

REKÜRSİF EN KÜÇÜK KARE KAFES FİLTRELERİ

YÜKSEK LİSANS TEZİ

Müh. Sadık Arslan

Tezin Enstitüye Verildiği Tarih : 22 Haziran 1992

Tezin Savunulduğu Tarih : 8 Temmuz 1992

Tez Danışmanı : Doç. Dr. Ahmet H. Kayran

Diğer Jüri Üyeleri : Prof. Dr. Ergül Akçakaya

: Yard. Doç. Dr. Levent Sevgi

TEMMUZ 1992

ÖNSÖZ

Adaptif filtre teorisi ile ilgili bu çalışmada, adaptif filtrelerle, özellikle kafes filreleri ile ilgili çalışmaları olan değerli hocam Doç. Dr. Ahmet H. Kayran'a yardımlarından dolayı teşekkür ederim.

Sadık Arslan

20 Haziran 1992



İÇİNDEKİLER

ÖZET	V
SUMMARY	VI
BÖLÜM 1. GİRİŞ	1
BÖLÜM 2. DOĞRUSAL EN KÜÇÜK KARE PROBLEMİ	3
2.1 Vektöre Göre Türev	6
2.2 Deterministik Normal Denklem	8
2.2.1 Deterministik Normal Denklemin Düzenlenmesi	10
2.3 En Küçük Hata Kareleri Toplamı	11
BÖLÜM 3. DOĞRUSAL KESTİRİM	12
3.1 İleri Doğrusal Kestirim	12
3.2 Geri Doğrusal Kestirim	16
3.3 Ardıl (a posteriori) İleri kestirim Hatası	20
3.4 Ardıl (a posteriori) Geri Kestirim Hatası	23
BÖLÜM 4. REKÜRSİF EN KÜÇÜK KARE KAFES FİLTRE ALGORİTMASI	25
4.1 Derece Güncelleme Rekürsionları	25
4.1.1 $K_{m-1}(n)$ ve $L_{m-1}(n)$ Arasındaki İlişki ...	30
4.1.2 Ardıl Kestirim Hataları İçin Derece Güncelleme Rekürsionları	31
4.1.3 Ardıl Kestirim Hataları Ağırlıklı Toplamlarının Derece Güncelleme Rekürsionları	35
4.1.4 Dönüşüm Faktörü İçin Derece Güncelleme Rekürsionu	35
4.2 Zaman Güncelleme Rekürsionları	36
4.3 Rekürsif EKK Algoritmasının Özeti	39
4.4 Rekürsif EKK Algoritmasının Önkoşullandırması	40
4.5 Birleşik Süreç Kestirimi	42
4.5.1 Regresyon Katsayısı Vektörünün Rekürsif Olarak Elde Edilmesi	47
BÖLÜM 5. BİLGİSAYAR DENEYLERİ	51
5.1 Doğrusal Kestirim Deneyi	51
5.2 Adaptif Kanal Dengeleme Deneyi	52

SONUÇLAR VE ÖNERİLER	54
KAYNAKLAR	55
EKLER	57
ÖZGEÇMİŞ	85



ÖZET

Adaptif filtre teorisinde üç temel yaklaşım vardır. Bunlardan ikisi, Wiener Filtre Teorisi'ne dayanan yaklaşım ve Kalman Filtre Teorisi'ne dayanan yaklaşımdır. Bu iki yaklaşım da istatistiksel kökenli yani bir istatistiksel ön bilgiye dayalıdır. Üçüncü ve diğer yaklaşım, klasik en küçük kare yöntemine dayanır ve diğer iki yöntemden deterministik yapısıyla ayrılır. Bu incelemede esas konu olan rekürsif en küçük kare kafes filtreleri (recursive least-square lattice filters) ele alınmadan önce yukarıda adı geçen, temel konu, en küçük kare yöntemi incelenmiştir.

Wiener filtresinin tasarımı işlenecek veri hakkında istatistiksel bir bilgi gerektirir. Filtre, ancak bu istatistiksel bilginin işlenecek olan verinininkilere uyması durumunda optimumdur. En küçük kare yöntemi, bir istatistiksel ön bilgiye gerek olmaksızın doğrusal filtreleme problemini çözmek için kullanılır.

En küçük kare yöntemi anlatılırken, deterministik normal denklem çıkarılmıştır. Bu denklemin çözümü optimum en küçük kare yöntemi filtre katsayılarını vermektedir. Bu denklemin çözümünü matris tersi alma işlemi gerektirmektedir. Rekürsif algoritmalar bu ters alma işlemi olmaksızın bu katsayıları kestirme üzerinde durmaktadırlar.

Rekürsif en küçük kare algoritmasına temel teşkil etmesi açısından ileri ve geri doğrusal kestirim, doğrusal kestirimden yola çıkılarak anlatılmıştır. Her iki kestirim için ayrı ayrı normal ve genişletilmiş denklem çıkarılarak adım adım algoritmaya yaklaşılmıştır.

Hesaplama karışıklığı problemi, çok evreli kafes kestirici yapısının adaptif filtreyi gerçekleştirmede kullanılmasıyla çözülmüştür.

Rekürsif en küçük kare kafes filtre algoritması, derece ve zaman güncelleme rekürsifonları elde edilerek çıkarılmış, bu algoritmanın da yardımıyla birleşik süreç kestirimi algoritmasına geçilmiştir. Yapılan bilgisayar deneylerinde de algoritmanın olumlu sonuç verdiği gözlenmiştir.

RECURSIVE LEAST SQUARES LATTICE FILTERS

SUMMARY

Adaptive filters is also an adaptive system. An adaptive system requires a recursive operation which starts from some predetermined set of initial conditions. Recursive algorithms reduce hardware cost. Because if it is used a nonrecursive algorithms to compute system parameters the hardware must be more parallel that is more complex and more expensive.

A recursive algorithm makes it possible for the filter to perform satisfactorily in the environment where complete knowledge of the relevant signal characteristics is not available. The algorithm starts with predetermined conditions, representing complete ignorance about the environment. Yet, in a stationary environment, It is found that after after successive iteration of the algorithm it converges to the optimum Wiener solution in some statistical sense. In a nonstationary environment, the algorithm offers a tracking capability, whereby it can track time variations in the statistics of the input data, provided that the variations are sufficiently slow.

As a direct consequence of the application of a recursive algorithm, whereby the parameters of an adaptive filter are updated from one iteration to the next, the parameters become data dependent. This, therefore, means that an adaptive filters is a nonlinear device.

In another context, an adaptive filter is often referred to as linear in the sense that the estimate of a quantity of interest is obtained adaptively at the output of the filter as a linear combination of the available set of observations applied to the filter input.

A wide variety of recursive algorithms have been developed in the literature for the operation of adaptive filters. In the final analysis, the choice of one algorithm over another is determined by various factor. There are six basic factors which are considered. These are rate of convergence, misadjustment, robustnes, computational requirements, structure and numerical properties.

It will be useful to explain those factors in brief. Rate of convergence is defined as the number of iterations required for the algorithm. Misadjustment is defined as the dimensionless ratio of the steady-state value of the average excess mean squared-error to the minimum mean-squared error. Robustness refers to the ability of the algorithm to operate satisfactorily with ill conditioned data. Computational requirements are the number of operation -such as multiplications, divisions and additions /subtractions- required to make one complete iteration of the algorithm, the size of memory locations required to store the data and the program, and the investment required to program the algorithm on a computer. Structure refers to the structure of information flow in the algorithm, determining the manner in which it is implemented in hardware form. For example an algorithm whose structure exhibits high modularity, parallelism or concurrency is well-suited for information using very-large scale integration. The last factor to be explain is the Numerical properties. When an algorithm is implemented numerically, inaccuracies are produced due to round-off noise and representation errors in the computer. There are two issues of concern, namely, the manner in which error introduced at an arbitrary point in the algorithm propagates to future time instant, and the effect and amplification of round-off noise on the output. For example, certain algorithms are known to be unstable with respect to such errors, which makes them unstable for continuous adaptation, unless some special rescue devices are incorporated.

There are three basic approaches to adaptive filter theory, each of which offers desirable features of its own. Two of these are the approach based on Wiener Filter Theory and the approach based on Kalman Filter theory. The theory both filters is statistical-oriented. The other approach based on the classical method of least squares differs from these two in that it is deterministic in its formulation right from the start. According to the method of least square, it minimizes an index of performance that consists of the sum of weighted error squares, where the error or residual is defined as the difference between some desired response and the actual filter output. Depending on the structure used for implementing the adaptive filter, it can be identify three basically different classes of adaptive filtering algorithms that originate from the method of least squares. These are Recursive least-squares, recursive least-squares lattice, which is the issue of this paper, and QR decomposition -least squares lattice algorithms, respectively.

The design of a Wiener filter requires a priori information about the statistics of the data to be processed. The filter is optimum only when the statistical

characteristic of the input data match the a priori information on which the design of the filter is based. Method of least squares is used to solve the linear filtering problem, without invoking assumption on the statistics of the input applied to the filter. To illustrate the basic idea of least squares, suppose there is a set of real valued measurements $u(1), u(2), \dots, u(N)$, made at times $t_1, t_2, \dots, t_N$, respectively, and the requirement is and the requirement is to construct a curve that is used to fit these point in some optimum fashion. Let the time dependence of this curve be denoted by $f(t_i)$ and $u(i)$ for $i = 1, 2, \dots, N$; hence, the name of the method.

The method of least squares may be viewed as the deterministic counter part of Wiener filter theory. Basically, wiener filters are derived from ensemble averages with the result that one filter (optimum in a probabilistic sense) is obtained for all realizations of the operational environment, assumed to be wide-sense stationary. On the other hand, the method of least squares yields a different filter for each collection of input data.

The study begins by deriving the deterministic form of normal equation defining the tap weights of a linear transversal filter that produces an estimate of some desired response due to a set of inputs, which is a optimum in the least-squares sense. It is referred to the resulting filter as the least-squares filter.

The study continues by presenting the linear prediction problem. The problem of linear prediction may be viewed as a special type of parameter estimation that is well suited for the method of least squares. In this subject, the least squares transversal filter is used as the linear predictor. Linear prediction and its other forms, forward and backward linear prediction, are fundamental subject of famous least square algorithm such as the fast transversal filter algorithm (FTF) and the subject of this paper, the recursive least squares lattice filters.

Some information about the prediction methods mentioned above may be useful to make the subject appear. In forward linear prediction, it is used the set of inputs $u(i-1), u(i-2), \dots, u(i-M+1), u(i-M)$ to make a linear prediction of $u(i)$. According to this notation, the prediction is made at time $(i-1)$, one step into the future. It is defined the forward linear prediction error as the difference between the desired response $u(i)$ and the prediction output produced by the tap inputs $u(i-1), u(i-2), \dots, u(i-M)$. When the method of least squares is used to design the predictor, it is chosen its tap

weight so as to minimize the sum of forward prediction error energy. It is referred to this method of design a predictor as the forward linear prediction method.

In backward linear prediction, it is used the set of inputs $u(i-M+1), \dots, u(i-1), u(i)$ to make a linear prediction of $u(i-M)$. According to the notation, the prediction is made as time $(i-M+1)$, one step into the past. It is defined the backward linear prediction error as the difference between the desired response $u(i-M)$ and the predictor output produced by the tap inputs $u(i-M+1), \dots, u(i-1)$. When the method of least squares is used to design the predictor, we chose its tap weight so as to minimize the sum of backward prediction-error squares or the backward prediction-error energy. It is referred to this second method of designing a predictor as the backward linear prediction method.

In this paper the normal equation for forward linear prediction and the normal equation for backward linear prediction are presented in deterministic sense. Thus the augmented normal equation for forward linear prediction and augmented normal equation for backward linear prediction which are used to construct the recursive least square lattice algorithm are determined.

The issue of computational complexity is resolved by using a multistage lattice predictor as the structural basis of implementing the adaptive filter. This predictor consists of a cascade of stages, each in the form of a lattice; hence, the name. An important property of the multistage lattice predictor is that its individual stages are decoupled from each other in a time-averaged sense. This property is exploited in the derivation of the recursive least-squares lattice (LSL) algorithm that involves both time and order updates. The algorithm is rapidly converged, robust, and computationally efficient. Moreover, the highly pipelined, modular structure of the multistage lattice predictor and associated parts makes the recursive LSL algorithm well suited for implementation on a single silicon chip very large scale integration technology.

The FTF algorithm uses a maximum of four transversal filters with a common input, one of which defines impulse of response of the adaptive filter. In this paper it is described another class of exact least-squares algorithms based on a different structure, the multistage lattice predictor, which is modular in form. They are known collectively as least-square lattice algorithms, involving both order-update and time-update recursions. These algorithms are as efficient as the FTF algorithm in that they both essentially realize the same

rate of convergence at the expense of a computational cost that increases linearly with the number of adjustable tap weights. The class of LSL algorithms is also just as robust as the RSL and FTF algorithms in that that they both essentially insensitive to variations in the eigen value spread of the correlation matrix of the input signal.

The lattice predictors in this paper may operate in nonstationary environments, though with time-invariant reflection coefficients. As such, they are basically different from the lattice predictor that were based on the assumption of an underlying stationary environment.

Because of the basic structural differences between the LSL and FTF algorithms, these two algorithms present the relevant information in different ways. The FTF algorithm presents information about the input data in the form of instantaneous values of transversal filter coefficients. By contrast, the LSL algorithm presents the information in the form of a corresponding set of reflection coefficients. Such a difference may influence the choice of one algorithm over the other, depending of the application of interest.

Finally, as a result, the least squares lattice algorithm is suitable for the environment in which no prior statistical information about input data. It makes no difference whether input data stationary stochastic process or not. Because it presents deterministic solution to adaptive linear filtering problem. Moreover this algorithm is rapidly converged and more pipelined due to its structure.

In this work there are two computer experiments which use adaptive LSL algorithm. One is linear prediction and the other is adaptive channel equalization. Sample outputs of both experiment are also given in the end of the paper.

BÖLÜM 1. GİRİŞ

Adaptif filtre teorisinde üç temel yaklaşım vardır. Bunlardan ikisi, Wiener filtre teorisi'ne dayanan yaklaşım ve Kalman filtre teorisi'ne [1] dayanan yaklaşımdır. Bu iki yaklaşım da istatiksel kökenli yani bir istatiksel ön bilgiye dayalıdır. Üçüncü ve diğer yaklaşım klasik en küçük kareler yöntemine dayanır ve diğer iki yöntemden deterministik yapısıyla ayrılır. Bu incelemede esas konu olan rekürsif en küçük kareler kafes filtreleri (recursive least-squares lattice filters) ele alınmadan önce yukarıda adı geçen, temel konu, en küçük kareler yöntemi incelenmiştir.

Wiener filtre teorisindeki gibi en küçük kareler yönteminde de normal denklem (Wiener-Hopf denklemi) elde edilir ve bu denklemin çözümü en uygun filtre katsayılarını verir, fakat elde edilen bu normal denklem deterministiktir. İkinci bölümde bu denklemin elde edilişi anlatılacaktır.

En küçük kareler yöntemini kullanan bir çok doğrusal filtre algoritması mevcuttur. Bu algoritmaların hepsi çok benzerdir, sadece filtre yapılarıyla birbirlerinden ayrılırlar.

Hızlı transversal filtre (FTF) algoritması [2] rekürsif en küçük kareler çözümünü tam olarak elde eden güvenli bir yöntemdir. Modüler bir yapıda olan çok evrelili kafes kestirici (lattice predictor) yapısını kullanan en küçük kareler kafes (LSL) algoritması yapı olarak farklı olmasına rağmen FTF algoritması ile aynı derecede verimli ve aynı yakınsama oranına sahiptir. Her iki algoritmada da katsayı sayısı ile hesap sayısı oranı doğrusaldır. Yalnız, kafes filte yapısının modüler

olmasıyla fiziksel olarak gerekleme kolaylıđı sayesinde ok geniř lekli tmleřik devrelerle gerekleřtirme imkanı vardır. FTF algoritması gibi LSL algoritması da giriř iřareti iliřki matrisinin zdeđer saılımindan (eigenvalue spread) etkilenmez.

Bu alıřmada ele alınan kafes kestirici durađan olmayan ortamlarda alıřabilir ve zamanla deđiřen yansıma katsayılarına sahiptir.

Zaman ve derece gncelleme rekrsiyonu ile yansıma katsayıları rekrsif olarak gvenli bir řekilde elde edilir. Daha sonra birleřik sre kestirimi algoritması ıkarılmıřtır.



BÖLÜM 2. DOĞRUSAL EN KÜÇÜK KARE PROBLEMİ

Bir stokastik süreç iki değişken dizisiyle ifade edilmek istensin. Bu değişkenlerden biri $d(i)$; i anında gözlenen değer, diğeri ise, $u(i)$; i anındaki giriş değeri olsun. Bu durumda, w_{ok} modelin bilinmeyen parametreleri olmak üzere $d(i)$ 'nin doğrusal ifadesi şöyle yazılabilir:

$$d(i) = \sum_{k=1}^M w_k^* u(i-k+1) + e(i) \quad (2.1)$$

Bu ifadenin en sağında görülen terim, $e_0(i)$, ölçüm hatasını temsil eder. Şekil 1-1 de de işaret akış gösterimi verilen bu ifadeye çoklu doğrusal regresyon modeli adı verilir [3].

Ölçüm hatası olan ve ölçülemeyen $e(i)$ hakkında alışıla geldiği üzere şu kabuller yapılsın:

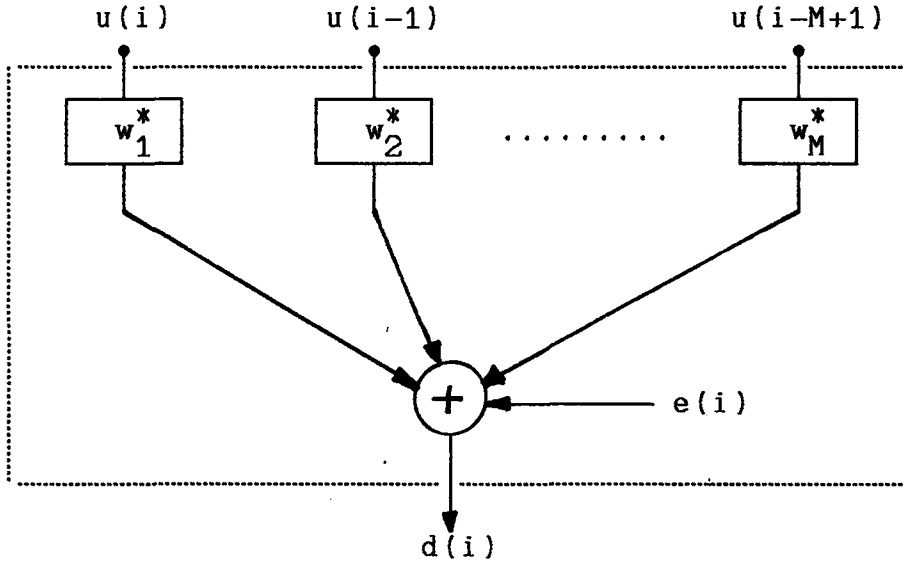
$$E[e(i)] = 0, \text{ her } i \text{ için}$$

ve

$$E[e(i) e^*(k)] = \begin{cases} \sigma^2, & i = k \\ 0, & i \neq k \end{cases}$$

Bu kabuller altına (1.1) ifadesi şu şekilde de yazılabilir:

$$E[d(i)] = \sum_{k=1}^M w_k^* u(i-k+1)$$



Şekil 2-1 Çoklu doğrusal regresyon modeli.

Problem, en az hata yapacak şekilde bilinmeyen parametreleri yani $w_1^*, w_2^*, \dots, w_M^*$ 'leri hesaplamaktır. Bu modelde, $u(i), u(i-1), \dots, u(i-M+1)$ bilinen değerler ve $d(i)$ 'nin de ortalama değeri teorik olarak belirlen bir değerdir.

Bu problem bir transversal filtre için düşünülürse, $d(i)$ arzulanan yanıt, toplamsal $u(i)$ filtre çıkışı ve $e(i)$ de kestirim hatasıdır. Şekil 2-1'de sözü edilen transversal filtre verilmistir.

Burada kestirim hatası şu şekilde yazılabilir:

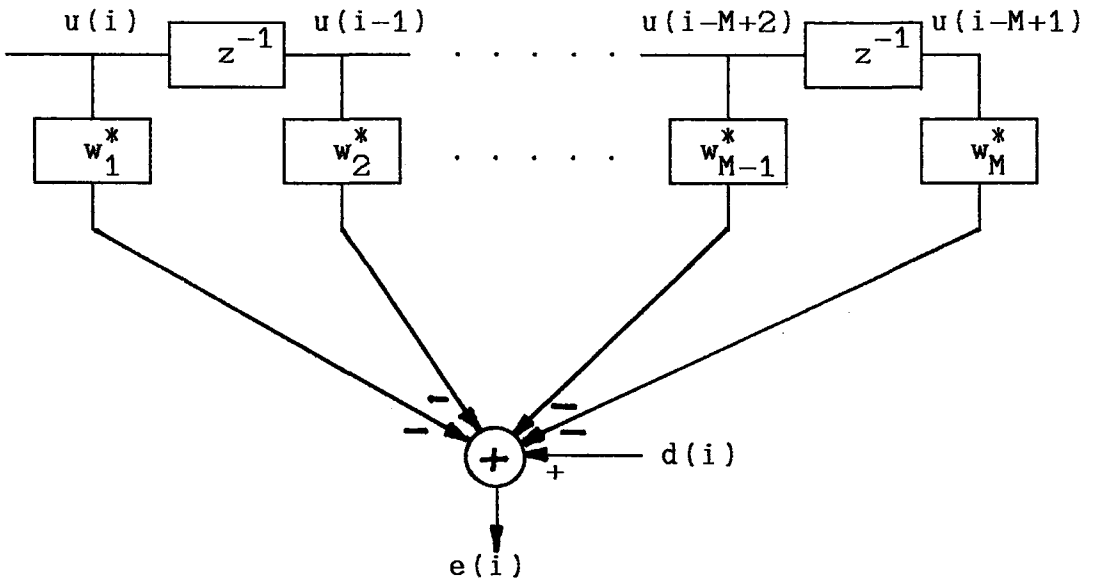
$$e(i) = d(i) - \sum_{k=1}^M w_k^* u(i-k+1) \quad (2.2)$$

En küçük kareler yöntemine göre arzulanan yanıtla en çok yaklaşmak için, belirlenen sınırlar içinde toplam karesel hata E , en küçük yapılmalıdır:

$$E(w_1, \dots, w_M) = \sum_{i=i_1}^{i_2} |e(i)|^2 \quad (2.3)$$

i_1 ve i_2 hatanın en küçük olduğu olduğu aralıktır. Bu aralığa pencereleme aralığı adı verilir. Temel olarak, kovaryans, otokorelasyon, ön-pencereleme (prewindowing), ve ileriden-pencereleme (postwindowing) olmak üzere dört farklı pencereleme yöntemi vardır [1]. Bu çalışmada kovaryans pencereleme yöntemi kullanılacaktır. Sıfır ortalamalı giriş işareti var ise pencerede kullanılan parametreler kovaryans matrisinin elemanlarını temsil ederler. $i_1=M$ ve $i_2=N$ dir ve $(1,N)$ aralığı dışında giriş verisi bir kabul yapılmaz. Burada M filtrenin derecesini, N ise giriş adedini gösterir. Sonuçta elde edilecek matris dikdörtgen şeklinde ve elemanlarını giriş değerlerinin oluşturduğu bir pencere matrisidir:

$$W(M,N) = \begin{bmatrix} u(M) & u(M+1) & \dots & u(N) \\ u(M-1) & u(M) & & u(N-1) \\ \vdots & \vdots & & \vdots \\ u(1) & u(2) & \dots & u(N-M+1) \end{bmatrix}$$



Şekil 1-2 Doğrusal transversal filtre modeli.

