

YILDIZ TEKNİK ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

Birleşik Kod. Mod. Tek. Day. bir
Oto. Yin. İst. Düz. Baş. Çöz.

Yüksek Lisans Tezi

Raci Algin

1992

2/167

YILDIZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİRLEŞİK KODLAMALI MODÜLASYON TEKNİĞİNE DAYALI BİR
OTOMATİK YİNELEME İSTEME DİZGESİNİN BAŞARIM ÇÖZÜMLEMESİ

YÜKSEK LİSANS TEZİ

MÜH. RACİ ALGIN

İSTANBUL 1992

YILDIZ TEKNİK ÜNİVERSİTESİ
KÜTÜPHANE DOKÜMANTASYON
DAİRE BAŞKANLIĞI

Kot	: R 152 167
Alındığı Yer	: ...Fen.Bil.Ens.....
Tarih	: ...12.4.95.....
Fatura	:
Fiyatı	: 186.000.-
Ayniyat No	: 1-6
Kayıt No	: 50996
UDC	:
Ek	:

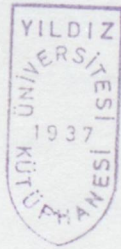
YILDIZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ



BİRLEŞİK KODLAMALI MODÜLASYON TEKNİĞİNE DAYALI BİR
OTOMATİK YİNELEME İSTEME DİZGESİNİN BAŞARIM ÇÖZÜMLEMESİ

YÜKSEK LİSANS TEZİ

MÜH. RACİ ALGIN



İSTANBUL 1992

İÇİNDEKİLER

1. Giriş	1
1.1 Konuyla ilgili yapılmış çalışmalar	7
1.2 Tezin amacı ve izlenecek yol	10
2. Otomatik yinelenme istemeli (OYİ) dizgeleri	12
2.1 Temel OYİ dizgeleri	13
2.1.1 Bur ve Debie OYİ dizgesi	13
2.1.2 H-1018 çarfi dönmeli OYİ dizgesi	15
2.1.3 Adecerek yinelenmeli OYİ dizgesi	16
2.1.4 Temel OYİ dizgelerinin güvenilirliği	18
2.2 Karma OYİ dizgeleri	20
Bu tezin hazırlanmasında bilgi ve uyarılarıyla yol gösteren Tez Yöneticim Yrd. Doç. Dr. Ümit AYGÖLÜ 'ne ve tezin yazılmasında değerli yardımlarıyla destek olan başta eşim Gülnur olmak üzere kardeşlerim ve dostlarıma teşekkürü borç bilirim.	22
3. HPKA OYİ dizgesinin yapısı	24
3.1 HPKA OYİ dizgesinin analitik yolla hazırlanması	26
3.2 HPKA OYİ dizgesinin belirlenmesi	28
3.3 HPKA OYİ dizgesinin Viterbi Algoritması ve bu algoritmanın başarısına etkileri	30
3.4 HPKA OYİ dizgesinin bellek kullanılması durumunda HPKA OYİ dizgesinin başarısı	32

İÇİNDEKİLER

. Özet	
5. Summary	
1. Giriş	1
1.1 Konuyla ilgili yapılmış çalışmalar	7
1.2 Tezin amacı ve izlenecek yol	10
2. Otomatik yineleme istemeli (OYİ) dizgeler	12
2.1 Temel OYİ dizgeler	13
2.1.1 Dur ve bekle OYİ dizgesi	13
2.1.2 N-Adım geri dönmeli OYİ dizgesi	15
2.1.3 Seçerek yinelemeli OYİ dizgesi	16
2.1.4 Temel OYİ dizgelerinin güvenilirliği	
ve iş çıkarma yeteneği	25
2.2 Karma OYİ dizgeler	30
3. Birleşik kodlamalı modülasyon tekniğine dayalı OYİ dizgeler	34
4. Hızlı frekans kaydırmalı anahtarlama tekniği kullanılan birleşik kodlama modülasyonlu bir OYİ dizgenin tasarımı	48
4.1 HFKA OYİ dizgesinin yapısı	50
4.2 HFKA OYİ Dizgesinin analitik yolla hata sınırlarının belirlenmesi	58
4.3 Yineleme istemeli Viterbi Algoritması ve bu algoritmanın başarıma etkileri	64
4.4 Alıcıda bellek kullanılması durumunda HFKA OYİ dizgesinin başarıımı	70

5. Hızlı frekans kaydırmalı anahtarlama tekniği
kullanan birleşik kodlama modülasyonlu OYİ dizgenin ba-
şarım çözümlemesi 80

6. Sonuç 100

7. Ek 102

. Kaynakça 112

. Özgeçmiş

kuramsal olarak elde edilmiş, karşılaştırmalar yapılmıştır.

Alıcı tarafta bellek kullanılarak aynı bir kod sözcük dizisinin her bir iletiminde belirli giriş dizilerine olan

Öklid Bu tezde, birleşik kodlamalı modülasyon tekniğine

dayalı yüksek iş çıkarma yeteneğine sahip, yüksek

güvenilirlikli yeni bir otomatik yineleme istemeli dizge

tasarlanmıştır. Modülasyon için, sabit zarf ve sürekli faz

özellikleriyle, doğrusal olmayan kuvvetlendirme içeren band

sınırlı kanallarda, çekici bir yöntem olan hızlı frekans

kaydırmalı anahtarlama tekniği kullanılmıştır. İlki;

demodüle edilmiş dizilerin hatalarını sezen $R=3/6$ oranlı

doğrusal blok kodu, ikincisi; modüle edilmiş kanal

dizileri arasındaki Öklid uzaklıklarını artırarak hata

başarımını yükseltmeyi amaçlayan $R=1/2$ oranlı bir kafes

kodlayıcı olmak üzere iki kodlayıcı kullanılmaktadır.

Alıcıda, gürültülü kanal çıkış simgeleri, yumuşak kararlı

Viterbi Algoritmasıyla çözülmektedir. "Dur ve bekle", "N

adım geri dön", "seçerek yineleme" gibi temel otomatik

yineleme istemeli yapılar için iş çıkarma yeteneği ve

güvenilirlik çözümlemesini gerçekleştirmek üzere bir

benzetim modeli oluşturulmuştur. Bu modelde iş çıkarma

yeteneğini biraz azaltan ancak güvenilirliği daha da

yükselten yineleme isteme için geliştirilmiş bir Viterbi

Algoritması da göz önünde bulundurulmuş, ayrıca "dur ve

bekle" yapısı için iş çıkarma yeteneği üst sınırları

kuramsal olarak elde edilmiş, karşılaştırmalar yapılmıştır. Alıcı tarafta bellek kullanılarak aynı bir kod sözcük dizisinin her bir iletiminde olurlu giriş dizilerine olan Öklid uzaklığı bilgilerini saklayıp, birikmiş Öklid uzaklıklarını toplayarak, Viterbi Algoritmasıyla daha sağlıklı karar verilmesi olanağı sağlanmıştır.

combined coding-modulation technique. The modulation used in the proposed system is the fast frequency modulation (FFSK) technique, which is very attractive for band limited channels with nonlinear amplification. Two successive encoders are used, the first is a $R=3/6$ rate linear block code used for detecting errors in the demodulated sequences and the second is a $R=1/2$ rate trellis code designed in order to improve the error performance by increasing the Euclidean distances between the modulated channel sequences. At the receiver, the noisy channel output is demodulated by the soft decision Viterbi algorithm. The considered system is simulated by a computer program to analyze its throughput and reliability for the basic "stop and wait", "go-back N" and selective repeat type ARQ schemes. A modified Viterbi algorithm for repeat-request, to improve the reliability more, is also introduced which leads to a slight degradation in the throughput performance. For the stop and wait case, theoretical upper bounds on the throughput for the two algorithms are compared. The error performance of

SUMMARY

In this thesis, a new hybrid automatic repeat request (ARQ) system with high throughput and high reliability is designed based on the combined coding-modulation technique. The modulation used in the proposed system is the fast frequency modulation (FFSK) technique, which is very attractive for band limited channels with nonlinear amplification. Two successive encoders, are used, the first is a $R=3/6$ rate linear block code used for detecting errors in the demodulated sequences and the second is a $R=1/2$ rate trellis code designed in order to improve the error performance by increasing the Euclidean distances between the modulated channel sequences. At the receiver, the noisy channel output is demodulated by the soft decision Viterbi algorithm. The considered system is simulated by a computer program to analyse its throughput and reliability for the basic "stop and wait", "go-back N" and selective repeat" types ARQ schemes. A modified Viterbi algorithm for repeat-request, to improve the reliability more, is also introduced which leads to a slight degradation in the throughput performance. For the stop and wait case, theoretical upper bounds on the throughput for the two algorithms, are compared. The error performance of

the Viterbi algorithm is improved by using memory in the receiver which stores the Euclidian distances computed in each transmission of the same codeword sequence and sums the accumulated Euclidian distances.

alanlarından biridir. Tasarlanacak bir iletişim dizgesinin niteliği, gönderilen bilginin ne kadar yüksek düzeyde doğru olarak alınabilmesiyle belirlenir. Dolayısıyla dizgeyi daha tasarlarken dizgeyi oluşturan her bir bloğun, dizgenin niteliği açısından neler getirip, neler göstereceği dikkate alınmalıdır. Dizgeyi oluşturan bloklarda yapılacak değişikliklerle veya bloklardaki çeşitli parametrelerin yeniden düzenlenmesiyle ne gibi iyileşmelere gidilebileceği ortaya konulmalıdır. Böylelikle oluşturulan iletişim dizgesinin nitelikliliği, olabilecek en üst düzeye getirilebilir.

Bir iletişim dizgesinin nitelik ölçütleri, iş çıkarma yeteneği (throughput) ve güvenilirlik (reliability) dir. Bu iki ölçüt en elverişli noktada belirlenmek gerekir. Ne iş çıkarma yeteneğini artırmak amacıyla güvenilirliği gözetmek, ne de güvenilirliği artırmak çabasıyla iş çıkarma yeteneğini kısmak abıdır.

En basit iletişim dizgesinin blok şeması Şekil 1.1 'de görülmektedir. Burada, iletilen bilgileri kodlayacak kodlayıcı, kodlanmış bilgilerin iletiildiği bir kanal ve kanal çıkışına erişen bilgileri göreceği bir kod gözetici

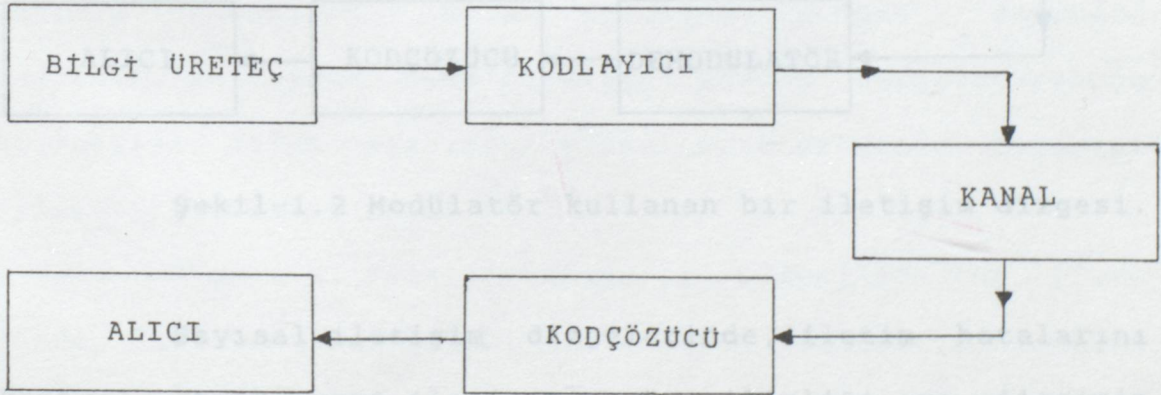
1.GİRİŞ

İletişim dizgelerinde, bilginin iletiminde oluşan hatalar ve bu hataların kontrol edilmesi temel uğraşı alanlarından biridir. Tasarlanmış bir iletişim dizgesinin niteliği; gönderilen bilginin ne kadar yüksek düzeyde doğru olarak alınabilmesiyle belirlenir. Dolayısıyla dizgeyi daha tasarlarken dizgeyi oluşturan her bir bloğun, dizgenin niteliği açısından neler getirip, neler götüreceği dikkate alınmalıdır. Dizgeyi oluşturan bloklarda yapılacak değişikliklerle veya bloklardaki çeşitli parametrelerin yeniden düzenlenmesiyle ne gibi iyileşmelere gidilebileceği ortaya konulmalıdır. Böylelikle oluşturulan iletişim dizgesinin nitelikliliği, olabilecek en üst düzeye getirilebilir.

Bir iletişim dizgesinin nitelik ölçütleri; iş çıkarma yeteneği (throughput) ve güvenilirlik (reliability) dir. Bu iki ölçütü en elverişli noktada belirlemek gerekir. Ne iş çıkarma yeteneğini artırmak amacıyla güvenilirliği göz ardı etmek, ne de güvenilirliği artırmak çabasıyla iş çıkarma yeteneğini kısmak akılcıdır.

En basit iletişim dizgesinin blok şeması Şekil 1.1 'de görülmektedir. Burada; üretilen bilgileri kodlayacak kodlayıcı, kodlanmış bilgilerin iletildiği bir kanal ve kanal çıkışına erişen bilgileri çözecek olan bir kod çözücü

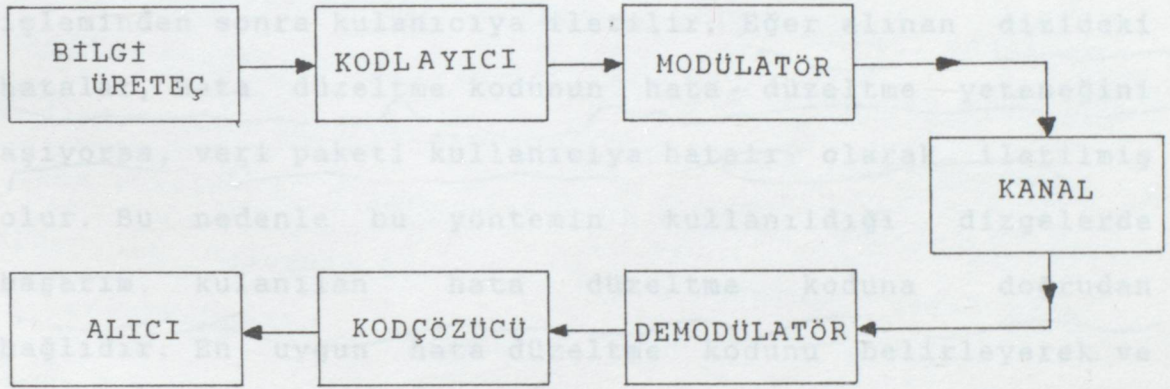
görülmektedir. İletilecek bilgiler, iletişim kanalından geçirilebilecek şekilde, uygun bir kodlama yöntemiyle kodlanarak iletilir. İletişim kanalı, uygulamada hiçbir zaman ideal olmayacağı için çeşitli gürültüler, iletilecek bilgimizde bozulmalara yol açabilir. Gürültü ise; kanalın yapısına, iletim ortamına doğrudan bağlıdır. Kanal çıkışında alınan işaret ise, kodlayıcıya uygun bir kodçözücü yardımıyla çözülerek iletilen bilginin elde edilmesine çalışılır. Bu bilgi artık alıcıya aktarılabilir.



Şekil-1.1 ilkel iletişim dizgesinin blok şeması.

İletim işlemini daha verimli kılmak için; yani hata olasılığını daha aza indirmek ve aynı anda birden çok ve farklı bilgiyi iletebilmek için modülasyon işleminden yararlanılır. Günümüzde sayısal iletişimin dolayısıyla sayısal modülasyonun üstünlükleri bilindiğine

göre; iletişim dizgesi, modülasyon ...v.b. terimlerden söz ederken sayısal düzenekleri öncelikle düşünmek yararlı olacaktır. Modülasyon işleminden yararlanabilmek amacıyla, kanal girişinde bir modülatör ve kanal çıkışında da demodülatör kullanmak gerekecektir. Böyle bir düzenek Şekil-1.2 de görülmektedir.



Şekil-1.2 Modülatör kullanan bir iletişim dizgesi.

Sayısal iletişim dizgelerinde, iletim hatalarını kontrol etmek veri iletiminin güvenilirliği ve iletimin başarımı açısından gereklidir. İletim ortamındaki kanal gürültüsünden kaynaklanan hatalar kontrol edilebilmeli, en az hatalı iletim gerçekleştirilebilmelidir. Bu soruna çözüm hataları sezen ve düzelten kodlama yöntemleriyle getirilebilir. Başarımın iyileştirilmesi amacıyla kullanılan iki ana yöntem vardır:

1.İleri Yönde Hata Düzeltme (FEC, Forward Error

Correction),

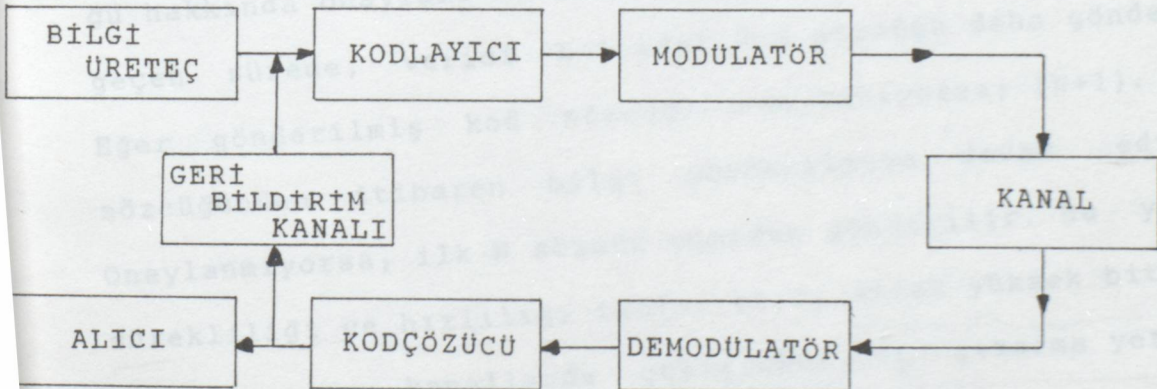
2. Otomatik Yineleme İsteme (ARQ, Automatic Repeat Request).

İleri Yönde Hata Düzeltmeli dizgelerde; alınan kod sözcüğünün hatalı bit içerdiği sezilirse, alıcıda kullanılan hata düzeltme kodu yardımıyla hatalı bitlerin düzeltilmesi yolunda işlemler yapılır. Düzeltilmiş bilgiler, kod çözme işleminden sonra kullanıcıya iletilir. Eğer alınan dizideki hatalar, hata düzeltme kodunun hata düzeltme yeteneğini aşıyorsa, veri paketi kullanıcıya hatalı olarak iletilmiş olur. Bu nedenle bu yöntemin kullanıldığı dizgelerde başarımla, kullanılan hata düzeltme koduna doğrudan bağlıdır. En uygun hata düzeltme kodunu belirleyerek ve kullanılan kodun başarısını yükselterek dizgenin başarımla iyileştirilebilir.

Otomatik Yineleme İstemeli Dizgelerde ise, İleri Yönde Hata Düzeltme (İYHD) yönteminde olduğu gibi hata düzeltme işlemi yapılmaz. Alınan kod sözcüğünün, hata sezici kod yardımıyla, hatalı bit içerip-içermediği kontrol edilir. Eğer alınan kod sözcüğünde hata sezilmişse, aynı kod sözcüğünün yeniden girilmesi istenir. Bu istem ayrı bir kanal üzerinden gönderilen işaret ile vericiye belirtilir. Alınan kod sözcüğünde hata sezilmedi ise, kod sözcüğü kullanıcıya iletilir. Bu yöntemde alınan kod sözcüğü, hata sezici kodun sezemediği bir hata içeriyorsa, o kod

sözcüğü hatalı olarak iletilecektir.

Bu iki yöntemin birleştirilerek kullanılması da mümkündür. Bu yönteme Karma OYİ (Hybrid ARQ) adı verilir. Bu yöntemle; İYHD alt dizgesi, kod sözcüğünde sezdiği hataları düzelterek yeniden gönderme işlemlerinin sayısını azaltır. Dolayısıyla İş Çıkarma Yeteneği artar. Yeniden gönderme ise, hatalı çözülmüş belirli kod değerlerinin (hataları düzeltilememiş), kullanıcıya iletimini engeller ve yeniden gönderilmesini sağlar. Sonuçta Karma OYİ dizgesi İYHD dizgesinden daha yüksek güvenilirlik ve OYİ dizgesinden daha yüksek İş Çıkarma Yeteneği sağlar. Şekil-1.3' de Modülasyonla tümleşmiş bir karma OYİ dizgesinin genel blok şeması görülmektedir.



Şekil-1.3 Birleşik modülasyonlu Karma OYİ dizge
lok şeması.

kod sözcük OYİ Dizgeleri, temelde üç farklı tipte oluşturulurlar. Bunlar:

- 1- Dur ve Bekle (stop and wait, SW),
 - 2- N-Adım Geri Dön (go back-N, GBN),
 - 3- Seçerek Yineleme (selective repeat, SR)
- 'dir. Bu yöntemler ikinci bölümde ayrıntılarıyla ele alınacaktır.

¹⁴
Dur ve Bekle (DB) Yöntemi kullanılan OYİ dizgelerde, gönderilen bir kod sözcüğü için hatasız alındığı yolunda onaylama bilgisi gelinceye kadar, verici yeni bir kod sözcüğü göndermeden bekler. Eğer onaylanmadıysa, aynı kod sözcüğü yeniden gönderilir.

N-Adım Geri Dönmeli (N-AGD) OYİ dizgelerde ise, ¹⁶
bir kod sözcüğü gönderildikten sonra, alıcının o kod sözcüğü hakkında onaylama-onaylamama bilgisinin alınmasına kadar geçen sürede, verici N-1 adet kod sözcüğü daha gönderir. Eğer gönderilmiş kod sözcüğü onaylanıyorsa; (N+1). kod sözcüğünden itibaren bilgi gönderilmeye devam edilir. Onaylanmıyorsa; ilk N sözcük yeniden gönderilir. Bu yöntem sürekliliği ve hızliliği sağlamıştır; ancak yüksek bit hata oranına sahip kanallarda çalışırken, iş çıkarma yeteneği düşer.

¹⁷
Seçerek Yinelemeli (SY) OYİ dizgeleri, DB OYİ ve N-AGD OYİ dizgelerin sakıncalarını gidermek amacıyla geliştirilmiştir. Bu dizgelerde, yalnızca hatalı alınan

kod sözcükleri yeniden gönderilir. Doğru olarak ulaşan kod sözcükleri alıcı tarafta bulunan ara bellekte bekletilir. Bu nedenle, bu dizgenin iş çıkarma yeteneğinin ve güvenilirliğinin yüksek olması, söz konusu ara belleğin genişliğine bağlıdır. Kuramsal olarak sonsuz uzunluklu bir ara bellek kullanıldığı varsayılır.

kod sözcüğünün tıbbi yerine, yalnızca eşlik bitlerinin

iletimi 1.1 KONUyla İLGİLİ YAPILMIŞ ÇALIŞMALAR

Otomatik yineleme istemeli dizgeler ile ilgili bugüne değin çok sayıda çalışma yapılmıştır. Bu çalışmalar, ilerleyen zaman içinde gelişmiş, kendine yeni açılımlar sağlamıştır. Bu konuda elle tutulur ilk çalışmalar 1960 sonları ve 1970 başlarında gerçekleşmiş ve hız kazanmıştır. Çalışmaların kalkış noktasını, hata ve hata kontrolüne yönelik çalışmalar oluşturmıştır.

1970' li yıllara kadar Dur ve Bekle Yöntemi, yüksek veri iletim hızlarına çıkılmadığı ve sürekli veri iletimi yapılmadığı için yaygın olarak kullanılmıştır. Bu yöntemle kurulmuş OYİ dizgelere, IBM'in ikili eşzamanlı iletişim (BISYNC) protokolü örnek verilebilir. Daha sonraki yıllarda, özellikle bilgisayar dünyasındaki hızlı gelişmeler, veri iletiminde hızlılığı ve sürekliliği gerekli kıldığından, N-Adım Geri Dönme Yöntemi kullanılmaya başlamıştır. N-AGD OYİ dizgelerin yüksek bit hata oranlı kanallardaki yetersizliği, SY OYİ dizgelerin kullanımını

kaçınılmaz kılmiştir. Bu arada, bu yöntemlerin bir arada veya ayırık, çeşitli şekillerde geliştirilmeleriyle daha yüksek başarımlar elde etme yolunda çalışmalar gerçekleştirilmiştir. Bunlardan özellikle belirtmeden geçilemeyecek önemde olanları anmak yararlı olacaktır. Bu çalışmaların esası genelde; yeniden iletim istemi alındığında kod sözcüğünün tümü yerine, yalnızca eşlik bitlerinin iletimi düşüncesine dayanmaktadır. Bu konuda incelenebilecek kayda değer ilk çalışmalar Sastry [1], Sindhu [2], Metzner [3] ve Lin ve Yu [4] tarafından gerçekleştirilmiştir. Sastry, bir kod sözcüğünün ard arda birkaç kez gönderilmesi temeline dayalı bir dizge ile başarımın arttığını ortaya koymuştur. Sindhu, Bellekli OYİ Dizge üzerinde yoğun çalışmalar sürdürerek bu konuda oldukça yetkin bir kaynak oluşturmuştur. Metzner'in çalışmaları bunun üzerine oturtulmuştur. Lin ve Yu'nun geliştirdiği SETRAN ARQ, Sastry'nin önerdiği yönteme göre daha olumlu başarımlar sonuçları vermiştir[4]. Yine Yu ve Lin [5]'in ortak bir çalışmasında seçerek yinelemeli OYİ dizgeleri, çeşitli ara bellek uzunlukları için incelenmiştir. Miller ve Lin [6]' ise, (SY+N-AGD) ve (SY+ST) olarak adlandırdıkları karmaşık OYİ uygulamaları üzerine çalışılmışlar ve her iki yöntem için de elde edilen sonuçların SETRAN ARQ dizgesinden daha iyi başarımlar sağladığı ortaya konulmuştur. Weldon [7] ise, yüksek veri hızına sahip

patlamalı kanallarda, alıcı ara belleğin kontrolüyle başarımın yüksek tutulması yönünde çalışmalar yapmıştır. Moeneclaey ve Bruneel [8] tasarladıkları çok basit yöntemle yüksek bit hata oranları durumlarında, N-AGD OYİ dizgesi için nerdeyse SY OYİ dizgesine eşit başarımlar elde etmişlerdir. Yöntem, aynı bir kod sözcüğünün m kez ard arda gönderilmesi temeline dayanmaktadır. Bu konudaki en yeni ve en gelişmiş çalışmaların sahibi olan Benelli ise [9], [10], [11], [12] çalışmalarında yumuşak (soft) kararlı demodülasyona dayanarak başarımda belli bir iyileşmenin sağlanabildiğini göstermiştir. Benelli, tümleşik modülasyonlu ve bellekli bir OYİ dizge üzerine yaptığı genel çalışma [11] ile, OYİ dizgelerde bellek, kodlama ve modülasyon işlerini bütünleştirmiş ve başarımın arttığını ortaya koymuştur. Bu çalışma özellikle yüksek bit hata oranlarına sahip kanallarda başarımın oldukça yüksek düzeylerde arttığını ortaya koyması ve OYİ dizgelerde kanal kodlama ve modülasyonun tümleştirilerek kullanılması yönleriyle önemlidir. Yine aynı yöndeki diğer çalışmasında da Benelli [12], yeni bir yöntem ortaya koyarak bu konuda o günlere değin yapılmış bütün çalışmaları bir araya getirmiştir. Benelli [12] daha sonra M'li modülasyon ile gerçekleştirilmiş OYİ dizgeleri incelemiştir.



1.2 TEZİN AMACI VE İZLENEN YOL

Bu çalışmada, bugüne değin yapılmış çalışmaların ışığında yeni bir tümleşik kodlamalı-modülasyon dizgesi tasarlanmıştır. Bu dizgenin çeşitli OYİ yöntemlerine göre ve dizgenin çeşitli parametrelerine göre başarımı incelenmiştir. Dizgede hata sezici olarak bir blok kodlayıcı ve hata düzeltici olarak katlamalı (Convolutional) kodlayıcı kullanılmıştır. Katlamalı kodlayıcı, kafes kodlamalı modülasyon tekniğine uygun olarak, kodlanmış işaret dizileri arasındaki Öklid uzaklıkları en büyük, olacak biçimde tasarlanmıştır. Modülasyon amacıyla, hızlı frekans kaydırmalı anahtarlama (HFKA), (FFSK, Fast Frequency Shift Keying) tekniği seçilmiştir. HFKA, sürekli faz, sabit zarf nitelikleriyle doğrusal olmayan kuvvetlendiriciler içeren band sınırlı iletişim ortamlarında çekici bir modülasyon tekniğidir. Kanal çıkışında kod çözme işlemi Viterbi kod çözme Algoritması ile gerçekleştirilmiştir. Aynı zamanda Yamamoto ve Itoh [13]'ün önerdikleri yineleme istemeli Viterbi kod çözme algoritması da kullanılarak başarımlar üzerindeki etkileri incelenmiştir.

Ayrıca, alıcıda ara bellek kullanılması durumu ele alınmıştır. Bu durumda, bir kod sözcüğünün alıcıya ulaşınca kadarki her bir iletiminde belirlenen Öklid uzaklığı bilgileri saklanarak ve ard arda toplanarak birikmiş toplamsal Öklid uzaklığı bilgisi elde edilmektedir.

Bu bilgiye bakarak, alınan dizinin kullanıcıya iletilmesine karar verilmektedir. Ele alınan bu yöntemin daha yüksek başarımlı sağladığı görülmüştür.

Tez çalışmasının ikinci bölümü genel olarak OYİ dizgelerin tanıtılmasını ve her bir yöntemin kuramsal başarımlarının incelenmesini içermektedir. Üçüncü bölümde, birleşik kodlamalı modülasyonlu OYİ dizgelerinin yapısından söz edilmektedir. Dördüncü bölüm, HFKA tekniği kullanan bir kodlamalı modülasyonlu OYİ dizgenin tasarımı anlatılarak tezin yöneldiği hedefin ortaya konulduğu bölümdür. Beşinci bölüm, bir önceki bölümde tasarlanan dizgenin başarımlarının incelenmesini içermektedir. Altıncı ve son bölümde ise, tez çalışmasında elde edilen sonuçlar yer almaktadır. Benzetim çalışmaları için, Turbo Pascal dilinde geliştirilmiş program metni ise tezin sonunda ek olarak verilmiştir.

şekmek gerekecektir. Doğal olarak bu gözlem doğrultusunda, ekonomik ve uygulamaya yönelik olabilmektedir[14].

OYİ yönteminde ise, kodlanmış bilgi kanaldan geçtikten sonra eşlik bitlerine bakılarak eşlik bitlerinin veri bitlerine uyumlu olup olmadığı incelenir. Eğer denetim sonucu yanlış verirse, kod sözcüğü alıcıya iletilir, veyahut da, bir geri besleme kanalı üzerinden verici uyarılır. Bu uyarı doğrultusunda kod sözcüğü yanlış bulununcaya kadar yeniden gönderilir. Bu yöntemde dizgenin iç içe kodlama yeteneği, kanaldaki bit hatalarının artmasıyla

2. OTOMATİK YİNELEME İSTEMELİ (OYİ) DİZGELER

Veri iletiminde, iletim hatalarının kontrolünde kullanılan iki yöntemden ileri Yönde Hata Düzeltme (İYHD) (Forward Error Correction, FEC) yöntemi, yüksek iş çıkarma yeteneği (throughput) sağlamasına karşın, güvenilirlikleri (reliability) düşüktür. Bu tür dizgelerde, alıcıda hata sezme ve sezilen bir hatayı düzeltme işlemleri gerçekleştirilir. Alınan herhangi bir kod sözcüğündeki hata düzelttikten sonra alıcıya gönderilir. Hata düzeltici kodun sezemediği dolayısıyla düzeltilemeyen hatalar varsa, bu kod sözcükleri alıcıya hatalı olarak iletilebilecektir. Buna göre bu tür dizgelerin başarımını artırmak için; bilgiyi uzun kod sözcükleri halinde oluşturmak ve hata sezip-düzeltme yeteneği yüksek olan kodlar seçmek gerekmektedir. Doğal olarak bu çözüm önerileri, ekonomik ve uygulamaya yönelik olmayabilir[14].

OYİ yöntemde ise, kodlanmış bilgi kanaldan geçtikten sonra eşlik bitlerine bakılarak eşlik bitlerinin veri bitlerine uyumlu olup olmadığı incelenir. Eğer denetim olumlu sonuç verirse; kod sözcüğü alıcıya iletilir, vermezse; bir geri besleme kanalı üzerinden verici uyarılır. Bu uyarı doğrultusunda kod sözcüğü uyumlu bulununcaya kadar yeniden gönderilir. Bu yöntemde dizgenin iş çıkarma yeteneği, kanalın bit hata olasılığının artmasıyla

düşecektir. Bu yöntemde dizgenin güvenilirliği çok yüksektir. Bu nedenle güvenilirlik açısından OYİ yöntemi tercih edilir. Buna karşılık; geri beslenmenin mümkün olmadığı durumlarda, uydu haberleşmesi gibi verici-alıcı uzaklığının fazla olmadığı durumlarda ve bir veri paketi hakkında bilgi almak için gereken bekleme süresinin başarıyı düşüreceği ortamlarda İYHD yöntemi tercih edilir. Ancak OYİ yöntemi üzerinde yapılan son çalışmalar yüksek bit hata olasılıklı durumlarda da OYİ dizgelerin başarılarının yüksek tutulmasını sağlamaktadır. Dolayısıyla hata olasılıkları büyük ölçüde düşürülmüştür.

