Információ- és kódelmélet

Györfi László Győri Sándor Vajda István

2002. november 29.

Tartalomjegyzék

Előszó

Az információelmélet a hírközlés matematikai elmélete. Születését lényegében Claude Shannon [43] művének 1948-as megjelenéséhez köthetjük. Ez a munka volt az első, amely matematikai alapossággal tárgyalta az adattömörítés, a biztonságos adatátvitel és a titkosítás problémáit. Shannon adott először kezelhető és hasznos matematikai modelleket az információs folyamatok leírására, mégpedig úgy, hogy az egyes problémák esetén tisztázta az elvi határokat, és többségében meg is konstruálta azokat a módszereket, amelyek ezeket az elvi határokat aszimptotikusan elérik. Ugyanakkor napjainkban tömegesen terjedtek el az egyes adattömörítő és hibakorlátozó eljárások, tehát indokolt, hogy ezek alapvető elveit is áttekintsük. Az 1. fejezetben ismertetjük a veszteségmentes adattömörítést, míg a 2. fejezetben a veszteséget (torzítást) megengedő adattömörítő (forráskódoló) eljárásokat tárgyaljuk. A 3. fejezet témája a csatornakódolás.

A forráskódolás elméletével ellentétben a csatornakódolás eredményei nem konstruktívak, tehát ma még nem ismertek olyan kódolási–dekódolási eljárások, amelyek tetszőleges csatorna esetén a csatornakapacitást megközelítenék. Ugyanakkor ismertek olyan algebrai hibavédő kódok, amelyek számos gyakorlati probléma megoldását segítik. A ’80-as évektől alkalmazzák polgári célokra is a hibavédő kódokat, napjainkban a CD-ben használt Reed–Solomon-kód szinte minden háztartásban megtalálható. A 4. fejezet összefoglalja a hibajavító kódok alapjait.

Az 5. fejezet témája a nyilvános kulcsú titkosítás, tehát az a probléma, hogy nyilvános hálózaton hogyan biztosítható az adat- illetve hozzáférésvédelem.

Ez a tankönyv a Linder – Lugosi [25] és a Györfi – Vajda [22] jegyzet „unió-jának” a kiegészítése azon tapasztalatok felhasználásával, amelyeket a BME műszaki informatikusoknak tartott Információelmélet és Kódelmélet tárgyak oktatása során szereztünk az elmúlt 10 évben.

A mű elkészítésében nyújtott segítségükért szeretnénk köszönetet mondani György Andrásnak, Laczay Bálintnak, Linder Tamásnak, Lois Lászlónak, Lugosi Gábornak, Pataricza Andrásnak és Pintér Mártának.

Budapest, 2000. augusztus 22.

Györfi László
Győri Sándor
Vajda István

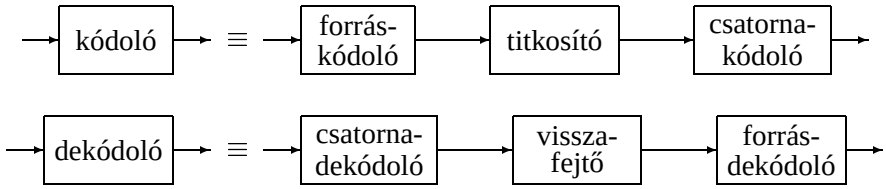
1. fejezet

Változó szóhosszúságú forráskódolás

Ebben a fejezetben bevezetjük a veszteségmentes adattömörítés fogalmát. Ennek célja az, hogy egy üzenetet, amely egy véges halmaz (forrásábécé) elemeiből álló véges sorozat, egy másik véges halmaz (kódábécé) elemeiből álló sorozattal reprezentáljunk úgy, hogy ez a reprezentálás a lehető legrövidebb (és belőle az üzenet visszaállítható) legyen. Ennek megfelelően azt vizsgáljuk, hogy az adattömörítésnek melyek az elvi határai, és ezeket a határokat hogyan közelíthetjük meg. A legismertebb példa erre az, amikor a kódábécé a $\{0,1\}$ halmaz, és egy üzenetet (pl. írott szöveg, fájl) bináris sorozatok formájában kódolunk tárolás, illetve átvitel céljából. Az üzenetet kibocsátó objektum — az információforrás — modellezésével és vizsgálatával az 1.6. szakasz foglalkozik (mivel valószínűségi modelleket használunk, ezért az üzenetek betűi valószínűségi változók lesznek). Célunk az, hogy az ehhez szükséges matematikai apparátust bevezessük. Megismerkedünk az üzenetek egy természetesen adódó kritérium szerinti kódolásával — az egyértelműen dekódolható kódolással —, mely később vizsgálataink középpontjában áll majd. Bevezetjük a diszkrét valószínűségi változó entrópiáját, és megmutatjuk, hogy az entrópia milyen alapvető szerepet játszik a kódolással kapcsolatban. Az elkövetkezőkben mindig azt tartjuk szem előtt, hogy az üzenetek kódja minél rövidebb legyen, ezért a kódok általunk vizsgált fő tulajdonsága a átlagos kódszóhossz lesz. A fejezetet az univerzális forráskódolással és annak gyakorlati alkalmazásaival zárjuk.

A hírközlési rendszerek működését egzakt matematikai eszközökkel fogjuk vizsgálni. Persze sokféle matematikai modellt állíthatunk hírközlési feladatokra, de kezdetben az egyik legegyszerűbbet választva jól kifejezhetjük a probléma lényegét. A hírközlés alapfeladata az, hogy valamely jelsorozatot („információt”)

1.1. ábra. Hírközlési rendszer blokkdiagramja.



1.2. ábra. A kódoló és a dekódoló felépítése.

el kell juttatni egyik helyről a másikra. A távolságot (vagy időt) áthidaló hírközlési eszköz — a csatorna — azonban csak meghatározott típusú jeleket képes átvinni. Az információforrás által produkált jelfolyamot kódolással meg kell feleltetni egy, a csatorna által használt jelekből álló jelfolyamnak. A felhasználó (nyelő) a csatorna kimenetén pontosan, vagy megközelítőleg visszaállítja, dekódolja az üzenetet. Egy ilyen rendszer blokkdiagrammja látható az 1.1. ábrán.

A kódoló általános esetben három részből áll. Az első a forráskódoló, amely a forrás üzeneteit gazdaságosan, tömören reprezentálja. Ezzel foglalkozik az első két fejezet. A második rész a titkosító (5. fejezet), és a harmadik a csatornakódoló (3. és 4. fejezet). Ennek megfelelően a dekódoló is három részből áll: csatornadekódoló, visszafejtő és forrásdekódoló (1.2. ábra).

1.1. Egyértelmű dekódolhatóság, prefix kódok

Jelöljön X egy diszkrét valószínűségi változót, amely az $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ véges halmazból veszi értékeit. Az $\mathcal{X}$ halmazt a továbbiakban **forrásábécé**nek, elemeit pedig betűknek nevezzük.

$\mathcal{Y}$ jelöljön egy s elemű $\{y_1, y_2, \ldots, y_s\}$ halmazt. Ezt **kódábécé**nek nevezzük. $\mathcal{Y}^*$ jelölje az $\mathcal{Y}$ elemeiből álló véges sorozatok halmazát. $\mathcal{Y}^*$ elemeit **kódszavak**nak nevezzük. Egy $f : \mathcal{X} \to \mathcal{Y}^*$ függvényt, amely megfeleltetést létesít a forrásábécé és a kódszavak között, **kód**nak nevezünk. Az $\mathcal{X}$ elemeiből alkotott véges sorozatok az **üzenet**ek vagy közlemények (értékeiket az $\mathcal{X}^*$ halmazból veszik). Amennyiben az f kód értékkészlete különböző hosszú kódszavakból áll, úgy változó szóhosszúságú kódolásról beszélünk.

1.1. definíció. *Az $f : \mathcal{X} \to \mathcal{Y}^*$ kód **egyértelműen dekódolható**, ha minden véges kódbetűsorozat legfeljebb egy közlemény kódolásával állhat elő, azaz ha $\mathbf{u} \in \mathcal{X}^*$, $\mathbf{v} \in \mathcal{X}^*$, $\mathbf{u} = u_1u_2\dots u_k$, $\mathbf{v} = v_1v_2\dots v_m$, $\mathbf{u} \neq \mathbf{v}$, akkor $f(u_1)f(u_2)\dots f(u_k) \neq f(v_1)f(v_2)\dots f(v_m)$. (Itt az $f(u)f(u')$ a két kódszó egymás után írását [konkatenáció] jelenti.)*

MEGJEGYZÉS:

a) Az egyértelmű dekódolhatóság több, mint az invertálhatóság. Ugyanis legyen $\mathcal{X} = \{a,b,c\}, \mathcal{Y} = \{0,1\}$ és $f(a) = 0, f(b) = 1, f(c) = 01$. Ekkor az $f : \mathcal{X} \to \mathcal{Y}^*$ leképezés invertálható, viszont a 01 kódszót dekódolhatjuk $f(a)f(b) = 01$ szerint *ab*-nek, vagy $f(c) = 01$ szerint *c*-nek is.

b) Az előbbi definícióban szereplő kódolási eljárást, amikor egy közlemény kódját az egyes forrásbetűkhöz rendelt kódszavak sorrendben egymás után írásával kapjuk, betűnkénti kódolásnak nevezzük. Természetesen a kódolást teljesen általánosan egy $g : \mathcal{X}^* \to \mathcal{Y}^*$ függvénnyel is definiálhatnánk, de később látni fogjuk, hogy a betűnkénti kódolás és annak természetes kiterjesztése, a blokk-kódolás elegendő számunkra.

1.2. definíció. *Az f kód **prefix**, ha a lehetséges kódszavak közül egyik sem folytatása a másiknak, vagyis bármely kódszó végéből bármekkora szegmenst levágva nem kapunk egy másik kódszót. Egy prefix kód egyben egyértelműen dekódolható is.*

Az eddig bevezetett fogalmak illusztrálására nézzük a következő példákat:

1.1. példa. $\mathcal{X} = \{a,b,c\}$; $\mathcal{Y} = \{0,1\}$ A kód legyen a következő: $f(a) = 0,\ f(b) = 10,\ f(c) = 110$. Könnyen ellenőrizhető, hogy a kód prefix.

Ha az *abccab* üzenetet kódoljuk, akkor a 010110110010 kódbetűsorozatot kapjuk. A kódból az üzenet visszafejtése nagyon egyszerű a prefix tulajdonság miatt; többek között ez a gyors dekódolási lehetőség teszi vonzóvá a prefix kódokat.

1.2. példa. $\mathcal{X} = \{a,b,c,d\}$; $\mathcal{Y} = \{0,1\}$; $f(a) = 0,\ f(b) = 01,\ f(c) = 011,\ f(d) = 0111$. Jól látható, hogy a kód nem prefix, de egyértelműen dekódolható, hiszen a 0 karakter egy új kódszó kezdetét jelzi.

1.1. lemma (McMillan). *Minden egyértelműen dekódolható $f : \mathcal{X} \to \mathcal{Y}^*$ kódra*

$$\sum_{i=1}^{n} s^{-|f(x_i)|} \leq 1, \tag{1.1}$$

ahol s a kódábécé elemszáma, és $|f(x_i)|$ jelöli az $f(x_i)$ kódszóhosszát.

BIZONYÍTÁS: Tekintsük az (1.1) összeg N-edik hatványát:

$$\left(\sum_{i=1}^{n} s^{-|f(x_i)|}\right)^N = \sum_{i_1=1}^{n} \cdots \sum_{i_N=1}^{n} s^{-(|f(x_{i_1})|+\cdots+|f(x_{i_N})|)} = \sum_{l=1}^{N\cdot L_{\max}} A_l s^{-l},$$

ahol $L_{\max} = \max\limits_{1\le i\le n} |f(x_i)|$, és A_l jelöli az összes l hosszúságú, N darab kódszó egymás után írásával keletkező kódbetűsorozatok számát. Mivel feltevésünk szerint f egyértelműen dekódolható, az összes ilyen l hosszú sorozat különböző, tehát

$$A_l \le s^l.$$

Ebből azt kapjuk, hogy

$$\left(\sum_{i=1}^{n} s^{-|f(x_i)|}\right)^N \le N \cdot L_{\max},$$

vagyis

$$\sum_{i=1}^{n} s^{-|f(x_i)|} \le \sqrt[N]{N}\,\sqrt[N]{L_{\max}}. \tag{1.2}$$

Mivel N tetszőleges, és tudjuk, hogy $\sqrt[N]{N} \to 1$, $\sqrt[N]{L_{\max}} \to 1$, ha $N \to \infty$, ezért (1.2) csak úgy állhat fenn minden N-re, ha (1.1) igaz. ■

A következő lemma bizonyos értelemben az előző megfordítása.

