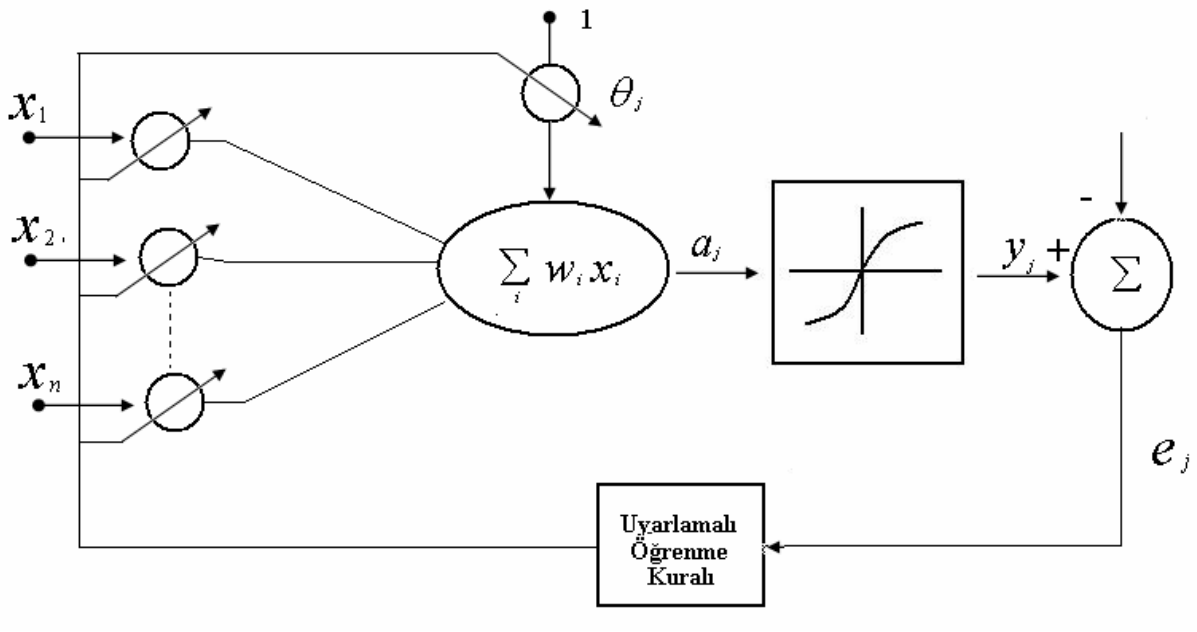
Yukarıdaki ifadede katsayılar, öğrenme katsayısına, η , dahil edilmiştir. Böylelikle ağırlıkları güncellemek için kullanılacak delta öğrenme kuralı bulunmuş olur.

$$\Delta w_i = \eta \cdot e \cdot x_i \quad \Rightarrow \quad w_i(k+1) = w_i(k) + \eta \cdot e \cdot x_i \tag{3.10}$$

Benzer şekilde eşik değer, aşağıdaki ifade yardımıyla değiştirilir.

$$\phi(k+1) = \phi(k) + \eta \cdot e \tag{3.11}$$

ADALINE modelinin yapısı şekil 3.13'te verilmektedir. Fakat ağırlıkların, çıkış ve hedef vektörler arasında tanımlanan hatandan yararlanarak güncellenmesi durumunu görebilmek için, ADALINE modeli, şekil 3.14'te ayrıntılı bir biçimde tekrar gösterilmiştir.



Şekil 3.14 ADALINE modelinin ayrıntılı yapısı

3.3.4. Perceptron ve ADALINE modellerinin karşılaştırılması

Bu kısımda perceptron ve ADALINE modellerinin karşılaştırılması yapılacak ve birbirlerine olan üstünlükleri incelenecektir. Bu amaç ile basamak aktivasyon fonksyonu kullanan perceptron ve sigmoidal aktivasyon fonksyonu kullanan ADALINE modelleri için örnekler verilmektedir. Öncelikle bu iki model sınıflandırma konusunda karşılaştırılacaktır. Lojik fonksyonların gerçeklenmesi, sınıflandırma problemi olarak

düşünülebilir. Örneğin lojik–VE fonksyonunda (0,0), (0,1) ve (1,0) girişleri 0-kümesine dahil edilirken, (1,1) girişi 1-kümesine dahil edilmektedir. Perceptron ve ADALNE modellerinin performansını, lojik-VE fonksyonunu gerçekleyerek inceleyelim.

Perceptronun Sınıflandırma Performansı : Yazılan komut, perceptronu geliştiren kişinin ismine ithafen rosenblatt olarak kullanılmıştır. Giriş vektörü, ‘‘giriş’’ ve yapay sinir hücresinin üretmesini istediğimiz çıkış vektörü ise ‘‘hedef’’ olarak tanımlanacaktır. ‘‘defa’’ ve ‘‘hata’’ parametreleri sırasıyla iterasyonun kaç kere tekrarlanacağını ve kabul edilebilecek en büyük hatayı göstermektedir. Bu kriterlerden birisinin sağlanması iterasyonu durduracaktır. Bu parametreler belirtilmezse, sırasıyla 1000 ve 10^{-12} olarak kabul edilecektir.

```
y =  rosenblatt (giriş, hedef, defa, hata)
    giriş   = [ 0 0 ; 0 1 ; 1 0 ; 1 1];
    hedef = [0 ; 0 ; 0 ; 1];

     >>  rosenblatt (giriş, hedef )

     Yapılan iterasyon sayısı 12 dir

     Nöronun eğitilmesi sonucunda bulunan çıkış vektörü

     0
     0
     0
     1

     Nöronun eğitilmesi sonucunda bulunan ağırlık vektörü

     2.535670381
```

3.434425738

Böylelikle yapılan karesel hata 0.0000000000e+000 dir

Bu işlem toplam 0.521 saniye sürmüştür.

ADALINE'in Sınıflandırma Performansı : Benzer işlemler ADALINE için de yapılabilir. Sadece adaline komutunda, perceptrondan farklı olarak öğrenme katsayısını gösteren ‘‘eta’’ parametresi vardır. Aynı giriş ve hedef değerleri için ADALINE ‘den alınan sonuçlar aşağıdaki gibidir.

```
y  =  adaline (giriş, hedef, eta, defa, hata)
    giriş   = [0 0 ; 0 1 ; 1 0 ;1 1];
    hedef = [0 ; 0; 0 ; 1];

    >>  adaline (giriş, hedef,0.1,100000)
```

Yapılan iterasyon sayısı 100000 dir.

Nöronun eğitilmesi sonucunda bulunan çıkış vektörü

0.0001000419811
0.0001000614995
0.0001000614996
0.9998999385

Nöronun eğitilmesi sonucunda bulunan ağırlık vektörü

-0.5479858383
-0.5398764858

Böylelikle yapılan karesel hata 4.0045309050e-008 dir

Bu işlem toplam 21.4 saniye sürmüştür.

Görüldüğü gibi, perceptron istenilen hedef vektörü 12 iterasyon yaparak tam olarak bulmuş ve hatayı sıfırlayabilmiştir. Fakat ADALINE modeli, daha fazla iterasyona rağmen, sonuca ancak 10^{-8} mertebesinde bir hata ile yaklaşabilmiştir. Bunun sebebi istenilen hedef değerlerinin 0 ve 1 (binary) olmasıdır. Hatırlanacağı gibi perceptron basamak aktivasyon fonksyonu kullanırken (sadece 0 ve 1 çıkışını üretebilmektedir), ADALINE sigmoidal aktivasyon fonksyonu kullanmaktadır. Bunun bir sonucu olarak perceptron problemlerinde daha üstün bir performans göstermektedir.

Perceptronun Lineer İlişkileri Belirleme Performansı : Giriş ve hedef vektörlerini uygun şekilde seçilerek, lineer ilişkileri belirlemek için kullanılabilir. Giriş matrisi tablo 3.3'te gösterildiği gibi seçilsin. Hedef vektör ise, birinci, üçüncü, beşinci elemanların toplanması, ikinci ve dördüncü elemanların çıkarılması ile oluşturulsun.

Tablo 3.3 : Giriş matrisi ve hedef vektörü

Giriş Matrisi					Hedef Vektörü
X_1	X_2	X_3	X_4	X_5	$X_1 - X_2 + X_3 - X_4 + X_5$
0.2	0.7	0.9	0.1	0.5	0.8
0.3	0.5	1.0	0.9	0.3	0.2
0.4	0.6	0.8	0.3	0.7	1.0
1.0	0.9	0.4	0.5	0.1	0.1

Görüldüğü gibi, perceptron $hedef = x_1 - x_2 + x_3 - x_4 + x_5$ şeklinde tanımlanan fonksyonu ortaya çıkaramamıştır. Yapılan hata çok büyüktür ve iterasyon sayısının artırılması, durumu fazla değiştirmemektedir.

>>rosenblatt (giriş, hedef, 100000)

Yapılan iterasyon sayısı 100000 dir.

Nöronun eğitilmesi sonucunda bulunan çıkış vektörü

0
0
1
0

Nöronun eğitilmesi sonuncunda bulunan ağırlık vektörü

2.729088764
-0.8487043736
-0.0467753825
1.964989341
9.207235741

Böylelikle yapılan karesel hata 6.9000000000e–001 dir.

Bu işlem toplam 26,6 saniye sürmüştür.

ADALINE'in Lineer İlişkileri Belirleme Performansı : Benzer işlemler ADALINE için de yapılabilir. Aynı giriş ve hedef değerleri için ADALINE modelinden alınan sonuçlar aşağıdaki gibidir.

>> adaline (giriş, hedef)

Yapılan iterasyon sayısı 10000 dir

Nöronun eğitilmesi sonucunda bulunan çıkış vektörü

0.8002624194

0.2001787478
0.9992995502
0.1001263783

Nöronun eğitilmesi sonucunda bulunan ağırlık vektörü

0.9000673777
-0.2487610977
-0.984165417
-0.8868143373
-2.790121933

Böylelikle yapılan karesel hata 6.0741615217e-007 dir.

Bu işlem toplam 2.05 saniye sürmüştür.

ADALINE modeli tarafından yapılan karesel hata 10^{-7} mertebesindedir. Sınıflandırma probleminin tam tersine, $hedef = x_1 - x_2 + x_3 - x_4 + x_5$ şeklinde tanımlanan fonksyonun saptanmasında ve bu lineer ilişkinin ortaya çıkartılmasında, ADALINE modelinin daha üstün bir performans gösterdiği açıktır. Bunun sebebi [0 1] aralığında değer alabilen aktivasyon fonksyonudur.

Sonuç olarak sınıflandırma probleminde, basamak aktivasyon fonksyonu kullanan perceptron daha iyi performans gösterirken, lineer ilişkileri belirleme probleminde, sigmoidal aktivasyon fonksyonu kullanan ADALINE ile daha üstün bir performans elde edilmiştir. Yapılan bu basit karşılaştırma sayesinde, uygulamanın niteliğine göre, doğru aktivasyon fonksyonu ile çalışmanın önemi anlaşılmaktadır (Hagan ve ark. 1996).

3.4. Çok Katmanlı Yapay Sinir Ağları

Bu bölümde hatanın geriye yayılımı ‘‘ backpropogation ’’ yöntemini kullanan ileribeslemeli çok katmanlı yapay sinir ağı modeli incelenecektir. Kısım 3.4.1’de ‘‘neden tek katmanlı algılayıcılar yeterli değildir?’’ sorusuna cevap aranacaktır. Kısım 3.4.2’de yapay sinir ağının yapısı incelenecektir. Kısım 3.4.3’de hatanın geriye yayılımı yönteminde kullanılan genişletilmiş delta öğrenme kuralı açıklanacaktır. Son olarak kısım 3.4.4’te ise ağların tasarımında karşılaşılan problemler ve dikkat edilmesi gereken noktalar belirtilecektir.

3.4.1. Çok katmanlı yapay sinir ağlarına olan gereksinim

Tek katmanlı algılayıcılar basit ve hızlı olmakla birlikte, sadece karmaşık olmayan lineer problemleri çözebildiklerinden, mühendislik uygulamalarında kullanılabilecek yeterlilikte değillerdir. Bu durum eğitilebilecek tek bir ağırlık matrisinin olmasından kaynaklanmaktadır. Bu yüzden karmaşık ve lineer olmayan problemlerin çözülmesinde çok katmanlı yapay sinir ağlarına ihtiyaç duyulmaktadır. Bu gereksinim, tek katmanlı algılayıcılar bölümünde lojik-VE probleminin çözülmesinde daha iyi bir performans gösteren ‘‘rosenblatt’’ modeli üzerinde gösterilecektir. Hatırlanacağı gibi, ‘‘rosenblatt’’, 12 iterasyonda, sıfır hata ile istenilen lojik-VE çıkışını üretebilmişti. Lojik-Özel Veya problemi için alınan sonuçlar aşağıda gösterilmektedir.

```
>> giriş   = [0 0 ; 0 1; 1 0 ; 1 1];
>> hedef  = [ 0 ; 1 ; 1 ; 0 ];
>>  rosenblatt (:giriş, hedef)
```

Yapılan iterasyon sayısı 1000 dir.

Nöronun eğitilmesi sonucunda bulunan çıkış vektörü

1
1
1
1

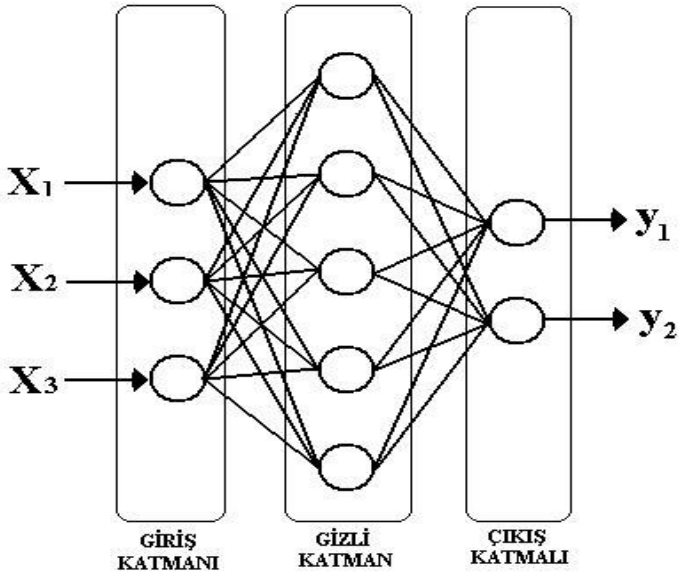
Böylelikle yapılan karesel hata 2.0000000000e+000 dir

Görüldüğü gibi fonksyonun ürettiği çıkış, istenilen lojik-Özel Veya çıkışı ile uyuşmamaktadır. Lineer olmayan bu problemin çözülmesi için daha gelişmiş modellere ihtiyaç duyulmaktadır.

3.4.2. Çok katmanlı yapay sinir ağlarının yapısı

Bu modelde, giriş ve çıkış katmanlarının dışında, sayısı tasarımcı tarafından belirlenen ara katman veya katmanlar bulunur. Bazı kaynaklarda ara katman yerine gizli katmanda denilmektedir.

Giriş ve çıkış katmanlarındaki hücre sayıları, uygulamanın niteliğine bağlıdır. Örneğin 3-girişli 2 çıkışlı bir sistem için tasarlanacak yapay sinir ağında, giriş katmanında 3 ve çıkış katmanında 2 adet hücre bulunacaktır. Gizli katman sayısı ve gizli katmanlarda bulunacak hücre sayılarının belirlenmesinde ise bir kısıtlama yoktur. Fakat gizli katman ve gizli hücre sayılarının uygun şekilde seçilmemesi, yapay sinir ağının performansını olumsuz şekilde etkileyecektir. Bu sayıların belirlenmesi bir optimizasyon problemidir. Şekil 3.15'te 3-girişli 2-çıkışlı bir sistem için tasarlanan çok katmanlı yapay sinir ağı gösterilmektedir. Bu tasarımda, tek bir gizli katman ve bu gizli katmanda 5 tane gizli hücre kullanılmıştır.



Şekil 3.15 İleri beslemeli çok katmanlı yapay sinir ağı

Şekil 3.15'teki gibi ileribeslemeli ağ yapılarında, her bir hücre, sonraki katmanda bulunan tüm hücrelere bağlıdır. Bu yapısından ötürü tam-olarak-bağlanmış ‘‘fullyconnected’’ terimi kullanılır. Fakat bu durum hücrelerin birden fazla çıkışı olduğu anlamına gelmez. Her hücrenin yalnız ve yalnız bir çıkışı vardır. Diğer bir ifade ile hücrelerin ürettikleri çıkış değeri tekdir. Fakat bu çıkış değeri, hücrelere aktarılırken ilgili ağırlıklar ile çarpılır ve böylelikle sonraki hücrelere kuvvetlendirilerek yada zayıflatılarak iletilir.

3.4.3. Çok katmanlı yapay sinir ağlarının öğrenme kuralı

Hatanın geriye yaylımı ‘‘ backpropogation ’’ yöntemini kullanan çok katmanlı yapay sinir ağlarında öğretmenli öğrenme stratejisi kullanılır. Diğer bir ifade ile ağa tanıtılan örnek kümesi hem giriş işaretlerini hemde hedef değerleri içerir.

Geriye yayılım yöntemini kullanılan yapay sinir ağlarının öğrenme kuralı, en küçük kareler yöntemine dayanan delta öğrenme kuralının genelleştirilmiş halidir. Genelleştirilmiş delta kuralı iki aşamadan oluşur. İleriye doğru hesaplama aşamasında başlangıç değerleri ile yapay sinir ağının çıkışı bulunur. Hedef değerler kullanılarak yapılan hata hesaplanır. Geriye doğru hesaplama aşamasında ise ilk aşamada hesaplanan hata geriye doğru yayılarak ağırlıkların nasıl güncelleneceği bulunur.

Yapılacak işlemler, kolaylık açısından tek ara katmanı bulunan bir yapay sinir ağı üzerinde anlatılacaktır. Daha fazla katmanı bulunan ağlar için formül sayısı artmakla birlikte yapılacak işlemler aynı olacaktır.

