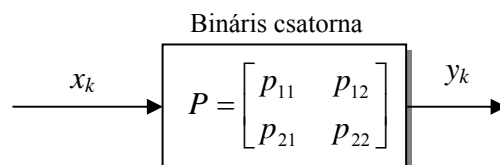


2. Hibajavító kódolás

2.1. A bináris csatorna

Ebben a fejezetben hibajavító kódolással kívánunk foglalkozni és ennek érdekében elsőként vizsgáljuk meg a hibák kialakulásának természetét.



A bináris csatorna bemenetére érkező x_k sorozat tagjai legyenek egymástól függetlenek. Ebben a sorozatban jelöljük P_{x0} -val a 0-ás és P_{x1} -el az 1-es értékű szimbólumok előfordulásának valószínűségét.

$$P_{x0} = \Pr\{x_k = 0\} \text{ és } P_{x1} = \Pr\{x_k = 1\} \quad (2.1.)$$

Feltételezzük, hogy a csatorna y_k kimenete csak x_k -től és a véletlenszerű zajtól függ (de nem függ pld az x_{k-1} -től). Az ilyen csatornát nevezzük **memória-mentes** csatornának. A zaj következtében előforduló tévesztéseket véletlen változó eseményeknek tekintjük. Jelöljük h_0 -val annak az eseménynek a valószínűségét, amikor $x_k = 0$ és $y_k = 1$ ill. h_1 -el mikor $x_k = 1$ és $y_k = 0$.

A bináris csatornát a hibák szempontjából ezek után a P feltételes valószínűségeket tartalmazó hiba-mátrixszal jellemezhetjük:

$$P = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} = \begin{bmatrix} 1-h_0 & h_1 \\ h_0 & 1-h_1 \end{bmatrix} \quad (2.2.)$$

ahol:

$$\begin{aligned} p_{11} &= \Pr\{y_k = 0 | x_k = 0\} = 1 - h_0 \\ p_{12} &= \Pr\{y_k = 0 | x_k = 1\} = h_1 \\ p_{21} &= \Pr\{y_k = 1 | x_k = 0\} = h_0 \\ p_{22} &= \Pr\{y_k = 1 | x_k = 1\} = 1 - h_1 \end{aligned} \quad (2.3.)$$

A kimenet eloszlása a bemenet eloszlásának és a hibamátrixnak az ismeretében számolható:

$$\begin{aligned} P_{y0} &= \Pr\{y_k=0\} = \Pr\{y_k=0 \text{ és } x_k=0\} + \Pr\{y_k=0 \text{ és } x_k=1\} = \\ &= \Pr\{y_k=0 | x_k=0\} P_{x0} + \Pr\{y_k=0 | x_k=1\} P_{x1} = \\ &= p_{11} P_{x0} + p_{12} P_{x1} = (1-h_0) P_{x0} + h_1 P_{x1} \end{aligned} \quad (2.4.)$$

Illetve:

$$P_{y1} = \Pr\{y_k=1\} = \dots = h_0 P_{x0} + (1-h_1) P_{x1} \quad (2.5.)$$

Az egy bit hibás átvitelének valószínűsége:

$$\begin{aligned}
 p_{berr} &= \Pr\{y_k=1 \text{ és } x_k=0\} + \Pr\{y_k=0 \text{ és } x_k=1\} = \\
 &= \Pr\{y_k=1 \mid x_k=0\}P_{x0} + \Pr\{y_k=0 \mid x_k=1\}P_{x1} = \\
 &= p_{21}P_{x0} + p_{12}P_{x1} = h_0P_{x0} + h_1P_{x1}
 \end{aligned} \tag{2.6.}$$

Ha a bemenet eloszlása egyenletes ($P_{x0} = P_{x1} = 0.5$):

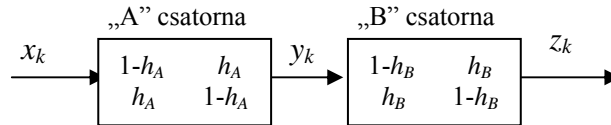
$$p_{berr} = \frac{h_0 + h_1}{2} \tag{2.7.}$$

Szimmetrikus bináris csatornáról (BSC) akkor beszélünk, ha $h_0 = h_1 = h$, vagyis a csatornában egyformán valószínű a bitrontás iránya. Ekkor:

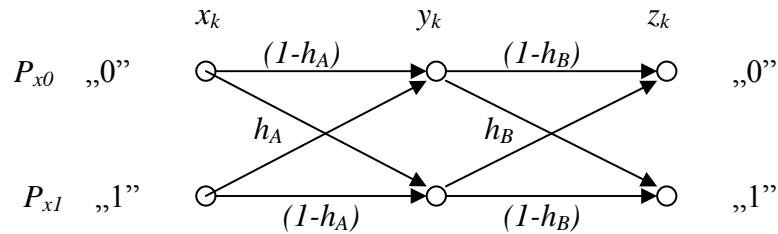
$$p_{berr} = h \tag{2.8.}$$

2.2. Bináris csatornák kaszkád kapcsolása

Kapcsoljunk egymás után két egymástól *független* BSC-t, melyek bithiba valószínűségei: h_A és h_B . Határozzuk meg az együttes hibavalószínűséget:



A választ legegyszerűbben a rendszer állapot-átmenet valószínűségeit mutató ábra segítségével kapjuk meg.