1.2. lemma (Kraft). *Ha az $l_1, l_2, \ldots, l_n$ pozitív egész számokra*

$$\sum_{i=1}^{n} s^{-l_i} \le 1, \tag{1.3}$$

akkor létezik olyan f prefix kód, hogy

$$|f(x_i)| = l_i, \qquad i = 1, \ldots, n.$$

BIZONYÍTÁS: Az egyszerűség kedvéért tegyük fel, hogy az l_i számok nagyság szerint növekvő sorrendben vannak: $l_1 \le l_2 \le \cdots \le l_n$. A bizonyítás technikája megkívánja, hogy a kódábécé $\{y_1, y_2, \ldots, y_s\}$ elemeit a $\{0, 1, \ldots, s-1\}$ számokkal helyettesítsük (például a következő megfeleltetés szerint: $y_i \mapsto i-1$, $i = 1, \ldots, s$).

Definiáljuk a w_j számokat a következőképpen:

$$w_1 = 0, \quad w_j = \sum_{i=1}^{j-1} s^{l_j - l_i}, \qquad j = 2, \ldots, n.$$

(1.3) mindkét oldalát s^{l_n}-nel szorozva, majd mindkét oldalból 1-et levonva kapjuk, hogy

$$w_n = \sum_{i=1}^{n-1} s^{l_n - l_i} \leq s^{l_n} - 1.$$

Hasonlóképp eljárva azt kapjuk minden j-re, hogy $w_j \leq s^{l_j} - 1$. Ezek után definiáljuk $f(x_j)$-t mint a w_j szám s alapú számrendszerben felírt alakját, amelynek elejére annyi 0-t írunk, hogy a hossza l_j legyen. A kódszavak nyilván különbözők lesznek mivel $w_j < w_k$, ha $j < k$. Most belátjuk, hogy az így kapott kód prefix. Tegyük fel ugyanis ennek az ellenkezőjét, vagyis azt, hogy valamilyen $j < k$ esetén $f(x_j)$ végéhez $l_k - l_j$ darab számjegyet hozzáadva megkapjuk $f(x_k)$-t. Ekkor $w_j = \left\lfloor \frac{w_k}{s^{l_k - l_j}} \right\rfloor$ következne. (Az $\lfloor x \rfloor$ jelölés az x valós szám alsó egész részét jelöli). Viszont

$$\frac{w_k}{s^{l_k - l_j}} = \sum_{i=1}^{k-1} s^{l_j - l_i} = w_j + \sum_{i=j}^{k-1} s^{l_j - l_i} \geq w_j + 1,$$

és így ellentmondásra jutottunk. ■

1.1. következmény. *Az 1.1. és 1.2. lemmákat összevetve levonhatjuk azt a fontos következtetést, hogy minden egyértelműen dekódolható kódhoz létezik vele ekvivalens (azonos kódszóhosszú) prefix kód, tehát nem vesztünk semmit, ha az egyértelmű dekódolhatóság helyett a speciálisabb, és ezért könnyebben kezelhető prefix tulajdonságot követeljük meg.*

1.2. Átlagos kódszóhossz, entrópia

Legyen X egy $\mathcal{X}$ értékű valószínűségi változó, amit kódolni akarunk. Vezessük be a következő $p: \mathcal{X} \to [0,1]$ függvényt:

$$p(x) = \mathbf{P}\{X = x\}, \qquad x \in \mathcal{X}.$$

A $p(x)$ függvény tehát az x forrásbetűhöz annak valószínűségét rendeli.

1.3. definíció. *Az X valószínűségi változó **entrópiá**ját, $H(X)$-et, a*

$$H(X) = \mathbf{E}(-\log p(X)) = -\sum_{i=1}^{n} p(x_i) \log p(x_i).$$

összeggel definiáljuk.

MEGJEGYZÉS:

a) A $\log z$ jelölés a z pozitív szám kettes alapú logaritmusát jelenti. Mivel az 1.3. definíció megkívánja, a logaritmusfüggvénnyel kapcsolatban a következő „számolási szabályokat” vezetjük be ($a \geq 0, b > 0$):

$$0 \log \frac{0}{a} = 0 \log \frac{a}{0} = 0; \qquad b \log \frac{b}{0} = +\infty; \qquad b \log \frac{0}{b} = -\infty$$

(E szabályok az adott pontban nem értelmezett függvények folytonos kiterjesztései.) A definícióból közvetlenül látszik, hogy az entrópia *nemnegatív*. Vegyük észre, hogy az entrópia értéke valójában nem függ az X valószínűségi változó értékeitől, csak az eloszlásától.

b) Az entrópia intuitív fogalmával kapcsolatban lásd az 1.6. tétel utáni megjegyzést.

1.4. definíció. *Egy f kód **átlagos kódszóhossz**án az*

$$\mathbf{E}|f(X)| = \sum_{i=1}^{n} p(x_i)|f(x_i)|$$

várható értéket értjük.

1.3. példa. Az 1.1. példa kódja esetén legyen $p(a) = 0.5,\ p(b) = 0.3, p(c) = 0.2$, ekkor az entrópia

$$H(X) = -0.5 \cdot \log 0.5 - 0.3 \cdot \log 0.3 - 0.2 \cdot \log 0.2 \approx 1.485,$$

az átlagos kódszóhossz pedig

$$\mathbf{E}|f(X)| = 0.5 \cdot 1 + 0.3 \cdot 2 + 0.2 \cdot 3 = 1.7.$$

Ahhoz, hogy az egyértelműen dekódolható kódok átlagos kódszóhossza és entrópiája közti alapvető összefüggést bebizonyítsuk (ami tulajdonképpen e fejezet fő állítása), szükségünk van két segédtételre:

1.1. tétel (Jensen-egyenlőtlenség). *Legyen h egy valós, konvex függvény az $[a,b]$ zárt intervallumon, azaz minden $x, y \in [a,b]$ és $0 < \lambda < 1$ esetén*

$$h(\lambda x + (1-\lambda)y) \leq \lambda h(x) + (1-\lambda)h(y), \tag{1.4}$$

és legyen Z egy valószínűségi változó, amely értékeit az $[a,b]$ intervallumban veszi fel. Ekkor

$$h[\mathbf{E}(Z)] \leq \mathbf{E}[h(Z)]. \tag{1.5}$$

Továbbá, ha h az $\mathbf{E}(Z)$ pontban szigorúan konvex, vagyis az (1.4)-ben $\lambda x + (1-\lambda)y = \mathbf{E}(Z)$ esetén határozott egyenlőtlenség teljesül $\forall x, y$-ra, akkor (1.5)-ben egyenlőség akkor és csak akkor áll fenn, ha

$$\mathbf{P}\{Z = \mathbf{E}(Z)\} = 1,$$

vagyis ha Z 1-valószínűséggel konstans.

BIZONYÍTÁS: Mivel h konvex, ezért az (a,b) nyílt intervallum minden pontjában létezik a jobb és a bal oldali deriváltja és a bal oldali derivált legfeljebb akkora mint a jobb oldali. Továbbá az is igaz, hogy ezekkel a deriváltakkal mint meredekségekkel húzott félérintők a függvénygörbe alatt fekszenek $[a,b]$-ben. Jelölje c a jobb oldali deriváltat az $\mathbf{E}(Z)$ pontban, és írjuk fel itt a jobb oldali félérintő egyenletét:

$$g(x) = c[x - \mathbf{E}(Z)] + h[\mathbf{E}(Z)].$$

Ekkor az előbbiek szerint $h(x) \geq g(x)$ bármely $x \in [a,b]$ pontra, vagyis

$$h(x) \geq c[x - \mathbf{E}(Z)] + h[\mathbf{E}(Z)]. \tag{1.6}$$

Tehát írhatjuk,hogy

$$h(Z) \geq c[Z - \mathbf{E}(Z)] + h[\mathbf{E}(Z)],$$

ahol mindkét oldal várható értékét véve megkapjuk az első állítást. Mivel szigorúan konvex esetben (1.6)-ban egyenlőség csak $x = \mathbf{E}(Z)$ esetben teljesül, a második állítás triviálisan adódik. ■

1.2. következmény.

a) Ha $p_i \geq 0,\ q_i > 0,\ i = 1,2,\ldots,n$ valós számokra

$$\sum_{i=1}^{n} p_i = 1 \qquad \text{és} \qquad \sum_{i=1}^{n} q_i = 1,$$

akkor

$$-\sum_{i=1}^{n} p_i \log p_i \leq -\sum_{i=1}^{n} p_i \log q_i \tag{1.7}$$

és egyenlőség akkor és csak akkor áll fenn, ha $p_i = q_i$ minden i-re.

b) Ha $a_i \geq 0,\ b_i > 0,\ i = 1,2,\ldots,n$ valós számokra

$$\sum_{i=1}^{n} a_i = a \qquad \text{és} \qquad \sum_{i=1}^{n} b_i = b,$$

akkor

$$\sum_{i=1}^{n} a_i \log \frac{b_i}{a_i} \leq a \cdot \log \frac{b}{a},$$

és egyenlőség pontosan akkor áll fenn, ha $\frac{b_i}{a_i} =$ *konstans minden* i*-re.*

BIZONYÍTÁS:

a) (1.7) ekvivalens azzal, hogy

$$-\sum_{i=1}^{n} p_i \log \frac{q_i}{p_i} \geq 0.$$

Mivel a $h(x) = -\log x$ függvény szigorúan konvex az értelmezési tartományán (a második deriváltja pozitív), így felhasználhatjuk a Jensen-egyenlőtlenséget a következőképpen:

Legyen Y egy olyan valószínűségi változó, hogy

$$\mathbf{P}\left\{Y = \frac{q_i}{p_i}\right\} = p_i, \qquad i = 1, \ldots, n.$$

Ekkor a Jensen-egyenlőtlenség szerint

$$-\sum_{i=1}^{n} p_i \log \frac{q_i}{p_i} = \mathbf{E}(-\log Y) \geq -\log \mathbf{E}(Y) = -\log \sum_{i=1}^{n} p_i \frac{q_i}{p_i} = 0,$$

és egyenlőség akkor és csak akkor teljesül, ha $\frac{q_i}{p_i} =$ konstans minden i-re, de ekkor az összegekre vonatkozó feltételek miatt $p_i = q_i$ minden i-re.

b) Ha $a = 0$, akkor $\forall a_i = 0$, s ekkor az állítás „számolási szabályaink” felhasználásával minden $b > 0$-ra teljesül. Ha $a \neq 0$, akkor az a) pont szerint

$$\frac{1}{a}\left(\sum_{i=1}^{n} a_i \log \frac{b_i}{a_i} - \sum_{i=1}^{n} a_i \log \frac{b}{a}\right) = \sum_{i=1}^{n} \frac{a_i}{a} \cdot \log \frac{b_i/b}{a_i/a} \leq 0,$$

tehát a zárójelben levő kifejezésre

$$\sum_{i=1}^{n} a_i \log \frac{b_i}{a_i} - \sum_{i=1}^{n} a_i \log \frac{b}{a} \leq 0.$$

Szintén az a) pont szerint az egyenlőség feltétele az $\frac{a_i}{a} = \frac{b_i}{b}$, vagyis az $\frac{a_i}{b_i} = \frac{a}{b} =$ konstans minden i-re. ■

Most kimondjuk a fejezet egyik főtételét.

