İSTANBUL TEKNİK ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

DİJİTAL İŞARET İŞLEME VE FIR FİLTRE TASARIMI ALGORİTMALARI

YÜKSEK LİSANS TEZİ

Müh. Metin KALAYCI

Tezin Enstitüye Verildiği Tarih : 31 OCAK 1992

Tezin Savunulduğu Tarih : 18 ŞUBAT 1992

Tez Danışmanı : Doç.Dr. Bülent ÖRENCİK

Diğer Jüri Üyeleri : Prof.Dr. Atilla BİR
Doç.Dr. Leyla GÖREN

ŞUBAT 1992

ÖNSÖZ

Yüksek Lisans Tezimin gerçekleştirmemde bana yardımcı olan başta danışmanım Doç. Dr. Bülent Örencik'e , işyerim UBM A.Ş yöneticilerine ve sayamadığım diğerlerine çok teşekkür ederim.

Mühendis Metin Kalaycı

İÇİNDEKİLER

ÖZET..... iv

SUMMARY..... v

BÖLÜM 1. GİRİŞ..... 1

BÖLÜM 2. SÜREKLİ VE AYRIK ZAMAN İŞARETLERİ..... 3
2.1 Sürekli zaman işaretleri..... 3
2.2 Ayrık zaman işaretleri..... 6
2.2.1 Sürekli zaman işaretlerinin örneklenmesi 6
2.2.2 Belirsizlik..... 6
2.2.3 Örnekleme Teoremi..... 7
2.2.4 Fourier dönüşümlerine nümerik yakınsama 7
2.2.5 Bir ayrık zaman işaretinin Fourier dönüşümü..... 8
2.2.6 Ayrık Fourier dönüşümü..... 9

BÖLÜM 3. AYRIK ZAMAN FOURIER DÖNÜŞÜMLERİNİN ÖZELLİKLERİ VE HESAPLAMA YOLLARI..... 11
3.1 İşaretin Spektrumu..... 11
3.2 Ters DFT..... 12
3.3 Periyodik konvolüsyon özelliği..... 13
3.4 x(n) ve X(k) nın periyodik özellilklleri. 14
3.5 Öteleme ve modülasyon Özellikleri..... 14
3.6 Örnekleme Özelliği..... 15
3.7 DFT ve Z domeni arasındaki bağıntılar.... 15
3.8 Simetri ve gerçek eleman özelliğinden kaynaklanan bağıntılar..... 16

3.9 DFT'nu hesaplama yolları..... 17
3.9.1 Goertzel Algoritması..... 17
3.9.2 Fast Fourier Dönüşümleri (FFT)..... 19
3.9.2.1 FFT (decimation-in-time)..... 21
3.9.2.2 FFT (decimation-in-frequency)..... 31
3.9.2.3 N çarpanlarına ayrılabilir ama 2 nin kuvveti değilse FFT hesabı..... 36

BÖLÜM 4. FIR FİLTRE TASARIMI 43

4.1 FIR filtrelerin özellikleri 43
4.2 FIR filtrelerin frekans domeninde tanımlanması 43
4.3 FIR filtrelerin dört tipi 47
4.4 FIR filtrelerin Frekans cevabının hesaplanması 49

BÖLÜM 5. LİNEER FAZ FIR FİLTRE TASARIMI 50

5.1 İdeal alçak geçiren filtrenin tasarımı 51
5.2 FIR filtre tasarım metodlarının incelenmesi 53
5.2.1 Frekans Örnekleme metodu 53
5.2.2 LS Hata frekans domeni tasarımı 58
5.2.2.1 Ayrık Frekans Örnekleri 53
5.2.2.2 İntegral Karesi Hata Yaklaşım kriteri 60
5.3 Gibss Olayı 61
5.4 Türev alıcının LS hata yakınsaması ile tasarımı 62
5.5 Geçiş bölgeli LS hata yaklaşımı 63
5.6 FIR filtre tasarımında geçiş bölgeleri ağırlık fonksiyonları ve pencereleme 66
5.6.1 Arzu edilen frekans cevabında düzetmeler 67
5.6.2 Geçiş bölgesi kullanımı 67
5.6.3 Ağırlıklı karesel hata kriteri kullanılması 67
5.6.4 FIR filtre tasarımında pencereleme fonksiyonlarının kullanımı 68
5.6.5 Chebyshev Yakınsaması 74
5.6.6 Remez Yerdeğiştirme Algoritması 77

SONUÇLAR VE ÖNERİLER..... 84

KAYNAKLAR 86

EKLER..... 88
EK.A Programlar 88
EK.B Program çıktıları 197
ÖZGEÇMİŞ..... 247

ÖZET

Dijital işaret işleme entegre ve bilgisayar teknolojisindeki gelişmelerin artması sonucu oldukça önem kazanmıştır. Önceleri işlem hızı yüzünden gerçek zamanda yapılamayan işlemler hem bu gelişmeler hemde etkin algoritmaların bulunması sonucu gerçeklenme şansı bulmuşlardır. Bu algoritmalardan en önemlisi fast Fourier algoritması , kısaca FFT dir. Bu algoritma ile çok hızlı şekilde fourier dönüşümleri yapılabilir. Ayrıca direkt metod ve Goertzel algoritmaları FFT kadar hızlı olmamakla birlikte daha basit ve hatta sınırlı birkaç durumda FFT ye göre tercih edilmektedir.

Bu çalışmada ilk önce birinci bölümde sürekli ve ayrık zaman işaretleri anlatılmıştır. İkinci bölümde ayrık zaman fourier dönüşümleri ve hesaplama yolları karşılaştırmalı olarak açıklanmış ve hangi durumlarda tercih edilmesi gerketiği verilmiştir. Dördüncü bölümde FIR filtrelerin genel özellikleri verildikten sonra beşinci bölümde FIR filtre tasarım algoritmaları incelenmiştir. Frekans örnekleme metodu , pencere kullanımı , yakınsama kriterleri , hata kriterleri , Remez algoritması ve Parks-McCellan genel amaçlı FIR tasarım algoritması v.s. incelenmiştir. Bunlarla ilgili program ve çıktılar ek.1 ve ek.2 de yer almıştır. Sonuç olarak genel amaçlı FIR filtre tasarım ve Dijital İşaret İşleme algoritmalarını gerçekleştiren esnek bir program paketi ortaya çıkmıştır.

SUMMARY

DIGITAL SIGNAL PROCESSING AND FIR FILTER DESIGN ALGORITHMS

In this study , digital signal processing , digital filter design methods and their algorithms has been given. Discrete Fourier transform and Fast Fourier Transform and other subroutines were written with MicrosftC . The main concept of this study is computing DFT , FFT and Finite Impulse Respose filter design by means of a user friendly software package.

Digital signal processing is a field which has its roots 18th century mathematics, has become an important modern tool in a multitude of diverse fields of science and technology. Until recently, signal processing has typically been carried out using analog equipments. Some exeptions to this were evident in the 1950s particulary in area where sophisticated signal processing was required. Because of flexibility of digital computers it was often useful to simulate a signal processing system on a digital computer befor implementing it in analog hardware. The vocoder simulation which were carried out in Lincoln and Bell Laboratories is a good example of such simulations. Despite the fact that flexibility of digital computer in signal processing , The processing can always no be done in real time. In keeping with that style , early work on digital filtering was very much concerned with ways in which a filter could be programmed on a digital computer so that with analog to digital conversion of the signal , followed by the digital filtering, followed by the digital to analog conversion , the overall system approximated a good analog filter.

The evolution of a new point of wiew toward digital signal processing was further acclerated by the disclosure in 1965 of an efficent algorithm for computation of the Fast Fourier Transform or FFT. Many signal processing algorithms which had been developed on digital computers required processing times several orders of magnitude greater than real time. Often this was tied to the facts that spectrum was an important component of signal processing and that no efficent means had been known for implementing it.

The fast fourier transform algorithm reduced computation time of the Fourier transform by orders of magbitude. This permitted the implementation of increasingly sophisticated signal processing algorithms with processing times that allowed interaction with the system.

Futhermore , with the realization that the fast Fourier transform algorithm might, in fact, be implementable in special purpose digital hardware, many signal processing algorithms which previously had appeared to be impractical began to appear to have practical implementations with special purpose digital hardware. Another important implication of the fast Fourier transform algorithm was tied to the fact that it was an inherently discrete time concept. It was directed toward the computation of the Fourier transform of a discrete time signal or sequence and involved a sert of properties and mathematics that were exact in discrete time domain. It was not simply an approximation to a continous-time Fourier transform. The importance of this was that it had the effect of stimulating a reformultion of many signal processing concepts and algorithms in terms of discrete-time mathematics and these techniques then formed an exact set of relationship in discrete time domain.

The Discrete Fourier Trasnforms plays an important role in the analysis , design and the implementation of digital signal processing algorithms and systems.

The used notation in the sections is consistent with the standart setup by the IEEE.

The Fourier Series :

The fourier series may be viewed a way to represent a continous time signal s(t) as a superposition of harmonically related sine and cosine waves. Instead of sine and cosine representation exponantiel representation is often used, especially when there are many harmonics in the fourier expansion. The set of complex coefficents in fourier transforms are calld as spectrum of signal.

Discrete Time Signals :

A discrete time signal is obtined from a continuous time signal by sampling at equally spaced time signal. If a discrete time signal is denoted by s_n then the sampling of s(t) every T second gives $s_n(t) = s(nT)$ and n = ..-1,-2,0,..

Amiguity (Aliasing) when Sampling a Cosine Wave :

If the samples of a cosine wave are obtained at a rate of five samples per second shown as sn=1,-.81,.31,.31.-.81, .31 .

İt is reasonable to assume a sampling of a 2 Hz signal gives the same samples when sampling a 3 Hz cosine. The usual convention used to relate a unique signal to a set of samples is to assume that the signal is a lowpass band limited signal. As long as the sampling rate is higher than twice the highest frequeny present in the signal , there is a unique continuous time signal associated with a discrete time signal, viewed as a set of samples. The phenomenon of having other names for continuous time signals with the same samples is called aliasing.

Sampling Theorem :

A band limited signal ,with highest frequency B Hz, can be uniquely recovered from its samples as long as the sampling rate is higher than 2B samples per second. Lower sampling rates may bring on aliasing.

Comparison of continious time and discrete time transforms:

Fourier Series-DFT :

These are signal representations using discrete ,harmonically related frequencies. The discrete

frequeny representation results in periodic time functions , both continuous time and discrete time. The spectrum of discrete time signal is periodic while the spectrum of continuous time signal is not.

Fourier Transforms :

The forier transform of a continuous time signal and the Fourier transform of a discrete time signal are continuous frequency represantations of the corresponding signals. The transforms are function of the real frequency variable w. The spectrum of the continuous time signal is not.

Convolution and Linear Systems :

While Fourier techniques certainly aid signal analysis, they also paly an important role in the study of linear time invariant systems. The time domain description of a linear time invariant system with an input x(t) ,unit impulse response h(t) and output y(t) given by the convolution ,

$$y(t) = \int_{\infty}^{-\infty} h(t-u)\, x(u)\, du.$$

Discrete Fourier Transform (DFT)

The spectrum of signal :

A discrete time signal can be viewed as a finite length sequence of numbers such as x(n) = 1,.6,2,-1,-.4.. .

$x(n) = \cos(2n\pi/N)$ and $DFT(x(n)) = X(k) = \sum_{n=0}^{N1} x(n)\, W^{nk}$

Inverse DFT (IDFT):

x(n) = IDFT(X(k)) = IDFT (DFT(x(n)))

Calculation of DFT from a signal has equal diffculties with calculation of IDFT.

Cyclic Convolution Property :

A fundamental operation in linear digital signal processing is convolution. Aperiodic convolution between two signals x(n) and h(n) is denoted by ,

$$y(n) = x(n) * h(n)$$

Cyclic convolution is denoted by ,

$$y(n) = \sum x(m)_N \; h((n-m)_N) \text{ , the residu of } n \text{ modulo } N = (n)_N$$

$$DFT(y(n)) = DFT(x(n)) \; DFT(h(n))$$

$$y(n) = IDFT(DFT (x(n)) \; DFT (h(n))$$

The Periodic property of x(n) and X(k) :

Strictly speaking, x(n) is defined only over the range of n from 0 to N-1. However , the IDFT of the DFT of x(n) can be calculated

Shifting and modulation properties :

When the signal x(n) is shifted in time , the spectrum X(k) is multiplied by a linear phase shift.

Sampling property :

The DFT of a subset of the original signal values can be found in terms of the DFT of the signal if the subset is viewed as a set of samples of the original signal.

Goertzel algorithm , direct method and the fast fourier transforms are considered while calculating DFT.

Direct Method :

An obvious way to calculate the DFT of a signal x(n) is to implement the definition of the DFT directly. This is done by computing the desired values of X(k). In the case x(n) is complex the real and the imaginary part of x(n) are denoted by X(n) and Y(n), respectively and the real and imaginary parts of the transform by A(k) and B(k).

Goertzel Algorithm for the DFT :

First order filter with the input being the data sequence in reverse order and the value of the polynomial at z being the solution sampled at M=N. Applying this to the DFT gives the Goertzel Algorithm.

Decimation in time Fast Fourier Transforms :

To achieve a dramatic increase in efficency , it ia necessary to decompose the DFT computation into succesively smaller DFT computations. İn this process both the symmeetry and the periodicity of the complex exponantial is used. Algorithms in which the decomposition is based on decomposing the sequence x(n), into succesively smaller subsequences are called decimation in time algorithms.

Decimation in frequency Fast Fourier Algirithms :

The decimation in time FFT algorithms were all based upon the decomposition of the DFT computation by forming smaller and smaller subsequences of input sequence x(n). Alternatively divideing the output sequence X(k) samaller and smaller subsequences in the same manner. This class of FFT algorithms based on this procedure is commonly referred as decimation in frequency.

Properties of FIR filters :

Digital filters with a finite duration impulse response (FIR) have characteristic that make them useful in many applications. Because of the method of implementation , The FIR filters is also called as nonrecursive or convolution filters. The duration or sequence length of the impulse response of these filters is by definition finite therefore , the output can be written as a finite convolution sum :

$$y(n) = \sum_{m=0}^{N-1} h(m)\, x(n-m) \quad \text{by changing index variable ,}$$

$$y(n) = \sum_{m=0}^{n-N+1} h(n-m)\, x(m)$$

$x(n)$ = input ,

y(n) = output,
h(n) = length-N impulse response.

1. FIR filters are always stable.
2. FIR filters can be exactly phase linear.
3. Coefficients can easily be set to 1 part in 65,000 and never change. (Analog filter response can be change because of temperature.)
4. Stopband floors of -50 or -60 dB are much more quickly reached using FIR filters than using analog filters.
5. FIR filters can be made to track simply by altering the clock rate. An example of where this is important would be in detecting (optically) the watermark on a moving paper web. As the paper speed moves faster or slower , the input signal will be shifted in frequency . İf an angle encoder is installed on the pinch rollers , the pulses from it could be used to clock the filter. Thus the filter will automatically track to shift.
6. FIR filters can be designed directly.
7. Complex filters with many bands of different gains , which are virtually impossible to design with analog technics are just as easy as to design for FIR filters as is simple low-pass filter. There are three definitions used by FIR filter design. Trasition width , passband ripple and stopband attenuation. Furthermore calculation of filter length is another problem to be considered in designing FIR filters.

FIR filters is defined in frequency domain. Thare are four types of FIR filters. Two of them are according to symmetry (even , odd). Two of them are according to filter length (even , odd).Frequency sampling , Least squared error , using a transition region and windows , approximation criterias (Remez , Chebyshev , İnteqral squared error , squared error) are given technics and are often used in designing FIR filters. The best window approximation function is given by Kaiser. İn addition , A transition region is used to minimize overshoot . Window is used for eleminating sidelobs in the frequency response. Remez exchange an Parks-McCellan algorithms are an effective way to design for FIR filters such as multiband , lowpass , highpass , differentiators and hillbert transformers.

GİRİŞ :

Lineer filtreleme ve spektrum analizi temel bilim ve teknolojinin pek çok dalında kullanılan işaret işleme konularıdır. Bunlar analog RLC devreleriyle veya dijital olarak genel amaçlı bilgisayar yardımıyla oluşturulabilir. Hem surekli hemde ayrık işaretişlemede işaretlerin veya lineer sistemlerin özellikle frekans domeni açıklamaları kullanılır. Frekans domeninde inceleme alçak geçiren band geçiren , fark alan ve interpolsayon şeklinde filtreleme söz konusu oldugunda zaman domenine göre tercih edilir. Sesli iletişim, sismoloji, sonar, radar ve tıp işaret işleme tekniklerinin kullanıldiği önemli alanlardan bazılarıdır.

Dijital işaret işleme işaretlerin sayısal sekanslar olarak gösterilmesi ve bunların işlenmesiyle ilgilidir. Bunun amacı işaretin karakteristiğini hesaplama veya sinyali arzu edilen bir şekle dönüştürmek içindir. Klasik nümerik analiz formülleri interpolasyon integral ve türev kesinlikle dijital işaret işleme analizleridir. Öte yandan yüksek hızlı dijital bilgisayarlarn elde edlilebilirliğinin artması kompleks ve gelişmiş dijital işaret işleme algoritmalarının ortaya çıkmasını sağlamıştır. Entegre devre teknolojisindeki son gelişmeler çok karmaşık dijital işaret işleme sistemlerini mümkün kılmıştır. İşaret işleme geniş ve zengin bir tarihe sahiptir. EEG , EKG , ses tanıma ve ses iletiminde çoklukla kullanılır. Gürültünün veya girişimin ortadan kaldırılması veya sinyallerin modülasyonu işaret işlemeyi gerektiren alanlardan bazılarıdır. İşaret işleme problemleri sadece tek boyutlu

işaretleri içermez. Aynı zamanda iki boyutlu işaret işleme de kullanılır. Buna örnek olarak resim işlemeyi verebiliriz. Uydudan elde edilen fotoğraflarda hava durumunun anlaşılması veya zarar gören tarım alanlarının hesabı iki boyutlu incelemeyi gerektirir. Son zamanlara kadar işaret işleme analog olarak yapıldı. Dijital işaret işlemeye ilk gereksinimler 1950 lerde karmaşık ve yüksek düzeyli işaret işlemeyi gerektiren bazı alanlarda kendini göstermiştir. Buna örenek olarak jeofiziksel dataların girildikten sonra dakikalar ve saatler süren işlenmesi verilebilir. Ancak bu uygulamalar anlaşılacağı üzere gerçek zamanda yapılmıyordu. Dijital bilgiiayarların gelişmesi bunu çok daha çekici kıldı. Bu süre içinde dijital bilgisayar aynı zamanda simlasyon için kullanıldı. Esnekliğinden ötürü dijital bilgisayar bir analog donanımın simülasyonu için oldukça faydalı oldu. Bu şekilde yeni bir işaret işleme algoritması esnek bir deney ortamında gerçeklenme şansı buldu. Hız maliyet ve büyüklük analog elemanların kullanmında önemli faktördür. Dijital işaret işleme 1965 de fourier dönüşümlerini hesaplayan algoritmalarla ivmelendi. Bu çeşit algoritmalara Fast Fourier Transformu kısaca FFT denir.

2. SÜREKLİ VE AYRIK ZAMAN İŞARETLERİ

2.1. Sürekli zaman işaretleri

Sürekli zaman işaretlerinin analizinde fonksiyonlar t ,zaman değişkeniyle gösterilirler. Bir işaret lineer time invariant sistemden geçtiğinde sadece genlik ve fazı değişir ama frekans aynı kalır. Sistemin frekans spektrumu üzerindeki etkisini incelemek, zaman domeninde etkisini incelemekten daha kolaydır. İşaretlerin frekans domeni açıklaması Fourier serileri ile sayılabilir çoklukta frekans elemanlarının toplamı şeklinde gösterilir. Sürekli bir işaretin Fourier dönüşümü elemanter işaretlerin integral süperpozisyonu ilkesiyle elde edilir. Buna örnek olarak kare dalga şeklindeki bir işaretin Fourier analizini inceleyelim.

2.1.1 Fourier Serileri

S(t) işareti aşağıdaki koşulları sağlıyor olsun :

1. F(t) $c < t < c + 2T$
2. F(t) $c < t < c + 2T$ aralığında sürekli
3. F(t+2T) = F(t) F(t) periyodik , Periyodu 2T

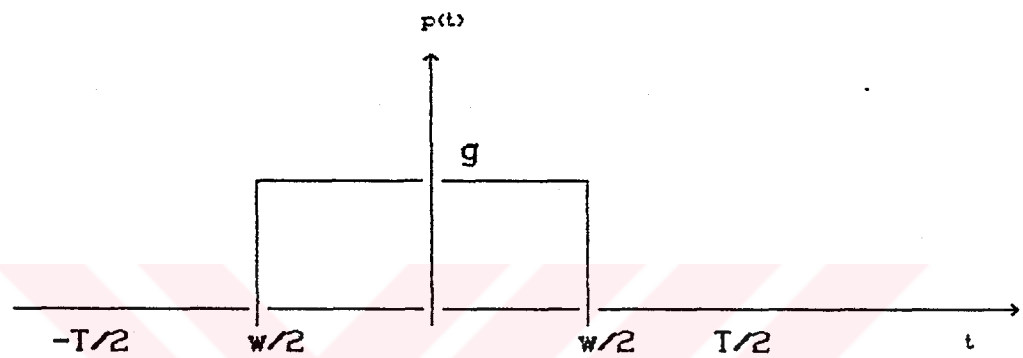
Bu koşullar altında her sürekli nokta için :

$$F(t) = a_o/2 + \sum_{n=1}^{\infty} a_n \cos(2n\pi t / T) + b_n \sin(2n\pi t / T) \qquad (2.1)$$

$$a_n = 1 / T \int_{c}^{c+2l} F(t) \cos (2n\pi t / T) dt \qquad (2.2)$$

$$b_n = 1 / T \int_{c}^{c+2l} F(t) \sin (2n\pi t / T) dt \qquad (2.3)$$

Bir kare dalganın fourier serisini incelersek :



p(t) nin fourier serisi p(t) çift fonksiyon olduğundan sadece cosinüs lü terimlerden oluşur :

$$b_n = 0$$

$$a_n = 1/T \int_{-w/2}^{+w/2} p(t) \cos (2n\pi t/T) dt \qquad (2.4.a)$$

$$a_n = g/T \int \cos (2n\pi t/T) dt \qquad (2.4.b)$$

$$a_n = g/T \; T/2n\pi \sin(2n\pi t/T) \Big|_{t=-w/2}^{w/2} \qquad (2.4.c)$$

$$a_n = g / n\pi \sin (n\pi W / T) \qquad (2.4.d)$$

$$a_o = 1/T \int_{-T/2}^{T/2} p(t) dt \qquad (2.4.e)$$

$$a_o = 1/W \int_{-w/2}^{w/2} g \, dt \;, \quad a_o = 1/W \; g \; t \Big|_{-w/2}^{w/2} \;, \quad a_o = g \qquad (2.4.f)$$

$$F(t) = g/2 + \sum_{n=1}^{\infty} g/n\pi \sin(n\pi W/T) \cos(n\pi t / T) \qquad (2.5.a)$$

Sonlu sayıda eleman alarak p(t) yi ilk üç terimin toplamı olarak şu şekilde yazabiliriz :

$$F(t) = ao + a_1 \cos(2\pi t/T) + a_2 \cos(4\pi t/T) \qquad (2.5.b)$$

Ötelenmiş kare dalga için :

$$F(t)=ao + a_1 \cos(2n\pi t/T) + a_2 \cos(4\pi t/T) + b_1 \sin(2\pi t/T) + b_2 \sin(4\pi t/T) \qquad (2.5.c)$$

Harmoniklerin çok olduğu durumlarda sinüs ve cosinüslü gösterim yerine kompleks üstel gösterim kullanılır.

$$F(t) = co + c_1 e^{j2\pi t/T} + c_2 e^{j4\pi t/T} + c_{-1} e^{2\pi t/T} + c_{-2} e^{j4\pi t/T} \qquad (2.5.d)$$

Şeklinde yazılabilir. x(t) işaretinin kompleks notasyonu :

$$c_k = 1 / T \int_{-T/2}^{T/2} x(t) e^{j2\pi kt/T} dt \qquad (2.6)$$

Ne kadar çok frekans içerilirse yakınsama iyileşeceğinden hata O'a yakınsar.

$$x(t) = \sum_{k=-\infty}^{\infty} c_k e^{j2\pi kt/T} \qquad (2.7.a)$$

i) c(k) ya x(t) nin spektrumu denir.

ii) x(t) reelse ,

$$c_k = c_{-k},$$

$$|c_{-k}| = |c_k|,$$

$$\text{Açı}(c_k) = - \text{Açı}(c_{-k}). \qquad (2.7.b)$$

iii) Kesilmiş, fourier serisi :

$$\hat{x}(t) = \sum_{k=-N}^{N} c_k e^{j2\pi kt/T} \qquad (2.7.c)$$

x(t) ye en iyi yakınsama :

$$E = \int |x(t) - \hat{x}(t)|^2 \, dt \qquad (2.8)$$

hata fonksiyonunun minimize edilmesiyle sağlanır.

iv) Parserval teoremine göre :

$$1/T \int_{-T/2}^{T/2} x(t)^2 \, dt = \sum_{k=-\infty}^{\infty} |c_k|^2 \qquad (2.9)$$

Bu teorem bir sinyalin enerjisinin zaman veya frekans domeninde hesaplanabilecegini gösterir.[1]

2.2 Ayrık Zaman İşaretleri

2.2.1 Sürekli zaman işaretlerinin örneklenmesi

Bir ayrık zaman işareti genellike sürekli zaman işaretinin eşit zaman aralıkarıyla örneklenmesiyle elde edilir. s(t) nin ayrık zamanda T örnekleme periyodu ile gösterilimi :

$s_n = s(nT)$, $n = \ldots -1, 0, 1, 2, \ldots$

2.2.2 Belirsizlik

Sinüs ve cosinüs işaretleri örneklenirken farklı frekanslardaki işaretler aynı örnekleme değerlerini verebilir. Buna belirsizlik denir.

2.2.3 Örnekleme teoremi

En yüksek frekansı B Hz olan bir bant-sınırlı işaret saniyede 2B örnekten fazla örneklenirse belirsizlik ortadan kalkar. Buna göre sonsuz sayıda örnekleme yapılabilir. Bu durumda :

T = 1/2B,

$$x(t) = \sum_{-\infty}^{\infty} x(nT)\ \mathrm{sinc}(t-nT) \qquad (2.10.a)$$

$$\mathrm{sinct}(t) = \sin(\pi t/T) / (\pi t/T) \qquad (2.10.b)$$

Pratikte bu mümkün olamayacağından buna interpolasyon yakınsama yapılır. N örnekleme sayısı için :

$$\hat{x}(t) = \sum_{n=0}^{N-1} x(nT)\ Q(n,t) \qquad (2.10.c)$$

Q(n,t) sınırlı örnekleri kompanze etmek için kullanılan interpolasyon fonksşyonudur. [2]

2.2.4 Fourier Dönüşümlerine Nümerik yakınsama

Genellikle bir işaretin elde edilmesi işaretin örneklerinden değil örneklerin fourier dönüşümünden elde edilir. Nümerik integral kullanılarak bir kare dalga işaretin fourier dönüşümü :

w işaretin genişliği ve adım sayısı n olmak üzere,

$s_n = s(nW)$ dır.

$$\bar{S}(w) = \int_{-\infty}^{\infty} \sum_{-\infty}^{\infty} s_n\ p(t-nW)\ e^{-jwt}\ dt \qquad (2.11)$$

Burada integral ve toplamayı yer değiştirirsek :

$$S(t) = \left(\sum_{n=-\infty}^{\infty} s_n e^{-jvnW} \right) W \left(\sin wW/2 \right) / \left(wW/2 \right) \quad (2.12)$$

2.2.5 Bir ayrık zaman işaretinin fourier dönüşümü

Sonsuz uzunlukta bir ayrık zaman işareti olan x_n ,

$$X(w) = \sum_{n=-\infty}^{\infty} x_n e^{-jvn} \quad (2.13.a)$$

Ters dönüşüm ise :

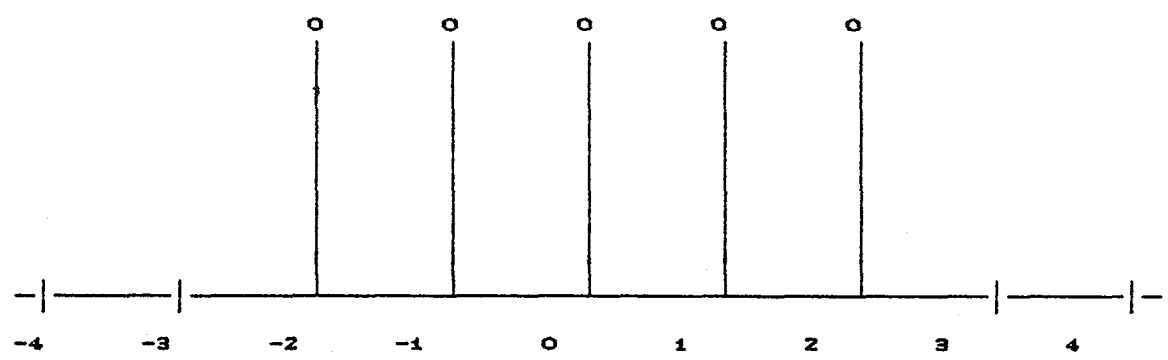
$$x_n = 1/2\Pi \int_{-\Pi}^{\Pi} X(w) e^{jvn} dw \quad (2.13.b)$$

Ayrık ve sürekli fourier dönüşümleri arasındaki temel fark ayrık fourier dönüşümünün frekans değişkeni w ile periyodik olmasıdır. m nin tamsayı değerleri için ,

$$X(w+2\Pi m) = \sum_{-\infty}^{\infty} x_n e^{-j(v+2\Pi m)n} = X(w) \text{ dir.} \quad (2.14)$$

$$p_n = \left\{ \begin{array}{ll} 1 & |n| \leq W/2 \\ 0 & |n| > W/2 \end{array} \right\} \text{ olsun.}$$

W = 4 için , aşağıdaki şekil elde edilir :



$$P(w) = \sum_{-\infty}^{\infty} p_n e^{-jvn} = \sum_{-v/2}^{v/2} e^{-jvn} \qquad (2.15.a)$$

Bu toplam aşağıdaki geometrik serinin toplam formülü kullanılarak :

$$\sum_{n-M}^{n-N} a^n = \frac{a^{M} - a^{N+1}}{1 - a} \qquad (2.15.b)$$

$$P(w) = \frac{\sin((W + 1) w/2)}{\sin(w/2)} \text{ olarak elde edilir.} \qquad (2.15.c)$$

P(w) fonksiyonuna Dirichlet kernel adı verilir.[3]

2.2.6 Ayrık Fourier Dönüşümü

Ayrık zaman işareti x_n $0 \le n \le N-1$ aralığında tanımlı olsun. Ayrık zaman fourier serisi :

$$x_n = 1/N \sum_{k=0}^{N-1} X_k e^{j(2\pi/N)kn} \qquad (2.16.a)$$

ve X_k katsayıları x_n in ayrık zaman fourier dönüşümü (DFT) tarafından verilir.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j(2n\pi/N)kn} \qquad (2.16.b)$$

$$X_k = DFT(x_n)$$

Periyodik fonksiyon periyodu N olmak üzere :

$$x_p(n) = \sum_{m=-\infty}^{\infty} x_r(n-mN) \qquad (2.16.c)$$

ve

$$x_r = \begin{cases} x_n & 0 \le n \le N-1 \\ 0 & \text{diğer} \end{cases}$$

şeklinde gösterilir.

Bir kare dalganın ayrık zaman fourier dönüşümü,

$$x(n) = \begin{cases} 1, & n= 0,1,2,6,7 \\ 0, & n= 3,4,5 \end{cases}$$

şeklinde verilmiş olsun :

N = 8 için DFT,

$$x_k = \sum_{n=0}^{7} x_n\, e^{-j\pi nk/4} \qquad \text{verilen değerleri yerine koyarak,} \qquad (2.17)$$

$$x_k = \sum_{n=0}^{2} e^{-j\pi nk/4} + \sum_{n=6}^{7} e^{-j\pi nk/4} \qquad \text{ve} \qquad e^{-j\pi nk/4} \text{ nın periyodu 8 dir.} \qquad (2.18.a)$$

$$x_k = \sum_{n=-2}^{2} e^{-j2\pi nk/4} \qquad \text{ve geometrik seri toplam formülünü kullanarak} \qquad (2.18.b)$$

$$x_k = \frac{\sin(5k\pi/8)}{\sin(k\pi/8)} \qquad \text{elde edilir.} \qquad (2.18.c)$$

3. AYRIK ZAMAN FOURIER DONÜŞÜMLERİNİN ÖZELLİKLERİ VE HESAPLAMA YOLLARI

Ayrık zaman fourier dönüşümleri ve ayrık konvolüsyon dijital işaret işlemenin temel iki işlemidir. DFT ayrık zaman sinyallerinin açıklanmasında simgelenmesinde ve analizinde kullanılır.

3.1 İşaretin Spektrumu

Ayrık zaman işareti sonlu uzunluktaki sayılardan oluşur. Örneğin , x(n) = 1,-3,3,0,-1.3,4,-0,6. Bu sekans bir sürekli zaman işaretinin bir hafta içindeki günlük değerleri olsun. İndeks n örnekleme zamanını gösterir. Bir işaretin DFT nu onun periyodik karakterini gösterir. DFT nun her bir elemanı bir işaretin belirli bir frekanstaki değerini belirtir.

x(n) = cos (2nπ / N) olsun. Bu işaretin DFT nu :

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W^{nk} \qquad (3.1.a)$$

$W = e^{-j2\pi/N}$ kullanılarak ,

$$x(k) = \begin{cases} 1/2N & k = 1 \text{ ve } N-1 \text{ için,} \\ 0 & \text{diğer k değerleri için} \end{cases}$$

Burada sadece X(1) ve X(2) sıfırdan farklı ve değerleri 1/2N dir. Diğer elemanları ise sıfırdır. X(1) = X(N-1) ve X(2) = X(N-2) dir. Eğer işaret tek fonksiyon şeklindeyse DFT her zaman eşit çiftler halinde bulunur. X(k) = X(N-k) dır. Bu Euler bağıntısından kolayca görülebilir.

x(n) fonksiyonunun DFT nu frekans elemanlarının direkt gösterimidir. Bunada x(n) in frekans spektrumu denir. Tek bir atımın tanımı :

$$d(n) = \begin{cases} 1, & n = 0 \\ 0, & \text{diğer n'ler} \end{cases}$$

DFT nu,

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W^{nk} \qquad (3.1.b)$$

$$X(k) = \sum_{n=0}^{N-1} d(n)\, W^{nk} \qquad (3.1.c)$$

$$= 1 \text{ tüm k değerleri için.}$$

Bunun anlamı bu işaret tüm frekanslar için aynı değere sahiptir.

3.2 Ters DFT

IDFT ters ayrık zaman fourier dönüşümünü göstersin :

$x(n) = IDFT(X(k)) = IDFT(DFT(x(n)))$ ve

$$x(n) = \sum_{k=0}^{N-1} X(k)\, W^{-nk} \qquad (3.2.a)$$

Bu bağıntının geçerliliği :

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W^{nk} \qquad (3.2.b)$$

$$X(k) = \sum_{n=0}^{N-1} \left\{ (1/N) \sum_{i=0}^{N-1} X(k)\, W^{-ni} \right\} W^{nk} \qquad (3.2.c)$$

X(k) = X(k) olarak ispatlanır.

3.3 Periyodik Konvolüsyon Özelliği

Lineer dijital işaret işlemenin temel bir işlemide konvolüsyondur. DFT sıkı bir şekilde ona bağlanmıştır. Periyodik olmayan x(n) ve h(n) işaretlerinin konvolüsyonu şu şekilde tanımlanır :

$$y(n) = \sum_{m=-\infty}^{\infty} x(m)\, h(n-m) \qquad (3.3.a)$$

y(n) = x(n) * h(n) şeklinde gösterilir.
Eğer x(n) in uzunluğu N ve h(n) in uzunluğu M ise , çıkış sekansı y(n) N+M-1 uzunluğundadır. Periyodik fonksiyonun konvolüsyonu ise :

$$y(n) = \sum_{m=0}^{N-1} x(m)\, h(n-m) \qquad (3.3.b)$$

Bu durumda ise üç sekansın uzunluğuda aynıdır. x(n) , y(n) ve h(n) 0 , N-1 aralığı dışında periyodik olarak tekrarlar. Alternatif bakış için bağımsız değişken modülo N e göre hesaplanan nn olsun.

n modulo N = (n)n .

$$y((n)_n) = \sum x((m)_n)\, h((n-m)_n) \text{ ç} \qquad (3.3.c)$$

DFT nin konvolüsyon özelliği aşağıdaki bağıntıyla verilir.
DFT(y(n)) = DFT(x(n)) DFT (h(n)) (3.4.a)

Bu gösteriyorki periyodik iki fonksiyonun DFT nu herbir işaretin DFT larının çarpımına eşittir. DFT periyodik konvolüsyonu çarpma işlemine çevirir. Bu özellik konvolüsyon için gereken işlem miktarını azaltmakta kullanılır :

$$y(n) = IDFT\,(\,DFT(\,x(n)\,)\,DFT(\,h(n)\,)\,) \qquad (3.4.b)$$

3.4 x(n) ve X(k) nın periyodik özellikleri

$x(\,n\,) = x(\,n + N\,)$, bütün n değerleri için
$X(\,n\,) = X(\,k + N\,)$, bütün k değerleri için
Bu özellik modulo N e göre hesaplanan terimlerin indislenmesiyle,

$$x(\,n\,) = x\,(\,(n)_n\,) \qquad (3.5)$$

$$X(\,k\,) = X\,(\,(k)_n\,) \qquad (3.6)$$

şeklinde gösterilebilir.

3.5 Öteleme ve Modülasyon özellikleri

x(n) işareti zamanda ötelenirse X(k) spektrumu lineer faz ötelemesiyle çarpılır. Eğer,
X(k) = DFT (x(n)) ozaman

$$DFT\,(\,x(\,n + K\,)\,) = X(k)\,e^{j(2n\Pi/N)Kk} \qquad (3.7)$$

Zamanda K kadar öteleme yapılırsa lineer faz ötelemesi spektrumda (2ΠK/N)k kadardır. İşaret üstel bir değerle çarpıldığında spektrumda ötelenir.

$$DFT\,(\,x(\,n\,)\,e^{j(2n\Pi/N)Mn}) = X(\,k - M\,), \qquad (3.8)$$

e^{jx}= cos x + j sin x ile çarpım işlemine modülasyon denir ve aynı zamanda modüle edilen işaretin spektrumunu da modülasyon bölgesine öteler.

3.6 Örnekleme Özelliği

DFT orjinal işaretin örnekleme sonucunda elde edilen alt kümesinden oluşur.

X(k) = DFT (x(n)) olsun.

ve $x_s(n)$ x(n) in her K.cı değerinin örneklenmesinden elde edilen sekans olsun.

$x_s(n) = x(nK)$, $n = 0,1,2,3,...,(N/K) - 1$

N = LK kabul edilirse

$$DFT (x_s(n)) = (1/K) \sum_{n=0}^{K-1} X(k + Mn/K) \qquad (3.9)$$

Bu gösteriyorki örneklerin DFT nu ötelenmiş ve toplanmış orjinal DFT na eşittir. Bu sürekli zaman işaretlerinin fourier dönüşümlerinin ötelenme ve toplanması işlemine benzemektedir. Aynı şekilde belirsizlik X(k) ötelenmesinin çakışmasıyla meydana gelebilir.[3]

3.7 DFT ve Z dönüşümü arasındaki ilşki

Uzunluğu N olan x(n) in z dönüşümü

$$Z(x(n)) = X(z) , \qquad (3.10)$$

$$X(z) = \sum_{n=0}^{N-1} x(n) z^{-n} \qquad (3.11)$$

Bu bağıntıdan görülüyorki x(n) işaretinin DFT nu z dönüşümünden hesaplanabilir.

$z = W^{-k} = e^{-j2\pi k/n}$ olsun

$$DFT (x(n)) = X(z) \Big|_{z=W^{-k}} \qquad (3.12)$$

$= X(W^{-k})$ bulunur. Başka bir deyişle x(n)

işaretinin DFT nu z düzleminde eşit aralıkla birim daire içinde dağılan , örneklenen X(k) kümesinden oluşur.

3.8 Simetri ve gerçek eleman özelliğinden kaynaklanan bağıntılar

DFT nun tanımından

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W^{nk} \qquad (3.13.a)$$

$$= \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi nk/N} \qquad (3.13.b)$$

$$= \sum_{n=0}^{N-1} x(n) \left| \cos(2n/N)nk - j \sin(2\pi/N)nk \right| \qquad (3.13.c)$$

Bu formda X(k) nın reel ve sanal kısımları ayrı ayrı analiz edilebilir ve şunlar söylenebilir :

i) Eğer x(n) reelse X(k) nın reel kısmı çift fonksiyonsa sanal kısmı tek dir.
ii) Eğer x(n) sanalsa X(k) nın reel kısmı tek ve sanal kısmı çift dir.
Bunları terimsel olarak yazarsak :

Çift fonksiyonsa ,

$$x(n) = x(N-n) = x(-n) \qquad (3.14)$$

Tek fonksiyonsa ,

$$x(n) = -x(N-n) = -x(-n) \qquad (3.15)$$

Genlik ve spektrumun fazıda benzer şekilde :
x(n) reelse genlik X(k) çift ve faz tek dir.

Tablo 3.1 DFT parçalarının karakteristikleri

u(n) Reel(x(n))	v(n) Sanal(x(n))	U(k) Reel(X(n))	V(k) Sanal(X(n))
çift	0	çift	0
tek	0	0	tek
0	çift	0	çift
0	tek	tek	0

3.9 DFT nu hesaplama yolları

3.9.1 Goertzel Algoritmasi

Goertzel algoritması direkt metoddan biraz daha etkilidir. Bu metod $W_N^{-k/n}$ sekansının periyodik olma özelliğinin hesaplamayı nasıl azalttığını gösterir bir örnektir.

Goertzel algoritmasını elde etmek için ,

$$W_N^{-kN} = e^{j2\pi/N)Nk} = e^{j2\pi k} = 1 \quad \text{olduğunu} \qquad (3.16)$$

hatırlayalım.

$$X(k) = \sum_{r=0}^{N-1} x(r)\, W_N^{-kr} = DFT\,(\,x(r)) \quad k= 0,1,2,,,,, N-1 \qquad (3.17.a)$$

Eşitliğin sağ tarafını W^{-kN} çarparsak eşitlik bozulmaz.

$$X(k) = W_N^{-kN} \sum_{r=0}^{N-1} x(r)\, W_N^{kr} \qquad (3.17.b)$$

$$= \sum_{r=0}^{N-1} x(r)\, W_N^{-k(N-r)} \qquad (3.17.c)$$

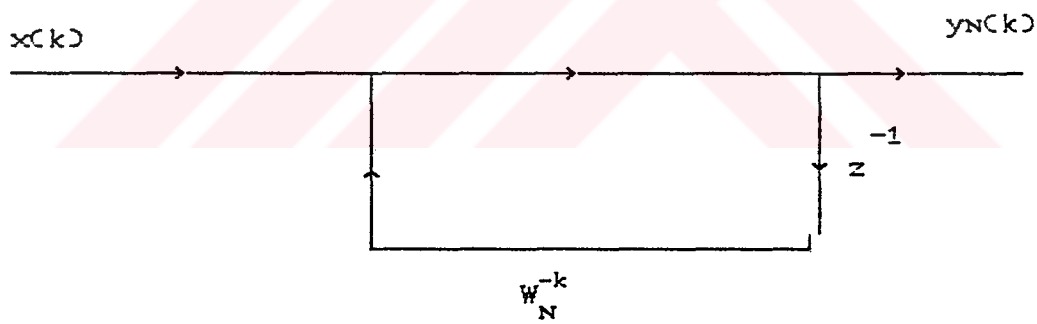
Buna uygun olarak bir y(n) tanımlayalım :

$$y(n) = \sum_{r=0}^{N-1} x(r)\, W_N^{-k(N-r)} \qquad (3.17.d)$$

Bu durumda ,

$$X(k) = y_k(n)\,|_{n=N} \text{ olur.} \qquad (3.17.e)$$

Bu eşitlik x(n) in sonlu aralık $0 \le n \le N-1$ de sekanslar W_N^{-k} ayrık konvolüsyonudur. Sonuç olarak $y_k(n)$ sistemin x(n) girişine W_N^{-kn} birim örneklemesiyle verdiği cevap olarak söylenebilir. özellike X(k) , n = N için sistemin çıkış değeridir.



Şekil 3.1X(k) nın birinci dereceden kompleks rekürsif hesaplanmasını gösteren akış diyagramı.

W_N^{-k} ve x(n) kompleks olduğu için her yeni $y_k(n)$ değerinin hesaplanması için dirt reel çarpma ve dört reel toplama işlemi yapılır. Bu durum direkt metoddan biraz daha az etkilidir. Bununla beraber şekil 3.1 de görülen

algoritmanın hesabında W_N^{-k} nın hesaplanması ve saklanması gerekmez çünkü bu katsayılar efektif olarak rekürsif program tarafından hesaplanır.

Aynı algoritmayı benzer basitlikte bırakarak hesaplama sayısını iki kat azaltabiliriz. Bu sistemin transfer fonksiyonu :

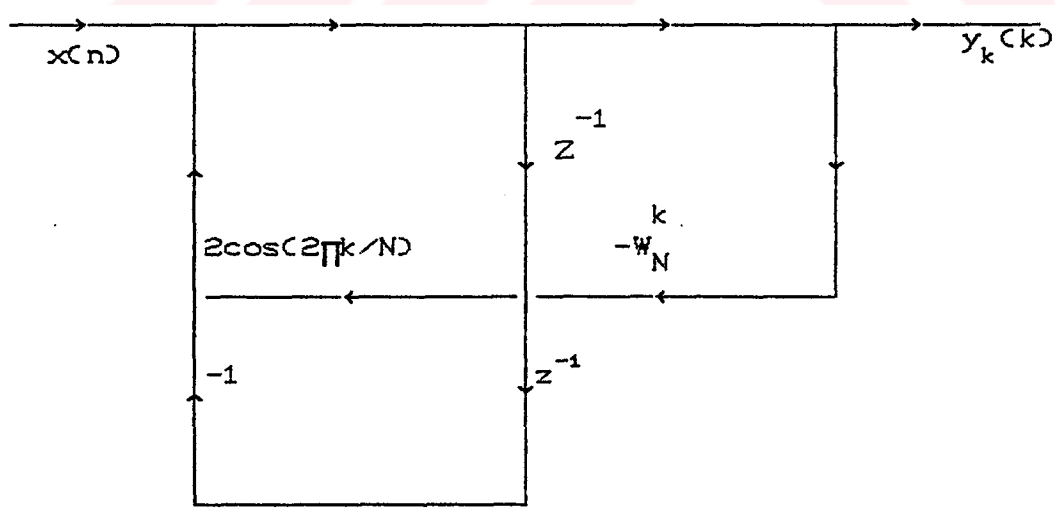
$$H_k(z) = \frac{1}{1 - W_N^{-k} z^{-1}} \quad \text{dır.} \qquad (3.18.a)$$

Bu eşitliğin $(1 - W_N^{k} z^{-1})$ ile çarpıp bölersek ,

$$H_k(z) = \frac{(1 - W_N^{k} z^{-1})}{(1 - W_N^{-k} z^{-1})(1 - W_N^{k} z^{-1})}, \qquad (3.18.b)$$

$$H_k(z) = \frac{(1 - W_N^{-k} z^{-1})}{1 - 2\cos((2\pi/N)k)\, z^{-1} + z^{-2}} \quad \text{elde} \qquad (3.18.c)$$

Bu sistemin işaret akiş diyagrami aşağıdaki gibidir.



Seki 3.2 İkinci dereceden X(k) nın rekürsif hesabı (Goertzel algoritması).

Burada katsayılar reel ve -1 olduğundan (-1 ile çarpma sayılmıyor) sadece iki tane çarpma işlemi gerekmektedir. Öncekinde olduğu gibi kutupları yerine getirmek için dört toplama işlemi gerekmektedir. Sistemi $y_k(N)$ in hesaplanabileceği duruma getirmek için N.ci itersayondan sonra sıfırları yerine getirmek için $-W_N^{-k}$ kompleks çarpımının hesaplanması gerekir. Toplam olarak 2N reel çarpma ve 4N reel toplama kutuplar için ve dört reel çarpma ve dört reel toplama sıfırlar için gerekir. Buradan toplam olarak 2(N+2) reel çarpma ve 4(N+1) reel toplama (direkt metoddaki reel işlemlerin yaklaşık yarısı) gerekmektedir. Bu algoritmanın diğer bir avantajı x(n) in z dönüşümü birim daire üzerinde konjuke ise x(k) ve X(N-k) nın hesaplanmasındadır. X(N-k) nın kutbu aynı ve sıfırları ise kompleks konjuke dir. 2N reel çarpma ve 4N reel toplama iki DFT değerinin hesaplanması demektir. Sıfırların implementasyonu sadece son iterasyonda yapılır. Böylece Goertzel Algoritması ile yaklaşık N^2 çarpma ve yaklaşık $2N^2$ toplama yapılır. Direkt metodda da işlem sayısı N^2 ile orantılıdır. Hem direkt hem de Goertzel metodunda X(k) nın tüm N değerlerinin hesaplanmasına ihtiyaç duyulmaz. M tane k değeri için işlem yapıldığını düşünürsek hesaplama NM ile orantılıdır. M in ufak olduğu durumlarda bu yollar kullanılır. Bununla beraber çok karmaşık algoritmalarda işlem sayısı $N\log_2 N$ ile orantılıdır (N 2 nin kuvvetiyse). M , $\log_2 N$ den küçükse Goertzel en iyi yoldur.

3.9.2 FFT (Fast Fourier Transforms)

DFT hesaplamasında artırmak için DFT nu daha küçük DFT larının hesabı şeklinde yazmak etkinliği dramatik ölçüde arttırır. x(n) işaretinin daha küçük parçalara bölünerek hesaplanması algoritmasına 'decimation-in-time' adı verilir. Decimation-in-time algoritmasına en uygun şekil

N in 2 nin kuvveti olması durumudur. Bu durumda X(k) iki tane N/2 lik sekansın toplamı şeklinde yazılır.

3.9.2.1 FFT (decimation-in-time)

$$N = 2^{r}$$

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{nk} \qquad , \; n = 0,1,2,3,,,,,,N-1 \qquad (3.19.a)$$

x(n) i tek ve çift sayılı noktalarına ayırırsak :

$$X(k) = \sum_{\text{çift.}} x(n)\, W_N^{nk} + \sum_{\text{tek}} x(n)\, W_N^{nk} \qquad (3.19.b)$$

çift için n = 2r ve tek için n = 2r + 1 yazarsak ,

$$X(k) = \sum_{r=0}^{(N/2)-1} x(2r)\, W_N^{2rk} + \sum_{r=0}^{(N/2)-1} x(2r+1)\, W_N^{(2r+1)k} \qquad (3.19.c)$$

$$= \sum_{n=0}^{(N/2)-1} x(2r)\, (W_N^{2})^{rk} + W_N^{k} \sum_{n=0}^{(N/2)-1} x(2r+1)\, (W_N^{2})^{rk} \qquad (3.19.d)$$

ve

$$W_N = e^{-2j(2\Pi/N)} = e^{-j2\Pi/(N/2)} = W_{N/2} \qquad (3.20.a)$$

buradan,

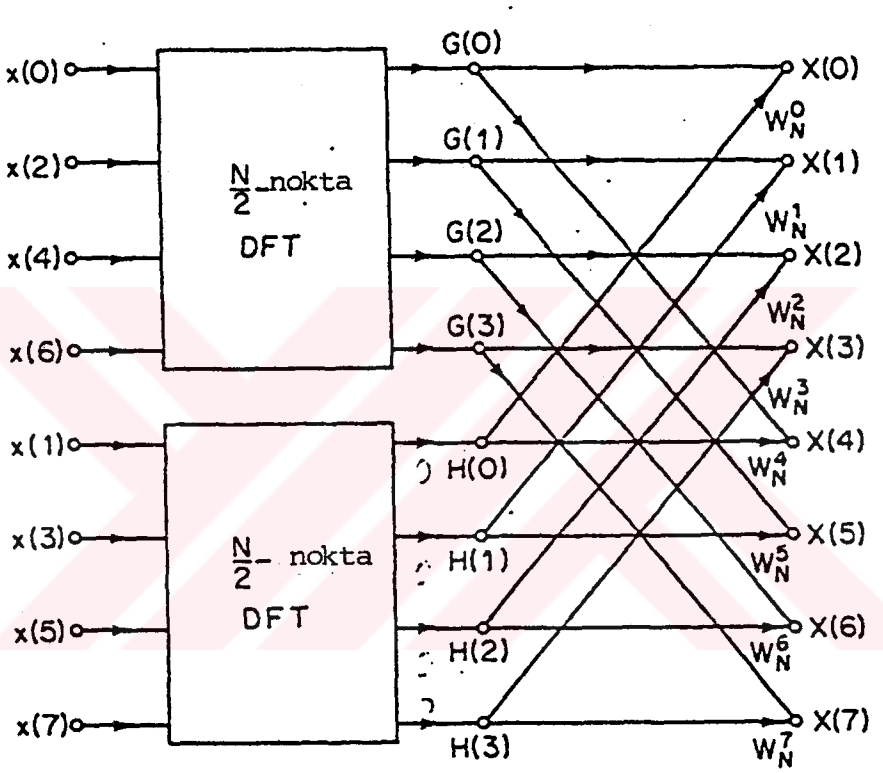
$$W_N^{2} = W_{N/2} \; . \qquad (3.20.b)$$

Sonuç olarak X(k) yı şu şekilde yazabiliriz :

$$X(k) = \sum_{r=0}^{(N/2)-1} x(2r)\, W_{N/2}^{rk} + W_N^{k} \sum_{r=0}^{(N/2)-1} x(2r+1)\, W_{N/2}^{rk} \qquad (3.21.a)$$

$$= G(k) + W_N^{k} H(k) \qquad (3.21.b)$$

Bu toplamların herbirine N/2 nokta DFT denir. İlk toplam N/2 nokta çift sayılı noktalar ikinci toplam ise N/2 nokta tek sayılı noktalardır. İndeks k 0...N-1 aralığında olsa bile toplamlar için k nın 0...N/2-1 aralığında değişmesi yeterlidir. G(k) ve H(k) 0...N/2-1 aralığında periyodiktir.[4]



Şekil 3.3 N=8 için DFT nun decimation-in-time metodu ile hesabı.

Şekil 3.3 deki akış diyagramında her noktada toplama , uzantılarındaki katsayılar çarpma işlemini göstermektedir. Katsayı yazılmamışsa değeri bir dir. DFT dört tane çift sayılı nokta G(k) ve dört tane tek sayılı nokta H(k) nın DFT larından elde edilir. Örneğin $X(1) = H(0)\, W_N^0 + G(0)$ dır. Benzer şekilde $X(1) = H(1)\, W_N^1 + G(1)$ dir.

Direkt metodda kompleks çarpımların sayısı simetri kullanılmadan N^2idi. İki N/2 nokta DFT hesabında ise $2(N/2)^2$ kompleks çarpım ve yaklaşık $2(N/2)^2$kompleks toplama işlemi gerekmektedir. Bu durumda N/2 nokta DFT ikinci toplamı W_N^k ile çarpımı için N kompleks çarpma ve N kompleks toplamada bunu ilk toplama eklemek için gerekir. Sonuç olarak DFT nin tüm k değerleri için hesabında $N + 2(N/2)^2$veya $N + (N/2)^2$kompleks çarpma ve toplama gerekmektedir. $N > 2$ için $N + N^2/2$ N^2den küçüktür.

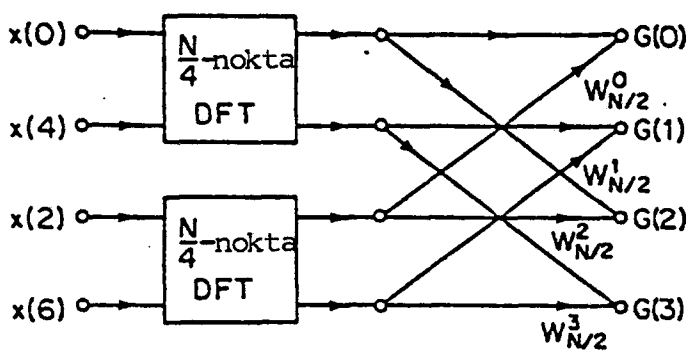
N ikinin kuvvetiyse benzer şekilde N/2 nokta DFT iki tane N/4 nokta DFT toplamı şeklinde yazılabilir. Buradan N/2 nokta DFT oluşturulur.

$$G(k) = \sum_{r=0}^{(N/2)-1} g(r)\, W_{N/2}^{rk} = \sum_{l=0}^{(N/4)-1} g(2l)\, W_{N/2}^{2lk} + \sum_{l=0}^{(N/4)-1} g(2l+1)\, W_{N/2}^{(2l+1)k} \qquad (3.22.a)$$

$$= \sum_{l=0}^{(N/4)-2} g(2l)\, W_{N/4}^{lk} + W_{N/2}^{k} \sum_{l=0}^{(N/4)-1} g(2l+1)\, W_{N/4}^{lk} \qquad (3.22.b)$$

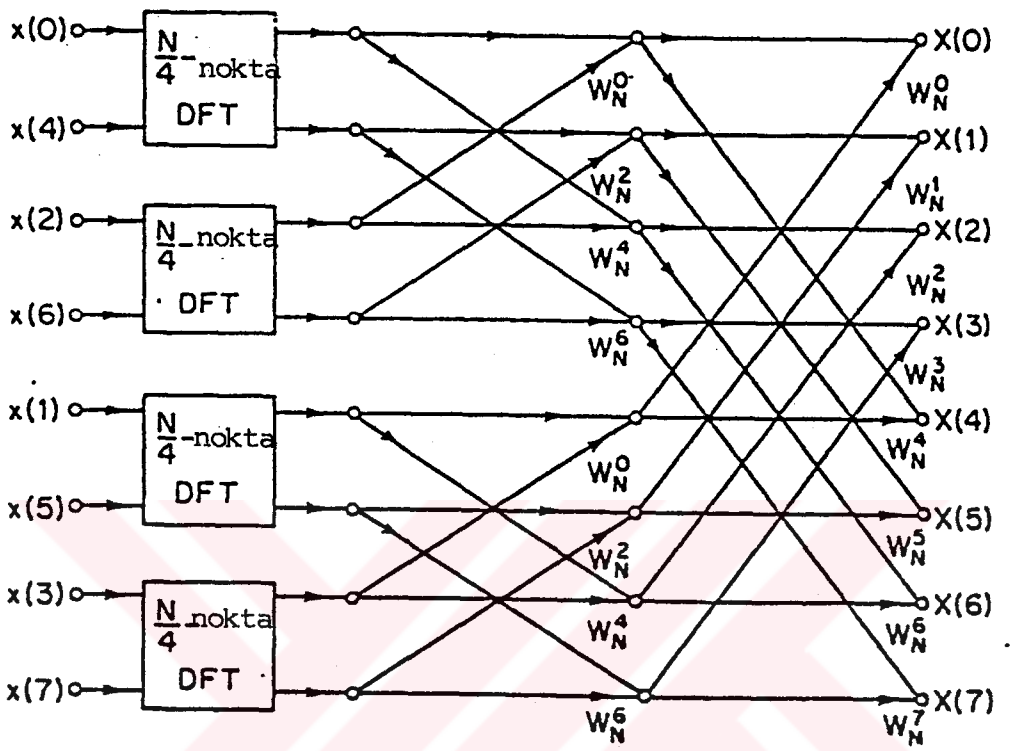
ve

$$H(k) = \sum_{l=0}^{(N/4)-1} h(2l)\, W_{N/4}^{lk} + W_{N/2}^{k} \sum_{l=0}^{(N/4)-1} h(2l+1)\, W_{N/4}^{lk} \qquad (3.22.c)$$



Şekil 3.4 (N=8 için) decimation-in-time metodu ile N/2 nokta DFT nun iki tane N/4 nokta DFT ile hesabı.