2.1 TEMEL OYİ DİZGELER

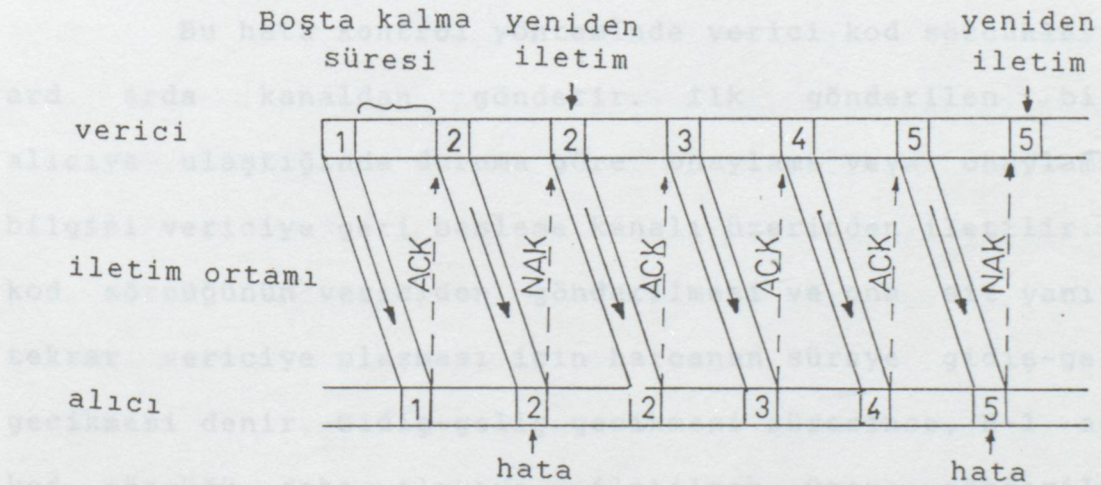
Temel OYİ dizgeler, yeniden gönderme yöntemlerine göre üç gruba ayrılırlar. Bunlar;

1. Dur ve Bekle OYİ dizgesi,
- 2.N-Adım Geri Dönmeli OYİ dizgesi,
- 3.Seçerek Yinelemeli OYİ dizgesi.

2.1.1 DUR ve BEKLE OYİ DİZGESİ

Dur ve Bekle (Stop and Wait, SW) dizgesi zamanda geliştirilen ilk hata kontrol dizgesidir. Şekil-2.1 'de çalışma düzeni görülen bu dizgede alıcı, vericinin gönderdiği bir kod sözcüğünü belli bir süre sonra alır. Alınan her bir kod sözcüğünün hatalı olup olmadığına

Bakılır. Eşlik bitleri, veri bitlerini sağlamıyorsa, hatalı olduğuna karar verilir. Eğer hatalı olduğuna karar verildiyse; bir geri besleme kanalı üzerinden onaylamama bilgisi, hatasız bulduysa onaylama bilgisi vericiye iletilir (onaylama ACK (acknowledgement), onaylamama NACK (negative acknowledgement)). Burada kullanılan geri besleme kanalı, yapılan çalışmada ideal varsayılmıştır. Verici gönderdiği veri paketine karşılık alıcıdan yanıt gelinceye değin bekler. Başka bir veri paketi gönderemez. Bu arada geçen süreye boşta kalma süresi (idle time) denir. Verici, geri besleme kanalından aldığı bilgi, onaylama bilgisi ise; sıradaki veri paketini iletir, onaylamama bilgisi ise; onaylanmayan veri paketini yeniden gönderir. Bu işlem bir veri paketinin alıcı tarafından onaylandığı bilgisi alınıncaya değin tekrarlanır.



Şekil-2.1 DB OYİ çalışma yapısı.

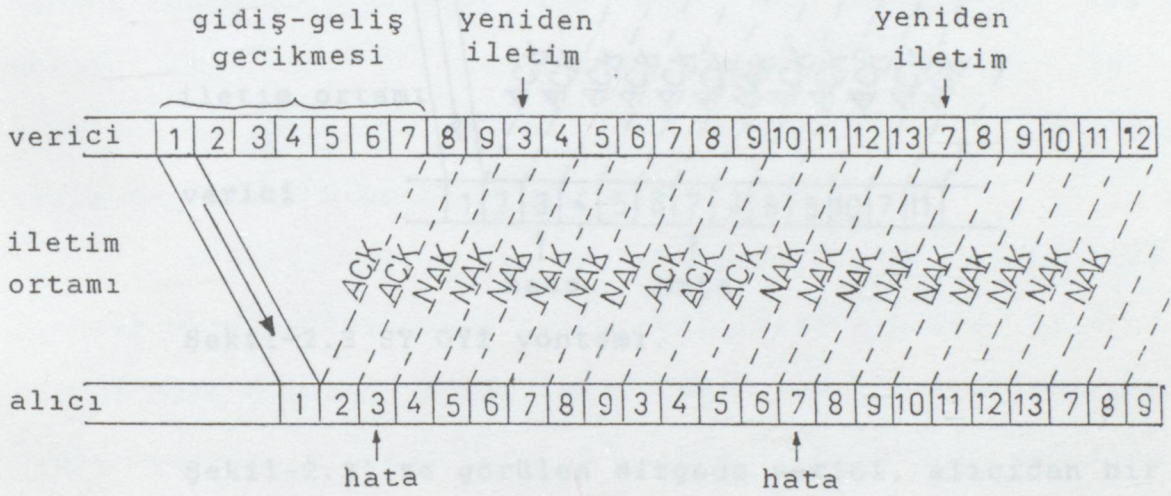
Basit olmasına karşın, harcadığı boşta kalma süresi nedeniyle iş çıkarma yeteneği oldukça düşüktür. Bunu gidermek için, kod sözcüğünü uzatma yoluna gidilebilir. Böylelikle harcanan boşta kalma süresi azaltılır. Ancak bu kez de bir bloğun hata içerme olasılığının blok uzunluğu ile artacağını, dolayısıyla yeniden gönderme işlemi sayısının artacağını dikkate almak gerekir.

2.1.2 N-ADIM GERİ DÖNMELİ OYİ DİZGESİ

Gidiş-geliş gecikmesi (round-trip delay) uzun olan uydu kanallarının kullanılması ve daha yüksek veri hızlarına çıkılması, sürekli bir biçimde veri iletilmesini gerekli kılmıştır. Bu gereksinmeden hareketle N-Adım Geri Dönmeli OYİ dizgesi ortaya atılmıştır. Özellikle paket bağlaşmada iki yönlü N-adım geri dönmeli (N-AGD), (Go-back N, GBN) OYİ, yeni bir yöntem olarak kabul edilmiştir.

Bu hata kontrol yönteminde verici kod sözcüklerini ard arda kanaldan gönderir. İlk gönderilen bilgi alıcıya ulaştığında duruma göre onaylama veya onaylamama bilgisi vericiye geri besleme kanalı üzerinden iletilir. Bu kod sözcüğünün vericiden gönderilmesi ve ona ait yanıtın tekrar vericiye ulaşması için harcanan süreye gidiş-geliş gecikmesi denir. Gidiş-geliş gecikmesi süresince, N-1 adet kod sözcüğü daha alıcıya iletilmek üzere gönderilir. Herhangi bir kod sözcüğü hakkında onaylama bilgisi a-

lındıysa; sıradaki kod sözcüğü vericiden gönderilir. Onaylanmadıysa; onaylanmayan kod sözcüğü ve onu izleyen N-1 adet kod sözcüğü ard arda yeniden iletilir. Yani kendisini izleyen N-1 adet kod sözcüğünün doğruluğu incelenmez bile. Söz konusu kod sözcüğü hakkında onaylama bilgisi alınıncaya kadar bu işleme devam edilir(Şekil-2.2).



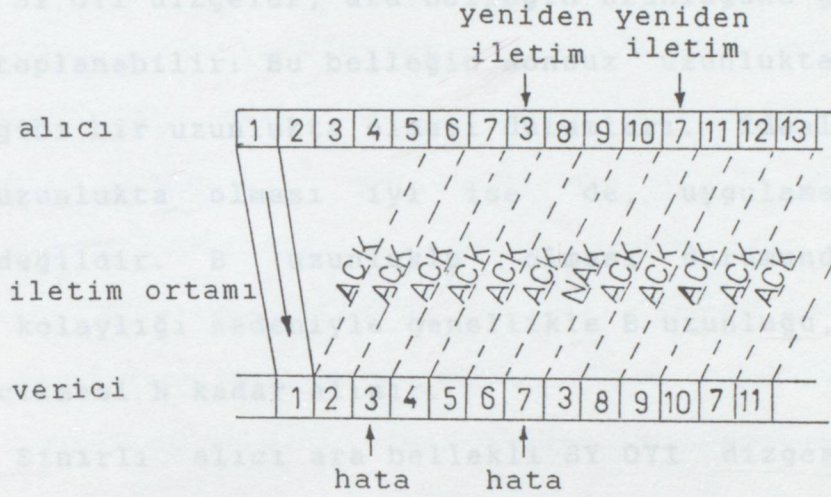
Şekil-2.2 N-Adım Geri Dönmeli OYİ dizgesi (N=7)

Bu yöntemin temel olumsuzluğu, bir kod sözcüğü hatalı olarak alındığında o kod sözcüğünden sonra gelen N-1 adet kod sözcüğünün, hatalı olup olmadığına bakılmaksızın hatalı kabul etmesidir. Ancak, kanal bit hata oranının düşük olduğu durumlarda Dur ve Bekle yönteminden çok daha yüksek başarımlar sağladığı bir gerçektir.

2.1.3 SEÇEREK YİNELEMELİ (SY) OYİ DİZGESİ

N-Adım Geri Dönmeli OYİ dizgesinin olumsuzlukları-

nı gidermek için seçerek yinelemeli (SY), (Selective-Repeat SR) yöntem geliştirilmiştir. Bu yöntemde de, N-AGD yönteminde olduğu gibi sürekli bir veri iletimi vardır.



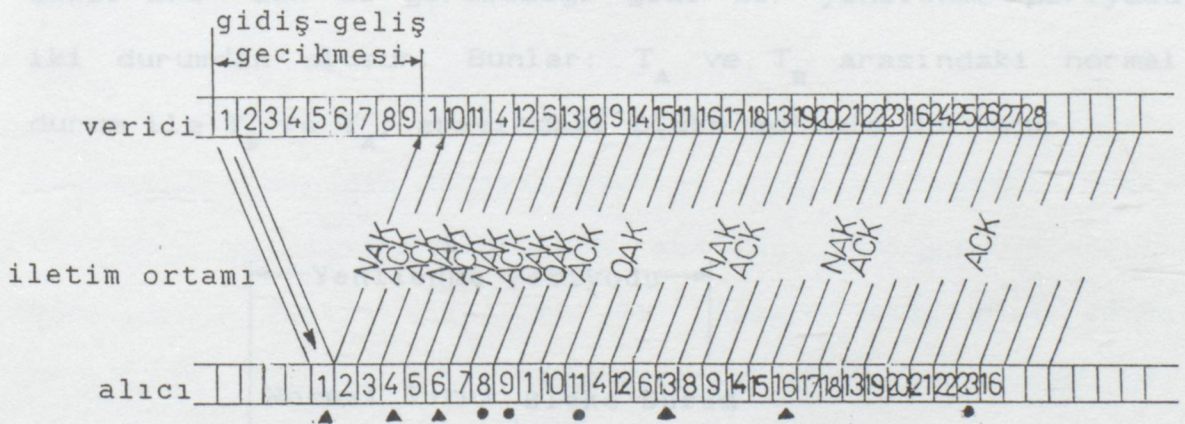
Şekil-2.3 SY OYİ yöntemi.

Şekil-2.3' te görülen dizgede verici, alıcıdan bir onaylamama bilgisi gelinceye kadar veri paketlerini gönderir. Herhangi bir veri bloğu için onaylamama bilgisi vericiye ulaştığı anda sıradaki veri bloğu yerine, onaylanmayan veri bloğunu yeniden gönderir, sonra kaldığı yerden gönderme işlemine devam eder. Bu dizgede, kod sözcüklerinin kullanıcıya doğru sırada iletilebilmesi için, hatalı kod sözcüklerini izleyen, hatasız kod sözcüklerinin alıcıda saklanmasını sağlayan bir ara bellek oluşturulmalıdır. Ara bellekte aşırı birikmeden dolayı taşmaya yol açmamak için ara belleğin mümkün olduğunca geniş olması gerekir. Ara bellekten taşma söz konusu olursa,

taşan kod sözcükleri kaybolacaktır. Hedeflenen başarının sağlanabilmesi için ara belleğin kuramsal olarak sonsuz uzunlukta olması gerekmektedir.

SY OYİ dizgeler, ara belleğin uzunluğuna göre iki grupta toplanabilir: Bu belleğin sonsuz uzunlukta olması veya B gibi bir uzunlukta olması durumları. İdeal olarak sonsuz uzunlukta olması iyi ise de, uygulamada olanaklı değildir. B uzunluklu olması durumunda ise; uygulama kolaylığı nedeniyle genellikle B uzunluğu, gidiş-geliş gecikmesi N kadar alınır.

Sınırlı alıcı ara bellekli SY OYİ dizgesinin iş çıkarma yeteneği iki ayrı durum için incelenebilir. Birinci durum; $B \leq N$ durumu, ikinci durum ise; $B = s(N-1) + 1 + b$ durumu olup, burada s ve b, $s \geq 0$ ve $0 < b \leq N-1$ koşulunu sağlayan pozitif tam sayılardır.



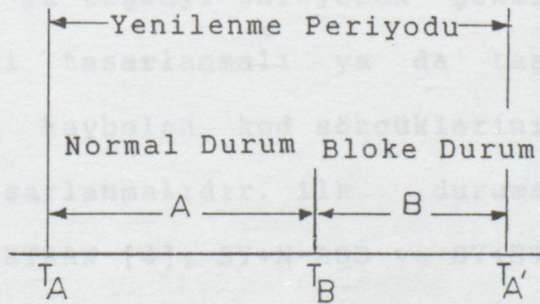
Şekil-2.4 Geliştirilmiş, $B \leq N$ durumlu SY OYİ

dizgesi ($B=7$, $N=9$). • Bellek taşmalarını, ▲ hatalı alınan blokları göstermektedir.

$B \leq N$ durumunda, alıcı ara belleğinde taşma olayına, bloke edilmiş çalışma durumu denir.

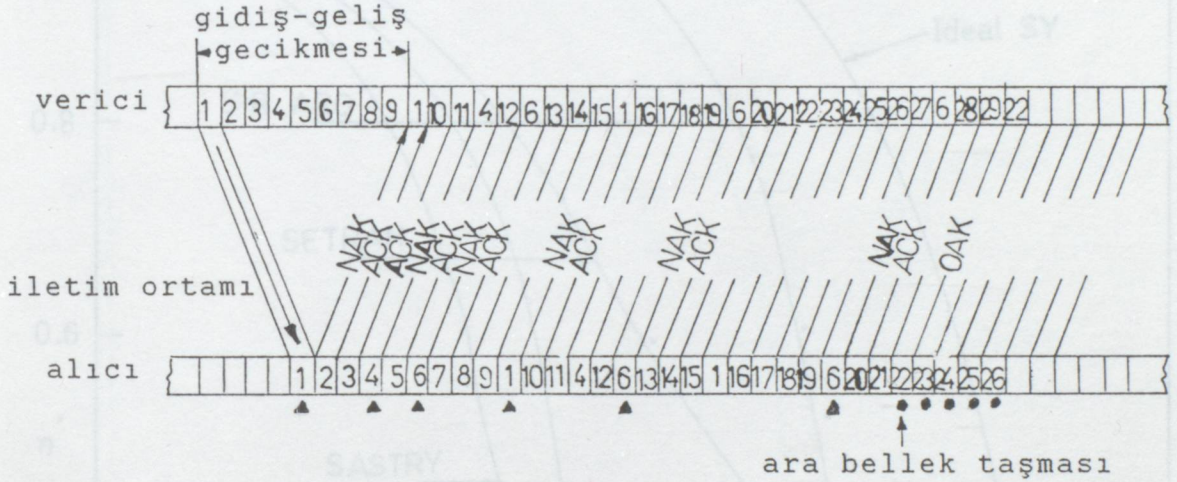
T_A , alıcının bloke edilmiş durumdan normal duruma döndüğü (veya alıcının ilk kez çalışmaya başladığı) zaman sabiti olsun. T_B , alıcının bloke edilmiş duruma girdiği zaman sabiti olsun. T_A , yenilenme noktası T_B ise geçiş noktası olarak adlandırılır. Geçiş noktasından itibaren alıcı, veri düzeltme işlemini başlatır ve bu işlem daha sonraki yenilenme noktasına kadar sürer. İki yenilenme noktası arasındaki olaylar daha önceki yenilenme noktaları arasındaki olaylardan bağımsızdır.

T_A ve $T_{A'}$, iki komşu yenilenme noktası olsun. Bu durumda T_B , T_A ile $T_{A'}$ arasında bir geçiş noktasıdır. T_A ve $T_{A'}$ aralığı yenilenme periyodu olarak adlandırılır. Şekil-2.4' den de görüleceği gibi bir yenilenme periyodu iki durumdan oluşur. Bunlar: T_A ve T_B arasındaki normal durum ile T_B ve $T_{A'}$ arasındaki bloke edilmiş durumdur.



Şekil-2.5 Yenilenme periyodu.

$B=s(N-1)+1+b$ durumunda ara bellek taşması olayı, alıcının $(s+1)$. iletiminden sonra (asıl iletim ve s kez yeniden gönderme işleminden sonra) ilk hatalı alınan veri bloğunun düzeltilmesi sonucu oluşur. Şekil-2.6' da bu durum örneklenmektedir.



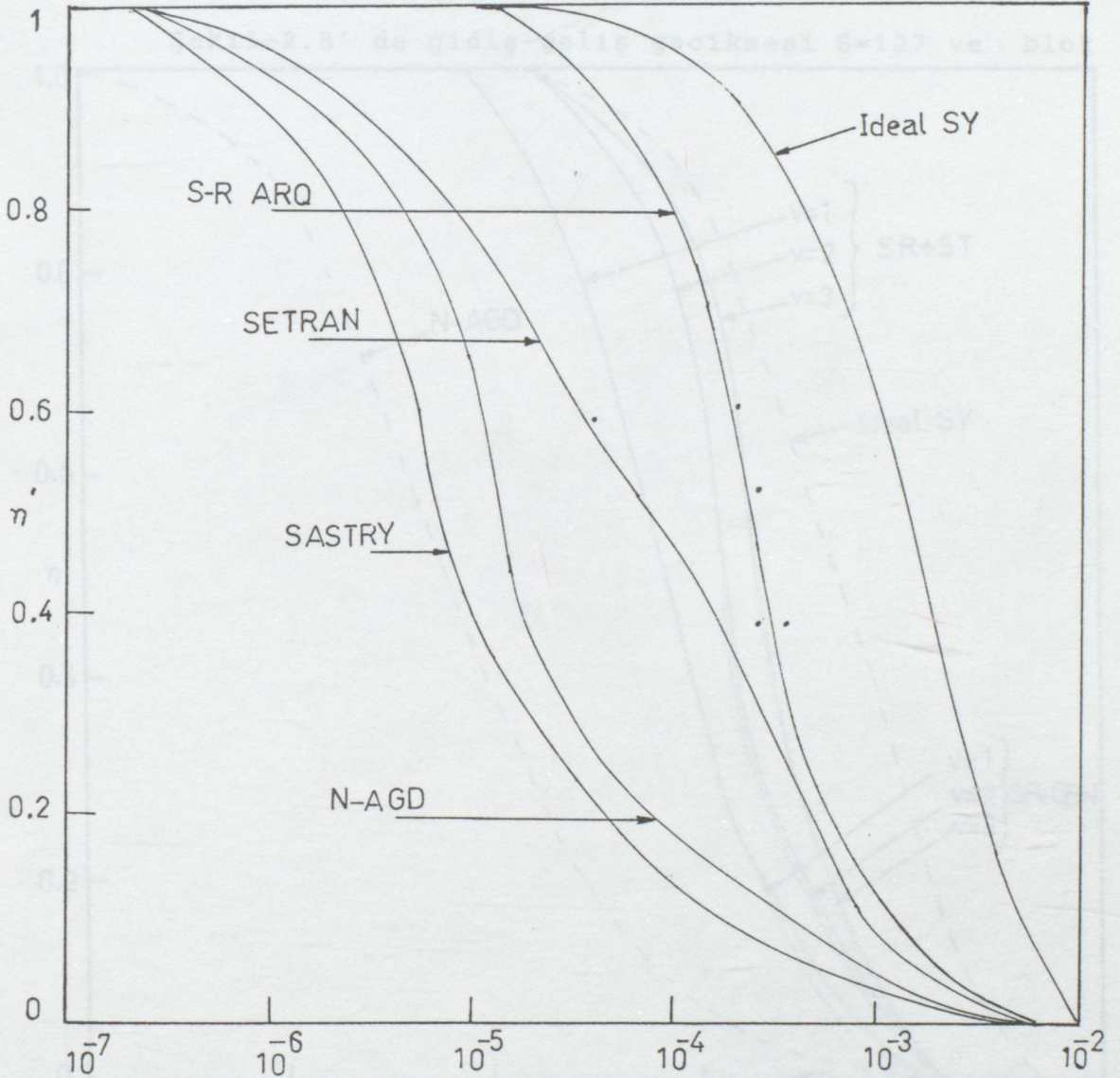
Şekil-2.6 $B=s(N-1)+1+b$ durumunda SY OYİ yapısı

($B=16$, $N=9$, $s=1$, $b=7$)

Uygulamada ara bellek uzunluğunun sonsuz olması sağlanamayacağından dolayı, taşma olacaktır. Taşmayı önlemek için, ya taşmayı önleyecek şekilde yeniden gönderme yöntemleri tasarlanmalı ya da taşmayı sezdikten sonra vericinin kaybolan kod sözcüklerini gönderebileceği bir yapı tasarlanmalıdır. İlk duruma uygun yöntemler olarak; SETRAN [4], SY+N-AGD ve SY+ST [6] diye adlandırılan karışık yöntemler uygulanabilir. Bu karışık yöntemler ile temel yöntemler arasında iş çıkarma yeteneği konusunda

karşılaştırmalar Şekil-2.7, Şekil-2.8 ve Şekil-2.9' da verilmiştir.

Şekil-2.7' de $n=524$ olan bir kod sözcüğü için, $N=128$ olması durumunda Sastry [1]' in geliştirdiği dizge, N-AGD dizge, SETRAN [5] dizge, N ara bellek uzunluklu SY

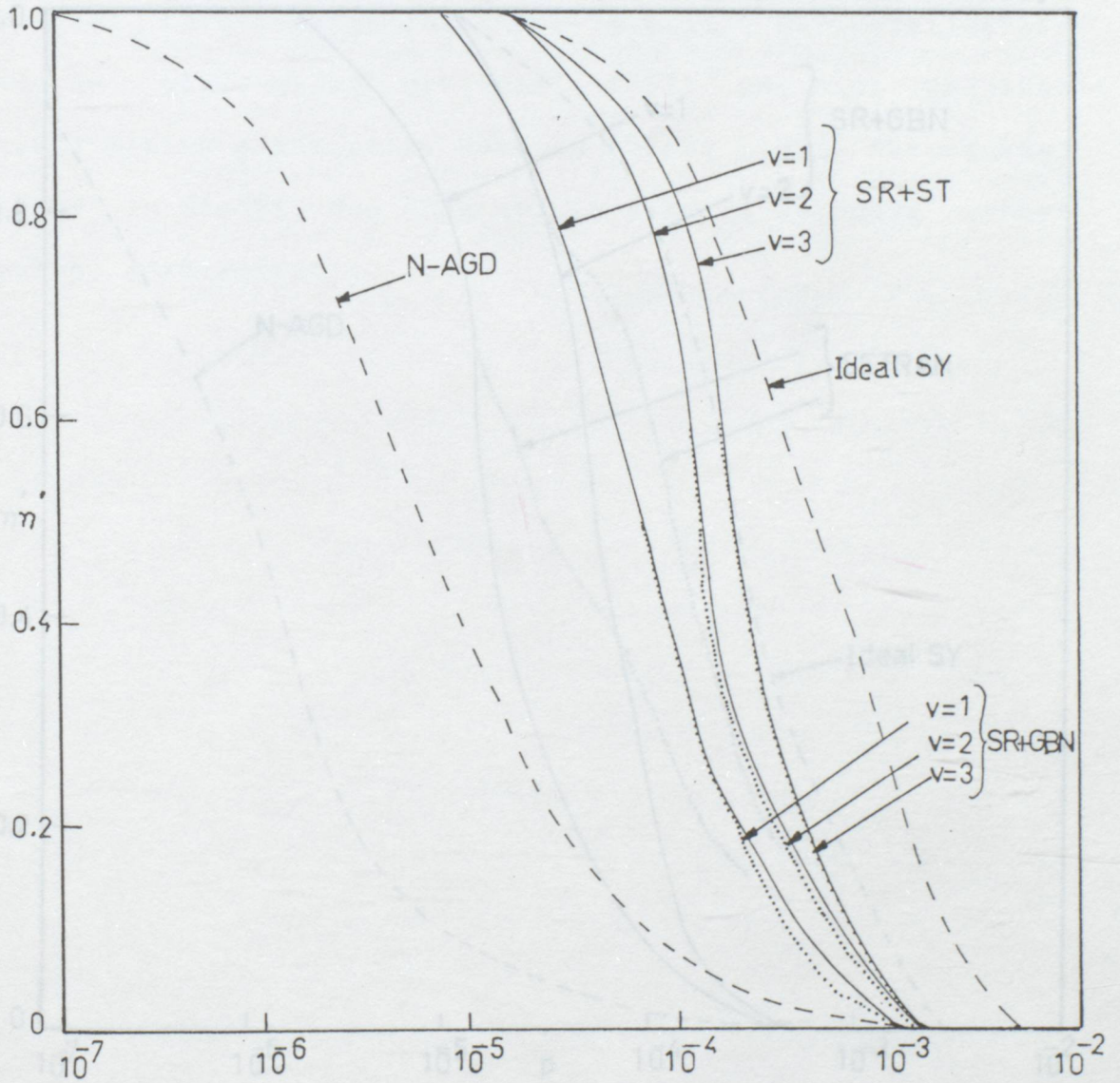


Şekil-2.7 Temel SY yapısı, Sastry [1]' in geliştirdiği yapı ve SETRAN'ın $N=128$ ve $n=524$ için iş çıkarma yeteneği karşılaştırmaları.

Şekil-2.8 Temel SY yapısı, Sastry [1]' in geliştirdiği yapı ve SETRAN'ın $N=128$ ve $n=524$ için iş çıkarma yeteneği karşılaştırmaları.

OYf dizge ve ideal SR dizge yapılarının iş çıkarma yetenekleri karşılaştırılmaktadır. Buradan, SETRAN çalışmasının N-AGD yapısına göre daha iyi, SY yapısına göre daha kötü iş çıkarma yeteneğine sahip olduğunu görmekteyiz.

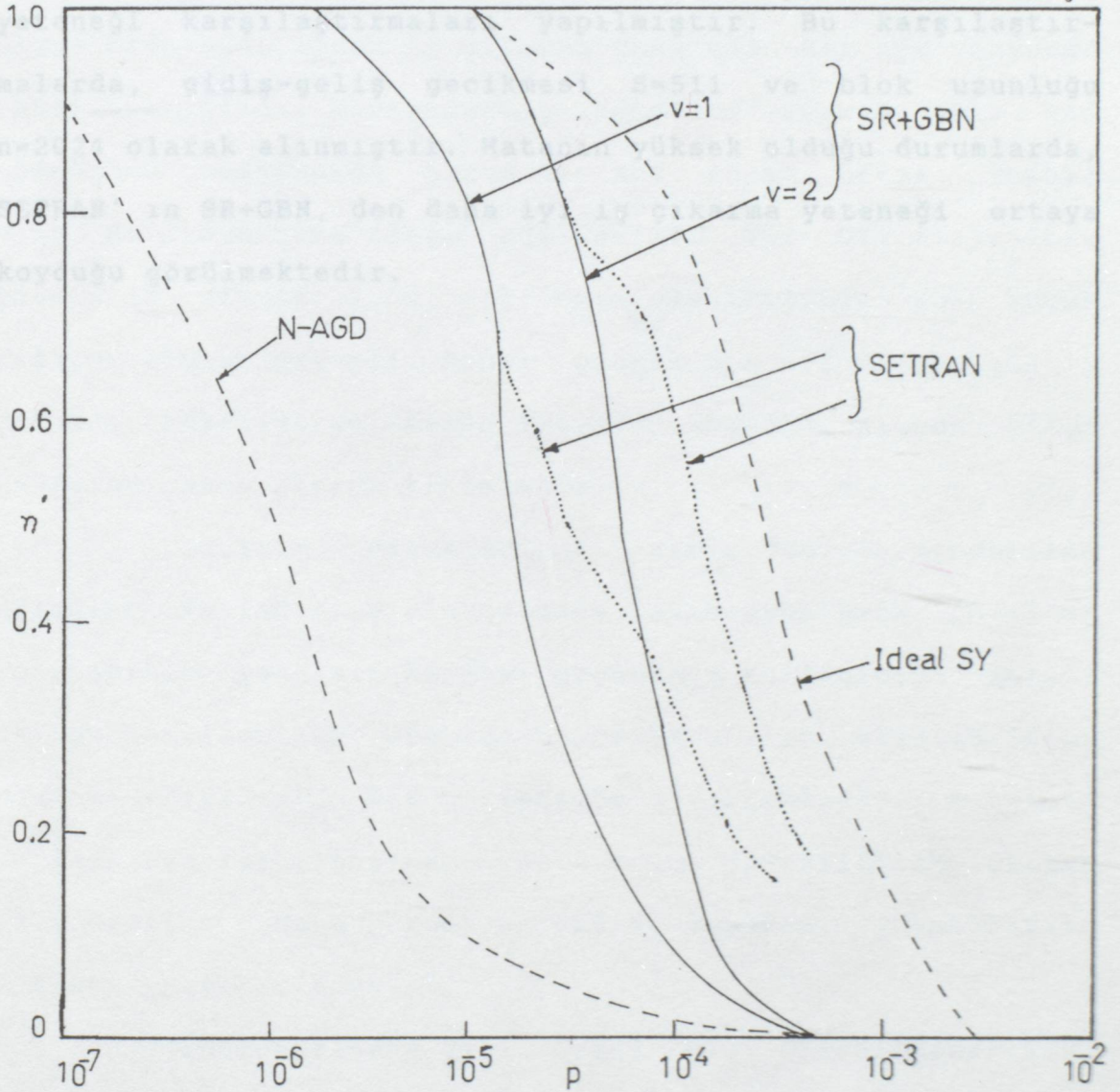
Şekil-2.8' de gidiş-geliş gecikmesi $S=127$ ve blok



Şekil-2.8 SR+ST ve SR+GBN yapıları için çeşitli

parametrelere göre iş çıkarma yeteneği karşılaştırmaları.

uznluđu $n=1024$ durumu için; N-AGD OYİ, ideal SY OYİ yapıları ile SR+ST ve SR+GBN [6] yapılarının farklı durumları için elde edilen iş çıkarma yeteneđi değeri karşılaştırılmaktadır. v parametresi, ortaya konulan yöntemler için yeniden gönderme zamanını belirleyici bir parametredir. Bu yöntemlerin ideal SY OYİ yöntemine oldukça



Şekil-2.9 SETRAN ve SR+GBN yapıları için iş

çıkarma yeteneđi karşılaştırmaları.

yakın iş çıkarma yeteneği sağladıkları, SR+ST yönteminin SR+GBN yöntemine göre daha iyi sonuç verdiği net bir şekilde görülmektedir.

Şekil-2.9' da ise; N-AGD yöntemi, ideal SY yöntemi ile SETRAN ve SR+GBN yöntemleri için iş çıkarma yeteneği karşılaştırmaları yapılmıştır. Bu karşılaştırmalarda, gidiş-geliş gecikmesi $S=511$ ve blok uzunluğu $n=2024$ olarak alınmıştır. Hatanın yüksek olduğu durumlarda, SETRAN' ın SR+GBN, den daha iyi iş çıkarma yeteneği ortaya koyduğu görülmektedir.

tanımlanabilir.

2.1.4 TEMEL OYİ DİZGELERİNİN GÜVENİLİRLİĞİ VE İŞ ÇIKARMA YETENEĞİ

Bir OYİ dizgenin başarımı, dizgenin güvenilirliği ve iş çıkarma yeteneği ile ölçülür. Güvenilirlik temel OYİ dizgelerde aynı ancak iş çıkarma yeteneği her bir yöntemde farklıdır. Alıcı, saptanamamış hataları içeren bir kod sözcüğünü onayladığı zaman bir kod çözme hatası yapmış olur. Bu olaya hata olayı adı verilir. Bir OYİ dizgesinin güvenilir olması için, $P(E)$ hata olasılığının çok küçük değerde olması gerekir. Genel olarak bir OYİ dizgesinin iş çıkarma yeteneği de, birim zamanda hatasız alınan bilgi sayısına oranı olarak tanımlanır.

İletişim dizgelerinde vericiden gönderilen bilgiler kullanıcıya ulaştığında, alıcıdaki hata düzeltme yeteneğinin yetersiz kalması durumunda kullanıcıya hatalı olarak iletilebilir. Uygulamada bu hataların ölçülebilmesi olanak dışıdır. Bir iletişim dizgesinde bilginin kullanıcıya iletilmesindeki hata oranı güvenilirlik olarak adlandırılır. Hata oranı ne kadar düşükse, güvenilirlik o kadar yüksek olacaktır.

Genelde olduğu gibi, temel bir OYİ dizgede hata sezme işlemi için (n,k) doğrusal blok kodunun kullanıldığını varsayarak aşağıdaki olasılıklar

tanımlanabilir:

P_c : n bitlik kod sözcüğünün hatalı alınma olasılığı,

P_d : alınan kod sözcüğünün sezilebilir hata içerme olasılığı,

P_o : alınan kod sözcüğünün sezilemeyen hata içerme olasılığı.

Bu olasılıkların toplam 1' e eşittir:

$$P_c + P_d + P_o = 1 \quad (2.1)$$

P_c hatası, kanalın hata istatistiklerine bağlıdır. P_d , P_o ise, hem kanal istatistiklerine hem de (n,k) hata sezici kodunun seçimine bağlıdır.

OYİ dizgelerde alıcıya ulaşan bir kod sözcüğü, ancak hatasızsa veya hatası sezilemezse onaylanır. Hatanın sezilememesi olasılığı ilk ve daha sonraki iletimlerinde eşit olduğundan (hatanın sezilememesi kodlayıcının yeteneğinin sınırlılığından kaynaklanmaktadır), alıcının hatalı bir kod sözcüğünü onaylama olasılığı:

$$\begin{aligned} P(E) &= P_o + P_d \cdot P_o + P_d^2 \cdot P_o + \dots \\ &= P_o (1 + P_d + P_d^2 + \dots) \\ &= P_o \cdot 1/(1-P_d) \end{aligned} \quad (2.2)$$

olup (2.1)' den yararlanarak,

$$P(E) = P_o / (P_c + P_o) \quad (2.3)$$

bulunur. Eğer hata sezici kod uygun olarak seçilirse; P_o , P_c ' ye göre oldukça küçük bir değerde olur ve $P(E)$ sıfıra

yaklaşır. İkili bakışimli kanal için bit hata olasılığı ε ise,

$$P_c = (1-\varepsilon)^n \quad (2.4)$$

şeklindedir. Sezilemeyen hata olasılığı;

$$P_o \leq [1-(1-\varepsilon)^k] \cdot 2^{-(n-k)} \quad (2.5)$$

koşulunu sağlayan uygun bir hata sezici kod seçilebilir. Çalışmalarda, iletim kanalı bit hata oranı ε olan raslantısal hata kanalı ve, geri besleme kanalı da ideal varsayılarak kolaylık sağlanır.

