

ÖZET

Kalay Kaplama süreci, tel çekme sanayisinde önemli bir yer teşkil etmektedir. Elektrik ve elektronikteki gelişmeye bağlı olarak kalaylı telin kullanım alanının genişlemesi, yüksek talep miktarı ve üretim sürecinin karmaşık olmayışı nedeniyle yaygın olarak üretilmektedir. Literatürdeki çalışmalar kaplama kalitesi, ürünün iletkenliği ve katkı malzemelerinin kaplama üzerine etkileri konularında yoğunlaşmıştır. Tabi bu çalışmaların hepsi dolaylı veya direk olarak üretim maliyetini azaltmaya yönelik çalışmalardır.

Üretime başlamak için kullanılan parametreler, deneysel verilere dayanmaktadır. Üretim ortamındaki şartların değişken olması sebebiyle kalay kalınlığı istenilen sınırlar dahilinde elde edilememektedir. Bu tezde, elde edilmek istenen kalay kaplama kalınlığına doğru ve hızlı bir şekilde ulaşmak amaçlanmıştır. Bu amaç doğrultusunda toplanan veriler, Yapay Sinir Ağları (Artificial Neural Network) ve Cevap Yüzeyi (Response Surface) metotları yardımı ile üretim ön hazırlık değerlerini tahmin eden matematiksel modeller oluşturulmasında kullanılmıştır. Sonuç olarak Cevap Yüzeyi metodu ile altıncı dereceden polinom kullanılarak oluşturulan matematiksel model % 4.9 yanılma oranı vererek başarılı olmuştur.

SUMMARY

Tin plating process, takes an important place in wire drawing industry. Usage of tin-plated wire grows wider according to the development of Electrical and Electronical Industry. High demand and easy production process makes tin plated wire production mostly preferred. Researches in the litterateur focus on plating quality, conductivity of product and the effects of additives on plating. All these researches are directed towards low cost production.

The parameters required for production are based on experimental data. Since production conditions can vary, coating thickness is very difficult to achieve within the required limits. In this study, to reach directly to the required coating thickness is targeted. Using Artificial Neural Network and Response Surface methods exploiting the experimental data are utilized to create mathematical models to predict coating thickness.

TEŞEKKÜR

Bu tez çalışması süresince yardımını ve hoşgörüsünü esirgemeyen tez danışmanım Yrd. Doç. Dr. Hasan KURTARAN'a,

Her zaman yanımda olan ve bana olan güvenini hiç azaltmayan Çiğdem İLERİ'ye, yardımlarını esirgemeyen Özgür KARACASU ve ailesine, moral destek ve katkılarını eksik etmeyen Ayhan FERİDUN ve diğer iş arkadaşlarıma,

Ve her zaman maddi ve manevi destekleriyle yanımda olan ve bu günlere gelmemi sağlayan ANNEM VE RAHMETLİ BABAMA teşekkürlerimi sunarım.

İÇİNDEKİLER DİZİNİ

ÖZET	i
SUMMARY	ii
TEŞEKKÜR	iii
İÇİNDEKİLER DİZİNİ	iv
TABLolar DİZİNİ	v
ŞEKİLLER DİZİNİ	vi
GRAFİKLER DİZİNİ	vii
1. GİRİŞ	1
1.1. Amaç	2
1.2. Sınırlar	2
1.3. Tezin Yapısı	3
2. KALAY KAPLAMA SÜRECİ	4
2.1. Kalay Kaplama Elemanları	4
2.2. Deneysel Veri Toplama	5
3. MATEMATİK MODEL OLUŞTURMA	13
4. YAPAY SİNİR AĞLARI	15
4.1. Yapay Sinir Ağlarının Tarihçesi	15
4.2. Yapay Sinir Ağlarının Tanımı	15
4.3. Yapay Sinir Ağlarının Genel Yapısı	17
4.4.1. Yapay Sinir Hücresi (Nöron)	18
4.4. İleri Sürümlü Yapay Sinir Ağları	20
4.5.1. Metodoloji	20
4.5. Hatanın Geri Yayılımı	22
4.6. Uygulama	24
4.7. Sonuçlar	26
5. CEVAP YÜZEYİ (RESPONSE SURFACE) YÖNTEMİ	30
5.1. Cevap Yüzeyi Tarihçesi	30
5.2. Cevap Yüzeyi Tanımı	30
5.3. Metodoloji	31
5.4. Uygulama	32
5.5. Sonuçlar	34
6. SONUÇ VE ÖNERİLER	37
KAYNAKLAR	43
ÖZGEÇMİŞ	44
EK 1 : YAPAY SİNİR AĞLARI MATLAB ARAYÜZÜ	45
EK 2 : CEVAP YÜZEYİ	57

TABLOLAR DİZİNİ

Tablo	Sayfa
2.1 1.33 mm'lik telin kalay kalınlığı değer tablosu.	7
2.2 1.60 mm'lik telin kalay kalınlığı değer tablosu.	8
2.3 1.80 mm'lik telin kalay kalınlığı değer tablosu.	9
2.4 2.20 mm'lik telin kalay kalınlığı değer tablosu.	10
2.5 2.60 mm'lik telin kalay kalınlığı değer tablosu.	11
2.6 Tüm değerler	12
4.1 2 katmanlı, $\eta = 0,000010$ olan örnekler	27
4.2 2 katmanlı, $\eta = 0,000500$ olan örnekler	27
4.3 3 katmanlı, $\eta = 0,000010$ olan örnekler	28
4.4 3 katmanlı, $\eta = 0,000500$ olan örnekler	29
5.1 Lineer model kullanıldığında elde edilen sonuçlar.	34
5.2 Parabolik model kullanıldığında elde edilen sonuçlar.	34
5.3 Kübik model kullanıldığında elde edilen sonuçlar.	35
5.4 4. dereceden model kullanıldığında elde edilen sonuçlar.	35
5.5 5. dereceden model kullanıldığında elde edilen sonuçlar.	35
5.6 6. dereceden model kullanıldığında elde edilen sonuçlar.	36
6.1 Yapay Sinir Ağları metodunun tahmin değerlerinin gerçek değerler ile karşılaştırılması.	38
6.2 Cevap Yüzeyi metodunun tahmin değerlerinin gerçek değerler ile karşılaştırılması.	40
6.3 Cevap Yüzeyi ve Yapay Sinir Ağları metotlarının hata yüzdeleri karşılaştırması.	41

ŞEKİLLER DİZİNİ

Şekil	Sayfa
2.1 Kalay kaplama süreci.	4
3.1 Matematiksel model	14
4.1 Yapay Sinir Ağlarının Yapısı	17
4.2 Yapay sinir hücresinin yapısı.	18
4.3 İki gizli katmanlı, üç giriş, bir çıkış elemanı olan bir yapay sinir ağı.	20
4.1 İleri Sürümlü bir ağ içerisinde bulunan katmanların ve nöronların sıralamaları	21
4.1 Çıkış katmanında hatanın geriye yayılımı	22
4.2 Gizli katmanlarda hatanın geriye yayılımı	23
5.1 Cevap Yüzeyi yönteminde kullanılan uygulama adımları	31

GRAFİKLER DİZİNİ

Grafik	Sayfa
2.1 1.33 mm'lik telin değişik hız ve akım değerlerinde (A) kaplanması sonucu elde edilen değerler.	7
2.2 1.60 mm'lik telin değişik hız ve akım değerlerinde (A) kaplanması sonucu elde edilen değerler.	8
2.3 1.80 mm'lik telin değişik hız ve akım değerlerinde (A) kaplanması sonucu elde edilen değerler.	9
2.4 2.20 mm'lik telin değişik hız ve akım değerlerinde (A) kaplanması sonucu elde edilen değerler.	10
2.5 2.60 mm'lik telin değişik hız ve akım değerlerinde (A) kaplanması sonucu elde edilen değerler.	11

1. GİRİŞ

Üretim sektörünün istisnasız her alanında, gün geçtikçe artan bir rekabet ortamı bulunmaktadır. Bu artan rekabet ortamı işletmelere, kullandıkları üretim yöntemlerinde sürekli iyileşmeye yönelik çalışmalar yapma mecburiyetini de beraberinde getirmektedir. Yapılan iyileştirmeler genel olarak sistemlerin teknik yeterlilikleri ile ilgili iyileştirmeler yapmaya yöneliktir. Buna ek olarak, kullanılan yöntemin teknik özelliklerinden ziyade, üretim sürecinde iyileştirme çalışmaları popülerlik kazanmaya başlamıştır.

Literatürdeki kalay kaplama süreci üzerine yapılan çalışmalarda, kalay kaplama miktarına etkisi olan değişkenlerin, diğer değişkenler sabit tutulduğunda, ne gibi bir etkisi olacağı gibi teknik değerler ile ilgili iyileştirmeler yapmayı hedeflemiştir. Örnek vermek gerekirse; hızı ve akımı sabitleyerek kaplanacak telin çapının kaplamaya ne gibi etkileri olmuştur, tel çapı ve akım sabitlendiğinde hız değişimi ne gibi sonuçlar vermiştir. Bunların haricinde kullanılan elektrolitik çözeltinin bileşenleri, çözelti sıcaklığı gibi etkenler hakkında da yaptıkları etki hakkında çalışmalar yapılmıştır.

Bu çalışmada üretim değerlerinin en kısa ve en doğru şekilde sağlanması hedeflenmiştir. Çalışma yapılacak süreç şu şekildedir. Belirli bir çaptaki tele, istenilen miktarda kalay kaplama yapılacaktır. Bunun için istenilen kalay miktarına göre, kaplama işlemi uygulanacak telin, hangi hızda çekileceği ve kaç amper akım verileceği değerlerine makine ile gelen tablodan bakılır. Daha sonra üretim aşamasına geçilir. Test yapmaya yetecek kadar tel kaplama işlemi yapıldıktan sonra, kaplama kalınlığı kontrol edilir. Eğer istenilen kalay kalınlığına ulaşılamamışsa gerekli düzenlemeler yapılır. Tekrar üretime geçilir. Bu işleme istenilen kalay kaplama kalınlığına ulaşana kadar devam edilir. İstenilen değere ulaşıldığında ise normal üretime geçilir.

İstenilen değerlere ulaşılması ne kadar uzarsa yani ne kadar çok test yapılırsa, o kadar çok kayıp verilir. Zaman kaybı, işçilik kaybı, malzeme kaybı v.s.. Bahsedilen

her kayıp üretim maliyetine ek bir yük getirmektedir. Bu probleme çözüm olarak aradığımız şey gerçek değere ulaşma süresini minimuma indirmektir. Böylece ek maliyetlerden kurtulmak mümkün olacak ve istenilenlere cevap verme süresi de kısılacaktır. Bunun içinde istenilen sonucu sağlayacak makine ayarlarına en yakın değerlerin belirlenmesi gerekmektedir.

Çalışma yapılacak olan şartların belirlenmesi için, mevcut yöntem, prosesin uygulandığı makine ile beraber gelen ve kullanılan kalay kaplama makinesine özel olarak hazırlanmış, deneysel üretimler sonucu elde edilen verilerden oluşmuş bir tablo yardımı ile tahmin edilmektedir. Fakat kullanım sonucu makine şartlarının değişmesi ve makinenin içinde bulunduğu ortamın laboratuvar ortamından farklı olması nedeni ile belirlenen değerlerin kullanımı sonucu elde edilen sonuçlar beklentileri karşılamamaktadır.

1.1. Amaç

Kalaylı tel üretim süreci ön hazırlık değerlerinin hatalı seçilmesi sonucu oluşan ek maliyetlerin ortadan kaldırılması amaçlanmaktadır.

1.2. Sınırlar

Bu çalışmada kullanılan verilerde tel çekme hızları 2 m/sn ile 7.5 m/sn arasında, uygulanan akım değerleri 600 A ile 2600 A arasında kalay kaplanan tel çapları ise 1.33 mm ile 2.60 mm arasında değişmektedir. Toplanan bu veriler, üretim sırasında yapılan deneylerden toplanmıştır.

1.3. Tezin Yapısı

Altı bölümden oluşan bu tezin ikinci kısmında, çalışma yapılan kalay kaplama süreci ile ilgili bilgiler verilmiştir. Ayrıca çalışmada kullanılan veriler de tablo halinde yine bu bölümde yer almaktadır.

Üçüncü bölümde, matematiksel model oluşturma ile ilgili genel bir bilgi verilerek çalışmanın çözüm aşamasında kullanılan metotlar tanıtılmıştır.

Dördüncü bölümde, çalışmada kullanılan model oluşturma metotlarından ilki olan Yapay Sinir Ağları yönteminin tanımlaması yapılmıştır. Daha sonra metodoloji ve programın girişlerinin yapılışı gösterilmiştir. Son olarak elde edilen sonuçlar sunulmuştur.

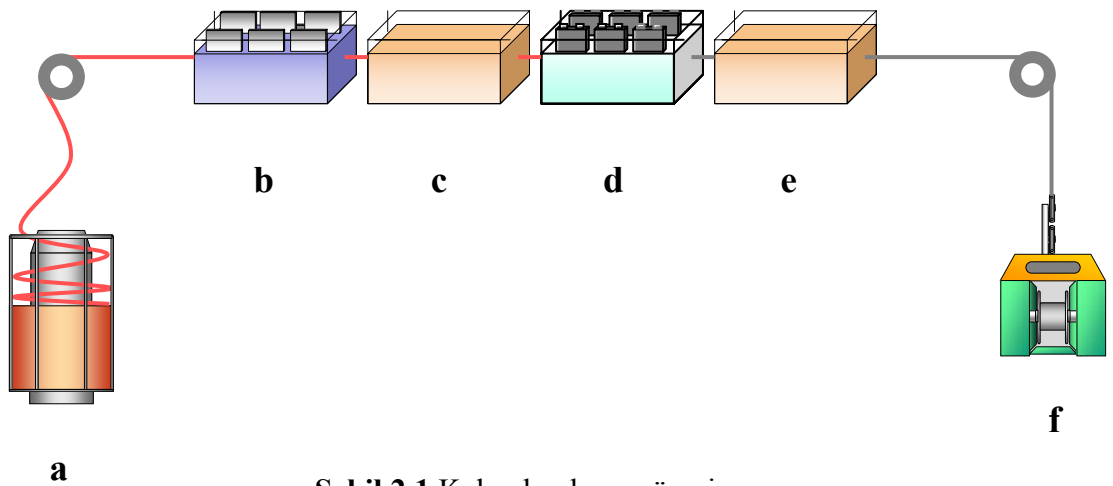
Beşinci bölümde diğer metot Cevap Yüzeyi ile ilgili bilgiler verilmiştir. Metodoloji ve program girişlerinin yapılışı anlatıldıktan sonra bölümün sonunda elde edilen sonuçlar tablo halinde verilmiştir.

Son bölümde matematiksel modeller kullanılarak elde edilen sonuçlar değerlendirilmiştir.

2. KALAY KAPLAMA SÜRECİ

Bu tezde kullanılan kalay kaplama süreci, elektrolitik metal kaplama esasına dayanmaktadır. Üretimin yapıldığı düzenekte bir elektroliz havuzu bulunmaktadır. Bu havuza anot olarak Kalay metali yerleştirilmiştir. Havuzun kenarlarında sabitlenen kalay anotları, yeterli akım verilmeye başlanınca çözünerek , havuz içinden geçen bakır telin üzerine bağlanmaya başlamaktadır. Kaplama yapılan telin kalınlığı, tel çekme hızının artması veya azalması ve son olarak havuza verilen akım kuvvetinin değiştirilmesi ile değişik kalınlıklarda kalay kaplaması yapmak mümkün olmaktadır. Kısaca süreçten aşağıda bahsedilmiştir.

2.1. Kalay Kaplama Elemanları



Şekil 2.1 Kalay kaplama süreci.

Kalay kaplama sürecinde kullanılan elemanlar:

a. Kaplama işlemi yapılacak olan çıplak bakır tel. Burada sepetten üretim yapılacağı gibi, makaraya üretilmiş tel de kullanılmaktadır.

b. Yağ alma havuzu. Malzeme üzerinde üretim safhasında oluşan yağ kalıntılarının temizlenmesinde kullanılır.

c. Dekapaj kaplama havuzu. Malzemenin yüzeyinin oksidasyon ve korozyona karşı daha dayanıklı hale getirilmesinde kullanılmaktadır.

d. Kaplama havuzu. Esas üretim bu kısımda gerçekleşmektedir. Elektroliz yöntemi ile havuzun içinde bulunan kalay anotları çözünerek, havuzun içinden geçen bakır telin üzerine yapışmak sureti ile kaplama işlemi gerçekleşir.

e. Yıkama havuzu. Kaplama işleminden sonra, tel üzerinde kalan asitli elektroliz çözeltisinin temizlenmesi işlemi bu bölümde yapılmaktadır.

f. Ürün. Bu kısımda da yine başlangıç aşamasında olduğu gibi makara veya sepete üretim yapabilmek mümkündür.

Kalay kaplama havuzunda, Fluoborat ve Sülfat havuzları seçeneklerinden daha basit ve kontrolü daha kolay olduğu için ikinci seçenek olan Sülfat havuzları seçilmiştir. (ÇAKIR, 1984, s.5) Bu havuzlar kalaysülfat ve sülfürik asitten oluşmaktadır. Havuzda kullanılacak kalay sülfat ve sülfürik asit arasında yakın bir ilişki bulunmaktadır. Havuzda bulunan sülfürik asit miktarının artışı çözünecek kalay miktarını da arttıracaktır. Eklenen bazı katkı malzemeleri ile verimlilik arttırımı da söz konusu olmaktadır. Ayrıca havuz sıcaklığının da çözünmeye etkisi olmaktadır.

