YILDIZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

14400 b/s HIZLI BİR MODEMİN HATA BAŞARIMININ İNCELENMESİ

YÜKSEK LİSANS TEZİ

MÜH. HAKAN GÖKDAN

İSTANBUL 1993

ÖNSÖZ

Hızla gelişen teknolojiye ayak uydurabilmek için günümüz insanının büyük bir çaba harcaması gerekmektedir. Zaman içerisinde bir çok alanda ortaya çıkan gelişmelerden diğer alanlarda da yararlanılmaktadır.

Yarı iletken teknolojisi ve bilgi işleme alanındaki hızlı gelişmeler, diğer bir çok alanda olduğu gibi haberleşme alanında da etkisini göstermiştir. Bu gelişmelerin sonucunda, daha küçük boyutlarda ve daha fazla işlem yapabilme yeteneklerine sahip haberleşme aygıtları insanlığın hizmetine sunulmuştur.

Sayısal haberleşme alanında başdöndürücü bir hızla ortaya çıkan gelişmeler, bu alan içerisinde önemli bir yer tutan modemlerin yüksek hızlarda çalışabilmelerini ve daha fazla işlevsel yeteneklere sahip olabilmelerini münkün hale getirmiştir. Başlangıç aşamasında faz ve frekans modülasyonu tekniğini kullanan ve düşük hızlarda çalışabilen modemlerin yerini, yeni kodlama teknikleri ile birlikte daha yüksek hızlarda çalışabilen ve daha yüksek hata başarımına sahip modemler almıştır.

Çalışmanın başlaması esnasında beni yönlendiren sayın Prof. Dr. Metin YÜCEL 'e, çalışmanın yürütülmesinde büyük katkıları olan tez hocam sayın Doç. Dr. H. Ümit AYGÖLÜ 'ne ve yardımlarından dolayı sayın Doç. Dr. A. Hamdi KAYRAN 'a teşekkürü borç bilirim. Ayrıca yüksek lisans öğrenimim boyunca emeği geçen hocalarımın hepsine teşekkür ederim.

Hakan GÖKDAN

İÇİNDEKİLER

ÖZET

Bu çalışmada, kafes kodlamalı modülasyon tekniğini kullanan CCITT V.33 modemin yapısı araştırılmış ve bu modemin kodlayıcı ve kod çözücü bölümlerini içeren bir haberleşme sisteminin bilgisayar benzetimi yardımıyla hata başarımları çeşitli etmenler altında incelenmiştir.

Çalışmanın birinci bölümünde, veri haberleşme sistemlerinin başlıca kısımları ve bu kısımlar içerisinde iletim sistemi olarak çok önemli yer tutan telefon şebekesi üzerinde veri iletişiminin yapılabilmesi için gerekli aygıtlar olan modemler üzerinde durulmuştur.

İkinci bölümde, modemlerin tiplerini, çalışma şekillerini ve üzerinde çalışabilecekleri kanal tiplerini içeren genel bilgiler verilmiştir. Ayrıca, CCITT V.33 modemin yapısı ve özellikleri de anlatılmıştır.

Üçüncü bölümde, kafes kodlamalı modülasyonun yapısı ve eşdeğer kodlamasız sistemlere göre başarım üstünlükleri incelenmiştir. Ayrıca, 90, 180, 270 derecelik dönmeyle değişmeyen işaret uzayına sahip geri beslemeli kodlar için kurallar ve tasarım ölçütleri belirtilmiştir.

Dördüncü bölümde, ideal olmayan analog kanallar üzerinde oluşan simgelerarası girişim olayı incelenmiş ve bu olayın giderilmesi için kullanılan dengeleme stratejilerinden söz edilmiştir. Dengeleme işlemi için kullanılan uyarlamalı dengeleyicilerin yapıları ele alınmıştır. Uyarlama algoritmaları içerisinde, bu çalışmada kullanılan en küçük karesel ortalama hata metodu tanıtılmıştır.

Beşinci bölümde, telefon kanalları üzerinde empedans süreksizlikleri nedeniyle ortaya çıkan yankı giderme problemi ele alınmıştır. İki telli devreler üzerinde tam çift yönlü veri iletiminin yapılabilmesi için gerekli olan değişik yankı giderici gerçeklemeleri üzerinde durulmuştur.

Altıncı bölümde, eşzamanlı veri iletiminde önemli yer tutan zamanlama problemi kısaca açıklanmıştır. Eşzamanlı iletim yapabilen modemler içerisinde, zamanlama bilgisinin ve taşıyıcı işaretinin çıkarılmasında kullanılan devrelerin yapısı hakkında da kısaca bilgi verilmiştir.

Yedinci bölümde, vericisinde 128-CR işaret uzayına sahip kafes kodlayıcı bulunan, alıcı kısmında ise kod çözme tekniği olarak Viterbi kod çözücüsünü kullanan bir sistemin, toplamsal Gauss gürültülü ideal kanal üzerindeki davranışı bilgisayar benzetimi yardımıyla değişik işaret gürültü oranları altında incelenmiş ve hata başarım eğrileri çıkarılmıştır. İletim ortamına gönderilen işaret elemanlarının, işaret uzayı içerisinde kanalın etkileri nedeniyle belirli açılarla dönmeleri durumunda sistem başarımı incelenmiştir. Daha sonra sisteme bozucu kanal eklenmiş ve getirdiği bozucu etkiyi gidermek için, bir uyarlamalı dengeleyici alıcı girişine yerleştirilmiştir. Bu durumda dengeleyici çıkışındaki ortalama hatanın iterasyon sayısına göre değişim eğrileri çıkarılmıştır. Dengeleyicinin davranışı, yakınsama süresini ve hata sınırını belirleyen değişik parametrelere göre incelenmiştir.

Sekizinci bölümde, benzetim sonuçları değerlendirilmiş ve sonuçlar yorumlanmıştır.

SUMMARY

In this study, the structure of a CCITT V.33 (14.4 kbps) modem using trellis coded modulation technique is investigated and the error performance of a communication system which concern the encoder and the decoder structure of this recommendation is analysed by computer simulations.

In the first chapter of this study, the essential components of the data communication systems and modems are discussed.

Chapter two describes the types and operating modes of the modems and gives some general information about the transmission channels on which modems can work. It also identifies the structure and specifications of the CCITT V.33 recommendation.

Chapter three is concerned with the trellis coded modulation (TCM) technique and the performance of TCM schemes compared to the other uncoded modulation schemes. The design rules and procedures for 90, 180, 270 degree rotationally invariant convolutional codes with expanded signal spaces are defined.

In the fourth chapter, intersymbol interferance phenomenon occuring on non-ideal noisy channels and the equalization strategies used in order to overcome its effects are mentioned. Also, the structures of the adaptive equalizers are identified. The least mean square algorithm which tries to minimize the mean-squared error at the output of the equalizer is outlined.

Chapter five builds on the echo cancelling problem arising from the impedance discontinuities on the telephone channels. Several echo canceller configurations are introduced.

Chapter seven describes the synchronization problem for digital communications. Also, it briefly describes the circuits which are used for timing extraction and carrier recovery in the modem systems.

In the seventh chapter, the performance of a simulation model which is composed of a transmitter using a trellis coded modulation scheme with 128-CR expanded signal space and a receiver with Viterbi decoder is studied on the ideal transmission channel with additive Gaussian noise for several signal - to - noise ratio values. Also, the error - event probability of the system as a function of signal - to - noise ratio is obtained. After that, a disturbing channel is added to the communication system and by using an adaptive equalizer this disturbing effect is considerably eliminated. Then, for disturbing channel, the change of the mean - squared error at the output of the equalizer with iteration number is studied. Performance of the equalizer according to the parameters which control error limit and convergence time of the adaptation algorithm is covered. Error event probability of the system with disturbing channel is also derived.

In the last chapter, the results are discussed and some observations are presented.

BÖLÜM 1. GİRİŞ

Telekomünikasyon endüstrisindeki güncel eğilimi ve gelecekteki eğilimlerin yönünü anlayabilmek için temel bazı ilkelerin anlaşılması son derece önemlidir.

Basit bir telekomünikasyon şebekesinin yapısını anlamak başlangıç aşamasında yapılacak en önemli çalışmadır. Telekomünikasyon şebekesi, iletim, anahtarlama, işaretleşme ve uç aygıtlarından oluşan bir düzendir.

Çoğu zaman, bir çok kollara ve bölgesel işlevlere sahip olan organizasyonlar, sık bir şekilde veri gönderme ve alma ihtiyacı duyarlar. Bu durumda, geniş komponent ve servis çeşitliliğine sahip bir veri haberleşme sistemini kullanabilirler. Seçilen kombinasyon, çok değişik şebekelerin sınırlama ve avantajlarına bağlı olacaktır. Kodlanmış bilginin bir noktadan diğer bir yere elektronik iletimi, değişik fiziksel elemanları, sistemleri, araçları ve standartları gerekli kılar. Bu tip şebekeler, dağıtık şebekeler(distrubuted networks) olarak adlandırılırlar.

İşte bu türden sayısal haberleşme bağlantılarının kurulmasında, telefon şebekesi veri iletim ortamı olarak önemli yer tutar. Telefon şebekesi veri iletimine uygun olmamasına rağmen çok geniş bir ağ oluşturduğu için oldukça ekonomiktir. Bu nedenle, bu son derece geniş şebeke üzerinde değişik hızlarda veri iletimini mümkün kılan modemler geliştirilmiştir.

Bu çalışmada 128-CR işaret uzayına sahip ve kafes kodlamalı modülasyon tekniğini kullanan CCITT V.33 modemin yapısı araştırılmış ve bu modemi içeren bir iletişim sisteminin bilgisayar benzetimi yoluyla hata başarımları incelenmiştir.

Başlangıçta faz ve frekans modülasyonu ilkesine göre tasarlanan ve bu nedenle düşük hızlarda çalışan modemlerin yerini günümüzde, daha yüksek hata başarımlarına sahip modemler almıştır.

İlk kez Ungerboeck [1], band genişliği sınırlı olan kanallar üzerinde bağımsız simge iletimi gerçekleştiren işaretleşme metodlarının, yüksek veri hızlarında yeterli olmadığını göstermiştir. Ayrıca Ungerboeck, iyileştirilmiş sistem başarımı için gerekli olan çok seviyeli kafes kodlarının gerçekleştirilmesini araştırmıştır. Bu araştırmalar sonucu kafes kodlamalı modülasyon tekniği günümüzde veri haberleşme sistemlerinde yerini almıştır. Bu metodda kodlama ve modülasyon işlemleri bir bütündür. İşaret uzayının genişletilmesi ile elde edilen fazlalık, katlamalı kodlayıcılarda olduğu gibi [2], mümkün olan simge dizileri arasındaki serbest Öklid uzaklığını arttırmak için kullanılmaktadır.

Alıcıda Viterbi kod çözücüsü [4] kullanılarak, kod çözme işleminde kafes kodunun özel yapısının da avantajları kullanılmaktadır. Bu şekilde kafes kodlu modülasyon işaretleri işlenmekte ve gelen işaret dizisine karar verilmektedir. Kodlayıcı yapısındaki durum geçişleri, modülasyonlu kanal işaretleri cinsinden belirlendiği için başarım ölçütü olarak Hamming uzaklığı yerine Öklid uzaklığı kullanılmaktadır.

Ungerboeck [1], [2], toplamsal Gauss gürültüsü altında, eşdeğer kodlamasız sistemlere göre 3-6 dB lik kodlama kazancının kafes yapıları ile sağlanabileceğini göstermiştir. Bu türden kodlama

teknikleri kullanılarak, aynı band genişliğinde daha yüksek hızlarda veri iletişiminin yapılması mümkündür.

Lee-Fang Wei [5], 90, 180 ve 270 derecelik faz dönmeleri ile değişmez olan kodlar tasarlamıştır. Böylece, bit dizilerinin simge elemanlarına uygun bir şekilde eşlenmesi ile bu tür belirsizlikler ortadan kaldırılmıştır.

Bu yapıdaki kodlama teknikleri ile aynı band genişliği içerisinde daha yüksek hızlarda veri iletimi yapılabilmesi olanaklı hale gelmiştir. CCITT tavsiyelerinde [6] bu türden yüksek hızlı modemlerin özellikleri belirtilmiştir.

Bununla birlikte iletim hızının artmasıyla, kullanılan kanalın neden olduğu bozucu etkiler sağlıklı iletimi engelleyebilir. Bu durumda, yüksek hızlarda simgelerarası girişimi engellemek için modemlerde uyarlamalı dengeleyicilerin [7], [8], [9] kullanılması son derece gereklidir. Üzerinde iletim yapılacak olan kanalın karakteristiklerinin tam olarak bilinememesi, katsayıları belirli bir yakınsama ölçüsünde ayarlanabilen uyarlamalı dengeleyicilerin geliştirilmesine neden olmuştur.

Telefon şebekesi üzerinde yapılan veri iletiminde diğer bir sınırlamada yankı giderme (echo cancelation) [10], [11], [12] problemidir. Telefon şebekesi, işaret anahtarlama istasyonları nedeniyle üretilen düşük güçlü işaret yansımalarını gidermek için yankı gidericileri kullanmaktadır. Ancak ses haberleşmesi için iyi ve temiz bağlantıların kurulmasını sağlayan ve yöne bağlı olan bu süzgeçler, yarı çift yönlü (half dublex) çalışmada veri iletimini sınırlamaktadırlar. Bu nedenle, veri iletimi için modemler üzerinde kullanılmak üzere değişik yankı gidericiler ve yankı giderme metodları geliştirilmiştir. Yankı giderme ve dengeleme işlemlerinin düşük haberleşme hızlarında kolay işlemler olmalarına rağmen, haberleşme hızı arttıkça çözülmesi daha zor bir problem olarak ortaya çıkmaktadırlar.

Eşzamanlı sayısal haberleşmenin yapıldığı yerlerde ortaya çıkan diğer bir olayda zamanlama [13] problemidir ve sistemin hata başarımı üzerinde büyük etkiye sahiptir. Bu problem karşımıza taşıyıcı, saat bilgisi ve sözcük zamanlaması zorunluluğu olarak çıkar. Modemlerin içerisinde kullanılan faz kilitlemeli çevrim devreleri, gelen veri içerisinden zamanlama bilgisinin çıkarılması ve taşıyıcı işaretinin yeniden elde edilmesi işlemlerini yerine getirmektedir.

BÖLÜM 2 . MODEMLER VE ÖZELLİKLERİ

Telefon şebekesinin çok geniş bir ağ olması, ses haberleşmesinin yanısıra veri haberleşmesi içinde kullanılmasını gerekli kılmaktadır. Fakat bu şebeke üzerinden sayısal işaretler doğrudan, sağlıklı bir şekilde iletilemez. Bunun nedeni, iki haberleşme noktası arasındaki mesafenin artmasının hat direnci, endüktansı ve kapasitesi nedeniyle sayısal işaretler üzerinde bozulmaya neden olmasıdır. Belirli bir uzaklıktan sonra, sayısal işaret darbeleri alıcı tarafından anlaşılamayacak hale gelmektedir. Telefon şebekesi yalnızca 300-3400 Hz arasındaki frekans bandını geçirecek, doğru akımı iletmeyecek ve grup gecikmeleri nedeniyle darbe şekilllerini bozacaktır. Haberleşme taşıyıcıları, telefon hatları üzerinde analog işaretleri yeniden oluşturup güçlendirmek için yükselticileri kullanırlar. Yükselticilerin, ses bilgilerini dünyanın her tarafına taşımalarına rağmen, sayısal bilgi üzerinde kullanılmaları sakıncalıdır. Çünkü, yükselticiler gürültü de dahil olmak üzere her işareti yeniden üretmektedir. Böylece, sayısal bir darbeye karşı gelen gürültü de yükseltici tarafından kuvvetlendirilecektir. Frekans bandının üstten sınırlaması da, darbelerin kenarlarının yuvarlatılmasına, çınlamalara ve hatta işaret frekansı band dışında kalıyorsa işaretin hiç iletilememesine yol açacaktır. Sayısal bilgi işlem aygıtları tarafından üretilen sayısal işaretler ile analog telefon hatları arasındaki bu uyumsuzluk, bu işaretlerin analog iletim ortamlarında taşınmasını mümkün kılacak bir dönüştürücü elemanı gerekli kılmaktadır. Yani sayısal işaretlerin, ses frekanslı işaretler şekline dönüştürülmeleri gereklidir. Bu dönüştürücü araçlar, isimleri modülator-demodülatör terimlerinin kısaltılmış birleşiminden oluşan modemlerdir.

Modemler iki taraflı olup, bir tarafı sayısal işaret kaynağına diğer tarafı ise telefon hattına bağlıdır. Modemin modülator kısmı, sayısal işaret kaynakları tarafından üretilen işaretleri, telefon hatları üzerinden iletebilmek için analog işaretler haline dönüştürür. Demodülatör kısmı ise iletilen analog işaretleri alır ve tekrar sayısal işaretler haline dönüştürür. Bu nedenle modülatör, haberleşme sisteminin iletim birimi ve demodülatör ise alış birimi olarak düşünülebilir. Şekil 2.1'de bir modemin verici kısmının, şekil 2.2'de ise alıcı kısmının blok diyagramı görülmektedir.

Modemler değişik özelliklerine göre sınıflandırılabilir. Bu özellikler, zeka içerme, fabrikasyon yöntemi, uyumluluk, modülasyon ve demodülasyon işlemlerinin uygulanabildiği veri tipi ve üzerinde çalışacakları kanal işleyiş modları olarak adlandırılabilir.

1970'li yıllara kadar modemler zeka içermeyen aygıtlar olarak sınıflandırılmaktaydı. Bunun nedeni, modemlerin kendilerine bağlı olan operatörünün isteklerine bağlı olan birtakım değişik işlevleri destekleyememeleriydi. Mikroişlemci teknolojisinin gelişimi ve bu modemlerin içine RAM (random-access memory), ROM (read-only memory), ve EPROM (erasable-programable-read-only memory) gibi bellek elemanlarının eklenmesi ile, modemlere bu özellik kazandırılmıştır. ROM birimi içerisine kazınmış olan hazır programlar sayesinde, komutları tanıma ve bu komutlar üzerine işlem yapma yetenekleri ile modemler, otomatik arama, otomatik cevap verme, uzak modem ile kurulacak

bağlantıda kullanılacak modülasyon türünün otomatik olarak belirlenmesi, hata belirleme ve düzeltme, haberleşme şebekesinin yönetimi ve bunun gibi birçok fonksiyonu yerine getirebilmektedirler.

Günümüzde modemler üç değişik gerçekleştirme türünde üretilmektedir. Masa üstü, (table-top modem), raf tipi (rack mounted modem) ve kartlar üzerine (card modem) yerleştirilmiş olan modemler bu üç türü temsil etmektedirler.

Hatta bağlanmış olan bir modem santral tarafından bakıldığı zaman bir telefon aygıtının özelliklerine sahip olmalıdır. Bu özellikler şu şekilde sıralanabilir :

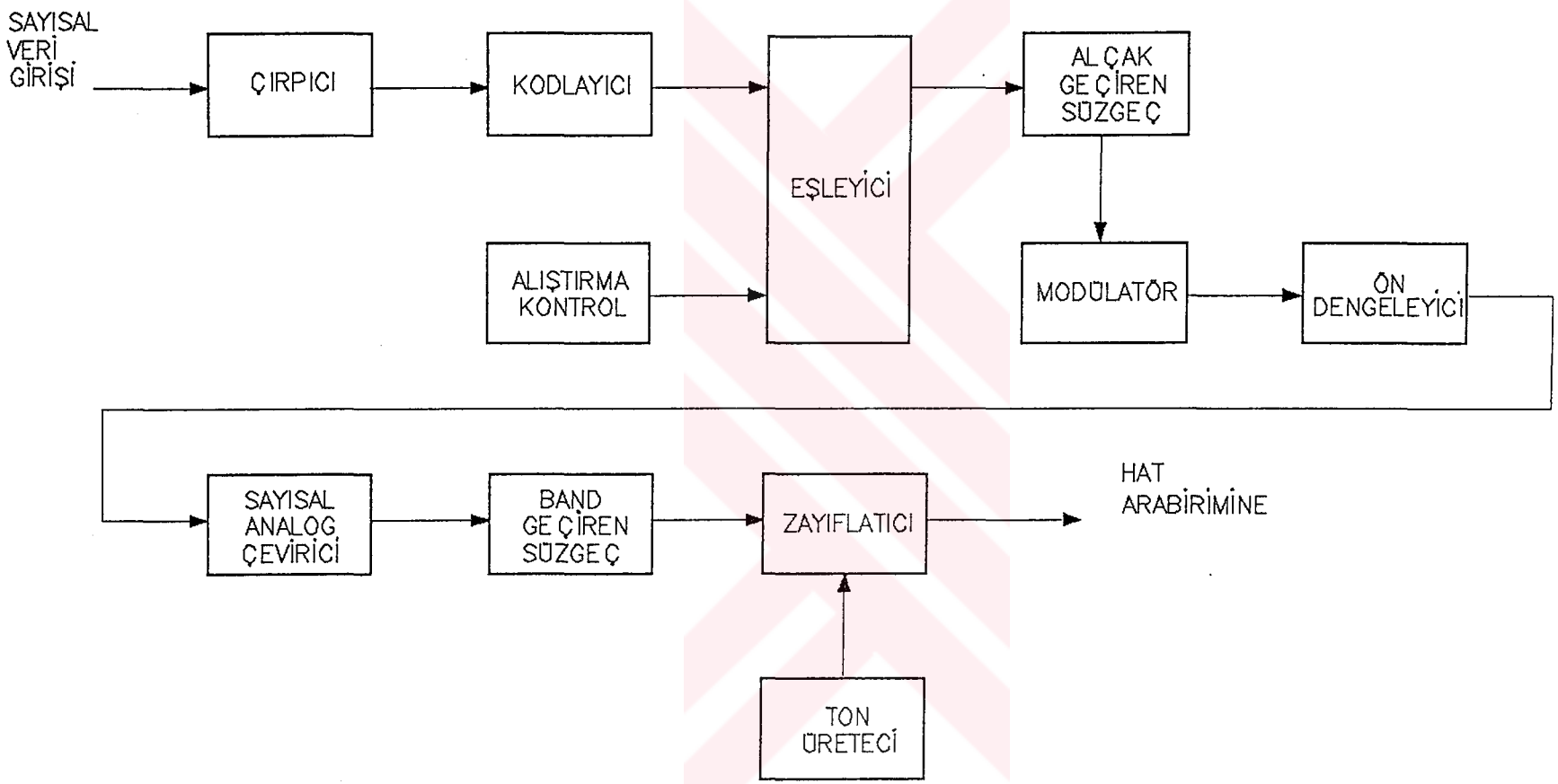
1. Modem üzerinden hat akımı akabilmelidir. Bu özellik santralin hattı tutabilmesi için gereklidir.
2. Hatta gösterilen empedans standart bir değerde olmalıdır.
3. Modem, hattan gelebilecek aşırı gerilim ve parazitlere karşı yeterince korunmuş olmalıdır.
4. Modem, tipine bağlı olarak iki veya dört telli çalışabilmelidir.

Hat akımı, modemin hat ucunda bulunan bir hat transformatörü üzerinden akıtılmaktadır. Analog arabirim üzerinde bulunan bir röle yardımıyla bu transformatör hatta bağlanmakta ve aynı röle yardımıyla modem telefon aygıtını hatta bağlayıp, ayırabilmektedir.

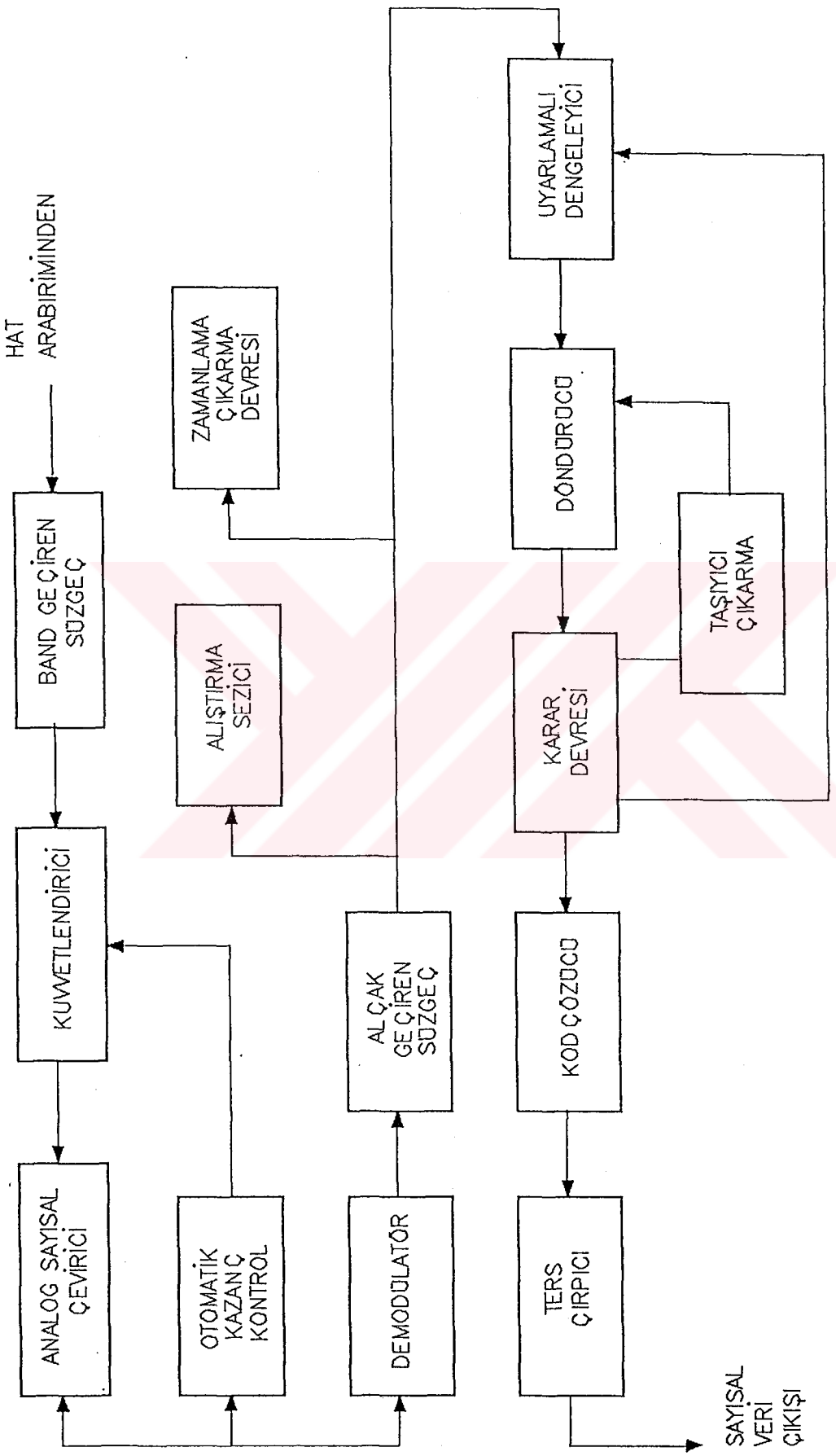
Modem üzerinde analog arabirimden başka, kendisine bağlanan sayısal işaret kaynağıyla haberleşmesini sağlayan bir sayısal arabirim de mevcuttur. Bu arabirim, bilgi alışveriş uçları yanısıra birtakım kontrol işaretlerinide kapsamaktadır. Bu tipteki veri iletişim arabirimlerini ve işaretleşme türlerini tanımlayan birçok standartlar ve tavsiyeler mevcuttur. Aslında veri haberleşmesinin temel probleminin, sayısal bilginin bir seri arabirim ve iletim kanalı üzerinden anlamını yitirmeden iletilmesi olduğu söylenebilir.

Bir çok arabirim arasında en önemli olanı, bilgiyi gönderip alabilen veri uç elemanı ile (data terminal equipment-DTE) bilgiyi bir yerden diğer bir yere aktaran veri haberleşme elemanı (data communication equipment-DCE) arasında olan arabirimdir. CCITT V24 ve V28 tavsiyeleri modem aygıtları üzerinde bu türden bağlantılar için gerekli olan bilgileri kapsamaktadır. Kullanılan bağlantı elemanı genelde 25 uçlu D tipi bağlayıcı olmakla birlikte daha değişik tiplerde ve standartlarda (V35,G703 vb.) bağlantı elemanı kullanan modemlerde vardır. RS 232-C, RS 422, RS 423, RS 449, X21 gibi terimler, bilgisayarların, modemlerin, şebekelerin birbirine bağlanmasında kullanılan arabirimlerin elektriksel ve mekanik özelliklerini belirleyen standartlardır. Modem üretecileri değişik uygulamalara göre üretim yapmaktadırlar. Uygulama türleri, noktadan noktaya (point-to-point), çok noktalı haberleşme şebekeleri (multipoint commünications networks) ve anahtarlamalı telefon şebekeleri (switched telephone networks) olarak sayılabilir. Bu uygulamalara göre üretilen modemler başlıca iki grup içerisine girmektedir.

Ses bandı modemleri (voice band modems) ve genişband modemleri (broadband modems). Ses bandı modemleri 19.2 kbps hızına kadar ve 3 khz 'lik band genişliği içerisinde çalışabilirler. Geniş bandlı modemler ise, 19.2 kbps hızından daha yukarı hızlarda çalışabilirler ve daha yüksek bir band genişliğine gereksinim duyarlar.



Şekil 2.1 Modem verici kısmı blok diyagramı

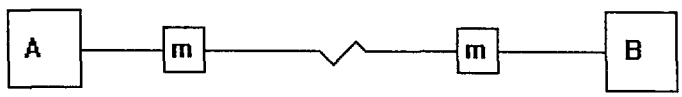


Şekil 2.2 Modem alıcı kısmı blok diyagramı

2.1 Kanal Tipleri Ve Kanal İşleyiş Modları

1. Noktadan noktaya (point-to-point) kanallar :

Noktadan noktaya kanallar, sadece iki istasyon arasında kurulan kanallardır. Bu tür kanallar, özel bir hat veya anahtarlamalı şebeke üzerinden kurulur. Eğer özel bir hat şeklinde kurulmuşlarsa, sadece bu hatta bağlanan iki istasyon tarafından kullanılırlar ve daimidirler. Bu hatlar iki veya dört telli olabililir.



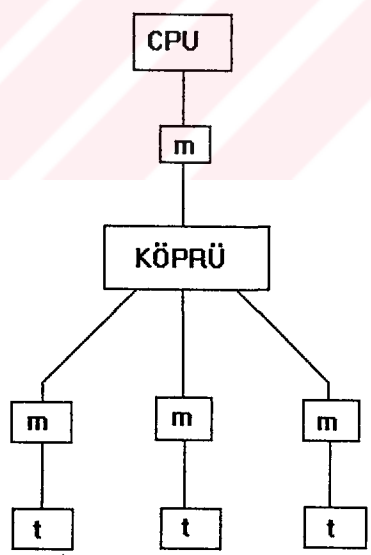
Noktadan noktaya kanal

Şekil 2.1.1

2. Çok noktalı (multipoint) kanallar:

Çok nokta tanımı, aynı zamanda ikiden fazla istasyon arasında oluşturulan bir komünikasyon bağlantısını ifade etmekte kullanılır.

3. Çok düşmeli (multi-drop) hatlar:



Şekil 2.1.2

Kanal İşleyiş Modları :

Bir komünikasyon kanalının işletilebileceği üç çalışma modu vardır :

1. Tek yönlü (Simplex)

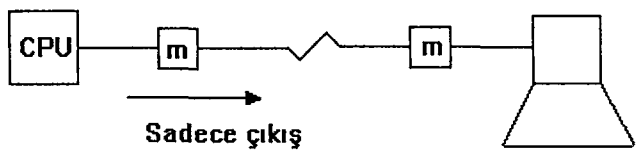
2. Yarı çift yönlü (Half-dublex)

3. Tam çift yönlü (Full -dublex)

Kanalın çalıştırılacağı mod, bu kanalı kullanacak olan aygıtın ihtiyaç ve kapasitesine bağlı olarak belirlenir. Bu aynı zamanda, kanalın iki veya dört telli olması halini de belirler.

1. Tek yönlü çalışma modu (simplex) :

Tek yönlü çalışma modu, en az gelişmiş ve belkide en az kullanılan bir çalışma modudur. Bu modda, biri sadece bilgi çıkışı yapabilen, diğeri ise sadece gönderilen bilgiyi alabilen iki istasyona gerek vardır. Bu mod için iki telli bir devreye gerek duyulacaktır.

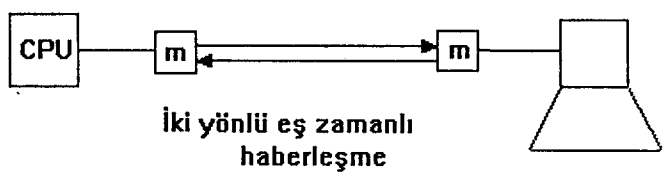


Tek yönlü haberleşme

Şekil 2.1.3

2. Yarı çift yönlü çalışma modu (Half-dublex) :

En yaygın olan çalışma modudur. Bu modda komünikasyon iki yönde de ilerleyebilir. Fakat aynı zamanda gerçekleştirilemez. Bu çalışma modunda iki telli bir devreye ihtiyaç duyulur. Fakat genelde dört telli devreler kullanırlar. İkinci durum, biri sadece giriş diğeri ise sadece çıkış olarak kullanılan, iki ayrı tek yönlü çalışma modundaki devre ile gerçekleştirilir..



Yarı çift yönlü haberleşme

Şekil 2.1.3

3. Tam çift yönlü çalışma modu (Full-dublex) :

Tam çift yönlü çalışma modu en gelişmiş çalışma modu olup, bu modda komünikasyon iki yönde ve aynı anda ilerleyebilir. Bu metod zamanın en etkin biçimde kullanımını sağlar. Tam çift yönlü çalışma için normalde dört telli bir devreye ihtiyaç duyulur. Bu çalışma modunun iki telli devreler üzerinden yapılması, özel olarak gerçekleştirilen modemlerle mümkündür. Bu türden bir çalışmada, kullanılan kanalın iletim bandı ikiye ayrılmaktadır. Bir yöndeki iletimde frekans bandının bir yarısı,

diğer doğrultudaki iletimde ise diğer yarısı kullanılır. Bu noktada, ters kanal işaretleşmesinden bahsetmek yerinde olacaktır.

Ters kanal işaretleşmesi, iki telli devreler üzerinde, yarı çift yönlü çalışma modunda kullanılabilen faydalı bir özellik olarak karşımıza çıkmaktadır. Bu durumda, band genişliğinin küçük bir kısmı, veri akışının ters yönünde özel bir fekansın gönderilmesi işlemi için ayrılır. Bu özel frekans, işaretleşme için kullanılır. Bu tür bir kullanıma örnek olarak, uzun bir veri bloğunun A istasyonundan B istasyonuna iletildiğini düşünelim. Veri alış işlemi sırasında, B istasyonunun bir hata algıladığını kabul edelim. İki telli yarı çift yönlü çalışmada, B istasyonu A ya hata olduğuna dair bilgiyi veri transferi tamamlanıncaya kadar gönderemez. Eğer ters kanal kullanılıyorsa, B istasyonu bu band genişliğini, A istasyonuna hatayı hemen bildirmede kullanabilir.

2.2 Genel İşaretleşme Yapısı

Veri haberleşme şebekeleri, haberleşen uç aygıtları arasında sağlıklı bir bilgi alışverişinin yapılabilmesi için, kendilerine ait bir takım kurallara sahip olmalıdır. Veri şebekeleri genişledikçe haberleşme protokolleri olarak alandırılan bu kurallar büyük önem kazanmaktadır. En basit anlamda bir haberleşme protokolü, çok noktalı veya anahtarlamalı kanallar üzerinde, verici ve alıcı uç haberleşme aygıtlarını belirleyen bir yapıya sahip olmalı ve buna ilave olarak gönderilen bilginin formatını, başlangıç ve bitiş özelliklerini de tayin etmelidir. Karmaşık bir haberleşme sistemi içerisinde, yerine getirilmesi gereken değişik işlevlerin gruplara ayrılabilmesi amacıyla protokol hiyerarşisi oluşturulmuştur. Modemlerin terminal tarafındaki sayısal arabirimin fiziksel ve elektriksel yapısını tanımlayan standartlar, bu protokol hiyerarşisi içerisinde birinci seviyede yer almaktadır. EIA (The Electronic Industries Association) organizasyonu, birinci seviye içerisinde yer alan sayısal arabirimin elektriksel ve fiziksel özelliklerini belirleyen bir standart olarak RS 232-C arabirimini oluşturmuştur. Temelde RS 232-C üç noktayı belirlemektedir.

a. Gerilim ve işaret seviyeleri uyumlu olmalıdır.

b. Arabirim bağlayıcıları birbirine uygun olmalıdır. Yani, aynı pin bağlantılarına sahip olmalıdır.

c. Bir haberleşme aygıtı tarafından sağlanan kontrol bilgisi, diğer bir aygıt tarafından anlaşılabilmelidir.

RS 232-C arabirimi 25 pinli D tipi bağlantı yapısına sahiptir. Birçok sistem bu pinlerin çoğunu kullanmakta, bazı sistemler ise yapılan bağlantı türüne göre bir bölümünü kullanmaktadırlar. Ancak, bütün sistemlerin aynı pinleri kullanması, RS 232-C arabirimi ile birbirine bağlı olan her sistem arasında sağlıklı bir bağlantının kurulacağı anlamına gelmez. Aynı zamanda, bu pinler üzerinde doğru işaret şekillerinin de oluşması gereklidir. Zamanlama işaretleri ve ihtiyaç duyulan kontrol işaretlerinin de uygun formatta olması gereklidir.

Tablo 2.2.1 de, modem-terminal arası RS 232-C sayısal arabirimi üzerindeki en çok kullanılan bazı pinlerin isimleri ve işlevleri belirtilmiştir.

RS 232-C arabiriminin işaret yapısı da göz önüne alınarak, tipik bir haberleşme bağlantısı aşağıdaki gibi kurulur :

1. Modem otomatik arama özelliğine sahip değilse, hattına paralel bağlı bir telefon yardımıyla istenilen numara aranılarak bağlantı kurulur. Karşı modemin cevap tonu alınır alınmaz, modem üzerinde veri iletim moduna geçme tuşuna basılarak, modem içerisindeki hat rölesi çekilir ve modem telefonu hattan ayırarak kendisi devreye girer. Telefon kapatılır.

2. Modeme bağlı olan terminal aktif durumda olmalıdır. Terminal kendisinin aktif durumda olduğunu CCITT 108 numaralı DTR devresini mantıksal "1" seviyesine çekerek modeme iletir. Bu durumda modem de bilgi alışverişi için hazır ve aktif durumda ise DSR devresini mantıksal "1" seviyesine çeker.

3. Terminal bilgi gönderme isteğine sahipse, CCITT 105 numaralı RTS devresini aktif ederek bunu modeme bildirir. Modem ise 106 numaralı CTS devresini aktif hale getirerek, terminale veri göndermeye hazır olduğunu bildirir. Bu arada değişik kanal çalışma modlarına ve taşıyıcı kontrol metodlarına bağlı olarak değişen bir gecikme mevcuttur. RTS-CTS işaretleri arasındaki gecikme süresi içerisinde modemler arasında el sıkışma (handshaking) işlemi başlar. Bu işlem esnasında modemler arasında, kullanılacak olan modülasyon türü ve haberleşme hızı gibi birçok parametre üzerinde anlaşma sağlanır.

4. Bu andan sonra terminal, 103 numaralı TxD ucundan sayısal bilgiyi gönderir.

5. Hattan bilgi alınıyorsa ve hat üzerinde taşıyıcı işareti varsa, modemler bu durumu 109 numaralı CD devrelerini aktif hale çekerek belli ederler. Hat üzerinde herhangi bir nedenle taşıyıcı işareti kesilirse, CD devresi tekrar mantıksal "0" seviyesine düşer. Bu andan sonra 104 numaralı RxD devresinden alınan işaretler terminal tarafından göz önüne alınmaz.

Başlangıç Dizisi (Turn-On Sequance)

Başlangıç dizisi iletimin başında, modem tarafından diğer modemin alıcısına, gelen işaret ile eşzamanlamayı sağlamak için gönderilir. Alıcı bu işarete göre, kendi uyarlamalı dengeleyicisi, otomatik kazanç kontrolü ve veri iletimi için gerekli olan diğer parametreleri kurar. Bu dizi aynı zamanda, veri iletimi sırasında modemlerden bir tanesi eşzamanlamayı sağlayamadığında ve yeniden bağlantıyı kurma durumuna (retraining) düştüğünde de kullanılır. Eşzamanlamayı sağlayamayan modem bir başlangıç dizisi yollar. Diğer modem aynı diziyi yollayarak yanıt verir. Böylece, eşzamanlamayı kaybeden modem, gelen bu yeni diziden yararlanarak alıcısını yeniden diğer modem ile eşzamanlı bir hale getirebilir. Bu işlem "Round-Robin" olarak adlandırılır ve sadece noktadan noktaya kanal tipleri için kullanılır. Çok noktalı devrelerde, yöneten modem çeşitli yönetilen modemlerden gelen ve bilgi almaya hazır olduklarını belirten her işaretin (poll) başlangıcında, başlangıç dizisi yardımıyla eşzamanlamayı sağlar. Eğer modem bu işaretinin gelişi sırasında eşzamanlamayı kaybederse işaret kaybolur ve yönetilen modem tarafından yeniden gönderilmesi gereklidir.Eğer yönetilen modemlerden biri eşzamanlamayı

kaybederse, bahsedilen yeniden eşzamanlama metodlarından hiçbirini kullanamaz. Bu bütün sistemin çökmesine neden olabilir.

İşaret İsmi	Kısaltma	Pin No	CCITT No	Kullanım
Koruyucu Toprak		1	101	Terminalin metal gövdesine bağlantıyı sağlar.
Gönderilen Veri	TDATA	2	103	Terminalin veri gönderme yolu
Alınan Veri	RDATA	3	104	Terminalin veri alış yolu
Gönderme İsteği	RTS	4	105	Terminal bilgi gönderme isteğini modeme belirtir.
Veri transferi için modem hazır	CTS	5	106	Modem bilgi almaya veya göndermeye hazır olduğunu belirtir.
Modem Hazır	DSR	6	107	Modem aktif olduğunu belirtir.
Hat taşıyıcı sezicisi	CD	8	109	Modem taşıyıcıyı aldığını belirtir.
Terminal Hazır	DTR	20	108	Terminal aktif olduğunu belirtir.

Tablo 2.2.1

Taşıyıcı Kontrolü (Carrier Control) :

1. Sürekli taşıyıcı (Continuous carrier)

Sürekli taşıyıcı, noktadan noktaya kanallar için çok genel bir bağlantı türüdür. Tam çift yönlü çalışma modlarında kullanılır. Bu tür bağlantılarda hat işaretini kontrol etmeye gerek yoktur. Bu durumda modem DTE 'den gelen RTS devresini kullanmaz ve CTS devresi sürekli olarak aktif durumdadır(mümkün olan bir başlangıç dizisinin iletim anı dışında). Eğer DTE RTS-CTS cevabı isterse, bunun için bir alternatif vardır. Bu alternatifde, modem hat üzerinde sürekli bir taşıyıcıya sahiptir fakat CTS devresi, RTS devresini izler. Bu konumda, RTS-CTS gecikmesi belirli bir sürede

olacaktır(Örneğin 1 veya 10 ms olabilir). Şekil 2.2.1 'de, sürekli taşıyıcı durumunda ileten ve alan modemlerin sayısal arabirimleri üzerindeki işaretlerin durumları görülmektedir.

2. Kontrollü taşıyıcı (Controlled carrier)

Kontrollü taşıyıcı, yarı çift yönlü çalışmada kullanılır. Çünkü bu tür çalışma modunda hat işaretinin kontrol edilmesi gerekmektedir.

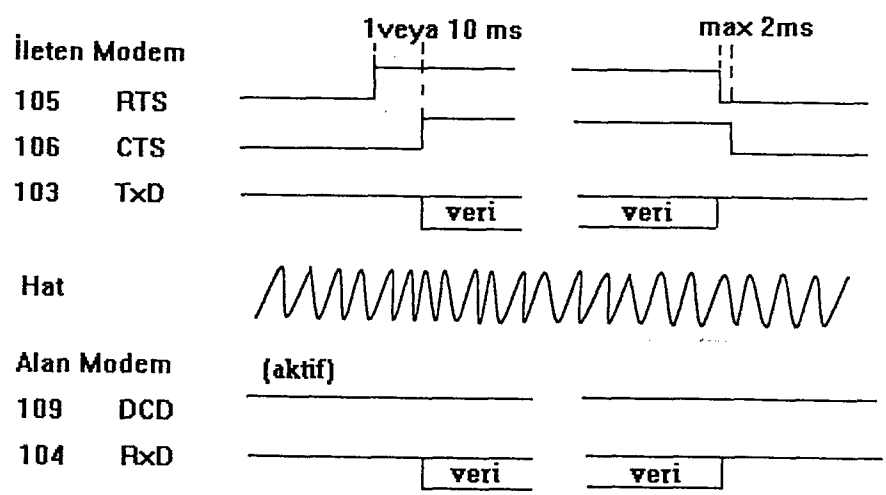
RTS devresi iletimi kontrol eder. Bu devre aktif durumda değil ise herhangi bir taşıyıcı işaretinin iletimi söz konusu değildir. RTS devresi aktif duruma geçtiğinde modem ilk önce bir başlangıç dizisi yollar. Daha sonra, CTS devresini aktif duruma getirdiğinde veri iletimi başlar. Aşağıda CCITT V.33 ve V.29 tavsiyesi modemler için başlangıç dizilerinin süreleri verilmiştir.

V.33 1393 ms V.29 253 ms

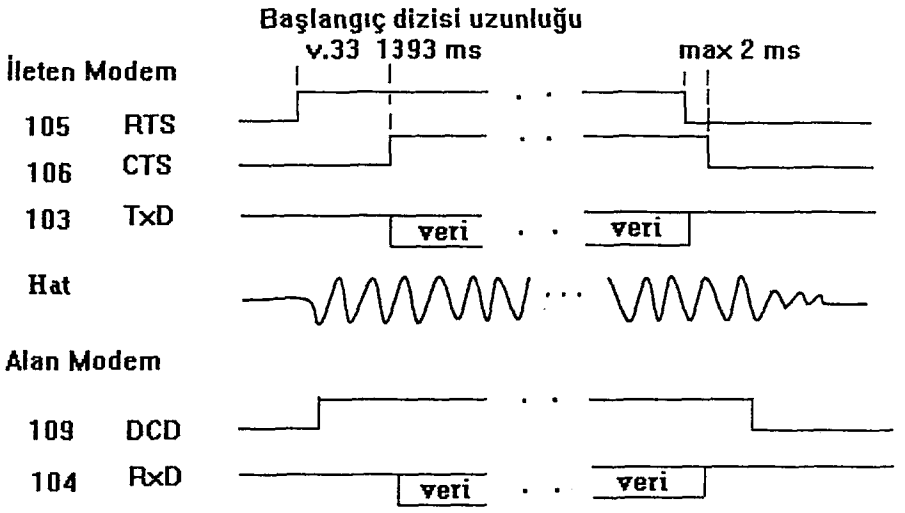
Şekil 2.2.2 de, kontrollü taşıyıcı durumunda işaret durumları görülmektedir.

3. Benzetimli taşıyıcı (Simulated carrier)

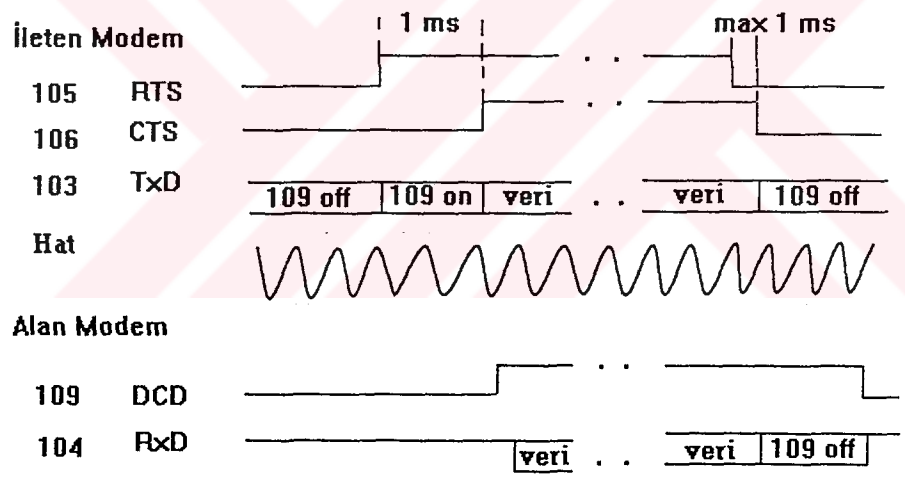
Benzetimli taşıyıcı yarı çift yönlü bağlantılar üzerinde ve gerçek hat işaretini kontrol etmenin gerekli olmadığı yerlerde kullanılır. Bu kontrollü taşıyıcıdan daha verimli bir yoldur. Çünkü, RTS-CTS devreleri arasındaki gecikme çok daha kısadır. Benzetimli taşıyıcıda, modemlerin RTS devrelerinin durumu iletilen veriyle birlikte, diğer modemin DCD devresine belirli bit dizileri kullanılarak iletilir. Bu yolla hat üzerinde daima bir taşıyıcı vardır. Ancak, RTS devresi aktif değil iken modem, diğer modemin DCD devresini aktif olmayan duruma getirmek için bir bit dizisi yollar. RTS devresinin aktif olmayan durumdan aktif duruma geçişinde, modem kısa bir DCD aktif işareti yollar (veri iletiminden önce) ve CTS devresini aktif duruma sokar. Bu durumda RTS-CTS devreleri arasındaki gecikme, modem tarafından yollanan DCD aktif işaretinin hat üzerinden diğer modeme ulaşım süresi kadardır. Şekil 2.2.3 de, benzetimli taşıyıcı durumunda ileten ve alan modemlerin sayısal arabirimleri üzerindeki işaretlerin durumları görülmektedir.



Şekil 2.2.1



Şekil 2.2.2



Şekil 2.2.3

Verici zamanlaması (Transmitter timing) :

Modemin verici devresi içerisinde değişik zamanlama alternatifleri kullanılabilir. Bunlar şu şekilde sıralanabilir :

1. Modemin dahili zamanlaması

Gönderilen verinin zamanlaması (103, TxD) modemin CCITT 114 (TxC) numaralı devresinden gelir.

2. Vericinin alınan zamanlama işareti ile eşzamanlı olarak çalışması

Gönderilen verinin zamanlama işareti modemin 114 numaralı devresinden gelir. Modemin verici zamanlama bilgisi, alıcı devresinin zamanlamasını izler (CCITT 115 numaralı RxC devresi). Böylece verici, uzak uçtaki modem ile eşzamanlamayı sağlamış olur.

3. DTE 'den elde edilen zamanlama (CCITT 113 numaralı devre)

Gönderilen verinin zamanlama bilgisi, DTE 113 devresinden (TCx) gelir.

2.3 Eşzamanlı Ve Eşzamanlı Olmayan Modemlerin Genel Yapıları

Veri normalde iki DTE arasında sabit bir büyüklüğün katları şeklindeki uzunluklarda iletilmektedir. Örneğin, bir terminal diğer bir terminal ile haberleşirken, terminal üzerindeki klavyede basılan her karakter sekiz bitten oluşan sayısal bir diziye dönüştürülmektedir. Bu durumda bütün mesaj aynı şekilde kodlanmış bloklardan oluşacaktır. Her karakter seri olarak iletildiği için, alıcı tarafta bulunan DTE, bit dizisine bağlı olarak değişen iki mantıksal seviyeden birini alacaktır. Bu durumda alıcı taraftaki haberleşme aygıtının, gelen bu bit dizisini doğru olarak algılayabilmesi için aşağıdaki noktaları bilmesi gerekmektedir.

1. Kullanılan sayısal darbe hızı (Bit rate),
2. Her simge veya karakterin başlangıç ve bitiş zamanı,
3. Bütün mesaj bloğunun başlangıç ve bitiş zamanı.

Yukarıda bahsedilen üç faktör sırasıyla bit, karakter (byte) ve çerçeve (frame) eşzamanlaması olarak sayılabilir.

Genelde eşzamanlama olayı iki şekilde gerçekleştirilir. Kullanılan metod , alıcı ve verici zamanlamasının birbirinden bağımsız veya eşzamanlı olmasına bağlıdır. Eğer gönderilecek olan veri, her karakter arasında rastgele bir zaman aralığı bulunan karakter dizilerinden oluşuyorsa, bu durumda her karakter bağımsız olarak yollanır. Alıcı her yeni alınan karakterin başında verici ile yeniden eşzamanlı hale gelir. Bu tipte bir iletim için eşzamansız iletim (asynchronous transmission) kullanılır. Eğer gönderilecek olan veri, bir çok karakterden oluşan çerçeve dizilerinden oluşuyorsa, bu durumda verici ve alıcı zamanlamalarının birbirleriyle aynı olması gerekmektedir. Bu yüzden eşzamanlı iletim (synchronous transmission) kullanılır.

Eşzamansız iletimde, alıcının her yeni alınan karakterin başında yeniden eşzamanlı bir hale gelmesi, her gönderilen karakterin ilave başlangıç ve bitiş darbeleri içerisinde saklanması ile mümkündür. Fakat önceden hazırlanmış büyük veri bloklarının iletimi söz konusu olduğu zaman, her karakter başına ilave çerçeveleme darbelerinin konulması iletimin etkinliğini azaltmaktadır.

Bu durumda eşzamanlı iletim kullanılarak bütün veri bloğu tek bir darbe dizisi olarak, karakterleri oluşturan her sekiz darbe arasında hiç bir gecikme olmadan iletilmektedir. Alıcı aygıtın çeşitli seviyelerdeki eşzamanlamayı sağlayabilmesi için aşağıdaki noktalar sağlanmalıdır.

1. İletilen bit dizisi uygun şekilde kodlanmalıdır.

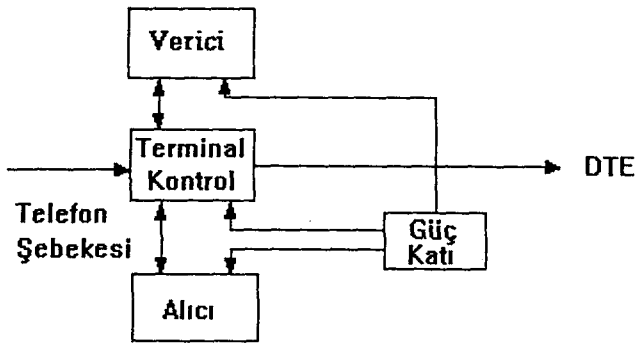
2. Alıcının alınan bit dizisini doğru karakter sınırında algılayabilmesi gerekmektedir.

3. Her çerçevenin içeriği bir çift karakter tarafından sınırlandırılmalıdır.

Düşük hızlarda çalışan modemler, genelde eşzamansız iletim modunu kullanırlar. Düşük hızlı eşzamansız modemlerde, ses bandı iki alt banda bölünerek modemlerden biri "cevap" (answer) modunda diğeri "başlatma" (originate) modunda ayarlanarak sağlanır. Çağrıyı başlatan modem 0 (space) seviyelerini 1070 Hz ve 1 (mark) seviyelerini 1270 Hz de iletir. Cevap modundaki modem ise, 0 seviyelerini 2025 Hz ve 1 seviyelerini 2225 Hz de iletir. Bu modülasyon tekniği FSK (frequency shift keying) olarak adlandırılır ve ancak düşük hızlı modemlerde kullanılabilir. Yüksek hızlara ulaşılmak istendiğinde bu metod yetersiz kalacaktır. Çünkü yüksek veri hızları daha fazla band genişliği gereksinimini beraberinde getirir. Böylece her alt band için "mark" ve "space" frekansları birbirlerinden daha uzakta olmak zorundadır. Buda bizim ses bandı dışına çıkmamıza neden olabilir.

Birim zaman içerisindeki işaretleşme elemanlarının sayısı arttıkça her işaretleşme elemanının kapsadığı sürede azalmaktadır. Eşzamansız modemlerde, alıcı ve verici birimler için zaman tabanının farklı olması nedeniyle iki zamanlama işareti süresi arasındaki küçük bir fark, verinin farklı zamanlarda örneklenmesine neden olacağı için hataya yol açacaktır. Eşzamanlı modemlerde, zamanlama bilgisinin alınan veri içerisinden çıkarılması sağlanarak bu problemin üstesinden gelinmiştir. Belirli bir band genişliğindeki kanal için maksimum işaretleşme hızı, kanalın band genişliği ve işaret gürültü oranı ile belirlenmektedir. Ses bandı kanalları 3 KHz civarında sabit bir band genişliğine sahip oldukları için sabit bir işaretleşme hızına sahiptirler (2400 baud). Bu durumda yapılacak olan, birden fazla sayısal darbenin paketlenerek bir işaret elemanı içerisinde gönderilmesidir.

Eşzamanlı modemler temelde şekil 2.3.1 de görüldüğü gibi, verici, alıcı, terminal kontrol ve güç katı olmak üzere dört kısımdan oluşmaktadırlar. Şekil 2.1 de görüldüğü gibi, verici kısım temelde zamanlama, çırpıcı, kodlayıcı ve eşleyici, modülatör, sayısal analog çevirici,süzgeç devreleri ile dengeleyici (compromise equalizer) kısımlarından oluşmaktadır.



Şekil 2.3.1

Zamanlama devresi, modem ve terminalin her ikisi içinde temel zamanlama bilgisini sağlamaktadır. Belirli veri devreleri, gönderilen veri için zamanlamanın DTE tarafından sağlanması ihtiyacını göstermektedir. Bu durumlarda modem içerisinde belirli bir opsiyon hazır bulunmaktadır.

Modemin dahili zamanlama birimi, harici zamanlama işaret kaynağına DTE sayısal arabirimi üzerinden kilitlenir. Modemin içerisinde bulunan dahili zamanlama devresi, genelde bir kristal osilatör tarafından kontrol edilir.

Alıcı zamanlama bilgisi alınan veri içerisinden çıkarıldığı için, bu verinin yeterli miktarda mantıksal "0" seviyesinden "1" seviyesine geçişleri kapsaması gerekmektedir. Bu durum, zamanlama çıkarma devresinin eşzamanlı bir durumda kalabilmesi için gereklidir. Prensipte, bir terminal veya bilgi işlem aygıtından çıkan veri rastgele bir sayısal darbe dizisi özelliğine sahiptir. Eğer veri dizisi aynı mantıksal seviyedeki uzun dizileri kapsıyorsa, alıcıya eşzamanlama için gerekli olan yeterli mantıksal seviye geçişlerini sağlamayacaktır.

Verici bu durumu, giriş sayısal işaret dizisini kontrollü bir şekilde değiştirerek önlemelidir. Verici birimin bu işi yapan kısmı çırpıcı devresi olarak adlandırılır. Çırpıcı devreleri, V.33 modemin yapısını anlatan alt bölümdede görüleceği gibi, geribeslemeli kayan saklayıcılar (shift registers) olarak gerçekleştirilirler. Çırpıcılar, mümkün olan her faz açı değişimini gerçekleştiricek ve alıcıya zamanlama işaretini elde edebilmesi için yeterli sayıda faz kaymaları sağlayacak şekilde gerçekleştirilirler.

Yukarıda sayılan nedenlerden dolayı çırpıcı devresinin gerekli olmasına rağmen, bu devre hata oranını arttıracaktır. Çünkü bir bit üzerinde görülecek olan hata, ilgili olduğu diğer birçok biti de etkileyecektir. Bu problemi ortadan kaldırmak için, bazı modemler çırpıcı devre girişindeki veriyi Gray kodunda kodlarlar. Böylece demodülasyon sırasında ortaya çıkan hatalı faz durumunun kodu çözüldüğü zaman, sadece bir bit hatası oluşmuş olacaktır.

Gray kodunun özelliği, peşi sıra gelen değişik bit dizileri arasında sadece bir bitlik değişikliğin olmasıdır.

Kodlayıcıya gelen veri dizisi, farksal ve katlamalı kodlama işleminden geçirilir. Daha sonra eşleyiciye gelen kodlanmış veri dizileri, tabloya bakma yöntemi kullanılarak modülasyon için gerekli olan bileşenler elde edilir. Bu bileşenlere karşı gelen işaretler daha sonra bir alçak geçiren süzgeçten geçirilir. Burada kanalın frekans bandı dışarısındaki frekans bileşenleri süzülür. Alçak geçiren süzgeçten geçirilen işaretler daha sonra belirli bir modülasyon tekniğine göre, modülasyon işleminden geçirilmek üzere modülatör devresine gönderilir.

Verici katında bulunan ön dengeleyici, istatistiksel dengeleyici olarak adlandırılır. Çünkü bu dengeleyici, iletim ortamının ortalama karakteristiklerini önceden kompanze etmek için kullanılır. Dengeleyici, iletim ortamı içerisindeki genlik bozulmalarını ve grup gecikmesi olayını gidermeye çalışır.

Sayısal olarak kodlanmış olan işaretler, ihtiyaç duyulan analog işareti üretmek üzere sayısal - analog dönüştürücüden geçirilir. Son olarak, işaret bir ayarlamalı zayıflatıcıdan geçirilerek hibrit devresi üzerinden kanalı aktarılır.

Alıcı kısımda ise şekil 2.2 de görüldüğü gibi temelde, dengeleyici, zamanlama çıkarma devresi, demodülatör, otomatik kazanç kontrolü, ters çırpıcı, kod çözme devresi, süzgeç devreleri ile dönüştürücü devreler yer almaktadır.

Vericide yer alan dengeleyici birimi, sadece kanal çıkışında oluşacağı tahmin edilen hataları gidereceği için oldukça basit olacaktır. Ancak alıcıda bulunan dengeleyici, iletim ortamında oluşan gerçek hataları gidermek zorundadır. Bu işlem, alınan işaret içerisindeki hataları sezen ve bu hatalara göre katsayılarını değiştiren uyarlamalı süzgeç ile yapılacaktır.

Alıcıda hat üzerinden alınan işaret, dahili zamanlama işareti kullanılarak frekans çevrimine uğratılır. Oluşan ara frekans değeri, verinin alınış hızıyla aynı oranda bir zamanlama işareti üretmek için işlenir. Bu işaret, referans işareti olarak bir faz kilitlemeli çevrim (Phase-locked-loop) osilatörüne uygulanır. Bu osilatörün çıkışı, gelen hat frekansına ve fazına kilitlenmiş kararlı bir işarettir.

