YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GEOMETRİK İNŞA KODLARININ ÜRETİMİ VE BAŞARIMI

Elektrik-Elektronik Yük. Müh. Tugay AKBAŞ

FBE Elektronik-Haberleşme Mühendisliği Anabilim Dalı Haberleşme Programında Hazırlanan

DOKTORA TEZİ

Tez Savunma Tarihi : 18.10.2007
Tez Danışmanı : Prof. Metin YÜCEL (YTÜ)
Prof. Dr. Osman N. UÇAN (İÜ)
Jüri Üyeleri : Prof. Dr. Aydın AKAN (İÜ)
: Prof. Dr. Mahmut ÜN (İÜ)
: Prof. Dr. Herman SEDEF (İÜ)
: Doç. Dr. Işın ERER (İTÜ)

İSTANBUL, 2007

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

G	Üreteç matrisi
C	Blok kod
E_b	Bir bilgi biti başına düşen enerji miktarı
N_0	Güç tayf yoğunluğu
σ^2	Varyans (Değişinti)
dB	Desibel
u	Sayısal bilgi kaynağından üretilen bilgi dizisi
v	Kod kelimesi
r'	Gürültüye uğrayan dalga biçimi
r	Alıcıdan alınan mesaj dizisi
u'	Kestirilen bilgi dizisi

KISALTMA LİSTESİ

GÜ	Geometrik Ürün
Gİ	Geometrik İnşa
TBGG	Toplamsal Beyaz Gauss gürültülü
AWGN	Additive White Gaussian Noise
İFKA	İkili Faz Kaydırmalı Anahtarlama
BHO	Bit Hata Oranı
KHO	Kelime Hata Oranı
BLHO	Blok Hata Oranı
BCJR	Bahl, Cocke, Jelinek ve Raviv
GAC	Genelleştirilmiş Dizi Kodları (Generalised Array Codes)
LDPC	Low Density Parity Check
DYPK	Düşük Yoğunluklu Parite Kontrol
BTC	Block Turbo Codes
TPC	Turbo Product Codes
PA	Product Accumulate (Ürün Yığılma)
TÜA	Toplam Ürün Algoritması
BCH	Base Chauolhurı and Hacquenghem
İGO	İşaret Gürültü Oranı
GTY	Güç Tayf Yoğunluğu
YGYÇ	Yumuşak Giriş Yumuşak Çıkış
GSY	Güç Spektrum Yoğunluğu

ŞEKİL LİSTESİ

Sayfa

ÖNSÖZ

Doktora öğrenimim ve tez çalışmamın başlatılması ve yürütülmesi konusunda, fikir ve eleştirileri ile yapmış olduğu yardımlardan dolayı çok değerli hocam Sayın Prof. Metin YÜCEL'e, tez çalışmam boyunca çalışmalarımda yardımlarını ve desteklerini eksik etmeyen, çalışmalarımı en doğru yönde yapmamı sağlayan ve her ihtiyaç duyduğumda yanımda olan ikinci danışmanım Sayın Prof. Dr. Osman Nuri UÇAN'a, bu tez çalışmasının tamamlanmasında en büyük pay sahiplerinden olan, her aşamada çalışmalarını ve bilgilerini benimle paylaşma nezaketinde bulunan Sayın Yrd. Doç. Dr. Gökmen ALTAY'a, doktora eğitimim sırasında en zor anlarımda destekleriyle yanımda olan Sayın Prof. Dr. Tülay YILDIRIM'a saygı dolu teşekkürlerimi sunarım. Ayrıca tez izleme komite çalışmaları ve doktora tez sınavı aşamalarında akademik değerlere sahip çıkarak yaptıkları yapıcı eleştirileri, sabır ve hoşgörüleri için Sayın Sayın Prof. Dr. Mahmut ÜN'e, Sayın Prof.Dr Aydın Akan'a, Sayın Prof. Dr. Herman SEDEF'e, Sayın Prof. Dr. Filiz GÜNEŞ'e ve Sayın Doç. Dr. Işın ERER'e teşekkür ediyorum.

Bana sunmuş oldukları çalışma ortamı ve desteklerinden dolayı, Vodafone GSM operatöründen yöneticilerim Sayın Kadir KUŞ ve Sayın Evren GÜVEN'e, İNTA Mühendislik Genel Müdürü Sayın Levent KOPER'e, NSN Kuveyt Ülke Servisleri Müdürü Sayın Risto Puumalainen'e, yöneticilerim Benny ONGGONO ve Abhishek Panditha'ya; bana zor günlerimde her konuda destek veren arkadaşlarım Doç. Dr. Reşit ŞENDAĞ, Doç. Dr. Taner AKKIN, Dr. Nuri YILMAZER, Tamer MİREL, Anıl T. ARSLAN, Asena AKÇAY, Barış SOYTÜRK, Ebru Çağlayan, Hüseyin EMİR, Murat ESENOLUK, Şule YAYCI ve Umut BAYSAL'a çok teşekkür ederim.

Ayrıca çalışmalarım sırasında bana verdikleri moral destekten ve gösterdikleri sabırdan dolayı çok kıymetli eşim Hamide AKBAŞ'a ve canım oğlum Emre Kaan AKBAŞ'a, bugünlere gelmemde büyük pay sahibi olan annem, babam ve kardeşlerime en içten teşekkürlerimi sunarım.

Tugay AKBAŞ

İstanbul, 2007

ÖZET

Kanal kodlama, kolaylıkla kodlanabilen ve kodu çözülebilen bir kod bulmayı ve aynı zamanda en küçük mesafe için yüksek oranda kodlama kazancı elde etmeyi amaçlar. Büyük uzunluklu doğrusal blok kodların kodlayıcı ve kod çözücü gerçeklemeleri pratikte zor olmakla birlikte geniş kapasiteli hafızalar gerektirmektedir. Düşük Yoğunluklu Parite Kontrol Kodlar, yüksek oranda Bit Hata Oranı (BHO) performansı gösterdikleri için kodlama alanında büyük ilgi görmektedirler. Buna rağmen, kodlayıcı kısmında bilgi bitlerini kodlamak için kullanılan yüksek yoğunluklu ve düzensiz üreteç matrislerinin, pratikte kodlayıcı gerçeklemesinde bazı zorluklara sebep olması bir dezavantaj olarak karşımıza çıkmaktadır.

Daha önceki çalışmada (Altay G., 2006) genel yapıda, tam bilgi oranı sağlayan ve Hamming mesafesi 4 olan bütün en iyi çift blok kodları, bir kod ailesi olarak üretebilen yeni bir ikili doğrusal blok kod inşa tekniği olan Geometrik Ürün (GÜ) kodları önerilmişti. Bu kodların üreteç matrislerinin, düşük yoğunlukta, düzenli ve yarı çevrimsel yapıda olması ve daha küçük alt üreteç matrislerinden oluşması gibi bazı yararlı özellikleri vardır. Çevrimsel ya da yarı çevrimsel yapıdaki bir kod daha az karmaşıklıkla kodlanabilir. Gİ kodları da bu özelliğe sahiptir ve buradan hareketle, kodlayıcı ve kod çözücü gerçeklemelerinde esneklik sağlaması bir avantajdır. Gİ kodları ile üretilmiş, minimum Hamming mesafesi 4 olan kod aileleri, uzatılmış Hamming kodlarını ve minimum Hamming mesafesi 4 olan Reed-Muller kodlarını da içermektedir. Çünkü literatürde uzun zamandır var olan bu kodların uzunlukları 8, 16, 32 gibi 2'nin kuvvetleri şeklinde değişmektedir. Aynı zamanda bu çalışmada da önerilen Gİ kod aileleri 8,10,12,14,16,18,..., gibi 2'nin katları şeklinde kod uzunluklarına sahiptir. Dolayısıyla Gİ kodları daha üst bir kod ailesi olarak karşımıza çıkmaktadır. Bu nedenle Blok Turbo kodları ve Turbo Ürün kodlarında bileşen olarak kullanılabilir. Ayrıca bu çalışmada önceki çalışmaya ek olarak daha uzun mesafedeki Hamming kodlarını elde etmek için genel bir algoritma da geliştirilmiş, kod matrisleri sistematik hale getirilerek performans analizleri yapılmıştır.

Bu çalışmada Gİ kodların hata başarım analizleri, bir kodun performans analizi yapılırken çoğunlukla kullanılan Toplamsal Beyaz Gauss Gürültülü, (TBGG) kanallar üzerinde yapılmış, bilgisayar benzetimleri ile elde edilen hata başarımları, birçok değişik kod boyutları ve değişik kod çözme yöntemleri için sunulmuştur. Gİ kodlarının parite kontrol matrisini kullanarak kod çözme işlemi için çok uygun olmasından dolayı, Gİ kodlarının kod çözümünde de Toplam Ürün Algoritması'nı kullandık. Gİ kodlarının üreteç matrisi, bu kodların parite kontrol matrisini elde etmek amacıyla Gauss-Jordan Yöntemi'ni kullanarak kolaylıkla sistematik biçime dönüştürülebilmektedir.

Çalışmanın birinci bölümünde konuya genel bir giriş yapılmıştır. İkinci olarak, "Genel Bölümler" başlığı altında blok kodların ve tezde sıkça geçen parametrelerin kısaca açıklamaları yapılmıştır. Sonraki bölümde "Malzeme ve Yöntem" başlığı altında tez çalışmasında yapılan bilgisayar benzetimlerinde kullanılan her türlü malzeme, yöntem ve teoriler üzerinde durulmuştur. Dördüncü bölümde "Bulgular" başlığı altında bölümler halinde, kullanılan kodlama yönteminin genel bir tanıtımı yapılarak ve alt bölümler halinde bu kodlama yöntemi ile elde edilen kod aileleri sunulmuştur. Son olarak önerilen kodların bilgisayar benzetimleri kullanılarak elde edilen hata başarımları verilmiştir. 5. bölümde de sonuçlar detaylı ve açık olarak ifade edilmiştir.

Anahtar Kelimeler: Doğrusal blok kodlar, hata başarımı, toplam ürün algoritması

ABSTRACT

The goal of channel coding is to find a code that is easy to encode and decode, and at the same time gives a high code rate for the largest minimum distance. Encoder and decoder implementation of long length linear block codes is difficult in practice and also requires large capacity of memory. Low Density Parity Check (LDPC) codes receive enormous interest from the coding community as they hold higher Bit Error Rate (BER) performances. Whereas, they have the disadvantage for encoding part as they have large, high-density and irregular generator matrices to encode information data, which cause difficulty in encoder implementations in practice. A generator matrix with low density of ones is considered as Low Density Generator Matrix (LDGM) and it was shown that it has practical advantage of decreased encoding complexity and increased decoder architecture flexibility and also achieves a performance close to the Shannon theoretical limit. Block Turbo Codes (BTC) or Turbo Product Codes (TPC) and Product Accumulate (PA) codes have been shown to provide near-capacity performance and low encoding and decoding complexity. They use simpler components codes to produce very long block codes.

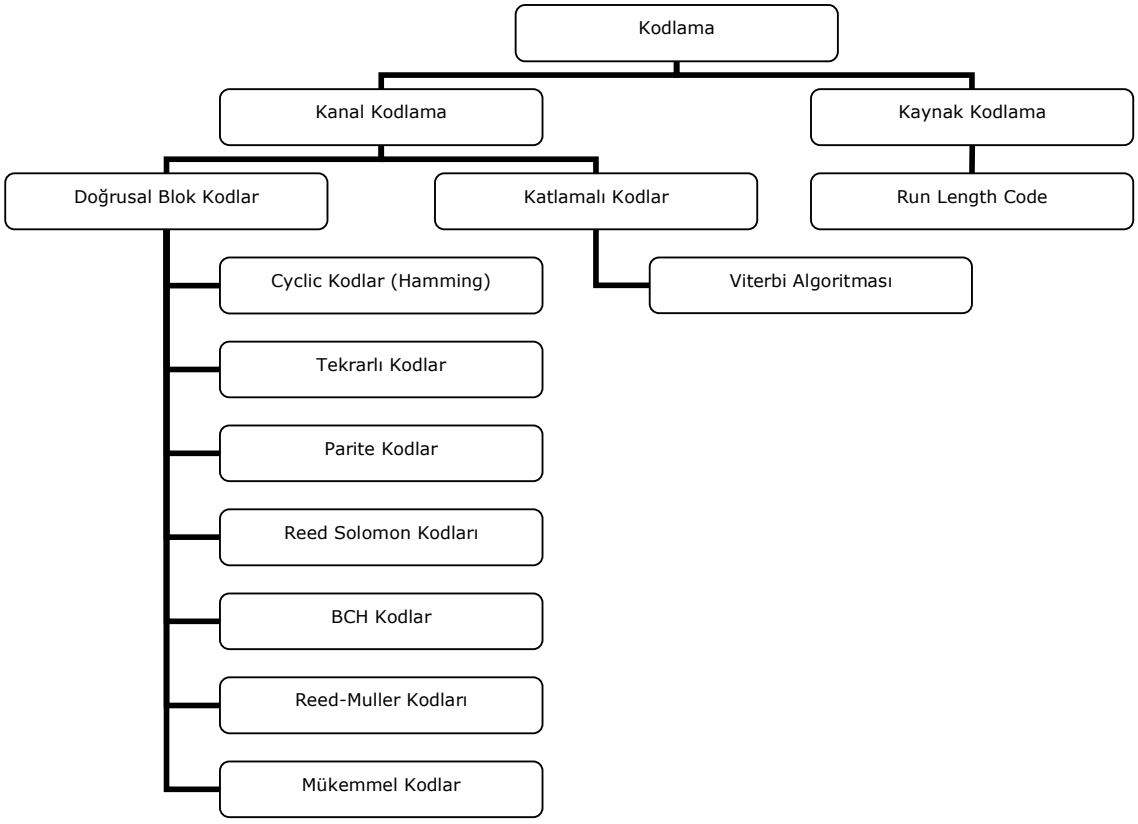
Recently (Altay G., 2006), a new binary linear block code construction technique, Geometric Construction (GC) codes was proposed that provide the full-information rate for all the Hamming distance-4 even codes. The generator matrix of these codes has very useful properties such that it contains the lowest density of ones, it has quasi-cyclic and regular structure, and it is composed of smaller group generator matrices. A code with cyclic or quasi-cyclic property can be encoded with less complexity and GC codes have this property. Therefore, it is advantages from practical point of view as it provides flexibility in encoder and decoder implementations. Extended Hamming codes and distance-4 Reed Muller (RM) codes are subsets of our GC construction codes since they generate codes as the power of 2 in length, whereas GC codes generates the codes as the multiple of 2 in length. Since the sizes of GC codes are multiple of 2 for Hamming distance-4, they can also be employed as component codes in BTC, TPC and PA codes.

In this thesis, we evaluate exhaustively error performances of GC codes over Additive White Gaussian Noise (AWGN) channel that is the most commonly used channel in evaluating the performance of a code. Sum-Product Algorithm (SPA) is an iterative decoding algorithm and it is extremely efficient for decoding LDPC codes. We employed SPA for decoding GC codes as they are very suitable for decoding using the parity control matrix of GC codes. Since GC codes are high rate, the parity control matrix of GC codes has the smallest possible size for Hamming distance-4 codes. Consequently the complexity of decoding process is reduced. The generator matrix of GC codes can easily be converted into systematic form to obtain parity-control matrix of it using Gauss-Jordan algorithm (Noble B., 1969). In order to keep the paper self-contained we first give an overview of construction for GC codes and then present the simulation results.

Keywords: Linear block codes, error performance, sum-product algorithm

1. GİRİŞ

Kod, birçok bilim dalı için önemli bir rol oynamaktadır. Bunlardan bazıları şöyle sıralanabilir: Matematik alanında istatistik biliminde, bilgisayar alanında şifreleme konusunda, haberleşme alanında, haberleşme karmaşıklığını en düşük seviyeye indirmek ve kaynakların etkin biçimde temsil edilmesini sağlamak için kullanılmaktadır. Kodlama, iletişim alanında kaynakların, kanalların, alıcıların bilgi karakteristiklerini incelemek, bilginin iletimini optimize etmek ve iletimin güvenilirliğinin düzeltilmesini sağlamak amacıyla kullanılmaktadır. (Richardson, Urbanke, 2005)



Şekil 1.1 Kodlama türleri

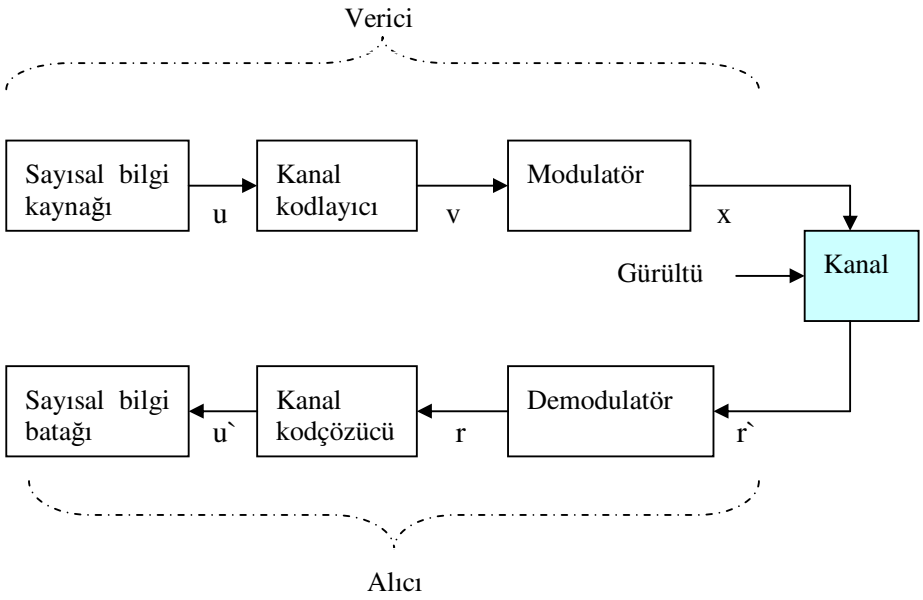
Kodlama, kanal ve kaynak kodlama olarak iki ana gruba ayrılabilir. Kaynak kodlamada, bilgiyi daha etkin biçimde temsil etmek için verinin boyutu azaltılır. Günlük hayatta sıkça karşılaştığımız "sıkıştırma" programları, bu kodlamanın bir uygulamasıdır. Sıkıştırma işlemi, daha az yer kaplamak amacıyla dosyaların boyutlarının küçültülmesidir. Kanal kodlamada ise verinin iletimini oluşabilecek bozulmalara karşı daha güvenli şekilde yapabilmek amacıyla bilgiye fazladan bitler eklenir. Bu nedenle bilginin boyutu artar. (Pless, 1989)

İletişim sistemlerinin günlük hayatımızın vazgeçilmez parçaları haline gelmesiyle, hızlı ve hatasız sayısal bilgi iletimi gerekliliği de önemli bir konu olarak karşımıza çıkmaktadır. Haberleşme sistemlerinde bilginin iletimi için vericiden gönderilen mesaj işareti, işaretin gönderildiği kanaldan ve aynı zamanda verici ve alıcı devrelerinden kaynaklanan işareti bozucu etkilerden dolayı, bozulmaya uğrar ve alıcıda gönderilenden farklı bir bilgi alınabilir. Bu durum haberleşme sisteminin hatalı mesaj göndermesi olarak adlandırılır. Kanal kodlama, sayısal haberleşme sisteminde, alıcıda alınan bilgi bitlerinde meydana gelebilecek bu hataları ortadan kaldırmayı amaçlayan geniş bir araştırma alanıdır.

Örneğin, bir müzik CD'sinde tozlardan ve çiziklerden oluşabilecek bozulmaları önlemek amacıyla Reed-Solomon kodlaması kullanılır. Bu örnekte iletim kanalı CD'nin kendisidir. Ayrıca, cep telefonlarında, yüksek frekanslı radyo iletiminden kaynaklanan gürültü ve bozulmaları düzeltmek için kanal kodlama kullanılmaktadır. Modemlerde ve telefon iletiminde de kanal kodlama kullanılmaktadır.

Kanal kodlama teoreminde kanal kapasitesinin teorik sınırı ile ilgili olarak, Shannon (Shannon, 1948) ortaya koyduğu teoreminde, kodlama oranının kanal kapasitesinden küçük olması durumunda, bilginin uygun bir biçimde kodlanmasıyla, gürültülü bir kanalda bilgi iletimi esnasında oluşan hataların istenilen en düşük oranlara indirilebileceğini ispatlamıştı. Ancak bu işlemin nasıl yapılacağı bir problem olarak bırakılmıştı (Lin, Costello, 2004). Bu teorem ile birlikte yeni kanal kodlama araştırma alanı ortaya çıkmıştır. Bu alanda yapılan çalışmaların en önemli amacı, kodlaması ve kod çözmesi kolay ve mümkün olduğunca küçük hata oranları sağlayan kodları bulmaktır (Bossert, 1999).

Sayısal haberleşme sisteminde iletilecek bilgi bitleri çok değişik amaçlara yönelik bir çok aşamadan geçer. Burada kanal kodlama üzerinde durduğumuzdan, Şekil 1.2 ile gösterilen basit bir haberleşme sisteminde, en temel işlem blokları ve kanal kodlama işlemi ile ilgili bloklar gösterilmiştir. Verici tarafında, sayısal bilgi kaynağından üretilen bilgi dizisi, u, kanal kodlayıcı bloğunda bir kanal kodlama algoritmasıyla kod kelimesi, v, olarak kodlanır. Modülatör, kanal kodlayıcı çıkışındaki sembolleri, kanalda iletime uygun dalga biçimine çevirir. Kanalda gürültüye maruz kalan ve bozulmaya uğrayan dalga biçimi, r', alıcı kısmındaki demodülatör aracılığı ile alınan mesaj dizisi, r, olarak kanal kod çözücü bloğuna iletilir. Böylece, kanal kod çözücü bloğunda en büyük olasılıkla kestirilen bilgi dizisi, u', elde edilir.



Şekil 1.2 Basit bir sayısal haberleşme sistemi.

Günümüzde kanal kodlama, "blok kodlar" ve "katlamalı kodlar" şeklinde iki ayrı temel biçimde incelenmektedir. Herhangi bir blok kodlama tekniğini kullanan bir kanal kodlayıcı, sayısal bilgi kaynağından üretilen bilgi dizisini k bitlik mesaj bloklarına ayırır ve bu $\mathbf{u} = (u_0, u_1, ..., u_{k-1})$ ile temsil edilir. Kodlayıcı bu u mesaj bloğunu n bitlik $\mathbf{v} = (v_0, v_1, ..., v_{n-1})$ kod kelimesi sembollerine kodlar. Kodlayıcının kodlama oranı $R = k / n$ ile tanımlanır. Bu durumda n bitlik kodlayıcı çıkışı sadece k bitlik giriş bitlerine bağlıdır ve bunu gerçekleyen kanal kodlayıcı da belleksiz bir kombinasyonel lojik devre ile gerçeklenebilir. Katlamalı kodlarda ise bellekli yapısından dolayı çıkış bitleri her zaman daha önce kodlayıcıya giren bitlere bağlı olacaktır (Lin, Costello, 2004 ve Bossert, 1999). Burada bahsedilecek tüm kodlama işlemleri ikili (binary) kodlama işlemine göre yani mod–2 aritmetiğine göre yapılacaktır.

Literatürde bulunan önemli blok kodlardan bazıları, Hamming kodları (Hamming, 1950), Reed-Muller kodları (Muller, 1954 ve Reed, 1954), ürün (product) kodları (Elias, 1954), |u|u+v| inşası kodları (Macwilliams, 1977 ve Plotkin), Golay kod (Golay, 1949 ve Wicker, 1995), çevrimsel (cyclic) kodlar (Prange, 1957), BCH (Bose, Chaudhuri and Hocquenghem) kodları (Hocquenghem, 1959 ve Bose, Chaudhuri, 1960), Reed-Solomon kodları (Reed, Solomon, 1960), Genelleştirilmiş dizi kodları (Generalised Array Codes, GAC) (Honary, Markarian, Farrel, 1993), Turbo kodları (Berrou, Glavieux, 1993, 1996, 1998), Düşük yoğunluklu parite kontrol kodları (Low Density Parity Check Codes, LDPC) (Gallager, 1962

ve 1963, Mackay, Neal 1996) ve ürün yığılma kodları (Product Accumulate, PA, codes) (Narayanan, Georghiades, 2004) olarak karşımıza çıkmaktadır. Bu kodlar kullanılarak farklı yapıda ve daha güçlü kodlar tasarlamak günümüzde başvurulan bir yöntemdir. Örneğin, Turbo kodları (Berrou, Glavieux, 1993, 1996) basit yapıda bazı katlamalı kodları yeni bir yöntemle birleştirilerek elde edilmiştir. Daha sonra da katlamalı kodlar yerine ürün kodları (Elias, 1954) gibi başka temel kodlar da kullanılarak Turbo kodlar elde edilmiştir. Bu tez çalışmasında literatürde bilinen ürün kodları temel alınarak, kodlama oranı, ürün kodlaması ile elde edilen kodlardan daha yüksek olan, yeni bir kodlama yöntemi olan Geometrik İnşa (Altay, 2006) kodlarının daha uzun kodlar için tasarlamaları, genel bir elde biçiminin elde edilmesi, kod matrislerinin sistematik hale getirilmesi ve bunların performans analizleri yapılmıştır.

Bu tezde, Geometrik İnşa kodlarının Toplamsal Beyaz Gauss gürültülü (TBGG), (Proakis, 2000) kanallardaki, bilgisayar benzetimleri ile elde edilen hata başarımları, birçok değişik kod boyutları ve değişik kod çözme yöntemleri için sunulacaktır.

Tezin bölümleri şu şekilde düzenlemiştir: 2. bölümde, kodlama, blok kodlar ve tezde sıkça geçen parametreler hakkında kısa bilgiler verilecektir. 3. bölümde “Malzeme ve Yöntem” başlığı altında tez çalışmasında yapılan bilgisayar benzetimlerinde kullanılan her türlü malzeme, yöntem ve teoriler üzerinde durulacaktır. 4. bölümde “Bulgular” başlığı altında bölümler halinde, kullanılan kodlama yönteminin genel bir tanıtımı yapılacak ve bu kodlama yöntemi ile elde edilen kod aileleri sunulacaktır. Bu kodları elde ederken oluşturulan algortimanın “karmaşıklık analizi” de yapılmıştır. Son olarak önerilen kodların bilgisayar benzetimleri kullanılarak elde edilen hata başarımları sunularak değişik iterasyonlar için karşılaştırmalar yapılacaktır. Bu bulgular değerlendirilecek, sonuçlar detaylı ve açık olarak ifade edilecektir.

2. BLOK KODLAR

Bu bölümde tez çalışmasıyla doğrudan ilgili olan ve tez çalışması sonunda ortaya konan bulguların daha iyi anlaşılabilmesi için gerekli temel konular incelenecektir.

Günümüzde veri iletiminde kodlama işlemi genellikle ikili biçimde yapıldığından, bu kısımda doğrusal blok kodlar tanıtılacaktır. Tez çalışmasında kullanılan kodlar da doğrusal blok kodların özelliğine sahiptirler. Doğrusal bir blok kod, **C**, en temelde bir üreteç matrisi, **G**, ile tanımlanır. Kodlanacak *k* bitlik mesaj bloğu *u* üreteç matrisi ile çarpılarak *n* bitlik kod kelimesine (*v*) kodlanır. Yapılan işlem matematiksel olarak ikili aritmetik uzayında şu denklemle tanımlanabilir.

$$v = u \,.\, G \qquad (2.1)$$

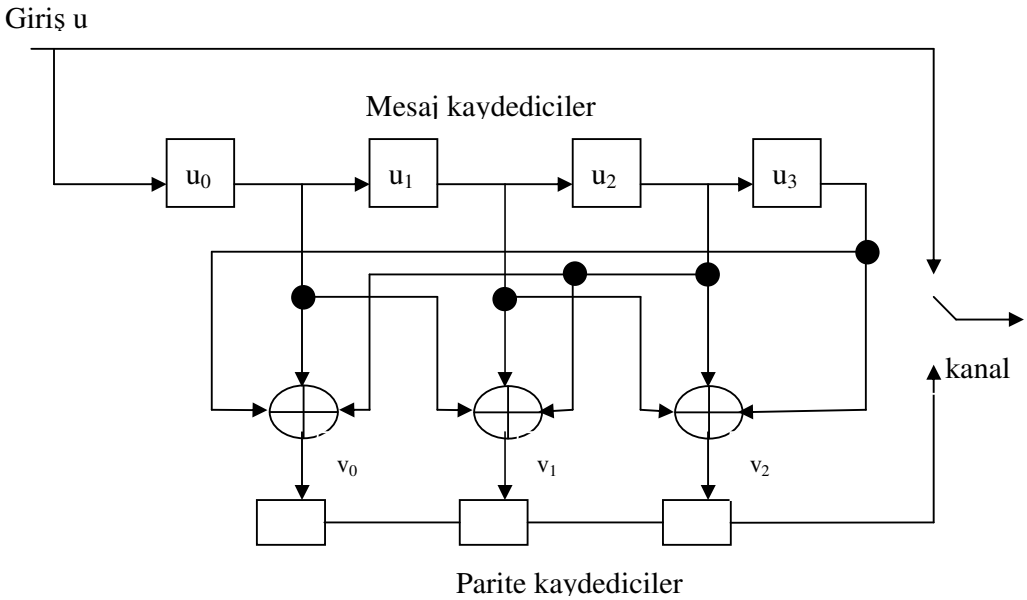
Bu denklemde **G** üreteç matrisinin boyutları *u* ve *v* dizilerinin boyutlarına bağlı olduğundan, **G** *(kxn)* boyutundadır ve *k* adet satırlarından her biri doğrusal bağımsız diziler, yani kod kelimeleridir. Bu matrisin doğrusal kombinasyonlarından oluşabilecek 2^k adet kod kelimesi vardır. Bunlardan oluşan kümeye blok kod denir ve böyle bir blok kod, **C(n, k)** blok kodu olarak tanımlanabilir. Bir *(n, k)* doğrusal blok kodu, herhangi *k* adet doğrusal bağımsız kod kelimelerinin oluşturduğu üreteç matrisi ile elde edilebilir. Yani, bir blok kodun birden fazla üreteç matrisi olabilir. Örnek 2.1 literatürde en çok karşılaşılan (Lin ve Costello, 2004; Bossert, 1999; Hamming, 1950) en basit yapıdaki olan bir (7, 4) Hamming doğrusal blok kodu ve üreteç matrisi gösterilmektedir.

Örnek 2.1: Bir (7,4) Hamming doğrusal blok kodu, (2.1) denklemiyle ikili aritmetik uzayında şöyle elde edilebilir (Lin ve Costello, 2004):

u	**v**
0000	0000000
0001	0001011
0010	0010110
0011	0011101
0100	0100111
0101	0101100
0110	0110001
0111	0111010
1000	1000101
1001	1001110
1010	1010011
1011	1011000
1100	1100010
1101	1101001
1110	1110100
1111	1111111

$$\mathbf{G} = \begin{bmatrix} 1000101 \\ 0100111 \\ 0010110 \\ 0001011 \end{bmatrix}$$

Böyle bir kodun kodlayıcı devresi aşağıdaki gibidir:



Şekil 2.1 (7,4) Hamming doğrusal blok kodu için kodlayıcı devresi.

Şekil 2.1 ile gösterilen kodlayıcı devresindeki bazı elemanları aşağıdaki gibi açıklayabiliriz:

: Bir kaydedicidir, bir sonra gelecek olan mesaj biti gelene kadar içindeki bilgiyi tutar.

: İkili toplayıcıdır.

: Lehim noktalarını belirtir.

Üreteç matrisi **G**'nin Örnek 2.1'deki şekilde ifade edilmesine "sistematik biçim" adı verilir.

$$G = \left[\begin{array}{cc} 1000 & 101 \\ 0100 & 111 \\ 0010 & 110 \\ \underbrace{0001}_{I_k} & \underbrace{011}_{P} \end{array}\right] \qquad (2.2)$$

Burada, **P** parite matrisi, $\mathbf{I_k}$ da birim alt matrisleridir. Sistematik kodlarda kodlayıcının sadece **P** alt matrisini kaydetmesi yeterlidir. Bu nedenle birim matris kısmının kayıtta tutulmasına gerek yoktur (Brink, 2001).

Tez çalışmasında üreteç matrisini sistematik hale getirebilmek için Gauss-Jordan algoritması (Noble, 1969) kullanılmıştır. Bu yöntemle ifade ettiğimiz üreteç matrisi (2.3) denkleminden de görüleceği gibi rahatlıkla **H**, parite kontrol matrisine dönüştürülebilir.

Blok kodların kod çözümünde kullanılan matris "parite kontrol matrisi" olarak adlandırılır ve genelde **H** ile temsil edilir. Parite kontrol matrisi, **H**, ait olduğu kodu üreten üreteç matrisi **G**'ye diktir ve aşağıda (2.3) denklemiyle tanımlanan özelliğe sahiptir.

$$G \cdot H^T = 0 \quad \text{veya} \quad H \cdot G^T = 0 \tag{2.3}$$

Bir blok kodun bazı önemli özelliklerini temsil eden minimum Hamming mesafesi, Hamming ağırlığı, parite kontrol matrisi gibi parametreleri vardır. Bir v kod kelimesinin ağırlığı (Hamming ağırlığı) $w(v)$ ile temsil edilir ve v'nin sıfırdan farklı elemanlarının sayısı olarak tanımlanır. Aynı uzunlukta iki ayrı kod kelimesi (v_1 ve v_2) arasındaki Hamming mesafesi veya sadece mesafesi $d(v_1,v_2)$ ile temsil edilir ve her iki kodkelimesinin birbirinden farklı bitlerinin sayısı olarak tanımlanır. Ayrıca aynı uzunlukta iki ayrı kod kelimesinin Hamming mesafesi, v_1 ve v_2 toplamından oluşan kod kelimesinin Hamming ağırlığı olarak da tanımlıdır (Lin ve Costello, 2004; Bossert, 1999; Proakis, 2000).

$$d(v_1,v_2) = w(v_1+v_2) \tag{2.4}$$

Bir doğrusal blok kodun minimum mesafesi yani Hamming mesafesi, d_{min}, o kodun kod kelimeleri içinde minimum Hamming ağırlıklı olan ağırlık değerine eşittir. Bir blok kodu **C(n,k,d)** şeklinde gösterilebilir. Burada, n kodun uzunluğu, yani kod kelimelerindeki bitlerin sayısı; k kodun boyutu, yani mesaj bitlerinin sayısı ve d ise kodun minimum Hamming mesafesidir.

3. MALZEME VE YÖNTEM

Bu çalışmada yeni bir kanal kodlama yönteminin büyük mesafeli kodlar için genelleştirilmesi ve en genel bir haberleşme sistemi olan TBGG kanalda iletimindeki hata performansının incelenmesi yapılmıştır. Bunun için MATLAB programı kullanılarak bilgisayarda benzetimler yapılmıştır. Benzetimler yapılırken İkili Faz Kaydırmalı Anahtarlama (İFKA) modülasyonu ile modülatörden gönderilen işaretler TBGG kanalda bozulduktan sonra Toplam-ürün algoritması ve iteratif kod çözme yöntemleri ile kod çözümü yapılmıştır. Bu bölümde Viterbi algoritması, Bahl, Cocke, Jelinek ve Raviv (BCJR) algoritması, Toplam-ürün algoritması ve iteratif kod çözme yöntemleri hakkında genel bir bilgi bu bölümde verilecektir. Ayrıca kod çözme ve hata başarımı, blok kodlamaların performans analizi hakkında da teorik bilgiler bu bölümde verilmiştir.

3.1 YUMUŞAK KARARLI KOD ÇÖZME VE HATA BAŞARIMI

Sert Kararlı Kod Çözme tekniğinde alıcıda alınan sembol değeri, eşik değerine göre kıyaslanarak mantıksal 1 veya 0 bitine karar verilir. Yumuşak Kararlı Kod Çözme tekniğinde ise sembol değerlerine göre elde edilen olasılıklardan yola çıkılarak karar verilir. Bu tez çalışmasında kullanılan İFKA modülasyonunda modülatör mantıksal 0 bitini -1 Volt ve mantıksal 1 bitini de +1 Volt değerine atar.

Bit Hata Oranı (BHO), veya diğer ifadeyle bit hatasının olasılığı, bir iletişim esnasında, alıcıdaki kod çözücü çıkışında hatalı olarak kodu çözülen bilgi bitlerinin, alınan toplam bilgi biti sayısına oranı olarak tanımlıdır. Bir kodun hata başarımı için en standart ölçüt BHO'dur. Bundan başka bir de Kelime Hata Oranı (KHO), veya Blok Hata Oranı, (BLHO), vardır ki bu da "kod çözücü çıkışındaki kodu çözülen kod kelimelerinin hatalı kod çözülme olasılığı" olarak tanımlanabilir.

Eğer vericiden iletilen sembolün yani elektriksel işaretin enerjisi E_s ise, gönderilen bir bilgi biti başına düşen enerji miktarı $E_b = E_s/R$ olur. Burada $R = k/n$ kod oranıdır. Buna göre başarım analizinde kullanılan temel parametre İşaret-Gürültü Oranı (İGO), E_b/N_0 olarak tanımlanır ki burada $N_0/2$ gürültü Güç Tayf Yoğunluğu (GTY)'dur. İGO veya BHO'ya bakılarak kodlama kazancı hesaplanabilir.

3.1.1 Doğrusal Blok Kodların Performans Analizi

Bu bölümdeki teorik bilgiler, İkili Faz Kaydırmalı Anahtarlama (İFKA) modülasyonunun kullanıldığı ve işaretlerin Toplamsal Beyaz Gauss Gürültülü (TBGG) kanaldan iletildiği durum için tanımlanmıştır..

ξ Sözcük başına düşen iletilen işaretin enerjisini, ξ_c de bir kod sözcüğündeki bir bitin iletilmesi için gerekli işaret enerjisini göstermektedir. Her bir kod sözcüğünde *n* bit bulunduğu için $\xi = n\xi_c$ yazılabilir. Bu arada her bir kod kelimesi de bilginin *k* bitini taşıdığından bilgi biti başına düşen enerji miktarı,

$$\xi_b = \frac{\xi}{k} = \frac{n}{k}\xi_c = \frac{\xi_c}{R_c} \tag{3.1}$$

Kod sözcüklerinin *1/M* eşit olasılıkla dağıldığı kabul edilmiştir. İFKA'da her bir kod kelimesi *M* dalga biçimlerinden biriyle sonuçlanır. En optimum alıcı, bir kod sözcüğünün ortalama olasılığını en aza indirgemek için *M* adet paralel süzgeç bankasıyla olası *M* adet iletilen dalga biçimiyle eşleştirilerek gerçeklenebilir. Her işaret aralığının sonundaki eşlenmiş *M* adet süzgecin çıktısı karşılaştırılır ve en geniş eşlenmiş süzgecin çıktısına denk gelen kod sözcüğü seçilir. Diğer bir yol da *M* adet çapraz ilgileşim kullanılmasıdır. Her iki yöntemde alıcı uygulaması basitleştirilmelidir. Bu da kod kelimesindeki her bitin iletiminde kullanılan İFKA dalga biçimine eşleştirilmiş bir süzgeç tarafından eşdeğer bir uygun değer alıcı gerçeklemesiyle yapılabilir.