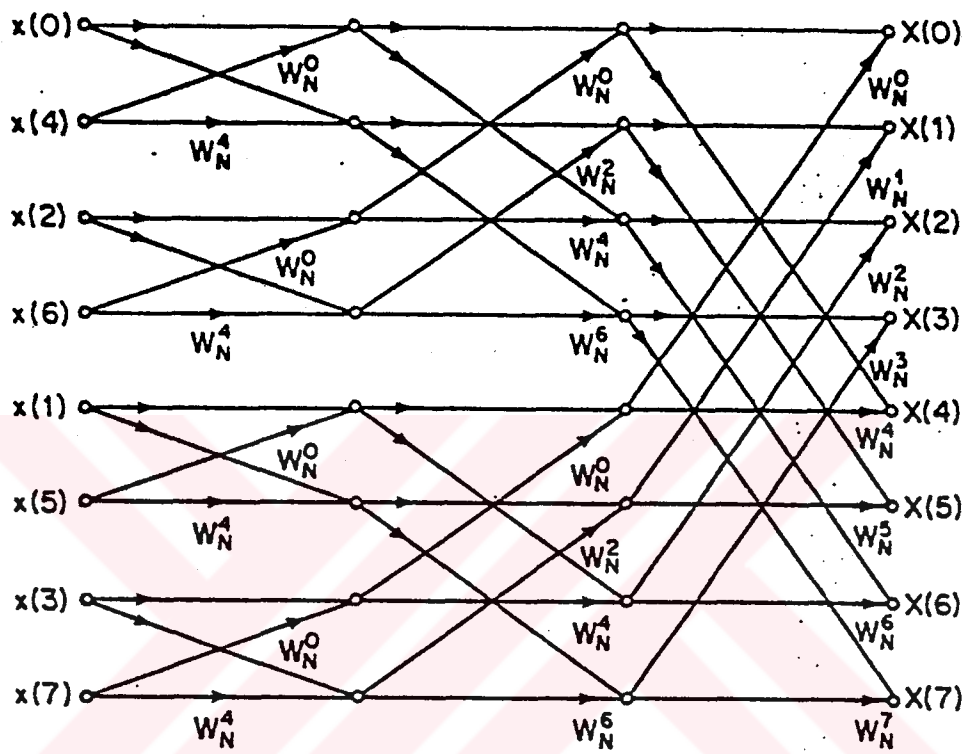
Şekil 3.4 deki yapıyı kullanarak N=8 için DFT hesabında şekil 3.3 de yerine koyarsak şekil 3.5 elde eilir.



Şekil 3.5 N/4 nokta DFT hesabının şekil 3.3 de yerine konması.

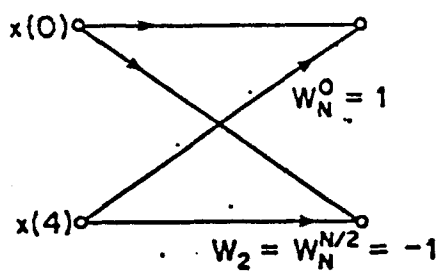
Genel olarak N ikinin kuvveti olduğunda ve üçten büyük olduğunda N/4 nokta , N/8 nokta ve iki nokta kalıncaya kadar buna devam edilir. Bu da $\log_2 N$ kısım işlem demektir. Önceki hesaplamada N nokta DFT nun N/2 nokta dönüşüme ayrıştırılmasında $N + (N/2)^2$ kompleks çarpma ve toplama gerekiyordu. N/2 nokta DTF nun iki tane N/4 noktaya arıştırılmasında N/2faktörü yarine konursa $N/2+(N/4)^2$elde edilir. Bu durumda toplam işlem sayısı $N+N+ 4(N/4)^2$kompleks çarpma ve toplama işlemi gerekmektedir. $N=2^v$ise bu işlem v= $\log_2 N$ kez yapılır.

Böylece kompleks çarpma ve toplamaların sayısı N $\log_2 N$ bulunur. Şekil 3.6 da her kısımda N çarmpa ve N toplamaya sahiptir. Toplam N $\log_2 N$kompleks çarpma ve toplama vardır.



Şekil 3.6 Sekiz nokta DFT nin decimation-in-tome metodu ile hesabındaki tüm akışını gösteren diyagram.

İki noktanın akış diyagramı ise şekil 3.7 de olduğu gibidir.

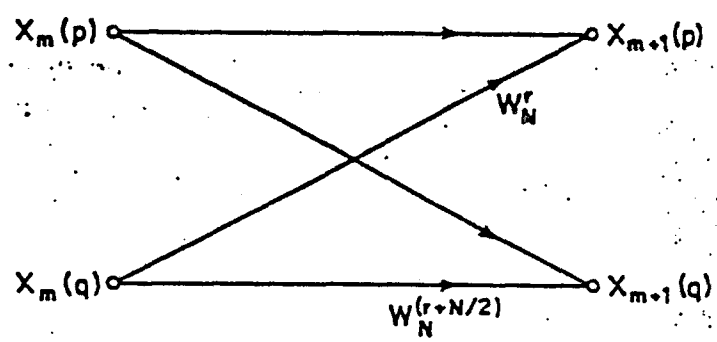


Şekil 3.7 İki nokta DFT nin akış diyagramı.

Şekil 3.6 deki akiş grafı DFT için bir algoritmayı açıklamaktadır. Bu grafta önemli lan dalların düğümlere bağlı olması ve bu dalların diğer tarafa iletilmesidir. Düğümlerin ne şekilde düzenlendiği önemli değildir. Önemli olan bunların düğümler ve ileticileri arasındahep aynı hesaplamayı gerektirmesidir. Buradaki algoritmada tek ve çift noktalar ayrılmıştır. Bunların hesabında iki tane kompleks dizi kullanıldığını varsayılırsa ilk dizi giriş işaretlerini ikincisi ise ilk kısmı tutacaktır. Kompleks sayıların şekil 3.6 daki gibi sırayla tutulduğu düşünülürse m.ci kısımdaki kompleks sayılar $X_m(l)$ (l=0,1,2,...N-1) olarak gösterilebilir. Ayrıca uygunluk açısından giriş örnekleri $X_o(l)$olsun. $X_m(l)$ giriş ve $X_{m+1}(l)$ de m"1.ci kısmın çıkışı olduğu düşünülürse N=8 için şekil 3.6da :

$X_o(0) = x(0)$
$X_o(1) = x(4)$
$X_o(2) = x(2)$
$X_o(3) = x(6)$
$X_o(4) = x(1)$
$X_o(5) = x(5)$
$X_o(6) = x(3)$
$X_o(7) = x(7)$

dir. Bu notasyonu kullanarak temel işlemi şekil 3.8 deki gibi tanımlayabiliriz.[5]



Şekil 3.8 Temel 'butterfly' hesaplaması akış diyagramı

Şekil 3.8 deki graf şöyle açıklanabilir :

$$X_{m+1}(p) = X_m(p) + W_N^r X_m(q) \qquad (3.22.a)$$

$$X_{m+1}(q) = X_m(p) + W_N^{r+N/2} X_m(q) \qquad (3.22.b)$$

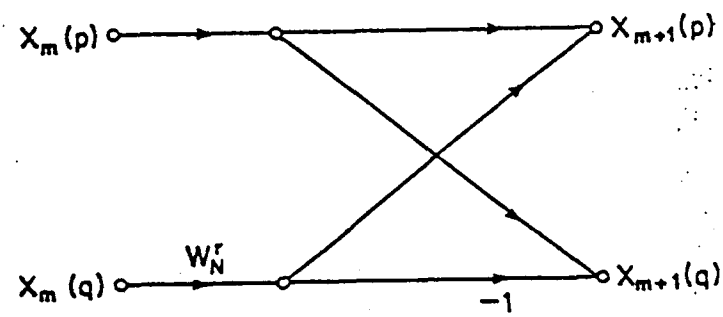
Bu eşitlik kompleks çarpımların nasıl azaltılabileceğini gösterir. Bunun için ,

$W_N^{N/2} = e^{-j(2\pi/N)N/2} = e^{-j\pi} = -1$ kullanılırsa, yukarıdaki eşitlik :

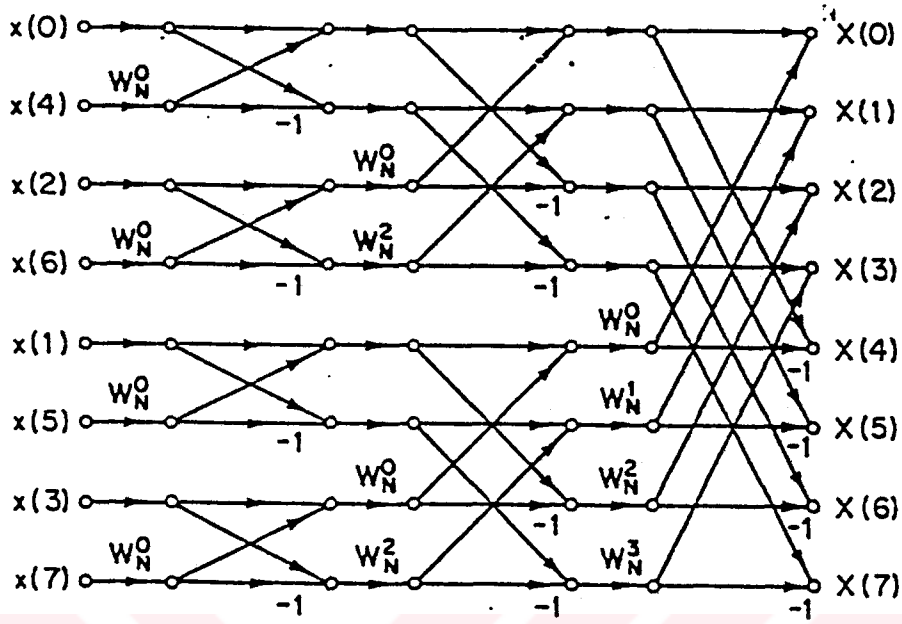
$$X_{m+1}(p) = X_m(p) + W_N^r X_m(q) \qquad (3.24.a)$$

$$X_{m+1}(q) = X_m(p) - W_N^r X_m(q) \quad \text{olur.} \qquad (3.24.b)$$

Bu eşitkiklerin akiş diyagramı şekil 3.9 da görülmektedir. Burada N/2 tane kısım vardır ve $\log_2 N$ kısım $N \log_2 N$ işlem yerine $N/2 \log_2 N$ kadar işleme ihtiyaç duyar. Bu temel kelebek grafı şekil 3.6 da yerine koyarsak şekil 3.10'u elde ederiz. Yukarıdaki eşitliklerde p,q ve r kısımdan kısıma değişmektedir. Şekil 3.9 ve 3.10 da görüldüğü gibi m+1.dizinin p ve q bölgesindeki kompleks sayılarının hesabı için m.ci bölgedeki dizinin p ve q elemanları gerekir. Böylece toplam N uzunluğunda kompleks dizi elemanı X (p) ve X (q) hesaplanmasında aynı bölgede saklanan X (p) ve X(q) ya ihtiyaç duyulur.



Şekil 3.9 İndirgenmiş butterfly (1 kompleks çarpım)

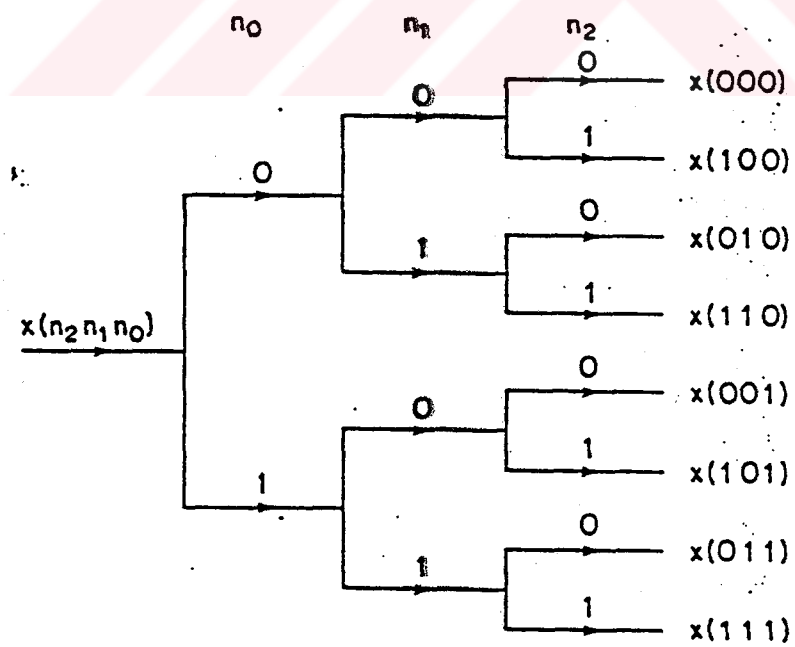


Şekil 3.10 Butterfly hesaplamasıyla 8 Nokta DFT nin akış diyagramı.

Bu tür hesaplamaya 'İn Place' hesaplama metodu denir. Avantajı yeni bir hesaplama yapıldığında orjinal veri ile aynı bölgede saklanabilmesidir. şekil 3.6 ve 3.10 in-place hesaplaması ile yapılmıştır. Burada giriş verileri orjinal sıralarında değildir (bit-reversed order). İndisleri ikili düzende gösterirsek :

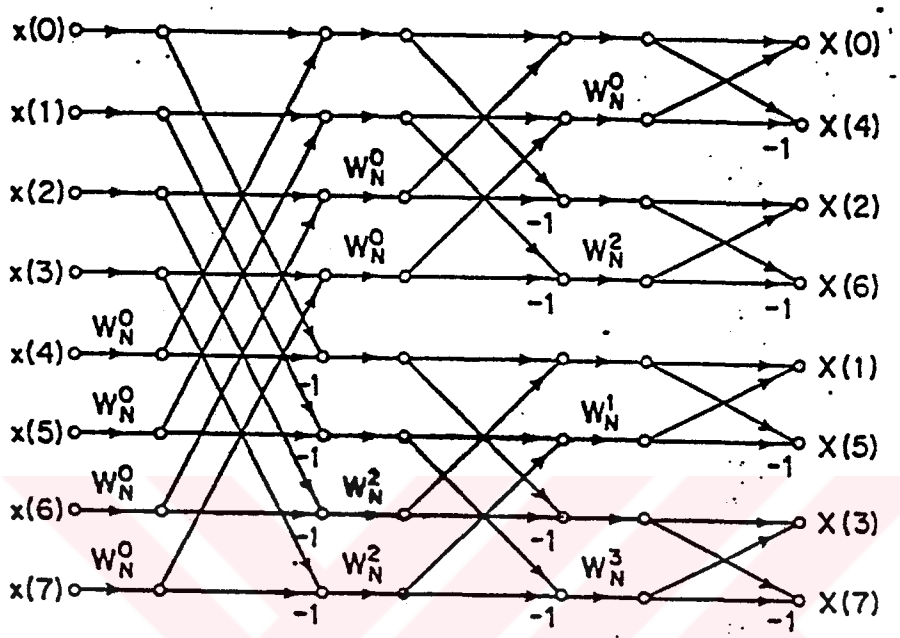
X (000) = x(000)
X (001) = x(100)
X (010) = x(010)
X (011) = x(110)
X (100) = x(001)
X (101) = x(101)
X (110) = x(011)
X (111) = x(111)
elde edilir.

x(n) sekansının $(n_2 n_1 n_0)$ ikili gösterilimiyle sekans degeri $X(n_0 n_1 n_2)$ de saklanır. $x(n_2 n_1 n_0)$ pozisyonunun hesabında indeks bitleri ters çevrilmelidir. Neden bitleri ters çevirerek hesaplama gerketiğini düşünürsek, ilk olarak x(n) in çift ve tek olarak ikiye ayrıldığını hatırlayalım. Bunu indeksteki en az anlamlı bit n_0 belirler. Sekans değeri çift örnekse $X(1)$ nin yukarıdaki kısmında , tek numaralı örnekse $X(1)$ nin alt kısmında çıkar. Sonra tek ve çift numaralı alt sekansları kendi aralarında sıralamak için ikinci en az anlamlı bit n_1 kullanılır. Çift numaralı bir sekansın ikinci en az anlamlı biti 0 sa çıkışı çift numaralı bir sekans , tekse çift numarali bir sekans elde edlilir. Bu işlemler tek numaralı sekanslar için de yapıldığında ve N kez tekrarlandığında uzunluğu 1 olan alt sekanslar elde edilir. Böylcece DFT bit-reversed sıralamayla başarılı bir şekilde daha küçük alt kısımlara ayrılmış olur. Bunun ağaç diyagramı şekil 3.11 de görülmektedir.[6]

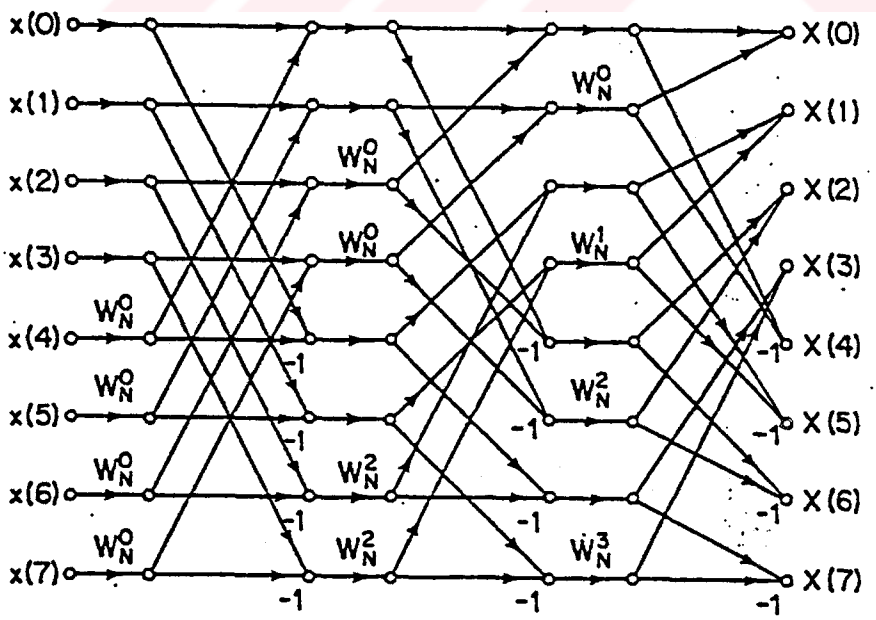


Şekil 3.11 Bit reversed sıralamayı gösteren ağaç diyagramı

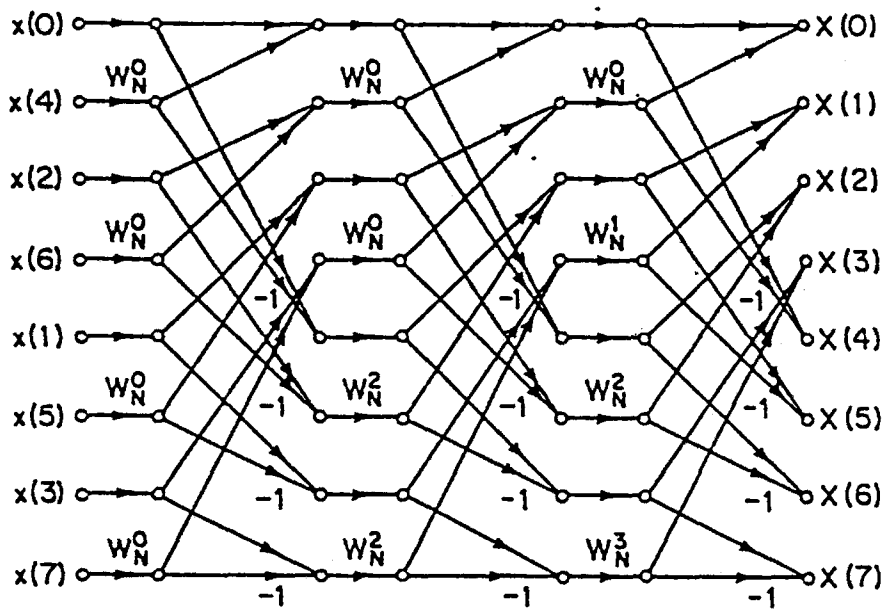
Bu grafın alternatif formları şekil 3.12 , 3.13 ve 3.14 de görülmektedir.



Şekil 3.12 x(n) normal sıralı X(n) bit-reversed



Şekil 3.13 x(n) ve X(n) normal sıralı



Şekil 3.14 Veri erişimi ve saklanması sıralı durum.

Yukarıdaki şekil RAM bellek gerektiğinde oldukça faydalıdır. En önemli özelliği graftaki her kısmın aynı geometriye sahip olmasıdır. Bu ardışıl veri erişimini mümkün kılar.

3.9.2.2 FFT (decimation-in-frequency)

Diğer bir FFT algoritması ise decimation-in-frequency dir. Decimation-in-time algoritmasında giriş sekansı x(n) alt sekanslara bölünüyordu. Burada çıkış sekansı X(k) benze şekilde alt sekanslara bölünür. Bu tipteki FFT hesaplama yoluna decimation-in-frequency denir. FFT nun decimation-in-time şeklindeki formunu üretmek için önce N in 2 nin kuvveti olduğunu varsayalım. Öncelikle giriş sekansı tek ve çift numaralı noktalar olarak ikiye ayrılır. Böylece ,

$$X(k) = \sum_{n=0}^{(N/2)-1} x(n)\, W_N^{nk} + \sum_{n=0}^{(N/2)-1} x(n)\, W_N^{nk} \qquad (3.25)$$

veya,

$$X(k) = \sum_{n=0}^{(N/2)-1} x(n)\, W_N^{rk} + W_N^{(N/2)k} \sum_{n=0}^{(N/2)-1} x(n + N/2)\, W_N^{nk} \qquad (3.26)$$

$$W_N^{(N/2)k} = (-1)^k \quad \text{bunu kullanarak,} \qquad (3.27)$$

$$X(k) = \sum_{n=0}^{(N/2)-1} \left\{ x(n) + (-1)^k x(n + N/2) \right\} W_N^{kn} \qquad (3.28)$$

elde edilir. Şimdi k nın tek ve çift olduğu noktaları X(2r) ve X(2r+1) ile simgelersek :

$$X(2r) = \sum_{n=0}^{(N/2)-1} \left\{ x(n) + x(n + N/2) \right\} W_N^{2rn} \qquad (3.29)$$

$$X(2r+1) = \sum_{n=0}^{(N/2)-1} \left\{ x(n) - x(n + N/2) \right\} W_N^{n}\, W_N^{2rn}, \qquad (3.30)$$

$r = 0,1,2 \ldots N-1$

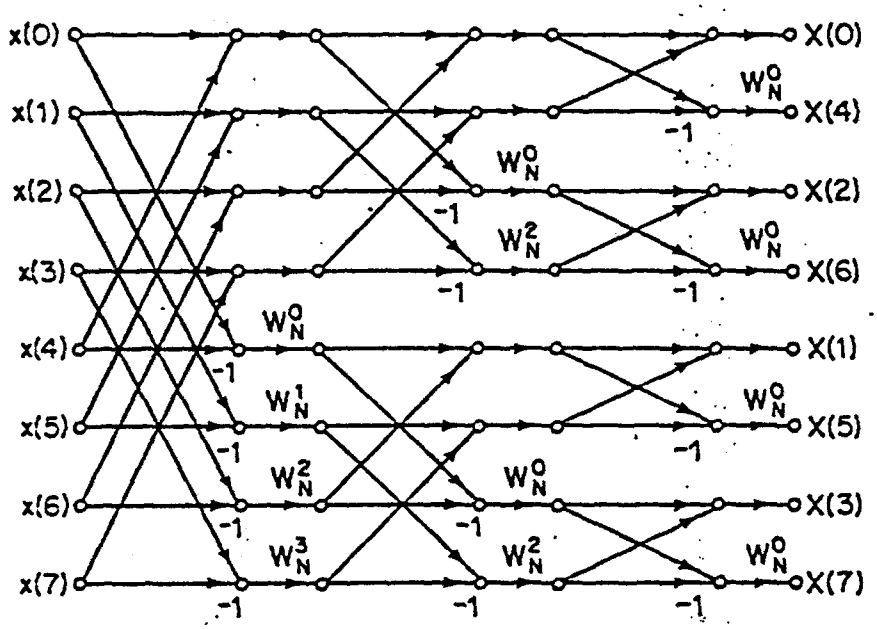
Bu iki eşitlik N/2 nokta DFT dur.

$$W_N^{2rn} = W_{N/2}^{rn}, \qquad (3.31)$$

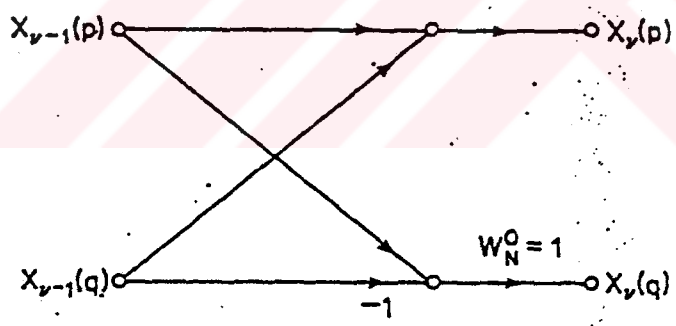
$$g(n) = x(n) + x(n + N/2) \qquad (3.32)$$

$$h(n) = x(n) - x(n + N/2) \qquad (3.33)$$

DFT $h(n)W_N^n$ ve g(n) sekanslarının hesaplanmasıyla ve bunun sonucunda iki tane N/2 lik DFT nin hesabıyla elde edilir. Bu metoda ilişkin graf şekil 3.15de görülmektedir. N ikinin kuvveti olarak düşünülmüştür. [7]



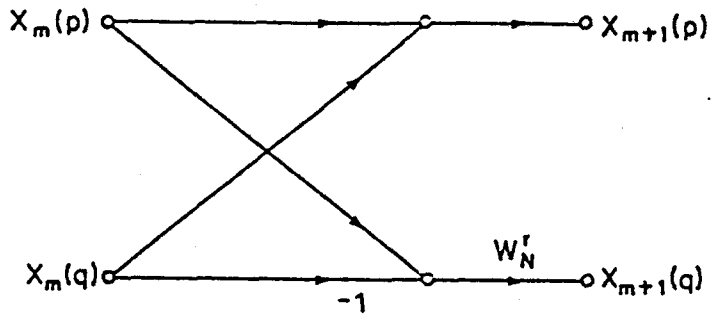
Şekil 3.17 decimation-in-frequency ayrıştırması kullanılarak sekiz nokta DFT nin tüm akış grafı.



Şekil 3.18 decimation in frequecy ayrıştırmasında son aşamada gereken iki nokta DFT .

Burada gerekli olan işlem sayısı $N = 2^r$ olma durumunda $N/2 \log_2 N$ tane kompleks çarpma ve $N \log_2 N$ komples toplamadır. Bu da decimation-in-time algoritmasıyla aynıdır.

Butterfly yapısı decimation-in-time metodundaki yapıdan daha farklıdır. Buna ait graf şekil 3.19 da görülmektedir.



Şekil 3.19 (Şekil 3.17deki) DFT nun hesabında kullanılacak butterfly graf.

Decimation-in-time hesaplamasında kullanılan eşitlikler :

$$X_{m+1}(p) = X_m(p) + W_N^r \, X_m(q) \qquad (3.34)$$

$$X_{m+1}(q) = X_m(p) - W_N^r \, X_m(q) \text{ idi.} \qquad (3.35)$$

Burada X_m ile X_{m+1}in yerini değiştirirsek ,

$$X_m(p) = 1/2 \, (\, X_{m+1}(p) + W_N \, X_{m+1}(q) \,) \qquad (3.36)$$

$$X_m(q) = 1/2 \, (\, X_{m+1}(p) - X_{m+1}(q) \,) \, W_N^{-r} \text{ olur.} \qquad (3.37)$$

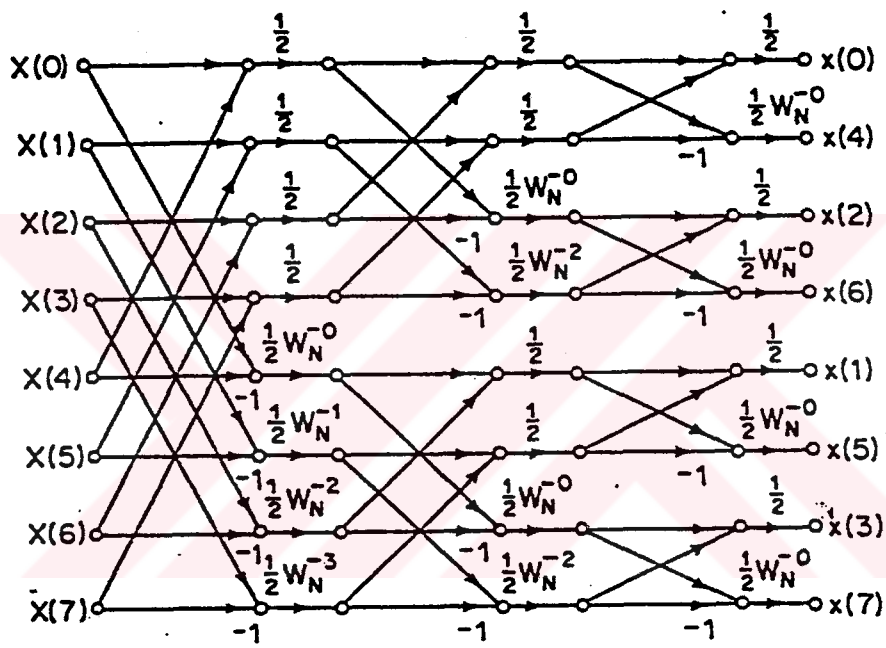
$$X_v = X(k) \qquad (3.38)$$

Bu formu kullanarak N=8 için şekil 3.20 elde edilir. $X_o(k)$ x(n) in bit reversed yapısına eşittir ve Ters FFT hesabının nasıl yapılacağını gösterir (IFFT). [8]

IFFT dönüşümü ,

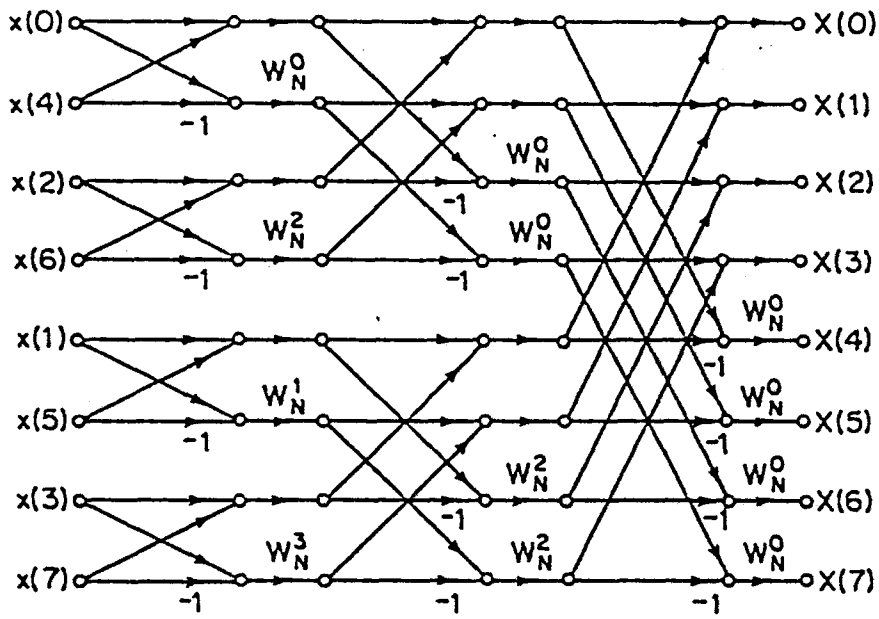
$$x(n) = 1/N \sum_{k=0}^{N-1} X(k)\, W_N^{-kn} \quad \text{ile elde edilir.} \qquad (3.39)$$

Sonuç olarak FFT algoritmasının çikişlarını N ile böler $W_N$in kuvvetleri yerine W_N^{-1}in kuvvetlerini kullanırsak Ters DFT (IDFT) nu hesaplamış oluruz.

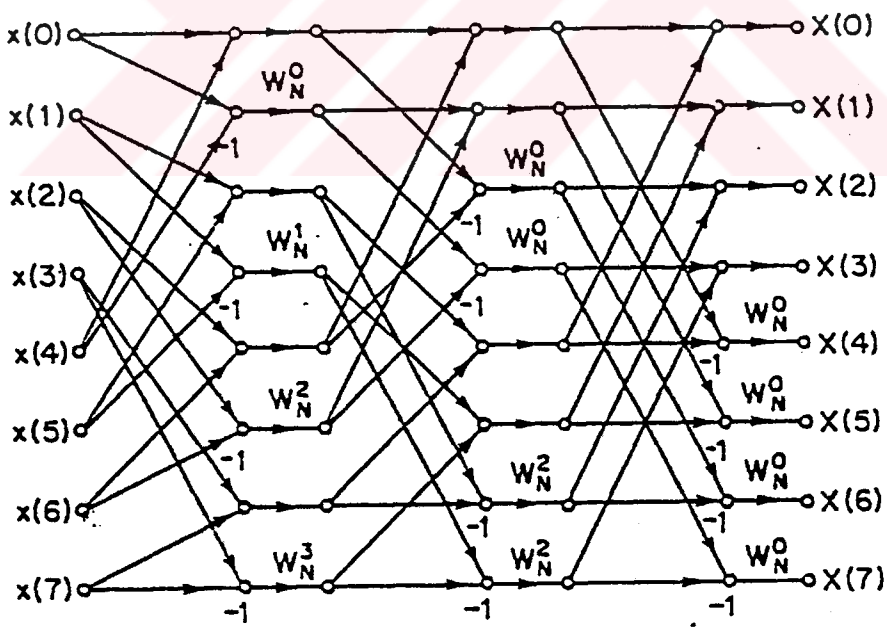


Şekil 3.20 Butterfly yapısı evrilerek IDFT nun hesaplanması.

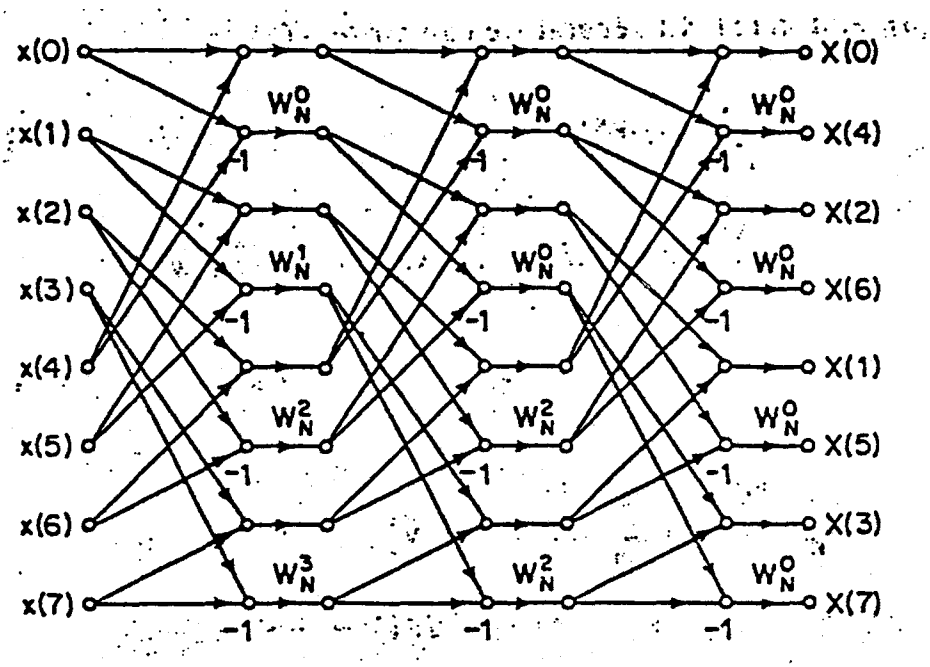
Şekil 3.20 deki IFFT algoritmasında $1/2\ W_N^r$ yerine W_N^r ile çarparak ve her aşamadaki 1/2 katsayısı çıkışı N ile çarpmaya denk olduğundan FFT algoritmasına kolayca dönüşebileceği görülmektedir. Buradan şu sonuç çıkarılır. Her decimation-in-time FFT algoritması için bir decimation-in-frequency algoritması vardır. Alternatif formları şekil 3.21, 3.22 ve 3.23 de görülmektedir. [9]



Şekil 3.21 Decimation-in-frequency DFT algoritması. Şekil 3.17 den elde edilmiştir. Giriş bit-reversed çıkış normal sıralı. Şekil 3.12 nin evriği.



Şekil 3.22 Şekil 3.17 nin yeniden düzenlenmiş hali giriş ve çikiş normal sıralı



Şekil 3.23Şekil 3.17 nin yeniden düzenlenerek ardışıl veri erişimini ve saklanmasını sağlar hale getirilmesi.

3.9.3 N çarpanlarına ayrılabilir ama 2 nin kuvveti değilse FFT hesabı

Decimation-in-time ve decimation-in-frequency FFT algoritmalarında N in özel durumu ikinin kuvveti olduğu düşünülmüştü. Şimdi N birkaç sayını çarpımı olarak alınacaktır. N= 5 9 8 gibi. $N = p_1 p_2 p_3 \ldots . p_v$ gibi sayıların çarpımıyla elde edilyor olsun.

Şimdi bir q çarpanı şöye tanımlanırsa ,

$q_1 = p_2 p_3 p_4 \ldots p_v$. Bu durumda ,

$N = p_1 \; q_1$ olur. N 2 nin kuvveti ise p_1 2 ve q N/2 seçilebilir. Benzer şekilde girişi p ve q uzunluğunda ikiye ayırabiliriz.

$p_1 = 3$ ve $q_1 = 4$ olsun. $N = 12$ dir. $x(n)$ sekasnlarını şu şekilde alt sekanslara ayırabiliriz.

1. sekans $x(0)\ x(3)\ x(6)\ x(9)$

2. sekans $x(1)\ x(4)\ x(7)\ x(10)$

3. sekans $x(2)\ x(5)\ x(8)\ x(11)$. Genel olarak $X(k)$,

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn} \quad , \qquad (3.40.a)$$

$$X(k) = \sum_{r=0}^{q_1-1} x(p_1 r)\, W_N^{p_1 rk} + \sum_{r=0}^{q_1-1} x(p_1 r + 1)\, W_N^{k} W_N^{p_1 rk} + \ldots\ldots \qquad (3.40.b)$$

$$+ \sum_{r=0}^{q_1-1} x(p_1 r + p_1 - 1)\, W_N^{(p_1-1)k} W_N^{p_1 rk} \quad \text{veya}$$

$$X(k) = \sum_{l=0}^{p_1-1} W_N^{lk} \sum_{r=0}^{q_1-1} x(p_1 r + l)\, W_{q_1}^{rk} \quad \text{şeklinde}$$

yazılabilir. (3.40.c)

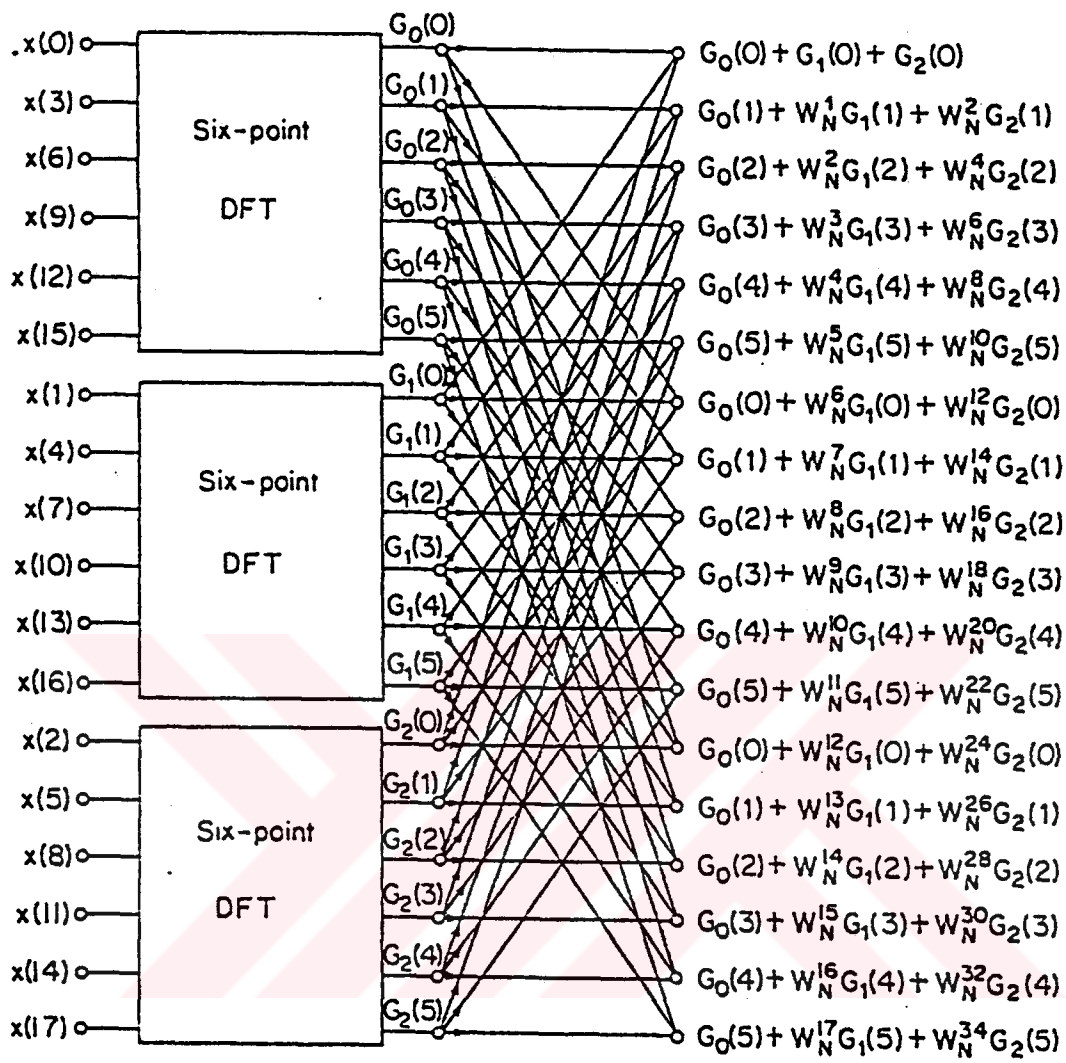
q_1 nokta DFT şu eşitlikle hesaplanır :

$$G(k) = \sum_{r=0}^{q_1-1} x(p_1 r + l)\, W_{q_1}^{rk} \qquad \text{bunun kontrolü} \qquad (3.41)$$

$$W_N^{p_1 rk} = W_{q_1}^{rk} \quad , \; N = p_1 q_1 \text{ için sağlanır.} \qquad (3.42)$$

Buradaki toplam işlem sayısı q_1 nokta DFT için $p_1 q_1^2$ kompleks çarpım ve toplamadır. Çift toplam ile gösterilen eşitlik için toplam olarak $N(p_1 - 1)$ kompleks çarpma ve toplama gerekmektedir. $p_1 q_1$ nokta DFT için ise toplam olarak $N(p_1 - 1) + p_1 q_1^2$ kompleks çarpma ve toplama gerekmektedir. q_1 nokta DFT de benze şekilde , $q_1 = p_2 q_2$ gibi ifade edilerek işlemlere devam edilir. Bu durmuda DFT için gerekli toplam kompleks çarpma ve toplamaların sayısı genel olarak :

$N(p_1 + p_2 + \ldots\ldots\ldots p_v - v)$ dir. [10]



Şekil 3.24 18 nokta DFT nin ayrıştırılmasındaki ilk aşama

N = 18 alırsak , N = 3 3 2 , $p_1 = 3$ ve $q_1 = 6$ olur.

1. sekans x(0) x(3) x(6) x(9) x(12) x(15)

2. sekans x(1) x(4) x(7) x(10) x(13) x(16)

3. sekans x(2) x(5) x(8) x(11) x(14) x(17)

herbiri 6 uzunlukta üç sekans alınır.

$$X(k) = \sum_{l=0}^{2} W_{18}^{lk} + \sum_{r=0}^{5} x(3r + 1) W_{18}^{3rk} \qquad (3.43.a)$$

$$= G_o(k) + W_{18}^{k} G_1(k) + W_{18}^{2k} G_2(k) \quad , \qquad (3.43.b)$$

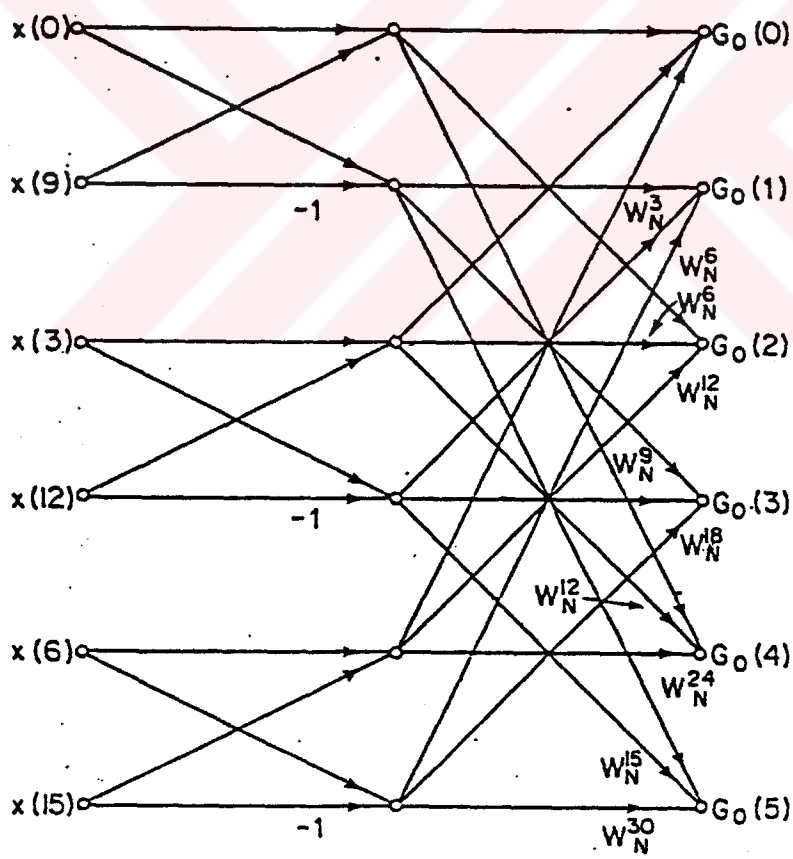
$$G(k) = \sum_{r=0}^{5} x(3r+1) W_6^{rk} = \sum_{s=0}^{2} W_6^{sk} \sum x(9p+3s+1) W_6^{3pk}$$

(3.43.c)

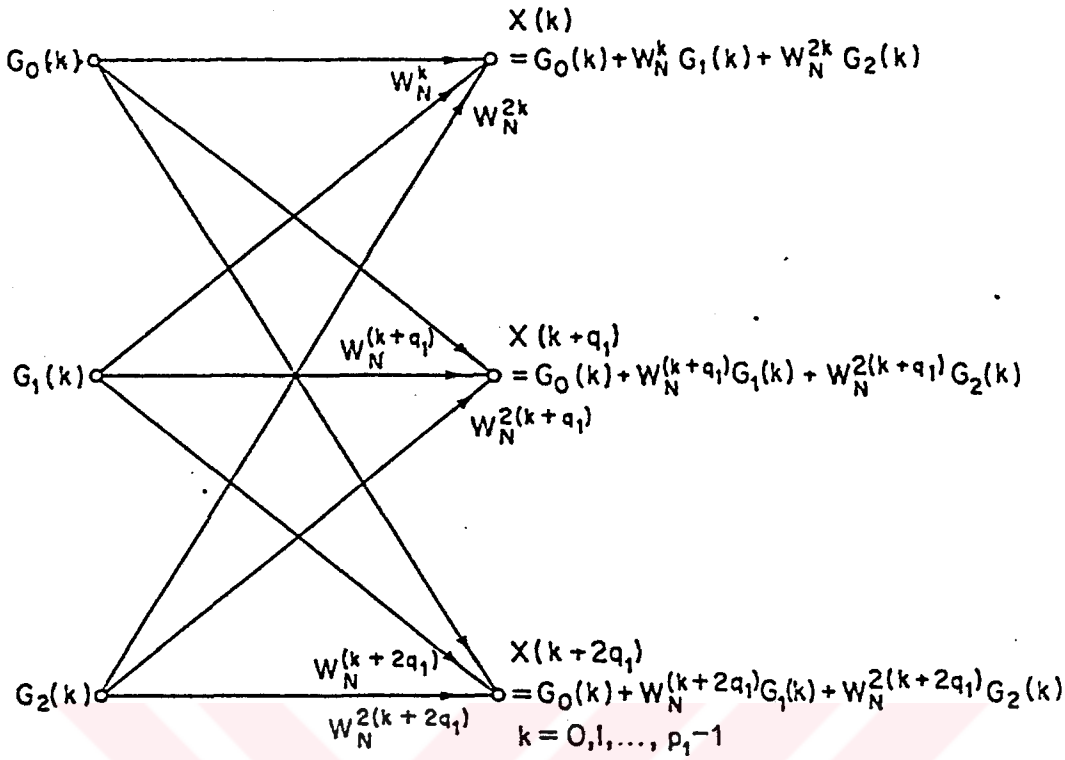
elde edilir. Bu durumda X(k) ,

$$X(k) = \sum_{l=0}^{2} W_{18}^{lk} \sum_{s=0}^{2} W_6^{sk} \sum_{p=0}^{1} x(9p+3s+1) W_2^{lk}$$ bulunur.

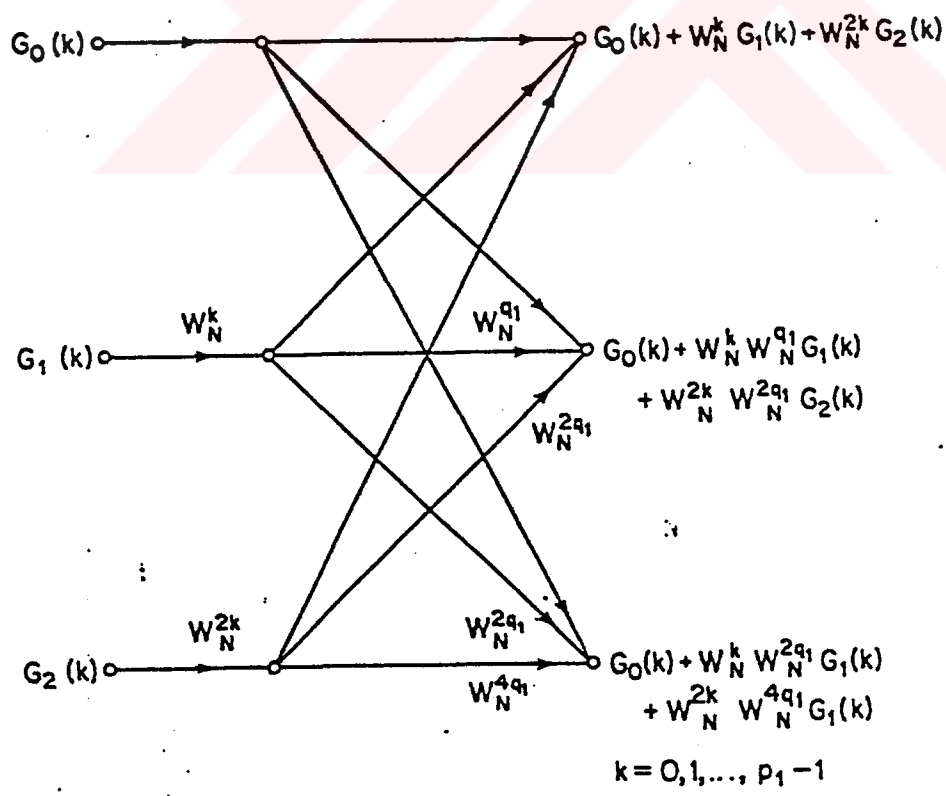
(3.43.d)



Şekil 3.25 6 nokta DFT nin ayrıştırılması.



Şekil 3.26 Üç nokta DFT nin hesabı



Şekil 3.27 Şekil 3.26 nın alternatif formu.

4. FIR FİLTRE TASARIMI

4.1 FIR Filtrelerin Özellikleri

FIR (finite duration impulse response) filtreler tümüyle stabil ve lineer fazlıdır. Genel amaçlı donanım elemanlarıyla rahatça realize edilebilirler. Tasarım metodları yüzünden rekürsif olmayan veya konvolüsyon filteresi olarak da adlandırılırlar. Filtrenin impuls cevabı sonlu olduğundan çıkış şu şekilde yazılabilir :

$$y(n) = \sum_{m=0}^{N-1} h(m)\, x(n-m) . \qquad (4.1)$$

$$y(n) = \sum_{m=n}^{n-N+1} h(n-m)\, x(m) \quad \text{yazabiliriz.} \qquad (4.2)$$

$h(n)$ = N uzunluğunda impulse cevabı , (4.3)

$x(n)$ = Giriş .

4.2 FIR Filtrelerin frekans domeninde tanımlanması

FIR filtrenin frekans domeni transfer fonksiyonu h(n) z dönüşümü olarak ,

$$H(z) = \sum_{n=0}^{N-1} h(n)\, z^{-n} , \quad z = e^{-jwn} \qquad (4.4)$$

T = 1 alındığından $(-jwnT) = -jwn$.

T = Örnekleme aralığında geçen süre.

$H(e^{jw})$ frekans cevabı.

İmpulse cevabı ayrık zaman değişkeni n in fonksiyonu olmasına rağmen , frekans cevabı 2π ile periyodiktir. [11]

$$H(w + 2\pi) = \sum h(n)\, e^{-jvn}\, e^{-jv2\pi n} = H(w). \quad (4.5)$$

$w = 2\pi f$,

DFT belirli frekanslardaki cevabı bulmak için kullanılabilir.

N uzunluğundaki impuls cevabı $h(n)$ şu şekilde tanımlanmıştır :

$$C(k) = \sum_{n=0}^{N-1} h(n)\, e^{-j2\pi nk/N} \qquad k = 0, 1, \ldots N-1 \quad (4.6.a)$$

$$C(k) = H(w)\Big|_{v = 2\pi k/N} = H(2\pi k / N) \quad (4.6.b)$$

Buradan $h(n)$ in DFT nu frekans cevabından N adet örneği verir. N nokta frekans cevabının tanımlanmasında yeterli olmayabilir. Bu durumda L uzunluğundaki DFT yi bulmak için basit bir şekilde L - N tane sıfır eklenerek frekans cevabı daha belirgin olarak elde edilir. L sonsuz yapılırsa DFT aynı zamanda $h(n)$ in fourier dönüşümü olur.

FIR filtreler özellikle Lineer Faz ötelemelidir. Faz kaydırma şöyle tanımlanabilir :

$$H(w) = R(w) + j\, I(w) \quad (4.7)$$

Genlik ve Faz

$$M(w) = |H(w)| = SQR(R^2 + I^2), \quad (4.8)$$

$$d(w) = \arctan(I / R), \quad (4.9)$$

Buradan ,

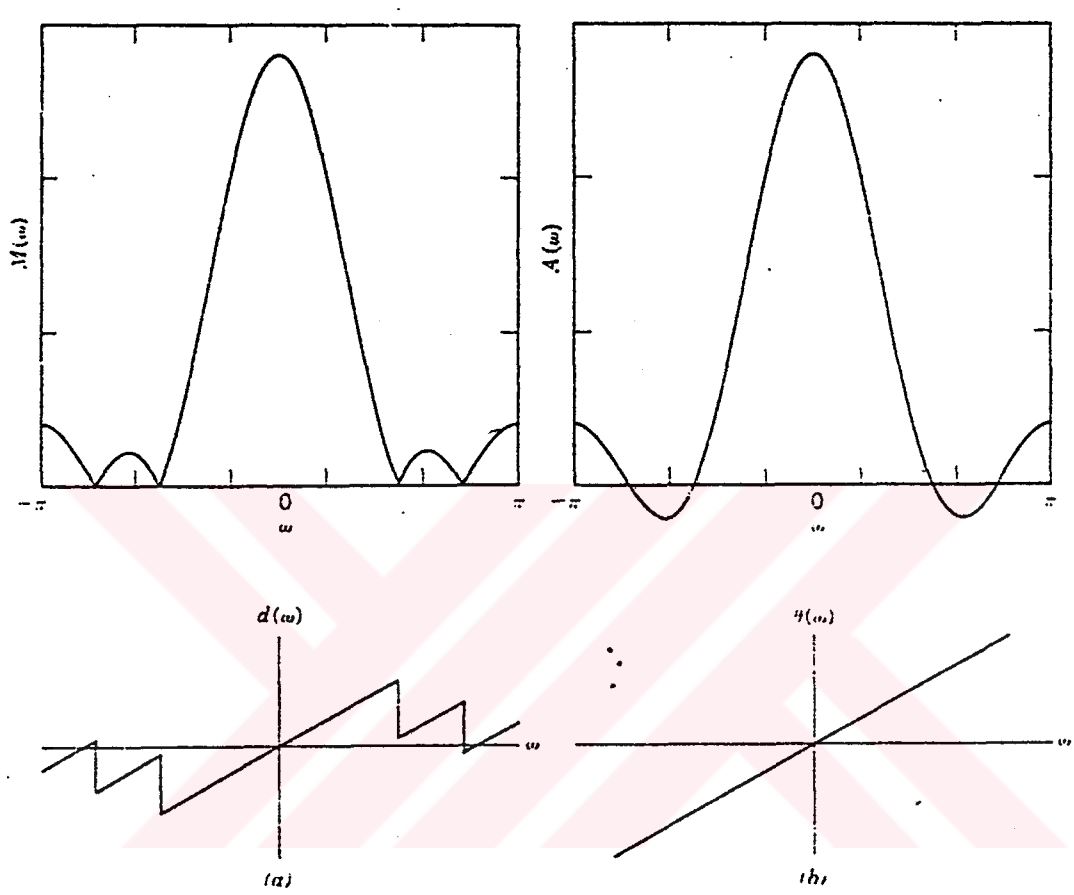
$H(w) = M(w)\, e^{-jd(w)}$ olarak yazılabilir.