Alıcının ters çırpıcı birimi, verici kısımda yer alan çırpıcı devresinin tersi bir işlevi yerine getirmektedir

Eşzamanlı modemlerin alıcılarına hibrit aktarıcısı üzerinden gelen işaret, band geçiren süzgeçten geçirilerek bir ön kuvvetlendiriciye uygulanır. Daha sonra bu işaret, örnekleyici ve tutucu devre yardımıyla yankı giderici devresi çıkışındaki işarettten çıkarılır. Bu şekilde iletim hatları üzerindeki empedans süreksizlikleri ve başka nedenlerle oluşan yankıların etkisi en aza indirilmiş olur. Yankı giderici çıkışındaki işaret aiçak geçiren bir süzgeçten geçirilir. Burada amaç sadece iletim bandı içerisindeki işaretlerin geçirilmesidir. Süzgeç çıkışındaki işaret, kontrollü bir kuvvetlendiriciye uygulanarak seviyesi uygun bir değere getirilir. Seviyesi ayarlanmış işaret bir örnekle-tut devresine uygulanır. Demodülasyon işleminden sonra işaret alıcı süzgeçten geçirilir ve uyarlamalı dengeleyiciye uygulanır. Alıcıdaki kontrollü kuvvetlendirici, bir otomatik kazanç kontrol devresi tarafından demodülatör devresi çıkışındaki işaretin değerlendirilmesiyle çalışmaktadır. Bu işaret aynı zamanda örnekle-tut devresine gerekli olan zamanlama işaretini veren zamanlama çıkarma devresini beslemektedir. Uyarlamalı dengeleyicideki transversal süzgeç katsayıları ile ağırlaştırılan işaret faz kilitlemeli çevrim devresine girmekte ve gerekli faz düzeltmeleri yapılmaktadır. Fazı düzeltilen işaret karar verme devresine girerek, işaret uzayı içerisindeki en yakın değere yuvarlatılır. Bu yeni değer, kod çözücü ve ters çırpıcıdan geçirilerek mesaj işareti elde edilir. Karar verme devresi çıkışında elde edilen işaretler aynı zamanda, dengeleyici devresi katsayılarının güncelleştirilmesinde ve faz kilitlemeli çevrim devresi için kontrol işareti olarak kullanılır.

2.4 Değişik Hızlardaki Modemler İçin CCITT V Serisi Tavsiyeleri

Birçok modem üreticisi kendi ürünlerinin sunduğu olanakları, uyumluluk veya Bell sistemi için Western Electric firmasının ürettiği modemlere eşdeğerlik bakımından açıklarlar. Diğer bir standart da CCITT (Commitee for International Telephone and Telegraphy) tavsiyeleridir. Uluslar arası telekomünikasyon birliğinin, merkezi Geneva kentinde olan bir parçası olan CCITT komitesi, Birleşik Devletler dışında kalan birok ülkenin PTT teşkilatları tarafından aynen kabul edilen bir seri modem standartı geliştirmiştir. Bu alt bölümde daha popüler olması nedeniyle sadece CCITT tavsiyesine sahip orta ve yüksek hızlardaki modemlerin özellikleri belirtilmiştir. CCITT V serisi tavsiyelerine göre, modemler hızlarına ve çalışma modlarına göre şu gruplara ayrılmaktadırlar:

1. Düşük hızlı modemler

0 -300	bps	Tam çift yönlü	(Faz modülasyonu)
600 -1200	bps	Yarı çift yönlü	(")
600 -1200	bps	Tam çift yönlü	(")
1200 /2400	bps	Yarı çift yönlü	(")
1200 /2400	bps	Tam çift yönlü	(")

2. Orta hızlı modemler

2400 /4800	bps	Yarı çift yönlü	(Faz modülasyonu)
4800	bps	Tam çift yönlü	(PM + AM)

3. Yüksek hızlı modemler

9600	bps	Yarı çift yönlü ve tam çift yönlü	(PM + AM)

Orta hızlı modemler :

Bu kısımda incelenecek olan CCITT standartları 4800 bps hızında çalışabilen modemlere ait olan standartlardır.

CCITT V.27 tavsiyesi

CCITT V.27 tavsiyesi 4800 bps hızında, eşzamanlı iletim yapabilen çok genel modemleri temsil etmektedir. V.27 tavsiyesi modemleri kiralık hatlar üzerinde, tam veya yarı çift yönlü modda çalışacak şekilde gerçekleştirilmişlerdir. Bu tavsiyeye sahip modemlerin genel özellileri şunlardır :

1. 4800 bps hızında, iki telli hatlar üzerinde yarı çift yönlü,dört telli hatlar üzerinde tam çift yönlü olarak çalışabilir.
2. 1800 Hz taşıyıcı frekansına sahiptir.
3. 8 fazdan oluşan farksal faz kaymalı anahtarlama (DPSK) modülasyonu kullanır.
4. 4800 bps hızda çalışabilmek ve 1600 baud modülasyon hızında bilgi iletebilmek için her haberleşme aralığında 3 bitin gönderilmesi gereklidir.
5. Manual olarak ayarlanabilen ve hattaki girişimleri gidermek etmek için kullanılan bir istatistiksel dengeleyiciye sahiptir.
6. Her iki haberleşme yönünde de, seçime bağlı olarak 75 bps hızında kontrol işaretleşmesi için ters kanal işaretleşmesi özelliğine sahiptir.

Yüksek hızlı modemler :

Bu kısımda incelenecek olan CCITT standartları, 9600 bps ve daha yüksek hızlarda çalışan modemleri temsil etmektedir. Bunlar içerisinde CCITT V.29 ve V.32 tavsiyeleri 9600 bps hızda çalışan

modemlerin özelliklerini belirlemekte, V.33 tavsiyesi ise, 14400 bps hızda çalışabilen modemleri kapsamaktadır.

CCITT V.29 tavsiyesi

CCITT V.29 tavsiyesi modemleri özel hatlar üzerinde, 9600 bps hızında, tam çift yönlü çalışma modunda, eşzamanlı olarak çalışabilecek şekilde gerçekleştirilmişlerdir. Bu tavsiye kapsamı içerisindeki modemlerin temel özellikleri şunlardır :

1. 9600 bps hızında, özel hatlar üzerinde yarı veya tam çift yönlü çalışma modunda çalışabilir.
2. 1700 Hz taşıyıcı fekansına sahiptir.
3. Hata oranını yüksek olması durumunda, otomatik olarak 7200 veya 4800 bps hızlarına düşme özelliğine sahiptir (Automatic fallback rate).
4. 9600 bps hızında, 2400 baud modülasyon hızında bilgi iletirler. Böylece her modülasyon aralığında dört bitin yollanması gereklidir.
5. Otomatik uyarlamalı dengeleyiciye sahiptir.
6. Seçime bağlı olarak, dört ayrı veri dizisini taşıyabilen (toplam hızları 9600 bps hızını aşamayan) bir çoğullayıcı (multiplexer) içerir.
7. Modülasyon türü olarak dik genlik modülasyonu (QAM) kullanır.

V.29 tavsiyesi modemlerinde iletilecek olan veri dizisi önce çırpıcı devresinden geçirilir. Daha sonra dört bitlik gruplara (quad bits) ayrılır. Her dört bitlik grup içerisinde, zaman içerisindeki ilk bit iletilen işaretin genliğini belirler. Geri kalan üç bit ise, uygulanacak olan faz kaymasını belirlemektedir. 7200 bps "fallback" hızında, her işaretleşme aralığında sadece üç bit gönderilir ve genlik değişmesi yoktur. 4800 bps hızında, her işaretleşme aralığında iki bit gönderilir ve buda sadece dört adet faz kayması durumunu belirler.

CCITT V.32 tavsiyesi

Bu standart değiştirilmiş dik genlik modülasyonu tekniği üzerine dayandırılmıştır ve bu tavsiye içerisinde yer alan modemler özel veya anahtarlamalı telefon şebekesi üzerinde 9600 bps hızında yarı veya tam çift yönlü olarak çalışabilirler. Bu tavsiye içerisindeki modemlerin temel özellikleri şunlardır :

1. 9600 bps hızında özel hatlar üzerinde veya anahtarlamalı telefon şebekesi üzerinde, tam çift yönlü çalışma modunda, eşzamanlı veya eşzamansız olarak çalışabilir.
2. 1800 Hz taşıyıcı frekansına sahiptir.
3. Frekans paylaşımlı kanal ayırma özelliğine sahiptir.
4. Uyarlamalı dengeleyici ile hattı dengeleme özelliğine sahiptir.
5. 9600 bps hızında, 2400 baud modülasyon hızında bilgi iletir.
6. Kafes (Trellis) kodlaması yapabilen alternatif modülasyon tekniğine sahiptir.

V.32 modem birbirine zıt yönde iki yüksek hızlı kanal kurabilir. Bu kanalların her ikisi de yaklaşık olarak aynı band genişliğini kullanmaktadır. Gönderilen ve alınan işaretlerin aynı band genişliğini kullanabilmesi için bir yankı giderme tekniği kullanılır.

V.32 tavsiyesi içerisinde, 2400, 4800 ve 9600 bps hızlarında eşzamanlı iletim, modeme bu hızlarda eşzamansız verinin verilmesi ile elde edilir. Bu destek, modem içerisine yerleştirilmiş olan bir dönüştürücü ile sağlanmaktadır.

V.32 tavsiyesi, 9600 bps hızında iki alternatif modülasyon tekniği tanımlar. Bunlar kafes kodlaması ve genişletilmiş işaret uzayına sahip olmayan (non-redundant) kodlama tekniğidir. Bu kodlama tekniğinin 16 noktalı, kafes kodlamasının kullanımı ise 32 noktalı işaret uzayının oluşmasına neden olur.

CCITT V.33 tavsiyesi

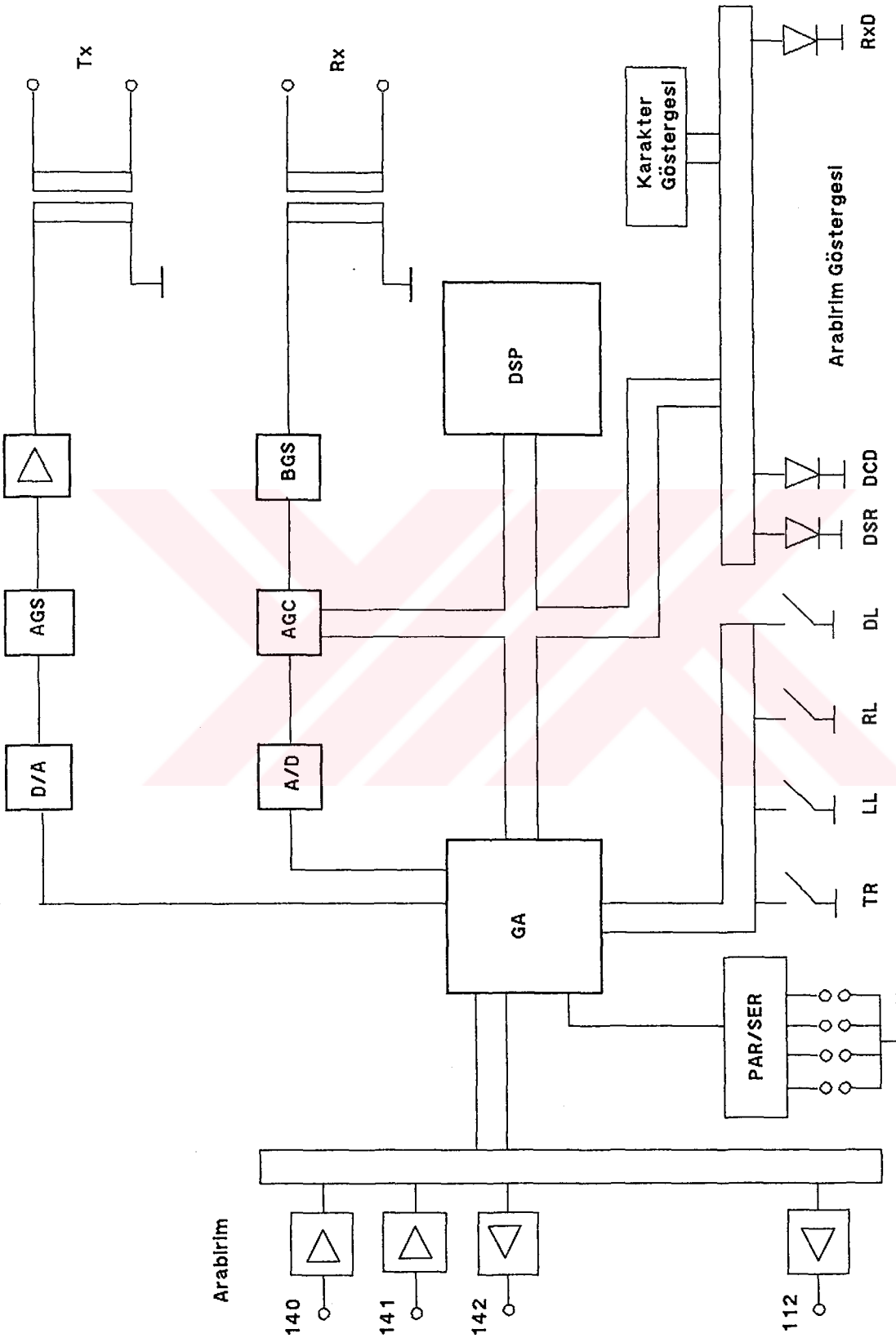
Bu tavsiye içerisindeki modemlerin temel özellikleri şunlardır :

1. 14400 bps hızında, dört telli özel devreler üzerinde, tam çift yönlü çalışma modunda, eşzamanlı ve eşzamansız olarak çalışabilir.
2. 1800 Hz taşıyıcı frekansına sahiptir.
3. Uyarlamalı dengeleyici ile hattı dengeleme özelliğine sahiptir.
4. Hata oranının yüksek olması durumunda otomatik olarak 12000 bps ve 9600 bps hızlarına düşme özelliğine sahiptir.
5. 14400 bps hızında, 2400 baud modülasyon hızında bilgi iletir.
6. 12000, 9600, 7200, 4800 ve 2400 bps veri hızları için bir çoğullayıcı içermesi seçeneği vardır.
7. Kafes kodlaması yapabilen alternatif modülasyon tekniğine sahiptir.

V.33 tavsiyesi modemi, kafes kodlamasını dik genlik modülasyonu ile birleştirir. V.33 modemin işaret uzayı, V.32 modemin kafes kodlamalı işaret uzayına benzerdir. İşaret uzayı V.33 modeminde 128 noktaya genişletilmiştir.

2.5 V.33 Modemin Özellikleri

Bu aileye ait olan modemler, esas olarak özel kalitedeki, kiralık hatlar üzerinde kullanılmak üzere tasarlanmışlardır. Hatların M1020 veya M1025 tavsiyelerine uygun devreler olması gereklidir. Ancak bu, modemin daha düşük kaliteli devreler üzerinde kullanımını engellemez. Tavsiye edilen bu modemin esas olarak kullanımı dört telli devreler üzerindedir. Anahtarlamalı devreler üzerinde yarı çift yönlü modda ve çok noktalı kanallar üzerinde çalışma özelliklerine de sahiptir. Eğer tavsiye edilen devreler üzerinde kabul edilebilir bir başarımın sağlanması isteniyorsa, modem uygulaması sırasında uygun dengeleme tekniklerinin seçilmesine dikkat edilmelidir. Bu modemin temel karakteristikleri şunlardır :



Şekil 2.5.1 V.33 modemin işlevsel yapısı

1. Düşük kalitedeki hatlar üzerindeki yüksek hata oranlı iletimde, otomatik olarak 12000 bps hızına düşme özelliği (12000 bps fallback rate),

2. Sürekli taşıyıcılı, tam çift yönlü çalışma modunda çalışabilme özelliği,

3. Eşzamanlı çalışma modunda birleştirilmiş genlik ve faz modülasyonu,

4. Sekiz durumlu kafes kodlu modülasyon tekniğine sahip olma özelliği,

5. 12000, 9600, 7200, 4800 ve 2400 bits/s hızlarının birleştirilerek kullanımını sağlayan, seçime bağlı çoğullama özelliği

Şekil 2.5.1 'de, V.33 modemin genel işlevsel devre yapısı görülmektedir.

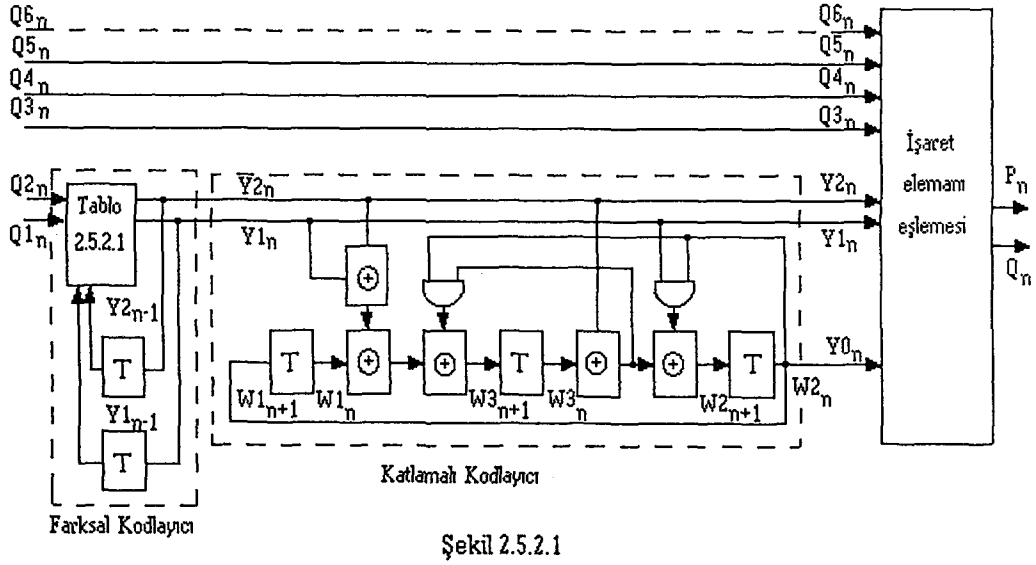
2.5.1 Hat İşaretleri

Taşıyıcı frekansı 1800 +/- 1 Hz olup güç seviyeleri CCITT V.2 tavsiyesine uygundur.

2.5.2 İşaret Uzayı Kodlaması

14400 bps hızında gönderilmek üzere çırpılmış olan veri bloğu, peşi sıra gelen altı bitten oluşan gruplara ayrılır.

Şekil 2.5.2.1 'de görüldüğü gibi, her grup içerisinde zaman içerisindeki ilk iki bit olan $Q1_n$ ve $Q2_n$ bitleri ilk önce $Y1_n$ ve $Y2_n$ olarak tablo 2.5.2.1 'e göre farksal olarak kodlanır. Farksal olarak kodlanmış olan (Differantially encoded) $Y1_n$ ve $Y2_n$ bitleri bir sistematik katlamalı kodlayıcıya giriş olarak uygulanır. Katlamalı kodlayıcı $Y0_n$ bitini üretir. Bu fazlalık bit ile birlikte $Y1_n$, $Y2_n$, $Q3_n$, $Q4_n$, $Q5_n$, $Q6_n$ veri bitleri şekil 2.5.2.2 'de görülen işaret uzayı diyagramına göre , gönderilecek olan işaret elemanını koordinatlarına eşlenir



Şekil 2.5.2.1
14400 ve 12000 bits/s hızları için kafes kodlayıcısı

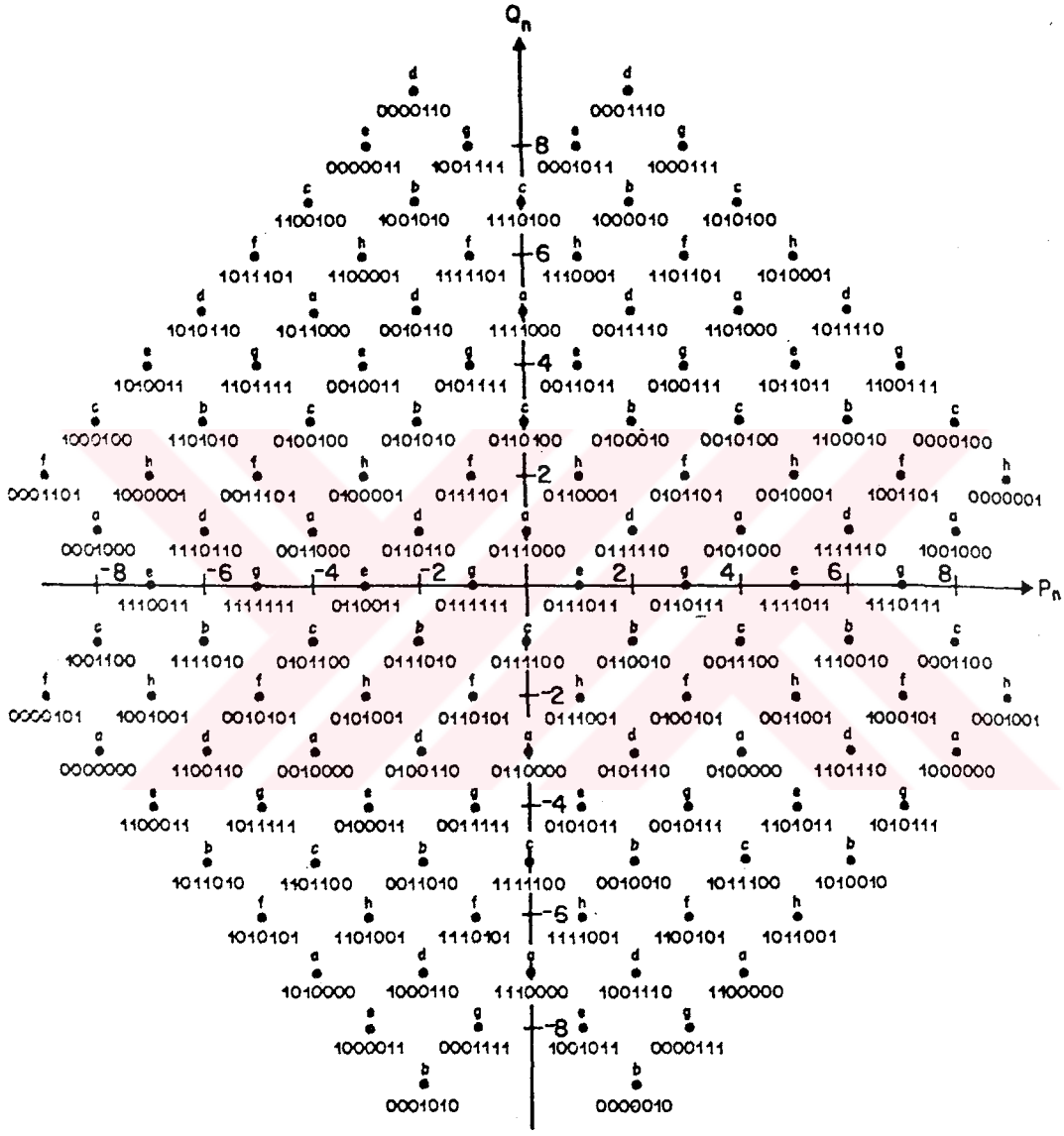
Girişler		Önceki Çıkışlar		Çıkışlar	
$Q1_n$	$Q2_n$	$Y1_{n-1}$	$Y2_{n-1}$	$Y1_n$	$Y2_n$
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	1	0	0
0	1	1	0	1	1
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	1	1
1	0	1	0	0	1
1	0	1	1	0	0
1	1	0	0	1	1
1	1	0	1	1	0
1	1	1	0	0	0
1	1	1	1	0	1

Tablo 2.5.2.1

Girişler		Önceki Çıkışlar		Faz	Çıkışlar		4800 bps için	Koordinatlar	
$Q1_n$	$Q2_n$	$Y1_{n-1}$	$Y2_{n-1}$	Quadrant	$Y1_n$	$Y2_n$	işaret elemanı	Re	Im
0	0	0	0	+90	0	1	D	-2	+6
0	0	0	1		1	1	A	-6	-2
0	0	1	0		0	0	C	+6	+2
0	0	1	1		1	0	B	+2	-6
0	1	0	0	0	0	0	C	+6	+2
0	1	0	1		0	1	D	-2	+6
0	1	1	0		1	0	B	+2	-6
0	1	1	1		1	1	A	-6	-2
1	0	0	0	+180	1	1	A	-6	-2
1	0	0	1		1	0	B	+2	-6
1	0	1	0		0	1	D	-2	+6
1	0	1	1		0	0	C	+6	+2
1	1	0	0	+270	1	0	B	+2	-6
1	1	0	1		0	0	C	+6	+2
1	1	1	0		1	1	A	-6	-2
1	1	1	1		0	1	D	-2	+6

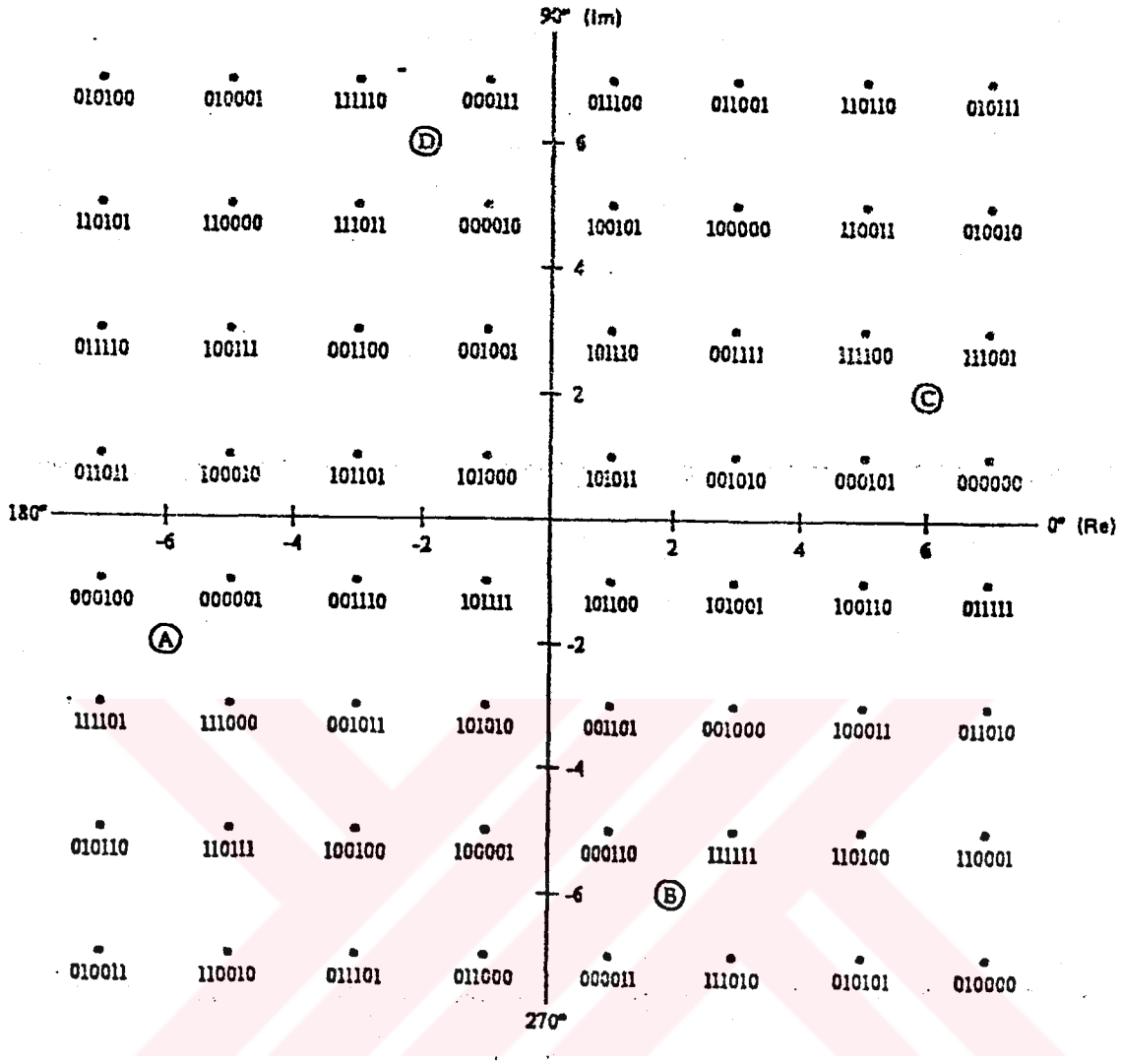
Tablo 2.5.2.2

Hat kalitesinin kötü olması durumunda alternatif hız olan 12000 bits/s 'de iletilmek üzere çırpılan veri bloğu, peşi sıra gelen 5 bitten oluşan gruplara ayrılır. Şekil 2.5.2.1 'de görülen kafes kodlama yapısı aynen kullanılır. Ancak burada, $Q6_n$ için olan hat kaldırılmakta ve işaret elemanı kodlaması şekil 2.5.2.3 'deki gibi olmaktadır.



Şekil 2.5.2.2

14400 bits/s hızda kafes kodlaması için işaret uzayı diyagramı ve eşleme
(İkili sayılar $Q6_n$,$Q5_n$,$Q4_n$,$Q3_n$,$Y2_n$,$Y1_n$,$Y0_n$ dizisine karşı gelmektedir.A, B, C, D zamanlama işaretlerini göstermektedir.)



Şekil 2.5.2.3

12000 bits/s hızında kafes kodlaması için işaret uzayı diyagramı ve eşleme

Tablo 2.5.2.2, zamanlama işaretlerinin üçüncü segmentindeki 4800 bits/s hızında işaret için farksal kodlamayı göstermektedir.

2.5.3 Veri İşaretleşme Ve Modülasyon Oranları

Veri işaretleşmesi hızları 14400 ve 12000 bits/s +/- 0.01% olabilir. Modülasyon hızı ise 2400 baud +/- 0.01% dir.

2.5.4 İşaret Frekansının Toleransı

Vericide taşıyıcı frekansı töleransı +/- 1 Hz dir. Modemler arasındaki bağlantıda, maximum +/- 6 Hz 'lik bir frekans kayması olduğunu kabul edelim. Bu durumda alıcı, alınan işaret frekansı içerisinde +/- 7 Hz 'lik hataları kabul etmelidir.

2.5.5 İşaretleşme (Interchange) Devreleri

No	İsim
102	İşaret toprağı veya genel dönüş
103	Gönderilen Veri
104	Alınan Veri
105	Gönderme isteği
106	Göndermeye Hazır
107	Veri Seti Hazır
109	Veri Kanalı Alınan Hat İşaret Sezicisi
111	Veri İşaretleşme Hız Seçicisi (DTE kaynaklı)
112	Veri İşaretleşme Hız Seçicisi (DCE kaynaklı)
113	Verici İşaret Elemanı Zamanlaması (DTE kaynaklı)
114	Verici İşaret Elemanı Zamanlaması (DCE kaynaklı)
115	Alıcı İşaret Elemanı Zamanlaması (DCE kaynaklı)
140	Lokal Çevrim/Kontrol Testi
141	Lokal Çevrim (Loopback)
142	Test Göstergesi

Tablo 2.5.5.1

İşaretleşme Devreleri

Yukarıda belirtilen bütün temel işaretleşme devreleri ve diğerleri, V.24 tavsiyesinin özelliklerini taşımalıdır.

109 Devresinin Eşik Ve Cevap Süreleri

-26 dB 'den daha büyük işaret seviyesi — 109 devresi aktif (on)

-33 dB 'den daha küçük işaret seviyesi — 109 devresi aktif değil (off)

109 devresinin -26 dB ve -33 dB arasındaki seviyelerdeki durumu belirtilmemiştir. Ancak, işaret sezicisi bir histerizis davranışı gösterebilir. Böylece, aktif olmayan durumdan aktif duruma geçişin gerçekleştiği seviye, tersi durumdaki geçiş seviyesinden 2 dB daha büyüktür.

Cevap Süreleri

Aktif durumdan aktif olmayan duruma geçiş — 40+/-10 ms

Aktif olmayan durumdan aktif duruma geçiş :

1. Başlangıçtaki dengeleme işlemi için, verinin 104 devresinde belirmesinden önce 109 devresi aktif (on) durumda olmalıdır.

2. Veri iletimi sırasındaki yeniden dengeleme (re-equalization) işlemi sırasında 109 devresi aktif durumunu korumalıdır. Bu süre esnasında, 104 devresi ikili seviyede 1 durumuna kenetlenebilir.

3. Aktif durumdan aktif olmayan duruma geçiş cevap süresinden daha uzun süren bir hat işaret kaybı yaşandığında ;

a. Yeniden dengeleme işlemi gerekiyorsa 25+/-10 ms

b. Yeniden dengeleme işlemi gerektiğinde, 104 devresi üzerinde veri belirmeden önce 109 devresi aktif durumda olmalıdır.

109 devresinin cevap süreleri, 103 devresine sayısal 1 seviyesinin uygulanması ile üretilen hat işaretinin modemin hat alış terminallerine bağlanması veya sökülmesi ile, buna bağlı olarak 109 devresinin durum değiştirmesi arasında geçen süredir.

106 Devresinin Cevap Süresi

Bütün alıştırma (training) işlemlerinin tamamlanmasından itibaren, 105 devresinin aktif olmayan durumdan aktif duruma geçmesi ile, 106 devresinde aynı olayın gözlenmesi arasında geçen süre 15+/-5 ms olacatır. 105 devresinin aktif halden aktif olmayan duruma geçmesi ile 106 devresinde aynı olayın gözlenmesi arasında geçen süre bütün geçerli işaret elemanlarının gönderilmesi tamamlanacak kadar uygun bir değerde seçilmelidir.

İşaretleşme Devrelerinin Elektriksel Karakteristikleri

V.28 tavsiyesine uygun elektriksel karakteristiklerin ve ISO 2110 tarafından belirlenen pin tanımlamalarının uygulanması tavsiye edilmiştir.

Zamanlama Düzenlemeleri

DTE elemanına, verici işaret elemanı zamanlaması (114 devresi) ve alıcı işaret elemanı zamanlamasını (115 devresi) sağlamak için, bir zamanlama kaynağı modem içerisinde bulunmalıdır. Bu düzenlemede verici bağımsız bir zamanlama kaynağı olarak veya lokal çevrim zamanlaması ile çalışabilir(Verici zamanlaması alıcı zamanlama bilgisini izler). Lokal çevrim zamanlaması bazı haberleşme şebekelerinde istenebilir. Alternatif olarak, verici zamanlama bilgisi DTE kaynaklı olabilir ve 113 numaralı işaretleşme devresi vasıtasıyla modeme gönderilebilir. Bu tür bir zamanlama düzenlemesi, üzerinde tek bir zamanlama bilgisi kaynağı olması istenen haberleşme şebekelerinde kullanılabilir. Bu şekilde, şebeke içerisinde bulunan DTE ve DCE elemanları arasında eşzamanlamanın sağlanamamasından dolayı oluşan haberleşme bağlantısı kopuklukları olmayacaktır.

2.5.6 Çırpıcı Ve Ters Çırpıcı

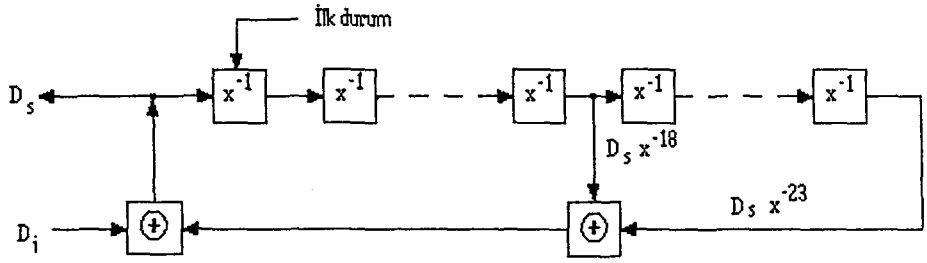
Modem içerisinde kendi kendini eşzamanlı hale getirebilen ve $1+x^{-18}+x^{-23}$ üretim polinomuna sahip olan bir çırpıcı ve ters çırpıcı birimi yer almaktadır.

Vericide çırpıcı, giriş veri dizisinin katsayılarını oluşturduğu mesaj polinomunu kendi üretim polinomu ile etkin bir şekilde bölerek iletilecek olan veri dizisini belirlemektedir. Alıcıda, alınan veri

dizisinin katsayılarının oluşturduğu polinom, çırpıcı üretim polinomu ile çarpılarak mesaj dizisi belirlenmektedir.

Çırpma İşlemi

Mesaj polinomu, üretim polinomu olan $1+x^{-18}+x^{-23}$ ile bölünmektedir. Bu bölüm sonucu ortaya çıkan polinomun katsayıları (azalan sırada), iletilecek olan veri dizisini belirlemektedir. Şekil 2.6.6.1 'de çırpıcı devresinin yapısı görülmektedir.



Şekil 2.5.6.1

Çırpıcı

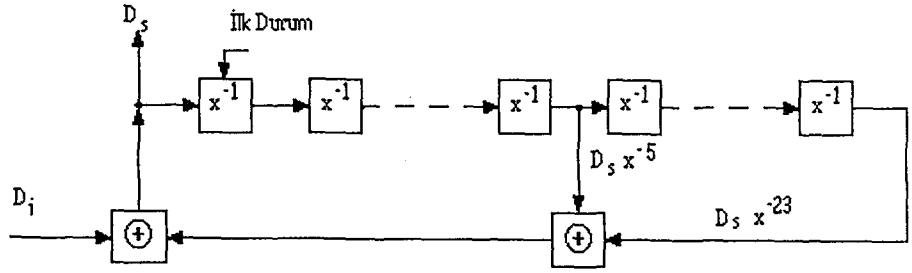
Çırpıcı, 2. ve 3. segmentlerde 4800 Hz frekansında zamanlama darbeleri ve 4. segmentte ise veri hızındaki zamanlama darbeleri ile beslenmelidir. 2, 3, 4. segmentlerde ve normal veri iletimi sırasında, kayan saklayıcı çırpılmış olan D_S verisi ile beslenir. Bu durumda çırpılmış olan veri ;

$$D_S = D_i + D_S x^{-18} + D_S x^{-23}$$

olacaktır. Burada D_i, 2. ve 4. segmentler ile 3. segment içerisindeki hız dizisi esnasında ikili seviyede "1" durumundadır.

Ters Çırpma İşlemi

Alınan diziyi temsil eden polinom, üretim polinomu ile çarpılarak mesaj polinomu tekrar elde edilir. Elde edilen polinomun katsayıları azalan sırada çıkış veri dizisi olan D_O 'ı verecektir. Aşağıdaki şekilde ters çırpıcı devresinin yapısı görülmektedir.



Şekil 2.5.6.2

Ters Çırpıcı

2.5.7 Eşzamanlama İşaretleri Ve Bitlerin Eşlenmesi

Eşzamanlama işaretlerinin iletimi modem tarafından başlatılabilir. Alıcı modem yeniden eşzamanlamaya ihtiyaç duyduğunda 106 devresini aktif olmayan duruma sokar ve bir eşzamanlama işaret dizisi üretir. Bütün veri işaretleşme hızları için eşzamanlama işaretleri dört segmente ayrılır. Bu segmentler tablo 2.5.7.1 'de görülmektedir.

Segment 1, şekil 2.5.2.2 de görülen A ve B durumları arasında, 256 simge aralığı süresince değişimleri kapsamaktadır.

Segment 2, dengeleyici alıştırma dizisidir. Bu kısım, A, B, C, D işaret elemanlarının sırayla iletiminden oluşmaktadır. Bu işaret elemanları şekil 2.5.2.2 ve 2.5.2.3 de görülmektedir. Dengeleyici alıştırma dizisi, $1+x^{-18}+x^{-23}$ veri çırpıcısı tarafından üretilmiş, "pseudo-random" bir dizidir. Segment 2 süresince farksal kodlama işlemi iptal edilir ve .çırpılan veri bitleri aşağıdaki gibi kodlanır.

00=C 01=D 11=A 10=B

	Segment 1	**Segment 2 TRN**	**Segment 3**	**Segment 4**	**Toplam**
Hat işaretinin tipi	Değişimler ABAB	Dengeleyici alıştırma dizisi	Hız dizisi	Çırpılmış ikili "1" dizisi	Toplam eşzamanlama işareti süresi
Simge aralıkları sayısı	256	2976	64	48	3344
Yaklaşık süre (ms)	106	1240	27	20	1393

Tablo 2.5.7.1

Çırpıcının başlangıç durumu, aşağıdaki çıkış dizisi ve bunlara karşı gelen işaret elemanlarını üretecek şekilde seçilmelidir. Bu durumda çırpıcı girişinde ikili "1" dizisi bulunmaktadır.

00 01 00 01 00 01 00 01 00 01 00 01 10 01 10 01

C D C D C D C D C D C D B D B D

Segment 2, 2976 simge aralığı süresince devam eder.

Segment 3, hız işaretidir. Hız işareti, 8 kere ard arda gönderilen, 16 bit uzunluğunda sayısal dizidir. Tablo 2.5.7.2 'de görülen dizi, çırpılır ve tablo 2.5.2.2 'de görüldüğü gibi farksal olarak kodlanarak 4800 bits\s hızında iletilir. Her hız dizisinin ilk iki biti ve peşi sıra gelen iki bitlik gruplar bir işaret durumu olarak kodlanır. Hız işareti modemler arasında veri işaretleşme hızını belirlemek ve diğer çalışma bilgisini (çoğullayıcı çalışması vb.) sağlamak içim kullanılır.B14 =0 olduğu zaman sadece

veri işaretleşme hızı bilgisi tablo 2.5.7.2 'ye göre taşınır. B14 =1 olduğu zaman tablo 2.5.7.3 'de görülen bit eşlemeleri kullanılır.

Hız dizisinin alıcı tarafından algılanabilmesi için, peşi sıra gelen iki aynı 16 bitlik dizinin alınması ve bu dizilerin B0-3, B11, ve B15 bitlerinin tablo 2.5.7.2 ve 2.5.7.3 'e uygun olması gereklidir. Hız dizisinin algılanmasından sonra, alıcı bu dizide belirtilen en yüksek veri hızı ve çoğullayıcı çalışması ile veri almaya kendini şartlamalıdır.

Bir hız işareti iletilirken, 4.segmentin veri işaretleşme hızı 14.4 kbits/s olacaksa, modem B8 B9 bitlerini 11 veya 01 'e eşleyerek gönderir. Eğer 4. segmentin veri işaretleşme hızı 12 kbits/s olacaksa, modem B8 B9 bitlerini 10 yaparak hız işaretini gönderir. B4 ve B5 bitlerinin diğer birleşimi, B6, B8, B10, B12 ve B13 bitlerinin diğer bir çalışma bilgisi taşıdığını gösterir.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0					1				1			0	1

B0-3, B7, B11, B15		Alınan hız işaretine eşzamanlama için
B4-6		Henüz tanımlı değil
B8	1	12000 bits\s hızında veri alma ve gönderebilme yeteneği
B9	1	14400 bits\s hızında veri alma ve gönderebilme yeteneği
B10, B12, B13		Henüz tanımlı değil

Tablo 2.5.7.2

Bit eşlemesi

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0				1				1			0	1

B0-3, B7, B11, B15		Alınan hız işaretine eşzamanlama için
B4, B5	00	B6, B10, B12, B13 çoğullayıcı çalışmasını belirlemektedir.
B8	1	12000 bits\s hızında veri alma ve gönderme yeteneği
B9	1	14400 bits\s hızında veri alma ve gönderme yeteneği
B6, B10, B12, B13		Çoğullayıcının çalışma seçimini gösterir.

Tablo 2.5.7.3

Bit eşlemesi

B4 ve B5 bitlerinin 00 olması durumunda B6, B10, B12, B13 bitlerinin gösterdiği çalışma bilgileri aşağıda belirtilmiştir.

a. B6, B10, B12, B13 =hepsi "0" otomatik olmayan çalışma şekli

b. B6, B10, B12, B13 =hepsi "1" uzaktan yönetilebilir çalışma şekli. Eğer modem bu şekilde çalışmak üzere ayarlanmış ise daima bu diziyi gönderir.

c. B6, B10, B12, B13 =1 ve 11 arasındaki ondalık sayıların ikili gösterimi (B6=MSB), istenilen değişik çoğullayıcı çalışma seçeneklerinden birini gösterir.

d. B6, B10, B12, B13 =kullanılmayan kombinasyonlar üreticilerin seçimine açık durumdadır.

e. Her iki modeminde aynı çoğullayıcı çalışma şekline ayarlanması veya modemlerden birinin uzaktan yönetilebilir çalışmaya ayarlanması tavsiye edilir.

Segment 4 süresince kullanılacak olan farksal kodlama tablo 2.5.2.1 'de görülmektedir. Farksal kodlayıcı, bir önceki segmentin son simgesi kullanılarak başlangıç durumuna getirilmelidir. Segment 4, katlamalı kodlayıcının gecikme elemanlarının ilk durumları "0" durumuna çekilmiş durumda başlamalıdır. Bu segment, çırpıcı girişine sürekli "1" dizisi uygulanmış halde segment 3 'de belirtilen en yüksek hız değerinde iletimi başlatır. 4. segmentin sonunda 106 devresi on durumuna geçirilir ve kullanıcı verisi çırpıcı girişine uygulanır.

2.5.8 Alıştırma Ve Yeniden Alıştırma İşlemi

Alıcı içerisinde bir otomatik uyarlamalı dengeleyici yer alacaktır. Alıcı, dengeleme kaybını sezecek ve lokal vericisinde bir eşzamanlama işareti başlatacaktır.

Alıcı, uzak vericiden gelen bir eşzamanlama işaretini ve hız dizisini sezecek, lokal vericisinde bir eşzamanlama işareti başlatacaktır. Bu işaret, uzak vericiden gelen eşzamanlama işaretinin alınışı sırasında herhangi bir anda, 105 devresinin durumuna bağlı kalınmaksızın üretilebilir.

Modemlerden herhangi biri eşzamanlama işaretini üretebilir. Eşzamanlama işareti, modemleren birinin alıcısı dengeleme kaybı sezdiğinde veya veri işaretleşme hızında ve çoğullayıcı çalışma seçiminde herhangi bir değişiklik oluştuğunda başlatılır. Eşzamanlama işaretinin başlatılmasından sonra modem, uzak vericiden bir eşzamanlama işareti bekler.

Gönderilen eşzamanlama işaret dizisinden sonra belirli bir zaman aralığı içerisinde (tahmin edilen maksimum çift yönlü yayılma gecikmesi), modem uzak vericiden bir eşzamanlama işareti alamazsa başka bir işaret yollar. Bu zaman aralığı 1.2 saniye olarak tavsiye edilir.

Eğer modem alınan işaret dizisine eşzamanlı hale gelmeyi ve hız işaretini sezmeyi başaramazsa veya istenilen hızı sağlayamıyorsa, yeni bir eşzamanlama işareti gönderir. Modem istenilen hızın üzerinde iletim yapabiliyorsa yeni bir eşzamanlama işareti gönderebilir. Bu eşzamanlama işareti, mümkün olan en yüksek veri işaretleşme hızı bilgisini içeren bir hız dizisini kapsayacaktır.

Eğer modem eşzamanlama işaretini başlatmadan bir başka işaret alırsa, alıcı bu işarete uygun olarak eşzamanlı hale gelir ve karşı modeme sadece bir eşzamanlama işareti yollar.

2.5.9 Çoğullama

12000, 9600, 7200, 4800 ve 2400 hızlarındaki veri alt kanallarını iletim için tek bir bit dizisinde birleştirmek üzere bir çoğullama seçeneği kapsanabilir.

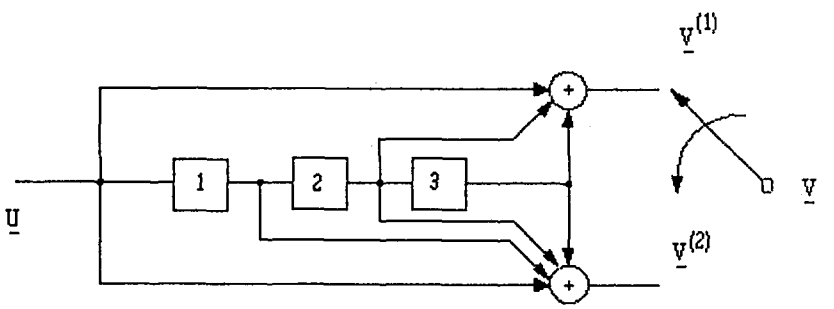
BÖLÜM 3 . KAFES KODLAMALI MODÜLASYON

Düşük ve orta veri hızlarında (< 4800 bits\s) bağımsız, simge-simge iletimi gerçekleştiren işaretleşme yöntemleri ses bandı devreleri üzerinde alçak hata oranlarını sağlamada yeterlidir. Yüksek veri hızlarında (> 9600 bits\s) ise bu durum geçerli değildir. Band verimlilikli QAM ve optimum işaret kümeleri bile kanallar üzerinde kabul edilebilir alçak hata oranlarını sağlamada yetersiz kalmaktadır. Yüksek hızlı veri iletiminin analog hatlar üzerinde istenileni gerçekleştirebilmesi için yeni işaretleşme metodlarının bulunması gereklidir.

Bu bölümde, yüksek hızlı veri iletimi için bir kanal kodlama türü açıklanacaktır. Bu nedenle katlamalı kodlar veya daha genel olarak kafes kodları, yumuşak karar vermeli Viterbi kod çözücüsü ile birlikte işaret algılama yeteneğinin arttırılması amacıyla ele alınacaktır. Ses bandlı kanallar üzerinde band genişliğinin arttırılması mümkün değildir. Bunun yerine, çok seviyeli işaretleri kullanan daha geniş bir işaret alfabesi gerekli olacaktır. Ungerboeck, iyileştirilmiş sistem başarımı için çok seviyeli kafes kodlarının tasarımını araştırmıştır Toplamsal Gauss gürültüsü altında eşdeğer kodlamasız sistemlere göre 3-6 dB 'lik kodlama kazancının 4-64 durumlu kafes yapıları ile sağlanabileceğini göstermiştir. İyi bilindiği gibi ses bandı kanalları üzerindeki sürekli durum hataları veya kesintileri, aynı zamanda doğrusal veya doğrusal olmayan gürültü, faz kaymaları, frekans bozuklukları ve toplamsal gürültüden oluşabilir. Burada basit bir şekilde, ses bandlı kanallara benzeyen kanallar üzerinde çok seviyeli kafes kodlarını kullanan alıcının ve vericinin gerçek zaman başarımındanda bahsedilecek ve kodlamalı ve kodlamasız sistemlerin toplamsal Gauss gürültüsü dışındaki hatalara olan duyarlılığı incelenecektir.

3.1 Katlamalı Kodlar

Doğrusal blok kodlarda kod sözcükleri ile mesaj işaretleri arasında birebir eşleme olmasına rağmen, katlamalı kodlarda herhangi bir t anı içerisinde bir mesaj işaretine karşı düşen kod sözcüğü değişebilir. (n,k,m) katlamalı kodunda, n çıkış biti sayısını, k giriş biti sayısını, m giriş belleği sayısını göstermektedir. R=k/n oranlı katlamalı kodlayıcı l=0,1,2,..... kodlama adımını göstermek üzere her l için, k uzunluklu U_l giriş dizisine n uzunluklu V_l ikili çıkış dizisini karşı düşürmektedir. Şekil 3.1.1 'de R=1/2 oranlı (2,1,3) katlamalı kodlayıcı yapısı görülmektedir.



Şekil 3.1.1

Bu şekil üzerinde enformasyon dizisi olan $\underline{U}$,

$\underline{U}=(U_0,U_1,U_2,\ldots\ldots)$ ve çıkış dizileri $\underline{V}^{(1)}$ ve $\underline{V}^{(2)}$,

$\underline{V}^{(1)}=(V_0^{(1)},V_1^{(1)},V_2^{(1)},\ldots\ldots)$,

$\underline{V}^{(2)}=(V_0^{(2)},V_1^{(2)},V_2^{(2)},\ldots\ldots)$,

olarak gösterilebilir. Burada çıkış dizileri, $\underline{U}$ enformasyon dizisi ile kodlayıcının impuls yanıtlarının katlanması şeklinde elde edilir. Yukarıdaki devrenin impuls yanıtı, girişe $\underline{U}=(10000\ldots\ldots)$ uygulanması ile bulunur. Giriş belleğinin uzunluğu m olduğundan impuls yanıtları en fazla m+1 uzunlukludur. Bu impuls yanıtlarını;

$g^{(1)}=(g_0^{(1)},g_1^{(1)},g_2^{(1)},\ldots\ldots)$,

$g^{(2)}=(g_0^{(2)},g_1^{(2)},g_2^{(2)},\ldots\ldots)$,

ile gösterirsek bu durumda kodlama denklemleri ;

$\underline{V}^{(1)}=\underline{U}*g^{(1)}$

$\underline{V}^{(2)}=\underline{U}*g^{(2)}$ olacaktır.

Yukarıdaki şekilde açıkça görülmektedir ki her yeni veri çıkışı, ötelemeli kaydedicinin 1 ve 2 numaralı gözlerinde saklanan önceki veri bitlerine bağlıdır. Bu bit dizileri kodlayıcının bulunduğu durumları göstermektedir. Bu durumda kodlayıcı çıkışı ;

$\underline{V}=(V_0^{(1)},V_0^{(2)},V_1^{(1)},V_1^{(2)},\ldots\ldots)$, olacaktır.

Görüldüğü gibi her bir giriş mesajına karşılık iki kod elemanı üretilmektedir. Buna göre katlamalı kodlama işlemi V=U*G olarak matris formatında gösterilebilir. G üreteç matrisi olarak adlandırılır ve R=k\n kodlama oranına sahip (n,k,m) genel durumu için bu matris;

$$G=\begin{bmatrix} G_0 & G_1 & G_2 & & & G_m & & \\ & G_0 & G_1 & G_2 & & & G_m & \\ & & G_0 & G_1 & G_2 & & & G_m \\ & & & \ddots & \ddots & \ddots & & \end{bmatrix}$$

olacaktır. Bu durumda ;

$\underline{U}=(U_0,U_1,\ldots\ldots)$

$=(U_0^{(1)},U_0^{(2)},\ldots\ldots,U_0^{(k)},U_1^{(1)},U_1^{(2)},\ldots\ldots,U_1^{(k)},\ldots\ldots)$

$\underline{V}=(V_1,V_2,\ldots\ldots)$

$=(V_0^{(1)},V_0^{(2)},\ldots\ldots,V_0^{(n)},V_1^{(1)},V_1^{(2)},\ldots\ldots,V_1^{(n)},\ldots\ldots)$

olacaktır.

3.2 Klasik Hata Düzeltmeli Kodlama

Klasik haberleşme sistemlerinde modülasyon ve hata düzeltme kodlaması işlemleri ayrılmıştır. Klasik çok seviyeli (genlik ve\veya faz) modülasyon sistemlerinde, her modülasyon aralığında, modülatör m sayıda simgeyi (bits) $M=2^m$ mümkün olan iletim işaretine eşler ve demodülatör en yakın komşu kararını, alınan her işaret üzerinde vererek m biti tekrar elde eder.

İki boyutlu taşıyıcı modülasyonunda, 1/T [işaret/saniye] modülasyon hızında işaretlerin birbiriyle girişime girmeden işaret gönderimi yapılabilmesi için, taşıyıcı frekansı civarında 1/T Hz'lik band genişliğine gereksinim gösterir.

Klasik kodlayıcı ve kod çözücüler hata düzeltme için, sayısal ve Q dizili ayrık bir kanal üzerinden gönderilmiş kod simgeleri üzerinde çalışırlar.$k/n < 1$ kod oranında, n-k sayıda kontrol simgesi her k bilgi simgesine eklenmiştir. Kod çözücü sadece ayrık kod simgelerini aldığı için, Hamming uzaklığı kod çözme ve kod tasarımı için uygun bir uzaklık ölçüsüdür. Katlamalı kodlarda d^h_{min} minumum Hamming uzaklığı, kod çözücünün en azından $[(d^h_{min}-1) / 2]$ kod simge hatasını düzeltebileceğini garanti eder.

Eğer düşük SNR oranları veya kararsız işaret hataları modülasyon sisteminin başarımını etkilerse, hata düzeltme yeteneği ilave olarak gönderilen hata düzeltme bitleri yüzünden hızın düşmesine neden olur. Hız kaybını karşılamak için iki yöntem vardır:

1. Eğer kanal band genişliğinin arttırılması mümkün ise modülasyon hızını arttırmak.
2. Kanal band sınırlı ise modülasyon sisteminin işaret kümesini genişletmek.

3.3 Yumuşak Karar Vermeli Kod Çözme

Kodlanmış bir sistemde, verilen bir kod için hata olasılığı minumum olacak şekilde kod çözücü yapısı oluşturulacaktır.

Sert karar vermede optimum işlem, alınmış diziden en az pozisyonda değişiklik gösteren kod sözcüğünü kabul etmektir. Bu, kod sözcüğü ve alınan dizi arasındaki uzaklığı minumum yapan kod sözcüğünün seçilmesidir. Bu maksimum benzerlik kod çözücüsüdür ve yumuşak karar vermeli duruma genişletilebilir. Alıcıda sert karar verme işlemleri geri kazanılamayacak bilgi kaybına neden olmaktadır.

Kod çözme problemini tanımlamak için, ikili işaretleşmeyi toplamsal Gauss karakteristikli kanal üzerinde gerçekleştirdiğimizi düşünelim. Demodülatör, birleştirilmiş süzgeç çıkışındaki gerçek örnek değerlerini gösteren n adet sayı dizisini kod çözücüye verecektir.q 'nun gerçek alınmış dizi olduğunu ve s_l'inde iletildiği tahmin edilen kod sözcüğü olduğunu düşünelim. Optimum kod çözücü, $P_r(s_l/q)$ olasılığını maksimum yapan s_l yi seçecektir. Bayes kuralı uygulanırsa,

$P_r(s_l/q)=P(s_l,q)/P(q)=P(q/s_l).P_r(s_l)/P(q)$ olur.

Eğer bütün mesajların eşit olasılıklı olduğunu kabul edersek, $P_r(s_l/q)$ nun maksimize edilmesi $P(q/s_l)$'in maksimize edilmesi demek olacaktır. Her simgeyi etkileyen gürültü bağımsız, sıfır ortalamalı Gauss gürültüsü olduğundan ve $\sigma^2=N_0/2$ varyansına sahip olduğu için, $P(q/s_l)$ n Gauss olasılık fonksiyonunun bir ürünü olarak yazılabilir.

$$p(q/s_l)= \prod_{i=1}^{n} \left[\frac{1}{(\Pi N_0)^{1/2}}\right] . \exp\left[-(q_i-s_{li})^2 / N_0\right]$$

$$\left[\frac{1}{(\Pi N_0)^{1/2}}\right]^n . \exp\left[-\sum_{i=1}^{n} (q_i-s_{li})^2 / N_0\right]$$

Burada s_{li} s_l nin bileşenleridir ve bu ifade,

$$d^2=\sum_{i=1}^{n} (q_i-s_{li})^2$$

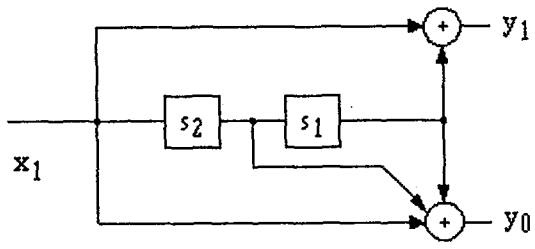
minimize edildiğinde maksimum olacaktır. Bu tahmin edilen ve alınan diziler arasındaki minumum Öklid uzaklığıdır. Kod çözme karar kuralı ile kanal çıkışında elde edilen $\{r_n\}$ dizisine en küçük karesel Öklid uzaklıklı bir $\{a_n\}$ dizisi, kodlanmış işaret kümesi arasından en uygun dizi olarak belirlenebilir. Öyle ki, $\{a_n\}$ dizisi aşağıdaki eşitliği sağlar :

$$|r_n-a_n|^2 = \min_{\{a_n\} \in C} \sum |r_n-a_n|^2$$

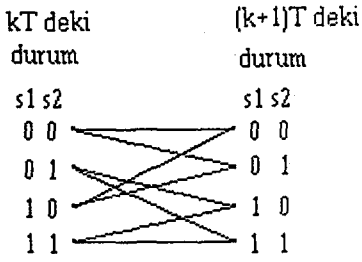
3.4 Kafes Kodlarının Tanıtımına Giriş Ve Kafes Kodları İçin Katlamalı Kodlar

İkili bilgiyi m/T bits/s oranında ileten bir sistem düşünelim.Burada m simge başına bit sayısı (bit/simge) ve 1/T de simge hızıdır (simge/saniye). Kodlamasız klasik QAM sistemlerinde, 2^m ayrık simge noktaları (genlik veya faz simgeleri) kullanılır ve bütün sürekli simgeler bağımsız olarak iletilir. Bir yaklaşım ile bu tür sistemlerin toplamsal Gauss gürültüsü altındaki hata başarımı simge noktaları arasındaki minimum uzaklığa bağlıdır. Bu uzaklık arttıkça hata olasılığı azalacaktır. Vericideki bu minimum uzaklık, devre üzerinde izin verilen ortalama güç, simge noktalarının seçimi ve sayısı ile belirlenmiştir. İki boyutta (genlik ve faz) bu işaret noktalarının optimal seçiminde bile noktalar arasındaki minimum uzaklık artan m ve sabit ortalama güç ile azalacaktır. Kodlama kullanmanın amacı iyileştirilmiş bir sistem başarımı sağlamaktır. Yani birbirine karışmaya yatkın olan işaretler arasındaki minimum uzaklığın, ortalama güç yükseltilmeden arttırılmasıdır. Kafes kodlaması m sayıdaki o andaki bit ve v sayıdaki geçmiş bit üzerinde m+j bit üretmek üzere işlem yapan bir katlamalı kodlayıcı ile gerçeklenebilir. Bu durumda kod oranı m/m+j olarak adlandırılır ve v sayıdaki geçmiş bit kodlayıcının durumu olarak adlandırılır. m+j bit iletimi için 2^{m+j} ayrık kanal simgesine gerek duyulacaktır. Dikkat edilen, genişletilmiş bir işaret kümesinin verilen bir ortalama güç için işaret noktaları arasında azaltılmış bir minimum uzaklığa neden olacağıdır. Bununla birlikte katlamalı kodlayıcının neden olduğu, peşpeşe gelen işaretler arasındaki bağımlılık yüzünden bu minimum uzaklık artık hata başarımının bir ölçüsü değildir. Bunun yerine hata başarımının ölçüsü, peşpeşe gelmesi izin verilen

simge dizileri arasındaki minimum uzaklıktır. Şekil 3.4.1 m=1, j=1 ve v=2 olan katlamalı kodlayıcıyı göstermektedir.Bu katlamalı kodlayıcının giriş çıkış ilişkisi bir kafes diyagramı ile kağıt üzerinde gösterilebilir. Şekil 3.4.2, bu diyagramı göstermektedir.