A bithiba valószínűsége:

$$\begin{aligned}
 p_{berr} &= [(1-h_A)h_B + h_A(1-h_B)]P_{x0} + [(1-h_A)h_B + h_A(1-h_B)]P_{x1} = \\
 &= [(1-h_A)h_B + h_A(1-h_B)][P_{x0} + P_{x1}] = (1-h_A)h_B + h_A(1-h_B)
 \end{aligned} \tag{2.9.}$$

Két egyforma csatornára ($h_A = h_B = h$):

$$p_{berr} = 2h(1-h) \tag{2.10.}$$

2.3. Hibajavító kódolás

Tételezzük fel, hogy sorozatokat (\mathbf{u}^i üzenet vektorokat) kívánunk átküldeni egy digitális csatornán, mely csatornában egy-egy bit átvitele $p_{berr} > 0$ valószínűséggel hibás. Ugyanakkor sok esetben nem megengedett, hogy az átvendő állományban egyetlen bit is hibás legyen. Kódolási eljárással hiba elleni védelmet kívánunk megvalósítani, azaz *automatizálni szeretnénk* a hiba felismerését és annak lehetőség szerinti kijavítását.

2.3.1. Ismétléses hibavédelem

A legegyszerűbb eljárás az lehet, hogy minden egyes bináris szimbólumot nem egyszer, hanem (páratlan számúszor) többször küldünk el (mondjuk: háromszor). A vételi oldalon ezután többségi döntőt használunk. Példánkban ha egy bit a háromból meghibásodik, akkor ki tudjuk javítani a hibát. Ha kettő is meghibásodik, észrevesszük a hibát, de rossz irányba javítunk. Három hiba esetén észre sem vesszük a hibát.

A többségi döntő használatát az indokolja, hogy memóriamentes csatornában a többszörös hibák valószínűsége a hibák számának függvényében rohamosan csökken. Annak a valószínűsége ugyanis, hogy N bitből k meghibásodik:

$$p_k = \binom{N}{k} p_{berr}^k (1 - p_{berr})^{N-k} = \frac{N!}{k!(N-k)!} p_{berr}^k (1 - p_{berr})^{N-k} \quad (2.11.)$$

Pld. Ha $N=5$ és $p_{berr} = 10^{-3}$ akkor:

$$\begin{aligned} p_0 &= 0.9860, & p_1 &= 4.980 \cdot 10^{-3}, & p_2 &= 9.997 \cdot 10^{-6}, \\ p_3 &= 9.998 \cdot 10^{-9}, & p_4 &= 4.995 \cdot 10^{-12}, & p_5 &= 10^{-15} \end{aligned}$$

Azt a döntési filozófiát, amikor a nagyobb valószínűségű esemény bekövetkeztének az irányában döntünk, az eredmény **maximum-likelihood** becslésnek nevezzük.

2.3.2. Egydimenziós paritás kód

Ebben az eljárásban a K bites üzeneteinkhez egyetlen bitet (az ún. paritás bitet) tesszük hozzá, mely bit a K darab üzenet bit modulo 2 szerinti összege. Ekkor beszélünk ún. páros paritásról, amikor is a $K+1$ bites kódszóban az 1-esek száma biztosan páros számú lesz. (A paritás bitet az összeg inverzének választva kapjuk a páratlan paritású kódot.) Ennek a kódnak kicsi a redundanciája, ugyanakkor a képessége is kicsi, legfeljebb észreveszi a hibát, de javítani nem tudja.

2.3.2. Kétdimenziós paritás kód

A kétdimenziós paritás kódot úgy képezzük, hogy a $K = I \cdot J$ darab üzenet bitet első lépésben egy I sorból és J oszlopból álló mátrixba írjuk. Ezután soronként majd oszloponként generáljuk a paritás biteket, melyeket a mátrix $(J+1)$ -ik oszlopába illetve $(I+1)$ -ik sorába írunk be. Majd képezzük a paritások paritását.

A vételi oldalon újra képezve a paritás biteket, feltárhatjuk illetve kijavíthatjuk a hibát. Az $(I+1) \cdot (J+1)$ bit átvitele során ha egy bit hibás, akkor pld. az i -ik sorban és

a j -ik oszlopban eltérés lesz az elküldött illetve az újrászámolt paritásbit értékekben. Ha egy hibát tételezünk fel, akkor ez csak úgy lehet, hogy a hibás bit az i -ik sor j -ik oszlopában helyezkedik el.

Két (vagy több) hiba esetén azt kell feltételeznünk, hogy a hibás bitek különböző sorokba és különböző oszlopokba esnek. Ekkor a hibát jelezni tudjuk, de kijavítani nem.