İleriye Doğru Hesaplama : Bu aşamada amaç, verilen girişler için yapay sinir ağı çıkışının bulunması ve hedef değerler kullanılarak yapılan hatanın hesaplanmasıdır. Bu amaç ile örnekler giriş katmanındaki giriş hücrelerine tanıtılır. Bu bilgiler giriş hücrelerinden ara katman hücrelerine aktarılırken ilgili ağırlıklar ile çarpılır ve ağırlıklı toplam fonksiyonuna göre her ara katman hücresinin NET girdisi hesaplanır.

$$NET_{ap} = \sum_{i} w_{gi,ap} x_i \tag{3.12}$$

Bu formülde kullanılan notasyon aşağıda açıklanmaktadır.

NET_{ap} : ara katmanın p. hücresine olan girdi

$w_{gi,ap}$: giriş katmanının i. hücresini, ara katmanın p. hücresine bağlayan ağırlık

x_i : giriş katmanındaki i. hücresinin çıkışı

Ağırlıklı toplam dışında kullanılan diğer toplama fonksyonları tablo 3.4'te gösterilmektedir.

Tablo 3.4 : Yapay sinir ağlarında kullanılan toplama fonksyonları

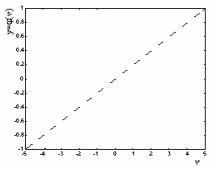
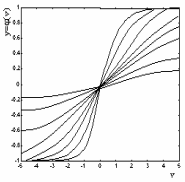
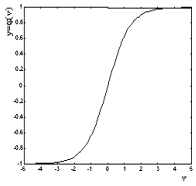
Fonksiyonun Adı	Fonksiyon *(i=1,2,…,n)*	Açıklama
Ağırlıklı Toplam	$NET = \sum x_i \cdot w_i$	Girişler ve ağırlık değerleri çarpılır. Bulunan değerler birbirleri ile toplanır.
Çarpım	$NET = \prod x_i \cdot w_i$	Girişler ve ağırlık değerleri çarpılır. Bulunan değerler birbirleri ile çarpılır.
Maksimum	$NET = Max(x_i . w_i)$	Girişler ve ağırlık değerleri çarpılır. Bulunan değerlerin en büyüğü alınır.
Minimum	$NET = Min(x_i . w_i)$	Girişler ve ağırlık değerleri çarpılır. Bulunan değerlerin en küçüğü alınır.
Artımlı Toplam	$NET_k = NET_{k-1} + \sum x_i \cdot w_i$	Ağırlıklı toplam bulunur. Bir önceki ağırlıklı toplam ile toplanır.

Bulunan NET girdi, bir aktivasyon fonksyonundan geçirilerek, ara katman hücrelerinin çıkışı bulunur. NET_{ap}, ara katmanın p. hücresine olan girdi, F_{ap} ise ara katmanın p. hücresinin aktivasyon fonksyonundan geçirildikten sonraki değerini göstermektedir.

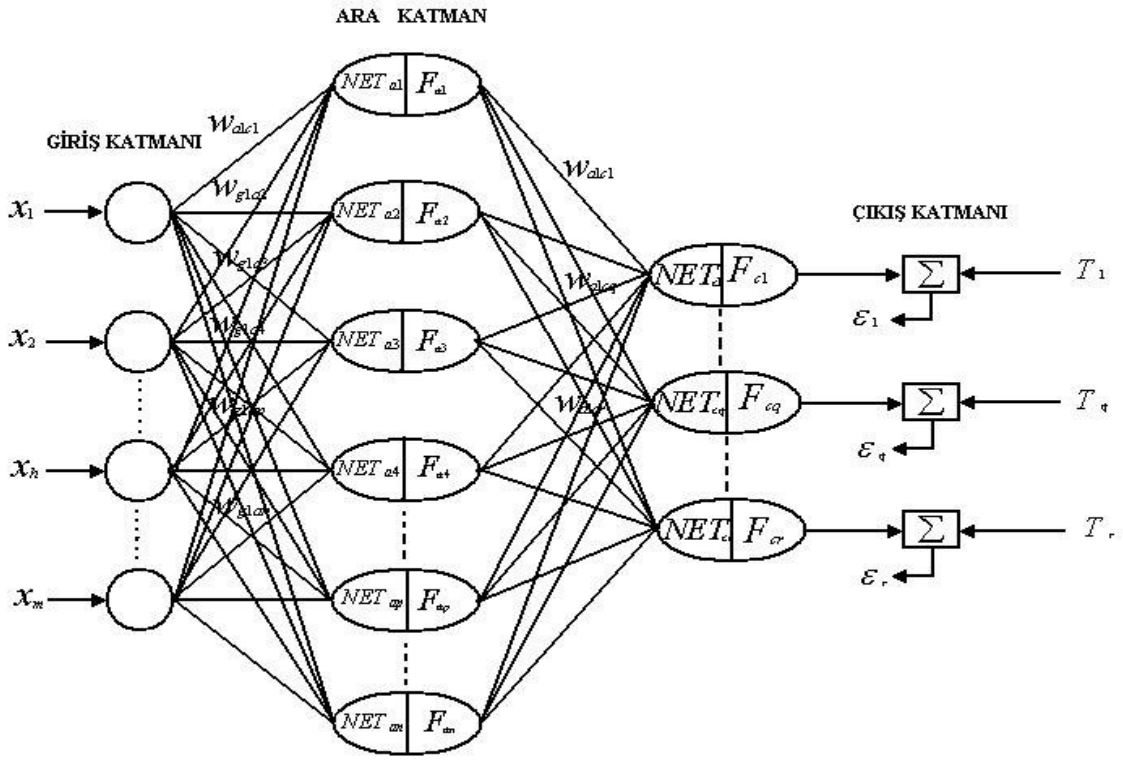
$$F_{ap} = f\left(NET_{ap}\right) \tag{3.13}$$

Kullanılacak aktivasyon fonksyonunun türevlenebilir olma şartı vardır. Çok katmanlı yapay sinir ağlarında yaygın kullanılan aktivasyon fonksyonları tablo 3.5'te gösterilmektedir.

Tablo 3.5 : Yapay sinir ağlarında kullanılan aktivasyon fonksyonları

Fonksiyon Adı	Fonksiyonun Şekli	Matematiksel İfadesi	Türevi	açıklama
Lineer Fonksyon		$f(NET)=NET$	$F^{'}=1$	Hesaplanan net giriş değeri, hücrenin çıkışı olarak kabul edilir
Sigmoid Fonksyonu		$f(NET)=\frac{1}{1+e^{-\beta NET}}$	$F^{'}=\beta \cdot F(1-F)$	B değeri keyfi şekilde değiştirilerek farklı eğimlere sahip eğriler elde edilebilir
Hiperbolik Tanjant Fonksyonu		$f(NET)=\frac{e^{NET}+e^{-NET}}{e^{NET}-e^{-NET}}$	$F^{'}(1-F)$	Hesaplanan net giriş değeri, tanjant fonksyonuna uygulanır

Şekil 3.16'da hesaplanan ifadeler ve ilgili notasyon, ayrıntılı bir şekilde gösterilmektedir.



Şekil 3.16 İleribeslemeli çok katmanlı yapay sinir ağının ayrıntılı yapısı

Ara katman ve çıkış hücreleri arasında da benzer işlemler yapılabilir. Çıkış katmanındaki hücrelerin NET girdisi aşağıdaki şekilde hesaplanabilir.

$$NET_{cq} = \sum_{p} w_{ap,cq} F_{ap} \tag{3.14}$$

Bu formülde kullanılan notasyon aşağıda açıklanmaktadır.

NET_{cq} : çıkış katmanının q. hücresine olan girdi

$w_{ap,cq}$: ara katmanın p.hücresini, çıkış katmanının q. hücresine bağlayan ağırlık

F_{ap} : ara katmandaki p. hücrenin çıkış değeri

Bulunan NET girdi, bir aktivasyon fonksyonundan geçirilerek, çıkış katmanındaki hücrelerin çıkışı, dolayısıyla yapay sinir ağının çıkışı bulunur. NET_{cq}, çıkış katmanındaki q. hücreye olan girdi, F_{cq} ise çıkış katmanının q. hücresinin aktivasyon fonksyonundan geçirildikten sonraki değerini göstermektedir.

$$F_{cq} = f\left(NET_{cp}\right) \tag{3.15}$$

Bundan sonra yapılacak işlem, çıkış hücresinin bulanan değerine karşı gelen hedef değeri kullanarak, yapılan hatanın hesaplanmasıdır. Eğer çıkış katmanındaki q. hücrenin hesaplanan değeri F_{cq} ve olması istenen çıkış değeri T_q ise, bu durumda yapılan hata ε_q aşağıdaki gibi hesaplanabilir.

$$\varepsilon_q = T_q - F_{cq} \tag{3.16}$$

Görüldüğü gibi hata ilk kez hesaplanırken, ağırlık değerlerine ihtiyaç duyulmaktadır. Bu yüzden ağırlıkların başlangıç değerlerinin saptanması gerekmektedir. Ağırlıkların başlangıç değerlerinin belirlenmesi, özen gösterilmesi gereken bir işlemdir. İlgili literatürde bu değerlerin, gaussian dağılım gösteren, büyük değerler almayan rasgele sayılar olması gerektiği söylenmektedir. Genellikle [-0.1 0.1] aralığı kullanılmaktadır.

Geriye Doğru Hesaplama : Bu aşamada ağırlıkların, yapılan hatayı azaltacak şekilde değiştirilmesi sağlanır. Bu amaç ile ilk aşamada hesaplanan hata değerleri geriye doğru yayılır ve ağırlıklara ait güncelleme kuralları bulunur. Yönteme ismini veren ‘‘backpropogation’’ işlemi bu şekilde gerçekleştirilmiş olur. Bu aşamada hata, çıkış katmanından giriş katmanına doğru yayılacağından, öncelikle çıkış katmanı ve ara katman arasındaki ağırlık değerleri daha sonra ara katman ve giriş katmanı arasındaki ağırlık değerlerine ait kurallar bulunur.

Ara katman ve çıkış katmanı arasındaki ağırlıkların değiştirilmesi : Çıkış katmanındaki q. hücre için yapılan hata ;

$$\varepsilon_q = T_q - F_{cq} \tag{3.17}$$

Hatanın negatif değerinden kurtulmak için karesel hata tanımlansın;

$$\varepsilon_q^2 = \left(T_q - F_{cq}\right)^2 \tag{3.18}$$

Delta kuralına göre, ağırlıklardaki değişim, karesel hatanın bu ağırlığa göre değişimiyle orantılıdır. Diğer bir ifade ile hatayı minimum yapabilmek içini ağırlıklar hatanın negatif gradyanı yönünde değiştirilmelidir. η terimi öğrenme katsayısıdır. Ara katmanındaki p. hücreyi, çıkış katmanındaki q. hücreye bağlayan ağırlık değeri $w_{ap,cp}$ ile gösterilsin. Giriş katmanında m adet, ara katmanda n adet ve çıkış katmanında r adet hücre olduğu hatırlanarak, ara katman çıkış katmanı arasındaki ağırlıkların değişimi;

$$\Delta w_{ap,cq} = -\eta_{ap,cq} \frac{\partial \varepsilon_q^2}{\partial w_{ap,cq}} \tag{3.19}$$

olacaktır.

Gradyanı hesaplayabilmek için zincir kuralı işletilmelidir**;**

$$\frac{\partial \varepsilon_q^2}{\partial w_{ap,cq}} = \frac{\partial \varepsilon_q^2}{\partial F_{ap}} \cdot \frac{\partial \varepsilon_{cq}}{\partial NET_{cq}} \cdot \frac{\partial NET_{cq}}{\partial w_{ap,cq}} \tag{3.20}$$

$$\frac{\partial \varepsilon_{cq}^2}{\partial F_{ap,cq}} = -2\left(T_q - F_{cq}\right) \tag{3.20.a}$$

Eşitlik 3.20'de ki ikinci terim kullanılan aktivasyon fonksyonuna göre değişir. Aktivasyon fonksyonunun tanjant hiperbolik veya sigmoidal olması durumlarında ;

$$Tanh \Rightarrow \frac{\partial F_{cq}}{\partial F_{cq}} = \left(1 - F_{cq}^2\right) \quad sigmoidal \Rightarrow \frac{\partial F_{cq}}{\partial NET_{cq}} = F_{cq}\left(1 - F_{cq}\right) \tag{3.20.b}$$

Eşitlik 3.20'de ki son terim ise;

$$\frac{\partial NET_{cq}}{\partial w_{ap,cq}} = F_{ap} \tag{3.20.c}$$

Sigmoildal aktivasyon fonksyonu kullanıldığında, eşitlik 3.20 yeniden yazılırsa ;

$$\frac{\partial \varepsilon_q^2}{\partial w_{ap,cq}} = -2\left(T_q - F_{cq}\right) \cdot F_{cd}\left(1 - F_{cq}\right) \cdot F_{ap} \tag{3.21}$$

Böylelikle ara katman ve çıkış katmanı arasındaki ağırlıkların nasıl değiştirilmesi gerektiği bulunmuş olur ;

$$\Delta w_{ap,cq} = -\eta_{ap,cq} \frac{\partial \varepsilon_q^2}{\partial \varepsilon_{ap,cq}} = -\eta_{ap,cq} \cdot \left(T_q - F_{cq}\right) \cdot F_{cq}\left(1 - F_{cq}\right) \cdot F_{ap} \tag{3.21.a}$$

$$w_{ap,cq}\left(N+1\right) = w_{ap,cq}\left(N\right) + \Delta w_{ap,cq} \tag{3.21.b}$$

Giriş katmanı ve ara katman arasındaki ağırlıkların değiştirilmesi : Ara katman-çıkış katmanı arasındaki ağırlıklardan farklı olarak, giriş katmanı-ara katman arasındaki ağırlıkları hesaplarken, herbir hücrenin hatası değil, toplam hata göz önünde bulundurulmalıdır.

$$\varepsilon^2 = \sum_{q=1}^{r} \varepsilon_q^2 \tag{3.22}$$

Giriş katmanındaki h. hücreyi, ara katmanda ki p. hücreye bağlayan ağırlık değeri $\Delta w_{gh,ap}$ ile gösterilsin. Çıkış katmanında r adet hücre olduğu hatırlanarak, giriş katmanı ve ara katman arasındaki ağırlıkların değişimi;

$$\Delta w_{gh,ap} = -\eta_{gh,ap} \frac{\partial \varepsilon^2}{\partial w_{gh,ap}} = -\eta_{gh,ap} \sum_{q=1}^{r} \frac{\partial \varepsilon_q^2}{\partial w_{gh,ap}} \tag{3.23}$$

olur.

Gradyanı hesaplayabilmek için zincir kuralı işletilmelidir;

$$\frac{\partial \varepsilon^2}{\partial w_{gh,ap}} = \sum_{q=1}^{r} \frac{\partial \varepsilon_q^2}{\partial F_{cd}} \cdot \frac{\partial F_{cq}}{\partial NET_{cd}} \cdot \frac{\partial NET_{cq}}{\partial F_{ap}} \cdot \frac{\partial F_{ap}}{\partial NET_{ap}} \cdot \frac{\partial NET_{ap}}{\partial w_{gh,ap}} \tag{3.24}$$

$$\frac{\partial \varepsilon^2}{\partial F_{cd}} = -2\left(T_q - F_{cq}\right) \tag{3.24.a}$$

$$\frac{\partial F_{cq}}{\partial NET_{cq}} = F_{cq}\left(1 - F_{cq}\right) \tag{3.24.b}$$

$$\frac{\partial NET_{cq}}{\partial F_{ap}} = w_{ap,cq} \tag{3.24.c}$$

$$\frac{\partial F_{ap}}{\partial NET_{ap}} = F\left(1 - F_{ap}\right) \tag{3.24.d}$$

$$\frac{\partial NET_{ap}}{\partial w_{gh,ap}} = x_h \tag{3.24.e}$$

Hesaplanan terimler ile eşitlik 3.24 yeniden yazılırsa;

$$\frac{\partial \varepsilon^2}{\partial w_{gh,ap}} = \sum_{q=1}^{r} 2\left(T_q - F_{cq}\right) \cdot F_{cq}\left(1 - F_{cq}\right) \cdot w_{ap,cq} \cdot F_{ap}\left(1 - F_{ap}\right) \cdot x_h \tag{3.25}$$

Böylelikle giriş katmanı-ara katman arasındaki ağırlıkların güncellenme kuralı ;

$$\Delta w_{gh,ap} = -\eta_{gh,ap} \sum_{q=1}^{r} \left(T_q - F_{cq}\right) \cdot F_{cq}\left(1 - F_{cq}\right) \cdot w_{ap,cq} \cdot F_{ap}\left(1 - F_{ap}\right) \cdot x_h \tag{3.25.a}$$

$$w_{gh,ap}\left(N+1\right) = w_{gh,ap}\left(N\right) + \Delta w_{gh,ap} \tag{3.25.b}$$

olarak bulunur.

3.4.4. Tasarımda karşılaşılabilecek problemler

Yapay sinir ağının ileriye doğru hesaplama kısmında, ilk iterasyonun yapabilmesi için ağırlıklara değer atamak gerekmektedir. Ağırlıklara atanacak başlangıç değerleri ağın performansını yakından ilgilendirmektedir. İlgili literatürde bu değerlerin, gaussian dağılım gösteren, büyük değerler almayan rastgele sayılar olması gerektiği söylenmektedir. Genellikle [-0.1 0.1] aralığı kullanılmaktadır (Haykin 1994).

Öğrenme katsayısının belirlenmesi, istenilen hata değerine ulaşması için yapılması gerekli iterasyon sayısını etkilemektedir. Öğrenme katsayısı [0 1] aralığında seçilebilir. Bu değer sıfıra yakın seçildiğinde istenilen hata değerine ulaşmak için daha fazla iterasyon yapılması gerekir. Diğer bir ifade ile yakınsama yavaş olur. Öğrenme katsayısının değeri arttıkça yapılması gerekli iterasyon sayısı azalır. Fakat büyük değerler yapay sinir ağının öğrenmesi yerine ezberlemesine neden olmaktadır. ‘‘Yapay sinir ağı öğreniyor mu yoksa ezberliyor mu?’’ sorusunun cevabını bir örnek ile açıklamak yararlı olacaktır. Eğitim verileri için yapılacak 10^{-4} mertebesindeki bir hatanın uygulama için yeterli olacağını düşünelim. Bu durumda test verileri için yapılacak $[10^{-3} \quad 10^{-5}]$ aralığındaki hatalar seçilen öğrenme katsayısının uygun olduğunu gösterir. Fakat test verileri için 10^{-1} mertebesinde bir hata yapılıyorsa bu durumda yapay sinir ağının öğrenmediğini, bunun yerine ezberlediğini söyleyebiliriz. Bu durumda öğrenme katsayısının küçültülmesi gerekir. Görüldüğü gibi öğrenme katsayısı küçük seçildiğinde yakınsama hızı düşmekte, büyük seçildiğinde ise öğrenme problemi doğmaktadır. Bu sorunu aşmak için öğrenme katsayısı adaptif olarak değiştirilebilir. Çözüm noktasından uzakta öğrenme katsayısı büyük seçilerek yakınsama hızlandırılabilir. Çözüm noktası civarında ise katsayı azaltılarak öğrenme problemi aşılabilir.