2.2. Deneysel Veri Toplama

Yapılan deneyler sırasında yukarıda bahsedilen etkenler sabitlenerek, tüm örneklerin, yani asit miktarı ve havuz sıcaklığı gibi değişken değerleri sabitlenerek, aynı şartlar altında alınması sağlanmıştır. Sadece etkileri tam olarak gözlenebilecek olan değişkenler seçilmiştir. Bu değişkenler:

1. Kaplanacak olan telin kalınlığı,
2. Tel çekme hızı,
3. Havuza verilecek olan akım miktarıdır.

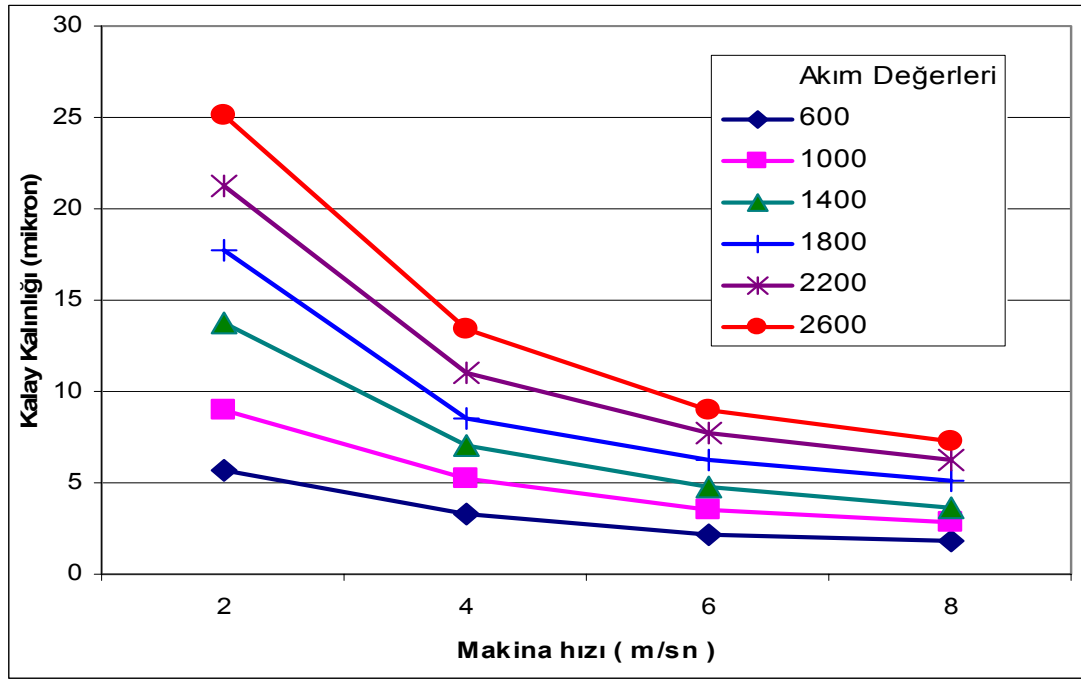
Bu üç değişken değiştirilerek, farklı kalınlıklarda kalay kaplaması elde etmek mümkündür. Gerçek uygulamada kullanılan teçhizatla beraber gelen ve her bir çap ve uygulanacak olan akım ve hız değerleri için deney yapılmış ve bu sonuçlardan

faydalanılarak üretilmiş bir değer tablosu bulunmaktadır. Fakat bu tabloda bulunan değerler hem alındıkları ortamın deneysel olmasından yani çalışma şartlarına uymayışından hem de kullanılan teçhizatda uzun süre kullanım sonucu oluşan eskimeden dolayı yeterliliğini kaybetmiştir.

Kullanılan tablodaki değerler prosesin başlangıç değerlerini oluşturmaktadır ve bu değerlerin doğruluğu tamamıyla prosesin istenilen zamanda başlamasını etkilemektedir. Üretim süreci şu şekilde yürümektedir: İstenilen çaptaki tele kaplanacak kalay kalınlığı değerlerine bakılarak bir hız ve akım değeri seçilmektedir. Daha sonra bir deneme üretimi yapılmakta ve bu denemeden gelen sonuçlara göre eğer istenilen kalınlık sağlanamadı ise gerekli değişiklikler yapıp tekrar bir deneme üretimine geçilmektedir. Bu süreç istenilen kaplama kalınlığına ulaşıncaya kadar devam etmektedir. Sonuç olarak; başlangıç değerlerimiz ne kadar doğru olursa o kadar az deneme ile sonuca gidilecektir.

Alınan örneklerin ölçümleri laboratuarda yapılır. Kalınlık ölçüm süreci basit olarak şu şekilde olmaktadır:

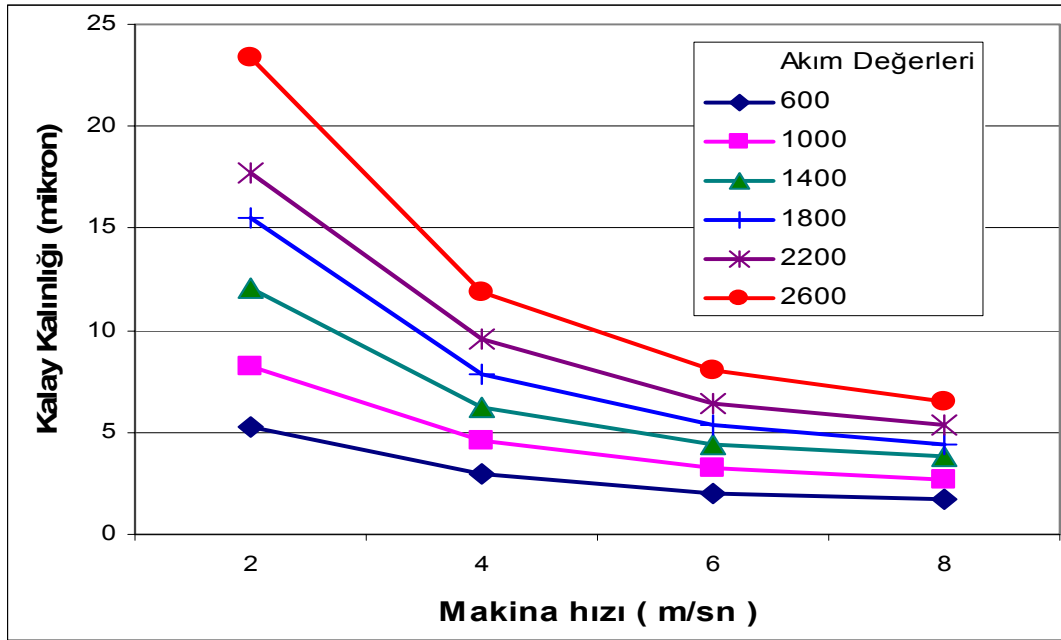
İşletmeden getirilen örnek ölçüm cihazında bulunan asit su karışımına daldırılır. Burada elektroliz işlemi ile numune üzerindeki kalay çözülür ve karışımın içinde bulunan katoda yapışır. Katodun bağlı olduğu gösterge, katoda bağlanan kalay miktarını gösterir. Gösterge sabitlendiğinde işlem tamamlanmış olur. Elde edilen değer numune telin çapına göre değişen bir sabite bölünerek kalay kalınlığı bulunur. Yapılan deneylerden elde edilen veriler aşağıda verilmiştir. Grafiklerin her biri, ayrı çap değerleri için yapılan deneyler sonucunda elde edilen sonuçları göstermektedir.



Grafik 2.1 1.33 mm'lik telin değişik hız ve akım değerlerinde (A) kaplanması sonucu elde edilen değerler.

Tablo 2.1 1.33 mm'lik telin kalay kalınlığı değer tablosu.

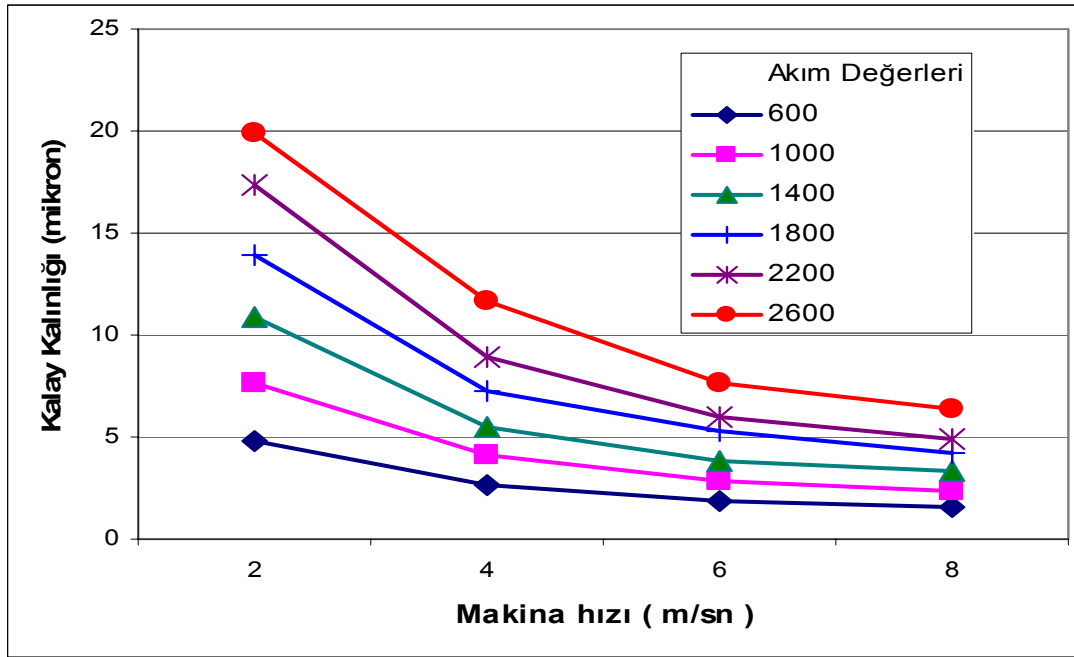
		AKIM					
		600	100	1400	1800	2200	2600
HIZ	2	5,65	8,93	13,74	17,77	21,28	25,14
	4	3,32	5,17	7,03	8,54	11,05	13,40
	6	2,20	3,50	4,72	6,22	7,71	9,01
	7,5	1,86	2,81	3,58	5,13	6,23	7,22



Grafik 2.2 1.60 mm'lik telin değişik hız ve akım değerlerinde (A) kaplanması sonucu elde edilen değerler.

Tablo 2.2 1.60 mm'lik telin kalay kalınlığı değer tablosu.

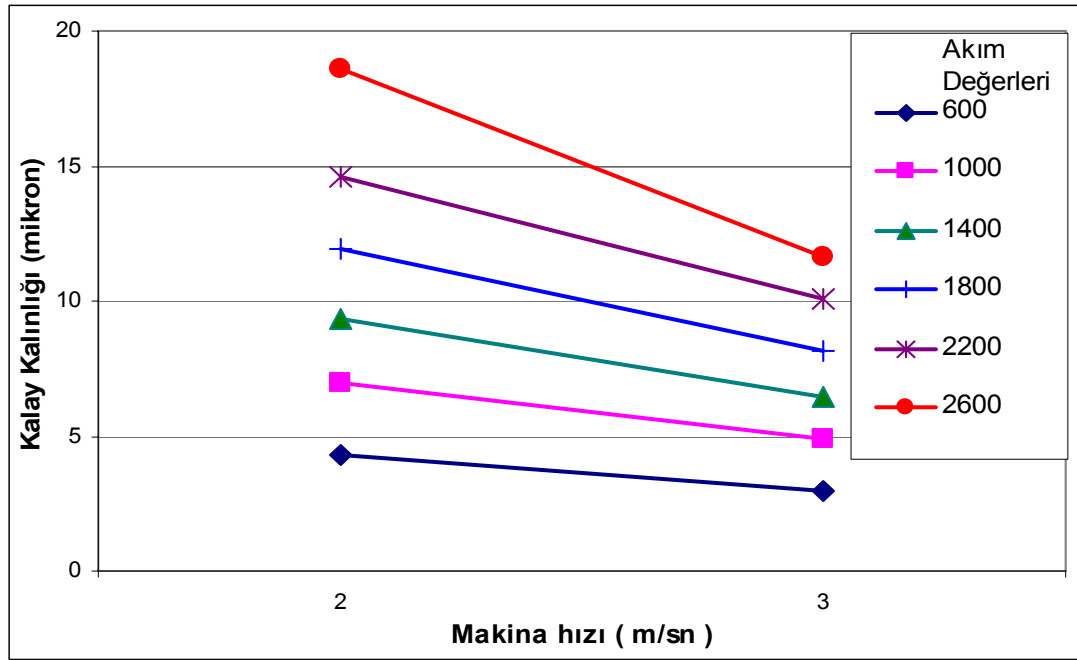
		AKIM					
		600	100	1400	1800	2200	2600
HIZ	2	5,30	8,24	12,05	15,51	17,75	23,39
	4	2,94	4,56	6,20	7,89	9,53	11,84
	6	2,06	3,28	4,42	5,36	6,46	8,07
	7,5	1,77	2,67	3,84	4,36	5,40	6,55



Grafik 2.3 1.80 mm'lik telin değişik hız ve akım değerlerinde (A) kaplanması sonucu elde edilen değerler.

Tablo 2.3 1.80 mm'lik telin kalay kalınlığı değer tablosu.

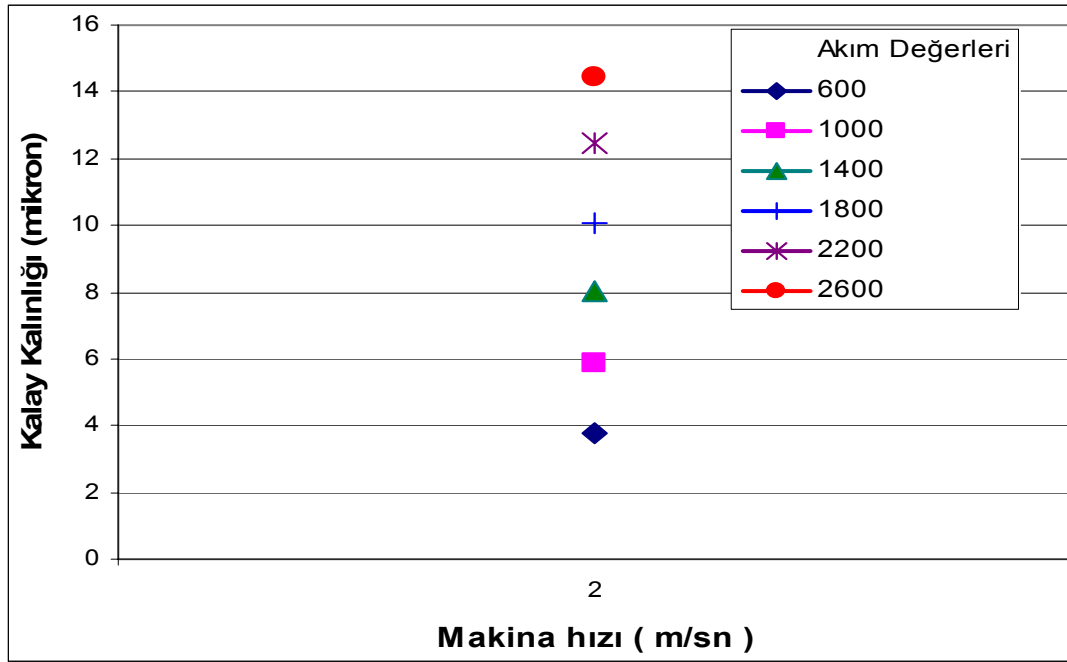
		AKIM					
		600	100	1400	1800	2200	2600
HIZ	2	4,78	7,67	10,91	13,87	17,32	19,93
	4	2,63	4,14	5,54	7,26	8,89	11,63
	6	1,84	2,86	3,84	5,31	6,02	7,63
	7,5	1,59	2,32	3,30	4,17	4,87	6,37



Grafik 2.4 2.20 mm'lik telin değişik hız ve akım değerlerinde (A) kaplanması sonucu elde edilen değerler.

Tablo 2.4 2.20 mm'lik telin kalay kalınlığı değer tablosu.

		AKIM					
		600	100	1400	1800	2200	2600
HIZ	2	4,32	6,98	9,33	11,93	14,62	18,56
	3	3,00	4,88	6,48	8,17	10,07	11,65



Grafik 2.5 2.60 mm'lik telin değişik hız ve akım değerlerinde (A) kaplanması sonucu elde edilen değerler.

Tablo 2.5 2.60 mm'lik telin kalay kalınlığı değer tablosu.

		AKIM					
		600	100	1400	1800	2200	2600
HIZ	2	3,76	5,88	8,02	10,07	12,49	14,45

Tablo 2.6 Tüm değerler.