Daha açık bir ifadeyle, r_j, $j=1, 2, \ldots, n$ olmak üzere herhangi belirli bir kod sözcüğü için *n* adet örneklenmiş eşlenik süzgecin çıktılarını temsil etsin. İşaretleşme İFKA olduğu için r_j çıktıları aşağıdaki iki denklemden biriyle ifade edilir. Bir kod sözcüğünün *j*. biti 0 olduğu durumda (3.2)'de olduğu gibi, 1 olduğu durumda da (3.3)'de olduğu gibi ifade edilir.

$$r_j = \sqrt{\xi_c} + n_j \tag{3.2}$$

$$r_j = -\sqrt{\xi_c} + n_j \tag{3.3}$$

$\{n_j\}$ Değişkenleri örnekleme anlarındaki TBGG'yi sembolize etmektedir ve sıfır ortalamalı, $\frac{1}{2}N_0$ değişintiye sahiptir. Olası *M* adet iletilen kod sözcüklerinin bilgisinden ve $\{r_j\}$'nin alınması üzerine, optimum kod çözücü *M* korelasyon metriğini şekillendirir:

$$CM_i = C(r, C_i) = \sum_{j=1}^{n} (2c_{ij} - 1) r_j, \quad \text{i} = 1, 2, ..., \text{M} \tag{3.4}$$

Burada c_{ij}, *i.* kod kelimesinin *j.* konumundaki biti göstermektedir. Böylece, $c_{ij} = 1$ ise ağırlık faktörü $2c_{ij} - 1 = 1$ ve $c_{ij} = 0$ ise $2c_{ij} - 1 = -1$ olduğu görülmektedir. Bu bağlamda ağırlık faktörü, iletilen kod kelimelerine karşılık gelen korelasyon metriğinin $\sqrt{\xi_c} n$ ortalama değerini, geriye kalan *M*-1 metriğin de daha küçük ortalama değerini alması için $\{r_j\}$ deki işaret bileşenlerini düzenler.

Doğrusal bir blok kodun hata olasılığının belirlenmesinde, yumuşak kararlı kod çözümünde TBGG kullanıldığında *m.* kod sözcüğünün iletimine ait hata olasılığının tüm *m*'ler için aynı olduğu dikkate alınmalıdır. Bu nedenle, işlemi basitleştirmek amacıyla tüm sıfır kod kelimeleri $\mathbf{C_1}$'in iletildiğini kabul edelim. $\mathbf{C_1}$'in doğru kod çözümü için korelasyon metriği CM_1, diğer *M*-1 korelasyon metriğini (CM_m, m = 2, … , M) aşmalıdır. Tüm *CM* değerleri gauss dağılımına sahiptir. CM_1'in ortalaması $\sqrt{\xi_c} n$, m = 2, … , M için CM_m'in ortalaması da

$\sqrt{\xi_c} n(1 - 2w_m / n)$'dir. Her bir karar değişkeninin değişintisi $\frac{1}{2} N_0$'dır. Doğru kod çözümü olasılığının tam ifadesinin elde edilmesi ya da başka bir deyişle bir kod kelimesinin hatasının olasılığı, *M* adet ilgileşim metrikleri arasından ilgileşimler tarafından karmaşıklaştırılmıştır. $\mathbf{C_1}$ ile diğer *M*-1 adet kod kelimeleri arasındaki çapraz ilgileşimler,

$$\rho_m = 1 - 2w_m / n \qquad\qquad m = 2, ..., M \tag{3.5}$$

şeklinde verilir. Burada w_m, *m*. kod kelimesinin ağırlığını göstermektedir. Hata olasılığının tam olarak elde edilmesi işlemine girişmek yerine bir sınır bağının elde edilmesine yönelelim. $CM_m > CM_1$ in olasılığı

$$P_2(m) = Q\left(\sqrt{\frac{\xi}{N_0}(1 - \rho_m)}\right) \tag{3.6}$$

Burada $\xi = k\xi_b$ dalga biçimi başına düşen iletilen enerji miktarıdır. ρ_m'nin (3.5)'deki değerini yerine yazarak ξ'nin yeni değeri

$$P_2(m) = Q\left(\sqrt{\frac{2\xi_b}{N_0} R_c w_m}\right) = Q\left(\sqrt{2\gamma_b R_c w_m}\right) \tag{3.7}$$

olur. Burada, γ_b bit başına düşen GHO, R_c de kod oranıdır. Öyleyse bir kod kelimesinin hatasının olasılığı, (3.7)'de verilen ikili hata olaylarının toplamı tarafından yukardan sınırlıdır. Sonuç olarak,

$$P_M \le \sum_{m=2}^{M} P_2(m) \le \sum_{m=2}^{M} Q\left(\sqrt{2\gamma_b R_c w_m}\right) \tag{3.8}$$

(3.8)'e göre yumuşak kararlı kod çözme için hatanın olasılığının hesabı kodun ağırlık dağılımı bilgisini gerektirir. Serbest eğri, aşağıdaki denklemden elde edilebilir:

$$Q\left(\sqrt{2\gamma_b R_c w_m}\right) \le Q\left(\sqrt{2\gamma_b R_c d_{\min}}\right) < \exp\left(-\gamma_b R_c d_{\min}\right) \tag{3.9}$$

Sonuç olarak,

$$P_M \le (M-1) Q\left(\sqrt{2\gamma_b R_c d_{\min}}\right) < \exp\left(-\gamma_b R_c d_{\min} + k \ln 2\right) \tag{3.10}$$

Bu eğri denklemi, kodun ağırlık dağılımı bilgisini içermediğinden oldukça faydalıdır. (3.10)'daki üst bant $1/2 \exp\left(-\gamma_b\right)$ ile üstten sınırlı kodlanmamış bir İFKA sistemiyle karşılaştırıldığında yaklaşık olarak kodlamanın $10\log\left(R_c d_{\min} - k \ln 2 / \gamma_b\right)$ kazancına ulaştığını görmekteyiz. Bu ifadeyi "kodlama kazancı" olarak adlandırabiliriz. Bu ifadenin kod parametrelerine ve γ_b bit başına düşen GHO olduğuna dikkat etmeliyiz.

Bir çift kodlanmış dalga biçiminin maksimum çapraz ilgileşim katsayısının

$$\rho_{\max} = 1 - \frac{2}{n} d_{\min} \tag{3.11}$$

olduğunu biliyoruz. Eğer en kötü durum olarak tüm *M* kod kelimelerinin çapraz ilgileşim katsayısının $\rho_{\max}$'a eşit olduğu yaklaşımını yaparsak kod kelimesinin hata olasılığı kolaylıkla yönetilebilir. Bazı kod kelimeleri, minimum mesafeden daha fazla uzaklıkta bulunduğundan $\rho_r = \rho_{\max}$ için hesaplanan hata olasılığı gerçekten bir üst sınırdır.

$$P_M \le 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-v^2/2} \left(\int_{-\infty}^{v+\sqrt{4\gamma_b R_c d_{\min}}} e^{-x^2/2} dx \right)^{M-1} dv \tag{3.12}$$

Yukarıdaki ifadede verilen doğrusal blok kodların performansındaki sınırlar, BLHO veya kod KHO olasılığına göredir. Buna eşdeğer BHO P_b'nin hesabı çok daha fazla karmaşıktır. Genelde, bir blok hatası yapıldığında bloktaki k adet bilgi bitlerinden bazıları doğru bazıları da hatalı olacaktır. Dikey dalga biçimleri için P_b'yi elde etmek için P_M ile çarpılan dönüşüm faktörü $2^{k-1}/\left(2^k - 1\right)$'dir. Bu çarpan, k = 1 için 1 olurken k değeri arttırıldıkça 1/2'ye yaklaşır. Burada, bir blok hatası ortaya çıktığında ortalamada k bitin yarısının hatalı olacağı yaklaşımı yapılabilir. Kodlanmış dalga biçimleri için dönüşüm faktörü karmaşık bir şekilde kodun mesafesine bağlıdır, fakat kesinlikle bir blok hatası ortaya çıktığında ortalamada k bitin yarısının hatalı olacağı yaklaşımından daha kötü değildir. Sonuç olarak, $P_b \le \frac{1}{2} P_M$ 'dir.

Kodlanmış dalga biçimini iletmek için gerekli kanal bant genişliği aşağıdaki şekilde elde edilir. Bir kod kelimesindeki her bir biti iletmek için İFKA kullanıldığında gerekli bant genişliği, yaklaşık olarak her bir bitin iletimine verilen zaman aralığının iki taraflısına eşittir. R bit/s'nin bir bilgi oranı için k adet bilgi bitini iletmek amacıyla uygun zaman ve n-k ekstra bit (toplam n adet bit) $T = k/R$ olmak üzere,

$$W = \frac{1}{T/n} = \frac{n}{k/R} = \frac{R}{R_c} \tag{3.13}$$

Bu nedenle, kodlanmış dalga biçimi için bant genişliği açılım faktörü B_e

$$\begin{aligned} B_e &= \frac{W}{R} \\ &= \frac{n}{k} = \frac{1}{R_c} \end{aligned} \tag{3.14}$$

olur.

3.2 TBGG KANAL MODELİ

Bir iletim kanalını matematiksel modellemek, bir kodun başarımını incelemek için çok faydalıdır. En genel ve en önemli kanal modeli Toplamsal Beyaz Gauss Gürültülü (TBGG) kanaldır. Kablo ve hava gibi ortamlar bu kanala örnek olarak verilebilir. Kanaldaki işaretin bozulmasının sebebi, sıfır ortalamalı, toplanır, beyaz, normal dağılımlı (gauss dağılımlı) ve varyansı $\sigma^2 = N_0/2$ olan gürültünün işarete eklenmesidir. Burada beyaz denilmesinin sebebi, gürültü gücü yoğunluğunun tüm frekans bölgesine eşit dağılmasındandır. TBGG sürekli, zamanla değişmeyen ve belleksiz bir kanaldır.

Vericiden gönderilen x vektörü kanalda rasgele gürültü vektörü n ile bozulsun. Bu durumda alıcıda alınan işaret i.anda, r_i,

$$r_i = x_i + n_i , \quad i \in [1,n] \tag{3.15}$$

Rasgele Gauss gürültü değerinin, n_i, Olasılık Yoğunluk Fonksiyonu (OYF), şöyle gösterilebilir:

$$p(n_i) = \frac{1}{\sigma\sqrt{2\pi}} e\left[-\frac{1}{2}\left(\frac{n_i}{\sigma} \right)^2 \right] \tag{3.16}$$

Bu eşitlikte (3.15) eşitliğinden elde edilen n_i değeri yerine konulursa (3.17) eşitliği elde edilir.

$$p(r_i / x_i) = \frac{1}{\sigma\sqrt{2\pi}} e\left[-\frac{1}{2}\left(\frac{r_i - x_i}{\sigma} \right)^2 \right] \tag{3.17}$$

Bu eşitlikle vericiden bir x_i işareti TBGG kanalda gönderildiğinde, alıcıda r_i değerini alma olasılığı elde edilir.

3.3 VITERBI ALGORİTMASI

1967'de Viterbi (Viterbi, 1967) tarafından bulunan Viterbi algoritması, temel olarak bir kodun kafes yapısını kullanarak en büyük olabilirlikli kod çözümünü gerçekleştiren bir kod çözme algoritmasıdır. Viterbi algoritması bir t_i anındaki durumlara giren her bir kafes patikaları ile alınan işaret değerleri arasındaki benzerliği ya da mesafeyi hesaplar. Kullanılan metrik genelde Öklid metriğidir ve "Karşılaştırılacak sayıların ayrı ayrı farklarının karelerinin toplamlarının karekökü" olarak hesaplanır. Bir anda aynı duruma giren kafes patikalarından metriği fazla olan elenir, az olan, yani en benzeri, en büyük olabilirli tercih edilen patika olarak kalır. Bu metrik karşılaştırmaları ve elemeler son duruma kadar devam eder ve en sonda en küçük metriğe sahip patika yani kodkelimesi seçilerek kodçözümü gerçeklenir. Her durumda düşük metrikli patikaları elemek işlem sayısında çok önemli miktarda azalma sağlar.

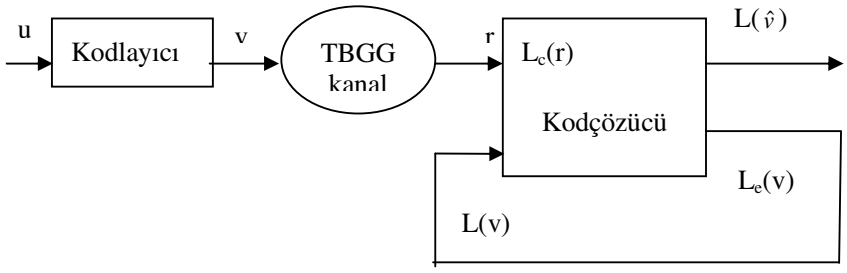
3.4 BCJR ALGORİTMASI

1974'te Bahl, Cocke, Jelinek ve Raviv (Bahl, Cocke, Jelinek, Raviv, 1974) tarafından geliştirilen bir kod çözme algoritması olduğu için BCJR algoritması olarak anılmakla birlikte MAP (Maximum a Posteriori) algoritması olarak da bilinmektedir. BCJR algoritması kod çözücüye giren yumuşak işaret değerlerini işler ve yumuşak sonuçlar üretir ki bu tip kod çözücülere "Yumuşak Girişli – Yumuşak Çıkışlı (YGYÇ) kod çözücüler" denir. Algoritma sembol hata olasılığını minimuma indirirken aynı zamanda sembol için karar verilen değerin güvenilirlik ölçüt değerini de verir. Ayrıca, algoritma alınan bir sembol dizisindeki her bir sembolü kodun kafes yapısını kullanarak ayrı ayrı kestirirken tüm sembol dizisini göz önüne alarak karar verir. BCJR algoritmasının Viterbi algoritmasına göre karmaşıklığı çok fazladır. Bu tezde yapılan hata başarımı analizlerinde Toplam-ürün algoritması ve iteratif kodçözme algoritması genel olarak kullanıldığından ve sonuçlar en çok bu iki algoritmaya göre değerlendirildiğinden BCJR ve Viterbi algoritmasının detaylarına burada girilmeyecektir.

3.5 İTERATİF KODÇÖZME ALGORİTMASI

İteratif (döngülü) kod çözme tekniği modern kanal kodlama yöntemlerinin önemli bir parçasıdır. Bu çalışmada kullanılan kodlar bir çeşit blok kodlama yöntemi olduğundan bu kısımda blok kodlar için iteratif kod çözme yöntemi açıklanacaktır. Bu tezde kullanılan iteratif kod çözme tekniği, Hagenauer tarafından ortaya konulan yöntem (Hagenauer, Offer, Papke, 1996) olmakla beraber gösterim olarak (Sklar, 1997) kaynağındakine uygun kullanılmıştır. Bu teknik önerilen kodlara Şekil 3.1 ile resmedilen iletişim sistemi modeline göre uygulandı. Şekil üzerinde görünen sembollerin başında 'L' olmasının sebebi ilgili değerlerin logaritmasının alındığını göstermek içindir. Ayrıca burada işlem kolaylığı sağlanması amacıyla kullanılan tüm logaritmalar doğal logaritmalardır.

Bu modelde de alınan *r* sembolü $r_j = v_j + n_j$ biçiminde (3.15) eşitliği ile gösterildiği gibidir. İteratif kodçözme işlemine alınan r sembol dizisinin, v kod kelime dizisi olma olasılığını logaritmik gösterimde inceleyerek başlanır. Kod kelimeleri kanala gönderilirken İFKA modülasyonu ile mantıksal 0, -1 Volt ve mantıksal 1, +1 Volt değerlerine çevrilir.



Şekil 3.1 İteratif kod çözücülü basit bir iletişim sistemi

$$L(v|r) = \log\left[\frac{P(v=+1|r)}{P(v=-1|r)}\right] = \log\left[\frac{P(r|v=+1)P(v=+1)}{P(r|v=-1)P(v=-1)}\right]$$
$$= \log\left[\frac{P(r|v=+1)}{P(r|v=-1)}\right] + \log\left[\frac{P(v=+1)}{P(v=-1)}\right] \qquad (3.18)$$

Bu durumda eşitlikteki ifadeler basit sembollerle ifade edilirse,

$$L(v|r) = L(r|v) + L(v) \qquad (3.19)$$

Bu eşitlikte $L_C(r)$ ile gösterilen ifade *r*'nin logaritmik kanal ölçütüdür. $L(v)$ ise *v*'nin önsel değeridir ki her bir bitin gönderilme olasılığı eşit olduğundan başlangıç değeri sıfırdır. (3.19) eşitliği daha basit biçimde de gösterilebilir,

$$L'(\hat{v}) = L_C(r) + L(v). \tag{3.20}$$

Bu eşitlikte ki $L_C(r)$ değeri Gauss gürültülü kanalda şöyle hesaplanır:

$$L_C(r_j) = \log\left[\frac{P(r_j | v_j = +1)}{P(r_j | v_j = -1)}\right] = \log_e\left[\frac{\frac{1}{\sigma\sqrt{2\pi}} e^{\left[-\frac{1}{2}\left(\frac{r_j - 1}{\sigma}\right)^2\right]}}{\frac{1}{\sigma\sqrt{2\pi}} e^{\left[-\frac{1}{2}\left(\frac{r_j + 1}{\sigma}\right)^2\right]}}\right] \tag{3.21}$$

$$L_C(r_j) = -\frac{1}{2}\left(\frac{r_j - 1}{\sigma}\right)^2 + \frac{1}{2}\left(\frac{r_j + 1}{\sigma}\right)^2 = \frac{2}{\sigma^2} r_j \tag{3.22}$$

Son olarak katkı değeri (extrinsic value) olarak isimlendirilen $L_e(v)$ değeri, kodçözümü esnasında her bir iterasyon yani döngü sırasında kod çözücü çıkışında elde edilir. Yumuşak kararlı kod çözücü çıkışındaki $L(\hat{v})$ değeri gerçel bir değerdir ve bu değerin işaretine bakılarak yani 0 eşik seviyesi ile kıyaslanır yani sert karar verilerek kodçözümü gerçeklenir.

$$L(\hat{v}) = L_C(r) + L(v) + L_e(v) \tag{3.23}$$

Burada $L(\hat{v})$ değerinin '+' veya '–' olması ile 1 veya 0 mantıksal değerlerine karar verilirken $L(\hat{v})$ değerinin büyüklüğü ise verilen kararın güvenilirliği hakkında bilgi verir. (3.23) denkleminde geçen $L_e(v)$ değeri *v* kodkelimesindeki bitlerin her birinin diğeriyle olan eşitliğinden yola çıkılarak bulunabilir. İteratif kod çözme yöntemi temelinde çok fazla çalışma vardır ve en çok kullanıldığı konular Turbo kodlarla ilgili olanlardır (Andersen, 1994; Robertson, 1994; Benedetto ve Montorsi, 1996; Ryan, 2002; Brink ve Ten, 1999)

3.6 TOPLAM-ÜRÜN ALGORİTMASI

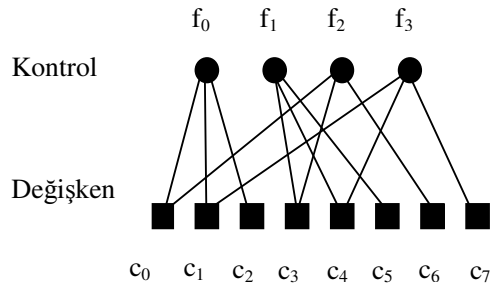
Düşük Yoğunluklu Parite Kontrol (DYPK) kodlarını 1960'ta geliştiren Gallager (Gallager, 1962), aynı zamanda iyi bir kod çözme algoritması da önermiştir. Daha sonra buna benzer algoritmalar da çeşitli uygulamalar için geliştirilmiştir (Pearl, 1998 ve Mackay, 1999). Bu kod çözme algoritması temelde, ilgili değişkenlerin iteratif olarak yani döngülü biçimde bir grafik üzerinden hesaplanması şeklinde olur. Literatürde Toplam-Ürün Algoritması (TÜA), İnanç-Yayılma Algoritması ve Mesaj Geçme Algoritması şeklinde karşımıza çıkmaktadır. Bu algoritmalarda kod çözme işlemi Tanner grafikleri yardımıyla yapılır.

Tanner, LDPC kodlarını etkin bir biçimde temsil edebilmek ve kod çözümünü gerçekleyebilmek için iki parçalı Tanner grafiğini geliştirdi (Tanner, 1981). Bu grafik, iki düğüm grubundan oluşur ve bir gruptaki bir düğüm ancak karşı gruptaki düğümlerle bağlanabilir yani kendi grubuyla direk bağlantı yapılamaz. İki düğüm grubu değişken düğümleri, v-düğümü; ve kontrol düğümleri, k-düğümü olarak isimlendirilirler. Tanner grafiği şu şekilde çizilir:

Parite kontrol matrisi **H** içindeki bir h_{ji} elemanı mantıksal 1 ise *j.* kontrol düğümü *i.* değişken düğümüne bağlanır. Örnek 3.2'de anlatılan işlem görülmektedir.

Örnek 3.2: Bir (8,4) ürün kodu parite kontrol matrisi ve Tanner çizgesi şöyledir.

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{3.24}$$



Şekil 3.2 Basit bir (8,4) ürün kodu için Tanner çizgesi.

Toplam-ürün algoritması açıklanırken yukarıdaki örnekte bulunan şekildeki parametreler ve değişken gösterimleri kullanılacaktır. Ayrıca algoritma açıklanırken (Sun, 2003) kaynağından yararlanıldığından, bu kısım kaynaktaki sembol gösterimlerine göre yapılacaktır. Buna göre enbüyük olabilirlikli kod çözmede alınan bir y dizisi için bir c_i bitinin 0 veya 1 olarak gönderilme olasılığı oranı L ise,

$$L = \frac{\Pr(c_i = 1 \mid y)}{\Pr(c_i = 0 \mid y)} \tag{3.25}$$

Bu durumda sert karar şu şekilde verilir,

$$\hat{c}_i = \begin{cases} 1 & if\ L \geq 1 \\ 0 & if\ L < 1 \end{cases} \tag{3.26}$$

Toplam-ürün algoritmasında kullanılacak değişkenler şu şekilde tanımlanabilir:

- q_{ij}: değişken düğümden, c_i, kontol düğümüne, f_j, geçilen mesajlar.
- r_{ji}: kontol düğümden, f_j, değişken düğümüne, c_i, geçilen mesajlar.
- $R_j = \{i: h_{ji} = 1\}$: j. satırdaki 1'lerin kolonlardaki yerlerinin kümesi.
- $R_{j\backslash i} = \{i': h_{ji'} = 1\}\backslash\{i\}$: i.kolonun yeri hariç, j. satırdaki 1'lerin kolonlardaki yerlerinin kümesi.
- $C_j = \{i: h_{ji} = 1\}$: i. kolondaki 1'lerin satırlardaki yerlerinin kümesi.
- $C_{i\backslash j} = \{i': h_{j'i} = 1\}\backslash\{j\}$: j.satırın yeri hariç, i. kolondaki 1'lerin satırlardaki yerlerinin kümesi.
- $p_i = \Pr(c_i = 1 \mid y_i)$

Algoritma adım adım şu şekilde yapılır:

1. Başlatma,

$$q_{ij}(0) = 1 - p_i = \Pr(c_i = 0 \mid y_i) = \frac{1}{1 + e^{-2y_i/\sigma^2}}$$
$$q_{ij}(1) = p_i = \Pr(c_i = 1 \mid y_i) = \frac{1}{1 + e^{2y_i/\sigma^2}} \tag{3.27}$$

2. Birinci yarım döngü,

$$r_{ji}(0) = \frac{1}{2} + \frac{1}{2} \prod_{i' \in R_{j\backslash i}} (1 - 2q_{i'j}(1))$$
$$r_{ji}(1) = 1 - r_{ji}(0) \tag{3.28}$$

3. İkinci yarım döngü,

$$q_{ij}(0) = K_{ij}(1-p_i)\prod_{j' \in C_{i\setminus j}} r_{j'i}(0)$$
$$q_{ij}(0) = K_{ij}p_i\prod_{j' \in C_{i\setminus j}} r_{j'i}(1) \qquad (3.29)$$

Burada k_{ij} sabitleri öyle seçilmeli ki $q_{ij}(0) + q_{ij}(1) = 1$ olmalı.

4. Yumuşak karar,

$$Q_i(0) = K_i(1-p_i)\prod_{j \in C_i} r_{ij}(0)$$
$$Q_i(1) = K_i p_i\prod_{j \in C_i} r_{ij}(1) \qquad (3.30)$$

Burada k_i sabitleri öyle seçilmeli ki $Q_i(0) + Q_i(1) = 1$ olmalı.

5. Sert karar,

$$\hat{c}_i = \begin{cases} 1 & if\ Q_i(1) > 0 \\ 0 & aksi\ halde \end{cases} \qquad (3.31)$$

Bu algoritma maksimum döngü sayısına ulaşıncaya kadar veya $\hat{c}H^T = 0$ şartı sağlanana kadar döndürülür. Günümüzde LDPC kodlarının koçözümünde ve diğer birçok alanda kullanılan bu algoritma ile ilgili literatürde birçok çalışma mevcuttur (Mackay, 1999; Richardson, Shokrollahi, Urbanke, 2001; Richardson ve Urbanke, 2001; Chung, Forney, Richardson, Urbanke, 2001; Kou, Lin, Fossorier, 2001; Mceliece, Mackay, Cheng, 1998; Kschischang ve Frey, 1998; Kschischang, Frey ve Loeliger, 2001) .

3.7 GAUSS-JORDAN YÖNTEMİYLE DOĞRUSAL DENKLEM ÇÖZÜMLERİ (Noble, 1969)

Tanım 3.7.1: Satır Basamaklı (Dizey) Biçim

Eğer bir matris aşağıdaki özellikleri sağlıyorsa "matris, satır basamaklı biçimdedir" denir:

a) eğer varsa elemanlarının tümü sıfırdan oluşan satır en alt satırdır ve
b) eğer art arda iki satır sıfırdan oluşmuyorsa ikinci satır, birinciden daha fazla sıfıra sahiptir (soldan-sağa doğru giderek)

Örneğin,

$$\begin{bmatrix} 0100 \\ 0010 \\ 0000 \\ 0000 \end{bmatrix} \tag{3.32}$$

Matrisi satır basamaklı biçimdedir. Fakat,

$$\begin{bmatrix} 0100 \\ 0100 \\ 0000 \\ 0000 \end{bmatrix} \tag{3.33}$$

matrisi satır basamaklı biçimde değildir.

Bu arada herhangi bir boyuttaki sıfır matrisi satır basamaklı biçimdedir.

Tanım 3.7.2: İndirgenmiş Satır Basamaklı (Dizey) Biçim

Eğer bir matris aşağıdaki özellikleri sağlıyorsa "matris, indirgenmiş satır basamaklı biçimdedir" denir:

1) satır basamaklı biçimde olmalı
2) tamamı sıfırlardan oluşmayan satır için ilk eleman 1 olmalı
3) 1'in bulunduğu kolonlarda diğer tüm elemanlar 0 olmalı

Örneğin,

$$\begin{bmatrix}100\\010\\001\end{bmatrix} \text{ ve } \begin{bmatrix}012002\\000103\\000014\\000000\end{bmatrix} \tag{3.34}$$

matrisleri indirgenmiş satır basamaklı biçimdedir.

Sıfır matrisi aynı zamanda indirgenmiş satır basamaklı biçimdedir.

Tanım 3.7.3: Temel Satır İşlemleri

Matrisler üzerine uygulanabilecek 3 adet temel satır işlemi vardır:

1) İki satırın yer değiştirilmesi

$$R_i \rightarrow R_j \tag{3.35}$$

2) Bir satırın sıfırdan farklı bir sayıyla çarpılması

$$R_i \rightarrow tR_i \tag{3.36}$$

3) Bir satırın herhangi bir katının diğer bir satıra eklenmesi

$$R_j \rightarrow R_j + tR_i \tag{3.37}$$

Tanım 3.7.4: Satır Eşliği

Bir **B** matrisi, bir **A** matrisinde temel satır işlemleri uygulanarak elde edilmişse "A matrisi B matrisine satır eştir" denir.

3.7.1 Gauss-Jordan Algoritması

Bu, verilen bir **A** matrisinden yine bu **A** matrisine satır-eş ve indirgenmiş satır basamaklı biçimde bir **B** matris üretme işlemidir.

1.Adım:

Soldan sağa sıfırlardan oluşmayan ilk kolonu bul (c_1 kolonu) ve bu kolondan sıfırlardan oluşmayan bir giriş seç. Eğer gerekliyse satırları değiştirerek, bu kolondaki ilk girişin sıfır olmadığından emin ol. 1.satırı a_{1c1} in çarpmaya göre tersi ile çarp. Sıfır olmayan her bir eleman $a_{ic1}, i > 1$ için eğer varsa c_1 kolonunda i.satıra $-a_{ic1}$ kat 1.satırı ekle. Böylece, c_1 kolonunda 1. eleman dışındaki tüm elemanlar sıfır olacaktır.

2.Adım:

Eğer, 1.Adım'da elde edilen matrisin 2., 3., ..., m. Satırları tamamen sıfırsa matris indirgenmiş satır basamaklı biçimdedir. Aksi takdirde, varsayın ki satırlarda sıfır olmayan elemana sahip birinci kolon, birincinin altında c_2 kolonunda olsun. O halde $c1<c2$. Eğer gerekliyse 1.'nin altında satırları değiştirerek a_{2c2} nin sıfır olmadığını sağlayın. O halde, a_{2c2}'yi 1'e çevir ve 2.satırın uygun bir katını gerekli olan diğer satırlara ekle. Böylece c_2 kolonunda kalan tüm elemanların sıfır olmasını sağla.

İşlem, tekrarlanır ve sonuçta r adım sonra ya tüm satırlar bittiği için ya da sıfır olmayan kolonlar bittiği için durur.

4. BULGULAR

Bu bölümde, temelde kodlayıcı yapısını kolaylaştıran kolay uygulanabilir, pratik, istenilen kod uzunluğunu sağlayan esneklikte ve daha da önemlisi az karmaşıklı kafes yapılarını tasarlamayı sağlayan yeni bir genel blok kod yapısı olan Geometrik İnşa (Gİ) kod matrislerinin elde edilmesi için genel bir algoritma geliştirilmiştir. Daha sonra bu matrisler sistematik hale getirilerek, kodların performans analizi yapılmıştır. Bu analizleri yapmadan önce GÜ ve Gİ kodlarının inşasına teorik (Altay, G., 2006) olarak yer verilmiştir. Buna ek olarak, önerilen kodların Toplamsal Beyaz Gauss Gürültülü (TBGG), kanallardaki bilgisayar benzetimleri ile elde edilen hata başarımları, değişik Hamming mesafeleri kullanılarak, birçok değişik kod boyutları ve değişik kod çözme yöntemleri için elde edilmiştir.

4.1 GEOMETRİK ÜRÜN KODLARI İNŞASI (Altay G., 2006)

Geometrik Ürün (GÜ) kodları yöntemiyle inşa edilen bir blok kodu $\mathbf{C = (n, k, d)}$ olarak tanımlansın; burada *n* kod uzunluğu, *k* kod boyutu yani saf bilgi bitleri sayısı ve *d* kodun minimum Hamming mesafesidir. **C** kodunu üreten üreteç matrisi **G**, üç veya daha fazla basit yapıdaki eleman kodları üreteç matrislerinin GÜ kod inşası olarak tanımlanan bir yöntemle birleşimi sonucu oluşur. Bu birleşme işlemi sonucu elde edilen ürün, üreteç matrisi **G** olduğundan, bu üreteç matrisi kullanılarak üretilen kodlar GÜ kodları olarak adlandırılmışlardır. GÜ kodu üreteç matrisi inşası genel bir ürün kodu üreteç matrisi üzerine, belirlenen başka basit yapıdaki eleman matrislerinin yine belirlenen kurallara göre eklenmesi ile gerçekleştirilir.

Genel bir ürün kodu $\mathbf{C_y = (n_y, k_y, d_y)}$ ile ve üreteç matrisi de $\mathbf{G_y}$ ile gösterilsin. $\mathbf{C_y}$ ürün kodu $\mathbf{C_1 = (n_1, k_1, d_1)}$ ve $\mathbf{C_2 = (n_2, k_2, d_2)}$ olan iki basit kodun birleşimi ile oluşur. Birleşim işlemi $\mathbf{C_1}$ ve $\mathbf{C_2}$ kodlarının $\mathbf{G_1}$ ve $\mathbf{G_2}$ üreteç matrislerinin Kronocker çarpımı sonucu oluşturulur ki bu işlem $\mathbf{G_y = G_2 \otimes G_1}$ şeklinde gösterilir. GÜ kod inşası bu şekilde elde edilen $\mathbf{G_y}$ matrisini temel alır ve ona başka bir basit eleman matrisi olan $\mathbf{G_z}$ matrisini belirli bir kurala göre ekler. Bazı durumlarda $\mathbf{G_z}$ haricinde başka eleman matrisleri de eklemek gerekebilir. Genel olarak tanıtılan GÜ kod inşası detayları şu biçimdedir:

Genel bir ürün kodu üreteç matrisi, $\mathbf{G_y}$, basit matrislerin, $\mathbf{G_1}$ ve $\mathbf{G_2}$, Kronocker çarpımları ile $\mathbf{G_y = G_2 \otimes G_1}$ şeklinde aşağıdaki gibi elde edilir. Burada $\mathbf{G_2}$ tekil parite kontrol matrisidir.

$$G_2 = \begin{bmatrix} 1 & & & 1 \\ & 1 & & 1 \\ & & \ddots & \vdots \\ & & 1 & 1 \end{bmatrix}_{k_2 \times n_2} \tag{4.1}$$

ve

$$G_y = \begin{bmatrix} G_1 & & & G_1 \\ & G_1 & & G_1 \\ & & \ddots & \vdots \\ & & G_1 & G_1 \end{bmatrix}_{k_y \times n_y} \tag{4.2}$$

Eleman matrislerinin ölçüleri belirlendiğinden ürün matrisinin, $\mathbf{G_y}$, ölçülerini bulmak kolay olacaktır. Burada $k_2 = n_2 - 1$, $k_y = k_1 \; x \; k_2$ ve $n_y = n_2 \; x \; n_1$, olmak üzere, $n_2 \geq 4$ seçilmelidir. GÜ kod inşası bu denklemler sonucu elde edilen $\mathbf{G_y}$ matrisini temel olarak kullanır ve bu matrisin geometrik olarak altına başka bir basit yapıda olan $\mathbf{G_z}$ matrisini veya eğer gerekliyse başka basit eleman matrislerini de belirli bir kurala göre yerleştirir. Bu şekilde inşa edilen GÜ kodlarının sonuç üreteç matrisi, **G**, aşağıdaki denklemde gösterilen genel yapıdadır.

$$G = \begin{bmatrix} \mathbf{G_y} & A_0 \\ G_z G_z G_z G_z & \\ \quad G_z G_z G_z G_z & A_1 \\ \qquad\qquad G_z G_z G_z G_z & \\ G_z \; G_z \; G_z \; G_z & \\ \quad G_z \; G_z \; G_z \; G_z & A_2 \\ \qquad\qquad G_z \; G_z \; G_z \; G_z & \\ G_z \qquad G_z \qquad G_z \qquad G_z & \\ \quad G_z \qquad G_z \qquad G_z \qquad G_z & A_3 \\ \cdots\cdots\cdots\cdots\cdots & \end{bmatrix} \tag{4.3}$$

Görüldüğü gibi genel bir ürün matrisi olan $\mathbf{G_y}$ altına $\mathbf{G_z}$ belli bir kurala göre yerleştirilmiştir. Bu belirlenen kurala uygun yerleştirme yapıldığında, temel alınan kodun uzunluğu ve minimum Hamming mesafesi değişmeden kodlanan bilgi biti sayısı artırılabilmekte yani kodlama oranı artırılmaktadır ki bu da iletişimde bant genişliğini etkin biçimde kullanmaya olanak verir. Belirlenen kural şu şekilde açıklanabilir:

$\mathbf{G_z}$ matrisleri sonradan eklendiği için "ekleme matrisleri", bunların yerleştirildiği matris satırları da "ekleme satırı" olarak adlandırılabilir. Ekleme satırları içerilerine yerleştirilen $\mathbf{G_z}$ matrisleri, yerleştiriliş biçimlerine göre $\{A_1, A_2,\ldots\}$ olarak gruplandırılmıştır. En üstteki A_0 grubu genel ürün matrisinin satırlarını temsil eder. Ekleme satrılarına 4 adet $\mathbf{G_z}$ matrisi yerleştirilir. Her bir ekleme satırına 4 adet $\mathbf{G_z}$ matrisini yerleştirirken dikkat edilecek önemli bir nokta, hiçbir ekleme satırı arasında 2 taneden fazla $\mathbf{G_z}$ matrisinin üst üste gelmemesidir. Ayrıca, aynı gruba ait ekleme satırları arasındaki peşpeşe gelen satırlarda 2 adet $\mathbf{G_z}$ matrisi üst üste gelmelidir. A_1 grubundaki tüm ekleme satırlarında her bir satıra 4 adet $\mathbf{G_z}$ matrisleri birbiri ardına hiç boşluk bırakılmadan yerleştirilir. A_1 grubunun ilk satırında yerleştirme ilk kolondan başlar. A_1 grubunun ikinci satırında ise peşpeşe yerleştirilen 4 adet $\mathbf{G_z}$ matrisi 2 $\mathbf{G_z}$ matrisi ölçeğinde sağa kaydırılır. Böylece iki satırdaki 2'şer adet $\mathbf{G_z}$ matrisi üst üste gelmiş olur. A_1 grubu için bu işlem 4 adet $\mathbf{G_z}$ matrisini kaydırdıkça yerleştirecek yer olduğu sürece devam eder. Eğer 4 adet $\mathbf{G_z}$ matrisini yerleştirecek kolon kalmamışsa A_1 grubu için yerleştirme işlemi sona erer. A_1 grubu için yerleştirme işlemi tamamlandıktan sonra A_2 grubu için eğer yer varsa yerleştirme işlemine başlanır. Bu durumda A_2 grubu için yerleştirilecek 4 adet $\mathbf{G_z}$ matrisi arasında birer aralık bırakılır. Buradaki 'aralıktan' kasıt bir $\mathbf{G_z}$ ölçeği kadarlık yerdir. A_2 grubunun ilk satırında da yerleştirme ilk kolondan başlar. A_2 grubunun ikinci satırında ise artarda yerleştirilen 4 adet birer aralıklı $\mathbf{G_z}$ matrisi, 4 $\mathbf{G_z}$ matrisi ölçeğinde sağa kaydırılır. Böylece aynı grupta bulunan peş peşe iki satır arasındaki 2'şer adet $\mathbf{G_z}$ matrisleri üst üste gelmiş olur. Bu işlemde bir öncekine benzer biçimde 4 adet $\mathbf{G_z}$ matrisini yerleştirecek kolon kalmayana kadar devam eder. A_2 grubu için yerleştirme işlemi tamamlandıktan sonra A_3 grubu için eğer yer varsa yerleştirme işlemine başlanır. Bu durumda ise A_3 grubu için yerleştirilecek 4 adet $\mathbf{G_z}$ matrisi arasında üçer aralık bırakılır. Yerleştirme işlemi öncekiler gibi yapılmakla beraber sağa kaydırma 8 aralık yapılır. Bu işlemler aynı şekilde, eğer yerleştirmeye yer varsa, diğer gruplar içinde devam eder. Tüm yerleştirme işlemi 4 adet $\mathbf{G_z}$ matrisini yerleştirecek ekleme satırı kalmayınca sona erer. Genel bir kural olarak, sağa kaydırma işlemi A_1 grubu için 2, A_2 grubu için 4, A_3 grubu için 8 daha sonraki gruplar için 8, 16,... ve bu şekilde 2'nin kuvvetleri halinde devam eder. 4 adet $\mathbf{G_z}$ matrisi arasında bırakılan aralıklar ise A_1 grubu için 0, A_2 grubu için 1, daha sonraki gruplar için 3, 7, 15,... ve bu şekilde kaydırma miktarının yarısından 1 eksik olarak devam eder.