$M(w)$ analitik değildir ve $d(w)$ sürekli değildir.

$A(w) = -/+ M(w)$ mutlak değeri alınırsa

$H(w) = A(w)\, e^{j\theta(w)}$ olarak yazılabilir.

Bu tanımlamayla A(w) analitik ve θ(w) sürekli kılınır.



Şekil 4.1 Genlik ve fazın sürekli kılınması
a= süreksiz , b= sürekli

Faz fonksiyonunu genel olarak

$$\theta(w) = K_1 + K_2 w \quad \text{olarak yazarsak,} \qquad (4.10)$$

N uzunluğundaki filtrenin frekans cevabı

$$H(w) = A(w)\, e^{j(K_1 + K_2 w)} \quad \text{olur.} \qquad (4.11)$$

M = (N - 1) / 2 veya M = N - M - 1 alınarak

$K_1 = 0$ ve $K_2 = \pi / 2$ alınırsa İlk durum için

tek simetri vardır .

$h(n) = h(N - n - 1)$ buradan

$H(w) = A(w)\, e^{-jMw}$ (4.12)

e^{-jMw} lineer fazı verir.

N tek ise

$$A(w) = \sum_{n=0}^{M-1} 2h(n) \cos(w(M - n)) + h(M) \quad (4.13.a)$$

$$A(w) = \sum_{n=1}^{M} 2h(M - n) \cos(wn) + h(M) \quad (4.13.b)$$

N çift ise

$$A(w) = \sum_{n=0}^{N/2-1} 2h(n) \cos(w(M - n)) \quad (4.13.c)$$

$$A(w) = \sum_{n=1}^{N/2} 2h(N/2-n) \cos(w(n - 1/2)) \quad (4.13.d)$$

$K_1 = \pi/2$ ise çift simetri vardır.

N tekse,

$h(n) = - h(N - n - 1)$ dir. (4.14)

$H(w) = j\, A(w)\, e^{-jMw}$ dır. (4.15)

$$A(w) = \sum_{n=0}^{M-1} 2h(n) \sin(w(M-n)) \quad (4.16)$$

N çiftse,

$$A(w) = \sum_{n=0}^{N/2-1} 2h(n) \sin(w(M-n)) \text{ dir.} \quad (4.17)$$

4.3 FIR Filtrenin Dört Tipi

1. Tip :

İmpuls cevabı tek uzunlukta ve n = M = (N-1/2) civarında çift simetrik.

2. Tip :

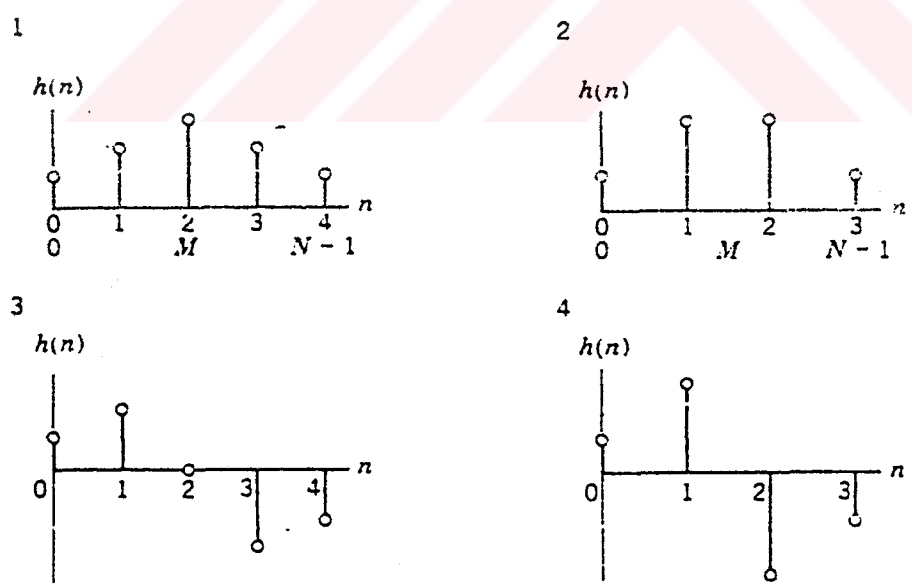
İmpuls cevabı çift uzunlukta ve M civarında tek simetrik ancak herhangibir h(n) simetri ekseninde yok.

3. Tip :

İmpuls cevabı tek uzunlukta ve tek simetrik.

4. Tip :

İmpuls cevanı çift uzunlukta ve tek simetrik.



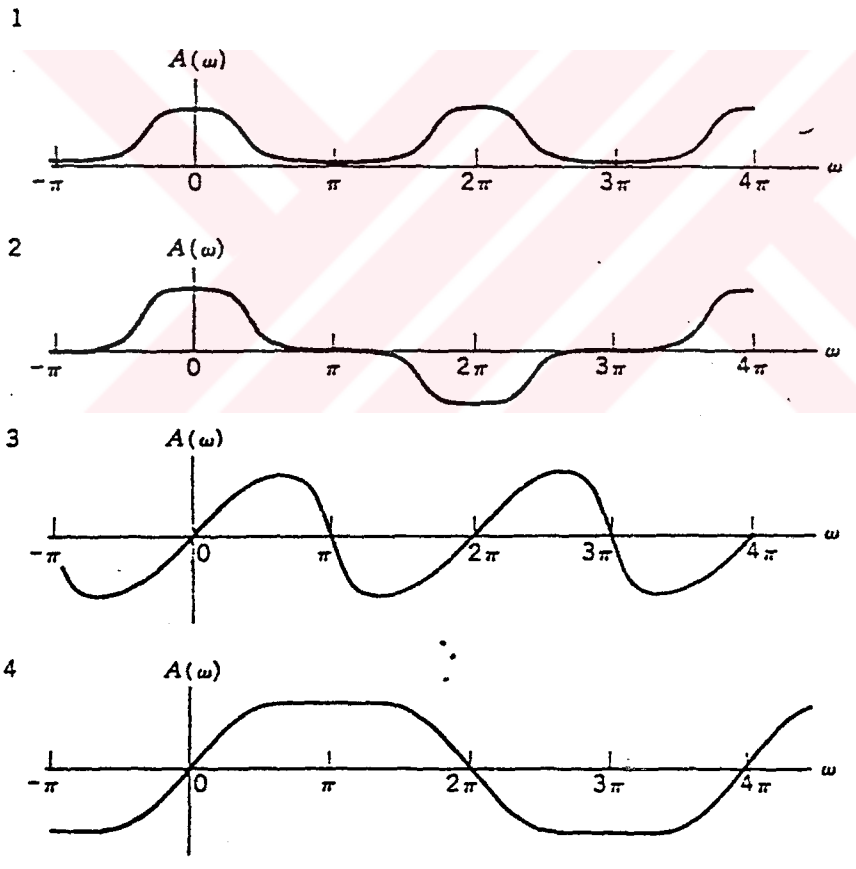
Şekil 4.2 FIR filtrenin 4 değişik tipi.

$A(\omega) = A(-\omega)$
$A(\pi + \omega) = A(\pi - \omega)$
$A(\omega + 2\pi) = A(\omega)$

$A(\omega) = A(-\omega)$
$A(\pi + \omega) = -A(\pi - \omega)$
$A(\omega + 4\pi) = A(\omega)$

$A(\omega) = -A(-\omega)$
$A(\pi + \omega) = -A(\pi - \omega)$
$A(\omega + 2\pi) = A(\omega)$

$A(\omega) = -A(-\omega)$
$A(\pi + \omega) = A(\pi - \omega)$
$A(\omega + 4\pi) = A(\omega)$



4.3 Dört tip filtrenin genlik fonksiyonları.

4.4 FIR Filtrenin Frekans Cevabının Hesaplanması

1. Tip :

$$A_k = A\left(\frac{2\pi k}{L}\right) = \sum_{n=0}^{M-1} 2h(n)\cos\left(\frac{2\pi(M-n)k}{L}\right) + h(M) \qquad (4.18)$$

2. Tip :

$$A_k = A\left(\frac{2\pi k}{L}\right) = \sum_{n=0}^{N/2-1} 2h(n)\cos\left(\frac{2\pi(M-n)k}{L}\right) \qquad (4.19)$$

3. Tip :

$$A_k = A\left(\frac{2\pi k}{L}\right) = \sum_{n=0}^{M-1} 2h(n)\sin\left(\frac{2\pi(M-n)k}{L}\right) \qquad (4.20)$$

4. Tip :

$$A_k = A\left(\frac{2\pi k}{L}\right) = \sum_{n=0}^{N/2-1} 2h(n)\sin\left(\frac{2\pi(M-n)k}{L}\right) \qquad (4.21)$$

Tüm durumlarda $M = (N-1)/2$, ortalama zaman gecikmesi verir. [12]

5 LİNEER FAZ FIR FILTRE TASARIMI

Temel olarak FIR filtre tasarımında şu adımlar izlenir :

1. Frekans domeninde ideal cevabın tanımlanması
2. N uzunluğunda FIR filtrenin seçilmesi
3. Filtrenin cevabının bir kritere göre arzulanan cevapla kıyaslanması
4. Kullanılan kritere göre FIR filtrenin seçilme işlemi için bir metod geliştirme.

İteratif olarak bu işlemlerin tekrarıyla FIR filtrenin arzu edilen cevaba yakınsaması.

FIR filtrelerin avantajları kısaca şöyle özetlenebilir.

1. Katsayıları 1 e kolayca yakınsarlar ve asla değişmezler oysa bir kapasitenin sıcaklıkla değişimi mümkündür.

2. Kompleks , farklı band ve farklı kazanclarda , analog tekniklerle yapılamayan , filtreler tasarlanabilir.

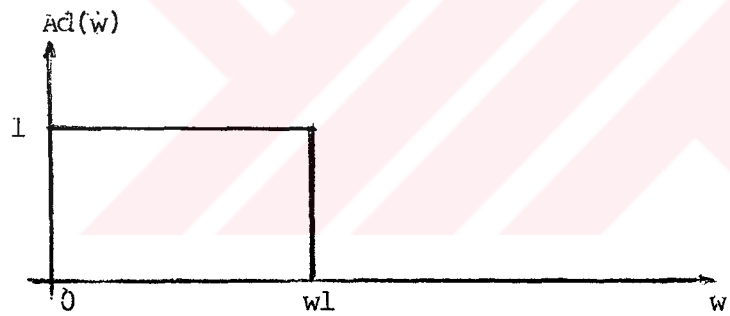
3. -50 - -60 dB kazanca durdurma bandında analog tekniklerdekinden daha çabuk ulaşılır.

4. FIR filtreler tümüyle stabildir ve lineer fazlıdır.

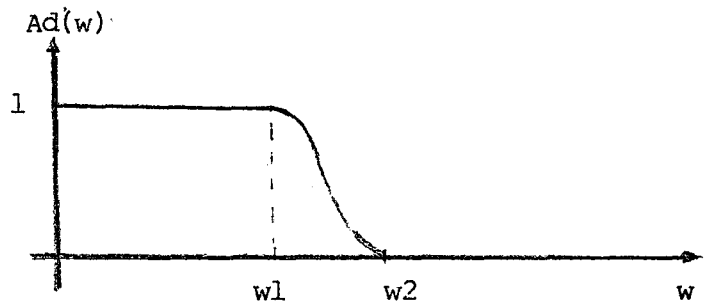
5. Geçirme bandında aşım < %1 olacak şekilde tasarlanabilirler.

5.1 İdeal Alçak geçiren filtrenin tasarımı

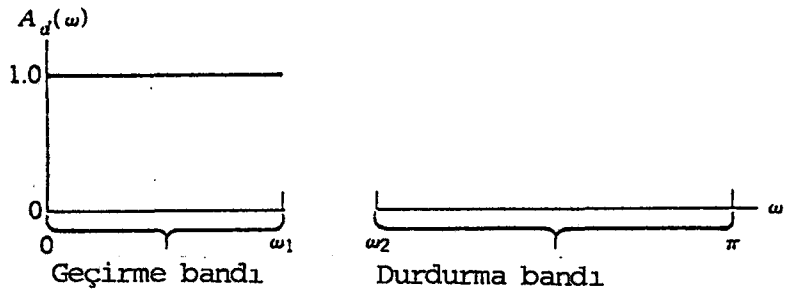
Alçakgeçiren filtrenin ideal cevabı w = 0 , w = w_1 arsındaki işaretleri geçirmesi ve w = w_1 ile w = π Nyquist frekansı arasındaki işaretleri geçirmemesidir. (Şek 5.1) Bazı durumlarda geçiş bandı ile durdurma bandı arasında bir geçiş bölgesi (arzulanan yada arzulanmayan işaretlerin bulunduğu) var olabilir veya bu bölge istenen ve istenmeyen işaretlerin çakıştığı bir bölge olarak düşünülebilir. Bu bölgede geçiş ve durdurma bandları arasında yumuşak bir geçiş vardır. Bu şekilde daha iyi bir yakınsama sağlanabilir.(Şek 5.2) Üçüncü olasılık ise geçiş bölgesinin yakınsamasının tanımlanmamasıdır.(Şek 5.3)



Şek 5.1 İdeal Alçakgeçiren Filtre Frekans Cevabı



Şek 5.2 Geçiş bölgesi olan Alçakgeçiren filtrenin frekans cevabı



Şek 5.3 Geçiş bölgesi yakınsaması tanımlanmamış Alçakgeçiren filtrenin frekans cevabı

Yakınsama Kriteri : Hata ölçümü genellikle FIR filtre tasarımında kullanılır. Birincisi frekans cevabına yakınsamadaki hatanın ortalama karesidir. İkincisi ise frekans cevabında belirtilen bölgeler için maksimum hatanın gözönüne alınmasıdır. Üçüncü yaklaşım ise arzu edien cevabın Taylor serisi yaklaşığının alınmasıdır.
1.ci yaklaşim Least Squared Approximation LS olarak
2.ci yaklaşım Chebyshev yakınsaması ,
3.cü yaklaşım Butterworth veya maximally flat yakınsaması olarak adlandırılır. [13]

Tasarım metodlarını irdelersek ,
1. yaklaşım : Frekans Örnekleme Metodu hızlı ve basittir. Ama frekans cevabı üzerinde kontrol çok azdır.
2. yaklaşım : LS hata metodunda kullanılan hata fonksiyonu işaretin veya gürültünün enerjisiyle bağıntılı olmasıdır.Bazen frekans cevabı osilasyonlu olmakla bereber pencereleme , geçiş bölgesi ve ağırlık fonksiyonu kullanılarak istenmeyen bu durumlar kontrol edilebilir.
3. yaklaşım : ParksMcCellan algoritması Chebyshev hatasını minimize eder ama tasarım algoritması biraz daha yavaştır. Bu metodlar bir sonraki bölümde tek tek ele alaınacaktır.

5.2 FIR Filtre Tasarımı Metodlarının İncelenmesi

5.2.1 Frekans Örnekleme Metodu

Bir önceki bölümde freans cevabını filtre katsayılarından yola çıkarak nasıl hesaplandığını incelemiştik. Frekans cevabından yola çıarak seçilen N adet örnek yardımıyla N adet filtre katsayısı C(k) = H(w)| adet eşitliğin $w=2\pi k/N$ çözülmesiyle bulunabilir. Filtrenin frekans cevabı mutlaka arzu edilen noktalardan geçecektir. N tane eşitliği direkt çözmek genellikle istenmeyen bir durumdur. Yaklaşık olarak N^3adet aritmetik işlem gerektirir. Frekans örnekleme eşit zaman aralıklarıyla yapılıyorsa DFT kullanılabilir. İmpuls cevabının DFT nu frekans cevabının örneklerini verir. Frekans örneklerinin ters DFT nu (IDFT) filtrenin impuls cevabını verir. Burada genel olaral N^2aritmetik işlem yapılır. Eğer FFT kullanılacak olursa N log N adet işlem yeterli olur. Frekans örnekleme metodunda IDFT kullanarak h(n) filtre katsayıları kolayca hesaplanabilir.

$$h(n) = 1/N \sum_{k=0}^{N-1} C_k \, e^{-j2\pi nk/N} \qquad (5.1)$$

Eğer filtrenin frekans cevabı H(w) lineer fazlı ise dördüncü bölümde verilen dört değişik filtre tipi için şu sadeleştirmeler yapılabilir :

1.Tip için :

N tek,

L = N,

M = (N - 1) / 2 ve w = 0 da 1 frekans örneği

alınarak ,

$$A_k = \sum_{n=0}^{N-1} 2h(n) \cos (2\pi k(M-N) / N) + h(M) \quad (5.2)$$

A(w) yı kullanarak IDFT nunu alırsak

$$h(n) = 1/N \sum_{k=0}^{N-1} e^{-j2\pi Mk/N} A_k e^{j2\pi nk/N} , \quad (5.3.a)$$

$$= 1/N \sum_{k=0}^{N-1} A_k e^{j2\pi (n-M)k/N} \text{ elde edilir.} \quad (5.3.b)$$

h(n) reel oldığundan $A_k = A_{N-k}$ dır. Bunu gözönüne alırsak,

$$H(n) = 1/N (A_o + \sum_{k=1}^{M} 2 A_k \cos (2\pi (n-M)k/N)) \text{ olur.} \quad (5.4)$$

Sadece h(n) in M noktasındaki değeri simetri yüzünden hesaplanması yeterlidir. Bu formülle impuls cevabı h(n) frekans cevabı örnekleri A_k yardımıyla bulunur. Gereken aritmetik işlem sayısı N^2 yerine M^2 dir.

2. Tip

N çift ise,

$$A_k = \sum_{n=0}^{N/2-1} 2h(n) \cos (2\pi (M-n)k /N) \text{ dir.} \quad (5.5)$$

tasarım formülü ,

$$h(n) = 1/N (A_o + \sum_{k=1}^{N/2-1} 2A_k \cos (2\pi (n-M)k/N) \quad (5.6)$$

N çift ve $A_{N/2} = 0$ dır.

Bu durum frekans örneklerinin $w = 2\pi k/N$ $k=0,1...,N-1$ şeklinde alınmasıyla elde edilir. N eşit aralıklı örnek $w = 0$ dan başlanarak alınır Diğer durumda ise $w = (2k+1)\pi / N$ $k= 0,1,... N-1$ ancak $w = 0$ da örnek alınmaz. Bu tip frekans örneklemenin DFT ile ilişkisini

kurmak daha zordur fakat A_kyı öteleyrek ve 2N DFT nunu alarak mümkün olur.

Tek ve çift uzunluk iki tipi , sıfırda örnek ve sıfır frekansında örnek almamak da diğer iki durumu belirlediğinden önceki bölümde bahsedilen 1.ci ve 2.ci tipteki FIR filtrelerin dört değişik frekans örnekleme metoduyla tasarımını elde ederiz.

Tek uzunluk ve sıfır da örnek olmazsa ,

$$A_k = \sum_{n=0}^{M-1} 2h(n) \cos\left(2\pi (M-N)(k+1/2) / N \right) + h(M) \qquad (5.7)$$

$$h(n) = 1/N \left(\sum_{k=0}^{M-1} 2A_k \cos(2\pi(n-M)(k+1/2) + A_M \cos(\pi(n-M)) \right) \qquad (5.8)$$

Dördüncü durum Çift uzunluk ve sıfırda ve w=0 da örnek olmazsa ,

$$A_k = \sum_{n=0}^{N-1} 2h(n) \cos\left(2\pi (M-n)(k+1/2) / N \right) \qquad (5.9)$$

$$h(n) = 1/N \left(\sum_{k=0}^{M-1} 2A \cos(2\pi(n-M)(k+1/2)/N \right) \qquad (5.10)$$

Çift simetrik filtreler 3.cü ve 4.tip için benzer ilişkileri kullanarak ,

3.tip için,

$$A_k = \sum_{n=0}^{M-1} 2h(n) \sin\left(2\pi (M-n)k/ N \right) \qquad (5.11)$$

$$h(n) = 1/N \left(\sum_{k=1}^{M} 2 A_k \sin \left(2\pi (M-n) k / N \right) \right) \qquad (5.12)$$

4.tip için ,

$$A_k = \sum_{n=0}^{N/2-1} 2h(n) \sin \left(2\pi (M-n) k / N \right) \qquad (5.13)$$

$$h(n) = 1/N \left(\left(\sum_{k=1}^{N/2-1} 2A_k \sin(2\pi(M-n)k / N \right) + A_{N/2} \sin(\pi(M-n)) \right) \qquad (5.14)$$

1.ci ve 2.ci tip için kullanılan frekans örnekleme yapısını kullanarak ,

3. tip için eşitlikler ,

$$A_k = \sum_{n=0}^{M-1} 2h(n) \sin \left(2\pi (M-n) (k+1/2) / N \right) \qquad (5.15)$$

$$h(n) = 1/N \left(\sum_{k=0}^{M-1} 2 A_k \sin \left(2\pi (M-n) (k+1/2) / N \right) \right) \qquad (5.16)$$

4. tip için eşitlikler ,

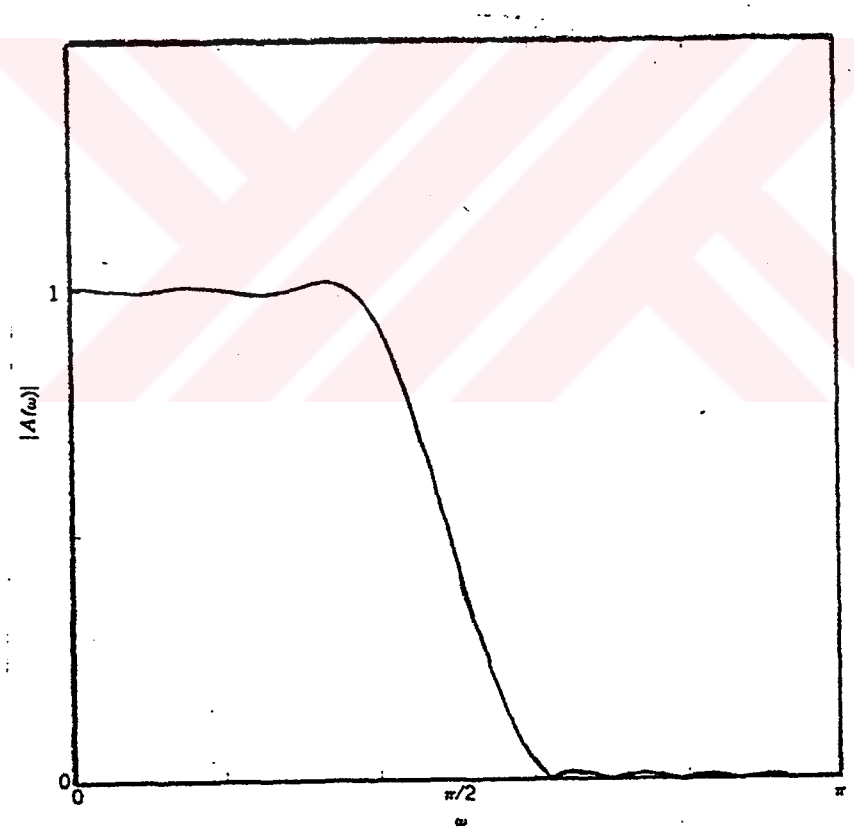
$$A_k = \sum_{n=0}^{N/2-1} 2h(n) \sin \left(2\pi (M-n) (k+1/2) / N \right) \qquad (5.17)$$

$$h(n) = 1/N \left(\left(\sum_{k=0}^{N/2-1} 2A_k \sin(2\pi(M-n)(k+1/2) / N \right) \right) \qquad (5.18)$$

3.cü ve 4.cü tip için verilen formüller fark alıcı ve hillber dönüstürücüleri tasarımında için LS hata metodu baz alınarak kullanılır.

Sonuç olarak :

Frekans örnekleme metoduyla FIR filtre tasarımında faz 0 ile Nyquist frekası (w=π , normalize olarak f= .5 arasında değiştirilir. Tecrübeler filtre uzunluğu arttığında filtre cevabının arzulanan ideal cevaba daha çok yaklaştığını göstermiştir. Filtre uzunluğunun tek veya çift olması dışında sıfır frekansında örnek alma veya almama ile geçiş ve durdurma bandları arasında orta noktada bir geçiş bölgesi tanımlamak mümkün olacaktır. Bu şekilde osilasyon ve aşım azaltılabilir.



Şekil 5.4 N=20 FIR filtre.Frekans Örnekleme Metoduyla tasarım , Frekans cevabı.

5.2.2 LS Hata Frekans Domeni Tasarımı

Genel olarak pek çok filtre istenen ve istenmeyen işaretlerin ayrıştırılmasında kullanılır. Sıklıkla işaretin tanımı frekans domeninde frekans bileşenleri veya frekans bandına işaretin enerjisiyle verilir. Bu yüzden filtre özellikleri frekans domeninde verilir. İşaretin enerjisi işaretin karesiyle bagıntılıdır. Karesel hata yakınsama kriteri tasarıma uygun düşmektedir. Bu bölümde temel olarak iki metod incelenecektir.

1. Hatanın karesi toplamı örneklenen frekanslarda ölçülen sonlu sayıda bir kümedir.
2. Hatanın karesinin sonlu yada sonsuz sayıda frekanslardaki integralidir.

5.2.2.1 Ayrık Frekans Örnekleri

Frekans örnekleme metodu gerçekte bir yakınsama olmamakla birlikte bir interpolasyondur. Frekans cevabı tümüyle örneklenen noktalardan geçer ama bu noktalar arasında cevabın ne şekilde olacagını kontrol eden faktörleri içermez. Bu yüzden istenmeyen sonuçlar alınabilir. Şimdi bu nokatalar arsındaki cevabı nasıl kontrol edeileceğini irdelermek için alınan örneklerin filtrenin uzunluğundan daha büyük olduğunu varsayalım. Bu bilinmeyenlerden daha fazla eşitliğe yolaçar. Bu kullanılarak bir yakınsama elde edilebilir. FIR filtrenin frekans cevabı ,

$$H(w) = \sum_{n=0}^{N-1} h(n)\, e^{-jwn} \quad \text{olarak verilmişti.} \qquad (5.19)$$

Tasarım problemi arzu edilen ve gerçekleşen frekans cevabının farkının karesi olarak L adet örnekteki hatanın toplamının ölçülmesiyle mümkün olmaktadır. Hata fonksiyonu şu şekilde tanımlanmaktadır.

$$E = \sum_{k=0}^{L-1} | H(w_k) - H(w_k) |^2 . \qquad (5.20)$$

$H_d(w_k)$ istenen cevabın L adet örneğidir.

Bu problemi formüle etmek frekans örneklemesi eşit aralıkla yapılıyorsa oldukça kolaydır.

$$w_k = 2\pi k / L , \qquad (5.21)$$

Aynı zamanda problem Lineer Faz filtrelerle sınırlanmıştır. Kompleks değerli H(w) yerine Reel değerli A(w) ya yakınsama yapılmaktadır. Bu koşullarla ,

$$E = \sum_{n=-(L-1)/2}^{(L-1)/2} | h_n - h_{dn}|^2 \text{ olur. Daha basit bir notasyonla ,} \qquad (5.22.a)$$

$$E = \sum_{k=0}^{L-1} | A_k - A_{dk} |^2 \text{ olarak yazılabilir.} \qquad (5.22.b)$$

Fourier dönüşümü LS tasarımına ilişkin güçlü özelliklere sahiptir. Parserval teoremi DFT nun ortogonallik özelliğni kullanır. Bu teoreme göre frekans domeninde verilen hata aynı zamanda zaman domeninde hesaplanabilir. L çift ise,

$$E = \sum_{n=-(L-1)/2}^{(L-1)/2} | h_n - h_{dn}|^2 \qquad (5.23.a)$$

h_d L uzunluğunda FIR filtre.
A_{dk} L tane frekans cevabı örneğidir.

Filtrenin frekans cevabını frekans örnekleme metodunda kullanılan formüllerle L uzunluğunda IDFT kullanarak bulabiliriz. Bu şekilde N uzunluğundaki filtrenin impuls cevabını L uzunluğunda frekans cevabı örneğiyle hesaplayabiliriz. Filtre uzunluğu N olduğunda Hata fonksiyonu iki toplama ayrılarak yazılabilir :

$$E = \sum_{n=-M}^{M} | h_n - h_{dn} |^2 + \sum_{n=M+1}^{(L-1)/2} 2 | h_{dn} | \qquad (5.23.b)$$

Bu eşitlik E hatasının nasıl minimize edilebileceğni gösterir. N tane h_n N tane h_{dn} değerine eşit olarak seçilir. Diğer bir deyişle h_n i h_{dn} i keserek elde ederiz.

Sonuç olarak çift uzunlukta bir filtre tasarımı için önce L (çift) uzunluğunda filtre frekans örnekleme metoduyla elde edilir. Simetrik olarak kesilerek (çift) N uzunluğa indirilir. Tek uzunlukta bir filtre tasarımı için önce L (tek) uzunluğunda filtre frekans örnekleme metoduyla elde edilir. Simetrik olarak kesilerek (tek) N uzunluğa indirilir. Elde edilen N uzunluğundaki FIR filtre LS hata yaklaşıklığıyla elde edilir.

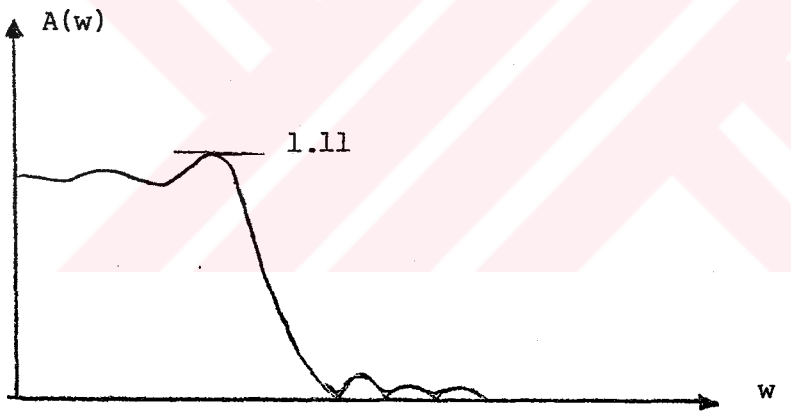
5.2.2.2 İntegral Karesel Hata Yaklaşım Kriteri

Hata arzu edilen genlik ile elde edilen genliğin farkının bir peryot boyunca $-\pi < w < \pi$ arasında integralidir.

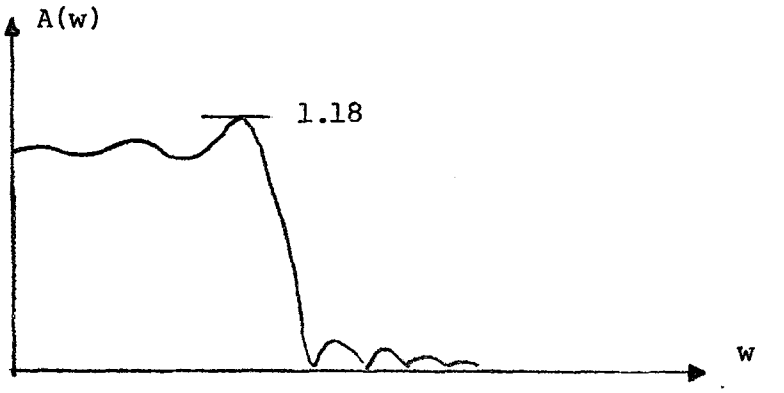
$$E = 1/2\pi \int_{-\pi}^{\pi} | A(w) - A(w) |^2 \, dw \qquad (5.24)$$

5.3 Gibss Olayı

Alçakgeçiren filtrenin genlik frekans cevabı geçirme bandının ucunda osilasyona sahiptir. Bu davranışa Gibss Olayı denir. Süreksizliği olan bir fonksiyonun fourier yaklaşığı alınırken süreksizlik bölgesi civarında aşım meydana geldiği görülür. Fourier serisi terimleri artırıldığında karesel hata azalır ve terim sayısı sonsuza götürüldüğünde hata sıfıra gider. Bununla beraber aşımın maksimum değeri ve maksimum hata sıfır olmaz yaklaşık %11 kadar aşım olur. Bu durun LS hata tasarım metodunda elde edilir. Frekans örnekleme metodunda süreksizlik civarında %18 kadardır. [14]



Şek 5.5 LS hata filtre tasarımında Gibbs olayı.



Şek 5.6 Frekans örnekleme metodundaki aşıma örnek.

5.4 Türev alıcının LS hata yakınsaması tasarımı

İdeal bir fark alıcı lineer faz FIR filtre ile elde edilebilir. Bir fark alıcının frekans cevabı

$$H(w) = jw \text{ veya } A(w) = w \text{ dır.}$$

A(w) tek bir fonksiyondur ve 90 derece faz ötelemesi vardır. 3.cü ve 4.cü tip filtreler bu amaçla kullanılabilir. Yüksek frekans girişimini engellemek için alçak geçiren 3.cü tip filtre eğer daha geniş bir band kullanılıyorsa 4.cü tip filtre kullanılır.

Fark alıcısı 3.tip filtre tasarımı için N tek olmak üzere $-\pi < w < \pi$ aralığında periyodu 2π olan $A_d(w)$tanımlarsak bu durumda filtre katsayıları aşağıdaki formülden hesaplanabilir :

$$h(n) = -1/2\pi \int_{-\pi}^{\pi} w \sin(wn)\, dw \qquad (5.25)$$

$$h(n) = \begin{cases} \cos(\pi n)/n & n \neq 0 \\ 0 & n = 0 \end{cases}$$

Geniş band fark alıcı tasarımı için 4.cü tip FIR filtre N çift olmak üzere $-2\pi < w < 2\pi$ aralığında $A_d(w)$ tanımlarsak , 4π periyotlu , ideal yanıt

$$A_d(w) = \begin{cases} -2\pi - w, & -2\pi < w < -\pi \\ w, & -\pi < w < \pi \\ 2\pi - w, & \pi < w < 2\pi \end{cases}$$

Filtre katsayıları

$$h(n) = -1/4\pi \int_{-2\pi}^{2\pi} A(w) \sin(wn/2)\, dw \qquad (5.26.a)$$

$$h(n) = -4 \sin(\pi(2n+1)/2) / \pi(4n^2 + 4n + 1)$$

bulunur. (5.26.b)

5.5 Geçiş bölgeli LS hata yaklaşımı

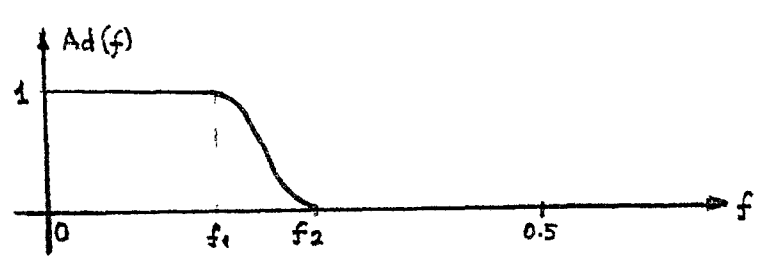
FIR filtre tasarım problemi geçiş ve durdurma bandları arasında bir geçiş bölgesi sağlanmasıyla oldukça esnek bir yapıya kavuşur. Geçiş bölgesi tanımıyla durdurma ve geçirme bandlarını tek bir frekans ayırmaz ve Gibss olayı ortadan kaldırılabilir. İstenen cevaba yakınsama geçiş bölgesi sayesinde daha kolay olur.

Geçiş bölgesi şöyle tanımlansın :

$0 < f < f_1$ geçirme bandı ,

$f_1 < f < f_2$ geçiş bölgesi ,

$f_2 < f < .5$ durdurma bandı. Bu yapıya ilişkin şekil aşağıda görülmektedir , $H_t(f)$ geçiş fonksiyonudur.



Şek 5.7 Geçiş bölgeli yapının ideal cevabı

Geçiş bandını durdurma bandına bağlıyan fonksiyon birinci dereceden basit bir çizgi şeklindeyse impuls cevabı şöyle yazılabilir :

$$h(n) = \begin{cases} \dfrac{\sin(\pi(f2-f1)(n-M))\sin(\pi(f2+f1)(n-M))}{\pi(f2-f1)(n-M)\ \pi(n-M)} & 0 \le n \le N-1 \\ 0, & \text{diğerleri} \end{cases}$$

(5.27)

Bu formül $A_d(w)$ ideal frekans cevabının ters fourier dönüşümü alınarak elde edilebilir. Geçiş bölgesindeki düz çizginin fonksiyonu ideal yüksekliği $1/(f2-f1)$ ve genişliği $f2-f1$ olan ve band uç noktası $(f2+f1)/2$ de olan dar bir dikdörtgensel fonksiyon olarak tanımlanabilir.

Eğer geçiş bölgesi 2. dereceden bir parabolse bu durumda $h(n)$ şöyle hesaplanabilir :

$$h(n) = \begin{cases} \dfrac{\sin(\pi(f2-f1)(n-M)/2)^2}{(\pi(f2-f1)(n-M)/2)^2} \dfrac{\sin(\pi(f2+f1)(n-M))}{\pi(n-M)} & 0 \le n \le N-1 \\ 0, & \text{diğerleri} \end{cases}$$

(5.28)

Bunu genelleştirirsek p. ci dereceden çizgi için $h(n)$ şu formülle hesaplanabilir :

$$h(n) = \begin{cases} \dfrac{\sin(\pi(f2-f1)(n-M)/p)^p}{(\pi(f2-f1)(n-M)/p)^p} \dfrac{\sin(\pi(f2+f1)(n-M))}{\pi(n-M)} & 0 \le n \le N-1 \\ 0, & \text{diğerleri} \end{cases}$$

(5.29)

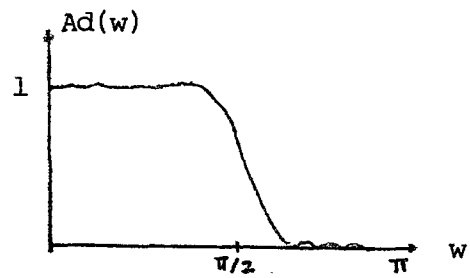
Alternatif geçiş bölgesi fonksiyonu olarak yükselen cosinüs fonksiyonu kullanılabilir.

$$A_d(w) = \begin{cases} 1 & 0 < f < f1 \\ 1 + \cos(\pi(f-f1) / (f2-f1) & f1 < f < f2 \\ 0 & f2 < f < 0.5 \end{cases} \qquad (5.30)$$

Bu tanımama kullanılarak ,

$$h(n) = \begin{cases} \dfrac{\cos(\pi(f2-f1)(n-M))}{1-4(f2-f1)^2(n-M)^2} \quad \dfrac{\sin(\pi(f2+f1)(n-M))}{\pi(n-M)}, & 0<n<N \\ 0 & \text{diğerleri} \end{cases} \qquad (5.31)$$

Bu fonksiyon cizgi fonksiyonuyla birleştirilerek daha yumuşak geçişler sağlanabilir. h(n) hesaplama hızı artan n ile azalır ama ters fourier dönüşümünde hata azalır. $A_d(w)$ nın Q kez türevi alınırsa sonlu h(n) 1/n in (Q+1).ci kuvvet çarpanıyla küçülecektir. Geçiş fonksiyonun seçimi geçiş bölgesinin büyüklüğüne , band genişliğine ve filtre uzunluğu N e bağlıdır. Geçiş bölgesi kavramı sadece alçakgeçiren filtrelerde değil yüksekgeçiren , hillbert ve türev alıcılardada kullanılır. [12]



Şekil 5.8 Geçiş bölgeli LS hata tasarımlı filtre

5.6 FIR filtre tasarımında Geçiş bölgeleri , Ağırlık Fonksiyonları ve Pencereleme

Hatayı minimize etmek için dört yaklaşım göz önüne alınabilir :

1. yaklaşım : Arzulana frekans cevabındaki süreksizlik bölgesini geçiş bölgesiyle yok ederek Gibss olayını ortadan kaldırmaktır. Bu şekilde dgeçiş ve durdurma bölgeleri birbirine bağlanır.

2. yaklaşım : Hata kriterini değiştirerek aşımı azaltmak veya ortadan kaldırmaktır. Bu bir frekans bölgesini optimizasyonda ortadan kaldırmakla sağlanır. Bu eşit aralıklı olamayan örneklemede ayrık LS hata kriteriyle sağlanır.

3. yaklaşım : Ağırlık fonksiyonları kullanarak ölçülen hatanın değiştirilmesidir. Hatanın fazla olduğu yerlerde ağırlığı fazla , düşük olan yerlerde ,örnegin geçiş bölgesinde küçük alarak sağlanır.

4. yaklaşım : LS hata tasarımının sonuçlarını kullanarak dirket olarak h_n katsayılarının aşımı azaltacak şekilde düzeltilmesidir. Sonuç artık optimal değildir. Aşım somlu L uzunluğundaki sekansın kesilmesi sonucu ortaya çıktığı için , zaman domeni pencereleri ile daha iyi kesme sonuçları elde edilebilir.

5.6.1 Arzu edilen Frekans cevabında düzeltmeler

En basit dizeltem genlik fonksiyonunun geçiş bandında 1 durdurma bandında 0 ve geçiş bölgesinde 1.ci dereden bir doğru olması durumudur. Genlik cevabı şöyle tanımlanabilir.

$$A_d(w) = \begin{cases} 1 & 0 < w < w1 \\ (w2-w)/(w2-w1) & w1 < w < w2 \\ 0 & w2 < w < \Pi \end{cases} \qquad (5.32)$$

5.6.2 Geçiş bölgesinin kullanımı

Geçiş bölgesi fonksiyonun geçiş bölgesinde ideal sürekli frekans cevabına yakınsamayı sağladığı önceki bölimde incelenmişti.Geçiş bölgesinin frekans bandı hata fonksiyonundan kaldırılır. Bu bölgeye dikkate alınmayan band denir. Hata fonksiyonu bu durumda :

$$E = \int_0^{v1} | A(w) - A_d(w) |^2 \, dw + \int_{v2}^{\Pi} |A(w)-A_d(w)|^2 \, dw \qquad (5.33)$$

olarak yazılabilir. w1 ve w2 arasındaki bölge kaldırılmıştır.

5.6.3 Ağırlıklı Karesel Hata Kriteri kullanılması

Hata tanımını esnekleştirmek için pozitif bir W(w) fonksiyonu kullanılırsa ,

$$E = 1/\Pi \int_0^{\Pi} W(w) \, |A(w) - A_d(w)|^2 \, dw \quad \text{şekline girer.} \qquad (5.34)$$

5.6.4 FIR filtre Tasarımında Pencereleme Fonksiyonlarının Kullanımı

Sonsuz terimin zaman domeninde L adet terim olarak kesilmesi frekans domeninde süreksizlik bülgesinde Gibss olayına yol açmaktaydı. Bunu karesel bir fonksiyonu ele alarak incelersek $-M < n < M$ olmak üzere 1 ve diğer bölgelerde sıfır olan ve sıfır noktasına göre simetrik olan fonksiyon ve $h(n)$ kesilen sonlu sonuç olmak üzere ,

$$h(n) = h_d(n)\, r(n) ,$$

ve

$$r(n) = \begin{cases} 1 , & |n| \le M \\ 0 , & |n| > M \end{cases} \qquad (5.35)$$

İki fonksiyonun zaman domneninde çarpımı iki işaretin Fourier dönüşüğünün konvolüsyonudur. Dikdörtgensel bir işaretin fourier dönüşümü alınırsa

$$R(w) = FT(r(n)) = \sum_{n=-\infty}^{\infty} r(n)\, e^{-jwn} \qquad N \text{ terim için , (5.36.a)}$$

$$= \sum_{n=-M}^{M} r(n)\, e^{-jwn} = \sin(wN/2)/\sin(w/2))$$

(5.36.b)

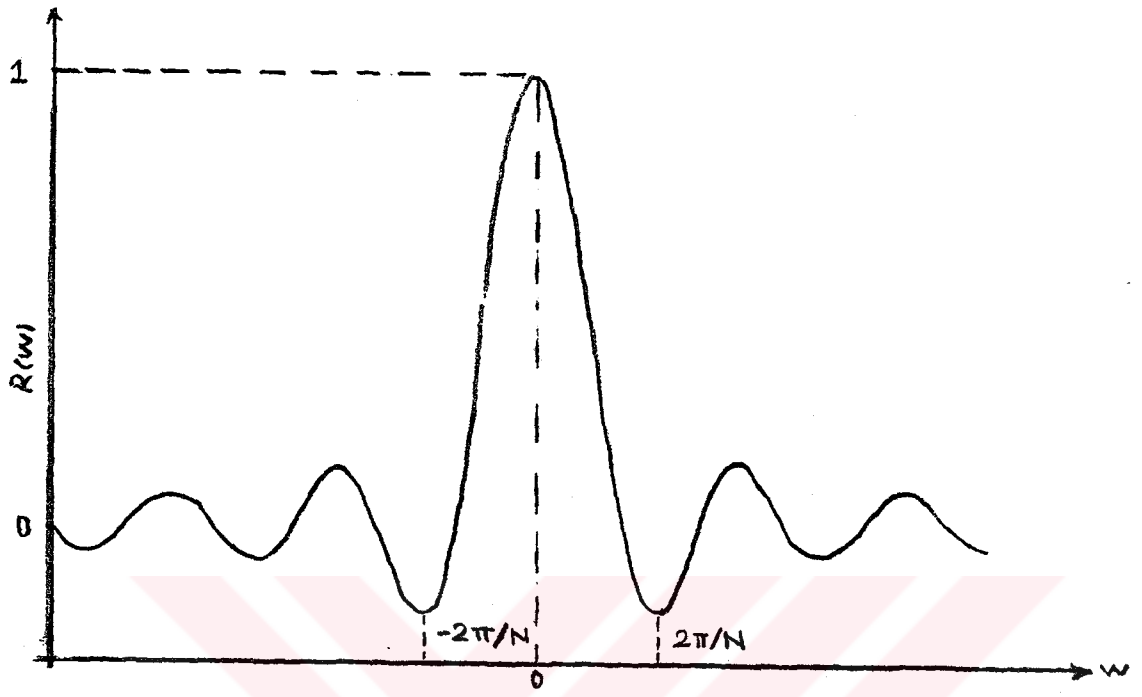
elde edilir.

Zaman domenindeki kesme frekans domeninde ,

$H(w) = H_d(w) * R(w)$ olarak gösterilir.

Sonlu uzunluktaki filtrenin frekans cevabı dikdörtgen fonksiyonuyla konvolüsyonu ideal frekans cevabına vermektedir. Uzunluğu N olan Dikdörgen fonksiyonunun

frekans cevabı şekil 5.9 da görülmektedir.



Şekil 5.9 Dikdörtgen fonksiyonun Fourier dönüşümü

İdeal frekans cevabıyla dikdörtgen fonksiyonunun konvolüsyonu Gibss olayını vermektedir. Şekilden görüleceği üzere N artırıldıkça osilasyonun ana parçası dahada daralmaktadır ama yüksekliği hep aynı kalmaktadır. Osilasyonlu kenar lopları Gibss olayına yol açmaktadır. İdeal R(w) da aşım ve yan loplar yoktur ve yumuşak bir geçiş vardır. İdeal R(w) nın genişliği sıfırdır. Buda DİRAC delta fonksiyonuna karşı düşmektedir. Ancak bu durum N sonsuza götürülürse elde edilir. Bu durumda sonlu uzunlukta filtre yapmak imkansızdır ama ideal R(w) ya yakınsayan literatürde altı tane pencere fonksiyonu tanımlanmıştır. Bunlar ,

1. Bartlett üçgensel penceresi :

$$W(n) = \begin{cases} 2(n+1) / (N+1) & n = 0,1 \ldots (N-1)/2 \\ 2 - 2(n+1) / (N+1) & n = (N-1)/2, \ldots N-1 \\ 0 & \text{diğerleri} \end{cases} \qquad (5.37)$$

2,3,4,5 Genelleştirilmiş kosinüs pencereleri

$$W(n) = \begin{cases} a - b\cos(2\pi(n+1)/(N+1)) + c\cos(4\pi(n+1)/(N+1)) & n = 0,1,2,\ldots,N-1 \\ 0 & \text{diğerleri} \end{cases} \qquad (5.38)$$

Pencere Tipi	a	b	c
Dikdörtgen	1	0	0
Hanning	.5	.5	0
Hamming	.54	.46	0
Blackman	.42	.5	.08

6. Kaiser penceresi β parametreli

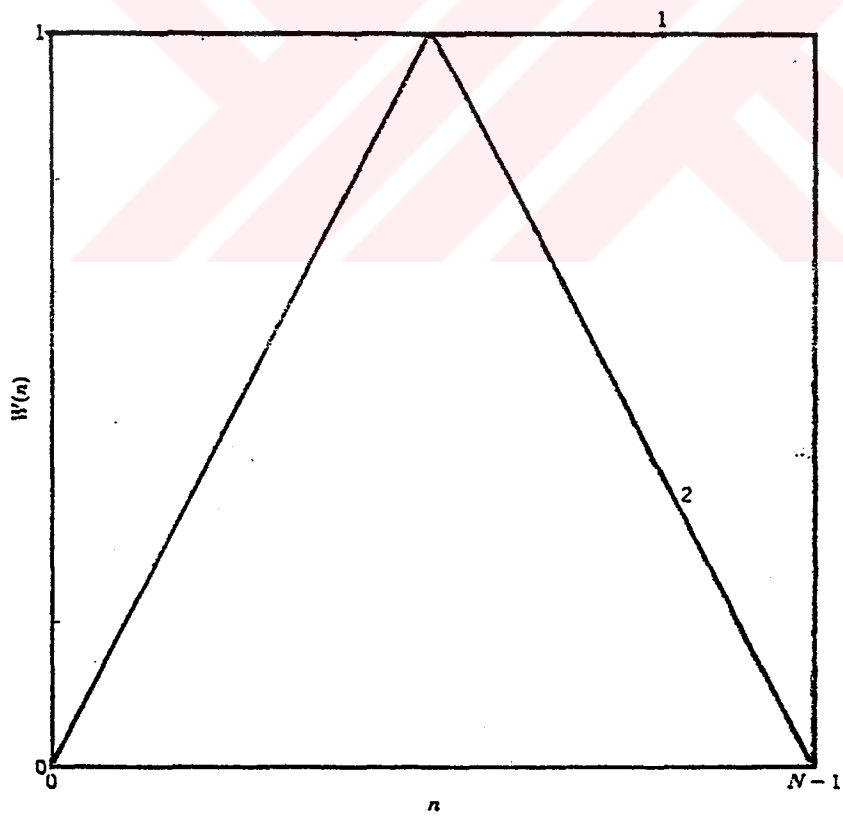
$$W(n) = \begin{cases} \dfrac{I_o\left(\beta \sqrt{1-\left((2n+1)/(N+1)\right)^2}\right)}{I_o(\beta)} & n=0,1..N-1 \\ 0 & \text{diğerleri} \end{cases} \qquad (5.39)$$

En basit pencere fonksiyonu dikdörtgen penceredir ve Gibss olayına yol açar.

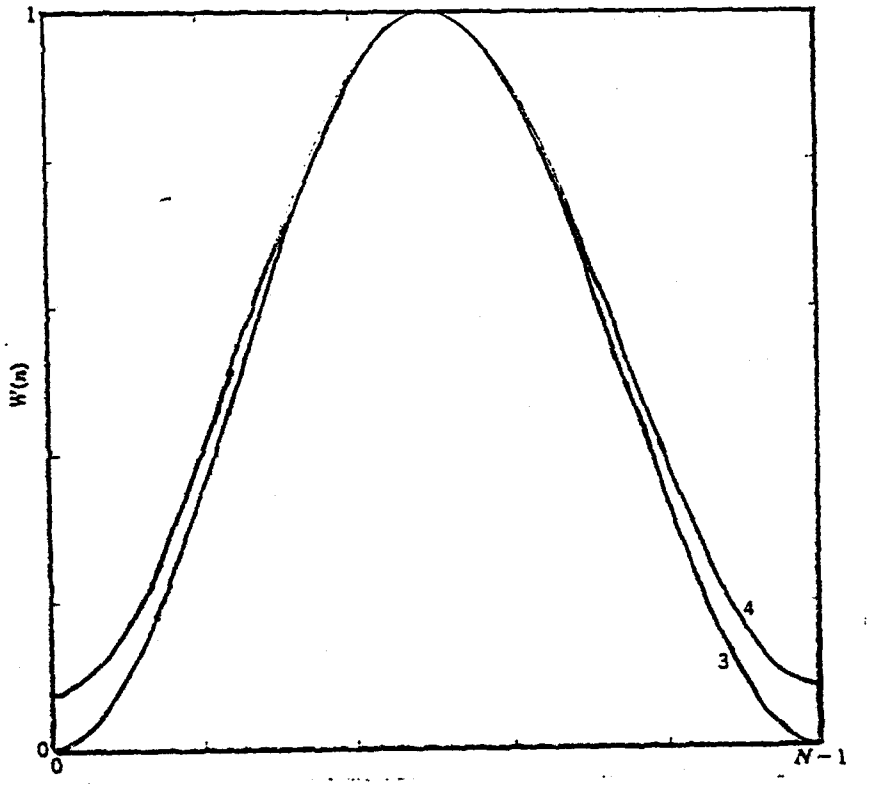
Bartlett penceresi aşımı azaltır ancak geçiş bölgesini büyütür.

Hanning , Hamming ve Blackman komplile kosinüs fonksiyonlarıdır ve ideal cevaba daha yumuşak bir geçiş sağlar.

