ATATÜRK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Y. LİSANS TEZİ

LİNEER KODLARIN BİLGİSAYAR DESTEĞİ İLE HESAPLANMASI

Aydın SEÇER

MATEMATİK ANABİLİM DALI

ERZURUM

2006

Doç.Dr.Abdullah KOPUZLU danışmanlığında, Aydın SEÇER tarafından hazırlanan bu çalışma 22.05.2006 tarihinde aşağıdaki jüri tarafından Matematik Anabilim Dalı'nda Yüksek Lisans tezi olarak kabul edilmiştir.

Başkan: Doç.Dr.Abdullah KOPUZLU *İmza :*

Üye: Prof.Dr.Hüseyin ATDIN *İmza :*

Üye: Doç.Dr.Uğur TAVUZ *İmza :*

Yukarıdaki sonucu onaylarım

(imza)

.....

Enstitü Müdürü

ÖZET

Yüksek Lisans Tezi

LİNEER KODLARIN BİLGİSAYAR DESTEĞİ İLE HESAPLANMASI

Aydın SEÇER

Atatürk Üniversitesi
Fen-Edebiyat Fakültesi
Matematik Anabilim Dalı

Danışman: Doç. Dr. Abdullah KOPUZLU

Bu tezde, lineer kodlar bilgisayar desteği ile çözümleri düşünüldü. Bunun için yöntem olarak sendrom çözüm algoritmasından faydalanıldı. Bu çalışma sonunda verilen bilgisayar programı girilen parity kontrol matrisine göre kod üreterek yaklaşık çözümleri üretti. Bu sayede hatalı kod sözcükleri, hata şablonları sayesinde en yakın kod sözcükleri ile mukayesesi kolaylaştırıldı. Hatalı kod sözcükleri program tarafından en uygun seçim yapılarak düzeltildi.

2006, 36 Sayfa

Anahtar Kelimeler: Lineer Kod, Sendrom Çözüm, Koset, Koset Lideri.

ABSTRACT

Master Thesis

SOLVING LINEAR CODES WITH COMPUTER

Aydın SEÇER

Atatürk University
Faculty of Arts and Sciences
Department of Mathematics

Supervisor: Assoc. Prof. Abdullah KOPUZLU

In this thesis, we consider solving linear codes with computer program. We have applied syndrom algorithm to solve linear codes with computer. We have given a computer program at the end of this study which calculates linear codes using syndrom algorithm. This algorithm generates linear codes from the parity control matrix given and it generates approximate solutions for the received code words. Therefore, comparision of codewords which are correct and uncorrect was facilitated. The codewords which have been received as uncorrect were corrected by this algorithm.

2006, 36 Pages

Keywords: Linear Code, Syndrome Solution, Coset, Coset Leader.

TEŞEKKÜR

Yüksek Lisans tezi olarak sunduğum bu çalışma Atatürk Üniversitesi Fen-Edebiyat Fakültesi Matematik Bölümü'nde hazırlanmıştır.

Bu çalışma esnasında tez danışmanlığımı yürüten hocam Atatürk Üniversitesi Fen Edebiyat Fakültesi Matematik Bölümü Öğretim Üyesi Sayın Doç. Dr. Abdullah KOPUZLU'ya yakın ilgi, teşvik ve yardımlarından dolayı teşekkürlerimi sunarım.

Çalışmalarım boyunca, yardımlarını esirgemeyen ve bu çalışmada büyük bir katkısı bulunan hocam ve aynı zamanda yardımcı danışmanım olan Yıldız Teknik Üniversitesi Fen-Edebiyat Fakültesi Matematik Bölümü Öğretim Üyesi Sayın Prof. Dr. Mustafa BAYRAM'a teşekkürlerimi sunarım.

Tez hazırlık aşamasında değerli fikir ve katkılarıyla bana yol gösteren ve bilgilerinden faydalandığım Atatürk Üniversitesi Fen-Edebiyat Fakültesi Matematik Bölümü araştırma görevlilerinden Sayın Arş.Gör. Turgut YELOĞLU'na teşekkürlerimi bir borç bilirim.

Çalışmalarım boyunca desteklerini ve sonsuz güven duygusunu esirgemeyen, beni bu günlere kadar getiren aileme en içten duygularımla teşekkürlerimi sunarım.

Aydın SEÇER
Nisan 2006

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Kodlama teorisi bilgisayarların ortaya çıkış tarihine dayanır. İlk bilgisayarlar çok büyük ebatlarda olup aynı zamanda mekanik röle tabanlı olduklarından günümüz bilgisayarlarıyla karşılaştırıldığında veri iletim güvenlikleri çok düşük olan hantal makinelerdi. Bu yüzden tek bir rölede hata çıksa bütün hesaplamalar hatalı sonuçlanmaktaydı. O zamanın mühendisleri hatalı röleyi tespit edip değiştirmek için değişik yollar geliştirmişlerdi.

Bell laboratuarları için çalışan R.W. Hamming, eğer makineler bir arıza olduğunu tespit edebilirlerse makinelerin bu hataları düzeltebilecekleri fikrini ortaya atmıştı. Daha sonra bu mantık üzerine kurulmuş, tek hatanın olduğu saptandığında bunu düzeltebilecek bir kodlama sistemi geliştirmiştir.

Claude Shannon (1948) tarafından yine bu mantık üzerine kurulu teorik bir yapı (framework) kodlama teorisine sunulmuştur.

Claude Elwood Shannon elektronik iletişim çağının babası olarak bilinir. 1948'de yayınlamış olduğu iletişim teorisi adlı makale hala günümüz bilgi teknolojisinin ana yapısını oluşturmaktadır. Bu makale 20. yüzyılın en büyük ve dahiyane başarısı olarak nitelendirilmektedir.

Golay (1949) Hamming kodlarının gelişimine yardımcı olacak çalışmalar yapmıştır. Aynı zamanda Golay kodlarını ortaya çıkarmıştır.

Richard Hamming (1950) tarafından verilen ve Hamming kodları olarak bilinen hata düzeltme kodları en önemli kodlama sistemlerinden birini teşkil eder. Hamming kodları hatayı bulan ve düzelten kodlardır.

Gilbert (1952), Varshamov (1957) verilen herhangi uzunlukta ve minimum mesafeli kodların alt sınırları ile ilgili teorilerini öne sürmüşlerdir bunlar Gilbert-Varshamov sınırları olarak da bilinir. Daha sonra lineer kodların önemli bir sınıfı olan cyclic kodlar bulunmuştur.

Slepian (1960) tarafından lineer kodlar için çözümleme tablosu verilmiştir.

Birbirlerinden bağımsız olarak R.C. Bose (1960), D.K. Ray-Chaudhuri ve A. Hocquenhgem tarafından hata düzeltmede, kodlama ve kod çözmede etkin bir özelliğe sahip olan cyclic kodların önemli bir ailesi teşkil eden BCH kodları keşfedilmiştir. BCH kodları hamming kodlarının genelleştirilmesidir. Daha sonraları çoklu kanallar ve çoklu alıcılar için Space-Time kodları geliştirilmiştir.

Irving Reed (1960) ve Gus Solomon, hata düzeltme kodlarında yeni bir sınıf olan ve günümüzde kompakt disk çalarlardan uzun mesafeli iletişim araçlarına kadar değişik alanlarda kullanılan Reed-Solomon kodlarını keşfetmişlerdir.

Nordstrom ve Robinson (1967), Nordstrom-Robinson kodu olarak bilinen nonlineer kodların en iyi temsilcilerinden olan 16 uzunluklu 256 sözcüklü bir kod keşfetmişlerdir.

V.D. Goppa (1977) cebirsel geometri kullanarak günümüzde kendi adıyla bilinen Goppa kodlarını keşfetmiştir.

2. KURAMSAL TEMELLER

2.1. Ön Bilgiler

Bu bölümde, tezde kullanılan bazı temel kavramlar verilecektir.

2.1.1. Tanım (Alfabe): Kodlama teorisinde alfabe p asal, k pozitif tamsayı olmak üzere ($q = p^k$) mertebeli sonlu cisimdir. Alfabe, F_q ile gösterilir. Alfabenin elemanları q tane farklı sembolden oluşur ve

$$F_q = \{0,1,2,...,q-1\} \tag{2.1}$$

kümesi ile verilir.