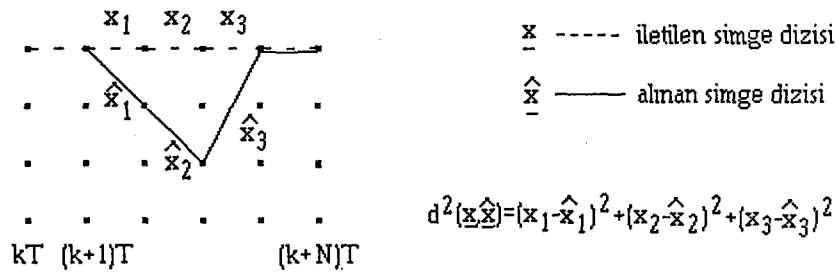


Şekil 3.4.1



Şekil 3.4.2

Bir durum anında yukarıda bulunan dal bir "0" bitinin neden olduğu geçişi göstermektedir. Modülasyonun (veya bit-simge eşlemesi) kodlanmış sistem başarımı üzerindeki etkisi kod çözme işlemi incelenirken ortaya çıkar. Optimum kod çözme, alınan gürültülü veya kanal hataları tarafından etkilenmiş analog simge dizilerine dayanarak, kafes içerisinde en benzer yolu araştırma işlemidir. Alınan bir simge dizisi $\underline{Y}=(y_1,y_2, \ldots\ldots\ldots,y_n)$, $\{\underline{X}_i\}$ takımı içerisinde izin verilen simge dizilerinden birine $\underline{X}_i=(X_1^i,X_2^i,\ldots\ldots\ldots,X_n^i)$ olmak üzere optimum karar kuralı esas alınarak kod çözme işlemi yapılır. Bu kurala göre kod çözme işlemi sırasında, j=k olmak üzere eğer olasılık $(\underline{Y}/\underline{X}_k)>(\underline{Y}/\underline{X}_j)$ ise $\underline{X}_k$ seçilir. Bir toplamsal Gauss gürültülü kanal için bu karar kuralı, her $\underline{X}_i$ ve $\underline{Y}$ arasındaki Öklid uzaklığını hesaplamaya ve eğer $(\underline{Y}-\underline{X}_k)^2<(\underline{Y}-\underline{X}_j)^2$ ise her j=k için $\underline{X}_k$ yı seçmeye eşdeğerdir. Diğer yönden, eğer izin verilen iki simge dizisi Öklid uzaklığı açısından birbirine yakın ise, bir dizi üzerindeki küçük gürültü ve buna benzer etkilerin neden olduğu değişiklikler kod çözme işlemi sonucunda diğer bir dizi olarak karar verilmesine neden olabilir. Eğer bütün izin verilen diziler eşit olasılıklı ise, Öklid uzaklığı açısından birbirine yakın olan diziler kodlanmış sistemin hata başarımını belirler. Böylece kodlu sistemin başarımını optimize etmek için, modülasyon işlemi bütün mümkün olan $\underline{X}_j$ ve $\underline{X}_k$ j=k dizileri arasındaki Öklid uzaklığını maksimum yapmaya çalışmalıdır.



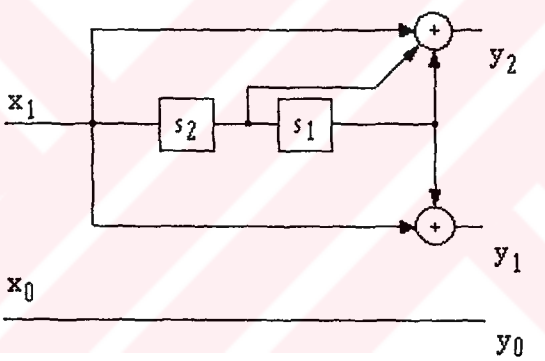
Şekil 3.4.3

Şekil 3.4.3 'de görüldüğü gibi, kodu çözülen yol simgeleri $\hat{\underline{x}}$=.....,$\hat{x}_1$,$\hat{x}_2$,$\hat{x}_3$,..... iletilen yol simgeleri $\underline{x}$=.....,x_1,x_2,x_3,..... ile her zaman aynı değildir. $\hat{\underline{x}}$ ile $\underline{x}$ arasındaki her sapma bir hata olayı olarak açıklanır. Bu tür hata olaylarının toplamsal Gauss gürültüsü altındaki oluşma oranı, bütün izin verilen mümkün simge dizi çiftleri arasındaki Öklid uzaklığına bağlıdır. Böylece işaret noktalarını kodlayıcı çıkışlarına bu minimum uzaklığı maksimum yapacak şekilde taşımak kodlu sistemin başarımını maksimum yapacaktır. Ungerboeck, bit-simge eşleme problemini araştırmıştır ve küme bölmeleme (=set partitioning) denilen metodu geliştirmiştir. Küme bölmeleme metodu üzerine dayandırılmış bit-simge eşlemesinin kuralları aşağıdaki gibi özetlenebilir :

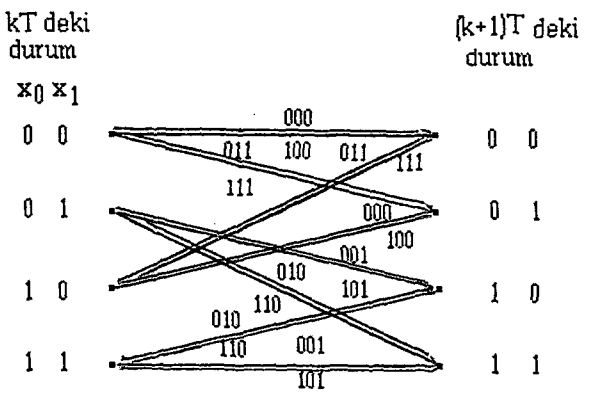
Kural 1. Kafes yapısı içerisindeki bütün paralel geçişler işaret uzayı içerisinde mümkün olan maksimum Öklid uzaklığını alır.

Kural 2. Bir kafes durumundan sapan veya o durumda çakışan bütün geçişler işaret uzayı içerisinde mümkün olan diğer maksimum uzaklığı alırlar.

Yukarıdaki fikirler, şekil 3.4.4 de görülen 2/3 oranlı, v=2 olan katlamalı kodlayıcı göz önüne alınarak gösterilecektir.



Şekil 3.4.4



Şekil 3.4.5

2/3 oranlı v=2 olan katlamalı kodlayıcının kafes diyagramına dikkat edersek, istenilen oran şekil 3.4.1 deki 1/2 oranlı katlamalı kodlayıcıya bir kodlanmamış bit ilave edilerek elde edilmiştir. Sonuç olarak elde edilen kafes diyagramı ise şekil 3.4.5 de görülmektedir. Burada paralel geçişler, kafes diyagramında en ağırlıklı bit olarak göstererilen x_0 yüzündendir. Böylece bir durumdan çıkan yukarıdaki iki dal, x_0x_1 in 00 ve 10 olmasından dolayı olan geçişlerdir. Aşağıdaki iki dal ise, X_0X_1 in 01 ve 11 olması yüzünden olan geçişleri göstermektedir. Kodlayıcı çıkışı kanal simgeleri üzerine daha önce bahsedilen kurallara uyularak eşlenebilir. Şekil 3.4.6 sekiz seviyeli bir PAM işareti için bir gösterilim sağlamaktadır. Şekil 3.4.5 deki kafes diyagramında görüldüğü gibi, bütün paralel geçişler sekiz birimlik bir mesafe ile ayrılmışlardır. Verilen bir durumdan sapan bütün dalllar dört birim mesafe ile birbirlerinden ayrılmışlardır.

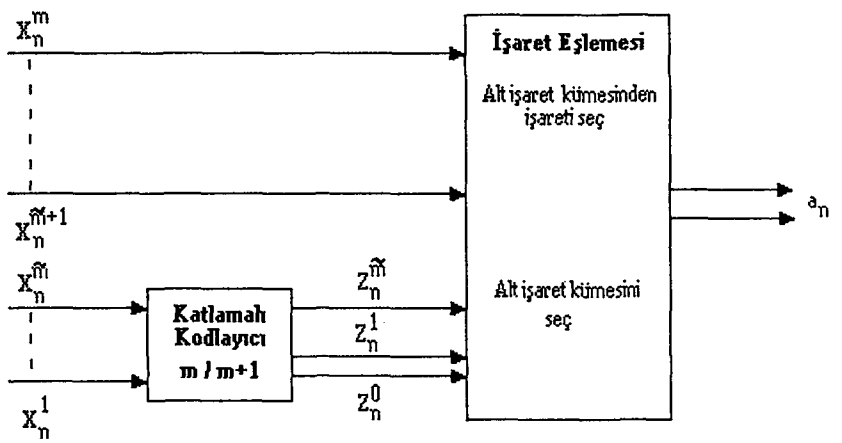
Şekil 3.4.6
8 AM işaret grubu

Şekil 3.4.7 'de görülen TCM kodlayıcı / eşleyici yapısı aşağıdaki işlemleri yerine getirir..

1. Her kodlayıcı işlevinde iletilecek olan m adet bit, $\tilde{m} \leqslant m$, $\tilde{m}/\tilde{m}+1$ oranlı katlamalı kodlayıcıdan geçirilerek $\tilde{m}+1$ kodlanmış bit elde edilir.

2. $m+1$ adet kodlanmış bit, 2^{m+1} alt küme içerisinden birini seçer.

3. Geriye kalan m-$\tilde{m}$ sayıdaki bit, seçilen alt işaret kümesi içerisindeki 2^{m-m} işaretten birini seçer.



Şekil 3.4.7
Kafes kodlamalı modülasyon için genel kodlayıcı / modülatör yapısı

3.5 Küme Bölmeleme

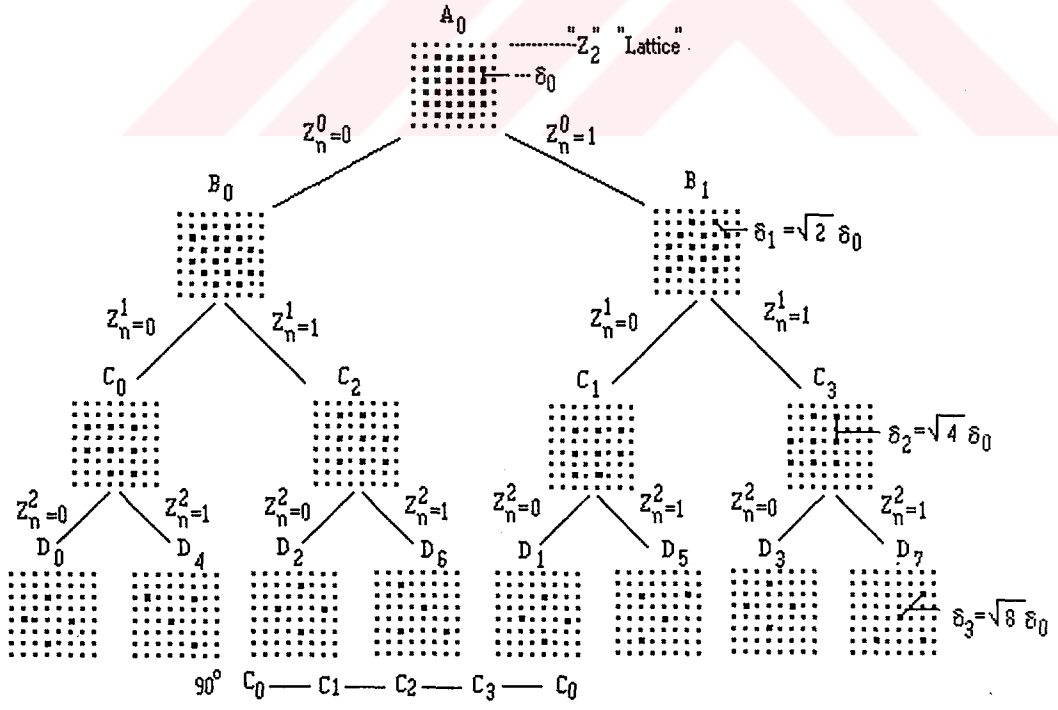
Küme bölmeleme, TCM işleyişi içerisinde merkezi bir öneme sahiptir. Şekil 3.5.1 bu konuyu 32-CR işaret kümesi için göstermektedir (Z_2 "Lattice" tipindeki işaret kümesi için). Küme bölmeleme, bir işaret kümesini en küçük kümeler arası uzaklıklar δ_i i=0,1,..... ile daha küçük alt kümelere böler. Her bölme iki yollıdur. Bölme işlemine δ_{m+1} gerçekleştirilmek istenen TCM işleyişinin, ihtiyaç duyulan serbest uzaklığına eşit veya büyük olana kadar m+1 kez devam edilir. $D_0,D_1,.....,D_7$ olarak isimlendirilen, en son elde edilen alt işaret kümeleri "alt kümeler" olarak tanımlanır. Bölmeleme ağacındaki dalların m+1 kodlanmış bit tarafından, $(Z_n^m,.....,Z_n^0)$ olarak şekil 3.5.1'deki sırada etiketlenmesi, her alt küme için Z_n ile gösterilen bir etiketin oluşmasına neden olur. Bu etiket, alt kümenin ağaç içerisindeki pozisyonunu belirtir. Eğer iki alt kümenin etiketleri en son q pozisyonda aynıysa (Z_n^q biti hariç olmak üzere) bu durumda iki alt kümenin işaretleri, paylaştırma ağacındaki q. seviyedeki aynı alt kümenin elemanlarıdır. Böylece bu işaretler en azından δ_q uzaklığına sahip olacaklardır.m-$\tilde{m}$ sayıda kodlanmamış bit, $X_n^m,.....,X_n^{\tilde{m}+1}$, seçilen alt küme içerisinden bir işareti seçmek için kullanılır. Kafes kodunda alt küme işaretleri $2^{m-\tilde{m}}$ paralel geçişlere denk düşmektedir. Böylece bir TCM kodunun serbest Öklid uzaklığı ,

$d_{free}=Min[\delta_{m+1},d_{free}(m)]$

δ_{m+1}=Paralel geçişler arasındaki minimum uzaklık

$d_{free}(m)$=Kafes diyagramında paralel olmayan geçişler arasındaki minimum uzaklık

$\tilde{m}=m$ olması özel durumunda alt kümeler sadece bir işareti kapsayacaktır ve bu nedenle kafes diyagramında paralel geçişler olmayacaktır.



Şekil 3.5.1
Küme ayrıştırma

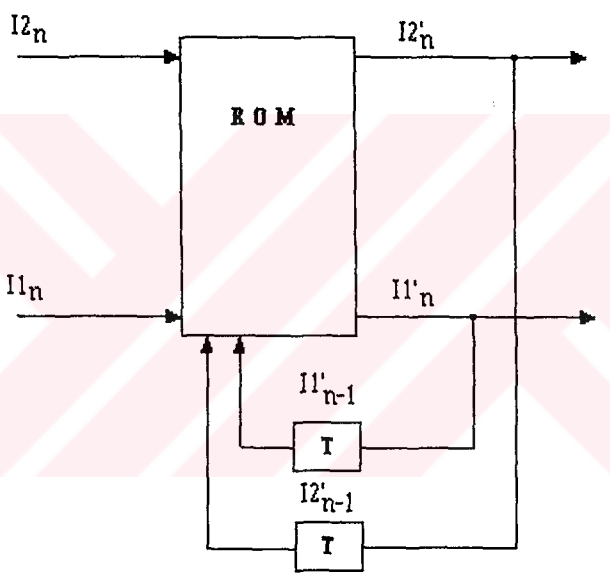
3.6 Genişletilmiş İşaret Uzayına Sahip Dönmeyle Değişmez Katlamalı Kanal Kodlaması (Doğrusal Olmayan Kodlar)

90, 180, 270 derecelik faz belirsizliklerine sahip olan işaret uzayları, sadece 180 dereceelik faz belirsizliklerine sahip olan işaret uzaylarına tercih edilir. Bunun nedenlerinden bir tanesi, bu türden işaret uzaylarının tepe/ortalama güç oranının, sadece 180 derece faz belirsizliğine sahip olan işaret uzaylarınınkine göre daha küçük olmasıdır.

3.6.1 90 Derece Dönmeyle Değişmez Katlamalı Kodlar

Farksal kodlayıcı ve kod çözücü

Şekil 3.6.1.1 de görülen $I1_n$ ve $I2_n$ veri bitleri her işaretleşme aralığında $I1'_n$ ve $I2'_n$ olarak farksal kodlama işleminden geçirilmektedir (Tablo 2.5.2.1 'e göre).



Şekil 3.6.1.1
Farksal Kodlayıcı

Farksal kodlama işlemi, bir önceki farksal olarak kodlanmış çıkış bit çifit $I2'_{n-1}$ $I1'_{n-1}$ 'in farksal kodlayıcı giriş bitleri $I2_n$ ve $I1_n$ ile uyumlu olarak döndürülmesidir. Dönmenin anlamına yaklaşmak için bir önceki farksal kodlayıcı çıkış bit çiftinin dört değer alabileceğini göz önüne alalım. Bu dört değer 00,01,10,11 dizisini oluşturacak şekilde düzenlenmiştir. Bu dizideki bir eleman, 0, 1, 2, ve 3 konum döndürülebilir. Örneğin ikinci eleman olan 01, değeri değiştirilmediği zaman 0 konum döndürülmüş olur. Bu eleman değeri 10, 11, 00 olduğu zaman sırasıyla 1, 2, 3 konum döndürülmüş olur. İlave olarak farksal kodlayıcının o andaki giriş bit çiftleri $I2_n$ ve $I1_n$ 'in mümkün olan dört bit çifti 00, 01, 10, 11 'in her birine özel bir konum değiştirme işlevi eşlenmiştir. Bu örnekte, 00, 01, 10, 11 giriş bit çiftleri sırasıyla 0, 1, 2, ve 3 konum değişikliklerine denk düşmektedir. Bir farksal kod çözücü,

Önceki Çıkış		Şimdiki Giriş		Şimdiki Çıkış	
$I2'_{n-1}$	$I1'_{n-1}$	$I2_n$	$I1_n$	$I2'_n$	$I1'_n$
0	0	0	0	0	0
0	1	0	0	0	1
1	0	0	0	1	0
1	1	0	0	1	1
0	0	0	1	0	1
0	1	0	1	1	0
1	0	0	1	1	1
1	1	0	1	0	0
0	0	1	0	1	0
0	1	1	0	1	1
1	0	1	0	0	0
1	1	1	0	0	1
0	0	1	1	1	1
0	1	1	1	0	0
1	0	1	1	0	1
1	1	1	1	1	0

Tablo 3.6.1.1

Önceki Giriş		Şimdiki Giriş		Şimdiki Çıkış	
$I2'_{n-1}$	$I1'_{n-1}$	$I2'_n$	$I1'_n$	$I2_n$	$I1_n$
0	0	0	0	0	0
0	1	0	1	0	0
1	0	1	0	0	0
1	1	1	1	0	0
0	0	0	1	0	1
0	1	1	0	0	1
1	0	1	1	0	1
1	1	0	0	0	1
0	0	1	0	1	0
0	1	1	1	1	0
1	0	0	0	1	0
1	1	0	1	1	0
0	0	1	1	1	1
0	1	0	0	1	1
1	0	0	1	1	1
1	1	1	0	1	1

Tablo 3.6.1.2

şimdiki ve önceki farksal kodlayıcı çıkış bit çiftleri $I2'_n$ $I1'_n$ ve $I2'_{n-1}$ $I1'_{n-1}$ 'in dört elemanlı dizi içerisindeki pozisyon farkına bakarak gerçek $I2_n$ $I1_n$ giriş bit çiftini yeniden elde eder. Bu şekilde bir kodlama ve kod çözme işleminin istenen bir özelliği, farksal kodlayıcı çıkışındaki tüm bit çiftlerinin dönmesine karşı bağışık olmasıdır. Tablo 3.6.1.1 de farksal kodlayıcının ve 3.6.1.2 de ise farksal kod çözücünün doğruluk tablosu görülmektedir.

Bu tablolarda bir önceki kodlayıcı çıkış bit dizisi 00,01,10,11 olarak kabul edildi. Kodlayıcı giriş bitlerine atanan dönmeler ise sırasıyla ;

00 ----------> 0 10 ----------> 2

01 ----------> 1 11----------> 3

olarak belirlendi.

V.32 ve V.33 tavsiyesi modemlerde kullanılan 90, 180, 270 derecelik faz belirsizliklerine karşı dönmeyle değişmez olan, geri beslemeli ve doğrusal olmayan katlamalı kodlayıcı ve farksal kodlayıcı yapıları, gerekli kurallar ile birlikte aşağıda belirtilecektir. Bu bölüm C1 kuralı ile başlayacaktır. Kural C1 'in sağlanması, bu kurala uyan bir durum geçiş diyagramının yardımıyla gerçekleştirilen geri beslemeli kodun, 90 derecelik işaret elemanı dönmelerine saydam hale gelmesini sağlar.

90 Derece dönmeyle değişmez kodlar için C1 kuralı

q durumlu bir kodlayıcının durumunu i ile gösterelim (i=0,1,.....,q-1). Bu durumda kod, aşağıdaki tanımı doğruluyacak şekilde $f_l\{0,1,.....,q-1\}$----------$>\{0,1,.....,q-1\}$ l=0,1,2,..... birebir fonksiyonu mevcut ise, 90, 180 ve 270 derecelik işaret elemanı dönmelerine saydam hale getirilebilir. i durumundan j durumuna olan her geçiş için C{0,1,.....,q-1} olmak üzere, U bu durum geçişine karşı gelen işaret elemanları grubu olsun ve V1, V2, V3 'de U işaret elemanları kümesi 90, 180, 270 derece döndürüldüğünde elde edilen işaret elemanları kümesi olsun. Bu durumda $f_l(i)$ durumundan $f_l(j)$ durumuna geçişle ilgili işaret elemanları kümesi V1, V2, V3 olacaktır. Eğer kural C1 sağlanırsa, her Z geçerli işaret elemanı dizisi için, dizi içerisindeki bütün elemanların 90, 180 veya 270 derece döndürülmesiyle oluşan işaret elemanları dizisinde geçerli bir dizidir.

Şimdi 2^p kodlayıcı durumuna sahip geri beslemeli katlamalı kodlayıcıları düşünelim. Burada p pozitif bir tamsayıdır. Bu durumda katlamalı kodlayıcının şimdiki durumu $W1_n,W2_n,.....,Wp_n$ olarak gösterilebilir. Burada WI_n, I=1,2,.....,P olmak üzere sayısal bir değişkendir. Bu sınıftaki katlamalı kodlar ;

1. Kodlayıcının çıkış bit kombinasyonları işaret elemanlarına atanarak,
2. Bu eşitlikten kodlayıcının doğruluk tablosunu veya mantık devresini bularak,
3. Durum geçiş diyagramı saptanarak gerçeklenebilir.

m/m+1 oranlı katlamalı kodlayıcının şimdiki çıkış bitleri, $Y0_n,Y1_n,.....,Ym_n$ olsun. Burada $Y1_n,Y2_n,.....,Ym_n$, kodlayıcı girişinden direkt olarak taşınan kodlayıcı giriş bitleri olsun. Genişletilmiş işaret uzayının 2^{k+1} sayıda işaret elemanları kümelerine ayrılmış olduğunu kabul edelim.

k birden büyük bir tamsayı olsun. Şimdi, $\{Y1_n, Y2_n, \ldots\ldots, Ym_n\}$ kümesi içerisinden rastgele k kodlayıcı çıkış biti seçelim. Genellemeden birşey kaybetmeden, seçilen bitlerin YI_n , $I=1,2,\ldots\ldots,k$ bitleri olduğunu kabul edelim. Bu durumda kodlayıcı çıkış bit kombinasyonlarının işaret elemanlarına atanması aşağıdaki kuralları sağlamalıdır.

90 Derece dönmeyle değişmez geri beslemeli katlamalı kod için C2 kuralı

2^{k+1} grup içerisindeki işaret elemanları aynı YI_n, $I=0,1,\ldots\ldots,k$ bitlerine sahip olmalıdır.

90 Derece dönmeyle değişmez katlamalı kod için C3 kuralı

Aynı kodlayıcı durumundan çıkan geçişlerle ilgili işaret elemanları kümelerine atanan YI_n bitlerinin $I=1,2,\ldots,k$ bütün değişik bit kombinasyonları, YI_n $I=1,2,\ldots\ldots,k$ bitlerinin bütün mümkün olan kombinasyon değerlerini kapsamalıdır.

90 Derece dönmeyle değişmez katlamalı kodlar için C4 kuralı

YI_n, $I=1,2,\ldots\ldots,k$ bitleri içerisinden iki kodlayıcı biti seçelim. Bunlar $Y1_n$ ve $Y2_n$ olsun. Bu durumda, aynı yarıçapa sahip fakat birbirlerinden 90 derece ayrı olan her dört işaret elemanı kümesi için $Y2_n$ ve $Y1_n$ bit çiftleri (bu dört işaret elemanı ile ilgili) birbirinden farklı olmalıdır.

Yukarıda tanımlanan, dört işaret elemanı içeren her küme için, $Y2_n$ $Y1_n$ bit çiftinin dört değeri dört elemandan oluşan bir diziye atanır. Bu dizinin ikinci, üçüncü ve dördüncü değerine karşı gelen işaret elemanları, dizinin ilk değerine karşı gelen işaret elemanının sırayla 90, 180, 270 derece döndürülmesiyle elde edilebilmelidir..

Kodlayıcı çıkış bitleri YI_n $I=3,4,\ldots\ldots,m$ 'in işaret elemanlarına eşlenmesinde değişik yollar izlenebilir. Örneğin, YI_n 'in $I=3,4,\ldots\ldots,m$ olmak üzere aynı değerleri, yukarıda belirtilen aynı yarıçapa sahip fakat birbirlerinden 90 derece ayrı dört işaret elemanına atanabilir.

Kodlayıcı çıkış bitlerinin işaret elemanlarına atanması işlemi bittikten sonra, mantık devre diyagramı ve doğruluk tablosu bu eşlemeler ve geçiş diyagramlarından elde edilir.

Burada doğruluk tablosunun giriş büyüklükleri, kodlayıcı durum değişkenleri olan $W1_n$, $W2_n$, $W3_n$ ile ilgili kodlayıcı çıkış bitleri $Y1_n$, $Y2_n$, ve Yk_n olan kodlayıcı giriş bitleridir. Çıkış büyüklükleri ise kodlayıcının bir sonraki durum değişkenleri olan $W1_{n+1}$, $W2_{n+1}$, $W3_{n+1}$ ve $Y0_n$ kodlayıcı çıkış bitidir.

Genişletilmiş işaret uzayının 90, 180 ve 270 derecelik faz belirsizlikleri, ilgili kodlayıcı çıkış bit çifti $Y1_n$ ve $Y2_n$ olan kodlayıcı giriş bit çiftleri katlamalı kodlayıcıya uygulanmadan önce farksal olarak kodlanarak ortadan kaldırılabilir. Katlamalı kod çözücünün çıkışında tam tersi işlem uygulanır.

Farksal kodlayıcının giriş ve çıkış bit çiftlerini $I2_n$ $I1_n$ ve $I2'_n$ $I1'_n$ ile gösterelim. Burada $I1'_n$ ve $I2'_n$ katlamalı kodlayıcının giriş bitleridir ve ilgili çıkış bitleride $Y2_n$ ve $Y1_n$ dir.

90 Derece dönmeyle değişmez geribeslemeli katlamalı kodlar için (90, 180 ve 270 derecelik faz

belisizliklerine sahip genişletilmiş işaret uzaylı) genel tasarım aşamaları şöyledir :

ADIM 1 :

Şimdiki kodlayıcı durumunu, $W1_n, W2_n, \ldots\ldots Wp_n$ ile gösterelim. Burada WI_n $I=1,2,\ldots\ldots,p$ olmak üzere sayısal bir değişken olsun. $m/m+1$ oranındaki katlamalı kodlayıcı, bütün kodlayıcı giriş bitleri direkt olarak kodlayıcı çıkışına taşınacak şekilde tasarlanır. Şimdiki kodlayıcı çıkış bitlerini $Y0_n, Y1_n, \ldots\ldots, Ym_n$ ile gösterelim. Bu çıkış bitleri arasındaki en son m bit direkt olarak kodlayıcı çıkışına taşınan bitlerdir.

ADIM 2 :

2^{m+1} işaret elemanı, k birden büyük bir tamsayı olmak üzere, "küme bölmeleme" kuralına göre 2^{k+1} alt kümeye bölünür.

ADIM 3 :

Kodlayıcı durum geçişleri seçilir ve adım 2 de elde edilen işaret elemanları, kodlayıcı durum geçişlerine atanır. Seçme ve atama işlemi, bütün geçerli işaret dizileri arasındaki minimum Öklid uzaklığı maksimize edilecek ve C1 kuralı sağlanacak şekilde yapılır.

ADIM 4 :

Kodlayıcı çıkış bitleri, C2, C3, C4 kuralları sağlanacak şekilde işaret elemanlarına atanır.

ADIM 5 :

Adım 3 de bulunan geçiş diyagramlarından katlamalı kodlayıcının mantık diygramı veya eşdeğer doğruluk tablosu bulunur.

ADIM 6 :

Giriş bit çifti $I2_n$ $I1_n$ ve çıkış bit çifti $I2'_n$ $I1'_n$ ile gösterilen bir farksal kodlayıcı seçilir.

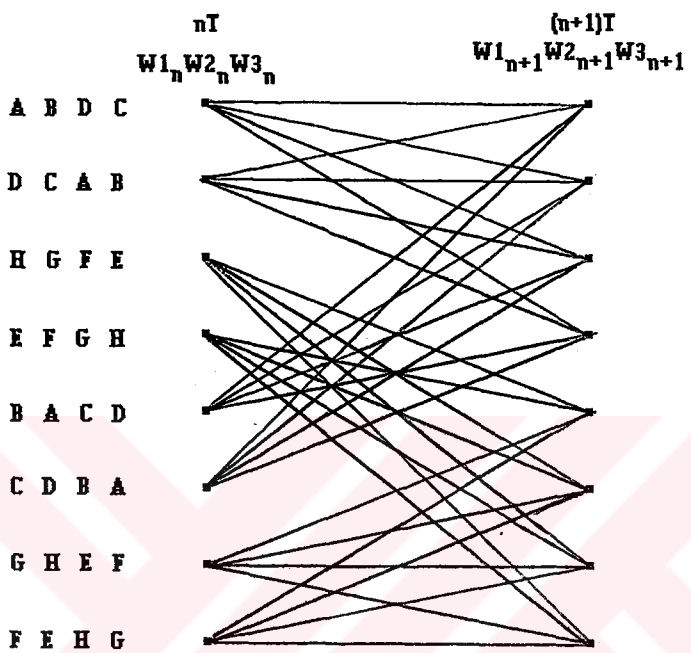
ADIM 7 :

Adım 6 da seçilen farksal kodlama işlemi, kodlayıcı giriş bit çiftine uygulanır. Katlamalı kodlayıcı çıkışındada tam tersi işlem uygulanır.

3.6.2 32-CR ve 128-CR genişletilmiş işaret uzayına sahip 90 derece dönmeyle değişmez katlamalı kodlar

Şekil 2.5.2.1 de görülen katlamalı kodlayıcının şimdiki durumunu $W1_n$, $W2_n$, $W3_n$, ($W1_n$, $W2_n$, $W3_n$ kodlayıcı giriş bitleri olmayan sayısal değişkenler olmak üzere), ile gösterelim. Genişletilmiş

32-CR işaret uzayı A, B, C, D, E, F ve G olarak adlandırılan sekiz alt kümeye ayrılmıştır.Bu kodlayıcıya ait durum geçiş diyagramı şekil 3.6.2.1 de görülmektedir. Bu durum geçiş diyagramı C1 kuralını sağlamaktadır. Çünkü burada aşağıdaki durumu sağlayan üç tane birebir fonksiyon bulunmaktadır. Bu tanımlama şu şekildedir; $f_l(W1_n,W2_n,W3_n)$ durumundan $f_l(W1_{n+1},W2_{n+1},W3_{n+1})$ $l=1,2,3,$ geçişlerle ilgili işaret elemanları kümeleri, $W1_n$, $W2_n$, $W3_n$ durumundan $W1_{n+1}$, $W2_{n+1}$, $W3_{n+1}$ durumuna geçişlerle ilgili işaret elemanları kümelerinden tüm işaret



Şekil 3.6.2.1

Durum geçiş diyagramı

elemanlarının saat yönünde sırasıyla 90, 180, 270 derece döndürülmesiyle elde edilir.

$Q6_n$, $Q5_n$, $Q4_n$, $Q3_n$, $Y2_n$, $Y1_n$, $Y0_n$ kodlayıcı çıkış bitlerinin işaret uzayındaki işaret elemanlarına eşlenmesi C2, C3, C4 kurallarını sağlar. Özellikle, $Y2_n$ $Y1_n$ bit çiftinin C4 de tanımlanan dört elemanlı değerler dizisi, 00,01,10,11 olarak tanımlanmıştır. Katlamalı kodlayıcının doğruluk tablosu tablo 3.6.2.1 de verilmiştir. 128-CR işaret uzayına sahip katlamalı kodlayıcı yapısı ile 32-CR işaret uzayına sahip kodlayıcı yapısı benzerdir. Ancak, 128-CR işaret uzayına sahip katlamalı kodlayıcıda, her 2^{k+1} işaret elemanı kümesi içerisinde 16 adet işaret elemanı bulunmaktadır.

Genişletilmiş işaret uzayının sadece 180 derece faz belirsizliğine sahip olduğu durumdan farklı olarak, 90, 180, 270 derece faz belirsizliklerine sahip olan genişletilmiş işaret uzaylı geri beslemeli kodda, farksal kodlama ve kod çözme uygulamaksızın kodu bu faz dönmelerinden etkilenmeyecek hale getirmek mümkün değildir.

128-CR işaret uzayına sahip 90 derece dönmeyle değişmez katlamalı kod, 8*8 QAM işaret

uzayına sahip kodsuz sisteme göre 4.1 dB 'lik bir kodlama kazancı sağlamaktadır. Kodlanmamış sistemlere göre genişletilmiş işaret uzayına sahip kodlanmış sistemlerin sağladığı kazanç aşağıdaki ifade ile bulunmaktadır.

Giriş					Çıkış			
$W1_n$	$W2_n$	$W3_n$	$Y2_n$	$Y1_n$	$W1_{n+1}$	$W2_{n+1}$	$W3_{n+1}$	$Y0_n$
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1	0
0	0	0	1	1	0	1	0	0
0	0	0	1	0	0	1	1	0
0	0	1	1	1	0	0	0	0
0	0	1	1	0	0	0	1	0
0	0	1	0	0	0	1	0	0
0	0	1	0	1	0	1	1	0
0	1	0	0	0	1	0	0	1
0	1	0	1	1	1	0	1	1
0	1	0	1	0	1	1	0	1
0	1	0	0	1	1	1	1	1
0	1	1	0	1	1	0	0	1
0	1	1	1	0	1	0	1	1
0	1	1	1	1	1	1	0	1
0	1	1	0	0	1	1	1	1
1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0
1	0	0	1	0	0	1	0	0
1	0	0	1	1	0	1	1	0
1	0	1	1	0	0	0	0	0
1	0	1	1	1	0	0	1	0
1	0	1	0	1	0	1	0	0
1	0	1	0	0	0	1	1	0
1	1	0	1	1	1	0	0	1
1	1	0	0	0	1	0	1	1
1	1	0	0	1	1	1	0	1
1	1	0	1	0	1	1	1	1
1	1	1	1	0	1	0	0	1
1	1	1	0	1	1	0	1	1
1	1	1	0	0	1	1	0	1
1	1	1	1	1	1	1	1	1

Tablo 3.6.2.1

Katlamalı kodlayıcı doğruluk tablosu

$$10*\log_{10}\left[\left(\frac{d_{min}^2}{P_{av}}\right)_{kodlanmış} \Big/ \left(\frac{d_{min}^2}{P_{av}}\right)_{kodlanmamış}\right]$$

Burada d_{min}^2 kodun karesel serbest Öklid uzaklığı, P_{av} ise ortalama işaret gücüdür. Bu ifade yardımıyla 128-CR genişletilmiş işaret uzayına sahip, 90, 180, 270 derece dönmeyle değişmez kafes kodunun, 8*8 işaret uzayına sahip klasik katlamasız koda göre kazancını şöyle bulabiliriz.

Kodsuz 8*8 QAM sisteminin ortalama işaret gücü ve minimum karesel Öklid uzaklığı, $Pav_{kodsuz}=42W$, $d^2min=4$ olarak bulunabilir. Kodlu sistemde ise $Pav_{kodlu}=41$ ve $d^2min=10$ olarak işaret uzaylarından kolayca bulunabilir. Bu değerler yukarıdaki formülde yerine konursa, kodlanmış sistemin, eşdeğer kodsuz sisteme göre kazancı 4.1 dB olarak bulunacaktır.

3.7 Viterbi Kod Çözücüsü

Viterbi algoritması, önceden belirlenmiş ölçüde alınan simge dizisine en yakın olan simge dizisini bulan en büyük benzerlikli dizi tahmin metodudur. Bu rekursif işlem, kalan düğüm (survivor) olarak adlandırılan ve kafesin her durumuna giren en kısa yolun k anında tutulmasıdır.k+1 zamanına ilerlemek için, bütün k zamanı kalan düğümleri genişletilen yol parçalarının metrikleri hesaplanarak genişletilir. Her durumdaki arttırılmış yolların metrikleri karşılaştırılır ve bunların en kısa olanı saklanır. Bu k+1 anındaki kalan düğümü gösteren en kısa yol olup tutulur. Aynı işlem k+2 anı içinde yapılır ve işleme devam edilir. Dikkat edilecek nokta kalan düğümlerin sayısının asla kafes yapısı içerisindeki durumları aşamayacağıdır. Bu açıklamaya dayanarak, yumuşak karar vermeli Viterbi kod çözücüsü için gerekli olan işlemler aşağıda verilmiştir :

1. Toplamsal Gauss gürültülü bir kanal için $(r-x_i)^2$ ile orantılı olan dal metriklerinin hesaplanması. Burada r alınan örnek ve x_i gürültüden bağımsız olarak kafes içerisindeki bir geçişle ilgili olan simgedir.

2. Arttırılmış yol metriklerinin bulunması için dal metrikleri ve kalan düğümlerin birbiriyle toplanması

3. Arttırılmış yol metrikleri arasındaki, her durumdaki kalan düğümü belirlemek için yapılan karşılaştırma

4. Alınan işaret sayısı v (kod çözme derinliği) ye ulaştığında, tüm kafes durumlarına ait metriklerin karşılaştırılarak en küçük metrik değerinin bulunması ve bu metriğe ait durumdan birer birer geriye giderek geçiş elemanlarının belirlenmesi. Bu elemanların oluşturduğu dizi bu algoritmaya göre karar verilen en uygun gönderilen işaret dizisidir.

BÖLÜM 4 DENGELEYİCİLER

Bilgisayar haberleşmesine olan ihtiyacın ani yükselişi, çok geniş bir şebeke oluşturan ve ses haberleşmesi için geliştirilmiş olan ses bandlı kanallar üzerinde yüksek hızlı veri iletimi ile karşılanmaya çalışılmıştır.

Analog kanallar, giriş dalga şekillerinin bozulmuş veya değiştirilmiş biçimlerini iletirler. Dalga biçiminin bozulması toplamsal veya çarpımsal olabilir. Bu bozulmanın nedeni ise ısıl gürültü, impuls gürültüsü ve zayıflamalardır. Kanal tarafından gerçekleştirilen değişiklikler ise, frekans çevrimi, doğrusal olmayan veya harmonik gürültü ve zaman saçılmasıdır (time dispersion) dır.

Telefon hatlarında kanal frekans cevabı, sabit genlik ve doğrusal faz (sabit gecikme) özelliğinden kayma gösterirse zaman saçılması ortaya çıkar.

Dengeleme bütün bu ideal olmayan karakteristikleri süzme işleminden geçirerek yok etmeye çalışır.

Bir eşzamanlı modem vericisi, belirli bir süre içerisinde sabit sayıdaki veri bitlerini toplar ve işaretleşme hızında göndermek üzere simge grupları olarak kodlar. Darbe genlik modülasyonunda (PAM) her işaret, genlik seviyesi bu simgeler tarafından belirlenen bir darbedir.

Band verimlilikli sayısal haberleşme sistemlerinde, bir zaman saçılmalı kanal üzerinden iletilen her simgenin etkisi, bu simgeyi temsil etmek için kullanılan zaman aralığının dışına taşar. Bunun sonucu olarak alınan simgelerde oluşan katlanmanın neden olduğu bozulma, simgelerarası girişim (intersymbol interference-ISI) olarak adlandırılır. Bu bozulma, gürültülü, sınırlı band genişlikli kanallar üzerinde güvenilir yüksek hızlı veri iletimine en büyük engellerden biridir.

Birçok pratik durumda kanal karakteristikleri önceden bilinmemektedir. Orta hızlı modemler için genellikle bir istatistiksel dengeleyicinin (compromise equalizer) tasarlanması ve kullanılması yeterlidir. Bu dengeleyici, tahmin edilen kanal genlik ve gecikme karakteristiklerinin ortalama bir değerini düzeltmek etmek üzere tasarlanmıştır.

Bununla birlikte, anahtarlamalı telefon şebekesi içerisindeki hatlarda olduğu gibi, belirli bir sınıf içerisindeki kanallarda bu karakteristiklerdeki değişme çok büyüktür. Bu nedenle genel olarak, 2400 bps 'den daha yüksek hızlarda otomatik uyarlamalı dengeleme kullanılır.

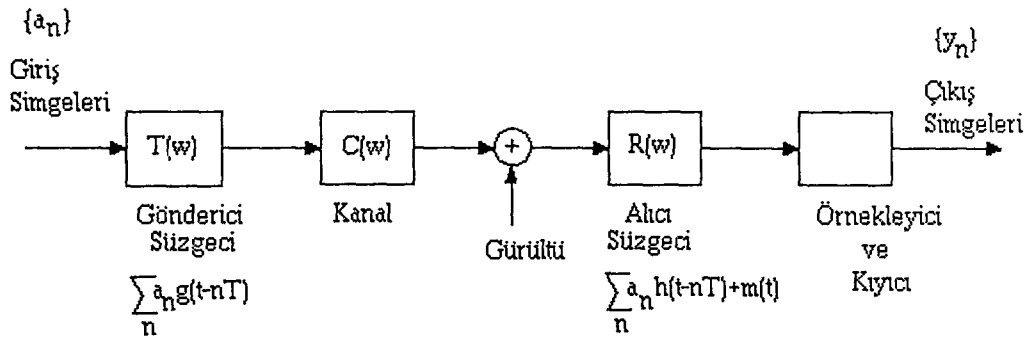
4.1 Simgelerarası Girişim Problemi

Simgelerarası girişim bütün sayısal modülasyonlu sistemlerde ortaya çıkar. Bununla birlikte etkisi en kolay biçimde, bir temel band darbe genlik modülasyon sistemi için gösterilebilir. Şekil 4.1.1 de kodlayıcı ve kodçözücü kısımları olmayan basit bir PAM sistemi görülmektedir.

Verici süzgeci, kanal ve alıcı süzgeci bilgi taşıyan darbelerin şeklini tayin eder.

$$h(t) = 1/2\pi \cdot \int T(\omega).C(\omega).R(\omega).e^{j\omega t}.d\omega$$

Bu durumda çıkış ;



Şekil 4.1.1
Darbe Genlik Modülasyonu Sistemi

$$y(t)=\sum_i a(i).h(t-iT)+\eta(t)$$

olacaktır. Burada, T örnekleme zamanı ve $\eta(t)$ ise toplamsal Gauss gürültüsüdür. $t=kT+t_0$ anında (t_0 verici-kanal-alıcı ard arda bağlı devresi içerisinde ortaya çıkan sabit gecikmedir. İstenen a(k) çıkışına karşı alıcı çıkışındaki örneklenmiş işaret ;

$$y(kT+t_0)=\sum a(i).h(kT+t_0-iT)+\eta(kT+t_0)$$

olacaktır. Bu ifadelerde k nın $kT+t_0$ 'ı gösterdiğini farzedelim. Bu durumda yukarıdaki ifade;

$$y(k)=\sum_i a(i).h(k-iT)+\eta(k)$$

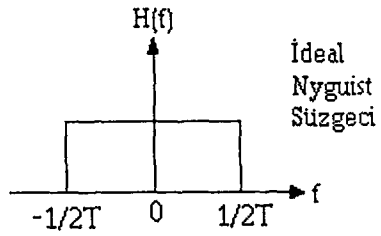
olacaktır. Dikkat edilirse gerçek çıkışın aşağıdaki şekildede ifade edilebileceği görülür.

$$y(k)=h(0).\{a(k)+1/h(0).\sum_{\substack{i \\ i\neq k}} a(i).h(k-iT)+\eta(k)/h(0)\}$$

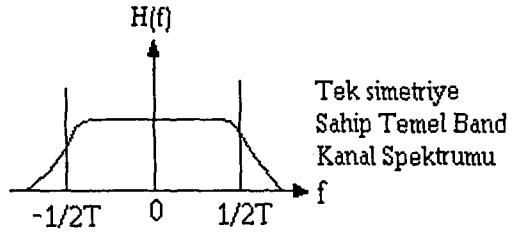
h(0) genlik faktörü birim değeri olarak seçilebilir. Bu durumda birinci değer istenen çıkış olup, ikinci ifade simgelerarası girişimi temsil etmektedir. Sonuncu terim ise gürültüyü temsil etmektedir. Bu ilave terimlerin toplamı kuantalama aralığını aşarsa bir hata meydana gelecektir. Her girişim terimi, kanal impuls yanıtının bir örneği olan $h(t_0+iT)$ ile orantılıdır. Simgelerarası girişim sadece $h(t_0+iT)=0$ $i=0$ ise ortadan kalkacaktır. Buda, kanal impuls yanıtının T aralıklı sıfır geçişlere sahip olmasıyla mümkündür. İmpuls yanıtının bu şekilde üniform sıfır geçişlerine sahip olması Nyquist 'in birinci kriterini sağlaması demektir. Frekans domeni terimleri ile bu durum aşağıdaki ifadeye eşittir.

$$H'(f)=\sum_n H(f-n/T)=\text{sabit} \quad |f| \leq 1/2T$$

Burada H(f) kanal ferakans cevabı ve H'(f) katlanmış kanal spektral cevabıdır. $|f| < 1/2T$ genel olarak Nyquist veya minimum band genişliği olarak bilinir. Şekil 4.1.2 ve 4.1.3 de iki tane doğrusal fazlı alçak geçiren süzgecin genlik yanıtları görülmektedir. Bunlardan birtanesi, Nyquist band genişlikli ideal süzgeç olup, diğeri 1/2T Hz etrafında tek simetriye sahiptir.



Şekil 4.1.2



Şekil 4.1.3

Pratikte çok olarak kullanılan ve literatürde sıkça rastlanan doğrusal fazlı süzgecin bir sınıfı yükseltilmiş kosinüs süzgeci olarak adlandırılır. Bu süzgeç 1/2T Hz etrafında kosinüs yuvarlanmaya sahiptir. Aşağıdaki eşitlikte yükseltilmiş kosinüs fonksiyonu verilmiştir.

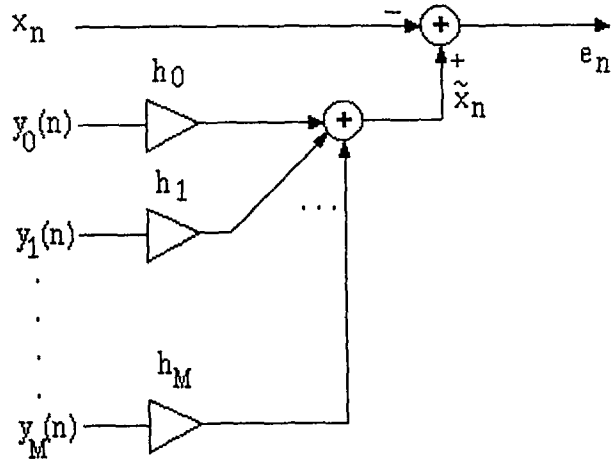
$$H(f)=\begin{cases} 1 & ; \quad 0<|f|<(1-\alpha)/2T \\ 1/2\left[1-\mathrm{Sin}\left(\frac{\pi T}{2\alpha}(2f-1/T)\right)\right] & ; \quad (1-\alpha)/2T<|f|<(1+\alpha)/2T \\ 0 & ; \quad |f|>(1+\alpha)/2T \end{cases}$$

Alınan işaretin yolu üzerine yerleştirilmiş olan dengeleyicinin amacı, ISI 'yı doğru kararlar verme olasılığını maksimum yapmak için mümkün olduğu kadar azaltmaktır.

4.2 Dengeleme Stratejileri

Dengeleme işlemi için kullanılan temel devre yapısı "Uyarlamalı Süzgeç" tir. Uyarlamalı süzgeçler, işaretin bilinmediği veya karakteristiklerinin zamanla değiştiği durumlarda kullanılırlar. Bu türden süzgeçlerin temel elemanı şekil 4.2.1 de görülen ağırlıklı toplayıcı (Adaptive Lineer Combiner) dır.

Ağırlıklı toplayıcıda bir tek esas x_n işareti ve çok sayıda $y_m(n)$ ikincil işaretleri $m=0,1,2,.....M$ yer almaktadır. Burada $(M+1)$ sayıdaki ikincil işaret, uygun $(h_0,h_1,.....,h_M)$ ağırlıkları ile ağırlaştırılarak , x_n tahminini elde etmek üzere doğrusal olarak birleştirilir. Bu durumda ;



Şekil 4.2.1

Doğrusal Toplayıcı

$$x_n = h_0.y_0(n) + h_1.y_1(n) + \ldots\ldots\ldots + h_M.y_M(n)$$

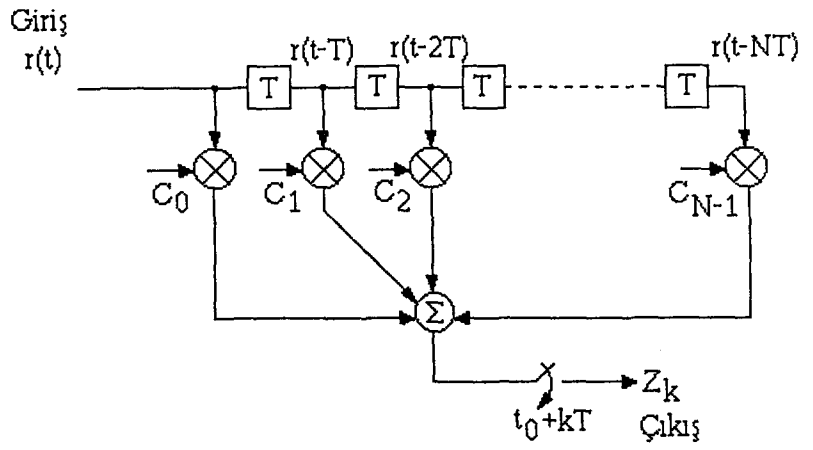
$$= [h_0, h_1, \ldots\ldots\ldots h_M] \cdot \begin{bmatrix} y_0(n) \\ y_1(n) \\ \vdots \\ y_M(n) \end{bmatrix}$$

$= h^T.y(n)$ olacaktır. Bu durumda ağırlıklı toplayıcının n. çıkış büyüklüğü ;

$$\tilde{x}_n = \sum_{i=1}^{M} h_i.y_i(n)$$

olacaktır. Eğer n anında çıkışta elde edilmesi istenen işaretin değeri biliniyorsa ve bu x_n ise, x_n ve $\tilde{x}_n$ karşılaştırılarak bir e_n hatası üretilir.Bu hata, $e_n = \tilde{x}_n - x_n$ olacaktır. Eğer h_i katsayıları e_n hatasını minimuma indirecek şekilde ayarlanırsa, ağırlıklı toplayıcı $y_i(n)$ girişlerinden, istenilen x_n çıkışlarının elde edilebileceği hale getirilmiş olur.$y_i(n)$ girişlerinin seçilme şekli uyarlamalı süzgecin türünü belirler. Genelde $y_i(n)$ işaretleri, aynı giriş işaretinden belirli bir T süresinin tam katlarında gecikmelerle alınmış örnekler olarak seçilir. Sayısal iletişim sistemlerinde kullanılan uyarlamalı dengeleyiciler de temelde böyle bir süzgeçtirler. Böyle bir süzgecin katsayıları dengeleyicinin impuls yanıtını belirler. Dengeleme için kullanılan birçok yapı içerisinde en basit olanı şekil 4.2.2 de görülen transversal (tapped delay line veya non-recursive) dengeleyicidir.

Bu şekilde bir dengeleyicide alınan işaretin o andaki ve geçmiş değerleri r(t-nT), dengeleyici sabitleri, C_n, ile doğrusal olarak ağırlaştırılır ve çıkışı elde etmek üzere toplanır. Eğer gecikmeler ve vurum kazançları analog ise, dengeleyicinin sürekli çıkışı olan Z(t) simge hızında örneklenir ve örnekler karar aygıtına gönderilir.



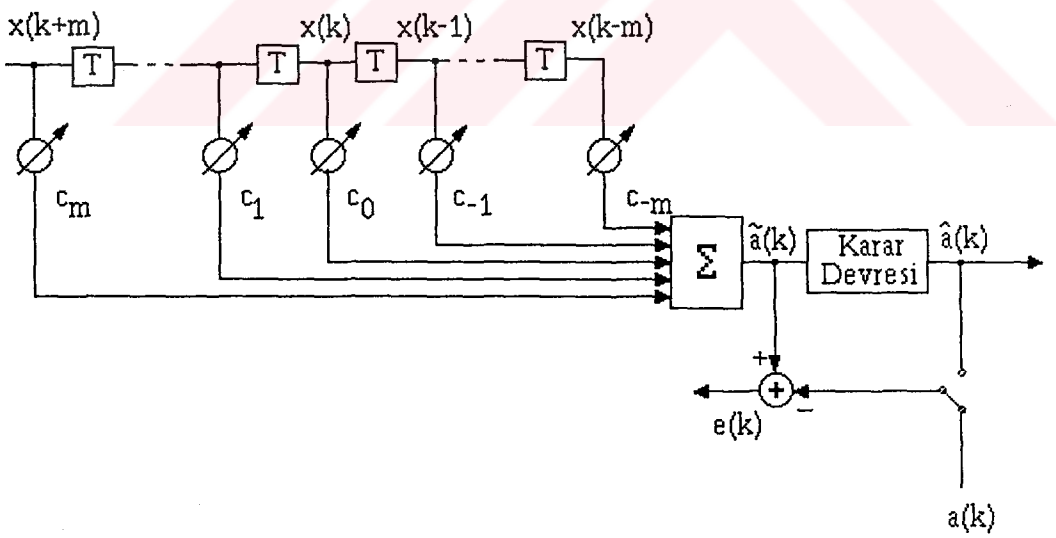
Şekil 4.2.2

Doğrusal Transversal Dengeleyici

Dengeleyici çıkış örnekleri $Z(t_0+kT)$ veya Z_k 'lar sayısal olarak hesaplanır.Buna göre ;

$$Z_k=\sum_{n=0}^{N-1} C_n r(t_0+kT-nT)$$

Burada N dengeleyici katsayılarının sayısıdır ve t_0 ise örnekleyici fazıdır. Şekil 4.2.3 'de temel dengeleyici blok diyagramı görülmektedir.



Şekil 4.2.3

Temel Dengeleyici Blok Diyagramı

1/T (baudrate) modülasyon hızında iletim yapan bir sistem düşünelim. Bu sistemin alıcısına gelen bozulmuş x işaretinin dengelenmesl gereklidir. Bunun için x işaretinden T simge aralığı kadar gecikmelerle alınan örnekler 2m+1 katlı bir süzgece verilir. Belli bir k anı için x işaretinin

$i=0,+1,+2,....,+m$ olmak üzere $(k+i)$. örnekleri süzgeçte yer alır. Daha sonra bu örneklere c_i katsayıları vasıtasıyla ağırlık kazandırılır. Süzgeç çıkışında x(k) örneğine karşılık vericiden yollanmış olan a(k) simgesine ilişkin a(k) kestirimi elde edilecektir.

$$\tilde{a}(k)=\sum_{i=-m}^{m} c_i x(k+i)$$

Bu işlemden sonra bir karar devresi yardımıyla $\tilde{a}(k)$ kestiriminin hangi simgeye karşı düştüğüne karar verilir ve bu karar $\hat{a}(k)$ olarak dengeleyici çıkışında görülür. Verilen kararın doğru olup olmadığını anlamak için dengeleme işlemi sırasında ve gerçek veri iletişimi başlamadan önce verici tarafından belli bir simge dizisi yollanır. Alıcıda bu simge dizisi aynen üretilerek simge kestirimine ilişkin yollanan simge ile karşılaştırılır.

$$e(k)=\tilde{a}(k)-a(k)$$

Yukarıdaki eşitlik ile hesaplanan fark, $e(k)=0$ ise kanal ve dengeleyici üzerinden sağlıklı bir veri iletişimi sağlanıyor demektir. Karşılaştırma sonunda $e(k)=0$ elde edilirse, bu hata değeri kullanılarak c_i, $i=0,+1,+2,.....,+m$ katsayıları ortaya çıkan hata azaltılacak şekilde ayarlanır. Kullanılan alıştırma dizisi yeterince uzun tutularak bu işlem $e(k)=0$ oluncaya kadar devam edilir. Neticede bu duruma erişildiğinde simgelerarası girişim olmayan sağlıklı veri iletişimi başlamış olur. Bu şekilde bilinen bir alıştırma dizisinin, veri iletiminden önce karşı tarafa gönderilmesiyle yapılan dengeleme işlemine otomatik dengeleme (automatic equalization) adı verilir. Bu şekildeki bir alıştırma süresi (training period) içerisinde, dengeleyici parametreleri iteratif olarak çıkıştaki hatayı minimize edecek şekilde ayarlanır. Kanal karakteristikleri zamanla değişmiyor ise otomatik dengeleme yeterlidir. Ancak kanal karakteristikleri zamanla değişiyor ise, dengeleyici sürekli olarak, bu değişiklikleri izleyebilecek şekilde kendini ayarlamalıdır. Bu durumda şekil 4.2.3 de görülen anahtar a(k) konumuna alınarak, dengeleyicinin kanal karakteristiklerindeki küçük değişimlere karşı ;

$$e(k)=\tilde{a}(k)-\hat{a}(k)$$

hatasını kullanarak durumunu koruması sağlanır. Böyle devam eden dengeleme işlemine ise uyarlamalı dengeleme (adaptive equalization) adı verilir. Eğer kanalın özellileri, yani kanal impuls yanıtının örnekleme anlarındaki değerleri biliniyor olsaydı, $2m+1$ sayıda doğrusal denklem her durum için çözülerek sabit dengeleyicinin $2m+1$ sayıdaki katsayıları bulunabilirdi. Ancak pratikte, kullanılacak kanalın özellikleri önceden bilinmediği için bu yol izlenemez. "Tapped-delay line" dengeleyiciler, katsayılarının belirlenmesi için çok değişik ayarlama algoritmaları kullanırlar ve yüksek hızlı ses bandı modemlerinde kullanılırlar. Bu dengeleme algoritmaları dengeleme işleminin işlevsel tanımındada değişiklik göstermektedir. Bununla beraber, iletim sisteminin ortalama karesel hatasının (mean-square error) minimize edilmesi en iyi dengeleme stratejisi olarak belirlenmiştir. Bu sınıftaki algoritmalar

kullanılan hesaplama metodundada değişiklik gösterirler. Değişik metodlar arasında "stochostic gradient" algoritması geniş ölçüde kullanılan bir tekniktir.

Bu algoritmanın oldukça popüler olmasına karşın, bazı durumlarda oldukça yavaş kararlı hale gelmesi gibi bir dezavantajı vardır. Bu algoritmanın yakınsama hızı Gittlin ve Ungerboeck tarafındanda gösterildiği gibi giriş ilişki matrisinin özdeğerlerine ve aynı zamanda dengeleyicinin ağırlık katsayılarının sayısına bağlıdır. Yüksek gürültü, giriş ilişki matrisinin değerlerinin birbirinden çok sapmasına yol açar ve bu durumda yakınsama hızını kontrol etmek çok zor bir hale gelir. Bu yüden, tüm iletim zamanının dikkate alınabilecek bir kısmını işgal eden bir alıştırma süresi zorunlu hale gelir. Yukarıda anlatılanlardan açıkça görülen şudur ki, giriş karakteristikleri süzgeç onları öğrenene ve optimum değerlere yakınsayana kadar değişmemelidir. Eğer yakınsamadan sonra giriş istatistikleri değişirse, süzgeç, ağırlıklarını yeni optimum değerlerine ayarlayacak şekilde değiştirerek cevap verecektir. Diğer bir deyişle, uyarlamalı süzgeç giriş istatistiklerinin kararlı olmayan değişikliklerini izleyecektir. Ancak bu işlem süzgeç için, değişiklikler arasında yakınsayabilecek ölçüde olmalıdır. Uyarlamalı gerçekleştirmedeki üç temel konu şunlardır :

1. Algoritmanın öğrenme ve yakınsama hızı,
2 Algoritmanın hesaplama karmaşıklığı,
3. Algoritmanın sayısal doğruluk ve kararlılığı

Yakınsama hızı önemli bir faktördür. Çünkü yakınsama hızı süzgeç tarafından kolayca izlenebilecek, girişteki maksimum değişme hızını belirler. İşlem karmaşıklığı bir zaman aralığından diğerine geçişte süzgecin yenilenmesi için gerekli olan işlem sayısını belirtir. Aşağıdaki tablo, değişik uyarlama algoritmaların bu ihtiyaçlar doğrultusundaki özelliklerini göstermektedir.

Algoritma	Hız	Karmaşıklık	Kararlılık
LMS	Yavaş	Basit	Kararlı
RLS	Hızlı	Karışık	Kararlı
FAST RLS	Hızlı	Basit	Kararsız
LATTİCE	Hızlı	Basit	Kararlı

Tablo 4.2.1

Şekil 4.2.2 de görülen dengeleyici katsayıları olan C_n, n=0,1,2,.....,N-1 , kanal ve dengeleyici impuls yanıtı birleştirilmiş örneklerinin sıfır olması sağlanacak şekilde seçilebilir. Bu şekildeki bir dengeleyici sıfıra zorlamalı (Zero-forcing--ZF) dengeleyici olarak adlandırılır. Eğer ZF dengeleyicisi sabitlerinin sayısının sınırlama olmadan artmasına olanak tanırsak, çıkışında sıfır ISI olan sonsuz uzunluklu bir dengeleyici elde ederiz.Bu şekilde bir dengeleyicinin frekans cevabı olan C(f), simge hızı olan T ile periyodiktir. Bunun nedeni T saniyelik vurum aralıklarıdır Örnekleme işleminden sonra alınan işaret üzerinde kanalın etkisi katlanmış frekans yanıtı olan H'(f) ile belirlenir. Kanal ve dengeleyicinin birleştirilmiş frekans cevabı aşağıdaki şartı sağlamalıdır.

