YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

EN KÜÇÜK KARESEL ORTALAMA ALGORİTMASI YARDIMIYLA DOĞRUSAL KANALLARIN DENGELENMESİ

Elektrik-Elektronik Müh. Aziz ÖZ

F.B.E. Elektronik ve Haberleşme Mühendisliği Anabilim Dalı
Haberleşme Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Yrd. Doç. Dr. Soner ÖZGÜNEL

İSTANBUL,1999

İÇİNDEKİLER

Sayfa

ÖNSÖZ.....iv

ÖZET.....v

ABSTRACT.....vi

1 GİRİŞ.....1

2 DURAĞAN AYRIK ZAMANLI RASGELE SÜREÇLER.....4

2.1 Ayrık Zamanlı Bir Rasgele Sürecin Kısmi Özellikleri.....5
2.2 İlişki Matrisi.....7
2.2.1 Ergodiklik.....8
2.2.2 İlişki matrisinin özellikleri.....8
2.3 Özdeğer Analizi.....15
2.3.1 Özdeğer ve özvektörlerin özellikleri.....18
2.4 Güç Spektral Yoğunluk.....20

3 WIENER FİLTRELER.....21

3.1 Optimum Filtreleme Problemi.....22
3.2 Hata Başarım Yüzeyi.....24
3.3 Normal Denklemin Bulunması.....28
3.4 Diklik Kuralı.....29
3.5 Minimum Karesel Ortalama Hata.....31
3.6 Sayısal Örnek.....33
3.7 Hata Başarım Yüzeyinin Kanonik Formu.....33

4 UYARLAMALI FİLTRE ALGORİTMALARI.....35

4.1 Giriş.....35
4.2 En Dik İniş Algoritması.....36

4.3 En Küçük Karesel Ortalama Algoritması (LMS).....42
4.3.1 Temel varsayımlar.....46
4.4 Uyarlamalı Filtre Uygulamaları.....48
4.4.1 Uyarlamalı doğrusal birleştirici 48
4.4.2 LMS uyarlamalı filtre.....49

5 UYARLAMALI KANAL DENGELEME VE UYGULAMALAR.....51

5.1 Uyarlamalı Dengeleme Kavramı 53
5.2 En Küçük Karesel Ortalama Algoritması (LMS) Kullanarak Kanal Dengeleme...57
5.3 Uyarlamalı Dengeleyicinin İmpuls Yanıtı.....63
5.4 Öğrenme Eğrileri 64

6 İŞARETLER ARASINDAKİ İLİŞKİLERİN İNCELENMESİ.....142

7 SONUÇLAR.....152

KAYNAKLAR 154

EKLER 156

EK-1 Farklı Adım Uzunluklarına Göre Öğrenme Eğrisini Çıkaran, C Dili ile Yazılmış Program.....157
EK-2 Farklı Kanal Kontrol Parametre Değerlerine Göre Öğrenme Eğrisini Çıkaran, C Dili ile Yazılmış Proram.....164
EK-3 Farklı Filtre Katsayılarına Göre Öğrenme Eğrisini Çıkaran, C Dili ile Yazılmış Program.....171
EK-4 Kanal Girişinin Özilişki Matrisini Hesaplayan, C Dili ile Yazılmış Program.....177
EK-5 Kanal Girişi ile Gürültülü Çıkış Arasındaki Çapraz İlişki Matrisini Hesaplayan, C Dili ile Yazılmış Program.....180
EK-6 Uyarlamalı Dengeleyicinin Çıkışı ile İstenen Yanıt Arasındaki Çapraz İlişki Matrisini Hesaplayan, C Dili ile Yazılmış Program.....183

EK-7 Kanalın Gürültülü Çıkışının Özilişki Matrisini Hesaplayan,
C Dili ile Yazılmış Program.....189

ÖZGEÇMİŞ 192

ÖNSÖZ

Bu çalışmanın hazırlanmasında çok büyük yardımları olan saygıdeğer hocam Yrd. Doç. Dr. Soner Özgünel'e ve tezin yazılmasında büyük bir çaba harcayan sevgili Çilem Küçükkeleş'e sonsuz teşekkürlerimi sunarım.

Aziz Öz

ÖZET

Uyarlamalı filtreler, kanal dengeleme, kanal özdeşleme, doğrusal öngörü gibi birçok problemin çözümünde kullanılmaktadır. Bu çalışmada, kanal dengeleme problemi incelenmektedir.

Kanal dengelemenin amacı, kanalın bozucu etkilerini gidererek, iletilen işareti alınan işaretten çeşitli algoritmalarla yeniden elde etmektir. Uyarlamalı kanal dengeleme ile ilgili olarak, Wiener filtre teorisini temel alan çeşitli algoritmalar vardır. Bu çalışmada doğrusal kanalları dengelemek üzere en küçük karesel ortalama (LMS) algoritması kullanılmaktadır. Bu algoritmanın avantajı, basit hesaplama gerektirmesi, özilişki fonksiyonları ve özilişki matrisinin tersinin hesaplanmasına gerek duyulmaması ve seçilen adım uzunluğuna göre hızlı yakınsama özelliğine sahip olmasıdır.

Bu çalışmada, en küçük karesel ortalama algoritması kullanılarak, doğrusal kanalların dengelenmesi problemi incelenmektedir. Farklı kanal kontrol parametresi, adım uzunluğu, gürültü varyansı ve dengeleyici parametre sayısı için, bilgisayar simulasyonuyla elde edilen öğrenme eğrileri yardımıyla, en küçük karesel ortalama algoritmasının başarımı incelenmektedir. Adım uzunluğu büyük seçildiği zaman, yakınsama hızının artmakta olduğu, fakat karesel ortalama hatanın durağan durum değerinin büyük olduğu görülmektedir. Kanal kontrol parametresi büyük seçildiğinde, hatanın durağan durum değerinin büyük olduğu ve ayrıca öğrenme eğrilerindeki dalgalanmaların fazla olduğu gözlenmektedir. Gürültü varyansı arttıkça, karesel ortalama hatanın da arttığı ve yakınsamanın yavaşladığı görülmektedir. Dengeleyicinin parametre sayısının çok büyük seçilmesi, öğrenme eğrilerinde dalgalanmalara ve hatanın durağan durum değerinin büyümesine neden olmaktadır.

Ayrıca bu çalışmada, sistemin farklı noktalarındaki işaretler arasındaki ilişkiler (korelasyon) incelenmektedir. Kanal giriş dizisinin istatistiksel bağımsız olduğu, dengeleyicinin girişi ile çıkışı arasında bir ilişkinin olmadığı gözlenmektedir. Ancak dengeleyicinin çıkışı ile istenen yanıt arasında ve kanal girişi ile gürültülü çıkışı arasında bir çapraz ilişkinin olduğu görülmektedir.

ABSTRACT

The purpose of all kind of equalization (adaptive or not) is to undo the distorting effects of the channel and recover the transmitted signal from the received signal. A channel equalizer is an optimal filter since it tries to produce as good estimate of the transmited signal as possible. For adaptive equalization, there are several methods based on Wiener filter theory. In this study least mean square (LMS) algorithm is used. Its advantages are; being simple, no need to calculate and to take the inverse of the autocorrelation matrix.

The performance analysis of this algorithm is investigated based on the learning curves. In this performance analysis, different channel models are chosen and the effect of noise variance, step size and channel control parameters are searched. It is seen that the resuls obtained are in accordance with the results in literature.

In addition, the correlations between the signals on some nodes of the system are calculated. By comparing the results obtained here with the results of the previous section, it is shown that there is an accordance.

1 GİRİŞ

Kanal dengeleme, birçok araştırmacının ilgi odağı olmuş ve bu konu üzerinde pek çok çalışma yapılmıştır. Bu çalışmada, en küçük karesel ortalama (Least Mean Square, LMS) algoritması kullanılarak doğrusal kanalların dengelenmesi incelenmektedir.

Kanal dengelemede amaç, kanalın bozucu etkilerini gidererek, verici tarafından iletilen işareti, alıcı tarafta alınan işaretten yeniden elde etmektir. Kanal dengeleyici, kanalın çıkışına yerleştirilerek, kanalın iletilen işaret üzerindeki etkileri yok edilmekte ve böylece gerçek işaretin doğru bir biçimde kestirimi sağlanmaktadır. Birçok durumda olduğu gibi, kanalın yapısı önceden bilinmiyor ya da zamanla değişiyor olabilir. Bu durumda, dengeleyicinin uyarlamalı bir biçimde tasarlanması gerekmektedir.

Uyarlamalı dengeleyiciler öğrenme modu ve çalışma modu olmak üzere iki modda çalışırlar. Öğrenme modunda, alıcıda da bilinen bir test işareti (öğrenme dizisi) verici tarafından iletilir. Kestirim hatası olarak bilinen, alıcıya gelen işaretle test işareti arasındaki fark, karesel ortalama hatayı azaltacak biçimde dengeleyicinin katsayılarının ayarlanmasında kullanılır. Bu durumda, dengeleyicinin aktarım işlevi, kanalın aktarım işlevinin tersi olmaktadır.

Uyarlamalı dengeleyicilerin çalışma ilkesini oluşturan yinelemeli algoritmalar, Wiener Filtre Teorisi, Kalman Filtre Teorisi gibi temel teoriler üzerine kurulmuştur. Wiener Filtre Teorisi, yalnızca durağan süreçlere uygulanmakla birlikte, Kalman Filtre Teorisi, durağan olmayan süreçler için de bir çözüm sunmaktadır. Wiener Filtreleme çözümü optimumdur ve bu çözüm Wiener çözümü olarak bilinir.

Wiener Filtre Teorisi üzerine kurulmuş olan, En Dik İniş Algoritması, En Küçük Karesel Ortalama Algoritması (LMS), Yinelemeli En Küçük Kareler Algoritması (RLS), Yinelemeli En Küçük Kareler Kafes Algoritması (RLSL) gibi birçok yinelemeli algoritma bulunmaktadır.

Bu çalışmada, en küçük karesel ortalama algoritması (LMS) kullanılmaktadır. Bu algoritma, daha çok doğrusal kanallar için optimum bir çözüm sunmaktadır. Adım uzunluğu, filtre katsayıları, gürültü varyansı ve kanal kontrol parametresi tarafından

belirlenen özdeğer yayılımı gibi parametrelerin bu algoritmanın başarımına etkileri, öğrenme eğrileri yardımıyla irdelenmektedir.

İkinci bölümde, durağan ayrık zamanlı rasgele süreçlerin karakteristikleri ele alınmaktadır. İlişki matrisinin çıkarılması, özellikleri ve özdeğer konusu incelenmektedir.

Üçüncü bölümde, uyarlamalı filtrelerin çalışma ilkelerini belirleyen Wiener Filtre Teorisi ele alınmakta ve ayrıca, durağan ortamda bir enine filtrenin hata başarım yüzeyinin kanonik formu incelenmektedir.

Dördüncü bölümde, önceki bölümlerde anlatılanların ışığı altında uyarlamalı filtre kavramı ele alınmakta, başta en küçük karesel ortalama hata algoritması olmak üzere ilgili algoritmalar ve uygulama alanları incelenmektedir.

Beşinci bölümde, uyarlamalı dengeleme kavramı ve kanal karakteristikleri incelenmektedir. Daha sonra, en küçük karesel ortalama hata algoritmasını kullanan bir uyarlamalı dengeleyici yardımıyla doğrusal kanalların dengelenmesine ilişkin sayısal uygulamalar gerçekleştirilmektedir. Bu algoritmanın başarımını ekileyen adım uzunluğu, gürültü varyansı, kanal kontrol parametresi ve dengeleyici katsayıları değiştirilerek öğrenme eğrileri bir bilgisayar simulasyonuyla elde edilmektedir. Bu parametrelerden adım uzunluğunun yakınsama hızı ve karesel ortalama hatanın durağan durum değeri üzerindeki etkileri, sözkonusu eğriler yardımıyla incelenmektedir. Adım uzunluğu arttıkça yakınsama hızının arttığı ancak, karesel ortalama hatanın durağan durum değerinin de büyük olduğu görülmektedir. Aynı biçimde, kanal kontrol parametresinin de yakınsama hızı, karesel ortalama hatanın durağan durum değeri üzerindeki etkileri irdelenmektedir. Kanal kontrol parametresi büyüdükçe, özellikle karesel ortalama hatanın durağan durum değerinin hızla büyüdüğü görülmektedir. Gürültü varyansı arttıkça, eğer kanal kontrol parametresi büyük ise öğrenme eğrisindeki dalgalanmalarda büyük bir artış gözlenmektedir. Kanal kontrol parametresinin ve gürültü varyansının artmasına ek olarak, dengeleyici parametre sayısı da büyük seçilerek karesel ortalama hatanın durağan durum değerindeki dalgalanmalar incelenmektedir. Yine adım uzunluğu ile dengeleyici parametre sayısı arasındaki ilişki öğrenme eğrileri yardımıyla irdelenmektedir.

Altıncı bölümde, bu çalışmada ele alınan sistemin farklı noktalarında elde edilen işaretler arasındaki ilişkiler (korelasyon) irdelenmektedir. Burada varılan sonuçların, daha önceki bölümlerde elde edilen sonuçları destekler biçimde olduğu gözlenmektedir. Kanal girişinin özilişki matrisinin aldığı değerler gözönüne alındığında, kanal giriş dizisinin istatistiksel bağımsız olduğu görülmektedir. Aynı biçimde istenen yanıt ile dengeleyici çıkışı arasındaki çapraz ilişki değerleri elde edilmekte ve bu değerlerden, bu işaretler arasında bir ilişkinin varolduğu sonucuna ulaşılmaktadır. Burada elde edilen sonuçlar, Beşinci Bölüm'de elde edilenlerle karşılaştırılarak yorumlanmaktadır. Ayrıca dengeleyici girişi ile çıkışı ve kanal girişi ile çıkışı arasındaki çapraz ilişkiler elde edilerek değerlendirilmektedir.

Yedinci bölümde, önceki bölümlerde elde edilen sonuçlar yorumlanarak, genel bir değerlendirme yapılmaktadır.

2 DURAĞAN AYRIK ZAMANLI RASGELE SÜREÇLER

Rasgele süreçler, olasılık yasalarına bağlı istatistiksel bir olayın zaman fonksiyonlarının bir toplamıdır. Bir olayın istatistiksel olması demek, o olayla ilgili bir deney gerçekleştirmeden önce o olayın zamanla değişim şeklinin nasıl olacağının tam olarak belirlenebilmesinin olanaksız olması demektir. Konuşma işaretleri, televizyon işaretleri, radar işaretleri, haberleşme kanalı çıkışındaki işaretler, sayısal bilgisayar verileri ve gürültü birer rasgele süreçtir. Genel olarak rasgele süreçlerin yalnızca bir örnek fonksiyonu gözlenebilir. Öteki tüm örnek fonksiyonlar, olmadığı halde, olması olası durumları belirtir. Rasgele süreçleri incelemede ilk adımlardan biri, verilen herhangi bir sürecin karekteristiklerinin tanımlanmasında kısa yoldan yaralanabilecek, uygun kavramları geliştirmektir.Bunu sağlamada elverişli bir yol, çiftler biçiminde düzenlenmiş tanımlayıcı setlerin kullanılmasıdır. Bu setler, sürekli-ayrık, deterministik-deterministik olmayan, durağan-durağan olmayan, ergodik-ergodik olmayan olarak sıralanabilir. Burada ilgileneceğimiz rasgele süreçler, daha çok ayrık durağan ve düzgün dağılımlı olanıdır. Rasgele süreci özgün haliyle gerçel değerli sürekli bir zaman aralığında tanımlamak da olasıdır ama, sürecin işlenmeden önce en büyük frekans bileşeninin iki katına eşit ya da daha büyük bir örnekleme hızıyla zamanda düzgün olarak örneklenmesi tercih edilmektedir. Sürekli rasgele süreçler, belirli sınırlar arasındaki olası değerlerden herhangi bir değeri alabileceği varsayılan süreçlerdir. Elektron tüplerinde oluşan çarpışma gürültüsü ya da rüzgarın hız buna örnektir. Ayrık rasgele bir süreç, yalnızca belirli ve çoğunlukla sonlu sayıda değerler alabilir. Genel olarak doğal olaylarda geçmiş değerlerden yola çıkarak gelecek değerler tayin edilemez ancak kimi durumda örnek fonksiyonların geçmiş değerlerine ait bigiler yardımıyla, gelecek değerlerin tam olarak elde edilebileceği bir rasgele süreç tanımlamak olasıdır.

Bir rasgele sürecin zamanın sadece tek bir fonksiyonu olarak görülmemesi gerekir. Teorik olarak, rasgele süreçlerin sonsuz sayıda farklı gerçeklemeler içerdikleri kabul edilmektedir. Ayrık zamanlı bir rasgele sürecin tek bir gerçeklemesine ayrık zaman serisi ya da kısaca zaman serisi adı verilir. Örneğin, u(n) , u(n-1),..., u(n-M-+1) serisi, sürecin n anında yapılan şimdiki gözlemi u(n) ile beraber n-1, n-2, ..., n-M+1 anlarında yapılmış (M-1) adet geçmiş gözleminden oluşan bir zaman serisini göstermektedir.

Eğer bir rasgele sürecin istatistiksel özellikleri zamanın ötelenmesi ile birlikte herhangi bir değişikliğe uğramıyor ise o rasgele süreç dar anlamda durağan olarak kabul edilir. u(n),u(n-1),..,u(n-M+1) zaman serisi ile belirtilen ayrık zamanlı bir rasgele sürecin dar anlamda durağan olabilmesi için, n,n-1,..,n-M+1 anlarında yapılmış bu gözlemlerine ait birleşik olasılık yoğunluk fonksiyonun sabit bir M değerine karşılık n'ye ne değer verilirse verilsin hiç değişmeden aynı kalması gerekir.

2.1 Ayrık Zamanlı Bir Rasgele Sürecin Kısmi Özellikleri

Pratikte birleşik olasılık yoğunluk fonksiyonunun ölçüm yapılarak elde edilmesi olanaksızdır. Diğer bir deyişle, rasgele süreç üzerinde yapılmış gözlem değerlerine bakarak bu gözlemlere ilişkin birleşik olasılık yoğunluk fonksiyonu belirlenemez. Bu nedenle sürecin birinci ve ikinci momentlerini belirleyip, kendimizi sürecin bu kısmi karekteristikleri ile sınırlamak durumundayız. u(n),u(n-1),...,u(n-M+1) zaman serisi ile gösterilen ve kompleks değerler de alabilen ayrık zamanlı bir rasgele süreç ele alınsın. Bu sürece ilişkin ortalama değer fonksiyonu,

$$\mu[n]=E[u(n)] \tag{2.1}$$

ve sürecin özilişki foksiyonu,

$$r(n,n-k) = E\,[u(n)\,u^*(n-k)]\ , \qquad k = 0,\pm 1\pm 2,...., \tag{2.2}$$

şeklinde tanımlanır. Burada (*) işareti, kompleks eşleniği göstermektedir. Sürecin özdeğişinti fonksiyonu ise,

$$c(n.n-k)=E[(u(n)-\mu(n))(u(n-k)-\mu(n-k))^*]\ , \quad k = 0,\pm 1 \pm 2,... \tag{2.3}$$

ile ifade edilir. Bu üç eşitlikten de görüldüğü gibi sürecin ortalama değer, özilişki ve özdeğişinti fonksiyonları arasında,

$$c(n,n-k) = r(n,n-k)-\mu(n)\mu^*(n-k) \tag{2.4}$$

şeklinde bir ilişki vardır. Öyleyse, çeşitli n ve k değerlerinde sürecin bu kısmi karakteristiklerini elde etmek için ortalama değer fonsiyonu $\mu(n)$ ile birlikte özilişki $r(n,n-k)$ ya da özdeğişinti $c(n,n-k)$ fonksiyonlarından herhangi birinin belirlenmesi gereklidir. Kısmi karekteristikler iki önemli avantaj sağlamaktadır;

1. Pratik ölçümler için oldukça elverişlidirler.
2. Rasgele süreçler ile yapılan doğrusal (linear) işlemlere uyumluluk gösterirler.

Dar anlamda durağan ayrık zamanlı bir rasgele süreç için (2.1), (2.2), (2.3), eşitlikleri daha da basitleşir. Özellikle sürecin ortalama değer fonksiyonunun μ gibi bir sabite eşit olduğu durumda,

$$\mu[n] = \mu \quad , \quad \text{her n için} \tag{2.5}$$

şeklinde yazılır. Ayrıca özilişki ve özdeğişinti fonksiyonlarının sadece (n) ile (n-k) gözlem anları arasındaki farka, başka bir deyişle k değerine bağlı olur. Bu durum,

$$r(n,n-k) = r(k) \tag{2.6}$$

$$c(n,n-k) = c(k) \tag{2.7}$$

şeklinde gösterilir. $k=0$ olmasına, yani birbiri ardına gelen iki gözlem zamanı arasında geçen sürenin sıfır olmasına, sıfır zaman farkı denir. Bu durumda $r(0)$ değeri $u(n)$'in karesel ortalama değerine eşit olurken,

$$r(0)=E[\mathbf{u}(n)\mathbf{u}^*(n)] \tag{2.8}$$

$$r(0)=E[|\mathbf{u}(n)|^2] \tag{2.9}$$

$c(0)$ değeri de $u(n)$'in değişintisine eşit olmaktadır;

$$c(0)=\sigma_u^2 \tag{2.10}$$

(2.5), (2.6), ve (2.7) eşitliklerindeki koşullar, ayrık zamanlı rasgele sürecin dar anlamda durağan olmasını garanti etmek için yeterli değildir. Diğer yandan, dar anlamda durağan

olmayan ama, bu koşulları sağlayan ayrık zamanlı rasgele süreçler geniş anlamda durağan, zayıf durağan ya da ikinci dereceden durağan (stationary to the second order) kabul edilir. Dikkat edilmesi gereken bir diğer ilginç ayrıntı da, geniş anlamda durağan olan bir Gauss sürecinin aynı zamanda dar anlamda da durağan varsayılabilmesidir. $\mu(n)=\mu=0$ olması durumunda geniş anlamda durağan bir sürecin özilişki ve özdeğişinti fonksiyonları birbirine eşit kabul edilir. Analizinin kolay olmasından dolayı, bundan böyle aksi belirtilmediği sürece tüm rasgele süreçler sıfır ortalamalı kabul edilecektir.

2.2 İlişki Matrisi

u(n) vektörü, u(n),u(n-1),....,u(n-M+1) zaman serisinin elemanlarını içeren Mx1 boyutlu gözlem vektörü (observation vector) olsun.

$$\mathbf{u}^T(n)=[u(n),u(n-1),\ldots,u(n-M+1)] \tag{2.11}$$

Burada üstel işaret T, vektörün açılımını tek bir satıra yazabilmek amacı ile kullanılan transpoz (transposition) operatörünü belirtmektedir. Böyle bir zaman serisi ile gösterilen durağan ayrık zamanlı bir rasgele sürece ait ilişki matrisi ise gözlem vektörünün kendisi ile dış çarpımının (outer product) beklendik değeri şeklinde tanımlanır. **R** bu tanıma uyan MxM boyutlu ilişki matrisini göstermek üzere,

$$\mathbf{R}=E[\mathbf{u}(n)\mathbf{u}^H(n)] \tag{2.12}$$

eşitliği varolacaktır. Üstel işaret H, vektörün kompleks eşleniğinin transpozundan $\mathbf{u}^H(n)=(\mathbf{u}^*(n))^T$ oluşan Hermitian traspozunu belirtmektedir. (2.11) eşitliğinin (2.12) eşitliğinde yerine yazılmasına ek olarak geniş anlamda durağanlık koşulunun da sağlandığı kabul edilir. **R** ilişki matrisinin açılımı,

$$\mathbf{R}=\begin{bmatrix} r(0) & r(1) & \cdots & r(M-1) \\ r(-1) & r(0) & \cdots & r(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r(-M+1) & r(-M+2) & \cdots & r(0) \end{bmatrix} \tag{2.13}$$

biçiminde ifade edilir. Ana diyagonaldeki r(0) elemanı her zaman gerçel değerlidir. Matrisin diğer elemanlarının ne olacağı ise gözlem değerlerine bağlıdır. Doğal olarak bu elemanlar kompleks değerli gözlemler için kompleks, gerçel değerli gözlemler için ise gerçel olmaktadırlar.

2.2.1 Ergodiklik

Eşitlik (2.12)'de ifade edilen ilişki matrisi **R**, toplamsal ortalamanın sonucunu göstermektedir.Kimi durağan rasgele süreçler, sürecin hemen hemen tüm üyeleri sürecin sahip olduğu aynı istatistiksel davranışı gösteren bir özelliğe sahiptirler. Bu biçimde tek bir tipik örnek fonksiyonun incelenmesi ile bu istatistik davranışı saptamak olsasıdır. Bu tip sürece ergodik süreç denir. Eğer bir durağan rasgele süreç ergodik ise, sürecin zamansal ortalamasını yaklaşık olarak sürecin toplamsal ortalaması kabul edebiliriz. Bir rasgele sürecin ergodik olması için o sürecin öncelikle durağan olması gerekir.Bir ergodik sürecin ilişki matrisi,

$$\mathbf{R} = \lim_{N\to\infty} (1/N)\sum_{n=1}^{N} \mathbf{u}(n)\mathbf{u}^{H}(n) \qquad (2.14)$$

biçiminde olur.

2.2.2 İlişki matrisinin özellikleri

Ayrık zamanlı filtre tasarımında ve istatistiksel analizde ilişki matrisi önemli bir rol oynar. Dolaysiyle ilişki matrisinin çeşitli özelliklerinin ve bu özelliklerin ne anlama geldiklerinin çok iyi kavranılması gerekmektedir. (2.12) eşitliğine ilişkin tanımın kullanılması ile durağan ayrık zamanlı bir rasgele sürecin ilişki matrisinin aşağıda verilen özelliklere sahip olduğu görülür.

Özellik-1 Durağan ayrık zamanlı rasgele süreçlerin ilişki matrisleri Hermitian'dir.

İlişki matrisi **R** daima kare matristir. Kendi transpozuna eşit olan, $\mathbf{R} = \mathbf{R}^{T}$, kare matrisine simetrik matris denir. Kompleks değerler içeren bir kare matrisin simetrik olabilmesi için kendi kompleks eşleniğinin transpozuna, yani Hermitian transpozuna eşit olması gerekir.

Bu koşulu sağlayan kare matrislere Hermitian matris denir. Öyleyse **R** ilişki matrisinin Hermitian özelliği,

$$\mathbf{R}^H = \mathbf{R} \tag{2.15}$$

şeklinde açıklanabilir. **R** matrisinin bu özelliğini göstermenin diğer bir yolu da,

$$r(-k) = r^*(k)$$

yazmaktır. r(k), k zaman farkı için özilişki fonksiyonunun aldığı değerdir. Buna göre , geniş anlamda durağan bir sürecin **R** ilişki matrisinin tanımlanabilmesi için, r(k) özilişki fonksiyonunun k = 0,1,2,...,M-1 için aldığı M adet değerin bilinmesi gerekir. Böylece (2.13) ile verilen açılım,

$$\mathbf{R} = \begin{bmatrix} r(0) & r(1) & \cdots & r(M-1) \\ r^*(1) & r(0) & \cdots & r(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r^*(M-1) & r^*(M-2) & \cdots & r(0) \end{bmatrix} \tag{2.16}$$

şeklinde yazılabilir. Rasgele sürece ait gözlemlerin hepsinin gerçel değerli (real valued) olması özel bir durum yaratır. Çünkü bu durumda özilişki fonksiyonu r(k) her k değeri için gerçel olacağından, **R** ilişki matrisi de Hermitian matris yerine simetrik matris olur.

Özellik-2 Durağan ayrık zamanlı rasgele süreçlerin ilişki matrisleri Toeplitz olmaktadır.

Sadece bir kare matris Toeplitz olabilir. Bir kare matrisin Toeplitz olması için aynı diyagonal üzerinde bulunan tüm elemanların birbirine eşit olması gerekir. (2.16) ile verilen açılımdan da görülebileceği gibi ana diyagonaldeki tüm elemanlar r(0) değerine eşit olurken, ana diyagonalin bir üstündeki diyagonalde bulunan tüm elemanlar r(1) değerine ve ana diyagonalin bir altındaki diyagonalde bulunan tüm elemanlarsa r(-1) değerine eşit olmaktadır. Diğer diyagonallerdeki elemanlar da benzer şekilde birbirlerine eşit olduğuna göre buradan elde edilen sonuç **R** ilişki matrisinin Teoplitz olduğudur.

Diğer yandan ilişki matrisinin Teoplitz özelliğinin u(n) gözlem vektörü ile sunulan ayrık zamanlı rasgele sürecin geniş anlamda durağan kabul edilmesinin bir sonucu olduğuna özellikle dikkat edilmelidir. Öyleyse, geniş anlamda durağan olan her ayrık zamanlı rasgele sürecin ilişki matrisi Toeplitz'dir ve bunun tersi de (ilişki matrisi Toeplitz olan her ayrık zamanlı rasgele süreç geniş anlamda durağandır) doğrudur.

Özellik-3 Durağan ayrık zamanlı rasgele süreçlerin ilişki matrisleri her zaman negatif olmayan belirli ve hemen hemen her zaman pozitif belirli olmaktadır. **x** vektörü sıfırdan farklı, kopleks değerli ve Mx1 boyutlu keyfi bir vektör, y ise **u**(n) gözlem vektörü ile **x** vektörünün iç çarpımı olarak tanımlı skaler bir raslantı değişkeni olsun.

$$y = \mathbf{x}^H \mathbf{u}(n) \tag{2.17}$$

Her iki tarafın Hermitian transpozu alındığında,

$$y^* = \mathbf{u}^H(n)\mathbf{x} \tag{2.18}$$

elde edilir. Burada y bir skaler olduğu için Hermitian transpozu kompleks eşleniğine eşit olmaktadır. y raslantı değişkeninin karesel beklendik değeri,

$$E[|v|^2] = E[vv^*]$$

olduğuna göre,

$$\mathbf{x}^H \mathbf{R}\mathbf{x} \geq 0 \qquad (2.19)$$

yazılabilir ve (2.19) koşulunu sağlayan bir Hermitian formunun negatif olmayan belirli ya da pozitif yarı belirli olduğu söylenir. Öyleyse, geniş anlamda durağan süreçlerin ilişki matrislerinin her zaman için negatif olmayan belirli oldukları söylenebilir.

Eğer Hermitian formu, sıfırdan farklı her **x** vektörü için $\mathbf{x}^H\mathbf{R}\mathbf{x} > 0$ koşulunu sağlıyorsa **R** ilişki matrisinin pozitif belirli olduğu ifade edilir. Bu koşul, **u**(n) gözlem vektörünün M adet elemanını oluşturan rastlantı değişkenleri arasında herhangi bir doğrusal bağımlılık olmadıkça geniş anlamda durağan her süreç için sağlanır. Böyle bir özel durum ancak {u(n)} süreci K ≤ M olmak üzere K tane sinüzoidin toplamından oluşuyor ise ortaya çıkar. Pratikte bu ideal durumla o denli az karşılaşılır ki **R** ilişki matrisi, başta da belirtildiği gibi hemen hemen her zaman pozitif belirli kabul edilir. İlişki matrisinin pozitif belirli olması demek, determinantının ve bütün ana minörlerinin (principal minors) sıfırdan büyük olması demektir. Örneğin, M=2 için,

$$\begin{bmatrix} r(0) & r(1) \\ r^*(1) & r(0) \end{bmatrix} > 0$$

koşulunun M = 3 için,

$$\begin{bmatrix} r(0) & r(1) \\ r^*(1) & r(0) \end{bmatrix} > 0$$

$$\begin{bmatrix} r(0) & r(2) \\ r^*(1) & r(0) \end{bmatrix} > 0$$

$$\begin{bmatrix} r(0) & r(1) & r(2) \\ r^*(1) & r(0) & r(1) \\ r^*(2) & r^*(1) & r(0) \end{bmatrix} > 0$$

koşullarının sağlanıyor olması demektir. Bu koşullardan çıkarılabilecek diğer bir anlam ise ilişki matrisinin tekil olmayan bir matris olduğudur.

Eğer bir matris tekil olmayan bir matris ise o matrisin tersi vardır, tersi yok ise o matris tekil matristir. Öyleyse, ilişki matrislerinin hemen hemen her zaman tekil olmayan birer matris oldukları söylenebilir.

Özellik-4 Durağan ayrık zamanlı bir rasgele sürecin gözlem vektörünü oluşturan elemanların geri yönde yeniden düzenlenmesinin sürecin ilişki matrisi üzerinde oluşturacağı etki transpoza eşdeğerdir.

$\mathbf{u}^B(n)$, u(n) vektörünün elemanlarının geri yönde tekrar düzenlenmeleri sonucunda oluşan Mx1 boyutlu vektör olsun. Bu vektör,

$$\mathbf{u}^B(n)=[u(n-M+1)\ u(n-M+2),\ ...,u(n)] \tag{2.20}$$

şeklinde olacaktır. Üstel işaret B, vektörün geri yönde düzenlendiğini gösterir. $\mathbf{u}^B(n)$ vektörünün ilişki matrisi ise,

$$E\left[\mathbf{u}^B(n)\,\mathbf{u}^{BH}(n)\right]=\begin{bmatrix} r(0) & r(-1) & \cdots & r(-M+1) \\ r(1) & r(0) & \cdots & r(-M+2) \\ \vdots & \vdots & \ddots & \vdots \\ r(M-1) & r(M-2) & \cdots & r(0) \end{bmatrix} \tag{2.21}$$

haline gelir. (2.21) ile verilen matris, (2.13) ile verilen matrisle karşılaştırıldığında,

$$E[\mathbf{u}^B(n)\ \mathbf{u}^{BH}(n)]=\mathbf{R}^T \tag{2.22}$$

olduğu, yani yukarıda belirtilen özelliğin sağlandığı görülür.

Özellik-5 Durağan ayrık zamanlı bir rasgele sürecin sırasıyla M ve (M+1) adet gözlem

değerlerine ait $\mathbf{R}_M$ ve $\mathbf{R}_{M+1}$ ilişki matrisleri olsun. Burada,

$$\mathbf{R}_{M+1} = \left[\begin{array}{c|c} \mathbf{r}(0) & \mathbf{r}^H \\ --- & --- \\ \mathbf{r} & \mathbf{R}_M \end{array}\right] \tag{2.23}$$

ya da eşdeğeri olarak,

$$\mathbf{R}_{M+1} = \left[\begin{array}{c|c} \mathbf{R}_M & \mathbf{r}^{B*} \\ --- & --- \\ \mathbf{r}^{BT} & \mathbf{r}(0) \end{array}\right] \tag{2.24}$$

eşitlikleri ile verilen bir ilişki vardır. r(0), sıfır zaman farkı için sürecin özilişki fonksiyonunun aldığı değer olup, $\mathbf{r}^H$ ve $\mathbf{r}^{BT}$ vektörleri

$$\mathbf{r}^H = [\, r(1)\;\; r(2)\;\; \ldots\;\; r(M)\,] \tag{2.25}$$

$$\mathbf{r}^{BT} = [\, r(-M)\;\; r(-M+1)\ldots r(-1)] \tag{2.26}$$

şeklinde olmaktadır.

Özellik-5 tanımlanırken **R** ilişki matrisinin alt indisi olarak yazılan M ve (M+1) sembolleri bu matrisin yapılan gözlem sayısına ne derece bağımlı olduğunu göstermek amacıyla kullanılmıştır.

(2.23) eşitliğini ispatlamak üzere $\mathbf{R}_{M+1}$ matrisinin açık formu,

$$\mathbf{R}_{M+1} = \left[\begin{array}{c|cccc} r(0) & r(1) & r(2) & \cdots & r(M) \\ --- & --- & --- & --- & --- \\ r(-1) & r(0) & r(1) & \cdots & r(M-1) \\ r(-2) & r(1) & r(0) & \cdots & r(M-2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r(-M) & r(-M+1) & r(-M+2) & \cdots & r(0) \end{array}\right] \tag{2.27}$$

şeklinde parçalara ayrılıp, (2.13), (2.15), ve (2.25) eşitlikleri de (2.27) içinde yerlerine konulduğunda (2.23) ile verilen sonuca ulaşılır. Dolaysiyle (M+1) adet gözlem değerinden oluşan $\mathbf{u}(n)_{M+1}$ vektörünün açık formu ise,

$$\mathbf{u}(n)_{M+1} = \begin{bmatrix} u(n) \\ \hline u(n-1) \\ \vdots \\ u(n-M) \end{bmatrix} = \begin{bmatrix} u(n) \\ \hline u_M(n-1) \end{bmatrix} \tag{2.28}$$

şeklinde parçalara ayrılarak yazılabilir.

(2.24) eşitliğini tanılamak içinse $\mathbf{R}_{M+1}$ matrisinin açık formu (2.29) eşitliğindeki gibi parçalara bölünüp, (2.13), (2.15), ve (2.26) eşitlikleri de (2.29)'de yerlerine konulduğunda (2.24) ile verilen sonuca ulaşılır.

$$\mathbf{R}_{M+1} = \left[\begin{array}{cccc|c} r(0) & r(1) & \cdots & r(M-1) & r(M) \\ r(-1) & r(0) & \cdots & r(M-2) & r(M-1) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ r(-M+1) & r(-M+2) & \cdots & r(0) & r(1) \\ \hline r(-M) & r(-M+1) & \cdots & r(-1) & r(0) \end{array}\right] \tag{2.29}$$

Bu ikinci sonuca göre, $\mathbf{u}(n)_{M+1}$ vektörünün açık formu,

$$\mathbf{u}(n)_{M+1} = \begin{bmatrix} u(n) \\ \vdots \\ u(n-M+1) \\ \hline u(n-M) \end{bmatrix} = \begin{bmatrix} u_M(n) \\ \hline u(n-M) \end{bmatrix} \tag{2.30}$$

biçiminde parçalara ayrılabilir.

2.3 Özdeğer Analizi

R, Mx1 boyutlu u(n) gözlem vektörü ile gösterilen durağan ayrık zamanlı bir rasgele sürece ait MxM boyutlu ilişki matrisi olsun. Genel olarak bu matris kompleks değerli elemanlar da içerebilir. Burada bulmak istediğimiz λ gibi bir sabit için,

$$\mathbf{Rq}=\lambda\mathbf{q} \tag{2.31}$$

koşulunu sağlayacak Mx1 boyutlu **q** vektörüdür. (2.31) eşitliği, **q** vektörünün **R** matrisi tarafından λ**q** vektörüne doğrusal dönüşümünü göstermektedir. λ sabit bir değer olduğu için yapılan bu doğrusal dönüşüm M boyutlu uzayda **q** vektörünün doğrultusunda herhangi bir değişikliğe neden olmamaktadır. MxM boyutlu tipik bir **R** matris için bu özellikte M adet **q** vektörü bulunabilir. Bunu göstermek amacıyla (2.31) eşitliği,

$$(\mathbf{R}-\lambda\mathbf{I})\mathbf{q}=\mathbf{0} \tag{2.32}$$

formunda yeniden yazılır. **I**=diag(1,1, . .,1) şeklinde tanımlı MxM boyutlu birim matris, **0** ise Mx1 boyutlu sıfır vektörüdür. (2.32) eşitliğinde **q** vektörünün sıfırdan farklı çözümünün olabilmesi için (**R**-λ**I**) matrisinin tekil matris olması yani

$$\det(\mathbf{R}-\lambda\mathbf{I})=\mathbf{0} \tag{2.33}$$

zorunludur. (2.33) eşitliği açıldığı zaman λ'nın M. dereceden bir polinom olduğu görülür. Diğer bir deyişle (2.33) eşitliğinin M adet kökü vardır. Bu durum karşısında (2.31) eşitliğinin de **q** vektörüne ait M adet çözümü olacaktır. Böylelikle MxM boyutlu tipik bir **R** matris için M adet **q** vektörünün bulunabileceği gösterilmiş olur. (2.33) eşitliği, **R** matrisinin karekteristik denklemi (characteristic equation) ve bu denklemin $\lambda_1, \lambda_2, \ldots, \lambda_M$ ile gösterilen M adet kökü ise **R** matrisinin özdeğerleri (eigenvalues) olarak adlandırılır. M adet özdeğerin olması bunların her zaman için birbirinden farklı değer alacağı anlamına gelmez. Eğer karekteristik denklemin katlı kökleri varsa, **R** ilişki matrisinin dejenere (degenerate) özdeğerlere sahip olduğu söylenir. Örnek olarak, beyaz gürültü sürecinin

$$\mathbf{R}=\text{diag}(\sigma^2\ \sigma^2 \ldots \sigma^2) \tag{2.34}$$

şeklinde tanımlanan MxM boyutlu **R** ilişki matrisi ele alınsın. σ^2, sürecin tek bir örneğinin değişintisini göstermektedir ve **R** ilişki matrisinin σ^2 varyansına eşit M katlı tek bir dejenere özdeğeri vardır. Bu nedenle Mx1 boyutlu rasgele herhangi bir **q** vektörü bu özdeğerin özvektörü (eigen vector) olarak nitelendirilebilir. Bu durum ise beyaz gürültünün rasgele bir yapıya sahip olduğunun kanıtıdır. (2x2) gibi basit boyutlu matrislerin dışında bir matrisin özdeğerlerini hesaplamak için o matrisin karakteristik denkleminin köklerini bulmak kaçınılması gereken zayıf yöntemler arasında yer almaktadır. Bu nedenle özdeğer hesabı için geliştirilmiş daha iyi yöntemlerden yararlanılmıştır. λ_i, **R** matrisine ait herhangi bir özdeğer ve $\mathbf{q}_i$ ise bu özdeğer için,

$$\mathbf{R}\mathbf{q}_i=\lambda_i\mathbf{q}_i \tag{2.35}$$

eşitliğinin sağlayan sıfırdan farklı bir vektör olsun. Bu durumda $\mathbf{q}_i$ vektörü λ_i özdeğerinin bir özvektörü olarak adlandırılır. Bir özvektör sadece tek bir özdeğere karşı gelebilirken, bir özdeğerin bir çok özvektörü olabilir. Örneğin, $\mathbf{q}_i$ vektörü λ_i özdeğerine ilişkin bir özvektör ise $\alpha\neq0$ şeklindeki herhangi bir skaler için $\alpha\mathbf{q}_i$ vektörü de λ_i özdeğerinin bir özvektörü olmaktadır. Bu nedenle özvektörlerin $\|\mathbf{q}_i\|=1$ ile gösterilen birim norma sahip olacak şekilde normalize edilmelidir. Vektörlerle gerçekleştirilen pek çok işlemde vektör büyüklüğü kullanılır. Euclidean normu, M boyutlu bir vektörün büyüklüğünü bulurken kullanılan norm çeşitlerinden biri olup,

$$\|x\| = \{\sum_{i=1}^{M}|x_i|^2\}^{1/2} \tag{2.36}$$

eşitliği ile hesaplanır. Geometrik açıdan Euclidean normu, o vektörün uzunluğuna eşit olmaktadır. Bu nedenle Euclidean normu, Euclidean uzunluğu olarak da adlandırılabilir.

$x^T = [\, x_1, x_2, \ldots, x_M]$ ve $y^T=[\, y_1, y_2, \ldots, y_M]$ gibi iki vektör arasındaki uzaklık

$$d(x,y) = \|x-y\| = \{\sum_{i=1}^{M}|x_i-y_i|^2\}^{1/2} \tag{2.37}$$

şeklinde elde edilir. Bu vektörlerin iç çarpımı ise,

$$\langle x,y\rangle = x^H y = \sum x_i^* y_i \qquad i=0,1,\ldots M-1 \tag{2.38}$$

eşitliği ile elde edilen skaler bir değerdir. İç çarpım, iki vektör arasındaki geometrik ilişkiyi tanımlar. Bu geometrik ilişki ise,

$$\langle x,y\rangle = \|x\|\|y\|\cos\theta \tag{2.39}$$

eşitliği ile verilir. θ iki vektör arasındaki açıdır. Buna göre, sıfırdan farklı x ve y vektörlerinin iç çarpımı sıfır $\langle x,y\rangle=0$ ise bu vektörlerin birbirine dik, orthogonal oldukları söylenir. Birim norma sahip ve birbirine dik olan vektörlere ise orthonormal denir. İç çarpımın kullanımlarından biri de, doğrusal ötelemeyle değişmeyen filtrelerin çıkışının elde edilmesidir. Örneğin, (M-1). dereceden bir FIR filtrenin birim örnek yanıtı h(n) ve girişide x(n) olsun. Filtre çıkışı y(n) ise, h(n) ve x(n)'in konvolusyonuyla (katlama) elde edilir:

$$y(n)=\sum h(k)x(n-k) \qquad k=0,1,\ldots M-1 \tag{2.40}$$

x(n) için $\mathbf{x}^T(n)=[x(n),x(n-1),..,x(n-M+1)]$ ve h(n) içinse $\mathbf{h}^T=[h(0),h(1),..,h(M-1)]$ vektörel formları kullanılarak (2.41) eşitliğinin bu iki vektörün iç çarpımı

$$y(n) = \mathbf{h}^T\mathbf{x} \tag{2.41}$$

şeklinde de yazılabilir ve **h** vektörlerinin gerçel değerlerden oluştuğu kabul edildiğinden iç çarpım ifadesinde Hermitian transpozu (H) yerine normal transpoz (T) kullanılmıştır.

2.3.1 Özdeğer ve özvektörlerin özellikleri

Özellik-1 $\lambda_1, \lambda_2,\ldots,\lambda_M$ değerleri **R** ilişki matrisinin özdeğerleri olsun. Her k>0 tamsayısı için $\mathbf{R}^k$ matrisinin özdeğerleri ise $\lambda^k_1, \lambda^k_2,\ldots, \lambda^k_M$ olur.

Özellik-2 $\lambda_1, \lambda_2,..,\lambda_M$ **R** ilişki matrisinin birbirinden farklı özdeğerleri olsun. $\mathbf{q}_1, \mathbf{q}_2,\ldots,\mathbf{q}_M$ özvektörleri olsun. Bu durumda $\mathbf{q}_1, \mathbf{q}_2,..,\mathbf{q}_M$ özvektörleri doğrusal bağımsızdır.

$\sum v_i \mathbf{q}_i = 0 \quad i=1,2,\ldots,M$ (2.42)

Özellik-3 Bir Hermitian matrisin farklı özdeğerlerine, $\lambda_1, \lambda_2, .., \lambda_M$, karşı düşen özvektörleri $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ birbirine dik olmaktadır. Başka bir deyişle,

$\lambda_i \neq \lambda_j$ ise $< \mathbf{q}_i, \mathbf{q}_j > = \mathbf{q}^H_i \mathbf{q}_j = 0$ (2.43)

olmaktadır.

Özellik-4 Pozitif özdeğerlere sahip Hermitian matrisler pozitif belirli, negatif olmayan özdeğerlere sahip Hermitian matrisler ise negatif olmayan olurlar. Dolaysiyle **R** ilişki matrisi, tüm özdeğerleri $\lambda_i > 0$ ise pozitif belirli, $\lambda_i \geq 0$ ise negatif olmayan belirlidir denir.

Özellik-5 Hermitian matrislerin tüm özdeğerleri gerçeldir. **R** ilişki matrisi de Hermitian olduğuna göre tüm özdeğerleri, $\lambda_1, \lambda_2, \ldots, \lambda_M$, gerçeldir.

Özellik-6 Birimsel benzerlik dönüşümü : $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ **R** matrisinin farklı $\lambda_1, \lambda_2, \ldots, \lambda_M$ özdeğerlerine karşılık düşen özdeğerler olsun. Ayrıca

$\mathbf{q}^H_i \mathbf{q}_j = \begin{cases} 1, i=j \\ 0, i=j \end{cases}$;

şeklinde otonormal vektörlerde oluşan MxM boyutlu $\mathbf{Q}=[\mathbf{q}_1 \ \mathbf{q}_2 .. \mathbf{q}_M]$ matrisi ile yine MxM boyutlu $\Lambda = \text{diag}(\lambda_1 \ \lambda_2 .. \lambda_M)$ diyagonal matris olsun. Bu durumda **R** ilişki matrisi $\mathbf{Q}^H\mathbf{R}\mathbf{Q} = \Lambda$ şeklinde diyagonalleştirilebilir.

Özellik-7 $\lambda_1, \lambda_2, .., \lambda_M$ MxM boyutlu **R** matrisinin özdeğerleri olsun. **R** matrisinin izi (trace) bu özdeğerlerin toplamına eşittir;

$\text{tr}[\mathbf{AB}] = \text{tr}[\mathbf{BA}]$ (2.44)

$\text{tr}[\mathbf{Q}^H\mathbf{R}\mathbf{Q}] = \text{tr}[\Lambda]$ $\Lambda = \text{diag}(\lambda_1 \ \lambda_2 .. \lambda_M)$ diyagonal matrisinin izi.

$tr[\Lambda]=\sum\lambda_i$ i=1,2,...,M

yazılırken, yukarıdaki eşitlikten dolayı,

$tr[\mathbf{R}]=\sum\lambda_i$ i=1,2,...,M

şeklinde yazılabilir.

Özellik-8 **R** ilişki matrisinin en küçük özdeğerinin (λ_{min}) en büyük özdeğerine (λ_{max}) oranı büyük ise **R** matrisine kötü durumlu (ill conditioned) denir. Kötü durum, matrisin bir ya da daha fazla özdeğerin sıfıra yakın değer almasında kaynaklanır. Herhangi bir **A** matrisinin kötü durumunu açıklayabilmek için Strang(1980) tarafından tanımlanmış olan koşul sayısı kullanılır.

$$\chi(\mathbf{A})=\|\mathbf{A}\|\|\mathbf{A}^{-1}\| \tag{2.45}$$

$\|\mathbf{A}\|$: **A** matrisinin , $\|\mathbf{A}^{-1}\|$: tersinin normudur.

$\|\mathbf{A}\|$ =($\mathbf{A}^H\mathbf{A}$ matris çarpımının en büyük özdeğeri)$^{1/2}$

R ilişki matrisi Hermitian olduğundan dolayı $\mathbf{R}^H\mathbf{R}=\mathbf{R}^2$ olacaktır.Eğer λ_{max} **R** matrisinin en büyük özdeğeri ise, $\mathbf{R}^2$ matrisinin en büyük özdeğeri de λ^2_{max} olacaktır. Bu durumda **R** matrisinin spektral formu,

$$\|\mathbf{R}\|_s=\lambda_{max} \tag{2.46}$$

olur. Öte yandan $\mathbf{R}^{-1}$ spektral formu,

$$\|\mathbf{R}^{-1}\|_s=1/\lambda_{min} \tag{2.47}$$

olmaktadır.Buradan hareketle **R** matrisinin koşul sayısı,

$$\chi(\mathbf{R})=\|\mathbf{R}\|\|\mathbf{R}^{-1}\|=\lambda_{max}//\lambda_{min} \tag{2.48}$$

$\chi(\mathbf{R})$ özdeğer yayılımı ya da özdeğer oranı olarak adlandırılır. Burada her zaman için $\chi(\mathbf{R})\geq 1$ koşulunun sağlandığı bilinmelidir.

2.4 Güç Spektral Yoğunluk

İlişki matrisi **R**, durağan ayrık rasgele süreçlerin ikinci derecdeki istatistiksel değerlerinin zaman domain'deki karşılığıdır. İlişki matrisinin elemanları olan r(n) (n=0,±1,±2,..,±(M-1) ilişki dizisine ayrık Fourier Transform uygularsak,

$$S(\omega)=\sum_{k=-(M-1)}^{M-1} r(k)e^{-j\omega k} \quad ,-\pi<\omega<\pi \tag{2.49}$$

elde edilir.ω açısal frekansı gösterir.

Yukarıda verilen ayrık zaman Fourier Transform ifadesi sözkonusu süreçlerin istatistiksel özelliklerinin frekans domainindeki gösterimidir. Ayrık zaman Fourier Transformunun tersinin alınması, tekrar zaman domainindeki ilişki dizisinin verir,

$$r(k)=(1/2\pi)\int_{-\pi}^{\pi} s(\omega)e^{j\omega k}d\omega \quad , \qquad k=0,\pm1,\pm2,..,\pm(M-1) \tag{2.50}$$

Bir filtrenin H(z) ayrık transfer fonksiyonuyla karekterize edilsin. Filtre girişi güç spektral yoğunluğu S(ω) olan ayrık zaman rasgele süreç olsun. $S_0(\omega)$ ise filtrenin çıkış güç spektral yoğunluğu olsun. Bu durumda,

$$S_0(\omega)=|H(e^{j\omega})|^2 S(\omega) \tag{2.51}$$

Olur. $H(e^{j\omega})$ filtrenin frekans yanıtıdır. Burada çıkarılacak sonuç, ω açısal frekansındaki çıkış güç spektral yoğunluk aynı açısal frekanstaki filterin yanıtının büyüklüğü ile filtrenin giriş güç spektral yoğunluğunun çarpımına eşittir.

3 WIENER FİLTRELER

Uyarlamalı dengeleyicilerin çalışma ilkesini oluşturan yinelemeli algoritmaların çıkarılması için, temel olarak birbirinden farklı yöntemler vardır : Wiener Filtre Teorisi, Kalman Filtre Teorisi vb.. Burada konumuzla doğrudan ilişkili olan Wiener Filtreleme yöntemi ele alınmaktadır.

Doğrusal filtreleme, Kolmogorov ve Wiener tarafından yeniden ele alınmış ve formüle edilmiştir. Kolmogorov ayrık zamanlı süreçler üzerinde çalışıp **Wold ayrışmasını (Wold decomposition)** problemin çözümü için önerirken, öte yandan Wiener sürekli zaman problemi üzerinde çalışmış ve çözüm için ünlü **Wiener-Hopf integral** denklemini önermiştir.Bu denklem, sinyalin ilşki matrisi bilgisine gereksinme duyar. Dolaysiyle basit problemler için denklemin çözümü kolay olmasına karşın, karmaşık problemler için oldukça zordur. Sözkonusu integral denkleminin ayrık zaman düzleminde karşılığı **normal denklem**dir.Wiener-Hopf integral denkleminin sürekli zamanda çözümü ile normal denklemin ayrık zamanda çözümü **Wiener filtre** olarak bilinir.

Bu bölümde kompleks değerli zaman serilerinin genel durumu için, Wiener filtrelerin ayrık zaman formülü üzerinde filtrenin impuls yanıtına göre çalışılacaktır. Sözü edilen serileri seçmenin nedeni, bir çok pratik alanda (radar,sonar), anaband sinyalinin kompleks formda olmasıdır.

Yukarda filtrelerin doğrusal ve ayrık zamanlı olmak üzere iki özelliğinden söz edildi. Filtrenin doğrusal olması problemin çözümünü kolaylaştırırken, ayrık zamanlı olması ise sayısal devrelerde kullanılmasını sağlar. Öte yandan filtre tasarımında, impul yanıtının sonlu ya da sonsuz süreli mi olacağı gibi bir durumla karşılaşılabilinir. Literatürde bunlar **FIR** (sonlu süreli impuls yanıtlı) ve **IIR** (sonsuz süreli impuls yanıtlı) filtreler olarak adlandırılır. Buradaki çalışmada **sonlu süreli impuls yanıtlı (FIR)** filtreler üzerinde çalışılacak. Bunun nedeni FIR filtrelerin kararlı olmalarıdır. Sonlu süreli impuls yanıtlı demek, giriş ile çıkış arasındaki ilişkinin ileri yönde yani girişten çıkışa doğru olmasıdır.

Öte yandan sonsuz süreli impuls yanıtlı (IIR) filtrelerde ise hem ileri yönde hem de geri yönde besleme vardır. Eğer uyum bu biçimde kontrol edilmez ise filtrede bulunan geri besleme filtrenin kararsız davranmasına ve sonucun salınımına yol açar. IIR filtrelerin

hem teorik hem de pratik açıdan kendi kendilerine kararlılık sorununun üstesinde geldikleri kabul edilir. Ancak filtrenin uyarlamalı olması istendiği zaman IIR filtrelerinin yapısında bulunan geribesleme çözümü karmaşık olan problemdir. Dolaysiyle uyarlamalı süreç durumlarında, FIR filtreler IIR filtrelere tercih edilir.

Bir filtrenin doğrusal olabilmesi için filtre çıkışının filtre girişine uygulanmış gözlem değerlerinin doğrusal bir fonksiyon olması gerekir.

