KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

TURBO KODLAR VE TURBO DENKLEŞTİRİCİLER

YÜKSEK LİSANS TEZİ

Elk. Elektronik Müh. Emin TUĞCU

TEMMUZ 2007
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

TURBO KODLAR VE TURBO DENKLEŞTİRİCİLER

Elektrik Elektronik Müh. Emin TUĞCU

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"Elektronik Yüksek Mühendisi"
Ünvanı Verilmesi İçin kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 08.06.2007
Tezin Savunma Tarihi : 04.07.2007

Tez Danışmanı : Yrd. Doç. Dr. İsmail KAYA
Jüri Üyesi : Prof. Dr. Vasıf V. NABİYEV
Jüri Üyesi : Yrd. Doç. Dr. Ali GANGAL

Enstitü Müdürü : Prof. Dr. Emin Zeki BAŞKENT

Trabzon 2007

ÖNSÖZ

Bu çalışma, Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalı, Elektronik Mühendisliği Yüksek Lisans Programı'nda yapılan bir çalışmadır. "Turbo Kodlar ve Turbo Denkleştiriciler" konulu bu çalışmada, ilk olarak turbo kodların AWGN (toplanır beyaz Gauss gürültü) ve dar band rayleigh kanallarda performansı incelenmiş, ikinci kısımda ise turbo kodlar kullanılarak gerçekleştirilen turbo denkleştiricilerin performansı incelenmiştir. Analizi yapılan sistemlerin simülasyonları gerçekleştirilmiş ve elde edilen sonuçlar sunulmuştur.

Çalışmamda danışmanlığımı üstlenerek, konu seçiminde ve çalışmaların yürütülmesi sırasında yardımlarını esirgemeyen değerli hocam Yrd. Doç. Dr. İsmail KAYA' ya teşekkür etmeyi bir borç bilirim. Ayrıca yardım ve desteklerini esirgemeyen değerli arkadaşlarım Eyüp GEDİKLİ ve Samet ÇALIŞKAN' a teşekkür ederim.

Ayrıca bugüne kadar maddi ve manevi benden desteğini hiç esirgemeyen değerli aileme çok teşekkür ederim.

Emin TUĞCU
Trabzon, 2007

İÇİNDEKİLER

Sayfa No

ÖZET

Bilgi iletiminin esas alındığı sayısal haberleşme alanında her geçen gün çok ciddi gelişmeler yaşanmaktadır. Özellikle düşük güçlerde gerçekleştirilen haberleşme sistemlerinde sistem performansının artırılması yani veri kaybının en aza indirgenmesi oldukça önemli bir noktadır.

Sayısal haberleşme sistemlerinde bilgi temel olarak ikili bitler ile gösterilmektedir. İletimi yapılacak bitler uygun modülasyon teknikleri ile modüle edilerek haberleşme kanalına verilmektedir. Genel anlamda haberleşme kanalı, işaret üzerinde kanalın ISI (semboller arası girişim) etkisinden dolayı semboller arası girişime ve toplamsal gürültünün eklenmesine yol açmaktadır. Alıcıda alınan işaret haberleşme kanalın etkilerinden dolayı değişmektedir ve alınan işaretten tekrar gönderilen bitler elde edilmeye çalışılmaktadır. Kanalın ISI etkisi ve toplanır gürültüden dolayı bit hataları meydana gelmektedir.

Kanalın yapısından kaynaklanan bit hatalarına karşın gönderilen bitleri korumak için kanal kodlama işlemi yapılmaktadır. İleri yönde hata düzelten kodlar (FEC) kullanılarak bilgiye artık bitlerin ilave edilmesiyle hatalı alınan bitler algılanmakta ve düzeltilmektedir. Kanalın ISI etkisine karşın ise denkleştiriciler kullanılmaktadır.

Bu tez çalışmasının amacı ileri yönde hata düzelten kodlar arasından turbo kodlar kullanılarak turbo denkleştiricilerin gerçeklenmesidir.

Turbo sistemler ardışıl bir yapıya sahip olduklarından performansları oldukça etkileyicidir. Turbo denkleştiricilerin performanslarını değerlendirebilmek için kanalın iletilen veri üzerindeki etkisinin (ISI+toplanır gürültü) ne ölçüde telafi edilebildiğinin incelenmesi gerekmektedir. Bu nokta dikkate alınarak elde edilen simülasyon sonuçları MFB (matched fitler bound) grafikleri elde edilerek verilmiştir. MFB grafiklerinin kullanılma sebebi belirli bir sistem için elde edilebilir en iyi performansı göstermeleridir.

Anahtar Kelimeler: Ardışıl Paralel Kod, Özyinelemeli Konvolüsyonel Kodlama, Düzensiz Serpiştirme, Özyinelemeli Kod Çözme Algoritması (MAP).

SUMMARY

Turbo Codes and Turbo Equalizers

Day by day new developments run into the information based digital communication. Particularly with the low powered communication systems improving the system performance which means keeping the data loss at the minimum level, is a very important point.

In digital communication systems information is characterized by binary bits. Bits which will be transported to the communication channel are being modulated with special modulation techniques. Generally the communication channel causes interference between symbols due to the intersymbol interference (ISI) effects and noise is being added. Signals received by the receiver is being changed due to communication channel effects and from the received signal transported bits are tried to be obtained. Bit errors occur due to the channels ISI effects and noise.

To protect the bits from the bit errors because of the channel effects channel coding is being operated. Using the forward error correction (FEC) codes residual bits are precieved and being corrected.

The aim of this thesis is to develop the turbo codes to design the turbo equalizers.

Turbo systems are iterative systems so their performance is really attractive. To evaluate the turbo equalizers performance, at what rate the channels effects on the transported data re reduced should be studied. Considering this point simulation results are presented with matched filter bound (MFB) graphics. Matched filter bound graphics are used because the best performance of a certain system can be obtained.

Key Words: Parallel Concatenation Codes, Recursive Convolutional Encoding, Nonuniform Interleaving, Iterative Decoding Algorithm (MAP).

ŞEKİLLER DİZİNİ

Sayfa No.

SEMBOLLER DİZİNİ

A/D : Analog-dijital çevirici

APP : Sonsal olasılık

ARQ : Otomatik tekrarlama isteği

AWGN : Toplanır beyaz Gauss gürültüsü

a : Bayılma katsayısı

BER : Bit-hata oranı

BPSK : İkili faz kaydırmalı anahtarlama

C : Kanal kapasitesi

c : Kodlayıcı çıkışı

c_k^i : k anında i. kodlayıcı çıkışından alınan çıkış biti

c_k^s : k anında kodlayıcı çıkışında alınan sistematik bit

c_k^p : k anında kodlayıcı çıkışında eşlik biti çifti

$c_{i,k}^p$: k anında kodlayıcının i. eşlik biti çıkışından alınan eşlik biti

d_{free} : Serbest mesafe

d_{min} : Minimum Hamming mesafesi

D/A : Dijital-analog çevirici

DEC : Kod çözücü

D : Gecikme zaman operatörü/Kaydırmalı kaydedici

D_i : i. kaydırmalı kaydedici

DSN : Derin uzay ağı

ESA : Avrupa Uzay Ajansı

ENC : Kodlayıcı

E_b : Bit başına düşen enerji

E_b/N_0 : Bit başına düşen enerjinin gürültü gücüne oranı

E_s : Sembol başına düşen enerji

FEC : İleri yön hata düzeltme

FIR : Sonlu vuruş tepkesi (filtresi)

G : Üreteç matrisi

$g_i^{(j)}$: Üreteç dizisi

G : Üreteç çokterimlisi

h_i : kanalın i. tabı

IIR : Sonsuz vuruş tepkesi (filtresi)

K : Sınır uzunluğu

k : Kodlayıcıya giren veri sayısı

LLR : Logaritmik olasılık oranı

L : Kanalın tap sayısı

$L(u_k)$: k anında u bilgi bitine ait logaritmik olasılık oranı

$L_a(u_k)$: Önsel bilgi

$L_e(u_k)$: Geçici bilgi

$L(u_k \mid \underline{y})$: Sonsal bilgi

L_c : Kanal güvenirlilik katsayısı

MFB : Matched fitler bound

MAP : En büyük sonsal (algoritması)

m : Kodlayıcıdaki kaydırmalı kaydedici sayısı

NSC : Özyinelemesiz sistematik olmayan konvolüsyonel (kodlayıcı)

NASA : Amerikan Uzay ve Havacılık Ajansı

N_0 : Tek taraflı gürültü gücü spektrumu

n : Kodlayıcıdan çıkan veri sayısı

N : Giriş dizisinin uzunluğu (blok uzunluğu)

P_e : Bit hata olasılığı

$P_k(s|s')$: Durum geçiş olasılığı

$P_k(x_k|\{s,s'\})$: Kodlayıcı çıkışı olasılığı

$P(r_k|x_k)$: Kanalın geçiş olasılığı

q : Blok kodun derecesi

R : Kod oranı

RM : Reed-Muller kod

RS : Reed-Solomon kod

RSC : Özyinelemeli sistematik konvolüsyonel (kodlayıcı)

r_i : i anında geniş band kanal çıkışında üzerine gaussian gürültü bileşeni eklenmiş veri

SISO : Yumuşak-giriş yumuşak-çıkış

SNR : İşaret/gürültü oranı

SOVA : Yumuşak çıkışlı Viterbi algoritması

S_k : Kodlayıcının k anında içinde bulunduğu durum

S_i : Kodlayıcının içinde bulunduğu i. durum

S_r : S-serpiştirici faktörü

t_e : Hatalı bit sayısı

u : Kodlayıcı girişi

u_k : t anında D_1 kaydırmalı kaydedicisinin girişi

v_i : i anında geniş band kanal çıkışındaki veri

w_i : i anındaki gaussian gürültü bileşeni

x_{kl} : k anında kanala giren kod sözcüğüne ait l. bit

y_1^N : Kanalın çıkışında alınan, gürültü ilave edilmiş veri biti dizisi

y_k : k anında kanalın çıkışında alınan gürültü ilave edilmiş veri dizisi

$y_{k,1}^j$: k anında kanalın çıkışında alınan gürültü ilave edilmiş j. kodlayıcı çıkışından gönderilen eşlik biti

y_{kl} : k anında kanal çıkışından alınan kod sözcüğüne ait l. veri

z : Tam sayı değerli tasarım parametresi

σ^2 : Gürültü varyansı

α, β, γ : Logaritmik olasılık oranını oluşturan bileşenler

1. GENEL BİLGİLER

1.1. Giriş

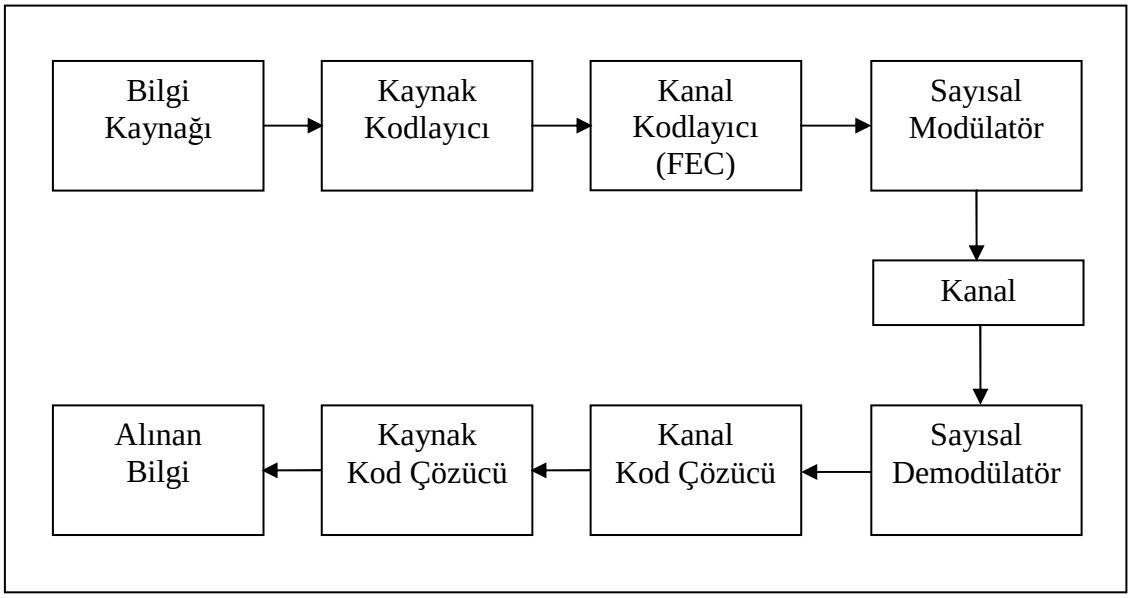
Temel olarak bir haberleşme sisteminin amacı, bilgiyi bir yerden başka bir yere iletmektir. İletim sırasında oluşan kaybın en aza indirilmesi gerekir. Sistemin performansı, iletim sırasında meydana gelen bilgi kaybı miktarı ile ölçülür. Sayısal işaret işleme tekniklerinin üstünlüklerinden yararlanmak için sayısal veri iletimine ihtiyaç vardır.

Şekil 1.1' de genel bir sayısal iletişim sisteminin blok diyagramı görülmektedir. Bu sistem üç ana bölümden oluşmaktadır: verici (kaynak), alıcı (varış yeri) ve haberleşme kanalı. Haberleşme kanalı, bükülü tel çifti, dalga kılavuzu ve fiber optik kablo gibi kablolu bir ortam olabileceği gibi hava, boşluk deniz suyu gibi kablosuz bir ortam da olabilmektedir[6].

Tipik bir sayısal iletişim sisteminde kaynak bilgi analog veya sayısal olabilir. Sayısal haberleşme sistemlerinde kaynak tarafından üretilen bilgi sayısal darbelere dönüştürülmelidir. Analog bilgiye örnek olarak ses, video, resim bilgisi ya da müzik; sayısal darbelere örnek olarak ise ikili kodlu sayılar, alfa sayısal kodlar, grafik semboller verilebilir.

Verimli bir sayısal iletişim sisteminin sağlanması için veri sıkıştırması yapılarak iletilmesi gereken veri miktarının azaltılması gerekir. Bu yüzden kaynak kodlayıcı kullanılır. Örneğin, bir video sinyalinin A/D çeviriciden geçirilerek MPEG kaynak kodlayıcıya sürülmesiyle verinin sıkıştırılması sağlanmaktadır.

Kaynak kodlaması ile sıkıştırılan verinin haberleşme kanalının istenmeyen etkilerine karşı dayanıklı hale getirmek için kanal kodlama işlemi yapılır. Kanal kodlama işlemiyle veriye artık bitler ilave edilir. Sayısal veri iletişiminde kullanılan kanal kodlama teknikleri ileri yönde hata düzeltme (FEC) ve otomatik tekrarlama isteği (ARQ) olmak üzere iki sınıfa ayrılır. ARQ, veri çerçeveleri ile işlem görür. Bir veri çerçevesi enformasyon bitlerinden ve az sayıda hata sezme bitlerinden oluşur. ARQ kanal kodlama tekniğinde, hata algılandığı zaman ters kanal üzerinden veri çerçevesi geri gönderilerek, veri çerçevesinin yeniden gönderilmesi istenir.



Şekil 1.1. Sayısal iletişim sisteminin blok diyagramı

Veri çerçevesi doğru alınana kadar bu böyle devam etmektedir. Fakat bu kanal kodlama tekniği, iki yönlü haberleşme söz konusu olduğu zaman kullanılabilmektedir. Bu nedenle, çoğu sayısal iletişim sisteminde ARQ kullanımı, gereksiz işlemleri ve bant genişliğini azaltmasına rağmen maliyeti artırıcı bir etken olmaktadır. Dolayısıyla, çoğu kez ARQ yerine FEC tercih edilmektedir. FEC, gönderilecek bilgiye artık bitler eklenerek gerçekleştirilir. Bu artık bitler hataların algılanıp düzeltilmesinde kullanılır. Ayrıca, başarımı artırmak için FEC ve ARQ teknikleri birleştirilerek Karma ARQ sistemleri oluşturulur. Karma ARQ sistemlerinde FEC alt sistemi, kod kelimesindeki hataları düzelterek yeniden gönderme sayısını azaltırken, ARQ alt sistemi ile hatalı kod kelimelerinin kullanıcıya iletilmesi önlenir. Böylece karma ARQ sistemlerinde yüksek sistem başarımı ve güvenirliliği sağlanmış olur.

Dikkat edilirse, kaynak kodlama işlemi ile kanal kodlama işlemi arasında bir zıtlık olduğu görülmektedir. Kaynak kodlama işlemi ile kaynak bilgi sıkıştırılarak veri miktarı azaltılırken, kanal kodlama işlemi ile kaynak bilgiye artık bitler eklenmektedir. Bundan dolayı kaynak kodlama işlemi kanal kodlama işleminin tam tersi bir işlem olarak düşünülebilmektedir. Dataya ilave edilen bu artık bitler kullanılarak alıcıda hatalı alınan bitler düzeltilir.

Bilgi dizisinin fiziksel kanal üzerinden iletime uygun olmamasından dolayı sayısal bir modülatör yardımı ile kanal kodlayıcı çıkışındaki veri modüle edilip sürekli zaman dalga şekline çevrildikten sonra fiziksel kanal üzerinden iletilmektedir. Şekil 1.1'de verilen

blok diyagramda genellikle iki tür fiziksel kanal kullanılmaktadır: kablolu ve kablosuz. Haberleşme kanalının etkilerinden dolayı alıcıda alınan bilgide kayıplar oluşur.

Alıcının girişinde bozulmuş işaret alınmaktadır. Alınan bu işaret sayısal detektör yardımı ile demodüle edilerek kodlanmış bitlerin tahmini yapılmaktadır. Kanal kod çözücü, verici tarafında kanal kodlayıcının bilgi bitlerine eklediği artık bitleri kullanarak, kanal etkisinden dolayı oluşan hataların algılanıp düzeltilmesini sağlamaktadır.

1.2. Hata Düzeltme Kodlaması

Herhangi bir iletişim kanalı üzerinden bilgi iletmeye çalışılırken esas amaçlardan biri güvenirliği sağlamaktır ve alıcıdaki doğru algılama oranı ile ölçülmektedir. Bu amaçlar doğrultusunda, iletişim sistemlerinde güvenirliliği sağlamak ve alıcıdaki doğru algılama oranını artırmak için hata düzeltme kodlamasından yararlanılmaktadır. Bilgi teorisinin ve hata düzeltme kodlamasının temelini, bilişim kuralının en temel sonuçlarından biri olan Shannon' un 1948 yılında ortaya koyduğu "kanal kapasite teoremi" oluşturmaktadır[3],[7]. Bu teoremde, kod oranına bağlı olarak gürültülü bir ortamda kanal kapasitesinin teorik olarak ulaşabileceği üst sınır belirtilmiştir. Bunun doğrultusunda, kodlama teorisine olan ilgi hızla artmaya başlamıştır. Günümüzde pratik olarak Shannon' un kanal kapasite sınırına ulaşabilmek için etkin kanal kodlama tekniklerini içinde barındıran sayısal iletişim sistemlerinin tasarımı üzerinde durulmaktadır.

1.2.1. Kod Oranı (Kod Hızı)

Kanal kodlayıcılarının performansını belirleyen en önemli kavramlardan biri kod oranıdır. Bir kanal kodlayıcısı, *k* bitlik veri dizisini *n* bitlik kod kelimesine dönüştürmekte *(n>k)* ve buna bağlı olarak $R=k/n$ oranı kod oranı olarak adlandırılmaktadır. n>k olmasından dolayı kod oranı birden küçüktür ($R<1$). Ayrıca, güvenilir bir haberleşme sağlamak için kod oranı kanal kapasitesini aşmamalıdır ($R \leq C$). Sıkça kullanılan kod oranları 1/4, 1/3, 1/2, 2/3 ve 3/4'tür. Fakat pratikte teorikten farklı olarak gerçekte kod oranı, ek kodlayıcı bitleri olarak nitelendirilen kuyruk bitlerinden ötürü yukarıda verilen kod oranı değerlerinden çok az da olsa küçüktür.

1.2.2. Shannon Kanal Kapasite Teoremi

Kod oranı, kanal kapasitesi olarak adlandırılan bir rakamdan daha küçük olduğu sürece, gürültülü kanallar üzerinden de güvenilir iletişim sağlamak mümkün olacaktır [7]. 1940'lı yıllarda Claude Shannon tarafından sunulan bu bulgu, gürültülü kanal kapasitesi teoremi [3] olarak bilinmektedir.

Bu teoreme göre, toplanır beyaz Gauss gürültüsünün (AWGN) getirdiği kısıtlama, iletişim güvenirliliği ile ilgili değil iletişim hızı ile bağlantılıdır. Bu teorem, hatanın oluşmadığı ya da yok olduğu bir kanal ile desteklenebilen maksimum veri oranını verir. AWGN kanal için kapasite aşağıda verilen (1.1) eşitliği ile verilir.

$$C = \frac{1}{2}\log_2\left(1 + \frac{2E_s}{N_0}\right) \tag{1.1}$$

$$E_s = E_b \times R \tag{1.2}$$

Burada C bilgi kapasitesi, E_s sembol enerjisi, N_0'da kanalın çift taraflı gürültü güç yoğunluk spektrumudur. Bu kapasiteye yakın değerlere pratikte, hata düzeltme kodlamasıyla (*error correction coding*) ulaşılabilir. Güvenilir bir haberleşme sağlamak için kod oranı, kanal kapasitesini aşmamalıdır($R \leq C$). Bu durumda gerekli minimum E_b/N_0 aşağıdaki denklemden bulunur.

$$\frac{E_b}{N_0} \geq \frac{1}{2R}\left(2^{2R} - 1\right) \tag{1.3}$$

(1.3) denkleminde eşitlik, sadece girişin Gauss dağılımlı olması durumunda geçerlidir. Shannon' un teoreminde, $R \leq C$ şartına bağlı olan bir rasgele kod için kod uzunluğu (n) sonsuza giderken, bit hata olasılığı da (P_e) sıfıra yaklaşır. Ancak pratikte rasgele kodlar gerçekleştirilemez. Ayrıca, kodlama ve kod çözme işlemlerinin verimli yapılabilmesi için kodların yapılandırılmış olması gerekir. Ancak Shannon, kanal kapasitesine pratik olarak nasıl ulaşılacağına açıklık getirmemiş, sadece elde edilebilir olduğunu göstermiştir.

1.3. Kanal Kodlama Teknikleri

Sayısal iletişim sisteminde kullanılan kanal kodlama teknikleri bilgiyi gürültü ve diğer bozucu etkenlerden korumak ve bit hata oranını düşürmek için kullanılır. Kodlama tekniğinde en büyük kaygı, güvenilir bir iletişim sağlayabilmek için hataların kontrolünün sağlanmasıdır. Kanal kodlama, genellikle gönderilecek bilgiye artık bitler eklenerek gerçekleştirilir. Bu artık bitler oluşan hataların algılanmasına ve düzeltilmesine olanak verir ve daha güvenilir bir bilgi akışı gerçekleştirmeyi sağlar. Bilgi korumak amaçlı kullanılan kanal kodlamanın kullanıcıya maliyeti, veri hızında bir azalma veya bant genişliğinde istenmeyen bir artımdır.

Kanal kodlama teknikleri başlıca dört ana başlık altında incelenebilir: blok kodlar, katlamalı kodlar, birleştirilmiş(concatenated) kodlar ve turbo kodlar.

1.3.1. Blok Kodlar

Shannon' un, güvenilir bir iletişimin sağlanabilmesi için gerekli olan teorik limitleri tanımlamasının ardından, blok kodlar ilk kez 1950 yılında Richard Hamming' in [33] bir bilgisayar simülasyonu sonucunda algılanan hatayı düzeltmeyi başarmasıyla ortaya çıkmıştır. Bu nedenle Richard Hamming, ilk hata düzeltme kodunu bulması ile tanınmakta ve bulmuş olduğu bu hata düzeltme kodu Hamming kod olarak adlandırılmaktadır. Hamming yapmış olduğu bu çalışmada, bilgi bitlerini dörder bitlik gruplar halinde gruplandırmakta ve gruplandırılmış olan bu dört biti kullanarak üç tane kontrol biti elde etmekteydi. Ardından, elde ettiği yedi bitlik kod kelimesini yazmış olduğu bir bilgisayar programına giriş olarak uygulamaktaydı. Hamming, bu bilgisayar programı vasıtasıyla, sadece kod kelimesindeki hataları algılamakla kalmayıp bir bitlik hatayı da düzeltmekteydi. Sonuç olarak Hamming kod, yedi bitlik bir kodlanmış bilgi bloğunda hatalı olan bir tane biti düzeltebilmekteydi.

Hamming kodlar kodlama alanında büyük ilerlemeler sağlamasına rağmen istenmeyen bir takım özelliklere de sahipti. Bunlardan ilki; Hamming kodlar, iletilecek olan dört tane bit için üç tane kontrol bitine ihtiyaç duymaktaydı. İkincisi ise, Hamming kodların bir blok içinde sadece bir tane hatalı biti düzeltebilme yeteneğine sahip olmasıydı. Bu gibi istenmeyen özelliklerin getirmiş olduğu olumsuzluklar, Marcel Golay tarafından Hamming kodların geliştirilmesiyle elde edilen Golay kodlar ile biraz olsun aşılmıştır.

Golay kodlar, 12'si bilgi biti, 11'i kontrol biti olmak üzere bir blokta 23 tane biti iletirken, her bir blokta üç tane hatalı biti de düzeltebilme yeteneğine sahiptir [34].

Hamming ve Golay kodların yapısı aynıdır. Her ikisinde de, k tane q. dereceden semboller gruplandırılarak, her bir gruba n-k tane kontrol bitinin eklenmesi ile n sembollük kod kelimesi oluşturulmaktadır. Oluşturulan bu kod kelimesi ile t_e adet hatalı sembol düzeltilebilmektedir. Genel olarak, bu yapıya sahip olan kodlar blok kod olarak tanımlanmakta ve kısaca (q,n,k,t_e) şeklinde ifade edilmektedir. Ayrıca, iki kod kelimesinin birleşmesiyle yeni bir kod kelimesi oluşmasından dolayı Hamming ve Golay kodlar lineerdir. Blok kodların keşfinden bu yana, yaklaşık 55 yıl içerisinde, blok kod sınıfına ait birçok hata düzeltici geliştirilmiş ve birçok uygulama alanında kullanılmıştır. Örneğin, Jüpiter'e gönderilen Voyager I' de ikili Golay kodları hata düzeltme için kullanılmıştır. Buna rağmen, günümüzde Golay kodlar birçok uygulama alanında yerini daha güçlü kodlara bırakmıştır.

Uzun süren çalışmalardan sonra, 1954 yılında Muller tarafından tanımlanan [35] ve Reed tarafından yeni bir hata düzeltme kodu olarak tanıtılan, Reed-Muller (RM) lineer blok kodları keşfedilmiştir. RM kodları, kod kelimesi uzunluğunda ve kod kelimesi başına düzeltilebilir hata sayısında daha ileri bir esnekliğe izin vermesinden dolayı Hamming ve Golay kodlara oranla daha ileri bir kodlama tekniğidir. RM kodları (q=2, n=32, k=2, t_e=7), özellikle 1969 ve 1977 yılları arasında Mars'ta Mariner görevinde büyük bir uygulama alanı bulmuştur. Yalnız, Mariner görevinin sona ermesiyle RM kodları eski popülerliğini kaybetmiştir.

RM kodlarının ardından, 1957 yılında Air Force Cambrige Araştırma Merkezi çalışanlarından biri olan Prange tarafından Cyclic kodlar keşfedilmiştir [36]. Cyclic kodlar lineer blok kod ailesinin bir alt kümesi olup, buna ek olarak Cyclic kodlarda kodlanmış dizinin mantıksal kaydırılmış hali yeni bir kodlanmış bilgiyi oluşturmaktadır. Yapılarından ötürü Cylic kodlarda kullanılan kodlayıcının ve kod çözücünün karmaşıklığı azdır. Cyclic kodlarda kullanılan kod çözücü Meggit kod çözücüsü olup [37], bu kod çözücünün karmaşıklığı düzeltilebilen hata sayısına bağlı olarak üssel olarak artmakta ve genellikle bir veya ardarda gelen iki hatayı düzeltmek için kullanılmaktadır. Bu nedenle, Cyclic kodlar günümüzde hata düzeltici olarak değil de hata algılayıcı olarak kullanılmaktadır.

1959 yılında Hocquenghem ve 1960 yılında Bose ve Ray-Chaudhuri grubunun eş zamanlı olarak yapmış oldukları çalışmalar sonucunda Cyclic kodların alt sınıfı olan BCH kodları keşfedilmiştir. BCH kodlarının uzunluğu $n=q^z-1$ olarak hesaplanırken, burada z

tamsayı değerli tasarım parametresidir. Ayrıca, (q=2) için ikili BCH kodların düzeltebileceği hata miktarı $t_e<(2^z-1)/2$ olarak belirtilmiştir. 1960 yılında ise Reed ve Solomon [38] tarafından BCH kodları geliştirilerek Reed-Solomon kodları ortaya çıkmıştır. Reed-Solomon (RS) kodlar, BCH kodların bir alt kümesi olup ikili olmayan veri dizileri üzerinde işlem yapmaktadır. Bu özelliği sayesinde, gruplar halinde hataların gelmesi durumunda oldukça iyi bir performans sergilemektedir. Dolayısıyla disk üzerindeki çizikler nedeni ile hataların gruplar halinde geldiği Compact Disc teknolojisinde bir standart olarak kullanılmaktadır [39].