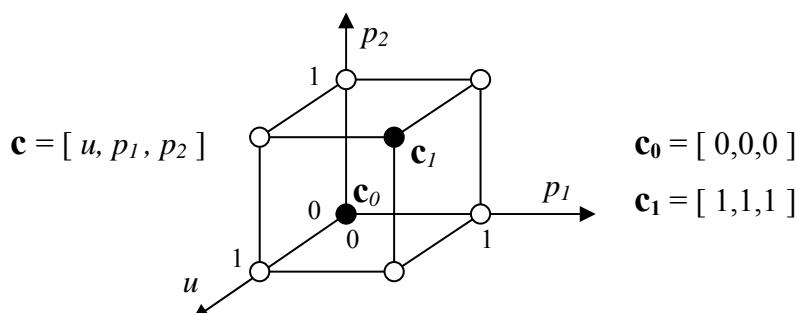
1	0	1	1	1
0	1	1	0	0
1	1	1	0	1
0	1	1	1	1
0	1	0	0	1

2.3.3. A redundancia és a hibajavító képesség kapcsolata

A mennyiségi tárgyalás érdekében rendezzük a K bit hosszúságú bináris üzenet sorozatot egy \mathbf{u}^i vektorba. Az összes lehetséges üzenet ekkor egy K dimenziós bináris teret feszít ki, melynek 2^K lehetséges pontja van. Minden pont egy üzenetnek felel meg. Ha csak az üzenetet továbbítjuk és abban hiba keletkezik, akkor ennek a térnek egy másik pontjába kerülünk, ami egy lehetséges másik üzenetet jelent. Ekkor persze nem vesszük észre, hogy a továbbítás során hiba történt.

A hiba felismerése érdekében a lehetséges pontok számát a térben növeljük meg oly módon, hogy a tér dimenzióját növeljük K -ról N -re. A K hosszúságú üzenet helyett az üzenethez hozzárendelt N hosszúságú $\mathbf{u}^i \Rightarrow \mathbf{c}^i$ kódot továbbítjuk a csatornán ($N > K$). A 2^N pontot tartalmazó kódtérben azonban csak 2^K ponthoz (a választott kódokhoz) tartozik üzenet. A növeléssel azt értük el, hogyha hiba van, akkor a tér egy olyan pontjába jutunk ami nem értelmezhető kódnak. Ez az eltérés alkalmas a **hiba felismerésére**.

Példa gyanánt tekintsük a $K=1$ és az $N=3$ esetet.



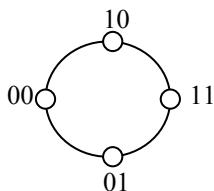
A kódok terében a távolságot (az euklideszi távolság helyett) az ún. **Hamming-távolsággal** mérjük. Két kód Hamming-távolságán azon pozíciók (koordináták) számát értjük, melyekben a két kód különbözik egymástól.

A fenti példában ez a távolság: $d(\mathbf{c}_0, \mathbf{c}_1) = 3$.

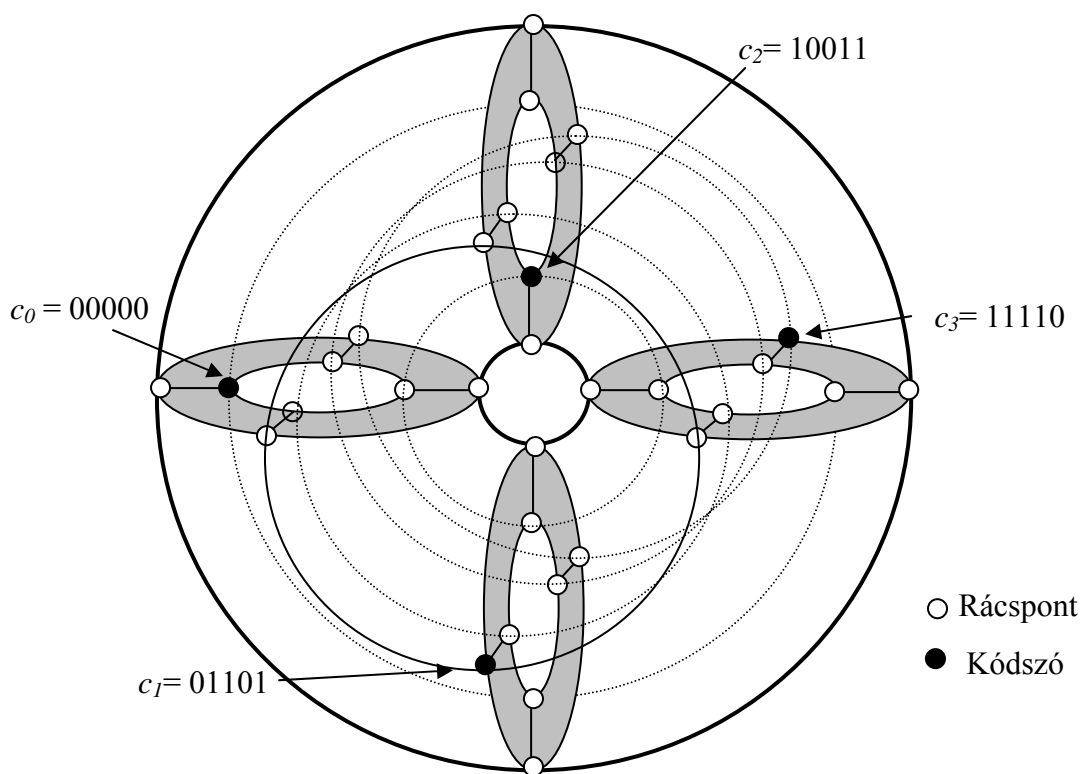
A Hamming-távolságra igaz az ún. háromszög egyenlőtlenség, mely szerint:

$$d(\mathbf{c}_A, \mathbf{c}_B) + d(\mathbf{c}_B, \mathbf{c}_C) \geq d(\mathbf{c}_A, \mathbf{c}_C) \quad (2.12.)$$