1.2. tétel. *Tetszőleges egyértelműen dekódolható $f: \mathcal{X} \to \mathcal{Y}^*$ kódra*

$$\mathbf{E}|f(X)| = \sum_{i=1}^{n} p(x_i)|f(x_i)| \geq \frac{H(X)}{\log s}. \tag{1.8}$$

MEGJEGYZÉS: Ha az entrópia definíciójában kettes alapú logaritmus helyett s-alapú logaritmust használnánk, akkor ezt az entrópiát $H_s(X)$-szel jelölve $\frac{H(X)}{\log s} = H_s(X)$ miatt (1.8) a következő alakú lenne:

$$\mathbf{E}|f(X)| \geq H_s(X).$$

BIZONYÍTÁS: Az (1.8) állítás ekvivalens a következővel:

$$H(X) \leq -\sum_{i=1}^{n} p(x_i) \log s^{-|f(x_i)|}.$$

A Jensen-egyenlőtlenség 1.2. a) következményét alkalmazva a

$$p_i = p(x_i), \qquad q_i = \frac{s^{-|f(x_i)|}}{\sum\limits_{j=1}^{n} s^{-|f(x_j)|}}, \qquad i = 1, \ldots, n;$$

szereposztásban megkapjuk az állítást:

$$\begin{aligned} H(X) &= -\sum_{i=1}^{n} p(x_i) \log p(x_i) \leq \\ &\leq -\sum_{i=1}^{n} p(x_i) \log \frac{s^{-|f(x_i)|}}{\sum\limits_{j=1}^{n} s^{-|f(x_j)|}} = \\ &= -\sum_{i=1}^{n} p(x_i) \log s^{-|f(x_i)|} + \log \left(\sum_{i=1}^{n} s^{-|f(x_i)|} \right) \leq \\ &\leq -\sum_{i=1}^{n} p(x_i) \log s^{-|f(x_i)|}, \end{aligned}$$

ahol az utolsó egyenlőtlenségnél az 1.1. lemmát, a McMillan-egyenlőtlenséget használtuk fel. ■

Miután láttuk, hogy egy X valószínűségi változó egyértelműen dekódolható kódjának átlagos kódszóhosszára a $\frac{H(X)}{\log s}$ mennyiség alsó korlátot ad, most megmutatjuk, hogy prefix kóddal ezt a korlátot jól meg lehet közelíteni.

1.3. tétel. *Létezik olyan $f : \mathcal{X} \to \mathcal{Y}^*$ prefix kód, amelyre*

$$\mathbf{E}|f(X)| < \frac{H(X)}{\log s} + 1.$$

1. BIZONYÍTÁS: **(Shannon–Fano-kód)** Feltehetjük, hogy $p(x_1) \geq p(x_2) \geq \cdots \geq p(x_{n-1}) \geq p(x_n) > 0$, mert különben az $\mathcal{X}$ elemeinek átindexelésével elérhetjük ezt. A bizonyítás technikája miatt ismét felhasználjuk az $y_i \mapsto i-1,\ i = 1,\ldots,s$ megfeleltetést. Legyenek a w_i számok a következők:

$$w_1 = 0, \qquad w_i = \sum_{l=1}^{i-1} p(x_l), \qquad i = 2,\ldots,n.$$

Kezdjük a w_i számokat felírni s-alapú számrendszerbeli tört alakban. A w_j-hez tartozó törtet olyan pontosságig írjuk le, ahol az először különbözik az összes többi w_i, $i \neq j$ szám hasonló alakbeli felírásától. Miután ily módon n darab véges hosszú törtet kapunk, az $f(x_j)$ legyen a w_j-hez tartozó tört az egészeket reprezentáló nulla nélkül. Könnyedén belátható, hogy az így kapott kód prefix. A konstrukció miatt x_j-hez létezik olyan x_k, hogy w_j és w_k végtelen tört alakjában az első $|f(x_j)|-1$ számjegy azonos. Nyilvánvaló, hogy vagy w_{j+1} vagy w_{j-1} tört alakja ilyen lesz, mivel ezek a w_j-hez legközelebbi számok. Az első esetben

$$p(x_j) = w_{j+1} - w_j < s^{-|f(x_j)|+1}, \tag{1.9}$$

a második esetben

$$p(x_{j-1}) = w_j - w_{j-1} < s^{-|f(x_j)|+1},$$

de ekkor $p(x_j) \leq p(x_{j-1})$ miatt (1.9) megint következik. Tehát mindkét esetben

$$-\log p(x_j) > (|f(x_j)|-1)\log s,$$

amiből mindkét oldalt $p(x_j)$-vel szorozva és minden j-re összegezve

$$-\sum_{j=1}^{n} p(x_j)\log p(x_j) > \left(\sum_{j=1}^{n} p(x_j)|f(x_j)| - 1\right)\log s$$

következik, így a tételt bebizonyítottuk. ■

2. BIZONYÍTÁS: Legyenek az l_i pozitív egész számok olyanok, hogy

$$-\log_s p(x_i) \leq l_i < -\log_s p(x_i) + 1, \qquad i = 1,\ldots,n \tag{1.10}$$

ahol $\log_s$ az s-alapú logaritmust jelöli. Az l_i számokat nyilván egyértelműen meg lehet így választani. A bal oldali egyenlőtlenségből következik, hogy

$$\sum_{i=1}^{n} s^{-l_i} \leq \sum_{i=1}^{n} s^{\log_s p(x_i)} = \sum_{i=1}^{n} p(x_i) = 1$$

tehát az l_i számok kielégítik az 1.2. lemma feltételét, és így létezik f prefix kód l_i hosszú kódszavakkal ($|f(x_i)| = l_i$). Ha a jobb oldali egyenlőtlenséget (1.10)-ben $p(x_i)$-vel megszorozzuk és minden i-re összegezzük, akkor azt kapjuk, hogy

$$\sum_{i=1}^{n} p(x_i) \cdot l_i < -\sum_{i=1}^{n} p(x_i) \log_s p(x_i) + \sum_{i=1}^{n} p(x_i) = \frac{H(X)}{\log s} + 1. \qquad \blacksquare$$

Az 1.1. definíció utáni megjegyzés b) pontjában említettük, hogy a betűnkénti kódolásnak van egy természetes általánosítása, a **blokk-kódolás**. Ezt formálisan egy $f : \mathcal{X}^m \to \mathcal{Y}^*$ leképezéssel definiálhatjuk, ahol tehát a forrásábécé betűiből alkotott rendezett m-eseket tekintjük forrásszimbólumoknak és ezeknek feleltetünk meg kódszavakat. A helyzet tulajdonképpen nem változik a betűnkénti kódolás esetéhez képest, hiszen egy új $\widehat{\mathcal{X}}$ forrásábécét definiálhatunk az $\widehat{\mathcal{X}} = \mathcal{X}^m$ jelöléssel, és az eddig elmondottak mind érvényben maradnak. Az egyértelmű dekódolhatóság definíciója ugyanaz marad, mint a betűnkénti kódolás esetében, az előbbieket szem előtt tartva. Legyen $\mathbf{X} = (X_1, \ldots, X_m)$ egy valószínűségi vektorváltozó, melynek koordinátái az $\mathcal{X}$-ből veszik értékeiket. Az entrópia 1.3. definíciójából jól látszik, hogy az csakis az eloszlástól függ. Mivel $\mathbf{X}$ is csak véges sok különböző értéket vehet fel, az 1.3. definíciót közvetlenül alkalmazhatjuk. Az $\mathbf{X}$ entrópiája tehát a

$$p(\mathbf{x}) = p(x_1, \ldots, x_m) = \mathbf{P}\{X_1 = x_1, \ldots, X_m = x_m\}, \qquad x_1, \ldots, x_m \in \mathcal{X},$$

jelölést bevezetve a következő:

$$\begin{aligned} H(\mathbf{X}) &= -\sum_{\mathbf{x} \in \mathcal{X}^m} p(\mathbf{x}) \log p(\mathbf{x}) = \\ &= -\sum_{x_1 \in \mathcal{X}} \cdots \sum_{x_m \in \mathcal{X}} p(x_1, \ldots, x_m) \log p(x_1, \ldots, x_m). \end{aligned}$$

Az $\mathbf{X} = (X_1, \ldots, X_m)$ entrópiájára a $H(\mathbf{X})$ jelölés mellett, ahol ez a célszerűbb, gyakran a $H(X_1, \ldots, X_m)$ jelölést fogjuk használni. Ha az $X_1, X_2, \ldots, X_m$ valószínűségi változók függetlenek, akkor $H(\mathbf{X})$ a koordináta valószínűségi változók entrópiáinak összege:

$$H(\mathbf{X}) = -\sum_{\mathbf{x} \in \mathcal{X}^m} p(\mathbf{x}) \log p(\mathbf{x}) =$$

$$
\begin{aligned}
&= -\sum_{x_1\in\mathcal{X}}\cdots\sum_{x_m\in\mathcal{X}} p_1(x_1)\cdots p_m(x_m)\log p_1(x_1)\cdots p_m(x_m) = \\
&= -\left(\sum_{x_1\in\mathcal{X}} p_1(x_1)\log p_1(x_1)+\cdots+\sum_{x_m\in\mathcal{X}} p_m(x_m)\log p_m(x_m)\right) = \\
&= \sum_{i=1}^{m} H(X_i), \qquad (1.11)
\end{aligned}
$$

ahol $p_i(x) = \mathbf{P}\{X_i = x\}, \quad i = 1,\ldots,m.$

Ha az $X_1,\ldots,X_m$ valószínűségi változók nemcsak függetlenek, hanem azonos eloszlásúak is, akkor (1.11)-ből

$$H(X_1,\ldots,X_m) = mH(X_1) \qquad (1.12)$$

következik.

A betűnkénti átlagos kódszóhossz definíciója értelemszerűen módosul a blokk-kódolás esetében: a kódszóhossz várható értékét el kell osztani az egy blokkot alkotó forrásbetűk számával, vagyis a betűnkénti átlagos kódszóhosszon a következő mennyiséget értjük:

$$\frac{1}{m}\mathbf{E}|f(\mathbf{X})| = \frac{1}{m}\sum_{\mathbf{x}\in\mathcal{X}^m} p(\mathbf{x})|f(\mathbf{x})|$$

Értelemszerűen az 1.2. tétel állítása blokk-kódolás esetén is igaz:

$$\mathbf{E}|f(\mathbf{X})| \geq \frac{H(\mathbf{X})}{\log s},$$

hiszen a tétel bizonyítása közvetlenül itt is alkalmazható.

Az 1.3. tétel következményeként megmutatjuk, hogy blokk-kódolás segítségével a betűnkénti átlagos kódszóhossz alsó korlátja tetszőlegesen közelíthető.

1.3. következmény. *Ha $X_1,\ldots,X_m$ független, X-szel azonos eloszlású valószínűségi változók, akkor létezik olyan $f:\ \mathcal{X}^m \to \mathcal{Y}^*$ prefix kód, hogy*

$$\frac{1}{m}\mathbf{E}|f(\mathbf{X})| < \frac{H(X)}{\log s} + \frac{1}{m},$$

tehát a betűnkénti átlagos kódszóhossz az m blokkhossz növelésével tetszőlegesen közelíti a $\frac{H(X)}{\log s}$ alsó korlátot.

BIZONYÍTÁS: Az 1.3. tételt felhasználva létezik olyan $f: \mathcal{X}^m \to \mathcal{Y}^*$ prefix kód, hogy

$$\sum_{\mathbf{x}\in\mathcal{X}^m} p(\mathbf{x})|f(\mathbf{x})| < \frac{H(\mathbf{X})}{\log s} + 1.$$

Ebből (1.12) miatt következik

$$\sum_{\mathbf{x}\in\mathcal{X}^m} p(\mathbf{x})|f(\mathbf{x})| < \frac{mH(X)}{\log s} + 1,$$

ahol mindkét oldalt m-mel osztva megkapjuk az állítást. ■

1.3. Optimális kódok, bináris Huffman-kód

Az 1.2. tétel alsó korlátot ad az egyértelműen dekódolható kódok átlagos kódszóhosszára, az 1.3. tétel pedig mutat egy olyan kódkonstrukciót, ahol ezt a korlátot jól megközelíthetjük. A jó kód konstrukciójának problémáját persze ennél általánosabban is felvethetjük: konstruáljuk meg az optimális, azaz legkisebb átlagos kódszóhosszú kódot, ha adott az X valószínűségi változó eloszlása. Először is gondoljuk át, hogy optimális kód valóban létezik. Ugyan véges kódábécé esetén is az egyértelműen dekódolható vagy prefix kódok halmaza végtelen, de a bizonyos átlagos kódszóhossznál (pl. $\frac{H(X)}{\log s}+1$) jobb kódok halmaza véges. Másodszor, vegyük észre, hogy az optimális kód nem feltétlenül egyértelmű; egyenlő valószínűségekhez tartozó kódszavakat felcserélhetünk, csakúgy, mint az egyenlő hosszú kódszavakat, anélkül, hogy az átlagos kódszóhosszat ezzel megváltoztatnánk.