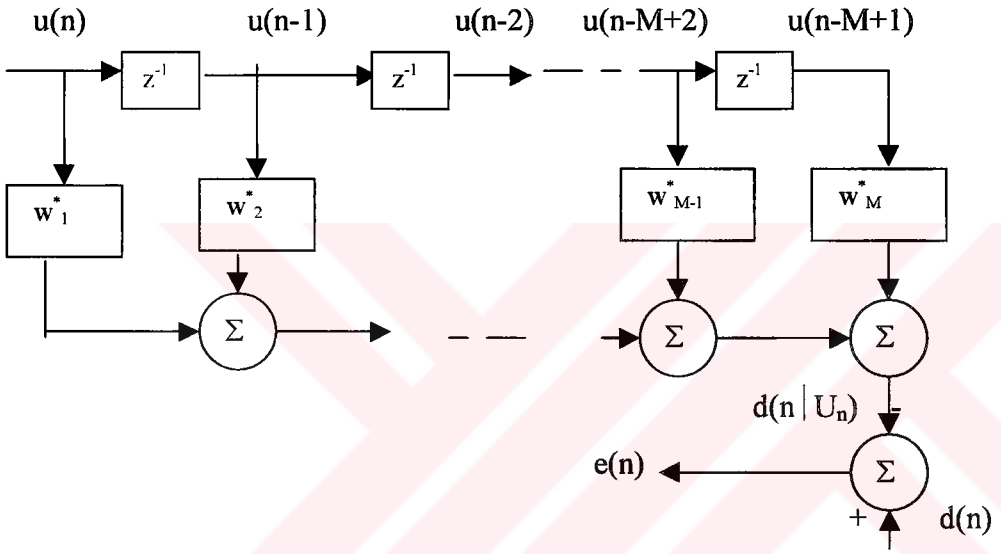
Doğrusal fitreleme probleminin çözümünde yararlı işaret ve istenmeyen toplamsal gürültüye ait beklendik değer ve ilişki fonksiyonları gibi bazı istatistiksel parametrelerin önceden bilinmesi gerekir. Bu bilgi tam olarak bilinmedikçe Wiener filtreyi tasarlamak ya da tasarlansa bile optimum sonuç elde etmek olanaksızdır. Çünkü filtre ancak girişine gelen işaretin istatistiği kendisinin tasarımında kullanılan ön bilgiye uyar ise optimum sonuç verir. Bu durumda geriye kalan tek şey girişindeki gürültülü işareti alıp bir takım istatistiksel kriterlere göre gürültünün etkisini minimum yaparak çıkışına ileten optimum doğrusal filtreyi tasarlamaktır. Bu filtre optimizasyonu sorununda yararlı bir yaklaşım da, filtre çıkışı ile istenen işaret arasındaki fark olarak tanımlanan hatanın karesel beklendik değerinin minimum yapılmasıdır. Böyle bir yaklaşımda durağan girişler için elde edilecek sonuç, karesel ortalama bakımından optimum olan Wiener filtre'dir.

Durağan olmayan girişler için Wiener filtre optimum çözüm değildir, dolaysiyle bu tür girişler için kullanılmaması gerekir. Kullanılsa bile karesel ortalama hata için optimum sonuç vermez. Bu nedenle durağan olmayan, diğer bir deyişle zamanla değişen (time-varying) girişler için optimum filtrenin de zamanla değişen yapıda olması gerekmektedir. Bu soruna en iyi ve en yaygın çözüm ise Kalman filtre ile bulunur. Aslında Wiener filtre Kalman filtrenin özel bir durumudur.

3.1 Optimum Filtreleme Problemi

Doğrusal enine bir filtre, yani bir FIR filtre ele alalım (Şekil 3.1). Bu filtre sırasıyla depolama (storage) , çarpma (multiplication) ve toplama (addition) olmak üzere üç temel işlem gerçekleştirir. Depolama işlemi, z^{-1} ile tanımlanmış tek bir örneklik gecikme sağlayan kaskad bağlı (M-1) adet birim gecikme elemanları ile yapılır. u(n), u(n-1),...,u(n-M+1), birim gecikme elemanlarının girişleri olup tap girişleri olarak adlandırılır. u(n)

filtrenin şu andaki giriş değerini, geriye kalan (M-1) adet tap girişi ise geçmiş giriş değerlerini belirtir. Filtredeki gecikme elemanlarının sayısı (buradaki filtre için bu sayı M-1), filtrenin derecesini belirtir. u(n),u(n-1),....,u(n-M+1) tap girişlerinin sırasıyla w_1, w_2 ....,w_M tap ağırlıkları ile iç çarpımına karşılık gelen w_1^*, w_2^*,...,w_M^* şeklinde blok gösterimli çarpıcılar tarafından gerçekleştirilir. Örneğin, u(n) tap girişi, w_1^* blok gösterimli çarpcının girişidir. Toplayıcıların görevi ise çarpıcı çıkışlarını toplayarak filtrenin çıkışlarını oluşturmaktır. Böylece enine filtrenin impuls yanıtının $\{h_k\}=\{w_k^*\}$, k=1,2,..,M şeklinde tap ağırlıkları ile ifade edilir.



Şekil 3.1 Doğrusal enine filtre blok diyagramı (Haykin, 1991)

u(n), u(n-1),...., u(n-M+1) tap girişleri ile sunulan ayrık zamanlı rasgele sürecin geniş anlamda durağan olduğu kabul edilir. Aynı zamanda yapılan genel kabullerde rasgele sürecin ortalama (beklendik) değeri de sıfır kabul edilebilir. n anında filtre çıkışında üretilen işaret $\hat{d}(n|U_n)$ ile gösterilmektedir.Burada amaç, filtre çıkışını istenen yanıtın, d(n)'nin kestirimi olarak kullanılmasıdır. Bu nedenle (^) işareti filtre çıkışı ile istenen yanıtı birbirinden ayırt etmek için kullanılmıştır. Filtre çıkışının gösteriminde kullanılan $(n|U_n)$ terimindeki n, bu çıkışın n anındaki çıkış olduğunu ve U_n ise n anı da dahil olmak üzere n anına kadar olan u(n) , u(n-1) , ..., u(n-M+1) tap girişlerinin oluşturduğu M boyutlu

uzayı gösterir. Filtre çıkışı, filtre girişleri ile filtrenin impuls yanıtı arasında doğrusal konvolüsyon toplamı olarak,

$$\hat{d}(n|U_n)=\sum w_k^* u(n-k+1) \quad ,,k=1,2,\ldots,M \tag{3.1}$$

bulunur. Şekil 3.1'de verilen filtrenin, istenen yanıt d(n) ile çıkışı arasındaki farkı, belli bir istatistiksel kriter gözönüne alınarak mümkün olduğunca küçük yapabilecek biçimde tasarlanması ile kestirim problemine bir çözüm bulunmuş olur. İstenen yanıtın n anındaki örnek değeri ile n anındaki filtre çıkışı arasındaki farka,

$$e(n)=d(n)-\hat{d}(n|U_n) \tag{3.2}$$

kestirim hatası denir. Wiener teorisinde, minimum karesel ortalama hata kriteri optimum filtreyi tasarlamak için kullanılır. Tap ağırlıkları $w_1,w_2,..,w_M$ kestirim hatasının karesel ortalaması ya da karesel ortalama hata olarak adlandırılan,

$$J(\mathbf{w}) = E[e(n)\, e^*(n)] \tag{3.3}$$

biçimindeki eşitliğin değerini enaz yapacak biçimde seçilir.

Burada **w** vektörü, $w_1,w_2,..,w_M$ elemanlarını içeren tap ağırlık vektörüdür. Karesel ortalama hata her zaman için gerçel ve pozitif bir değerdir. **J(w)** başarım göstergesi minimum yapılarak minimum karesel ortalama anlamında elde edilebilecek en iyi filtreye ya da diğer bir ifade ile optimum filtreye ulaşılır.

3.2 Hata Başarım Yüzeyi

Eşitliklerin(denklemlerin) matris formunu kullanmak, Wiener filtre teorisinin ayrık zamanlı versiyonunu geliştirmede kolaylık sağlayacaktır. Filtrenin tap ağırlıklarının oluşturduğu M x 1 boyutlu tap ağırlık vektörü,

$$\mathbf{W}^T=[w_1,w_2,..,w_M] \tag{3.4}$$

şeklinde olurken n anındaki M x 1 boyutlu u(n) tap ağırlıkları giriş vektörü ise,

$$\mathbf{u}^T = [u(n), u(n-1), u(n-2), .., u(n-M+1)] \quad (3.5)$$

şeklinde gösterilmektedir.Öyleyse (3.1) eşitliği bu iki vektörün iç çarpımı olarak,

$$\hat{d}(n|U_n) = \mathbf{w}^H \mathbf{u}(n) \quad (3.6)$$

biçiminde yazılabilir. Her iki tarafın Hermitian transpozu alınırsa,

$$\hat{d}^*(n|U_n) = \mathbf{u}^H(n)\mathbf{w} \quad (3.7)$$

elde edilir. Böylece (3.2) eşitliği ile verilen kestirim hatası ve kompleks eşleniği sırasıyla,

$$e(n) = d(n) - \mathbf{w}^H \mathbf{u}(n) \quad (3.8)$$

$$e^*(n) = d^*(n) - \mathbf{u}^H(n)\mathbf{w} \quad (3.9)$$

vektörel formda yazılabilir. (3.8) ve (3.9) eşitlikleri (3.3) eşitliğinde yerine konulduğunda karesel ortalama hata,

$$J(\mathbf{w}) = E[(d(n) - \mathbf{w}^H \mathbf{u}(n))(d^*(n) - \mathbf{u}^H(n)\mathbf{w})] \quad (3.10)$$

haline gelir. (3.10) eşitliğinde **w** tap ağırlık vektörünün zamanla değişmeyen sabit bir vektördür. Bu da gözönüne alınarak eşitliğin sağ tarafı yeniden düzenlenirse,

$$J(\mathbf{w}) = E[d(n)d^*(n)] - \mathbf{w}^H E[\mathbf{u}(n)d^*(n)] - E[d(n)\mathbf{u}^H(n)\mathbf{w}] + \mathbf{w}^H E[\mathbf{u}(n)\mathbf{u}^H(n)]\mathbf{w} \quad (3.11)$$

elde edilir. Tap giriş vektörü **u**(n) ile d(n) istenen yanıtının birleşik durağan olduğu varsayılarak (3.11) eşitliğinde bulunan dört adet beklendik değer terimi şu şekilde yorumlanabilir;

i. İstenen yanıt d(n), sıfır ortalamalı kabul edildiğinde E[d(n)d*(n)] beklendik değeri istenen yanıtın değişintisine eşit olur.

$$\sigma^2_d=E[d(n)d^*(n)] \tag{3.12}$$

ii. E[**u**(n)d*(n)] beklendik diğeri tap giriş vektörü **u**(n) ile d(n) istenen yanıtı arasındaki Mx1 boyutlu çapraz ilişki vektörüdür ve **P** ile gösterilir.

$$\mathbf{P}=E[\mathbf{u}(n)d^*(n)] \tag{3.13}$$

$$\mathbf{P}^T=[p(0),p(-1),\ldots,p(1-M)] \tag{3.14}$$

$$p(1-k)=E[u(n-k+1)d^*(n)] \quad , \quad k=1,2,\ldots..,M \tag{3.15}$$

iii. E[d*(n)$\mathbf{u}^H$(n)] beklendik değeri, **P** vektörünün Hermitian transpozualınmış biçimidir,

$$\mathbf{P}^H=E[d(n)\mathbf{u}^H(n)] \tag{3.16}$$

iv.E[**u**(n)$\mathbf{u}^H$(n)] beklendik değerleri 2.Bölümde açıklandığı gibi u(n) tap giriş vektörünün MxM boyutlu ilişki matrisine eşittir.Yani,

$$\mathbf{R}=E[\mathbf{u}(n)\mathbf{u}^H(n)] \tag{3.17}$$

olur. Açık formu ise,

$$\mathbf{R}=\begin{bmatrix} r(0) & r(1) & \Lambda & r(M-1) \\ r(-1) & r(0) & \Lambda & r(M-2) \\ M & M & O & M \\ r(-M+1) & r(-M+2) & \Lambda & r(0) \end{bmatrix} \tag{3.18}$$

olur.Satır sayısı k ve sütun sayısı i ile gösterildiğinde k,i=1,2,..,M olmak üzere **R** ilişki matrisinin (k,i) elemanı,

$$r(i-k)=E[u(n-k+1)u^*(n-i+1)] \tag{3.19}$$

ifadesi ile tanımlanır. Burada **R** matrisinin Hermitian özelliğinden dolayı,

$$r^*(i-k)=r(k-i) \tag{3.20}$$

olur. Öyleyse, durağan bir sürecin MxM boyutlu **R** ilişki matrisi girişin özilişki fonksiyonunun aldığı r(0),r(1),..,r(M-1) değerleri ile tanımlanabilir.

(3.12), (3.13), (3.16), ve (3.17) eşitliklerini (3.11) eşitliğinde yerlerine yazarak karesel ortalama hata için daha kısa bir ifade

$$J(\mathbf{w})=\sigma^2_d-\mathbf{P}^H\mathbf{w}-\mathbf{w}^H\mathbf{P}+\mathbf{w}^H\mathbf{R}\mathbf{w} \tag{3.21}$$

elde edilmiş olur. (3.21) eşitliğindeki $\mathbf{w}^H\mathbf{R}\mathbf{w}$ terimi, tap giriş vektörü **u**(n) ile istenen yanıt d(n) birleşik durağan olduğu zaman karesel ortalama hatanın, **w** tap ağırlık vektörünün ikinci dereceden bir fonksiyonu olduğunu ortaya koyar. Bu nedenle karesel ortalama hatanın tap ağırlık vektörünün elemanlarına olan bağımlılığı tek bir minimum noktası olan (M+1) boyutlu kase şekilli (bowl–shaped) bir yüzey olarak düşünülebilir. Bu yüzeye enine filtrenin hata başarım yüzeyi adı verilir. Bu durumda gerek duyulan tek şey hata başarım yüzeyinin minimum noktasında ya da diğer bir deyişle dibinde çalışabilecek bir filrte tasarlamaktır.Hata başarım yüzeyinin minimum noktasında karesel ortalama hata J(**w**), J_{min} ile gösterilen en küçük değerini alırken, bu hata değerine karşı düşen **w** tap ağırlık vektörü de $\mathbf{w}_o$ ile gösterilen optimum değerini alır. Bu şekilde elde edilen enine filtreye ise karesel ortalama hata anlamında optimum filtre denir. $\mathbf{w}_o$ optimum tap ağırlık vektörünü elde edebilmek için (3.21) eşitliği ile verilen karesel ortalama hatanın, **w** tap ağırlık vektörüne göre türevi alınarak sıfıra eşitlenir. Elde edilen sonuç optimum tap ağırlık vektörü $\mathbf{w}_o$ olur. Fakat bundan önce skaler değerli bir fonksiyonun (J(**w**)) kompleks değerli bir vektöre (**w**) göre türevinin nasıl alınacağının bilinmesi gerekir.

3.3 Normal Denklemin Bulunması

(3.21) eşitliğinde, σ^2_d sabit bir değer olduğu için **w** vektörüne göre türevi de sıfır olur. Sözkonusu eşitliğin geriye kalan üç terimin **w** vektörüne göre türevleri ise,

$$d(\mathbf{P}^H\mathbf{w})/d\mathbf{w}=\mathbf{0}\ ,\ d(\mathbf{w}^H\mathbf{P})/d\mathbf{w}=\mathbf{0} \text{ ve } d(\mathbf{w}^H\mathbf{R}\mathbf{w})/d\mathbf{w}=2\mathbf{R}\mathbf{w}$$

olmaktadır. Böylece J(**w**) karesel ortalama hatasının **w** tap ağırlık vektörüne göre türevi olan ∇ gradyen vektörü,

$$\nabla(J(\mathbf{w}))=d(J(\mathbf{w}))/d\mathbf{w}=-2\mathbf{P}+2\mathbf{R}\mathbf{w} \tag{3.22}$$

olur. $\mathbf{w}_o$ gradyen vektörü sıfır vektör yapan optimum tap ağırlık vektörünü göstersin. Bu durumda (3.22) eşitliğinden,

$$\mathbf{R}\mathbf{w}_o=\mathbf{P} \tag{3.23}$$

sonucu elde edilir. (3.23) eşitliği Wiener-Hopf integral denklemlerinin ayrık zamandaki halidir ve normal denklem olarak adlandırılır. (3.23) eşitliğinin çözümü $\mathbf{w}_o$ optimum tap ağırlık vektörünü verir. Bu çözüm için eşitlik sol taraftan $\mathbf{R}^{-1}$ ters matrisi ile çarpılır.

$$\mathbf{w}_o=\mathbf{R}^{-1}\mathbf{P} \tag{3.24}$$

(3.24) eşitliğine göre optimum $\mathbf{w}_o$ tap ağırlık vektörünü bulabilmek için, **u**(n) tap giriş vektörünün **R** ilişki matrisi ve **u**(n) ile d(n) istenen yanıtı arasındaki **P** çapraz ilişki vektörü gibi iki istatistiksel parametrenin önceden bilinmesi gerekir. (3.4), (3.15) ve (3.17) eşitlikleri (3.24) eşitliğinden yerlerine yazılarak, eşitliğin açık formu,

$$\begin{bmatrix} r(0) & r(1) & \Lambda & r(M-1) \\ r(-1) & r(0) & \Lambda & r(M-2) \\ M & M & O & M \\ r(-M+1) & r(-M+2) & \Lambda & r(0) \end{bmatrix} \begin{bmatrix} w_{01} \\ w_{02} \\ M \\ w_{03} \end{bmatrix} = \begin{bmatrix} p(0) \\ p(-1) \\ M \\ p(1-M) \end{bmatrix} \tag{3.25}$$

olup, bu açık gösterimin,

$$\Sigma wr(i-k)=p(1-k)\ ,,i=1,2\ldots,M,,\ k=1,2\ldots,M \tag{3.26}$$

gibi bir başka eşdeğer formu da yazılabilir.

3.4 Diklik Kuralı

(3.24) eşitliği, minimum karesel ortalama anlamında optimum olan bir enine filtrenin $\mathbf{w}_o$ tap ağırlık vektörünü tanımlar. Bu eşitlik, (3.13) ve (3.17) eşitlikleri kullanılarak,

$$E\mathbf{u}(n)\mathbf{u}^H(n)]\mathbf{w}_o=E[\mathbf{u}(n)d^*(n)] \tag{3.27}$$

şeklinde yazılabilir. Daha önce belirtildiği gibi, $\mathbf{w}_o$ vektörü zamana bağlı olmayan sabit değerler içeren bir vektör olduğundan beklendik değer operatörünün içine alınabilir. (3.27) eşitliği yeniden düzenlenirse,

$$E\,[\mathbf{u}(n)(d^*(n)-\mathbf{u}^H(n)\,\mathbf{w}_o)]=\mathbf{0} \tag{3.28}$$

olur. (3.9) eşitliğinden yararlanılarak optimum fitreye ait kestirim hatası $e_o(n)$,

$$e_o^*(n)=d^*(n)-\mathbf{u}(n)\mathbf{w}_o \tag{3.29}$$

biçiminde yazılır. Öyleyse (3.28) eşitliği,

$$E[\mathbf{u}(n)e_o^*(n)]=\mathbf{0} \tag{3.30}$$

olarak yazılabilir. Burada **0**, Mx1 boyutlu sıfır vektörüdür.

(3.30) eşitliği şöyle yorumlanabilir; bir enine filtre optimum koşulda çalıştığı zaman filtrenin $e_o(n)$ kestirim hatası ile $\mathbf{u}(n)$ tap giriş vektörünün her bir elemanı birbirine dik (orthogonal) olmaktadır. Bu sonuç ise diklik kuralı (principle of orthogonality) olarak bilinir. Bu da şunu ortaya koymaktadır; minimum karesel ortalama hata kriteri ile girişler ve kestrim hatası arasındaki diklik kriteri birbirleriyle özdeş sonuçlar verir. Diğer bir deyişle, iki kriterden birinin gerçekleşmesi ile elde edilecek sonuca eşdeğer olacaktır. Optimum filtrenin bir başka yararlı özelliği de (3.31) eşitliğinin optimum tap ağırlık vektörünün Hermitian transpozu ile sol taraftan çarpılması ile elde edilir.

$$\mathbf{w}_o^H E[\mathbf{u}(n)e_o^*(n)]=\mathbf{0} \tag{3.31}$$

Tap ağırlık vektörü beklendik değer operatörünün içine alınırsa,

$$E[\mathbf{w}_o^H \mathbf{u}(n) e_o^*(n)] = \mathbf{0} \tag{3.32}$$

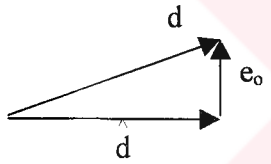
elde edilir ve u(n) tap girişine karşı optimum filtrenin çıkışı,

$$\hat{d}(n|U_n) = \mathbf{w}_o^H \mathbf{u}(n) \tag{3.33}$$

olduğu gözönünde bulundurularak (3.30) eşitliği,

$$E[\hat{d}(n|U_n) e_o^*(n)] = \mathbf{0} \tag{3.34}$$

biçiminde yazılabilir. Bu eşitlik de göstermektedir ki; bir enine filtre optimum koşulda çalıştığı zaman filtrenin $e_o(n)$ kestirim hatası ile filtrenin çıkışındaki $\hat{d}(n|U_n)$ kestirimi de birbirine dik olmaktadır. (3.34) eşitliği optimum filtrenin çıkışında var olan koşullara ait ilginç bir geometrik yorumun (Şekil 3.2) yapılmasını sağlar.



Şekil 3.2 İstenen yanıt,filtre kestirimi ve filtre kestrim hatası arasındaki ilişki

Şekilde istenen yanıt d, filtre çıkışı $\hat{d}$ ve bu ikisine karşı gelen kestirim hatası ise e_o ile gösterilmektedir. Görüldüğü gibi optimum filtrenin kestirim hatasını gösteren vektör bu filtrenin çıkışına normal (dik) olmaktadır ve işte bu nedenle optimum filtreyi tanımlayan (3.23) eşitliğine normal denklem denir.

3.5 Minimum Karesel Ortalama Hata

Herhangi bir **w** tap ağırlık vektörü için karesel ortalama hatayı elde ederken (3.21) eşitliği kullanılır. Tap ağırlık vektörü optimum değerini $\mathbf{w}_o$ alırsa, karesel ortalama hata J_{min} ile gösterilen minimum değerini alır. Dolaysiyle (3.21) eşitliği,

$$J_{min} = \sigma_d^2 - \mathbf{P}^H \mathbf{w}_o - \mathbf{w}_o^H \mathbf{P} + \mathbf{w}_o^H \mathbf{R} \mathbf{w}_o \tag{3.35}$$

olur. (3.23) ile verilen normal denklemi (3.35) eşitliğinde yerine koyarsak,

$J_{min}= \sigma^2_d - \mathbf{P}^H \mathbf{w}_0$

elde edilir. (3.24) eşitliğini kullanırsak,

$$J_{min}=\sigma^2_d - \mathbf{P}^H \mathbf{R}^{-1} \mathbf{P} \quad (3.37)$$

eşitliği elde edilir. Bu eşitliğin matrissiz biçimi,

$$J_{min} = \sigma_d^2 + \sum w_{0k}\, p^*(k-1) \quad (3.38)$$

olmaktadır .Minimum karesel ortalama hata J_{min} için bir başka yararlı ifade ise,

$$d(n)=\hat{d}(n|U_n)+e_o(n) \quad (3.39)$$

eşitliğinin her iki tarafının karesel beklendik değeri alınarak bulunur.

$$E[d(n)d^*(n)]=E[(\hat{d}(n|U_n)+e_o(n))(\hat{d}(n|U_n)+e_o(n))^*] \quad (3.40)$$

Optimum filtre için, filtre çıkışı $\hat{d}(n|U_n)$ ile kestirim hatası $e_o(n)$ birbirine dik (orthogonal) olduğu için (3.40) eşitliğinde,

$$E[\hat{d}^*(n|U_n)e_0(n)]=E[e_o^*(n)\hat{d}(n|U_n)]=0 \quad (3.41)$$

değerleri yerlerine yazılarak,

$$E[|d(n)|^2]=[|\hat{d}^*(n|U_n)|^2]+E[|e_o(n)|^2] \quad (3.42)$$

$$\sigma^2_d=\sigma^2_d+J_{min} \quad (3.43)$$

eşitliği elde edilir. Burada d(n) istenen yanıtı ile u(n) tap girişi sıfır ortalamalı kabul edilmiştir. (3.41) eşitliğinin minimum karesel ortalama hata için çözümü,

$$J_{min}=\sigma^2_d-\sigma^2_d \tag{3.44}$$

şeklinde olur. Bu eşitlikten optimum filtre için minimum karesel ortalama hatanın istenen yanıtın değişintisi ile filtre kestirimi arasındaki farka eşit olduğu görülür.(3.44) eşitliğinin karesel ortalama hatanın alacağı minimum değerin daima sıfır ile bir arasında değişecek şekilde normalize edilmesi bazı açıklamaların yapılmasında kolaylık sağlayacaktır. Bu nedenle eşitliğin her iki tarafı istenen yanıtın değişintisine bölünerek,

$$J_{min}/\sigma^2_d=1-\sigma^2_d/\sigma^2_d \tag{3.45}$$

bu normalizasyon gerçekleştirilir. σ_d^2 bir tek özel durum dışında asla sıfır olamayacağı için, eşitliğin her iki tarafının istenen yanıtın değişintisine bölünmesinde herhangi bir sakınca yoktur. Bu özel durum ise d(n) istenen yanıtının her n değeri için sıfır olması ile ortaya çıkar. (3.45) eşitliğinin sol tarafındaki bölüm normalize karesel ortalama hata olarak adlandırılan ε ile gösterilirse bu eşitlik,

$$\varepsilon=1-\sigma^2_d/\sigma^2_d \tag{3.46}$$

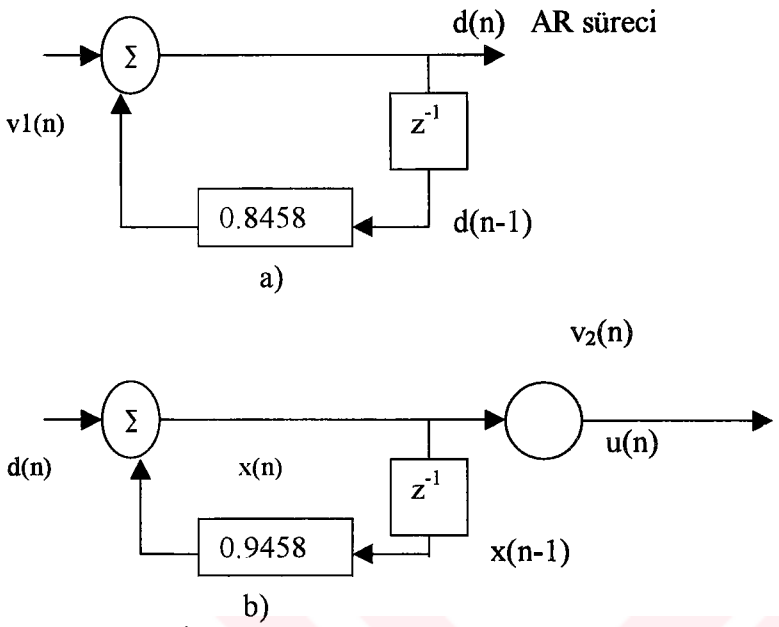
şeklinde yazılır. (3.46)'daki σ^2_d/σ^2_d bölümü daima birden küçük ve pozitif değerler aldığından normalize karesel ortalama hatanın alacağı değerler yukarıda söylendiği gibi,

$$0 \le \varepsilon \le 1 \tag{3.47}$$

sınırları arasında olacaktır. Eğer ε sıfıra eşit ise optimum filtre çıkıştaki $\hat{d}(n|U_n)$ kestrimi ile d(n) istenen yanıtı arasında tam bir uyum sağlayabilme bakımından mükemmel çalışıyor demektir. Öte yandan, ε bire eşit ise bu optimum filtre için en kötü durumdur, çünkü bu durumda $\hat{d}(n|U_n)$ ile d(n) arasında hiçbir uyum yok demektir. Filtredeki tap ağırlıklarının sayısı (Şekil 3.1'de M adet vardır.) arttıkça herhangi bir d(n) istenen yanıtı için optimum enine filtrenin göstereceği başarım da artmaktadır. Filtrenin başarımı arttıkça da ε değeri azalacaktır.

3.6 Sayısal Örnek

Buraya dek anlatılanlarla ilgili Şekil 3.3'te verilen sayısal örnek üzerinde çalışılacaktır.



Şekil 3.3 a) İstenen yanıtın AR süreci modeli.
b) Gürültülü kanalın modeli

Beyaz gürültü süreçleri $v_1(n)$ ve $v_2(n)$ sıfır ortalamalı olduğu, aralarında ilişki olmadığı ve değişintilerinin $\sigma^2_1=0.27$ ile $\sigma^2_2=0.1$ kabul edilecektir. Ayrıca d(n) ve u(n) dolaysiyle $v_1(n)$ ve $v_2(n)$ gerçel değerlidir. (3.48) eşitliğinden $\sigma^2_d=0.9486$ bulunur ($a_1=0.8458$).

$$\sigma^2_d= \sigma^2_1/(1-a^2) \qquad (3.48)$$

$x(n)+c_1x(n-1)+c_2x(n-2)=v(n)$ eşitliğinden yola çıkarak,

$$\mathbf{R}=\begin{bmatrix}1.1 & 0.5\\ 0.5 & 1.1\end{bmatrix},\quad \mathbf{P}=\begin{bmatrix}0.5772\\ -0.4458\end{bmatrix},\quad \mathbf{w}_0=\begin{bmatrix}0.8360\\ -0.7853\end{bmatrix},\quad J_{min}=0.1579 \text{ bulunur.}$$

3.7 Hata Başarım Yüzeyinin Kanonik Formu

(3.20) eşitliğinden (3.37) eşitliği çıkarılarak elde edilen sonuçtan **P** vektörü normal denklem kullanılarak yok edilir.Bu işlem sonucunda karesel ortalama hata,

$$J(w) = J_{min} + (\mathbf{w} - \mathbf{w}_o)^H \mathbf{R}(\mathbf{w} - \mathbf{w}_o) \quad (3.49)$$

şeklinde kuadratik formda yazılmış olur. Eşitlikten de görülebileceği gibi J(w) 'yi minimum yapabilen sadece tek bir tap ağırlık vektörü $\mathbf{w}_o$ vardır. Bu ise $\mathbf{w}_o$ vektörünün dışında başka bir optimal vektör olmadığı anlamına gelir. Yeterince açıklayıcı olmasına karşın hata başarım yüzeyinin gösterimini daha da basitleştirebilmek amacıyla (3.49) eşitliğinin tabanında bir dönüşüm yapılacaktır. Bu dönüşüm için,

$$\mathbf{R} = \mathbf{Q}\,\Lambda\,\mathbf{Q}^H \quad (3.50)$$

eşitliğinden yararlanılacaktır. Bu durumda (3.49) eşitliğinin aldığı yeni durum,

$$J(\mathbf{w}) = J_{min} + (\mathbf{w} - \mathbf{w}_o)^H \mathbf{Q}\Lambda\mathbf{Q}^H(\mathbf{w} - \mathbf{w}_o) \quad (3.51)$$

olur. Tap ağırlık vektörü **w** ile optimum çözüm olan $\mathbf{w}_o$ vektörü arasındaki fark

$$\nu = \mathbf{Q}^H(\mathbf{w} - \mathbf{w}_o) \quad (3.52)$$

şeklinde bir ν vektörüne dönüştürülsün. Bu durumda (3.48) eşitliğinin kuadratik formu

$$J(\nu) = J_{min} + \nu^H \Lambda \nu \quad (3.53)$$

şeklinde tanımlanan kanonik forma dönüşür. ν vektörünün k. elemanı ν_k ile gösterilmek üzere (3.53) eşitliğinin açık formu yazıldığında,

$$J(\nu) = J_{min} + \sum_{k=1}^{M} \lambda_k \nu_k \nu_k^* \quad , \qquad J(\nu) = J_{min} + \sum_{k=1}^{M} \lambda_k |\nu_k|^2 \quad (3.54)$$

w vektöründen ν wektörüne yapılan taban dönüşümü ile elde edilen kanonik formdaki karesel ortalama hata formülünün kuadretik formdaki karesel ortalama hata formülüne oranla daha basit olduğu görülür. Hata başarım yüzeyinin gösterilmesinde kanonik formu daha uygun kılan özellik, ν dönüşüm vektörünün elemanlarının hata başarım yüzeyinin ana eksenlerini (principal axes) oluşturuyor olmasıdır

4 UYARLAMALI FİLTRE ALGORİTMALARI

4.1 Giriş

Uyarlamalı filtre işlevi, sürekli ayarlanan ağırlık katsayıları (w), bunların giriş dizisi ile çarpımından elde edilen filtre çıkışı ($\hat{d}(n|U_n)$ ya da y(n)) ve istenen yanıt (d(n)) ile kestirilen değer arasındaki fark ile kontrol edilen bir uyarlamalı algoritmadan oluşur. Ağırlık katsayılarını hesaplamak için, daha önce belirtildiği gibi Wiener-Hopf normal eşitliği kullanılır. Bu eşitlik, giriş dizisine ait ilişki matrisinin tersine gereksinir. Ancak, her giriş örneği için matris tersi almak, zaman alıcı bir işlemdir. Bu nedenle ağırlık katsayılarının hesaplanması için değişik algoritmalar geliştirilmiştir. Bunların bazıları, **En Dik İniş Algoritması (Steepest Descent), En Küçük Kareler Algoritması** ya da **En Küçük Karesel Ortalama Algoritması (LMS), Yinelemeli En Küçük Kareler (RMS)** ve **Yinelemeli En Küçük Kareler Kafes algoritması (RLSL).**

En Dik İniş Algoritması, hata başarım yüzeyinin minimum noktasını, yüzey hakkında bilgiye gereksinmeksizin bulan algoritmadır. Şekil 4.1, M dereceden bir uyarlamalı filitre yapısını göstermektedir. Burada $w_1(n), w_2(n), ..., w_M(n)$ ayarlanabilen ağırlık katsayılarıdır.

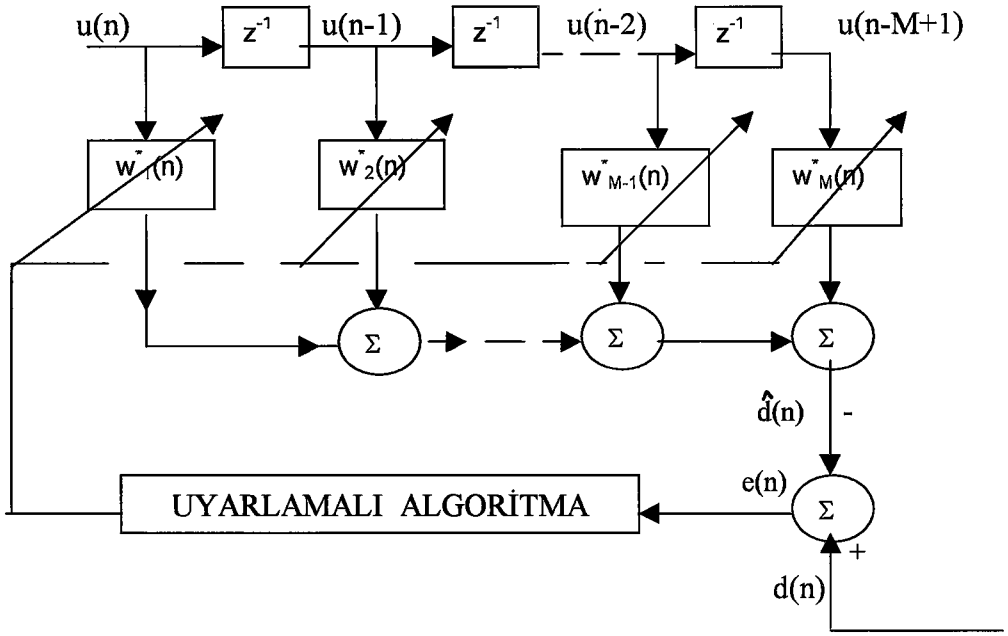
Doğrusal birleştirici çıkışındaki $\hat{d}(n)$ kestirimi, istenen d(n) yanıtı ile karşılaştırılarak e(n) hatası üretilir.

$$e(n)=d(n)-\hat{d}(n)$$

$$e(n)=d(n)-\mathbf{w}^H(n)\mathbf{u}(n) \qquad (4.1)$$

$\mathbf{w}^H(n)$ $\mathbf{u}(n)$ terimi, kompleks değerli $\mathbf{w}(n)$ ve $\mathbf{u}(n)$ vektörlerinin iç çarpımını, "H" ise Hermitian dönüşümünü göstermektedir. $\mathbf{u}(n)$ giriş dizisi ve d(n) istenen yanıtı uzun bir aralıkta durağan iseler karesel ortalama hata (hata başarım yüzeyi) $J(\mathbf{w})$, şöyle yazılabilir:

$$J(\mathbf{w})=\sigma^2_d-\mathbf{w}^H(n)\mathbf{P}-\mathbf{P}^H\mathbf{w}(n)+\mathbf{w}^H(n)\mathbf{R}\mathbf{w}(n) \qquad (4.2)$$



Şekil-4-1 Uyarlamalı enine filtre (FIR) yapısı.

J(**w**) hata başarım yüzeyi, n anında **w**(n) ağırlık vektörünün ikinci dereceden fonksiyonu olup tek bir minimum noktaya sahiptir. Daha öncede belirtildiği gibi(4.2) eşitliğinde $\sigma^2_d(n)$ arzu edilen yanıtın varyansını; **P**, **u**(n) giriş dizisi ile d(n) yanıtı arasındaki çapraz ilişki vektörünü; **R**, **u**(n) giriş dizisine ait ilişki matrisini göstermektedir. Uyarlamalı işlemde de Wiener filitre gibi J(**w**) performans yüzeyinin, minimum noktasında elde edilen optimum $\mathbf{w}_o$ vektörü, her giriş örneği için,

$$\mathbf{w}_o=\mathbf{R}^{-1}\mathbf{P} \qquad (4.3)$$

biçimindeki normal denklem için yeniden hesaplanacaktır.

4.2 En Dik İniş Algoritması

Optimum ağırlık katsayıları vektörü (4.3) eşitliği ile bulunabilir. Fakat bu yöntemin $\mathbf{R}^{-1}$' den dolayı işlem uzunluğu sakıncası vardır. Bu yönteme alternatif olarak en dik iniş algoritması kullanılabilir. Bu algoritma ile, karesel ortalama hatayı minimum yapan $\mathbf{w}_o$ vektörü şöyle bulunur: Önce **w** vektörüne bir başlangıç değeri verilir. Bu **w**(0) vektörü kullanılarak J(**w**)

başarım yüzeyi oluşturulur ve bunun gradyen vektörü hesaplanır. **w** vektöründe, gradyen vektörüyle ters yönde bir adım değişiklik yapılıp hesap yinelenir.

∇, n anındaki gradyen vektörü ve **w**(n) n anındaki ağırlık vektörü olmak üzere, en dik iniş algoritması ile (n+1) anındaki ağırlık vektörü,

$$\mathbf{w}(n+1)=\mathbf{w}(n)+(1/2)\mu[-\nabla] \tag{4.4}$$

şeklinde bulunabilir. Burada μ adım uzunluğu olup, pozitif gerçel bir sabittir. Gradyen vektörü,

$$\nabla=d(J(\mathbf{w}))/d(\mathbf{w}(n))$$

$$=-2\mathbf{P}+2\mathbf{R}\mathbf{w}(n) \tag{4.5}$$

olarak elde edilir. **R** giriş özilişki matrisinin ve **P** çapraz ilişki vektörünün bilindiği kabul edilip, (4.5) bağıntısı (4.4) 'de yerine konularak,

$$\mathbf{w}(n+1)=\mathbf{w}(n)+\mu[\mathbf{P}-\mathbf{R}\mathbf{w}(n)] \tag{4.6}$$

bulunur. μ katsayısının, her adım sırasında **w**(n) vektöründe yapılan düzeltmeyi kontrol ettiği görülmektedir. (4.6) eşitliği, en dik iniş algoritmasının matematiksel gösterimidir. Bu algoritma geri beslemeli olduğu için karasızlığa geçebilir. Algoritmanın karasızlığını, geri besleme kolundaki μ adım uzunluğu ve **u**(n) giriş vektörüne ait **R** ilişki matrisi belirler. n anında hesaplanan ağırlık vektöründeki hata şöyle tanımlanabilir:

$$\mathbf{c}(n)=\mathbf{w}(n)-\mathbf{w}_o \tag{4.7}$$

Buradaki $\mathbf{w}_o$, (4.3)'deki normal eşitlikten çözülen optimum ağırlık vektörü, **w**(n) ise n anında tahmin edilecek ağırlık vektörü ve hatası, (4.3) ve (4.6) eşitlikleri kullanılarak,

$$\mathbf{w}(n+1)=\mathbf{w}(n)[\mathbf{I}-\mu\mathbf{R}]+\mu\mathbf{R}\mathbf{w}_o \tag{4.8}$$

$\mathbf{c}(n+1)=\mathbf{w}(n+1)-\mathbf{w}_o$

$=[\mathbf{I}-\mu\mathbf{R}]\mathbf{w}(n)+[\mu\mathbf{R}-\mathbf{I}]\mathbf{w}_o$

$=[\mathbf{I}-\mu\mathbf{R}][\mathbf{w}(n)-\mathbf{w}_o$

$=[\mathbf{I}-\mu\mathbf{R}]\mathbf{c}(n)$ (4.9)

bulunur. Burada **I**, M boyutlu birim matristir. Benzerlik dönüşümü kullanılarak **R** matrisini,

$\mathbf{R}=\mathbf{Q}\Lambda\mathbf{Q}^H$ (4.10)

(4.9) eşitliğinde yerine koyarak,

$\mathbf{c}(n+1)=(\mathbf{I}-\mu\mathbf{Q}\Lambda\mathbf{Q}^H)\mathbf{c}(n)$ (4.11)

elde edilir. (4.11) eşitliğinin her iki tarafi $\mathbf{Q}^H$ ile çarpılır ve **Q** matrisinin ortogonal olduğu düşünülürse,

$\mathbf{Q}^H\mathbf{c}(n+1)=(\mathbf{I}-\mu\Lambda)\mathbf{Q}^H\mathbf{c}(n)$ (4.12)

eşitliği ortaya çıkar. Yeni bir koordinat takımı şöyle tanımlanabilir:

$\mathbf{v}(n)=\mathbf{Q}^H\mathbf{c}(n)$

$=\mathbf{Q}^H[\mathbf{w}(n)-\mathbf{w}_o]$ (4.13)

(4.9) eşitliğine benzer şekilde,

$\mathbf{v}(n+1)=(\mathbf{I}-\mu\Lambda)\mathbf{v}(n)$ (4.14)

yazılabilir. **v**(n)'nin başlangıç değeri,

$$\mathbf{v}(0)=\mathbf{Q}^H[\mathbf{w}(0)-\mathbf{w}_o] \quad (4.15)$$

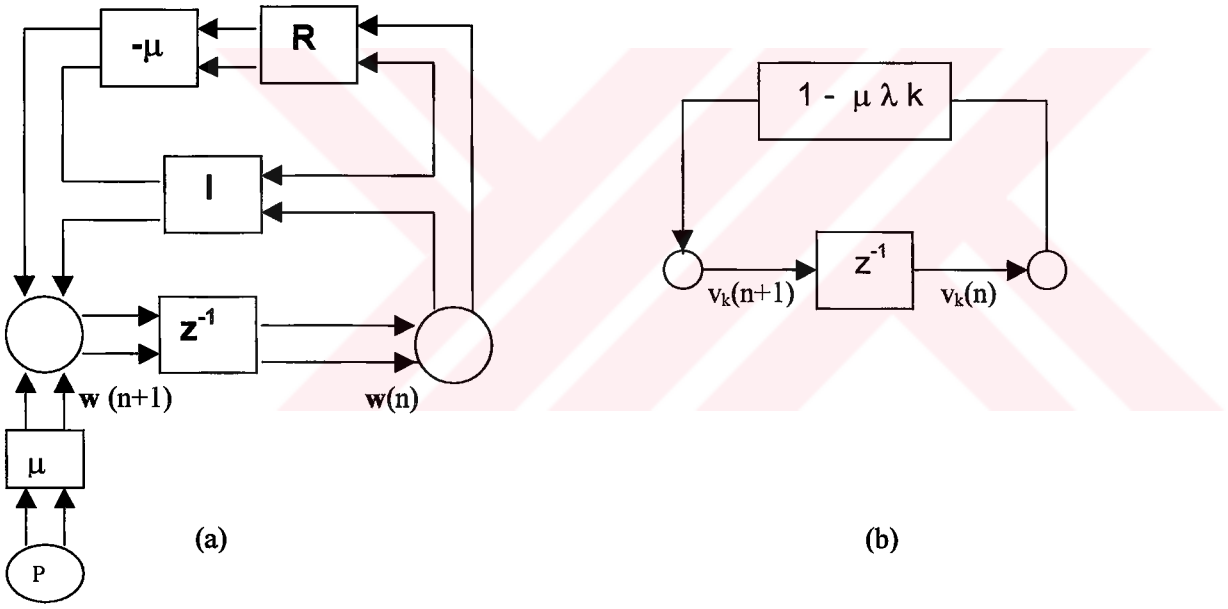
olup $\mathbf{w}(0)$ başlangıç vektötörü sıfir kabul edilerek,

$$\mathbf{v}(0)=-\mathbf{Q}^H[\mathbf{w}(0)-\mathbf{w}_o] \quad (4.16)$$

bulunur. en dik iniş algoritmasının k. doğal modunun matematiksel gösterilimi,

$$v_k(n+1)=(1-\mu\lambda k)v_k(n) \quad (4.17)$$

şeklindedir. Buradaki λ_k, **R** özilişki matrisinin k. özdeğeridir. (4.6) eşitliğinde matris formundaki koordinat sistemi (4.17)'de skaler hale indirgenmiştir. Şekil 4.2 en dik iniş algoritmasının matris ve skaler formlarının akış şemasını göstermektedir.



Şekil 4-2 En dik iniş algoritması
a) Matris formunda işaret akış şeması.
b) k. doğal modunun işaret akış şeması.

(4.17) eşitliği, birinci dereceden bir fark denklemidir ve çözümü,

$$v_k(n)=(1-\mu\lambda_k)^n v_k(0) \quad , \quad k=1,2,....,M \quad (4.18)$$

şeklinde yazılabilir. **R** matrisinin özdeğerleri pozitif gerçel olduklarından, $v_k(n)$ yanıtları osilasyona girmeyecektir. $v_k(n)$ sayıları, $(1-\mu\lambda_k)$ geometrik oranı ile geometrik bir seri oluşturur. Endik iniş algoritmasının kararlı olması için, bu geometrik oranın genliği 1 'den küçük olmalıdır.

$-1< 1-\mu\lambda<1 \quad , \quad k = 1,2, ...,M$

$0<\mu<2/\lambda_{max}$ (4.19)

λ_{max} , **R** özilişki matrisinin en büyük özdeğeridir. Sözü edilen geometrik serinin elemanlarını birleştiren zarf üstel olarak,

$1-\mu\lambda k=e^{-1/Tk}$ (4.20)

şeklinde düşünülebilir. k. zaman sabiti T_k, adım uzunluğu μ ve k. özdeğer λ_k cinsinden,

$T_k=-1/(\ln(1- \mu\lambda k)$ eşitliğinden μ çok küçük kabul edilrse, T_k

$T_k=1/(\mu\lambda k)$ (4.21)

olarak ifade edilebilir. T_k , k . doğal mod olan $v_k(n)$ 'in başlangıç değerinin , 1/e ' sine düşmesi için gereken süredir . Gerçek ağırlık vektörü **w**(n) , (4.13) eşitliği kullanılarak,

$\mathbf{w}(n) = \mathbf{w}_o + \mathbf{Q}\mathbf{v}(n)$ (4.22)

$w(n)= \mathbf{w}_o+\sum q_k v_k(n)$,, $k=1,2,\ldots,M$

bulunur. Daha önce de belirtildiği gibi q_k, **R** özilişkiu matrisnin λ_k özdeğerine ait özvektörüdür. i. ağırlık katsayısı, (4.18) ve (4.22) eşitlikleri birlikte kulanılarak,

$$\mathbf{w}_i(n)=\mathbf{w}_{oi}+\sum_{k=1}^{M} q_{ki} v_k(0)(1-\mu\lambda_k)^n \qquad (4.23)$$

şeklinde elde edilir (i = 1, 2, ..., M) . w_{oi} , i. ağırlığının optimum değeri, q_{ki} k. özvektörün i. bileşenidir. (4.23) eşitliği dik iniş algoritması, bütün ağırlık katsayılarının (1- μλk)^n üstel ifadesinin ağırlıklandırılmış toplamı ile elde edileceğini göstermektedir. Buradaki her üstel terim T_k süresine gerek duyduğu için, (4.23)' deki toplam ifadesinin, başlangıç değerinin 1/e'sine düşmesi için gereken T_a toplam zaman sabiti denebilir.

En düşük yaklaşma hızı q_{ki} $v_k(0)$ çarpanının yanlızca λ_{max} için sıfırdan farklı olduğu,

$$-1/(\ln(1-\mu\lambda_{max} \leq Ta \leq -1/(\ln(1-\mu\lambda_{max}) \qquad (4.24)$$

durumda elde edilir. Toplam zaman sabiti bu hızlar ile sınırlıdır. λ_{max}, **R** özilişki matrisinin en büyük, λ_{min} ise en küçük özdeğeridir. En dik iniş algoritmasının, karesel ortalama hatası şöyle verilir;

$$J(w)=J_{min}+\sum|v_k(n)|^2\lambda_k \ , \ k=1,2,\ldots,M \qquad (4.25)$$

(4.18) eşitliğini, (4.25) 'de yerine koyarak,

$$J(w)=J_{min}+\sum|v_k(0)|^2\lambda_k(1 - \mu\lambda_k)^{2n} \ , \ k=1,2,\ldots,M \qquad (4.26)$$

elde edilir. En dik iniş algoritmasının yakınsak olması için, μ adım uzunluğu (4.19) koşulunu sağlayacak şekilde seçilmelidir. Bu durumda,

$$\lim_{n \to \infty} J(\mathbf{w})=J_{min} \qquad (4.28)$$

olur. J(**w**) 'in, n'ye bağlı olarak çizilen değişimine Öğrenme Eğrisi, J(w(0)) başlangıç değerinden J_{min} minimum değerine erişmesi için geçen süreye, Öğrenme süresi denir ve T_{kj} ile gösterilir,

$$\cong 1/2\mu\lambda_k \ , \ \mu<<1$$

$T_{kj} = -1(\ln(1-\mu\lambda_k)$

Kısaca:

Bu yöntemle ortalama karesel hatanın minimum değeri J_{min}' i bulmak için :

1. Keyfi olarak seçilen, katsayı vektörü başlangıç değeri **w**(0) ile başlanacaktır. Bu **w**(0) değeri, hata-başarım yüzeyinin minimum noktasının nerede olabileceği gibi bir başlangıç kestirimi yapar. Genellikle **w**(0), sıfır vektöre eşit seçilir.

2. Bu başlangıç değeri ya da şimdiki kestirim kullanılarak, gradyen vektörü hesaplanır. Bu vektör, n anında katsayı vektörü **w**(n) e ait ortalama karesel hata J(**w**) nin gradyeni olarak tanımlanır.

3. Katsayı vektörünün bir sonraki kestirimi hesaplanır.

4. İkinci adıma geri dönülerek işlem yinelenir.

Katsayı vektörünün negatif yönünde ardarda düzeltilmesi sonucu optimum değeri varsayılan $\mathbf{w}_o$ noktasına ulaşıldığında, karesel ortalama hatanın da minimum değeri J_{min}'e yakınsayacağı düşünülür. $\nabla(n)$ gradyen vektörünün n anındaki değerini (kısaca ∇), **w**(n) ise katsayı vektörünün n anındaki değerini göstersin. Katsayı vektörünün n + 1 anındaki güncellenmiş değeri aşağıdaki basit rekürsif ilişki kullanılarak hesaplanabilir.

$\mathbf{w}(n+1)=\mathbf{w}(n)+((1/2)\mu[-\nabla]$

μ pozitif gerçek değerli bir sabittir.

4.3 En Küçük Karesel Ortalama Algoritması (LMS)

En dik iniş algoritması, her iterasyonda gradyen vektörünün kesin değerinin bilinmesine ve μ adım uzunluğunun uygun seçilmesine gerek duyar. Bu koşullarda en dik iniş algoritması

optimum Wiener çözümüne yakınsar.Ancak, gradyen vektörünün gerçek değerini ölçmek değil, eldeki verilerden kestirmek olasıdır.

Bu nedenle, Widrow ve Hopf tarafından (1960) farklı bir algoritma geliştirilmiştir. En Küçük Karesel Ortalama (Least Mean Square) adı verilen bu algoritmanın en önemli üstünlükleri şunlardır: Basittir, ilişki fonksiyonları ve matris tersi hesaplanmasına gereklilik göstermez. **u**(n) giriş dizisi ve d(n) istenen yanıtının çok kısa süreler içinde ortalama değeri, kendisine eşit olacaktır. $\underline{\mathbf{w}}$(n) anlık (intantaneous) değer olarak düşünelim,

$$\underline{\mathbf{R}}(n)=\mathbf{u}(n)\mathbf{u}^{H}(n) \tag{4.29}$$

$$\underline{\mathbf{P}}=\mathbf{u}(n)d^{*}(n) \tag{4.30}$$

$\underline{\mathbf{R}}$ özilişki matrisi ve $\underline{\mathbf{P}}$ çapraz ilişki vektörünün bu yaklaşımla kestirilen anlık değerleri (4.5) eşitliğinde yerine konursa,

$$\underline{\nabla}=2\mathbf{u}(n)d^{*}(n)+2\mathbf{u}(n)u^{H}(n)\underline{\mathbf{w}}(n) \tag{4.31}$$

elde edilir. Gradyen vektörünün bu anlık kestirim değeri, (4.6) 'daki en dik iniş algoritmasında yerine konarak,

$$\underline{\mathbf{w}}(n+1)=\underline{\mathbf{w}}(n)+\mu\mathbf{u}(n)[d^{*}(n)-\mathbf{u}^{H}(n)\underline{\mathbf{w}}(n)] \tag{4.32}$$

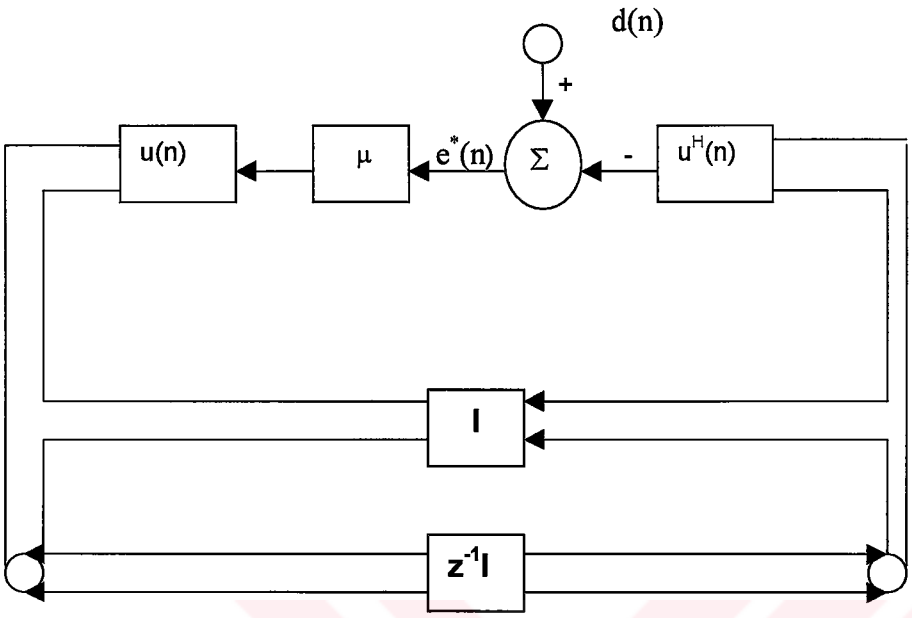
bulunur.

$$e(n)=d(n)-\underline{\mathbf{w}}(n)\mathbf{u}(n) \tag{4.33}$$

dikkate alınarak,

$$\underline{\mathbf{w}}(n+1)=\underline{\mathbf{w}}(n)+\mu\mathbf{u}(n)e^{*}(n) \tag{4.34}$$

yazılabilir. (4.34) eşitliği kompleks formda, En Küçük Karelesel Ortalama (LMS) Algoritması olarak bilinir. İterasyona $\underline{\mathbf{w}}(0)$ başlangıç değeri ile girilir ki, genellikle $\underline{\mathbf{w}}(0)=\underline{\mathbf{0}}$'dır. LMS algoritmasının işaret akış şeması Şekil 4.3' de gösterilmiştir.