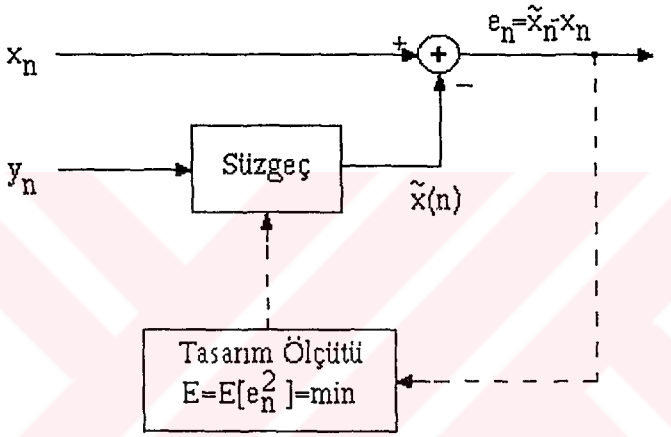
$C(f).H'(f)=1 \quad |f| < 1/2T$

Yukarıdaki ifadeden de görüldüğü gibi, sonsuz uzunluklu bir ZF dengeleyicisi basit olarak bir ters süzgeçtir. Sonlu uzunluklu ZF dengeleyicisi bu ters duruma yaklaşmaya çalışır.

LMS dengeleyicisi çok daha sağlam bir yapıya sahiptir. Burada dengeleyici sabitleri ortalama karesel hatayı minimize edecek şekilde seçilirler. Bu nedenle LMS dengeleyicisi çıkışındaki işaret gürültü oranını maksimum yapar.

4.3 En küçük Karesel Ortalama Hata (Least-Mean Square--LMS) Yöntemi

En küçük karesel ortalamalı hata yönteminde dengeleyici katsayıları, ISI ve gürültü nedeniyle dengeleyici çıkışında oluşan ortalama karesel hatayı en aza indirmeye çalışır. Bu yönteme ilişkin blok diyagram şekil 4.3.1 'de görülmektedir.



Şekil 4.3.1

Dengeleyici otomatik dengeleme modunda çalışıyorsa ortaya çıkan k. hata değeri ;

$$e(k)=\tilde{a}(k)-a(k) \tag{4.3.1}$$

olacaktır. Bu durumda ortaya çıkan hatanın karesel beklenen değeri olan $E[e^2(k)]$ büyüklüğünün sıfır olması istenir.

$$\begin{aligned} E[e^2(k)]&=0 \\ &=E[(\tilde{a}(k)-a(k))^2] \\ &=E[(\sum_{i=-m}^{m} c_i x(k+i)-a(k))^2] \end{aligned} \tag{4.3.2}$$

4.3.2 ifadesinin c_i katsayılarına göre kısmi türevleri alınır ve sıfıra eşitlenirse $2m+1$ bilinmeyenli bir denklem takımı elde edilecektir.

$$\frac{\partial E[e^2(k)]}{\partial c_{i=l}} = 2\,E[\{\sum_{i=-m}^{m} c_i x(k+i) - a(k)\}\, x(k+l)]$$
$$= 2E[e(k)\, x(k+l)]$$
$$= 0$$
$$l = 0, \pm 1, \pm 2, \ldots\ldots \pm m \qquad (4.3.3)$$

Bu denklem takımı çözülürse c_i katsayılarının optimal değerleri elde edilir. Fakat, uygulamada hattın karakteristikleri kesin olarak bilinemeyeceği $E[e^2(k)]$ nın analitik ifadesi elde edilemez. Bunun yerine c_i katsayısı ;

$$c_i(k+1) = c_i(k) - \alpha' \frac{\partial E[e^2(k)]}{\partial c_i(k)} \qquad (4.3.4)$$

şeklinde iteratif olarak kısmi türev ile orantılı ve kısmi türevin işeretine göre ters yönde değiştirilirse optimum değerine yakınsayacaktır.

$$c_i(k+1) = c_i(k) - \alpha' 2\, E[e(k)\, x(k+i)] \qquad (4.3.5)$$

4.3.5 ifadesinde görülen $E[e(k).x(k+i)]$ büyüklüğü yerine yaklaşıklık yapılarak büyüklüğün sadece o andaki değeri olan $e(k).x(k+i)$ kullanılırsa ve $\alpha = 2\alpha'$ olarak düşünülürse, pratik olarak hesaplanabilecek bir yöntem elde edilmiş olur. Bu işlemlerden sonra optimum katsayıların iteratif olarak hesaplanabilmesi için aşağıda belirtilen ifade kullanılabilir.

$$c_i(k+1) = c_i(k) - \alpha\, x(k+i).e(k) \qquad (4.3.6)$$

4.3.6 ifadesinde görülen α, uyarlama parametresi olarak adlandırılır ve 1' den küçük pozitif bir sayıdır. Bu parametrenin değeri yakınsama hızını ve varılabilecek en küçük hata sınırını belirler.

Eğer dengeleyici girişine gelen işaretin güç spektrumu düzgün bir dağılım gösteriyorsa, uyarlama parametresinin değeri, alınan ortalama işaret gücü ile dengeleyici vurum sayısının çarpımının tersine eşit alınarak hızlı bir yakınsama sağlanabilir. Yani;

$$0 < \alpha < 1 / N.P_{av}$$

olarak seçilmelidir. Girişteki işaretin güç spektrumu büyük değişiklikler gösteriyorsa, α dahada küçük tutulmalıdır. Ancak bu durumda yakınsama hızı düşecektir. Varılabilecek minimum hata sınırını belirleyen diğer etkenler, dengeleyici vurum sayısı ve toplamsal Gauss gürültüsünün ortalama gücü olarak belirtilebilir. İşte bütün bu özellikler göz önüne alınarak yakınsama hızı ve hata sınırı arasında optimum bir nokta seçilmeli ve α bu nokta için belirlenmelidir.

Otomatik çalışma modunda, alıştırma dizisi kullanılarak dengeleme sağlandıktan sonra ortaya çıkan hata;

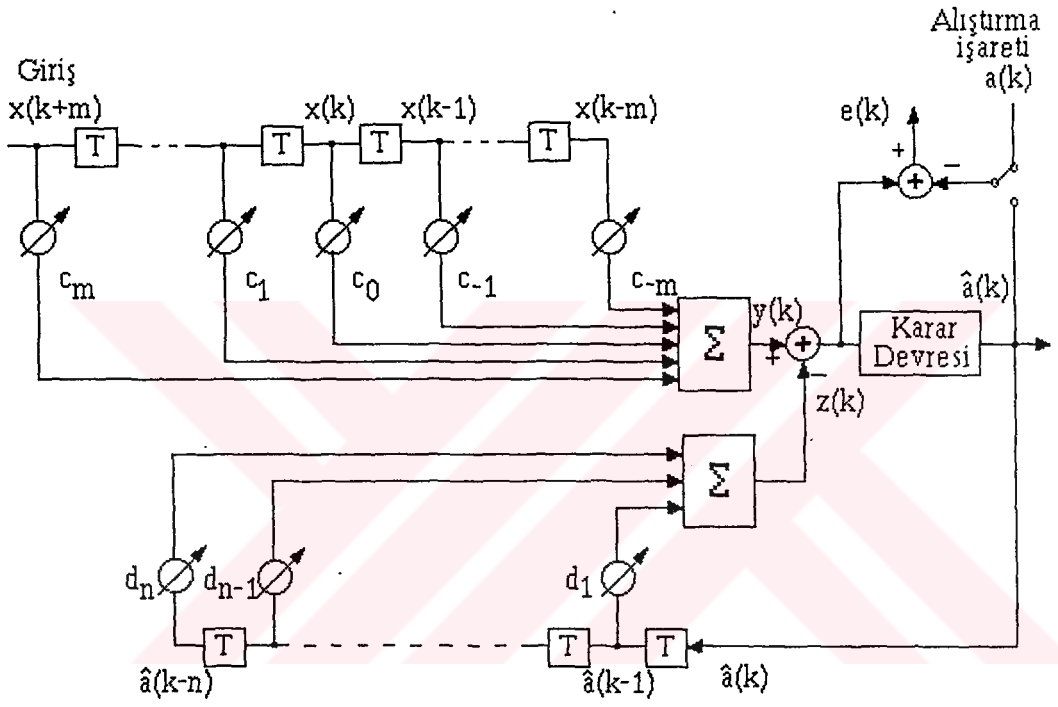
$$e(k) = \tilde{a}(k) - \hat{a}(k) \qquad (4.3.7)$$

olarak hesaplanabilir. Burada $\hat{a}(k)$ nın mutlaka doğru olması şart değildir. Bununla birlikte otomatik çalışma esnasında dengeleme büyük ölçüde sağlandığı için, $\hat{a}(k)$ yüksek olasılıkla doğrudur. Böylece

dengeleyici uyarlamalı çalışma esnasında kanal karakteristiklerindeki küçük değişimleri ve örnekleyicinin fazındaki titreşimleri izleyebilir.

4.4 Karar Geri Beslemeli Dengeleyiciler (Decision Feedback Equalizers)

Temel dengeleyici yapısı doğrusal ve rekursif olmayan bir süzgeç olarak ortaya çıkmıştır. Basit bir doğrusal olmayan dengeleyici, kötü genlik bozulmasına sahip kanallar için özellikle yararlıdır ve tespit edilmiş olan simgelerden dolayı oluşan girişim için karar geri beslemesi kullanılır. Şekil 4.4.1 bu şekildeki bir karar geri beslemeli dengeleyiciyi göstermektedir.



Şekil 4.4.1
Karar geri beslemeli dengeleyici

Dengelenen işaret, dengeleyicinin ileri ve geri besleme kısımlarının çıkışlarının toplamıdır. İleri kısım bir doğrusal transversal süzgeç gibidir. Dengelenen işaret üzerinde verilen kararlar, diğer bir ikinci transversal süzgeç yardımıyla geriye beslenir. Şekil 4.4.1 de görülen karar geri beslemeli dengeleyicide ileri kısım 2m+1 kattan oluşmaktadır. Geri besleme yolu üzerinde bulunan doğrusal transversal süzgeç ise n kattan oluşmaktadır ve d_j, j=0,1,2,.....,n katsayılarına sahiptir.

Karar verilen işaretlerin geri besleme yolu üzerindeki doğrusal süzgeç yardımıyla tekrar geriye beslenmesindeki temel fikir şudur :

Eğer halihazırda tespit edilmiş olan simgelerin değerleri biliniyor ise (geçmiş kararlar doğru olarak kabul edilmektedir), bu simgelerin neden olduğu ISI da, geçmiş simge değerlerinin uygun bir ağırlaştırma yapılarak dengeleyici çıkışından çıkarılması ile tam olarak giderilebilir. Ağırlıklar, kanal ve

dengeleyicinin ileri kısmını kapsayan sistemin impuls yanıtının uç kısımlarının örnekleridir. İleri ve geri besleme katsayıları eşzamanlı olarak karesel ortalama hatayı minimum yapacak şekilde ayarlanabilir. Bu durumda dengeleyici kestirimi olan a(k) ;

$$\tilde{a}(k)=\sum_{i=-m}^{m} c_i x(k+i)-\sum_{j=1}^{n} d_j \hat{a}(k-j) \qquad (4.4.1)$$

olacaktır. İleri yol ve geri besleme katsayıları daha önce açıklanan en küçük karesel ortalama hata yönteminde olduğu gibi ayarlanmaktadır.

$$E[e^2(k)=0$$

$$=E[\{\tilde{a}(k)-a(k)\}^2]$$

$$=E[\{\sum_{i=-m}^{m} c_i x(k+i)-\sum_{j=1}^{n} d_j \hat{a}(k-j)-a(k)\}^2]$$

$$\frac{\partial E[e^2(k)]}{\partial c_{i=l}}=2\,E[\{\sum_{i=-m}^{m} c_i x(k+i)-\sum_{j=1}^{n} d_j \hat{a}(k-j)-a(k)\}\,x(k+l)]$$

$$=2E[e(k)\,x(k+l)]$$

$$=0$$

$$l=0,\pm1,\pm2,.....\pm m \qquad (4.4.2)$$

4.4.1 ve 4.4.2 ile bulunan $2m+n+1$ bilinmeyenli denklem seti c_i ve d_j katsayılarının optimum değerlerini verecektir. Uygulamada yapılan iteratif metodda ise c_i katsayıları 4.3.5 ifadesinde verildiği gibi, d_j katsayılarıda buna benzer şekilde hesaplanabilir.

$$c_i(k+1)=c_i(k)-\alpha x(k+i)\,e(k) \qquad (4.4.3)$$

$$d_j(k+1)=d_j(k)+\beta \hat{a}(k-j)\,e(k) \qquad (4.4.4)$$

Yukarıdaki ifadelerde -β, α 'ya karşılık gelmek üzere geri besleme için kullanılan bir katsayıdır. Bu sistemde ileri yol süzgeci daha önce anlatılan doğrusal süzgecin görevini aynen görür. Geri besleme süzgeci ise, kanal ve ileri yol süzgecinin toplam impuls yanıtında sıfır olması gereken fakat sıfırdan farklı kalmış olan örnekleri sıfırlamaya çalışır. d_j katsayılarının optimum LMS değerleri, ISI yı sıfıra indiren değerlerdir. Bu ZF süzgecindekine benzer bir şekilde yapılmaktadır. Dikkat edilirse, DFE'nin geri besleme kısmının çıkışı, gürültüsüz geçmiş kararların ağırlaştırılmış bir toplamı olduğu için, geri besleme katsayıları dengeleyici çıkışındaki gürültü gücünü belirlemede hiç bir rol oynamamaktadır.

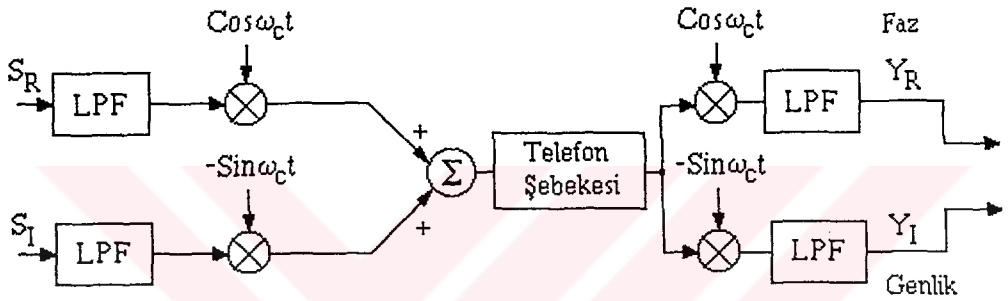
Aynı sayıda katsayı verildiğinde, bir geri beslemeli dengeleyici, bir doğrusal dengeleyiciden daha küçük karesel ortalamayı başarabilirmi ?

Bu soruya ilişkin kesin bir cevap yoktur. Her iki tipteki dengeleyicinin başarımı, belirli kanal karakteristikleri ve örnekleyici fazı ile belirlenmiştir. Katsayıların gerçek sayısı ve esas vurumun yeride başarımı belirleyen etkenler arasındadır. Bununla birlikte DFE başarımı örnekleyici fazına daha az bağlıdır. Doğrusal transversal dengeleyicinin katsayıları, birleştirilmiş kanal ve dengeleyici impuls yanıtı birim impulsa yaklaşacak şekilde seçilir. DFE 'de ise belirli sayıda geçmiş simgelerin kullanılmasıyla,

geri besleme kısmının ISI yı yok etme yeteneği, ileri kısım için kullanılacak katsayıların seçimine belirli bir serbestlik sağlar. Kanal ve ileri kısmın birleştirilmiş impuls yanıtı, ana impuls dışında sıfır olmayan örneklere sahip olabilir. Buda, DFE nin ileri kısmının ters kanal karakteristiklerine çok fazla yaklaşma zorunluluğunu, aşırı gürültü iyileştirme ve örnekleyici fazına duyarlılığını ortadan kaldırmaktadır.

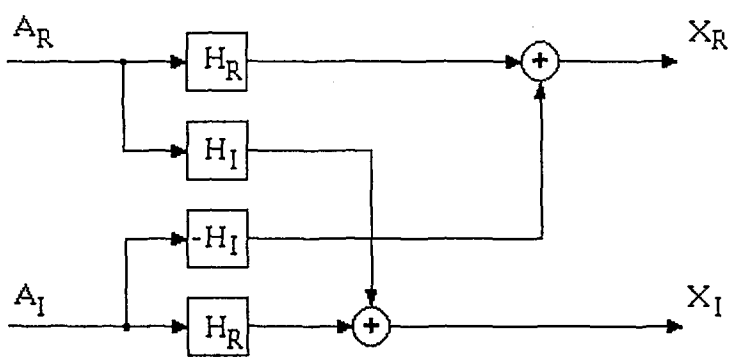
4.5 QAM Sistemleri İçin Dengeleyiciler

Modern yüksek hızlı ses bandı modemleri, yüksek hızlar için QAM modülasyonunu kullanırlar. Gürültü ve diğer kanal bozukluklarının belirgin olduğu yüksek hızlarda, QAM modülasyonunun kodlanmış türünü kullanan modemler yüksek başarım sağlamak için kullanılırlar. Şekil 4.5.1 de klasik bir QAM sistemi görülmektedir.



Şekil 4.5.1
QAM Sistemi

Alıcıdaki alçak geçiren süzgeç çıkış işaretleri $y_r(t)$ ve $y_i(t)$, y(t) kompleks işaretinin gerçel ve sanal bileşenleri olarak gösterilebilir. Bu modülasyon yönteminde, modülasyonlu işaret birbirinden 90 derece farklı iki taşıyıcı içermektedir. İletilen veriye ilişkin bilgi bu iki taşıyıcı üzerinde bölüştürülür. Bu iki bileşen gerçek olmakla beraber kompleks bir büyüklüğün gerçel ve sanal bileşenleri olarak modellenebilir. Kompleks olarak modellenmiş kanal şekil 4.5.2 de görülmektedir.

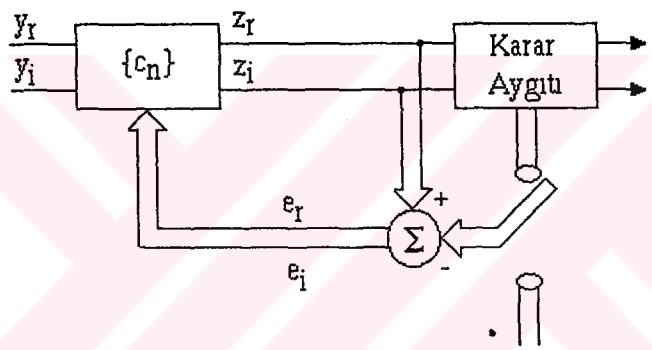


Şekil 4.5.2

Burada A_R ve A_I modülasyonlu işaretin farklı taşıyıcılara sahip iki bileşeni olup, kompleks uzayda $A=A_R+jA_I$ şeklinde gösterilebilir. $H=H_R+jH_I$ kompleks kanalın transfer fonksiyonu ve $X=X_R+jX_I$ kompleks kanal çıkışıdır. Burada belirtilen H ve X büyüklükleri frekans domeninde gösterilmektedir. Dolayısıyla kompleks kanal çıkışındaki X büyüklüğünün ifadesi, zaman domeninde, $a(t)=a_r(t)+ja_i(t)$ kompleks kanal giriş işareti ile $h(t)=h_r(t)+jh_i(t)$ kompleks kanal impuls yanıtının katlanmasına eşit olacaktır. Frekans domeninde ise katlama işlemi çarpma işlemine denk düştüğü için kanal çıkışındaki kompleks büyüklük şu şekilde yazılabilir ;

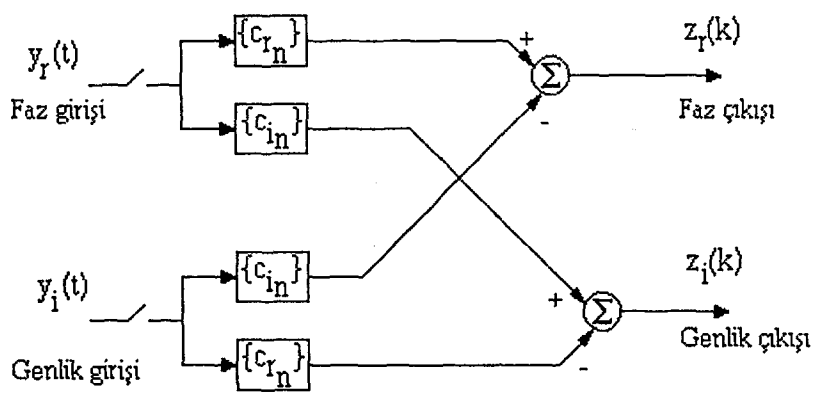
$$X=A.H=(A_RH_R-A_IH_I)+j(A_RH_I+A_IH_R) \tag{4.5.1}$$

Bu şekilde iki boyutlu bir işareti dengelemek için kullanılacak olan dengeleyici sistemininde iki boyutlu olması kaçınılmazdır. Kompleks c_n katsayılarına sahip olan temel band dengeleyicisi, y(t) kompleks işaretinin örnekleri üzerinde işlem yapar ve dengelenmiş kompleks örnekler olan $z(k)=z_r(k)+jz_i(k)$ üretir. Bu durum şekil 4.5.3 'de görülmektedir.



Şekil 4.5.3

Kompleks dengeleyiciyi, dört gerçek transversal süzgeçten oluşmuş bir yapı olarak gösterilebilir. Bu durum şekil 4.5.4 de görülmektedir.



Şekil 4.5.4

QAM modemleri için kompleks transversal dengeleyici

Gerçel katsayılar olan c_{r_n} n=0,1,.....,N-1 katsayıları, faz ve genlik kanallarındaki simgelerarası girişimi yenmeye çalışırken, sanal katsayılar olan c_{i_n} katsayıları iki kanal arasındaki çapraz girişimi etkisiz hale getirmektedir. Katsayılar, kompleks hata işareti olan $e(k)=e_r(k)+je_i(k)$ 'nın karesel genliğinin ortalamasını minimize edecek şekilde ayarlanır. Burada e_r ve e_i, z_r ve z_i ile istenen değerler arasındaki farklardır. Katsayıların güncelleştirilmesi için kullanılan metod tek boyutlu dengeleyici ile aynıdır. Tek fark bütün eğişkenleri kompleks değerlikli olmasıdır. Şekil 4.5.4 'de görülen kompleks transversal dengeleyici çıkışı $z(k)=y(k)*c(k)$ olarak belirlenecektir. Frekans domeninde dengeleyici çıkışı;

$$Z=Y.C=(Y_RC_R-Y_IC_I)+j(Y_RC_I+Y_IC_R) \tag{4.5.2}$$

olacaktır. Aşağıda, şekil 4.4.1 'de verilen karar geri beslemeli dengeleyici sisteminin iki boyutlu hali için değişkenler ve sistem denklem takımı verilmiştir.

Değişkenler :

$X(k)=X_R(k)+jX_I(k)$	; k. dengeleyici giriş işareti.
$C_i(k)=C_{iR}(k)+jC_{iI}(k)$	; ileri yol süzgeci i. katsayısının k. değeri.
$Y(k)=Y_R(k)+jY_I(k)$	; ileri yol süzgeci k. çıkış işareti.
$A(k)=A_R(k)+jA_I(k)$	; k. dengeleyici kestirimi.
$A(k)=A_R(k)+jA_I(k)$	; k. dengeleyici kararı.
$D_j(k)=D_{jR}(k)+D_{jI}(k)$	; geri besleme süzgeci j. katsayısının k. değeri.
$Z(k)=Z_R(k)+jZ_I(k)$	; geri besleme süzgecinin k. çıkış işareti.
$E(k)=E_R(k)+jE_I(k)$	; k. hata işareti.

Denklem Takımı :

$$Y_R(k)=\sum_{i=-m}^{m}[C_{iR}(k)X_R(k)-C_{iI}(k)X_I(k)] \tag{4.5.3}$$

$$Y_I(k)=\sum_{i=-m}^{m}[C_{iR}(k)X_I(k)+C_{iI}(k)X_R(k)] \tag{4.5.4}$$

$$Z_R(k)=\sum_{j=1}^{n}[D_{jR}(k)\hat{A}_R(k)-D_{jI}(k)\hat{A}_I(k)] \tag{4.5.5}$$

$$Z_I(k)=\sum_{j=1}^{n}[D_{jR}(k)\hat{A}_I(k)+D_{jI}(k)\hat{A}_R(k)] \tag{4.5.6}$$

$$C_{iR}(k)=C_{iR}(k-1)-\alpha[X_R(k)E_R(k-1)+X_I(k)E_I(k-1)] \tag{4.5.7}$$

$$C_{iI}(k)=C_{iI}(k-1)-\alpha[X_R(k)E_I(k-1)-X_I(k)E_R(k-1)] \tag{4.5.8}$$

$$D_{jR}(k)=D_{jR}(k-1)+\beta\,[\hat{A}_R(k)\,E_R(k-1)+\hat{A}_I(k)\,E_I(k-1)] \quad (4.5.9)$$

$$D_{jI}(k)=D_{jI}(k-1)+\beta\,[\hat{A}_R(k)\,E_I(k-1)-\hat{A}_I(k)\,E_R(k-1)] \quad (4.5.10)$$

$$\tilde{A}_R(k)=Y_R(k)-Z_R(k) \quad (4.5.11)$$

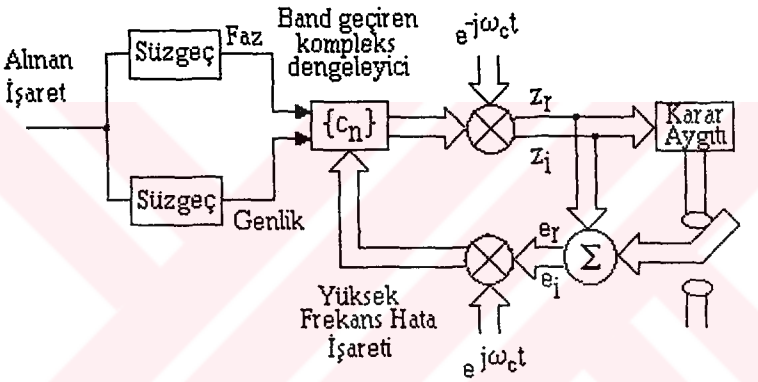
$$\tilde{A}_I(k)=Y_I(k)-Z_I(k) \quad (4.5.12)$$

$$E_R(k)=\tilde{A}_R(k)-\hat{A}_R(k) \quad (4.5.13)$$

$$E_I(k)=\tilde{A}_I(k)-\hat{A}_I(k) \quad (4.5.14)$$

4.6 Yüksek Frekans Bandı Dengeleyicisi

Kompleks dengeleyiciler geçirme bandında da kullanılabilir. Bu dengeleyici demodülasyondan önce alınan işaretin dengelenmesi için kullanılır. Bu durum şekil 4.6.1 'de görülmektedir.



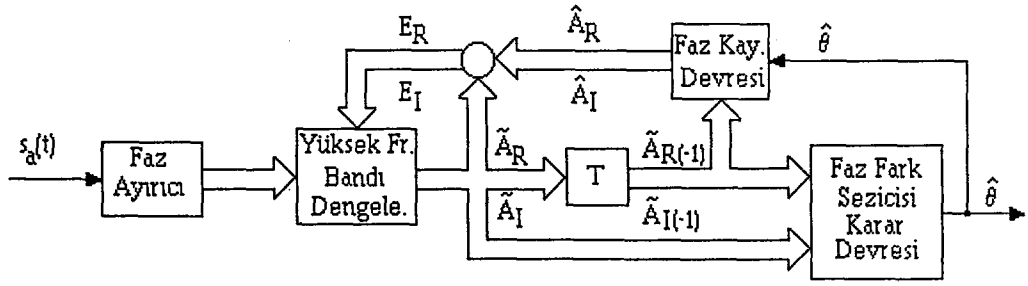
Şekil 4.6.1

Yüksek Frekans Bandı Dengeleyicisi

Burada alınan işaret faz ayırma süzgeç çifti yardımıyla faz ve genlik bileşenlerine ayrılır. Bu süzgecin çıkışındaki kompleks, geçirme bandlı işaret simge hızında örneklenir ve temel bandda olduğu gibi dengeleyicinin gecikme hattına uygulanır. Dengeleyicinin kompleks çıkışı demodülasyon işleminden geçirilir (kompleks bir üstel işaret ile çarpılarak) . Demodülasyon olayı kararlar verilmeden önce yapılır ve kompleks hata temel band içerisinde hesaplanır. Bu dengeleyicinin yakınsama hızı açısından gösterdiği başarım temel band dengeleyicisine denktir. Buna karşılık demodülasyonun dengeleyici girişinde olmaması nedeniyle, bu işlemin getirdiği bozucu etkilerden sakınılmış olur. Yöntemin tek sakıncası, kompleks frekans geçişlerinin gerçekleştirilmesidir. Bunun sağlanması oldukça zordur. Faz kilitlenmesi gerekliliği olduğundan dolayı bu yöntem "Bağdaşık (coherent)" olarak adlandırılır.

Bir problem olarak ortaya çıkan frekans geçişlerini ortadan kaldırmak için yeni bir yöntem geliştirilmiştir. Bu yöntemde faz geçişleri bulunmadığı dolayısıyla faz kilitlenmesinede gerek olmadığı

için "Bağdaşık olmayan (incoherent) yöntem olarak adlandırılır. Bu türden bir yüksek frekans dengeleyicisinin şeması şekil 4.6.2 'de görülmektedir.



Şekil 4.6.2

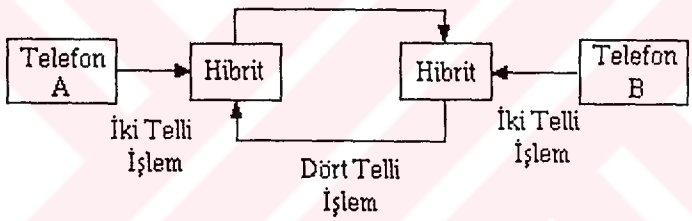
BÖLÜM 5 YANKI GİDERİCİLER

Telefon şebekesindeki yankılar gerek sesin nitelik ve anlaşılabilirliğini olumsuz yönde etkilediği gerekse tam çift yönlü (full dublex) veri iletişimine engel olduğu için önemli bir sorundur. Yankı sorununun kökeninde, iletim ortamındaki empedans süreksizlikleri ve bu süreksizliklerdeki yansımalar yatmaktadır.

Şebekede çeşitli konumlarda süreksizlikler olabilir. Ancak en etkili süreksizlikler iki /dört telli değiştiricilerdir. 2/4 telli değiştirici, çift yönlü ve iki telli olan abone döngüsü ile dört telli ve teker yönlü olan uzun erişimli şebeke arasında bağlantıyı sağlar.

5.1 Ses Ve Veri İletiminde Yankı Problemi

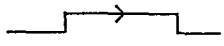
Bir telefon şebekesi, telefon bağlantısının yakınında ve sonunda yankılar üretir. Yankı giderme, ses ve veri iletiminde bu yankıyı gidermek için kullanılan bir terimdir. Ses ve veri için bunun sağlanabilmesi için duyulan ihtiyaçlar değişiktir. Yankı teknolojisine şekil 5.1.1 ile başlamak yerinde olacaktır.



Şekil 5.1.1

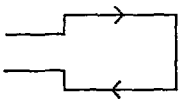
Şekil 5.1.1 'de basitleştirilmiş bir telefon bağlantısı görülmektedir. Bu bağlantı şekli, uç noktalarda iki telli parçaları (abone çevrimleri ve lokal şebekenin bir kısmı) kapsaması açısından tipiktir. Bu iki telli parçalarda her yöneki iletim tek bir kablo çifti üzerinden yapılmaktadır. Bağlantının orta noktası dört tellidir. Bu kısımda iletimin iki doğrultusu, değişik fiziksel taşıyıcılar üzerinden ayrılmıştır. Bağlantının dört telli kısmı etrafında potansiyel bir geri besleme çevrimi vardır ve bu yol üzerinde yeterli bir kayıp olmaksızın, iletim bozuklukları ve bazı durumlarda osilasyonlar (singing) ortaya çıkacaktır.

Hibrit bu çevrim etrafında, iki konuşmacının konuşma yolu üzerindeki kaybını etkilemeksizin, büyük bir kayıp sağlayan bir aygıttır. Şekil 5.1.2, 5.1.3 ve 5.1.4 hibrit devresinin fonksiyonunu çok daha etkin bir şekilde göstermektedir. Şekil 5.1.2 'de iki konuşmacının konuşma yollarından biri görülmektedir.

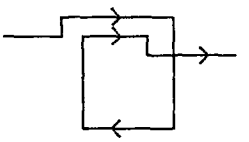


Şekil 5.1.2

Bu yolun büyük bir kayba uğramaması için, hibrit devresinin iki telli portu ile diğer dört telli portları arasında kayda değer bir zayıflamanın olmaması gereklidir. Şekil 5.1.3 ve 5.1.4 'de görüldüğü gibi bu yapı üzerinde iki belirgin yankı mekanizması vardır



Şekil 5.1.3
Konuşmacı Yankısı



Şekil 5.1.4
Dinleyici Yankısı

Konuşmacı yankısında, konuşmacı kendi konuşmasının gecikmiş bir halini işitecektir. Dinleyici yankısında ise, dinleyici konuşmacının anlatımının gecikmiş bir halini işitir. Konuşanın kulağına birkaç milisaniye içerisinde ulaşan yankı, yerel yankı olarak alandırılır. Şebekenin öteki ucunda alıcı abonenin 2/4 telli değiştiricisinden sızan ikinci yankı birinci uzak yankı (fist remote echo) diye nitelendirilir. İki abone arasında birkezden çok dolaşıp, konuşan aboneye ulaşan yankı ikinci uzak yankı olarak adlandırılır.

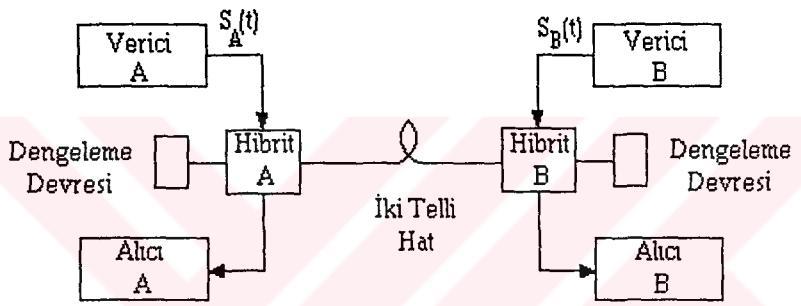
Küçük yayılma gecikmesine sahip bağlantılarda, bu yankı olayından doğan rahatsızlık bağlantıya bir kayıp sokulması ile kontrol edilebilir. Çünkü yankı işareti bu kayba iki veye üç kez maruz kaldığı halde konuşmacı yolu sadece bir kez maruz kalacaktır. Ancak yüksek yayılma gecikmesine sahip hatlarda yankı gidermeye yönelik zayıflatma, algılanan seside çok zayıflatacağı için bu yöntem kullanılmaz. Bunun yerine, yankı problemleri geleneksel olarak yankı bastırıcılar (echo suppressor) ile giderilmeye çalışılır.

Tekrar etmek gerekirse, yankının bastırılması yada giderilmesinin amacı tam çift yönlü (FDX) iletişim yapabilmektir. Sözlü iletişimde amaç konuşmanın nitelik ve anlaşılabilirliğini arttırmaktır. Veri iletişiminde ise, bir yandan çift yönlülüğün verimliliği, ucuzluğu ve güvenilirliği söz konusu olurken, öte yandan "videotext", "teletex", "fax" gibi hizmetlerde tam çift yönlü iletim zorunlu olmaktadır.

5.2 İki Telli Devreler Üzerinde Eşzamanlı Çift Yönlü Veri İletimi Ve Yankı Giderme

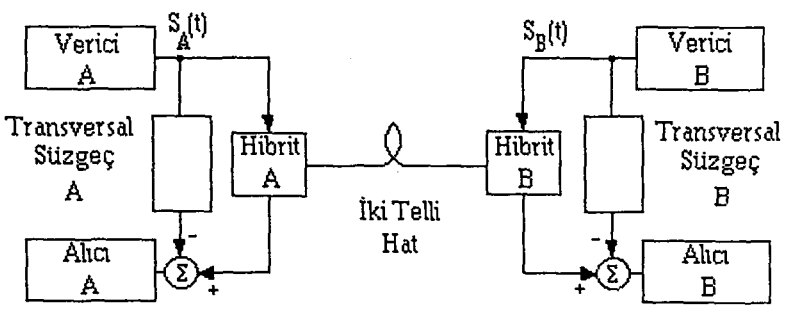
Dört telli çalışmanın sağlanmasının pratik olmadığı durumlarda, iki telli devreler üzerinde eşzamanlı, iki yönlü iletime ihtiyaç duyulur. Aynı zamanda, tam çift yönlü olarak çalışabilen bir modemi

iki telli devreye birleştirecek olan aygıtında modülasyon türünden bağımsız olarak kendi kendini ayarlayabilir olması ve çok pahalı olmaması gerekmektedir. İlk başta, iki telli hatlar üzerinde iki doğrultuda hareket eden işaretleri ayırmak için hibrit aktarıcıları (hybrid couplers) kullanılmaktaydı. Bununla birlikte, hibrit aktarıcının iletim terminalinden çıkış terminaline olan sızıntının bastırılma derecesi, aktarıcının dengeleme devresinin hat empedansına nekadar iyi uyduğuna bağlıdır. Hat empedansı ise genellikle kesin olarak bilinemez. Ses iletiminde, dengeleyici devresi sızıntıyı kabul edilebilir bir seviyede tutmak için yeterlidir. Ancak bu seviye, yüksek kaliteli veri iletimi için yeterince düşük olmayabilir. İlave olarak, iletim devresindeki empedans süreksizlikleri iletilen işaretin yankısını geriye yansıtır ve bu yankı işareti alınan işarete karışacaktır. Bu olay hibrit devresinin başarımından bağımsız olarak gerçekleşir. Lokal çevrimler dışında, uzun mesafe telefon devrelerinin dört telli olması nedeniyle yankılar genelde yakın ve uzak gruplar olarak oluşacaktır. Şekil 5.2.1 'de sadece sadece hibrit devreleri kullanılarak yapılan tam çift yönlü veri iletimi görülmektedir.



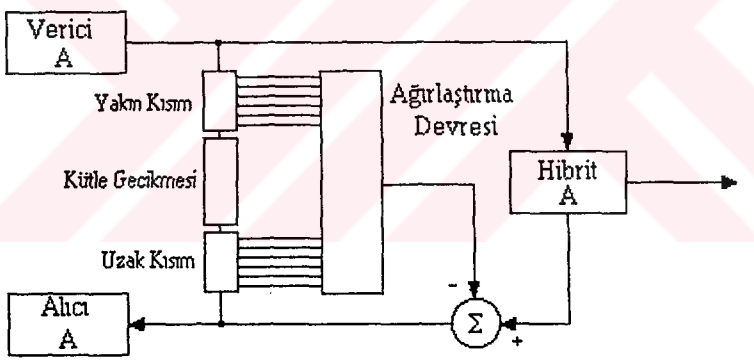
Şekil 5.2.1

Yankı bozukluklarının ve aktarıcının uçnoktaları arasındaki sızıntının şekil 5.2.2 'de görüldüğü gibi alıcı devrelerine ulaşmadan yok edilmesi mantıklı bir düşüncedir. Bu şekilde hibrit devreleri ve transversal süzgeç yankı gidericilerden oluşan bir tam çift yönlü iletim çalışma şekli görülmektedir. Bu çalışma şekli, yankı gidericilerin hasasiyet ve uzunluk ihtiyaçlarını azaltmak için hibrit aktarıcılarını ve dengeleme devrelerini kapsamaktadır.



Şekil 5.2.2

A hibrit devresinin alış uç noktası üzerindeki işaretin, gönderilen $S_A(t)$ ve $S_B(t)$ işaretlerinin zamanla değişmeyen doğrusal değişimlerinin toplamı olarak düşünülebilir. $S_A(t)$ lokal işareti üzerindeki değişim, doğrudan hibrit sızıntısını ve iletim devresinden gelen yankıları kapsamaktadır. Eğer A transversal süzgeci aynı değişikliği üretecek şekilde ayarlanırsa, toplayıcının çıkışı sadece $S_B(t)$ işaretinin fonksiyonu olacaktır. Buda gerçekte alınması arzu edilen işarettir. $S_A(t)$ işaretinin dönüşümünün zamanla değişmez olduğu kabul edilirse, transversal süzgeç bir başlangıç ayarlamasına sokulabilir. B vericisi etkisiz hale getirilir ve A süzgecinin katsayıları, toplayıcı çıkışındaki fark işaretinin minimum karesel ortalamasını elde etmeye çalışan bir algoritma yardımıyla ayarlanır. Bu andan sonra, iki yönlü haberleşme başlayabilir. Transversal süzgecin uyarlamalı çalışma esnasında süzgeç katsayılarının ayarlanma hızı, simge hızına oranla çok düşük olacaktır. Bunun nedeni alınan büyük genlikli işaretten karışıklığa neden olan işaretin çıkarılması için gerekli olan uzun hesaplama süresidir. Uzak yankı üreten işlev, zamanla değişmez olsa bile, yankı işareti iletim hattı üzerindeki devreler nedeniyle 60 ms kadar uzunca bir süre geciktirilebilir. Bu ise pratik uzunluktaki bir transversal süzgecin toplam gecikmesinden çok daha fazladır. Bu nedenle uzak yankının gecikme süresini ölçebilecek bir metod oluşturup buna göre transversal süzgecin uzak ve yakın kısmının katsayıları arasına yeterli miktarda bir kütlesel gecikme konmalıdır. Bu durum şekil 5.2.3 'de görülmektedir.



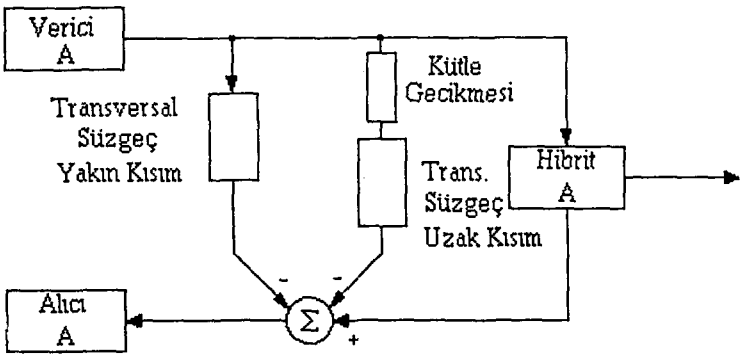
Şekil 5.2.3

Yakın Ve Uzak Yankı İşaretleinin Giderilmesi İçin Seri Gerçekleştirme

Diğer bir gerçekleme metoduda, iki paralel ve bağımsız olarak ayarlanabilen transversal süzgeç yapısıdır. Bu gerçekleştirme şekil 5.2.4 de görülmektedir.

a. Ses iletişimindeki birkaç milisaniye sonra gelen ve yerel santralden yansıyan yankı zarar vermez. Oysa veri iletişiminde bu yerel yankı uzak yankılar kadar zararlıdır.Bu bakımdan yankı giderici şekil 5.2.2 'de görüldüğü gibi veri aygıtının içine yerleştirilmelidir. Böylece şebeke yankısız olsabile, yerel yankıyı gidermek üzere söndürücü yine gerekli olacaktır.

b Hem yerel hemde uzak yankılar ayrı ayrı giderileceği için, transversal süzgeç şekil 5.2.3 veya 5.2.4 'deki gibi ikiye ayrılmalıdır ve araya bir kütlesel gecikme konmalıdır.



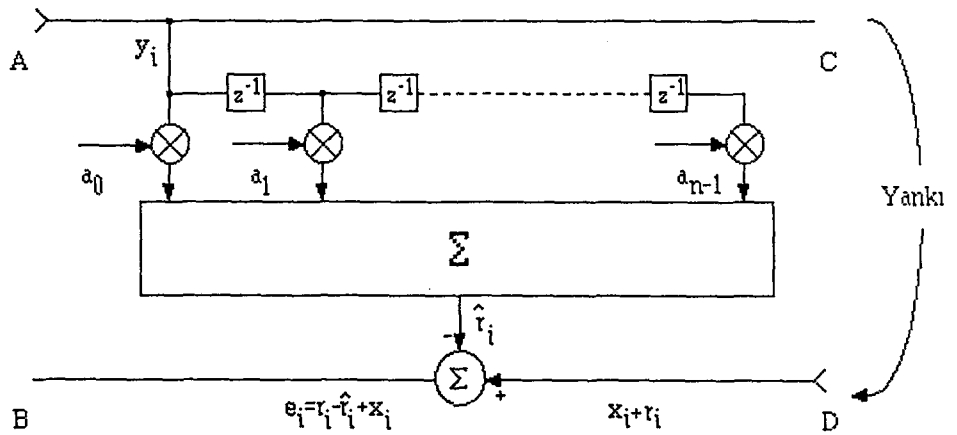
Şekil 5.2.4

Yakın Ve Uzak Yankı İşaretlerinin Giderilmesi İçin Paralel Gerçekleştirme

Veri iletimindeki yankı giderme problemi, ses iletişimindeki yankı gidermeye göre farklılıklar gösterir.

c Sesin istatistiksel özellikleri karmaşıktır. Oysa, veri dizilerinin istatistiksel özellikleri çok daha sadedir. Temel banddaki işaretin, kodlanmış band geçiren işaretin ve yankının iletilen bilgi simgelerinden kaynaklandığını algıladığımızda, yankının sentezlenmeside bu bilgi simgelerinden hareketle gerçekleştirilir.

İletimin bir yönü için yankı gidericinin prensibi şekil 5.2.5 'de görülmektedir. Yankı giderici, dört telli yol üzerinde yankının kaynağına yakın olarak yerleştirilmiştir. İletimin bir yönü A uçnoktasından C uçnoktasına ve diğer yönü ise D uçnoktasından B uçnoktasına doğrudur. Yankı kanalından geçen y_i referans işareti, r_i yankı işaretini üretir. Bu yankı işareti, uzak uç konuşmacı işareti x_i ile birlikte D uçnoktasında belirir.x_i işaretinin içerisinde ısıl gürültülerin etkiside kapsanmaktadır. Referans işaretinin transversal süzgece uygulanmasıyla r_i yankı işaretinin bir tekrarı üretilir ve bu işaret D uçnoktasındaki işaretten çıkarılır.Böylece yankı giderme hata işareti olan e_i oluşturulmuş olur.



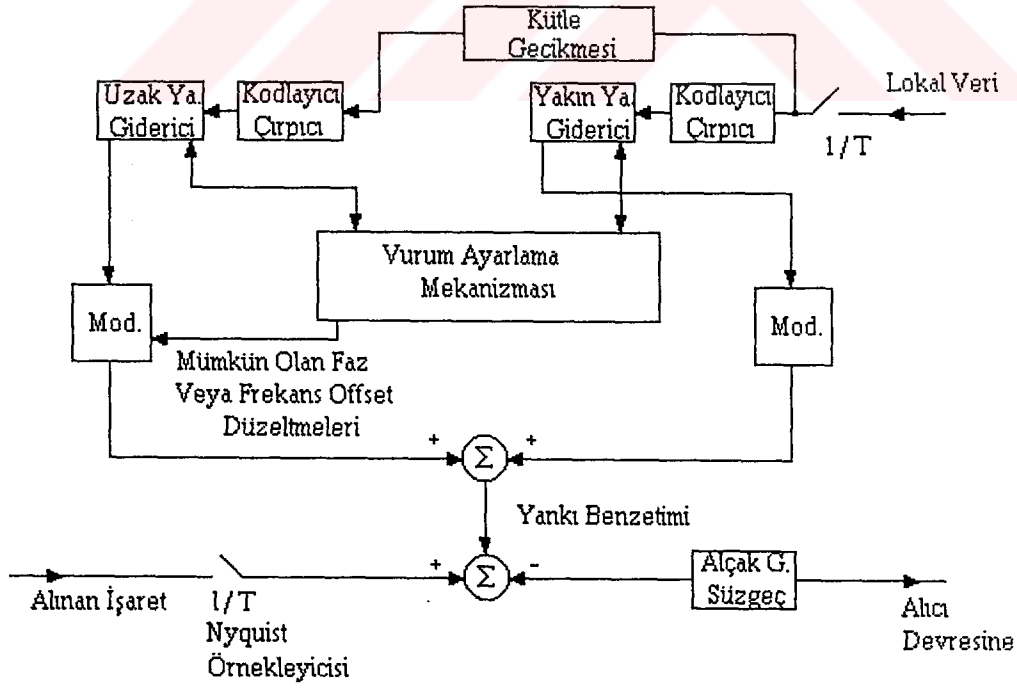
Şekil 5.2.5

n-1 sayıdaki $a_0,.....,a_{n-1}$ süzgeç katsayıları yankı transfer fonksiyonuna uyarlanmaya zorlanır.

Çok gelişmiş uyarlama algoritmaları olduğu halde bu basit algoritma bir çok yankı giderme uygulamaları için yeterlidir. Giderme hatası olan e_k, bu hatadan yararlanarak transversal süzgeç katsayılarında gerekli düzeltmelerin yapılması için kullanılır. Katsayılardaki düzeltme işlemi e_k hatası azaltılacak ve yankı gidericinin uyarlanması sağlanacak şekilde yapılır. Buradaki toplama noktası, çıkışı katsayılardan biri olan bir akümülatördür. Burada toplanan değer β adım büyüklüğü ile çarpılarak uyarlamanın hızı kontrol edilmiş olur. Hata işareti e_k, yankı giderme hatası ile birlikte uzak uç konuşmacı işareti x_k işaretinide kapsamaktadır. Eğer y_k, x_k ile ilişkisiz ise, uzak uç konuşma işareti süzgeç katsayılarının ortalama değerini etkilemeyecektir. Fakat süzgeç katsayılarının bu ortalama değer etrafındaki değişimleri, uzak uç konuşma işaretinin varlığı altında dahada artacaktır. Bu etki çok küçük bir uyarlama hızının seçilmesi ile (küçük β) karşılanabilir. Fakat genelde bir uzak uç konuşma sezicisi kullanmak çok daha uygundur. Uyarlama işlemi, hissedilebilecek derecede bir uzak uç konuşma işareti yok iken ilerler.

5.3 İki Telli Devreler Üzerinde Tam Çift Yönlü İletim İçin Geçirme Bandlı, Veri Güdümlü Yankı Giderici

Dikkat edilmesi gerekir ki, iletim hattının diğer ucundan dönen yankı (bu bir uydu hattıda olabilir) 500 ms ye varan gecikmeye uğratılmış olabilir. Bu nedenle şekil 5.3.1 'de görüldüğü gibi, yankı giderme süzgecini yakın ve uzak yankı giderici olarak iki kısıma ayırmak son derece doğaldır.



Şekil 5.3.1

Şekil 5.2.1 'deki her istasyon tarafından gerçekleştirilen başlangıç işlemi, devre içerisindeki bütün yankı bastırıcıların devreden çıkarılması işlemidir. Daha sonra, yankı kanalına uygun olarak tasarlanmış bir darbe gönderilir. Bunun amacı, uzak yankının yerini tespit edebilmektir. Uzak yankının, yakın yankıdan tipik olarak 10dB daha düşük olmasına rağmen, başarımın düşmesi için bu seviye bile yeterlidir.

İzleme ve zamanlama devresi, gönderilen darbenin yayılma gecikmesinin ölçülmesi ve gidericiyi doğru kütle gecikmesi ile başlatmak için gereklidir. Giderici uygun bir şekilde tasarlandıktan sonra, verici gerçek veriye benzer uzun bir iletimi başlatır. Bu dizinin yankısı vericiye döndüğü zaman, yankı değeri giderici tarafından "gradient" algoritmasına göre katsayıların ayarlanması için kullanılır.

Tam çift yönlü iletim başladıktan sonra, yankı giderici çok daha düşük bir hızda uyarlama işlemine devam eder.

BÖLÜM 6 TAŞIYICI VE SAAT EŞZAMANLAMASI

Bir bit dizisi sayısal olarak herhangi bir coğrafya bölgesinden diğerine iletilirken, alıcının sürekli zamanda alınan işareti anlamlı veri simgeleri dizisi haline çevirebilmesi için, vericinin zaman tabanını bir şekilde yeniden oluşturması gereklidir. En genel tanımlanma şekliyle, alıcı içerisinde bu zaman tabanının yeniden oluşturulması işlemi "eşzamanlama (synchronization)" olarak adlandırılır. Bir noktadan noktaya iletim içerisindeki değişik seviyelerdeki eşzamanlama işlemleri aşağıda verilmiştir.

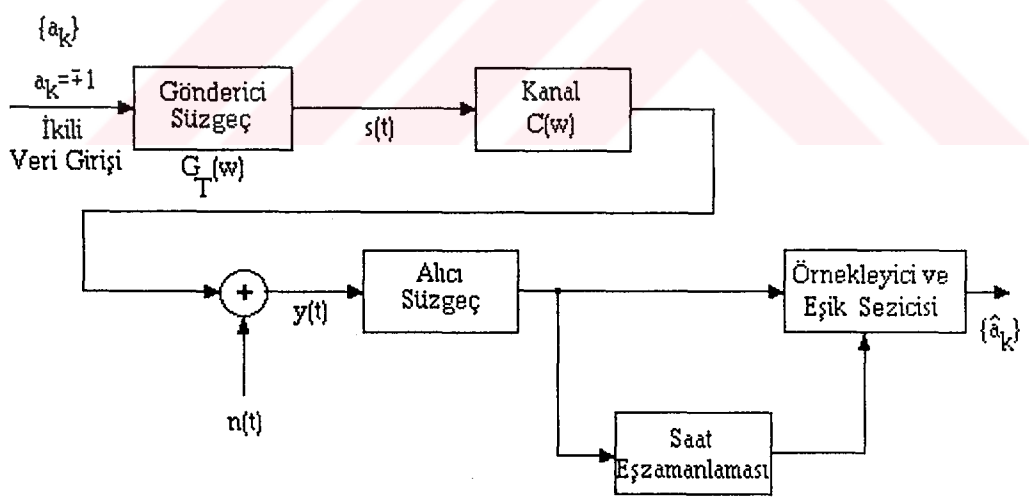
Taşıyıcı Eşzamanlaması : Eğer bit dizisi bir taşıyıcı işaret üzerine bindirilmiş ise, alıcı osilatörünün fazının ve frekansının, vericideki osilatör ile aynı duruma getirilmesi işlemidir.

Saat Eşzamanlaması : Alıcıdaki, giriş işaretinin örneklenmesini kontrol eden saatin vericideki simge hızında çalışan saate ayarlanması işlemidir.

Sözcük Eşzamanlaması : Simge gruplarının (kelime) başlangıç ve bitişlerinin belirlenmesi işlemidir.

6.1 Saat Bilgisi Eşzamanlaması

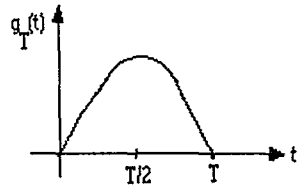
Şekil 6.1.1'de basitleştirilmiş bir temel band iletim sistemi görülmektedir.



Şekil 6.1.1
Noktadan Noktaya, Basitleştirilmiş Sayısal İletim Sistemi

Yukarıdaki sistemde kanal kodlayıcısı her T saniyede bir a_k simgesi üretir. En basit durumda a_k iki değerden birini alabilmektedir Bu bilgi daha sonra, $g_T(t)$ darbe şeklindeki işaretin genliğinin

kodlanmasında kullanılmaktadır. Bu darbe şekli şekil 6.1.2 'de görülmektedir.



Şekil 6.1.2
Temel Band Darbe Şekli

Böylece gönderilen işaret, katlanmaya uğramamış darbelerin bir toplamı şeklinde yazılabilir;

$$s(t)=\sum_k a_k g_T(t-kT-\varepsilon_0 T) \qquad (6.1.1)$$

Dikkat edilirse, s(t) bir ε_0 sabiti içermektedir. Bu sabit, verici zaman eksenindeki bilinmeyen zaman kaymasını en iyi şekilde temsil etmektedir.

s(t) işareti daha sonra bir kanal üzerinden iletilir. En basit durumda s(t) işareti herhangi bir değişikliğe uğramadan ($|C(w)=1|$) D saniye sonra alıcıya ulaşır. Daima D yayılma gecikmesi, T simge aralıklarının toplamı ve ε_c .T zaman gecikmesinin toplamı olarak yazılabilir.

$$D=MT+\varepsilon_c \cdot T \qquad (6.1.2)$$

Bu durumda alınan işaret aşağıdaki gibi yazılabilir ;

$$\begin{aligned} y(t) &= s(t-\varepsilon_c T)+n(t) \\ &= \sum_k a_k g_T(t-kT-\varepsilon_T T)+n(t) \end{aligned} \qquad (6.1.3)$$

Burada $\varepsilon_T=\varepsilon_0+\varepsilon_c$ ve n(t) ise toplamsal Gauss gürültüsüdür.

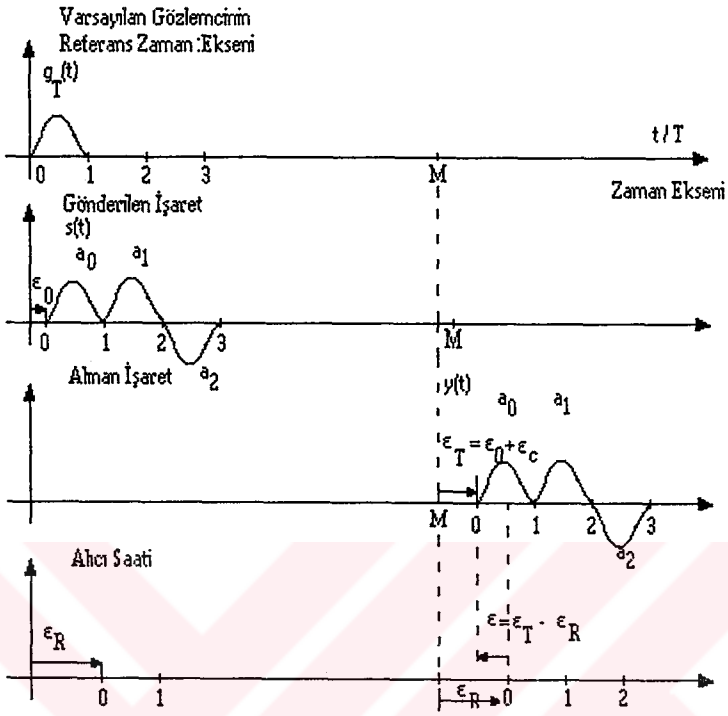
$\{a_k\}$ enformasyonunu optimal olarak alabilmek için alıcı, ulaşan işareti her T saniye aralıklarla, darbelerin genliği maksimum iken örneklemelidir. Fakat alıcı bu zamanlama bilgisini nereden alacaktır ? Bu soruna teknik açıdan sunulabilecek tek gerçeklenebilir çözüm alıcının bir şekilde bu zamanlama işaretini alınan işaretten çıkarmasıdır.

Alıcının, verici saati ile aynı saat frekansında (1/T) çalışan mükemmel bir saate sahip olduğunu ve bilinmeyen ε_R bağıl zaman kaymasına sahip olduğunu düşünelim. İşareti optimal olarak örnekleyebilmek için, alınan işaret ile alıcı saati arasındaki farklılığın alıcı tarafından bilinir hale getirilmesi gereklidir. Gözleyiciye göre olan ε_c ve ε_0 zaman kaymaları önemli değildir. Önemli olan (ε_T-ε_R) arasındaki zaman farkıdır. Alıcı saati zaman referansı olarak kullanılabilir. Bu durumda saat eşzamanlamasının tanımı aşağıdaki gibi yapılabilir ;

1. Alınan işaret ile alıcı saati arasındaki $\varepsilon=\varepsilon_T-\varepsilon_R$ zaman farkının tahmini

2. Bu tahmin kullanılarak, alınana işarete uygun bir referans zaman ekseninin üretilmesi

Şekil 6.1.4 'de saat eşzamanlaması olayı görülmektedir.



Şekil 6.1.4

Saat Eşzamanlaması

6.1.1 Bir Temel Band İşareti İçin Saat Bilgisinin Çıkarılması İşlemi

Alıcının eşzamanlama parametrelerini ve simgeleri doğru olarak tahmin edebilmesi için mümkün olan iki yaklaşım vardır. Bunlar ;

"Ad-hoc" Yaklaşımı

"Derived" Yapısı

Zamanlama bilgisinin, veri taşıyan işaretten elde edilebilmesi için pratik ve çok sık olarak kullanılan metod, "kare alma ve süzgeçleme" yaklaşımıdır. Alınan,

$y(t)=\sum_k a_k g(t-kT-\varepsilon_T T)$ işaretinin karesinin alınması ile ;

$$y^2(t)=\left[\sum_k a_k g(t-kT-\varepsilon_T T)\right]^2$$

$$= \sum_k (a_k)^2 . \; g_T^2(t-kT-\varepsilon_T T) + \sum_k \sum_{m \neq 0} a_k . a_{k+m} \, g_T(t-kT-\varepsilon_T T) . g_T(t-(k+m)T-\varepsilon_T T)$$

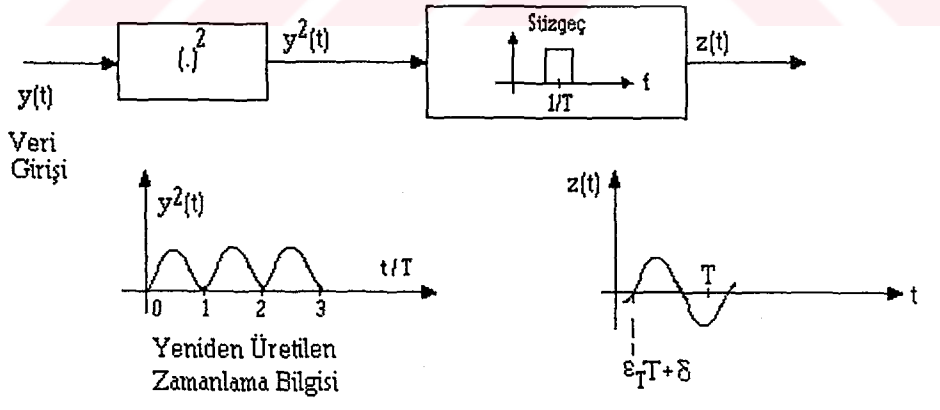
ifadesi bulunacaktır. Burada toplamsal Gauss gürültüsü ihmal edilmiştir. Veri simgelerinin istatistiksel olarak bağımsız ve eşit olasılıklı olduğunu kabul edersek ikinci taraf sıfır ortalama değere sahip olacaktır. $a_k^2 = 1$ olacağından dolayı yukarıdaki ifadenin birinci kısmı, gönderilen simgelerden bağımsız olarak periyodik bir işaret haline gelecektir ;

$$y^2(t) = \sum_k g_T^2 \; (t-kT-\varepsilon_T T) \qquad (6.1.1.1)$$

Bu işaret saat bilgisini elde edebilmek için gerekli olan gerekli enformasyonu taşımaktadır. Örneğin, $y^2(t)$ işareti 1/T de merkezlenmiş olan bir alçak geçiren süzgeçten geçirilecek olursa, süzgeç çıkışı, bir rastgele dalgalanma tarafından bozulmuş olan yaklaşık bir sinüsoidal işaret olacaktır.

$$z(t) = \sqrt{2} \, . \, A.\mathrm{Sin}\left[\frac{2\pi}{T}(t-\varepsilon_T T-\delta)\right] + \text{gürültü} \qquad (6.1.1.2)$$

δ zaman kayması tamamen $g_T(t)$ darbe şekli tarafından beklirlenmektedir ve ε_T 'den bağımsızdır. Buradaki tartışma için sıfıra indirgenebilir. Böylece z(t) işaretinin sıfır geçişleri alıcıda ihtiyaç duyulan zaman referansını sağlar. Toplamsal bozucu etki, z(t)'nin nominal sıfır geçişleri etrafında dalgalanmalara neden olacaktır.Şekil 6.1.1.1 'de "ad-hoc" yaklaşımı ile saat bilgisini çıkarma işleminin blok diyagramı görülmektedir.