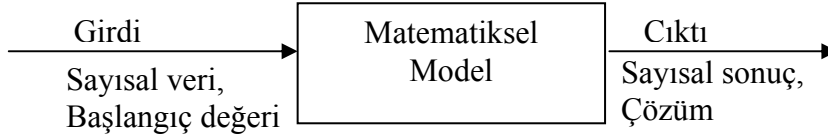
Sıra No	Çap (mm)	Akım (A)	Hız (m/sn)	Kalay Kalınlığı (µm)	Sıra No	Çap (mm)	Akım (A)	Hız (m/sn)	Kalay Kalınlığı (µm)
1	1,33	600	2,0	5,651	46	1,60	2600	4,0	11,840
2	1,33	600	4,0	3,320	47	1,60	2600	6,0	8,069
3	1,33	600	6,0	2,198	48	1,60	2600	7,5	6,553
4	1,33	600	7,5	1,863	49	1,80	600	2,0	4,782
5	1,33	1000	2,0	8,934	50	1,80	600	4,0	2,633
6	1,33	1000	4,0	5,174	51	1,80	600	6,0	1,840
7	1,33	1000	6,0	3,500	52	1,80	600	7,5	1,590
8	1,33	1000	7,5	2,811	53	1,80	1000	2,0	7,671
9	1,33	1400	2,0	13,739	54	1,80	1000	4,0	4,136
10	1,33	1400	4,0	7,030	55	1,80	1000	6,0	2,863
11	1,33	1400	6,0	4,719	56	1,80	1000	7,5	2,319
12	1,33	1400	7,5	3,580	57	1,80	1400	2,0	10,910
13	1,33	1800	2,0	17,766	58	1,80	1400	4,0	5,539
14	1,33	1800	4,0	8,537	59	1,80	1400	6,0	3,835
15	1,33	1800	6,0	6,219	60	1,80	1400	7,5	3,303
16	1,33	1800	7,5	5,128	61	1,80	1800	2,0	13,874
17	1,33	2200	2,0	21,280	62	1,80	1800	4,0	7,261
18	1,33	2200	4,0	11,054	63	1,80	1800	6,0	5,310
19	1,33	2200	6,0	7,713	64	1,80	1800	7,5	4,171
20	1,33	2200	7,5	6,234	65	1,80	2200	2,0	17,322
21	1,33	2600	2,0	25,143	66	1,80	2200	4,0	8,887
22	1,33	2600	4,0	13,399	67	1,80	2200	6,0	6,017
23	1,33	2600	6,0	9,011	68	1,80	2200	7,5	4,874
24	1,33	2600	7,5	7,217	69	1,80	2600	2,0	19,934
25	1,60	600	2,0	5,296	70	1,80	2600	4,0	11,633
26	1,60	600	4,0	2,942	71	1,80	2600	6,0	7,632
27	1,60	600	6,0	2,057	72	1,80	2600	7,5	6,372
28	1,60	600	7,5	1,768	73	2,20	600	2,0	4,320
29	1,60	1000	2,0	8,239	74	2,20	600	3,0	2,996
30	1,60	1000	4,0	4,559	75	2,20	1000	2,0	6,981
31	1,60	1000	6,0	3,276	76	2,20	1000	3,0	4,883
32	1,60	1000	7,5	2,667	77	2,20	1400	2,0	9,327
33	1,60	1400	2,0	12,053	78	2,20	1400	3,0	6,481
34	1,60	1400	4,0	6,202	79	2,20	1800	2,0	11,934
35	1,60	1400	6,0	4,416	80	2,20	1800	3,0	8,165
36	1,60	1400	7,5	3,835	81	2,20	2200	2,0	14,615
37	1,60	1800	2,0	15,506	82	2,20	2200	3,0	10,073
38	1,60	1800	4,0	7,891	83	2,20	2600	2,0	18,557
39	1,60	1800	6,0	5,355	84	2,20	2600	3,0	11,649
40	1,60	1800	7,5	4,359	85	2,60	600	2,0	3,763
41	1,60	2200	2,0	17,745	86	2,60	1000	2,0	5,884
42	1,60	2200	4,0	9,533	87	2,60	1400	2,0	8,019
43	1,60	2200	6,0	6,463	88	2,60	1800	2,0	10,072
44	1,60	2200	7,5	5,396	89	2,60	2200	2,0	12,486
45	1,60	2600	2,0	23,387	90	2,60	2600	2,0	14,453

3. MATEMATİK MODEL OLUŞTURMA

Gerçek dünyadaki bir olayın, sürecin veya birimlerden oluşan ve birimleri arasındaki iç ilişkiler yanında çevre ile dış ilişkilere göre işleyen bir sistemin belli bir anlatımına model denir. (<http://science.ankara.edu.tr/~ozbek/model.htm> , 2004) Sözel, fiziksel ve sayısal yöntemler kullanılarak bir model oluşturulması işlemine ise **modelleme** denilir. Tanım olarak ele alınan olaya ilişkin teoride yer alan değişkenlerin ve bunların aralarındaki ilişkilerin gözlenebildiği, değişkenler arası ilişkilerin test edilmesi amacıyla yönelik bir gösterim işlemidir. (<http://science.ankara.edu.tr/~ozbek/gulcan.htm> , 2004) Model oluşturmak için sözel anlatım, taslak çizimler, 3 boyutlu fiziki modeller yada matematiksel ifadeler kullanılmaktadır.

Bir sayısal çözüm metodu olan matematiksel model oluşturma, gerektirdiği çok fazla basit aritmetik işlemleri ve buna bağlı olarak ta çok fazla zaman kaybı ve yüksek hata ihtimali sebebiyle geçmişte çok fazla tercih edilen bir yöntem olarak kullanılmamaktadır. Fakat gelişen bilgisayar teknolojisi ile bu yöntemleri günümüzde çok rağbet görmeye başlamıştır. (Karadeniz, 2001, s. 8) 1980’li yıllarda kişisel ev bilgisayarlarının da yaygınlaşması ile bu yöntemlerin kullanımı hız kazanmış ve bilgisayar teknolojisine paralel bir hızda gelişmeye devam etmektedir.

Matematiksel modellerin anlatım güçleri çok yüksektir. Girdi olarak deneysel ölçüm değerleri kullanılır ve oluşturulan model sayesinde gerçek hayatta gerçekleştirilmemiş deneylerin sonuçları tahmin edilir. Basit bir matematiksel model yapısı şekil ‘de verilmiştir.



Şekil 3.1 Matematiksel model

Bu çalışmada mühendislik uygulamalarında en çok tercih edilen Yapay Sinir Ağları ve Cevap Yüzeyi metotları kullanılarak Kalay kaplama süreci için yapılan deney sonuçlarının dışında kalan değerleri tahmin etmemize yarayan bir model oluşturulmaktadır.

4. YAPAY SİNİR AĞLARI

4.1. Yapay Sinir Ağlarının Tarihçesi

McCulloch ve Pitts ile 1943 yılında başlayan, Von Neumann ve Minsky ile devam eden Yapay Sinir Ağları çalışmaları, Widrow, Kohonen, Minsky, Papert, Grossberg ve Hopfield gibi bir çok bilim adamlarının katkılarıyla hızla gelişmiştir. 1987 yılında yapılan ilk Yapay Sinir Ağları sempozyumundan sonra Yapay Sinir Ağı uygulamaları yaygınlaşmıştır. Günümüzde, Yapay Sinir Ağları ile ilgili araştırmalar yapan çok sayıda bilim adamı ve araştırma grupları vardır.

4.2. Yapay Sinir Ağlarının Tanımı

Yapay Sinir Ağları, insan beyninin yeni bilgiler türetmek, yeni sonuçlar çıkarmak ve yeni bir şeyler keşfedebilmek için kullandığı öğrenme yolunu kullanarak herhangi bir yardım almadan otomatik olarak bu olayları gerçekleştirmek amacı ile geliştirilen bilgisayar sistemleridir [Öztemel, 2003, s.10]. Öğrenme yeteneği klasik matematiksel yöntemlerde oldukça zor veya mümkün olmamaktadır. Bu sebeple Yapay Sinir Ağları, programlanması çok zor veya mümkün olmayan olaylar için geliştirilmiş bilgi işleme ile ilgilenen bir bilgisayar bilim dalı olduğu söylenebilir.

Yapay Sinir Ağları deneyler sonucu elde edilmiş verileri kullanarak olayları öğrenebilen, olayın çevre etkenlerinden nasıl etkileneceğini ve nasıl tepkiler vereceğini tahmin eden bir bilgisayar sistemidir. Örneklerden elde edilen bilgileri kullanarak kendi deneyimlerini geliştirir ve daha sonra benzer şartları oluşturarak kendi sonuçlarını ortaya çıkarır.

Genelleme yapılacak olursa Yapay Sinir Ağları'nın uygulamaları aşağıdaki başlıklarda toplanabilir:

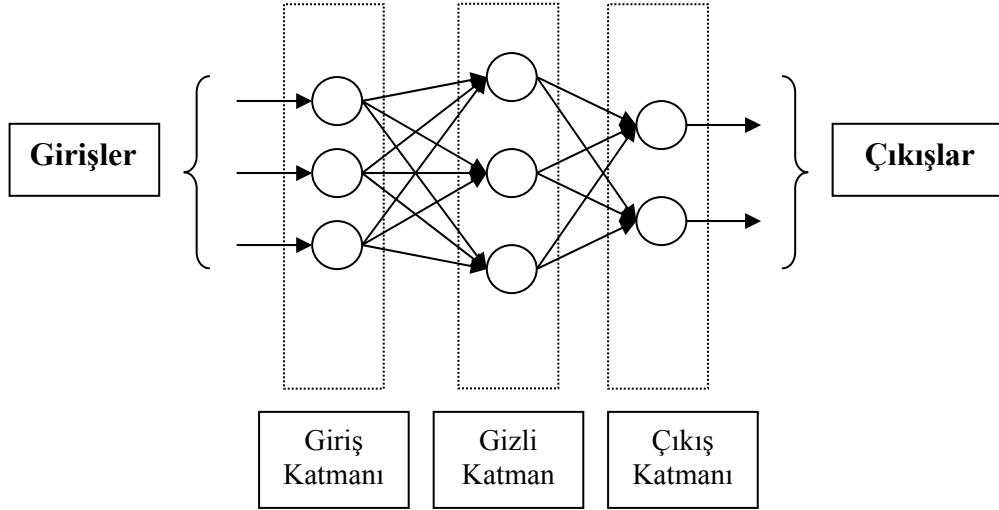
- Öngörü
- Sınıflandırma
- Veri birleştirilmesi
- Veri kavramlaştırılması
- Veri süzülmesi
- Resim veya görüntü işleme (Elmas, 2003, s.175)

Endüstriyel alanda kullanımı son zamanlarda artmakta olan Yapay Sinir Ağları, kalite kontrol, üretim süreci ve makine bakımı, cep telefonlarında ses ile arama, optimizasyon çalışmaları, robotik faaliyetler vb. bir çok yerde kullanılmaktadır.

Finansal uygulamalar olarak ta makro ekonomik tahminler, kredi kartı hilelerinin tahmini, banka kredilerinin değerlendirilmesi gibi tahmin ve karar verme mekanizmalarında karşımıza çıkmaktadır. Bunların haricinde askeri hedef takip sistemleri, radar kontrolleri, mayın detektörleri; sağlık alanında solunum hastalıklarının teşhisi, hamilelik ile ilgili takiplerde ve daha bir çok alanda uygulamalar bulunmakta ve gün geçtikçe daha çok alana yayılmaktadır. (Öztemel, 2003, s. 205)

4.3. Yapay Sinir Ağlarının Genel Yapısı

3 ana katmandan oluşur. Girdi katmanı, Gizli katman ve Çıktı katmanı.



Şekil 4.1 Yapay Sinir Ağlarının yapısı

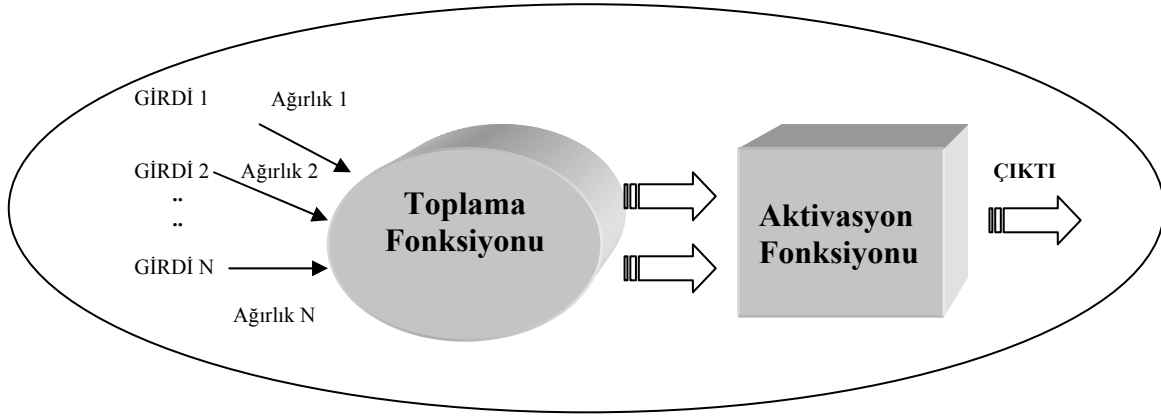
Şekil 4.2’de 3 adet giriş elemanı, 1 adet gizli katmanlı, 3 adet nöronlu ve 2 adet sonuç çıktısı olan bir ağa örnek verilmiştir. Problemin durumuna göre bu rakamlar değişebilmektedir.

- **Girdi Katmanı** : Bu katmanda bulunan elemanlar, dış dünyadan gelen verileri gizli katmanlara iletmekle sorumludurlar. Bazı durumlarda gelen veriler hiçbir değişime uğramadan gizli katman kısmına gönderilebilirler.
- **Gizli Katmanlar** : Çıkış katmanındakine benzer bir işlem söz konusudur. Seçilen transfer fonksiyonu ile işlenen veriler çıktı katmanına yönlendirilir. Problemin yapısına göre birden çok gizli katman olabilir.
- **Çıktı Katmanı** : Gizli katmanlardan gelen veriler toplanarak transfer fonksiyonu ile işleme tabi tutulur. Elde edilen çıktı dış dünyaya gönderilir.

Yapay Sinir Ağ’ları birbiri ile ilişki içinde olan sinir hücrelerinden oluşur bu hücrelere nöron adı verilir.

4.4.1. Yapay Sinir Hücresi (Nöron)

Yapay Sinir Hücreleri, Yapay Sinir Ağları'nı oluşturan yapı taşlarıdır. Yapay sinir hücreleri mühendislik biliminde proses elemanları olarak da adlandırılmaktadır.



Şekil 4.2 Yapay sinir hücresinin yapısı.

Her bir nöronun 5 temel elemanı vardır.

1-Girdiler: Bir yapay sinir hücresine verilen deney sonuçlarını ifade eder. Bunlar ağırlık öğrenmesi istenen örnekler tarafından belirlenir. Yapay sinir hücresine dış dünyadan olduğu gibi başka hücrelerden veya kendi kendisinden de bilgiler gelebilir.

2-Ağırlıklar: Ağırlıklar bir yapay hücreye gelen bilginin hücre üzerindeki etkisini gösterir. Şekil 4.1.'deki Ağırlık 1, Girdi 1' in hücre üzerindeki etkisini göstermektedir. Ağırlığın büyük yada küçük olması, önemli veya önemsiz olduğu anlamına gelmez. Eksi değerler önemsiz demek değildir. O nedenle artı veya eksi olması etkisinin pozitif veya negatif olduğunu gösterir. Sıfır olması ise herhangi bir etkinin olmadığını gösterir. Ağırlıklar değişken veya sabit değerler olabilirler.

3-Toplama fonksiyonu: Bu fonksiyon, bir hücreye gelen net girdiği hesaplar. Burada her gelen girdi değeri kendi ağırlığı ile çarpılarak toplanır. Böylece ağırlıklı girdilerin net toplamı bulunmuş olur.

$$S = \sum_i^n G_i A_i \quad (4.1)$$

(4.1) formülünde **S** toplam ağırlık değerini, **G** girdileri, **A** ağırlıkları ve **n** ise hücreye gelen toplam girdi sayısını ifade etmektedir. Her bir girdinin kendine özgü bir ağırlık değeri vardır. Ağırlık değerleri ile çarpılan girdilerin sonuçları toplanarak, nöronun bir diğer elemanı olan aktivasyon fonksiyonuna gönderilir.

4- Aktivasyon fonksiyonu: Bu fonksiyon hücreye gelen girdiyi işleme tabi tutarak, hücrenin bu girdiye sonuç olarak üreteceği değeri hesaplar. İsteğe göre bu kısımda değişik yapıda fonksiyonlar kullanabilmek mümkündür. Her bir nöron ayrı bir fonksiyonla çalışabilir. Uygun fonksiyonun seçimi, ancak deneme yanılma yöntemi ile belirlenebilmektedir.

Bu kısımda en çok tercih edilen fonksiyonlardan bazıları aşağıda gösterilmiştir.

$$\Psi_1(S) = \frac{1}{1 + e^{-S}} \quad \text{Sigmoid tipi aktivasyon fonksiyonu} \quad (4.2)$$

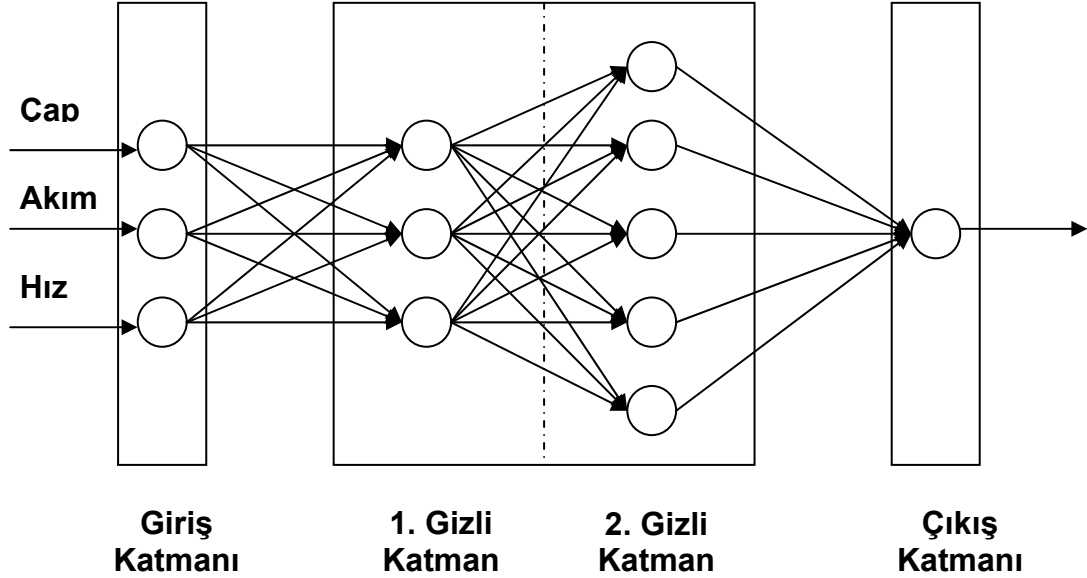
$$\Psi_2(S) = \frac{e^S - e^{-S}}{e^S + e^{-S}} = \tanh(S) \quad \text{Hiperbolik tanjant tipi aktivasyon fonksiyonu} \quad (4.3)$$

$$\Psi_3(S) = \begin{cases} 0 & S \leq 0 \\ 1 & S > 0 \end{cases} \quad \text{Sert geçişli tipte aktivasyon fonksiyonu} \quad (4.4)$$

Bu çalışmada tanjant ve logaritmik sigmoid fonksiyonları kullanılmıştır.