Hatanın değişiminde büyük salınımlar görülüyorsa, öğrenme katsayısının değeri küçültülmelidir (GÜLTEKİN 2002).

Momentum katsayısı, yerel minimum noktalarında takılmayı önlemektedir. Ağırlıklardaki değişimin bir önceki değerleri momentum katsayısı ile çarpılarak yerel minimumlarda sıçrama sağlanır. Momentum katsayısı [0 1] aralığında seçilebilir. Bu katsayı küçük seçildiğinde daha iyi sonuçlar alındığı gözlenmiştir (GÜLTEKİN ve ark. 2003)

Yapay sinir ağlarında kullanılan aktivasyon fonksyonları sebebi ile giriş ve çıkış değerlerinin ölçeklendirilmesi gerekebilir. Örneğin giriş kümesinde bulunan 10, 100 ve 1000 değerleri için tanjant hiperbolik aktivasyon fonksyonu 1 değerini üretecektir. Aktivasyon fonksyonlarının doymasını ‘‘saturation’’ önlemek için bu değerlerin ölçeklendirilmesi gerekmektedir. Çıkışta ise ölçeklendirme işlemi tersine çevrilmelidir. Lineer aktivasyon fonksyonunu kullanan ağlar için ölçeklendirme işlemine gerek yoktur.

4. UYGULAMA

Bu bölümde öncelikle klasik sistemlerin dezavantajlarına, daha sonraki kısımda kısaca tasarıma değinilecek, son kısımda ise ölçüm basamakları anlatılarak yapılan deneme sonuçları verilecektir.

4.1. Klasik Sistemlerin Dezavantajları

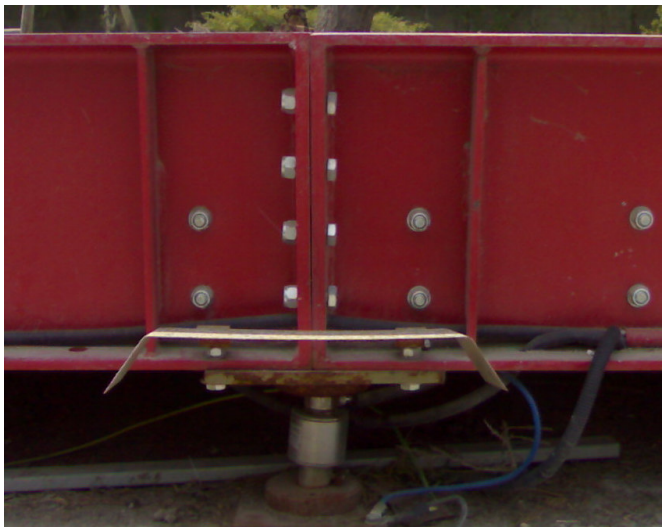
Kantar kurulumu, zeminin uygun şekilde hazırlanması ile başlayan bir süreçtir. Kurulum yapılacak zeminin, düzleştirilerek engebelerin ortadan kaldırılması, oluşabilecek toprak kayması ya da çökme ihtimallerine karşı güçlendirilmesi gerekmektedir. Aksi takdirde zeminde oluşacak deformasyon, sistemin işleyemez hale gelmesine sebep olabilir. Bu nedenle kurulacak herhangi tip kantar için bu işlem, kurulumun değişmez bir parçasıdır.

Bu adımı iskeletin kurulumu takip etmektedir. İskelet montajı, oldukça hassas ve dikkat gerektiren bir işlemdir. Mekanik düzende oluşacak bir hata, kalibrasyon esnasında çözülemeyecek problemlere sebep olabilir. Ölçüm sonucunun doğruluğu, yük hücreleri birbirlerine bağlı oldukları için, üzerlerine uygulanan baskının homojen dağılımıyla orantılıdır. Eğer iskelet, düzgün monte edilmemişse, sisteme yük uygulandığında, yük hücreleri üzerinde gerekli baskı oluşmayacaktır. Yada herhangi bir yük hücresi üzerinde gereğinden fazla baskı oluşacaktır. Böylece, baskının homojen dağılımı imkansız hale gelir. Şekil 4.1'de montajı hatalı yapılmış örnek bir iskelet desteği görülmektedir.



Şekil 4.1 Hatalı yapılmış montaj

Bu durum yük hücrelerinin montajı içinde geçerlidir. Hatalı yerleştirilmiş yük hücreleri, gerekli baskıya maruz kalamıyacaklarından, sistemin yanlış tartım yapmasına sebebiyet vereceklerdir.



Şekil 4.2 Eğimli monte edilmiş yük hücresi

Şekilde görülen sistem için, meydana gelen kayma sonucu, baskı dengesiz dağılmaktadır. Bu tarz mekanik hatalar sistem kurulumunu hassas kılmakta ve yük hücreleri çevrim sonuçlarını etkileyerek ölçümün sistem üzerinde farklı noktalarda farklı değerler almasına sebep olmaktadır.

Mekaniksel hataların yanı sıra sistemin diğer dez avantajları da şu şekilde sıralanabilir ;

- Zaman içerisinde, trimpotların ayar seviyesi değişir ve değerlerinde kaymalar oluşur. Buda sistem sonucunun doğruluğunu etkileyerek, yanlış ölçüm yapılmasına sebebiyet verir.
- Kantar hacminden daha küçük bir yük tartılmak için kantar üzerine çıkar ve yükün dengeli dagılım koşulu ortadan kalkarsa yanıltıcı sonuçlar elde edilir.
- Kalibrasyon esnasında hassas ayarın trimpot değerleri ile yapılma işlemi zahmetli bir iştir.
- Yük hücrelerinden birinde meydana gelebilecek bir arıza, birbirlerine bağlı diğer yük hücrelerinide etkiler.
- Değiştirilmek istenen bir yük hücresinin, diğer yük hücrelerine uyumlu olma zorunluluğu, bakım ve servis olayını maliyetli kılmaktadır.

4.2. Tasarım

Değinilen olumsuzlukların giderilebilmesi ya da etkilerinin minimize edilebilmesi için, sisteme farklı bir mantıkla yaklaşılmaya çalışılacaktır. Bu amaçla, sistem, birbirinden bağımsız parçalardan oluşan bir yapı olarak düşünülecek, sonuç, bağımsız bu

parçaların, yapay sinir ağı ile verilerinin işlenerek elde edilmesiyle sağlanmaya çalışılacaktır.

Proje kapsamında takip edilecek yol, 3 adımdan oluşmaktadır ;

- Elektronik indikatör tasarımı
- Donanım yazılımı
- Bilgisayar ortamında görsel arayüz tasarımı

Bu süreçte ilk adım uygun donanımın gerçeklenmesidir.

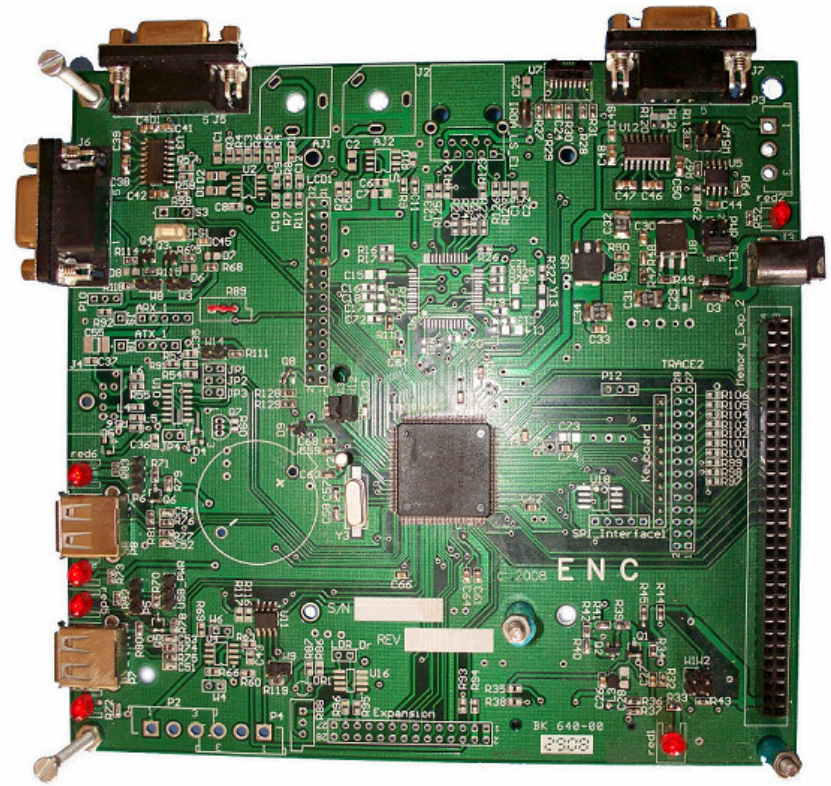
4.2.1. Donanım

Donanım, iki farklı tasarımdan oluşmaktadır. Birinci tasarım geliştirme platformu diye adlandırabileceğimiz, ARM mimarisine sahip bir mikrodenetleyici temel alınarak oluşturulmuş ana kart, ikinci tasarım ise uygulamaya yönelik çevrimin yapılabilmesi için tasarlanan analog/dijital çevrim modülüdür.

Ana kart, NXP firmasının LPC2378 kodlu ARM7 mimarisine sahip mikrodenetleyicisi temel alınarak tasarlanmıştır. Kullanılan mikrodenetleyicinin USB, UART, IRDA, I2C, SPI gibi çok çeşitli donanımsal protokolleri barındırması, 512 Kbyte gibi endüstriyel uygulamalar için yeterli kabul edilebilecek program hafızasına ve 256 Kbyte gibi bir RAM' e sahip olması, MMC / SD kart uygulamaları için dahili protokolleri ve donanımı içermesi, 32 bitlik bir denetleyici olması sayesinde toplama, çarpma işlemlerinde avantaj sağlaması, bu mikrodenetleyiciyi birinci tercih sebebi

kılmıştır. Teknolojik açıdan gelişmelere adapte olabilecek donanımlara sahip olması, uzun ömürlü bir tasarım olmasını sağlamaktadır. Donanımsal bir değişiklik olmasın düşüncesi ile proje esnasında kullanılmayacak özelliklerde düşünülerek devre üzerine eklenmiştir. Bu özelliklerden bazıları şu şekilde sıralanabilir;

- 1 x Harici EEPROM (en çok 256 KByte)
- 1 x SD/MMC Kart soketi
- 1 x IrDA arayüz dönüştürücü
- 3 x UART
- 2 x USB portu (Sunucu ve cihaz)
- 2 x SPI arayüzü
- 1 x RS485 / RS422 dahili dönüştürücü
- CAN protokol sürücüsü ve dönüştürücüsü
- 1 x 4*20 LCD Ekran
- 1 x 25 tuş klavye
- PS/2 klavye ve mouse girişi
- Seri kablosuz alıcı vericiler için genişletme yuvaları
- 100 Mbit Ethernet Portu
- Modül kartlar için ek genişletme yuvaları
- Genişletilmiş adres bağlantı yolları
- PWM çıkışları
- AUDIO IN ve AUDIO OUT
- Saat için pil yuvası



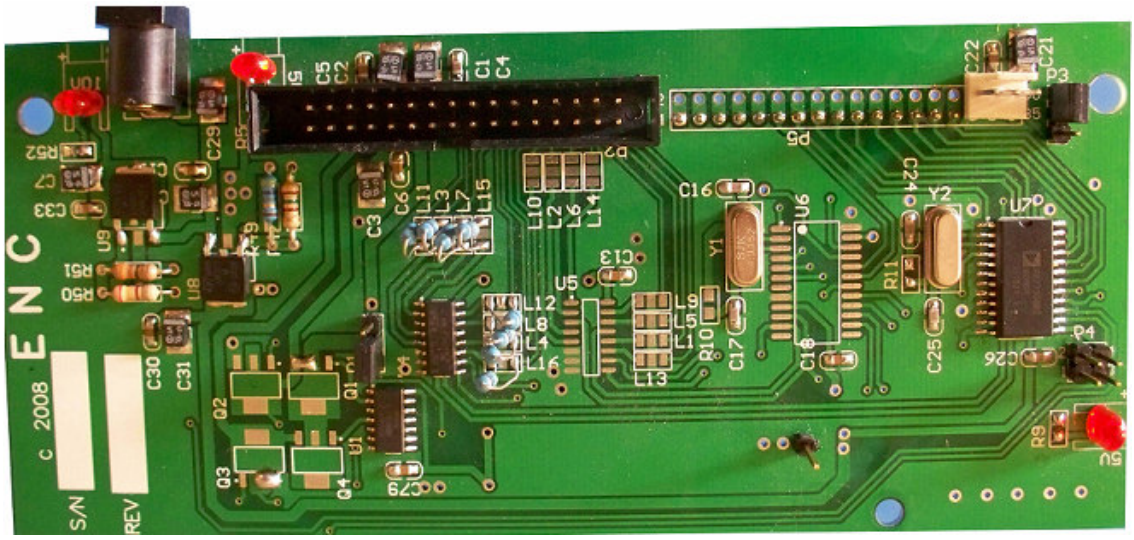
Şekil 4.3 Ana kart görünümü

Uygulamada, veriler bilgisayar üzerinde işlenmekte, aktarım için seri port yeterli olmaktadır. LCD ekran, aktarım verilerinin gösterimi ve bazı uyarılar için kullanılmıştır. Bu nedenle diğer donanımlar, proje esnasında gerekli olmadığından aktif edilmemiştir.

Tasarımın en kritik basamagını, çevirici modül kartı oluşturmaktadır. Modül kart, hem devre tasarımı hemde kullanılan malzeme açısından, sistem performansını direk etkilemektedir. Öyle ki ölçülecek sinyallerin, mikro volt mertebesine kadar indiği düşünülecek olursa, kart üzerindeki komponentlerden, sinyal bağlantı yollarına kadar her bir aşama ayrı bir önem taşımaktadır. Bu nedenle, hem sistem tasarımının yenilenebilme ihtimali, hemde, sistemin ana kart ve diğer dış etkenlerin gürültülerinden izole

edilebilmesi için kart ayrı bir modül şeklinde tasarlanmıştır. Bu, olası herhangi bir problemin, sadece modül üzerindeki değişikliklerle giderilebilmesi avantajını sağlamaktadır. Tasarım aşamasında, kart üzerindeki analog komponentlerin besleme kaynağı diğer komponentlerden ayrılmıştır. Kart yüzeyinde ki analog sinyal bağlantıları, en kısa yoldan ve yüzey değiştirilmeden yapılmaya çalışılmıştır. Kartın bağlantıları tamamlandıktan sonra kalan tüm yüzey, devre toprağı ile kaplanmıştır. Proje süresi göz önünde bulundurulduğu için kartın üretimi Türkiye şartlarında gerçekleştirilmiştir. Bu nedenle çok katlı olması düşünülen kart için, analog sinyal yolları ara katlarda koruma altına alınamamış, tasarım iki katla sınırlı kaldığından, yollar dış gürültü kaynaklarına karşı birazda olsa duyarlı kalmıştır.

Modül kart üzerinde ki en kritik komponent ise analog/dijital çevrici AD7730 entegresidir. Tasarım ne kadar hassas ele alınırsa alınsın, çevirim için kullanılan entegre, beklenti değerlerini sağlayacak oranda hassas değilse, doğru sonuçların elde edilmesi imkansız olacaktır. Bu nedenle kullanılacak entegre, istenilen değerleri sağlayıp sağlamadığı göz önüne alınarak seçilmelidir. Çevrime tabi yük hücresi sinyallerinin değişim oranı çok düşük olacak, elde edilen sinyal değeri mikro voltlarla tabir edilecek ve çevrim hassasiyetinin minimum 15 bit olacağı göz önünde bulundurulacak olursa, seçim için çokta fazla bir alternatifin olmadığı görülmektedir. Araştırmalarımız sonucu, özellikle bu tür sistemler için tasarlanmış, hassasiyetini ispatlamış ve profesyonel ürünlerde de sıkça karşımıza çıkan, Analog Device firmasinin üretmiş oldugu AD7730 entegresi güzel bir seçim olacaktır. Bu entegre, 24 bit'lik bir analog/digital çevrici olup, yüksek hassaiyetlerde sinyal kararlılığı sağlanabilirse kesin sonuçlar vermekte, elde edilmeye çalışılan çözünürlük oranlarını rahatlıkla sağlayabilmektedir. Yapısal olarak Sigma – Delta çeviriciler kategorisinden olup, dahili FIR ve $Sinc^3$ filtreleme yapabilmektedir. Buda yüksek frekanslı salınımlar için sistem kararlılığını arttırmaktadır. Şekil 4.4'te bu entegre baz alınarak yapılmış kart tasarımı görülmektedir.



Şekil 4.4 Modül kart görünümü

Modül kart üzerinde, sinyaller çeviriciye bir analog seçici üzerinden verilmiş, bu sayede her bir yük hücresi ayrı ayrı ölçülerek bir birine olan etkileri yok edilmeye çalışılmıştır. Bu mantıkla, yük hücrelerinin besleme değerleri, marka farklılıkları, teknik özellikleri, yerleşim farklılıkları gibi bir çok parametreye olan bağımlılık ortadan kalkmakta yada azalmaktadır. Böylece, yük hücreleri sayısı kadar farklı ölçüm düzeneği varmış gibi sonuçlar elde edilecek, asıl çevrim sonucuna her bir yük hücresinin sonuçları üzerinden ulaşılmaya çalışılacaktır. Bu işlem, her sinyalin farklı bir çeviricide ölçümü ile aynı anlama gelmektedir. Sinyaller için ayrı birer çeviricinin kullanılmama nedeni ise, AD7730 entegresinin birim fiyatinin toplam maliyeti en fazla etkilyen parametre olmasından ve proje maliyetinin kabul edilebilir sınırlar içerisinde kalmasını sağlamak içindir. Analog seçici kullanımının oluşturduğu dezavantaj, yük hücrelerinin ayrı ayrı ölçüm sonuçlarının toplamı alınacağı için sistem cevbında meydana gelen gecikme olacaktır. Fakat bu gecikme sistem performansını etkileyecek seviyede olmadığından problem oluşturmamaktadır.