Şekil 6.1.1.1

Özetle, saat eşzamanlaması sağlayıcının gerçekleştirilebilmesi için aşağıdaki iki bloğun gerçekleştirilmesi gereklidir ;

1. Belleğe sahip olmayan ve doğrusal olmayan aygıt, zamanlama dalgasının üretilebilmesi için,
2. Bir doğrusal zamanla değişmeyen süzgeç, saat oran bileşeninin çıkarılması için,

Zamanlama dalga şekli olan z(t) 'nin, nominal sıfır geçişleri etrafındaki rastgele dalgalanmalar

minimize edilmelidir. Sıfır geçişlerin rastgele dalagalanmalarını belirlemek için $g_T(t)$ darbe şeklinin ve simgelerin istatistiksel özellikleri bilinmelidir.

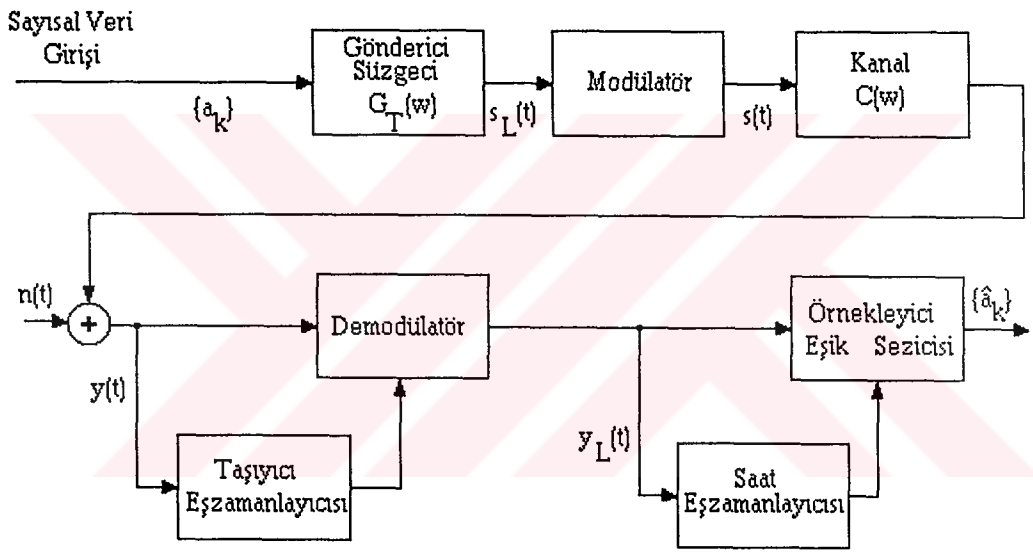
6.2 Modülasyonlu İletimde Taşıyıcı Eşzamanlaması

Eğer gönderilen işaret modülasyonlu ise, temel band işareti bir $\cos(\omega_0 t)$ taşıyıcı işareti üzerine bindirilir. Bu durumda ;

$$s(t)=\sqrt{2}\,A.s_L(t).\cos(\omega_0 t-\theta_0) \tag{6.2.1}$$

$$s_L(t)=\sum_k a_k.g_T(t-kT-\varepsilon_0 T) \tag{6.2.2}$$

olacaktır. θ_0 sabiti verici osilatörünün, varsayılan gözlemcinin osilatörünün fazına göre olan faz kaymasını göstermektedir. Şekil 6.2.1 'de modülasyonlu iletim hattı görülmektedir.



Şekil 6.2.1

Modülasyonlu İletim Sistemi

İşaretin D saniye sonra bozulmaya uğramadan alıcıya ulaştığını düşünelim. Bu durumda alınan işaret için şu ifadeyi yazabiliriz;

$$y(t)=\sqrt{2}\,A.s_L(t-\varepsilon_c T)\cos(\omega_0 t-\theta_T) \tag{6.2.3}$$

$\theta_T=\theta_0+\theta_c$ olmak üzere. Burada $\omega_0 D$ fazından modülo 2π çıkarılmış faz değeridir. Temel band işareti olan $s_L(t-\varepsilon_c T)$ 'nin yeniden elde edilebilmesi için, y(t) işaretinin yeniden temel banda döndürülmesi gereklidir.

Alıcının, ω_0 açısal frekansında çalışan ve düşünsel gözleyicinin osilatörüne göre θ_R bağıl faz kaymasına sahip olan, mükemmel bir osilatör içerdiğini kabul edelim. Alınan y(t) işaretinin, lokal olarak alıcıda üretilen $\cos(\omega_0 t-\theta_R)$ osilayonu ile çarpılması ;

$$y(t).\sqrt{2}\cos(\omega_0 t - \theta_R) = y_L(t).\cos(\theta_T - \theta_R) + \text{Yüksek Frekanslı Bileşenler} \qquad (6.2.4)$$

sonucunu verecektir. Eşitlik 6.2.4 açıkça göstermektedir ki, temel band işareti osilasyonlar mükemmel olarak paralelize edildiği zaman tam olarak elde edilmektedir ($\theta_T = \theta_R$).

Böylece alıcı, taşıyıcı eşzamanlaması görevini yerine getirmelidir. Buda aşağıdaki adımları kapsar ;

1. Faz farklılığının kestirimi ;

$\phi = \theta_T - \theta_R$

2. ϕ faz hatası sıfır olacak şekilde bir referans fazının üretilmesi

Saat ve taşıyıcı eşzamanlaması sağlandığında, bir kod sözcüğünün veya kod sözcüğü gruplarının başlangıç ve bitişi belirlenmelidir.

Bu bölümde ve daha önceki bölümde verilen saat ve taşıyıcı eşzamanlaması, eşzamanlama işlevinin tanıtımı için iyi örnekler olmalarına rağmen, bir çok pratik uygulama durumu için model çok idealize edilmiş durumdadır. İdeal modelde saat ve taşıyıcı osilatörleri vericide ve alıcıda aynı olmak üzere sabit frekanslarda çalışırlar. Bununla beraber çok hassas osilatörler bile, değişik nedenlerden kaynaklanan faz ve frekens kaymalarından etkilenmektedirler. Özel uygulamaya bağlı olarak bu sapmalar göz önüne alınmalıdır. Bu osilatöre özgü kaymalar yanında, diğer sapmalar, alıcı ve verici arasındaki bağıl hareketten kaynaklanabilir. Bunlar "Doppler Kaymaları" olarak bilinir. Matematiksel olarak bu olay nominal faz ve frekans değerinden sapmalar olarak modellenebilir. Örneğin osilatörün çıkışı ;

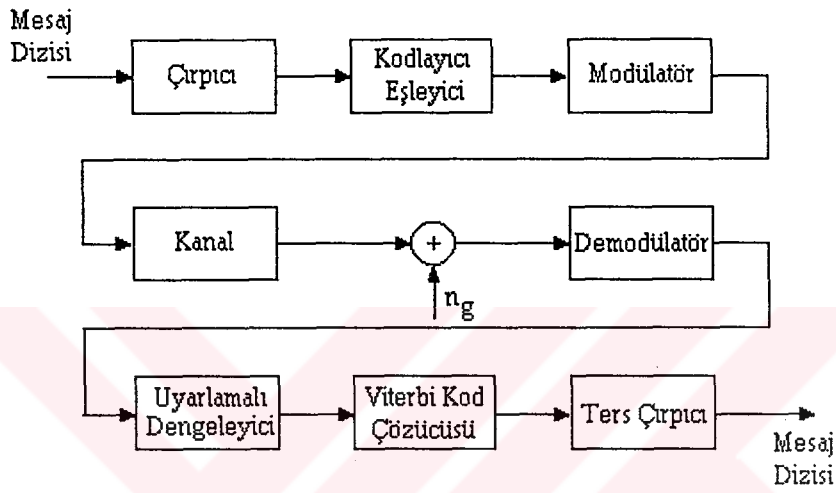
$\sqrt{2}A.\sin\phi(t)$ olarak gösterilebilir. Burada $\phi(t)$ toplam fazı göstermektedir. $\phi(t)$ fazı, mükemmel osilatör tarafından nominal açısal ω_0 frekansında üretilen bir terimden ve zamanla değişen faz sapması olan $\theta(t)$ 'den oluşmaktadır.

$$\phi(t) = \omega_0(t) + \theta_0 + \theta(t) \qquad (6.2.5)$$

$\theta(t)$ fazı, rastlantısal ve deterministik bileşenleri kapsar. Sonuç olarak alıcı, eşzamanlama işlemini zamanla değişen büyüklüklerle yerine getirmek zorundadır. Alıcı osilatörünün, $\omega_0 + \delta\omega$ tolerans bandı içerisindeki nominal değerleri almasına müsade edilmiştir. Eğer, $\theta(t)$ 'nin rastlantısal bileşenleri ihmal edilir ise alıcı $\delta\omega$ frekans sapmasını, bir faz bağdaşık osilasyonun üretilmesi için bilmesi gereklidir.

BÖLÜM 7 . BENZETİM MODELİ

Bu çalışmada ele alınan iletim sisteminin blok diyagramı şekil 7.1 'de verilmiştir. Bu sistemde altı bitlik gruplar halinde gelen veri dizileri (mesaj) kodlanıp, modülasyon işleminden geçirildikten sonra kanala verilmektedir. Alıcıda ise demodülasyon ve kod çözme işlemleri ile tekrar elde edilmektedir.



Şekil 7.1
Benzetim Modeli

7.1 Benzetim Modelinin Açıklanması

Bu alt bölümde benzetim modelinin nasıl oluşturulduğu üzerinde durulacaktır. Genel olarak mesajlar altı bitlik gruplar halinde kodlayıcıya, çırpma işlemine tabi tutulduktan sonra gelmektedirler. Bu bitlerden en düşük anlamlı olan iki tanesi, $Q1_n$ ve $Q2_n$ farksal kodlama devresine girerler. Farksal kodlama işleminin sonucunda aşağıda verilen eşitlikler ile $Y1_n$ ve $Y2_n$ bitleri elde edilir.

$$Y1_n = Q1_n + Y1_{n-1}$$
$$Y2_n = Q2_n + Y2_{n-1} + (Q1_n . Y1_{n-1}) \qquad (7.1.1)$$

Yukarıdaki eşitliklerde belirtilen "+" operatörü mod 2 de toplama işlemidir. Buradan anlaşılan, farksal kodlayıcının şimdiki girişler ile bir önceki çıkışları toplayan bir devre olduğudur. Farksal kodlama devresinde oluşturulan $Y1_n$ ve $Y2_n$ çıkışları katlamalı kodlayıcıya girer. Aşağıdaki eşitlikler katlamalı kodlayıcının durum geçiş denklemlerini vermektedir.

$$W1_{n+1}=W2_n \qquad Wa2_n=W1_n+Y1_n+Y2_n$$
$$Wa1_n=W3_n+Y2_n \qquad W3_{n+1}=Wa2_n+(Wa1_n.W2_n)$$
$$W2_{n+1}=Wa1_n+(W2_n.Y1_n) \qquad Y0_n=W2_n \tag{7.1.2}$$

Üç adet biti katlamalı kodlayıcıdan, diğer dört biti ise gelen diziden elde edilen yedi bit uzunluğundaki ikili dizi, en fazla 128 işaret elemanını gösterebilir. Burada eşleyicinin görevi bu 128 adet yedi bit uzunluğundaki dizileri iki boyutlu vektörlerden oluşan iki boyutlu bir kümeye birebir eşlemektir. Bu işlem iki boyutlu vektör kümesi bir dizi olarak tanımlanarak yapılabilir.Şöyle ki, iki boyutlu işaret kümesi gerçel ve sanal kısımlar olmak üzere iki kısma ayrılabilir. Burada kullanılan programda eşleyici bilgilerini içeren dizinin indisi ;

$$i=64.Y0_n+32.Y1_n+16.Y2_n+8.Q3_n+4.Q4_n+2.Q5_n+Q6_n \tag{7.1.3}$$

şeklinde hesaplanmaktadır. Burada $Y0_n$ en anlamlı biti temsil etmektedir. Dizinin i. elemanı bu bit dizisine karşı gelen eşleyici çıkışını gösterir. Böylece bu işaretler $S_x=Re[i]$ ve $S_y=Im[i]$ olarak elde edilebilir.

Gauss gürültüsü özel bir rastlantı süreci olup haberleşme sistemlerinde görülen gürültüler genelde bu türdendir. Bilgisayarda rastgele üretilen işaretlerin genlikleri hemen hemen düzgün olarak dağılır. Bunu bir şekilde Gauss dağılımına dönüştürmek gereklidir. Bu dönüşümü yapabilmek için aşağıdaki eşitlikler kullanılmıştır ;

$$n_x=ss*\sqrt{(-2).\log r1}*\cos 2.\pi.r2$$
$$n_y=ss*\sqrt{(-2).\log r1}*\sin 2.\pi.r2 \tag{7.1.4}$$

Burada, ss standart sapma olup, r1 ve r2 normal dağılımlı rastgele değişkenlerdir. İki boyutlu uzayda gönderilen işaretlerin simge başına ortalama gücü ;

$$P_{av}=\frac{1}{128}\sum_{i=1}^{128}(S_{x_i}^2+S_{y_i}^2) \tag{7.1.5}$$

olarak bulunabilir. Burada toplamsal Gauss gürültüsünün ortalama gücü ise $\sigma^2=N_0/2$ olarak ifade edilir. Bu ifadede σ standart sapma olup, bu durumda işaret gürültü oranı ;

$$SNR\ (dB)=10.\log_{10}(P_{av}/N_0) \tag{7.1.6}$$

olarak tanımlanır.

Kod çözme işlemi Viterbi algoritmasına göre yapılmaktadır. Bu yöntemde, alıcıda kod çözme derinliği adı verilen v sayıda işaret alındıktan sonra kod çözme işlemine başlanılmaktadır. Kod çözme işleminde gelen işaret sayısı v=10 'a ulaşana kadar aşağıda belirtilen işlemler yapılmakta ve daha sonra metriklerin karşılaştırılması işlemine geçilmektedir.

Kafes kodlayıcı yapısında bir durumdan diğerine geçerken paralel geçiş kümeleri bulunduğu için alıcıda kod çözme işlemi iki aşamada yapılmaktadır. İlk önce bütün paralel geçiş alt kümeleri içerisindeki işaret elemanları ile gönderilen işaret arasındaki Öklid uzaklıları hesaplanır ve her alt küme için bu uzaklığı minimum yapan elemanlar seçilir. Tablo 7.1.1 a, b, ve c 'de bu paralel geçiş kümeleri ve eşlendikleri bit dizileri ile işaret elemanları gösterilmiştir.

Bu geçişlerin belirlenebilmesi için şekil 7.1.1 'de gösterilen paralel durum geçiş matrisi (PDGM) tanımlanmıştır. Bu matrisin elemanları bu kod içerisinde i. durumdan j. duruma geçişlerin hangi alt geçiş kümeleri ile olduğunu göstermektedir.

Kod çözme işleminin devamında her işaretleşme aralığında tüm durumlar hesaplanır. Burada istenen, alınan v adımlı işaret dizisine en yakın yolun bulunmasıdır. En son aşamada hesaplanan dal metriklerinin en küçüğünü oluşturan durum geçişleri saptanarak, bu geçiş küme elemanlarından en küçük uzaklıklı olanları seçilir. Gönderilen v=10 adet işaretten biri bile hatalı olarak elde edilmişse, bir hata olayı gerçekleşmiş olur ve hata sayısına bir eklenir.

A	Kodlanmış Çıkışlar $Y0_n$ $Y1_n$ $Y2_n$ $Q3_n$ $Q4_n$ $Q5_n$ $Q6_n$	Kanal İşareti (Re , Im)	C	Kodlanmış Çıkışlar $Y0_n$ $Y1_n$ $Y2_n$ $Q3_n$ $Q4_n$ $Q5_n$ $Q6_n$	Kanal İşareti. (Re ,Im)
a0	0 0 0 0 0 0 0	(-8 , -3)	c0	0 0 1 0 0 0 0	(8 . 3)
a1	0 0 0 0 0 0 1	(8 , -3)	c1	0 0 1 0 0 0 1	(-8 . 3)
a2	0 0 0 0 0 1 0	(4 , -3)	c2	0 0 1 0 0 1 0	(-4 . 3)
a3	0 0 0 0 0 1 1	(4 , -7)	c3	0 0 1 0 0 1 1	(-4 . 7)
a4	0 0 0 0 1 0 0	(-4 , -3)	c4	0 0 1 0 1 0 0	(4 . 3)
a5	0 0 0 0 1 0 1	(-4 , -7)	c5	0 0 1 0 1 0 1	(4 . 7)
a6	0 0 0 0 1 1 0	(0 , -3)	c6	0 0 1 0 1 1 0	(0 . 3)
a7	0 0 0 0 1 1 1	(0 , -7)	c7	0 0 1 0 1 1 1	(0 . 7)
a8	0 0 0 1 0 0 0	(-8 , 1)	c8	0 0 1 1 0 0 0	(8 . -1)
a9	0 0 0 1 0 0 1	(8 , 1)	c9	0 0 1 1 0 0 1	(-8 . -1)
a10	0 0 0 1 0 1 0	(4 , 1)	c10	0 0 1 1 0 1 0	(-4 . -1)
a11	0 0 0 1 0 1 1	(4 , 5)	c11	0 0 1 1 0 1 1	(-4 . -5)
a12	0 0 0 1 1 0 0	(-4 , 1)	c12	0 0 1 1 1 0 0	(4 . -1)
a13	0 0 0 1 1 0 1	(-4 , 5)	c13	0 0 1 1 1 0 1	(4 . -5)
a14	0 0 0 1 1 1 0	(0 , 1)	c14	0 0 1 1 1 1 0	(0 . -1)
a_{15}	0 0 0 1 1 1 1	(0 , 5)	c15	0 0 1 1 1 1 1	(0 , -5)

Tablo 7.1.1 a

Paralel Geçiş Küme Elemanları

B	Kodlanmış Çıkışlar $Y0_n$ $Y1_n$ $Y2_n$ $Q3_n$ $Q4_n$ $Q5_n$ $Q6_n$	Kanal İşareti (Re , Im)	D	Kodlanmış Çıkışlar $Y0_n$ $Y1_n$ $Y2_n$ $Q3_n$ $Q4_n$ $Q5_n$ $Q6_n$	Kanal İşareti (Re , Im)
b0	0 1 0 0 0 0 0	(2 , -9)	d0	0 1 1 0 0 0 0	(-2 , 9)
b1	0 1 0 0 0 0 1	(2 , 7)	d1	0 1 1 0 0 0 1	(-2 , -7)
b2	0 1 0 0 0 1 0	(2 , 3)	d2	0 1 1 0 0 1 0	(-2 , -3)
b3	0 1 0 0 0 1 1	(6 , 3)	d3	0 1 1 0 0 1 1	(-6 , -3)
b4	0 1 0 0 1 0 0	(2 , -5)	d4	0 1 1 0 1 0 0	(-2 , 5)
b5	0 1 0 0 1 0 1	(6 , -5)	d5	0 1 1 0 1 0 1	(-6 , 5)
b6	0 1 0 0 1 1 0	(2 , -1)	d6	0 1 1 0 1 1 0	(-2 , 1)
b7	0 1 0 0 1 1 1	(6 , -1)	d7	0 1 1 0 1 1 1	(-6 , 1)
b8	0 1 0 1 0 0 0	(-2 , -9)	d8	0 1 1 1 0 0 0	(2 , 9)
b9	0 1 0 1 0 0 1	(-2 , 7)	d9	0 1 1 1 0 0 1	(2 , -7)
b10	0 1 0 1 0 1 0	(-2 , 3)	d10	0 1 1 1 0 1 0	(2 , -3)
b11	0 1 0 1 0 1 1	(-6 , 3)	d11	0 1 1 1 0 1 1	(6 , -3)
b12	0 1 0 1 1 0 0	(-2 , -5)	d12	0 1 1 1 1 0 0	(2 , 5)
b13	0 1 0 1 1 0 1	(-6 , -5)	d13	0 1 1 1 1 0 1	(6 , 5)
b14	0 1 0 1 1 1 0	(-2 , -1)	d14	0 1 1 1 1 1 0	(2 , 1)
b15	0 1 0 1 1 1 1	(-6 , -1)	d15	0 1 1 1 1 1 1	(6 , 1)
H	$Y0_n$ $Y1_n$ $Y2_n$ $Q3_n$ $Q4_n$ $Q5_n$ $Q6_n$	(Re , Im)	F	$Y0_n$ $Y1_n$ $Y2_n$ $Q3_n$ $Q4_n$ $Q5_n$ Q6n	(Re . Im)
h0	1 0 0 0 0 0 0	(9 , 2)	f0	1 0 1 0 0 0 0	(-9 , -2)
h1	1 0 0 0 0 0 1	(-7 , 2)	f1	1 0 1 0 0 0 1	(7 , -2)
h2	1 0 0 0 0 1 0	(-3 , 2)	f2	1 0 1 0 0 1 0	(3 , -2)
h3	1 0 0 0 0 1 1	(-3 , 6)	f3	1 0 1 0 0 1 1	(3 , -6)
h4	1 0 0 0 1 0 0	(5 , 2)	f4	1 0 1 0 1 0 0	(-5 , -2)
h5	1 0 0 0 1 0 1	(5 , 6)	f5	1 0 1 0 1 0 1	(-5 , -6)
h6	1 0 0 0 1 1 0	(1 , 2)	f6	1 0 1 0 1 1 0	(-1 , -2)
h7	1 0 0 0 1 1 1	(1 , 6)	f7	1 0 1 0 1 1 1	(-1 , -6)
h8	1 0 0 1 0 0 0	(9 , -2)	f8	1 0 1 1 0 0 0	(-9 , 2)
h9	1 0 0 1 0 0 1	(-7 , -2)	f9	1 0 1 1 0 0 1	(7 , 2)
h10	1 0 0 1 0 1 0	(-3 , -2)	f10	1 0 1 1 0 1 0	(3 , 2)
h11	1 0 0 1 0 1 1	(-3 , -6)	f11	1 0 1 1 0 1 1	(3 , 6)
h12	1 0 0 1 1 0 0	(5 , -2)	f12	1 0 1 1 1 0 0	(-5 , 2)
h13	1 0 0 1 1 0 1	(5 , -6)	f13	1 0 1 1 1 0 1	(-5 , 6)
h14	1 0 0 1 1 1 0	(1 , -2)	f14	1 0 1 1 1 1 0	(-1 , 2)
h15	1 0 0 1 1 1 1	(1 , -6)	f15	1 0 1 1 1 1 1	(-1 , 6)

Tablo 7.1.1 b

Paralel Geçiş Küme Elemanları

E	Kodlanmış Çıkışlar $Y0_n$ $Y1_n$ $Y2_n$ $Q3_n$ $Q4_n$ $Q5_n$ $Q6_n$	Kanal İşareti (Re , Im)	G	Kodlanmış Çıkışlar $Y0_n$ $Y1_n$ $Y2_n$ $Q3_n$ $Q4_n$ $Q5_n$ $Q6_n$	Kanal İşareti (Re , Im)
e0	1 1 0 0 0 0 0	(-3 , 8)	g0	1 1 1 0 0 0 0	(3 . -8)
e1	1 1 0 0 0 0 1	(-3 , -8)	g1	1 1 1 0 0 0 1	(3 . 8)
e2	1 1 0 0 0 1 0	(-3 , -4)	g2	1 1 1 0 0 1 0	(3 . 4)
e3	1 1 0 0 0 1 1	(-7 , -4)	g3	1 1 1 0 0 1 1	(7 . 4)
e4	1 1 0 0 1 0 0	(-3 , 4)	g4	1 1 1 0 1 0 0	(3 . -4)
e5	1 1 0 0 1 0 1	(-7 , 4)	g5	1 1 1 0 1 0 1	(7 . -4)
e6	1 1 0 0 1 1 0	(-3 , 0)	g6	1 1 1 0 1 1 0	(3 . 0)
e7	1 1 0 0 1 1 1	(-7 , 0)	g7	1 1 1 0 1 1 1	(7 . 0)
e8	1 1 0 1 0 0 0	(1 , 8)	g8	1 1 1 1 0 0 0	(-1 . -8)
e9	1 1 0 1 0 0 1	(1 , -8)	g9	1 1 1 1 0 0 1	(-1 . 8)
e10	1 1 0 1 0 1 0	(1 , -4)	g10	1 1 1 1 0 1 0	(-1 . 4)
e11	1 1 0 1 0 1 1	(5 , -4)	g11	1 1 1 1 0 1 1	(-5 . 4)
e12	1 1 0 1 1 0 0	(1 , 4)	g12	1 1 1 1 1 0 0	(-1 . -4)
e13	1 1 0 1 1 0 1	(5 , 4)	g13	1 1 1 1 1 0 1	(-5 . -4)
e14	1 1 0 1 1 1 0	(1 , 0)	g14	1 1 1 1 1 1 0	(-1 , 0)
e15	1 1 0 1 1 1 1	(5 , 0)	g15	1 1 1 1 1 1 1	(-5 , 0)

Tablo 7.1.1 c

Paralel Geçiş Küme Elemanları

	0	1	2	3	4	5	6	7
0	1	3	4	2	0	0	0	0
1	4	2	1	3	0	0	0	0
2	0	0	0	0	5	8	6	7
3	0	0	0	0	7	6	8	5
4	3	1	2	4	0	0	0	0
5	2	4	3	1	0	0	0	0
6	0	0	0	0	8	5	7	6
7	0	0	0	0	6	7	5	8

Şimdiki Durum

Bir Önceki Durum

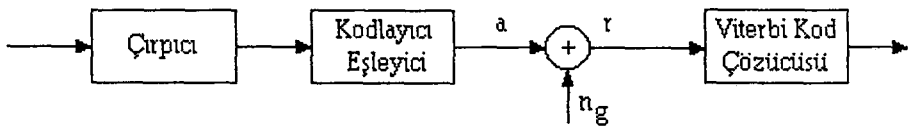
Şekil 7.1.1

Paralel Durum Geçiş Matrisi

7.2 Toplamsal Gauss Gürültülü İdeal Karakteristikli Kanallar İçin Benzetim Modeli

Eşleyici devresinden gelen modülasyon giriş işaretlerine, dik genlik modülasyonu uygulanmak suretiyle kanal giriş işaretlerinin elde edilebileceğini biliyoruz. Bu bileşenlere daha sonra toplamsal

Gauss gürültüsü de eklenebilir. Bu modelde bazı ihmaller sonucunda, analog süreç ayrık bir süreç haline getirilebilir. Bu şekilde elde edilecek olan model incelemeyi kolaylaştıracaktır. Burada alçak geçiren süzgeçler, modülasyon ve demodülasyon işlemleri ile kanalın bozucu etkisi ve gecikme göz önüne alınmamıştır. Şekil 7.2.1 'de benzetim modelinin şekli görülmektedir.



Şekil 7.2.1
Toplamsal Gauss Gürültülü İdeal Karakteristikli Kanal Modeli

Modelde, a eşleyici çıkış işaretleri vektörünü, n_g Gauss gürültü vektörünü ve r ise alıcıya gelen işaret vektörünü göstermektedir. Bu tanımlamalara göre aşağıdaki ifade yazılabilir ;

$$r = a + n_g \qquad (r_x , r_y) = (a_x , a_y) + (n_x , n_y) \qquad (7.2.1)$$

Benzetim modelinin akış diyagramı bir sonraki sayfadan başlamak üzere verilmiştir. Model için oluşturulan programın tam listesi, "sim1.c" adı ile ekler bölümünde verilmiştir.

Bu ve bundan sonraki alt bölümlerde oluşturulan benzetim modelleri için hazırlanan programlar, turbo C programlama dilinde yazılmışlardır.

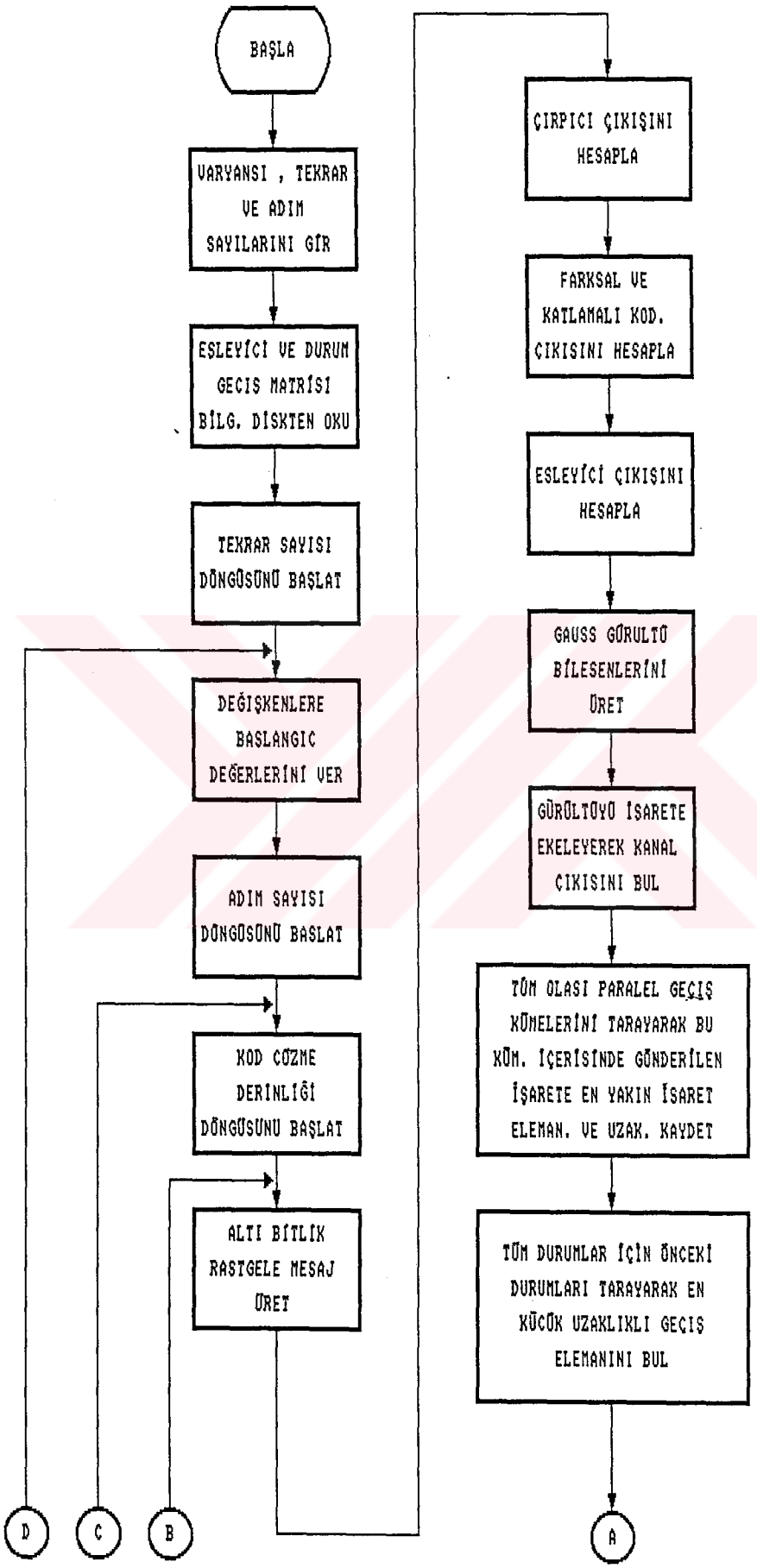
Program 8 farklı işaret gürültü oranı ve 200 adım uzunluğu için çalıştırılmış ve şekil 7.2.2 'de görülen hata başarım eğrisi elde edilmiştir.

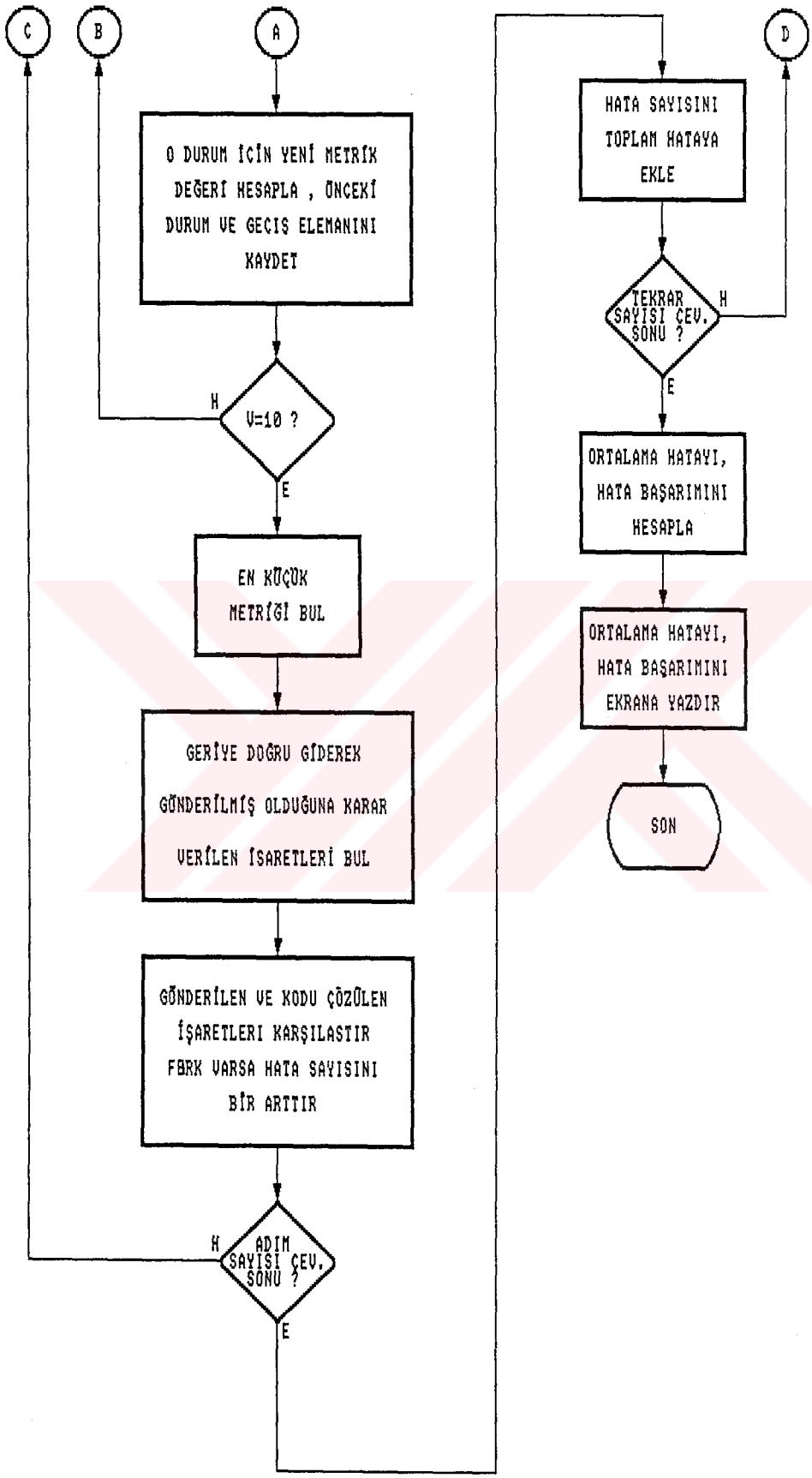
Her adımda, kod çözme derinliği kadar simge gönderilmekte ve alıcıda, gönderilen simgeler ile kodu çözülen simgeler karşılaştırılmaktadır. 8 farklı varyans değeri için işaret-gürültü oranlarının bulunabilmesi için, bit başına düşen güç miktarı , $P_b = P_{av} / b$ ifadesi ile hesaplanabilir. 14.400 hızındaki modemin simge başına ortalama gücü, 7.1.5 ifadesi kullanılarak $P_{av} = 41.0$ W olarak bulunacaktır. Her işaretleşme aralığında gönderilen bit sayısı b=6 olduğu için, P_b yaklaşık olarak 6.833 W olarak bulunacaktır.

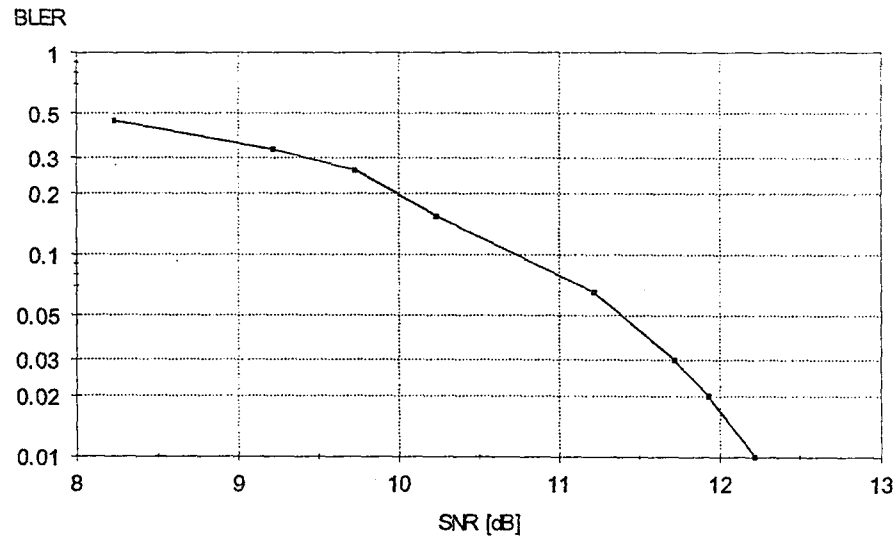
Gauss gürültüsünün ortalama gücü, standart sapmanın varyansın karekökü olduğu hatırlanarak, daha öncede verilen $\sigma^2 = N_0/2$ ifadesi ile hesaplanabilir. Bu durumda işaret-gürültü oranları,

$$SNR = 10 * \log_{10} (P_b / N_0) \quad [dB]$$

ifadesi ile bulunabilir.


BAŞLA
VARYANSI , TEKRAR VE ADIM SAYILARINI GİR
EŞLEYİCİ VE DURUM GEÇİŞ MATRİSİ BİLG. DİSKTEN OKU
TEKRAR SAYISI DÖNGÜSÜNÜ BAŞLAT
DEĞİŞKENLERE BAŞLANGIÇ DEĞERLERİNİ VER
ADIM SAYISI DÖNGÜSÜNÜ BAŞLAT
KOD ÇÖZME DERİNLİĞİ DÖNGÜSÜNÜ BAŞLAT
ALTI BİTLİK RASTGELE MESAJ ÜRET
D
C
B
ÇIRPICI ÇIKIŞINI HESAPLA
FARKSAL VE KATLAMALI KOD. ÇIKIŞINI HESAPLA
EŞLEYİCİ ÇIKIŞINI HESAPLA
GAUSS GÜRÜLTÜ BİLEŞENLERİNİ ÜRET
GÜRÜLTÜYÜ İŞARETE EKELEYEREK KANAL ÇIKIŞINI BUL
TÜM OLASI PARALEL GEÇİŞ KÜMELERİNİ TARAYARAK BU KÜM. İÇERİSİNDE GÖNDERİLEN İŞARETE EN YAKIN İSARET ELEMAN. VE UZAK. KAYDET
TÜM DURUMLAR İÇİN ÖNCEKİ DURUMLARI TARAYARAK EN KÜÇÜK UZAKLIKLI GEÇİŞ ELEMANINI BUL
A


C
B
A
D
O DURUM İÇİN YENİ METRİK DEĞERİ HESAPLA , ÖNCEKİ DURUM VE GEÇİŞ ELEMANINI KAYDET
V=10 ?
H
E
EN KÜÇÜK METRİĞİ BUL
GERİYE DOĞRU GİDEREK GÖNDERİLMİŞ OLDUĞUNA KARAR VERİLEN İSARETLERİ BUL
GÖNDERİLEN VE KODU ÇÖZÜLEN İŞARETLERI KARŞILASTIR FARK VARSA HATA SAYISINI BİR ARTTIR
ADIM SAYISI ÇEV. SONU ?
H
E
HATA SAYISINI TOPLAM HATAYA EKLE
TEKRAR SAYISI ÇEV. SONU ?
H
E
ORTALAMA HATAYI, HATA BAŞARIMINI HESAPLA
ORTALAMA HATAYI, HATA BAŞARIMINI EKRANA YAZDIR
SON



Şekil 7.2.2

İdeal Karakteristikli, Gauss Gürültülü Kanal İçin Hata Başarım Eğrisi

Benzetim modelinde hata başarımı incelenirken, v=10 uzunluklu simge dizileri bir blok olarak kabul edilmiş ve blok içerisindeki bir simge bile hatalı karar verilmiş ise, simge bloğu hatalı olarak kabul edilmiştir. Bu durumda hata başarımı, "Blok Hata Oranı (=Block error rate) "olarak ele alınmıştır. BLER, hatalı blok sayısının toplam blok sayısına oranı olarak tanımlanmıştır.

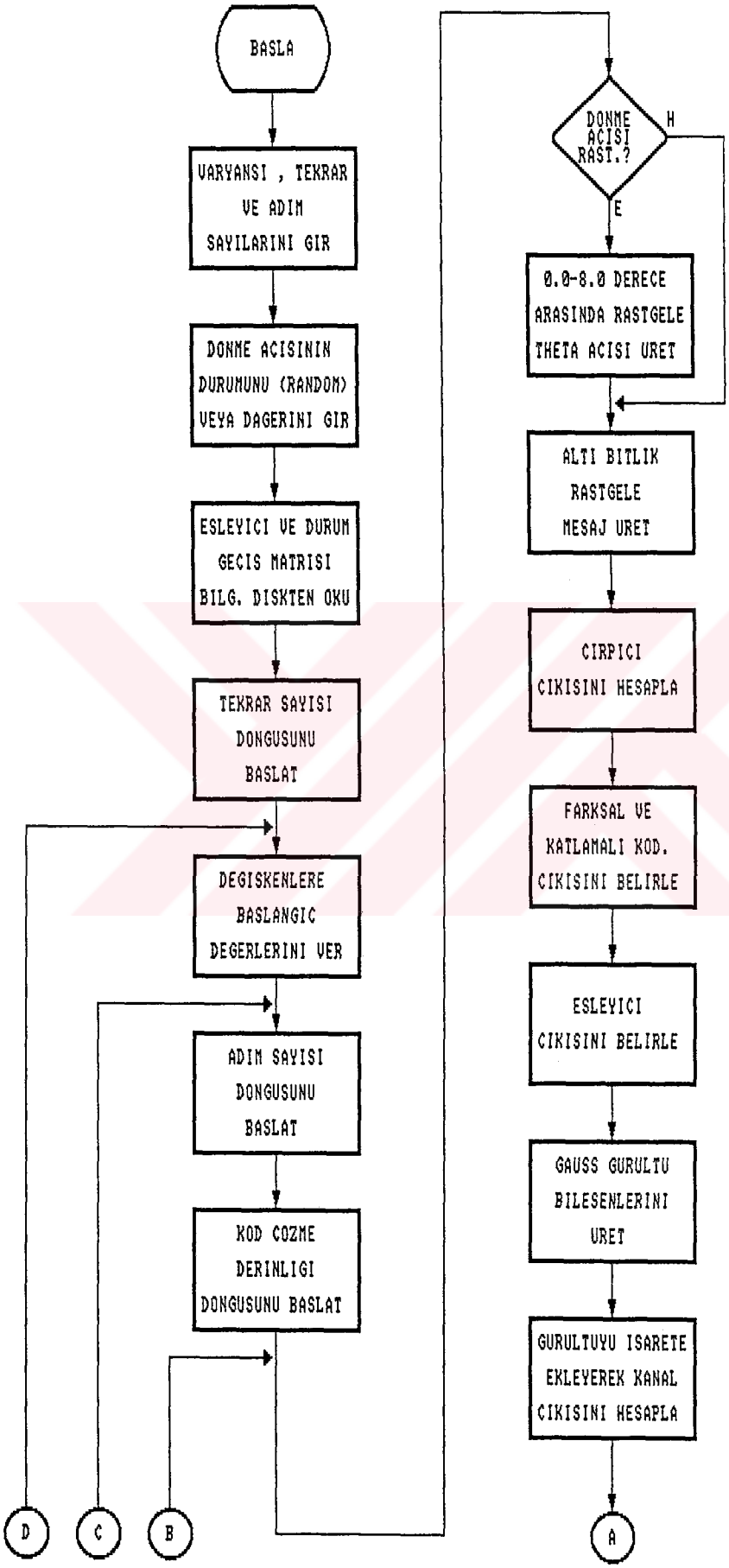
7.2.1 İşaret Elemanlarının Belirli Veya Rastgele Açılarla Dönmelerinin Hata Başarımına Etkileri

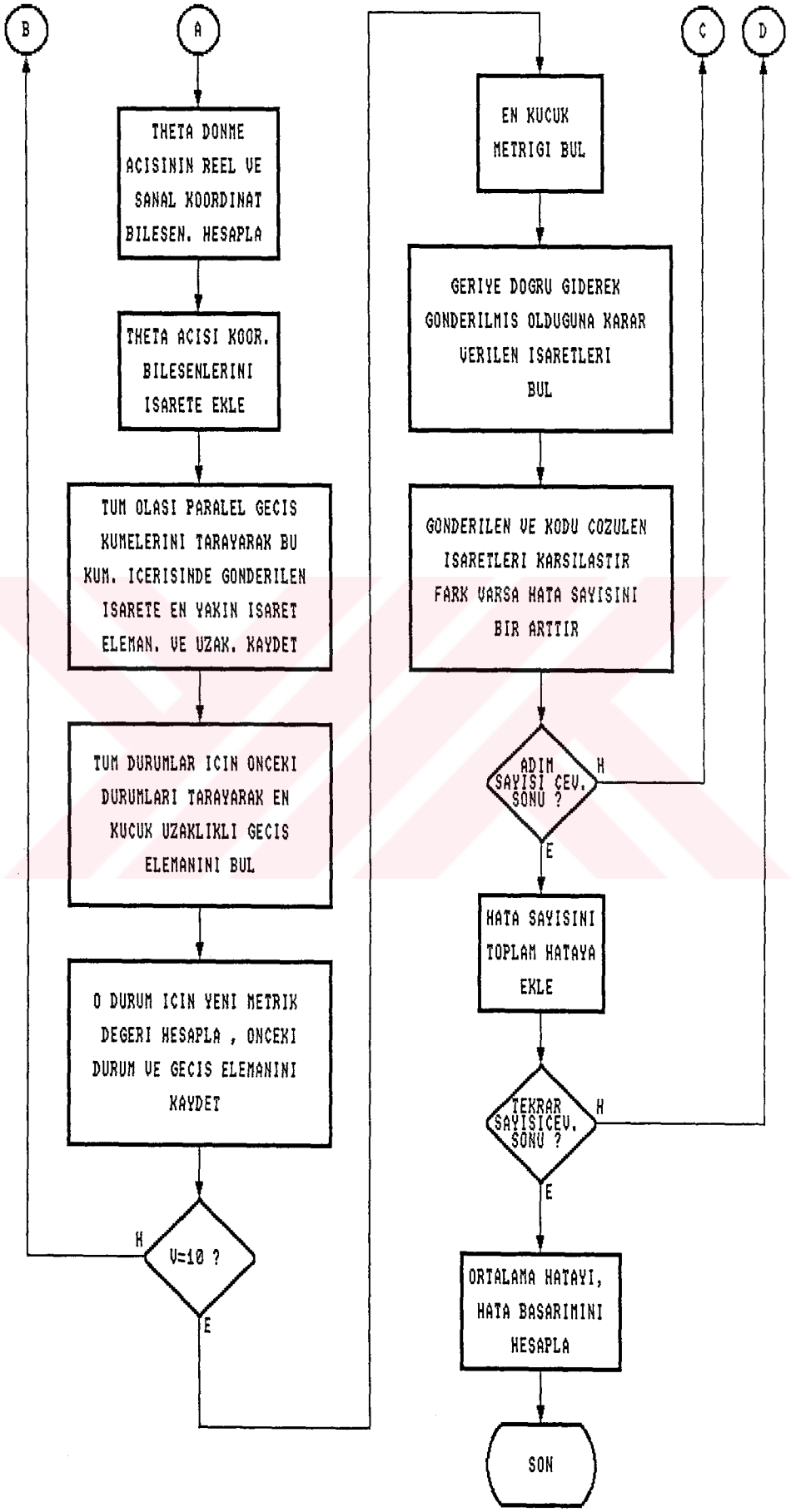
Birinci benzetim modelinin bu kısmında, işaret uzayı içerisindeki işaret elemanlarının kanalın neden olduğu bozucu etkiler ve demodülasyon işlemi sırasında ortaya çıkan faz belirsizlikleri nedeniyle işaret uzayı içerisinde, 3, 5, 6 derecelik sabit veya rastgele açılarla dönmeleri sonucu hata başarımının değişimi incelenmiştir. Rastgele açı dönmeleri, 0.0 ile 8.0 derece arasında değerler alabilen, düzgün bir dağılıma sahiptirler.

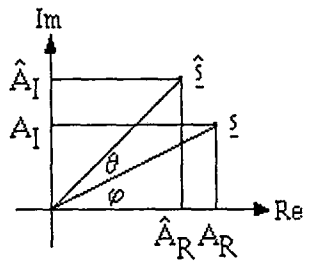
Dönme gerçekleşmeden önceki işaret elemanı vektörünü s ve koordinatlarınıda (A_R , A_I) ile gösterirsek, dönme sonucu oluşan yeni işaret elemanı vektörü ve koordinatları 7.2.1.1 'de verilen ifade yardımıyla bulunabilir.

Bu inceleme için kullanılan programın tam listesi "sim1a.c" adı ile ekler bölümünde verilmiştir. Programın akış diyagramı bir sonraki sayfadan itibaren verilmiştir.

Akış diyagramı bir önceki incelemeden çok az farklılık göstermektedir. Farklı olarak, işaret elemanları, sabit değerde verilen veya 0.0-8.0 derece arasında düzgün dağılımlı olarak üretilen θ açılarının değeri kadar işaret uzayı içerisinde döndürülmektedir.


BASLA
VARYANSI , TEKRAR VE ADIM SAYILARINI GIR
DONME ACISININ DURUMUNU (RANDOM) VEYA DAGERINI GIR
ESLEYICI VE DURUM GEÇIS MATRISI BILG. DISKTEN OKU
TEKRAR SAYISI DONGUSUNU BASLAT
DEGISKENLERE BASLANGIC DEGERLERINI VER
ADIM SAYISI DONGUSUNU BASLAT
KOD COZME DERINLIGI DONGUSUNU BASLAT
D
C
B
DONME ACISI RAST.?
H
E
0.0-8.0 DERECE ARASINDA RASTGELE THETA ACISI URET
ALTI BITLIK RASTGELE MESAJ URET
CIRPICI CIKISINI HESAPLA
FARKSAL VE KATLAMALI KOD. CIKISINI BELIRLE
ESLEYICI CIKISINI BELIRLE
GAUSS GURULTU BILESENLERINI URET
GURULTUYU ISARETE EKLEYEREK KANAL CIKISINI HESAPLA
A


B
A
THETA DONME ACISININ REEL VE SANAL KOORDINAT BILESEN. HESAPLA
THETA ACISI KOOR. BILESENLERINI ISARETE EKLE
TUM OLASI PARALEL GECIS KUMELERINI TARAYARAK BU KUM. ICERISINDE GONDERILEN ISARETE EN YAKIN ISARET ELEMAN. VE UZAK. KAYDET
TUM DURUMLAR ICIN ONCEKI DURUMLARI TARAYARAK EN KUCUK UZAKLIKLI GECIS ELEMANINI BUL
O DURUM ICIN YENI METRIK DEGERI HESAPLA , ONCEKI DURUM VE GECIS ELEMANINI KAYDET
V=10 ?
H
E
EN KUCUK METRIGI BUL
GERIYE DOGRU GIDEREK GONDERILMIS OLDUGUNA KARAR VERILEN ISARETLERI BUL
GONDERILEN VE KODU COZULEN ISARETLERI KARSILASTIR FARK VARSA HATA SAYISINI BIR ARTTIR
ADIM SAYISI CEV. SONU ?
H
E
HATA SAYISINI TOPLAM HATAYA EKLE
TEKRAR SAYISICEV. SONU ?
H
E
ORTALAMA HATAYI, HATA BASARIMINI HESAPLA
SON
C
D



Şekil 7.2.1.1.

$$\underline{s}=(A_R, A_I) \qquad \hat{\underline{s}}=(\hat{A}_R,\hat{A}_I)$$

$$A_R=|\underline{s}|\cos\varphi\ ,\ A_I=|\underline{s}|\sin\varphi$$

$$\hat{A}_R=|\underline{s}|\cos(\varphi+\theta)=|\underline{s}|\cos\varphi\cos\theta-|\underline{s}|\sin\varphi\sin\theta=A_R\cos\theta-A_I\sin\theta$$

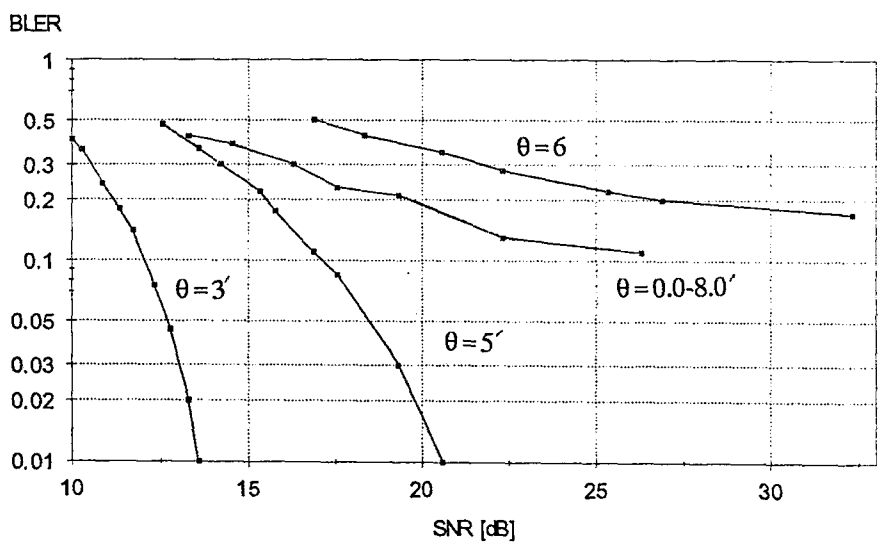
$$\hat{A}_I=|\underline{s}|\sin(\varphi+\theta)=|\underline{s}|\sin\varphi\cos\theta+|\underline{s}|\cos\varphi\sin\theta=A_I\cos\theta+A_R\sin\theta$$

$$\hat{A}_R=A_R\cos\theta-A_I\sin\theta$$

$$\hat{A}_I=A_I\cos\theta+A_R\sin\theta \qquad (7.2.1.1)$$

Böylece vericiden gönderilen işaret elemanları, 7.2.1.1 'deki ifadeye göre θ açısı kadar döndürülecek ve daha sonra elde edilen gerçel ve sanal bileşenlere, toplamsal Gauss gürültüsünün gerçel ve sanal bileşenleri eklenecektir.

Şekil 7.2.1.2 'de, θ açısının 3, 5, 6 derece olarak seçilmesi ve 0.0-8.0 derece arasında rastgele değişmesi durumunda elde edilen hata başarım eğrileri görülmektedir.



Şekil 7.2.1.2

θ dönme açısının 0.0-8.0 derece arasında düzgün dağılımlı rastgele olarak değiştiği çalışmada, varyans değeri sıfır olarak alınır ise elde edilen hata başarımı 0.1 olarak bulunmuştur.

7.3 Bozucu Kanal İçeren İletim Sistemi İçerisinde, Kanalın Bozucu Etkisinin Dengeleyici İle Giderilmesi

Benzetim modelinin bu bölümünde kanalın bozucu etkisi göz önüne alınmıştır. Aynı zamanda bu bozucu etki bir uyarlamalı dengeleyici kullanılarak giderilmeye çalışılmıştır.

Bozucu etkiyi ortaya çıkaran iletim kanalı frekans bölgesinde genlik ve faz eğrileri ile gösterilebilir. Şekil 7.3.1 ve 7.3.2 'de, bu benzetim modeli için kullanılan iki farklı iletim kanalının karakteristikleri görülmektedir. Bu iki kanalın impuls yanıtları ile T=1/1600 zaman aralıklarındaki impuls yanıtı örnekleri bulunup, enterpolasyon yardımıyla aynı impuls yanıtlarının T=1/2400 simge aralıklarındaki örnek değerleri elde edilmiştir. Tablo 7.3.1 'de, örnek olarak bozucu kanal$_2$ için elde edilen impuls yanıtı örneklerinin gerçel ve sanal bileşenleri verilmiştir.

Bu modellemede kanal giriş işareti ;

$a(t)=a_R(t)+ja_I(t)$ olarak gösterilebilir. Çünkü dik genlik modülasyonunda kanala verilen işaret birbirinden 90 derece farklı iki taşıyıcıyı içermektedir. İletilen bilgi bu taşıyıcılar üzerinde bölüşülmektedir. Bu durumda kanal girişindeki işaret kompleks bir işaret olarak düşünülebilir. Kanal da kompleks bir işlev gibi davranacaktır. Kompleks kanal impuls yanıtı ise, $h(t)=h_R(t)+jh_I(t)$ olarak ifade edilebilir. Hilbert dönüşümü yardımıyla kanalın kompleks temel band eşdeğeri ;

$$h(t)=1/\pi t.h(t)$$
$$h_R=h(t).\cos\omega_c t+h(t).\sin\omega_c t$$
$$h_I=-h(t).\cos\omega_c t+h(t)\sin\omega_c t \qquad 7.3.1$$

olarak hesaplanabilir. Görüldüğü gibi, taşıyıcı frekansı eşitliklerde görülmektedir. Modülasyon ve demodülasyon işlemleride kanal impuls yanıtı içerisinde yer almaktadır.