Şekil 4.3 LMS algoritmasının akış şeması.

n anında kestirilen ağırlık vektörü ile optimum Wiener çözümü arasındaki farka, ağırlık vektörü hatası denir ve,

$$\varepsilon(n)=\underline{\mathbf{w}}(n)-\mathbf{w}_o \tag{4.35}$$

olarak tanımlanır. n sonsuza yaklaştığında $\varepsilon(n)=\mathbf{0}$ olacaktır. LMS algoritması gradyen vektörü ∇'in kesin değeri yerine, kestirimini ($\underline{\nabla}$) kullandığı için $\mathbf{w}_o$ optimum vektörüne oldukça fazla iterasyondan sonra, çevresinde salınarak erişebilir. Sonsuz sayıda iterasyondan sonra dahi, en küçük karesel ortalama algoritmasının karesel ortalama hatası ya da başarım yüzeyi $J(\underline{\mathbf{w}})$, J_{min}'den, erişim hatası $J_{ex}(\underline{\mathbf{w}})$ kadar fazla olacaktır.

$$J(\underline{\mathbf{w}})=J_{min}+(\underline{\mathbf{w}}(n)-\mathbf{w}_o)^H\underline{\mathbf{R}}(\underline{\mathbf{w}}(n)-\mathbf{w}_o) \tag{4.36}$$

$$J_{ex}(\underline{\mathbf{w}})=(\underline{\mathbf{w}}(n)-\mathbf{w}_o)^H\underline{\mathbf{R}}(\underline{\mathbf{w}}(n)-\mathbf{w}_o) \tag{4.37}$$

$J_{ex}(\underline{\mathbf{w}})=J(\mathbf{w})-J_{min}$ (4.38)

LMS algoritmasının yakınsaklığı ya da kararlılığı aşağıdaki koşula bağlıdır,

$0<\mu<2/\lambda_{max}$ (4.39)

λ_{max}, $\underline{\mathbf{R}}$ özilişki matrisinin en büyük özdeğeridir. Ancak, özilişki matrisinin özdeğerleri genellikle bilinmezler. Bunun yanında, özilişki matrisinin özdeğerlerinin toplamı bulunabilir. Daha önce verilen bu eşitlikleri yeniden anımsayalım,

$\lambda_{max}=\sum\lambda_i(i=1,2,\ldots,M)$ (4.40)

olduğundan (4.39) yerine,

$0<\mu<2/\sum\lambda_I$ (4.41)

kullanılabilir. (4.41) koşulu, (4.40)'ı sağlamaktadır. **R** özilişki matrisinin özdeğerleri,

$\sum\lambda_i=tr[\mathbf{R}]$

$\sum\lambda_i=ME[\mathbf{u}(n)\mathbf{u}^*(n)]$ (4.42)

$\sum\lambda_i=Mr(0)$ (4.43)

özelliğine sahiptir. Sonuç olarak, $\underline{\mathbf{w}}$ ağırlık vektörünün, $\mathbf{w}_o$ optimum ağırlık vektörüne yakınsayabilmesi için,

$0<\mu<2/$toplam giriş gücü (4.44)

koşulu sağlanmalıdır.

4.3.1 Temel varsayımlar

LMS algoritmasının, matematiksel olarak kolayca işlenebilir istatistiksel analizini yapmak için, aşağıdaki dört koşulu içeren temel varsayımlar kullanılacaktır:

1. Giriş işleminin her örnek vektörü u(n), önceki tüm örnek vektörleri u(k), k=0,1,...,n-1' den istatiksel bağımsızdır.

$E[\mathbf{u}(n)\mathbf{u}^H(k)]=\mathbf{0}$, k=0,1,2,31,...,n-1

2. Giriş işleminin her örnek vektörü u(n), önceki tüm istenen yanıt örnekleri d(k), k=0,1,..,n-1'den istatiksel bağımsızdır.

$E[\mathbf{u}(n)d^*(k)]=\mathbf{0}$, k=0,1,2,3.....,n-1

3. İstenen yanıt örneği d(n), giriş işleminin uygun örnek vektörü **u**(n) e bağımlıdır ama, istenen yanıtın önceki tüm örneklernden istatiksel bağımsızdır.

4. Giriş vektörü **u**(n) ve istenen yanıt d(n), her n için karşılıklı Gauss-dağılımlı rasgele değişkenler içerir.

Bu temel varsayımlara dayalı LMS algoritmasının istatiksel analizi Bağımsız Teori diye adlandırılır.

(3.6) eşitliğinden, (**P** ve **R** değerleri yerine konursa) n+1 anında sadece şu üç girişe bağlı katsayı vektörü $\underline{\mathbf{w}}(n+1)$ gözlensin:

1. Giriş işleminin önceki örnek vektörleri, **u**(n), **u**(n-1),...,**u**(0).

2. İstenen yanıtın önceki örnekleri, d(n),d(n-1),...,d(0).

3.Katsayı vektörünün başlangıç değeri, $\underline{\mathbf{w}}(0)$.

Varsayımlardaki 1 ve 2 ye göre $\underline{\mathbf{w}}(n+1)$ katsayı vektörü, **u**(n+1) ve d(n+1) den bağımsızdır.

Katsayı vektörünün kestiriminin ilk iki anını çıkarmak için temel varsayımlar kullanılır;

(i) ortalama değer ve (ii) ilişki matrisi. Katsayı vektörünün kendisinden çok ağırlık-hata vektörü ile çalışmak daha uygun olacaktır.

$\varepsilon(n)=\underline{\mathbf{w}}(n)-\mathbf{w}_o$

$\mathbf{w}_o$ katsayısı vektörü için optimum Wiener çözümü, $\underline{\mathbf{w}}(n)$ n anında LMS algoritması ile elde edilen kestirimi gösterir.

Kısaca LMS Algoritması,

Parametreler :
M = Katsayı adedi ya da filtre derecesi
μ= Adım-uzunluğu parametresi

Başlangıç Koşulları :

$\underline{\mathbf{w}}(0) = \mathbf{0}$

Veri:
(i) Verilen : u(n) = n anındaki Mx1 boyutlu giriş vektörü
d(n) = n anındaki istenen yanıt
(ii) Hesaplanacak olan: $\underline{\mathbf{w}}(n+1)$: n+1 anındaki katsayı vektörünün kestirimi (n = 0,1,2, ...) :

$e(n)=d(n)-\underline{\mathbf{w}}^H(n)\mathbf{u}(n)$

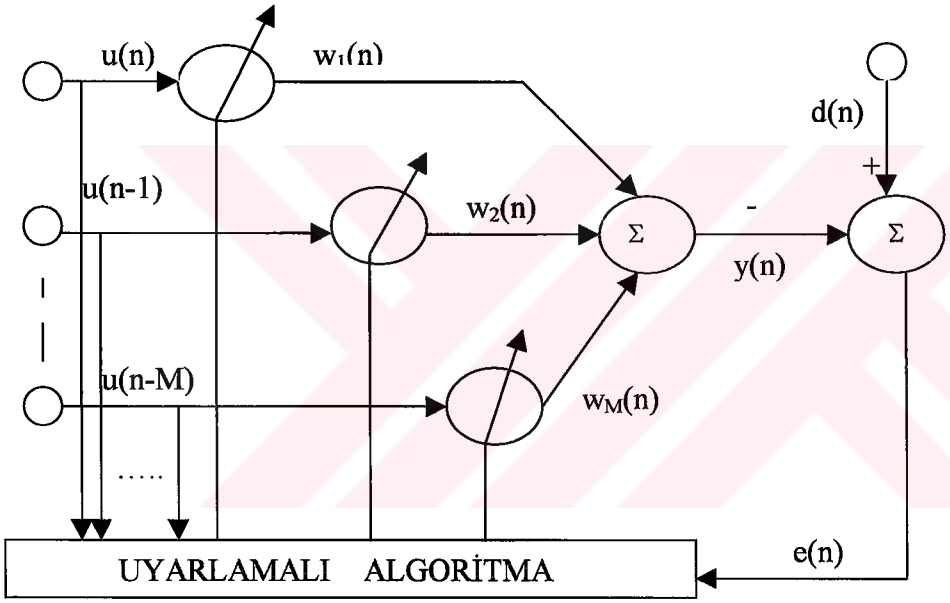
$\underline{\mathbf{w}}(n+1) = \underline{\mathbf{w}}(n)+\mu\mathbf{u}(n)e^*(n)$

Uyarı :Katsayı vektörünün başlangıç değeri $\underline{\mathbf{w}}(0)$, sıfıra eşit alındığında ve ayrıca enine filtrenin başlangıçtaki bütün girişleri sıfır alındığında ikinci ifadeden $\underline{\mathbf{w}}(1)$'in de sıfır olacağı görülmektedir. Buna göre LMS algoritmasının ilk iterasyonundan sonra daima e(1)=d(1) bulunur.

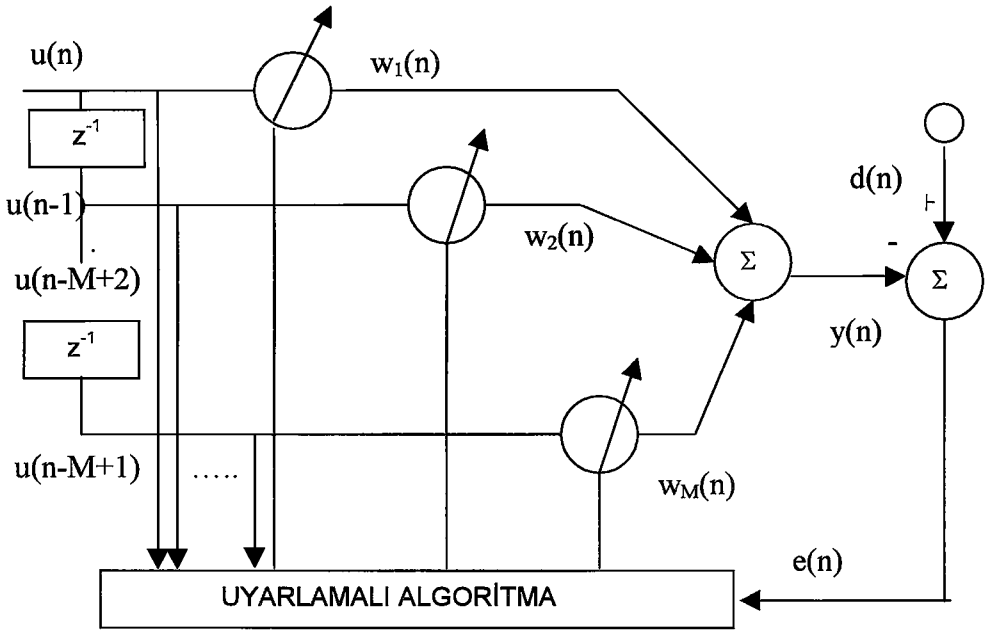
4.4 Uyarlamalı Filtre Uygulamaları

4.4.1 Uyarlamalı doğrusal birleştirici

u(n),u(n-1),...,u(n-M+1) referans girişler olup, uygun ağırlıklar ile çarpılarak istenen bir çıkış(yanıt) dizisini üretebilirler. {u(n)} dizisi ayrı ayrı uygulanabileceği gibi, herhangi bir işaretten birim gecikme elemanları kullanılarak da elde edilebilir. d(n) istenen yanıt dizisi, y(n) ise onun, uyarlamalı doğrusal birleştirici tarafından kestirilen değeridir. Aslında uyarlamalı doğrusal birleştirici ile d(n) dizisi üretebilmek için y(n) yanıtına gerek yoktur.



(a)

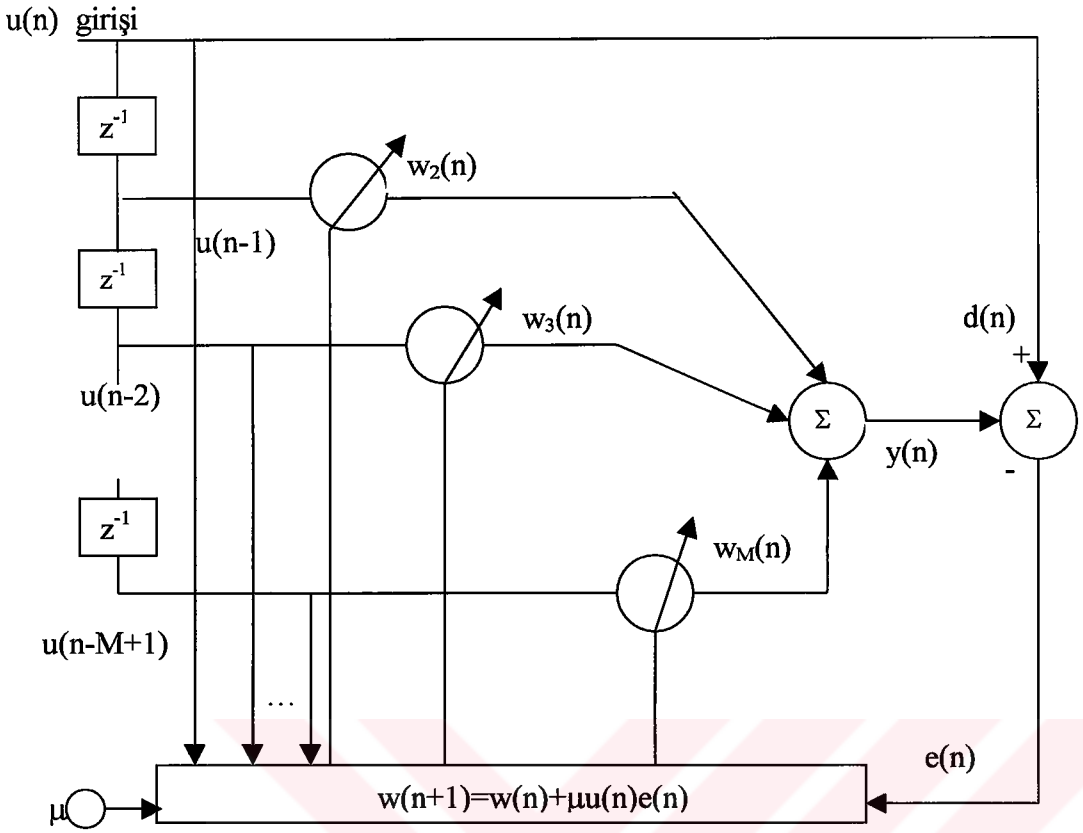


(b)

Şekil 4.4 a) Çok girişli doğrusal birleştirici
b) Doğrusal birleştirici

4.4.2 LMS uyarlamalı filtre

Şekil 4.5 'de verilen LMS uyarlamalı filitrede referans girişler u(n) primer girişinden, birim gecikme elemanları ile elde edilir. Bu filtre, durağan olmayan işaretler için, ağırlık katsayılarını kestirmek ve izlemek için kullanılır. LMS uyarlamalı filtre bölüm 4.4.3.'de anlatılacak olan uyarlamalı gürültü azaltıcıda, gürültü kestirimini elde etmek için kullanılacaktır.



Şekil 4.5 LMS uyarlamalı filtre.

5 UYARLAMALI KANAL DENGELEME VE UYGULAMALARI

Bilinmeyen bir ortamda oldukça verimli çalışabilmek ve de giriş istatistiklerinin zamanla değişimini izleyebilmek, uyarlamalı filtreleri, işaret-işleme ve kontrol uygulamaları için vazgeçilmez hale getiren özelliklerdir. Uygulamalı filtreler ile haberleşme, kontrol, radar, sonar, deprem bilimi, görüntü işleme ve biomedikal mühendislik gibi farklı alanlarda başarılı uygulamalar gerçekleştirilmiştir. Nitelik olarak birbirlerinden farklı olmalarına karşın, bu uygulamaların tek ortak özelliği; bir giriş vektörüne ve bir istenen yanıta sahip olmalarıdır. Bu iki değer sayesinde, filtrenin ayarlanabilir katsayılarının alacağı değerleri kontrol etmeye yarayan kestrim hatası hesaplanır. Ayarlanabilir katsayılar, kullanılan filtrenin yapısına göre, tap ağırlıkları (enine filtre) ya da yansıma katsayıları (kafes filtre) olarak adlandırılır. Uyarlamalı filtre uygulamaları arasındaki en önemli fark ise, istenen yanıtın elde ediliş biçimidir. Bu farklılığa göre, uyarlamalı filtre uygulamaları dört temel sınıfa ayrılabilir (Tablo 5.1).

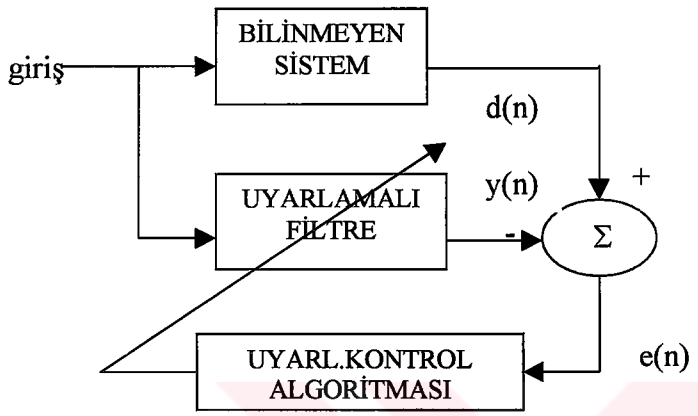
Tablo 5.1 Uyarlamalı filtre uygulamaları

Uygulamanın Sınıfı	Uygulamanın Adı
1. ÖZDEŞLEME	Sistem Özdeşleme
	Katmanlı Yer Modelleme
2. TERSİNİ MODELLEME	Öngörüsel Katlama Çözme
	Uyarlamalı Dengeleme
3. ÖNGÖRÜ	Doğrusal Öngörülü Kodlama
	Uyarlamalı Farksal Darbe Kod Modulasyonu
	Özyinelemeli Spektrum Analizi
	Sinyal Sezme
4. GİRİŞİM GİDERME	Uyarlamalı Gürültü Giderme
	Yankı Giderme
	Uyarlamalı Işın Biçimlendirme

Bu uygulamalarda, uyarlamalı filtrenin ne amaçla kullanıldığı ve istenen yanıtın nasıl elde edildiği aşağıda kısaca açıklanmaktadır:

1. Sistem Özdeşleme (System Identification)

Uyarlamalı filtrenin bu uygulama sınıfındaki görevi: Bilinmeyen bir sisteme en çok benzeyecek bir doğrusal modeli sağlamaktır. Bilinmeyen sistem ve uyarlamalı filtre aynı giriş ile sürülür.Bilinmeyen sistemin çıkışı uyarlamalı filtre için gereken istenen yanıt olur (Şekil 5.1).

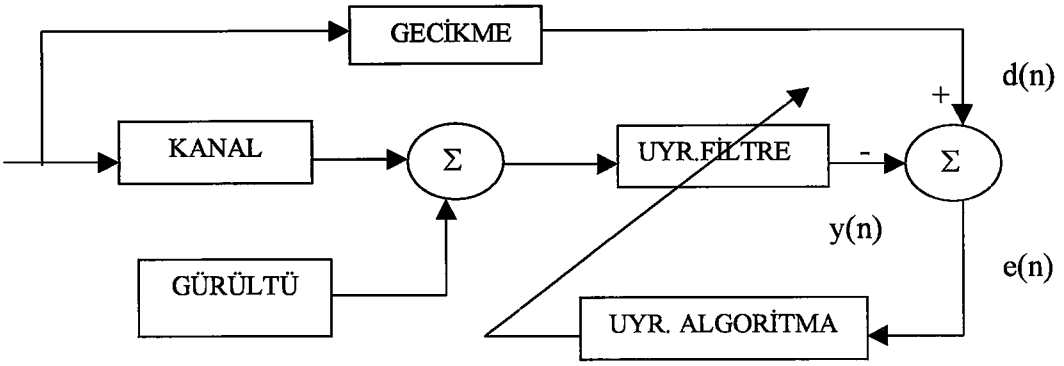


Şekil 5.1 Sistem Özdeşleme

2. Tersini Modelleme (Inverse Modeling)

Uyarlamalı filtrenin bu uygulama sınıfındaki görevi: Bilinmeyen gürültülü bir sistemin tersine en çok benzeyecek olan bir ters model sağlamaktır (Şekil 5.2).

Elde edilen ters modelin transfer fonksiyonu bilinmeyen sistemin transfer fonksiyonunun tersine eşit oluyor ise, ideal durum geciktirilerek sağlanır. Gecikme miktarı, bilinmeyen sisteme ve kullanılan uyarlamalı filtrenin yapısına bağlı olarak değişir. Bazı uygulamalarda, bilinmeyen sistemin girişi hiç geciktirilmeden de istenen yanıt olarak kullanılmaktadır



Şekil 5.2 Tersini Modelleme

3. Öngörü (Prediction)

Uyarlamalı filtrenin buradaki görevi: Bir rasgele işaretin şimdiki ana ait değerinin en iyi öngörüsünü elde etmektir. Buna göre, rasgele işaretin şimdiki değeri, uyarlamalı filtre için gereken istenen yanıt olacaktır. İşaretin geçmiş değerleri ise, filtrenin girişi olarak kullanılır.

4. Girişim Giderme (Interference Cancelling)

Uyarlamalı filtre burada, bir ana işaretin bilgi taşıyan bileşeninde bulunan bilinmeyen girişimi gidermek için kulanılır. Bu durumda ana işaret, uyarlamalı filtre için gereken istenen yanıt olacaktır. Filtrenin girişi ise, bir referans işaret ile sürülür.

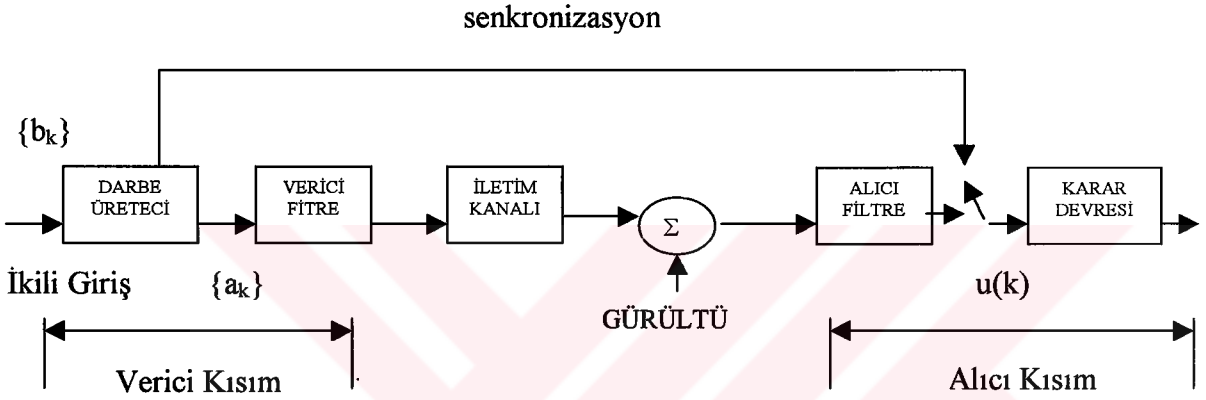
5.1 Uyarlamalı Dengeleme Kavramı

Dengeleyici (Equalizer): Daha çok telefon kanallarında sinyalin büyüklüğü ve kanalın gecikme karakteristiklerini ayarlamak için kullanılan filtre olarak tanımlanır.

Kanalın sınırlı olan bant genişliğini en verimli şekilde kullanabilecek veri iletim sistemleri geliştirmek, üzerinde yoğun çalışmaların yapıldığı bir konudur. Amaç, kabul edilebilir bir hata oranı (error rate) ya da ortalama sembol hata olasılığı (average probability of symbol error) ile mümkün olan en yüksek hızda veri iletimi yapabilecek bir sistem tasarlamaktır. Doğrusal bir haberleşme kanalından sayısal veri iletimini sınırlayan iki etken vardır:

1. Simgelerarası Girişim (İntersymbol interference, ISI): Verici filtre, iletim kanalı ve alıcı filtre üçlüsünde meydana gelen dağılma (dispersion) nedeniyle oluşur.

2. Toplamsal Isıl Gürültü (Additive Thermal Noise): Telefon kanalları gibi sınırlı bant genişliğine sahip kanallar için simgelerarası girişim, yüksek hızlı veri iletim sistemlerinin tasarımında belirleyici baş etkendir. İkili darbe genlik modülasyonu (binary pulse-amplitude modulation) sisteminin eşdeğer temel bant modeli, Şekil 6.1'de gösterilmiştir. Temel bant terimi, orjinal işarete ait frekansların bulunduğu frekans bandına verilen addır.



Şekil 5.3 Bir temel bant veri iletim sistemine ait blok şema (dengelemesiz)(Haykin, 1991)

Sistemin verici kısmının girişine uygulanan işaret, 0 ve 1 değerlerinden oluşan $\{b_k\}$ ikili veri dizidir. Bu dizi sırasıyla darbe üretici, verici filtre ve iletim kanalı yolunu izleyerek alıcı filtreye varır. u(k), alıcı filtrenin örneklenmiş çıkışını göstersin. Örnekleyici, vericideki darbe üretici ile senkron olarak çalıştırılır. u(k) çıkışına, karar devresi aracılığıyla bir eşik seviyesi aşılıyor ise 1 sembolüne karar verir.

$$a_k = \begin{cases} +1 \ , & b_k \ \ giriş \ \ biti:1 \ \ ise \\ -1 \ , & b_k \ \ giriş \ \ biti:0 \ \ ise \end{cases} \qquad (5.1)$$

şeklinde bir çarpan tanımlansın. Buna göre gürültünün olmadığı durumlarda u(k) için

$u(k)=\sum a_n p(k-n)$

$u(n)=a_n p(0)+\sum a_n p(k-n)\ ,\ (k\neq n)$ (5.2)

eşitliği yazılabilir. p(n), birbirine kaskad bağlı verici filtre, iletim kanalı ve alıcı filtre üçlüsünün oluşturduğu sisteme ait p(t) impuls yanıtının zaman domeninde örneklenmiş halidir. (5.2) eşitliğinin sağ tarafındaki ilk terim $a_k p(0)$ istenen yanıtı, ikinci terim ise kaskad bağlı verici filtre, iletim kanalı üçlüsünün oluşturduğu sistemden kaynaklanan simgelerarası girişimi gösterir. Eğer bu simgelerarası girişim kontrol altına alınmaz ise, karar devresi girişine gelen örneklenmiş işaret hakkıda hatalı kararlar vermeye başlar. Öyleyse, simgelerarası girişimin üstesinden gelebilmek için, p(n) fonksiyonunun kontrol altında tutulması gerekir.

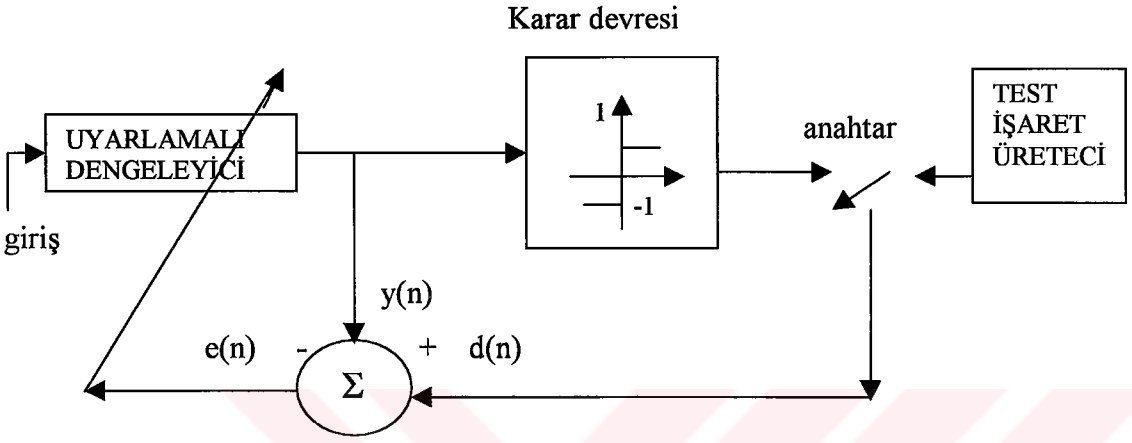
Teorik olarak, iletim kanalının karakteristikleri tam olarak bilinirse, simgelerarası girişim örnekleme anlarındaki etkisini azaltacak bir verici ve alıcı filtre çiftinin tasarlanabileceği kabul edilir. Oysa pratikte, durum farklıdır; iletim kanalı, halen bilinmekte olan iletim kanalı modelleri içinden rasgele herhangi bir tanesine eşit ya da benzer olmaktadır. Bu nedenle varolan kanal modelleri göz önüne alınarak elde edilen ortalama kanal karakteristiklerine göre tasarlanmış sabit bir verici ve alıcı filtre çifti simgelerarası girişimi istenilen ölçüde azaltmayabilir. Bu durum, kanalın zaman yanıtı üzerinde tam bir kontrol sağlayan uyarlamalı dengeleyici kullanımını gerektirir (Lucky, 1965,1966; Lucky vd.,1968; Proakis, 1975; Qureshi, 1985).

Dengeleyici terimi, telefon kanallarında kanalın genlik ve gecikme karakteristiklerini düzenlemekle görevlendirilen bir filtreyi tanımlamak için kullanılır.

Veri iletim sistemlerinin dengelenmesi için öne sürülen temel görüşler arasında, iletim sisteminin önce verici kısımda ve sonrada alıcı kısımda olmak üzere iki kez dengelenmesi gerektiğide yer almaktadır. Bu eski teknik, bir geribesleme yolu kullanmayı gerektirdiği için, burada sadece alıcı kısmında, yapılan dengeleme işlemi ele alınacaktır. Uyarlamalı dengeleyici alıcı kısımda, örnekleyici ile karar devresi arasına yerleştirilir. Teoride simgelerarası girişim etkisinin, uyarlamalı dengeleyicideki ayarlanabilir katsayıların (tap ağırlıklarının ve/veya yansıma katsayılarının) sayısı sınırsız derecede arttırılarak azaltılabileceği kabul edilir.

Uyarlamalı filtreleme algoritmaları, ancak e(n) hata işareti ile çalışabilirler. Hata işaretini hesaplayabilmek içinse, LMS algoritması gereği, d(n) istenen yanıtının bilinmesi gerekir.

Teoride, verici kısım çıkışındaki iletilen işaret, uyarlamalı dengeleyici için gereken istenen yanıttır. Oysa pratikte, uyarlamalı dengeleyici alıcıya yerleştirildiği için, dengeleyici ile istenen yanıt fiziksel olarak birbirlerinden ayrılmışlardır. Bu yüzden, alıcı kısımda istenen yanıtın tıpkısının üretilmesi gerekmektedir. Şekil 5.4'de blok şema ile gösterilen uyarlamalı dengeleyici, alıcı kısma yerleştirilmiş olup, istenen yanıtın kopyasını üretmek için birbirini izleyen şu iki mod kullanır:



Şekil 5.4 Çalışma modu ile çalışan bir uyarlamalı dengeleyiciye ait blok diyagram (Hayes,1996)

1. Öğrenme Modu (Training Mode): Bu mod, alıcı ve vericinin birbirine bağlandıkları ilk anda gerçekleşmeye başlayan başlangıç eğitim aşaması (initial training phase) sırasında yeralır. Bu aşamada iletim kanalının nasıl olduğunu test etmek amacıyla verici kısımdan alıcıya sözde rasgele olan aslında alıcı tarafından bilinen bir test işareti gönderilir. Test işaretinin alıcı tarafından biliniyor olmasının nedeni, bu işaretin alıcı kısımda depolanmış olmasından veya alıcı kısımda bulunan bir test işareti üreteci tarafından üretiliyor olmasından dolayıdır. Gönderilen test işaretine dengeleyicinin verdiği yanıt ile istenen yanıt görevini gören test işareti birbirinden çıkarılarak kestrim hatası elde edilir. Elde edilen bu hata ile dengeleyicinin ayarlanabilir katsayılarına başlangıç değerleri verilmiş olur

2. Çalışma Modu(Decision-directed Mode): Öğrenme modu ile dengeleyicinin ayarlanabilir katsayılarına başlangıç değerleri verildikten hemen sonra, alıcının verici

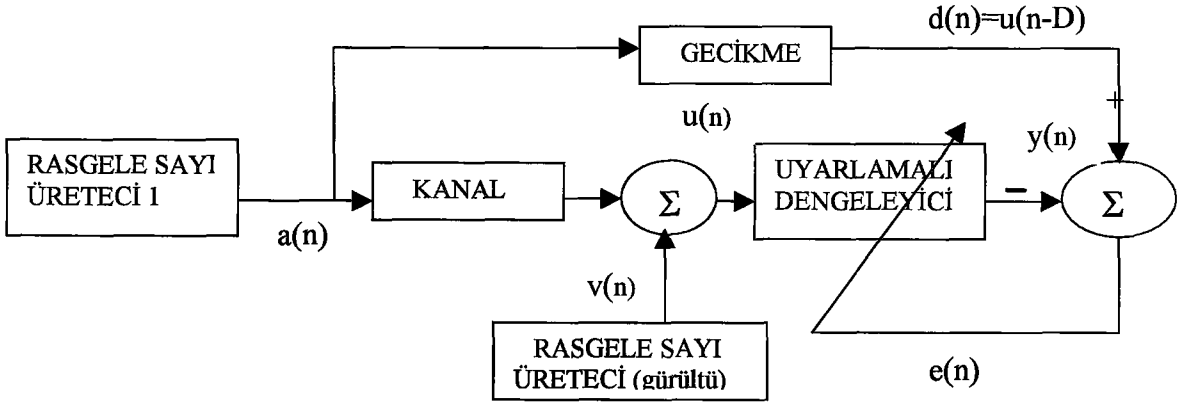
kısımdan ne gönderildiğine dair hiçbir ön bilgisi olmaksızın uyarlamalı dengeleyicinin katsayıları bu mod vasıtasıyla sürekli olarak ayarlanır. Bu mod dengeleyici çıkışı bir karar devresinin girişine uygulanmakta ve karar devresinin çıkışı da dengeleyici için gereken istenen yanıtı oluşturmaktadır.

İstenen yanıtın kopyası olarak alıcının kendi verdiği kararlar kullanıldığı için, bu moda karar-yönetimli denilmiştir. Dolayısiyle, bu modda üretilen istenen yanıtın aslına ne denli benzer olacağı karar devresinin vereceği doğru kararlara bağlıdır. Bu nedenle, ortalama simge hata olasılığı belli bir seviyeyi aşmadığı (yüzde 10'dan az olduğu) sürece, verilen kararların, dengeleyici katsayılarının uygun olarak ayarlanabilmesine yetecek kadar doğru olduğu kabul edilir.

Eğer belirlenen düzey aşılıyor ise, hata işareti dengeleyicinin doğru ayarları yapmasına engel olur. Alıcı, böyle bir durum gerçekleşmeden önce, test işaretinin tekrar iletilerek dengeleyici katsayılarının yeniden koşullandırılmasını isteyebilir.

5.2 En Küçük Karesel Ortalama Algoritması (LMS) Kullanarak Kanal Dengeleme

Bu bölümde, alınan sinyalde bozulmalar (distortion) yaratan doğrusal dağılımlı (linear dispersive) bir iletim kanalının uyarlamalı dengelenmesi için LMS algoritması kullanılmaktadır. Üzerinde çalışacağımız sistemin blok şeması Şekil 5-5'te verilmiştir.



Şekil 5-5 Uyarlamalı kanal dengeleme uygulamasına ait blok şeması (Haykin 1991)

Rasgele sayı üreteci-1, kanalın nasıl olduğunu test etmek amacıyla kullanılan u(n) test işaretini üretir. Rasgele sayı üreteci-2 ise, kanalın çıkışındaki işarete karşın v(n) toplamsal beyaz gürültü kaynağı olarak kullanılır. Bu iki rasgele sayı üreteci, birbirinden bağımsızdır.Uyarlamalı dengeleyicinin görevi, iletilen u(n) işaretinde kanalın yarattığı bozulmayı, toplamsal beyaz gürültünün varlığına karşın düzeltmektir. Bu göreve genel bir ifade ile kanalın tersini modellemek (inverse modelling) denir.

Kanalın girişine uygulanan rasgele dizi {a(n)}, eşit olasılıkla üretilen ±1 değerlerinden oluşmakta ve sıfır ortalamalı kabul edilmektedir. Kanalın impuls yanıtı olarak,

$$h(n)=\begin{cases}\frac{1}{2}\left[1+\cos\left(\frac{2\pi}{W}(n-2)\right)\right] & , \quad n=1,2,3\\ 0 & ,\end{cases} \tag{5.3}$$

biçiminde tanımlanan yükseltilmiş kosinüs (raised cosine) kullanılacaktır. W parametresi asıl olarak, kanalın yarattığı bozulmanın genliğini kontrol eder. Kanalın yarattığı bozulmanın miktarı dengeleyicinin girişindeki işareti etkilediği için, W parametresi dolaylı olarak, dengeleyicinin tap girişlerine ait ilişki matrisinin özdeğer yayılımı $\chi(R)$'yi de kontrol etmiş olur. Öğrenme eğrilerinde de görüleceği gibi, W değeri artıkça, kanalın yarattığı bozulmanın genliği artar. Bozulma miktarının artması ise, özdeğer yayılımını arttırır. W kanal parametresinin $\chi(\mathbf{R})$'yi bu örnekten görüleceği gibi doğrudan etkilemesi, bütünüyle bu kanalın özelliğidir.Yoksa her kanal için, bu durum geçerli değildir.

Adım uzunluğu(μ) ayrıca uyarlamlı filtrenin çıkışını doğrudan etkileyen bir başka önemli parametre olduğu için, çeşitli değerlerle hesaplama yapılacaktır. Rasgele sayı üreteci(2) tarafından üretilen rasgele dizi {v(n)}, sıfır ortalamalı σ^2_v değişintili olup, gerçel sayılardan oluşur. σ_v^2 değeri, kanal çıkışında ölçülen işaret-gürültü oranınıda gösterir. Örneğin, σ^2_v=0.001 için kanal çıkışındaki işaret-gürültü oranı 30 dB olur. Buraya kadar anlatılanlara göre, uyarlamalı dengeleyicinin n anındaki girişi,

$$u(n)=\sum_{k=1}^{3}a(n-k)h(k)+v(n) \tag{5.4}$$

olarak yazılabilir.

Uyarlamalı filtrenin tap ağırlıkları sayısı (M+1) aynı zamanda bir gecikmeye neden olmaktadır. a(n) işaretinin uyarlamalı dengeleyicinin çıkışına ulaşıncaya kadar kanal ve uyarlamalı dengeleyicide toplam D miktar gecikmeye uğradığı düşünülür. Bu nedenle a(n) işaretinin d(n)=a(n) şeklinde doğrudan istenen yanıt olarak kullanmak yerine kanal ve uyarlamalı dengeleyicide ugradığı toplam gecikme miktarı D kadar geciktirip d(n)=a(n-D) şeklinde istenen yanıt olarak kullanılır. Uyarlamalı bir enine filtrenin tap ağırlıkları öğrenme modu boyunca kestirim hataları kullanılarak yapılan sürekli ayarlamalar sonucunda optimum değerlerine ulaşılırlar ve böylece enine filtre kalıcı halde çalışmaya başlamış olur. D genel olarak kanal katsayısı ve filtrenin tap ağırlıklarının toplamının yarısı olarak alınır.

Giriş verisinin karakteristikleri değiştiği zaman öğrenme modu yeniden başlar. (Uyarlamalı ya da uyarlamasız) bir enine filtrenin impuls yanıtı o filtrenin optimum tap ağırlıklarıyla tanımlanır. Öyleyse bir enine filtrenin impuls yanıtının uzunluğu tap ağırlıklarının sayısına eşittir.

Gecikme D'nin değerinin hesaplanması:
İmpuls yanıtın uzunluğu enine filtrenin tap ağırlıklarının sayısına (M+1) eşittir.

Kanalın impuls yanıtı h(n)'nin tanım aralığının orta noktası :

n=(kanalın impulse yanıtının uzunluğu +1)/2 (5.5)

olur. Uyarlamalı dengeleyicinin impuls yanıtı w(n):M için seçilen değerin tek sayı olması durumunda,

n=((M+1)/2) (5.6)

anına göre yaklaşık olarak simetrik; M için seçilen değerin çift sayı olması durumunda ise,

n=M/2 (5.7)

anına göre tam simetrik olmaktadır. Buna karşılık, uyarlamalı dengeleyici için gereken istenen yanıtı sağlamak amacıyla, u(n) işaretinin toplam,

$$D = \begin{cases} (\text{Kanal impuls yaniti uzunlugu} + 1) + (M+1)/2 \text{ , M tek sayı} \\ (\text{Kanal impuls yaniti uzunlugu} + 1) + M/2 \text{ , M çift sayı} \end{cases} \quad (5.8)$$

örnek geciktirilmesi gerekir. Örneğin, kanalın impuls yanıtının (5.3) eşitliğindeki gibi tanımlandığı ve uyarlamalı dengeleyicideki filtrenin M=10 modüllü olduğu kabul edilsin. Kanalın impuls yanıtı n=2 anına göre simetrik olduğundan, uyarlamalı dengeleyici için gereken istenen yanıt d(n), u(n) işareti toplam,

D=(3+1)/2+10/2=7

örnek geciktrilerek sağlanmalıdır. Dolaysiyle d(n)=u(n-7) olmalıdır. Burada bulunan kuramsal değerdir. Şimdi sistemimizdeki kanalın ve uyarlamalı dengeleyicinin gecikme miktarlarını bulalım. Gecikme değeri aynı zamanda Kanalın (ve uyarlamalı dengeleyicinin) impuls yanıtı grafiğinin simetrik olduğu noktadır.

W = 2.9 alınırsa (5.3) eşitliğinde şu değerler bulunur:

$$h(n) = \begin{cases} 0.2194 & , \quad n = 1,3 \\ 1 & , \quad n = 2 \end{cases}$$

Buna ilişkin impuls yanıtın n=2 anında simetrik olduğu Şekil 5.6'da verilmektedir.

Şekil 5.6 Kanalın impuls yanıtı

Öte yanda W'nın bu çalışmada kullanılan diğer değerleri için hesaplanan h(n) değerleri aşağıda verilmiştir. Bunlar incelendiğinde n=2'de h(n)'nin simetrik olduğu görülmektedir.

W=3.1 için
h(n)=0.2798 n=1,3
h(n)=1 n=2

W=3.3 için
h(n)=0.3365 n=1,3
h(n)=1 n=2

W=3.5 için
h(n)=0.3887 n=1,3
h(n)=1 n=2

Şimdi bu kanala ilişkin kanal dengeleyicinin katsayılarının farklı W değerleri için simetri noktasını bulalım. Önce kullandığımız LMS algoritmasını kısaca anımsayalım.

Parametreler:

M+1=Filtre katsayısı (parametre sayısı)
μ=adım uzunluğu
$0<\mu<2/$(toplam giriş gücü)
Toplam giriş gücü=Mr(0)
Başlangıç koşulları:
w(0)=**0**

Verilenler :

u(n)=(M+1)x1 giriş vektörü
d(n)=n anında istenen yanıt
İstenenler :
w(n+1)

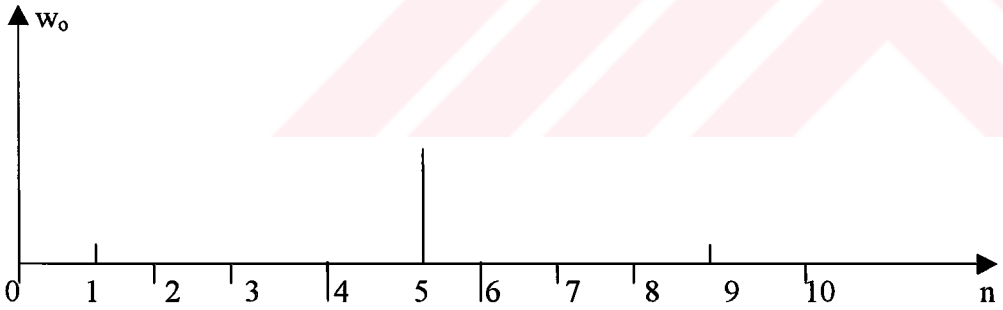
formülleri:

$e(n) = d(n) - \mathbf{w}^H(n)\,\mathbf{u}(n)$

$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{u}(n)e^*(n)$

1. W=2.9, iterasyon sayısı 1000, μ=0.0075 iken

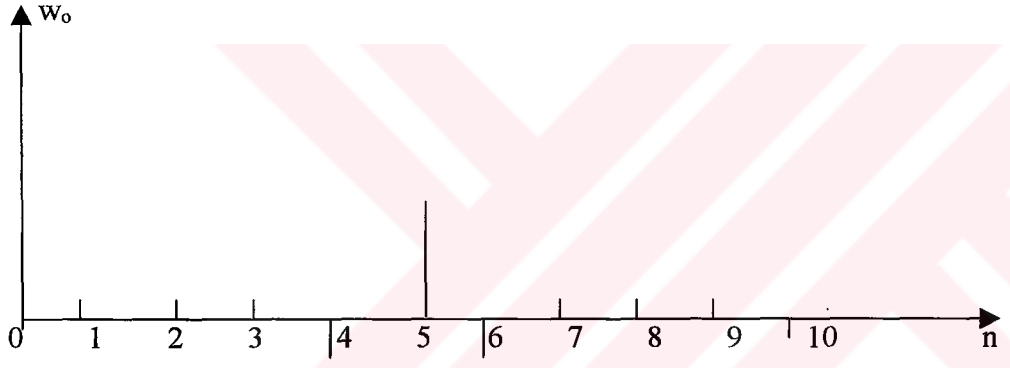
w(0)=-0.022845
w(1)=0.052816
w(2)=-0.006460
w(3)=-0.005516
w(4)=-0.019101
w(5)=0.092256
w(6)=-0.018367
w(7)=-0.031829
w(8)=-0.017778
w(9)=0.027086
w(10)=-0.005355



Şekil 5.7 Uyarlamalı dengeleyicinin impuls yanıtı

2. W=3.1, deneme sayısı 1000, μ=0.0075 iken

w(0)=-0.003009
w(1)=0.011607
w(2)=0.030328
w(3)=0.013297
w(4)=-0.01507
w(5)=0.081918
w(6)=-0.015897
w(7)=0.005423
w(8)=0.001781
w(9)=0.003917
w(10)=-0.01053



Şekil 5.8 Uyarlamalı dengeleyicinin impuls yanıtı

Yukarıdaki sonuçlar göstermektedir ki, n=5 anında simetri vardır. Dolayısıyla daha önce bulunduğu gibi gecikme D=2+5=7 olur.

5.3 Uyarlamalı Dengeleyicinin İmpuls Yanıtı

Uyarlamalı dengeleyicinin görevi, iletilen a(n) işaretinde kanalın yaratığı bozulmayı, toplamsal beyaz gürültüye karşın düzeltmektir. Dengeleyici bu düzeltmeyi, kendi transfer fonksiyonu W(w)'i kanalın transfer fonksiyonu H(w)'in tersine mümkün olabildiğince benzeterek yapar; idealde W(w)H(w)=1 olması istenir. Öyleyse, |W(w)| eğrisi |1/H(w)|

eğrisine ne kadar çok benziyor ise, uyarlamalı dengeleyici de o kadar başarılı çalışıyor demektir.

5.4 Öğrenme Eğrileri

Bir algoritmanın öğrenme eğrisi, o algoritmanın ürettiği kestirim hatasının karesel ortalama değerinin, iterasyon sayısına göre değişimdir. Bilgisayar simülasyonuyla farklı adım uzunlukları, gürültü varyansı, dengeleyici katsayıları ve kanal kontrol parametresi için öğrenme eğrileri elde edilmektedir. Öğrenme eğrilerinin elde edilişinde kullanılan değerler şunlardır: Kanal kontrol parametresi değerleri W=2.9, 3.1, 3.3, 3.5; adım uzunluğu değerleri μ=0.005, 0.0075, 0.025, 0.075 ;gürültü varyansı değerleri σ^2=0.001, 0.01, 0.1 ve uyarlamalı dengeleyici katsayıları M+1=7,9,11.

Öğrenme eğrileri elde edilirken kullanılan kanal impuls yanıtları eşitlik (5.3) kullanarak hesaplanmış ve aşağıda verilmiştir:

1. $W = 2.9$

$$h(n) = \begin{cases} 0.2194 & , \quad n = 1,3 \\ 1 & , \quad n = 2 \end{cases} \tag{5.9}$$

2. $W = 3.1$

$$h(n) = \begin{cases} 0.2798 & , \quad n = 1,3 \\ 1 & , \quad n = 2 \end{cases} \tag{5.10}$$

3. $W = 3.3$

$$h(n) = \begin{cases} 0.3365 & , \quad n = 1,3 \\ 1 & , \quad n = 2 \end{cases} \tag{5.11}$$

4. W = 3.5

$$h(n) = \begin{cases} 0.38873 & , \quad n = 1,3 \\ 1 & , \quad n = 2 \end{cases}$$

Daha önce de belirtildiği gibi, üç parametre LMS algoritmasının impuls yanıtını etkilemektedir. Bunlar adım uzunluğu, uyarlamalı dengeleyici parametre(katsayı) sayısı, dengeleyicinin özilişki matrisinin özilişki değerleri(yani kanal kontrol parametresi W). Bunlara ek olarak gürültü varyansı da LMS impuls yanıtını etkilemektedir.Bu dört parametreye ilişkin farklı değerler için öğrenme eğrileri elde edilerek bunların, dengeleme başarımına etkileri incelenmekte ve yorumlanmaktadır.

Genel olarak adım uzunluğu küçük seçilirse, yakınsama o ölçüde yavaş olmaktadır. Bunun yanında durağan durum değeri küçük olmaktadır.

Eğer adım uzunluğu büyük seçilirse, yakınsama görece hızlı olmaktadır. Ancak karesel ortalama hatanın durağan durum değeri daha büyük olmaktadır. Karesel ortalama hatanın yakınsama biçimi aynı zamanda filtrenin parametre sayısına bağlıdır. Bunun için gerekli ve yeterli koşul,

$0<\mu<2/(\sum\lambda_i)$'dir

λ_i, özdeğer,i=0,1,…,M+1

toplam giriş gücü=$\sum\lambda i$

Eğer bu koşul sağlanırsa LMS algoritması yakınsaktır denir.Bu çalışmada kanalın yapısı gereği W, kanalın **R** matrisinin özdeğerlerini doğrudan etkiler. Özdeğer yayılımı (ya da dağılımı), $\chi(\mathbf{R})=\lambda_{max}/\lambda_{min}$, özdeğerlerin bir fonksiyonu olduğu için, W değişiminde etkilenmektedir.

Örneğin, W=2.9, W=3.5 değerleri için özdeğer yayılım aralığı:

$\chi(\mathbf{R})$ =6.0782 , $\chi(\mathbf{R})$ =46.8216 olur.

Dağılım değeri arttıkça (yani W arttıkça) uyarlamlı dengeleyicinin yakınsaklığı yavaşlamaktadır.Örneğin Uyarlamalı dengeleyicinin yakınsaması için ($\chi(\mathbf{R})$=6.0782) yaklaşık 80 iterasyon gerekirken, aynı yakınsama değeri için ($\chi(\mathbf{R})$=46.8216) yaklaşık 500 iterasyon gerekmektedir.

Dengeleyici parametre sayısı da öğrenme eğrilerini dolaysiyle yakınsaklığı ve karesel ortalama hata durağan durum değerini etkilemektedir. Genellikle W ve σ^2 'nin büyük değerleri için dengeleyici parametre sayısının etkisi daha belirginleşmektedir.

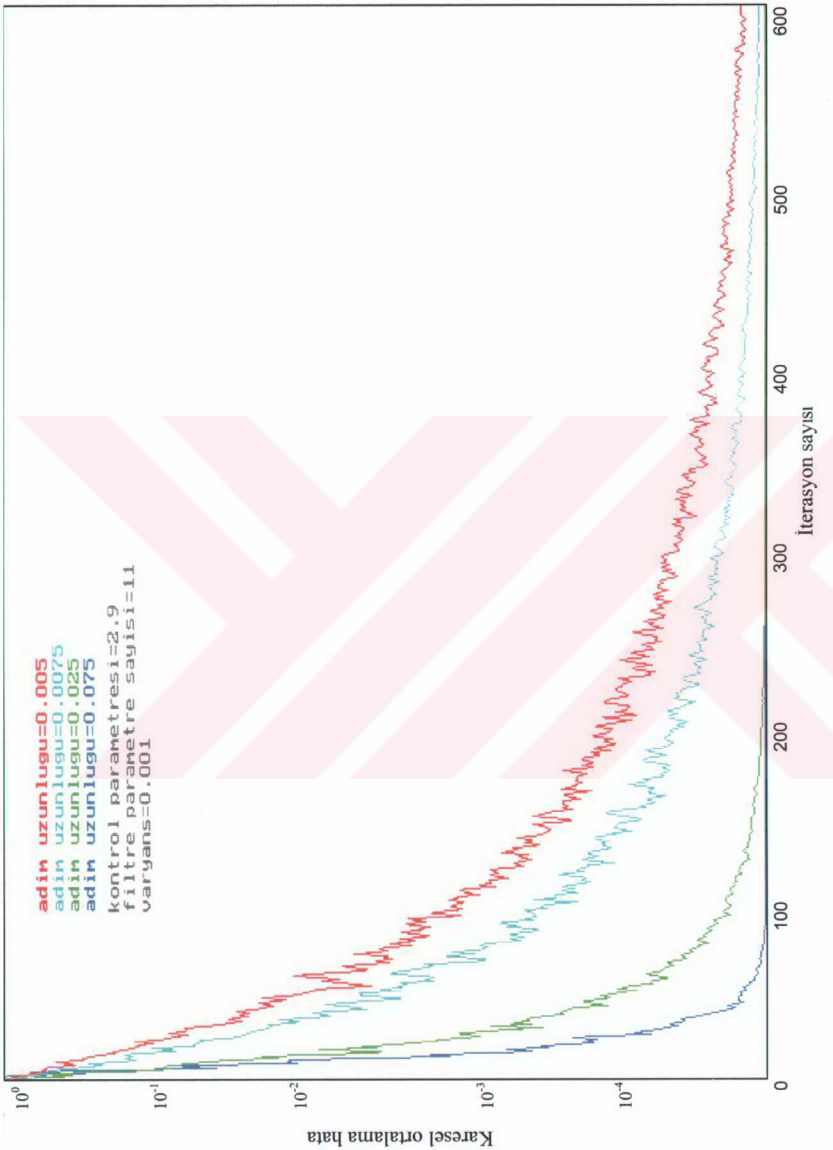
Yukarıda anlatılanları desteklemek amacıyla çizilen eğriler adım uzunluğu, kanal kontrol parametresi ve dengeleyici parametre sayısı değişimleri gözönöne alınarak üç grupta incelenmektedir. Her grupta ayrıca gürültü varyansının üç farklı değeri için (σ^2=0.001,0.01,0.1) eğriler verilmektedir.

Birinci gruptaki her şekilde adım uzunluğunun dört değeri (0.005, 0.0075, 0.025, 0.075) için eğriler elde edilmektedir. Bu grupta önce filtre parametre sayısı=11, kanal kontrol parametresi=2.9, gürültü varyansı=0.001 için öğrenme eğrileri verilmektedir. Böylelikle aynı şekil üzerinde adım uzunlukları arasında karşılaştırma yapma olanağı doğmaktadır. Aynı gruptaki ikinci şekil, dengeleyici parametre sayısı, kanal kontrol parametresi sabit tutularak, gürültü varyansının değeri değiştirilerek (σ^2=0.01 yapılarak) elde edilmektedir. Buruda gürültünün adım uzunluğu değiştikçe karesel hata ve (bir önceki şekilde gözönüne alınarak) aynı adım uzunlukları üzerindeki etkisi gösterilmektedir.Gürltü varyansının son değeri (σ^2= 0.1) için de eğriler elde edildikten sonra, dengeleyici parametre sayısı sabit tutulmakta, kanal kontrol parametresi (W=3.1) değiştirilip, gürültü varyansına yeniden 0.001 değeri verilerek eğriler elde edilmektedir. Kanal kontrol parametresinin tüm değerleri (W=2.9, 3.1, 3.3, 3.5) için eğriler elde edildikten sonra, dengeleyici parametre sayısı değiştirilerek (M+1=7,9,11), yeniden incelemeler yapılmaktadır.