2.1.2. Tanım ($\left(F_q\right)^n$ Uzayı): $a_i \in \left(F_q\right)^n$ olmak üzere $a_1, a_2, a_3, ..., a_n$ şeklindeki tüm sıralı n-lilerin kümesi $\left(F_q\right)^n$ ile gösterilir. Bu uzaya n- uzunluklu q –lu bütün kod sözcüklerinin uzayı denir. Bir sonraki tanımda verilecek olan kod kümesi bu uzaydan alınan bir alt uzaydır

2.1.3. Tanım (Kod): Genel anlamda q lu bir kod denildiğinde q tane farklı sembolün oluşturduğu $F_q = \{0,1,2,...,q-1\}$ kümesinin elemanlarıyla oluşturulan dizilerden kurulan bir küme anlaşılır ve C ile gösterilir. Burada $q = 2$ alınırsa kod binary kod adını alır. $q = 3$ alınırsa ternary kod ve $q = 4$ alınırsa qua-ternary kod adını almaktadır. Bir kodda her kod sözcüğü n tane sembolden oluşmuş ise bu koda n uzunluklu blok kod yada kısaca n uzunluklu kod diyeceğiz.

2.1.4. Tanım (Denk Kodlar): Verilen iki tane q-lu koddan biri diğerinden aşağıda verilmiş işlemlerle elde edilebiliyorsa bu kodlara denk kodlar diyeceğiz.

I. Koddaki konumların (sütunların) permütasyonu.

II. Belli bir konumdaki (sütundaki) sembollerin permütasyonu.

Bir C kodunu $M \times n$ şeklinde bir matris ile belirtmek mümkündür. Burada matrisin satırları kod sözcüklerinden oluşur. Buna göre **I.** işlem matrislerin sütunlarının bir permütasyonuna karşılık gelir. **II.** işlem de verilen herhangi bir sütundaki sembollerin permütasyonuna karşılık gelir. Bu işlemler altında kod sözcüklerinin arasındaki mesafe değişmez.

2.1.5. Tanım (Ağırlık): $x \in (F_2)^n$ vektörü verilsin. x vektörünün ağırlığı demek, x vektöründeki 1 lerin adedi demektir. Ağırlık $W(x)$ notasyonu ile gösterilir.

2.1.6. Tanım: Bir (n,M,d)-kodu, uzunluğu n olan, M tane kod sözcüğüne içeren ve minimum mesafesi d olan bir koddur. Bu tezde q-lu bir (n,M,d)-kodda M'nin alabileceği en büyük değer $A_q(n,d)$ ile gösterilecektir.

2.2. Kodlama Teorisinde Uzaklık Kavramı

2.2.1. Tanım (Hamming-Uzaklığı): Hamming uzaklığı demek, iki kod sözcüğün farklı bileşenlerinin sayısı demektir. $x,y \in (F_q)^n$ olmak üzere hamming uzaklığı $d(x,y)$ ile gösterilir. Örneğin; $(F_2)^5$ de $d(00101,11001)=5$, $(F_3)^4$ de $d(1122,1201)=3$ dür. Hamming uzaklığı,

i. $d(x,y)=0 \Leftrightarrow x=y$

ii. Her $x,y \in (F_q)^n$ için $d(x,y)=d(y,x)$ (2.2)

iii. Her $x,y,z \in (F_q)^n$ için $d(x,y) \le d(x,z)+d(z,y)$

metrik şartlarını sağladığından aynı zamanda bir metriktir.

2.2.2. Minimum Uzaklık: C kodunun minimum uzaklığı, bu kodda alınan farklı kod sözcükleri arasındaki uzaklıklardan minimum olanı demektir ve $d(C)$ ile gösterilir ve

$$d(C)=\min\left\{d(x,y)\middle| x,y \in C, x \neq y\right\} \tag{2.3}$$

olarak tanımlanır. Örneğin,

$$C_1=\begin{cases}00\\01\\10\\11\end{cases},\ d(C_1)=1,\ C_2=\begin{cases}000\\011\\101\\110\end{cases},\ d(C_2)=2,\ C_3=\begin{cases}00000\\01101\\10110\\11011\end{cases},\quad d(C_3)=3$$

2.2.3. Teorem: i. $d(C)$ minimum mesafe olmak üzere $d(C)\geq s+1$ şartını sağlayan C kodu s kadar hatayı her kod sözcüğünde tespit edebilir.

ii. $d(C)\geq 2t+1$ şartını sağlayan C kodu t adet hatayı her kod sözcüğünde düzeltebilir.

İspat: i. $d(C)\geq s+1$ olsun. Bir x sözcüğü gönderilsin ve s tane veya s den daha az hata ortaya çıkmış olsun. Bu durumda alınan sözcük, yeni bir kod sözcüğü olması imkansızdır. Böylece hatalı olduğu tespit edilmiş olur.

ii. $d(C)\geq 2t+1$ kabul edelim. Bir x sözcüğü gönderilsin ve t adet yada daha az sayıda hata oluşarak y sözcüğünün alındığını kabul edelim. Bu durumda $d(x,y)\leq t$ olur. x', x den farklı herhangi bir kod sözcüğü olsun. $d(x',y)\geq t+1$ şeklindedir. Aksi halde $d(x',y)\leq t$ olursa üçgen eşitsizliğinden

$$d(x,x')\leq d(x',y)+d(x,y)\leq 2t \tag{2.4}$$

bulunur. Bu $d(C)\geq 2t+1$ ifadesiyle çelişir. Böylece x in y ye en yakın kod sözcüğü olduğunu düşünürüz.

2.2.4. Sonuç: d minimum uzaklığına sahip bir C kodu her kod sözcüğünde $d-1$ kadar hatayı sezer ve $\left[\frac{(d-1)}{2}\right]$ kadar hatayı düzeltebilir. Örneğin $d(C)=3$ ise, C kodu ya tek hata düzeltme kodu yada iki hata tespit eden kod olarak kullanılır.

2.3. F_q^n de Küre Kavramı

2.3.1. Tanım: $u\in(F_q)^n$ ve bir $r\geq 0$ tamsayısı verilsin. u merkezli r yarıçaplı bir küre $S(u,r)$ ile gösterilir.

$$S(u,r)=\left\{v\in(F_q)^n \mid d(u,v)\le r\right\} \tag{2.5}$$

şeklinde tanımlanır.

2.3.2. Yardımcı Teorem: $0\le r\le n$ olmak üzere $(F_q)^n$ de r yarıçaplı bir küre

$$\binom{n}{0}+\binom{n}{1}(q-1)+\binom{n}{2}(q-1)^2+.....+\binom{n}{r}(q-1)^r \tag{2.6}$$

adet vektör içerir.

İspat: $u\in(F_q)^n$ olsun. $m\le n$ olmak üzere u dan uzaklığı m olan kaç adet v vektörü bulunduğunu bulmaya çalışalım. $d(u,v)=m$ olduğunda v nin u dan farklı olduğu konumların sayısı m dir. u ve v vektörlerinde m tane konum farklı olacak şekilde $\binom{n}{m}$ tane seçenek vardır. $(F_q)^n$ de q tane farklı sembol vardır. O zaman bu m konumun her birinde v vektörünün elemanları sözü edilen konumdaki sembol, u dan farklı olacak şekilde $q-1$ türlü seçilebilir. O zaman u dan m uzaklıktaki vektörlerin toplam sayısı,

$$\binom{n}{m}(q-1)^m \tag{2.7}$$

kadar olur. Sonuç olarak $S(u,r)$ deki vektörlerin sayısı,

$$\binom{n}{0}+\binom{n}{1}(q-1)+\binom{n}{2}(q-1)^2+.....+\binom{n}{r}(q-1)^r \tag{2.8}$$

olarak bulunur.

2.3.3. Teorem (Hamming Sınırı): q-lu $(n,M,2t+1)$ kod

$$M\left\{\binom{n}{0}+\binom{n}{1}(q-1)+\binom{n}{2}(q-1)^2+...+\binom{n}{t}(q-1)^t\right\}\le q^n \tag{2.9}$$

eşitsizliğini sağlar.

İspat: q-lu bir $(n,M,2t+1)$-kod C olsun. Küre tanımında belirtildiği gibi farklı kod sözcüklerini merkez kabul eden t yarıçaplı iki kürenin ortak hiçbir vektörü yoktur.