Buraya kadar açıklanan kurallar uygun eleman matrisleri seçildiğinde eleman matrislerinin nasıl yerleştirildiği ile ilgiliydi. Eleman matrislerinin nasıl seçileceği ile ilgili genel kurallar koyulmamakla birlikte bunların nasıl seçileceğini en başta istenilen kodun uzunluğu belirler. İstenilen uzunluklu koda göre $\mathbf{C_1}$ ve $\mathbf{C_2}$ kodlarının üreteç matrisleri birleştirilerek genel ürün

matrisi yapısı elde edilir. $\mathbf{G_z}$ matrisi seçilirken dikkat edilmesi gereken konu ise 4 adet $\mathbf{G_z}$ matrisinde bulunan mantıksal 1'lerin toplamı, kodun istenilen minimum Hamming mesafesine eşit olmasıdır. Bu konuda daha sonraki kısımlarda açıklayıcı örnekler verilecektir. Bunlara ek olarak, elde edilen GÜ kod yapısı bilgi bitleri ve parite kontrol bitleri ayırt edilebildiği için sistematik bir yapıdadır.

4.2 HAMMING MESAFESİ 4 OLAN GEOMETRİK ÜRÜN KOD AİLESİ

GÜ kodlamasının (4.3) denklemiyle önerilen yapısı kullanılarak, minimum Hamming mesafesi 4 olan eniyi özellikli çift kodlar elde edilmiştir. Buradaki 'eniyi' kavramının anlamı minimum Hamming mesafesi 4 olan çift kodların alabileceği en büyük uzunluk, *n*, ve boyut, *k*, parametrelerinin bu kodlama yöntemi ile elde edilebilmesidir. Blok kodların belli bir minimum Hamming mesafesi için alabileceği en iyi yani maksimum uzunluk ve boyut değerleri 'en iyi-bilinen kodlar tablosu' olarak (Brouwer ve Verhoeff, 1993) literatürde yayınlanmış olup, bu tezde elde edilen kodların en iyi kodlardan olup olmadığına karar vermek için bu tablo kullanılmıştır.

Minimum Hamming mesafesi 4 olan en iyi çift kodların inşası için, eleman matrisleri $C_1 = (2, 1, 2)$ ve $C_z = (2, 1, 1)$ boyutlarındaki basit kodlar seçilmiş olup bu kodların üreteç matrisleri $\mathbf{G_1 = [1\ 1]}$ ve $\mathbf{G_z = [1\ 0]}$ olan en basit yapıdaki matrislerdir. Temel eleman kodlarından olan C_2 kodunun üreteç matrisi önceki bölümde de açıklandığı gibi tekil-parite kontrol matrisi olup bu kodun uzunluğunu temsil eden n_2 değeri 4'ten büyük veya 4'e eşit istenilen çift tamsayı değerini alabilir. (4.1) ve (4.2) denkleminden de görülebileceği gibi n_2 değerine farklı değerler verilerek istenilen kod uzunluğu elde edilebilir. Çünkü $\mathbf{G_1 = [1\ 1]}$ gibi basit bir matris olarak seçilip değiştirilmediğinden $\mathbf{G_y}$ matrisini oluşturan diğer matris $\mathbf{G_2}$ değiştirilerek $\mathbf{G_y}$ temel matrisinin uzunluğu ayarlanabilir. Buradan elde edilen uzunluk, GÜ kodlarının uzunluk değeri *n*'ye eşittir. Çünkü $n = n_2 \times n_1$ olduğu matris özelliklerinden görülmektedir. Sonuç matrisinin sadece boyutu, *k*, ekleme matrisleri ile artırılır. Yani n_2 değeri istenilen kodun uzunluğunu elde edebilmek için ilk başta ayarlanması gereken temel parametredir. Bundan başka herhangi bir parametreyi keyfi değiştirebilme sözkonusu değildir. Bir önceki bölümde açıklanan GÜ kod inşası kurallarına uyularak ve bu bölümde belirlenen eleman matrisleri kullanılarak minimum Hamming mesafesi 4 olan en iyi yapıda bir GÜ kodu üreteç matrisi (4.3) denkleminde olduğu gibi otomatik olarak inşa edilebilir.

Bu şekilde minimum Hamming mesafesi 4 olan, uzunluğu 8'den büyük ve çift olan, eniyi yapıda GÜ kodları bir kod ailesi olarak elde edilmiştir. Bu kod ailesinin ölçülerini formülleştirebilmek için çok sayıda üreteç matrisi bilgisayar programı vasıtasıyla oluşturulmuş olup, bu matrislerin ölçüleri kaydedilmiş ve kod parametreleri *n, k* ve *d* arasındaki ilişki şu şekilde formülleştirilmiştir.

$$C = (n, k, d) = \left(n, n - \lceil \log_2 n + 1 \rceil, 4\right) . \tag{4.4}$$

Burada *n* değeri 8'den büyük veya 8'e eşit çift bir tamsayı olup $\lceil b \rceil$ gösterimindeki işlem de *b* değerini pozitif sonsuza doğru en yakın tamsayıya yuvarlar. Bu denklemde görülen minimum Hamming mesafesinin 4 değerine eşit olduğu Teorem 4.1 ile aşağıda ispatlanmaktadır.

Teorem 4.1: Eğer eleman matrisleri $\mathbf{G_1}$ **= [1 1]** ve $\mathbf{G_2}$ **= [1 0]** seçilirse, o zaman (4.3) denklemiyle inşa edilen bir GÜ üreteç matrisi, **G**, ile üretilen bir **C** kodunun minimum Hamming mesafesi 4 değerine eşittir.

İspat: Bu teoremi ispatlamak için **G** matrisi geometrik yerleştirilmelerine göre gruplandırılan satır gruplarından oluştuğundan, bu grupların kendisi ve grupların birbirleriyle olan ilişkilerinin incelenmesi gerekir. (4.3) denkleminde matris inşasının yerleştirme kuralından dolayı hiçbir durumda **G** matrisinin herhangi iki matris satırı arasında 2 adetten fazla eleman matrisleri üstüste gelmez. A_0 grubunda en fazla 1 adet $\mathbf{G_1}$ matrisi üstüste gelir ve dolayısıyle her bir satırdan 1'er adet $\mathbf{G_1}$ matrisi herhangi bir ikili toplama işleminde boşta kalır, yani etkilenmez. Herbir $\mathbf{G_1}$ matrisi 2 adet mantıksal 1 içerdiğinden A_0 grubunun doğrusal kombinasyonları sonucu oluşabilecek herhangi bir kodkelimesinin minimum Hamming mesafesi 4 değerine eşittir. Kalan satırlar A_i (i = 1, 2, …) gruplarına ait olup her bir satır 4 adet $\mathbf{G_2}$ matrisi içerir ve bu gruplardaki herhangi iki satır arasında da en fazla 2 adet $\mathbf{G_2}$ matrisi üstüste gelir, yani herhangi iki satır arasındaki ikili toplamada 4 adet $\mathbf{G_2}$ matrisi bozulmadan sonuç satırda kalır. Her bir $\mathbf{G_2}$ matrisi 1 adet mantıksal 1 içerdiğinden en kötü durumda A_i gruplarındaki satırların doğrusal kombinasyonları sonucu oluşabilecek herhangi bir kodkelimesinin minimum Hamming mesafesi 4 değerine eşittir. İncelenmeyen tek durum A_0 ve A_i grupları arasındaki ilişkidir. Bu durumda da en kötü durumda her iki grup arasında üst üste gelebilecek matris sayısı en fazla 2'dir. Bu en kötü durumda oluşabilecek sonuçtaki kodkelimesinde 2 adet $\mathbf{G_2}$ matrisi (yani toplam 2 adet mantıksal 1) ve 2 adet $\mathbf{G_1} \oplus \mathbf{G_2}$ matrisi (**[1 1]** $\oplus$ **[1 0] = [0 1]**) yani yine toplam 2 adet mantıksal 1 ve toplam olarak bu satırda 4 adet mantıksal 1 olur ki bu da satırın minimum Hamming mesafesinin 4 değerine eşit

olduğunu gösterir. Dolayısıyla (4.3) denklemiyle inşa edilen bir üreteç matrisi, **G**, ile üretilen bir **C** kodunun minimum Hamming mesafesi 4 değerine eşittir.

Şimdiye kadar anlatılan kod inşasının daha açık olarak gösterilmesi için Örnek 4.1 verilmiştir.

Örnek 4.1: Eğer kod uzunluğu n = 52 ve minimum Hamming mesafesi 4 olan bir GÜ kodu üreteç matrisi, **G**, inşa edilmek istenirse, o zaman n_2 öyle seçilir ki $n_2 = 52/n_1 = 52/2 = 26$ olur. Ayrıca $\mathbf{G_1 = [1\ 1]}$ ve $\mathbf{G_z = [1\ 0]}$ olduğu bilinmekte olup $\mathbf{G_y}$ matrisi (4.2) denkleminde olduğu gibi oluşturulduğun da (25 x 52) ölçülerinde bir temel matris elde edilir. Bundan sonra temel matrisin altına $\mathbf{G_z}$ eleman matrislerini ekleme işlemi (4.3) denkleminde gösterildiği gibi yapılır. Aşağıda sonuçta elde edilen **G** üreteç matrisi görülmekte olup, burada $\mathbf{G_z}$ basitlik açısından sadece *z* harfi ile temsil edilmektedir. Sonuçta (4.3) denklemi ile elde edilen GÜ kod üreteç matrisi (45x52) ölçülerinde olup 4 ekleme grubundan oluşmaktadır. İlk grup olan A_1 grubunda 12 satır eklenebilmektedir ki buradaki sağa kaydırma 2 aralıklıdır ve eleman matrisleri arasında aralık yoktur. İkinci grup olan A_2 grubunda 5 satır eklenebilmektedir ki buradaki sağa kaydırma 4 aralıklıdır ve eleman matrisleri arasında 1 aralık vardır. A_3 grubunda 2 satır eklenebilmektedir ki buradaki sağa kaydırma 8 aralıklıdır ve eleman matrisleri arasında 3 aralık vardır. Son olarak A_4 grubunda 1 satır eklenebilmektedir ki buradaki eleman matrisleri arasında 7 aralık vardır ve sağa kaydırma yer kalmadığından olmamıştır. Bu aşamadan sonra $\mathbf{G_z}$ matrislerini kurala uygun biçimde yerleştirecek yer kalmadığından işlem tamamlanmıştır. Toplamda GÜ kod inşası tekniği ile genel bir ürün kodu matrisi üzerine 20 satır eklenebilmiştir ki bu da elde edilebilecek maksimum boyuta ulaşabilmeyi sağlamış ve eniyi kod ölçülerinde, $C_1 = (52, 45, 4)$, bir GÜ blok kodu elde etmeyi sağlamıştır.

$$G=\begin{bmatrix} G_{y\ \{25x52\}} \\ \mathbf{ZZZZ} \\ \mathbf{\ ZZZZ} \\ \mathbf{\ \ ZZZZ} \\ \mathbf{\ \ \ ZZZZ} \\ \mathbf{\ \ \ \ ZZZZ} \\ \mathbf{\ \ \ \ \ ZZZZ} \\ \mathbf{\ \ \ \ \ \ ZZZZ} \\ \mathbf{\ \ \ \ \ \ \ ZZZZ} \\ \mathbf{\ \ \ \ \ \ \ \ ZZZZ} \\ \mathbf{\ \ \ \ \ \ \ \ \ ZZZZ} \\ \mathbf{Z\ Z\ Z\ Z} \\ \mathbf{\ \ Z\ Z\ Z\ Z} \\ \mathbf{\ \ \ Z\ Z\ Z\ Z} \\ \mathbf{\ \ \ \ Z\ Z\ Z\ Z} \\ \mathbf{\ \ \ \ \ Z\ Z\ Z\ Z} \\ \mathbf{Z\ \ \ Z\ \ \ Z\ \ \ Z} \\ \mathbf{\ \ \ Z\ \ \ Z\ \ \ Z\ \ \ Z} \\ \mathbf{Z\ \ \ \ Z\ \ \ \ Z\ \ \ \ Z} \end{bmatrix} \begin{matrix} \\ \end{matrix} \begin{matrix} A_0 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ A_1 \\ \\ \\ \\ \\ \\ \\ A_2 \\ \\ \\ \\ A_3 \\ \\ A_4 \end{matrix} \qquad (4.5)$$

Genel olarak minimum Hamming mesafesi 4 olan GÜ kodları tanıtıldıktan sonra bu kodlardan bazılarının üreteç matrisleri aşağıdaki örneklerde verilmektedir.

Örnek 4.2: C = (12, 7, 4) olan bir GÜ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix} 110000000011 \\ 001100000011 \\ 000011000011 \\ 000000110011 \\ 000000001111 \\ 101010100000 \\ 000010101010 \end{bmatrix} \tag{4.6}$$

Örnek 4.3: C = (14, 9, 4) olan bir GÜ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix} 11000000000011 \\ 00110000000011 \\ 00001100000011 \\ 00000011000011 \\ 00000000110011 \\ 00000000001111 \\ 10101010000000 \\ 00001010101000 \\ 10001000100010 \end{bmatrix} \tag{4.7}$$

Örnek 4.4: C = (16, 11, 4) olan bir GÜ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix} 1100000000000011 \\ 0011000000000011 \\ 0000110000000011 \\ 0000001100000011 \\ 0000000011000011 \\ 0000000000110011 \\ 0000000000001111 \\ 1010101000000000 \\ 0000101010100000 \\ 0000000010101010 \\ 1000100010001000 \end{bmatrix} \tag{4.8}$$

Örnek 4.5: C = (40, 33, 4) olan bir GÜ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix}
1100000000000000000000000000000000000011 \\
0011000000000000000000000000000000000011 \\
0000110000000000000000000000000000000011 \\
0000001100000000000000000000000000000011 \\
0000000110000000000000000000000000000011 \\
0000000001100000000000000000000000000011 \\
0000000000110000000000000000000000000011 \\
0000000000001100000000000000000000000011 \\
0000000000000110000000000000000000000011 \\
0000000000000001100000000000000000000011 \\
0000000000000000110000000000000000000011 \\
0000000000000000001100000000000000000011 \\
0000000000000000000110000000000000000011 \\
0000000000000000000001100000000000000011 \\
0000000000000000000000110000000000000011 \\
0000000000000000000000011000000000000011 \\
0000000000000000000000001100000000000011 \\
0000000000000000000000000110000000000011 \\
0000000000000000000000000011000000000011 \\
0000000000000000000000000001100000000011 \\
0000000000000000000000000000110000000011 \\
0000000000000000000000000000011000000011 \\
0000000000000000000000000000001100000011 \\
0000000000000000000000000000000110000011 \\
0000000000000000000000000000000011000011 \\
0000000000000000000000000000000000110011 \\
0000000000000000000000000000000000001111 \\
1010101000000000000000000000000000000000 \\
0000101010100000000000000000000000000000 \\
0000000010101010000000000000000000000000 \\
0000000000001010101000000000000000000000 \\
0000000000000000101010100000000000000000 \\
0000000000000000000010101010000000000000 \\
0000000000000000000000101010100000000000 \\
0000000000000000000000001010101000000000 \\
0000000000000000000000000010101010000000 \\
0000000000000000000000000000101010100000 \\
0000000000000000000000000000001010101010 \\
1000100010001000000000000000000000000000 \\
0000000010001000100010000000000000000000 \\
0000000000000000100010001000100000000000 \\
0000000000000000000000001000100010001000 \\
1000000010000000100000001000000000000000
\end{bmatrix} \tag{4.9}$$

4.3 DAHA BÜYÜK HAMMING MESAFELİ GEOMETRİK ÜRÜN KODLARI İNŞASI

GÜ kod inşası, uygun eleman üreteç matrislerinin seçildiği varsayılarak bu matrislerden oluşturulacak bir büyük üreteç matrisinin elde edilmesi amacıyla bu eleman matrislerinin nasıl yerleştirileceğine dair kesin ve genel geometrik kurallar sağlar. Fakat kullanılacak eleman matrislerinin seçilmesi için kesin ve genel kurallar koymaz. Sonuç üreteç matrisinin istenilen ölçülerine göre gerekli eleman matrislerinin boyutları büyük ölçüde bellidir fakat kesin ölçütler olmadığından eleman matrisleri denenerek bulunmalıdır. Bu şekilde elde edilen eleman matrisleri kullanılarak elde edilen minimum Hamming mesafesi 4 olan eniyi kodlar bir önceki bölümde sunulmuştu. Bu bölümde ise minimum Hamming mesafesi 8 olan kodların inşası için gerekli eleman matrisleri ve ayrıca daha da büyük Hamming mesafeli kodların elde edilmesi için gerekli eleman üreteç matrisleri gösterilecek ve bu eleman matrisleri seçilirken nelere dikkat edilmesi gerektiğinden bahsedilecektir. GÜ kod inşası ile elde edilen 4'ten büyük Hamming mesafeli kodlardan bazıları en iyi kodlardan olup bazıları en iyi değildir. Yani bu durumda 4 Hamming mesafeli kodlarda olduğu gibi tüm kodlar en iyi kodlardan olmayabilir. Öncelikle minimum Hamming mesafesi 8 olan kodların inşası için gerekli eleman matrisleri sunulacaktır.

$$G_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \tag{4.10}$$

ve

$$G_z = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} \tag{4.11}$$

olarak seçilmelidir. Bu eleman matrisleri kullanılarak ve (4.3) denklemiyle gösterilen GÜ kod inşası yönteminden yararlanılarak (16, 5, 8), (24, 9, 8), (32, 15, 8) kodlarının üreteç matrisleri inşa edilmiş ve kodların parametrelerinin doğruluğunun sağlaması bilgisayar programı yardımıyla test edilmiştir. Diğer taraftan inşa edilen kodların uzunlukları artırıldıkça, inşa edilen kodların ölçüleri eniyi olma özelliğinden gitgide sapmaktadır. Bu durumu bertaraf etmek için kod inşa yöntemi geliştirilmeye çalışılmış ve genel kod inşasına ufak bir ekleme ile (32,16,8) eniyi kodu elde edilebilmiştir. Maalesef, her kod uzunluğu için bu durumu genelleştirebilmek mümkün olmamıştır. Buradan şu sonuca varıldı ki GÜ kod inşası, kendi üzerinde yapılabilecek ufak eklemeler ile daha büyük minimum Hamming mesafeli kodlarda da en iyi kodları üretebilme potansiyeline sahiptir. Elde edilen (32, 16, 8) geliştirilmiş GÜ kodu inşası Örnek 4.6 ile aşağıda gösterilmektedir.

Örnek 4.6: C = (32, 16, 8) eniyi ölçütlerindeki bir GÜ blok kodunu üretebilmek için önce n_2 değeri $n_2 = 32/n_1 = 32/4 = 8$ olarak hesaplanır ki burada n_1 (4.10) eşitliğinden görüldüğü gibi 4'tür. Daha sonra $\mathbf{G_2}$ and $\mathbf{G_y}$ matrisleri (4.1) ve (4.2) deki gibi oluşturulur. $\mathbf{G_1}$ ve $\mathbf{G_z}$ (4.10) ve (4.11) eşitliklerinde belirtildiği gibi seçilir. En son olarak bu koda özel geliştirilmiş GÜ kod inşasında kullanılmak üzere bir başka eleman matrisi $\mathbf{G_3}$ şu şekilde seçilir.

$$G_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \tag{4.12}$$

Son olarak bu eleman matrisleri aşağıda (4.13) eşitliğinde görüldüğü gibi yerleştirilerek sonuç üreteç matrisi inşa edilir.

$$G = \begin{bmatrix} G_{y\ \{7x32\}} \\ G_z G_z G_z G_z \\ G_z G_z G_z G_z \\ G_z G_z G_z G_z \\ G_z \quad G_z \quad G_z \quad G_z \\ G_3 G_3 G_3 G_3 G_3 G_3 G_3 G_3 \end{bmatrix}_{\{16x32\}} \tag{4.13}$$

Daha da büyük minimum Hamming mesafeli kodlar için eleman matrislerini seçme işlemi daha da güçleşmektedir. Aşağıda minimum Hamming mesafesi 16 olan kodları üretmek için seçilmesi gereken eleman matrisleri gösterilmektedir.

$$G_3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{4.14}$$

ve

$$G_z = \begin{bmatrix} 11110000 \\ 00111100 \\ 10101010 \end{bmatrix} \tag{4.15}$$

Bu eleman matrislerini kullanarak ve (4.3) ile gösterilen GÜ kod inşasından yararlanılarak (64, 19, 16) GÜ kodu üretilmiş ve parametrelerin doğruluğunun sağlaması bilgisayar programı ile test edilmiştir. Aşağıda verilecek örnekler ile büyük minimum Hamming mesafeli kodların üreteç matrislerinden bazıları sunulacaktır.

Örnek 4.7: C = (16, 5, 8) olan bir GÜ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix} 1111000000001111 \\ 0000111100001111 \\ 0000000011111111 \\ 1100110011001100 \\ 0110011001100110 \end{bmatrix} \tag{4.16}$$

Örnek 4.8: C = (24, 9, 8) olan bir GÜ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix} 111100000000000000001111 \\ 000011110000000000001111 \\ 000000001111000000001111 \\ 000000000000111100001111 \\ 000000000000000011111111 \\ 110011001100110000000000 \\ 011001100110011000000000 \\ 000000001100110011001100 \\ 000000001100110011001100 \end{bmatrix} \tag{4.17}$$

Örnek 4.9: C = (32, 15, 8) olan bir GÜ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix} 11110000000000000000000000001111 \\ 00001111000000000000000000001111 \\ 00000000111100000000000000001111 \\ 00000000000011110000000000001111 \\ 00000000000000001111000000001111 \\ 00000000000000000000111100001111 \\ 00000000000000000000000011111111 \\ 11001100110011000000000000000000 \\ 01100110011001100000000000000000 \\ 00000000110011001100110000000000 \\ 00000000110011001100110000000000 \\ 00000000000000001100110011001100 \\ 00000000000000001100110011001100 \\ 11000000110000001100000011000000 \\ 01100000011000000110000001100000 \end{bmatrix} \tag{4.18}$$

Örnek 4.10: C = (48, 25, 8) olan bir GÜ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix}
111100000000000000000000000000000000000000001111 \\
000011110000000000000000000000000000000000001111 \\
000000001111000000000000000000000000000000001111 \\
000000000000111100000000000000000000000000001111 \\
000000000000000011110000000000000000000000001111 \\
000000000000000000001111000000000000000000001111 \\
000000000000000000000011110000000000000000001111 \\
000000000000000000000000111100000000000000001111 \\
000000000000000000000000001111000000000000001111 \\
000000000000000000000000000011110000000000001111 \\
000000000000000000000000000000111100000000001111 \\
000000000000000000000000000000001111000000001111 \\
000000000000000000000000000000000011110000001111 \\
000000000000000000000000000000000000111100001111 \\
000000000000000000000000000000000000001111111111 \\
110011001100110000000000000000000000000000000000 \\
011001100110011000000000000000000000000000000000 \\
000000011001100110011000000000000000000000000000 \\
000000001100110011001100000000000000000000000000 \\
000000000000000011001100110011000000000000000000 \\
000000000000000011001100110011000000000000000000 \\
000000000000000000000000110011001100110000000000 \\
000000000000000000000000011001100110011000000000 \\
000000000000000000000000001100110011001100000000 \\
000000000000000000000000000011001100110011000000 \\
000000000000000000000000000000110011001100110000 \\
000000000000000000000000000000011001100110011000 \\
110000001100000011000000110000000000000000000000 \\
011000000110000001100000011000000000000000000000 \\
000000000000000011000000110000001100000011000000 \\
000000000000000001100000011000000110000001100000
\end{bmatrix} \tag{4.19}$$

Örnek 4.11: C = (64, 19, 16) olan bir GÜ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix}
1111111100000000000000000000000000000000000000000000000011111111 \\
0000000011111111000000000000000000000000000000000000000011111111 \\
0000000000000000111111110000000000000000000000000000000011111111 \\
0000000000000000000000001111111100000000000000000000000011111111 \\
0000000000000000000000000000000011111111000000000000000011111111 \\
0000000000000000000000000000000000000000111111110000000011111111 \\
0000000000000000000000000000000000000000000000001111111111111111 \\
1111000011110000111100001111000000000000000000000000000000000000 \\
0011110000111100001111000011110000000000000000000000000000000000 \\
1010101010101010101010101010100000000000000000000000000000000000 \\
0000000000000000111100001111000011110000111100000000000000000000 \\
0000000000000000111100001111000011110000111100000000000000000000 \\
0000000000000000101010101010101010101010101000000000000000000000 \\
0000000000000000000000000000000011110000111100001111000011110000 \\
0000000000000000000000000000000011110000111100001111000011110000 \\
0000000000000000000000000000000010101010101010101010101010101010 \\
1111000000000000111100000000000011110000000000001111000000000000 \\
0011110000000000111100000000000011110000000000001111000000000000 \\
1010101000000000101010100000000010101010000000001010101000000000
\end{bmatrix} \quad (4.20)$$

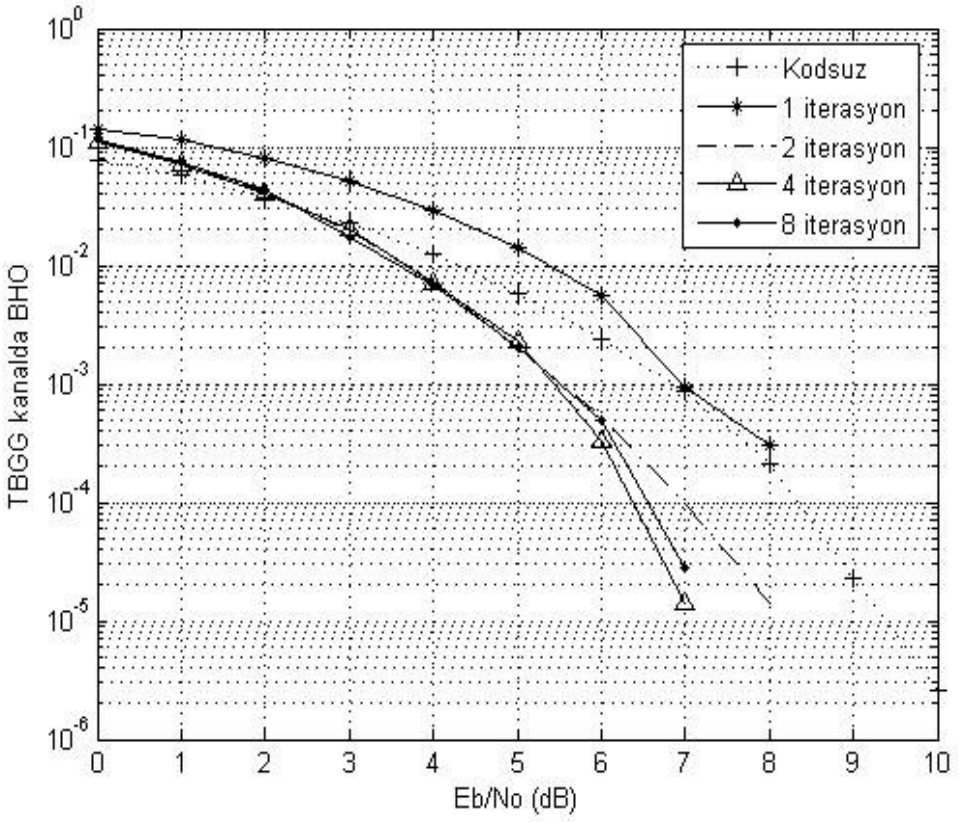
4.4 GEOMETRİK ÜRÜN KODLARI İÇİN BİLGİSAYAR BENZETİMİ İLE HATA BAŞARIMI

Bilgisayar benzetimleri, sıfır ortalamalı ve varyansı $\sigma^2 = N_0/2$ olan, Toplamsal Beyaz Gauss Gürültülü (TBGG) kanal için gerçeklenmiştir. Burada $N_0/2$ çift taraflı Güç Spektrum Yoğunluğudur (GSY). Bazı GÜ kodları için bit hata başarımları E_b/N_0 (desibel biriminde) ve Bit Hata Oranı (BHO) parametrelerine göre elde edilmiş olup burada E_b bir bilgi biti başına düşen enerji miktarıdır. Bilgisayar benzetimleriyle gerçeklenen iletişim modelinde GÜ kodları İkili Faz Kaydırmalı anahtarlama (İFKA) modülasyonu kullanılarak mantıksal 1 biti +1 ve mantıksal 0 biti -1 değerlerine atanır. Alıcı kısımda, ideal kanal durum bilgisinin bulunduğu varsayılmış ve iteratif kod çözme kullanılarak başarım analizi de bazı kodlar için yapılmıştır. İncelenen tüm GÜ kodların mükemmel bit ve blok senkronizasyonu (eşzamanlama) olduğu varsayılmıştır. Bu açıklanan haberleşme modelinde bazı GÜ kodları için elde edilen bit hata oranları için eğriler aşağıdaki şekillerde çizdirilmişir. Ayrıca karşılaştırma için kullanabilmek

amacıyla 11 adet bitlerden oluşan bloklar İFKA modülasyonu gerçeklendikten sonra kodlayıcı ve kod çözücü kullanılmadan BHO değerleri elde edilmiş ve tüm eğrilerde kodlanmamış İFKA modülasyonunu temsil etmek üzere çizdirilmiştir. Bir örnek olarak bunun nasıl kullanılacağını göstermek gerekirse, mesela (16, 11, 4) blok kodu kodlayıcı kullanıldığında, kodlanmamış duruma göre, 2.6 dB (desibel) kodlama kazancı sağlar.

4.4.1 İteratif Kod Çözümüyle Geometrik Ürün Kodunun Hata Başarımı

(12, 7, 4) GÜ kodu için, iteratif kod çözme algoritması kullanılarak, en büyük olabilirlikli yumuşak kararlı kod çözümüyle elde edilen, TBGG kanal ortamında ki BHO başarımı farklı iterasyon sayıları için aşağıdaki şekilde sunulmuştur.

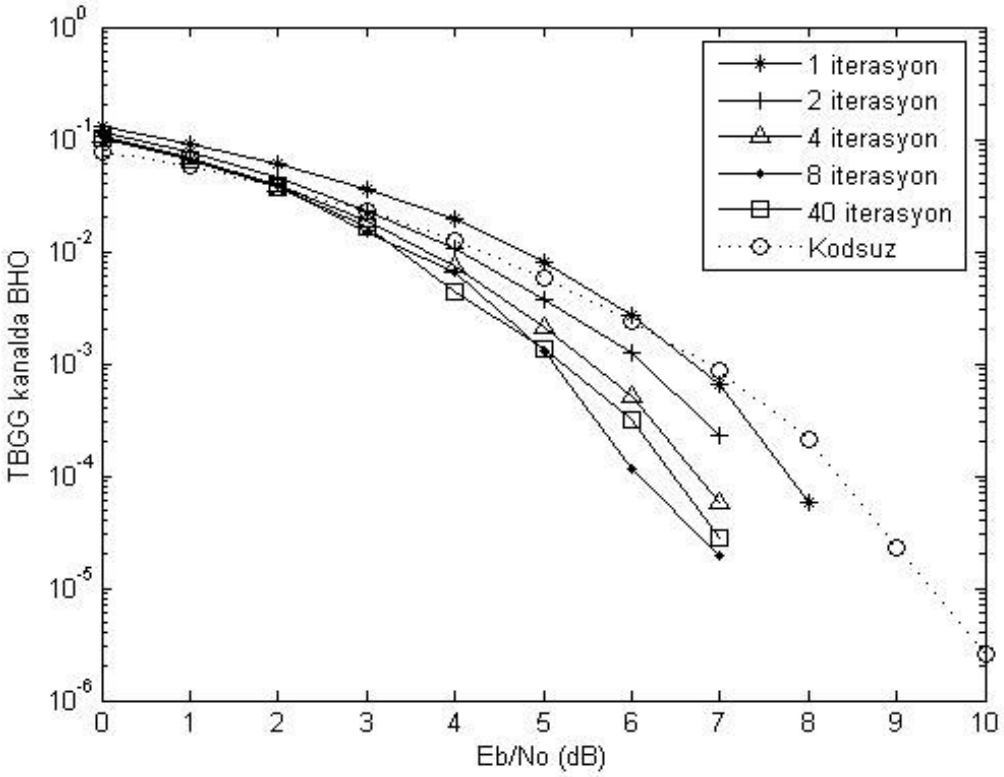


Şekil 4.1 C = (12, 7, 4) GÜ kodunun ve kodlanmamış bitlerin (kodsuz) TBGG kanalda iteratif kod çözme algoritması yardımıyla elde edilen BHO.

Şekilden de görüldüğü gibi iteratif yöntemle kodçözümü gerçeklenen (12,7,4) GÜ kodu en iyi başarım eğrisine hızlı bir şekilde 4 iterasyon sonucu yakınsar. Daha büyük uzunluklu kodlar için bu kod çözme algoritmasının kullanılması aşırı karmaşıklığından dolayı uygun olmadığından kullanılmamıştır.

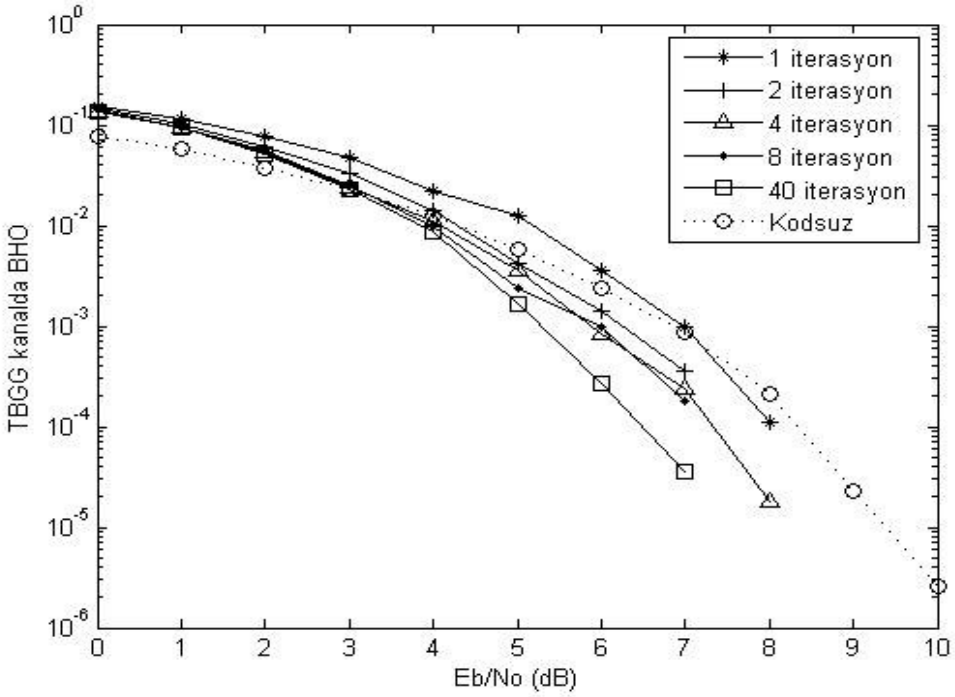
4.4.2 Toplam-Ürün Algoritması Kod Çözümüyle Geometrik Ürün Kodlarının Hata Başarımı

Bu bölümde bazı GÜ kodları için Toplam-Ürün Algoritması yardımıyla en büyük olabilirlikli yumuşak kararlı kod çözümüyle elde edilen, TBGG kanal ortamında ki BHO başarımları bu bölümdeki şekillerle gösterilmiştir

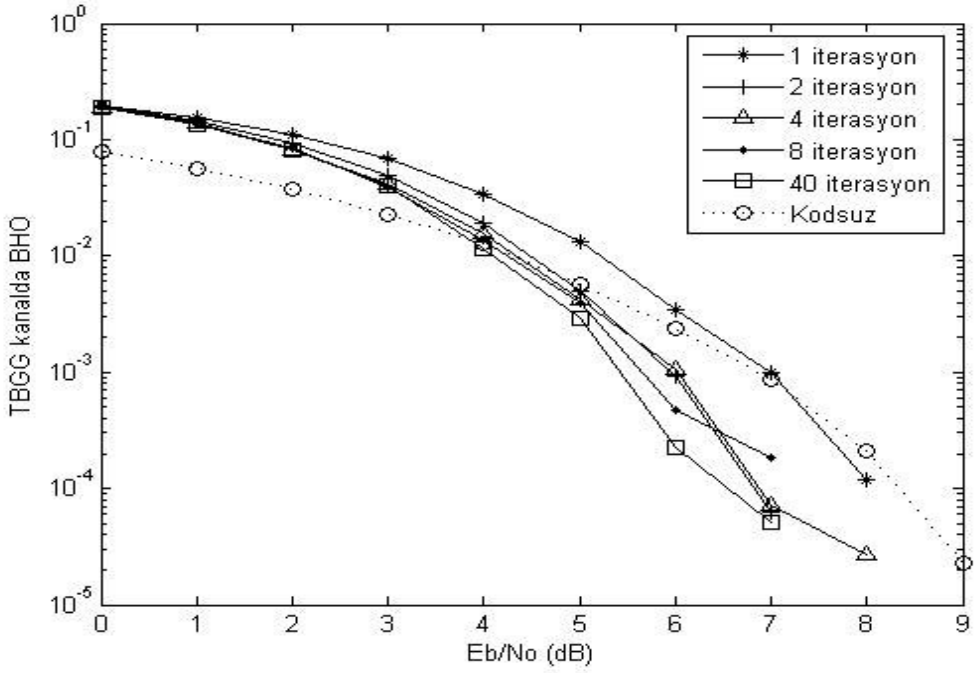


Şekil 4.2 C = (12, 7, 4) GÜ kodunun ve kodlanmamış bitlerin (kodsuz) TBGG kanalda toplam-ürün kod çözme algoritması yardımıyla elde edilen BHO.