2.1 Vektöre Göre Türev

g , $M \times 1$ boyutlu $\underline{w}$ vektörünün skaler bir fonksiyonu olsun. Bu durumda $\underline{w}$ vektörünün k . elemanı w_k , a_k ve b_k sırasıyla gerçel ve sanal kısımlar olmak üzere

$$w_k = a_k + j b_k \quad (2.4)$$

şeklinde yazılabilir. $k=1,2,\dots,M$ olmak üzere g , a_k ve b_k lardan oluşan $2M$ değişkenli bir fonksiyon olarak tanımlanırsa $\underline{w}$ ya göre türevi aşağıdaki gibi yazılabilir.

$$\frac{\partial g}{\partial \underline{w}} = \begin{bmatrix} \frac{\partial g}{\partial a_1} + j \frac{\partial g}{\partial b_1} \\ \vdots \\ \frac{\partial g}{\partial a_M} + j \frac{\partial g}{\partial b_M} \end{bmatrix} \quad (2.5)$$

Bölüm 2.2 de anlatılan deterministik normal denklemin çıkarılmasında kullanılacak olan bazı özel durumlar aşağıda verilmiştir.

1. Durum:

$$g = \underline{c}^H \underline{w} \quad (\underline{c} \text{ ve } \underline{w}, M \times 1 \text{ vektörler})$$

Bu ifade açılarak yazılırsa:

$$\begin{aligned} g &= \sum_{k=1}^M c_k^* w_k \\ &= \sum_{k=1}^M c_k^* (a_k + j b_k) \end{aligned} \quad (2.6)$$

Böylece

$$\frac{\partial g}{\partial a_k} = c_k^* , \quad k = 1, 2, \dots, M \quad (2.7)$$

$$\frac{\partial g}{\partial b_k} = j c_k^* , \quad k = 1, 2, \dots, M \quad (2.8)$$

yazılır. (2.8), (2.7) ve (2.5) ifadeleri birleştirilirse

$$\frac{d}{dw} (\underline{c}^H \underline{w}) = \underline{0} \quad (2.9)$$

ifadesi elde edilir. Burada $\underline{0}$, $1 \times M$ sıfır vektördür.

2. Durum:

$$g = \underline{w}^H \underline{c}$$

Bu ifade açık şekilde yazılırsa

$$g = \sum_{k=1}^M c_k w_k^* \quad (2.10)$$

$$g = \sum_{k=1}^M c_k (a_k - j b_k)$$

Böylece

$$\frac{\partial g}{\partial a_k} = c_k , \quad k = 1, 2, \dots, M \quad (2.11)$$

$$\frac{\partial g}{\partial b_k} = -j c_k , \quad k = 1, 2, \dots, M \quad (2.12)$$

(2.11), (2.12) ve (2.5) ifadeleri birleştirilirse

$$\frac{d}{dw} (\underline{w}^H \underline{c}) = \underline{2c} \quad (2.13)$$

ifadesi elde edilir.

3. Durum:

$$g = \underline{w}^H \underline{Q} \underline{w} \quad (\underline{Q} : M \times M \text{ matris})$$

$$\underline{c}_1 = \underline{Q} \underline{w} \text{ olarak tanımlanırsa}$$

buradan $\underline{c}^H = \underline{w}^H \underline{Q}$ elde edilir. Bu ifade

yerine konduğu takdirde

$$g = c_1^H \underline{w} \quad \text{yazılır.}$$

c_1 e sabit gibi davranılıp, g , $\underline{w}$ ya göre türetilirse

1. Durum'daki sonucu kullanarak

$$\frac{d}{d\underline{w}} (c_1^H \underline{w}) = 0 \quad (2.14)$$

elde edilir.

$$c_2 = Q \underline{w} \quad \text{olarak tanımlanırsa}$$

$$g = \underline{w}^H c_2 \quad \text{yazılır.}$$

c_2 ye sabit gibi davranılarak, g , $\underline{w}$ ya göre türetilirse

2. Durum'daki sonucu kullanarak

$$\frac{d}{d\underline{w}} (\underline{w}^H c_2) = 2c_2 \quad (2.15)$$

elde edilir. (2.14) ve (2.15) deki ifadeler toplanırsa sonuç bulunmuş olur.

$$\frac{d}{d\underline{w}} (\underline{w}^H Q \underline{w}) = 2Q \underline{w} \quad (2.16)$$

2.2 Deterministik Normal Denklem

Kovaryans pencereleme yöntemi kullanıldığı zaman (2.3) ifadesindeki toplam karesel hata şöyle olur:

$$E(w_1, \dots, w_M) = \sum_{i=1}^N |e(i)|^2 \quad (2.17)$$

$$w_k \neq 0, \quad k = 1, \dots, M$$

En küçük kareler probleminin çözümü, daha önceden de söylendiği gibi bu hata ifadesini en küçük yapmaktır. Problemin çözümüne matrissel yolla gidilecektir. Sonuçta Şekil 1-2 deki transversal filtrenin en uygun katsayıları bulunacaktır.

$M \times 1$ boyutlu katsayı vektörü $\underline{w}$:

$$\underline{w}^T = [w_1, w_2, \dots, w_M] \quad (2.18)$$

$M \times 1$ boyutlu giriş vektörü $\underline{u}(i)$:

$$\underline{u}^T(i) = [u(i), u(i-1), \dots, u(i-M+1)] \quad (2.19)$$

$$M \leq i \leq N$$

olarak tanımlansın. Bu durumda kestirim hatası, (2.2) ifadesi kullanılarak şu şekilde yazılabilir:

$$e(i) = d(i) - \underline{w}^H \underline{u}(i), \quad M \leq i \leq N \quad (2.20)$$

$(N-M+1) \times 1$ boyutlu kestirim hata vektörü şu şekilde tanımlanabilir:

$$\underline{e}^H = [e(M), e(M+1), \dots, e(N)] \quad (2.21)$$

$(N-M+1) \times 1$ boyutlu arzulanan yanıt vektörü de şöyle tanımlanırsa

$$\underline{b}^H = [d(M), d(M+1), \dots, d(N)] \quad (2.22)$$

(2.20) eşitliği matris biçiminde yeniden yazılabilir:

$$\begin{aligned} \underline{e}^H &= \underline{b} - \underline{w}^H [\underline{u}(M), \underline{u}(M+1), \dots, \underline{u}(N)] \\ &= \underline{b}^H - \underline{w}^H A^H \end{aligned} \quad (2.23)$$

Burada,

$$A^H = [\underline{u}(M), \underline{u}(M+1), \dots, \underline{u}(N)] \text{ dir.} \quad (2.24)$$

(2.23) ifadesinin her iki yanına Hermit transpoz işlemi uygulanırsa

$$\underline{e} = \underline{b} - A \underline{w} \quad \text{olur.} \quad (2.25)$$

(2.17) ifadesi vektörel biçimde yazılırsa şöyle olur:

$$E(\underline{w}) = \underline{e}^H \underline{e} \quad (2.26)$$

(2.23), (2.25) ve (2.26) ifadeleri birleştirilirse

$$E(\underline{w}) = \underline{b}^H \underline{b} - \underline{b}^H A \underline{w} - \underline{w}^H A^H \underline{b} + \underline{w}^H A^H A \underline{w} \quad (2.27)$$

elde edilir. Bu ifadenin , (2.9), (2.13) ve (2.16) ifadelerindeki sonuçları kullanarak $\underline{w}$ ya göre türevi yazılırsa şu sonuç elde edilir:

$$\frac{dE(\underline{w})}{d\underline{w}} = -2A^H \underline{b} + 2 A^H A \underline{w} \quad (2.28)$$

Toplam karesel hata yani $E(\underline{w})$ nin en küçük olması için $dE/d\underline{w}$ nin sıfıra eşit olması gerekir. Buradan deterministik normal denklem yazılır:

$$A^H A \underline{w} = A^H \underline{b} \quad (2.29)$$

Bu deklemdaki $\underline{w}$ katsayı vektörü toplam karesel hatayı en küçük yapan çözümdür.

2.2.1 Deterministik Normal Denklemin Düzenlenmesi

(2.29) ifadesindeki normal denklemde görülen $A^H \underline{b}$ ve $A^H A$ nin özel anlamları vardır. $A^H \underline{b}$ çapraz-ilişki fonksiyonudur ve $\underline{x}$ ile temsil edilecektir:

$$\underline{x} = A^H \underline{b} \quad (2.30)$$

Bu ifadenin açılmış hali:

$$\underline{x} = \sum_{i=M}^N \underline{u}(i) \underline{d}^*(i) \quad (2.31)$$

$\underline{x}$ nin t. elemanı:

$$x(t) = \sum_{i=M}^N \underline{u}(i-t) \underline{d}^*(i), \quad 0 \leq t \leq M-1 \quad (2.32)$$

$A^H A$ ise ilişki matrisidir ve C ile temsil edilecektir:

$$C = A^H A \quad (2.33)$$

Bu ifade şöyle de yazılabilir:

$$C = \sum_{i=M}^N \underline{u}(i) \underline{u}^H(i) \quad (2.34)$$

İlişki matrisinin t,k. elemanı:

$$C(t,k) = \sum_{i=M}^N u(i-t) u^*(i-k), \quad 0 \leq t,k \leq M-1 \quad (2.35)$$

(2.30), (2.33) ve (2.29) ifadeleri birleştirilerek deterministik normal denklem yeniden yazılabilir:

$$C \underline{w} = \underline{x} \quad (2.36)$$

2.3 En Küçük Hata Kareleri Toplamı

En küçük kareler kestirimini kullanan transver-sal filtreye deterministik en küçük kareler filtresi adı verilir. Bu filtre için hata karelerinin toplamı en küçük değerdedir:

$$\begin{aligned} E_{\min} &= \underline{e}_{\min}^H \underline{e}_{\min} \\ &= (\underline{b}^H - \underline{w}^H A^H) (\underline{b} - A \underline{w}) \\ &= \underline{b}^H \underline{b} - \underline{w}^H A \underline{b} - \underline{b}^H A \underline{w} + \underline{w}^H A^H A \underline{w} \end{aligned} \quad (2.37)$$

(2.29) ve (2.37) ifadeleri birleştirilirse toplam karesel hata daha basit biçimde ifade edilir:

$$E_{\min} = \underline{b}^H \underline{b} - \underline{b}^H A \underline{w} \quad (2.38)$$

BÖLÜM 3. DOĞRUSAL KESTİRİM

Bu bölümde transversal filtrelerin kestirici (predictor) olarak kullanılması üzerinde durulacaktır. Aslında kafes yapısının temelinde olan ileri kestirici ve geri kestiriciden bahsedilecek bunlara ilişkin deterministik normal denklem ve genişletilmiş normal denklem ifadeleri çıkarılacaktır.

Kafes kestirici de bir yerde ileri ve geri kestiricilerin bileşimi sayılır. Kafes filtrenin yansıma katsayıları da bu ileri ve geri kestirim hataları cinsinden ifade edilecektir.

3.1 İleri Doğrusal Kestirim

Şekil 3-1 de görülen ileri doğrusal kestiricide $w_1, w_2, \dots, w_M$ katasayıları, $u(i-1), u(i-2), \dots, u(i-M)$ girişleri ve $u(i)$ de arzulanan çıkışı, yani kestirilmek istenen işareti göstermektedir. $f_M(i)$ ile gösterilen ileri kestirim hatası şöyle ifade edilir:

$$f_M(i) = u(i) - \sum_{k=1}^M w_k^* u(i-k) \quad (3.1)$$

Bu ifadedeki konvolüsyon toplamı filtre çıkışını göstermektedir. Görüldüğü gibi ileri kestirim hatası filtre çıkışı ile arzulanan çıkışın farkıdır. Bu ifade vektör biçiminde yazılırsa:

$$f_M(i) = u(i) - \underline{u}^T(i-1) \underline{w}^* \quad (3.2)$$

Kestiricinin son girişi $u(i-M)$ dir. Bu yüzden alt sınır

M+1 den başlamalıdır. Bu durumda i, M+1 ile N ile değer alacaktır.

İleri doğrusal kestiricinin giriş vektörü şudur:

$$\underline{u}_f^T(i) = \underline{u}^T(i-1) = [u(i-1), u(i-2), \dots, u(i-M)] \quad (3.3)$$

İleri doğrusal kestirici veri matrisi şöyle tanımlanır:

$$A_f^H = [\underline{u}_f(M+1), \underline{u}_f(M), \dots, \underline{u}_f(N)]$$

Bu ifade açılarak yazılırsa:

$$A_f^H = \begin{bmatrix} u(M) & u(M+1) & \dots & u(N-1) \\ u(M-1) & u(M) & & u(N-2) \\ \vdots & \vdots & & \vdots \\ u(1) & u(2) & \dots & u(N-M) \end{bmatrix} \quad (3.4)$$

(2.34) ifadesini kullanarak ileri doğrusal kestirici ilişki matrisi şöyle ifade edilir:

$$\begin{aligned} C_M &= \sum_{i=M+1}^N \underline{u}_f(i) \underline{u}_f^H(i) \\ &= \sum_{i=M+1}^N \underline{u}(i-1) \underline{u}^H(i-1) \end{aligned} \quad (3.5)$$

Bu ifade açılarak yazılırsa:

$$C_M = \sum_{i=M+1}^N \begin{bmatrix} u(i-1)u(i-1)^* & \dots & u(i-1)u(i-M)^* \\ u(i-2)u(i-1)^* & & u(i-2)u(i-M)^* \\ \vdots & & \vdots \\ u(i-M)u(i-1)^* & \dots & u(N-M)u(N-M)^* \end{bmatrix} \quad (3.6)$$

İleri doğrusal kestiricinin arzulanan yanıt vektörü şöyle yazılabilir:

$$b_f^H = [u(M+1), u(M+2), \dots, u(N)] \quad (3.7)$$

Çapraz ilişki fonksiyonu ise (2.30) ifadesinden:

$$\underline{x}_f = A_f^H \underline{b}_f \quad (3.8)$$

(3.4), (3.7) ve (3.8) ifadelerini kullanarak çapraz ilişki fonksiyonu açık olarak elde edilebilir:

$$\underline{x}_f = \begin{bmatrix} u(M+1)u(M+1)^* + \dots + u(N)u(N)^* \\ u(M)u(M+1)^* + \dots + u(N-1)u(N)^* \\ \vdots \\ u(1)u(M+1)^* + \dots + u(N-M)u(N)^* \end{bmatrix}$$

$$= \sum_{i=M+1}^N \begin{bmatrix} u(i-1)u(i)^* \\ u(i-2)u(i)^* \\ \vdots \\ u(i-M)u(i)^* \end{bmatrix} \quad (3.9)$$

(2.38) ifadesinden yararlanarak ileri doğrusal kestirim hata enerjisini şöyle yazabiliriz:

$$F_M = \underline{b}_f^H \underline{b}_f - \underline{b}_f^H A_f \underline{w} \quad (3.10)$$

(2.29) ifadesinden ileri doğrusal kestirici için deterministik normal denklem yazılabilir:

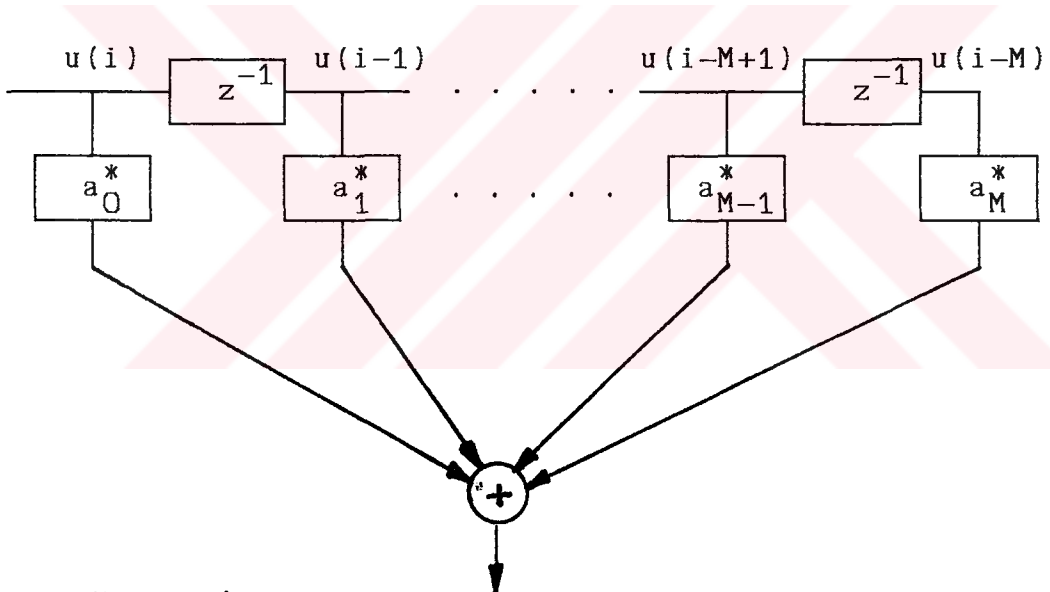
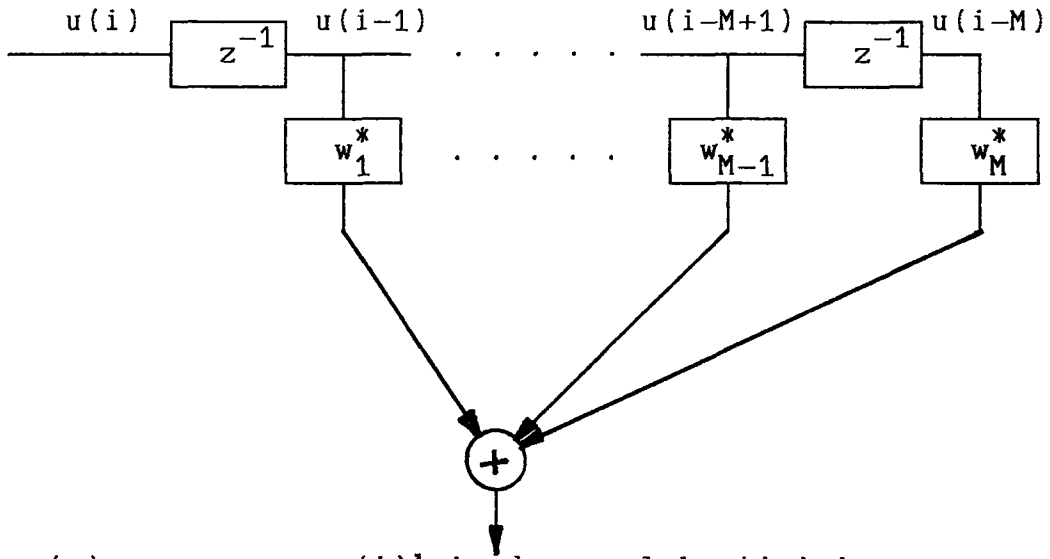
$$A_f^H A_f \underline{w} = A_f^H \underline{b} \quad (3.11)$$

(3.10) ve (3.11) ifadelerinden, ileri doğrusal kestirici için genişletilmiş normal denklem elde edilir:

$$\begin{bmatrix} \underline{b}_f^H \underline{b}_f & \underline{b}_f^H A_f \\ A_f^H \underline{b}_f & A_f^H A_f \end{bmatrix} \begin{bmatrix} 1 \\ -\underline{w} \end{bmatrix} = \begin{bmatrix} F_M \\ \underline{0} \end{bmatrix} \quad (3.12)$$

Burada $\underline{0}$ sıfır $M \times 1$ boyutlu sıfır vektördür. $\underline{b}_f^H \underline{b}_f$ ise (3.7) ifadesinden:

$$\underline{b}_f^H \underline{b}_f = \sum_{i=M+1}^N u(i)u(i)^* \quad (3.13)$$



Şekil 3-1 (a) M dereceli ileri doğrusal kestirici. (b) Karşılık gelen kestirim hata filtresi.

(2.36), (3.8) ve (3.13) ifadeleri kullanılarak (3.12) ifadesi yeniden yazılabilir:

$$\begin{bmatrix} \underline{b}_f^H & \underline{b} \\ \underline{x} & C_{fM} \end{bmatrix} \begin{bmatrix} 1 \\ -\underline{w} \end{bmatrix} = \begin{bmatrix} F_M \\ \underline{0} \end{bmatrix} \quad (3.12)$$

Bu ifade açılarak yazılırsa: (3.13)

$$\sum_{i=M+1}^N \begin{bmatrix} u(i)u(i)^* & u(i)u(i-1)^* & \dots & u(i)u(i-M)^* \\ u(i-1)u(i)^* & u(i-1)u(i-1)^* & \dots & u(i-1)u(i-M)^* \\ \vdots & \vdots & \ddots & \vdots \\ u(i-M)u(i)^* & u(i-M)u(i-1)^* & \dots & u(i-M)u(i-M)^* \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ -w \end{bmatrix} = \begin{bmatrix} F_M \\ \vdots \\ 0 \end{bmatrix}$$

Şekil 3-1.b'de görülen ileri kestirim hata filtresinde

$$\underline{a} = \begin{bmatrix} 1 \\ \vdots \\ -w \end{bmatrix} \quad \text{dir.} \quad (3.14)$$

Görüldüğü gibi (3.13) ifadesindeki ilişki matrisi C_{M+1} dir. Bu durumda (3.13) ifadesi yeniden yazılırsa:

$$C_{M+1} \underline{a} = \begin{bmatrix} F_M \\ \vdots \\ 0 \end{bmatrix} \quad (3.15)$$

Elde edilir. Bu ifadeye, ileri kestirim deterministik genişletilmiş normal denklem adı verilir.

3.2 Geri Doğrusal Kestirim

Şekil 3-2 de görülen geri doğrusal kestiricide $w_1, w_2, \dots, w_M$ katasayıları, $u(i), u(i-1), \dots, u(i-M+1)$ girişleri ve $u(i-M)$ de arzulanan çıkışı, yani kestirilmek istenen işareti göstermektedir. $b_M(i)$ ile gösterilen ileri kestirim hatası şöyle ifade edilir:

$$b_M(i) = u(i-M) - \sum_{k=1}^M g_k^* u(i-M+k) \quad (3.16)$$

Bu ifadedeki konvolüsyon toplamı filtre çıkışını göstermektedir. Görüldüğü gibi geri ileri kestirim hatası filtre çıkışı ile arzulanan çıkışın farkıdır. Bu ifade

vektör biçiminde yazılırsa:

$$\mathbf{b}_M(i) = \mathbf{u}(i-M) - \mathbf{u}^{BT}(i)\mathbf{a}^* \quad (3.17)$$

Kestiricinin arzulanan yanıtı $\mathbf{u}(i-M)$ dir. Bu yüzden alt sınır $M+1$ den başlamalıdır. Bu durumda i , $M+1$ ile N ile değer alacaktır [4].

