DÜŞÜK YOĞUNLUKLU EŞİTLİK KONTROL KODLARININ AYRIMSAL KİPLENİM İLE SERİ BİRLEŞTİRİLMESİ

SERIAL CONCATENATION OF LOW DENSITY PARITY CHECK CODES AND DIFFERENTIAL MODULATION

ERSİN BARAN AKKUŞ

Hacettepe Üniversitesi

Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin

ELEKTRİK ve ELEKTRONİK Mühendisliği Anabilim Dalı İçin Öngördüğü

YÜKSEK LİSANS TEZİ

olarak hazırlanmıştır.

2008

Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Bu çalışma jürimiz tarafından ELEKTRİK ve ELEKTRONİK Mühendisliği ANABİLİM DALI'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Başkan	:	Prof. Dr. Mehmet ŞAFAK
Üye (Danışman)	:	Yrd. Doç. Dr. Emre AKTAŞ
Üye	:	Yrd. Doç. Dr. Mücahit ÜNER
Üye	:	Yrd. Doç. Dr. Cenk TOKER
Üye	:	Yrd. Doç. Dr. Ali Özgür YILMAZ

ONAY

Bu tez ..../..../2008 tarihinde Enstitü Yönetim Kurulu tarafından kabul edilmiştir.

..../..../2008

Prof. Dr. Erdem YAZGAN
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRÜ

DÜŞÜK YOĞUNLUKLU EŞİTLİK KONTROL KODLARININ AYRIMSAL KİPLENİM İLE SERİ BİRLEŞTİRİLMESİ

Ersin Baran AKKUŞ

ÖZ

Veri iletişiminin etkin ve güvenli bir şekilde gerçekleştirilmesi haberleşme dünyasında her zaman önemli olan bir konudur. Etkin ve güvenli bir veri iletişiminin oluşturulması, verilerin kanaldaki hataya karşı vericide kodlanması ve alıcıda çözümlenmesi ile mümkündür. Son on beş yılda, döngülü çözümleyiciler ile kodların kullanılmasının daha önce görülmeyen çok düşük hata oranlarını ortaya çıkardığı gözlemlenmiştir. Düşük yoğunluklu eşitlik kontrol (LDPC) kodları bu tip döngülü alıcıya sahip kodlardan biridir.

Kodlanmış veriler kanal üzerinden alıcıya gönderilmek için kiplenir. Bu kiplenim tekniklerinden biri de ayrımsal kiplenim tekniğidir. Ayrımsal kiplenim tekniğinin faz belirsiz kanallar üzerinde kullanışlı olduğu bilinmektedir.

Bu tezde, LDPC kodlaması ile ayrımsal kiplenimin birleştirilmesi çalışılmıştır. Vericide kodlama ve kiplenim, alıcıda kipçözme ve çözümleme incelenmiş ve detayları ile sunulmuştur. Simülasyonlarla bu sistemin bit hata oranı incelenmiştir. Performansı geliştirmek için Monte Carlo tekniği kullanılarak, LDPC derece dağılımlarının eniyilemesi için EXIT grafiği eniyileme tekniği kullanılmış ve performansı geliştiren bu EXIT grafikleri tez içerisinde sergilenmiştir.

Anahtar Kelimeler: Çarpan Çizgesi, Döngülü Çözümleyiciler, LDPC, BCJR, Ayrımsal Kiplenim, EXIT Grafiği, Birleştirilmiş Kodlar

Danışman: Yrd.Doç.Dr. Emre AKTAŞ, Hacettepe Üniversitesi, Elektrik-Elektronik Mühendisliği Bölümü

SERIAL CONCATENATION OF LOW DENSITY PARITY CHECK CODES AND DIFFERENTIAL MODULATION

Ersin Baran AKKUŞ

ABSTRACT

Efficient and reliable data transmission is an ever important task of communication systems. Efficient and reliable data transmission is feasible with channel coding where data is encoded at the transmitter against errors in the channel, and decoded at the receiver. It has been observed in the last fifteen years that using codes with iterative decoders can result in very low error rates that have not been seen before. Low density parity check (LDPC) codes are one of the codes with iterative receivers.

Coded data have to be modulated in order to be sent to the receiver over the channel. Differential encoding is one of these modulation techniques. It is known that differential encoding can be useful over phase uncertain channels.

In this thesis, concatenation of LDPC and differential encoding is studied. Coding and modulation at the transmitter, demodulation and decoding at the receiver are investigated and presented in detail. The bit error rate of this system is investigated via simulations. In order to improve performance, EXIT chart optimization technique is applied to optimize the degree distributions of LDPC, using Monte Carlo techniques. EXIT charts that the performance improved using them, are presented.

Keywords: Factor Graphs, Iterative Decoders, LDPC, BCJR, Differential Modulation, EXIT Charts, Concatenated Codes

Advisor: Asst.Prof.Dr. Emre AKTAŞ, Hacettepe University, Department of Electrical and Electronics Engineering

TEŞEKKÜR

Bu tezin gerçekleştirilmesi sırasında bana yön gösteren ve benden ilgisini ve desteğini hiçbir zaman esirgemeyen danışmanım Yrd.Doç.Dr. Emre AKTAŞ'a teşekkür ederim.

Bana güvenen ve desteğini bir an olsun esirgemeyen anneme ve babama teşekkür ederim.

Bu tezin yazılması sırasında bana gösterdikleri anlayış ve yardımları için iş ve okul arkadaşlarıma teşekkür ederim.

Bu tezin yazılması için bilgi birikimimi oluşturmamı sağlayan tüm hocalarıma teşekkür ederim.

İÇİNDEKİLER DİZİNİ

Sayfa

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

APP : A Posteriori Probability (Sonsal Olasılık)

AWGN : Additive White Gaussian Noise (Toplanabilir Beyaz Gaussian Gürültü)

BCJR : Bahl,Cocke,Jelinek and Reviv

BER : Bit Error Rate (Bit Hata Oranı)

BPSK : Binary Phase Shift Keying (İkili Faz Kaydırmalı Anahtarlama)

DE-PSK : Differentially Encoded-Phase Shift Keying (Ayrımsal Kodlanmış Faz Kaydırmalı Anahtarlama)

DPSK : Differential Phase Shift Keying (Ayrımsal Faz Kaydırmalı Anahtarlama)

EXIT : Extrinsic Information Transfer (Dışsal Bilgi Transferi)

LDPC : Low Density Parity Check (Düşük Yoğunluklu Eşitlik Kontrol)

LLR : Log-Likelihood Ratio (Log-Olabilirlik Oranı)

NRZ : Non-Return-to-Zero

QPSK : Quadrature Phase Shift Keying (Dörtlü Faz Kaydırmalı Anahtarlama)

SISO : Soft Input Soft Output (Yumuşak Giriş Yumuşak Çıkış)

SNR : Signal to Noise Ratio (İşaret Gürültü Oranı)

SPA : Sum-Product Algorithm (Topla-Çarp Algoritması)

YGYÇ : Yumuşak Giriş Yumuşak Çıkış

SÖZLÜK DİZİNİ

Ayrımsal	: Differential
Bilgi	: Information
Çarpan Çizgesi	: Factor Graph
Çerçeve	: Frame
Çözücü	: Decoder
Çözümleme	: Decoding
Değişken Düğümü	: Variable Node
Değişken Uzunluk Kodları	: Variable Length Codes
Dikgen	: Orthogonal
Doğrusal	: Linear
Döngü	: Iteration
Durum Geçiş Matrisi	: State Transition Matrix
Eniyileme	: Optimization
Eşitlik-Kontrol	: Parity Check
Eşleyici	: Mapper
Evrişimli	: Convolutional
Gerçel	: Real
İkili	: Binary
İkili Grafik	: Bipartite Graph
İkili Kodlama	: Binary Coding
İleri Geri Algoritması	: Forward Backward Algorithm
İletim	: Transmission
İlinti	: Correlation
İnanç Yayılması	: Belief Propagation
İşaret Yıldıztakımı	: Signal Constellation
Kafes	: Trellis
Karıştırıcı	: Interleaver
Kesikli	: Discrete
Kipçözme	: Demodulation
Kipçözücü	: Demodulator
Kiplemek	: Modulate

Kiplenim	: Modulation
Kipleyici	: Modulator
Kod Kelimesi	: Codeword
Kodlayıcı	: Encoder
Karmaşık	: Compleks
Kontrol Düğümü	: Check Node
Olabilirlik	: Likelihood
Oluşturma Matrisi	: Generator Matrix
Önsel	: A Priori
Ortak	: Mutual
Sanal	: Imaginary
Seri Birleştirilmiş	: Serial Concatenated
Sonsal	: A Posteriori
Tekdüze	: Uniform
Topla Çarp Algoritması	: Sum-Product Algorithm
Uyumlu	: Coherent
Yazmaç	: Register
Yoğunluk Evrimi	: Density Evolution
Yumuşak Karar	: Soft Decision

1. SAYISAL HABERLEŞMEDE KODLAMA PRENSİPLERİ

1.1 HATA KONTROLÜ

Hata kontrol kodlaması veya kanal kodlaması, bilginin kaynaktan hedefe ulaştırılırken en düşük seviyede hata ile gönderilmesi için kullanılan metodlara verilen isimdir. Kodlama teorisi, bilgi teorisinin bir dalı olarak görülebilir ve kaynağı Shannon'ın 1940'ların sonunda yaptığı çalışmalara dayanmaktadır. Önceleri yapılan teorik çalışmalar, hata kodlamasının genel prensiplerini belirtmekteydi ve pratik yapılan çalışmalar çağın getirdiği koşullardan dolayı kısıtlanmaktaydı ve çoğunlukla teorikte görülmesi gereken sonuçlara pratikte ulaşılamamaktaydı.

Her kanal, C ile gösterilen kanal kapasitesi ile ilişkilendirilir. Öyle hata kontrol kodları vardır ki bilgi, kanaldan C'den daha küçük bir oranla, istenildiği kadar düşük bit hata oranları ile gönderilebilir [23].

Shannon bu teoremde, düşük hata oranları elde etmek için kullanılabilecek kodların varlığından söz etmektedir. İdealde sıfır hata elde edilmek istenmesine rağmen pratikte bu mümkün değildir. Sonsuz bit uzunluğunda kod kelimeleri kullanılırsa, kanal kapasitesine eşit bir oranda veri gönderilmesi sonucunda hata ancak sıfır yapılabilir. Bu noktada hata kontrol kodları devreye girmekte ve hata oranları sıfıra yaklaşırken, bilgi iletim hızının Shannon limitine yaklaşmasını sağlamaktadır.

Shannon'ın çalışmaları, herhangi bir haberleşme kanalının, kapasitesi ile karakterize edilebileceğini göstermektedir. Kapasite, ne kadar bilginin güvenilir bir şekilde gönderileceğini tanımlar. Kanal kapasitesine kadar oranlarda yapılan bilgi transferlerinde, istenilen seviyedeki hata oranlarında bilgi transfer etmek mümkündür. Hata kontrolü iletimde fazlalıkların eklenmesi ile sağlanmaktadır. Bu, alıcının mesajı çözmesi için gerekli olan sembolden daha fazla sembolün verici tarafından gönderilmesi anlamına gelmektedir. Yeterli seviyede hata kontrolü sağlandığında, kodun uzunluğu arttırılarak hata oranları daha düşük yapılabilir [24].

1.2 SAYISAL HABERLEŞME SİSTEMLERİNİN ELEMANLARI

Kodlamayı da içeren tipik bir haberleşme sistemi Şekil 1.1'de gösterilmiştir [24]. Hata kontrol kodlaması, kaynak bilgisi sayısal formata dönüştürüldükten sonra uygulanır. Kodlama ile kiplenim arasındaki farklılık bu sistemde açık bir şekilde gözükmektedir. Buna rağmen, ileriki bölümlerde de anlatılacağı üzere kodlama ve

kiplenim beraber dizayn edilebilir. Alıcı tarafında, işlemler vericiye göre ters sırada ilerler.

1.3 KAYNAK KODLAMASI

Kaynak kodlaması yapılırken, seçilmek için M mesaj varsa ve m'inci mesajın olasılığı p_m olarak ifade edilirse, bir mesajdaki ortalama bilginin büyüklüğü şu şekilde hesaplanır:

$$H = \sum_{m=0}^{M-1} p_m log_2(1/p_m) \tag{1.1}$$

Bu formül $\sum_{m=0}^{M-1} p_m = 1$ kısıtına bağlıdır.

Eğer mesajlar aynı olasılıklıysa, yani $p_m = 1/M$ ise, gönderilen ortalama bilgi $log_2(M)$'e eşittir. Bu sayı mesajların gösterimi için kullanılması gereken bit sayısına eşittir. Örnek olarak, 256 mesaj 8-bitlik kodla gösterilir ve bu mesajların gönderilme olasılıkları aynı ise herhangi bir mesajın bilgisi 8 bite eşittir.

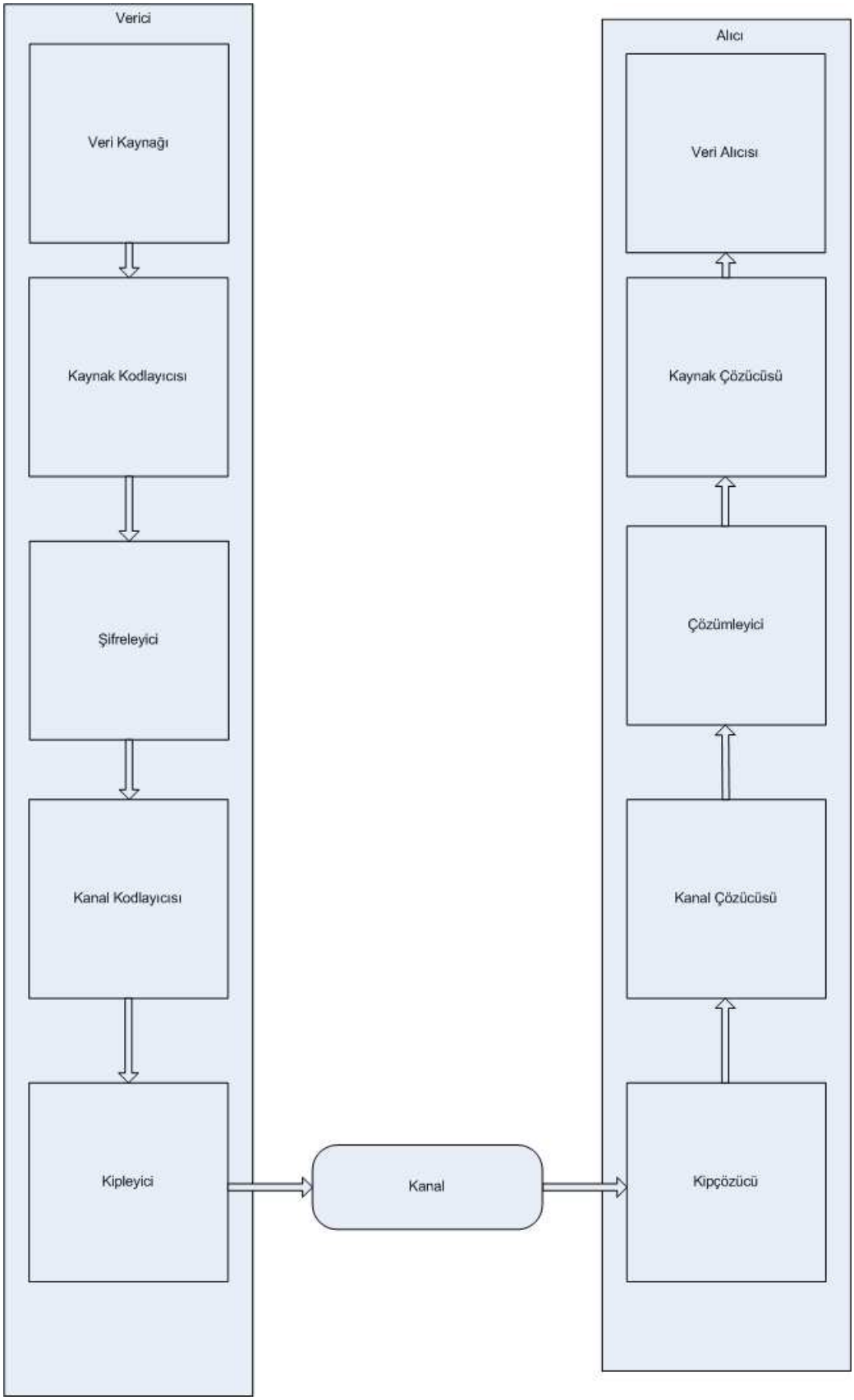
Gönderilecek olan mesajların olasılıkları birbirine eşit değilse, mesajın ortalama bilgisi $log_2(M)$ bitten daha düşük olacaktır. Bu durumda sayısal gösterim için daha az bit kullanılması istenir. Bu durum değişken uzunluk kodları ile yapılabilir ve bu tip kodlara Huffman kodları örnek verilebilir. Bu kodlarda gönderilecek dizinin uzunluğu mesajın bilgisi ile uyumludur.

Bu tezde kaynak kodlaması ile ilgilenilmemiş, kaynak kodlamasının uygulanmış olduğu bir verinin üzerinde işlem yapıldığı varsayılmıştır.

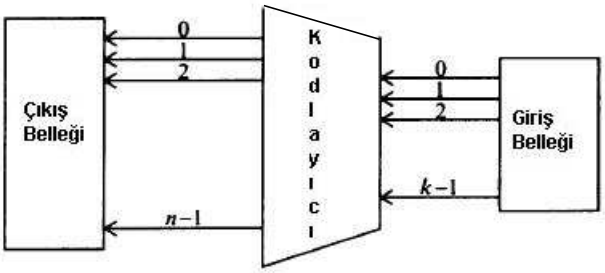
1.4 HATA KONTROL KODLAMASI

Hata kontrol kodlaması, gönderilen dizilere fazlalıklar ekler. Hata kontrol kodlaması sonucunda gönderilen bitlerin sayısı, o bitlere ait bilginin gerçek gösterimi için gerekli olan bit sayısından fazladır. Bu özelliği olmadan kodlar, hata olduğunu bulamaz ve böylelikle herhangi bir hata kontrol özelliğine sahip olamaz. Pratikte, kaynak bilgisi sıkıştırıldıktan sonra hata kontrolü için en iyi etkiyi almak için kaynak kodlamasının hemen ardından fazlalıkların eklenmesi gerekmektedir [23].

Bir kodlayıcı örneği Şekil 1.2'de gösterilmiştir. Bilgi çerçeveler şeklinde kodlayıcıya giriş yapar ve her bir çerçeve belli sayıdaki sembolden oluşmaktadır. Çoğu durumda kodlayıcının girişindeki semboller bitlerdir.



Şekil 1.1: Kodlama için sayısal haberleşme modeli



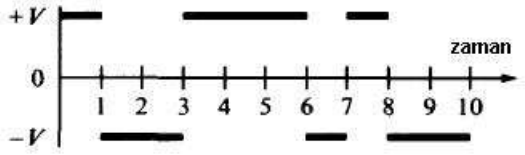
Şekil 1.2: Kodlayıcı örneği

Kodlayıcının çıkışında girişinden daha fazla sembol bulunur, yani fazlalıklar eklenmiştir. Kodu tanımlayan önemli tanımlardan biri *kod oranıdır*. Kod oranı bir çerçevedeki giriş sembollerinin çıkış sembollerine oranıdır. Düşük kod oranı yüksek dereceli fazlalık belirtir ve yüksek kod oranlarına göre daha etkili hata kontrolü sağlar.

Eğer kodlayıcı çıkış üretmek için o anki çerçeveyi kullanıyorsa, bu kod (n, k) blok kodu olarak adlandırılır ve giriş sembollerinin sayısı k ve çıkış sembollerinin sayısı da n olarak ifade edilir.

1.5 KİPLENİM

Kipleyici, bir çeşit sayısal-analog çevirici olarak düşünülebilir. Kipleyici, sayısal kod dizilerini gerçek (analog) dünyaya hazırlar. Analog sinyale en basit örneklerden biri NRZ formatıdır. Bu formatta bitler, Şekil 1.3'te görüldüğü gibi, $+V$ veya $-V$ sinyal seviyeleri ile gösterilirler.



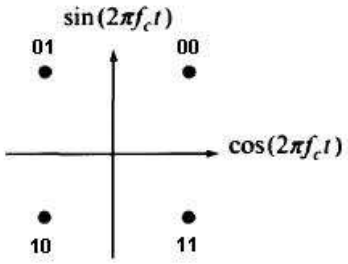
Şekil 1.3: NRZ formatı ([11]'den alınmıştır.)

Bu sinyaller, bu şekilde gönderilebileceği gibi daha yüksek frekans aralıklarına da çevrilebilir. Bunun yapılmasının sebepleri arasında spektrumun gönderim için farklı kısımlarının kullanılması ve yüksek frekanslarda daha küçük dalga boylarına sahip olunması ve daha küçük antenlerin kullanılması sayılabilir. Çoğunlukla NRZ formatındaki sinyalin kiplenimi, bu sinyallerin sinüs taşıyıcısıyla çarpılması ile gerçekleştirilir.

Sonuçta, gönderilen sinyal ya sinüs taşıyıcısı ya da onun tersidir. Bu İkili Faz Kaydırmalı Anahtarlama (BPSK) olarak bilinir.

İkinci bir taşıyıcı (ilkine göre faz farkı 90° olan) kullanılarak veriler kiplenebilir ve bu sinyal ilk taşıyıcı ile toplanarak kullanılabilir. Başka bir deyişle, BPSK sinyali, f_c taşıyıcı frekansı ve t zamanı belirtmek üzere, $\pm\cos 2\pi f_c t$ ise, ikinci sinyal $\pm\sin 2\pi f_c t$ olur. Bu yöntem Dörtlü Faz Kaydırmalı Anahtarlama (QPSK) olarak bilinir ve BPSK'e göre avantajları vardır. Bu avantajlar, aynı zamanda iki katı bit gönderilmesi ve aynı bant genişliğini kullanmasıdır.

QPSK'e ait faz diyagramı Şekil 1.4'te gösterilmiştir [24]. Sembollerin fazların üzerine eşlenmesi ikili kodlama ile yapılmıştır. İkili kodlama dışında, her bir sembol arasında sadece bir bitin değiştiği Gray kodlaması da kullanılabilir.



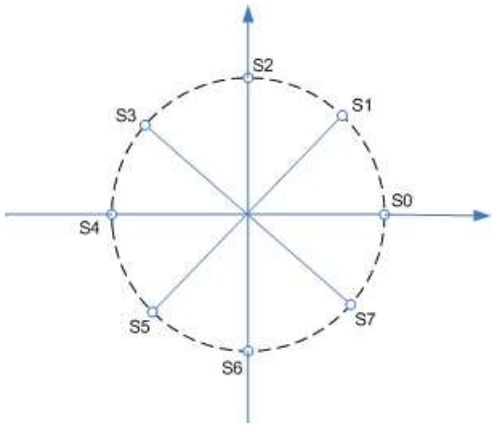
Şekil 1.4: QPSK kiplenimi

Yukarıda sayılan kiplenim teknikleri dışında Frekans Kaydırmalı Anahtarlama da kiplenim teknikleri arasında sayılabilir. Bunun dışında bant genişliğinin kısıtlı olduğu durumlarda, çoklu seviye kiplenim teknikleri kullanılabilir. Bu kiplenim teknikleri, BPSK veya QPSK'e göre aynı bant genişliğinde daha yüksek bit oranlarında kiplenim sağlar.

Bu tez çalışmasında, PSK kiplenim tekniği kullanılmıştır. Kullanılan PSK, sekiz sembol üzerinde tanımlı olan 8-PSK'dir. 8-PSK kipleniminde, QPSK'de olduğu gibi, faz farkları 90° olan iki taşıyıcı kullanılır. Sekiz sembol için işaret yıldıztakımı gösterimi Şekil 1.5'te gösterilmiştir.

1.6 KANAL

İletim ortamı zayıflatma, bozma, üzerine bindirme ve gürültü gibi etkilere sahiptir. Bu etkiler bilginin doğru alınma durumunu belirsiz hale getirir. Hataları

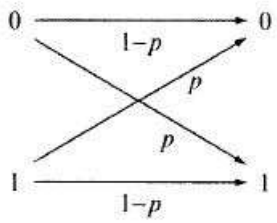


Şekil 1.5: 8-PSK kiplenimi

kanalın meydana getirdiği düşünülse de, hataların olmasının sebebi kanalın çözümleyici üzerindeki etkisidir.

Gönderilen sembollerin nasıl bozulduğu ile ilgili aşağıda birkaç kanal tanımı sıralanmıştır:

- Hafızasız kanal– hata olasılığı bir sembolden diğerine bağımsızdır.
- Simetrik kanal– Değeri i olarak gönderilen bir sembolün j olarak alınma olasılığı, değeri j olarak gönderilen sembolün i olarak alınma olasılığına, i ve j'nin bütün değerleri için eşittir. Ortak örneklerden biri, bit hata olasılığı p olan İkili Simetrik Kanal (BSC)'dır.



Şekil 1.6: BSC modeli

- Toplanabilir Beyaz Gaussian Gürültü Kanalı (AWGN) gönderilen sembollerin, büyüklüğü Gaussian dağılımlı rastgele değişken olan geniş bantlı gürültüyle toplandığı hafızasız bir kanaldır. Bu tez çalışmasında AWGN kanal modeli kullanılmıştır.

- Birleşik Kanal– hatalar, dağılmış ve rastgele hataların karışımından oluşur. Gerçekte bütün kanallar birleşik kanal davranışı sergilerler.

1.7 KİPÇÖZME

1.7.1 Uyumlu Kipçözme

Kipçözücü gönderilen sembollerin değerlerini belirler ve bu değerleri bir sonraki katmana gönderir. Bu genellikle olabilecek gönderilen sinyaller ile ilintiye bakılarak yapılır. Örnek olarak BPSK ele alınırsa, buradaki ilinti taşıyıcının fazıdır ve belirleyicide ya pozitif ya da negatif bir değer üretilir. Gürültü olmadığı durumda, bulunan sinyal seviyesi $\sqrt{E_r}$ olarak alınır, burada E_r alınan her bitin enerjisidir. AWGN kanalın etkisi, sıfır ortalama ve σ standart sapmasına sahip Gaussian dağılımından alınan gürültü örneklerinin toplanması ile bulunur. Olasılık yoğunluğu şu şekilde verilir:

$$p(n) = \frac{1}{\sigma\sqrt{2\pi}} e^{-n^2/2\sigma^2} \tag{1.2}$$

Beyaz Gaussian gürültü düz bir spektruma sahiptir ve gürültü seviyesi N_0 olarak yazılan tek taraflı gürültü güç spektral yoğunluğu ile tanımlanır. Gaussian gürültünün varyansı (σ^2), bir bit aralığında integrallenir ve değeri $N_0/2$'ye eşittir.

Kipçözücü karar vermeyi alınan sinyalin işaretine göre yapar. Böylece, eğer gürültü seviyesi $\pm\sqrt{E_s}$'den büyükse hata meydana geleceği anlamı taşımaktadır [11]:

$$p = \frac{1}{\sqrt{\pi N_0}} \int_{\sqrt{E_r}}^{\infty} e^{-n^2/N_0} dn \tag{1.3}$$

$t = n/\sqrt{N_0}$ eşitliğini yer değiştirirsek

$$p = \frac{1}{\pi} \int_{\sqrt{\frac{E_b}{N_0}}}^{\infty} e^{-t^2/N_0} dt \tag{1.4}$$

$$p = \frac{1}{2} erfc\sqrt{\frac{E_b}{N_0}} \tag{1.5}$$

olur ve $erfc(x) = \frac{2}{\sqrt{\pi}} \int_t^{\infty} e^{-t^2} dt$ olarak tanımlanmaktadır.

$erfc(x)$'in en büyük değeri 1'dir ve böylelikle Eşitlik 1.5'deki en yüksek bit hata olasılığı 0.5 olur. Bu sonuç mantıklıdır çünkü daha alınan sinyal olmadan, o bit değerini tahmin etmenin olasılığı %50'dir.

1.7.2 Ayrımsal Kiplenim

BPSK veya QPSK iletiminde, hatalardan kaynaklanan faz bilgisini belirleyememe durumunda alıcı karar veremez. Bu hatalar kanaldaki gecikmelerden kaynaklanıyor olabilir. Gecikmeler faz kaymalarına sebep olur ve bu kaymalar alıcı tarafından bilinmez. Bu sebepten bit değerlerinin gösterimi fazlar arasındaki farkla gösterilir. Bu metod Ayrımsal Kodlanmış Faz Kaydırmalı Anahtarlama (DEPSK) veya Ayrımsal Faz Kaydırmalı Anahtarlama (DPSK) olarak bilinir [21]. Bu ikisi kiplenim açısından bakıldığında birbirinin aynısıdır. Alıcı ardışık gelen iki sembolün fazlarını karşılaştırarak gönderilen bitlere karar verir. Alıcı kesin faz değerlerini bilmez, faz değerinin değişip değişmediği bilgisine sahiptir.

1.7.3 Yumuşak-Karar Verme Kipçözücüsü

Sert kipçözmede, alınan sinyale bakılarak kesin bir karar verilir. Bir başka kipçözme yöntemi ise alınan semboller hakkında kesin bir karar vermek yerine, emin olma dereceleri vermektir. Bu tip yapılan kipçözmeye yumuşak-karar verme kipçözmesi adı verilir. Bu kipçözme yöntemi, belirlenen seviyeyi gerçel bir sayı olarak işlemeyi esas alır.

Yumuşak çözmenin amacı, çözücüyü doğru karar vermesi için yönlendirmek olduğundan, kipçözücünün çıkışını performansın olasılıksal ölçümüyle ilişkilendirmek kullanışlı olur. Ortak ölçümlerden biri *log-olabilirlik oranı* olarak bilinir ve $log[p(1|r_i)/p(0|r_i)]$ olarak tanımlanır. Bu değerin hesaplanması başka bir şekilde şöyle ifade edilir:

$$log\left(\frac{p(1|r_i)}{p(0|r_i)}\right) = log\left(\frac{p(r_i|1)}{p(r_i|0)}\right) + log\left(\frac{p(1)}{p(0)}\right) \tag{1.6}$$

Mesaj bitleri olan 0 ve 1 değerlerinin aynı olasılığa sahip olduğu varsayılırsa, $log(\frac{p(1)}{p(0)}) = 0$ olur ve r_i alınan seviyesi için, log-olabilirlik değeri $log(\frac{p(r_i|1)}{p(r_i|0)})$ olur. Bu değer sinyal seviyesinin ve gürültü istatistiklerinin bilinmesi ile hesaplanabilir.

Gönderilen sinyalin Gaussian gürültüye maruz kaldığı kabul edilirse ve x sembolü gönderildiyse, alınan değer olan r_i'nin olasılık yoğunluk fonsiyonu

$$p(r_i|x) = \frac{1}{\sqrt{\pi N_0}} e^{-(r_i-x)^2/N_0} \tag{1.7}$$

olur.

Eşitlik 1.7'de, x değeri 1 ve 0 bitleri için sırasıyla $+\sqrt{E_r}$ ve $-\sqrt{E_r}$ olur. Böylelikle log-olabilirlik oranları $log[e^{-(r_i-\sqrt{E_r})^2/N_0}/e^{-(r_i+\sqrt{E_r})^2/N_0}]$ ile doğrusal olarak artar.

1.8 KOD ÇÖZME

Hafızasız bir kanalda, en iyi çözümleme stratejisi kod çözücünün alınan verileri bütün kod kelimeleriyle karşılaştırmasıdır ve alınan veriye en yakın kod kelimesini seçmesidir. Bu *en yüksek olabilirlik çözmesi* olarak bilinir.

1.8.1 Kodlama ve Kod Çözme Örnekleri

Şekil 1.7'de bir blok kod verilmiştir. Bu kod sistematiktir. Bu şu anlama gelmektedir: kod kelimesi bilgi bitlerini ve eşitlik kontrol bitlerini sırasıyla içerir. Burada eşitlik kontrol bitleri, bilgi bitlerinden hesaplanmaktadır. Şekil 1.7'de, bir kod kelimesi diğer bir kod kelimesinden en az üç yerde farklılık göstermektedir. Bu değer *en düşük Hamming uzaklığı* veya kısaca *en düşük uzaklık* olarak adlandırılır. Eğer bir bit yanlışsa, alınan dizi gönderilen kod kelimesine hala yakındır ancak iki veya daha fazla bit yanlışsa, alınan dizi başka bir kod kelimesine yakınsar. 00101 kod kelimesi için uzaklıklar Şekil 1.8'de verilmiştir. Bu örnekte eğer kipçözücüye gelen kod kelimesi 00101 ise, kod çözücü 10101'in gönderildiğine karar verir.