5- Nöronun çıktısı : Aktivasyon fonksiyonunun sonucuna verilen isimdir. Elde edilen sonuç dış dünyaya yada diğer bir nörona girdi olarak gönderilir. Nöronun oluşturduğu sonucu aynı zamanda kendisinde girdi olarak kullanabilir. Her nörondan sadece bir çıktı alınır. Bu tek çıktıyı birden çok yerde kullanmak mümkündür.

Bu tezde kullanılan yapay sinir ağlarından bir tanesi örnek olarak aşağıda verilmiştir.



Şekil 4.3 İki gizli katmanlı, üç giriş, bir çıkış elemanı olan bir yapay sinir ağı.

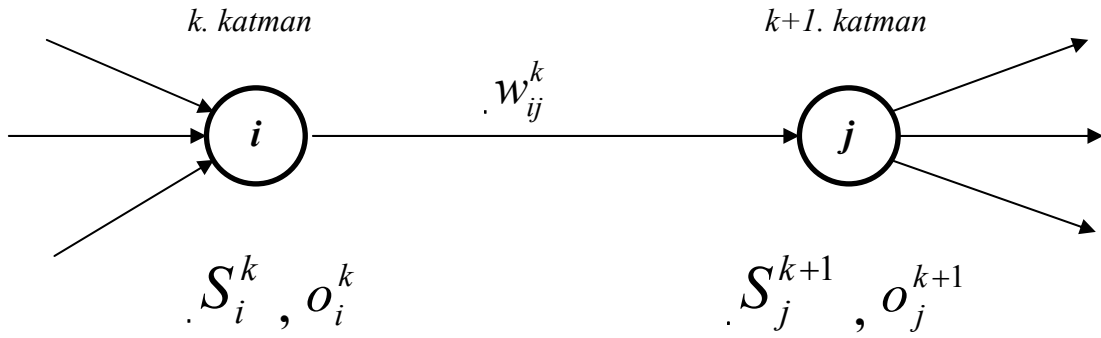
4.4. İleri Sürümlü Yapay Sinir Ağları

Bu tezde yapı olarak İleri Sürümlü Hata Geri Beslemeli Yapay Sinir Ağları kullanılmıştır. İleri sürümlü sinir ağlarının en tipik şekli Şekil 4.2.'de de gösterilmiş olan nöron modeli ile oluşturulmuş katmanların ardışık bir biçimde bir araya gelmesi ile kurulabilir. Daha öncede bahsedildiği üzere her bir katmandaki nöron sayısı ve girdi ve çıktı katmanları arasındaki gizli katman sayıları model oluşturulacak problemin yapısına göre değişmektedir.

4.5.1. Metodoloji

Ağ üzerindeki katmanlar k indisi ile sıralansın., ve L adet gizli katman olduğu varsayalım. Yani Şekil 4.2.'deki ağ için $L = 1$ olmaktadır. Eğer bu ağ yapısı içindeki k 'inci katmanın i 'inci numaralı nöronunu, $k+1$ 'inci katmanın j 'inci nöronuna bağlayan bağlantının ağırlığı da w_{ij}^k olarak ifade edilirse ve k 'inci katmanın i 'inci

numaralı nöronunun çıkış değeri o_i^k ile gösterilirse (Bkz. Şekil 4.3), $k+1$ 'inci katmanın j 'inci nöronunun net toplamı S_j^{k+1} ve çıkış değeri o_j^{k+1} (4.5) ve (4.6) denklemleri ile hesaplanır. (Efe ve Kaynak, 2000, s.14)



Şekil 4.4 İleri Sürümlü bir ağ içerisinde bulunan katmanların ve nöronların sıralamaları

$$S_j^{k+1} = \sum_{i=1}^{n_k} w_{ij}^k o_i^k \quad (4.5)$$

$$o_j^{k+1} = \Psi(S_j^{k+1}) \quad (4.6)$$

Yukarıdaki bağıntılarda:

S_j^{k+1} = k+1. katmandaki j. nöronun girişinde oluşan net toplam,

$\Psi(S_j^{k+1})$ = k+1. katmandaki j. nöronun transfer fonksiyonu

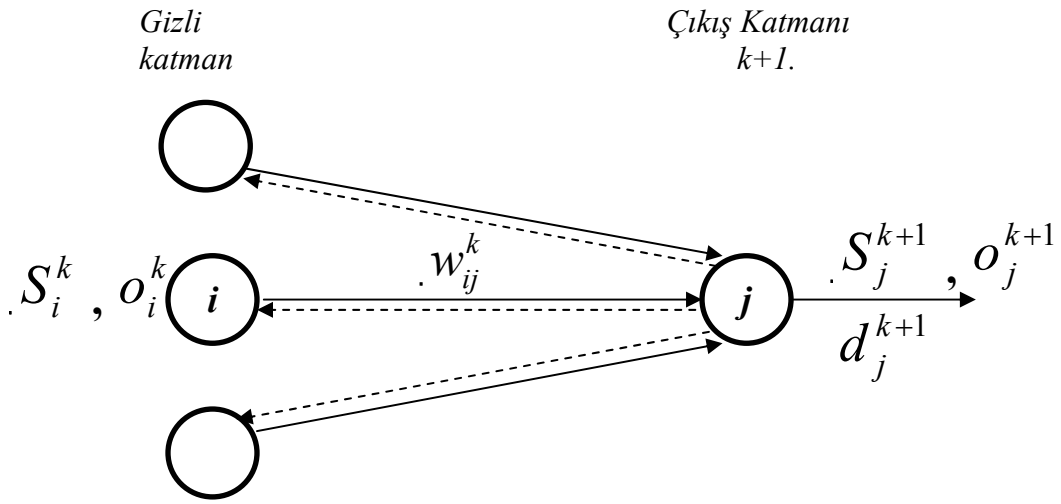
o_j^{k+1} = k+1. katmandaki j. nöronun transfer fonksiyonunu kullanarak, S_j^{k+1}

için ürettiği çıkış değeri.

Bu şekilde son gizli katmandan çıkış katmanındaki her bir nörona değerler ulaştığında, ağı ürettiği çıkış değerleri bulunmuş olur.

4.5. Hatanın Geri Yayılımı

Yapay sinir ağlarının parametrelerinin güncellenmesi için literatürde en çok kullanılan yöntem hatanın geriye yayılması yöntemidir. Ses tanıma problemlerinden nonlinear sistem tanımlama ve denetim problemlerine kadar çok değişik alanlarda başarı ile uygulanmıştır. (Efe ve Kaynak, 2000, s.15) bir diğer tercih sebebi de anlaşılması kolay ve matematiksel olarak ispatlanabilir olmasıdır. (http://batitrakya.dostweb.com/yapaysiniraglari/ek_dokuman.htm , 2004)



Şekil 4.5 Çıkış katmanında hatanın geriye yayılımı

Hatayı geri yayma metodu, gradient (eğim) düşürme yöntemini kullanarak toplam hataları minimize etmeyi hedefler. Toplam hatalara ilişkin bağıntı (4.7) aşağıda verilmiştir. Yöntem bu fonksiyonun minimum noktasını bulmak için (4.8) bağıntısından faydalanarak iteratif bir çözüm sunmaktadır

$$J_r = \frac{1}{2} \sum_{j=1}^{n_{k+1}} (d_j - i_j)^2 \quad (4.7)$$

$$\Delta w = -\eta \nabla_w J_r \quad (4.8)$$

Yukarıdaki bağıntılarda:

$d_j = j$. numaralı çıkış nöronun gerçek değeri.

η = Adım büyüklüğü.

Δ_w = Parametre güncelleme bağıntısı.

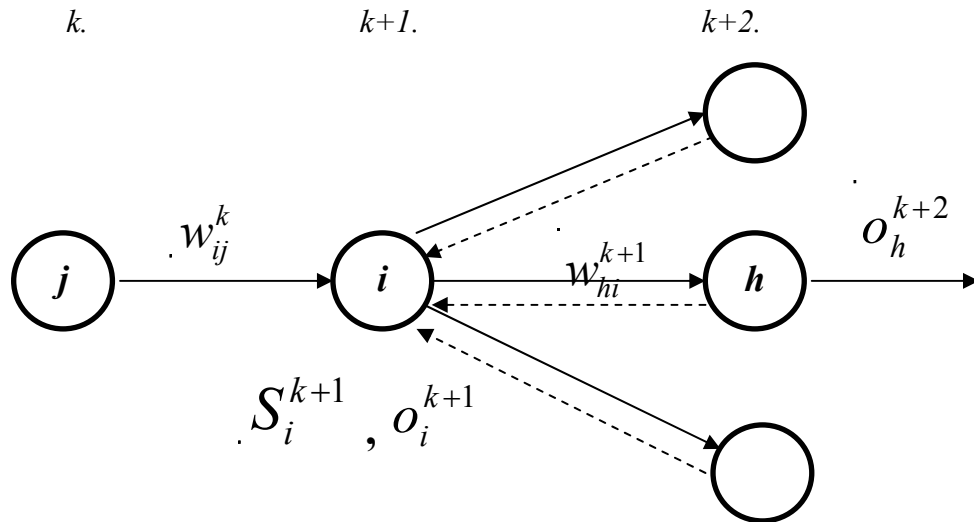
$\nabla_w = J_r$ fonksiyonunu w ağırlık katsayısına göre kısmi türevi.

(4.8) bağıntısındaki kısmi türev işlemi yapıldıktan sonra ortaya (4.9) bağıntısı çıkmaktadır. (4.9) bağıntısındaki değer (4.10) bağıntısında kullanılarak çıkış nöronları için yeni ağırlık değeri belirlenir.

$$\Delta w_{ji}^k = \eta (d_j - o_j^{k+1}) \Psi'(S_j^{k+1}) o_i^k \quad (4.9)$$

$$w_{ji}^{k(yeni)} = w_{ji}^{k(eski)} + \Delta w_{ji}^k \quad (4.10)$$

Gizli katmanlar için herhangi bir beklenen değer olmadığı için, parametre güncelleme değerini (4.11) bağıntısı kullanılarak hesaplanır.



Şekil 4.6 Gizli katmanlarda hatanın geriye yayılımı

$$\Delta w_{ji}^k = -\eta \left(\sum_{h=1}^{n_{k+2}} \frac{\partial J_r}{\partial S_h^{k+2}} w_{hi}^{k+1} \right) \Psi'(S_i^{k+1}) p_j^k \quad (4.11)$$

4.6. Uygulama

Bu çalışmalar, Matlab 6.5 Neural Network Toolbox kullanılarak yazılan bir arayüz kullanılarak gerçekleştirilmiştir. Arayüz yardımı ile, aşağıda belirtilen girişler yapılarak çalıştırılacak modellerin özellikleri girilir. Arayüz yazılımı EK:1’de verilmiştir. Girişlerin nasıl yapılacağı aşağıda açıklanmıştır.

Yapılacak ilk belirleme değişken sayısı (NumVar), yapılan deney sayısı (NumTrainSet) ve sonuçları test etmek için kullanılacak olan deney sayısıdır (NumCheckSet). Bizim deneyimiz için 3 değişken, 85 öğrenim seti ve 5 tanede test seti bulunmaktadır.

NumVar=3;	Değişken sayısı
NumTrainSet=85;	Deney seti sayısı
NumCheckSet=5;	Kontrol seti sayısı

Daha sonra kullanılacak olan değişkenlerin üst ve alt sınırları girilir. Bizim deneyimizde tel çapı için alt ve üst sınır 1,33 mm. ve 2,60 mm., akım için 600 Amper ve 2600 Amper, tel çekme hızı içinse 2 m/sn. ile 7,5 m/sn. olarak belirlenmiştir.

range=[1.33 2.6; 600 2600; 2 7.5];

Deney sonuçları ise bir önceki adımda alt ve üst sınırları verilen sırayla girişi yapılır. Her deney setinin sonuna deney sonucu da eklenir.

%1. Train Set	1. Öğrenme seti
TSET(1, 1)= 1.33;	1. set Tel çapı değeri
TSET(1, 2)= 600;	1. set Akım değeri
TSET(1, 3)= 2;	1. set Hız değeri

TVALUE(1)= 5.650614754;

1. set Kalay kalınlığı

%2. Train Set

2. Öğrenme seti

TSET(2, 1)= 1.33;

2. set Tel çapı değeri

TSET(2, 2)= 600;

2. set Akım değeri

TSET(2, 3)= 4;

2. set Hız değeri

TVALUE(2)= 3.319672131;

2. set Kalay kalınlığı

..

..

..

%85. Train Set

85. Öğrenme seti

TSET(85, 1)= 2.6;

85. set Tel çapı değeri

TSET(85, 2)= 2600;

85. set Akım değeri

TSET(85, 3)= 2.0;

85. set Hız değeri

TVALUE(85)= 14.453;

85. set Kalay kalınlığı

Öğrenme setlerinden sonra kontrol setlerinin girişi yapılır.

%1. Check Set

1. Kontrol seti

CSET(1, 1)= 2.2;

1. set Tel çapı değeri

CSET(1, 2)= 1800;

1. set Akım değeri

CSET(1, 3)= 3.0;

1. set Hız değeri

CVALUE(1)= 8.165;

1. set Kalay kalınlığı

..

..

..

%5. Check Set

5. Kontrol seti

CSET(5, 1)= 1.8;

5. set Tel çapı değeri

CSET(5, 2)= 2200;

5. set Akım değeri

CSET(5, 3)= 2.0;

5. set Hız değeri

CVALUE(5)= 17.322;

5. set Kalay kalınlığı

Son olarak Yapay Sinir Ağ'ında kaç adet gizli katman, her bir katmanda kaç nöron bulunacağı ve bu katmanlarda kullanılacak fonksiyon tipi seçilir. Her bir

değişik seçim önerilen değişik bir modeldir ve her biri için değişik sonuçlar elde edilmiştir. Örnek olarak aşağıda verilen girişte 3 gizli katman bulunmaktadır.

Layernumber=3;	Gizli katman sayısı
Layer1neuronnumber=7;	1. Katman nöron sayısı
Layer1transferfun='tansig';	1. Katman fonksiyon tipi
Layer2neuronnumber=7;	2. Katman nöron sayısı
Layer2transferfun='logsig';	2. Katman fonksiyon tipi
Layer3neuronnumber=3;	3. Katman nöron sayısı
Layer3transferfun='tansig';	3. Katman fonksiyon tipi

Bunlara ek olarak istenirse iterasyon sayısını ve adım büyüklüğünü değiştirebiliriz. Bu tezdeki uygulamalarda iterasyon sayısı başlangıç olarak 25.000, adım büyüklüğü ise 0.0005 alınmıştır.

net.trainParam.lr=0.0005;	Adım büyüklüğü
net.trainParam.epochs=25000;	İterasyon sayısı

Bütün bu ayarlamalar bittikten sonra arayüz çalıştırılır.

4.7. Sonuçlar

Deney, gizli katman sayıları, her katmandaki nöron sayısı ve fonksiyon tipleri rasgele değiştirilerek tekrarlanmıştır. Alınan sonuçlarda Root Mean Square (Rms) değerine yani hata kareleri toplamının karekökü değerine ve gerçek değer ile kullanılan model sonucu elde edilen değer arasında ki yüzde hata oranına bakılacaktır. Hata kareleri toplamının karekökü (HKTK), istatistiksel kontrol şemalarında kullanılan bir değerdir. Gerçek değerden ne kadar sapıldığını ifade eder. Değer ne kadar küçük olursa sapma o kadar az demektir.

Bu tezde hedef olarak, HKTK’de 0.2, maksimum hata oranında da % 10 seçilmiştir. Sonuçlar aşağıda verilmiştir.