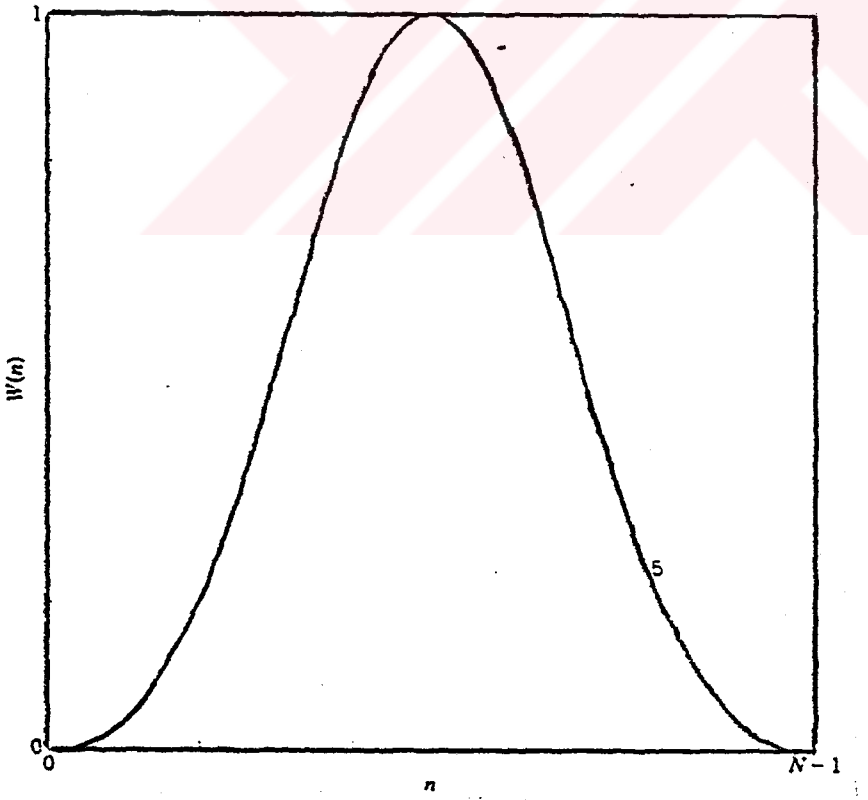
Kaiser penceresi ve kullanılan β parametresi aşımın azaltılması ve geçiş bölgesi genişliğinin ayarlanmasına olanak verir. Geçiş ve durdurma bandlarındaki aşım filtre uzunluğu N e ve β parametesine çevrilir. Kaiser penceresi bessel fonksiyonunun çözülmesini gerektirir. Verilen en iyi pencere fonksiyonudur. [15]



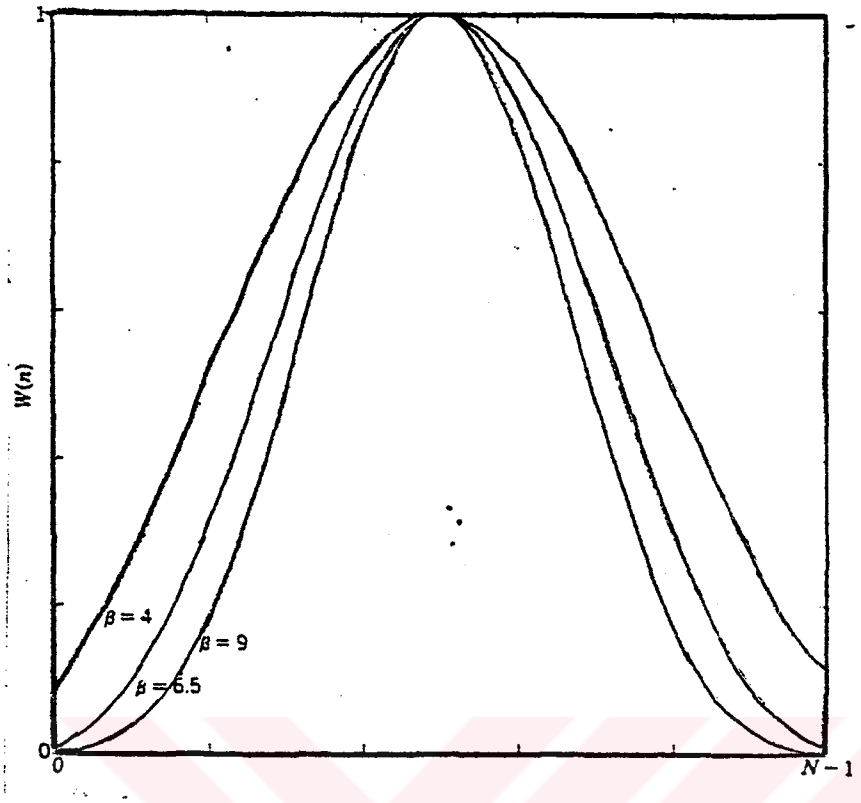
Şekil 5.10 1-Dikdörtgen pencere
2-Bartlett penceresi



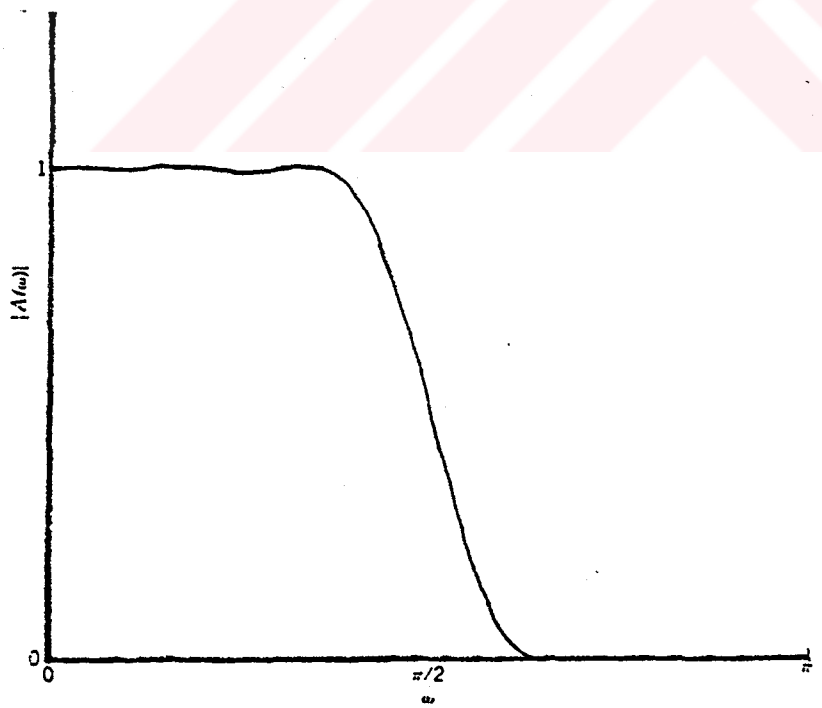
Şekil 5.11 3-Hanning penceresi
4-Hamming penceresi



Şekil 5.12 5-Blackman penceresi



Şekil 5.13 Kaiser penceresi



Şekil 5.14 N=21 FIR filtre , Kaiser penceresi ile β=9

5.6.5 Chebyshev Yakınsaması

Frekans kümesindeki maksimum hataları minimize etmeyi amaçlayarak FIR filtre katsayılarını hesaplayan yakınsama metodudur. Bundan sonraki bölümde incelenecek olan Parks-McCellan algoritmasıda bu yakınsamayı kullanmaktadır. Dördüncü bölümde bahsedilen 4 tip filtre için kullanılan yakınsama fonksiyonları şunlardır :

Genel fonsiyon ,

$$H(f) = (f)^{m} A(f) e^{-j((N-1)/2)2\pi f} \qquad (5.40)$$

ve m = 0 , 1 olmak üzere ,

1. Tek Simetri + Tek Uzunluk için,

$$h(n) = h(N-n-1) \quad , \quad (m=0) \qquad (5.41)$$

$$A(f) = \sum_{k=0}^{(N-1)/2} a_k \cos(2\pi f k) \qquad (5.42)$$

$$a_0 = h(N-1)/2 \qquad (5.43.a)$$

$$a_k = 2h(-k + (N-1)/2) \qquad (5.43.b)$$

$$k = 1,2,\ldots,N/2-1 \qquad (5.43.c)$$

2. Tek Simetri + Çift Uzunluk için,

$$h(n) = -h(N-n-1) \quad , \quad (m=1) \qquad (5.44)$$

$$A(f) = \sin(2\pi f) \sum_{k=0}^{(N-3)/2} c_k \cos(2\pi f k) \qquad (5.45)$$

$$c_0 - 1/5\, c(2) = 2h((N-3)/2) \qquad (5.46.a)$$

$$c((N-5)/2) = 4h(1) \qquad (5.46.b)$$

$$c((N-3)/2) = 4h(0) \qquad (5.46.c)$$

$$c(k-1) - c(k+1) = 2h(-k + (N-1)/2) \qquad (5.46.d)$$

$$k = 2,\ldots\ldots, (N-5)/2$$

3. Çift Simetri + Tek Uzunluk için,

$$h(n) = h(N-n-1)\ ,\ (m=0)$$

$$A(f) = \cos(\pi f) \sum_{k=0}^{(N-3)/2} b_k \cos(2\pi f k) \qquad (5.47)$$

$$b_o + 1/2\ b(1) = 2h(N-3)/2 \qquad (5.48.a)$$

$$b((N-3)/2) = 4h(0) \qquad (5.48.b)$$

$$b(k-1) + b(k) = 4h(-k+(N-1)/2) \qquad (5.48.c)$$

$$k = 2\ ,\ldots\ldots,\ (N-3)/2$$

4. Çift Simetri + Çift Uzunluk ,

$$h(n) = -h(N-n-1)\ ,\ (m=1) \qquad (5.49)$$

$$A(f) = \sin(\pi f) \sum_{k=0}^{(N-3)/2} b_k \cos(2\pi f k) \qquad (5.50)$$

$$d_o + 1/2\ d(1) = 2h(N-3)/2 \qquad (5.51.a)$$

$$d((N-3)/2) = 4h(0) \qquad (5.51.b)$$

$$d(k-1) + d(k) = 4h(-k + (N-1)/2) \qquad (5.51.c)$$

$$k = 2\ ,\ldots\ldots,\ (N-3)/2$$

Chebyshev yakınsaması için şu şartlar sağlanmalıdır ,

1. Lineer faz
2. Durdurma ve gecirme bandları arasında Δf gibi bir geçiş bölgesi olmalı
3. Birim degerinde salınım $-/+\delta_1$ kadar , geçirme bandında

4. Sıfır degerinde salınım $-\delta_z/+\delta_z$ kadar salınım , durdurma bandında

Teoreme göre ,

1.F [0,.5] arasında tanımlı,
2.D(f) arzu edilen cevap F üzerinde sürekli ve tanımlı ,
3.Pozitif ağırlık fonksiyonu W(f) F Uzrerinde sürekli ve tanımlı olsun ,

$$A(f) = Q(f) \sum_{k=0}^{r-1} c_k \cos (2\pi f k) \qquad (5.52)$$

burada amaç c_k yı minimize etmektir.

$$| E(f) | = \max_{f \in F} W (f) | D(f)-A(f) | \qquad (5.53)$$

A(f) nin seçimine göre Q(f) ağırlık fonksiyonu şu şekilde seçilir :

$$Q(f) = \begin{cases} 1 , \\ \cos \pi f , \\ \sin 2\pi f , \\ \sin \pi f , \end{cases} \qquad (5.54)$$

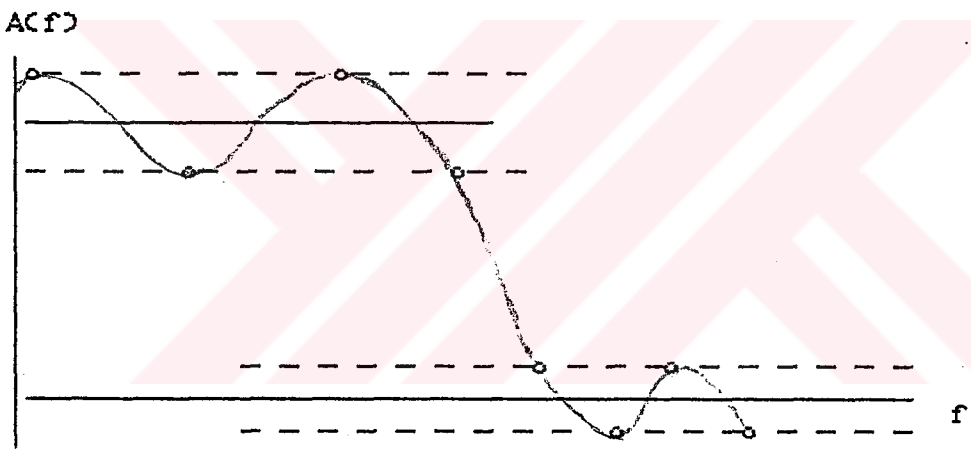
Filtre katsayıları A(f) bulunduktan sonra verilen genel fonksiyonlar kullanılarak frekans cevabı hesaplanır. Problem bir frekans setindeki maksimum deviasyonu minimize etmektir. Buda alternasyon teoremi ile doğrudan bağlantılıdır. Alternsayon teoremi 1.tip filtrelere direkt olark onun dışındakilere Q(f) ile birlikte uygulanır. Zaten 1.tip için Q(f) = 1 dir. Alternsayon teoremine göre A(f) r tane lineer kosinüs fonksiyonundan oluşmuştur ,

$$A(f) = \sum_{k=0}^{r-1} c \cos(2\pi f k) \qquad (5.55)$$

Gerek ve yeter koşulu sağlayan A(f) tektir ,en iyi Chebyshev yakınsamasını sağlayan sürekli fonksiyon F kümesinde D olmak üzere , ağırlıklı hata fonksiyonu , E(f) = W(f) [D(f) - A(f)], olmak üzere en az r+1 tane uç frekans gösterir. Bu frekanslar ,

$f1 < f2 < .... < f_r < f_{r-1}$ dir.

$$E(f_m) = - E(f_{m+1}) \quad ve \quad m = 1,2,.....,r \qquad (5.56)$$



Şekil 5.15 Uç frekanslar , N=13 için 8 adet.

5.6.6 Remez Yerdeğiştirme Algoritması

Remez yerdeğiştirme algoritması her zaman hata fonksiyonun aşağıdaki formda olabileceğini kullanarak ,

$$E(f) = D(f) - \sum_{k=0}^{r-1} c_k \cos(2\pi f k)$$ ve herhangibir verilen

r+1 frekans noktasında -/+σ değerlerini alarak ,

$$D(f_m) = \sum_{k=0}^{r-1} c_k \cos(2\pi f_m k) + (-1)^m \sigma \quad , \quad m=1,\ldots.r+1 \qquad (5.57)$$

denkleminin c_k ve genlik için tek bir çözümü vardır. σ f_m frekansındaki osilasyonun değeridir. Bu algoritma Parks-McCellan tarafından geliştirilmiştir. F frekans kümesi yaklaşık olarak filtre uzunluğunun 10 katı eleman içermektedir. Eğer F kümesi r+1 frekansdan oluşuyorsa yakınsama ilk adımda gerçekleşir. F kimesinin maksimum hatası $|\sigma|$ kadardır. F r+1 eleman içeriyorsa ,

$$\max_{f \in F} \left| D(f) - \sum_{k=0}^{r-1} c_k \cos(2\pi f k) \right| = |\sigma| \text{ dir.} \qquad (5.58)$$

Eğer F r+1 den fazla elemana sahipse problem yeni bir r+1 uç frekans bularak hatanın maksimum olduğu frekans setini oluşturmaktır. Verilen bir frekans kümesinde Remez algoritmasına göre şu işlemler yapılır :

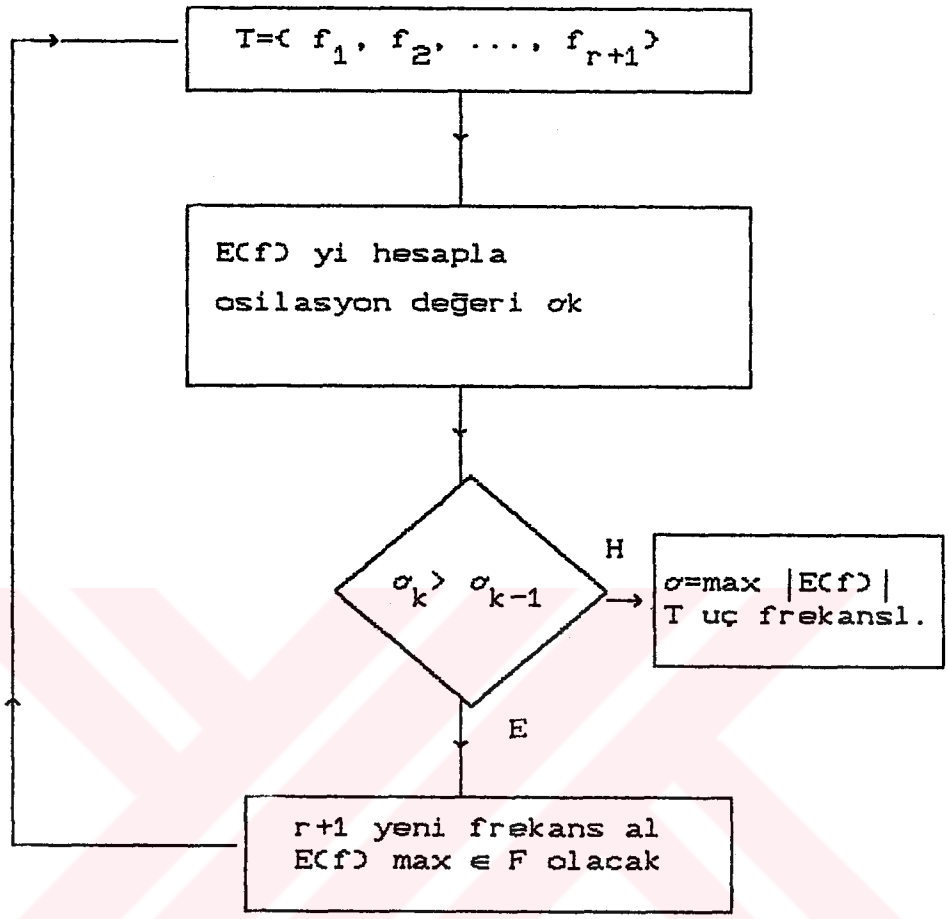
$$T = (f_1, f_2, f_3 \ldots\ldots f_{r+1})$$

1. D(f) eşitliğini çöz. Bu çözümün hatası freknas setinin k.cı iterasyonunda σ_k kadardır.

2. Frekans cevabını bulmak için tüm frekans kümesini kullanarak yer değiştir.

3. Tüm frekans kümesini araştır ,bulunan hatanın maksimum değeri 1.ci adımda bulunan $\sigma_k$dan büyükmü.

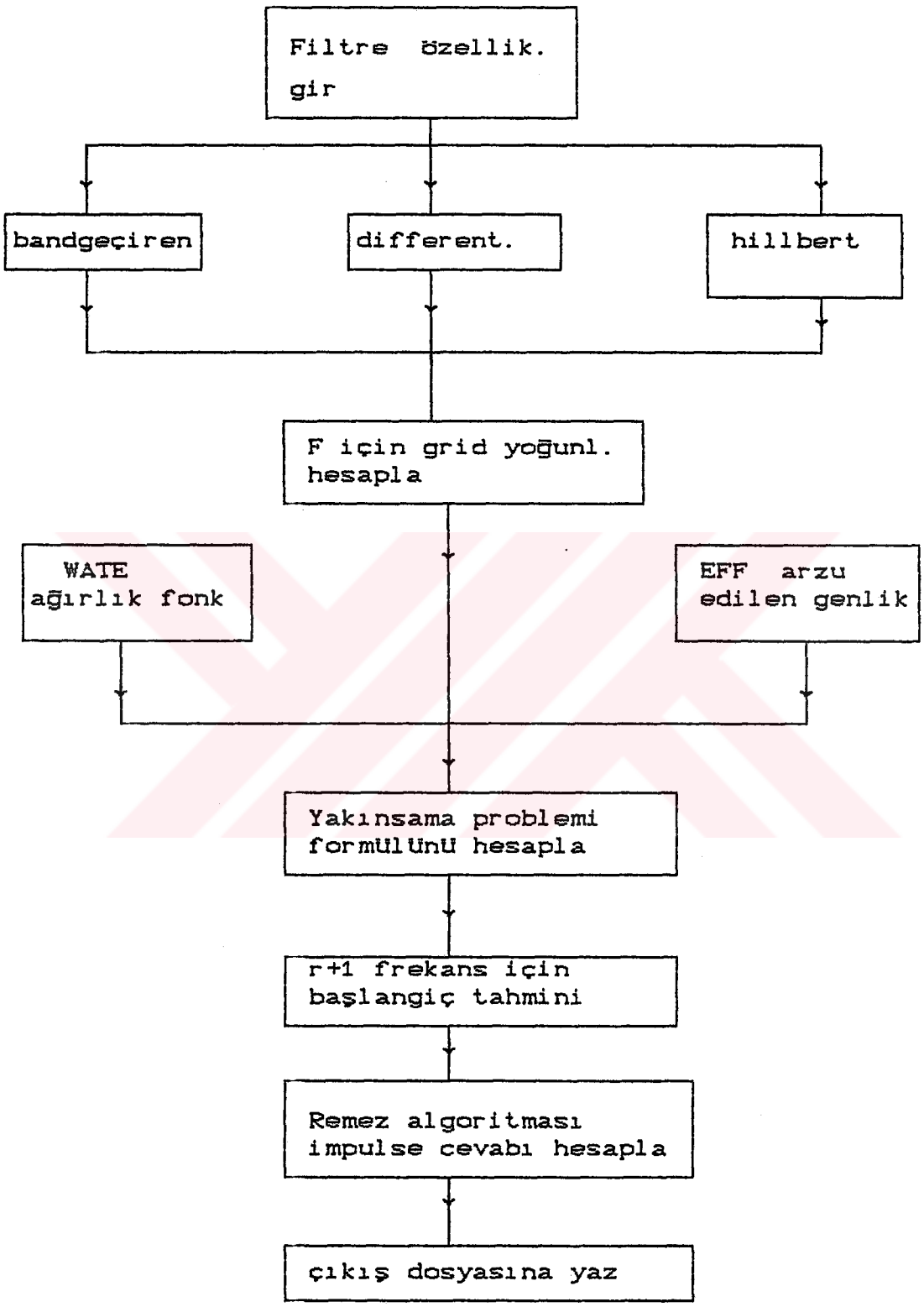
4. 3.cü adımda bulunan hatanın maksimum değeri $\sigma_k$ya eşitse dur , değilse r+1 frekans alarak hatanın maksimum olabileceği yeni bir frekans kümesi alarak 1.ci adıma geri dön. [12]



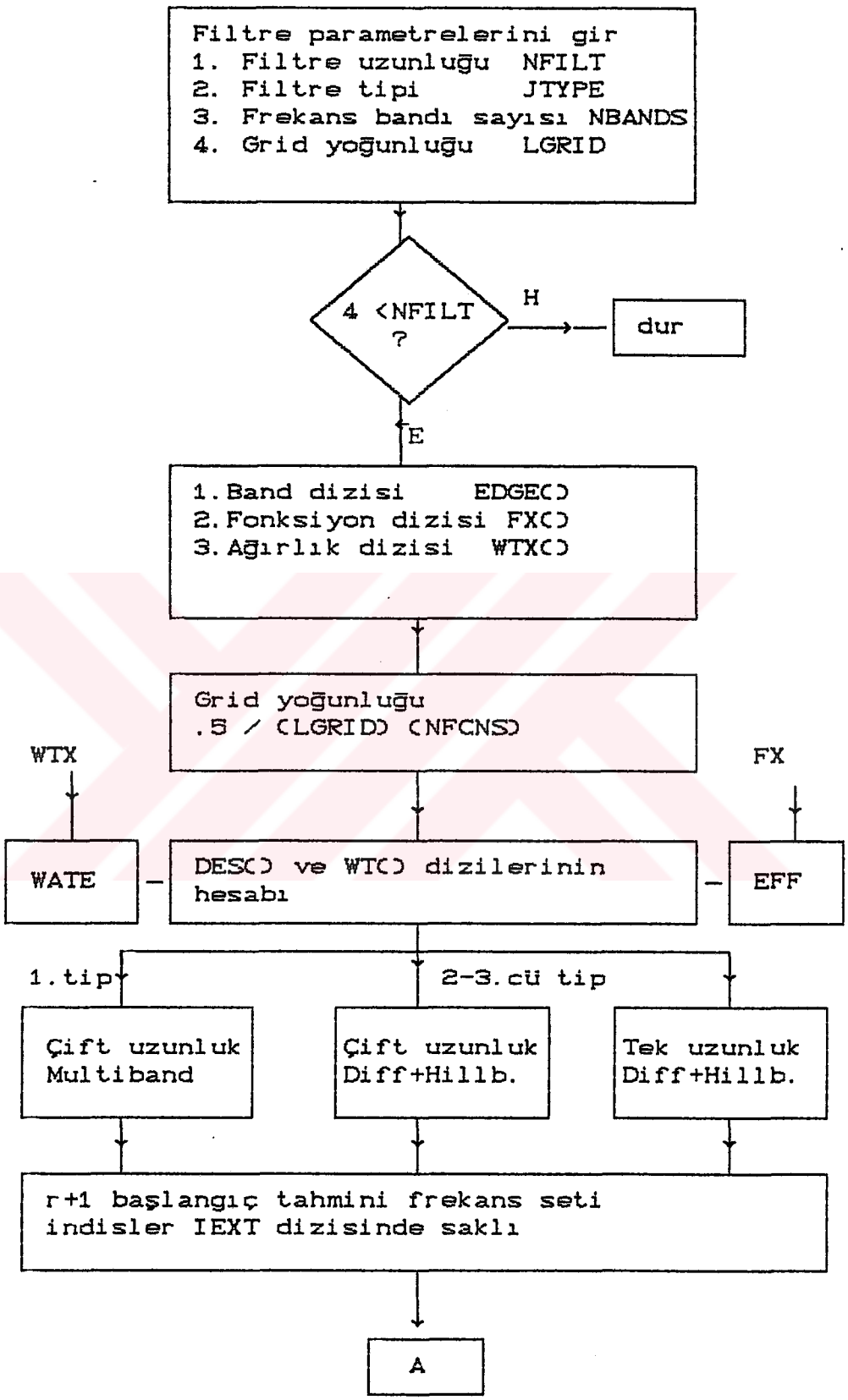
Şekil 5.16 Remez algoritmasının akış diyagramı.

Parks-McCellan algoitması Remez değiştirme metodunu kullanır. Diğer algoritmalardan üstünlüğü Remez metoduyla değişik sınıflarda FIR filtrelerin yapılabilmesidir. (Bandgeçiren , çoklu band , hillbert dönüşüm filtreleri, fark alıcı filtreler).

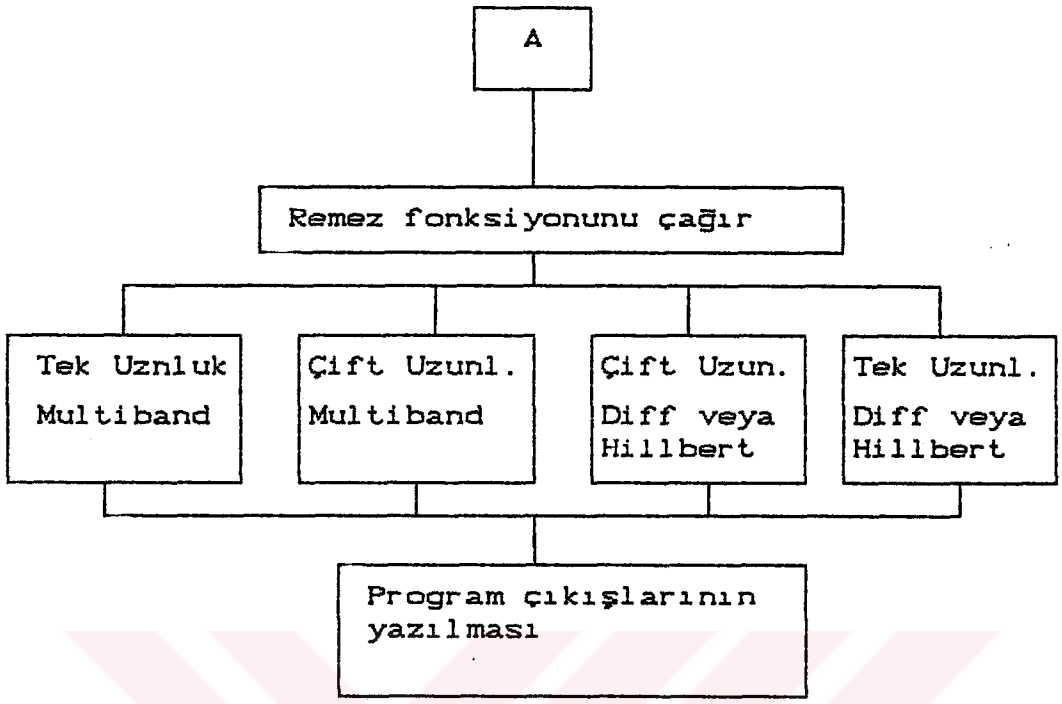
Program giriş , yakınsama , Remez çözümü , filtre impuls cevabının hesabı ve bunların çıkışından oluşmaktadır. Programın ayrıntılı akış diyagramları şekil 5.17 - 5.21 de verilmiştir. [15]



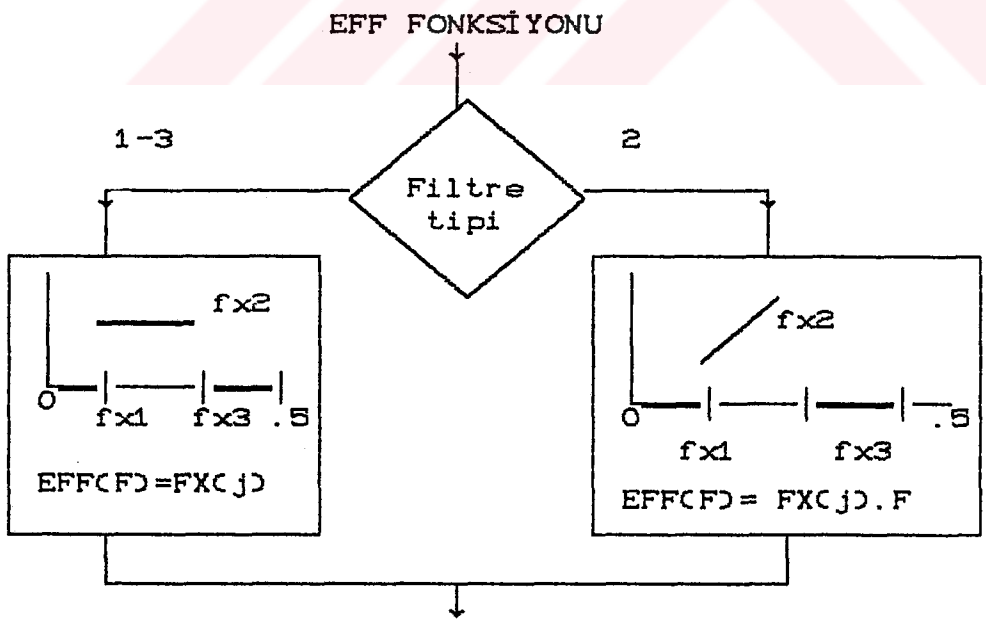
Şek 5.17 Parks-McCellan Algoritması bütün blok şeması.



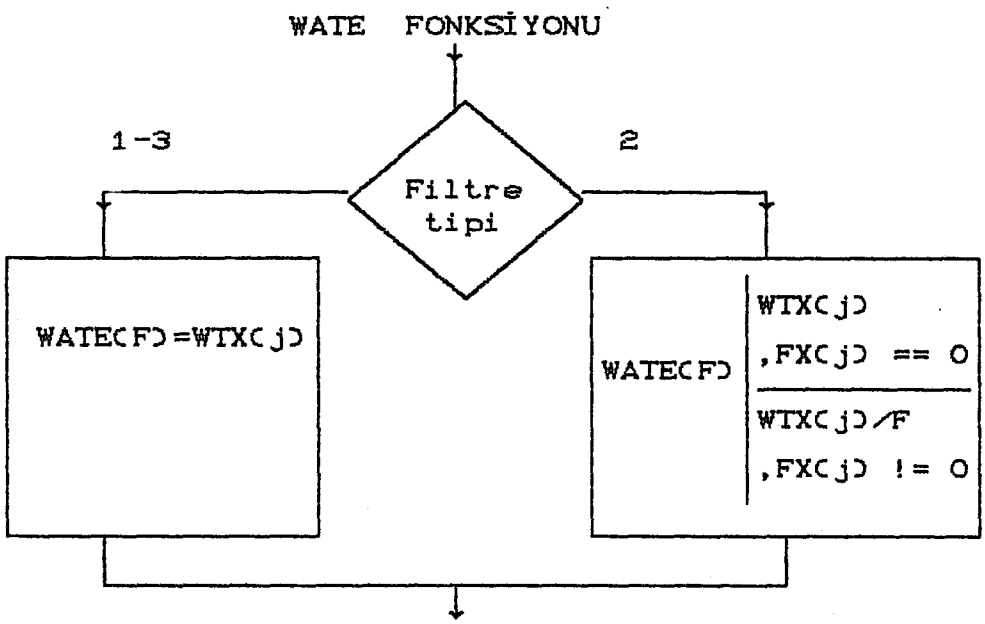
Şekil 5.18 Detaylı akış diyagramı. A şekil 5.19 devamı



5.19 Detaylı akış diyagramının devamı.



5.20 EFF fonksiyonu akiş diyagramı



5.21 WATE fonksiyonu akiş diyagramı

SONUÇLAR VE ÖNERİLER

Dijital işaret işlemede FFT en etkin algoritmalardan biri dir. Örnek alma sayısının (N) , çift olması durumunda giriş yada çikiş sekansları sürekli ikiye bölünerek hesaplama yapılır. Bu durumdan yapılan işlem sayısı N $\log_2 N$ ile orantılıdır. Genel halinde ise giriş işareti örnekleme sayısı çarpanlarına ayrılarak DFT hesabı için ayriştırma yapılır. Giriş işaretleri guruplanarak DFT ve çikiş kombinezonlarını kullanarak ters DFT hesaplanabilir. hesaplamak mümkündür. Bu algoritmalar dijital filtre realizasyonunda kullanılır. FFT klasik fourier analizine göre oldukça hızlıdır. Bu yüzden gerçek zaman uygulamalarına da uygundur.

Dijital filtrelerin temelde üç ana avantajı vardr :

1) Esneklik :

Dijital filtreler parametrelerin yeniden okunmasyla kolayca değişitirilebilir. Bu yüzden zamanla değişen filtreler kolayca gereklenebilir. Çok daha önemlisi tek bir filtre yapısı zaman paylaşımlı olarak çoklu filtrelemede kullanlabilir.

2) Güvenilirlik :

Dijital elemanlarn kullanımı , eleman toleransı , fiziksel büyüklüğü ve düşük frekanslarda çalişma özelliği gibi pek çok problemi ortadan kaldırır.

3) Modülerlik :

Dijital filtreler çok yüksek yapıda modüler olarak oluturulabilir. LSI (Large Scale Integration) için uygundur. Örneğin iki tane ikinci dereceden filtre programlanabılır karakterıstıklerıyle birlikte tek bir LSI yonga içine konabilir. [2]

FIR filtre algoritmaları arasında Parks-McCellan yötemi oldukça önemli bir yer tutmaktadır. Bu algoritma ile genel amaçlı tüm FIR filtreler , fark alıcılar ve hillbert dönüştürücüleri tasarlanabilmektedir. En iyi pencere fonksiyonu ise Kaiser tarafından verilen bir bessel denklemidir. Yakınsama kriterleri arasında da Chebyshev ve Remez değiştirme metodları etkin olarak kullanılmaktadır.

KAYNAKLAR

[1] BURRUS, C.S., Dft and Convolution Algorithms, Prentice-Hall,1984

[2] OPHENHEIM, Alan V., Digital Signal Processing Prentice-Hall,1975

[3] JAKSON, Leland B., Digital Filters and Signal Processing, Kluwer Academic Publishers, 1986

[4] GABEL, Robert A., Signals and Linear Systems, John Wiley & Sons Inc., 1987

[5] LIE, Bedu Digital Filters and the FFT, Halsted Press, 1975

[6] GOLD, Bernard Digital Processing of Signals, McGraw-Hill, 1969

[7] BRIHAM, E., The Fast Fourier Trasnform, Prentice-Hall,1975

[8] MORRIS, L.R., Digital Signal Processing Software, DSPSW Inc, 1983

[9] RABINER, L.R., Theory and Application of Digital Signal Processing, IEE , 1972

[10] BLAHUT, R.E., Fast Algorithms For Digital Signal Processing, Addison Weely Inc, 1984

[11] JOHN LOY, Nicholas — An Engineer's guide to FIR Digital Filters , Prentice Hall , 1988

[12] PARKS, T.W. — Digital Filter Design, John Wiley & Sons Inc., 1987

[13] BOSE, N.K. — Digital Filters Theory and Applications, North Holland ,1985

[14] OPPENHEIM, Alan V. — Disrete Time Signal Processing , Prentice-Hall , 1989

[15] THOMAS, J.B — Benchmark Papers in Electrical Engineering and Computer Science, Digital Filters and the Fast Fourier Trasnform, David Hutchingon & Ross Inc., 1975

EK.A PROGRAMLAR

DFT.C

RADIX-2 DFT algoritmasıdır. Herhangibir örnek doyasını alarak DFT'nunu hesaplar. Sonuç o anda directoryde bulun mayan bir dosya ismi otomatik oalarak bulunarak bu dosyaya yazılır. Dosyaların ilk dört harfi RFFT ile başlar.

```
int  dft_win(void);
int  Calc_DFT(unsigned char  *SampFil,int  mwn);
struct  kompleks *komp_sub(struct  kompleks *a,struct  kompleks *b);
struct  kompleks *komp_add(struct  kompleks *a,struct  kompleks *b);
struct  kompleks *komp_mul(struct  kompleks *a,struct  kompleks *b);
double  komp_mag(struct  kompleks *a);
double  komp_aci(struct  kompleks *a);
int  calc_fft(struct  kompleks *ary,int  n);
int  calc_BF(struct  kompleks *ary,int  N,int  r,int  p,int  q);
struct  kompleks *WW(int  N,int  r);
int  bit_rev(int  gg,int  s);
struct  kompleks *OrnekOku(unsigned char  *Fname,int  *Elem);
int  Yaz_Sonuc(struct  kompleks *r_ary,int  length,int  mwn);
void  FreMem(struct  kompleks *ary,int  length);
```

```
#include <vcstdio.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <malloc.h>
#include <direct.h>
#include "komp.h"
#define PI      (double)3.141592654
#define TEXT    unsigned char
#define COUNT   unsigned short

double  atof();
char    *tmpnam();

typedef struct kompleks {
        double reel;
        double sanal;
} KOMP;
void    FreMem();
KOMP    *OrnekOku();
KOMP    *komp_mul(),
                *komp_sub(),
                *komp_add();
double  komp_mag(),
                komp_aci();
KOMP    *WW();
int             bit_rev();

/*************************************************/

dft_win()
{
TEXT    SFil[13];
int             mwn0,
                mwn;

                empty ( SFil , 13 );
        mwn0 =  wxopen( 9,
                                                12,
                                                20,
                                                66,
                                                "",
                                                ACTIVE+COOKED,
                                                0,
                                                0,
                                                4,
                                                178 );

        mwn = wopen   ( 8,
                                               10,
                                               19,
                                               64,
```

```
                                        "DFT HESAPLAMA"
                                        );
                wxatsay ( mwn ,  5 , 1 , "ORNEKLEME DOSYA ADI : ", 7 );
                xatget  ( 5, 23, SFil, "XXXXXXXX.XXX", NULLFUNC, "", "", vc.dflt, vc.dflt );
                switch ( readgets() ) {
                        case    RET:
                                        wxatsay ( mwn ,  6 , 1 , "DFT Hesaplaniyor Lutfen Bekleyiniz .
                                        Calc_DFT ( SFil  , mwn );
                                        break;
                        case    ESC:
                        deafult    :
                                        break;
                }
                wclose ( mwn );
                wclose ( mwn0 );

}

Calc_DFT( SampFil , mwn )
TEXT    *SampFil;
int             mwn;
{
KOMP    *r_ary;         /* it's going to be allocated in function OrnekOku  */
int             length;

        r_ary = OrnekOku( SampFil  , &length );     /* Reads the samples    */
                                                                                /* Calculates
        calc_fft ( r_ary , length );                            /* Calculates  the DFT  */
        Yaz_Sonuc( r_ary , length , mwn );              /* Writes it to a file  */
}
/*****************************************************/
KOMP *komp_sub( a , b )
KOMP *a , *b ;
{
static KOMP c;
        c.reel  = a->reel  - b->reel;
        c.sanal = a->sanal - b->sanal;
        return (( KOMP * ) ( &c ) );
}
/*****************************************************/

KOMP *komp_add( a , b )
KOMP *a , *b ;
{
static KOMP c;
        c.reel  = a->reel  + b->reel;
        c.sanal = a->sanal + b->sanal;
        return (( KOMP * ) ( &c ) );
}
/*****************************************************/

KOMP *komp_mul( a , b )
```

```
KOMP *a , *b;
{
static KOMP c;
       c.sanal  = a->reel  * b->sanal + a->sanal * b->reel;
       c.reel   = a->reel  * b->reel - a->sanal * b->sanal;
       return (( KOMP * ) ( &c ) );
}
/*****************************************************/

double komp_mag( a )
KOMP *a;              /* Calculates the magnitude of complex var a */
{
       return ( sqrt ( pow ( a->reel , 2 ) + pow ( a->sanal , 2 ) ) );

}
/*****************************************************/

double komp_aci( a )
KOMP *a;
{
       if ( a->sanal == 0 ) {
              if ( a->reel >= 0 )
                     return ( double )  PI/2;
              else
                     return ( double ) -PI/2;
       }
       return ( atan ( a->reel / a->sanal ) );
}
/*****************************************************/

calc_fft(  ary , n )
KOMP   *ary;   /* array */
int            n;            /* length of array */
{
int    p,      /* first  element of bf */
       q,      /* second element of bf */
       R,      /* Stages + 1           */
       z,
       k,
       N2,
       NH,
       KK,
       US,
       r;      /* stages               */

       R = (int ) ( log( n ) / log( 2 ) + .5 );/* N = 2^r */
       N2 = n;
       NH = n/2;

       for ( r = 0 ; r < R ; ++r ) {
              N2 = N2 / 2;
              k = 0;
```

```
                US = KK = 0;
                for ( z = 0 ; z < NH ; ++z ) {
                        p = k;
                        q = k + N2;
                        calc_BF( ary , n , KK , p , q );
                        k = k + 1;
                        if ( (z + 1)  % N2  == 0 || N2 == 1 )  {
                                k = k + N2;
                                ++US;
                                KK = N2 * bit_rev( US , r - 1 );
                        }
                }
        }
}
/*************************************************/

/*****************************/
/* Calculate a butterfly   */
/* 15.haz.91 Metin Kalayci */
/*****************************/
calc_BF ( ary ,  N , r , p , q )
KOMP    *ary;   /* complex array ;length must be N */
int             N;              /* Samples  ;Array length       */
int             r;              /* Stages   ;r = log2N -1       */
int             p;              /* Butterfly first element      */
int             q;              /* Butterfly second element     */
{
KOMP    W_ary,
                T_ary;  /* Twiddle factor storage register */

                W_ary = *komp_mul ( WW( N , r ) ,  ary + q  );
                T_ary = * ( ary +  p );
                *( ary + p ) = *komp_add ( ary + p  , &W_ary );
                *( ary + q ) = *komp_sub ( &T_ary   , &W_ary );
}
/*************************************************/

/*******************************/
/* Calculate twiddle factors */
/* 15.haz.91 Metin Kalayci   */
/*******************************/
KOMP    *WW( N , r )
int     N,      /* samples */
        r;      /* stage   */
                /* funtion returns exp( -j(2PI/N)*r ) */
{
static  KOMP Z;
                Z.reel  = cos ( 2*PI*r/N );
                Z.sanal = -sin ( 2*PI*r/N );
                return ( (KOMP *)&Z );
}
/*************************************************/
```

```
/**/
/* Reverses the bits of GG and returns the result */
/* Reversing length is S                          */
/**/

int     bit_rev( gg  , s )
int     gg,
              s;
{
int     t,tmp[10],tp,dd;
        tp = 0;
        dd = 0X0001;
        gg = gg & 0XFFFF;
        for ( t = s ; t >=0 ; --t ) {
              tmp[s-t] = 0X0000;
              tmp[s-t] =  gg & dd;
              dd = dd * 2;
              tp = tp + tmp[s-t ] *  pow( 2 , t ) / pow( 2 , s - t );
        }
        return tp;
}
/*************************************************/

/**/
/* 28.6.1991  Metin KALAYCI                                    */
/* Reads the samples from the file named Fname and returns     */
/*  reallocated array Pointer                                                */
/* Samples length is restricted with allocated memory!         */
/* each samples requires  2*( sizeof ( double ) ~16 bytes      */
/* 1 K can hold ~64 samples                                    */
/* 1024 samples requires 16 K free memory                      */
/* Format of the file  :                                       */
/* Each line represents one complex  sample separated by comma */
/* double reel part , double imaginal part                     */
/**/

KOMP    *OrnekOku( Fname  , Elem )
TEXT    *Fname;
int             *Elem;
{
COUNT   Ofset;
KOMP    *iary;
FILE    *sd;
TEXT    line[256],
              Rbuff[128],
              Kbuff[128];

        if ( ( sd = fopen ( Fname , "r" ) ) == ( FILE *) NULL ) {
              printf ( "Ornek Dosyasi Acilamadai\n" );
              exit(0);
        }
```

```
        *Elem = 0;
        iary  = ( KOMP * ) NULL;                                /* Not Allocated        */
        while ( readline ( sd , line ) != EOF ) {
                Ofset = 0;                                                      /* Search from the head  */
                Satircoz ( &Ofset , line , Rbuff );     /* Get the real part     */
                Satircoz ( &Ofset , line , Kbuff ); /* Get the imaginal part */
                iary = ( KOMP * ) realloc ( iary , ( *Elem + 1 ) * sizeof ( KOMP ) );
                if ( iary == NULL ) {
                        printf ( "Can not allocate memory for samples! exiting to DOS\n" );
                        exit(0);
                }
                ( iary +*Elem)->reel  = atof ( Rbuff );         /* define atof as double */
                ( iary +*Elem)->sanal = atof ( Kbuff );         /* in the header !       */
                (*Elem)++;
        }
        fclose ( sd );
        return ( KOMP * ) iary;
}
/*************************************************/
/**/
/* Writes the FFT of the samples  to a temporary file */
/**/

Yaz_Sonuc( r_ary , length , mwn )
KOMP    *r_ary;
int             length;
int             mwn;
{
FILE    *fd;
char    fnam[13];
char    Dadi[ _MAX_PATH ];
char    *TmpName,
                *CurDir;
int             p,
                i,
                k;


        CurDir  = getcwd( NULL , _MAX_PATH );   /* MAX_PATH defined in stdlib.h */
        if ( CurDir == NULL ) {
                printf ( "Cannot get current directory path !. exiting to DOS \n" );
                exit(0);
        }
        TmpName = tempnam( CurDir , "R_FFT" );  /* Get an inexsistent name     */
                                                                                        /* Creates it in cur
        fd = fopen ( TmpName , "w" );                   /* Each time a different file */
        if ( fd == ( FILE * ) NULL ) {
                printf ( "Temporary FFT result file can not be opened !. exiting to DOS \n" );
                exit(0);
        }
        wxatsay ( mwn , 7 , 1 , "DFT Sonuclari " , 7 );
        strcpy ( Dadi , TmpName );
```

```
        k = strlen ( Dadi );
        while ( k >= 0 && Dadi[--k] != '\\' );
        strcpy  ( fnam , Dadi + k + 1 );
        wxatsay ( mwn , 7 , 14 , fnam , 7 );
        wxatsay ( mwn , 7 , 28 , " Dosyasina Yaziliyor !...", 7 );
        p = (int ) ( log( length ) / log( 2 ) + .5 ) - 1;

        fprintf ( fd ,  "#DFT\n" );
        fprintf ( fd , "DISCRETE FOURIER TRANSFORM\n" );
        fprintf ( fd , "\n" );
        fprintf ( fd , "SAMPLES,REEL,IMAGINAL,MAGNITUDE,PHASE\n" );
        for ( i = 0 ; i < length ; ++i ) {
                k = bit_rev ( i , p );
                fprintf ( fd , "%7d->, %-6.6lf , %-6.6lf , %6.6lf, %6.6lf\n",
                                                i , r_ary[k].reel , r_ary[k].sanal , komp_mag ( r_ary + k ) ,
        }
        fclose ( fd );
}

/**/
/* Frees allocated memory */
/**/
void    FreMem( ary , length )
KOMP    *ary;
int             length;
{
        while ( length )
                free ( ( char * ) ary + length-- );
}
```

```
int  fhesap(double  huge *x,double  huge *b,double  huge *a,double  fp,double  n,double  dc,double  k);
int  mabs(double  huge *bval);
int  myfr(double  huge *x,double  huge *b,double  n,double  k);
int  yazfir001(double  huge *x,double  huge *b,double  huge *a,int  m,double  fp,double  n,double  dc,double
```