Egy optimális kód konstruálását az 1.1. következmény értelmében egy optimális prefix kód konstruálására lehet visszavezetni. Ezért a következőkben kimondjuk az optimális prefix kódok néhány tulajdonságát. A továbbiakban az egyszerűség kedvéért a bináris, $s = 2$ esettel foglalkozunk; feltesszük, hogy $\mathcal{Y} = \{0,1\}$. Az általános, $s > 2$ eset bonyolultabb, és a dolog lényege így is jól látható.

1.4. tétel. *Ha az $f: \mathcal{X} \to \{0,1\}^*$ prefix kód optimális, és $\mathcal{X}$ elemei úgy vannak indexelve, hogy $p(x_1) \geq p(x_2) \geq \cdots \geq p(x_{n-1}) \geq p(x_n) > 0$, akkor feltehető, hogy f-re a következő három tulajdonság teljesül:*

a) $|f(x_1)| \leq |f(x_2)| \leq \cdots \leq |f(x_{n-1})| \leq |f(x_n)|$, *vagyis nagyobb valószínűségekhez kisebb kódszóhosszak tartoznak.*

b) $|f(x_{n-1})| = |f(x_n)|$, *vagyis a két legkisebb valószínűségű forrásbetűhöz tartozó kódszó egyenlő hosszú.*

c) Az $f(x_{n-1})$ és az $f(x_n)$ kódszavak csak az utolsó bitben különböznek.

BIZONYÍTÁS:

a) Tegyük fel, hogy $p(x_k) > p(x_j)$ és $|f(x_k)| > |f(x_j)|$. Ekkor x_j és x_k kódját felcserélve egy új f^* kódot vezethetünk be, amelyre

$$\begin{aligned}\sum_{i=1}^{n} p(x_i)|f(x_i)| - \sum_{i=1}^{n} p(x_i)|f^*(x_i)| = \\ = p(x_k)|f(x_k)| + p(x_j)|f(x_j)| - p(x_k)|f(x_j)| - p(x_j)|f(x_k)| = \\ = p(x_k)(|f(x_k)| - |f(x_j)|) - p(x_j)(|f(x_k)| - |f(x_j)|) = \\ = (p(x_k) - p(x_j))(|f(x_k)| - |f(x_j)|) > 0,\end{aligned}$$

tehát f nem lehet optimális.

b) Az állítás egyszerűen belátható, ha arra gondolunk, hogy $|f(x_{n-1})| < |f(x_n)|$ esetén az $|f(x_n)|$ utolsó bitjét levágva az optimálisnál kisebb átlagos kódszóhosszú kódot kapnánk, ami még mindig prefix tulajdonságú. Valóban, mivel az eredeti kódszó minden más kódszónál legalább eggyel hosszabb volt, a csonkítással kapott új kódszó az eredeti kód prefix tulajdonsága miatt nem azonos semelyik másikkal, és ugyanezen okok miatt nem is folytatása semmilyen más kódszónak.

c) Az előző gondolatmenetből világos, hogy ha létezik olyan $f(x_i)$ kódszó, hogy $f(x_i)$ és $f(x_n)$ csak az utolsó bitben különböznek, akkor az a) és b) miatt $|f(x_i)| = |f(x_{n-1})| = |f(x_n)|$. Így ha $i \neq n-1$, akkor x_i és x_{n-1} kódját felcserélve c) teljesül, és a kód optimális marad. ■

1.5. tétel. *Tegyük most fel, hogy az 1.4. tétel feltételei teljesülnek, és hogy a $\{p(x_1), p(x_2), \ldots, p(x_{n-1}) + p(x_n)\}$ valószínűségeloszláshoz ismerünk egy g optimális bináris prefix kódot. (Az x_{n-1} és x_n forrásbetűket összevonjuk egy $\bar{x}_{n-1}$ szimbólumba; $p(\bar{x}_{n-1}) = p(x_{n-1}) + p(x_n)$). Ekkor az eredeti $\{p(x_1), p(x_2), \ldots, p(x_{n-1}), p(x_n)\}$ eloszlás egy optimális f prefix kódját kapjuk, ha a $g(\bar{x}_{n-1})$ kódszót egy nullával, illetve egy egyessel kiegészítjük (a többi kódszót pedig változatlanul hagyjuk).*

BIZONYÍTÁS: A g és az f átlagos kódszóhosszát L_g-vel és L_f-fel jelölve azt kapjuk, hogy

$$L_f = L_g + p(x_{n-1}) + p(x_n).$$

Ha f nem lenne optimális, akkor a nála kisebb L_{f^*} átlagos kódszóhosszú f^* kódról az 1.4. tétel c) pontja szerint feltehetnénk, hogy $f^*(x_{n-1})$ és $f^*(x_n)$ nem rövidebbek semelyik más kódszónál, és csak az utolsó bitben különböznek egymástól. Ezt az utolsó bitet elhagyva a $\{p(x_1), p(x_2), \ldots, p(x_{n-1}) + p(x_n)\}$ eloszlásra egy g^* kódot kapnánk, amire

$$L_{g^*} = L_{f^*} - p(x_{n-1}) - p(x_n) < L_f - p(x_{n-1}) - p(x_n) = L_g$$

teljesülne, tehát az optimális g-nél jobb kódot kapnánk, ami lehetetlen. ■

Az előző tétel alapján már megadhatjuk az optimális prefix kód Huffman-féle konstrukcióját: A két legkisebb valószínűség összevonásával addig redukáljuk a problémát, amíg az triviális nem lesz, vagyis amíg összesen két valószínűségünk marad. Ezt $n-2$ lépésben érhetjük el. Ezután az összevonások megfordításával, mindig a megfelelő kódszó kétféle kiegészítésével újabb $n-2$ lépésben felépítjük az optimális kódot, a Huffman-kódot. A következő példában nyomon követhetjük az algoritmus egyes lépéseit.

1.4. példa. Legyen $n = 5$ és $p(x_1) = 0.4,\ p(x_2) = 0.3,\ p(x_3) = 0.15,\ p(x_4) = 0.1,\ p(x_5) = 0.05$, és keressük meg az ehhez tartozó Huffman-kódot:

Az egyes lépéseket az oszlopok számozása jelzi balról-jobbra, az összevonásokat a nyilak mentén lehet nyomon követni.

1.		**2.**		**3.**		**4.**
0.4		0.4		0.4		0.4
0.3		0.3		0.3	$\rightarrow$	0.6
0.15		0.15	$\rightarrow$	0.3	$\nearrow$	
0.1	$\rightarrow$	0.15	$\nearrow$			
0.05	$\nearrow$					

A 0.4 és 0.6 valószínűségek optimális prefix kódja nyilván a 0 és az 1 (vagy fordítva). Az összevonások megfordításával elvégezhetjük a Huffman-kód felépítését. A valószínűségek mellett szögletes zárójelben az egyes lépésekben kapott kódszavak állnak.

4.			**3.**			**2.**			**1.**	
0.4	[0]		0.4	[0]		0.4	[0]		0.4	[0]
0.6	[1]	$\rightarrow$	0.3	[10]		0.3	[10]		0.3	[10]
		$\searrow$	0.3	[11]	$\rightarrow$	0.15	[110]		0.15	[110]
					$\searrow$	0.15	[111]	$\rightarrow$	0.1	[1110]
								$\searrow$	0.05	[1111]

Tehát $f(x_1) = 0,\ f(x_2) = 10,\ f(x_3) = 110,\ f(x_4) = 1110,\ f(x_5) = 1111.$

Az előzőekben feltételeztük, hogy ismerjük a bemenet eloszlását. Ez azonban nincs mindig így, ekkor az eloszlást a relatív gyakoriságokkal kell becsülnünk.

A gyakorlati problémák során általában adott a $Z_1, \ldots, Z_N$ bemeneti sorozat (szöveg, fájl), amelyet szeretnénk optimálisan kódolni. Feladatunk a $\sum_{i=1}^{N} |f(Z_i)|$ minimalizálása, vagyis az optimális f kódolófüggvény megválasztása. Ez egyenértékű a kódszóhosszak átlagának, $\frac{1}{N}\sum_{i=1}^{N} |f(Z_i)|$-nek a minimalizálásával.

Belátjuk, hogy $\frac{1}{N}\sum_{i=1}^{N} |f(Z_i)| = \mathbf{E}_N|f(Z)|$, ha a várható értéket az empirikus eloszlás szerint vesszük, tehát a forrásbetűk valószínűségét a relatív gyakoriságukkal definiáljuk: $p_N(x_j) = \frac{1}{N}\sum_{i=1}^{N} I_{\{Z_i=x_j\}}$, ahol $I_{\{\,\}}$ az indikátor függvény. Nyilván

$$\begin{aligned}
\mathbf{E}_N|f(Z)| &= \sum_{j=1}^{n} p_N(x_j)|f(x_j)| = \\
&= \sum_{j=1}^{n} \frac{1}{N}\sum_{i=1}^{N} I_{\{Z_i=x_j\}}|f(x_j)| = \\
&= \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{n} I_{\{Z_i=x_j\}}|f(x_j)| = \\
&= \frac{1}{N}\sum_{i=1}^{N} |f(Z_i)|
\end{aligned}$$

(A gyakorlatban magát a kódoló függvényt is le kell írni, át kell vinni a dekódolóhoz, és ez a „fejléc” így a fentinél nagyobb átlagos kódszóhosszat eredményez. Az aszimptotikus vizsgálat során azonban ettől a konstans költségtől eltekinthetünk.)

A Huffman-kódolás fenti algoritmusát csak két lépésben tudjuk végrehajtani. Először meghatározzuk a forrásbetűk relatív gyakoriságát, ami az előzőek értelmében megegyezik a valószínűségekkel, majd ennek felhasználásával elvégezzük a tényleges kódolást. Nem mindig engedhető meg azonban olyan nagy mértékű késleltetés, hogy csak az összes bemeneti adat megérkezése után kezdünk hozzá a kimenet előállításához, másrészt a kétmenetes beolvasás akkor is lassítja az algoritmust (bár kétségkívül optimális kódot eredményez), ha a bemenet már rendelkezésre áll. A gyakorlatban ezért sokszor érdemes „egymenetes” algoritmust használni. Így az optimalitás rovására időt takaríthatunk meg. Egy forrásbetűt az előző forrásbetűk előfordulásai alapján kódolunk, s ezzel együtt lépésenként változik maga a kód is. Tehát az aktuális forrásbetű kódolását egy, az előzőleg

feldolgozott forrásbetűkre optimális kóddal hajtjuk végre. Ezt az eljárást **adaptív Huffman-kódolás**nak nevezzük.

A Huffman-kódot bináris faként is ábrázolhatjuk. A leveleket a forrásszimbólumokkal címkézzük, az éleken pedig a kódábécé ($\{0,1\}$) elemei szerepelnek. A csúcsokban a relatív gyakoriságok állnak. Egy forrásszimbólumhoz tartozó kódszót úgy kapunk meg, hogy a fa gyökerétől a megfelelő levélig húzódó út élein szereplő kódbetűket sorban összeolvassuk (konkatenáljuk). A Huffman-kódolás szemléletesen úgy történik, hogy kiindulásként felvesszük a forrásszimbólumokhoz tartozó leveleket, a csúcsokba a forrásszimbólumok relatív gyakoriságait írjuk, majd lépésenként mindig a két legkisebb értéket tartalmazó csúcs fölé teszünk egy új csúcsot (szülőt), s ebbe a két régi érték összegét írjuk. Az eljárás végén kialakul az összefüggő fa, melynek a legutolsó lépésben megkapott csúcsa lesz a gyökér.

A bináris Huffman-fában a relatív gyakoriságok szerepelnek. Adaptív Huffman-kódolás esetén ezek helyett a gyakoriságokat használhatjuk (hiszen utóbbiakat a bemenet hosszával elosztva megkapjuk a relatív gyakoriságokat, és számunkra csak az értékek egymáshoz való aránya érdekes). Két összevont betű szülőjének súlyát a szokásos módon, a két gyakoriság összegeként kapjuk.

Amennyiben a fa testvér (vagyis közös szülővel rendelkező) pontpárjait a gyökértől a levelek felé haladva fel tudjuk sorolni súlyuk szerint nemnövekvő sorrendben, a fa rendelkezik a testvérpár tulajdonsággal (sibling pair property). Bebizonyítható, hogy egy Huffman-fa akkor és csak akkor optimális, ha rendelkezik a testvérpár tulajdonsággal.