Eğrilerden de görüldüğü gibi, iterasyonların başarıma etkisi 4 iterasyona kadar yüksek, 4'ten sonra çok az bir etkisi vardır. Hata başarım eğrileri Viterbi veya BCJR algoritmalarıyla elde edilen sonuçlara oldukça yakınsamaktadır fakat onlar kadar iyi değildir (Altay, 2005).



Şekil 4.3 C = (16, 11, 4) GÜ kodunun ve kodlanmamış bitlerin (kodsuz) TBGG kanalda toplam-ürün kod çözme algoritması yardımıyla elde edilen BHO



Şekil 4.4 C = (28, 22, 4) GÜ kodunun ve kodlanmamış bitlerin (kodsuz) TBGG kanalda toplam-ürün kod çözme algoritması yardımıyla elde edilen BHO.

4.5 GEOMETRİK İNŞA KODLARI

Önceki bölümde anlatılan GÜ kodlarının, genel bir ürün kodu üreteç matrisi üzerine GÜ kod inşasının belirlediği kurallara göre oluşturulan üreteç matrisi ile elde edildiğini gördük. Bu bölümde, genel bir ürün kodu üreteç matrisi kullanmadan tamamen sıfırdan üreteç matrisi inşa edilerek elde edilen yeni kodlar, Geometrik İnşa Kodları (Gİ), adı altında sunulacaktır. Üreteç matrisi inşa edilirken yerleştirilen eleman matrisleri basamaklı bir yapı özelliği gösterdiği için bu üreteç matrisleri ile elde edilen kodlar, "Basamak Blok Kodları" olarak da adlandırılabilir. Genel olarak Gİ kodlarının GÜ kodlarından tek farkı, GÜ kodu üreteç matrisi inşasında kullanılan genel bir ürün kodu üreteç matrisi G_y'nin, Gİ kodları üreteç matrisi inşasında kullanılmaması ve bunun yerine Gİ kodlarına özgü bir yapının ortaya konulmasıdır. Geri kalan inşa özellikleri ve elde edilen kodların özellikleri GÜ kodları ile aynıdır. Gİ kodları üreteç matrislerinin GÜ kodları üreteç matrislerine göre avantajı, genel ürün kodu üreteç matrisi G_y'nin yerine konulan matrisin aynen kendisine eklenen eleman matrisleri gibi basamaklı yapıda olması yani kaydırmalı kaydedicilerle her matris satırı belli sayıda kaydırılarak bir sonraki satırı elde etme olanağına sahip olmasıdır ki bu da kodlayıcı tasarımında önemli bir avantaj sağlar.

4.5.1 Geometrik İnşa Kodları Üreteç Matrisi Oluşturulması

Geometrik İnşa (Gİ) kodları yöntemiyle inşa edilen bir blok kodu **C = (n, k, d)** olarak verilsin. Burada *n* kod uzunluğu, *k* kod boyutu yani bilgi bitleri sayısı ve *d* kodun Hamming mesafesidir. **C** kodunu üreten üreteç matrisi **G**, üç veya daha fazla basit yapıdaki eleman kodları üreteç matrislerinin Gİ kod inşası yöntemiyle birleşimi sonucu oluşur. Gİ kodu üreteç matrisi GÜ kodu üreteç matrisinin tüm özelliklerin sahiptir.

Hamming mesafesi 4 olan bir Gİ kodunun üreteç matrisi, **G**, $\mathbf{C_1 = (n_1, k_1, d_1)} = (2,1,2)$ ve $\mathbf{C_2 = (n_2, k_2, d_2)} = (2,1,1)$ ölçülerindeki basit blok kodların $\mathbf{G_1 = [1\ 1]}$ and $\mathbf{G_2 = [1\ 0]}$ üreteç matrisleri kullanılarak (4.22) denklemine göre yerleştirilmesi ile elde edilir. Elde edilen Gİ kodlarının ölçüleri de GÜ kodları ile aynı olup (4.21) eşitliği ile aşağıda gösterilmektedir.

$$C = (n, k, d) = \left(n, n - \lceil \log_2 n + 1 \rceil, 4\right) . \tag{4.21}$$

üreteç matrisi **G** ise,

$$
G = \begin{pmatrix}
G_1G_1 & & & & & & & & & & & & \\
 & G_1G_1 & & & & & & & & & & & \\
 & & \bullet & & \bullet & & \bullet & & & & & & \\
 & & & & & & & & G_1G_1 & & & & \\
 & & & & & & & & & G_1G_1 & & & \\
G_2G_2G_2G_2 & & & & & & & & & & & & \\
 & G_2G_2G_2G_2 & & & & & & & & & & & \\
 & & & \bullet & & \bullet & & \bullet & & & & & \\
 & & & & & & & & & & G_2G_2G_2G_2 & & \\
G_2 & G_2 & G_2 & G_2 & & & & & & & & & \\
 & & G_2 & G_2 & G_2 & G_2 & & & & & & & \\
 & & & & & & \bullet & \bullet & \bullet & & & & \\
 & & & & & & & & & G_2 & G_2 & G_2 & G_2 \\
G_2 & & G_2 & & G_2 & & G_2 & & & & & & \\
 & & & & G_2 & & G_2 & & G_2 & & G_2 & & \\
 & & & & & & & & & & & \bullet \; \bullet \; \bullet & \\
 & & \bullet & \bullet & \bullet & & & & & & & &
\end{pmatrix}
\begin{matrix}
\\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\
\} E_1 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\
\} E_2 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\
\} E_3 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\
\} E_4 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\
\end{matrix}
\tag{4.22}
$$

eşitliği kullanılarak inşa edilir. Daha basit gösterim için matris şöyle de yazılabilir.

$$
G = \begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_t \end{bmatrix} \begin{matrix} \} E_1 \\ \} E_2 \\ \vdots \\ \} E_t \end{matrix} \tag{4.23}
$$

Burada $\mathbf{E_i}$ (i=1,2,…t) eleman matrislerinin geometrik yerleştirilmelerine göre ayrılan grup etiketleri olup bu gruplar $\mathbf{D_i}$ matrisleri ile temsil edilmiştir. Burada *t* grupların sayısını temsil etmektedir. (4.22) denklemiyle gösterilen Gİ kodlarının üreteç matrislerinin yerleştirilmesi aşağıda anlatılmıştır:

Gİ kodlarının üreteç matrisi, **G**, satırları yarleştirilme biçimlerine göre {E_1, E_2,…} olarak gruplandırılır. E_1 grubu için eleman matrislerinin satırlara yerleştirilmesinde 2 adet $\mathbf{G_1}$ eleman matrisi artarda hiç aralık bırakmadan ilk sütundan başlayarak yerleştirilir. İlk satıra yerleştirilen kolonlar 1 $\mathbf{G_1}$ matris ölçeği kadar sağa kaydırılarak 2. satıra yerleştirilir. Bu çevrimsel işleme E_1 grubunun tüm satırları için aynen devam edilir. Bu işlem, $\mathbf{G_1}$ matrisi en son sütuna ulaşıncaya kadar sürer. Gİ kodlarının GÜ kodlarından farklı olduğu inşa kısmı burasıdır. Çevrimsel özelliğinden dolayı kaydırmalı kaydedicilerle kodlamaya imkan sağladığından ve herhangi bir matris çarpma işlemi gerektirmediğinden GÜ kodunun inşasından daha avantajlıdır. Kalan kısımların inşası GÜ kodundakine benzer olmakla beraber

burada Gİ kodları için detaylarıyla açıklanacaktır.

E_2 grubu için eleman matrislerinin satırlara yerleştirilmesinde E_2 grubundaki tüm ekleme satırlarında her bir satıra 4 adet $\mathbf{G_2}$ matrisi birbiri ardına hiç boşluk bırakılmadan yerleştirilir. E_2 grubunun ilk satırında yerleştirme ilk kolondan başlar. E_2 grubunun ikinci satırında ise ardarda yerleştirilen 4 adet $\mathbf{G_2}$ matrisi 2 $\mathbf{G_2}$ matrisi ölçeğinde sağa kaydırılır. Böylece iki satırdaki 2'şer adet $\mathbf{G_2}$ matrisi üst üste gelmiş olur. Eğer 4 adet $\mathbf{G_2}$ matrisini yerleştirecek kolon kalmamışsa E_2 grubu için yerleştirme işlemi sona erer. E_2 grubu için yerleştirme işlemi tamamlandıktan sonra E_3 grubu için eğer yer varsa yerleştirme işlemine başlanır. Bu durumda E_3 grubu için yerleştirilecek 4 adet $\mathbf{G_2}$ matrisi arasında birer aralık bırakılır. Buradaki 'aralıktan' kasıt bir $\mathbf{G_2}$ ölçeği kadarlık yerdir. E_3 grubunun ilk satırında da yerleştirme ilk kolondan başlar. E_3 grubunun ikinci satırında ise artarda yerleştirilen 4 adet birer aralıklı $\mathbf{G_2}$ matrisi, 4 $\mathbf{G_2}$ matrisi ölçeğinde sağa kaydırılır ki böylece aynı grupta bulunan peş peşe iki satır arasındaki 2'şer adet $\mathbf{G_2}$ matrisleri üst üste gelmiş olur. Bu işlemde bir öncekine benzer biçimde 4 adet $\mathbf{G_2}$ matrisini yerleştirecek kolon kalmayana kadar devam eder. E_3 grubu için yerleştirme işlemi tamamlandıktan sonra E_4 grubu için eğer yer varsa yerleştirme işlemine başlanır. Bu durumda ise E_4 grubu için yerleştirilecek 4 adet $\mathbf{G_2}$ matrisi arasında üçer aralık bırakılır. Yerleştirme işlemi öncekiler gibi yapılmakla beraber sağa kaydırma 8 aralık yapılır. Bu işlemler aynı şekilde, eğer yerleştirmeye yer varsa, diğer gruplar içinde devam eder. Tüm yerleştirme işlemi 4 adet $\mathbf{G_2}$ matrisini yerleştirecek ekleme satırı kalmayınca sona erer. Genel bir kural olarak, E_1 grubu ayrı tutulmak üzere, sağa kaydırma işlemi E_2 grubu için 2, E_3 grubu için 4, E_4 grubu için 8 daha sonraki gruplar için 8, 16,... ve bu şekilde 2'nin kuvvetleri halinde devam eder. 4 adet $\mathbf{G_2}$ matrisi arasında bırakılan aralıklar ise E_2 grubu için 0, E_3 grubu için 1, daha sonraki gruplar için 3, 7, 15,... ve bu şekilde kaydırma miktarının yarısından 1 eksik olarak devam eder. Şu ana kadar açıklanan yerleştirme işlemi $\mathbf{C_1}$ ve $\mathbf{C_2}$ kodlarının üreteç matrisleri $\mathbf{G_1}$ ve $\mathbf{G_2}$ kullanılarak yapılmaktadır. Fakat özellikle büyük Hamming mesafeli kodların inşasında karşılaşılacağı gibi bazen $\mathbf{G_2}$ matrisine ek olarak bir başka benzer ölçülerde eleman matrisi, mesela $\mathbf{G_3}$, eklenmesi gerekebilir ki, böyle bir durumda yerleştirme işlemi aynen $\mathbf{G_2}$ matrisine yapıldığı gibi yapılır ve yerleştirmeye $\mathbf{G_2}$ matrisinden sonra devam edilir.

Şimdiye kadar açıklanan kurallar uygun eleman matrisleri seçildiği varsayımı altında eleman matrislerinin nasıl yerleştirildiği ile ilgiliydi. Eleman matrislerinin nasıl seçileceği ile ilgili genel kurallar bulunmamakla beraber bunların nasıl seçileceğini en başta istenilen kodun uzunluğu belirler. İstenilen uzunluklu koda göre $\mathbf{C_1}$ ve $\mathbf{C_2}$ kodlarının üreteç matrisleri birleştirilerek genel ürün matrisi yapısı elde edilir. $\mathbf{G_z}$ matrisi seçilirken dikkat edilmesi

gereken husus ise 4 adet $\mathbf{G_2}$ matrisinde bulunan mantıksal 1'lerin toplamı, kodun istenilen minimum Hamming mesafesine eşit olmasıdır. Bu konuda daha sonraki kısımlarda açıklayıcı örnekler verilecektir. Gİ kodlarının bir kod ailesi olan minimum Hamming mesafeli kodların Hamming mesafelerinin 4 olduğunu ispatlamak için Teorem 4.2 önerilip ispatlanacaktır. Fakat daha önce ispat yapabilmek için geliştirilen yardımcı teoremler (Lemma) ve bazı temel teoremler sunulacaktır.

Öncelikle teoremlerin anlaşılması için şu özdeşliği hatırlatmakta fayda var: Eğer **u** ve **v** ikili vektörler ise, **u•v** işlemi **u** ve **v'**nin her ikisinde de aynı yerde bulunan l'lerin sayısını gösterir. Bundan yola çıkarak $wt(u + v) = wt(u) + wt(v) – 2(u•v)$ eşitliği çok bilinen ve çok faydalı bir eşitlik olup burada $wt(c)$ gösterimi *c*'nin Hamming ağırlığını gösterir.

Teorem 4.2: Eğer, bir ikili *(n,k)* kodu **C** için, **G** üreteç matrisinin satırları çift ağırlık değerine sahipse ve satırları birbirine dikse, o zaman **C** "kendi-diktir" (Pless, V., 1989).

Teorem 4.3: Eğer, bir ikili *(n,k)* kodu **C** için **G** üreteç matrisinin satırları 4 ile bölünebilir bir ağırlığa sahipse ve birbirine dikse, o zaman **C** kendi-diktir ve **C** kodundaki tüm ağırlıklar 4 ile bölünebilir (Pless, V., 1989).

Yardımcı Teorem 4.1: Denklem (4.23) ile oluşturulan bir grup üreteç matrisi, D_i, ile üretilen bir kod kelimesinin Hamming ağırlığı en az 4 tür.

İspat: Denklem (4.23) ile oluşturulan bir grup üreteç matrisinin, D_i, satırlarının Hamming ağırlığı 4'tür yani çifttir. Ayrıca bu satırlar birbirlerine diktir. Her bir grup içinde diklik özelliği sağlanmaktadır. Bu durum Teorem 4.2 de aranan şartı sağlar, dolayısıyla bir D_i grup üreteç matrisi ile üretilen bir C_i kodu kendi-diktir. D_i grup üreteç matrislerindeki satırların ağırlıkları 4 olduğundan, yani 4 ile bölünebildiğinden ve C_i kendi-dik olduğundan, Teorem 4.3'ten yola çıkılırsa, o zaman D_i ile üretilen bütün C_i kodlarının ağırlıkları 4 ile bölünebilir. D_i içindeki her bir satırın ağırlığı 4 olduğundan, yani 4 ile bölünebilir olduğundan, C_i kodunun minimum Hamming ağırlığı yani minimum Hamming mesafesi 4'tür. Sonuç olarak, Yardımcı Teorem 4.1, (4.22) veya (4.23) denklemlerindeki her bir grup minimum Hamming mesafesi 4 olan kod kelimeleri üretir.

Yardımcı Teorem 4.2: Eğer **u** ve **v** (4.22) denklemiyle inşa edilen üreteç matrisinin, **G**, ayrık ikili satır vektörleri ise, ozaman **u•v** işlemi sonucu en fazla 2'dir.

İspat: (4.22) denklemiyle oluşturulan **G** üreteç matrisinden de gözlenebileceği gibi matrisin satırlarının yerleştirilme kuralı doğal olarak kendiliğinden bunu sağlar. Çünkü **G** içindeki üst

üste gelen eleman matrislerinin sayısı en fazla 2'dir.

Teorem 4.4: (4.22) denklemiyle oluşturulan bir **G** üreteç matrisi ile üretilen bir **C** kodunun minimum Hamming mesafesi 4'tür.

İspat: Eğer **u** ve **v**, (4.22) denklemiyle inşa edilen üreteç matrisi **G**'nin ayrık ikili satır vektörleri ise $wt(u + v) = wt(u) + wt(v) – 2(u \bullet v)$ özdeşliğinden yararlanılarak ispat yapılabilir. Yardımcı Teorem 4.1 kullanılırsa, $wt(u) \geq 4$ ve $wt(v) \geq 4$ olduğu görülür. Yardımcı Teorem 4.2'den de $u \bullet v \leq 2$ bulunur. O halde en kötü sonuca göre hesap yapıldığında minimum Hamming mesafesi $wt(u + v) = 4 + 4 – 2*2 = 4$ bulunur. Dolayısıyla C kodunun minimum Hamming mesafesi 4'tür.

Literatürde birçok uygulama için kod ölçüsünü kontrol edebilme özelliğinin önemli olduğu görülmektedir. Mesela (Isaka, M. ve Fossorier, M., 2005) kaynağında 256 uzunluklu bir kodu inşa edebilmek için bir adet (128, 120) ve iki adet (64, 57) "uzatılmış Hamming kodu" kullanılmış ve sonuçta kodlama oranı $R = (120 + 57*2)/(128 + 64*2) = 0.914$ olan bir kod elde edilmiştir. Aynı uzunluktaki kod, (4.22) denklemiyle üreteç matrisi oluşturulan Gİ kodu olarak (256, 247,4) ölçülerinde inşa edilirse bu kodun kodlama oranı $R = 247/256 = 0.964$ olur. Buda gösteriyor ki Gİ kodu 0.05 daha iyi kodlama oranı sağlar. Bu sadece Gİ kodlarının kodlama oranındaki avantajını göstermektedir. Ayrıca, üreteç matrisinin yarı çevrimsel bir yapıda olmasından dolayı pratik kodlama uygulamaları için de çok avantajlı olduğu söylenebilir.

Şu ana kadar minimum Hamming mesafesi 4 olan Gİ kodlarının inşası gösterildi. Daha büyük Hamming mesafeli Gİ kodları da (4.22) denklemi ile gösterilen üreteç matrisi **G** kullanılarak oluşturulabilir. Bu durumda sadece eleman matrislerini değiştirmek yeterlidir. Örneğin minimum Hamming mesafesi 8 olan Gİ kodlarını inşa edebilmek için eleman matrisleri $\mathbf{G_1 = [1\ 1\ 1\ 1]}$ ve $\mathbf{G_2 = [1\ 1\ 0\ 0]}$ seçilebilir. Fakat daha iyi kodlama oranı elde edebilmek için başka eleman matrislerini de eklemek mümkünse eklenmelidir. Bilgisayar programlarıyla yapılan araştırmalar sonucu bir başka eleman matrisinin, $\mathbf{G_3 = [0\ 1\ 1\ 0]}$, aynen $\mathbf{G_2}$ matrisi gibi **G** içine $\mathbf{G_2}$ matrisleri yerleştirildikten sonra yerleştirilebileceği görülmüştür. Bu üç eleman matrisi (4.22) denkleminde açıklandığı şekilde yerleştirildiğinde minimum Hamming mesafesi 8 olan Gİ kodları inşa edebilirler. Söz konusu kodlardan bazıları bir sonraki kısımda sunulacaktır.

Minimum Hamming mesafesi 16 olan Gİ kodları inşası için $\mathbf{G_1 = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]}$, $\mathbf{G_2 = [1\ 1\ 1\ 1\ 0\ 0\ 0\ 0]}$, $\mathbf{G_3 = [0\ 0\ 1\ 1\ 1\ 1\ 0\ 0]}$ and $\mathbf{G_4 = [1\ 0\ 1\ 0\ 1\ 0\ 1\ 0]}$ eleman matrisleri

seçilir. Burada $\mathbf{G_2}$, $\mathbf{G_3}$ ve $\mathbf{G_4}$ aynı biçim ve sırayla yerleştirilir. Daha büyük Hamming mesafeli kodların inşası için aşağıdaki eleman matrisleri kullanılır.

d=32

G_1=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]

G_2=[1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0]

G_3=[1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0] (4.24)

G_4=[1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0]

G_5=[1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0]

d=64

G_1=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]

G_2=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

G_3=[1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0]

G_4=[1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0] (4.25)

G_5=[1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0]

G_6=[1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0]

d=128

G_1=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]

G_2=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

G_3=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

G_4=[1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 (4.26)

1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0]

G_5=[1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0

1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0]

G_6=[1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0

1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0]

G_7=[1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0

1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0]

d=256

G_1=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]

G_2=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

G_3=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

G_4=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

G_5=[1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1
1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0]

G_6=[1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1
1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0]

G_7=[1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0]

G_8=[1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0]

(4.27)

4.5.2 Geometrik İnşa Kodlarının Bazı Üreteç Matrisleri

Gİ kodlarından bazıları aşağıdaki örneklerde sunulmuştur. Bu kodlar, bilgisayar yardımıyla hazırladığımız yazılımlardan elde edilebilmektedir.

Örnek 4.12: C = (12, 7, 4) olan bir Gİ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix} 111100000000 \\ 001111000000 \\ 000011110000 \\ 000000111100 \\ 000000001111 \\ 101010100000 \\ 000010101010 \end{bmatrix} \tag{4.28}$$

Örnek 4.13: C = (14, 9, 4) olan bir Gİ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix} 11110000000000 \\ 00111100000000 \\ 00001111000000 \\ 00000011110000 \\ 00000000111100 \\ 00000000001111 \\ 10101010000000 \\ 00001010101000 \\ 10001000100010 \end{bmatrix} \tag{4.29}$$

Örnek 4.14: C = (16, 11, 4) olan bir Gİ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix} 1111000000000000 \\ 0011110000000000 \\ 0000111100000000 \\ 0000001111000000 \\ 0000000011110000 \\ 0000000000111100 \\ 0000000000001111 \\ 1010101000000000 \\ 0000101010100000 \\ 0000000010101010 \\ 1000100010001000 \end{bmatrix} \tag{4.30}$$

Örnek 4.15: C = (18, 12, 4) olan bir Gİ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix} 111100000000000000 \\ 001111000000000000 \\ 000011110000000000 \\ 000000111100000000 \\ 000000001111000000 \\ 000000000011110000 \\ 000000000000111100 \\ 000000000000001111 \\ 101010100000000000 \\ 000010101010000000 \\ 000000001010101000 \\ 100010001000100000 \end{bmatrix} \tag{4.31}$$

Örnek 4.16: C = (20, 14, 4) olan bir Gİ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix} 11110000000000000000 \\ 00111100000000000000 \\ 00001111000000000000 \\ 00000011110000000000 \\ 00000000111100000000 \\ 00000000001111000000 \\ 00000000000011110000 \\ 00000000000000111100 \\ 00000000000000001111 \\ 10101010000000000000 \\ 00001010101000000000 \\ 00000000101010100000 \\ 00000000000010101010 \\ 10001000100010000000 \end{bmatrix} \tag{4.32}$$

Örnek 4.17: C = (22, 16, 4) olan bir Gİ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix}
1111000000000000000000 \\
0011110000000000000000 \\
0000111100000000000000 \\
0000001111000000000000 \\
0000000111100000000000 \\
0000000001111000000000 \\
0000000000111100000000 \\
0000000000001111000000 \\
0000000000000011110000 \\
0000000000000000111100 \\
0000000000000000001111 \\
1010101000000000000000 \\
0000101010100000000000 \\
0000000010101010000000 \\
0000000000001010101000 \\
1000100010001000000000 \\
0000000010001000100010
\end{bmatrix} \tag{4.33}$$

Örnek 4.18: C = (24, 18, 4) olan bir Gİ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix}
111100000000000000000000 \\
001111000000000000000000 \\
000011110000000000000000 \\
000000111100000000000000 \\
000000001111000000000000 \\
000000000011110000000000 \\
000000000000111100000000 \\
000000000000001111000000 \\
000000000000000011110000 \\
000000000000000000111100 \\
000000000000000000001111 \\
101010100000000000000000 \\
000010101010000000000000 \\
000000001010101000000000 \\
000000000000101010100000 \\
000000000000000010101010 \\
100010001000100000000000 \\
000000001000100010001000
\end{bmatrix} \tag{4.34}$$

Örnek 4.19: C = (26, 20, 4) olan bir Gİ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix}
11110000000000000000000000 \\
00111100000000000000000000 \\
00001111000000000000000000 \\
00000011110000000000000000 \\
00000000111100000000000000 \\
00000000001111000000000000 \\
00000000000011110000000000 \\
00000000000000111100000000 \\
00000000000000001111000000 \\
00000000000000000011110000 \\
00000000000000000000111100 \\
00000000000000000000001111 \\
10101010000000000000000000 \\
00001010101000000000000000 \\
00000000101010100000000000 \\
00000000000010101010000000 \\
00000000000000001010101000 \\
10001000100010000000000000 \\
00000000100010001000100000 \\
10000000100000001000000010
\end{bmatrix} \tag{4.35}$$

Örnek 4.20: C = (28, 22, 4) olan bir Gİ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix}
1111000000000000000000000000 \\
0011110000000000000000000000 \\
0000111100000000000000000000 \\
0000001111000000000000000000 \\
0000000011110000000000000000 \\
0000000000111100000000000000 \\
0000000000001111000000000000 \\
0000000000000011110000000000 \\
0000000000000000111100000000 \\
0000000000000000001111000000 \\
0000000000000000000011110000 \\
0000000000000000000000111100 \\
0000000000000000000000001111 \\
1010101000000000000000000000 \\
0000101010100000000000000000 \\
0000000010101010000000000000 \\
0000000000001010101000000000 \\
0000000000000000101010100000 \\
0000000000000000000010101010 \\
1000100010001000000000000000 \\
0000000010001000100010000000 \\
1000000010000000100000001000
\end{bmatrix} \tag{4.36}$$

Örnek 4.21: C = (30, 24, 4) olan bir Gİ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix}
111100000000000000000000000000 \\
001111000000000000000000000000 \\
000011110000000000000000000000 \\
000000111100000000000000000000 \\
000000001111000000000000000000 \\
000000000011110000000000000000 \\
000000000000111100000000000000 \\
000000000000001111000000000000 \\
000000000000000011110000000000 \\
000000000000000000111100000000 \\
000000000000000000001111000000 \\
000000000000000000000011110000 \\
000000000000000000000000111100 \\
000000000000000000000000001111 \\
101010100000000000000000000000 \\
000010101010000000000000000000 \\
000000001010101000000000000000 \\
000000000000101010100000000000 \\
000000000000000010101010000000 \\
000000000000000000001010101000 \\
100010001000100000000000000000 \\
000000001000100010001000000000 \\
000000000000000010001000100010 \\
100000001000000010000000100000
\end{bmatrix} \tag{4.37}$$

Örnek 4.22: C = (32, 26, 4) olan bir Gİ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix}
11110000000000000000000000000000 \\
00111100000000000000000000000000 \\
00001111000000000000000000000000 \\
00000011110000000000000000000000 \\
00000000111100000000000000000000 \\
00000000001111000000000000000000 \\
00000000000011110000000000000000 \\
00000000000000111100000000000000 \\
00000000000000001111000000000000 \\
00000000000000000011110000000000 \\
00000000000000000000111100000000 \\
00000000000000000000001111000000 \\
00000000000000000000000011110000 \\
00000000000000000000000000111100 \\
00000000000000000000000000001111 \\
10101010000000000000000000000000 \\
00001010101000000000000000000000 \\
00000000101010100000000000000000 \\
00000000000010101010000000000000 \\
00000000000000001010101000000000 \\
00000000000000000010101010000000 \\
00000000000000000000101010100000 \\
00000000000000000000001010101010 \\
10001000100010000000000000000000 \\
00000000100010001000100000000000 \\
00000000000000001000100010001000 \\
10000000100000001000000010000000
\end{bmatrix} \tag{4.38}$$

Örnek 4.23: C = (34, 27, 4) olan bir Gİ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix}
1111000000000000000000000000000000 \\
0011110000000000000000000000000000 \\
0000111100000000000000000000000000 \\
0000001111000000000000000000000000 \\
0000000011110000000000000000000000 \\
0000000000111100000000000000000000 \\
0000000000001111000000000000000000 \\
0000000000000011110000000000000000 \\
0000000000000000111100000000000000 \\
0000000000000000001111000000000000 \\
0000000000000000000111100000000000 \\
0000000000000000000011110000000000 \\
0000000000000000000001111000000000 \\
0000000000000000000000111100000000 \\
0000000000000000000000011110000000 \\
0000000000000000000000001111000000 \\
0000000000000000000000000111100000 \\
0000000000000000000000000011110000 \\
0000000000000000000000000001111000 \\
0000000000000000000000000000111100 \\
0000000000000000000000000000011110 \\
0000000000000000000000000000001111 \\
1010101000000000000000000000000000 \\
0000101010100000000000000000000000 \\
0000000010101010000000000000000000 \\
0000000000001010101000000000000000 \\
0000000000000010101010000000000000 \\
0000000000000000101010100000000000 \\
0000000000000000001010101000000000 \\
0000000000000000000010101010000000 \\
0000000000000000000000101010100000 \\
1000100010001000000000000000000000 \\
0000000010001000100010000000000000 \\
0000000000000000100010001000100000 \\
1000000010000000100000001000000000
\end{bmatrix} \qquad (4.39)$$

Örnek 4.24: C = (40, 33, 4) olan bir Gİ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix}
1111000000000000000000000000000000000000\\
0011110000000000000000000000000000000000\\
0000111100000000000000000000000000000000\\
0000001111000000000000000000000000000000\\
0000000011110000000000000000000000000000\\
0000000000111100000000000000000000000000\\
0000000000001111000000000000000000000000\\
0000000000000011110000000000000000000000\\
0000000000000000111100000000000000000000\\
0000000000000000001111000000000000000000\\
0000000000000000000011110000000000000000\\
0000000000000000000000111100000000000000\\
0000000000000000000000001111000000000000\\
0000000000000000000000000011110000000000\\
0000000000000000000000000000111100000000\\
0000000000000000000000000000001111000000\\
0000000000000000000000000000000011110000\\
0000000000000000000000000000000000111100\\
0000000000000000000000000000000000001111\\
1010101000000000000000000000000000000000\\
0000101010100000000000000000000000000000\\
0000000010101010000000000000000000000000\\
0000000000001010101000000000000000000000\\
0000000000000010101010000000000000000000\\
0000000000000000101010100000000000000000\\
0000000000000000001010101000000000000000\\
0000000000000000000010101010000000000000\\
0000000000000000000000101010100000000000\\
0000000000000000000000001010101000000000\\
0000000000000000000000000010101010000000\\
1000100010001000000000000000000000000000\\
0000000010001000100010000000000000000000\\
0000000000000000100010001000100000000000\\
0000000000000000000000001000100010001000\\
1000000010000000100000001000000000000000
\end{bmatrix} \tag{4.40}$$

Örnek 4.25: C = (16, 5, 8) olan bir Gİ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix} 1111111100000000 \\ 0000111111110000 \\ 0000000011111111 \\ 1100110011001100 \\ 0110011001100110 \end{bmatrix} \tag{4.41}$$

Örnek 4.26: C = (24, 9, 8) olan bir Gİ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix} 111111110000000000000000 \\ 000011111111000000000000 \\ 000000001111111100000000 \\ 000000000000111111110000 \\ 000000000000000011111111 \\ 110011001100110000000000 \\ 000000001100110011001100 \\ 011001100110011000000000 \\ 000000000110011001100110 \end{bmatrix} \tag{4.42}$$

Örnek 4.27: C = (32, 15, 8) olan bir Gİ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix} 11111111000000000000000000000000 \\ 00001111111100000000000000000000 \\ 00000000111111110000000000000000 \\ 00000000000011111111000000000000 \\ 00000000000000001111111100000000 \\ 00000000000000000000111111110000 \\ 00000000000000000000000011111111 \\ 11001100110011000000000000000000 \\ 00000000110011001100110000000000 \\ 00000000000000001100110011001100 \\ 11000000110000001100000011000000 \\ 01100110011001100000000000000000 \\ 00000000011001100110011000000000 \\ 00000000000000000110011001100110 \\ 01100000011000000110000001100000 \end{bmatrix} \tag{4.43}$$

Örnek 4.28: C = (48, 25, 8) olan bir Gİ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix}
11111111000000000000000000000000000000000000000000 \\
00001111111100000000000000000000000000000000000000 \\
00000000111111110000000000000000000000000000000000 \\
00000000000011111111000000000000000000000000000000 \\
00000000000000001111111100000000000000000000000000 \\
00000000000000000000111111110000000000000000000000 \\
00000000000000000000001111111100000000000000000000 \\
00000000000000000000000011111111000000000000000000 \\
00000000000000000000000000111111110000000000000000 \\
00000000000000000000000000001111111100000000000000 \\
00000000000000000000000000000011111111000000000000 \\
00000000000000000000000000000000111111110000000000 \\
00000000000000000000000000000000001111111100000000 \\
00000000000000000000000000000000000011111111000000 \\
11001100110011000000000000000000000000000000000000 \\
00000000110011001100110000000000000000000000000000 \\
00000000000000001100110011001100000000000000000000 \\
00000000000000000000000011001100110011000000000000 \\
00000000000000000000000000001100110011001100000000 \\
11000000110000001100000011000000000000000000000000 \\
00000000000000001100000011000000110000001100000000 \\
01100110011001100000000000000000000000000000000000 \\
00000000110011001100110000000000000000000000000000 \\
00000000000000001100110011001100000000000000000000 \\
00000000000000000000000011001100110011000000000000 \\
00000000000000000000000000001100110011001100110000 \\
01100000011000000110000001100000000000000000000000 \\
00000000000000001100000011000000110000001100000000
\end{bmatrix} \tag{4.44}$$

Örnek 4.29: C = (64, 19, 16) olan bir Gİ kodu üreteç matrisi şu şekildedir.

$$G = \begin{bmatrix}
1111111111111110000000000000000000000000000000000000000000000000 \\
0000000011111111111111100000000000000000000000000000000000000000 \\
0000000000000001111111111111110000000000000000000000000000000000 \\
0000000000000000000001111111111111110000000000000000000000000000 \\
0000000000000000000000000111111111111110000000000000000000000000 \\
0000000000000000000000000000011111111111110000000000000000000000 \\
0000000000000000000000000000000111111111111100000000000000000000 \\
0000000000000000000000000000000000000000001111111111111111111111 \\
1111000011110000111100001111000000000000000000000000000000000000 \\
0000000000000000111100001111000011110000111100000000000000000000 \\
0000000000000000000000000000000011110000111100001111000011110000 \\
1111000000000000111100000000000011110000000000001111000000000000 \\
0011110000111100001111000011110000000000000000000000000000000000 \\
0000000000000000001111000011110000111100001111000000000000000000 \\
0000000000000000000000000000000011110000111100001111000011110000 \\
0011110000000000001111000000000000111100000000000011110000000000 \\
1010101010101010101010101010101000000000000000000000000000000000 \\
0000000000000000101010101010101010101010101010100000000000000000 \\
0000000000000000000000000000000010101010101010101010101010101010 \\
1010101000000000101010100000000010101010000000001010101000000000
\end{bmatrix} \quad (4.45)$$

Buraya kadar bazı temel Gİ kodları için üreteç matrisleri örnek olarak sunulmuştur. Bu matrisler, yarı-çevrimsel yapısı nedeniyle özellikle kodlayıcı tasarımı uygulamalarında avantaj sağlar.

4.5.3 Algoritma Analizi

Analiz kavramı, bir algoritmanın hem teorik hem de pratik açıdan direnç noktalarını tayin etmeyi amaçlar. Algoritmaları analiz etmenin en temel sebepleri çeşitli uygulamalar için algoritmanın uygunluğunu ifade edebilmek için algoritmanın karakteristiğini bilmek veya belirli bir uygulamada sözkonusu bir algoritmayı diğerleriyle karşılaştırmaktır.

Belirli bir algoritmanın işletimi için bilgisayar ortamında ne kadar süreye ve belleğe gereksinim duyulduğunu bilmek isteriz. Gerçekleştirmeden bağımsız olan algoritma analizi, bilgisayar ortamında ihtiyaç duyulacak kaynakları daha duyarlı bir biçimde tahmin edilmesine fırsat veren temel ayırıcı nitelikleri belirlemede önemli rol oynar.

Algoritma analizinin algoritmanın gerçekleştiriminden bağımsız olarak yapılması kolay değildir. Derleyici özellikleri, bilgisayar mimarisi, gerçekleştirim kalitesi vb. konular elbette algoritmanın performansını etkiler. Fakat analiz süreci, söz konusu etkenleri ve programlama ortamının olası tüm avantajlarını tanımamıza da fırsat verir.

4.5.3.1 Algoritma Karmaşıklığı

Algoritmadan elde edilecek ve algoritmanın gerçekleştirildiği ortamlarda da sabit bir bir niteliğe ihtiyaç vardır. Bu nitelik algoritma karmaşıklığı olarak karşımıza çıkmaktadır.

Bir algoritmanın performansı iç ve dış faktörlere bağlıdır. İç faktörler, algoritmayı çalıştırmak için gereken zaman ve gereken bellek olarak sıralanabilir. Dış faktörler ise giriş verisinin büyüklüğü, kullanılan bilgisayarın hızı ve bilgisayar işlemcisinin kalitesidir. Karmaşıklık terimi iç faktörlere ve daha çok zamana bağlıdır.

4.5.3.2 Büyüme Hızı ve Büyük- O Gösterimi

Büyüme hızı, bir algoritmanın performansını yansıtan en iyi göstergedir. *Büyük-O* gösterimi, büyüme hızını gösterir. Bir algoritmanın performansını en iyi tanımlanan matematiksel bir formüldür ve algoritmanın iç detaylarına bakılarak elde edilir. Ayrıca, giriş verisinin büyüklüğünü gösteren bir *N* prarametresine dayanan bir fonksiyondur.

Örneğin *n* değerine bağlı olarak performansı (sabit *a, b, c* değerleri için) $an^2 + bn + c$ olan bir algoritmanın performansı $O(N^2)$'dir.

N değeri arttıkça N^2 terimi baskın olacağı için *büyük-O* gösteriminde sadece baskın olan terimler kullanılır.

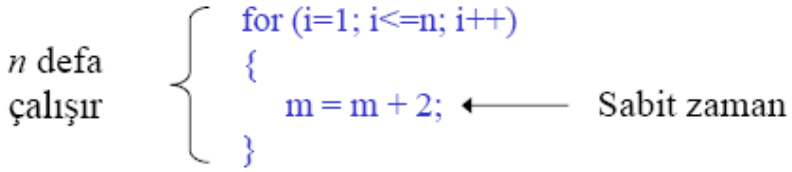
4.5.3.3 Büyük- O Hesaplanması

Bir program kodunun zaman karmaşıklığını hesaplamak için beş adet kural bulunmaktadır.