Çevrim sonucu her ne kadar 24 bit olsada, en düşük bitler matematiksel olarak ihmal edilebilecek derecede kücük değerleri temsil etmektedir. Pratik anlamda ise bu değerler, uygulanan sistem için zaten kararsız durumdadır. Bu kararsızlıkta en önemli etken, yük hücrelerinin çözünürlük sınırları olmakla beraber, ortam sıcaklığı, nem, atmosfer basıncı ve benzeri ortam koşulları da bu kararsızlıkta etkili birer parametre konumundadır. Bu nedenle ölçüm sonuçları son 4 bit ihmal edilerek, 20 bit şeklinde bilgisayar ortamına aktarılmaktadır.

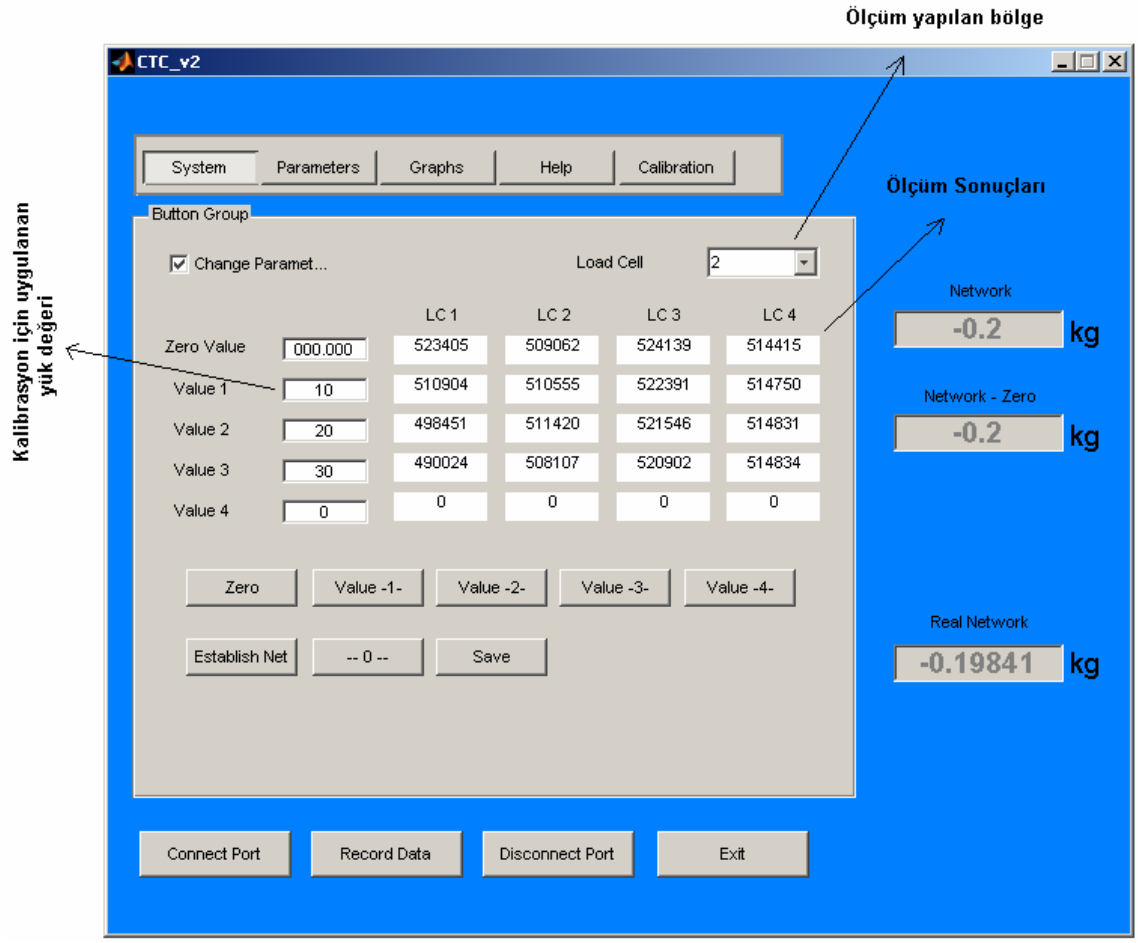
4.2.2. Donanım yazılımı

Donanım için hazırlanan yazılım, yük hücreleri ölçümlerini yaparak, verilerin bilgisayar ortamına aktarılmasını sağlamaktadır. Bununla beraber donanım yazılımı, geliştirme platformunun harici donanımlarla haberleşebilmesi, analog ölçüm için gerekli parametrelerin ayarlanması, LCD ekran konfigurasyonu, bazı protokoller için gerekli dönüşümler gibi düşük seviye ayarları da içermektedir.

4.2.3. Yazılım

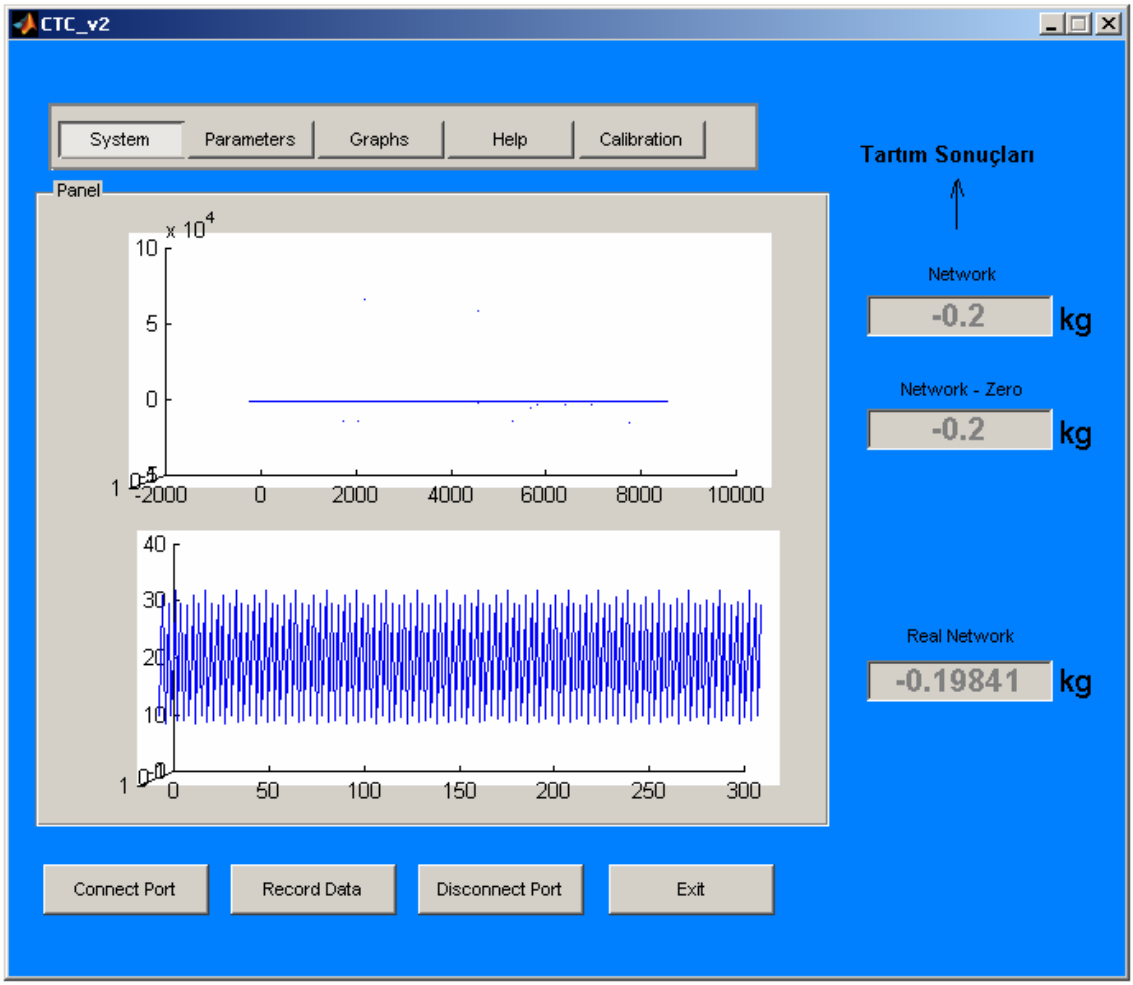
Veriler, her bir yük hücresi için 20 bitlik formlarda bilgisayar ortamına aktarılmaktadır. Verilerin işlenmesi ve sinir ağı modelinin oluşturulması için, Matlab GUI (Graphical User Interface) yardımıyla windows platformu üzerinde bir arayuz programı hazırlanmıştır. Program ölçüm işlemini gerçekleştirebilmek için öncelikle, ölçüm verileri ile bir sinir ağı oluşturarak, ağın eğitilmesini sağlamaktadır. Bu işlem kalibrasyon işlemi olarak adlandırılabilir.

Kalibrasyon için sisteme uygulanan yük değerleri Şekil 4.5'te görüldüğü gibi girilerek, bu değerlere ait ölçüm sonuçları uygulamaya tabi sistem için kaydedilmektedir.



Şekil 4.5 Arayüz programı genel görünümü

Kaydedilen veriler ile sistem cevabının elde edilmesine yönelik lineer bir sinir ağı modeli oluşturulur. Oluşturulan model eğitilerek, modele ilişkin eğitim, ölçüm, sistem cevabı ve hata gibi raporlar grafiksel olarak yine program aracılığıyla gösterilmektedir.



Şekil 4.6 Arayüz programı rapor görünümü

Sonraki adım, oluşturulmuş ve egitilmiş model icin, sistem simulasyonu sağlanarak normal ölçümlerin yapay sinir ağı cevaplarının elde edilmesi, ve ölçüm sonuçlarının gösteriminden oluşmaktadır.

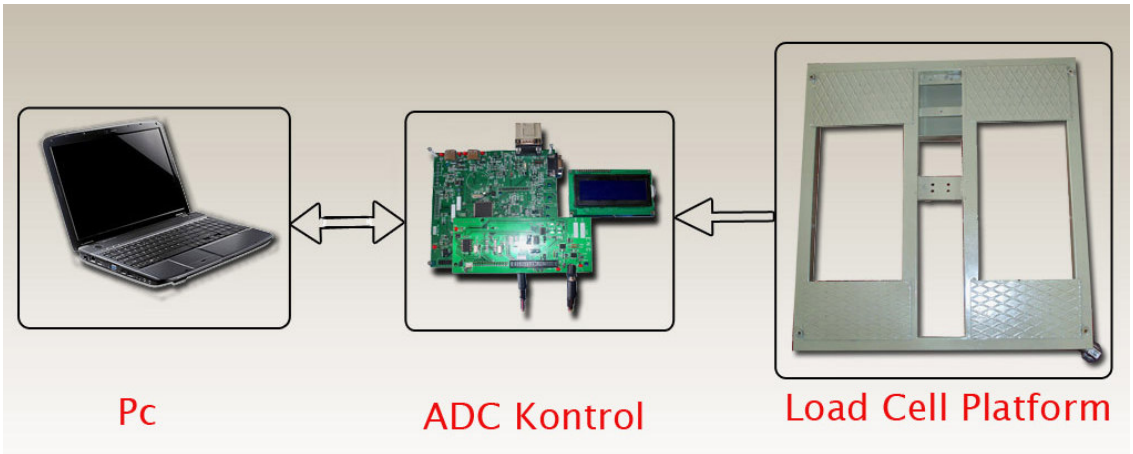
Bu adımda, sisteme anlık uygulanan yükler için gerekli ölçümler yapılarak, bu değerler modele uygulanır. Yapay sinir ağı modeline uygulanan verilere karşı oluşan cevap ölçüm sonucu olarak gösterilmektedir..

4.3. Simulasyon ve Sonuçlar

Klasik indikatör yapıları için tek bir sinyal girişi mevcuttur. Pratikte, tonajı yüksek, yada ölçüm alan ihtiyacı büyük sistemler için birden çok yük hücresi kullanılmaktadır. Bu ise tek girişli bir indikatör için mevcut tüm yük hücrelerinin çıkışlarının birleştirilerek indikatöre giriş olarak sağlanmasını zorunlu kılmaktadır. Sadece bu durum bile, sistemde kullanılacak yük hücreleri için ayni tip ve marka olma zorunlulugunu, besleme gerilimlerinin, montaj koşullarının, hatta ve hatta yol dirençlerini etkileyeceği için aktarım kablolarının bile aynı olmasını zorunlu kılmaktadır. Yapılan tasarımın ilk adımı olarak indikatörde, her bir yük hücresi bir analog seçici ile çeviriciye uygulandığı için bu tarz sorunlar elimine olacaktır.

İkinci temel nokta, ölçüm sonuçlarının toplamlarının liner olmadığını varsaymak olmuştur. Elde edilecek sonucun lineer olabilmesi için yük hücrelerine birer ağırlık çarpım katsayısı belirlenecektir. Böylece sistem, farklı parametre ve yapıya sahip yük hücrelerinin çarpım katsayıları ile dengelenmesi sonucu lineer davranış gösterecektir. Bunun için bilgisayar ortamına aktarılan verilerle lineer bir sinir ağı modeli oluşturulacak, cevap ağdan elde edilmeye çalışılacaktır.

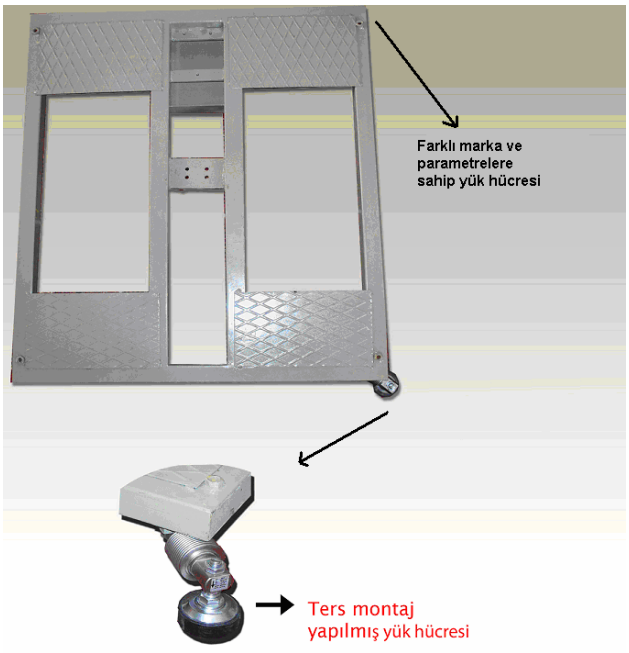
Tasarlanan sistemin performansını test etmek ve sonuçları değerlendirebilmek için basit bir ölçüm düzeneği hazırlanmış, sistem mekaniğinin çok büyük hacim kaplaması ve kurulumunun zorluğu nedeniyle, kantar mekaniği yerine prototip bir model oluşturulmuştur. Aynı tipte 3 adet olmak üzere, 4 adet 100 kg kapasiteli yük hücreleri kullanılarak toplamda 400 kg kapasiteli basit bir ölçüm düzeneği elde edilmiştir. Şekil 4.7'de oluşturulan ölçüm düzeneği ilişkisel olarak gösterilmektedir.



Şekil 4.7 Hazırlanan ölçüm düzeneği bloğu

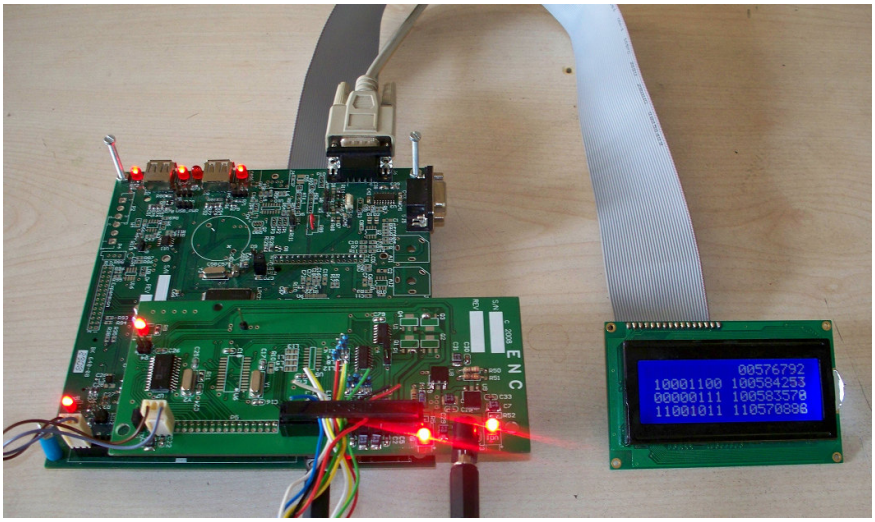
Prototipin mekaniksel hataları simule edebilmesi için;

- Yük hücrelerinden birisi farklı özellik ve markada seçilmiş
- Aynı tip yük hücrelerinden birisi, platformun dışında kalacak şekilde montaj yapılmış
- Tüm yük hücreleri rastlantısal yerleştirilerek, platform ayaklarının zemine basma yüklerinin eşit olmaması sağlanmaya çalışılmıştır.