Böylece M tane kod sözcüğünü merkez kabul eden t yarıçaplı M tane küre içindeki vektörlerin toplam sayısı (2.9) eşitsizliğin sol tarafına eşittir. Bu sayı ise $(F_q)^n$ deki tüm vektörlerin sayısı olan q^n den küçük veya eşit olmak zorundadır. (2.9) eşitsizliği binary kodlar için yazılırsa, herhangi bir binary $(n, M, 2t+1)$

$$M\left\{\binom{n}{0}+\binom{n}{1}+\binom{n}{2}+...+\binom{n}{t}\right\} \le 2^n \tag{2.10}$$

eşitsizliğini sağlar. Verilen bir q, n, d değerleri için Hamming sınırı vasıtasıyla M için bir üst sınır elde edilir.

2.3.4. Tanım (Mükemmel Kodlar): Eğer yukarıda (2.9) ile verilen Hamming sınırı eşitsizliğinde eşitlik hali sağlanıyorsa mükemmel kod adını alır. Daha açık bir anlatımla q-lu bir $(n, M, 2t+1)$-kod,

$$M\left\{\binom{n}{0}+\binom{n}{1}(q-1)+\binom{n}{2}(q-2)^2+...+\binom{n}{t}(q-1)^t\right\} = 2^n \tag{2.11}$$

eşitliğini sağlıyorsa mükemmel kod adını alır.

2.4. Lineer Kodlar

2.4.1. Tanım (Lineer Kod): F_q Alfabesini düşünelim. Burada q asal kuvvet ve $(F_q)^n$ de $V(n,q)$ vektör uzayı olarak alınacaktır. Bir vektör $(x_1, x_2, x_3, ..., x_n)$ kısaca $x_1x_2x_3...x_n$ olarak yazılacaktır. $GF(q)$ üzerinde bir lineer kod bazı n tamsayıları için $V(n,q)$ nun bir alt uzayı olması için gerek ve yeter şart,

i. $u+v \in C, \forall u, v \in C$ (2.12)

ii. $au \in C, \forall u \in C, a \in GF(q)$

şartlarının sağlanmasıdır. Bu şartları sağlayan $V(n,q)$ nun bir C alt kümesine lineer kod diyeceğiz. Özel olarak bir binary kod ancak ve ancak iki kod sözcüğünün toplamı bir kod sözcüğü ise lineer bir koddur. Örneğin,

$$C_1=\begin{Bmatrix}0&0\\0&1\\1&0\\1&1\end{Bmatrix} \qquad C_2=\begin{Bmatrix}0&0&0\\0&1&1\\1&0&1\\1&1&0\end{Bmatrix} \qquad C_3=\begin{Bmatrix}0&0&0&0&0\\0&1&1&0&1\\1&0&1&1&0\\1&1&0&1&1\end{Bmatrix}$$

2.4.2. Teorem: C bir lineer kod ve $W(C)$ de C de sıfırdan farklı kod kelimelerinin en küçüğü olsun. O zaman

$$d(C)=W(C) \tag{2.13}$$

olur.

İspat: $x,y\in C$ için,

$$d(C)=d(x,y) \tag{2.14}$$

ve bir önceki teoremden

$$d(C)=d(x,y)=W(x-y)\geq W(C) \tag{2.14a}$$

yazılabilir. Çünkü $x-y$, C lineer kodunun bir kod sözcüğüdür. Şimdi $x\in C$ alırsak,

$$W(C)=W(x)=d(x,0)\geq d(C)$$

(2.14b)

$\left(0\in C\right)$ olur. (2.14a) ve (2.14b) den

$$d(C)=W(C)$$

elde edilir.

2.4.3. Lineer Kodların Özellikleri: Lineer kodlar aşağıdaki özelliklere sahiptir.

i. 0 vektörü bir lineer kodun elemanıdır.

ii. Lineer kod deyimi yerine bazı kitaplar grup kod deyimi de kullanmaktadırlar.

iii. M tane kod sözcüğü içeren genel bir kodda minimum mesafeyi bulmak için genel bir kural olmadığına göre

$$\binom{M}{2}=\frac{1}{2}M(M-1) \tag{2.15}$$

kez kıyaslama yapmak gerekir.

Lineer kodda ise bir önceki teoremden sadece $M-1$ kez kod sözcüğünün ağırlığını kontrol etmek yeterlidir. Çünkü

$$d(C)=W(C) \tag{2.16}$$

şeklndedir.

iv. Lineer olmayan bir kodu belirtmek için tüm kod sözcüklerini liste halinde yazmamız gerekir. Oysa lineer bir $[n,k]$-kodun bir bazındaki vektörleri satırlara yazarak elde edilen bir $k\times n$ matris bu lineer kodun üreteç matrisi adını alır.

2.4.4. Tanım (Üreteç Matrisi): Bir $[n,k]-C$ kodunun *G* üreteç matrisi, satırları vektör uzayının bir tabanı olan vektörlerden seçilmiş bir $k\times n$ matrisidir.

2.4.5. Tanım (Denk Lineer Kodlar): $GF(q)$ üzerinde iki lineer kodun biri aşağıdaki işlemlerin bir kombinasyonu ile diğerinden elde edilebiliyorsa bu kodlar **denk lineer kodlar** denir,

i. Koddaki konumların permütasyonu,

ii. Belli bir konumdaki sembollerin sıfırdan farklı bir skalerle çarpımı.

2.4.6. Teorem: İki tane $k\times n$ üreteç matristen biri diğerinden aşağıdaki işlemlerle elde edilebiliyorsa bu iki matris $GF(q)$ üzerinde denk lineer $[n,k,d]$-kod üretir.

i. Satırların permütasyonu,

ii. Bir satırın sıfırdan faklı bir skalerle çarpımı,

iii. Bir satırın bir skalerle çarpımının diğer satıra eklenmesi,

iv. Sütunların permütasyonu,

v. Herhangi bir sütunun sıfırdan farklı bir skalerle çarpımı.

2.4.7. Tanım (Üreteç Matrisinin Standart Formu): Bir $[n,k]$-kodun üreteç matrisi *G* olsun. i, ii, iii, iv, v işlemlerinin uygulanması ile *G*-üreteç matrisi

$$[I_k : A] \tag{2.17}$$

şeklinde düşünülebilir. Buna üreteç matrisin **standart formu** denir. Burada I_k, $k \times k$ boyutlu birim matris, A ise $k \times (n-k)$ boyutlu matristir.

2.4.8. Tanım (Koset): $GF(q)$ üzerinde bir $[n,k]$-C kodu alalım. Bir $v \in V(n,q)$ alalım. Bu durumda,

$$v + C = \{v + x \mid x \in C\} \tag{2.18}$$

kümesi C nin bir **koseti** (kalan sınıfı) adını alır.

2.4.9. Tanım (Dual Kod): Lineer bir $[n,k] - C$ ele alalım. C nin duali $C^{\perp}$ ile gösterilir ve

$$C^{\perp} = \{v \in V(n,q) \mid vu = 0,\ \forall\ u \in C \text{ için}\} \tag{2.19}$$

şeklinde tanımlanır.

2.4.10. Tanım (Parity-Kontrol Matrisi): Bir $[n,k]$- C kod verilsin. $C^{\perp}$ in bir H üreteç matrisi C nin bir parity kontrol matrisi adını alır. Bu tanıma göre, H bir $(n-k) \times n$ matristir.

$$G \cdot H' = 0$$

$$C = \{x \in V(n,q) \mid x \cdot H' = 0\} \tag{2.20}$$

şeklindedir. Bir parity kontrol matrisi ile bir lineer kod tamamen belirtilebilir. Örneğin parity-kontrol matrisi,

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

olan bir C kodu

$$\{(x_1x_2x_3x_4) \in V(4,2) \mid x_1 + x_2 = 0, x_3 + x_4 = 0\}$$

şeklindedir ve,

$$(x_1x_2x_3x_4)\,H' = (x_1x_2x_3x_4) \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} = 0$$

olur. Burada,

$$\left.\begin{aligned} x_1 + x_2 &= 0 \\ x_3 + x_4 &= 0 \end{aligned}\right\} \tag{2.21}$$

denklemleri parity-kontrol denklemleri adını alır.

2.4.11. Teorem (Lagrange Teoremi): $GF(q)$ üzerinde bir $[n,k]$-C kodu verilsin. O zaman,

i. $V(q,n)$ nun her vektörü C nin bir kosetinde bulunur.

ii. Her Koset kesin olarak q^k tane vektör içerir.

iii. İki Koset ya ayrıktır yada birbirinin aynıdır.