1.3.2. Katlamalı Kodlar

Blok kodların kullanımıyla performans başarımında önemli bir artış elde edilmesine rağmen, blok kodların kendi iç yapısından kaynaklanan birkaç dezavantaja sahip olması nedeniyle, bu tür kanal kodlama tekniğinin iletişim sistemlerinin tasarımında kullanılması sıkıntı yaratmaktadır: 1- Blok kodlar çerçeve uyumlu olması nedeniyle kod çözme işlemi başlamadan önce iletilen bütün blokların alıcı tarafından alınması gerekmektedir. Bunun sonucunda, özellikle büyük blok uzunluklarında sistemde tahammül edilemeyecek bir gecikme oluşmaktadır. 2- Blok kodlar, kesin (çok iyi) bir çerçeve senkronizasyonuna gerek duymaktadır. 3- Blok kodlarda kullanılan kod çözücüler, sıfır-bir kararına dayalı olarak çalışmaktadır. Sıfır-bir kararına dayalı olarak çalışan kod çözücülerde, kanal çıkışında alınan bilgi ikili düzende (0 veya 1) olmasına rağmen, yumuşak-tahminli kod çözücülerde kanalın çıkışında alınan bilgi sürekli (reel) bir değer olacaktır. Oysa Shannon tarafından tanımlanan teorik performans sınırına ulaşabilmek için sürekli değerde kanal çıkışına ihtiyaç duyulmaktadır. Bu nedenden dolayı, genel olarak blok kodlar, iyi kanallarda etkileyici bir performans oranı yakalamasına rağmen, güç tüketiminde verimli olamadıkları için işaret/gürültü oranının düşük olduğu değerlerde kötü bir performans sergilemektedir. Yalnız unutulmamalıdır ki, blok kodların düşük işaret/gürültü oranı değerlerinde kötü bir performans sergilemesi, blok kodların kendi iç yapılarından dolayı değil de daha çok bu kodların kod çözme işleminde kullanmış olduğu sıfır-bir kararına dayalı olarak çalışan kod çözücülerden kaynaklanmaktadır. Gerçekte, blok kodlarda yumuşak-tahminli kod çözücüler kullanma olası bir ihtimal olmasına rağmen, işlem karmaşıklığının artacağı düşünülmesinden dolayı tercih edilmemektedir.

Blok kodların dezavantajlarından, kodlama için farklı bir yaklaşım olan ve ilk kez 1955'te Elias tarafından ileri sürülen katlamalı kodların ele alınması ile kaçınılmıştır [40]. Katlamalı kodlar günümüzde kullanımı yaygınlaşmış güçlü kodlar olup giriş bilgisi bloklar halinde gruplanmayıp, bir kaç tane giriş bilgisi üzerinden koda özgü fonksiyonlar kullanarak katlanması ile kodlanmaktadır. Bir başka değişle katlamalı kodlar, veri bitlerinin sonlu duruma sahip lineer kaydırmalı kaydedicilerden geçirilerek iletilmesi ile oluşmaktadır. Bu özelliği sayesinde katlamalı kodlarda hem blok kodlarda olduğu gibi veri bloğunun hazır olup olmaması gibi bir sorunla karşılaşılmamakta hem de blok kodlara oranla katlamalı kodlarda kodlama işlemi daha kısa bir süre içerisinde gerçekleşmektedir. Genel olarak katlamalı kodlayıcının çıkışındaki n adet çıkış biti, k adet giriş bitinin, kaydırmalı kaydediciler içerisinde saklanan m adet bit ile lineer kombinasyonu sonucunda elde edilmektedir. Bu durumda kodlayıcı hızı $R=k/n$ olmaktadır. Her bir çıkışın bağlı olduğu bitlerin toplam sayısı, sınır uzunluğu, K, olarak adlandırılmaktadır. Ayrıca, katlamalı kod çözücüde kullanılan algoritma tamamen demodülatör çıkışındaki yumuşak bilgiye dayalıdır.

İlk pratik kod çözücü algoritması 1961'de Wozencraft ve Reiffen' in [41] ardışık kod çözücü algoritmasıdır. Bu algoritma daha sonra 1963'te Fano [42] ve 1969'da Jelinek [43] tarafından geliştirilmiştir. Yalnız bu algoritma, kodlamada büyük bir başarı sağlamasına rağmen 1967 yılında ortaya çıkan Viterbi algoritması [12],[13] ile popülerliğini kaybetmiştir. Günümüzde Viterbi algoritması katlamalı kodlar için optimum çözümü teşkil etmektedir.

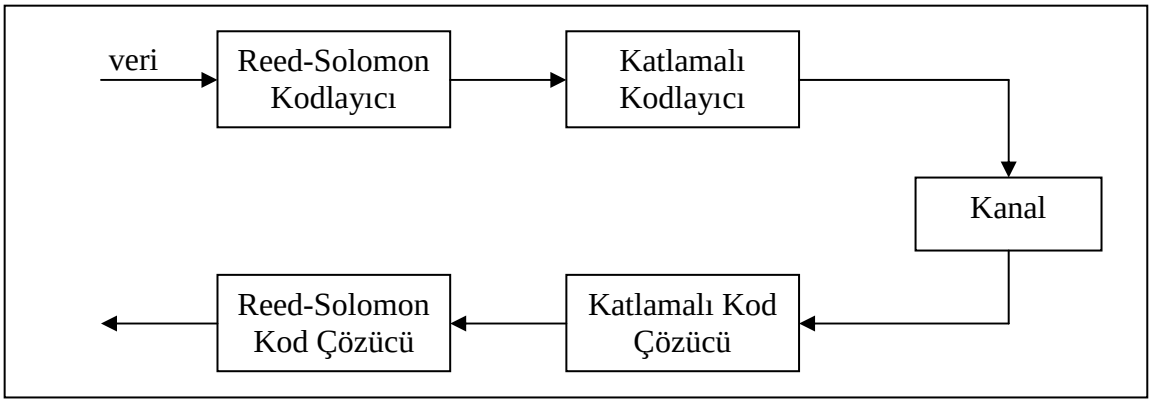
Viterbi algoritmasının keşfinden sonra katlamalı kodlayıcılar, iletişimde geniş bir uygulama alanı bulmuştur. Sınır uzunluğu K=5, kod oranı R=1/2 ve R=1/3 olan "Odenwalder" katlamalı kodu ticari uydular arası iletişim uygulamalarında bir standart olmuştur [39]. "Voyager" ve " Pioneer" gibi birkaç katlamalı kod derin uzay araştırmalarında kullanılmıştır [41]. Benzer şekilde, ikinci nesil tüm sayısal hücresel standartlar katlamalı kodları kullanmaktadır. GSM, R=1/2, K=5 katlamalı kod [3], Globalstar R=1/2, K=9 katlamalı kod, Iridium R=3/4, K=7 katlamalı kod [44] kullanmaktadır.

1.3.3. Birleştirilmiş(Concatenated) Kodlar

Katlamalı kodlar günümüzde, sıkça kullanılan bir kodlama tekniği olmasına rağmen blok kodlardaki kadar çok olmasa da önemli bir dezavantaja sahiptir. Katlamalı kodlar, hata patlaması (ardarda hatalı bitlerin gelmesi) olması durumunda istenilen düzeyde bir performans sergileyememektedir. Bu dezavantajı ortadan kaldırmak için yapılan çalışmalar sonucunda, 1966 yılında David Forney tarafından [45] ardışıl kod olarak adlandırılan yeni bir kodlama tekniği tasarlanmıştır. Ardışıl kodlar, iki veya daha fazla basit kodlayıcının yüksek kod kazancına ulaşmak için birleşmesinden ibarettir.

Katlamalı kodlar, sahip oldukları özellikler bakımından Reed-Solomon kodlara oranla daha üstün bir kodlama tekniğidir. Ancak, katlamalı kodlar hata patlaması olması durumda iyi bir performans sergileyemezken, Reed-Solomon kodlar, katlamalı kodlara oranla hata patlaması olması durumunda daha iyi bir sonuç vermektedir. Yalnız, unutulmamalıdır ki, düşük işaret/gürültü oranında, yumuşak-tahminli kod çözücü kullanan katlamalı kodlar, aynı karmaşıklığa sahip Reed-Solomon kodlara oranla daha iyi bir performansa sahiptir [13]. Bu durumları göz önünde bulundurarak Forney, katlamalı kod ve Reed-Solomon kodu ardışıl bir biçimde seri olarak kullanarak Şekil 1.2’ deki seri ardışıl kod blok şemasını oluşturmuştur.

Bu şemada veri, ilk önce Reed-Solomon (RS) kodlayıcıya verilmektedir. Ardından RS kodlayıcının çıkışı katlamalı kodlayıcının girişine uygulanmaktadır. Böylelikle veri iki kere kodlanmış olacaktır. Alıcı tarafında ise ilk önce, gürültülü kanal üzerinden iletilen veri katlamalı kod çözücü tarafından alınarak, hatalı gelen veri düzeltilmeye çalışılmaktadır. Katlamalı kod çözücünün çıkışında yükseltilmiş bir işaret/gürültü oranı elde edilmesine rağmen, katlamalı kodların yapısından ötürü, hatalı veriler gruplandırılmış olacaktır. Gruplanmış olan hatalı veriler Reed-Solomon kod çözücüden geçirilerek hatalar düzeltilmekte ve böylelikle kod çözme işleminin son aşaması da tamamlanmaktadır.



Şekil 1.2. Reed-Solomon ve katlamalı kodların bileşiminden oluşan seri ardışıl kod

Şekil 1.2' deki ardışıl kod şeması, 1987 yılında NASA ve ESA tarafından oluşturulan Derin Uzay Ağı (*Deep Space Network*-DSN) standardında kullanılmıştır.

1.3.4. Turbo Kodlar

Kodlama kuramında, keşfi en fazla yankı uyandıran kodlar turbo kodlar olmuştur [1]. Turbo kodlar, 1993 yılında Berrou, Glavieux ve Thitimajshima tarafından önerilmiştir[2]. Önerilen yapıda kodlayıcı, paralel birleştirilmiş iki RSC kodlayıcı ve bu iki kodlayıcı arasına yerleştirilmiş serpiştiriciden oluşmaktadır. Kod çözücü ise iteratif bir yapıdan oluşmakta ve RSC kodların kod çözümünde 1974 yılında Bahl tarafından geliştirilen MAP algoritması kullanılmaktadır[14]. Turbo kodların keşfine kadar MAP algoritması karmaşık yapısından dolayı kullanılmamıştır. İteratif yapılarından dolayı turbo kod çözücüde bileşen kod çözücüler arasında aktarılan bilginin etkisiyle performansları Shannon tarafından belirlenen sınıra çok yaklaşmaktadır[2]. Yapılan çalışmalar sonucunda, 10^{-5} bit hata olasılığı (*probability of bit error* – P_e) için sinyal/gürültü oranını 0,7 dB bulunmuştur. 1993 yılından günümüze kadar birçok çalışma yapılmış ve özellikle turbo sistem yapısı, sayısal haberleşme alanında değişik problemlerin çözümüne uyarlanmaya çalışılmıştır. Turbo kodların pratikte kullanımını sağlamak için kod çözücüde kullanılan MAP algoritmasının karmaşıklığı azaltılmaya çalışılmıştır. Bunun sonucunda Max-Log-MAP algoritması Koch ve Baier tarafından önerilmiştir[50]. Benzer şekilde yine kod çözücünün karmaşıklığını azaltmak için Log-MAP algoritması Robertson, Villebrun ve Hoeher tarafından önerilmiştir[47]. Kod çözücüde bileşen kod çözücüler arasındaki bilgi aktarımının log-benzerlik(log-likelihood) oranı biçiminde aktarılmasıyla kod çözücünün yapısının

basitleşeceği gösterilmiştir[46]. Serpiştirici tasarımının performans üzerindeki etkileri incelenmiştir[21],[23],[5]. Turbo kodların genel yapıların anlaşılabilmesine yönelik çalışmalar yapılmıştır[16],[25]. Log-benzerlik bölgesindeki kod çözücü algoritmaların sadece katlamalı kodlarla sınırlı olmayıp, herhangi bir ikili sistematik kodlar için kullanılabildiği gösterilmiştir[10].

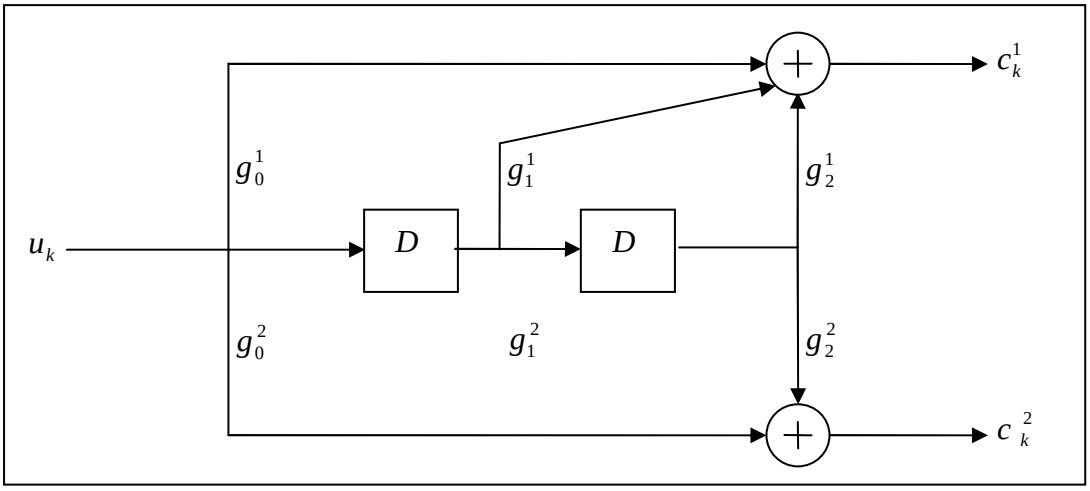
1.4. Turbo Dekleştiriciler

Turbo denkleştiriciler ilk olarak 1995 yılında Douillard tarafından önerilmiştir[4]. Önerilen sistem seri birleştirilmiş katlamalı kodlayıcı, serpiştirici ve çok yollu kanaldan oluşmuştur. Çok yollu kanalın markov zinciri[49] yapısında olduğu düşünülmüştür. Denkleştirici ve kod çözücü birleştirilerek ardışıl bir yapıyla uygulanmıştır[28],[29],[30].

2. YAPILAN ÇALIŞMALAR

2.1. Katlamalı Kodlar

Katlamalı kodlayıcıda giriş bilgisi, bir kaç tane giriş bilgisi üzerinden koda özgü fonksiyonlar kullanarak katlanması ile kodlanır. Genelde, bir (*n*, *k*, *m*) katlamalı kodlayıcı içerisinde, *k* tane giriş biti için doğrudan doğruya *n* tane çıkış biti üretilmektedir. *m* ise kodlayıcıdaki hafıza elemanı olarak adlandırılan kaydırmalı kaydedicilerin sayısıdır. Bu durumda, kod oranı ise *R=k/n* olarak tanımlanmaktadır.



Şekil 2.1. *R*=1/2 olan katlamalı kodlayıcı

Şekil 2.1' de kod oranı *R*=1/2 olan lineer bir katlamalı kodlayıcı görülmektedir. Şekilde görüldüğü gibi $u = \{u_0, u_1, ..., u_N\}$'den oluşan ikili veri dizisi, bellek elemanı olarak görev yapan iki bitlik kaydırmalı kaydedicilere uygulanmaktadır. Kodlayıcının çıkışında ise her bir giriş bitine ve koda özgü fonksiyonlara bağlı olarak üretilen $c^{(1)} = \{c_0^{(1)}, c_1^{(1)}, ..., c_N^{(1)}\}$ ve $c^{(2)} = \{c_0^{(2)}, c_1^{(2)}, ..., c_N^{(2)}\}$ olarak adlandırılan ikili kod sözcükleri elde edilmektedir.

Dikkat edilirse, örnekteki katlamalı kodlayıcıda iki tane kaydırmalı kaydedici bulunmakta ve kodlama üç bit üzerinden yapılmaktadır. Kullanılan kaydırmalı kaydedici sayısı, *m*, direkt olarak kodlayıcının düzeltebileceği hata miktarına etki etmektedir. Her bir

çıkışın bağlı olduğu bitlerin toplam sayısı, kısıt uzunluğu, *K*, olarak adlandırılmakta ve pratikte kısıt uzunluğu, toplam kaydedici sayısı artı bir olarak belirlenmektedir.

$$K = 1 + m_{\max} \tag{2.1}$$

Şekil 2.1' de gösterilen katlamalı kodun kısıt uzunluğu *K*=3 'tür.

Katlamalı kodların yapısını göstermenin en basit yolu onların üreteç dizilerini vermektir [27], [35]. Üreteç dizisi $g_i^{(j)}$ yi, kodlayıcının *i*. girişine kendisini sıfır dizisi takip eden sadece bir tane "1" biti uyguladığı zaman, kodlayıcının *j*. çıkışında elde edilen vuruş tepkesi olarak ifade etmek mümkündür. Üreteç dizilerinde, çıkışa etki eden kaydediciler "1" ve kullanılmayan kaydediciler "0" olarak nitelendirilerek devre bağlantılarının yapısı belirlenmektedir. Örneğin, Şekil 2.1' deki katlamalı kodlayıcının üreteç dizileri şunlardır:

$$g^{(1)} = \left\{g_0^1, g_1^1, g_2^1\right\} = \{1,1,1\} \tag{2.2}$$

$$g^{(2)} = \left\{g_0^2, g_1^2, g_2^2\right\} = \{1,0,1\} \tag{2.3}$$

Gecikme zaman operatörü *D* üreteç dizilerinin tanımında kullanırsa, üreteç dizilerinin çokterimli formu elde edilmiş olur.

$$\left\{g^{(1)}, g^{(2)}\right\} = \left\{1 + D + D^2, 1 + D^2\right\} \tag{2.4}$$

Kodlayıcının yapısı belirtilirken ileri yöndeki üreteç dizilerinin verilmesi yerine genellikle bu üreteç dizileri kullanılarak sekizli formda genel bir üreteç yapısı, *G*, verilmektedir. Şekil 2.1' deki kodlayıcının genel üreteç fonksiyonu şu şekilde yazılabilir.

$$G = (7,5)_8 \tag{2.5}$$

Tek girişli kodlayıcının çıkışları üreteç dizileri cinsinden

$$c_i^{(j)} = \sum_{l=0}^{m} u_{i-1} . g_l^{(j)} \tag{2.6}$$

olarak ifade edilebilir. Bu ifadeden de görüldüğü gibi kodlayıcının çıkışı c, girişi u ve üreteç dizisi (vuruş tepkesi) *g*' nin ayrık katlamasıdır. Bu nedenle, bu tür kodlar "katlamalı (katlamalı) kodlar" olarak adlandırılmaktadır. Bu durum aşağıdaki gibi ifade edilebilir.

$$c^{(j)} = u * g^{(j)} \tag{2.7}$$

Denklem (2.6)' yı, *k* tane girişe sahip katlamalı kod için genelleştirilerek şu şekilde ifade etmek mümkündür:

$$c_i^{(j)} = \sum_{i=0}^{k-1} \left(\sum_{l=0}^{m} u_{i-l}^{(j)} . g_l^{(j)} \right) \tag{2.8}$$

Katlamalı kodlarda blok kodlardan farklı olarak, herhangi iki kod kelimesi arasındaki minimum Hamming mesafesi serbest mesafe (*free distance-*d_{free}) olarak tanımlanmaktadır. Benzer olarak d_{free}, tüm bitleri sıfır olmayan kod kelimeleri arasındaki minimum Hamming ağırlığı olarak da ifade edilebilir [36].

Üç çeşit katlamalı kodlayıcı bulunmaktadır: Sistematik katlamalı kodlayıcı, özyinelemesiz sistematik olmayan katlamalı kodlayıcı (NSC), özyinelemeli sistematik katlamalı kodlayıcı (RSC). Turbo kodlarda bileşen kodlayıcı olarak RSC kodlayıcı kullanılmaktadır.

2.1.1. Özyinelemesiz Sistematik Olmayan Katlamalı Kodlayıcı (NSC)

Şekil 2.2' de özyinelemesiz sistematik olmayan katlamalı (NSC) kodlayıcının yapısı gösterilmektedir. Kodlayıcı yapısındaki $\oplus$ sembolü, ikili tabanda toplamayı temsil etmektedir.

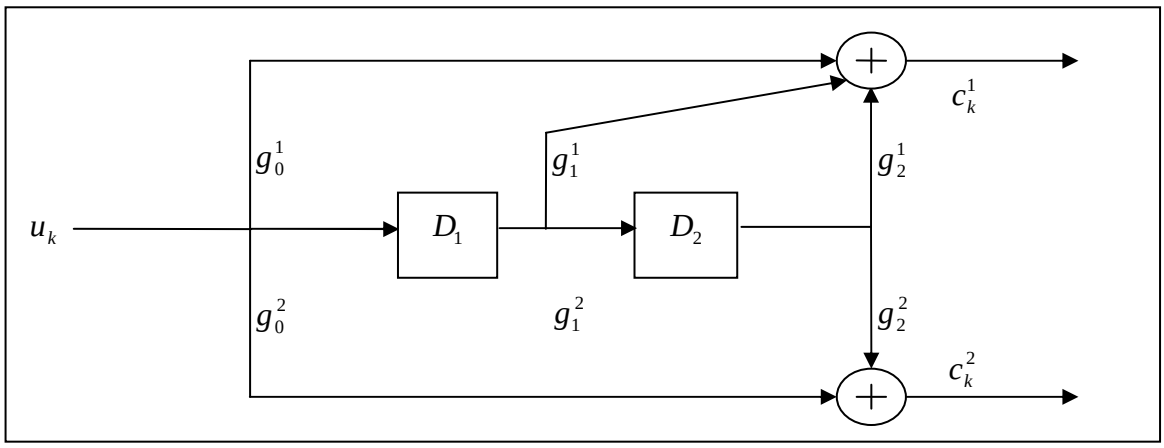
Daha önce belirtildiği gibi, bir kodlayıcının yapısı kendi üreteç dizileri ile ifade edilmektedir. Şekil 1.2' deki NSC kodlayıcının üreteç dizileri şunlardır:

$$\left.\begin{aligned} g^{(1)} &= \{g_0^1, g_1^1, g_2^1\} = \{1,1,1\} \\ g^{(2)} &= \{g_0^2, g_1^2, g_2^2\} = \{1,0,1\} \end{aligned}\right\} \Rightarrow G = (7,5)_8 \tag{2.9}$$

Benzer şekilde, k anındaki çıkış bitleri $\left(c_k^1, c_k^2\right)$ ile *k* anındaki giriş biti u_k arasındaki bağıntı şu şekilde ifade edilebilir:

$$c_k^1 = \sum_i^m g_i^1 . u_{k-1} \tag{2.10}$$

$$c_k^2 = \sum_i^m g_i^2 . u_{k-1} \tag{2.11}$$



Şekil 2.2. *R=1/2, K=3, G=(7,5)*$_8$ özyinelemesiz sistematik olmayan katlamalı kodlayıcı

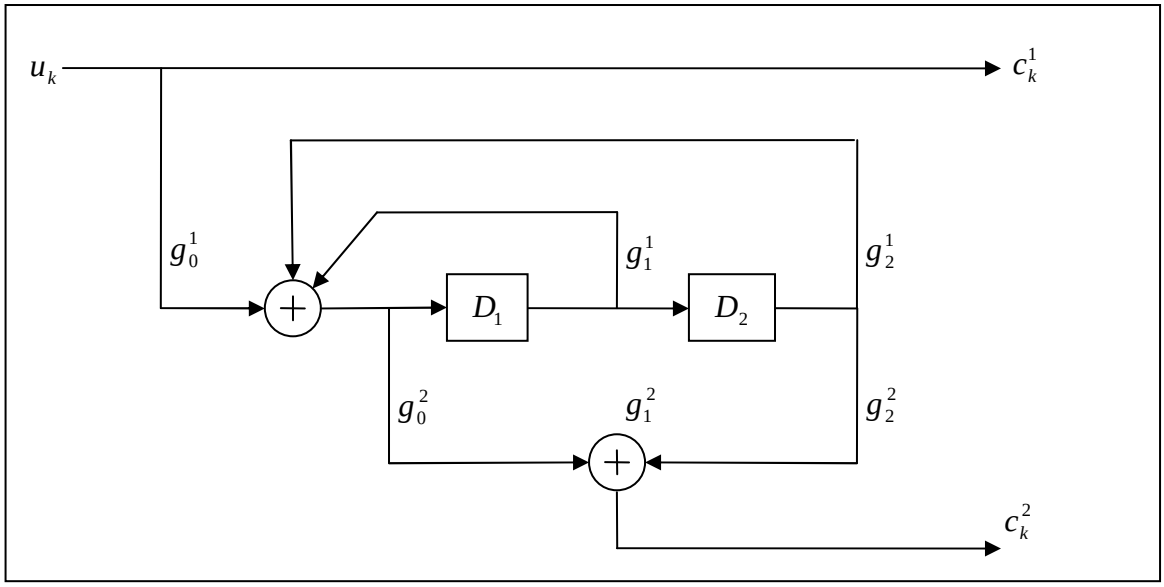
Giriş biti u_k, D_1 kaydedicisine iletilirken bu kaydedici içerisindeki bir önceki giriş biti olan u_{k-1} de bir sonraki kaydedici D_2'ye iletilmekte ve kodlama işlemi süresince bu böyle devam etmektedir.

2.1.2. Özyinelemeli Sistematik Katlamalı Kodlayıcı (RSC)

Özyinelemeli sistematik katlamalı (RSC) kodlayıcı, NSC kodlayıcının çıkışlarından birinin geri besleme çevrimi oluşturularak tekrar girişine verilmesiyle elde edilir. Özyinelemeli olarak adlandırılmasının nedeni yapılan geri besleme çevrimidir. Sistematik özelliği kazandırmak için ise kodlayıcı çıkışlarından bir tanesine direkt olarak giriş bitleri iletilir.

Genellikle katlamalı kodlayıcılar, NSC kodlayıcılar gibi, sonlu vuruş tepkesi (FIR) filtresi olarak düşünülse de, bunun aksine RSC kodlayıcılar sonsuz vuruş tepkesi (IIR)

filtresi olarak düşünülebilir. Ayrıca RSC kodlayıcıda bulunan geri besleme döngüsünden ve RSC kodlayıcının sistematik olmasından dolayı, RSC kodlayıcının üreteci NSC kodlayıcıdan farklı olacaktır.



Şekil 2.3. *R=1/2, K=3, G=(1,5/7)*$_8$ özyinelemeli sistematik katlamalı kodlayıcı

g_0^1, toplama düğümünden önceki kolu ifade ederken, g_0^2 ise toplama düğümü ile ilk kaydedici olan D_1 arasında kalan kolu ifade etmektedir. Bu nedenle RSC kodlayıcılarda g_0^1 her zaman "1" iken, diğer kollar "0" veya "1" olabilir. Bunları dikkate alarak bir RSC kodlayıcı üretecinin sekizli formu

$$G = \left(1, \frac{g^{(2)}}{g^{(1)}}\right) \tag{2.12}$$

olarak genelleştirilebilir. Denklem (2.12)' den yararlanılarak, Şekil 2.3' deki RSC kodlayıcının üreteç fonksiyonu şu şekilde ifade edilebilir:

$$\left.\begin{aligned} g^{(1)} &= \{g_0^1, g_1^1, g_2^1\} = \{1,1,1\} \\ g^{(2)} &= \{g_0^2, g_1^2, g_2^2\} = \{1,0,1\} \end{aligned}\right\} \Rightarrow G = \left(1, \frac{5}{7}\right)_8 \tag{2.13}$$

RSC kodlayıcı, sistematik ve bir geri besleme döngüsüne sahip olmasından dolayı, giriş bitleri ile çıkış bitleri arasındaki bağıntı da NSC kodlayıcıya göre farklı olacaktır. D_1 kaydırmalı kaydedicisinin girişi aşağıdaki gibi tanımlanırsa,

$$d_k = u_k + \sum_{i=0}^{m} g_i^1 . d_{k-1} \tag{2.14}$$

Bağıntısı elde edilir. Bu bağıntı kullanılarak, RSC kodlayıcının çıkışları şu şekilde ifade edilebilir:

$$c_k^1 = u_k \tag{2.15}$$

$$c_k^2 = \sum_{i=0}^{m} g_i^2 . d_{k-1} \tag{2.16}$$

RSC kodların sistematik bir yapıya sahip olmaları onların paralel bir bileşen olarak kullanılmalarını kolaylaştırmaktadır. Yine de, sistematik yapıları kullanışlı olmasına rağmen RSC kodların bir bileşen olarak seçilmesinin temel nedeni özyinelemeli olmalarıdır. Özyineleme özelliği sayesinde RSC kodların çıkışında, yüksek ağırlıklı kod kelimelerinin elde edilme olasılığı yüksek, düşük ağırlıklı kod kelimelerinin elde edilme olasılığı az olacaktır. Örneğin, Şekil 2.2 ve Şekil 2.3' deki NSC ve RSC kodlayıcıları ele alalım. Her iki kodlayıcının girişine sekiz bit uzunluğundaki $[1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$ giriş dizisi uygulanırsa aşağıdaki çıkışlar elde edilir:

- NSC kodlayıcı çıkışları:

$$c^{(1)} = [1\,1\,1\,0\,0\,0\,0\,0]$$
$$c^{(2)} = [1\,0\,1\,0\,0\,0\,0\,0]$$

- RSC kodlayıcı çıkışları:

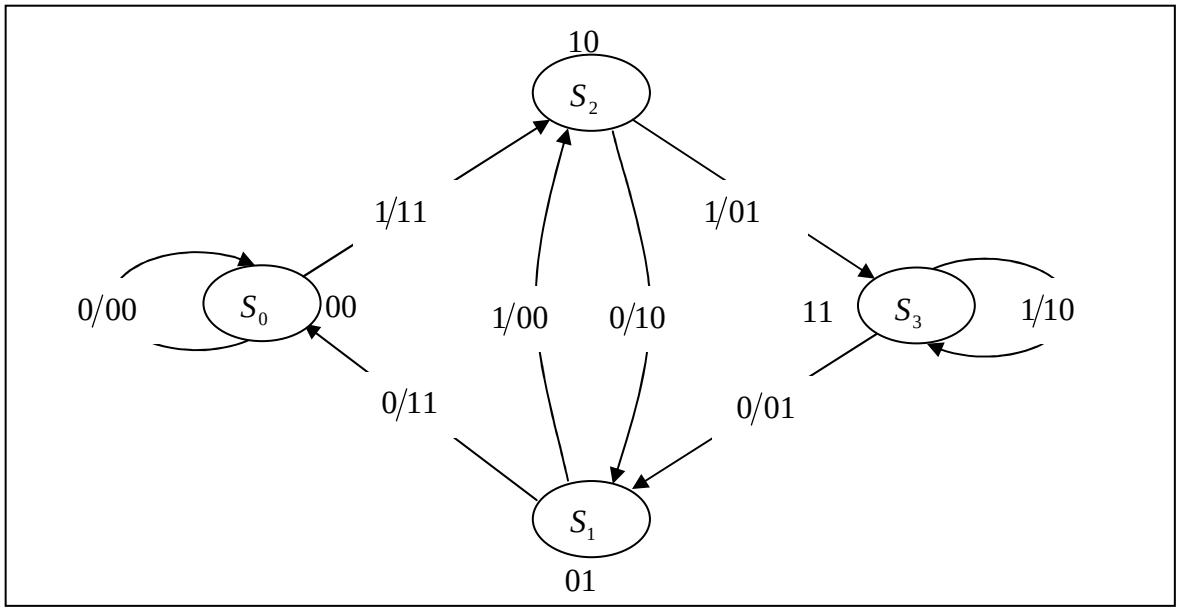
$$c^{(1)} = [1\,0\,0\,0\,0\,0\,0\,0]$$
$$c^{(2)} = [1\,1\,1\,0\,1\,1\,0\,1]$$

Elde edilen çıkış dizilerinden de görüleceği gibi, girişine kendisini 7 tane sıfır bitinin takip ettiği bir "1" biti uygulanırsa, NSC kodlayıcı çıkışında Hamming ağırlığı 5 olan bir kod kelimesi oluşurken, RSC kodlayıcı çıkışında ise Hamming ağırlığı 7 olan bir kod kelimesi oluşmaktadır.