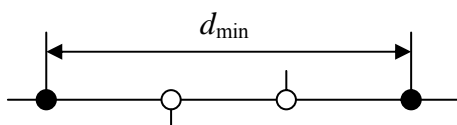
Ha $N > 3$, akkor a kódtér derékszögű koordináta-rendszerként való *ábrázolása* nehézségekbe ütközik. Mivel a Hamming-távolság különbözik az euklideszi távolságtól, azért nincs annak különösebb jelentősége, hogy az ábrázolásban a rácspontok azonos euklideszi távolságra legyenek (ui csak a szomszédság a fontos). A koordináták számát lehet csökkenteni azzal, ha pld két bitet összefogunk



Az alábbi ábrán az $N = 5$ esethez tartozó 32 rácspontot tüntettük fel. A pontok a háromdimenziós euklideszi térbe helyezett két koncentrikus toroid felületén helyezkednek el. Az első koordináta (2 bit) lehet az, hogy melyik satírozott metszeten, a második koordináta (1 bit) a belső vagy a külső toroidot, míg a harmadik koordináta (2 bit) a metszet körén elfoglalt pozíciót határozza meg.



A lehetséges rácspontok közül a kódok kiválasztása történjen olyan módon, hogy *maximalizálni* igyekezzünk a kódszavak közötti *minimális távolságot*. A fenti példában ez a távolság: $d_{\min} = 3$.



A d_{\min} ismeretében határozhatjuk meg a kód **hibajelző képességét**:

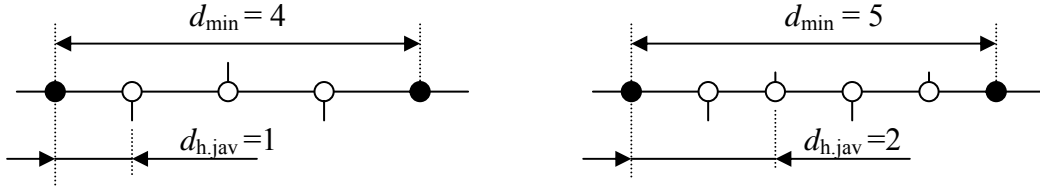
$$d_{h,jel} = d_{\min} - 1 \quad (2.13.)$$

Ugyanis amíg a hibák száma kisebb d_{\min} -nél, addig a kódtérben olyan rácsponatok vagyunk, amelyek nem kódszavak és ez alkalmas a hiba jelzésére.

A hiba javítását a maximum-likelihood elvből következően a közelebbi kódszó irányába fogjuk elvégezni. A kód **hibajavító képességén** azt a maximális Hamming-távolságot értjük, melyre még értelmezhető a *közelebbi kód*.

$$d_{h,jav} = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor \quad (2.14.)$$

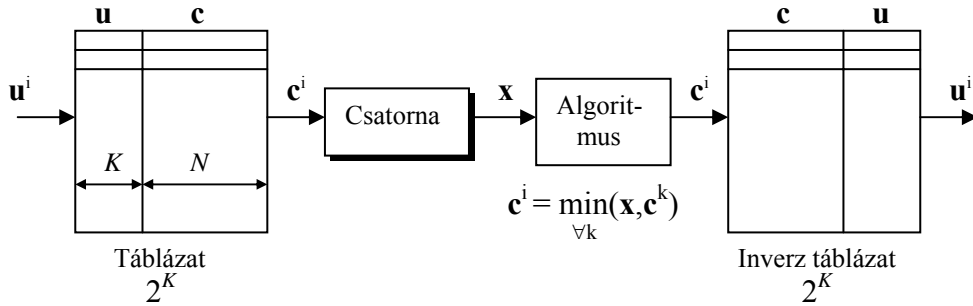
Ahol a $\lfloor \dots \rfloor$ a lefelé kerekítést jelöli.



Ha a hibák száma kisebb vagy egyenlő $d_{h,jav}$ értékénél, az automatikus javítás mindig elvégezhető.

A kódolót ezek után úgy képzelhetjük el, hogy az u^i üzenet vektor címez egy táblázatot, amiből kiolvassuk a c^i kódot.

A dekódolóban az első feladat annak megállapítása, hogy a vett (esetleg hibás) x kód melyik c^i kódszóhoz esik a legközelebb. Egy inverz táblázat segítségével a kódszóhoz tartozó üzenet ezek után meghatározható.



Mint láttuk a redundáns bitek száma $N-K$. Ennek a különbségnek nem kell túlságosan nagyok lenni ahhoz, hogy a kódtér meglehetősen "üres" legyen (2^{N-K} – szoros a viszony). A ritkán elhelyezett kódok eredményezhetik ugyanis a nagy minimális Hamming-távolságot, azaz a több hiba kijavításának a lehetőségét.