Tablo 4.1 2 katmanlı, $\eta = 0,000010$ olan örnekler

Deney No.	İterasyon Sayısı	Nöron Sayısı	Nöron Fonk.	Öğrenim Seti		Kontrol Seti	
				HKTK	% Hata	HKTK	% Hata
1	50.000	3	tansig	5,2	%387	2,7	%102
		3	logsig				
2	200.000	100	logsig	0,9	%26	1,1	%23
		100	logsig				
3	100.000	50	logsig	2,5	%92	1,7	%34
		75	logsig				
4	50.000	7	logsig	5,1	%386	2,7	%113
		5	logsig				
5	100.000	75	logsig	1,3	%39	1,3	%27
		75	logsig				
6	200.000	100	tansig	5,1	%416	3,1	%114
		100	tansig				
7	200.000	100	logsig	0,7	%15	0,2	%9
		100	tansig				
8	200.000	100	tansig	4,8	%360	2,9	%119
		100	logsig				
9	75.000	75	logsig	2,2	%71	1,8	%36
		50	logsig				

Tablo 4.2 2 katmanlı, $\eta = 0,000500$ olan örnekler

Deney No.	İterasyon Sayısı	Nöron Sayısı	Nöron Fonk.	Öğrenim Seti		Kontrol Seti	
				HKTK	% Hata	HKTK	% Hata
10	2.500	3	tansig	5,3	%396	2,7	%106
		3	logsig				
11	2.500	3	logsig	5,3	%396	2,7	%106
		3	tansig				
12	2.500	7	logsig	5,3	%396	2,7	%106
		5	logsig				
13	2.500	7	tansig	5,3	%396	2,7	%106
		10	tansig				

Tablo 4.3 3 katmanlı, $\eta = 0,000010$ olan örnekler

Deney No.	İterasyon Sayısı	Nöron Sayısı	Nöron Fonk.	Öğrenim Seti		Kontrol Seti	
				HKTK	% Hata	HKTK	% Hata
14	100.000	100	tansig	5,0	%429	2,8	%119
		100	tansig				
		100	tansig				
15	50.000	100	logsig	0,8	%21	0,4	%7
		100	logsig				
		100	logsig				
16	50.000	50	logsig	2,3	%75	1,7	%45
		50	logsig				
		50	logsig				
17	100.000	3	logsig	5,2	%395	2,7	%105
		3	logsig				
		3	logsig				
18	100.000	100	logsig	0,6	%18	0,3	%6
		100	logsig				
		100	logsig				
19	200.000	100	logsig	0,5	%16	0,6	%13
		100	logsig				
		100	logsig				
20	50.000	5	tansig	5,2	%387	2,7	%102
		7	tansig				
		3	logsig				

Tablo 4.4 3 katmanlı, $\eta = 0,000500$ olan örnekler

Deney No.	İterasyon Sayısı	Nöron Sayısı	Nöron Fonk.	Öğrenim Seti		Kontrol Seti	
				HKTK	% Hata	HKTK	% Hata
21	2.500	30	tansig	5,3	%396	2,7	%106
		30	tansig				
		30	tansig				
22	2.500	7	tansig	5,3	%396	2,7	%106
		5	logsig				
		5	tansig				
23	2.500	5	tansig	5,3	%396	2,7	%106
		7	tansig				
		3	logsig				
24	2.500	10	tansig	5,3	%396	2,7	%106
		9	tansig				
		12	tansig				
25	2.500	3	logsig	5,3	%395	2,7	%105
		3	logsig				
		3	logsig				
26	2.500	30	logsig	5,3	%396	2,7	%106
		30	logsig				
		30	logsig				

Kullanılan modellerden 7 ve 19 numaralı olanlar, sırasıyla % 17'lik ve % 16'lık hatayla en iyi sonuçları vermiştir.

5. CEVAP YÜZEYİ (RESPONSE SURFACE) YÖNTEMİ

5.1. Cevap Yüzeyi Tarihçesi

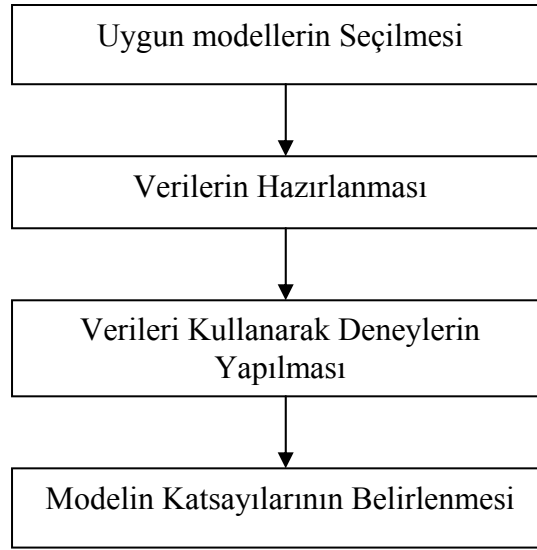
Bu tezde kullanılacak olan Cevap Yüzeyi (Response Surface) yöntemi, ilk olarak Box ve Draper (Box and Draper, 1987, 15) tarafından fiziksel deney sonuçlarına model uydurmak için tasarlanmıştır. 1988 – 1990 yıllarında Taguchi ve Myers & Montgomery Deney Dizayn etme (Design of Experiments) metodu ile birlikte kullanmışlardır. 1992-1994 yılları arasında ise yeni bir alanda, kalite kontrol ve kalite geliştirmede kullanılmıştır. 1995 yılından sonra nümerik analiz kullanılan yöntemlerinin optimize edilmesi için kullanılmıştır. (http://www-personal.engin.umich.edu/~kikuchi/Research/rsm_amago.pdf, 2004)

5.2. Cevap Yüzeyi Tanımı

Matematiksel ve istatistiksel tekniklerin beraber çalışması ile ortaya çıkan ve bağımsız değişkenlerin bir bağımlı değişkeni (çıktıyı) etkilediği problemlerde, çıktıyı optimize etmek üzere kullanılan bir tekniktir. (Akhun, 2001, s2)

Birçok Cevap Yüzeyi probleminde çıktı ve bağımsız değişkenlerin arasındaki bağıntı bilinmez. Bu yüzden yapılacak ilk işlem uygun bir model bulmaktır. Kullanılacak deneysel veriler ile ilk olarak lineer model uygulanır. Daha sonra en uygun model bulunana kadar polinomun derecesi artırılır.

Cevap yüzeyi yönteminin yapısı basit olarak aşağıda ifade edilmiştir.



Şekil 5.1 Cevap Yüzeyi yönteminde kullanılan uygulama adımları

5.3. Metodoloji

Genel olarak, tahmin fonksiyonu $\tilde{y}_f^k(b)$, dönüştürücü bir parametre ile çarpılmış temel fonksiyonların toplamı olarak ifade edilebilir.

$$\tilde{y}(b) = \sum_{i=1}^L a_i \phi_i(b) \quad (5.1)$$

Formülde L temel fonksiyon sayısı $\phi = [\phi_1, \phi_2, \dots, \phi_L]$, a ise dönüştürücü parametreleri $a = [a_1, a_2, \dots, a_L]$ ifade etmektedir.

Eğer dikkatlice seçilmiş ve dorulularından emin olunan deneysel veriler kullanılırsa (deneylerin değişken sayısından fazla olması gerekmektedir), dönüştürücü parametre a en küçük kareler yöntemi kullanılarak tahmin edilebilir.

$$E(a) = \sum_{i=1}^p [y(b_i) - \tilde{y}(b_i)]^2 \rightarrow \min \quad (5.2)$$

(5.2) eşitliğinde b_i deneysel verileri, p deney sayısını, $\tilde{y}$ tahmin fonksiyonunu ve E ise hata toplamalarını ifade etmektedir. E 'nin minimizasyonu ile elde edilecek olan a parametresi, en uygun sonucun alınmasını sağlayacaktır. Hatayı minimize edecek olan a parametresi (5.3) eşitliği ile hesaplanır.

$$a = (X^T X)^{-1} X^T y \quad (5.3)$$

Yukarıda verilen X matrisi aşağıdaki şekilde ifade edilmektedir.

$$X = \begin{bmatrix} 1 & b_{11} & \cdots & b_{1L} \\ 1 & b_{21} & \cdots & b_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & b_{p1} & \cdots & b_{pL} \end{bmatrix}_{p \times L} \quad (5.4)$$

5.4. Uygulama

Bu metodu uygulamak için yine MATLAB yazılımı kullanılmıştır. Fakat bu sefer bir arayüz değil, direk MATLAB yazılımının C benzeri programlama dili kullanılarak bir program yazılmıştır. Bu programında da Yapay Sinir Ağları'nda olduğu gibi bazı ön bilgi girişlerinin yapılması gerekmektedir. İlk olarak yine Yapay Sinir Ağları'ndaki gibi değişken sayısı (NumVar), yapılan deney sayısı (NumTrainSet) ve sonuçları test etmek için kullanılacak olan deney sayısı (NumCheckSet) girişleri yapılır.

NumVar=3;	Değişken sayısı
NumTrainSet=85;	Deney seti sayısı
NumCheckSet=5;	Kontrol seti sayısı

Daha sonra sıra programda kullanılacak olan fonksiyonun seçimine gelir. Elimizde bulunan deney sayısından dolayı en fazla 6. dereceden fonksiyonla

çalışabilmektedir. Aşağıdaki şekilde girişi yapılan fonksiyon çözüm aşamasında kullanılacaktır.

ModelOrder=1 : "Linear",	Lineer fonksiyon
ModelOrder=2 : "Parabolic",	Parabolik fonksiyon
ModelOrder=3 : "Cubic",	Kübik fonksiyon
ModelOrder=4 : "4th order",	4. dereceden fonksiyon
ModelOrder=5 : "5th order",	5. dereceden fonksiyon
ModelOrder=6 : "6th order"	6. dereceden fonksiyon
ModelOrder=7 : "7th order",	7. dereceden fonksiyon
ModelOrder=8 : "8th order",	8. dereceden fonksiyon
ModelOrder=9 : "9th order",	9. dereceden fonksiyon
ModelOrder=10 : "10th order"	10. dereceden fonksiyon

Bunlara ek olarak kullanılan değişkenlerin sonuçla aralarında doğru veya ters orantı olduğu, kullanılacak olan modelin tipi ve backward pres eliminasyonunun kullanılıp kullanılmayacağı bilgileri girilir. Bu girişler aşağıda tanımlanmıştır.

ReciprocalFlag=[1 0 1];..... Reciprocal Flag (0=No Reciprocal, 1=Yes Reciprocal)

Burada her bir değişken için ayrı tanımlama yapılır. Tanımlama yapılırken kullanılan “0” in anlamı doğru orantı, “1” in anlamı ters orantıdır.

ModelType=1;.....ModelType=1 (Polynomial), ModelType=2 (Logarithmic)

Sırasıyla her denemede iki model tipide denenmiştir. Bu yüzden her bir değişik fonksiyondan iki sonuç elde edilmiştir.

Backpress=0;.....1=Use Backward Press elimination, 0=Do not use BPRESS

Tüm ayarlamalar yapıldıktan sonra uygulamaya geçilir.

5.5. Sonuçlar

Lineer denklemden 6. dereceden denkleme kadar her denklem, hem polinom hem de logaritmik tipler için denenmiştir. Alınan sonuçlarda Rms fit (Root Mean Square) değerine ve gerçek değer ile kullanılan model sonucu elde edilen değer arasında ki yüzde hata oranına bakılacaktır. Sonuçlar aşağıda verilmiştir.

Tablo 5.1 Lineer model kullanıldığında elde edilen sonuçlar.

Orantı Durumları			LOGARİTMİK				POLİNOM			
			Öğrenim Seti		Kontrol Seti		Öğrenim Seti		Kontrol Seti	
			HKTK	% Hata	HKTK	% Hata	HKTK	% Hata	HKTK	% Hata
D.O.	D.O.	D.O.	7,4	%650	0,2	%6	2,0	%269	1,6	%51
T.O.	D.O.	D.O.	7,4	%650	0,2	%6	2,0	%276	1,6	%53
D.O.	T.O.	D.O.	7,4	%650	0,2	%6	2,5	%339	1,9	%66
D.O.	D.O.	T.O.	7,4	%650	0,2	%6	1,7	%199	1,9	%74
T.O.	T.O.	D.O.	7,4	%650	0,2	%6	2,5	%345	1,9	%61
T.O.	D.O.	T.O.	7,4	%650	0,2	%6	1,7	%206	2,0	%74
D.O.	T.O.	T.O.	7,4	%650	0,2	%6	2,2	%269	1,8	%54
T.O.	T.O.	T.O.	7,4	%650	0,2	%6	2,2	%276	2,0	%54

Tablo 5.2 Parabolik model kullanıldığında elde edilen sonuçlar.

Orantı Durumları			LOGARİTMİK				POLİNOM			
			Öğrenim Seti		Kontrol Seti		Öğrenim Seti		Kontrol Seti	
			HKTK	% Hata	HKTK	% Hata	HKTK	% Hata	HKTK	% Hata
D.O.	D.O.	D.O.	7,6	%697	0,2	%4	0,7	%89	0,7	%20
T.O.	D.O.	D.O.	7,6	%697	0,2	%4	0,7	%91	0,6	%19
D.O.	T.O.	D.O.	7,6	%697	0,2	%4	1,1	%149	0,7	%19
D.O.	D.O.	T.O.	7,6	%697	0,2	%4	0,3	%27	0,2	%8
T.O.	T.O.	D.O.	7,6	%697	0,2	%4	1,1	%152	0,7	%20
T.O.	D.O.	T.O.	7,6	%697	0,2	%4	0,4	%36	0,3	%10
D.O.	T.O.	T.O.	7,6	%697	0,2	%4	0,9	%76	0,5	%18
T.O.	T.O.	T.O.	7,6	%697	0,2	%4	0,9	%81	0,5	%15

Tablo 5.3 Kübik model kullanıldığında elde edilen sonuçlar.

Orantı Durumları			LOGARİTMİK				POLİNOM			
			Öğrenim Seti		Kontrol Seti		Öğrenim Seti		Kontrol Seti	
			HKTK	% Hata	HKTK	% Hata	HKTK	% Hata	HKTK	% Hata
D.O.	D.O.	D.O.	7,5	%701	0,1	%3	0,3	%23	0,2	%6
T.O.	D.O.	D.O.	7,5	%701	0,1	%3	0,3	%26	0,2	%6
D.O.	T.O.	D.O.	7,5	%701	0,1	%3	0,4	%41	0,3	%12
D.O.	D.O.	T.O.	7,5	%701	0,1	%3	0,3	%9	0,1	%3
T.O.	T.O.	D.O.	7,5	%701	0,1	%3	0,5	%38	0,3	%11
T.O.	D.O.	T.O.	7,5	%701	0,1	%3	0,3	%9	0,1	%3
D.O.	T.O.	T.O.	7,5	%701	0,1	%3	0,4	%26	0,1	%4
T.O.	T.O.	T.O.	7,5	%701	0,1	%3	0,4	%28	0,2	%5

Tablo 5.4 4. dereceden model kullanıldığında elde edilen sonuçlar.

Orantı Durumları			LOGARİTMİK				POLİNOM			
			Öğrenim Seti		Kontrol Seti		Öğrenim Seti		Kontrol Seti	
			HKTK	% Hata	HKTK	% Hata	HKTK	% Hata	HKTK	% Hata
D.O.	D.O.	D.O.	7,5	%689	0,1	%3	0,2	%10	0,1	%4
T.O.	D.O.	D.O.	7,5	%689	0,1	%3	0,2	%9	0,1	%4
D.O.	T.O.	D.O.	7,5	%689	0,1	%3	0,3	%12	0,2	%9
D.O.	D.O.	T.O.	7,5	%689	0,1	%3	0,2	%10	0,2	%4
T.O.	T.O.	D.O.	7,5	%689	0,1	%3	0,3	%13	0,2	%9
T.O.	D.O.	T.O.	7,5	%689	0,1	%3	0,2	%9	0,2	%4
D.O.	T.O.	T.O.	7,5	%689	0,1	%3	0,3	%10	0,1	%7
T.O.	T.O.	T.O.	7,5	%689	0,1	%3	0,3	%11	0,2	%7

Tablo 5.5 5. dereceden model kullanıldığında elde edilen sonuçlar.

Orantı Durumları			LOGARİTMİK				POLİNOM			
			Öğrenim Seti		Kontrol Seti		Öğrenim Seti		Kontrol Seti	
			HKTK	% Hata	HKTK	% Hata	HKTK	% Hata	HKTK	% Hata
D.O.	D.O.	D.O.	7,5	%677	0,1	%3	0,2	%9	0,2	%5
T.O.	D.O.	D.O.	7,5	%677	0,1	%3	0,2	%8	0,2	%5
D.O.	T.O.	D.O.	7,5	%677	0,1	%3	0,2	%9	0,1	%3
D.O.	D.O.	T.O.	7,5	%677	0,1	%3	0,2	%8	0,1	%3
T.O.	T.O.	D.O.	7,5	%677	0,1	%3	0,2	%10	0,1	%2
T.O.	D.O.	T.O.	7,5	%677	0,1	%3	0,2	%9	0,1	%2
D.O.	T.O.	T.O.	7,5	%677	0,1	%3	0,2	%9	0,1	%2
T.O.	T.O.	T.O.	7,5	%677	0,1	%3	0,2	%9	0,1	%3

Tablo 5.6 6. dereceden model kullanıldığında elde edilen sonuçlar.