Bilgi	Kod Kelimesi
00	00000
01	01011
10	10101
11	11110

Şekil 1.7: Örnek blok kod ([24]'den alınmıştır.)

Kod Kelimesi	Uzaklık
00000	2
01011	3
10101	1
11110	4

Şekil 1.8: Uzaklıklar ([24]'den alınmıştır.)

1.8.2 Yumuşak-Karar Verme Kod Çözmesi

Bir r dizisi alındığında, n uzunluğunda bir c dizisinin gönderilme olasılığı $\prod_{i=0}^{n-1} p(c_i|r_i)$'dir. Bu değer tüm kod kelimeleri için bulunabilir. Çarpımlara alternatif olarak, bu çarpımların logaritmaları alınıp $\sum_{i=0}^{n-1} log\left(p(c_i|r_i)\right)$ değerlerinin en büyük olanı bulunur. Bu tez çalışmasında yumuşak-karar verme kod çözmesi kullanılacaktır.

1.9 KODLAMA İLE İLGİLİ TERİMLER

Bu tez içerisinde sıkça kullanacağımız ve kodlama teorisindeki bazı terimler ve açıklamaları ilerleyen satırlarda verilmiştir.

Bir A alfabesi üzerinde tanımlı n uzunluğunda bir blok kod A^n kümesinin alt kümesidir. Bu alt kümenin elemanlarına *kod kelimeleri* adı verilir. Kodların hem analiz hem de tasarımında yeni yapılar ile karşılaşılabilir. Bugün kullanılan çoğu kod doğrusal koddur ve doğrusal kodlar A alfabesi bir alan oluşturduğunda tanımlanan kodlardır. F alanı üzerinde tanımlı (n,k) doğrusal kodu, F^n'nin k-boyutlu vektör alt uzayıdır. k kodun boyutu olarak adlandırılmaktadır. $r = n - k$ ise kodun fazlalığı ve $R = k/n$ kodun oranı olarak tanımlanmaktadır. Örnek olarak ikili doğrusal kodlar, ikili alanlar üzerinde tanımlıdır.

Bir doğrusal kodu tanımlamak için, kod alanının bütün elemanlarını listelememize gerek yoktur. Bunu bir temel çerçevesinde gösterebiliriz. Bu gösterim *oluşturma matrisi* gösterimidir. (n,k) doğrusal kodu için tanımlı bir oluşturma matrisi (G), $k \times n$ boyutludur. Bu matrisin satırları kodun temelini oluşturmaktadır. Veri sembollerini u vektörü temsil etsin. Bu durumda u, F^k uzayı üzerinde tanımlanır ve uG, kod kelimelerinin kümesinde tanımlanır. u ve G ile c kod kelimesi şu şekilde birbirine bağıntılıdır:

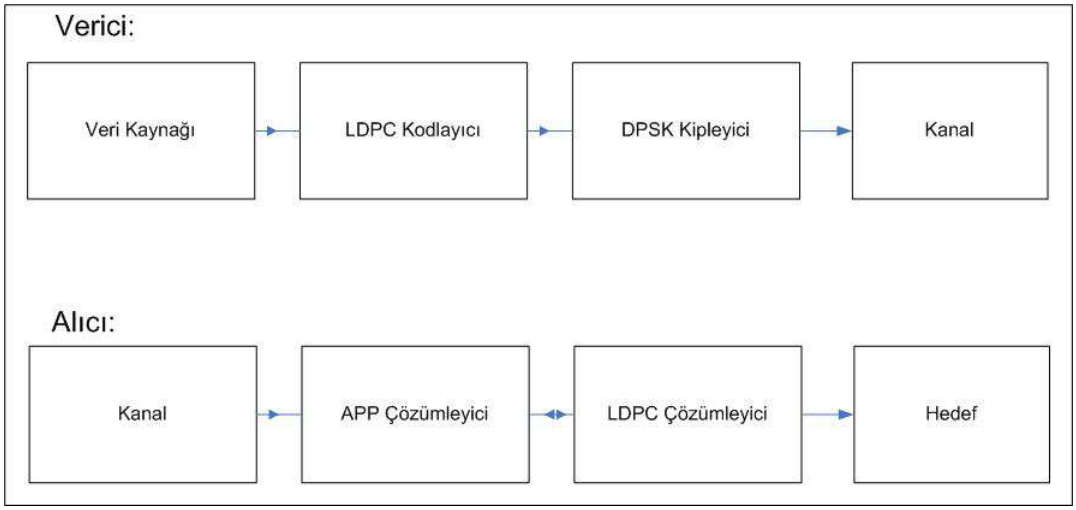
$$c = uG \tag{1.8}$$

Doğrusal kodlar için başka bir gösterim de *eşitlik kontrol matrisi* gösterimidir. (n,k) doğrusal kodu için tanımlı eşitlik kontrol matrisi $(n - k) \times n$ boyutludur ve bu matrisin satırları bu koda dikgen olan vektör uzayının temelini oluşturur. Bir c kod kelimesinin geçerli bir kod kelimesi olabilmesi için aşağıdaki eşitliği sağlaması gereklidir:

$$Hc = 0 \tag{1.9}$$

1.10 TEZİN AMACI VE KAPSAMI

Kodlamanın performansını artıran tekniklerden biri kanal kodlamasını ve DPSK kiplemeyi seri birleştirmektir. Bu yapının çözümlenmesi, alıcıda döngülü yapıda gerçekleştirilebilir. Bu şekilde bir seri birleştirme, kısa blok kodlar için [18] nolu referansta incelenmiştir. Bu tezdeki amaç ise, LDPC kodların DPSK ile seri birleştirilmesini uyumlu kipçözümleme için incelemektir. Tez çalışmaları sırasında [8] nolu referansta bu birleştirmenin önerildiği ve incelendiği görülmüştür. Dolayısı ile bu tezdeki çalışmalar literatüre orijinal bir katkı olarak değerlendirilmemelidir. Bu tezin amacı, [8] nolu referansta yüzeysel olarak bahsedilen kavramları ve sistem bileşenlerini derinlemesine kavramak, gerekli çıkarımları yapmak, bunları simülasyon ortamında gerçekleştirmek ve elde edilen kazanımları konuya yeni başlayan birinin kavrayabileceği şekilde sunmaktır. Tezde ele alınan haberleşme sistemine ait yapı Şekil 1.9'da gösterilmiştir:



Şekil 1.9: Alıcı ve vericiye ait genel yapı

Tezin yapısı ve içeriği şu şekildedir: Bölüm 2'de DPSK kiplenimi ve DPSK'in yumuşak kipçözmesine ait bilgiler verilecektir. Bu bölümde BCJR algoritması ve olasılık etki alanında işleyişi anlatılacaktır. Bölüm 3'te Düşük Yoğunluklu Eşitlik Kontrol (LDPC) Kodlarına ait bilgiler verilecektir. Çarpan çizgeleri, Tanner grafikleri ve LDPC kodların çözümlenmesi için kullanılan topla-çarp algoritmasına ait bilgiler açıklanacaktır. Bölüm 4'te, bu tezde simüle edilen haberleşme sistemine ait bilgiler detaylı olarak verilecektir. Döngülü çözümlenen kodların birleştirilmesinin yumuşak çözümlenmesi anlatılacak ve bu kodlarla dizayn edilen alıcının performansı bu bölümde

işlenecektir. Bölüm 4'te anlatılan haberleşme sisteminin alıcısını eniyilemek için kullanılan EXIT grafiği ve histogram ile pdf kestirimi metodu ile çizdirilen grafikler ise Bölüm 5'te verilecektir. Ayrıca eniyileme yapılmış derece dağılımları kullanılarak oluşturulan alıcıya ait işaret gürültü oranına (SNR) karşılık bit hata oranı (BER) performansları da Bölüm 5'te sergilenecektir. Son olarak, Bölüm 6'da, yapılan bütün bu çalışmalardan çıkarılan sonuçlar değerlendirilecektir.

2. AYRIMSAL KODLAMA VE İLERİ-GERİ ALGORİTMASI

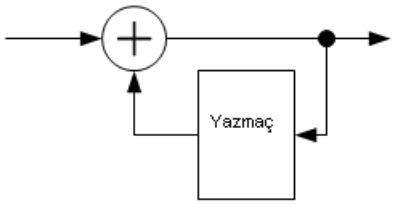
2.1 AYRIMSAL KODLAMA

Sayısal iletişimde, bit verileri iletişim kanalında plansız bir şekilde ters çevrilebilir. Bu ters çevrilmelere karşı ayrımsal kodlama tekniği kullanılmaktadır ve bu teknikte oluşturulan mesajlar o an gönderilen bit dışında kendisinden önce gelen bit değerine de bağlıdır.

Gönderilmesi istenen bitin x_i ve o an gönderilen bitin y_i olduğu kabul edilirse, verici tarafında kodlama:

$$y_i = y_{i-1} \oplus x_i \tag{2.1}$$

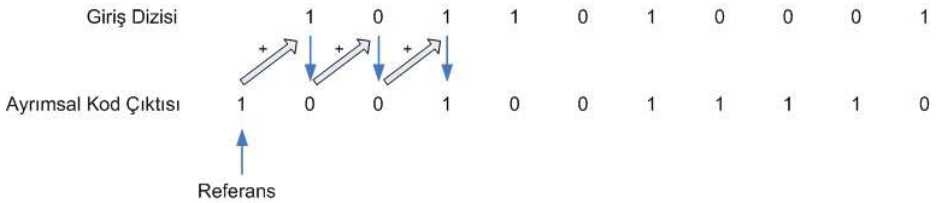
eşitliği ile yapılır. Burada $\oplus$ işareti *modulo-2* toplamayı simgelemektedir. Verici tarafındaki bu kodlama Şekil 2.1'de temsili olarak verilmiştir:



Şekil 2.1: Ayrımsal kodlama

Bu şeklin nasıl çalıştığı bir örnekle açıklanacaktır. Şekil 2.2'de görüldüğü gibi bir giriş dizisi tanımlansın. Ayrıca kodlamaya başlamadan önce bir referans biti tanımlansın. Referans biti "1" veya "0" olabilir. Aşağıdaki örnekte referans biti "1" alınmıştır. Gelen veri, Eşitlik 2.1 kullanılarak referans bitine eklenir ve bu böyle devam ederek kodlanmış veri elde edilir [21].

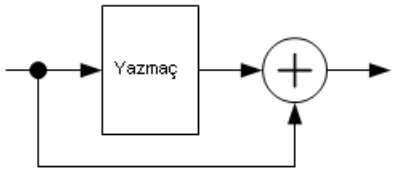
Örnek:



Şekil 2.2: Ayrımsal kodlama örneği

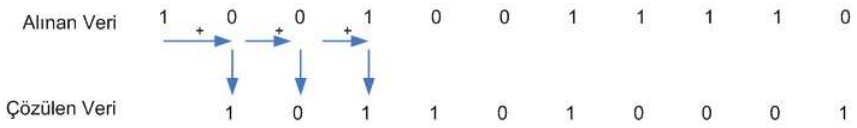
Ayrımsal kodlanmış verinin çözümlenmesi için, kodlanırken yapılan işlemin tersi işlem yapılacaktır. Çözmeye ait formülasyon Eşitlik 2.2'de ve çözme işlemi Şekil 2.3'te gösterilmiştir:

$$x_i = y_i \oplus y_{i-1} \tag{2.2}$$



Şekil 2.3: Ayrımsal çözme

Herhangi bir hata olmadığı varsayılarak yapılan çözme işlemi Şekil 2.4'te verilmiştir:



Şekil 2.4: Ayrımsal çözme örneği

Sembol temelli düşünüldüğünde ve kiplenim tekniği olarak PSK (veya DPSK) kullanıldığı varsayılırsa, ayrımsal kodlama kullanılarak bu semboller arasındaki faz belirsizliğine önlem alınmış olunur.

Kaynaktan gelen veri sembolleri ayrımsal kodlama ile kodlandığında bu kodlanmış veri sert şekilde alıcı tarafında çözülebilir. Ancak yumuşak çözmenin sert çözmeye karşı daha iyi sonuçlar vermesinin yanı sıra oluşturulacak döngülü alıcıda olasılık değerlerinin kullanılacak olması, dizayn eden kişiyi ayrımsal kodlanmış bir dizinin yumuşak bir şekilde çözümlenmesine götürmektedir.

Eğer PSK kipleniminde M adet sembol varsa, ayrımsal kodlanmış verinin çözümlenmesinin gösterimi M durumlu bir kafes diyagramı ile yapılabilir. Oluşturulan kafes diyagramını baz alarak ve yumuşak çözümleme yapılacağını varsayarak, kafes üzerinde yapılan yumuşak çözme algoritmalarından, sembol hata oranlarında en iyi

sonucu veren algoritma olan ileri-geri algoritmasının (BCJR) kullanılması daha etkin olacaktır.

Bundan sonraki bölümlerde ileri-geri algoritmasına giriş yapılarak, bu algoritmanın çalışma prensipleri anlatılacaktır.

2.2 MARKOV KAYNAK MODELİ

İleri-geri algoritması, çıkışları hafızasız bir kanaldan gözlemlenen bir Markov kaynağının durumları ve geçişleri ile ilişkilendirilmiştir ve bu algoritma *sonsal* olasılıkları hesaplamak için kullanılır. Böylelikle her adımda çözme işlemi uygulanmaz yani Markov kaynağının her durumunda ve geçişinde karar verilmez.

İleri-geri algoritması tezdeki çalışmalarda yumuşak giriş yumuşak çıkış çözücülerinde kullanılacaktır. Bu algoritmanın bir diğer adı da BCJR algoritmasıdır [2]. Bu bölümün geri kalanında [2] nolu referansta kullanılan gösterim kullanılmıştır.

Bu bölümde bir sınırlı-durum kesikli Markov kaynağının bazı özellikleri verilecektir. Markov kaynağı t örnekleminde, kaynağın durumu $S_t = m, m \in (0, ..., M-1)$ ve kaynağın çıkışı X_t olacak şekilde karakterize edilir. M durumların sayısını ve X ise χ alfabesinin elemanlarını gösterir. χ'in eleman sayısı sınırlıdır.

Kaynak durumlarının bir Markov zinciri oluşturması şu koşula bağlıdır:

$$Prob(S_t = m_t | S_{t-1} = m_{t-1}, ..., S_1 = m_1) = Prob(S_t = m_t | S_{t-1} = m_{t-1}) \tag{2.3}$$

Modelde t anındaki çıkış X_t, sadece t ve $t-1$ anlarındaki durumlara bağlıdır. Kaynak çıkışları ise Eşitlik 2.4 ile verilen olasılık değerleri ile karakterize edilir:

$$Prob_t(X_t = X | S_t = m, S_{t-1} = m^{'}) \tag{2.4}$$

Burada bir kaç tanımlama yapmak gerekirse:

$$p_t(m|m^{'}) = Prob_t(S_t = m | S_{t-1} = m^{'}) \tag{2.5}$$

$$q_t(X|m, m^{'}) = Prob_t(X_t = X | S_t = m, S_{t-1} = m^{'}) \tag{2.6}$$

eşitlikleri ile p_t ve q_t değerleri tanımlanır.

2.3 KESİKLİ HAFIZASIZ KANAL MODELİ

Bir kesikli hafızasız kanal şu özelliğe sahiptir: X_1^t, χ alfabesinden olan t uzunluğunda kanal giriş sembol dizisini ve Y_1^t, kanal çıkışı dizisini belirtmektedir. Kanalın t anındaki geçiş olasılıkları r_t olarak ifade edilirse,

$$P(Y_1^t|X_1^t) = \prod_{j=1}^{t} r_t(Y_j|X_j) \tag{2.7}$$

eşitliği yazılır. Çözücünün görevi, Y_1^T'yi bulup, Markov kaynağının durumlarına ve geçişlerine ait APP değerlerini tahmin etmektir. Burada Y_1^T, $t = 1$'den $t = T$'ye kadar olan Y değerlerini ifade etmektedir.

2.4 İLERİ-GERİ ALGORİTMASI

Kesikli hafızasız kanal modelinde, Markov kaynağının X_t çıkış sembolleri kesikli hafızasız kanalın girişleri olarak ve bu kanalın çıkışları da Y_t olarak adlandırılsın. $t = 0$ ve $t = T$ zamanlarında kaynağın durumları biliniyor olsun ve bu durumlar $S_0 = 0$ ve $S_T = 0$ olarak ifade edilsin.

İleri-geri algoritmasının amacı aşağıdaki *sonsal* olasılıkları hesaplamaktır:

$$Prob(S_t = m|Y_1^T)$$
$$Prob(S_t = m, S_{t-1} = m^{'}|Y_1^T) \tag{2.8}$$

Bu eşitlikler sırasıyla durum ve durum geçiş olasılıklarıdır.

Birkaç tanımlama daha yapmak gerekirse:

$$\lambda_t(m) = Prob(S_t = m, Y_1^T) = Prob(S_t = m|Y_1^T)P(Y_1^T) \tag{2.9}$$

ve

$$\sigma_t(m, m^{'}) = Prob(S_t = m, S_{t-1} = m^{'}, Y_1^T) = Prob(S_t = m, S_{t-1} = m^{'}|Y_1^T)P(Y_1^T) \tag{2.10}$$

Bu denklemlerde, $P(Y_1^T)$ değeri S_t ve S_{t-1} değerlerinden bağımsız olduğu için Eşitlik 2.8'deki olasılıklar yerine λ_t ve σ_t değerleri hesaplanabilir.

2.4.1 α, β ve γ Büyüklüklerinin Tanımları

İleri-geri algoritmasını anlatmak için aşağıdaki tanımlar kullanılacaktır [2]:

$$\alpha_t(m) = Prob(S_t = m, Y_1^t) \tag{2.11}$$

$$\beta_t(m) = Prob(Y_{t+1}^T | S_t = m) \tag{2.12}$$

ve

$$\gamma_t(m, m') = Prob(S_t = m, Y_t | S_{t-1} = m') \tag{2.13}$$

λ_t ve σ_t değerleri, α, β ve γ'nın fonksiyonları şeklinde yazılabilir:

$$\begin{aligned}
\lambda_t(m) &= Prob(S_t = m, Y_1^T) \\
&= Prob(S_t = m, Y_1^t, Y_{t+1}^T) \\
&= Prob(S_t = m, Y_1^t) Prob(Y_{t+1}^T | S_t = m, Y_1^t) \\
&= Prob(S_t = m, Y_1^t) Prob(Y_{t+1}^T | S_t = m) \\
&= \alpha_t(m)\beta_t(m)
\end{aligned} \tag{2.14}$$

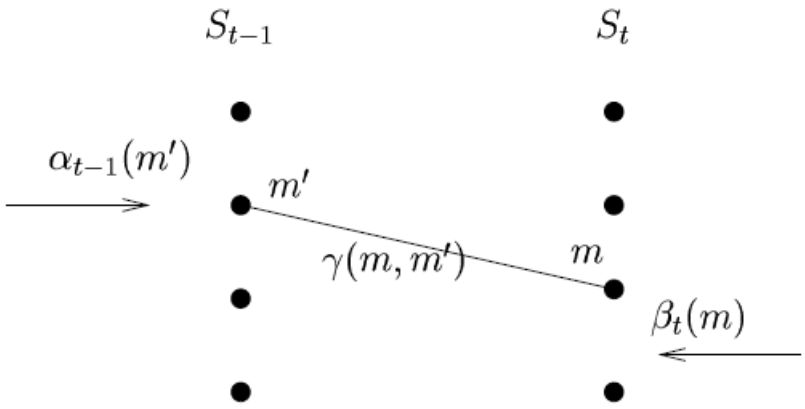
Eşitlik 2.14'te Markov kaynağının çıkışları, eğer t anındaki durum verildiyse, geçmiş çıkışlara bağlı değildir sonucu kullanılmıştır.

Aynı sonucu kullanarak:

$$\begin{aligned}
\sigma_t(m, m') &= Prob(S_t = m, S_{t-1} = m', Y_1^T) \\
&= Prob(S_t = m, S_{t-1} = m', Y_1^t, Y_{t+1}^T) \\
&= Prob(S_t = m, S_{t-1} = m', Y_1^t) Prob(Y_{t+1}^T | S_t = m, S_{t-1} = m', Y_1^t) \\
&= Prob(S_t = m, S_{t-1} = m', Y_1^t) Prob(Y_{t+1}^T | S_t = m) \\
&= Prob(S_{t-1} = m', Y_1^{t-1}) Prob(S_t = m, Y_t | S_{t-1} = m', Y_1^{t-1}) \times \\
&\quad Prob(Y_{t+1}^T | S_t = m) \\
&= Prob(S_{t-1} = m', Y_1^{t-1}) Prob(S_t = m, Y_t | S_{t-1} = m') Prob(Y_{t+1}^T | S_t = m) \\
&= \alpha_{t-1}(m')\gamma_t(m, m')\beta_t(m)
\end{aligned} \tag{2.15}$$

Eşitlik 2.15'e ait görsel gösterim yapılırsa: α geçmişteki gözlemlerin ve başlangıç geçiş durumu arasındaki ortak olasılığı, β bitiş geçiş durumu verildiğinde gelecek gözlemlere ait olasılık değerlerini ve γ geçiş metriklerini gösterir.

Artık α,β ve γ değerleri hesaplanabilir.



Şekil 2.5: α, β ve γ'nın gösterimi ([6] nolu referanstan alınmıştır.)

2.4.2 α'nın Döngüsel Hesaplanması

$$
\begin{aligned}
\alpha_t(m) &= Prob(S_t = m, Y_1^t) \\
&= \sum_{m'=0}^{M-1} Prob(S_t = m, S_{t-1} = m^{'}, Y_1^t) \\
&= \sum_{m'=0}^{M-1} Prob(S_{t-1} = m^{'}, Y_1^{t-1}) Prob(S_t = m, Y_t | S_{t-1} = m^{'}, Y_1^{t-1}) \\
&= \sum_{m'=0}^{M-1} Prob(S_{t-1} = m^{'}, Y_1^{t-1}) Prob(S_t = m, Y_t | S_{t-1} = m^{'}) \\
&= \sum_{m'=0}^{M-1} \alpha_{t-1}(m^{'}) \gamma_t(m, m^{'}) \qquad (2.16)
\end{aligned}
$$

Böylelikle α'lar döngüsel olarak γ'lar biliniyorsa hesaplanabilir. Bu döngüsellik, algoritmanın "ileri" kısmını oluşturmaktadır.

2.4.3 β'nın Döngüsel Hesaplanması

$$
\begin{aligned}
\beta_t(m) &= Prob(Y_{t+1}^T|S_t = m) \\
&= \sum_{m'=0}^{M-1} Prob(S_{t+1} = m', Y_{t+1}^T|S_t = m) \\
&= \sum_{m'=0}^{M-1} Prob(S_{t+1} = m', Y_{t+1}, Y_{t+2}^T|S_t = m) \\
&= \sum_{m'=0}^{M-1} Prob(S_{t+1} = m', Y_{t+1}|S_t = m) Prob(Y_{t+2}^T|S_t = m, S_{t+1} = m', Y_{t+1}) \\
&= \sum_{m'=0}^{M-1} Prob(S_{t+1} = m', Y_{t+1}|S_t = m) Prob(Y_{t+2}^T|S_{t+1} = m') \\
&= \sum_{m'=0}^{M-1} \beta_{t+1}(m')\gamma_{t+1}(m, m') \qquad (2.17)
\end{aligned}
$$

Böylelikle α'larda olduğu gibi β'lar da döngüsel olarak hesaplanmaktadır. Bu döngüsellik, algoritmanın "geri" kısmını oluşturmaktadır.

2.4.4 γ'nın Hesaplanması

Algoritmanın ileri ve geri kısımlarında döngüselliği gerçekleştirmek için γ değerlerinin bilinmesi gereklidir. Kesikli hafızasız kanal düşünülerek:

$$
\begin{aligned}
\gamma_t(m, m') &= Prob(S_t = m, Y_t|S_{t-1} = m') \\
&= Prob(S_t = m|S_{t-1} = m') Prob(Y_t|S_t = m, S_{t-1} = m') \\
&= \sum_{X\epsilon\chi} Prob(S_t = m|S_{t-1} = m') Prob(X_t = X, Y_t|S_t = m, S_{t-1} = m') \\
&= \sum_{X\epsilon\chi} Prob(S_t = m|S_{t-1} = m') Prob(X_t = X|S_t = m, S_{t-1} = m') \times \\
&\quad Prob(Y_t|X_t = X, S_t = m, S_{t-1} = m') \\
&= \sum_{X\epsilon\chi} Prob(S_t = m|S_{t-1} = m') Prob(X_t = X|S_t = m, S_{t-1} = m') \times \\
&\quad Prob(Y_t|X_t) \\
&= \sum_{X\epsilon\chi} p_t(m|m') q_t(X|m, m') r_t(Y_t|X_t) \qquad (2.18)
\end{aligned}
$$

şeklinde hesaplanabilir. Sonuçta bulunan p_t, q_t ve r_t değerleri sırasıyla Eşitlik 2.5, 2.6 ve 2.7'de tanımlanmıştır.

2.4.5 Algoritma Başlangıcı

Eşitlik 2.16 ve 2.17'nin ilk ve son terimleri için başlangıç değerlerinin atanmasına ihtiyaç vardır. Başlangıç ve son örneklemlerdeki Markov kaynağının durumlarına ait kabullenme kullanılacaktır. Bu kabullenmeler şu şekildedir: $S_0 = 0$ ve $S_T = 0$. Bu durumda başlangıç ve bitiş değerleri şöyle olmaktadır:

$$\begin{cases} \alpha_0(0) = 1, \\ \alpha_0(m) = 0, \qquad \forall m \in 1, ..., M-1 \end{cases} \tag{2.19}$$

ve

$$\begin{cases} \beta_T(0) = 1, \\ \beta_T(m) = 0, \qquad \forall m \in 1, ..., M-1 \end{cases} \tag{2.20}$$

Buradaki başlangıç değerleri, kafes diyagramlarının son durumunun sıfır olduğu kabul edilerek yazılmıştır. Bu kabullenme, ileri-geri algoritmasının kullanıldığı her durum için geçerli olmayabilir. Nitekim bu tez çalışmasında kullanılan başlangıç değerleri, buradaki değerlerden farklılık göstermektedir ve bu değerler Bölüm 4 içerisinde verilmiştir.

2.4.6 İleri-Geri Algoritmasının Özeti

İleri-geri algoritması Markov kaynağının *önsel* bilgisine ihtiyaç duymaktadır. Markov kaynağı ile ilgili bilgiler aşağıda sıralanmıştır:

- $p_t(m|m^{'})$, geçiş olasılığı
- $q_t(X|m, m^{'})$, çıkış olasılığı
- $r_t(Y|X)$, kanalın çıkış olasılığı

Kanal çıkışındaki gözlemler (Y_1^T), algoritmanın girişleridir. Bundan sonra amaç *sonsal* olasılıkları yani $t = 1, ..., T$ için λ_t'yi hesaplamaktır.
Algoritmanın adımları aşağıda verilmiştir [2]:

1. Başlangıç: α_t ve β_t'ların başlangıç değerleri, Eşitlik 2.19 ve 2.20 kullanılarak $t = 0$ ve $t = T$ için atanır.

2. γ_t'lerın Hesaplanması: Eşitlik 2.18 kullanılarak γ_t değerleri sırasıyla $t = 1$, $t = 2$, ..., $t = T$ için hesaplanır.

3. İleri Döngüsellik: α_t'ler Eşitlik 2.16 kullanılarak hesaplanır.

4. Geri Döngüsellik: β_t'ler Eşitlik 2.17 kullanılarak sırasıyla $t = T-1$, $t = T-2$, ..., $t = 0$ için hesaplanır. Bu hesaplamalar sadece son gözlemler uygun olduğunda başlayabilir.

5. *Sonsal* Olasılıkların Hesaplanması: Eşitlik 2.15 ve 2.14 kullanılarak hesaplamalar yapılır.

Markov kaynağının M adet durumu olduğu kabul edilirse, ileri-geri algoritmasının simülasyon ortamında gerçeklenmesi sırasında α_t ve β_t'lerin, her bir t zamanında $\Sigma_{m=0}^{M-1}\alpha_t = 1$ ve $\Sigma_{m=0}^{M-1}\beta_t = 1$ kısıtlarını sağlayacak şekilde normalizasyonu yapılır. Bunun sebebi, simülasyon ortamında döngüsel olarak hesaplanan α_t ve β_t'lerin t'nin büyük değerleri için sıfıra gitmesidir.

3. DÜŞÜK YOĞUNLUKLU EŞİTLİK KONTROL (LDPC) KODLARI

LDPC kodları ilk olarak Gallager tarafından 1960'ların başlarında bulunmuştur [9][10]. Gallager'ın bu buluşu 1981 yılında Tanner, LDPC kodlarına grafiksel bir bakış açısı katana kadar [25] araştırmacılar tarafından 20 sene kadar bir süre dikkate alınmamıştır. Tanner'ın bu çalışması da 14 sene kadar bir süre dikkate alınmamış ve 1990'ların sonlarına doğru gelişen teknolojinin etkisiyle araştırmacılar grafiksel ve döngüsel çözümlenen kodları keşfetmeye başlamıştır.

Gallager'ın çalışmalarından sonra LDPC kodları üzerine çalışılmaya devam edilmiş ve döngüsel çözme yöntemi kullanılan uzun LDPC kodların hata performanslarının, Shannon'ın limitine çok yakın değerler verdiği görülmüştür. Bu bulgular, yüksek güvenilirliğin önem kazandığı bu günlerde LDPC kodları, turbo kodlar ile sıkı bir rekabete sokmuştur.

Bu bölümde LDPC kodlarının temel çalışma prensiplerinden bahsedilecektir. LDPC kodlarına ait kodlama ve karmaşık çözme teknikleri ve bu kodların çarpan çizgesine dayalı döngülü çözümlenmesi bu bölümde ele alınacaktır.

3.1 TANIM

Doğrusal blok kodlar, oluşturma matrisi veya eşitlik kontrol matrisi ile tanımlanır. Bir LDPC kod C, oluşturma matrisi G ve eşitlik kontrol matrisi H ile tanımlanmaktadır. Eğer bu kod sadece eşitlik kontrol matrisi olan H ile tanımlanmışsa, C kodu H matrisinin boş (null) alanıdır. Boyutu n olan v=(v_0,v_1,...,v_{n-1}) vektörünün kod kelimesi olabilmesi için $vH^T = 0$ denklemini sağlaması gereklidir. Bu eşitlik aslında bize kod kelimesindeki bitlerin H'ın satırları tarafından tanımlanan eşitlik kontrol denklemlerini sağlaması gerektiğini göstermektedir. LDPC kodlar bu sebepten H matrisleri ile tanımlanırlar.

Düzenli LDPC kodlarında H eşitlik kontrol matrisleri şu sıralanan özelliklere sahiptir: (1) her satır ρ tane 1'den oluşmaktadır,(2) her sütun γ tane 1'den oluşmaktadır, (3) İki sütun arasındaki ortak 1'lerin sayısı λ, 1'den büyük değildir; yani λ=1 veya λ=0'dır, (4) ρ ve γ değerleri, kod uzunluğu ve H matrisinin satır sayısı ile karşılaştırıldığında olabildiğince küçük olmalıdır [9].

1 ve 2 numaralı özellikler H matrisinin sabit satır (ρ) ve sütun (γ) ağırlıklarına sahip olduğunu gösterir. 3 numaralı özellik H matrisinin iki satırının birden fazla ortak

1'e sahip olamayacağını açıklamaktadır. γ ve ρ değerleri kodun uzunluğu ve H matrisinin satır sayısı ile karşılaştırıldığında oldukça küçük olduğu için, H matrisi düşük yoğunlukta 1'e sahiptir. Bu sebepten, H matrisi *düşük yoğunluklu eşitlik kontrol matrisi* ve bu matrisin oluşturduğu kod da *düşük yoğunluklu eşitlik kontrol kodu* olarak adlandırılır. H matrisinin r değerini, toplam 1 sayısının toplam girdiye oranı olarak alırsak

$$r = \rho/n = \gamma/J \tag{3.1}$$

olur ve burada H matrisinin satır sayısı J olarak tanımlanmıştır. H'ın düşük yoğunluklu olması seyrek bir matris olması anlamına gelmektedir. Örnek olarak Şekil 3.1, $n = 20$, $J = 15$, $\rho = 4$, $\gamma = 3$ değerlerine sahip, düşük yoğunluklu bir eşitlik kontrol matrisini örneklemektedir.