1.5. példa. Az 1.3. ábrán látható fára teljesül a testvérpár tulajdonság. A párok a következők:

$$(28,18);\ (15,13);\ (11,7);\ (7,6);\ (6,5);\ (4,3);\ (3,2);\ (2,2);\ (2,1)$$

Ezek nemnövekvő rendezése:

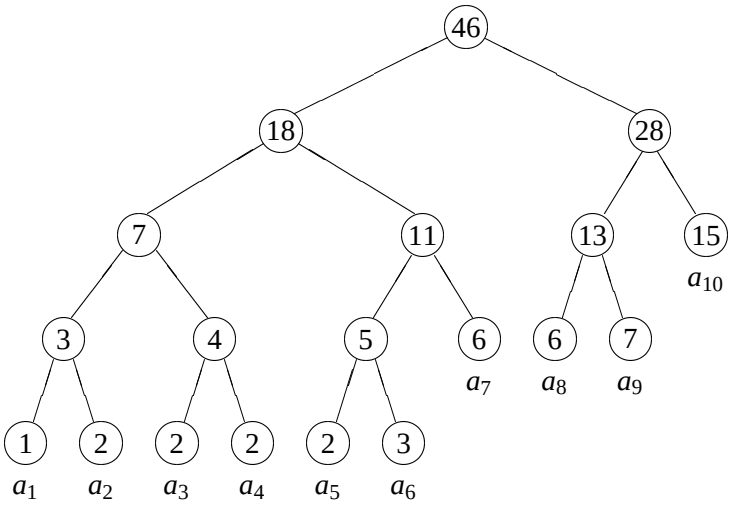
$$28 \geq 18 \geq 15 \geq 13 \geq 11 \geq 7 \geq 7 \geq 6 \geq 6 \geq$$

$$\geq 5 \geq 4 \geq 3 \geq 3 \geq 2 \geq 2 \geq 2 \geq 2 \geq 1$$

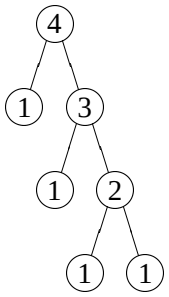
Az 1.4. ábra kódfájának párjai:

$$(3,1);\quad (2,1);\quad (1,1)$$

Ezeket nem tudjuk megfelelően sorba rendezni, így nem teljesül a fára a testvérpár tulajdonság.



1.3. ábra. Kódfa, amelyre teljesül a testvérpár tulajdonság.



1.4. ábra. Kódfa, amelyre nem teljesül a testvérpár tulajdonság.

Az adaptív algoritmus lépései a következők:
Ismerjük a forrásszimbólumokat: $\mathcal{X} = \{a_1, a_2, \ldots, a_k\}$. Kiindulásként felépítünk egy olyan Huffman-fát, amelyben — ha nincs információnk a betűk előfordulásának valószínűségéről — minden forrásszimbólum azonos, 1 gyakorisággal szerepel. Így egy kiegyensúlyozott bináris fát kapunk. (Ha ismerjük a kezdeti eloszlást, megtehetjük, hogy erre építjük fel az adaptív algoritmus kiindulási fáját.) Elküldjük a dekódolónak sorrendben (pl. a gyökértől a levelek felé, szintenként, balról jobbra) a lehetséges forrásszimbólumokat, amelyből az szintén felépíti a kiindulási Huffman-fát. (Ügyelni kell arra, hogy a kódoló és a dekódoló azonos algoritmust használjon.) A kódolás során beolvassuk a k-adik forrásszimbólumot. Ezt a

korábban beolvasott $k-1$ betű feldolgozásával kialakult, lokálisan optimális kódfával kódoljuk, majd eggyel növeljük a forrásszimbólum gyakoriságszámlálóját. Aktualizáljuk a fát, vagyis megvizsgáljuk, hogy fennáll-e még a testvérpár tulajdonság. Amennyiben nem, helyreállítjuk, így biztosítva az optimalitást. Ezeket a lépéseket a dekóder is elvégzi, tehát a következő kódbetű dekódolásakor ugyanazt a fát fogja használni, mint amelyet a kódoló használt annak előállításakor. A $(k+1)$-edik betű beolvasása után az előzőekhez hasonlóan folytatjuk.

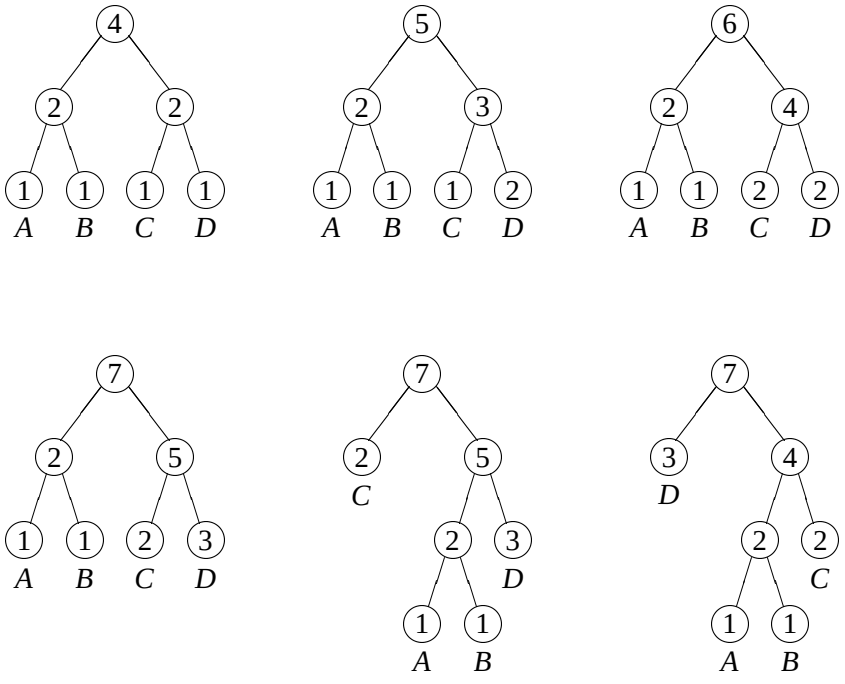
A következő két szabály használható a fa aktualizálása során a testvérpár tulajdonság helyreállítására:

1. Két, azonos súlyú csúcs a hozzá tartozó részfákkal együtt megcserélhető. Ez természetesen nem befolyásolja a súlyokat.

2. Két levél a súlyukkal együtt megcserélhető.

A szabályok kombinálásával átalakítható egy nem megfelelő kódfa egy olyanná, amely teljesíti a testvérpár tulajdonságot.

Ügyelni kell arra, hogy hosszú bemenet esetén előfordulhat a szimbólumok gyakoriságszámlálójának túlcsordulása. Ezt megfelelő technikával orvosolni kell.



1.5. ábra. Az 1.6. példa kódfájának alakulása.

1.6. példa. A bemeneti ábécénk: $\mathcal{X} = \{A, B, C, D\}$. Adaptív Huffman-kóddal kódoljuk a beérkező forrásszimbólumokat, melyek legyenek a következők: *DCDA*. Az 1.5. ábrán követhetjük nyomon a kódfa alakulását. Látható, hogy a testvérpár tulajdonság az *DCD* feldolgozása után sérül, a *D* gyakoriságszámlálójának 3-ra növelésével. Ekkor az 1. majd a 2. szabály alkalmazásával állíthatjuk helyre a fát, majd ezzel kódolható az utolsó, *A* szimbólum.

1.4. Aritmetikai kódolás

Az 1.3. tétel szerint prefix kódok használatával (például Huffman-kódolással) elérhető, hogy $\mathbf{E}|f(X)| < H(X) + 1$. Tehát karakterenként akár 1 bitet is veszíthetünk a tömörítés alsó korlátjához képest. Ezen blokkonkénti kódolással segíthetünk, ilyenkor ugyanis az 1.3. következmény értelmében $\frac{1}{m}\mathbf{E}|f(\mathbf{X})| < \frac{1}{m}H(\mathbf{X}) + \frac{1}{m}$, ahol m az $\mathbf{X}$ üzenetvektor hossza. Látható, hogy minél nagyobb m-et választunk, annál kisebb lesz a veszteség. Azonban korlátozva vagyunk m kiválasztásában, ugyanis Huffman-kód esetén a kódszó elküldése csak a teljes m hosszú blokk beolvasása után lehetséges, és emiatt nagy m esetén jelentős késleltetés keletkezhet. Továbbá a kód bonyolultsága — azaz a kódoló- és dekódolótáblák mérete, a Huffman-kód fejléce — m növelésével exponenciálisan nő, és ez praktikus szempontból is korlátot állít m növelése elé. Erre a problémára ad megoldást az **aritmetikai kódolás**, melyet kifejezetten blokkonkénti kódoláshoz alkalmaznak, és jellegzetessége, hogy valós időben történik a kódszó előállítása és visszafejtése, tehát nem lép fel jelentős késleltetés, akármilyen hosszú blokkokat is kódolunk.

Az aritmetikai kódolást itt végtelen pontosságú lebegőpontos aritmetikával mutatjuk be. A gyakorlatban természetesen csak véges pontosságú aritmetikák léteznek. A módszer ezeken is implementálható, mint majd később látni fogjuk.

A kódszó előállításához először vesszük a $[0, 1)$ intervallumot, majd a forrásábécé minden eleméhez ennek egy részintervallumát rendeljük olyan módon, hogy ezek a részintervallumok teljesen lefedjék az eredeti intervallumot, továbbá diszjunktak legyenek (vagyis partíciót alkossanak), és méretük legyen arányos a hozzájuk rendelt forráskarakter valószínűségével.

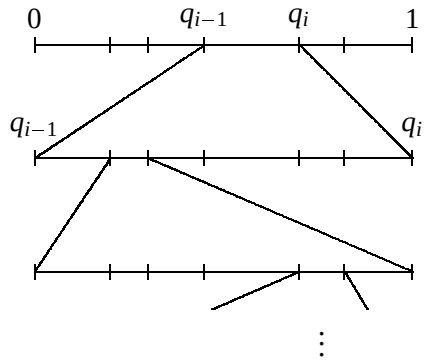
Ilyen feltételeket teljesít például a következő hozzárendelés:

$$x_i \mapsto [q_{i-1}, q_i),$$

ahol

$$q_0 = 0, \qquad q_i = \sum_{j=1}^{i} \mathbf{P}\{X = x_j\}, \qquad 1 \le i \le n,$$

n pedig a kódábécé elemszáma.



1.6. ábra. Az intervallum felosztása aritmetikai kódolás során.

Következő lépésként válasszuk az üzenet első karakterének megfelelő intervallumot. Folytassuk az eljárást úgy, hogy most ezt a kiválasztott intervallumot vágjuk részekre, ugyanúgy mint az előző lépésnél, majd válasszuk ki az üzenet második karakterének megfelelő részintervallumot, és az eljárást ezzel folytassuk tovább.

Látható, hogy az így kapott egymásba skatulyázott intervallumsorozat utolsó eleme egyértelműen meghatározza az előtte lévőket, és emiatt az utolsó elemből a teljes üzenet visszafejthető. Ez lesz maga a kódszó.

A gyakorlatban nem az intervallum alsó és felső határa, hanem egy tetszőlegesen választott eleme adja a kódszót. Ebből is visszaállítható ugyanis az üzenet. Bebizonyítható, hogy az $\mathbf{x}$ üzenetblokkhoz tartozó elemet elég $\left\lceil \log \frac{1}{p(\mathbf{x})} \right\rceil + 1$ bit pontossággal leírni — ilyen pontosság mellett az eredeti üzenetblokk rekonstruálható lesz. Ebből megkapható egy k katarteres blokkhoz tartozó kódszó hosszának várható értéke:

$$\begin{aligned}
\mathbf{E}\left(\left\lceil \log \frac{1}{p(\mathbf{X})} \right\rceil + 1\right) &= \sum p(\mathbf{x}) \left\lceil \log \frac{1}{p(\mathbf{x})} \right\rceil + 1 < \\
&< \sum p(\mathbf{x}) \left(\log \frac{1}{p(\mathbf{x})} + 1 \right) + 1 = \\
&= -\sum p(\mathbf{x}) \log p(\mathbf{x}) + \sum p(\mathbf{x}) + 1 = \\
&= H(\mathbf{X}) + 2
\end{aligned}$$

Ha az $\mathbf{X}$ üzenetvektor komponensei független azonos eloszlásúak, akkor $H(\mathbf{X}) = kH(X_1)$, tehát a betűnkénti átlagos kódszóhosszra

$$\frac{1}{k}\mathbf{E}|f(\mathbf{X})| \leq H(X_1) + \frac{2}{k},$$

azaz a k blokkhossz növelésével tetszőlegesen megközelíti $H(X_1)$-et. Fontos, hogy ellentétben a blokkonkénti Huffman-kódolással itt a bonyolultság nem nő a blokkok hosszának növelésével.