Geri doğrusal kestiricinin giriş vektörü şudur:

$$\mathbf{u}_b^T(i) = \mathbf{u}(i)^{BT} = [\mathbf{u}(i-M+1), \dots, \mathbf{u}(i-1), \mathbf{u}(i)] \quad (3.18)$$

Geri doğrusal kestirici geri matrisi şöyle tanımlanır:

$$\mathbf{A}_b^H = [\mathbf{u}_b(M+1), \mathbf{u}_b(M), \dots, \mathbf{u}_b(N)]$$

Bu ifade açılarak yazılırsa:

$$\mathbf{A}_b^H = \begin{bmatrix} u(2) & u(3) & \dots & u(N-M+1) \\ u(3) & u(4) & \dots & u(N-M+2) \\ \vdots & \vdots & \ddots & \vdots \\ u(M+1) & u(M+2) & \dots & u(N) \end{bmatrix} \quad (3.19)$$

(2.34) ifadesini kullanarak ileri doğrusal kestirici ilişki matrisi şöyle ifade edilir:

$$\begin{aligned} C_M &= \sum_{i=M+1}^N \mathbf{u}_b(i) \mathbf{u}_b^H(i) \\ &= \sum_{i=M+1}^N \mathbf{u}^B(i) \mathbf{u}^{BH}(i) \end{aligned} \quad (3.20)$$

Bu ifade açılarak yazılırsa:

$$C_M = \sum_{i=M+1}^N \begin{bmatrix} u(i-M+1)u(i-M+1)^* & \dots & u(i-M+1)u(i)^* \\ u(i-M+2)u(i-M+1)^* & \dots & u(i-M+2)u(i)^* \\ \vdots & & \vdots \\ u(i)u(i-M+1)^* & \dots & u(i)u(i)^* \end{bmatrix} \quad (3.21)$$

Geri doğrusal kestiricinin arzulanan yanıt vektörü şöyle yazılabilir:

$$\mathbf{b}_b^H = [u(1), u(2), \dots, u(N-M)] \quad (3.22)$$

Çapraz ilişki fonksiyonu ise (2.30) ifadesinden:

$$\mathbf{x}_b = \mathbf{A}_b^H \mathbf{b}_b \quad (3.23)$$

(3.4), (3.7) ve (3.8) ifadelerini kullanarak çapraz ilişki fonksiyonu açık olarak elde edilebilir:

$$\begin{aligned} \mathbf{x}_b &= \begin{bmatrix} u(M+1)u(1)^* + \dots + u(N)u(N-M)^* \\ u(M)u(M+1)^* + \dots + u(N-1)u(N)^* \\ \vdots \\ u(1)u(1)^* + \dots + u(N-M)u(N-M)^* \end{bmatrix} \\ &= \sum_{i=M+1}^N \begin{bmatrix} u(i)u(i-M)^* \\ u(i-1)u(i-M)^* \\ \vdots \\ u(i-M)u(i-M)^* \end{bmatrix} \end{aligned} \quad (3.24)$$

(2.38) ifadesinden yararlanarak ileri doğrusal kestirim hata enerjisi şöyle yazılabilir:

$$B_M = \mathbf{b}_b^H \mathbf{b}_b - \mathbf{b}_b^H \mathbf{A}_b \mathbf{w} \quad (3.25)$$

(2.29) ifadesinden geri doğrusal kestirici için deterministik normal denklem yazılabilir:

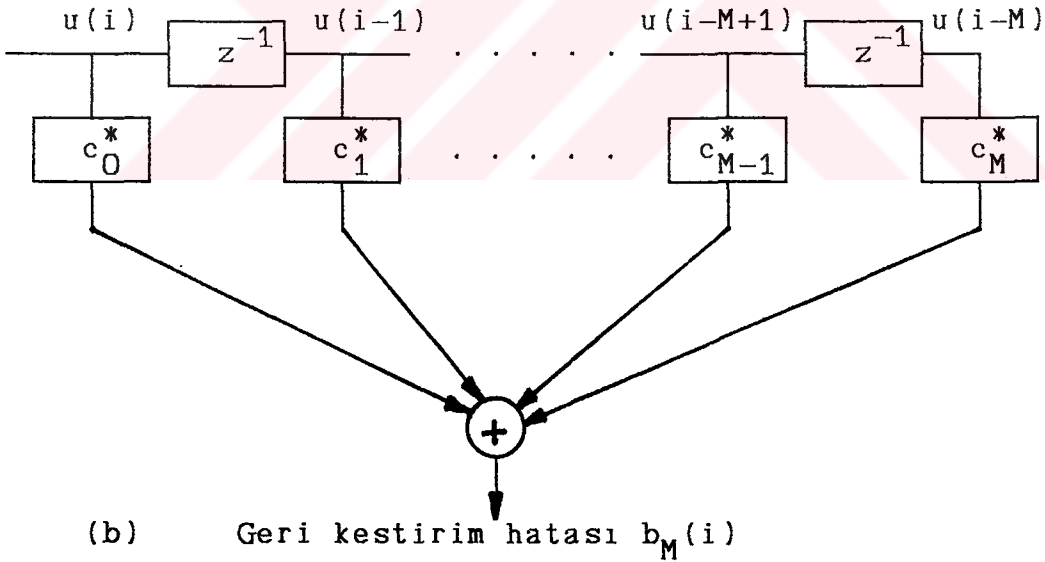
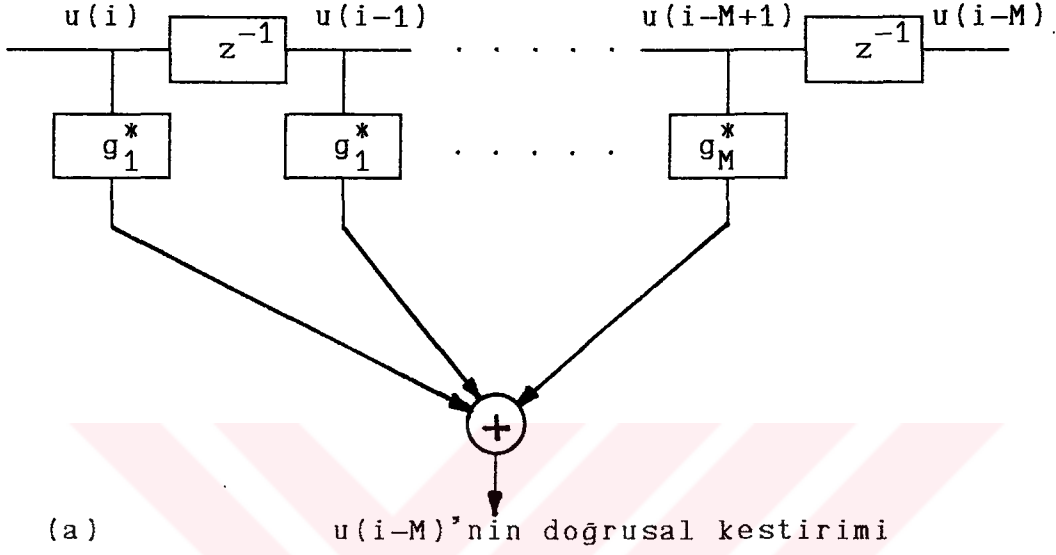
$$\mathbf{A}_b^H \mathbf{A}_b \mathbf{w} = \mathbf{A}_b^H \mathbf{b}_b \quad (3.26)$$

(3.25) ve (3.26) ifadelerinden, ileri doğrusal kestirici için genişletilmiş normal denklem elde edilir:

$$\begin{bmatrix} \mathbf{A}_b^H \mathbf{A}_b & \mathbf{A}_b^H \mathbf{b}_b \\ \mathbf{b}_b^H \mathbf{A}_b & \mathbf{b}_b^H \mathbf{b}_b \end{bmatrix} \begin{bmatrix} -\mathbf{q} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ B_M \end{bmatrix} \quad (3.27)$$

Burada $\underline{0}$ sıfır $M \times 1$ boyutlu sıfır vektördür. $\underline{b}_b^H \underline{b}_b$ ise (3.7) ifadesinden:

$$\underline{b}_b^H \underline{b}_b = \sum_{i=M+1}^N u(i-M)u(i-M)^* \quad (3.28)$$



Şekil 3-2 (a) M dereceli geri doğrusal kestirici. (b) Karşılık gelen kestirim hata filtresi.

(2.36), (3.8) ve (3.13) ifadeleri kullanılarak (3.12) ifadesi yeniden yazılabilir:

$$\begin{bmatrix} C_{fM} & \underline{x}_b \\ \underline{x}_b^H & \underline{b}_f^H \underline{b} \end{bmatrix} \begin{bmatrix} -\underline{g} \\ 1 \end{bmatrix} = \begin{bmatrix} \underline{0} \\ B_M \end{bmatrix} \quad (3.29)$$

Bu ifade açılarak yazılırsa: (3.30)

$$\sum_{i=M+1}^N \begin{bmatrix} u(i-M+1)u(i-M+1)^* & \dots & u(i-M+1)u(i)^* & u(i-M+1)u(i-1)^* \\ \vdots & & \vdots & \vdots \\ u(i)u(i-M+1)^* & \dots & u(i)u(i)^* & u(i)u(i-1)^* \\ \hline (i-1)u(i-M+1)^* & \dots & u(i-1)u(i)^* & u(i-1)u(i-1)^* \end{bmatrix} \begin{bmatrix} -\underline{g} \\ B_M \end{bmatrix} = \begin{bmatrix} \underline{0}_M \\ B_M \end{bmatrix}$$

Şekil 3-1.b'de görülen ileri kestirim hata filtresinde

$$= \begin{bmatrix} -\underline{g} \\ 1 \end{bmatrix} \quad \text{dir.} \quad (3.31)$$

Görüldüğü gibi (3.30) ifadesindeki ilişki matrisi C_{M+1} dir. Bu durumda (3.30) ifadesi yeniden yazılırsa:

$$C_{M+1} \underline{c} = \begin{bmatrix} \underline{0} \\ B_M \end{bmatrix} \quad (3.32)$$

Elde edilir. Bu ifadeye, ileri kestirim deterministik genişletilmiş normal denklem adı verilir.

3.3 Ardıl (a posteriori) İleri Kestirim Hatası

Şekil 3-3.a'da görülen M. dereceden ileri doğrusal kestiricinin, $\underline{w}(n)$ katsayı vektörü en küçük kare yöntemine göre $1 \leq i \leq n$ aralığında gözlenerek optimize edilmiş olsun. i anındaki ileri kestirim hatası şöyle

yazılabilir:

$$f_M(i) = u(i) - \underline{w}^H(n) \underline{u}_M(i-1) \quad (3.33)$$

Burada $u(i)$ arzu edilen cevap ve

$$\underline{u}_M^T(i-1) = [u(i-1), u(i-2), \dots, u(i-M)] \quad (3.34)$$

giriş vektörüdür.

Şekil 3-3.b de görülen ileri kestirim hata filtresinin $(M+1) \times 1$ katsayı vektörü:

$$\underline{a}_M(n) = \begin{bmatrix} 1 \\ -\underline{w}(n) \end{bmatrix} \quad \text{dir.} \quad (3.35)$$

$(M+1) \times 1$ boyutlu $\underline{u}_{M+1}(i)$ vektörü de şöyle tanımlanabilir:

$$\underline{u}_{M+1}(i) = \begin{bmatrix} u(i) \\ \underline{u}_M(i-1) \end{bmatrix} \quad (3.36)$$

Böylece ardıl ileri kestirim hatası yeniden yazılabilir:

$$f_M(i) = \underline{a}^H(n) \underline{u}_{M+1}(i) \quad (3.37)$$

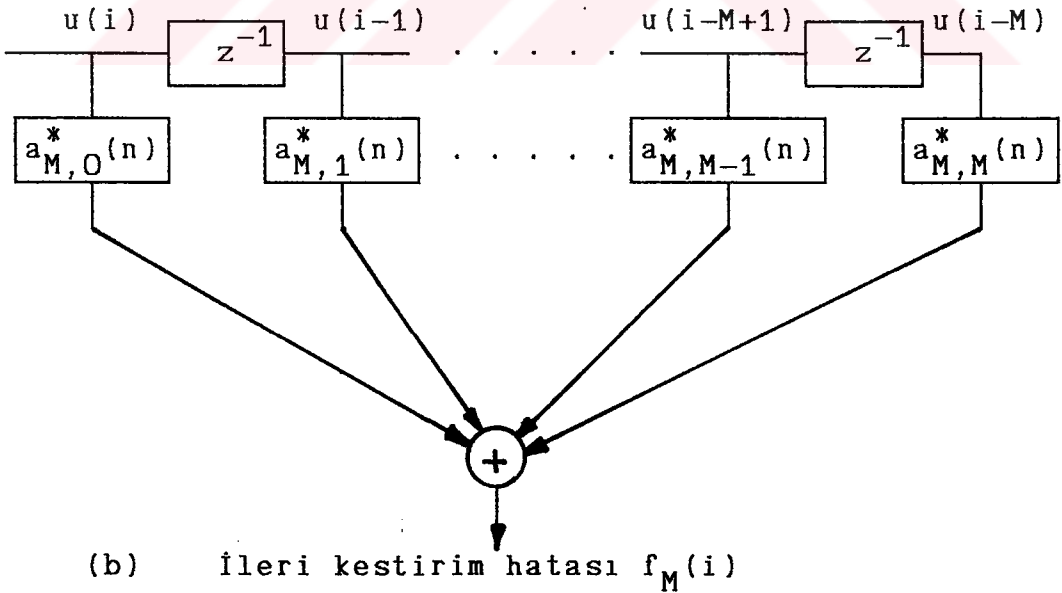
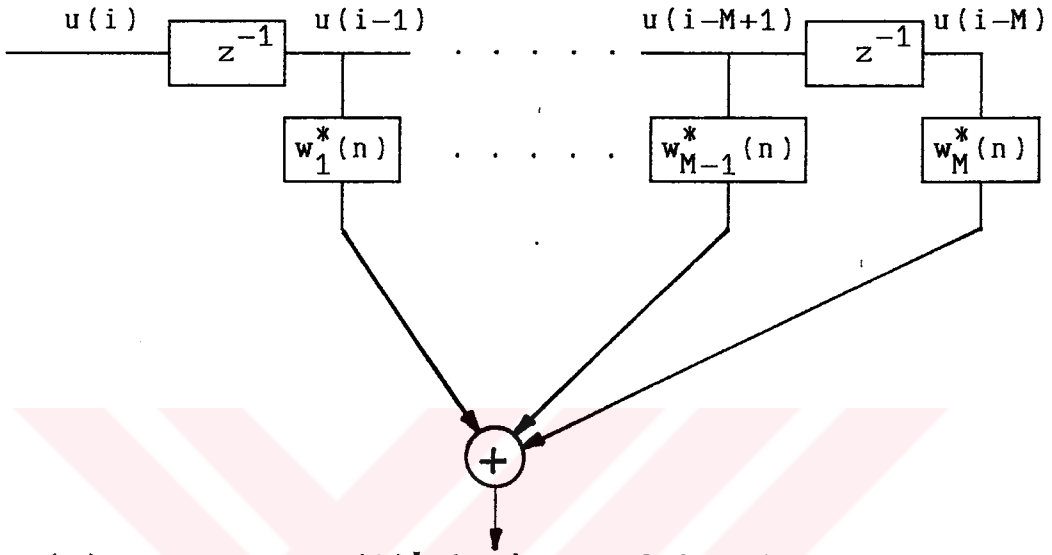
(3.15) ifadesindeki ileri doğrusal kestirim için genişletilmiş normal denklem

$$C_{M+1}(n) \underline{a}_M(n) = \begin{bmatrix} F_M(n) \\ \underline{0} \end{bmatrix} \quad (3.38)$$

Biçiminde yazılır. Burada, $F_M(n)$ ardıl ileri kestirim hata toplamalarının ağırlıklı ortalamasının en küçük değeridir ve şöyle ifade edilir:

$$F_M(n) = \sum_{i=1}^n \lambda^{n-1} |f_M(i)|^2 \quad (3.39)$$

Burada, λ üstel ağırlık faktörüdür.



Şekil 3-3 (a) M dereceli ileri doğrusal kestirici. (b) Karşılık gelen kestirim hata filtresi.

3.4 Ardıl (a posteriori) Geri Kestirim Hatası

Şekil 3-4.a'da görülen M. dereceden geri doğrusal kestiricinin, $\underline{w}(n)$ katsayı vektörü en küçük kare yöntemine göre $1 \leq i \leq n$ aralığında gözlenerek optimize edilmiş olsun. i anındaki geri kestirim hatası şöyle yazılabilir:

$$b_M(i) = u(i-M) - \underline{u}_M^H(i) g(n) \quad (3.40)$$

Burada $u(i)$ arzu edilen cevap ve

$$\underline{u}_M^T(i) = [u(i), u(i-1), \dots, u(i-M+1)] \quad (3.41)$$

giriş vektörüdür.

Şekil 3-4.b de görülen geri kestirim hata filtresinin $(M+1) \times 1$ katsayı vektörü:

$$\underline{c}_M(n) = \begin{bmatrix} -g(n) \\ 1 \end{bmatrix} \quad \text{dir.} \quad (3.42)$$

$(M+1) \times 1$ boyutlu $\underline{u}_{M+1}(i)$ vektörü de şöyle tanımlanabilir:

$$\underline{u}_{M+1}(i) = \begin{bmatrix} \underline{u}_M(i) \\ u_M(i-M) \end{bmatrix} \quad (3.43)$$

Böylece ardıl geri kestirim hatası yeniden yazılabilir:

$$b_M(i) = \underline{c}_M^H(n) \underline{u}_{M+1}(i) \quad (3.44)$$

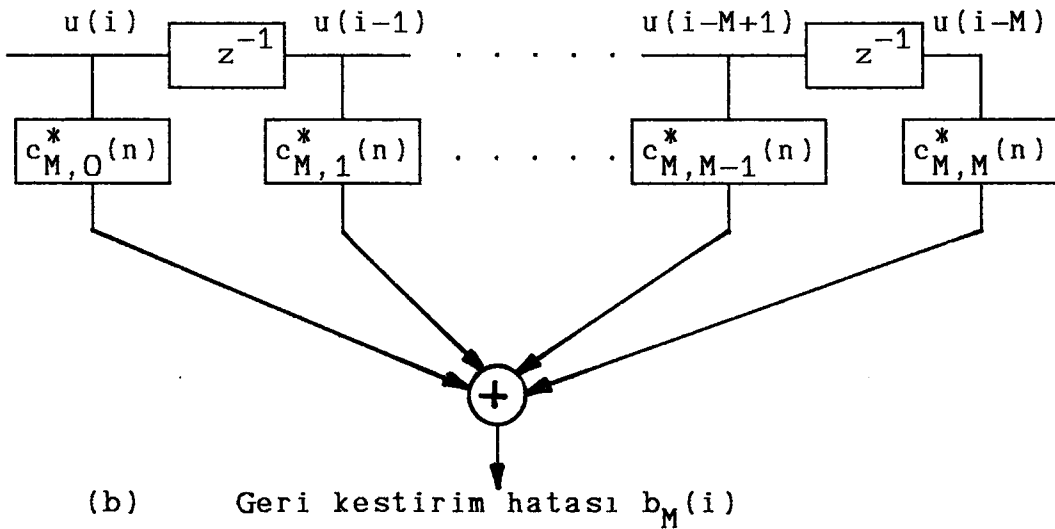
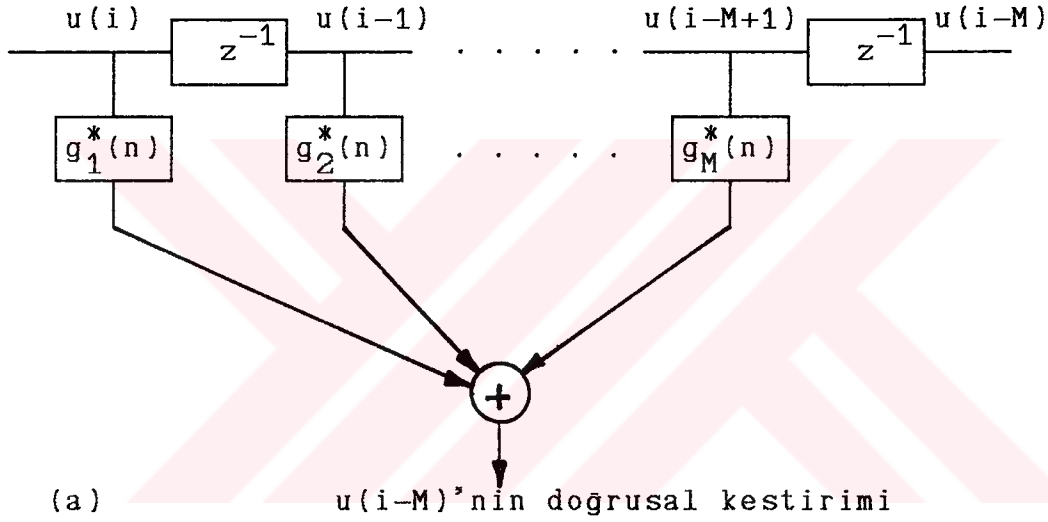
(3.32) ifadesindeki geri doğrusal kestirim için genişletilmiş normal denklem

$$\mathbf{c}_{M+1}(n) = \begin{bmatrix} 0 \\ \mathbf{B}_M(n) \end{bmatrix} \quad (3.45)$$

Bu biçimde yazılır. Burada, $\mathbf{B}_M(n)$ ardıl geri kestirim hata toplamlarının ağırlıklı ortalamasının en küçük değeridir ve şöyle ifade edilir:

$$\mathbf{B}_M(n) = \sum_{i=1}^n \lambda^{n-1} |b_M(i)|^2 \quad (3.46)$$

Burada, λ üstel ağırlık faktörüdür.



Şekil 3-4 (a) M dereceli geri doğrusal kestirici. (b) Karşılık gelen kestirim hata filtresi.

BÖLÜM 4. REKÜRSİF EN KÜÇÜK KARE KAFES FİLTRE ALGORİTMASI

Şimdiye kadar olan bölümlerde temel olarak en küçük kare yöntemi ve en küçük kare kafes filtre algoritmasına temel teşkil eden hususlar incelendi. Bundan sonra bu algoritmanın teşkil edilmesine çalışılacaktır.

4.1 Derece Güncelleme Rekürsifonları

$C_{m+1}(n)$, m . dereceden ileri kestirim hata filtresine uygulanan $\underline{u}_{m+1}(i)$ giriş vektörünün $(m+1) \times (m+1)$ boyutlu ilişki matrisidir. Burada, $1 \leq i \leq n$ dir. Bu filtre aşağıdaki genişletilmiş normal denklemle karakterize edilebilir [bkz. (3.38)]:

$$C_{m+1}(n) \underline{a}_m(n) = \begin{bmatrix} F_m(n) \\ \underline{0}_m \end{bmatrix} \quad (4.1)$$

İlişki fonksiyonunu şu şekilde parçalara ayırarak yazmak mümkündür [bkz. (3.29)]:

$$C_{m+1}(n) = \begin{bmatrix} C_m(n) & \underline{x}_b(n) \\ \underline{x}_b^H(n) & U(n-m) \end{bmatrix} \quad (4.2)$$

Matrisi oluşturan terimler ayrı ayrı yazılacak olursa: Geri kestirim arzulanan cevaplarının karesel ağırlıklı toplamı $U(n-m)$:

$$U(n-m) = \sum_{i=1}^n \lambda^{n-i} |u(i-m)|^2$$

$$= \sum_{i=1}^{n-m} \lambda^{(n-m)-i} |u(i)|^2 \quad (4.3)$$

Kestiricinin [bkz. Şekil 3-4.a] girişlerinin $m \times 1$ çapraz ilişki fonksiyonu $x_n(n)$:

$$x_b(n) = \sum_{i=1}^n \lambda^{n-i} u(i) u^*(i-m) \quad (4.4)$$

Burada $u(i-m)$ arzulanan cevaptır.

Giriş vektörü $u_m(i)$ nin $m \times m$ ilişki matrisi $C_m(n)$:

$$C_m(n) = \sum_{i=1}^n \lambda^{n-i} u_m(i) u_m^H(i) \quad (4.5)$$

(4.2) deki eşitliğin her iki tarafı, ilk m elemanını $a_{m-1}(n)$ nin tanımladığı ve son elemanının sıfır olduğu $(m+1) \times 1$ boyutlu bir vektörle çarpılırsa:

$$\begin{aligned} C_{m+1}(n) \begin{bmatrix} a_{m-1}(n) \\ 0 \end{bmatrix} &= \begin{bmatrix} C_m(n) & x_b(n) \\ x_b^H(n) & U(n-m) \end{bmatrix} \begin{bmatrix} a_{m-1}(n) \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} C_m(n) a_{m-1}(n) \\ x_b^H(n) a_{m-1}(n) \end{bmatrix} \end{aligned} \quad (4.6)$$

(4.1) denkleminde m yerine $m-1$ koyularak şu denklem elde edilir:

$$C_m(n) a_{m-1}(n) = \begin{bmatrix} F_{m-1}(n) \\ 0_{m-1} \end{bmatrix} \quad (4.7)$$

$$K_{m-1}(n) = x_b^H(n) a_{m-1}(n) \quad (4.8)$$

Tanımı yapılırsa ve (4.7) ile (4.8), (4.6) eşitliğinde

yerine konursa:

$$C_{m+1}(n) \begin{bmatrix} a_{m-1}(n) \\ 0 \end{bmatrix} = \begin{bmatrix} F_{m-1}(n) \\ O_{m-1} \\ K_{m-1}(n) \end{bmatrix} \quad (4.9)$$

elde edilir.

m dereceli bir geri kestirim hata filresi aşağıdaki gibi bir genişletilmiş normal denklemle ifade edilebilir [bkz. (3.45)]:

$$C_{m+1}(n) \underline{c}_m(n) = \begin{bmatrix} O_m \\ B_m(n) \end{bmatrix} \quad (4.10)$$

İlişki fonksiyonunu şu şekilde parçalara ayırarak yazmak mümkündür [bkz. (3.12)]:

$$C_{m+1}(n) = \begin{bmatrix} U(n) & x_f^H(n) \\ x_f(n) & C_m(n-1) \end{bmatrix} \quad (4.11)$$

Matrisi oluşturan terimler ayrı ayrı yazılacak olursa: İleri kestirimin arzulanan cevaplarının karelerini ağırlıklı toplamı $U(n)$:

$$U(n-m) = \sum_{i=1}^n \lambda^{n-i} |u(i)|^2 \quad (4.12)$$

Kestiricinin [bkz. Şekil 3-3.a] girişlerinin $m \times 1$ çapraz ilişki fonksiyonu $x_f(n)$:

$$x_f(n) = \sum_{i=1}^n \lambda^{n-1} \underline{u}_m(i-1) \underline{u}^*(i) \quad (4.13)$$

Burada $u(i)$ arzulanan cevaptır.