PARK-MC CELLAN. C

GENEL AMAÇLI FIR FILTRE , DIFFERENTIATOR VE HILLBERT TRANSFORMER TASARIMI

```
/*******************************/
/* PARKS-MAC CELLAN ALGORITHM */
/*******************************/
/*
inputs
        nfilt : filter length
        jtype : type of filter
        1= multiband passband/stopband
        2= diferentiator
        3= hillbert transform filter
nbands  : number of bands
edge    : ( 2*nbands )
fx(nbands) : desired funtion array in eachband ( or desired slope if a
                        differentiator ) for each band.

wtx(nbands): weight function array in each band , for a diferentiator,
                        the weight function is inversely proportional to f
*/
#define Extern /* */
#include<stdio.h>
#include<vcstdio.h>
#include<malloc.h>
#include<math.h>
#include"pmar.h"
double  atof();
void    sifirla ( double * , int );
TEXT    mes003[20][81] = {
                "MULTIBAND/DIFFERENTIATOR/HILLBERT TRANSFORMER DESIGN ",
                "FILENAME         :",
                "FILTER LENGTH    :",
                "FILTER TYPE      :",
                "BAND NO          :",
                "EDGE INPUT FOR BAND   ",
                "LOWER BAND EDGE  :",
                "UPPER BAND EDGE  :",
                "DESIRED VALUE FOR BAND",
                "DESIRED VALUE    :",
                "WEIGHTING FOR BAND    ",
                "WEIGHTING        :"
};
int             ed_dv_wg();     /* function to input band edges & desired value & weights */
TEXT    remes_nam[15];


parks()
{
        getpvar();
}

getpvar()
{
TEXT    remes_bnd[5];
```

```
TEXT    remes_typ[5];
TEXT    remes_uzn[5];
int      mwn0,
         mwn;

        mwn0 =  wxxopen( 5,
                                              12,
                                              22,
                                              66,
                                              "",
                                              ACTIVE+COOKED,
                                              0,
                                              0,
                                              4,
                                              178 );

        mwn = wopen   ( 4,
                                             10,
                                             21,
                                             64,
                                             "PARKS-MC CELLAN FIR FILTER DESIGN"
                                             );

              wxatsay ( mwn ,  1 , 0 , mes003[0], 112 );
              wxatsay ( mwn ,  3 , 1 , mes003[1], 7   );
              wxatsay ( mwn ,  4 , 1 , mes003[2], 7   );
              wxatsay ( mwn ,  5 , 1 , mes003[3], 7   );
              wxatsay ( mwn ,  6 , 1 , mes003[4], 7   );
              wxatsay ( mwn ,  9 , 1 , mes003[5], 112 );
              wxatsay ( mwn , 10 , 1 , mes003[6], 7   );
              wxatsay ( mwn , 11 , 1 , mes003[7], 7   );
              wxatsay ( mwn , 12 , 1 , mes003[8], 112 );
              wxatsay ( mwn , 13 , 1 , mes003[9], 7   );
              wxatsay ( mwn , 14 , 1 , mes003[10],112 );
              wxatsay ( mwn , 15 , 1 , mes003[11], 7  );

              empty ( remes_nam , 13 );
              empty ( remes_typ , 2 );
              empty ( remes_bnd , 3 );
              empty ( remes_uzn , 4 );

              xatget  ( 3, 25, remes_nam , "REMESXXX.FIR", NULLFUNC, "", "", vc.dflt, vc.dflt );
              xatget  ( 4, 25, remes_uzn, "999", NULLFUNC, "", "", vc.dflt, vc.dflt );
              xatget  ( 5, 25, remes_typ, "9", NULLFUNC, "", "", vc.dflt, vc.dflt );
              xatget  ( 6, 25, remes_bnd, "99", NULLFUNC, "", "", vc.dflt, vc.dflt );

              switch ( readgets() ) {
                     case    RET:
                                             nfilt  = atoi ( remes_uzn );
                                             jtype  = atoi ( remes_typ );
                                             nbands = atoi ( remes_bnd );
                                             ed_dv_wg( remes_bnd );
```

```
                                        break;
                case    ESC:
                deafult    :
                                        wclose ( mwn );
                                        wclose ( mwn0 );
                                        return 0;
                }
                getinput();
                getout( remes_nam );
                wclose ( mwn );
                wclose ( mwn0 );

}

int             ed_dv_wg( tstr )
TEXT    *tstr;
{
TEXT    remes_wt[15];
TEXT    remes_dv[15];
TEXT    remes_up[15];
TEXT    remes_low[15];
TEXT    cvsay[10];
int             cevnum,
                cvr;

        cevnum = atoi ( tstr );
        for ( cvr = 0 ; cvr < 2 * cevnum ; cvr = cvr + 2 ) {

                empty ( remes_up , 6 );
                empty ( remes_low, 6 );
                sprintf ( cvsay    , "%4d" , cvr / 2 + 1  );
                atsay   (  9       , 25    , cvsay );

                xatget  ( 10, 25, remes_low , ".9999", NULLFUNC, "", "", vc.dflt, vc.dflt );
                xatget  ( 11, 25, remes_up  , ".9999", NULLFUNC, "", "", vc.dflt, vc.dflt );

                switch ( readgets () ) {

                        case  RET:
                                edge [ cvr + 1 ] = atof ( remes_low );
                                edge [ cvr + 2 ] = atof ( remes_up  );
                                break;
                        case  ESC:
                                return 1;
                        }
        }

        for ( cvr = 1 ; cvr <= cevnum ; cvr++ ) {

                empty ( remes_dv , 9 );
                sprintf ( cvsay    , "%4d" , cvr   );
                atsay   ( 12       , 25    , cvsay );
```

```
                xatget  ( 13, 25, remes_dv , "####.###", NULLFUNC, "", "", vc.dflt, vc.dflt );
                switch ( readgets () ) {

                        case  RET:
                                fx [ cvr ] = atof ( remes_dv  );
                                break;
                        case  ESC:
                                return 1;
                        }
        }

        for ( cvr = 1 ; cvr <= cevnum ; cvr++ ) {

                empty   ( remes_wt , 9 );
                sprintf ( cvsay    , "%4d" , cvr   );
                atsay   ( 14       , 25    , cvsay );

                xatget  ( 15, 25, remes_wt , "####.###", NULLFUNC, "", "", vc.dflt, vc.dflt );
                switch ( readgets () ) {

                        case  RET:
                                wtx [ cvr ] = atof ( remes_wt  );
                                break;
                        case  ESC:
                                return 1;
                        }
        }
        return 0;
}

void    sifirla( veri  , leni )
double  *veri;
int     leni;
{
        while ( --leni >= 0 )
                *( veri + leni ) = DZERO;
}

getinput()
{
        /*********************************/
        /** calculation of pi value     */
        /*********************************/
        iout = 3;
        pi   = 4.0 * atan ( 1 );
        pi2  = 2.0 * pi;
        /*****************************/
        nfmax = 128;
        /*
        scanf ( "%d" , &nfilt );
        scanf ( "%d" , &jtype );
        scanf ( "%d" , &nbands);
```

```
*/

if ( nfilt == 0 )
                return 0;
if ( !( nfilt <= nfmax || nfilt >= 3 ) )
                return 0;
if ( nbands <= 0 )
                nbands = 1;

/***********************************/
/* grid density assumed to be 16 */
/***********************************/

lgrid = 16;
jb = 2*nbands;

/***********************************/
/* read for edge(j) j = 1 to jb  */
/***********************************/

/*
for ( j = 1 ; j <= jb ; ++ j )
        scanf ( "%lf" , &edge[ j ] );
*/
/*****************************************/
/* read for desired value for band edges  */
/*      fx(j) , j = 1, nbands            */
/*****************************************/
/*
for ( j = 1 ; j <= nbands ; ++ j )
        scanf ( "%lf" , &fx[ j ] );
for ( j = 1 ; j <= nbands ; ++ j )
        scanf ( "%lf" , &wtx[ j ] );
*/

if ( !( jtype > 0 || jtype <= 3 ) )
                return 0;
neg = 1;
if ( jtype == 1 )
                neg = 0;
nodd  = nfilt / 2;
nodd  = nfilt - 2*nodd;
nfcns = nfilt / 2;
if ( nodd == 1 && neg == 0 )
                nfcns++;
/*********************************************************/
/* setup the dense grid number of points in the grid   */
/* is ( filter lenght  + 1 ) * grid density / 2        */
/*********************************************************/

grid[1] = edge[1];
delf = (double) lgrid * nfcns;
```

```
delf = .5 / delf;
if ( neg != 0 )
        if ( edge[1] < delf )
                grid[1] = delf;
j = 1;
l = 1;
lband = 1;

/****************************************************/
/*   Calculate the desired magnitude response and   */
/*   weight function on the grid                    */
/****************************************************/

do {
        fup  = edge[l+1];

        do {
                temp = grid[j];
                des[j] = eff ( temp , fx[lband] , wtx[lband] , jtype );
                wt[j]  = wate( temp , fx[lband] , wtx[lband] , jtype );
                j++;
                grid[j]  = temp + delf;
         } while ( grid[j] <= fup );

        grid[ j-1 ] = fup;
        des [ j-1 ] = eff ( fup , fx[lband] , wtx[lband] , jtype );
        wt  [ j-1 ] = wate( fup , fx[lband] , wtx[lband] , jtype );
        lband++;
        l += 2;
        if ( lband > nbands )
                break;
        grid[j] = edge[l];

} while ( lband <= nbands );

ngrid = j-1;
if ( neg == nodd )
        if (  grid[ngrid] > ( .5 - delf ) )
                ngrid--;
switch ( neg <= 0 ) {
        case 1:
                if ( nodd == 1 )
                        break;
                for ( j = 1 ; j <= ngrid ; ++j  ){
                        change = cos ( (double )pi * grid[j] );
                        des[j] = des[j] / change;
                        wt[j]  = wt[j] * change;
                }
                break;
        case 0:
                if ( nodd == 1 ) {
                        for ( j = 1 ; j <= ngrid ; ++j  ){
```

```
                        change = sin ( (double ) pi2 * grid[j] );
                        des[j] = des[j] / change;
                        wt [j] = wt[j] * change;
                        }
                } else {
                        for ( j = 1 ; j <= ngrid ; ++j  ){
                                change = sin ( (double ) pi * grid[j] );
                                des[j] = des[j] / change;
                                wt [j] = wt[j] * change;
                        }
                }
                break;
        default:
                break;
}
temp = (double) ( ngrid - 1 ) / nfcns;
for ( j = 1 ; j <= nfcns ; ++j ) {
        xt = j - 1;
        iext[j] = xt * temp + 1.0;
}
iext[ nfcns + 1 ] = ngrid;
nml = nfcns-1;
nz  = nfcns+1;

/**
call remes exchange algorithm  to the approximation problem
**/

remes();

/* Calculate the impulse response */

switch ( neg <= 0 ) {
        case 1:
                if ( nodd == 0 )  {
                        h[1] = (double)0.25 * alpha[nfcns];
                        for ( j = 2 ; j <= nml ; ++j ) {
                                nzmj = nz - j;
                                nf2j = nfcns + 2 - j;
                                h[j] = (double)0.25 * ( alpha[nzmj] + alpha[nf2j] );
                        }
                        h[nfcns] = (double)0.5 * alpha[1] + .25 * alpha[2];
                } else {
                        for ( j = 1 ; j <= nml ; ++j ) {
                                nzmj = nz - j;
                                h[j] = (double)0.5 * alpha [ nzmj ];
                        }
                        h[nfcns] = alpha[1];
                }
                break;
        case 0:
                if ( nodd == 0 ) {
```

```
                        h[1] = (double).25 * alpha[nfcns];
                        for ( j = 2 ; j <= nml ; ++j ) {
                                nzmj = nz - j;
                                nf2j = nfcns + 2 - j;
                                h[j] = (double).25 * ( alpha[nzmj] - alpha[nf2j] );
                        }
                        h[nfcns] = (double).5 * alpha[1] - .25 * alpha[2];
                } else {
                        h[1] = (double)0.25 * alpha[nfcns];
                        h[2] = (double)0.25 * alpha[nml];
                        for ( j = 3 ; j <= nml ; ++j ) {
                                nzmj = nz - j;
                                nf3j = nfcns + 3 - j;
                                h[j] =(double) 0.25 * (alpha[nzmj] - alpha[nf3j]);
                        }
                        h[nfcns] = (double)0.5 * alpha[1] - 0.25 *alpha[3];
                        h[nz] = (double)0.0;
                }
                break;
        default:
                break;
    }
}
/*****************/
/* REMES         */
/*****************/

remes ()
{
double          xe;
int             jml,
                jpl,
                nflj,
                nzz,
                nzzmj,
                jxt;
double  itrmax,
                a [ 67 ],
                dak,
                dk,
                p [ 67 ],
                q [ 67 ],
                dtemp,
                jet,
                dnum,
                dden,
                nu,
                devl,
                jchnge,
                kl,
                knz,
                klow,
```

```
        nut,
        comp,
        err,
        nut1,
        ynz,
        ymy,
        luck,
        kn,
        fsh,
        gtemp,
        cn,
        kkk,
        aa,
        bb,
        ft,
        xt1;

    itrmax = 25;
    devl = (double) -1.0;
    nz  = nfcns + 1;
    nzz = nfcns + 2;
    niter = DZERO;
A100:
    iext [ nzz ] = ngrid + 1;
    niter++;
    if ( niter > itrmax )
                goto A400;
    for ( j = 1 ; j <= nz ; ++j ) {
        jxt = iext [ j ];
        dtemp = grid [ jxt ];
        dtemp = cos ( (double )dtemp * pi2 );
        x[j] = dtemp;
    }
    jet = ( nfcns - 1 ) / 15 + 1;
    for ( j = 1 ; j <= nz ; ++j )
        ad[ j ] = dval ( j , nz , ( int ) jet );
    dnum = DZERO;
    dden = DZERO;
    k = 1;
    for ( j = 1 ; j <= nz ; ++j ) {
        l = iext [ j ];
        dtemp = ( double ) ( ad[ j ] * des [ l ] );
        dnum  = ( double ) ( dnum + dtemp );
        dtemp = ( double ) k * ad [ j ] / wt[ l ];
        dden  = dden + dtemp;
        k = -k;
    }
    dev = ( double ) ( dnum / dden );
    nu  = ( double ) 1;
    if ( dev > DZERO )
                nu =( double ) -1;
    dev = ( double ) -nu * dev;
```

```
        k = ( int ) nu;
        for ( j = 1 ; j <= nz ; ++j ) {
                l = iext [ j ];
                dtemp = k * dev / wt [ l ];
                y[ j ] = des [ l ] + dtemp;
                k = -k;
        }
        if ( dev > devl )
                        goto A150;
        ouch();
        goto A400;
A150:
        devl = dev;
        jchnge = 0;
        kl = iext [ 1 ];
        knz = iext[ nz ];
        klow = 0;
        nut = -nu;
        j = 1;
A200:
        if ( j == nzz )
                        ynz = comp;
        if ( j >= nzz )
                        goto A300;
        kup = iext[ j + 1 ];
        l = iext [ j ] + 1;
        nut = -nut;
        if ( j == 2 )
                ymy = comp;
        comp = dev;
        if ( l >= kup )
                        goto A220;
        err = gee ( l , nz );
        err = ( err - des [ l ] ) * wt [ l ];
        dtemp = nut * err - comp;
        if ( dtemp <= DZERO )
                        goto A220;
        comp = nut  * err;
A210:
        l++;
        if ( l >= kup )
                        goto A215;
        err = gee ( l , nz );
        err = ( err - des [ l ] ) * wt [ l ];
        dtemp = nut * err - comp;
        if ( dtemp <= DZERO )
                        goto A215;
        comp = nut  * err;
        goto A210;
A215:
        iext[j] = l - 1;
        j++;
```

```
        klow = l -1;
        jchnge++;
        goto A200;
A220:
        l--;
A225:
        l--;
        if ( l <= klow )
                        goto A250;
        err = gee ( l , nz );
        err = ( err - des [ l ] ) * wt [ l ];
        dtemp = nut * err - comp;
        if ( dtemp > DZERO )
                        goto A230;
        if ( jchnge <= DZERO )
                        goto A225;
        goto A260;
A230:
        comp = nut  * err;
A235:
        l--;
        if ( l <= klow )
                        goto A240;
        err = gee ( l , nz );
        err = ( err - des [ l ] ) * wt [ l ];
        dtemp = nut * err - comp;
        if ( dtemp <= DZERO )
                        goto A240;
        comp = nut * err;
        goto A235;
A240:
        klow = iext [ j ];
        iext[ j ] = l + 1;
        j++;
        jchnge++;
        goto A200;
A250:
        l = iext [ j ] + 1;
        if ( jchnge > DZERO )
                goto A215;
A255:
        l++;
        if ( l >= kup )
                goto A260;
        err = gee ( l , nz );
        err = ( err - des [ l ] ) * wt [ l ];
        dtemp = nut * err - comp;
        if ( dtemp <= DZERO )
                        goto A255;
        comp = nut * err;
        goto A210;
A260:
```

```
        klow = iext [ j ];
        j++;
        goto A200;
A300:
        if ( j > nzz )
                goto A320;
        if ( kl > iext [ 1 ] )
                        kl = iext[ 1 ];
        if ( knz < iext [ nz ] )
                        knz = iext[ nz ];
        nutl = nut;
        nut  = -nu;
        l = 0;
        kup = kl;
        comp = ynz * (double) ( 1.00001 );
        luck = 1;
A310:
        l++;
        if ( l >= kup )
                goto A315;
        err = gee ( l , nz );
        err = ( err - des [ l ] ) * wt [ l ];
        dtemp = nut * err - comp;
        if ( dtemp <= DZERO )
                        goto A310;
        comp = nut * err;
        j = nzz;
        goto A210;
A315:
        luck = 6;
        goto A325;
A320:
        if ( luck > 9 )
                        goto A350;
        if ( comp > ymy )
                        ymy = comp;
        kl = iext [ nzz ];
A325:
        l= ngrid + 1;
        klow = knz;
        nut = -nutl;
        comp = ymy * (double)( 1.00001 );
A330:
        l--;
        if ( l <= klow )
                        goto A340;
        err = gee ( l , nz );
        err = ( err - des [ l ] ) * wt [ l ];
        dtemp = nut * err - comp;
        if ( dtemp <= DZERO )
                        goto A330;
        j = nzz;
```

```
        comp = nut * err;
        luck = luck + 10;
        goto A235;
A340:
        if ( luck == 6 )
                        goto A370;
        for ( j = 1; j <= nfcns ; ++j ) {
                nzzmj = nzz - j;
                nzmj  = nz - j;
                iext[ nzzmj ] = iext [ nzmj ];
        }
        iext[ 1 ] = kl;
        goto A100;
A350:
        kn = iext [ nzz ];
        for ( j = 1 ; j <= nfcns ; ++j )
                iext[ j ] = iext [ j + 1 ];
        iext [ nz ] = kn;
        goto A100;
A370:
        if ( jchnge > DZERO )
                goto A100;
A400:
        nml = nfcns - 1;
        fsh = (double)0.000001;
        gtemp = grid [ 1 ];
        x [ nzz ] = (double)-2.0;
        cn = (double) ( 2 * nfcns - 1 );
        delf = ( double ) 1.0 / cn;
        l = 1;
        kkk = 0;
        if ( ( edge[ 1 ] == ( double ) 0.0 ) && ( edge[2*nbands] ==( double ) 0.5  ) )
                        kkk = 1;
        if ( nfcns <= 3 )
                        kkk = 1;
        if ( kkk == 1 )
                        goto A405;
        dtemp = cos ( (double )pi2 * grid[ 1 ] );
        dnum  = cos ( (double )pi2 * grid[ ngrid ] );

        aa  = (double ) 2.0 / ( dtemp - dnum );
        bb  = (double ) - ( dtemp + dnum ) / ( dtemp - dnum );

A405:
        for ( j = 1 ; j <= nfcns ; ++j ) {
                ft = j - 1;
                ft = ft * delf;
                xt = (double) cos ( (double )pi2 * ft );
                if ( kkk == 1 )
                        goto A410;
                xt  = (double) ( xt - bb ) / aa;
                xt1 = (double) sqrt ( (double) (1.0 - xt * xt  ) );
```

```
            ft  = (double)atan2 ( (double)xt1 ,(double)xt ) / pi2;
A410:
            xe = x[ l ];
            if ( xt > xe )
                        goto A420;
            if ( ( xe - xt )  < fsh )
                        goto A415;
            l++;
            goto A410;
A415:
            a[j] = y[ l ];
            goto A425;
A420:
            if ( ( xt - xe ) < fsh )
                        goto A415;
            grid [ 1 ] = ft;
            a[ j ] = gee ( 1 , nz );

A425:
            if ( l > 1 )
                  l--;
      }
      grid[ 1 ] = gtemp;
      dden = ( double ) pi2 / cn;
      for ( j = 1 ; j <= nfcns ; ++j ) {
            dtemp = DZERO;
            dnum = (double) (j - 1);
            dnum = dnum * dden;
            if ( nml < 1 )
                  goto A505;
            for ( k = 1; k <= nml ; ++k ){
                  dak = a [ k + 1 ];
                  dk  = (double)k;
                  dtemp = dtemp + dak * cos ( (double )dnum * dk );
            }
A505:
      dtemp = (double)2.0 * dtemp + a [ 1 ];
A510:
      alpha [ j ] = dtemp;
      }

      for ( j =  2 ; j <= nfcns ; ++j )
                  alpha [ j ] = 2.0 * alpha [ j ] / cn;
      alpha [ 1 ] = alpha [ 1 ] / cn;
      if ( kkk == 1 )
            goto A545;
      p[ 1 ] = 2.0 * alpha [ nfcns ] * bb + alpha [ nml ];
      p[ 2 ] = 2.0 * aa * alpha [ nfcns ];
      q[ 1 ] = alpha[ nfcns - 2 ] - alpha [ nfcns ];
      for ( j = 2 ; j <= nml ; ++j ) {
            if ( j < nml  )
                  goto A515;
```

```
                aa = .5 * aa;
                bb = .5 * bb;
A515:
                p[ j + 1 ] = DZERO;
                for ( k = 1 ; k <= j ; k++ ) {
                        a [ k ] = p [ k ];
A520:                   p[ k ] = 2.0 * bb * a[ k ];
                }
                p[ 2 ] = p[ 2 ] + a [ 1 ] * 2.0 * aa;
                jm1 = j - 1;
A525:
                for ( k = 1; k <= jm1 ; ++k )
                        p[ k ] = p [ k ] + q [ k ] + aa * a[ k + 1 ];

A530:
                jp1 = j + 1;
                for ( k = 3; k <= jp1 ; ++k )
                        p[ k ] = p [ k ] + aa * a[ k - 1 ];

                if (  j == nm1  )
                        continue;
                for ( k = 1 ; k <= j ; ++k )
                        q [ k ] = -a[ k ];
                nflj = nfcns - 1 - j;
                q[ 1 ] = q [ 1 ] + alpha [ nflj ];
        }
        for ( j = 1 ; j <= nfcns ; ++j )
                        alpha [ j ] = p [ j ];
A545:
        if ( nfcns > 3 )
                return 0;
        alpha [ nfcns + 1 ] = DZERO;
        alpha [ nfcns + 2 ] = DZERO;
        return 0;
}
double  fcevi[100];
/***************/
/* D         */
/***************/

double  dval( k1 , n1 , m1 )
int     k1,
        n1,
        m1;
{
int     l1;
int     jp1;
double  d1;
double  q1;

        d1 = (double)1.0;
        q1 = x[ k1 ];
```

```
        for ( l1 = 1 ; l1 <= m1 ; ++l1 ){
                for ( jp1 = l1 ; jp1 <= n1 ; jp1 = jp1 + m1 ) {
                        switch ( ( jp1 - k1 )  != 0 ) {
                                case 1:
                                        d1 = (double) 2.0 * d1 * ( q1-x[ jp1 ] );
                                        break;
                                case 0:
                                default:
                                        break;
                                }
                        }
        }
        d1 = (double) ( 1.0 / d1 );
        return ( double ) d1;
}

/*****************/
/* GEE           */
/*****************/

double  gee ( k2 , n2 )
int             k2,
                n2;
{
int             jp1;
double  p2,
                c2,
                xf,
                d2;

                p2 = DZERO;
                xf = grid [ k2 ];
                xf = cos ( (double )pi2 * xf );
                d2  = DZERO;
                for ( jp1 = 1 ; jp1 <= n2 ; ++jp1 ) {
                        c2 = xf - x[jp1];
                        c2 = ad[ jp1 ] / c2;
                        d2 = d2 + c2;
                        p2 = p2 + c2 * y[ jp1 ];
                }
                return (double) ( p2 / d2 );
}

/***************/
/* OUCH       *
/***************/

void    ouch()
{
        printf ( "probale cause is machine rounding error\n" );
        printf ( "number of iterations = %lf " , niter        );
        printf ( "If the number of iterations exceeds 3\n"   );
```

```
        printf ( "the design may be correct, but should be verified\n"   );
}

/*********************/
/* FUNCTIONS         */
/*********************/

/*********************/
/* EFF               */
/*********************/

double  eff ( freq , fx2 , wtx2 , jtyp )
double  freq;
int             jtyp;
double  fx2 ,
                wtx2;
{
                if ( jtyp == 2 )
                                return ( double) ( fx2 * freq );
                else
                                return ( double ) fx2;
}

/*********************/
/* WATE              */
/*********************/
double  wate ( freq , fx1 , wtx1 , jtyp )
double  freq;
int             jtyp;
double  fx1,
                wtx1;
{
double  wat;

                if ( jtyp != 2 )
                                return ( double ) wtx1;
                if ( fx1 < (double) .0001 )
                                return ( double ) wtx1;
                else
                                return ( double ) ( wtx1 / freq );
}

/*****************/
/* ERROR         */
/*****************/

void error()
{
        printf ( "error in input data....!\n" );
}
/****************************/
/* PROGRAM OUTPUT SECTION */
```

```
/**************************/
getout( nam )
TEXT    *nam;
{
FILE    *fd;
double  psvar;
double  fr;

                fd = fopen ( nam , "w" );
                if ( fd == ( FILE * ) NULL ) {
                        printf ( "Dosya acma hatasi .... Exiting to DOS!\n" );
                        exit(0);
                }

                fprintf ( fd , "#COEFFICENTS\n" );
                if ( neg  == 0 )
                                fprintf ( fd , "BANDPASS FIR FILTER " );
                else
                                fprintf ( fd , "DIFFERENTIATOR - HILLBERT" );
                if ( nodd == 0 )
                                fprintf ( fd , "UZUNLUK CIFT" );
                else
                                fprintf ( fd , "UUZUNLUK TEK" );
                fprintf ( fd , "\nCOEFFICENT-NO,\n" );
                fprintf ( fd , "AMPLITUDE,\n" );

                if ( neg == 1 && nodd == 1 )
                        fprintf ( fd , "0.0\n" );
                for (  i = 1 ; i <= nfcns ; ++i )
                        hh[i] = h[i];
                if ( neg == 0 && nodd == 0 )  {
                        for ( n = 1 ; n <= nfcns ; ++n )
                                hh[nfcns+n] = h[nfcns-n+1];
                        for      ( i = 1 ; i <= nfcns * 2 ; ++i )
                                fprintf ( fd , "%6.0d,%3.91f\n" , i , hh[i] );
                } else if ( neg ==  0 && nodd == 1 ) {
                                for ( n = 1 ; n <= nfcns - 1 ; ++n )
                                        hh[ nfcns+n ] = h[ nfcns - n ];
                                for ( i =  1; i <= nfcns * 2 - 1 ; ++i )
                                        fprintf ( fd , "%6.0d,%3.91f\n" , i , hh[i] );
                } else if ( neg == 1 && nodd == 0 ) {
                                for ( n = 1 ; n <= nfcns ; ++n )
                                        hh[nfcns+n] = -h[nfcns-n+1];
                                for ( i = 1 ; i <= nfcns *2 ; ++i )
                                        fprintf ( fd , "%6.0d,%3.91f\n" , i , hh[i] );
                } else if ( neg == 1 && nodd == 1 ) {
                                for ( n = 1 ; n <= nfcns ; ++n )
                                        hh[nfcns+n+1] = -h[nfcns-n+1];
                                for ( i = 1 ; i <= nfcns *2 + 1 ; ++ i )
                                        fprintf ( fd , "%6.0d,%3.91f\n" , i , hh[i] );
                }
                fprintf ( fd , "))\n" );
```

```
fre_cev(  &hh[0]  , &fcevi[0] , nfilt , 80 );
fprintf ( fd , "#FREQUENCY RESP( dB )\n" );
fprintf ( fd , "FREQUENCY RESPONSE REMES IN DB\n\n" );
fprintf ( fd , "FREQUENCY,AMPLITUDE,\n" );
for ( k = 1 ; k <= 80 ; ++k )  {
              fr = (double) .5 * ( k - 1 ) / 80;
              if ( fcevi[ k ] < DZERO )
                            fcevi[ k ] = -fcevi[ k ];
              psvar = 20 * log10 ( fcevi[ k ] );
              fprintf ( fd , "%4.10lf , %4.10lf\n" , fr , psvar );
}
fprintf ( fd , "))\n" );

fprintf ( fd , "#FREQUENCY RESPONSE\n" );
fprintf ( fd , "FREQUENCY RESPONSE REMES\n\n" );
fprintf ( fd , "FREQUENCY,AMPLITUDE,\n" );
for ( k = 1 ; k <= 80 ; ++k )  {
              fr = (double) .5 * ( k - 1 ) / 80;
              fprintf ( fd , "%4.10lf , %4.10lf\n" , fr , fcevi[k] );
}
fprintf ( fd , "))\n" );

fprintf ( fd , "#HEADER\n" );
fprintf ( fd , "finite impulse response (fir)\n" );
fprintf ( fd , "linear phase digital filter design\n" );
fprintf ( fd , "remez exchange algorithm\n" );
switch ( jtype ) {
       case 1:
                     fprintf ( fd , "Bandpass Filter\n" );
                     break;
       case 2:
                     fprintf ( fd , "Differentiator\n" );
                     break;
       case 3:
                     fprintf ( fd , "Hillbert Transformer\n" );
                     break;
       default:
                     break;
}

fprintf ( fd , "filter length = %5d\n" , nfilt );
for ( j = 1 ; j <= nbands ; ++j )
       fprintf ( fd , "lower band edge[%5d]=,%3.91f, upper band edge=[%5d]=, %3.91f\n" , j ,
       for ( j = 1 ; j <= nbands ; ++j )  {
              if ( jtype != 2 )
                     fprintf ( fd , "Desired value[%d]=" , j );
              else
                     fprintf ( fd , "Desired slope[%d]=" , j );
              fprintf ( fd , ", %3.91f \n" ,  fx[j]);
       }
fprintf ( fd ,"\n Weights\n" );
```

```
                for ( j = 1 ; j <= nbands ; ++j )
                        fprintf ( fd , "[%5d] , %3.9lf\n" , j , wtx[j] );
                fprintf ( fd ,"\ndeviations\n" );
                for ( j = 1 ; j <= nbands ; ++j ) {
                                deviat[j] = dev / wtx[j];
                                fprintf ( fd , "%6d , %3.9lf\n" , j , deviat[j] );
                }
                fprintf ( fd ,"\ndeviations in dB\n" );
                for ( j = 1 ; j <= nbands ; ++j ) {
                        deviat[j] = 20.0 * log10 ( deviat[j] );
                        fprintf ( fd , "%6.0d , %3.9lf \n" , j , deviat[j] );
                }
                fprintf ( fd , "))\n" );
                for ( j = 1 ; j <= nz ; ++j ) {
                        ix = ( int ) iext[j];
                        grid [ j ] = grid[ ix ];
                }
                fprintf ( fd , "#EXTREMAL FREQUENCIES\n" );
                fprintf ( fd , "EXTREMAL FREQUENCIES\n" );
                fprintf ( fd , "BAND-NO,\n" );
                fprintf ( fd , "FREQUENCY,\n" );
                for ( j = 1 ; j <= nz ; ++j )
                        fprintf ( fd , "%5d , %3.9lf \n" , j , grid[j] );
                fprintf ( fd , "))\n" );

                fclose ( fd );
}

fre_cev( fr_x, fr_b, fr_n, fr_k )
double  *fr_x,          /** array of filter coefficents **/
                *fr_b;          /** array of frequency response **/
int             fr_n,           /** filter length **/
                fr_k;           /** fresponse number **/
{
double  fr_q,
                fr_am,
                fr_at,
                fr_f;
int             fr_i,
                fr_j,
                fr_m,
                fr_n2,
                fr_c1;

                fr_c1 = fr_n /2;
                fr_c1 = fr_n - 2*fr_c1;
                fr_q  = (double) (3.141592654 / fr_k);
            fr_am = ( fr_n +1 ) *.5;
                fr_m  =  ( fr_n +1 ) /2;
                fr_n2 = fr_n/2;
                for ( fr_j = 1 ; fr_j <= fr_k+1 ; ++fr_j ) {
                        fr_at = (double)0;
```

```
                if ( fr_c1 )
                        fr_at = .5 * fr_x[fr_m];
                for ( fr_i = 1 ; fr_i <= fr_n2 ; ++fr_i )
                        fr_at = fr_at + fr_x[fr_i] * cos (  (double) (fr_q*(fr_am-fr_i)*(fr_j-1) ) );
                fr_b[fr_j] = 2*fr_at;
        }
}
```

```
#define DZERO (double) 0
#define TEXT    unsigned char
void    ouch();
double  gee ( int , int );
double  dval( int , int , int );
double  eff ( double , double , double , int );
double  wate( double , double , double , int );
/************/
/* GLOBALS */
/************/

/* INTEGERS */
Extern  static  int     nfmax,
                nml,
                l,
                n,
                k,
                ix,
                nz,
                j,
                i,
                nzmj,
                nf2j,
                nf3j,
                lgrid,
                jb,
                kup,
                neg,
                jtype,
                lband,
                nodd,
                nfilt,
                nfcns,
                ngrid,
                nbands;
/* DOUBLES  */
Extern  static  double  xt,
                temp,
                fup,
                change,
                delf,
                pi2,
                pi,
                dev,
                niter,
                iout,
                iext[100],
                ad[100],
                alpha[100],
                x[100],
                y[100],
                h[100],
```

```
hh[256],
des[1250],
grid[1250],
wt[1250],
edge[25],
fx[25],
wtx[25],
deviat[25];
```

DİFERENT.C

Differentiator tasarım programıdır.

```
int  main(void);
int  diferent(double  huge *x,double  *b,double  n,double k);
int  myfr(double  huge *x,double  huge *b,double  n,double  k);
int  yazfir005(double  huge *x,double  huge *b,int  m,double  n,double  k);
int  mabs(double  huge *bval);
```

```
#include<math.h>
#include<utl2dfn.h>
#include<utl2def.h>
#include<vcstdio.h>
#include"diferent.h"
huge *halloc();

extern  double  atof();
#define DZERO (double)0


TEXT    fuzn[5],
                frcv[7],
                dadi[13];

TEXT    mes005[4][81] = {
                "          LS DIFFERENTIATOR DESIGN                ",
                "Dosya Adi            :",
                "Filtre Uzunlugu      :",
                "Frekans Cevabi Sayisi :"
                };



main()
{
int     mwn;
double  huge *x,                /* array of coeff */
                huge *b;                /* array of fresp */
double  n,              /* filter length */
                k;              /* # of fresponse calculated */
COUNT   girdi;
int     i;
int     mwn0;

                empty ( fuzn , 5  );
                empty ( frcv , 7  );
                empty ( dadi , 13 );

        mwn0 =  wxxopen( 5,
                                                12,
                                                20,
                                                66,
                                                "",
                                                ACTIVE+COOKED,
                                                0,
                                                0,
                                                4,
                                                178 );

        mwn = wopen   ( 4,
                                                10,
```

```
                                        19,
                                        64,
                                        "ALCAKGECIREN FILTRE TANIMLAMA"
                                        );

                wxatsay ( mwn ,  1 , 0 , mes005[0], 112 );
                wxatsay ( mwn ,  3 , 1 , mes005[1], 7 );
                wxatsay ( mwn ,  5 , 1 , mes005[2], 7 );
                wxatsay ( mwn ,  7 , 1 , mes005[3], 7 );

                xatget  ( 3, 25, dadi, "DIFFRXXX.DIF", NULLFUNC, "", "", vc.dflt, vc.dflt );
                xatget  ( 5, 25, fuzn, "9999", NULLFUNC, "", "", vc.dflt, vc.dflt );
                xatget  ( 7, 25, frcv, "999999" , NULLFUNC, "", "", vc.dflt, vc.dflt );
                girdi = 0;
                switch ( readgets() ) {
                        case    RET:
                                        k  = atof ( frcv );
                                        n  = atof ( fuzn );
                                        b  = x = ( double * ) NULL;
                                        b  = (double huge * ) halloc ( (long)(k+1) , sizeof ( double )
                                        if ( b == ( double * ) NULL )  {
                                                        printf ( "b=null " );
                                                        exit(0);
                                                }
                                        x  = (double huge * ) halloc ( (long)(n/2+2) , sizeof ( double
                                        if ( x == ( double * ) NULL )  {
                                                        printf ( "x=null " );
                                                        exit(0);
                                                }
                                        diferent( x , b , n, k );
                                        girdi = 1;
                                        break;
                        case    ESC:
                        deafult    :
                                        break;
                }
                if ( girdi ) {
                        hfree ( (char *) b );
                        hfree ( (char *) x );
                }
                wclose ( mwn );
                wclose ( mwn0 );
}


diferent( x , b, n , k )
double  huge    *x,
                        *b;
double  n,              /* fir filter length */
                k;              /* freq. calc # of claculted points */
{
double  am;
```

```
int             m,
                n2;
double  p,
                q;
int             j;

        m  = ( n+1 ) / 2;
        am = ( n+1 ) / 2;
        n2 = n/2;
        p  = 4 * atan ( 1 );
        if ( m == am ) {        /* For odd  N */
                for ( j = 1 ; j <= m-1 ; ++j )
                        x[j] = cos ( p * ( m-j )) / ( m-j );
        } else {                        /* For even N */
                for ( j = 1 ; j <= n2 ; ++j )  {
                        q = p * ( j - am );
                        x[j] = sin ( q ) / ( q*(j-am) );
                }
        }
        myfr( x , b  , n , k );
        yazfir005( x, b, m, n, k );
}
/*****************************************/
/*** CALCULATION OF FREQUENCY RESPONSE ***/
/*****************************************/
myfr( x, b, n, k )
double  huge *x,                /** array of filter coefficents **/
                huge *b;                /** array of frequency response **/
double  n,                              /** filter length **/
                k;                              /** fresponse number **/
{
double  q,
                am,
                at,
                f;
int             i,
                j,
                m,
                n2,
                c1;

                c1 = n /2;
                c1 = n - 2*c1;
                q  = (double) (3.141592654 / k);
            am = ( n +1 ) *.5;
                m  =  ( n +1 ) /2;
                n2 = n/2;
                for ( j = 1 ; j <= k+1 ; ++j ) {
                        at = (double)0;
                        if ( c1 )
                                at = .5 * x[m];
                        for ( i = 1 ; i <= n2 ; ++i )
```

```
                        at = at + x[i] * sin (  (double) (q*(am-i)*(j-1) ) );
                b[j] = 2*at;
            }
}
/*****************************************/
/*** ENF OF CALCULATION              ***/
/*****************************************/

yazfir005( x, b, m, n , k )
double  huge *x,
            huge *b;
int         m;
double  k,
            n;
{
FILE    *fd;
int         j;
double  f;

            fd = fopen ( dadi , "w" );          /** DOSYAYI AC  **/
            if ( fd == ( FILE * ) NULL ) {
                    printf ( "Dosya Acma Hatasi !\n");
                    exit(0);
            }
            /***************/
            fprintf ( fd , "#HEADER\n\n" ); /** HEADER YAZ **/
            fprintf ( fd , "FREKANS ORNEKLEME METODU\n" );
            fprintf ( fd , "LINEAR PHASE DIFFERENTIATOR\n\n\n" );
            fprintf ( fd , "FILTRE UZUNLUGU         : %-5.0lf\n\n"  , n  );
            fprintf ( fd , "HESAPLANAN FRE.CEV.SAY. : %-6.0lf\n\n"  , k  );
            fprintf ( fd , "))\n" );            /** END OF HEADER **/
            /***************/

            /***************/
            fprintf ( fd , "#COEFFICENTS\n" );      /** COEFFICENTS **/
            fprintf ( fd , "FOURIER COEFFICENTS\n" );
            fprintf ( fd , "COEFFICENT-NO,\n" );
            fprintf ( fd , "AMPLITUDE,\n" );
            for ( j = 1 ; j <= m ;++j )
                                fprintf ( fd , "%d,%-6.6lf\n" , j,x[j] );
            fprintf ( fd , "))\n" );                    /** END OF COEFFICENTS **/
            /***************/

            /***************/
            fprintf ( fd , "#FREQUENCY RESPONSE\n" );       /** FREQUENCY RESPONSE **/
            fprintf ( fd , "LENGTH %-5.0lf DIFFERENTIATOR LS DESIGN\n" , n );
            fprintf ( fd , "FREQUENCY,\n" );
            fprintf ( fd , "AMPLITUDE,\n" );
            for ( j = 1 ; j <= k+1 ; ++j ) {
                        f = (double).5 * (j-1)/k;
                        mabs( &b[j] );
                        fprintf ( fd , "%-6.6lf , %-6.6lf\n" , f , b[j] );
```

```
            }
            fprintf ( fd , "))\n" );                    /** END OF FREQUENCY RESPONSE **/
            /***************/

            fprintf ( fd , "))\n" );                    /** END OF FREQUENCY RESPONSE **/
            fclose ( fd );
}
/*********************************************/
/** absender function of frequency reponse **/
/*********************************************/
mabs( bval )
double  huge *bval;
{
       if ( *bval < (double)0 )
                    (*bval) = -(*bval);
}
/*********************************************/
/** end of absender function             **/
/*********************************************/
```

LSWIN.C

Window kullanarak LS kriterine göre FIR filtre tasarımı yapan programdır. Tanımlı window fonksiyonları Bartlett, Hanning , Hamming , Kaiser , Rectangular widow'dur.

```
int  main(void);
int  ls(double  huge *x,double n,double  fp);
int  wind(double  huge *x,double n,int  tp,double  betas);
double  fio(double  z);
int  myfr(double  huge *x,double  huge *b,double  n,double  k);
int  yazfir004(double  huge *x,double  huge *b,double  huge *a,int  m,double  fp,double  n,int  typs,double  l
int  mabs(double  huge *bval);
```

```
#include<math.h>
#include<utl2dfn.h>
#include<utl2def.h>
#include<vcstdio.h>
#include"lswin.h"
huge *halloc();
double  fio();
TEXT    filtname[5][20] = { "Bartlett",
                                        "Hanning",
                                        "Hamming",
                                        "Blackmann",
                                        "Kaiser" };


extern  double  atof();
#define DZERO (double)0


TEXT    fuzn[5],
                tipi[2],
                kosf[13],
                frcv[7],
                bsta[5],
                dadi[13];

TEXT    mes004[8][81] = {
                "FREKANS ORNEKLEME METODU-OPTIMAL LEAST SQUARES ERROR ",
                "Dosya Adi              :",
                "Filtre Uzunlugu        :",
                "Kose Frekansi          :",
                "Pencere Tipi ( 1-5 )   :",
                "Beta ( Kaiser icin )   :",
                "Frekans Cevabi Sayisi :"
                };

main()
{
int     mwn;
double  huge *x,                /* array of coeff */
                huge *b,                /* array of fresp */
                huge *a;                /* array of samps */
int     typs;                   /* window type    */
double  fp,             /* edge freq      */
                n,
                k;              /* # of fresponse calculated */
COUNT   girdi;
int             i,
                m,
                mwn0;
double  beta;

                vcstart( CLRSCRN );
```

```
        empty ( fuzn , 5  );
        empty ( tipi , 2  );
        empty ( kosf , 5  );
        empty ( frcv , 7  );
        empty ( bsta , 5  );
        empty ( dadi , 13 );

mwn0 =  wxxopen( 5,
                                        12,
                                        20,
                                        66,
                                        "",
                                        ACTIVE+COOKED,
                                        0,
                                        0,
                                        4,
                                        178 );

mwn = wopen   ( 4,
                                       10,
                                       19,
                                       64,
                                       "ALCAKGECIREN FILTRE TANIMLAMA"
                                       );

        wxatsay ( mwn ,  1 , 0 , mes004[0], 112 );
        wxatsay ( mwn ,  3 , 1 , mes004[1], 7 );
        wxatsay ( mwn ,  5 , 1 , mes004[2], 7 );
        wxatsay ( mwn ,  7 , 1 , mes004[3], 7 );
        wxatsay ( mwn ,  9 , 1 , mes004[4], 7 );
        wxatsay ( mwn , 11 , 1 , mes004[5], 7 );
        wxatsay ( mwn , 13 , 1 , mes004[6], 7 );

        xatget  ( 3, 25, dadi, "LSWINXXX.FIR", NULLFUNC, "", "", vc.dflt, vc.dflt );
        xatget  ( 5, 25, fuzn, "9999", NULLFUNC, "", "", vc.dflt, vc.dflt );
        xatget  ( 7, 25, kosf, ".###", NULLFUNC, "", "",  vc.dflt, vc.dflt );
        xatget  ( 9, 25, tipi, "9", NULLFUNC, "", "", vc.dflt, vc.dflt );
        xatget  (11, 25, bsta, "99", NULLFUNC, "", "", vc.dflt, vc.dflt );
        xatget  (13, 25, frcv, "999999" , NULLFUNC, "", "", vc.dflt, vc.dflt );
        girdi = 0;
        switch ( readgets() ) {
               case    RET:
                                       fp = atof ( kosf );
                                       typs = atoi ( tipi );
                                       k  = atof ( frcv );
                                       n  = atof ( fuzn );
                                       beta = atof ( bsta );

                                       b  = x = a = ( double * ) NULL;
                                       b  = (double huge * ) halloc ( (long)(k+1) , sizeof ( double )
                                       if ( b == ( double * ) NULL )  {
                                                   printf ( "b=null " );
```

```
                                        exit(0);
                                }
                        x  = (double huge * ) halloc ( (long)(n/2+2) , sizeof ( doubl
                        if ( x == ( double * ) NULL )  {
                                        printf ( "x=null " );
                                        exit(0);
                                }
                        a  = (double huge * ) halloc ( (long)(n/2+2) , sizeof ( doubl
                        if ( a == ( double * ) NULL )  {
                                        printf ( "a=null " );
                                        exit(0);
                                }
                        ls  ( x , n , fp );
                        wind( x , n , typs , beta );
                        myfr( x , b, n, k);
                        m = ( n+1 ) /2;
                        yazfir004( x, b, a , m, fp , n , typs , k );
                        girdi = 1;
                        break;
                case    ESC:
                deafult    :

                        break;
        }
        if ( girdi ) {
                hfree ( (char *) b );
                hfree ( (char *) x );
                hfree ( (char *) a );
        }
        wclose ( mwn );
        wclose ( mwn0 );
        vcend ( CLOSE );
}

/*************************/
/* Set parameters       */
/* call subrouitnes     */
/*************************/


/*************************/
/* LeastSquares Funtion */
/*************************/
ls( x , n , fp )
double  huge *x;                /* array of samples */
double  n;              /* filter length    */
double  fp;             /* band edge in Hz. */
{
int             m,
                j;
double  p,
                q,
```

```
                am,
                n2;

                p  = 4 * atan (1);
                m  = (n+1)/2;
                am = (double)(n+1)/2;
                n2 = n/2;

                if ( m == am )
                        x[m] = (double)2*fp;
                for ( j = 1; j <=n2 ; ++j ) {
                        q = p * ( j - am );
                        x[j] = sin( fp*2*q ) / q;
                }
}


wind( x , n, tp , betas )
double  huge *x; /* array of samples */
double  n ; /* filter length    */
int             tp; /* window type      */
double  betas; /* beta for kaiser */
{
int             j;
double  fio1,
                q,
                wt,
                am,
                ag,
                q1;

                am = ( n + 1 ) /2;
                q  = 4 * atan(1) / am;
                q1 = q / ( am - 1 );

                switch ( tp ) {

                        case 1:         /* Bartlett window */
                                        for ( j = 1 ; j <= am ; ++j ) {
                                                wt = j / am;
                                                x[j] = wt * x[j];
                                        }
                                        break;

                        case 2:         /* Hanning window  */
                                        for ( j = 1 ; j <= am ; ++j ) {
                                                wt = 0.5 - 0.5 * cos(j * q );
                                                x[j] = wt * x[j];
                                        }
                                        break;

                        case 3:         /* Hamming         */
```

```
                                for ( j = 1 ; j <= am ; ++j ) {
                                        wt = 0.54 - 0.46 * cos( (j-1) * q1 );
                                        x[j] = wt * x[j];
                                }
                                break;

                        case 4:         /* Blackmann       */

                                for ( j = 1 ; j <= am ; ++j ) {
                                        wt = 0.42 - 0.5 * cos( j * q ) + .08 * cos( 2*j*q );
                                        x[j] = wt * x[j];
                                }
                                break;

                        case 5:         /* Kaiser          */

                                fio1 = fio( betas );
                                for ( j = 1 ; j <= am ; ++j ) {
                                        ag = betas * sqrt ( 1- pow ( (am-j)/(am-1) , 2 )  );
                                        wt  = fio( ag ) / fio1;
                                        x[j] = wt * x[j];
                                }
                                break;
                        default:
                                break;
                }
}


/*****************************/
/* Bessel for Kaiser Window */
/*****************************/

double  fio( z )
double  z;
{
double  y,
                e,
                d2,
                d;
int             j;



                y = (double)z/2.0;
                e = (double)1.0;
                d = 1;
                for ( j = 1 ; j <= 25 ; ++j ) {

                        d = d * y / j;
                        d2 = d * d;
                        e += d2;
```

```
                if ( d2 < e*1e-7 )
                        return ( double )e;
        }
        printf ( "IO failed to converage \n" );
        return (double)e;
}

/*************************************/
/* Funtion to calculate freq response */
/*************************************/
myfr( x, b, n, k )
double  huge *x,            /** array of filter coefficents **/
        huge *b;            /** array of frequency response **/
double  n,          /** filter length **/
        k;          /** fresponse number **/
{
double  q,
        am,
        at,
        f;
int     i,
        j,
        m,
        n2,
        c1;

        c1 = n /2;
        c1 = n - 2*c1;
        q  = (double) (3.141592654 / k);
    am = ( n +1 ) *.5;
        m  =  ( n +1 ) /2;
        n2 = n/2;
        for ( j = 1 ; j <= k+1 ; ++j ) {
                at = (double)0;
                if ( c1 )
                        at = .5 * x[m];
                for ( i = 1 ; i <= n2 ; ++i )
                        at = at + x[i] * cos (  (double) (q*(am-i)*(j-1) ) );
                b[j] = 2*at;
        }
}
/*****************************************/
/*** ENF OF CALCULATION               ***/
/*****************************************/


/*****************************************/
/* funtion to write results to a file   */
/*****************************************/
yazfir004( x, b, a , m, fp , n , typs , k )
double  huge *x,
        huge *b,
```

```
                huge *a;
int             m;
int             typs;
double  k,
                fp,
                n;
{
FILE    *fd;
int             j;
double  f;

                fd = fopen ( dadi , "w" );              /** DOSYAYI AC  **/
                if ( fd == ( FILE * ) NULL ) {
                        printf ( "Dosya Acma Hatasi !\n");
                        exit(0);
                }
                /***************/
                fprintf ( fd , "#HEADER\n\n" ); /** HEADER YAZ **/
                fprintf ( fd , "LS ERROR DESIGN\n" );
                fprintf ( fd , "WINDOWS FOR FIR FILTERS \n\n\n" );
                fprintf ( fd , "KOSE FREKANSI           : %-6.3lf\n\n", fp );
                fprintf ( fd , "FILTRE UZUNLUGU         : %-5.0lf\n\n"  , n  );
                fprintf ( fd , "FILTRE TIPI             : %-1.0lf\n\n"   , filtname[typs -1]);
                fprintf ( fd , "HESAPLANAN FRE.CEV.SAY. : %-6.0lf\n\n"  , k  );
                fprintf ( fd , "))\n" );                /** END OF HEADER **/
                /***************/

                /***************/
                fprintf ( fd , "#COEFFICENTS\n" );      /** COEFFICENTS **/
                fprintf ( fd , "FOURIER COEFFICENTS - LS+WINDOW : %s UZUNLUK : %-5.0lf\n" , filtname[typs-1]
                fprintf ( fd , "COEFFICENT-NO,\n" );
                fprintf ( fd , "AMPLITUDE,\n" );
                for ( j = 1 ; j <= m ;++j )
                                fprintf ( fd , "%d,%-6.6lf\n" , j,x[j] );
                fprintf ( fd , "))\n" );                        /** END OF COEFFICENTS **/
                /***************/

                /***************/
                fprintf ( fd , "#FREQUENCY RESPONSE\n" );       /** FREQUENCY RESPONSE **/
                fprintf ( fd , "FIR FILTER DESIGN - LS+WINDOW : %s UZUNLUK : %-5.0lf\n"  , filtname[typs-1] , 1
                fprintf ( fd , "FREQUENCY,\n" );
                fprintf ( fd , "AMPLITUDE,\n" );
                for ( j = 1 ; j <= k+1 ; ++j ) {
                        f = (double).5 * (j-1)/k;
                        mabs( &b[j] );
                        fprintf ( fd , "%-6.6lf , %-6.6lf\n" , f , b[j] );
                }
                fprintf ( fd , "))\n" );                        /** END OF FREQUENCY RESPONSE **/
                /***************/
                fprintf ( fd , "#FREQ. RESPONSE(in dB)\n" );    /** FREQUENCY RESPONSE **/
                fprintf ( fd , "FIR FILTER DESIGN - LS+WINDOW : %s UZUNLUK : %-5.0lf\n"  , filtname[typs-1], n
                fprintf ( fd , "FREQUENCY,\n" );
```

```
            fprintf ( fd , "AMPLITUDE (in dB ),\n" );
            for ( j = 1 ; j <= k+1 ; ++j ) {
                        f = (double).5 * (j-1)/k;
                        fprintf ( fd , "%-6.6lf , %-6.6lf\n" , f , 20 * log10( b[j] ) );
            }
            fprintf ( fd , "))\n" );                    /** END OF FREQUENCY RESPONSE **/

            fclose ( fd );
}
/*********************************************/
/** absender function of frequency reponse **/
/*********************************************/
mabs( bval )
double  huge *bval;
{
       if ( *bval < (double)0 )
                    (*bval) = -(*bval);
}
/*********************************************/
/** end of absender function             **/
/*********************************************/
```

FSM.C

Frekans örnekleme Metodu , LS FIR filtre tasarım programıdır.

```
#include<math.h>
#include<utl2dfn.h>
#include<utl2def.h>
#include<vcstdio.h>
#include"vap.h"
huge    *halloc();
int             sts_knt( char * );

extern  double  atof();
#define DZERO (double)0


TEXT    fuzn[6],
                dcsp[3],
                kosf[14],
                frcv[8],
                dadi[14];

TEXT    mes001[7][81] = {
                "FREKANS ORNEKLEME METODU-OPTIMAL LEAST SQUARES ERROR ",
                "Dosya Adi              :",
                "Filtre Uzunlugu        :",
                "Kose Frekansi          :",
                "DC Ornek (1/0)         :",
                "Frekans Cevabi Sayisi :"
                };

filt_fsm_ls()
{
int     mwn;
double  huge *x,                /* array of coeff */
                huge *b,                /* array of fresp */
                huge *a;                /* array of samps */
double  fp,             /* edge freq      */
                n,              /* filter length */
                dc,             /* # of dc samples */
                k;              /* # of fresponse calculated */
COUNT   girdi;
int     i;
int     mwn0;

                empty ( fuzn , 5  );
                empty ( dcsp , 2  );
                empty ( kosf , 5  );
                empty ( frcv , 7  );
                empty ( dadi , 13 );

        mwn0 =  wxxopen( 5,
                                                12,
                                                20,
                                                66,
                                                "",
```

```
                                        ACTIVE+COOKED,
                                        0,
                                        0,
                                        4,
                                        178 );

mwn = wopen   ( 4,
                                        10,
                                        19,
                                        64,
                                        "ALCAKGECIREN FILTRE TANIMLAMA"
                                        );

        wxatsay ( mwn ,  1 , 0 , mes001[0], 112 );
        wxatsay ( mwn ,  3 , 1 , mes001[1], 7 );
        wxatsay ( mwn ,  5 , 1 , mes001[2], 7 );
        wxatsay ( mwn ,  7 , 1 , mes001[3], 7 );
        wxatsay ( mwn ,  9 , 1 , mes001[4], 7 );
        wxatsay ( mwn , 11 , 1 , mes001[5], 7 );

        xatget  ( 3, 25, dadi, "FSLMSXXX.FIR", NULLFUNC, "", "", vc.dflt, vc.dflt );
        xatget  ( 5, 25, fuzn, "9999", sts_knt , "", "", vc.dflt, vc.dflt );
        xatget  ( 7, 25, kosf, ".###", sts_knt, "", "",  vc.dflt, vc.dflt );
        xatget  ( 9, 25, dcsp, "9", NULLFUNC, "", "", vc.dflt, vc.dflt );
        xatget  (11, 25, frcv, "999999" , sts_knt, "", "", vc.dflt, vc.dflt );
        girdi = 0;
        switch ( readgets() ) {
                case    RET:
                                        fp = atof ( kosf );
                                        dc = atof ( dcsp );
                                        k  = atof ( frcv );
                                        n  = atof ( fuzn );
                                        b  = x = a = ( double * ) NULL;
                                        b  = (double huge * ) halloc ( (long)(k+1) , sizeof ( double )
                                        if ( b == ( double * ) NULL )  {
                                                printf ( "b=null " );
                                                exit(0);
                                              }
                                        x  = (double huge * ) halloc ( (long)(n/2+2) , sizeof ( double
                                        if ( x == ( double * ) NULL )  {
                                                printf ( "x=null " );
                                                exit(0);
                                              }
                                        a  = (double huge * ) halloc ( (long)(n/2+2) , sizeof ( double
                                        if ( a == ( double * ) NULL )  {
                                                printf ( "a=null " );
                                                exit(0);
                                              }
                                        fhesap( x , b, a, fp, n, dc, k );
                                        girdi = 1;
                                        break;
                case    ESC:
```

```
                deafult    :
                                        break;
        }
        if ( girdi ) {
                hfree ( (char *) b );
                hfree ( (char *) x );
                hfree ( (char *) a );
        }
        wclose ( mwn );
        wclose ( mwn0 );
}


/*****************************************/
/** Calculation of fourier coefficents **/
/*****************************************/
fhesap( x, b, a, fp, n , dc , k )
double  huge *x,                /* array of coeff */
        huge *b,                /* array of fresp */
        huge *a;                /* array of samps */
double  fp,             /* edge freq      */
        n,              /* filter length */
        dc,             /* # of dc samples */
        k;              /* # of fresponse calculated */
{
double  q,
        am,
        np,
        at,
        xt,
        f;
int     i,
        j,
        m,
        m1,
        n2;

        m = ( n+1 ) /2;
        am = (double)( n + 1.0 ) /2.0;
        m1 = n/2+1;
        q = 6.283185530717959 / n;
        n2 = n /2;
        np = fp * n + 1.0;

        if ( dc == 1 )
                np = n * fp + .5;

        for ( j = 1 ; j <= (int)np ; ++j )
                        a[j]  = (double)1.0;
        for ( j = np+1 ; j <= (int)m1 ; ++j )
                        a[j] = (double)0;
```

```
                if ( dc == 0 ) {
                        for ( j = 1 ; j <=m ; ++j ) {
                                xt = (double)a[1] / 2;
                                for ( i = 2 ; i <= m ; ++i )
                                        xt = xt + a[i] * cos ( (double)q*(double)(am-j)*(double)(i-1))
                                x[j] = (double) 2.0*xt /n;
                        }
                } else {
                        for ( j = 1 ; j <=m ; ++j ) {
                                xt = (double)0;
                                for ( i = 1 ; i <= n2 ; ++i )
                                        xt = xt + a[i] * cos ( q*(am-j)*(i-.5));
                                if ( am != m )
                                        xt = xt + a[m] * cos( (double)3.141592654*(double)(am-j)) /2;
                                x[j] = (double) 2.0*xt /n;
                        }
                }
                myfr( x, b, n, k );
                yazfir001( x, b, a , m, fp , n , dc , k );
}
/*****************************************/
/** END of fourier coefficents calc.   **/
/*****************************************/

/***********************************************/
/** absender function of frequency reponse **/
/***********************************************/
mabs( bval )
double  huge *bval;
{
        if ( *bval < (double)0 )
                        (*bval) = -(*bval);
}
/***********************************************/
/** end of absender function              **/
/***********************************************/


/*****************************************/
/*** CALCULATION OF FREQUENCY RESPONSE ***/
/*****************************************/
myfr( x, b, n, k )
double  huge *x,                /** array of filter coefficents **/
                huge *b;                /** array of frequency response **/
double  n,              /** filter length **/
                k;              /** fresponse number **/
{
double  q,
                am,
                at,
                f;
int             i,
```

```
            j,
            m,
            n2,
            c1;

            c1 = n /2;
            c1 = n - 2*c1;
            q  = (double) (3.141592654 / k);
          am = ( n +1 ) *.5;
            m  =  ( n +1 ) /2;
            n2 = n/2;
            for ( j = 1 ; j <= k+1 ; ++j ) {
                  at = (double)0;
                  if ( c1 )
                        at = .5 * x[m];
                  for ( i = 1 ; i <= n2 ; ++i )
                        at = at + x[i] * cos (  (double) (q*(am-i)*(j-1) ) );
                  b[j] = 2*at;
            }
}
/*****************************************/
/*** ENF OF CALCULATION              ***/
/*****************************************/

yazfir001( x, b, a , m, fp , n , dc , k )
double  huge *x,
            huge *b,
            huge *a;
int         m;
double  k,
            fp,
            dc,
            n;
{
FILE    *fd;
int         j;
double  f;

            fd = fopen ( dadi , "w" );          /** DOSYAYI AC  **/
            if ( fd == ( FILE * ) NULL ) {
                        printf ( "Dosya Acma Hatasi !\n");
                        exit(0);
            }
            /***************/
            fprintf ( fd , "#HEADER\n\n" ); /** HEADER YAZ **/
            fprintf ( fd , "FREKANS ORNEKLEME METODU\n" );
            fprintf ( fd , "OPTIMUM LEAST SQUARES ERROR\n\n\n" );
            fprintf ( fd , "KOSE FREKANSI          : %-6.3lf\n\n", fp );
            fprintf ( fd , "FILTRE UZUNLUGU        : %-5.0lf\n\n"  , n  );
            fprintf ( fd , "DC ORNEK ( 1/0 )       : %-1.0lf\n\n"   , dc );
            fprintf ( fd , "HESAPLANAN FRE.CEV.SAY. : %-6.0lf\n\n"  , k  );
            fprintf ( fd , "))\n" );            /** END OF HEADER **/
```

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

                /***************/
                fprintf ( fd , "#COEFFICENTS\n" );      /** COEFFICENTS **/
                fprintf ( fd , "FOURIER COEFFICENTS\n" );
                fprintf ( fd , "COEFFICENT-NO,\n" );
                fprintf ( fd , "AMPLITUDE,\n" );
                for ( j = 1 ; j <= m ;++j )
                                fprintf ( fd , "%d,%-6.6lf\n" , j,x[j] );
                fprintf ( fd , "))\n" );                        /** END OF COEFFICENTS **/
                /***************/

                /***************/
                fprintf ( fd , "#FREQUENCY RESPONSE\n" );       /** FREQUENCY RESPONSE **/
                fprintf ( fd , "FIR FILTER DESIGN - OPTIMAL LEAST SQUARES ERROR\n" );
                fprintf ( fd , "FREQUENCY,\n" );
                fprintf ( fd , "AMPLITUDE,\n" );
                for ( j = 1 ; j <= k+1 ; ++j ) {
                                f = (double).5 * (j-1)/k;
                                mabs( &b[j] );
                                fprintf ( fd , "%-6.6lf , %-6.6lf\n" , f , b[j] );
                }
                fprintf ( fd , "))\n" );                        /** END OF FREQUENCY RESPONSE **/
                /***************/

                fclose ( fd );
}

int     sts_knt( sts )
char    *sts;
{
        if ( atof ( sts ) == DZERO )
                return 1;
        else
                return 0;
}
```

FSMIW.C

Geçiş bölgeli FIR filtre tasarım programıdır.

```
/******
22.12.1991 Metin KALAYCI

LEAST SQUARES ERROR FIR FILTERS WITH A TRANSITION REGION DESIGN PROGRAM
FOR A LINEAR PHASE LOWPASS FILTER
       FILTER LENGTH = N
              PASSBAND EDGE IN HZ = FP
                     STOP BAND EDGE IN HZ = FS
                     SAMPLING RATE = 1
              TRANSITION TYPE TP :
                     0.RAISED COSINE
                     1.2nd ORDER
                     3.3rd ORDER
                     4.4th ORDER , etc
              FP=FS  NO TRANSITION REGION

******/

#include<math.h>
#include<utl2dfn.h>
#include<utl2def.h>
#include<vcstdio.h>
extern  myfr();         /* Frequency response function external */
huge    *halloc();

extern  double  atof();
#define DZERO (double)0

TEXT    fil2nm[15],     /* Dosya Adi */
              fil2uz[10], /* filtre uzunlugu */
              fil2bs[10], /* filtre gecis bandi baslangici */
              fil2sn[10], /* filtre gecis bandi sonu       */
              fil2tp[2],      /* gecis tipi */
              fil2fc[10];     /* hesaplanacak filtre frekans cevabi sayisi */

TEXT    mes002[7][81] = {
              "      LINEER FAZ FIR FILTRE - SAMPLING RATE = 1      ",
              "Dosya Adi              :",
              "Filtre Uzunlugu        :",
              "Baslangic Kose Frekansi :",
              "Bitis Kose Frekansi    :",
              "Gecis Tipi             :",
              "Frekans Cevabi Sayisi  :"
              };

filt_fpb_ls()
{
int           k;
double  fp,
              fq,
              fp1,
              fq1,
```

```
                tp,
                trt,
                n;
double  huge *x,                /* array of coeff */
                huge *b,                /* array of fresp */
                huge *a;                /* array of samps */
COUNT   girdi;
int     i,
        mwn0,
        mwn;

                empty ( fil2nm,13);     /* Dosya Adi */
                empty ( fil2uz,10); /* filtre uzunlugu */
                empty ( fil2bs, 5); /* filtre gecis bandi baslangici */
                empty ( fil2sn, 5); /* filtre gecis bandi sonu       */
                empty ( fil2tp, 2); /* gecis tipi */
                empty ( fil2fc, 7);     /* hesaplanacak filtre frekans cevabi sayisi */
        mwn0 =  wxxopen( 5,
                                                12,
                                                20,
                                                66,
                                                "",
                                                ACTIVE+COOKED,
                                                0,
                                                0,
                                                4,
                                                178 );

        mwn = wopen   ( 4,
                                                10,
                                                19,
                                                64,
                                                "ALCAKGECIREN FILTRE TANIMLAMA-GECIS BOLGELI"
                                                );

                wxatsay ( mwn ,  1 , 0 , mes002[0], 112 );
                wxatsay ( mwn ,  3 , 1 , mes002[1], 7 );
                wxatsay ( mwn ,  5 , 1 , mes002[2], 7 );
                wxatsay ( mwn ,  7 , 1 , mes002[3], 7 );
                wxatsay ( mwn ,  9 , 1 , mes002[4], 7 );
                wxatsay ( mwn , 11 , 1 , mes002[5], 7 );
                wxatsay ( mwn , 13 , 1 , mes002[6], 7 );

                xatget  ( 3, 25, fil2nm, "FSLPBXXX.FIR", NULLFUNC, "", "", vc.dflt, vc.dflt );
                xatget  ( 5, 25, fil2uz, "999999999", NULLFUNC, "", "", vc.dflt, vc.dflt );
                xatget  ( 7, 25, fil2bs, ".9999", NULLFUNC, "", "", vc.dflt, vc.dflt );
                xatget  ( 9, 25, fil2sn, ".9999", NULLFUNC, "", "", vc.dflt, vc.dflt );
                xatget  (11, 25, fil2tp, "9", NULLFUNC, "", "", vc.dflt, vc.dflt );
                xatget  (13, 25, fil2fc, "99999", NULLFUNC, "", "", vc.dflt, vc.dflt );
                girdi = 0;
                switch ( readgets() ) {
                        case    RET:
```

```
                                        fp = atof ( fil2bs );
                                        fq = atof ( fil2sn );
                                        if ( fp > fq ) {
                                               trt = fp;
                                               fp = fq;
                                               fq = trt;
                                        }
                                        fp1 = fq - fp;
                                        fq1 = fp + fq;
                                        n  = atof ( fil2uz );
                                        tp = atof ( fil2tp );
                                        k  = atoi ( fil2fc );
                                        b  = x = ( double * ) NULL;
                                        b  = (double huge * ) halloc ( (long)(k+1) , sizeof ( double )
                                        if ( b == ( double * ) NULL )  {
                                                     printf ( "b=null " );
                                                     exit(0);
                                               }
                                        x  = (double huge * ) halloc ( (long)(n/2+2) , sizeof ( double
                                        if ( x == ( double * ) NULL )  {
                                                     printf ( "x=null " );
                                                     exit(0);
                                               }
                                        girdi = 1;
                                        ls ( x , n , fq1 , fp1 );
                                        wgt( x , n, tp , fq1 , fp1 );
                                        atsay ( 0, 0 , "wgt geldi" );
                                        myfr( x, b, n, (double)k );
                                        atsay ( 0, 0 , "frc geldi" );
                                        yazfir002( fil2nm , x, b, n, k , tp, fp, fq );
                                        atsay ( 0, 0 , "yazfir geldi" );
                                        break;
                        case    ESC:
                        deafult    :

                                        break;

                }
                if ( girdi ) {
                        hfree ( (char *) b );
                        hfree ( (char *) x );
                }
                wclose ( mwn );
                wclose ( mwn0 );
}

/*******************/
/* LS              */
/*******************/

int             ls ( x , n , fp , fq )
double  huge *x;
double  fp,
                n,
```

```
                fq;
{
int             n2,
                j;
double  p,
                am,
                m,
                q;

                        p  = (double) 3.141592654;
                        m  = (double) ( n + 1.0 ) / 2.0;
                        am = (double) ( n + 1 ) / 2;
                        n2 = (int) n / 2;
                        if ( m == am )
                                        x[(int)m] = fp;
                        for ( j = 1 ; j <= n2 ; ++j ) {
                                q = p * ( j - am );
                                x[j] = sin ( fp * q ) / q;
                        }
}

/*******************/
/* WEIGHTS         */
/*******************/
int             wgt( x , n, tp , fp , fq )
double  huge *x;
double  tp,
                n,
                fp,
                fq;

{
double  wt,
                am,
                q,
                q1,
                p;
int             j;
double  pt;

                am = ( n + 1.0 ) / 2.0;
                p  = (double) 3.141592654;
                q  = p * fq;

                if ( fq == 0 )
                                return 0;
                if ( tp != DZERO )  {           /* Calculation of tp order spline */
                        for ( j =  1 ; j <= am-1 ; ++j ) {
                                q1 = (double)q * (double)( j - am ) / (double)tp;
                                pt = (double)sin( (double)q1 ) /(double) q1;
                                wt =  pow ( pt  , tp );
                                x[j] = wt * x[j];
```

```
                }
        } else {                                /* Raised COSINE */

                for ( j = 1 ; j <= am-1 ; ++j ) {
                        wt = cos ( q * ( j - am ) );
                        if ( wt  < 0.0000001 && wt > -0.0000001 )
                                        wt = wt / ( 1 - ( 2 * fq * pow ( ( j - am ) , 2 ) ) );
                        else
                                        wt = p / 4.0;
                        x[j] = wt * x[j];
                }
        }
}
/*****************************************/
/* FREQUENCY RESPONSE                    */
/* Same function that is given in vap.c */
/*****************************************/
yazfir002( fnams , x, b, m, k , tp, fp, fq )
TEXT    *fnams;
double  huge *x,
        huge *b;
double  m,
        tp,
        fp,
        fq;
int     k;
{
FILE    *fd;
int     j;
double  f;

        fd = fopen ( fnams , "w" );             /** DOSYAYI AC  **/
        if ( fd == ( FILE * ) NULL ) {
                        printf ( "Dosya Acma Hatasi !\n");
                        exit(0);
        }
        /***************/
        fprintf ( fd , "#HEADER\n\n" ); /** HEADER YAZ **/
        fprintf ( fd , "OPTIMUM LEAST SQUARES ERROR\n" );
        fprintf ( fd , "GECIS BANDLI - FIR FILTRE  \n" );
        fprintf ( fd , "0. raised cosine\n" );
        fprintf ( fd , "1. 1ci derece\n" );
        fprintf ( fd , "n. n.ci derece\n\n" );
        fprintf ( fd , "KOSE FREKANSI BASLANGICI: %-6.3lf\n\n", fp );
        fprintf ( fd , "KOSE FREKANSI SONU      : %-6.3lf\n\n", fq );
        fprintf ( fd , "FILTRE UZUNLUGU         : %-5.0lf\n\n", m  );
        fprintf ( fd , "GECIS TIPI              : %-1.0lf\n\n", tp );
        fprintf ( fd , "HESAPLANAN FRE.CEV.SAY. : %-6.0lf\n\n", k  );
        fprintf ( fd , "))\n" );                /** END OF HEADER **/
        /***************/