1) Döngüler
2) İç içe döngüler
3) Ardışık deyimler
4) If-then-else deyimleri
5) Logaritmik karmaşıklık

1) **Döngüler:** Bir döngünün çalışma zamanı, en çok döngü içindeki deyimlerin çalışma zamanının iterasyon saysıyla çarpımı kadardır.

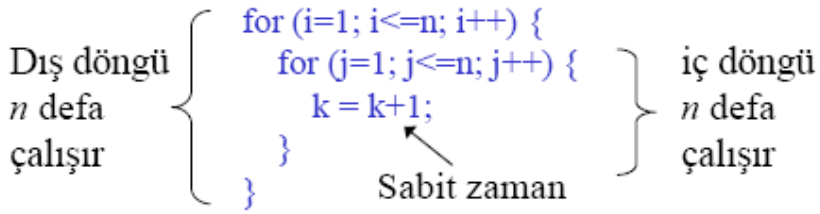
Örnek 4.30:



Toplam zaman = sabit $c * n = cn = O(N)$

2) **İç içe döngüler:** İçteki analiz yapılır. Toplam zaman bütün döngülerin çalışma sayılarının çarpımına eşittir.

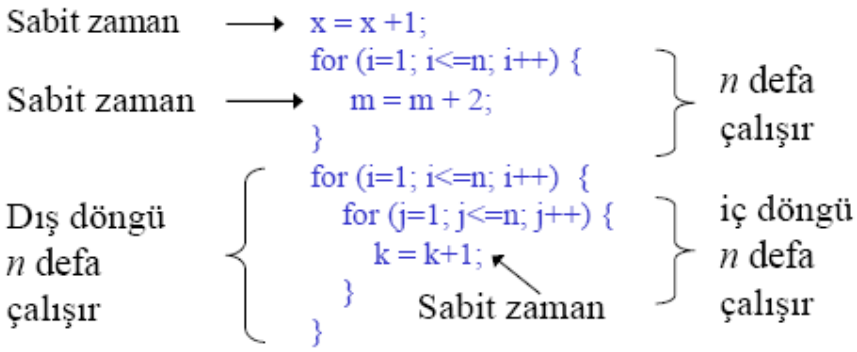
Örnek 4.31:



Toplam zaman = sabit $c * n * n = cn^2 = O(N^2)$

3) **Ardışık deyimler:** Her deyimin zamanı birbirine eklenir.

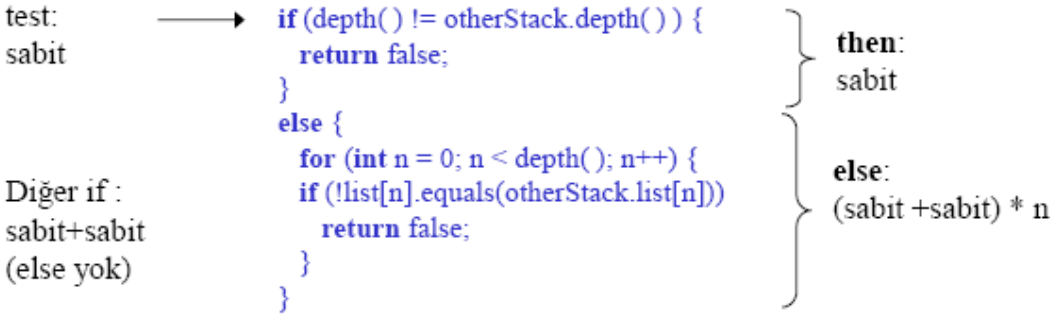
Örnek 4.32:



Toplam zaman = $c_0 + c_1 n + c_2 n^2 = O(N^2)$

4) **If-then-else deyimleri:** En kötü çalışma zamanı. Test zamanına then veya else kısmındaki çalışma zamanının hangisi büyükse o kısım eklenir.

Örnek 4.33:



Toplam zaman = $c_0 + c_1 + (c_2 + c_3) * n = O(N)$

5) **Logaritmik Karmaşıklık:** Problemin büyüklüğünü belli oranda (genelde ½) azaltmak için sabit bir zaman harcanıyorsa bu algoritma *O (log N)*'dir.

Örnek algoritma: N sayfalı bir sözlükten bir sözcük arama.

Önce sözlüğün orta kısmına bakılır. Daha sonra sözcük ortaya göre sağda mı solda mı kaldığı bulunur. Bu işlem, sağ veya solda sözcük bulunana kadar tekrarlanır.

4.5.3.4 Tezde Kullanılan Algoritmanın Karmaşıklığı

Tezde farklı Hamming mesafeleri için üç alt program bir de ana program olmak üzere toplam dört adet program kodu hazırlanmıştır. Bu kodların ve tüm algoritmanın karmaşıklık hesabı aşağıdaki şekilde yapılmıştır.

Ana program: $O(n^2) = c_0 + c_1 + c_2 * n + c_3 + (c_4 + c_5 + c_6 + c_7) * n^2$

Alt program1: $O(n^2) = c_0 + (c_1 + c_2 + c_3 + c_7 + c_8) * n + (c_4 + c_5 + c_6) * n^2$

Alt program2: $O(n) = c_0 + (c_1 + c_2 + c_3) * n$

Alt program3: $O(n) = (c_0 + c_1) * n + c_2 * n$

Tüm algoritmanın karmaşıklığı ise $O(n^2)$'dir.

4.5.4 Toplam-Ürün Algoritması Kod Çözümüyle Geometrik İnşa Kodlarının Hata Başarımı

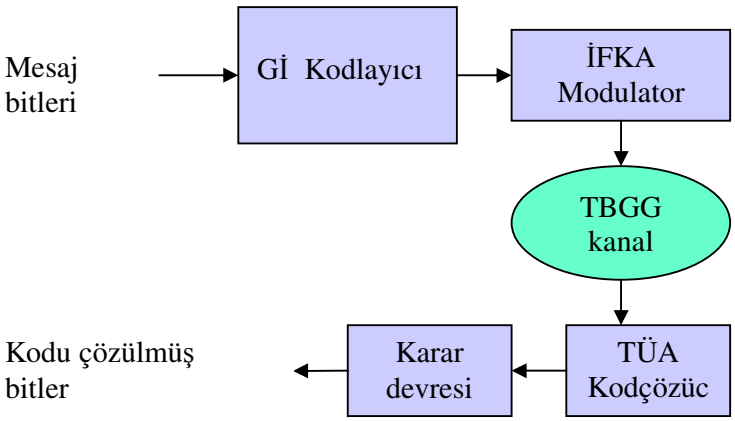
Bu bölümde Gİ kodlarının Toplam Ürün Algoritması ile kod çözülmesi durumunda hata başarımları bilgisayar benzetimleri yardımıyla elde edilmiştir. Buradaki bilgisayar benzetimi ile oluşturulan modelde İkili Faz Kaydırmalı anahtarlama (İFKA) modülasyonu kullanılmış ve işaretler Toplamsal Beyaz Gauss Gürültülü (TBGG) kanaldan iletilmiştir. Öncelikle, Gİ kodlarının parite kontrol matrisini elde etmek amacıyla Gİ'nin üreteç matrisi Gauss-Jordan Eleme yöntemiyle (Noble, B., 1969) sistematik forma getirilmiştir.

Bilgisayar benzetimleri, sıfır ortalamalı ve varyansı $\sigma^2 = N_0/2$ olan, TBGG kanal için gerçeklenmiştir ki burada $N_0/2$ tek taraflı Güç Spektrum Yoğunluğudur (GSY). Bazı Gİ kodları için bit hata başarımları E_b/N_0 (desibel biriminde) ve Bit Hata Oranı (BHO) ve Kelime Hata Oranı (KHO) parametrelerine göre elde edilmiş olup burada E_b bir bilgi biti başına düşen enerji miktarıdır.

Şekil 4.5'de görülen modelimizde, Gİ kodları İkili Faz Kaydırmalı anahtarlama (İFKA) modülasyonu kullanılarak mantıksal 1 biti +1 ve mantıksal 0 biti -1 değerlerine atanır. Alıcı kısımda, ideal kanal durum bilgisinin bulunduğu varsayılmış ve yumuşak kararlı Toplam Ürün Algoritması kod çözme uygulanmıştır. İncelenen tüm Gİ kodları için mükemmel bit ve blok senkronizasyonu (eşzamanlama) olduğu varsayılmıştır. Bu haberleşme modelinde bazı Gİ kodları için elde edilen bit hata oranları için eğriler bilgisayar kodları yardımıyla

çizdirilerek aşağıda gösterilmiştir.

Ayrıca karşılaştırma için kullanabilmek amacıyla bitlerden oluşan bloklar İFKA modülasyonu gerçeklendikten sonra kodlayıcı ve kod çözücü kullanılmadan BHO değerleri elde edilmiş ve tüm eğrilerde kodlanmamış İFKA modülasyonunu temsil etmek üzere çizdirilmiş, çok yakın olan durumlar için Shannon sınırı da şekillerde belirtilmiştir.

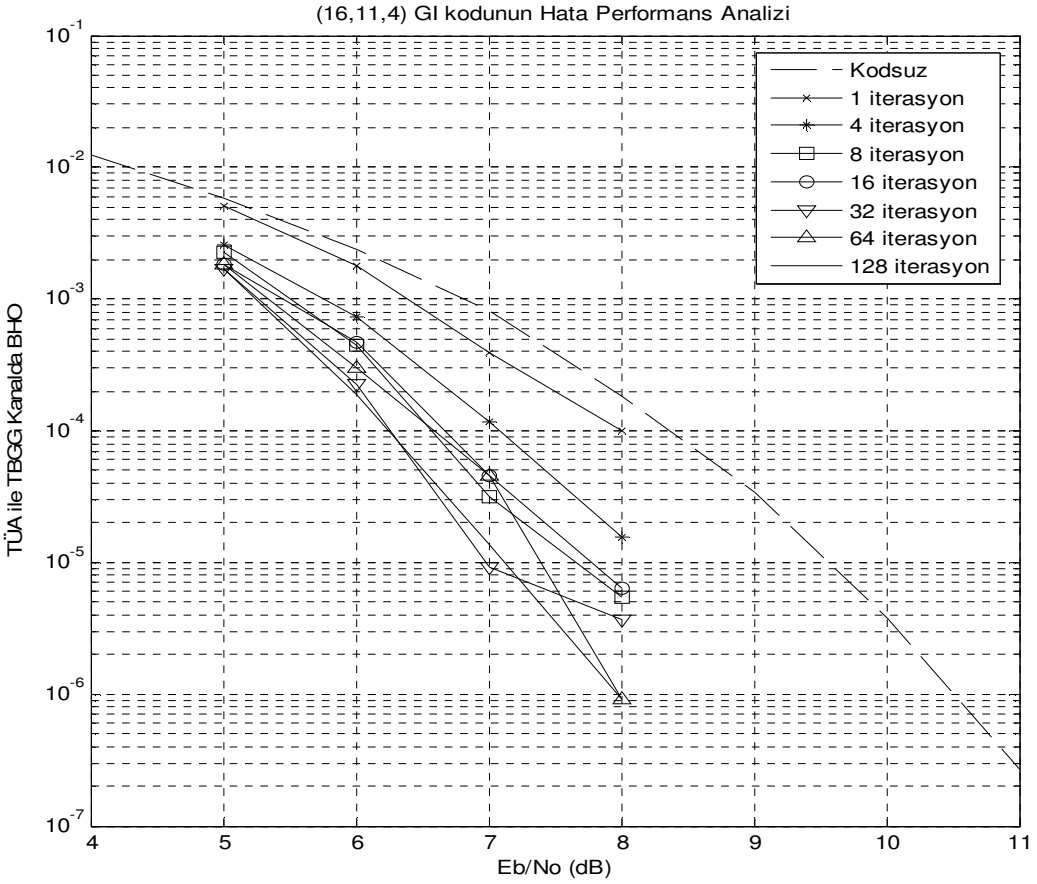


Şekil 4.5 Benzetimi yapılan haberleşme sistemi

Yukarıdaki sistemin benzetiminden BHO ve KHO olarak aşağıdaki şekiller elde edilmiştir. Tüm şekiller için hemen yanında yorumları da yapılmıştır. Bu bölümün sonunda da şekillerden çıkarılan sonuçlar topluca anlatılmıştır.

4.5.4.1 İkinin Kuvvetleri Şeklinde Olan Gİ Kodlarının Performans Analizi

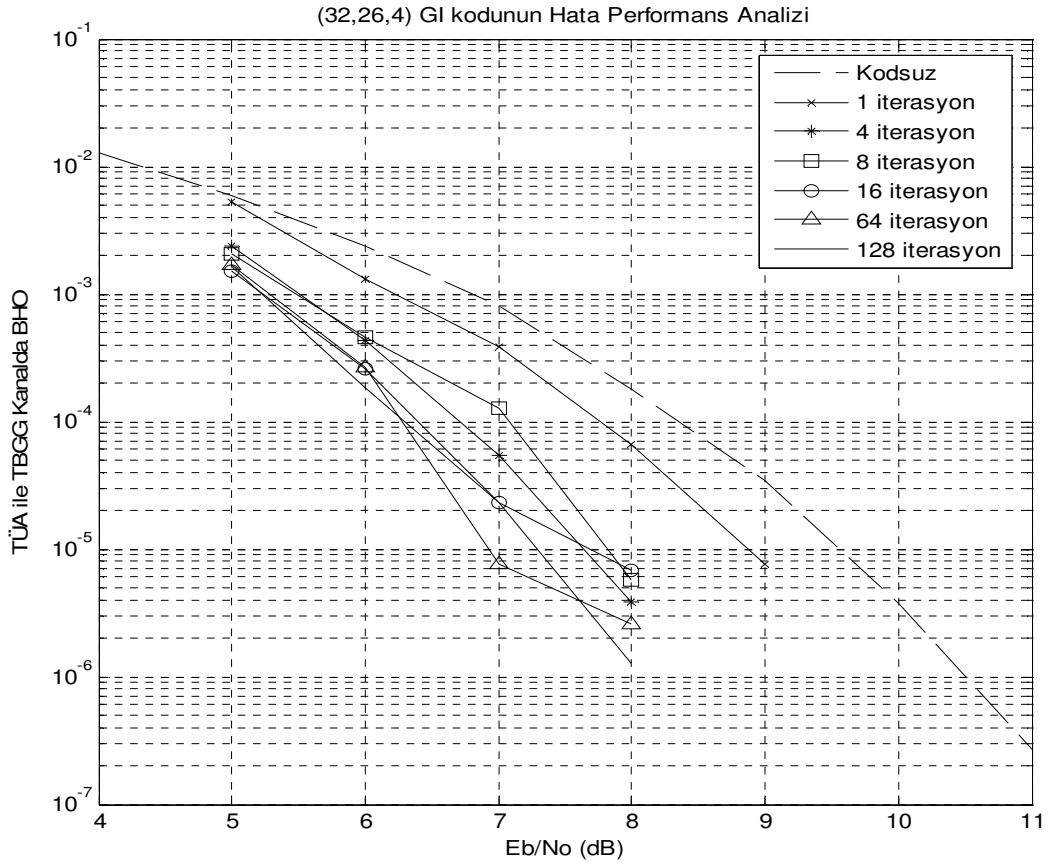
Şekil 4.6'da görüldüğü gibi, (16, 11, 4) Gİ kodu kullanılarak 32 iterasyonla elde edilen kodlama kazancı, 10^{-5} BHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 2.7 dB'dir. 32 iterasyondan sonra performansta kayda değer bir gelişme olmadığı görülmüştür. Bu da Gİ kodu TÜA kullanıldığında en uygun BHO değerine hızlı bir şekilde ulaşıldığını göstermektedir.



İter-Eb/N0	4	5	6	7	8	9	10	11
Kodsuz	1.2593E-02	5.8520E-03	2.4010E-03	8.1700E-04	1.8100E-04	3.4000E-05	3.8000E-06	2.7000E-07
1		5.0682E-03	1.7773E-03	3.9545E-04	1.0091E-04			
4		2.5864E-03	7.3636E-04	1.1818E-04	1.5455E-05			
8		2.2364E-03	4.5000E-04	3.1818E-05	5.4545E-06			
16		1.8318E-03	4.6364E-04	4.5455E-05	6.3636E-06			
32		1.7091E-03	2.2727E-04	9.0909E-06	3.6364E-06			
64		1.8545E-03	3.0455E-04	4.5455E-05	9.0909E-07			
128		1.6864E-03	1.8636E-04	1.3636E-05	9.0909E-07			

Şekil 4.6 C = (16, 11, 4) Gİ kodunun TBGG kanalda toplam ürün kod çözme algoritması yardımıyla elde edilen BHO.

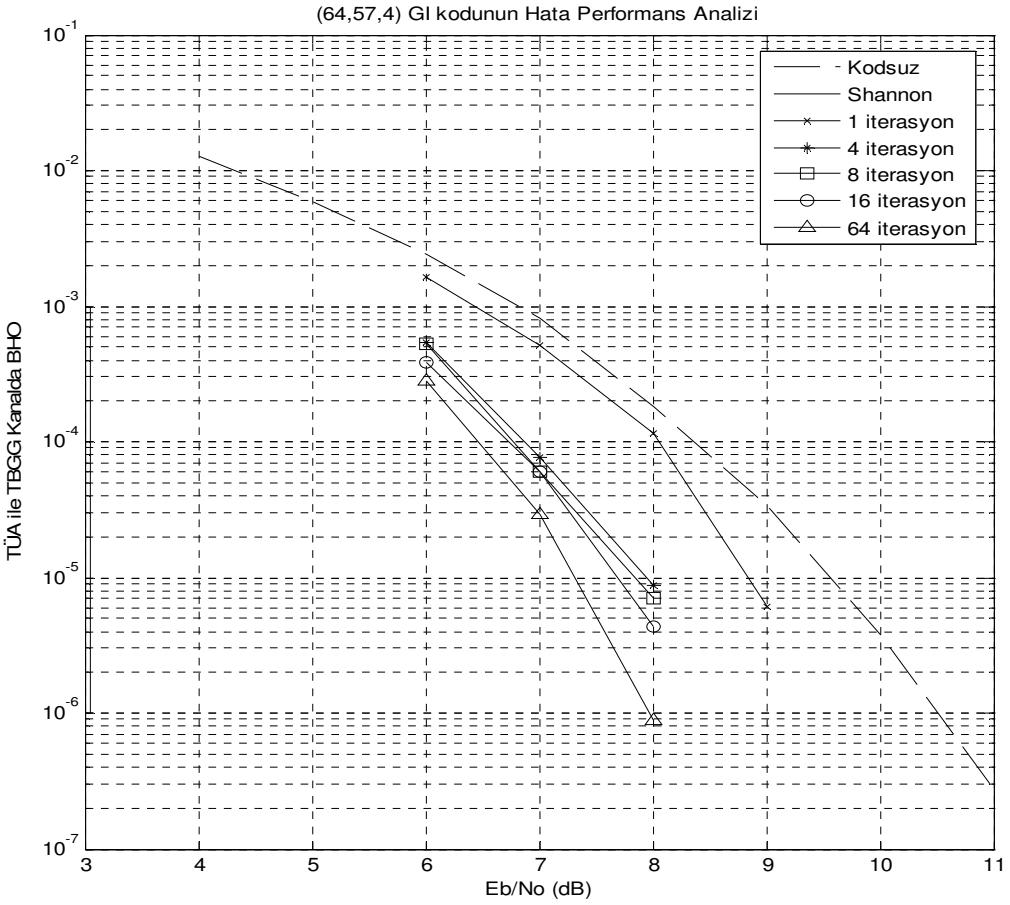
Şekil 4.7'de görüldüğü gibi, (32, 26, 4) Gİ kodu kullanılarak 64 iterasyonla elde edilen kodlama kazancı, 10^{-5} BHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 2.8 dB'dir. 64 iterasyondan sonra SNR'ın 7dB değerine ulaşmasına kadar performansta kayda değer bir gelişme olmadığı görülmüştür. Bundan sonra, iterasyon sayısının arttırılması küçük bir BHO kazancına sebep olmaktadır. Öte yandan, sistemin performansını değerlendirmek için genellikle BHO'yu 10^{-5} olarak dikkate almaktayız.



İter-Eb/N0	4	5	6	7	8	9	10	11
Kodsuz	1.2593E-02	5.8520E-03	2.4010E-03	8.1700E-04	1.8100E-04	3.4000E-05	3.8000E-06	2.7000E-07
1		5.2577E-03	1.3269E-03	3.8462E-04	6.6346E-05	7.6923E-06		
4		2.3885E-03	4.3462E-04	5.3846E-05	3.8462E-06			
8		2.0615E-03	4.5769E-04	1.2692E-04	5.7692E-06			
16		1.5000E-03	2.5769E-04	2.3077E-05	6.7308E-06			
64		1.6769E-03	2.6538E-04	7.6923E-06	2.5641E-06			
128		1.6269E-03	1.8462E-04	2.3077E-05	1.2821E-06			

Şekil 4.7 C = (32, 26, 4) Gİ kodunun TBGG kanalda toplam ürün kod çözme algoritması yardımıyla elde edilen BHO.

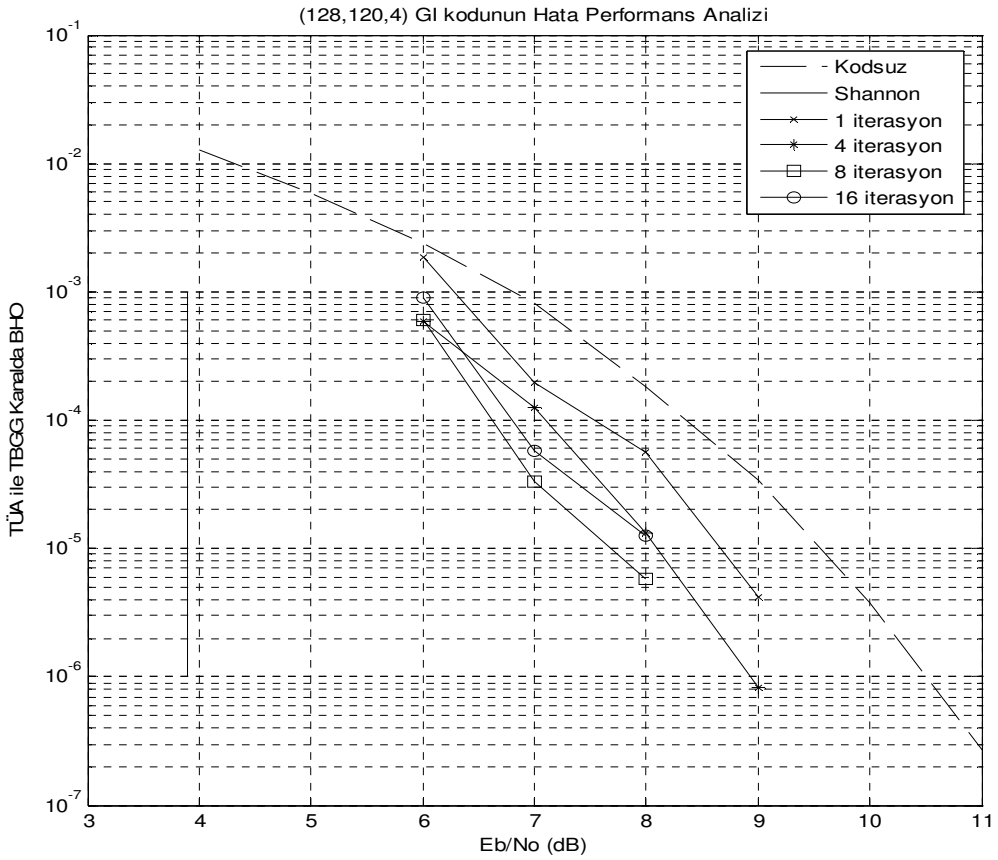
Şekil 4.8'de görüldüğü gibi, (64, 57, 4) Gİ kodu kullanılarak 64 iterasyonla elde edilen kodlama kazancı, 10^{-5} BHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 2.2 dB'dir. Benzer şekilde 16 iterasyonla karşılaştırıldığında kod, sadece 0.35 dB'lik daha iyi bir kodlama kazancı sağlamaktadır. 64 iterasyondan sonra SNR'ın 8dB değerine ulaşmasına kadar performansta kayda değer bir gelişme olmadığı görülmektedir.



İter-Eb/N0	4	5	6	7	8	9	10	11
Kodsuz	1.2593E-02	5.8520E-03	2.4010E-03	8.1700E-04	1.8100E-04	3.4000E-05	3.8000E-06	2.7000E-07
1			1.6404E-03	5.1491E-04	1.1667E-04	6.1404E-06		
4			5.4474E-04	7.7193E-05	8.7719E-06			
8			5.2719E-04	5.9649E-05	7.0175E-06			
16			3.8684E-04	5.9649E-05	4.3860E-06			
64			2.8509E-04	2.8947E-05	8.7719E-07			

Şekil 4.8 C = (64, 57, 4) Gİ kodunun TBGG kanalda toplam ürün kod çözme algoritması yardımıyla elde edilen BHO.

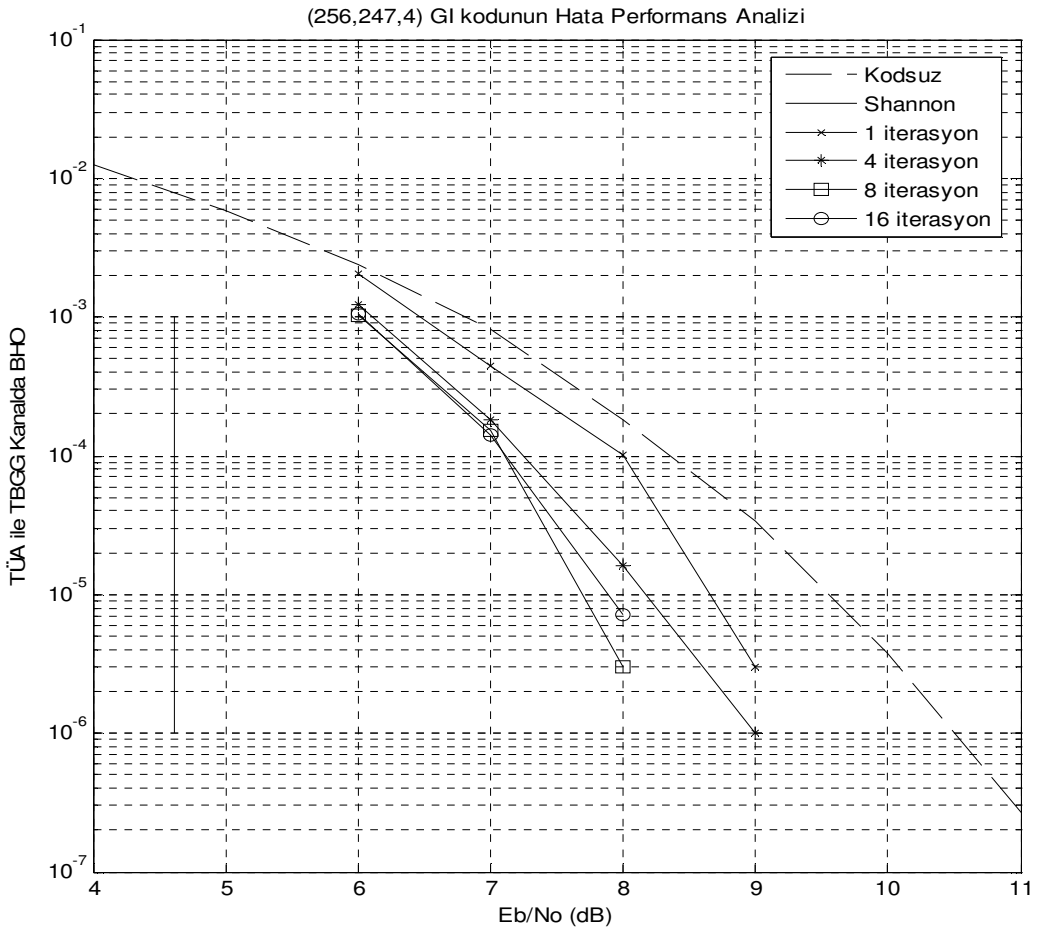
Şekil 4.9'da görüldüğü gibi, (128, 120, 4) kodlama kazancı $R=k/n$=120/128= 0.9375 olan Gİ kodu kullanılarak 8 iterasyonla elde edilen kodlama kazancı, 10^{-5} BHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 1.9 dB'dir. 8 iterasyondan sonra SNR'ın 8dB değerine ulaşmasına kadar performansta kayda değer bir gelişme olmadığı görülmüştür. Bu da Gİ kodunun en iyi başarım değerine hızlı bir şekilde ulaştığını göstermektedir. Bu arada, Shannon sınırı (3 dB) kodun BHO eğrisine daha yakındır.



İter-Eb/N0	4	5	6	7	8	9	10	11
Kodsuz	1.2593E-02	5.8520E-03	2.4010E-03	8.1700E-04	1.8100E-04	3.4000E-05	3.8000E-06	2.7000E-07
1			1.8500E-03	1.9833E-04	5.5833E-05	4.1667E-06		
4			5.8333E-04	1.2500E-04	1.3333E-05	8.3333E-07		
8			6.0000E-04	3.3333E-05	5.8333E-06			
16			8.9167E-04	5.8333E-05	1.2667E-05			

Şekil 4.9 C = (128, 120, 4) Gİ kodunun TBGG kanalda toplam ürün kod çözme algoritması yardımıyla elde edilen BHO.

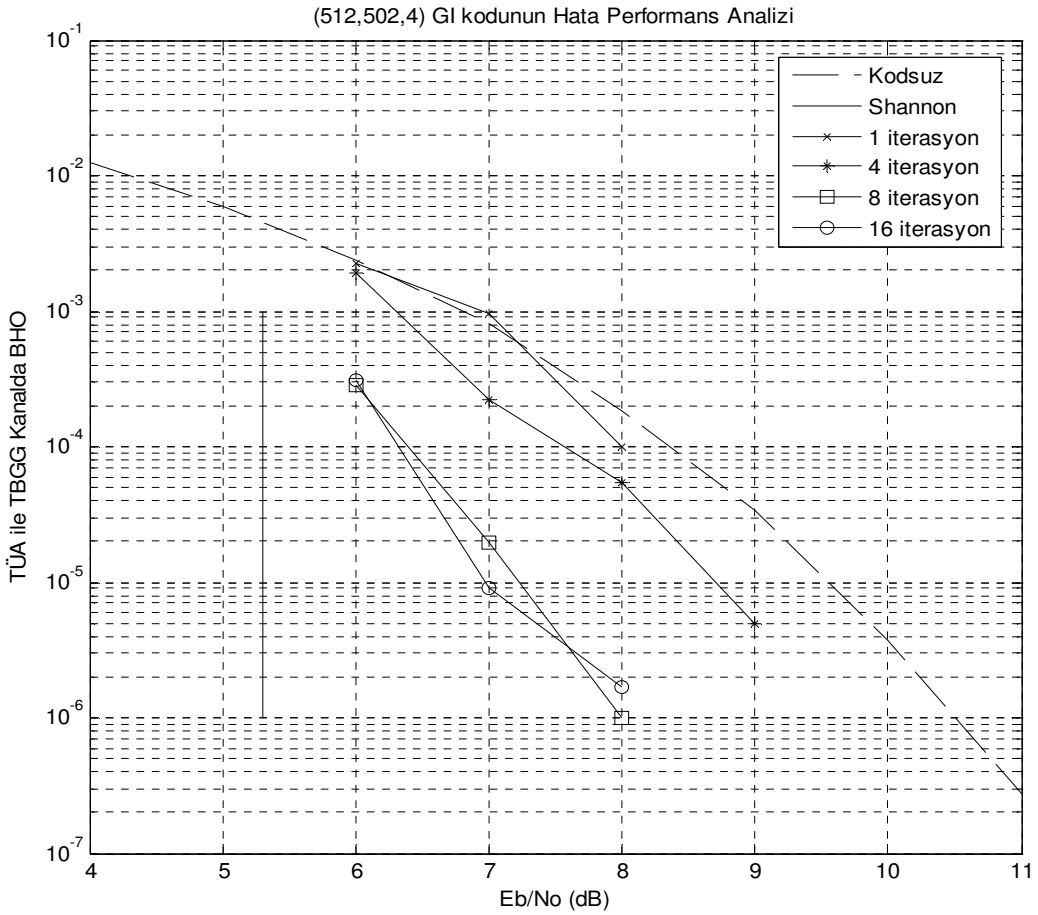
Şekil 4.10'da görüldüğü gibi, (256, 247, 4) kodlama kazancı R = 0.9648 olan Gİ kodu kullanılarak sadece 8 iterasyonla elde edilen kodlama kazancı, 10^{-5} BHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 1.9 dB'dir. 8 iterasyondan sonra SNR'ın 8dB değerine ulaşmasına kadar performansta kayda değer bir gelişme olmadığı görülmüştür. Şekilden de görüldüğü gibi Shannon sınırı (3.8 dB) kodun BHO eğrisine yakındır.



İter-Eb/N0	4	5	6	7	8	9	10	11
Kodsuz	1.2593E-02	5.8520E-03	2.4010E-03	8.1700E-04	1.8100E-04	3.4000E-05	3.8000E-06	2.7000E-07
1			2.0344E-03	4.4534E-04	1.0121E-04	3.0364E-06		
4			1.2449E-03	1.8219E-04	1.6194E-05	1.0121E-06		
8			1.0324E-03	1.5182E-04	3.0364E-06			
16			1.0526E-03	1.4170E-04	7.0850E-06			

Şekil 4.10 C = (256, 247, 4) Gİ kodunun TBGG kanalda toplam ürün kod çözme algoritması yardımıyla elde edilen BHO.

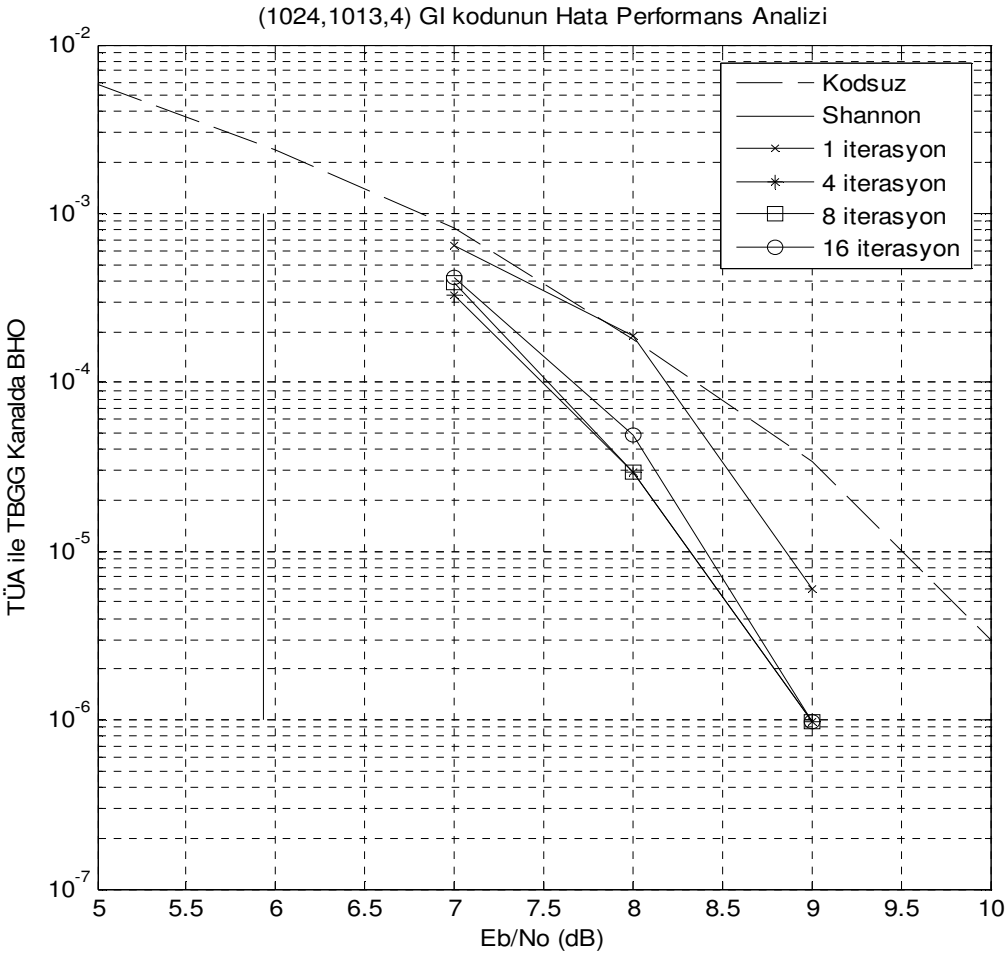
Şekil 4.11'de görüldüğü gibi, (512, 502, 4) kodlama kazancı R = 0.9805 olan Gİ kodu kullanılarak 16 iterasyonla elde edilen kodlama kazancı, 10^{-5} BHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 1.7 dB'dir ve Shannon sınırından 2.7 dB uzaktır.16 iterasyondan sonra SNR'ın 7dB değerine ulaşmasına kadar performansta kayda değer bir gelişme olmadığı görülmüştür.



İter-Eb/N0	4	5	6	7	8	9	10	11
Kodsuz	1.2593E-02	5.8520E-03	2.4010E-03	8.1700E-04	1.8100E-04	3.4000E-05	3.8000E-06	2.7000E-07
1			2.2659E-03	9.7112E-04	9.9602E-05			
4			1.9074E-03	2.2410E-04	5.4781E-05	4.9801E-06		
8			2.8884E-04	1.9920E-05	9.9602E-07			
16			3.0876E-04	8.9641E-06	1.6920E-06			

Şekil 4.11 C = (512, 502, 4) Gİ kodunun TBGG kanalda toplam ürün kod çözme algoritması yardımıyla elde edilen BHO.

Diğer iyi bir sonuç da, Şekil 4.12'de görüldüğü gibi, (1024, 1013, 4) kodlama oranı R = 0.9893 olan Gİ kodu kullanılarak 8 iterasyonla elde edilen kodlama kazancı, 10^{-5} BHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 1.2 dB'dir ve Shannon sınırına oldukça yakındır (2.35 dB). 8 iterasyondan sonra SNR'ın 8dB değerine ulaşmasına kadar performansta kayda değer bir gelişme olmadığı görülmüştür.