Yukarıda belirtilen özelliklerden dolayı, turbo kodlarda paralel bileşenler olarak RSC kodlayıcılar kullanılmaktadır.

2.1.3. Katlamalı Kodların Analizi

Katlamalı kodlayıcılar sonlu vuruş tepkesi filtresi olarak düşünülebilir. Kodlayıcı sabit sayıda bellek elemanı içerdiğinden, belirli bir zamanda olası durumlardan birinde olduğu düşünülebilir. Kaydedicilerin sayısı *m* olarak tanımlanırsa, giriş bilgisine bağlı olarak 2^m tane olası durum ortaya çıkmaktadır. Belirli bir durumdan bir sonraki sabit duruma geçiş giriş dizisine bağlı olduğundan, katlamalı kodlayıcılar zaman-sabitli Markov zinciri yapısındadır [36], [37]. Durum geçişleri, durum diyagramı yardımı ile gösterilmektedir. Şekil 2.4'de Şekil 2.2'deki kodlayıcının durum diyagramı gösterilmektedir.



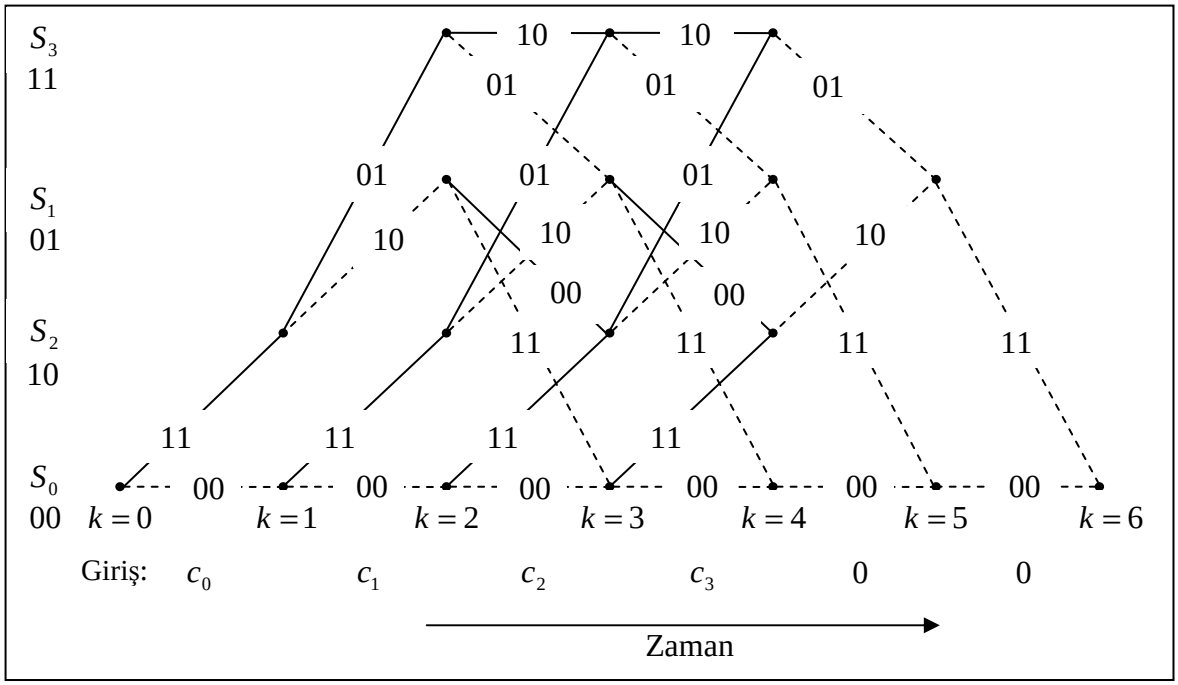
Şekil 2.4. *R=1/2, K=3, G=(7,5)₈* katlamalı kodlayıcı için durum diyagramı

Kodlayıcı iki tane kaydedici içermesinden dolayı belirli bir zaman içerisinde olası dört durumdan birinde bulunabilir. Bu nedenle, Şekil 2.4' deki durum diyagramında S_0, S_1, S_2 ve S_3 olmak üzere dört tane durum bulunmaktadır. Durum diyagramında geçişler u/c_1c_2 şeklinde ifade edilmektedir. Burada *u*, durumlar arası geçişe neden olan giriş bitini, c_1c_2 ise giriş bitinin kodlanması sonucunda elde edilen çıkış bitlerini temsil etmektedir.

Katlamalı kodların analizinde kullanılan diğer bir diyagram ise, durum diyagramının genişletilmiş hali olan ve zaman akışı içerisinde gerçekleşen olayları açıkça gösteren kafes diyagramıdır. Şekil 2.2'deki kodlayıcı için kafes diyagramı, Şekil 2.5'de gösterilmektedir.

Kafes diyagramı, t=0 anında ve “00” durumunda başlamaktadır [36]. Toplam (N+m) tane adımdan oluşmaktadır. Burada N giriş dizisinin boyu ve m ise kodlayıcıyı tekrar sıfır durumuna döndürmek için giriş dizisine eklenen kuyruk bitlerinin (*tail bits*) sayısıdır. Kuyruk bitlerinin sayısı, kodlayıcıda bulunan kaydedicilerin sayısına eşittir. Şekil 2.5’de verilen veri dizisinin boyu N=4’tür. Şekil 2.2’de ise, kaydedici sayısı m=2’dir. Bu nedenle, Şekil 2.5’deki kafes diyagramında altı tane adım ve her adım için de 2^m=4 tane durum bulunmaktadır. Eğer kullanılan kodlayıcı k tane girişe sahipse, kafes diyagramındaki her bir düğümden 2^k tane dal çıkacaktır. Benzer şekilde, k=m anından k=L anına kadar ise her bir düğüme 2^k tane dal girmiş olacaktır.

Kafes diyagramında, durum geçişlerine bağlı olarak oluşan çıkış bitleri gösterilmektedir. Örneğin, Şekil 2.5’deki kafes diyagramında düz çizgiler giriş bitinin “1” olması durumunda durum geçişini, kesikli çizgiler ise giriş bitinin “0” olması durumunda durum geçişini gösterirken, aynı zamanda çizgilerin yanındaki değerler ise kodlayıcının o anki çıkış değerini göstermektedir. k=0 anında kodlayıcının sıfır durumunda olduğunu kabul edersek, k=1 anında kafes diyagramı bize kodlayıcının olası iki durumdan sadece birinde olabileceğini göstermektedir: S_0=00 durumunda kodlayıcının girişine c_0=0 biti uygulandığı zaman, k=1 anında çıkış bitlerinin değeri “00” veya kodlayıcının girişine $c_0 = 1$ biti uygulandığı zaman k=1 anında çıkış bitlerinin değeri “11” olacaktır. Benzer şekilde, kodlayıcının k=2 anında sadece S_1 ve S_3 durumlarına geçebileceği görülmektedir. Diğer kafes adımları Şekil 2.5’de görülmektedir. k=4‘den sonraki girişler ise kuyruk bitleridir. Bu nedenle, bu andan itibaren durum geçişleri sadece girişin “0” olması durumundaki geçişler olacaktır.

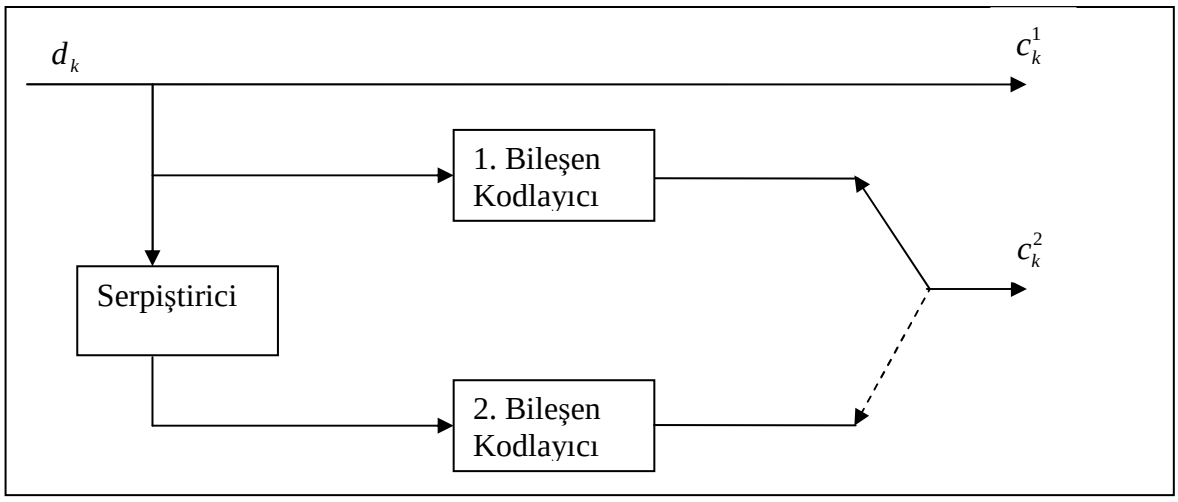


Şekil 2.5. *R=1/2, K=3, G=(7,5)$_8$* katlamalı kodlayıcı için kafes diyagramı

2.2. Turbo Kodlar

2.2.1. Turbo Kodlayıcı

Turbo kodlar yapısal olarak, serpiştirici ve bileşen kodlayıcı olarak kullanılan RSC kodların birleştirilmesi ile oluşturulan ileri yönde hata düzelmeyi sağlayan kodlardır[5],[16]. Şekil 2.6'da genel bir turbo kodlayıcının yapısı görülmektedir.



Şekil 2.6. Turbo kodlayıcı

Şekil 2.6'da görüldüğü gibi 1. bileşen kodlayıcı veri dizisini aynen alırken, 2. bileşen kodlayıcıya veri dizisi serpiştirilme işleminden sonra gönderilmektedir. Serpiştirici kaynak bitlerinin sırasını belirli bir kurala göre değiştirmektedir. Çünkü kod çözme işleminin yapılabilmesi için serpiştirici yapısının bilinmesi gerekmektedir. Şekil 2.6'da verilen turbo kodlayıcıda bileşen kodlayıcıların hızları sırası ile R_1 ve R_2 ise turbo kodlayıcının hızı aşağıdaki formül ile hesaplanmaktadır.

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} - 1 \tag{2.17}$$

Turbo kodlayıcıda sistematik kodların kullanılması, sistematik bitlerin kolay iletilmesini sağlamaktadır. Çünkü ikinci kodlayıcıya gelen sistematik bitler birinci kodlayıcıya gelen bitlerin serpiştirilmesiyle elde edildiğinden iletilmesine gerek yoktur. Her iki kodlayıcı çıkışındaki eşlik bitleri iletilmektedir. Kodlayıcı çıkışında istenen kod oranını elde etmek için delme işlemi uygulanmaktadır.

2.2.2. Serpiştirici

Katlamalı kodlar ve blok kodların büyük bir çoğunluğu tek şekilde (*uniform*) dağılmış hatalara karşı dayanıklı olup hataların gruplar halinde gelmesi (hata patlaması) durumunda hata düzeltme kapasiteleri oldukça düşmektedir. Pratikte, özellikle gezgin haberleşmede bayılmalardan (*fading*) dolayı oluşan düşük SNR oranları, hataların gruplar halinde gelmesine sebep olmakta[6],[8] ve kullanılan kodlama türüne göre bitlerin uygun bir şekilde serpiştirilmesini zorunlu kılmaktadır. Bir serpiştirici, önceden belirlenmiş fonksiyonu sayesinde giriş dizisindeki bitlerin yerlerini değiştirerek çıkışında giriş dizisiyle olabildiğince ilişkisiz bir dizi üreten devredir[23]. Zamanda giriş dizisinde birbirine yakın olan bitler serpiştiricinin çıkışında birbirinden uzaklaştırılarak giriş dizisiyle çıkış dizisi arasındaki ilişki küçültülmektedir. Genellikle bir serpiştirici hata patlamalarını düzgün dağıtmak için kullanılmaktadır. Düzgün dağıtmakta amaçlanan, simge bloklarının haberleşme kanalından iletiminde bilgi taşıyan simgeleri bozan kanal gürültüsünün yeniden şekillendirilmesidir. Bu şekillendirme alıcıya varan hatalı olarak alınan ardışıl simgelerin birbirinden serpiştirici sayesinde uzaklaştırılmasıyla yapılmaktadır. Kanal içerisinde gruplar halinde hatalar oluşacağı dikkate alınırsa, en mantıklı olanı klasik

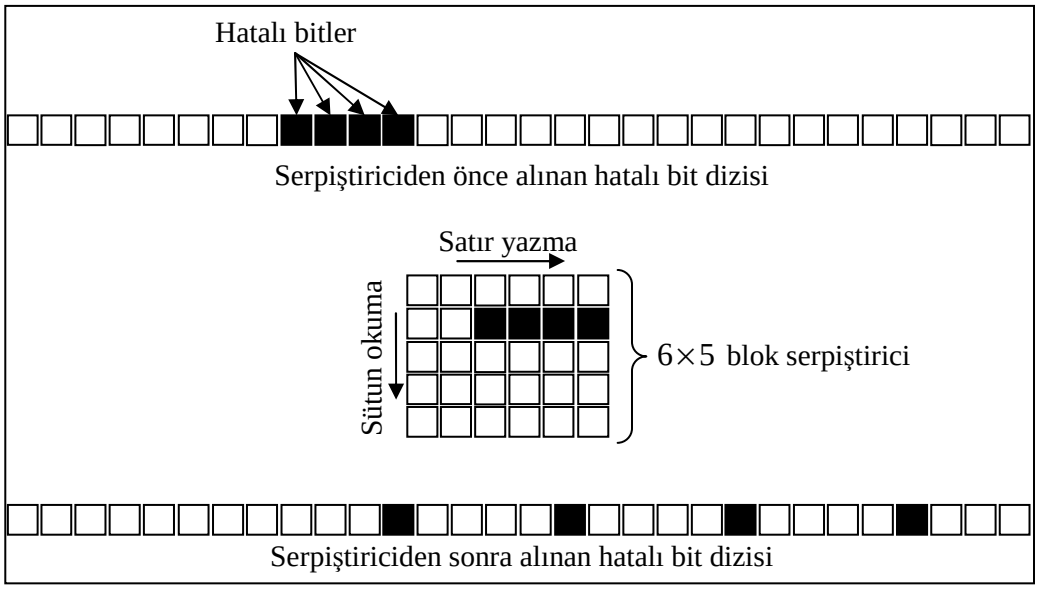
kodlama tekniklerinde olduğu gibi serpiştiriciyi kanal kodlayıcı ile kanal arasına yerleştirmektir. Fakat turbo kodlarda bu böyle olmamakla birlikte serpiştirici, turbo kodlayıcı içerisinde paralel olarak yerleştirilmiş RSC kodlayıcılar arasına yerleştirilmektedir. Bunun nedenleri kısaca şu şekilde özetlenebilir:

- Koda düzensizlik kazandırmak: Uzun düzensiz kodların, olabilecek en iyi performansı sağladıkları çok iyi bir şekilde bilinmektedir (Shannon Kapasite Limiti). Fakat bununla birlikte tamamen düzensiz kodlarda kod çözme işlemi gerçeklenemez. Bu nedenle alıcı tarafında kod çözme işlemini kolaylaştıracak bir kod yapısına ihtiyaç vardır. Serpiştirici, turbo kodlarda koda ek bir düzensizlik kazandırmakta ve alıcıda kod çözme işlemine izin verecek yeterli bir yapı oluşturmaktadır.
- Kodlayıcı çıkışında elde edilen düşük ağırlıklı (*low-weight*) kod kelimelerinin sayısını azaltmak: Eğer birinci RSC kodlayıcının girişinde, RSC kodlayıcının çıkışında düşük ağırlıklı kod kelimesi oluşturacak bir yapı varsa serpiştirici yardımıyla girişteki yapı yeniden düzenlenerek (bitlerin sırasını değiştirerek) diğer RSC kodlayıcının çıkışında düşük ağırlıklı kod kelimesinin oluşma olasılığını oldukça azaltmaktadır. Böylelikle, turbo kodlayıcının genelinde kodlayıcının çıkışında düşük ağırlıklı kod kelimelerinin meydana gelme olasılığı en aza indirilmektedir.
- Hata patlaması olması durumunda turbo kodlarda performansı artırmak: Hata patlaması kodun performansına zarar veren en büyük etkenlerden biridir. Serpiştirici, RSC kodlayıcı girişindeki veri bitlerini serpiştirerek orijinal veri dizisinde bulunan komşu bitleri birbirinden uzaklaştırmak suretiyle farklı yerlere yerleştirmektedir. Böylelikle, hata patlaması olması durumunda, serpiştirici sayesinde veri bloğu içinde hatalı olarak alınan bitler birbirinden uzaklaştırılarak turbo kodların performansı artırılmaktadır.

Turbo kodların başarımını etkileyen en önemli faktörlerden biri kullanılacak olan serpiştiricinin yapısıdır. Bu amaçla literatürde çeşitli serpiştirici oluşturma yöntemleri geliştirilmiştir. Bu serpiştirici yöntemlerinden üç tanesi, turbo kodlarda sıkça kullanılmaktadır[21]:

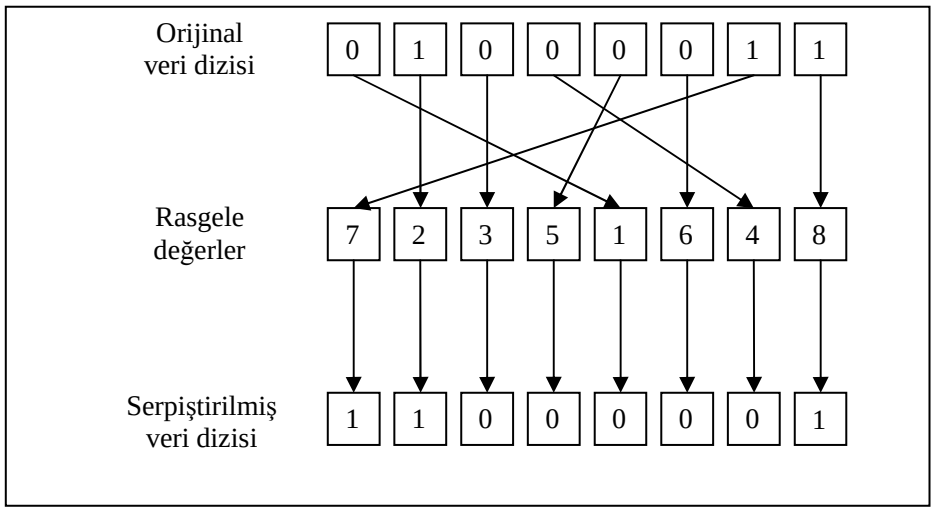
1. Blok serpiştirici (*block interleaver*): Turbo kodlarda kullanılan en basit serpiştirici yöntemidir. Bu yöntemde $a \times b$ boyutlu bir giriş dizisi, serpiştirme için kullanılan $a \times b$ boyutlu bir matrise satır satır yazılıp sütun sütun okunmaktadır. Böylelikle,

hata patlaması olması durumunda hatalı gelen bitler serpiştirici vasıtasıyla farklı yerlere serpiştirilmektedir. Şekil 14'de, grup halinde gelen dört tane hatalı bitin blok serpiştirici yardımı ile nasıl dağıtıldığı gösterilmektedir.



Şekil 2.7. (6×5) Blok serpiştirici

2. Rasgele serpiştirici (*pseudo-random interleaver*): Rasgele serpiştirici, orijinal veri dizisini belli bir kural çerçevesinde düzensiz olarak dağıtmaktadır.



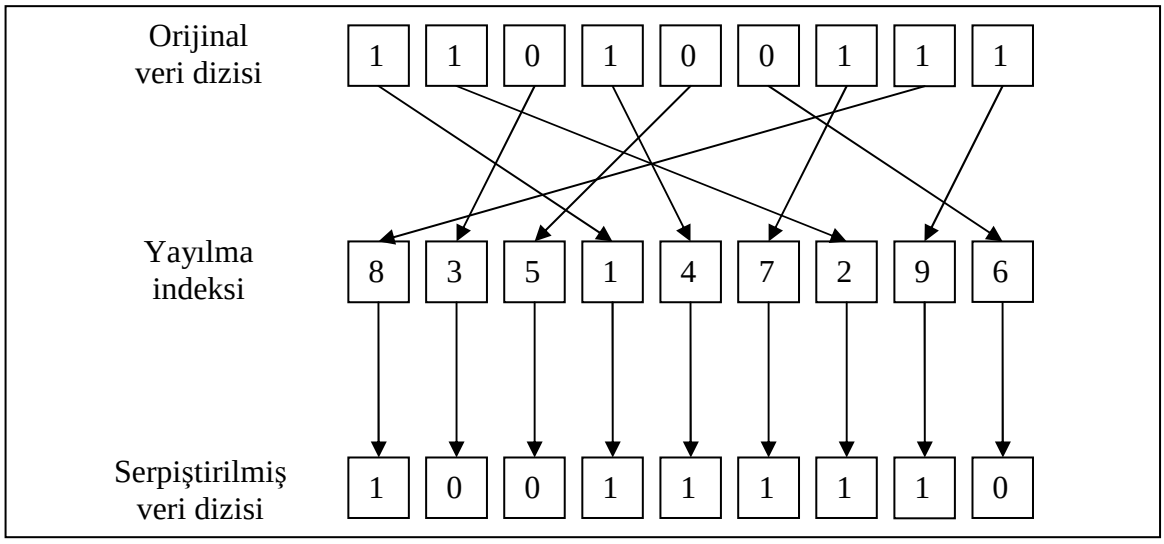
Şekil 2.8. Rasgele serpiştirici

Şekil 2.8‘den de görüldüğü gibi, rasgele serpiştirici kullanılarak orijinal veri dizisinden rasgele i. sıradaki veri alınarak, serpiştirilmiş veri dizisinde j. sıraya denk düşen yere yerleştirilmektedir.

3. Yarı-rasgele serpiştirici (*semi-random interleaver*): Yarı-rasgele serpiştirici, rasgele serpiştiriciye iyileştirme uygulanarak elde edilmektedir. Bu yöntemde, giriş dizisindeki bütün komşu bitler en az çerçeve uzunluğu N ile ilişkili olan S_r faktörü kadar birbirinden uzaklaştırılmaktadır.

$$S_r < \sqrt{\frac{N}{2}} \tag{2.18}$$

Bu serpiştirici yönteminde, çerçeve boyutundan küçük olması şartıyla daha önce seçilmemiş bir sayı rasgele seçilmekte ve bir önceki rasgele seçilmiş sayı ile karşılaştırılmaktadır. Eğer rasgele seçilen sayı, daha önce belirlenmiş olan sayının $\pm S_r$ aralığında ise tekrar yeni bir sayı seçilmektedir. Aksi takdirde, seçilen sayı hafızada tutulmaktadır. Bu işlemler, çerçeve boyutundan küçük tüm sayılar bir kere seçilene dek devam etmektedir. Şekil 2.9’da, N=9ve S_r=2 olan bir yarı rasgele serpiştirici örneği verilmiştir.



Şekil 2.9. Yarı-rasgele serpiştirici

Yapılan çalışmalar sonucunda, turbo kodlar için rasgele serpiştiricilerin blok serpiştiricilere oranla daha iyi bir performans sergilediği gözlenmiştir[5].

2.2.3. Turbo Kodlayıcıda Delme İşlemi

Şekil 2.6' da bütünü oluşturan her bir RSC kodlayıcının hızı 1/2 olduğu düşünülürse, turbo kodlayıcının hızı 1/3 olmaktadır. Bütünü oluşturan RSC kodlayıcılar c_k^s ve c_k^p olmak üzere iki çıkışa sahiptir. Fakat her iki kodlayıcı da sistematik olmasından dolayı sadece bir tane sistematik çıkış iletilmektedir. Böylelikle 2. bileşen kodlayıcının sistematik çıkışı elimine edilerek turbo kodlayıcının kod oranı 1/3 olarak belirlenmektedir.

Çeşitli oranlarda turbo kod oluşturmada, delme yönteminden yararlanmak tasarım kolaylığı sağlamaktadır. Önerilen yöntem kullanılarak, 1/3 ile 1 arasındaki herhangi bir kodlama oranı, turbo kodun iki bileşeni olan 1/2 oranlı katlamalı kodlar kullanılarak gerçekleştirilebilmektedir. Delme yöntemi kullanıldığında bazı eşlik bitleri kanaldan iletilmemektedir. Alıcı, iletilmeyen bu bitleri MAP kod çözücü girişine uygulamadan önce "-1" ile "+1" iletim düzeyleri arasında olan "0" değerinde yeniden üretmektedir. Kanaldan iletilen veya iletilmeyen eşlik bitlerini delme matrisi belirler. Delme matrisinin satır sayısı, turbo kodlayıcının paralel çıkış biti sayısına eşittir.

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Delme Matrisi

v_t^s	v_0^s	v_1^s	v_2^s	v_3^s	v_4^s	v_5^s	v_6^s	v_7^s	v_8^s	v_9^s	v_{10}^s	v_{11}^s	v_{12}^s
$v_{1,t}^p$	-	$v_{1,1}^p$	-	$v_{1,3}^p$	-	$v_{1,5}^p$	-	$v_{1,7}^p$	-	$v_{1,9}^p$	-	$v_{1,11}^p$	-
$v_{2,t}^p$	$v_{2,0}^p$	-	$v_{2,2}^p$	-	$v_{2,4}^p$	-	$v_{2,6}^p$	-	$v_{2,8}^p$	-	$v_{2,10}^p$	-	$v_{2,12}^p$

Şekil 2.10. Belirlenen delme matrisiyle, istenilen eşlik bitlerinin elimine edilmesi sonucunda kod oranının 1/3'ten 1/2'ye yükseltilmesi

Örneğin, Şekil 2.6'daki turbo kodlayıcı üç tane çıkışa sahip olmasından dolayı, Şekil 2.10'da gösterilen delme matrisinin satır sayısının üç olacağı açıktır. İlk satır sistematik

bitlere uygulanan delmeyi gösterir ve sistematik bitlerde delme olamayacağından tümü bir “1” vektörüdür. İkinci ve üçüncü satır, “1” ve “0” dan oluşmuş vektörlerdir ve sırasıyla birinci ve ikinci bileşen katlamalı kodun ürettiği eşlik bitlerine uygulanan delmeyi belirtmektedir. Delme matrisindeki “1” ve “0” lar, sırasıyla, delinmeyen ve delinen bitleri göstermektedir.