DB OYİ DİZGELERİ İÇİN İŞ ÇIKARMA YETENEĞİ

Vericinin bilgi dizisini göndermesi ve buna ilişkin onay bilgisini alması arasında geçen süre D (boşta kalma süresi), vericinin veri hızı ise, v olsun. Verici, bir gidiş-geliş gecikmesi boyunca sürekli iletimdeyse; $n+Dv$ bit iletebilecektir, k/n kod oranlı dizge için, vericinin boşta kalma süresi boyunca iletebileceği kod dizisi sayısı, Dv/n olacaktır. Ancak bu dizge sürekli iletim sağlamadığı için bu süre içinde yalnızca bir kod sözcüğü dizisi iletebilecektir. Bir kod dizisinin doğru olarak alınabilmesi için vericinin göndermesi gereken ortalama bit sayısı:

$$\begin{aligned} T_{DB} &= (n+Dv)P_c + 2(n+Dv)P_c(1-P_c) + 3(n+Dv)P_c(1-P_c)^2 + \\ &+ \dots + i(n+Dv)P_c(1-P_c)^{i-1} + \dots \\ &= (n+Dv)P_c [1 + 2(1-P_c) + 3(1-P_c)^2 + \dots] \end{aligned}$$

$$\begin{aligned}
&= (n+Dv)P_c / (1-P_c) \sum_{i=1}^{\infty} i(1-P_c)^i \\
&= (n+Dv)P_c / (1-P_c) [(1-P_c) / (1-(1-P_c))^2] \\
&= (n+Dv)P_c / P_c^2 = (n+Dv) / P_c \quad (2.6)
\end{aligned}$$

olarak bulunur. İş çıkarma yeteneği de

$$\eta_{DB} = k/T_{DB} = P_c / (1+Dv/n) \cdot (k/n) \quad (2.7)$$

biçiminde ifade edilir.

Görüleceği gibi yüksek veri hızına ve uzun bekleme süresine sahip DB OYİ dizgelerde iş çıkarma yeteneği oldukça düşüktür.

N-AGD OYİ DİZGELERİ İÇİN İŞ ÇIKARMA YETENEĞİ

Bu dizgede, bir kod sözcüğü hatalı alındığında, kendisinden sonra gönderilen N-1 kod sözcüğünün de hatalı kabul edildiğini göz önünde tutarak, bir kod dizisinin doğru olarak alınabilmesi için gerekli ortalama iletim sayısı;

$$\begin{aligned}
T_{N-AGD} &= P_c + (N+1)P_c(1-P_c) + (2N+1)P_c(1-P_c)^2 + \dots \\
&\quad + (iN+1)P_c(1-P_c)^i + \dots \\
&= P_c + NP_c(1-P_c) + P_c(1-P_c) + 2NP_c(1-P_c)^2 + \\
&\quad P_c(1-P_c)^2 + \dots + iNP_c(1-P_c)^i + P_c(1-P_c)^i + \dots \\
&= P_c + P_c(1-P_c) + P_c(1-P_c)^2 + \dots + P_c(1-P_c)^i + \dots \\
&\quad + NP_c(1-P_c) + 2NP_c(1-P_c)^2 + \dots + iNP_c(1-P_c)^i + \dots \\
&= P_c(1 + (1-P_c) + (1-P_c)^2 + \dots + (1-P_c)^i + \dots) + \\
&\quad NP_c[(1-P_c) + 2(1-P_c)^2 + \dots + i(1-P_c)^i + \dots] \\
&= P_c \sum_{i=0}^{\infty} (1-P_c)^i + NP_c \left(\sum_{i=0}^{\infty} i(1-P_c)^i \right)
\end{aligned}$$

$$\begin{aligned}
 &= P_c (1/[1-(1-P_c)]) + NP_c \{ (1-P_c)/[1-(1-P_c)]^2 \} \\
 &= [P_c + N(1-P_c)]/P_c \quad (2.8)
 \end{aligned}$$

biçimindedir.

Buna göre dizgenin iş çıkarma yeteneği:

$$\eta_{N-AGD} = 1/T_{N-AGD} \cdot (k/n) = P_c (k/n) / [P_c + N(1-P_c)] \quad (2.9)$$

olarak bulunur.

(2.9) bağıntısında görüldüğü gibi, dizgenin iş çıkarma yeteneği hem kanal hata olasılığına hem de gidiş-geliş gecikmesine bağlıdır. Yüksek veri oranlı ve uzun gidiş-geliş gecikmeli dizgeler için N değerinin çok büyük olması nedeniyle iş çıkarma yeteneği hızla düşer.

SEÇEREK YİNELEMELİ OYİ DİZGELER İÇİN İŞ ÇIKARMA YETENEĞİ

Bu dizgeyi, alıcıda bulunan ara belleğin uzunluğunu sonsuz sayarak ele alalım. Böyle bir dizge, ideal seçerek yinelemeli olarak adlandırılır. Bir kod sözcüğünün hatasız olarak kabul edilebilmesi için:

$$\begin{aligned}
 T_{SY} &= P_c + 2P_c(1-P_c) + 3P_c(1-P_c)^2 + \dots \\
 &+ iP_c(1-P_c)^{i-1} + \dots \\
 &= P_c [1 - 2(1-P_c) + 3(1-P_c)^2 + \dots \\
 &+ i(1-P_c)^{i-1} + \dots] \\
 &= P_c / (1-P_c) \sum_{i=0}^{\infty} i(1-P_c)^i \\
 &= P_c / (1-P_c) \cdot \{ (1-P_c) / [1-(1-P_c)]^2 \}
 \end{aligned}$$

$$=P_c/P_c^2 = 1/P_c \quad (2.10)$$

olup, iş çıkarma yeteneği ise;

$$\eta_{sy} = 1/T_{sy} \cdot (k/n) = P_c k/n \quad (2.11)$$

dır.

Buradan çıkaracağımız sonuç, bu dizgenin iş çıkarma yeteneğinin yalnızca kanal bit hata olasılığına ve kod oranına bağlı olduğudur. Dolayısıyla uzun gidiş-geliş gecikmeli ve yüksek veri hızlı haberleşme dizgeleri için diğer yöntemlerden daha iyi iş çıkarma yeteneğine sahiptir.

2.2 KARMA OYİ DİZGELER

Temel OYİ yöntemi ile İYHD yönteminin birlikte kullanılmasıyla gerçekleştirilen dizgelere, Karma OYİ dizgeler denir. Bilindiği gibi İYHD yönteminde iş çıkarma yeteneği yüksektir. OYİ yönteminde ise güvenilirlik yüksektir. Hem güvenilir hem de yetenekli bir haberleşme dizgesi oluşturmak, istenen bir durumdur. Bu nedenle İYHD ve OYİ yöntemleri uygun bir şekilde birleştirilerek güvenilir ve yetenekli bir haberleşme dizgesi gerçekleştirilebilir. Bu dizgeler iki temel yapıda olabilirler ve 1. tip karma OYİ dizgeler ve 2. tip karma OYİ dizgeler olarak adlandırılırlar.

1. TİP KARMA OYİ DİZGELER

Bu dizgelerde hata sezme ve düzeltme için bir tek

kod kullanılır. Alınan bir kod sözcüğünde hatalı bit sezildiğinde, kullanılan kodun hata düzeltme yeteneğine göre hata düzeltme yapılır ve kod çözme işleminden sonra hatasız veri bloğu duruma göre ya doğrudan kullanıcıya iletilir ya da iletilmek üzere alıcıdaki ara belleğe yerleştirilir. Sezilen ama düzeltilemeyen bir hata ile karşılaşır, o kod sözcüğünün yeniden iletilmesi istenir. O kod sözcüğü hatasız iletilene kadar, yeniden gönderilmesi istenir. Bu dizgelerde iletim, yeniden iletim şekli üç temel OYİ yönteminden herhangi biriyle olabilir.

Hataların sezilmesi ve düzeltilmesi bir kod ile gerçekleştirildiğinden iş çıkarma yeteneği bu kodun uygun seçilmesine bağlıdır. Seçilen kod, temel OYİ dizgelerde kullanılan kodlara göre çok daha fazla eşlik biti içeren uzun bir kod olmalıdır. Bunun getirdiği olumsuzluk, düşük kanal bit hata olasılığı durumlarında iş başarıma yeteneğinin temel OYİ dizgelerden daha düşük olmasıdır. Gürültü düzeyinin sabit olduğu ve girişim etkisinin göz ardı edilebileceği kanallarda yeniden iletim sıklığı, bu dizgelerde azalacağından, iş başarıma yeteneği artar. Yalnız durağan olmayan kanallarda bu dizgeler iyi başarımlar sağlayamazlar.

2. TİP KARMA OYİ DİZGELER

Bu tip dizgelerde, bir bilgi dizisi yalnızca hata

sezme amacıyla eşlik bitleriyle kodlanır ve gönderilir. Hata düzeltmek amacıyla kullanılan eşlik bitleri ise yalnızca alıcı tarafından gereksinim duyulduğunda gönderilir. Bu nedenle durağan olmayan bir kanal için böyle bir dengeleyici dizge çok uygundur. Kanalda gürültünün etkinliğinin büyük olmadığı durumlarda dizge tamamen OYI dizgesi gibi davranır ve her iletimde yalnızca bir kod sezme için kullanılan eşlik kontrol bitleri ile kodlanmış kod dizisi gönderilir. Kanal gürültüsünün yeterince büyük olduğu durumlarda ise kod sözcüğünde hata saptanır. Bu hataların düzeltilmesi amacıyla kod sözcüğü, alıcı ara belleğinde saklanır ve kod sözcüğünün yeniden iletimi için verici uyarılır. Yeniden iletim, esas kod sözcüğünün aynısı olmayıp, esas iletim, bir hata sezen kod ve bir hata düzelten koda göre oluşturulan bir eşlik kanal dizisidir. Eşlik kontrol dizisi, alıcı ara belleğinde saklanan hatalı sözcükteki hataları düzeltme işleminde kullanılır. Eğer hata düzeltme işlemi başarılı olmazsa, alıcı alınmış olumsuz kod dizisinin ikinci kez iletimini ister. İkinci kez yeniden iletim işlemi, ya esas kod sözcüğünün ikinci kez iletimidir, ya da başka bir eşlik-kontrol dizisinin iletimidir. Bu seçenek, kullanılan hata düzelten kodun tipine ve yeniden iletim yöntemine bağlıdır.

Eğer hata düzeltme işlemi için kullanılan kod ve yeniden gönderme yöntemi uygun bir şekilde seçilirse, ikin-

ci türden karma OYİ dizgesi temel OYİ dizgelerinden ve birinci türden karma OYİ dizgelerinden daha iyi iş çıkarma yeteneği sağlar. Yeniden gönderme yöntemi olarak temel OYİ dizgelerinin herhangi biri seçilebilir. SY OYİ yöntemi seçilirse en iyi başarımlar sağlanır.

Ard arda iki yineleme noktası arasında iletilen blokların sayısı X ve aynı aralık boyunca alıcı tarafından kabul edilip kullanıcıya iletilen blok sayısı Y ile gösterilse, iş çıkarma yeteneği genel olarak;

$$\eta = \frac{E[Y]}{E[X]} \quad (2.12)$$

bağıntısıyla verilmektedir. $E[X]$ ve $E[Y]$, X ve Y raslantı değişkenlerinin beklenen değerleridir. Karma OYİ dizgeleri için iş çıkarma yeteneği bağıntılarını çıkarmak oldukça güçtür. Bu nedenle, daha basit yapıya sahip alt dizgeler için bağıntılar çıkarılarak bir alt sınır belirlemek kolaylık sağlar. Hata sezme için, sistematik (n,k) C_0 doğrusal blok kodu ve hata düzeltme için de $1/2$ oranlı bir $(2,1,m)$ C_1 katlamalı kodu kullanan bir karma OYİ dizge için iş çıkarma yeteneği ve güvenilirlik bağıntıları için alt sınırlar [15]' de çıkarılmıştır. Buna göre güvenilirliğin temel OYİ dizgelere göre en az iki kat daha iyi olduğu sonucuna varılmıştır.

3. BİRLEŞİK KODLAMALI MODÜLASYON TEKNIĞİNE DAYALI OTOMATİK YİNELEME İSTEMELİ DİZGELER.

Genellikle, band genişliğini sınırlı tutarak işaret gücünü artırmak ya da güç sınırlaması altında bant genişliğini artırmak dizgelerin hata olasılığını küçültebilmektedir. İlk kez Ungerboeck [16] tarafından ortaya atılan ve kafes kodlamalı modülasyon (KKM) (trellis coded modulation, TCM) olarak adlandırılan bir yöntem ise, kullanılan işaret kümesini gereğinden fazla tutma ilkesine dayalıdır. Kafes kodlamalı modülasyon yönteminde kodlama ve modülasyon bir bütün olarak düşünülmekte, sabit iletim band genişliği için aynı iletim hızında önemli güç kazançları sağlanmaktadır. Gerektiğinden iki kat fazla kanal işaretinden oluşan işaret kümesi, kodlanmış işaret dizilerini oluşturmak üzere modülasyonlu işaretlerin seçimini yönlendiren sonlu durumlu bir kodlayıcıyla birlikte kullanılmaktadır. Böylece faz kaydırmalı anahtarlama (phase shift keying, PSK) ile dik genlik modülasyonlu dizgeler için 3-4 dB'lik kodlama kazançları sağlanmaktadır. Daha karmaşık KKM yapılarıyla 6 dB ve daha fazla kazanç sağlanabilmektedir. Bu kazançlar, alışılmış hata düzeltme yöntemlerinin gerektirdiği etkin kod oranları ve band genişliklerine gerek duyulmadan elde

edilebilmektedir.

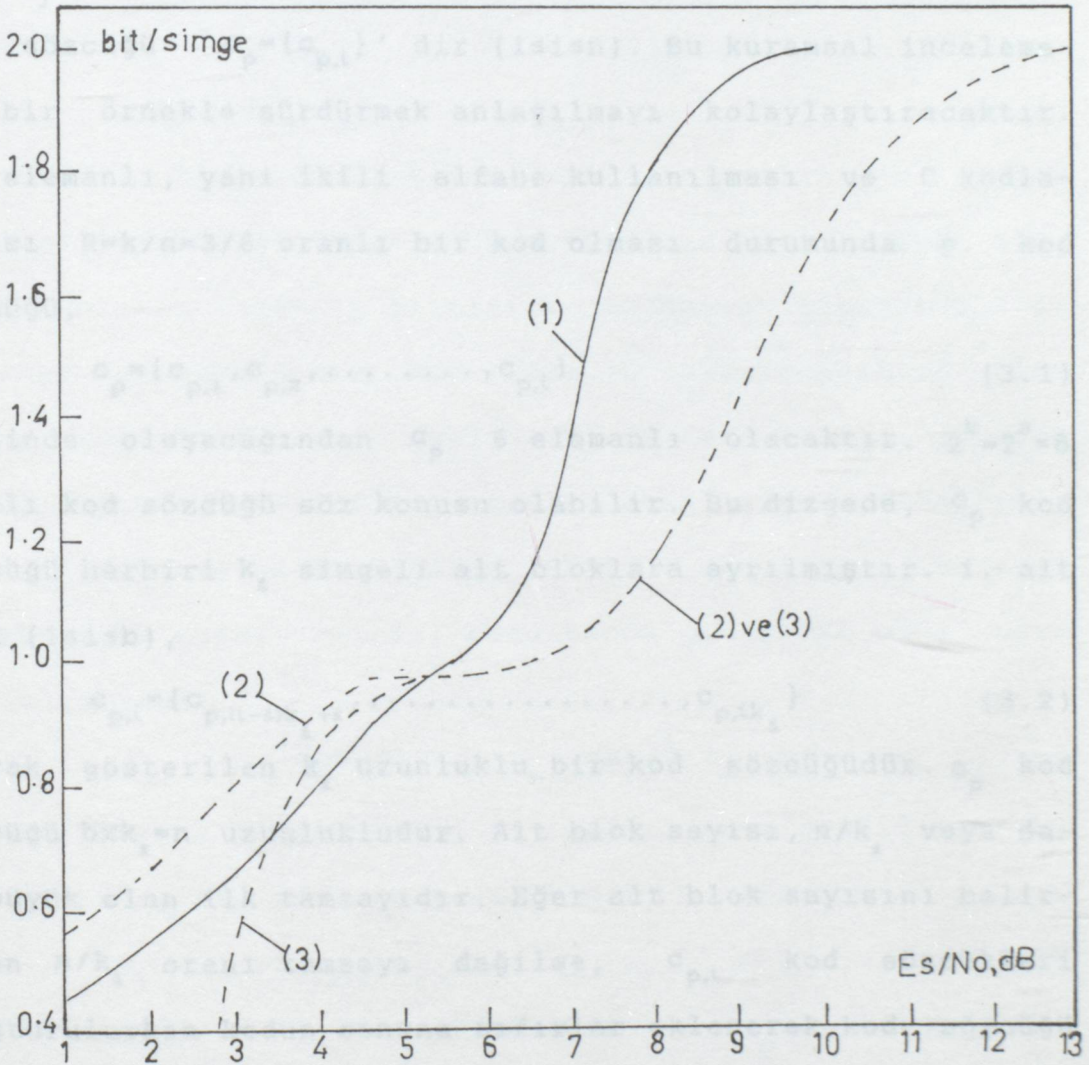
Birleşik Shannon' un bilişim kuramı otuz yıl öncesinden kodlamalı modülasyonu öngörebilmiştir. İşaretler arası serbest uzaklığın (en küçük Öklid uzaklığı) büyük olması, gürültünün etkisini azaltıcı bir etkidir. KKM'de serbest uzaklığı artırarak başarıımı yükseltmek amaçlanmaktadır. Kodlama ve modülasyonun birlikte kullanılması, kodlanmış işaret dizileri arasındaki Öklid uzaklıklarını büyütür. Bu birleşik kodlamalı modülasyon yöntemine kafes kodlamalı modülasyon adının verilmesi; bu yapıların ikili katlamalı kodların kafes gösterimine benzemesinden kaynaklanmaktadır. Aralarındaki tek fark; kafes kollarının ikili kod simgeleri yerine, ikili olmayan modülasyonlu işaretlerle adlandırılmasıdır. Bu konudaki çalışmalar 1984' de, 4 dB kazançlı birleşik kodlamalı modülasyonun Uluslararası Telgraf ve Telefon Kuruluşu (CCITT) tarafından kabul edilmesiyle, yüksek hızlı ses bandı modemlerinde kullanılmıştır. KKM' den önce, kodsuz veri aktarım modemlerinde sınır olarak 9.6 kbit/s hızlara ulaşılmıyorken, KKM ile 14.4 kbit/s ve daha yüksek hızlarda veri akışı güvenilir bir şekilde sağlanabilmektedir.

Temel OYİ yöntemlerinden en iyi başarıımı sağlayanın SY yöntemi olduğu, ancak karma OYİ dizgeler kullanılarak başarımın daha da iyileştirilebileceği, ikinci bölümde ayrıntılı olarak belirtilmiştir. Bu dizgelerde kodlama ve

modülasyon işlemleri ayrı düşünülmüştür. Daha sonraları birleşik kodlamalı modülasyon konusunda yapılan çalışmalar ve alınan olumlu sonuçlar OYİ dizgelere de bu yöntem uygulanarak Öklid uzaklıklarının en yüksek değerlerde tutulabileceği, dolayısıyla dizge başarımının artırılabilceği düşünülmüştür. Bu çalışmaların büyük çoğunluğu Benelli [11], [12], [17] tarafından gerçekleştirilmiştir. Benelli, çalışmalarında sürekli fazlı frekans kaydırmalı anahtarlama modülasyonunu ve M' li modülasyon türlerini ele almaktadır. Bunların dışında, Deng [18] M' li faz kaydırmalı anahtarlama (M' ary phase shift keying, MPSK) temeline dayandırdığı KKM kullanan karma OYİ dizgesinin işaret-gürültü oranının (I/GO), (E_s/N_o) 5 dB' den büyük değerleri için, katlamalı kod kullanan karma II. türden OYİ dizgelerinden daha iyi iş çıkarma yeteneği sağladığını ortaya koymuştur. Şekil-3.1' de 1 numaralı eğri tasarlanan KKM 'li karma OYİ dizgesinin, 2 numaralı eğri (2,1,2) katlamalı kodunu ve QPSK birleşik modülasyonunu kullanan bir karma OYİ dizgesinin, 3 numaralı eğri ise; (2,1,2) katlamalı kodunu kullanan bir karma OYİ dizgesinin iş çıkarma yeteneklerini ortaya koymaktadır. Eğriler, I/GO değerlerine bağlı olarak verilmiştir.

Benelli [12]' nin ortaya koyduğu çalışmada, yüksek hata oranı durumlarında hatalı olduğu sezilen bir kod sözcüğünün m kez yeniden iletilmesi ve birleşik kodlama

uygulanması iş çıkarma yeteneğini artırmaktadır. Kurulan dizgede alıcı tarafta bellek bulunması ve bulunmaması durumları ayrı ayrı incelenmiştir. Kaynak M elemanlı sonlu $A=\{a_1, a_2, \dots, a_M\}$ alfabesinden simgeler üretmektedir. Kaynağın ürettiği k bilgi simgeli her blok, n simge uzunluklu (n, k) tipi bir C koduyla kodlanmıştır. C kodu



Şekil-3.1 İş çıkarma yeteneği karşılaştırmaları.

(1); KKM kullanan OYİ, (2); katlamalı kod ve QPSK modüleyonlu OYİ, (3); yalnızca katlamalı kod kullanan OYİ.

$s > t$ hatayı sezme ve $t \geq 0$ hatayı düzeltme yeteneğine sahip ve $R = k/n$ oranlıdır. Bu kodun oluşturduğu herhangi bir c kod sözcüğünün j . iletimi sırasında gönderilen kopyalarının sayısı m_j ile gösterildiğine göre; ya j 'nin olası tüm değerleri için $m_j = m$ gibi sabit bir değer olarak ya da $m_1 = 1$ ve $m_j = m$, $j > 1$ olarak belirlenir. $1 \leq p \leq 2^k$ olmak üzere, p . kod sözcüğü $c_p = \{c_{p,i}\}$ 'dir ($1 \leq i \leq n$). Bu kuramsal incelemeyi bir örnekle sürdürmek anlaşılmayı kolaylaştıracaktır. $M=2$ elemanlı, yani ikili alfabe kullanılması ve C kodlayıcısı $R = k/n = 3/6$ oranlı bir kod olması durumunda p . kod sözcüğü,

$$c_p = (c_{p,1}, c_{p,2}, \dots, c_{p,i}) \quad (3.1)$$

şeklinde oluşacağından c_p 6 elemanlı olacaktır. $2^k = 2^3 = 8$ farklı kod sözcüğü söz konusu olabilir. Bu dizgede, c_p kod sözcüğü herbiri k_1 simgeli alt bloklara ayrılmıştır. i . alt blok ($1 \leq i \leq b$),

$$c_{p,i} = \{c_{p,(i-1)k_1+1}, \dots, c_{p,ik_1}\} \quad (3.2)$$

olarak gösterilen k_1 uzunluklu bir kod sözcüğüdür. c_p kod sözcüğü $b \times k_1 = n$ uzunlukludur. Alt blok sayısı, n/k_1 veya daha büyük olan ilk tamsayıdır. Eğer alt blok sayısını belirleyen n/k_1 oranı tamsayı değilse, $c_{p,i}$ kod sözcükleri oluşturulurken kodun sonuna sıfırlar eklenerek kod sözcüğü tamamlanır. $k_1 = 1$ seçilmesi durumunda $b = 6$ alt blok oluşturulabilir. Buna göre, c_p yi oluşturan herbir karakter bir alt blok ifade etmektedir. Oluşturulan kod sözcükleri,

$$\mathbf{c}_p = (c_{p,1}, c_{p,2}, \dots, c_{p,b}) \quad (3.4)$$

şeklindedir. Herbir alt blok, ikinci bir kodlayıcı olan C1 kodlayıcısıyla kodlanır. Bu kodun bilgi simgeleri k_1 tane ve bir kod sözcüğünün uzunluğu ise $n_1 = m_j \times k_1$ olarak belirlenmektedir. $R = k_1 / m_j \times k_1 = 1/m_j$, C1 kodlayıcısının kod oranını ifade eder. C1 kodlayıcısının oluşturacağı kod sözcükleri ,

$$\mathbf{c}'_p = (c'_{p,1}, c'_{p,2}, \dots, c'_{p,b}) \quad (3.5)$$

şeklindedir. C1 kodu blok kod veya katlamalı kod olarak seçilebilir. $b=6$ alt bloklu bir uygulama için; $m_j=2$ ise, $n_1=2$ uzunluklu karakter çiftleri, kodlayıcı çıkışında 12 karakter uzunluklu bir kod sözcüğü oluşturacaktır. Elde edilen kod sözcükleri, şekil-3.2' de faz kafesi görülen sürekli fazlı frekans kaydırmalı anahtarlama (continuous phase frequency shift keying, CPFSK) modülasyon yöntemi kullanılarak modüle edilir. Modülasyon çıkışında elde edilen dalga şekli,

$$s_{p,i}(t) = \sqrt{(2E/T)} \cdot \cos(\omega_0 t + \pi h t / T \cdot c'_{p,i} + x_i) \quad (3.6)$$

olup, E; işaret enerjisini, T; işaret zaman aralığını, h; modülasyon katsayısını göstermektedir. x_i ise,

$$x_i = x_{i-1} + (c'_{p,i-1} - c'_{p,i}) h \pi (i-1) \quad (3.7)$$

şeklinde belirtilen faz terimidir ve $x_0 = 0$ varsayılır. Şekil-3.2' de görülen CPFSK modülatörünün modülasyon katsayısı, $h=1/2$ dir.

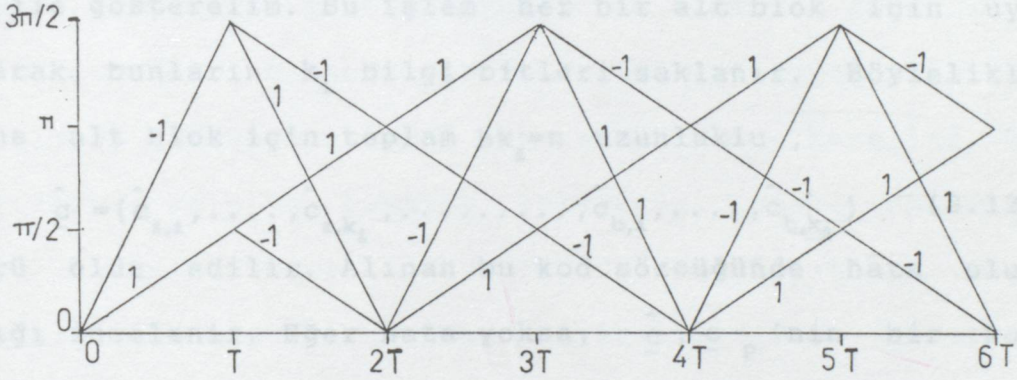
$\mathbf{c}'_{p,i}$ vektörüne karşı düşen $n_1 T$ sn uzunluklu

işaret vektörü $\underline{s}_{p,i}$ dir.

$$\underline{s}_{p,i} = (\underline{s}_{p,(i-1)k_1+1}(t), \dots, \underline{s}_{p,ik_1}(t)) \quad (3.8)$$

Bu yaklaşımla c_p kod sözcüğü, $bn_1T = n/k_1 m_j k_1 T = nm_j T$ sn uzunluklu $\underline{s}_p$ işaret vektörü ile ifade edilmektedir:

$$\underline{s}_p = (\underline{s}_{p,1}, \underline{s}_{p,2}, \dots, \underline{s}_{p,b}) \quad (3.10)$$



Şekil-3.2 $h=1/2$ olan bir CPFSK modülasyonunun faz kafes gösterilimi.

Bu işaret kanaldan gönderildikten sonra, kanal çıkışında yine $nm_j T$ sn uzunluklu olan $\underline{r}$ vektörü elde edilir. Bu vektör $n_1 T$ uzunluklu b tane alt bloktan oluşmuştur. i . alt blok işareti,

$$\underline{r}_i = (\underline{r}_{(i-1)n_1+1}(t), \dots, \underline{r}_{in_1}(t)) \quad (3.11)$$

şeklindedir. Burada $\underline{r}_j(t)$, j . işaret zaman aralığında alı-

naın işarettir. Her bir alt blok, farklı bilgi vektörlerine karşı düşen M^k_1 farklı yapıya sahip olacaktır.

Alıcı, i. alınan alt blok $\underline{r}_i$ ile p. kod sözcüğünün i. alt bloğu arasındaki, her bir temel öyl yöntemi için

$$d^2_{i,p} = d^2(\underline{r}_i, \underline{s}_{p,i}), \quad 1 \leq p \leq 2^k_1 \quad (3.12)$$

$1 \leq p \leq 2^k_1$ şeklinde ifade edilen Öklid uzaklığını hesaplar.

Aynı kod sözcüğünün i. alt bloğu için M^k_1 farklı Öklid uzaklıkları bulunduktan sonra, bunlardan en küçük uzaklıklı olana karar verilmektedir. Demodüle edilmiş i. alt bloğu, $\hat{c}_{p,i}$ ile gösterelim. Bu işlem her bir alt blok için uygulanarak, bunların k_1 bilgi bitleri saklanır. Böylelikle b tane alt blok için toplam $bk_1 = n$ uzunluklu, karşılık

$$\hat{\underline{c}} = (\hat{c}_{1,1}, \dots, \hat{c}_{1,k_1}, \dots, \hat{c}_{b,1}, \dots, \hat{c}_{b,k_1}) \quad (3.13)$$

vektörü elde edilir. Alınan bu kod sözcüğünde hata olup olmadığı incelenir. Eğer hata yoksa; $\hat{\underline{c}}, \underline{c}_p$ 'nin bir kod sözcüğüdür diye yorumlayarak geri besleme kanalından vericiye bir onaylama bilgisi gönderilir ve tersi durumda vericinin aynı kod sözcüğünü yeniden iletmesi sağlanır.

Alıcıda bellek kullanılması durumunda ise, her bir iletimde bulunan Öklid uzaklıkları 2 matrisi yapısında saklanır ve bulunan her bir Öklid uzaklıkları toplanarak toplamsal Öklid uzaklıkları,

$$D_{i,p}(j) = D_{i,p}(j-1) + d^2_{i,p}(j) \quad n=3 \text{ olan ya da} \quad (3.14)$$

belirlenir. Burada; $1 \leq p \leq M^k_1$, $1 \leq i \leq b$, j iletim sayısı ve $D(j)$ Öklid uzaklıklarının matris terimidir. Bu durumda da $D_p(j)$

değerlerine bakarak, belleksiz yapıda uygulanan tüm işlemler aynen uygulanmaktadır.

Bu çalışmada ortaya koyulan yapı, bir karma OYİ dizge yapısıdır ve her bir temel OYİ yöntemi için uygulanabilmektedir. Modülasyon işleminde, Öklid uzaklığını en büyük tutacak modülasyon çarpanı ile çalışan bir modülasyon yapısı kullanılmıştır. Çeşitli kodlar için yapılan çalışmalar, modülasyon katsayısının 0.5 ve yakınındaki değerlerinde Öklid uzaklığının en yüksek değerlere eriştiğini gösterdiğinden $h=0.5$ olarak alınmıştır.

Şekil-3.3 C1 kodu olarak değişik blok kodlayıcılar kullanılması durumunda kanal hata olasılığına karşılık DB OYİ yöntemi için iş çıkarma yeteneğini göstermektedir. Burada, $Dv/n=100$ olarak alınmış, üç farklı tipte blok kodlayıcı için inceleme yapılmıştır. Sonuçlar, modülasyon katsayısının 0.4 ve 0.5 değerleri için gösterilmiştir. Ayrıca karşılaştırma yapmak amacıyla, temel DB yapısı a eğrisiyle ve Moeneclaey ve Bruneel [8] tarafından önerilen yapı b eğrisiyle verilmiştir.

Şekil-3.4 C1 kodu olarak $R=1/2$ oranlı katlamalı kodlar kullanılması durumunda DB yöntemi için iş çıkarma yeteneğini farklı Öklid uzaklıkları için göstermektedir.

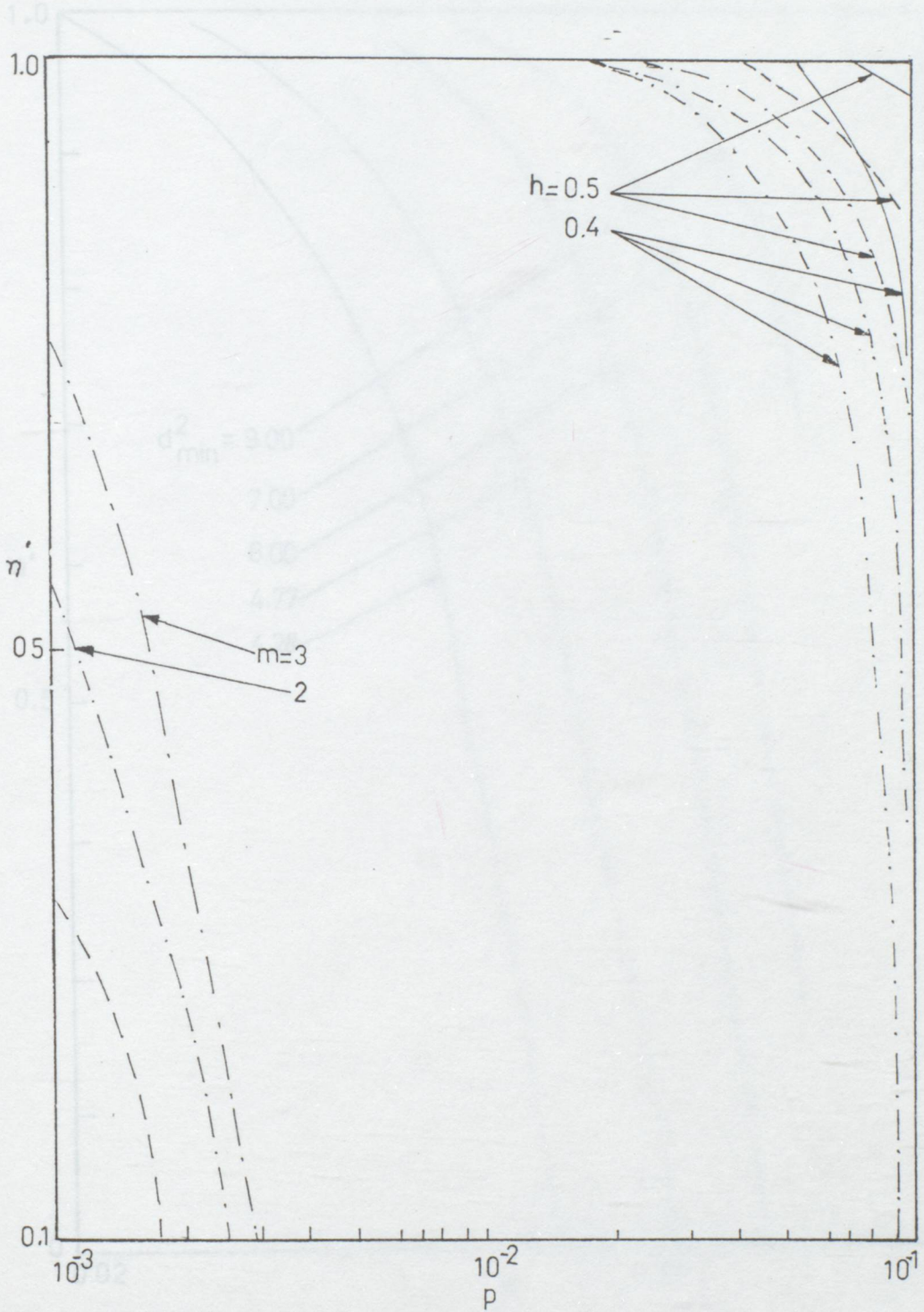
Şekil-3.5 ise, $R=1/3$ ve $m=3$ olan ya da $R=1/4$ ve $m=2$ olan katlamalı kodlar için DB yöntemine göre iş çıkarma yeteneğini yine farklı Öklid uzaklığı değerleri

için göstermektedir.