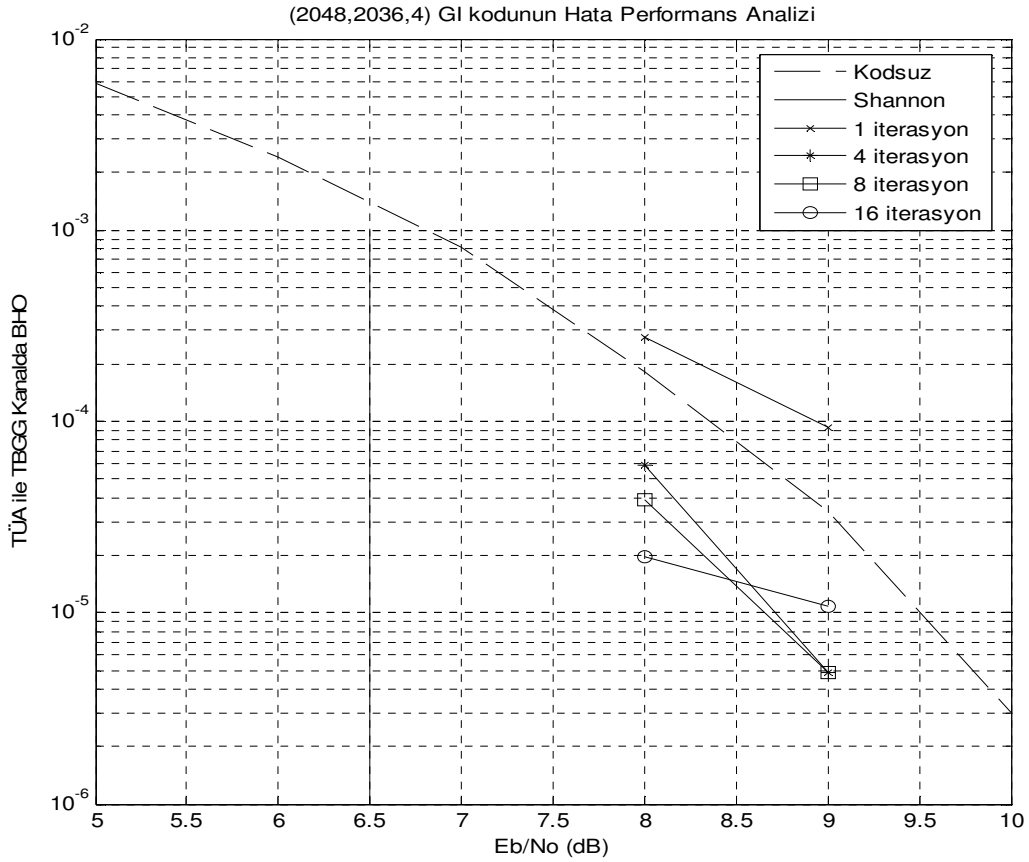


İter-Eb/N0	4	5	6	7	8	9	10
Kodsuz	1.2593E-02	5.8520E-03	2.4010E-03	8.1700E-04	1.8100E-04	3.4000E-05	3.8000E-06
1				6.5153E-04	1.8756E-04	5.9230E-06	
4				3.2577E-04	2.9615E-05	9.8717E-07	
8				3.8500E-04	2.9615E-05	9.8717E-07	
16				4.1461E-04	4.9358E-05	9.8717E-07	

Şekil 4.12 C = (1024, 1013, 4) Gİ kodunun TBGG kanalda toplam ürün kod çözme algoritması yardımıyla elde edilen BHO.

Şekil 4.13'de görüldüğü gibi, (2048, 2036, 4) kodlama kazancı R = 0.9941 olan Gİ kodu kullanılarak 16 iterasyonla elde edilen kodlama kazancı, 10^{-5} BHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 1.2 dB'dir. Bu arada, Shannon sınırından yalnızca 1.5 dB uzak olması oldukça önemlidir. BHO 10^{-5} ve sadece 8 iterasyonla eğri, Shannon sınırından 2.2 dB uzaktır ve kodlama kazancı da 0.9 dB'dir.

16 iterasyondan sonra SNR'ın 7dB değerine ulaşmasına kadar performansta kayda değer bir gelişme olmadığı görülmüştür.

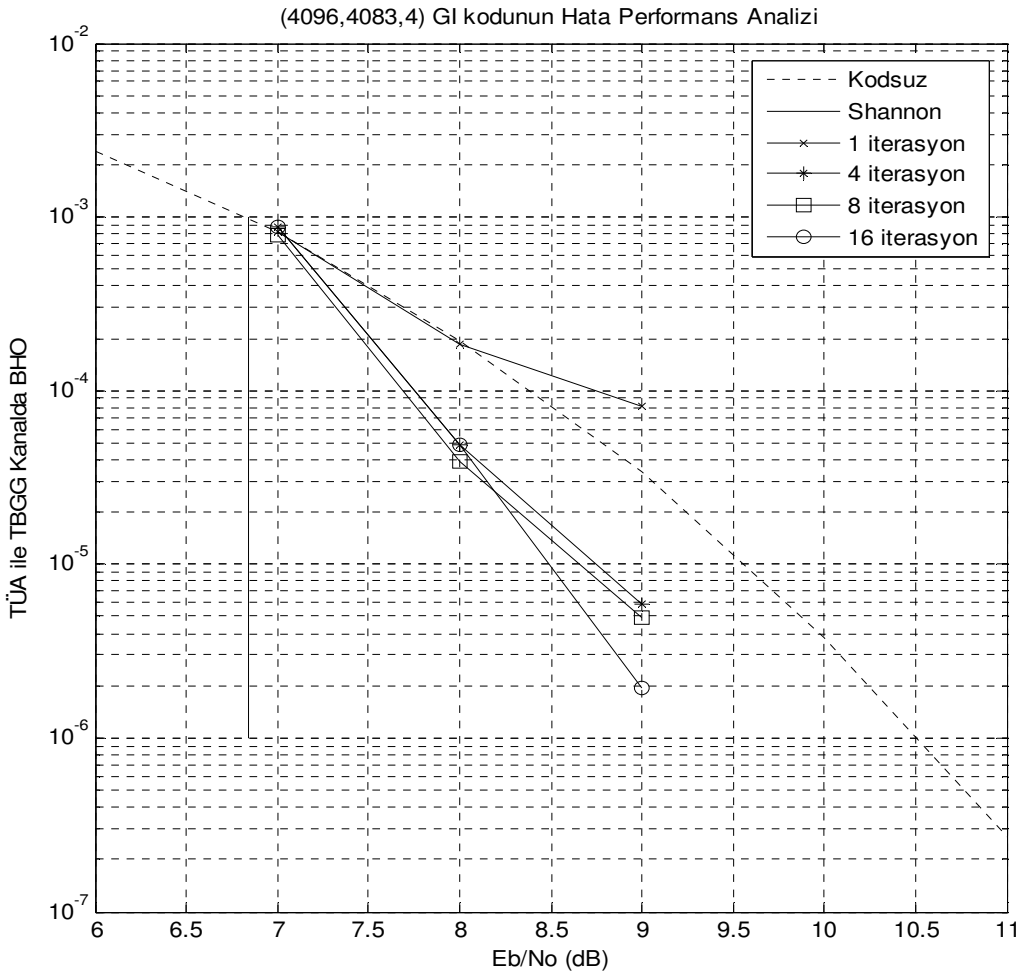


İter-Eb/N0	4	5	6	7	8	9	10
Kodsuz	1.2593E-02	5.8520E-03	2.4010E-03	8.1700E-04	1.8100E-04	3.4000E-05	3.8000E-06
1					2.7505E-04	9.3320E-05	
4					5.8939E-05	4.9116E-06	
8					3.9293E-05	4.9116E-06	
16					1.9646E-05	1.0752E-05	

Şekil 4.13 C = (2048, 2036, 4) Gİ kodunun TBGG kanalda toplam ürün kod çözme algoritması yardımıyla elde edilen BHO.

Şekil 4.14'de görüldüğü gibi, (4096, 4083, 4) kodlama kazancı R = 0.9968 olan Gİ kodu kullanılarak 16 iterasyonla elde edilen kodlama kazancı, 10^{-5} BHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 1.1 dB'dir. Bu arada, eğri, Shannon sınırından 1.6 dB uzaktadır.

16 iterasyondan sonra SNR'ın 9dB değerine ulaşmasına kadar performansta kayda değer bir gelişme olmadığı görülmüştür.

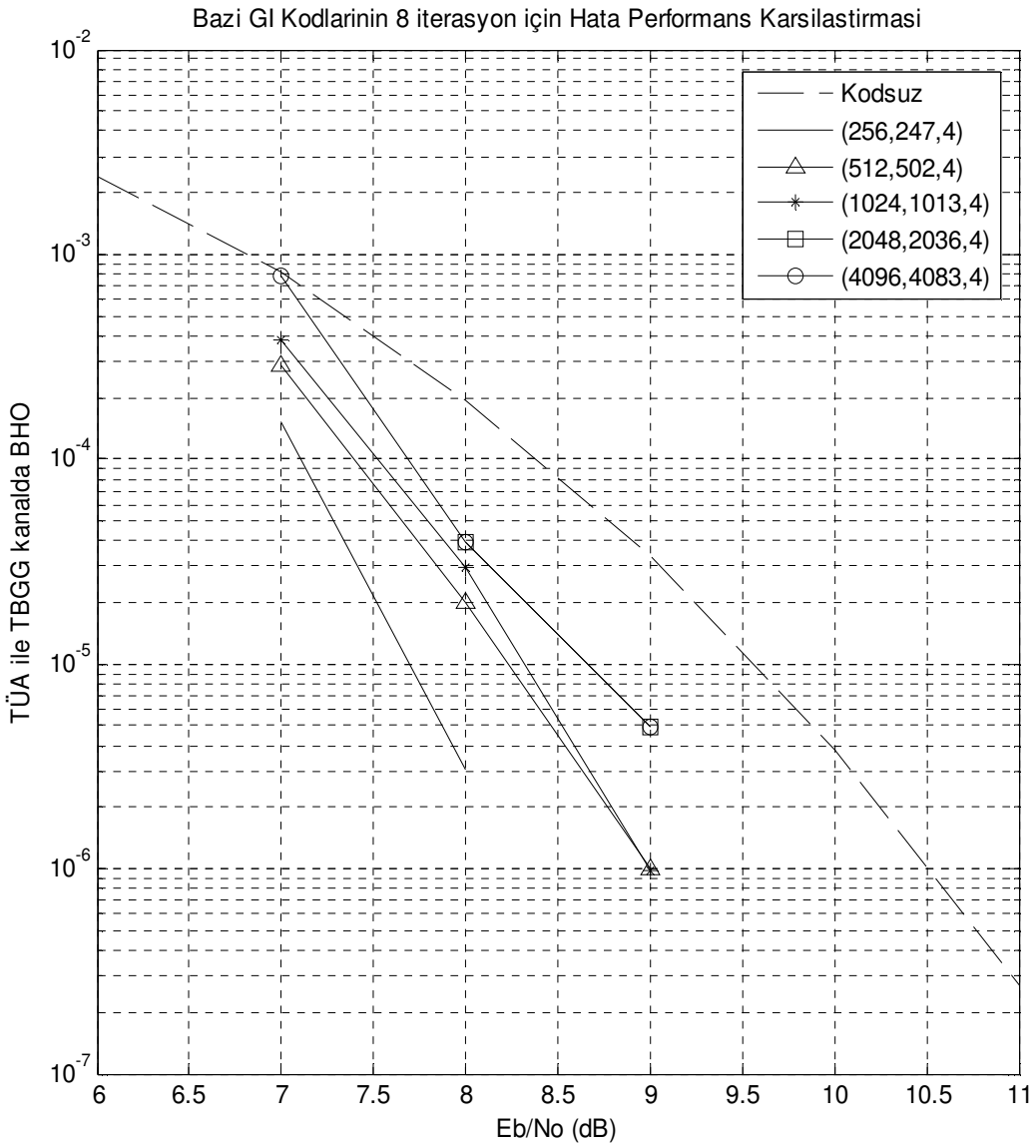


İter-Eb/N0	4	5	6	7	8	9	10
Kodsuz	1.2593E-02	5.8520E-03	2.4010E-03	8.1700E-04	1.8100E-04	3.4000E-05	3.8000E-06
1				8.1700E-04	1.8614E-04	8.1102E-05	
4				8.8170E-04	4.8984E-05	5.8780E-06	
8				7.8374E-04	3.9187E-05	4.8984E-06	
16				8.8170E-04	4.8984E-05	1.9593E-06	

Şekil 4.14 C = (4096, 4083, 4) Gİ kodunun TBGG kanalda toplam ürün kod çözme algoritması yardımıyla elde edilen BHO.

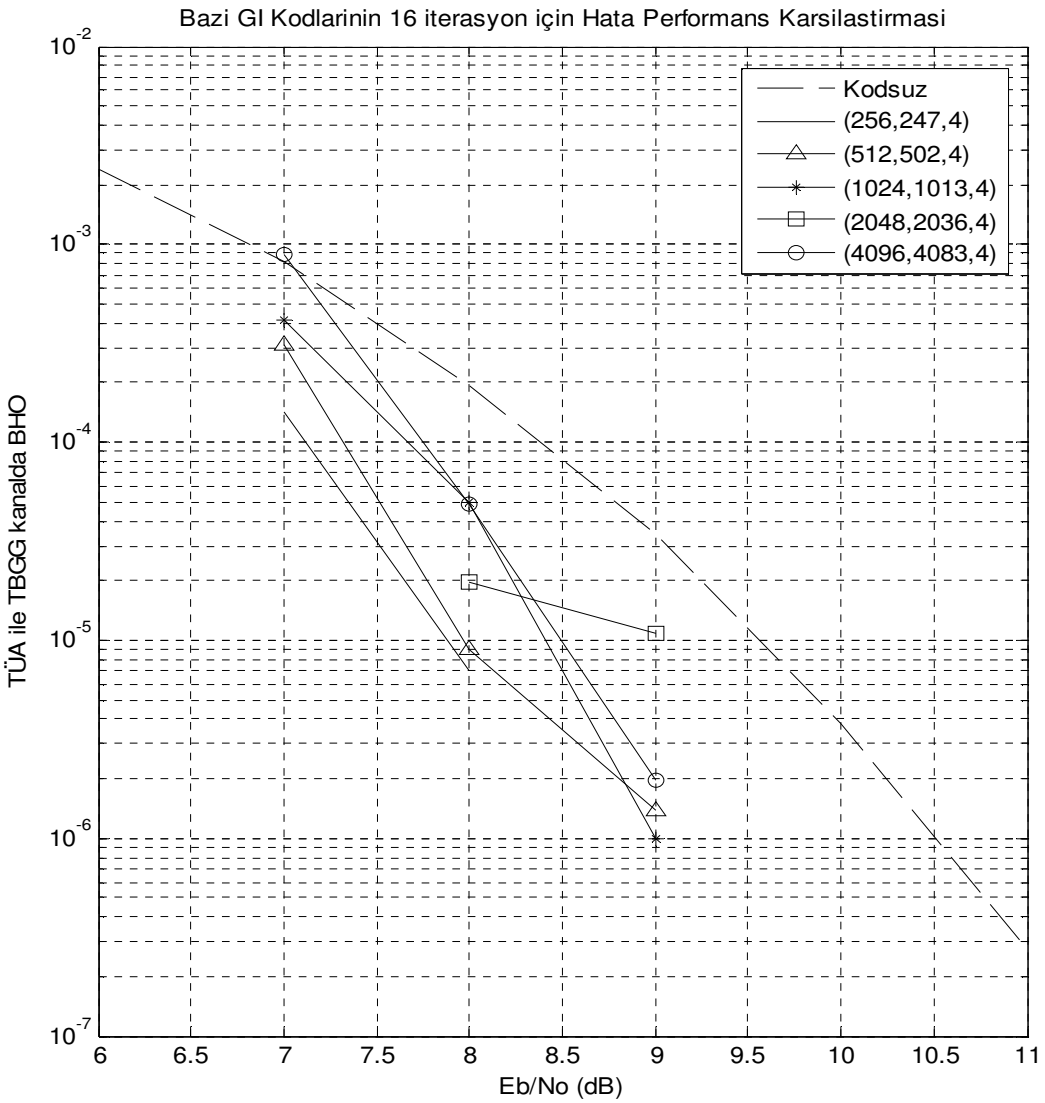
Kodlar arasındaki bazı karşılaştırmalar Şekil 4.15 ve 4.16'da gösterilmiştir. Bu şekillerde çok yüksek oranda ve büyüklükteki Gİ kodlarını dikkate aldık. 10^{-5} BHO'nı göz önüne aldığımızda 8 ve 16 iterasyon için her iki şeklin de başarı açısından aynı dereceye sahip olduğunu görürüz. İki şekilde de görüldüğü gibi, (256, 247, 4) Gİ kodu en iyi performansa sahiptir. Diğer kodları da performans açısından sıralarsak, (512,502, 4) , (1024, 1013, 4), (2048, 2036, 4), and (4096, 4083, 4) şeklinde bir sıralama yapabiliriz. İterasyon sayısının arttırılmasının olumlu etkileri bu şekillerden daha iyi anlaşılmaktadır. Eğer, bu kodları 10^{-5} BHO'sunda Shannon sınırından uzaklığına göre sıralarsak, bu kez sıralamada birinci 1.6 dB ile (4096, 4083, 4) Gİ kodudur. Daha sonra, 2.2 dB ile (2048, 2036, 4), 2.35 dB ile (1024, 1013, 4), 2.7 dB ile (512, 502, 4), 2.9 dB ile (256, 247, 4) gelmektedir. Bu sıralamaların ışığında, Gİ kod oranı azalırken, kodlama kazancı artmakta ve kod oranı artarken performans Shannon sınırına yaklaştığı sonucuna varılmaktadır.

Buraya kadar, sıkça kullanılan bazı Gİ kodları üzerinde durduk. Aslında, Hamming mesafesi 4 olan Gİ kodları, kod boyutu 2'nin katları şeklindedir. Kod uzunluğunun 2'nin kuvvetleri şeklinde seçilmesinin sebebi, Hamming kodları gibi aynı oranlardaki kodlarla karşılaştırma yapılabilmesini sağlamaktır. Aynı zamanda, aynı orandaki Gİ kodlarını Hamming kodlarıyla karşılaştırmak için bazı Hamming kodlarının performansı da incelenmiş ve Gİ kodlarının 10^{-5} BHO'da 0.2 dB'den daha iyi bir kodlama kazancı sağladığı görülmüştür. Ayrıca, daha büyük Hamming mesafeli bazı kodların performansı incelendiğinde Hamming mesafesi 4 olan kodlar kadar iyi performansa sahip olmadıkları sonucuna varılmıştır. Sonuç olarak, çok yüksek oranlı Gİ kodları, 1.7 dB civarında önemli kodlama kazançlarına sahip ve performansları da Shannon limitine yakın iken daha fazla mesafedeki Gİ kodları, daha iyi kodlama oranı için bazı düzenlemelere ihtiyaç duymaktadır.



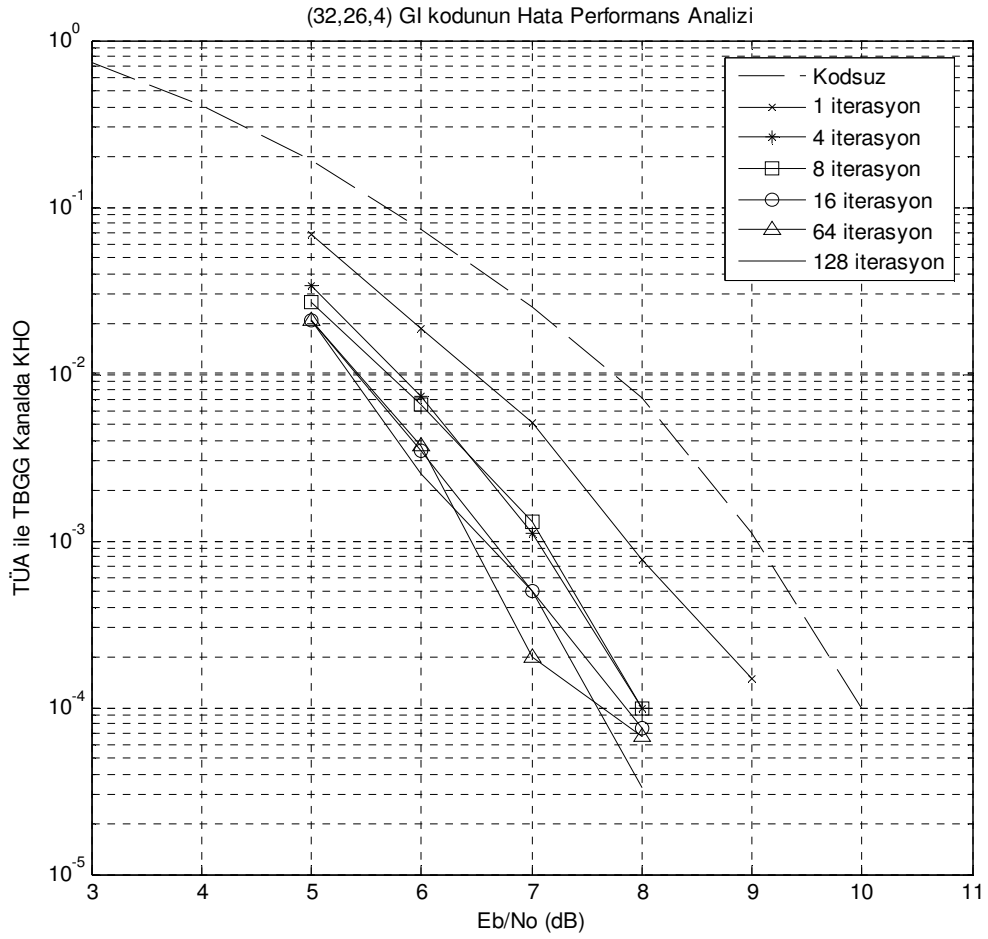
Kod-Eb/N0	6	7	8	9
(256,247,4)	1.0324E-03	1.5182E-04	3.0364E-06	
(512,502,4)	2.8884E-04	1.9920E-05	9.9602E-07	
(1024,1013,4)		3.8500E-04	2.9615E-05	9.8717E-07
(2048,2036,4)			3.9293E-05	4.9116E-06
(4096,4083,4)		7.8374E-04	3.9187E-05	4.8984E-06

Şekil 4.15 8 iterasyonla bazı Gİ kodlarının BHO Karşılaştırması



Kod-Eb/N0	6	7	8	9
(256,247,4)	1.0324E-03	1.5182E-04	3.0364E-06	
(512,502,4)	2.8884E-04	1.9920E-05	9.9602E-07	
(1024,1013,4)		3.8500E-04	2.9615E-05	9.8717E-07
(2048,2036,4)			3.9293E-05	4.9116E-06
(4096,4083,4)		7.8374E-04	3.9187E-05	4.8984E-06

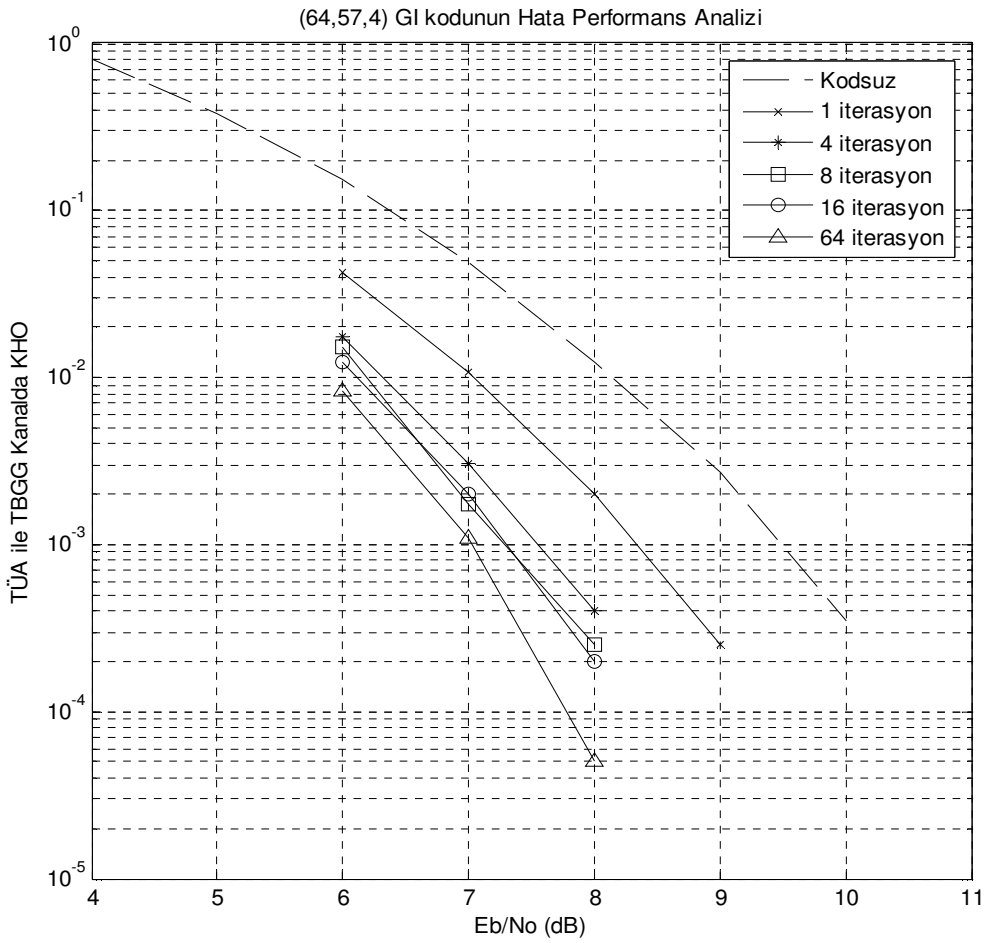
Şekil 4.16 16 iterasyonla bazı Gİ kodlarının BHO Karşılaştırması



İter-Eb/N0	5	6	7	8	9
1	6.8400E-02	1.8600E-02	5.1000E-03	7.7500E-04	1.5000E-04
4	3.3900E-02	7.3000E-03	1.1000E-03	1.0000E-04	
8	2.6900E-02	6.6000E-03	1.3000E-03	1.0000E-04	
16	2.0800E-02	3.5000E-03	5.0000E-04	7.5000E-05	
64	2.1200E-02	3.7000E-03	2.0000E-04	6.6667E-05	
128	2.1200E-02	2.5000E-03	5.0000E-04	3.3333E-05	

Şekil 4.17 C = (32, 26, 4) Gİ kodunun TBGG kanalda toplam ürün kod çözme algoritması yardımıyla elde edilen KHO.

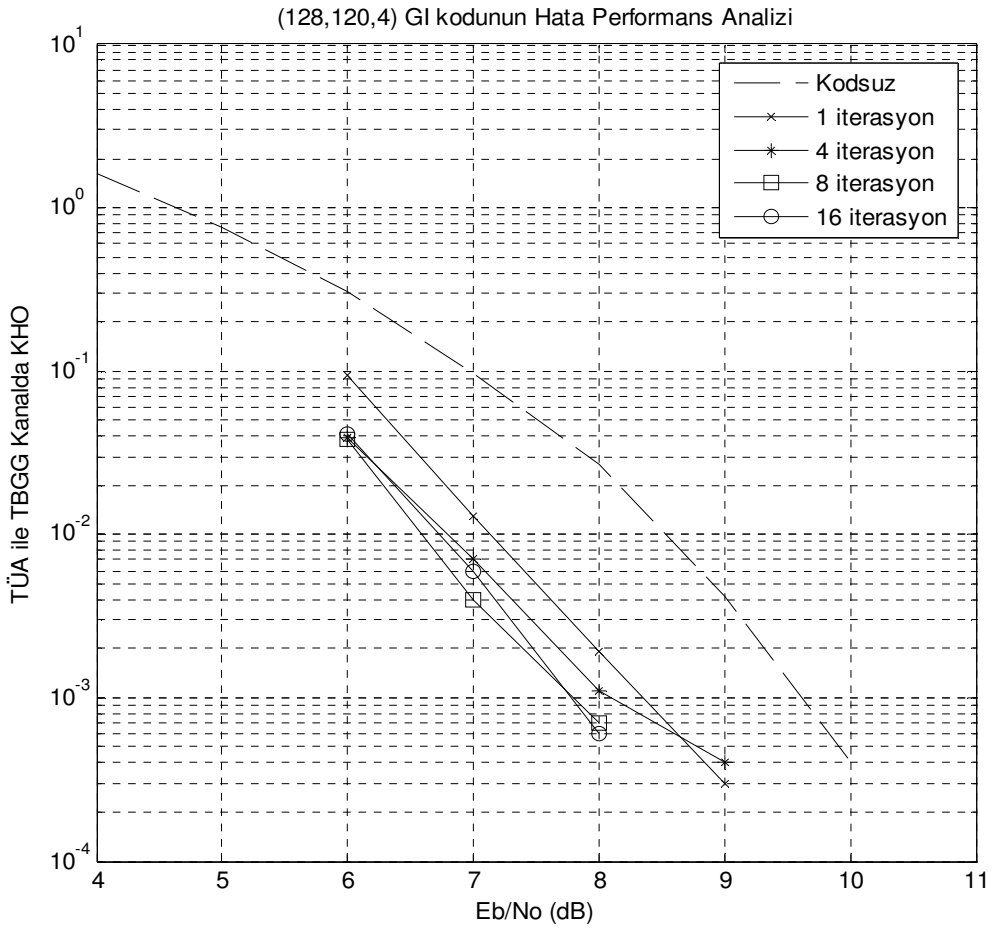
Şekil 4.17'de görüldüğü gibi, (32, 26, 4) Gİ kodu kullanılarak 64 iterasyonla elde edilen kodlama kazancı, 10^{-4} KHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 2.4 dB'dir. 64 iterasyondan sonra performansta kayda değer bir gelişme olmadığı görülmüştür. Bu da Gİ kodu TÜA kullanıldığında en uygun KHO değerine hızlı bir şekilde ulaşıldığını göstermektedir.



İter-Eb/N0	6	7	8	9
1	4.2950E-02	1.0800E-02	2.0000E-03	2.5000E-04
4	1.7400E-02	3.0500E-03	4.0000E-04	
8	1.5100E-02	1.7500E-03	2.5000E-04	
16	1.2250E-02	2.0000E-03	2.0000E-04	
64	8.3000E-03	1.1000E-03	5.0000E-05	

Şekil 4.18 C = (64, 57, 4) Gİ kodunun TBGG kanalda toplam ürün kodçözme algoritması yardımıyla elde edilen KHO

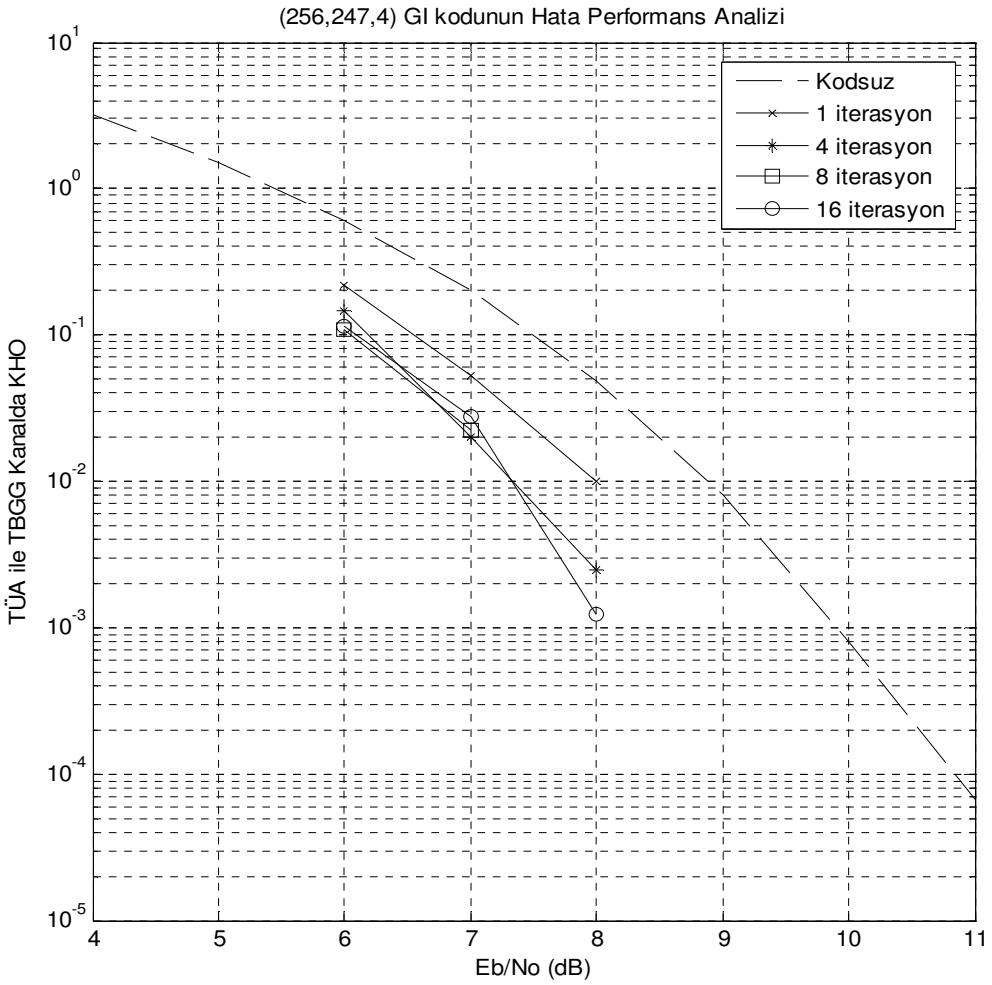
Şekil 4.18'de görüldüğü gibi, (64, 57, 4) Gİ kodu kullanılarak 64 iterasyonla elde edilen kodlama kazancı, 10-3 KHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 2.4 dB'dir. Benzer şekilde 8 ve 16 iterasyonla karşılaştırıldığında kod, sadece 2 dB'lik daha iyi bir kodlama kazancı sağlamaktadır. 64 iterasyondan sonra SNR'ın 8dB değerine ulaşmasına kadar performansta kayda değer bir gelişme olmamıştır.



İter-Eb/N0	6	7	8	9
1	9.4000E-02	1.3000E-02	1.9000E-03	3.0000E-04
4	3.9000E-02	7.0000E-03	1.1000E-03	4.0000E-04
8	3.8000E-02	4.0000E-03	7.0000E-04	
16	4.1000E-02	6.0000E-03	6.0000E-04	

Şekil 4.19 C = (128,120, 4) Gİ kodunun TBGG kanalda toplam ürün kodçözme algoritması yardımıyla elde edilen KHO

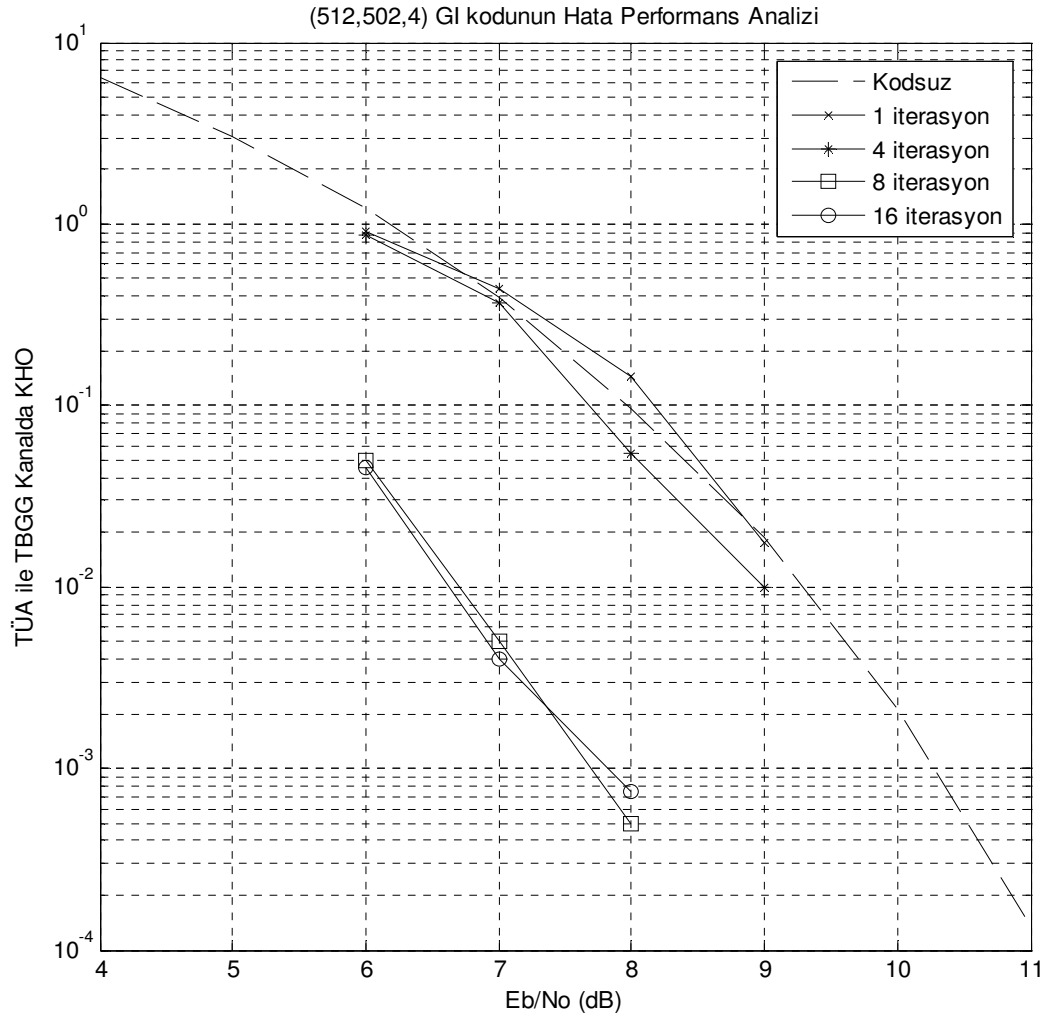
Şekil 4.19'da görüldüğü gibi, (128, 120, 4) kodlama kazancı $R=k/n$=120/128= 0.9375 olan Gİ kodu kullanılarak 8 ve 16 iterasyonla elde edilen kodlama kazancı, 10-3 KHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 1.9 dB'dir. 16 iterasyondan sonra SNR'ın 8dB değerine ulaşmasına kadar performansta kayda değer bir gelişme olmadığı görülmüştür. Bu da Gİ kodunun en iyi başarım değerine hızlı bir şekilde ulaştığını göstermektedir.