İspat: i. $a \in V(q,n)$ ise,

$$a = a + 0 \in a + C \tag{2.22}$$

olur.

ii.

$$f : C \to a + C$$

fonsiyonunu tanımlayalım. Burada,

$$f(x) = a + x$$

olup f, 1-1 dir. Böylece,

$$|a + C| = |C| = q^k \tag{2.23}$$

eşitliği bulunur.

iii. $a + C$ ve $b + C$ kosetlerinin örtüştüğünü farzedelim. O zaman bazı v vektörleri için

$$v \in (a + C) \cap (b + C)$$

eşitliğine sahip olduğumuz görülür. Böylece bazı $x, y \in C$ için

$$v = a + x = b + y$$

olur. Buradan,

$$b = a + (x - y) \in C$$

ve

$$b + C = a + C \tag{2.24}$$

eşitlikleri elde edilir.

2.4.12. Teorem: Bir $[n,k]$-kod C olsun. C nin üreteç matrisinin standart formu

$$G=\left[I_k \mid A\right]$$

olsun. O zaman C nin bir parity-kontrol matrisi

$$H=\left[-A^T \mid I_{n-k}\right] \tag{2.25}$$

şeklinde olur.

İspat:

$$G=\left[\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & a_{11} & a_{12} & \cdots & a_{1,n-k} \\ 0 & 1 & \cdots & 0 & a_{21} & a_{22} & \cdots & a_{2,n-k} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & a_{k1} & a_{k2} & \cdots & a_{k,n-k} \end{array}\right] \tag{2.26}$$

$$H=\left[\begin{array}{cccc|cccc} -a_{11} & -a_{21} & \cdots & -a_{k1} & 1 & 0 & \cdots & 0 \\ -a_{12} & -a_{22} & \cdots & -a_{k2} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -a_{1,n-k} & -a_{2,n-k} & \cdots & -a_{k,n-k} & 0 & 0 & \cdots & 1 \end{array}\right] \tag{2.27}$$

Burada H parity-kontrol matrisinin boyutlarına sahiptir ve satırları lineer bağımsızdır. Bu yüzden H nın her satırının G nin her bir satırına ortogonal olduğu rahatlıkla görülebilir. G nin *i.* satırının H nin *j.* satırıyla iç çarpımı,

$$0+\ldots+0+\left(-a_{ij}\right)+0+\ldots+0+\ldots+a_{ij}+\ldots+0=0$$

şeklinde olur.

2.4.13. Tanım (Koset Lideri): Bir kalan sınıfında minimum ağırlığa sahip bir vektör koset lideri adını alır. Bir kalan sınıfında minimum ağırlıklı birden fazla vektör varsa bunlardan herhangi birini koset lideri olarak alabiliriz. Lagrange teoremi,

$$V(n,q)=\left(0+C\right)\cup\left(a_1+C\right)\cup\ldots\cup\left(a_s+C\right) \tag{2.28}$$

şeklindedir. Burada,

$$s=q^{n-k}-1 \tag{2.29}$$

olur. Biz $0, a_1, a_2, a_3, ..., a_s$ leri koset lideri olarak alabiliriz. Bir $[n,k]$- C kodun (Slepian) standart dizilişi, $V(n,q)$ deki bütün vektörlerin

$$q^{n-k} \times q^k \tag{2.30}$$

boyutunda dizilişidir. Bu dizilişte ilk satır

$$0 + C \tag{2.31}$$

diğer satırlar ise,

$$a_i + C \tag{2.32}$$

şeklindedir.

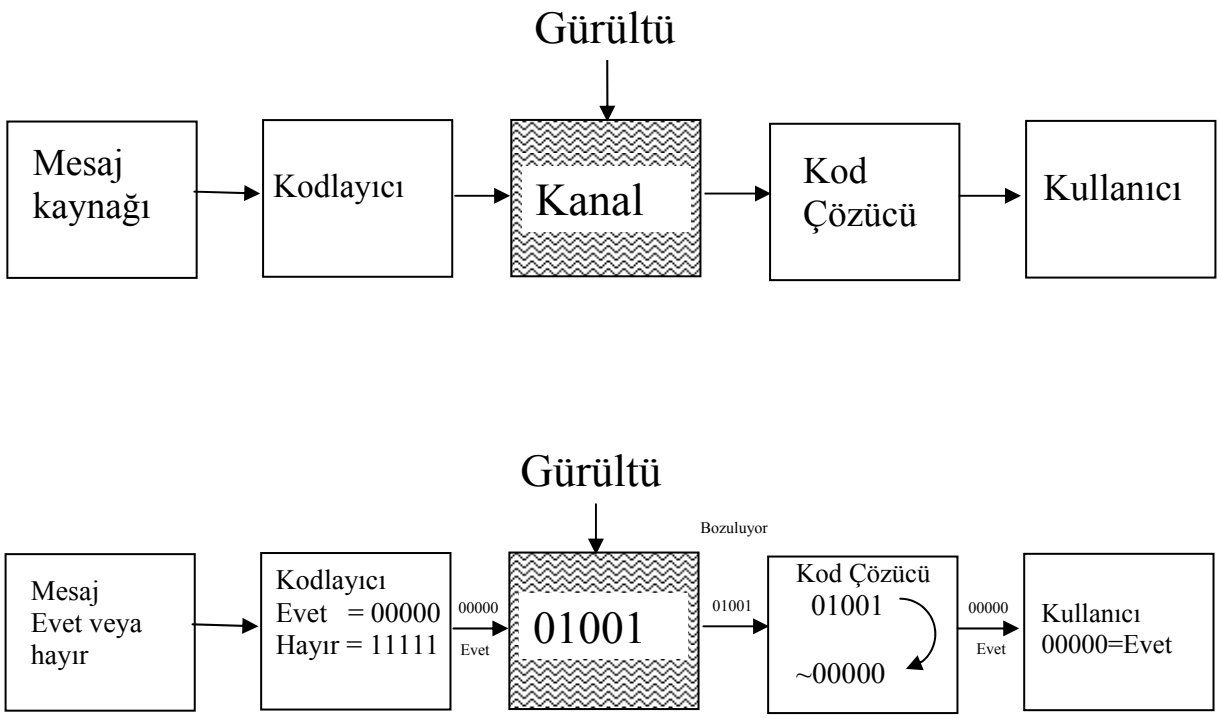
2.4.14. Teorem: $u, v \in C$ olsun. u ve v vektörlerinin aynı kalan sınıfında bulunması için gerek ve yeter şart bu vektörlerin aynı sendroma sahip olmalarıdır.

İspat: u ve v aynı kalan sınıfında bulunsunlar,

$$\Leftrightarrow u + C = v + C$$
$$\Leftrightarrow u - v \in C$$
$$\Leftrightarrow (u - v) H^T = 0$$
$$\Leftrightarrow uH^T = vH^T$$
$$\Leftrightarrow S(u) = S(v)$$

bulunur.

2.5. Genel Bir Dijital İletişim Sistemi



Şekil 2.1. Genel bir dijital iletişim sistemi

3. MATERYAL ve YÖNTEM

3.1. Lineer Kodlarla Kodlama İşlemi

$GF(q)$ üzerinde üreteç matrisi G olan bir $[n,k]$- C kodu alalım. Burada q^k adet farklı mesaj iletmek mümkündür. Mesajları $V(k,q)$ uzayının q^k tane sıralı k-lıları ile belirtilebilir. Burada,

$$u = u_1 u_2 u_3 ... u_k \tag{3.1}$$

mesaj vektörü

$$u \cdot G = \sum_{i=1}^{k} u_i r_i \tag{3.2}$$

şeklinde kodlanır. Burada r_i ler G nin satırlarıdır. Ayrıca $u \cdot G$ nin C nin bir kod sözcüğü olduğu da görülebilir.

$$u \to u \cdot G \tag{3.3}$$

kodlaması $V(n,q)$ nun k-boyutlu $V(k,q)$ alt uzayına bir tasviridir. Eğer G standart formda ise kodlama işlemi çok kolay bir şekilde yapılabilir. Örneğin,

$$G = [I_k \mid A]$$

olsun. Burada A bir $k \times (n-k)$ boyutlu bir matristir. u mesaj vektörü,

$$u = u_1 u_2 u_3 ... u_k$$

şeklindedir. Kodlama işlemi,

$$x = u \cdot G = x_1 x_2 x_3 ... x_k x_{k+1} ... x_n \tag{3.4}$$

şeklinde yapılır. Burada $1 \le i \le k$ için $x_i = u_i$ olur. Diğer yandan, $1 \le i \le n-k$ için,

$$x_{k+i} = \sum_{j=1}^{k} a_{ji} u_j \tag{3.5}$$

şeklindedir. Burada x_{k+i} ler kontrol sembolleridir. Kontrol sembolleri mesajı korumak için eklenen fazla sembollerdir. Örneğin, Binary $[7,4]$ kodun üreteç matrisi,