2.2.4. Turbo Kodlarda Kafes Sonlandırma

Kafes sonlandırma (*trellis termination*), kodlayıcının içeriğinin tekrar sıfır durumuna (başlangıç durumu) dönmesi ya da bir başka değişle kodlayıcıdaki tüm kaydırmalı kaydedicilerin içeriğinin sıfıra zorlanması olarak tanımlanabilir. Herhangi bir kodlayıcıda kafes sonlandırma, kuyruk bitlerinin yardımı ile gerçekleşmektedir. Daha öncede bahsedildiği gibi kuyruk bitleri, belirli bir kural çerçevesinde orijinal giriş dizisinin sonuna eklenen bit dizisidir. Kuyruk bitlerinin sayısı, kodlayıcı içerisindeki kaydırmalı kaydedicilerin sayısına eşittir.

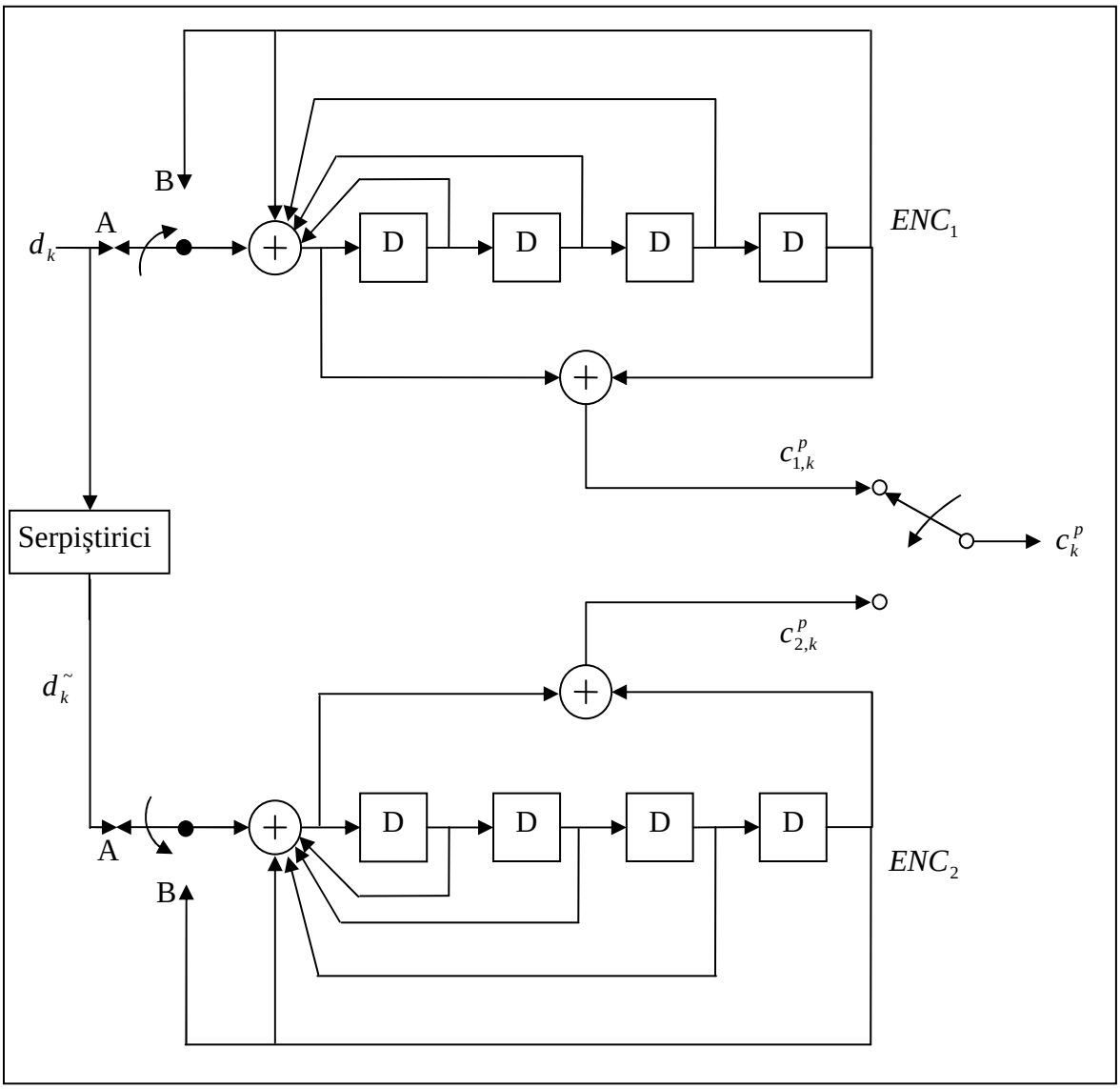
NSC kodlarda kafes sonlandırma, kodda bulunan kaydırmalı kaydedici sayısı kadar sıfır biti (kuyruk biti) giriş veri dizisine sonradan ilave edilerek sağlanmaktadır. Oysa, bu işlem turbo kodlarda bu kadar kolay değildir. Aşağıdaki iki faktör turbo kodlayıcılarda kafes sonlandırmasını zorlaştırmaktadır:

- Turbo kodlayıcıda bütünü oluşturan RSC kodlayıcılar geri besleme özelliği taşımasından dolayı, giriş veri dizisine kaydırmalı kaydedici sayısı kadar sıfır biti ilave etmek turbo kodlarda kafes sonlandırma için yeterli olmayacaktır.
- RSC kodlayıcılar arasına serpiştirici yerleştirilmesinden dolayı, RSC kodlayıcılardan birinde kafes sonlandırma için kullanılan kuyruk bitleri diğer RSC kodlayıcıda kafes sonlandırma için yeterli olmayacaktır. Serpiştiricinin giriş veri dizisini yeniden düzenlemesi nedeniyle her iki RSC kodlayıcıda da kafes sonlandırma için kullanılan kuyruk bitleri farklı olacaktır.

Sonuç olarak turbo kodlarda, RSC kodlayıcılardan biri için kafes sonlandırma kolay olurken diğeri için daha zor olacaktır. Bu nedenle, çoğu turbo kod uygulamalarında RSC kodlayıcılardan sadece bir tanesinde kafes sonlandırılmaktadır. Genellikle, girişinde serpiştirici bulunmayan RSC kodlayıcı seçilmektedir. Bu durum da, RSC kodlayıcının geri besleme kısmında bulunan kaydırmalı kaydedicilerin durumuna bakarak sırasıyla geri besleme değişkenini sıfır yapacak kuyruk bitini bulmak suretiyle sağlanmaktadır. RSC

kodlayıcıyı başlangıç durumuna sokmak için bu işlem kaydırmalı kaydedici sayısı kadar tekrarlanmalıdır.

Turbo kodlarda kafes sonlandırma için kullanılan bir diğer yöntem ise Şekil 2.11' da gösterilen zorlanmış kafes sonlandırma diyagramıdır[48]. Bu yöntem kullanılarak, bütünü oluşturan her iki RSC kodlayıcıda da kafes sonlandırılabilmektedir.



Şekil 2.11. *R=1/3, K=5, G=(37,21)*$_8$ turbo kodlayıcı için zorlanmış kafes sonlandırması

Şekil 2.11'den görüldüğü gibi, her iki RSC kodlayıcının girişinde birer anahtar bulunmaktadır. Kodlama işlemi süresince anahtarlar *A* konumunda bulunurken, kodlama işleminin tamamlanmasından sonra anahtarlar *B* konumuna geçecek ve böylece her iki

kodlayıcıya özgü kuyruk bitleri kodlayıcıya uygulanmış olacaktır. Kodlayıcıların başlangıç durumuna dönmesiyle birlikte anahtarlar tekrar *A* konumuna geçecektir. Yalnız unutulmamalıdır ki, her iki kodlayıcı için elde edilen kuyruk biti birbirinden farklı olacaktır. Bu da, kodlayıcıların farklı veriler üzerinde işlem yapmasına ve kod çözücüde ise bir karışıklığın oluşmasına neden olacaktır. Bu nedenle bu yöntem pek fazla tercih edilmemektedir.

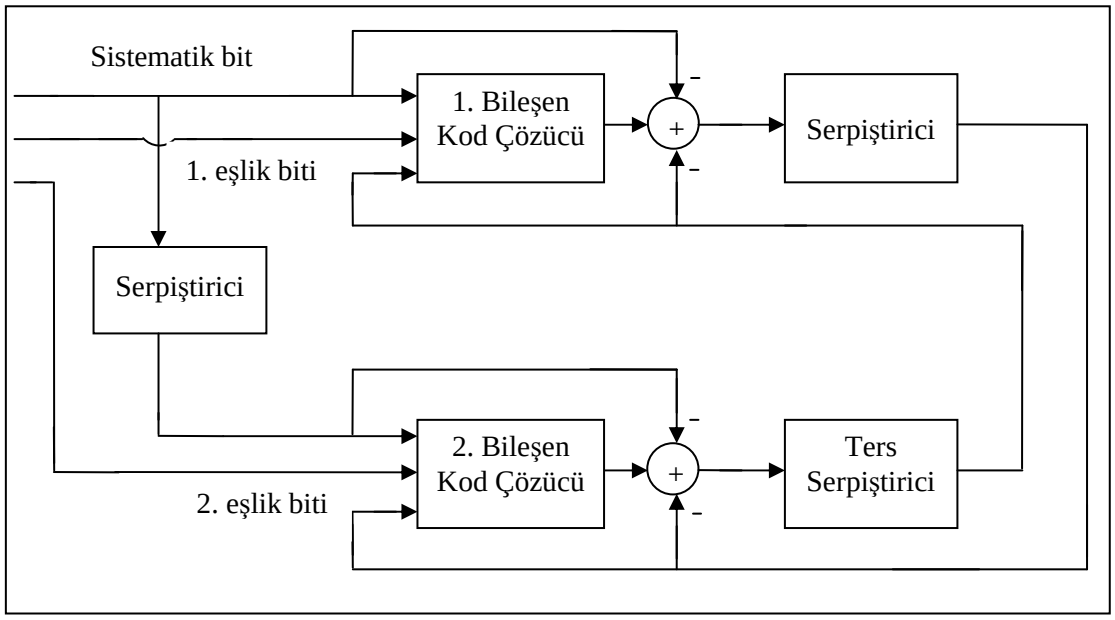
2.2.5. Turbo Kod Çözücü

Şekil 2.12' de turbo kod çözücünün genel yapısı görülmektedir. Şekilden de görüldüğü gibi turbo kod çözücü turbo kodlayıcıda yer alan her bir bileşen kodlayıcı için yumuşak-giriş yumuşak-çıkış (SISO) kod çözücü elemanlar içermektedir. Her bir kod çözücü üç giriş alır: kanal çıkışındaki kodlanmış sistematik bitler, ilgili kodlayıcıya ait iletilen eşlik biti ve diğer bileşen kodlayıcı tarafından üretilen ön (a-priori) bilgi[15].

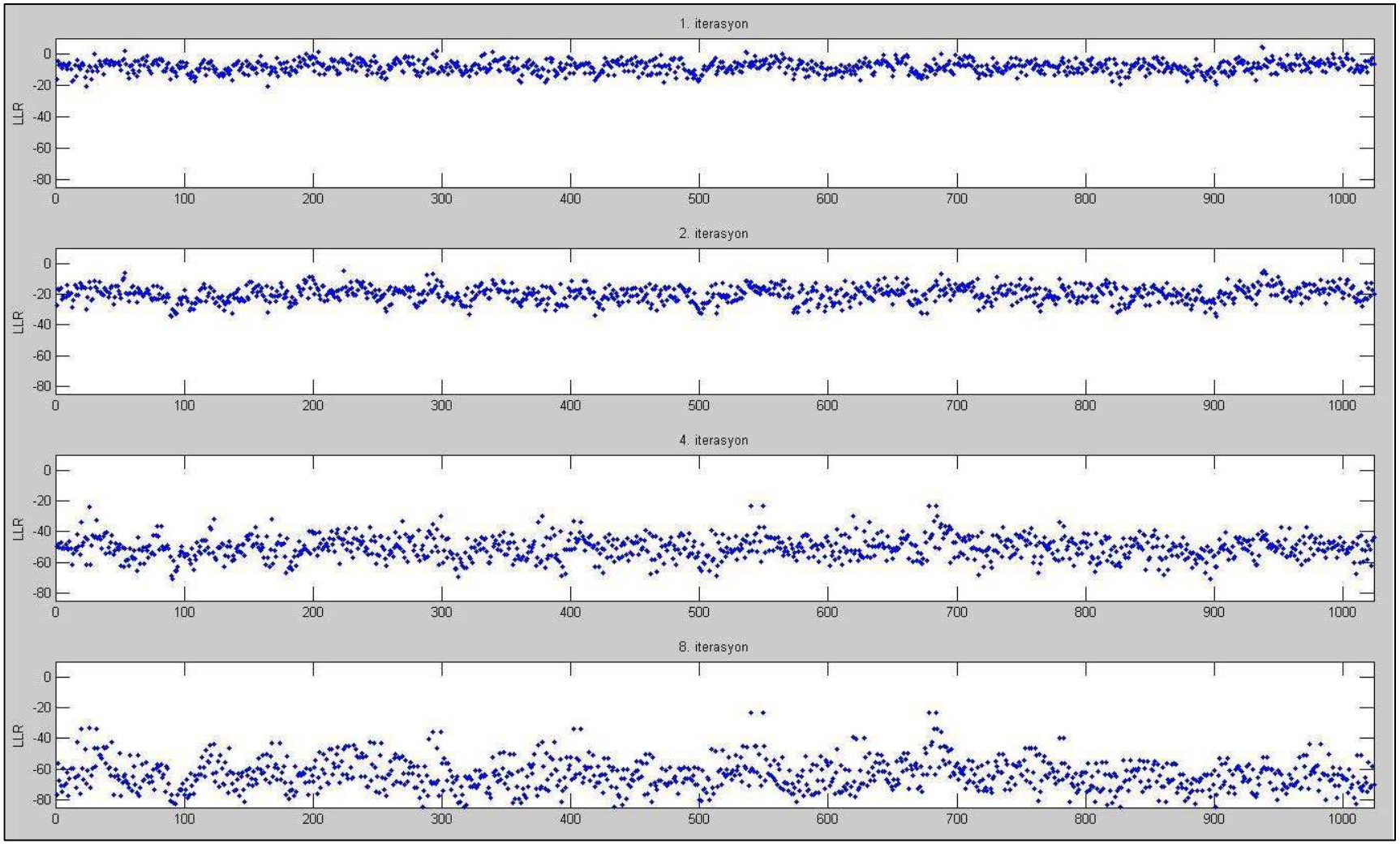
Şekil 2.12' de görüldüğü gibi ikinci bileşen kod çözücüye kanal çıkışındaki sistematik bitler serpiştirildikten sonra gönderilmektedir. Çünkü kodlayıcıda, bileşen kodlayıcı olarak RSC kodlar kullanıldığından ikinci bileşen kodlayıcı çıkışındaki sistematik bitler iletilmemektedir.

Turbo kod çözücü, iteratif olarak iki bileşen kod çözücü arasında yan bilgi (extrinsic information) iletimine dayalı kod çözme işlemi gerçekleştirmektedir[8],[9]. Birinci bileşen kod çözücünün ilk iterasyonda olduğu düşünülürse, kanal çıkışındaki sistematik bitleri ve kendine ait eşlik bitlerini giriş olarak almaktadır. İlk iterasyonda sistematik bitler hakkında ön (a–priori) bilgiye sahip değildir. Yani bu bilgi sıfır olmaktadır. Birinci bileşen kod çözücü, aldığı girişlerle veri bitleri hakkında kendi yumuşak (soft) çıkışlarını üretmektedir. Bu işlemden sonra ikinci bileşen kod çözücü işleme girmektedir. Bu kod çözücü sistematik bitlerin serpiştirilmiş halini, kendine ait eşlik bitlerini ve veri bitleri hakkında birinci bileşen kod çözücü tarafında üretilen ön (a-priori) bilgiyi giriş olarak almaktadır. Bu girişleri kullanarak kendi yumuşak (soft) çıkışlarını üretmekte ve böylece ilk iterasyon sonlanmaktadır. İkinci iterasyon başladığında birinci bileşen kod çözücü ilk iterasyondan farklı olarak ikinci bileşen kod çözücü tarafından üretilen ön bilgiyi de giriş olarak almakta ve birinci iterasyonda yapılan işlemler belirlenen sayıda devam etmektedir. Her bir iterasyon sonucunda veri hakkında elde edilen bilginin doğruluğu artmaktadır. Şekil 2.12 incelendiğinde kod çözücü çıkışındaki yumuşak (soft)

değerlerin iterasyon sayısı ile genlik değerlerinin arttığı görülmektedir. Bu, iterasyon sayısı ile birlikte hatalı bit sayısının azaldığı anlamına gelmektedir[9].



Şekil 2.12. Turbo kod çözücünün genel yapısı



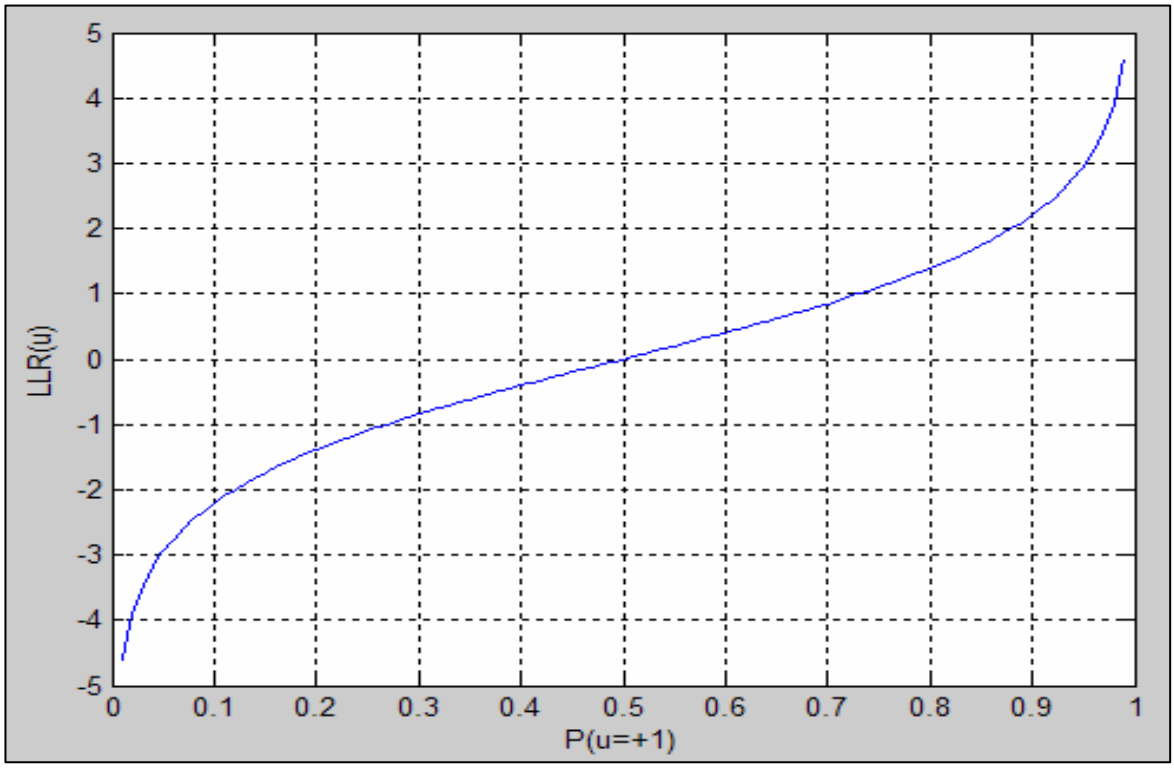
Şekil 2.13. İletilen bit dizisinin -1 olması durumunda kod çözücü çıkışından elde edilen yumuşak (soft) değerler

2.2.6. Logaritmik Benzerlik Oranı (Log Likelihood Ratio)

LLR kavramı turbo kod çözücüde bileşen kod çözücüler arasında bilgi aktarımında kullanılmak üzere ilk defa Patrick Robertson tarafından tanımlanmıştır[15]. u_k veri bitinin logaritmik benzerlik oranı $L(u_k)$ olarak ifade edilmekte ve

$$L(u_k) \stackrel{\Delta}{=} \ln\left(\frac{P(u_k = +1)}{P(u_k = -1)}\right) \tag{2.19}$$

olarak tanımlanmaktadır. (2.19) denkleminden görüldüğü gibi u_k bitinin iki farklı değer alma olasılığı vardır ve LLR ifadesi bu değerlerin olasılıklarının logaritmik oranı olarak tanımlanmaktadır.



Şekil 2.14. $P(u=+1)$ olasılığı ile LLR ($L(u)$) değişimi

Şekil 2.14'den görüldüğü gibi LLR ($L(u_k)$) değerinin işareti u_k bitinin değerini (+1 veya -1) genliği ise u_k biti hakkında elde edilen olasılık değerinin doğruluğunun bir göstergesidir[9]. (2.19) bağıntısı incelendiğinde u_k bitine ait LLR değeri biliniyorsa, u_k bitine ait $P(u_k=+1)$ ve $P(u_k=-1)$ olasılık değerleri hesaplanabilmektedir.

$L(u_k) \approx 0$ ise $P(u_k = +1) \approx P(u_k = -1) \approx 0.5$

$L(u_k) >> 0$ ise $P(u_k = +1) >> P(u_k = -1)$

$L(u_k) << 0$ ise $P(u_k = +1) << P(u_k = -1)$ dır.

u_k bitinin iki farklı değer aldığı dikkate alınarak aşağıdaki eşitlikler yazılabilir.

$$P(u_k = +1) + P(u_k = -1) = 1 \tag{2.20}$$

$$P(u_k = -1) = 1 - P(u_k = +1) \tag{2.21}$$

(2.19) ve (2.21) eşitliklerinden yararlanarak;

$$e^{L(u_k)} = \frac{P(u_k = +1)}{1 - P(u_k = +1)} \tag{2.22}$$

ifadesi elde edilir. Bu eşitlik kullanılarak

$$\begin{aligned} P(u_k = +1) &= \frac{e^{L(u_k)}}{1 + e^{L(u_k)}} \\ &= \frac{1}{1 + e^{-L(u_k)}} \end{aligned} \tag{2.23}$$

$$\begin{aligned} P(u_k = -1) &= \frac{1}{1 + e^{L(u_k)}} \\ &= \frac{e^{-L(u_k)}}{1 + e^{-L(u_k)}} \end{aligned} \tag{2.24}$$

ifadeleri elde edilir. (2.23) ve (2.24) bağıntıları birleştirilirse

$$P(u_k = \pm 1) = \left(\frac{e^{-L(u_k)/2}}{1 + e^{-L(u_k)}} \right) . e^{\pm L(u_k)/2} \tag{2.25}$$

bağıntısı elde edilmektedir. (2.19) ifadesinde u_k bitine ait LLR değeri ifade edilirken taban olasılıklar dikkate alınmıştı. Kanal kodlama teorisinde bu olasılık değerleri ifade edilirken

koşullu olasılık kavramı dikkate alınarak belirli gözlem değerlerine bağlı olarak ifade edilmesi gerekmektedir. Bundan dolayı (2.19) ifadesi

$$L(u_k \mid \underline{y}) \stackrel{\Delta}{=} \ln\left(\frac{P(u_k = +1 \mid \underline{y})}{P(u_k = -1 \mid \underline{y})}\right) \tag{2.26}$$

şeklinde yazılır.

$P\left(u_k = \pm 1 \mid \underline{y}\right)$ koşullu olasılığı u_k bitine ait sonsal (a – posteriori) olasılık olarak bilinmektedir. $P(u_k=\pm 1)$ olasılığı ise a-priori olasılık olarak ifade edilmektedir. Her hangi bir olayın a-posteriori (APP) olasılığı bir başka olayın gerçekleşmesinin fonksiyonudur.

BPSK modülasyonu kullanılarak $x_k=\pm 1$ biti gaussian kanaldan iletilirse ve iletilen bite karşılık y_k bitinin alındığı kabul edilirse, y_k' nın genlik dağılımları

$$P(y_k \mid x_k = +1) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{E_b}{2\sigma^2}\left(y_k - a\right)^2\right) \tag{2.27}$$

benzer şekilde

$$P(y_k \mid x_k = -1) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{E_b}{2\sigma^2}\left(y_k + a\right)^2\right) \tag{2.28}$$

şeklinde olacaktır. Bu denklemlerde E_b iletilen bitin enerjisini, σ^2 gürültünün varyansını, a ise bayılma genliğini ifade etmekte ve gaussian kanallar için a=1' dir.

$L\left(y_k \mid x_k\right)$ benzerlik oranı

$$L(y_k \mid x_k) \stackrel{\Delta}{=} \ln\left(\frac{P(y_k \mid x_k = +1)}{P(y_k \mid x_k = -1)}\right) \tag{2.29}$$

olarak tanımlanmaktadır.

(2.27) ve (2.28) bağıntısı (2.29) ifadesinde kullanılarak

$$
\begin{aligned}
L(y_k \mid x_k) &\stackrel{\Delta}{=} \ln\left(\frac{P(y_k \mid x_k = +1)}{P(y_k \mid x_k = -1)}\right) \\
&= \ln\left(\frac{\exp(\frac{-E_b}{2\sigma^2}(y_k - a^2))}{\exp(\frac{-E_b}{2\sigma^2}(y_k + a^2))}\right) \\
&= \left(\frac{-E_b}{2\sigma^2}(y_k - a^2)\right) - \left(\frac{-E_b}{2\sigma^2}(y_k + a^2)\right) \\
&= \frac{E_b}{2\sigma^2} 4a.y_k \\
&= L_c y_k
\end{aligned}
\tag{2.30}
$$

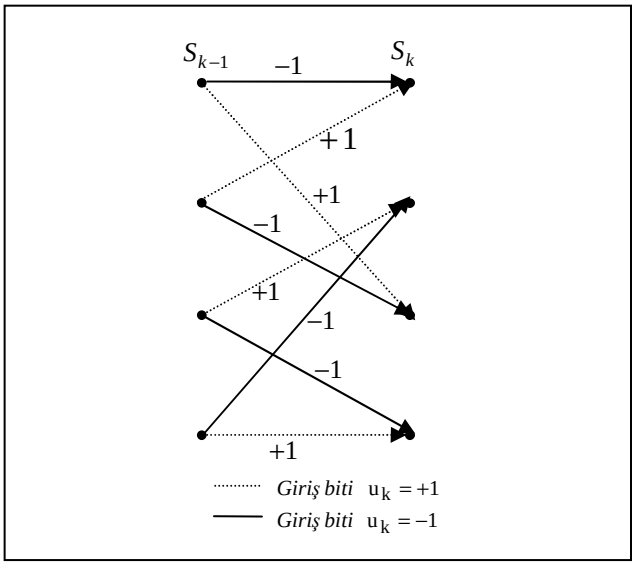
ifadesi elde edilir. Burada

$$L_c = 4a\frac{E_b}{2\sigma^2} \tag{2.31}$$

L_c kanal güvenilirlik katsayısı olarak tanımlanmaktadır. (2.31) bağıntısından görüldüğü gibi L_c sadece işaret-gürültü oranına (SNR) ve kanalın bayılma genliğine bağlıdır.

2.2.7. MAP (Maximum A-Posteriori) Algoritması

MAP (Maximum A-Posteriori) algoritması 1974 yılında L.R. Bahl, J. Cocke, F. Jelinek ve J. Raviv tarafından önerilmiştir[14]. BCJR algoritması olarak da bilinmektedir. MAP kod çözücü algoritması kanal çıkışındaki N blok uzunluğundaki yumuşak (soft) veriyi kullanarak her bir bitin LLR(Log Likelihood ratio)' sini bir başka ifade ile $P(u_k=+1)$ ve $P(u_k=-1)$ olasılıklarını hesaplayan iteratif bir tekniktir[17],[18]. Şekil 2.15' deki trellis diyagramı incelendiğinde S_{k-1} ve S_k durumları bilindiğinde bu durumlar arası geçişe neden olan giriş bitinin bilineceği açıktır. Bu ifadeden $P(u_k=+1)$ olasılığı trellis diyagramında, kodlayıcı girişindeki $u_k=+1$ giriş biti için, bir önceki durum S_{k-1}'den şimdiki durum S_k'ya olan tüm geçişlerin olasılıklarının toplamına eşit olmaktadır. Şekil 2.15' deki trellis diyagramı incelendiğinde bu geçiş kollarının dört tane olduğu görülmektedir. $P(u_k=-1)$ olasılığı da benzer şekilde hesaplanmaktadır.



Şekil 2.15. *K*=3 olan RSC kodun (Şekil 1.3) trellis diyagramı

Bayes teoremi kullanılarak iki olayın koşullu olasılığı

$$P(a,b) = P(a \mid b)P(b) \tag{2.32}$$

şeklinde ifade edilmektedir. (2.32) denkleminde *P(a|b)* terimi "b" olayı gerçekleştiğinde "a" olayının oluşma olasılığını vermektedir. Eğer "*a*" olayı daima "*b*" olayı ile gerçekleşiyorsa "a" olayının mutlak olasılığı

$$P(a) = \sum_{b} P(a,b) \tag{2.33}$$

şeklinde ifade edilmektedir. (2.32) ifadesinden türetilerek

$$P(\{a,b\} \mid c) = P(a \mid \{b,c\}).P(b \mid c) \tag{2.34}$$

ifadesi yazılabilmektedir.

Aşağıda (2.34) ifadesinin (2.32) ifadesinden nasıl türetildiği gösterilmiştir.

$x \equiv a,b$, $y \equiv b,c$ olduğu düşünülürse

$$
\begin{aligned}
P(\{a,b\}\mid c) = P(x\mid c) &= \frac{P(x,c)}{P(c)} \\
&= \frac{P(a,b,c)}{P(c)} \equiv \frac{P(a,y)}{P(c)} \\
&= \frac{P(a\mid y).P(y)}{P(c)} \equiv P(a\mid\{b,c\}).\frac{P(b,c)}{P(c)} \\
&= P(a\mid\{b,c\}).P(b\mid c)
\end{aligned}
\tag{2.35}
$$

ifadesi elde edilmektedir.