İter-Eb/N0	6	7	8
1	2.2000E-01	5.2500E-02	1.0000E-02
4	1.4750E-01	2.0000E-02	2.5000E-03
8	1.1000E-01	2.2500E-02	
16	1.1500E-01	2.7500E-02	1.2500E-03

Şekil 4.20 C = (256,247, 4) Gİ kodunun TBGG kanalda toplam ürün kodçözme algoritması yardımıyla elde edilen KHO

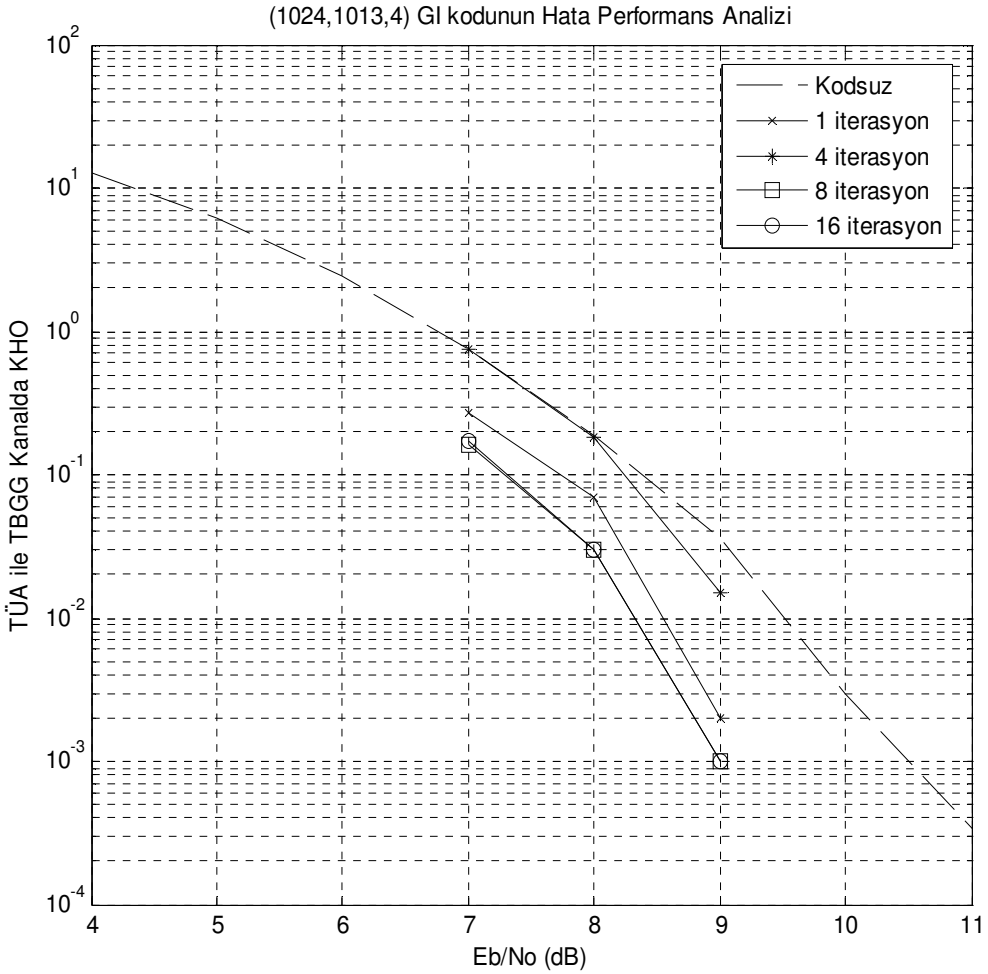
Şekil 4.20'de görüldüğü gibi, (256, 247, 4) kodlama kazancı R = 0.9648 olan Gİ kodu kullanılarak sadece 16 iterasyonla elde edilen kodlama kazancı, 10^{-3} KHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 1.9 dB'dir. 16 iterasyondan sonra SNR'ın 8dB değerine ulaşmasına kadar performansta kayda değer bir gelişme olmadığı görülmüştür.



İter-Eb/N0	6	7	8	9
1	9.0500E-01	4.4250E-01	1.4500E-01	1.7500E-02
4	8.6250E-01	3.6500E-01	5.5000E-02	1.0000E-02
8	5.0000E-02	5.0000E-03	5.0000E-04	
16	4.5000E-02	4.0000E-03	7.5000E-04	

Şekil 4.21 C = (512, 502, 4) Gİ kodunun TBGG kanalda toplam ürün kodçözme algoritması yardımıyla elde edilen KHO

Şekil 4.21'de görüldüğü gibi, (512, 502, 4) kodlama kazancı R = 0.9805 olan Gİ kodu kullanılarak 8 iterasyonla elde edilen kodlama kazancı, 10^{-3} KHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 2.5 dB'dir.8 iterasyondan sonra SNR'ın 8dB değerine ulaşmasına kadar performansta kayda değer bir gelişme olmadığı görülmüştür.



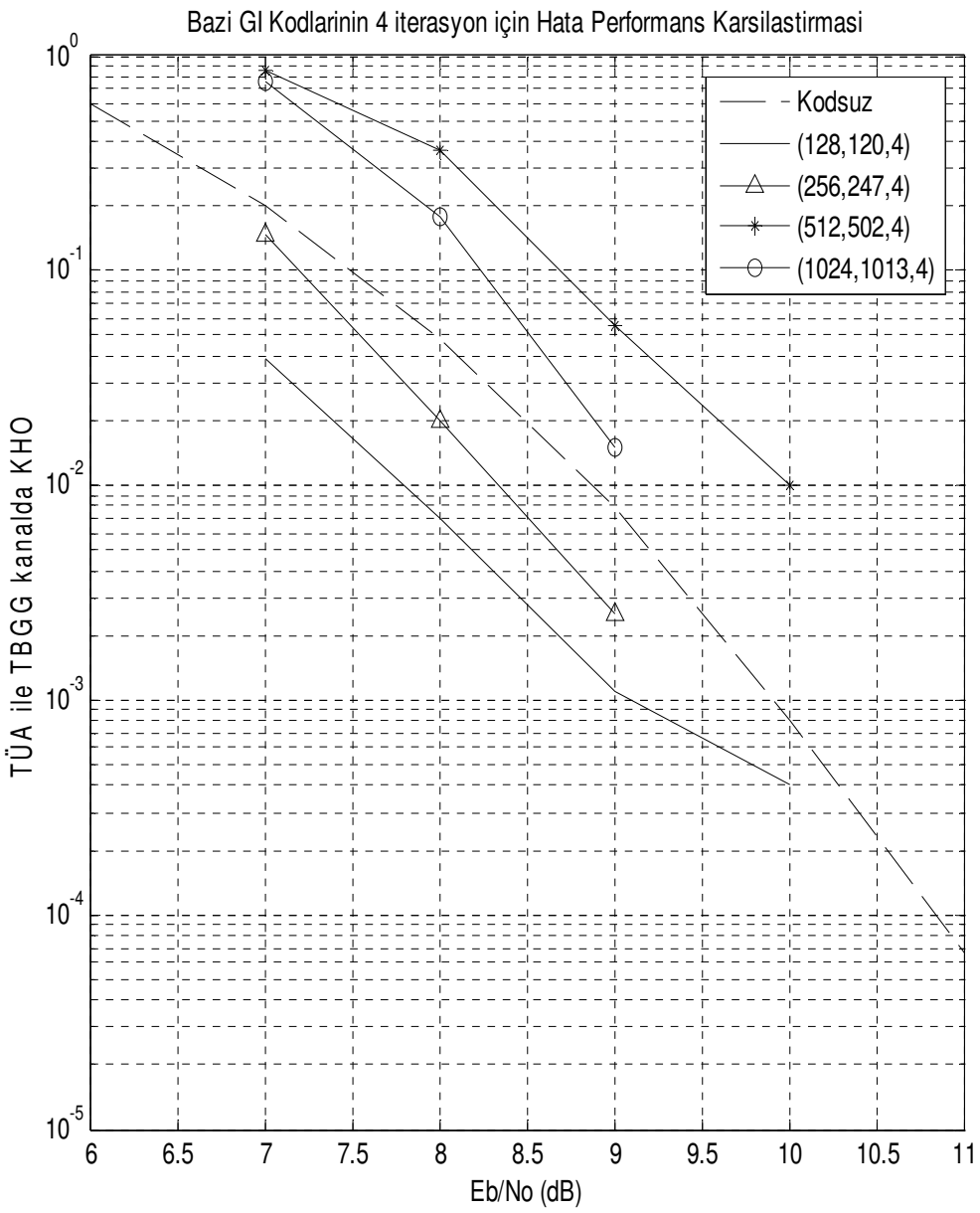
İter-Eb/N0	6	7	8
1	2.7000E-01	7.0000E-02	2.0000E-03
4	7.6000E-01	1.8000E-01	1.5000E-02
8	1.6000E-01	3.0000E-02	1.0000E-03
16	1.7000E-01	3.0000E-02	1.0000E-03

Şekil 4.22 C = (1024, 1013, 4) Gİ kodunun TBGG kanalda toplam ürün kodçözme algoritması yardımıyla elde edilen KHO

Diğer bir sonuç da Şekil 4.22'de görüldüğü gibi, (1024, 1013, 4) kodlama oranı $R = 0.9893$ olan Gİ kodu kullanılarak 16 iterasyonla elde edilen kodlama kazancı, 10^{-3} KHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 1.5 dB'dir. 16 iterasyondan sonra SNR'ın 8dB değerine ulaşmasına kadar performansta kayda değer bir gelişme olmadığı görülmüştür.

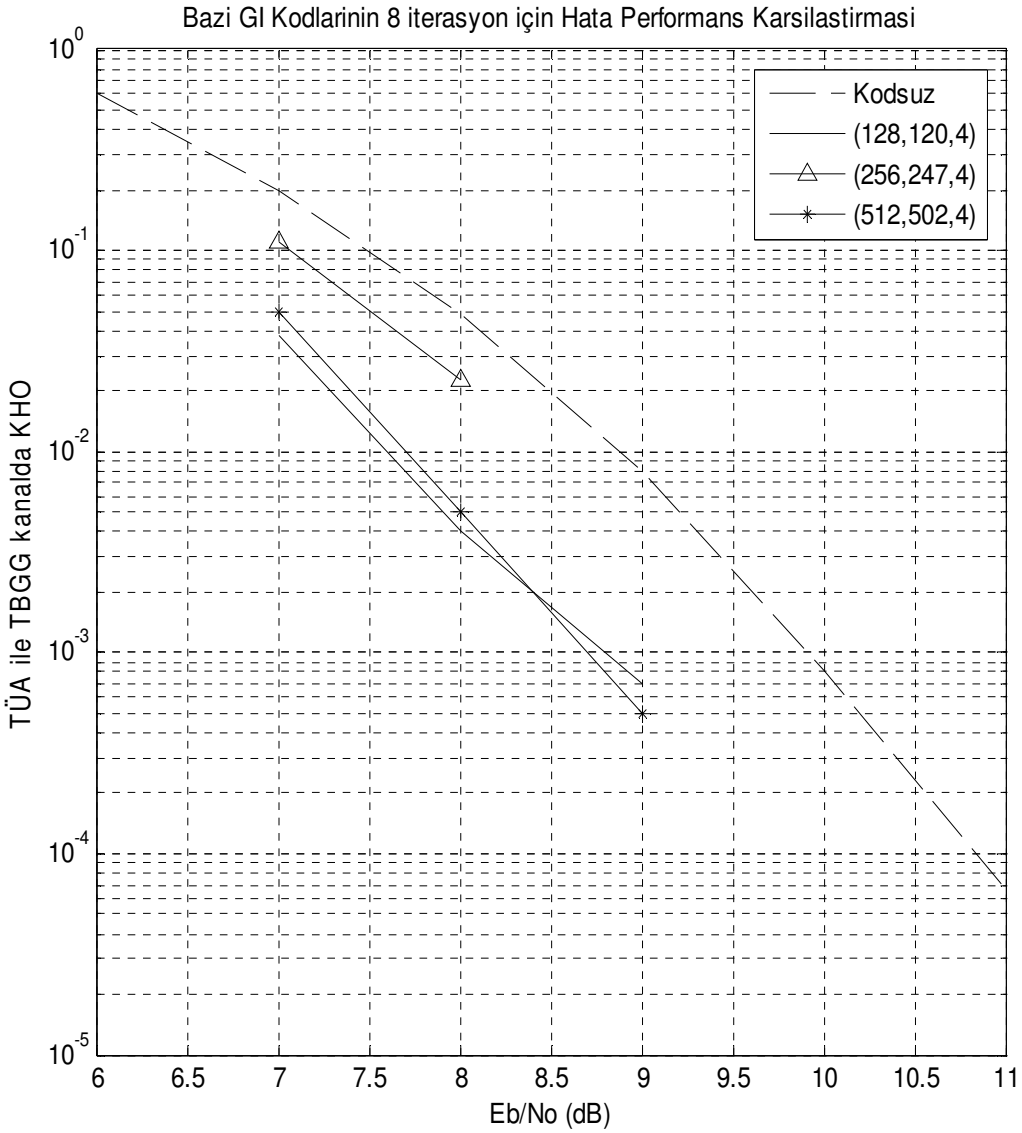
Kodlar arasındaki bazı karşılaştırmalar Şekil 4.23, 4.24 ve 4.25'de gösterilmiştir. Bu şekillerde çok yüksek oranda ve büyüklükteki Gİ kodlarını dikkate aldık. 10^{-3} BHO'yu göz önüne aldığımızda 8 ve 16 iterasyon için her iki şeklin de başarı açısından aynı dereceye sahip olduğunu görürüz. Şekillerden görüldüğü gibi, 4 iterasyon için (128, 120, 4), 8 iterasyon için (512, 502, 4), 16 iterasyon için (128, 120, 4) Gİ kodu en iyi perfomansa sahiptir. Diğer kodları da performans açısından sıralarsak, (512,502, 4) , (256, 247, 4), ve (1024, 1013, 4) şeklinde bir sıralama yapabiliriz. İterasyon sayısının arttırılmasının olumlu etkileri elde edilen şekillerden daha iyi anlaşılmaktadır. Bu sıralamaların ışığında, Gİ kod oranı azalırken, kodlama kazancının artmakta olduğu görülmüştür.

Sonuç olarak, çok yüksek oranlı Gİ kodları, BHO sonuçlarında olduğu gibi KHO parametresine göre de 1.7 dB civarında önemli kodlama kazançlarına sahipken daha fazla mesafedeki Gİ kodları, daha iyi kodlama oranı için bazı düzenlemelere ihtiyaç duymaktadır.



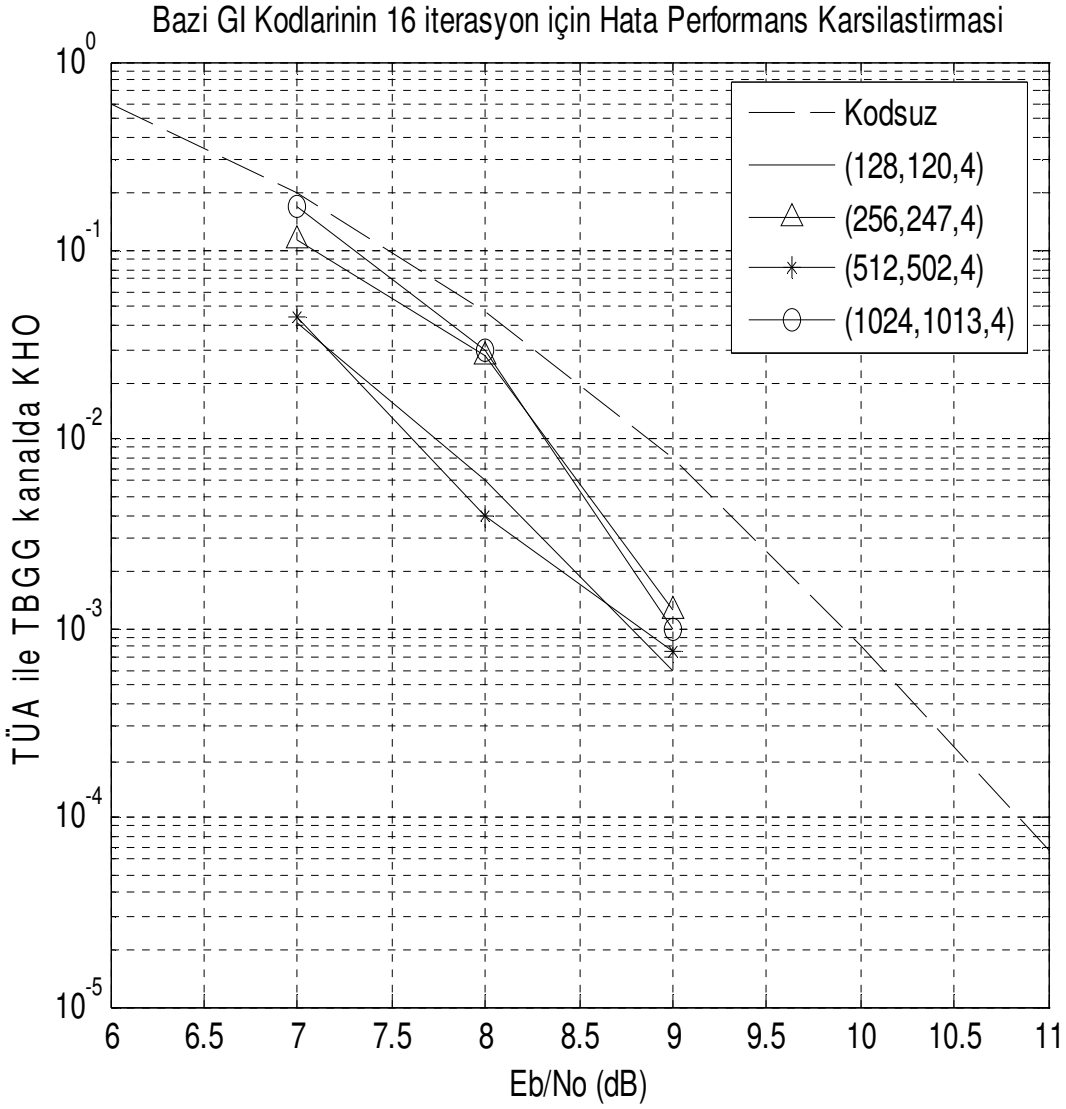
Kod-Eb/N0	6	7	8	9
(128,120,4)	3.9000E-02	7.0000E-03	1.1000E-03	4.0000E-04
(256,247,4)	1.4750E-01	2.0000E-02	2.5000E-03	
(512,502,4)	8.6250E-01	3.6500E-01	5.5000E-02	1.0000E-02
(1024,1013,4)	7.6000E-01	1.8000E-01	1.5000E-02	

Şekil 4.23 4 iterasyonla bazı Gİ kodlarının KHO Karşılaştırması



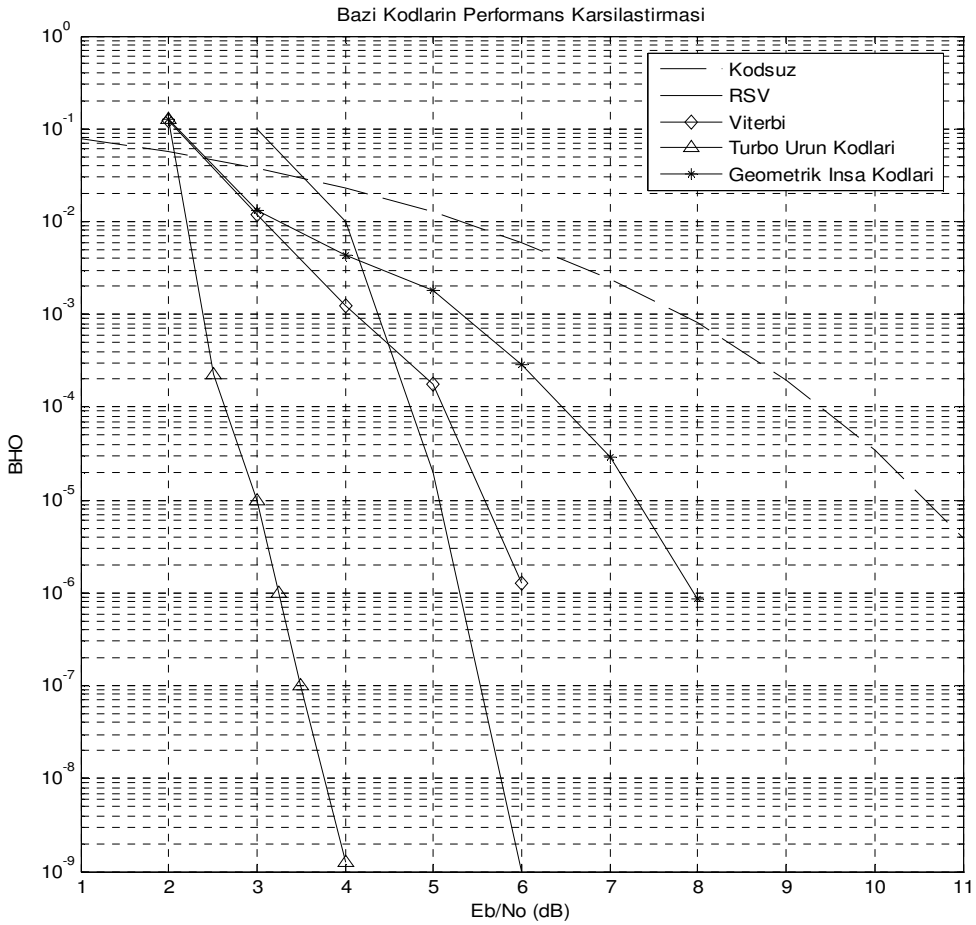
Kod-Eb/N0	6	7	8
(128,120,4)	3.8000E-02	4.0000E-03	7.0000E-04
(256,247,4)	1.1000E-01	2.2500E-02	
(512,502,4)	5.0000E-02	5.0000E-03	5.0000E-04
(1024,1013,4)	1.6000E-01	3.0000E-02	1.0000E-03

Şekil 4.24 8 iterasyonla bazı Gİ kodlarının KHO Karşılaştırması



Kod-Eb/N0	6	7	8
(128,120,4)	4.1000E-02	6.0000E-03	6.0000E-04
(256,247,4)	1.1500E-01	2.7500E-02	1.2500E-03
(512,502,4)	4.5000E-02	4.0000E-03	7.5000E-04
(1024,1013,4)	1.7000E-01	3.0000E-02	1.0000E-03

Şekil 4.25 16 iterasyonla bazı Gİ kodlarının KHO Karşılaştırması

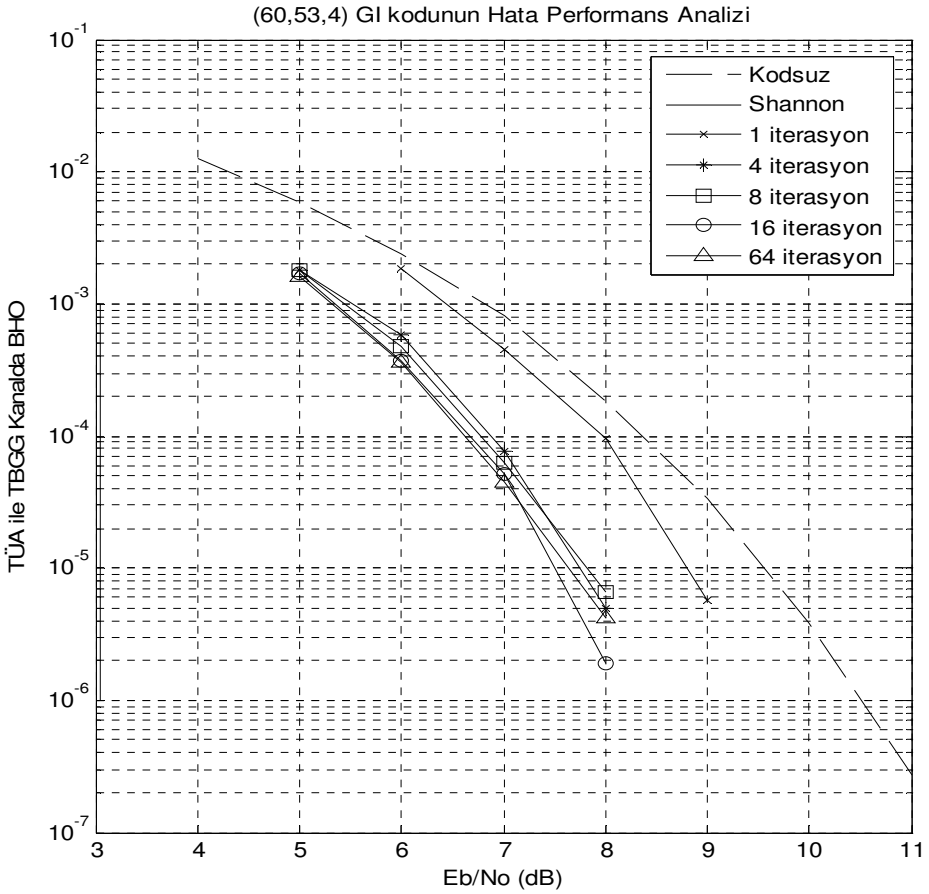


Kod-Eb/N0	2	3	4	5	6	7	8	9
Kodsuz	1.2593E-02	5.8520E-03	2.4010E-03	8.1700E-04	1.8100E-04	3.4000E-05	3.8000E-06	2.7000E-07
RSV		1.0245E-01	1.0350E-02	2.0230E-05	1.6300E-09			
Viterbi	1.2000E-01	1.2000E-02	1.2500E-03	1.7500E-04	1.2500E-06			
Turbo Ürün	1.2500E-01	1.0000E-05	1.2500E-09					
Gİ	1.2800E-01	1.3000E-02	4.3500E-03	1.8000E-03	2.8509E-04	2.8947E-05	8.7710E-07	

Şekil 4.26 Reed-Solomon-Viterbi, Viterbi, Turbo-Ürün, Geometrik İnşa Kodları'nın Performans Karşılaştırması

Şekil 4.26'dan görülebileceği gibi Geometrik İnşa Kodları'nın performansı diğer yöntemlere göre daha iyi değildir. 10^{-5} kodsuz durumla karşılaştırıldığında kodlama kazançları sırasıyla Turbo Ürün kodlarında 7.5 dB, Reed-Solomon- Viterbi Kodlarında 5.5 dB, Viterbi Kodları'nda 5 dB ve Geometrik İnşa Kodları'nda 3.25 dB'dir. Burada tasarlanan yöntemin avantajı, işlem karmaşıklığını azaltması ve daha büyük boyuttaki kodlar için de kod üretimi yapabilmesidir. Ayrıca, Gİ kodları daha yüksek oranda kod oranı sağlamaktadır.

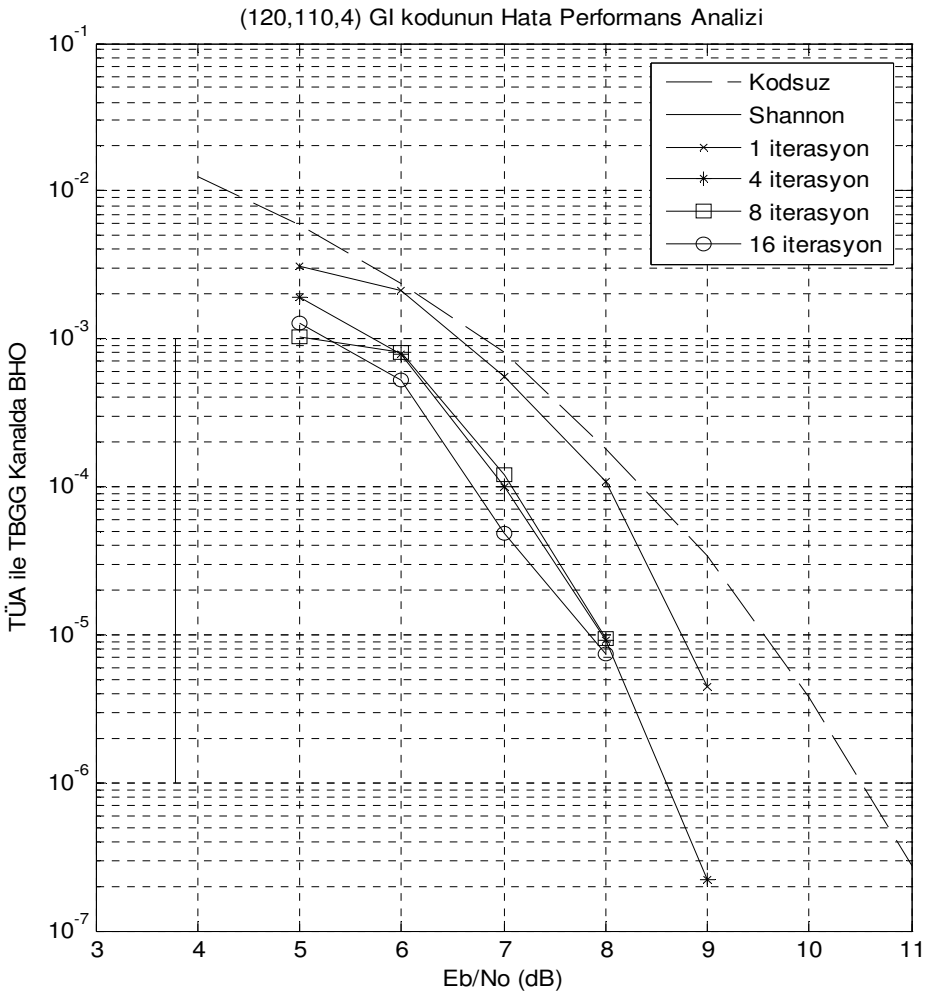
4.5.4.2 İkinin Katları Şeklinde Olan Gİ Kodlarının Performans Analizi



İter-Eb/N0	4	5	6	7	8	9	10	11
Kodsuz	1.2593E-02	5.8520E-03	2.4010E-03	8.1700E-04	1.8100E-04	3.4000E-05	3.8000E-06	2.7000E-07
1			1.8585E-03	4.5132E-04	9.5849E-05	5.6604E-06		
4		1.8239E-03	5.8239E-04	7.7358E-05	5.0314E-06			
8		2.8050E-03	4.8050E-04	6.3522E-05	6.6038E-06			
16		3.6981E-03	3.6981E-04	5.0943E-05	1.8868E-06			
64		1.5943E-03	3.5943E-04	4.4811E-05	4.2453E-06			

Şekil 4.27 C = (60,53, 4) Gİ kodunun TBGG kanalda toplam ürün kodçözme algoritması yardımıyla elde edilen BHO

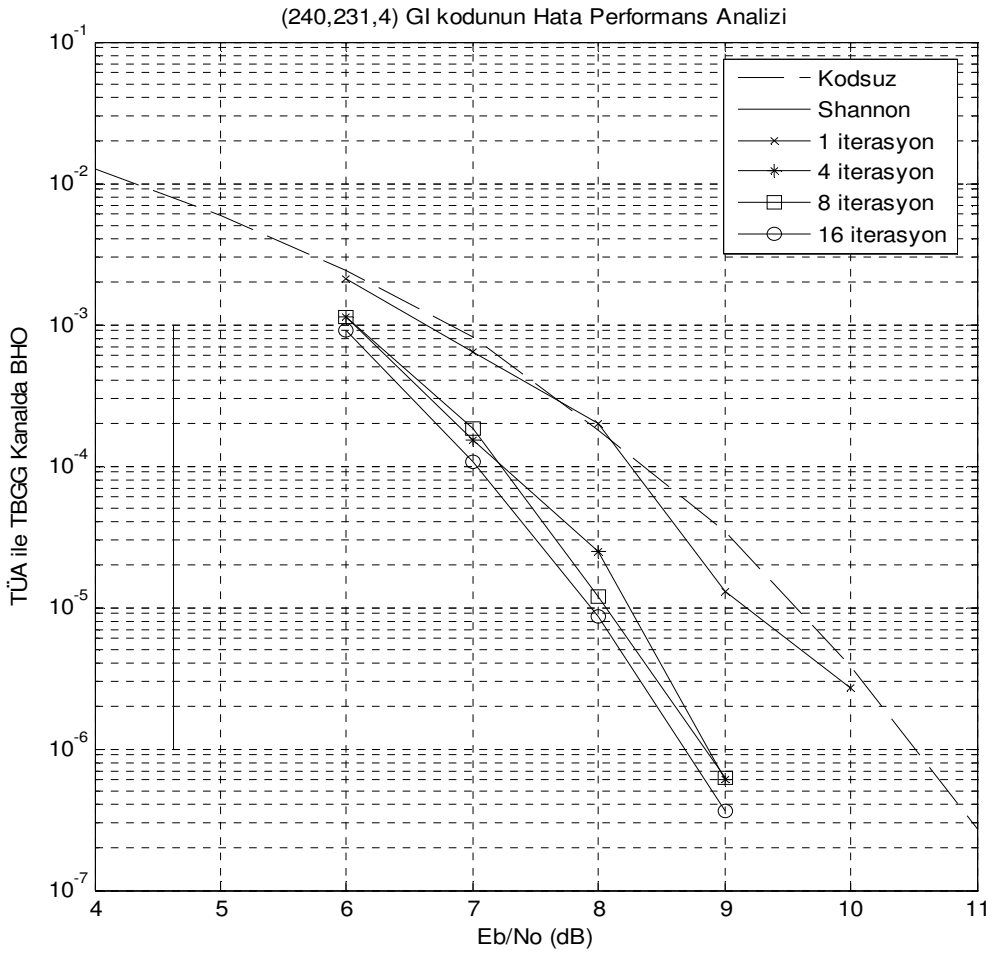
Şekil 4.27'de görüldüğü gibi, (60, 53, 4) kodlama kazancı $R=k/n$=53/60= 0.8833 olan Gİ kodu kullanılarak 16 iterasyonla elde edilen kodlama kazancı, 10^{-5} BHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 2.1 dB'dir. 16 iterasyondan sonra SNR'ın 8dB değerine ulaşmasına kadar performansta kayda değer bir gelişme olmadığı görülmüştür. Bu arada, Shannon sınırı (3.04 dB) kodun BHO eğrisine daha yakındır.



İter-Eb/N0	4	5	6	7	8	9	10	11
Kodsuz	1.2593E-02	5.8520E-03	2.4010E-03	8.1700E-04	1.8100E-04	3.4000E-05	3.8000E-06	2.7000E-07
1		3.1344E-03	2.1344E-03	5.5000E-04	1.0908E-04	4.4643E-06		
4		1.9321E-03	7.9321E-04	1.0022E-04	9.1518E-06	2.2321E-07		
8		2.0313E-03	8.0313E-04	1.1942E-04	9.4643E-06			
16		2.2714E-03	5.2714E-04	4.8333E-05	7.3667E-06			

Şekil 4.28 C = (120,110, 4) Gİ kodunun TBGG kanalda toplam ürün kodçözme algoritması yardımıyla elde edilen BHO

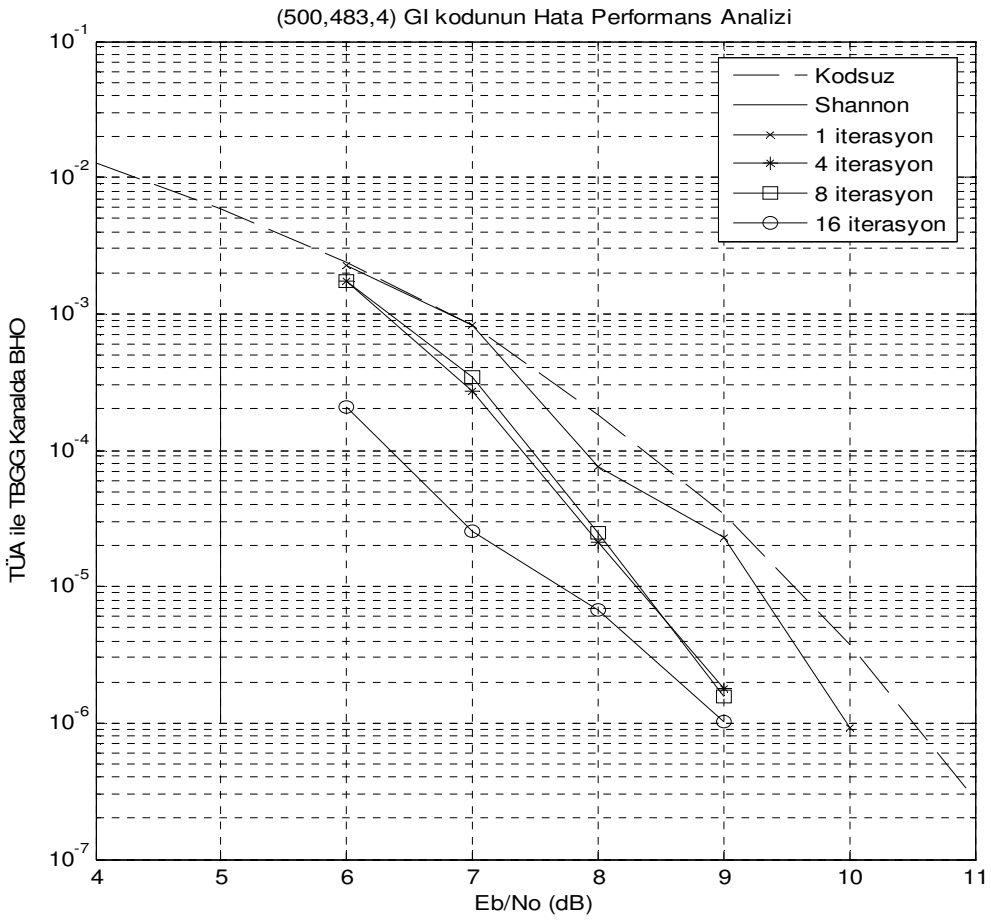
Şekil 4.28'de görüldüğü gibi, (120, 110, 4) kodlama kazancı $R=k/n$=110/120= 0.916 olan Gİ kodu kullanılarak 16 iterasyonla elde edilen kodlama kazancı, 10^{-5} BHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 1.7 dB'dir. Bu arada, Shannon sınırı (3.79 dB) kodun BHO eğrisine daha yakındır.