Orantı Durumları			LOGARİTMİK				POLİNOM			
			Öğrenim Seti		Kontrol Seti		Öğrenim Seti		Kontrol Seti	
			HKTK	% Hata	HKTK	% Hata	HKTK	% Hata	HKTK	% Hata
D.O.	D.O.	D.O.	7,6	%693	0,2	%3	0,2	%5,1	0,1	%2
T.O.	D.O.	D.O.	7,6	%693	0,2	%3	0,2	%5,1	0,1	%1
D.O.	T.O.	D.O.	7,6	%693	0,2	%3	0,2	%5,9	0,2	%5
D.O.	D.O.	T.O.	7,6	%693	0,2	%3	0,2	%5,4	0,1	%4
T.O.	T.O.	D.O.	7,6	%693	0,2	%3	0,2	%5,0	0,2	%5
T.O.	D.O.	T.O.	7,6	%693	0,2	%3	0,1	%4,9	0,1	%4
D.O.	T.O.	T.O.	7,6	%693	0,2	%3	0,2	%6,1	0,1	%4
T.O.	T.O.	T.O.	7,6	%693	0,2	%3	0,2	%5,9	0,1	%3

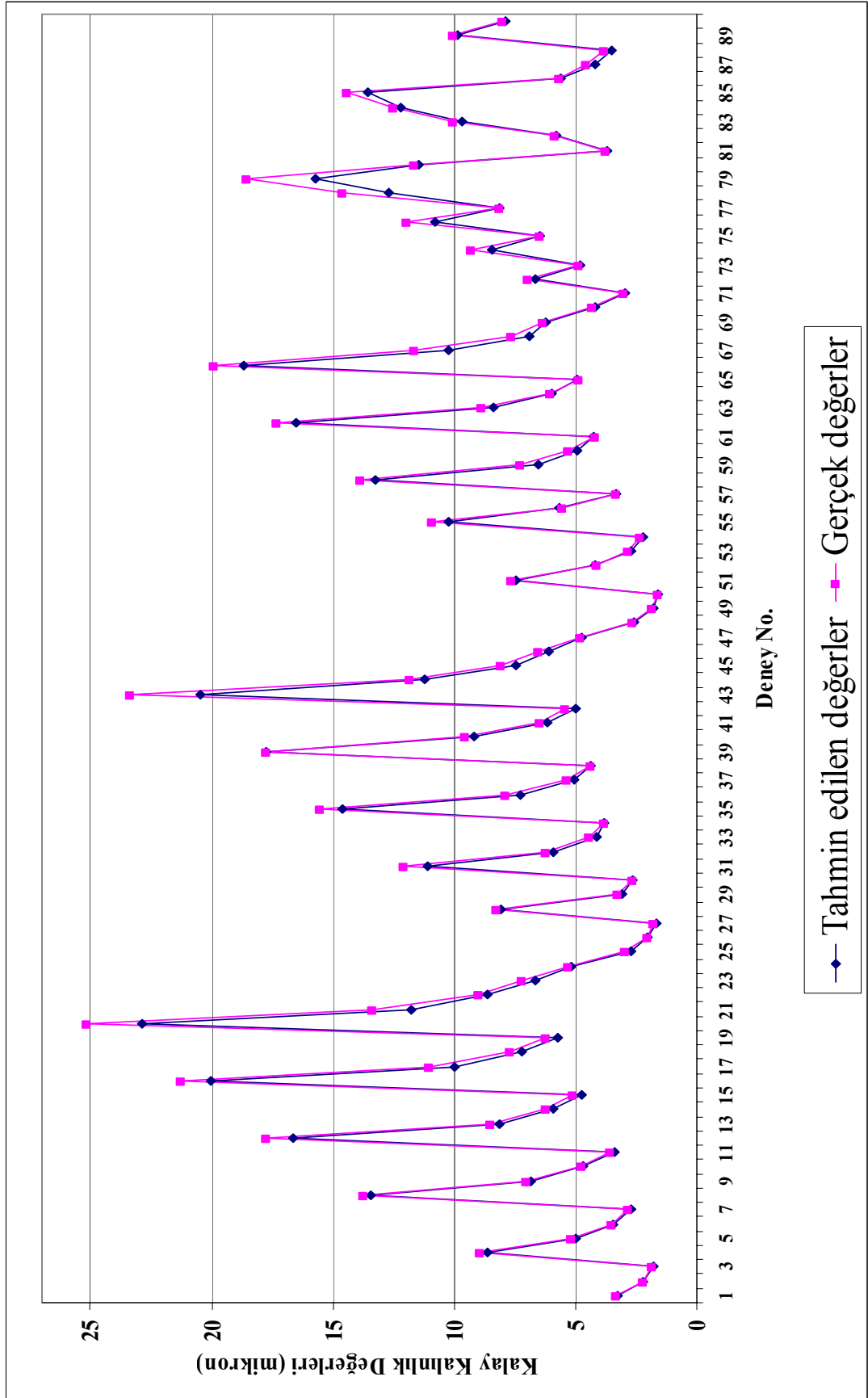
6. dereceden polinom tipli model de, birinci ve üçüncü değişkenler ters, ikinci değişken doğru orantılı olduğunda % 4,9'luk bir yanılma ile başarılı bir sonuç elde edilmiştir.

6. SONUÇ VE ÖNERİLER

Bu çalışmada, kalaylı tel üretimi süreci ön hazırlık değerlerinin tahmini için matematiksel bir model oluşturma hedeflenmiştir. Bunun için iki metot kullanılmıştır. Yapay Sinir Ağları ve Cevap Yüzeyi metotları.

Bu tezde İleri Sürümlü ve Hatayı Geri Yaymalı Yapay Sinir Ağı kullanılmıştır. Metot da kullanılan gizli katman sayısı, bu katmanlardaki nöron sayıları ve nöronlarda kullanılan transfer fonksiyonu için herhangi bir yöntem olmadığı için rasgele seçilmiştir. Kullanılan arayüz en fazla 3 gizli katman kullanılmasına izin vermektedir. Bu sınır dahilinde tüm sistemi ifade edeceği düşünülen, 2 ve 3 adet gizli katmanlı, değişik nöron sayıları ve transfer fonksiyonları kullanarak deneyler yapılmıştır.

Elde edilen sonuçlar neticesinde Yapay Sinir Ağları metodunun bu problem ile istenilen derecede başarılı sonuç veremediği anlaşılmıştır. Yapılan ilk denemelerde transfer fonksiyonları rasgele değiştirilerek, nöron sayısı 3 ile 10 adet arasında sayılarda kullanılmış fakat alınan sonuçlar istenilen değerlerden çok uzak çıkmıştır. Bunun üzerine başlangıçta 2.500 olarak belirlenen iterasyon sayısı 50.000'e çıkarılmış fakat yine bir gelişme sağlanamamıştır. Bu modellerde alınan sonuçların tümünde çok benzer sonuçlara gidildiği için yakınsama hatası olduğu varsayılarak başlangıçta 0,0005 olarak belirlenen iterasyon adım büyüklüğü 1/50 oranında küçültülerek 0,00001 değerine değiştirilmiştir. Yine istenilen değerlere ulaşamayınca katmanlardaki nöron sayısı arttırılmıştır. Bu değişiklik hesap maliyetini arttırdığı için tercih edilmemekle beraber, yapılan değişim olumlu sonuçlar getirmiştir. Nöron sayısının arttırılması ile birlikte, yeni adım büyüklüğü ve arttırılmış iterasyon sayısı ile yapılan denemelerde başlangıç denemelerine göre daha iyi sonuçlar elde edilmiştir. Bu sonuçlar ışığında denemelere iterasyon sayısı arttırılarak devam edilmiştir. Sırayla 75.000, 100.000 ve 200.000 olarak arttırılan iterasyon sayıları 0,5 ile 1,5 saat süren zamanlarda tamamlanmışlardır.



Tablo 6.1Yapay Sinir Ağları metodunun tahmin değerlerinin gerçek değerler ile karşılaştırılması.

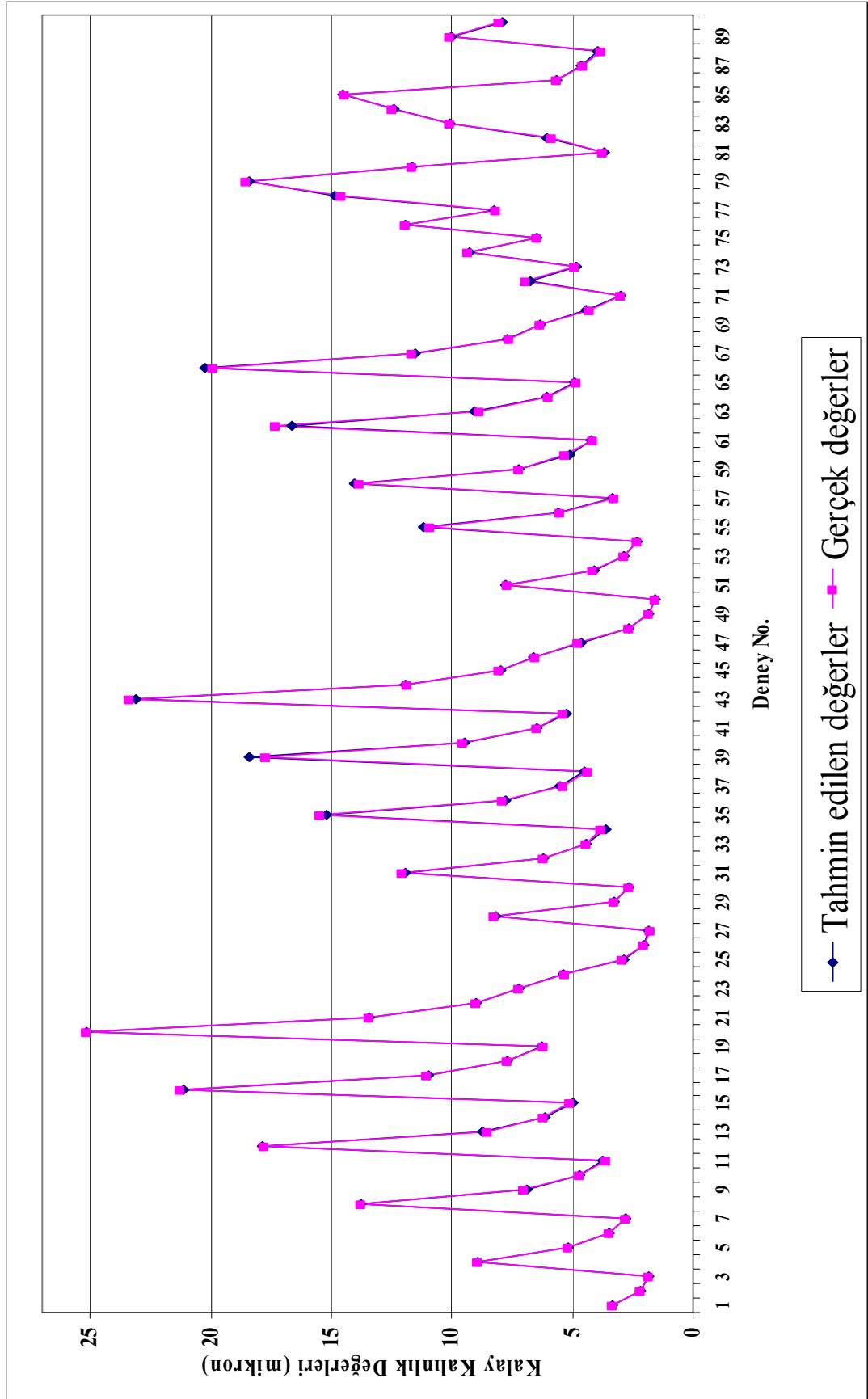
Tablo (6.1)'de denemeler sonucunda en iyi sonucu veren, 2 gizli katmanlı, her bir katmanında 100 nöron olan, ilk katmanı logaritmik ikinci katmanı tanjant sigmoid olan modelin 200.000 iterasyon çalıştırılması sonucunda elde edilen sonuçların gerçek değerler ile karşılaştırılması gösterilmiştir.

Yapay Sinir Ağları yöntemi'nin daha iyi sonuçlar vermesi için yapılabilecek ilk işlem deney sayısını arttırılmasıdır . Fakat kalay kaplama sürecinin anlatıldığı bölümde bahsettiğimiz, miktarları sabitlenen kullanılan emülsiyonun asit ve katkı değerlerinin zamanla değişmesi, aynı şartlarda örnek alımını zorlaştırmaktadır. Gerekli şartlar sağlandığında daha fazla örnek ile bu yöntem tekrar denenir ve iterasyon sayısı da arttırılırsa daha iyi sonuçlar elde etme şansı vardır.

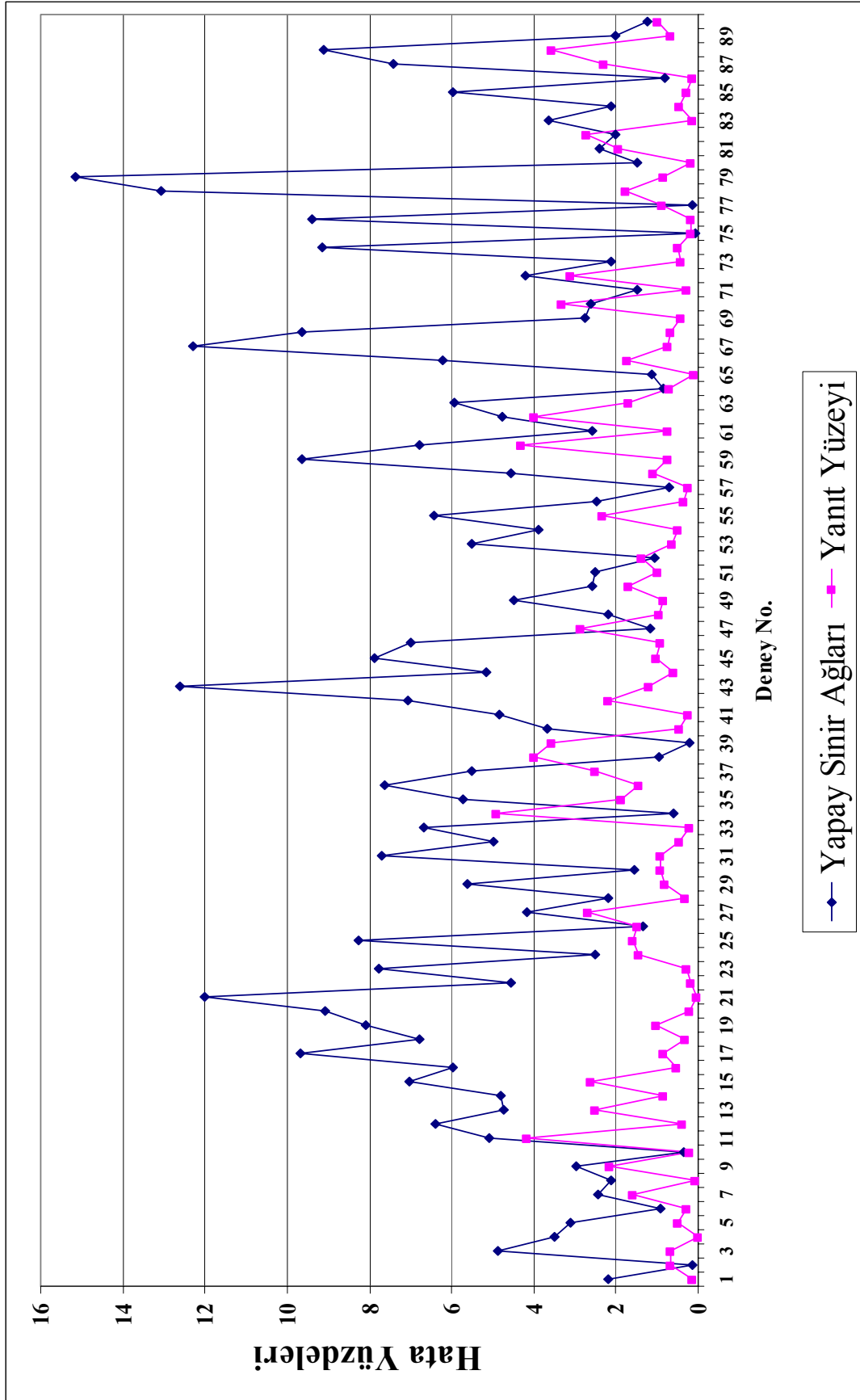
Kullanılan diğer metot Cevap Yüzeyi'dir. Sırayla lineer yani 1. derece denklemlerden başlayarak, parabolik, kübik, 4. dereceden, 5. dereceden ve 6. dereceden denklemler ile çalışılmıştır. Bir önceki paragrafta bahsettiğimiz nedenlerle elimizdeki deney sayısının arttırılamaması nedeni ile daha yüksek dereceli denklemler denenememiştir. 6. dereceden polinom ile birinci ve üçüncü değişken ters, ikinci değişken doğru orantılı olarak alınan model kullanıldığında elde edilen maksimum sapma % 4.9 olarak elde edilmiştir. Bu değer bizim kabul koşullarımız olan % 10'luk hata değerini sağlamaktadır. Logaritmik model hiçbir deneyde başarı sağlayamamıştır. Kübik polinom modellerinin bazılarında istenilen % 10'luk hata değerine ulaşılmıştır. Ama polinom derecesinin arttırılması ile sonuçların giderek iyileştiği görüldüğü için denemeler, deney sayısı eksikliği ortaya çıkana kadar durdurulmamıştır.

Fakat alınan sonuçlardan görülmektedir ki kullandığımız modelin derecesini her arttırdığımızda daha iyi sonuçlar elde edilmektedir. Bu yüzden, kullanılan modelin daha başarılı sonuçlar elde etmesi için kullanılan deney sayısı arttırılmalı ve deneyler tekrarlanmalıdır. Ayrıca bu çalışmada değişken olarak kullanılmayan elektroliz suyu sıcaklığı, asit miktarı ve katkı malzemeleri gibi etkenlerinde katılımıyla üretim sürecine çok daha adaptif bir model elde etme şansı vardır.

Tablo (6.2)'de seçilen Cevap Yüzeyi modelinin sonuçlarının gerçek sonuçlar ile karşılaştırılması verilmiştir.



Tablo 6.2 Cevap Yüzeyi metodunun tahmin değerlerinin gerçek değerler ile karşılaştırılması.



Tablo 6.3 Cevap Yüzeyi ve Yapay Sinir Ağları metotlarının hata yüzdeleri karşılaştırması.

Kullanılan metotlardan Cevap Yüzeyi'nde işlem süresinin saniyeler oluşu ve amacımızın maliyet ve zamandan tasarruf olduğunu göz önünde bulundurularak, beraberinde yüksek hesap maliyetini de getiren Yapay Sinir Ağları metodu ile çalışmalara devam edilmesi gereksiz gözükmektedir. Yüksek teknoloji bilgisayar ihtiyacı ve çok uzun süren işlem süresi, Yapay Sinir Ağları metodu için ek dezavantajlar oluşturmaktadır. 100 nöronlu, 2 katmanlı bir model 200.000 iterasyon çalıştırıldığında işlem Pentium 4 - 2.6 gh ve 512 mb ram'li bir bilgisayarda yaklaşık 1,5 saat sürmüştür. Bu sonuçlar ışığında Kalay Kaplama üretim sürecinde üretim ön hazırlık koşullarının tahmini için, Cevap Yüzeyi metodunda altıncı dereceden polinom kullanılarak oluşturulan model başarılı olmuştur.