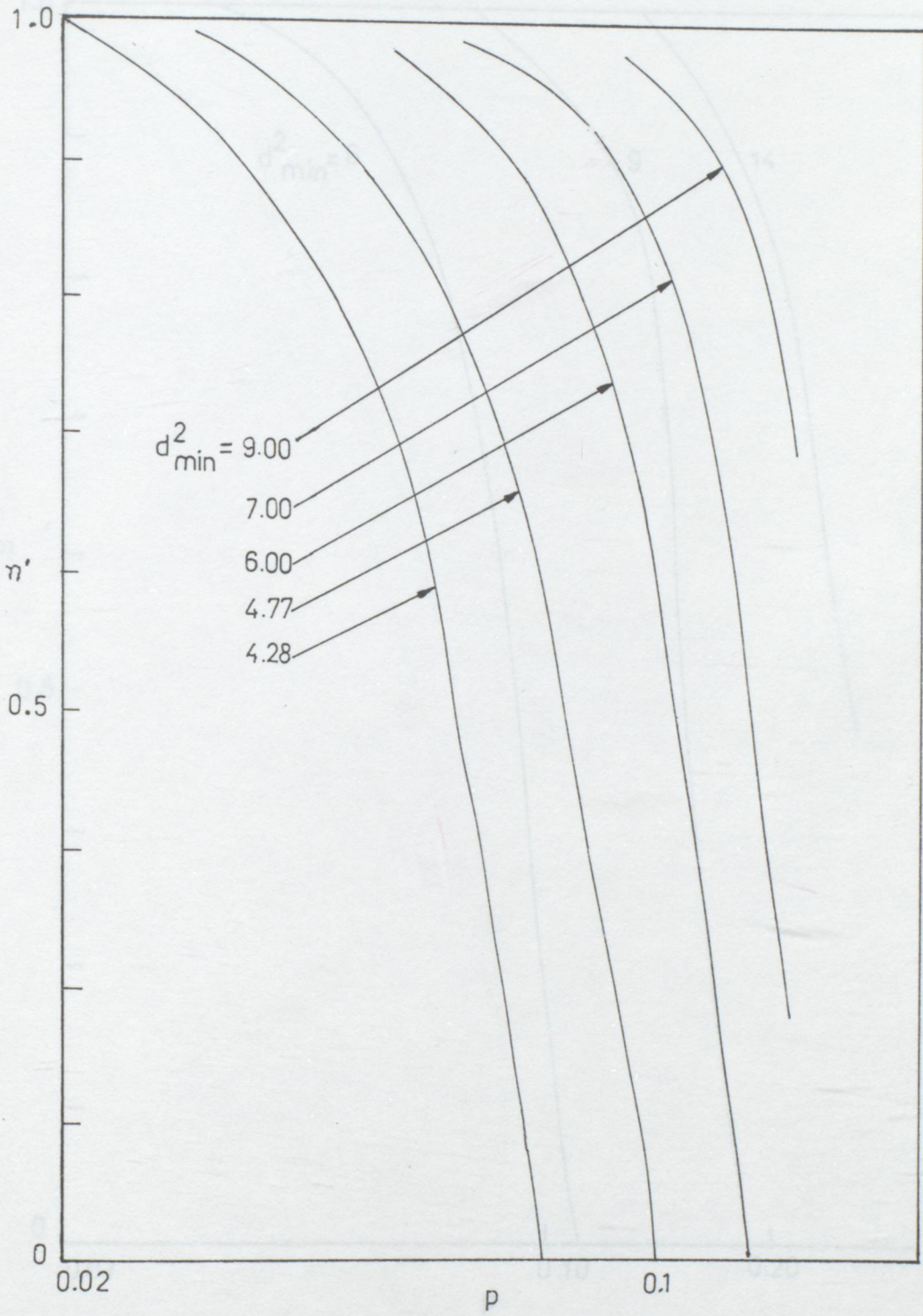
Şekil-3.6' da ise sürekli yinelemeye örnek olmak üzere; N-AGD yöntemi için , $R=1/2$ oranlı katlamalı kodlar kullanan yapıda kanal hata olasılığına göre iş çıkarma yeteneği karşılaştırmaları yer almaktadır. Elde edilen gelişmeleri vurgulamak üzere, temel N-AGD yöntemi a eğri- siyle ve Sastry [1]' in geliştirdiği yapı b eğrisiyle, oluşturulan dizgenin de farklı Öklid uzaklıkları için aldığı değerler c,d,e eğrileriyle ortaya konulmuş ve karşılaştırmalar yapılmıştır. Bu eğriler $m=2$ ve $N=100$ de- ğerleri için iş çıkarma yeteneğinin değişimlerini göster- mektedir.



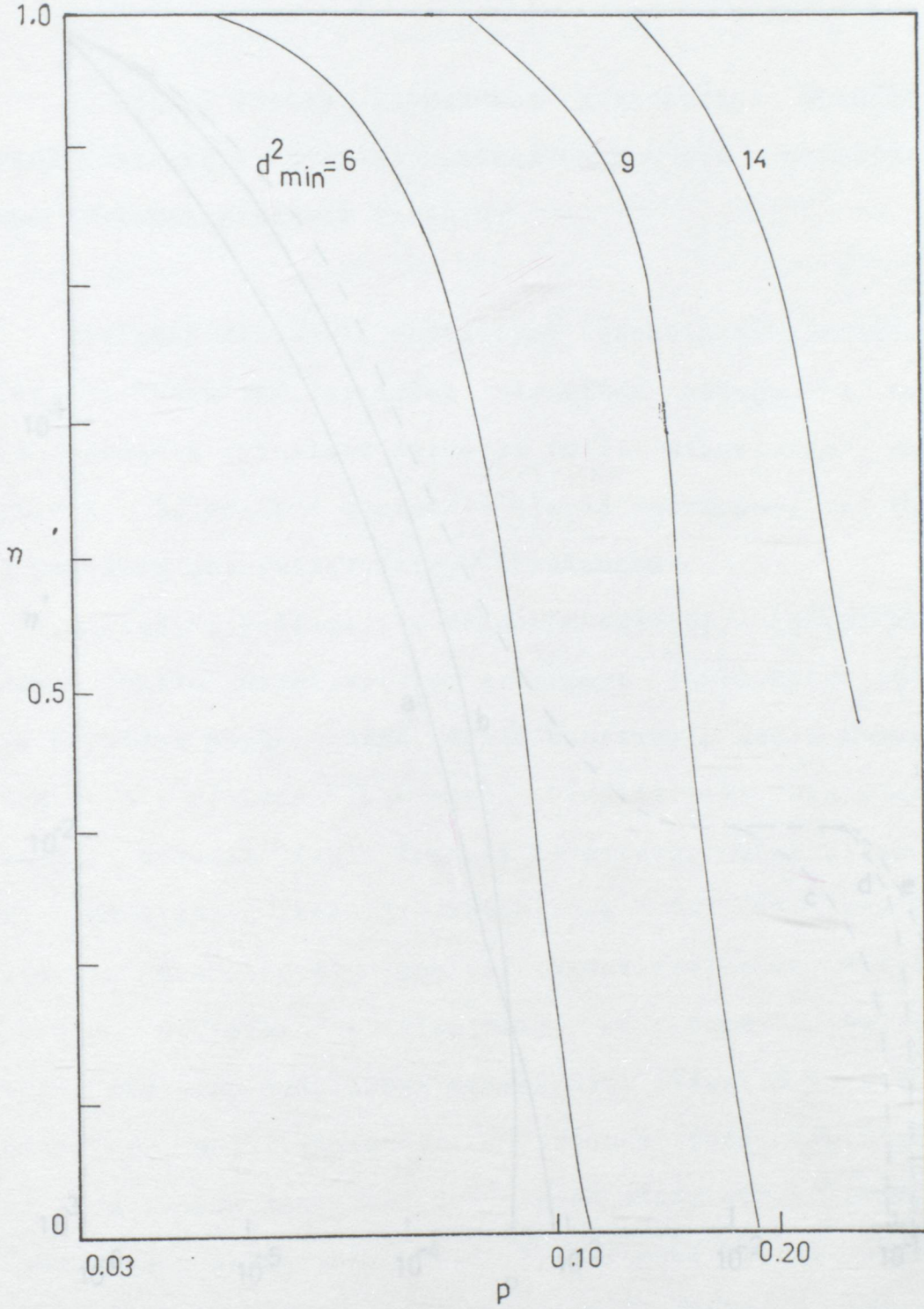
Şekil-3.3 Blok kodlar kullanan DB yapısı için Kanal Hata olasılığına karşı iş çıkarma yeteneği
— (15,5) kodu, - - - (10,5) kodu, ... (5,5) kodu



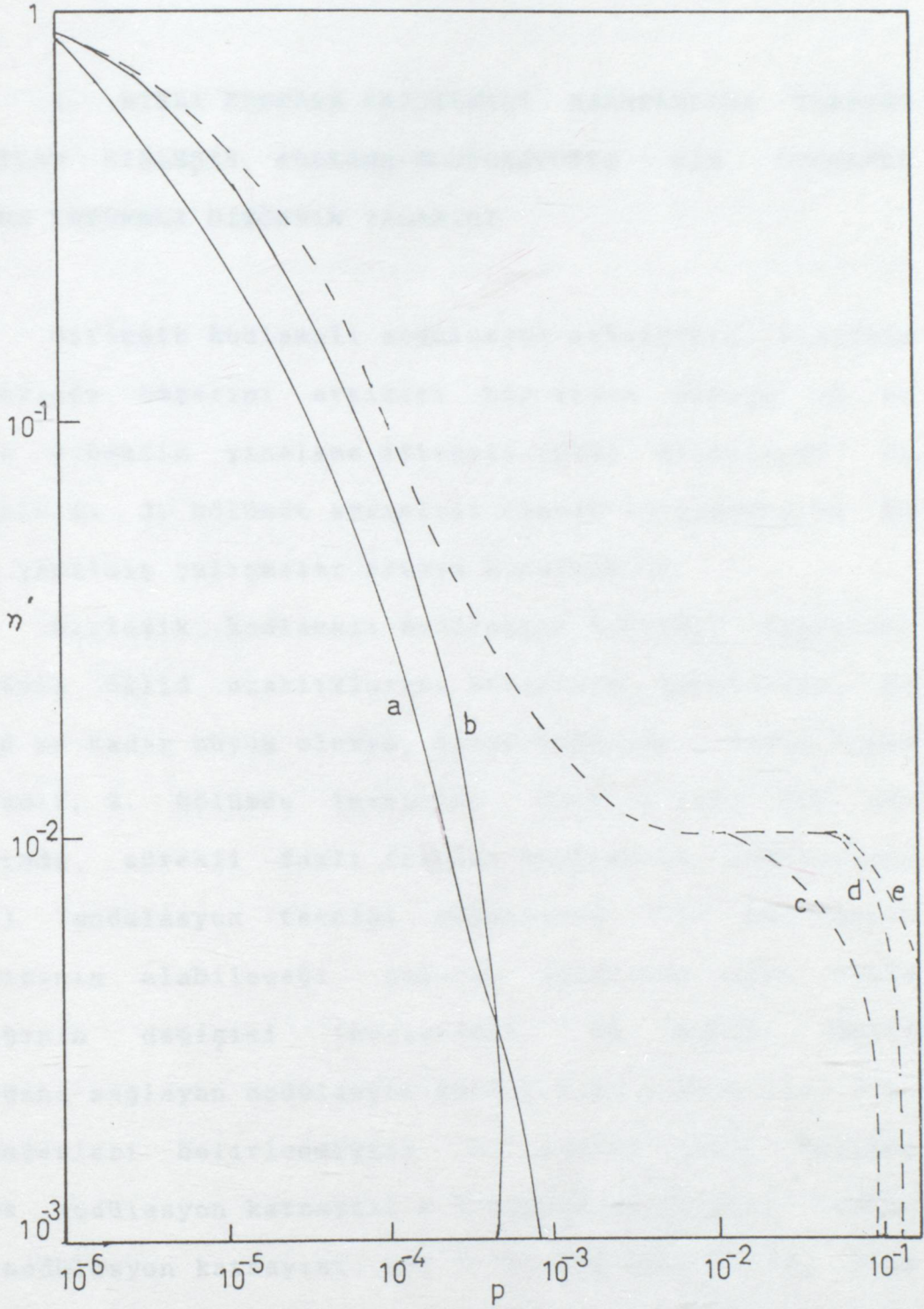
Şekil-3.3 Blok kodlar kullanan DB yapısı için kanal hata olasılığına karşı iş çıkarma yeteneği.
 — (15,5) kodu, - - - (10,5) kodu, ._. (8,4) kodu



Şekil-3.4 $R=1/2$ oranlı katlamalı kodlar kullanılan DB yapısının kanal hata olasılığına karşı, iş çıkarma yeteneği.



Şekil-3.5 $R=1/3$ veya $R=1/4$ oranlı katlamalı kodlar kullanılan DB yapısı için kanal hata olasılığına karşı, iş çıkarma yeteneği.



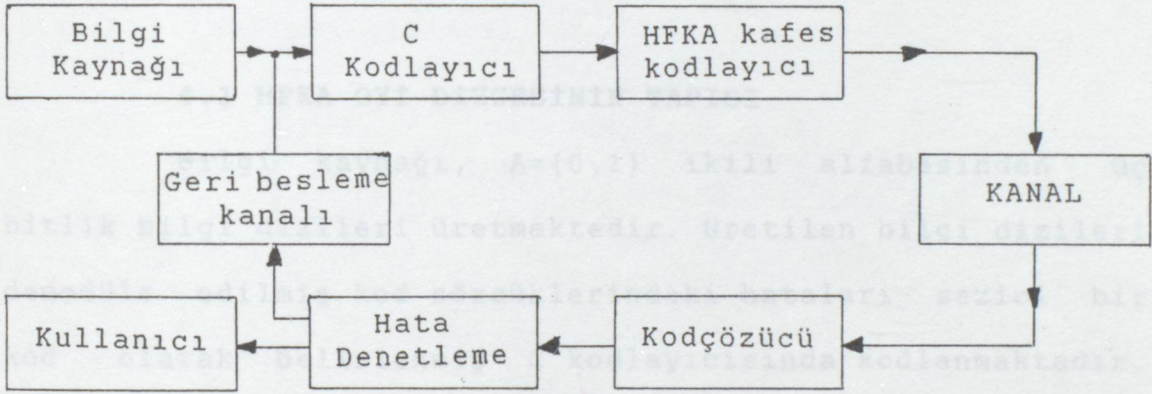
Şekil-3.6 $R=1/2$ oranlı katlamalı kodlar kullanılan N-AGD yapısının kanal hata olasılığına göre iş çıkarma yeteneği. $d_{\min}^2 = c: 4.3, d: 7, e: 9$

4. HIZLI FREKANS KAYDIRMALI ANAHTARLAMA TEKNIĞİ KULLANILAN BİRLEŞİK KODLAMA-MODÜLASYONLU BİR OTOMATİK YİNELEME İSTEMELİ DİZGENİN TASARIMI

Birleşik kodlamalı modülasyon tekniğinin, iletişim dizgelerinde başarıımı artırıcı bir etken olduğu ve bu nedenle otomatik yineleme istemeli (OYİ) dizgelerde de kullanıldığı 3. bölümde ayrıntılı olarak incelenmiş ve bu konuda yapılmış çalışmalar ortaya konulmuştur.

Birleşik kodlamalı modülasyon tekniği, işaretler arasındaki Öklid uzaklıklarını artırmaya yöneliktir. Bu uzaklık ne kadar büyük olursa, dizge başarıımı o denli büyük olmaktadır. 3. Bölümde incelenen Benelli [12]'nin çalışmasında, sürekli fazlı frekans kaydırmalı anahtarlama (CPFSK) modülasyon tekniği kullanılmış ve modülasyon katsayısının alabileceği çeşitli değerlere göre Öklid uzaklığının değişimi incelenerek, en büyük Öklid uzaklığını sağlayan modülasyon katsayıları olarak 0.4, 0.5, 0.6 değerleri belirlenmiştir. Bu sonuca göre, kurulan dizgede modülasyon katsayısı 0.5 olarak alınmıştır. CPFSK'nın modülasyon katsayısı, $h=0.5$ ile bir özel biçimi olan hızlı frekans kaydırmalı anahtarlama (HFKA), (fast frequency shift keying, FFSK) sabit zarf, düşük spektral yayılma ve ikili faz kaydırmalı anahtarlama eşdeğer hata

başarımı özellikleriyle bant verimlilikli sayısal iletişim dizgeleri için yetkin bir modülasyon tekniğidir. Bu tez çalışmasında da sıralanan üstünlüklerinden dolayı modülasyon tekniği olarak HFKA kullanılmıştır. Ancak, bu çalışmada HFKA modülasyonuna ilişkin faz durumlarından kaynaklanan kafes yapı, katlamalı kodlayıcının yapısı ile bütünleştirilmektedir.



Şekil-4.1 Tasarlanan dizgenin blok şeması.

Tasarlanan dizgenin blok şeması, şekil-4.1' de görülmektedir. Bu dizge, temel OYİ yöntemlerinin (DB,N-AGD, SY) tümü için kullanılabilir. Bilgi kaynağınca üretilen bilgi dizileri, C kodlayıcısında kodlandıktan sonra HFKA kafes kodlamalı modülatörüne gelmektedir. Burada elde edilen modülasyonlu işaretlerin, toplamsal beyaz Gauss gürültülü kanal (additive white Gaussian noise, AWGN) üzerinden alıcıya ulaştığı varsayılmaktadır. Alınan

modülasyonlu işaretler, Viterbi Algoritmasıyla çözülerek hata denetleyici birime ulaşır. Burada kod sözcüklerinin hataları, C kodlayıcısının yapısına uygun olarak sezilir. Hata yoksa; kod sözcüğü kullanıcıya iletilir ve vericinin bilgi gönderme işine devam etmesi yolunda vericiye gürültüsüz bir geri besleme kanalından bilgi verildiği varsayılır. Hata varsa; vericinin aynı bilgi dizisini yeniden iletmesi doğrultusunda bilgi gönderilir.

4.1 HFKA OYİ DİZGESİNİN YAPISI

Bilgi kaynağı, $A=\{0,1\}$ ikili alfabesinden üç bitlik bilgi dizileri üretmektedir. Üretilen bilgi dizileri demodüle edilmiş kod sözcüklerindeki hataları sezici bir kod olarak belirlenmiş C kodlayıcısında kodlanmaktadır. C kodlayıcısı, $R=3/6$ oranlı bir doğrusal blok kodlayıcı olarak belirlenmiştir. Dolayısıyla $n=6$ bit uzunluklu, k bilgi biti ve $n-k$ eşlik bitine sahip kod sözcükleri elde edilmektedir. (n,k) şeklinde gösterilen doğrusal blok kodlayıcılarda, eşlik bitleri değişik şekillerde oluşturulabilmektedir. Kodlayıcı, k bitlik bilgiyi kod sözcüğünde ilk bitler olarak aynen yansıtıyorsa, bu tür doğrusal blok kodlayıcılara sistematik blok kodlayıcı adı verilmektedir. Kodlayıcıya gelen bilgi dizisi,

$$u \equiv (u^1, u^2, u^3) \quad (4.1)$$

ve kodlanmış sözcük de,

$$v \equiv (v^1, v^2, v^3, v^4, v^5, v^6) \quad (4.2)$$

şeklinde belirtilmişse, v kod sözcüğü (4.3)' deki ilişkiyle belirlenir.

$$v^i = \sum_{j=1}^k u^j g_{ji} \quad , 1 \leq i \leq k \quad (4.3)$$

$$, k+1 \leq i \leq n$$

Burada g_{ji} değerleri, üreteç matrisi G 'nin elemanlarıdır. Sistematik doğrusal blok kodlar için üreteç matrisi (4.4) ' de belirtildiği gibidir.

$$G = \begin{bmatrix} 1 & 0 & 0 & g_{1,4} & g_{1,5} & g_{1,6} \\ 0 & 1 & 0 & g_{2,4} & g_{2,5} & g_{2,6} \\ 0 & 0 & 1 & g_{3,4} & g_{3,5} & g_{3,6} \end{bmatrix} \quad (4.4)$$

Buna göre,

$$\underline{v} = \underline{u} \cdot G \quad (4.5)$$

şeklinde belirtilebilir. Tasarlanan dizge için üreteç matrisi (4.6)' da görülmektedir.

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \quad (4.6)$$

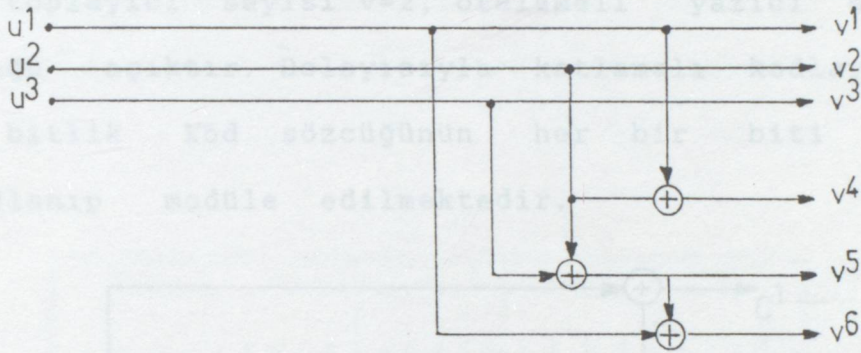
Buna göre, kodlanmış sözcük (4.7)' deki gibi olacaktır.

$$v^1 = u^1, v^2 = u^2, v^3 = u^3, v^4 = u^1 \oplus u^2, v^5 = u^2 \oplus u^3, v^6 = u^1 \oplus u^2 \oplus u^3 \quad (4.7)$$

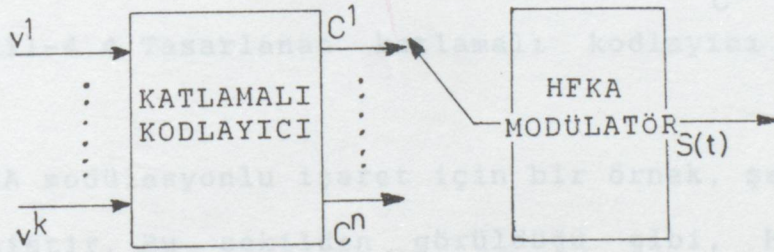
C kodlayıcı devresi de şekil-4.2' de görülmektedir.

Doğrusal blok kodlayıcı çıkışında elde edilen kod sözcüğü, HFKA kafes kodlayıcısına gelmektedir. HFKA kafes kodlayıcısı, birleşik kodlamalı modülasyon tekniğine

göre tasarlanmıştır. Modülasyon tekniği olarak, HFKA yapısı ve kodlayıcı olarak da $R=1/2$ oranlı bir katlamalı kod birarada düşünülmektedir. HFKA kafes kodlayıcı yapısı şekil-4.3' de görülmektedir.



Şekil-4.2 Tasarlanan sistematik doğrusal blok kodlayıcı devresi.



Şekil-4.3 HFKA kafes kodlayıcı genel yapısı.

Tasarlanan dizgede katlamalı kodlayıcı olarak, $(2,1)$ kodu seçilmiştir. Seçilen bu kodun devre şeması şekil-4.4' de görülmektedir.

Kodlayıcı her kodlama adımında,

$$c^1 = v \oplus s^2 \quad \text{ve} \quad c^2 = s^1 \oplus s^2 \quad (4.8)$$

bitlerini üretmektedir. Burada,

$$s^1 = v(-1), \quad s^2 = v(-2) \quad (4.9)$$

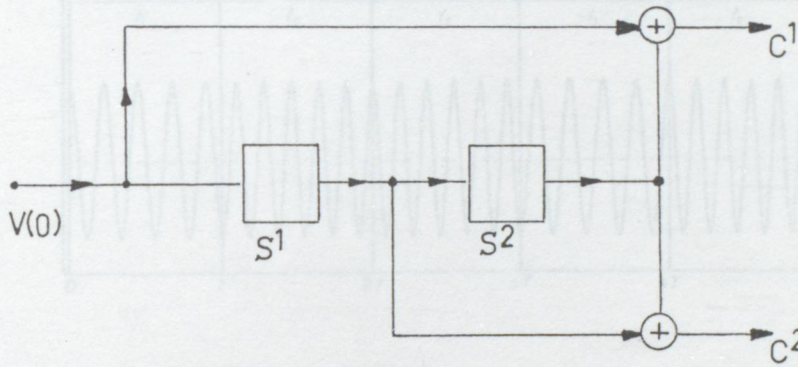
şeklinde tanımlı gecikme elemanlarıdır. Buna göre, katlama-

lı kodun toplayıcı sayısı $v=2$, ötelemeli yazıcı sayısı

$K=2$ olduğu açıktır. Dolayısıyla katlamalı kodlayıcıya

gelen 6 bitlik kod sözcüğünün her bir biti teker

teker kodlanıp modüle edilmektedir.

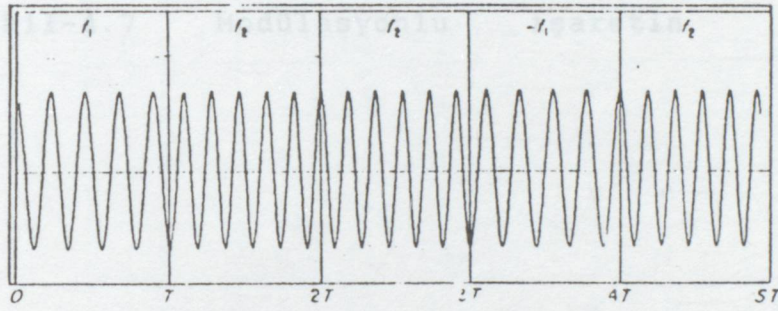


Şekil-4.4 Tasarlanan katlamalı kodlayıcı devre şeması.

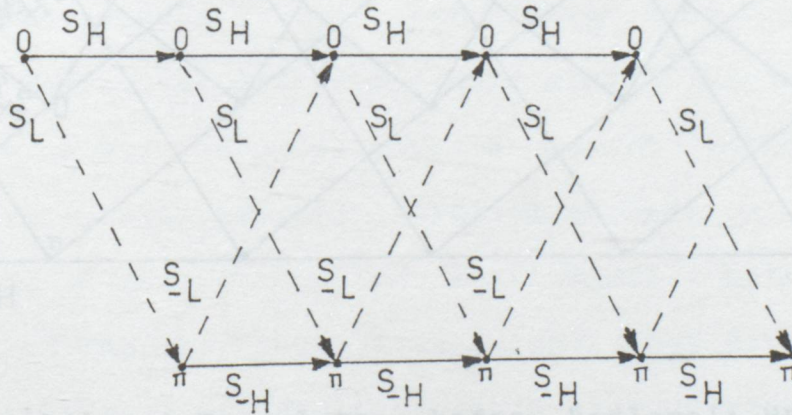
HFKA modülasyonlu işaret için bir örnek, şekil-4.5 ' de verilmiştir. Bu şekilden görüldüğü gibi, HFKA modülasyonunun temel özelliği, değişik periyotlar içinde farklı frekanslar ve sürekli faz göstermesidir. Bir aralıktan diğerine geçiş, sinüsoidal işaretin tepe noktalarında gerçekleşir.

Şekil 4.6 ' da ise, HFKA tekniğinin kafes durum gösterilimi verilmektedir. Her bir giriş bilgi biti için, çıkış işaretleri s_H , s_{-H} , s_L veya s_{-L} olacaktır. Göste-

rilimde düğümler, bit değişim noktalarını göstermektedir. Her bir düğümde modüle edilmiş işaretlerin pozitif tepe noktalarını 0, negatif tepe noktalarını da π göstermektedir. Kesiksiz çizgiler 0, kesikli çizgiler ise 1 gönderildiğini ifade etmektedir. Modülasyonlu işaretin vektörel gösterilimi de şekil-4.7' de görüldüğü gibidir.

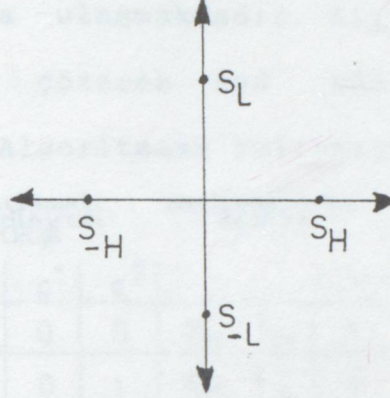


Şekil-4.5 HFKA modülasyonlu bir işaret örneği.

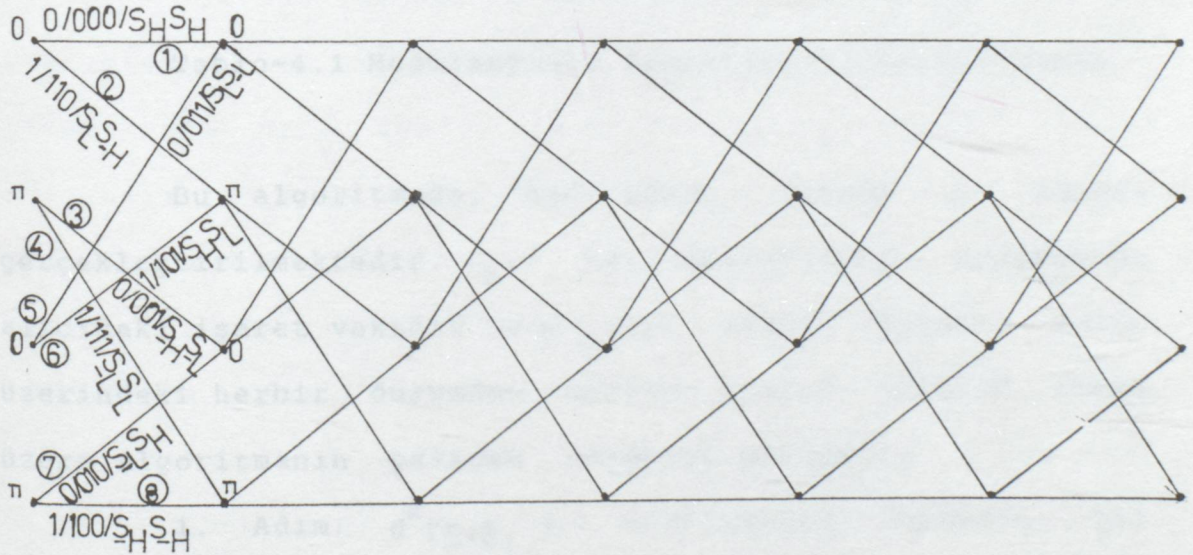


Şekil-4.6 HFKA kafes durum gösterilimi.

Bu çalışmada kullanılan, HFKA kafes kodlayıcı yapısı şekil 4.8 'de gösterildiği gibi tasarlanmıştır.



Şekil-4.7 Modülasyonlu işaretin vektörel gösterilimi.



Şekil-4.8 Tasarlanan kafes kodlamalı HFKA kodlayıcı yapısı.

olarak Modülasyonlu işaretler, tablo-4.1' de görüldüğü gibi oluşmaktadır. Bu işaretler, kanalda bozulmalara uğrayarak alıcı tarafa ulaşmaktadır. Alıcıda, alınan modülasyonlu işaretleri çözerek kod sözcükleri oluşturmak amacıyla Viterbi Algoritması kullanılmaktadır.

Katlamlı Kodlayıcı Çıktısı			Modülasyonlu İşaret		İşaret Sembolü			
v^1	c^1	c^2						
0	0	0	s_H	s_H	1	0	1	0
0	0	1	s_H	s_L	-1	0	0	-1
0	1	0	s_L	s_H	0	-1	1	0
0	1	1	s_L	s_L	0	1	0	-1
1	0	0	s_H	s_H	1	0	-1	0
1	0	1	s_H	s_L	1	0	0	1
1	1	0	s_L	s_H	0	1	-1	0
1	1	1	s_L	s_L	0	-1	0	1

Tablo-4.1 Modülasyonlu işaretlerin oluşturulması.

Bu algoritmada, kod çözme işlemi L adımda gerçekleştirilmektedir. $\underline{r}$: her işaretleşme aralığında alıcıdaki işaret vektörü ve $\underline{s}_i$: her aralık boyunca kafes üzerindeki her bir durumdan ayrılan işaret vektörü olmak üzere algoritmanın çalışma adımları şöyledir:

1. Adım: $d^2(\underline{r}, \underline{s}_i)$ uzaklıkları, kafesin ilk aralığındaki her bir düğüm için bulunur. Her varış düğümü için $\underline{r}$, iki $\underline{s}_i$ 'den hangisine yakınsa; o uzaklık ve o $\underline{s}_i$ 'nin kalkış düğümünün hangisi olduğunu kalan durum bilgisi

olarak saklar.

2. Adım : $d^2(r, s_i)$ uzaklığı, ikinci aralık için de bulunur. Her varış düğümü için, $d^2(r, s_i) + o$ düğüme gelen düğüm için bir önceki bulunan uzaklık hesaplanır. İki s_i 'den hangisi r 'ye yakınsa; ona ait toplam uzaklık ve o s_i nin kalkış düğümü saklanır.

3. Adım : 2. adımda belirtilen işlem, L. aralık da dahil olmak üzere tekrarlanır.

4. Adım : Son aralıkta, birikmiş uzaklığı en küçük olan düğüm belirlenir ve o düğüm için kalan durum bilgisine göre karar verilir. Bu işlem en başa gidinceye kadar devam eder. Sonuçta karar verilecek kod sözcüğü belirlenir[19].

Viterbi kod çözücüsü çıkışında altı bitlik kod sözcükleri elde edilir. Bu kod sözcüklerinin hatalı olup olmadıkları hata sezici C kodunun yapısına uygun olarak araştırılır. Eğer kod sözcüğünün eşlik bitleri, kodun yapısına uygun eşlik bitleri değilse, alınan kod sözcüğünün hatalı olduğuna karar verilmektedir. Eğer alınan bir kod sözcüğünün hatalı olduğuna karar verilmişse, gürültüsüz bir geri besleme kanalı üzerinden onaylamama yolunda bir bilgi gönderilerek vericinin o kod sözcüğünü yeniden iletmesini ister. Hatasız olduğuna karar verilmişse, geri besleme kanalı üzerinden onaylama bilgisi gönderilir ve vericinin sıradaki kod sözcüğünü iletmeye devam etmesi

sağlanmış olur. Hatasız olduğuna karar verilen kod sözcükleri artık kullanıcıya iletilebilir.