Giriş vektörü $\underline{u}_m(i-1)$ nin $m \times m$ ilişki matrisi $C_m(n-1)$:

$$C_m(n-1) = \sum_{i=1}^n k^{n-i} \underline{u}_m(i-1) \underline{u}_m^H(i-1)$$

$$= \sum_{i=1}^n \lambda^{(n-i)-1} \underline{u}_m(i) \underline{u}_m^H(i) \quad (4.14)$$

(4.11) deki eşitliğin her iki tarafı, ilk m elemanını $\underline{c}_{m-1}(n-1)$ nin tanımlağı ve son elemanınının sıfır olduğu $(m+1) \times 1$ boyutlu bir vektörle çarpılırsa:

$$\begin{aligned} C_{m+1}(n) \begin{bmatrix} 0 \\ \underline{c}_{m-1}(n-1) \end{bmatrix} &= \begin{bmatrix} U(n) & \underline{x}_f^H(n) \\ \underline{x}_f(n) & C_m(n-1) \end{bmatrix} \begin{bmatrix} 0 \\ \underline{c}_{m-1}(n-1) \end{bmatrix} \\ &= \begin{bmatrix} \underline{x}_f^H(n) C_{m-1}(n-1) \\ C_m(n-1) \underline{c}_{m-1}(n-1) \end{bmatrix} \end{aligned} \quad (4.15)$$

(4.10) denkleminde m yerine $m-1$ koyarak $(n-1)$ anında şu denklem elde edilir:

$$C_m(n-1) \underline{a}_{m-1}(n-1) = \begin{bmatrix} \underline{0}_{m-1} \\ B_{m-1}(n-1) \end{bmatrix} \quad (4.16)$$

$$L_{m-1}(n) = \underline{x}_f^H(n) \underline{c}_{m-1}(n-1) \quad (4.17)$$

Tanımı yapılırsa ve (4.16) ile (4.17), (4.15) eşitliğinde yerine konursa:

$$C_{m+1}(n) \begin{bmatrix} 0 \\ \underline{c}_{m-1}(n-1) \end{bmatrix} = \begin{bmatrix} L_{m-1}(n) \\ \underline{0}_{m-1} \\ B_{m-1}(n-1) \end{bmatrix} \quad (4.18)$$

elde edilir.

(4.9) ve (4.18) eşitliklerinden şu ifade yazılabilir:

$$C_{m+1}(n) \begin{bmatrix} \underline{a}_{m-1}(n) \\ 0 \end{bmatrix} - \frac{K_{m-1}(n)}{B_{m-1}(n-1)} \begin{bmatrix} 0 \\ \underline{c}_{m-1}(n-1) \end{bmatrix} =$$

$$\left[F_{m-1}(n) - \frac{K_{m-1}(n) L_{m-1}(n)}{B_{m-1}(n-1)} \right] \quad (4.19)$$

(4.1) ve (4.19) karşılaştırılırsa şu derece güncelleme rekürsionları elde edilir:

$$\underline{a}_m(n) = \begin{bmatrix} a_{m-1}(n) \\ 0 \end{bmatrix} - \frac{K_{m-1}(n)}{B_{m-1}(n-1)} \begin{bmatrix} 0 \\ \underline{c}_{m-1}(n-1) \end{bmatrix} \quad (4.20)$$

ve

$$F_m(n) = F_{m-1}(n) - \frac{K_{m-1}(n) L_{m-1}(n)}{B_{m-1}(n-1)} \quad (4.21)$$

(4.9) ve (4.18) eşitliklerinden şu ifadeyi de çıkarmak mümkündür:

$$\underline{c}_{m+1}(n) \left[\begin{bmatrix} 0 \\ \underline{c}_{m-1}(n-1) \end{bmatrix} - \frac{L_{m-1}(n)}{F_{m-1}(n)} \begin{bmatrix} a_{m-1}(n) \\ 0 \end{bmatrix} \right] = \left[B_{m-1}(n-1) - \frac{K_{m-1}(n) L_{m-1}(n)}{F_{m-1}(n)} \right] \quad (4.22)$$

(4.10) ve (4.22) karşılaştırılırsa şu derece güncelleme rekürsionları da elde edilir:

$$\underline{c}_m(n) = \begin{bmatrix} 0 \\ \underline{c}_{m-1}(n-1) \end{bmatrix} - \frac{L_{m-1}(n)}{F_{m-1}(n)} \begin{bmatrix} a_{m-1}(n) \\ 0 \end{bmatrix} \quad (4.23)$$

$$B_m(n) = B_{m-1}(n-1) - \frac{K_{m-1}(n) L_{m-1}(n)}{F_{m-1}(n)} \quad (4.24)$$

4.1.1 $K_{m-1}(n)$ ve $L_{m-1}(n)$ Arasındaki İlişki

(4.8) ve (4.17) de tanımlanan $K_{m-1}(n)$ ve $L_{m-1}(n)$ birbirlerinin kompleks eşleniğidir:

$$K_{m-1}(n) = L_{m-1}^*(n) \quad (4.25)$$

Bu iddia üç aşamada ispatlanacaktır:

1. (4.9) eşitliğinin her iki tarafı

$$[0, \underline{c}_{m-1}^H(n-1)]$$

çarpılırsa çıkan sonuç skalerdir:

$$[0, \underline{c}_{m-1}^H(n-1)] C_{m+1}(n) \begin{bmatrix} a_{m-1}(n) \\ 0 \end{bmatrix} = \quad (4.26)$$

$$[0, \underline{c}_{m-1}^H(n-1)] \begin{bmatrix} F_{m-1}(n) \\ \underline{0}_{m-1} \\ K_{m-1}(n) \end{bmatrix} = K_{m-1}(n)$$

2. (4.18) deki eşitliğin her iki tarafına hermitian işlemi (transpoz ve kompleks eşlenik alma) uygulanırsa:

$$[0, \underline{c}_{m-1}^H(n-1)] C_{m+1}(n) = [L_{m-1}^*(n), \underline{0}_{m-1}^T, B_{m-1}(n-1)]$$

Burada $C_{m+1}(n)$ 'nin hermitianı yine kendisine eşit ve $B_{m-1}(n-1)$ skalerdir. Üstteki eşitliğin her iki tarafı

$$\begin{bmatrix} a_{m-1}(n) \\ 0 \end{bmatrix} \quad \text{ile çarpılırsa:}$$

$$[0, \underline{c}_{m-1}^H(n-1)] C_{m+1}(n) \begin{bmatrix} a_{m-1}(n) \\ 0 \end{bmatrix} = \quad (4.27)$$

$$[L_{m-1}^*(n), \underline{0}_{m-1}^T, B_{m-1}(n-1)] \begin{bmatrix} a_{m-1}(n) \\ 0 \end{bmatrix} = L_{m-1}^*(n)$$

3. (4.26) ve (4.27) eşitlikleri karşılaştırılırsa ispatın yapılmış olduğu görülür.

4.1.2 Ardıl Kestirim Hataları İçin Derece Güncelleme Rekürsiyonları

İleri kestirim yansımaya katsayısı aşağıdaki gibi tanımlanır:

$$\Gamma_{f,m}(n) = - \frac{K_{m-1}(n)}{B_{m-1}(n-1)}, \quad m = 1, 2, \dots, M \quad (4.28)$$

Burada M kafes kestiricinin sonuç derecesidir. Geri kestirim yansımaya katsayısı da benzer şekilde aşağıdaki gibi tanımlanır:

$$\begin{aligned} \Gamma_{b,m}(n) &= - \frac{L_{m-1}(n)}{F_{m-1}(n)} \\ &= - \frac{K_{m-1}^*(n)}{F_{m-1}(n)}, \quad m = 1, 2, \dots, M \end{aligned} \quad (4.29)$$

Bu tanımlar kullanılarak (4.20) ve (4.23) ifadeleri yeniden yazılabilir:

$$\underline{a}_m(n) = \begin{bmatrix} a_{m-1}(n) \\ 0 \end{bmatrix} + \Gamma_{f,m}(n) \begin{bmatrix} 0 \\ \underline{c}_{m-1}(n-1) \end{bmatrix} \quad (4.30)$$

ve

$$\underline{c}_m(n) = \begin{bmatrix} 0 \\ \underline{c}_{m-1}(n-1) \end{bmatrix} + \Gamma_{b,m}(n) \begin{bmatrix} a_{m-1}(n) \\ 0 \end{bmatrix} \quad (4.31)$$

Bu iki rekürsif denklem Levinson-Durbin rekürsiyonunun deterministik olanıdır [1].

Ardıl ileri kestirim hatası $f_m(n)$, m . dereceden giriş vektörü $\underline{u}_{m+1}(n)$ olan ileri kestirim hata filtresi- nin çıkışına eşittir:

$$f_m(n) = \underline{a}_m^H(n) \underline{u}_{m+1}(n) \quad (4.32)$$

Giriş vektörü $\underline{u}_{m+1}(n)$ şu şekilde parçalara ayrılarak yazılabilir:

$$\underline{u}_{m+1}(n) = \begin{bmatrix} \underline{u}_m(n) \\ u(n-m) \end{bmatrix} \quad (4.33)$$

Bunu şu şekilde de ifade etmek mümkündür:

$$\underline{u}_{m+1}(n) = \begin{bmatrix} u(n) \\ \underline{u}_m(n-1) \end{bmatrix} \quad (4.34)$$

Ardıl geri kestirim hatası $b_m(n)$, m . dereceden giriş vektörü $\underline{u}_{m+1}(n)$ olan geri kestirim hata filtresi- nin çıkışına eşittir:

$$b_m(n) = \underline{c}_m^H(n) \underline{u}_{m+1}(n) \quad (4.35)$$

Şimdi ardıl ileri kestirim hatasının derece güncelleme rekürsiyonu çıkarılamaya çalışılacaktır. Bunun için önce (9.28) eşitiğinin her iki tarafı hermitian işlemine tabi tutulur ve $\underline{u}_{m+1}(n)$ ile çarpılır:

$$\begin{aligned} \underline{a}_m^H(n) \underline{u}_{m+1}(n) &= [\underline{a}_{m-1}^H(n), 0] \underline{u}_{m+1}(n) + \\ &\Gamma_{f,m}^*(n) [0, \underline{c}_{m-1}^H(n-1)] \underline{u}_{m+1}(n) \end{aligned} \quad (4.36)$$

(4.32)'den görüldüğü üzere eşitliğin sol tarafı $f_m(n)$ 'ye eşittir. Sol taraftaki ilk $u_{m+1}(n)$ 'nin yerine (4.33)'teki ifade, ikinci $u_{m+1}(n)$ 'nin yerine de (4.34)'teki ifade konursa (4.36) eşitiği aşağıdaki hale gelir:

$$f_m(n) = [a_{m-1}^H(n) , 0] \begin{bmatrix} u_m(n) \\ u(n-m) \end{bmatrix} + \Gamma_{f,m}^* [0 , c_{m-1}^H(n-1)] \begin{bmatrix} u(n) \\ u_m(n-1) \end{bmatrix}$$

Çarpma işlemleri yapılırsa:

$$f_m(n) = a_{m-1}^H(n) u_m(n) + \Gamma_{f,m}^* c_{m-1}^H(n-1) u_m(n-1) \quad (4.37)$$

(4.32)'den şu ifade:

$$f_{m-1}(n) = a_{m-1}^H(n) u_m(n) , \quad (4.38)$$

(4.35)'den de şu ifade yazılır:

$$b_{m-1}(n-1) = c_{m-1}^H(n-1) u_m(n-1) \quad (4.39)$$

(4.38) ve (4.39) ifadeleri (4.37)'de yerine yazılırsa ardıl ileri kestirim hatasının derece güncelleme rekürsasyonu elde edilmiş olur:

$$f_m(n) = f_{m-1}(n) + \Gamma_{f,m}^* b_{m-1}(n-1) \quad (4.40)$$

(4.31), (4.32), (4.33), (4.34), (4.35), (4.38) ve (4.39) kullanılarak benzer işlemler yapılırsa ardıl geri kestirim derece güncelleme rekürsasyonu elde edilir:

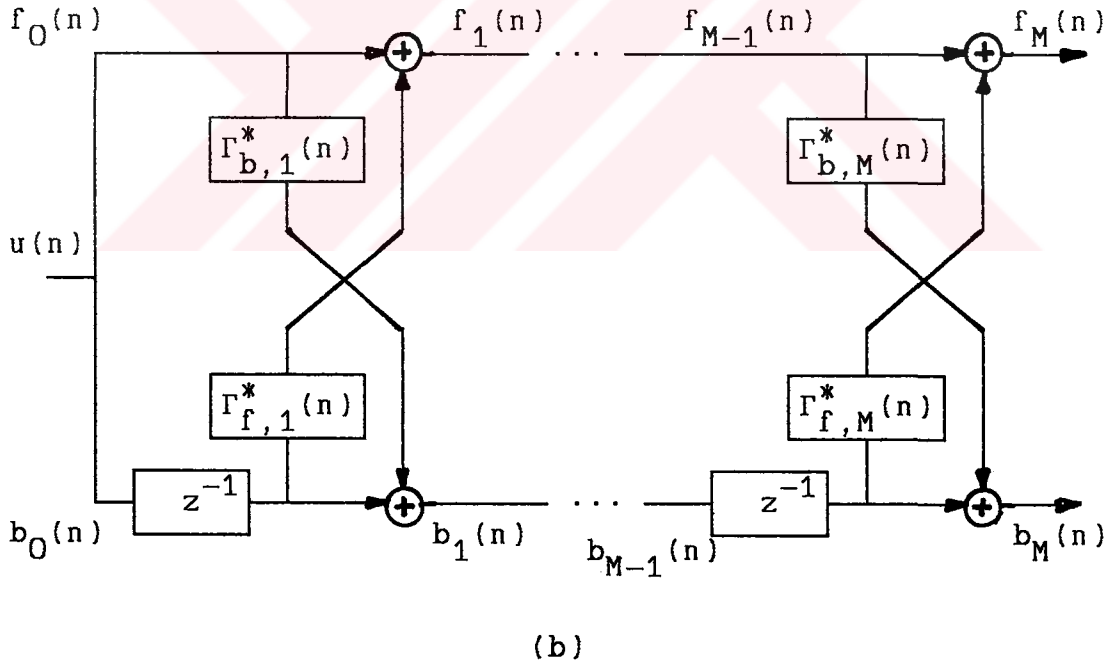
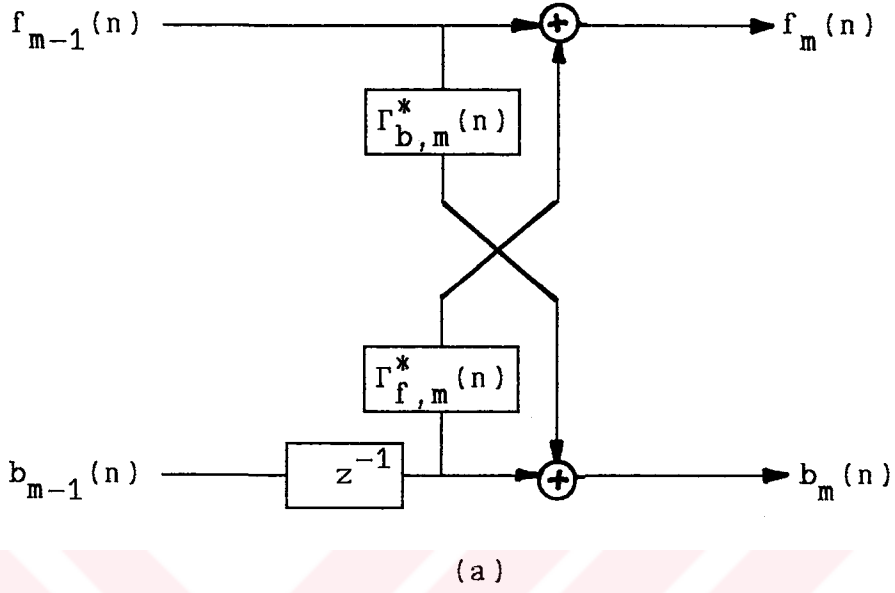
$$b_m(n) = f_{m-1}(n-1) + \Gamma_{b,m}^* f_{m-1}(n) \quad (4.41)$$

(3.35), (3.42), Şekil 3-3, Şekil 3-4'den görüldüğü gibi $m=0$ olduğu zaman ($m=0,1,\dots,M$):

$$f_0(n) = b_0(n) = u(n) \quad \text{dir.}$$

Şekil 4-1'de (4.40) ve (4.41) rekürsionlarının işaret akış grafiği çizilmiştir ki; bu da kafes filtre yapısını

oluşturmaktadır.



Şekil 4-1 (a) Tipik bir en küçük kare kafes modülü.
(b) Çok modüllü en küçük kare kafes kestirici.

4.1.3 Ardıl Kestirim Hataları Ağırlıklı Toplamlarının Derece Güncelleme Rekürsiyonları

(4.21) ve (4.24) sırasıyla ardıl ileri kestirim hatalarının ağırlıklı toplamı $F_m(n)$ 'nin ve ardıl geri kestirim hatalarının ağırlıklı toplamı $B_m(n)$ 'nin derece güncelleme rekürsiyonlarını gösterir. (4.25)'de verilen özellikten yararlanılarak bu rekürsiyonlar düzenlenerek yeniden yazılır:

$$F_m(n) = F_{m-1}(n) - \frac{|K_{m-1}(n)|^2}{B_{m-1}(n-1)} \quad (4.42)$$

$$B_m(n) = B_{m-1}(n-1) - \frac{|K_{m-1}(n)|^2}{F_{m-1}(n)} \quad (4.43)$$

4.1.4 Dönüşüm Faktörü İçin Derece Güncelleme Rekürsiyonu

En küçük kare kafes filte (EKK) algoritması - Least-squares lattice (LSL) - için gerekli bir kavram olan dönüşüm faktörü (veya, gecikmiş kestirim hata faktörü) $\gamma_m(n-1)$ 'in derece güncelleme rekürsiyonu çıkarılacaktır. Bu faktör 4.2 bölümünde kullanılacaktır.

Kazanç vektörü $\underline{k}_m(n)$ şöyle tanımlanır:

$$\underline{k}_m(n) = C_m^{-1}(n) \underline{u}(n) \quad (4.44)$$

Bu tanımdan da şu ifade yazılabilir:

$$\underline{k}_m(n-1) = C_m^{-1}(n-1) \underline{u}(n-1) \quad (4.45)$$

(4.45) ifadesi ve ilişkişki matrisi ile ilgili diğer bağıntılar kullanılarak aşağıdaki rekürsif ifade yazılabilir (Ayrıntılı matematiksel bilgi için [1]'e bakınız):

$$\underline{k}_m(n-1) = \begin{bmatrix} \underline{k}_{m-1}(n-1) \\ 0 \end{bmatrix} + \frac{b_{m-1}(n-1)}{B_{m-1}(n-1)} \underline{c}_{m-1}(n-1) \quad (4.46)$$

Bu ifadeye hermitian işlemi uygulanıp her iki tarafı $\underline{u}_m(n-1)$ ile çarpılırsa:

$$\underline{k}_m^H(n-1) \underline{u}_m(n-1) = [\underline{k}_{m-1}^H(n-1), 0] \underline{u}_m(n-1) + \frac{b_{m-1}^*(n-1)}{B_{m-1}(n-1)} \underline{c}_{m-1}^H(n-1) \underline{u}_m(n-1) \quad (4.47)$$

$$\underline{k}_m^H(n-1) \underline{u}_m(n-1) = 1 - \gamma_m(n-1) \quad (4.48)$$

Tanımı yapılır, (3.44) ifadesinden de:

$$\underline{c}_{m-1}^H(n-1) \underline{u}_m(n-1) = b_{m-1}(n-1) \quad (4.49)$$

yazılırsa, (4.48) ve (4.49), (4.47)'de yerine konarak şu ifade elde edilir:

$$\gamma_m(n-1) = \gamma_{m-1}(n-1) - \frac{|b_{m-1}(n-1)|^2}{B_{m-1}(n-1)} \quad (4.50)$$

Bu ifade derece güncelleme rekürsionlarının sonuncusudur.

4.2 Zaman Güncelleme Rekürsionları

Bölüm 4.1'de verilen rekürsionlarda zaman sabit, ifadeler derece olarak bir öncekinden elde ediliyordu. Şimdi ise dereceler sabit, ifadeler zaman olarak bir öncekinden elde edilecek. Bu tür bir yaklaşıma zaman güncelleme adı verilir.

(4.44) ve (4.46)' dan şu ifade yazılabilir:

$$K_{m-1}(n) = [K_{m-1}(n), \underline{O}_{m-1}^T, B_{m-1}(n-1)] \begin{bmatrix} \underline{a}_{m-1}(n-1) \\ 0 \end{bmatrix} \quad (4.51)$$

Her iki tarafa hermitian işlemi uygulanırsa, (4.44) ifadesinden faydalanılarak:

$$[O, \underline{c}_{m-1}^H(n-1)] C_{m+1}(n) = [K_{m-1}(n), \underline{O}_{m-1}^T, B_{m-1}(n-1)] \quad (4.52)$$

yazılabilir. Bu sonuç (4.51)'de kullanılırsa:

$$K_{m-1}(n) = [O, \underline{c}_{m-1}^H(n-1)] C_{m+1}(n) \begin{bmatrix} \underline{a}_{m-1}(n-1) \\ 0 \end{bmatrix} \quad (4.53)$$

Bulunur. (4.14) eşitliğinden şu ifade yazılabilir:

$$C_{m+1}(n) = \lambda \left[\sum_{i=1}^{n-1} \lambda^{n-1-i} \underline{u}_{m+1}(i) \underline{u}_{m+1}^H(i) \right] + \underline{u}_{m+1}(n) \underline{u}_{m+1}^H(n)$$

Bu ifaden de:

$$C_{m+1}(n) = \lambda C_{m+1}(n-1) + \underline{u}_{m+1}(n) \underline{u}_{m+1}^H(n) \quad (4.54)$$

rekürsiyonu bulunur. Bu rekürsiyon kullanılarak (4.53) yeniden yazılırsa:

$$K_{m-1}(n) = \lambda [O, \underline{c}_{m-1}^H(n-1)] C_{m+1}(n-1) \begin{bmatrix} \underline{a}_{m-1}(n-1) \\ 0 \end{bmatrix} + [O, \underline{c}_{m-1}^H(n-1)] \underline{u}_{m+1}(n) \underline{u}_{m+1}^H(n) \begin{bmatrix} \underline{a}_{m-1}(n-1) \\ 0 \end{bmatrix} \quad (4.55)$$

elde edilir. Öncül (a priori) ileri kestirim hatası tanımı aşağıdaki gibi yapılırsa:

$$\eta_m(n) = \underline{u}_{m+1}^T(n) \underline{a}_m^*(n-1) \quad (4.56)$$

(4.55) ifadesinde görülen

$$\begin{aligned} \underline{u}_{m+1}^H(n) \begin{bmatrix} \underline{a}_{m-1}(n-1) \\ 0 \end{bmatrix} &= [\underline{u}_{m+1}(n), \underline{u}^*(n-m)] \begin{bmatrix} \underline{a}_{m-1}(n-1) \\ 0 \end{bmatrix} \\ &= \underline{u}_m^H(n) \underline{a}_{m-1}(n-1) = \eta_{m-1}^*(n) \end{aligned} \quad (4.57)$$

olarak elde edilir.