A redundáns bitek akkor nem okoznak relatíve nagy kódszóhosszúság növekedést, ha maga a K értéke nagy (pld.50). Ez viszont felveti azt a kérdést, hogy a keresés 2^K db reláció kiértékelését igényli, (amire nagysebességű adatátvitelnél általában nincs elég idő) vagy 2^N méretű táblázatokat kell tárolni (nincs elég hely).

2.4. Lineáris, szisztematikus kódok

A táblázatok használata elkerülhető, ha a hozzárendeléseket függvények formájában végezzük el. Lineáris, szisztematikus hibajavító kódok alkalmazásával a függvények kiszámításának műveleti igénye K növelésével csak polinomiálisan növekszik, ellentétben az előző esettel, ahol is a táblázatok mérete exponenciálisan függött az üzenetvektor hosszúságától.

Egy kód **lineáris**, ha: $\mathbf{c}^i \in \mathbf{C}$ és $\mathbf{c}^j \in \mathbf{C}$, akkor

$$\mathbf{c}^k = \mathbf{c}^i + \mathbf{c}^j \in \mathbf{C} \quad \forall i, j \quad (2.15.)$$

Bináris vektorokról lévén szó, a koordinátánkénti összegzés átvitel nélküli összeadást (bitenkénti XOR műveletet) jelent. (Az összeadás és a kivonás művelete megegyezik.)

Lineáris kód esetén így :

$$\mathbf{c}^0 = \mathbf{c}^i + \mathbf{c}^i = \mathbf{0} \in \mathbf{C} \quad \forall i$$

azaz a nullvektor mindig kódszó.

Egy kódszó w **súlyán** a benne előforduló 1-esek számát értjük. A kód súlyának segítségével beláthatjuk, hogy lineáris kódok esetén a kódok közötti minimális távolság d_{\min} , a legkisebb (de nem zérus) súlyú kód súlyával egyezik meg.

$$d_{\min} = \min_{\substack{\forall i, j \\ i \neq j}} d(\mathbf{c}^i, \mathbf{c}^j) = \min_{\substack{\forall i, j \\ i \neq j}} w(\mathbf{c}^i - \mathbf{c}^j) = \min_{\substack{\forall k \\ k \neq 0}} w(\mathbf{c}^k) \quad (2.16.)$$

Legyen:

$$\mathbf{u}^i = [u_1^i \quad u_2^i \quad \dots \quad u_K^i] \text{ és } \mathbf{c}^i = [c_1^i \quad c_2^i \quad \dots \quad c_K^i \quad \dots \quad c_N^i]$$

Lineáris kódok esetén a kód képzése a $K \times N$ dimenziós **G generátor mátrix** segítségével történik:

$$\mathbf{c} = \mathbf{u} \mathbf{G}_{K,N} \quad (2.17.)$$

Szisztematikus kód (a kód eleje maga az üzenet):

$$\mathbf{c} = [\mathbf{u}, \mathbf{p}] = [u_1 \quad u_2 \quad \dots \quad u_K \quad p_{K+1} \quad \dots \quad p_N] \quad (2.18.)$$

Ekkor **G** speciális felépítésű:

$$\mathbf{G}_{K,N} = [\mathbf{I}_{K,K} \quad \mathbf{A}_{K,(N-K)}] \quad (2.19.)$$

ahol: $\mathbf{I}_{K,K}$ egy $K \times K$ dimenziós egységmátrix.

A vételi oldalon az átviteli hibák felfedésére és javítására az $(N-K) \times N$ dimenziós **H** (u.n. **paritás ellenőrző**) mátrixot használjuk

Ettől követeljük meg, hogy:

$$\mathbf{H}\mathbf{c}^T = \bar{\mathbf{0}} \quad (2.20.)$$

ahol T a transzponáltat jelöli.

A (2.17.) alapján írhatjuk:

$$\mathbf{H}\mathbf{c}^T = \mathbf{H}\mathbf{G}^T \mathbf{u}^T = \bar{\mathbf{0}} \quad \forall \mathbf{u} \quad (2.21.)$$

amiből:

$$\mathbf{H}\mathbf{G}^T = \bar{\mathbf{0}} \quad (2.22.)$$

Egyszerűen belátható, hogy a szorzat akkor adja a zérus mátrixot, ha:

$$\mathbf{H}_{(N-K),N} = [\mathbf{A}_{(N-K),K}^T \quad \mathbf{I}_{(N-K),(N-K)}] \quad (2.23.)$$

Írjuk fel az \mathbf{x} vett üzenetet a küldött \mathbf{c} kód és egy \mathbf{e} hibavektor (error) összegeként:

$$\mathbf{x} = \mathbf{c} + \mathbf{e} \quad (2.24.)$$

Ahol az \mathbf{e} vektorban „1” a koordináta értéke, ott az aktuális bit az ellenkezőjére vált, azaz hiba keletkezik (XOR művelet).

Ugyanakkor a (2.24.) összefüggés fordítottja is érvényes. Ha ismernénk az \mathbf{e} vektort, akkor vissza tudnánk állítani az \mathbf{x} hibás üzenetből a hibátlan kódot(XOR):

$$\mathbf{c}' = \mathbf{x} + \mathbf{e} \quad (2.25.)$$

Vegyük észre, hogy az \mathbf{e} -ben lévő „1”-esek száma: az üzenetben lévő *hibák számát* adja.