4.2 HFKA OYİ DİZGESİNİN ANALİTİK YOLLA HATA SINIRLARININ BELİRLENMESİ

Tasarlanan dizgenin başarımını inceleyebilmek için, dizgenin hata sınırlarını belirlemek gerekir. $R=1/2$ oranlı HFKA kafes kodlayıcısının geliştirilmiş aktarım işlevi Zehavi, Wolf [20] yöntemiyle elde edilebilmektedir. Bu yöntemde göre, toplamsal beyaz Gauss gürültüsü ile modellenen kanallar için bit hata olasılığı üst sınırı, (4.10) 'daki gibidir.

$$P_b \leq \frac{1}{n-1} Q(\sqrt{d_s^2 E_s / 2N_0}) \cdot \exp(d_s^2 E_s / 4N_0) \left. \frac{\partial T(W, L, I)}{\partial I} \right|_{\substack{L=2 \\ I=1 \\ W=\exp(-E_s / 4N_0)}}^{1-n} \quad (4.10)$$

Bu bağıntı dizge için E_b ' ye göre normalize edilirse,

$$P_b \leq Q(\sqrt{d_s^2 E_b / 2N_0}) \cdot \exp(d_s^2 E_b / 4N_0) \left. \frac{\partial T(W, L, I)}{\partial I} \right|_{\substack{L=2 \\ I=1 \\ W=\exp(-E_b / 4N_0)}}^{1-n} \quad (4.11)$$

Bu bağıntılarda kullanılan $Q(.)$ işlevi,

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-y^2/2} dy \quad (4.12)$$

şeklinde tanımlıdır. Bu bağıntıları, tasarlanan dizgeye uygulayabilmek amacıyla Zehavi-Wolf [20] tekniğinden

yararlanarak aktarım işlevi çıkarılmaktadır.

Tasarlanan dizgenin aktarım işlevini belirlemek üzere, bir düğümden ayrılan iki yol için tablo-2' de görülen hata tablosu oluşturulmalıdır.

Bu tabloda, e bir dala ilişkin hata dizilerini, y katlamalı kodlayıcının olurlu çıkışlarını, $d^2(y, y \oplus e)$ ise, olurlu doğru ve hatalı diziler arasındaki ilişkin Öklid uzaklığını göstermektedir. Bu uzaklıklar, yol kazançlarını bulmak amacıyla W gibi bir yapay değişkenin kuvveti olarak kullanılmaktadır. Buna göre, tablo-2 yardımıyla oluşturulan hata durum diyagramı şekil-4.9' da verilmektedir.

Buradan tüm hata olaylarını içeren, kodlayıcı aktarım işlevi,

$$T(W, L, I) = \frac{8W^{16} \cdot L^3 \cdot I - 16W^{24} \cdot L^4 \cdot I^2 + 16W^{16} \cdot L^4 \cdot I^2}{1 - 4W^8 \cdot L^2 \cdot I - 2W^8 \cdot L \cdot I + 8W^{16} \cdot L^3 \cdot I^2 - 8W^8 \cdot L^3 \cdot I^2} \quad (4.13)$$

biçiminde bulunur. $L=1/2$ ve $I=1$ için,

$$T(W) = \frac{2W^{16} - W^{24}}{1 - 3W^8 + W^{16}} \quad (4.14)$$

yazılır.

$T(W)$, $E_b = 2E_s = 2$ 'ye göre normalize edilirse,

$$T(W) = \frac{2W^8 - W^{12}}{1 - 3W^4 + W^8} \quad (4.15)$$

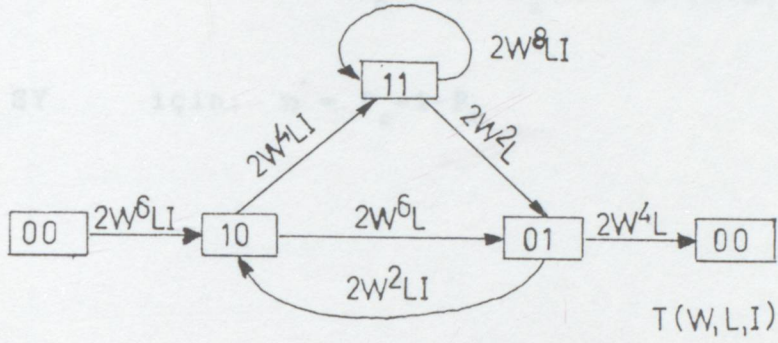
bulunur. (4.14)' de görülen aktarım işlevinin payı, paydasına oranlandığında elde edilen bölümün ilk teriminin

kuvveti, d_{serbest}^2 değerini vermektedir.

$T(W)$ yardımıyla, hata üst sınırları çeşitli E_b/N_o oranları için (4.11) ve (4.13)' den hesaplanabilir. E_b/N_o değerlerine karşılık hata olasılığının değişimi şekil-4.10 ' da görülmektedir.

e	y	$y \oplus e$	$d(y, y \oplus e)$	Wd^2
0 0 0	0 0 0	0 0 0	0	2
0 0 0	1 1 0	1 1 0	0	
0 0 1	0 0 0	0 0 1	6	$2W^6$
0 0 1	1 1 0	1 1 1	6	
0 1 0	0 0 0	0 1 0	2	$2W^2$
0 1 0	1 1 0	1 0 0	2	
0 1 1	0 0 0	0 1 1	4	$2W^4$
0 1 1	1 1 0	1 0 1	4	
1 0 0	0 0 0	1 0 0	8	$2W^8$
1 0 0	1 1 0	0 1 0	8	
1 0 1	0 0 0	1 0 1	2	$2W^2$
1 0 1	1 1 0	0 1 1	2	
1 1 0	0 0 0	1 1 0	6	$2W^6$
1 1 0	1 1 0	0 0 0	6	
1 1 1	0 0 0	1 1 1	4	$2W^4$
1 1 1	1 1 0	0 0 1	4	

Tablo-4.2 Aktarım işlevi için hata tablosu.



Şekil-4.9 Hata durum diyagramı.

Tasarlanan dizgenin kuramsal çözümlemeleri yapılırken çeşitli E_b/N_0 değerlerine göre (4.11) bağıntısından yararlanılarak, bu orana karşı düşecek bit hata olasılığına göre iş çıkarma yetenekleri belirlenmiştir. Bir kod sözcüğünün hatasız alınma olasılığı,

$$P_C = 1 - P_B \quad (4.16)$$

olarak belirlenmiştir. İş çıkarma yeteneği de , en yüksek değerine normalize edilmiş olarak ifade edilmiştir:

$$\eta' = \frac{\eta}{\eta_{\text{on büyük}}} \quad (4.17)$$

Herbir temel OYİ yapısı için iş çıkarma yeteneği bağıntıları, (2.7), (2.9), (2.11), (4.16) ve (4.17) 'den yararlanılarak;

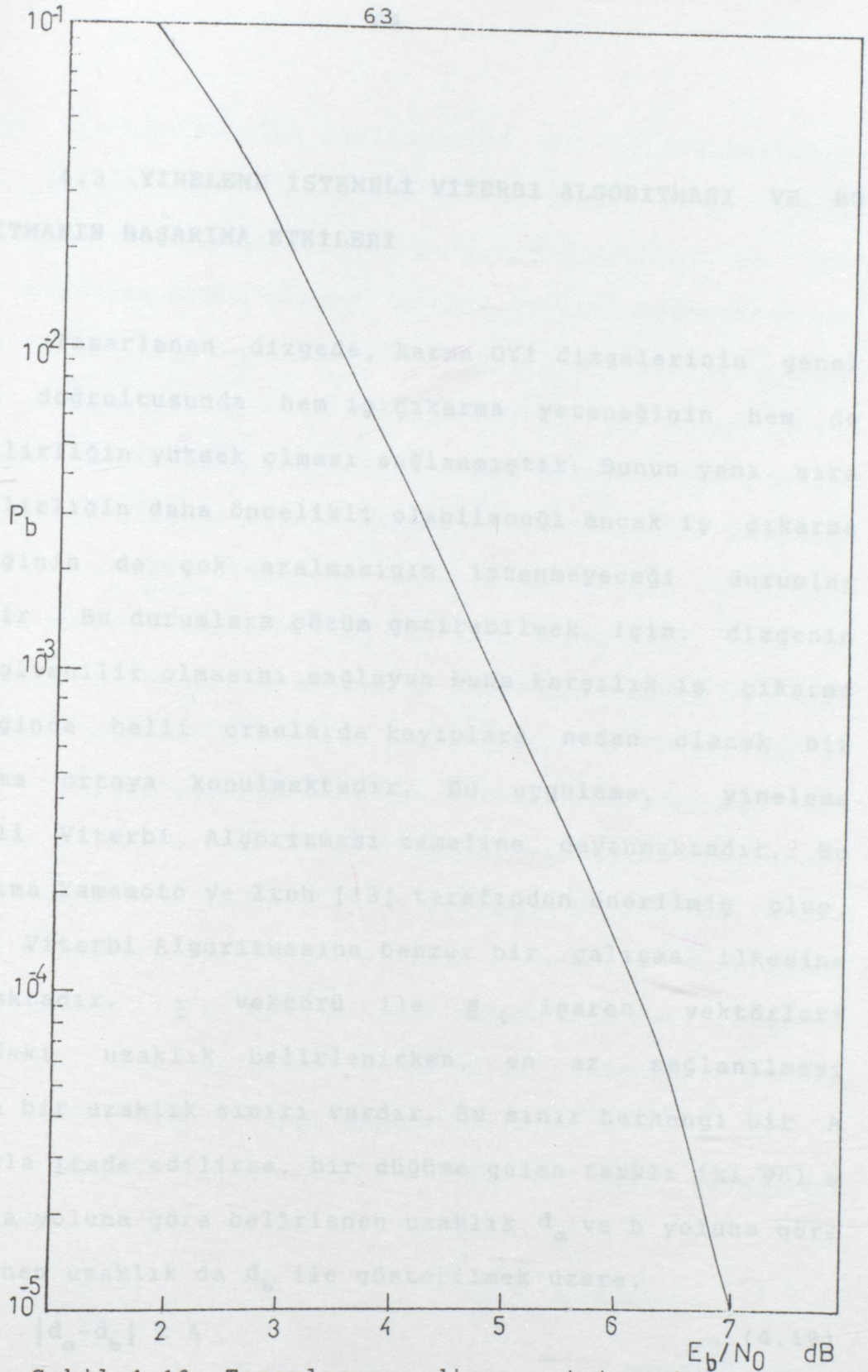
DB için: $\eta' = P_c = 1 - P_b$

N-AGD için: $\eta' = \frac{P_c}{P_c + N(1 - P_c)} = \frac{1 - P_b}{1 - (1 - N)P_b} \quad (4.18)$

SY için: $\eta' = P_c = 1 - P_b$

Şekil-4.10 Tasarlanan düzgeç için bit hata

oranının E_b/N_0 değeriyle göre değişimi.



Şekil-4.10 Tasarlanan dizge için bit hata

olasılığının E_b/N_0 değerine göre değişimi.

4.3 YİNELEME İSTEMELİ VİTERBİ ALGORİTMASI VE BU ALGORİTMANIN BAŞARIMA ETKİLERİ

Tasarlanan dizgede, karma OYİ dizgelerinin genel hedefi doğrultusunda hem iş çıkarma yeteneğinin hem de güvenilirliğin yüksek olması sağlanmıştır. Bunun yanı sıra güvenilirliğin daha öncelikli olabileceği ancak iş çıkarma yeteneğinin de çok azalmasının istenmeyeceği durumlar olabilir. Bu durumlara çözüm getirebilmek için, dizgenin daha güvenilir olmasını sağlayan buna karşılık iş çıkarma yeteneğinde belli oranlarda kayıplara neden olacak bir uygulama ortaya konulmaktadır. Bu uygulama, yineleme istemeli Viterbi Algoritması temeline dayanmaktadır. Bu algoritma Yamamoto ve Itoh [13] tarafından önerilmiş olup, normal Viterbi Algoritmasına benzer bir çalışma ilkesine dayanmaktadır. $\underline{r}$ vektörü ile $\underline{s}_i$ işaret vektörleri arasındaki uzaklık belirlenirken, en az sağlanması gereken bir uzaklık sınırı vardır. Bu sınır herhangi bir A sayısı ile ifade edilirse, bir düğüme gelen farklı iki yol a ve b, a yoluna göre belirlenen uzaklık d_a ve b yoluna göre belirlenen uzaklık d_b ile gösterilmek üzere,

$$|d_a - d_b| \geq A \quad (4.19)$$

koşulu sağlanmak koşuluyla herhangi $\underline{s}_i$ kabul edilebilecektir. Eğer koşul sağlanmıyorsa, kalan durum

bilgisi olarak koşulun sağlanmadığı da belirtilmelidir. Normal Viterbi Algoritmasında olduğu gibi, kabul edilen uzaklık da saklanmalıdır. Daha sonraki adımlar da yine (4.19) koşuluna uygun olarak, normal Viterbi Algoritmasında olduğu gibi uygulanmalıdır. Herhangi bir düğümde (4.19) koşulu sağlanıyorsa; aynı düğümün bir önceki kalan durum bilgisine bakılarak, yeni kalan durum bilgisi bir önceki kalan durum bilgisi olarak kaydedilir. Bu işlem, herhangi bir adımda bütün düğümlerde kalan durum bilgileri olarak (4.19) koşulunun sağlanmadığı bilgisi oluşuncaya veya L. adım sonuna gelinceye kadar devam edecektir. Eğer herhangi bir adımda, bütün düğümlerde (4.19) koşulunun sağlanmadığı bilgisi oluştuysa, işlemlere son verilir ve herhangi bir kod çözme işlemi oluşmadan ve herhangi bir hata incelemesi yapmadan işaret dizisinin hatalı olduğuna karar verilecek, dolayısıyla geri besleme kanalından , vericiye onaylamama bilgisi gönderilerek aynı kod sözcüğünü yeniden iletmesi sağlanacaktır. A sabitinin sıfır olması durumu, normal Viterbi Algoritmasına karşı düşmektedir.

Yineleme istemeli Viterbi için bit hata olasılığı üst sınırı,

$$P_b \leq Q\left[\sqrt{\frac{d_s^2 \cdot E_b}{2N_o}} + u\sqrt{\frac{E_b}{2N_o}}\right] \cdot \exp\left(-\frac{d_s^2 \cdot E_b}{4N_o}\right) \cdot \frac{T(W, L, I)}{\partial I} \quad (4.20)$$

$\begin{matrix} I=4 \\ L=2 \\ W=\exp(-E_b/4N_o) \end{matrix}$

olacaktır[13]. Burada u,

$$u = A.1/\sqrt{(E_b/2N_o)} \quad (4.21)$$

dir. Temelli yazıcısı olan dizgelerin iş çıkarma yetenekleri

kargılas Yineleme istemeli Viterbi Algoritması kullanan dizgeler için iş çıkarma yeteneği,

$$R_o = \frac{\text{Viterbi kod çözme işlemi uygulanmış kod sözcüğü sayısı}}{\text{kod sözcüklerinin toplam iletim sayısı}} \quad (4.22)$$

şeklinde tanımlıdır. Bu bağıntı benzetim çalışmaları için yaklaşıklıkla,

$$R_o \simeq \frac{R}{1 + P_x \cdot L} \quad (4.23)$$

şeklinde verilebilir. Bu yaklaşım, $P_x L \ll 1$ için geçerlidir. L ; bilindiği gibi kafesteki adım sayısı, R ; kod oranı ve P_x ise ,

$$P_x \leq Q\left[\sqrt{\frac{d_s^2 \cdot E_b}{2N_o}} - u\sqrt{\frac{E_b}{2N_o}}\right] \cdot \exp\left(-\frac{d_s^2 \cdot E_b}{4N_o}\right) \cdot T(W, L, I) \quad (4.24)$$

$\begin{matrix} I=1 \\ L=2 \\ W=\exp(-E_b/4N_o) \end{matrix}$

şeklinde tanımlıdır. (4.20) bağıntısına dikkat edilecek olursa, A' 'nın sıfır olması durumunda bu bağıntının, (4.11) bağıntısına özdeş olduğu görülecektir.

Şekil-4.11' da $K=4$ ötelemeli yazıcıya sahip, $E_b/N_o=1$ dB/kod simgesi olan bir yineleme istemeli Viterbi kod çözücü için farklı u katsayıları için iş çıkarma yeteneği gözlenmektedir.

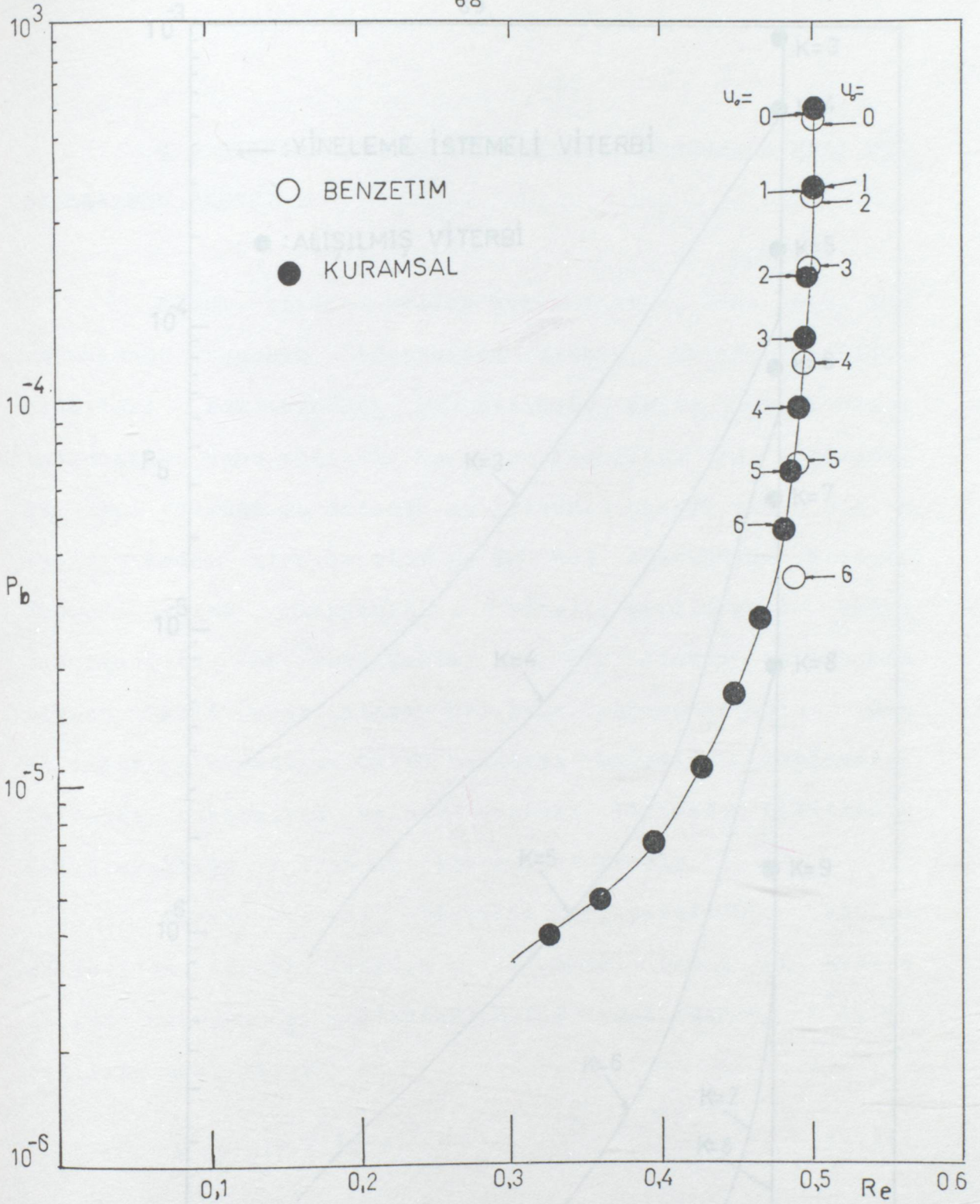
Şekil-4.12' de ise $E_b/N_o=1$ dB/kod simgesi olan

yineleme istemeli Viterbi kod çözücü için farklı sayıda ötelemeli yazıcısı olan dizgelerin iş çıkarma yetenekleri karşılaştırılmaktadır. Dikkat edilecek olursa herbir yineleme istemeli dizge için elde edilen eğrinin, alışılmış Viterbi kodçözücüsünün iş çıkarma yeteneğine teğetsel olarak yaklaştığı görülebilir.



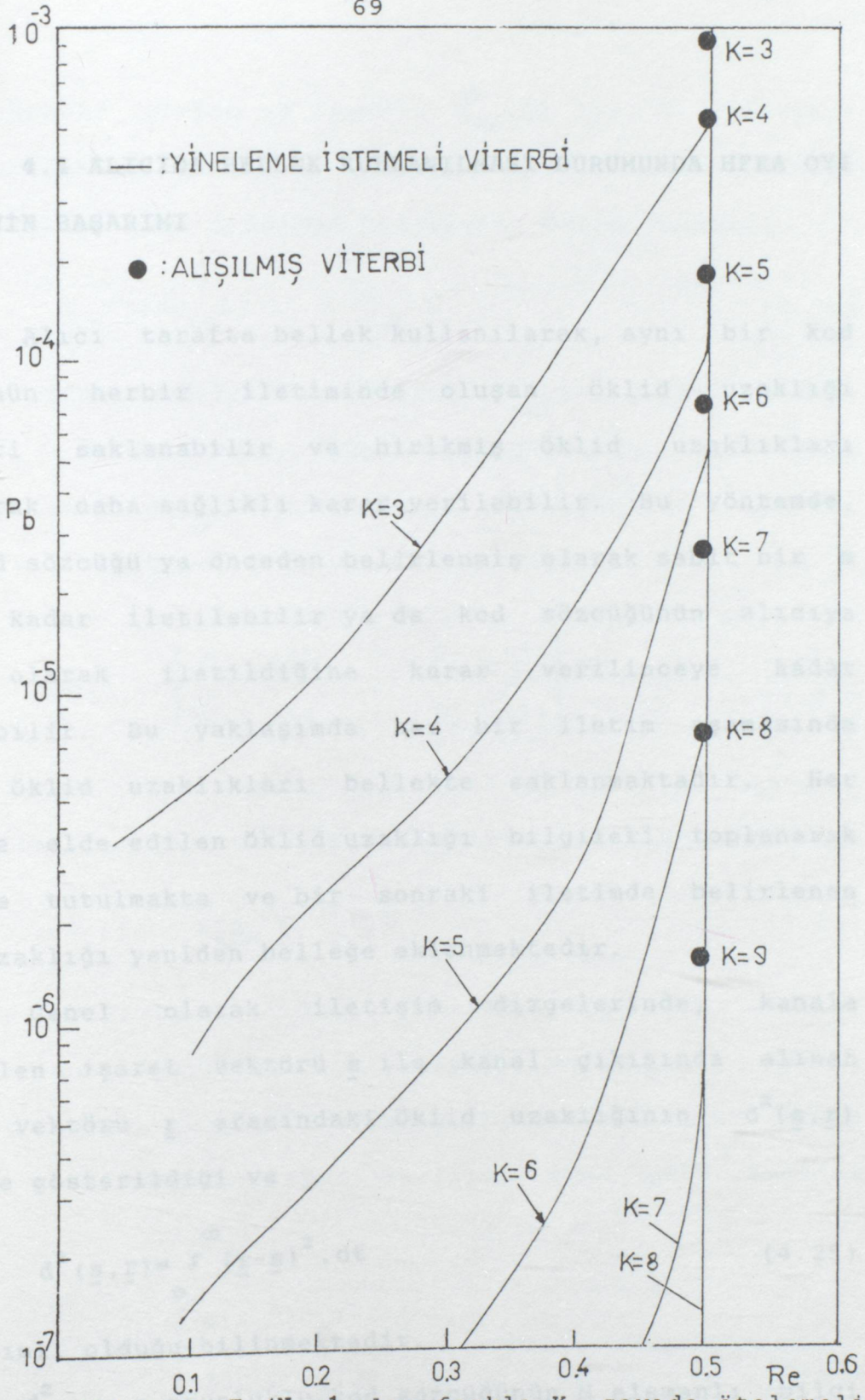
Şekil-4.11 $K=4$ ötelemeli yazıcıya sahip, $E_b/N_0=1$

dB/kod simgesi olan yineleme istemeli Viterbi kod çözücüsüne farklı N değerlerine göre iş çıkarma yeteneği.



Şekil-4.11 $K=4$ ötelemeli yazıcıya sahip, $E_b/N_0=1$

dB/kod simgesi olan yineleme istemeli Viterbi kod çizücünün farklı u değerlerine göre iş çıkarma yeteneği.



Şekil-4.12 $E_b/N_0=1$ olan farklı K sayıda ötelemeli

yazıcıya sahip dizgeler için iş çıkarma yeteneği.

4.4 ALICIDA BELLEK KULLANILMASI DURUMUNDA HFKA OYI DIZGESİNİN BAŞARIMI

Alıcı tarafta bellek kullanılarak, aynı bir kod sözcüğünün her bir iletiminde oluşan Öklid uzaklığı bilgileri saklanabilir ve birikmiş Öklid uzaklıkları toplanarak daha sağlıklı karar verilebilir. Bu yöntemde, bir kod sözcüğü ya önceden belirlenmiş olarak sabit bir m sayısı kadar iletilebilir ya da kod sözcüğünün alıcıya doğru olarak iletildiğine karar verilinceye kadar iletilebilir. Bu yaklaşımda her bir iletim aşamasında oluşan Öklid uzaklıkları bellekte saklanmaktadır. Her iletimde elde edilen Öklid uzaklığı bilgileri toplanarak bellekte tutulmakta ve bir sonraki iletimde belirlenen Öklid uzaklığı yeniden belleğe eklenmektedir.

Genel olarak iletişim dizgelerinde, kanala gönderilen işaret vektörü $\underline{s}$ ile kanal çıkışında alınan işaret vektörü $\underline{r}$ arasındaki Öklid uzaklığının $d^2(\underline{s}, \underline{r})$ şeklinde gösterildiği ve

$$d^2(\underline{s}, \underline{r}) = \int_0^{\infty} (\underline{r} - \underline{s})^2 \cdot dt \quad (4.25)$$

ile tanımlı olduğu bilinmektedir.

$d_{i,p}^2$, n uzunluklu kod sözcüğünün M elemanlı bilgi alfabesindeki her bir işaret için Öklid uzaklığını

göstermektedir ($1 \leq i \leq n$ ve $1 \leq p \leq M$). $d_{i,p}^2(j)$ ise, j . iletimde belirlenen Öklid uzaklığını ifade etmektedir. Burada ortaya atılan yöntemle göre j adımda belirlenen Öklid uzaklığı

$$D_{i,p}(j) = D_{i,p}(j-1) + d_{i,p}^2(j) \quad (4.26)$$

şeklinde ifade edilebilir. Burada $D_{i,p}(j-1)$, bir önceki iletimde belirlenmiş toplamsal Öklid uzaklığını göstermektedir. $D_{i,p}(0)$ uzaklığının sıfır olacağı açıktır. Bu açıklamalara göre; $D(j)$ uzaklık matrisi,

$$D(j) = \begin{bmatrix} D_{1,1}(j) & \dots & D_{1,M}(j) \\ \vdots & \ddots & \vdots \\ D_{n,1}(j) & \dots & D_{n,M}(j) \end{bmatrix} \quad (4.27)$$

olarak ifade edilmektedir. Kod sözcüğünün i . bileşeni,

$$D_i(j) = \min_{1 \leq p \leq M} D_{i,p}(j) \quad (4.28)$$

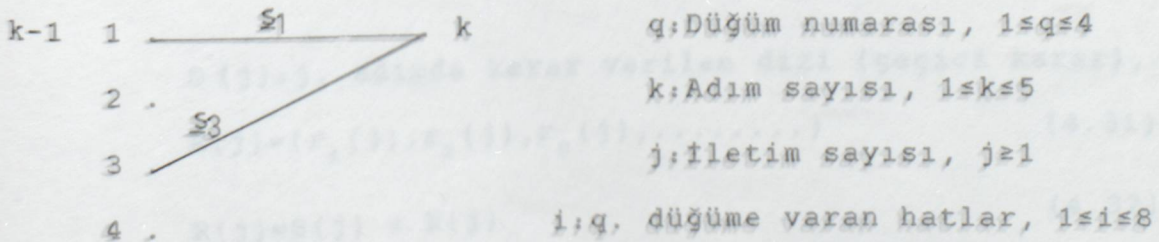
şeklinde belirlenir. Yani, alınan işaretin olası işaretlerden hangisine eşit olduğu veya eşit kabul edilebileceğine; bunlar arasından Öklid uzaklığı en küçük olanını belirleyerek karar verilir. Buna göre uzaklık matrisi,

$$D(j) = \begin{bmatrix} \min D_{1,p}(j) & \dots & \min D_{1,p}(j) \\ \vdots & \ddots & \vdots \\ \min D_{n,p}(j) & \dots & \min D_{n,p}(j) \end{bmatrix} \quad (4.29)$$

olarak gösterilebilir.

Uzaklık matrisinde karar verilen işaretlere göre alınan kod sözcüğü belirlenir. Bu kod sözcüğünün, gönderilen bir kod sözcüğü olup olmadığı incelenerek yeniden iletme veya başka bir kod sözcüğünün gönderilmesi gerektiğine karar verilir.

Bu tezde tasarlanan dizgenin kafes yapısının şekil-4.8' de gösterildiği gibi olduğu bölüm-4.1' de belirtilmişti. Kafesin yapısı, sekiz daldan oluşmaktadır. Adım sayısı ise, altı olarak belirlenmiştir. Kafesteki her bir dal, kodlayıcının ürettiği olası bilgileri ifade etmektedir. Bir adımdaki her bir dal için Öklid uzaklıkları hesaplanmaktadır. Son adımda her bir düğümde birikmiş uzaklıklara bakılarak alınan bilgi dizisine karar verilir. Karar verilen bilgi dizisinin, eşlik bitlerine bakılarak olası bir kod sözcüğü olup olmadığı belirlenir. Olası bir kod sözcüğü değilse, yeniden iletime karar verilir. Her yeniden iletim aşamasında, kafesteki her adımdaki her bir dal için Öklid uzaklığını belirlerken o dal için daha önceki iletimde belirlenmiş olan Öklid uzaklığı eklenmektedir. Buna göre,



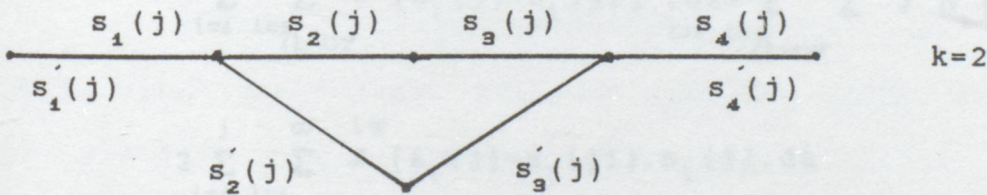
$$l_q^k(j) = \min_i \left\{ \sum_{p=1}^j [|r(p) - s_i|^2 + l_i^{k-1}(p)] \right\} \quad (4.30)$$

bağıntısı herhangi bir k. adımdaki q. düğüme gelen dallar için Öklid uzaklığını ifade etmektedir. Bu sonuca göre, Viterbi Algoritması uygulanırken, her bir iletimdeki her bir adımda bütün düğümlerde oluşan dal uzunlukları bellekte saklanmalıdır. İlk adımda her bir düğüme Öklid uzaklığı $l_i^0(p)$; sıfır olarak alınmaktadır.

BAŞARIMIN İNCELENMESİ

j: bir dizinin ard arda gönderilme sayısı,

k: alınan dizinin çözüldüğü dizi ile çözülmesi gereken (gönderilen) dizi arasındaki farklı işaret sayısı,



$R(j)$: j. adımda alınan dizi,

$S(j)$: j. " gönderilen dizi,

$S'(j)$: j. adımda karar verilen dizi (geçici karar),

$$R(j) = (r_1(j), r_2(j), r_3(j), \dots) \quad (4.31)$$

$$R(j) = S(j) + N(j) \quad (4.32)$$

olup burada $N(j)$ gürültüyü ifade etmektedir. $R(j)$ ' nin $S'(j)$ ye uzaklığı:

$$d^2\{R(j)-S'(j)\} = D'(j) \quad (4.33)$$

$$D'(j) = \sum_{i=1}^j \int_0^{\infty} \{R(j)-S'(j)\}^2 \cdot dt$$

$$= \sum_{i=1}^j \sum_{l=1}^{\infty} \int_{(l-1)T}^{lT} [r_l^2(j) - 2r_l(j)s_l'(j) + s_l'^2(j)] \cdot dt \quad (4.34)$$

$$r_l(j) = s_l(j) + n_l(j) \quad (4.35)$$

olduğuna göre,

$$D'(j) = \sum_{i=1}^j \sum_{l=1}^{\infty} \int_{(l-1)T}^{lT} \{s_l^2(j) + 2s_l(j)n_l(j) + n_l^2(j) - 2n_l(j)s_l'(j)$$

$$- 2s_l(j)s_l'(j) + s_l'^2(j)\} \cdot dt$$

$$= \sum_{i=1}^j \sum_{l=1}^{\infty} \int_{(l-1)T}^{lT} [s_l(j) - s_l'(j)]^2 \cdot dt + \sum_{i=1}^j \sum_{l=1}^{\infty} \int_{(l-1)T}^{lT} n_l^2(j) \cdot dt +$$

$$2 \sum_{i=1}^j \sum_{l=1}^{\infty} \int_{(l-1)T}^{lT} [s_l(j) - s_l'(j)] \cdot n_l(j) \cdot dt \quad (4.36)$$

olarak bulunur. $R(j)$ ' nin $S(j)$ ' ye uzaklığı:

$$d^2\{R(j)-S(j)\} = D(j) \quad (4.37)$$

olup,

$$D(j) = \sum_{i=1}^j \sum_{l=1}^{\infty} \int_{(l-1)T}^{lT} n_l^2(j).dt \quad (4.38)$$

şeklindedir. Buna göre, $\Delta(j) = D(j) - D'(j) > 0$ ise $R(j)$, $S'(j)$ 'ye $S(j)$ den daha yakın olur ve hata oluşur.

$$\begin{aligned} \Delta(j) = & - \sum_{i=1}^j \sum_{l=1}^{\infty} \int_{(l-1)T}^{lT} [s_l(j) - s'_l(j)]^2 .dt - \\ & 2 \sum_{i=1}^j \sum_{l=1}^{\infty} \int_{(l-1)T}^{lT} [s_l(j) - s'_l(j)] n_l(j).dt \end{aligned} \quad (4.39)$$

$l=1,2,3,\dots,\infty$ iken $S(j)$ ile $S'(j)$ nin k bileşeni farklı olacağından "pairwise error probability" (j gönderme için),

$$P_k(j) = \Pr(\Delta(j) > 0) \quad (4.40)$$

olarak belirlenir.

k farklı bileşen arasındaki karesel Öklid uzaklığı $-d_k^2 E$ olmak üzere j kez gönderme sonunda $-j d_k^2 E$ dir. Dolaşısıyla $\Delta(j)$ nin 1. terimi $j d_k^2 E$ dir [11], [19].

İkinci terim ele alınırsa, ortalaması 0 ' dir.

Varyansı:

$$n_o = -2 \sum_{l=1}^{\infty} \int_{(l-1)T}^{lT} [s_l(j) - s'_l(j)] n_l(j).dt \quad (4.41)$$

$$E[n_o^2] = 4 \sum_{l=1}^{\infty} \int_{(l-1)T}^{lT} [s_l(j) - s'_l(j)] . E[n_l^2(j)].dt$$

$$4N_o \sum_{l=1}^{\infty} \int_{(l-1)T}^{lT} [s_l(j) - s'_l(j)]^2 dt = 2N_o d_k^2 E \quad (4.42)$$

olup, j. iletimden sonra $2jN_o d_k^2 E$ ye yükselir.