Ardıl geri kestirim hatası tanımından:

$$\begin{aligned} [0, \underline{c}_{m-1}^H(n-1)] \underline{u}_{m+1}(n) &= [0, \underline{c}_{m-1}^H(n-1)] \begin{bmatrix} u(n) \\ \underline{u}_m(n-1) \end{bmatrix} \\ &= \underline{c}_{m-1}^H(n-1) \underline{u}_m(n-1) \\ &= b_{m-1}(n-1) \end{aligned} \quad (4.58)$$

yazılır.

(4.9) eşitliğinde n yerine n-1 konularak şu ifade elde edilir:

$$\underline{c}_{m+1}(n-1) \begin{bmatrix} \underline{a}_{m-1}(n-1) \\ 0 \end{bmatrix} = \begin{bmatrix} F_{m-1}(n-1) \\ \underline{0}_{m-1} \\ K_{m-1}(n-1) \end{bmatrix} \quad (4.59)$$

Bu ifadeyi kullanarak (4.55) ifadesinin sağ tarafını, λ hariç şu şekilde yazmak mümkündür:

$$\begin{aligned} [0, \underline{c}_{m-1}^H(n-1)] \underline{c}_{m+1}(n-1) \begin{bmatrix} \underline{a}_{m-1}(n-1) \\ 0 \end{bmatrix} &= \\ [0, \underline{c}_{m-1}^H(n-1)] \begin{bmatrix} F_{m-1}(n-1) \\ \underline{0}_{m-1} \\ K_{m-1}(n-1) \end{bmatrix} &= K_{m-1}(n-1) \end{aligned} \quad (4.60)$$

(4.57), (4.58) ve (4.60) ifadeleri (4.55)'de yerlerine konursa, $K_{m-1}(n)$ için zamam güncelleme rekürsyonu elde edilir:

$$K_{m-1}(n) = \lambda K_{m-1}(n-1) + b_{m-1}(n-1) \eta_{m-1}^*(n) \quad (4.61)$$

(4.44), (4.48) ve (4.56)'dan şu ilişki yazılırsa:

$$\eta_{m-1}(n) = \frac{f_{m-1}(n)}{\gamma_{m-1}(n-1)} \quad (4.62)$$

(4.61)'daki rekürsion şu şekilde yazılır:

$$K_{m-1}(n) = \lambda K_{m-1}(n-1) + \frac{b_{m-1}(n-1)f_{m-1}^*(n)}{\gamma_{m-1}(n-1)} \quad (4.63)$$

4.3 Rekürsif EKK Algoritmasının Özeti

Bölüm 4.1 ve 4.2'de çıkarılan derece ve zaman güncelleme rekürsionları EKK algoritmasını teşkil etmektedir. Bu rekürsionlar Tablo 4-1'de verilmiştir.

Bu algoritma ileri ve geri yönlerde doğrusal kestirim sağlar. En önemli özelliklerinden birisi modüller olmasıdır. Filtre derecesi modüllerin sayısını belirler. Derecenin ne olduğunun önemi olmaksızın bir önceki derece için bulunan değerler kullanılarak kestiricinin derecesi arttırılabilir [5].

Tablo 4-1 Rekürsif EKK algoritmasının özeti

$n=1..N$, $m=1..M$ (N : örnek sayısı, M filtre derecesi)

$$K_{m-1}(n) = \lambda K_{m-1}(n-1) + \frac{b_{m-1}(n-1) f_{m-1}^*(n)}{\gamma_{m-1}(n-1)}$$

$$\Gamma_{f,m}(n) = - \frac{K_{m-1}(n)}{B_{m-1}(n-1)}$$

$$\Gamma_{b,m}(n) = - \frac{K_{m-1}^*(n)}{F_{m-1}(n)}$$

$$f_m(n) = f_{m-1}(n) + \Gamma_{f,m}^*(n) b_{m-1}(n-1)$$

$$b_m(n) = b_{m-1}(n-1) + \Gamma_{b,m}^*(n) f_{m-1}(n)$$

$$F_m(n) = F_{m-1}(n) - \frac{|K_{m-1}(n)|^2}{B_{m-1}(n-1)}$$

$$B_m(n) = B_{m-1}(n-1) - \frac{|K_{m-1}(n)|^2}{F_{m-1}(n)}$$

$$\gamma_m(n-1) = \gamma_{m-1}(n-1) - \frac{|b_{m-1}(n-1)|^2}{B_{m-1}(n-1)}$$

Birleşik süreç kestirimi (Bkz. 4.5) için:

$n=1..N, m=0..M$

$$\rho_m(n) = \lambda \rho_m(n-1) + \frac{b_m(n)}{\gamma_m(n)} e_m^*(n)$$

$$\varphi_m(n) = \frac{\rho_m(n)}{B_m(n)}$$

$$e_{m+1}(n) = e_m(n) - \varphi_m^*(n) b_m(n)$$

4.4 Rekürsif EKK Algoritmasının Önkoşullandırması

Rekürsif EKK algoritmasının önkoşullandırmasında ilk önce, $m=0$ için ileri ve geri doğrusal kestirim hatalarının ilk değerleri (4.32) ve (4.35)'den

yararlanılarak yazılır:

$$f_0(n) = b_0(n) = u(n)$$

(3.39), (4.32) ve (4.40)'dan yararlanılarak aşağıdaki ardıl ileri kestirim hataları ağırlıklı toplamının zaman güncelleme rekürsiyonu yazılabilir (ayrıntılı matematiksel bilgi için bkz. [1], [6]):

$$F_m(n) = \lambda F_m(n-1) + f_m(n) \eta_m^*(n) \quad (4.64)$$

Burada, $f_m(n)$ ve $\eta_m(n)$ m. dereceden ardıl ve öncül ileri kestirim hatalarıdır. $m=0$ için:

$$f_0(n) = \eta_0(n) = u(n)$$

yazılır. Böylece, $m=0$ için $m=0$ için (4.64)'deki zaman güncelleme rekürsiyonu yazılabilir:

$$F_0(n) = \lambda F_0(n-1) + |u(n)|^2 \quad (4.65)$$

Benzer şekilde ardıl geri kestirim hataları ağırlıklı toplamı zaman güncelleme rekürsiyonu $m=0$ için aşağıdaki gibi yazılabilir:

$$B_0(n) = \lambda B_0(n-1) + |u(n)|^2 \quad (4.66)$$

Kestirim hata faktörü $\gamma_m(n-1)$ gerçekte sayıdır ve değeri 0 ile 1 arasında değişir [7]. Bu durumda, $m=0$ için bu faktöre 1 değerini vermek mantıklıdır:

$$\gamma_0(n-1) = 1 \quad (4.67)$$

$n=0$ için aşağıdaki önkoşullandırmalar yapılırsa rekürsif EKK algoritmasının önkoşullandırması tamamlanmış olur:

$$K_{m-1}(0) = 0, \quad (4.68)$$

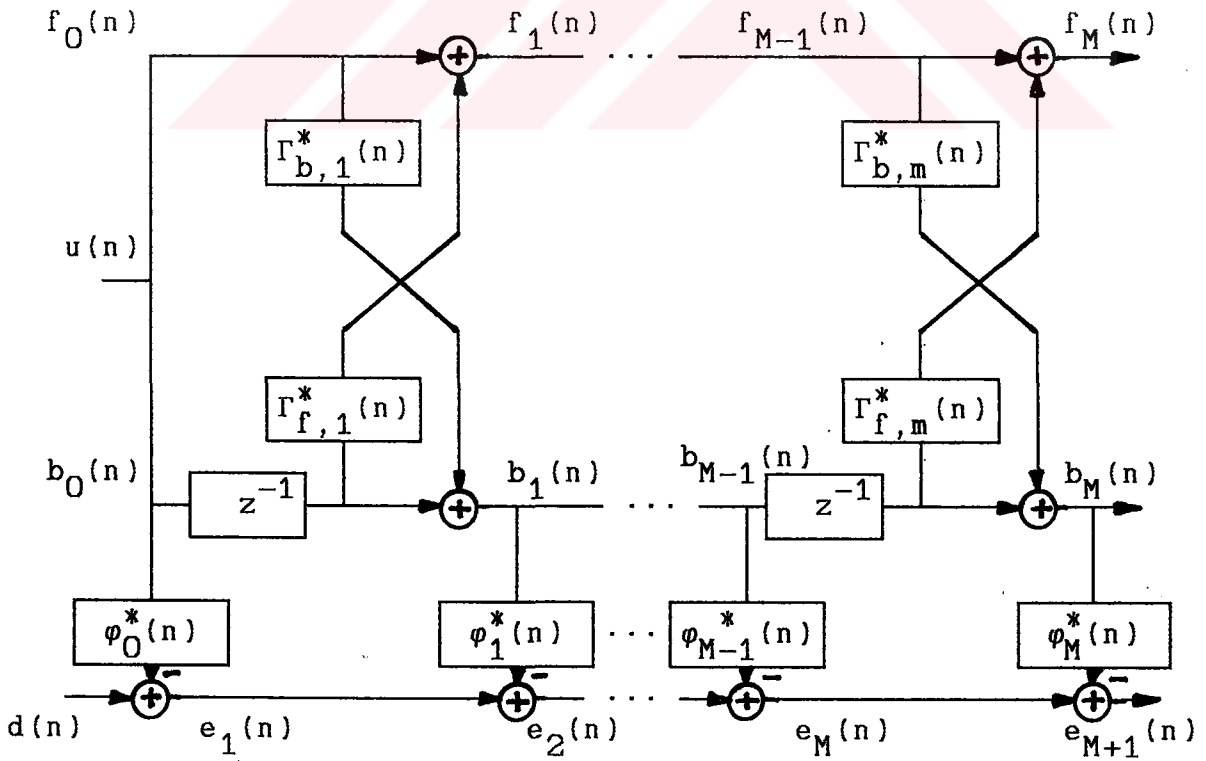
$$F_0(0) = B_0(0) = \delta \quad (4.69)$$

(4.47)'de görünen δ , küçük, pozitif bir sayıdır. Rekürsiflikte sifıra bölme olmaması için böyle seçilmiştir.

Durağan ortamlarda, üstel ağırlık faktörü λ için 1 değerini vermek en iyi sonucu verir, oysa durağan olmayan ortamlarda, $\lambda < 1$ uygun seçimdir [8].

4.5 Birleşik Süreç Kestirimi

Arzu edilen bir işaretin elde edilebilmesi için geri kestirim hatalarının doğrusal regresyonu sonucu elde edilen ardıl kestirim hatalarından yaralanan birleşik süreç kestiricisi kullanılmaktadır. Bu kestiricinin yapısı şekil 4-2'de gösterilmiştir. Bu kestiricide, $\{ d(n) \}$ arzulanan süreç, $\{ u(n) \}$ de gözlem sürecidir. Her iki süreç de kestiriciye diğer bir deyişle filtreye giriş olarak geldikleri için $\{ d(n), u(n) \}$ 'ne birleşik süreç, kestiriciye de birleşik süreç kestiricisi adı verilir [9]. Regresyon katsayıları ise $\varphi_m(n)$ ile gösterilir.



Şekil 4-2 Birleşik süreç kestiricisi.

m . dereceden bir geri kestirim hata filtresi göz önünde bulundurulsun. Bu filtrenin $(m+1) \times 1$ boyutlu katsayı vektörü en küçük kare yöntemine göre, $1 \leq i \leq n$ aralığında en uygun şekilde tespit edilmiş olsun. Bu katsayı vektörü genişletilmiş formda şu şekilde yazılır:

$$\underline{c}_m^T(n) = [c_{m,m}(n), c_{m,m-1}(n), \dots, 1] \quad (4.70)$$

$(m+1) \times 1$ boyutlu giriş vektörü $\underline{u}_{m+1}(n)$, genişletilmiş formda aşağıdaki gibi yazılır:

$$\underline{u}_{m+1}^T(i) = [u(i), u(i-1), \dots, u(i-m)], \quad i > m \quad (4.71)$$

$\underline{u}_{m+1}(i)$ girişine karşılık olan geri kestirim hatası $b_m(i)$, de aşağıdaki gibi ifade edilir:

$$\begin{aligned} b_m(i) &= \underline{c}_m^H(n) \underline{u}_{m+1}(i) \\ &= \sum_{k=0}^m c_{m,k}^*(n) u(i-m+k), \quad m < i \leq n, \\ &\quad m=0, 1, 2, \dots \end{aligned} \quad (4.72)$$

$(m+1) \times 1$ geri kestirim hata vektörü aşağıdaki gibi ifade edilsin:

$$\underline{b}_{m+1}^T(i) = [b_0(i), b_1(i), \dots, b_m(i)], \quad i > m, \quad (4.73)$$

$m=0, 1, 2, \dots$

(4.72) ve (4.73) birleştirilirse aşağıdaki dönüşüm yazılır:

$$\underline{b}_{m+1}(i) = \underline{\Lambda}_m(n) \underline{u}_{m+1}(n) \quad (4.74)$$

Burada $(m+1) \times (m+1)$ boyutlu dönüşüm matrisi şu şekilde tanımlanır:

$$\Lambda_m(n) = \begin{bmatrix} 1 & 0 & \dots & 0 \\ c_{1,1}^*(n) & 1 & & 0 \\ \vdots & \vdots & & \vdots \\ c_{m,m}^*(n) & c_{m,m-1}^*(n) & \dots & 1 \end{bmatrix} \quad (4.75)$$

Yukarıdaki matriste 1. satırın sıfır olmayan elemanları, (1-1) dereceli geri kestirim hata filtresinin katsayılarıdır. Ana diagonal üzerindeki elemanlar 1 dir ki, bu da en son dereceli katsayının değeri 1 olduğunu gösterir. Ayrıca, matrisin bütün m'ler için determinantı 1'e eşit olduğu için tersi mevcuttur.

Ardıl ileri kestirim hatasının (m+1)x(m+1) boyutlu deterministik ilişki matrisi aşağıdaki gibi tanımlanır:

$$D_{m+1}(n) = \sum_{i=1}^n \lambda^{n-i} \underline{b}_{m+1}(i) \underline{b}_{m+1}^H(i), \quad m < i \leq n, \quad (4.76)$$

m=0, 1, 2,....

(4.74) ifadesi (4.76)'da yerine yazılırsa aşağıdaki ifade elde edilir:

$$\begin{aligned} D_{m+1}(n) &= \sum_{i=1}^n \lambda^{n-i} \Lambda_m(n) \underline{u}_{m+1}(i) \underline{u}_{m+1}^H(i) \Lambda_m^H(n) \\ &= \Lambda_m(n) \left[\sum_{i=1}^n \lambda^{n-i} \underline{u}_{m+1}(i) \underline{u}_{m+1}^H(i) \right] \Lambda_m^H(n) \end{aligned} \quad (4.77)$$

Köşeli parantez içindeki ifade, giriş vektörünün deterministik ilişki matrisidir:

$$C_{m+1}(n) = \sum_{i=1}^n \lambda^{n-i} \underline{u}_{m+1}(n) \underline{u}_{m+1}^H(i) \quad (4.78)$$

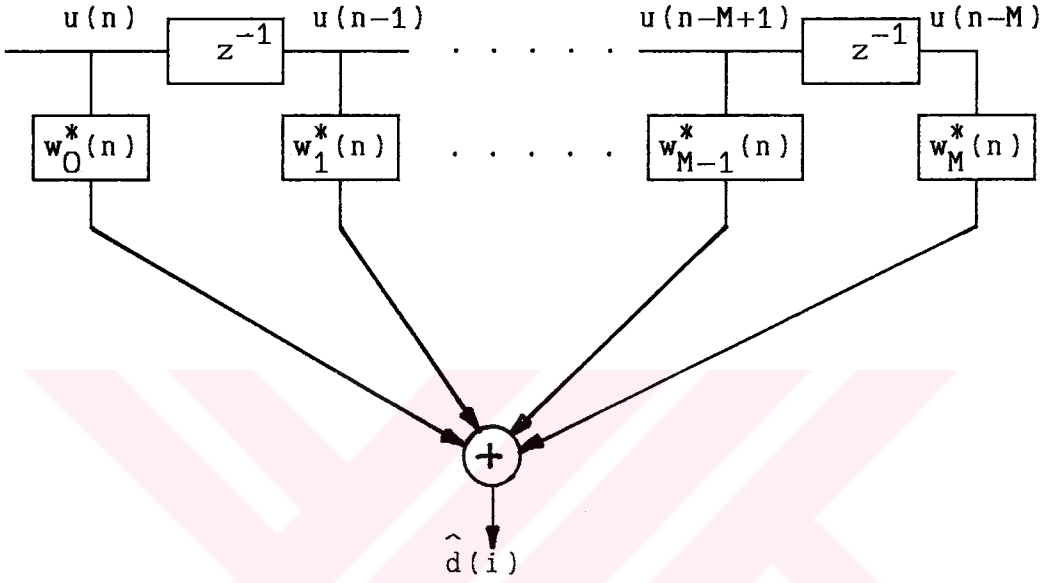
Buna göre, (4.77) ifadesi basitleştirilebilir:

$$D_{m+1}(n) = \Lambda_m(n) C_{m+1}(n) \Lambda_m^H(n) \quad (4.79)$$

(3.32) ifadesindeki geri kestirim genişletilmiş normal denkleminde ve (4.79) ifadesinden:

$$D_{m+1}(n) = \text{diag} [B_0(n), B_1(n) \dots B_m(n)] \quad (4.80)$$

çıkarılır. ("diag" operatörü ana diagonal üzerindeki elemanların parantezin içindekiler, diğerlerinin sıfır olduğunu belirtir.)



Şekil 4-3 Konvansiyonel transversal filtre modeli.

Şekil 4-3'de verilen konvansiyonel transversal filtre göz önünde bulundurulsun. Bu filtenin katsayılarının en küçük kare yöntemine göre aşağıdaki deterministik normal denklem kullanılarak bulunur [bkz. (2.36)]:

$$C_{m+1}(n) \underline{w}_m(n) = \underline{x}_{m+1}(n) \quad (4.81)$$

Burada, $C_{m+1}(n)$, $(m+1) \times (m+1)$ boyutlu ilişki matrisi, $\underline{x}_{m+1}(n)$, $(m+1) \times 1$ boyutlu çapraz ilişki vektörü ve $\underline{w}_m(n)$ de $(m+1) \times 1$ katsayı vektörüdür. Bu denklem üzerine iki işlem yapılabilir: Birincisi, her iki taraf $(m+1) \times (m+1)$ boyutlu dönüşüm matrisi $\Lambda_m(n)$ ile çarpılır; ikincisi, ilişki matrisi ile katsayı matrisi arasına birim matris-e eşit olan $\Lambda_m^H(n) \Lambda_m^{-H}(n)$ sokulur. Λ_m^{-H} , Λ_m^{-1} 'nin hermitian transpozudur. Sonuçta aşağıdaki denklem elde edilir:

$$\Lambda_m(n) C_{m+1}(n) \Lambda_m^H(n) \Lambda_m^{-H}(n) \underline{w}_m(n) = \Lambda_m(n) \underline{x}_{m+1}(n) \quad (4.82)$$

(4.79) ifadesinden $\Lambda_m(n) C_{m+1}(n) \Lambda_m^H(n)$ 'nin $D_{m+1}(n)$ 'ye eşit olduğu görülür. Bu matris, regresyon katsayıları tarafından giriş olarak kullanılan ardıl geri kestirim hatalarının deterministik ilişki matrisidir. (4.82) eşitliğinin sağ tarafındaki $\Lambda_m(n) \underline{x}_{m+1}(n)$ ifadesi ise bu geri kestirim hataları ile arzulanan cevap arasındaki $(m+1) \times 1$ boyutlu çapraz ilişki vektörüdür. $\underline{t}_{m+1}(n)$ bu vektörü temsil etsin. Tanım gereği bu vektör şöyledir:

$$\underline{t}_{m+1}(n) = \sum_{i=1}^n \lambda^{n-i} \underline{b}_{m+1}(i) d^*(i) \quad (4.83)$$

Burada $d(i)$ arzulanan cevabı göstermektedir. (4.74) ifadesi (4.83)'de yerine konursa:

$$\begin{aligned} \underline{t}_{m+1}(n) &= \sum_{i=1}^n \lambda^{n-i} \Lambda_m(n) \underline{u}_{m+1}(i) d^*(i) \\ &= \Lambda_m(n) \sum_{i=1}^n \lambda^{n-i} \underline{u}_{m+1}(i) d^*(i) \quad (4.84) \\ &= \Lambda_m(n) \underline{x}_{m+1}(n) \end{aligned}$$

elde edilir ki, bu da arzu edilen sonuçtur. (4.79) ve (4.84) ifadeleri (4.82)'de kullanılırsa dönüştürülmüş EKK çözümü elde edilir:

$$D_{m+1}(n) \Lambda_m^{-H}(n) \underline{w}_m(n) = \underline{t}_{m+1}(n) \quad (4.85)$$

Şekil 4-2'de görülen regresyon katsayısı vektörü aşağıdaki gibi tanımlanır:

$$\underline{\varphi}_m^T = [\varphi_0(n), \varphi_1(n), \dots, \varphi_m(n)] \quad (4.86)$$

Regresyon katsayısı vektörü $\underline{\varphi}_m(n)$, aşağıdaki ağırlıklı karesel hata ifadesi minimize edilerek elde edilir:

$$\sum_{i=1}^n \lambda^{n-i} |d(i) - \underline{b}_{m+1}^T(i) \underline{\varphi}_m^*(n)|^2$$

$1 \leq i \leq n$ aralığında, $\underline{\varphi}_m(n)$ sabittir. Deterministik normal denklemle bu rekürsif en küçük kare problemi ifade edilebilir:

$$D_{m+1}(n) \underline{\varphi}_m(n) = \underline{t}_{m+1}(n) \quad (4.87)$$

(4.85) ve (4.87) ifadeleri karşılaştırılarak Şekil 4-3'deki katsayı vektörü $\underline{w}_m(n)$ ile Şekil 4-2'de görülen regresyon katsayısı vektörü $\underline{\varphi}_m(n)$ arasındaki ilişki şöyle yazılabilir:

$$\underline{\varphi}_m(n) = \underline{\Lambda}_m^{-H}(n) \underline{w}_m(n) \quad (4.88)$$

veya

$$\underline{w}_m(n) = \underline{\Lambda}_m^H(n) \underline{\varphi}_m(n) \quad (4.89)$$

4.5.1 Regresyon Katsayısı Vektörünün Rekürsif Olarak Elde Edilmesi

(4.84) ve (4.87) eşitlikleri çözülerek regresyon katsayısı vektörü $\underline{\varphi}_m(n)$ aşağıdaki gibi elde edilir:

$$\underline{\varphi}_m(n) = D_{m+1}^{-1}(n) \underline{\Lambda}_m(n) \underline{x}_{m+1}(n) \quad (4.90)$$

(4.80) ifadesinden aşağıdaki sonuç çıkarılabilir:

$$D_{m+1}^{-1}(n) = \text{diag} [B_0^{-1}(n), B_1^{-1}(n), \dots, B_m^{-1}(n)] \quad (4.91)$$

Bu demektir ki, regresyon katsayısı vektörü $\underline{\varphi}_m(n)$ 'yi hesaplamak için sadece $m+1$ skaler bölme yapmak yeterlidir. (4.75) ve (4.91) ifadeleri (4.91)'de yerlerine konursa şu ifade elde edilir:

$$\underline{\varphi}_m(n) = B_m^{-1}(n) \underline{c}_m^H(n) \underline{x}_{m+1}(n) \quad (4.92)$$