Cevap Yüzeyi metodunun bu üretim sürecine uygunluğu göz önüne alınarak piyasada kullanılan diğer kaplama süreçleriyle de çalışılması uygun olacaktır. Örneğin maliyeti kalay kaplama işleminden çok daha fazla tutan nikel, gümüş ve ya altın kaplama işlemleri içinde bu tarz bir model geliştirme, bu tarz üretim yapan firmalara faydalı olacaktır.

KAYNAKLAR

- ÇAKIR, A. F., 1984: Metalik Kaplamalar, SEGEM, Bursa
- ÖZTEMEL, E., 2003: Yapay Sinir Ağları, Papatya Yayıncılık, İSTANBUL
- <http://science.ankara.edu.tr/~ozbek/model.htm> , 2004
- <http://science.ankara.edu.tr/~ozbek/gulcan.htm> , 2004
- BEZİR ve ark. ŞENCAN, ÖZEK, YAKUT, 2003: Güneş Enerjili Sistemlerde Yapay Sinir Ağları Metodunun Kullanılabilirliği ve Bir Uygulama, S.D.Ü. Fen Bilimleri Enstitüsü Dergisi. V 7.1, S. 47-51
- ELMAS, Ç., 2003: Yapay Sinir Ağları, Seçkin Yayıncılık, ANKARA
- http://batitrakya.dostweb.com/yapaysiniraglari/ek_dokuman.htm , 2004
- EFE, Ö., KAYNAK, O., 2000: Yapay Sinir Ağları ve Uygulamaları, Boğaziçi Üniversitesi, İSTANBUL
- KARADENİZ, İ., 2001: Sayısal Analiz ve Mühendislik Uygulamaları, Uludağ Üniversitesi, BURSA
- http://www-personal.engin.umich.edu/~kikuchi/Research/rsm_amago.pdf, 2004
- AKHUN, M., 2001: Çifte Kaynak Kısıtlı Grup Teknolojisi Üretim Sistemlerinin Bozucu Faktörlere Dayanıklı Tasarımı, İTÜ Sosyal Bilimler Enstitüsü
- BOX, G.E.P., DARPER, N. R., 1987: Empirical Model-Building and Response Surfaces, John Wiley and Sons Inc., NY, USA

ÖZGEÇMİŞ

Okan GÜRBÜZ, 1976 yılında İstanbul’da doğdu. İlk ve orta öğretimini 1982-1993 yılları arasında İstanbul’da tamamladı. 1994 yılında girmeye hak kazandığı Kocaeli Üniversitesi Mühendislik Fakültesi Endüstri Mühendisliği Bölümü’nden 1999 yılında mezun oldu. 2000 yılında Gebze Yüksek Teknoloji Enstitüsü Mühendislik Fakültesi Tasarım ve İmalat Mühendisliği Bölümü’nde yüksek lisans öğrenimine başladı.

EK 1 : YAPAY SİNİR AĞLARI MATLAB ARAYÜZÜ

Aşağıda en iyi sonuç alınan 2 gizli katmanlı, her katmanında 100 nöron bulunan, ilk katman transfer fonksiyonu logaritmik, ikinci katman değeri tanjant olan modelin örnek teşkil etmesi için ilk beş deney setinin girişi yapılmış MATLAB arayüz yazılımı verilmiştir.

```
%Program: APPROXIMATE MODEL GENERATION USING ARTIFICIAL
NEURAL NETWORK
```

```
%
```

```
=====
=====
```

```
%
```

```
=====
=====
```

```
%          USER INPUT: PART-1
```

```
%
```

```
=====
=====
```

```
% Enter Number of Variables, Train Set Data and Check Set Data
```

```
NumVar=3;
```

```
NumTrainSet=85;
```

```
NumCheckSet=5;
```

```
%
```

```
=====
=====
```

```
% Memory allocation (Do not change anything below!!!)
```

```
TSET=zeros(NumTrainSet,NumVar);
```

```
TVALUE=zeros(NumTrainSet,1);
```

```
Tapprox=zeros(NumTrainSet,1);
```

```

Error=zeros(NumTrainSet,1);
Tdeviate=zeros(NumTrainSet,1);
CSET=zeros(NumCheckSet,NumVar);
CVALUE=zeros(NumCheckSet,1);
Capprox=zeros(NumCheckSet,1);
Cerror=zeros(NumCheckSet,1);
Cdeviate=zeros(NumCheckSet,1);
X=zeros(NumVar,1);
range=zeros(NumVar,2);

%
=====

=====

%          USER INPUT:  PART-2
%
=====

=====

% Enter Variable Ranges by specifying Lower bound (lb) and Upper bound (ub)
% range=[lb(1) ub(1); lb(2) ub(2); ... ;lb(n) ub(n)]

range=[1.33 2.6; 600 2600; 2 7.5];

%
=====

% Enter TRAIN SETS: TSET(NumTrainSet,NumVar)
%
=====

%1. Train Set
TSET(1, 1)= 1.33;
TSET(1, 2)= 600;
TSET(1, 3)= 4;
TVALUE(1)=3.319672131;

%2. Train Set

```

```

TSET(2, 1)= 1.33;
TSET(2, 2)= 600;
TSET(2, 3)= 6;
TVALUE(2)=2.197745902;

```

```
%3. Train Set
```

```

TSET(3, 1)= 1.33;
TSET(3, 2)= 600;
TSET(3, 3)= 7.5;
TVALUE(3)=1.862704918;

```

```
%4. Train Set
```

```

TSET(4, 1)= 1.33;
TSET(4, 2)= 1000;
TSET(4, 3)= 2;
TVALUE(4)=8.93442623;

```

```
%5. Train Set
```

```

TSET(5, 1)= 1.33;
TSET(5, 2)= 1000;
TSET(5, 3)= 4;
TVALUE(5)=5.174180328;

```

```
..
```

```
..
```

```
..
```

```
% *****
```

```
% *****
```

```
% Check Set
```

```
%1. Check Set
```

```

CSET(1, 1)= 1.33;
CSET(1, 2)= 600;
CSET(1, 3)= 2;
CVALUE(1)=5.650614754;

```

%2. Check Set

CSET(2, 1)= 1.60;

CSET(2, 2)= 1000;

CSET(2, 3)= 4.0;

CVALUE(2)=4.559;

%3. Check Set

CSET(3, 1)= 1.8;

CSET(3, 2)= 1400;

CSET(3, 3)= 6.0;

CVALUE(3)=3.835;

%4. Check Set

CSET(4, 1)= 2.2;

CSET(4, 2)= 2200;

CSET(4, 3)= 3.0;

CVALUE(4)=10.073;

%5. Check Set

CSET(5, 1)= 2.6;

CSET(5, 2)= 1400;

CSET(5, 3)= 2.0;

CVALUE(5)=8.019;

%

=====

=====

% USER INPUT: PART-3

%

=====

=====

```
Layernumber=2;
```

```
Layer1neuronnumber=100;
```

```
Layer1transferfun='logsig';
```

```
Layer2neuronnumber=100;
```

```
Layer2transferfun='tansig';
```

```
%
```

```
=====
```

```
=====
```

```
%                END OF USER INPUTS
```

```
%
```

```
=====
```

```
=====
```

```
% Print Inputs on the Screen before Neural Network Training
```

```
TSET
```

```
TVALUE
```

```
CSET
```

```
CVALUE
```

```
range
```

```
switch Layernumber
```

```
case 1,
```

```
    net=newff([range],[Layer1neuronnumber,1],...
```

```
        {Layer1transferfun,'purelin'},'traingd');
```

```
case 2,
```

```
    net=newff([range],[Layer1neuronnumber,Layer2neuronnumber,1],...
```

```
        {Layer1transferfun,Layer2transferfun,'purelin'},'traingd');
```

```
case 3,
```

```

net=newff([range],[Layer1neuronnumber,Layer2neuronnumber,...
Layer3neuronnumber,1],{Layer1transferfun,Layer2transferfun,...
Layer3transferfun,'purelin'},'traingd');
otherwise,
error('Hidden Layer Numbers are > 3 !!!');
end

```

```

net.trainParam.show=5000;
net.trainParam.lr=0.00001;
net.trainParam.epochs=200000;
net.trainParam.goal=1e-1;
net=train(net,TSET',TVALUE');

```

```

TrainRMSfit=0.0;
for i=1:NumTrainSet
    for j=1:NumVar
        X(j)=TSET(i,j);
    end
    Tapprox(i)=sim(net,X);
    Tdeviate(i)=abs(Tapprox(i)-TVALUE(i));
    Terror(i)=abs(Tdeviate(i)/TVALUE(i));
    TrainRMSfit=TrainRMSfit+Tdeviate(i)^2;
end
[Tmaxdeviate,maxTdeviateIndex]=max(Tdeviate);
[Terror,maxTerrorIndex]=max(Terror);
TrainRMSfit=sqrt(TrainRMSfit/NumTrainSet);

```

```

CheckRMSfit=0.0;
for i=1:NumCheckSet
    for j=1:NumVar
        X(j)=CSET(i,j);
    end
    Capprox(i)=sim(net,X);

```

```

Cdeviate(i)=abs(Capprox(i)-CVALUE(i));
Cerror(i)=abs(Cdeviate(i)/CVALUE(i));
CheckRMSfit=CheckRMSfit+Cdeviate(i)^2;
end
[Cmaxdeviate,maxCdeviateIndex]=max(Cdeviate);
[Cerror,maxCerrorIndex]=max(Cerror);
CheckRMSfit=sqrt(CheckRMSfit/NumCheckSet);

% *****

% Print Results to the screen
fprintf('-----\n');
fprintf('-----\n');
fprintf('      NEURAL NETWORK RESULTS      \n');
fprintf('-----\n');
fprintf('-----\n\n\n');
fprintf('Total Hidden Layer Numbers= %i \n\n', Layernumber);

switch Layernumber
case 1,
    fprintf('Layer 1 Neuron Number= %i \n', Layer1neuronnumber);
    fprintf('Layer 1 Transfer Function= %s \n', Layer1transferfun);
    fprintf('\n');
case 2,
    fprintf('Layer 1 Neuron Number= %i \n', Layer1neuronnumber);
    fprintf('Layer 1 Transfer Function= %s \n', Layer1transferfun);
    fprintf('\n');

    fprintf('Layer 2 Neuron Number= %i \n', Layer2neuronnumber);
    fprintf('Layer 2 Transfer Function= %s \n', Layer2transferfun);
    fprintf('\n');
case 3,
    fprintf('Layer 1 Neuron Number= %i \n', Layer1neuronnumber);
    fprintf('Layer 1 Transfer Function= %s \n', Layer1transferfun);
    fprintf('\n');

```

```

fprintf('Layer 2 Neuron Number= %i \n', Layer2neuronnumber);
fprintf('Layer 2 Transfer Function= %s \n', Layer2transferfun);
fprintf('\n');

fprintf('Layer 3 Neuron Number= %i \n', Layer3neuronnumber);
fprintf('Layer 3 Transfer Function= %s \n', Layer3transferfun);
fprintf('\n');
end

fprintf('Network Learning rate(step size)= %f\n',net.trainParam.lr);
fprintf('Network Epochs(iteration) number= %i\n',net.trainParam.epochs);
fprintf('Network Goal(convergence) criteria= %f\n',net.trainParam.goal);

fprintf('\n');
fprintf('Number of TRAIN Set= %i\n',NumTrainSet);
fprintf('Number of CHECK Set= %i\n',NumCheckSet);

fprintf('\n\n');
fprintf('Set#   TrainApproximate   TrainTest\n');
fprintf('-----\n');

for i=1:NumTrainSet
    fprintf('%i   %f   %f\n', i, Tapprox(i), TVALUE(i));
end

fprintf('\n\n');

fprintf('Set#   CheckApproximate   CheckTest\n');
fprintf('-----\n');

for i=1:NumCheckSet
    fprintf('%i   %f   %f\n', i, Capprox(i), CVALUE(i));
end

```

```
fprintf("\n\n");
```

```
fprintf('TrainRMSfit CheckRMSfit\n');
```

```
fprintf('----- \n');
```

```
fprintf('%f    %f\n', TrainRMSfit, CheckRMSfit);
```

```
fprintf("\n\n");
```

```
fprintf('MaxTrainDeviation MaxCheckDeviation\n');
```

```
fprintf('----- \n');
```

```
fprintf('%f    %f\n', Tmaxdeviate, Cmaxdeviate);
```

```
fprintf("\n\n");
```

```
fprintf('MaxTrainDeviateIndex MaxCheckDeviateIndex\n');
```

```
fprintf('----- \n');
```

```
fprintf('%i    %i \n', maxTdeviateIndex, maxCdeviateIndex);
```

```
fprintf("\n\n");
```

```
fprintf('MaxTrainError MaxCheckError%%\n');
```

```
fprintf('----- \n');
```

```
fprintf('%f    %f\n', 100*Termax, 100*Cermax);
```

```
fprintf("\n\n");
```

```
fprintf('MaxTrainErrorIndex MaxCheckErrorIndex\n');
```

```
fprintf('----- \n');
```

```
fprintf('%i    %i \n',maxTerrorIndex, maxCerrorIndex);
```

```
fprintf("\n\n");
```

```

%
*****

**

% Print Results to the File: "NeuralNetworkResults.m"
file1 = fopen('NeuralNetworkResults.m','w');
fprintf(file1,'-----\n');
fprintf(file1,'-----\n');
fprintf(file1,'      NEURAL NETWORK RESULTS      \n');
fprintf(file1,'-----\n');
fprintf(file1,'-----\n\n');
fprintf(file1,'Total Hidden Layer Numbers= %i \n\n', Layernumber);

switch Layernumber
case 1,
    fprintf(file1,'Layer 1 Neuron Number= %i \n', Layer1neuronnumber);
    fprintf(file1,'Layer 1 Transfer Function= %s \n', Layer1transferfun);
    fprintf(file1,'\n');
case 2,
    fprintf(file1,'Layer 1 Neuron Number= %i \n', Layer1neuronnumber);
    fprintf(file1,'Layer 1 Transfer Function= %s \n', Layer1transferfun);
    fprintf(file1,'\n');

    fprintf(file1,'Layer 2 Neuron Number= %i \n', Layer2neuronnumber);
    fprintf(file1,'Layer 2 Transfer Function= %s \n', Layer2transferfun);
    fprintf(file1,'\n');
case 3,
    fprintf(file1,'Layer 1 Neuron Number= %i \n', Layer1neuronnumber);
    fprintf(file1,'Layer 1 Transfer Function= %s \n', Layer1transferfun);
    fprintf(file1,'\n');

    fprintf(file1,'Layer 2 Neuron Number= %i \n', Layer2neuronnumber);
    fprintf(file1,'Layer 2 Transfer Function= %s \n', Layer2transferfun);
    fprintf(file1,'\n');

```

```

fprintf(file1,'Layer 3 Neuron Number= %i \n', Layer3neuronnumber);
fprintf(file1,'Layer 3 Transfer Function= %s \n', Layer3transferfun);
fprintf(file1,'\n');
end

fprintf(file1,'Network Learning rate(step size)= %f\n',net.trainParam.lr);
fprintf(file1,'Network Epochs(iteration) number= %i\n',net.trainParam.epochs);
fprintf(file1,'Network Goal(convergence) criteria= %f\n',net.trainParam.goal);

fprintf(file1,'\n');
fprintf(file1,'Number of TRAIN Set= %i\n',NumTrainSet);
fprintf(file1,'Number of CHECK Set= %i\n',NumCheckSet);

fprintf(file1,'\n\n');
fprintf(file1,'Set#  TrainApproximate  TrainTest\n');
fprintf(file1,'-----\n');

for i=1:NumTrainSet
    fprintf(file1,'%i  %f  %f\n', i, Tapprox(i), TVALUE(i));
end

fprintf(file1,'\n\n');

fprintf(file1,'Set#  CheckApproximate  CheckTest\n');
fprintf(file1,'-----\n');

for i=1:NumCheckSet
    fprintf(file1,'%i  %f  %f\n', i, Capprox(i), CVALUE(i));
end

fprintf(file1,'\n\n');

fprintf(file1,'TrainRMSfit  CheckRMSfit\n');
fprintf(file1,'-----\n');

```

```

fprintf(file1,'%f    %f\n', TrainRMSfit, CheckRMSfit);

fprintf(file1,'\n\n');

fprintf(file1,'MaxTrainDeviation  MaxCheckDeviation\n');
fprintf(file1,'----- \n');
fprintf(file1,'%f    %f\n', Tmaxdeviate, Cmaxdeviate);

fprintf(file1,'\n\n');

fprintf(file1,'MaxTrainDeviatIndex  MaxCheckDeviatIndex\n');
fprintf(file1,'----- \n');
fprintf(file1,'%i    %i\n', maxTdeviateIndex, maxCdeviateIndex);

fprintf(file1,'\n\n');

fprintf(file1,'MaxTrainError  MaxCheckError%%\n');
fprintf(file1,'----- \n');
fprintf(file1,'%f    %f\n', 100*Termax, 100*Cermax);

fprintf(file1,'\n\n');

fprintf(file1,'MaxTrainErrorIndex  MaxCheckErrorIndex\n');
fprintf(file1,'----- \n');
fprintf(file1,'%i    %i\n',maxTerrorIndex, maxCerrorIndex);

fprintf(file1,'\n\n');
fclose(file1);