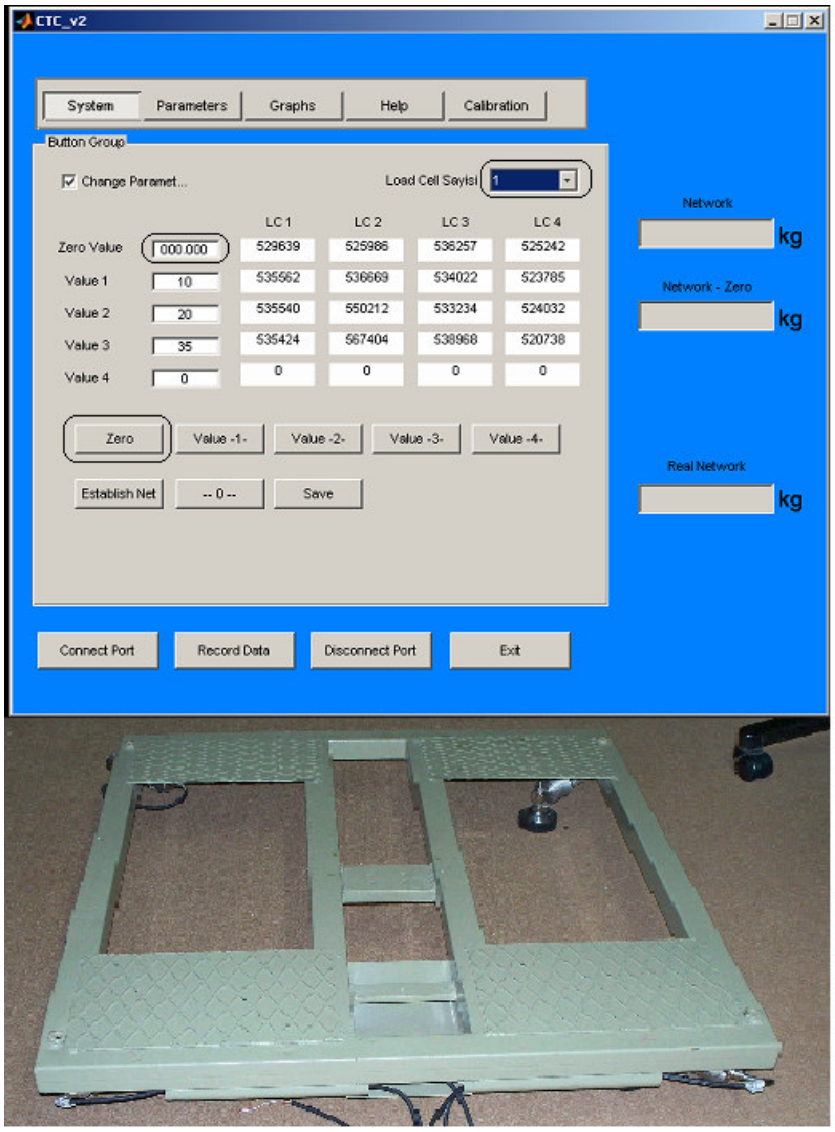
Şekil 4.8 Platform hataları

Montajın basitleştirilmesi ve kullanılan kablo sayısının minimum tutulabilmesi için 5V ortak besleme kullanılmaktadır.



Şekil 4.9 Ölçüm işlemi

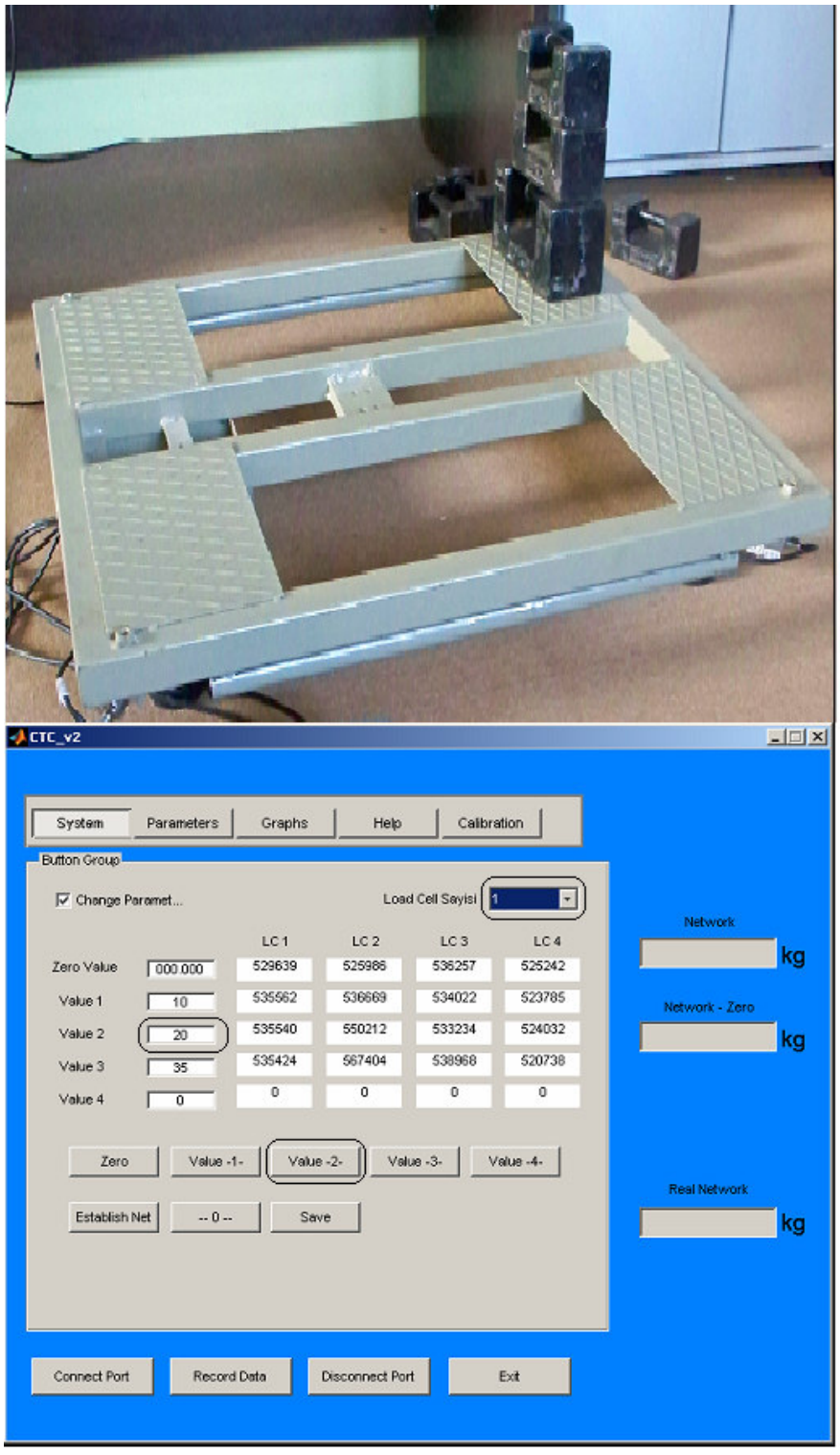
Ölçüm için hazır hale getirilen sistemde ilk adım "ölü ağırlığın" belirlenmesi olacaktır. Sistem boş konumda iken, yük hücreleri üzerinde, yerleştirilmiş olan platformdan kaynaklı bir baskı olacak, bu baskı sayesinde yük hücreleri bir çıkış üretecektir. Sistemin bu konumda vermiş olduğu ölçüm değeri, 'ölü ağırlık' olarak adlandırılır. Sistemin ölü ağırlığının belirlenmesi için, sistemin boş ve stabil konumda kalması sağlanarak, ilk ölçümler yapılır.



Şekil 4.10 Sistem ölü ağırlığının ölçümü

Sistem cevabı, küçük mertebelerde de olsa salınım yapmakta ve stabil kalmamaktadır. Salınımın nedeni şebeke, ortam koşulları, yük hücresi hassasiyeti ve tasarımda kullanılan komponentlerin etkisi olarak düşünülebilir. Ayrıca 20 bit'lik çözünürlük değeri, ondalık olarak 2,097,151 sayısına denk düşmekte ve bu değer çözünürlük için birim adımı ifade etmektedir. Bu ise kabaca 2 tonluk bir sistem için 1 gramlık hassasiyet demektir. Kapasitesi 400 kg olan ölçüm düzeneğinde bu çözünürlük 20 mg'a denk düşmektedir. Bu değerin kullanılan komponent ve yük hücrelerinin hassasiyeti göz önünde bulundurulduğunda ölçülemeyeceği açıktır. Bu yüzden salınımların olması normaldir. Yalnız salınımlar yüzler mertebesinde olduğu zaman sistem cevabını etkilemektedir. Ticari ürünlerde bu salınımın giderilip, sinyalin stabilleştirlmesi için çeşitli yazılımsal filtreler tasarlanmaktadır. Yapılan tasrımda, yapay sinir ağı etkisinin net bir şekilde görülmesi ve salınımlarında ağ ile işlenebilmesi için filtre tasarlanmamış, sonuçlar olduğu gibi ağa uygulanmıştır. Yalnız etkinin hesaba katılabilmesi için ölçüm değerleri, belirli zaman aralıklarında alınarak, ölçülen yük için birden fazla değer kaydedilmektedir.

İkinci adım, kalibrasyon için sisteme uygulanan belirli bir yüke karşı ölçüm sonuçlarının konumsal olarak kaydedilmesi olacaktır. Sistem 5 farklı bölge olarak, 4 köşe ve bir merkez düşünüldüğü için, uygulanan yük, herbir farklı bölgede ölçülerek kaydedilecektir. Yükün hangi bölgeye uygulandığı, arayüz aracılığıyla belirlenir ve kayıt o bölge için yapılır. Şekil 4.11'de 20 kilogramlık yükün sisteme uygulanması ve yazılım aracılığıyla değerin ölçülmesi görülmektedir.



Şekil 4.11 20 kg'lık yük ile yapılan örnek ölçüm

Aynı yük ile tüm bölgeler için ölçümler yapılarak 2. adım tamamlanır. 3. ve 4. adımlar, 2. adımın tekrarı niteliğindedir. Uygulanacak farklı 2 yükle bu adımlar da tamamlanarak sinir ağı modelinin oluşturulması için tüm ölçümler yapılmış olmaktadır.

Klasik uygulamalarda lineer sistem için eğimin bulunması amacıyla iki nokta belirlenerek ölçümler yapılmaktadır. Birinci nokta ölü ağırlık denilen sistemin sıfır konumudur. İkinci nokta, eğimde ki sapmanın minimum olabilmesi için kantar kapasitesinin en azından %70'i oranında bir yükle belirlenmektedir. Uygulamada sistem kalibrasyonu için kullanılan yükler, 10 kg, 20 kg ve 35 kg seçilerek ölçümler yapılmıştır. Kullanılan en büyük değer 35 kg sistem kapasitesinin %8,75'ine denk düşmekte, eğimin belirlenmesi tamamen sinir ağının performansına bırakılmakta, böylece kalibrasyon işleminin basitleştirlmesi amaçlanmaktadır.

Sistemin kalibrasyonu için 3 farklı yük sisteme uygulanmıştır. Sistemin sıfır değeri de sisteme uygulanmış bir yük olarak kabul edilecek olursa, toplamda 4 farklı yük değeri için alınan ölçüm sonuçları ile sinir ağı modeli oluşturulur. Yük hücrelerinin kendi karakteristikleri lineer olduğu için ağ yapısıda lineer seçilmiştir.

Model için giriş verileri, ve girişlere denk düşen hedef veriler matris formunda düzenlenir.

X : 1,2,3,4,5
Y : 0,1,2,3 (0 kg, 10 kg, 20 kg, 35 kg yükler için)
n :1,2,3,…N Toplam yapılan ölçüm
PX : X. Bölge için giriş verileri vektörü
ÖlçümXYn : X. Bölge için Y yük değerinin n. ölçüm sonucu

olmak üzere her bölge için ;

PX = [ÖlçümX01 ÖlçümX11 ÖlçümX21 ÖlçümX31 ÖlçümX02……ÖlçümXYN]

giriş vektörü ve

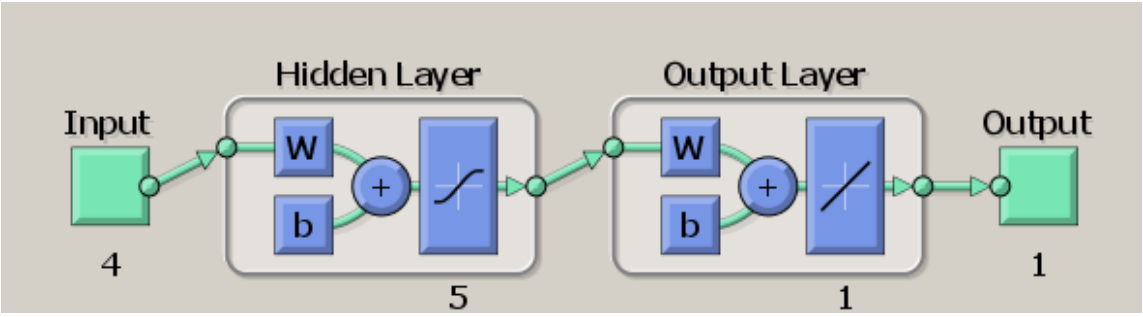
TX : X. bölge için hedef veri vektörü
DeğerN : Hedef kilo değeri

olmak üzere :

TX = [Değer1 Değer2 Değer3……..DeğerN]

vektörleri elde edilir.

Giriş ve hedef verileri için elde edilen vektörler ile arayüzde 'Establish Net' butonu yardımıyla sinir ağı oluşturularak işlem tamamlanmış olur. Modellerin oluşturulması ve ölçümlerin similasyonu için hazırlanan arayüz kaynak kodu EK - A' da sunulmuştur. Olşturulan Lineer model yapısı Şekil 4.12a da görülmektedir.



Şekil 4.12a Model ağ yapısı

Model için kullanılan veriler birkaç deneme ile şu şekilde belirlenmiştir ;

```
net =

    Neural Network object:

    architecture:

        numInputs      : 4
        numLayers      : 3
        biasConnect    : [1; 1; 1]
        inputConnect   : [1; 0; 0]
        layerConnect   : [0 0 0; 1 0 0; 0 1 0]
        outputConnect  : [0 0 1]

        numOutputs     : 1  (read-only)
    numInputDelays     : 0  (read-only)
    numLayerDelays     : 0  (read-only)

    subobject structures:

        inputs         : {1x4 cell} of inputs
        layers         : {3x1 cell} of layers
        outputs        : {1x3 cell} containing 1 output
        biases         : {3x1 cell} containing 3 biases
        inputWeights   : {3x1 cell} containing 1 input weight
        layerWeights   : {3x3 cell} containing 2 layer weights

    functions:
```

```
    adaptFcn      : 'trainlm'
    divideFcn     : (none)
    gradientFcn   : 'gdefaults'
   initFcn        : 'initlay'
    performFcn    : 'mse'
    plotFcns      : {'plotperform','plottrainstate','plotregression'}
   trainFcn       : 'trainlm'

parameters:

    adaptParam    : .passes
    divideParam   : (none)
    gradientParam : (none)
    initParam     : (none)
    performParam  : (none)
    trainParam    : .show, .showWindow, .showCommandLine, .epochs,
                    .time, .goal, .max_fail, .mem_reduc,
                    .min_grad, .mu, .mu_dec, .mu_inc,
                    .mu_max

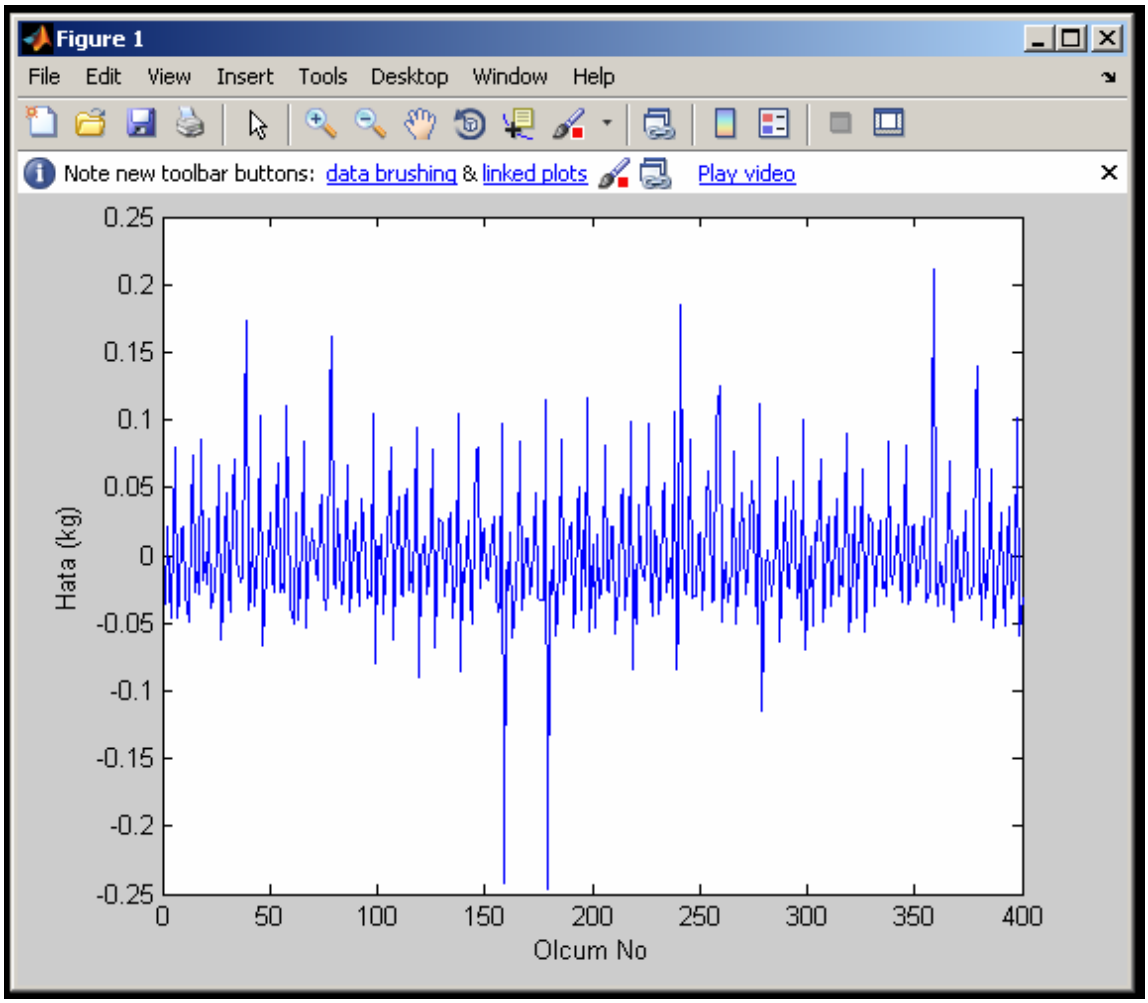
weight and bias values:

      IW          : {4x1 cell} containing 1 input weight matrix
      LW          : {3x3 cell} containing 2 layer weight matrices
       B          : {3x1 cell} containing 3 bias vectors

other:

     name: ''
  userdata: (user information)
```

Uygulamada 10 kg, 20 kg ve 35 kg için alınan ölçüm sonuçları için model oluşturulmuş ve girilen ölçüm değerleri için ağ cevabının similasyonu yapılmıştır. Girilen hedef değerler ve ağ cevabı arasında ki farkın grafiksel olarak gösterimi Şekil 4.12'de görülmektedir.