Aşağıdaki sabit tanımlanırsa:

$$\rho_m(n) = \underline{c}_m^H(n) \underline{x}_{m+1}(n) \quad (4.93)$$

(4.92) ifadesi şöyle yazılabilir:

$$\varphi_m(n) = \frac{\rho_m(n)}{B_m(n)} \quad (4.94)$$

Geri kestirim hata filtresinin zaman güncelleme rekürsiyo-aşağıdaki gibidir [1]:

$$\underline{c}_m(n) = \underline{c}_m(n-1) - \frac{b_m^*(n)}{\gamma_m(n)} \begin{bmatrix} a_{m-1}(n) \\ 0 \end{bmatrix} \quad (4.95)$$

(4.4) ifadesinden:

$$\underline{x}_{m+1}(n) = \sum_{i=1}^n \lambda^{n-i} \underline{u}_{m+1}(i) d^*(i) \quad (4.96)$$

dir. Bu ifade aşağıdaki hale getirilebilir:

$$\underline{x}_{m+1}(n) = \lambda \left[\sum_{i=1}^{n-1} \lambda^{n-1-i} \underline{u}_{m+1}(i) d^*(i) \right] + \underline{u}_{m+1}(n) d^*(n)$$

Bu ifaden de:

$$\underline{x}_{m+1}(n) = \lambda \underline{x}_{m+1}(n-1) + \underline{u}_m(n) d^*(n) \quad (4.97)$$

elde edilir. Böylece, (4.95) ve (4.97) ifadeleri (4.93) de yerlerine konursa aşağıdaki ifade elde edilir:

$$\begin{aligned} \rho_m(n) &= \lambda \underline{c}_m^H(n-1) \underline{x}_{m+1}(n-1) + \underline{c}_m^H(n-1) \underline{u}_m(n) d^*(n) - \\ &\quad - \frac{b_m}{\gamma_m(n)} \underline{k}_m^H(n) \underline{x}_m(n) \end{aligned} \quad (4.98)$$

(4.93) ifadesinden:

$$\rho_m(n-1) = \underline{c}_m^H(n-1) \underline{x}_{m+1}(n-1)$$

yazılabilir. Öncül geri kestirim hatası aşağıdaki gibi tanımlanır:

$$\psi_m(n) = \underline{c}_m^H(n-1) \underline{u}_{m+1}(n) \quad (4.99)$$

(4.44), (4.48) ve (4.99) ifadelerinden yararlanılarak:

$$\psi_m(n) = b_m(n)/\gamma_m(n) \quad (4.100)$$

yazılabilir. (4.42) ifadesinden yararlanılarak da:

$$\begin{aligned} k_m^H(n) x_m(n) &= u_m^H(n) C_m^{-1}(n) \\ &= u_m^H(n) w_{m-1}(n) \end{aligned}$$

yazılabilir. Yukarıdaki son satır arzulan işaretin en küçük kare kestirimini verir. Bu değer $\hat{d}(n)$ ile temsil edilsin. Buna göre (4.98) ifadesi yeniden yazılırsa aşağıdaki gibi basitleşir:

$$\rho_m(n) = \lambda \rho_m(n-1) + \frac{b_m(n)}{\gamma_m(n)} [d^*(n) - \hat{d}^*(n)] \quad (4.101)$$

Ardıl kestirim hatası aşağıdaki gibi tanımlanırsa:

$$\begin{aligned} e_m(n) &= d(n) - \hat{d}(n) \\ &= d(n) - w_{m-1}^H(n) u_m(n) \end{aligned} \quad (4.102)$$

(4.101)'deki zaman güncelleme rekürsriyonu basitleşerek aşağıdaki hale gelir:

$$\rho_m(n) = \lambda \rho_m(n-1) + \frac{b_m(n)}{\gamma_m(n)} e_m^*(n), \quad (4.103)$$

$m = 0, 1, \dots, M$

(4.102) ifadesinde, m yerine $m+1$ yazılırsa:

$$e_{m+1}(n) = d(n) - \hat{w}_m(n) u_{m+1}(n)$$

elde edilir. (4.74) ve (4.89) ifadeleri kullanılarak ardıl ileri kestirim hatası $e_{m+1}(n)$ aşağıdaki gibi ifade edilebilir:

$$\begin{aligned} e_{m+1}(n) &= d(n) - \phi_m^H(n) b_{m+1}(n) \\ &= d(n) - \sum_{i=0}^m \phi_i^*(n) b_i(n) \end{aligned} \quad (4.104)$$

$$d(n) - \sum_{i=0}^m \phi_i^*(n) b_i(n) = e_m(n)$$

elde edilir. Böylece ardıl kestirim hatasının derece güncelleme rekürsiyonu aşağıdaki gibi yazılır:

$$e_{m+1}(n) = e_m(n) - \phi_m^*(n) b_m(n), \quad m=0, 1, \dots, M \quad (4.105)$$

Böylece birleşik süreç kestiriminin rekürsiyonları tamamlanmıştır. Regresyon katsayılarını (4.94) ifadesinden hesaplamak mümkündür.

Algoritmanın önkoşullandırılmasında ilk önce $m=0$ için:

$$e_0(n) = d(n) \quad (4.106)$$

alınır [bkz. Şekil 4-2]. Sonra da $n=0$ için:

$$\rho_m(0) = 0, \quad m = 0, 1, \dots, M \quad (4.107)$$

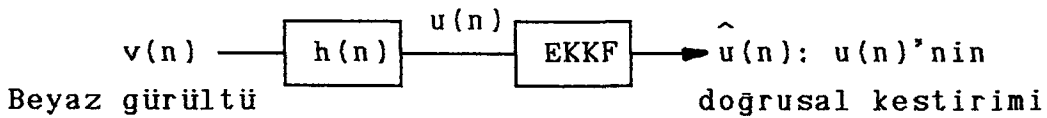
Birleşik süreç kestirimi algoritması Tablo 4-1'e dahil edilmiştir.

BÖLÜM 5. BİLGİSAYAR DENEYLERİ

Rekürsif EKK algoritmasıyla ilgili, doğrusal kestirim ve adaptif kanal dengeleme deneyi olmak üzere iki ayrı deney yapılmıştır. Bilgisayar deneyleri bir bilgisayar programı olarak birleştirilmiş, programda seçimli hale getirilmiştir. Bilgisayar deneyinin program kullanım klavuzu EK A'da, program listesi EK D'de verilmiştir.

5.1 Doğrusal Kestirim Deneyi

Bu deneyde bir beyaz gürültü işareti $v(n)$, yükseltilmiş kosinüs filtesine sokularak $u(n)$ elde edilmiş, $u(n)$ de EKK filtresine giriş işareti olarak verilerek filtre çıkışında $u(n)$ 'nin doğrusal kestirimi elde edilmiştir. Sistemin blok şeması Şekil 5-1'de verilmiştir.



Şekil 5-1 Doğrusal kestirim deneyi blok şeması

Yukarıdaki şekilde $h(n)$ yükseltilmiş kosinüs filtesinin birim impuls cevabıdır ve aşağıdaki gibi verilir [10]:

$$h(n) = \begin{cases} 1/2 \left[1 + \cos(2\pi(n-2)/W) \right], & n=1,2,3 \\ 0, & \text{diğer yerlerde} \end{cases}$$

Yükseltilmiş kosinüs filtresinin çıkışında $u(n)$ şöyle elde edilir [11]:

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

$\hat{u}(n)$ ise $u(n)$ 'den ileri kestirim hatasının çıkarılmasından elde edilir:

$$\hat{u}(n) = u(n) - f_M(n)$$

W , yani kanal parametresi özdeğer saçılımını (eigenvalue spread) kontrol eder. W arttıkça özdeğer saçılımı da artar. Bu deneyde $v(n)$, $u(n)$, $\hat{u}(n)$ ve kestirim hata faktörü $\gamma_{M+1}(n)$ verilen derece, örnek sayısı ve özdeğer saçılım faktörüne göre hesaplanarak çizilir.

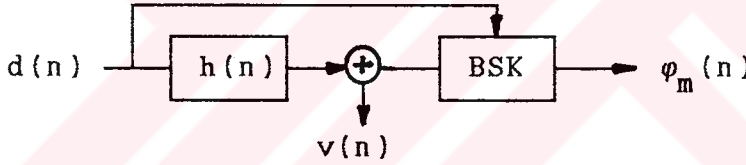
Ek B'de 250 örnek için 3. derece ve 8. derece için yapılan deneyler görülmektedir. Bu deneylerde beyaz gürültü varyansı 0.05 ve özdeğer saçılım faktörü 2.9 alınmıştır. Deney sonuçlarından görüldüğü gibi 3. derece için sistem normal olarak daha çabuk yakınsamaktadır.

5.2 Adaptif Kanal Dengeleme Deneyi

Bu deneyde Rekürsif EKK birleşik süreç kestirimi ile adaptif kanal dengeleme yapılacaktır. Sistemin blok şeması Şekil 5-2'de verilmiştir. Bu deneyde, arzu edilen işaret $d(n)$ de bir beyaz gürültü sürecidir. Kanal, bir önceki deneyde olduğu gibi yükseltilmiş kosinüs filtresidir. Arzu edilen işaret $d(n)$ ve $d(n)$ kanaldan geçtikten sonra, çıkan işaretin üzerine beyaz gürültü eklenerek elde edilen işaret birleşik süreç kestiricisine giriş olarak gelir. Bu işaretlerin her örneğinde diğer bir deyişle zaman geçtikçe sistem regresyon katsayılarını öğrenir. Bu öğrenmeye öncül kestirim hatalarının ($\alpha_{M+1}(n)$) her bağımsız deneme için karesel

ortalamalarına (ensembled-averaged squared) bakılarak karar verilir. Bu hata faktörü belli bir örnekten sonra azalmamaya başlar ve bir değer etrafında sabitleşir. Bu faktörün limit değerinin özdeğer saçılımı ile ilişkisi bu deneyde gözlenmiştir. Özdeğer saçılımı arttıkça hata faktörünün limit değeri büyümektedir. Özdeğer saçılımı ilişki (korelasyon) matrisinin durumunu gösterir. Durumdan kasıt, matrisin tersinin alınabilirliğidir. Öz-Özdeğer saçılımı ne kadar büyükse matrisin durumu o kadar kötüdür [12].

Bu deneyde 100 örnek ve 3, derece için, özdeğer saçılım faktörü 2.9, 3.1, 3.3 ve 3.5 için denemeler yapılmıştır. Beyaz gürültü varyansı 0.05 dir. Sonuçlar EK C'de verilmiştir. Şekillerden görüldüğü gibi özdeğer saçılımı arttıkça ortalama karesel hatanın limit değeri artmaktadır.



Şekil 5-2 Adaptif kanal dengeleme deneyi blok şeması

SONUÇLAR VE ÖNERİLER

Rekürsif en küçük kareler algoritması hızlı yakınsayan güvenli bir algoritmadır. Aslında yakınsama hızı olarak FTF algoritmasıyla benzer olup modüler yapıyla tercih edilen bir algoritmadır. Bu modülerlik sayesinde geniş ölçekli tümleşik devrelerle gerçekleştirme kolaylığı vardır. Bu yüzden tercih edilen bir yöntemdir.

En küçük kareler yönteminin temel olarak kullanılması rekürsif en küçük kareler yöntemine bir özellik daha sağlamıştır; giriş işaretinin istatistiksel özelliklerine bakmaksızın yöntem rahatlıkla uygulanabilmektedir. Bu özelliği ile algoritma deterministik kabul edilir. Anlaşılacağı üzere algoritma hiç bir istatistiksel ön bilgiye muhtaç değildir. LMS yönteminden, bu yönü ile yöntemiyle oldukça farklıdır. Zaten uygulamalar göstermiştir; LMS algoritmasından daha hızlı ve daha güvenlidir.

EKK algoritmasının deterministik olması durağan olmayan ortamlar da çalışmasını sağlar. Eğer ortam durağan değilse EKK algoritmasının kullanılması öneri olarak söylenebilir.

Algoritmada kullanılan ağırlık faktörü λ 'nın durağan ortamlarda 1 alındığı, durağan ortamlarda ise $\lambda < 1$ olması gerektiği Bölüm 4'de belirtilmişti. Bu konu da ayrı bir araştırma konusudur.

Bir önerilecek husus da yukarıda sözü geçen algoritmaların simüle edilerek karşılaştırılmasıdır. Bu durumda aralarındaki farklar çok daha iyi görülebilir.

KAYNAKLAR

- [1] HAYKIN, S., Adaptive Filter Theory, Prentice Hall, Englewood Cliffs, 1986.
- [2] CIOFFI, J. M., KAILATH T., Fast, Recursive-Least Square Transversal Filters For Adaptive Filtering, IEEE Trans. Acoust., Speech, and Signal Processing, vol. ASSP-32, pp.304-337, 1984.
- [3] MAKHOUL, J., Linear Prediction: A tutorial Review, Proc. IEEE, vol 63, pp. 561-580, 1975.
- [4] MAKHOUL, J., Stable and Efficient Lattice Methods For Linear Prediction, IEEE Trans. Acoust., Speech, and Signal Processing, vol. ASSP-25, pp. 423-428, 1977.
- [5] LEV-ARI, H., KAILATH, T., CIOFFI, J., Least-Squares Adaptive Lattice and Transversal Filters: A Unified Geometric Theory, IEEE Trans. Information Theory, vol. IT-30, pp. 222-236, 1984.
- [6] ORFANIDIS, S. J., Optimal Signal Processing, McMillan, New York, 1988.
- [7] LEE, D. T. L., MORF, M., FRIEDLANDER, B., Recursive Least-squares Ladder Estimation algorithms, IEEE Trans. Circuits and Systems, vol CAS-28, pp. 467-481, 1981.
- [8] ELEFThERIOU, E., FALCONER, D. D., Tracking Properties and Steady State Performance of RLS Adaptive Filter Algorithms, Report SCE-84-14, Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, 1984.
- [9] GRIFFITS, L. J., An adaptive Lattice Structure For Noise-cancelling Applications, Proc. IEEE Intern. Conf. Acoust., Speech, and Signal processing, Tulsa, Okla., pp. 87-90, 1978.
- [10] SATORIUS, E., ALEXANDER, S. T., Channel Equalization Using Adaptive Lattice Algorithms, IEEE Trans. Commun., vol COM-27, pp. 899-905, 1979.
- [11] SATORIUS, E. H., PACK, J. D., Application of Least Squares Lattice Algorithms to Adaptive

Equalization, IEEE Trans, Commun., vol.
COM-29, pp. 136-142, 1981.

- [12] STEWART, G. W., Introduction to Matrix
Computations, Academic Press, New York, 1973.

EK A

Bilgisayar deneyi programının kaynak programının adı EKK.C, sistem satırından çalışabilir programın adı EKK.EXE dir. EKK.EXE ve grafik sürücülerin bulunduğu disket bilgisayara sokulduktan sonra ya bu program paketi sabit diske kopyalanır ya da disketin takıldığı disket sürücüyü geçilir. Bu geçme, sistem satırından sürücünün adı ve yanına iki nokta konulup <ENTER> tuşuna basılarak yapılır. Örneğin "A" sürücüsüne disket takıldıysa sistem satırına (C> görülür) A: yazılarak <ENTER> tuşuna basılır. Bundan sonra programın adı (EKK) yazılıp <ENTER> tuşuna basılarak program çalıştırılır. Programla ilgili açıklamalar okunduktan sonra bir tuşa basılarak program verilerinin girilmesine geçilir. İlk önce filtre derecesi, ardından; örnek sayısı, beyaz gürültü varyansı ve özdeşger saçılım faktörü istenir bu değerler verildikten sonra bir mönü çıkar. Burada "0" yazıldığı taktirde "Dorusal Kestirim Deneyi", "1" yazıldığı taktirde "Adaptif Kanal Dengeleme Deneyi" yapılacağı söylenir. Seçime göre deneylerden biri yapılır. Doğrusal Kestirim Deneyi seçilirse hesaplardan sonra ekrana beyaz gürültü, giriş işareti çizilir. Bir tuşa basılarak kestirilmiş giriş işareti kestirim hata faktörü görülür. Adaptif Kanal Dengeleme Deneyi seçilirse kaç tane bağımsız deney yapılacağı sorulur bu cevaplandırıldıktan sonra hesaplamalara geçilir. Yapılan deney numarası, ekranda bir sayaç vasıtasıyla gösterilir. Deneyler bittikten sonra ekrana ortalama karesel hata faktörü çizilir.

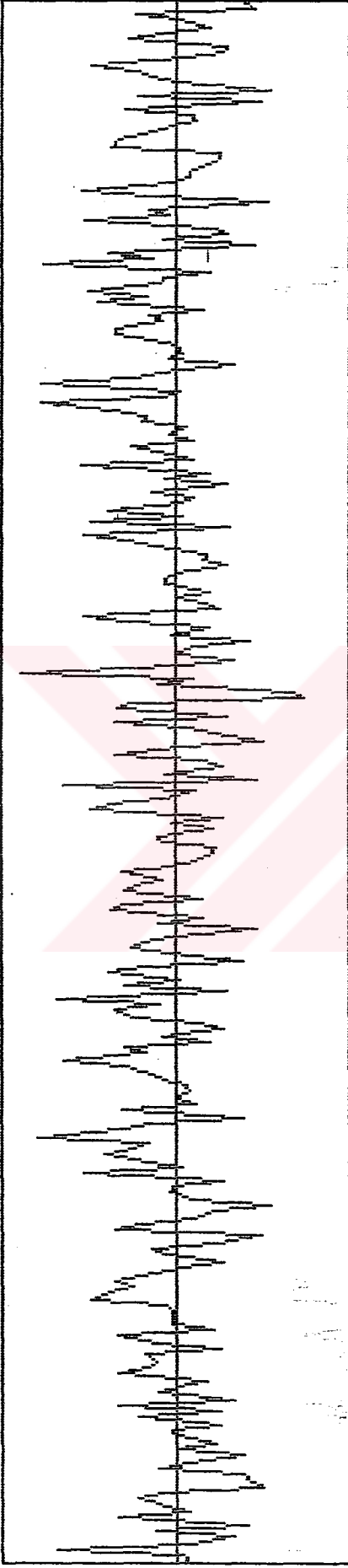
EK B

Bu ekte Doğrusal Kestirim Deneyi'nin örnek bilgisayar grafik çıkışları verilmiştir. Takip eden ilk iki sayfada $M=3$ için, onları takip eden iki sayfada $M=8$ için çıkışlar yer almaktadır.



* u(n) :beyaz gürültü dizisi (white noise array) *

0.7

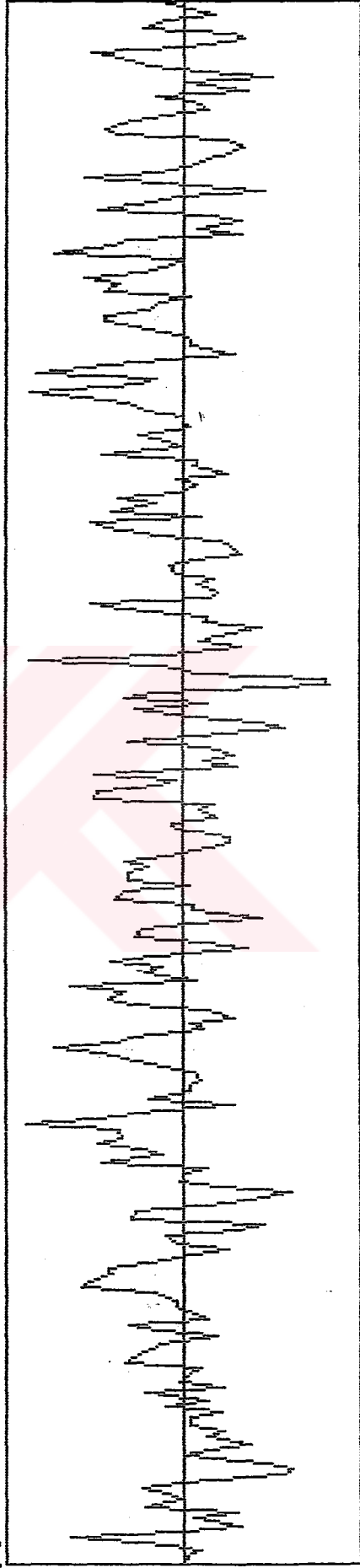


-0.7

249

* u(n) : kestiricinin giriş dizisi (Tap-input array) *

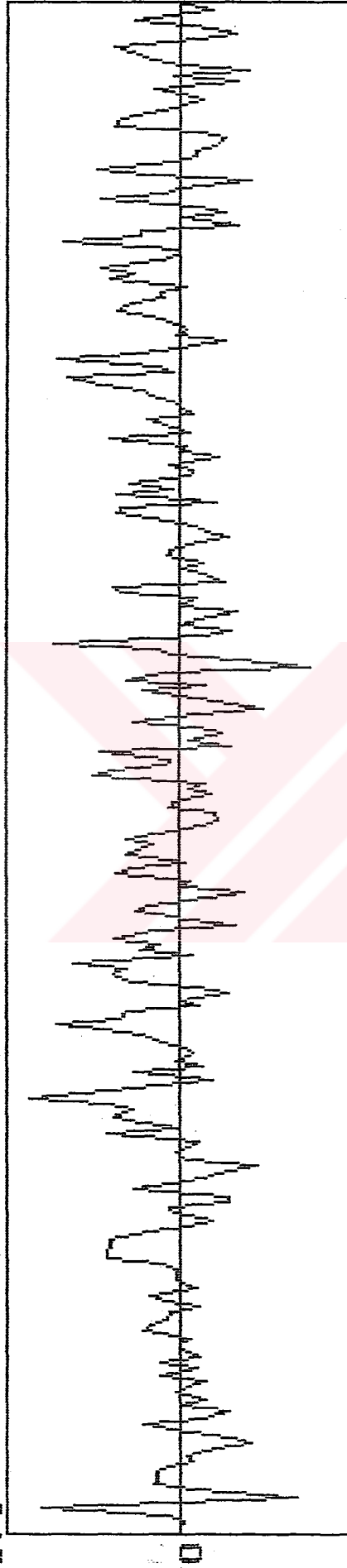
0.7



-0.7

249

* Kestirilmis (predicted) $u(n)$ *



-0.4 249

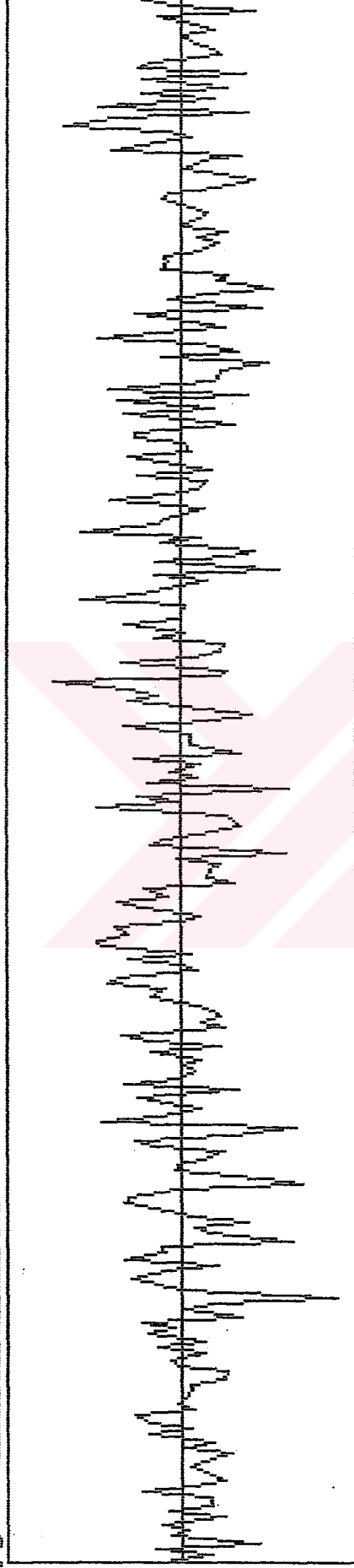
* Kestirim Hata Faktoru (Prediction Error Factor) *