A hiba vektor meghatározását az \mathbf{s} (**szindróma vektor**) kiszámításával fogjuk kezdeni.

$$\mathbf{s} = \mathbf{H}\mathbf{x}^T \quad (2.26.)$$

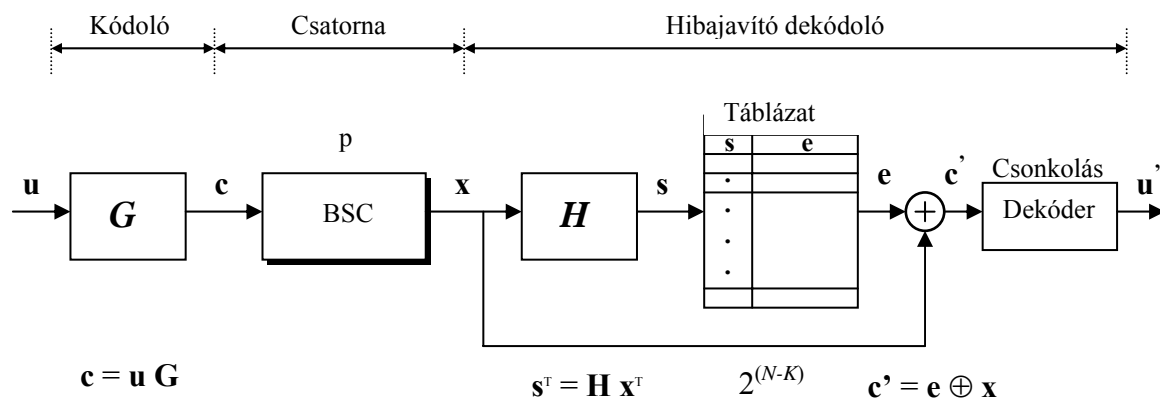
A (2.22.) és a (2.24.) alapján:

$$\mathbf{s} = \mathbf{H}\mathbf{x}^T = \mathbf{H}(\mathbf{c}^T + \mathbf{e}^T) = \mathbf{H}\mathbf{c}^T + \mathbf{H}\mathbf{e}^T = \bar{\mathbf{0}} + \mathbf{H}\mathbf{e}^T = \mathbf{H}\mathbf{e}^T \quad (2.27.)$$

Az eredményül kapott $\mathbf{s} = \mathbf{H}\mathbf{e}^T$ összefüggésből a hibavektort egyszerűen nem tudjuk kifejezni, mivel a \mathbf{H} paritás ellenőrző mátrix nem négyzetes mátrix, így az nem invertálható. (Viszont a tervezés fázisában kiszámíthatjuk és táblázatba foglalhatjuk az összes lehetséges hibavektorhoz tartozó szindróma vektort, mely táblázatból - fordított irányban- ki tudjuk olvasni az adott szindrómához rendelt hiba vektort.)

A vételi oldalon elvégezzük a (2.26.) műveletet, majd az \mathbf{s} szindróma vektor ismeretében az inverz táblázat segítségével meghatározzuk az \mathbf{e} hiba vektort.

A (2.25.) felhasználásával kiszámítjuk a javított \mathbf{c}' kódvektort, amiből egyszerű csonkolással kinyerhető az \mathbf{u} üzenet (szisztematikus kód).



A fenti eljárás természetesen az összes elképzelhető hibát nem tudja kijavítani. Az inverz táblázat elkészítése ugyanis nem egyértelmű. Az N dimenziós hibavektorok lehetséges száma 2^N , az $(N-K)$ dimenziós szindróma vektorok lehetséges száma viszont csak $2^{(N-K)}$. Így az inverz táblázat nem lesz egyértelmű: azonos szindrómákhoz több hiba vektor is tartozik.

Az egyes szindrómákhoz az összetartozó hibavektorok közül azt a hibavektort választjuk, melyekben az „1”-esek száma a lehető legkisebb. Ugyanis kis hibavalószínűségű csatorna esetén a kevés hibának sokkal nagyobb a valószínűsége, mint a több hibának (*maximum-likelihood* becslés).

2.5. Példa: Hibajavító lineáris kódoló tervezése

Tervezzünk lineáris hibajavító kódolót $C(5,2)$ kódok használatával ($N = 5$ és $K = 2$). A kódoló generátor mátrixa legyen:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix} \quad \text{ahol:} \quad \mathbf{A} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

- Adjuk meg a kódszavakat! Lineáris-e a kód?
- Milyen hibajavítási és hibajelzési képességgel rendelkezik a kód?
- Határozzuk meg a \mathbf{H} paritásellenőrző mátrixot!
- Tervezzük meg a szindróma dekódolási táblázatot!
- Ha a memóriamentes, szimmetrikus, bináris csatorna hibavalószínűsége $p = 10^{-3}$, mekkora lesz annak a valószínűsége, hogy a csatornában kettőnél több hiba keletkezik?

Megoldás:

a.)