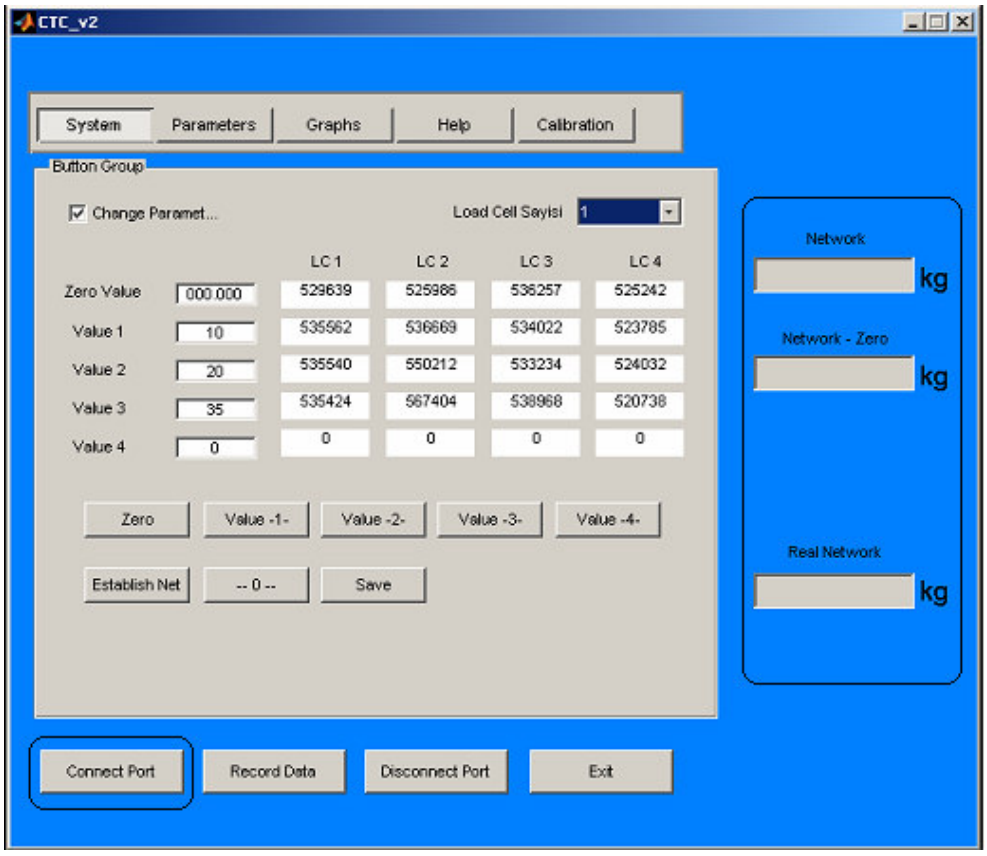


Şekil 4.12b Ağ ölçüm hatası

Grafikten görgleceği üzere, en fazla sapma 250 gram civarında ve belirli bir noktada oluşmaktadır. Oluşan sapmanın sistemin 400 kg ölçüm kapasitesine sahip

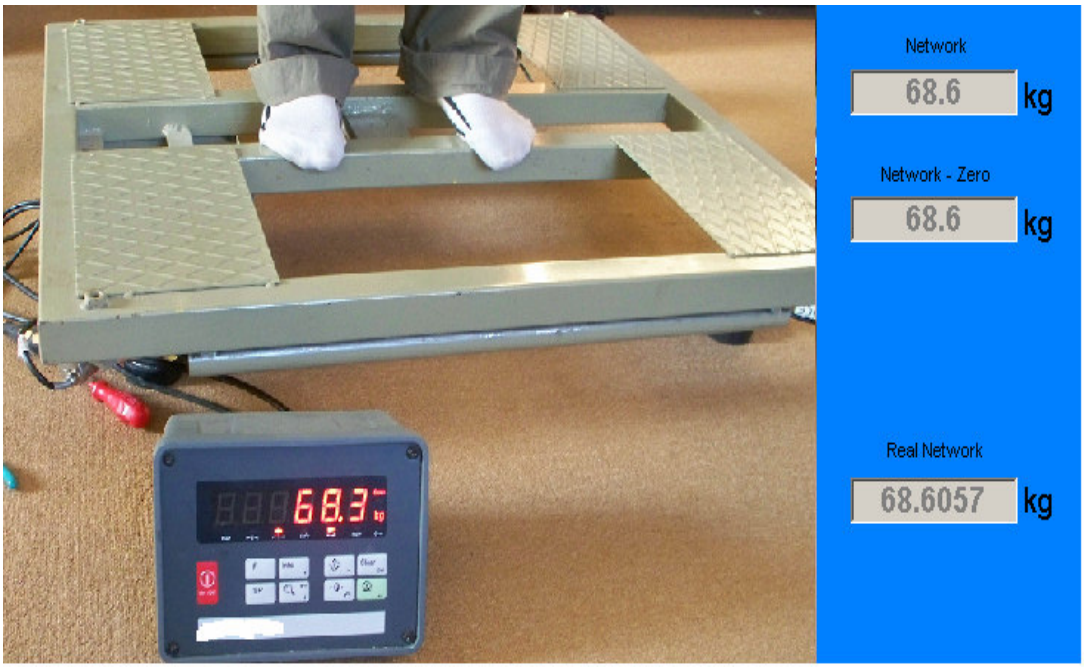
olduğu, sinyalin filtre edilmeden sisteme uygulandığı düşünülecek olursa kabul edilebilir sınırlar içerisinde olduğu görülmektedir. Burada başlıca amaç oluşturulan modelin sistem performansına etkisinin net bir şekilde görülmeye çalışılmasıdır. Kalibrasyonda kullanılan yük değerinin üzerinde bir değerle ölçümler yapmak etkinin daha net bir şekilde görülebilmesi için yardımcı olacaktır.

Gerekli ölçümler yapılarak kalibrasyon tamamlanmış ve ağ oluşturularak sistem tartım için hazır hale gelmiştir. Tartım işlemi "Connect Port " butonu yardımıyla başlatılarak sisteme uygulanacak herhangi bir yük için çevrim sonuçlarının ağa uygulanması ve sonucun ekranda gösterilmesi sağlanmış olur.



Şekil 4.13 Tartım işleminin başlatılması

Denemeler gerçek değeri 68.600 kg ölçülmüş bir kişi ile yapılmıştır. Yükün sabit bir yük seçilmemesinin nedeni, sistem üzerinde değişik yük dağılımlarının sağlanmak istenmesinden ileri gelmiştir. Öncelikle sistem üzerinde her bir köşe için, sonra ki adımda rastlantısal dağılımlar için değerler ölçülmüş, aynı işlem ticari olarak kullanılan bir indikatör ilede yapılarak değerler karşılaştırılmıştır. Aşağıda ki şekillerde, yapılan ölçümler ayrıntılı şekilde görülmektedir.



Şekil 4.14 Merkez nokta deneme ölçümü



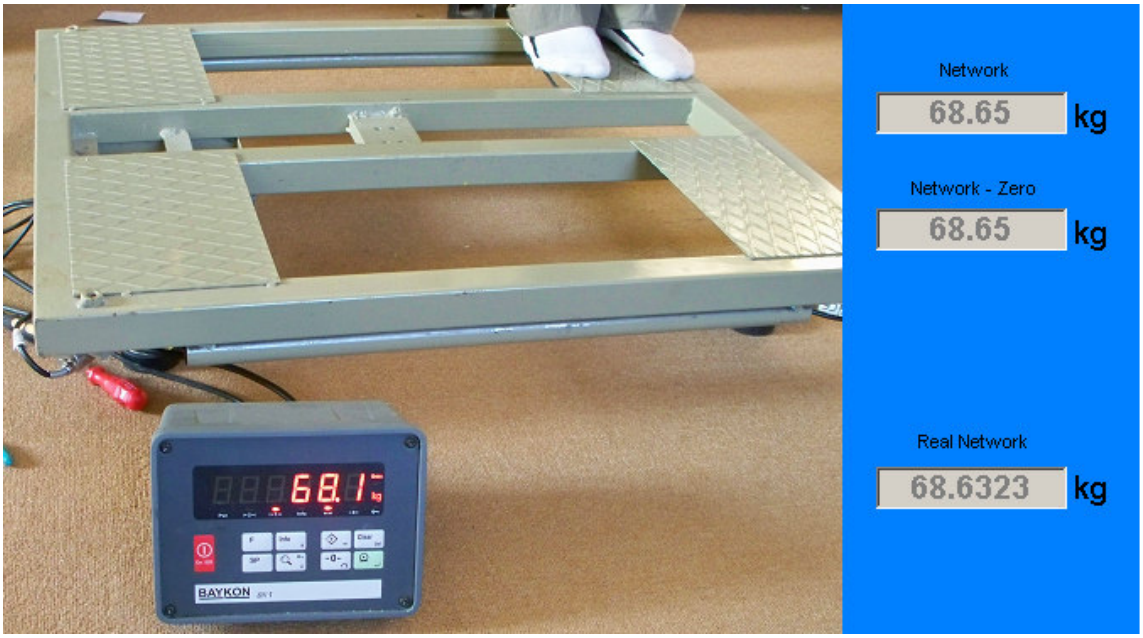
Şekil 4.15 1. Köşe deneme ölçümü



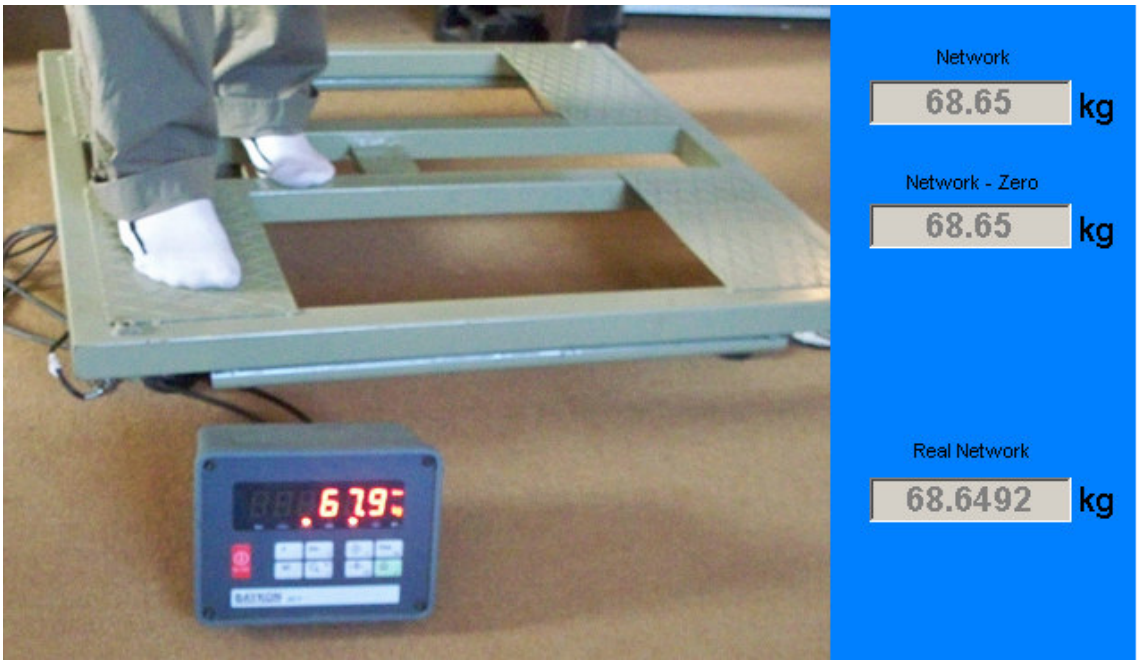
Şekil 4.16 2. Köşe deneme ölçümü



Şekil 4.17 3. Köşe deneme ölçümü



Şekil 4.18 4. Köşe deneme ölçümü



Şekil 4.19 1. rastlantısal ölçüm denemesi



Şekil 4.20 2. rastlantısal ölçüm denemesi



Şekil 4.21 3. rastlantısal ölçüm denemesi

Şekillerden de görüleceği üzere sinir ağı ile yapılan tartım sonucu en fazla sapma 200 gram olmaktadır. Ölçüm sonuçları sabitleştirici bir filtre uygulanmadığı için 150 gram civarında salınım yapmakta, oluşan salınım kabul edilebilir sınırlar içerisinde kalmaktadır. Yük hücreleri bağımsız ölçüldükleri için sistem cevabında biraz da olsa gecikme olmaktadır, fakat hareketli tartım sistemleri dışında bu gecikme performansı olumsuz etkileyecek seviyede olmamaktadır.

Ticari indikatör ile yapılan ölçüm sonucunun ise 2.5 kg'a kadar farklılık gösterdiği görülmekte, ölçüm yapılan köşe değerleri biribirinden farklı sonuç vermektedir. Rastlantısal yapılan ölçüm sonucu, yük merkeze yaklaştıkça, tartım sonucu gerçek değere yakınsamakta, tam merkezde gerçek değer ile tartım sonucu arasında ki değer minimum olmaktadır. Burdan anlaşılacağı üzere standart tip sistemler için yük hücreleri birbirlerini olumsuz etkilemekte, yükün sistem üzerindeki dağılımı önemli bir parametre haline gelmektedir. Oluşan bu farkı giderebilmek için "junction box" üzerinde trimpotlar

vasıtasıyla yük hücresi farklılıkları giderilmeye çalışır. Yalnız bu durum bile, yükün tek bir noktadan sisteme uygulanması durumunda, oluşan farkı gideremektedir.

5. SONUÇ VE ÖNERİLER

Bu çalışmada birbirinden bağımsız yük hücreleri ölçümlerine dayalı bir kantar sistemi tasarlanarak, sonuç yapay sinir ağlarıyla belirlenmeye çalışılmıştır. Ölçüm için hazırlanan prototip üzerinde, 4 adet yük hücresi kullanılmıştır. Yük hücrelerinden birisi farklı marka da seçilmiş, montajı yapılan bir yük hücresi ise, diğer yük hücrelerinden farklı olarak platformun dış tarafına gelecek şekilde yerleştirilmiştir. Böylece mekanik anlamda hatalı bir düzenek elde edilmiştir.

Hazırlanan bu düzenek tasarlanan indikatöre bağlanarak ölçümler yapılmış, ölçüm sonuçları bilgisayar ortamına aktarılarak bir yapay sinir ağı modeli oluşturulmuştur. Sistem yapısı lineer olduğu için 4 girişli 1 çıkışlı, 2 ara katmana sahip lineer bir model oluşturulmuştur. Ara katman nöron sayıları 5 ve 1 seçilerek Levenberg-Marquardt geri yayılım algoritmasıyla ağın eğitimi sağlanmıştır.

Modelden elde edilen sonuçlar, ölçülen ağırlık değerlerine kabul edilebilir oranlarda yaklaşmaktadır. Bu da tasarımın, başarılı sayılabileceğini göstermektedir. Ölçülen yük, platform üzerinde farklı noktalarda birbirine yakın değerlerde ölçülmekte, yükün homojen dağılım koşulu ortadan kalkmaktadır. Sistem mekaniği yada zemin hataları dinamik olmadığı sürece, kalibrasyon esnasında, ağ bu hataları olduğu gibi kabul ederek sonuç üretmekte, kurulan sistemin yapısı değişmediği sürece ölçümler yapılabilmektedir. Böylece, üretim safhasında hatalı üretilmiş mamüller, tekrar üretilmek zorunda kalmayacak, kurulum safhasının zorluk ve hassasiyeti minimuma çekilerek sistem maliyetinde avantaj sağlanabilecektir. Ayrıca, ölçüm, yük hücreleri paramtrelerinden bağımsız hale gelmektedir. Kantar imalatında kullanılacak yük hücreleri, farklı marka ve özellikte olabileceklerinden, bakım ve servis maliyetleri düşecektir.

Tasarımın sağladığı avantajlara rağmen, yük hücreleri farklı sistemler baz alınarak ölçülüp, sonuç bu ölçümlerin toplamından elde edildiği için zamansal bir gecikme meydana gelmektedir. Oluşan gecikme, bir çok uygulama için dikkate alınmayacak büyüklükte olmasına rağmen, akışkan tartımı yada torbalama sistemleri gibi hareketli tartımların yapıldığı sistemler için kullanımı mümkün kılmamaktadır. Ayrıca, sistem fonksiyonunun belirlenme başarımı, ağa tanıtılan farklı örnek sayısı ile orantılı olacaktır. Yalnız, kalibrasyon işleminin pratikte de uygulanabilir olması için ölçümler belirli bir sayıda sınırlandırılmalıdır. Bu çelişki, kabul edilebilir sonuçlar elde etmek için optimum bir çözüm bulunmasını zorunlu kılmaktadır. Uygulamada kalibrasyon, 3 farklı yükle yapılmış, gerçek değere kabul edilebilir oranlarda yaklaşılmıştır. Ölçüm sonucunun daha hassas bir şekilde belirlenebilmesi için kalibrasyon aşamasında, farklı değerlerde bir çok ölçüm yapılması gerektiği görülmektedir.

Bu çalışmada, oluşturulan sinir ağı modelinin devre üzerinde tasarımı amaçlanmamıştır. Yalnız, donanım ileriye yönelik oluşabilecek ihtiyaçlar gözönünde bulundurularak gerekli teknolojik alt yapı düşünülerek tasarlanmıştır. Böylece, ileride tasarım bir adım ileriye taşınarak, ölçüm sonuçlarına dayalı sinir ağı modeli bilgisayar ortamında oluşturulup eğitilir, eğitim sonucu elde edilen ağırlık katsayıları yardımıyla model, devre üzerinde kolaylıkla gerçeklenebilir. Bu sayede, bilgisayara ihtiyaç kalmadan düşük maliyetli, yapay sinir ağına dayalı bir indikatör elde edilmiş olur.