$\Delta(j)$, ortalaması $-jd_k^2 E$, varyansı $2jN_o d_k^2 E$ olarak

Gauss dağılımlıdır.

$$P_k(j) = \Pr[\Delta(j) \geq 0]$$

$$= \int_0^{\infty} \frac{1}{\sqrt{2\pi(2jN_o d_k^2 E)}} \cdot \exp\{-(\alpha + jd_k^2 E)^2 / (4jN_o d_k^2 E)\} \cdot d\alpha \quad (4.43)$$

olarak belirlenir.

$$\gamma = \frac{\alpha + jd_k^2 E}{\sqrt{2jN_o d_k^2 E}} \quad (4.44)$$

dönüşümü yapılırsa,

$$P_k(j) = Q(\sqrt{j d_k^2 E / 2N_o}) = P_k \quad (4.45)$$

olarak belirlenir.

"Error event probability" üst sınırı,

$$P_E \leq \sum_{k=d}^{\infty} a_k P_k \quad (4.46)$$

burada a_k , k adımı farklı yol çiftleri sayısıdır ($T(W, L, I)$ 'nin katsayıları).

$$Q(\sqrt{x+y}) \leq \exp(-y/2) \cdot Q(\sqrt{x}) \quad (4.47)$$

ve $d_k^2 \geq d_s^2$ $x = jd_s^2 E / 2N_0$ $y = (d_k^2 - d_s^2) jE / 2N_0$ olursa,

$$Q(\sqrt{(d_s^2 + y)jE / 2N_0}) \leq \exp(-jE(d_k^2 - d_s^2) / 4N_0) \cdot Q(\sqrt{jEd_s^2 / 2N_0})$$

$$P_E \leq \sum_{k=d_s}^{\infty} a_k \cdot P_k \leq Q(\sqrt{jEd_s^2 / 2N_0}) \sum_{k=d_s}^{\infty} a_k \exp[-jE(d_k^2 - d_s^2) / 4N_0]$$

$$P_E \leq Q(\sqrt{jEd_s^2 / 2N_0}) \cdot \exp(jEd_s^2 / 4N_0) \cdot \sum_{k=d_s}^{\infty} a_k \cdot \exp(-jd_k^2 E / 4N_0) \quad (4.48)$$

olarak belirlenir.

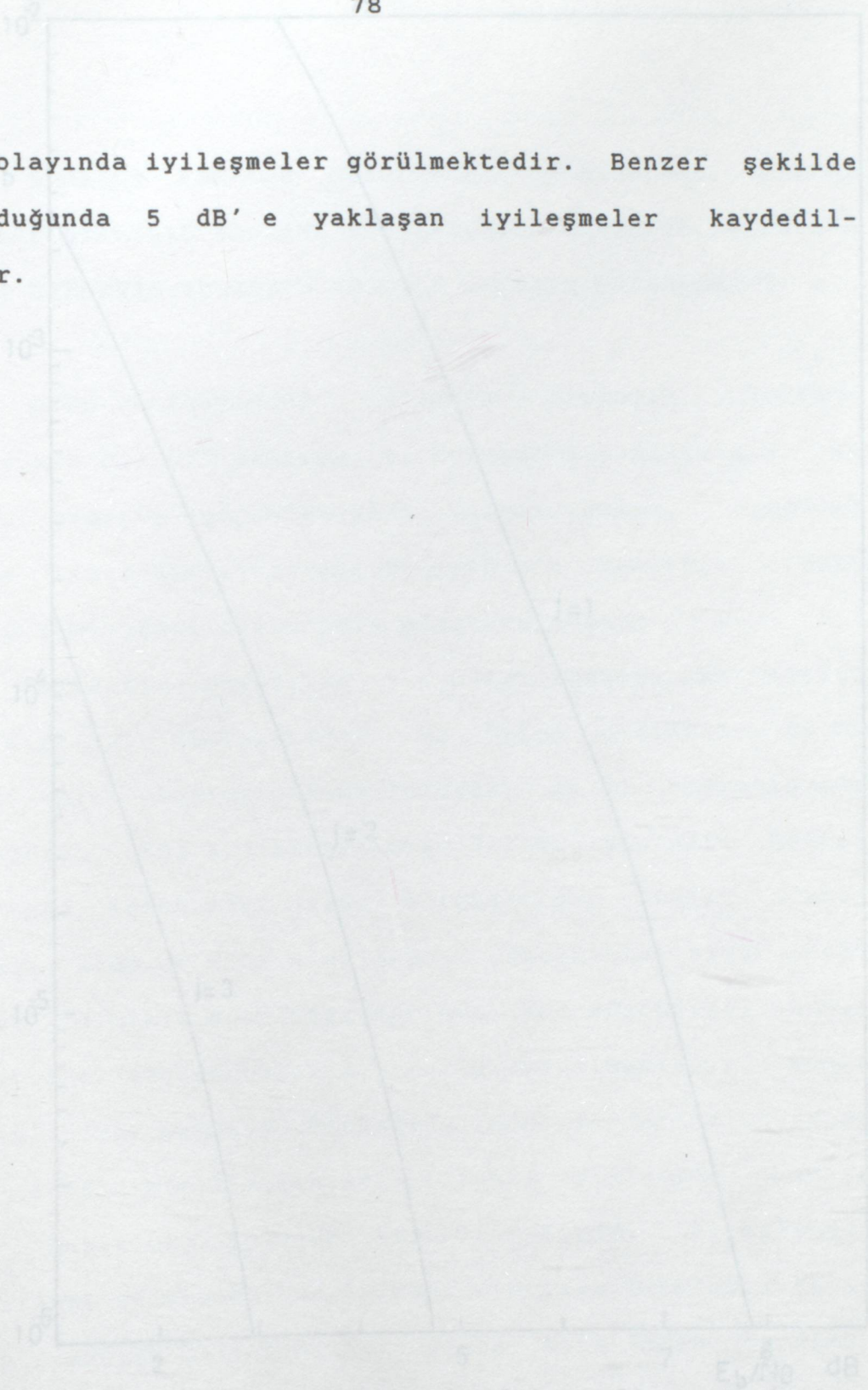
Burada belirlenen P_E ifadesinde toplam altındaki son terim

$$T(W) \Big|_{W = \exp(-jE / 4N_0)} \quad (4.49)$$

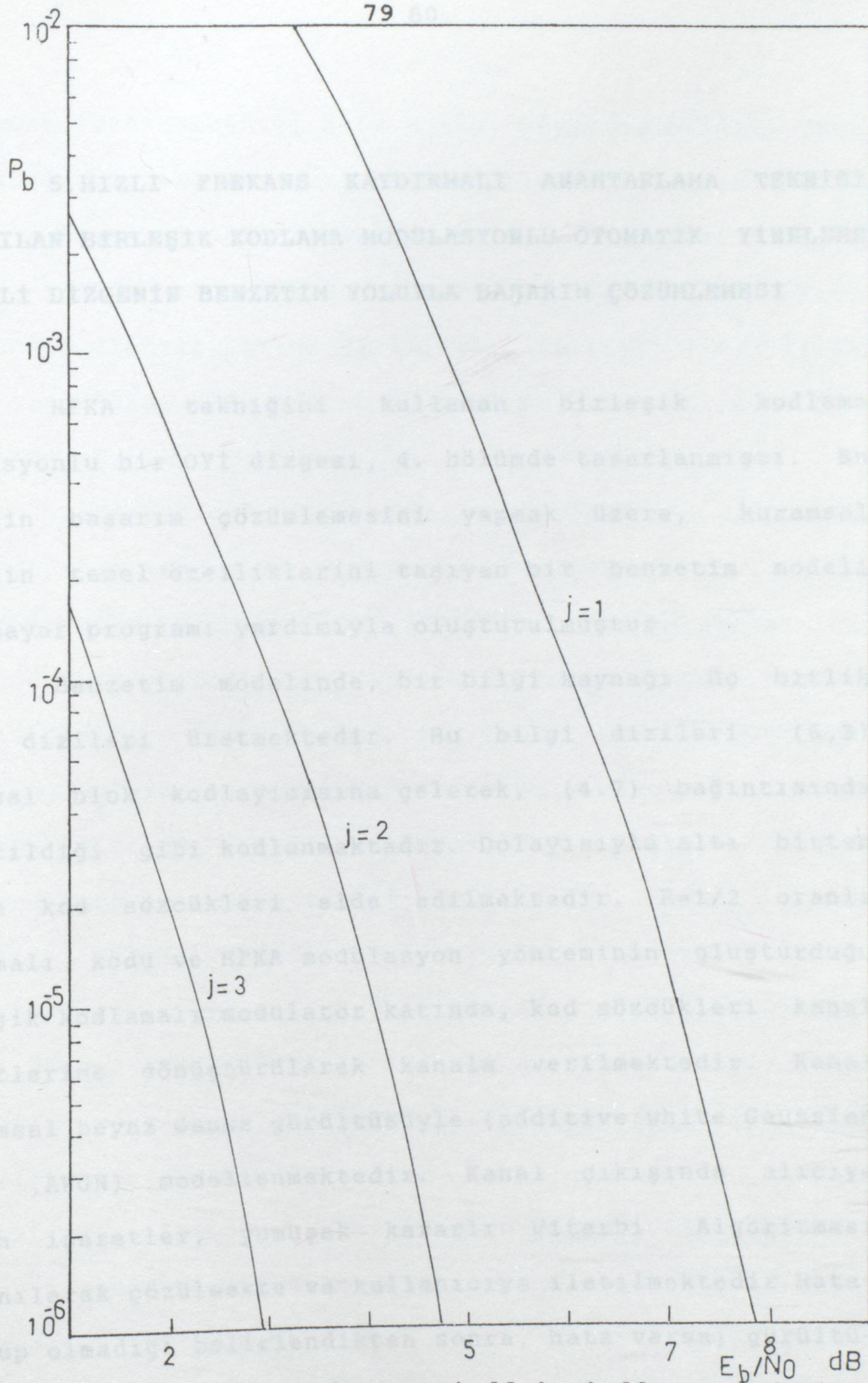
şeklinde dizgenin aktarım işlevidir.

Bölüm-4.2' de belirlenen hata ifadesiyle karşılaştırılacak olursa, bu bölümde incelenen j kez gönderme işlemi, bütün terimlere bir j çarpanı getirmektedir. Burada elde edilen sonuçlara göre; $j=1$ olması (yeniden gönderme olmaması) ve j' nin artan değerleri için hata olasılığının işaret gürültü oranına göre değişimi Şekil-4.13' de gösterilmektedir. Bu eğrilere bakarak incelenen yöntemin, dizge başarımına olumlu katkılar getirdiğini belirtmek gerekir. Bu katkı hiç de azımsanır bir boyutta değildir. $j=2$ olması durumunda bile,

3 dB dolayında iyileşmeler görülmektedir. Benzer şekilde $j=3$ olduğunda 5 dB' e yaklaşan iyileşmeler kaydedilmektedir.



Şekil-4.13 Alıcıda ara bellek kullanılarak, j kez iletim yönteminde farklı j ler için hata olasılığının işaret gücü oranına göre değişimi.



Şekil-4.13 Alıcıda ara bellek kullanan, j kez iletim yönteminde farklı j ler için hata olasılığının işaret gürültü oranına göre değişimi.

5. HIZLI FREKANS KAYDIRMALI ANAHTARLAMA TEKNİĞİ KULLANILAN BİRLEŞİK KODLAMA MODÜLASYONLU OTOMATİK YİNELEME İSTEMELİ DİZGENİN BENZETİM YOLUYLA BAŞARIM ÇÖZÜMLEMESİ

HFKA tekniğini kullanan birleşik kodlama modülasyonlu bir OFDM dizgesi, 4. bölümde tasarlanmıştır. Bu dizgenin başarımlarını yapmak üzere, kuramsal dizgenin temel özelliklerini taşıyan bir benzetim modeli bilgisayar programı yardımıyla oluşturulmuştur.

Benzetim modelinde, bir bilgi kaynağı üç bitlik bilgi dizileri üretmektedir. Bu bilgi dizileri $(6,3)$ doğrusal blok kodlayıcısına gelerek, $(4,7)$ bağıntısında belirtildiği gibi kodlanmaktadır. Dolayısıyla altı bitten oluşan kod sözcükleri elde edilmektedir. $R=1/2$ oranlı katlamalı kodu ve HFKA modülasyon yönteminin oluşturduğu birleşik kodlamalı modülatör katında, kod sözcükleri kanal işaretlerine dönüştürülerek kanala verilmektedir. Kanal toplamsal beyaz Gauss gürültüsüyle (additive white Gaussian noise, AWGN) modellenmektedir. Kanal çıkışında alıcıya ulaşan işaretler, yumuşak kararlı Viterbi Algoritması kullanılarak çözülmekte ve kullanıcıya iletilmektedir. Hatalı olup olmadığı belirlendikten sonra, hata varsa; gürültüsüz bir geri besleme kanalı yoluyla vericiye onaylamama bilgisi gönderilmekte ve aynı bilgi dizisinin yeniden gön-

derilmesi istenmektedir, hata yoksa; onaylama bilgisi vericiye gönderilerek, sıradaki bilgiyi iletmesi istenmektedir.

Kuramsal çözümlemeler yaparken çeşitli E_b/N_0 işaret gürültü oranı (İGO) değerlerine ve bunlara karşı düşen hata olasılıklarına (4.11) göre (4.16), (4.17), (4.18) bağıntılarından elde edilen iş çıkarma yeteneği değerleri kullanılmıştır.

Benzetim çalışmalarında ise, farklı İGO değerleri için yapılan denemeler yardımıyla elde edilen sonuçlardan yararlanılmıştır. Bu sonuçlar 500 'lük bilgi paketleri için elde edilmiş olduğundan, 500 bilgi paketinin alıcıya iletilebilmesi için toplam kaç kez iletim yapılması gerektiği bilgisi ve uygulamada ölçülmesi olanak dışı olan; alıcıya iletilen bilgi paketlerinden kaçının hatalı olduğunu gösteren bilgilerdir. Kullanıcıya hatalı olarak iletilen bilgi paketlerinin sayısının azlığı, dizge güvenilirliğinin göstergesidir. Benzetim çalışmalarından, 500 'lük bilgi paketlerinin 10 'ar kez denenmesiyle daha güvenilir sonuçlar elde edilmiştir. İş çıkarma yeteneği, benzetim sonuçlarına göre;

$$\eta = \frac{\text{Hatasız iletildiğine karar verilen kod sözcüğü sayısı}}{\text{Gönderilen toplam kod sözcüğü sayısı}} \quad (5.1)$$

bağıntısından hesaplanmıştır.

Şekil-5.1, şekil-5.2 ve şekil-5.3, temel OYİ dizgeleri ile tasarlanan birleşik kodlama modülasyonlu

karma OYİ dizgelerinin iş çıkarma yeteneklerini çeşitli E_b/N_0 ' a bağlı olarak vermektedir. İş çıkarma yeteneğinde temel dizgelere göre önemli yükselmeler olduğu açıkça gözlenmektedir. Şekil-5.4 farklı N değerlerine karşılık N-AGD yöntemi için belirlenmiş kuramsal sonuçları göstermektedir. E_b/N_0 gibi güvenilirlik değerlerinde

belli iyi Kuramsal iş çıkarma yeteneği belirlenirken hata olasılığı için en iyi durumu (en az hata) belirleyen,

$$P_B \cong Q(\sqrt{d_e^2 E_b / (2N_0)}) = Q(\sqrt{4E_b / N_0}) \quad (5.2)$$

ilişkisinden yararlanılmıştır. Şekil-5.5 ve şekil-5.6 ' da sırasıyla DB ve farklı N değerlerine göre N-AGD dizgeleri için benzetim sonuçları karşılaştırılmaktadır. Kuramsal sonuçlar, en küçük hata için çıkarılmış sonuçlar olduğundan benzetim eğrileri bu eğrilerin altında kalmaktadır. Ayrıca N-AGD yönteminde N değerlerinin büyük olmasının başarımı olumsuz etkilediği de gözlenmektedir.

Şekil-5.7, farklı gidiş geliş zamanı değerleri (N) ve farklı ara bellek uzunlukları (B) için SY dizgesinin iş çıkarma yeteneğinin değişimlerini, benzetim sonuçlarından göstermektedir. Bu eğrilerden de görülmektedir ki; gidiş geliş gecikmesinin küçük değerlerde olması ve ara bellek uzunluğunun ise olabildiğince büyük olması başarımı olumlu yönde etkilemektedir.

Yineleme istemeli Viterbi Algoritması kullanılan benzetim modeline ilişkin sonuçlar, şekil-5.8 ve şekil-5.9

' da DB ve N-AGD yapıları için, farklı A katsayıları kullanılarak verilmektedir. Bu benzetim sonuçları da 4. bölümde ortaya konulan yargılar doğrultusunda gerçekleşmiş ve iş çıkarma yeteneğinde A' nın artmasıyla bir miktar düşmeler olduğu gözlenmiştir. Buna karşılık, Şekil-5.10 ve Şekil-5.11' de de görüldüğü gibi güvenilirlik değerlerinde belli iyileşmeler kaydedilmiştir.

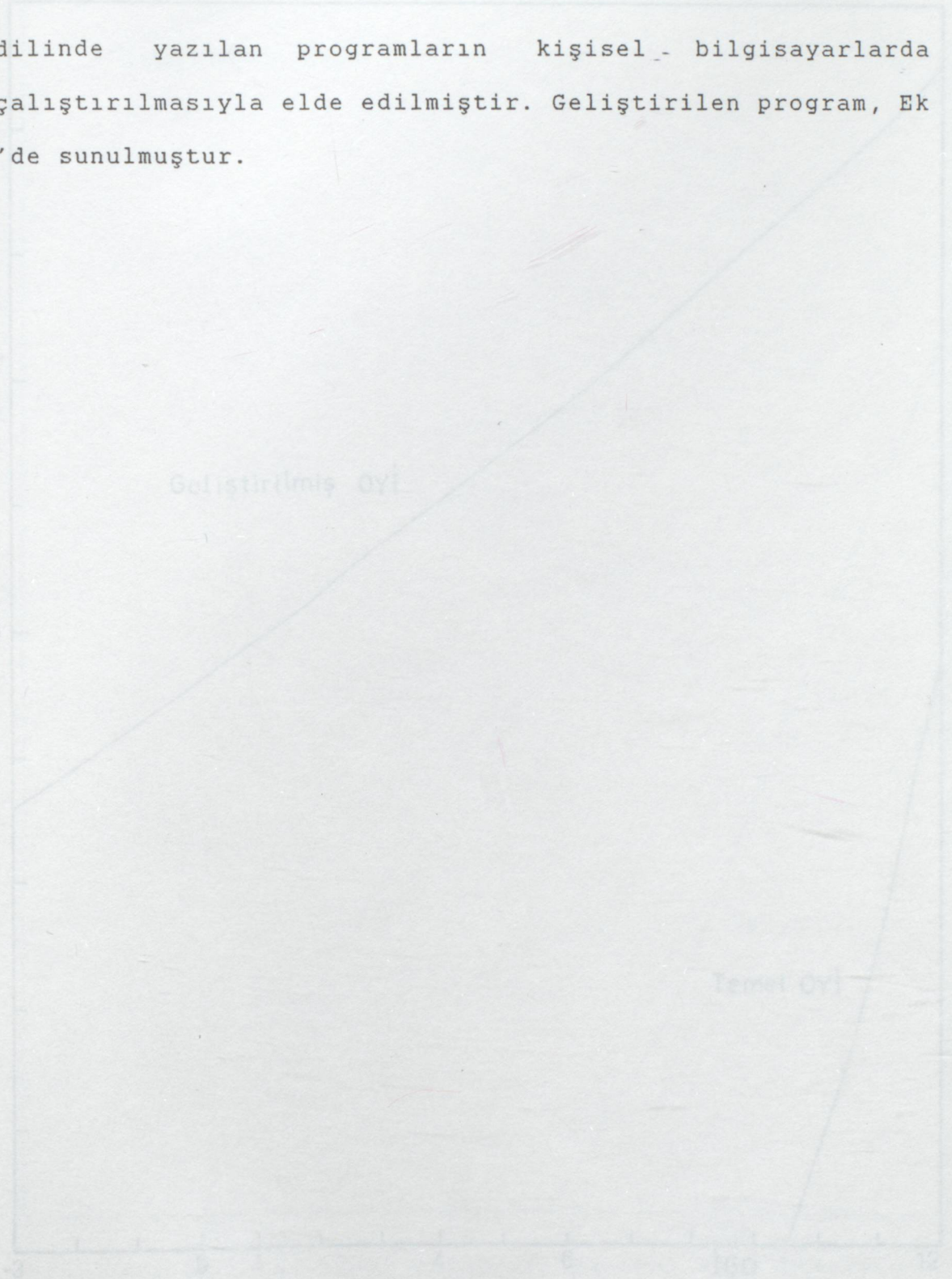
Şekil-5.12' de, tasarlanan dizge için $E_b/N_0 = 1$ dB alınarak bit hata olasılığına göre iş çıkarma yeteneği, DB OYİ yapısı için kuramsal olarak farklı u değerlerine göre çizilmiş normal Viterbi Algoritmasının vereceği sonuçla karşılaştırılmıştır. Şekil-5.13' de ise aynı koşullar için benzetim çalışmasının sonuçları verilmiştir. Her iki çalışma da DB OYİ yapısı için gerçekleştirilmiştir.

Burada, iş çıkarma yeteneklerinin belirlenmesinde (4.20), (4.21), (4.22), (4.23) ve (4.24) bağıntıları kullanılmıştır.

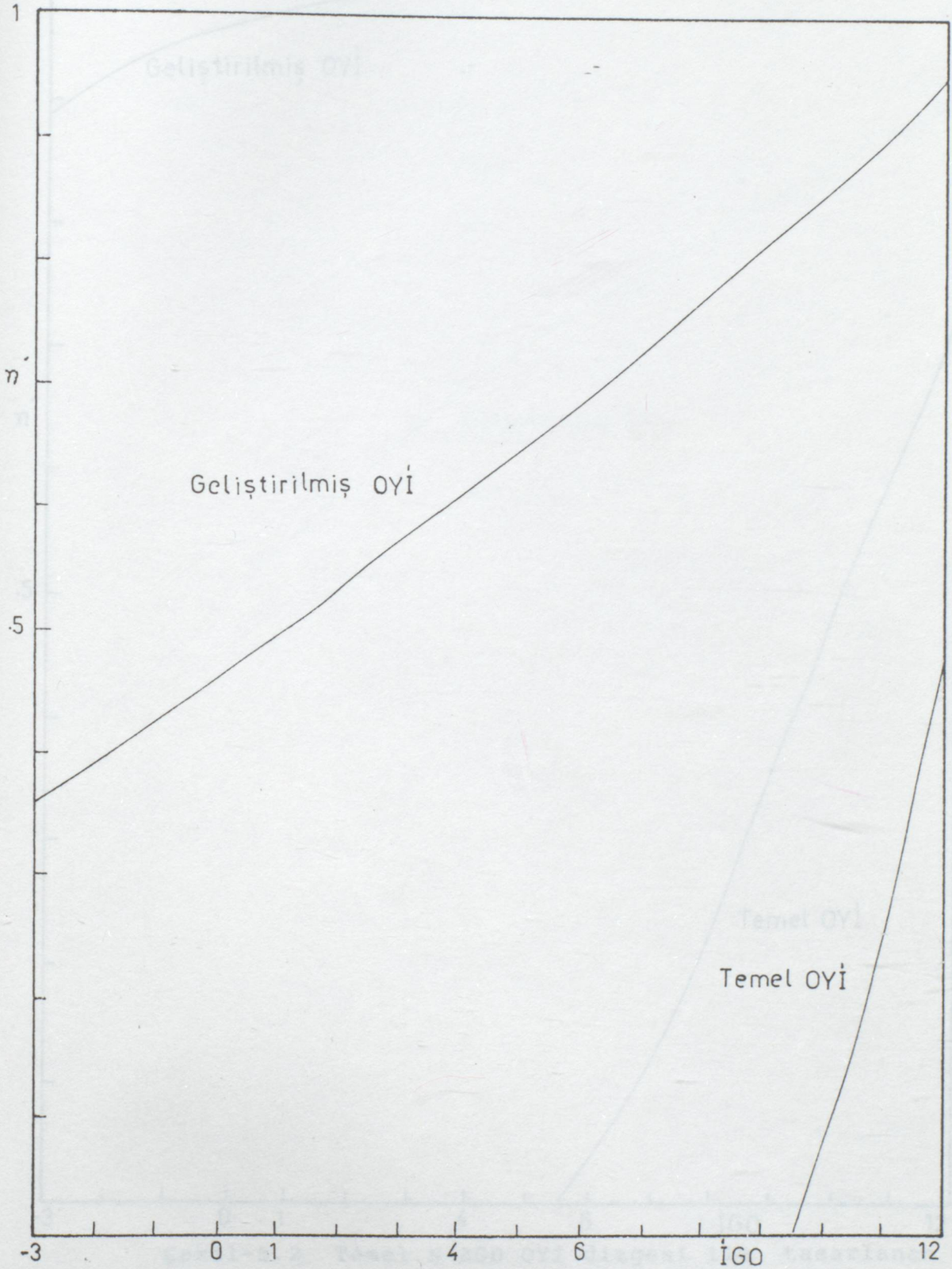
Şekil-5.14 ve Şekil-5.15' de ise alıcı tarafta ara bellek kullanılması durumunda, bir kod sözcüğünün doğru olarak alınıncaya kadar j kez yeniden iletilmesi ve her bir iletimde elde edilen öklid uzaklığı bilgilerinin saklanarak dikkate alınması yöntemi ele alınmıştır. Kurulan benzetim modelinden elde edilen sonuçlar DB OYİ ve N-AGD OYİ yapıları için ayrı ayrı görülmektedir.

Benzetim çalışmaları, Turbo Pascal programlama

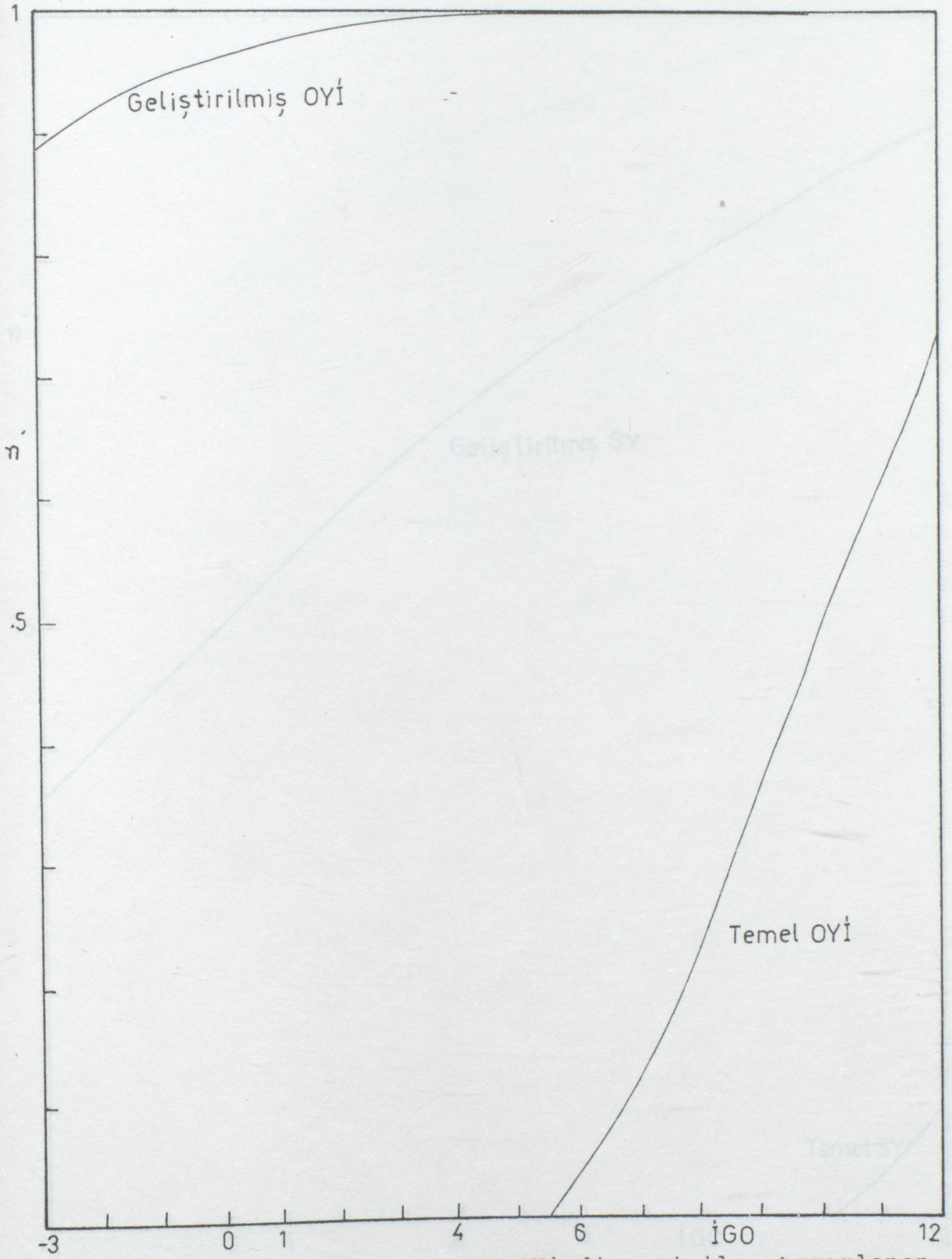
dilinde yazılan programların kişisel bilgisayarlar da geliştirilmesiyle elde edilmiştir. Geliştirilen program, Ek 'de sunulmuştur.



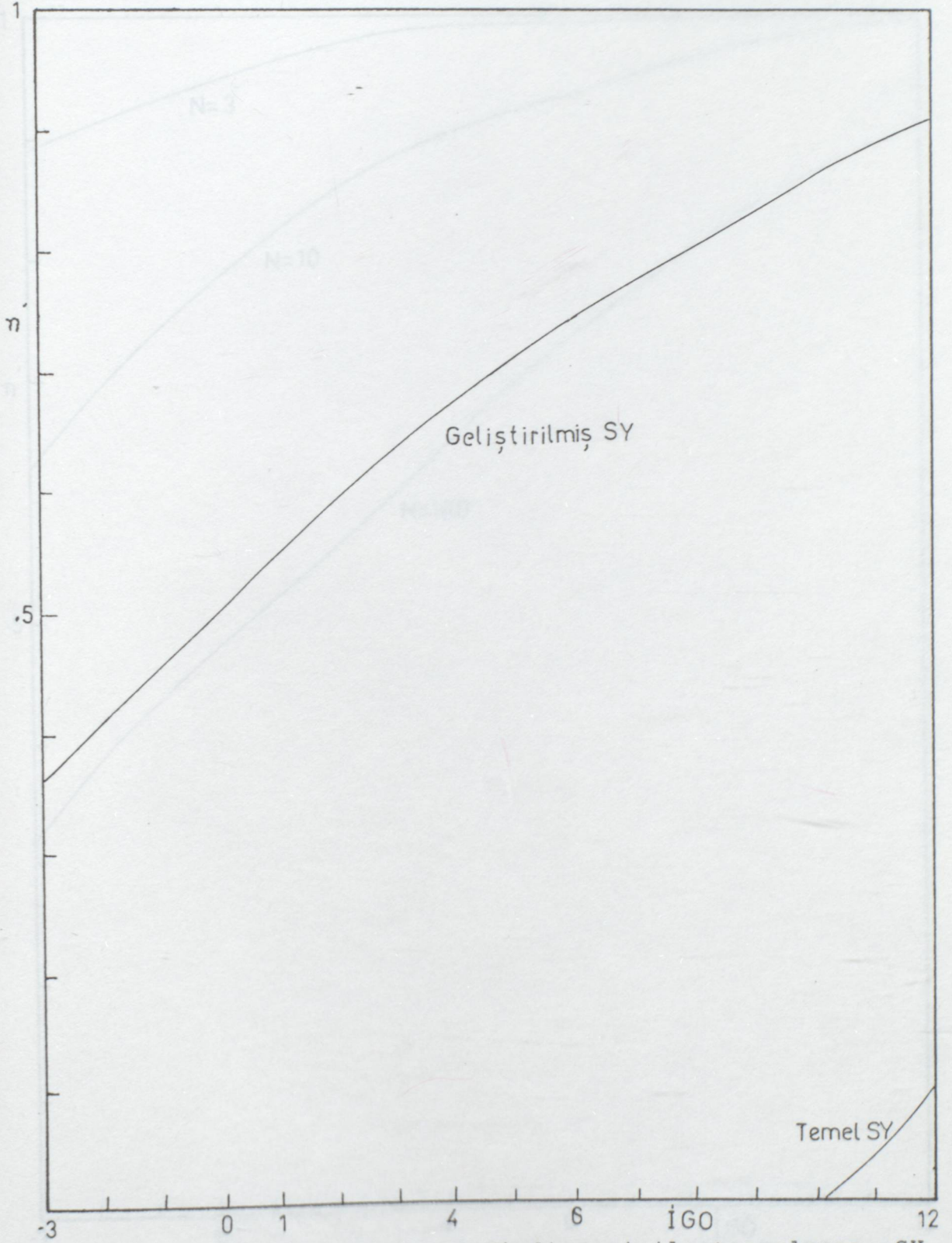
Şekil 1: Temel DB OYI dizgisi ile tanımlanan DB OYI ile geliştirilmiş DB OYI arasındaki farkın karşılaştırılması.