Yukarıda bahsedildiği gibi MAP algoritması her bir u_k giriş biti için LLR ifadesini hesaplamaktadır. Kanal çıkışından elde edilen yumuşak (soft) değerleri $\underline{y}$ vektörüyle gösterilirse LLR $L(u_k \mid \underline{y})$ ifadesi

$$
L(u_k \mid \underline{y}) = \ln\left(\frac{P(u_k = +1 \mid \underline{y})}{P(u_k = -1 \mid \underline{y})}\right) \tag{2.36}
$$

şeklinde olur.

Bayes teoremi kullanılarak (2.36) eşitliği tekrar düzenlenip (2.37) eşitliği olarak ifade edilir.

$$
L(u_k \mid \underline{y}) = \ln\left(\frac{P(u_k = +1, \underline{y})}{P(u_k = -1, \underline{y})}\right) \tag{2.37}
$$

(2.36) ve (2.37) eşitlikleri incelendiğinde (2.36) eşitliğinde kanal çıkışından elde edilen yumuşak (soft) değerler ile u_k giriş biti birbirinden bağımsızdır. (2.37) eşitliğinde ise bağımsız değillerdir. Kodlayıcının kafes diyagramındaki geçişler ile birbirlerine bağlıdırlar. Bu geçişler gürültü tarafından bozulur.

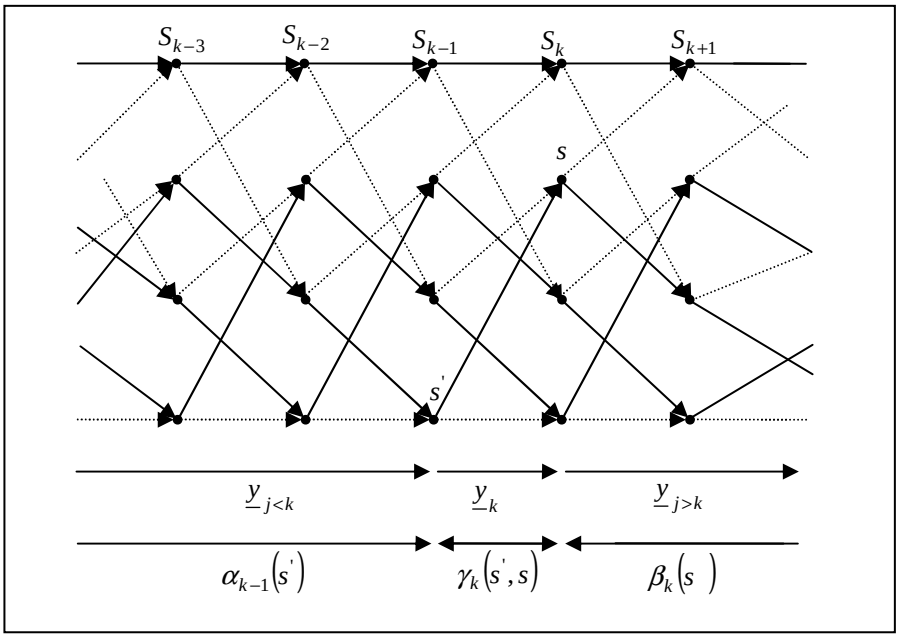
Şekil 2.14’ deki kafes diyagramı incelendiğinde kodlayıcının dört duruma sahip olduğu görülmektedir. Kodlayıcıya olası iki farklı bit girişi olduğu göz önüne alınırsa (ikili kod), kodlayıcının her bir durumundan giriş bitinin değerine bağlı olarak olası iki farklı duruma geçişi mümkündür. Bu durum geçişlerinin biri +1 giriş biti için diğeri ise -1 giriş biti için olur.

Bit olasılıkları hesaplanırken durum geçişleri dikkate alındığından (2.37) eşitliği

$$L(u_k \mid \underline{y}) = \ln\left(\frac{\sum_{\substack{(s',s)\Rightarrow \\ u_k=+1}} P(S_{k-1}=s',\, S_k=s,\, \underline{y})}{\sum_{\substack{(s',s)\Rightarrow \\ u_k=-1}} P(S_{k-1}=s',\, S_k=s,\, \underline{y})}\right) \tag{2.38}$$

şeklinde ifade edilmektedir.

Bu ifadede yer alan $(s',s)\Rightarrow u_k=+1$ ve $(s',s)\Rightarrow u_k=-1$ ifadeleri sırası ile u_k=+1 ve u_k=−1 giriş biti için önceki durum s' den mevcut durum s'e olan durum geçişlerinin kümesini belirtmektedirler. Sade olması için (2.38) denklemindeki $P(S_{k-1}=s', S_k=s, \underline{y})$ terimi $P(s', s, \underline{y})$ olarak kullanılacaktır. Şekil 2.16' da görüldüğü gibi $\underline{y}$ vektörü üç parçaya ayrılmıştır. $\underline{y}_k$ mevcut durum geçişi ile alınan kod sözcüğüne, $\underline{y}_{j<k}$ mevcut durum geçişinden önce alınan diziye ve $\underline{y}_{j>k}$ mevcut durum geçişinden sonra alınan diziye karşılık gelmektedir.



Şekil 2.16. K=3 RSC kodlayıcı için MAP kod çözücü kafes diyagramı

$P(s', s, \underline{y})$ terimi

$$P(s', s, \underline{y}) = P(s', s, \underline{y}_{j<k}, \underline{y}_k, \underline{y}_{j>k}) \tag{2.39}$$

şeklinde yazılabilir. (2.32) denklemiyle verilen Bayes kuralı kullanılarak ve kanalın hafızasız olduğu kabul edilirse $\underline{y}_{j>k}$ terimi sadece mevcut durum *s* ' e bağlı olmaktadır. Yani bir önceki durum s' ve mevcut ve önceki alınan diziler $\underline{y}_k$ ve $\underline{y}_{j<k}$' ya bağlı olmayacaktır. Bu durumda (2.39) ifadesi tekrar düzenlenirse

$$\begin{aligned} P(s', s, \underline{y}) &= P(\underline{y}_{j>k} \mid \{s', s, \underline{y}_{j<k}, \underline{y}_k\}).P(s', s, \underline{y}_{j<k}, \underline{y}_k) \\ &= P(\underline{y}_{j>k} \mid s).P(s', s, \underline{y}_{j<k}, \underline{y}_k) \end{aligned} \tag{2.40}$$

elde edilmektedir. (2.40) denklemi tekrar kanalın hafızasız olduğu kabul edilip, bayes kuralından yararlanılıp yeniden düzenlenirse

$$\begin{aligned} P(s', s, \underline{y}) &= P(y_{-j>k} \mid s).P(s', s, \underline{y}_{j<k}, \underline{y}_k) \\ &= P(\underline{y}_{j>k} \mid s).P(\{\underline{y}_k, s\} \mid \{s', \underline{y}_{j<k}\}).P(s', \underline{y}_{j<k}) \\ &= P(\underline{y}_{j>k} \mid s).P(\{\underline{y}_k, s\} \mid s').P(s', \underline{y}_{j<k}) \\ &= \beta_k(s).\gamma_k(s', s).\alpha_{k-1}(s') \end{aligned} \tag{2.41}$$

elde edilmektedir. Bu ifadede yer alan $\alpha_{k-1}(s')$, $\beta_k(s)$ ve $\gamma_{k-1}(s', s)$ terimlerinin açıklamaları aşağıda verilmiştir.

$$\alpha_{k-1}(s') = P(S_{k-1} = s', \underline{y}_{j<k}) \tag{2.42}$$

$$\beta_k(s) = P(\underline{y}_{j>k} \mid S_k = s) \tag{2.43}$$

$$\gamma_k(s', s) = P(\{\underline{y}, S_k = s\} \mid S_{k-1} = s') \tag{2.44}$$

$\alpha_{k-1}(s')$ terimi durum diyagramında $k-1$ anında ve s' durumundaki olasılığa karşılık gelmektedir. Bu ana kadar kanaldan alınan dizi $\underline{y}_{j<k}$ ' dır.

$\beta_k(s)$ terimi durum diyagramında *k* anında *s* durumunda kanaldan alınacak dizi $\underline{y}_{j>k}$ olduğundaki olasılığa karşılık gelmektedir.

Son olarak $\gamma_k\left(s^{'},s\right)$ terimi ise durum diyagramında $k-1$ anında $s^{'}$ durumundan s durumuna geçildiğinde ve bu geçişe karşılık $\underline{y}_k$ dizisi alındığında bu duruma karşılık gelen olasılığı belirtmektedir.

2.2.8. $\alpha_k(s)$ Değerlerinin Hesaplanması

$\alpha_k(s)$ terimi (2.42) bağıntısı ile verilen $\gamma_{k-1}(s^{'},s)$ ifadesinden yaralanılarak

$$\begin{aligned}\alpha_k(s) &= P\left(S_k = s\,,\,\underline{y}_{j<k+1}\right)\\ &= P\left(s\,,\,\underline{y}_{j<k}\,,\,\underline{y}_k\right)\\ &= \sum_{tüm\ s^{'}} P\left(s\,,\,s^{'}\,,\,\underline{y}_{j<k}\,,\,\underline{y}_k\right)\end{aligned} \tag{2.45}$$

şeklinde ifade edilebilmektedir.

Kanalın hafızasız olduğu kabul edilip Bayes kuralı kullanılarak (2.45) ifadesi aşağıdaki gibi yazılabilir.

$$\begin{aligned}\alpha_k(s) &= \sum_{tüm\ s^{'}} P\left(s,s^{'},\underline{y}_{j<k}\,,\,\underline{y}_k\right)\\ &= \sum_{tüm\ s^{'}} P\left(\{s,\underline{y}_k\}\,|\,\{s^{'}\,,\underline{y}_{j<k}\}\right).P\left(s^{'}\,,\,\underline{y}_{j<k}\right)\\ &= \sum_{tüm\ s^{'}} P\left(\{s\,,\underline{y}_k\}\,|\,s^{'}\right).P\left(s^{'}\,,\,\underline{y}_{j<k}\right)\\ &= \sum_{tüm\ s^{'}} \gamma_k\left(s^{'},s\right).\alpha_{k-1}\left(s^{'}\right)\end{aligned} \tag{2.46}$$

(2.46) ifadesi incelendiğinde $\alpha_k(s)$ değerlerinin hesaplanabilmesi için ilk olarak $\gamma_{k-1}(s^{'},s)$ değerlerinin bilinmesi gerekmektedir. $\gamma_{k-1}(s^{'},s)$ değerleri elde edildikten sonra $\alpha_k(s)$ değerleri iteratif olarak hesaplanmaktadır. Kafes diyagramının başlangıçta $S_0=0$ durumunda olduğu kabul edilirse $\alpha_k(s)$ teriminin başlangıç değerleri

$$\begin{aligned}&\alpha_0(S_0 = 0) = 1\\ &\alpha_0(S_0 = s) = 0, \qquad s \neq 0\ için\end{aligned} \tag{2.47}$$

şeklinde olmaktadır.

2.2.9. $\beta_k(s)$ Değerlerinin Hesaplanması

(2.43) bağıntısı ile verilen $\beta_k(s)$ ifadesinden yaralanılarak $\beta_{k-1}(s')$ terimi

$$\beta_{k-1}(s') = P(\underline{y}_{j>k-1} \mid S_{k-1} = s') \tag{2.48}$$

şeklinde ifade edilebilmektedir.

Kanalın hafızasız olduğu kabul edilip Bayes kuralı kullanılarak (2.48) ifadesi tekrar tanımlanırsa aşağıdaki ifade elde edilmektedir.

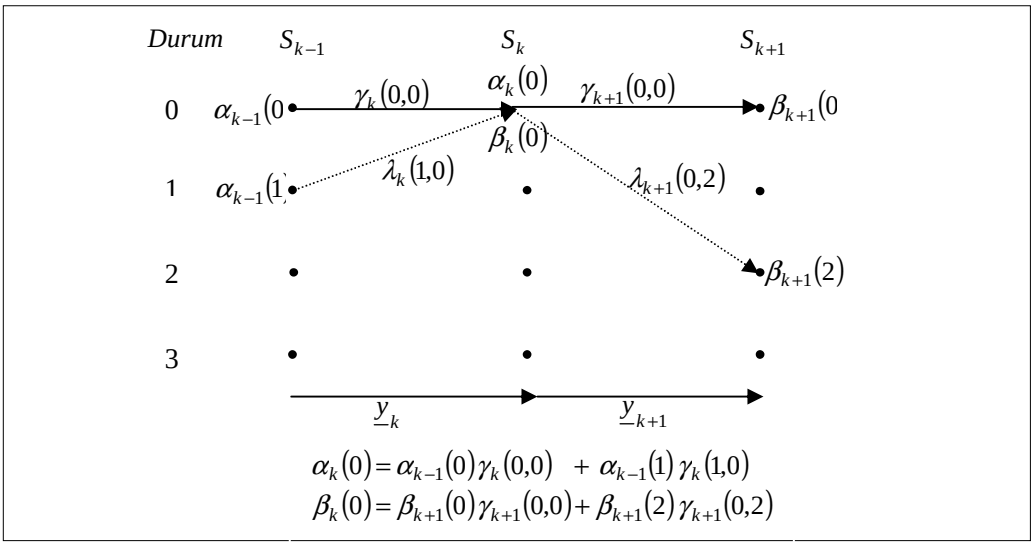
$$\begin{aligned}
\beta_{k-1}(s') &= P(\underline{y}_{j>k-1} \mid s') \\
&= \sum_{tüm\ s} P(\{\underline{y}_{j>k-1}, s\} \mid s') \\
&= \sum_{tüm\ s} P(\{\underline{y}_k, \underline{y}_{j>k}, s\} \mid s') \\
&= \sum_{tüm\ s} P(\underline{y}_{j>k} \mid \{s', s, \underline{y}_k\}).P(\{\underline{y}_k, s\} \mid s') \\
&= \sum_{tüm\ s} P(\underline{y}_{j>k} \mid s).P(\{\underline{y}_k, s\} \mid s') \\
&= \sum_{tüm\ s} \beta_k(s).\gamma_k(s', s)
\end{aligned} \tag{2.49}$$

(2.49) ifadesi incelendiğinde $\beta_{k-1}(s')$ değerlerinin hesaplanabilmesi için yine ilk olarak $\gamma_k(s',s)$ değerlerinin bilinmesi gerekmektedir. $\gamma_k(s',s)$ değerleri elde edildikten sonra $\beta_{k-1}(s')$ değerleri iteratif olarak hesaplanmaktadır. Kafes diyagramının sonlandırıldığı kabul edilirse $\beta_k(s)$ teriminin başlangıç değerleri

$$\begin{aligned}
&\beta_N(S_N = 0) = 1 \\
&\beta_N(S_N = s) = 0, \qquad s \neq 0 \text{ için}
\end{aligned} \tag{2.50}$$

şeklinde olmaktadır.

Şekil 2.17' de $\alpha_k(0)$ ve $\beta_k(0)$ ' ın iteratif olarak nasıl hesaplandığı gösterilmiştir.



Şekil 2.17. $\alpha_k(0)$ ve $\beta_k(0)$' ın iteratif olarak hesaplanması

2.2.10. $\gamma_k(s',s)$ Değerlerinin Hesaplanması

(2.44) ifadesinden ve Bayes kuralından yaralanılarak $\gamma_k(s',s)$ terimi

$$\begin{aligned}\gamma_k(s',s) &= P(\{\underline{y}_k, s\} \mid s') \\ &= P(\underline{y}_k \mid \{s', s\}).P(s \mid s') \\ &= P(\underline{y}_k \mid \{s', s\}).P(u_k)\end{aligned} \tag{2.51}$$

şeklinde ifade edilebilmektedir. (2.51) ifadesinde sonucun nasıl elde edildiği aşağıda gösterilmektedir.

$$\begin{aligned}P(\underline{y}_k, s \mid s') &= \frac{P(\underline{y}_k, s, s')}{P(s')} \\ &= \frac{P(\underline{y}_k \mid s, s').\mathrm{P}(s, s')}{P(s')} \\ &= \frac{P(\underline{y}_k \mid s, s').\mathrm{P}(s \mid s').\mathrm{P}(s')}{P(s')} \\ &= P(y_k \mid s, s').\mathrm{P}(s \mid s') \\ &= P(y_k \mid s, s').\mathrm{P}(u_k)\end{aligned}$$

Burada u_k, $S_{k-1}=s'$ durumundan $S_k=s$ durumuna geçişe neden olan giriş bitini, $P(u_k)$ ise bu bitin önsel (a-priori) olasılığını göstermektedir.

Eşitlik (2.25)' den

$$P(u_k) = \left(\frac{e^{-L(u_k)/2}}{1+e^{-L(u_k)}} \right) . e^{(u_k L(u_k)/2)} = C^{(1)}_{L(u_k)} . e^{(u_k L(u_k)/2)} \quad (2.52)$$

olarak yazılmaktadır. (2.52) eşitliğinden görüldüğü gibi

$$C^{(1)}_{L(u_k)} = \left(\frac{e^{-L(u_k)/2}}{1+e^{-L(u_k)}} \right) \quad (2.53)$$

ifadesi sadece LLR ($L(u_k)$) değerine bağlıdır. u_k bitine bağlı değildir.

(2.51) ifadesinde yer alan $P\left(\underline{y}_k \mid \{s', s\}\right)$ terimi $P\left(\underline{y}_k \mid \underline{x}_k\right)$ terimine eşdeğerdir. Burada $\underline{x}_k$ $S_{k-1}=s'$ durumundan $S_k=s$ durumuna geçişle ilişkili olan iletilen kod sözcüğüne karşılık gelmektedir.

İletim yapılan kanalın hafızasız olduğu kabul edilirse

$$P\left(\underline{y}_k \mid \{s', s\}\right) \equiv P\left(\underline{y}_k \mid \underline{x}_k\right) = \prod_{l=1}^{n} P\left(y_{kl} \mid x_{kl}\right) \quad (2.54)$$

eşitliği yazılabilmektedir.

Burada x_{kl} ve y_{kl} sırasıyla iletilen ve alınan kod sözcüklerindeki ($\underline{x}_k$ ve $\underline{y}_k$) her bir bite karşılık gelmektedir.

İletilen bitlerin (x_{kl}) BPSK modülasyonu yapılarak Gaussian kanal üzerinden iletildiği kabul edilirse (bu durumda iletilen semboller +1 ve -1 olacaktır) alınan sembollerin genlik dağılımları

$$P\left(y_{kl} \mid x_{kl}\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{E_b}{2\sigma^2} \left(y_{kl} - a x_{kl}\right)^2 \right) \quad (2.55)$$

şeklinde olur. Burada E_b iletilen her bitin enerjisini, σ^2 gürültünün varyansını ve a bayılma genliğini göstermektedir.

(2.55) eşitliği (2.54) eşitliğinde yerine koyulursa

$$
\begin{aligned}
P\left(\underline{y}_k \mid \{s', s\}\right) &= \prod_{l=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{E_b}{2\sigma^2} \left(y_{kl} - a x_{kl}\right)^2 \right) \\
&= \frac{1}{\left(\sqrt{2\pi}\sigma\right)^n} \exp\left(-\frac{E_b}{2\sigma^2} \sum_{l=1}^{n} \left(y_{kl} - a x_{kl}\right)^2 \right) \\
&= \frac{1}{\left(\sqrt{2\pi}\sigma\right)^n} \exp\left(-\frac{E_b}{2\sigma^2} \sum_{l=1}^{n} \left(y_{kl}^2 + a^2 x_{kl}^2 - 2 a x_{kl} y_{kl}\right) \right) \\
&= C_{\underline{y}_k}^{(2)} \cdot C_{\underline{x}_k}^{(3)} \cdot exp\left(\frac{E_b}{2\sigma^2} 2a \sum_{l=1}^{n} y_{kl} x_{kl} \right)
\end{aligned}
\tag{2.56}
$$

$$
C_{\underline{y}_k}^{(2)} = \frac{1}{\left(\sqrt{2\pi}\sigma\right)^n} \; exp\left(-\frac{E_b}{2\sigma^2} \sum_{l=1}^{n} y_{kl}^2 \right) \tag{2.57}
$$

elde edilir. (2.57) ifadesi sadece SNR ve alınan dizinin $\underline{y}_k$ genliğine bağlıdır.

$$
\begin{aligned}
C_{\underline{x}_k}^{(3)} &= \exp\left(-\frac{E_b}{2\sigma^2} a^2 \sum_{l=1}^{n} x_{kl}^2 \right) \\
&= exp\left(-\frac{E_b}{2\sigma^2} a^2 n \right)
\end{aligned}
\tag{2.58}
$$

ifadesi ise sadece SNR ve bayılma genliğine bağlıdır.

$\gamma_k\left(s', s\right)$ terimi (2.52) ve (2.56) eşitlikleri kullanılarak yeniden yazılırsa

$$
\begin{aligned}
\gamma_k\left(s', s\right) &= P\left(u_k\right) . P\left(\underline{y}_k \mid \{s', s\}\right) \\
&= C \cdot e^{\left(u_k L\left(u_k\right)/2\right)} \cdot exp\left(\frac{E_b}{2\sigma^2} 2a \sum_{l=1}^{n} y_{kl} x_{kl} \right) \\
&= C \cdot e^{\left(u_k L\left(u_k\right)/2\right)} \cdot exp\left(\frac{L_c}{2} \sum_{l=1}^{n} y_{kl} x_{kl} \right)
\end{aligned}
\tag{2.59}
$$

elde edilmektedir. Bu eşitlikte yer alan

$$
C = C_{L\left(u_k\right)}^{(1)} \cdot C_{\underline{y}_k}^{(2)} \cdot C_{\underline{x}_k}^{(3)} \tag{2.60}
$$

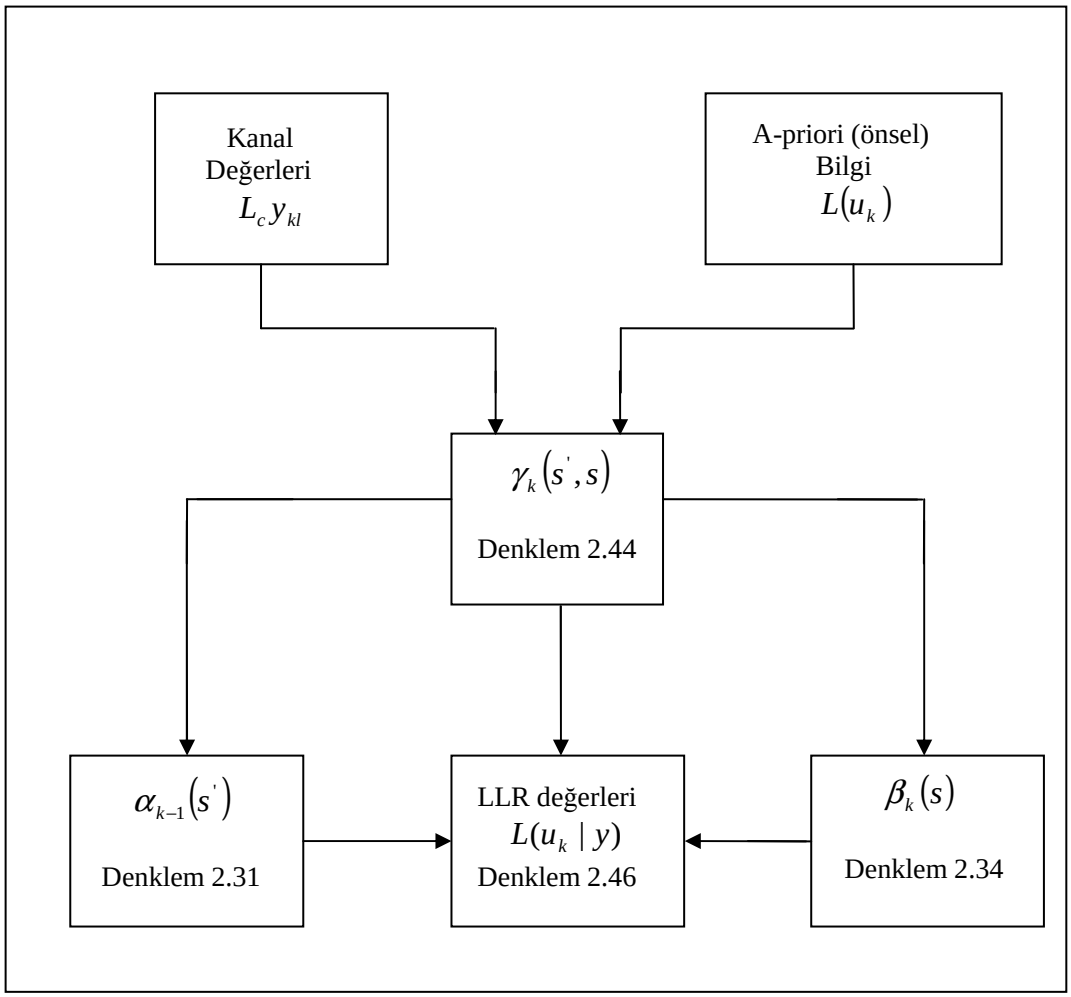
terimi iletilen kod sözcüğüne ve bitin (u_k) işaretine bağlı değildir. (2.38) eşitliğinden görüldüğü gibi bu terimin sonuç üzerinde etkisi olmayacaktır ve bu nedenle ihmal

edilmektedir. (2.38) ve (2.39) ifadelerinden kanal çıkışından alınan $\underline{y}_k$ dizisine bağlı olarak u_k bitinin LLR ifadesi

$$L\left(u_k \mid \underline{y}\right) = \ln\left(\frac{\sum_{\substack{(s',s)\Rightarrow \\ u_k=+1}} P\left(S_{k-1}=s',S_k=s,\underline{y}\right)}{\sum_{\substack{(s',s)\Rightarrow \\ u_k=-1}} P\left(S_{k-1}=s',S_k=s,\underline{y}\right)}\right) = \ln\left(\frac{\sum_{\substack{(s',s)\Rightarrow \\ u_k=+1}} \alpha_{k-1}\left(s'\right).\lambda_k\left(s',s\right).\beta_k\left(s\right)}{\sum_{\substack{(s',s)\Rightarrow \\ u_k=-1}} \alpha_{k-1}\left(s'\right).\lambda_k\left(s',s\right).\beta_k\left(s\right)}\right) \tag{2.61}$$

olarak yazılmaktadır.

Aşağıda MAP algoritmasında kullanılan temel işlemler bloklar halinde verilmektedir.



Şekil 2.18. MAP algoritmasındaki temel işlemler

2.2.11. Turbo Kodlarda İteratif Kod Çözme

Bu bölümde MAP algoritmasının turbo kodların iteratif kod çözümünde nasıl kullanıldığı açıklanmaktadır.

$\gamma_k(s',s)$ terimi için verilen (2.59) eşitliği yeniden düzenlenecektir.

$$\gamma_k(s',s) = C \cdot e^{(u_k L(u_k)/2)} \cdot exp\left(\frac{L_c}{2}\sum_{l=1}^{n} y_{kl} x_{kl}\right) \tag{2.62}$$

Şekil 2.6 da verilen genel turbo kodlayıcının yapısından görüldüğü gibi, turbo kodlar sistematik bir yapıya sahip olduğundan iletilen *n* (her bir kod sözcüğündeki bit sayısı) bitten bir tanesi sistematik bit u_k olmaktadır. Sistematik bitin iletilen bitlerden ilk bit olduğu kabul edilirse $x_{k1} = u_k$ olur. Bu durumda (2.52) eşitliği yeniden yazılırsa

$$\begin{aligned}\gamma_k\left(s^{'},s\right)&=C.e^{\left(u_kL(u_k)/2\right)}.exp\left(\frac{L_c}{2}y_{ks}u_k\right).exp\left(\frac{L_c}{2}\sum_{l=2}^{n}y_{kl}x_{kl}\right)\\&=C.e^{\left(u_kL(u_k)/2\right)}.exp\left(\frac{L_c}{2}y_{ks}u_k\right).\chi_k\left(s^{'},s\right)\end{aligned}\qquad(2.63)$$

ifadesi elde edilmektedir. Burada yer alan y_{ks} terimi iletilen sistematik bit $x_{k1}= u_k$'nın kanal çıkışındaki karşılığına denk gelmektedir.

$$\chi_k\left(s^{'},s\right)=\exp\left(\frac{L_c}{2}\sum_{l=2}^{n}y_{kl}x_{kl}\right)\qquad(2.64)$$

(2.63) ifadesinden yararlanılarak (2.61) ifadesi yeniden düzenlenirse

$$\begin{aligned}L\left(u\ \middle|\ \underline{y}\right)&=\ln\left(\frac{\sum_{\substack{\left(s^{'},s\right)\Rightarrow\\u_k=+1}}\alpha_{k-1}\left(s^{'}\right).\gamma_k\left(s^{'},s\right).\beta_k(s)}{\sum_{\substack{\left(s^{'},s\right)\Rightarrow\\u_k=-1}}\alpha_{k-1}\left(s^{'}\right).\gamma_k\left(s^{'},s\right).\beta_k(s)}\right)\\&=\ln\left(\frac{\sum_{\substack{\left(s^{'},s\right)\Rightarrow\\u_k=+1}}\alpha_{k-1}\left(s^{'}\right).e^{+L(u_k)/2}.e^{+L_cy_{ks}/2}.\chi_k\left(s^{'},s\right).\beta_k(s)}{\sum_{\substack{\left(s^{'},s\right)\Rightarrow\\u_k=-1}}\alpha_{k-1}\left(s^{'}\right).e^{-L(u_k)/2}.e^{-L_cy_{ks}/2}.\chi_k\left(s^{'},s\right).\beta_k(s)}\right)\\&=L\left(u_k\right)+L_cy_{ks}+ln\left(\frac{\sum_{\substack{\left(s^{'},s\right)\Rightarrow\\u_k=+1}}\alpha_{k-1}\left(s^{'}\right).\chi_k\left(s^{'},s\right).\beta_k(s)}{\sum_{\substack{\left(s^{'},s\right)\Rightarrow\\u_k=-1}}\alpha_{k-1}\left(s^{'}\right).\chi_k\left(s^{'},s\right).\beta_k(s)}\right)\\&=L\left(u_k\right)+L_cy_{ks}+L_e\left(u_k\right)\end{aligned}\qquad(2.65)$$

elde edilir ve burada $L_e(u_k)$ aşağıda (2.66) eşitliği ile verilen ifadeye eşittir.

$$L_e\left(u_k\right)=\ln\left(\frac{\sum_{\substack{\left(s^{'},s\right)\Rightarrow\\u_k=+1}}\alpha_{k-1}\left(s^{'}\right).\chi_k\left(s^{'},s\right).\beta_k(s)}{\sum_{\substack{\left(s^{'},s\right)\Rightarrow\\u_k=-1}}\alpha_{k-1}\left(s^{'}\right).\chi_k\left(s^{'},s\right).\beta_k(s)}\right)\qquad(2.66)$$

Eşitlik (2.65)' den görüldüğü gibi u_k biti için hesaplanan LLR ($L(u_k \mid \underline{y})$) değeri üç terimden oluşmaktadır: $L(u_k)$, $L_c y_{ks}$ ve $L_e(u_k)$.