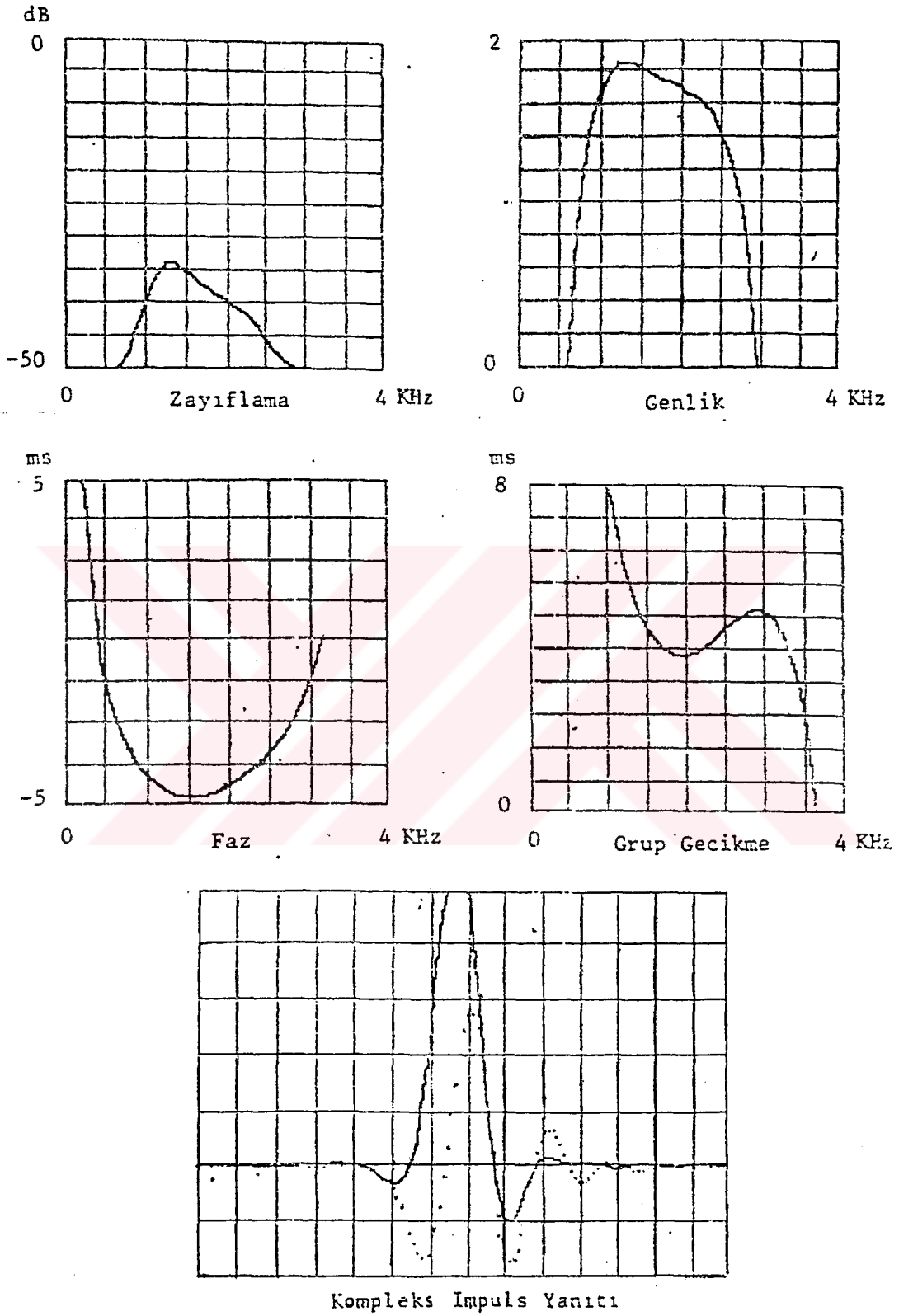
Bu durumda kanal çıkış işareti olan x(t) , $x(t)=a(t)*h(t)$ olarak bulunabilir. Yani,

$$X_R(f)+jX_I(f)=[A_R(f)+jA_I(f)].[H_R(f)+jH_I(f)]$$
$$=(A_R(f).H_R(f)-A_I(f).H_I(f))+j\,(A_I(f).H_R(f)+A_R(f).H_I(f))$$
$$X_R(f) \qquad\qquad X_I(f)$$

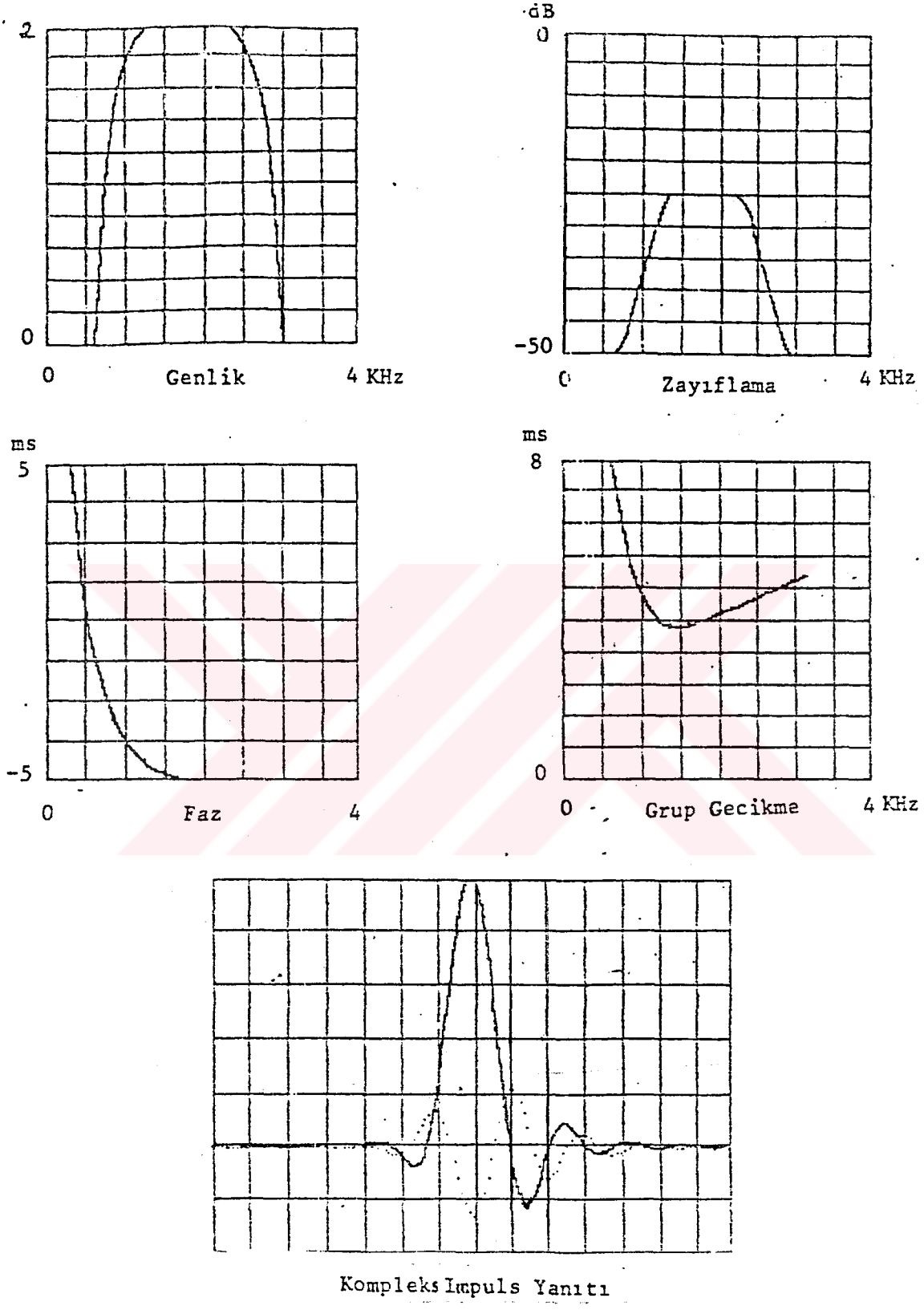
7.3.2

Bu çalışmada bozucu kanallar olan "kanal1" ve "kanal2" 'nin impuls yanıtı örneklerinin 1024 noktalı kompleks Fourier dönüşümü alınarak kanal1.dat ve kanal2.dat isimli iki ayrı dosyada saklanmıştır. Bozucu kanalların işaret elemanları üzerindeki etkisi frekans domeninde gerçekleştirilmiştir.

7.3.2 'deki ifade ile X(f) işareti her frekans örneği için elde edildikten sonra, x(t) işareti, $x(t)=IFFT[A(f).H(f)]$ ifadesinden bulunabilir. Dengeleyicinin impuls yanıtı $c(t)=c_R(t)+jc_I(t)$ olarak düşünülerek,dengeleyici çıkışındaki z(t) kompleks işareti , $z(t)=c(t)*x(t)$ olarak bulunabilir.



Şekil 7.3.1
Bozucu Kanal$_1$ Karakteristikleri



Şekil 7.3.2
Bozucu Kanal$_2$ Karakteristikleri

h_R	h_I
- 0.1473680840750D - 03	0.7341973301252D - 03
- 0.5330120320000D - 03	0.5375299300000D - 03
- 0.2751174111360D - 03	- 0.9197576559010D - 03
- 0.3314238152731D - 02	- 0.7603641145050D - 03
0.3204867757182D - 02	- 0.2328359629386D - 02
0.2222783023134D - 01	- 0.3896339009620D - 02
- 0.4341701801177D - 01	0.3558669187799D - 01
- 0.6777609080474D - 01	- 0.4618212677603D - 01
0.2401613092898D 00	- 0.3589317432155D 00
0.8134989408872D 00	- 0.1334820103993D - 01
0.1000000000000D 01	0.5537353738803D 00
0.1763375491728D 00	0.1428697289024D - 01
- 0.2616273029660D 00	- 0.2525343837139D 00
0.2140340182006D - 01	0.9847061689583D - 01
0.5646403937685D - 01	0.3301931224478D - 01
- 0.1345175674643D - 01	- 0.6191386752801D - 01
- 0.9254691548061D - 02	0.2177254538719D - 01
- 0.1210531694289D - 03	0.1710702600420D - 01
0.2385540125815D - 02	- 0.1024350523186D - 01
0.1958242729589D - 02	- 0.3856059127117D - 02
- 0.7556790257783D - 06	0.9801914775523D - 03

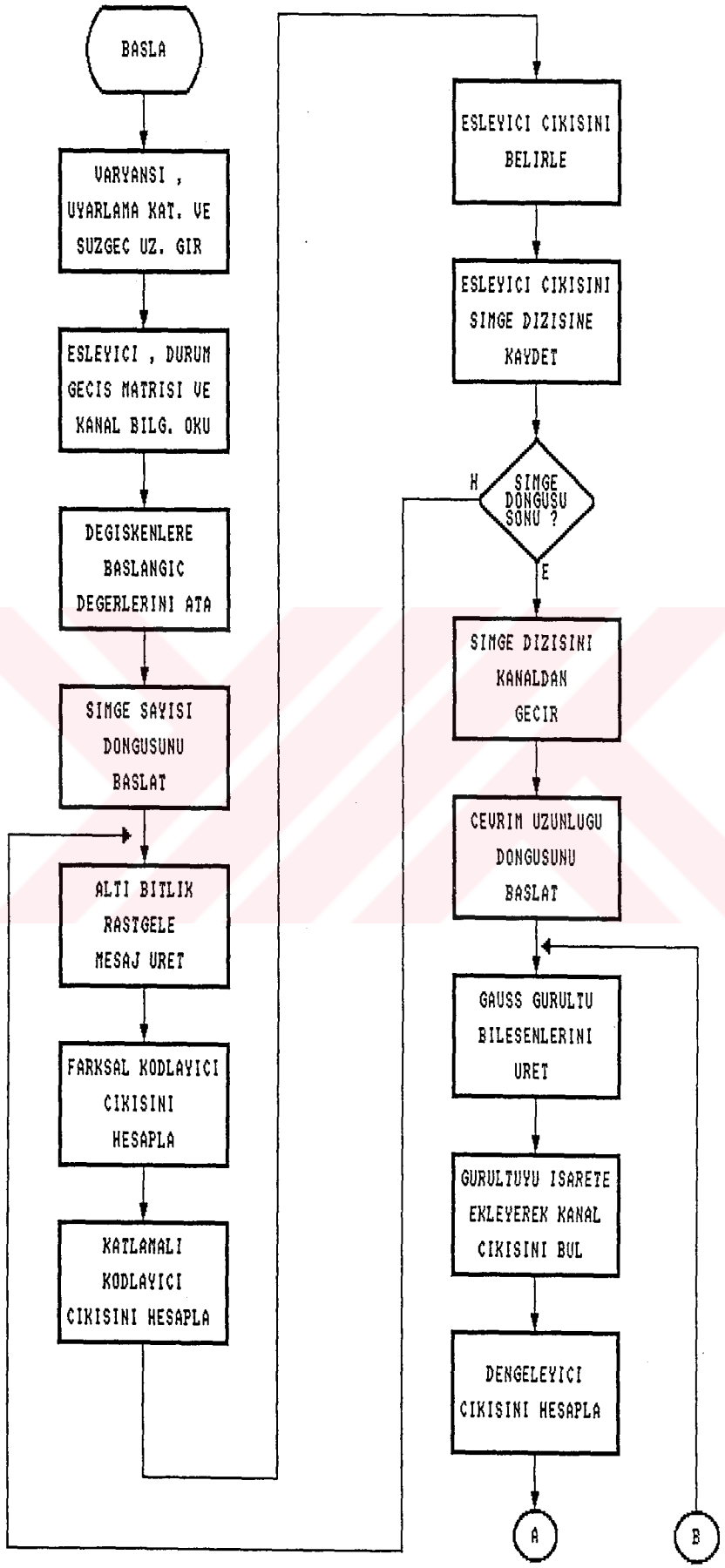
Tablo 7.3.1
Bozucu Kanal$_2$ İmpuls Yanıtı Örnekleri

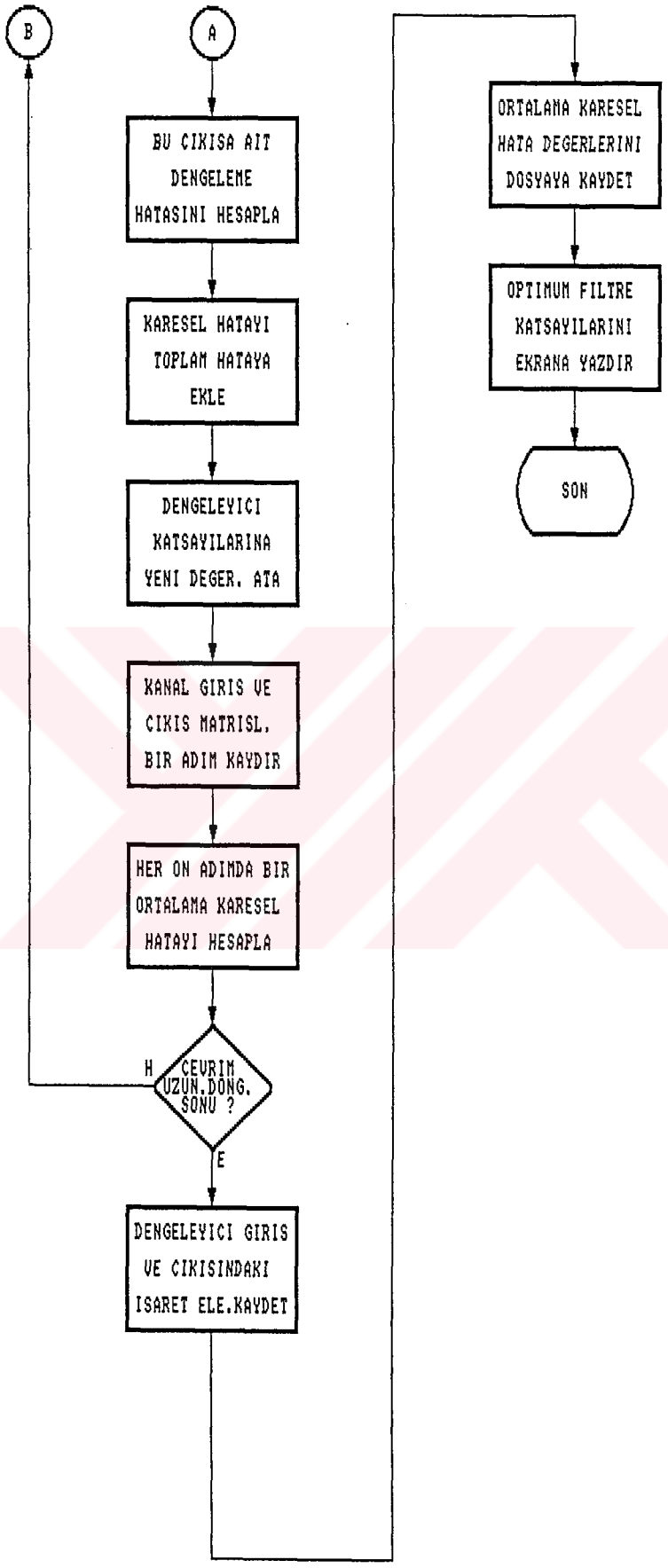
7.3.1 Doğrusal Dengeleyici Çıkışındaki Ortalama Hatanın İterasyon Sayısı Ve Uyarlama Parametreleri İle Değişiminin İncelenmesi

Sisteme kanal ve uyarlamalı dengeleyici eklendikten sonra, her iki kanal için dengeleyicinin ortalama karesel hatasının iterasyon sayısı ile değişimi incelenmiştir. Dengeleyici çıkışındaki ortalama karesel hatanın, varyans ve uyarlama parametreleri olan, uyarlama katsayısı ve süzgeç uzunluğu ile olan değişimi de incelenmiştir.

Bu bölüm için hazırlanan programın tam listesi "sim2.c" adı altında ekler bölümünde verilmiştir. Programın akış diyagramı bir sonraki sayfadan itibaren verilmiştir. Çalışmanın başında , varyans ve uyarlama parametreleri sabit olarak tutularak, bozucu kanal$_1$ ve kanal$_2$ için ortalama karesel hatanın iterasyon sayısı ile değişimi incelenmiştir. Daha sonra kanal$_2$ için, sırasıyla varyans ve uyarlama parametrelerinin değiştirilmesi ile karesel ortalama hatanın değişimi incelenmiştir. Burada hata değeri bulunurken, ard arda gelen on adet karesel hata değeri toplanıp on'a bölünmektedir.

Yapılan incelemeler için çevrim uzunluğu 1000 olarak alınmıştır. Yani 1000 tane işaret elemanı üretilerek bozucu kanaldan geçirilmiş ve dengeleyici girişine uygulanmıştır.

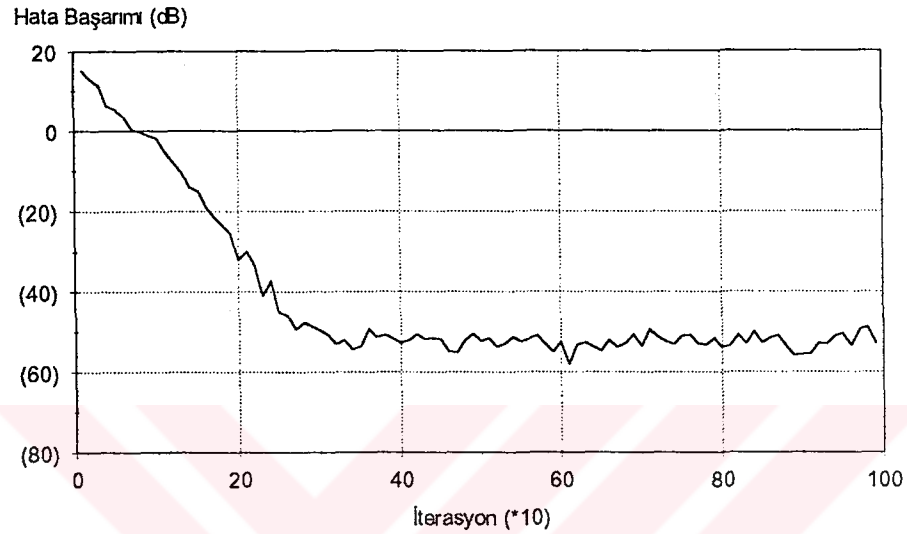

BASLA
VARYANSI , UYARLAMA KAT. VE SUZGEC UZ. GIR
ESLEYICI , DURUM GECIS MATRISI VE KANAL BILG. OKU
DEGISKENLERE BASLANGIC DEGERLERINI ATA
SIMGE SAYISI DONGUSUNU BASLAT
ALTI BITLIK RASTGELE MESAJ URET
FARKSAL KODLAYICI CIKISINI HESAPLA
KATLANALI KODLAYICI CIKISINI HESAPLA
ESLEYICI CIKISINI BELIRLE
ESLEYICI CIKISINI SIMGE DIZISINE KAYDET
SIMGE DONGUSU SONU ?
H
E
SIMGE DIZISINI KANALDAN GECIR
CEVRIM UZUNLUGU DONGUSUNU BASLAT
GAUSS GURULTU BILESENLERINI URET
GURULTUYU ISARETE EKLEYEREK KANAL CIKISINI BUL
DENGELEYICI CIKISINI HESAPLA
A
B


B
A
BU CIKISA AIT DENGELEME HATASINI HESAPLA
KARESEL HATAYI TOPLAM HATAYA EKLE
DENGELEYICI KATSAYILARINA YENI DEGER. ATA
KANAL GIRIS VE CIKIS MATRISL. BIR ADIM KAYDIR
HER ON ADIMDA BIR ORTALAMA KARESEL HATAYI HESAPLA
H
CEVRIM UZUN.DONG. SONU ?
E
DENGELEYICI GIRIS VE CIKISINDAKI ISARET ELE.KAYDET
ORTALAMA KARESEL HATA DEGERLERINI DOSYAYA KAYDET
OPTIMUM FILTRE KATSAYILARINI EKRANA YAZDIR
SON

Her iki kanal için sabit olarak seçilen varyans ve parametre değerleri aşağıda verilmiştir. Şekil 7.3.1.1 ve 7.3.1.2 'de sırasıyla, bozucu kanal$_1$ ve kanal$_2$ için elde edilen hata başarım eğrileri görülmektedir.

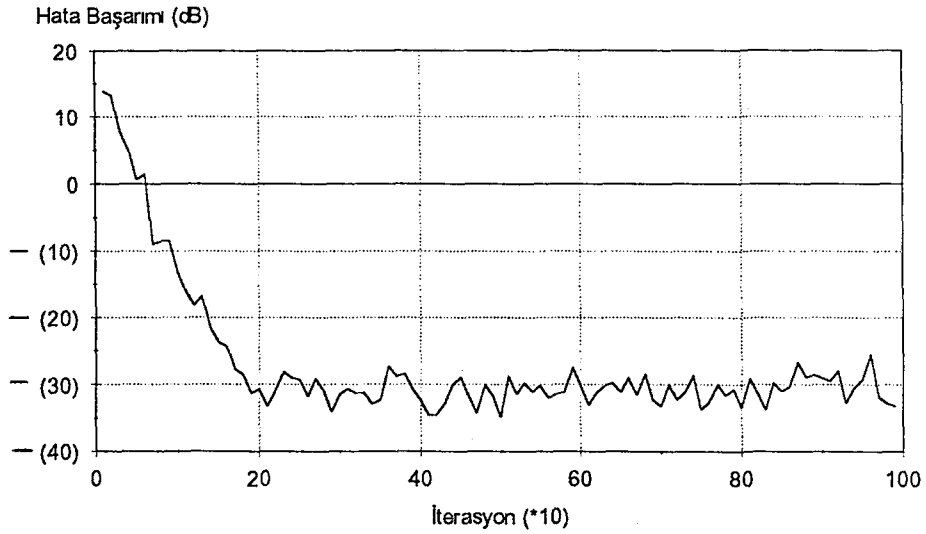
Varyans	: 0.0	Süzgeç Uzunluğu	: 21
Uyarlama Katsayısı	: 0.00085		

Kanal Dosyası : Kanal2.dat



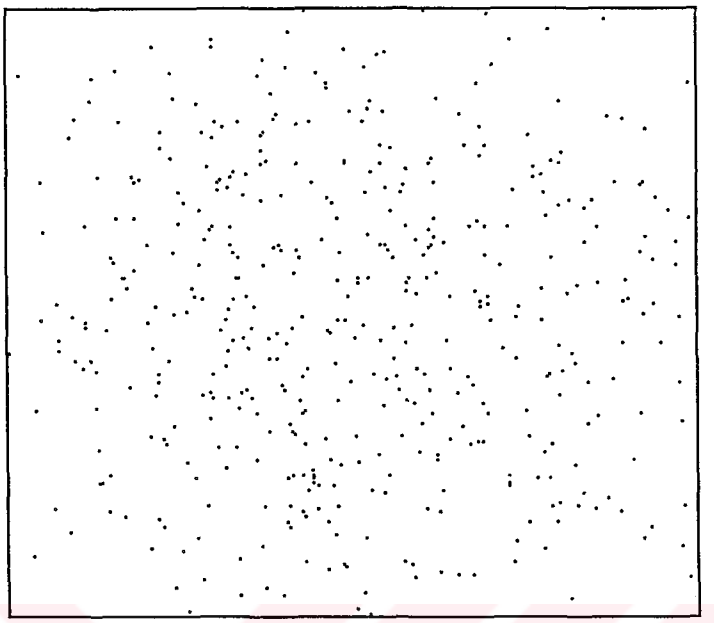
Şekil 7.3.1.1

Kanal Dosyası : Kanal1.dat

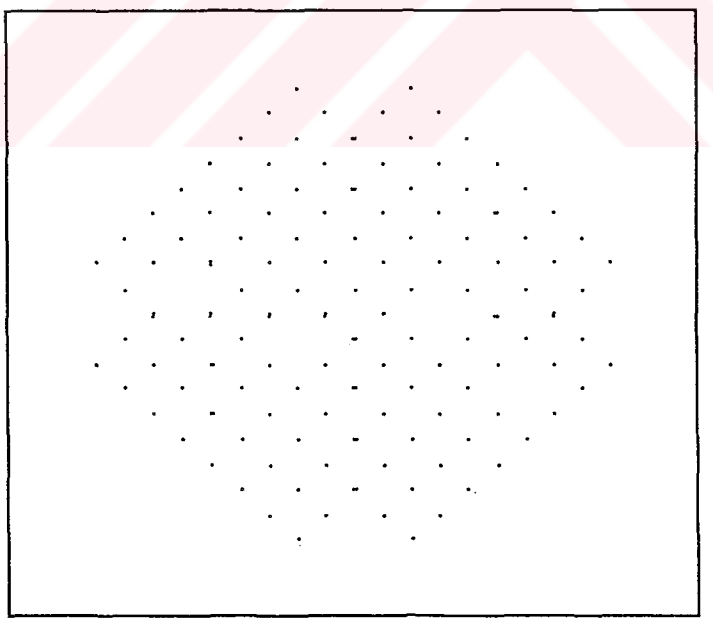


Şekil 7.3.1.2

Şekil 7.3.1.3, 7.3.1.4 'de ise bozucu kanal$_1$ ve kanal$_2$ için , dengeleyici giriş ve çıkışındaki işaret elemanları iki boyutlu düzlemde çizdirilmiştir.

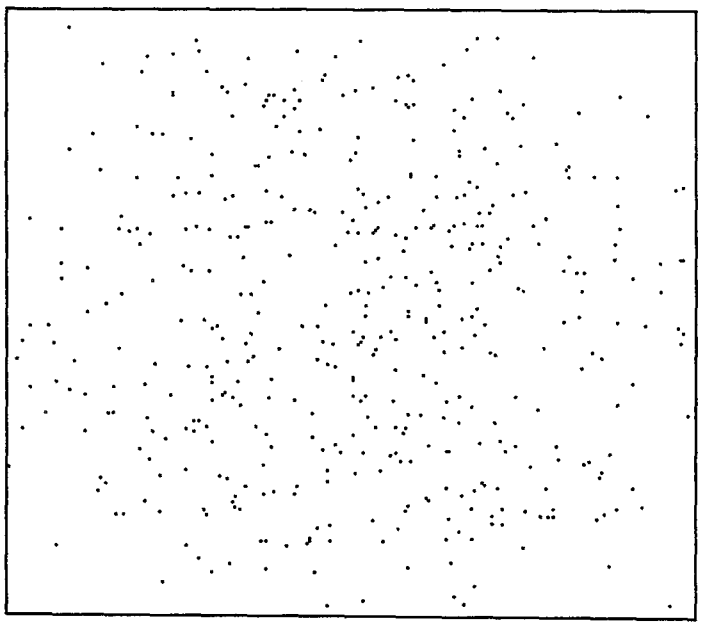


Gürültüsüz Bozucu Kanal$_2$ İçin Dengeleyici Girişi

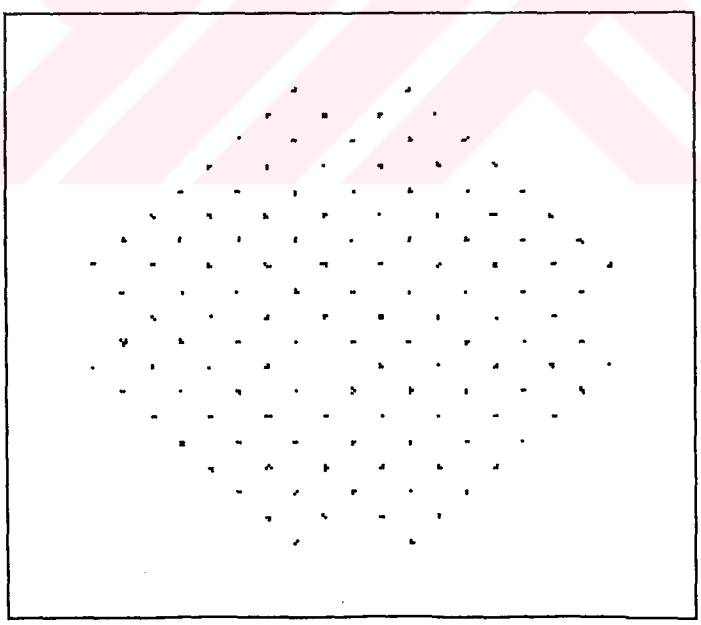


Gürültüsüz Bozucu Kanal$_2$ İçin Dengeleyici Çıkışı

Şekil 7.3.1.3



Gürültüsüz Bozucu Kanal$_1$ İçin Dengeleyici Girişi

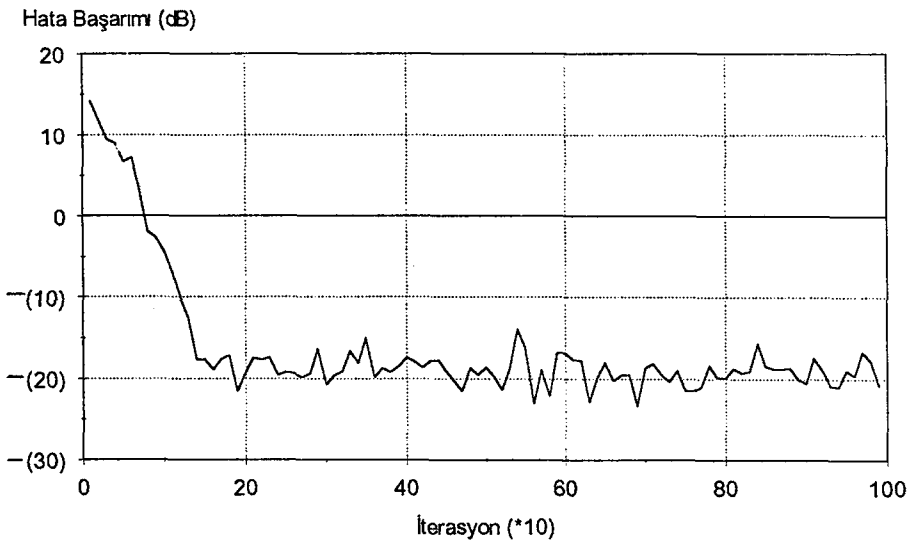


Gürültüsüz Bozucu Kanal$_1$ İçin Dengeleyici Çıkışı

Şekil 7.3.1.4

Çalışmanın bu kısmında, bozucu kanal$_2$ için uyarlama parametreleri sabit tutularak, varyans değerinin değiştirilmesi ile hata başarımının değişimi incelenmiştir.

Kanal Dosyası : Kanal2.dat

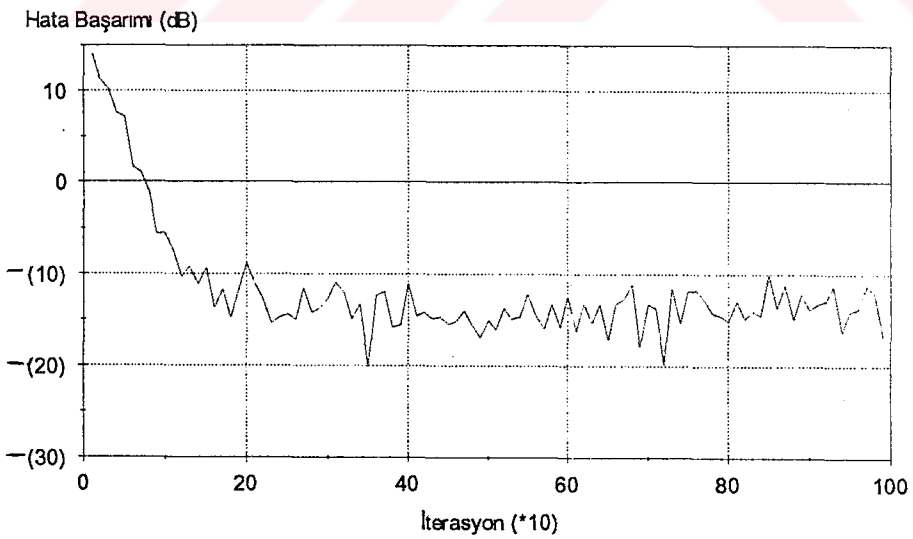


Varyans : 0.01 Süzgeç Uzunluğu : 21

Uyarlama Katsayısı : 0.00085

Şekil 7.3.1.5

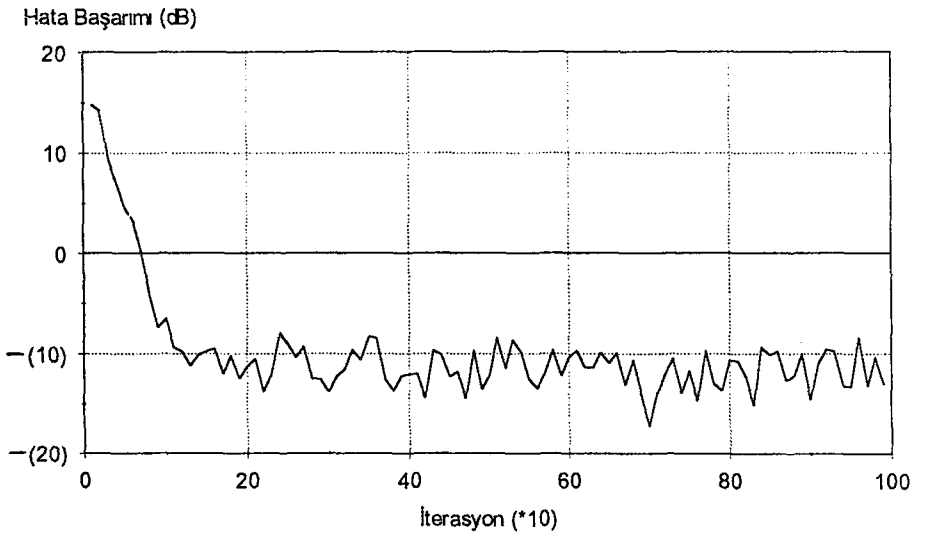
Kanal Dosyası Kanal2.dat



Varyans : 0.03

Şekil 7.3.1.6

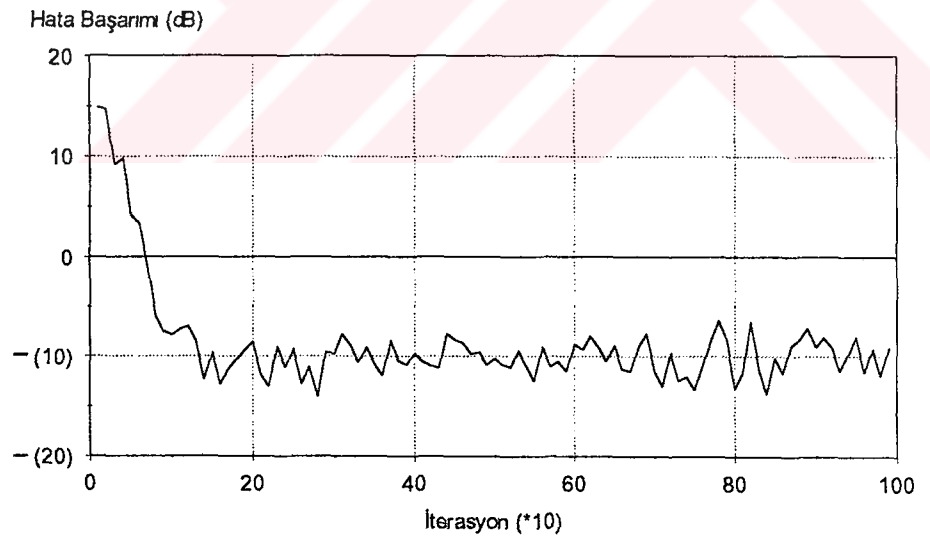
Kanal Dosyası : Kanal2.dat



Varyans : 0.06

Şekil 7.3.1.7

Kanal Dosyası : Kanal2.dat

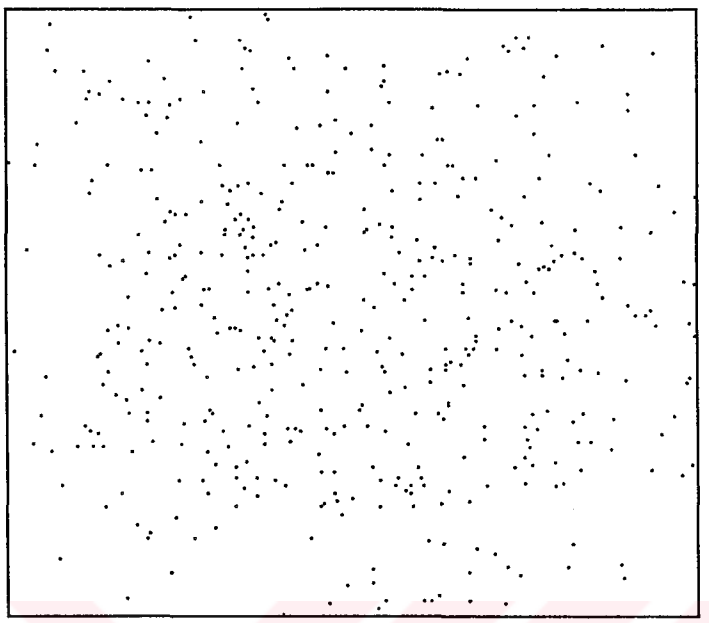


Varyans : 0.08

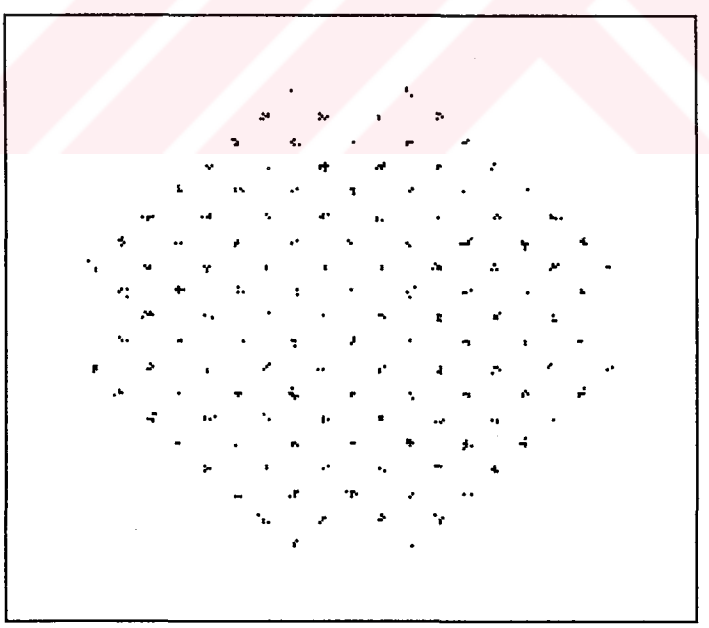
Şekil 7.3.1.8

Şekil 7.3.1.5 'den, 7.3.1.8 'e kadar olan şekillerde, bozucu kanal$_2$ için değişik varyans değerleri altında hata başarım eğrileri verilmiştir.

Şekil 7.3.1.9 ile 7.3.1.12 arasındaki şekillerde, yukarıda verilen hata başarım eğrileri için dengeleyici girişindeki ve çıkışındaki işaret elemanları iki boyutlu düzlemde gösterilmiştir.

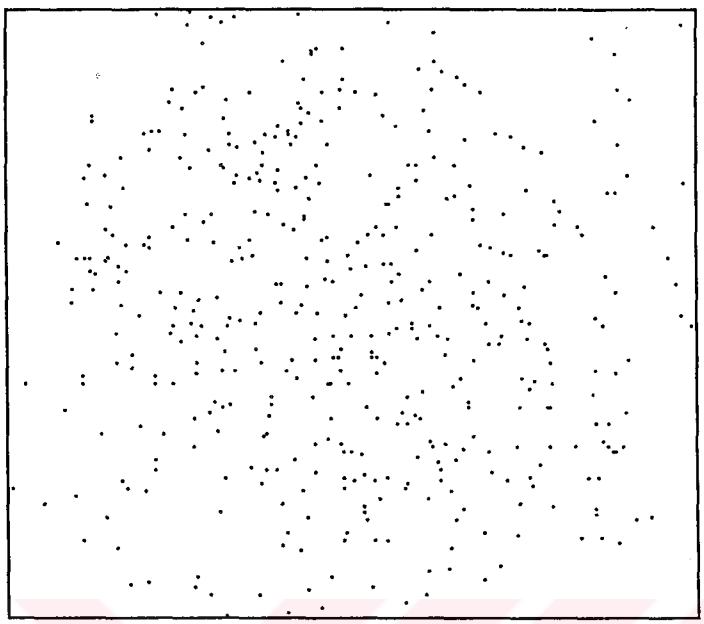


0.01 Varyanslı Bozucu Kanal$_2$ İçin Dengeleyici Girişi

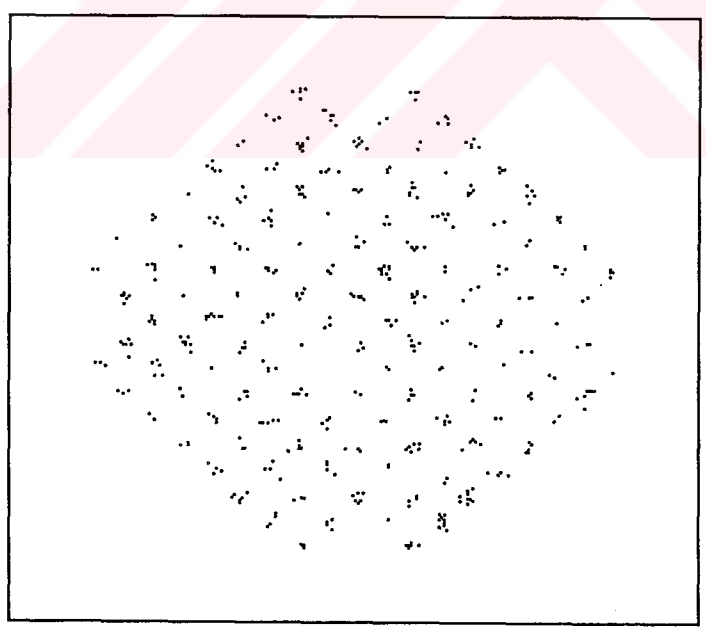


0.01 Varyanslı Bozucu Kanal$_2$ İçin Dengeleyici Çıkışı

Şekil 7.3.1.9

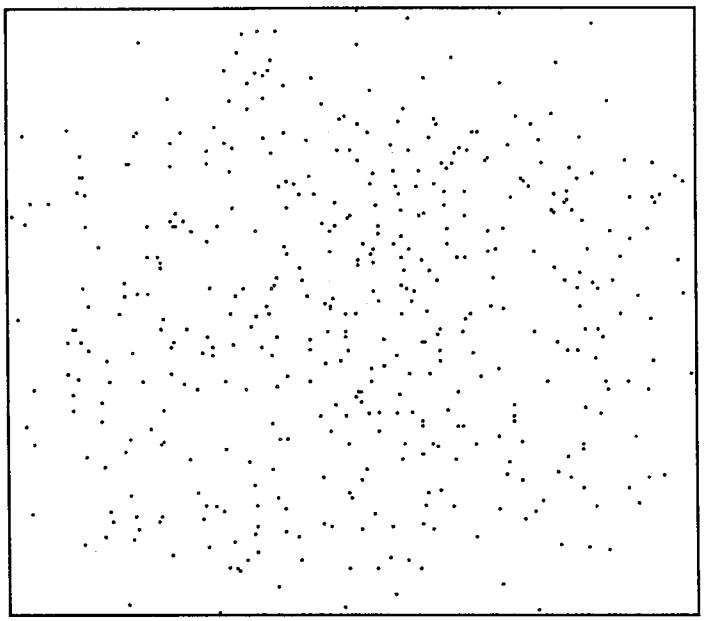


0.03 Varyanslı Bozucu Kanal$_2$ İçin Dengeleyici Girişi

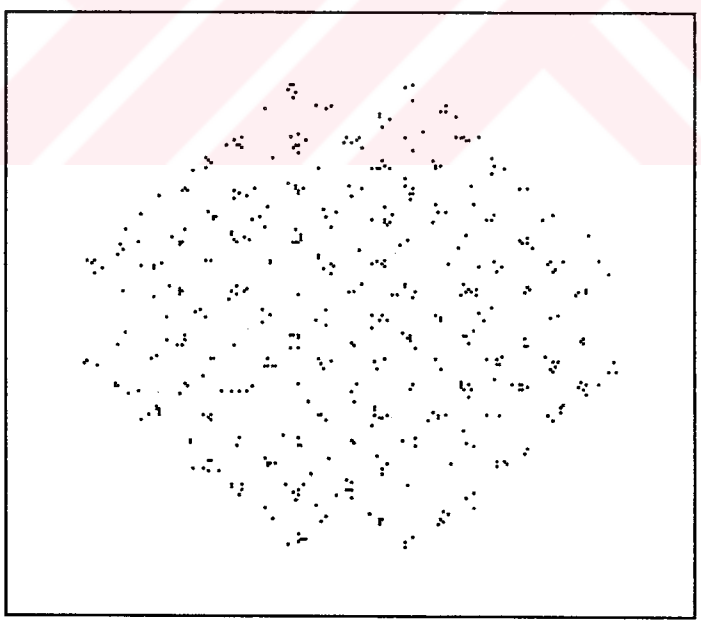


0.03 Varyanslı Bozucu Kanal$_2$ İçin Dengeleyici Çıkışı

Şekil 7.3.1.10

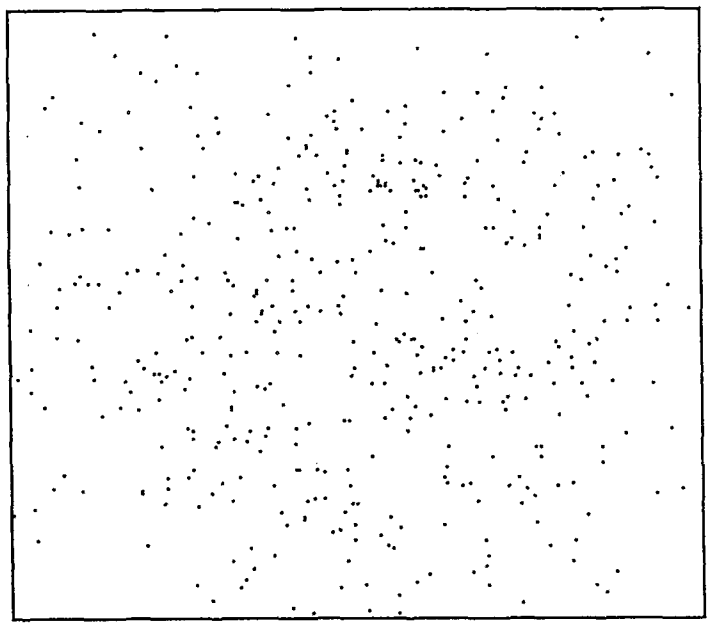


0.06 Varyanslı Bozucu Kanal$_2$ İçin Dengeleyici Girişi

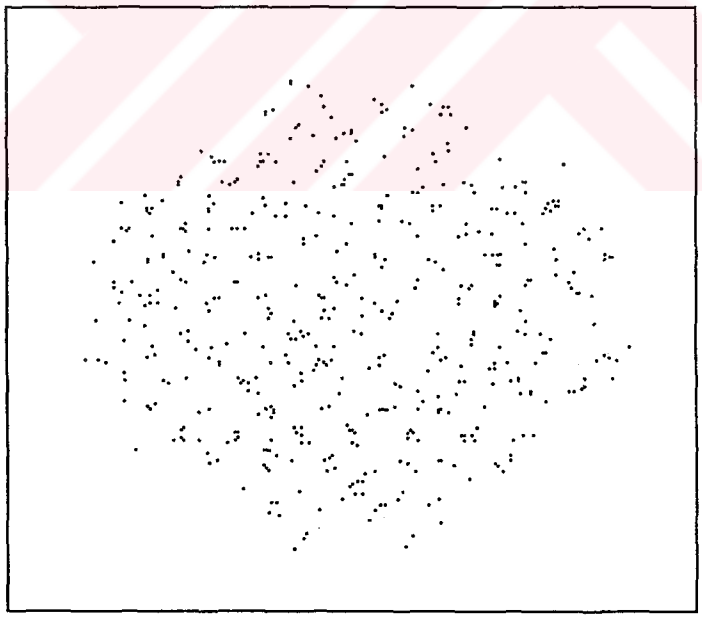


0.06 Varyanslı Bozucu Kanal$_2$ İçin Dengeleyici Çıkışı

Şekil 7.3.1.11



0.08 Varyanslı Bozucu Kanal$_2$ İçin Dengeleyici Girişi

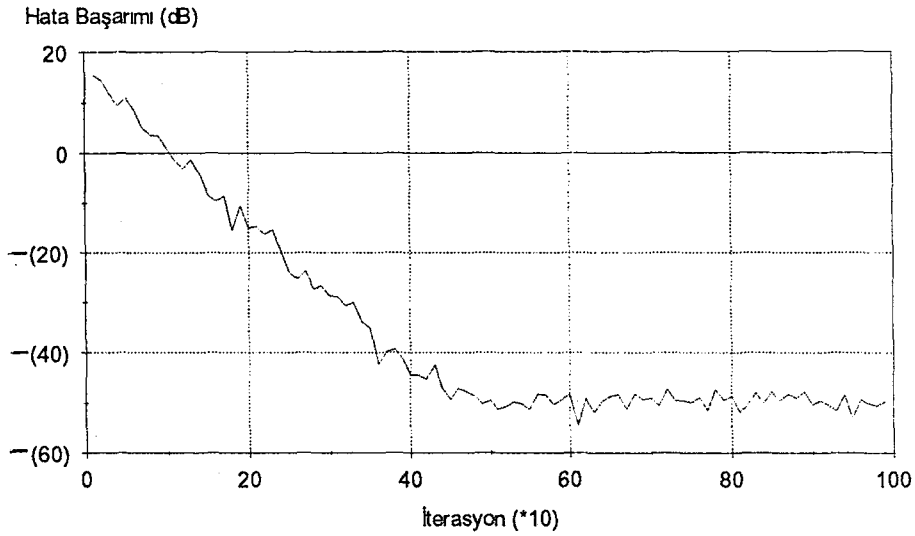


0.08 Varyanslı Bozucu Kanal$_2$ İçin Dengeleyici Çıkışı

Şekil 7.3.1.12

Bu kısımda bozucu kanal$_2$ için, varyans değeri sabit tutularak, uyarlama parametrelerinin değişimi ile dengeleyicinin davranışı ve ortalama karesel hata eğrisindeki değişimler incelenmiştir.

Kanal Dosyası :Kanal2.dat

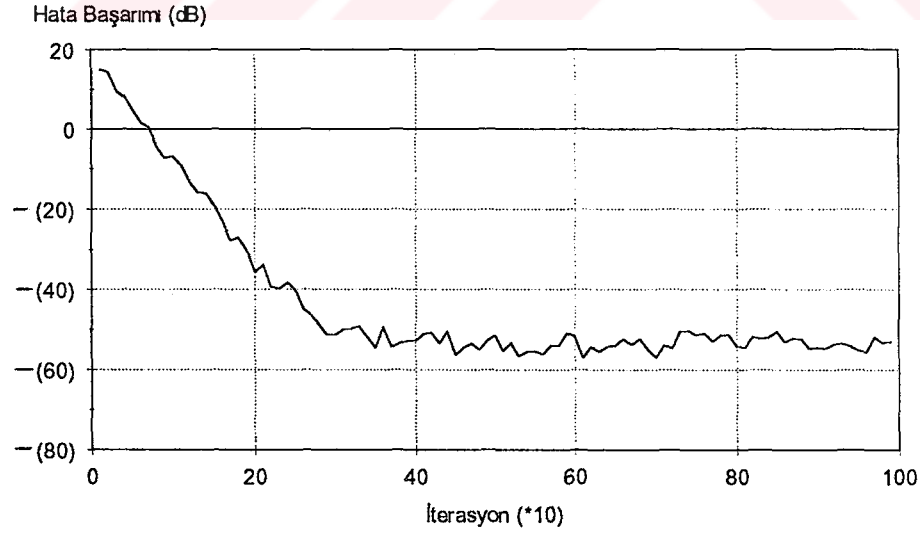


Varyans : 0.0 Süzgeç Uzunluğu : 21

Uyarlama Katsayısı : 0.0003

Şekil 7.3.1.13

Kanal Dosyası : Kanal2.dat

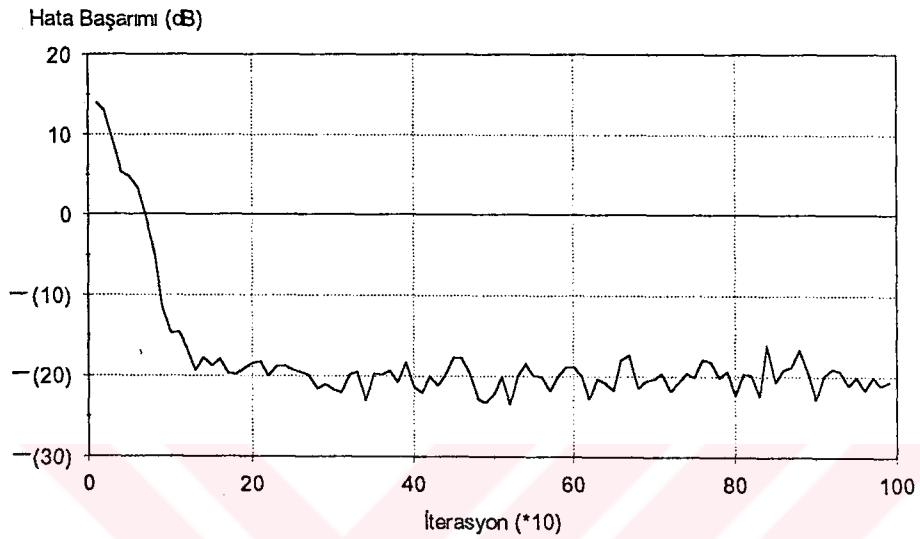


Uyarlama Katsayısı : 0.00090

Şekil 7.3.1.14

Çalışmanın bu bölümünde, bozucu kanal$_2$ için uyarlama katsayısı ve varyans değeri sabit tutularak, süzgeç uzunluğunun değiştirilmesi ile dengeleyicinin davranışındaki ve dolayısıyla ortalama karesel hata eğrisindeki değişimler incelenmiştir.

Kanal Dosyası : Kanal2.dat

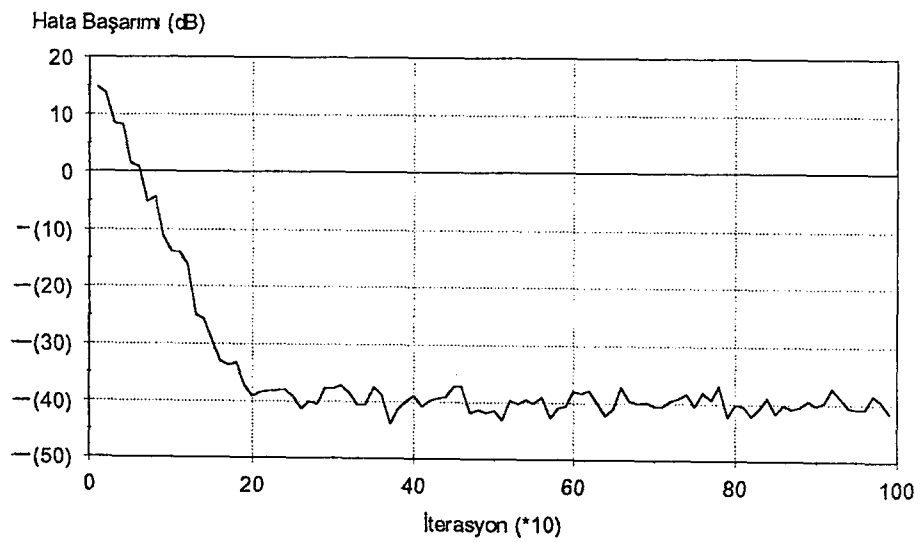


Varyans : 0.0 Süzgeç Uzunluğu : 9

Uyarlama Katsayısı : 0.00070

Şekil 7.3.1.15

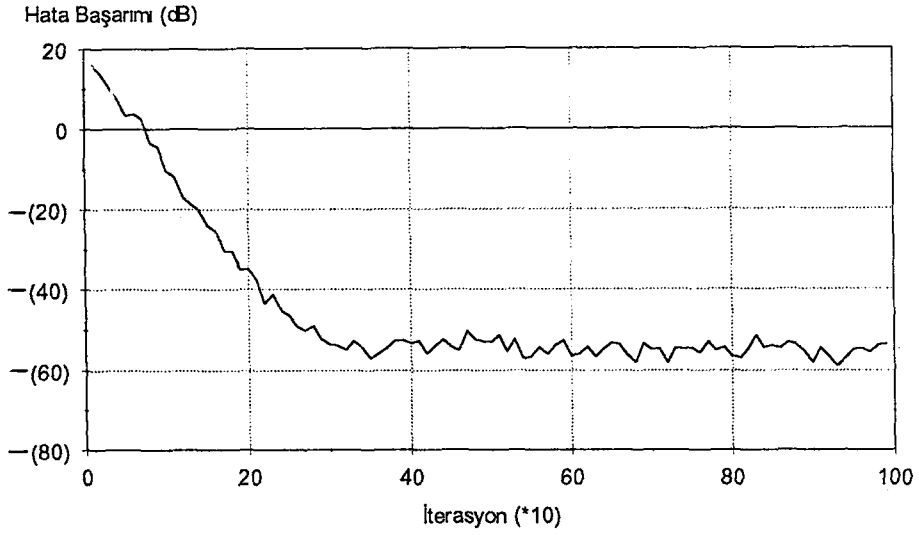
Kanal Dosyası : Kanal2.dat



Süzgeç Uzunluğu : 15

Şekil 7.3.1.16

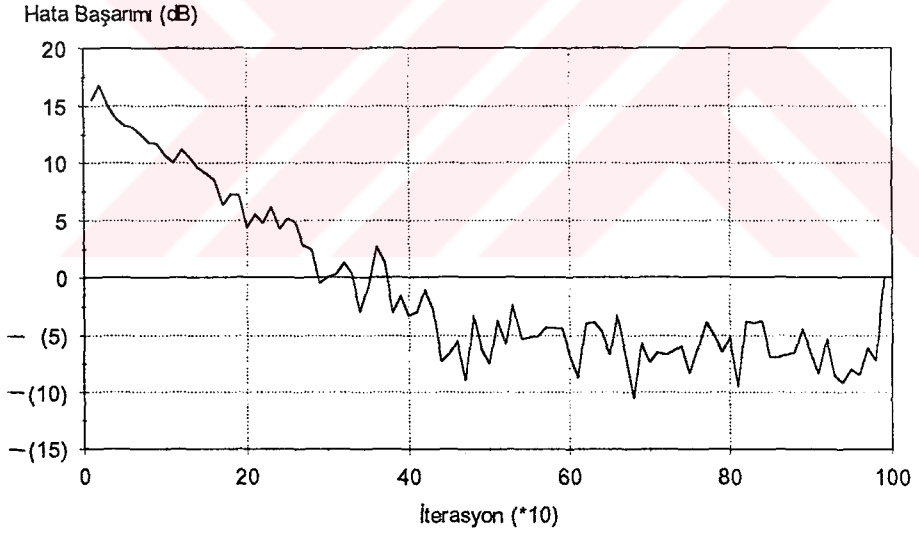
Kanal Dosyası : Kanal2.dat



Suzgeç Uzunluğu : 21

Şekil 7.3.1.17

Kanal Dosyası Kanal2.dat

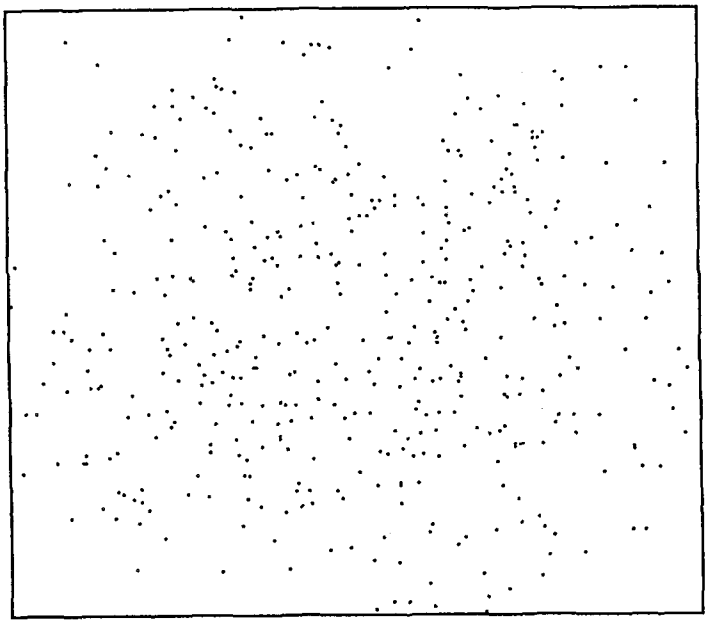


Süzgeç Uzunluğu : 27

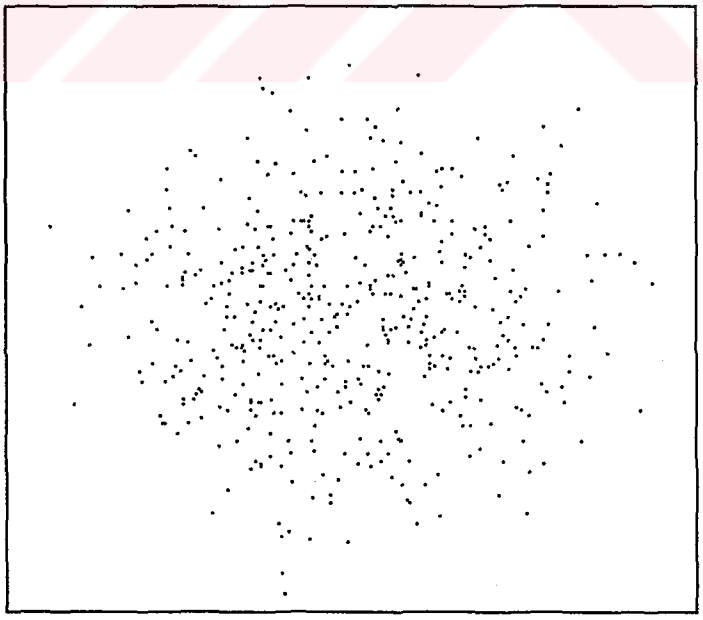
Şekil 7.3.1.18

Süzgeç uzunluğunun çok büyük seçildiği şekil 7.3.1.18 'de görülen durum için, dengeleyici giriş ve çıkışındaki işaret elemanlarının durumu şekil 7.3.1.19 ve 7.3.1.20 'de görülmektedir.

Tablo 7.3.1.1 'de ise belirtilen bozucu kanal, varyans ve uyarlama parametreleri için, doğrusal dengeleyicinin optimum süzgeç katsayıları verilmiştir.