1 0 249

* u(n) :beyaz gürültü dizisi (white noise array) *

0.8

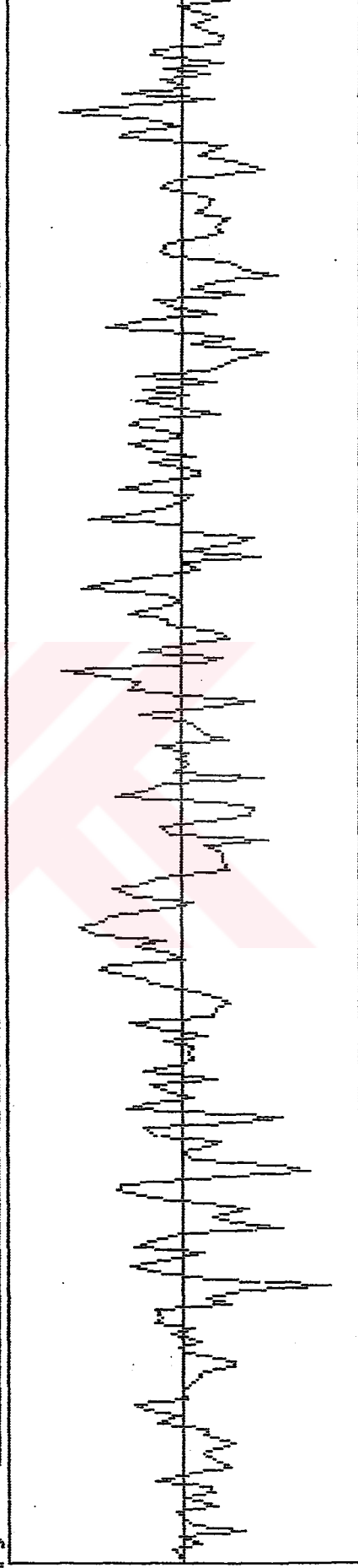


-0.8

249

* u(n) : kestiricinin giris dizisi (Tap-input array) *

0.9

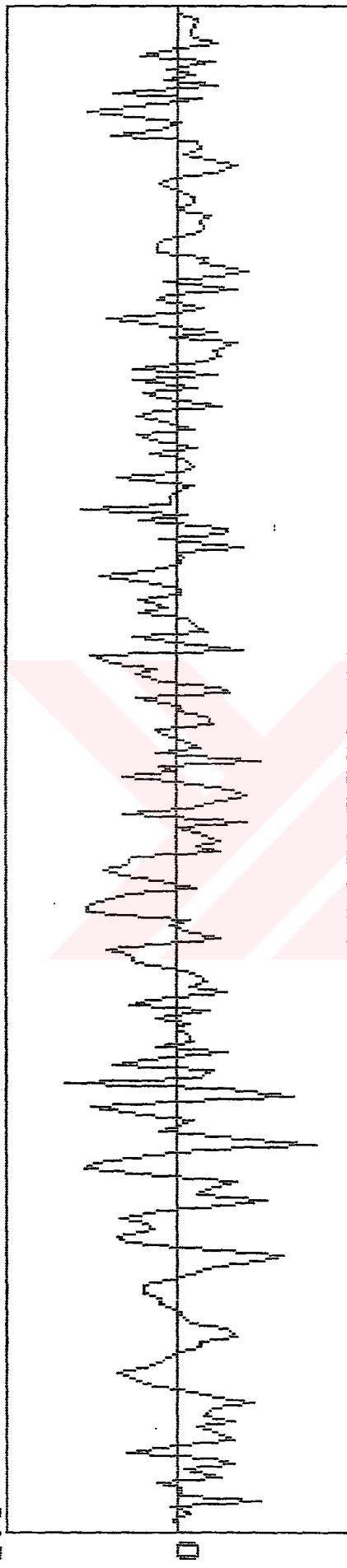


-0.9

249

* Kestirilmis (predicted) u(n) *

0.5

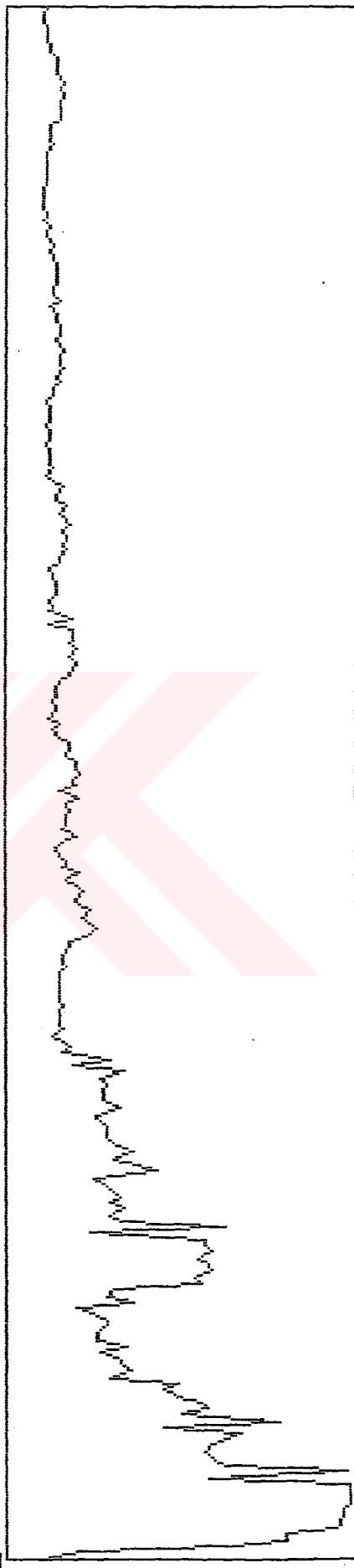


-0.5

249

* Kestirim Hata Faktoru (Prediction Error Factor) *

1



0

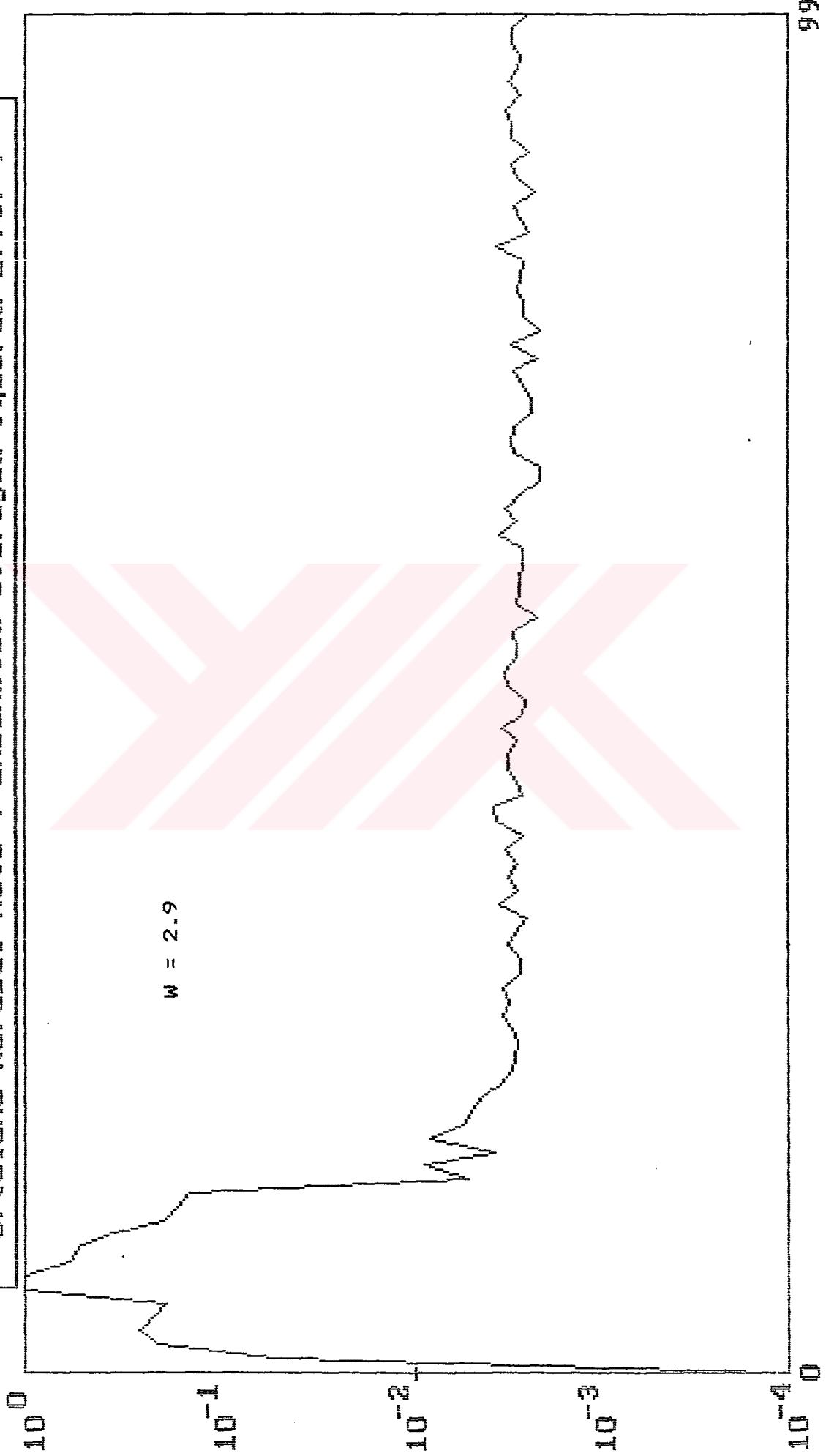
249

EK C

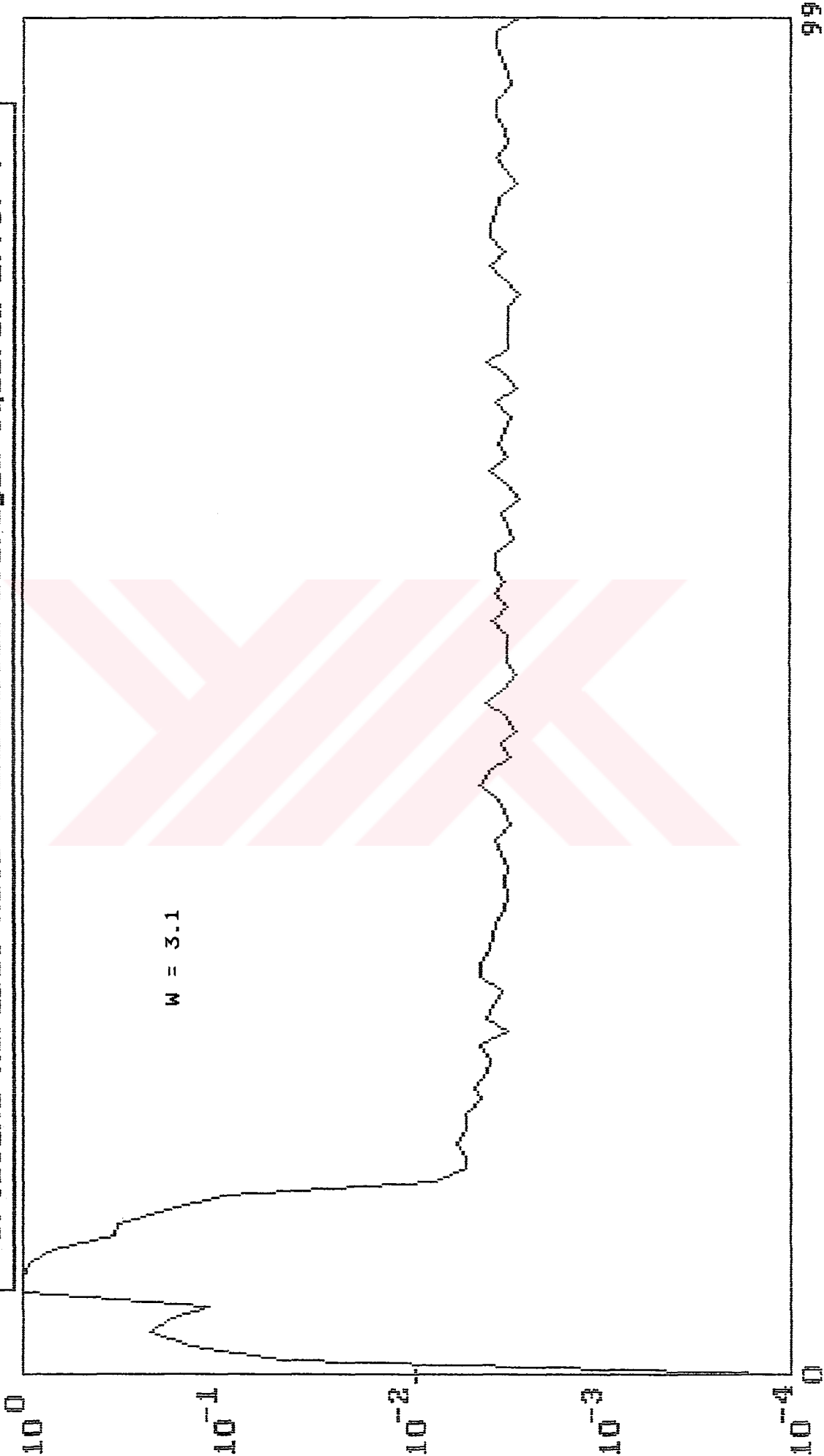
Bu ekte Adaptif Kanal Dengeleme Deneyi'ne ait örnek bilgisayar grafik çıkışları verilmiştir. Takip eden dört sayfada sırasıyla özdeğer saçılım faktörü 2.9, 3.1, 3.3 ve 3.5 için ortalama karesel hata faktörü çizimleri yer almaktadır.



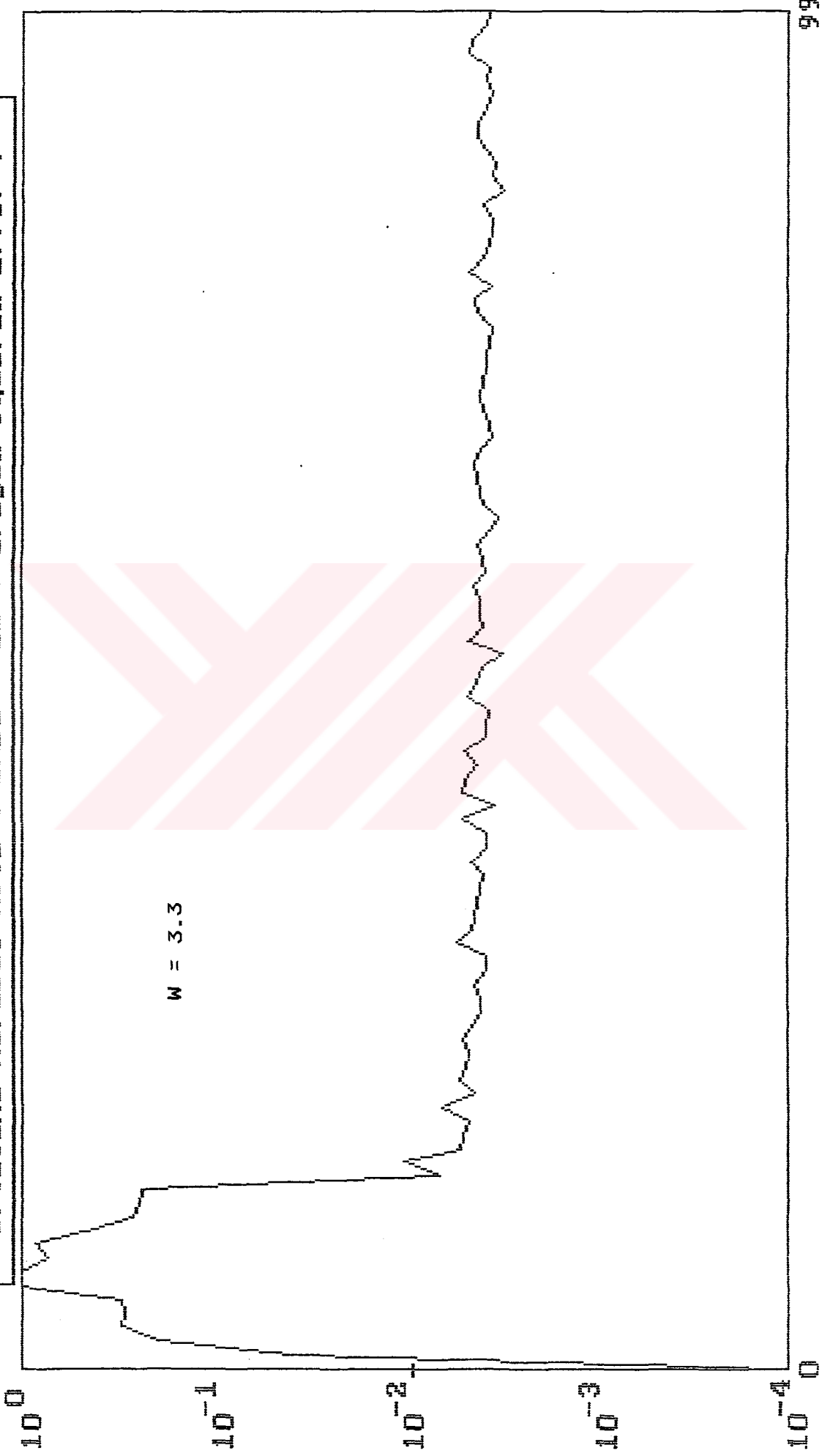
Ortalama Karesel Hata (Ensembled-averaged Squared Error)



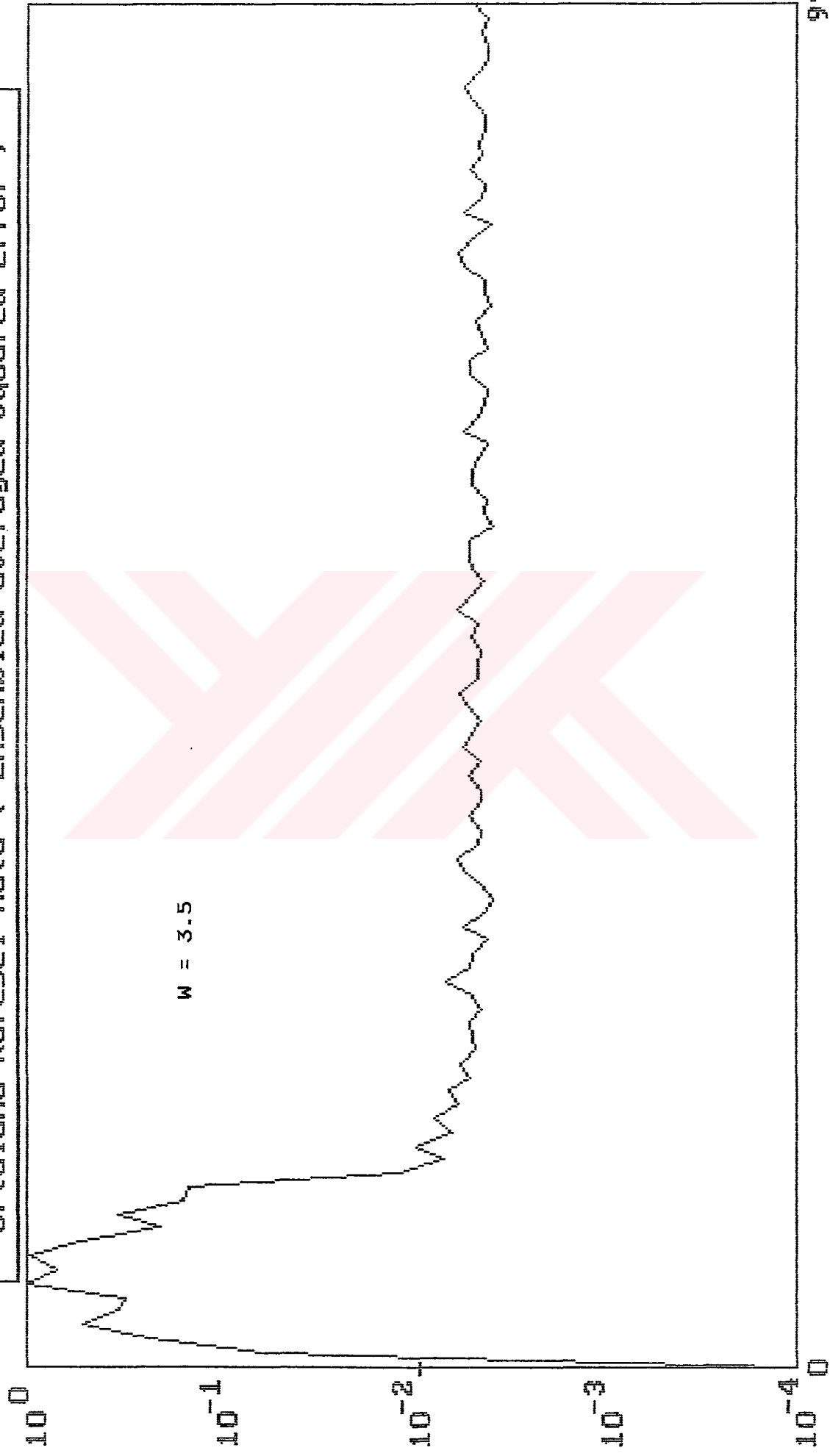
Ortalama Karesel Hata (Ensembled-averaged Squared Error)



Ortalama Karesel Hata (Ensembled-averaged Squared Error)



Ortalama Karese1 Hata (Ensembled-averaged Squared Error)



EK D

```

/*****
/*
/*
/*      Program Adı      : EKK.C
/*      Versiyon         : 1.0
/*      Programlayan     : Sadik Arslan
/*      Konu             : Rekursif En Kucuk Kare Kafes
/*      Filtre Algoritmasi kullanilarak dogrusal kesti-
/*      rim ve adaptif kanal dengeleme gerceklestirmek.
/*
/*
/*
/*      Aciklama        : Bu program grafik islemleri icermekte-
/*      dir.  IBM PS/2 veya AT ( 80286 islemcili) uyumlu,
/*      grafik cizebilen tum bilgisayarlarda calisir.
/*      Program makinede bulunan grafik kartina gore
/*      HERC.BGI, CGA.BGI, EGAVGA.BGI, ATT.BGI,
/*      IBM8514.BGI ve PC3270.BGI dosyalarindan birini
/*      yukler. Bu grafik surucu dosyalarinin tumu varsa
/*      program her ortamda calisir.
/*
/*
/*
/*****/

```

```
#ifndef __TINY__
#error bu program "tiny" bellek modelinde calisamaz !!!
#endif
```

```
#include <dos.h>
#include <math.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <alloc.h>
#include <stdarg.h>
#include <graphics.h>
```

```
#define ESC      0x1b
#define TRUE     1
#define FALSE    0
#define ON       1
#define OFF      0
#define pi_nu    3.14
#define pi       *pi_nu
#define XL       1000
#define YL       1000
#define xx       *kx
#define yy       *ky
#define lamda    1
```

```

#define delta    0.003
#define LOCATION_0 0
#define LOCATION_1 1

#define K(i,j)    K[N*(i)+j]
#define F(i,j)    F[N*(i)+j]
#define B(i,j)    B[N*(i)+j]
#define Tf(i,j)   Tf[N*(i)+j]
#define Tb(i,j)   Tb[N*(i)+j]
#define f(i,j)    f[N*(i)+j]
#define b(i,j)    b[N*(i)+j]
#define nu(i,j)   nu[N*(i)+j]
#define ro(i,j)   ro[N*(i)+j]
#define e(i,j)    e[N*(i)+j]
#define k(i,j)    k[N*(i)+j]

#define sgn(x)    fabs(x)/(x+1e-6)

double  *v;
double  *u;
double  *eu;

double  *K;
double  *F;
double  *B;
double  *nu;
double  *Tf;
double  *Tb;
double  *f;
double  *b;
double  *ro;
double  *k;
double  *e;
double  *alfa;

double  W;
double  varyans;
double  a_eksi_1;
double  a_eksi_2;
double  a_eksi_3;
int      N,M;

char *Fonts[] = {"DefaultFont"};

char *LineStyle[] = {"SolidLn"};

char *TextDirect[] = {"HorizDir", "VertDir"};

char *HorizJust[] =
    { "LeftText", "CenterText", "RightText"};

char *VertJust[] =
    { "BottomText", "CenterText", "TopText"};

struct PTS
{
    int x, y;
};