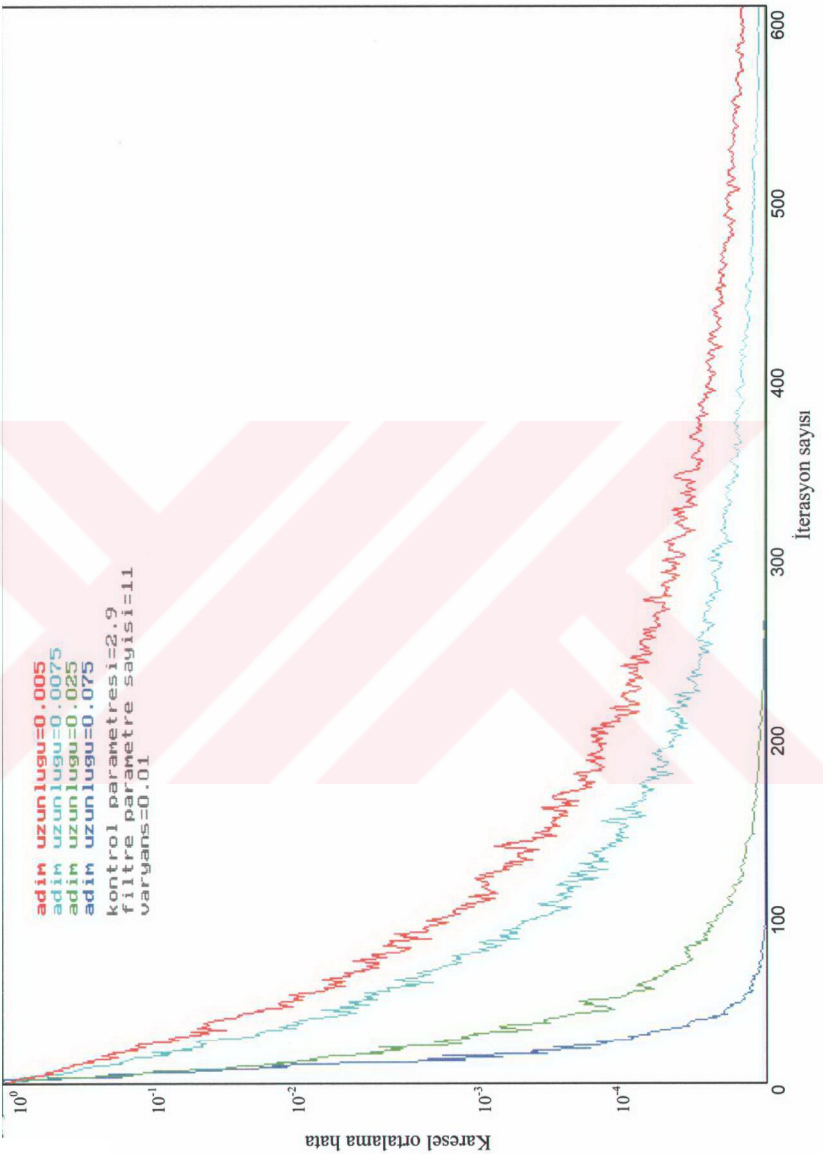
İkinci grupta, aynı şekil üzerinde kanal kontrol parametresinin dört değeri (W=2.9,3.1,3.3,3.5) için eğriler elde edilmektedir. Burada da yukarıda belirtilen kontrol parametresi yerine adım uzunluğu değiştirilmektedir.

Üçüncü grupta, aynı şekil üzerinde dengeleyici parametre sayısının üç değeri için öğrenme eğrileri elde edilmektedir. Böylece adım uzunluğu, kanal kontrol parametresi ve gürültü varyansı değiştirilerek, dengeleyicinin farklı parametre sayıları için elde edilen eğrileri karşılaştırma olanağı oluşmaktadır.

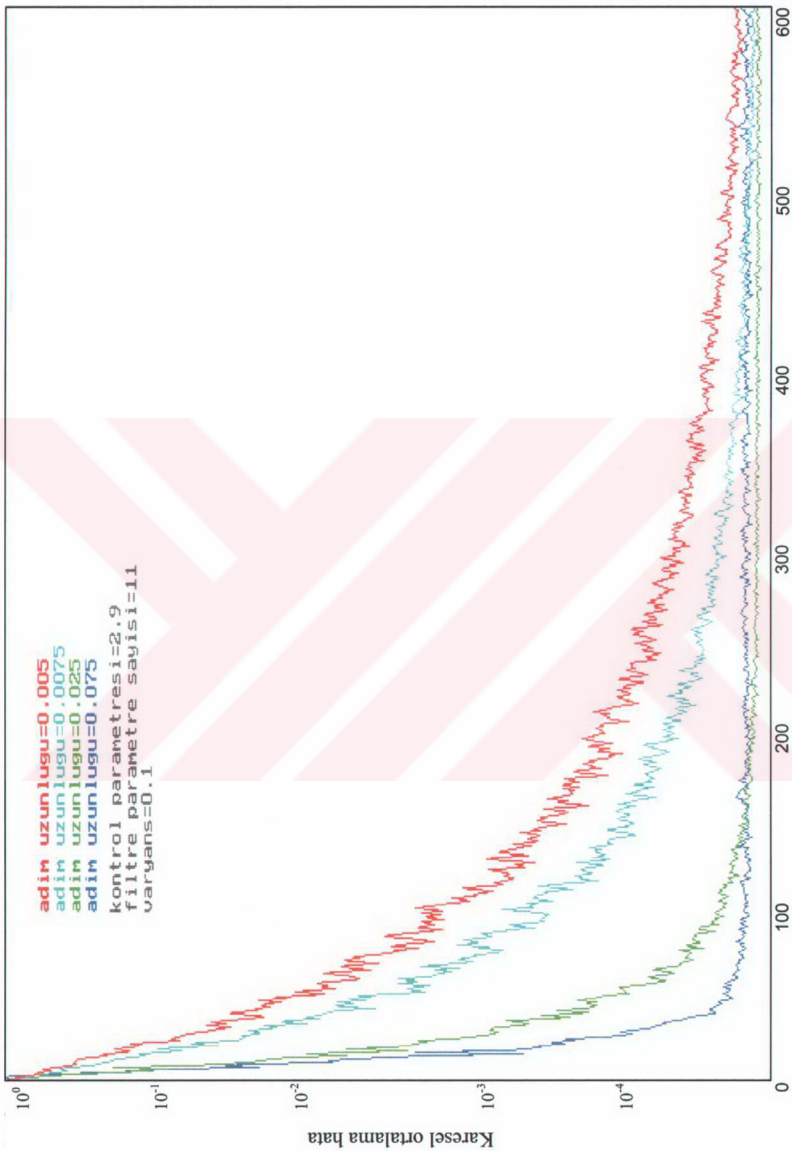
Bu şekillere ilişkin ayrıntılı inceleme ve yorumlar her şekil grubu sonunda verilmektedir.


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=2.9
filtre parametre sayisi=11
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

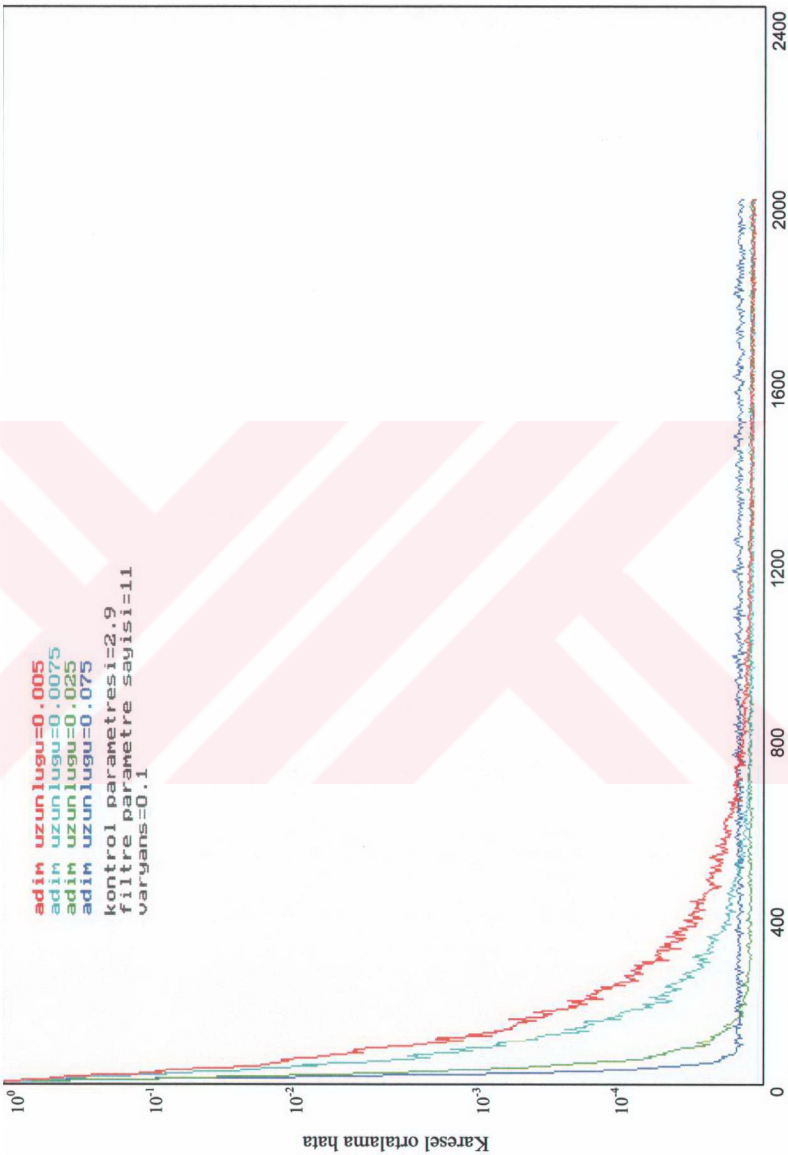
Şekil 5.9


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=2.9
filtre parametre sayisi=11
varyans=0.01
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

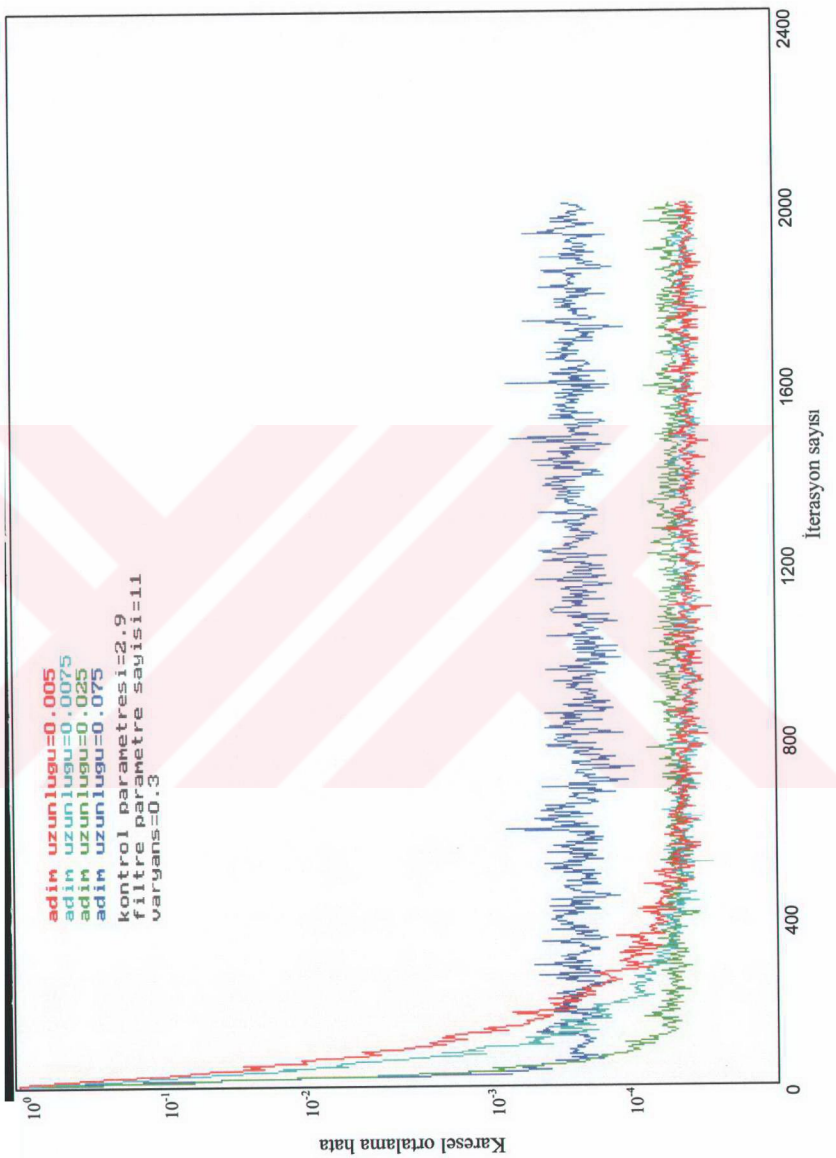
Şekil 5.10


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=2.9
filtre parametre sayisi=11
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

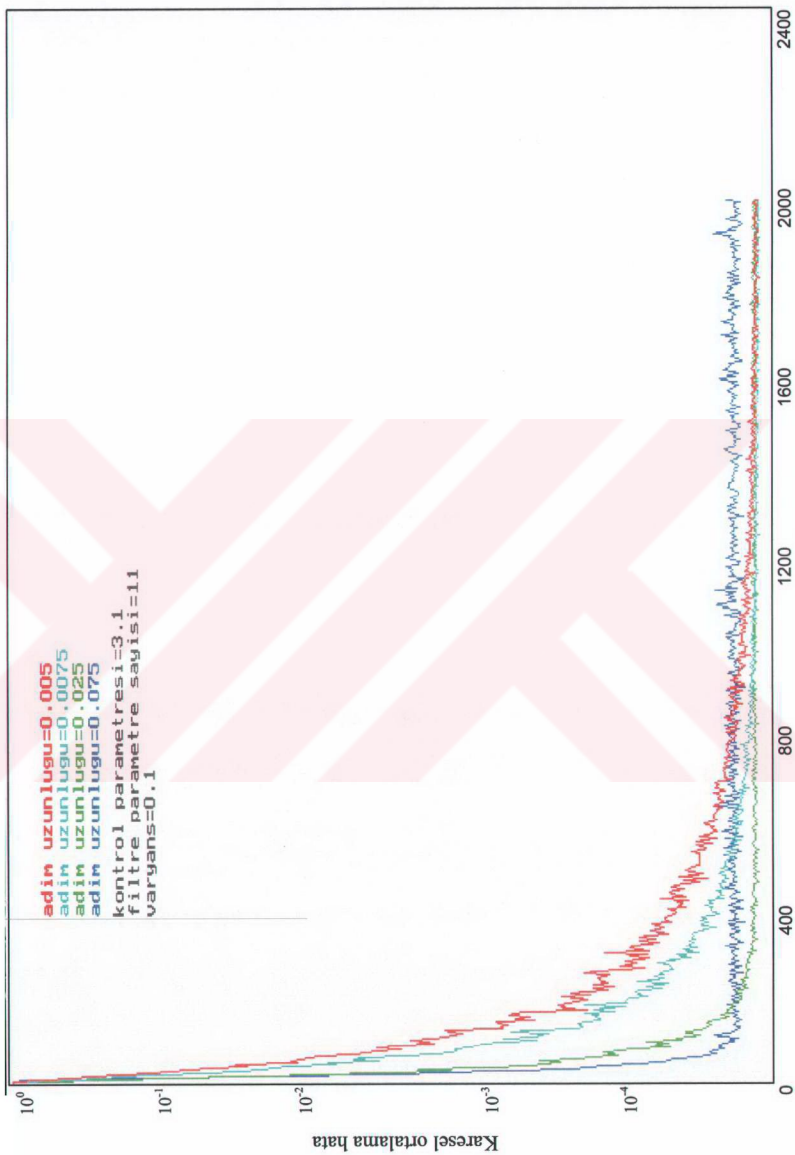
Şekil 5.11


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=2.9
filtre parametre sayisi=11
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
400
800
1200
1600
2000
2400

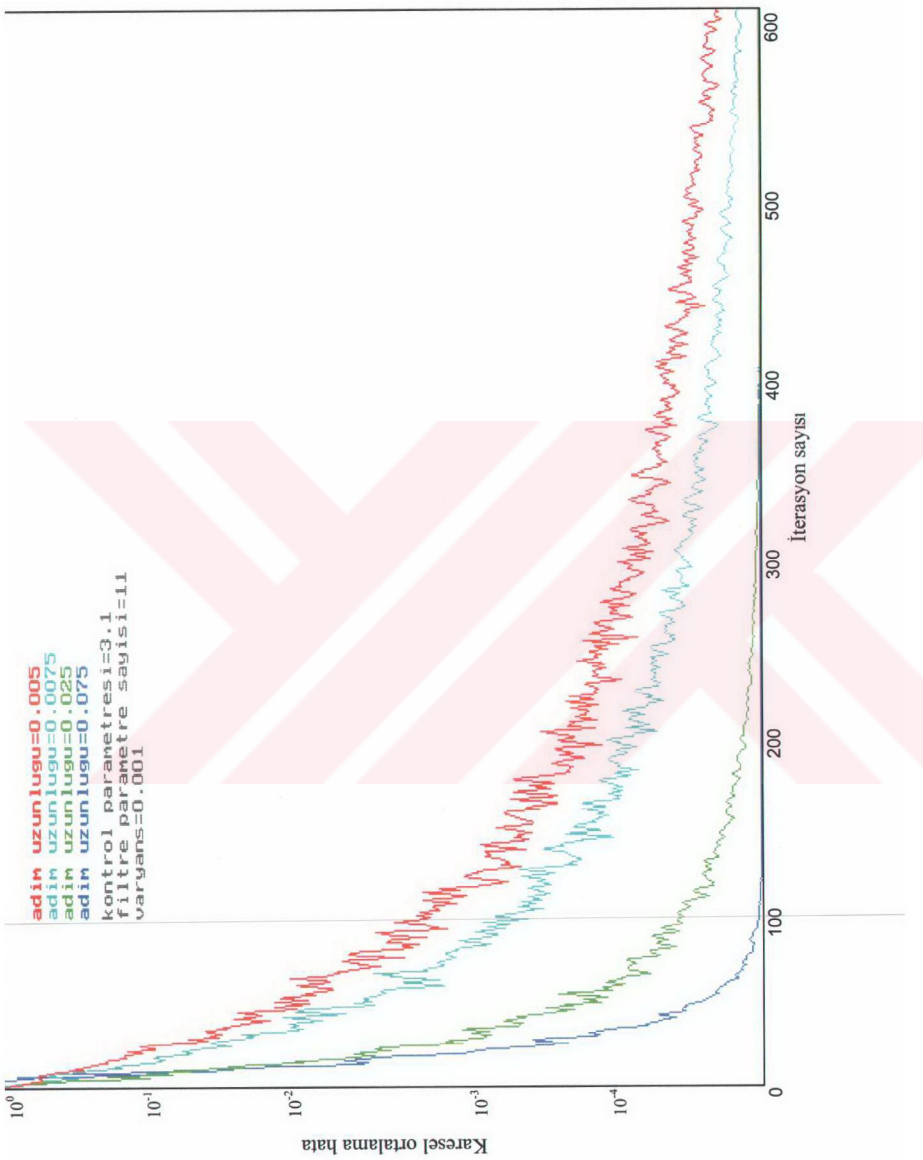
Şekil 5.12


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=2.9
filtre parametre sayisi=11
varyans=0.3
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
400
800
1200
1600
2000
2400

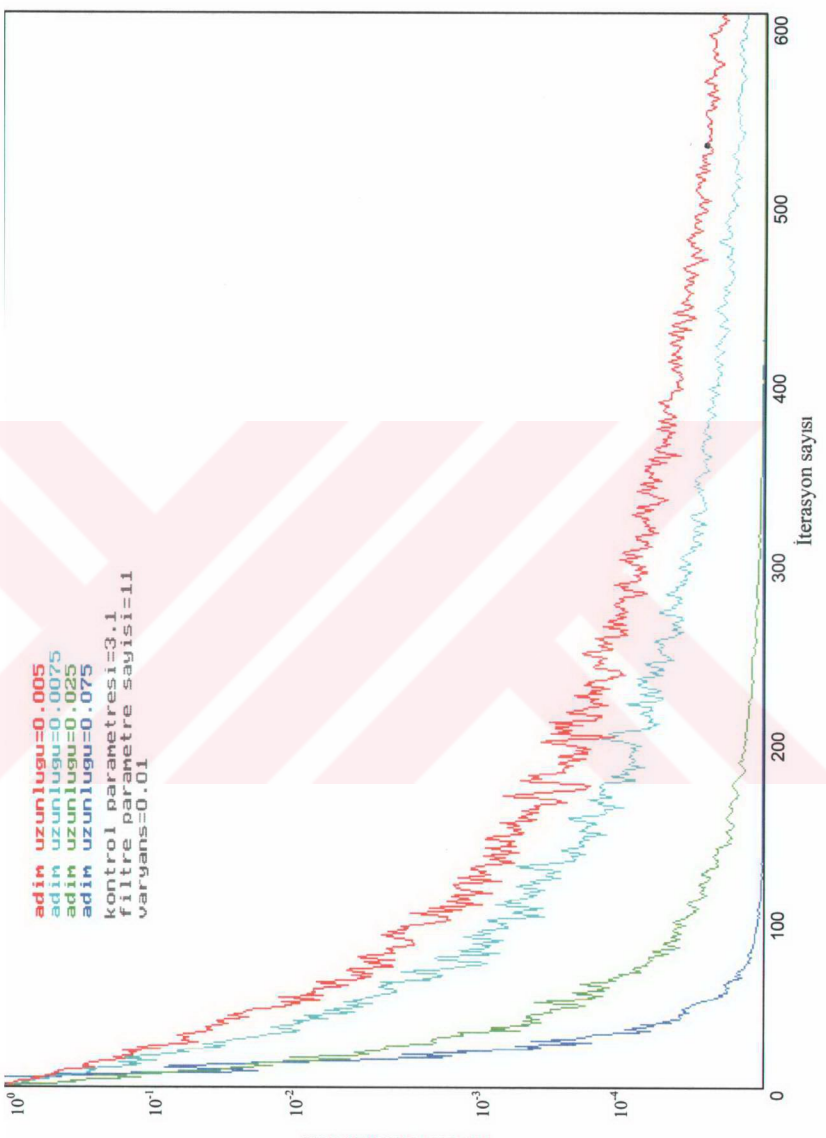
Şekil 5.13


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.1
filtre parametre sayisi=11
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
400
800
1200
1600
2000
2400

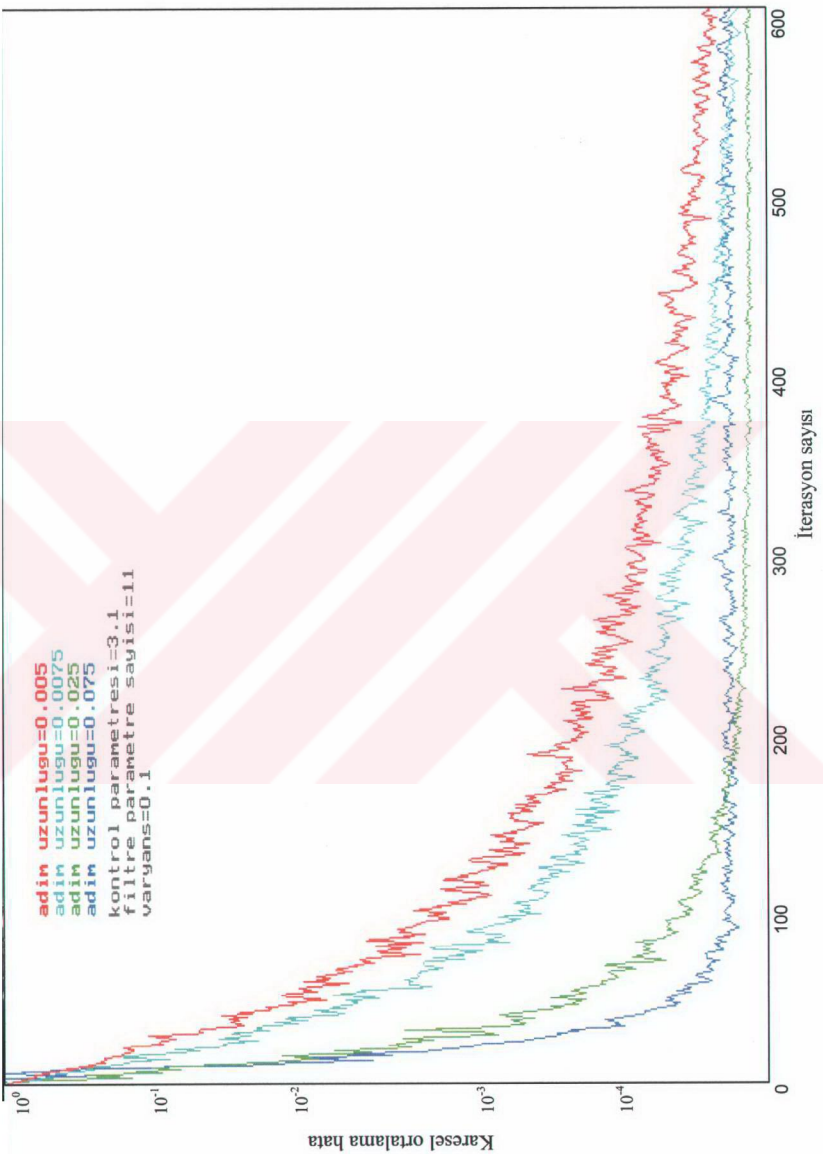
Şekil 5.14


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.1
filtre parametre sayisi=11
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

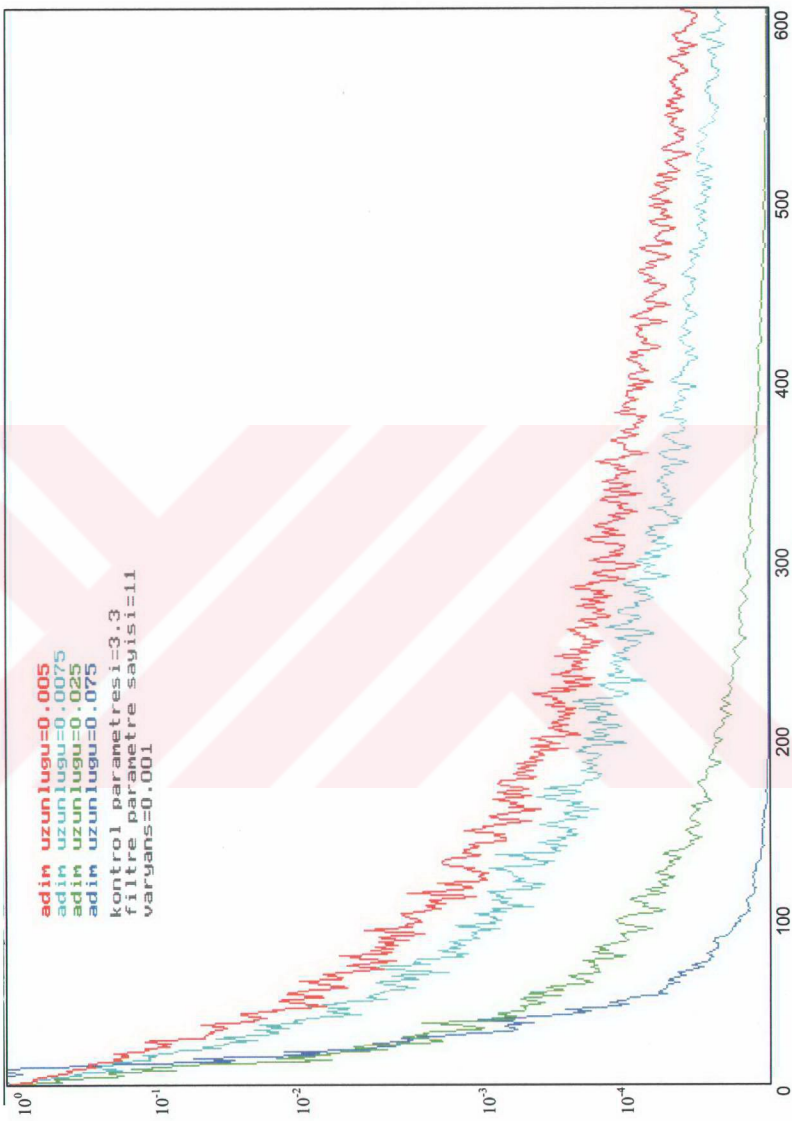
Şekil 5.15


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.1
filtre parametre sayisi=11
varyans=0.01
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

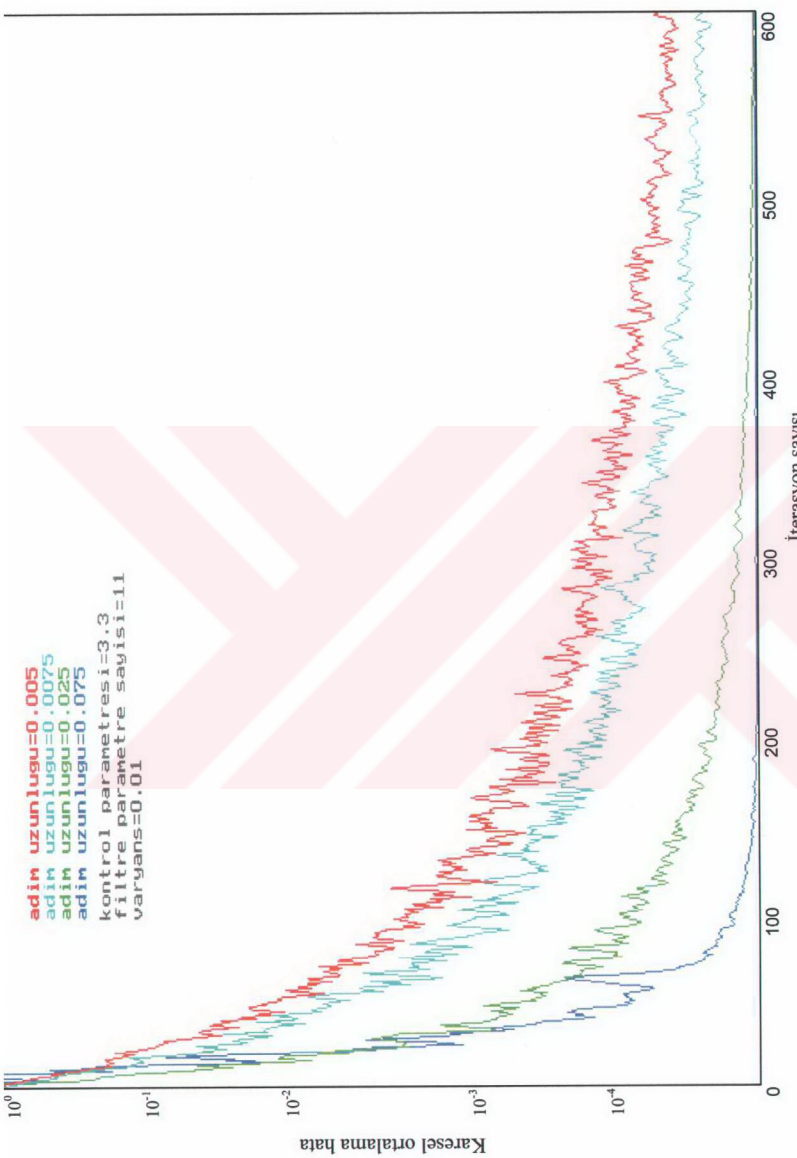
Şekil 5.16


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.1
filtre parametre sayisi=11
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

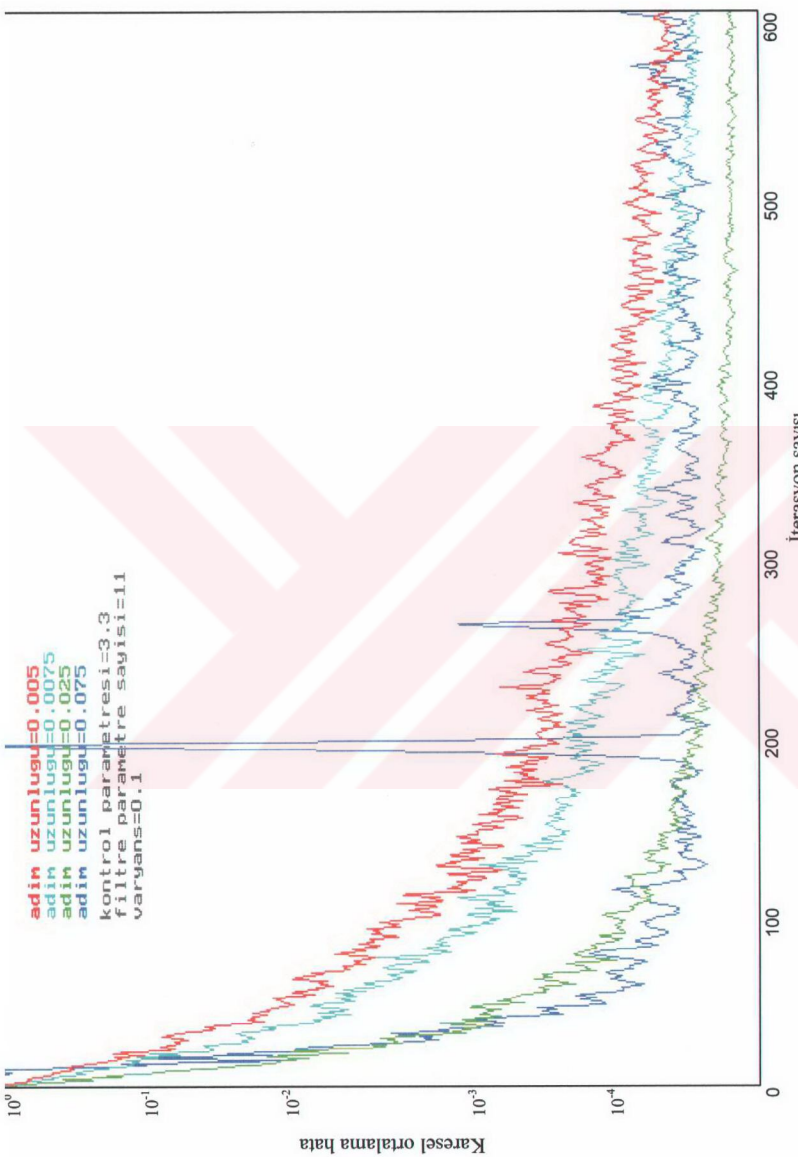
Şekil 5.17


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.3
filtre parametre sayisi=11
varyans=0.001
Ortalama karesel hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

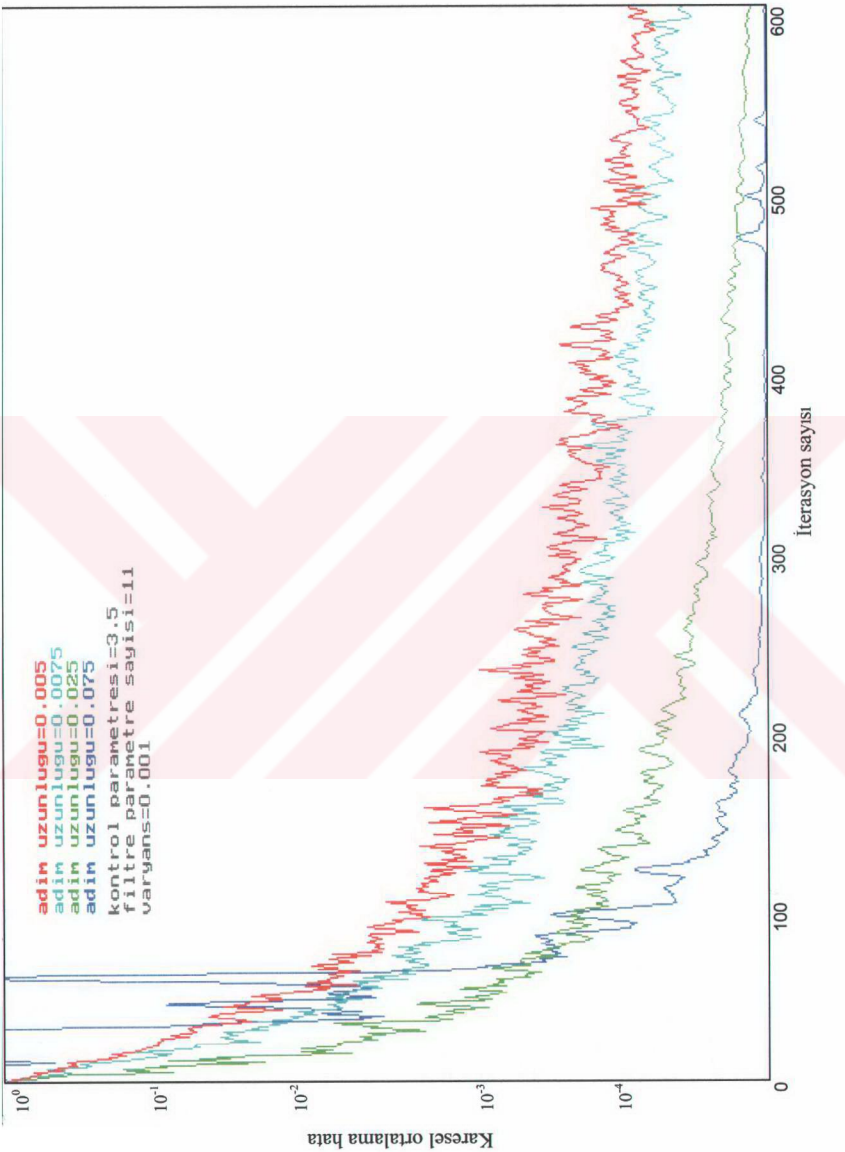
Şekil 5.18


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.3
filtre parametre sayisi=11
varyans=0.01
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

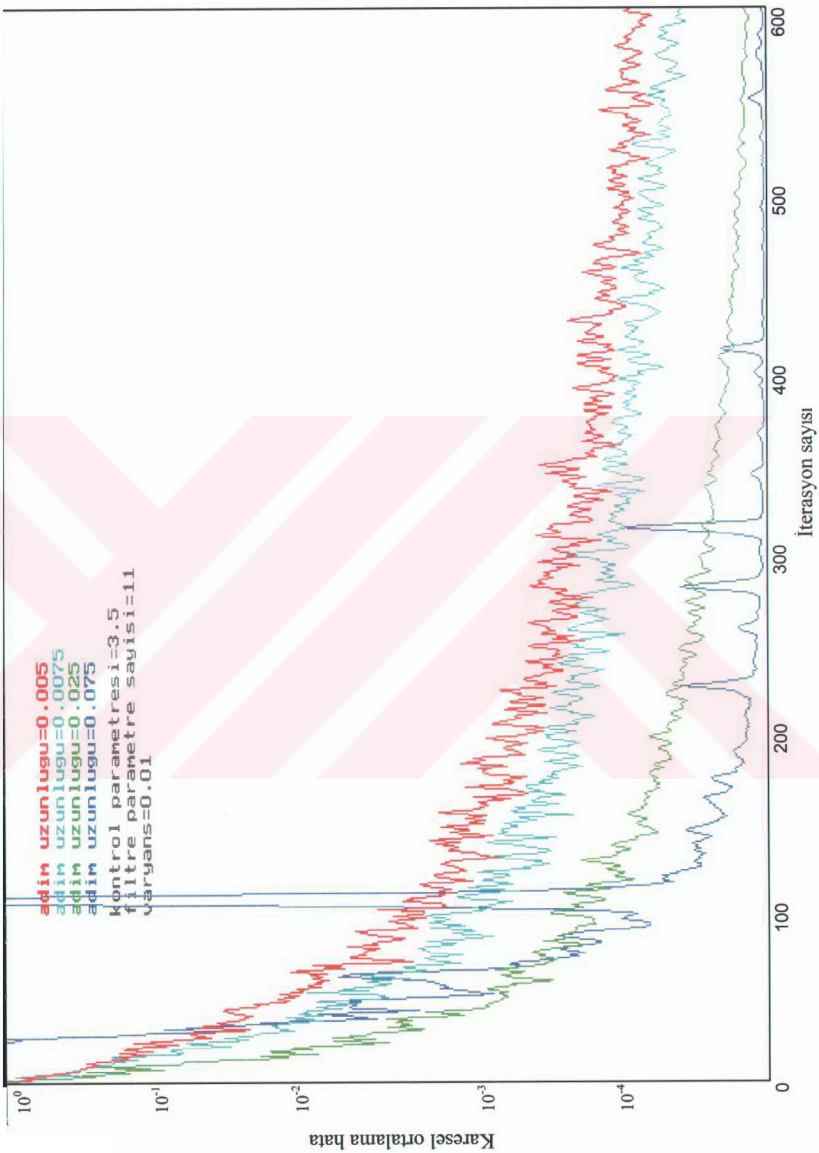
Şekil 5.19


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.3
filtre parametre sayisi=11
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

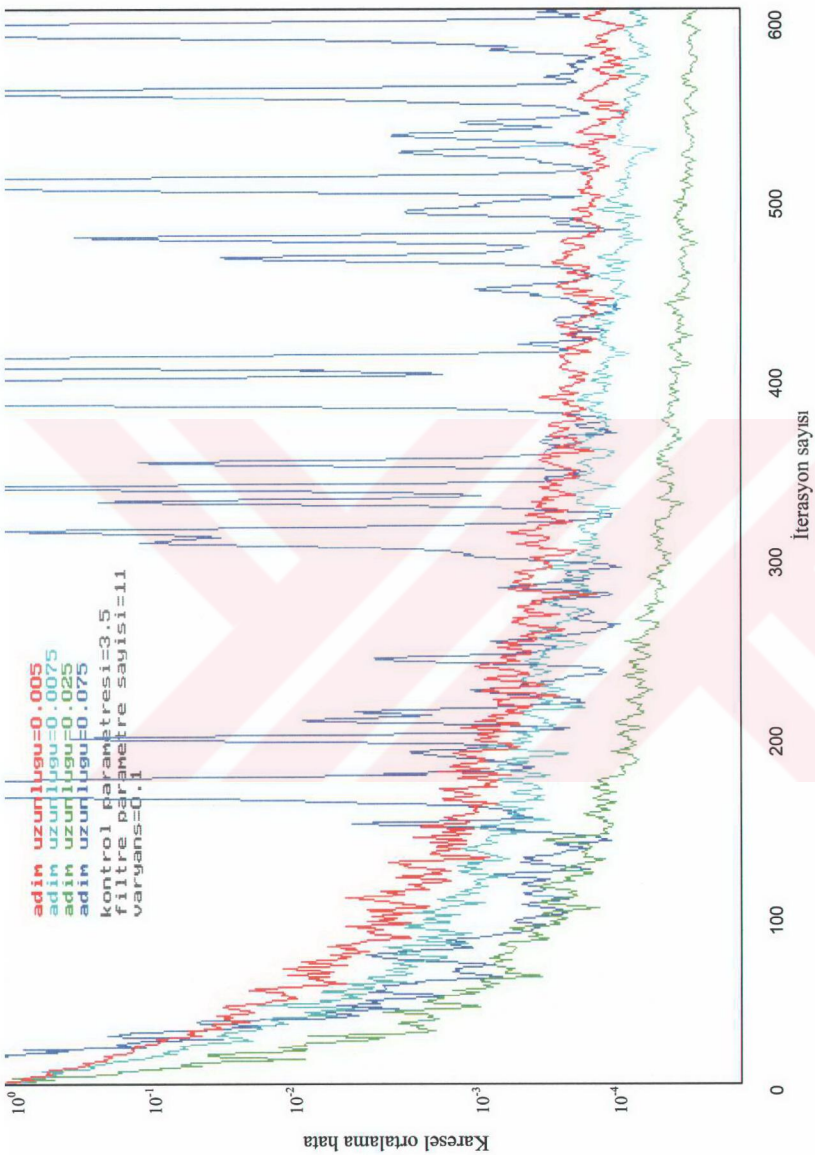
Şekil 5.20


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.5
filtre parametre sayisi=11
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

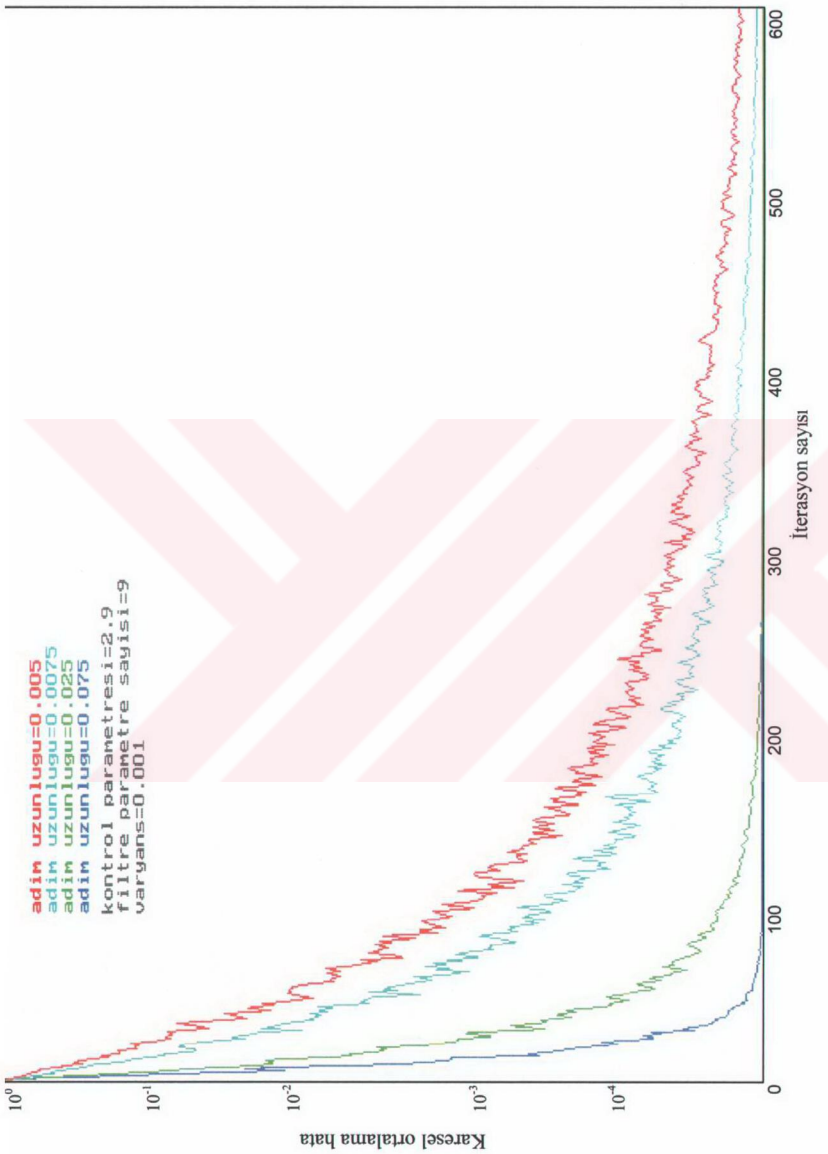
Şekil 5.21


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
Kontrol parametresi=3.5
filtre parametre sayisi=11
yargans=0.01
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

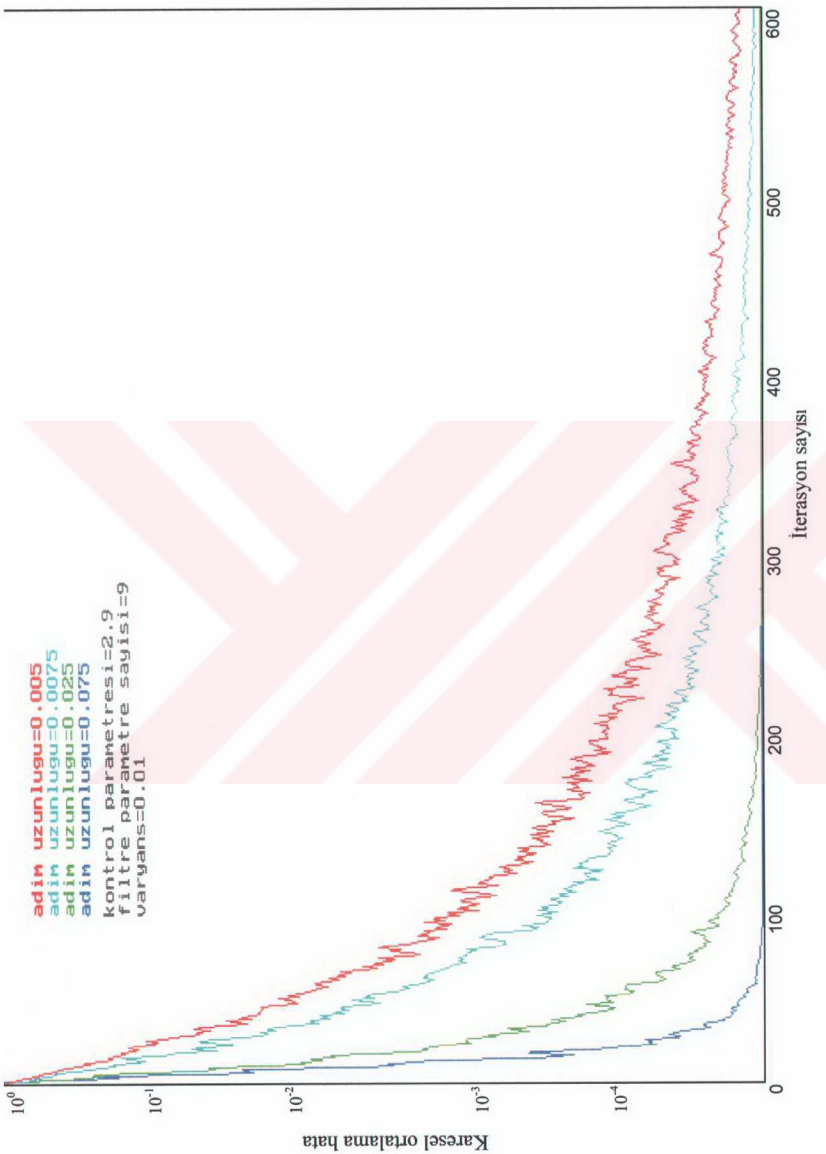
Şekil 5.22


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.5
filtre parametre sayisi=11
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

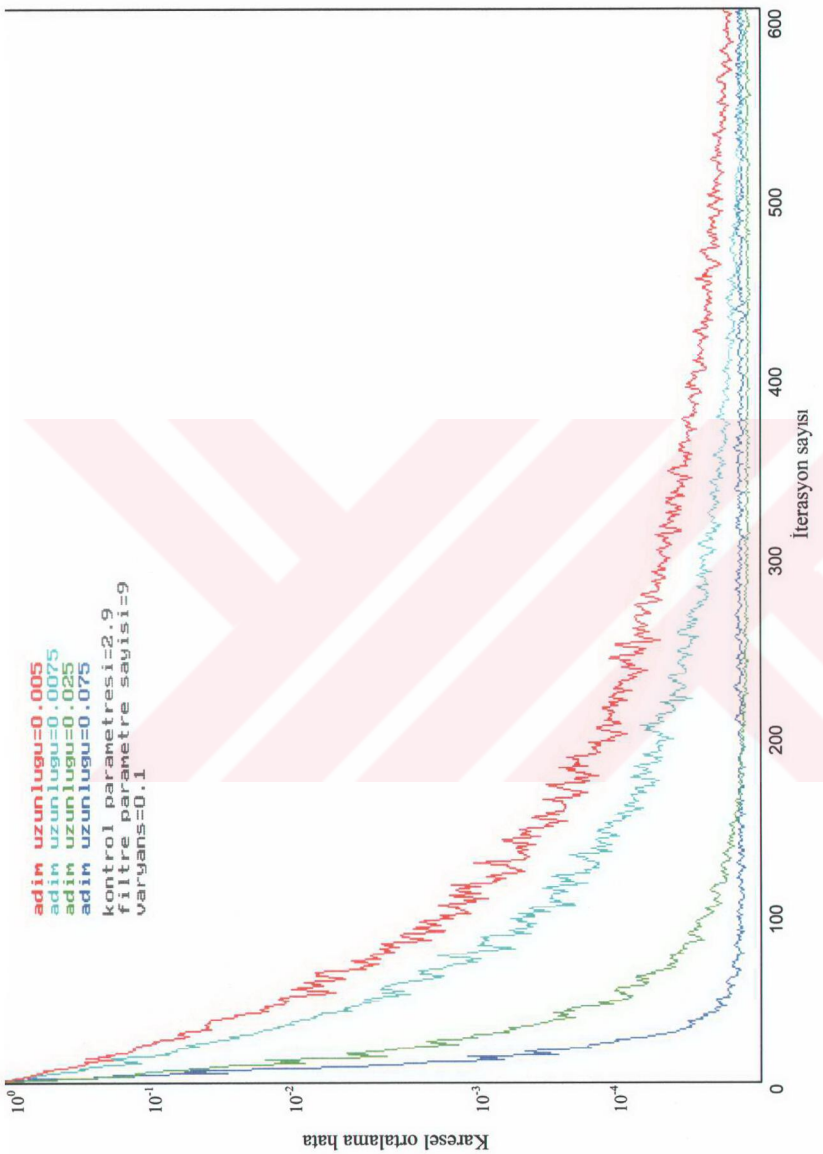
Şekil 5.23


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=2.9
filtre parametre sayisi=9
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

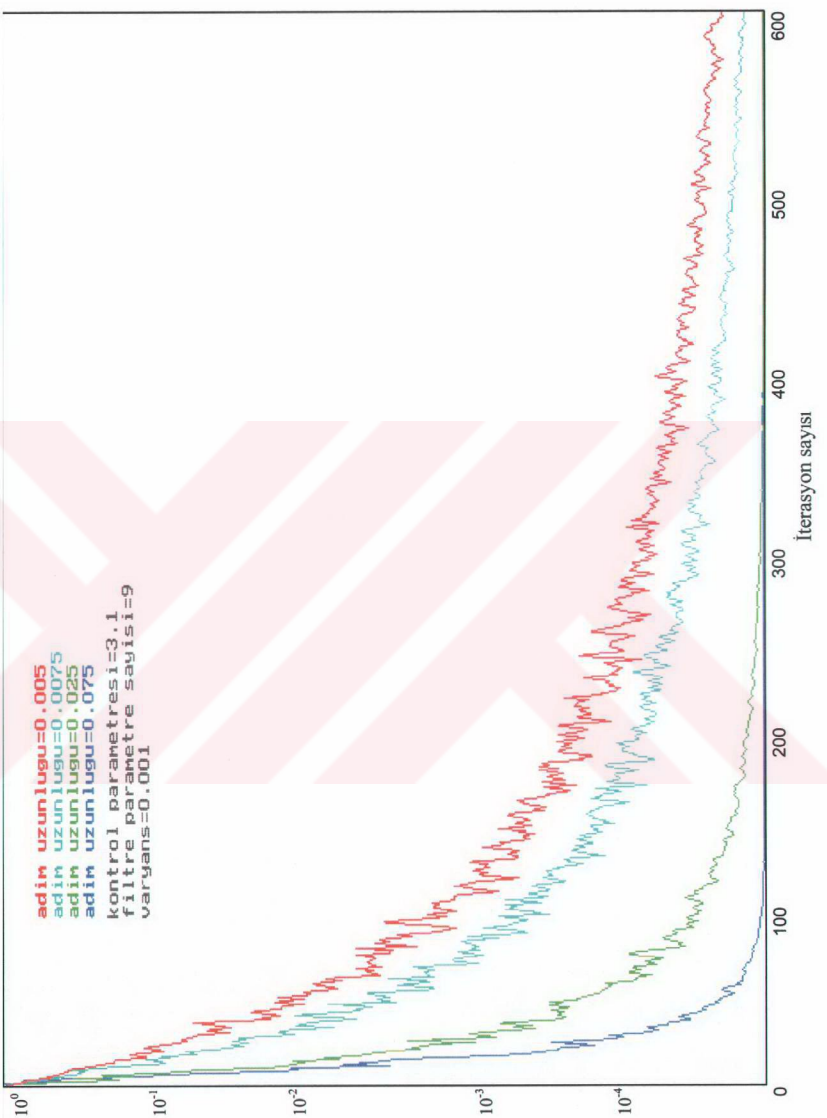
Şekil 5.24



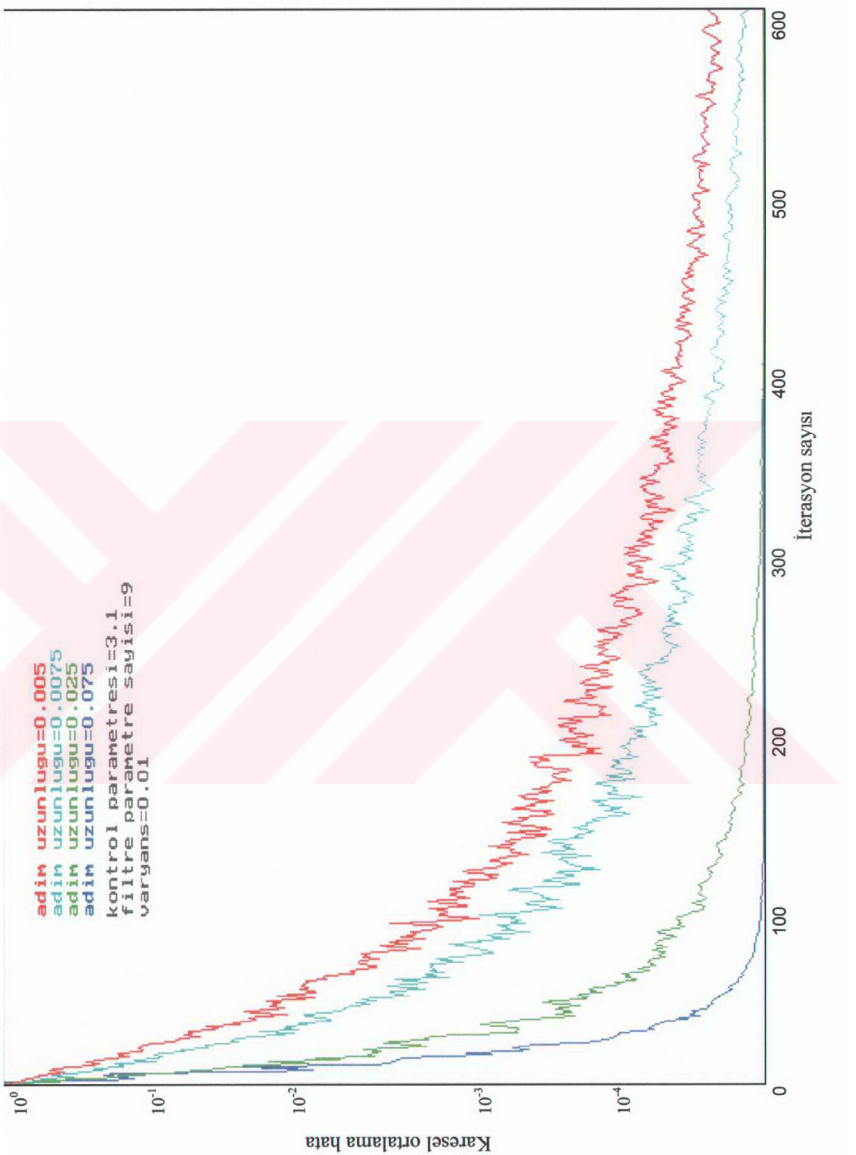
Şekil 5.25


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=2.9
filtre parametre sayisi=9
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