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	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	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

Şekil 3.1: n=20, $\rho = 4$, $\gamma = 3$ değerlerine sahip LDPC eşitlik kontrol matrisi

Burada bahsedilen γ ve ρ değerleri ile oluşturulan (γ,ρ)-LDPC kodları düzenli kodlardır. Eğer H eşitlik kontrol matrisinin satır veya sütunları eşit ağırlığa sahip değilse bu LDPC kodu düzensizdir. Düzensiz LDPC kodları Bölüm 3.5'te açıklanmıştır.

3.2 ÇARPAN ÇİZGELERİ VE TOPLA-ÇARP ALGORİTMASI

3.2.1 Çarpan Çizgeleri

Çarpan çizgeleri, grafik modelleri ailesinde yer almaktadır. Grafiksel modelden kastedilen, bir matematik modelinin grafiksel gösterimidir. Bu grafiklerle, parametreler arasındaki etkileşimler grafiksel olarak gösterilir [15]. Bu modellerin kullanım alanlarına, hata düzeltme kodları, klasik filtre ve kontrol teorisi gibi giriş-çıkış sistem-

leri, elektriksel ağlar, sıradan veya parçalı diferansiyel denklemler ve ses, görüntü ve videonun istatistiksel modelleri örnek olarak verilebilir.

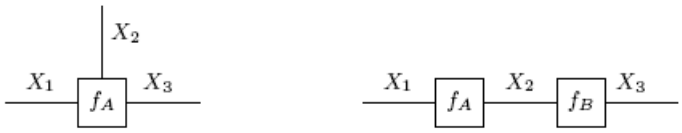
Çarpan çizgeleri aşağıda sıralanan sebeplerden dolayı kullanılır [17]:

- Çarpan çizgeleri, hiyerarşik modellemeye müsaade eder.
- Standart blok diyagramlarla uyumludurlar.
- Topla-çarp mesaj güncelleme kuralı, çarpan çizgesi üzerinde formüle edilebilmektedir.

Çarpan çizgeleri, fonksiyonların gösterimini belirtmektedir. Bu olaya birkaç örnekle bakılabilir.

Örnek 3.1: Yapısı olmayan bir fonksiyonun çarpan çizgesi

$f_A(x_1,x_2,x_3)$ fonsiyonunun çarpan çizgesi Şekil 3.2'de (solda) gösterilmiştir [6]. Düğümler fonksiyonların çarpanlarını, kenarlar ise değişkenleri tanımlamaktadır. Bir kenar bir düğüme ancak ve ancak o değişken belirtilen fonksiyonun bir argümanı ise bağlanır.



Şekil 3.2: Yapısız(solda) ve yapılı(sağda) fonksiyonların çarpan çizgeleri

Çarpan çizgesinin konsepti, gösterimi yapılacak fonksiyonun yapısının olmasıyla daha ilginç hale gelir. Bir fonksiyonun yapısının olması, o fonksiyonun çarpanlarına ayrılabildiği anlamı taşımaktadır.

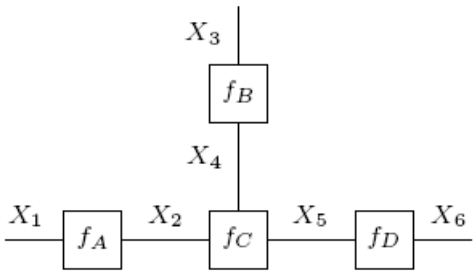
Örnek 3.2: Yapısı olan bir fonksiyonun çarpan çizgesi

$f(x_1, x_2, x_3)$ fonksiyonu, $f(x_1, x_2, x_3) = f_A(x_1, x_2).f_B(x_2, x_3)$ şeklinde çarpanlarına ayrılabiliyor olsun. Bu fonksiyona ait çarpan çizgesinin gösterimi Şekil 3.2'de (sağda) yapılmıştır. f *evrensel fonksiyon* ve f_A ve f_B *yerel fonksiyon* olarak tanımlanır.

Örnek 3.3: Evrensel Fonksiyon

$f(x_1,x_2,x_3,x_4,x_5,x_6) = f_A(x_1,x_2)\ .\ f_B(x_3,x_4)\ f_C(x_2,x_4,x_5)\ .\ f_D(x_5,x_6)$

olarak tanımlanırsa, bu fonksiyonun çarpan çizgesi Şekil 3.3'te gösterilmiştir [6].



Şekil 3.3: Çarpan çizgesi örneği

Bu bölümde çarpan çizgesi olarak, Forney çarpan çizgesi kullanılmıştır. Forney çarpan çizgesi şu şekilde tanımlanmaktadır [7]:

- Çarpan Çizgesi: Çarpan Çizgesi f fonksiyonunu göstermektedir ve düğüm ve kenarlardan oluşmaktadır. Burada f fonksiyonunun, çarpanlarına ayrılabildiği kabul edilmiştir.
- Evrensel Fonksiyonlar: f fonksiyonu evrensel fonksiyon olarak adlandırılır.
- Düğüm/Yerel Fonksiyonlar: Her bir çarpan için bir düğüm vardır ve bu düğüme yerel fonksiyon denir.
- Kenarlar/Değişkenler: Her bir değişken için bir adet kenar bulunmaktadır. Bir değişken bir düğümle de tanımlanabilir. Bu stildeki çarpan çizgeleri [15] nolu referansta tanımlanmıştır.
- Bağlantılar: Bir X değişkenini simgeleyen bir kenar, f çarpanının düğümüne ancak ve ancak X, f'in bir argümanı ise bağlanır.
- Konfigürasyon: Konfigürasyon, bütün değişkenlere belirli bir değer atanmasıdır. Bilinmeyen değişkenler için büyük harf kullanılır ve bilinen ve gözlenebilen değişkenler için küçük harfler kullanılır.

- Konfigürasyon Uzayı: Konfigürasyon Uzayı Ω, konfigürasyonların bütün kümesidir, evrensel fonksiyon f'in etki alanıdır. Değişkenlerin w konfigürasyonunun fonksiyonu olmasına dikkat edilmelidir.

- Geçerli Konfigürasyon: $w \epsilon \Omega$ konfigürasyonu $f(w) \neq 0$ ise geçerlidir.

3.2.2 Topla-Çarp Algoritması

Birçok problemde, sistemdeki değişkenlerin marjinal olasılıklarına ihtiyaç duyulabilir. Topla-çarp algoritması, çarpan çizgesi üzerinde tanımlı ve verilen bir evrensel dağılım fonksiyonundan, marjinal olasılıkların hesaplanmasını nümerik olarak etkili bir şekilde yapan bir araçtır. Bu algoritma [15] numaralı referansta detaylı bir şekilde anlatılmıştır. Algoritma çarpan çizgesi üzerinde mesaj geçişleri ile tanımlanır. Bu bölümde çarpan çizgesi üzerinde mesaj geçişleri ile marjinallerin nasıl hesaplanacağı anlatılacaktır.

Örnek 3.4: Çarpanlarına ayrılabilir bir fonksiyonun marjinal dağılımını bulma

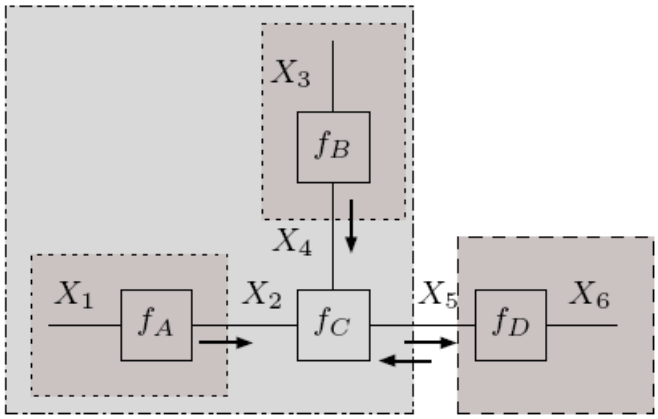
$f(x_1,x_2,x_3,x_4,x_5,x_6)$ evrensel fonksiyonu $f(...) = f_A(...)f_B(...)f_C(...)f_D(...)$ şeklinde çarpanlarına ayrılabiliyor olsun ve $f(x_5)$ marjinal fonksiyonu Eşitlik 3.2'deki gibi tanımlanmış olsun.

$$f(x_5) = \sum_{x_1,x_2,x_3,x_4,x_6} f(x_1, x_2, x_3, x_4, x_5, x_6) \tag{3.2}$$

Eşitlik 3.2 çarpanlarına ayrıldığında şu şekilde yazılır:

$$\begin{aligned} f(x_5) &= \sum_{x_1,x_2,x_3,x_4,x_6} f_A(x_1, x_2) f_B(x_3, x_4) f_C(x_2, x_4, x_5) f_D(x_5, x_6) \\ &= \underbrace{\sum_{x_2,x_4} f_C(x_2, x_4, x_5) \left(\underbrace{\sum_{x_1} f_A(x_1, x_2)}_{\mu_{f_{A\to x_2}}(x_2)} \right) \cdot \left(\underbrace{\sum_{x_3} f_B(x_3, x_4)}_{\mu_{f_{B\to x_4}}(x_4)} \right)}_{\mu_{f_{C\to x_5}}(x_5)} \\ &\quad \cdot \left(\underbrace{\sum_{x_6} f_D(x_5, x_6)}_{\mu_{f_{D\to x_5}}(x_5)} \right) \end{aligned} \tag{3.3}$$

Eşitlik 3.3'ün arkasında, toplamları olabildiğince sağa kaydırmak düşüncesi yatıyor. Örnek olarak, X_6'ya göre toplam alındığında, toplam işareti her çarpanın sağ tarafına kaydırılabilir, çünkü bu çarpan X_6'ya bağlıdır. Sonuç olarak, yüksek dereceli bir toplama yapmaktansa, daha basit toplamaları yapmak daha iyidir. Ara terimler olan $\mu_{f_{j\to x_i}}(x_i)$, X_i'nin fonksiyonlarıdır. Böyle bir fonksiyonun etki alanı X_i alfabesidir. Bunların anlamları Şekil 3.4'e bakıldığında daha iyi anlaşılabilir.



Şekil 3.4: $f(x_5)$'i hesaplamak için topla-çarp dağılımı

Ara sonuçlar, grafiğin kenarlarından akan mesajlar anlamına gelmektedir. Örnek olarak, $\sum_{x_1} f_A(x_1, x_2)$'nin toplamı olan $\mu_{f_{A\to x_2}}(x_2)$ mesajı, f_A düğümünü X_2 kenarından terk eden mesaj olarak yorumlanabilir. $\mu_{f_{A\to x_2}}(x_2)$ ve $\mu_{f_{B\to x_4}}(x_4)$ uygunsa, $\mu_{f_{C\to x_5}}(x_5)$ mesajı f_C düğümünün X_5 kenarına doğru giden çıkış mesajı olarak hesaplanır. Bu bilgiler ışığında Eşitlik 3.3 en son olarak şu şekilde yazılır:

$$f(x_5) = \mu_{f_{C\to x_5}}(x_5).\mu_{f_{D\to x_5}}(x_5) \tag{3.4}$$

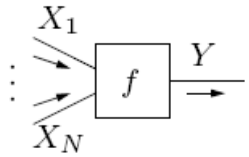
Bu formülasyon, aynı kenardaki iki mesajın çarpımını göstermektedir. Her mesaj arkasında ne yatıyorsa onun bir *özeti* gibidir. Bu, Şekil 3.4'teki kutuların yardımıyla daha iyi anlaşılabilir. Bir mesajın hesaplanması, grafiğin bir parçasının kapanması anlamına gelmektedir. Her bir kutucuk içerisinde detaylar toplanır ve özetleri yayılır. İlk adımda, Şekil 3.4'teki karanlık bölge kutucuk içerisine alınmıştır (sonuç olarak $\mu_{f_{A\to x_2}}(x_2)$, $\mu_{f_{B\to x_4}}(x_4)$ ve $\mu_{f_{D\to x_5}}(x_5)$ vermektedir). Daha sonra, Eşitlik 3.4'e varana kadar daha açık renk olan büyük kutucuk kapatılmıştır ($\mu_{f_{C\to x_5}}(x_5)$ değerini vermektedir).

Yarım kenarlar (X_1 gibi), bağlı olduğu düğüme bir mesaj taşımamaktadır. Alternatif olarak bu kenarlar nötr çarpan olan 1 mesajını taşıyan kenarlar olarak düşünülebilir. Bu akılda tutularak, Eşitlik 3.3'teki her mesaj (yani ara mesaj) aynı şekilde hesaplanır.

Topla-Çarp Kuralı [15]:

Şekil 3.5'teki X_1,...,X_N kenarlarının bağlandığı genel düğüm düşünüldüğünde, y kenarından ayrılan mesaj aşağıdaki kural ile hesaplanır.

$$\mu_{f\to y}(y) = \sum_{x_1,\ldots,x_N} f(y, x_1, \ldots, x_N).\mu_{f_{x_1}\to f}(x_1)\ldots\mu_{f_{x_N}\to f}(x_N) \tag{3.5}$$



Şekil 3.5: Genel bir kenardan giden mesajlar [6]

Bir f düğümünden Y kenarına giden mesaj, f fonksiyonu ve f'e yönelmiş diğer bütün kenarlardan gelen ve Y hariç bütün değişkenler üzerinde tanımlı mesajların çarpımıdır. Bu topla-çarp kuralıdır.

Örnek 3.5: Çevrimsel Mesajlar

Örnek 3.3'deki $f(x_1, x_2, x_3, x_4, x_5, x_6)$ evrensel fonksiyonunun $f(x_2)$ marjinal fonksiyonu şu şekilde yazılır:

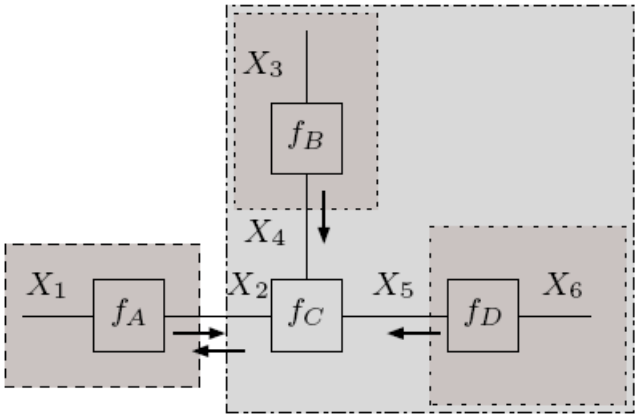
$$f(x_2) = \sum_{x_1,x_3,x_4,x_5,x_6} f(x_1, x_2, x_3, x_4, x_5, x_6) \tag{3.6}$$

Bu marjinal fonksiyon Şekil 3.6'da gösterildiği üzere topla-çarp kuralı ile hesaplanabilir. $\mu_{f_A\to x_2}(x_2)$, $\mu_{f_B\to x_4}(x_4)$ ve $\mu_{f_D\to x_5}(x_5)$ değerleri Örnek 3.4'de hesaplanan değerlerle aynı olduğundan, bu değerler $f(x_2)$ marjinal fonksiyonunda yerine konulursa, $f(x_2)$ şu şekilde yazılır:

$$f(x_2) = \mu_{f_A\to x_2}(x_2).\mu_{f_C\to x_2}(x_2) \tag{3.7}$$

Son örnekten, bir fonksiyona ait marjinallerin hesaplanması için bir kenardaki iki ayrı mesajın çarpıldığı görülür. Y değişkeninin $f(y)$ marjinali, Eşitlik 3.4 ve 3.7'de belirtildiği gibi aynı kenardaki iki mesajın çarpımıdır. Genel olarak,

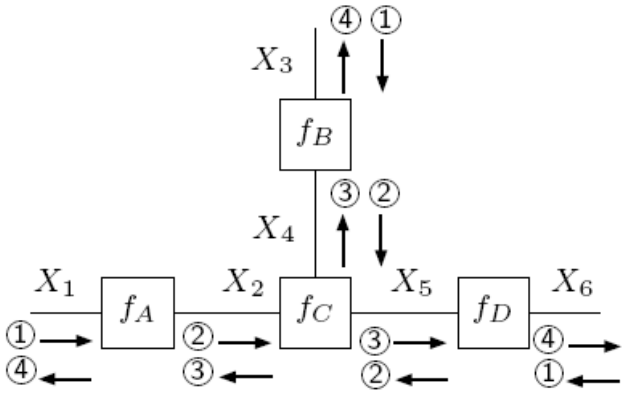
$$f(y) = \mu_{f_A\to y}(y).\mu_{f_B\to y}(y) \tag{3.8}$$



Şekil 3.6: $f(x_2)$'yi hesaplamak için topla-çarp dağılımı [6]

olarak formüle edilir. f_A ve f_B, Y kenarına bağlanmış iki düğümdür.

Topla-çarp algoritması, her bir kenardaki iki mesajı hesaplar. Döngülü olmayan çizge çarpanlarında (çizge ağaçlarında), marjinaller en uygun sayıdaki hesaplamayla bulunur. Mesaj hesaplanmasına en dışarıdaki kenarlardan başlanır ve bu hesaplamaya giriş mesajları uygun hale gelen düğümlerle devam edilir. Bu yolla her mesaj bir sefer hesaplanır. Mesajların hesaplanmasıyla ilgili örnek Şekil 3.7'de verilmiştir:



Şekil 3.7: Topla-çarp kuralına göre her bir kenardaki iki mesaj hesaplanır. Bu mesajlar $f(x_1)$, $f(x_2)$, $f(x_3)$, $f(x_4)$, $f(x_5)$ ve $f(x_6)$ marjinal fonksiyonlarının hesaplanması için gereklidir. Çember içerisine alınmış numaralar mesaj hesaplanma sırasını göstermektedir [6].

Özet olarak;

- Eşitlik 3.2'de örneği verilen bir fonksiyona ait marjinaller Eşitlik 3.8'deki iki mesajın çarpımı olarak hesaplanabilir.

- Bu mesajlar kendinden önceki çizge parçacığının özeti gibidir.
- Bütün mesajlar topla-çarp kuralına uygun olarak diğer mesajlar yardımıyla hesaplanır.

Çevrimsel Çarpan Çizgelerinde Topla-Çarp Algoritması:

Eğer çarpan çizgesi çevrimsel bir yapıdaysa durum biraz farklılaşır. Bu durumda, topla-çarp algoritması döngüsel hale gelir. Bir düğümdeki yeni bir çıkış mesajı aynı düğümdeki giriş mesajlarını başka bir yoldan etkileyebilir. Bu durumda algoritma, tam marjinal fonksiyonların karşılığını vermez. Yani bu algoritmanın yakınsayacağı garanti değildir. Buna rağmen, topla-çarp algoritmasını çevrimsel grafiklerde uygulamak mükemmel sonuçlar verir. Çoğu pratik durumda, algoritma kararlı bir noktaya ulaşır ve elde edilen marjinal fonksiyonlar yeterlidir, yani bu marjinaller üzerinden yapılan karar verme işlemleri doğru sonuçlara yeterince yakındır.

Çevrimsel grafiklerdeki topla-çarp algoritması aşağıdaki adımlardan oluşur:

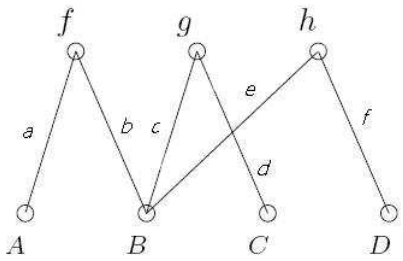
- İlk olarak bütün kenarlar nötr mesajı (yani $\mu(.)=1$) ile başlatılır.
- Bütün mesajlar döngüsel olarak belirli bir prosedüre göre güncellenir. Bu prosedür uygulamaya göre değişiklik gösterir.
- Yukarıdaki adımlardan sonra, marjinal fonksiyonlar Eşitlik 3.8'e uygun olarak hesaplanır.
- O anki marjinal fonksiyonlar üzerinden karar verme işlemi gerçekleştirilir.
- Algoritma, belirli bir zaman aşıldığında veya durma kriteri sağlandığında durur.

3.3 TANNER GRAFİKLERİ

3.3.1 Tanner Grafikleri İle İlgili Genel Tanımlar

Bir G grafiği V=(v_1,v_2,...) olarak simgelenen *doruk* kümelerinden ve ε=(e_1,e_2,...) olarak adlandırılan *kenar* kümelerinden oluşmaktadır. e_k kenarı sıralanmamış durumda olan (v_i, v_j) dorukları ile tanımlanmaktadır. Böyle bir G grafiği $G = (V, \varepsilon)$ olarak simgelenir. e_k kenarı ile ilişkilendirilmiş olan v_i ve v_j dorukları e_k'nın *son dorukları* olarak adlandırılır. Bir grafiğin dorukları nokta ile ve her bir kenarı

bir çizgi ile gösterilir. Bu grafiksel gösterimle, bir kenarın iki son doruğu bir çizgi ile bağlanmaktadır ve kenarlar kendi son doruklarının *olayını* taşırlar. v_i doruğunun olayını taşıyan kenarların sayısı v_i doruğunun *derecesi* olarak adlandırılır ve $d(v_i)$ olarak ifade edilir. Şekil 3.8'de 7 doruk ve 6 kenardan oluşan bir grafik gösterilmektedir.



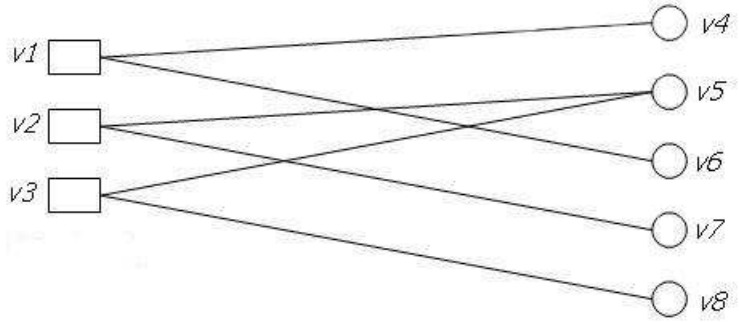
Şekil 3.8: 7 doruk ve 6 kenardan oluşan bir grafik

Kenar a, f ve A doruklarını bağlamaktadır. f doruğuna a ve b kenarları bağlıdır ve bundan dolayı f doruğunun derecesi ikidir. Ortak bir doruğa bilgi taşıyan iki kenar birbirinin *komşusu* veya *bağlanmışıdır*. İki doruk bir kenar ile bağlandığında, bu iki doruk komşu olur. Sonlu sayıda doruktan ve dolayısıyla sonlu sayıda kenardan oluşan grafiğe *sonlu grafik* denir.

Grafikteki bir *yol* sonlu sayıdaki alternatif doruk ve kenar sıralarından oluşmaktadır. Bu yol bir dorukla başlayıp bir dorukla bitmekte, her bir kenar bağlı olduğu doruklara olayları taşımakta ve hiçbir doruk bu yol üzerinde birden fazla bulunmamaktadır. Bu yol üzerindeki kenarların sayısı o yolun *uzunluğunu* vermektedir. Şekil 3.8'de uzunluğu 4 olan A,a,f,b,B,e,h,f,D yolu gösterilmektedir. Bir yolun aynı dorukta başlayıp aynı dorukta bitmesi olasıdır. Bu durumda bu yola *çevrim* denir. Çevrim üzerindeki başlangıç ve bitiş dorukları hariç hiçbir doruk birden fazla yer almaz. Çevrimleri olmayan bir grafik *ağaç* yapısındadır. Grafikteki en kısa çevrim, grafiğin *ölçüsü* olarak adlandırılmaktadır.

$G = (V, \varepsilon)$ grafiğinin V doruk kümesi iki ayrı V_1 ve V_2 alt kümelerine ayrılıyorsa ve ε içerisindeki her bir kenar, V_1 ve V_2 doruklarından birine bağlanıyorsa ve V_1 veya V_2'deki iki doruk birbirine bağlanmıyorsa, bu grafiğe *ikili* grafik denir.

Şekil 3.9'da $V_1 = (v_1, v_2, v_3)$ ve $V_2 = (v_4, v_5, v_6, v_7, v_8)$ olan bir ikili grafik verilmiştir. Eğer bir ikili grafik, $G = (V, \varepsilon)$, çevrimler içeriyorsa bu çevrimlerin uzunluğu çifttir. Grafikteki her bir doruk aynı zamanda *düğüm* olarak da adlandırılır.



Şekil 3.9: Bir ikili grafik

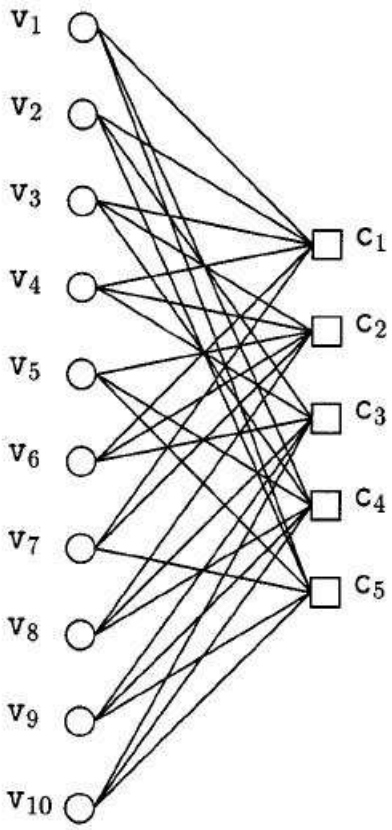
3.3.2 Tanner Grafiği Gösterimi

Doğrusal kodlar değişik yollarla grafikler ile gösterilirler. En çok bilinen grafiksel gösterimlerden biri kafes gösterimidir. Bir kodun Trellis gösterimi, evrişimsel bir koda ait kafes tabanlı çözme algoritmasının tasarlanmasına yardımcı olur. Doğrusal blok kodların gösterimi için ise *Tanner* grafikleri kullanılabilir. Tanner grafikleri, kodun bitleri ile eşitlik kontrolleri arasındaki ilişkiyi gösterir [25].

Uzunluğu n olan ve J satıra sahip eşitlik kontrol matrisi H'a sahip olan bir doğrusal blok kod için, iki doruk kümesinden (V_1 ve V_2'den) oluşan G_T grafiği oluşturulabilir. İlk V_1 kümesi n kod bitini gösteren n adet doruktan oluşur. $v_0, v_1, ..., v_{n-1}$ olarak ifade edilen bu doruklar kod-bit dorukları (veya değişken düğümleri) olarak adlandırılır. İkinci V_2 seti J doruktan oluşur ve $c_1, c_2, ..., c_j$ olarak gösterilen J tane eşitlik kontrol denklemlerini ifade eder. Bir kod bu eşitlikleri sağlamak zorundadır. Bu doruklar kontrol-toplam dorukları (veya kontrol düğümleri) olarak adlandırılır.

Değişken düğümü v_l, c_j kontrol düğümüne (v_l, c_j) olarak ifade edilen bir kenar ile bağlıdır ve bu kenar ancak ve ancak v_l kod biti c_j eşitlik kontrolünde yer alıyorsa bağlıdır. İki kod-bit doruğu birbirine bağlanmaz ve iki kontrol düğümü de aynı şekilde birbirine bağlanmaz. G_T grafiğinin ikili bir grafik olduğu açıktır. Bu grafikler ilk olarak Tanner tarafından önerilmiş ve LDPC kodlarının döngülü çözümü için kullanılmıştır. Bu sebepten bu grafiklere *Tanner grafikleri* adı verilir. v_l kod-bitinin derecesi, v_l doruğuna sahip olan kontrol-eşitlik toplamlarının sayısına eşittir ve aynı şekilde c_j kontrol toplamının derecesi, c_j tarafından kontrol edilen kod-bitlerinin sayısına eşittir.

Düzenli LDPC kodlar için oluşturulan Tanner grafiğinde kod-bit doruklarının dereceleri birbirine eşittir ve γ olarak gösterilir. Aynı şekilde düzenli LDPC kodlarda kontrol-toplam doruklarının derecesi birbirine eşit ve ρ olarak ifade edilir. Böyle bir Tanner grafiği *düzenli* olarak adlandırılır. LDPC kodlar topla-çarp algoritmasına dayalı döngülü çözme yöntemiyle çözülürler ve bu kodlara ait Tanner grafiklerinin kısa uzunlukta çevrimlere sahip olmaması istenir çünkü kısa uzunluktaki çevrimler çözme performansını limitler ve çözülmenin yakınsamasını önler [10]. Düzenli LDPC kodlar için kullanılan Tanner grafiği örneği şekil 3.10'da verilmiştir [25].



Şekil 3.10: (3,6)-düzenli LDPC koduna ait Tanner grafiği. Burada kullanılan kodun uzunluğu 10'dur. 5 adet kontrol düğümüne sahip olduğu için, eşitlik kontrol matrisi 5 satırdan oluşmaktadır.

Tanner grafikleri, çarpan çizgelerinin alt kümesinde tanımlıdır. Bölüm 3.2'de anlatılan çarpan çizgeleri üzerinde tanımlanan topla-çarp algoritması, Tanner grafikleri üzerinde de uygulanabilir. LDPC kodlarına ait H matrislerinin gösterimi, Tanner grafikleri ile yapılabildiğinden, herhangi alınan bir kodun çözümlenmesi bu grafikler üzerinde uygulanacak topla-çarp algoritması ile gerçekleştirilir. Topla-çarp algorit-

masının, LDPC kodlarının çözümlenmesindeki uygulanışı Bölüm 3.4'te verilmiştir.

3.4 LDPC KODLARINDA TOPLA-ÇARP KOD ÇÖZÜLMESİ

LDPC kodların çözümü için değişik algoritmalar geliştirilmiştir. Bunları; önemli-mantık çözme algoritması, bit-çevirme çözme algoritması, ağırlıklandırılmış bit-çevirme çözme algoritması, sonsal olasılık (APP) çözme algoritması ve topla-çarp algoritması olarak sıralayabiliriz. Bunlardan ilk ikisi sert karar verme çözümlemesi yaparken son ikisi yumuşak karar verme çözümlemesi yapar, ortadaki ise ikisinin karışımıdır. Önemli-mantık çözme tekniği içlerinde en az karmaşıklığa sahip çözme algoritmasıdır. Bit-çevirme çözümlemesi, önemli-mantık çözümlemesine göre daha karmaşıktır fakat daha iyi hata performansı verir. APP ve topla-çarp algoritması daha iyi hata performansına sahipken, önemli-mantık ve bit-çevirme algoritmalarına göre daha karmaşıklardır. Topla-çarp algoritması, LDPC kodların çözülmesi için kullanılan bu beş tip algoritma içerisinde en iyi hata performansı sunan çözme algoritmasıdır.

Topla-çarp algoritmasında mesajlar olasılık değerleri üzerinden hesaplanır. Bu olasılık değerleri log-olabilirlik oranlarıdır.

İkili sinyaller için p, mesaj bitinin 1 olma olasılığı ise, $1-p$ mesaj bitinin 0 olma olasılığıdır ve böylelikle log-olabilirlik oranının (LLR) tanımından

$$LLR(p) = \log\left(\frac{1-p}{p}\right) \tag{3.9}$$

olduğu görülür. $LLR(p)$'nin işareti sert karar vermek içindir ve $LLR(p)$'nin büyüklüğü ise kararın emin olma derecesi ile ilgilidir. Bu bölümdeki işlemler logaritmik gösterimlerle gerçekleştirilecektir çünkü olasılık değerleri çarpımla, log-olabilirlik oranlar ise toplanarak hesaplandığı için, logaritmik değerler çözme karmaşıklığını düşürmektedir.

Topla-çarp algoritmasında amaç, her bir kod biti için sonsal olasılıkları (APP) hesaplamaktır. Bu $P_i = P\{c_i = 1|N\}$ olarak ifade edilir ve bu ifade, i numaralı kod bitinin N olayı altında 1'e eşit olması olasılığıdır. N olayı ise bütün eşitlik-kontrol kısıtlarının sağlandığı olaydır. *İçsel* veya *önsel olasılık*, P_i^{int}, kod kısıtlamalarının bilgisinden bağımsız olan orjinal bit olasılığıdır ve *dışsal olasılık* P_i^{ext}, N olayından ne öğrenildiğini betimlemektedir.