Şekil 7.3.1.19



Şekil 7.3.1.20

OPTİMUM SÜZGEÇ KATSAYILARI

Kanal Dosyası : Kanal2.dat

C_R	C_I
- 0.2810000D - 04	- 0.1513000D - 03
- 0.1694000D - 03	0.3073000D - 03
0.7694000D - 03	- 0.1616000D - 03
- 0.1397400D - 02	- 0.1022200D - 02
0.7946000D - 03	0.3562500D - 02
0.4241400D - 02	- 0.8216700D - 02
- 0.1947730D - 01	0.8792200D - 02
0.4923080D - 01	0.1546830D - 01
- 0.7321050D - 01	- 0.1094485D 00
- 0.5507250D - 01	0.3190776D 00
0.6549363D - 00	- 0.2270971D 00
0.2284620D 00	- 0.8070800D - 02
- 0.1683650D 01	- 0.3153000D - 03
0.6934600D - 02	0.9367700D - 02
0.1511600D - 02	0.7937000D - 03
- 0.2555700D - 02	- 0.4658900D - 02
0.6562000D - 03	0.2045200D - 02
- 0.4400000D - 05	0.8665000D - 03
0.2222000D - 03	- 0.6032000D - 03
- 0.1020000D - 04	- 0.2460000D - 04
- 0.5620000D - 04	0.1660000D - 04

Uyarlama Katsayısı : 0.0007 Süzgeç Uzunluğu : 21

Varyans : 0.0

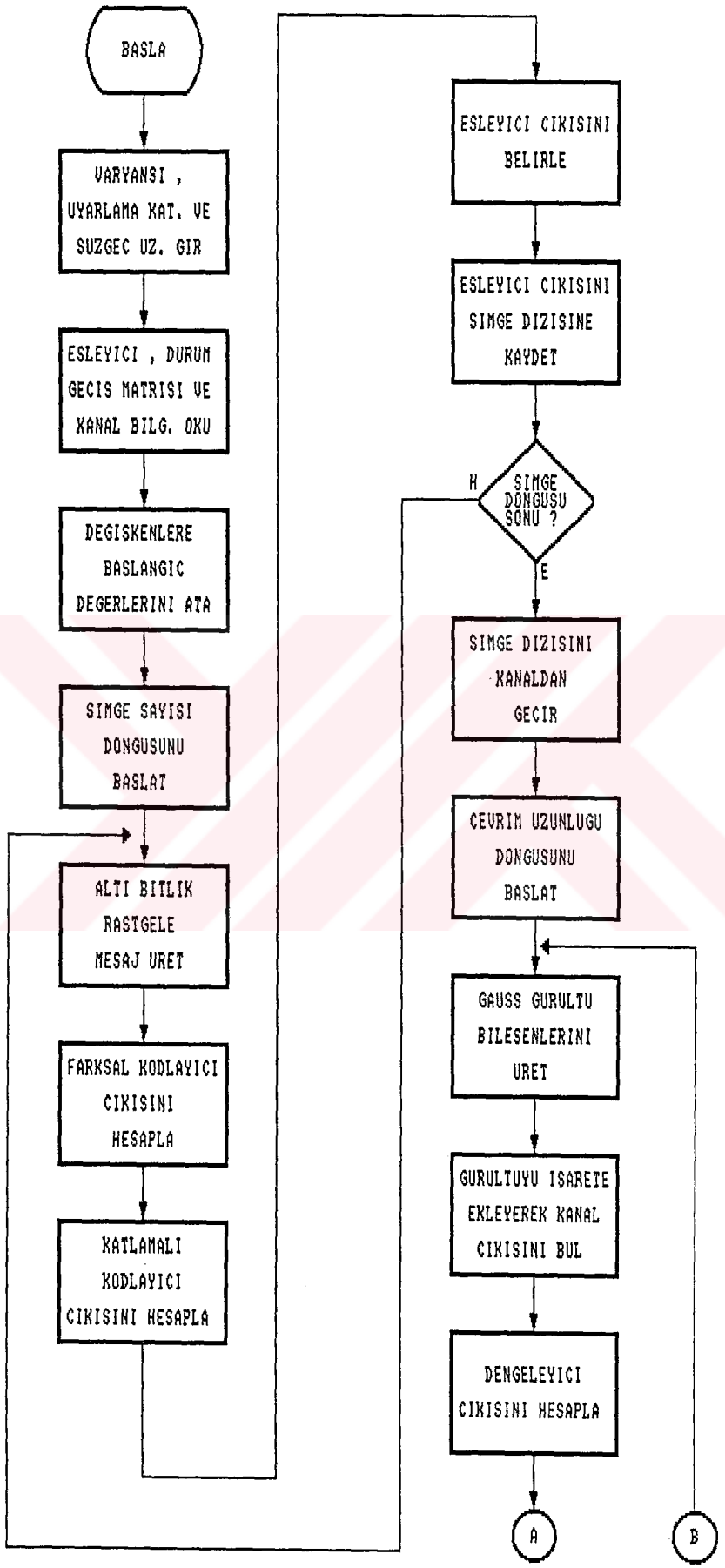
Tablo 7.3.1.1

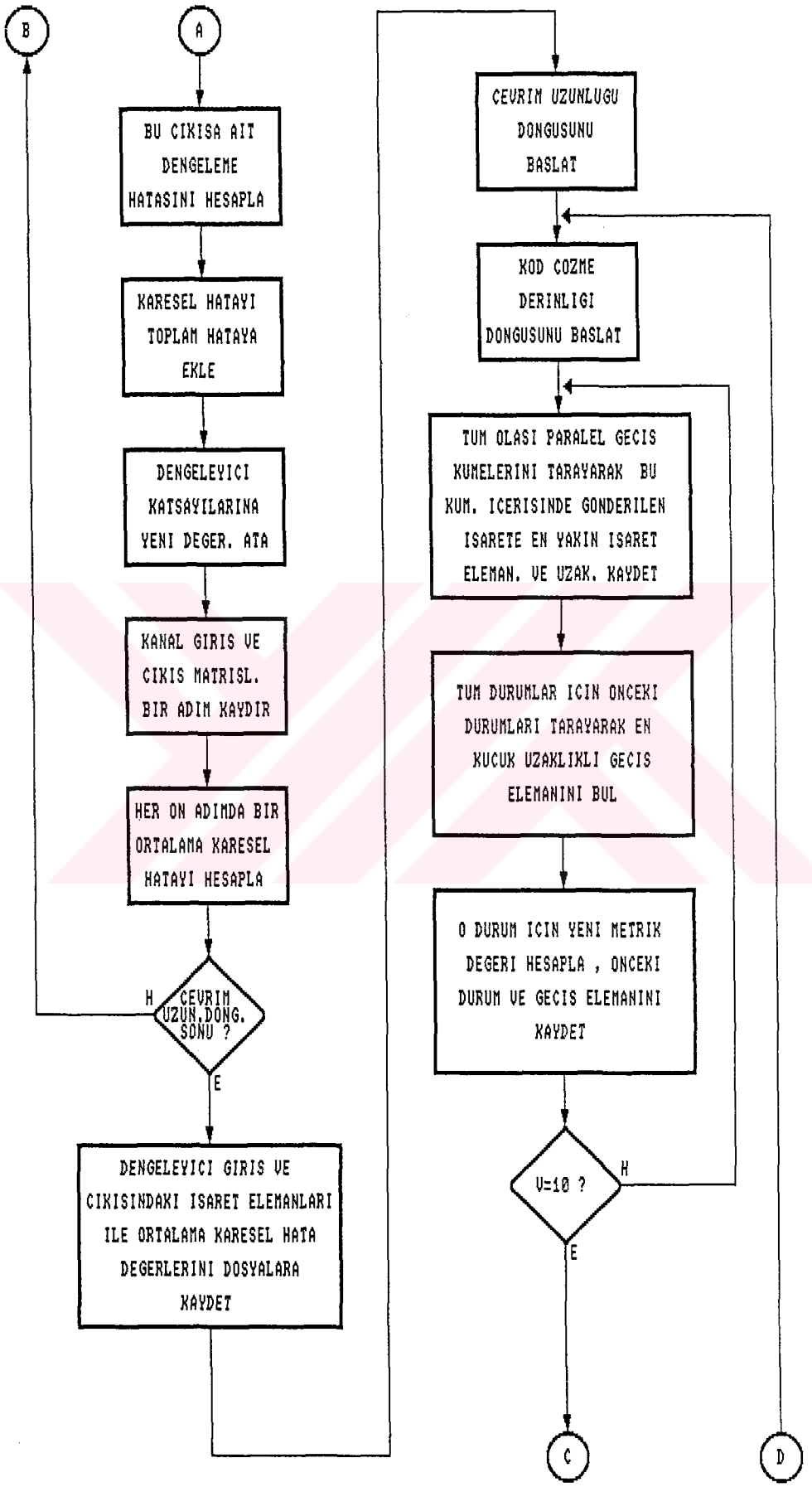
Sim2.c programının çalıştırılması ile her seçilen parametre değeri için optimum süzgeç katsayıları ekrana yazdırılmaktadır.

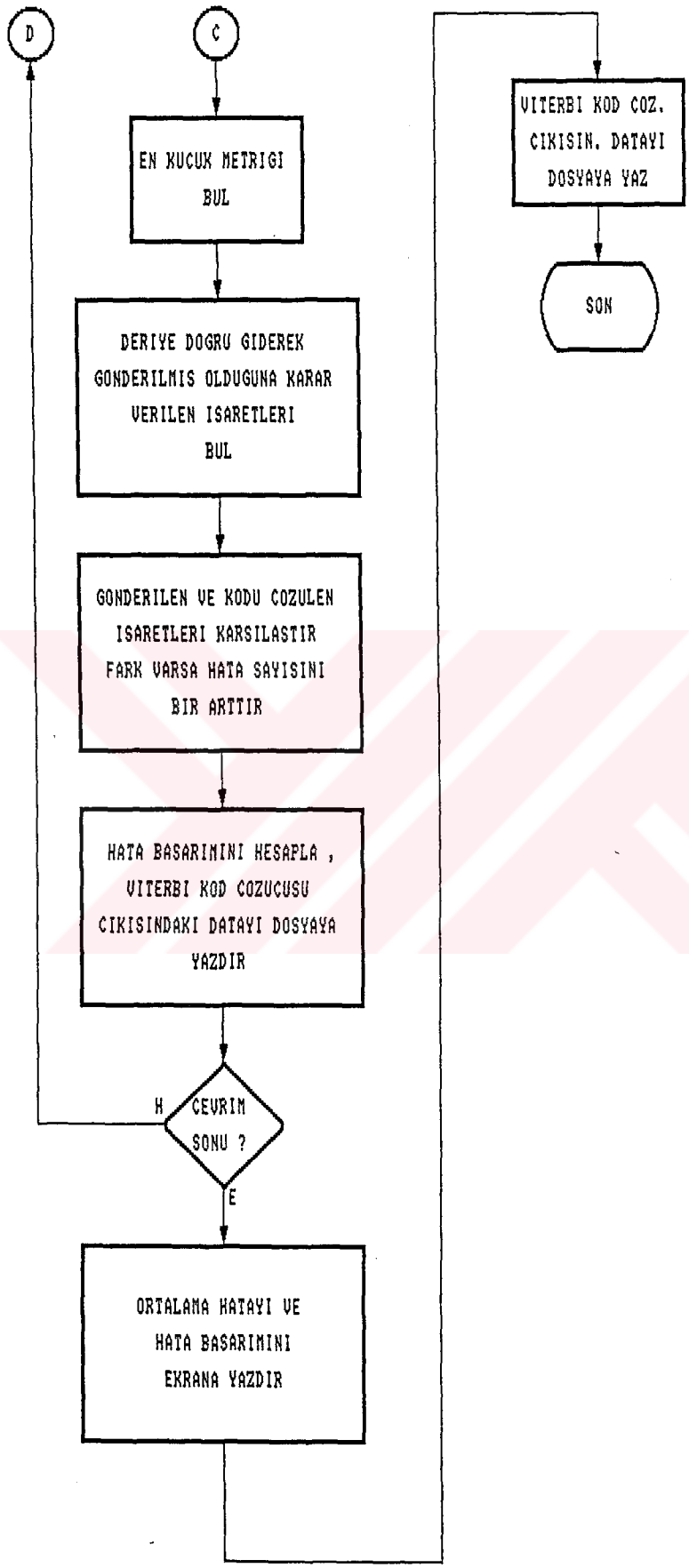
7.3.2 Değişik İşaret Gürültü Oranları İçin Doğrusal Dengeleyici İçeren Sistemin Hata Başarımının İncelenmesi

Bu bölümde bozucu kanal ve doğrusal dengeleyici içeren sisteme, Viterbi Kod Çözücüsü de eklenmiş ve değişik işaret - gürültü oranları için hata başarımının değişimi incelenmiştir. Elde edilen sonuçlara göre sistemin hata başarım eğrileri verilmiştir. Burada adım uzunluğu 50 olarak alınmıştır.

Bu bölüm için hazırlanan "sim3.c" isimli programın tam listesi ekler bölümünde verilmiştir. Programın akış diyagramı bir sonraki sayfadan itibaren gösterilmektedir.

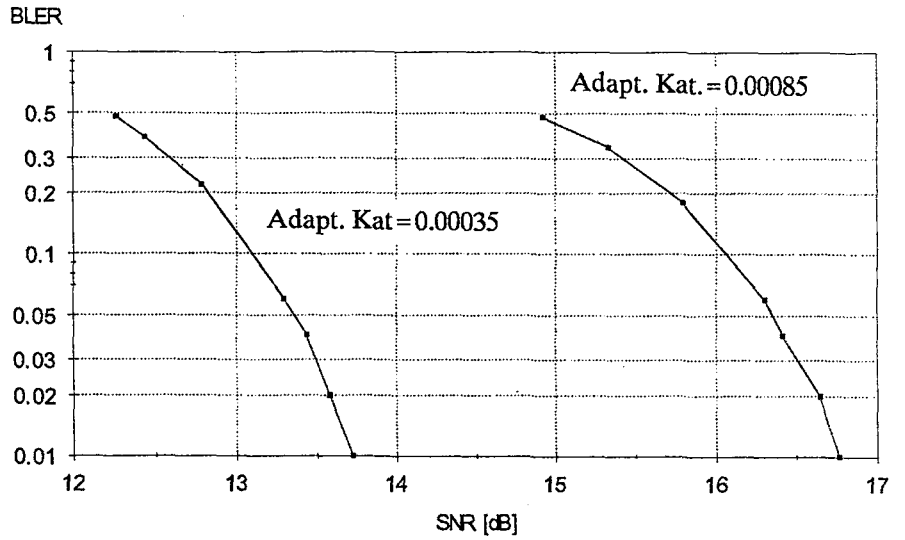

BASLA
VARYANSI , UYARLAMA KAT. VE SUZGEC UZ. GIR
ESLEYICI , DURUM GECIS MATRISI VE KANAL BILG. OKU
DEGISKENLERE BASLANGIC DEGERLERINI ATA
SIMGE SAYISI DONGUSUNU BASLAT
ALTI BITLIK RASTGELE MESAJ URET
FARKSAL KODLAYICI CIKISINI HESAPLA
KATLAMALI KODLAYICI CIKISINI HESAPLA
ESLEYICI CIKISINI BELIRLE
ESLEYICI CIKISINI SIMGE DIZISINE KAYDET
H
SIMGE DONGUSU SONU ?
E
SIMGE DIZISINI KANALDAN GECIR
CEVRIM UZUNLUGU DONGUSUNU BASLAT
GAUSS GURULTU BILESENLERINI URET
GURULTUYU ISARETE EKLEYEREK KANAL CIKISINI BUL
DENGELEYICI CIKISINI HESAPLA
A
B


B
A
BU CIKISA AIT DENGELEME HATASINI HESAPLA
KARESEL HATAYI TOPLAM HATAYA EKLE
DENGELEYICI KATSAYILARINA YENI DEGER. ATA
KANAL GIRIS VE CIKIS MATRISL. BIR ADIM KAYDIR
HER ON ADIMDA BIR ORTALAMA KARESEL HATAYI HESAPLA
H
CEVRIM UZUN.DONG. SONU ?
E
DENGELEYICI GIRIS VE CIKISINDAKI ISARET ELEMANLARI ILE ORTALAMA KARESEL HATA DEGERLERINI DOSYALARA KAYDET
CEVRIM UZUNLUGU DONGUSUNU BASLAT
KOD COZME DERINLIGI DONGUSUNU BASLAT
TUM OLASI PARALEL GECIS KUMELERINI TARAYARAK BU KUM. ICERISINDE GONDERILEN ISARETE EN YAKIN ISARET ELEMAN. VE UZAK. KAYDET
TUM DURUMLAR ICIN ONCEKI DURUMLARI TARAYARAK EN KUCUK UZAKLIKLI GECIS ELEMANINI BUL
O DURUM ICIN YENI METRIK DEGERI HESAPLA , ONCEKI DURUM VE GECIS ELEMANINI KAYDET
V=10 ?
H
E
C
D


D
C
EN KUCUK METRIGI BUL
DERIYE DOGRU GIDEREK GONDERILMIS OLDUGUNA KARAR VERILEN ISARETLERI BUL
GONDERILEN VE KODU COZULEN ISARETLERI KARSILASTIR FARK VARSA HATA SAYISINI BIR ARTTIR
HATA BASARIMINI HESAPLA , VITERBI KOD COZUCUSU CIKISINDAKI DATAYI DOSYAYA YAZDIR
H
CEVRIM SONU ?
E
ORTALAMA HATAYI VE HATA BASARIMINI EKRANA YAZDIR
VITERBI KOD COZ. CIKISIN. DATAYI DOSYAYA YAZ
SON

Sistemin hata başarımı, bozucu kanal$_2$ ve iki ayrı uyarlama katsayısı değeri için incelenmiştir. Şekil 7.3.2.1 'de, iki ayrı uyarlama katsayısı için elde edilen hata başarım eğrileri görülmektedir.

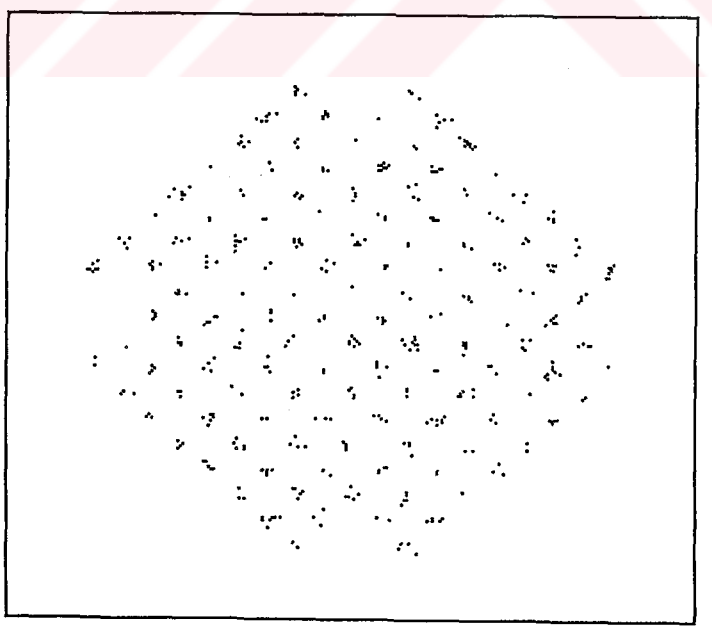
Kanal Dosyası : Kanal2.dat



Süzgeç Uzunluğu : 21

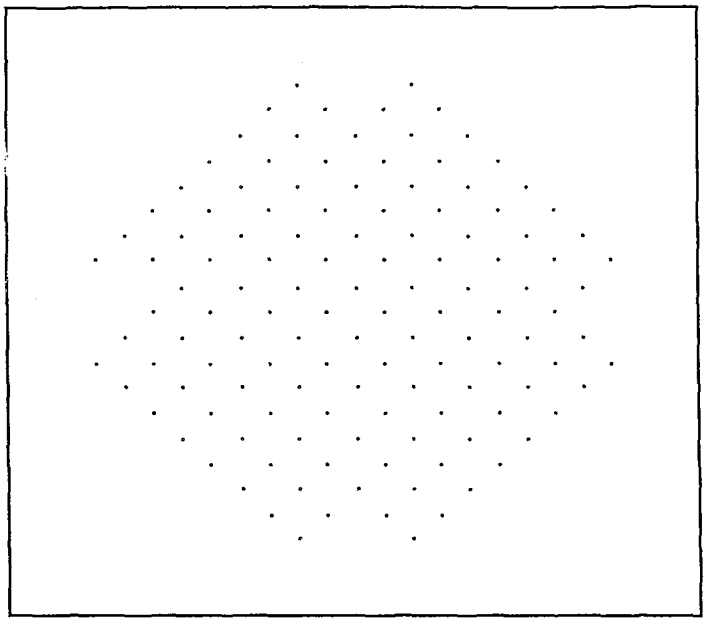
Şekil 7.3.2.1

Çalışmanın bu bölümünde, doğrusal uyarlamalı dengeleyici ve Viterbi Kod Çözücüsü içeren sistemde, belirli bir varyans değeri ve uyarlama parametreleri için, dengeleyici ve çıkışı ile kod çözücü çıkışındaki işaret elemanlarının durumu iki boyutlu düzlemde gösterilmiştir.



0.04 Varyanslı Bozucu Kanal$_2$ İçin Dengeleyici Çıkışı

Şekil 7.3.2.2



0.04 Varyanslı Bozucu Kanal$_2$ İçin Viterbi Kod Çözücüsü Çıkışı

Şekil 7.3.2.3

Adaptasyon Katsayısı	: 0.0007	Varyans	: 0.04
Süzgeç Uzunluğu	: 21		

Şekil 7.3.2.2 ve 7.3.2.3 'de, işaret elemanlarının işaret uzayı içerisindeki durumları gösterilmiştir. Bu çalışma için alınan parametre değerleri yukarıda verilmiştir.

7.3.3 Doğrusal Dengeleyici İçeren Sistemde, İşaret Elemanlarının Belirli Veya Rastgele Açılarla Dönmesinin Hata Başarımınına Etkileri

Bu kısımda, bozucu kanal ve doğrusal dengeleyici içeren benzetim modelinde, işaret uzayı içerisindeki işaret elemanlarının 3, 5, 8 derecelik sabit veya rastgele açılarla dönmelerinin, değişik işaret gürültü oranları için hata başarımına etkileri incelenmiştir. Rastgele açı dönmeleri, 0.0-8.0 derece arasında değerler alabilen düzgün bir dağılıma sahiptirler.

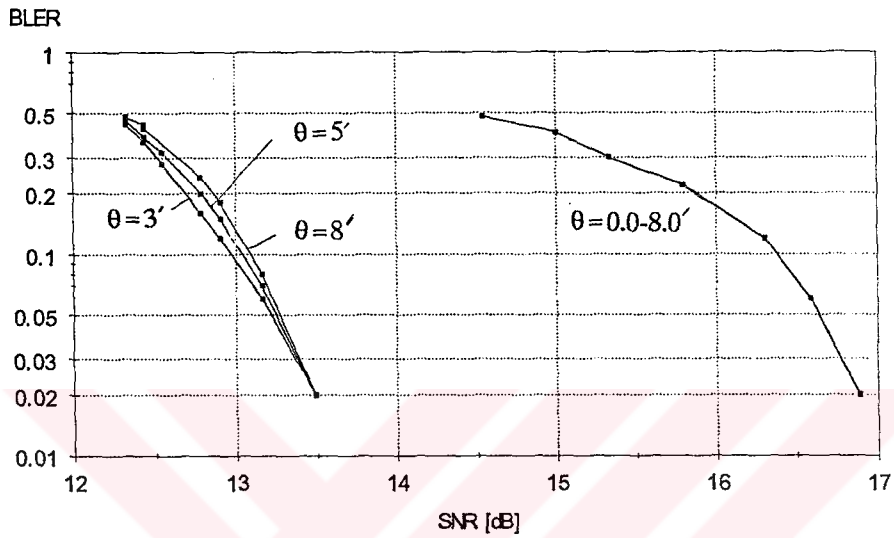
Vericiden gönderilen işaret elemanları, daha önceki alt bölümde elde edilen 7.2.1.1 ifadesine göre θ açısı kadar döndürülmekte ve daha sonra bozucu kanala gönderilmektedir. Bozucu kanaldan çıkan mesaj dizilerine toplamsal Gauss gürültüsünün gerçel ve sanal bileşenleri de eklenmektedir.

Bu inceleme için kullanılan programın tam listesi ekler bölümünde "sim3a.c" adı altında verilmektedir. Kullanılan programın akış diyagramı "sim3.c" adı ile verilen programın akış diyagramı ile

aynıdır. Tek fark olarak, işaret elemanları sabit veya rastgele açılarla işaret uzayı içerisinde döndürülmektedir.

Şekil 7.3.3.1 'de, θ dönme açısının 3, 5, 8 derece olarak alarak alınması ve 0.0 derece ile 8.0 derece arasında rastgele olarak değişmesi sonucu elde edilen hata başarım eğrileri verilmiştir.

Kanal Dosyası : Kanal2.dat



Süzgeç Uzunluğu : 21

Şekil 7.3.3.1

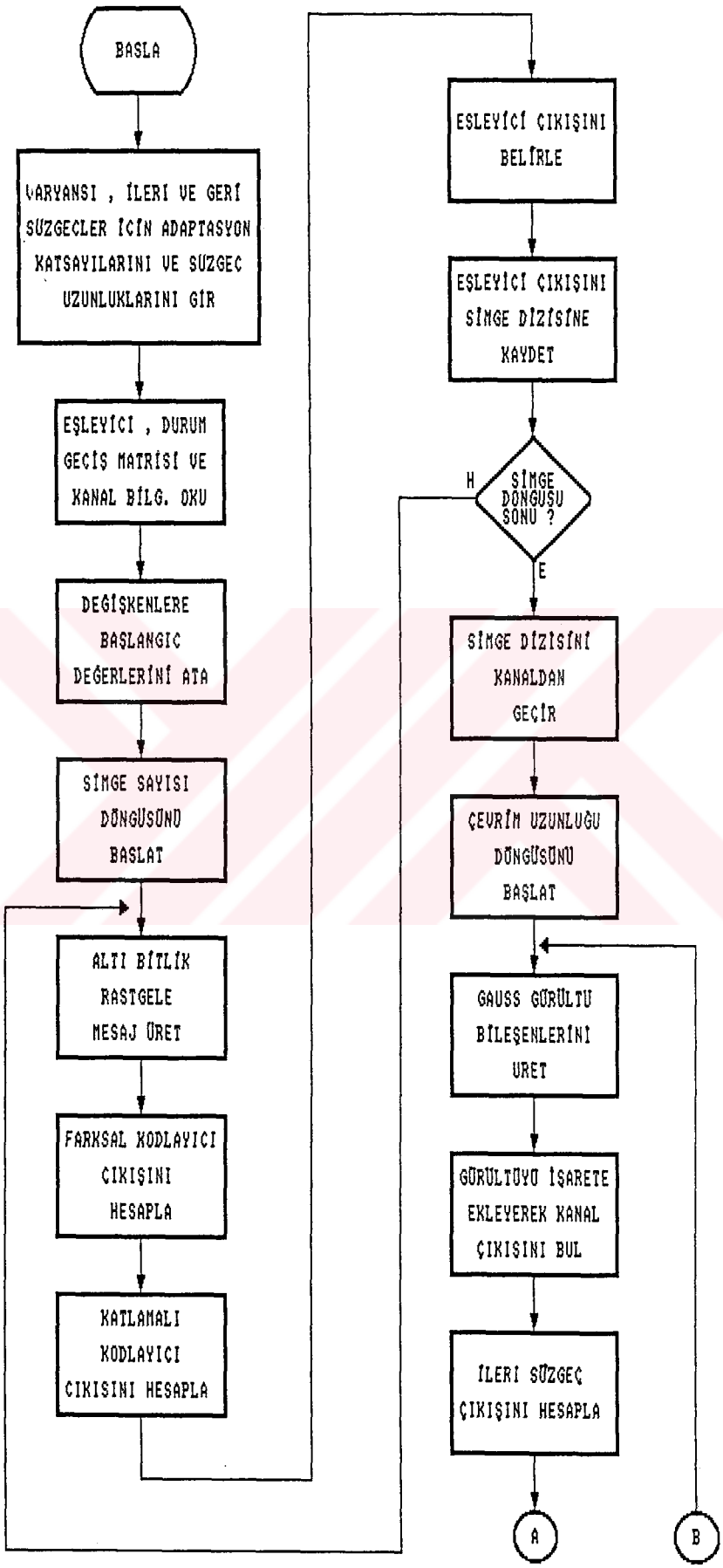
Görüldüğü gibi, θ dönme açısının rastgele biçimde seçildiği durum dışındaki hata başarım eğrilerinde birbirine yakın sonuçlar elde edilmiştir. Bu da doğrusal uyarlamalı dengeleyicinin, işaret elemanlarının işaret uzayı içerisinde sabit açılarla dönmeleri sonucu ortaya çıkan etkileri giderebildiğini göstermektedir. Öte yandan,dönme açısının düzgün dağılımlı rastgele değerler alması durumunda doğrusal uyarlamalı dengeleyici aynı hata başarım değerlerine daha yüksek işaret-gürültü oranı değerlerinde ulaşabilmektedir.

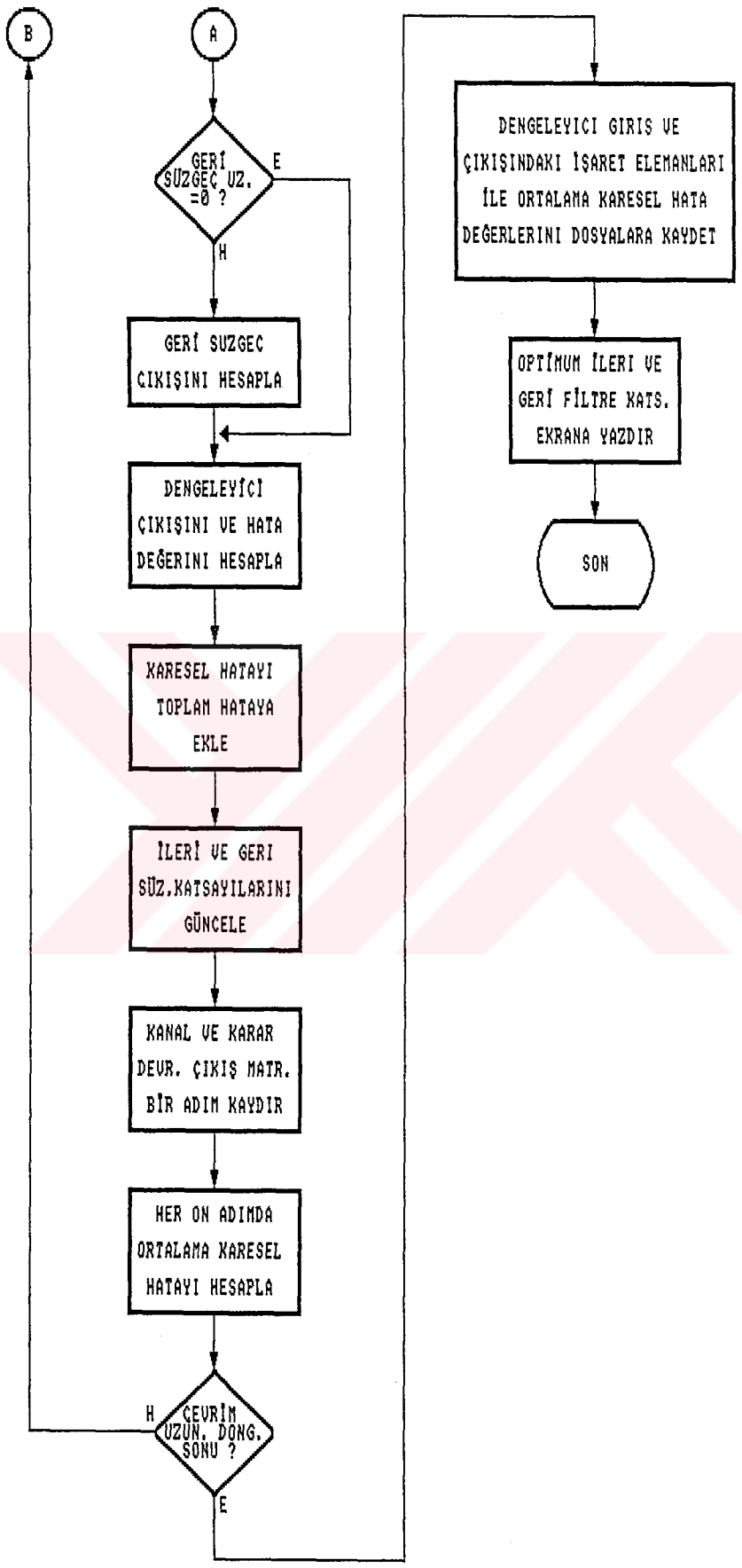
7.4 Doğrusal Olmayan Karar Geri Beslemeli Dengeleyici İle Kanalın Bozucu Etkisinin Giderilmesi

Bu alt bölümde, bozucu kanal içeren sistemde kanal çıkışına doğrusal olmayan karar geri beslemeli bir dengeleyici eklenerek kanalın bozucu etkisinin giderilmesi olayı incelenmiştir.

Sistemin benzetimini gerçekleştiren program, bozucu $kanal_2$ ve belirli varyans değeri için çalıştırılarak dengeleyicinin ortalama karesel hatasının iterasyon sayısı ile değişimi incelenmiştir.

Burada kullanılan "sim4.c" isimli programın tam listesi ekler bölümünde verilmiştir. Programın akış diyagramı bir sonraki sayfadan itibaren verilmektedir.

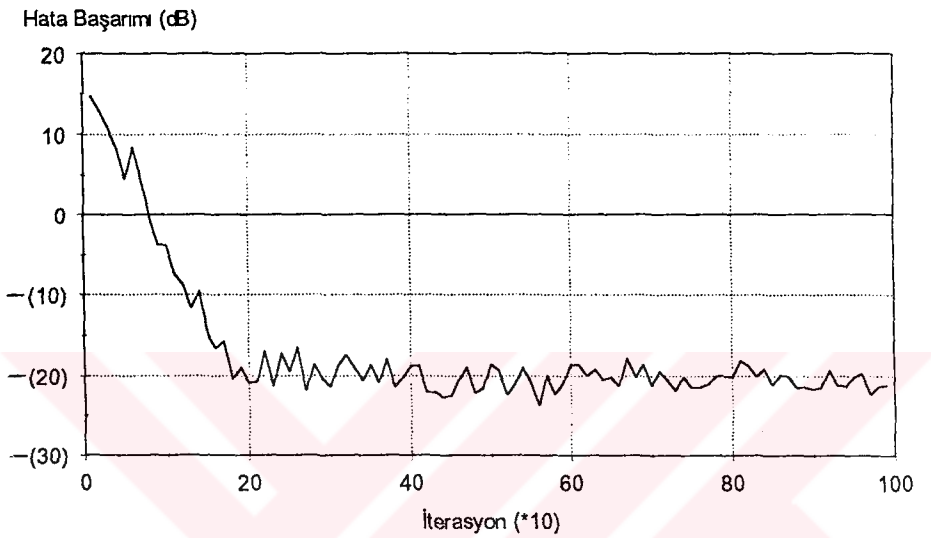

BASLA
VARYANSI , İLERİ VE GERİ SÜZGECLER İCİN ADAPTASYON KATSAYILARINI VE SÜZGEC UZUNLUKLARINI GİR
EŞLEYİCİ , DURUM GECİŞ MATRİSİ VE KANAL BİLG. OKU
DEĞİŞKENLERE BAŞLANGIC DEĞERLERİNİ ATA
SİMGE SAYISI DÖNGÜSÜNÜ BASLAT
ALTI BİTLİK RASTGELE MESAJ ÜRET
FARKSAL KODLAYICI CIKIŞINI HESAPLA
KATLAMALI KODLAYICI CIKISINI HESAPLA
EŞLEYİCİ CIKIŞINI BELİRLE
EŞLEYİCİ CIKIŞINI SİMGE DİZİSİNE KAYDET
H
SİMGE DÖNGÜSÜ SONU ?
E
SİMGE DİZİSİNİ KANALDAN GEÇİR
ÇEVRİM UZUNLUĞU DÖNGÜSÜNÜ BAŞLAT
GAUSS GÜRÜLTÜ BİLEŞENLERİNİ URET
GÜRÜLTÜYÜ İŞARETE EKLEYEREK KANAL ÇIKISINI BUL
İLERİ SÜZGEÇ ÇIKIŞINI HESAPLA
A
B


B
A
GERİ SÜZGEÇ UZ. =0 ?
E
H
GERİ SUZGEC ÇIKIŞINI HESAPLA
DENGELEYİCİ ÇIKIŞINI VE HATA DEĞERINI HESAPLA
KARESEL HATAYI TOPLAM HATAYA EKLE
İLERİ VE GERI SÜZ.KATSAYILARINI GÜNCELE
KANAL VE KARAR DEVR. ÇIKIŞ HATR. BİR ADIM KAYDIR
HER ON ADIMDA ORTALAMA KARESEL HATAYI HESAPLA
H
ÇEVRİM UZUN. DONG. SONU ?
E
DENGELEYICI GIRIŞ VE ÇIKIŞINDAKI İŞARET ELEMANLARI İLE ORTALAMA KARESEL HATA DEĞERLERINI DOSYALARA KAYDET
OPTİMUM İLERI VE GERİ FİLTRE KATS. EKRANA YAZDIR
SON

Şekil 7.4.1 'de bozucu kanal$_2$ için, karar geri beslemeli dengeleyici çıkışındaki ortalama karesel hatanın iterasyon sayısı ile değişimi verilmiştir. Burada ileri yol süzgeç uzunluğu "su", geri besleme süzgeç uzunluğu "su1" ile belirtilmiştir. Ayrıca ileri yol süzgeci uyarlama katsayısı "α" ve geri besleme süzgeci uyarlama katsayısı "β" ile gösterilmiştir.

Varyans	: 0.01	su	: 9	su1	: 11
α	: 0.0006	β	: 0.000055		

Kanal Dosyası : Kanal2.dat



Şekil 7.4.1

BÖLÜM 8. SONUÇ

Hızla artan veri iletişim istekleri nedeniyle, gerek özel hatlar (leased-lines), gerekse anahtarlamalı şebekeler üzerinde bu işlemin daha hızlı ve güvenilir bir şekilde gerçekleştirilmesi gerekmektedir. Birçok uzun mesafe iletim devresi kötü denebilecek derecede gürültü barındırmaktadır. Klasik QAM modemlerinin iletim bozukluklarına karşı olan duyarlılığı, gürültülü devreler üzerinde 9.6 kbs ve daha yüksek veri hızlarında çalışmalarını olanaksız kılmaktadır. Kafes kodlamalı QAM ile hat üzerindeki gürültüye karşı duyarlılık önemli ölçüde azaltılabilmektedir. İşaret kümesi elemanlarının sayısının arttırılması ile alıcı tarafta kod çözme sırasında oluşabilecek hataları azaltmak olasıdır.

Band sınırlı iletim kanallarının neden olduğu zaman saçılması sonucu ortaya çıkan, simgelerarası girişim probleminin üstesinden uyarlamalı dengeleyiciler yardımıyla gelinebilmekte, bu kanallar üzerinde daha yüksek hızlarda veri iletimi sağlanabilmektedir.

14.4 kbps hızında çalışan V.33 modemi kafes kodlamalı dik genlik modülasyonunu kullanmaktadır. Bu çalışmada 14.4 kbps hızındaki modem incelenmiş ve Gauss gürültüsüne sahip kanallarda değişik varyans değerleri için hata başarım eğrileri elde edilmiştir. Ayrıca, işaret uzayı içerisindeki işaret elemanlarının faz kaymaları ve gürültü nedeniyle iletim sırasında, belirli veya rastgele açılar ile dönmelerinin sistemin hata başarımına etkileri incelenmiştir.

Benzetim modelleri kullanılarak yapılan çalışmalarda, dengeleyiciyi belirleyen özelliklerden biri olan süzgeç uzunluklarının gerçekte kanala bağlı olmakla birlikte en az 3 olması gerektiği, ayrıca çok fazla seçildiğinde yakınsamanın sağlanamadığı görülmüştür. Dengeleyicinin yakınsamasındaki diğer bir önemli parametre de uyarlama katsayılarıdır. Bu katsayılar genel olarak süzgeç uzunlukları ile ters orantılı olarak seçilirler. Uygun süzgeç uzunlukları ve uyarlama katsayıları ile genlik ve faz değerleri dengeleyici çıkışında bir nokta halinde elde edilebilir.

Kanalın sadece toplamsal Gauss gürültüsü ile bozucu etkiye sahip olduğu benzetimde, 200 adım sayısı için ve 0.200 varyans değeri için hatalı blokların gönderilen tüm bloklara oranı 0.01 olarak bulunmuştur. Daha sonra bu ideal durumda, işaret uzayı içerisindeki işaret elemanlarının 3, 5, 6 derecelik sabit ve 0.0-8.0 derece arasında rastgele açılarla dönmelerinin hata başarımına etkileri incelenmiştir. Burada varılan sonuç, dönme açılarının artışıyla birlikte, aynı hata başarımlarının elde edilebilmesi için daha yüksek SNR değerlerinin gerekliliğidir. 0.0-8.0 derece arasında rastgele dönme açısı değeri için, varyans değeri sıfıra indirilse dahi elde edilebilecek olan hata başarımı 0.1 olarak sınırlanmaktadır. Toplamsal Gauss gürültüsüne sahip ideal karakteristikli kanalda, dönme açılarının düzgün dağılımlı rastgele olarak değişmesi sonucu gözlenen hata başarım değeri, 6 derece ve daha yüksek, sabit açı değerleri için elde edilen hata başarımından daha iyidir.

Bozucu kanal içeren ve kanalın bozucu etkisinin dengeleyici ile giderilmeye çalışıldığı modelde, 50 adım uzunluğu ve 0.150 varyans değeri için ulaşılan hata başarımı 0.02 olarak belirlenmiştir. Toplamsal Gauss gürültüsünün gücü, dengeleyici çıkışındaki ortalama karesel hatanın minimum değerini belirlemektedir.

Bu sistemde işaret elemanlarının işaret uzayı içerisinde sabit açılarla dönmeleri sonucu ortaya çıkan bozucu etkinin dengeleyici tarafından tamamen giderildiği görülmüştür. Açı değerlerinin 0.0-8.0 derece arasında düzgüm dağılımlı rastgele olarak değişmesi ile, 50 adım uzunluğu için ulaşılabilecek minimum hata başarım değeri 0.07 varyans değeri için 0.020 olarak belirlenmiştir. Dönme açılarının sabit değerlerde alınması sonucu elde edilen hata başarımları, açı değerlerinin düzgün dağılımlı rastgele olarak değişmeleri sonucu bulunan hata başarımından daha iyidir.

KAYNAKLAR

[1] G. Ungerboeck, "Channel Coding with Multilevel / Phase Signals", IEEE Trans. on Inform. Theory, vol. IT-27, pp. 55-67, 1982.

[2] G. Ungerboeck, "Trellis Coded Modulation with Redundant Signal Sets Part 1 and 2", IEEE Com. Magazine, vol. 25, pp. 5-21, 1987.

[3] G. C. Clark, Jr. and J. B. Cain, Error-Correction Coding for Digital Communications. Newyork, 1981.

[4] J. Heller, "Viterbi Decoding For Satellite And Space Commun." , IEEE Trans. on Inform. Theory, vol. COM-19, pp. 835-848, 1975.

[5] L. F. Wei, "Rotationaly Invariant Convolutional Channel Coding with Expanded Signal Space Part1 and 2", vol. 2, pp. 659-685, 1984.

[6] CCITT , rec. V.32-V.33, Fascicle VIII:, pp. 240-266, 1988.

[7] Q. Shahid, "Adaptive Equalization", Proceedings of The IEEE, vol. 73, pp. 1349-1383, 1985.

[8] S. Ağabay, "4800 Modem İçin Adaptif Dengeleyici Simulasyon Çalışmaları", TUBİTAK Raporu, 1984.

[9] S. Haykın, Adaptive Filter Theory. Newjersey, 1986.

[10] S.B. Weinstein, "Simultaneous Two Way Data Transmission Over a Two Wire Circuit", IEEE Trans. on Commun., vol. 24, pp. 143-144, 1973.

[11] D. Messerscmitt, "Echo Cancellation In Speech and Data Transmission", IEEE Journal on Selec Areas in Commun., vol. SAC-2, pp. 283-297, 1984.

[12] W. Stephen, "A Passband Data Driven Echo Canceller for Full Dublex Transmission on Two Wire Circuits", IEEE trans. on Commun., vol. 25, pp. 654-665, 1977.

[13] H.Meyr, Synchronization in Digital Communications. Newyork, 1990.

EK A

İdeal Karakteristikli, Toplamsal Gauss Gürültülü Kanal Hata Başarımını Hesaplayan Program Listesi

```
/*sim1.c*/
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <dos.h>
#include <alloc.h>
#include <conio.h>
#define PI 3.1428
main()
{
 struct isuzay
 {
  float re;
  float im;
 };
 struct isuzay coord[128];
 float var,ss,dr,min,hb;
 float nx=0.0;float ny=0.0; float oh=0.0;
 float r1,r2;
 int n=0;int te=0;
 int i,j,k,pgk,eleman,ee,os,s,ts,as,u,w;
 float rx[11],ry[11],sx[11],sy[11];
 float gm[11][8],nmetrik[11][8];
 int ge[11][8],nsurvivor[11][8],neleman[11][8],pdgm[8][8];
 int indis[11],indis1[11];
 unsigned char q[11][7],d[11][7],r[24];
 unsigned char y0[11],y1[11],y2[11],w1[12],w2[12],w3[12],wa1[11],wa2[11];
 char numstr[40];
 FILE *fptr; FILE *fptr1; FILE *fptr2;
 struct time *sure;
 struct time *sure1;
 sure=malloc(sizeof(sure));
 gettime(sure);
 sure1=malloc(sizeof(sure1));
 fptr1=fopen("isuzay.rec","rb");
 while(fread(&coord[n],sizeof(coord[n]),1,fptr1)==1)
    n++;
 fclose(fptr1);
 fptr2=fopen("pdgm.rec","rb");
 fread(pdgm,sizeof(pdgm),1,fptr2);
 fclose(fptr2);
 randomize();
 clrscr();
 gotoxy(30,5);
 printf("\n\n Lütfen tekrar sayısını giriniz.....:");
 gets(numstr); ts=atoi(numstr);
 printf("\n\n Lütfen adım sayısını giriniz.....:");
 gets(numstr); as=atoi(numstr);
 printf("\n\n Lütfen varyansı giriniz.....:");
```

```
gets(numstr); var=atof(numstr);
ss=sqrt(var);
for(n=0;n<8;n++) nmetrik[0][n]=0.0;
for(u=1;u<ts+1;u++)                    /*Tekrar sayısı döngü başlangıcı*/
 {
  ee=0;
  for(w=1;w<as+1;w++)                  /*Adım sayısı döngü başlangıcı*/
       {
        for(k=1;k<24;k++) r[k]=0;
        y0[0]=0;y1[0]=0;y2[0]=0;w1[1]=0;w2[1]=0;w3[1]=0;
        for(i=1;i<11;i++)              /*Adim döngüsü başlangıcı*/
         {

          for (j=1;j<7;j++)
               d[i][j]=random(2);
          for (j=1;j<7;j++)
          for (j=1;j<7;j++)
               {
                q[i][j]=d[i][j]^(r[23]^r[18]);
                for(k=23;k>1;k--) r[k]=r[k-1];
                r[1]=q[i][j];
               }
          y1[i]=q[i][1]^y1[i-1];
          y2[i]=q[i][2]^y2[i-1]^(q[i][1] & y1[i-1]);
          wa1[i]=w3[i]^y2[i];
          wa2[i]=w1[i]^y1[i]^y2[i];
          w1[i+1]=w2[i];
          w2[i+1]=wa1[i]^(w2[i]&y1[i]);
          w3[i+1]=wa2[i]^(wa1[i]&w2[i]);
          y0[i]=w2[i];
          indis[i]=64*y0[i]+32*y1[i]+16*y2[i]+8*q[i][3]+4*q[i][4]+2*q[i][5]+q[i][6];
          sx[i]=coord[indis[i]].re;
          sy[i]=coord[indis[i]].im;
          r1=random(100)+1; r2=random(100)+1;
          r1=r1/100;r2=r2/100;
          nx=ss*sqrt((-2)*log10(r1))*cos(2*PI*r2);
          ny=ss*sqrt((-2)*log10(r1))*sin(2*PI*r2);
          rx[i]=sx[i]+nx;
          ry[i]=sy[i]+ny;
          for(k=0;k<8;k++)
           {
               min=1500.0;
               for(j=0;j<16;j++)
                {
                 dr=sqrt(pow((rx[i]-coord[k*16+j].re),2)+pow((ry[i]-coord[k*16+j].im),2));
                 if(min>dr)
                  {
                       eleman=j;
                       min=dr;
                  }
```

```
        }
      gm[i][k]=min;
      ge[i][k]=eleman;

  }
 for(s=0;s<8;s++)
  {
      min=1500.0;
      for(os=0;os<8;os++)
       {
        if(pdgm[os][s]!=0)
          {
            k=pdgm[os][s]-1;
            if(min > gm[i][k])
             {
              min=gm[i][k];
              eleman=k*16+ge[i][k];
              pgk=os;
             }
          }
       }
      neleman[i][s]=eleman;
      nsurvivor[i][s]=pgk;
      nmetrik[i][s]=nmetrik[i-1][pgk]+min;
  }
 }          /*Karar verme başlangıcı*/

min=1500.1;
for(s=0;s<8;s++)
 {
  if(min>nmetrik[10][s])
      {
       k=s;
       min=nmetrik[10][s];
      }
 }
for(i=10;i>0;i--)
 {
  os=nsurvivor[i][k];
  indis1[i]=neleman[i][k];
  k=os;
 }
n=0;
clrscr();
for(i=1;i<11;i++)
 {
  printf("\n\n (indis[%d],indis1[%d])-----(%d,%d)",i,i,indis[i],indis1[i]);
  if(indis[i]!=indis1[i] && n==0)
      {
       ee++;
```

```
                    sound(1300);
                    delay(200);
                    nosound();
                    n=1;
                   }
              }
          }         /*Adım sayısı döngüsü sonu*/
    te=te+ee;
   }            /*Tekrar sayısı döngüsü sonu*/
  u--; w--;
  oh=(te*1.0)/(u*w)*1.0;
  gettime(sure1);
  clrscr();
  gotoxy(30,5);
  printf("\n PROGRAM BAŞLANGIÇ ZAMANI              PROGRAM  BİTİŞ ZAMANI");
  printf("\n    %d : %d : %d                    %d : %d : %d
",(*sure).ti_hour,(*sure).ti_min,(*sure).ti_sec,(*sure1).ti_hour,(*sure1).ti_min,(*sure1).ti_sec);
  if(oh==0)
    printf("\n\n HATALI ADIM YOK.....!");
  else
   {
    hb=10*log10(oh);
    printf("\n\n TOPLAM ADIM SAYISI----------:%d",ts*as);
    printf("\n\n HATALI ADIM SAYISI----------:%d",te);
    printf("\n\n ORTALAMA HATA----------:%f",oh);
    printf("\n\n HATA BAŞARIMI----------:%f db",hb);
   }
}
```

EK B

Gauss Karakteristikli Kanalda, İşaret Elemanlarının İşaret Uzayı İçerisinde Sabit Veya Rastgele Açılarla Dönmeleri Durumu İçin Hata Başarımını Hesaplayan Program Listesi

```
/*sim1a.c*/
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <dos.h>
#include <alloc.h>
#include <conio.h>
#define PI 3.14159265
main()
{
 struct isuzay
 {
  float re;
  float im;
 };
 struct isuzay coord[128];
 float var,ss,dr,min,hb;
 float nx=0.0;float ny=0.0; float oh=0.0;
 float r1,r2;
 double theta;
 int n=0;int te=0;
 int i,j,k,pgk,eleman,ee,os,s,ts,as,u,w;
 float rx[11],ry[11],sx[11],sy[11],sx1[11],sy1[11];
 float gm[11][8],nmetrik[11][8];
 int ge[11][8],nsurvivor[11][8],neleman[11][8],pdgm[8][8];
 int indis[11],indis1[11];
 unsigned char q[11][7],d[11][7],r[24];
 unsigned char y0[11],y1[11],y2[11],w1[12],w2[12],w3[12],wa1[11],wa2[11];
 char numstr[40],bh;
 FILE *fptr; FILE *fptr1; FILE *fptr2;
 struct time *sure;
 struct time *sure1;
 sure=malloc(sizeof(sure));
 gettime(sure);
 sure1=malloc(sizeof(sure1));
 fptr1=fopen("isuzay.rec","rb");
 while(fread(&coord[n],sizeof(coord[n]),1,fptr1)==1)
    n++;
 fclose(fptr1);
 fptr2=fopen("pdgm.rec","rb");
 fread(pdgm,sizeof(pdgm),1,fptr2);
 fclose(fptr2);
 randomize();
 clrscr();
 srand(time(NULL) % 37);
 gotoxy(30,5);
 printf("\n\n Lütfen tekrar sayısını giriniz.....:");
 gets(numstr); ts=atoi(numstr);
 printf("\n\n Lütfen adım sayısını giriniz.....:");
```

```
gets(numstr); as=atoi(numstr);
printf("\n\n Lütfen varyansı giriniz.....:");
gets(numstr); var=atof(numstr);
printf("\n\n Random faz kaymaları için 'r' tuşuna basınız.");
bh=getch();
if(bh!='r')
 {
   printf("\n\n Lütfen dönme açısını (0-8) giriniz.....:");
   gets(numstr); theta=atof(numstr);
   theta=(theta*PI)/180.0;
   printf("\n theta=%f",theta);
 }
ss=sqrt(var);
for(n=0;n<8;n++) nmetrik[0][n]=0.0;
for(u=1;u<ts+1;u++)                    /*Tekrar sayısı döngü başlangıcı*/
 {
  ee=0;
  for(w=1;w<as+1;w++)                  /*Adım sayısı döngü başlangıcı*/
      {
       for(k=1;k<24;k++) r[k]=0;
       y0[0]=0;y1[0]=0;y2[0]=0;w1[1]=0;w2[1]=0;w3[1]=0;
       for(i=1;i<11;i++)               /*Adım döngüsü başlangıcı*/
        {
         if(bh==='r')
              {
               theta=8.0*rand()/(1.0+RAND_MAX);
               theta=(theta*PI)/180;
              }
         for (j=1;j<7;j++)
              d[i][j]=random(2);
         for (j=1;j<7;j++)
         for (j=1;j<7;j++)
              {
               q[i][j]=d[i][j]^(r[23]^r[18]);
               for(k=23;k>1;k--) r[k]=r[k-1];
               r[1]=q[i][j];
              }
        y1[i]=q[i][1]^y1[i-1];
        y2[i]=q[i][2]^y2[i-1]^(q[i][1] & y1[i-1]);
        wa1[i]=w3[i]^y2[i];
        wa2[i]=w1[i]^y1[i]^y2[i];
        w1[i+1]=w2[i];
        w2[i+1]=wa1[i]^(w2[i]&y1[i]);
        w3[i+1]=wa2[i]^(wa1[i]&w2[i]);
        y0[i]=w2[i];
        indis[i]=64*y0[i]+32*y1[i]+16*y2[i]+8*q[i][3]+4*q[i][4]+2*q[i][5]+q[i][6];
        sx[i]=coord[indis[i]].re;
        sy[i]=coord[indis[i]].im;
        r1=random(100)+1; r2=random(100)+1;
        r1=r1/100;r2=r2/100;
```

```
  nx=ss*sqrt((-2)*log10(r1))*cos(2*PI*r2);
  ny=ss*sqrt((-2)*log10(r1))*sin(2*PI*r2);
  sx1[i]=(sx[i]*cos(theta)-sy[i]*sin(theta));
  sy1[i]=(sy[i]*cos(theta)+sx[i]*sin(theta));
  rx[i]=sx1[i]+nx;
  ry[i]=sy1[i]+ny;
/*  printf("\nsx[%d]=%f-----sy[%d]=%f",i,sx[i],i,sy[i]);
  printf("\nsx1[%d]=%f----sy1[%d]=%f",i,sx1[i],i,sy1[i]);
  printf("\nx=%f---------ny=%f",nx,ny);
  printf("\nrx[%d]=%f-----ry[%d]=%f",i,rx[i],i,ry[i]);*/
  for(k=0;k<8;k++)
   {
        min=1500.0;
        for(j=0;j<16;j++)
         {
          dr=sqrt(pow((rx[i]-coord[k*16+j].re),2)+pow((ry[i]-coord[k*16+j].im),2));
          if(min>dr)
           {
                eleman=j;
                min=dr;
           }
         }
        gm[i][k]=min;
        ge[i][k]=eleman;

   }
  for(s=0;s<8;s++)
   {
        min=1500.0;
        for(os=0;os<8;os++)
         {
          if(pdgm[os][s]!=0)
           {
                k=pdgm[os][s]-1;
                if(min > gm[i][k])
                 {
                  min=gm[i][k];
                  eleman=k*16+ge[i][k];
                  pgk=os;
                 }
           }
         }
        neleman[i][s]=eleman;
        nsurvivor[i][s]=pgk;
        nmetrik[i][s]=nmetrik[i-1][pgk]+min;
   }
  }          /*Karar verme başlangıcı*/

 min=1500.1;
 for(s=0;s<8;s++)
```

```
            {
             if(min>nmetrik[10][s])
                   {
                    k=s;
                    min=nmetrik[10][s];
                   }
            }
           for(i=10;i>0;i--)
            {
             os=nsurvivor[i][k];
             indis1[i]=neleman[i][k];
             k=os;
            }
           n=0;
           clrscr();
           for(i=1;i<11;i++)
            {
             printf("\n\n (indis[%d],indis1[%d])-----(%d,%d)",i,i,indis[i],indis1[i]);
             if(indis[i]!=indis1[i] && n==0)
                   {
                    ee++;
                    sound(1300);
                    delay(100);
                    nosound();
                    n=1;
                   }
            }
          }          /*Adım sayısı döngüsü sonu*/
    te=te+ee;
   }             /*Tekrar sayısı döngüsü sonu*/
  u--; w--;
  oh=(te*1.0)/(u*w)*1.0;
  gettime(sure1);
  clrscr();
  gotoxy(30,5);
  printf("\n PROGRAM BAŞLANGIÇ ZAMANI              PROGRAM  BİTİŞ ZAMANI");
  printf("\n    %d : %d : %d                    %d : %d : %d
",(*sure).ti_hour,(*sure).ti_min,(*sure).ti_sec,(*sure1).ti_hour,(*sure1).ti_min,(*sure1).ti_sec);
  if(oh==0)
    printf("\n\n HATALI BLOK YOK.....!");
  else
    {
     hb=10*log10(oh);
     printf("\n\n TOPLAM ADIM SAYISI----------:%d",ts*as);
     printf("\n\n HATALI ADIM SAYISI----------:%d",te);
     printf("\n\n ORTALAMA HATA----------:%f",oh);
     printf("\n\n HATA BAŞARIMI----------:%f db",hb);
    }
}
```