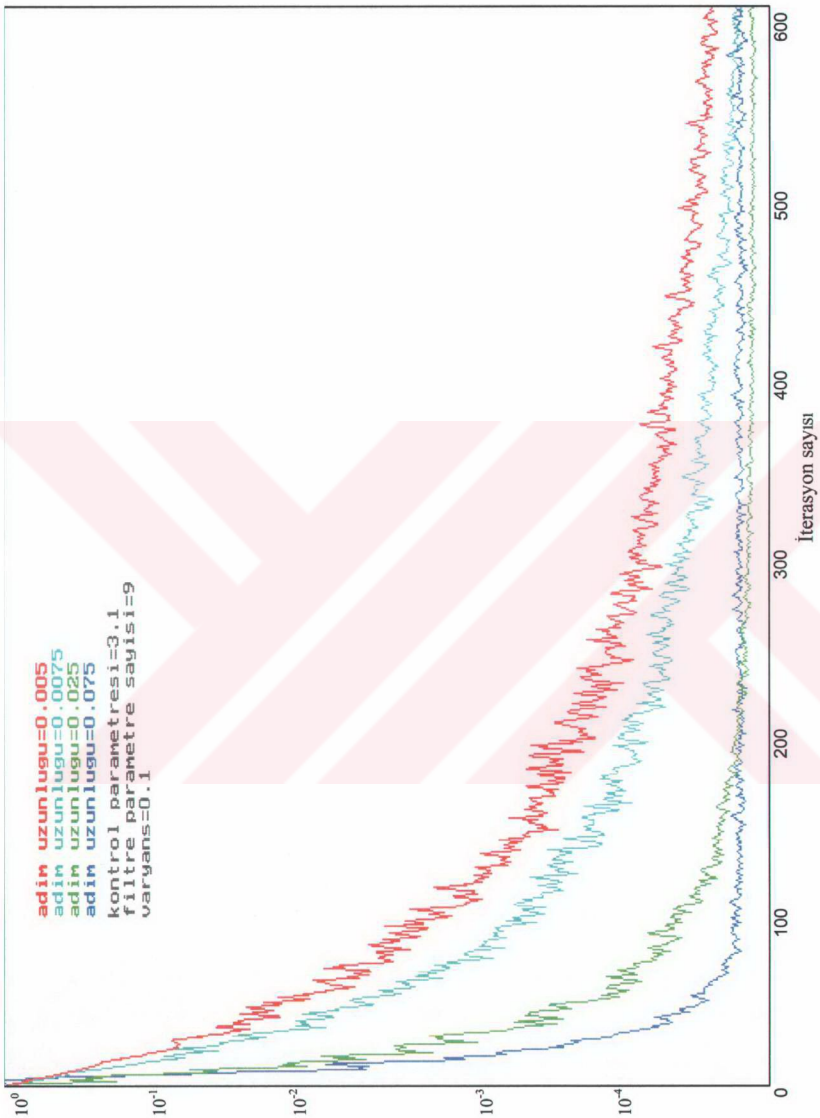
Şekil 5.26


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.1
filtre parametre sayisi=9
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

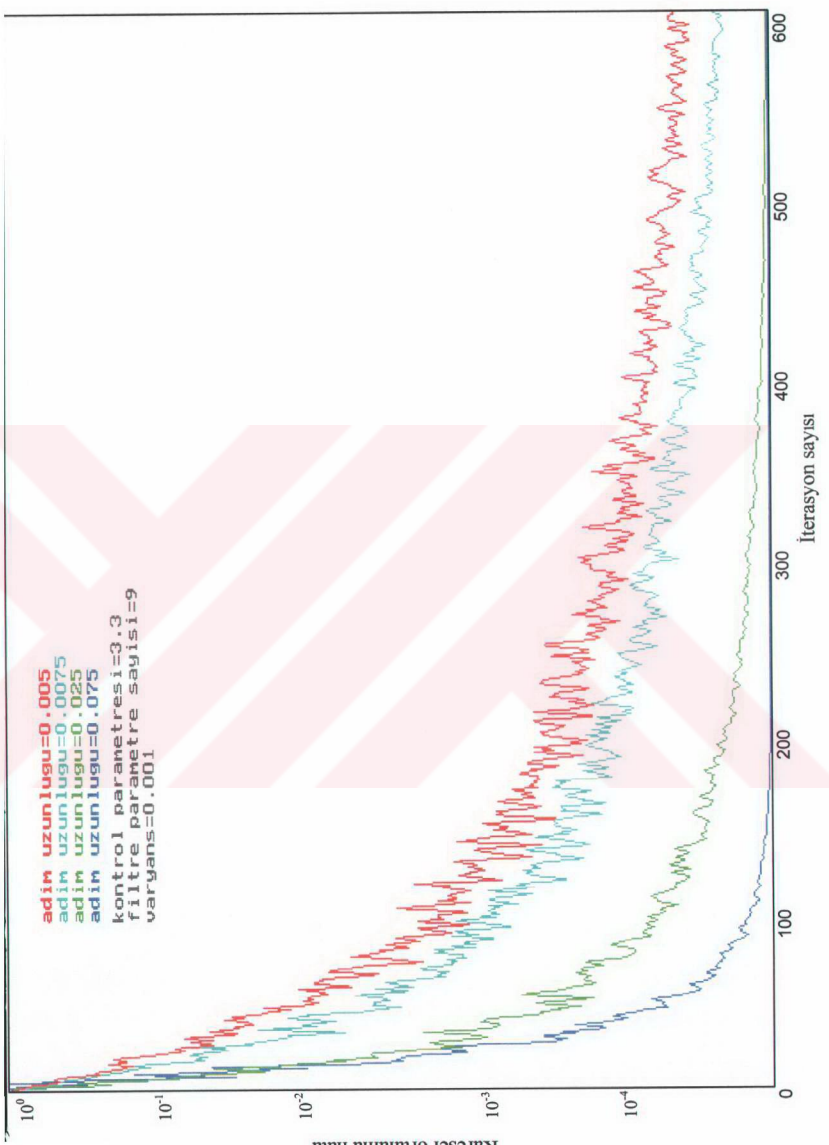
Şekil 5.27


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.1
filtre parametre sayisi=9
varyans=0.01
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

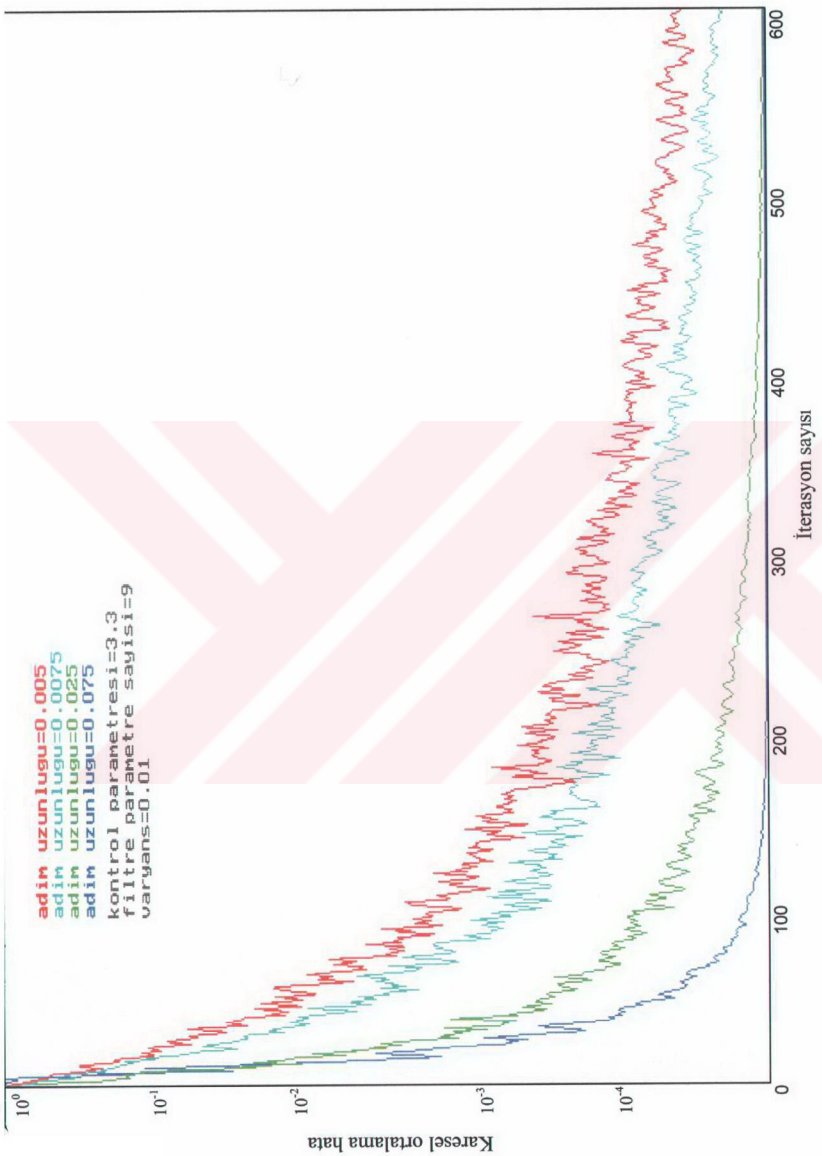
Şekil 5.28


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.1
filtre parametre sayisi=9
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

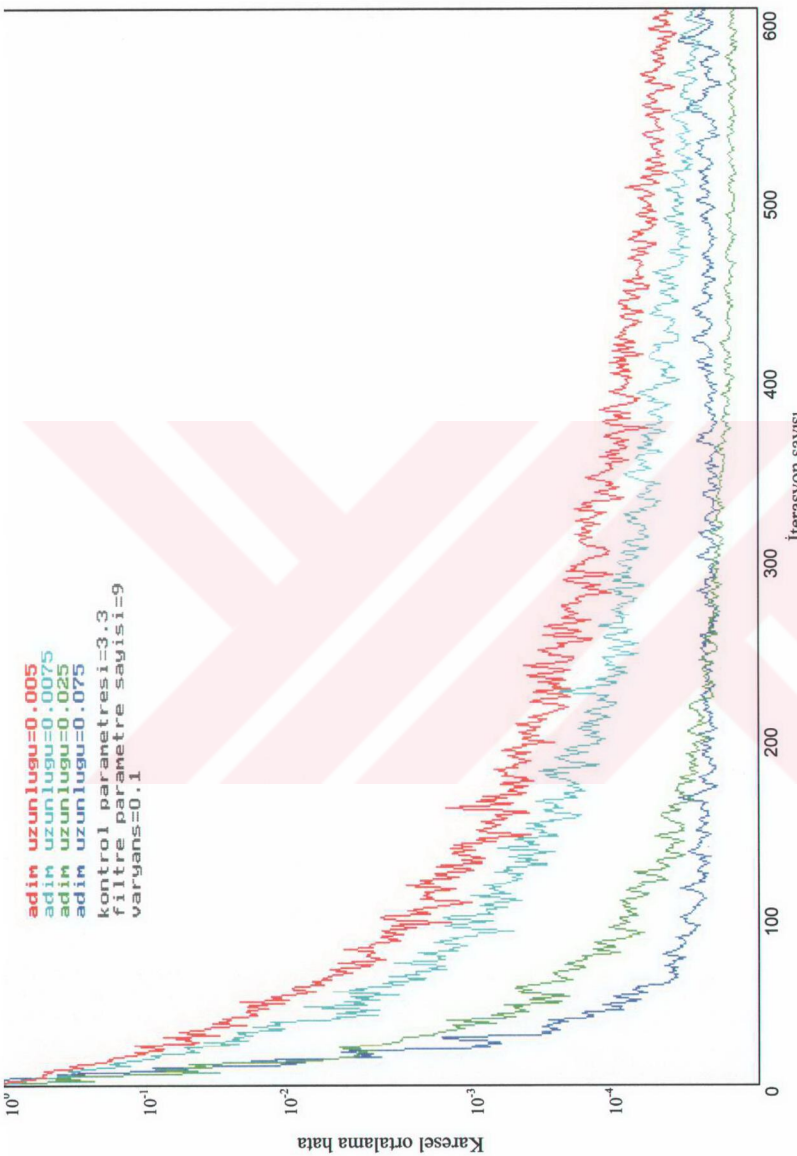
Şekil 5.29


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.3
filtre parametre sayisi=9
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

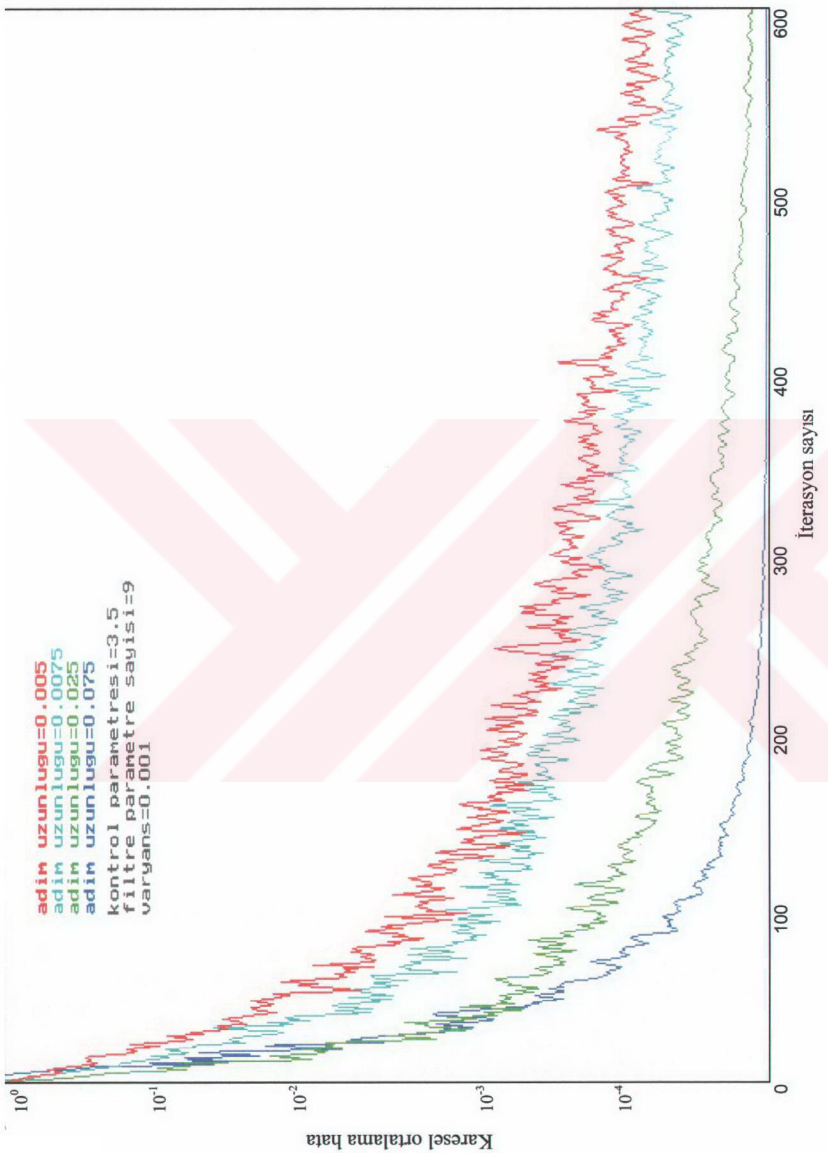
Şekil 5.30


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.3
filtre parametre sayisi=9
varyans=0.01
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

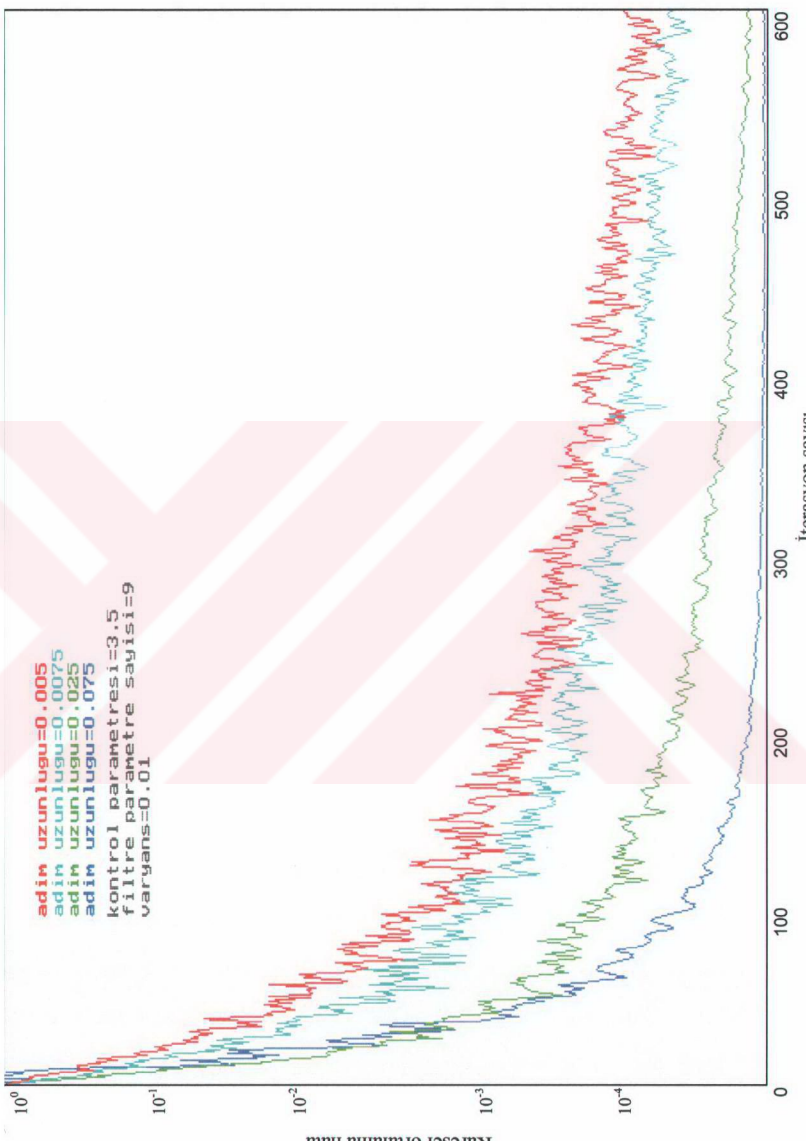
Şekil 5.31


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.3
filtre parametre sayisi=9
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

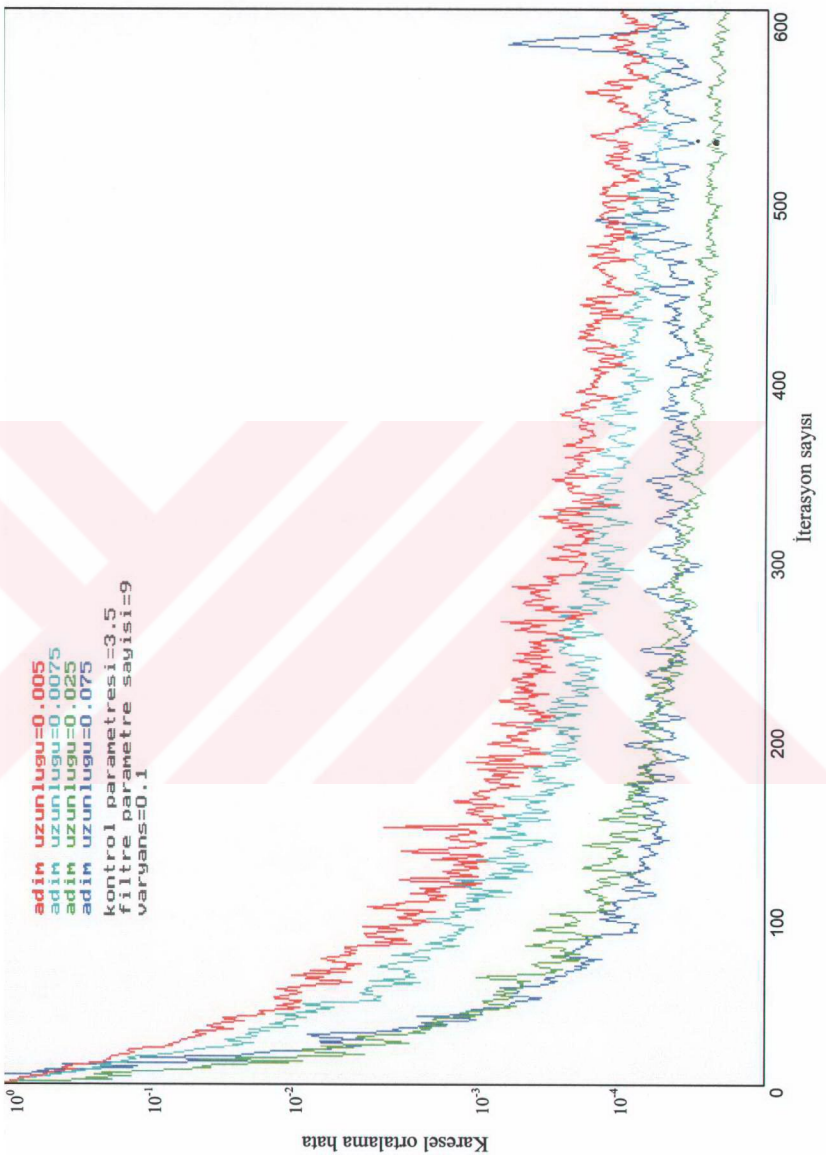
Şekil 5.32


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.5
filtre parametre sayisi=9
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

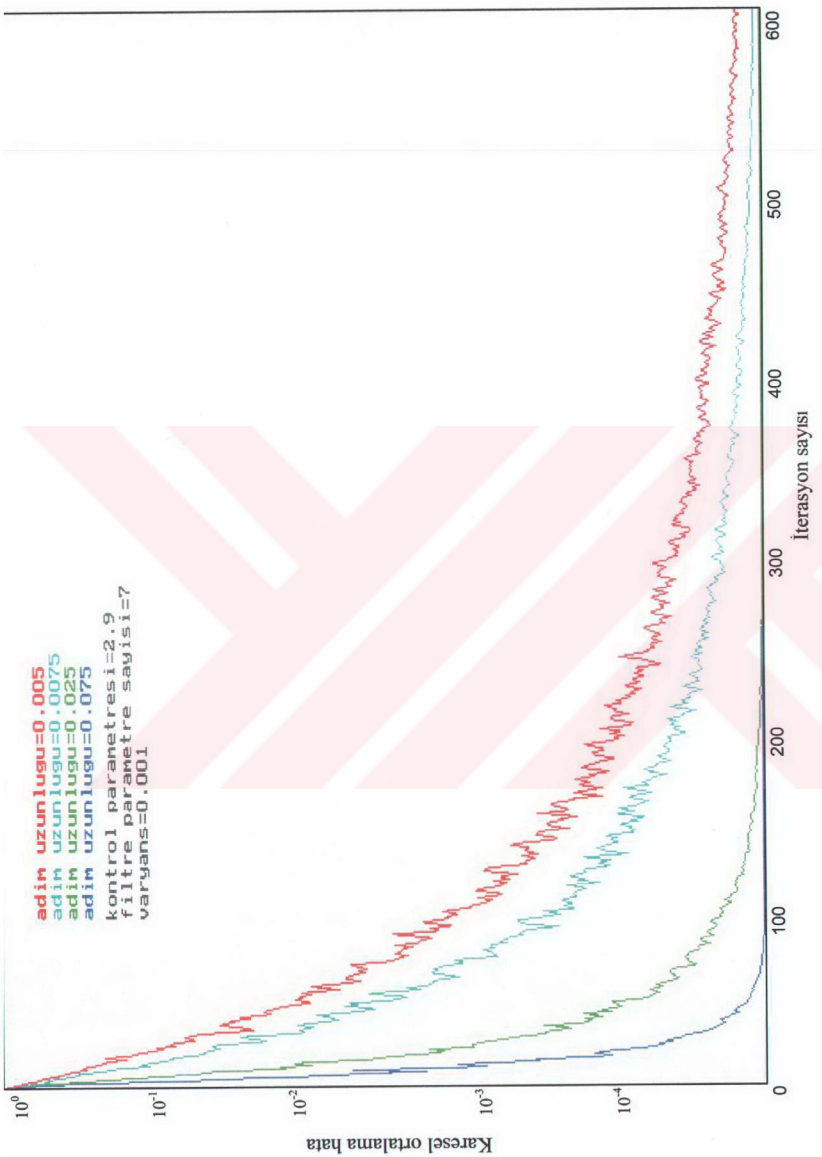
Şekil 5.33


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.5
filtre parametre sayisi=9
varyans=0.01
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

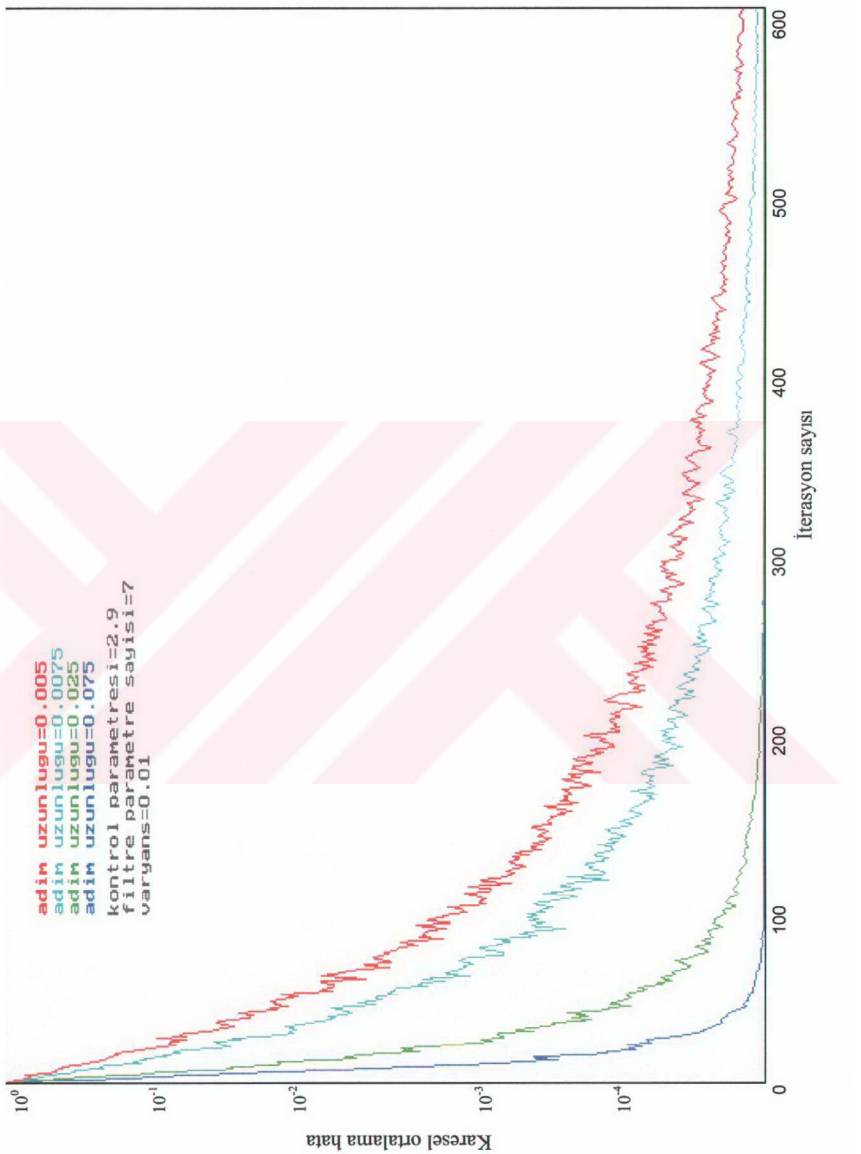
Şekil 5.34


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.5
filtre parametre sayisi=9
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

Şekil 5.35


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=2.9
filtre parametre sayisi=7
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

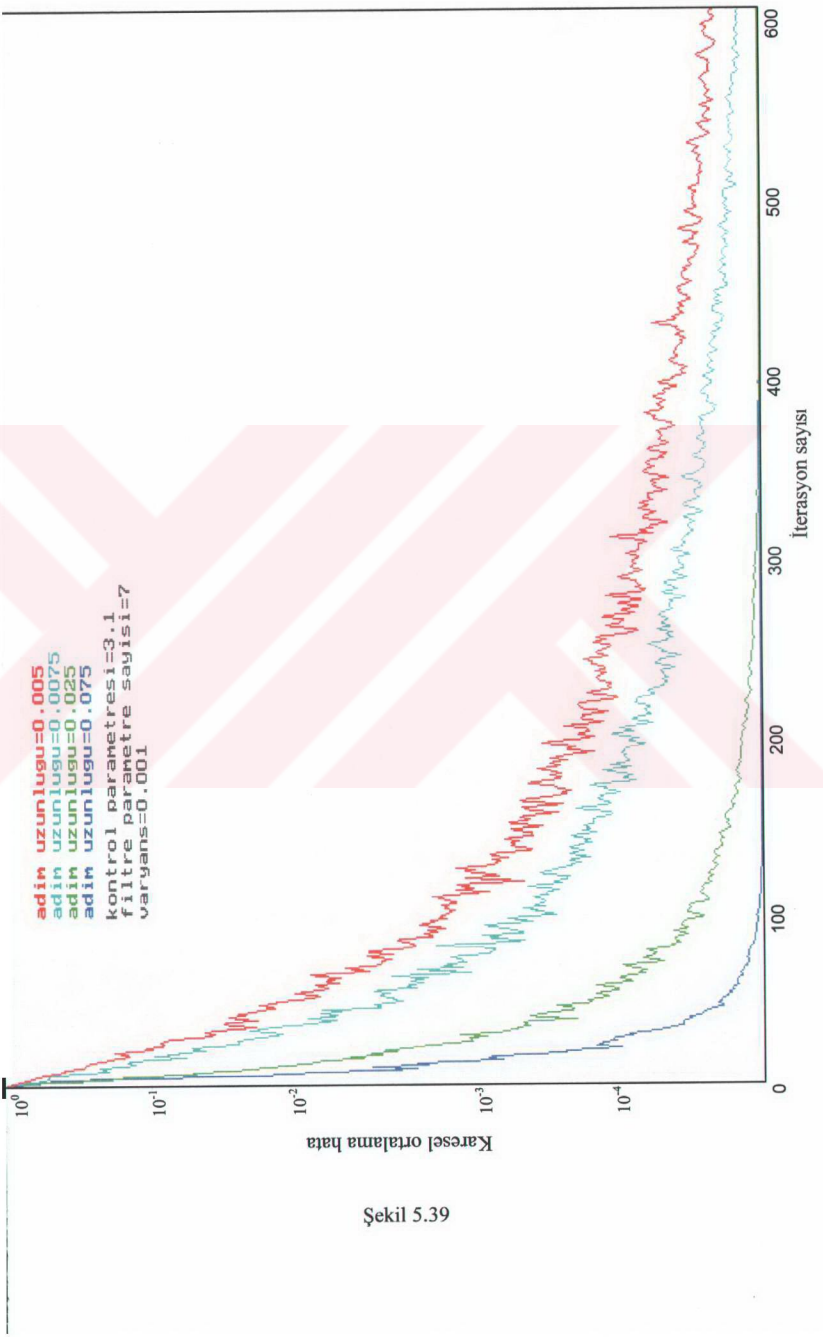
Şekil 5.36


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=2.9
filtre parametre sayisi=7
varyans=0.01
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

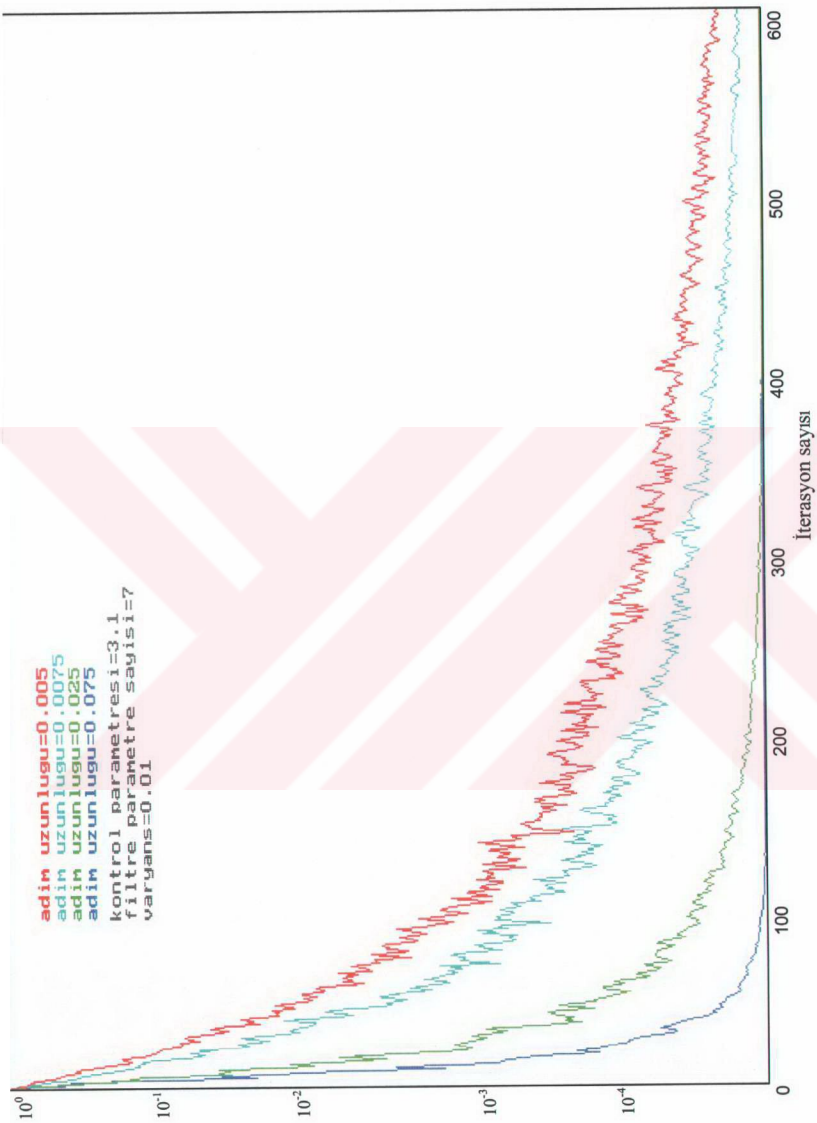
Şekil 5.37

adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=2.9
filtre parametre sayisi=7
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

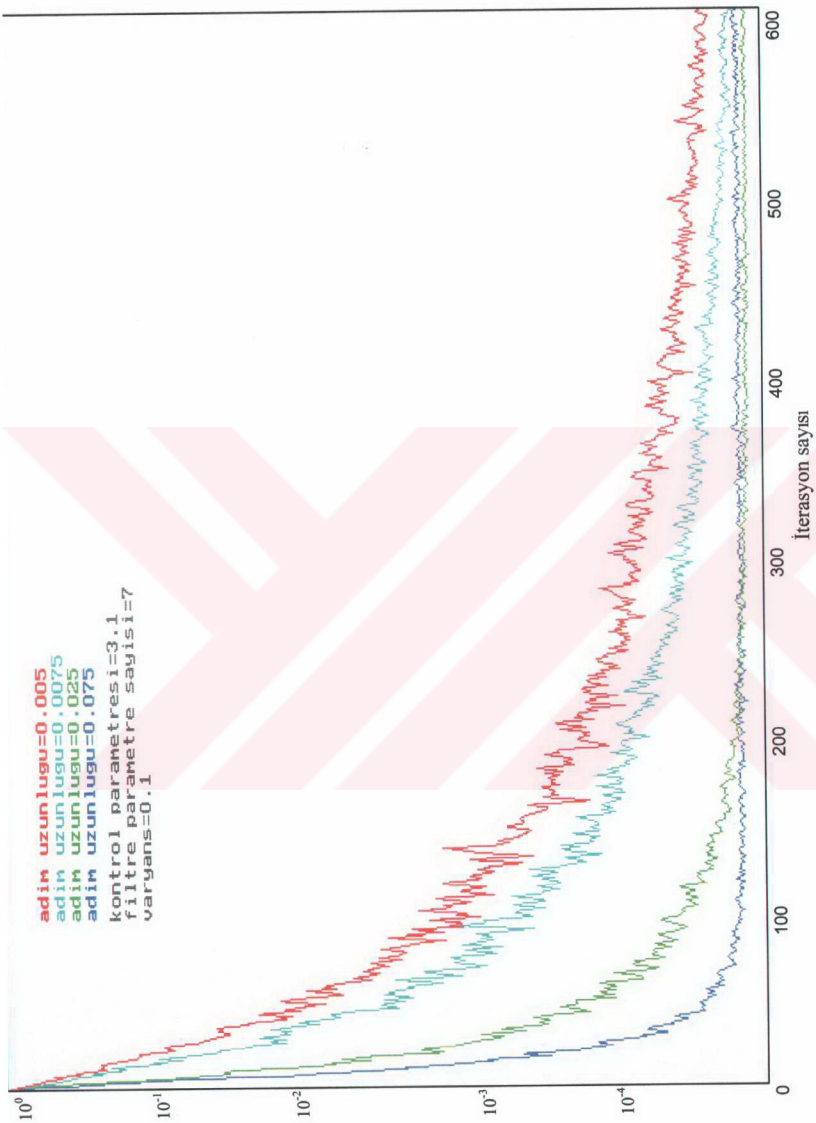
Şekil 5.38


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.1
filtre parametre sayisi=7
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

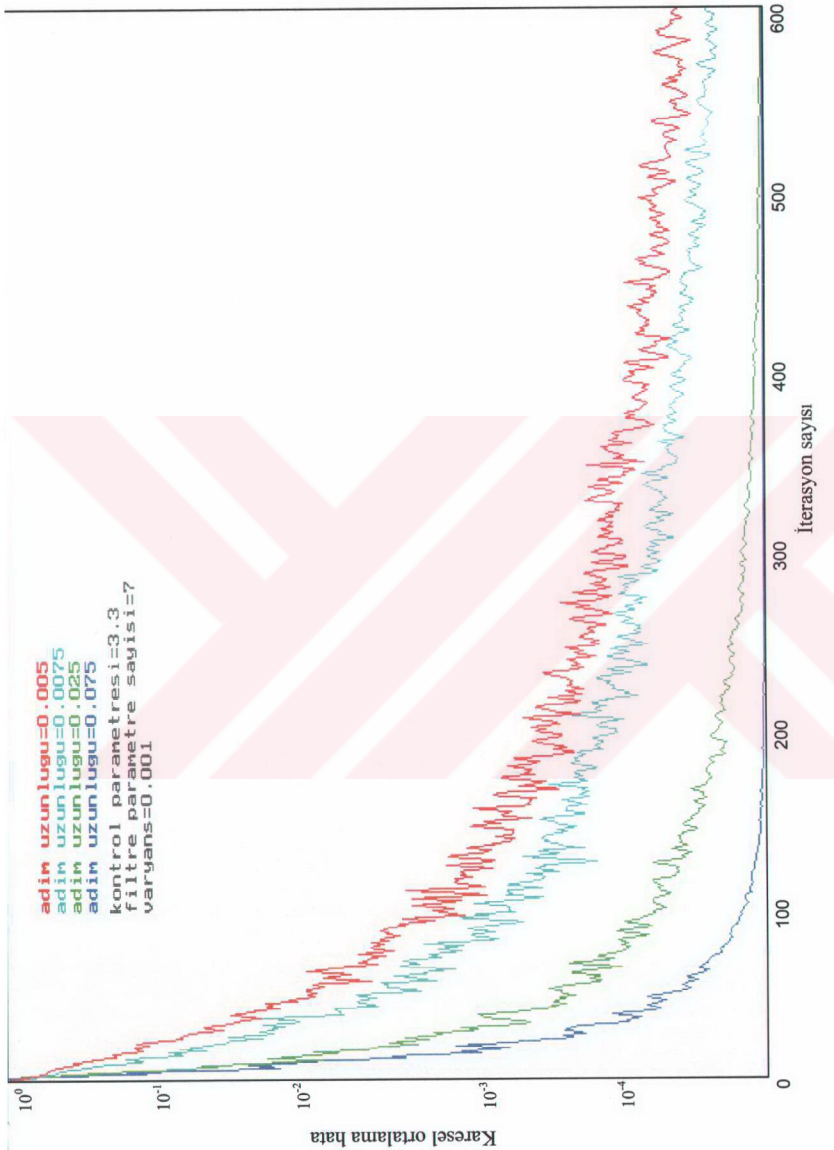
Şekil 5.39


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.1
filtre parametre sayisi=7
varyans=0.01
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

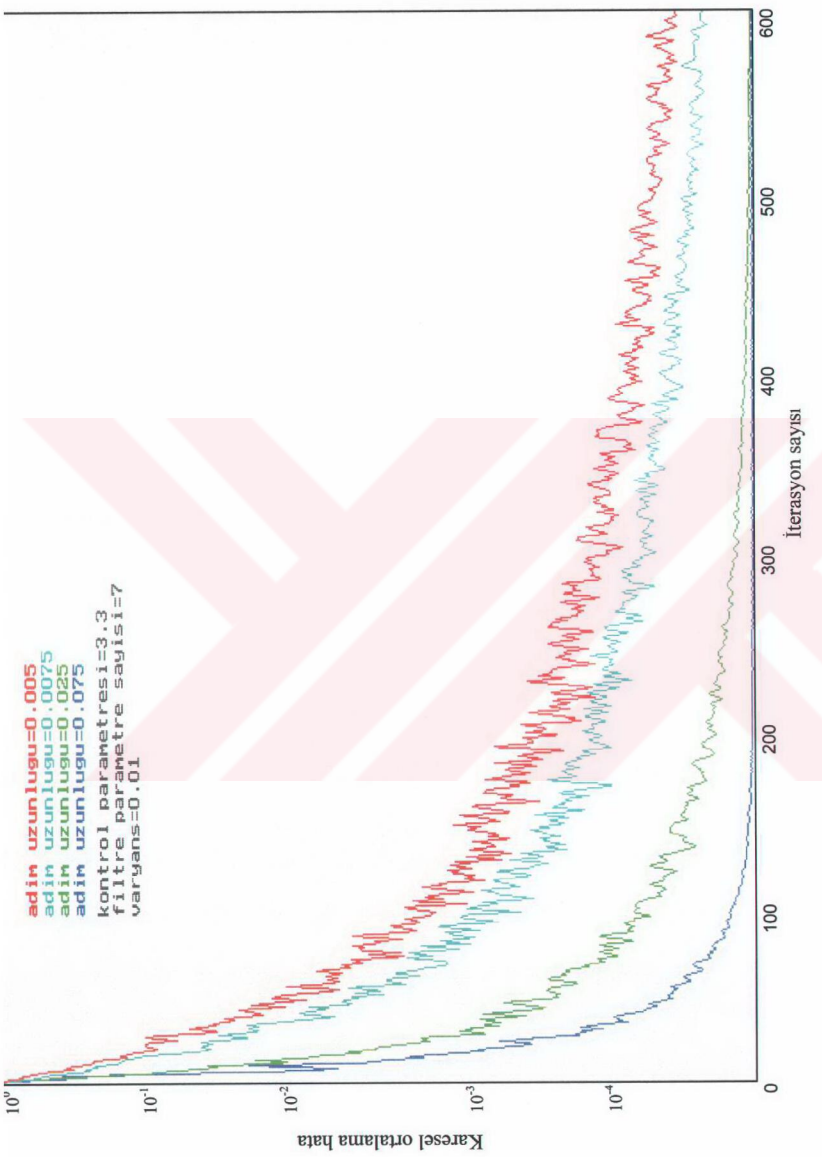
Şekil 5.40


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.1
filtre parametre sayisi=7
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

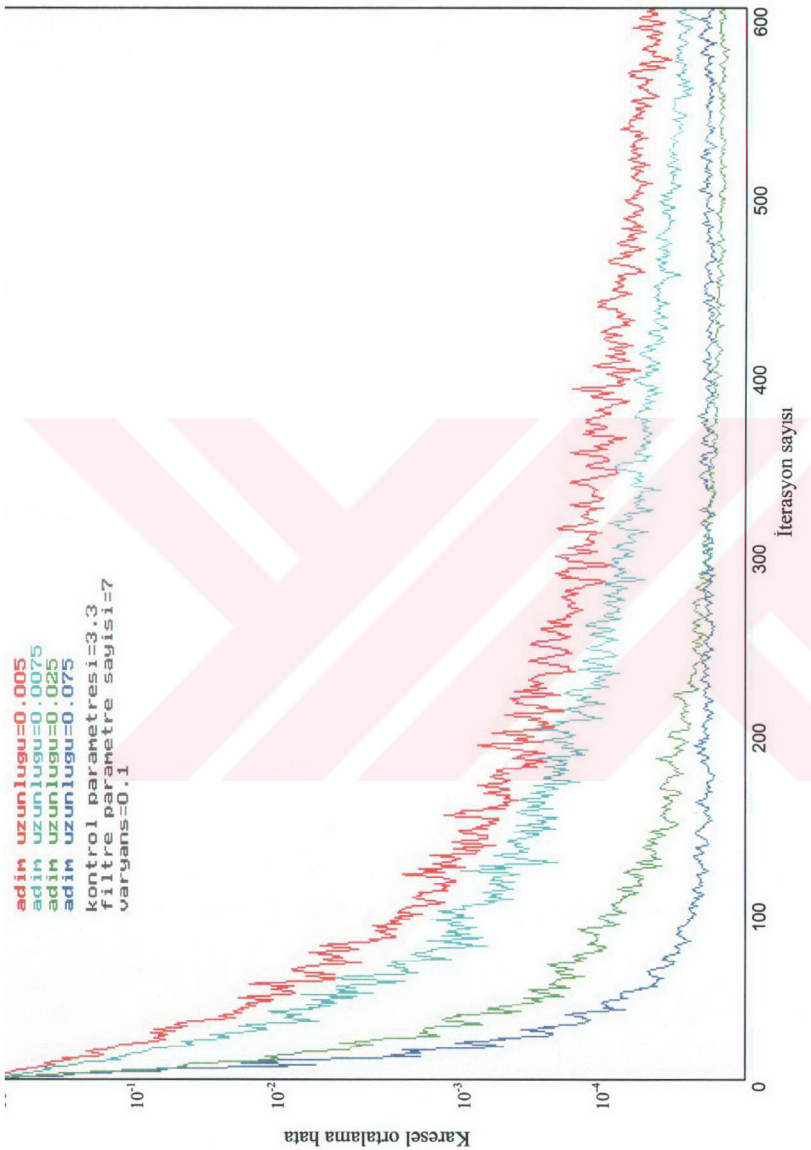
Şekil 5.41


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.3
filtre parametre sayisi=7
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

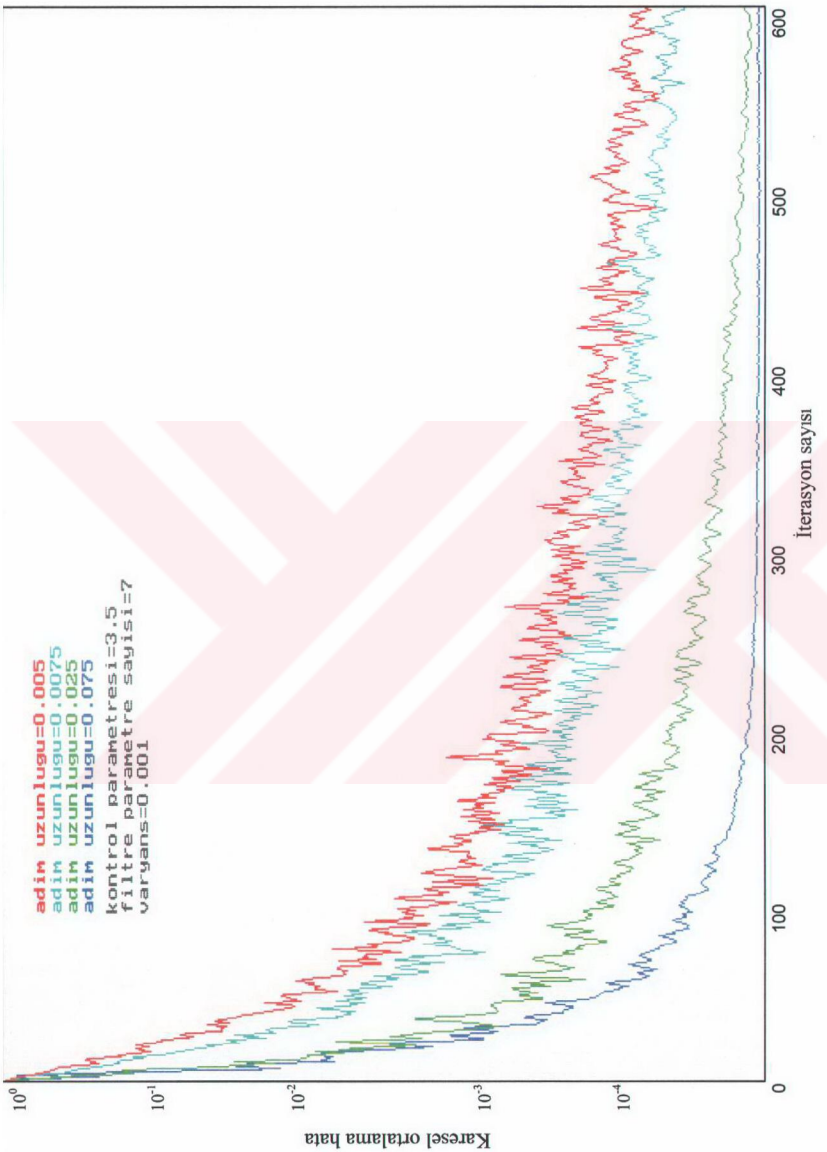
Şekil 5.42


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.3
filtre parametre sayisi=7
varyans=0.01
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

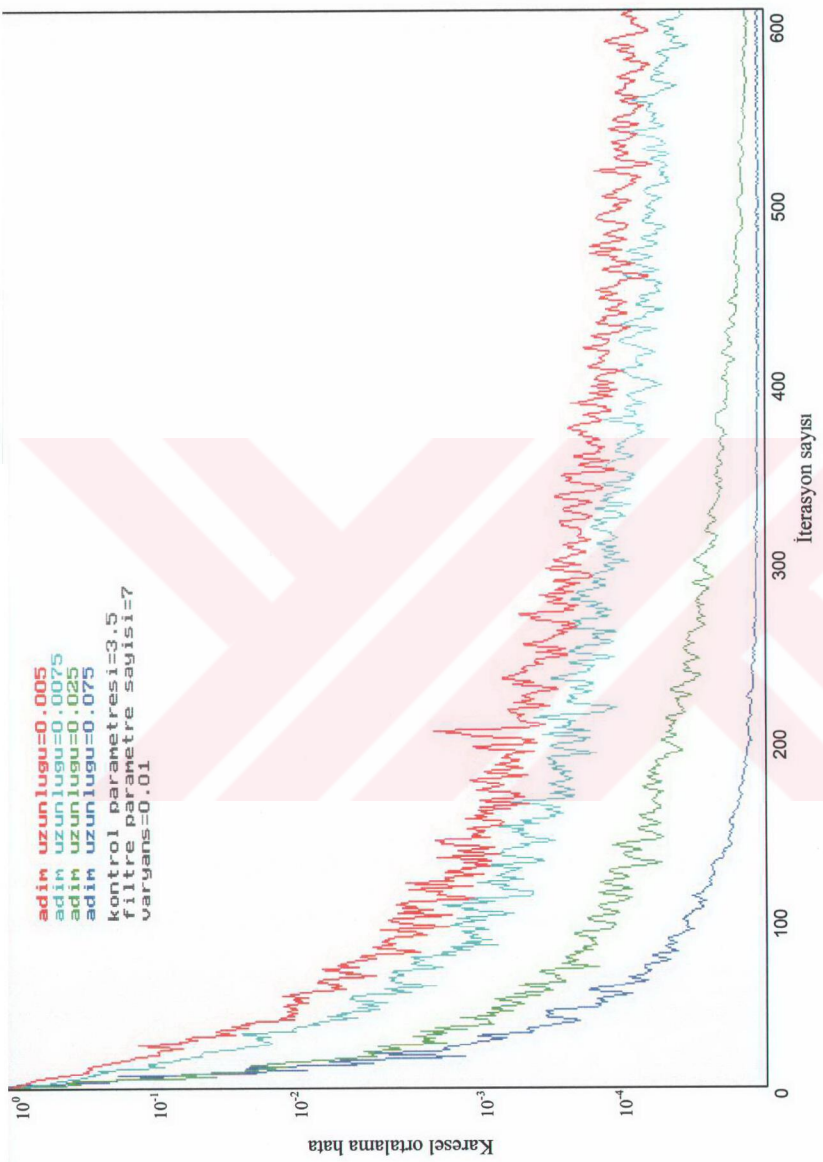
Şekil 5.43


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.3
filtre parametre sayisi=7
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
0
100
200
300
400
500
600
10^{-1}
10^{-2}
10^{-3}
10^{-4}