A valós idejű működés lehetőségét az teremti meg, hogy a kódszó eleje (tehát a legutolsó intervallum tetszőlegesen kiválasztott elemének bináris tört ábrázolásban vett első néhány bitje) már az első néhány forráskarakterből meghatározható.

Fixpontos aritmetikán történő implementációnál épp ezt használjuk ki. Ilyenkor ugyanis a kódolás során az intervallum alsó és felső határát tároljuk fix pontossággal. Mivel az intervallum egyre rövidül, ezek egyre közelebb kerülnek egymáshoz. Emiatt ezek magasabb helyiértékű bitjei egyezni fognak. Ilyen egyezés esetén az egyező részt átküldjük a dekódolóba, és a maradék különböző részeket felskálázzuk. Így mindig az aritmetika teljes pontossága áll a rendelkezésünkre, továbbá az átküldött bitmintából a dekódolóban hasonló skálázásos módszerrel az üzenet visszaállítható.

1.5. Az entrópia néhány tulajdonsága

Ebben a szakaszban megadjuk az entrópia néhány lényeges tulajdonságát amelyeket a későbbiek során gyakran fel fogunk használni.

Legyenek X és Y valószínűségi változók, amelyek a véges $\mathcal{X}$ illetve $\mathcal{Y}$ halmazból veszik értékeiket. $x \in \mathcal{X}$ és $y \in \mathcal{Y}$ esetén a következő jelöléseket használjuk:

$$\begin{aligned} p(x) &= \mathbf{P}\{X = x\} \\ p(y) &= \mathbf{P}\{Y = y\} \\ p(x,y) &= \mathbf{P}\{X = x, Y = y\} \\ p(x \mid y) &= \mathbf{P}\{X = x \mid Y = y\} \\ p(y \mid x) &= \mathbf{P}\{Y = y \mid X = x\}. \end{aligned}$$

1.6. tétel.

a) Ha az X valószínűségi változó n különböző értéket vehet fel pozitív valószínűséggel, akkor

$$0 \le H(X) \le \log n,$$

és a bal oldalon egyenlőség akkor és csak akkor áll fenn, ha X 1-valószínűséggel konstans, a jobb oldalon pedig akkor és csak akkor, ha X egyenletes eloszlású, azaz $p(x_i) = \frac{1}{n}$, $i = 1, \ldots, n$.

b) X és Y diszkrét valószínűségi változókra

$$H(X,Y) \le H(X) + H(Y),$$

és az egyenlőség szükséges és elégséges feltétele X és Y függetlensége. ($H(X,Y)$ az (X,Y) valószínűségi változópár együttes eloszlásához rendelt entrópiát jelöli.)

c) *Az X tetszőleges $g(X)$ függvényére*

$$H(g(X)) \leq H(X),$$

és itt az egyenlőség szükséges és elégséges feltétele az, hogy g invertálható legyen.

BIZONYÍTÁS:

a) A bal oldali egyenlőtlenség triviális (lásd az 1.3. definíció utáni megjegyzést), a jobb oldalit a Jensen-egyenlőtlenség 1.2. a) következményét a $p_i = p(x_i),\ q_i = \frac{1}{n},\ i = 1,\ldots,n$ szereposztásban felhasználva kapjuk:

$$-\sum_{i=1}^{n} p(x_i)\log p(x_i) \leq -\sum_{i=1}^{n} p(x_i)\log\frac{1}{n} = \log n.$$

b) Az állítás rövid átalakítás után a következő alakba írható:

$$-\sum_{x,y} p(x,y)\log p(x,y) \leq -\sum_{x,y} p(x,y)\log p(x)p(y)$$

(felhasználtuk, hogy $\sum_y p(x,y) = p(x)$ illetve $\sum_x p(x,y) = p(y)$), és itt megint alkalmazhatjuk a Jensen-egyenlőtlenséget. Egyenlőség akkor és csak akkor áll, ha $p(x,y) = p(x)p(y)$ minden x,y-ra ami X és Y függetlenségét jelenti.

c) Az állítást az 1.4. következménynél bizonyítjuk. ■

MEGJEGYZÉS: Mint az az entrópia tulajdonságaiból kitűnik, egy valószínűségi változó entrópiája úgy fogható fel, mint az általa reprezentált véletlen kísérlet kimenetelének bizonytalansága. Így például nulla az entrópiája (a „bizonytalansága") a biztosan bekövetkező eseményt reprezentáló 1-valószínűséggel konstans valószínűségi változónak, és maximális az entrópiája a „legbizonytalanabb kimenetű" egyenletes eloszlású valószínűségi változónak. Sok hasonló analógia található még, de nem szabad elfelejtenünk, hogy az ilyen érvekkel alátámasztott gondolatmenetek nem bizonyítások. Az entrópia igazi jelentőségét a kódolási tételekben játszott alapvető szerepe adja.

Az 1.6. szakaszban megismerjük az információforrás fogalmát, és ennek egyértelműen dekódolható kódolását változó szóhosszú kódokkal. Ehhez szükségünk lesz a feltételes entrópia bevezetésére.

1.5. definíció. *X-nek az Y feltétellel vett **feltételes entrópiá**ja a következő:*

$$H(X \mid Y) = -\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x,y)\log p(x \mid y). \tag{1.13}$$

MEGJEGYZÉS:

a) A feltételes entrópia (1.13) kifejezését átalakítva azt kapjuk, hogy

$$H(X \mid Y) = -\sum_{y\in\mathcal{Y}} p(y)\sum_{x\in\mathcal{X}} p(x \mid y)\log p(x \mid y),$$

amiből látható, hogy $H(X \mid Y)$ az X valószínűségi változó $Y = y$ feltételekkel vett feltételes eloszlásaihoz tartozó

$$H(X \mid Y = y) = -\sum_{x\in\mathcal{X}} p(x \mid y)\log p(x \mid y)$$

típusú entrópiáinak várható értéke.

b) Mivel az 1.5. definícióban csak $\mathcal{X}$ és $\mathcal{Y}$ végességét kötöttük ki, a feltételes entrópia definícióját véges értékkészletű valószínűségi vektorváltozókra is kiterjeszthetjük. Legyen $\mathbf{X} = (X_1,\ldots,X_k)$ és $\mathbf{Y} = (Y_1,\ldots,Y_m)$, ahol az X_i-k és Y_i-k véges halmazokból veszik értékeiket. Ekkor

$$\begin{aligned} H(\mathbf{X} \mid \mathbf{Y}) &= H(X_1,\ldots,X_k \mid Y_1,\ldots,Y_m) = \\ &= -\sum_{\mathbf{y}}\sum_{\mathbf{x}} p(\mathbf{x},\mathbf{y})\log p(\mathbf{x} \mid \mathbf{y}), \end{aligned}$$

ahol $\mathbf{x} = (x_1,\ldots,x_k)$ és $\mathbf{y} = (y_1,\ldots,y_m)$ minden lehetséges értékére kell összegezni.

1.7. tétel (A feltételes entrópia tulajdonságai). *X, Y, Z legyenek véges értékkészletű valószínűségi változók. Ekkor*

a)

$$H(X,Y) = H(Y) + H(X \mid Y) = H(X) + H(Y \mid X).$$

b)

$$0 \le H(X \mid Y) \le H(X),$$

és a bal oldalon egyenlőség akkor és csak akkor teljesül, ha X 1-valószínűséggel függvénye Y-nak, a jobb oldali egyenlőség akkor és csak akkor teljesül, ha X és Y függetlenek.

c)

$$H(X \mid Z,Y) \leq H(X \mid Z),$$

és egyenlőség akkor és csak akkor áll, ha $p(x \mid z,y) = p(x \mid z)$ *minden olyan* x,y,z*-re amelyre* $p(x,y,z) > 0$*, ami azt jelenti, hogy* X *és* Y *feltételesen függetlenek, ha* Z *értéke adott (azaz* X,Z,Y *egy Markov-lánc).*

d) *Az* Y *valószínűségi változó minden* f *függvényére*

$$H(X \mid Y) \leq H(X \mid f(Y)),$$

és egyenlőség pontosan akkor áll fenn, ha minden rögzített z*-re* $p(x \mid y) = \mathbf{P}\{X = x \mid f(Y) = z\}$ *minden* x*-re és* y*-ra, amelyre* $f(y) = z$ *és* $p(y) > 0$*.*

e) *Az* $X_1, X_2, \ldots, X_n$ *valószínűségi változók együttes entrópiáját megkaphatjuk az ún. láncszabály segítségével:*

$$\begin{aligned} H(X_1,\ldots,X_n) &= H(X_1) + H(X_2 \mid X_1) + H(X_3 \mid X_1,X_2) + \cdots \\ &\quad \cdots + H(X_n \mid X_1,\ldots,X_{n-1}). \end{aligned}$$

BIZONYÍTÁS:

a)

$$\begin{aligned} H(X,Y) &= -\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x,y)\log p(x,y) = \\ &= -\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x,y)\log\left(p(x \mid y)p(y)\right) = \\ &= -\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x,y)\log p(y) - \sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x,y)\log p(x \mid y) = \\ &= H(Y) + H(X \mid Y), \end{aligned}$$

mert $\sum_{y\in\mathcal{Y}} p(x,y) = p(x)$. A $H(X,Y) = H(X) + H(Y \mid X)$ egyenlőség a bizonyítás szimmetriájából következik.

b) A bal oldali egyenlőtlenség az entrópia 1.6. a) tulajdonságából és az 1.5. definíció utáni megjegyzésből következik. $H(X \mid Y) = 0$ pontosan akkor, ha $H(X \mid Y = y) = 0$ minden olyan y-ra, amire $p(y) > 0$, vagyis akkor, ha minden ilyen y-ra $p(x \mid y) = 1$ pontosan egy x-re. Ez azt jelenti, hogy X az Y függvénye (1-valószínűséggel).

A jobb oldali $H(X \mid Y) \leq H(X)$ egyenlőtlenség az a) tulajdonság szerint ekvivalens a $H(X,Y) \leq H(X) + H(Y)$ egyenlőtlenséggel, amit az 1.6. b) pontban bebizonyítottunk.

c) Az állítás ekvivalens a következővel:

$$-\sum_{y,z} p(y,z)\left(\sum_{x} p(x \mid y,z)\log\frac{p(x \mid z)}{p(x \mid y,z)}\right) \geq 0,$$

amit könnyen beláthatunk, ha észrevesszük (Jensen-egyenlőtlenség), hogy a zárójelben levő összeg minden y,z-re nempozitív. Az egyenlőség feltétele a szokásos módon következik.

d) Mivel

$$\mathbf{P}\{f(Y)=z\} = \sum_{y:\, f(y)=z} p(y)$$

és

$$\mathbf{P}\{X=x, f(Y)=z\} = \sum_{y:\, f(y)=z} p(x,y),$$

ezért a Jensen-egyenlőtlenség 1.2. b) következménye szerint

$$-\mathbf{P}\{X=x, f(Y)=z\}\log\frac{\mathbf{P}\{X=x, f(Y)=z\}}{\mathbf{P}\{f(Y)=z\}} \geq$$
$$\geq -\sum_{y:\, f(y)=z} p(x,y)\log\frac{p(x,y)}{p(y)},$$

amit minden z-re és x-re összegezve megkapjuk az állítást. Az egyenlőség feltétele az, hogy rögzített z-re $\frac{p(x,y)}{p(y)} = p(x \mid y) =$ konstans minden y-ra, amelyre $f(y)=z$, és a konstans nyilván egyenlő
$\mathbf{P}\{X=x \mid f(Y)=z\}$-vel.

e) Az a) tulajdonság szerint

$$H(X_1,\ldots,X_n) = H(X_1,\ldots,X_{n-1}) + H(X_n \mid X_1,\ldots,X_{n-1}),$$

amelyből az eljárást tovább folytatva az állítás teljes indukcióval következik. ■

1.4. következmény. *Adósak vagyunk még az 1.6. tétel c) állításának igazolásával, vagyis azzal, hogy az X valószínűségi változó tetszőleges g függvényére*

$$H(g(X)) \leq H(X),$$

és az egyenlőség akkor és csak akkor áll fenn, ha g invertálható.