```

```

int    GraphDriver;
int    GraphMode;
double AspectRatio;
int    MaxX, MaxY;
double kx,ky;
int    MaxColors;
int    ErrorCode;
struct palettetype palette;

void EKK_loop(void);
void EKK(int LOOP);
void Initialize(void);
void Pause(void);
void MainWindow(char *header);
void StatusLine(char *msg);
void DrawBorder(void);
void changetextstyle(int font, int direction,
                    int charsize);
int  gprintf(int *xloc, int *yloc, char *fmt, ... );
//=====

/*****
/* freeall: EKK algoritmasinda kullanılan dizi ve mat-*/
/* risler için yapılan dinamik bellek tahsisini iptal */
/* ederyapar */
*****/

void freeall(void)

{
    farfree((double far *)v);
    farfree((double far *)u);
    farfree((double far *)eu);
    farfree((double far *)K);
    farfree((double far *)F);
    farfree((double far *)B);
    farfree((double far *)Tf);
    farfree((double far *)Tb);
    farfree((double far *)f);
    farfree((double far *)b);
    farfree((double far *)nu);
    farfree((double far *)ro);
    farfree((double far *)k);
    farfree((double far *)e);
    farfree((double far *)alfa);
}

/*****
/* openall: EKK algoritmasinda kullanılan dizi ve mat-*/
/* risler için dinamik bellek tahsisi yapar */
*****/

int  openall(void)

{
    v=(double *)farcalloc(N,sizeof(double));

```



```

if (v==NULL) return -1;

u=(double *)farcalloc(N,sizeof(double));
if (u==NULL) return -1;

eu=(double *)farcalloc(N,sizeof(double));
if (eu==NULL) return -1;

alfa=(double *)farcalloc(N,sizeof(double));
if (alfa==NULL) return -1;

K=(double *)farcalloc(N*M,sizeof(double));
if (K==NULL) return -1;


F=(double *) farcalloc(N*M,sizeof(double));
if (F==NULL) return -1;

B=(double *)farcalloc(N*M,sizeof(double));
if (B==NULL) return -1;

nu=(double *)farcalloc(N*(M+1),sizeof(double));
if (nu==NULL) return -1;

Tf=(double *)farcalloc(N*M,sizeof(double));
if (Tf==NULL) return -1;

Tb=(double *)farcalloc(N*M,sizeof(double));
if (Tb==NULL) return -1;

f=(double *)farcalloc(N*M,sizeof(double));
if (f==NULL) return -1;

b=(double *)farcalloc(N*M,sizeof(double));
if (b==NULL) return -1;

ro=(double *)farcalloc(N*M,sizeof(double));
if (ro==NULL) return -1;

k=(double *)farcalloc(N*M,sizeof(double));
if (b==NULL) return -1;

e=(double *)farcalloc(N*(M+1),sizeof(double));
if (b==NULL) return -1;

return 0;
}

/*****
/* PlotChart : Istenilen buyuklukte bir adaptif grafik*/
/* penceresini acararak verilen diziyi pencereye sigdi- */
/* raarak cizer */
*****/

void PlotChart(int N,double x[],
               double px1,double py1,double px2,double py2)

```

```

{
    int i;
    char str[25];
    int gpx,gpy;
    double vxadim, vyadim,vymax,max;
    vymax = fabs(x[0]);
    for (i=1; i<N; i++)
        { max=fabs(x[i]);
          if (max > vymax ) vymax = max;
        }
    vymax = ceil(vymax * 10)/10;
    vyadim = (py2-py1)/(2.2 * vymax);
    vxadim = (px2-px1)/(N-1);
    rectangle (px1,py1,px2,py2);

    line(px1,(py1+py2)/2,px2,(py1+py2)/2);
    gcvt(vymax,4,str);
    gpx = (int) px1; gpy = (int) (py1 - 25yy);
    gprintf(&gpx,&gpy,str);
    gcvt(-vymax,4,str);
    gpy = (int) (py2 + 10yy);
    gprintf(&gpx,&gpy,str);
    gpx = (int) (px1 - 10xx); gpy = (int) (py1+py2)/2;
    gprintf(&gpx,&gpy,"0");
    itoa(N-1,str,10);
    gpx= (int) px2; gpy = (int) (py2 + 10yy);
    gprintf(&gpx,&gpy,str);

    for (i=0; i<N-1; i++)
        line ( px1 + i * vxadim , (py1+py2)/2 - x[i] * vyadim ,
              px1 +(i+1) * vxadim , (py1+py2)/2 - x[i+1] * vyadim);
    }

/*****
/*  PlotChartP : PlotChart gibi, fakat pozitif isaret-*/
/*  ler icin kullanilir.                               */
*****/

void PlotChartP(int N,double x[],
                double px1,double py1,double px2,double py2)

{
    int i;
    char str[25];
    int gpx,gpy;
    double vxadim, vyadim,vymax,max;
    vymax = fabs(x[0]);

    for (i=1; i<N; i++)
        {
            max=fabs(x[i]);
            if (max > vymax ) vymax = max;
        }
    vymax = ceil(vymax * 10)/10;
    vyadim = (py2-py1)/(1.1 * vymax);
    vxadim = (px2-px1)/(N-1);
    rectangle (px1,py1,px2,py2);
    gcvt(vymax,4,str);

```

```

gpx = (int) px1; gpy = (int) (py1 - 25yy);
gprintf(&gpx,&gpy,str);
gpy = (int) (py2 + 10yy);
gprintf(&gpx,&gpy,"0");
itoa(N-1,str,10);
gpx=(int) px2; gpy = (int) (py2 + 10yy);
gprintf(&gpx,&gpy,str);

for (i=0; i<N-1; i++)
line ( px1 + i * vxadim , py2 - x[i] * vyadim ,
px1 +(i+1) * vxadim , py2 - x[i+1] * vyadim);
}

/*****
/* LogScale : PlotChartP tarafından kullanilir. Loga-*/
/* ritmik dusey eksenin skalasini numaralar. */
*****/

void LogScale(int x,int y,int number)

{

char str[10];
x -=35xx;
gprintf(&x,&y,"10");
itoa(number,str,10);
x+=20xx;
y-=45yy;
gprintf(&x,&y,str);
}

/*****
/* PlotChartL : PlotChartP gibi, fakat dikey eksen */
/* logaritmik olarak ayarlanir. */
*****/

void PlotChartL(int N,double x[],
double px1,double py1,double px2,double py2)

{
int i;
char str[25];
int gpx,gpy;
double vxadim, vyadim,vymax,max;
vymax = fabs(x[0]);
for (i=1; i<N; i++)
{
max=fabs(x[i]);
if (max > vymax ) vymax = max;
}
vyadim = (py2-py1)/4;
vxadim = (px2-px1)/(N-1);
rectangle (px1,py1,px2,py2);
for(i=0; i<=4; i++)
LogScale(100xx,py1+i*(py2-py1)/4,-i);
gpx = px1;
gpy = (int) (py2 + 10yy);
gprintf(&gpx,&gpy,"0");

```

```

gpy = (int) pyl + vyadim * 2;
gprintf(&gpx,&gpy,"-");
itoa(N-1,str,10);
gpx= (int) px2;
gpy = (int) (py2 + 10yy);
gprintf(&gpx,&gpy,str);
for (i=0; i<N-1; i++)
line ( px1 + i * vxadim , pyl - log10(x[i]) * vyadim ,
      px1 +(i+1) * vxadim , pyl - log10(x[i+1]) * vyadim);
}

/*****
/* GraphTittle : Kullanilan alt ve ust grafik pencere- */
/* relerine      baslik atmak icin kullanilir.      */
*****/

void GraphTittle(int Location,char *msg)

{
int gpx,gpy;

switch (Location)
{
case 0: rectangle(150xx,50yy,850xx,90yy);
        gpx = 500xx;
        gpy = 60yy;
        break;
case 1: rectangle(150xx,500yy,850xx,540yy);
        gpx = 500xx;
        gpy = 510yy;
        break;
default: StatusLine(
        " GraphTittle : Boyle bir yer tanimli deyil!!!");
}
gprintf(&gpx,&gpy,msg);
}

/*****
/* Plot: Kullanilan grafik pencerelerin koordinatla- */
/* rini belirler, verilen diziyi ve mesajı gerekli */
/* programlara gönderir */
*****/

void Plot(char Location,int N,double x[],char *msg)

{
switch (Location)
{
case 0: PlotChart(N,x,100xx,100yy,900xx,400yy);
        GraphTittle(0,msg);
        break;
case 1: PlotChart(N,x,100xx,550yy,900xx,850yy);
        GraphTittle(1,msg);
        break;
case 2: PlotChartP(N,x,100xx,550yy,900xx,850yy);
        GraphTittle(1,msg);
        break;
case 3: PlotChartL(N,x,100xx,550yy,900xx,850yy);

```

```

        GraphTittle(1,msg);
        break;
    case 4: PlotChartL(N,x,100xx,100yy,900xx,850yy);
        GraphTittle(0,msg);
        break;
    default: StatusLine(
        "Plot : Boyle bir yer tanimli degil!!!");
    }
}

/*****
/* ScaleNorm : Verilen diziyi cizilecek pencereye */
/* sigdirir. */
*****/

double *ScaleNorm(int N,double x[])

{
    double absmean;
    int n,bool;
    cevrim:
    for (n=0; n<N; n++) absmean +=fabs(x[n]);
    absmean = absmean/N;
    bool = 0;
    for (n=0; n<N; n++)
    {
        if (fabs(x[n]) > (4 * absmean) )
        {
            x[n] = 0.9 * x[n];
            bool =1;
        }
    }
    if (bool==1) goto cevrim;

    return (double *) x;
}

/*****
/* gran : Beyaz Gurultu Ureteci Version 1.0 */
/* Prototip: bknz. [6]:Optimal Signal Processing */
/* - Gaussian Random Number Generator - */
*****/

double gran(mean,sigma)

double mean,sigma;
{
    double u=0;
    int i;
    for (i=0;i<12;i++) u+= (double) random (10000) / 10000;
    u = sigma * (u-6) + mean;
    if (fabs(u)> sigma * 10) return sigma * 10 * sgn(u);
    else return u;
}

/*****
/* Gauss : Beyaz gurultu dizisi  uretir. */
*****/

```

```

void Gauss (int N,double x[],double mean,double sigma)

{
int i;
for (i=0; i<N ; i++)
x[i]=gran(mean,sigma);
}

/*****
/* h(n) : kanalın impulse cevabi. Kanal olarak */
/* yukseltilmiş kosinus filtresi kullanımdır. */
*****/

double h(n)
int n;

{
if (n<1) return 0;
else
return 0.5 * ( 1 + cos(2pi * (n-2) / W) );
}

/*****
/* determine_neg_a : Kanal giriş isaretinin ilk kosul*/
/* isaretlerini üretir. */
*****/

void determine_neg_a(double varyans)

{
a_eksi_1 = gran(0,sqrt(varyans));

a_eksi_2 = gran(0,sqrt(varyans));
a_eksi_3 = gran(0,sqrt(varyans));
}

/*****
/* noise : Kanalın giriş dizisini üretir */
*****/

double noise(int n)
{
if (n<0)
{
switch (n)
{
case -1: return a_eksi_1;
case -2: return a_eksi_2;
case -3: return a_eksi_3;
default: return 0;
}
}
return v[n];
}

```

```

/*****
/*
/*  Ana fonsiyon: İnsan makina iletisiminin yapildigi,*/
/*  EKK algoritmasının kurulduğu ve grafiklerin çizil-*/
/*  diği ana fonksiyon. */
/*
/*****/

```

```
int main(void)
```

```

{
double temp;
int n,m;
int i;
int LOOP;
unsigned long Nmax;
char res,deney;

randomize();
puts("Rekursif
      En Kucuk Kare Kafes Filtre Algoritması Deneyi");
puts("(Recursive
Least Squares Lattice Filter Algorithm Demonstration)");
puts("");
puts("Versiyon (Version) 1.0  Sadık Arslan 1991");
puts("");
puts("Butun haklari serbesttir.");
puts("(All rights free.)");
puts("");
puts("Bir tusa basin... (press a key...)");
getch();
clrscr();
printf("Filtre derecesi (order of the filter)= ");
scanf("%d",&M);
M++;
Nmax = ( (farcoreleft()-12000)/sizeof(double))/(3+8*M);
printf(
"Ornek sayisi (Number of samples) - max %d - = ",Nmax);
scanf("%d",&N);
if (openall()=-1)
{
printf("Bellek yetmiyor (Not enough memory) !!!\n");
freeall();
exit(1);
}
printf(
"Beyaz gurultu varyansi (variance of white noise) = ");
scanf("%lf",&varyans);
printf(
"Ozdeger sacilim (eigenvalue spread) faktoru; W = ");
scanf("%lf",&W);
printf("\n
[0] Dogrusal Kestirim Deneyi (Linear Prediction Demo)");
printf("\n [1] Adaptif Kanal Dengeleme Deneyi");
printf(" ( Adaptive Channel Equalization Demo)\n\n");
printf("Seciminiz ( Choice ):");
choku:

```

```

deney=getch();
switch(deney)
{
    case '0'      :
    case '1'      : putchar(deney);
                    break;
    default       : goto choku;
}
if (deney == '1')
{
printf("\n\nBagimsiz
deney sayisi ( Number of indepedent trials) = ");
scanf("%d",&LOOP);
}
else LOOP=1;
printf("\nHesaplamalar yapiliyor, lutfen bekleyin... ");
printf("( Computing, please wait... )\n");

//%%%%%%%%%%%%%%
EKK(LOOP);
//%%%%%%%%%%%%%%

for (n=0; n<N; n++) eu[n] = u[n]-f(M-1,n);

Initialize();
setlinestyle(SOLID_LINE,0,NORM_WIDTH);
setcolor(MaxColors-1);
DrawBorder();
if (deney=='0')
{
MainWindow(" *** Cizimler ( Graphics ) ***");
StatusLine("Lutfen bekleyin... (Please wait...)");
Plot(0,N,v,
"* v(n) :beyaz gurultu dizisi (white noise array) *");
Plot(1,N,u,"*
u(n) : kestiricinin giris dizisi (Tap-input array) *");
Pause();
cleardevice();
DrawBorder();
StatusLine("Lutfen bekleyin... (Please wait...)");
Plot(LOCATION_0,N,eu,
"*      Kestirilmis (predicted)  u(n)      *");

// nu icin skala ayarlamasi
//=====
for (n=0; n<N; n++)
{
if ( nu(M,n)>1 ) v[n] = 1; else v[n] = nu(M,n);
if ( nu(M,n)<0 ) v[n] = 1e-4;
}
//=====
Plot(2,N,v,
"* Kestirim Hata Faktoru (Prediction Error Factor) *");
goto cik;
}
DrawBorder();
MainWindow("");
Plot(4,N,alfa,"Ortalama

```



```

Karesel Hata ( Ensembled-averaged Squared Error )");
cik:
StatusLine(
"Program bitti (End of program)
<bir tusa basın (press a key ) >");
getch();
closegraph();
freeall();
return(0);

}

/*****
/* EKKL_loop : v[n] ve u[n] elde edilir, EKK algorit- */
/* sinin ilklestirmesi yapilir ve rekursiyonlari      */
/* gerceklenir.                                     */
*****/

void EKK_loop(void)
{
int n,m;
double temp;

Gauss(N,v,0.,sqrt(varyans));

//=====
// v(n) yükseltilmiş kosinus filtresine sokularak u(n)
// elde edilecek
//=====

determine_neg_a(varyans);
for (n=0; n<N; n++)
{
u[n] =0;
for (m=1; m<=3;m++) u[n] += h(m) * noise(n-m);
u[n] +=gran(0,varyans);
}
for (n=0; n<N; n++)
{
if (fabs(u[n]) > sqrt(varyans) * 10 )
u[n] = sqrt(varyans) * 10 * sgn(u[n]);
}

//=====
// EKK algoritmasının ilkleştirilmesi
//=====
for (m = 1 ; m <= M ; m++)
{
K(m-1,0) = 0;
F(m-1,0) = B(m-1,0) = delta;
}

for (n = 1 ; n < N ; n++)
{
f(0,n) = b(0,n) = u[n];
temp = u[n] * u[n];
F(0,n) = B(0,n) = lamda * F(0,n-1) + temp;
nu(0,n-1) =1;

```

```

}
nu(0,N-1)=1;

for (m=0; m<M; m++) ro(m,0)=0;
for (n=0; n<N; n++) e(0,n)=v[n];

//=====
// EKK algoritmasının rekursiyonlarının gerçekleştiril-
//mesi
//=====

for (m = 1; m<M ; m++)
{
for (n =1; n < N ; n++)
{
K(m-1,n) = lamda *
K(m-1,n-1) + ( b(m-1,n-1) * f(m-1,n) ) / nu(m-1,n-1);
Tf(m,n) = - K(m-1,n) / B(m-1,n-1);
Tb(m,n) = - K(m-1,n) / F(m-1,n);
f(m,n) = f(m-1,n) + Tf(m,n) * b(m-1,n-1);
b(m,n) = b(m-1,n-1) + Tb(m,n) * f(m-1,n);
temp = K(m-1,n) * K(m-1,n);
F(m,n) = F(m-1,n) - temp / B(m-1,n-1);
B(m,n) = B(m-1,n-1) - temp / F(m-1,n);
temp = b(m-1,n-1) * b(m-1,n-1);
nu(m,n-1) = nu(m-1,n-1) - temp / B(m-1,n-1);
}
temp = b(m-1,N-1) * b(m-1,N-1);
nu(m,N-1)= nu(m-1,N-1) - temp / B(m-1,N-1);
}
for(n=1; n<N; n++)
{
for (m=0; m<M;m++)
{
ro(m,n) = lamda * ro(m,n-1) +
b(m,n) * e(m,n) /nu(m,n);
k(m,n)=ro(m,n)/B(m,n);
e(m+1,n)=e(m,n) - k(m,n) * b(m,n);
}
}
for (n=0; n<N; n++)
nu(M,n) = nu(M-1,n) -
b(M-1,n-1) * b(M-1,n-1) / B(M-1,n-1);

for(n=0; n<N; n++)

{
temp =e(M,n)/nu(M,n);
temp = temp * temp;
alfa[n] += temp;
}

}

```

```

/*****
/* EKK : EKK algoritmasını istenildiği kadar çağırır */
/* ve toplam karesel kestirim hatasının ortalaması */
/* bulunur. */
*****/

void EKK(int LOOP)

{
    int i;
    for (i=0; i<N; i++) alfa[i]=0;
    for (i=1; i<=LOOP; i++)
    {
        EKK_loop();
        gotoxy(1,16);
        puts(" ");
        gotoxy(1,16);
        printf("%d",i);
    }

    for (i=0; i<N; i++) alfa[i] = alfa[i]/LOOP;
    for (i=0; i<N; i++)
    {
        if (alfa[i]<1e-4) alfa[i]=1e-4;
        if (alfa[i]>1) alfa[i]=1;
    }
}

/*****
/* Initialize : Grafik sistemi kosullandırır ve olu- */
/* san grafik hatalarını rapor eder. */
*****/

void Initialize(void)

{
    int xasp, yasp;

    GraphDriver = DETECT;
    initgraph( &GraphDriver, &GraphMode, "" );
    ErrorCode = graphresult();
    if( ErrorCode != grOk )
    {
        printf(" Grafik Sistem Hatası:
        %s\n", grapherrormsg( ErrorCode ) );
        freeall();
        exit( 1 );
    }
    MaxColors = getmaxcolor() + 1;
    MaxX = getmaxx();
    MaxY = getmaxy();
    kx = (double) MaxX/XL;
    ky = (double) MaxY/YL;
    getaspectratio( &xasp, &yasp );
    AspectRatio = (double)xasp / (double)yasp;
}

```

```

}

/*****
/*  Pause : Grafik cizen bolumler arasi gecislerden  */
/*  herhangi birinde istege bagli olarak programdan  */
/*  cikmayi saglar.                                   */
*****/

void Pause(void)

{
static char msg[] =
"Programdan cikmak icin Esc tusuna,
devam etmek icin herhangi bir tusabasin.";
int c;
StatusLine( msg );
c = getch();
if( ESC == c ){
closegraph();
freeall();
exit( 1 );
}

if( 0 == c ){
c = getch();
}

cleardevice();
}
/*****
/*  MainWindow : Ana grafik penceresini tanimlar ve  */
/*  grafik portuna kullanima hazirlar.                */
*****/

void MainWindow( char *header )

{
int height;

cleardevice();
setcolor( MaxColors - 1 );
setviewport( 0, 0, MaxX, MaxY, 1 );

height = textheight( "H" );

changetextstyle( DEFAULT_FONT, HORIZ_DIR, 1 );
settextjustify( CENTER_TEXT, TOP_TEXT );
outtextxy( MaxX/2, 2, header );
setviewport( 0, height+4, MaxX, MaxY-(height+4), 1 );
DrawBorder();
setviewport( 1, height+5, MaxX-1, MaxY-(height+5), 1 );
}

/*****
/*  StatusLine : Durum penceresi hazirlanir ve icine  */
/*  verilen mesajı yazar.                               */
*****/

```

```

void StatusLine( char *msg )
{
    int height;

    setviewport( 0, 0, MaxX, MaxY, 1 );
    setcolor( MaxColors - 1 );

    changetextstyle( DEFAULT_FONT, HORIZ_DIR, 1 );
    setttextjustify( CENTER_TEXT, TOP_TEXT );
    setlinestyle( SOLID_LINE, 0, NORM_WIDTH );
    setfillstyle( EMPTY_FILL, 0 );

    height = textheight( "H" );
    bar( 0, MaxY-(height+4), MaxX, MaxY );
    rectangle( 0, MaxY-(height+4), MaxX, MaxY );
    outtextxy( MaxX/2, MaxY-(height+2), msg );
    setviewport( 1, height+5, MaxX-1, MaxY-(height+5), 1 );
}

/*****
/* DrawBorder : Yapilan grafik penceresi kosullarini */
/* tespit ederek pencere etrafina cizgi cizer.      */
*****/

void DrawBorder(void)
{
    struct viewporttype vp;
    setcolor( MaxColors - 1 );
    setlinestyle( SOLID_LINE, 0, NORM_WIDTH );
    getviewsettings( &vp );
    rectangle( 0, 0, vp.right-vp.left, vp.bottom-vp.top );
}

/*****
/* Changetextstyle : Standart grafik kutuphanesi fonk-*/
/* siyonu setttextstyle gibidir fakat font dosyasina */
/* erisirken olusabilecek hatalari kontrol eder      */
*****/

void changetextstyle(
    int font, int direction, int charsize)
{
    int ErrorCode;

    graphresult();
    setttextstyle(font, direction, charsize);
    ErrorCode = graphresult();
    if( ErrorCode != grOk )
    {
        closegraph();
        printf(" Grafik sistem hatasi :
            %s\n", grapherrormsg( ErrorCode ) );
        exit( 1 );
    }
}

```

```

}

/*****
/* gprintf : Standart giris cikis fonksiyonu printf */
/* gibidir fakat grafik ekranda istenilen koordinat- */
/* lara istenilen mesajlari yazar. */
*****/

int gprintf( int *xloc, int *yloc, char *fmt, ... )

{
va_list argptr;
char str[140];
int cnt;

va_start( argptr, fmt );

cnt = vsprintf( str, fmt, argptr );
outtextxy( *xloc, *yloc, str );
*yloc += textheight( "H" ) + 2;
va_end( argptr );
return( cnt );
}

```

ÖZGEÇMİŞ

1966 yılında İstanbul'da doğan Sadık Arslan, 1983 yılında Ümraniye Lisesi'nden mezun oldu. 1984 yılında İstanbul Teknik Üniversitesi Elektirik-Elektronik Fakültesi Elekonik ve Haberleşme Bölümü'nde lisans öğrenimine başladı. Bu bölümden 1989 yılında mezun oldu ve aynı yıl İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme bölümünde yüksek lisans öğrenimine başladı. Sadık Arslan, halen Teletaş Araştırma ve Geliştirme Bölümü'nde çalışmaktadır.