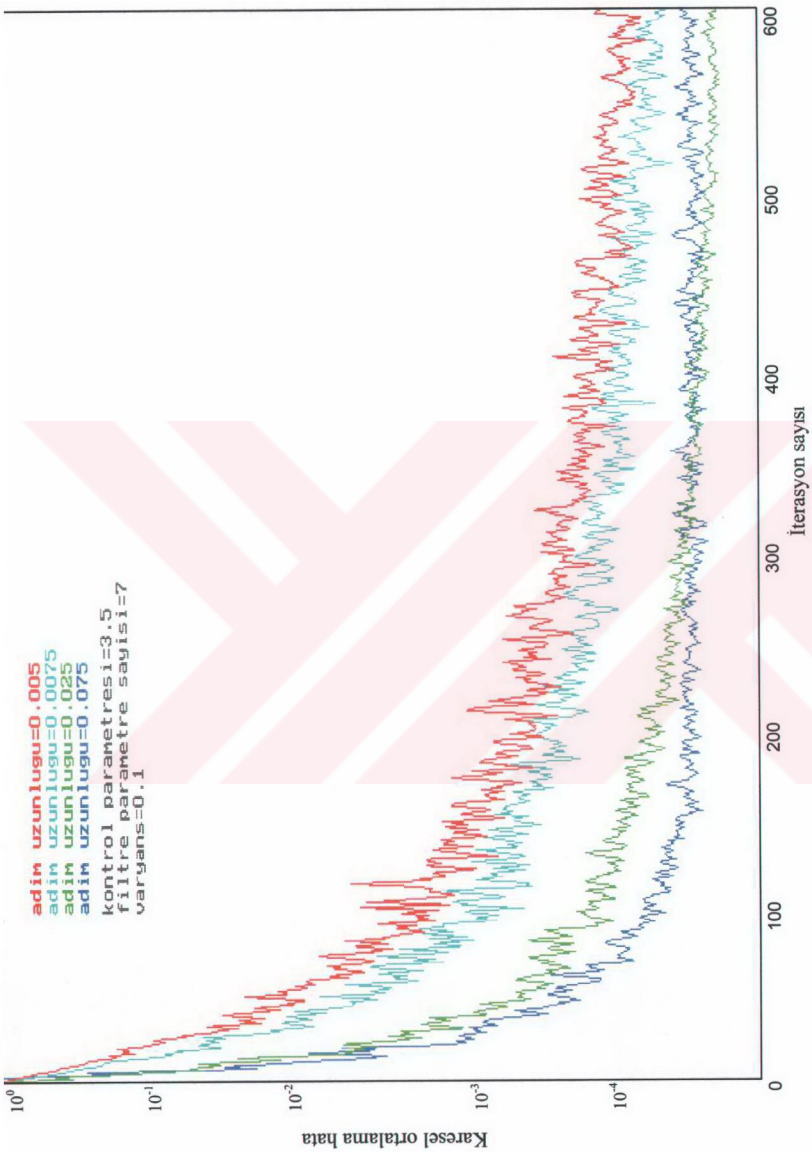
Şekil 5.44


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.5
filtre parametre sayisi=7
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

Şekil 5.45


adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.5
filtre parametre sayisi=7
varyans=0.01
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

Şekil 5.46

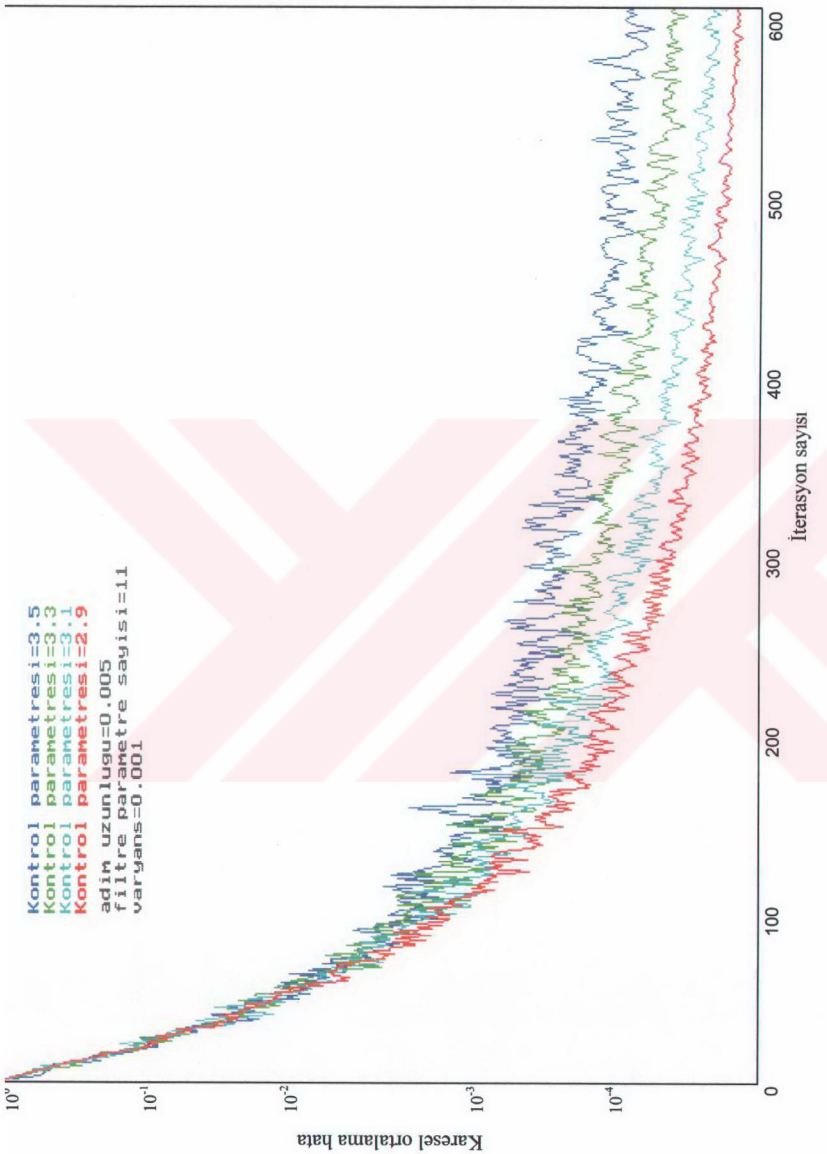

adim uzunlugu=0.005
adim uzunlugu=0.0075
adim uzunlugu=0.025
adim uzunlugu=0.075
kontrol parametresi=3.5
filtre parametre sayisi=7
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

Şekil 5.47

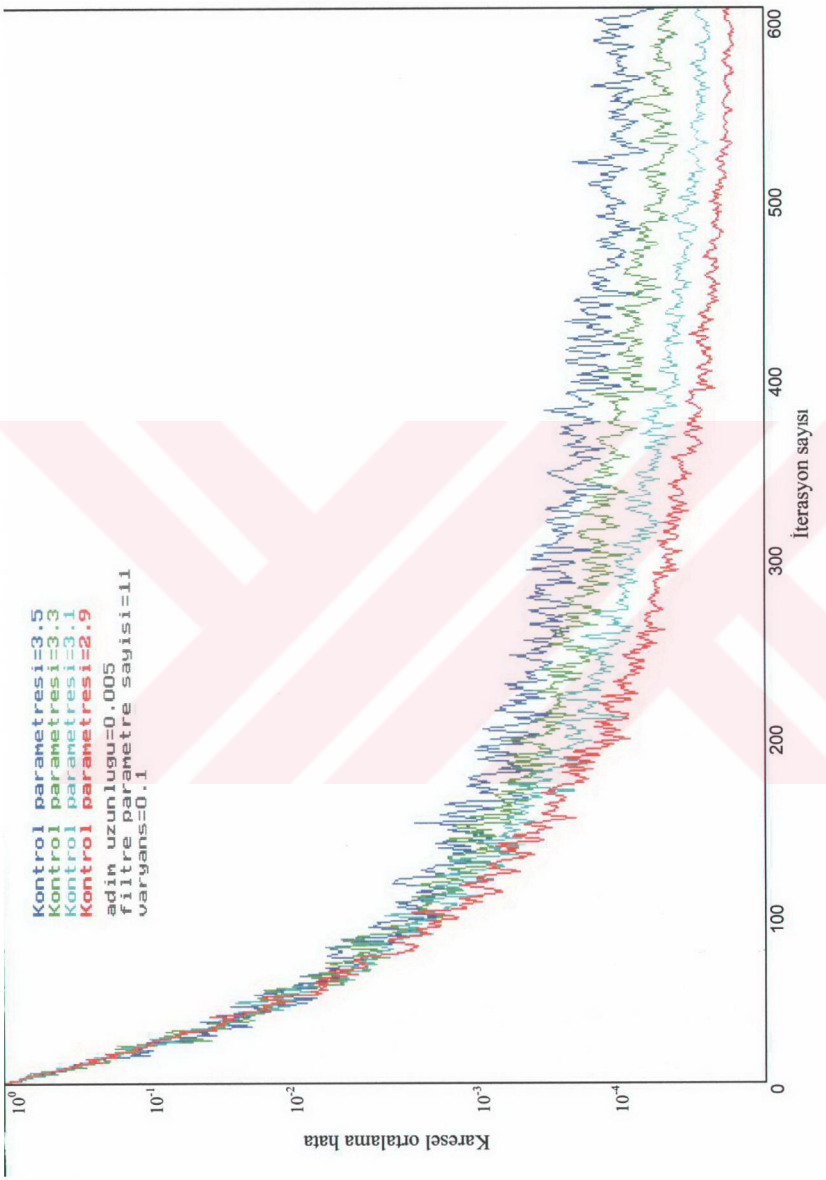
Adım uzunluğu değişimi esas alınarak, elde edilen eğriler (Şekil 5.9-5.47) incelendiğinde şu gözlemlere varmak mümkündür:

Adım uzunluğu azaldıkça karesel ortalama hatanın yakınsama hızı azalmaktadır (Şekil 5.9-5.47). Bu da kullanılan algoritmanın öğrenme durumundan durağan duruma geçişini geciktirmektedir (Şekil 5.11). Bununla birlikte karesel ortalama hatanın durağan durum değeri gittikçe azalmaktadır (Şekil 5.9, 5.24, 5.36). Ayrıca eğrilerdeki dalgalanmalar daha çok artmaktadır (Şekil 5.9-5.47). Adım uzunluğunun küçük olması, gürültü varyansının görece yüksek olmasının yarattığı bozulmaları giderici yönde rol oynar (Şekil 5.19-5.22). Gürültü varyansı ve kanal kontrol parametresinin yüksek seçilmesi yanında, dengeleyicinin parametre sayısı da arttırılırsa karesel ortalama hatadaki dalgalanmalar artmaktadır. Adım uzunluğunun azaltılması, eğer kanal kontrol parametresi ve gürültü varyansı yüksek ise, karesel ortalama hatanın düşmesine neden olur ancak, durağan duruma geçiş süresinin de artmasına neden olmaktadır.

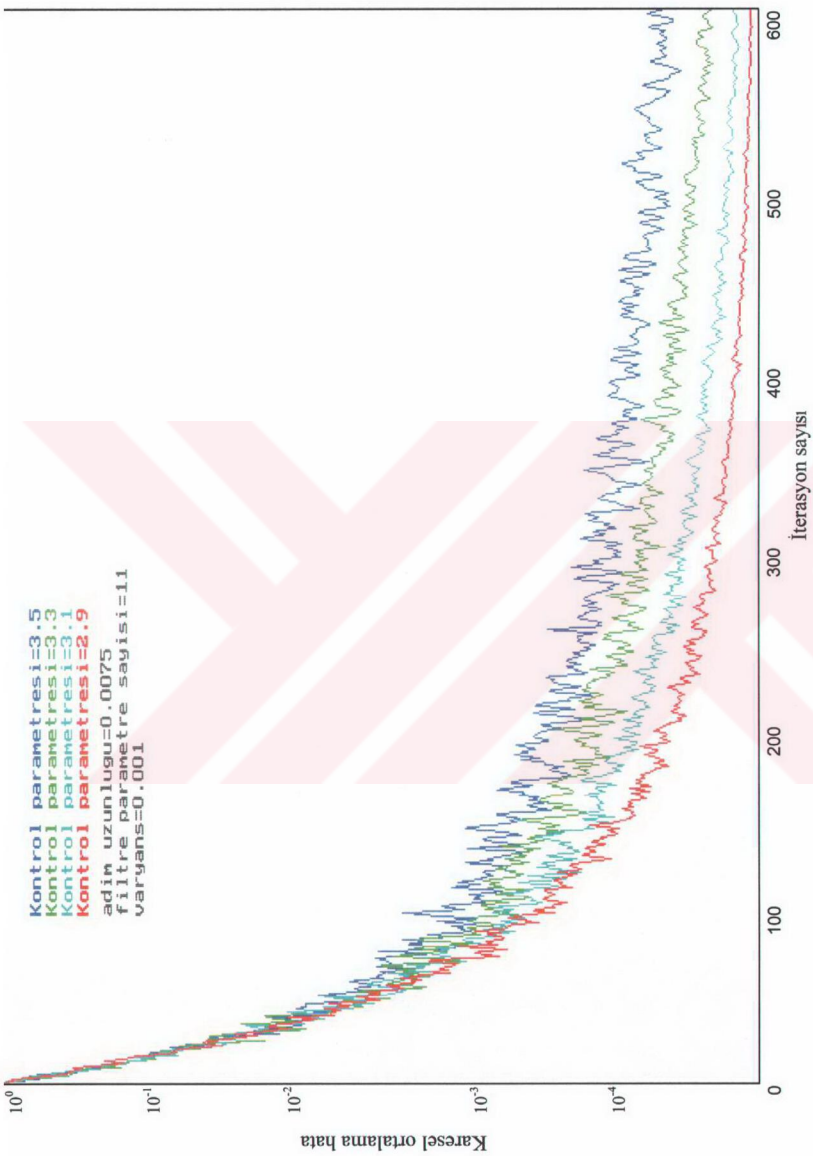
Adım uzunluğunun büyük olması karesel ortalama hatanın yakınsama hızını artırmaktadır (Şekil 5.9-5.47). Ancak durağan durum değeri yüksek kalmaktadır (Şekil 5.11, 5.13, 5.16, 5.22, 5.28, 5.34, 5.40). Adım uzunluğunun büyüklüğü yanında, gürültü varyansının artması hem eğrideki dalgalanmaları, hem de durağan durum değerini arttırmaktadır (Şekil 5.16-5.19). Bunların yanında kanal kontrol parametresi yüksek seçilirse, eğrideki dalgalanmalar çok artmakta ve iterasyon sayısı ne kadar arttırılırsa arttırılsın, gidermek olanaksızlaşmaktadır (Şekil 5.16-5.19). İterasyon sayısını arttırmak da aynı zamanda dengeleyicinin öğrenme durumundan durağan duruma geçişini geciktirmektedir. Eğer adım uzunluğu ve kanal kontrol parametresi büyükse, gürültü varyansı yüksek olması durumunda, dengeleyici parametre sayısının düşük tutulması gerekir (karşılaştırmak için Şekil 5.23 ve Şekil 5.47).


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.005
filtre parametre sayisi=11
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

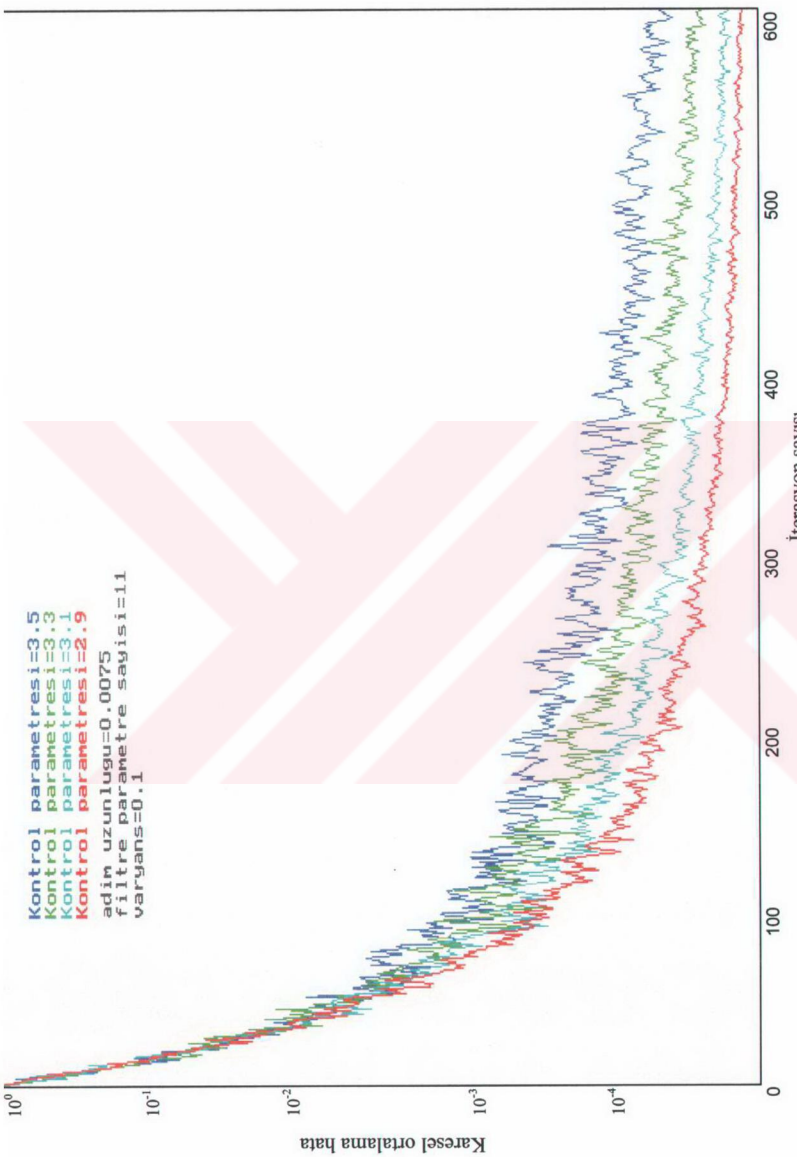
Şekil 5.48


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.005
filtre parametre sayisi=11
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

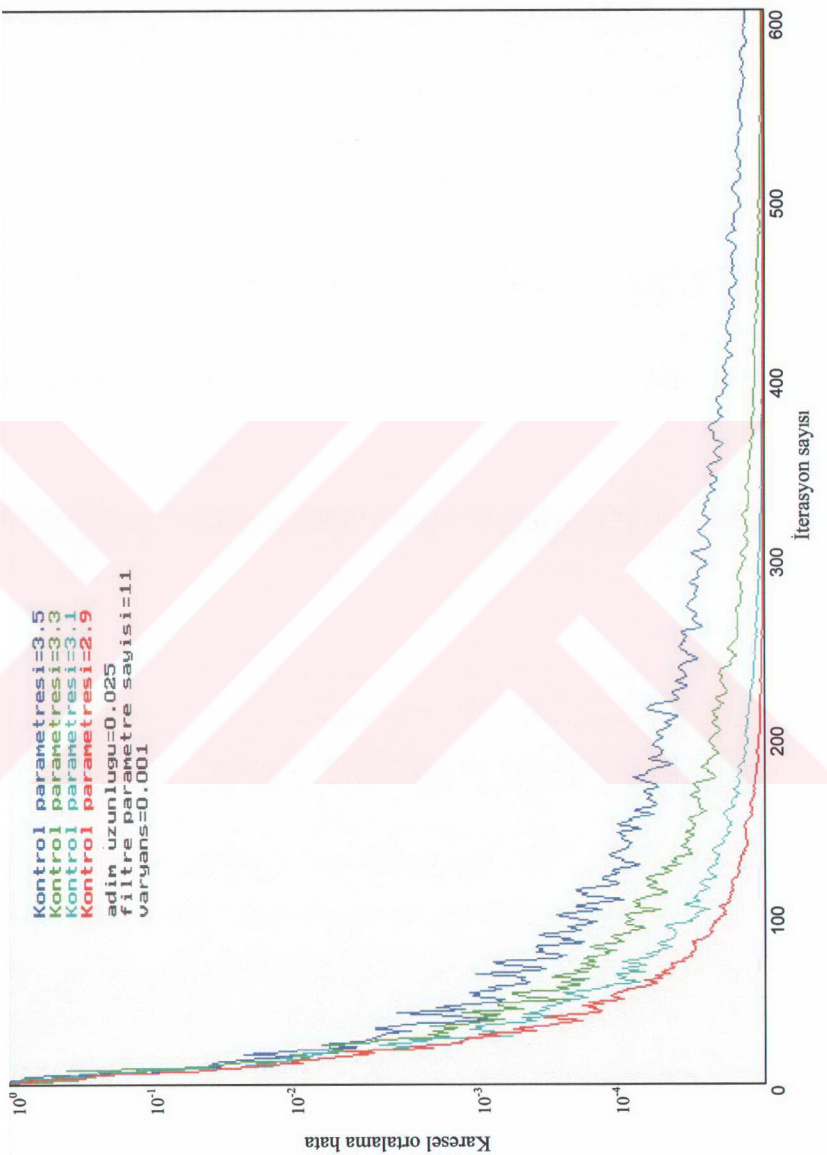
Şekil 5.49


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.0075
filtre parametre sayisi=11
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

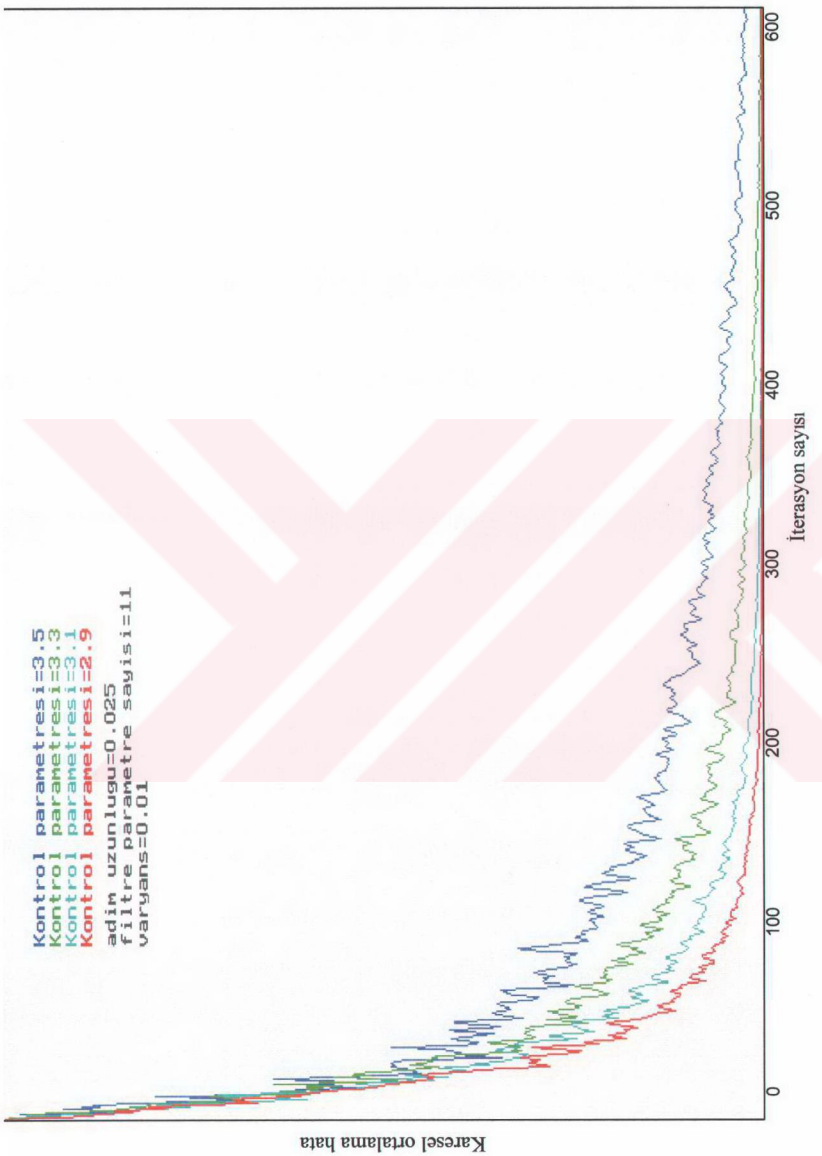
Şekil 5.50


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.0075
filtre parametre sayisi=11
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

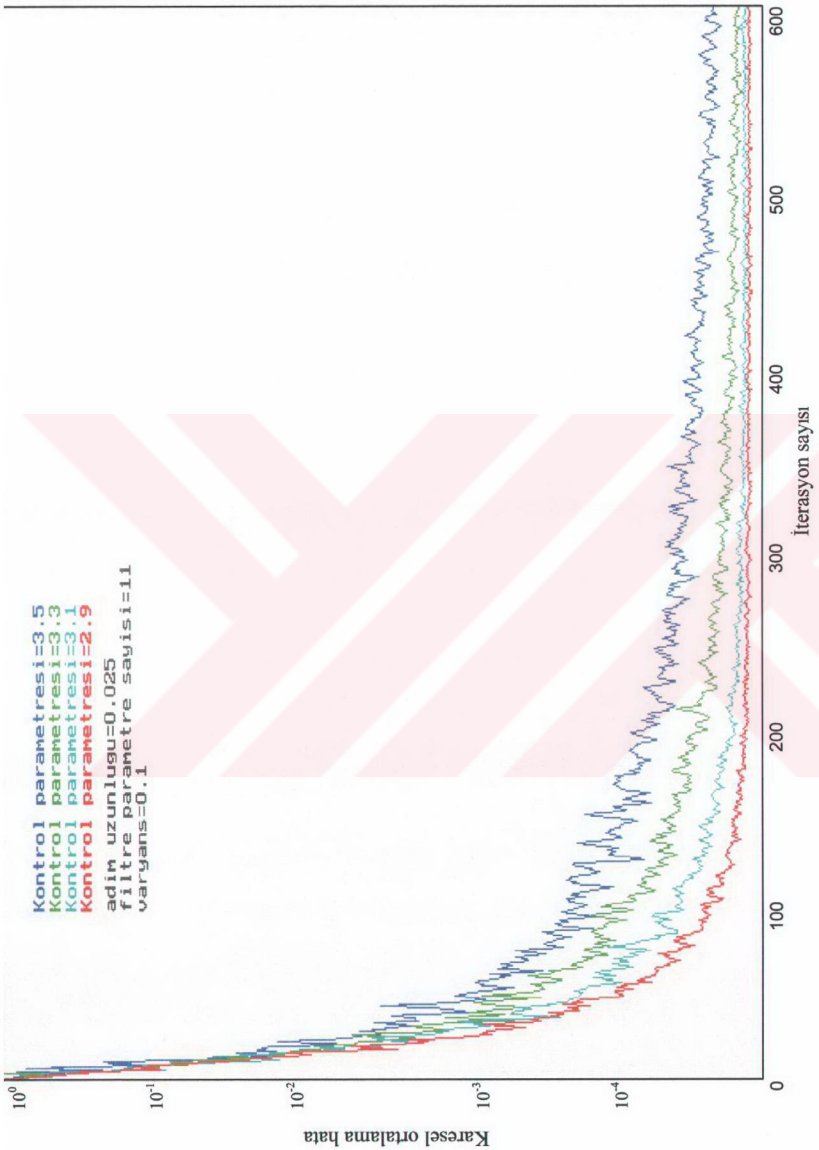
Şekil 5.51


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.025
filtre parametre sayisi=11
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

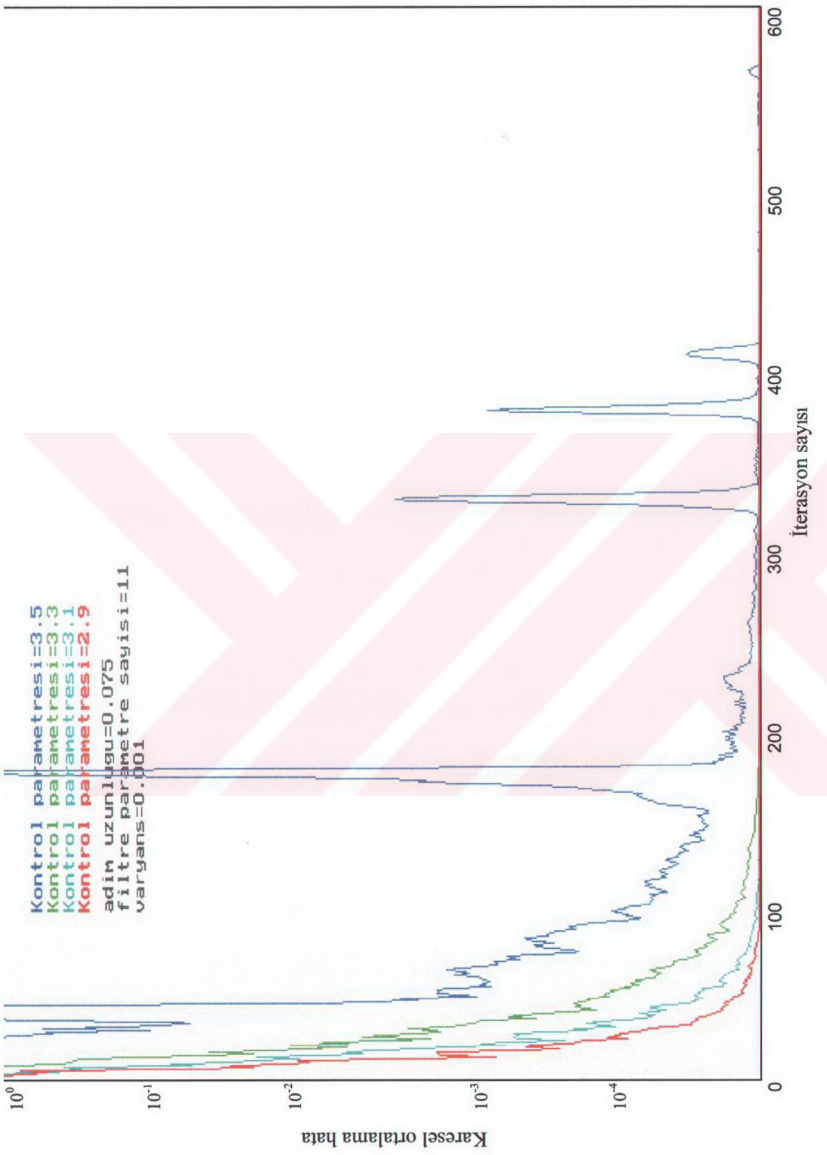
Şekil 5.52


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.025
filtre parametre sayisi=11
varyans=0.01
Karesel ortalama hata
İterasyon sayısı
0
100
200
300
400
500
600

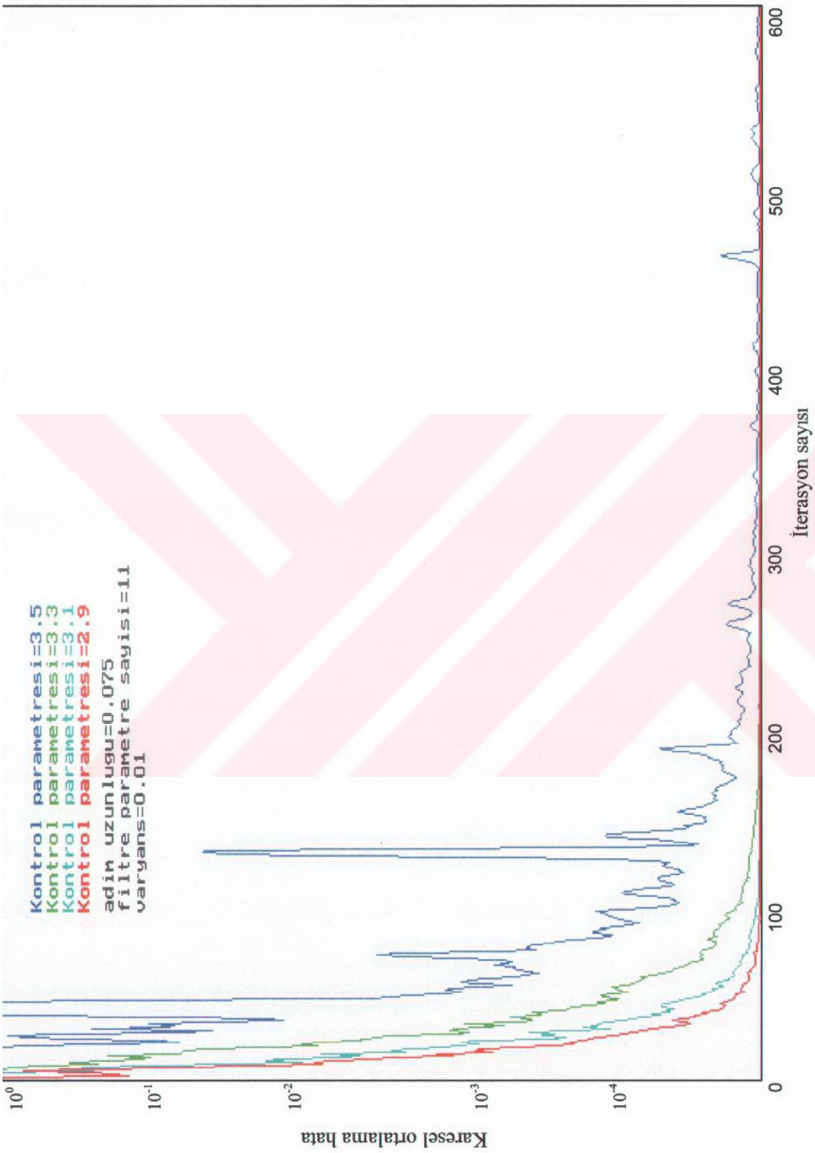
Şekil 5.53


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.025
filtre parametre sayisi=11
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

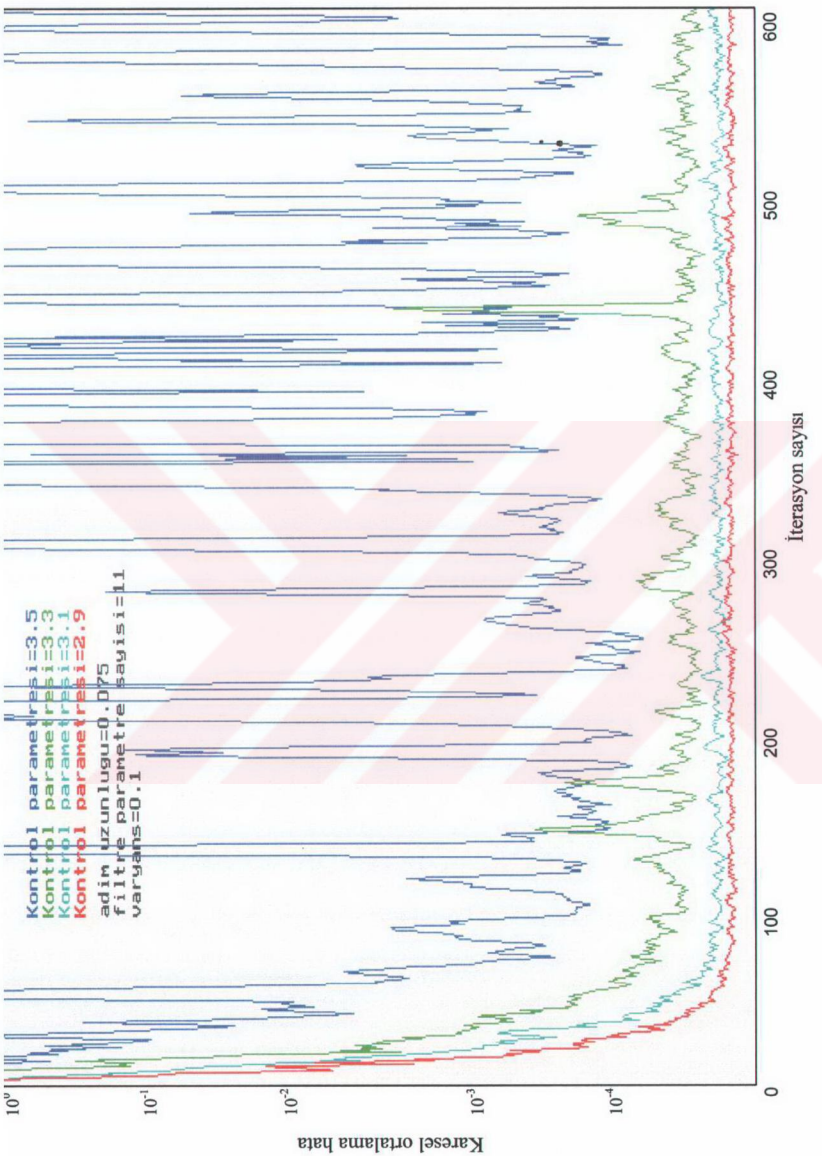
Şekil 5.54


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.075
filtre parametre sayisi=11
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

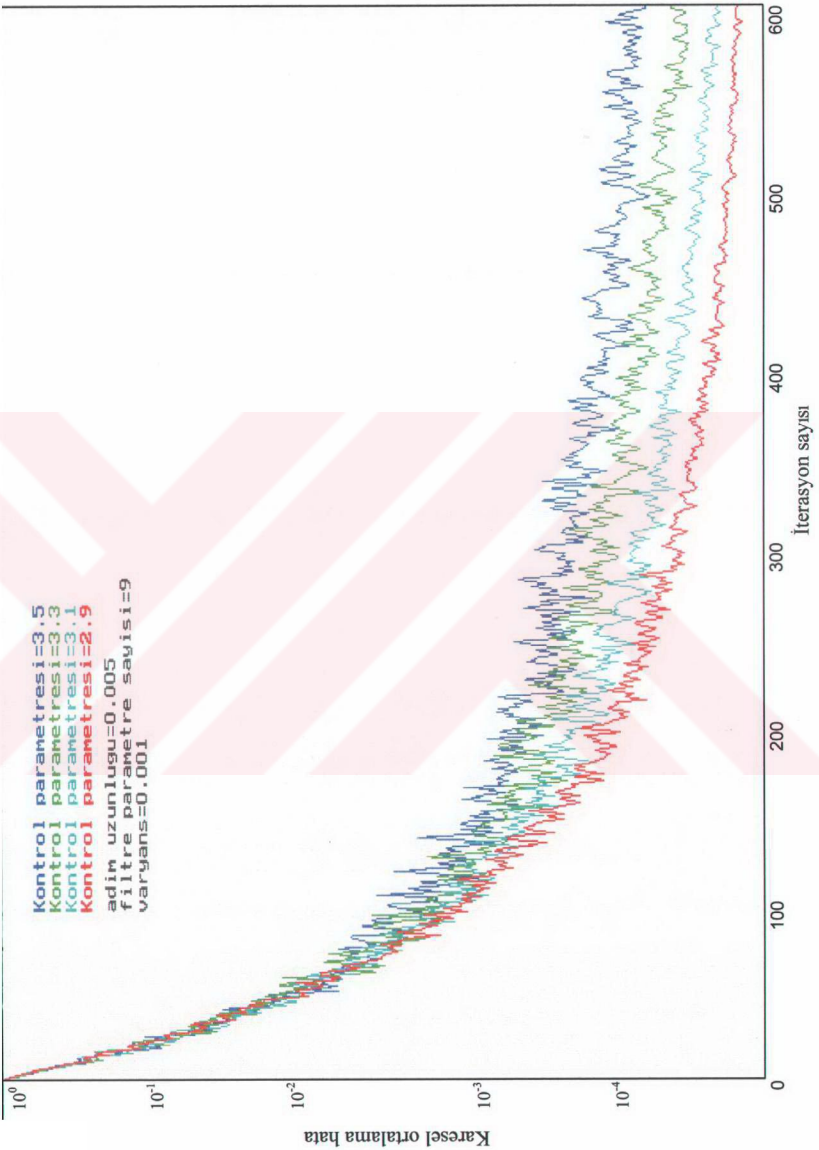
Şekil 5.55


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.075
filtre parametre sayisi=11
varyans=0.01
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

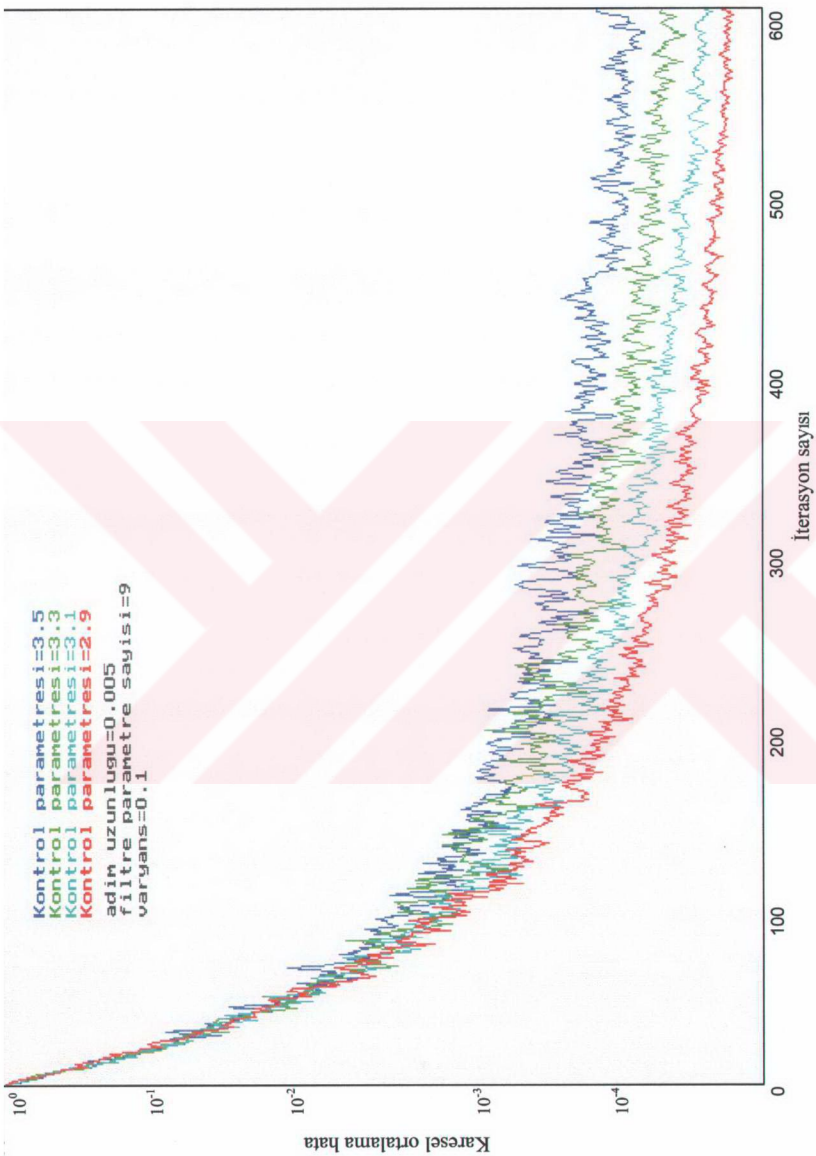
Şekil 5.56


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.075
filtre parametre sayisi=11
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

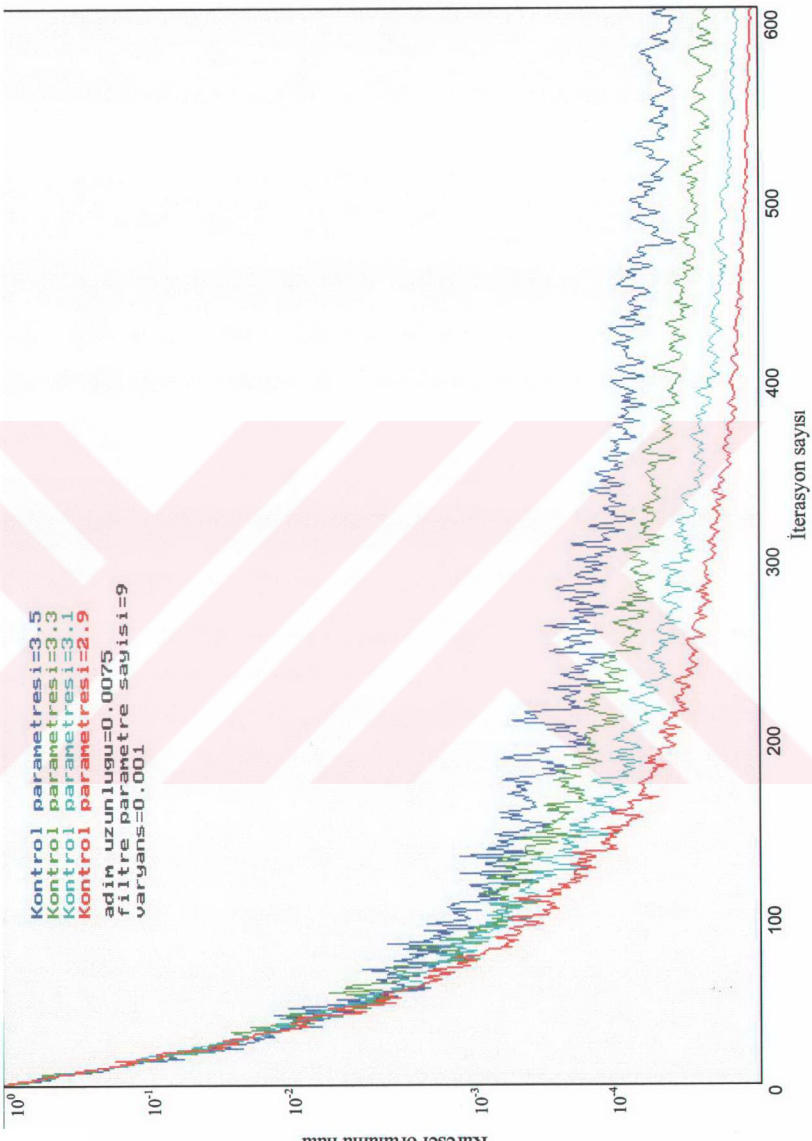
Şekil 5.57


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.005
filtre parametre sayisi=9
varyans=0.001
Karesel ortalama hata
İterasyon sayisi
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

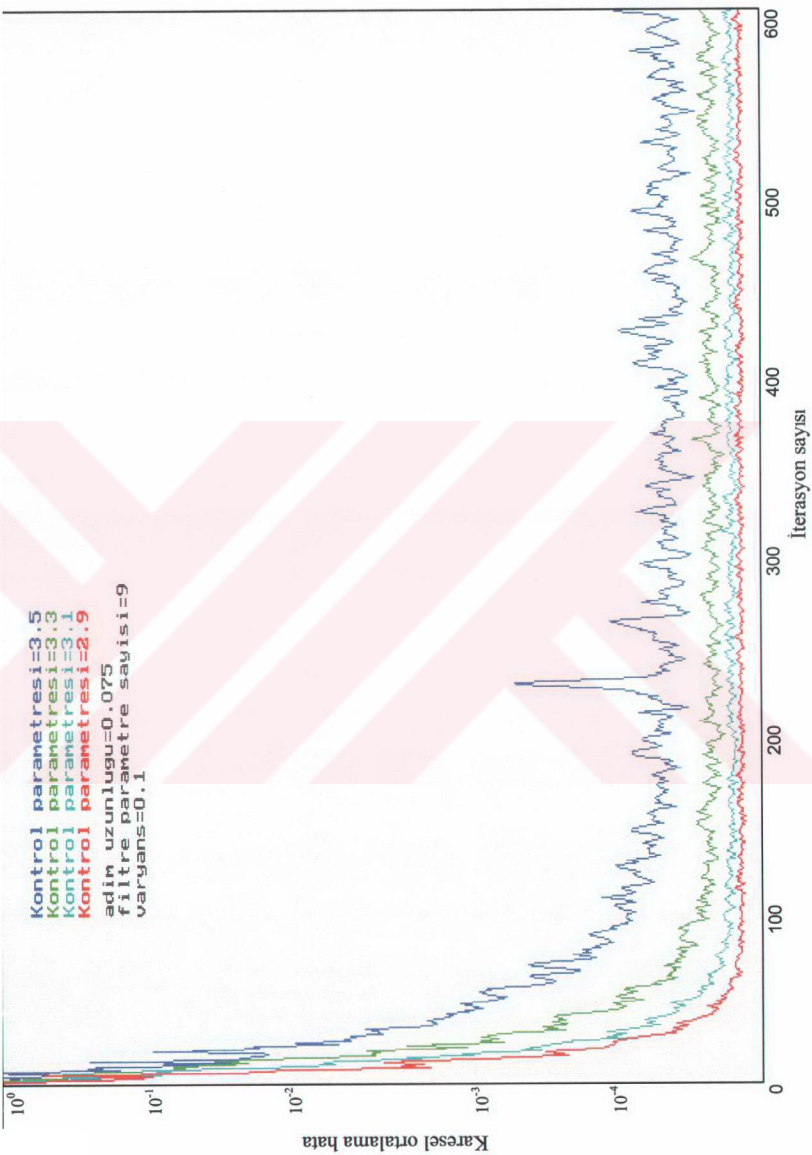
Şekil 5.58


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.005
filtre parametre sayisi=9
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

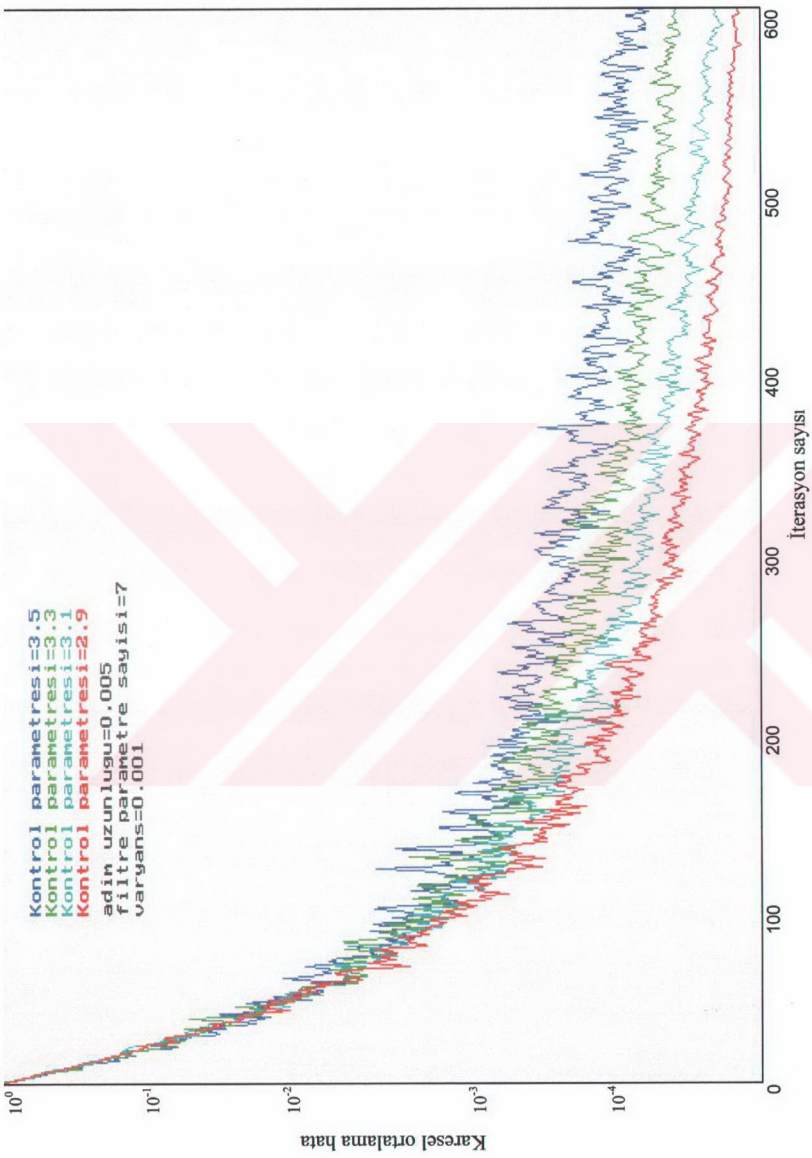
Şekil 5.59


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.0075
filtre parametre sayisi=9
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

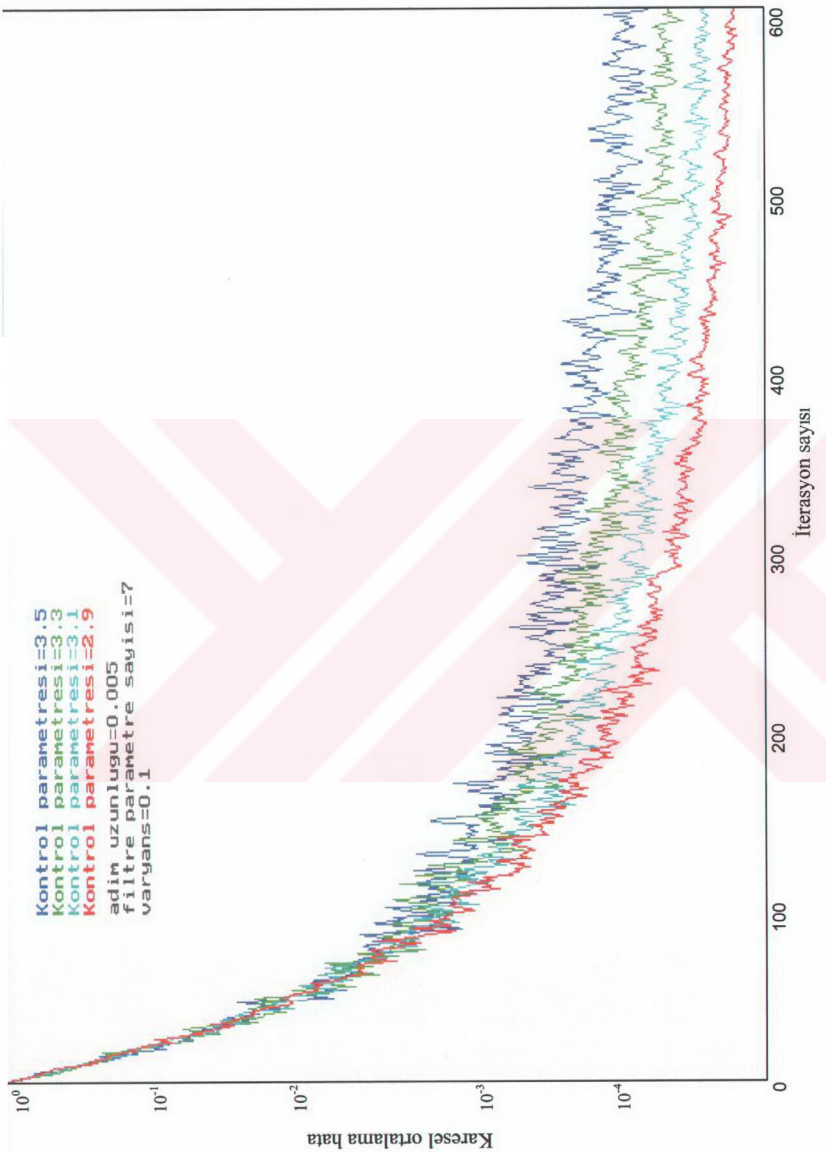
Şekil5.60


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.075
filtre parametre sayisi=9
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

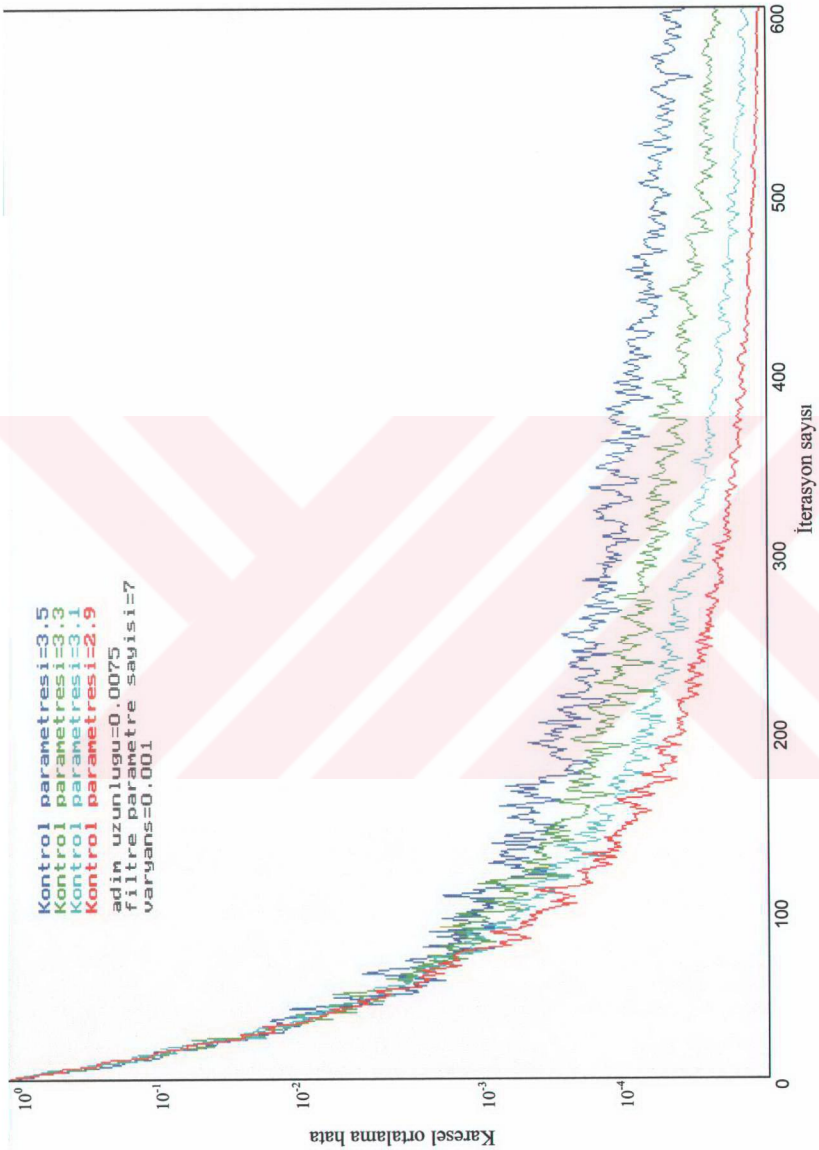
Şekil 5.61


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.005
filtre parametre sayisi=7
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