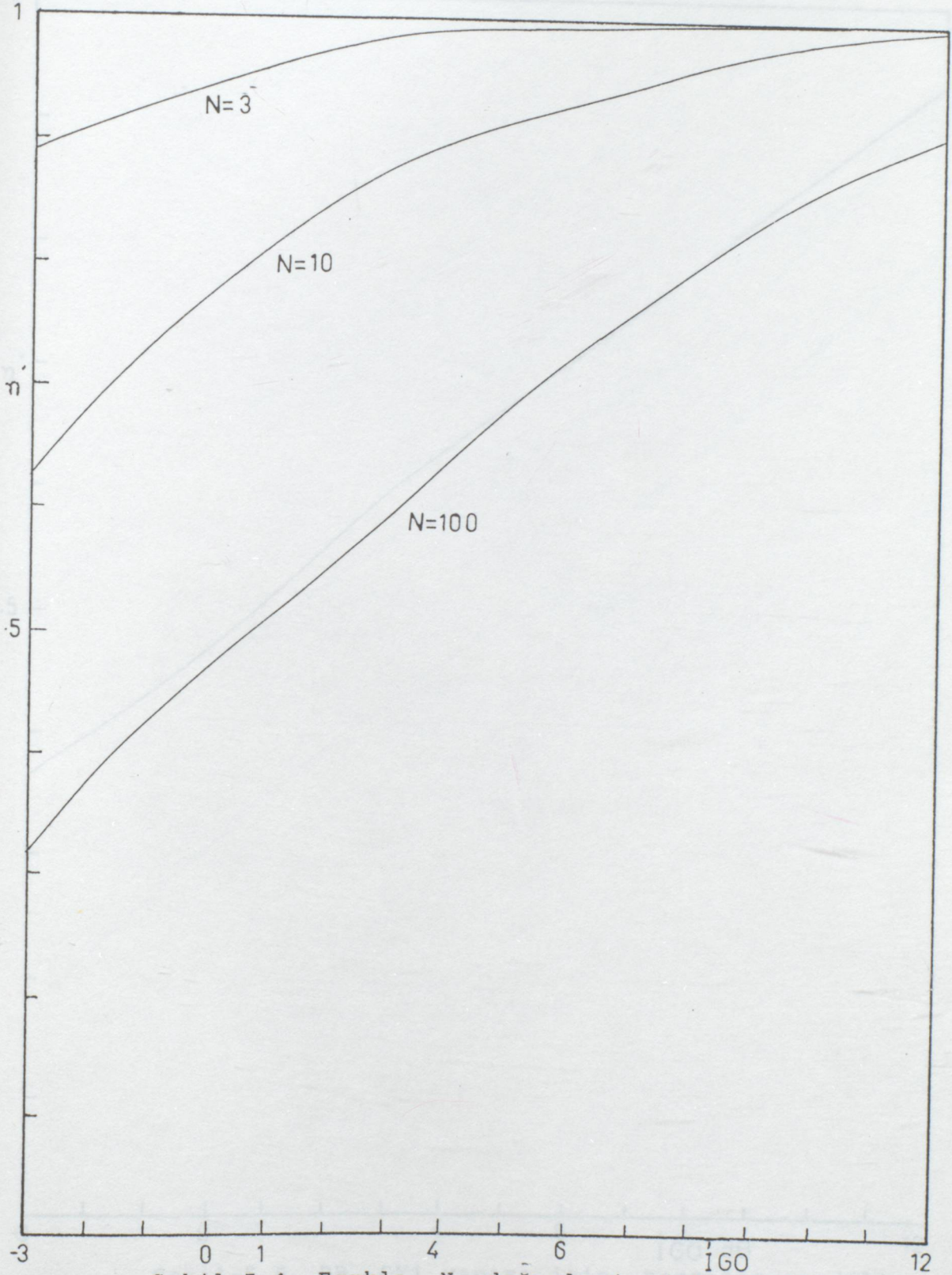
Şekil-5.1 Temel DB OYİ dizgesi ile tasarlanan DB OYİ dizgesinin iş çıkarma yeteneği değerlerinin karşılaştırılması.



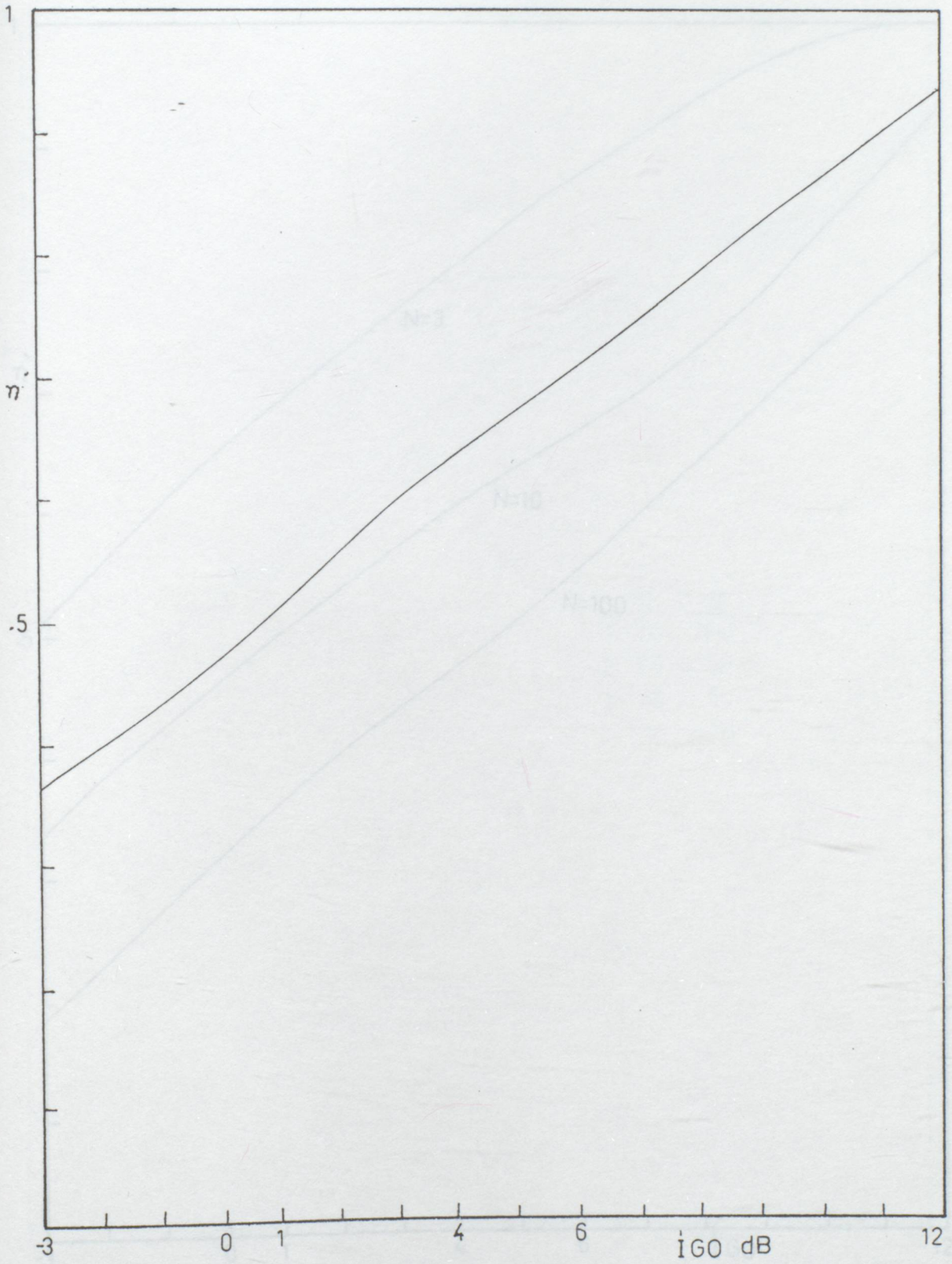
Şekil-5.2 Temel N-AGD OYİ dizgesi ile tasarlanan N-AGD OYİ dizgenin iş çıkarma yeteneği karşılaştırmaları.



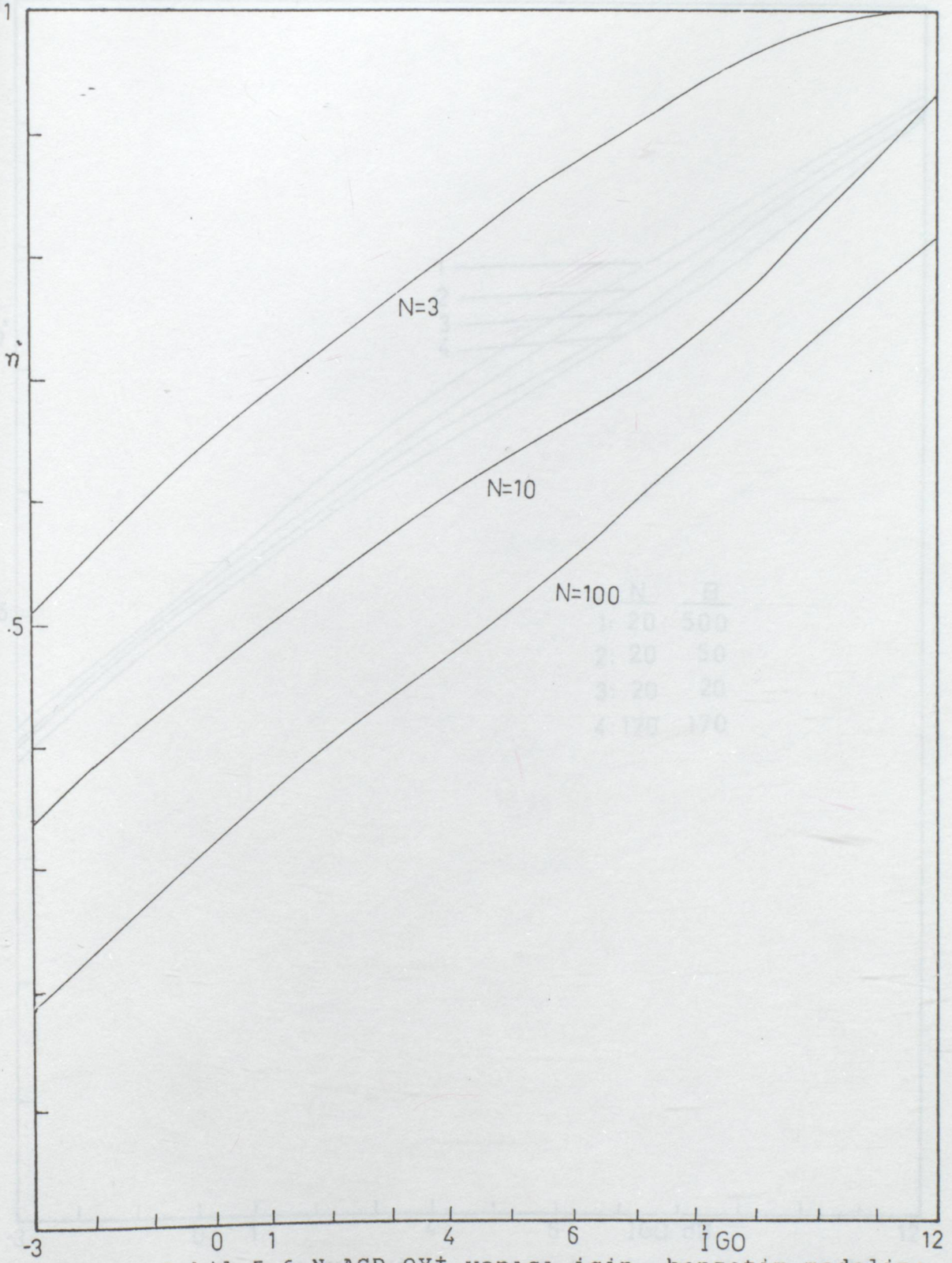
Şekil-5.3 Temel SY OYİ dizgesi ile tasarlanan SY OYİ dizgenin iş çıkarma yeteneğinin karşılaştırılması.



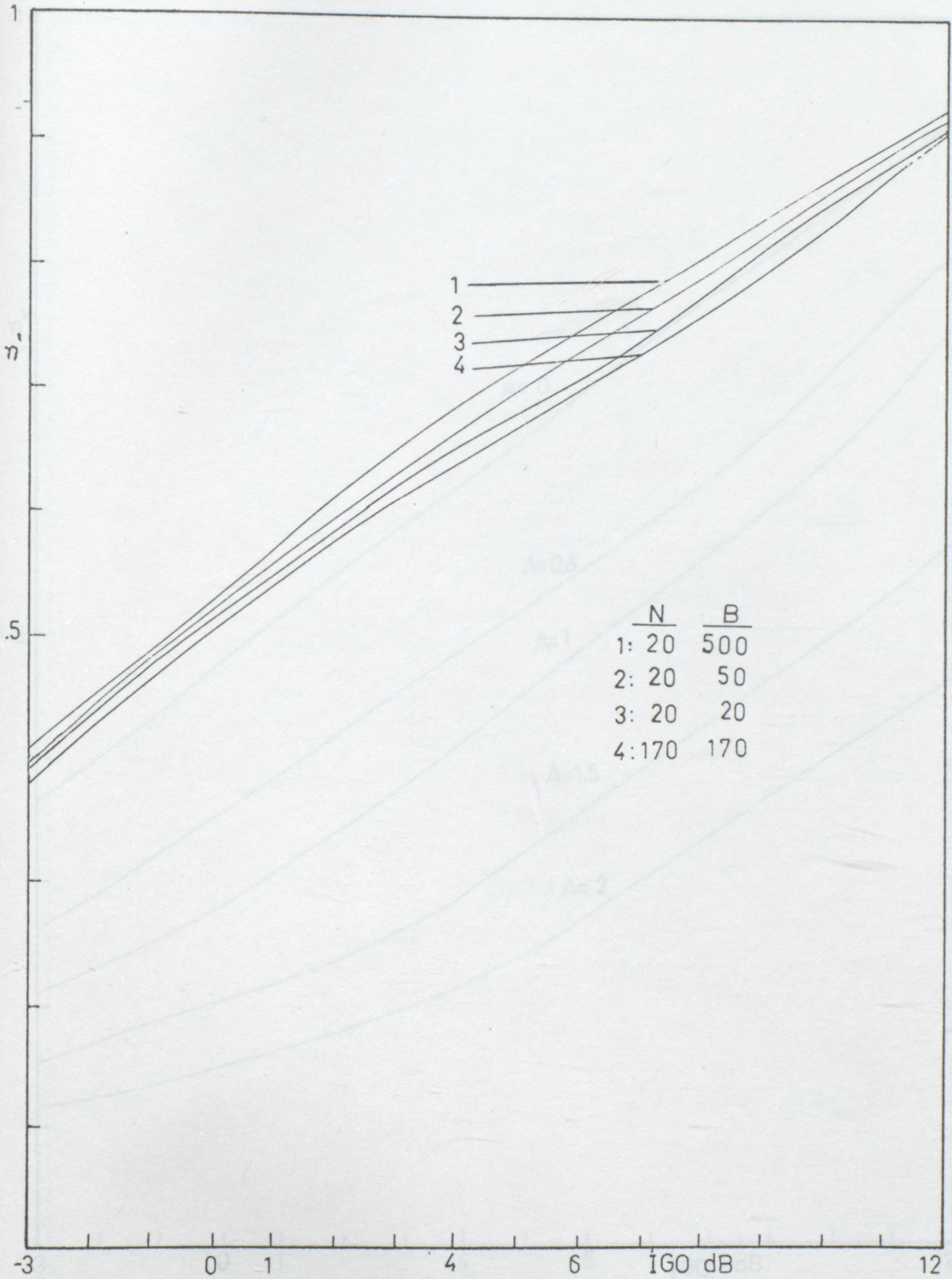
Şekil-5.4 Farklı N değerleri için N -AGD OYI dizgesinin iş çıkarma yeteneği karşılaştırmaları.



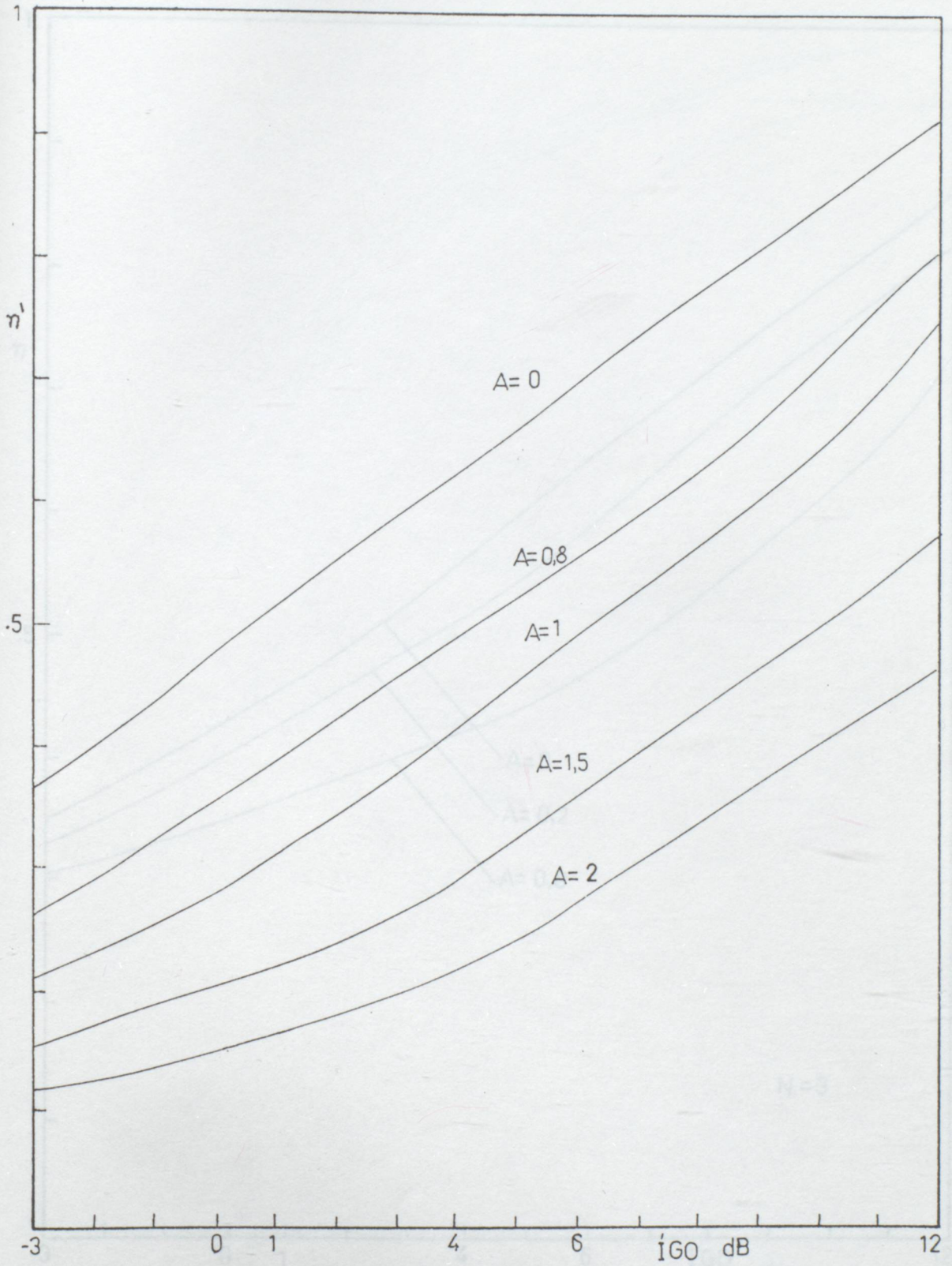
Şekil-5.5 DB OYİ yapısı için benzetim modeline göre iş çıkarma yeteneği.



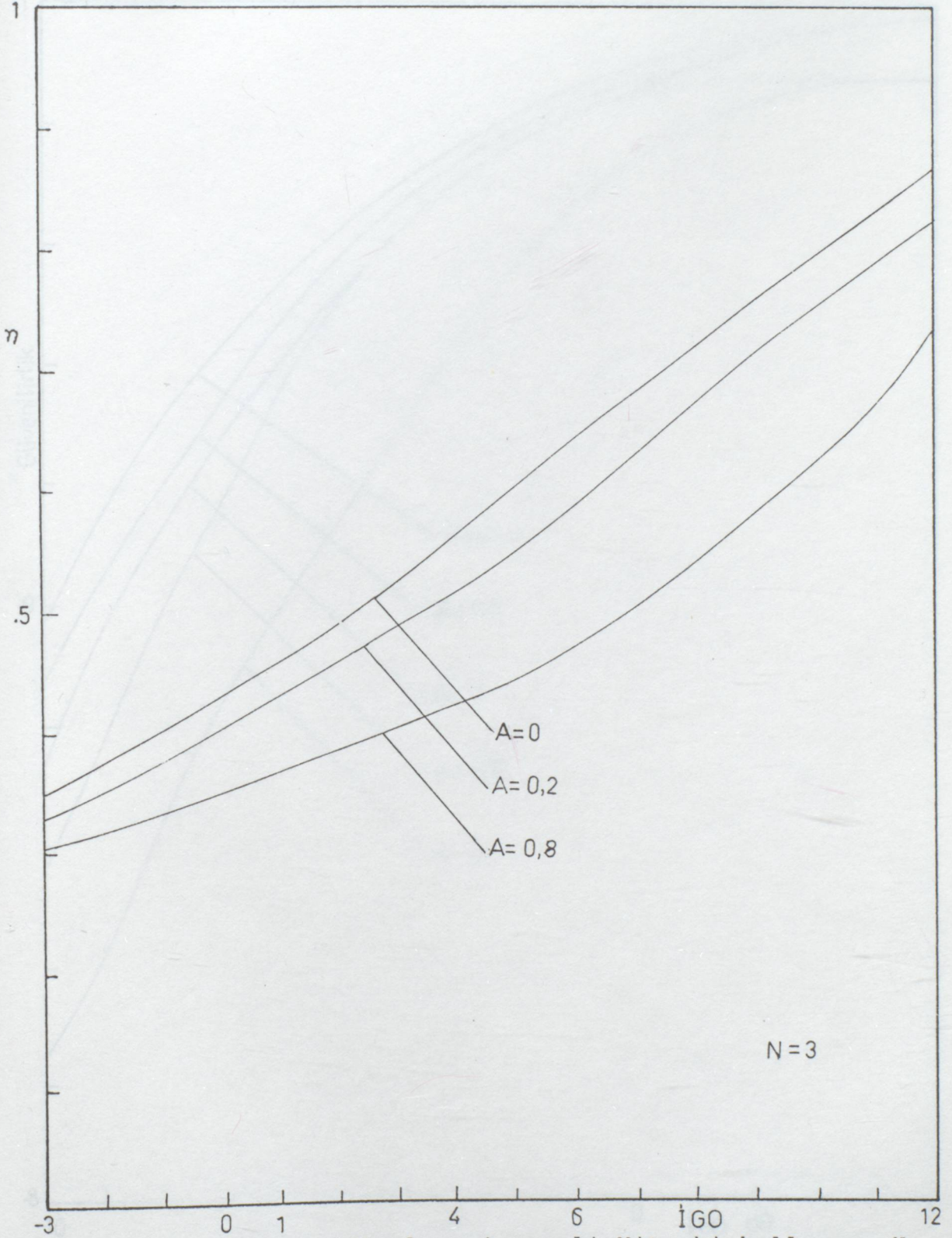
Şekil-5.6 N-AGD OYI yapısı için benzetim modeline göre iş çıkarma yeteneği.



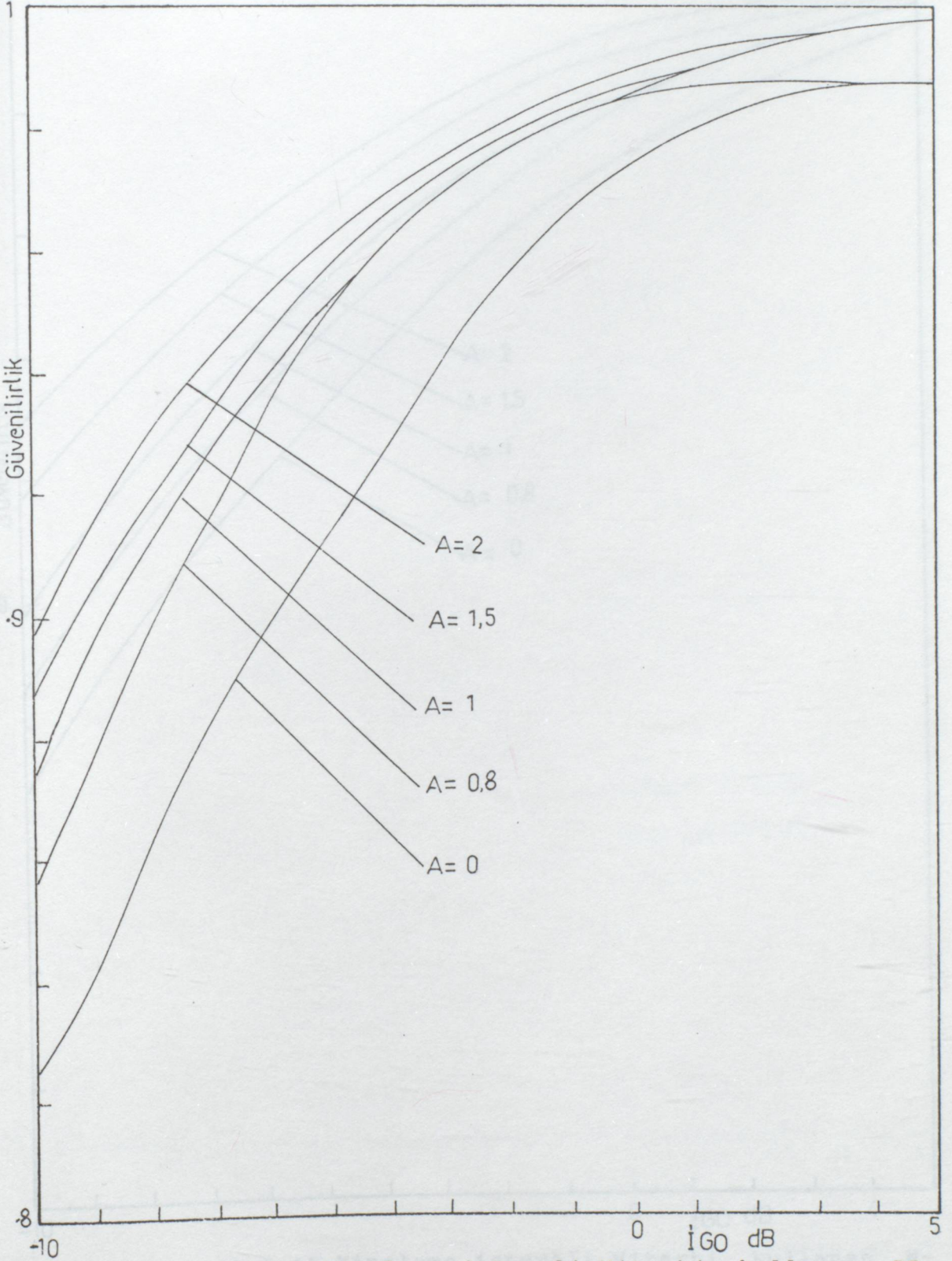
Şekil-5.7 SY OYİ yapısı için benzetim modeline göre iş çıkarma yeteneği.



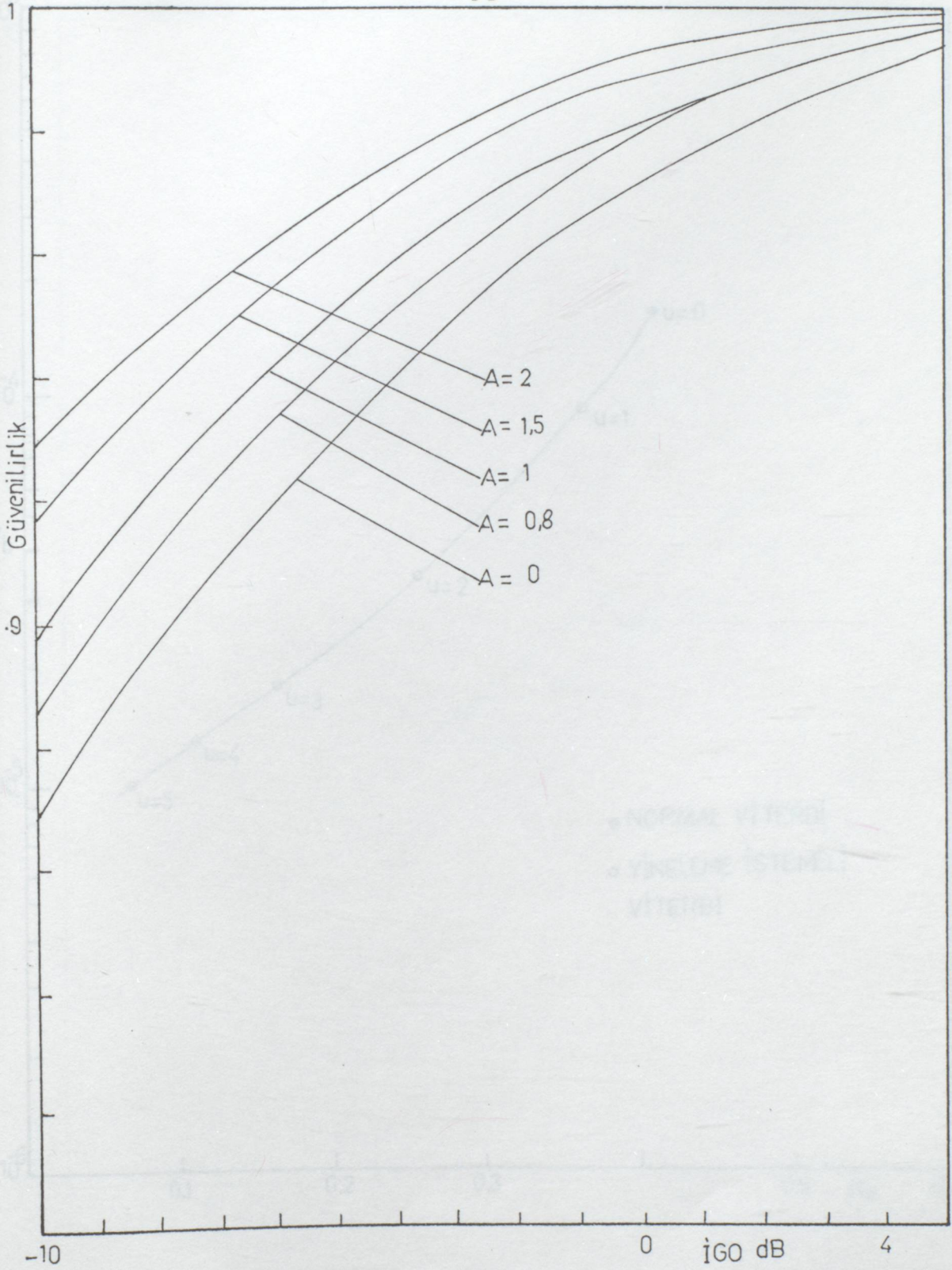
Şekil-5.8 Yineleme istemeli Viterbi kullanan DB OYİ yapısı için benzetim modeline göre farklı A değerleri için iş çıkarma yeteneği.



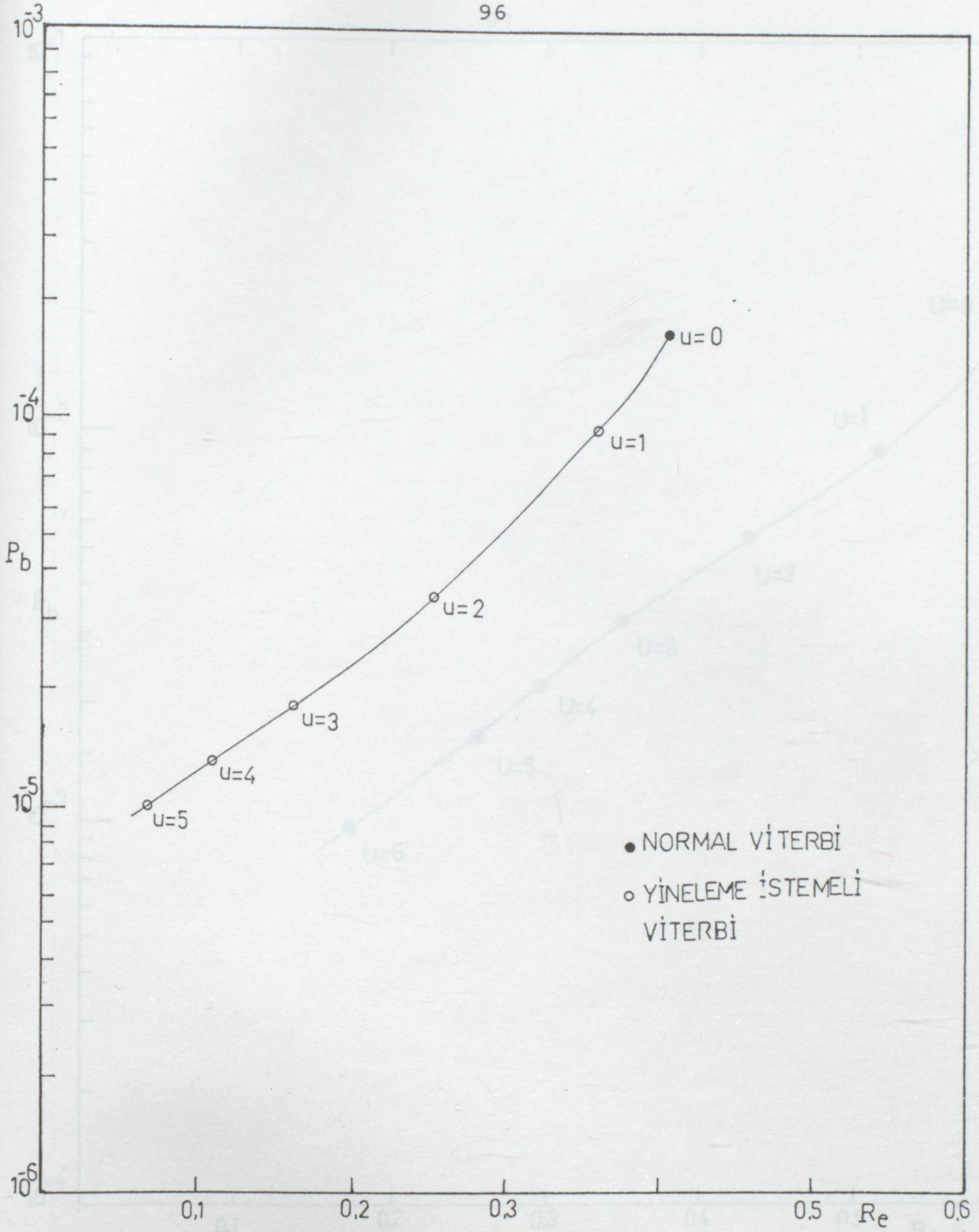
Şekil-5.9 Yineleme istemeli Viterbi kullanan N-AGD OYİ yapısı için benzetim modeline göre farklı A değerleri için iş çıkarma yeteneği.