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

şeklindedir. Bu üreteç matrisinin standart formu ise,

$$G=\left[\begin{array}{cccc:ccc}1&0&0&0&1&0&1\\0&1&0&0&1&1&1\\0&0&1&0&1&1&0\\0&0&0&1&0&1&1\end{array}\right]$$

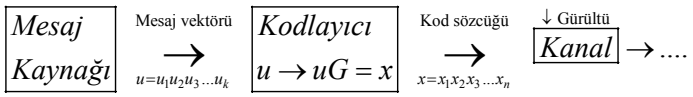
şeklinde bulunur. Burada,

$$u=\left(u_1u_2u_3u_4\right)$$

mesaj vektörü ele alınsın. Bu vektör $u\cdot G$ işlemi ile,

$$\left(u_1,u_2,u_3,u_4,u_1+u_2+u_3,u_2+u_3+u_4,u_1+u_2+u_4\right) \tag{3.6}$$

şeklinde kodlanır.



Şekil 3.1. Lineer kodlama işlemi

3.2. Sydrome (Sendrom) Çözüm

C bir $[n,k]$ kod olsun. H da C nin bir parity-kontrol matrisi olsun. $y\in V(n,q)$ olmak üzere,

$$S(y)=yH^T \tag{3.7}$$

$1\times(n-k)$ boyutlu bu satır vektörüne y nin sendromu denir. H parity kontrol matrisinin satırları,

$$h_1,h_2,h_3,\ldots,h_{n-k} \tag{3.8}$$

olarak verilmiş ise,

$$S(y) = \left(yh_1, yh_2, yh_3, ..., yh_{n-k} \right) \tag{3.9}$$

şeklinde olur. Aynı zamanda,

$$S(y) = 0 \Leftrightarrow y \in C \tag{3.10}$$

olması demektir.

2.4.14. Teoremin sonucu olarak sendromlar ve kosetler arasında bire-bir eşleme vardır diyebiliriz. Standart tablo oluşturarak kod çözümünde eğer n küçükse alınan y vektörünü tabloya yerleştirmek kolaydır. Fakat n büyük olduğu zaman y yi hangi kalan sınıfında olduğunu bulmak için sendromu kullanmak oldukça zaman alır. Bunu aşağıdaki gibi gerçekleştirebiliriz. Her bir e koset öncüsü için $S(e)$ sendromu hesaplayıp bunları önceden oluşturduğumuz standart tabloya sütun olarak eklememiz gerekecektir.

4. ARAŞTIRMA BULGULARI

Buraya kadar kodlama teorisinin temel tanımlarını ve teoremleri ile birlikte tek hata düzelten sendrom algoritmasını verdik. Şimdiye kadar bütün hesaplamaları el ile yaptık. Fakat kodların uzunlukları değiştikçe ve aralarındaki mesafeler arttıkça bunların çözümünü el ile yapmak oldukça zordur. Hatta bu bazı durumlarda mümkün değildir. Bunun için bu konuyu bilgisayar ortamına taşıyıp hesaplamalarımızı orada yaptık. Hesaplamalar için MAPLE bilgisayar programını kullandık.

4.1. Test Problemi

Üreteç matrisi,

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}_{2\times 4} \tag{4.1}$$

ve $k = 2, n = 4$ olan bir C kodunu ele alalım. Bunun için parity-kontrol matrisi,

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \tag{4.2}$$

olarak buluruz. Koset liderlerinin sendromları,

$$\left.\begin{array}{l} S(0000) = (0000)H^T = (00), S(1000) = (1000)H^T = (11) \\ S(0100) = (0100)H^T = (01), S(0010) = (0010)H^T = (10) \end{array}\right\} \tag{4.3}$$

şeklinde olur. Bu durumda standart tablo aşağıdaki çizelgedeki gibidir.

Çizelge 4.1. Standart tablo

Koset Liderleri				Sendromlar
0000	1011	0101	1110	00
1000	0011	1101	0110	11
0100	1111	0001	1010	01
0010	1011	0111	1100	10

Bu tablodan faydalanarak kod çözümü aşağıdaki gibi yapılır:

u vektörü alındığında, $S(u)$ hesaplanır ve sendrom sütununa yerleştirilir. Buna karşılık gelen satırda u bulunur ve bulunduğu yerin en üstündeki sözcük çözülmüş kod sözcüğü olur. Örneğin, $u = 0111$ alalım. Bu durumda,

$$S(0111) = (0111)\begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = (10) \tag{4.4}$$

olduğundan çözülmüş kod sözcüğü 0101 olur. Eğer standart tablo çözümü için bilgisayar kullanıyorsak tüm tabloyu girmek yerine bilgisayarımıza koset liderlerini ve bunlara ait sendromları girmemiz yeterli olacaktır. Örneğin sendromlar v, koset öncüleri $f(v)$ ile gösterilirse, aşağıdaki çizelge ele edilir.

Çizelge 4.2. Sendrom ve koset liderlerinin tablosu

Sendromlar (v)		Koset Öncüleri $f(v)$
00		0000
11		1000
01		0100
10		0010

Bu durumda kod çözme işlemi aşağıdaki gibi yapılmalıdır.

1. Adım: Alınan u vektörü için $S(u) = uH^T$ sendromu hesaplanır.

2. Adım: $v = S(u)$ alınır. v birinci sütunda buluruz. Örneğin $v = 10$, $f(v) = 0010$.

3. Adım: u vektörünü $u - f(v)$ şeklinde çözeriz.

Buna göre 0111 kod sözcüğü, 0111-0010 = 0101 şeklinde çözülür.

4.2. Test Problemi

Üreteç matrisi

$$G=\begin{bmatrix}1 & 0 & 1 & 1\\ 0 & 1 & 0 & 1\end{bmatrix}_{2\times 4}$$

ve $k=2,\ n=4$ olan bir C kodunu ele alalım. Bunun için parity-kontrol matrisi

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

olarak buluruz. Koset liderlerinin sendromları,

$$\left.\begin{array}{l}S(0000)=(0000)H^T=(00), S(1000)=(1000)H^T=(11)\\ S(0100)=(0100)H^T=(01), S(0010)=(0010)H^T=(10)\end{array}\right\}$$

şeklindedir. Buna göre standart tablo aşağıdaki çizelgedeki gibi olur.

Çizelge 4.3. Standart tablo

Koset Liderleri				Sendromlar
0000	1011	**0101**	1110	00
1000	0011	**1101**	0110	**11**
0100	1111	0001	1010	01
0010	1011	0111	1100	10

Bu sefer $u=1101$ alırsak,

$$S(1101)=(1101)\begin{bmatrix}1 & 1\\ 0 & 1\\ 1 & 0\\ 0 & 1\end{bmatrix}=(11)$$

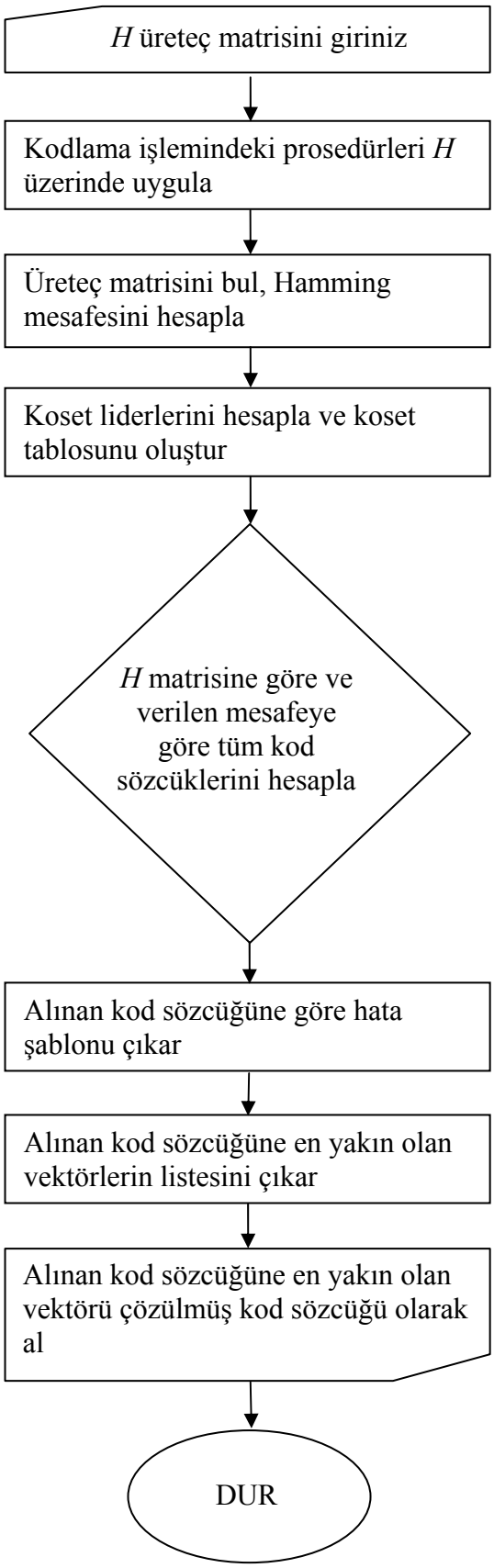
olduğundan çözülmüş kod sözcüğü 0101 olur.