Topla-çarp algoritması döngülü bir şekilde her bir kod biti için APP değerlerinin tahminini hesaplar. O kod için çizilen Tanner grafiğinde çevrimler

yoksa bu tahminler doğrudur. Bir döngü sırasında eşitlik-kontrol kısıtlamalarından kazanılan dışsal bilgi bir sonraki döngüde *önsel* bilgi olarak kullanılır. Eşitlik-kontrol kısıtlamalarından kazanılan dışsal bilgi, döngüye ilk başlandığı zaman sahip olunan *önsel* bilgiden bağımsızdır. Ardışık devam eden döngülerde edinilen dışsal bilgi düğümlere bir çevrimle geri dönmeyene kadar orjinal *önsel* olasılıktan bağımsızdır.

LDPC kodları için topla-çarp algoritması aşağıdaki şekildedir [13]:

Adım 1. Başlangıç: i numaralı bit düğümünden j numaralı kontrol düğümüne giden başlangıç mesajları kanal çıktısı olan y_i'den elde edilen LLR değerleridir. Kanalın AWGN olduğu düşünülürse LLR değerleri işaret-gürültü oranı üzerinden

$$L_{i,j} = R_i = 4y_i \frac{E_b}{N_0} \tag{3.10}$$

şeklinde hesaplanır [13]. Eşitlik 3.10'da R_i, i numaralı kod bitinin orjinal LLR'ı ve E_b/N_0 ise sinyal gürültü oranı olarak tanımlanır. Tezde gerçekleştirilen modellemede, başlangıç LLR değerleri olarak, APP kipçözüsünden gelen LLR değerleri kullanılmıştır.

Adım 2. Kontrol düğümlerinden bit düğümlerine: j numaralı kontrol düğümünden i numaralı bit düğümüne giden dışsal mesaj, i numaralı bit 1 olarak kabul edildiğinde j numaralı kontrol denkleminin sağlanma olasılığına eşittir. Bu dışsal mesaj LLR olarak

$$E_{i,j} = \log\left(\frac{1 + \prod_{i' \epsilon B_j, i' \neq i} \tanh\left(L_{i',j}/2\right)}{1 - \prod_{i' \epsilon B_j, i' \neq i} \tanh\left(L_{i',j}/2\right)}\right) \tag{3.11}$$

şeklinde hesaplanır [13]. B_j gösterimi, bitlerin j numaralı eşitlik-kontrol denklemi içerisinde yer alan sütun numaralarının kümesini belirtmektedir.

Adım 3. Kod testi: Birleştirilmiş LLR, dışsal LLR ile 1 numaralı adımda hesaplanan orjinal LLR'ın toplamıdır [13]:

$$L_i = \sum_{j \epsilon A_i} E_{i,j} + R_i \tag{3.12}$$

Her bir bit için sert karar verme yapılır:

$$z_i = \begin{cases} 1, & L_i \leq 0 \\ 0, & L_i > 0 \end{cases} \tag{3.13}$$

Eğer $z = [z_1, ..., z_n]$ geçerli bir kodsa (yani $Hz^T = 0$ ise) veya izin verilen en yüksek sayıdaki döngü tamamlanmışsa algoritma durur.

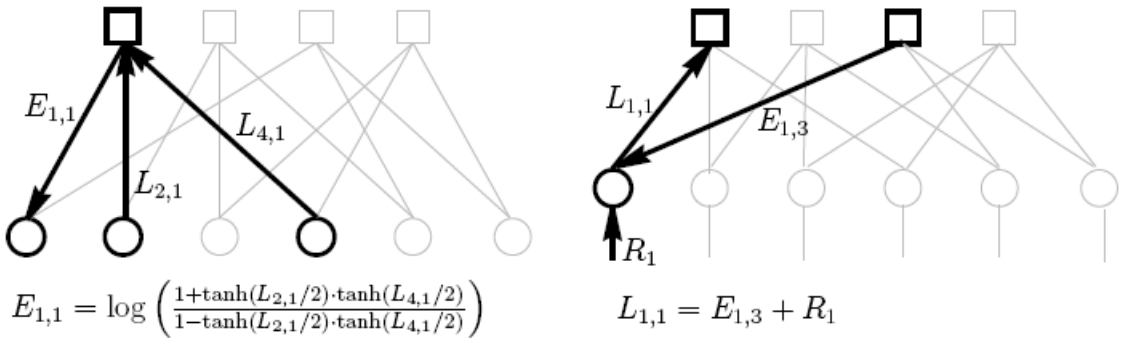
Adım 4. Bit düğümlerinden kontrol düğümlerine: Bit düğümlerinden kontrol düğümlerine giden mesajlar LLR olarak:

$$L_{i,j} = \sum_{j' \epsilon A_i, j' \neq j} E_{i,j'} + R_i \tag{3.14}$$

hesaplanır [13]. A_i gösterimi, kodun i numaralı bitinin kontrol edildiği eşitlik-kontrol denklemlerinin satır numaralarının kümesini belirtmektedir.

Adım 2'ye dönülür.

Eşitlik 3.11 ve 3.14'in uygulamalarına yönelik örnek, Şekil 3.11'de verilmiştir. Kontrol düğümünden bit düğümüne giden dışsal bilgi, bu bitin sahip olduğu önsel bilgiden bağımsızdır. Bir sonraki döngüde kontrol düğümlerinden giden dışsal mesajlar bit düğümleri için önsel olasılıkları olarak kullanılır.

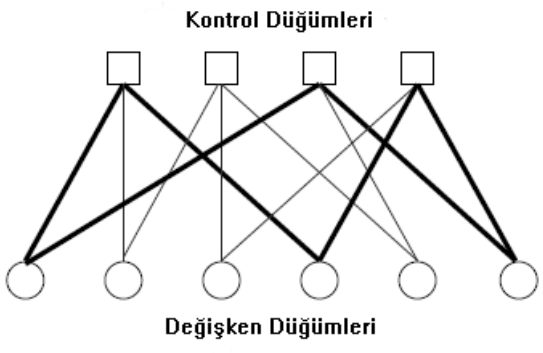


Şekil 3.11: Topla-çarp algoritması için mesaj örnekleri (Bu örnek için verilen H matrisi ve Tanner grafiği Şekil 3.12 ve 3.13'de verilmiştir [13].)

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

Şekil 3.12: H matris örneği

Örnek 3.6: Topla-çarp algoritmasının gücünü bir örnekle göreceğiz. Şekil 3.12'de verilen H matrisinin oluşturduğu LDPC kodu için Tanner grafiği Şekil 3.13'de verilmiştir.



Şekil 3.13: Şekil 3.12'de verilen H matrisine ait Tanner grafiği gösterimi. 6'lı çevrim koyu ile gösterilmiştir.

Örnek olarak kod kelimesi 0 0 1 0 1 1 gönderilsin. Kanalın AWGN ve $E_b/N_0 = 1.25$ olsun. Alınan sinyal ise şöyle kabul edilsin: $y = -0.1, 0.5, -0.8, 1.0, -0.7, 0.5$. Alınan sinyale sert karar verme yöntemi uygulanırsa 1 ve 6 numaralı bitlerin hatalı olarak çözüldüğü görülür. Ancak topla-çarp algoritmasına devam edilirse bu hatalar düzeltilebilir. Şekil 3.14'de topla-çarp algoritmasının operasyonları gösterilmiştir [13]. Bu çözme işlemi 3 döngü sonunda durmaktadır. Topla-çarp algoritmasının durma işleminin iki önemli faydası vardır; ilki, yakınsamada hata olduğunda bu tespit edilir ve ikincisi, fazla döngülerle doğru bir çözümün bulunması sağlanır.

Döngü 1

$$R = \begin{bmatrix} -0.5000 & 2.5000 & -4.0000 & 5.0000 & -3.5000 & 2.5000 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \text{ hard karar verme}$$

$$E = \begin{bmatrix} 2.4217 & -0.4930 & . & -0.4217 & . & . \\ . & 3.0265 & -2.1892 & . & -2.3001 & . \\ -2.1892 & . & . & . & -0.4217 & 0.4696 \\ . & . & 2.4217 & -2.3001 & . & -3.6869 \end{bmatrix}$$

$$L = \begin{bmatrix} -0.2676 & 5.0334 & -3.7676 & 2.2783 & -6.2217 & -0.7173 \end{bmatrix}$$

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

$$Hz^T = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}^T \Rightarrow \text{Devam}$$

$$L = \begin{bmatrix} -2.6892 & 5.5265 & . & 2.6999 & . & . \\ . & 2.0070 & -1.5783 & . & -3.9217 & . \\ 1.9217 & . & . & . & -5.8001 & -1.1869 \\ . & . & -6.1892 & 4.5783 & . & 2.9696 \end{bmatrix}$$

Döngü 2

$$E = \begin{bmatrix} 2.6426 & -2.0060 & . & -2.6326 & . & . \\ . & 1.4907 & -1.8721 & . & -1.1041 & . \\ 1.1779 & . & . & . & -0.8388 & -1.9016 \\ . & . & 2.7877 & -2.9305 & . & -4.3963 \end{bmatrix}$$

$$L = \begin{bmatrix} 3.3206 & 1.9848 & -3.0845 & -0.5630 & -5.4429 & -3.7979 \end{bmatrix}$$

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

$$Hz^T = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}^T \Rightarrow \text{Devam}$$

$$L = \begin{bmatrix} 0.6779 & 3.9907 & . & 2.0695 & . & . \\ . & 0.4940 & -1.2123 & . & -4.3388 & . \\ 2.1426 & . & . & . & -4.6041 & -1.8963 \\ . & . & -5.8721 & 2.3674 & . & 0.5984 \end{bmatrix}$$

Döngü 3

$$E = \begin{bmatrix} 1.9352 & 0.5180 & . & 0.6515 & . & . \\ . & 1.1733 & -0.4808 & . & -0.2637 & . \\ 1.8332 & . & . & . & -1.3362 & -2.0620 \\ . & . & 0.4912 & -0.5948 & . & -2.3381 \end{bmatrix}$$

$$L = \begin{bmatrix} 3.2684 & 4.1912 & -3.9896 & 5.0567 & -5.0999 & -1.9001 \end{bmatrix}$$

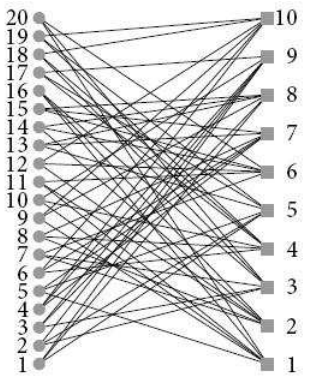
$$z = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$Hz^T = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T \Rightarrow \text{Bitir}$$

Şekil 3.14: Topla-çarp algoritmasının Şekil 3.12'de verilen kod için işleyişi [13]

3.5 DÜZENSİZ LDPC KODLARI

LDPC kodları ve döngülü çözüm algoritması Bölüm 3.1 ve 3.4'de anlatılmıştır. Bölüm 3.1'de düzenli LDPC kodlarına ait eşitlik kontrol matrisi şu şekilde tanımlanmıştı: (1) her satır ρ tane 1'den oluşmaktadır,(2) her sütun γ tane 1'den oluşmaktadır, (3) İki sütun arasındaki ortak 1'lerin sayısı λ, 1'den büyük değildir; yani λ=1 veya λ=0'dır, (4) ρ ve γ değerleri, kod uzunluğu ve H matrisinin satır sayısı ile karşılaştırıldığında olabildiğince küçük olmalıdır [9]. Bu tanıma uymayan ve eşitlik kontrol matrislerinin satır ve sütunları kendi içlerinde eşit sayıda 1'den oluşmayan kodlara *düzensiz LDPC kodlar* denir. Bu bölümde düzensiz LDPC kodlar ile ilgili bazı tanımlar verilmektedir. Bu tanımlar, Bölüm 3.1'den farklı olarak [22] nolu referanstaki gösterim takip edilerek verilmiştir.



Şekil 3.15: Düzenli LDPC koda ait Tanner grafiği

Şekil 3.15'deki Tanner grafiğinde, her değişken düğümü 3 ve her bir kontrol düğümü 6 derecesine sahiptir. Böyle bir kod (3,6)-düzenli LDPC kod olarak adlandırılır. Daha genel ifadeyle, (l, r)-düzenli LDPC kodlar her bir değişken düğümü l ve her bir kontrol düğümü r derecesine sahip doğrusal kodlardır. (l, r)-düzenli LDPC koda ait Tanner grafiğindeki kenarların sayısı ln kadardır, burada n kodun uzunluğudur. n arttığında, Tanner grafiğinde yer alan kenarların sayısı n ile birlikte doğrusal olarak büyüyecektir.

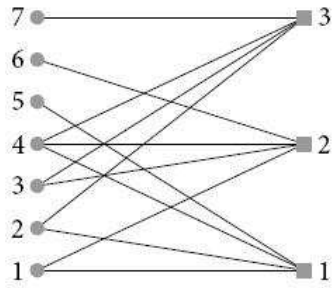
LDPC kodlarının performansı düğümlerin derecelerinin farklı farklı yapılmasıyla yani kodların düzensiz olmasına izin verilmesiyle geliştirilebilir [22].

Örnek 3.7:

Düzensiz bir LDPC koda ait H matrisi Şekil 3.16'da, buna karşılık gelen Tanner grafiği

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Şekil 3.16: Düzensiz LDPC koda ait eşitlik kontrol matrisi



Şekil 3.17: Düzensiz LDPC koda ait Tanner grafiği

ise Şekil 3.17'de verilmiştir. Bu grafikte üç kontrol düğümü de 4 derecesine sahiptir. Bu grafikte, derecesi 1 olan 3 adet değişken düğümü, derecesi 2 olan 3 adet değişken düğümü ve derecesi 3 olan 1 adet değişken düğümü bulunmaktadır.

Düzensiz bir LDPC kodunun n uzunluğuna sahip olduğunu ve i dereceli değişken düğümlerinin sayısının Λ_i olduğunu kabul edelim. Böylelikle $\Sigma_i \Lambda_i = n$ olur. Aynı şekilde derecesi i olan kontrol düğümlerinin sayısı P_i olsun. Böylelikle $\Sigma_i P_i = n\overline{r}$ olur, burada r kodun oranını (uzunluk eksi kısıtların sayısının uzunluğa oranı) ve $\overline{r}$ ise $1 - r$'nin kısa yoldan ifade edilişini göstermektedir. İlave olarak, kenar sayılarının birbiriyle tutması gerektiğinden $\Sigma_i i\Lambda_i = \Sigma_i i P_i$ eşitliği yazılabilir. Eşitlik 3.15 ve 3.16'da $\Lambda(x)$ ve $P(x)$ tanımlamaları yapılmıştır:

$$\Lambda(x) = \sum_{i=1}^{l_{max}} \Lambda_i x^i \tag{3.15}$$

$$P(x) = \sum_{i=1}^{r_{max}} P_i x^i \tag{3.16}$$

Burada $\Lambda(x)$ ve $P(x)$ sıfır etrafında negatif olmayan katsayılara sahip olan polinomlardır.

Bu polinomların katsayıları çeşitli derecedeki düğümlerin sayısına eşittir. Bu tanımlardan, aşağıdaki eşitlikler çıkarılabilir:

$$\Lambda(1) = n \qquad P(1) = n\overline{r} \tag{3.17}$$

$$r = 1 - \frac{P(1)}{\Lambda(1)} \tag{3.18}$$

$$\Lambda^{'}(1) = P^{'}(1) \tag{3.19}$$

Eşitlik 3.19'da ($'$), birinci dereceden türev anlamına gelmektedir.

Λ ve P, değişken ve kontrol derece dağılımları olarak adlandırılır. Bazı durumlarda bu dağılımların, aşağıda tanımlanan normalize edilmiş derece dağılımlarını kullanmak daha uygundur [22].

$$L(x) = \frac{\Lambda(x)}{\Lambda(1)} \tag{3.20}$$

$$R(x) = \frac{P(x)}{P(1)} \tag{3.21}$$

Örnek 3.8: Şekil 3.17'de tanımlanan kod için dağılımlar aşağıdaki şekilde yazılır:

$$\begin{aligned}
\Lambda(x) &= 3x + 3x^2 + x^3 \\
L(x) &= \frac{3}{7}x + \frac{3}{7}x^2 + \frac{1}{7}x^3 \\
P(x) &= 3x^4 \\
R(x) &= x^4
\end{aligned}$$

Bir (Λ, P) derece dağılımı ikilisine karşılık gelen birden fazla Tanner grafiği vardır. (Λ, P) LDPC kodundaki her bir grafik $\Lambda(1)$ değişken düğümüne ve $P(1)$ kontrol düğümüne sahiptir.

Analizlerin daha kolay yapılması için aşağıdaki tanımlamalar yapılır:

$$\lambda(x) = \sum_i \lambda_i x^{i-1} = \frac{\Lambda^{'}(x)}{\Lambda^{'}(1)} = \frac{L^{'}(x)}{L^{'}(1)} \tag{3.22}$$

$$\rho(x) = \sum_i p_i x^{i-1} = \frac{P^{'}(x)}{P^{'}(1)} = \frac{R^{'}(x)}{R^{'}(1)} \tag{3.23}$$

λ ve ρ polinomları sıfır etrafında negatif olmayan katsayılara sahiptir. Bu λ ve ρ, değişken ve kontrol derece dağılımları olarak adlandırılır. Tanımlardan yola çıkarak,

$$\frac{\Lambda(x)}{n} = L(x) = \frac{\int_0^x \lambda(z)dz}{\int_0^1 \lambda(z)dz} \tag{3.24}$$

$$\frac{P(x)}{n\overline{r}} = R(x) = \frac{\int_0^x \rho(z)dz}{\int_0^1 \rho(z)dz} \tag{3.25}$$

Ortalama değişken (l_{avg}) ve kontrol (r_{avg}) düğüm dereceleri şu şekilde ifade edilir:

$$l_{avg} = L^{'}(x) = \frac{1}{\int_0^1 \lambda(x)dx} \tag{3.26}$$

$$r_{avg} = R^{'}(x) = \frac{1}{\int_0^1 \rho(x)dx} \tag{3.27}$$

Kod oranı, kodun bütün kısıtları doğrusal olarak birbirinden bağımsız olduğundaki oran olarak tanımlanır ve aşağıdaki şekilde ifade edilir:

$$r(\lambda, \rho) = 1 - \frac{l_{avg}}{r_{avg}} = 1 - \frac{L^{'}(1)}{R^{'}(1)} = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx} \tag{3.28}$$

Örnek 3.9: Şekil 3.17'de verilen kod ele elınırsa,

$$\begin{aligned} \lambda(x) &= \frac{1}{4} + \frac{1}{2}x + \frac{1}{4}x^2 \\ \rho(x) &= x^3 \end{aligned}$$

Örnek 3.10: (Λ, P) ikilisi aşağıdaki şekilde düşünülürse,

$$\begin{aligned} \Lambda(x) &= 613x^2 + 202x^3 + 57x^4 + 84x^7 + 44x^8 \\ P(x) &= 500x^6 \end{aligned}$$

ve

$$\Lambda(1) = 1000 \qquad P(1) = 500 \qquad \Lambda^{'}(1) = P^{'}(1) = 3000$$

Bu ikili dağılımlara göre kod uzunluğu 1000 ve dizayn oranı 0.5 alınırsa, bu dağılımlar λ ve ρ dağılımlarına çevrildiğinde,

$$\begin{aligned} \lambda(x) &= \frac{1226}{3000}x + \frac{606}{3000}x^2 + \frac{228}{3000}x^3 + \frac{588}{3000}x^6 + \frac{352}{3000}x^7 \\ \rho(x) &= x^5 \end{aligned}$$

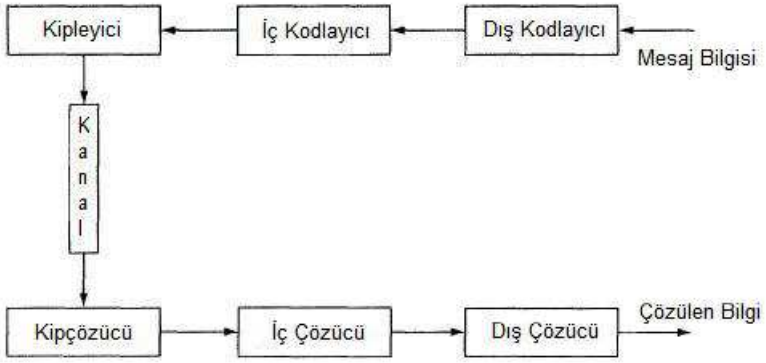
olur.

(Λ, P), (n, L, R) ve (n, λ, ρ) eşdeğer bilgiler içermektedir, bu sebepten bu perspektifler arasında serbestçe geçiş yapılabilir [22]. Bu tezde standart gösterim olan (n, λ, ρ) gösterimi kullanılacaktır.

4. DÖNGÜSEL ÇÖZÜMLEMELİ SERİ BİRLEŞTİRİLMİŞ KODLAR

4.1 SERİ BİRLEŞTİRİLMİŞ KODLAR

Kodlama teorisinde kullanılan kodları seri birleştirmek için birçok iyi yöntem bulunmaktadır. Buradaki amaç, uzun kodları daha basit çözücü bileşenleri ile yaratmaktır. Ortak kullanılan yöntemlerden biri Şekil 4.1'de gösterildiği gibi iki blok kodu seri olarak birbirine bağlamak şeklindedir. Bilgiye ilk uygulanan kod dışsal kod ve ikinci uygulanan kod iç koddur ve bu kod kanal için dizayn edilir.



Şekil 4.1: Seri birleştirilmiş kodlar

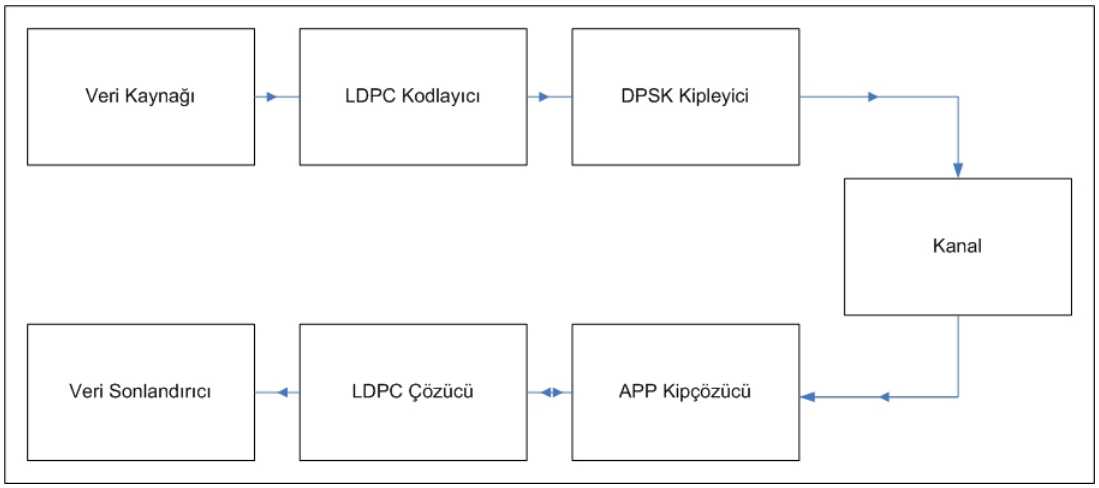
Makul birleştirilmiş kodlar üretmek zor bir iştir. Toplam kod oranı sabit tutulmuş bir kod şeması içerisinde, bir kodun oranını düşürmek ve kodun gücünü arttırmak veya oranı arttırıp kodun hata düzeltme gücünü azaltmak mümkündür. Burada aradaki dengeyi en iyi seviyede tutmak kolay değildir. İç kod, kanal üzerinde makul bir seviyede hata kontrol kabiliyetine sahip olmalıdır. Eğer iç kod yeterince güçlü değilse veya kanalla uyumlu değilse, dışsal kodun performansını düşürmenin yanı sıra hata oranını da yükseltir. Bu sebepten, içsel kodlar en iyi performansa sahip olacak şekilde seçilirler [24].

Seri olarak birleştirilmiş kodlar döngüsel çözümleme için uygundur. Bu tez çalışmasında, LDPC kodları ile ayrımsal kiplenim birleştirilecektir. Bu iki öğenin birleştirilmesi ile elde edilen kod kelimesinin AWGN kanaldan geçmiş hali, ileri-geri algoritması ve LDPC çözümleyicinin birleşiminden oluşan döngüsel alıcıda çözülecektir. Bu çözümleme işlemi, yumuşak bilgilerle yapılacaktır. Eğer çözümleme sert karar verme yöntemleri ile yapılsaydı (yani Bölüm 2'de anlatılan DPSK için kipçözmesi, sert yöntemlerle gerçekleştirilseydi), yumuşak karar vermenin sert karar vermeye olan

performans üstünlüğünden dolayı, burada elde edilen sonuçlardan daha kötü sonuçlar elde edilirdi. Ayrıca alıcıdaki döngüsellik, sert karar verme çözümlemesi ile mümkün olmazdı. Bu sebeplerden alıcıda, yumuşak bilgiler kullanılmış ve seri birleştirilmiş kodlara ait bu haberleşme sisteminin detaylı bilgileri ve bu sisteme ait simülasyon sonuçları ilerleyen bölümlerde verilmiştir.

4.2 HABERLEŞME SİSTEM MODELİ

Bu tezde ele alınan sistemin (verici-kanal-alıcı) genel blok şeması Şekil 4.2'de verilmiştir:



Şekil 4.2: Sistem blok şeması

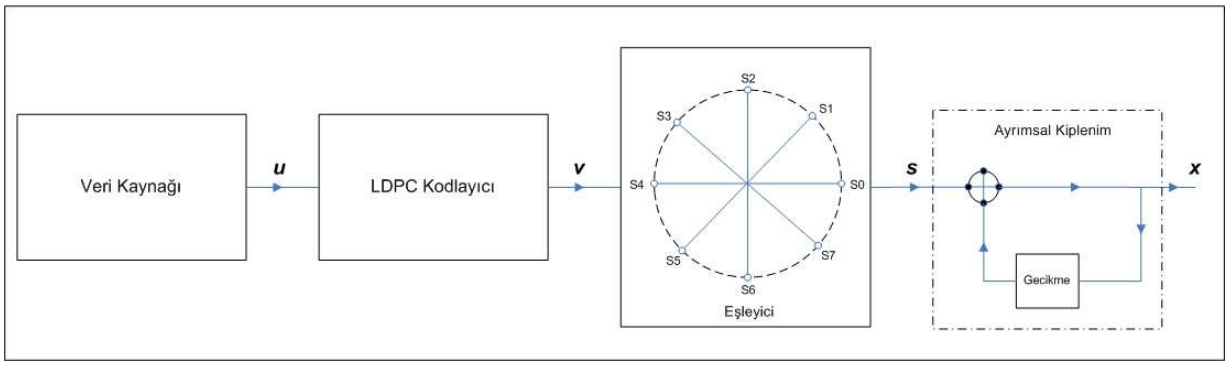
Sistem blok şemasında görülen veri kaynağı, kaç adet olduğu daha önceden belirlenen rastgele bit dizileri üretmektedir. Bu oluşturulan bit dizileri, LDPC kodlayıcı ile kodlanarak fazlalık olan bitler kaynakta üretilen bitlere eklenmekte ve böylelikle hata kontrolü sağlanmaktadır. Kodlanan bit dizisi DPSK kipleyici ile kanala verilmek üzere sembollere dönüştürülür. Kullanılan DPSK kipleyici 8-DPSK olduğu için 3-bitlik kod dizileri üzerinde çalışmaktadır ve bu 3-bitlik dizilerden birer sembol oluşmaktadır. Oluşturulan sembol dizileri AWGN kanalına verilir. Bu kanaldan çıkan sembol dizileri 8-DPSK yumuşak çözücüsüne (APP kipçözücü) girer. Buradan çıkan sembol olasılıkları, LLR değerlerine döndürüldükten sonra, LDPC çözümleyicisine girer. Bu LLR girdileri burada belirli sayıda döngüden geçtikten sonra ortaya çıkan yeni LLR değerleri bütün alıcıda döngüselliği sağlamak amacıyla tekrar 8-DPSK yumuşak çözümleyicisine gönderilir. LDPC çözümleyicisinde çözümlenen bit dizisi kod kısıtlarını sağlıyorsa,

bu durumda döngü biter ve veri sonlandırılır. Eğer veri önceden belirlenmiş sayıda döngüde sonlandırılmazsa, otomatik olarak en son döngüde bulunan bit dizisi veri sonlandırıcısına gönderilir [8].

Yukarıda temelleriyle değinilen sisteme ait verici, kanal ve alıcı bölümleri ilerleyen bölümlerde daha detaylı bir şekilde açıklanacaktır.

4.2.1 Verici

Vericiye ait blok diyagramı aşağıdaki şekilde verilmiştir:



Şekil 4.3: Verici blok diyagramı

Şekil 4.3'te yer alan blokları tek tek ele almak gerekirse:

- Veri Kaynağı: Veri kaynağı, önceden sayısı belirlenmiş mesaj uzunluğu kadar bit dizisi üretmektedir. Veri kaynağının rastgeleliği tekdüzedir. 0 ve 1 değerlerini tekdüze şekilde üretmektedir. Üretilen bit dizisinin gösterimi $u = u_1, u_2, ..., u_k$ ile yapılmaktadır.

- LDPC Kodlayıcı: Kodlanmış bit dizisi $v = uG$ ile bulunur ve $v = v_1, v_2, ..., v_n$ şeklinde gösterilir ve n, LDPC kodunun uzunluğunu temsil etmektedir. G ise oluşturma matrisini sembolize eder.

- Eşleyici: Eşleyici, LDPC koddaki bit dizilerinin üçerli şekilde sembollere eşlenmesi işini görür. Bit dizilerinin üçerli olmasının sebebi, eşleyicinin 8 adet sembole eşleme yapabiliyor olmasıdır. 8 adet sembol, eşleyici üzerinde karmaşık sayı düzlemine yerleştirilir. Çizelge 4.1'de bit dizilerinin hangi sembole denk geldiği gösterilmiştir.

Sembol	İkili Gösterim	Gerçel Değer	Sanal Değer
s_0	000	0	0
s_1	001	0.707	0.707
s_2	010	0	1
s_3	011	-0.707	0.707
s_4	100	-1	0
s_5	101	-0.707	-0.707
s_6	110	0	-1
s_7	111	0.707	-0.707

Çizelge 4.1: Sembollerin ikili gösterimi ve karmaşık düzlemdeki yerleşimi

Bu eşleyicide her bir sembolün enerjisi 1'dir. Bu sebepten her bitin enerjisi şu şekilde ifade edilir:

$$E_s = 1 = 3E_b \Rightarrow E_b = 1/3 \tag{4.1}$$

Eşlenmiş sembol dizisinin gösterimi $s = s_1, s_2, ..., s_{n/3}$ şeklinde yapılır.

- Ayrımsal Kiplenim: Bununla ilgili detaylı bilgi Bölüm 2.1'de verilmiştir. Ayrımsal kiplenim sonucunda oluşan sembol dizisinin gösterimi $x = x_0, x_1, ..., x_{n/3}$ şeklinde yapılır.

  Burada x_0 sembolü referans sembolüdür ve alıcı tarafındaki 8-DPSK APP yumuşak çözücüsündeki BCJR algoritmasının başlangıç değeri olarak kullanılacağı için değeri s_0 olarak alınmıştır. x dizisi aşağıdaki formüle bağlı kalınarak hesaplanır:

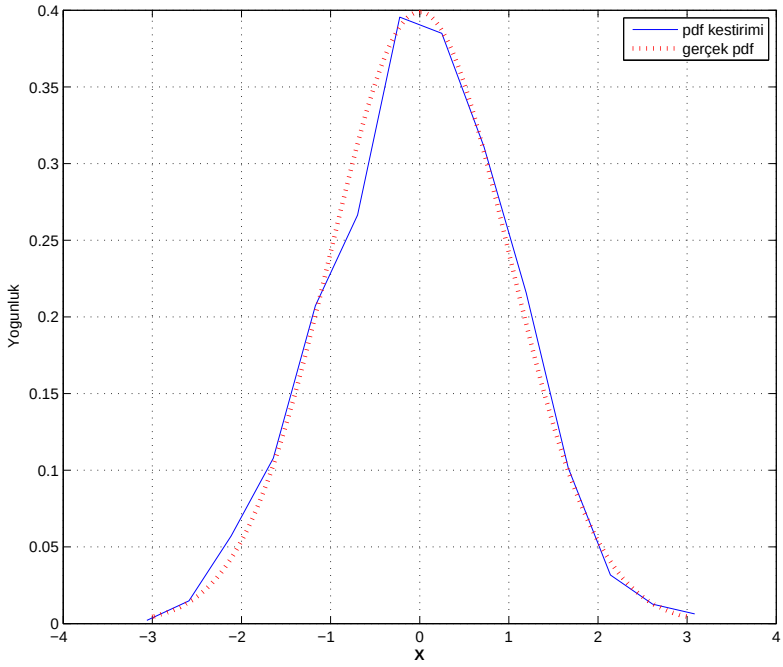
$$x_t = x_{t-1} \bigoplus s_t \qquad mod(8) \tag{4.2}$$

  t burada zaman indeksini belirtmektedir. x dizisinin uzunluğu, s dizisinin uzunluğundan 1 fazladır, bunun sebebi fazlalık olan referans sembolünün ayrımsal kodlamaya dahil edilmesidir.