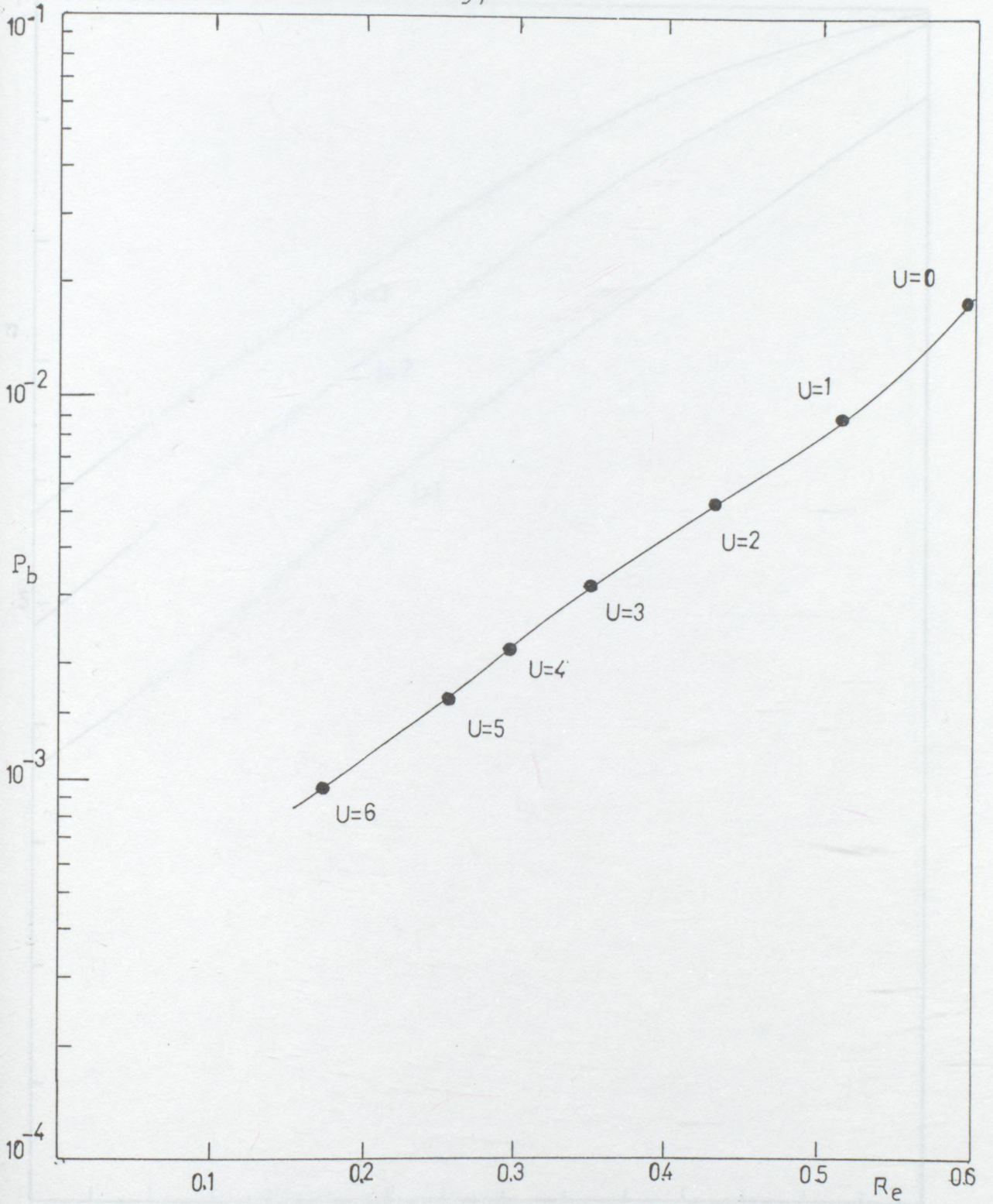
Şekil-5.10 Yineleme istemeli Viterbi kullanan DB OYİ yapısı için benzetim modeline göre farklı A değerleri için güvenilirlik.



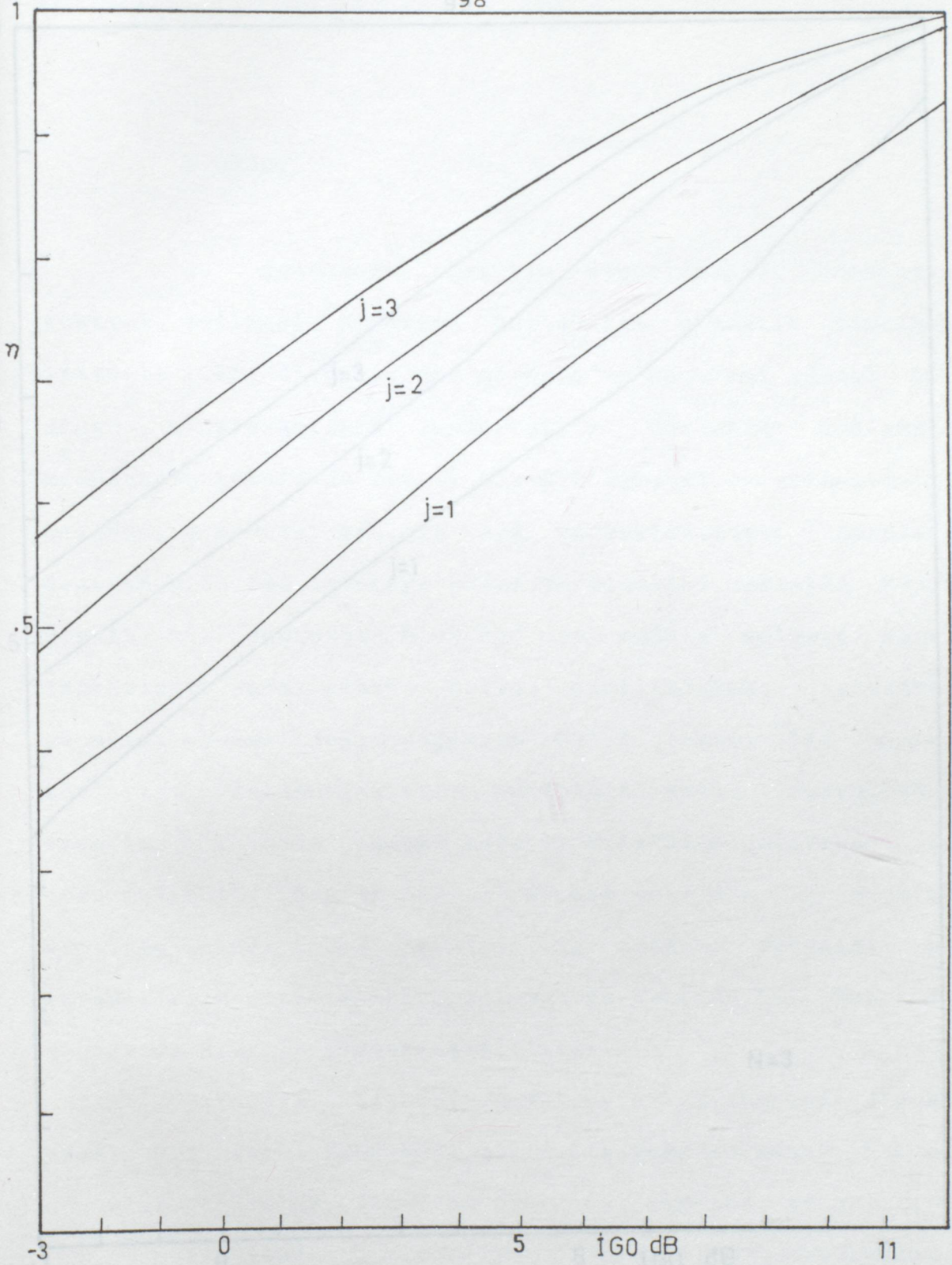
Şekil-5.11 Yineleme istemeli Viterbi kullanan N-AGD OYİ yapısı için benzetim modeline göre farklı A değerleri için güvenilirlik.



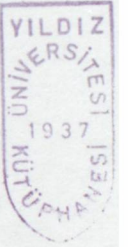
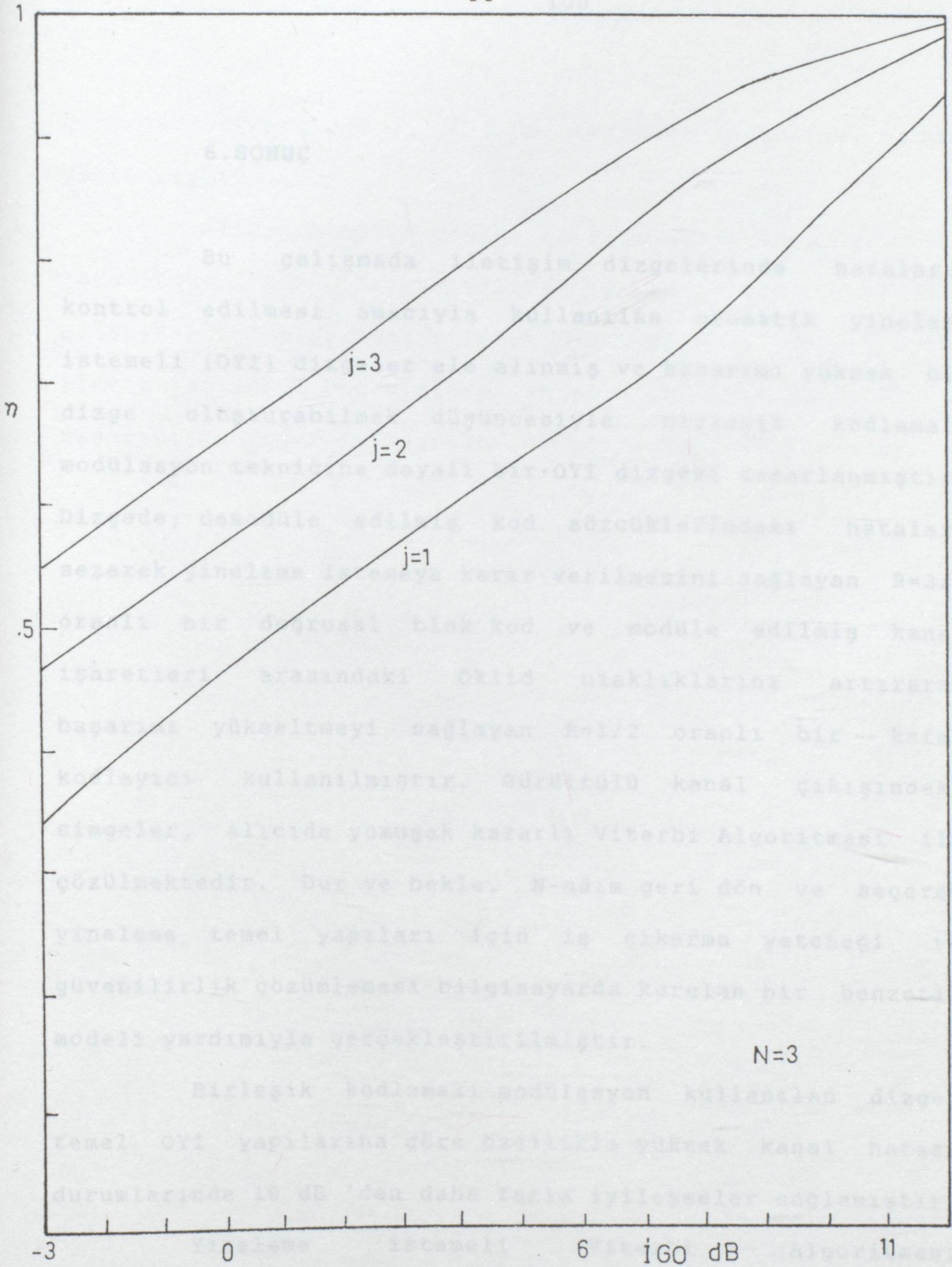
Şekil-5.12 Yineleme istemeli Viterbi kullanılan DB OYİ yapısı için $iGO=1$ dB değeri için farklı u 'lara göre iş çıkarma yeteneğinin değişimi.



Şekil-5.13 Yineleme istemeli Viterbi kullanılan DB OYİ yapısı benzetim modeli için $\text{fGO}=1$ dB değeri için farklı u 'lara göre iş çıkarma yeteneğinin değişimi.



Şekil-5.14 Alıcı tarafta ara bellek kullanılması durumunda, j kez iletim gerçekleştiğinde DB OYI yapısı için benzetim modeline göre iş çıkarma yeteneği.



Şekil-5.15 Alıcı tarafta ara bellek kullanılması durumunda, j kez iletim gerçekleştiğinde N -AGD OYİ yapısı için benzetim modeline göre iş çıkarma yeteneği.

6.SONUÇ

Bu çalışmada iletişim dizgelerinde hataların kontrol edilmesi amacıyla kullanılan otomatik yineleme istemeli (OYİ) dizgeler ele alınmış ve başarımı yüksek bir dizge oluşturabilmek düşüncesiyle birleşik kodlamalı modülasyon tekniğine dayalı bir OYİ dizgesi tasarlanmıştır. Dizgede; demodüle edilmiş kod sözcüklerindeki hataları sezerek yineleme istemeye karar verilmesini sağlayan $R=3/6$ oranlı bir doğrusal blok kod ve modüle edilmiş kanal işaretleri arasındaki Öklid uzaklıklarını artırarak başarımı yükseltmeyi sağlayan $R=1/2$ oranlı bir kafes kodlayıcı kullanılmıştır. Gürültülü kanal çıkışındaki simgeler, alıcıda yumuşak kararlı Viterbi Algoritması ile çözülmektedir. Dur ve bekle, N-adım geri dön ve seçerek yineleme temel yapıları için iş çıkarma yeteneği ve güvenilirlik çözümlemesi bilgisayarda kurulan bir benzetim modeli yardımıyla gerçekleştirilmiştir.

Birleşik kodlamalı modülasyon kullanılan dizge, temel OYİ yapılarına göre özellikle yüksek kanal hatası durumlarında 10 dB 'den daha fazla iyileşmeler sağlamıştır.

Yineleme istemeli Viterbi Algoritması kullanıldığında güvenilirliğin, A sabitinin seçilen değerlerine göre 1-5 dB arasında arttığı buna karşılık iş

çıkarma yeteneğinde , 1-7 dB arasında düşmeler olduğu gözlenmiştir.

Alıcı tarafta bellek kullanılarak, aynı bir kod sözcüğünün her bir iletiminde oluşan Öklid uzaklığı bilgilerinin değerlendirilmesi yöntemi ile de, tekrar iletim sayısına bağlı olarak en az 3 dB olmak üzere 20 dB değerlerine kadar iyileşmeler olduğu ortaya konulmuştur.

Tasarlanan dizgenin benzetim modeli aşağıda
 aşağıda Turbo Pascal dilinde yazılmış program ile
 açıklanmıştır.

7. EK

```

PROGRAM ARQ;
USES CRT, PRINTER;
label 1,2,3,4,5,6,7,1000,1001,1002,1003,1004,1005;
const a:integer=0; n:integer=0; k:integer=1; bkn:integer=1;
var packsay, bagil, w, i, ww, ali, fy, p, say1, teay, f, so, b, a, an,
    bny, z, adet: integer;
    tekrar, sakla, gr, gu, urus, so, vs, q, h, ad, sibir, iz: integer;
    day: text;
    sacis, kbn: char;
    testi, test2, is: boolean;
    vita, vry, e, vryi, vrys, r, r1, r2, eb, ek, dntek, dncirv: real;
    kat: array[1..4, 1..6] of boolean;
    ts: array[1..500] of boolean;
    ka: array[1..500] of boolean;
    bp: array[1..5000] of integer;
    br: array[1..300] of integer;
    bf: array[1..500] of boolean;
    bilgi: array[1..500, 1..3] of byte;
    c: array[1..6] of byte;
    si: array[1..2] of byte;
    cl: array[1..6, 1..2] of byte;
    sr: array[1..6, 1..4] of integer;
    q: array[1..6, 1..4] of real;
    re: array[1..6, 1..4] of real;
    kanal: array[1..6, 1..4] of integer;
    d: array[1..4, 1..6, 1..8, 1..4] of real;
    df: array[1..4, 1..6] of real;
    vel: array[1..4, 1..6] of byte;
    ccm: array[1..6] of byte;
    ht: array[1..6, 1..3] of byte;
    sy: array[1..3] of byte;
    sm: array[1..6] of byte;
    alimn: array[1..500, 1..6] of byte;
    c: array[1..500, 1..3] of byte;
    tampon: array[1..7] of string[3];
procedure bilgi_dizil;
begin
    kbn:=1;
    for i:=1 to k do
        begin
            r:=random; if r<0.5 then
                bilgi[i,1]:=0 else bilgi[i,1]:=1;
            (write('bilgi[i,1], '));
            end; (write('n'));
end;

```


Tasarlanan dizgenin benzetim modeli aşağıda sunulan Turbo Pascal dilinde yazılmış program ile sağlanmıştır.

```

PROGRAM ARQ;
USES CRT, PRINTER;
label 1,2,3,4,5,6,7,1000,1001,1002,1003,1004,1005;
const m:integer=0;n:integer=0;k:integer=1;bktn:integer=1;
var nacksay,bagil,w,i,ww,ali,fy,p,sayi,tsay,f,so,b,a,hn,
    hny,z,adet:integer;
    tekrar,sakla,gr,gu,urun,sc,ra,q,h,sd,sinir,iz:integer;
    dsy:text;
    secim,kon:char;
    test1,test2,is:boolean;
    vita,vry,v,vryi,vrys,r,r1,r2,eb,ek,dmtek,dmcift:real;
    knt:array[1..4,1..6] of boolean;
    ts:array[1..500] of boolean;
    ks:array[1..500] of boolean;
    bp:array[1..5000] of integer;
    br:array[1..500] of integer;
    bf:array[1..500] of boolean;
    bilgi:array[1..500,1..3] of byte;
    c:array[1..6] of byte;
    s:array[1..2] of byte;
    cl:array[1..6,1..2] of byte;
    sr:array[1..6,1..4] of integer;
    g:array[1..6,1..4] of real;
    rs:array[1..6,1..4] of real;
    kanal:array[1..8,1..4] of integer;
    d:array[1..4,1..6,1..8,1..4] of real;
    df:array[1..4,1..6] of real;
    yol:array[1..4,1..6] of byte;
    cozum:array[1..6] of byte;
    ht:array[1..6,1..3] of byte;
    sy:array[1..3] of byte;
    emin:array[1..6] of byte;
    alinan:array[1..500,1..6] of byte;
    o:array[1..500,1..3] of byte;
    tampon:array[1..7] of string[3];
procedure bilgi_dizisi;
begin
    m:=m+1;
    for i:=1 to k do
        begin
            r:=random;if r<0.5 then
                bilgi[m,i]:=0 else bilgi[m,i]:=1;
            {write(lst,bilgi[m,i],' ');}
        end;{writeln(lst);}
    end;

```



```

procedure c_blokkodlayici;
begin
    if secim='3' then sc:=gr else sc:=m;
    for i:=1 to k do
        c[i]:=bilgi[sc,i];
        c[4]:=c[1] xor c[2];
        c[5]:=c[2] xor c[3];
        c[6]:=c[1] xor c[2] xor c[3];
    end;
procedure c1_hammingkodlayici;
begin
    b:=0;
    for i:=1 to 2*k do
        begin
            s[2]:=s[1];
            if i<>1 then s[1]:=c[i-1];b:=b+1;
            c1[i,b]:=c[i] xor s[2];b:=b+1;
            c1[i,b]:=s[1] xor s[2];b:=0;
        end;
    end;
procedure ffsk;
begin
    i:=0;
    repeat
        i:=i+1;w:=1;
        if c[i]=0 then
            if c1[i,1]=0 then
                if c1[i,2]=0 then begin
                    sr[i,w]:=1;w:=w+1;sr[i,w]:=0;w:=w+1;sr[i,w]:=1;
                    w:=w+1;sr[i,w]:=0;
                end
                else begin
                    sr[i,w]:=-1;w:=w+1;sr[i,w]:=0;w:=w+1;sr[i,w]:=0;
                    w:=w+1;sr[i,w]:=-1;
                end
            else
                if c1[i,2]=0 then begin
                    sr[i,w]:=0;w:=w+1;sr[i,w]:=-1;w:=w+1;sr[i,w]:=1;
                    w:=w+1;sr[i,w]:=0;
                end
                else begin
                    sr[i,w]:=0;w:=w+1;sr[i,w]:=1;w:=w+1;sr[i,w]:=0;
                    w:=w+1;sr[i,w]:=-1;
                end
            end
        else
            if c1[i,1]=0 then
                if c1[i,2]=0 then begin
                    sr[i,w]:=-1;w:=w+1;sr[i,w]:=0;w:=w+1;
                    sr[i,w]:=-1;w:=w+1;sr[i,w]:=0;
                end
            end
        end
    until i=k;
end;

```



```

                                end
                                else begin
sr[i,w]:=1;w:=w+1;sr[i,w]:=0;w:=w+1;
sr[i,w]:=0;w:=w+1;sr[i,w]:=1;
                                end
else
    if c1[i,2]=0 then begin
sr[i,w]:=0;w:=w+1;sr[i,w]:=1;w:=w+1;
sr[i,w]:=-1;w:=w+1;sr[i,w]:=0;
                                end
else begin
sr[i,w]:=0;w:=w+1;sr[i,w]:=-1;w:=w+1;
sr[i,w]:=0;w:=w+1;sr[i,w]:=1;
                                end;
until i=2*k;
{for i:=1 to 6 do begin for w:=1 to 4 do
                                write(lst,sr[i,w], ' ');
write(lst, ' ');end;writeln(lst);}end;
procedure gurultu_ureteci;
begin
    randomize;
    for w:=1 to 2*k do begin
        for i:=1 to 4 do
            begin
                r1:=random;r2:=random;
                if odd(i) then
                    g[w,i]:=sqrt(-2*ln(r1))*cos(2*pi*r2)*vry
                else
                    g[w,i]:=sqrt(-2*ln(r1))*sin(2*pi*r2)*vry;
                    {write(lst,g[w,i], ' ');}
                end;{write(lst, ' ');}end;{writeln(lst);}
end;
procedure alinanisaret;
begin
    for i:=1 to 2*k do begin
        for w:=1 to 4 do begin
            rs[i,w]:=sr[i,w]+g[i,w];
            {write(lst,rs[i,w], ' ');}
            end;
            {write(lst, ' ');}end;{writeln(lst);}
        end;
procedure viterbi_kodcozucu;
label 100;
begin
    ts[sc]:=false;
    for i:=1 to 6 do
        begin
            for w:=1 to 4 do
                begin

```



```

for b:=w to w+4 do
begin
if (b=1) or (b=2) then f:=1; if (b=3) or (b=4) then f:=2;
if (b=5) or (b=6) then f:=3; if (b=7) or (b=8) then f:=4;
d[w,i,b,1]:=sqr(kanal[b,1]-rs[i,1]);
for a:=2 to 4 do
d[w,i,b,a]:=d[w,i,b,a-1]+sqr(kanal[b,a]-rs[i,a]);
if b=w then begin so:=f; dmtek:=d[w,i,b,a]; b:=b+3;
end
else
begin
dmcift:=d[w,i,b,a];
if dmtek<dmcift then
begin
if i=1 then df[w,i]:=dmtek else
df[w,i]:=df[so,i-1]+dmtek;
yol[w,i]:=so;
end else
begin
if i=1 then df[w,i]:=dmcift
else df[w,i]:=df[f,i-1]+dmcift;
yol[w,i]:=f;
end;
end;
end;
if i=1 then knt[w,i]:=true
else
if (abs(dmtek-dmcift)>=vita) and (knt[yol[w,i],i-1]) then
knt[w,i]:=true else knt[w,i]:=false;
end;
if vita<>0 then
if not (knt[1,i] or knt[2,i] or knt[3,i] or knt[4,i]) then
begin ts[sc]:=true; goto 100; end;
end;
ek:=df[1,6]; sayi:=1;
for w:=2 to 4 do
if ek>df[w,6] then begin
ek:=df[w,6]; sayi:=w;
end;
for i:=6 downto 1 do
begin
if (sayi=1) or (sayi=3) then alinan[sc,i]:=0
else alinan[sc,i]:=1;
sayi:=yol[sayi,i]; end;
{ for i:=1 to 6 do
write(lst, alinan[m,i]); writeln(lst); }
100: end;

```



```

procedure altprog;
begin
  c_blokkodlayici;
  c1_hammingkodlayici;
  ffsk;
  gurultu_ureteci;
  alinanisaret;
  viterbi_kodcozucu;
  {syndromkodcozucu;}
end;
procedure dogrulama;
begin o[gr,1]:=alinan[gr,1] xor alinan[gr,2];
      o[gr,2]:=alinan[gr,2] xor alinan[gr,3];
      o[gr,3]:=alinan[gr,3] xor o[gr,1];
end;
if avia='1' then begin write(' '); readln(n); end;
gotoxy(70,11); write('Blok Kodlayici'nin Tipi (n,k): ');
read(bta); gotoxy(53,11); write(' ');
readln(k); gotoxy(10,12);
write('Kanal Gurultusunun Baslangic, Bitis ve Artin Deger
leri: ');
read(vry1); gotoxy(70,12); write(' '); read(vry2);
gotoxy(74,12); write(' '); readln(v);
gotoxy(20,13); write('Program Kac Bilgi Uretecek? ');
readln(adet); gotoxy(20,14);
write('Benzetici Kac Kere Gerceklestirilecek? ');
readln(tekrr); gotoxy(20,15);
write('VITERBI de A Katsayisi: '); readln(vita);
assign(day,concat('rtex', avia, '.arg'));
rewrite(day); vry:=vry1;
writeln(day, vry, m:=2, bta:=1, k:=1, vry:=3, vry:=3,
v:=5,3, adet:=4, tekrr:=2, vita:=5,2);
1. gackay:=0; bagil:=0; a:=0; append(day, randomize; q:=0;
h:=0; is:=false;
for i:=1 to n do begin br[i]:='0'; bf[i]:='false'; end;
2. case saca of
  1. begin
    1. begin
      3. altprog; gr:=dogrulama;
      for i:=1 to k do begin
        test1:=bf[i] xor (alinan[gr,i] xor (o[gr,i-1] xor
          (if test1 then begin
            gackay:=gackay+1; write(gackay); goto 3;
            end;
          end;
        for i:=1 to k do begin
          test2:=bf[i] xor (alinan[gr,i]; if test2 then
            begin bagil:=bagil+1; goto 4; end;
          end;
        end;
      end;
    end;
  end;
end;

```


BEGIN

```

  s[1]:=0;s[2]:=0;
  kanal[1,1]:=1;kanal[1,2]:=0;kanal[1,3]:=1;kanal[1,4]:=0;
  kanal[2,1]:=0;kanal[2,2]:=1;kanal[2,3]:=-1;kanal[2,4]:=0;
  kanal[3,1]:=-1;kanal[3,2]:=0;kanal[3,3]:=0;kanal[3,4]:=-1;
  kanal[4,1]:=0;kanal[4,2]:=-1;kanal[4,3]:=0;kanal[4,4]:=1;
  kanal[5,1]:=0;kanal[5,2]:=1;kanal[5,3]:=0;kanal[5,4]:=-1;
  kanal[6,1]:=1;kanal[6,2]:=0;kanal[6,3]:=0;kanal[6,4]:=1;
  kanal[7,1]:=0;kanal[7,2]:=-1;kanal[7,3]:=1;kanal[7,4]:=0;
  kanal[8,1]:=-1;kanal[8,2]:=0;kanal[8,3]:=-1;kanal[8,4]:=0;
  clrscr;gotoxy(34,3);writeln('ARQ TEKNIKLER');gotoxy(38,5);
  WRITELN('1...SW');gotoxy(38,6);WRITELN('2...GBN');
  gotoxy(38,7);WRITELN('3...SR');
  gotoxy(35,9);WRITE('SECIMINIZ?');READLN(SECIM);
  if secim<>'1' then begin write('N=');readln(n);end;
  gotoxy(20,11);write('Blok Kodlayicinin Tipi (n,k):');
  read(bktn);gotoxy(53,11);write(',');
  readln(k);gotoxy(10,12);
  write('Kanal Gurultusunun Baslangic,Bitis ve Artim Deger
  leri:');
  read(vryi);gotoxy(70,12);write(',');read(vrys);
  gotoxy(74,12);write(',');readln(v);
  gotoxy(20,13);write('Program Kac Bilgi Uretecek?');
  readln(adet); gotoxy(20,14);
  write('Benzetim Kac Kez Gerceklestirilecek?');
  readln(tekrar);gotoxy(20,15);
  write('VITERBI de A katsayisi:');readln(vita);
  assign(dsy,CONCAT('rtez',secim,'.arq'));
  rewrite(dsy);vry:=vryi;
  writeln(dsy,secim:2,n:2,bktn:3,k:2,vryi:3,vrys:5,
  v:5:3,adet:4,tekrar:2,vita:5:2);
1:  nacksay:=0;bagil:=0; m:=0;append(dsy);randomize;q:=0;
  h:=0;is:=false;
  for i:=1 to n do begin br[i]:=0;bf[i]:=false;end;
2:  case secim of
    '1':begin bilgi_dizisi;
        3:altprog;gr:=m;dogrulama;
        for w:=k+1 to bktn do begin
          test1:=ts[gr] or ((alınan[gr,w])<>(o[gr,w-3]));
          if test1 then begin
            nacksay:=nacksay+1;{write(nacksay);}goto 3;
          end;
        end;
        for i:=1 to k do begin
          test2:=bilgi[gr,i]<>alınan[gr,i]; if test2 then
            begin bagil:=bagil+1;goto 4;end;
        end;
    end;
  end;
end;

```



```

'2':begin
    test1:=false;
    5:if (adet-m)<n then urun:=adet-m else urun:=n;
        for ali:=1 to urun do
            begin { write(ali,' ',n);}
        {if m=adet then begin a:=i;writeln('*');goto 6;end;}
        if not test1 then bilgi_dizisi else
            if m<fy then m:=m+1 else bilgi_dizisi;
            altprog;
        {for p:=1 to 3 do write(bilgi[m,p],' ');write(' ');}
        { for p:=1 to 6 do write(c[p]); write(' ');}
        for p:=1 to 6 do write(alinan[m,p]); write(' ');}
        gr:=m;dogrulama;
        { for p:=1 to 3 do write(o[gr,p]);writeln;}
        end;fy:=m;
        a:=m-urun+1;{writeln(a,' ',m);}
    6:for i:=a to m do
        begin
            for ww:=k+1 to bktn do
                begin test1:=ts[i] or (alinan[i,ww]<>o[i,ww-3]);
                    if test1 then
                        begin nacksay:=nacksay+1;{writeln(nacksay);}
                            m:=i-1;goto 5;
                        end;
                    end;
                for ww:=1 to k do begin
                    test2:=bilgi[i,ww]<>alinan[i,ww];
                    if test2 then begin
                        bagil:=bagil+1;{writeln('& ',bagil);}goto 7;
                    end;
                end;
            end;
        7:end;
    end;

```



```

h:=h+1;br[h]:=gu;bf[h]:=ks[gu];{write('#',br[h],' ');}
      if (h=n) and (bp[q-n+2]<>br[1]) then
begin is:=true;gr:=br[n]+1;goto 1001;end;
      goto 1001;
      end;
      if br[1]<>0 then begin

```



```

if h=n then if is then begin gr:=gu;goto 1001;end
                else begin gr:=gu+1;goto 1001;end;
h:=h+1;br[h]:=gu;bf[h]:=ks[gu];{write('* ',br[h], ' ');}
    if (h=n) and (bp[q-n+2]<>br[1]) then
        begin is:=true;gr:=br[n]+1;goto 1001;end;
    end;
    1003:end;iz:=q;
    if m=adet then
        for w:=iz-n+1 to iz do
            begin sinir:=bp[w];
                1004:if ks[bp[w]] then
                    begin {iz:='&';goto 1001;end;}
                    gr:=sinir;
altprog;dogrulama;q:=q+1;{write(q:3,m:3,gr:3,ra:3);}
                for i:=k+1 to bktn do
                    begin
Retransmit test1:=ts[gr] or (alanan[gr,w]<>o[gr,w-3]);
                    ks[gr]:=test1;{writeln(ks[gr]:6);}
                    if test1 then goto 1004;
                    end;
                end;
            end;
        end;
    END;
end;
4: if (m<adet) then goto 2;
    IF SECIM='3' THEN begin
        for w:=1 to adet do
            begin for i:=1 to k do
                if bilgi[w,i]<>alanan[w,i] then begin
                    bagil:=bagil+1;goto 1005;end;
                    1005:end;
                    writeln(dsy, vry:5:3,q:7,bagil:7);
                end
            ELSE writeln(dsy,vry:5:3,nacksay:7,bagil:7);
        close(dsy);vry:=vry+v;
        if vry>vrys then begin tsay:=tsay+1;
            if tsay=tekrar then halt;vry:=vryi;end;
            goto 1;
        end.

```


KAYNAKÇA

1. A. R. K. Sastry, "Improving ARQ Performance on Satellite Channels Under High Error Rate Conditions", IEEE Tran. on Com., April 1975, pp 436-439.
2. P. S. Sindhu, "Retransmission Error Control with Memory", IEEE Tran. on Com., may 1977, pp 473-479.
3. J. J. Metzner, "Improvements in Block-Retransmission Schemes", IEEE Tran. on Com., feb 1979, pp 524-532.
4. S. Lin & P. S. Yu, "An Effective Error Control Scheme for Satellite Communications", IEEE Tran. on Com., March 1980, pp 395-401.
5. P. S. Yu & S. Lin, "An Efficient Selective-Repeat ARQ Scheme for Satellite Channels and Its Throughput Analysis", IEEE Tran. on Com., March 1981, pp 353-363.
6. M. J. Miller & S. Lin, "The Analysis of Some Selective Repeat ARQ Schemes with Finite Receiver Buffer", IEEE Tran. on Com., Sep. 1981, pp 1307-1315.
7. E. J. Weldon, "An Improved Selective Repeat ARQ Strategy", IEEE Tran. on Com., march 1982, pp 480-486.
8. M. Moeneclaey & H. Bruneel, "Efficient ARQ Scheme for High Error Rate Channels", Electronics Letters, Nov. 1984, pp 986-987.
9. G. Benelli, "An ARQ Scheme with Memory and Soft

Error Detectors", IEEE Tran. on Com., march 1985, pp 285-288.

10. G. Benelli, "Efficient ARQ Technique Using Soft Demodulation", Electronics Letters, Feb. 1986, pp 205-206.

11. G. Benelli, "An ARQ Scheme with Memory and Integrated Modulation", IEEE Tran. on Com., july 1987, pp 689-697.

12. G. Benelli, "Integration of channel coding and modulation in the ARQ protocols", IEE Proceedings, dec. 1988, pp 577-583.

13. H. Yamamoto & K. Itoh, "Viterbi Decoding Algorithm for Convolutional Codes with Repeat Request", IEEE Tran. on Inf. Theory, sep. 1980, pp 540-547.

14. S. Lin & D. J. Costello, "Protocols and Techniques for Data Communication Networks, Prentice-Hall, 1979.

15. Y. M. Wang & S. Lin, "A modified selective repeat type II hybrid ARQ system and its performance analysis", IEEE Tran. on Com., may 1983, pp 593-608.

16. G. Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Sets Part I & Part II", IEEE Feb. 1987, pp 5-21.

17. G. Benelli, "New mapping rules for combination of M-ary modulation and error-detecting codes in ARQ systems", IEE Proceedings, aug 1990, pp 197-204.

18. R. H. Deng, "Hybrid ARQ scheme using TCM and

code combining", Electronics Letters, May 1991, pp 866-868.

19. A. J. Viterbi, "Convolutional codes and their performance in communication systems", IEEE Tran. on Com., oct. 1971, pp 751-772.

20. E. Zehavi & J. Wolf, "On the Performance Evaluation of Trellis Codes", IEEE Tran. on Inf. Theory, march 1987, pp 196-202.

21. Thomas C. Barfee, "Data Communications, Networks and Systems", Howard W. Sams & Co., SAMS 22235.

22. V. Bhargava, D. Haccoun, R. Matyas, "Digital Communication By Satellite", John Wiley, 1981.

23. Proakis, "Digital Communications", 1983.

24. H. O. Burton & D. D. Sullivan, "Errors and Error Control", IEEE Proceedings, nov. 1972, pp 1293-1301.



ÖZGEÇMİŞ

Raci ALGIN, 1967 yılında Erzurum' da doğdu. 1984 yılında Yıldız Üniversitesi Elektronik ve Haberleşme Mühendisliği Bölümüne girdi. 1989 yılında bu bölümü bitirdikten sonra, aynı yıl Yıldız Üniversitesi Fen Bilimleri Enstitüsünde yüksek lisans programına başladı. 1990 yılında, aynı mühendislik bölümünde araştırma görevlisi olarak çalışmaya başladı. Halen aynı göreve devam etmektedir.