$L(u_k)$ terimi, u_k bitine ait olan önsel (a-priori) olasılık bilgisine karşılık gelmekte ve bağımsız bir kaynak, diğer bileşen kod çözücü tarafından üretilmektedir. u_k biti hakkında böyle bir bilgi yoksa $L(u_k)$ bilgisi "0" olmaktadır.

Bunun anlamı kaynak bitlerinin olasılıklarının eşit olduğudur.

$$P(u_k = \pm 1) = 0.5 \tag{2.67}$$

$L_c y_{ks}$ terimi kanal çıkışındaki sistematik bite ait olan yumuşak (soft) bilgiye karşılık gelmektedir. (2.31) ve (2.65) eşitliklerinden kanalın SNR' si yüksek olduğunda $L_c y_{ks}$ teriminin $L(u_k \mid \underline{y})$ değeri üzerinde daha büyük bir etkiye sahip olduğu görülmektedir. Benzer şekilde kanalın SNR' si düşük olduğunda daha düşük bir etkiye sahip olmaktadır.

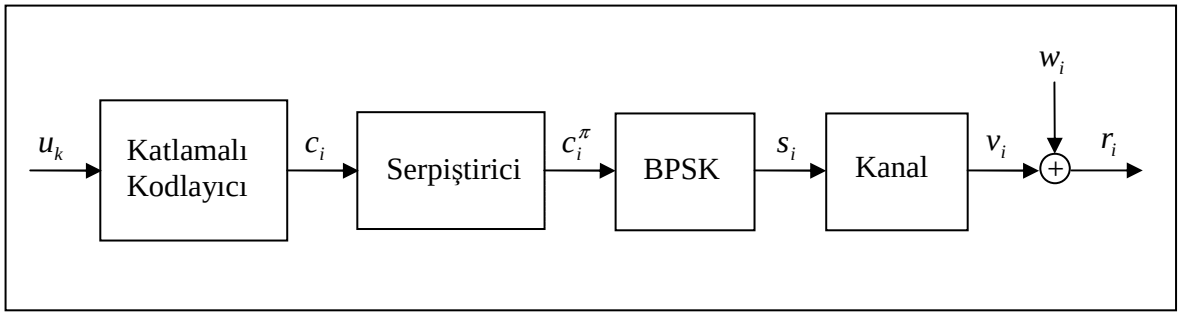
$L_e(u_k)$ terimi ise kanal çıkışından alınan eşlik bitlerinden elde edilen yan bilgidir ve u_k biti için extrinsic bilgi olarak adlandırılmaktadır. (2.65) eşitliğinden

$$L_e(u_k) = L(u \mid \underline{y}) - L(u_k) + L_c y_{ks} \tag{2.68}$$

elde edilmektedir. (2.68) eşitliğinden görüldüğü gibi bu bilgi MAP kod çözücü çıkışında elde edilen $L(u_k \mid \underline{y})$ değerinden u_k biti hakkında ön bilgiyi gösteren $L(u_k)$ ve kanal çıkışındaki sistematik veriye karşılık gelen $L_c y_{ks}$ değerlerinin çıkartılmasıyla elde edilmektedir.

2.3. Turbo Denkleştiriciler

Bir önceki bölümde açıklanan turbo kodların kod çözümünde kullanılan prensipler sayısal haberleşme alanında farklı problemlerin çözümünde kullanılabilmektedir. Bu bölümde haberleşme kanalı olarak frekans seçici kanalların olduğu kabul edilerek alıcıda, denkleştirme ve kod çözme işlemi birleştirilerek analizi yapılmıştır. Bu alıcı yapısı turbo denkleştirici olarak adlandırılır. Turbo denkleştiriciler ilk olarak 1995 yılında Douillard tarafından Şekil 2.19' de verilen iletim şemasına göre önerilmiştir[4].

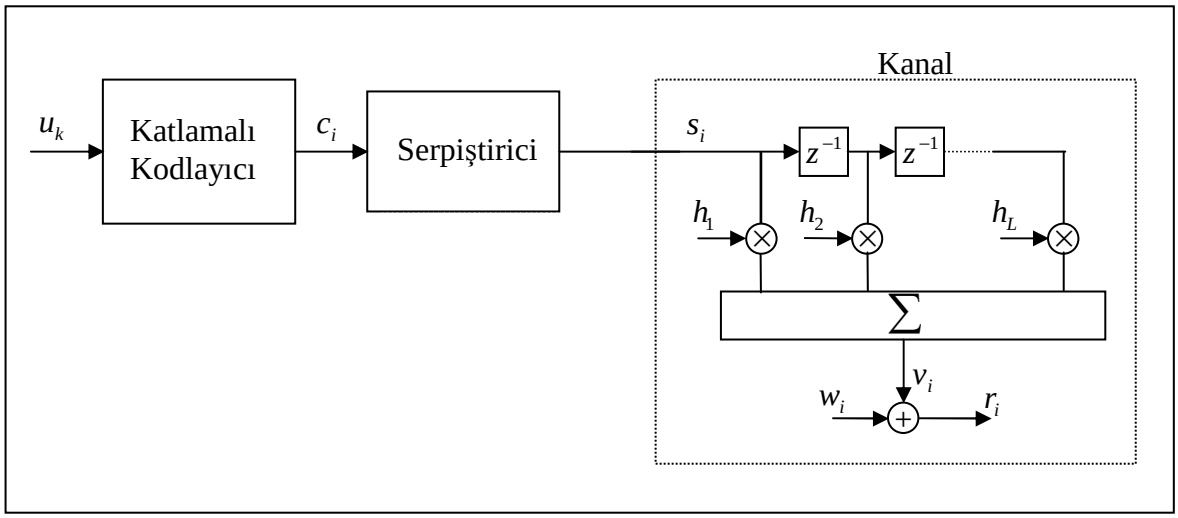


Şekil 2.19. İletim şeması

Şekil 2.19' daki iletim şemasından görüldüğü gibi, sistem seri birleştirilmiş hata kodlayıcı, serpiştirici ve frekans seçici kanaldan oluşmaktadır. Hata kodlayıcı olarak *r=1/n* olan katlamalı kodlayıcı olduğu kabul edilmiştir. Burada, u_k (k=1,…,N) ikili bilgi bitlerini, c_i (i=1,…,n×N) kodlayıcı çıkışını gösterir. Kodlayıcı çıkışında elde edilen kod sözcükleri serpiştirildikten sonra BPSK modülasyonu uygulanarak elde edilen s_i (i=1,…,n×N) sembolleri kanala verilir. s_i sembol değerleri c_i^{π} değerlerinden aşağıda verilen eşitlik kullanılarak elde edilir.

$$s_i = 2 \times c_i^{\pi} - 1 \tag{3.1}$$

Kanalın ayrık modeli dikkate alınarak Şekil 2.19 da verilen iletim şeması Şekil 2.20 de görüldüğü gibi olmaktadır.

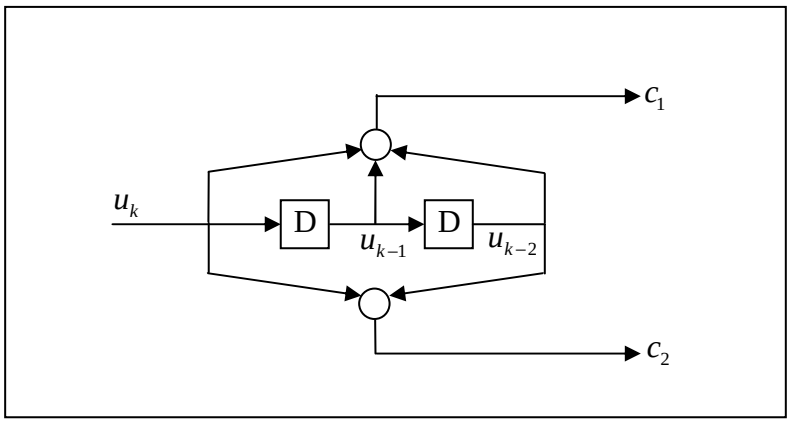


Şekli 2.20. Kanalın ayrık modeli dikkate alınarak oluşturulan iletim şeması

Şekil 2.20’ de görüldüğü gibi kanal ayrık olarak modellenmiş ve L (kanalın tap sayısı) tane katsayıya sahiptir. Kanalın çıkışı

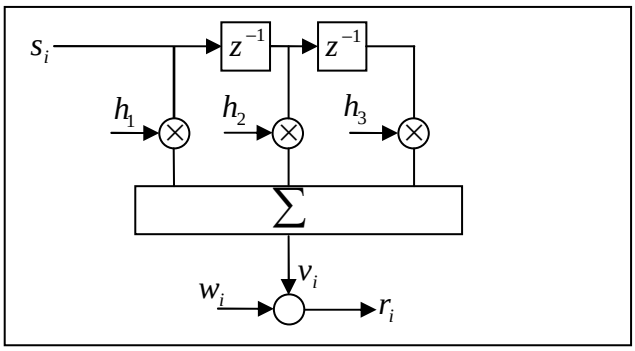
$$r_i = v_i + w_i$$
$$r_i = \sum_{j=0}^{L-1} h_j s_{i-j} + w_i \qquad (3.2)$$

olarak elde edilmektedir.

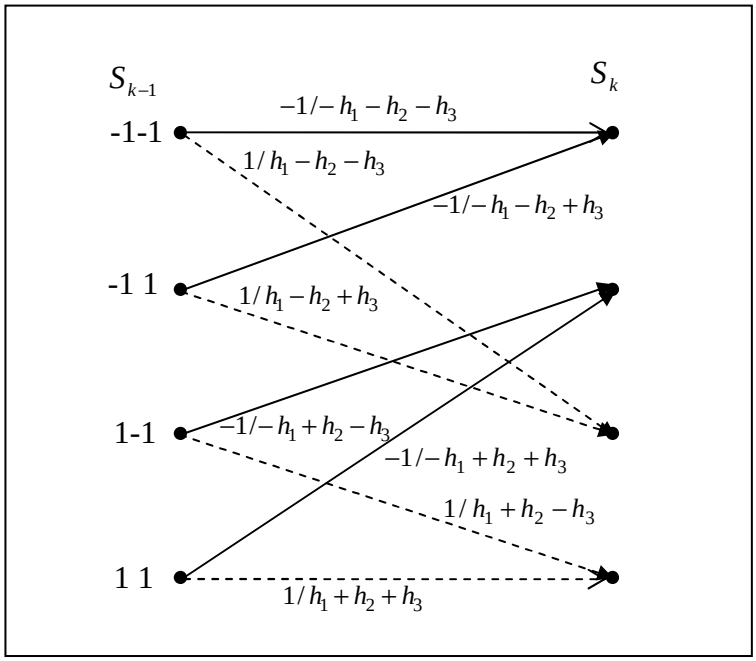


Şekil 2.21. Katlamalı kodlayıcı

Şekil 2.20’ de verilen kanal modeli ve Şekil 2.21’ de verilen katlamalı kodlayıcı incelendiğinde kanal kod oranı $R = 1$ olan bir kodlayıcı olarak düşünülebilmektedir [29],[30]. Bundan dolayı Şekil 2.20’ de verilen iletim şeması seri birleştirilmiş turbo kodlayıcı olarak düşünülebilmektedir. Kanalın ayrık modeli kullanılarak Şekil 2.22’ de verilen 3 taplı kanalın kafes diyagramı Şekil 2.23’ de görülmektedir.

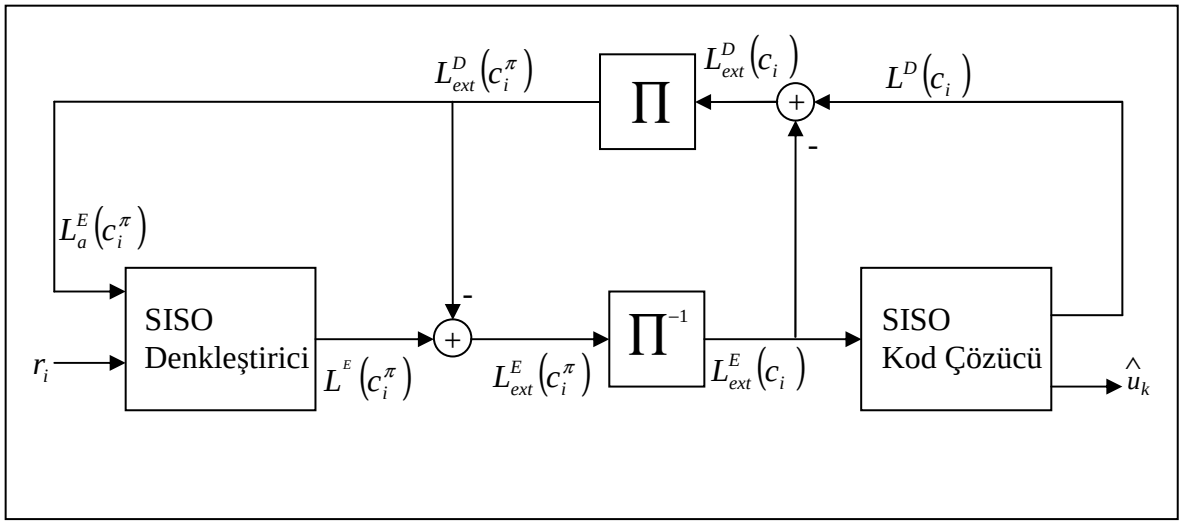


Şekli 2.22. 3 taplı kanalın ayrık modeli



Şekil 2.23. 3 taplı kanalın kafes diyagramı

Şekil 2.23 da yer alan ifadelerden örnek $-1/-h_1-h_2-h_3$ olarak ifadesi incelenirse; -1 kanala giren sembolü $-h_1-h_2-h_3$ ise kanal çıkışını göstermektedir. Şekil 2.20' de verilen iletim şeması için iteratif turbo denkleştirici yapısı Şekil 2.24' da görülmektedir. Şekilden görüldüğü gibi turbo kodların kod çözümünden farklı olarak burada SISO kod çözücü kanal çıkışındaki yumuşak (soft) veriyi giriş olarak almaz. İkinci bir faklılık ise kod çözücüde sadece u_k bilgi bitleri için değil, eşlik bitleri için de sonsal (a-posteriori) olasılık değerleri hesaplanmaktadır. Hesaplanan bu değerler denkleştiriciye aktarılan ön (a-priori) bilginin elde edilmesinde kullanılmaktadır. Kod çözücü sadece Şekli 2.24' de görüldüğü gibi denkleştiriciden gelen ön (a-priori) bilgiyi giriş olarak almaktadır. Şekil 2.24'de görüldüğü gibi turbo denkleştirici temel olarak iki bloktan oluşmaktadır. SISO denkleştirici ve SISO kod çözücü. Aşağıda bu iki bloğun analizi yapılmıştır.



Şekil 2.24. Turbo denkleştirici

2.3.1. SISO Denkleştirici

SISO denkleştirici kanal çıkışındaki yumuşak (soft) veriyi ve kod çözücü tarafından kod sözcükleri için elde edilen ön (a-priori) bilgiyi kullanarak c_i^{π} kod sözcük değerleri için MAP algoritmasını kullanarak LLR ($L^{E}(c_i^{\pi})$) değerlerini hesaplar. MAP algoritması bu bölümde tekrar incelenmeyecektir.

$$L^{E}(c_i^{\pi}) = \ln\left(\frac{P(c_i^{\pi}=1\,|\,\underline{r})}{P(c_i^{\pi}=0\,|\,\underline{r})}\right) = \ln\left(\frac{P(s_i=+1\,|\,\underline{r})}{P(s_i=-1\,|\,\underline{r})}\right) \tag{3.3}$$

Bayes kuralı (Denklem 2.17) kullanılarak (3.3) eşitliği

$$\begin{aligned} L^{E}(c_i^{\pi}) &= \ln\left(\frac{P(\underline{r}\,|\,c_i^{\pi}=1)}{P(\underline{r}\,|\,c_i^{\pi}=0)}\right) + \ln\left(\frac{P(c_i^{\pi}=1)}{P(c_i^{\pi}=0)}\right) \\ &\equiv L_e^{E}(c_i^{\pi}) + L_a^{E}(c_i^{\pi}) \end{aligned} \tag{3.4}$$

şeklinde ifade edilebilmektedir.

Bu denklemden de görüldüğü gibi kod çözücünün girişi, yani kod sembolleri hakkındaki önsel (a-priori) bilgi, SISO denkleştirici çıkışındaki yumuşak(soft) bilgiden SISO denkleştirici girişindeki önsel(a-priori) bilginin çıkartılmasıyla elde edilmektedir.

$$L_e^E\left(c_i^\pi\right)=L^E\left(c_i^\pi\right)-L_a^E\left(c_i^\pi\right) \tag{3.5}$$

Kanalın durumlar arası geçiş ifadesi ($\gamma_k(s',s)$) 2.36 eşitliğinden yararlanılarak

$$\begin{aligned}\gamma_i\left(s',s\right)&=P\left(\{r_i,s\}\mid s'\right)\\&=P\left(r_i\mid\{s',s\}\right).P\left(s\mid s'\right)\\&=P\left(r_i\mid\{s',s\}\right).P(c_i^\pi)\end{aligned} \tag{3.6}$$

olarak elde edilmektedir. (3.6) eşitliğinde yer alan $P(r_i|\{s',s\})$ifadesi kanal çıkışının (v_i) üzerine gaussian gürültü eklendiğinde elde edilen işaretin genlik dağılımını ($P(r_i|v_i)$) göstermektedir. Bağımsız iki olayın toplamının sonucunda elde edilen işaretin olasılık yoğunluk fonksiyonu, bu iki işaretin olasılık yoğunluk fonksiyonlarının katlanmasıyla elde edildiğinden[31] kanalın geçiş ifadesi

$$\begin{aligned}\gamma\left(s',s\right)&=P\left(c_i^\pi\right)P\left(r_i\mid v_i\right)\\&=P\left(c_i^\pi\right)\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{E_b}{2\sigma^2}\left(r_i-v_i\right)^2\right)\end{aligned} \tag{3.7}$$

olarak elde edilmektedir. Elde edilen eşitlikte (2.36) denkleminde yer alan ve bayılma genliğini olarak adlandırılan "a" katsayısı "1" (AWGN kanal) olarak kabul edilmiştir.
(3.5) ifadesinde yer alan $P\left(c_i^\pi\right)$ teriminin değerleri ise kod çözücü tarafından sağlanan bilgiden elde edilmektedir. Şekil 3.6' da verilen yapı ve (2.5) eşitliği dikkate alınarak $P\left(c_i^\pi\right)$ değerleri aşağıda belirtilen şekilde ifade edilir.

$$\begin{aligned}L_e^D\left(c_i^\pi\right)&=\ln\left(\frac{P\left(c_i^\pi=1\right)}{P\left(c_i^\pi=0\right)}\right)=\ln\left(\frac{P\left(s_i=+1\right)}{P\left(s_i=-1\right)}\right)\\&=\ln\left(\frac{P\left(c_i^\pi=1\right)}{1-P\left(c_i^\pi=0\right)}\right)\end{aligned} \tag{3.8}$$

(3.8) eşitliğinden

$$P\left(c_i^\pi=1\right)=\frac{\exp\left(L_e^D\left(c_i^\pi\right)\right)}{1+\exp\left(L_e^D\left(c_i^\pi\right)\right)} \tag{3.9}$$

elde edilmektedir. Benzer şekilde

$$L_e^D\left(c_i^\pi\right)=\ln\left(\frac{P\left(c_i^\pi=1\right)}{P\left(c_i^\pi=0\right)}\right)=\ln\left(\frac{1-P\left(c_i^\pi=-1\right)}{P\left(c_i^\pi=-1\right)}\right) \tag{3.10}$$

eşitliğinden

$$P\left(c_n^\pi=0\right)=\frac{1}{1+\exp\left(L_e^D\left(c_i^\pi\right)\right)} \tag{3.11}$$

olarak elde edilmektedir. (3.9) ve (3.11) eşitliklerinin birleştirilmesiyle genel olarak $P\left(c_i^\pi\right)$ ifadesi aşağıdaki şekilde elde edilmektedir.

$$P\left(c_i^\pi=c\right)=\frac{\exp(c\,.\,L_e^D(c_i^\pi))}{1+\exp(L_e^D(c_i^\pi))} \qquad c\in\{0,1\} \tag{3.12}$$

SISO denkleştiricinin çıkış ifadesi aşağıda verilmiştir.

$$L\left(c_i^\pi\mid\underline{r}\right)=\ln\left(\frac{\sum_{\substack{(s',s)\Rightarrow\\ c_n^\pi=+1}}P\left(S_{n-1}=s',S_n=s,\underline{r}\right)}{\sum_{\substack{(s',s)\Rightarrow\\ c_n^\pi=-1}}P\left(S_{n-1}=s',S_n=s,\underline{r}\right)}\right)=\ln\left(\frac{\sum_{\substack{(s',s)\Rightarrow\\ c_n^\pi=+1}}\alpha_{n-1}\left(s'\right).\lambda_n\left(s',s\right).\beta_n\left(s\right)}{\sum_{\substack{(s',s)\Rightarrow\\ c_n^\pi=-1}}\alpha_{n-1}\left(s'\right).\lambda_n\left(s',s\right).\beta_n\left(s\right)}\right) \tag{3.13}$$

2.3.2. SISO Kod Çözücü

SISO kod çözücü Şekil 3.6'dan da görülüğü gibi giriş olarak sadece denkleştiriciden gelen yumuşak (soft) bilgiyi almakta ve bu bilgiden yararlanarak c_i kodlanmış bitler ve u_k giriş bitleri için LLR değerlerini üretmektedir. SISO kod çözücünün çıkış ifadesi

$$L^D(c_i) \equiv \ln\left(\frac{P(c_i = 1 \mid Z)}{P(c_i = 0 \mid Z)}\right) = L_e^D(c_i) + L_a^D(c_i) \tag{3.14}$$

olarak ifade edilmektedir. (3.14) eşitliğinde kullanılan "Z" teriminin açıklaması aşağıda yapılmıştır.

$$Z = [P(c_1 \mid r)P(c_2 \mid r)...P(c_N \mid r)]^T \tag{3.15}$$

(3.15) eşitliğindeki terimlerin değerleri denkleştiriciden gelen bilgiden aşağıda verilen eşitlikle elde edilir.

$$P(c_n = c \mid r) = \frac{\exp\left(c \,.\, L_{ext}^E(c_i)\right)}{1 + \exp\left(c \,.\, L_{ext}^E(c_i)\right)} \qquad c \in \{0,1\} \tag{3.16}$$

Kodlayıcının kafes diyagramında durumlar arası geçiş ifadesi (2.36) eşitliğinden ve Şekil 3.3'de verilen katlamalı kodlayıcının yapısından yararlanılarak

$$\gamma_n(s', s) = P(u_k)P(c_1 \mid r)P(c_2 \mid r) \tag{3.17}$$

olarak elde edilmektedir.

SISO kod çözücüden denkleştiriciye aktarılan bilgi (3.14) eşitliğinden yararlanılarak

$$L_e^D(c_i) = L^D(c_i) - L_a^D(c_i) \tag{3.18}$$

olarak elde edilmektedir.

SISO kod çözücünün çıkış ifadesi aşağıda verilmiştir.

$$L^D(c_i \mid \underline{r}) = \ln\left(\frac{\sum_{\substack{(s',s) \mapsto \\ c_n = +1}} \alpha_{n-1}(s').\lambda_n(s', s).\beta_n(s)}{\sum_{\substack{(s',s) \mapsto \\ c_n = -1}} \alpha_{n-1}(s').\lambda_n(s', s).\beta_n(s)}\right) \tag{3.19}$$

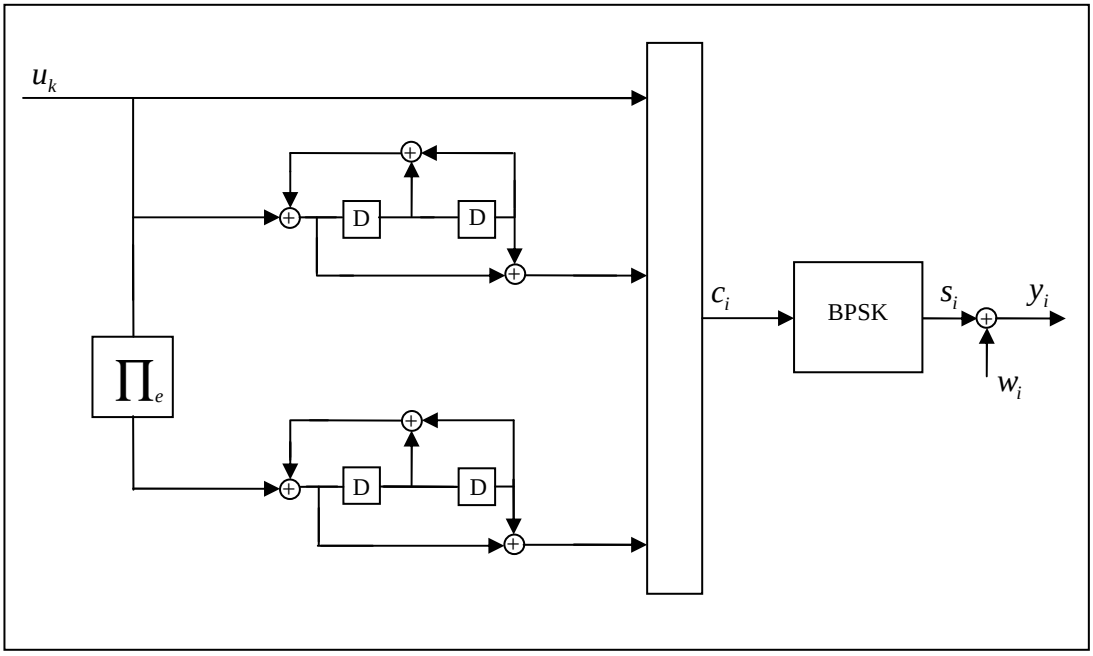
Burada SISO denkleştirici ve SISO kod çözücü çıkış ifadelerinde yer alan α ve β terimlerinin eşitlikleri ve nasıl hesaplandıkları bir önceki bölümde açıklandığından tekrar açıklanmamıştır.