Çizelge 4.4. Sendrom ve koset liderlerinin tablosu

Sendromlar (v)		Koset Öncüleri $f(v)$
00		0000
11		1000
01		0100
10		0010

Buna göre 1101 kod sözcüğünü 1101-1000 = 0101 olarak çözeriz.

4.3. Kod Çözme Algoritması



4.4. Programdan Elde Ettiğimiz Sonuç

```
H2:=mat(["1010","1101"]);
```

$$H2 := \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

```
C2:=KOD(H2);
```

```
HAMMINGMESAFESI(C2);
Bu kodun hamming mesafesi 2
```

```
KODADONUSTUR(C2);
```

$$\{[1, 1, 0, 1, 1], [0, 0, 1, 1, 1], [1, 0, 1, 0, 1], [1, 1, 1, 0, 0]\}$$

```
KOSETTABLOSU1(H2);
                    {0000, 1011, 1110, 0101}
                    {0110, 0011, 1000, 1101}
                    {0111, 0010, 1100, 1001}
                    {0001, 1111, 0100, 1010}
```

```
KOSETTABLOSU2(H2);
                   Sendrom     Koset Liderleri
                     11              1000
                     10              0010
                     00              0000
                     01              0001
```

```
COZUMTABLOSU1(C2);
```

```
          KOD KUMESI  C2={0101, 0000, 1011, 1110}
```

Çizelge 4.5. Kod çözüm tablosu (*) 4.1. Test problemi, (**) 4.2. Test Problemi

Alınan Sözcük	**Hata Şablonu**	**Düzeltilmiş Sözcük**
1101 (**)	**[1101, 0110, 1000, 0011]**	**0101**
0000	[0000, 1011, 0101, 1110]	0000
0111 (*)	**[0111, 1100, 0010, 1001]**	**0101**
0001	[0001, 1010, 0100, 1111]	0000
1011	[1011, 0000, 1110, 0101]	1011
0110	[0110, 1101, 0011, 1000]	1110
0011	[0011, 1000, 0110, 1101]	1011
1010	[1010, 0001, 1111, 0100]	1011
0101	[0101, 1110, 0000, 1011]	0101
1111	[1111, 0100, 1010, 0001]	1011
1100	[1100, 0111, 1001, 0010]	1110
1001	[1001, 0010, 1100, 0111]	1011
1110	[1110, 0101, 1011, 0000]	1110
1000	[1000, 0011, 1101, 0110]	0000
0100	[0100, 1111, 0001, 1010]	0000
0010	[0010, 1001, 0111, 1100]	0000

5. TARTIŞMA ve SONUÇ

Bu çalışmada iletişim çağının temel yapıtaşı olan kodlama teorisinin yöntemlerinden tek hata düzelten sendrom algoritmasının bilgisayar uygulamasını yaptık. Bu uygulama verilen parity kotrol matrisine göre kod üretir. Bu koda ait Hamming mesafesi, elemanlarına ait kosetler, koset liderlerini hesaplayıp koset liderlerinin standart dizilişini bulur. Üretilen kodun ait olduğu en geniş kod kümesini hesaplayıp her bir kodun sözcüğünün hatalı alınmasına ilişkin hata şablonu ve en yakın kod sözcüğünün bir listesini verir. Hesaplamalar için MAPLE bilgisayar programını kullandık. Verdiğimiz algoritma, verilen üreteç matrisine değişik sonuçlar üretir. Burada biz kod sözcüğü uzunluğu 4 olan bir test problemi seçtik. Fakat daha uzun kodlar için hesaplama yapmakda mümkündür. Algoritma, bilgisayara girilen keyfi uzunluktaki kodun parity kontrol matrisine göre çözümleme işlemi yapar.

KAYNAKLAR

Bose R. C., Ray-Chaudhuri D. K., 1960, On a class of error-correcting binary group codes, *Info and Control* 3, 68-79.

Goppa V. D., 1970, A new class of linear error-correcting codes. *Problems of info. Transmission* 6(3), 207-12.

Gilbert E. N., 1952, A comparision of signalling alphabets. *Bell Syst. Tech.J.* 31, 504-22.

Golay M. J. E., 1949, Notes on digital coding. *Proc. IEEE* 37, 657.

Hall J. I., 2003, Notes on coding theory. Department of Mathematics, Michigan State University.

Hill R., 1986, A first course in coding theory, Clarendon Press, Oxford.

Huffman W. C. and Pless V., 2003, Fundamental of Coding Theory, Cambridge University Press.

Hamming R. W., 1950, Error detecting and error-correcting codes. *Bell Syst. T.J.* 29, 147-60

Lemmermeyer F., 2005, Error-correcting codes.

Morandi P. J., 2001, Error-correcting codes and algebraic curves, New Mexico State University, http://emmy.nmsu.edu/~pmorandi/math601f01/LectureNotes.pdf (15.12.2005)

Monagan M. B., Geddes K. O., Heal K. M., Labahn G., Vorkoetter S. M., McCarron J. and DeMarco P., 2003, Maple 9 Advanced Programming Guide. *Maplesoft, a division of Waterloo Maple Inc.*

Nordstrom A. W. and Robinson J. P., 1967, An optimum nonlinear code. *Info. and Control* 11, 613-16.

Shannon C. E., 1948, A mathematical theory of communication. *Bell Syst. T.J.* 27, 379-423.

Varshamov R. R., 1957, Estimate of the number of signals in error-correcting codes. *Dokl. Nauk SSSR* 117, 739-41.

Slepian D., 1960, Some further theory of group codes. *Bell Syst. Tech.J.* 39, 1219-52.

EKLER

EK 1 (Kod Çözme Programı)

1.*Özellik:*

AD = PROC (Değişken(=Matris, vektör, sayı, karakter vs.))

Prosedür, belirlediğimiz bir isme atama işlemi yapan hazır bir MAPLE fonksiyonudur.

2.*Metot:*

Metot (3.2) de verilen Sendrom çözüm algoritmasına ve kodlama işleminde verdiğimiz yöntemlere dayanır.

3.*Kod Çözme Programında Kullandığımız Prosedürler:*

VECTOR2STRING: Bir vektörü string formatına çevirir.
STRINGVECTORSET: String formatına dönüştürülmüş vektörlerin kümesini verir.
KISAYAZ: STRINGVECTORSET kümesinin kısaca yazılışını verir.
LISTEYECEVIR: Bir vektörün bileşenlerini listeye çevirir.
LISTEYECEVIR2: Bir vektör yada string'i listeye çevirir.
KODADONUSTUR: String formatındaki sözcükleri kod kümesi olarak yazar.
AGIRLIK: Bir vektörün ağırlığını hesaplar.
HAMMINGUZAKLIGI: Bir kodun Haming uzaklığını hesaplar.
KOSET1: Bir kodun kosetini bulur.
KOSET2: Bir vektöre ait koseti bulur.
TUMKOSET: Bir kodun bütün kosetlerini bulur (Slepian Standart Dizilişi).
KOSETTABLOSU: Sendrom çözüm için koset tablosunu oluşturur.
URETECMATRISI: Verilen parity kontrol matrisinden üreteç matrisini hesaplar.
PARITYKONROL: Verilen bir kodun parity kontrol matrisini hesaplar.
COZUMTABLOSU1-2: Verilen parity kontrol matrisine göre üretilen kodlarda alınan vektörleri, bu vektörlerin hatalı alınması durumunda oluşabilecek hata şablonları ve gerçek vektöre en yakın kod sözcüklerinin listesini verir.