4.2.2 Kanal

Bu tez çalışmasında kullanılan kanal, AWGN kanaldır. AWGN kanalın ürettiği rastgele Gaussian sayıların ortalaması sıfırdır ve standart sapması ise σ ile tanımlanır.

Bu σ'ya bağlı olarak üretilen Gaussian değerler çeşitlilik göstermektedir. Şekil 4.4, 4.5, 4.6'da E_b/N_0'nun 0 dB, 5 dB ve 10 dB değerleri için simülasyonlarda kullanılan üreteç ile üretilen Gaussian değerler kullanılarak hazırlanan olasılık yoğunluk fonksiyonu gerçek dağılımlarla karşılaştırılarak verilmiştir.

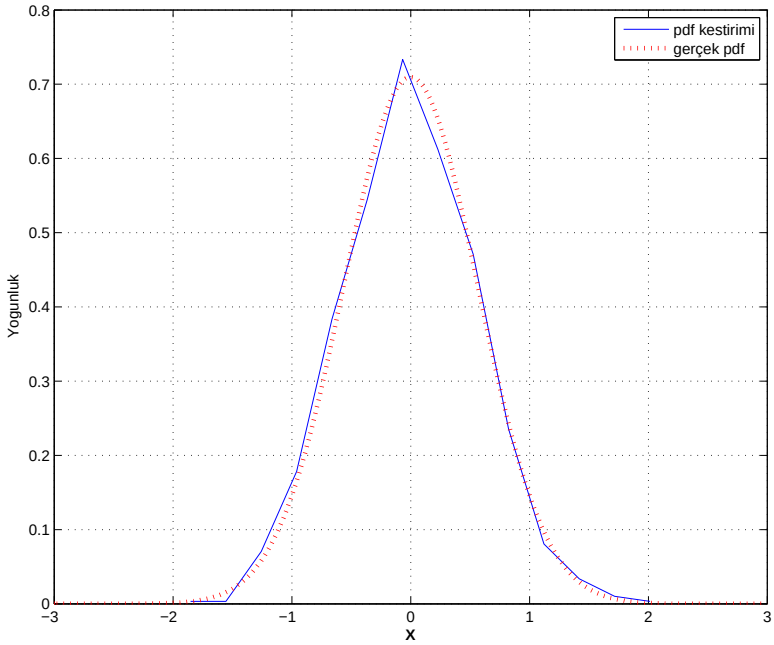


Şekil 4.4: SNR=0dB için Gaussian yoğunluk fonksiyonu

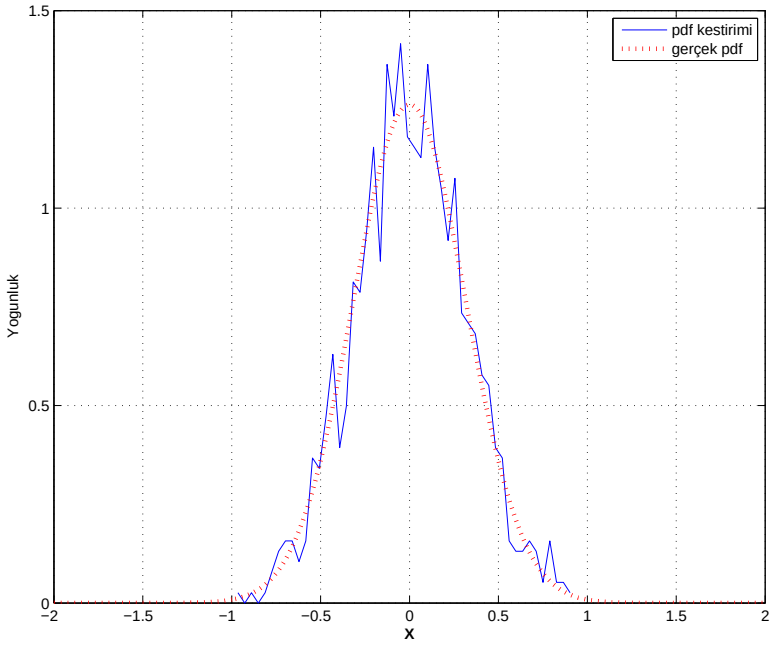
Şekil 4.4, 4.5, 4.6'da oluşturulan dağılımlar için kullanılan standart sapma değerleri aşağıdaki formülasyon kullanılarak hesaplanmıştır:

$$\begin{aligned} 10log_{10}\frac{E_b}{N_0} &= S \\ \frac{E_b}{N_0} &= 10^{\frac{S}{10}} \qquad (E_b = 1/3) \\ N_0 &= \frac{1}{3 \cdot 10^{S/10}} \qquad (\sigma^2 = N_0/2) \\ \sigma^2 &= \frac{1}{6 \cdot 10^{S/10}} \\ \sigma &= \sqrt{\frac{1}{6 \cdot 10^{S/10}}} \end{aligned} \tag{4.3}$$

Standart sapma değeri olan σ'ya uygun olarak üretilen Gaussian rastgele değişken değerleri, verici tarafından gönderilen karmaşık düzlem üzerinde gösterimi



Şekil 4.5: SNR=5dB için Gaussian yoğunluk fonksiyonu



Şekil 4.6: SNR=10dB için Gaussian yoğunluk fonksiyonu

yapılan sembol seviyeleri üzerine bindirilir:

$$\begin{aligned} x &= a + jb \\ y &= (a + n_1) + j(b + n_2) \end{aligned} \tag{4.4}$$

Burada,

a =Gönderilen sembolün karmaşık düzlem üzerindeki gerçel kısmı

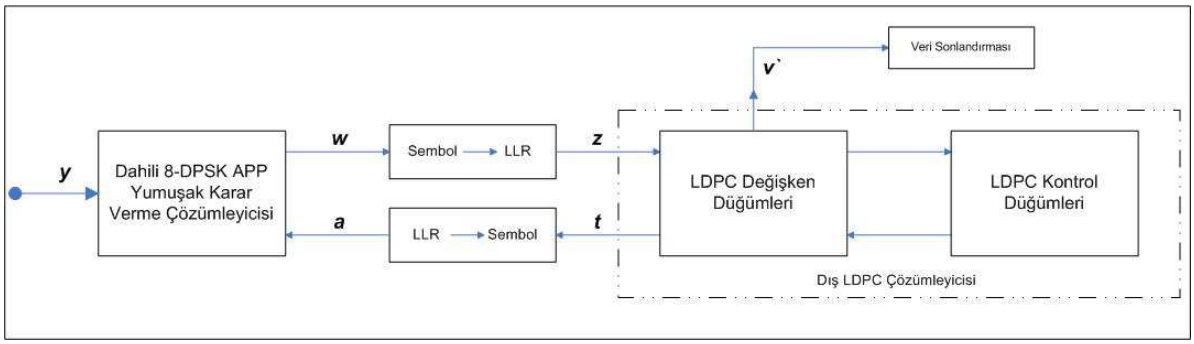
b =Gönderilen sembolün karmaşık düzlem üzerindeki imajiner sanal kısmı

n_1, n_2 =Gaussian rastgele değişken değerleri

Kanaldan çıkan y değerleri alıcı için giriş değerleri olur.

4.2.3 Alıcı

Alıcı tasarımına ait blok şeması Şekil 4.7'de verilmiştir.



Şekil 4.7: Alıcı blok şeması

Bu alıcı modelinde, dahili 8-DPSK APP yumuşak karar verme çözümleyicisi, sembol olasılıklarından LLR'a, LLR'dan sembol olasılıklarına çeviren bloklar ve LDPC çözmesine ait değişken ve kontrol düğüm blokları vardır. Bu blokların detayları ilerleyen satırlarda anlatılacaktır. Dahili 8-DPSK APP yumuşak karar verme çözümleyicisi, Bölüm 2.4'te anlatılan ileri-geri algoritmasını kullanmaktadır. Bölüm 4.2.1'de anlatıldığı üzere, kiplenim ve kipçözme sekiz olası sembol üzerinde yapılmaktadır. Modelleme yapılırken kullanılan ileri-geri algoritması Eşitlik 4.11'de verilmiştir [1]. Bu eşitlikleri incelemeden önce, Bölüm 2.4'te verilen ileri-geri algoritması ile bu eşitlikler arasındaki gösterim farkları ve Eşitlik 4.11'deki formüllerin nasıl çıkarıldığı aşağıdaki eşitliklerde gösterilmiştir:

$$\begin{aligned}\lambda_t(m) = p(S_t = m, Y_1^T) \rightarrow \lambda(s_i) &= p(s_i, y_1^N) \\ &= \alpha.p(s_i|y) \\ &= \alpha.\alpha(s_i).\beta(s_i) \end{aligned} \tag{4.5}$$

$$\begin{aligned}\gamma_t(m, m^{'}) = Prob(S_t = m, Y_t|S_{t-1} = m^{'}) \rightarrow \gamma(s_i, s_{i-1}) &= p(s_i, y_i|s_{i-1}) \\ &= p(y_i|s_i, s_{i-1})p(s_i|s_{i-1}) \\ &= p(y_i|s_i)p(s_i|s_{i-1}) \end{aligned} \tag{4.6}$$

$$\begin{aligned}\alpha_t(m) = \sum_{m^{'}=0}^{M-1} \alpha_{t-1}(m^{'})\gamma_t(m, m^{'}) \rightarrow \alpha(s_i) &= \sum_{s_{i-1}} \alpha(s_{i-1})p(y_i|s_i)p(s_i|s_{i-1}) \\ &= p(y_i|s_i) \sum_{s_{i-1}} \alpha(s_{i-1})p(s_i|s_{i-1}) \end{aligned} \tag{4.7}$$

$$\begin{aligned}\beta_t(m) = \sum_{m^{'}=0}^{M-1} \beta_{t+1}(m^{'})\gamma_{t+1}(m, m^{'}) \rightarrow \beta(s_i) &= \sum_{s_{i+1}} \beta(s_{i+1})p(y_{i+1}|s_{i+1})p(s_{i+1}|s_i) \\ \beta(s_{i-1}) &= \sum_{s_i} \beta(s_i)p(y_i|s_i)p(s_i|s_{i-1}) \end{aligned} \tag{4.8}$$

Bölüm 2.4'te anlatılan ileri-geri algoritmasında başlangıç değerleri, başlangıç ve bitiş durumlarının sıfır olduğu kabul edilerek yazılmıştı. Ancak modelleme için kullanılacak ileri-geri algoritmasında başlangıç ve bitiş durumları kesin olarak bilinemediği için, algoritma başlangıcı şu şekilde yapılır:

$$\alpha(s_1) = p(s_1, y_1^1) = p(s_1, y_1) = p(y_1|s_1)p(s_1) \tag{4.9}$$

$$\beta(s_N) = p(y_{N+1}^N|s_N) = p(y_{N+1}^N, s_N)/p(s_N) = p(s_N)/p(s_N) = 1 \tag{4.10}$$

Eşitlik 4.5, 4.6, 4.7, 4.8, 4.9 ve 4.10 takip edilerek ileri-geri algoritması şu şekilde özetlenir [1]:

$$
\begin{aligned}
\alpha(s_1) &= p(y_1|s_1)p(s_1) \\
\alpha(s_i) &= p(y_i|s_i)\sum_{s_{i-1}} p(s_i|s_{i-1})\alpha(s_{i-1}) \qquad i = 2, 3, ..., N \\
\beta(s_N) &= 1 \\
\beta(s_{i-1}) &= \sum_{s_i} p(y_i|s_i)p(s_i|s_{i-1})\beta(s_i) \qquad i = N-1, N-2, ..., 1 \\
p(s_i|y) &= \alpha.\alpha(s_i).\beta(s_i)
\end{aligned} \tag{4.11}
$$

Bu eşitliklerde α ileri yöndeki parametreleri, β ise geri yöndeki parametreleri temsil etmektedir. Denklem 4.11'de geçen α parametresi normalize etme sabiti olarak geçmektedir. Bu α parametresi $\alpha(s_i)$ parametreleri ile karıştırılmamalıdır. Pratikte, $\alpha(s_i)$ ve $\beta(s_i)$ değerleri sıfıra gittiğinden simüle edilirken sorunlar çıkmaktadır. Bu sebepten $\alpha(s_i)$ ve $\beta(s_i)$ değerleri toplamları 1'e eşit olacak şekilde kendi içlerinde normalize edilirler.

İleri-geri algoritmasının girişleri, kanal çıktısı y_1^Nve LDPC bloğundan gelen önsel bilgi, $p(s_1)$ ve $p(s_i|s_{i-1})$, $i = 2, 3, ..., N$'dir. Bu blok içerisinde ileri-geri algoritması kullanılarak, i numaralı sembole karşılık gelen her bir sembol için yani toplam sekiz adet olasılık değeri hesaplanır ve $p(x_i|y)$, bu bloğun çıktısı olarak tanımlanır. $p(x_i|y)$, semboller arası geçiş olasılıklarını tanımlamaktadır ve ileri- geri algoritmasının çıkışları olan $p(s_i|y)$ olasılıkları ile ilişkisi aşağıda verilmiştir:

$$
p(x_i(m)|y) = \sum_{k=0}^{7} p(s_{i-1}(k)|y)p(s_i(k+m)|y) \qquad m = 0, 1, ..., 7 \tag{4.12}
$$

Bu 8-DPSK APP çözümleyici ayrımsal kodlamayı, olasılık etki alanında yapılan hesaplamalarla tersine çevirir ve çıkışında, bu sembollere ait olasılıksal değerleri bir sonraki bloğa aktarır.

8-DPSK APP çözümleyicisinden çıkan sembol olasılıkları, bit LLR'larına çevrilir. Bu blok, sembol olasılıklarını ikili gösterim doğrultusunda LLR değerlerine çevirir. Her bir sembol 8-DPSK kiplenim gereği 3-bit ile gösterilmiştir. Dolayısıyla her bir sembol, LLR değerine döndürülürken üç ayrı LLR değerine denk gelir. Bir sembole karşılık gelen üç adet LLR şu şekilde hesaplanmaktadır:

$$
\begin{aligned}
LLR(b_2|x_i) &= log\frac{p(x_i=0|y)+p(x_i=1|y)+p(x_i=2|y)+p(x_i=3|y)}{p(x_i=4|y)+p(x_i=5|y)+p(x_i=6|y)+p(x_i=7|y)} \\
LLR(b_1|x_i) &= log\frac{p(x_i=0|y)+p(x_i=1|y)+p(x_i=4|y)+p(x_i=5|y)}{p(x_i=2|y)+p(x_i=3|y)+p(x_i=6|y)+p(x_i=7|y)} \\
LLR(b_0|x_i) &= log\frac{p(x_i=0|y)+p(x_i=2|y)+p(x_i=4|y)+p(x_i=6|y)}{p(x_i=1|y)+p(x_i=3|y)+p(x_i=5|y)+p(x_i=7|y)}
\end{aligned} \tag{4.13}
$$

Sembol olasılıklarından LLR'a çeviren bloktan çıkan bilgiler, LDPC çözücüye girer. LLR bilgileri ilk olarak çözücünün değişken düğümlerine aktarılır. Bu düğümlerden bilgiler H matrisi ile tanımlanmış ve bu değişken düğümlerine kenarlarla bağlı kontrol düğümlerine aktarılır ve döngüsellik başlar. Değişken ve kontrol düğümleri ile tanımlanan LDPC çözücüsü üzerinde kullanılan topla-çarp algoritması Bölüm 3.2.2'de anlatılmıştır ve bu sistemde bu algoritma değiştirilmeden kullanılmaktadır.

LDPC çözümleyici üzerinde N_{LDPC} parametresi tanımlanmıştır. LDPC çözümleyici kendi içerisinde döngüyü N_{LDPC} kere tekrarlar ve kodu doğru olarak çözdüyse çözme işlemini sonlandırır. Eğer çözme işlemi N_{LDPC} kadar döngü sonunda gerçekleştirilmezse, en son döngüde hesaplanan değişken düğümleri üzerindeki güncellenmiş LLR değerleri, alıcıda döngüselliği sağlamak üzere 8-DPSK APP çözücüsüne gönderilir.

LDPC bloğundan 8-DPSK APP çözümleyici bloğuna gönderilen LLR bilgileri, bu bloğa girmeden önce sembol olasılıklarına çevrilir. Bu çevrim, LLR'lardan elde edilen bit olasılıklarının çarpımıyla gerçekleştirilir. Bu sembol olasılıkları, 8-DPSK APP çözümleyicisi için artık önsel olasılıklarıdır. Bu önsel olasılıkları ileri-geri algoritması içerisindeki $p(s_i|s_{i-1})$ bilgilerini günceller. Elde edilen bu güncellenmiş $p(s_i|s_{i-1})$ bilgisi, 8-DPSK APP çözümleyicisi tarafından Eşitlik 4.11'de kullanılır ve yeni $p(s_i|y)$ bilgisi, LDPC çözümleyici tarafından kullanılmak üzere elde edilir. Döngüsel çözümleme yukarıda anlatıldığı şekilde, doğru kod çözümlenene veya N olarak ifade edilen alıcıdaki bloklar arasındaki toplam döngü sayısına ulaşılına kadar devam eder.

Böylelikle alıcı blokları arasında gidip gelen LLR bilgileri ile alıcının tamamında döngüsellik sağlanmış olur. Şekil 4.7'de verilen blok şemaya ait alıcı tasarımına yönelik yapılan simülasyon çalışmaları ve bu çalışmalara ait sonuçlar ilerleyen bölümlerde verilmiştir.

4.3 SİMÜLASYON SONUÇLARI

Bölüm 4.2'de anlatılan haberleşme sistemine ait simülasyon sonuçları bu bölümde verilmiştir. Bu paragraflarda SNR'a karşılık BER grafikleri ile sistemin performansı gözlenecektir. Sistem içerisindeki kullanılan parametrelerin değişik değerleri için bu grafikler çizdirilmiştir. Bu değişen parametreler ve sistem performansına yönelik yorumlar BER-SNR şekillerini takip eden satırlarda verilmiştir.

İlk olarak Bölüm 3'te anlatılan düzenli LDPC kodlara ait sistem performansı gözlemlenecektir. Sistem oluşturulurken kullanılan değişken parametreler Çizelge 4.2'de verilmiştir. Sisteme ait BER-SNR grafiği Şekil 4.8'de verilmiştir:

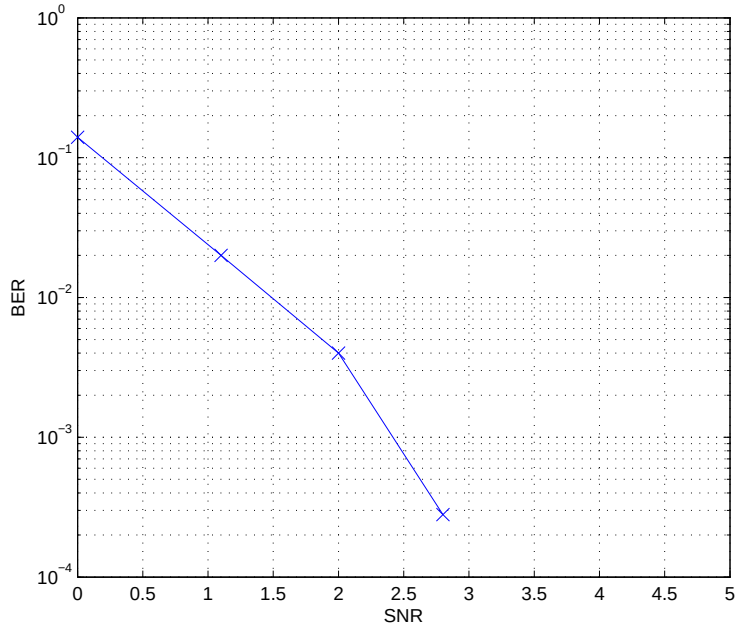
Parametre	Değeri
Mesaj Uzunluğu	300
Kod Uzunluğu	600
Kod Oranı	0.5
N	10
N_{LDPC}	5
$\lambda(x)$	x^2
$\rho(x)$	x^5

Çizelge 4.2: Şekil 4.8 için oluşturulan sisteme ait parametre değerleri

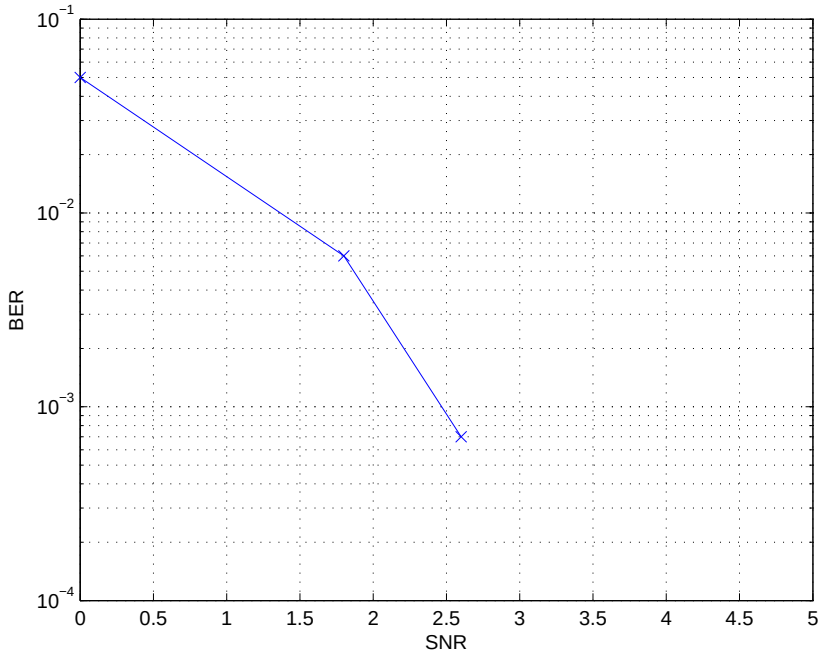
Çizelge 4.3'teki parametreler ile tanımlı sistem için hem alıcıda hem LDPC bloğundaki *döngü sayıları* artırılmış ve bu durumun sistemin performansına olan etkileri BER-SNR grafiği yardımıyla gözlenmiştir. Bu tanımlanan sisteme ait BER-SNR grafiği Şekil 4.9'da verilmiştir.

Parametre	Değeri
Mesaj Uzunluğu	300
Kod Uzunluğu	600
Kod Oranı	0.5
N	20
N_{LDPC}	10
$\lambda(x)$	x^2
$\rho(x)$	x^5

Çizelge 4.3: Şekil 4.9 için oluşturulan sisteme ait parametre değerleri



Şekil 4.8: Kod Uzunluğu=600 için oluşturulan düzenli LDPC kodlara ait BER-SNR grafiği



Şekil 4.9: Döngü sayıları değiştirilerek hazırlanan düzenli LDPC kodlara ait BER-SNR grafiği

Şekil 4.9 ve Şekil 4.8 karşılaştırılmasından açıkça görüldüğü üzere döngü sayısını artırmak sistemin performansını pozitif yönde etkilemektedir. Performansları karşılaştırmak amacıyla BER-SNR eğrilerinin birleştirilmiş grafiği Şekil 5.21'de verilmiştir. Döngü sayısını artırmak yaklaşık 0.2 dB'lik bir performans artımına sebep olmuştur. Ancak döngü sayısını artırmak özellikle düşük SNR seviyelerinde zaman etkinliğini düşürmektedir. Bundan sonra oluşturulan sistemlere ait performans analizleri için $N = 10$ (alıcıdaki döngü sayısı) ve $N_{LDPC} = 20$ (LDPC blokları arasındaki döngü sayısı) değerleri kullanılmıştır.

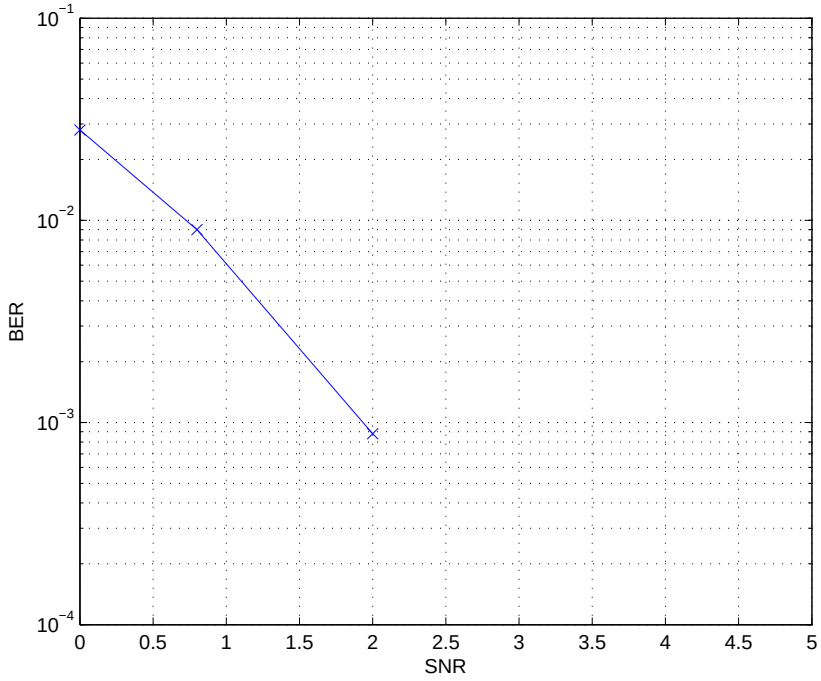
Çizelge 4.4'teki parametrelerde bundan önce oluşturulan sistemlerden farklı olarak kullanılan *kodun uzunluğu* artırılmış ve bu kodun BER-SNR grafiği Şekil 4.10'da verilmiştir.

Parametre	Değeri
Mesaj Uzunluğu	600
Kod Uzunluğu	1200
Kod Oranı	0.5
N	20
N_{LDPC}	10
$\lambda(x)$	x^2
$\rho(x)$	x^5

Çizelge 4.4: Şekil 4.10 için oluşturulan sisteme ait parametre değerleri

Şekil 4.10'daki grafikte, 1200 bit uzunluğunda oluşturulan düzenli LDPC kod için sistem performansı görülmektedir. Bu BER-SNR grafiği bir önceki grafikle karşılaştırıldığında sistemin daha iyi bir performansa sahip olduğunu söylemek mümkündür. Kod uzunluğu hariç diğer tüm parametrelerin aynı olduğu iki sistem arasındaki performans farkını yaratan neden Shannon'ın da açıkladığı gibi kod uzunluğudur. Shannon sonsuz uzunlukta bir kodla hatasız iletişim yapılacağını öngörmüştür. Bu yapılan çalışmada da, kullanılan LDPC kodunun uzunluğunu artırmanın performans üzerinde iyileştirici etki yaptığı görülmüştür. Kod uzunluğunu artırmanın sistemin performansına pozitif yönde etkileri olduğu gibi, alıcının karmaşıklığı, bant genişliği ve çözümleme zamanı açısından da negatif etkileri bulunmaktadır.

İleri-geri algoritmasının bir kere kullanıldığı sistem modeline ait performans

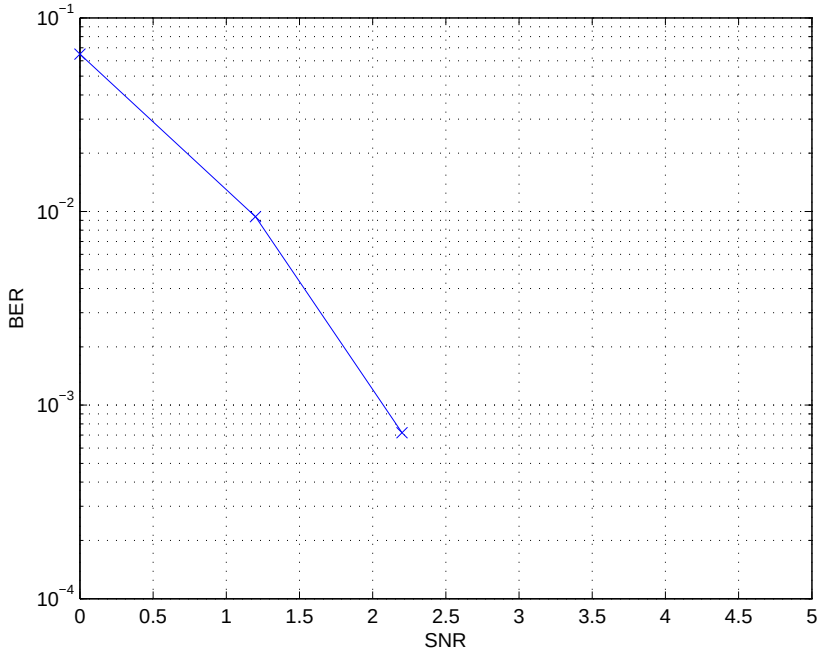


Şekil 4.10: Kod Uzunluğu=1200 için oluşturulan düzenli LDPC kodlara ait BER-SNR grafiği

analizleri de incelenmiştir. Alıcı kanaldan gelen bilgiyi aldıktan sonra sadece bir kere ileri-geri algoritması uygulayarak verileri LDPC bloğuna gönderir. LDPC bu verileri kendi düğümleri arasında topla-çarp algoritmasına uygun olarak çözümler. Böylelikle alıcıda döngüsellik sadece LDPC bloğunda olur. Bu tip bir sistemde kullanılan diğer parametreler Çizelge 4.5'te sıralanmıştır ve sistemin BER-SNR grafiği 4.11'de verilmiştir.

Parametre	Değeri
Mesaj Uzunluğu	600
Kod Uzunluğu	1200
Kod Oranı	0.5
N	1
N_{LDPC}	40
$\lambda(x)$	x^2
$\rho(x)$	x^5

Çizelge 4.5: Şekil 4.11 için oluşturulan sisteme ait parametre değerleri



Şekil 4.11: Bir kere ayrımsal çözümleme kullanılan düzenli LDPC kodlar için BER-SNR grafiği

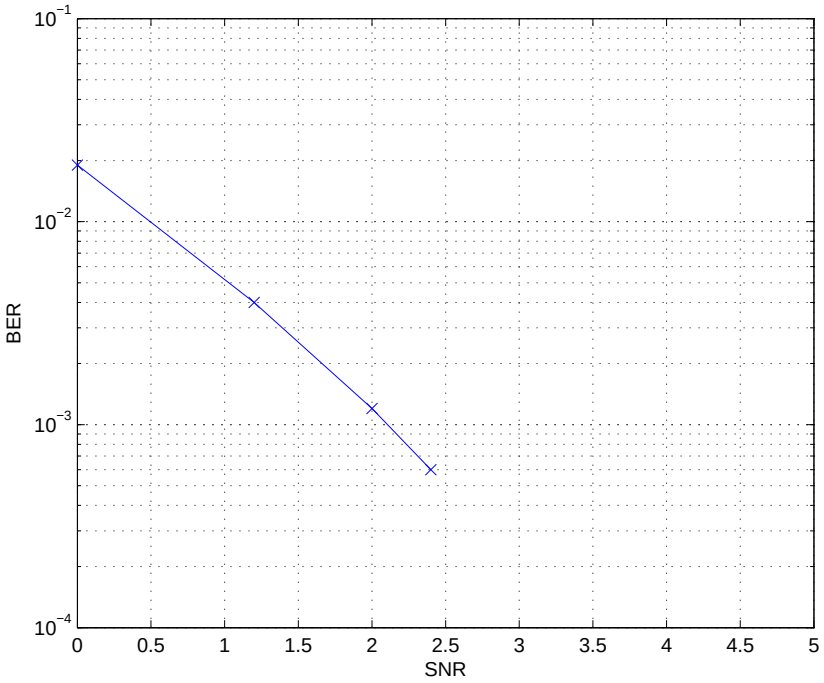
Şekil 4.11'de BER-SNR grafiği bulunan sisteme ait gözlemler ve yorumlar şöyledir: LDPC kodları güçlü kodlar oldukları için sadece bir kere kipçözülen veriyi (döngü sayısı da yüksek tutularak) iyi bir performansla çözmek mümkündür. Ancak döngüselliği bütün alıcıda sağlamanın düzenli kodlar için dahi daha iyi performans verdiği açıktır. Bunun yanı sıra Bölüm 5'te anlatılacağı üzere ayrımsal çözücü ile LDPC blokları arasında eniyileme yapıldıktan sonra yani bütün alıcıda eniyileme yapıldıktan sonra döngüselliği tüm alıcıda sağlamanın daha iyi bir performans vermesi beklenmektedir.