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

```
        fprintf ( fd , "#COEFFICENTS\n" );      /** COEFFICENTS **/
        fprintf ( fd , "FOURIER COEFFICENTS\n" );
        fprintf ( fd , "COEFFICENT-NO,\n" );
        fprintf ( fd , "AMPLITUDE,\n" );
        for ( j = 1 ; j <= (m+1)/2 ;++j )
                        fprintf ( fd , "%d,%-6.6lf\n" , j,x[j] );
        fprintf ( fd , "))\n" );                        /** END OF COEFFICENTS **/
        /***************/

        /***************/
        fprintf ( fd , "#FREQUENCY RESPONSE\n" );       /** FREQUENCY RESPONSE **/
        fprintf ( fd , "FIR FILTER DESIGN - OPTIMAL LEAST SQUARES ERROR\n" );
        fprintf ( fd , "FREQUENCY,\n" );
        fprintf ( fd , "AMPLITUDE,\n" );
        for ( j = 1 ; j <= k+1 ; ++j ) {
                f = (double).5 * (j-1)/k;
                mabs( &b[j] );
                fprintf ( fd , "%-6.6lf , %-6.6lf\n" , f , b[j] );
        }
        fprintf ( fd , "))\n" );                        /** END OF FREQUENCY RESPONSE **/
        /***************/

        fclose ( fd );
}
```

DRAW.C

Genel amaçlı grafik çizim programıdır. Bir veya birden fazla grafiği ekrana çizebilir. Ekran kartı tipine otomatik olarak uyar. Eksen bölmeleme otomatik olarak hesaplanır. Tasarım sonucunda oluşan dosyalar çizdirlebileceği gibi herhangibir text dosyanın içindeki başka programların nümerik çıktılarıda çizdirilebilir. Tümüyle bağımsız ve esnek bir programdır.

```
int  init_g(unsigned char  *Fnam,unsigned char  *SrcStr,int  PlacX,int  PlacY);
struct  xyval *DeGer(unsigned char  *Fnam,int  *Elem,int  Xdeg_Place,int  Ydeg_Place,unsigned char  *SrcStr);
int  HxBul(char  *Fnam,char  *SrcStr);
int  AxisBul(char  *Fnam,char  *SrcStr);
int  dr_win(void);
int  p1(char  *des);
int  pp(char  *des);
int  p2(char  *des);
```

```
#include <vcstdio.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <malloc.h>
#include <string.h>
#include <pgchart.h>
#include <graph.h>
#include "draw.h"
#define NFONTS 6
#define TEXT    unsigned char
#define COUNT   unsigned short
extern  ConFile( char * , char * );

static  TEXT    grn[4][81],
                                vxa[4][81],
                                vya[4][81];
static  int             xyer[4],
                                yyer[4];
struct ldef {
        TEXT    dstr[80];
        TEXT    astr[80];
        struct  ldef    *next;
        struct  ldef    *prev;
};

extern  TEXT    *pen_haz();
typedef struct ldef PLIN;
static  int             screen_num;

TEXT    mnam[256],
                xnam[256],
                ynam[256];
int             MAX_XPIXEL;
int             MAX_YPIXEL;
float   *xvalue;
float   *yvalue;
double  atof();
chartenv        myenv;
extern  PLIN    *ElemEk ();
int             p1(),
                pp(),
                p2();
PLIN    *AxRoot,
                *MxRoot;
int             AxElem,
                MxElem;

typedef struct  xyval {
        float   *xxx;
        float   *yyy;
} DEGER;
```

```
DEGER   *DeGer();
TEXT    Decfile[15];

init_g ( Fnam , SrcStr , PlacX , PlacY )
TEXT    *SrcStr;
TEXT    *Fnam;
int             PlacX,
                PlacY;
{
int             Elem;
DEGER   X_Y;
struct  videoconfig     vc;
int             PX,
                PY;
int             chrtsay,
                mchrt;
int             basx,
                basy,
                sonx,
                sony;
int             s;

        *mnam = *xnam = *ynam = 0;
        strcpy ( Decfile , Fnam );
        MxBul  ( Fnam , SrcStr );
        dr_win ();
        if ( _setvideomode ( _MAXRESMODE ) == 0 )
                                                        exit(0);
        _getvideoconfig ( &vc );
        _pg_initchart();
        _pg_defaultchart ( &myenv , _PG_SCATTERCHART , _PG_POINTANDLINE );

        myenv.xaxis.grid = 1;
        myenv.yaxis.grid = 1;
        myenv.xaxis.gridstyle = 7;
        myenv.yaxis.gridstyle = 7;

        chrtsay = 0;
        for ( s = 0 ; s <= 3 ; ++s )
                if ( xyer[s] != 0 && yyer[s] != 0 )
                                chrtsay++;
        if ( chrtsay >= 2 )
                        mchrt = 2;
        else
                        mchrt = 1;
        for ( s = 0 ; s <= 3; ++s ) {

        if ( yyer[ s ] != 0 && xyer[ s ] != 0 ) {
                X_Y = *DeGer( Fnam , &Elem , xyer[s]  , yyer[s] , grn[s] );
                xvalue = X_Y.xxx;
                yvalue = X_Y.yyy;
        }
```

```
switch ( s ) {
        case 0:
                        basx = 0;
                        basy = 0;
                        sonx = vc.numxpixels / mchrt ;
                        sony = vc.numypixels / mchrt ;
                        break;
        case 1:
                        basx = vc.numxpixels / mchrt;
                        basy = 0;
                        sonx = vc.numxpixels;
                        sony = vc.numypixels / mchrt ;
                        break;
        case 2:
                        basx = 0;
                        basy = vc.numypixels / mchrt;
                        sonx = vc.numxpixels / mchrt ;
                        sony = vc.numypixels ;
                        break;
        case 3:
                        basx = vc.numxpixels/ mchrt;
                        basy = vc.numypixels/ mchrt;
                        sonx = vc.numxpixels;
                        sony = vc.numypixels;
                        break;
}

myenv.chartwindow.x1  =  basx;
myenv.datawindow.x1   =  basx;
myenv.chartwindow.y1  =  basy;
myenv.datawindow.y1   =  basy;

myenv.datawindow.x2   =  sonx-1;
myenv.chartwindow.x2  =  sonx-1;
myenv.datawindow.y2   =  sony-1;
myenv.chartwindow.y2  =  sony-1;

if ( yyer[ s ] != 0 && xyer[ s ] != 0 ) {
        strcpy ( myenv.maintitle.title , mnam );
        strcpy ( myenv.xaxis.axistitle.title , vxa[s] );
        myenv.xaxis.axistitle.justify = _PG_RIGHT;
        myenv.yaxis.axistitle.justify = _PG_LEFT;
        strcpy ( myenv.yaxis.axistitle.title , vya[s] );
        _pg_chartscatter( &myenv , xvalue , yvalue , Elem );
        free ( ( char * ) X_Y.xxx );
        free ( ( char * ) X_Y.yyy );
        if ( --chrtsay == 0 )
                        break;
 }
}
getone();
```

```
        _setvideomode( _DEFAULTMODE );

}

DEGER   *DeGer( Fnam , Elem , Xdeg_Place , Ydeg_Place , SrcStr )
TEXT    *SrcStr;
TEXT    *Fnam;
int             *Elem;
int     Xdeg_Place,
                Ydeg_Place;
{
float   *yv,
                *xv;
TEXT    line[256];
FILE    *fd;
TEXT    Rbuff[128];
COUNT   Ofset;
static  DEGER   x_y;
int             min,
                max,
                tmp;

        *Elem = 0;

        if ( ( fd = fopen ( Fnam , "r" ) ) == ( FILE * ) NULL ) {
                printf ( "Can not open DFT result file exiting to DOS" );
                exit(0);
        }

        yv = NULL;
        xv = NULL;
        min = Xdeg_Place;
        max = Ydeg_Place;
        if ( Xdeg_Place > Ydeg_Place )  {
                        min = Ydeg_Place;
                        max = Xdeg_Place;
        }
        sagbosluk ( SrcStr );
        solbosluk ( SrcStr );
        while ( readline ( fd , line ) != EOF )
                        if ( !strncmp ( SrcStr , line+1  , strlen ( SrcStr ) ) )
                                        break;

        if ( readline ( fd , mnam ) != -1 )
                if (  readline ( fd , xnam )  != -1 )
                                readline ( fd , ynam );

        while ( ( readline ( fd , line ) )  != EOF  ) {
                if ( *line == ')' && *( line + 1 ) == ')' )
                                break;
                Ofset = 0;
                tmp = min;
```

```
                while ( tmp-- > 0 )
                        Satircoz ( &Ofset , line , Rbuff );
                switch ( Xdeg_Place > Ydeg_Place ) {
                        case 1:
                        yv = (float *) realloc ( yv , (size_t)( *Elem + 1 ) * sizeof ( float ) ) ;
                        *( yv +  *Elem ) = (float) atof ( Rbuff );
                        tmp = max - min ;
                        while ( tmp--  > 0 )
                                Satircoz ( &Ofset , line , Rbuff );
                        xv = (float *) realloc ( xv , (size_t)( *Elem + 1 ) * sizeof ( float ) ) ;
                        *( xv +  *Elem ) = (float) atof ( Rbuff );
                        break;

                        case 0:
                        xv = (float *) realloc ( xv , (size_t)( *Elem + 1 ) * sizeof ( float ) ) ;
                        *( xv +  *Elem ) = (float) atof ( Rbuff );
                        tmp = max - min ;
                        while ( tmp-- > 0)
                                Satircoz ( &Ofset , line , Rbuff );
                        yv = (float *) realloc ( yv , (size_t)( *Elem + 1 ) * sizeof ( float ) ) ;
                        *( yv +  *Elem ) = (float) atof ( Rbuff );
                        break;
                }
                (*Elem)++;
        }
        x_y.xxx = xv;
        x_y.yyy = yv;
        fclose ( fd );
        return ( ( DEGER * ) &x_y );
}
/*************************************************/

MxBul( Fnam , SrcStr )
char    *SrcStr,
                *Fnam;
{
FILE    *fd;
TEXT    line[256];
TEXT    tmpS[256];
static  TEXT    curfile;
COUNT   oft;
int             sira;

                MxRoot = ( PLIN * ) NULL;
                fd = fopen ( Fnam , "r" );
                if ( fd ==  ( FILE * ) NULL )  {
                                printf ( "Dosya Acma Hatasi ... exiting to DOS\n" );
                                exit(0);
                }
                MxElem = 0;
                oft = 0;
                while ( readline ( fd , line ) != EOF )
```

```
dr_win()
{
int             s;
int             mwn0,
                mwn;

                for ( s = 0 ; s <= 3 ; ++s ) {
                        *grn[s] = *vxa[s] = *vya[s] = 0;
                        xyer[ s ] = yyer [ s ] = 0;
                }

        mwn0 =  wxxopen( 3,
                                                16,
                                                22,
                                                70,
                                                "",
                                                ACTIVE+COOKED,
                                                0,
                                                0,
                                                4,
                                                178 );

        mwn = wopen   ( 2,
                                                14,
                                                21,
                                                68,
                                                "X-Y PARAMETRE SECIMI"
                                                );

                wxatsay ( mwn ,  3 , 1 , "Grafik Adi-1 :", 7 );
                wxatsay ( mwn ,  4 , 1 , "X-axis Var-1 :", 7 );
                wxatsay ( mwn ,  5 , 1 , "Y-axis Var-1 :", 7 );


                wxatsay ( mwn ,  6 , 1 , "Grafik Adi-2 :", 7 );
                wxatsay ( mwn ,  7 , 1 , "X-axis Var-2 :", 7 );
                wxatsay ( mwn ,  8 , 1 , "Y-axis Var-2 :", 7 );

                wxatsay ( mwn ,  9 , 1 , "Grafik Adi-3 :", 7 );
                wxatsay ( mwn ,  10, 1 , "X-axis Var-3 :", 7 );
                wxatsay ( mwn ,  11, 1 , "Y-axis Var-3 :", 7 );


                wxatsay ( mwn ,  12, 1 , "Grafik Adi-4 :", 7 );
                wxatsay ( mwn ,  13, 1 , "X-axis Var-4 :", 7 );
                wxatsay ( mwn ,  14, 1 , "Y-axis Var-4 :", 7 );


                /* SCREEN 1 */
```

```
screen_num = 0;
xatget  ( 3, 21, grn[0]  , "XXXXXXXXXXXXXXXXXXX", p1 , "", "", vc.dflt, vc.dflt );
xatget  ( 4, 21, vxa[0]  , "XXXXXXXXXXXXXXXXXXX", pp , "", "", vc.dflt, vc.dflt );
xatget  ( 5, 21, vya[0]  , "XXXXXXXXXXXXXXXXXXX", p2 , "", "", vc.dflt, vc.dflt );
switch ( readgets() ) {
                case RET:
                                break;
                case ESC:
                default :
                                wclose ( mwn0 );
                                wclose ( mwn  );
                                return 0;
}

/* SCREEN 2 */
screen_num = 1;
xatget  ( 6, 21, grn[1]  , "XXXXXXXXXXXXXXXXXXX", p1 , "", "", vc.dflt, vc.dflt );
xatget  ( 7, 21, vxa[1]  , "XXXXXXXXXXXXXXXXXXX", pp , "", "", vc.dflt, vc.dflt );
xatget  ( 8, 21, vya[1]  , "XXXXXXXXXXXXXXXXXXX", p2 , "", "", vc.dflt, vc.dflt );
switch ( readgets() ) {
                case RET:
                                break;
                case ESC:
                default :
                                wclose ( mwn0 );
                                wclose ( mwn  );
                                return 0;
}

/* SCREEN 3 */
screen_num = 2;
xatget  ( 9, 21, grn[2]  , "XXXXXXXXXXXXXXXXXXX", p1 , "", "", vc.dflt, vc.dflt );
xatget  ( 10, 21, vxa[2]  , "XXXXXXXXXXXXXXXXXXX", pp , "", "", vc.dflt, vc.dflt );
xatget  ( 11, 21, vya[2]  , "XXXXXXXXXXXXXXXXXXX", p2 , "", "", vc.dflt, vc.dflt );
switch ( readgets() ) {
                case RET:
                                break;
                case ESC:
                default :
                                wclose ( mwn0 );
                                wclose ( mwn  );
                                return 0;
}

/* SCREEN 4 */
screen_num = 3;
xatget  ( 12, 21, grn[3]  , "XXXXXXXXXXXXXXXXXXX", p1 , "", "", vc.dflt, vc.dflt );
xatget  ( 13, 21, vxa[3]  , "XXXXXXXXXXXXXXXXXXX", pp , "", "", vc.dflt, vc.dflt );
xatget  ( 14, 21, vya[3]  , "XXXXXXXXXXXXXXXXXXX", p2 , "", "", vc.dflt, vc.dflt );

switch ( readgets() ) {
                case RET:
```

```
                                        break;
                        case ESC:
                        default :
                                        break;
                }

                wclose ( mwn0 );
                wclose ( mwn  );
}

int     p1( des )
char    *des;
{
TEXT    *x;

        x=      pen_haz( 40 , 4 , 62 , 19 , 7 , 32 , WDWRAP + BORDER+ACTIVE+BD1+COOKED , "CIZILEBILEN" , M:
        if ( x == ( char * ) NULL )  {
                *des = 0;
                return 0;
        }
        sprintf ( des , "%-20.20s" , x );
        AxisBul ( Decfile , x );
        return 0;
}

int     pp( des )
char    *des;
{
TEXT    *x;
PLIN    *tmpRoot;

        x=      pen_haz( 40 , 4 , 62 , 19 , 7 , 32 , WDWRAP + BORDER+ACTIVE+BD1+COOKED , "AXISTYPE" , AxRo(
        if ( x == ( char * ) NULL )  {
                *des = 0;
                return 0;
        }
        tmpRoot = AxRoot;
        while ( strcmp ( tmpRoot->dstr , x ) && tmpRoot->next != ( PLIN * ) NULL )
                        tmpRoot = tmpRoot->next;
        xyer[ screen_num ] = atoi ( tmpRoot->astr );
        sprintf ( des , "%-20.20s" , x );
        return 0;
}

int     p2( des )
char    *des;
{
TEXT    *x;
PLIN    *tmpRoot;

        x=      pen_haz( 40 , 4 , 62 , 19 , 7 , 32 , WDWRAP + BORDER+ACTIVE+BD1+COOKED , "AXISTYPE" , AxRoo
        if ( x == ( char * ) NULL )  {
```

```
            *des = 0;
            return 0;
        }
        tmpRoot = AxRoot;
        while ( strcmp ( tmpRoot->dstr , x ) && tmpRoot->next != ( PLIN * ) NULL )
                    tmpRoot = tmpRoot->next;
        yyer[ screen_num ] = atoi ( tmpRoot->astr );
        sprintf ( des , "%-20.20s" , x );
        return 0;
}
```

MENU.C

Genel amaçlı menu programı. Rekürsif olarak menu.scr text dosyasını okuyarak programların birbirleriyle olan ilişkilerini düzenler.

```
#include <utl2dfn.h>
#include <utl2def.h>
#include <vcstdio.h>
#define LINES   25
#define COLS    80
/*****/
extern TEXT *pen();
TEXT    CurFile[15];
/*****/
typedef struct  menscr MNSCR;
struct  menscr {
        COUNT   win_y0;         /* window lefty  */
        COUNT   win_x0;         /* window leftx  */
        COUNT   win_y1;         /* window righty */
        COUNT   win_x1;         /* window rightx */
        TEXT    win_ac[80];     /* window Acikla */
        int             win_ct;         /* window Ctrl   */
        COUNT   win_rw;         /* window row    */
        COUNT   win_cl;         /* window column */
        COUNT   win_at;         /* window attrib */
        COUNT   win_fc;         /* window filchar*/
};

typedef struct  menus MENU;
struct  menus {
        short   menu_x;         /* x ekseni kordinati                      */
        short   menu_y;         /* y ekseni kordinati                      */
        TEXT    menuop[25]; /* Secenek stringi                         */
        TEXT    menuCp[2];      /* Buyuk Harf                              */
        short   menbof;         /* Buyuk Harf  Kacinci Karakteri           */
        TEXT    menuac[80]; /* Secenek Aciklama                        */
        int             (*fnk)();   /* Secenegin Cagirdigi                             */
        TEXT    melm;           /* Serial Kacinci eleman ->root = 1 tail = 0 */
        MNSCR   ekr;            /* Window yapisi                           */
        MENU    *sag;           /* Sonraki sag secenek                     */
        MENU    *sol;           /* Sonraki sol secenek                     */
        MENU    *asg;           /* Assagi  secenek                         */
        MENU    *yuk;           /* Yukari  ecenek                          */
};

/* ----------FUNCTIONS -----------*/
MENU    *readmenu( TEXT * );
MENU    *menubir ( TEXT * );
MENU    *TusMenu ( COUNT * , MENU * );
int             Satircoz ( COUNT * , TEXT * , TEXT * );
COUNT   pos      ( TEXT  * , TEXT   , COUNT  );
/*  ---------- GLOBALS ----------*/
COUNT   back;
COUNT   front;
int             Stack[50];
int             Sp=0;
/*  --------END OF GLOBALS ------*/
```

```
typedef struct {
        int             (*fonk)();
} FSC;
int ffile(),
        dftfl(),
        idftf(),
        sampl(),
        c_dft(),
        c_idft(),
        drw_dft(),
        filt_fsmls(),
        filt_fsmpb(),
        filt_parksmac(),
        drw_idft(),
        xy_param();
int     nulfunc();
FSC     M_Fonk[50] = {
        NULL,                   /* 0  reserved for exit */
        nulfunc,                /* 1  reserved for nop  */
    ffile,                      /* >=2  user defined functions */
        dftfl,                  /* 3  */
        idftf,                  /* 4  */
        sampl,                  /* 5  */
        drw_dft,                /* 6  */
        xy_param,               /* 7  */
        c_dft,                  /* 8  */
        c_idft,                 /* 9  */
        filt_fsmls,             /* 10 */
        filt_fsmpb,             /* 11 */
        filt_parksmac,  /* 12 */
        nulfunc,
        nulfunc,
        nulfunc,
        nulfunc,
        nulfunc,
        nulfunc,
        nulfunc,
        nulfunc,
        nulfunc,
        nulfunc,
        nulfunc,
        nulfunc,
        nulfunc,
        nulfunc,
        nulfunc,
        nulfunc,
        nulfunc,
        nulfunc
};
int     Back_Win;

main()
{
FILE    *fd;
```

```
MENU    *kok;
TEXT    Strip[80];
int             Win,i;
        front = 7;
        back  = 112;
        *CurFile = 0;
        strcpy ( Strip , "ANAMENU" );
        kok = readmenu( Strip );
        i = kok->ekr.win_at;
    myvcstart( CLRSCRN );
        Back_Win = Bak_Scr_Ac();
        kok->ekr.win_at = i;
        Ac( kok );
        menuyaz ( kok , front , kok->melm );
        menuopr ( kok );
        Kapa();
        Bak_Scr_Kp( Back_Win );
    vcend(CLOSE);
}

MENU    *readmenu( String )
TEXT    *String;
{
MENU    *tmp,
                *root,
                *otmp;
TEXT    line[256];
COUNT   secsay;
COUNT   ofs;
COUNT   icerde;
FILE    *fd;

                secsay = 0;
                icerde = 1;
                root   = otmp = tmp = ( MENU * ) NULL;
        if ( ( fd = fopen ( "menu.scr" , "r" ) ) == ( FILE * ) NULL )
                        return (MENU *) NULL;

                while ( readline ( fd , line ) != -1 ) {

                        if ( ! ( *line == '#' && !strcmp ( line + 1 , String ) ) && icerde )
                                                continue;
                        if ( icerde == 1 ) {
                                icerde = 0;
                                continue;
                        }
                        if ( *line == ')' && * ( line + 1 ) == ')' )
                                                break;
                        otmp = tmp;
                        tmp  = menubir ( line );
                        tmp->sol = otmp;
                        if ( otmp != ( MENU * ) NULL )
```

```
                        otmp->sag = tmp;
                secsay++;
                tmp->melm = secsay;
                if ( secsay == 1 )
                        root = tmp;

                if ( *line == '@' ) {
                        ofs = pos ( line , ',' , 0 );
                        line[ ofs ] = '\0';
                        strcpy ( String, line + 1 );
                        tmp->asg = readmenu (  String );
                        if ( tmp->asg != ( MENU * ) NULL ) {
                                MENU *tmpR;
                                tmpR = tmp->asg;
                                do {
                                        tmpR->yuk = tmp;
                                        tmpR = tmpR->sag;
                                } while ( tmpR->melm != 1 );
                        }
                }
        }
        fclose ( fd );
        if ( tmp != ( MENU * ) NULL )  {
                tmp->sag  = root;
                tmp->melm = 0;
        }
        if ( root != ( MENU * ) NULL )
                root->sol = tmp;
        return ( ( MENU * ) root );
}

MENU    *menubir( Satir )
TEXT    *Satir;
{
MENU    *nod;
COUNT   yrl;
TEXT    tmpStr[256];

        yrl = 0;
        nod = ( MENU * ) malloc ( sizeof ( MENU ) );

        Satircoz ( &yrl , Satir , tmpStr );             /* Alt menu gostergesi */

        Satircoz ( &yrl , Satir , tmpStr );
        nod->fnk = M_Fonk[ atoi ( tmpStr ) ].fonk;      /* Cagrilacak Fonksiyon */

        Satircoz ( &yrl , Satir , tmpStr );
        nod->menu_x = atoi ( tmpStr );                  /* x coordinati */

        Satircoz ( &yrl , Satir , tmpStr );
        nod->menu_y = atoi ( tmpStr );                  /* y coordinati */
```

```
                Satircoz ( &yr1 , Satir , tmpStr );
                sprintf  ( nod->menuop ,"%s", tmpStr ); /* Secenek     */
                nod->menbof = 0;
                sprintf  ( nod->menuCp ,"%c",  BuyukHarf ( &nod->menbof ,tmpStr ) );

                Satircoz ( &yr1 , Satir , tmpStr );
                strcpy   ( nod->menuac  , tmpStr ) ;    /* Secenek Aciklamasi */

                Satircoz ( &yr1 , Satir , tmpStr );             /* RESERVED COMMA */

                Satircoz ( &yr1 , Satir , tmpStr );
                nod->ekr.win_y0 = atoi ( tmpStr );              /* window lefty  */

                Satircoz ( &yr1 , Satir , tmpStr );
                nod->ekr.win_x0 = atoi ( tmpStr );              /* window leftx  */

                Satircoz ( &yr1 , Satir , tmpStr );
                nod->ekr.win_y1 = atoi ( tmpStr );              /* window righty */

                Satircoz ( &yr1 , Satir , tmpStr );
                nod->ekr.win_x1 = atoi ( tmpStr );              /* window rightx */

                Satircoz ( &yr1 , Satir , tmpStr );
                strcpy ( nod->ekr.win_ac, tmpStr );         /* window Acikla */

                Satircoz ( &yr1 , Satir , tmpStr );
                nod->ekr.win_ct = atoi ( tmpStr );              /* window ctrl   */

                Satircoz ( &yr1 , Satir , tmpStr );
                nod->ekr.win_rw = atoi ( tmpStr );              /* window row    */

                Satircoz ( &yr1 , Satir , tmpStr );
                nod->ekr.win_cl = atoi ( tmpStr );              /* window column */

                Satircoz ( &yr1 , Satir , tmpStr );
                nod->ekr.win_at = atoi ( tmpStr );              /* window attributes */

                Satircoz ( &yr1 , Satir , tmpStr );
                nod->ekr.win_fc = atoi ( tmpStr );      /* window fillchar */

                nod->sag = nod->sol = nod->asg = nod->yuk = ( MENU * ) NULL;
                return   ( ( MENU * ) nod );
}

int             Satircoz( off1 , MStr , Veri )
COUNT   *off1;
TEXT    *MStr;
TEXT    *Veri;
{
TEXT    tmpStr[256];
COUNT   off2;
                off2 = pos ( MStr , ',' , *off1 );
```

```
                if ( off2 != -1 )
                        sprintf ( Veri , "%-*.*s" , off2-*off1 , off2-*off1, MStr + *off1 );
                else
                        strcpy  ( Veri , MStr + *off1 );
                *off1 = ++off2;
                if ( off2 == 0 )
                                return -1;
                else
                                return  0;
}

int             BuyukHarf( offset , String )
TEXT    *String;
COUNT   *offset;
{
        while ( *String )  {
                if ( *String >= 65 && *String <= 91 )
                                return ( *String );
                String++;
                (*offset )++;
        }
        *offset = -1;
        return  0;
}

menuyaz  ( toy , attr , elm )
MENU    *toy;
COUNT   attr;
COUNT   elm;
{
        wxatsay ( Stack [Sp-1] ,toy->menu_y , toy->menu_x , toy->menuop , attr );
        if ( toy->menbof != -1 )
        wxatsay ( Stack [Sp-1] ,toy->menu_y , toy->menu_x + toy->menbof , toy->menuCp , 15 );
        if ( elm != 0 )
                        menuyaz ( toy->sag , back , toy->sag->melm );
}

menuopr ( toy )
MENU    *toy;
{
COUNT   tus,
                hep,
                Ass;
                Ass = 0;
                for ( ;; ) {
                        if ( Ass == 1 && toy->asg != ( MENU *) NULL && Sp == 1 ) {
                                        toy = toy->asg;
                                        Ac ( toy );
                                        menuyaz  ( toy , front , toy->melm );
                        }
                        Y: tus = getone();
```

```
                                        menuyaz  ( toy->yuk , back, 0
                                        menuyaz  ( toy->yuk->sag , fr
                                        toy = toy->yuk->sag;
                                } else
                                        toy = toy->yuk;
                        }
                break;
        }
case    SPACE:
case    CUR_DOWN:
                toy = toy->sag;
        break;
case    RET:
        if ( toy->fnk != NULL ) {
                if ( toy->fnk == nulfunc )
                        return 0;
                else
                        toy->fnk();
        } else
        if ( toy->asg != ( MENU *) NULL ) {
                menuyaz  ( toy , front , 0 );
                toy = toy->asg;
                Ass = 1;
                Ac( toy );
                hep = 1;
        } else
                toy = toy->sag;
        break;
case    -100  :
        if ( toy->fnk != NULL ) {
                if ( toy->fnk == nulfunc )
                        return 0;
                else
                        toy->fnk();
        } else
        if ( toy->asg != ( MENU *) NULL ) {
                menuyaz  ( toy , front , 0 );
                toy = toy->asg;
                Ass = 1;
                Ac( toy );
                hep = 1;
        }
        break;
case    CUR_UP:
case    ESC:
        if ( toy->yuk != ( MENU *) NULL && ( toy->melm == 1 || tus ==
                toy = toy->yuk;
                Kapa();
                menuyaz  ( toy , front , 0 );
                hep = 1;
                if ( Sp == 1 )
                        Ass = 0;
```

```
hep = 0;
menuyaz  ( toy , back , hep );
toy = TusMenu( &tus , toy );
switch( tus ) {
        case    ACF9:
                        system ( "cls" );
                        system ( "command.com" );
                        Call_Back_Windows();
                        break;
        case    CUR_LEFT:
                        if ( toy->yuk != ( MENU * ) NULL )  {
                                Kapa();
                                if ( toy->yuk->sol->asg != ( MENU * ) NULL )  {
                                                hep = 1;
                                                if ( Sp == 1 ) {
                                                        menuyaz  ( toy->yuk, back, 0
                                                        menuyaz  ( toy->yuk->sol , fr
                                                        toy = toy->yuk->sol->asg;
                                                        Ac( toy );
                                                } else {
                                                        toy = toy->yuk;
                                                        menuyaz  ( toy , front , 0 );
                                                        goto Y;
                                                }
                                        } else  {
                                                if ( Sp == 1 ) {
                                                        menuyaz  ( toy->yuk , back, 0
                                                        menuyaz  ( toy->yuk->sol , fr
                                                        toy = toy->yuk->sol;
                                                } else
                                                        toy = toy->yuk;
                                        }
                        } else
                                toy = toy->sol;
                        break;
        case    CUR_RIGHT:
                        if ( toy->yuk != ( MENU * ) NULL )  {
                                Kapa();
                                if ( toy->yuk->sag->asg != ( MENU * ) NULL ) {
                                                hep = 1;
                                                if ( Sp == 1 ) {
                                                        menuyaz  ( toy->yuk , back, (
                                                        menuyaz  ( toy->yuk->sag , fr
                                                        toy = toy->yuk->sag->asg;
                                                        Ac( toy );
                                                } else {
                                                        toy = toy->yuk;
                                                        menuyaz  ( toy , front , 0 );
                                                        goto Y;
                                                }
                                        } else {
                                                if ( Sp == 1 ) {
```

```
                                                goto Y;
                                        } else
                                                toy = toy->sol;

                                        break;
                        }
                        menuyaz  ( toy , front , hep );
                }
}

Ac( toy )
MENU    *toy;
{
int             mywin;
        mywin = wxxopen(toy->ekr.win_y0,
                                        toy->ekr.win_x0,
                                        toy->ekr.win_y1,
                                        toy->ekr.win_x1,
                                        toy->ekr.win_ac,
                                        toy->ekr.win_ct,
                                        toy->ekr.win_rw,
                                        toy->ekr.win_cl,
                                        toy->ekr.win_at,
                                        toy->ekr.win_fc);

                winpush( &Sp , mywin );
                if ( Sp > 1  ) {
                        front = 112;
                        back  = 7;
                } else {
                        front = 7;
                        back  = 112;
                }
}

Kapa()
{
                wclose ( winpop( &Sp ) );
                if ( Sp == 1 ) {
                        front = 7;
                        back  = 112;
                } else {
                        front = 112;
                        back  = 7;
                }
}

winpush( sp, var )
int     *sp,
        var;
{
        Stack [ ( *sp )++ ] =  var;
```

```
}

int     winpop( sp )
int     *sp;
{
        if ( *sp > 0 )
                return ( Stack [ --( *sp ) ] );
}

MENU    *TusMenu ( kod, Struk )
MENU    *Struk;
COUNT   *kod;
{
COUNT   Cpos;
MENU    *curpos;
                Cpos   = Struk->melm;
                curpos = Struk;
                if ( *kod >= 97 && *kod <= 122 )
                                *kod -= 32;
                do {
                        if ( *curpos->menuCp == *kod ) {
                                *kod = -100;
                                return ( MENU *) curpos;
                        }
                        curpos = curpos->sag;
                } while ( curpos->melm  != Cpos );
                return ( MENU *) Struk;
}

Bak_Scr_Ac()
{
int     mywin;
        mywin = wxopen(0,
                                                0,
                                                LINES-1,
                                                COLS-1,
                                                "",
                                                132,
                                                LINES,
                                                COLS,
                                                3,
                                                177);
                        return mywin;

}

Bak_Scr_Kp( bwin )
int     bwin;
{
        wclose ( bwin );
}
int nulfunc()
```

```
{
}

myvcstart( no )
COUNT   no;
{
        vcstart ( no );
                wtable[1].bd_t   = 112;
                wtable[1].bg_t   = 112;
                wtable[1].say_t  = 112;
                wtable[1].nget_t =   7;
                wtable[1].get_t  = 112;
                wtable[1].tit_t  = 112;

                wtable[2].bd_t   = 112;
                wtable[2].bg_t   = 112;
                wtable[2].say_t  = 112;
                wtable[2].nget_t =   7;
                wtable[2].get_t  = 112;
                wtable[2].tit_t  = 112;

                wtable[3].bd_t   = 15;
                wtable[3].bg_t   = 15;
                wtable[3].say_t  = 112;
                wtable[3].nget_t =   7;
                wtable[3].get_t  = 112;
                wtable[3].tit_t  = 112;

                wtable[4].bd_t   =   7;
                wtable[4].bg_t   =   7;
                wtable[4].say_t  = 112;
                wtable[4].nget_t =   7;
                wtable[4].get_t  = 112;
                wtable[4].tit_t  = 112;
}
/**********/
int ffile()
{
TEXT    *p;

        p = pen ( "**", "FIR" );
        if ( p != NULL ) {
                strcpy ( CurFile , p );
        }
}

int     dftfl()
{
TEXT    *p;

        p = pen ( "R_FFT**", "**" );
        if ( p != NULL )
```

```
                strcpy ( CurFile , p );
}

int     idftf()
{
TEXT    *p;

        p = pen ( "*", "IDF" );
        if ( p != NULL )
                strcpy ( CurFile , p );
}


int     sampl()
{
TEXT    *p;

        p = pen ( "*", "SAM" );
        if ( p != NULL )
                strcpy ( CurFile , p );
}

Call_Back_Windows()
{
int     t;
                whide ( Back_Win );
                wshow ( Back_Win );
                for ( t = 0 ; t < Sp ;t++)  {
                        whide( Stack[t] );
                        wshow( Stack[t] );
                        }
}
/*************************************************/
/************* GRAPHIC UTILITIES *****************/
/*************************************************/
int     drw_dft()
{

        if ( *CurFile != 0 ) {
                init_g( CurFile , "" , 0 , 0  );
                Call_Back_Windows();
                }
}
/*** End of  Drawings ***/


/*********************************/
/*** Filter Design Programmes ***/
/*** 08.12.1991 Metin KALAYCI ***/
/*********************************/

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

```
/**** Program NO : 1        *******/
/**** Frequency Sampling Method   */
/**** Optimum Least Squares Error */
/**** Metin KALAYCI 08.12.1991    */
/***********************************/
int     filt_fsmls()
{
        filt_fsm_ls();
}

int     filt_fsmpb()
{
        filt_fpb_ls();
}

int     filt_parksmac()
{
        parks();
}

int     xy_param()
{
}

int     c_dft()
{
        dft_win();
}

int     c_idft()
{

}
```

PEN.C

DIROK.C ile okunan directory listesini alfabetik olarak sıraya dizerek dosya seçimine uygun halde bir menu yapsına sokar. Dosya seçme işlemleri bu programın fonksiyonlarıya gerçekleşir.

```
unsigned char  *pen(unsigned char  *pfil,unsigned char  *pext);
struct  ldef *pget_new(void);
int  mpen_link(struct  ldef *pkoku,struct  ldef *pnext);
int  pen_dat_trns(struct  ldef *pkok,unsigned char  *pdata,unsigned char  *adata);
char  *pen_haz(int  wnx,int  wny,int  wnx1,int  wny1,int  wattr,int  wfc,int  wctrl,unsigned char  *whdr,s
char  *pen_sec(int  p_win,struct  ldef *p_data,int  tlin,int  tcol,int  pelem,int  sirali,int  acikla);
int  p_yaz(int  p_win,struct  ldef *p_data,int  sec);
int  imm_yaz(int  p_win,unsigned char  *dtsi,int  line,int  tcol,int  attr,int  flag);
int  invert(int  color);
int  BHarf(unsigned char  *String);
int  dggwin(void);
int  descrwin(void);
int  descr(unsigned char  *fnam,int  fwin,int  curwin);
int  ConFile(unsigned char  *var1,unsigned char  *var2);
```

```
#include <utl2dfn.h>
#include <utl2def.h>
#include <vcstdio.h>
#include "penpro.h"
#define LINES   25
#define COLS    80

struct ldef {
        TEXT    dstr[80];
        TEXT    astr[80];
        struct  ldef    *next;
        struct  ldef    *prev;
};

typedef struct ldef PLIN;
TEXT    *pen();

PLIN    *pget_new();
PLIN    *DirRoot;
int             Direlem;

TEXT    *pen( pfil , pext )
TEXT    *pfil;
TEXT    *pext;
{
PLIN    *proot, *ptmp0, *ptmp1;
TEXT    tdata[15];
int             i,k;
TEXT    *x;

        Direlem = 0;
        directory ( pfil , pext );

        x=      pen_haz( 10 , 4 , 25 , 19 , 7 , 32 , WDWRAP + BORDER+ACTIVE+BD1+COOKED , "DOSYALAR" , DirRo
        if ( x == ( char * ) NULL )
                return ( char * ) NULL;
        temizle ( DirRoot ) ;
        ConFile ( x , tdata );
        return ( TEXT * ) tdata;
}

temizle ( kokus )
PLIN    *kokus;
{
PLIN    *mps;

                while ( kokus != ( PLIN * ) NULL ) {
                        mps = kokus->next;
                        free ( (char  *) kokus );
                        kokus = mps;
                }
}
```

```
PLIN    *pget_new()
{
static  PLIN    *nkok;

                nkok = ( PLIN * ) malloc ( sizeof ( PLIN ) );
                if ( nkok == ( PLIN * ) NULL )  {
                        printf ( "Memory Allocation Error !...<PLIN>\n" );
                        exit(0);
                }
                nkok->next = nkok->prev = ( PLIN *) NULL;
                return  (PLIN *) nkok;
}


mpen_link( pkoku , pnext )
PLIN    *pkoku,
                *pnext;
{
                pkoku->next = pnext;
                pnext->prev = pkoku;
}

pen_dat_trns( pkok , pdata , adata )
PLIN    *pkok;
TEXT    *pdata;
TEXT    *adata;
{
                if ( strlen ( pdata ) < 80 )
                        sprintf ( pkok->dstr , "%s" , pdata );
                else
                        sprintf ( pkok->dstr , "%-79.79s" , pdata );

                if ( strlen ( adata ) < 80 )
                        sprintf ( pkok->astr , "%s" , adata );
                else
                        sprintf ( pkok->astr , "%-79.79s" , adata );
}

char    *pen_haz( wnx , wny , wnxl , wnyl , wattr , wfc , wctrl , whdr , pkok , pelem , sirali , acikla )
int             wnx,
                wny,
                wnxl,
                wnyl,
                wattr,
                wfc,
                wctrl;
TEXT    *whdr;
PLIN    *pkok;
COUNT   pelem;
COUNT   sirali;
COUNT   acikla;
```

```
{
int     mywin;
int     mywi;
char *css;

        mywi =  wxxopen(wny  + 1,
                                        wnx  + 2,
                                        wny1 + 1,
                                        wnx1 + 2,
                                        "",
                                        ACTIVE+COOKED,
                                        0,
                                        0,
                                        4,
                                        178);

        mywin = wxxopen(wny,
                                        wnx,
                                        wny1,
                                        wnx1,
                                        whdr,
                                        wctrl,
                                        0,
                                        0,
                                        wattr,
                                        wfc);
                css = pen_sec ( mywin , pkok , wny1-wny-1 , wnx1 -wnx- 1  , pelem , sirali , acikla );
                wclose  ( mywin );
                wclose  ( mywi  );
                return (char  *) css;

}

char    *pen_sec( p_win , p_data , tlin , tcol , pelem , sirali , acikla )
PLIN    *p_data;
int             p_win;
int             tlin;
int             tcol;
COUNT   pelem;
COUNT   sirali,
                acikla;
{
COUNT   secenek;
COUNT   attr1;
COUNT   attr;
COUNT   tus;
COUNT   i;
int             flag,
                dgg_win,
                des_win;
PLIN    *ptmp_data;
WPTR    wptr;
```

```
COUNT   curelem,
                ok,
                old,
                new,
                oldcur;

                curelem = oldcur = 1;
                wptr  = &winfo[ p_win ];
                attr1  = wptr->say_at;
                attr   = invert ( wptr->say_at );

                secenek = 0;
                ok = 0;
                ptmp_data = p_data;
                if ( p_data != ( PLIN * ) NULL )
                        p_yaz ( p_win ,  p_data , 0 );
                else
                        pelem = 1;

                do {

                if ( acikla && p_data != ( PLIN * ) NULL && ok )  {
                        werabox ( des_win , 0 , 0 ,  17 , 40 , 7 );
                        descr ( p_data->dstr , des_win , p_win );
                }
                old = oldcur * tlin / pelem;
                new = curelem* tlin / pelem;
                old = ( old == 0) ? 1 : old;
                new = ( new == 0) ? 1 : new;

                if ( p_data != ( PLIN  * ) NULL )
                        on_border ( p_win , old , new );
                oldcur = curelem;

                tus = getone ();
                if ( p_data != ( PLIN * ) NULL )
                switch ( tus ) {
                        case  ACF1:
                                        break;
                                        if ( acikla && !ok ) {
                                                dgg_win = dggwin();
                                                des_win = descrwin();
                                        } else  {
                                                wclose ( des_win );
                                                wclose ( dgg_win );
                                        }

                                        ok = !ok;
                                        break;
                        case  HOME_KEY:
                                                if ( p_data != ptmp_data ) {
                                                        secenek = 0;
```

```
                                curelem = 1;
                                p_data = ptmp_data;
                                p_yaz ( p_win ,  p_data , secenek );
                        }
                        break;
        case  END_KEY:
                        if ( p_data != ptmp_data->prev ) {
                                secenek = 0;
                                curelem = pelem;
                                p_data = ptmp_data->prev;
                                p_yaz ( p_win ,  p_data , secenek );
                        }
                        break;
        case  PGUP:
                        secenek = 0;
                        for ( i = 0 ; i < tlin ; ++i )
                                if ( p_data->prev != ptmp_data->prev )  {
                                                p_data = p_data->prev;
                                                curelem--;
                                }
                        p_yaz ( p_win ,  p_data , secenek );
                        break;
        case  PGDN:
                        secenek = 0;
                        for ( i = 0 ; i < tlin ; ++i )
                                if ( p_data->next != ( PLIN * ) NULL )  {
                                                p_data = p_data->next;
                                                curelem++;
                                }
                        p_yaz ( p_win ,  p_data , secenek );
                        break;
        case  '-':
        case  BTAB:
        case  CUR_UP:
        case  CUR_LEFT:
                        flag = -1;
                        if ( p_data->prev != ptmp_data->prev )
                                p_data = p_data->prev;
                        else
                                break;
                --secenek;
                if ( secenek == -1 ) {
                                secenek = 0;
                                myscroll_updown( p_win , 1, 0 , -1 );
                                flag = 0;
                }
                curelem--;
                imm_yaz ( p_win ,  p_data->dstr , secenek , tcol , attr1 , flag );
                break;
        case  '+':
        case  TAB:
        case  CUR_DOWN:
```

```
case  CUR_RIGHT:
                        flag = 1;
                        if ( p_data->next != ( PLIN *)NULL )
                                p_data = p_data->next;
                        else
                                break;
                if ( secenek == tlin -1 ) {
                        myscroll_updown( p_win , 1, 0 , 1 );
                        flag = 0;
                }
                else
                        ++secenek;
                curelem++;
                imm_yaz ( p_win , p_data->dstr , secenek , tcol , attri , flag );
                break;

case    RET:
case    ESC:
                if ( acikla && ok ) {
                        wselect ( des_win );
                        wclose  ( des_win );
                        wselect ( dgg_win );
                        wclose  ( dgg_win );
                        wselect ( p_win   );
                }
                if ( tus == RET )
                        return ( char * ) p_data->dstr;
                else
                        return ( char * ) NULL;

default:
                if ( sirali && ( tus >= 65 || tus <= 90 ) && ( tus >= 48 || tus <=

                        if (  BHarf ( p_data->dstr ) <= tus ) {

                                do {
                                        if ( p_data->next != ( PLIN * ) NULL )  {
                                                if ( BHarf( p_data->next->dstr ) <=
                                                        p_data = p_data->ne
                                                        curelem++;
                                                } else break;
                                        } else break;

                                }       while ( BHarf( p_data->dstr ) < tus );

                        } else  {

                                do {
                                        if ( p_data->prev != ( PLIN * ) ptmp_data->
                                                if ( BHarf( p_data->prev->dstr ) >=
                                                        p_data = p_data->prev;
                                                        curelem--;
```

```
                                                        } else break;
                                                } else break;

                                        } while ( BHarf( p_data->dstr ) >= tus );
                                }

                                secenek = 0;
                                p_yaz ( p_win ,  p_data , secenek );
                        }
                        break;
                }
                } while ( tus != RET && tus != ESC );
}

p_yaz ( p_win ,  p_data , sec )

PLIN    *p_data;
int             p_win;
int             sec;
{
COUNT   cattr,
                i,
                k,
                tcol,
                tlin,
                attr,
            bd,
                iatt;
WPTR    wptr;
TEXT    SpcTxt[80];
COUNT   lo_r,
                lo_c,
                up_r,
                up_c;

        wptr = &winfo[ p_win ];
        iatt = wptr->say_at;
        attr = invert ( wptr->say_at );
    bd=(wptr->control & BORDER ? 1 : 0);
        up_r=wptr->upper_r+bd;
    up_c=wptr->upper_c+bd;
    lo_r=wptr->lower_r-bd;
    lo_c=wptr->lower_c-bd;
    tlin = lo_r-up_r+1;
    tcol = lo_c-up_c+1;

        woff();
        for ( i= 0 ; i < tlin ; ++i )  {
                if ( i == sec )
                                cattr = attr;
                else
                                cattr = iatt;
```

```
                sprintf ( SpcTxt , "%-*.*s" , tcol , tcol , p_data->dstr );
                wxatsay ( p_win , i , 0 , SpcTxt , cattr );
                if ( p_data->next != (PLIN *) NULL )
                        p_data = p_data->next;
                else
                        break;
        }
        for ( k= i + 1; k < tlin ; ++k )  {
                sprintf ( SpcTxt , "%-*.*s" , tcol , tcol , " " );
                wxatsay ( p_win , k , 0 , SpcTxt , iatt );
        }
        won();
}

imm_yaz ( p_win  , dtsi , line , tcol , attr  , flag )
int             p_win,
                attr,
                flag,
                tcol,
                line;
TEXT    *dtsi;
{
TEXT    SpcTxt[255];

        woff();
        sprintf ( SpcTxt , "%-*.*s" , tcol , tcol , dtsi );
        if ( flag != 0 )  {
                at ( line , 0 );
                wputattr( p_win , invert ( attr  ) , tcol );
                at ( line - flag, 0 );
                wputattr( p_win , attr , tcol );
        } else
                wxatsay ( p_win , line , 0 , SpcTxt , invert ( attr ) );
        won();
}

COUNT   invert ( color )
COUNT   color;
{
COUNT   fg,
                bg,
                nc;

        fg = ( color & 7  );
        bg = ( color >> 4 ) & 7;
        nc = fg * 16 + bg;
        return ( nc );
}

int             BHarf( String )
TEXT    *String;
{
```

```
        while ( *String )  {
                if ( *String >= 65 && *String <= 91 )
                                return ( *String );
                String++;
        }
        return  0;
}

int             dggwin()
{
        return( wxxopen( 4,
                                                32,
                                                21,
                                                72,
                                                "",
                                                ACTIVE+COOKED,
                                                0,
                                                0,
                                                4,
                                                178) );
}
int             descrwin()
{
        return( wxxopen( 3,
                                                30,
                                                20,
                                                70,
                                                "FILTRE OZELLIKLERI",
                                                ACTIVE+BORDER + BD2 + COOKED,
                                                0,
                                                0,
                                                7,
                                                32) );
}

descr( fnam  , fwin , curwin )
TEXT    *fnam;
int             fwin,
                curwin;
{
FILE    *fd;
COUNT   girdi;
COUNT   i;
TEXT    line[256];
TEXT    fcvnam[15];


                ConFile( fnam , fcvnam );
                fd = fopen ( fcvnam , "r" );
                if ( fd == ( FILE * ) NULL )
                                return 0;
                wselect ( fwin );
```

```
                i = 0;
                girdi = 1;
                woff();
                while ( readline ( fd , line ) != -1 ) {
                        if ( strcmp ( line , "#HEADER" ) && girdi )
                                        continue;
                        girdi = 0;
                        if ( !strncmp ( line , "))"  , 2 ) )
                                        break;
                        if ( *line != '#' )
                                        wxatsay ( fwin , i++ , 1 , line , 7 );
                }
                won();
                wselect ( curwin );
                fclose ( fd );
}

ConFile( var1  , var2 )
TEXT    *var1;
TEXT    *var2;
{
TEXT    fcvnam[15];
int             px;

                if ( var1 == ( TEXT * ) NULL ) {
                                *var2 = 0;
                                return 0;
                }
                strcpy    ( var2 , var1 );
                sagbosluk ( var2 );
                solbosluk ( var2 );
                px = pos ( var2 , ' ' , 0 );
                if ( px != -1 ) {
                        var2 [ px ] = '.';
                        solbosluk ( var2 + px + 1 );
                }
}
```

DIROK.C

BOIS interruptlarını değiştirerek dosya ve/veya directory'leri okur.

```
int  directory(char  *nam,char  *ara);
int  dir_first(char  _far *fil_nam,int  attrib);
int  set_dta(unsigned int  offset,unsigned int  segment);
int  dir_list(int  mod);
struct  ldef *ElemEk(struct  ldef *proot,unsigned char  *dts,unsigned char  *pts,int  *dcnt);
void  MyQuickSort(struct  ldef *prot,int  elems);
int  compl(char  *varl,char  *var2);
```

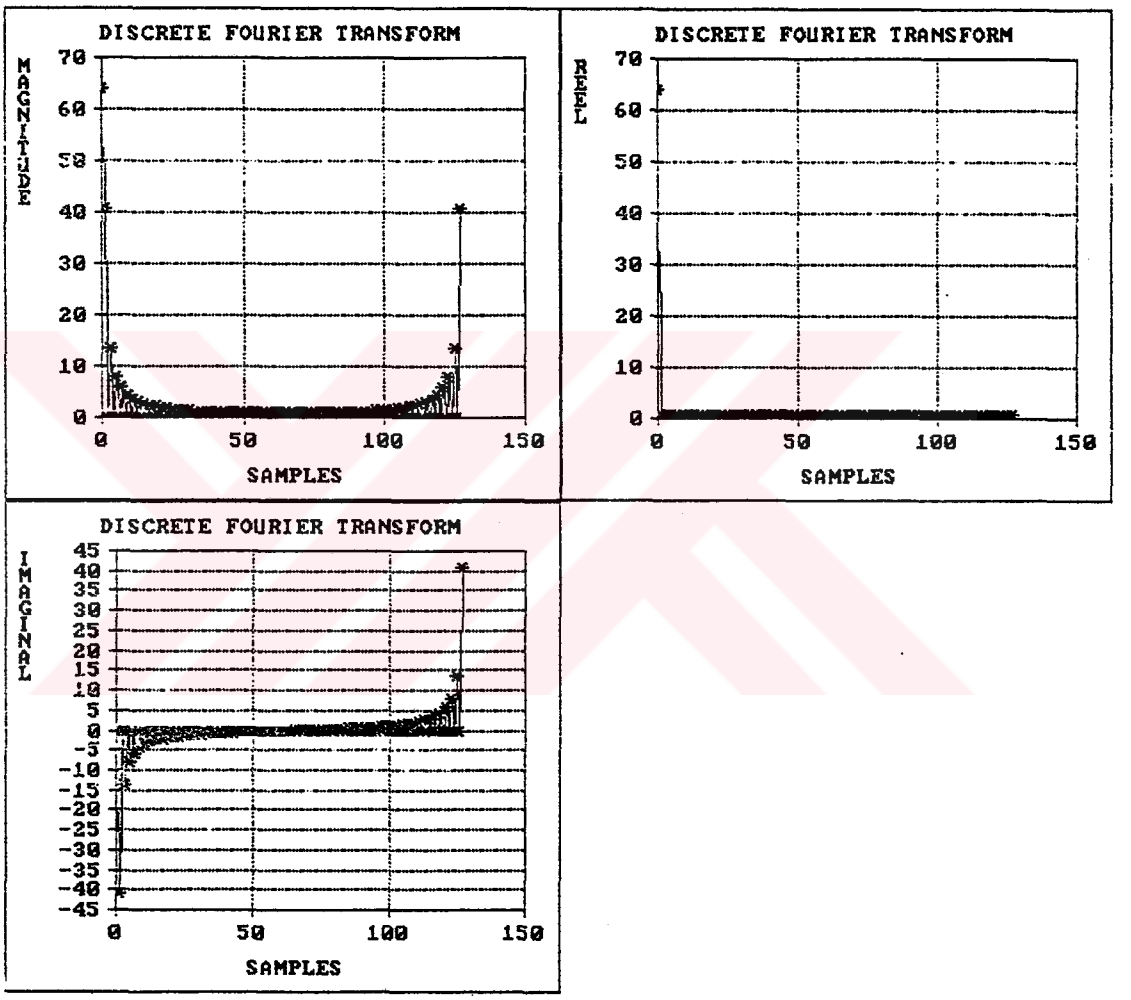
```
#include <utl2dfn.h>
#include <utl2def.h>
#include <stdio.h>
#include <dos.h>
#include <search.h>
#include "doku.h"
int             compi ( char * , char * );
char    far     *fil_nam;
char            mybuf[65];
char    far     *dta = &mybuf[0];
struct ldef {
        TEXT    dstr[80];
        TEXT    astr[80];
        struct  ldef    *next;
        struct  ldef    *prev;
};

typedef struct ldef PLIN;
void    MyQuickSort ( PLIN * , int );


extern  PLIN    *DirRoot;
extern  int             Direlem;
extern  PLIN    *pget_new();
extern  pen_dat_trns ();
PLIN    *ElemEk();


directory( nam , ara )
char    *nam , *ara;
{
char    file[255];
int     ATTrib;
                Direlem = 0;
                DirRoot = ( PLIN * ) NULL;
        if ( !*nam )    /* name */
                nam = "*";
        if ( !*ara )    /* ext  */
                ara = "*";
        if ( !strcmp( ara , "<dir>" ) )  {
                ATTrib = 16;/* DIRECTORY */
                ara = "*";
        } else
                ATTrib = 0;/* ONLY_FILES */

        sprintf ( file , "%s.%s" , nam , ara );
        fil_nam = (char far *) file;
        dir_first( fil_nam , ATTrib );
}
dir_first( fil_nam  , attrib )
char    far     *fil_nam;
int             attrib;
```

```
{
        static  union   REGS    inregs , outregs;
        static  struct  SREGS   segregs;
        static  unsigned        int     myreg_es ,
                                        myreg_bx;

                        /* get the original DTA location es:bx */
        inregs.h.ah = 0X2F;
        int86x( 0X21 , &inregs , &outregs , &segregs );
        myreg_es = segregs.es;
        myreg_bx = outregs.x.bx;
                        /* set DTA to mybuf */
        set_dta( FP_OFF( dta ) , FP_SEG( dta ) );

                        /* ds:dx --> mypath */
        inregs.x.dx = FP_OFF ( fil_nam );
        segregs.ds  = FP_SEG ( fil_nam );
        inregs.x.cx = attrib;   /* set file attrib*/
                                /* 00H  -->NORMAL */
                                /* 01H  -->RD_ONL */
                                /* 00H  -->HIDDEN */
                                /* 00H  -->SYSTEM */
        inregs.h.ah = 0X4E;     /* interrupt 4E of 21 */
                                /* read the first match */
        int86x( 0X21 , &inregs , &outregs , &segregs );
        if ( outregs.x.cflag ) {
                set_dta( myreg_bx , myreg_es );
                return -1;      /* match not found */
        } else
                dir_list ( attrib );    /* add the file to the list */

        for(;;) {

                                /* find the next match */
                inregs.h.ah = 0X4F;
                int86x ( 0X21 , &inregs , &outregs , &segregs );
                if ( outregs.x.cflag )
                        break;
                else
                        dir_list( attrib );

        }
        set_dta( myreg_bx , myreg_es );
        if ( DirRoot != ( PLIN * ) NULL )
                MyQuickSort ( DirRoot , Direlem );
}

set_dta( offset , segment )
unsigned        int     segment,
                        offset;
{
        static  union   REGS    inregs , outregs;
```

```
        static  struct  SREGS   segregs;
                        /*set the DTA to buffer */
        inregs.x.dx = offset;
        segregs.ds  = segment;

                        /*set DTA location ds:dx   */
        inregs.h.ah = 0X1A;
        int86x( 0X21 , &inregs , &outregs , &segregs );

}

dir_list( mod )
int     mod;
{
COUNT   OK,
                ofs;
char    patStr[81];
char    patS[81];

        OK = 0;
        switch ( mod ) {
                case    16:/*DIR*/
                                if ( (int ) mybuf[21] == 16 ) {
                                        OK = 1;
                                        sprintf ( patS , "%-*.*s" , 12 , 12 , mybuf + 30 );
                                        strcat  ( patS , "<DIR>");
                                }
                                break;
                case    0:/*FILES*/
                                if ( (int ) mybuf[21] == 32 ) {
                                        OK = 1;
                                        sprintf ( patStr , "%-*.*s" , 12 , 12 , mybuf + 30 );
                                        ofs = pos ( patStr , '.' , 0 );
                                        if ( ofs != -1 ) {
                                                patStr[ofs] = 0;
                                            sprintf (  patS , " %-8.8s %-3.3s", patStr , patStr + ofs + 1 )
                                        } else
                                                strcpy ( patS , patStr );
                                }
                                break;
        }
        if ( OK )
                DirRoot = ElemEk ( DirRoot , patS , "", &Direlem );
}