BIZONYÍTÁS: Az 1.7. tétel a) és b) pontja szerint

$$H(X, g(X)) = H(g(X)) + H(X \mid g(X)) \geq H(g(X)), \tag{1.14}$$

és a jobb oldalon egyenlőség pontosan akkor van, ha X a $g(X)$ függvénye, vagyis ha g invertálható. De 1.7. a)-t és b)-t újra használva, azt is tudjuk, hogy

$$H(X, g(X)) = H(X) + H(g(X) \mid X) = H(X),$$

és ebből (1.14)-gyel együtt következik az állítás. ■

1.6. Információforrások változó szóhosszúságú kódolása

Az **információforrás**t $\mathbb{X}$ betűvel jelöljük, és az $X_1, X_2, \ldots$ valószínűségi változók végtelen sorozatával modellezzük, azaz a forrás az i-edik időpillanatban az X_i jelet bocsátja ki. Az X_i valószínűségi változók mindegyike ugyanabból a véges $\mathcal{X} = \{x_1, \ldots, x_n\}$ halmazból, a forrásábécéből veszi az értékét. Az $\mathbb{X}$ forrást statisztikai tulajdonságaival jellemezzük, vagyis adottnak vesszük, ha minden véges dimenziós eloszlását ismerjük. Egy $\mathbb{X}$ forrást emlékezetnélkülinek vagy memóriamentesnek mondunk, ha az $X_1, X_2, \ldots$ valószínűségi változók függetlenek. Az $\mathbb{X}$ forrás stacionárius, ha $X_1, X_2, \ldots$ stacionárius sztochasztikus folyamat, vagyis ha az $X_1, X_2, \ldots, X_n$ és az $X_{k+1}, X_{k+2}, \ldots, X_{k+n}$ valószínűségi változók együttes eloszlása megegyezik minden pozitív n-re és k-ra, tehát a véges dimenziós eloszlások az „időeltolásra” invariánsak. Az $\mathbb{X}$ stacionárius forrás ergodikus, ha tetszőleges $f(x_1, \ldots, x_k)$ függvényre az $\frac{1}{n} \sum_{i=1}^{n} f(X_i, X_{i+1}, \ldots, X_{i+k-1})$ 1-valószínűséggel konvergál az $\mathbf{E} f(X_1, \ldots, X_k)$ várható értékhez, amennyiben az véges.

A kódoló a forrás által kibocsátott szimbólumokat a csatornán átvihető szimbólumokká alakítja. Jelöljük a csatorna által használt szimbólumok véges, s-elemű halmazát $\mathcal{Y}$-nal (kódábécé). Ebben a szakaszban az előzőekben megismert egyértelműen dekódolható kódokat használjuk, betűnkénti, illetve blokkonkénti kódolással. A kódolás során a forrás által produkált szimbólumsorozatot k-hosszú blokkokra vágjuk (betűnkénti kódolásnál $k = 1$), és a blokkokat változó szóhosszú kódolással (pl. prefix kód) kódoljuk. A kapott kódszavakat egymás mellé írva kapjuk a csatornán továbbítható kódjelsorozatot.

Feltesszük, hogy a használt kód egyértelműen dekódolható (lásd az 1.1. definíciót), így a felhasználó visszakapja a forrás üzenetét. A kód kiválasztásánál az a célunk, hogy L, az egy forrásbetűre eső átlagos kódszóhossz minél kisebb legyen.

Tételezzük most fel, hogy az $\mathbb{X} = X_1, X_2, \ldots$ emlékezetnélküli és stacionárius, vagyis az $X_1, X_2, \ldots$ független, azonos eloszlású valószínűségi változók sorozata.

Legyen $f : \mathcal{X} \to \mathcal{Y}^*$ egyértelműen dekódolható kód, és kódoljuk $\mathbb{X}$-et betűnként f-fel. Ekkor egy k-hosszú üzenetet kódolva a betűnkénti átlagos kódszóhossz az X_i-k azonos eloszlása miatt független k-tól:

$$L = \frac{1}{k}\mathbf{E}\left(|f(X_1)| + \cdots + |f(X_k)|\right) = \mathbf{E}|f(X_1)|$$

Az 1.2. tétel szerint

$$\mathbf{E}|f(X_1)| \geq \frac{H(X_1)}{\log s} \tag{1.15}$$

ahol s a kódábécé elemszáma, $H(X_1)$ pedig az X_1 — és az azonos eloszlás miatt minden X_i — entrópiája. Az ilyen forrás betűnkénti kódolással való „tömörítésére" tehát elvi korlátot kaptunk. Másfelől viszont tudjuk (1.3. tétel), hogy van olyan f prefix kód, amelyre

$$\mathbf{E}|f(X_1)| < \frac{H(X_1)}{\log s} + 1,$$

tehát az alsó korlátot megközelíthetjük. Kódoljuk most $\mathbb{X}$ k-hosszú blokkjait egyszerre, vagyis használjuk az egyértelműen dekódolható $f : \mathcal{X}^k \to \mathcal{Y}^*$ kódot. Ekkor az 1.2. tétel szerint

$$L = \frac{1}{k}\mathbf{E}|f(X_1,\ldots,X_k)| \geq \frac{1}{k}\frac{H(X_1,\ldots,X_k)}{\log s},$$

amiből az X_i-k függetlensége miatt (1.12) szerint

$$L \geq \frac{H(X_1)}{\log s}$$

következik, hasonlóan (1.15)-höz, tehát a blokk-kódolás nem módosította az alsó korlátot. Azonban az 1.3. következmény megmutatta, hogy ezt az alsó korlátot megfelelően nagy blokkhosszt alkalmazva tetszőlegesen megközelíthetjük, ugyanis minden k-ra létezik olyan L betűnkénti átlagos kódszóhosszú $f : \mathcal{X}^k \to \mathcal{Y}^*$ prefix kód, hogy

$$L < \frac{H(X_1)}{\log s} + \frac{1}{k}. \tag{1.16}$$

Amennyiben általánosabb információforrások kódolását akarjuk vizsgálni, szükségünk lesz a forrás entrópiájának fogalmára:

1.6. definíció. *Az $\mathbb{X} = X_1, X_2, \ldots$ forrás **forrásentrópiá**ja a*

$$H(\mathbb{X}) = \lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, \ldots, X_n)$$

mennyiség, amennyiben a határérték létezik.

A következő tétel megmutatja, hogy egy stacionárius forrásnak mindig létezik az entrópiája.

1.8. tétel. *Ha az $\mathbb{X} = X_1, X_2, \ldots$ forrás stacionárius, akkor létezik az entrópiája, és*

$$H(\mathbb{X}) = \lim_{n\to\infty} H(X_n \mid X_1, X_2, \ldots, X_{n-1}).$$

BIZONYÍTÁS: Az 1.7. e) tulajdonságból

$$\frac{1}{n}H(X_1, X_2, \ldots, X_n) = \frac{1}{n}\left(H(X_1) + \sum_{i=2}^{n} H(X_i \mid X_1, X_2, \ldots, X_{i-1})\right). \qquad (1.17)$$

Másrészt a forrás stacionaritása miatt

$$\begin{aligned} H(X_i \mid X_1, X_2, \ldots, X_{i-1}) &= H(X_{i+1} \mid X_2, X_3, \ldots, X_i) \geq \\ &\geq H(X_{i+1} \mid X_1, X_2, \ldots, X_i) \end{aligned}$$

ahol az egyenlőtlenség az 1.7. c) tulajdonság következménye.
Így a $H(X_n \mid X_1, X_2, \ldots, X_{n-1})$ sorozat n-nel monoton csökkenő és nemnegatív, tehát létezik a $H = \lim_{n\to\infty} H(X_n \mid X_1, X_2, \ldots, X_{n-1})$ határérték.

A bizonyítás második felében szükségünk lesz az egyszerűen bizonyítható Toeplitz-lemmára, amely azt mondja ki, hogy ha az $\{a_n\}$ valós számsorozat a véges a számhoz konvergál ($\lim_{n\to\infty} a_n = a$), akkor a $b_n = \frac{1}{n}\sum_{k=1}^{n} a_k$ sorozatnak is ugyanez az a szám a határértéke. A Toeplitz-lemmát az $a_n = H(X_n \mid X_1, X_2, \ldots, X_{n-1})$ szereposztásban felhasználva, (1.17)-ből azt kapjuk, hogy

$$\lim_{n\to\infty} \frac{1}{n}H(X_1, X_2, \ldots, X_n) = \lim_{n\to\infty} H(X_n \mid X_1, X_2, \ldots, X_{n-1}).$$

Ekkor viszont (1.17) szerint az $\frac{1}{n}H(X_1, X_2, \ldots, X_n)$ sorozat is monoton csökkenő. ■

MEGJEGYZÉS: Ha $\mathbb{X}$ emlékezetnélküli és stacionárius, akkor a forrásentrópia létezése triviális, hiszen felhasználva az entrópia tulajdonságát, valamint az X_i-k függetlenségét és azonos eloszlását, azt kapjuk, hogy

$$H(\mathbb{X}) = \lim_{n\to\infty} \frac{1}{n}H(X_1, X_2, \ldots, X_n) = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} H(X_i) = H(X_1).$$

A forrásentrópia fogalmát felhasználva már kimondhatjuk a változó szóhosszúságú kódolás tételét stacionárius forrásokra:

1.9. tétel. *Ha az $\mathbb{X} = X_1, X_2, \ldots$ stacionárius forrást k-blokkhosszal blokkonként kódoljuk az $f : \mathcal{X}^k \to \mathcal{Y}^*$ egyértelműen dekódolható kóddal, akkor a kód L betűnkénti átlagos kódszóhosszára mindig fennáll az*

$$L \geq \frac{H(\mathbb{X})}{\log s}$$

egyenlőtlenség. A k blokkhosszt elég nagyra választva létezik olyan f kód, amelynek L betűnkénti átlagos kódszóhossza tetszőlegesen megközelíti ezt az alsó korlátot.

BIZONYÍTÁS: Az 1.2. tétel és az $\frac{1}{k}H(X_1, X_2, \ldots, X_k)$ monoton csökkenő konvergenciája miatt

$$L = \frac{1}{k}\mathbf{E}|f(X_1, \ldots, X_k)| \geq \frac{1}{k}\frac{H(X_1, \ldots, X_k)}{\log s} \geq \frac{H(\mathbb{X})}{\log s}$$

tehát a tétel első felét beláttuk. Legyen most $\varepsilon > 0$ tetszőleges valós szám és válasszuk k_0 pozitív egészet úgy, hogy

$$\frac{1}{k_0}H(X_1, \ldots, X_{k_0}) - H(\mathbb{X}) < \frac{\varepsilon}{2}\log s, \qquad \text{és} \qquad \frac{1}{k_0} < \frac{\varepsilon}{2}. \tag{1.18}$$

Mivel a Shannon–Fano-konstrukcióval (1.3. tétel) készíthetünk olyan $f : \mathcal{X}^{k_0} \to \mathcal{Y}^*$ prefix kódot, hogy

$$\mathbf{E}\left(|f(X_1, \ldots, X_{k_0})|\right) < \frac{H(X_1, \ldots, X_{k_0})}{\log s} + 1,$$

ezért (1.18)-at felhasználva azt kapjuk, hogy

$$\begin{aligned} L &= \frac{1}{k_0}\mathbf{E}\left(|f(X_1, \ldots, X_{k_0})|\right) < \\ &< \frac{1}{k_0}\frac{H(X_1, \ldots, X_{k_0})}{\log s} + \frac{1}{k_0} < \\ &< \frac{H(\mathbb{X})}{\log s} + \frac{\varepsilon}{2} + \frac{1}{k_0} < \\ &< \frac{H(\mathbb{X})}{\log s} + \varepsilon. \end{aligned}$$

Mivel ε tetszőlegesen kicsi lehet, ezzel megmutattuk, hogy a betűnkénti átlagos kódszóhossz alsó korlátját tetszőlegesen megközelíthetjük. ■

1.7. Markov-forrás és entrópiája

A forrásentrópia kiszámítása általános forrásokra sokszor reménytelen feladat. A következőkben a források egy olyan osztályát — a Markov-forrásokat — ismerjük meg, amelyekre a forrásentrópiát sok esetben ki tudjuk számolni. Meg kell jegyezni, hogy a Markov-források jelentőségét az adja, hogy segítségükkel gyakorlati „információforrások” (írott szöveg, beszéd) modellezhetők kezelhető módon. A Markov-források ismertetését a Markov-láncok fogalmának felelevenítésével kezdjük.