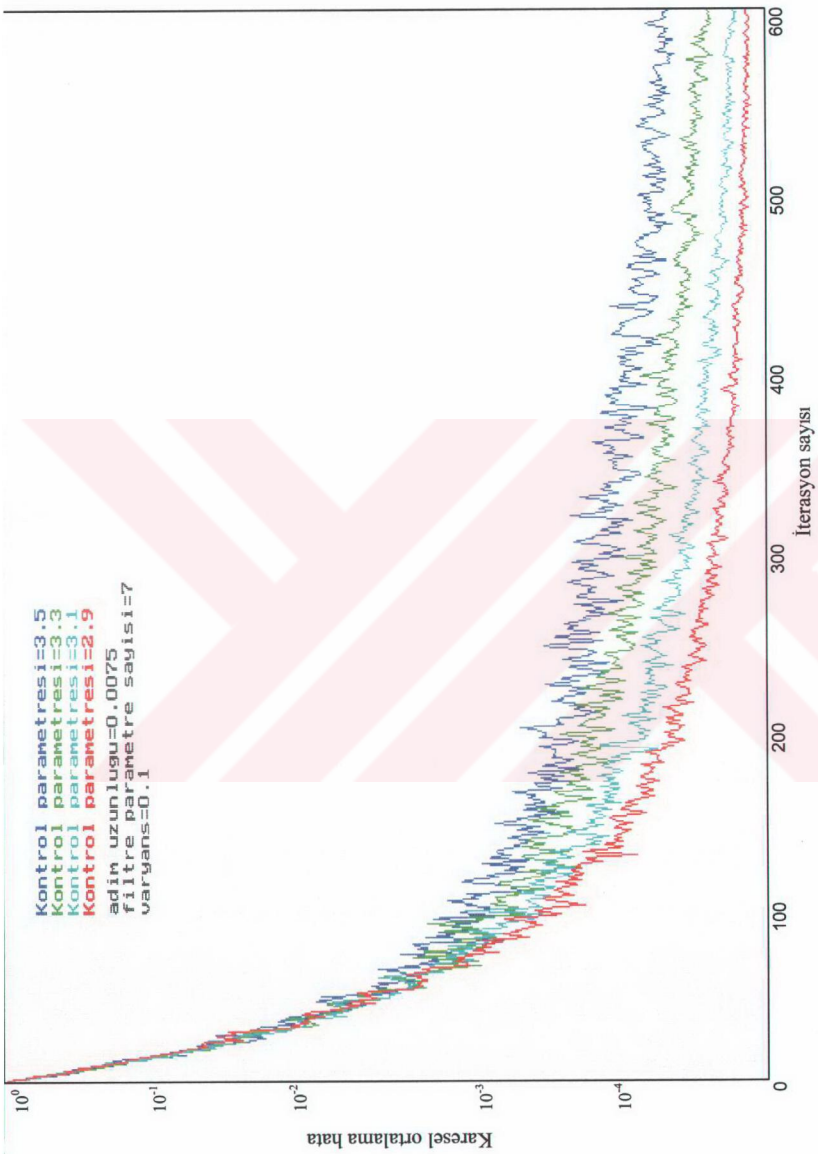
Şekil 5.62


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.005
filtre parametre sayisi=7
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

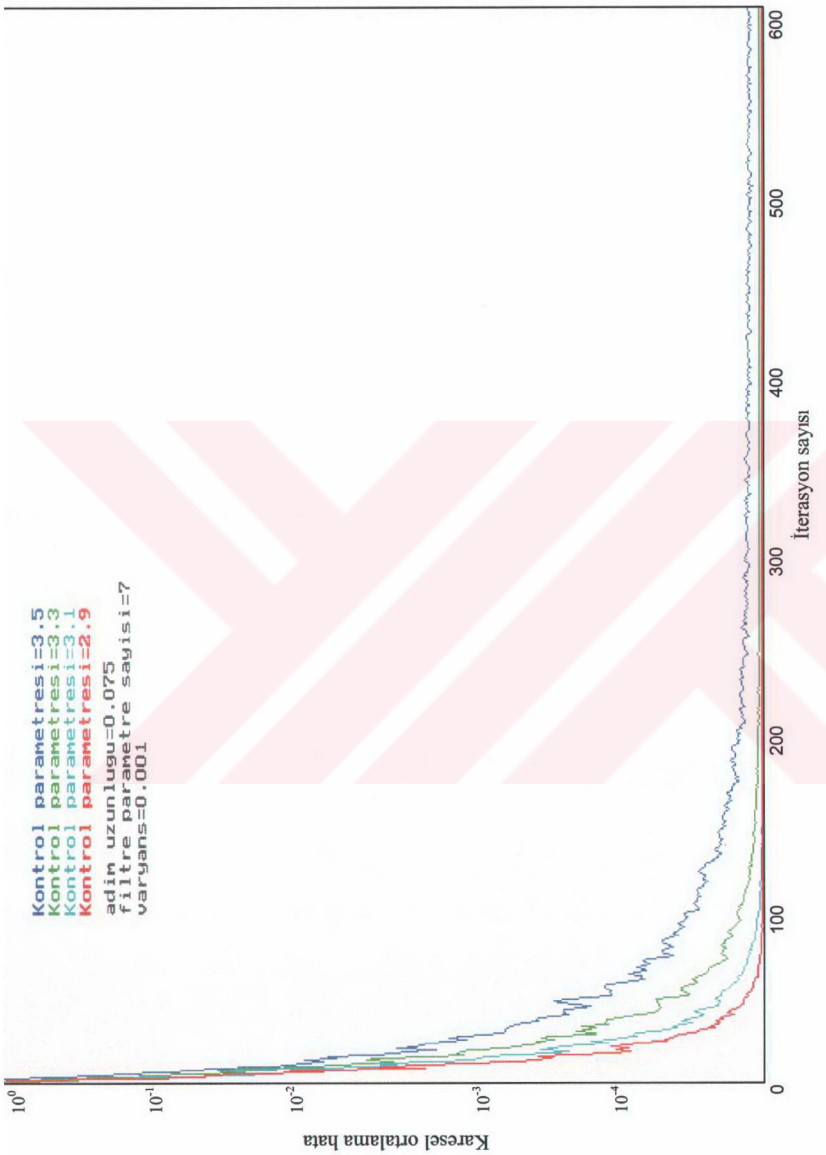
Şekil 5.63



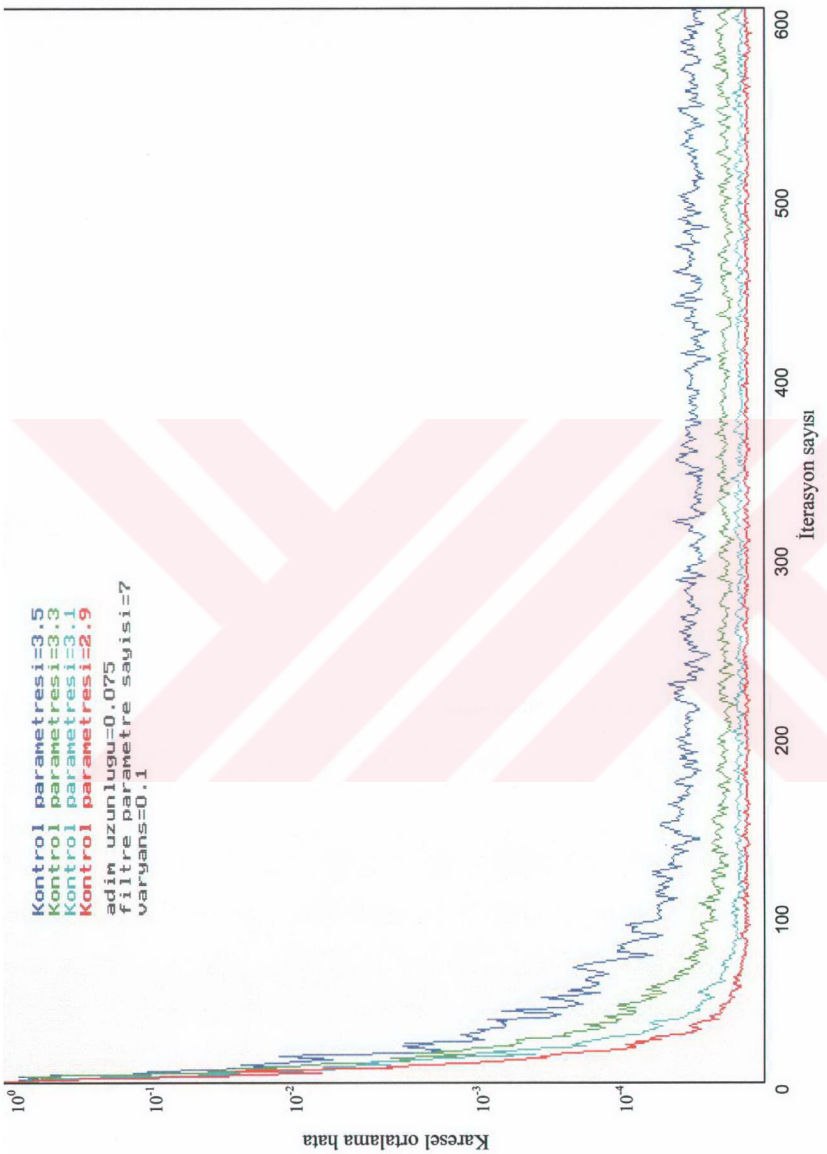
Şekil 5.64


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.0075
filtre parametre sayisi=7
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

Şekil 5.65


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.075
filtre parametre sayisi=7
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

Şekil 5.66


Kontrol parametresi=3.5
Kontrol parametresi=3.3
Kontrol parametresi=3.1
Kontrol parametresi=2.9
adim uzunlugu=0.075
filtre parametre sayisi=7
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

Şekil 5.67

Daha önce birinci grup için belirttiğimiz düşünceler bu şekillerde de doğrulanmaktadır (Şekil 5.48-5.67).

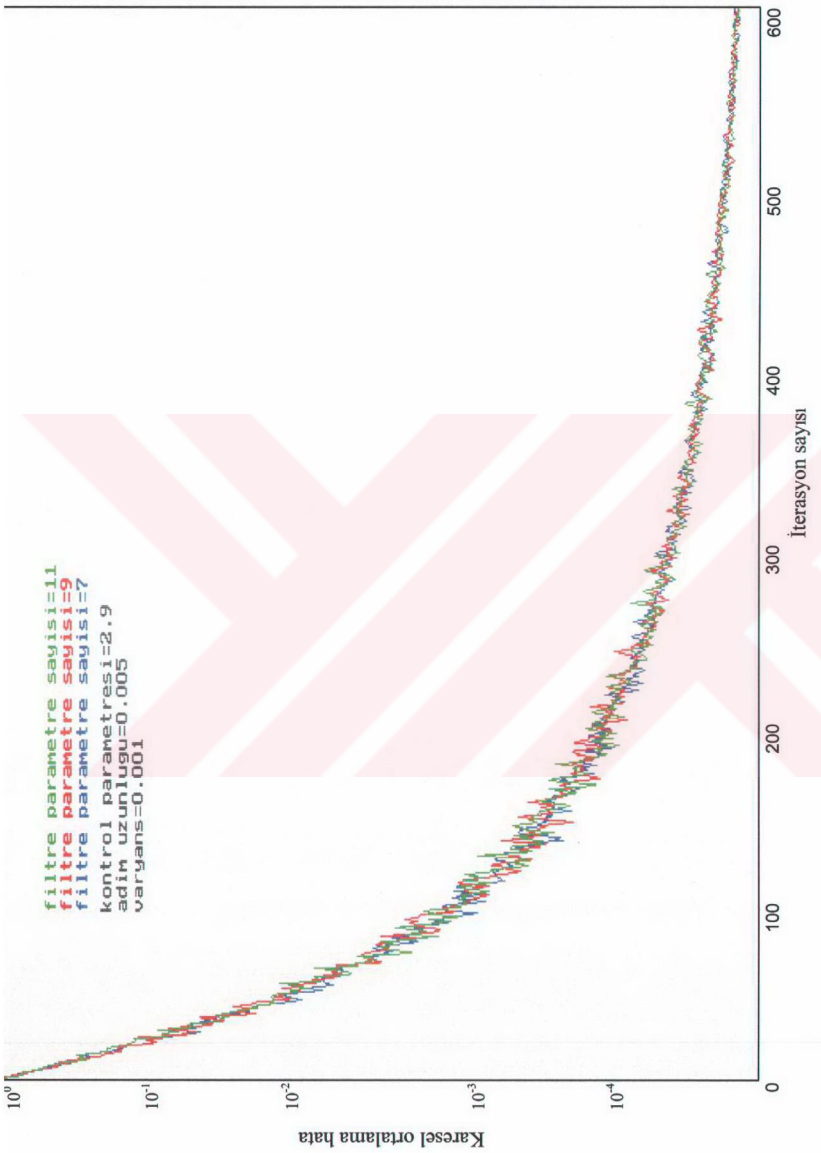
İster adım uzunluğu ve gürültü varyansı (hatta filtre parametre sayısı) yüksek, İsterse düşük olsunlar, (bunların aynı değerlerinde) kanal kontrol parametre değeri en yüksek olan durumun karesel ortalama hata durağan durum değeri daima yüksektir. Başka bir deyişle aynı adım uzunluğu ve gürültü varyansında değeri büyük olan kanal kontrol parametresi eğrisinin karesel ortalama hata durağan durum değeri en yüksektir (Şekil 5.48-5.67).Buna ek olarak, kanal kontrol prametresi yüksek ise, karesel ortalama hatadaki dalgalanmalar daha fazladır.

Bu eğrilerde kanal kontrol parametresi-filtre parametre sayısı ilişkisi de görülmektedir.

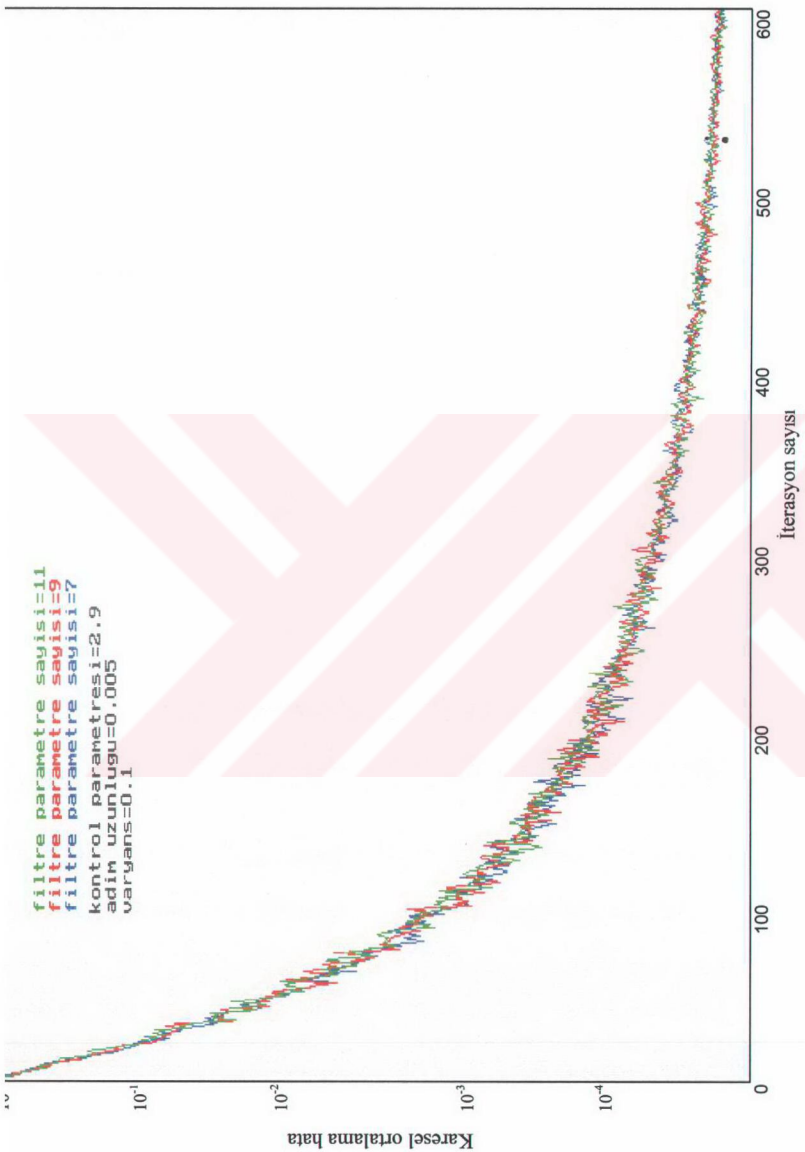
Filtre parametre sayısı düşük ise (M=7, M=9), adım uzunluğu ve gürültü varyansının görece yüksek olması eğrilerde dalgalanmaları arttırmakta ancak, kontrol edilebilir bir düzeyde olduğu görülmektedir(Şekil 5.61, 5.67).

Filtre parametre sayısı yüksek tutulursa (M=11), adım uzunluğu ve varyansın yüksek olması kanal kontrol parametresinin büyük değerleri ile elde edilen eğrilerde kontrol edilemez dalgalanmalar oluşmaktadır. Aynı zamanda karesel hata durağan durum değeri de artmaktadır.

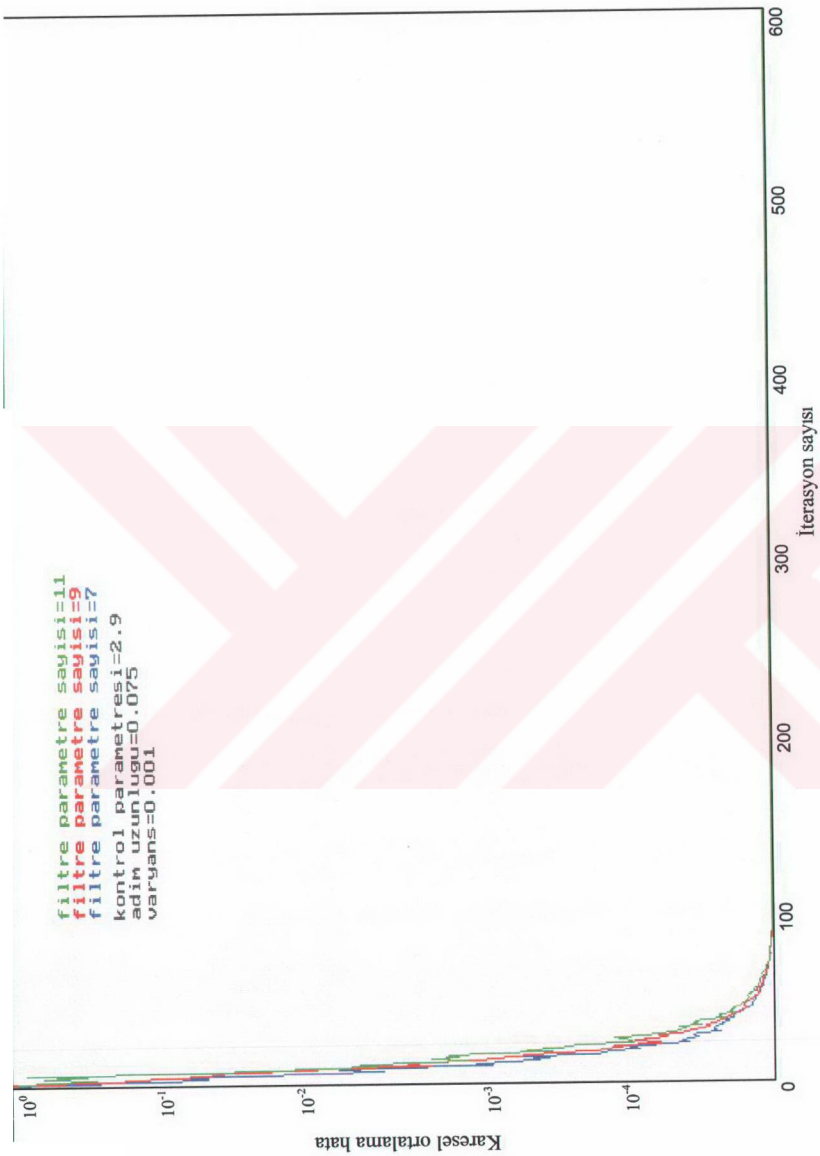
Şekil 5.48-5.49'da (adım uzunluğu (μ=0.005) ve filtre parametre sayısıı (M=10) sabit tutularak) gürültü varyansının iki değeri için(σ^2 =0.001, 0.1) karesel ortalama hata değişimi görülmektedir. Gürültü varyansı arttıkça, dalgalanmalar ve karesel ortalama hata durağan durum değeri artmaktadır.Şekil 5.50-5.54'te adım uzunluğu (μ=0.0075,0.025) yapılarak aynı sonuçlar elde edilmektedir.Şekil 5.55-5.58'de, adım uzunluğu (μ=0.075) sabit tutulup, gürültü varyansı değiştirilmiştir (σ^2 =0.001, 0.01, 0.1). Yukarda da belirtildiği gibi, gürültü varyansı arttıkça (sinyal-gürültü oranı SNR) kanal kontrol parametresinin yüksek değerli olması durumunda karesel ortalama hatada oluşan dalgalanmaları gidermek olanak dışı olmaktadır. Denilebilir ki dengeleyici öğrenme durumundan çıkmamaktadır.


filtre parametre sayisi=11
filtre parametre sayisi=9
filtre parametre sayisi=7
kontrol parametresi=2.9
adim uzunlugu=0.005
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

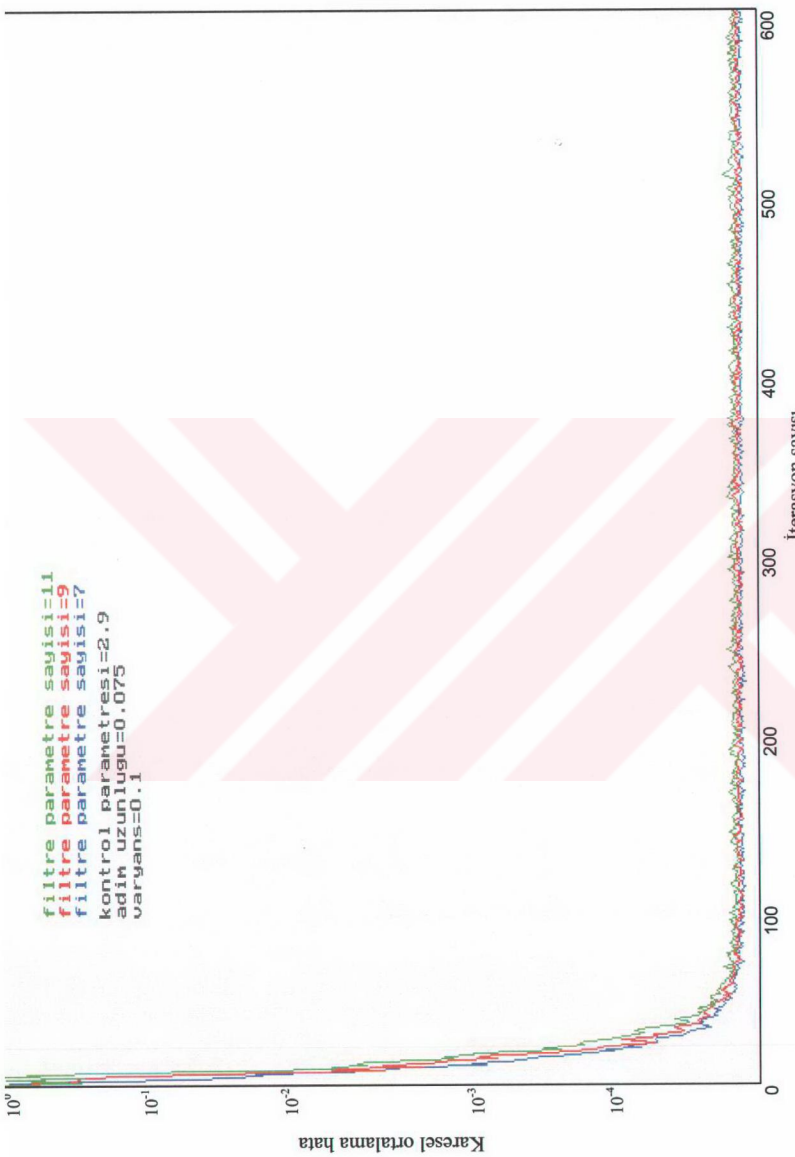
Şekil 5.68


filtre parametre sayisi=11
filtre parametre sayisi=9
filtre parametre sayisi=7
kontrol parametresi=2.9
adim uzunlugu=0.005
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
0
100
200
300
400
500
600
10^{-1}
10^{-2}
10^{-3}
10^{-4}

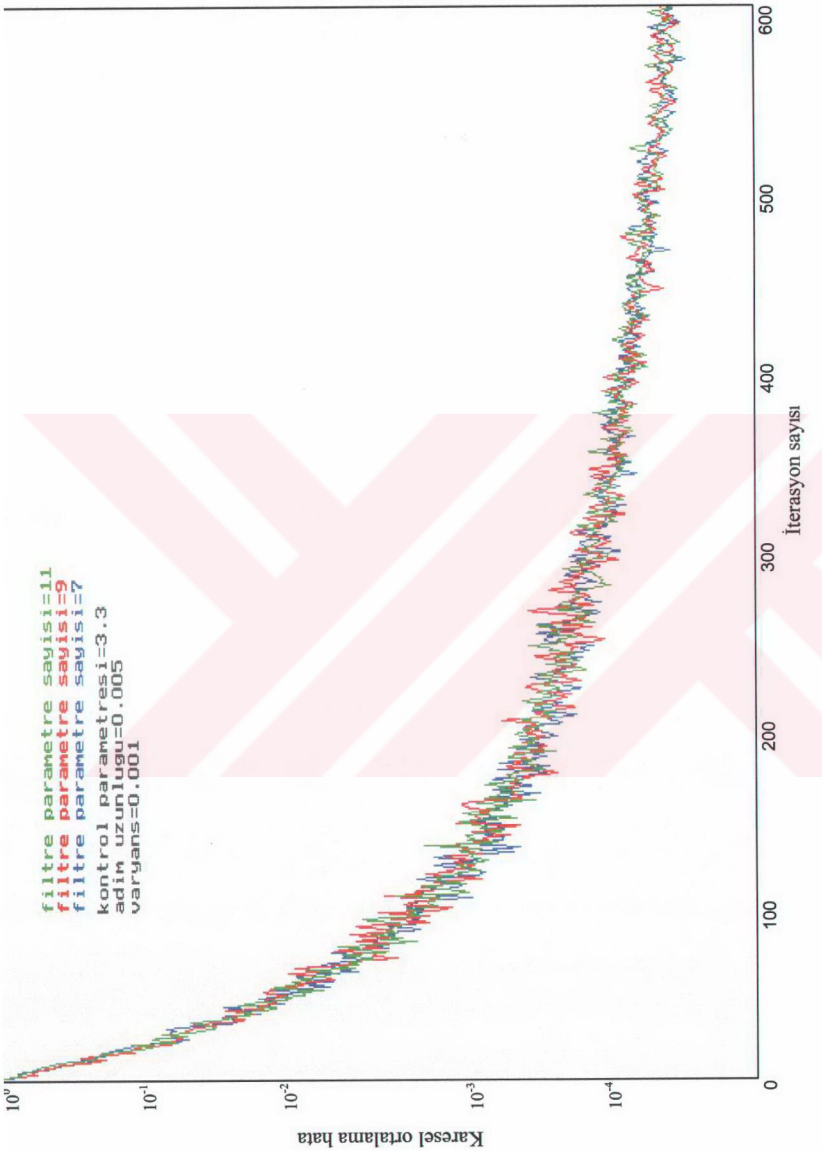
Şekil 5.69


filtre parametre sayisi=11
filtre parametre sayisi=9
filtre parametre sayisi=7
kontrol parametresi=2.9
adim uzunlugu=0.075
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

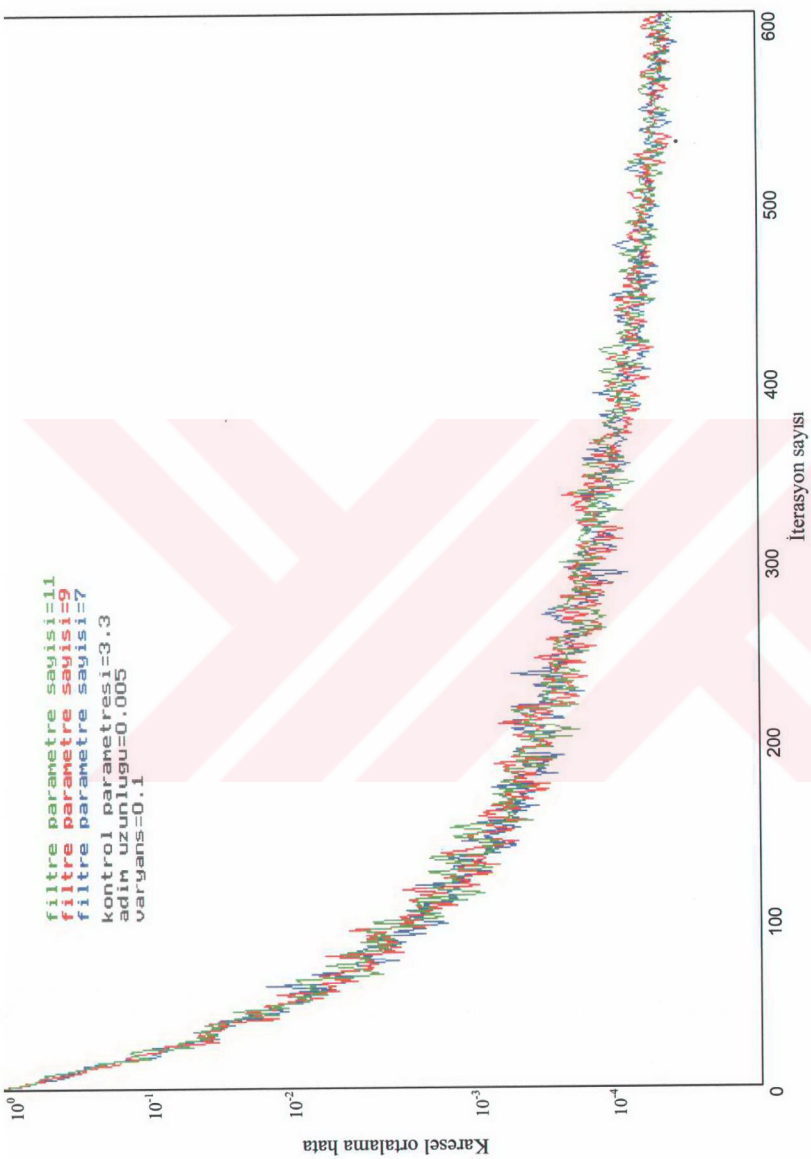
Şekil 5.70


filtre parametre sayisi=11
filtre parametre sayisi=9
filtre parametre sayisi=7
kontrol parametresi=2.9
adim uzunlugu=0.075
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

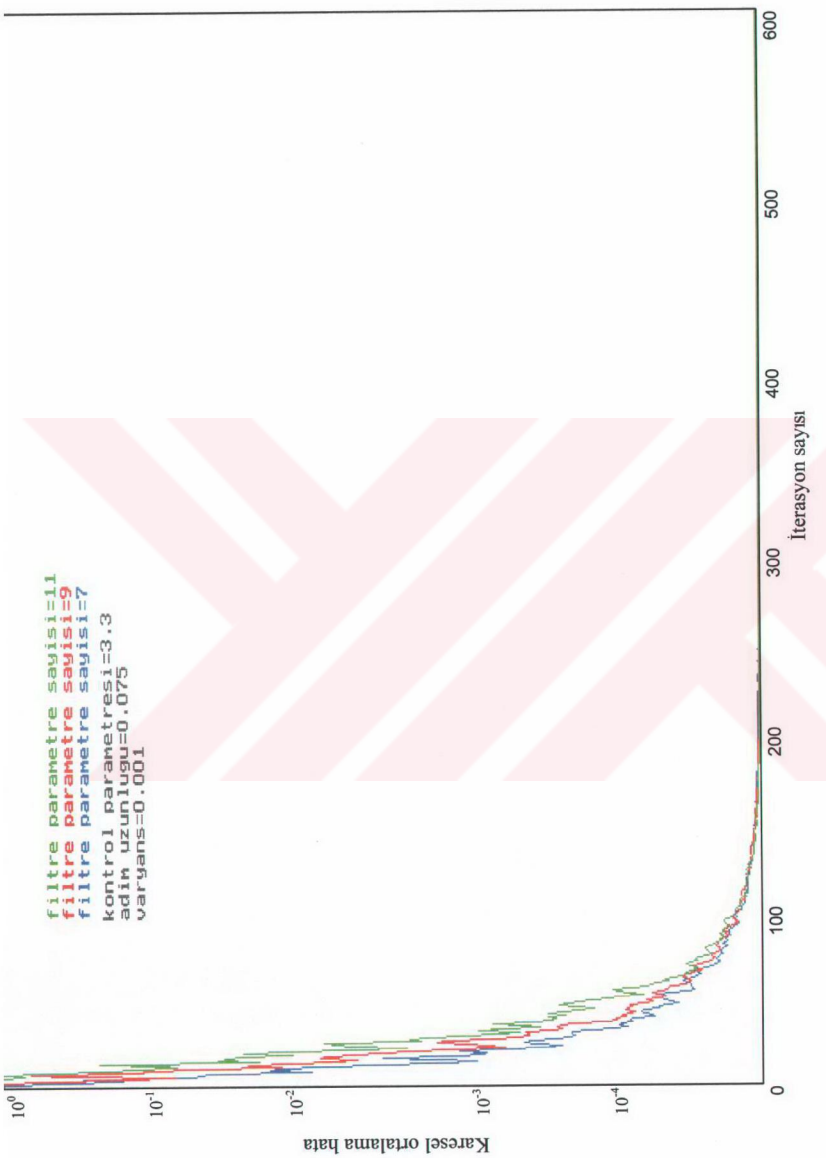
Şekil 5.71


filtre parametre sayisi=11
filtre parametre sayisi=9
filtre parametre sayisi=7
kontrol parametresi=3.3
adim uzunlugu=0.005
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

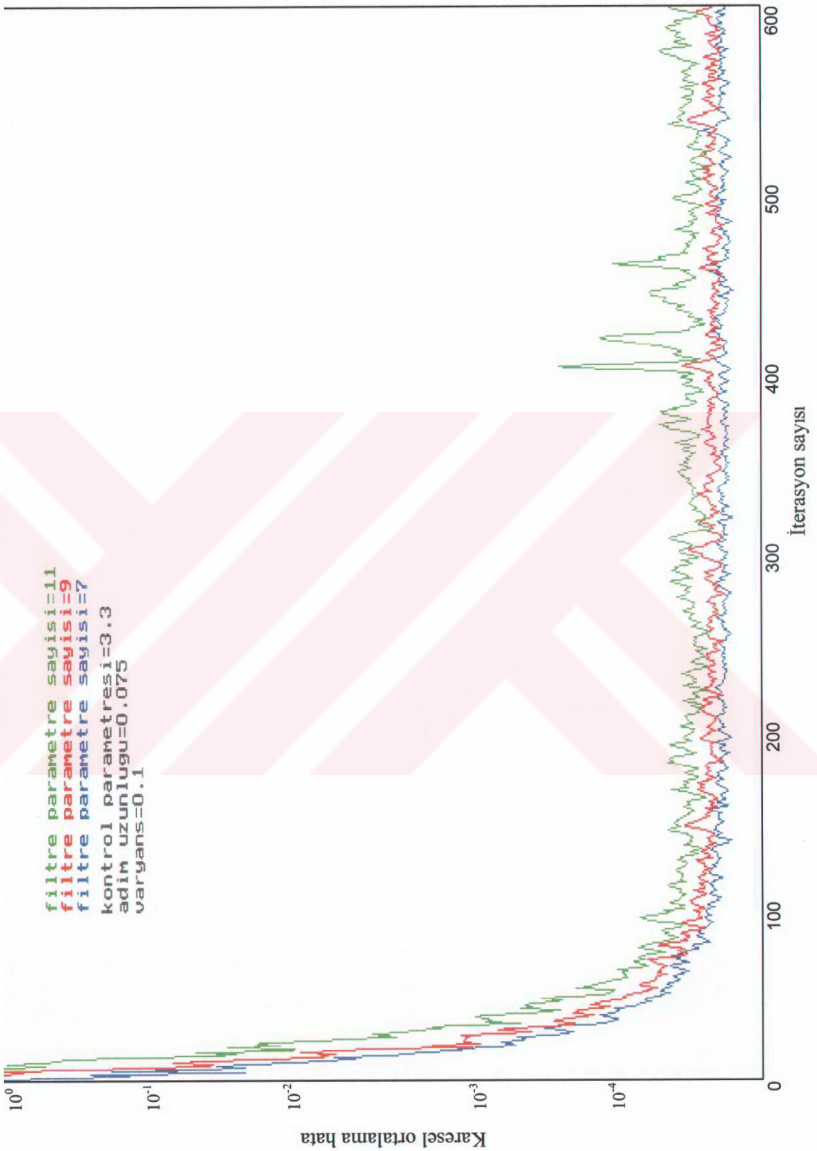
Şekil 5.72


filtre parametre sayisi=11
filtre parametre sayisi=9
filtre parametre sayisi=7
kontrol parametresi=3.3
adim uzunlugu=0.005
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

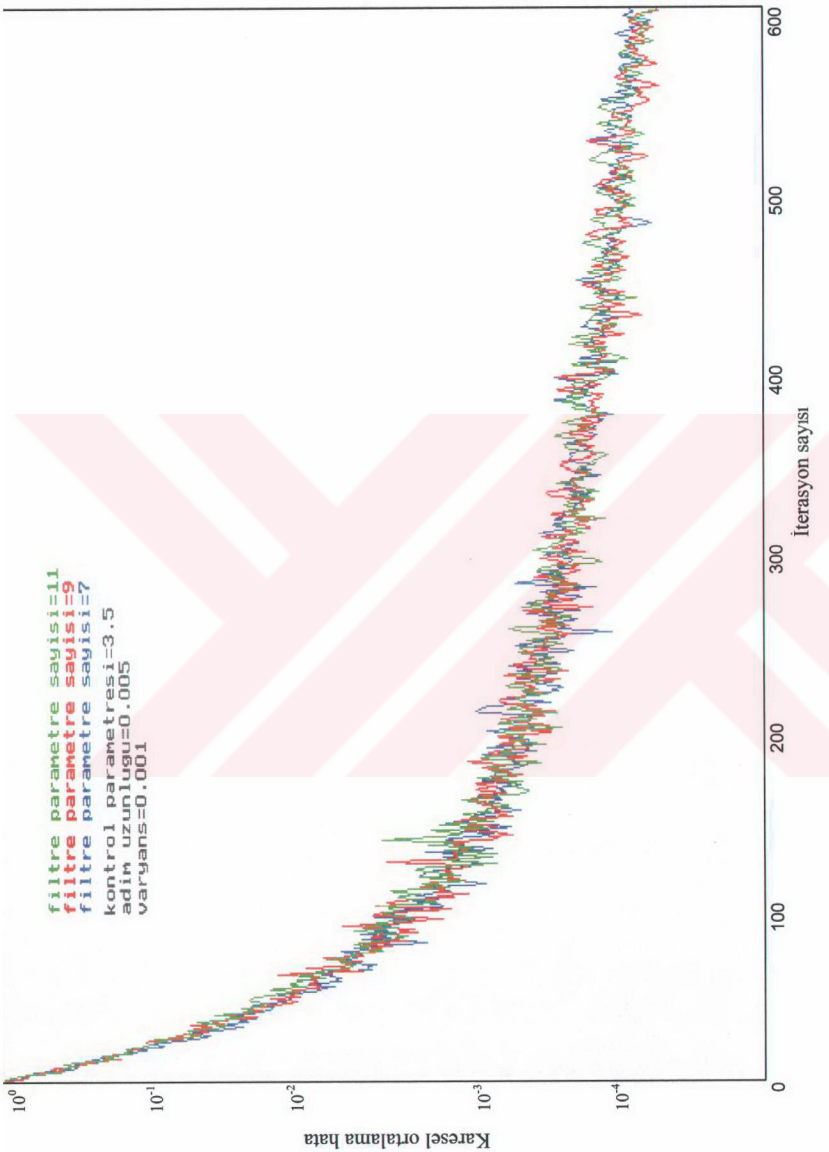
Şekil 5.73


filtre parametre sayisi=11
filtre parametre sayisi=9
filtre parametre sayisi=7
kontrol parametresi=3.3
adim uzunlugu=0.075
varyans=0.001
Karesel ortalama hata
İterasyon sayisi
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

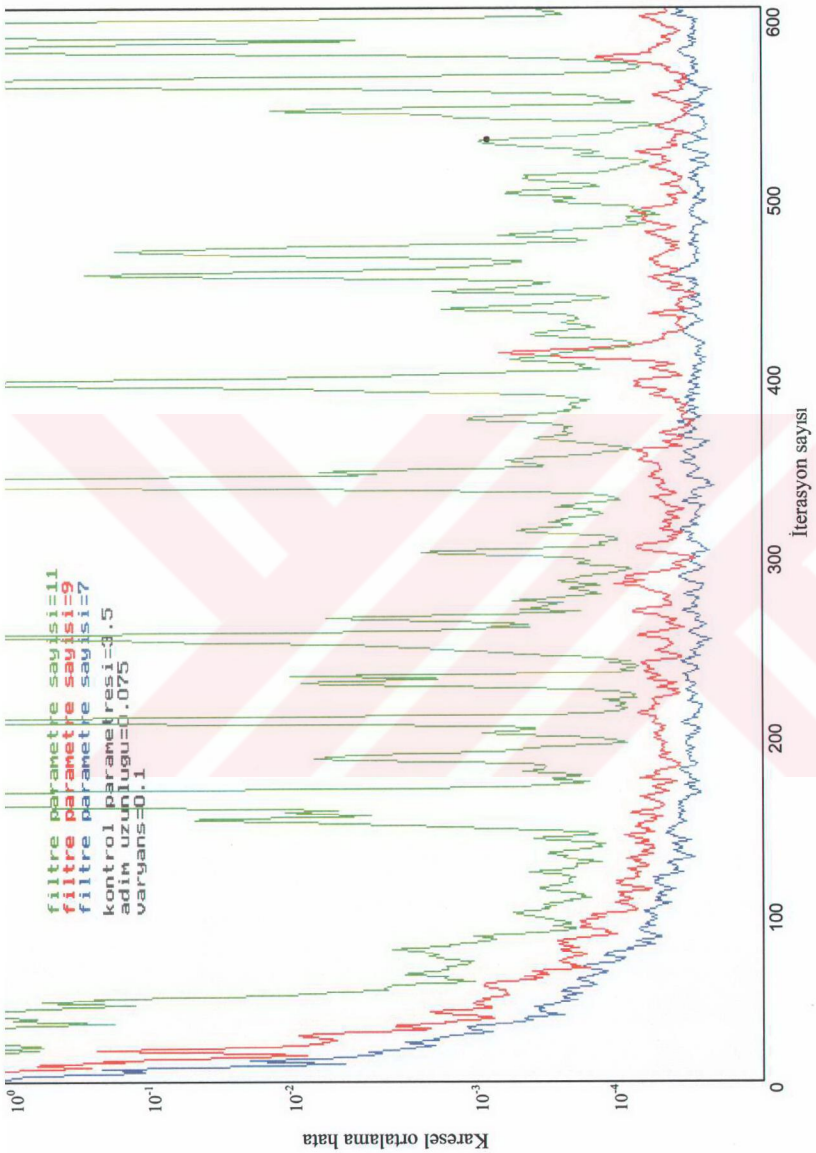
Şekil 5.74


filtre parametre sayisi=11
filtre parametre sayisi=9
filtre parametre sayisi=7
kontrol parametresi=3.3
adim uzunlugu=0.075
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^-1
10^-2
10^-3
10^-4
0
100
200
300
400
500
600

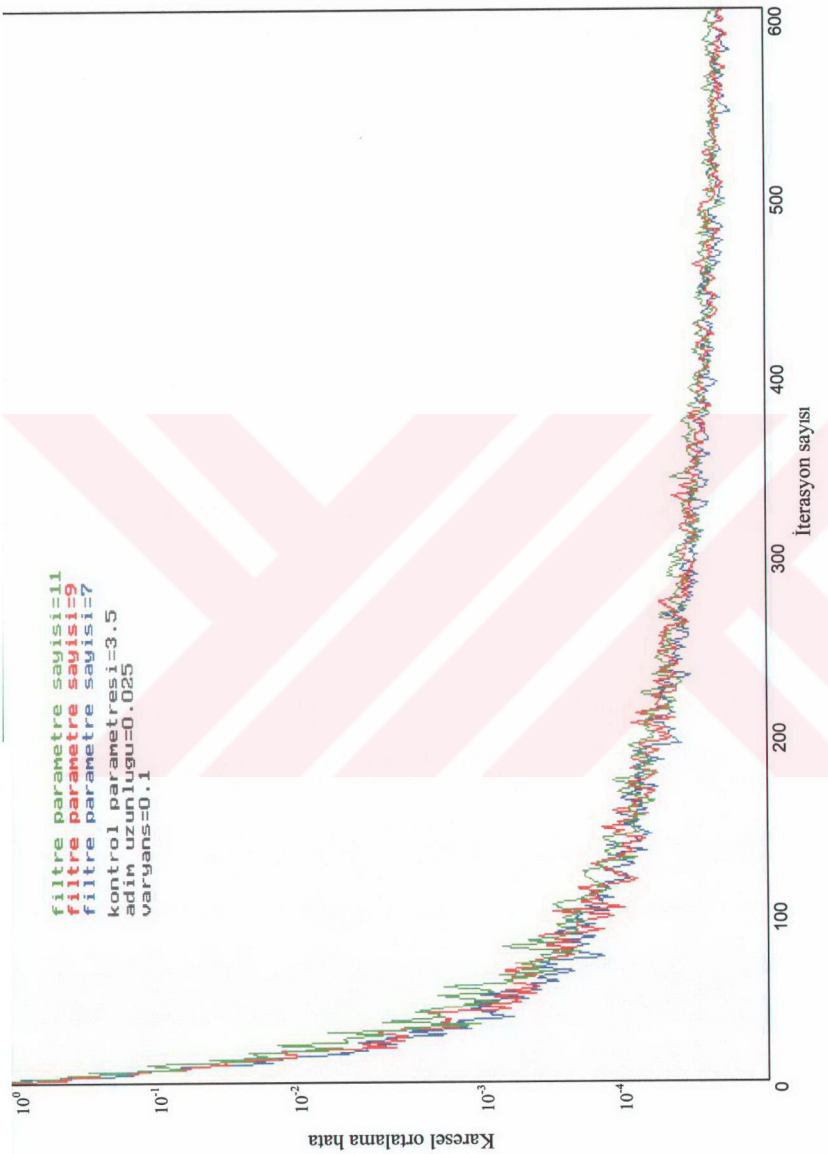
Şekil 5.75


filtre parametre sayisi=11
filtre parametre sayisi=9
filtre parametre sayisi=7
kontrol parametresi=3.5
adim uzunlugu=0.005
varyans=0.001
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

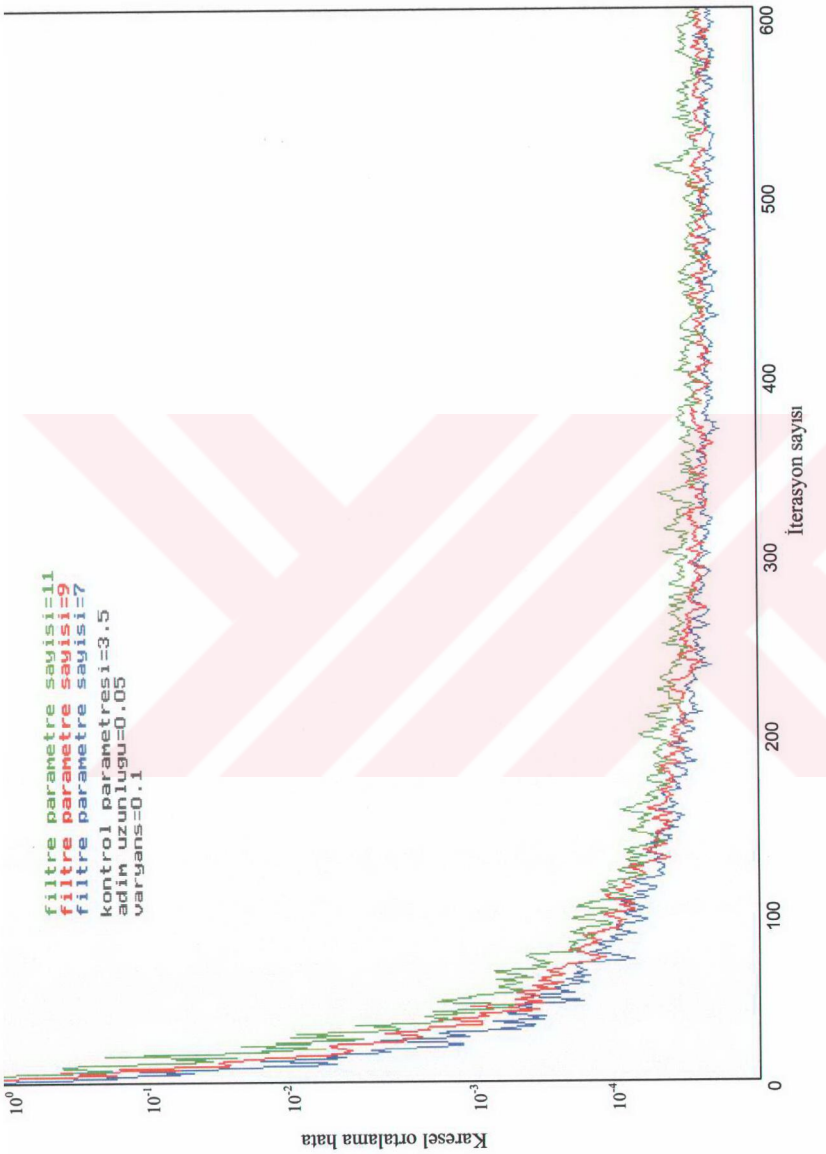
Şekil 5.76


filtre parametre sayisi=11
filtre parametre sayisi=9
filtre parametre sayisi=7
kontrol parametresi=3.5
adim uzunlugu=0.075
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

Şekil 5.77


filtre parametre sayisi=11
filtre parametre sayisi=9
filtre parametre sayisi=7
kontrol parametresi=3.5
adim uzunlugu=0.025
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

Şekil 5.78


filtre parametre sayisi=11
filtre parametre sayisi=9
filtre parametre sayisi=7
kontrol parametresi=3.5
adim uzunlugu=0.05
varyans=0.1
Karesel ortalama hata
İterasyon sayısı
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600

Şekil 5.79

Üçüncü grup şekil incelendiğinde iki önemli sonuç görmek mümkündür:

1. Kanal kontrol parametresi ve varyans düşük ise, filtre parametre sayısının aldığı tüm değerler için (M+1=7, 9, 11) elde edilen sonuçlar hemen hemen eşittir. Filtre parametre sayısı ne olursa olsun, gerek yakınsama hızları, gerekse karesel ortalama hata durağan durum değerleri yaklaşık olarak eşittir. Ayrıca filtre parametre sayısının aldığı üç değer için de karesel ortalama hatada oluşan dalgalanmalar az olduğu gibi, hemen hemen aynı değerlerdedir. Adım uzunluğu büyükse, yakınsama hızları artmakla birlikte, filtre parametre sayısı açısında büyük bir fark oluşmamaktadır. Aynı şey karesel hata durağan durum değeri için de geçerlidir. Gürültü varyansı sabit tutulup, kanal kontrol parametresi daha da arttırılırsa karesel ortalama hatada dalgalanmalar artmaktadır (Şekil 5.68-5.73). Burada özellikle filtre parametre sayısı fazla olan eğride dalgalanmalar ve karesel hata durağan durum değeri artmaktadır (Şekil 5.75).

2. Kanal kontrol parametresi ve gürültü varyansı yüksek ise filtre parametre sayısı fazla olan eğride dalgalar çok artmaktadır. Filtre parametre sayısı arttıkça dalgalanmalar kontrol edilemez noktaya gelmektedir. Filtre parametre sayısının çok artması ya da çok azalması karesel ortalama hatanın yükselmesine neden olmaktadır.

Genel olarak LMS algoritması açısında, filtre parametre sayısının yüksek olması durumunda; hesaplama süresini arttırır, yani yakınsama hızı düşer ve gürültülü bir ortamda karesel ortalama hatada büyük dalgalanmalar olur.

Yukarıdaki sonuçlardan hareketle parametre sayısı düşük olan dengeleyiciler özellikle kanal kontrol parametresi ve gürültü varyansı yüksek olan durumlarda tercih edilir.

6 İŞARETLER ARASINDAKİ İLİŞKİLERİN İNCELENMESİ

Sistemin değişik noktalarındaki işaretlere ilişkin özilişki ve çapraz ilişki önemlidir. Beklenenin dışında bir ilişkinin bulunması, sistemin sağlıklı bir biçimde çalışmadığını göstermektedir. Daha önceki bölümlerde ele alınan dengeleme probleminin çözümünde ulaşılan sonuçlar, sistemin sağlıklı bir biçimde çalıştığını göstermekle birlikte yine de bu bölümde sistemin çeşitli noktalarındaki işaretler arasındaki özilişki ve çapraz ilişkinin incelenerek, daha önceki bölümlerde elde edilen sonuçlarla uyumlu bir ilişkinin bulunduğu gösterilmektedir.

Beşinci bölümdeki kanal dengelemeye ilişkin blok şema (Şekil 5.5) göz önüne alınarak farklı noktalardaki işaretler arasındaki ilişkileri inceleyelim.

Şekil 5.5'te görüldüğü gibi kanalın girişi a(n), kanalın gürültülü çıkışı u(n) (aynı zamanda uygulamalı dengeleyicinin girişidir), uyarlamalı dengeleyicinin çıkışı y(n) ile gösterilmiştir.
Kanal girişine ait özilişki matrisi,

$$\mathbf{R} = E[\mathbf{a}(n)\mathbf{a}(n)] = \begin{bmatrix} r(0) & r(1) & \dots & r(13) & r(14) \\ r(-1) & r(0) & \dots & r(12) & r(13) \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ r(-14) & r(-13) & \dots & r(-1) & r(0) \end{bmatrix} \tag{6.1}$$

biçiminde tanımlanmaktadır. a(n)'nin özilişki matrisinin (6.1) eşitliği kullanılarak bilgisayar ile hesaplanmış değerleri (EK-4) Tablo 6.1'de verilmiştir.

Tablo 6.1 Kanal girişinin özilişki değerleri

İterasyon sayısı

	10	**100**	**1000**	**2000**
r(0)=	1.000000	1.000000	1.000000	1.000000
r(1)=	0.026667	0.002667	-0.002133	-0.001000
r(2)=	0.040000	-0.037333	-0.000400	0.003267
r(3)=	-0.013333	0.018667	0.002677	-0.013667
r(4)=	-0.080000	0.012000	-0.000400	-0.001800
r(5)=	-0.040000	0.036000	0.009467	-0.006400
r(6)=	0.160000	-0.001333	0.003600	0.011600
r(7)=	0.040000	0.041333	0.000533	-0.003267
r(8)=	0.053333	-0.032000	0.003467	-0.000600
r(9)=	-0.133333	-0.016000	-0.001467	0.000933
r(10)=	-0.066667	-0.028000	0.000800	-0.001933
r(11)=	-0.053333	0.053333	0.006133	-0.004267
r(12)=	0.040000	0.072000	0.002533	0.001333
r(13)=	0.173333	-0.004000	-0.003733	-0.001200
r(14)=	0.133333	-0.050667	0.000400	0.013200

Tablo 6.1'den de görüleceği gibi r(0) dışındaki değerlerin sıfıra çok yakın olduğu, yani giriş işaretlerinin istatistiksel bağımsız oldukları sonucu çıkarılmaktadır.