Bölüm 3.5'te tanımlanan düzensiz LDPC kodların kullanılması ile oluşturulan haberleşme sistemine ait performans analizleri yapılmıştır. Çizelge 4.6'da parametreleri verilen simülasyon, düzensiz LDPC kodların ayrımsal kiplenim ile birleştirilmesinden oluşan sisteme aittir ve bu parametrelerle oluşturulan sistemin performans grafiği Şekil 4.12'de gösterilmiştir.

Şekil 4.12'de eniyileme yapılmamış düzensiz LDPC kodlara ait performans grafiği verilmiştir. Bu sisteme ait BER-SNR grafiği Şekil 4.10'daki düzenli LDPC kodlar için çizdirilen BER-SNR grafiği ile kıyaslandığında, düzenli LDPC kodların yüksek

Parametre	Değeri
Mesaj Uzunluğu	600
Kod Uzunluğu	1200
Kod Oranı	0.5
N	20
N_{LDPC}	10
$\Lambda(x)$	$1000x^2 + 50x^3 + 50x^8 + 100x^{15}$
P(x)	$110x^3 + 50x^4 + 440x^8$

Çizelge 4.6: Şekil 4.12 için oluşturulan sisteme ait parametre değerleri



Şekil 4.12: Eniyileme yapılmamış düzensiz LDPC koda ait BER-SNR grafiği

SNR'larda daha iyi sonuçlar verdiği görülmektedir. Ancak şunu not etmek gerekir ki, Şekil 4.12'de görülen BER-SNR grafiğinin çizdirilmesinde kullanılan düzensiz LDPC kodlara alıcı için eniyileme yapılmamıştır.

Bölüm 3.5'te değinildiği üzere düzensiz LDPC kodlar $\lambda(x)$ ve $\rho(x)$ dağılımları ile tanımlanmaktadır. Aynı $\lambda(x)$ ve $\rho(x)$ dağılımları ile oluşturulan kodlar arasında farklılık olabilir. Çünkü $\lambda(x)$ ve $\rho(x)$ dağılımları, H matrisinin satır ve sütunlarında kaç adet 1 olacağının sayısını belirtmektedir ancak bu 1'lerin nereye yerleştirileceğini göstermemektedir. Bu sebepten aynı dağılımlarla farklı H matrisleri dolayısıyla farklı LDPC

kodlar elde etmek mümkündür. Bunun yanı sıra bu dağılımları kendimiz kod oranına göre belirlediğimiz için bu kodlara eniyileme yapmak mümkündür [8]. Bölüm 5'te düzensiz LDPC kodlara nasıl eniyileme yapılacağı anlatılmış ve bu eniyileme çalışması sonucunda oluşturulan bir düzensiz LDPC kodun performansı incelenmiş ve bu düzensiz LDPC kodunun performansı bu bölümde elde edilen sonuçlar ile karşılaştırılmıştır.

5. EXIT GRAFİĞİ ANALİZLERİ

Bölüm 4'te tasarlanan haberleşme sistemine ait çeşitli BER-SNR grafikleri ilgili bölümde verilmişti. Bu bölümde ise bu sistem için eniyileme çalışmaları yapılacaktır. En iyi düzensiz LDPC dağılımları elde etmek için EXIT grafiği metodu kullanılacaktır.

EXIT grafik metodu, iyi döngüsel çözümlenen hata kontrol kodları oluşturmaya yarayan bir tekniktir. EXIT grafikleri, hata kontrol kodlarından olan LDPC ve turbo kodlar için kullanılır. EXIT grafikleri, Brink tarafından önerilmiştir [3],[4]. EXIT grafikleri çözümleyici içerisindeki elemanların tepkilerini içerir. Bu tepkiler çoğunlukla dışsal bilgiye bağlı olan tepkilerdir. Eğer çözümleyici içerisinde mesaj değişimi yapan iki adet eleman varsa, çözümleyicinin davranışı iki boyutlu bir grafik üzerinde gösterilebilir. Çözümleyicideki bir elemanın tepkisi, yatay eksen o elemanın girişlerini ve dikey eksen çıkışlarını gösterecek şekilde çizilirse, çözümleyicideki diğer elemanın tepkisi, dikey eksen girişleri ve yatay eksen çıkışları gösterecek şekilde çizilir. İki eleman arasındaki mesaj değişimleri, çizilen bu iki eğri arasında ilerlemeyle bulunur. Başarılı bir çözümleme için bu iki eğri arasında bir alan olmalıdır ve bu iki eğri birbirini kesmemelidir. Bu alanda meydana gelen mesaj değişimleri ile dışsal bilgi sıfırdan bire doğru ilerler. Çözümleyici üzerinde eniyileme çalışmaları yapılırken iyi bir çözümleme yapılmak isteniyorsa, iki elemanın tepkileri için çizilen eğrilerin birbirlerine oldukça yakın olması istenir.

EXIT grafikleri ilk olarak turbo kodların eniyilemesi için önerildiğinden, EXIT grafikleri ile ilgili temel teorik bilgi, turbo kodlar için EXIT grafikleri oluşturmayı anlatan bilgilerle verilecektir.

5.1 TURBO KODLAR İÇİN EXIT GRAFİKLERİ

EXIT grafikleri alıcıda döngüsel işlemler sırasında bir çözücüden diğerine gidip gelen yumuşak bilgileri analiz etmeye olanak tanır. Bu süreçte, çözücüler arasındaki bilgi değişimi grafiklerle gösterilir ve bu grafikler, çözücülerin girişlerindeki önsel bilgi ile bu çözücüler tarafından üretilen dışsal bilginin, ortak bilgi şeklinde transfer edildiği düşünülerek çizdirilmiş grafiklerdir.

EXIT grafikleri, döngüsel çözümleme analizleri için kullanılan yoğunluk evrimi tekniğinin gelişmesi ile meydana çıkmıştır. Her iki analiz çeşidi de döngüsel çözümleme için uygundur fakat EXIT grafiği metodu daha az karmaşık hesaplamalar içerdiği için

kullanımı daha kolay ve anlaşılırdır.

EXIT grafikleri metodunda, hem önsel bilgi hem de dışsal bilgi bu değerler ile mesaj bitleri arasındaki ortak bilgi ile ölçülür. Bu büyüklükler aşağıdaki denklemlerle birbirine bağıntılıdır. Brink'in gösterimi takip edilirse [4];

$$L_e(b_i) = L(b_i|Y) - L(b_i) - L_{cy_i} = E_i = D_i - A_i - Y_i \tag{5.1}$$

olur.

Bu denklemde A, önsel bilgiyi; E, dışsal bilgiyi ve Y, kanal bilgisini temsil etmektedir. Bütün bu değerler LLR olarak tanımlanır. $L_e(b_i)$, b_i bitine ait dışsal bilginin LLR değeridir. L_{cy_i} değeri alıcıya giren kanal bilgisini i numaralı bit için göstermektedir.

5.1.1 EXIT Grafiklerine Giriş

Turbo kodların BER-SNR grafikleri temel olarak üç ana bölgeye ayrılır [19]:

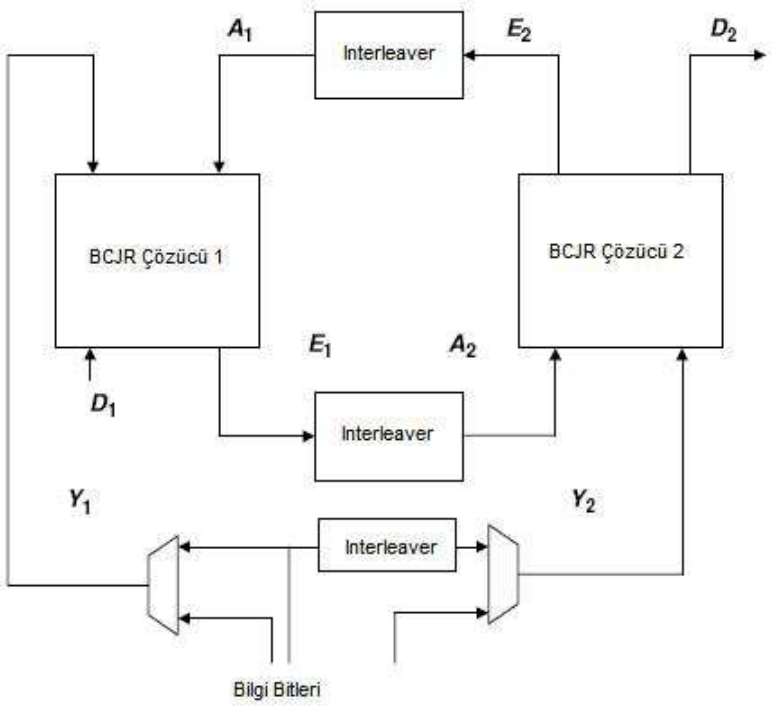
- Birinci bölge SNR değerlerinin düşük olduğu bölgedir. Bu bölgede döngüsel çözümleme, kodlanmamış iletime göre daha kötü sonuçlar vermektedir.
- İkinci bölge SNR değerlerinin orta değerlerinin olduğu bölgedir. Bu bölgede döngüsel çözümleme oldukça etkindir. Bu bölgeye şelale bölgesi adı verilir ve bu bölgede performans doğrusal olmasa da, döngü sayısındaki artışla artar.
- Üçüncü bölge SNR'ın yüksek olduğu bölgedir. Bu bölgede çözümleme birkaç döngüden sonra yakınsar fakat performans, SNR'ı arttırdıkça daha yavaş artar.

EXIT grafikleri özellikle şelale bölgesinin analizi için iyi bir araçtır ve aynı zamanda diğer bölgelerdeki kod davranışlarını da gösterir. Grafikler, önsel bilgi ile mesaj bitleri bilgisi arasındaki ortak bilgi ile dışsal bilgi ile mesaj bitleri bilgisi arasındaki ortak bilgiler ile oluşturulur.

Daha önce de belirtildiği üzere, döngüsel çözümlemenin performansı döngü sayısını artırmakla artar. Ancak bu artış doğrusal değildir. Döngü sayısını arttırmak da performansı pratik anlamda bir limite yaklaştırır. Bu anlamda etkin döngü sayıları EXIT grafikleri ile belirlenebilir.

5.1.2 EXIT Grafikleri Oluşturma

Brink [4] nolu referansta EXIT grafiklerini, Şekil 5.1'de görülen model üzerinde önermiştir. Bu modeller şekilde görülen iki adet BCJR çözücünün hem seri hem de paralel bağlanması için tanımlanmıştır. EXIT grafikleri [4] nolu referansta ilk olarak turbo kodlar için önerilmiş ve bu kodlara uygun çözme algoritması olan BCJR çözücüler kullanılmıştır. Brink'in önerdiği bu EXIT grafik metodunu anlamak için, bu tezde kullanılmamasına rağmen turbo kodlar için kullanılan bu çözücüler için EXIT grafiğinin nasıl çizdirildiğini kavramak gerekmektedir. Şekil 5.1, [4] nolu referanstan alınmıştır ve bu şekle ait herhangi bir simülasyon çalışması bu tez kapsamında yapılmamıştır. Bu şekil EXIT grafiği metoduna ait teorik bilgilerin kavranması için kullanılmıştır.



Şekil 5.1: BCJR çözücülerinin operasyonları

Şunu not etmek gerekir ki, Brink EXIT grafiklerini önerirken turbo kodlar için Şekil 5.1'de görüleceği üzere karıştırıcılar kullanmıştır. Karıştırıcılar, bir değişkene ait dağılımın istatistiksel özelliklerini değiştirmemekte ve turbo kodlarda, gürültünün ortalamasını bütün mesajlar üzerine dağıttığından iyi yönde etkileri olmaktadır. Bölüm 4'te anlatılan haberleşme modeli oluşturulurken sistemde karıştırıcı kullanılmamış ve

bu durumun [8] nolu referans ile uyumlu olduğu görülmüştür. Ancak EXIT grafiklerine teorik giriş yapacağımız bu bölümlerde, hem turbo kodları hem de sadece LDPC koduna ait çözücülerin EXIT grafikleri çizdirilirken kullanılan model şekillerinde karıştırıcılar yer alacaktır.

Şekil 5.1'de yer alan ilk çözücü önsel bilgiyi, kanal bilgisini ve birinci kodlayıcı tarafından üretilen eşitlik kontrol bitlerini kullanır. Bu bilgiler dışsal bilgiyi oluşturmak için kullanılır. Bu dışsal bilgi vektörü

$$E_1 = D_1 - A_1 - Y_1 \tag{5.2}$$

olarak tanımlanır. Bu vektörün elemanları

$$E_{i1} = L_{e1}(b_i) = L_1(b_i|Y_1) - L_1(b_i) - L_{cy_{i1}} = D_{i1} - A_{i1} - Y_{i1} \tag{5.3}$$

olarak tanımlanır. Bu eşitlikteki gösterimin Eşitlik 5.1'den farkı 1 numaralı çözücüye ait bilgileri sembolize etmesidir.

İkinci çözücü, kanal bilgisi Y_2 ve ilk çözücüden gelen önsel A_2 bilgilerini kullanarak, dışsal bilgi E_2'yi üretir.

$$E_2 = D_2 - A_2 - Y_2 \tag{5.4}$$

olarak tanımlanır. Bu vektörün elemanları

$$E_{i2} = L_{e2}(b_i) = L_1(b_i/Y_2) - L_2(b_i) - L_{cy_{i2}} = D_{i2} - A_{i2} - Y_{i2} \tag{5.5}$$

olarak tanımlanır ve bu dışsal bilgi, bir sonraki döngüde ilk çözücü için önsel bilgi (A_1) olarak kullanılacaktır.

AWGN kanal için giriş ve çıkış değişkenleri rastgele değişkenler olarak tanımlanır ve

$$Y = X + N \tag{5.6}$$

ile bağıntılıdır. Burada X, x mesaj bitlerini gösteren bir rastgele değişkendir. Y gönderilen bilginin bulunması sonucunda elde edilen rastgele değişkendir ve N gürültü rastgele değişkenini gösterir. Bu kanal Gaussian yoğunluk fonksiyonu ile tanımlanır ve bu tanım LLR'da yerine koyulursa:

$$L(y|x) = 2\frac{E_b}{\sigma^2}y = \frac{2}{\sigma^2}y \tag{5.7}$$

elde edilir. Burada ortalama bit enerjisi $E_b = 1$'dir. Bu alınan sinyali şu şekilde tanımlamaya sebep olur:

$$\begin{aligned} Y &= \frac{2}{\sigma^2} y = \frac{2}{\sigma^2}(x+n) \\ Y &= \frac{2}{\sigma^2} y = \mu_Y x + n_Y \end{aligned} \tag{5.8}$$

Burada

$$\mu_Y = 2/\sigma^2 \tag{5.9}$$

ve n sıfır ortalamalı ve varyansı

$$\sigma_Y^2 = 4/\sigma^2 \tag{5.10}$$

olan bir rastgele değişkendir. Bu değişkenin ortalaması ve varyansı şu formülle birbirine bağıntılıdır [4]:

$$\mu_Y = \sigma_Y^2/2 \tag{5.11}$$

Bu ilişki EXIT grafiklerinin oluşturulmasında faydalı olacaktır. Turbo kodlarda her iki çözücü de aynı çözme algoritmasını, aynı kodu ve dolayısıyla aynı kafes diyagramını kullanmaktadır. Buna rağmen seri kodlardan birinci ve ikinci kodlar birbirinden farklılık gösterebilir. Her iki çözücünün aynı olduğu durumlarda, örnek olarak sadece birinci çözücü için EXIT grafik analizleri uygulanabilir.

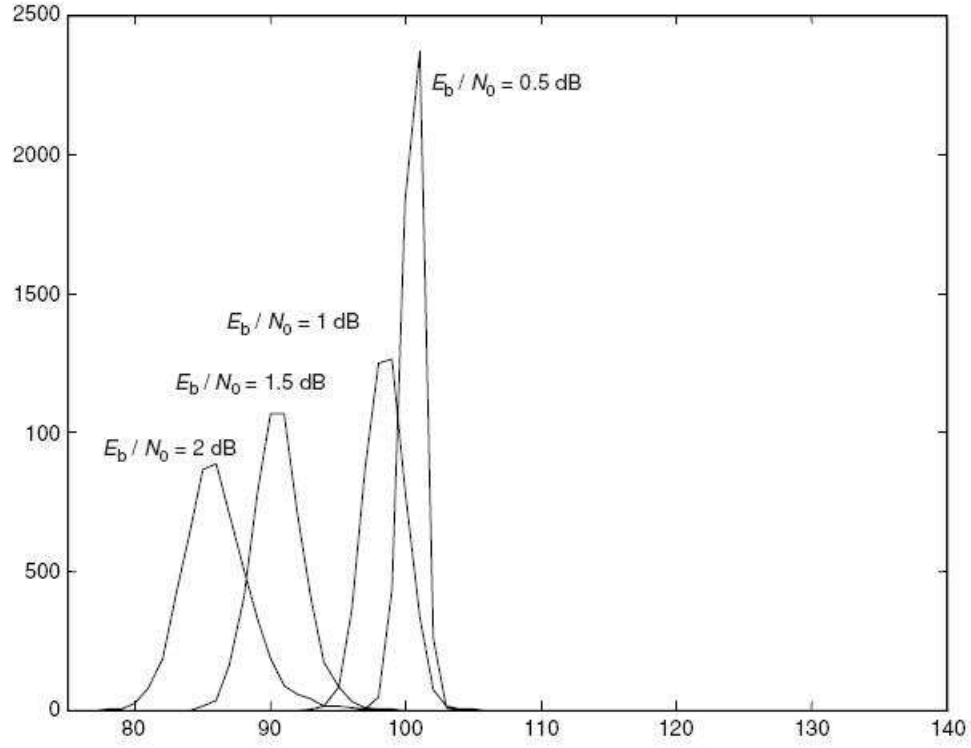
5.1.3 Seri Çözücüler için EXIT Grafikleri Analizi

EXIT grafikleri seri çözücülerden bir tanesinin çözme operasyonu için oluşturulabilir. Aşağıda yapılan analizler sadece bir çözücü için yapılmıştır. Analizlere geçmeden önce şunu not etmek gerekir ki, BCJR çözme algoritması karmaşık bir yapıya sahip olduğu için her analiz edilmek istenen parametre teorik yollarla hesaplanamayabilir. Bu parametreler üzerinde Monte Carlo simülasyonları yapılarak istenilen değerler kestirilir ve bu değerler EXIT grafik metodunda kullanılır [4].

BCJR çözücüde önsel (A) bilginin değerleri kanal gözlemlerinden (Y) bağımsızdır. BCJR çözücünün ürettiği dışsal bilginin (E) dağılımının Gaussian olduğu

görülmüştür. Dolayısı ile bir sonraki döngü için önsel değerler Gaussian dağılımlı olacaktır. Dışsal bilginin Gaussian dağılımlı olduğu, bu bilginin histogramında nümerik olarak gözlenmiştir [19].

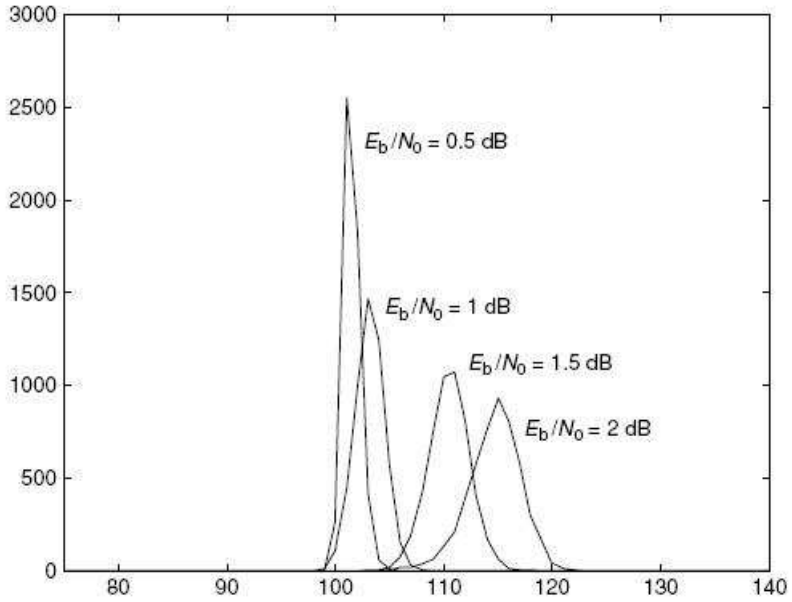
Şekil 5.2 ve 5.3'te BCJR çözücünün operasyonu sonucunda üretilen dışsal bilginin normalize edilmemiş histogramları verilmiştir. Bu şekiller, Monte Carlo simülasyonları ile elde edilen LLR değerlerinin histogramını vermektedir. Şekil 5.2'de, x mesaj bitlerinin 0 olduğu durum için dışsal bilginin normalize edilmemiş histogramı ve şekil 5.3'de, x mesaj bitlerinin 1 olduğu durum dışsal bilginin normalize edilmemiş histogramı değişik SNR değerleri için verilmiştir.



Şekil 5.2: $x = 0$ için dışsal bilginin normalize edilmemiş histogramı (Bu şekil [19] nolu referanstan alınmıştır.)

Şekil 5.2 ve 5.3, BCJR çözücü tarafından üretilen dışsal bilginin Gaussian dağılım ile karakterize edilebilir olduğunu göstermiştir. SNR seviyelerini arttırmanın, $x = 0$ ve $x = 1$ için çizdirilen histogramları birbirinden uzaklaştırdığı sonucu bu grafiklerden çıkarılmıştır. Histogram ile pdf kestirimi metodu ile ilgili detaylı bilgi EK 1'de verilmiştir.

Monte Carlo simülasyonları ile BCJR çözücü tarafından üretilen dışsal bil-



Şekil 5.3: $x = 1$ için dışsal bilginin normalize edilmemiş histogramı (Bu şekil [19] nolu referanstan alınmıştır.)

ginin ortalamasının sıfır ve varyansının σ_E^2 olduğu görülmüştür. Bu dışsal bilgi, gönderilen x bitinin μ_E ile çarpımına, bir Gaussian rastgele değişken eklenmesi ile modellenir. Bu dışsal bilgi, $x = 0$ $(E^{(0)})$ ve $x = 1$ $(E^{(1)})$ için şu şekilde ifade edilir:

$$\begin{aligned} E^{(0)} &= \mu_{E0}x + n_{E0} \\ E^{(1)} &= \mu_{E1}x + n_{E1} \end{aligned} \tag{5.12}$$

Burada

$$\begin{aligned} \mu_{E0} &= \sigma_{E0}^2/2 \\ \mu_{E1} &= \sigma_{E1}^2/2 \end{aligned} \tag{5.13}$$

olur.

$x = 0$ için dışsal bilgiye ait olasılık yoğunluk fonksiyonu $p_E(\xi|X = -1)$ şu şekilde yazılır:

$$p_E(\xi|X = -1) = \frac{1}{\sqrt{2\pi}\sigma_{E0}} e^{-(\xi+\mu_{E0})^2/2\sigma_{E0}^2} \tag{5.14}$$

$x = 1$ için dışsal bilgiye ait olasılık yoğunluk fonksiyonu $p_E(\xi|X = +1)$ şu şekilde yazılır:

$$p_E(\xi|X = +1) = \frac{1}{\sqrt{2\pi}\sigma_{E1}} e^{-(\xi-\mu_{E1})^2/2\sigma_{E1}^2} \tag{5.15}$$

Simülasyonlardan, $x = 0$ ve $x = 1$ için çizdirilen dağılımlara ait ortalamaların (yani u_{E0} ve u_{E1}'in) birbirine yakın olması durumunda kodun performansının, kodlanmamış olan alıcının performansından daha kötü olduğu sonucu çıkarılmıştır. Bu değerler birbirinden ne kadar farklı olursa turbo kodu şelale bölgesine o kadar yakınlaşır.

Şunu da not etmek gerekir ki, Şekil 5.1'de karıştırıcı kullanmak bir sonraki döngüde önsel bilgi olacak olan dışsal bilginin yerlerinin farklı olmasını sağlar ancak istatistiksel özelliklerini değiştirmez. Böylelikle önsel bilgi şu şekilde modellenir,

$$A = \mu_A x + n_A \tag{5.16}$$

ve

$$\mu_A = \sigma_A^2/2 \tag{5.17}$$

olur. Bu parametrelerle önsel bilginin olasılık yoğunluk fonksiyonu

$$p_A(\xi|X = x) = \frac{1}{\sqrt{2\pi}\sigma_A} e^{-(\xi-(\sigma_A^2/2)x)^2/2\sigma_A^2} \tag{5.18}$$

şeklinde yazılır.

Önsel bilgi A rastgele değişkeni ile X mesaj bitlerinin rastgele değişkeni arasındaki ortak bilgi, önsel ortak bilgiyi ölçmek için kullanılır. Bu ortak bilgi şu şekilde hesaplanır:

$$I_A = I(X;A) = \frac{1}{2} \sum_{x=-1,+1} \int p_A(\xi|X = x) log_2 \frac{2p_A(\xi|X = x)}{p_A(\xi|X = -1) + p_A(\xi|X = +1)} d\xi \tag{5.19}$$

Denklem 5.19, denklem 5.18 ile birleştirilirse [4]:

$$I_A = 1 - \int \frac{1}{\sqrt{2\pi}\sigma_A} e^{-(\xi-\sigma_A^2/2)^2/2\sigma_A^2} log_2(1 + e^{-\xi}) d\xi \tag{5.20}$$

Denklem 5.20, kısa şekilde J fonksiyonu ile aşağıdaki şekilde ifade edilir. J fonksiyonu ortak bilginin pratik bir şekilde hesaplanması sırasında faydalı olacaktır [5]. J fonksiyonu ile detaylı bilgi EK 2'de verilmiştir.

$$J(\sigma) = I_A(\sigma_A = \sigma) \tag{5.21}$$

J fonksiyonu kapalı formda ifade edilemez. Bu fonksiyon monotonik olarak artan olduğu için tersi alınabilir:

$$\sigma_A = J^{-1}(I_A) \tag{5.22}$$

Ortak önsel bilginin hesaplanmasında kullanılan mantıkla, dışsal bilgi E rastgele değişkeni ile X mesaj bitlerinin rastgele değişkeni arasındaki ortak bilgi, dışsal ortak bilgiyi ölçmek için kullanılır. Bu ortak bilgi şu şekilde hesaplanır:

$$I_E = I(X;E) = \frac{1}{2}\sum_{x=-1,+1}\int p_E(\xi|X=x) log_2 \frac{2p_E(\xi|X=x)}{p_E(\xi|X=-1)+p_E(\xi|X=+1)} d\xi \tag{5.23}$$

EXIT grafiği her bir SNR değeri için önsel bilgi ile mesaj bitleri arasındaki ortak bilgiyi (I_A) ve dışsal bilgi ile mesaj bitleri arasındaki ortak bilgiyi (I_E) tanımlar. Bu dışsal bilgiye ait transfer fonksiyonu şu şekilde ifade edilir:

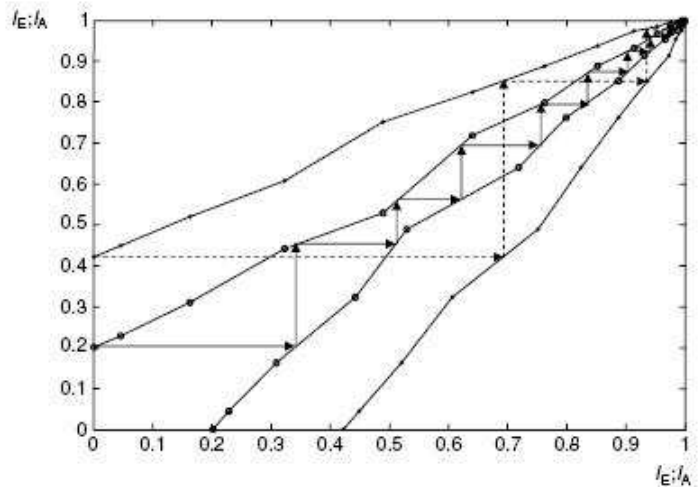
$$I_E = T(I_A, E_b/N_0) \tag{5.24}$$

Bu eğri (T fonksiyonu), I_A'in belli bir değeri için ve belli bir SNR değeri için hesaplanan I_E değerleri ile çizdirilir. Bu önsel bilgi, BCJR çözücüye E_b/N_0 değeri ile belirlenen Gaussian gürültüden geçen kod kelimeleri ile birlikte uygulanır. BCJR çözücüsü I_E değeri ile karakterize edilen E değerleri üretir. Monte Carlo simülasyonları ile E değerinin olasılık yoğunluk fonksiyonu, $p_E(\xi|X=x)$, elde edilir. $x=0$ mesaj bitine ait tahminler ile $hist_E(\xi|X=-1)$ ve $x=1$ mesaj bitine ait tahminler ile $hist_E(\xi|X=+1)$ histogramları elde edilir. $I_E = I(X;E)$ ortak bilgisi bu histogramlar üzerinden hesaplanır. Bu hesaplamaya ait formülasyon aşağıdaki şekildedir:

$$\begin{aligned} I_E &= I(X;E) \\ &= \frac{1}{2}\times\int hist_E(\xi|X=+1) log_2 \frac{2hist_E(\xi|X=+1)}{hist_E(\xi|X=-1)+hist_E(\xi|X=+1)} d\xi \\ &+ \frac{1}{2}\times\int hist_E(\xi|X=-1) log_2 \frac{2hist_E(\xi|X=-1)}{hist_E(\xi|X=-1)+hist_E(\xi|X=+1)} d\xi \end{aligned} \tag{5.25}$$

I_E ve I_A değerleri SNR'la dolayısı ile σ_A ile değişmektedir. Böylelikle EXIT grafiklerinde kullanılan parametreler SNR bağımlıdır. EXIT grafikleri I_A'i I_E'nin fonksiyonu olarak tanımlar. Döngüsel bir alıcı kullanıldığı için bir sonraki döngüde önsel bilgi dışsal bilgi olur ve sonraki döngüde bu dışsal bilgi tekrar önsel bilgi olur. EXIT grafikleri, önsel bilgi ile dışsal bilginin döngüsel bağımlılığını da göstermektedir.

Turbo kodlara ait örnek EXIT grafiği, SNR'ın 1 dB ve 2 dB değerleri için Şekil 5.4'te verilmiştir. Bu EXIT grafiği üzerinde döngüler açık bir şekilde görülmektedir. Çözme operasyonu ilk çözücü ile başlar ve önsel bilgi burada sıfırdır. Önsel bilgi sıfırken elde edilen dışsal bilgi bir sonraki döngüde ikinci çözücü için önsel bilgi olacaktır. EXIT grafiklerinin simetrik olmasının sebebi iki çözücünün birbirinin aynısı olmasıdır. Bu süreç her bir güncellenmiş I_A değeri için I_E değeri bulunmasıyla (eğriler arasındaki dikey geçişler) ve bu I_E değerinin diğer çözücü için I_A değerine döndürülmesiyle (eğriler arasındaki yatay geçişler) devam eder.



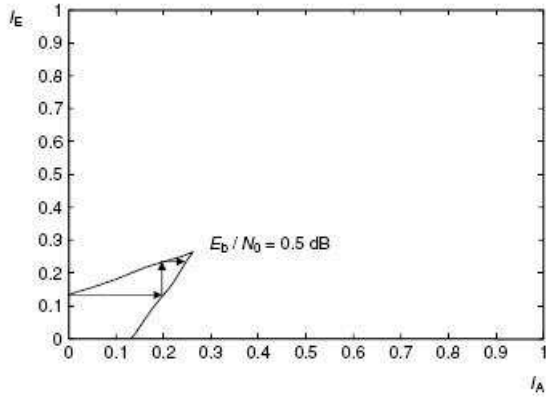
Şekil 5.4: Turbo kodlar için EXIT grafikleri (Bu şekil [19] nolu referanstan alınmıştır.)