İter-Eb/N0	4	5	6	7	8	9	10	11
Kodsuz	1.2593E-02	5.8520E-03	2.4010E-03	8.1700E-04	1.8100E-04	3.4000E-05	3.8000E-06	2.7000E-07
1			2.1292E-03	6.3276E-04		1.2989E-05	2.6800E-06	
4			1.1236E-03	1.5325E-04	2.5000E-05	6.0606E-07		
8			1.1201E-03	1.8593E-04	1.1797E-05	6.2747E-07		
16			9.1439E-04	1.0682E-04	8.6580E-06	3.6580E-07		

Şekil 4.29 C = (240,231, 4) Gİ kodunun TBGG kanalda toplam ürün kodçözme algoritması yardımıyla elde edilen BHO

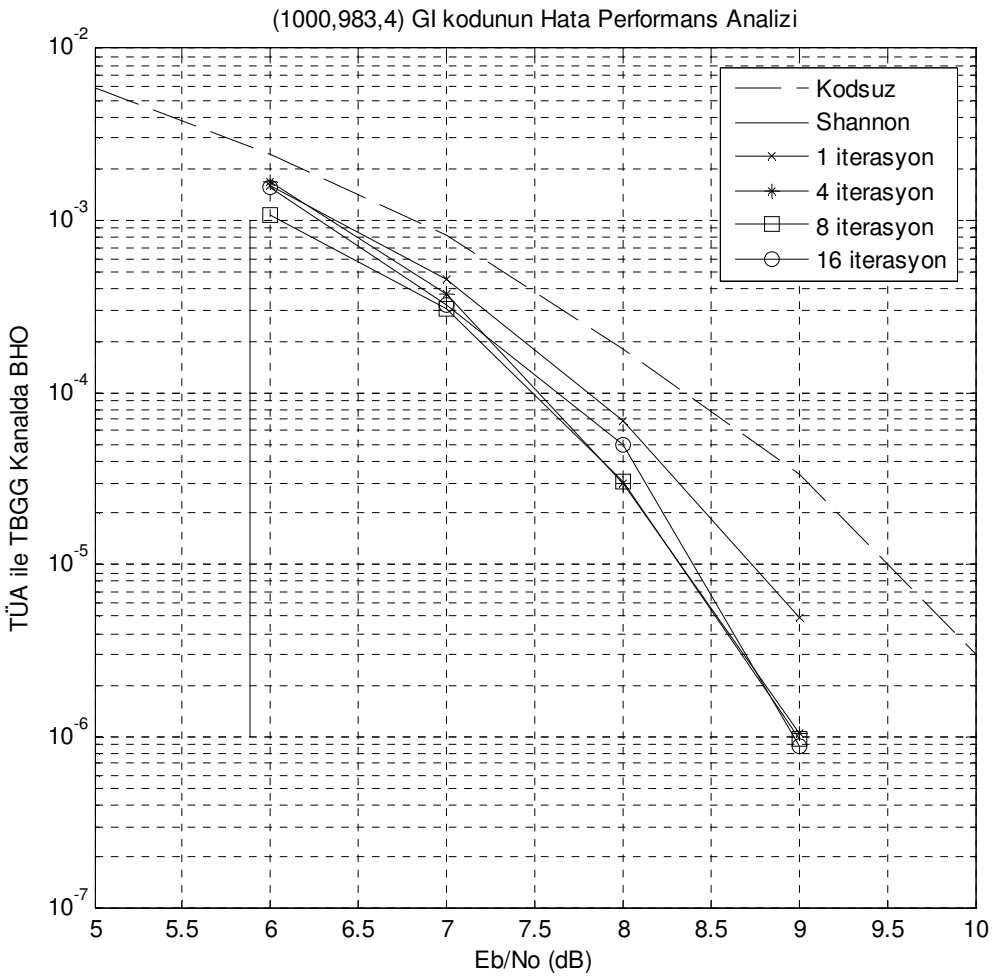
Şekil 4.29'da görüldüğü gibi, (240, 231, 4) kodlama kazancı $R=k/n$=231/240= 0.962 olan Gİ kodu kullanılarak 16 iterasyonla elde edilen kodlama kazancı, 10^{-5} BHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 1.6 dB'dir. 8 iterasyondan sonra SNR'ın 9dB değerine ulaşmasına kadar performansta kayda değer bir gelişme olmadığı görülmüştür. Bu arada, Shannon sınırı (4.62 dB) kodun BHO eğrisine daha yakındır.



İter-Eb/N0	4	5	6	7	8	9	10	11
Kodsuz	1.2593E-02	5.8520E-03	2.4010E-03	8.1700E-04	1.8100E-04	3.4000E-05	3.8000E-06	2.7000E-07
1			2.2539E-03	8.3633E-04	7.6016E-05	2.2694E-05	9.1837E-07	
4			1.7524E-03	2.6830E-04	2.1173E-05	1.7857E-06		
8			1.7463E-03	2.4592E-05	3.4592E-04	1.5600E-06		
16			2.0768E-04	2.5762E-05	6.7682E-06	1.0327E-06		

Şekil 4.30 C = (500,483, 4) Gİ kodunun TBGG kanalda toplam ürün kodçözme algoritması yardımıyla elde edilen BHO

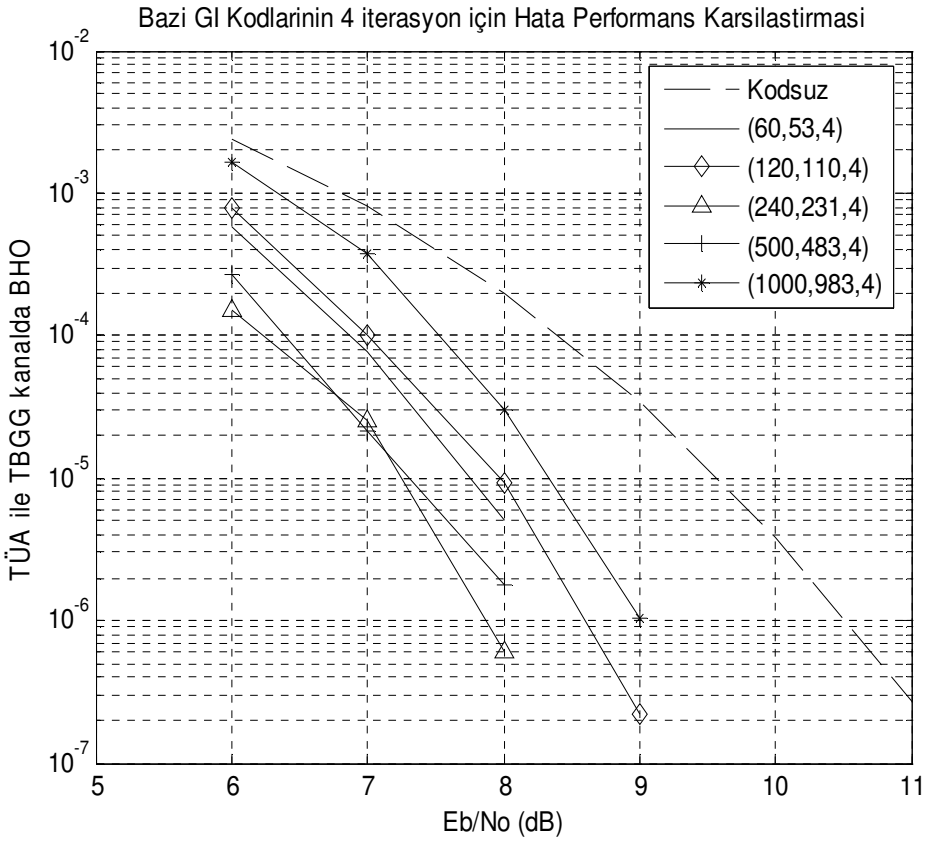
Şekil 4.30'da görüldüğü gibi, (500, 483, 4) kodlama kazancı $R=k/n$=483/500= 0.9666 olan Gİ kodu kullanılarak 16 iterasyonla elde edilen kodlama kazancı, 10^{-5} BHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 1.8 dB'dir. Bu arada, Shannon sınırı (5 dB) kodun BHO eğrisine daha yakındır.



İter-Eb/N0	4	5	6	7	8	9	10
Kodsuz	1.2593E-02	5.8520E-03	2.4010E-03	8.1700E-04	1.8100E-04	3.4000E-05	3.8000E-06
1			1.6152E-03	7.6152E-04	6.9137E-05	4.9240E-06	
4			1.7657E-03	3.7657E-04	3.0100E-05	1.0400E-06	
8			1.0766E-03	3.0766E-04	3.0153E-05	9.8056E-07	
16			1.5600E-03	3.2560E-04	5.0105E-05	8.7600E-07	

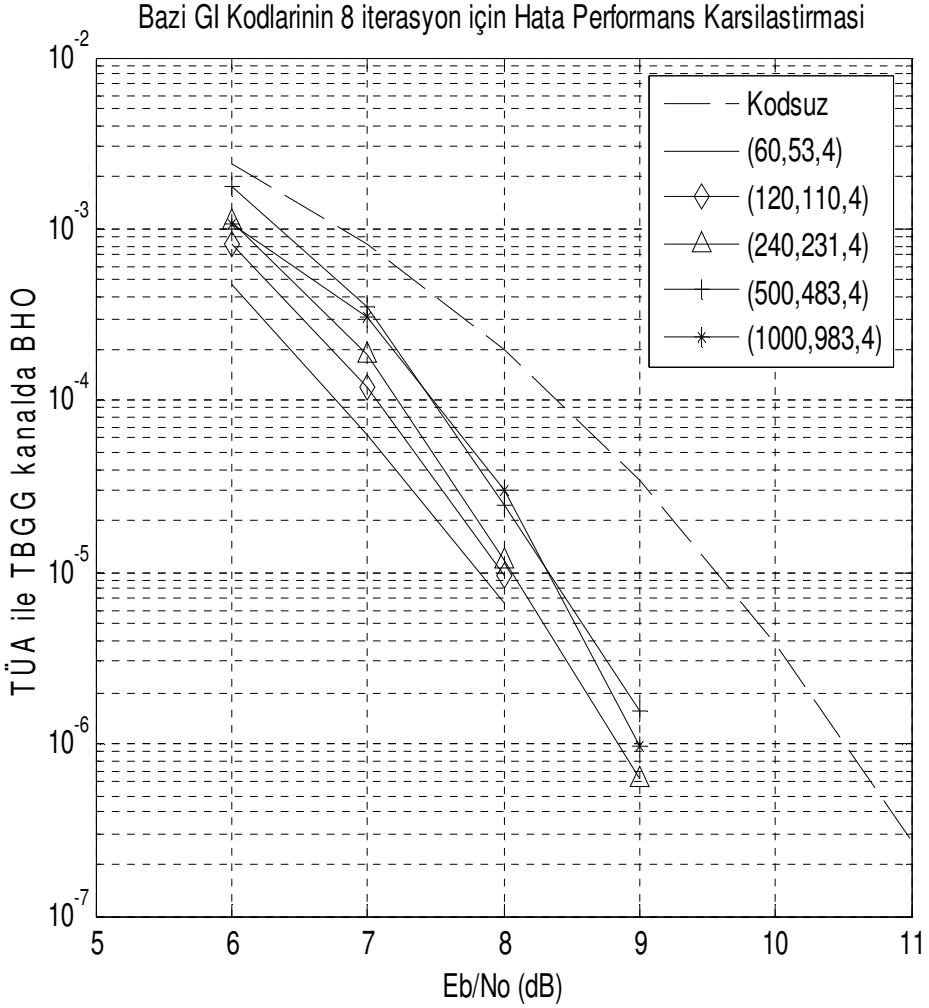
Şekil 4.31 C = (1000,983, 4) Gİ kodunun TBGG kanalda toplam ürün kodçözme algoritması yardımıyla elde edilen BHO

Şekil 4.31'de görüldüğü gibi, (1000, 983, 4) kodlama kazancı $R=k/n$=983/1000= 0.983 olan Gİ kodu kullanılarak 8 iterasyonla elde edilen kodlama kazancı, 10^{-5} BHO oranında kodsuz durumla karşılaştırıldığında yaklaşık 1.1 dB'dir. Bu da Gİ kodunun en iyi başarım değerine hızlı bir şekilde ulaştığını göstermektedir. Bu arada, Shannon sınırı (5.88 dB) kodun BHO eğrisine daha yakındır.



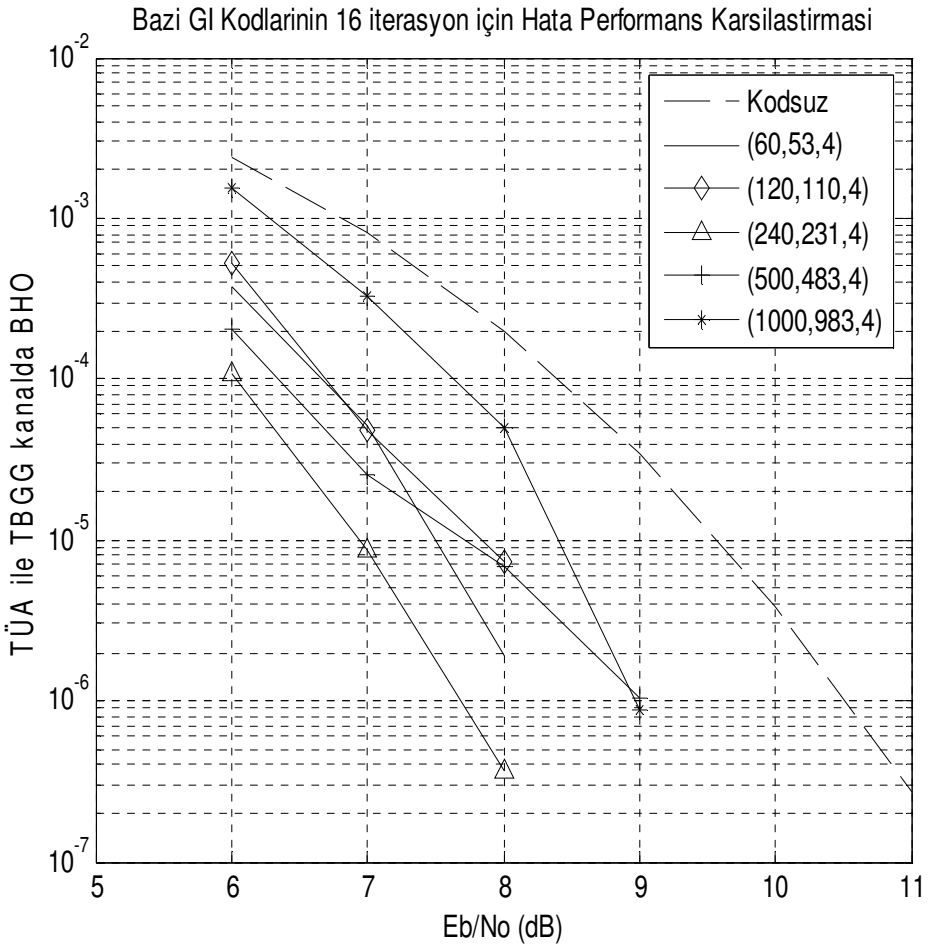
Kod-Eb/N0	6	7	8	9
(60,53,4)	5.8239E-04	7.7358E-05	5.0314E-06	
(120,110,4)	7.9321E-04	1.0022E-04	9.1518E-06	2.2321E-07
(240,231,4)	1.1236E-03	1.5325E-04	2.5000E-05	6.0606E-07
(500,483,4)	1.7524E-03	2.6830E-04	2.1173E-05	1.7857E-06
(1000,981,4)	1.7657E-03	3.7657E-04	3.0100E-05	1.0400E-06

Şekil 4.32 4 iterasyonla bazı Gİ kodlarının BHO Karşılaştırması



Kod-Eb/N0	6	7	8	9
(60,53,4)	4.8050E-04	6.3522E-05	6.6038E-06	
(120,110,4)	8.0313E-04	1.1942E-04	9.4643E-06	
(240,231,4)	1.1201E-03	1.8593E-04	1.1797E-05	6.2747E-07
(500,483,4)	1.7463E-03	2.4592E-05	3.4592E-04	1.5600E-06
(1000,981,4)	1.0766E-03	3.0766E-04	3.0153E-05	9.8056E-07

Şekil 4.33 8 iterasyonla bazı Gİ kodlarının BHO Karşılaştırması

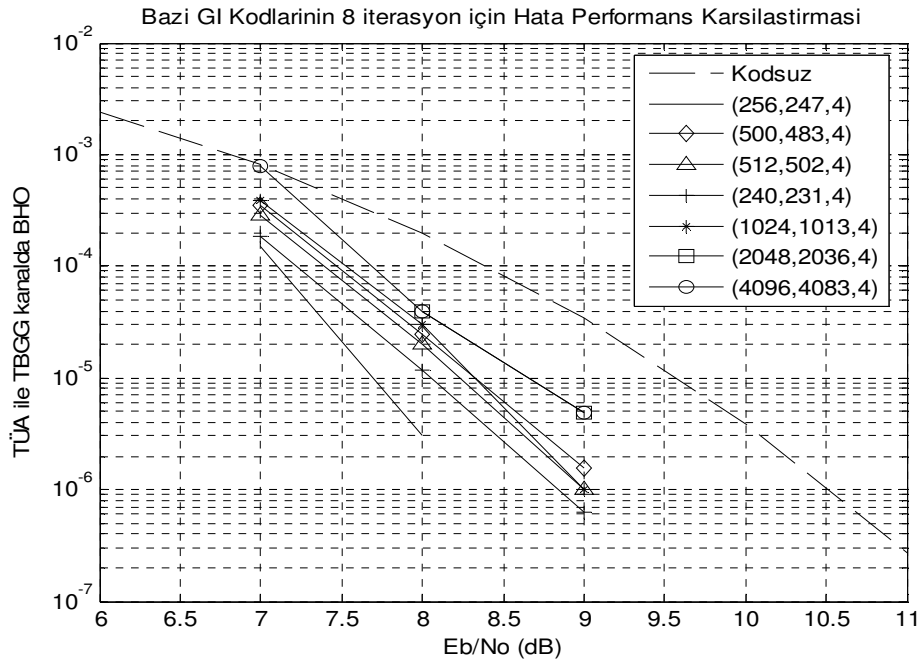


Kod-Eb/N0	6	7	8	9
(60,53,4)	3.6981E-04	5.0943E-05	1.8868E-06	
(120,110,4)	5.2714E-04	4.8333E-05	7.3667E-06	
(240,231,4)	9.1439E-04	1.0682E-04	8.6580E-06	3.6580E-07
(500,483,4)	2.0768E-04	2.5762E-05	6.7682E-06	1.0327E-06
(1000,981,4)	1.5600E-03	3.2560E-04	5.0105E-05	8.7600E-07

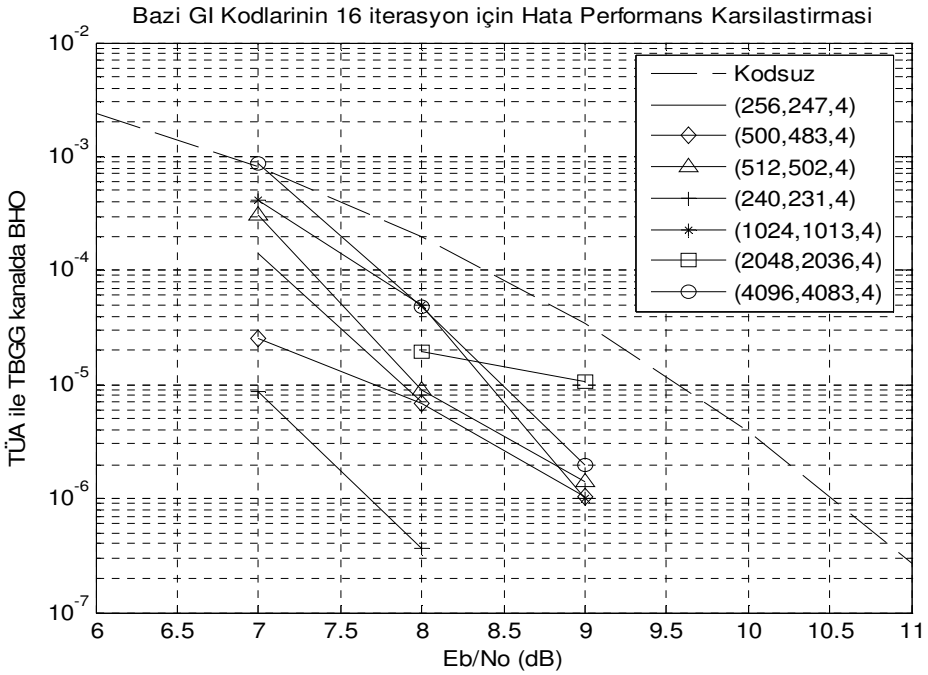
Şekil 4.34 16 iterasyonla bazı Gİ kodlarının BHO Karşılaştırması

Kodlar arasındaki bazı karşılaştırmalar 4, 8, 16 iterasyonlar için sırasıyla Şekil 4.32, Şekil 4.33 ve Şekil 4.34'de verilmiştir. 10^{-5} BHO'nı göz önüne aldığımızda 4 ve 16 iterasyon için her iki şeklin de başarı açısından aynı dereceye sahip olduğunu görürüz. İki şekilde de görüldüğü gibi, (240, 231, 4) Gİ kodu en iyi performansa sahipken en kötü performansa (1000, 983, 4) kodu sahiptir. 8 iterasyon için (60, 53, 4) kodu en iyi performansa sahiptir. En kötü performans yine (1000, 983, 4) koduna aittir. İterasyon sayısının arttırılmasının olumlu etkileri bu şekillerden daha iyi anlaşılmaktadır. Eğer, bu kodları 10^{-5} BHO'sunda Shannon sınırından uzaklığına göre sıralarsak, bu kez sıralamada birinci 2.42 dB ile (1000, 983, 4) Gİ kodudur. Daha sonra, 2.8 dB ile (500, 483, 4), 3.38 dB ile (240, 231, 4), 4.11 dB ile (120, 110, 4), 4.5 dB ile (60, 53, 4) gelmektedir. Bu sıralamaların ışığında, Gİ kod oranı azalırken, kodlama kazancı artmakta ve kod oranı artarken performans Shannon sınırına yaklaştığı sonucuna varılmaktadır. İkinin kuvvetleri şeklinde olan kodlara göre ikinin katları olan kodlar Shannon sınırına daha uzaktırlar.

Diğer ilginç bir sonuç da iterasyon sayısı arttırıldıkça ikinin katı olan kodların daha iyi performans sergilemesidir.



Şekil 4.35 8 iterasyonla ikinin katı ve ikinin kuvvetleri şeklinde olan kodların BHO Karşılaştırması



Şekil 4.36 16 iterasyonla ikinin katı ve ikinin kuvvetleri şeklinde olan kodların BHO Karşılaştırması

4.5.5 Gİ Kodlarıyla GÜ Kodları Arasındaki Temel Farklar

1. GÜ kodları, genel bir ürün kodu üreteç matrisi G_y'nin üzerine GÜ kod inşasının belirlediği kurallara göre oluşturulan üreteç matrisi ile elde edilirken Gİ kodları, genel bir ürün kodu üreteç matrisi kullanmadan tamamen sıfırdan üreteç matrisi inşa edilerek elde edilir.

2. Gİ kodları üreteç matrislerinin GÜ kodları üreteç matrislerine göre avantajı, genel ürün kodu üreteç matrisi G_y'nin yerine konulan matrisin aynen kendisine eklenen eleman matrisleri gibi basamaklı yapıda olması yani kaydırmalı kaydedicilerle her matris satırı belli sayıda kaydırılarak bir sonraki satırı elde etme olanağına sahip olmasıdır ki bu da kodlayıcı tasarımında önemli bir avantaj sağlar.

3. Gİ kodları inşasında herhangi bir matris çarpması gerekmediğinden kod matrislerinin elde edilmesi GÜ'ye göre daha kolaydır.

4. Performans analizleri incelendiğinde Gİ kodlarında daha az iterasyonla daha yüksek kazanç elde edilebildiği görülmektedir. Bu da Gİ kodlarının istenen BHO'na hızlı bir şekilde yaklaştığını göstermektedir.

5. Gİ kodlarının performans analizinde Shannon limitine oldukça yakın değerler elde edilebilmiştir.

6. Gİ kodları için yüksek uzunluklu kodların performansının GÜ kodlarına göre daha başarılı olduğu görülmüştür.

5. SONUÇLAR VE TARTIŞMA

Bu çalışmada yeni bir kanal kodlama yöntemi olan Geometrik İnşa (Gİ) kodlamasının, 16, 32, 64, 128, 256, 512 gibi büyük uzunluklu kodlar için genel bir algoritması sunulmuş ve kod matrisleri sistematik hale getirilerek kodların performans analizi yapılmıştır. Geometrik Ürün kodları genel yapıda olup uzunluğu 8'den büyük ve Hamming mesafesi 4 olan bütün en iyi çift ikili blok kodları, bir kod ailesi olarak, üretebilmektedir. Bu kodların minimum Hamming mesafesinin 4 olduğu Teorem 2.1 ile önerilerek ispat edilmiştir. Ayrıca Hamming mesafesi 8, 16 ve daha büyük olan bazı en iyi ve iyi kodları da üretmektedir.

Geometrik İnşa (Gİ) kodları, GÜ kod inşasından daha kolay yapıda fakat GÜ kodları ile aynı ölçülerde olacak şekilde elde edilmiştir. Gİ kodlarının GÜ kodlarından esas farkı; Gİ kodları, GÜ kodlarında temel olarak kullanılan genel bir ürün kodu üreteç matrisini hiç kullanmadan sıfırdan yeni bir üreteç matrisi yapısına sahiptir. Gİ kodları da GÜ kodlarında olduğu gibi Hamming mesafesi 4 olan bütün en iyi çift kodları, bir kod ailesi olarak, üretebilmektedir. Bu kodların minimum Hamming mesafesinin 4 olduğu Yardımcı Teorem 2.1, Yardımcı Teorem 2.2 ve Teorem 2.4 ile önerilerek ispat edilmiştir. Gİ kodları da genel bir yapı olup minimum Hamming mesafesi daha büyük olan kodları üretebilir.

Gİ kodlarından bazıları için hata başarımları, TBGG kanal ortamında, bilgisayar benzetimi aracılığıyla toplam-ürün kodçözme algoritması kullanılarak elde edilmiştir.

Ayrıca özellikle Gİ kodları için kod inşası çok kolay, düzenli ve yarı çevrimsel bir yapıdadır. Dolayısıyla kaydırmalı kaydediciler kullanarak çok büyük uzunluktaki kodları bile çok daha kolay üretebilme olanağı sağlamaktadır. Daha da önemlisi Gİ ve GÜ kodlarının üreteç matrisleri seyrek yoğunluklu üreteç matrisi yapılarına sahip olup kodlayıcı uygulamalarında çok faydalı olan bir özelliği barındırmaktadır.

GÜ ve Gİ kodları bilgisayar içi haberleşme, saklayıcılar, optik haberleşme gibi özellikle büyük oranlı kodlar gerektiren, bant-genişliği etkin uygulamalarda kullanılabilir.

Bazı önemli Gİ kodlarının hata başarım analizleri Toplam Ürün Algoritması kullanılarak TBGG üzerinde yapıldı. Başarım sonuçları, Hamming mesafesi 4 olan Gİ kodlarda özellikle büyük ölçekli (2000 civarı) kodlarda oldukça umut vericidir. Daha yüksek mesafeli Hamming kodlarındaki Gİ kodlarının başarımının daha iyi olabilmesi için kodlama oranında geliştirmeler yapmak gerekmektedir. İteratif toplam ürün algoritmasıyla kodçözümü hemen hemen en iyi hata başarımına ulaşmaktadır (16 iterasyon). Gİ kodları bilgisayar hafıza sistemleri, bilgisayar içi haberleşmeler gibi çok yüksek kodlama kazancı gerektiren($R \geq 0.98$)

uygulamalar için iyi bir seçimdir. Hamming mesafesi 4 olan Gİ kodları, bilgi oranı çift olan bir kod ailesi olduğundan daha etkin kodlar elde etmek için kullanılabilir.

Ayrıca ikinin katı olan Gİ kodları elde edilerek performans analizleri yapılmıştır. Elde edilen sonuçlara göre, Gİ kod oranı azalırken, kodlama kazancı artmakta ve kod oranı artarken performans Shannon sınırına yaklaştığı sonucuna varılmaktadır. İkinin kuvvetleri şeklinde olan kodlara göre ikinin katları olan kodlar Shannon sınırına daha uzaktırlar.

Diğer ilginç bir sonuç da iterasyon sayısı arttırıldıkça ikinin katı olan kodların daha iyi performans sergilemesidir.

Gİ kodlarıyla GÜ kodlarının performans analizleri incelendiğinde Gİ kodlarının düşük iterasyonlarda daha hızlı sonuca ulaştığı görülmüştür. Ayrıca Gİ kodlarının BHO eğrilerinin Shannon sınırına oldukça yakın olduğu sonucuna varılmıştır.

Bu çalışmanın ileri aşamalarında Gİ kodlarının Hamming mesafesinin arttırılarak EXIT (Extrinsic Information Transfer) -Dışınsal Bilgi Nakli- (Oenning, 2001) şemalarıyla performans analizlerinin yapılması planlanmaktadır. Bunun sonucunda da Düşük Yoğunluklu Parite Kontrol kodlama ile performans açısından karşılaştırma da kolaylıkla yapılabilecektir.

KAYNAKLAR

Altay, G., (2006), "İteratif Ayrıştırılabilir Blok Kodlar", Doktora Tezi, İstanbul Üniversitesi.

Pless, V., (1989), "Introduction to the Theory of Error-Correcting Codes", 2nd edition, John Wiley and Sons, Inc., 0-471-19047-0.

Richardson, T., Urbanke, R., (2005), "Modern Coding Theory", http://lthcwww.epfl.ch/mct/index.php

Brouwer, A. E. ve Verhoeff, T., (1993), "An Updated Table of Minimum-Distance Bounds for Binary Linear Codes", IEEE Trans. Inform. Theory, vol. 39, No. 2, pp. 664-677.

Isaka, M. ve Fossorier, M., (2005), "High-rate Serially Concatenated Coding with Extended Hamming Codes", IEEE Comm. Letter, vol. 9, pp. 160-162, No.2.

Noble, B., (1969), "Applied Linear Algebra". Prentice Hall.

Brink, S.T., (2001), IEEE Trans. On Comm, vol. 49, pp. 1727-1738, No.10

Shannon, C. E., (1948), "A Mathematical Theory of Communication", Bell Sys. Tech. J., pp.379-423 (Part 1); pp. 623-656 (Part 2).

Lin, S.ve Costello, D. J., (2004), "Error Control Coding", second edition, Prentice Hall, 0-13-017973-6.

Bossert, M., (1999), "Channel Coding", Wiley, 0-471-98277.

Hamming, R. W., (1950), "Error Detecting and Error Correcting Codes", Bell Sys. Tech. J., 29:147-60.

Muller, D.E., (1954), "Applications of Boolean Algebra to SwiKKhing Circuits Design and to Error Detection", IRE Trans., EC-3: 6-12.

Reed, I. S., (1954), "A class of Multiple-Error-Correcting Codes and the Decoding Scheme", IRE Trans., IT-4: 38-49.

Elias, P., (1954), "Error-Free Coding", IRE Trans. Inform. Theory, PGIT-4: 29-37.

Macwilliams, F. J. ve Sloane N. J. A., (1977), "The Theory of Error-Correcting Codes", North Holland, Amsterdam, 0-444-85193-1.

Plotkin, M., (1960), "Binary Codes with Specific Minimum Distance", IEEE Trans. Inform. Theory, IT-34: 1152-1187.

Golay, M. J. E., (1949), "Notes on Digital Coding", Proc. IEEE, 37:657.

Wicker, S. B., (1995), "Error Control Systems for Digital Communications and Storage", Prentice Hall, Englewood Cliffs,N.J., 0132008092.

Prange, E., (1957), "Cyclic Error-Correcting Codes in two Symbols", AFCRC-TN-57,103, Air Force Cambridge Research Center, Cambridge.

Hocquenghem, A., (1959), "Codes Corecteurs d'erreurs", Chiffres, 2:147-56.

Bose, R. C. ve Ray-Chaudhuri, D. K., (1960), "On a Class of Error Correcting Binary Group Codes", Information Control, 3: 68-79.

Reed, I. S. ve Solomon, G., (1960), "Polynomial Codes over Certain Finite Fields", J. Soc. Ind. Applied Mathematics, 8: 300-304.

Berrou, C., Glavieux, A. ve Thitimajshima, P., (1993), "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes", Proc. IEEE Intl Conf. Commun. (ICC 93), pp.1064-70, Geneva, Switzerland.

Berrou, C. ve Glavieux, A., (1996), "Near Optimum Error Correcting and Decoding: Turbo codes", IEEE Trans. Commun., COM-44:1261-71.

Gallager, R. G., (1962), "Low Density Parity Check Codes", IRE Trans. Inform. Theory, IT-8: 21-28.

Gallager, R. G., (1963), "Low Density Parity Check Codes MIT Press, Cambridge.

Mackay, D. J. C. ve Neal, R. M., (1996), "Near Shannon Limit performance of Low Density Parity Check Codes, Electronics Letter, 32(18): 1645-46.

Li, J., Narayanan, K. R. ve Georghiades, C. N., (2004), "Product Accumulate Codes: A Class of Codes with Near-Capacity Performance and Low Decoding Complexity", IEEE Trans. Inform. Theory, vol.50, no.1, pp. 31-46.

Proakis, J. G., (2000), "Digital Communications", 4th ed., Mc-Graw-Hill, New York, 0072321113.

Viterbi, A. J., (1967), "Error Bounds for Convolutional Codes and Asymptoticaly Optimum Decoding Algorithm", IEEE Trans. Inform. Theory, IT-13: 260-69.

Bahl, L. R., Cocke, J., Jelinek, F. ve Raviv, J., (1974), "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate", IEEE Trans. Inform. Theory, IT-20: 284-87.

Hagenauer, J., Offer, E. ve Papke, L., (1996), "Iterative Decoding of Binary Block and Convolutional Codes", IEEE Trans. Inform. Theory, vol. 42, pp. 429-445.

Sklar, B., (1997), "A Primer on Turbo Code Concepts", IEEE Comm. Magazine, pp. 94-102.

Andersen, J. D., (1994), "The Turbo Coding Scheme", IEEE Intl. Symp. Inform. Theory (ISIT 1994), Trondheim, Norway.

Robertson, P., (1994), "Illuminating the Structure of Code and Decoder of Parallel Concatenated Recursive Systematic (Turbo) Codes", Proc. IEEE Global Commun., pp.343-46, Kingston, Ontario.

Benedetto, S. ve Montorsi, G., (1996), "Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding", IEEE Trans. Inform. Theory, IT-42: 409-428.

Benedetto, S. ve Montorsi, G., (1996), "Design of Parallel Concatenated Convolutional Codes", IEEE Trans.Commun., COM-44: 591-600.

Ryan, W. E., (2002), "Concatenated Convolutional Codes and Iterative Decoding", in Wiley Encyclopedia of Telecommunications, edited by Proakis, J. G., John Wiley, New York.

Brink, S., (1999), "Convergence of Iterative Decoding", IEE Electron. Lett., 35: 806-8.

Mackay, D., (1999), "Good Error Correcting Codes Based on Very Sparse Matrices", IEEE Trans. Inform. Theory, pp. 399-431.

Tanner, R. M., (1981), "A Recursive Approach to Low Complexity Codes", IEEE Trans. Inform. Theory, pp. 533-547.

Sun, J., (2003), "An Introduction to Low Density Parity Check (LDPC) Codes", Lane Dept. of Computer Science & Electrical Engineering, West Virginia University, USA, www.csee.wvu.edu/wcrl/public/slideldpc.pdf

Mackay, D., (1999), "Sparse Graph Codes", Proc. 5th Intl. Symp. Commun. Theory and Applications, pp. 2-4, Ambleside, UK.

Richardson, T., Shrokrollahi, A. ve Urbanke, R., (2001), "Design of Capacity Approaching Irregular Codes", IEEE Trans. Inform. Theory, 47(2): 619-37.

Richardson, T. ve Urbanke, R., (2001), "The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding", IEEE Trans. Inform. Theory, 47: 599-618.

Chung, S. Y., Forney, G. D. Jr., Richardson, T. ve Urbanke, R., (2001), "On the Design of Low Density Parity Check Codes within 0.0045 dB of the Shannon Limit", IEEE Commun, Lett., 5:58-60.

Kou, Y., Lin, S. ve Fossorier, M., (2001), "Low Density Parity Check Codes Based on Finite geometries: A rediscovery and more, IEEE Trans. Inform. Theory, 47(6): 2711-36.

Mceliece, R. J., Mackay, D. J. C. ve Cheng, J., (1998), "Turbo Decoding as an Instance of Pearl's Belief Propagation Algorithm", IEEE J. Select. Areas Commun., 16: 140-52.

Kschischang, F. R. ve Frey, B. J., (1998), "Iterative Decoding of Compound Codes by Probability Propagation in Graphical Models", IEEE J. Select. Areas Commun., 16(12): 219-30

Kschischang, F. R., Frey, B. J., ve Loeliger, H. A., (2001), "Factor Graphs and the Sum-Product Algorithm", IEEE Trans. Inform. Theory, 47: 498-519.

Oenning, T.R., Moon, J., (2001), "A Low Density Generator Matrix Interpretation of Parallel Concatenated Single Bit Parity Codes", IEEE Trans. Magn., vol. 37, pp. 737-741, No.2.

Altay, G., Uçan, O. N., Uğurdağ, H. F., (2005), "Geometric Augmented Product Codes", IEEE Proc. Communications..

Altay, G., Uçan, O. N., (2005), "Iterative Decoding of Geometric Product Codes", Istanbul University - Journal of Electrical and Electronics Engineering, Volume: 5, Number: 2.

Altay, G., Uçan, O. N., (2005), "Maximum A Posteriori Decoding of Geometric Product Codes", *Trends* in the Development of Machinery and Associated Technology (TMT 2005), pp.1181-1184, Antalya.

Altay, G., Uçan, O. N., (2005), "Iterative Decoding Performance of Geometric Augmented Product Codes Over AWGN Channel", International Conference on Electrical & Electronics Engineering, (ELECO), Bursa

Altay, G., Uçan, O. N., (2005), "EAPCC Blok Kodları İçin Genel Kafes Yapısı Tasarımı", IEEE SİU'05 Kurultayı, Kayseri.

Altay, G., Uçan, O. N., (2003), "Construction and Minimal Trellis for (32,16, 8) Decomposable Code", International Conference on Electrical & Electronics Engineering, (ELECO), Bursa.

Altay, G., Uçan, O. N., (2006), "Heuristic Construction of High -Rate Linear Block Codes", International Journal of Electronics and Communications (AEUE: Archiv fuer Elektronik und Uebertragungstechnik).

Akbaş, T., Uçan, O. N., Yücel M., (2005), "Düşük Yoğunluklu Parite Kontrol Kod Tasarımı ve Gerçeklemesi", ITUSEM, Adana.

Altay, G., Akbaş, T., Uçan, O. N., (2006), "Peformance of Geometric Construction Codes Using Sum-Product Algorithm", Journal of Computers and Electrical Engineering (sunuldu)

Williams, D., (2000), "Turbo Product Code Tutorial", IEEE 802.16 Session 7 Monday Night Tutorial Session,

ÖZGEÇMİŞ

Doğum tarihi	20.12.1974	
Doğum yeri	Silifke	
Lise	1986-1992	Erdemli Lisesi
Lisans	1992-1997	Çukurova Üniversitesi Mühendislik-Mimarlık Fak. Elektrik-Elektronik Mühendisliği Bölümü
Yüksek Lisans	1998-2001	Çukurova Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalı
Doktora	2002-2007	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik-Haberleşme Mühendisliği Anabilim Dalı, Haberleşme Programı

Çalıştığı kurum(lar)

1998-2000	Çukurova Üniversitesi Fen Bilimleri Enstitüsü Araştırma Görevlisi
2000-2005	Telsim Mobil Telekomünikasyon Hiz. A.Ş., NSS İşletme Uzman Mühendisi
2005- 2006	INTA Mühendislik A.Ş., NSS İşletme Uzman Mühendisi
2006- ...	Nokia Siemens Networks, Customer Care Specialist