HATADUZELT: Alınan hatalı vektörü, çözüm tablosuna bakarak en yakın olan vektörle değiştirip hatayı düzeltir.

GEREN1-2: Verilen vektör uzayının gerenlerini hesaplar.

TABAN1-2: Verilen vektör uzayının tabanını bulur.

ORTOGONAL1-2 : Verilen vektör uzayının ortogonal alt uzayını bulur.

ICCARPIM: İki vektörün iç çarpımlarını hesaplar.

MATKISAYAZ: Kısaca matris girmemizi sağlar.

```
with(linalg):

VECTOR2STRING := proc(vv) local l,i,v,stg:
v := convert(vv,list):
l := nops(v):
stg := "":
for i from 1 to l do
stg := cat(stg,convert(v[i],string)):
od:
stg:
end:

STRINGVECTORSET := proc(S) local i,T:
T := {}:
for i from 1 to nops(S) do
T := T union {VECTOR2STRING(S[i])}:
od:
T:
end:

KISAYAZ := proc(S) local i:
if nops(S) = 0 then printf("{ }") fi:
if nops(S) = 1 then printf("%A",S)
else
for i from 1 to nops(S) do
  if i = 1 then printf("{%s, ",S[1])
  else if i < nops(S) then printf("%s, ",S[i]) else
printf("%s}",S[i]) fi:
  fi:
  od:
fi:
end:

LISTEYECEVIR := proc(S) local i,T:
T := {}:
```

```
for i from 1 to nops(S) do
  T := T union {convert(S[i],list)}:
od:
T:
end:
LISTEYECEVIR2 := proc(v) local i,n,w:
if type(v,string) = true then
  n := length(v):
  w := array(1..n):
  for i from 1 to n do
    if v[i] = "0" then w[i] := 0 else w[i] := 1 fi:
  od:
  convert(w,list):
else
  convert(v,list):
fi:
end:

KODADONUSTUR := proc(S) local i,t,T:
t := nops(S):
if t = 0 then T := {}
else
  T := {}:
  for i from 1 to t do
    T := T union {LISTEYECEVIR2(S[i])}:
  od:
fi:
T:
end:

AGIRLIK := proc(v) local i,n,w:
if type(v,list) = true then w := v else w :=
LISTEYECEVIR2(v) fi:
n := nops(convert(w,list)):
sum(w[i],i=1..n);
end:

HAMMINGUZAKLIGI := proc(H) local n,i,T,H2,w,zero;
if type(H,matrix) = true then
  n := nops(convert(row(H,1),list)):
  zero := convert(vector(n,0),list):
  T := KOSET2(H,zero) minus {zero}:
  w := AGIRLIK(T[1]):
else
  if type(H[1],string) = false then H2 := H else H2 :=
GEREN2(H) fi:
  n := nops(H2[1]):
  zero := convert(vector(n,0),list):
```

```
  T := H2 minus {zero}:
  w := AGIRLIK(T[1]):
fi:
for i from 2 to nops(T) do
  if w > AGIRLIK(T[i]) then w := AGIRLIK(T[i]): fi:
od:
printf("Bu kodun Hamming uzaklığı %a",w):
end:

KOSET1 := proc(H,v) local i,j,k,t,w,C,S,T:
if type(H,matrix) = true then
  C := Nullspace(H) mod 2:
else
  C := TABAN2(H):
fi:
k := nops(C):
if type(v,string) = false then w := convert(v,list) else w
:= LISTEYECEVIR2(v) fi:
T := {w}:
t := 1:
for i from 1 to k do
  S := T:
  for j from 1 to t do
    w := evalm(C[i] + T[j]):
    w := map(`mod`,w,2):
    w := convert(w,list):
    S := S union {w}:
  od:
  T := T union S:
  t := 2*t:
od:
KISAYAZ(STRINGVECTORSET(T)):
end:

KOD := proc(H) local i,n:
n := nops(convert(row(H,1),list)):
KOSET1(H,vector(n,0)):
end:

An := proc(n):
KOSET2(matrix(n,n,0),vector(n,0)):
end:

KOSET2 := proc(H,v) local i,j,k,t,w,C,S,T:
if type(H,matrix) = true then C := Nullspace(H) mod 2: else
C := TABAN2(H) fi:
k := nops(C):
```

```
if type(v,string) = false then w := convert(v,list) else w
:= LISTEYECEVIR2(v) fi:
T := {w}:
t := 1:
for i from 1 to k do
  S := T:
  for j from 1 to t do
    w := evalm(C[i] + T[j]):
    w := map(`mod`,w,2):
    w := convert(w,list):
    S := S union {w}:
  od:
  T := T union S:
  t := 2*t:
od:
T:
end:

TUMKOSET := proc(H) local k,l,n,s,v,C,R,T,TT:
if type(H,matrix) = true then C := Nullspace(H) mod 2: else
C := TABAN2(H) fi:
k := nops(C):
n := nops(convert(C[1],list)):
s := 2^(n-k):
R := KOSET2(H,vector(n,0)):
TT[1] := R:
KISAYAZ(STRINGVECTORSET(R)):
printf("\n\n"):
l := 1:
while l < s do
  v := randvector(n):
  v := map(`mod`,v,2):
  v := convert(v,list):
  if(member(v,R)=false) then
    T := KOSET2(H,v):
    l := l+1:
    TT[l] := T:
    R := R union T:
    KISAYAZ(STRINGVECTORSET(T)):
    printf("\n\n"):
  fi:
od:
end:

KOSETTABLOSU := proc(H) local
i,l,k,n,s,v,w,C,R,T,weight,xx,yy;
C := Nullspace(H) mod 2:
k := nops(C):
```

```
n := nops(convert(C[1],list)):
s := 2^(n-k):
R := {}:
l := 0:
printf("%-15A %-15A\n","SENDROM","KOSET LİDERİ"):
while l < s do
v := randvector(n):
v := map(`mod`,v,2):
v := convert(v,list):
if member(v,R) = false then
T := KOSET2(H,v):
R := R union T:
l := l+1:
w := T[1]:
weight := AGIRLIK(w):
for i from 1 to 2^k do
if AGIRLIK(T[i]) < AGIRLIK(w) then w := T[i]: weight :=
AGIRLIK(w): fi:
od:
xx := convert(map(`mod`,evalm(H&*w),2),list):
yy := convert(w,list):
printf(" %-15A %-
15A\n",VECTOR2STRING(xx),VECTOR2STRING(yy)):
fi:
od:
end:

URETECMATRISI := proc(H) local k,n,B,C,G:
if type(H,matrix) = true then
  C := Nullspace(H) mod 2:
  G:=matrix(nops(C),nops(convert(C[1],list)),(i,j)-
>C[i][j]):
  evalm(G):
else
  B:=TABAN2(H):
  k:=nops(B):
  n:=nops(B[1]):
  G:=matrix(k,n,(i,j)->B[i][j]):
  evalm(G):
fi:
end:

PARITYKONTROL:=proc(G):
if type(G,matrix) = true then URETECMATRISI(G): else
URETECMATRISI(ORTOGONAL2(G)) fi:
end:
```

```
COZUMTABLOSU1 := proc(H) local i,j,v,e,n,E,H2,zero: global
code,words,corr:
if type(H,matrix) = true then
  n := nops(convert(row(H,1),list)):
  zero := convert(vector(n,0),list):
  code := KOSET2(H,zero):
  printf("The code is "):
KOD(H):
else
  if type(H[1],string) = false then H2 := H else H2 :=
KODADONUSTUR(H) fi:
  n := nops(H2[1]):
  code := {}:
  for i from 1 to nops(H2) do
    code := code union {H2[i]}:
  od:
  printf("Kod Kümesi"):
  KISAYAZ(STRINGVECTORSET(code)):
fi:
printf("\n\n"):
words := An(n):
printf("%-20A %-35A %-20A\n","ALINAN SÖZCÜK", "HATA
ŞABLONU", "ENYAKIN KOD SÖZCÜĞÜ"):
for i from 1 to nops(words) do
  e := map(`mod`, words[i] + code[1], 2):
  E := [VECTOR2STRING(e)]:
  for j from 2 to nops(code) do
    v := map(`mod`, words[i] + code[j], 2):
    E := [op(E),VECTOR2STRING(v)]:
   if AGIRLIK(v) < AGIRLIK(e) then e := v fi:
  od:
  corr[i] := map(`mod`, words[i] + e, 2):
  printf(" %-20A %-35A %-20A\n", VECTOR2STRING(words[i]),
E, VECTOR2STRING(corr[i])):
od:
end:

COZUMTABLOSU2 := proc(H) local
i,j,v,e,n,t,E,H2,blank,code,words,corr,zero:
if type(H,matrix) = true then
  n := nops(convert(row(H,1),list)):
  zero := convert(vector(n,0),list):
  code := KOSET2(H,zero):
  printf("Kod Kümesi"):
  KOD(H):
else
  if type(H[1],string) = false then H2:=H else
H2:=KODADONUSTUR(H) fi:
```

```
  n := nops(H2[1]):
  code := {}:
  for i from 1 to nops(H2) do
    code := code union {H2[i]}:
  od:
  printf("Kod Kümesi "):
  KISAYAZ(STRINGVECTORSET(code)):
fi:
blank := "":
for i from 1 to n do
  blank := cat(blank,"-"):
od:
printf("\n\n"):
words := An(n):
printf("%-20A %-35A %-20A\n", "ALINAN SÖZCÜK", "HATA
ŞABLONU", "ENYAKIN KOD SÖZCÜĞÜ"):
for i from 1 to nops(words) do
  e := map(`mod`, words[i] + code[1], 2):
  E := [VECTOR2STRING(e)]:
  for j from 2 to nops(code) do
    v := map(`mod`, words[i] + code[j], 2):
    E := [op(E), VECTOR2STRING(v)]:
   if AGIRLIK(v) < AGIRLIK(e) then e := v fi:
  od:
  corr[i] := map(`mod`, words[i] + e, 2):
  t := 0:
  for j from 1 to nops(code) do
    if AGIRLIK(map(`mod`, words[i] + code[j], 2)) =
AGIRLIK(e) then t := t+1 fi:
  od:
  if t = 1 then
    printf(" %-20A %-35A %-20A\n", VECTOR2STRING(words[i]),
E, VECTOR2STRING(corr[i]))
  else printf(" %-20A %-35A %-20A\n",
VECTOR2STRING(words[i]), E, blank):
  fi:
od:
end:

HATADUZELT := proc(v) local i,w:
if type(v,string) = false then w := convert(v,list) else w
:= LISTEYECEVIR2(v) fi:
for i from 1 to nops(words) do
  if AGIRLIK(map(`mod`,words[i]+w,2)) = 0 then
    printf("The best decoding of %A is
%A",VECTOR2STRING(words[i]),VECTOR2STRING(corr[i])): break:
fi:
od:
```

```
end:

GEREN1 := proc(S) local i,j,n,zero,T,U,S2:
if nops(S) = 0 then printf("{0}") else
if type(S[1],string) = false then S2 := S else S2 :=
KODADONUSTUR(S) fi:
n := nops(S2[1]):
zero := convert(vector(n,0),list):
T := {zero,S2[1]}:
for i from 2 to nops(S2) do
  U := {}:
  for j from 1 to nops(T) do
    if member(S2[i],T) = false then U := U union
{map(`mod`,T[j]+S2[i],2)} fi:
  od:
T := T union U:
od:
KISAYAZ(STRINGVECTORSET(T)):
fi:
end:

GEREN2 := proc(S) local i,j,n,zero,T,U,S2:
if nops(S) = 0 then T:={0} else
if type(S[1],string) = false then S2 := S else S2 :=
KODADONUSTUR(S) fi:
n := nops(S2[1]):
zero := convert(vector(n,0),list):
T := {zero,S2[1]}:
for i from 2 to nops(S2) do
  U := {}:
  for j from 1 to nops(T) do
    if member(S2[i],T) = false then U := U union
{map(`mod`,T[j]+S2[i],2)} fi:
  od:
T := T union U:
od:
T:
fi:
end:

TABAN1 := proc(C) local i,n,B,S,C2,zero,card:
if type(C[1],string) = false then C2 := C else C2 :=
KODADONUSTUR(C) fi:
card := nops(C2):
n := nops(C2[1]):
if card = 1 then if AGIRLIK(C2[1]) = 0 then B := {} else B
:= {C2[1]} fi:
else
```

```
  zero := convert(vector(n,0),list):
  if C2[1] = zero then B := {C2[2]} else B := {C2[1]} fi:
  S := GEREN2(B):
  for i from 1 to card do
    if member(C2[i],S) = false then B := B union {C2[i]}: S
:= GEREN2(B): fi:
  od:
fi:
if card = 1 then KISAYAZ(STRINGVECTORSET(B))
else
  KISAYAZ(STRINGVECTORSET(B)):
fi:
end:

TABAN2 := proc(C) local i,n,B,S,C2,zero,card:
if type(C[1],string) = false then C2 := C else C2 :=
KODADONUSTUR(C) fi:
card := nops(C2):
n := nops(C2[1]):
if card = 1 then if AGIRLIK(C2[1]) = 0 then B := {} else B
:= {C2[1]} fi:
else
  zero := convert(vector(n,0),list):
  if C2[1] = zero then B := {C2[2]} else B := {C2[1]} fi:
  S := GEREN2(B):
  for i from 1 to card do
    if member(C2[i],S) = false then B := B union {C2[i]}: S
:= GEREN2(B): fi:
  od:
fi:
B:
end:

ORTOGONAL1 := proc(H) local i,k,n,C,C2,G,H2,T:
if type(H,matrix) = true then
  G := URETECMATRISI(H):
else
  if type(H[1],string) = false then H2 := H else H2 :=
KODADONUSTUR(H) fi:
  C := TABAN2(H2):
  k := nops(C):
  n := nops(C[1]):
  G := matrix(k,n,(i,j)->C[i][j]):
fi:
C2 := Nullspace(G) mod 2:
T := {}:
if nops(C2) > 0 then
  for i from 1 to nops(C2) do
```

```
    T := T union {convert(C2[i],list)}:
  od:
fi:
GEREN1(T):
end:

ORTOGONAL2 := proc(H) local i,k,n,C,C2,G,H2,T:
if type(H,matrix) = true then
  G := URETECMATRISI(H):
else
  if type(H[1],string) = false then H2 := H else H2 :=
KODADONUSTUR(H) fi:
  C := TABAN2(H2):
  k := nops(C):
  n := nops(C[1]):
  G := matrix(k,n,(i,j)->C[i][j]):
fi:
C2 := Nullspace(G) mod 2:
T := {}:
if nops(C2) > 0 then
  for i from 1 to nops(C2) do
    T := T union {convert(C2[i],list)}:
  od:
fi:
GEREN2(T):
end:

ICCARPIM := proc(u,v) local i,n,u2,v2,sum:
if type(u,string) = false then u2 := u else u2 :=
LISTEYECEVIR2(u) fi:
if type(v,string) = false then v2 := v else v2 :=
LISTEYECEVIR2(v) fi:
n := nops(convert(u2,list)):
sum := 0:
for i from 1 to n do
 sum := sum + u2[i] * v2[i]:
od:
sum mod 2:
end:

MATKISAYAZ := proc(C) local i,n,m,B,H:
n := nops(C):
for i from 1 to n do
B[i] := convert(LISTEYECEVIR2(C[i]),vector):
od:
m:=nops(convert(B[1],list)):
H:=matrix(n,m,(i,j)->B[i][j]):
end:
```

ÖZGEÇMİŞ

1979 yılında Rize'nin Çamlıhemşin ilçesinde doğdu. İlk, orta ve lise öğrenimini Çamlıhemşinde tamamladı. 1997 yılında Atatürk Üniversitesi Fen-Edebiyat Fakültesi Matematik Bölümünü kazandı. 2001 yılında aynı bölümden mezun oldu. 2001- 2003 yılları arasında matematik öğretmenliği yaptı.

2002 yılı sonunda Bayburt Meslek Yüksekokulu Teknik Programlar Bölümünde araştırma görevlisi olarak göreve başladı. 2003 yılında Atatürk Üniversitesi Fen-Edebiyat Fakültesi Matematik Bölümünde yüksek lisans öğrenimine başladı.

2005 yılında Atatürk Üniversitesi Fen-Edebiyat Fakültesi Matematik Bölümüne görevlendirildi. Halen bu görevine devam etmektedir.