6. KAYNAKLAR

Bakioglu M., Cisimlerin Mukavemeti, Beta Yayinevi, 2001.

Beer F. and Johnston E.R., Mechanics of Materials, Mc-Graw Hill, New York 1996.

Broomhead D. S. ve Lowe D., "Multivariable Functional Interpolation and Adaptive Networks," Complex systems, 2, s. 321-355, 1988.

GÜLTEKİN S.S., "Çeşitli Mikroşerit Antenlerin Karakteristik Parametrelerinin Farklı Tipteki Algoritmalarla Eğitilen Yapay Sinir Ağları İle Hesaplanması" Doktora Tezi, Selçuk Üniversitesi Fen Bilimleri Enstitüsü, 2002, Konya.

GÜLTEKİN S.S., GÜNEY K. ve SAĞIROĞLU Ş. "Neural Networks For The Calculation of Bandwidth of Rectangular Microstrip Antennas" Applied Computational Electromagnetics Society Journal (ACES) p.18-2., 2003.

Hagan, M.T., H.B. Demuth, and M.H. Beale, Neural Network Design, Boston, MA: PWS Publishing, 1996.

HAYKIN S., Neural Networks, A Comprehensive Foundation, Macmillan College Publishing Comp. Inc., 1994.

Hibbeler R.C., "Mechanics of Materials", 1998

Hopfield J.J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," In: Proc. National Academy of Sciences, USA, April, 79, s. 2554-2558, 1982.

Inan M., Cisimlerin Mukavemeti, ITÜ Vakfi, Yayin No.25, Istanbul,1988.

Karna K.N. ve Breen D.M., "An Artificial Neural Networks Tutorial: Basics," Neural Networks, 1, no:1, s. 4 -23, 1989.

Kohonen T., "Statistical Patern Recognition with Neural Networks:" Benchmark Studies, Porceedings of The Second Annual IEEE International Conference on Neural Networks, 1, 1988.

Kohonen T., Self-Organisation and Associative Memory (3rd ed), SpRinger-Verlag, Berlin, 1989.

Lowe D., "Adaptive Radial Basis Function Nonlinearities and The Problem of Generalization," In Proceedings of The First IEEE International Conference on Artificial Neural Networks, London, 6, s.171-175, 1989.

Mcculloch W., Pitts W., “A Logical Calculus Of The Ideas Immanent In Nervous Activity,” Bulletin Of Mathematical Biophysics., 5, s. 115-133, 1943.

Özbek T., Mukavemet, Birsen Yayinevi, Istanbul 1978.

ÖZTEMEL E., 2006 Yapay Sinir Ağları, İkinci Basım, Papatya Yayıncılık.

RUMELHART D.E. ve McClelland J.L., “Parallel Distributed Processing: Explorations in the Microstructure of Cognition,” vol.1, MIT Press, Cambridge, Massachusetts, 1986.

SAĞIROĞLU Ş., BEŞDOK E., ERLER M., “Mühendislikte Yapay Zeka Uygulamaları – I yapay Sinir Ağları” Ufuk Yayıncılık Ağustos 2003.

SARAÇ T., 2004, Yapay Sinir Ağları Seminer Projesi Gazi Üniversitesi Endüstri Mühendisliği Bölümü Ana Bilim Dalı Haziran 2004.

Simpson P.S., Artificial Neural Networks, A Foundation, Paradigm, Aplications and Implamantation, Pergamon Press., 1990.

Zurada J.M., Introduction to Artificial Neural Networks, West Publishing Company, 1992.