Az üzenet vektorok: $\mathbf{u}^0 = [0 \ 0]$,
 $\mathbf{u}^1 = [0 \ 1]$,
 $\mathbf{u}^2 = [1 \ 0]$,
 $\mathbf{u}^3 = [1 \ 1]$.

A kód vektorok: $\mathbf{c} = \mathbf{u} \mathbf{G}$

$$\mathbf{c}^0 = [0,0] \begin{bmatrix} 1,0,0,1,1 \\ 0,1,1,0,1 \end{bmatrix} = [0,0,0,0,0] \quad \mathbf{c}^1 = [0,1] \begin{bmatrix} 1,0,0,1,1 \\ 0,1,1,0,1 \end{bmatrix} = [0,1,1,0,1]$$

$$\mathbf{c}^2 = [1,0] \begin{bmatrix} 1,0,0,1,1 \\ 0,1,1,0,1 \end{bmatrix} = [1,0,0,1,1] \quad \mathbf{c}^3 = [1,1] \begin{bmatrix} 1,0,0,1,1 \\ 0,1,1,0,1 \end{bmatrix} = [1,1,1,1,0]$$

A kódszavakat a fenti mátrix szorzások elvégzése nélkül is megkaphatjuk, ha felismerjük, hogy a \mathbf{G} generátor mátrix sorai kódszavak és ezek lineár-kombinációi is azok.

A kód lineáris, mert: $\mathbf{c}^0 = \bar{0}$ és teljesül a:

$$\mathbf{c}^k = \mathbf{c}^i + \mathbf{c}^j \in \mathbf{C} \quad \text{ahol: } i,j,k = 0,1,2,3$$

A kód szisztematikus, mert $\mathbf{c}^i = [\mathbf{u}^i, \mathbf{p}^i]$

b.) A kódszavak Hamming távolsága:

$$\begin{aligned} d(0,1) &= 3, & d(0,2) &= 3, & d(0,3) &= 4 \\ d(1,2) &= 4, & d(1,3) &= 3, \\ d(2,3) &= 3 \end{aligned}$$

A minimális kódtávolságot a 2.16. szerint határozzuk meg:

$$d_{\min} = \min_{\substack{\forall k \\ k \neq 0}} w(\mathbf{c}^k) = w(\mathbf{c}^1) = 3$$

A minimális Hamming távolság: $d_{\min} = 3$

A kód hibajelző képessége: $d_{\min} - 1 = 2$

A kód hibajavító képessége: $\left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor = 1$

c.) A paritás ellenőrző mátrix:

$$\mathbf{H}_{3,5} = [\mathbf{A}_{3,2}^T \quad \mathbf{I}_{3,3}] = \begin{bmatrix} 0 & 1 & \vdots & 1 & 0 & 0 \\ 1 & 0 & \vdots & 0 & 1 & 0 \\ 1 & 1 & \vdots & 0 & 0 & 1 \end{bmatrix}$$

d.)

Az $\mathbf{s} = \mathbf{H}\mathbf{e}^T$ összefüggésből kiszámítjuk a szindróma vektorokat olyan sorrendben, hogy a hibavektorokban az „1”-esek száma növekedjen:

$\mathbf{e}(\text{dec})$	$\mathbf{e}(\text{bin})$	“1” száma	$\mathbf{s}(\text{bin})$	$\mathbf{s}(\text{dec})$
0.	00000	0	000	0.
1.	00001	1	001	1.
2.	00010	1	010	2.
4.	00100	1	100	4.
8.	01000	1	101	5.
16.	10000	1	011	3.
3.	00011	2	011	-
5.	00101	2	101	-
6.	00110	2	110	6.
9.	01001	2	100	-
10.	01010	2	111	7.
12.	01100	2	001	-
17.	10001	2	010	-
18.	10010	2	001	-
20.	10100	2	111	-
24.	11000	2	110	-
7.	00111	3	111	-
11.	01011	3	110	-
13.	01101	3	000	-
14.	01110	3	011	-
19.	10011	3	000	-
21.	10101	3	110	-
22.	10110	3	101	-
25.	11001	3	111	-
26.	11010	3	100	-
28.	11100	3	010	-
23.	10111	4	100	-
15.	01111	4	010	-
27.	11011	4	101	-
29.	11101	4	011	-
30.	11110	4	000	-
31.	11111	5	001	-

A táblázat kitöltését a 10. jelű sora után abbahagyhattuk volna, mivel addigra a szindróma szavak összes lehetséges kombinációja már előállt.

A szindróma szavakhoz tartozó hibavektorok egy lehetséges hozzárendelése:

$\mathbf{s}(\text{dec})$	$\mathbf{s}(\text{bin})$	$\mathbf{e}(\text{bin})$
0.	000	00000
1.	001	00001
2.	010	00010
3.	011	10000
4.	100	00100
5.	101	01000
6.	110	00110
7.	111	01010

A dekódoló rendszer kialakíthatjuk úgy, hogy 6. és 7. szindróma szavak kialakulását észelve, a rendszer hibajelzést (figyelmeztetést) szolgáltatson.

Hibajavító kódolás

Az alábbi táblázat a dekódolás c' eredményét (számítógépes kiértékelését) mutatja az összes lehetséges x bemeneti vektor függvényében. Tételezzük fel, hogy az átvitel hibák száma (hsz) egy szóban maximálisan 2. A táblázatban c jelöli a küldött kódot, az x -ben a „bold” karakter jelöli a hiba helyét.