Kanalın girişi a(n) ile gürültülü çıkışı u(n) arasındaki çapraz ilişkinin incelenmesi:

Kanalın girişiyle gürültülü çıkışı arasındaki çapraz ilişki matrisi,

$$\mathbf{P} = [\mathbf{u}(n)\mathbf{a}(n-k)] = \begin{bmatrix} \rho(0) \\ \rho(1) \\ \cdots \\ \rho(14) \end{bmatrix} \qquad (6.2)$$

biçiminde tanımlanmaktadır. (6.2) eşitliği kullanılarak farklı W değerleri için bilgisayar ile hesaplanmış **P** değerleri (EK-4) Tablo 6.2, 6.3, 6.4 ve 6.5'te verilmiştir.

Tablo 6.2 Kanal girişi ile gürültülü çıkış arasındaki çapraz ilişki değerleri, W=2.9

	İterasyon sayısı			
	10	**100**	**1000**	**2000**
ρ(0)=	0.258254	0.222689	0.211324	0.211772
ρ(1)=	1.014657	0.998241	0.997949	1.001846
ρ(2)=	0.271587	0.206689	0.214257	0.222305
ρ(3)=	0.010402	-0.027461	-0.007301	-0.007169
ρ(4)=	-0.094607	-0.029759	0.011136	0.000421
ρ(5)=	-0.032813	0.004057	0.011657	0.005053
ρ(6)=	0.152813	0.002289	0.004218	0.000994
ρ(7)=	0.036502	0.016908	0.005260	0.006330
ρ(8)=	0.138913	0.015839	-0.012506	0.006088
ρ(9)=	0.162931	-0.020851	-0.007748	0.001132
ρ(10)=	0.006620	-0.025069	-0.000660	0.005611
ρ(11)=	0.142128	0.013872	0.000128	-0.006113
ρ(12)=	0.104776	0.018024	0.007258	0.004578
ρ(13)=	0.175698	-0.003149	0.006792	-0.003013
ρ(14)=	0.140510	0.003953	0.009060	0.007612

Tablo 6.3 Kanal girişi ile gürültülü çıkış arasındaki çapraz ilişki değerleri, W=3.1

İterasyon sayısı

	10	100	1000	2000
ρ(0)=	0.322238	0.278901	0.271221	0.287997
ρ(1)=	1.050448	1.007847	0.995796	1.003477
ρ(2)=	0.352238	0.302901	0.267021	0.287397
ρ(3)=	0.011974	0.002574	-0.013168	0.0
ρ(4)=	-0.054395	0.009542	0.006117	0.0
ρ(5)=	0.116816	-0.039453	-0.000154	0.0
ρ(6)=	0.264079	-0.053216	0.005480	0.0
ρ(7)=	0.148026	0.024879	0.011201	0.0
ρ(8)=	-0.011974	0.051542	0.000474	0.0
ρ(9)=	-0.126816	0.061211	0.002986	0.0

Tablo 6.4 Kanal girişi ile gürültülü çıkış arasındaki çapraz ilişki değerleri, W=3.3

İterasyon sayısı

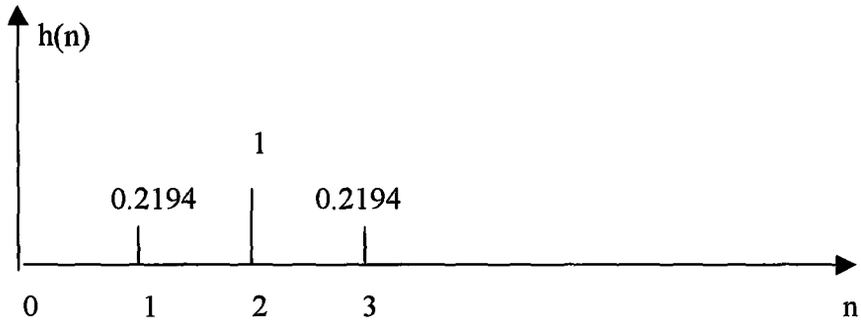
	10	100	1000	2000	3000
ρ(0)=	0.477138	0.366879	0.329881	0.386666	0.334555
ρ(1)=	1.060646	1.026954	0.991981	1.001078	0.998360
ρ(2)=	0.417137	0.378879	0.324880	0.336067	0.332222
ρ(3)=	0.026308	-0.009218	-0.000654	0.002043	0.0
ρ(4)=	-0.133262	0.027477	0.005331	0.009986	0.0
ρ(5)=	-0.080215	0.097563	0.020929		
ρ(6)=	-0.080215	0.094388	0.000802		
ρ(7)=	-0.173692	0.023477	-0.011706		
ρ(8)=	-0.027600	0.020698	-0.011890		
ρ(9)=	0.006738	0.005957	0.001508		

Tablo 6.5 Kanal girişi ile gürültülü çıkış arasındaki çapraz ilişki değerleri, W=3.5

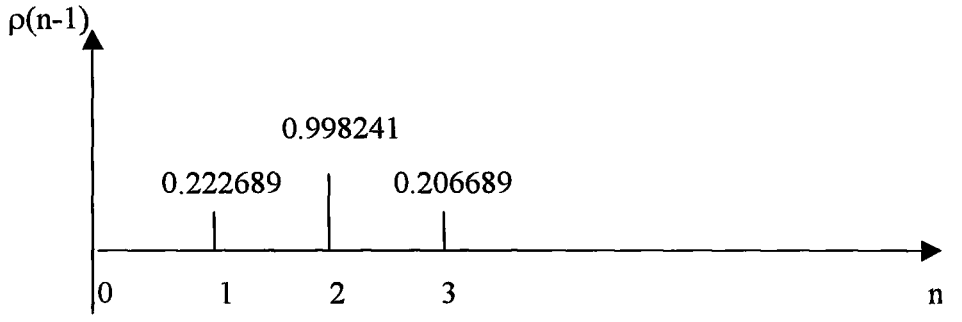
İterasyon sayısı

	10	**100**	**1000**	**2000**	**3000**
ρ(0)=	0.341400	0.419864	0.377219	0.385087	0.395583
ρ(1)=	0.976649	1.014010	0.987469	0.998793	1.004307
ρ(2)=	0.361400	0.375019	0.375019	0.382787	0.393932
ρ(3)=	-0.036649	0.058011	-0.009571	0.011696	0.0
ρ(4)=	-0.023351	0.051798	0.005630	0.0	0.0
ρ(5)=	0.056649	0.018670	-0.002232	0.0	0.0
ρ(6)=	-0.063351	-0.001773	-0.014082	0.0	0.0
ρ(7)=	-0.163351	-0.010443	-0.013168	0.0	0.0
ρ(8)=	-0.194486	-0.015124	0.002786	0.0	0.0
ρ(9)=	-0.021188	-0.007989	0.004044	0.0	0.0

Tablo 6.2-6.5 incelendiğinde her iterasyon değeri içinde ilk üç terim (ρ(0), ρ(1), ρ(2)) dışındaki terimlerin çok küçük oldukları görülmektedir. Yani a(n) ile u(n) arasında ilişkinin olmadığı sonucuna varılabilir.Ayrıca iterasyon sayısı arttıkça değerlerin sıfıra yaklaştığı, yani durağan durumda ilişkinin olmadığı daha açık görülmektedir. İlk terim incelendiğinde ise a(n) ile u(n) arasında bir ilişkinin olduğu ve bu ilişkinin kanalın impuls yanıtı h(n) ile bir bağlantısı olduğu görülmektedir (Şekil 6.1, 6.2). Çünkü h(n)'nin ilk üç terimi sıfırdan farklıdır.



Şekil 6.1 Kanalın impuls yanıtı (W=2.9)



Şekil 6.3 Çapraz ilişkinin ilk üç terimi, W=2.9

İstenen yanıt d(n) ile filtre çıkışı y(n) arasındaki çapraz ilişkinin incelenmesi,

Tablo 6.6 İstenen yanıt ile dengeleyici çıkışı arasındaki çapraz ilişki değerleri

İterasyon sayısı	Çapraz ilişki ρ(0)	İterasyon sayısı	Çapraz ilişki ρ(5)
0	0.0	180	0.7053328
1	0.012015	181	0.690724
2	0.015821	182	0.718091
3	0.020254	183	0.719822
4	0.023694	184	0.692509
5	0.033867	185	0.685423
6	0.048700	186	0.711006
7	0.057098	187	0.669995
8	0.054867	188	0.651594
9	0.066282	189	0.697246
10	0.077822	190	0.683047
11	0.089696	191	0.692795
12	0.090250	192	0.692795
13	0.104166	193	0.714581
14	0.113519	194	0.702309
15	0.121569	195	0.725768
16	0.125683	196	0.719760
17	0.122736	197	0.728534
18	0.133219	198	0.712663
19	0.141934	199	0.735408

Tablo 6.6'da değerlere baktığımızda ρ(0) zaman zaman sapmalar göstersede, bir(1)'e doğru bir artış görülmektedir. Ancak bir değerine ulaşmamaktadır. Bunun anlamı istenen yanıt ile dengeleyici çıkışı arasında bir ilişkinin olduğudur. Zaten amaç istenen yanıtı elde etmek olduğuna göre, bu sonuç doğru bir sonuç olarak karşımıza çıkmaktadır. Şekil 6.3'te de açıkça görülmektedir.

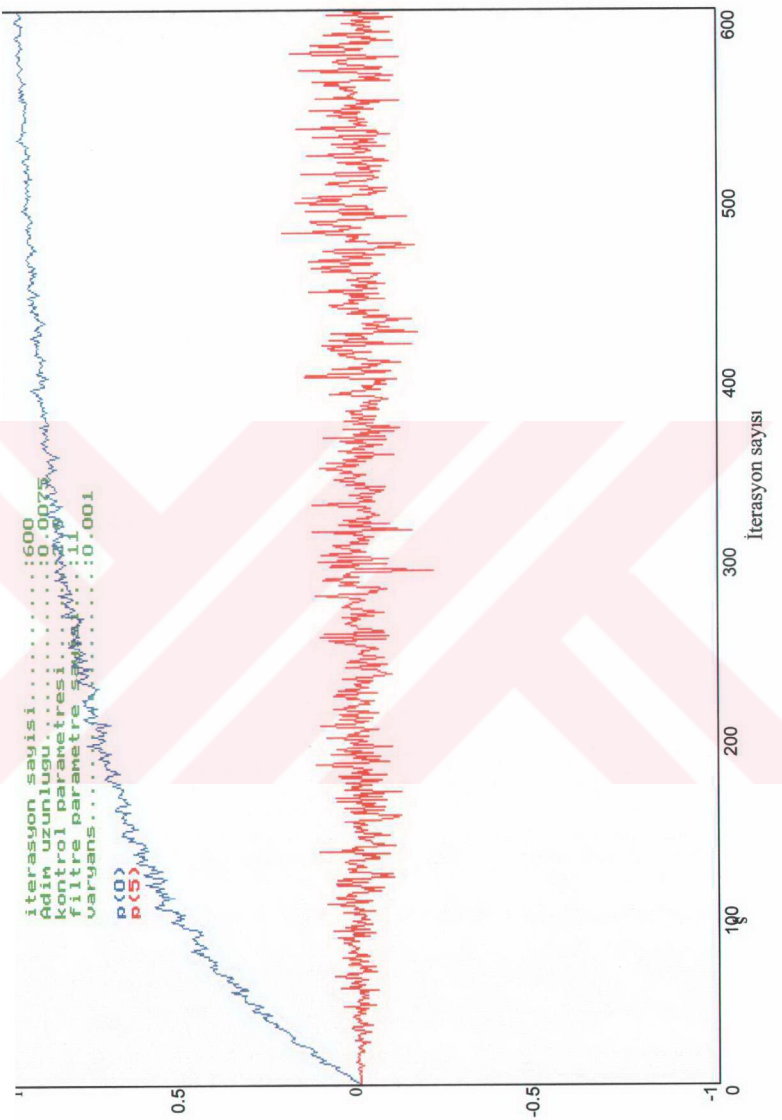
Öte yandan ρ(5) (ρ(5) herhangi bir değer olarak seçilmiştir) değerleri için Şekil.6.3'e tekrar bakıldığında, yukarda sözkonusu olan ilişkinin dengeleyici çıkışının geciktirilmiş bileşenleri ile istenen yanıt arasında olmadığı görülmektedir.

Dengeleyici giriş ve çıkışı arasındaki çapraz ilşkiyi inceliyelim,

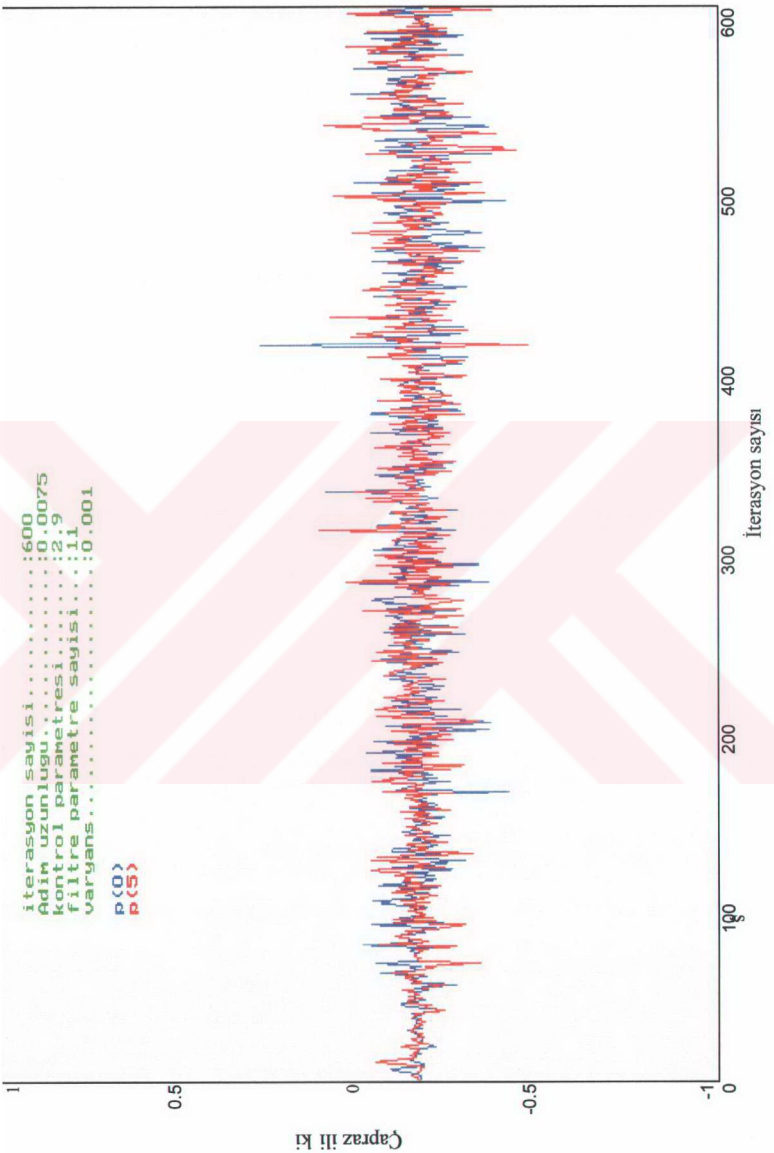
Şekil 6.4'te p(0), p(5) için elde edilen sonuçlara göre, bir çapraz ilişkinin olmadığı görülmektedir. Çünkü değeler sıfıra çok yakındır. Kimi zaman artı, kimi zaman eksi olmaktadır. Bu da bir ilişkinin olmadığının kanıtıdır.

Son olarak kanalın girişiyle dengeleyici çıkışı arasındaki ilişkiye ait Şekil 6.5'i inceliyelim. Şekil 6.4 için yaptığımız yorumlar burada da geçerlidir. Dolaysiyle araların bir ilişki yoktur.

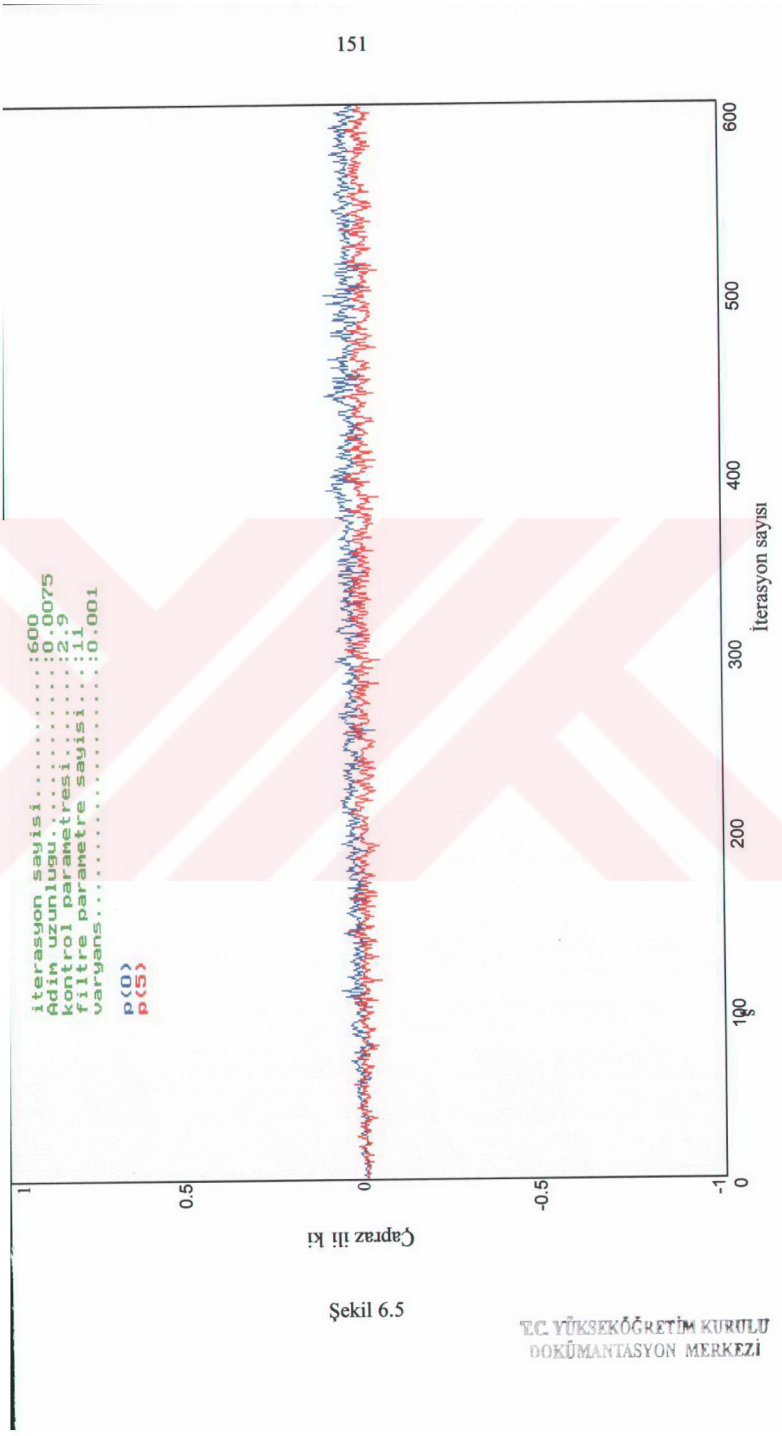
Toparlayacak olursak kanalın girişi ile gürültülü çıkışı ve istenen yanıt ile uyarlamalı dengeleyici çıkışı arasında ilişki vardır. Ancak dengeleyici girişi ile çıkışı ve dengeleyici çıkışı ile kanal girişi arasında bir ilişki yoktur.


iterasyon sayisi.....:600
Adim uzunlugu.....:0.0075
kontrol parametresi.....:1
filtre parametre sayisi.....:11
varyans.....:0.001
p(0)
p(5)
Çapraz ili ki
İterasyon sayısı
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600

Şekil 6.3


iterasyon sayisi.....:600
Adim uzunlugu.....:0.0075
kontrol parametresi.....:2.9
filtre parametre sayisi.....:11
varyans.....:0.001
p(0)
p(5)
Çapraz ili ki
İterasyon sayısı
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600

Şekil 6.4



Şekil 6.5

7 SONUÇLAR

Bu çalışmada, en küçük karesel ortalama algoritması (LMS) kullanılarak doğrusal kanalların uyarlamalı dengelenmesi incelenmiştir. Dengeleyicinin başarımı, farklı kanal modelleri için elde edilen öğrenme eğrileri gözönüne alınarak irdelenmiştir.

Öğrenme eğrilerinde başarım ölçütü olarak, yakınsama hızı ve karesel ortalama hatanın değeri gözönüne alınmıştır. Karesel ortalama hatanın en aza inmiş olması, dengeleyicinin aktarım işlevinin, kanalın aktarım işlevinin tersine benzemiş olduğu anlamına gelir.

Uyarlamalı kanal dengeleme çalışmalarında (simülasyonunda), LMS algoritmasının başarımı, adım uzunluğu, kanal kontrol parametresi, beyaz gürültünün varyansı ve dengeleyicinin parametre sayısı değerlerine bağlıdır.

Bu çalışmada, en küçük karesel ortalama algoritmasını kullanan bir uyarlamalı dengeleyici yardımıyla doğrusal kanalların dengelenmesine ilişkin sayısal uygulamalar gerçekleştirilmiştir. Bu algoritmanın başarımını ekileyen adım uzunluğu, gürültü varyansı, kanal kontrol parametresi ve dengeleyici katsayıları değiştirilerek öğrenme eğrileri bilgisayar simulasyonuyla elde edilmiştir. Bu parametrelerden adım uzunluğunun yakınsama hızı ve karesel ortalama hatanın durağan durum değeri üzerindeki etkileri, sözkonusu eğriler yardımıyla incelenmiştir. Adım uzunluğu arttıkça yakınsama hızının arttığı ancak, karesel ortalama hatanın durağan durum değerinin de büyük olduğu görülmüştür. Aynı biçimde, kanal kontrol parametresinin de yakınsama hızı ve karesel ortalama hatanın durağan durum değeri üzerindeki etkileri irdelenmiştir. Kanal kontrol parametresi büyüdükçe, özellikle karesel ortalama hatanın durağan durum değerinin hızla büyüdüğü görülmüştür. Gürültü varyansı arttıkça, karesel ortalama hatanın arttığı, ayrıca kanal kontrol parametresi de büyük olduğunda öğrenme eğrisindeki dalgalanmalarda büyük bir artış gözlenmiştir. Kanal kontrol parametresinin ve gürültü varyansının artmasına ek olarak, dengeleyici parametre sayısı da büyük olduğunda, öğrenme eğrilerindeki dalgalanmaların kontrol edilemeyecek kadar büyüdüğü görülmüştür. Dengeleyicinin parametre sayısının çok fazla artması, algoritmanın hesaplama süresini arttırmakta ve dolayısıyla durağan duruma geçiş süresi uzamaktadır.

Ayrıca bu çalışmada, sistemin farklı noktalarındaki işaretler arasındaki özilişki ve çapraz ilişkiler hesaplanarak diğer bölümlerde yapılan yorumlarla uyumu gözlenmiştir. Kanalın girişiyle gürültülü çıkışı arasında bir ilişkinin olduğu görülmüştür. Bu ilişkinin, kanalın impuls yanıtının uzunluğuna doğrudan bağlı olduğu sonucuna varılmıştır. Uyarlamalı dengeleyicinin çıkışı ile istenen yanıt arasındaki çapraz ilişki incelenmiş ve bir ilişkinin olduğu görülmüştür. Öte yandan kanal giriş dizisinin özilişkisi incelenmiş ve sıfıra yakın değerler elde edildiği için, bir özilişkinin olmadığı sonucuna varılmıştır. Ayrıca elde edilen değerlerden, uyarlamalı dengeleyicinin girişi ile çıkışı arasında bir ilişkinin olmadığı sonucuna varılmıştır. Bu sonuçların Beşinci Bölüm'de elde edilen sonuçlarla uyumlu olduğu görülmüştür.

KAYNAKLAR

Doop, L.J., (1953), Stochastic Processes, Wiley, New York

Gökgül, Özgür, "Yinelemeli En Küçük Kareler Kafes Algoritmasını kullanarak Dengeleme", Yüksek Lisans Tezi , Yıldız Teknik Üniversitesi.

Harutunyan, Kirkor, İstatistik Teorisine Giriş, Yıldız Teknik Üniversitesi Yayınları Sayı: 260

Hayes, M. H., (1996) Statistical Digital Signal Processing and Modeling, John Wiley & Sons, New York

Haykin, S., (1991), Adaptive Filter Theory, Prentice-Hall, Englewood Cliffs, New Tersy

Haykin , S., (1998), Digital Communications, Wiley, New York.

Kayran Ahmet H. , Panayırcı Erdal , Ümit Aygölü , Sayısal Haberleşme, Sistem yayıncılık

Lucky, R. W., (1965) " Automatic Equalization for Digital Communication" , Bell Syst. Tech. J., 44: 547-588

Lucky, R.W., Salz J. Ve Weldon E. J., (1968), Principles of Data Communication, Mc Graw-Hill, New York

Lucky, R.W., (1966), "Techniques for adaptive Equalization of Digital Communication Systems", Bell Syst. Tech. J., 45: 255-286.

Makhoul, J., (1988), "A Class of All-Zero Lattice Digital Filters: Properties and Applications", IEEE Trans. Acoust. Speech Signal Process., ASSP-26 : 304-314

Yücel, M. İşaret ve Sistem Analizinde Olasılık Yöntemleri, Yıldız Teknik Üniversitesi Yayınları Sayı: 228

Openheim, A.V. ve Schafer, R.W, (1975), Digital Signal Processing, Prentice-Hall, Englewood Cliffs, New Jersey.

Proakis, J. G., (1975), "Advances in Equalization for Intersymbol Interference", in Advances in Communication Systems, Academic Press, New York, 4 : 123-198

Widrow, Bernard "Adaptive Filter Prosesing", Printice-Hill Inc. N.J. 07632

EKLER

EK-1 Farklı Adım Uzunluklarına Göre Öğrenme Eğrisini Çıkaran, C Dili ile Yazılmış Program.

EK-2 Farklı Kanal Kontrol Parametre Değerlerine Göre Öğrenme Eğrisini Çıkaran, C Dili ile Yazılmış Program.

EK-3 Farklı Filtre Katsayılarına Göre Öğrenme Eğrisini Çıkaran, C Dili ile Yazılmış Program.

EK-4 Kanal Girişinin Özilişki Matrisini Hesaplayan, C Dili ile Yazılmış Program.

EK-5 Kanal Girişi ile Gürültülü Çıkış Arasındaki Çapraz İlişki Matrisini Hesaplayan, C Dili ile Yazılmış Program.

EK-6 Uyarlamalı Dengeleyicinin Çıkışı ile İstenen Yanıt Arasındaki Çapraz İlişki Matrisini Hesaplayan, C Dili ile Yazılmış Proram.

EK-7 Kanalın Gürültülü Çıkışının Özilişki Matrisini Hesaplayan, C Dili ile Yazılmış Program.

EK-1

```
//adım uzunluğuna göre öğrenme eğrisi çizmer.
#include<stdlib.h>
#include<time.h>
                                        #include<math.h>
#include<conio.h>
#include<graphics.h>
#include<float.h>
#define REVR  480.0
#define SCALE (480.0)
#define VARIANCE  0.1
#define SAMP 700       //no. of sample
#define ITER 200       //no. of iteration
#define N 10          //no of filter coefficient-1
#define PI 3.14
//#define MU 0.0025
#define W  2.9
#define M  3          //no of channel coefficient

int  l,i,j,r,D;
//int  SAMP,ITER;
float MU;
float w[N+1]={0.0},E,Y,u[2000+1]={0.0};
int a[2000+1]={0};

double  sum_data[2000+1]={0.0};
double  sum_data1[2000+1]={0.0};
float h[M+1]={0.0};
void graph_draw(int,int);
void calc_output_adaptive_filter(void);
int plus_mines_one(void);
float calc_u_of_n(void);
float noise(void);
```

```
//************************MAIN FUNCTION*******************
int main(void)
        { int  k,k2,k1,m,n,pp=0;
            int g_driver=DETECT,g_mode,errorcode;
            detectgraph(&g_driver,&g_mode);
            clrscr();
            for(k=0;k<=SAMP;k++)
                    {
                     sum_data[k]=0.0;
                     sum_data1[k]=0.0;
                     a[k]=0;
                     u[k]=0.0;
                    }
            for(k=0;k<=10;k++)
                    {
                     w[k]=0.0;
                    }
            clrscr();
            srand((unsigned) time(NULL));
            initgraph(&g_driver,&g_mode,"C:\\TC\\BGI");
            errorcode=graphresult();
            if(errorcode!=grOk)
             {
                    printf("Grafik moduna gecemedim!.....!");
                    exit(1);
             }

            for(k2=1;k2<=4;k2++)
                    {
                     if(k2==1)
                                    MU=0.075;
                     if(k2==2)
                                    MU=0.025;
```

```
                if(k2==3)

                                MU=0.0075;
                if(k2==4)
                                MU=0.005;
                for(k=0;k<=SAMP;k++)
               {
                sum_data[k]=0.0;
                sum_data1[k]=0.0;
                a[k]=0;
                u[k]=0.0;
                }
        for(k=0;k<ITER;k++)
                {
                        for(k1=0;k1<=N;k1++)
                                {w[k1]=0.0;}


                 for(i=0;i<SAMP;i++)
                        {
                                u[0]=calc_u_of_n();
//                              printf("%d",ITER);
                                calc_output_adaptive_filter();
                                sum_data1[i]=E*E;
                                sum_data[i]=sum_data[i]+E*E;
                        }

                }

setcolor(k2);
if(k2==1)
        {   outtextxy(100,60-k2*10,"adim uzunlugu=0.075");
            setcolor(8);
            outtextxy(100,60+5,"kontrol parametresi=3.5");
```

```
                outtextxy(100,60+25,"varyans=0.1" );
                outtextxy(100,60+15,"filtre parametre sayisi=11" );

        }
    if(k2==2)
        {
                outtextxy(100,60-k2*10,"adim uzunlugu=0.025");
        }
    if(k2==3)
        {
                outtextxy(100,60-k2*10,"adim uzunlugu=0.0075");
        }
    if(k2==4)
        {
                outtextxy(100,60-k2*10,"adim uzunlugu=0.005");
        }
     graph_draw(ITER,k2);
    }
    getchar();
    closegraph();
    clrscr();

    for(k=0;k<SAMP;k++){
                                        ++pp;
            if(pp==24) {
                    pp=0;
                    clrscr();
                    }
            }

    return 0;

    }
void calc_output_adaptive_filter(void)
```

```
                {
                        int j,k;
                        Y=0.0;

                             for(j=0;j<=N;j++)
                                {
                                 Y+=u[j]*w[j];
                                 }
                        E=D-Y;
                        for(j=N;j>=0;j--)
                                {
                                 w[j]=w[j]+MU*E*u[j];
                                 if(j!=0)
                                 u[j]=u[j-1];
                                 }
                        return ;
                }

void graph_draw(int p,int color)
        {
         int t;

         setbkcolor(WHITE);
         setcolor(color);
         putpixel(0,REVR-SCALE*(sum_data[0]/p),5);
         for(t=1;t<SAMP;t++)
                 {
                         line((t-1),(REVR-SCALE*sum_data[t-1]/(p)),t,(REVR-
SCALE*sum_data[t]/(p)));
                 }
        }
//***********u() hesaplama****************
float calc_u_of_n(void)
                {
```

```
                float G=0.0,NOISE;
                a[1]=plus_mines_one();
                D=a[7];
                NOISE=noise();

                for(l=1;l<=M;l++)
                        {
                        h[l]=(0.5)*(1+cos(2*PI*(l-2)/W));
                        G=G+a[l]*h[l];
                        };
                for(l=M+4;l>=1;l--){
                        a[l]=a[l-1];
                        }
        return G+NOISE;
        }
int plus_mines_one(void)
        {
        int r;
        r=rand()%10;
        return r>=5?1:-1;
        }
float noise(void)
        {  float nois_e,rr;
                int r;
                r=rand()%10;
                if(r==0)
                        {
                                r=1;
                        }
                rr=(float)r/10;
         if(r<=5)
                {
                nois_e=VARIANCE*sqrt(-2*log(rr))*sin(2*PI*rr);
                }
```

```
        nois_e=VARIANCE*sqrt(-2*log(rr))*cos(2*PI*rr);
        return nois_e;
}
```

EK-2

```
// kanal kontrol parametresine göre öğrenme  eğrisi çizer.
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
#include<conio.h>
#include<graphics.h>
#include<float.h>
#define REVR  480.0
#define SCALE (480.0)
#define VARIANCE  0.1
#define SAMP 700       //no. of sample
#define ITER 200       //no. of iteration
#define N 6          //no of filter coefficient
#define PI 3.14
#define MU 0.075
//#define W  2.9
#define M  3          //no of channel coefficient

int  l,i,j,r,D;
//int  SAMP,ITER;
//float MU;
float w[N+1]={0.0},E,Y,W,u[2000+1]={0.0};
int a[2000+1]={0};

double  sum_data[2000+1]={0.0};
double  sum_data1[2000+1]={0.0};
float h[M+1]={0.0};
void graph_draw(int,int);
void calc_output_adaptive_filter(void);
int plus_mines_one(void);
float calc_u_of_n(void);
```

```
float noise(void);
 //************************MAIN FUNCTION*******************
int main(void)

                { int  k,k2,k1,m,n,pp=0;
                        int g_driver=DETECT,g_mode,errorcode;
                        detectgraph(&g_driver,&g_mode);
                        clrscr();
                        for(k=0;k<=SAMP;k++)
                                {
                                 sum_data[k]=0.0;
                                 sum_data1[k]=0.0;
                                 a[k]=0;
                                 u[k]=0.0;
                                }
                        for(k=0;k<=10;k++)
                                {
                                 w[k]=0.0;
                                }
                        clrscr();
                        srand((unsigned) time(NULL));
                        initgraph(&g_driver,&g_mode,"C:\\TC\\BGI");
                        errorcode=graphresult();
                        if(errorcode!=grOk)
                         {
                                printf("Grafik moduna gecemedim!.....!");
                                exit(1);
                         }

                        for(k2=1;k2<=4;k2++)
                                {
                                 if(k2==1)
                                                W=3.5;
                                 if(k2==2)
```

```
                W=3.3;
 if(k2==3)
                W=3.1;
 if(k2==4)
                W=2.9;
 for(k=0;k<=SAMP;k++)
{
 sum_data[k]=0.0;
 sum_data1[k]=0.0;
 a[k]=0;
 u[k]=0.0;
}


for(k=0;k<ITER;k++)
 {
        for(k1=0;k1<=N;k1++)
                {w[k1]=0.0;}


 for(i=0;i<SAMP;i++)
        {
                u[0]=calc_u_of_n();
```

```
                outtextxy(100,60+25,"varyans=0.1" );
                outtextxy(100,60+15,"filtre parametre sayisi=7" );
            }
        if(k2==2)
            {
                outtextxy(100,60-(k2+1)*10,"Kontrol parametresi=3.3");
            }
        if(k2==3)
            {
                outtextxy(100,60-(k2-1)*10,"Kontrol parametresi=3.1");
            }
        if(k2==4)
            {
                outtextxy(100,60-(k2-3)*10,"Kontrol parametresi=2.9");
            }
            graph_draw(ITER,k2);
        }
         getchar();
        closegraph();
        clrscr();
        for(k=0;k<SAMP;k++){
                                        ++pp;
            if(pp==24) {
                pp=0;
                clrscr();
                }
            }
        return 0;

    }
void calc_output_adaptive_filter(void)
            {
                int j,k;
                Y=0.0;
```

```
                for(j=0;j<=N;j++)
                        {
                         Y+=u[j]*w[j];
                        }
                E=D-Y;
                for(j=N;j>=0;j--)
                        {
                         w[j]=w[j]+MU*E*u[j];
                         if(j!=0)
                         u[j]=u[j-1];
                        }
                return ;
        }



void graph_draw(int p,int color)
        {
         int t;
         setcolor(color);
         setbkcolor(WHITE);
         putpixel(0,REVR-SCALE*(sum_data[0]/p),5);
         for(t=1;t<SAMP;t++)
                {
                        line((t-1),(REVR-SCALE*sum_data[t-1]/(p)),t,(REVR-
SCALE*sum_data[t]/(p)));
                }
        }



float calc_u_of_n(void)
                {
```

```
        float G=0.0,NOISE;
        a[1]=plus_mines_one();
        D=a[5];
        NOISE=noise();
        for(l=1;l<=M;l++)
                {
                h[l]=(0.5)*(1+cos(2*PI*(l-2)/W));
                G=G+a[l]*h[l];
                };
        for(l=M+4;l>=1;l--){
                a[l]=a[l-1];
                }
    return G+NOISE;

    }
```

```
int plus_mines_one(void)
    {
    int r;
    r=rand()%10;
    return r>=5?1:-1;
    }
```

```
float noise(void)
    {  float nois_e,rr;
        int r;
        r=rand()%10;
        if(r==0)
```

```
                {
                        r=1;
                }
        rr=(float)r/10;
 if(r<=5)
        {
         nois_e=VARIANCE*sqrt(-2*log(rr))*sin(2*PI*rr);
        }
        nois_e=VARIANCE*sqrt(-2*log(rr))*cos(2*PI*rr);
        return nois_e;
}
```

EK-3

```
// filtre katsayılarına göre  öğrenme eğrisi çizer.
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
#include<conio.h>
#include<graphics.h>
#include<float.h>
#define REVR  480.0
#define SCALE (480.0)
#define VARIANCE  0.1
#define SAMP 700       //no. of sample
#define ITER 200       //no. of iteration
#define PI 3.14
#define W  3.5
#define M  3          //no of channel coefficient
#define MU 0.05
int  l,i,j,r,D,N,GECIKME,renk;
//float MU;
float w[10+1]={0.0},E,Y,u[2000+1]={0.0};
int a[2000+1]={0};

double  sum_data[2000+1]={0.0};
double  sum_data1[2000+1]={0.0};
float h[M+1]={0.0};
void graph_draw(int,int);
void calc_output_adaptive_filter(void);
int plus_mines_one(void);
float calc_u_of_n(void);
float noise(void);
 //**********************MAIN FUNCTION*******************
int main(void)
```

```
{ int  k,k2,k1,m,n,pp=0;
        int g_driver=DETECT,g_mode,errorcode;
        detectgraph(&g_driver,&g_mode);
        clrscr();
        for(k=0;k<=SAMP;k++)
                {
                 sum_data[k]=0.0;
                 sum_data1[k]=0.0;
                 a[k]=0;
                 u[k]=0.0;
                }
        for(k=0;k<=10;k++)
                {
                 w[k]=0.0;
                }
        srand((unsigned) time(NULL));
        initgraph(&g_driver,&g_mode,"C:\\TC\\BGI");
        errorcode=graphresult();
        if(errorcode!=grOk)
          {
                 printf("Grafik moduna gecemedim!.....!");
                 exit(1);
          }
        for(k2=1;k2<=3;k2++)
                {
                 if(k2==1){ GECIKME=5;
                                 N=6;
                                 renk=BLUE;}
                 if(k2==2) {GECIKME=6;
                                 N=8;
                                 renk=RED;}
                 if(k2==3) {GECIKME=7;
                                 N=10;
```

```
                                renk=GREEN;}
                 for(k=0;k<=SAMP;k++)
                {
                 sum_data[k]=0.0;
                 sum_data1[k]=0.0;
                 a[k]=0;
                 u[k]=0.0;
                }

        for(k=0;k<ITER;k++)
                {
                        for(k1=0;k1<=N;k1++)
                                {w[k1]=0.0;}
                 for(i=0;i<SAMP;i++)
                        {
                                u[0]=calc_u_of_n();
//                              printf("%d",ITER);
                                calc_output_adaptive_filter();
                                sum_data1[i]=E*E;
//                              printf("%f",sum_data1[i]);
                                sum_data[i]=sum_data[i]+E*E;
                        }

                }

setcolor(renk);
if(k2==1)
        {   outtextxy(100,60-k2*10,"filtre parametre sayisi=7");
            setcolor(8);
            outtextxy(100,60+5,"kontrol parametresi=3.5");
            outtextxy(100,60+25,"varyans=0.1" );
            outtextxy(100,60+15,"adim uzunlugu=0.05" );
        }
if(k2==2)
```

```
                {
                        outtextxy(100,60-k2*10,"filtre parametre sayisi=9");
                }
        if(k2==3)
                {
                        outtextxy(100,60-k2*10,"filtre parametre sayisi=11");
                }
//              printf("%d",k2);
         graph_draw(ITER,renk);
        }
        getchar();

        closegraph();
        clrscr();

        for(k=0;k<SAMP;k++){
                        ++pp;
                if(pp==24) {
                        pp=0;
                        clrscr();
                        }
                }
        return 0;
       }
void calc_output_adaptive_filter(void)
                {       int j,k;
                        Y=0.0;
                        for(j=0;j<=N;j++)
                                {
                                 Y+=u[j]*w[j];

                                }
                        E=D-Y;
                        for(j=N;j>=0;j--)
```

```
                                {
                                 w[j]=w[j]+MU*E*u[j];
                                 if(j!=0)
                                 u[j]=u[j-1];
                                 }
                        return ;
                }

void graph_draw(int p,int color)
        {
         int t;
         setbkcolor(WHITE);
         setcolor(color);
         putpixel(0,REVR-SCALE*(sum_data[0]/p),5);
         for(t=1;t<SAMP;t++)
                 {
                         line((t-1),(REVR-SCALE*sum_data[t-1]/(p)),t,(REVR-
SCALE*sum_data[t]/(p)));
                 }
        }
//***********u() hesaplama*****************
float calc_u_of_n(void)
                        {
                                float G=0.0,NOISE;
                                a[1]=plus_mines_one();
                                D=a[GECIKME];
                                NOISE=noise();
                                for(l=1;l<=M;l++)
                                        {
                                         h[l]=(0.5)*(1+cos(2*PI*(l-2)/W));
                                         G=G+a[l]*h[l];
                                        };

                                for(l=GECIKME;l>=1;l--){
```

```
                    a[l]=a[l-1];
                    }
                return G+NOISE;

                }
int plus_mines_one(void)
                {
                int r;
                r=rand()%10;
                return r>=5?1:-1;
                }
float noise(void)
                {  float nois_e,rr;
                    int r;
                    r=rand()%10;
                    if(r==0)
                        {
                            r=1;
                        }
                    rr=(float)r/10;
                 if(r<=5)
                     {
                     nois_e=VARIANCE*sqrt(-2*log(rr))*sin(2*PI*rr);
                     }
                    nois_e=VARIANCE*sqrt(-2*log(rr))*cos(2*PI*rr);
                    return nois_e;
                }
```

EK-4

```
//kanal girişinin özilişki matrisini hesaplar.
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
#include<conio.h>
#include<graphics.h>
#include<float.h>
#define REVR  480.0
#define SCALE (480.0)
#define N 10          //no of filter coefficient
#define PI 3.14
#define M  3          //no of channel coefficient
int  l,i,j,D;
int  SAMP,ITER;
float MU;
float w[N+1]={0.0},E,Y,W,u[2000+1]={0.0};
int a[2000+1]={0};
float sum_a[2000+1]={0};
double  sum_data[2000+1]={0.0};
double  sum_data1[2000+1]={0.0};
int plus_mines_one(void);


 //************************MAIN FUNCTION*******************
int main(void)
                {
                int  r=0,d,r2,r1=0,tt,ITERASYON;
                float k0=0.0;
                        clrscr();
                        gotoxy(30,12);
                        printf("ornek sayisini.....:");
```

```
                scanf(" %d",&SAMP);
                gotoxy(30,13);
                printf("iterasyon sayisini.....:");

                 scanf(" %d",&ITERASYON);
                getchar();
                clrscr();
                srand((unsigned) time(NULL));
for(tt=0;tt<ITERASYON;tt++)
  {  for(l=SAMP*2-1;l>=0;l--)
            {
             a[l]=plus_mines_one();
            }
    d=0;
    for(r1=0;r1<SAMP;r1++)
        {
         for(l=0;l<SAMP;l++)
           {
               r=r+a[l]*a[d+l];
           }
         d++;
         k0=r/(float)(SAMP);
         sum_a[r1]=sum_a[r1]+k0;
         r=0;
        }
     }
     for(l=0;l<SAMP;l++)
     printf("r[%d]..=%f\n",l,sum_a[l]/(float)ITERASYON);
          getchar();
          return 0;
          }
```

```
//************************************************************

int plus_mines_one(void)
                        {
                         int r;
                         r=rand()%10;
                         return r>=5?1:-1;
                        }
```

EK-5

```
//kontrol parametresi disardan giriliyor.
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
#include<conio.h>
#include<graphics.h>
#include<float.h>
#define REVR  480.0
#define SCALE (480.0)
#define N 10          //no of filter coefficient
#define PI 3.14
#define M  3          //no of channel coefficient
int  l,i,j,D;
int  SAMP,ITER,ITERASYON;
float MU;
float w[N+1]={0.0},E,Y,W=3.3,u[2000+1]={0.0};
int a[2000+1]={0},c1[2000+1]={0};
float sum_ua[2000+1]={0.0};
double  sum_data[2000+1]={0.0};
double  sum_data1[2000+1]={0.0};
float h[M+1]={0.0};
int plus_mines_one(void);
float calc_u_of_n(void);
 //************************MAIN FUNCTION*******************
int main(void)
        {       int  r2,l,c,r1=0,IT=0;
                float r,k0=0.0;
                        clrscr();
                        gotoxy(30,12);
                        printf("ornek sayisini.....:");
                        scanf(" %d",&SAMP);
```

```
                gotoxy(30,13);

                printf("iterasyon sayisini.....:");
                scanf(" %d",&ITERASYON);
                getchar();
                clrscr();
                srand((unsigned) time(NULL));
                r=0.0;
                //****r[0] hesaplama
        for(IT=0;IT<ITERASYON;IT++)
            {
            for(j=SAMP*2;j>=0;j--)
              {
               u[j]=calc_u_of_n();
//            printf("u[%d]..=%f\n",j,u[j]);
//             printf("c1[%d]..=%d\n",j,c1[j]);
              }
//            getchar();
            for(l=0;l<SAMP;l++)
              {
               for(c=0;c<SAMP;c++)
                    {
                     r=r+u[c]*c1[c+l];
                    }
               k0=r/((float)(SAMP));
//             printf("r[%d]..=%f\n",l,r/((float)(SAMP-l)));
               sum_ua[l]=sum_ua[l]+k0;
               r=0.0;
              }
            }
        for(l=0;l<SAMP;l++)
            {
             printf("sum_ua[%d]=%f\n",l,sum_ua[l]/(float)ITERASYON);
            }
```

```
        getchar();
        return 0;
    }
//*************************************************************
float calc_u_of_n(void)
                    {
                        float G=0.0;
                        a[1]=plus_mines_one();
                        c1[j]=a[1];
                        D=a[7];
                        for(l=1;l<=M;l++)
                            {
                             h[l]=(0.5)*(1+cos(2*PI*(l-2)/W));
                             G=G+a[l]*h[l];
                            };
                        for(l=M+4;l>=1;l--){
                            // printf("a[%d]=%d..;",l,a[l]);
                             a[l]=a[l-1];
                             //printf("c1[%d]=%d..;",l,a[l]);
                          }
                    return G;
                    }
//*************************************************************************
 int plus_mines_one(void)
                    {
                    int r;
                    r=rand()%10;
                    return r>=5?1:-1;
                    }
```

EK-6

```
//istenen yanıt ile uyarlamalı dengeleyicinin çıkışı arasındaki çapraz ilişki matrisini
hesaplar.
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
#include<conio.h>
#include<graphics.h>
#include<float.h>
#define REVR  480.0
#define SCALE (480.0)
#define N 10          //no of filter coefficient
#define PI 3.14
#define MU 0.0075
#define M  3          //no of channel coefficient
#define W  2.9
#define VARIANCE  0.01
int  l=0,ll=0,i,j=0,D,loop1;
int  SAMP,ITER;
float w[N+1]={0.0},E,Y,u[2000+1]={0.0},r=0.0;
int a[2000+1]={0},d[2000+1]={0};
double  sum_data[2000+1]={0.0};
double  sum_data1[1000+1]={0.0};
double  sum_data2[1000+1]={0.0};
float h[M+1]={0.0};
void calc_output_adaptive_filter(void);
void graph_draw(int,int);
int plus_mines_one(void);
float calc_u_of_n(void);
float noise(void);
 //************************MAIN FUNCTION*******************
int main(void)
```

```
{ int  k,k2,k1,m,n;
        int g_driver=DETECT,g_mode,errorcode;

                detectgraph(&g_driver,&g_mode);
                clrscr();
                gotoxy(30,12);
                printf("ornek sayisini.....:");
                scanf(" %d",&SAMP);
                gotoxy(30,13);
                printf("iterasyon sayisini....:");
                scanf(" %d",&ITER);
                getchar();
                clrscr();
                srand((unsigned) time(NULL));
                initgraph(&g_driver,&g_mode,"C:\\TC\\BGI\\");
                errorcode=graphresult();
                if(errorcode!=grOk)
                 {
                        printf("Grafik moduna gecemedim!.....!");
                        exit(1);
                 }
                for(k=0;k<=10;k++)
                  {
                   w[k]=0.0;
                  }
                srand((unsigned) time(NULL));

                for(k=0;k<=SAMP;k++)
                  {
                   sum_data[k]=0.0;
                   sum_data1[k]=0.0;
                   a[k]=0;
                   u[k]=0.0;
                  }
```

```
for(loop1=0;loop1<SAMP*2;loop1++)
        {d[loop1]=D;
         u[loop1]=calc_u_of_n();
        }
for(k=0;k<ITER;k++)
  {
   for(k1=0;k1<=N;k1++)
        {
         w[k1]=0.0;
        }
   for(i=0;i<SAMP+20;i++)
        {d[i]=D;
         calc_output_adaptive_filter();
         u[0]=calc_u_of_n();
         sum_data[i]=Y;
         if((i==10) || (i==SAMP-15))
          {if(i==SAMP-15){ m=SAMP-15;ll=1;
}//printf("iterasyon=%d",m);
           if(i==10){j=1;m=10;}
           for(n=i;n>=i-10;n--)
           {
            sum_data2[i-n+m]=sum_data2[i-
n+m]+d[i]*sum_data[n];
           // printf("p[%d]=%f\n",i-n,sum_data2[i-n]);
            }
           }
         } }

         if(j==1) {m=10;
         printf("iterasyon-j=%d\n",j);
         for(i=m;i<m+11;i++)
         {
          printf("p[%d]=%f\n",i,sum_data2[i]/(float)ITER);
         } }
```

```
                    getchar();
                    m=SAMP-15;
                   printf("iterasyon-ll=%d\n",ll);



                   for(i=m;i<m+11;i++)
                   {
                    printf("p[%d]=%f\n",i,sum_data2[i]/(float)ITER);
                   }



          graph_draw(ITER,10);
          getchar();
          return 0;
         }

//**********************end of main()*****************************

void calc_output_adaptive_filter(void)
                {
                        int j,k;
                        Y=0.0;
                        for(j=0;j<=N;j++)
                          {
                           Y+=u[j]*w[j];
                          }
                        E=D-Y;
                        for(j=N;j>=0;j--)
                          {
                           w[j]=w[j]+MU*E*u[j];
                           if(j!=0)
```

```
                u[j]=u[j-1];
                }
            return ;
        }


int plus_mines_one(void)
        {
        int r;
        r=rand()%10;
        return r>=5?1:-1;
        }

float calc_u_of_n(void)
        {
            float G=0.0,NOISE;
            a[1]=plus_mines_one();
            //d[loop1]=a[1];
            D=a[7];
            NOISE=noise();
            for(l=1;l<=M;l++)
                {
                h[l]=(0.5)*(1+cos(2*PI*(l-2)/W));
                G=G+a[l]*h[l];
                };
            for(l=M+4;l>=1;l--){
                a[l]=a[l-1];
                }
        return G+NOISE;
}

void graph_draw(int p,int color)
    {
```

```
        int t;
        setcolor(color);
        putpixel(0,REVR-SCALE*(sum_data2[0]),5);
        loop1=5;
        for(t=SAMP-15;t<SAMP;t=t+1)
                {       //loop1=loop1+10;
line((loop1),(REVR-SCALE*sum_data2[t-1]),loop1+20,(REVR-SCALE*sum_data2[t]));

                    loop1=loop1+20;
                }
        }

float noise(void)
                    { float nois_e,rr;
                        int r;
                        r=rand()%10;
                        if(r==0)
                            {
                                r=1;
                            }
                        rr=(float)r/10;
                     if(r<=5)
                        {
                         nois_e=VARIANCE*sqrt(-2*log(rr))*sin(2*PI*rr);
                         }
                        nois_e=VARIANCE*sqrt(-2*log(rr))*cos(2*PI*rr);
                        return nois_e;
                    }
```

EK-7

```
//kontrol parametresi disardan giriliyor.
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
#include<conio.h>
#include<graphics.h>
#include<float.h>
#define REVR  480.0
#define SCALE (480.0)
#define N 10          //no of filter coefficient
#define PI 3.14
#define M  3          //no of channel coefficient

int  l,i,j,D;
int  SAMP,ITER,ITERASYON;
float MU;
float w[N+1]={0.0},E,Y,W=2.9,u[2000+1]={0.0};
int a[2000+1]={0};
float sum_u[2000+1]={0.0};

double  sum_data[2000+1]={0.0};
double  sum_data1[2000+1]={0.0};
float h[M+1]={0.0};
int plus_mines_one(void);
float calc_u_of_n(void);
 //************************MAIN FUNCTION*******************
int main(void)
        {
                int  r2,l,c,r1=0,IT=0;
                float r,k0=0.0;
                        clrscr();
```

```
                gotoxy(30,12);
                printf("ornek sayisini.....:");
                scanf(" %d",&SAMP);
                gotoxy(30,13);
                printf("iterasyon sayisini.....:");
                scanf(" %d",&ITERASYON);
                getchar();
                clrscr();
                srand((unsigned) time(NULL));
                //****r[0] hesaplama
        for(IT=0;IT<ITERASYON;IT++)
            {
            for(c=SAMP*2-1;c>=0;c--)
              {
              u[c]=calc_u_of_n();
//            printf("u[%d]..=%f\n",c,u[c]);
              }

            for(l=0;l<SAMP;l++)
              {
              for(c=0;c<SAMP;c++)
                    {
                     r=r+u[c]*u[c+l];
                    }
              k0=r/((float)(SAMP));
//            printf("r[%d]..=%f\n",l,r/((float)(SAMP-l)));
              sum_u[l]=sum_u[l]+k0;
              r=0;
              }
            }
        for(l=0;l<SAMP;l++)
            {
             printf("sum_u[%d]=%f\n",l,sum_u[l]/(float)ITERASYON);
```

```
            }

        getchar();
        return 0;
    }
//*************************************************************
float calc_u_of_n(void)
                    {
                        float G=0.0;
                        a[1]=plus_mines_one();
                        //c1[1]=a[1];

                        D=a[7];
                        for(l=1;l<=M;l++)
                            {
                            h[l]=(0.5)*(1+cos(2*PI*(l-2)/W));
                            G=G+a[l]*h[l];
                            };
                        for(l=M+4;l>=1;l--){
                            a[l]=a[l-1];
//                          printf("c1[%d]=%d..;",l,a[l]);
                          }
                    return G;
                    }
//*************************************************************************
 int plus_mines_one(void)
                    {
                    int r;
                    r=rand()%10;
                    return r>=5?1:-1;
                    }
```

ÖZGEÇMİŞ

Doğum Tarih	15/09/1962	
Doğum Yeri	Pertek / Tunceli	
Lise	1977-1980	Pertek Lisesi
Lisans	1984-1989	ODTÜ- Gaziantep Yerleşkesi Müh. Fak. Elektrik-Elektronik Müh. Bölümü
Yüksek Lisans	1990-1999	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü
Çalıştığı Kurumlar	1990-1994	PROFİLO/TELRA
	1995-1996	PESTAŞ
	1996-1998	DELTA ELEKTRONİK
	1999-	ERICSSON