Şekil 5.4'te, SNR'ın 1 dB değeri için çizilen EXIT grafiği ile SNR'ın 2 dB değeri için çizilen EXIT grafiği kıyaslandığında, SNR'ın 1 dB değeri için daha fazla döngü gerektiği görülür. Burada amaç en az sayıda döngüyle olabilecek en düşük SNR eşik değerini belirlemektir.

Döngüsel çözümleme prosedürü $I_E = T(I_A, E_b/N_0)$ eğrisi ile $I_A = T(I_E, E_b/N_0)$ eğrisi kesiştiğinde etkin değildir. Bu durum Şekil 5.5'te gösterilmiştir. Bu durumda döngüsel çözümleme kodlanmamış iletime göre daha kötü sonuçlar verebilir. Bu sebepten EXIT grafiklerinde bu eğrileri kesiştirmeden döngüsel çözümleme yapılabilecek en düşük SNR eşik değeri bulunur.

5.2 LDPC KODLARI İÇİN EXIT GRAFİKLERİ

Bölüm 4'te tasarımı yapılan haberleşme sistemine eniyileme yapabilmek için, turbo kodlar için çizdirilen EXIT grafiklerinde öğrenilen bilgilerin yanı sıra



Şekil 5.5: Döngüsel çözümleme için EXIT grafiği (Bu şekil [19] nolu referanstan alınmıştır.)

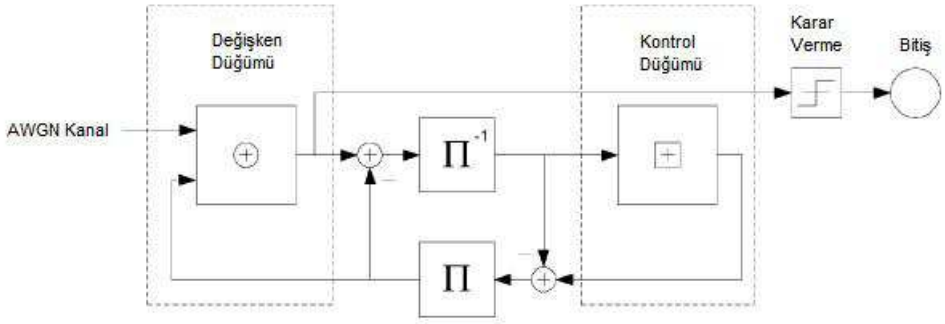
çözümleme blokları sadece LDPC bloklarından (değişken ve kontrol düğümleri) oluşan bir çözümleyici için çizdirilen EXIT grafiği metodunun da öğrenilmesi gereklidir. Bu bölümde anlatılacak olan bilgilere ait herhangi bir simülasyon çalışması yapılmamıştır. Burada anlatılan bilgiler, Bölüm 4.2'de tasarlanan sistem için çizdirilecek EXIT grafiklerinin temellerini oluşturmaktadır.

Brink, EXIT grafiği metodunu turbo kodlardan sonra LDPC kodlar için de [5] nolu referansta önermiştir. Ancak bu önermede turbo kodlar için yaptığı çıkarımları ve gösterimleri LDPC kodlar için uyumlandırmıştır. Bu sebepten bu tez çalışmasında LDPC kodlar için çizdirilecek EXIT grafikleri ile ilgilenmemize rağmen, Brink LDPC kodlar için EXIT grafiklerinin çizimini, turbo kodlar için yaptığı çıkarımlardan önerdiğinden bir önceki bölümde turbo kodlar için EXIT grafikleri oluşturma yöntemi anlatılmıştır.

LDPC kodları için EXIT grafiği oluşturma mantığı, turbo kodlarda kullanılan mantıkla aynıdır [5]. Turbo kodlarda gelen ve giden ortak bilgiler birbirinin aynı iki BCJR çözücüsü arasında olurken, LDPC kodlarda I_A ve I_E ortak bilgileri LDPC çözücüsünün değişken ve kontrol düğümleri arasında gidip gelmektedir. LDPC için EXIT grafiği analizlerinin yapılacağı döngüsel çözücü modeli Şekil 5.6'da verilmiştir.

Derecesi d_v olan bir değişken düğümü $d_v + 1$ adet gelen mesaja sahiptir. Bu mesajlardan d_v adeti karıştırıcıdan bir tanesi de kanaldan gelmektedir. Değişken düğümü aşağıdaki formülasyona uygun hesaplamalar yapar:

$$L_{i,out} = L_{ch} + \sum_{j \neq i} L_{j,in} \tag{5.26}$$



Şekil 5.6: LDPC kodlar için döngüsel çözümleyici [19]

Burada $L_{j,in}$ değişken düğümüne giren j numaralı önsel bilgiyi, $L_{i,out}$ ise değişken düğümünden çıkan i numaralı dışsal bilgiyi tanımlar. L_{ch} ise kanal bilgisidir.

Buradaki kiplenimde AWGN kanalı kullandığımızı ve σ_n^2 gürültü varyansı ile BPSK kullandığımızı varsayarsak kanal bilgisi,

$$L_{ch} = log\frac{p(y|x=+1)}{p(y|x=-1)} = \frac{2}{\sigma_n^2}y \tag{5.27}$$

şeklinde ifade edilir. X ve Y rastgele değişkenler olduğu için, X üzerinde koşullandırılmış L_{ch} değeri

$$\sigma_{ch}^2 = \frac{4}{\sigma_n^2} = 8R\frac{E_b}{N_0} \tag{5.28}$$

şeklinde karakterize edilir. Burada R kod oranını göstermektedir.

EXIT fonksiyonunu hesaplamak için $L_{j,in}$ AWGN kanalın çıktısı olan ve j numaralı karıştırıcıdan BPSK ile gönderilen LLR değerleridir. d_v dereceli değişken düğümüne ait EXIT fonksiyonu şu şekilde ifade edilir [5]:

$$I_{E,VND}\left(I_A, d_v, E_b/N_0, R\right) = J\left(\sqrt{(d_v-1)[J^{-1}(I_A)]^2 + \sigma_{ch}^2}\right) \tag{5.29}$$

J ve J^{-1} fonksiyonları ile ilgili bilgi EK 2'de verilmiştir.

d_c dereceli bir kontrol düğümünün çözümlenmesi, uzunluğu d_c olan tekli eşitlik kontrol kodunun çözümüne eştir. Çıkış LLR değerleri:

$$L_{i,out} = ln\frac{1-\prod_{j\neq i}\frac{1-e^{L_{j,in}}}{1+e^{L_{j,in}}}}{1+\prod_{j\neq i}\frac{1-e^{L_{j,in}}}{1+e^{L_{j,in}}}} \tag{5.30}$$

şeklinde ifade edilir. $L_{j,in}$ değerleri, AWGN kanalın çıkışı ve BPSK ile gönderilen j numaralı karıştırıcıdan gelen LLR değerleri ile modellenir. Kontrol düğümü EXIT eğrileri,

kapalı formda veya Monte Carlo simülasyonları ile hesaplanabilir. Bu yöntemlere alternatif olarak, ikili hata kanalı (BEC) için ikilem teoremi (duality property [5]) kullanılarak, uzunluğu d_c olan tek eşitlik kontrolü (SPC) için ortak bilgi olan $I_{E,SPC}$'ye ait EXIT eğrisi, uzunluğu d_c olan tekrarlamalı (REP) kodun EXIT eğrisi $I_{E,REP}$ ile ifade edilebilir [5]:

$$I_{E,SPC}(I_A, d_c) = 1 - I_{E,REP}(1 - I_A, d_c) \tag{5.31}$$

Bu özellik AWGN kanal ve BPSK kiplenimi için tam olarak doğru sonuç vermese de güvenilir sonuçlar elde edilir [5]:

$$\begin{aligned} I_{E,CND}(I_A, d_c) &= 1 - I_{E,REP}(1 - I_A, d_c) \\ &= 1 - J(\sqrt{d_c - 1}.J^{-1}(1 - I_A)) \end{aligned} \tag{5.32}$$

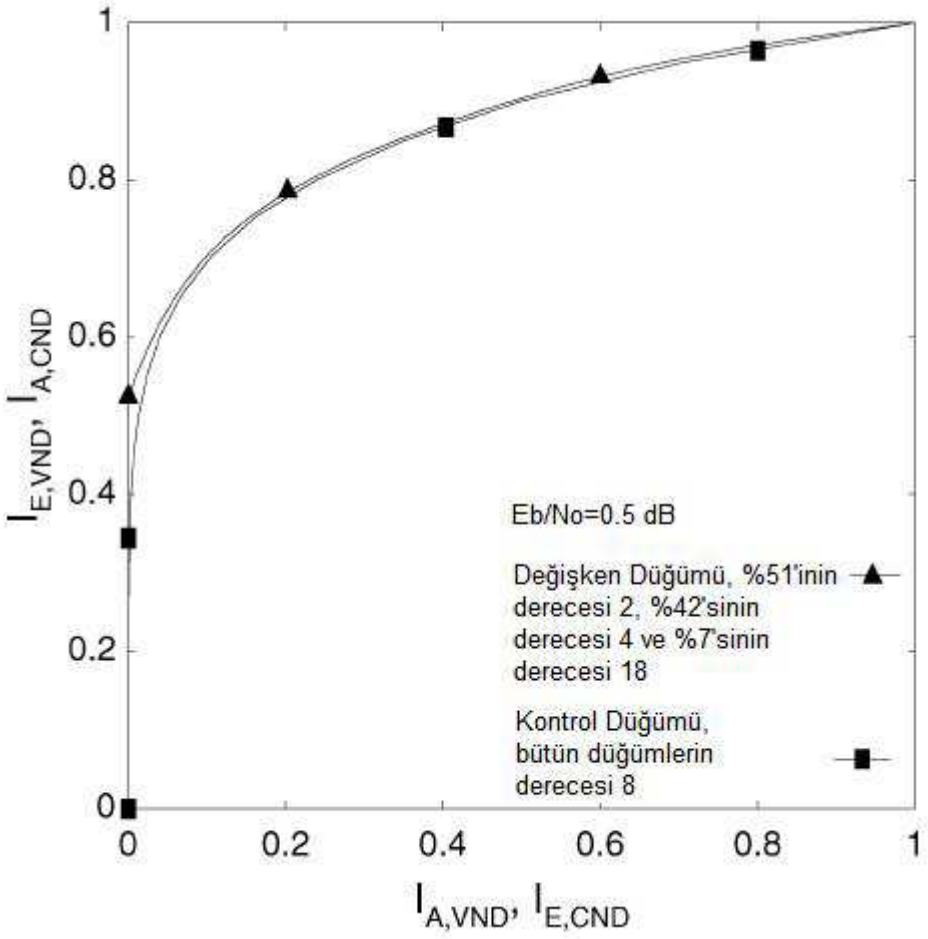
Bu ifadenin tersi olan ifadeyi kullanmak daha kullanışlı olacaktır [5]:

$$I_{A,CND}(I_E, d_c) = 1 - J\left(\frac{J^{-1}(1 - I_E)}{\sqrt{d_c - 1}}\right) \tag{5.33}$$

Yukarıda verilen denklemlere uygun olarak çizdirilen LDPC kodlar için örnek EXIT grafiği Şekil 5.7'de verilmiştir. LDPC EXIT grafiklerinde, değişken düğümüne ait eğrinin kontrol düğümüne ait olan eğrinin her zaman üstünde yer alması istenir. SNR değerleri değişken düğümlere ait EXIT grafiklerini etkilediği için, SNR azaldıkça değişken düğümlerine ait eğri, x eksenine doğru yaklaşır. Eğer değişken düğümüne ait eğri kontrol düğümüne ait olan eğriyi keserse, LDPC'ye ait döngüsel çözümleme kodlanmamış iletime kıyasla daha az etkin olacaktır. LDPC kodlarına ait EXIT grafiklerinde amaç, bu iki eğriyi birbirini kesmeyecek şekilde ancak en yakın noktada tutmaktır. Böylelikle SNR'a karşılık BER grafiklerinin, en az hangi SNR değerinde şelale bölgesine gireceği belirlenmiş olur.

5.3 DİZAYN EDİLEN ALICIYA AİT EXIT GRAFİĞİ ANALİZLERİ

Bu bölümde Bölüm 4.2'de ele alınan haberleşme sistemine ait EXIT grafiği analizleri yapılacaktır. Yukarıdaki bölümlerde, EXIT grafiklerinin temel çıkış noktası olan turbo kodlar için EXIT grafikleri ile LDPC kodlar için EXIT grafik analizlerinin nasıl yapılacağı gösterildi. Bu bölümde ise, EXIT grafiği metodu ile dizaynını

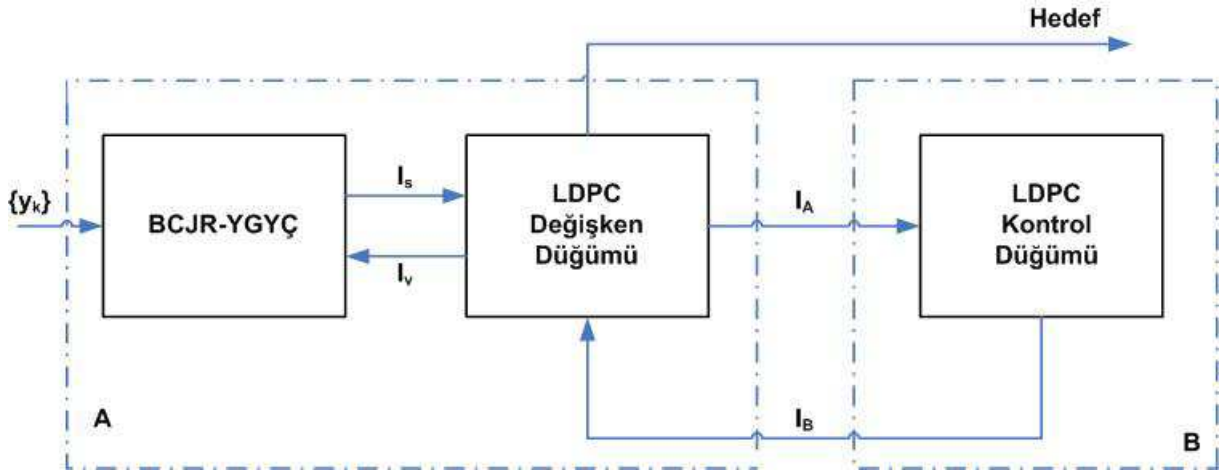


Şekil 5.7: LDPC kodlar için EXIT grafiği örneği (Şekil [5] nolu referanstan alınmıştır.)

yaptığımız alıcının nasıl analiz edileceğinin, [8] nolu referansta verilen düzensiz LDPC kodu dağılımları için bu EXIT grafiğine ait simülasyon çalışmalarının ve bu dağılımlara ait SNR'a karşılık BER performanslarının gösterimi yapılacaktır.

Şekil 5.8'de gösterilen her bir blok için EXIT eğrileri çizdirilebilir. Şekil 5.8'de A ve B bloklarının çıkışlarındaki ortak bilgiler sırasıyla I_A ve I_B olarak tanımlanır ve A bloğunun içerisindeki BCJR-YGYÇ bloğunun giriş ve çıkışındaki ortak bilgi sırasıyla I_v ve I_s olarak tanımlanır. Çözümleme süreci, EXIT grafiklerindeki bu ortak bilginin güncellenmesi ile gösterilir. Eğer ortak bilgi 1'e yakınsarsa, bit hata olasılığı 0'a yakınsayacaktır.

Bu noktada, A ve B blokları için EXIT grafikleri ile ilgilenilmesi gereklidir. B bloğu sadece LDPC kontrol düğümlerinin EXIT eğrisi ile karakterize edilebilir ancak blok A'nın EXIT eğrisi LDPC değişken düğümünün EXIT eğrisinin BCJR-YGYÇ ile



Şekil 5.8: Bölüm 4.2'deki alıcıya ait EXIT modeli

birleşmesinden elde edilir. Bölüm 5.2'de LDPC'nin değişken ve kontrol düğümleri için EXIT fonksiyonlarına ait eşitlikler verilmişti. Bu eşitlikler bloklar arasında değiştirilen bilgilerin Gaussian olduğu kabul edilerek çıkarılmıştı ve bu kabullenme basitlik ve iyi bir güvenilirlik vermişti. BCJR-YGYÇ bloğunun EXIT eğrisine ait nümerik hesaplamalar yapılması zor bir iş olduğundan, tahmini hesaplamalar Monte Carlo simülasyonları ile yapılacaktır [8].

Bunu takiben, A ve B bloklarının çıkışlarındaki yaklaşık ortak bilgiler aşağıdaki eşitliklerle hesaplanacaktır [8]:

$$I_B = 1 - \sum_j \rho_j J\left(\sqrt{j-1}J^{-1}(1-I_A)\right) \tag{5.34}$$

$$I_A = \sum_i J\left(\sqrt{(i-1)(J^{-1}(I_B))^2 + (J^{-1}(I_S))^2}\right) \tag{5.35}$$

Yukarıdaki eşitliklerde kullanılan J fonksiyonuna ait detaylı bilgi EK 2'de verilmiştir.

BCJR-YGYÇ bloğunun çıkışındaki I_S ortak bilgisi, değişken düğümlerinden BCJR-YGYÇ bloğuna geçen I_V ortak bilgilerinin bir fonksiyonudur. LDPC değişken düğümlerinden BCJR-YGYÇ bloğuna geçen I_V ortak bilgileri aşağıdaki eşitlikle hesaplanır [8]:

$$I_V = \sum_i \lambda_i J\left(\sqrt{i}.J^{-1}(I_B)\right) \tag{5.36}$$

I_A ile I_B, Eşitlik 5.34 ve 5.35'e uygun olarak kapalı formda tanımlanabilir. B bloğu sadece LDPC'nin kontrol düğümlerinden oluştuğu için bu bloğa ait EXIT

eğrisinin çizdirilmesi için uygun bir derece dağılımının olması yeterlidir. Ancak Eşitlik 5.35'teki I_S değerlerini hesaplamak gereklidir. Burada BCJR-YGYÇ bloğunun girişine uygulanan I_V ortak bilgileri kullanılarak, I_S değerleri Monte Carlo simülasyonu yardımıyla bulunur.

Şekil 5.9: I_S değerinin hesaplanması

I_S değerinin hesaplanmasına ait blok diyagram Şekil 5.9'da verilmiştir. I_S değerlerinin hesaplanması için ilk olarak [0-1] aralığında I_B değerlerine karşılık gelen I_V değerleri Eşitlik 5.36 yardımıyla hesaplanır. Hesaplanan bu I_V değerlerinin daha sonra J^{-1} fonksiyonuna karşılık gelen değeri bulunur. Bu fonksiyon yardımıyla σ_A değeri bulunmuş olur. Bu σ_A değeri kullanılarak önsel bilgiler (A) Eşitlik 5.16 ve 5.17 kullanılarak oluşturulur ve bu oluşturulan değerler BCJR-YGYÇ bloğunun girişi olarak kullanılır. BCJR-YGYÇ bloğunun ikinci girişi ise kanal gözlemleri olan Y rastgele değişkenleridir. Bu değişkenler sayesinde SNR değerleri EXIT grafiklerine dahil olmuş olur. SNR değerinden hesaplanan σ_n değeri AWGN kanalına ait gürültü üretmek için kullanılır. Bu gürültü üretilen sembollerin üzerine eklenerek Y değerleri elde edilir ve bu değerler BCJR-YGYÇ bloğuna uygulanır. BCJR-YGYÇ bloğu çözümleme sürecini gerçekleştirdikten sonra LLR sonuçlarını üretir ve bu LLR sonuçları işlenmek üzere histogram ile pdf kestirimi bloğuna gönderilir. Monte Carlo simülasyonlarında, üretilen dışsal bilgiye ait LLR değerleri, mesaj bitinin $x = 0$ ve $x = 1$ değerleri için ikiye ayrılır. Bundan sonra bu değerlere Monte Carlo simülasyonları uygulanarak $p(E|X = -1)$ ve $p(E|X = +1)$ için dağılımlar bulunur. Bu dağılımlar bulunduktan sonra her bir I_B değerine karşılık gelen I_S değeri Eşitlik 5.25 yardımıyla hesaplanır. Bu hesaplanan I_S değerleri kapalı formdaki Eşitlik 5.35'de yerine konulur ve A bloğuna ait EXIT eğrisi de çizdirilir.

EXIT grafikleri üzerindeki eniyileme ise, LDPC kodların EXIT grafiklerinde yapılan eniyileme ile benzerlik göstermektedir. Aynı LDPC EXIT grafiklerinde olduğu

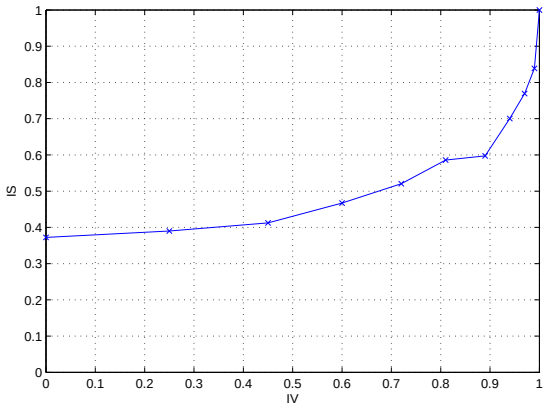
gibi buradaki amaç da bu iki bloğa ait EXIT eğrileri arasındaki *tüneli* açık tutmaktır. Alıcı bloklarına ait bu iki eğri birbirini kesmeyecek şekilde ancak birbirine en yakın noktada tutularak, alıcının en düşük SNR seviyesinde çözümleme yapması sağlanır [8].

Bu tez çalışmasında [8] numaralı referansta yapılan çalışmalar sonucunda elde edilen değişken ve kontrol düğümlerine ait derece dağılımları kullanılacak ve bu derece dağılımlarına ait EXIT grafikleri metodu ile eniyileme yapılacak ve alıcının SNR'a karşılık BER performansı incelenecektir. Eşitlik 5.37'de, [8] numaralı referansta kullanılan derece dağılımları gösterilmiştir. Bu dağılımlar kullanılarak kodun tanımlanması (yani H matrisinin oluşturulması) için rastgele oluşturma tekniği kullanılmıştır. Rastgele oluşturma tekniği [16] nolu referansta açıklanmıştır.

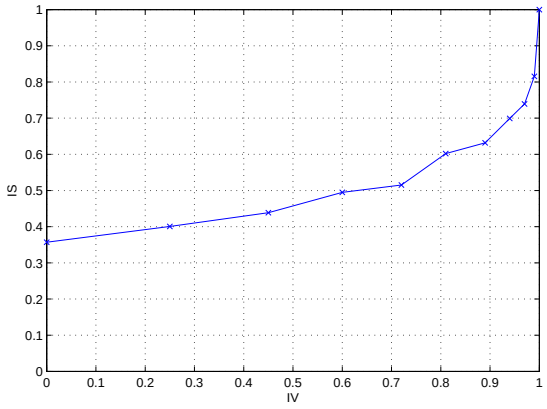
$$\begin{aligned} \rho_3 &= 0.3157, \qquad \lambda_2 = 0.5473 \\ \rho_4 &= 0.2259, \qquad \lambda_3 = 0.0116 \\ \rho_8 &= 0.0273, \qquad \lambda_4 = 0.4411 \\ \rho_{15} &= 0.4311 \end{aligned} \tag{5.37}$$

Bu dağılımlar öncelikle Monte Carlo simülasyonları kullanılarak I_S değerinin hesaplanması için BCJR-YGYÇ çözücüsüne uygulanacak ve bulunan bu I_S değerleri Eşitlik 5.35'de yerine konularak A bloğuna ait EXIT grafiği çizdirilecektir. Eşitlik 5.34 kullanılarak B bloğuna ait EXIT grafiği çizdirilir ve A bloğuna ait EXIT grafiği yüksek SNR'lardan başlanarak yavaş yavaş düşürülür ve B bloğuna ait EXIT eğrisi ile aralarında en düşük seviyede boşluk kalıncaya kadar yaklaştırılır. Simülasyonlar sonucunda bazı SNR değerlerinde elde edilen I_V değerlerine karşılık I_S değerleri, Şekil 5.10, 5.11 ve 5.12'de gösterilmiştir.

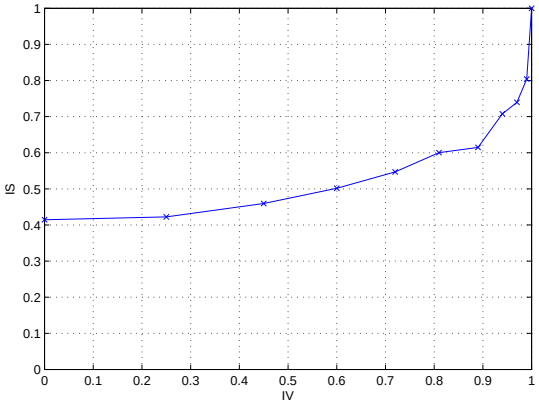
Şekil 5.10, 5.11 ve 5.12'de edilen I_S değerleri için dizayn edilen alıcıya ait EXIT grafikleri sırası ile Şekil 5.13, 5.14 ve 5.15'te verilmiştir.



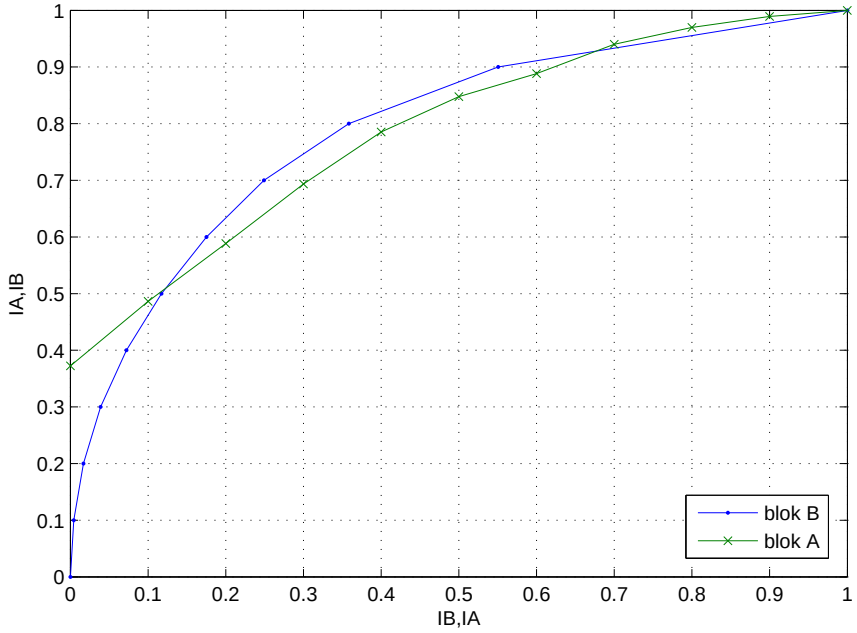
Şekil 5.10: SNR=0.8 dB değeri için $I_S vs I_V$ grafiği



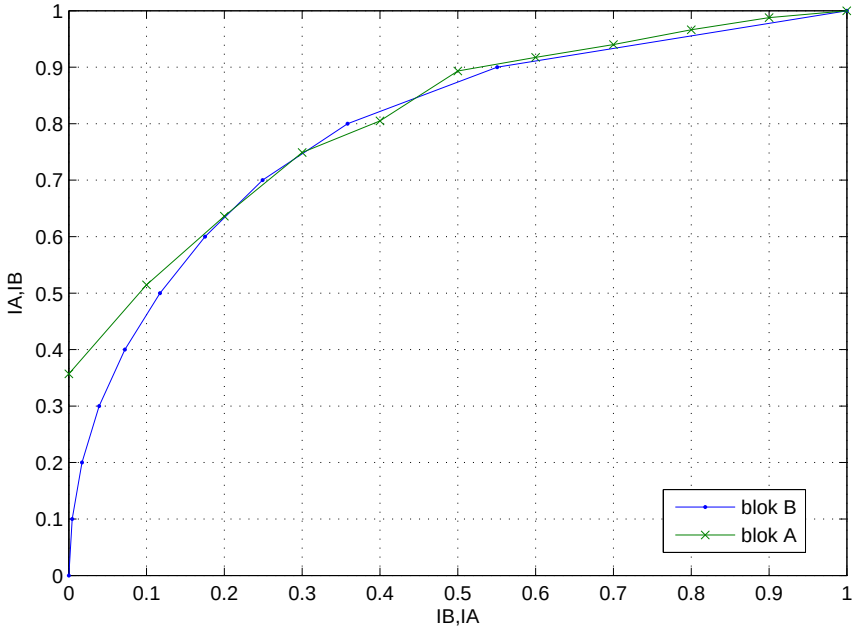
Şekil 5.11: SNR=1 dB değeri için $I_S vs I_V$ grafiği



Şekil 5.12: SNR=1.2 dB değeri için $I_S vs I_V$ grafiği

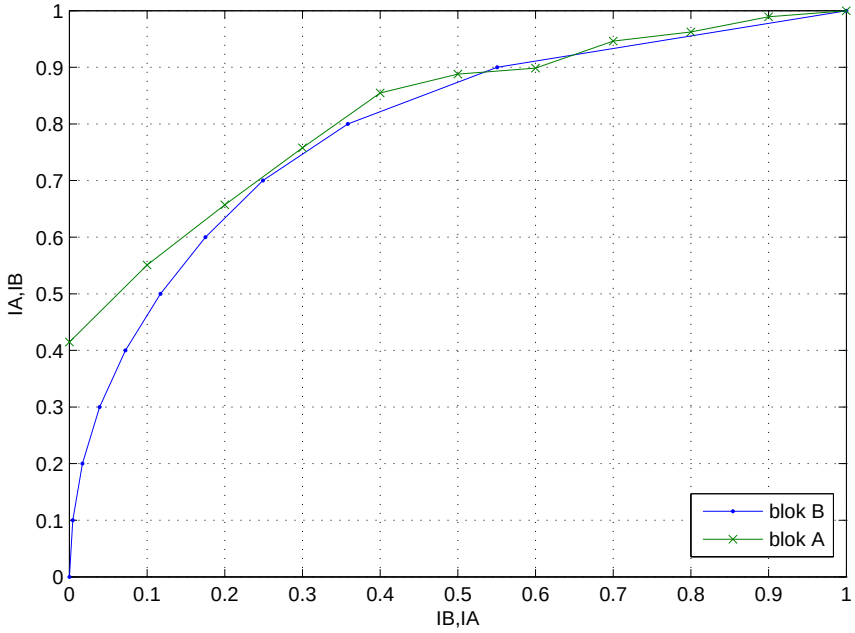


Şekil 5.13: Şekil 5.10'daki I_S değerleri ile çizdirilen EXIT grafiği



Şekil 5.14: Şekil 5.11'deki I_S değerleri ile çizdirilen EXIT grafiği

Şekil 5.14'te elde edilen EXIT grafiği eğriler arasındaki tünelin en düşük seviyede olduğu grafiktir ve bu grafik SNR=1 dB için çizdirilmiştir. Bu grafikte A bloğuna ait EXIT eğrisinin, B bloğuna ait EXIT eğrisini kestiği görülmekte ancak



Şekil 5.15: Şekil 5.12'deki I_S değerleri ile çizdirilen EXIT grafiği

özellikle bu grafiğin Şekil 5.15 ile karşılaştırılmasından bu kesişmenin histograma ait integralleme sırasındaki hatadan veya analizdeki rastgele değişkenlerden kaynaklanan bir hata olduğu değerlendirilmektedir. Dolayısı ile alıcının şelale bölgesine girdiği eşik değeri 1 dB'dir. Bu EXIT grafiklerine ait alıcının performans grafikleri bir sonraki bölümde verilecektir.