/*********************************************************************/
/* Construct two-ways linked list and calculate list elem number */
/* root->prev is linked to tail and tail->next is NULL           */
/*********************************************************************/
PLIN    *ElemEk ( proot , dts  , pts ,  dcnt )
PLIN    *proot;
TEXT    *dts,
```

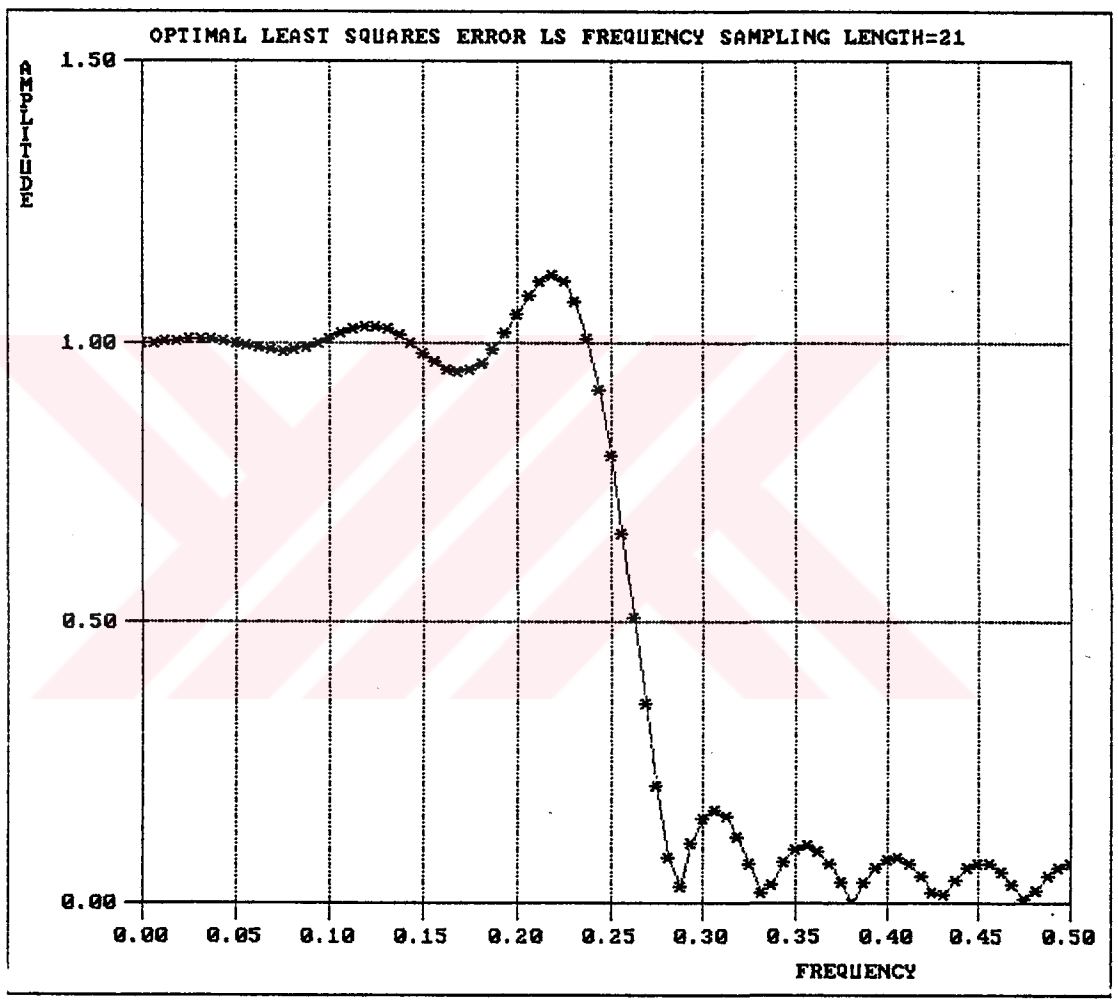
```
                *pts;
int             *dcnt;
{
static  PLIN    *ptmp1,
                                *ptmp0;

                (*dcnt)++;
                ptmp1 = pget_new ();
                pen_dat_trns ( ptmp1 , dts , pts );
                if ( proot == ( PLIN * ) NULL ) {
                                proot = ptmp1;
                } else {
                                ptmp1->prev = ptmp0;
                                ptmp0->next = ptmp1;
                }
                ptmp0 = ptmp1;
                proot->prev = ptmp0;
                return ( PLIN * ) proot;
}

void    MyQuickSort ( prot  , elems )
PLIN    *prot;
int             elems;
{
TEXT    *key;
int             ofs;
PLIN    *tmrot;

                tmrot = prot;
                key = ( TEXT *  ) malloc ( 80 * elems );
                ofs = 0;
                while ( tmrot != ( PLIN * ) NULL ) {
                        strcpy ( key + 80*ofs , tmrot->dstr );
                        ofs++;
                        tmrot = tmrot->next;
                }
                qsort ( (void *)key , (size_t)elems , 80 * sizeof ( char ) , comp1 );
                tmrot = prot;
                ofs = 0;
                while ( tmrot != ( PLIN * ) NULL ) {
                        strcpy ( tmrot->dstr , key + 80*ofs );
                        ofs++;
                        tmrot = tmrot->next;
                }
                free ( ( char * ) key );
}

int     comp1 ( var1 , var2 )
char    *var1,
                *var2;
{
                return ( strcmp ( var1 , var2 ) );
```

EK.B PROGRAM CIKTILARI

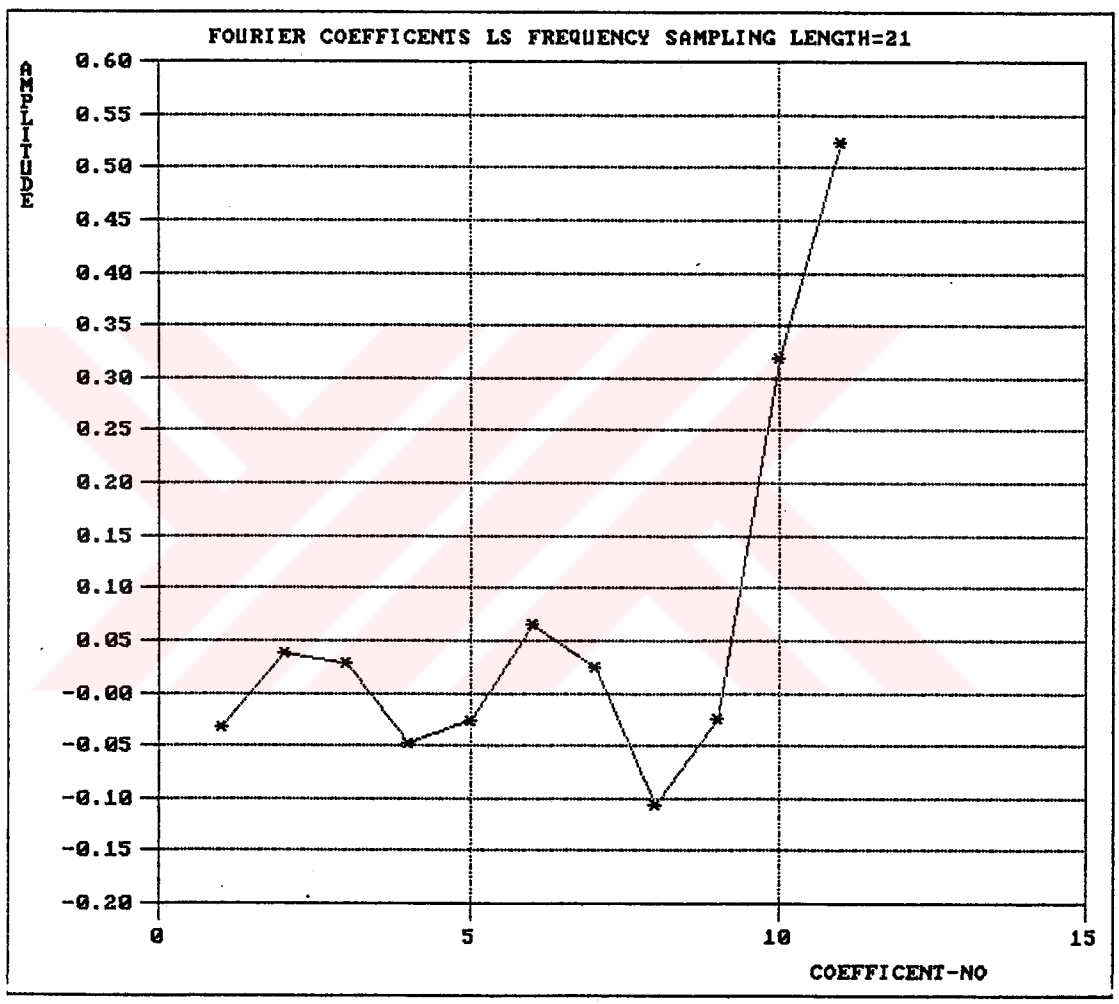


Şekil B.1 N=128 Kare dalganın DFT'si

```
FSLMS039.FIR

FREKANS ORNEKLEME METODU
OPTIMUM LEAST SQUARES ERROR


KOSE FREKANSI            : 0.250

FILTRE UZUNLUGU          : 21

DC ORNEK ( 1/0 )         : 0

HESAPLANAN FRE.CEV.SAY. : 80

))
FOURIER COEFFICENTS LS FREQUENCY SAMPLING LENGTH=21
COEFFICENT-NO,
AMPLITUDE,
1,-0.032480
2,0.038187
3,0.028817
4,-0.047619
5,-0.026427
6,0.065171
7,0.024917
8,-0.106999
9,-0.024078
10,0.318607
11,0.523810
))
```



Şekil B.2 N=21 Alçakgeçiren filtrenin (frekans örnekleme metodu) frekans cevabı



Şekil B.3 N=21 Alçakgeçiren filtrenin (frekans örnekleme metodu) fourier katsayıları.

FSLMS040.FIR

FREKANS ORNEKLEME METODU
OPTIMUM LEAST SQUARES ERROR

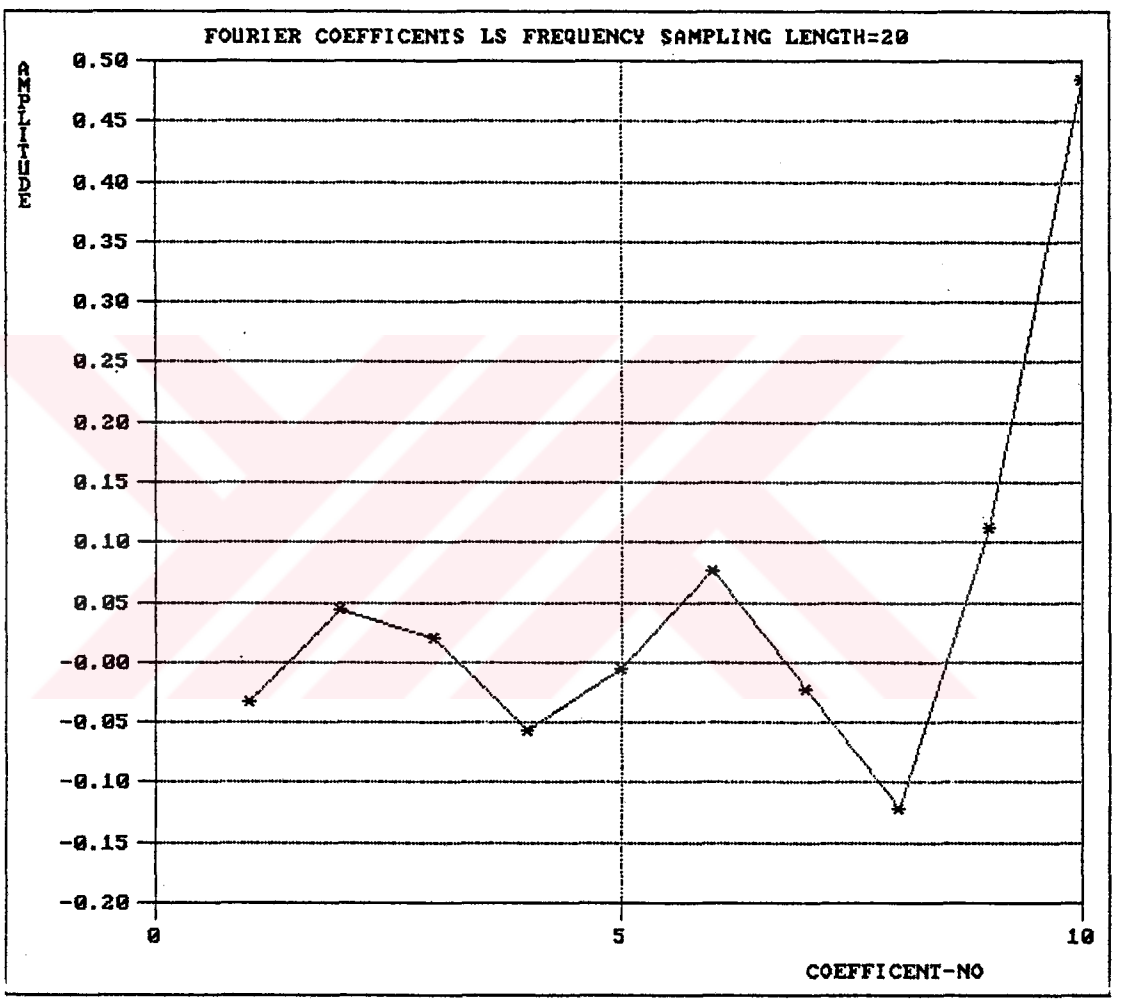
KOSE FREKANSI : 0.250

FILTRE UZUNLUGU : 20

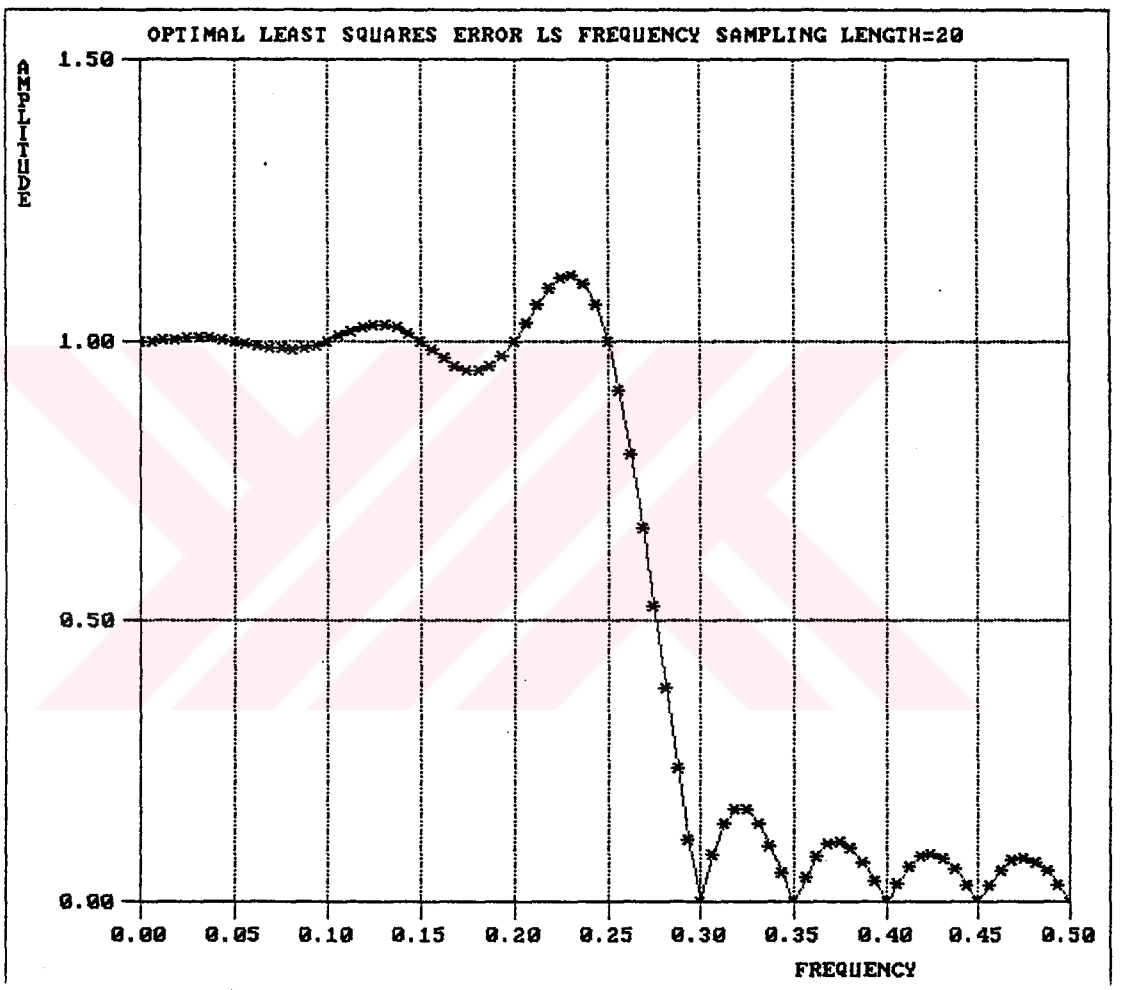
DC ORNEK (1/0) : 0

HESAPLANAN FRE.CEV.SAY. : 80

))
FOURIER COEFFICENTS LS FREQUENCY SAMPLING LENGTH=20
COEFFICENT-NO,
AMPLITUDE,
1,-0.032573
2,0.043843
3,0.020711
4,-0.057021
5,-0.005159
6,0.076751
7,-0.022339
8,-0.120711
9,0.111910
10,0.484588
))

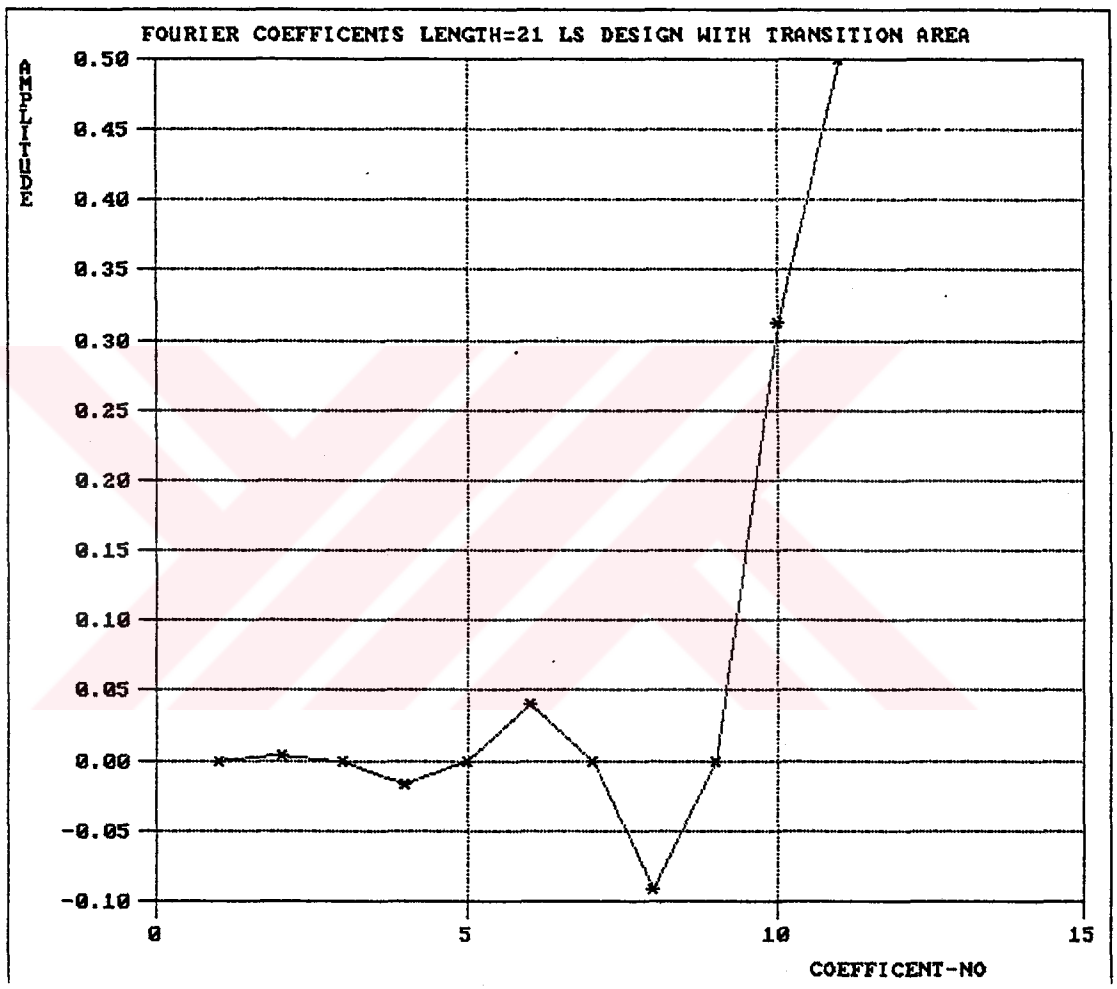


Şekil B.4 N=20 Alçakgeçiren filtrenin (frekans örnekleme metodu) fourier katsayıları.

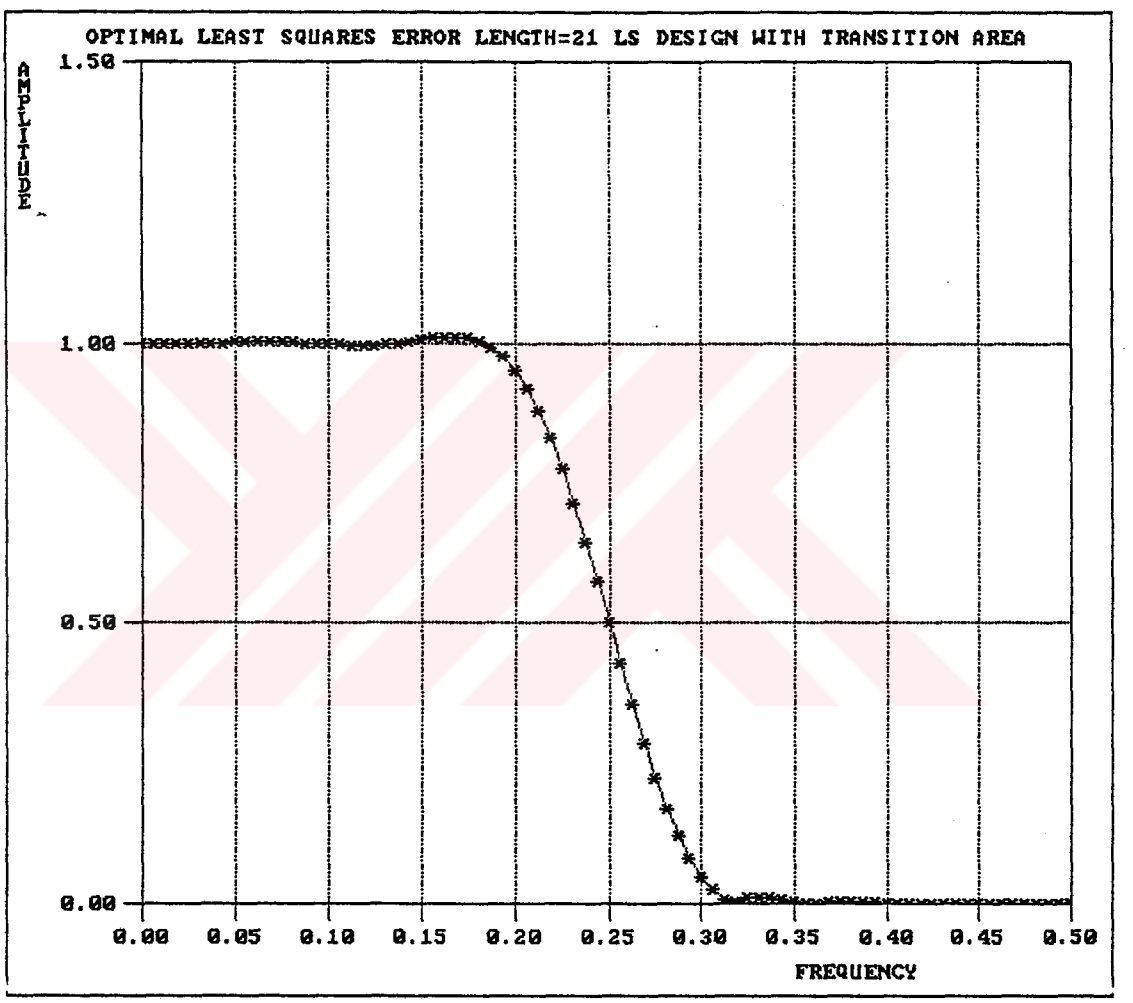


Şekil B.5 N=20 Alçakgeçiren filtrenin (frekans örnekleme metodu) frekans cevabı.

```
FSLPB067.FIR

OPTIMUM LEAST SQUARES ERROR
GECIS BANDLI - FIR FILTRE
0. raised cosine
1. 1ci derece
n. n.ci derece

KOSE FREKANSI BASLANGICI: 0.200

KOSE FREKANSI SONU      : 0.300

FILTRE UZUNLUGU         : 21

GECIS TIPI              : 1

HESAPLANAN FRE.CEV.SAY. : 0

))
FOURIER COEFFICENTS LENGTH=21 LS DESIGN WITH TRANSITION AREA
COEFFICENT-NO,
AMPLITUDE,
1,0.000000
2,0.003865
3,0.000000
4,-0.016729
5,-0.000000
6,0.040528
7,0.000000
8,-0.091078
9,-0.000000
10,0.313100
11,0.500000
))
```



Şekil B.6 N=21 Geçiş bandlı alçakgeçiren filtrenin fourier katsayıları.



Şekil B.7 N=21 Geçiş bandlı alçakgeçiren filtrenin frekans cevabı.

LSWIN073.FIR

LS ERROR DESIGN
WINDOWS FOR FIR FILTERS

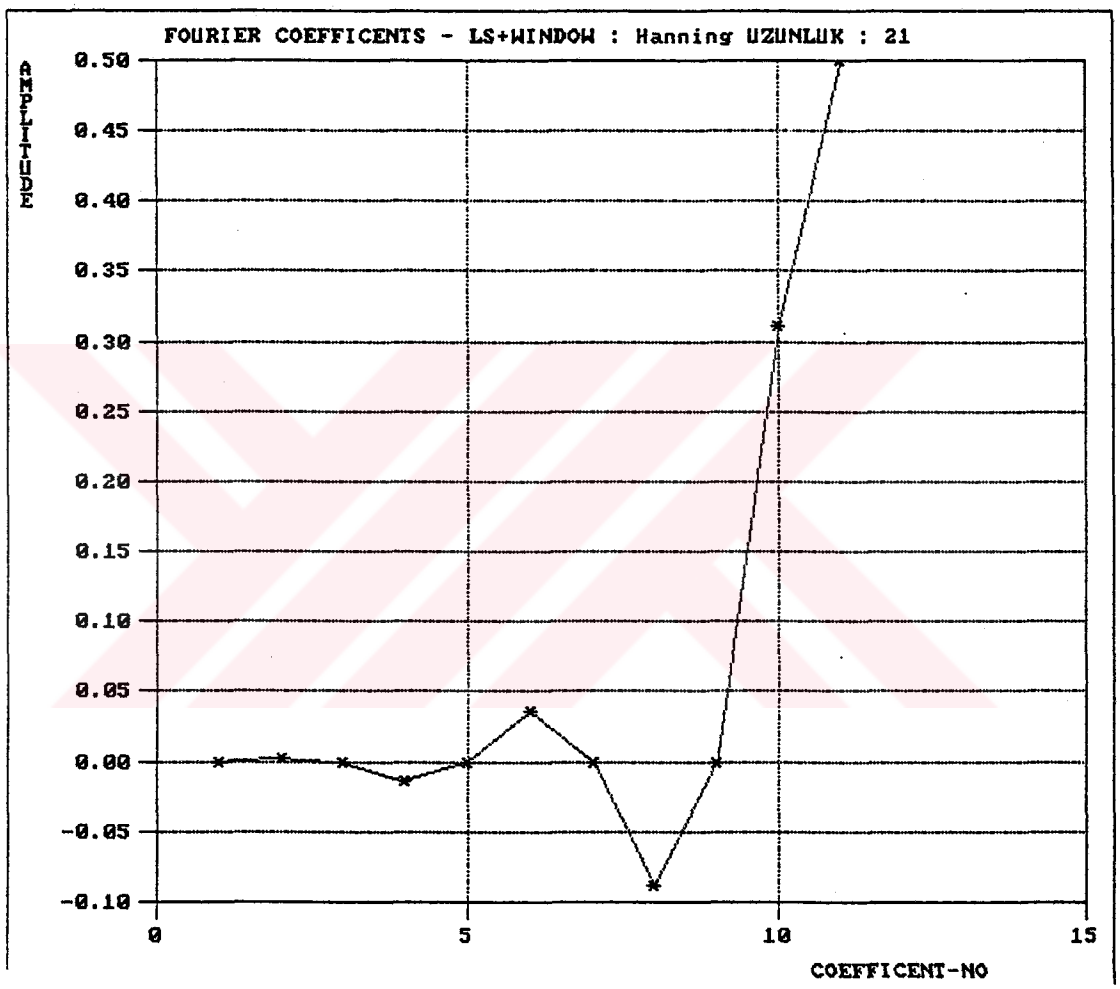
KOSE FREKANSI : 0.250

FILTRE UZUNLUGU : 21

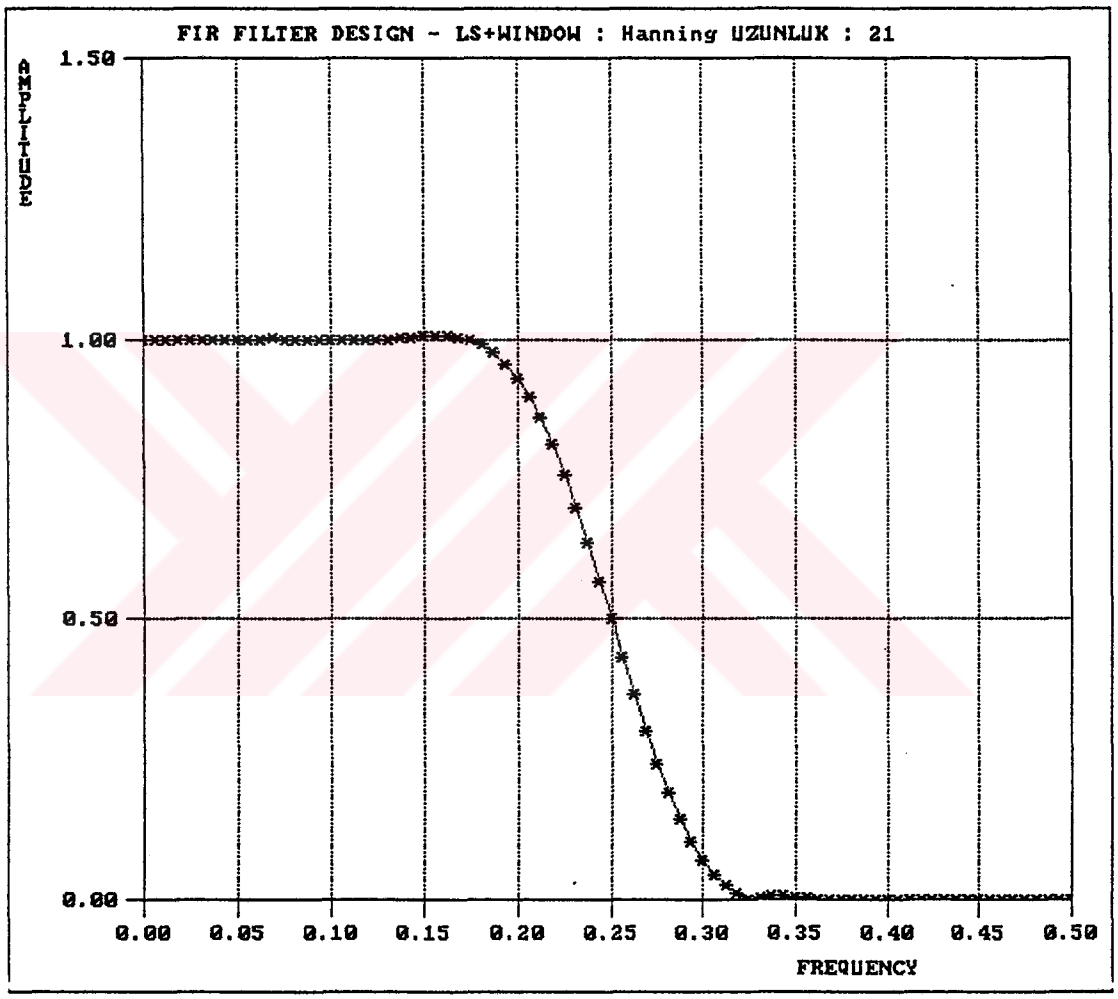
FILTRE TIPI : 2

HESAPLANAN FRE.CEV.SAY. : 80

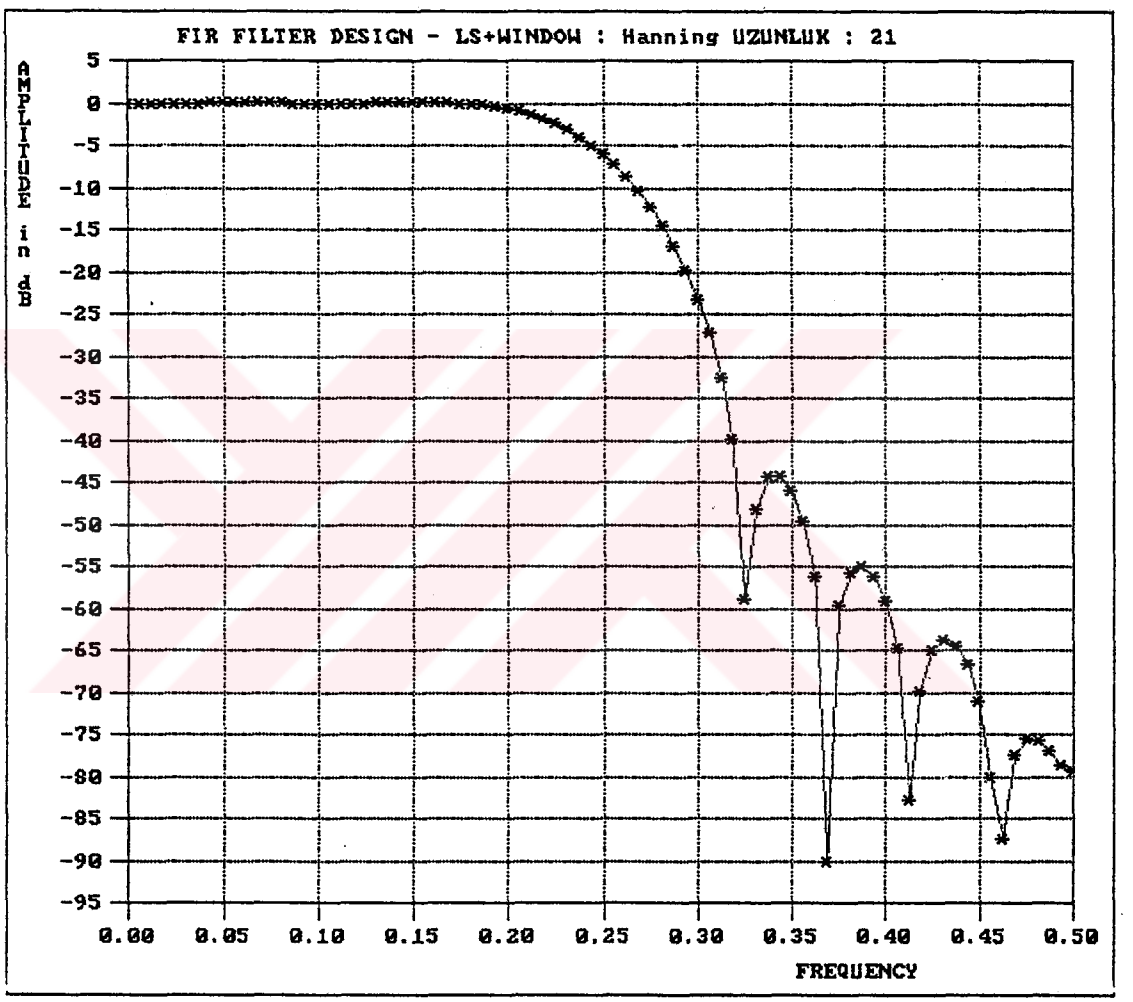
))
FOURIER COEFFICENTS - LS+WINDOW : Hanning UZUNLUK : 21
COEFFICENT-NO,
AMPLITUDE,
1,0.000000
2,0.002807
3,-0.000000
4,-0.013291
5,0.000000
6,0.036361
7,-0.000000
8,-0.087793
9,0.000000
10,0.311863
11,0.500000
))



Şekil B.8 N=21 Alçakgeçiren filtrenin (window: hanning) fourier katsayıları.



Şekil B.9 N=21 Alçakgeçiren filtrenin (window : hanning) frekans cevabı.



Şekil B.10 N=21 Alçakgeçiren filtrenin (window : hanning) frekans cevabı. (dB)

LSWIN077.FIR

LS ERROR DESIGN
WINDOWS FOR FIR FILTERS

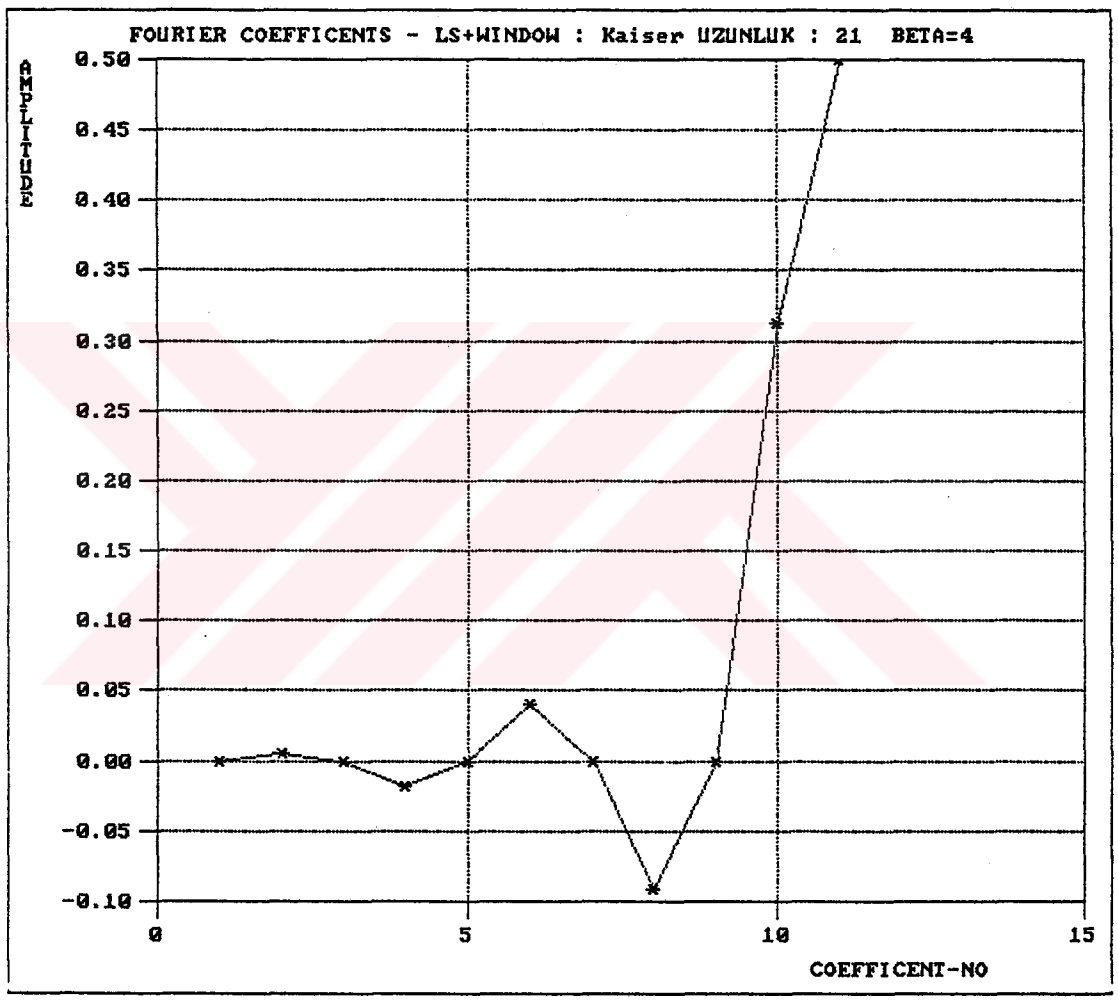
KOSE FREKANSI : 0.250

FILTRE UZUNLUGU : 21

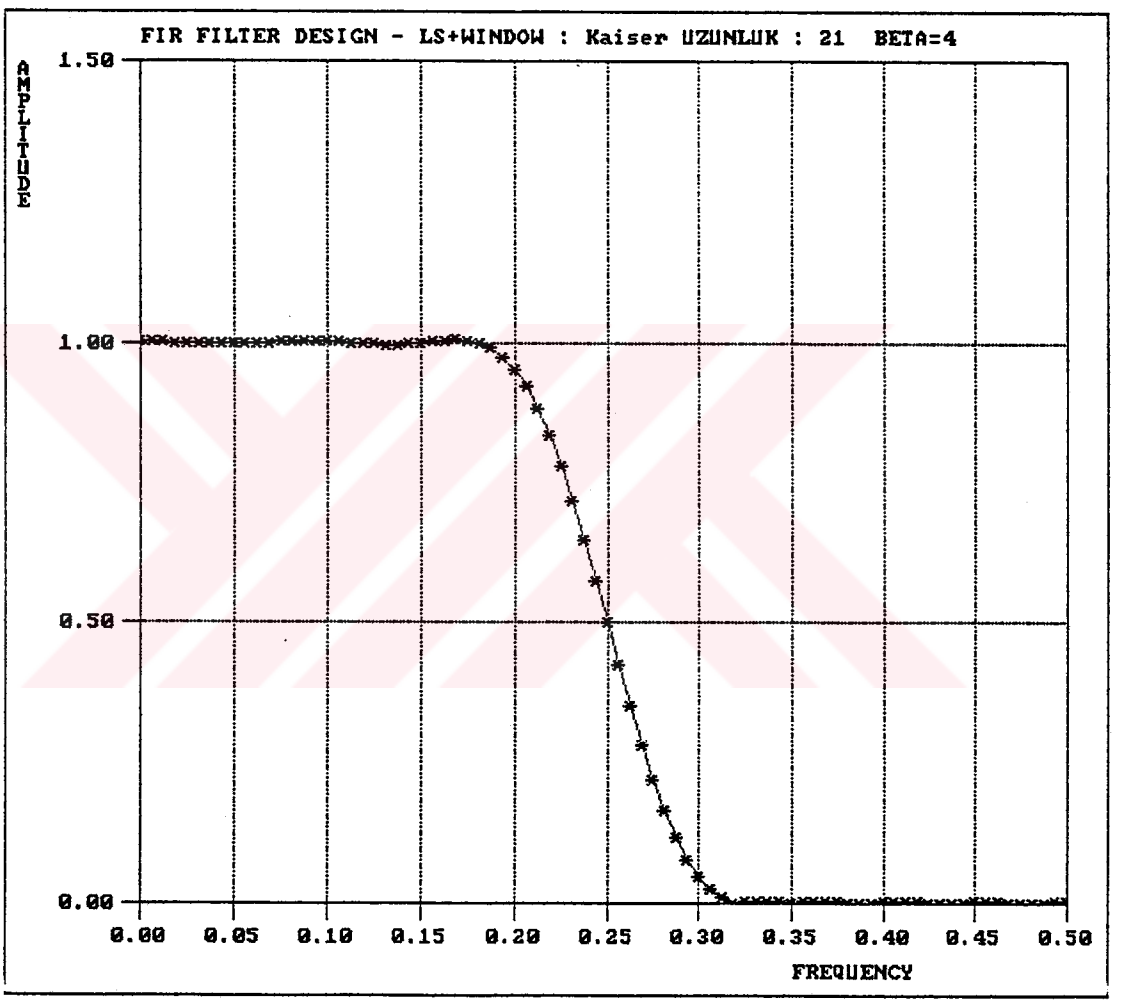
FILTRE TIPI : 5 , Kaiser , BETA=4

HESAPLANAN FRE.CEV.SAY. : 80

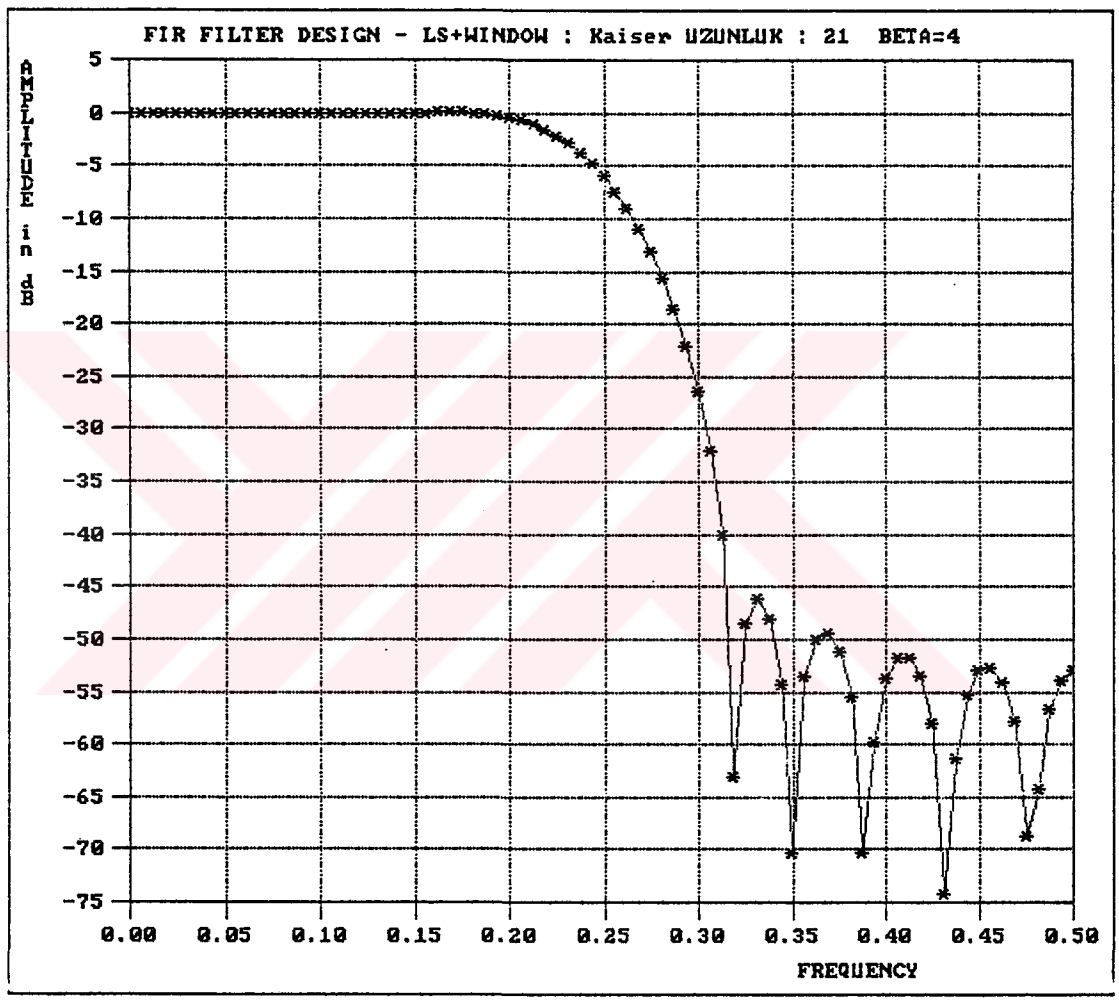
))
FOURIER COEFFICENTS - LS+WINDOW : Kaiser UZUNLUK : 21 BETA=4
COEFFICENT-NO,
AMPLITUDE,
1,0.000000
2,0.006000
3,-0.000000
4,-0.017498
5,0.000000
6,0.040326
7,-0.000000
8,-0.090558
9,0.000000
10,0.312849
11,0.500000
))



Şekil B.11 N=21 Alçakgeçiren filtrenin (window : Kaiser : beta=4) fourier katsayıları.

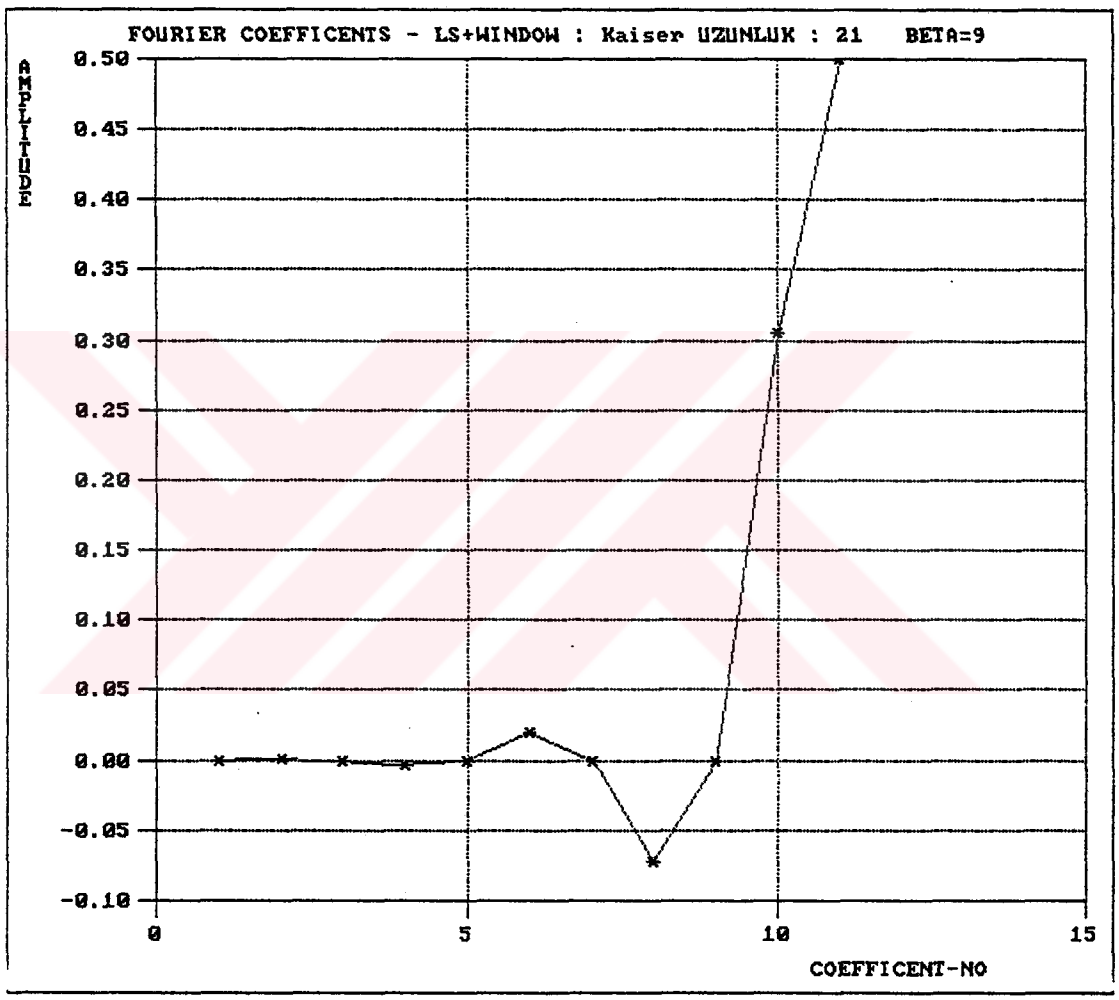


Şekil B.12 N=21 Alçakgeçiren filtrenin (window : Kaiser : beta=4) frekans cevabı.

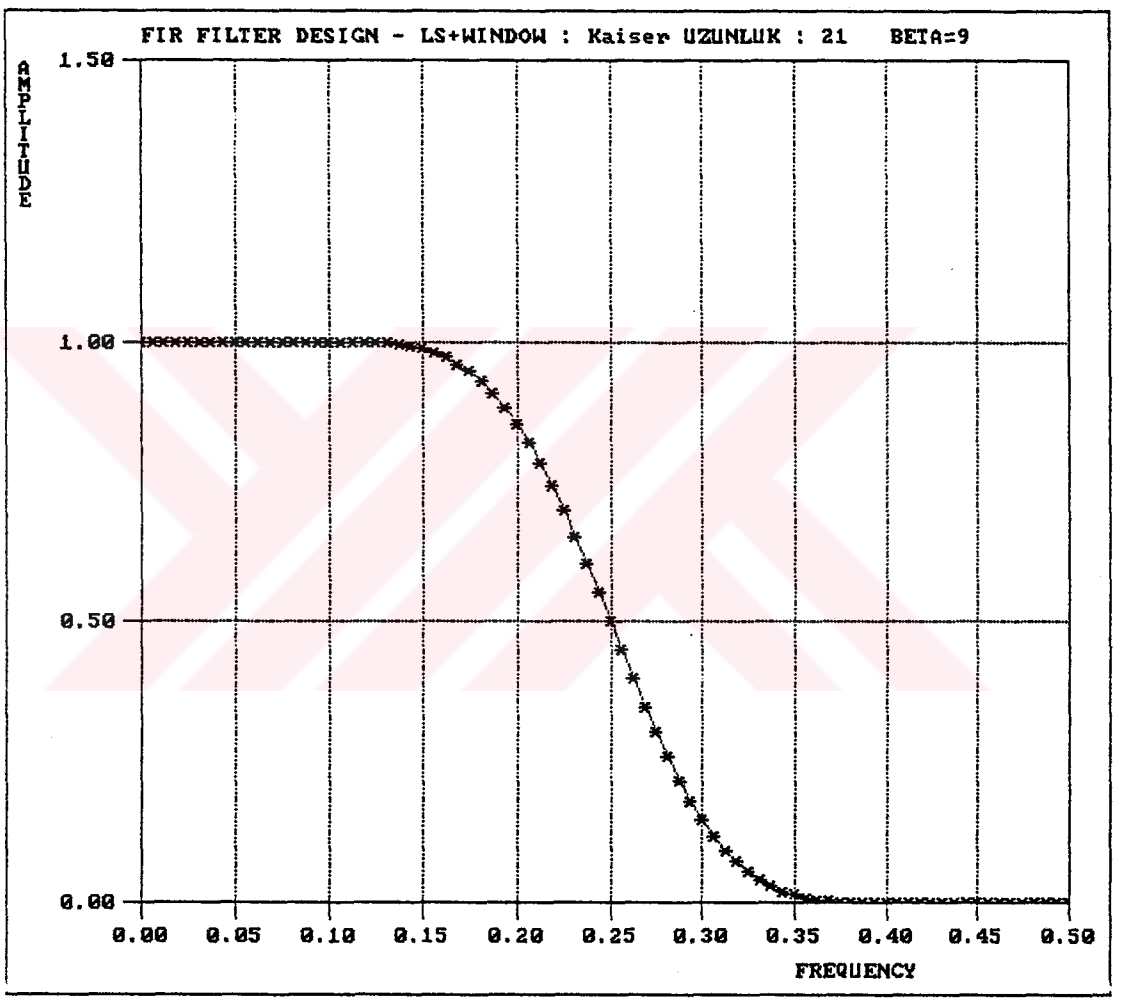


Şekil B. 13 N=21 Alçakgeçiren filtrenin (window : Kaiser : beta=4) frekans cevabı. (dB)

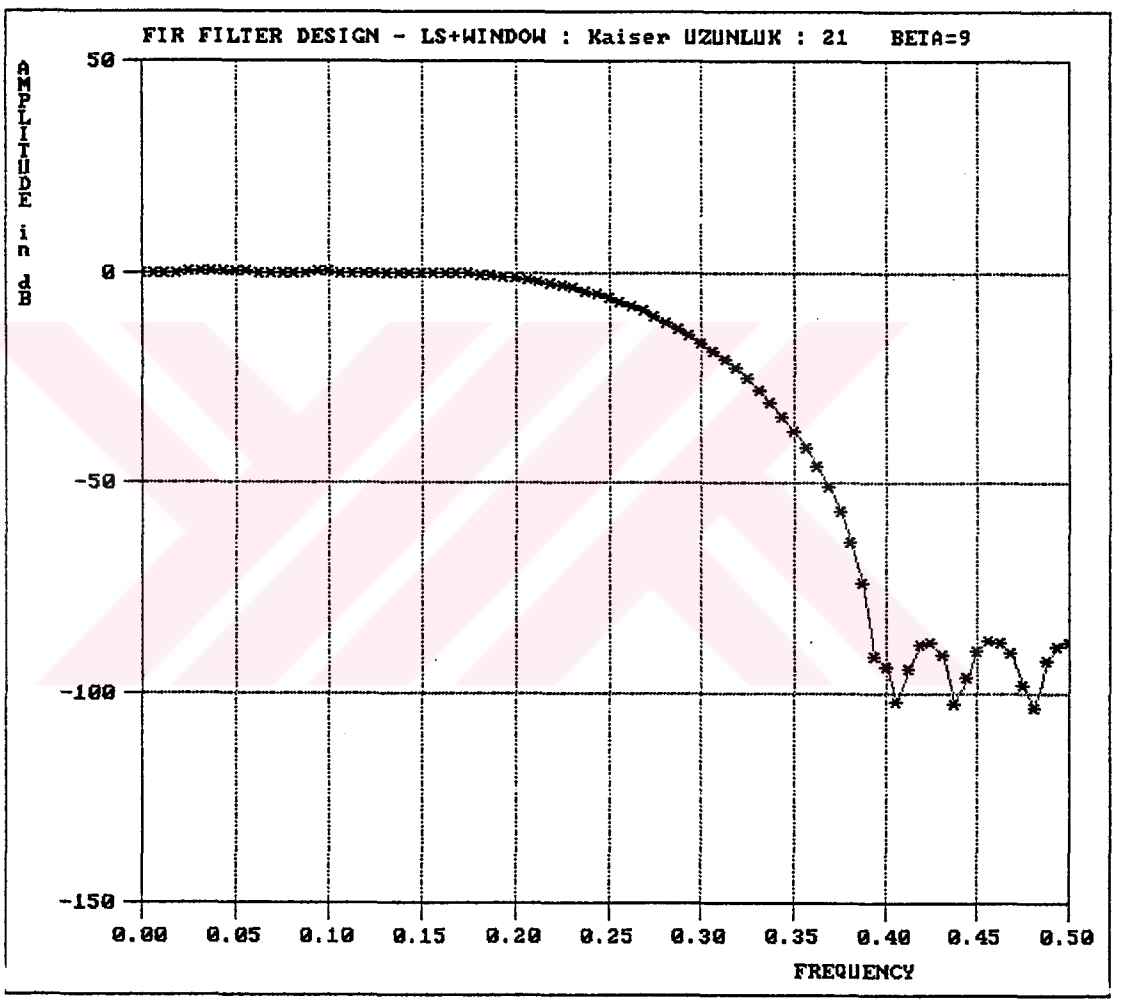
```
LSWIN078.FIR