3. BULGULAR VE TARTIŞMA

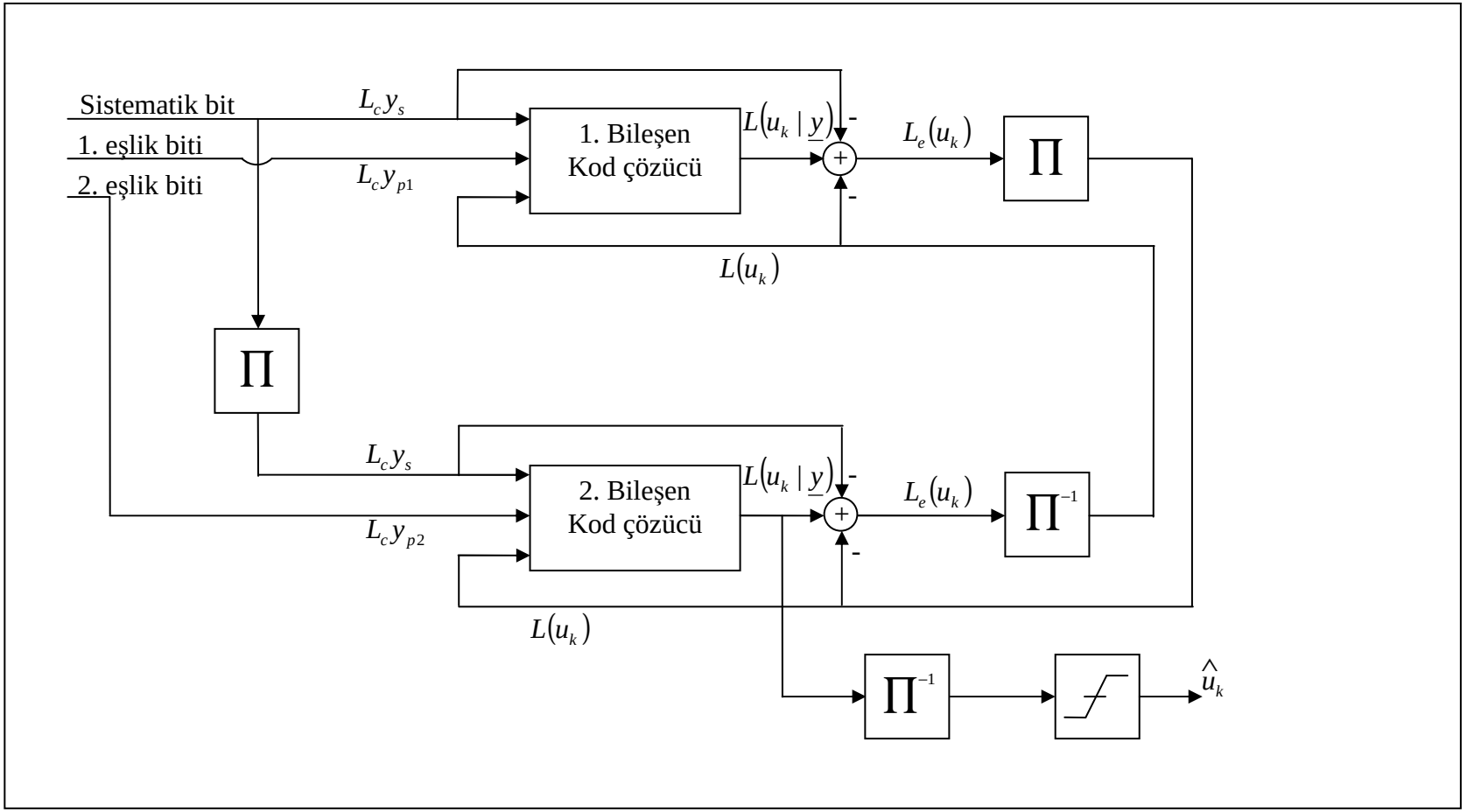
3.1. Turbo Kodların AWGN Kanallar Üzerinde Başarım Analizleri

Bu bölümde, ilk olarak Şekil 3.1'de verilen verici ve Şekil 3.2'de verilen alıcı yapısı kullanılarak Turbo kodların AWGN kanallar üzerinde başarım analizleri gerçekleştirilmiştir.

Simülasyonlarda, modülasyon tekniği olarak Şekil 3.1'den görüldüğü gibi BPSK, bileşen kodlayıcı olarak kod oranı R=1/2, sınır uzunluğu K=3 ve üreteç polinomu $G=(1,5/7)_8$ olan RSC kodlayıcı ve serpiştirici olarak rasgele serpiştirici kullanılmıştır. Turbo kodlayıcının kod oranı delme işlemi uygulanarak R=1/2 olarak seçilmiştir.



Şekil 3.1. AWGN kanal için Turbo verici yapısı



Şekil 3.2. Turbo alıcı yapısı

Simülasyon programları C++ programlama dili kullanılarak yazılmıştır. Simülasyon sonuçlarının grafikleri MATLAB programında elde edilmiştir. AWGN kanal için gauss gürültüsü aşağıda verilen fonksiyon kullanılarak elde edilmiştir.

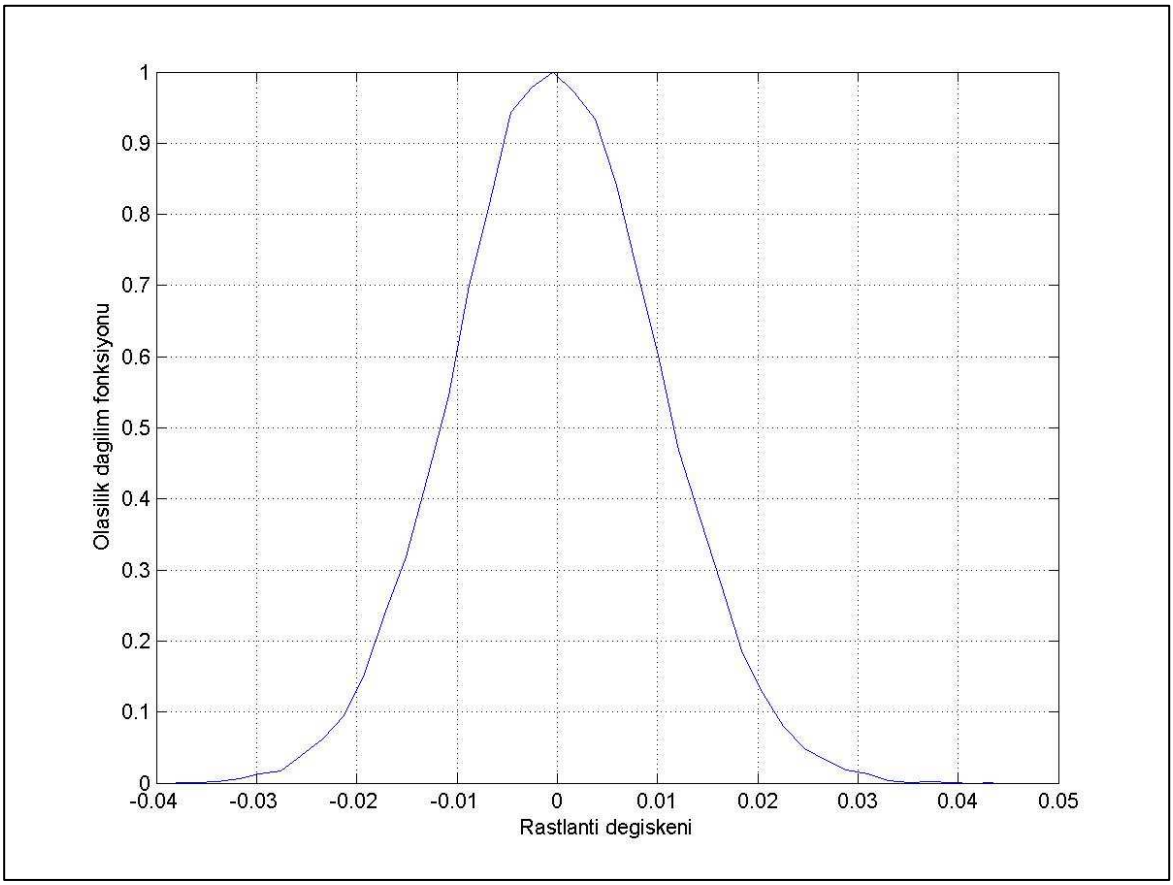
```
Matris<double> gauss(double sigma, double mean, int boyut)
{
    Matris<double> gurultu(1,boyut,0.0);
    double u1,u2,v1,v2,s;
    double temp1,temp2;
    int i;

    for(i=0; i<boyut; i+=2)
    {
        do
        {
            u1=static_cast<double>(rand())/RAND_MAX;
            u2=static_cast<double>(rand())/RAND_MAX;
            v1=2.*u1–1.0;
            v2=2.*u2–1.0;
            s=v1*v1+v2*v2;
        }while(s>=1.0);

        temp1=log(s);
        temp2=sqrt(-2.0*temp1/s);
        gurultu.set_value(0,i)=(v1*temp2)*sigma+mean;
        gurultu.set_value(0,i+1)=(v2*temp2)*sigma+mean;
    }
    return gurultu;
}
```

Burada fonksiyonun aldığı parametreler sırası ile aşağıda açıklanmıştır. sigma değişkeni, gürültünün standart sapmasına, mean değişkeni, gürültünün ortalamasına, boyut değişkeni ise gürültü vektörünün boyutuna karşılık gelir.

Verilen fonksiyon kullanılarak elde edilen, standart sapması 0.01 ve ortalaması 0 olan gauss gürültüsünün dağılımı Şekil 3.3' de verilmiştir.

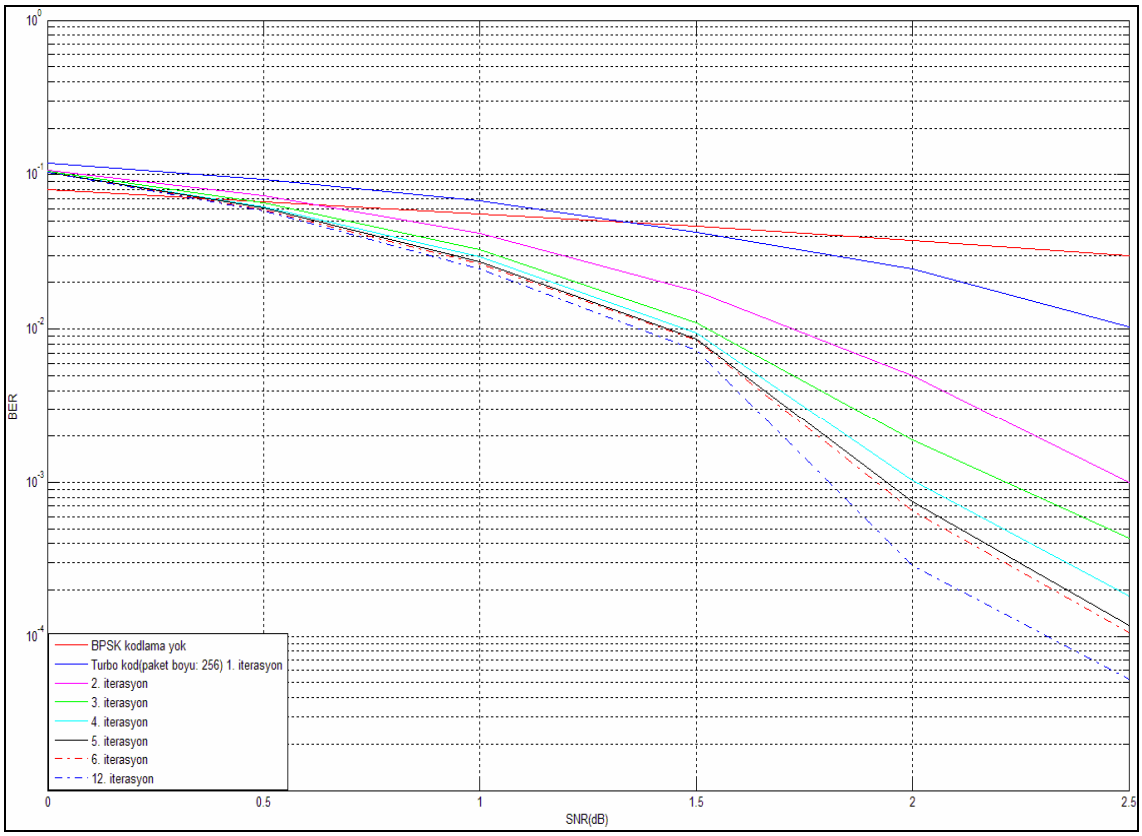


Şekil 3.3. Ortalaması "0" Standart sapması "0.01" olan Gauss dağılımı

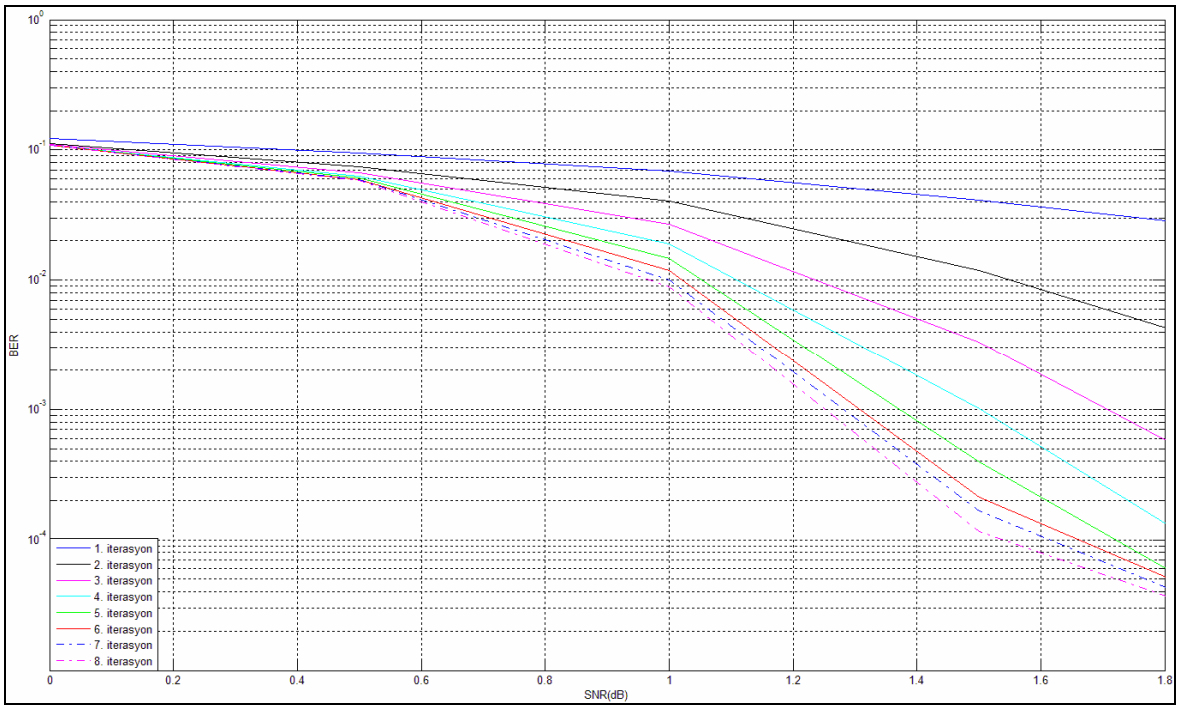
Elde edilen gauss dağılımının değerlerinden ortalaması ve standart sapması hesaplandığında ise aşağıdaki sonuçlar bulunmuştur. Vektörünün boyu 30000 olarak alınmıştır.

standart sapması(sigma) = 0.00998230631616
ortalaması(mean) = 5.880574289609997e-005

Aşağıda iterasyon sayısı ve paket boyu dikkate alınarak elde edilen simülasyon sonuçları verilmiştir[5].



Şekil 3.4. AWGN kanalda Turbo kodların ve BPSK modülasyonunun başarımı



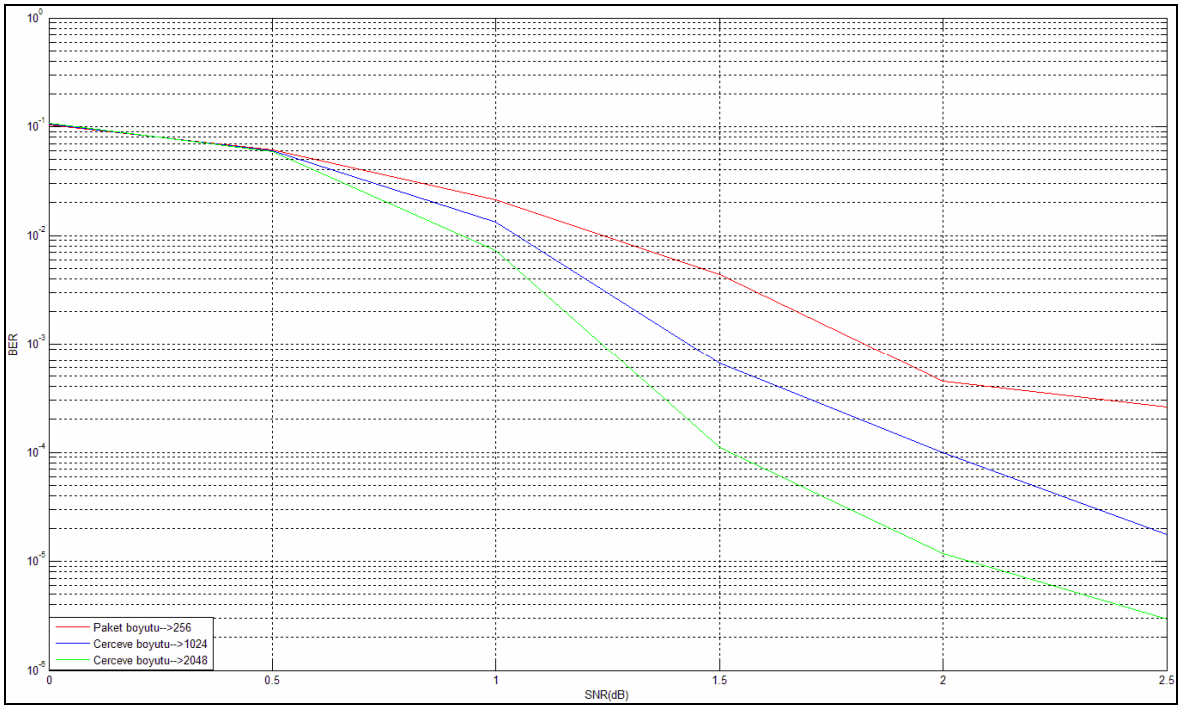
Şekil 3.5. Turbo kodlarda AWGN kanalda özyineleme sayısının etkisi (paket boyu=2048)

Turbo kodların yapısından bahsedilirken kod çözme işleminde temel noktanın bileşen kod çözücüler arasında taşınan yan bilginin(extrinsic information) olduğundan bahsedilmişti. Kod çözücüler arasında taşınan bu bilgi her bir iterasyon sonucunda düzeltilen hatalı bit miktarını artırır. Şekil 3.5' de verilen grafiklerden de görüldüğü gibi iterasyon sayısıyla birlikte performansta artış gözlenmektedir. Yüksek iterasyon sayılarına çıkıldığında elde edilen başarım azalmaktadır. İlk altı iterasyona dikkat edildiğinde performansta belirgin bir artış görülmektedir. Altıncı iterasyon sonrasında elde edilen performans, iterasyon kazancı azalmaktadır. Data blok halinde kod çözücülere uygulandığından iterasyon sayısının artması sistemde gecikmelere yol açmaktadır. Belirli iterasyon değerlerinden sonra ciddi bir performans artışı elde edilmediğinden genelde yaklaşık 8 iterasyon uygulanmaktadır.

Aşağıda MAP algoritmasının logaritmik domende karşılığı olan Log-MAP algoritmasının karmaşıklığı, kodlayıcının bellek sayısı dikkate alınarak verilmiştir[52].

İşlem	Toplama Sayısı
Branch Metric (Dal) Hesabı	12×2^{M}
Path Metrics – Forward(İleri yön)	9×2^{M}
Path Metrics – Backward (Geri yön)	9×2^{M}
Soft Decision – (Yumuşak karar)	$18\times2^{M}-13$
Toplam	$48\times2^{M}-13$
M=Kodlayıcı bellek sayısı	

Şekil 3.6. Log-MAP algoritmasının işlem karmaşıklığı

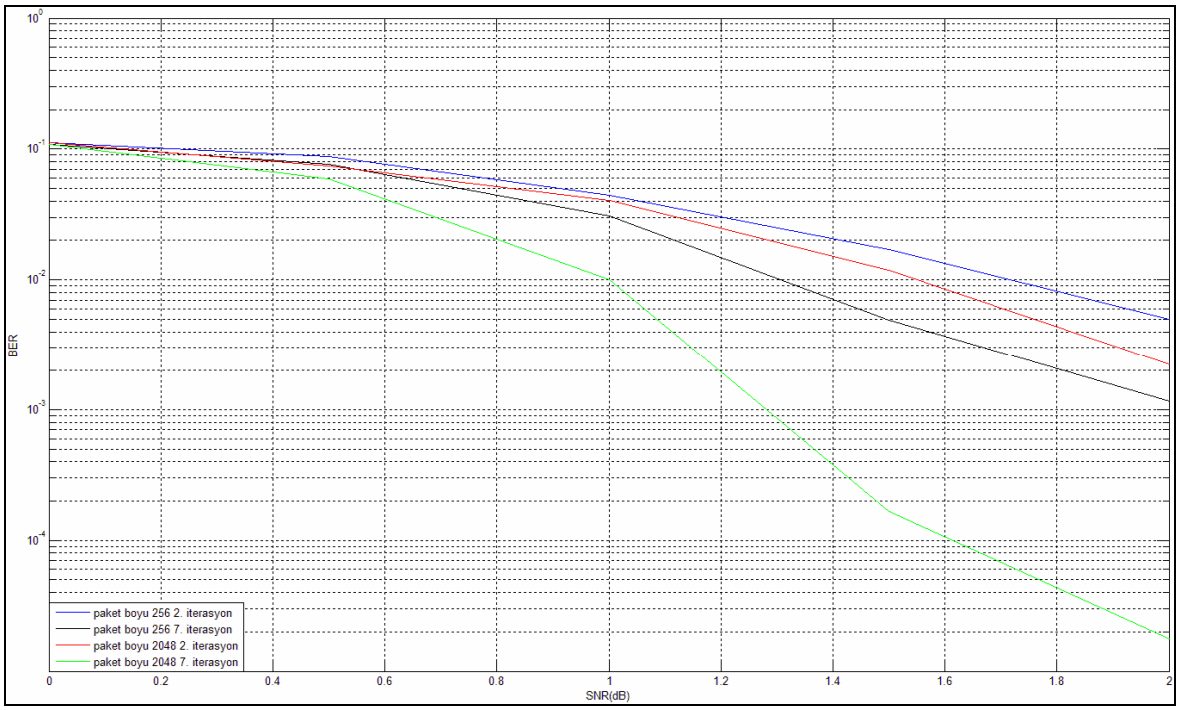


Şekil 3.7. Turbo kodlarda paket boyunun etkisi

Şekil 3.7'de verilen grafiklerden görüldüğü gibi paket boyunun artması performansı önemli ölçüde artırmaktadır. Rasgele serpiştiriciler kullanıldığında paket boyu performansı etkileyen önemli bir etkendir. Kodlama teorisinden rasgele(random) kodların en iyi kodlar olduğu bilinmektedir. Paket boyunun artırılması kodun rasgeleliğini artıracağından performans üzerinde önemli bir etkisi olmaktadır.

Grafikler incelendiğinde 2.5 dB işaret gürültü oranında paket boyu 256 iken elde edilen bit-hata oranı 2.2601×10^{-4}, paket boyu 1024 iken bit-hata oranı 1.763×10^{-5} ve paket

boyu 2048 iken elde edilen bit-hata oranı 2.935×10^{-6} olmaktadır. Bu sonuçlardan da görüldüğü gibi paket boyunun artması performansı artırmaktadır.



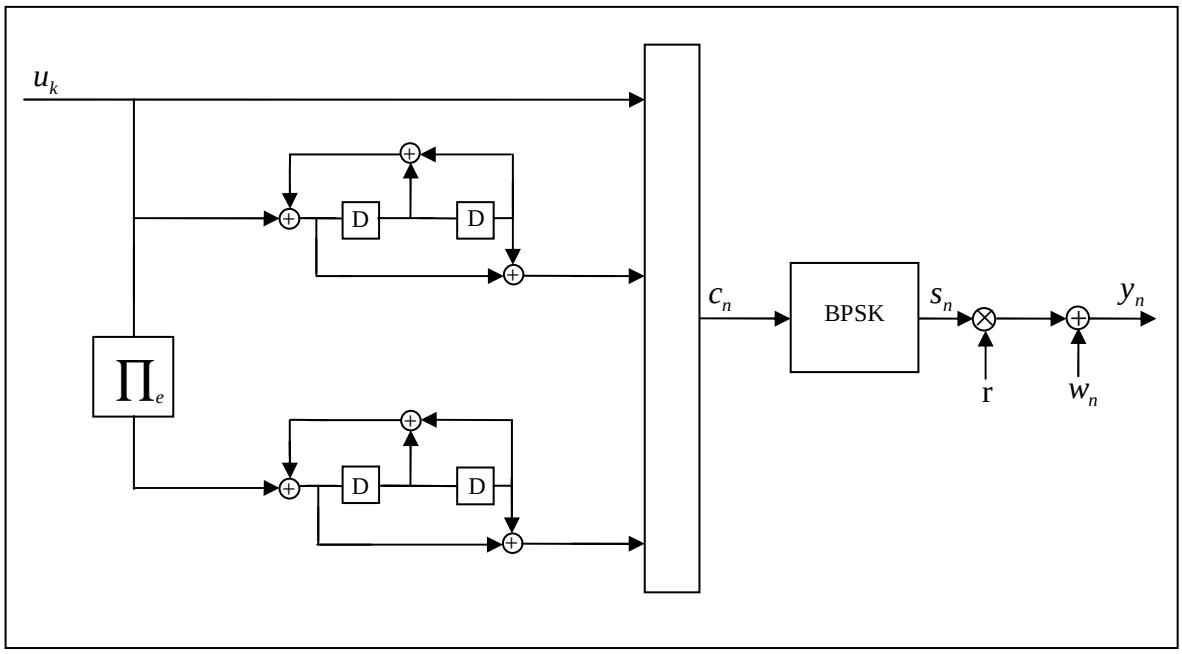
Şekil 3.8. Turbo kodlarda paket boyu ile iterasyon kazancının ilişkisi

Şekil 3.8'da verilen grafiklere dikkat edildiğinde, 5×10^{-3} bit-hata oranı için paket boyu 256 iken 2. ve 7. iterasyon değerleri arasındaki kazanç 0.5 dB olurken paket boyu 2048 iken aynı iterasyon değerleri arasındaki kazanç 0.67 dB olmaktadır. Bu sonuçlardan da görüldüğü gibi paket boyunun artması iterasyon sayısıyla elde edilen kazancı da artırmaktadır. Bu performans artışına rağmen blok uzunluğunun artmasıyla kod çözme zamanı artacağından sistemde oluşacak bu gecikme göz önünde bulundurulmalıdır.

Delme işleminin turbo kodların başarımı üzerindeki etkisine değinilecek olursa, başarımı azaltacağı açıktır[51]. Çünkü delme işlemiyle birlikte eşlik bitlerinden her hangi bir anda bir tanesi iletildiğinden ve alıcıda iletilmeyen bu eşlik bitinin "0" olduğu kabul edildiğinden başarımı azaltacaktır.

3.2. Turbo Kodların Dar Band Rayleigh Kanallar Üzerinde Başarım Analizleri

Bu bölümde turbo kodların dar bandlı rayleigh kanallar üzerinde başarımı gerçekleştirilmiş ve elde edilen simülasyon sonuçları verilmiştir. Kullanılan Turbo verici yapısı aşağıda Şekil 3.8'de verilmiştir.

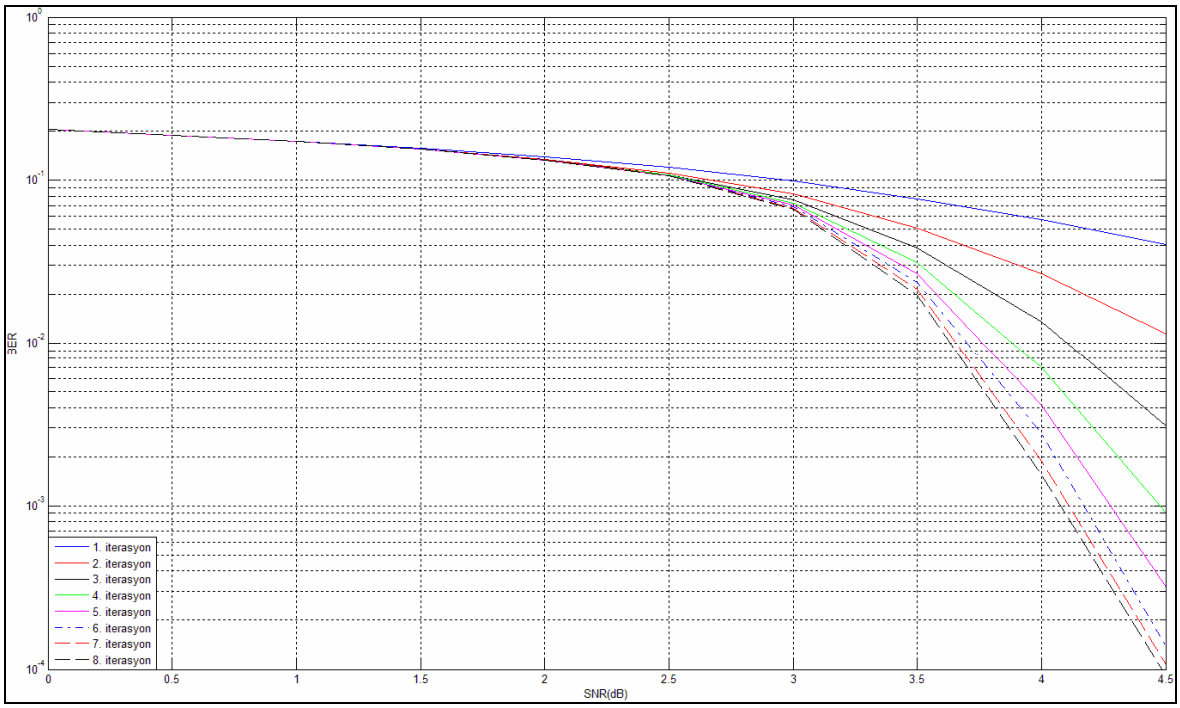


Şekil 3.9. Dar bandlı rayleigh kanal için Turbo verici yapısı

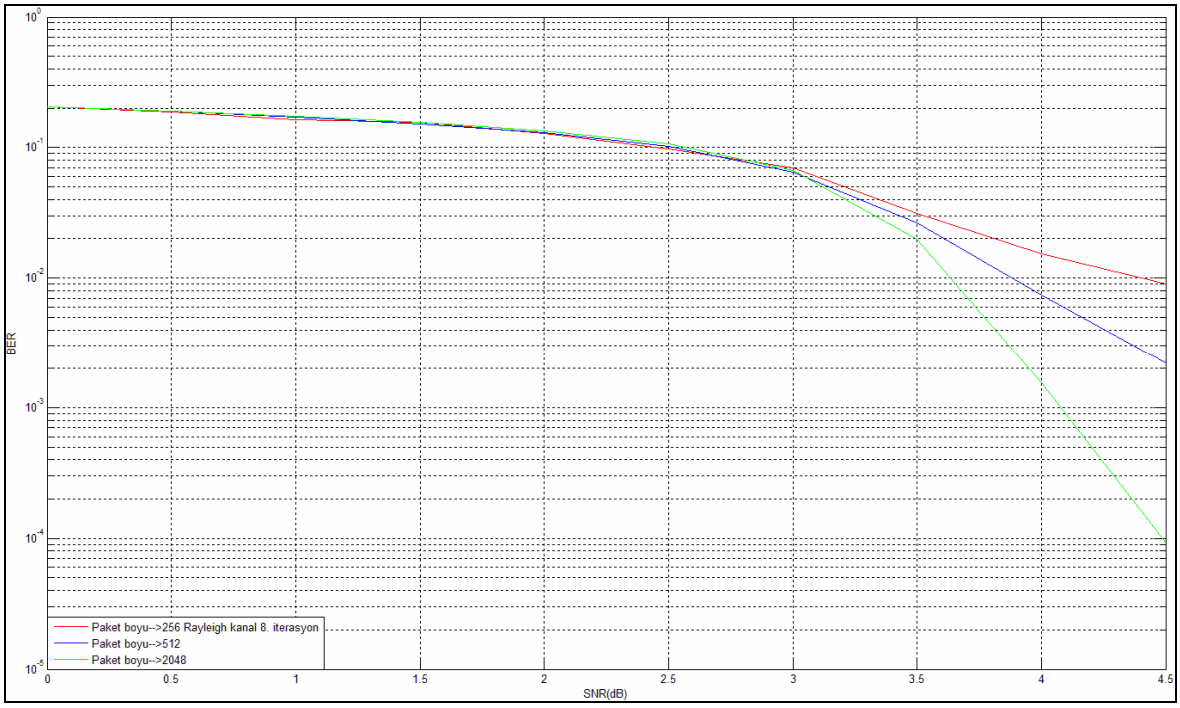
Şekil 3.9' de verilen turbo verici yapısında görülen r değişkeni rayleigh dağılıma sahiptir ve kanal kazancı veya bayılma katsayısı olarak adlandırılır ve aşağıda verilen bağıntı kullanılarak elde edilmiştir.

$$r = \sqrt{x^2 + z^2} \tag{3.1}$$

(3.1) eşitliğinde yer alan x ve z değişkenleri gauss dağılımına sahip rastlantısal değişkenlerdir.