EK C

Bozucu Kanal Ve Uyarlamalı Dengeleyici İçeren Sistemde, Dengeleyici Çıkışındaki Ortalama Karesel Hatanın İterasyon Sayısı İle Değişimini Hesaplayan Program Listesi

```
/*sim2.c*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <dos.h>
#include <time.h>
#define PI 3.14159265
#define SWAP(x,y) tempr=(x);(x)=(y);(y)=tempr
#define NN 1024
#define ZA 1000
#define ADAPT 500
main(argc,argv)
int argc;
char *argv[];
{
 struct isuzay
 {
  float re;
  float im;
 };
 struct isuzay coord[128];
 struct time *sure;
 struct time *sure1;
 FILE *fptr1; FILE *fptr2; FILE *fptr3; FILE *fptr4;
 FILE *fptr5; FILE *fptr6;
 unsigned char q[ZA+1][7],d[ZA+1][7],r[24];
 unsigned char
y0[ZA+1],y1[ZA+1],y2[ZA+1],w1[ZA+2],w2[ZA+2],w3[ZA+2],wa1[ZA+1],wa2[ZA+1];
 unsigned int indis[ZA+1];
 int pdgm[8][8];
 float *fvector();
 void free_vector();
 float  outkan[2*NN+1],datakan[2*NN+1];
 float hata[(ZA/10)+1];
 float *inkan,*fdata1,*fdata2;
 float *cr,*ci,*xr,*xi;
 float tempr,xt,yt,nx,ny,r1,r2,dr,min;
 float p=0.0;float ex=0.0;float ey=0.0;
 float VAR,KS;
 int f,pp,i,j,k,xl,ele,k1,SU;
 int u=1;int n=0;int jk=0;int lm=1;
 clrscr();
 if(argc!=2)
  {
    printf("\n Örnek Kullanım : C>sim2 <Dosya İsmi>");
    exit(1);
  }
 gotoxy(30,5);
 printf("\nLütfen varyans değerini giriniz.....:");
```

```
scanf("%f",&VAR);
printf("\nLütfen adaptasyon katsayısının değerini giriniz.....:");
scanf("%f",&KS);
printf("\nLütfen filtre uzunluğunu giriniz.....:");
scanf("%d",&SU);
cr=fvector(0,2*SU); ci=fvector(0,2*SU);
xr=fvector(0,2*SU+1); xi=fvector(0,2*SU+1);
inkan=fvector(1,(2*SU)+(2*ADAPT));
fdata1=fvector(1,2*(ZA-ADAPT));
fdata2=fvector(1,2*(ZA-ADAPT));
sure=malloc(sizeof(sure));
gettime(sure);
sure1=malloc(sizeof(sure1));
fptr1=fopen("isuzay.rec","rb");
while(fread(&coord[n],sizeof(coord[n]),1,fptr1)==1)
  n++;
fclose(fptr1);
fptr2=fopen("pdgm.rec","rb");
fread(pdgm,sizeof(pdgm),1,fptr2);
fclose(fptr2);
n=1;
fptr3=fopen(argv[1],"rb");
while(fread(&datakan[n],sizeof(datakan[n]),1,fptr3)==1)
  n++;
fclose(fptr3);
for(n=0;n<=2*SU+1;n++)
 {
  xr[n]=0.0;
  xi[n]=0.0;
 }
for(n=0;n<=2*SU;n++)
 {
  cr[n]=0.0;
  ci[n]=0.0;
 }
for(n=0;n<=2*SU;n++) inkan[n]=0.0;
for(n=1;n<=(ZA/10);n++)
  hata[n]=0.0;
randomize();
y0[0]=0;y1[0]=0;y2[0]=0;w1[1]=0;w2[1]=0;w3[1]=0;
for(n=1;n<=2*NN;n++) outkan[n]=0.0;
for (k=1;k<24;k++) r[k]=0;
for (i=1;i<=ZA;i++)                    /* i döngüsü başlangıcı */
  {
        for (j=1;j<7;j++)
        d[i][j]=random(2);
        for (j=1;j<7;j++)
          {
           q[i][j]=d[i][j]^(r[23]^r[18]);
           for(k=23;k>1;k--) r[k]=r[k-1];
```

```
            r[1]=q[i][j];
           }
        y1[i]=q[i][1]^y1[i-1];
        y2[i]=q[i][2]^y2[i-1]^(q[i][1] & y1[i-1]);
        wa1[i]=w3[i]^y2[i];
        wa2[i]=w1[i]^y1[i]^y2[i];
        w1[i+1]=w2[i];
        w2[i+1]=wa1[i]^(w2[i]&y1[i]);
        w3[i+1]=wa2[i]^(wa1[i]&w2[i]);
        y0[i]=w2[i];
        indis[i]=64*y0[i]+32*y1[i]+16*y2[i]+8*q[i][3]+4*q[i][4]+2*q[i][5]+q[i][6];
        if(i<=ADAPT)
          {
            inkan[(2*SU)+(2*i-1)]=coord[indis[i]].re;
            inkan[(2*SU)+(2*i)]=coord[indis[i]].im;
          }
        outkan[2*i-1]=coord[indis[i]].re; outkan[2*i]=coord[indis[i]].im;
}                                   /* i döngüsü sonu */
clrscr();
gotoxy(30,5);
sound(500); delay(200); nosound();
printf("\n128-CR işaret uzayı eşleme prosedürü tamamlandı.");
n=1;
four(outkan,NN,n);
printf("\n\n Lütfen birkaç saniye bekleyiniz.....");
for(n=1;n<2*NN;n+=2)
       {
         tempr=datakan[n]*outkan[n]-datakan[n+1]*outkan[n+1];
         outkan[n+1]=datakan[n]*outkan[n+1]+datakan[n+1]*outkan[n];
         outkan[n]=tempr;
       }
n=-1;
four(outkan,NN,n);
clrscr();
gotoxy(30,5);
sound(500); delay(200); nosound();
printf("\nBozucu kanalın etkilerinin ");
printf("mesaj dizisi üzerinde bezetim işlemi tamamlandı.");
printf("\n\nOtomatik dengeleme prosedürü başladı.");
for(pp=1;pp<=ZA;pp++)
       {
         r1=random(100)+1; r2=random(100)+1;
         r1=r1/100; r2=r2/100;
         nx=sqrt(VAR)*sqrt((-2)*log10(r1))*cos(2*PI*r2);
         ny=sqrt(VAR)*sqrt((-2)*log10(r1))*sin(2*PI*r2);
         xr[0]=(outkan[2*pp-1]/NN)+nx;
         xi[0]=(outkan[2*pp]/NN)+ny;
         if(pp>ADAPT)
           {
                fdata1[2*u-1]=xr[SU];
```

```
        fdata1[2*u]=xi[SU];
  }
xt=0.0; yt=0.0;
for(i=0;i<=2*SU;i++)
  {
        xt+=cr[i]*xr[i]-ci[i]*xi[i];
        yt+=cr[i]*xi[i]+ci[i]*xr[i];
  }
if(pp==ADAPT+1)
 {
      clrscr();
      gotoxy(30,5);
      sound(500);delay(200);nosound();
      printf("\n Adaptif dengeleme prosedürü başladı.");
 }
if(pp>ADAPT)
  {
       min=1500.0;
       for(k=0;k<=7;k++)
         {
              for(f=0;f<=15;f++)
                {
                  dr=sqrt(pow((xt-coord[k*16+f].re),2)+pow((yt-coord[k*16+f].im),2));
                  if(min>dr)
                      {
                       ele=f;
                       min=dr;
                       k1=k;
                      }
                }
         }
         ex=(xt-coord[k1*16+ele].re);
         ey=(yt-coord[k1*16+ele].im);
         p+=ex*ex+ey*ey;
         fdata2[2*u-1]=xt;
         fdata2[2*u]=yt;
         u++;
       }
 else
       {
         ex=xt-inkan[2*pp-1];
         ey=yt-inkan[2*pp];
         p+=ex*ex+ey*ey;
       }
 for(i=0;i<=2*SU;i++)
       {
        cr[i]=cr[i]-KS*ex*xr[i]-KS*ey*xi[i];
        ci[i]=ci[i]-KS*ey*xr[i]+KS*ex*xi[i];
       }
 for(i=2*SU;i>=0;i--)
```

```
                {
                  xr[i+1]=xr[i];
                  xi[i+1]=xi[i];
                }
            if(pp>=SU)
             {
                if(lm<10)
                  lm++;
                else
                  {
                   jk++;
                   xl=lm;
                   hata[jk]=10*log10(p/xl);
                   p=0.0;
                   lm=1;
                  }
             }                                  /* pp döngüsü sonu */
           }
 clrscr(); gotoxy(30,5);
 sound(500); delay(600); nosound();
 printf("\n Dengeleme işlemi tamamlandı.");
 fptr4=fopen("dinph4.dat","wb");
 fwrite(&fdata1[1],sizeof(fdata1[1]),2*(ZA-ADAPT),fptr4);
 fclose(fptr4);
 printf("\n Dengeleyici giriş dizisi bir dosyaya kaydedildi.");
 fptr5=fopen("doutph4.dat","wb");
 fwrite(&fdata2[1],sizeof(fdata2[1]),2*(ZA-ADAPT),fptr5);
 fclose(fptr5);
 printf("\n Dengeleyici çıkış dizisi bir dosyaya kaydedildi.");
 fptr6=fopen("error4.dat","wb");
 fwrite(&hata[1],sizeof(hata[1]),(ZA/10)-1,fptr6);
 fclose(fptr6);
 printf("\n Hata değerleri bir dosyaya kaydedildi.");
 delay(2600);clrscr();
 gettime(sure1);
 printf("\n   BAŞLANGIÇ ZAMANI          BİTİŞ ZAMANI");
 printf("\n  %d : %d : %d        %d : %d : %d
",(*sure).ti_hour,(*sure).ti_min,(*sure).ti_sec,(*sure1).ti_hour,(*sure1).ti_min,(*sure1).ti_sec);
 printf("\n\n***BU KANAL İÇİN OPTİMUM FİLTRE KATSAYILARI***\n");
 for(i=0;i<=2*SU;i++)
  {
   if(i>9)
        printf("\ncr[%d]=%10.7f        ci[%d]=%10.7f ",i,cr[i],i,ci[i]);
   else
        printf("\ncr[%d]=%10.7f        ci[%d]=%10.7f",i,cr[i],i,ci[i]);
  }
 return(0);
}

 float *fvector(nl,nh)
```

```
int nl,nh;
{
 float *v;
 v=(float *)malloc((unsigned) (nh-nl+1)*sizeof(float));
 return(v-nl);
}



/* FFT veya IFFT hesaplar */
four(data1,nn,isign)
float data1[];
int nn,isign;
{
 int n,mmax,m,j,istep,i;
 double wtemp,wr,wpr,wpi,wi,theta;
 float tempr,tempi;
 n=nn<<1;
 j=1;
 for(i=1;i<n;i+=2)
  {
       if(j>i)
        {
         SWAP(data1[j],data1[i]);
         SWAP(data1[j+1],data1[i+1]);
        }
       m=n>>1;
       while(m>=2 && j>m)
        {
         j-=m;
         m>>=1;
        }
       j+=m;
  }
 mmax=2;
 while(n>mmax)
  {
       istep=2*mmax;
       theta=6.283118530717959/(isign*mmax);
       wtemp=sin(0.5*theta);
       wpr=-2.0*wtemp*wtemp;
       wpi=sin(theta);
       wr=1.0;
       wi=0.0;
       for(m=1;m<mmax;m+=2)
        {
         for(i=m;i<=n;i+=istep)
          {
            j=i+mmax;
            tempr=wr*data1[j]-wi*data1[j+1];
```

```
                tempi=wr*data1[j+1]+wi*data1[j];
                data1[j]=data1[i]-tempr;
                data1[j+1]=data1[i+1]-tempi;
                data1[i]+=tempr;
                data1[i+1]+=tempi;
            }
            wr=(wtemp=wr)*wpr-wi*wpi+wr;
            wi=wi*wpr+wtemp*wpi+wi;
          }

        mmax=istep;
    }
    return(0);
}
```

EK D

Bozucu Kanal Ve Uyarlamalı Doğrusal Dengeleyici İçeren Sistemin Hata Başarımını hesaplayan program listesi

```
/*sim3.c*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <dos.h>
#include <time.h>
#define PI 3.14159265
#define SWAP(x,y) tempr=(x);(x)=(y);(y)=tempr
#define NN 1024
#define ZA 1000
#define ADAPT 500
main(argc,argv)
int argc;
char *argv[];
{
 struct isuzay
 {
  float re;
  float im;
 };
 struct isuzay coord[128];
 struct time *sure;
 struct time *sure1;
 FILE *fptr1; FILE *fptr2; FILE *fptr3; FILE *fptr4;
 FILE *fptr5; FILE *fptr6; FILE *fptr7;
 unsigned char q[ZA+1][7],d[ZA+1][7],r[24];
 unsigned char
y0[ZA+1],y1[ZA+1],y2[ZA+1],w1[ZA+2],w2[ZA+2],w3[ZA+2],wa1[ZA+1],wa2[ZA+1];
 unsigned int indis[ZA+1];
 int pdgm[8][8];
 float *fvector();
 void free_vector();
 float outkan[2*NN+1],datakan[2*NN+1];
 float hata[(ZA/10)+1];
 float gm[11][8],nmetrik[11][8];
 float *inkan,*fdata1,*fdata2,*fdata3;
 float *cr,*ci,*xi,*xr;
 float tempr,xt,yt,nx,ny,r1,r2,dr,min,hb,KS,VAR;
 float p=0.0; float ex=0.0;
 float ey=0.0; float oh=0.0;
 int f,pp,i,j,k,ele,k1,eleman,pgk,os,s,SU,xl;
 int n=0; int jk=0; int lm=1;
 int u=1; int ee=0; int te=0;
 int ge[11][8],nsurvivor[11][8],neleman[11][8],indis1[11];
 clrscr();
 gotoxy(30,5);
 printf("\nLütfen varyans değerini giriniz.....:");
 scanf("%f",&VAR);
 printf("\nLütfen adaptasyon katsayısının değerini giriniz.....:");
```

```
scanf("%f",&KS);
printf("\nLütfen filtre uzunluğunu giriniz.....:");
scanf("%d",&SU);
if(argc!=2)
 {
  printf("\nÖrnek kullanım : C>sim3 <dosya ismi>");
  exit(1);
 }
cr=fvector(0,2*SU);ci=fvector(0,2*SU);
xr=fvector(0,2*SU+1); xi=fvector(0,2*SU+1);
inkan=fvector(1,(2*SU)+(2*ADAPT));
fdata1=fvector(1,2*(ZA-ADAPT));
fdata2=fvector(1,2*(ZA-ADAPT));
fdata3=fvector(1,2*(ZA-ADAPT));
sure=malloc(sizeof(sure));
gettime(sure);
sure1=malloc(sizeof(sure1));
fptr1=fopen("isuzay.rec","rb");
while(fread(&coord[n],sizeof(coord[n]),1,fptr1)==1)
  n++;
fclose(fptr1);
clrscr();
printf("\n\n Dosya başarıyla okundu.Toplam işaret elemanı sayısı %d",n);
fptr2=fopen("pdgm.rec","rb");
fread(pdgm,sizeof(pdgm),1,fptr2);
fclose(fptr2);
printf("\n pdgm dizisi başarıyla okundu");
n=1;
fptr3=fopen(argv[1],"rb");
while(fread(&datakan[n],sizeof(datakan[n]),1,fptr3)==1)
  n++;
fclose(fptr3);
printf("\n Kanal dosyası başarıyla okundu.");
for(n=0;n<=2*SU+1;n++)
 {
  xr[n]=0.0;
  xi[n]=0.0;
 }
for(n=0;n<=2*SU;n++)
 {
  cr[n]=0.0;
  ci[n]=0.0;
 }
for(n=0;n<=2*SU;n++) inkan[n]=0.0;
for(n=1;n<=(ZA/10);n++)
  hata[n]=0.0;
randomize();
y0[0]=0;y1[0]=0;y2[0]=0;w1[1]=0;w2[1]=0;w3[1]=0;
for(n=1;n<=2*NN;n++) outkan[n]=0.0;
for (k=1;k<24;k++) r[k]=0;
```

```
for (i=1;i<=ZA;i++)                    /* i döngüsü başlangıcı */
  {
        for (j=1;j<7;j++)
        d[i][j]=random(2);
        for (j=1;j<7;j++)
          {
           q[i][j]=d[i][j]^(r[23]^r[18]);
           for(k=23;k>1;k--) r[k]=r[k-1];
           r[1]=q[i][j];
          }
         y1[i]=q[i][1]^y1[i-1];
         y2[i]=q[i][2]^y2[i-1]^(q[i][1] & y1[i-1]);
         wa1[i]=w3[i]^y2[i];
         wa2[i]=w1[i]^y1[i]^y2[i];
         w1[i+1]=w2[i];
         w2[i+1]=wa1[i]^(w2[i]&y1[i]);
         w3[i+1]=wa2[i]^(wa1[i]&w2[i]);
         y0[i]=w2[i];
         indis[i]=64*y0[i]+32*y1[i]+16*y2[i]+8*q[i][3]+4*q[i][4]+2*q[i][5]+q[i][6];
         if(i<=ADAPT)
          {
            inkan[(2*SU)+(2*i-1)]=coord[indis[i]].re;
            inkan[(2*SU)+(2*i)]=coord[indis[i]].im;
          }
         outkan[2*i-1]=coord[indis[i]].re;
         outkan[2*i]=coord[indis[i]].im;
  }                                    /* i döngüsü sonu */
  clrscr();
  gotoxy(30,5);
  sound(500); delay(200); nosound();
  printf("\n128-CR işaret uzayı eşleme prosedürü tamamlandı.");
  n=1;
  four(outkan,NN,n);
  printf("\n\n\n Lütfen birkaç saniye bekleyiniz.....");
  for(n=1;n<2*NN;n+=2)
        {
         tempr=datakan[n]*outkan[n]-datakan[n+1]*outkan[n+1];
         outkan[n+1]=datakan[n]*outkan[n+1]+datakan[n+1]*outkan[n];
         outkan[n]=tempr;
        }
  n=-1;
  four(outkan,NN,n);
  clrscr();
  gotoxy(30,5);
  sound(500); delay(200); nosound();
  printf("\nBozucu kanalın etkilerinin ");
  printf("\nmesaj dizisi üzerindeki benzetimi tamamlandı.");
  printf("\n\nOtomatik dengeleme prosedürü başladı.");
  for(pp=1;pp<=ZA;pp++)
        {
```

```
r1=random(100)+1; r2=random(100)+1;
r1=r1/100; r2=r2/100;
nx=sqrt(VAR)*sqrt((-2)*log10(r1))*cos(2*PI*r2);
ny=sqrt(VAR)*sqrt((-2)*log10(r1))*sin(2*PI*r2);
xr[0]=(outkan[2*pp-1]/NN)+nx;
xi[0]=(outkan[2*pp]/NN)+ny;
if(pp>ADAPT)
  {
        fdata1[2*u-1]=xr[SU];
        fdata1[2*u]=xi[SU];
  }
xt=0.0; yt=0.0;
for(i=0;i<=2*SU;i++)
  {
        xt+=cr[i]*xr[i]-ci[i]*xi[i];
        yt+=cr[i]*xi[i]+ci[i]*xr[i];
  }
if(pp==ADAPT+1)
 {
      clrscr();
      gotoxy(30,5);
      sound(500);delay(200);nosound();
      printf("\n Otomatik denegeleme prodedürü başladı.");
 }
if(pp>ADAPT)
  {
        min=1500.0;
        for(k=0;k<=7;k++)
          {
              for(f=0;f<=15;f++)
                {
                  dr=sqrt(pow((xt-coord[k*16+f].re),2)+pow((yt-coord[k*16+f].im),2));
                  if(min>dr)
                       {
                        ele=f;
                        min=dr;
                        k1=k;
                       }
                }
              }
          ex=(xt-coord[k1*16+ele].re);
          ey=(yt-coord[k1*16+ele].im);
          p+=ex*ex+ey*ey;
          fdata2[2*u-1]=xt;
          fdata2[2*u]=yt;
          u++;
       }
else
       {
          ex=xt-inkan[2*pp-1];
```

```
                ey=yt-inkan[2*pp];
                p+=ex*ex+ey*ey;
              }
        for(i=0;i<=2*SU;i++)
              {
                cr[i]=cr[i]-KS*ex*xr[i]-KS*ey*xi[i];
                ci[i]=ci[i]-KS*ey*xr[i]+KS*ex*xi[i];
              }
        for(i=2*SU;i>=0;i--)
              {
                xr[i+1]=xr[i];
                xi[i+1]=xi[i];
              }
        if(pp>=SU)
          {
              if(lm<10)
                lm++;
              else
                {
                 jk++;
                 xl=lm;
                 hata[jk]=10*log10(p/xl);
                 p=0.0;
                 lm=1;
                }
              }
      }                          /* pp döngüsü sonu*/

clrscr(); gotoxy(30,5);
sound(500); delay(600);nosound();
fptr4=fopen("dinph4.dat","wb");
fwrite(&fdata1[1],sizeof(fdata1[1]),2*(ZA-ADAPT),fptr4);
fclose(fptr4);
printf("\n Dengeleyici giriş dizisi bir dosyaya kaydedildi");
fptr5=fopen("doutph4.dat","wb");
fwrite(&fdata2[1],sizeof(fdata2[1]),2*(ZA-ADAPT),fptr5);
fclose(fptr5);
printf("\n Dengeleyici çıkış dizisi bir dosyaya kaydedildi");
fptr6=fopen("error4.dat","wb");
fwrite(&hata[1],sizeof(hata[1]),(ZA/10)-1,fptr6);
fclose(fptr6);
printf("\n Hata değerleri bir dosyaya kaydedildi.");
for(n=0;n<8;n++) nmetrik[0][n]=0.0;
delay(2600);clrscr(); gotoxy(30,5);
sound(500);delay(600);nosound();
printf("\nViterbi ko çözme prosedürü başladı.");
for(pp=0;pp<=((ZA-ADAPT)/10)-1;pp++)
 {
  ee=0.0;
  for(i=1;i<=10;i++)
```

```
        {
         for(k=0;k<8;k++)
           {
                 min=1500.0;
                 for(j=0;j<16;j++)
                  {
                    dr=sqrt(pow((fdata2[2*(pp*10+i)-1]-
coord[k*16+j].re),2)+pow((fdata2[2*(pp*10+i)]-coord[k*16+j].im),2));
                    if(min>dr)
                     {
                          eleman=j;
                          min=dr;
                     }
                  }
                 gm[i][k]=min;
                 ge[i][k]=eleman;
            }
          for(s=0;s<8;s++)
           {
                 min=1500.0;
                 for(os=0;os<8;os++)
                  {
                   if(pdgm[os][s]!=0)
                         {
                           k=pdgm[os][s]-1;
                           if(min>gm[i][k])
                             {
                                   min=gm[i][k];
                                   eleman=k*16+ge[i][k];
                                   pgk=os;
                             }
                         }
                  }
                 neleman[i][s]=eleman;
                 nsurvivor[i][s]=pgk;
                 nmetrik[i][s]=nmetrik[i-1][pgk]+min;
            }
        }          /*Karar prosedürü başlangıcı*/
    clrscr(); gotoxy(30,5);
    min=1500.0;
    for(s=0;s<8;s++)
        {
         if(min>nmetrik[10][s])
           {
                 k=s;
                 min=nmetrik[10][s];
           }
        }
    for(i=10;i>0;i--)
        {
```

```
            os=nsurvivor[i][k];
            indis1[i]=neleman[i][k];
            k=os;
           }
    n=0;
    clrscr(); gotoxy(30,5);
    for(i=1;i<11;i++)
          {
           printf("\n\n (indis,indis1)------(%d,%d)",indis[(ADAPT-SU)+(pp*10+i)],indis1[i]);
           fdata3[2*(pp*10+i)-1]=coord[indis1[i]].re;
           fdata3[2*(pp*10+i)]=coord[indis1[i]].im;
           if(indis[(ADAPT-SU)+(pp*10+i)]!=indis1[i] && n==0)
            {
             ee++;
             sound(1700);
             delay(200);
             nosound();
             n=1;
            }
          }
     te+=ee;
    }                          /*pp çevrimi sonu*/
  fptr7=fopen("voutph4.dat","wb");
  fwrite(&fdata3[1],sizeof(fdata3[1]),2*(ZA-ADAPT),fptr7);
  clrscr(); gotoxy(30,5);
  sound(500); delay(600);nosound();
  printf("\n Viterbi kod çözücü çıkışı bir dosyaya kaydedildi");
  fclose(fptr7);
  delay(2000); clrscr();
  gettime(sure1);
  printf("\n\n   BAŞLANGIÇ ZAMANI        BİTİŞ ZAMANI ");
  printf("\n   %d : %d : %d       %d : %d : %d
",(*sure).ti_hour,(*sure).ti_min,(*sure).ti_sec,(*sure1).ti_hour,(*sure1).ti_min,(*sure1).ti_sec);
  gotoxy(30,5);
  oh=((float) te)/((float) ADAPT/10.0);
  if(oh==0)
    printf("\n\nHatalı blok yok.....!");
  else
   {
    hb=10*log10(oh);
    printf("\n\nTOPLAM ADIM SAYISI----------: %d",ADAPT/10);
    printf("\n\nHATALI ADIM SAYISI----------: %d",te);
    printf("\n\nORTALAMA HATA----------: %f",oh);
    printf("\n\nHATA BAŞARIMI (dB)----------: %f",hb);
   }
}

 float *fvector(nl,nh)
 int nl,nh;
 {
```

```
 float *v;
 v=(float *)malloc((unsigned) (nh-nl+1)*sizeof(float));
 return(v-nl);
}



/* FFT veya IFFT hesaplar */
four(data1,nn,isign)
float data1[];
int nn,isign;
{
 int n,mmax,m,j,istep,i;
 double wtemp,wr,wpr,wpi,wi,theta;
 float tempr,tempi;
 n=nn<<1;
 j=1;
 for(i=1;i<n;i+=2)
  {
       if(j>i)
        {
         SWAP(data1[j],data1[i]);
         SWAP(data1[j+1],data1[i+1]);
        }
       m=n>>1;
       while(m>=2 && j>m)
        {
         j-=m;
         m>>=1;
        }
       j+=m;
  }
 mmax=2;
 while(n>mmax)
  {
       istep=2*mmax;
       theta=6.283118530717959/(isign*mmax);
       wtemp=sin(0.5*theta);
       wpr=-2.0*wtemp*wtemp;
       wpi=sin(theta);
       wr=1.0;
       wi=0.0;
       for(m=1;m<mmax;m+=2)
        {
         for(i=m;i<=n;i+=istep)
          {
               j=i+mmax;
               tempr=wr*data1[j]-wi*data1[j+1];
               tempi=wr*data1[j+1]+wi*data1[j];
               data1[j]=data1[i]-tempr;
```

```
                data1[j+1]=data1[i+1]-tempi;
                data1[i]+=tempr;
                data1[i+1]+=tempi;
            }
            wr=(wtemp=wr)*wpr-wi*wpi+wr;
            wi=wi*wpr+wtemp*wpi+wi;
          }

        mmax=istep;
    }
    return(0);
}
```

EK E

Bozucu Kanal Ve Uyarlamalı Doğrusal Dengeleyici İçeren Sistemde, İşaret Elemanlarının İşaret Uzayı İçerisinde Sabit Veya Rastgele Açılarla Dönmeleri Durumunda, Sistemin Hata Başarımını Hesaplayan Program Listesi

```
/*sim3a.c*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <dos.h>
#include <time.h>
#include <alloc.h>
#define PI 3.14159265
#define SWAP(x,y) tempr=(x);(x)=(y);(y)=tempr
#define NN 1024
#define ZA 1000
#define ADAPT 500
main(argc,argv)
int argc;
char *argv[];
{
 struct isuzay
 {
  float re;
  float im;
 };
 struct isuzay coord[128];
 FILE *fptr1; FILE *fptr2; FILE *fptr3; FILE *fptr4;
 FILE *fptr5; FILE *fptr6; FILE *fptr7;
 unsigned char q[ZA+1][7],d[ZA+1][7],r[24];
 unsigned char
y0[ZA+1],y1[ZA+1],y2[ZA+1],w1[ZA+2],w2[ZA+2],w3[ZA+2],wa1[ZA+1],wa2[ZA+1];
 char numstr[40],bh;
 unsigned int indis[ZA+1];
 int pdgm[8][8];
 float *fvector();
 void free_vector();
 float outkan[2*NN+1],datakan[2*NN+1];
 float hata[(ZA/10)+1];
 float gm[11][8],nmetrik[11][8];
 float *inkan,*fdata1,*fdata2,*fdata3;
 float *cr,*ci,*xr,*xi;
 float tempr,xt,yt,edr,nx,ny,r1,r2,min,hb;
 float VAR;float KS;
 float p=0.0; float ex=0.0;
 float ey=0.0; float oh=0.0;
 double theta;
 int SU; int SU1;
 int f,pp,i,j,k,ele,k1,eleman,pgk,os,s,xl;
 int jk=0; int lm=1;
 int u=1; int ee=0; int te=0;  int n=0;
 int ge[11][8],nsurvivor[11][8],neleman[11][8],indis1[11];
 if(argc!=2)
```

```
 {
   printf("Example usage : C>sim3a <filename>");
   exit(1);
 }
clrscr();
gotoxy(30,5);
printf("\nLütfen varyans değerini giriniz.....:");
scanf("%f",&VAR);
printf("\nLütfen adaptasyon katsayısının değerini giriniz..:");
scanf("%f",&KS);
printf("\nLütfen filtre uzunluğunu giriniz.....:");
scanf("%d",&SU);
cr=fvector(0,2*SU);ci=fvector(0,2*SU);
xr=fvector(0,2*SU+1); xi=fvector(0,2*SU+1);
inkan=fvector(1,(2*SU)+(2*ADAPT));
fdata1=fvector(1,2*(ZA-ADAPT));
fdata2=fvector(1,2*(ZA-ADAPT));
fptr1=fopen("isuzay.rec","rb");
while(fread(&coord[n],sizeof(coord[n]),1,fptr1)==1)
   n++;
fclose(fptr1);
clrscr();
printf("\n\n Dosya başarıyla okundu.Toplam işaret elemanı sayısı %d",n);
fptr2=fopen("pdgm.rec","rb");
fread(pdgm,sizeof(pdgm),1,fptr2);
fclose(fptr2);
printf("\n pdgm dizisi başarıyla okundu");
n=1;
fptr3=fopen(argv[1],"rb");
while(fread(&datakan[n],sizeof(datakan[n]),1,fptr3)==1)
  n++;
fclose(fptr3);
printf("\n Kanal dosyası başarıyla okundu.");
for(n=0;n<=2*SU+1;n++)
 {
   xr[n]=0.0;
   xi[n]=0.0;
 }
for(n=0;n<=2*SU;n++)
 {
   cr[n]=0.0;
   ci[n]=0.0;
   cr[SU]=1.0;
 }
for(n=0;n<=2*SU;n++) inkan[n]=0.0;
for(n=1;n<=(ZA/10);n++)
  hata[n]=0.0;
randomize();
clrscr();
srand(time(NULL) % 37);
```

```
gotoxy(30,5);
printf("\nRandom dönme açıları için 'r' tuşuna basınız.");
bh=getch();
if(bh!='r')
 {
   clrscr();
   gotoxy(30,5);
   printf("\nLütfen dönme açısını (0-8) giriniz.....:");
   scanf("%s",numstr);
   theta=atof(numstr);
   theta=(theta*PI)/180.0;
 }
y0[0]=0;y1[0]=0;y2[0]=0;w1[1]=0;w2[1]=0;w3[1]=0;
for(n=1;n<=2*NN;n++) outkan[n]=0.0;
for (k=1;k<24;k++) r[k]=0;
for (i=1;i<=ZA;i++)                    /* i döngüsü başlangıcı */
  {
        for (j=1;j<7;j++)
        d[i][j]=random(2);
        for (j=1;j<7;j++)
          {
           q[i][j]=d[i][j]^(r[23]^r[18]);
           for(k=23;k>1;k--) r[k]=r[k-1];
           r[1]=q[i][j];
          }
        y1[i]=q[i][1]^y1[i-1];
        y2[i]=q[i][2]^y2[i-1]^(q[i][1] & y1[i-1]);
        wa1[i]=w3[i]^y2[i];
        wa2[i]=w1[i]^y1[i]^y2[i];
        w1[i+1]=w2[i];
        w2[i+1]=wa1[i]^(w2[i]&y1[i]);
        w3[i+1]=wa2[i]^(wa1[i]&w2[i]);
        y0[i]=w2[i];
        indis[i]=64*y0[i]+32*y1[i]+16*y2[i]+8*q[i][3]+4*q[i][4]+2*q[i][5]+q[i][6];
        if(i<=ADAPT)
          {
            inkan[(2*SU)+(2*i-1)]=coord[indis[i]].re;
            inkan[(2*SU)+(2*i)]=coord[indis[i]].im;
          }
        if(bh==='r')
         {
           theta=10.0*rand()/(1.0+RAND_MAX);
           theta=(theta*PI)/180;
         }
        outkan[2*i-1]=coord[indis[i]].re*cos(theta)-coord[indis[i]].im*sin(theta);
        outkan[2*i]=coord[indis[i]].im*cos(theta)+coord[indis[i]].re*sin(theta);
  }
                                                   /* i d"ng▒s▒ sonu */
  fbeep();
  printf("\n128-CR işaret uzayı eşleme prosedürü tamamlandı.");
```

```
n=1;
four(outkan,NN,n);
printf("\n\n\n Lütfen birkaç saniye bekleyiniz.....");
for(n=1;n<2*NN;n+=2)
      {
       tempr=datakan[n]*outkan[n]-datakan[n+1]*outkan[n+1];
       outkan[n+1]=datakan[n]*outkan[n+1]+datakan[n+1]*outkan[n];
       outkan[n]=tempr;
      }
n=-1;
four(outkan,NN,n);
fbeep();
printf("\nBozucu kanalın etkilerinin ");
printf("\n.mesaj dizisi üzerindeki benzetimi tamamlandı");
printf("\n\nOtomatik dengeleme prosedürü başladı.");
for(pp=1;pp<=ZA;pp++)
      {
       r1=random(100)+1; r2=random(100)+1;
       r1=r1/100; r2=r2/100;
       nx=sqrt(VAR)*sqrt((-2)*log10(r1))*cos(2*PI*r2);
       ny=sqrt(VAR)*sqrt((-2)*log10(r1))*sin(2*PI*r2);
       xr[0]=(outkan[2*pp-1]/NN)+nx;
       xi[0]=(outkan[2*pp]/NN)+ny;
       if(pp>ADAPT)
         {
              fdata1[2*u-1]=xr[SU];
              fdata1[2*u]=xi[SU];
         }
       xt=0.0; yt=0.0;
       for(i=0;i<=2*SU;i++)
         {
              xt+=cr[i]*xr[i]-ci[i]*xi[i];
              yt+=cr[i]*xi[i]+ci[i]*xr[i];
         }
       if(pp==ADAPT+1)
        {
             fbeep();
             printf("\n Adaptif dengeleme prosedürü başladı.");
        }
       if(pp>ADAPT)
        {
         min=1500.0;
         for(k=0;k<=7;k++)
              {
               for(f=0;f<=15;f++)
                 {
                      edr=sqrt(pow((xt-coord[k*16+f].re),2)+pow((yt-coord[k*16+f].im),2));
                      if(min>edr)
                        {
                         ele=f;
```

```
                    min=edr;
                    k1=k;
                   }
                }
              }
            }
          if(pp>ADAPT)
            {
              ex=(xt-coord[k1*16+ele].re);
              ey=(yt-coord[k1*16+ele].im);
              p+=ex*ex+ey*ey;
              fdata2[2*u-1]=xt;
              fdata2[2*u]=yt;
              u++;
            }
          else
              {
                ex=xt-inkan[2*pp-1];
                ey=yt-inkan[2*pp];
                p+=ex*ex+ey*ey;
              }
           for(i=0;i<=2*SU;i++)
              {
                cr[i]=cr[i]-KS*ex*xr[i]-KS*ey*xi[i];
                ci[i]=ci[i]-KS*ey*xr[i]+KS*ex*xi[i];
              }
           for(i=2*SU;i>=0;i--)
              {
                xr[i+1]=xr[i];
                xi[i+1]=xi[i];
              }
           if(pp>=SU)
            {
              if(lm<10)
                lm++;
              else
                {
                 jk++;
                 xl=lm;
                 hata[jk]=10*log10(p/xl);
                 p=0.0;
                 lm=1;
                }
              }
          }                          /* pp döngüsü sonu*/

clrscr(); gotoxy(30,5);
sound(500); delay(600);nosound();
fptr4=fopen("dinph.dat","wb");
fwrite(&fdata1[1],sizeof(fdata1[1]),2*(ZA-ADAPT),fptr4);
```

```
fclose(fptr4);
printf("\n Dengeleyici giriş dizisi bir dosyaya kaydedildi");
fptr5=fopen("doutph.dat","wb");
fwrite(&fdata2[1],sizeof(fdata2[1]),2*(ZA-ADAPT),fptr5);
fclose(fptr5);
printf("\n Dengeleyici çıkış dizisi bir dosyaya kaydedildi");
fptr6=fopen("error.dat","wb");
fwrite(&hata[1],sizeof(hata[1]),(ZA/10)-1,fptr6);
fclose(fptr6);
printf("\n Hata değerleri bir dosyaya kaydedildi.");
for(n=0;n<8;n++) nmetrik[0][n]=0.0;
delay(2600);
fbeep();
printf("\nViterbi kod çözme prosedürü başladı.");
for(pp=0;pp<=((ZA-ADAPT)/10)-1;pp++)
 {
  ee=0.0;
  for(i=1;i<=10;i++)
      {
       for(k=0;k<8;k++)
         {
            min=1500.0;
            for(j=0;j<16;j++)
             {
              edr=sqrt(pow((fdata2[2*(pp*10+i)-1]-
coord[k*16+j].re),2)+pow((fdata2[2*(pp*10+i)]-coord[k*16+j].im),2));
              if(min>edr)
               {
                  eleman=j;
                  min=edr;
               }
             }
            gm[i][k]=min;
            ge[i][k]=eleman;
         }
       for(s=0;s<8;s++)
         {
            min=1500.0;
            for(os=0;os<8;os++)
              {
               if(pdgm[os][s]!=0)
                    {
                     k=pdgm[os][s]-1;
                     if(min>gm[i][k])
                       {
                            min=gm[i][k];

                            eleman=k*16+ge[i][k];
                            pgk=os;
                       }
```

```
                    }
                  }
                neleman[i][s]=eleman;
                nsurvivor[i][s]=pgk;
                nmetrik[i][s]=nmetrik[i-1][pgk]+min;
             }
        }          /*Karar verme prosedürü başlangıcı*/
 min=1500.0;
 for(s=0;s<8;s++)
        {
         if(min>nmetrik[10][s])
           {
                k=s;
                min=nmetrik[10][s];
           }
        }
for(i=10;i>0;i--)
       {
        os=nsurvivor[i][k];
        indis1[i]=neleman[i][k];
        k=os;
       }
 n=0;
 clrscr(); gotoxy(30,5);
 for(i=1;i<11;i++)
        {
         printf("\n\n (indis,indis1)------(%d,%d)",indis[(ADAPT-SU)+(pp*10+i)],indis1[i]);
         fdata1[2*(pp*10+i)-1]=coord[indis1[i]].re;
         fdata1[2*(pp*10+i)]=coord[indis1[i]].im;
         if(indis[(ADAPT-SU)+(pp*10+i)]!=indis1[i] && n==0)
           {
            ee++;
            sound(1700);
            delay(200);
            nosound();
            n=1;
           }
        }
  te+=ee;
 }                          /*pp çevrimi sonu*/
fptr7=fopen("voutph.dat","wb");
fwrite(&fdata1[1],sizeof(fdata1[1]),2*(ZA-ADAPT),fptr7);
fbeep();
printf("\n Viterbi kod çözücü çıkışı bir dosyaya kaydedildi");
fclose(fptr7);
delay(1700); clrscr();
gotoxy(30,5);
oh=((float) te)/((float) ADAPT/10.0);
if(oh==0)
  printf("\n\nHatalı blok yok.....!");
```

```
else
 {
  hb=10*log10(oh);
  printf("\n\nTOPLAM ADIM SAYISI----------: %d",ADAPT/10);
  printf("\n\nhATALI ADIM SAYISI----------: %d",te);
  printf("\n\nORTALAMA HATA----------: %f",oh);
  printf("\n\nHATA BAŞARIMI (dB)----------: %f",hb);
 }
getch();
fbeep();
clrscr();
for(i=0;i<=2*SU;i++)
 {
  if(i>9)
    printf("\ncr[%d]=%10.7f          ci[%d]=%10.7f",i,cr[i],i,ci[i]);
  else
    printf("\ncr[%d]=%10.7f          ci[%d]=%10.7f",i,cr[i],i,ci[i]);
 }
return(0);
}

float *fvector(nl,nh)
int nl,nh;
{
 float *v;
 v=(float *)malloc((unsigned) (nh-nl+1)*sizeof(float));
 return(v-nl);
}

fbeep()
{
 clrscr();
 gotoxy(30,5);
 sound(500); delay(200);
 nosound();
}



/* FFT veya IFFT hesaplar */
four(data1,nn,isign)
float data1[];
int nn,isign;
{
 int n,mmax,m,j,istep,i;
 double wtemp,wr,wpr,wpi,wi,theta;
 float tempr,tempi;
 n=nn<<1;
 j=1;
 for(i=1;i<n;i+=2)
```

```
 {
      if(j>i)
       {
        SWAP(data1[j],data1[i]);
        SWAP(data1[j+1],data1[i+1]);
       }
      m=n>>1;
      while(m>=2 && j>m)
       {
        j-=m;
        m>>=1;
       }
      j+=m;
 }
mmax=2;
while(n>mmax)
 {
      istep=2*mmax;
      theta=6.283118530717959/(isign*mmax);
      wtemp=sin(0.5*theta);
      wpr=-2.0*wtemp*wtemp;
      wpi=sin(theta);
      wr=1.0;
      wi=0.0;
      for(m=1;m<mmax;m+=2)
       {
        for(i=m;i<=n;i+=istep)
          {
               j=i+mmax;
               tempr=wr*data1[j]-wi*data1[j+1];
               tempi=wr*data1[j+1]+wi*data1[j];
               data1[j]=data1[i]-tempr;
               data1[j+1]=data1[i+1]-tempi;
               data1[i]+=tempr;
               data1[i+1]+=tempi;
          }
        wr=(wtemp=wr)*wpr-wi*wpi+wr;
        wi=wi*wpr+wtemp*wpi+wi;
       }

      mmax=istep;
 }
 return(0);
}
```

EK F

Bozucu Kanal Ve Karar Geri Beslemeli Uyarlamalı Dengeleyici İçeren Sistemde, Dengeleyici Çıkışındaki Ortalama Karesel Hatanın İterasyon Sayısı İle Değişimini Hesaplayan Program Listesi

```
/*sim4.c*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <dos.h>
#include <time.h>
#include <alloc.h>
#define PI 3.14159265
#define SWAP(x,y) tempr=(x);(x)=(y);(y)=tempr
#define NN 1024
#define ZA 1000
#define ADAPT 500
main()
{
 struct isuzay
 {
  float re;
  float im;
 };
 struct isuzay coord[128];
 FILE *fptr1; FILE *fptr2; FILE *fptr3; FILE *fptr4;
 FILE *fptr5; FILE *fptr6; FILE *fptr7;
 unsigned char q[ZA+1][7],d[ZA+1][7],r[24];
 unsigned char
y0[ZA+1],y1[ZA+1],y2[ZA+1],w1[ZA+2],w2[ZA+2],w3[ZA+2],wa1[ZA+1],wa2[ZA+1];
 unsigned int indis[ZA+1];
 int pdgm[8][8];
 float *fvector();
 void free_vector();
 float outkan[2*NN+1],datakan[2*NN+1];
 float hata[(ZA/10)+1];
 float gm[11][8],nmetrik[11][8];
 float *inkan,*fdata1,*fdata2,*fdata3;
 float *cr,*ci,*xi,*xr,*dr,*di,*xr1,*xi1;
 float tempr,xt,yt,xt1,yt1,txt,tyt,edr,nx,ny,r1,r2,min,hb;
 float VAR;float KS; float KS1;
 float p=0.0; float ex=0.0;
 float ey=0.0; float oh=0.0;
 int SU; int SU1;
 int f,pp,i,j,k,ele,k1,eleman,pgk,os,s,xl;
 int jk=0; int lm=1;
 int u=1; int ee=0; int te=0;  int n=0;
 int ge[11][8],nsurvivor[11][8],neleman[11][8],indis1[11];
 clrscr();
 gotoxy(30,5);
 printf("\nLütfen varyans değerini giriniz.....:");
 scanf("%f",&VAR);
 printf("\nLütfen ileri yol filtresi adaptasyon katsayısını giriniz...:");
 scanf("%f",&KS);
```

```
printf("\nLütfen geri yol filtresi adaptasyon katsayısını giriniz...:");
scanf("%f",&KS1);
printf("\nLütfen ileri yol filtre uzunluğunu giriniz.....:");
scanf("%d",&SU);
printf("\nLütfen geri yol filtre uzunluğunu giriniz.....:");
scanf("%d",&SU1);
cr=fvector(0,2*SU);ci=fvector(0,2*SU);
dr=fvector(0,2*SU1);di=fvector(0,2*SU1);
xr=fvector(0,2*SU+1); xi=fvector(0,2*SU+1);
 if(SU1!=0)
  {
   xr1=fvector(0,2*SU1+1);
   xi1=fvector(0,2*SU1+1);
  }
inkan=fvector(1,(2*SU)+(2*ADAPT));
fdata1=fvector(1,2*(ZA-ADAPT));
fdata2=fvector(1,2*(ZA-ADAPT));
fptr1=fopen("isuzay.rec","rb");
while(fread(&coord[n],sizeof(coord[n]),1,fptr1)==1)
   n++;
fclose(fptr1);
clrscr();
printf("\n\n Dosya başarıyla okundu Toplam işaret elemanı sayısı%d",n);
fptr2=fopen("pdgm.rec","rb");
fread(pdgm,sizeof(pdgm),1,fptr2);
fclose(fptr2);
printf("\n pdgm dizisi başarıyla okundu");
n=1;
fptr3=fopen("kanal1.dat","rb");
while(fread(&datakan[n],sizeof(datakan[n]),1,fptr3)==1)
  n++;
fclose(fptr3);
printf("\n Kanal dosyası başarıyla okundu.");
for(n=0;n<=2*SU+1;n++)
 {
  xr[n]=0.0;
  xi[n]=0.0;
 }
xr[SU]=1.0;
for(n=0;n<=2*SU;n++)
 {
  cr[n]=0.0;
  ci[n]=0.0;
 }
for(n=0;n<=2*SU1;n++)
 {
  dr[n]=0.0;
  di[n]=0.0;
  xr1[n]=0.0;
  xi1[n]=0.0;
```

```
 }
xr1[2*SU1+1]=0.0; xi1[2*SU1+1]=0.0;
for(n=0;n<=2*SU;n++) inkan[n]=0.0;
for(n=1;n<=(ZA/10);n++)
  hata[n]=0.0;
randomize();
y0[0]=0;y1[0]=0;y2[0]=0;w1[1]=0;w2[1]=0;w3[1]=0;
for(n=1;n<=2*NN;n++) outkan[n]=0.0;
for (k=1;k<24;k++) r[k]=0;
for (i=1;i<=ZA;i++)                    /* i döngüsü başlangıcı */
  {
        for (j=1;j<7;j++)
        d[i][j]=random(2);
        for (j=1;j<7;j++)
          {
           q[i][j]=d[i][j]^(r[23]^r[18]);
           for(k=23;k>1;k--) r[k]=r[k-1];
           r[1]=q[i][j];
          }
         y1[i]=q[i][1]^y1[i-1];
         y2[i]=q[i][2]^y2[i-1]^(q[i][1] & y1[i-1]);
         wa1[i]=w3[i]^y2[i];
         wa2[i]=w1[i]^y1[i]^y2[i];
         w1[i+1]=w2[i];
         w2[i+1]=wa1[i]^(w2[i]&y1[i]);
         w3[i+1]=wa2[i]^(wa1[i]&w2[i]);
         y0[i]=w2[i];
         indis[i]=64*y0[i]+32*y1[i]+16*y2[i]+8*q[i][3]+4*q[i][4]+2*q[i][5]+q[i][6];
         if(i<=ADAPT)
          {
           inkan[(2*SU)+(2*i-1)]=coord[indis[i]].re;
           inkan[(2*SU)+(2*i)]=coord[indis[i]].im;
          }
         outkan[2*i-1]=coord[indis[i]].re;
         outkan[2*i]=coord[indis[i]].im;
  }                                    /* i döngüsü sonu */
  fbeep();
  printf("\n128-CR işaret uzayı eşeleme prosedürü.tamamlandı");
  n=1;
  four(outkan,NN,n);
  printf("\n\n\n Lütfen birkaç saniye bekleyiniz.....");
  for(n=1;n<2*NN;n+=2)
        {
         tempr=datakan[n]*outkan[n]-datakan[n+1]*outkan[n+1];
         outkan[n+1]=datakan[n]*outkan[n+1]+datakan[n+1]*outkan[n];
         outkan[n]=tempr;
        }
  n=-1;
  four(outkan,NN,n);
  fbeep();
```

```
printf("\nBozucu kanalın etkilerinin ");
printf("\nmesaj dizisi üzerindeki benzetimi tamamlandı.");
printf("\n\nOtomatik dengeleme prosedürü başladı.");
for(pp=1;pp<=ZA;pp++)
      {
        r1=random(100)+1; r2=random(100)+1;
        r1=r1/100; r2=r2/100;
        nx=sqrt(VAR)*sqrt((-2)*log10(r1))*cos(2*PI*r2);
        ny=sqrt(VAR)*sqrt((-2)*log10(r1))*sin(2*PI*r2);
        xr[0]=(outkan[2*pp-1]/NN)+nx;
        xi[0]=(outkan[2*pp]/NN)+ny;
        if(pp>ADAPT)
          {
              fdata1[2*u-1]=xr[SU];
              fdata1[2*u]=xi[SU];
          }
        xt=0.0; yt=0.0; xt1=0.0; yt1=0.0;
        txt=0.0; tyt=0.0;
        for(i=0;i<=2*SU;i++)
          {
              xt+=cr[i]*xr[i]-ci[i]*xi[i];
              yt+=cr[i]*xi[i]+ci[i]*xr[i];
          }
        if(SU1!=0)
          {
              for(i=0;i<=2*SU1;i++)
                {
                  xt1+=dr[i]*xr1[i]-di[i]*xi1[i];
                  yt1+=dr[i]*xi1[i]+di[i]*xr1[i];
                }
          }
        txt=xt-xt1;
        tyt=yt-yt1;
        if(pp==ADAPT+1)
          {
             fbeep();
             printf("\n Adaptif denegeleme prosedürü başladı.");
          }
        if(pp>SU)
         {
          min=1500.0;
          for(k=0;k<=7;k++)
              {
                for(f=0;f<=15;f++)
                  {
                      edr=sqrt(pow((txt-coord[k*16+f].re),2)+pow((tyt-coord[k*16+f].im),2));
                      if(min>edr)
                        {
                          ele=f;
                          min=edr;
```

```
                    k1=k;
                   }
              }
          }
}
if(pp>ADAPT)
  {
        ex=(txt-coord[k1*16+ele].re);
        ey=(tyt-coord[k1*16+ele].im);
        p+=ex*ex+ey*ey;
        fdata2[2*u-1]=txt;
        fdata2[2*u]=tyt;
        u++;
  }
else
       {
         ex=txt-inkan[2*pp-1];
         ey=tyt-inkan[2*pp];
         p+=ex*ex+ey*ey;
       }
 for(i=0;i<=2*SU;i++)
       {
         cr[i]=cr[i]-KS*ex*xr[i]-KS*ey*xi[i];
         ci[i]=ci[i]-KS*ey*xr[i]+KS*ex*xi[i];
       }
 if(SU1!=0)
       {
         for(i=0;i<=2*SU1;i++)
          {
                dr[i]=dr[i]+xr1[i]*KS1*ex+xi1[i]*KS1*ey;
                di[i]=di[i]+xr1[i]*KS1*ey-xi1[i]*KS1*ex;
          }
       }
 for(i=2*SU;i>=0;i--)
       {
         xr[i+1]=xr[i];
         xi[i+1]=xi[i];
       }
 if(SU1!=0 && pp>SU)
       {
        for(i=2*SU1;i>=0;i--)
         {
           xr1[i+1]=xr1[i];
           xi1[i+1]=xi1[i];
         }
         if(xt==0 && yt==0)
          {
           xr1[0]=0.0;
           xi1[0]=1.0;
          }
```

```
                else
                  {
                   xr1[0]=coord[k1*16+ele].re;
                   xi1[0]=coord[k1*16+ele].im;
                  }
              }
          if(pp>=SU)
            {
                if(lm<10)
                  lm++;
                else
                  {
                   jk++;
                   xl=lm;
                   hata[jk]=10*log10(p/xl);
                   p=0.0;
                   lm=1;
                  }
              }
        }                          /* pp döngüsü sonu*/

clrscr(); gotoxy(30,5);
sound(500); delay(600);nosound();
fptr4=fopen("dinph.dat","wb");
fwrite(&fdata1[1],sizeof(fdata1[1]),2*(ZA-ADAPT),fptr4);
fclose(fptr4);
printf("\n Dengeleyici giriş dizisi bir dosyaya kaydedildi");
fptr5=fopen("doutph.dat","wb");
fwrite(&fdata2[1],sizeof(fdata2[1]),2*(ZA-ADAPT),fptr5);
fclose(fptr5);
printf("\n Dengeleyici çıkış dizisi bir dosyaya kaydedildi");
fptr6=fopen("error.dat","wb");
fwrite(&hata[1],sizeof(hata[1]),(ZA/10)-1,fptr6);
fclose(fptr6);
printf("\n Hata değerleri bir dosyaya yazıldı.");
for(n=0;n<8;n++)  nmetrik[0][n]=0.0;
delay(2600);
fbeep();
printf("\nViterbi kod çözme prosedürü başladı.");
for(pp=0;pp<=((ZA-ADAPT)/10)-1;pp++)
 {
  ee=0.0;
  for(i=1;i<=10;i++)
       {
        for(k=0;k<8;k++)
          {
              min=1500.0;
              for(j=0;j<16;j++)
                {
```

```
                edr=sqrt(pow((fdata2[2*(pp*10+i)-1]-
coord[k*16+j].re),2)+pow((fdata2[2*(pp*10+i)]-coord[k*16+j].im),2));
                if(min>edr)
                 {
                      eleman=j;
                      min=edr;
                 }
                }
               gm[i][k]=min;
               ge[i][k]=eleman;
            }
          for(s=0;s<8;s++)
            {
                min=1500.0;
                for(os=0;os<8;os++)
                  {
                   if(pdgm[os][s]!=0)
                        {
                          k=pdgm[os][s]-1;
                          if(min>gm[i][k])
                            {
                                min=gm[i][k];

                                eleman=k*16+ge[i][k];
                                pgk=os;
                            }
                         }
                  }
                neleman[i][s]=eleman;
                nsurvivor[i][s]=pgk;
                nmetrik[i][s]=nmetrik[i-1][pgk]+min;
             }
          }          /*Karar verme prosedürü başlangıcı*/
    min=1500.0;
    for(s=0;s<8;s++)
          {
           if(min>nmetrik[10][s])
             {
                   k=s;
                   min=nmetrik[10][s];
             }
          }
    for(i=10;i>0;i--)
          {
           os=nsurvivor[i][k];
           indis1[i]=neleman[i][k];
           k=os;
          }
    n=0;
    clrscr(); gotoxy(30,5);
```

```
 for(i=1;i<11;i++)
       {
        printf("\n\n (indis,indis1)------(%d,%d)",indis[(ADAPT-SU)+(pp*10+i)],indis1[i]);
        fdata1[2*(pp*10+i)-1]=coord[indis1[i]].re;
        fdata1[2*(pp*10+i)]=coord[indis1[i]].im;
        if(indis[(ADAPT-SU)+(pp*10+i)]!=indis1[i] && n==0)
         {
          ee++;
          sound(1700);
          delay(200);
          nosound();
          n=1;
         }
       }
  te+=ee;
 }                          /*pp çevrimi sonu*/
fptr7=fopen("voutph.dat","wb");
fwrite(&fdata1[1],sizeof(fdata1[1]),2*(ZA-ADAPT),fptr7);
fbeep();
printf("\n Viterbi kod çözücü çıkışı bir dosyaya kaydedildi");
fclose(fptr7);
delay(1700); clrscr();
gotoxy(30,5);
oh=((float) te)/((float) ADAPT/10.0);
if(oh==0)
  printf("\n\nHatalı blok yok.....!");
else
 {
  hb=10*log10(oh);
  printf("\n\nTOPLAM ADIM SAYISI----------: %d",ADAPT/10);
  printf("\n\nHATALI ADIM SAYISI----------: %d",te);
  printf("\n\nORTALAMA HATA----------: %f",oh);
  printf("\n\nHATA BAŞARIMI (dB)----------: %f",hb);
 }
getch();
fbeep();
printf("\n****OPTİMUM FİLTRE KATSAYILARI****");
for(i=0;i<=2*SU1;i++)
 {
  if(i>9)
       printf("\ndr[%d]=%10.7f        di[%d]=%10.7f",i,dr[i],i,di[i]);
  else
       printf("\ndr[%d]=%10.7f        di[%d]=%10.7f",i,dr[i],i,di[i]);
 }
getch();
clrscr();
for(i=0;i<=2*SU;i++)
 {
  if(i>9)
   printf("\ncr[%d]=%10.7f        ci[%d]=%10.7f",i,cr[i],i,ci[i]);
```

```
   else
     printf("\ncr[%d]=%10.7f          ci[%d]=%10.7f",i,cr[i],i,ci[i]);
  }
 return(0);
}

 float *fvector(nl,nh)
 int nl,nh;
 {
  float *v;
  v=(float *)malloc((unsigned) (nh-nl+1)*sizeof(float));
  return(v-nl);
 }

fbeep()
 {
  clrscr();
  gotoxy(30,5);
  sound(500); delay(200);
  nosound();
 }



 /* FFT veya IFFT hesaplar */
 four(data1,nn,isign)
 float data1[];
 int nn,isign;
 {
  int n,mmax,m,j,istep,i;
  double wtemp,wr,wpr,wpi,wi,theta;
  float tempr,tempi;
  n=nn<<1;
  j=1;
  for(i=1;i<n;i+=2)
    {
         if(j>i)
           {
            SWAP(data1[j],data1[i]);
            SWAP(data1[j+1],data1[i+1]);
           }
         m=n>>1;
         while(m>=2 && j>m)
           {
            j-=m;
            m>>=1;
           }
         j+=m;
    }
  mmax=2;
```

```
while(n>mmax)
 {
      istep=2*mmax;
      theta=6.283118530717959/(isign*mmax);
      wtemp=sin(0.5*theta);
      wpr=-2.0*wtemp*wtemp;
      wpi=sin(theta);
      wr=1.0;
      wi=0.0;
      for(m=1;m<mmax;m+=2)
       {
        for(i=m;i<=n;i+=istep)
         {
              j=i+mmax;
              tempr=wr*data1[j]-wi*data1[j+1];
              tempi=wr*data1[j+1]+wi*data1[j];
              data1[j]=data1[i]-tempr;
              data1[j+1]=data1[i+1]-tempi;
              data1[i]+=tempr;
              data1[i+1]+=tempi;
         }
        wr=(wtemp=wr)*wpr-wi*wpi+wr;
        wi=wi*wpr+wtemp*wpi+wi;
       }

      mmax=istep;
 }
 return(0);
}
```

EK G

Yardımcı Program Listeleri

1. Dengeleyici Çıkışındaki Ortalama Karesel Hatanın İterasyon Sayısı İle Değişimini Ekrana Yazdıran Program Listesi

```
/*errgraph.c*/
# include<stdio.h>
# include<graphics.h>
# include<conio.h>
# include<stdlib.h>
# include<dos.h>
# include<math.h>
# define N 99
main(argc,argv)
int argc;
char *argv[];
{
 float hata[N+1];
 int x,y,originx,originy,delta,deltay,deltax,cpx,cpy,s,k;
 FILE *fptr;
 int n=1;
 float hmax;
 int graphdriver=DETECT,graphmode,grapherror;
 if(argc!=2)
  {
    printf("\n Örnek kullanım : C>errgraph  <dosya ismi>");
    exit(1);
  }
 fptr=fopen(argv[1],"rb");
 while(fread(&hata[n],sizeof(hata[n]),1,fptr)==1)
   n++;
 fclose(fptr);
 hmax=0.0;
 for(s=1;s<=N;s++)
   {
    if(fabs(hata[s])>hmax)
          {
           hmax=fabs(hata[s]);
          }
   }
 if(graphdriver<0)
  {
   printf("\n Grafik donanımı bulunamadı!.");
   exit(1);
  }
 clrscr();
 initgraph(&graphdriver,&graphmode,"");
 grapherror=graphresult();
 if(grapherror<0)
  {
   printf("\n In▒▒t graph error :%s",grapherrormsg(grapherror));
   exit(1);
  }
 getch();
 cleardevice();
```

```
 x=getmaxx()+1; y=getmaxy()+1;
 k=10;
 rectangle(3*k,2*k,x-k,y-4*k);
 if(hmax<=80)
  {
   deltay=(y-6*k)/10;
  }
 if(hmax>80 && hmax<=180)
  {
   deltay=(y-6*k)/20;
  }
 if(hmax>180)
  {
   printf("\ EKRANA DÖKÜLEMEZ...!");
   exit(1);
  }
 deltax=(x-4*k)/5;
 setlinestyle(1,0,1);
 for(s=1;s<=((y-6*k)/deltay)-1;s++)
  {
   line(3*k,(2*k)+(s*deltay),x-k,(2*k)+(s*deltay));
  }
 for(s=1;s<=4;s++)
  {
   line((3*k)+s*deltax,2*k,(3*k)+s*deltax,y-(4*k));
  }
 setlinestyle(0,0,1);
 outtextxy(0,k-2,"Hb(x10 dB)");
 originx=3*k; originy=(2*k)+2*deltay;
 outtextxy(0,originy,"0.0");
 outtextxy(x-8*k,y-3*k,"iter(x200)");
 delta=deltax/20;
 moveto(originx,originy);
 for(s=1;s<=N;s++)
  {
    cpx=originx+s*delta;
    cpy=originy-fonk1(hata[s]/10.0,deltay);
    lineto(cpx,cpy);
  }
 getch();
 closegraph();
}

fonk1(c,delt)
float c; int delt;
{
 float b; int a,cp;
 a=floor(c);
 b=(c-(float)a)/(1.0/delt);
 if(fabs(b-floor(b))>=0.5)
```

```
  {
        if((b-floor(b))<0)
          {
           cp=a*delt+floor(b)-1;
          }
        else
          {
           cp=a*delt+floor(b)+1;
          }
  }
else
   cp=a*delt+floor(b);
return (cp);
}
```

2. İşaret Elemanlarının İşaret Uzayı İçerisindeki Durumlarını Ekrana Yazdıran Program Listesi

```
/*iugraph.c*/
# include<stdio.h>
# include<graphics.h>
# include<conio.h>
# include<stdlib.h>
# include<dos.h>
# include<math.h>
# define N 900
main(argc,argv)
int argc;
char *argv[];
{
 float data[2*N+1];
 int x,y,originx,originy,delta,cpx,cpy,s;
 FILE *fptr; int n=1;
 int graphdriver=DETECT,graphmode,grapherror;
 if(graphdriver<0)
  {
   printf("\n Grafik donanımı bulunamadı!.");
   exit(1);
  }
 if(argc!=2)
  {
   printf("\n Örnek Kullanım : C>iugraph  <Dosya ismi>");
   exit(1);
  }
 clrscr();
 initgraph(&graphdriver,&graphmode,"");
 grapherror=graphresult();
 if(grapherror<0)
  {
   printf("\n Grafik başlatma hatası :%s",grapherrormsg(grapherror));
   exit(1);
  }
 fptr=fopen(argv[1],"rb");
 while(fread(&data[n],sizeof(data[n]),1,fptr)==1)
   n++;
 fclose(fptr);
 getch();
 x=getmaxx()+1; y=getmaxy()+1;
 originx=x/2-1; originy=y/2-1;
 delta=(y/2)/12;
 for(s=1;s<=(n-1)/2;s++)
  {
   cpx=fonk1(data[2*s-1],delta)+originx;
   cpy=originy-fonk1(data[2*s],delta);
   putpixel(cpx,cpy,15);
  }

 getch();
```

```
 closegraph();
 return(0);
}
fonk1(c,delt)
float c; int delt;
{
 float b; int a,cp;
 a=floor(c);
 b=(c-(float)a)/(1.0/delt);
 if(fabs(b-floor(b))>=0.5)
   {
        if((b-floor(b))<0)
          {
           cp=a*delt+floor(b)-1;
          }
        else
          {
           cp=a*delt+floor(b)+1;
          }
   }
  else
     cp=a*delt+floor(b);
  return (cp);
 }
```

EK H

Benzetim Programlarının Kullandığı Dosyalar

kanal1.dat , kanal2.dat	Bozucu Kanal impuls yanıtlarının, FFT işleminden geçirilmiş değerlerini içeren dosyalar
işuzay.dat	İşaret uzayı içerisindeki işaret elemanlarının koordinatlarını içeren dosya
pdgm.rec	Paralel durum geçiş matrisini içeren dosya
error.dat	Dengeleyici çıkışındaki ortalama karesel hata değerlerinin içeren dosya
doutph.dat	Dengeleyici çıkışındaki işaret elemanlarının işaret uzayı içerisindeki koordinatlarını içeren dosya
dinph.dat	Dengeleyici girişindeki işaret elemanlarının işaret uzayı içerisindeki koordinatlarını içeren dosya
voutph.dat	Viterbi kod çözücüsü çıkışındaki işaret elemanlarının koordinatlarını içeren dosya

ÖZGEÇMİŞ

1968 yılında İzmir'de doğan Hakan Gökdan, ilk ve orta öğrenimini bu ilde tamamladı. 1985 yılında Yıldız Üniversitesi, Elektronik ve Haberleşme bölümüne girdi. 1989 yılında bu bölümden mezun olduktan sonra aynı yıl Yıldız üniversitesi Fen Bilimleri Enstitüsü, Elektronik ve Haberleşme anabilim dalı, Telekomünikasyon bilim dalında yüksek lisans eğitimine başladı. Halen özel bir şirkette çalışmaktadır.