```

EK 2 : CEVAP YÜZEYİ

Programda sadece örnek teşkil etmesi için ilk beş deney setinin girişi yapılarak en iyi sonucu veren *Altıncı dereceden polinom ve birinci ve üçüncü değişken ters, ikinci değişken doğru orantılı* olarak seçilmiş olan model girişleri yapılmış hali verilmiştir.

```
% Program: RESPONSE SURFACE METHODOLOGY implementation
% This program creates Polynomial Response Surface
%
=====
% ===== USER INPUT: PART-1
% =====
%
=====
% Enter Number of Variables, Train Data Sets and Check Data Sets

NumVar=3;
NumTrainSet=85;
NumCheckSet=5;

% Select the ORDER of Polynomial Response Surface by Model Order
% ModelOrder=1 : "Linear", ModelOrder=2 : "Parabolic",
% ModelOrder=3 : "Cubic", ModelOrder=4 : "4th order",
% ModelOrder=5 : "5th order", ModelOrder=6 : "6th order"
% ModelOrder=7 : "7th order", ModelOrder=8 : "8th order",
% ModelOrder=9 : "9th order", ModelOrder=10 : "10th order"

ModelOrder=6;
ReciprocalFlag=[1 0 1]; % Reciprocal Flag (0= No Reciprocal, 1=Yes Reciprocal)
ModelType=1; % ModelType=1 (Polynomial), ModelType=2 (Logarithmic)
Backpress=0; % 1=Use Backward Press elimination, 0=Do not use BPRESS

%
=====
% ===== END OF USER INPUT: PART-1
% =====
%
=====

NumTerm=1;
for i=1:ModelOrder
    fact=1;
```

```

    for j=NumVar:NumVar+i-1
        fact=fact*j;
    end
    fact=fact/factorial(i);
    NumTerm=NumTerm+fact;
end

NumTermNeeded=NumTerm;
fprintf('Number of Data Set Needed is :%i\n\n',NumTermNeeded)

if ModelOrder>10
    error('Maximum order allowed for rsmodel has been exceeded!!!');
end

% Memory allocation
TSET=zeros(NumTrainSet,NumVar);
TCOEFF=zeros(NumTrainSet,NumTerm);
TTPOZE=zeros(NumTerm,NumTrainSet);
X=zeros(NumVar,1);
TVALUE=zeros(NumTrainSet,1);
Fitererror=zeros(NumTrainSet,1);
Tdeviate=zeros(NumTrainSet,1);
Cdeviate=zeros(NumCheckSet,1);
CSET=zeros(NumCheckSet,NumVar);
CVALUE=zeros(NumCheckSet,1);
Capprox=zeros(NumCheckSet,1);
Approxerror=zeros(NumCheckSet,1);
A=zeros(NumTerm,NumTerm);
B=zeros(NumTerm,1);
% BETA=zeros(NumTerm,1);

%
=====
% ===== USER INPUT: PART-2 =====
%
=====

%1. Train Set
TSET(1, 1)= 1.33;
TSET(1, 2)= 600;
TSET(1, 3)= 4;
TVALUE(1)=3.319672131;

%2. Train Set
TSET(2, 1)= 1.33;
TSET(2, 2)= 600;
TSET(2, 3)= 6;
TVALUE(2)=2.197745902;

```

```
%3. Train Set
TSET(3, 1)= 1.33;
TSET(3, 2)= 600;
TSET(3, 3)= 7.5;
TVALUE(3)=1.862704918;
```

```
%4. Train Set
TSET(4, 1)= 1.33;
TSET(4, 2)= 1000;
TSET(4, 3)= 2;
TVALUE(4)=8.93442623;
```

```
%5. Train Set
TSET(5, 1)= 1.33;
TSET(5, 2)= 1000;
TSET(5, 3)= 4;
TVALUE(5)=5.174180328;
```

```
..
..
..
..
..
```

```
% *****
% *****
```

```
%1. Check Set
CSET(1, 1)= 1.33;
CSET(1, 2)= 600;
CSET(1, 3)= 2;
CVALUE(1)=5.650614754;
```

```
%2. Check Set
CSET(2, 1)= 1.60;
CSET(2, 2)= 1000;
CSET(2, 3)= 4.0;
CVALUE(2)=4.559;
```

```
%3. Check Set
CSET(3, 1)= 1.8;
CSET(3, 2)= 1400;
CSET(3, 3)= 6.0;
CVALUE(3)=3.835;
```

```
%4. Check Set
CSET(4, 1)= 2.2;
CSET(4, 2)= 2200;
CSET(4, 3)= 3.0;
CVALUE(4)=10.073;
```

```
%5. Check Set
CSET(5, 1)= 2.6;
CSET(5, 2)= 1400;
CSET(5, 3)= 2.0;
CVALUE(5)=8.019;
```

```
%
=====
%
=====
% =====      END OF USER INPUT: PART-2      =====
%
=====
%
=====
```

```
if NumTrainSet<NumTerm
    error('Data is insufficient to create the model!!!');
end

if ModelType==1
    for i=1:NumTrainSet
        for j=1:NumVar
            if ReciprocalFlag(j)==1
                if TSET(i,j)==0
                    error('Reciprocal of zero Training value can lead to infinity!!!');
                else
                    TSET(i,j)=1/TSET(i,j);
                end
            end
        end
    end
end
```

```
for i=1:NumCheckSet
    for j=1:NumVar
        if ReciprocalFlag(j)==1
            if CSET(i,j)==0
                error('Reciprocal of zero Check value can lead to infinity!!!');
            else
                CSET(i,j)=1/CSET(i,j);
            end
        end
    end
end
end
```

```
if ModelType==2
```

```

for i=1:NumTrainSet
    if TVALUE(i)<=0
        error('Negative or zero Training value can not be used with logarithmic
scale!!!');
    else
        TVALUE(i)=log(TVALUE(i));
    end

    for j=1:NumVar
        if TSET(i,j)<=0
            error('Negative or zero Training variable can not be used with logarithmic
scale!!!');
        else
            TSET(i,j)=log(TSET(i,j));
        end
    end
end

for i=1:NumCheckSet
    for j=1:NumVar
        if CSET(i,j)<=0
            error('Negative Check variable can not be used with logarithmic scale!!!');
        else
            CSET(i,j)=log(CSET(i,j));
        end
    end
end

for i=1:NumTrainSet
    TCOEFF(i,1)=1;
end

for NS=1:NumTrainSet
    for i=1:NumVar
        X(i)=TSET(NS,i);
    end

    % linear response surface model
    if ModelOrder>=1
        NUM=1;
        for i=1:NumVar
            NUM=NUM+1;
            TCOEFF(NS,NUM)=X(i);
        end
    end
end

```

```

% quadratic response surface model
if ModelOrder>=2
    for i=1:NumVar
        for j=i:NumVar
            NUM=NUM+1;
            TCOEFF(NS,NUM)=X(i)*X(j);
        end
    end
end
end

```

```

% cubic response surface model
if ModelOrder>=3
    for i=1:NumVar
        for j=i:NumVar
            for k=j:NumVar
                NUM=NUM+1;
                TCOEFF(NS,NUM)=X(i)*X(j)*X(k);
            end
        end
    end
end
end

```

```

% 4 th order response surface model
if ModelOrder>=4
    for i=1:NumVar
        for j=i:NumVar
            for k=j:NumVar
                for l=k:NumVar
                    NUM=NUM+1;
                    TCOEFF(NS,NUM)=X(i)*X(j)*X(k)*X(l);
                end
            end
        end
    end
end
end

```

```

% 5 th order response surface model
if ModelOrder>=5
    for i=1:NumVar
        for j=i:NumVar
            for k=j:NumVar
                for l=k:NumVar
                    for m=l:NumVar
                        NUM=NUM+1;
                        TCOEFF(NS,NUM)=X(i)*X(j)*X(k)*X(l)*X(m);
                    end
                end
            end
        end
    end
end
end

```

end

% 6 th order response surface model

if ModelOrder>=6

for i=1:NumVar

for j=i:NumVar

for k=j:NumVar

for l=k:NumVar

for m=l:NumVar

for n=m:NumVar

NUM=NUM+1;

TCOEFF(NS,NUM)=X(i)*X(j)*X(k)*X(l)*X(m)*X(n);

end

end

end

end

end

end

end

% 7 th order response surface model

if ModelOrder>=7

for i=1:NumVar

for j=i:NumVar

for k=j:NumVar

for l=k:NumVar

for m=l:NumVar

for n=m:NumVar

for o=n:NumVar

NUM=NUM+1;

TCOEFF(NS,NUM)=X(i)*X(j)*X(k)*X(l)...

*X(m)*X(n)*X(o);

end

end

end

end

end

end

end

end

% 8 th order response surface model

if ModelOrder>=8

for i=1:NumVar

for j=i:NumVar

for k=j:NumVar

for l=k:NumVar

for m=l:NumVar

for n=m:NumVar

for o=n:NumVar

```

        for p=o:NumVar
            NUM=NUM+1;
            TCOEFF(NS,NUM)=X(i)*X(j)*X(k)*X(l)...
                *X(m)*X(n)*X(o)*X(p);
        end
    end
end
end
end
end
end
end
end
end

% 9 th order response surface model
if ModelOrder>=9
    for i=1:NumVar
        for j=i:NumVar
            for k=j:NumVar
                for l=k:NumVar
                    for m=l:NumVar
                        for n=m:NumVar
                            for o=n:NumVar
                                for p=o:NumVar
                                    for q=p:NumVar
                                        NUM=NUM+1;
                                        TCOEFF(NS,NUM)=X(i)*X(j)*X(k)*X(l)...
                                            *X(m)*X(n)*X(o)*X(p)*X(q);
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end
end
end
end
end
end
end
end

% 10 th order response surface model
if ModelOrder>=10
    for i=1:NumVar
        for j=i:NumVar
            for k=j:NumVar
                for l=k:NumVar
                    for m=l:NumVar
                        for n=m:NumVar
                            for o=n:NumVar
                                for p=o:NumVar
                                    for q=p:NumVar

```

```

        for r=q:NumVar
            NUM=NUM+1;
            TCOEFF(NS,NUM)=X(i)*X(j)*X(k)*X(l)...
                *X(m)*X(n)*X(o)*X(p)*X(q)*X(r);
        end
    end
end
end
end
end
end
end
end
end
end
end

end

if Backpress==0

[BETA,RMStrain,maxTdeviate,maxTrainError]=fitting(TVALUE,NumTrainSet,NumTerm,...
    TCOEFF,ModelType);
else

[BETA,RMStrain,maxTdeviate,maxTrainError,NumTermLeft]=backwardpress(TVALUE,...
    NumTrainSet,NumTerm,TCOEFF,ModelType);
end

[Tapprox,TrainRMSfit,Termax,maxTerrorIndex,Tmaxdeviate,maxTdeviateIndex]=...
AccuracyCalculation(TSET,TVALUE,BETA,NumTrainSet,NumVar,ModelOrder,ModelType);

[Capprox,CheckRMSfit,Cermax,maxCerrorIndex,Cmaxdeviate,maxCdeviateIndex]=...
AccuracyCalculation(CSET,CVALUE,BETA,NumCheckSet,NumVar,ModelOrder,ModelType);

RecFlag=0;
for i=1:NumVar
    if ReciprocalFlag(i)==1
        RecFlag=1;
        break;
    end
end
end

```

```

%
=====
=====
% Print Results to the screen
fprintf('-----\n');
fprintf('-----\n');
fprintf('          RESPONSE SURFACE RESULTS          \n');
fprintf('-----\n');
fprintf('-----\n\n\n');

if ModelType==1
    if RecFlag==0
        fprintf('Model Type is: POLYNOMIAL\n');
    else
        fprintf('Model Type is: RECIPROCAL POLYNOMIAL\n');
    end
else
    fprintf('Model Type is: LOGARITHMIC\n');
end

switch ModelOrder
    case 1,
        fprintf('Model Order is: Linear\n\n');
    case 2,
        fprintf('Model Order is: Parabolic\n\n');
    case 3,
        fprintf('Model Order is: Cubic\n\n');
    case 4,
        fprintf('Model Order is: 4th order\n\n');
    case 5,
        fprintf('Model Order is: 5th order\n\n');
    case 6,
        fprintf('Model Order is: 6th order\n\n');
    case 7,
        fprintf('Model Order is: 7th order\n\n');
    case 8,
        fprintf('Model Order is: 8th order\n\n');
    case 9,
        fprintf('Model Order is: 9th order\n\n');
    case 10,
        fprintf('Model Order is: 10th order\n\n');
    otherwise,
        fprintf('Very High order selected. Check or change the model order!!!\n\n');
end

fprintf('\n');
fprintf('Number of TRAIN Set= %i\n',NumTrainSet);
fprintf('Number of CHECK Set= %i\n',NumCheckSet);

fprintf('\n\n');

```

```

fprintf('Set# TrainApproximate TrainTest\n');
fprintf('-----\n');

for i=1:NumTrainSet
    fprintf('%i %f %f\n', i, Tapprox(i), TVALUE(i));
end

fprintf('\n\n');

fprintf('Set# CheckApproximate CheckTest\n');
fprintf('-----\n');

for i=1:NumCheckSet
    fprintf('%i %f %f\n', i, Capprox(i), CVALUE(i));
end

fprintf('\n\n');

fprintf('TrainRMSfit CheckRMSfit\n');
fprintf('-----\n');
fprintf('%f %f\n', TrainRMSfit, CheckRMSfit);

fprintf('\n\n');

fprintf('MaxTrainDeviation MaxCheckDeviation\n');
fprintf('-----\n');
fprintf('%f %f\n', Tmaxdeviate, Cmaxdeviate);

fprintf('\n\n');

fprintf('MaxTrainDeviateIndex MaxCheckDeviateIndex\n');
fprintf('-----\n');
fprintf('%i %i\n', maxTdeviateIndex, maxCdeviateIndex);

fprintf('\n\n');

fprintf('MaxTrainError MaxCheckError%%\n');
fprintf('-----\n');
fprintf('%f %f\n', 100*Termax, 100*Cermax);

fprintf('\n\n');

fprintf('MaxTrainErrorIndex MaxCheckErrorIndex\n');
fprintf('-----\n');
fprintf('%i %i\n', maxTerrorIndex, maxCerrorIndex);

if Backpress==1
    fprintf('\n\n');
    fprintf('BACKWARD PRESS elimination used\n');
    fprintf('Number of Initial model constants is: %i\n', NumTerm);

```

```

    fprintf('Number of model constants after BACKWARD PRESS is: %i',...
    NumTermLeft);
end

fprintf('\n\n');

%
*****
*****
% Print Results to the File: "ResponseSurfaceResults.m"
file1 = fopen('ResponseSurfaceResults.m','w');
fprintf(file1,'-----\n');
fprintf(file1,'-----\n');
fprintf(file1,'          RESPONSE SURFACE RESULTS          \n');
fprintf(file1,'-----\n');
fprintf(file1,'-----\n\n');

if ModelType==1
    if RecFlag==0
        fprintf(file1,'Model Type is: POLYNOMIAL\n');
    else
        fprintf(file1,'Model Type is: RECIPROCAL POLYNOMIAL\n');
    end
else
    fprintf(file1,'Model Type is: LOGARITHMIC\n');
end

switch ModelOrder
case 1,
    fprintf(file1,'Model Order is: Linear\n\n');
case 2,
    fprintf(file1,'Model Order is: Parabolic\n\n');
case 3,
    fprintf(file1,'Model Order is: Cubic\n\n');
case 4,
    fprintf(file1,'Model Order is: 4th order\n\n');
case 5,
    fprintf(file1,'Model Order is: 5th order\n\n');
case 6,
    fprintf(file1,'Model Order is: 6th order\n\n');
case 7,
    fprintf(file1,'Model Order is: 7th order\n\n');
case 8,
    fprintf(file1,'Model Order is: 8th order\n\n');
case 9,
    fprintf(file1,'Model Order is: 9th order\n\n');
case 10,
    fprintf(file1,'Model Order is: 10th order\n\n');

```

```

    otherwise,
        fprintf(file1,'Very High order selected. Check or change the model order!!!\n\n');
end

fprintf(file1,'\n');
fprintf(file1,'Number of TRAIN Set= %i\n',NumTrainSet);
fprintf(file1,'Number of CHECK Set= %i\n',NumCheckSet);

fprintf(file1,'\n\n');
fprintf(file1,'Set#   TrainApproximate   TrainTest\n');
fprintf(file1,'-----\n');

for i=1:NumTrainSet
    fprintf(file1,'%i   %f   %f\n', i, Tapprox(i), TVALUE(i));
end

fprintf(file1,'\n\n');

fprintf(file1,'Set#   CheckApproximate   CheckTest\n');
fprintf(file1,'-----\n');

for i=1:NumCheckSet
    fprintf(file1,'%i   %f   %f\n', i, Capprox(i), CVALUE(i));
end

fprintf(file1,'\n\n');

fprintf(file1,'TrainRMSfit   CheckRMSfit\n');
fprintf(file1,'-----\n');
fprintf(file1,'%f   %f\n', TrainRMSfit, CheckRMSfit);

fprintf(file1,'\n\n');

fprintf(file1,'MaxTrainDeviation   MaxCheckDeviation\n');
fprintf(file1,'-----\n');
fprintf(file1,'%f   %f\n', Tmaxdeviate, Cmaxdeviate);

fprintf(file1,'\n\n');

fprintf(file1,'MaxTrainDeviateIndex   MaxCheckDeviateIndex\n');
fprintf(file1,'-----\n');
fprintf(file1,'%i   %i\n', maxTdeviateIndex, maxCdeviateIndex);

fprintf(file1,'\n\n');

fprintf(file1,'MaxTrainError   MaxCheckError%%\n');
fprintf(file1,'-----\n');
fprintf(file1,'%f   %f\n', 100*Termax, 100*Cermax);

fprintf(file1,'\n\n');

```

```

fprintf(file1,'MaxTrainErrorIndex  MaxCheckErrorIndex\n');
fprintf(file1,'----- \n');
fprintf(file1,'%i          %i \n',maxTerrorIndex, maxCerrorIndex);

if Backpress==1
    fprintf(file1,'\n\n');
    fprintf(file1,'BACKWARD PRESS elimination used\n');
    fprintf(file1,'Number of Initial model constants is: %i\n',NumTerm);
    fprintf(file1,'Number of model constants after BACKWARD PRESS is: %i',...
        NumTermLeft);
end

fprintf(file1,'\n\n');

fclose(file1);

```