EK – A Arayüz Kaynak Kodu

```
function varargout = CTC_v2(varargin)
% CTC_V2 M-file for CTC_v2.fig
%      CTC_V2, by itself, creates a new CTC_V2 or raises the existing
%      singleton*.
%
%      H = CTC_V2 returns the handle to a new CTC_V2 or the handle to
%      the existing singleton*.
%
%      CTC_V2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CTC_V2.M with the given input
arguments.
%
%      CTC_V2('Property','Value',...) creates a new CTC_V2 or raises
the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before CTC_v2_OpeningFunction gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to CTC_v2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help CTC_v2

% Last Modified by GUIDE v2.5 06-May-2009 17:45:36

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @CTC_v2_OpeningFcn, ...
                   'gui_OutputFcn',  @CTC_v2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%guidata(hObject, handles)
enc = getappdata(handles.figure1,'enc_data');

enc.Sifir = enc.Y

setappdata(handles.figure1,'enc_data', enc);
disp(enc.Sifir)
%guidata(handles.figure1,handles)

% --- Executes just before CTC_v2 is made visible.
function CTC_v2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to CTC_v2 (see VARARGIN)



% Choose default command line output for CTC_v2
handles.output = hObject;

set(handles.edit2,'String', 'Not connected to device...');

fid = fopen('ENC_matfile.txt', 'r');

handles.Store_data = fread(fid, 2005, 'double');

enc.record = 0 ;

enc.scale1 = 0;
enc.scale2 = 0;
enc.scale3 = 0;
enc.scale4 = 0;
enc.scale5 = 0;

enc.old_time = 0 ;

enc.Button_value = 0 ;


for k = 1: 1: 400
    enc.deger0(k) = handles.Store_data(k*5 + 1);
    enc.deger1(k) = handles.Store_data(k*5 + 2);
    enc.deger2(k) = handles.Store_data(k*5 + 3);
```

```
    enc.deger3(k) = handles.Store_data(k*5 + 4);
    enc.deger4(k) = handles.Store_data(k*5 + 5);
end


enc.Y = 0;
enc.Y_2 = 0;
enc.Sifir = 0 ;

handles.s_port = serial('COM9');
set(handles.s_port, 'Terminator', 'W');
set(handles.s_port, 'BaudRate' , 38400);

set(handles.s_port, 'ErrorFcn' , @pushbutton1_Callback);

set(handles.s_port, 'Timeout', 6);
set(handles.s_port, 'ReadAsyncMode', 'continuous');

set(handles.s_port, 'BytesAvailableFcnMode', 'terminator');


enc.Sifir = 0 ;


enc.V1_gr   = handles.Store_data(1);
enc.V2_gr   = handles.Store_data(2);
enc.V3_gr   = handles.Store_data(3);
enc.V4_gr   = handles.Store_data(4);

enc.Zero_1 = handles.Store_data(6);
enc.V1_1   = handles.Store_data(7);
enc.V2_1   = handles.Store_data(8);
enc.V3_1   = handles.Store_data(9);
enc.V4_1   = handles.Store_data(10);

enc.Zero_2 = handles.Store_data(11);
enc.V1_2   = handles.Store_data(12);
enc.V2_2   = handles.Store_data(13);
enc.V3_2   = handles.Store_data(14);
enc.V4_2   = handles.Store_data(15);


enc.Zero_3 = handles.Store_data(16);
enc.V1_3   = handles.Store_data(17);
enc.V2_3   = handles.Store_data(18);
enc.V3_3   = handles.Store_data(19);
enc.V4_3   = handles.Store_data(20);

enc.Zero_4   = handles.Store_data(21);
enc.V1_4   = handles.Store_data(22);
enc.V2_4   = handles.Store_data(23);
enc.V3_4   = handles.Store_data(24);
enc.V4_4   = handles.Store_data(25);
```

```
set(handles.text48, 'String', enc.Zero_1);
set(handles.text56, 'String', enc.V1_1);
set(handles.text60, 'String', enc.V2_1);
set(handles.text64, 'String', enc.V3_1);
set(handles.text68, 'String', enc.V4_1);

set(handles.text53, 'String', enc.Zero_2);
set(handles.text57, 'String', enc.V1_2);
set(handles.text61, 'String', enc.V2_2);
set(handles.text65, 'String', enc.V3_2);
set(handles.text69, 'String', enc.V4_2);

set(handles.text54, 'String', enc.Zero_3);
set(handles.text58, 'String', enc.V1_3);
set(handles.text62, 'String', enc.V2_3);
set(handles.text66, 'String', enc.V3_3);
set(handles.text70, 'String', enc.V4_3);

set(handles.text55, 'String', enc.Zero_4);
set(handles.text59, 'String', enc.V1_4);
set(handles.text63, 'String', enc.V2_4);
set(handles.text67, 'String', enc.V3_4);
set(handles.text71, 'String', enc.V4_4);

set(handles.edit15, 'String', enc.V1_gr);
set(handles.edit18, 'String', enc.V2_gr);
set(handles.edit19, 'String', enc.V3_gr);
set(handles.edit20, 'String', enc.V4_gr);


fclose(fid);

enc.net_AD  = 0;
enc.net_AD_2 = 0;
setappdata(handles.figure1,'enc_data', enc);
fopen(handles.s_port);
handles.T=timer('TimerFcn',{@pushbutton19_Callback,handles},...
'ExecutionMode','fixedDelay','Period',1, 'StartDelay', 2);
set(handles.T, 'ErrorFcn' , @pushbutton1_Callback);

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes CTC_v2 wait for user response (see UIRESUME)
uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = CTC_v2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
%varargout{1} = handles.output;


% --- Connect Port
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


gst1 = instrfind('BaudRate', 38400);
if(strcmp(get(gst1, 'Status'), 'closed'))
    fopen(gst1);
end

gst_timer = timerfind('ExecutionMode','fixedDelay');

if(strcmp(gst_timer, 'off'))
    start(gst_timer);
end
    start(gst_timer);



% --- Record data
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- disconnect Port
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

gst_timer = timerfind('ExecutionMode','fixedDelay');

if(strcmp(gst_timer, 'on'))
    stop(gst_timer);
end
    stop(gst_timer);


if(strcmp(handles.s_port.status, 'open'))
    fclose(handles.s_port);
end
```

```
% --- Send command   / LCD ye rutinleri gonderecek
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global com_flag



p1 = findobj('Tag', 'popupmenu2')
val = get(p1, 'Value')


switch val
    case 1
        Command = char (248);
        com_flag = 1
    case 2
        Command = 80;
    case 3
        Command = 247;
end


y = strcat('Command :   ', num2str(Command));
y = strvcat(y,'           ');
k = strvcat(y, get(findobj('Tag', 'edit2'), 'String'))
t = strvcat(' ', k)
set(findobj('Tag', 'edit2'), 'String', t)

gst1 = instrfind('Baudrate', 9600);

fprintf(gst1, Command);

while(get(gst1, 'BytesAvailable') > 0)
    uiwait();
end



% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu2 contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from
%        popupmenu2
```

```
% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Exit
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if(strcmp(get(handles.s_port, 'Status'), 'open'))
    fclose(handles.s_port);
end
delete(handles.s_port);

if(strcmp(handles.T.Running, 'on'))
    stop(handles.T);
end
delete(handles.T);


% Hint: delete(hObject) closes the figure
%delete(figure1);



% --- Executes during object creation, after setting all properties.
function figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called


function Update_text(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


gst1 = instrfind('Baudrate', 38400);
```

```
gst_data = 0;
gst_data2 = 0;
gst_data3 = 0;
gst_data4 = 0;

L_index = 1;
L_loadcell = 1;




if(strcmp(get(gst1, 'Status'), 'closed'))

    fopen(gst1);
    disp('open')
end


get(handles.popupmenu14, 'Value');

enc = getappdata(handles.figure1,'enc_data');

while L_index < 21
    fwrite(gst1, 71);
    gst_data = fscanf(gst1);


L_loadcell = (get(handles.popupmenu14, 'Value')*4) -3;

L_loadcell = L_loadcell + (L_index - 1) * 20  ;

value_gst (L_loadcell) = (gst_data (1)-48)*100000000 + (gst_data (2)-
48)*10000000 + (gst_data(3)-48)*1000000 + ...
    (gst_data (4)-48)*100000 + (gst_data (5)-48)*10000 + (gst_data (6)-
48)*1000 ...
    + (gst_data (7)-48)*100 + (gst_data (8)-48)*10 + (gst_data (9)-48)
;

switch enc.Button_value
    case 0
        set(handles.text48, 'String', value_gst(L_loadcell));
        enc.deger0(L_loadcell) = value_gst (L_loadcell);
    case 1
        set(handles.text56, 'String', value_gst(L_loadcell));
        enc.deger1(L_loadcell) = value_gst (L_loadcell);
    case 2
        set(handles.text60, 'String', value_gst(L_loadcell));
        enc.deger2(L_loadcell) = value_gst (L_loadcell);
    case 3
        set(handles.text64, 'String', value_gst(L_loadcell));
        enc.deger3(L_loadcell) = value_gst (L_loadcell);
    case 4
        set(handles.text68, 'String', value_gst(L_loadcell));
        enc.deger4(L_loadcell) = value_gst (L_loadcell);
```

```
end

value_gst (L_loadcell + 1) = (gst_data (11)-48)*100000000 + (gst_data
(12)-48)*10000000 + (gst_data(13)-48)*1000000 + ...
    (gst_data (14)-48)*100000 + (gst_data (15)-48)*10000 + (gst_data
(16)-48)*1000 ...
    + (gst_data (17)-48)*100 + (gst_data (18)-48)*10 + (gst_data (19)-
48);


switch enc.Button_value
    case 0
        set(handles.text53, 'String', value_gst(L_loadcell + 1));
        enc.deger0(L_loadcell + 1) = value_gst (L_loadcell + 1);
    case 1
        set(handles.text57, 'String', value_gst(L_loadcell + 1));
        enc.deger1(L_loadcell + 1) = value_gst (L_loadcell + 1);
    case 2
        set(handles.text61, 'String', value_gst(L_loadcell + 1));
        enc.deger2(L_loadcell + 1) = value_gst (L_loadcell + 1);
    case 3
        set(handles.text65, 'String', value_gst(L_loadcell + 1));
        enc.deger3(L_loadcell + 1) = value_gst (L_loadcell + 1);
    case 4
        set(handles.text69, 'String', value_gst(L_loadcell + 1));
        enc.deger4(L_loadcell + 1) = value_gst (L_loadcell + 1);
end

value_gst (L_loadcell + 2) = (gst_data (21)-48)*100000000 + (gst_data
(22)-48)*10000000 + (gst_data(23)-48)*1000000 + ...
    (gst_data (24)-48)*100000 + (gst_data (25)-48)*10000 + (gst_data
(26)-48)*1000 ...
    + (gst_data (27)-48)*100 + (gst_data (28)-48)*10 + (gst_data (29)-
48);

switch enc.Button_value
    case 0
        set(handles.text54, 'String', value_gst(L_loadcell + 2));
        enc.deger0(L_loadcell + 2) = value_gst (L_loadcell + 2);
    case 1
        set(handles.text58, 'String', value_gst(L_loadcell + 2));
        enc.deger1(L_loadcell + 2) = value_gst (L_loadcell + 2);
    case 2
        set(handles.text62, 'String', value_gst(L_loadcell + 2));
        enc.deger2(L_loadcell + 2) = value_gst (L_loadcell + 2);
    case 3
        set(handles.text66, 'String', value_gst(L_loadcell + 2));
        enc.deger3(L_loadcell + 2) = value_gst (L_loadcell + 2);
    case 4
        set(handles.text70, 'String', value_gst(L_loadcell + 2));
        enc.deger4(L_loadcell + 2) = value_gst (L_loadcell + 2);

end
```

```
value_gst (L_loadcell + 3) = (gst_data (31)-48)*100000000 + (gst_data
(32)-48)*10000000 + (gst_data(33)-48)*1000000 + ...
    (gst_data (34)-48)*100000 + (gst_data (35)-48)*10000 + (gst_data
(36)-48)*1000 ...
    + (gst_data (37)-48)*100 + (gst_data (38)-48)*10 + (gst_data (39)-
48);

switch enc.Button_value
    case 0
        set(handles.text55, 'String', value_gst(L_loadcell + 3));
        enc.deger0(L_loadcell + 3) = value_gst (L_loadcell + 3);
    case 1
        set(handles.text59, 'String', value_gst(L_loadcell + 3));
        enc.deger1(L_loadcell + 3) = value_gst (L_loadcell + 3);
    case 2
        set(handles.text63, 'String', value_gst(L_loadcell + 3));
        enc.deger2(L_loadcell + 3) = value_gst (L_loadcell + 3);
    case 3
        set(handles.text67, 'String', value_gst(L_loadcell + 3));
        enc.deger3(L_loadcell + 3) = value_gst (L_loadcell + 3);
    case 4
        set(handles.text71, 'String', value_gst(L_loadcell + 3));
        enc.deger4(L_loadcell + 3) = value_gst (L_loadcell + 3);
end

L_loadcell = L_loadcell + 1 ;
L_loadcell = L_loadcell + 1 ;
L_loadcell = L_loadcell + 1 ;
L_loadcell = L_loadcell + 1 ;

L_index = L_index + 1 ;
pause (2)
end

disp('ok')

setappdata(handles.figure1,'enc_data', enc);


% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


fclose(handles.s_port);
delete(handles.s_port);

gst_timer = timerfind('ExecutionMode','fixedDelay');

if(strcmp(gst_timer, 'on'))
    stop(gst_timer);
    delete(gst_timer);
end
```

```
    stop(gst_timer);
    delete(gst_timer);



% Hint: delete(hObject) closes the figure
delete(hObject);




% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end





% --- Executes when selected object is changed in uipanel6.
function uipanel6_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uipanel6
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

switch get(hObject,'Tag')   % Get Tag of selected object
    case 'togglebutton1'

        set(handles.uipanel7, 'visible', 'off');
        set(handles.uipanel8, 'visible', 'off');
        set(handles.uipanel9, 'visible', 'off');
        set(handles.uipanel11, 'visible', 'off');
        set(handles.uipanel1, 'visible', 'on');
        disp('button1')
    case 'togglebutton2'

        set(handles.uipanel1, 'visible', 'off');
        set(handles.uipanel8, 'visible', 'off');
        set(handles.uipanel9, 'visible', 'off');
        set(handles.uipanel11, 'visible', 'off');
        set(handles.uipanel7, 'visible', 'on');
        disp('button2')
    case 'togglebutton3'
```

```
        set(handles.uipanel1, 'visible', 'off');
        set(handles.uipanel7, 'visible', 'off');
        set(handles.uipanel9, 'visible', 'off');
        set(handles.uipanel11, 'visible', 'off');
        set(handles.uipanel8, 'visible', 'on');
        disp('button3')
    case 'togglebutton4'

        set(handles.uipanel1, 'visible', 'off');
        set(handles.uipanel7, 'visible', 'off');
        set(handles.uipanel8, 'visible', 'off');
        set(handles.uipanel11, 'visible', 'off');
        set(handles.uipanel9, 'visible', 'on');
        disp('button4')

    case 'togglebutton5'

        set(handles.uipanel1, 'visible', 'off');
        set(handles.uipanel7, 'visible', 'off');
        set(handles.uipanel8, 'visible', 'off');
        set(handles.uipanel9, 'visible', 'off');
        set(handles.uipanel11, 'visible', 'on');
        disp('button4')

end


guidata(hObject,handles)




% --- Executes on button press in checkbox8.
function checkbox8_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox8
if(get(handles.checkbox8,'Value') == 0)
    set(handles.radiobutton31, 'Value',0);
else
    set(handles.radiobutton31, 'Value',1);
end

% --- Executes on button press in checkbox9.
function checkbox9_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox9
if(get(handles.checkbox9,'Value') == 0)
```

```
    set(handles.radiobutton23, 'Value',0);
else
    set(handles.radiobutton23, 'Value',1);
end

% --- Executes on button press in checkbox10.
function checkbox10_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox10
if(get(handles.checkbox10,'Value') == 0)
    set(handles.radiobutton22, 'Value',0);
else
    set(handles.radiobutton22, 'Value',1);
end

% --- Executes on button press in pushbutton13.
function pushbutton13_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
gst8 = instrfind('Baudrate', 9600)

if(strcmp(get(gst8, 'Status'), 'closed'))
    fopen(gst8);
    data_veri = 35;
    fwrite(gst8,data_veri, 'char')
    fclose(gst8);
else
    data_veri = 36;
    fwrite(gst8,data_veri, 'char')
end



% --- Executes when selected object is changed in uipanel18.
function uipanel18_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uipanel18
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if(get(handles.radiobutton60,'Value') == 1)
    set(handles.radiobutton32, 'Value',0);
end

if(get(handles.radiobutton61,'Value') == 1)
    set(handles.radiobutton32, 'Value',1);
end



% --- Executes when selected object is changed in uipanel17.
function uipanel17_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uipanel17
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

if(get(handles.radiobutton52,'Value') == 1)
    set(handles.radiobutton28, 'Value',0);
end

if(get(handles.radiobutton53,'Value') == 1)
    set(handles.radiobutton28, 'Value',1);
end


% --- Executes when selected object is changed in uipanel19.
function uipanel19_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uipanel19
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if(get(handles.radiobutton62,'Value') == 1)
    set(handles.radiobutton27, 'Value',0);
end

if(get(handles.radiobutton63,'Value') == 1)
    set(handles.radiobutton27, 'Value',1);
end


% --- Executes when selected object is changed in uipanel20.
function uipanel20_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uipanel20
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if(get(handles.radiobutton64,'Value') == 1)
    set(handles.radiobutton25, 'Value',0);
    set(handles.radiobutton24, 'Value',0);
end

if(get(handles.radiobutton65,'Value') == 1)
    set(handles.radiobutton25, 'Value',0);
    set(handles.radiobutton24, 'Value',1);
end

if(get(handles.radiobutton66,'Value') == 1)
    set(handles.radiobutton25, 'Value',1);
    set(handles.radiobutton24, 'Value',0);
end

if(get(handles.radiobutton67,'Value') == 1)
    set(handles.radiobutton25, 'Value',1);
    set(handles.radiobutton24, 'Value',1);
end


% --- Executes when selected object is changed in uipanel21.
```

```
function uipanel21_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uipanel21
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if(get(handles.radiobutton72,'Value') == 1)
    set(handles.radiobutton21, 'Value',0);
    set(handles.radiobutton20, 'Value',0);
end


if(get(handles.radiobutton73,'Value') == 1)
    set(handles.radiobutton21, 'Value',0);
    set(handles.radiobutton20, 'Value',1);
end

if(get(handles.radiobutton74,'Value') == 1)
    set(handles.radiobutton21, 'Value',1);
    set(handles.radiobutton20, 'Value',0);
end


% --- Executes on selection change in popupmenu14.
function popupmenu14_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu14 contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu14


% --- Executes during object creation, after setting all properties.
function popupmenu14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit15 as text
%        str2double(get(hObject,'String')) returns contents of edit15
as a double
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit17_Callback(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit17 as text
%        str2double(get(hObject,'String')) returns contents of edit17
as a double


% --- Executes during object creation, after setting all properties.
function edit17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit18_Callback(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit18 as text
```

```
%        str2double(get(hObject,'String')) returns contents of edit18
as a double


% --- Executes during object creation, after setting all properties.
function edit18_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit19_Callback(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit19 as text
%        str2double(get(hObject,'String')) returns contents of edit19
as a double


% --- Executes during object creation, after setting all properties.
function edit19_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit20_Callback(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit20 as text
%        str2double(get(hObject,'String')) returns contents of edit20
as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit20_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in checkbox11.
function checkbox11_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox11

state_check_box = get(hObject, 'Value')

if(state_check_box == 1)
    set(handles.edit15, 'Enable', 'on');
    set(handles.edit18, 'Enable', 'on');
    set(handles.edit19, 'Enable', 'on');
    set(handles.edit20, 'Enable', 'on');

    set(handles.text48, 'Enable', 'on');

    set(handles.text53, 'Enable', 'on');
    set(handles.text54, 'Enable', 'on');
    set(handles.text55, 'Enable', 'on');
    set(handles.text56, 'Enable', 'on');
    set(handles.text57, 'Enable', 'on');
    set(handles.text58, 'Enable', 'on');
    set(handles.text59, 'Enable', 'on');
    set(handles.text60, 'Enable', 'on');
    set(handles.text61, 'Enable', 'on');
    set(handles.text62, 'Enable', 'on');
    set(handles.text63, 'Enable', 'on');
    set(handles.text64, 'Enable', 'on');
    set(handles.text65, 'Enable', 'on');
    set(handles.text66, 'Enable', 'on');
    set(handles.text67, 'Enable', 'on');
    set(handles.text68, 'Enable', 'on');
    set(handles.text69, 'Enable', 'on');
    set(handles.text70, 'Enable', 'on');
    set(handles.text71, 'Enable', 'on');
```

```
    set(handles.popupmenu14, 'Enable', 'on');

    set(handles.pushbutton18, 'Enable', 'on');
    set(handles.pushbutton14, 'Enable', 'on');
    set(handles.pushbutton15, 'Enable', 'on');
    set(handles.pushbutton16, 'Enable', 'on');
    set(handles.pushbutton17, 'Enable', 'on');
    set(handles.pushbutton19, 'Enable', 'on');
    set(handles.pushbutton10, 'Enable', 'on');
    set(handles.pushbutton22, 'Enable', 'on');
else
    set(handles.edit15, 'Enable', 'off');
    set(handles.edit18, 'Enable', 'off');
    set(handles.edit19, 'Enable', 'off');
    set(handles.edit20, 'Enable', 'off');

    set(handles.text48, 'Enable', 'off');

    set(handles.text53, 'Enable', 'off');
    set(handles.text54, 'Enable', 'off');
    set(handles.text55, 'Enable', 'off');
    set(handles.text56, 'Enable', 'off');
    set(handles.text57, 'Enable', 'off');
    set(handles.text58, 'Enable', 'off');
    set(handles.text59, 'Enable', 'off');
    set(handles.text60, 'Enable', 'off');
    set(handles.text61, 'Enable', 'off');
    set(handles.text62, 'Enable', 'off');
    set(handles.text63, 'Enable', 'off');
    set(handles.text64, 'Enable', 'off');
    set(handles.text65, 'Enable', 'off');
    set(handles.text66, 'Enable', 'off');
    set(handles.text67, 'Enable', 'off');
    set(handles.text68, 'Enable', 'off');
    set(handles.text69, 'Enable', 'off');
    set(handles.text70, 'Enable', 'off');
    set(handles.text71, 'Enable', 'off');

    set(handles.popupmenu14, 'Enable', 'off');

    set(handles.pushbutton18, 'Enable', 'off');
    set(handles.pushbutton14, 'Enable', 'off');
    set(handles.pushbutton15, 'Enable', 'off');
    set(handles.pushbutton16, 'Enable', 'off');
    set(handles.pushbutton17, 'Enable', 'off');
    set(handles.pushbutton19, 'Enable', 'off');
    set(handles.pushbutton10, 'Enable', 'off');
    set(handles.pushbutton22, 'Enable', 'off');

end

% --- Executes on button press in pushbutton14.
function pushbutton14_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

enc = getappdata(handles.figure1,'enc_data');
enc.Button_value = 1 ;
setappdata(handles.figure1,'enc_data', enc);

Update_text(hObject, eventdata, handles);

enc = getappdata(handles.figure1,'enc_data');
enc.deger1


% --- Executes on button press in pushbutton15.
function pushbutton15_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


enc = getappdata(handles.figure1,'enc_data');
enc.Button_value = 2 ;
setappdata(handles.figure1,'enc_data', enc);

Update_text(hObject, eventdata, handles);

enc = getappdata(handles.figure1,'enc_data');
enc.deger2


% --- Executes on button press in pushbutton16.
function pushbutton16_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


enc = getappdata(handles.figure1,'enc_data');
enc.Button_value = 3 ;
setappdata(handles.figure1,'enc_data', enc);

Update_text(hObject, eventdata, handles);

enc = getappdata(handles.figure1,'enc_data');
enc.deger3


% --- Executes on button press in pushbutton17.
function pushbutton17_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
enc = getappdata(handles.figure1,'enc_data');
enc.Button_value = 4 ;
setappdata(handles.figure1,'enc_data', enc);

Update_text(hObject, eventdata, handles);
enc = getappdata(handles.figure1,'enc_data');
enc.deger4


% --- Executes on button press in pushbutton18.
function pushbutton18_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


enc = getappdata(handles.figure1,'enc_data');
enc.Button_value = 0 ;
setappdata(handles.figure1,'enc_data', enc);

Update_text(hObject, eventdata, handles);
enc = getappdata(handles.figure1,'enc_data');
enc.deger0


% --- Executes on button press in pushbutton19.
function pushbutton19_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%if get(handles.popupmenu14, 'Value')

enc = getappdata(handles.figure1,'enc_data');

enc.Zero_1 = str2num(get(handles.text48, 'String'));
enc.V1_1   = str2num(get(handles.text56, 'String'));
enc.V2_1   = str2num(get(handles.text60, 'String'));
enc.V3_1   = str2num(get(handles.text64, 'String'));
enc.V4_1   = str2num(get(handles.text68, 'String'));

enc.V1_gr = str2num(get(handles.edit15, 'String'));
enc.V2_gr = str2num(get(handles.edit18, 'String'));
enc.V3_gr = str2num(get(handles.edit19, 'String'));
enc.V4_gr = str2num(get(handles.edit20, 'String'));

gst1 = instrfind('BaudRate', 38400);

gst_data  = 0;
gst_data2 = 0;
gst_data3 = 0;
gst_data4 = 0;

L_index = 0;
```

```
L_loadcell = 1;



if(strcmp(get(gst1, 'Status'), 'closed'))
    fopen(gst1);
    disp('open')
end

get(handles.popupmenu14, 'Value');
fwrite(gst1, 71 );
gst_data = fscanf(gst1);


value_gst (L_loadcell) = (gst_data (1)-48)*100000000 + (gst_data (2)-
48)*10000000 + (gst_data(3)-48)*1000000 + ...
    (gst_data (4)-48)*100000 + (gst_data (5)-48)*10000 + (gst_data (6)-
48)*1000 ...
    + (gst_data (7)-48)*100 + (gst_data (8)-48)*10 + (gst_data (9)-48)


L_loadcell = L_loadcell + 1 ;

value_gst (L_loadcell) = (gst_data (11)-48)*100000000 + (gst_data (12)-
48)*10000000 + (gst_data(13)-48)*1000000 + ...
    (gst_data (14)-48)*100000 + (gst_data (15)-48)*10000 + (gst_data
(16)-48)*1000 ...
    + (gst_data (17)-48)*100 + (gst_data (18)-48)*10 + (gst_data (19)-
48)

L_loadcell = L_loadcell + 1 ;

value_gst (L_loadcell) = (gst_data (21)-48)*100000000 + (gst_data (22)-
48)*10000000 + (gst_data(23)-48)*1000000 + ...
    (gst_data (24)-48)*100000 + (gst_data (25)-48)*10000 + (gst_data
(26)-48)*1000 ...
    + (gst_data (27)-48)*100 + (gst_data (28)-48)*10 + (gst_data (29)-
48)

L_loadcell = L_loadcell + 1 ;

value_gst (L_loadcell) = (gst_data (31)-48)*100000000 + (gst_data (32)-
48)*10000000 + (gst_data(33)-48)*1000000 + ...
    (gst_data (34)-48)*100000 + (gst_data (35)-48)*10000 + (gst_data
(36)-48)*1000 ...
    + (gst_data (37)-48)*100 + (gst_data (38)-48)*10 + (gst_data (39)-
48)

L_loadcell = L_loadcell + 1 ;



enc.scale4 = value_gst (4) ;
```

```
enc.scale3 = value_gst (3) ;
enc.scale2 = value_gst (2) ;
enc.scale1 = value_gst (1) ;

if(enc.record == 0)
  enc.record = 1 ;

k = 1;
for z = 1 : 1 : 100
    P1 (k) = enc.deger1((z-1)*4 + 1)  ;
    P2 (k) = enc.deger1((z-1)*4 + 2)  ;
    P3 (k) = enc.deger1((z-1)*4 + 3)  ;
    P4 (k) = enc.deger1((z-1)*4 + 4)  ;
    k = k + 1;
    P1 (k) = enc.deger2((z-1)*4 + 1)  ;
    P2 (k) = enc.deger2((z-1)*4 + 2)  ;
    P3 (k) = enc.deger2((z-1)*4 + 3)  ;
    P4 (k) = enc.deger2((z-1)*4 + 4)  ;
    k = k + 1;
    P1 (k) = enc.deger3((z-1)*4 + 1)  ;
    P2 (k) = enc.deger3((z-1)*4 + 2)  ;
    P3 (k) = enc.deger3((z-1)*4 + 3)  ;
    P4 (k) = enc.deger3((z-1)*4 + 4)  ;
    k = k + 1;
    P1 (k) = enc.deger0((z-1)*4 + 1) ;
    P2 (k) = enc.deger0((z-1)*4 + 2)  ;
    P3 (k) = enc.deger0((z-1)*4 + 3)  ;
    P4 (k) = enc.deger0((z-1)*4 + 4)  ;
    k = k + 1;
end


k = 1;
for z = 1 : 1 : 100
    T (k) = enc.V1_gr ;
    k = k + 1;
    T (k) = enc.V2_gr;
    k = k + 1;
    T (k) = enc.V3_gr ;
    k = k + 1;
            T (k) = 0 ;
            k = k + 1;
end



P = [ P1 ; P2 ; P3 ; P4 ] ;


disp('network')
enc.net_AD = newlind(P,T);
enc.net_AD.biasConnect = 0;
wts = enc.net_AD.IW{1,1}
```

```
R = [(enc.deger1(1) - enc.deger0(1)); (enc.deger1(2) - enc.deger0(2));
(enc.deger1(3)-enc.deger0(3)); (enc.deger1(4)-enc.deger0(4))]

enc.Y = sim(enc.net_AD,P)

plot(handles.axes_1,enc.Y,T)
handles.axes_2
plot(handles.axes_2,T-enc.Y)
colormap(handles.axes_3,[1 0 0;0 1 0;0 0 1])
plot(handles.axes_4,P2,T)


else

        enc.Y = sim(enc.net_AD,[(enc.scale1 - enc.deger0(1));
(enc.scale2 - enc.deger0(2)); (enc.scale3 - enc.deger0(3)); (enc.scale4
- enc.deger0(4))])

        set(handles.edit22, 'String', num2str(enc.Y));

        if mod(enc.Y,0.05) > 0.025
            enc.Y = enc.Y - mod(enc.Y,0.05) + 0.05;
        else
            enc.Y = enc.Y - mod(enc.Y,0.05);
        end

        set(handles.text72, 'String', num2str(enc.Y));
        set(handles.text74, 'String', num2str(enc.Y - enc.Sifir));
        set(handles.text76, 'String', num2str(Olcum_degeri));

        plot(handles.axes_1,enc.old_time, enc.Y)

        enc.old_time = enc.old_time + 1 ;
end


 setappdata(handles.figure1,'enc_data', enc);

guidata(handles.figure1,handles)


% --- Executes on button press in togglebutton5.
function togglebutton5_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton5




% --- Executes on button press in pushbutton22.
```

```
function pushbutton22_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%V_Zero = [2 34.823456789 198274653.6024 93 0 45.679 4 3]

enc = getappdata(handles.figure1,'enc_data');

enc.V1_gr = str2num(get(handles.edit15, 'String'));
enc.V2_gr = str2num(get(handles.edit18, 'String'));
enc.V3_gr = str2num(get(handles.edit19, 'String'));
enc.V4_gr = str2num(get(handles.edit20, 'String'));


      k = 1;
for z = 1 : 1 : 400
    V_Zero (k*5 + 1) = enc.deger0(z);
    V_Zero (k*5 + 2) = enc.deger1(z);
    V_Zero (k*5 + 3) = enc.deger2(z) ;
    V_Zero (k*5 + 4) = enc.deger3(z) ;
    V_Zero (k*5 + 5) = enc.deger4(z) ;

    k = k + 1;
    z;
    k;
end


V_Zero (1) =  enc.V1_gr ;
V_Zero (2) =  enc.V2_gr ;
V_Zero (3) =  enc.V3_gr ;
V_Zero (4) =  enc.V4_gr ;
V_Zero (5) =  enc.Sifir ;

V_Zero

fid = fopen('ENC_Matfile.txt', 'w+');
fwrite(fid, V_Zero, 'double');
fclose(fid);

guidata(hObject,handles)


function edit22_Callback(hObject, eventdata, handles)
% hObject    handle to edit22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit22 as text
%        str2double(get(hObject,'String')) returns contents of edit22
as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit22_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```