5.3.1 Eniyileme Yapılmış Alıcıya Ait BER-SNR Grafikleri

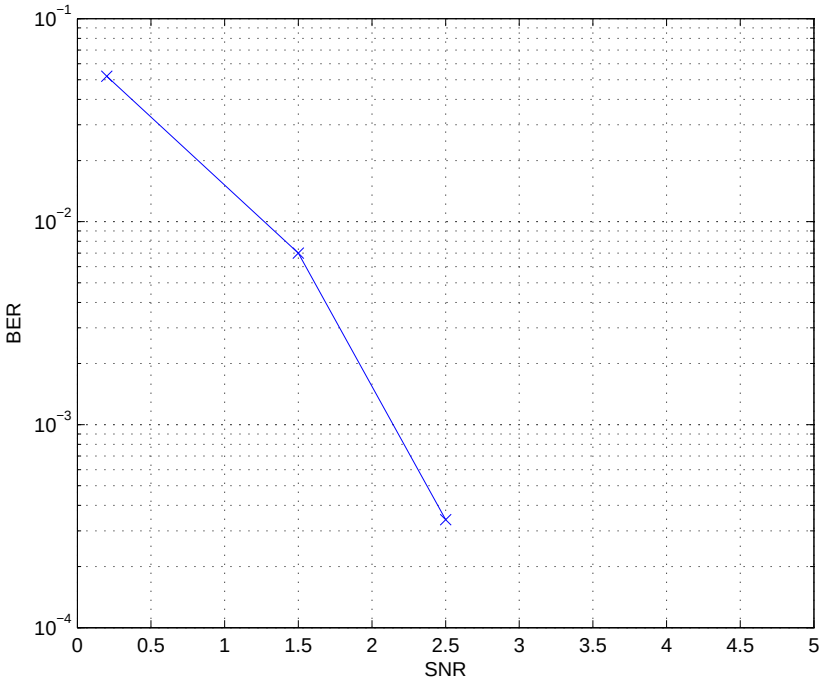
Bu bölümde, bir önceki bölümde $I_S - I_V$ grafikleri verilen LDPC dağılımlarının SNR'a karşılık BER performansları gösterilecek ve bu grafiklerle ilgili çıkarımlar yapılacaktır.

Aşağıdaki grafikte eniyileme yapılmış düzensiz LDPC kodlara ait sistem performansı gözlemlenecektir. Düzensiz LDPC kodlarına ait dağılımlar için yapılan eniyileme çalışmaları [8] nolu referansta yapılmış ve bu eniyileme çalışmaları sonucunda elde edilen dağılımlar Eşitlik 5.37'de verilmiştir. Sisteme ait diğer parametreler Çizelge 5.1'de verilmiştir. Sisteme ait BER-SNR grafiği Şekil 5.16'da verilmiştir.

Ayrımsal çözme sürecini sadece bir kere gerçekleştirerek ($N = 1$), döngüselliği LDPC çözücüsünde sağladığımızda elde edilen BER-SNR grafiği Şekil 5.17'de verilmiş-

Parametre	Değeri
Mesaj Uzunluğu	600
Kod Uzunluğu	1200
Kod Oranı	0.5
N	10
N_{LDPC}	20
$\lambda(x)$	$0.5473x^2 + 0.0116x^3 + 0.4411x^4$
$\rho(x)$	$0.3157x^3 + 0.2259x^4 + 0.0273x^8 + 0.4311x^{15}$

Çizelge 5.1: Şekil 5.16 için oluşturulan sisteme ait parametre değerleri



Şekil 5.16: Çizelge 5.1'de parametreleri verilen alıcıya ait BER-SNR grafiği

tir. Sisteme ait diğer parametreler Çizelge 5.2'de verilmiştir.

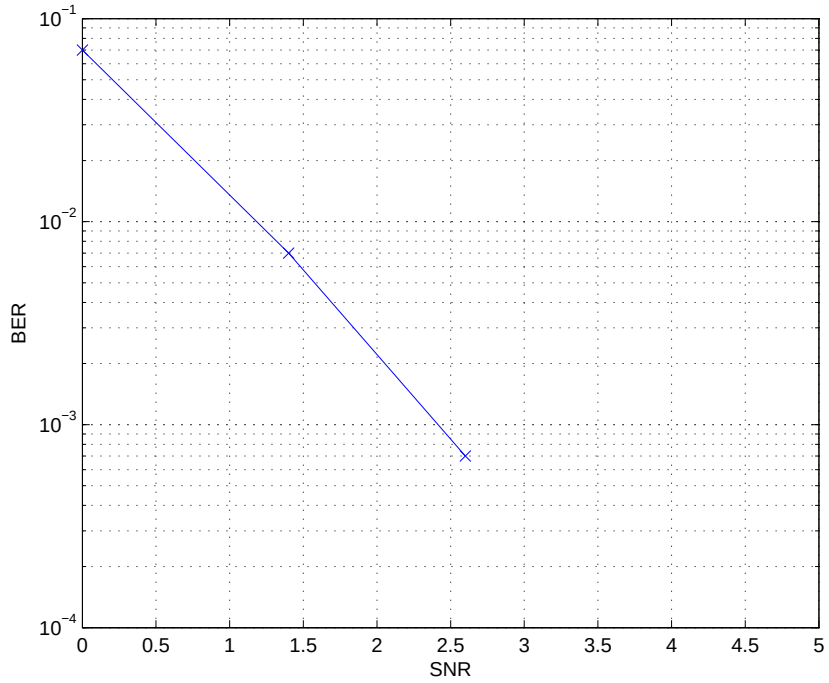
Şekil 5.17'deki BER-SNR grafiği düzenli kodlarda elde edilen sonuçlara yakın sonuçlar içermektedir ancak EXIT grafiklerinde alıcı için eniyileme yapıldığından düzensiz kodların daha iyi sonuçlar vermesi beklenir.

LDPC çözücüsü için döngü sayısını 1 yaparak döngüselliği bütün alıcıda sağladığımızda elde edilen BER-SNR grafiği Şekil 5.18'de verilmiştir. Sisteme ait diğer parametreler Çizelge 5.3'te verilmiştir.

EXIT grafiği metodu ile eniyileme çalışmaları, A ve B blokları için

Parametre	Değeri
Mesaj Uzunluğu	600
Kod Uzunluğu	1200
Kod Oranı	0.5
N	1
N_{LDPC}	40
$\lambda(x)$	$0.5473x^2 + 0.0116x^3 + 0.4411x^4$
$\rho(x)$	$0.3157x^3 + 0.2259x^4 + 0.0273x^8 + 0.4311x^{15}$

Çizelge 5.2: Şekil 5.17 için oluşturulan sisteme ait parametre değerleri



Şekil 5.17: Çizelge 5.2'de parametreleri verilen alıcıya ait BER-SNR grafiği

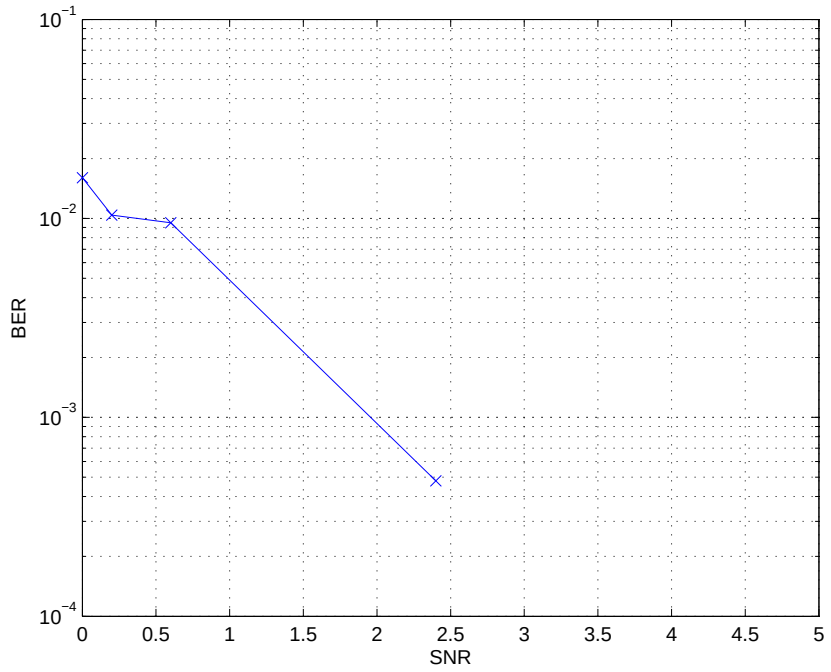
yapıldığından, N_{LDPC} değeri 1 alınarak ve N değeri yüksek bir değer seçilerek daha iyi sonuçlar elde edildiği görüldü.

En son olarak alıcının döngü sayısını (N), LDPC bloğunun döngü sayısından (N_{LDPC}) daha yüksek tutarak alıcının performansı gözlenecektir. Buna ait BER-SNR grafiği Şekil 5.19'da ve alıcıya ait parametreler Çizelge 5.4'te verilmiştir.

Şekil 5.19'dan elde edilen grafiğe bakılarak, alıcıya ait en iyi performansın Çizelge 5.4'teki parametrelerle alındığı gözlemlenmiştir. EXIT grafik metodunda bulunan SNR eşik değeri olan 1 dB ise burada açıkça görülmektedir. EXIT grafiği ile

Parametre	Değeri
Mesaj Uzunluğu	600
Kod Uzunluğu	1200
Kod Oranı	0.5
N	40
N_{LDPC}	1
$\lambda(x)$	$0.5473x^2 + 0.0116x^3 + 0.4411x^4$
$\rho(x)$	$0.3157x^3 + 0.2259x^4 + 0.0273x^8 + 0.4311x^{15}$

Çizelge 5.3: Şekil 5.18 için oluşturulan sisteme ait parametre değerleri



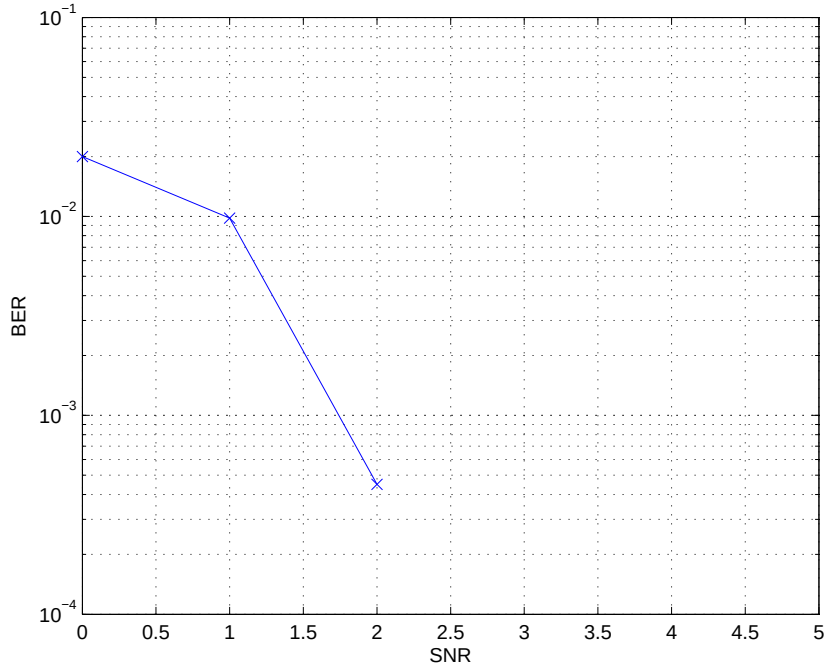
Şekil 5.18: Tablo 5.3'te parametreleri verilen alıcıya ait BER-SNR grafiği

yapılan eniyileme bütün alıcıda (A ve B blokları için) yapılmasına rağmen Şekil 5.18, Şekil 5.19 ile karşılaştırıldığında LDPC çözümlemesine ait döngü sayısının (N_{LDPC}) değerini 1 almak yerine, bu değeri bütün alıcının döngü sayısı olan N değerinden daha küçük ancak 1'den daha büyük bir değer almanın alıcının performansını artırdığı açıkça gözlemlenmiştir.

Döngü sayılarını ifade eden N ve N_{LDPC} değerleri ile BER arasındaki ilişkinin daha iyi analiz edilebilmesi amacıyla Şekil 5.20'de, SNR=1.5 dB değeri için bir grafik çizdirilmiştir. Alıcının oluşturulması için kullanılan parametreler, Çizelge 5.4'te kul-

Parametre	Değeri
Mesaj Uzunluğu	600
Kod Uzunluğu	1200
Kod Oranı	0.5
N	30
N_{LDPC}	5
$\lambda(x)$	$0.5473x^2 + 0.0116x^3 + 0.4411x^4$
$\rho(x)$	$0.3157x^3 + 0.2259x^4 + 0.0273x^8 + 0.4311x^{15}$

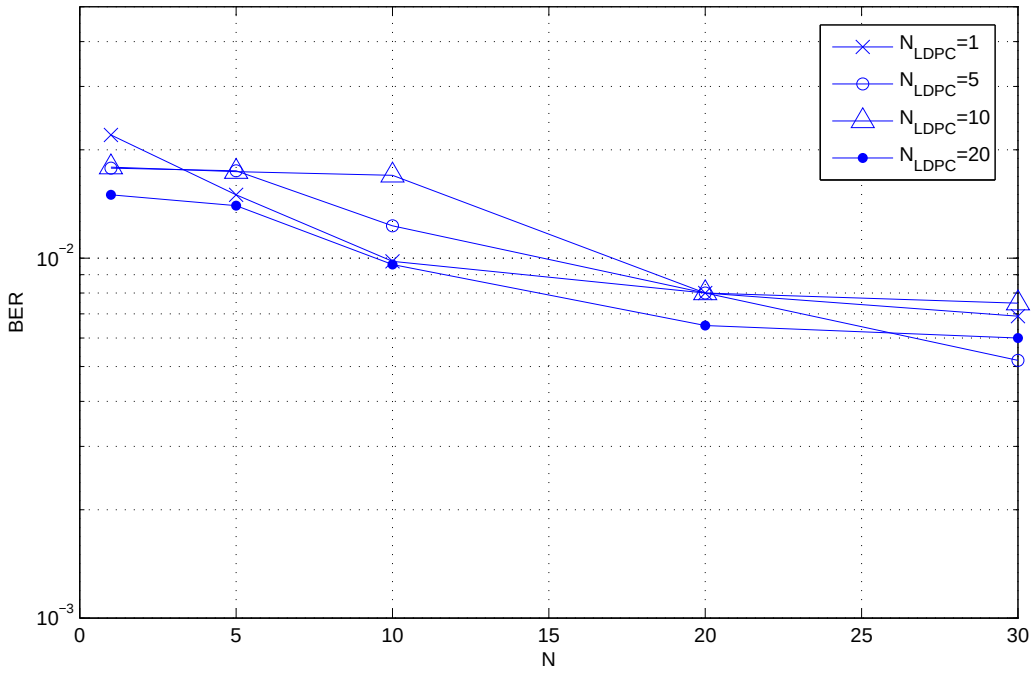
Çizelge 5.4: Şekil 5.19 için oluşturulan sisteme ait parametre değerleri



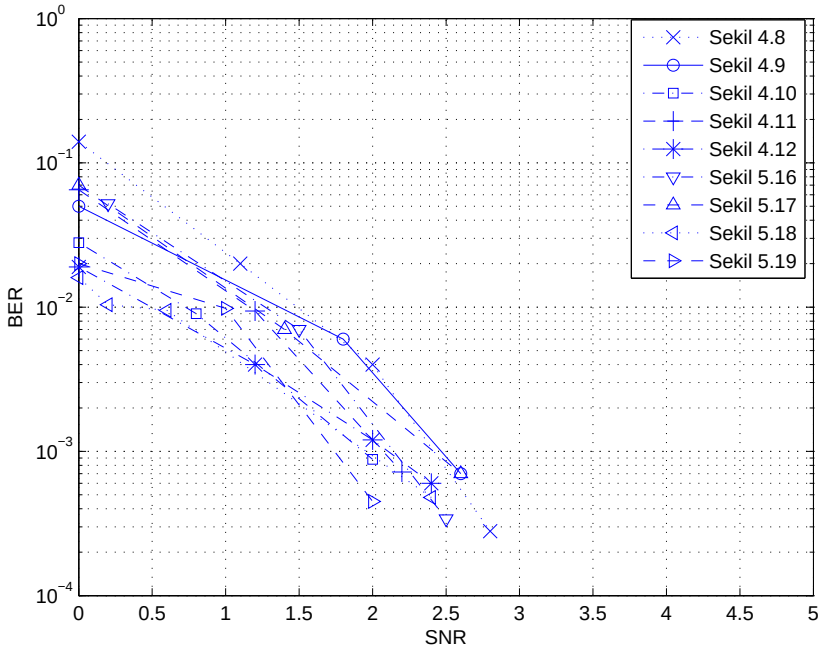
Şekil 5.19: Tablo 5.4'te parametreleri verilen alıcıya ait BER-SNR grafiği

lanılan parametreler (N ve N_{LDPC} hariç) ile aynıdır. Bu grafikte, N ve N_{LDPC} değerlerinin yüksek değerleri için, BER'in belli bir değere yakınsadığı görülmektedir. Bu grafikten, Şekil 5.19'un çizilmesi için kullanılan N ve N_{LDPC} değerlerinin en iyi değerlere yakın olduğu değerlendirilmiştir.

Şekil 4.8, 4.9, 4.10, 4.11, 4.12, 5.16, 5.17, 5.18 ve 5.19'da çizilen BER-SNR eğrileri, bu eğriler arasında karşılaştırma yapılabilmesi amacıyla Şekil 5.21'de birleştirilerek çizilmiştir.



Şekil 5.20: SNR=1.5 dB için döngü sayıları analiz grafiği



Şekil 5.21: Karşılaştırma için birleştirilmiş BER-SNR grafikleri

6. SONUÇ

Kodlama teorisi, sayısal haberleşmede her geçen gün önem kazanmaktadır. Haberleşmenin esası, kaynaktaki bilgiyi hedefe kadar hatasız taşımak olduğundan kodlama teorisi bu noktada haberleşme sistemlerinin performanslarını iyileştirmektedir. Bu tez çalışmasında da kodlama ve kiplemeyi birleştiren bir haberleşme sistemi ele alınmıştır.

Yapılan çalışmalar, şu anda var olan en iyi kodlar arasında olan LDPC kodlama ile ayrımsal kiplenimin incelenmesine yöneliktir. Bunu incelemenin motivasyonu, literatürde kodlama ve ayrımsal kiplenimin uygun şekilde yapıldığı zaman iyi sonuçlar verdiğinin görülmüş olmasıdır. Referans [8]'de de önerilmiş olan bu birleştirmenin detayları bu tez çalışmasında incelenmiş ve tezde sunulmuştur.

Ele alınan iletişim sistemi modelindeki temel bloklar, vericide LDPC kodlayıcı ve DPSK kipleyici, alıcıda ise döngüsel olarak çalışan APP kipçözücü ve LDPC kod çözücüsüdür. Tezde ilk olarak bu blokların her birinin nasıl işlediği anlatılmıştır. APP kipçözücünün çalışma prensibini oluşturan ileri-geri algoritması çıkarılmıştır. LDPC kodlar, çözümleme algoritması olan topla-çarp algoritması ve bu algoritmanın dayandığı grafiksel modeller anlatılmıştır. APP kipçözücü ve LDPC kod çözücüsünün değişken ve kontrol düğümlerinin döngüsel çalışmasına ve aralarındaki bilgi alışverişine dayanan döngüsel alıcı yapısı çıkarılmıştır. Bunlara karşılık gelen ve SNR'a karşılık BER sonuçlarını veren simülasyonlar yapılmıştır.

Simülasyonlarda, LDPC kodlarına eniyileme yapılmasının, ele alınan yapının performansını iyileştirdiği görülmüştür. Eniyileme, LDPC kodun eşitlik kontrol matrisindeki 1'lerin yoğunluğunu gösteren derece dağılımlarını seçerek yapılır. Eniyileme yöntemlerinden biri olan EXIT grafiği yöntemi, döngüsel APP kipçözücüsü ve LDPC kod çözücüsü için bu tezde çıkarılmıştır. En iyi LDPC kodu elde etmek için EXIT grafiği yöntemi kullanılarak elde edilen ve [8] nolu referansta verilen derece dağılımları kullanılmıştır.

Simülasyonlarda ayrıca, APP kipçözücü ile LDPC değişken düğümleri arasında yapılan bir döngüye karşılık, LDPC değişken ve kontrol düğümleri arasında birden fazla döngü yapmanın performansı iyileştirdiği görülmüştür. Bunun yanı sıra, en iyi alıcı performansının, LDPC değişken ve kontrol düğümleri arasında döngüselliğin sağlanmasıyla birlikte döngüselliğin bütün alıcıda sağlandığı durumda

olduğu görülmüştür. Bununla birlikte, kodun uzunluğunu arttırmanın, beklenildiği gibi, performansı arttırdığı görülmüştür.

KAYNAKLAR

[1] Aktaş,E., Evans,J.,Hanly,S., 2004, Distributed Decoding In A Cellular Multiple-Access Channel , IEEE International Symposium on Information Theory, 5-6

[2] Bahl,L.R., Cocke,J.,Jelinek,F.,Raviv,J., 1974, Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate , IEEE Transaction on Information Theory, 284-287

[3] Brink, S.ten, 1999, Convergence of Iterative Decoding, Electronic Letter, 806-808

[4] Brink, S.ten, 2001, Convergence Behaviour of Iteratively Decoded Parallel Concatenated Codes, IEEE Transactions on Communications, 49

[5] Brink, S.ten,Kramer G.,Ashikhmin, A., 2004, Design of Low Density Parity Check Codes for Modulation and Detection, IEEE Transactions on Communications, 52

[6] Dauwels, H.G., 1977, On Graphical Models for Communications and Machine Learning, MIT Press, Doctorate Thesis, Swiss Federal Institute of Technology, 43-54

[7] Forney, G.D Jr., 1966, Concatenated Codes, MIT Press

[8] Franceschini,M., Ferrari,G., Raheli,R., Curtoni,A., 2005, Serial Concatenation of LDPC Codes and Differential Modulation, IEEE Journal on Selected areas in Communications, 23, 9, 1758-1768

[9] Gallager, R.G., 1962, Low Density Parity Check Codes, IRE Transactions Information Theory, 8, 21-28

[10] Gallager, R.G., 1963, Low Density Parity Check Codes, MIT Press

[11] Haykin, S., 2001, Communication Systems, John Wiley& Sons

[12] Howard,S.L.,Schlegel,C., 2006, Differential Turbo-Coded Modulation with APP Channel Estimation, 54

[13] Johnson, S.J.,Weller,S.R., 2002, Low Density Parity Check Codes:Design and Decoding, Technical Report, University of Newcastle, 11-18

[14] Kay,S.M., Fundamentals of Statistical Signal Processing:Estimation Theory, Prentice Hall, 205-207

[15] Kschischang, F.R.,Frey,B.J,Loeliger H.A., 2001 ,Factor Graphs and the Sum-Product Algorithm, IEEE Transactions on Information Theory,47,498-519

[16] Lin,S., Costello,D., 2004, Error Control Coding, Prentice Hall

[17] Loeliger,H.A., 2004, An Introduction to Factor Graphs, IEEE Signal Processing Magazine, 28-41

[18] Mitra,J., Lampe,L., 2006, Simple Concatenated Codes Using Differential PSK, IEEE Global Telecommunications Conference, 1-6

[19] Moreira,J.C.,Farrell,P.G., 2006, Essentials of Error Control Coding, John Wiley&Sons, 257-269

[20] Narayanan, K.R.,Stuber,G.L., 1998, A Serial Concatenation Approach to Iterative Demodulation and Decoding, IEEE Transactions on Communications, 145, 53-59

[21] Proakis, J.G., 2001, Digital Communications, McGraw Hill

[22] Richardson,T., Urbanke,R., Modern Coding Theory, Cambridge University Press, 77-80

[23] Shannon,C.E., 1956, The Zero-Error Capacity of A Noisy Channel, IRE Transactions Information Theory 2, 8-19

[24] Sweeney, P, 2002, Error Control Coding Theory-From Theory to Practice, John Wiley&Sons

[25] Tanner, R.M., 1981, A Recursive Approach to Low-Complexity Codes, IEEE Transactions on Information Theory, 27, 533-547

EKLER

EK 1: HİSTOGRAM İLE PDF KESTİRİMİ

Bu ekte histogram ile pdf kestirimi metodu hakkında bilgi verilecektir. Histogram ile pdf kestirimi metodu, Monte Carlo simülasyonlarının bir parçasıdır. Histogram ile pdf kestirimi metodu, rastgele değişkenlerin olasılık yoğunluk fonksiyonlarını simülasyon yöntemi ile bulmaya yarayan bir araçtır [14].

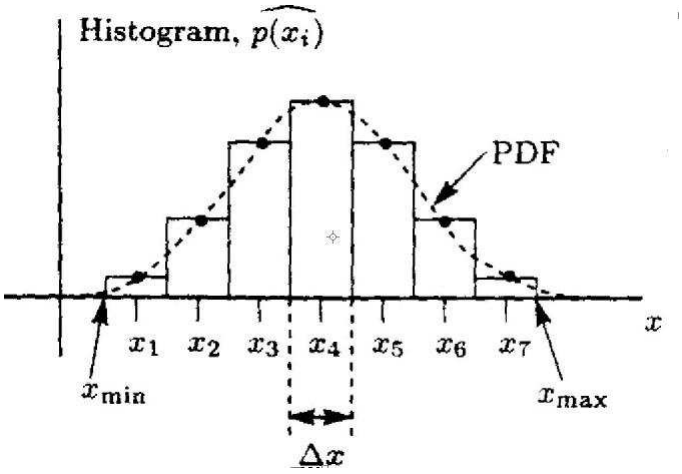
Burada histogram ile pdf kestirimi metodu tanımı aşağıda verilen A rastgele değişkeninin gerçeklenmesi ile bulunacaktır.

$$A' = -\frac{1}{2} + \sqrt{\frac{1}{N}\sum_{n=0}^{N-1} x^2[n] + \frac{1}{4}} \tag{A-1}$$

Burada $x[n] = A + w[n]$'dur ve $w[n]$ varyansı $A > 0$ olan Gaussian gürültüdür. A'nın ortalaması, varyansı ve olasılık yoğunluk fonksiyonunun nasıl belirleneceği anlatılacaktır. Burada anlatılacaklar çok genel olduğundan her türlü tahmin edilen rastgele değişkene uygulanabilir. Bu metoddaki adımlar aşağıda sıralanmıştır:

- N adet birbirinden bağımsız sabit rastgele değişken üretilir.
- Bu sabit rastgele değişkenlerden $w[n]$'i üretmek için Gaussian rastgele değişkenler üretilir.
- $x[n]$'i bulmak için A $w[n]$'e toplanır ve A' hesaplanır.
- Prosedür M kez tekrarlanır ve M adet A' değeri hesaplanır.
- Olasılık yoğunluk fonksiyonunun kestirimi histogram ile bulunur.

A''a ait olasılık yoğunluk fonksiyonunun bulunması için tahmin edilen olasılık yoğunluk fonksiyonu olan *histogram* kullanılır. Histogram pdf'i, A''ın belirli bir aralığa ne kadar düştüğünü bularak tahmin eder. Daha sonra toplam gözlem sayısına ardından da aralık uzunluğuna bölüm pdf'i normalize eder ve tahmin edilen pdf'i verir. Buna ait örnek Şekil A.1'de gösterilmiştir. Bu şekilde pdf (x_{min}, x_{max}) aralığında tahmin



Şekil A.1: Bilgisayarda üretilmiş veri ile hazırlanan histogram (Bu şekil [14] nolu referanstan alınmıştır.)

edilmiştir. Her bir alt aralık $(x_i - \triangle x/2, x_i + \triangle x/2)$hücre olarak adlandırılır. Her bir $i.$ aralıkdaki değer

$$p(x_i) = \frac{L_i/M}{\triangle x} \tag{A-2}$$

değerine eşit olur. Burada L_i, x_i'ın i. hücreye düşmesinin sayısıdır. Böylelikle, L_i/M x_i'ın i. hücreye düşme olasılığını tahmin eder ve bu değerin $\triangle x$ değerine bölünmesi gözlenen değişkenin olasılık yoğunluk fonksiyonunu verir.

EK 2 J ve J^{-1} FONKSİYONLARI

$Y = X + N$ olduğu ve $Pr(X = \pm 1) = 1/2$ ve N'in sıfır ortalamalı, varyansı σ_n^2 olan Gaussian gürültü olduğu düşünülürse, kanaldan gelen LLR değerleri y'nin bir fonksiyonudur ve bu $L_{ch}(y)$ olarak yazılır. $X = \pm 1$ üzerinde koşullu olan $L_{ch}(y)$ değeri, ortalaması $\mu_{ch} = \pm 2/\sigma_n^2$ ve varyansı $\sigma_{ch}^2 = 4/\sigma_n^2$ olan Gaussian değerdir. Böylece $\mu_{ch} = \frac{\pm\sigma_{ch}^2}{2}$ olur [5].

$J(\sigma_{ch})$, $I(X; L_{ch}(Y))$ ortak bilgisini tanımlasın:

$$\begin{aligned} J(\sigma_{ch}) &= H(X) - H(X|L_{ch}(Y)) \\ &= 1 - \int \frac{1}{\sqrt{2\pi}\sigma_{ch}} e^{-(\xi - \sigma_{ch}^2/2)^2/2\sigma_{ch}^2} log_2(1 + e^{-\xi}) d\xi \end{aligned} \tag{B-1}$$

Burada $H(X)$ X'in entropisi ve $H(X|L_{ch}(Y))$'da X'in $L_{ch}(Y)$ üzerine koşullandırılmış entropisidir. Ayrıca $I(X; L_{ch}(Y))$ $I(X; Y)$ ile aynıdır.

Bilgisayar hesaplamaları için, J fonksiyonu $0 \leq \sigma \leq \sigma^*$ ve $\sigma > \sigma^*$ aralıklarında ikiye bölünebilir. Bu aralıklarda $\sigma^* = 1.6363$ değerine eşittir. Soldaki aralık için bir polinom ve sağdaki aralık için eksponensiyel bir fonksiyon yaklaşımı kullanılır. Böylelikle J fonksiyonu aşağıdaki şekilde hesaplanabilir [5]:

$$J(\sigma) \approx \begin{cases} a_{J,1}\sigma^3 + b_{J,1}\sigma^2 + c_{J,1}\sigma, & 0 \leq \sigma \leq \sigma^* \\ 1 - e^{a_{J,2}\sigma^3 + b_{J,2}\sigma^2 + c_{J,2}\sigma + d_{J,2}}, & \sigma^* < \sigma < 10 \\ 1, & \sigma \geq 10 \end{cases} \tag{B-2}$$

Yukarıdaki denklemlerde yer alan katsayılar ise şöyledir [5]:

$$\begin{aligned} a_{J,1} &= -0.0421061, \qquad a_{J,2} = 0.00181491 \\ b_{J,1} &= 0.209252, \qquad b_{J,2} = -0.142675 \\ c_{J,1} &= -0.00640081, \qquad c_{J,2} = -0.0822054 \\ d_{J,2} &= 0.0549608 \end{aligned}$$

J'in ters fonksiyonunun hesaplanması için eğri $I^* = 0.3646$ değerinde iki aralığa bölünür ve bu aralıklarda J^{-1} şu şekilde hesaplanır [5]:

$$J^{-1}(I) \approx \begin{cases} a_{\sigma,1}I^2 + b_{\sigma,1}I + c_{\sigma,1}\sqrt{I}, & 0 \leq I \leq I^* \\ -a_{\sigma,2}ln[b_{\sigma,2}(1 - I)] - c_{\sigma,2}I, & I^* < I < 1 \end{cases} \tag{B-3}$$

Yukarıdaki denklemlerde yer alan katsayılar şu şekildedir [5]:

$$
\begin{aligned}
a_{\sigma,1} &= 1.09542, \qquad a_{\sigma,2} = 0.706692 \\
b_{\sigma,1} &= 0.214217, \qquad b_{\sigma,2} = 0.386013 \\
c_{\sigma,1} &= 2.33727, \qquad c_{\sigma,2} = -1.75017
\end{aligned}
$$

ÖZGEÇMİŞ

Adı Soyadı	:	ERSİN BARAN AKKUŞ
Doğum Yeri	:	ANKARA
Doğum Yılı	:	1983
Medeni Hali	:	Bekar

Eğitim ve Akademik Durumu

Lise 1994-2001	:	Kalaba Anadolu Lisesi, ANKARA
Lisans 2001-2005	:	Hacettepe Üniversitesi Elektrik ve Elektronik Mühendisliği Bölümü, ANKARA
Yabancı Dil	:	İngilizce

İş Tecrübesi

2005-...	:	Aydın Yazılım ve Elektronik Sanayi A.Ş ANKARA Sistem Mühendisi