Şekil 3.10. Turbo kodlarda Rayleigh kanalda özyineleme sayısının etkisi



Şekil 3.11. Turbo kodların Rayleigh kanalda paket boyunun etkisi

Şekil 3.10 ve 3.11' de verilen grafiklerden görüldüğü gibi yaklaşık 3 dB' ye kadar iterasyon sayısı ve paket boyunun değişmesiyle bir kazançtan söz edilemez. 3 dB' den

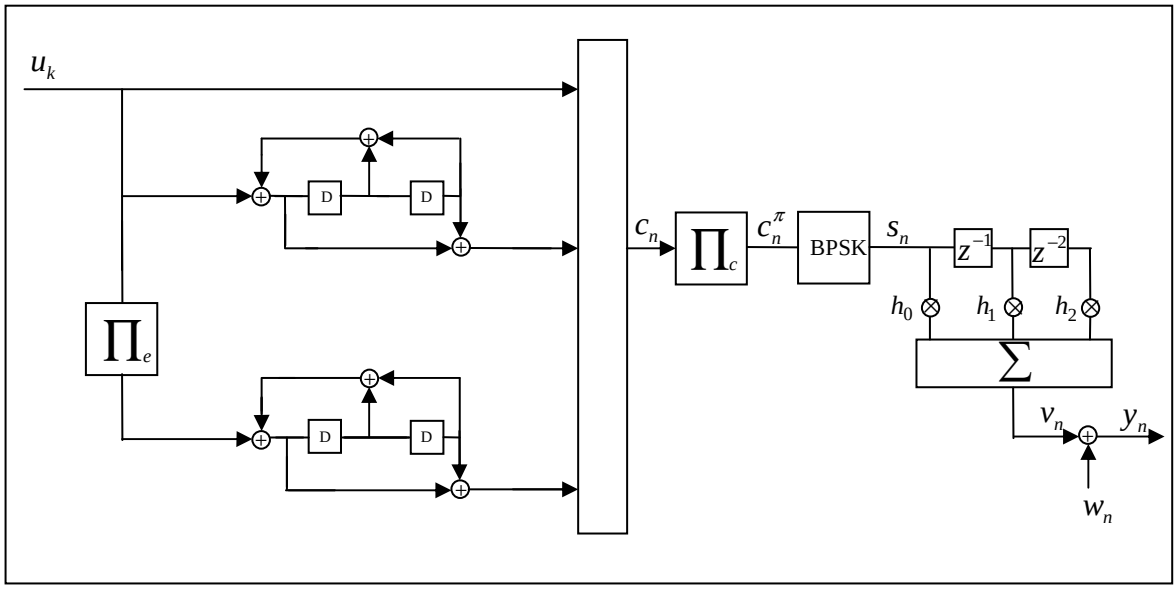
sonra yukarıda açıklanan nedenlerden dolayı iterasyon kazancından ve paket boyuna ilişkin bir kazançtan söz edilebilmektedir.

3.3. Turbo Kodların Geniş Band Kanallar Üzerinde Başarım Analizleri

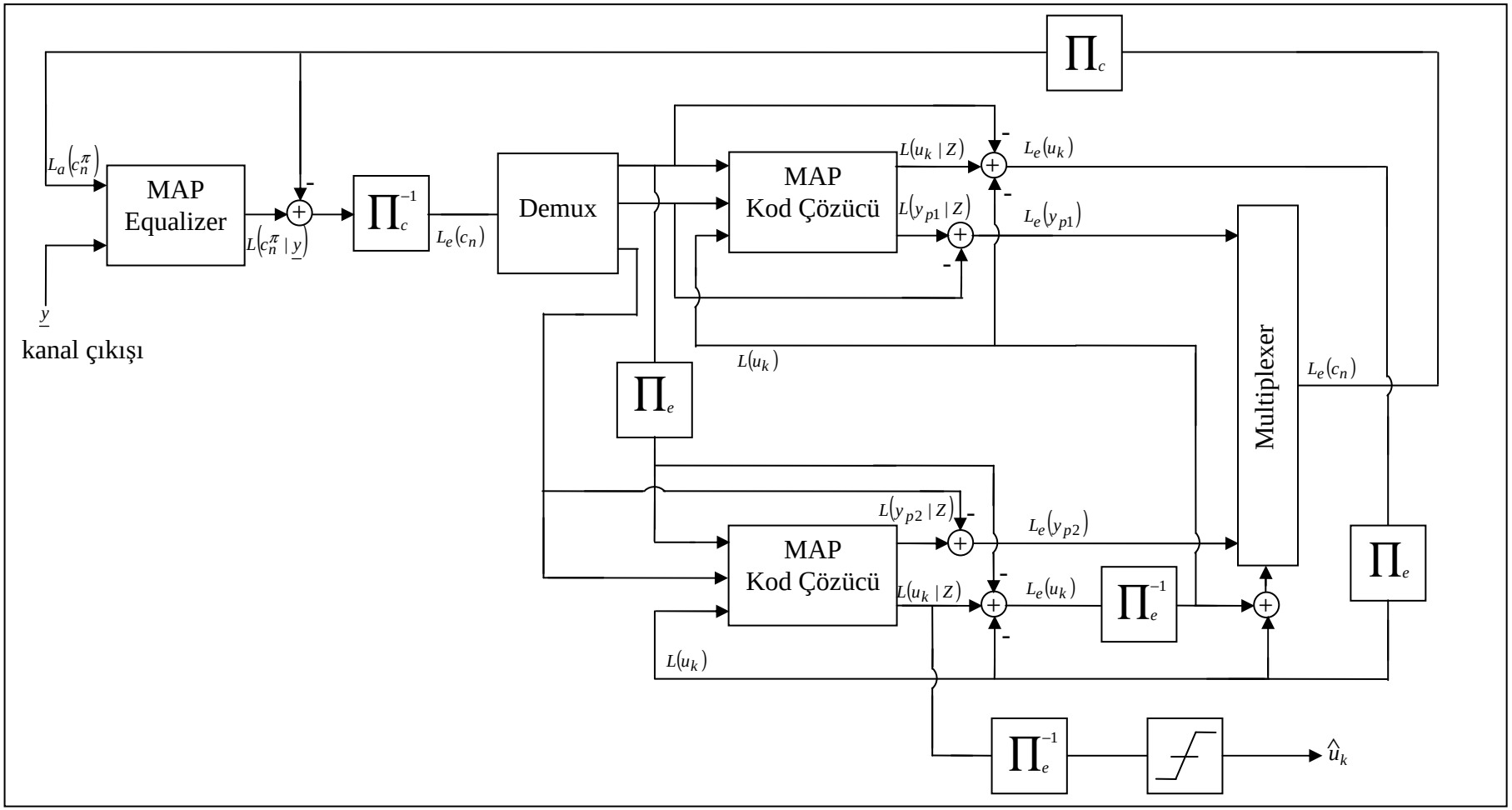
Bu bölümde Şekil 3.12'de verilen verici ve Şekil 3.13'de verilen alıcı yapısı kullanılarak turbo denkleştiricinin frekans seçici kanallarda başarım analizi yapılmıştır. Elde edilen simülasyon sonuçları aşağıda verilmiştir. Kanal olarak Proakis B 3 taplı kanalı kullanılmıştır. Kanalın transfer fonksiyonu aşağıda verilmiştir.

$$H(z) = 0.407 + 0.815z^{-1} + 0.407z^{-2} \tag{3.2}$$

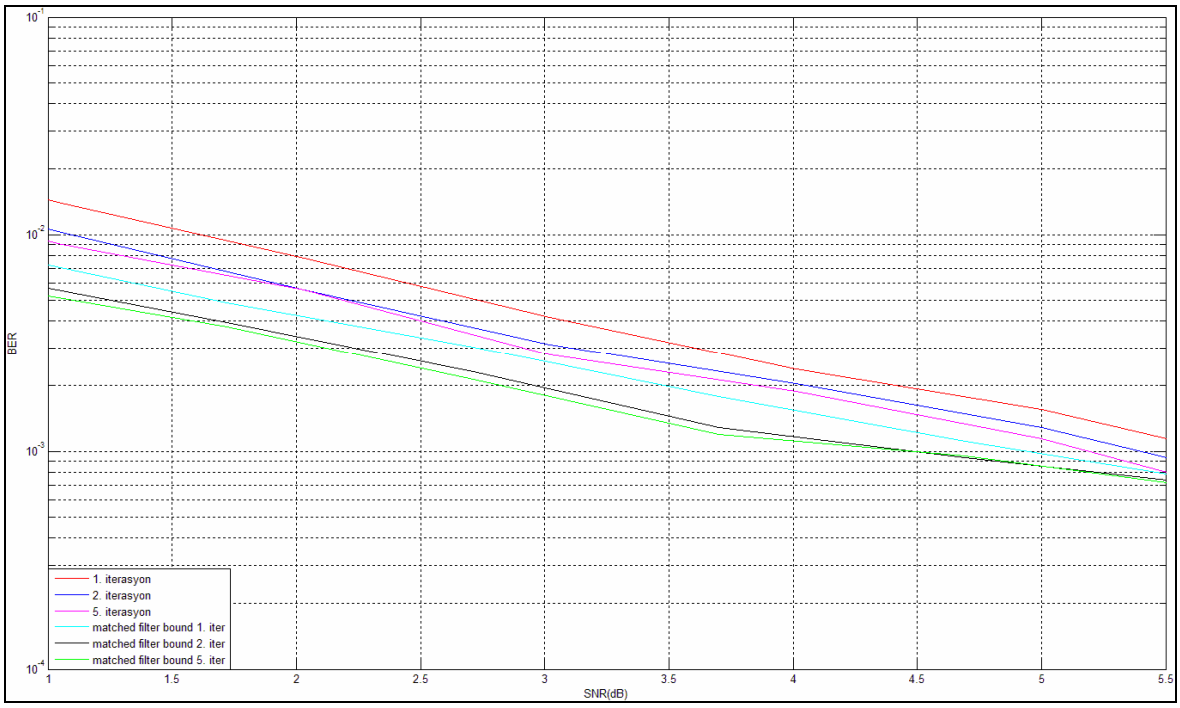
Simülasyonlarda kanalın transfer fonksiyonundan da görüldüğü gibi karmaşık (complex) bir yapıya sahip olduğu düşünülmüştür. Bundan dolayı kullanılan gauss gürültüsü de karmaşık(complex) gürültü olarak alınmıştır. Alıcıda kanal bilgisinin bilindiği kabul edilmiştir. Yani kanal kestirim işlemi yapılmamıştır. Simülasyon sonuçları 1000 kanal üzerinden elde edilmiştir.



Şekil 3.12. 3 taplı kanal modeli için turbo verici yapısı

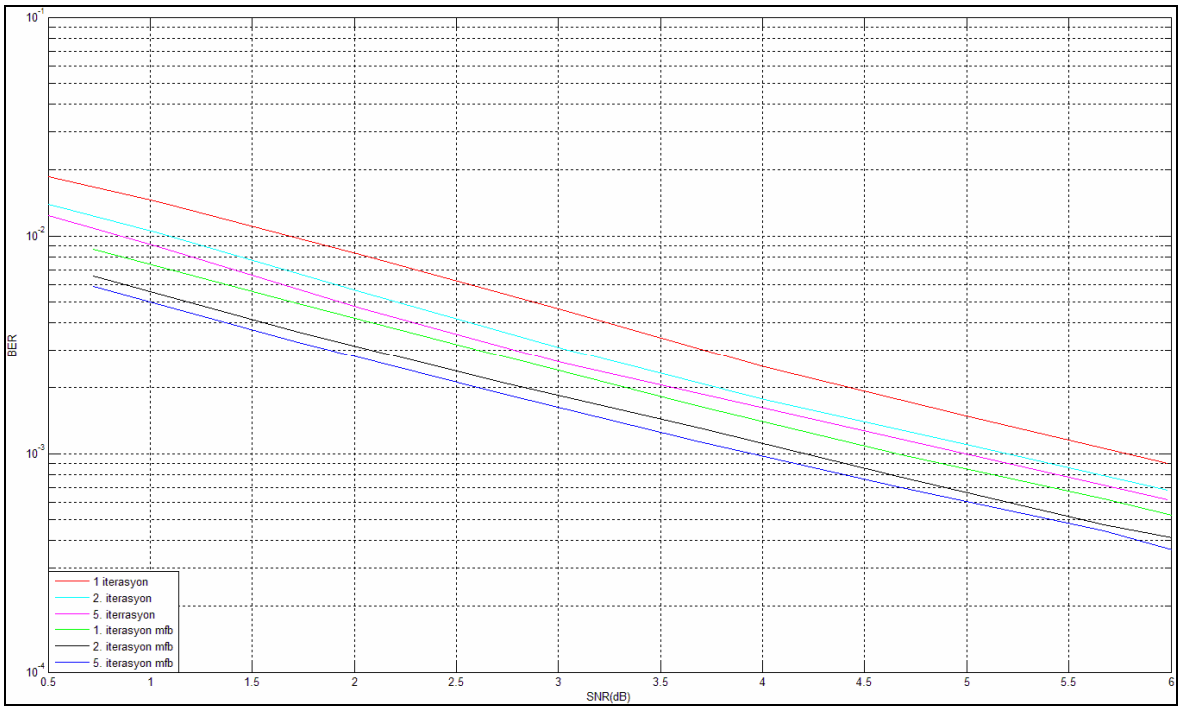


Şekil 3.13. Turbo kodların kullanılmasıyla elde edilen turbo denkleştirici yapısı



Şekil 3.14. Turbo denkleştiricinin Proakis B kanalındaki başarımı (Paket boyu=128)

MFB grafikleri belirli bir sistem için elde edilebilir en iyi performansı gösterdiğinden Şekil 3.14' de verilen grafikler incelendiğinde sistem performansının yüksek işaret-gürültü oranlarında optimum noktaya yaklaştığı görülmektedir. MFB grafikleri elde edilirken geniş bandlı kanalın tüm taplara yayılmış enerjisinin tek tapda toplandığı dikkate alınmaktadır. İterasyon kazancına dikkat edilirse MFB grafiklerinde ikinci iterasyon sonrasında bir kazançtan söz edilemeyeceği görülmektedir. Sistemin performans grafikleri incelendiğinde, örneğin 10^{-3} bit-hata oranı için 2. iterasyon ile 5. iterasyon arasında 0.2 dB' lik bir kazancın olduğu görülmektedir. Benzer şekilde yine 10^{-3} bit-hata oranı için 5. iterasyon sonucunda sistem performansı ile optimum nokta arasında 0.63 dB olduğu görülmektedir. Yüksek seviyeli modülasyon tekniklerinin kullanılmasıyla birlikte iterasyon kazancı artacaktır. Çünkü konstellasyon diyagramlarında iki komşu nokta arasındaki Euclidean mesafesi azalacağından ISI ve gürültünün etkisi daha fazla olacaktır.



Şekil 3.15. Turbo denkleştiricinin Proakis B kanalındaki başarımı (Paket boyu=1024)

Şekil 3.15' de verilen grafikler incelendiğinde paket boyunun artması ile birlikte serpiştiriciden dolayı kodun rasgeleliği arttığından başarımın artığı görülmektedir. Örneğin 10^{-3} bit-hata oranı için 5. iterasyonda paket boyu=128 bit olduğunda yaklaşık 5.2 dB işaret-gürültü oranı gerekli iken paket boyu=1024 olduğunda 5 dB' lik bir işaret gürültü oranı gereklidir.

4. SONUÇLAR

- Turbo kod çözücünün iteratif yapısından dolayı iterasyon sayısı ile sistemin başarımı artmaktadır.
- Paket boyunun artması sistemin başarımını artırmaktadır.
- Turbo denkleştiricinin performansı iterasyon sayısı ile MFB grafiklerine oldukça yaklaşmaktadır.
- Turbo denkleştirici (MAP Equaliser) nin turbo kodlama olmadan uygulanması durumunda başarımı denkleştirici süzgeçlerinden farklı olmamaktadır.
- Turbo denkleştirici (MAP Equaliser) kodlama kazancı olmadan Viterbi algoritmasının kazancına çok uzun iterasyonlardan sonra yaklaşmaktadır (Kaynaklardan gözlenen VA sonuçları ile elde edilen başarım sonuçları karşılaştırılarak gözlemlenmiştir).

5. ÖNERİLER

Bu tez çalışmasında öncelikle Turbo kodlar yapısal olarak incelenmiş ve kod çözme algoritmaları ayrıntılı bir şekilde ele alınmıştır. AWGN kanallar üzerinde simülasyonları gerçekleştirilmiş olup performans üzerinde, yapısal olarak etkileyici parametreler belirlenmiştir. Daha sonra dar band rayleigh kanal üzerinde simülasyonlar gerçekleştirilmiştir.

Son olarak geniş band kanallar üzerinden (Proaki B kanalı) turbo kodların simülayonları, turbo denkleştirici kullanılarak gerçekleştirilmiş ve elde edilen simülasyon grafikleri MFB(matched fitler bound) grafikleri esas alınarak verilmiştir. Simülasyon sonuçlarının MFB grafikleri baz alınarak verilmesinin nedeni, MFB grafiklerinin belirli bir sistem için optimum başarımı göstermeleridir. Bu çalışmada, bu değerlendirme kullanılan turbo denkleştiricinin, kanalın etkisini (ISI+toplanır gürültü) ne ölçüde telafi edebildiği ile ölçülmüştür.

Turbo kod çözücünün ardışıl yapısından dolayı iterasyon sayısı ile elde edilen kodlama kazancı artmaktadır. Yalnız elde edilen bu kazanca karşılık sistemde gecikme oluşmaktadır. Bunu telafi edebilmek için kod çözücüde kullanılan algoritmaların karmaşıklığının azaltılması gerekmektedir. Serpiştirici boyutunun artması kodun rasgeleliğini artırdığından kodlama kazancını artırmaktadır. Benzer şekilde serpiştirici boyutunun artması sistemde gecikmeye yol açacaktır.

Turbo denkleştiricilerde de benzer şekilde turbo sistem yapısı kullanıldığından iterasyon sayısı ile kodlama kazancı artmaktadır ve bu kazanca karşılık sistemde gecikmeye yol açmaktadır. MFB grafikleri baz alındığında sistemin performansının iterasyon sayısı arttıkça optimum noktaya yaklaştığı görülmektedir. Yalnız yapılan simülasyon çalışmalarında alıcıda kanal bilgisinin bilindiği kabul edilmiştir. Gerçekleştirilen bu simülasyonların gerçel zamanlı olarak denenebilmesi için kanal kestirim işleminin dahil edilmesi gerekmektedir.

Modülasyon tekniği olarak BPSK modülasyonu kullanılmıştır. Turbo denkleştiricinin yüksek data hızına sahip sistemlere uyarlanabilmesi için band verimliliği açısından çok seviyeli modülasyon türleri için simülasyonların gerçeklenmesi gerekmektedir.

6. KAYNAKLAR

1. Berrou, C., Glavieux, A. ve Thitimajshima, P., Near Shannon Limit Error-Correcting Coding and Decoding, Turbo-Codes, Proceedings of ICC'93, Geneve, Switzerland,(1993) 1064-1070.

2. Berrou, C. and Glavieux, A., Near optimum error Correcting coding and Decoding, Turbo codes, IEEE Transactions on Communications, (1996) 44, 1261-1271.

3. Shannon, C. E., A mathematical theory of communication, Bell System Technical Journal, (1948) 379-427.

4. Douillard, C., Picart, A., Jezequel, M., Didier, P., Berrou, C. ve Glavieux, A., Iterative correction of intersymbol interference, Turbo-equalization, European Transactions on Communications, (1995) 507-511.

5. Benar, M., Turbo kodlama ve AWGN kanallarda başarım analizi,Yüksek Lisans Tezi, 2002.

6. Proakis, J., G., Digital Communication, Fourth Edition.

7. Shannon, C. E, Communication in the Presence of the Noise, Proceedings of the IEEE, 1998.

8. Hanzo, L., Liew, T., H. ve Yeap, B., L, Turbo Coding, Turbo Equalisation and Space-Time Coding for Transmission over Fading Channels, 2002.

9. Woodard, J. P. Ve Hanzo, L., Comperative Study of Turbo Decoding Techniques, An Overview, IEEE Transactions on Vehicular Technology, 2000.

10. Hagenauer, J., Offer E., ve Papke, L., Iterative Decoding of Binary Block and Convolutional Codes, IEEE Transactions on Information Theory, (1996) 429-445.

11. Valenti, M. C., Turbo Codes and Iterative Processing,1998.

12. Forney, G. D., The Viterbi Algorithm, IEEE Transactions on Communicaitons, (1973) 268-278.

13. Viterbi, A. J., Error Bounds for Convolutional Codes and An Asymptotically Optimum Decoding Algorithm, IEEE Transactions on Information Theory, (1967) 260-269.

14. Bahl, L., R., Cocke, J., Jelinek, F. Ve Raviv, J., Optimal Decoding of Linear Codes for Minimising Symbol Error Rate, IEEE Transactions on Information Theory, (1974) 248-287.

15. Robertson, P., Illuminating the Structure of Code and Decoder of Paralel Concatenated Recursive Systematic (Turbo) Codes, 1994.

16. Sklar, B., A Primer on Turbo Code Concepts, IEEE Communications Magazine, 1997.

17. Sadjadpour, H., R., Maximum A Posteriori Decoding Algorithms For Turbo Codes, AT&T research-Shannon Labs, 2000.

18. Pietrobon, S., Robertson, P. Ve Papke, L., Iterative (turbo) decoding of systematic convolutional codes with the MAP and SOVA Algorithms, 1994.

19. Benedetto, S., Montorsi, G., Divsalar, D., ve Polara, F., Soft-output decoding algorithms in iterative decoding of turbo codes, TDA Progress Report, (1996) 42-124.

20. Pietrobon, S. S., Implementation and Performance of a Serial MAP Decoder for use in an Iterative Turbo Decoder, Proceedings of 1995 IEEE International Symposium on Information Theory , Whistler ,British Columbia, Canada, 1995.

21. Barbulescu, A. S. Ve Pietrobon, S. S., Interleaver Design for Turbo Codes, Electronics Letters, (1994) 2107-2108 .

22. Benedetto, S. ve Montorsi, G., Design of paralel concatenated convolutional codes, IEEE Transactions on Communication, (1996) 591-600.

23. Yuan, J., Vucetic, B., and Feng, W., 1999. Combined turbo codes and interleaver design, IEEE Transactions on Communications, (1996) 484-487.

24. Hall, E. K. ve Wilson, S. G., Design and analysis of turbo codes on rayleigh fading channel, IEEE Journal on Selected Areas in Communication, (1998) 160-174.

25. Sklar, B., Fundamentals of Turbo Codes, 2002.

26. Ryan, W., E., A Turbo Code Tutorial, 1998.

27. Pietrobon, S., S. ve Barbulescu, A. S., A Simplification of The Modified Bahl Decoding Algorithm for Systematic Convolitional Codes, Australian Space Centre for Signal Processing, 1996.

28. Sauer-Greff, W., Siegrist, M.,Turbo-Equalization of Nonlinear ISI-Channels Using High-Rate FEC Codes, 2005.

29. Dejonghe, A., Novel Turbo-Equalization Techniques for Coded Digital Transmission, Phd Thesis, 2004.

30. Kaynak, M. N., Turbo Codes for Partial Response Channels, Arizona State University, 2002.

31. Ertürk, S., Sayısal Haberleşme, 2005.

32. Burr, A., Modulation and Coding for Wireless Communication.

33. Hamming, R.W., Error Detecting and Error Correcting Codes, Bell Sys. Tech. (1950) 147-160.

34. Clark K. ve Marley T, The Perfect Code, Golay Codes, 2005.

35. Muller, D.E., Application of Boolean Algebra To Switching Circuit Design, IEEE Transactions on Computers, 3 (1954) 6-12.

36. Prange, E., Cyclic Error-Correcting Codes and Their Implementation, Air Force Cambridge Araştırma Merkezi, Teknik Bilgiler TN-57-103, Cambridge, Eylül, 1957.

37. Meggitt, J.E., Error Correcting Codes and Their Implementation, IRE Transactions on Information Theory, 7 (1961) 232-244.

38. Reed, I.S., Solomon, G., Polynomial Codes Over Certain Finite Fields, SIAM Journal on Applied Mathematics, 8 (1960) 300-304.

39. Berlekamp, E.R., Peile, R.E. ve Pope, S.P., The Application of Error Control To Communications, IEEE Communication Magazine, 25 (1987) 44-57.

40. Elias, P., Coding for Noisy Channels, IRE Conv. Record, 4 (1955) 37-47.

41. Wicker, S., Error Control Systems for Digital Communications and Storage, Prentice Hall Inc., Englewood Cliffs, 1995.

42. Fano, R.M., A Heurstic Discussion of Probabilistic Decoding, IEEE Transactions on Information Theory, 9 (1963) 64-74.

43. Jelinek, F., An Upper Bound on Moments of Sequential Decoding Effort, IEEE Transactions Information Theory, 15 (1969) 464-468.

44. Costello, D.J., Hagenauer, J., Imai, H., Wicker, S., B., Applications of Error- Control Coding, IEEE Transactions on Information Theory, 44 (1998) 2531-2560.

45. Forney, G.D., Concatenated Codes, MIT Press, Cambridge, 1966.

46. Robertson, P., Improving Decoder and Code Structure of Parallel Concatenated Recursive Systematic (Turbo) Codes, International Conference on Universal Personal Communications, (1994) 183-187.

47. Robertson, P.,Villebrun, E. ve Hoeher, P., A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain, International Conference on Communications (1995) 1009-1013.

48. Blackert, W.J., Hall, E.K., Wilson, S.G., Turbo Code Termination and Interleaver Conditions, Electronics Letters, 31 (1995) 2082-2083.

49. Akbar, I. A., Markov Modeling of Third Generation Wireless Channels, Master of Science, 2003.

50. Koch, W. ve Baier, A., Optimum and sub-optimum detection of coded data disturbed by time-varying intersymbol-interference, IEEE Globecom, (1990) 1679-1684.

51. Qi, J., Turbo Code in IS-2000 Code Division MultipleAccess Communications under Fading, Master of Science, 1999.

52. Chatzigeorgiou, I.A., Rodrigues, M.R.D., Wassell, I.J., Carrasco, R., A Comparison of Convolutional and Turbo Coding Schemes For Broadband FWA Systems, 2005.

ÖZGEÇMİŞ

21.11.1982 tarihinde Trabzon'da doğdu. Lise öğrenimini Trabzon Lisesi'nde tamamladı. 1999 yılında Karadeniz Teknik Üniversitesi Mühendislik Mimarlık Fakültesi Elektrik-Elektronik Mühendisliği Bölümü'nü kazandı. 2004 yılında bu bölümden mezun oldu. Aynı yıl Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalı'nda yüksek lisans öğrenimine başladı. 2005 yılında Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü kadrosunda Araştırma Görevlisi olarak atandı. Halen bu görevine devam etmektedir.