LS ERROR DESIGN
WINDOWS FOR FIR FILTERS


KOSE FREKANSI           : 0.250

FILTRE UZUNLUGU         : 21

FILTRE TIPI             : 5, Kaiser , BETA=9

HESAPLANAN FRE.CEV.SAY. : 80

))
FOURIER COEFFICENTS - LS+WINDOW : Kaiser UZUNLUK : 21   BETA=9
COEFFICENT-NO,
AMPLITUDE,
1,0.000000
2,0.000342
3,-0.000000
4,-0.004134
5,0.000000
6,0.020537
7,-0.000000
8,-0.071823
9,0.000000
10,0.305059
11,0.500000
))
```



Şekil B.14 N=21 Alçakgeçiren filtrenin (window : Kaiser : beta=9) fourier katsayıları.



Şekil B.15 N=21 Alçakgeciren filtrenin (window : Kaiser : beta=9) frekans cevabı.



Şekil B.16 N=21 Alçakgeçiren filtrenin frekans cevabı. (dB)

(window : Kaiser : beta=9)

LSWIN079.FIR

LS ERROR DESIGN
WINDOWS FOR FIR FILTERS

KOSE FREKANSI : 0.250

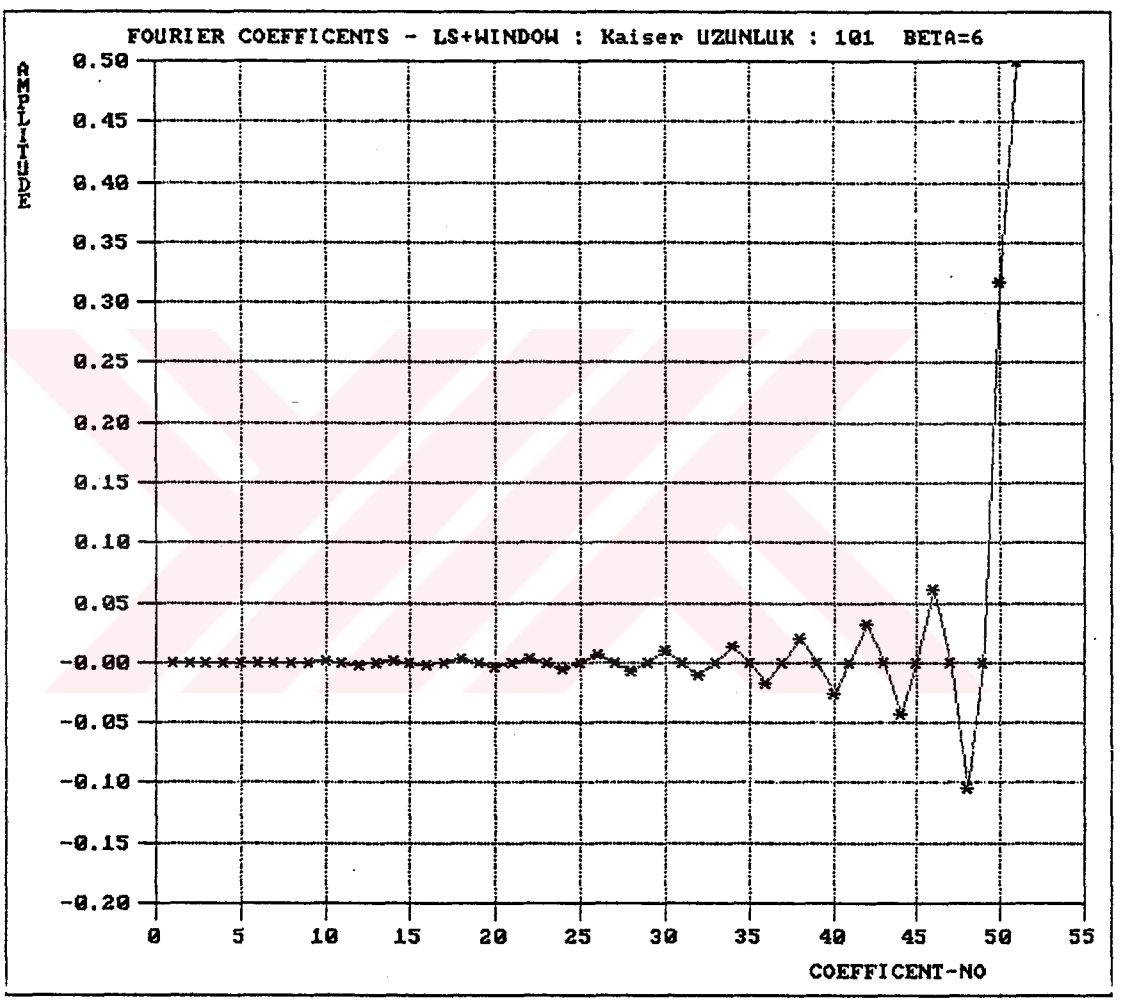
FILTRE UZUNLUGU : 101

FILTRE TIPI : 5, Kaiser , BETA=6

HESAPLANAN FRE.CEV.SAY. : 80

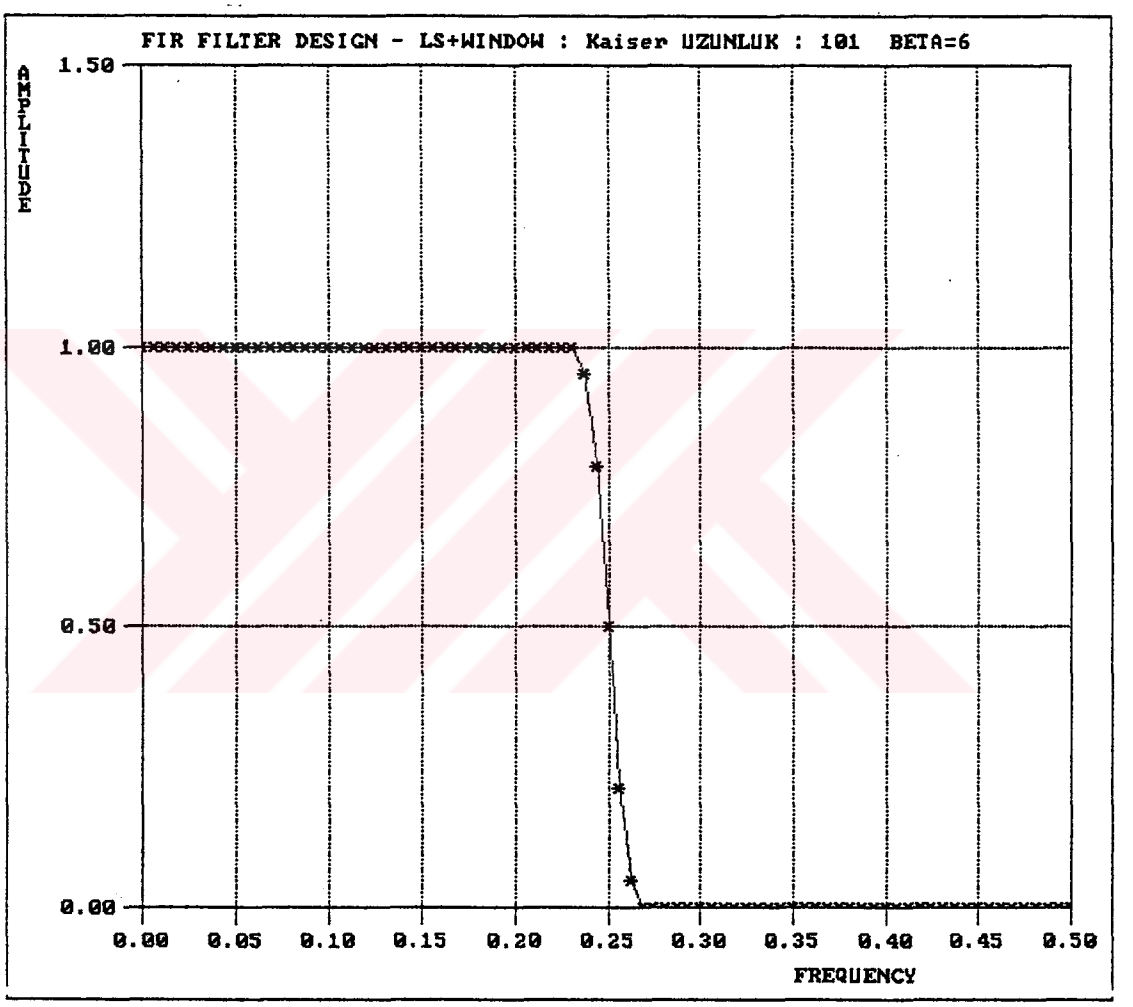
))
FOURIER COEFFICENTS - LS+WINDOW : Kaiser UZUNLUK : 101 BETA=6
COEFFICENT-NO,
AMPLITUDE,
1,-0.000000
2,0.000134
3,-0.000000
4,-0.000237
5,0.000000
6,0.000378
7,-0.000000
8,-0.000565
9,-0.000000
10,0.000806
11,-0.000000
12,-0.001111
13,0.000000
14,0.001491
15,-0.000000
16,-0.001958
17,-0.000000
18,0.002525
19,-0.000000
20,-0.003207
21,0.000000
22,0.004023
23,-0.000000
24,-0.004995
25,-0.000000
26,0.006149
27,-0.000000
28,-0.007522
29,0.000000
30,0.009160
31,-0.000000
32,-0.011130
33,0.000000
34,0.013528
35,-0.000000

```
36,-0.016501
37,0.000000
38,0.020290
39,-0.000000
40,-0.025310
41,0.000000
42,0.032345
43,-0.000000
44,-0.043087
45,0.000000
46,0.061939
47,-0.000000
48,-0.105062
49,0.000000
50,0.317962
51,0.500000
))
```



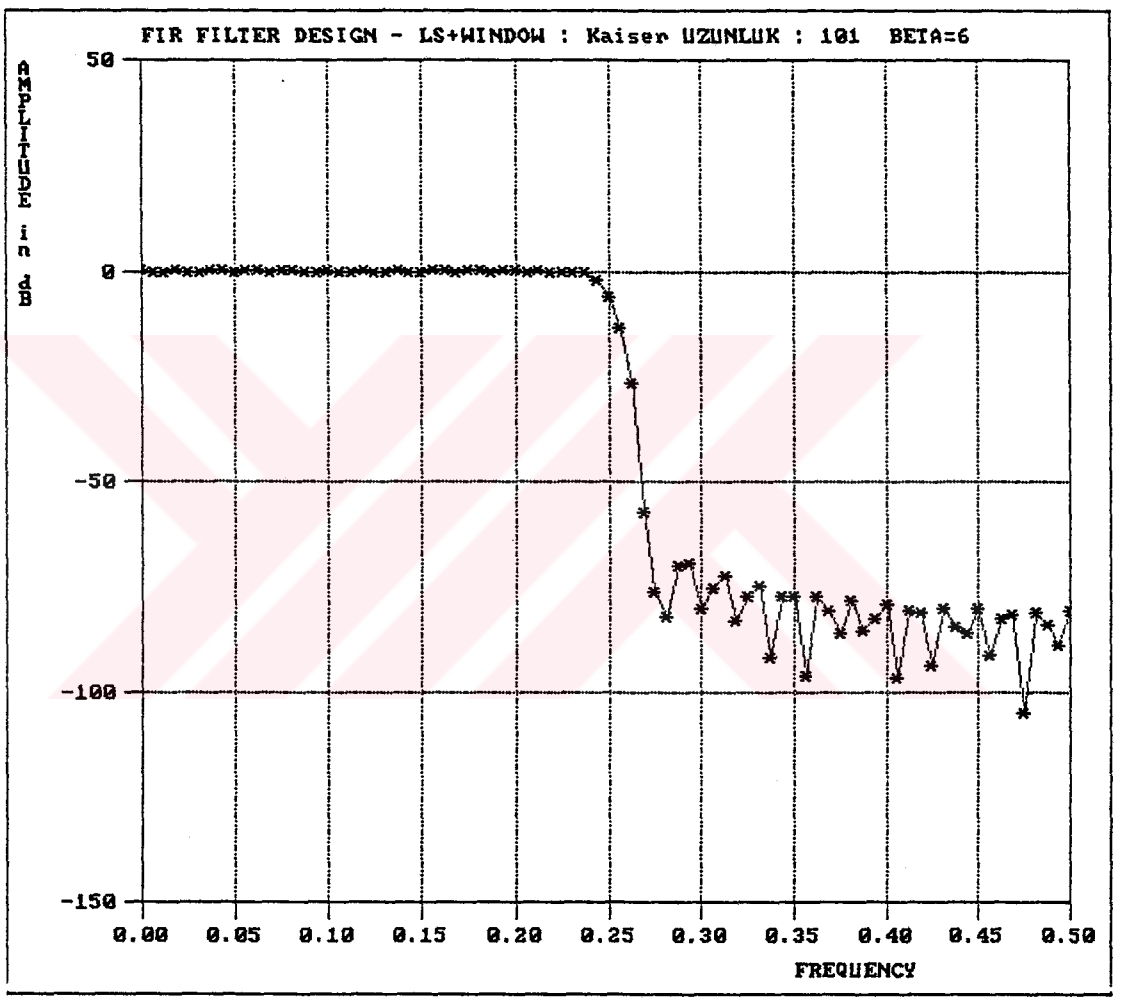
Şekil B.17 N=101 Alçakgeçiren filtrenin fourier katsayıları.

(window : Kaiser : beta = 6)



Şekil B.18 N=101 Alçakgeçiren filtrenin frekans cevabı.

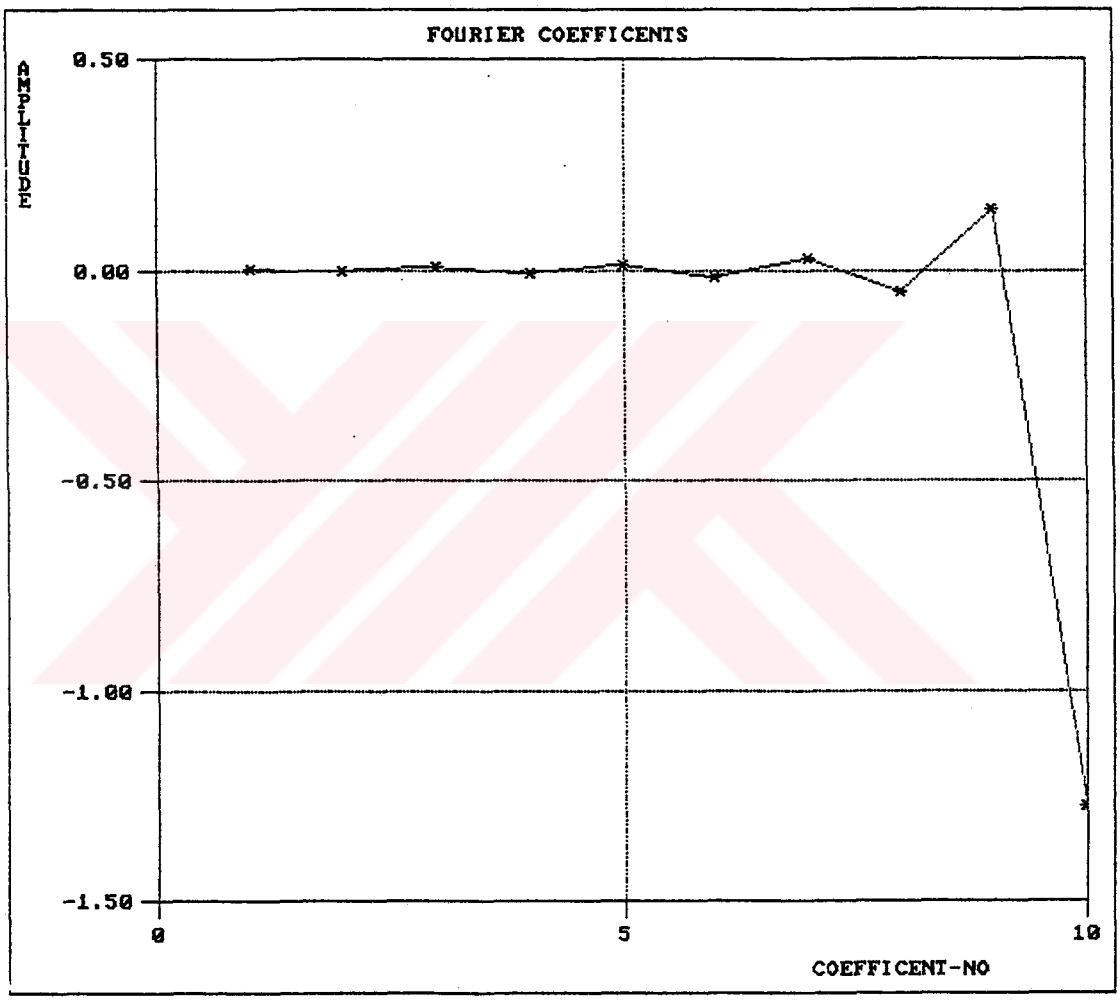
(window : Kaiser : beta= 6)



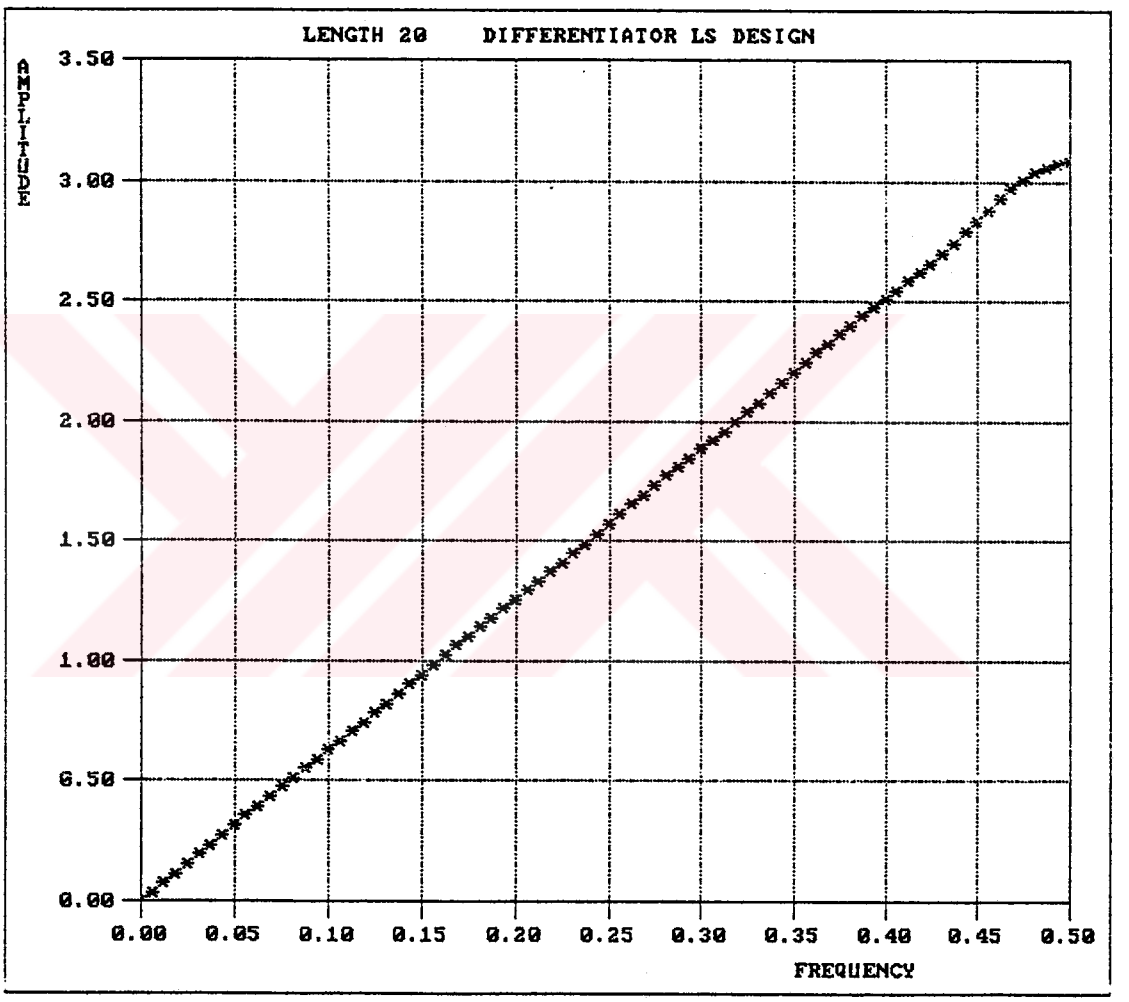
Şekil B.19 N=101 Alçakgeçiren filtrenin frekans cevabı. (dB)

(window : Kaiser : beta = 6)

```
DIFFR064.FIR

FREKANS ORNEKLEME METODU
LINEAR PHASE DIFFERENTIATOR


FILTRE UZUNLUGU          : 20

HESAPLANAN FRE.CEV.SAY. : 80

))
FOURIER COEFFICENTS
COEFFICENT-NO,
AMPLITUDE,
1,0.003527
2,-0.004406
3,0.005659
4,-0.007534
5,0.010523
6,-0.015719
7,0.025984
8,-0.050930
9,0.141471
10,-1.273240
))
```

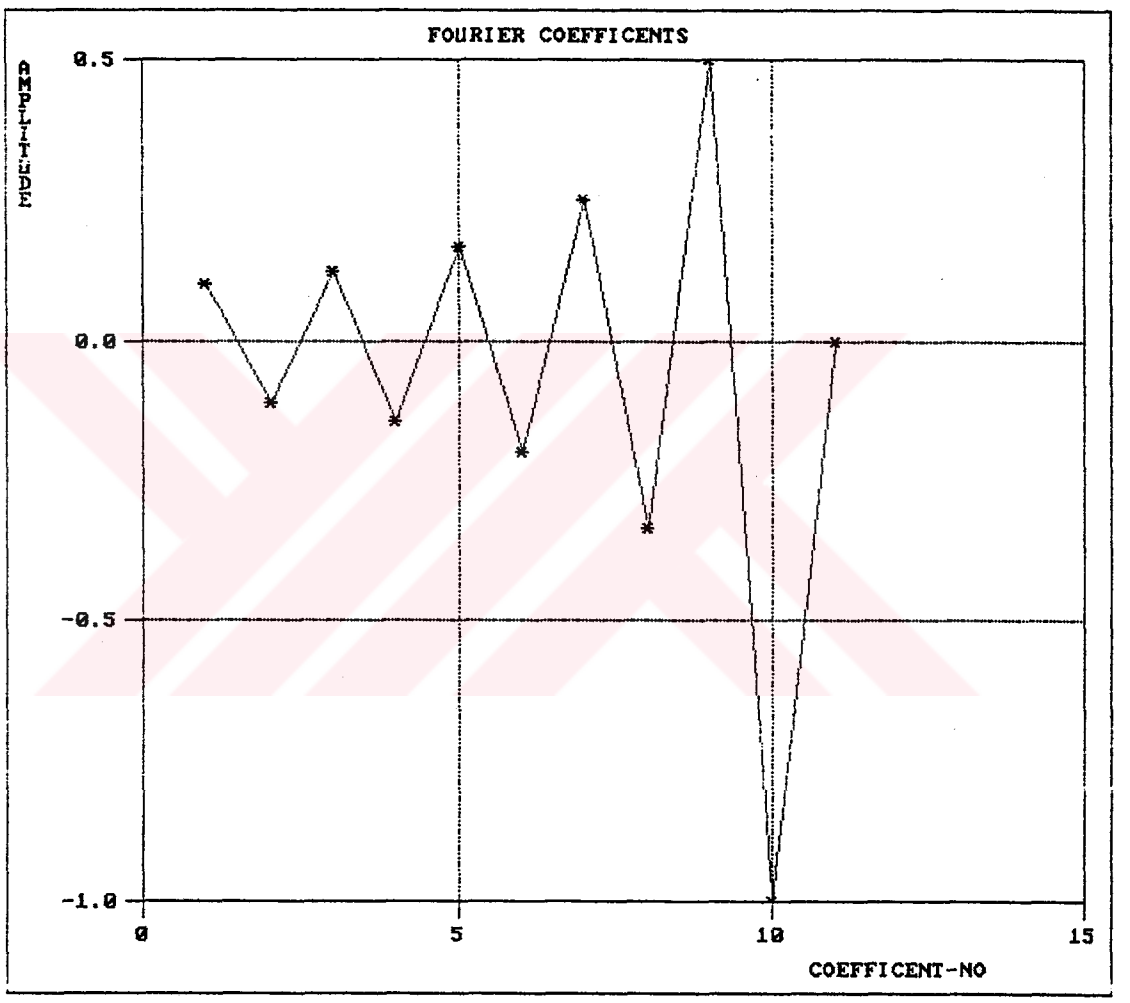


Şekil B.20 N=20 Differentiator'ın katsayıları.

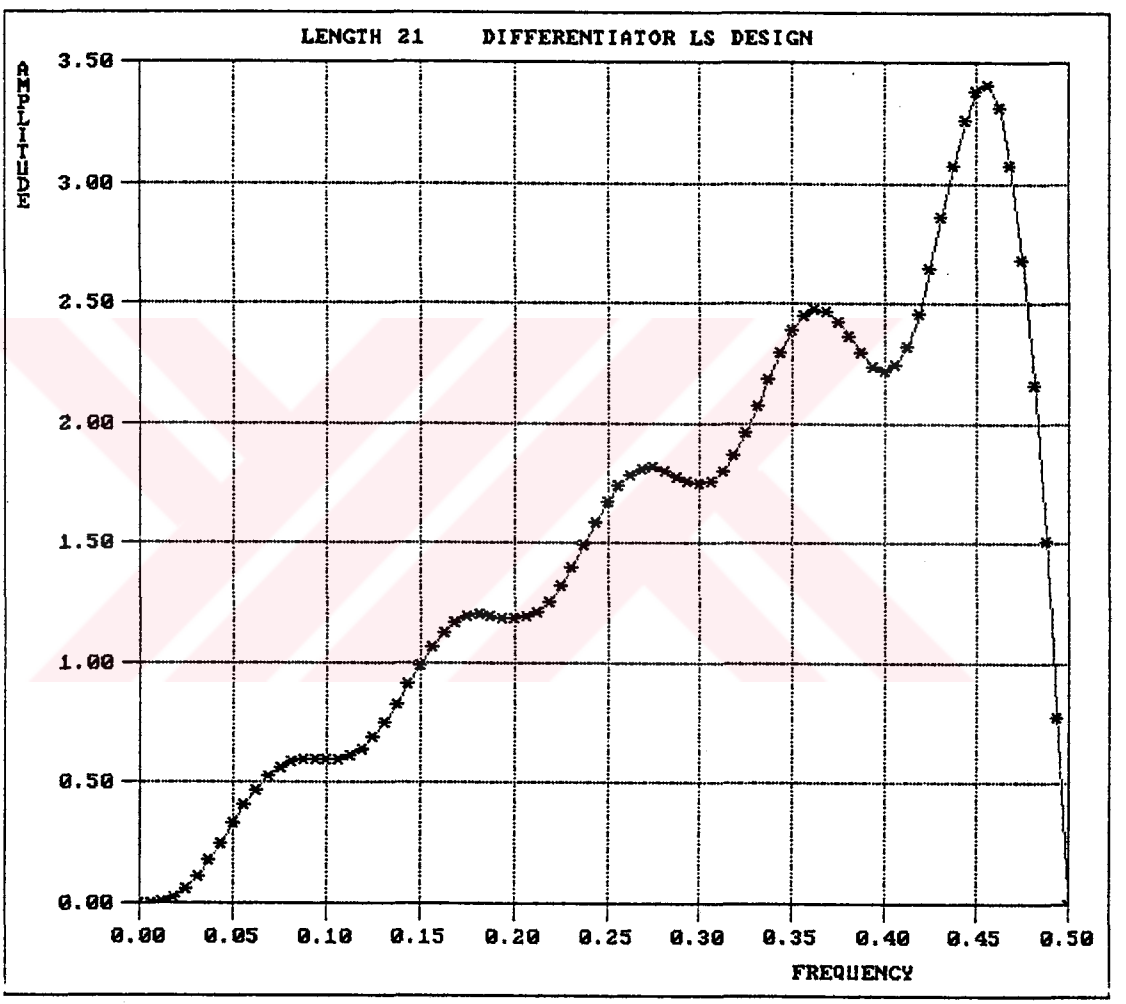


Şekil B.21 N=20 Differentiator'ın frekans cevabı.

```
DIFFR063.FIR

FREKANS ORNEKLEME METODU
LINEAR PHASE DIFFERENTIATOR


FILTRE UZUNLUGU          : 21

HESAPLANAN FRE.CEV.SAY. : 80

))
FOURIER COEFFICENTS
COEFFICENT-NO,
AMPLITUDE,
1,0.100000
2,-0.111111
3,0.125000
4,-0.142857
5,0.166667
6,-0.200000
7,0.250000
8,-0.333333
9,0.500000
10,-1.000000
11,0.000000
))
```



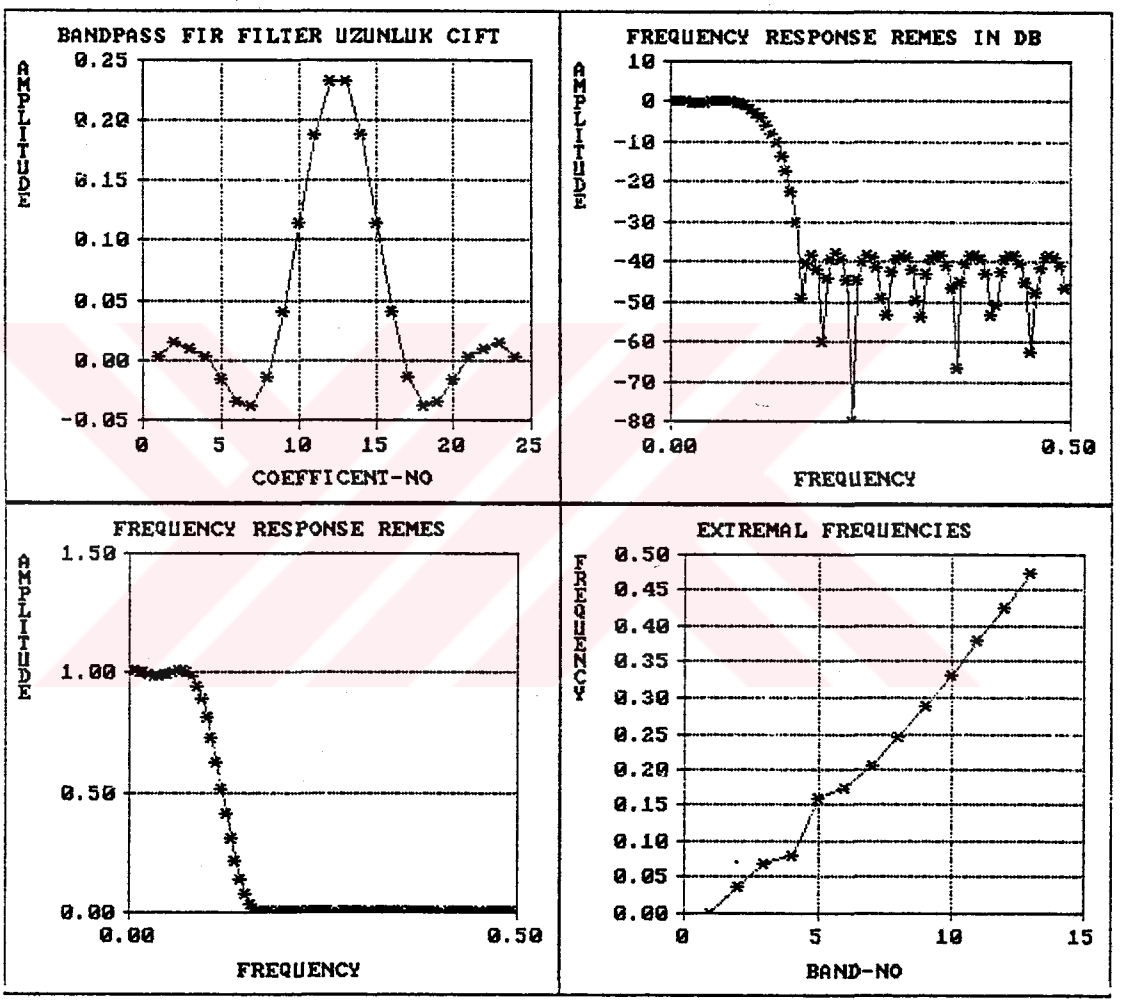
Şekil B.22 N=21 Differentiator'ın katsayıları.



Şekil B.23 N=21 Differentiator'ın frekans cevabı.

```
REMES514.FIR
BANDPASS FIR FILTER UZUNLUK CIFT
COEFFICENT-NO,
AMPLITUDE,
     1,0.003374091
     2,0.014938298
     3,0.010569358
     4,0.002541507
     5,-0.015929993
     6,-0.034085342
     7,-0.038112175
     8,-0.014629168
     9,0.040089542
    10,0.115407127
    11,0.188507516
    12,0.233546058
    13,0.233546058
    14,0.188507516
    15,0.115407127
    16,0.040089542
    17,-0.014629168
    18,-0.038112175
    19,-0.034085342
    20,-0.015929993
    21,0.002541507
    22,0.010569358
    23,0.014938298
    24,0.003374091
))
finite impulse response (fir)
linear phase digital filter design
remez exchange algorithm
Bandpass Filter
filter length =    24
lower band edge[    1]=,0.000000000, upper band edge=[    1]=, 0.080000000
lower band edge[    2]=,0.160000000, upper band edge=[    2]=, 0.500000000
Desired value[1]=, 1.000000000
Desired value[2]=, 0.000000000

 Weights
[    1] , 1.000000000
[    2] , 1.000000000

deviations
     1 , 0.012433639
     2 , 0.012433639

deviations in dB
     1 , -38.108035083
     2 , -38.108035083
))
EXTREMAL FREQUENCIES
BAND-NO,
```

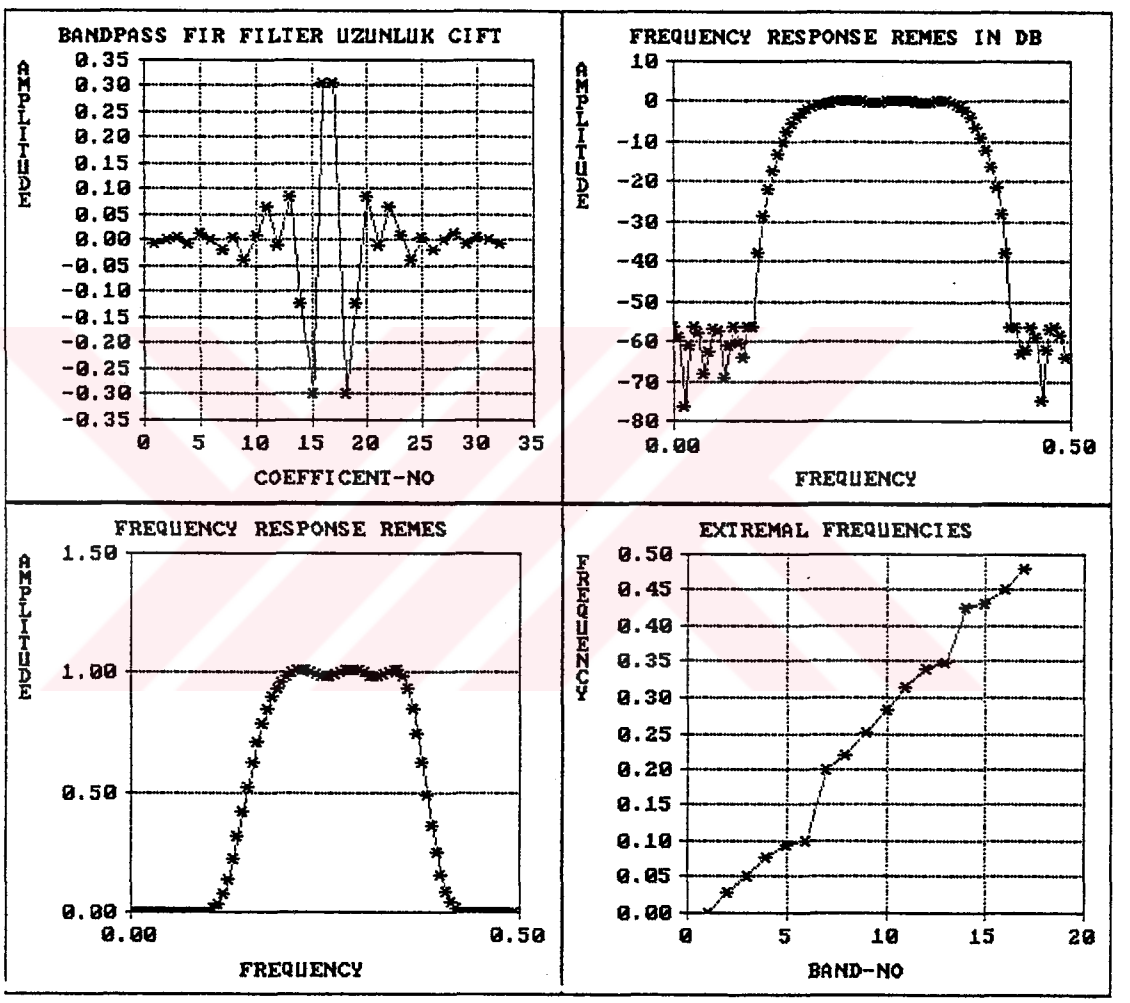
```
FREQUENCY,
    1 , 0.000000000
    2 , 0.036458333
    3 , 0.067708333
    4 , 0.080000000
    5 , 0.160000000
    6 , 0.173020833
    7 , 0.206875000
    8 , 0.245937500
    9 , 0.287604167
   10 , 0.331875000
   11 , 0.378750000
   12 , 0.425625000
   13 , 0.475104167
))
```



Şekil B.24 N=24 Alçakgeçiren filtre. (Parks-McCellan)

```
REMES515.FIR
BANDPASS FIR FILTER UZUNLUK CIFT
COEFFICENT-NO,
AMPLITUDE,
     1,-0.005753412
     2,0.000990273
     3,0.007573354
     4,-0.006514120
     5,0.013960526
     6,0.002295146
     7,-0.019994065
     8,0.007136958
     9,-0.039657363
    10,0.011260113
    11,0.066233640
    12,-0.010497221
    13,0.085136137
    14,-0.120249928
    15,-0.296785766
    16,0.304109170
    17,0.304109170
    18,-0.296785766
    19,-0.120249928
    20,0.085136137
    21,-0.010497221
    22,0.066233640
    23,0.011260113
    24,-0.039657363
    25,0.007136958
    26,-0.019994065
    27,0.002295146
    28,0.013960526
    29,-0.006514120
    30,0.007573354
    31,0.000990273
    32,-0.005753412
))
finite impulse response (fir)
linear phase digital filter design
remez exchange algorithm
Bandpass Filter
filter length =    32
lower band edge[    1]=,0.000000000, upper band edge=[    1]=, 0.100000000
lower band edge[    2]=,0.200000000, upper band edge=[    2]=, 0.350000000
lower band edge[    3]=,0.425000000, upper band edge=[    3]=, 0.500000000
Desired value[1]=, 0.000000000
Desired value[2]=, 1.000000000
Desired value[3]=, 0.000000000

 Weights
[    1] , 10.000000000
[    2] , 1.000000000
[    3] , 10.000000000
```

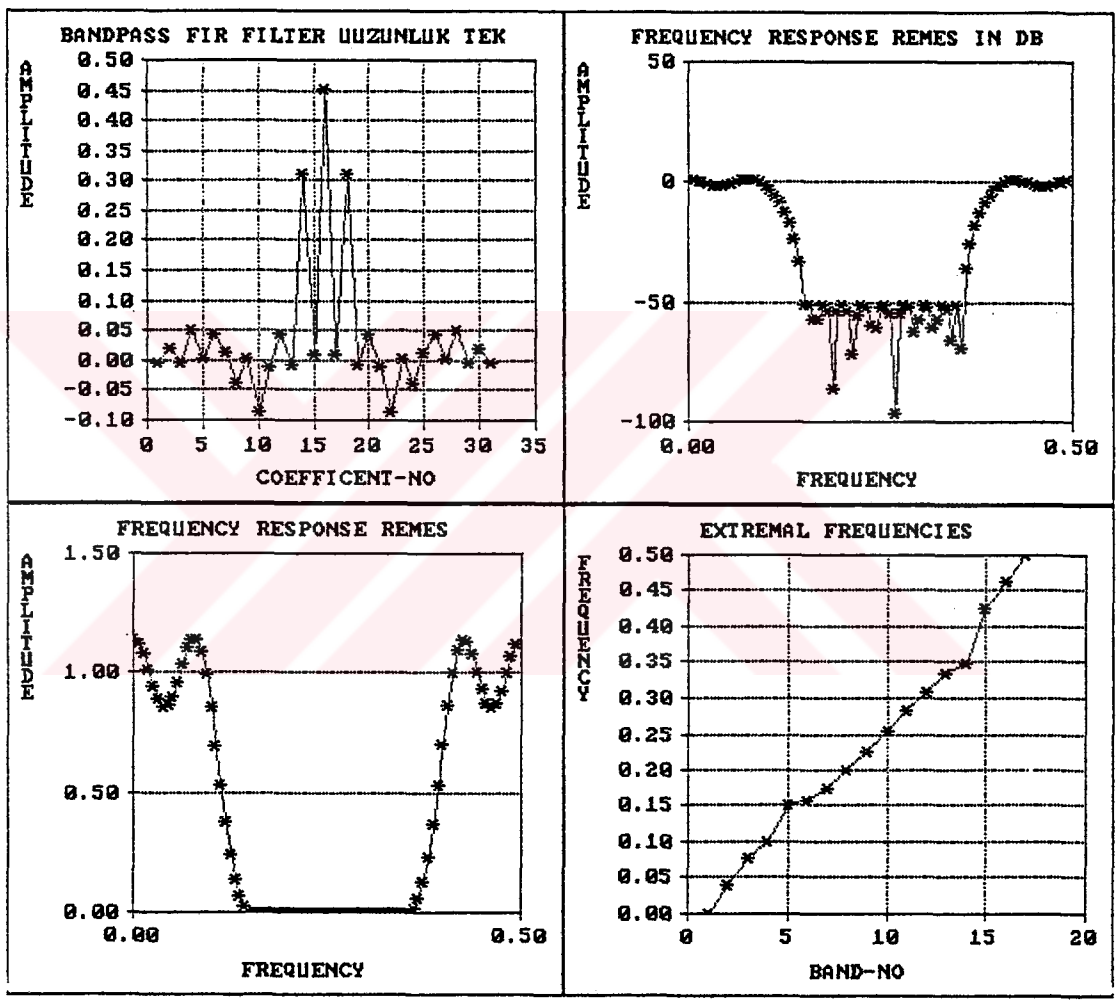
```
deviations
     1 , 0.001513118
     2 , 0.015131176
     3 , 0.001513118

deviations in dB
     1 , -56.402546362
     2 , -36.402546362
     3 , -56.402546362
))
EXTREMAL FREQUENCIES
BAND-NO,
FREQUENCY,
    1 , 0.000000000
    2 , 0.027343750
    3 , 0.052734375
    4 , 0.076171875
    5 , 0.093750000
    6 , 0.100000000
    7 , 0.200000000
    8 , 0.219531250
    9 , 0.252734375
   10 , 0.283984375
   11 , 0.313281250
   12 , 0.338671875
   13 , 0.350000000
   14 , 0.425000000
   15 , 0.432812500
   16 , 0.450390625
   17 , 0.479687500
))
```



Şekil B.25 N=32 Bandgeçiren filtre. (Parks-McCellan)

```
REMES517.FIR
BANDPASS FIR FILTER UUZUNLUK TEK
COEFFICENT-NO,
AMPLITUDE,
     1,-0.004372580
     2,0.019295933
     3,-0.005698289
     4,0.052360281
     5,0.003155024
     6,0.043481228
     7,0.011696225
     8,-0.037915417
     9,0.003484416
    10,-0.087599028
    11,-0.010993060
    12,0.044455164
    13,-0.006934717
    14,0.311448245
    15,0.009662981
    16,0.452967337
    17,0.009662981
    18,0.311448245
    19,-0.006934717
    20,0.044455164
    21,-0.010993060
    22,-0.087599028
    23,0.003484416
    24,-0.037915417
    25,0.011696225
    26,0.043481228
    27,0.003155024
    28,0.052360281
    29,-0.005698289
    30,0.019295933
    31,-0.004372580
))
finite impulse response (fir)
linear phase digital filter design
remez exchange algorithm
Bandpass Filter
filter length =    31
lower band edge[    1]=,0.000000000, upper band edge=[    1]=, 0.100000000
lower band edge[    2]=,0.150000000, upper band edge=[    2]=, 0.350000000
lower band edge[    3]=,0.420000000, upper band edge=[    3]=, 0.500000000
Desired value[1]=, 1.000000000
Desired value[2]=, 0.000000000
Desired value[3]=, 1.000000000

 Weights
[    1] , 1.000000000
[    2] , 50.000000000
[    3] , 1.000000000
```

```
deviations
     1 , 0.144020150
     2 , 0.002880403
     3 , 0.144020150

deviations in dB
     1 , -16.831534819
     2 , -50.810934906
     3 , -16.831534819
))
EXTREMAL FREQUENCIES
BAND-NO,
FREQUENCY,
    1 , 0.000000000
    2 , 0.039062500
    3 , 0.078125000
    4 , 0.100000000
    5 , 0.150000000
    6 , 0.157812500
    7 , 0.175390625
    8 , 0.200781250
    9 , 0.226171875
   10 , 0.255468750
   11 , 0.282812500
   12 , 0.308203125
   13 , 0.333593750
   14 , 0.350000000
   15 , 0.425859375
   16 , 0.462968750
   17 , 0.500000000
))
```

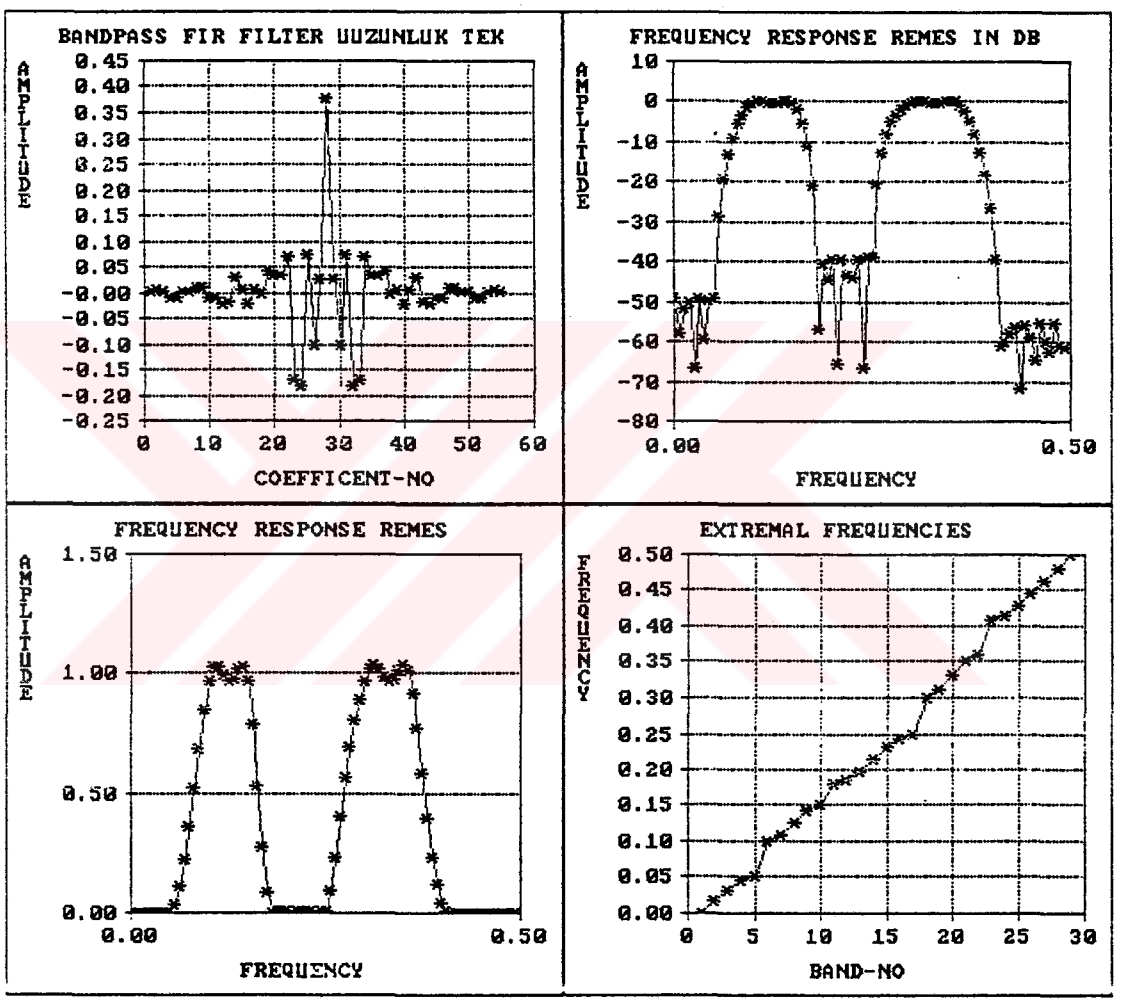


Şekil B.26 N=31 Bandsöndüren filtre. (Parks-McCellan).

```
REMES518.FIR
BANDPASS FIR FILTER UUZUNLUK TEK
COEFFICENT-NO,
AMPLITUDE,
     1,0.001066267
     2,0.006377761
     3,0.003575560
     4,-0.009067788
     5,-0.009090699
     6,0.002915567
     7,0.003963797
     8,0.011172049
     9,0.011646758
    10,-0.009963080
    11,-0.009238423
    12,-0.020406392
    13,-0.019460481
    14,0.031243017
    15,0.006304552
    16,-0.020482807
    17,0.006574053
    18,-0.001120211
    19,0.041956986
    20,0.035784268
    21,0.034744804
    22,0.071496355
    23,-0.171388314
    24,-0.182550435
    25,0.074059027
    26,-0.103174214
    27,0.025716719
    28,0.378135465
    29,0.025716719
    30,-0.103174214
    31,0.074059027
    32,-0.182550435
    33,-0.171388314
    34,0.071496355
    35,0.034744804
    36,0.035784268
    37,0.041956986
    38,-0.001120211
    39,0.006574053
    40,-0.020482807
    41,0.006304552
    42,0.031243017
    43,-0.019460481
    44,-0.020406392
    45,-0.009238423
    46,-0.009963080
    47,0.011646758
    48,0.011172049
    49,0.003963797
```

```
    50,0.002915567
    51,-0.009090699
    52,-0.009067788
    53,0.003575560
    54,0.006377761
    55,0.001066267
))
finite impulse response (fir)
linear phase digital filter design
remez exchange algorithm
Bandpass Filter
filter length =    55
lower band edge[    1]=,0.000000000, upper band edge=[    1]=, 0.050000000
lower band edge[    2]=,0.100000000, upper band edge=[    2]=, 0.150000000
lower band edge[    3]=,0.180000000, upper band edge=[    3]=, 0.250000000
lower band edge[    4]=,0.300000000, upper band edge=[    4]=, 0.360000000
lower band edge[    5]=,0.410000000, upper band edge=[    5]=, 0.500000000
Desired value[1]=, 0.000000000
Desired value[2]=, 1.000000000
Desired value[3]=, 0.000000000
Desired value[4]=, 1.000000000
Desired value[5]=, 0.000000000

 Weights
[    1] , 10.000000000
[    2] , 1.000000000
[    3] , 3.000000000
[    4] , 1.000000000
[    5] , 20.000000000

deviations
     1 , 0.003444859
     2 , 0.034448592
     3 , 0.011482864
     4 , 0.034448592
     5 , 0.001722430

deviations in dB
     1 , -49.256570538
     2 , -29.256570538
     3 , -38.798995632
     4 , -29.256570538
     5 , -55.277170451
))
EXTREMAL FREQUENCIES
BAND-NO,
FREQUENCY,
    1 , 0.000000000
    2 , 0.016741071
    3 , 0.032366071
    4 , 0.044642857
    5 , 0.050000000
```

```
     6 , 0.100000000
     7 , 0.108928571
     8 , 0.126785714
     9 , 0.142410714
    10 , 0.150000000
    11 , 0.180000000
    12 , 0.185580357
    13 , 0.197857143
    14 , 0.213482143
    15 , 0.230223214
    16 , 0.243616071
    17 ,
         0.250000000
    18 , 0.300000000
    19 , 0.312276786
    20 , 0.332366071
    21 , 0.350223214
    22 , 0.360000000
    23 , 0.410000000
    24 , 0.415580357
    25 , 0.428973214
    26 , 0.445714286
    27 , 0.463571429
    28 , 0.481428571
    29 , 0.500000000
))
```

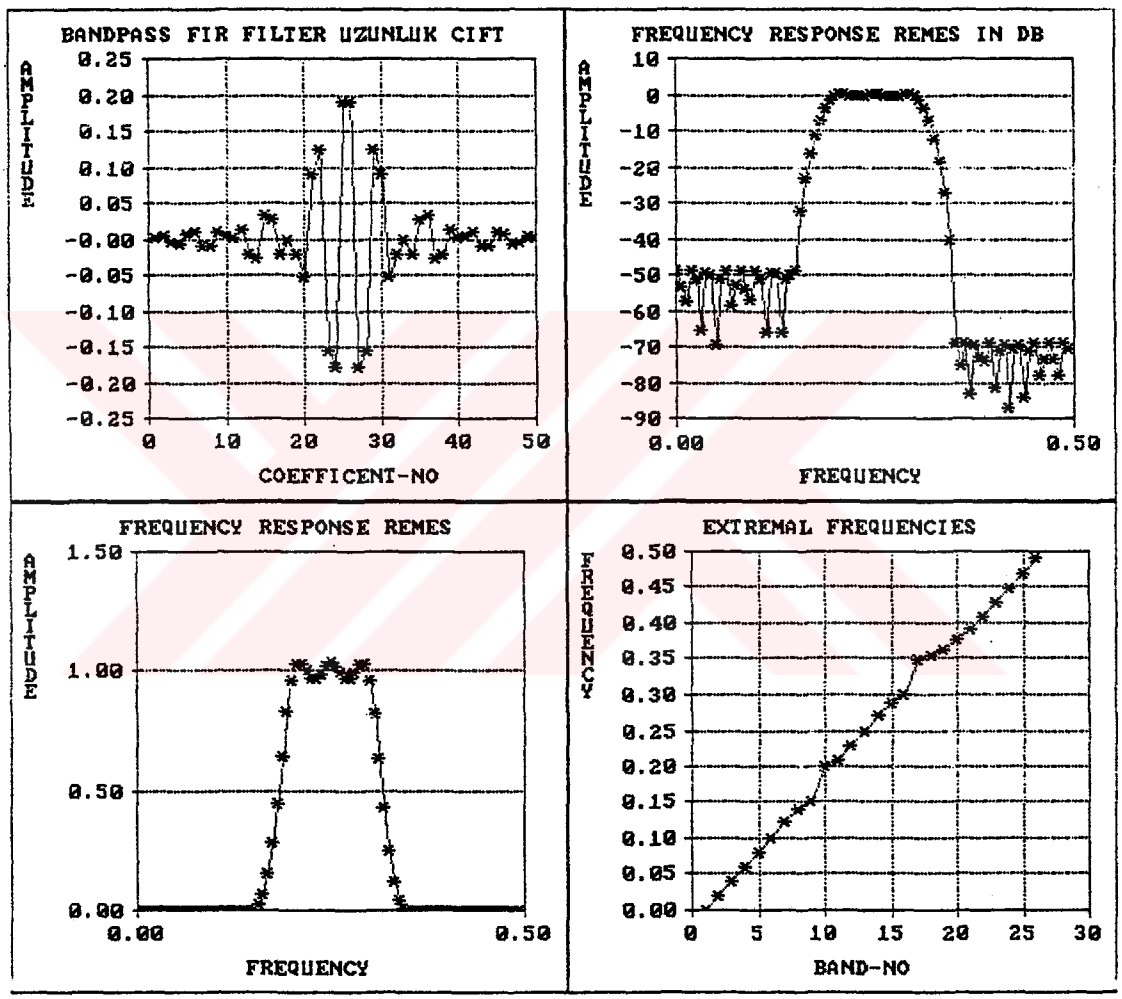


Şekil B.27 N=55 Multiband filtre. (Parks-McCellan).

```
remes516.fir
BANDPASS FIR FILTER UZUNLUK CIFT
COEFFICENT-NO,
AMPLITUDE,
     1,0.001564841
     2,0.003081630
     3,-0.003174526
     4,-0.006198003
     5,0.007435068
     6,0.009836896
     7,-0.011103734
     8,-0.010101927
     9,0.008994919
    10,0.002898019
    11,0.002663300
    12,0.012021958
    13,-0.020657141
    14,-0.027189007
    15,0.032337126
    16,0.028305612
    17,-0.020922036
    18,-0.001876113
    19,-0.022823360
    20,-0.053926219
    21,0.090472538
    22,0.123157720
    23,-0.156392205
    24,-0.177334476
    25,0.190781642
    26,0.190781642
    27,-0.177334476
    28,-0.156392205
    29,0.123157720
    30,0.090472538
    31,-0.053926219
    32,-0.022823360
    33,-0.001876113
    34,-0.020922036
    35,0.028305612
    36,0.032337126
    37,-0.027189007
    38,-0.020657141
    39,0.012021958
    40,0.002663300
    41,0.002898019
    42,0.008994919
    43,-0.010101927
    44,-0.011103734
    45,0.009836896
    46,0.007435068
    47,-0.006198003
    48,-0.003174526
    49,0.003081630
```

```
    50,0.001564841
))
finite impulse response (fir)
linear phase digital filter design
remez exchange algorithm
Bandpass Filter
filter length =    50
lower band edge[    1]=,0.000000000, upper band edge=[    1]=, 0.150000000
lower band edge[    2]=,0.200000000, upper band edge=[    2]=, 0.300000000
lower band edge[    3]=,0.350000000, upper band edge=[    3]=, 0.500000000
Desired value[1]=, 0.000000000
Desired value[2]=, 1.000000000
Desired value[3]=, 0.000000000

 Weights
[    1] , 10.000000000
[    2] , 1.000000000
[    3] , 100.000000000

deviations
     1 , 0.003705049
     2 , 0.037050487
     3 , 0.000370505

deviations in dB
     1 , -48.624121639
     2 , -28.624121639
     3 , -68.624121639
))
EXTREMAL FREQUENCIES
BAND-NO,
FREQUENCY,
    1 , 0.000000000
    2 , 0.020000000
    3 , 0.040000000
    4 , 0.061250000
    5 , 0.081250000
    6 , 0.101250000
    7 , 0.122500000
    8 , 0.141250000
    9 , 0.150000000
   10 , 0.200000000
   11 , 0.210000000
   12 , 0.228750000
   13 , 0.250000000
   14 , 0.271250000
   15 , 0.290000000
   16 , 0.300000000
   17 , 0.350000000
   18 , 0.353750000
   19 , 0.363750000
   20 , 0.376250000
```

```
   21 , 0.392500000
   22 , 0.410000000
   23 , 0.428750000
   24 , 0.448750000
   25 , 0.468750000
   26 , 0.490000000
))
```



Şekil B.28 N=50 Bandgeçiren filtre. (Parks-McCellan).

ÖZGEÇMİŞ

Metin Kalaycı 26 kasım 1966 da İstanbulda doğdu. Cengizhan ilk ve ortaokulundan sonra Yeni Levent Lisesini 1983 yılında bitirdi. Aynı yıl lisans öğrenimine İTÜ Kontrol ve Bilgisayar Mühendisliği bölümünde başladı. 1988 kış yarıyılında lisans ögrenimini bitirdi.

1988 yılında İ.T.Ü Fen Bilimleri Enstitüsü Kontrol ve Bilgisayar Yüksek lisans programına başladı.

İki yıl Login Bilgisayar Programları A.Ş.'de çalişti. Halen U.B.M. A.Ş.'de çalışmaktadır.