$x(\text{dec})$	$x(\text{bin})$	hsz	c	s	e	c'	javítás után	hiba jelzés
0.	00000	0	00000	000	00000	00000	ok	-
1.	00001	1	00000	001	00001	00000	ok	-
2.	00010	1	00000	010	00010	00000	ok	-
3.	00011	1	10011	011	10000	10011	ok	-
4.	00100	1	00000	100	00100	00000	ok	-
5.	00101	1	01101	101	01000	01101	ok	-
6.	00110	2	00000	110	00110	00000	ok	+
7.	00111	2	01101	111	01010	01101	ok	+
8.	01000	1	00000	101	01000	00000	ok	-
9.	01001	1	01101	100	00100	01101	ok	-
10.	01010	2	00000	111	01010	00000	ok	+
11.	01011	2	01101	110	00110	01101	ok	+
12.	01100	1	01101	001	00001	01101	ok	-
13.	01101	0	01101	000	00000	01101	ok	-
14.	01110	1	11110	011	10000	11110	ok	-
15.	01111	1	01101	010	00010	01101	ok	-
16.	10000	1	00000	011	10000	00000	ok	-
17.	10001	1	10011	010	00010	10011	ok	-
18.	10010	1	10011	001	00001	10011	ok	-
19.	10011	0	10011	000	00000	10011	ok	-
20.	10100	2	00000	111	01010	11110	hiba	+
21.	10101	2	01101	110	00110	10011	hiba	+
22.	10110	1	11110	101	01000	11110	ok	-
23.	10111	1	10011	100	00100	10011	ok	-
24.	11000	2	00000	110	00110	11110	hiba	+
25.	11001	2	01101	111	01010	10011	hiba	+
26.	11010	1	11110	100	00100	11110	ok	-
27.	11011	1	10011	101	01000	10011	ok	-
28.	11100	1	11110	010	00010	11110	ok	-
29.	11101	1	01101	011	10000	01101	ok	-
30.	11110	0	11110	000	00000	11110	ok	-
31.	11111	1	11110	001	00001	11110	ok	-

Ugyanakkor ha a hibák száma nagyobb mint 2, akkor előfordulhat, hogy **nem kapunk hibajelzést** miközben az átvitel hibás.

Például ha a küldött kód [00000] és a hibák száma 4 (úgy, hogy a táblázat 29. sora szerinti x vektort kapjuk), a „javított” c' [01101] lesz.

$x(\text{dec})$	$x(\text{bin})$	hsz	c	s	e	c'	„javítás” után	hiba jelzés
29.	11101	4	00000	011	10000	01101	hiba	-

e.)

A memória mentes, bináris, szimmetrikus csatornában egy bit átvitelekor keletkező hiba valószínűsége legyen: p ($= 10^{-3}$). Annak az eseménynek a valószínűsége, hogy a hiba nem lép fel: $1-p$ ($= 0.999$).

Több bit átvitelekor a csatornában a hibák elmaradása illetve bekövetkezése egymástól **független eseménynek** tekinthető. Annak valószínűsége ezért, hogy mind az 5 bit hibátlan:

$$P_0 = \Pr\{e_1=0 \text{ és } e_2=0 \text{ és } e_3=0 \text{ és } e_4=0 \text{ és } e_5=0\} = (1-p)^5 = 0.9860 \dots$$

Hasonlóan, ha mind az 5 bit hibás:

$$P_5 = \Pr\{e_1=1 \text{ és } e_2=1 \text{ és } e_3=1 \text{ és } e_4=1 \text{ és } e_5=1\} = p^5 = 10^{-15}.$$

Ha az 5 bitből k darab hibás (ez az esemény „5 alatt a k ”-féle képpen lehetséges):

$$P_k = \binom{5}{k} p^k (1-p)^{5-k}$$

Így:

$$P_1 = \binom{5}{1} p (1-p)^4 = \frac{5!}{1!4!} p (1-p)^4 = 5 p (1-p)^4 = 4.980 \cdot 10^{-3}$$

$$P_2 = \binom{5}{2} p^2 (1-p)^3 = \frac{5!}{2!3!} p^2 (1-p)^3 = 10 p^2 (1-p)^3 = 9.997 \cdot 10^{-6}$$

$$P_3 = \binom{5}{3} p^3 (1-p)^2 = \frac{5!}{3!2!} p^3 (1-p)^2 = 10 p^3 (1-p)^2 = 9.998 \cdot 10^{-9}$$

$$P_4 = \binom{5}{4} p^4 (1-p) = \frac{5!}{1!4!} p^4 (1-p) = 5 p^4 (1-p) = 4.995 \cdot 10^{-12}$$

Tehát annak a valószínűsége, hogy a hibák száma nagyobb mint kettő:

$$\Pr\{h.sz. > 2\} = P_3 + P_4 + P_5 \cong 10^{-8}.$$

Ebből a numerikus példából látható a *maximum likelihood* filozófia alapja, nevezetesen az, hogy: sokkal valószínűbb a kevés hiba mint a sok hiba.