A $Z_1, Z_2, \ldots$ valószínűségi változók Markov-láncot alkotnak, ha

$$\begin{aligned}\mathbf{P}\{Z_k = z_k \mid Z_1 = z_1, Z_2 = z_2, \ldots, Z_{k-1} = z_{k-1}\} = \\ = \mathbf{P}\{Z_k = z_k \mid Z_{k-1} = z_{k-1}\}\end{aligned} \tag{1.19}$$

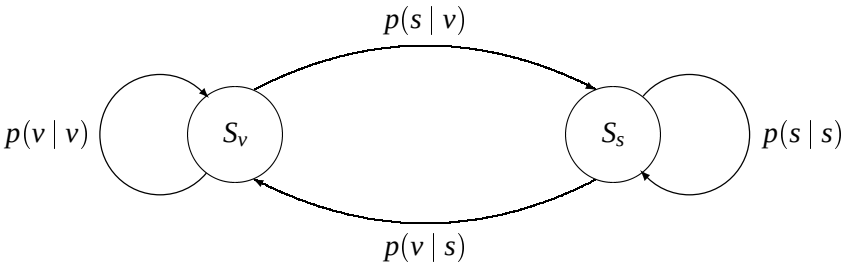
minden $k \geq 2$-re és minden lehetséges $z_1, z_2, \ldots, z_k$ sorozatra. A Z_i értékeit a Markov-lánc állapotainak, az állapotok $\mathcal{Z}$ halmazát pedig a Markov-lánc állapotterének nevezzük. Feltesszük, hogy $|\mathcal{Z}| < \infty$, vagyis az állapottér véges. A $Z_1, Z_2, \ldots$ Markov-lánc homogén, ha

$$\mathbf{P}\{Z_k = z_2 \mid Z_{k-1} = z_1\} = \mathbf{P}\{Z_2 = z_2 \mid Z_1 = z_1\}$$

minden $z_1, z_2 \in \mathcal{Z}$-re és minden $k \geq 2$-re. Végül, egy Markov-lánc stacionárius, ha mint sztochasztikus folyamat stacionárius.

A $\mathbf{P}\{Z_k = z_2 \mid Z_{k-1} = z_1\}$ átmenetvalószínűségek ismeretében a Markov-modell jól használható szövegek és képek tömörítésére. Angol nyelvű szövegek entrópiájának meghatározására elsőként Shannon végzett kísérleteket. A nyelvi redundanciát modellezte Markov-láncokkal. A 26 betűs angol ábécét használó szövegek esetén másodrendű Markov-lánc modell használatával 3.1 bit/betű tömörítési arányt ért el. Ez 2.4 bit/betű-re szorítható le, ha betűk helyett szavak szerepelnek a modellben. Az angol nyelv entrópiájának becslésére Shannon kísérleti személyeket is alkalmazott a statisztikai modellek helyett. A résztvevőknek a szöveg aktuális betűjét kellett kitalálniuk a megelőző 100 betűs környezet alapján. Így alsó határnak 0.6 bit/betű, míg felsőnek 1.3 bit/betű adódott. Minél hosszabb környezetet vizsgálunk, annál jobb a jóslás eredménye, azonban a környezet hosszával exponenciálisan növekszik a lehetséges megelőző állapotok száma.

1.7. példa. Egy fekete-fehér kép sorait modellezzünk egy $\{Z_i\}$ stacionárius forrásként, az 1.7. ábrán látható homogén, stacionárius Markov-lánccal. S_v állapot jelöli azt, hogy az aktuális képpont világos, míg S_s, hogy sötét. $p(v)$ a világos,



1.7. ábra. Egy fekete-fehér kép Markov-lánc modellje.

$p(s)$ a sötét képpont kezdeti valószínűsége. Az átmenetvalószínűségek közül például $p(v \mid s)$ jelenti annak az eseménynek a valószínűségét, amikor egy sötét képpont után egy világos következik.

Számítsuk ki egy tetszőleges Markov-lánc forrásentrópiáját:

$$H(\mathbb{Z}) = \lim_{n\to\infty} \frac{1}{n} H(Z_1, Z_2, \ldots, Z_n) =$$

mivel $\mathbb{Z}$ stacionárius, ezért

$$= \lim_{n\to\infty} H(Z_n \mid Z_1, \ldots, Z_{n-1}) =$$

de $\mathbb{Z}$ Markov-lánc is, ezért az (1.19) Markov-tulajdonság miatt

$$= \lim_{n\to\infty} H(Z_n \mid Z_{n-1}) =$$

és $\mathbb{Z}$ stacionaritását újra felhasználva azt kapjuk, hogy

$$\begin{aligned} &= \lim_{n\to\infty} H(Z_2 \mid Z_1) = \\ &= H(Z_2 \mid Z_1) = \\ &= -\sum_{z_1}\sum_{z_2} p(z_1, z_2) \log p(z_2 \mid z_1) = \\ &= -\sum_{z_1}\sum_{z_2} p(z_2 \mid z_1) p(z_1) \log p(z_2 \mid z_1). \end{aligned}$$

1.8. példa. Legyenek az 1.7. példa valószínűségei a következőek:

$$p(s \mid s) = 0.88, \quad p(v \mid s) = 0.12, \quad p(s \mid v) = 0.03, \quad p(v \mid v) = 0.97$$

Ekkor a stacionárius eloszlás:

$$p(s) = 0.2, \quad p(v) = 0.8$$

A Markov-lánc (vagyis az átmenetvalószínűségek) figyelembe vétele nélkül egy betű entrópiája:

$$H(Z_1) = -0.8\log 0.8 - 0.2\log 0.2 = 0.722$$

Ugyanez a modellünk szerint:

$$\begin{aligned} H(Z_2 \mid Z_1) &= -\sum_{z_1}\sum_{z_2} p(z_2 \mid z_1)p(z_1)\log p(z_2 \mid z_1) = \\ &= -p(s \mid s)p(s)\log p(s \mid s) - p(v \mid s)p(s)\log p(v \mid s) \\ &\quad -p(s \mid v)p(v)\log p(s \mid v) - p(v \mid v)p(v)\log p(v \mid v) = \\ &= -0.88\cdot 0.2\cdot\log 0.88 - 0.12\cdot 0.2\cdot\log 0.12 \\ &\quad -0.03\cdot 0.8\cdot\log 0.03 - 0.97\cdot 0.8\cdot\log 0.97 = \\ &= 0.261 \end{aligned}$$

Látható, hogy nagyjából a harmadára csökkent az entrópia az átmenetvalószínűségek ismerete miatt.

Számoljuk ki egy speciális Markov-lánc entrópiáját:

1.9. példa. Jelöljük p_{ij}-vel a $\mathbf{P}\{Z_2 = z_j \mid Z_1 = z_i\}$ átmenetvalószínűséget, és legyen a $[p_{ij}]$ $n \times n$-es mátrix olyan, hogy minden sora ugyanazon $u_1, u_2, \ldots, u_n$ $(u_i > 0,\ \sum u_i = 1)$ számok egy permutációja. Ekkor

$$H(Z_2 \mid Z_1) = -\sum_{z_1 \in \mathcal{Z}} p(z_1) \sum_{z_2 \in \mathcal{Z}} p(z_2 \mid z_1)\log p(z_2 \mid z_1).$$

Az átmenetvalószínűség-mátrix soraira vonatkozó feltétel miatt azonban

$$-\sum_{z_2 \in \mathcal{Z}} p(z_2 \mid z_1)\log p(z_2 \mid z_1) = -\sum_{i=1}^{n} u_i \log u_i$$

z_1 értékétől függetlenül, tehát

$$H(Z_2 \mid Z_1) = -\sum_{i=1}^{n} u_i \log u_i.$$

Nézzük azt az esetet, amikor $\mathbb{Z}$ bináris, szimmetrikus Markov-lánc, azaz $\mathcal{Z} = \{0,1\}$ és

$$[p_{ij}] = \begin{bmatrix} p & 1-p \\ 1-p & p \end{bmatrix} \qquad (0 < p < 1).$$

Itt

$$H(Z_2 \mid Z_1) = -p\log p - (1-p)\log(1-p) =: h(p). \tag{1.20}$$

A $h(p)$ függvényt bináris entrópiafüggvénynek nevezzük. Ha $\mathbf{P}\{Z_1 = 0\} = q$ és $\mathbf{P}\{Z_1 = 1\} = 1-q$, akkor a stacionaritás miatt a $\mathbf{P}\{Z_2 = 0\} = q$ és $\mathbf{P}\{Z_2 = 1\} = 1-q$ feltételeknek is teljesülniük kell, tehát

$$\left[\begin{array}{cc} q, & 1-q \end{array}\right] \cdot \left[\begin{array}{cc} p & 1-p \\ 1-p & p \end{array}\right] = \left[\begin{array}{cc} q, & 1-q \end{array}\right]$$

amiből $q = \frac{1}{2}$ következik, és így $H(Z_1) = 1$. Megvizsgálva (1.20)-at észrevehetjük, hogy ha p közel van 0-hoz illetve 1-hez, akkor $H(\mathbb{Z}) = H(Z_2 \mid Z_1)$ értéke $H(Z_1) = H(Z_2) = 1$-nél, a lehetséges maximális értéknél sokkal kisebb. Ez intuitíve érthető is, hiszen ha

$$p = \mathbf{P}\{Z_2 = 0 \mid Z_1 = 0\} = \mathbf{P}\{Z_2 = 1 \mid Z_1 = 1\}$$

1-hez közeli, akkor hosszú 0-s illetve 1-es sorozatokat várhatunk, míg ha

$$1-p = \mathbf{P}\{Z_2 = 1 \mid Z_1 = 0\} = \mathbf{P}\{Z_2 = 0 \mid Z_1 = 1\}$$

1-hez közeli, akkor a 0-k és 1-esek közel szabályos váltakozását várhatjuk.

Az 1.8. példa átmenetvalószínűségei jellegzetesek egy túlnyomóan szöveget tartalmazó (pl. üzleti) dokumentum esetén. Megfigyelhető, hogy a következő képpont színe sokkal nagyobb valószínűséggel lesz azonos az előzőével, mint eltérő (különösen világos esetében). Ahelyett, hogy a képpontok színét külön-külön kódolnánk, kódoljuk egyszerűen az azonos színű képpontok (vagyis a futamok) hosszát, tehát azt a hosszt, amíg a Markov-lánc azonos állapotában maradunk. Ezt a technikát **futamhossz kódolás**nak nevezzük. (A futamhossz kódolás optimalitására vonatkozóan lásd az 1.17. feladatot.) Például, ha 190 világos pixelt 30 sötét követ, majd 210 világos jön, a 430 képpont egyenkénti kódolása helyett a $190, 30, 210$ sorozatot fogjuk kódolni, valamint jeleznünk kell azt, hogy az első pontsorozat milyen színű volt.

Futamhossz kódolást alkalmaznak a CCITT (ma: ITU-T) fax-szabványaiban is. Számunkra a Group 3 illetve Group 4 ajánlások érdekesek, ugyanis a korábbi Group 1 és 2 technikák csak analóg módszereket használtak, ennélfogva nem tömörítettek.

Az 1980-ban megjelent Group 3 szabvány egydimenziós futamhossz kódolással dolgozik. Ez azt jelenti, hogy az egymás alatti vízszintes sorokat egymástól függetlenül kódolja, a futamok pedig az egy soron belül váltakozó fehér és fekete képpontokból állnak. Minden sor első futama fehér képpontokból áll; ha egy sor fekete pixellel kezdődik, akkor az első futamot egy 0 hosszúságú fehér futamnak

kell tekinteni. A különböző hosszúságú futamok eltérő valószínűséggel fordulnak elő egy dokumentumban, ezért ezeket változó szóhosszúságú kóddal, mégpedig a szabvány szerint Huffman-kóddal kódolják. A futamok összhossza, vagyis egy sor hosszúsága 1728 képpont. Ez túl sok lehetséges futamhosszt eredményez, nincs értelme ilyen nagyméretű kódkönyvet (= kódszavak halmaza) alkalmazni. Ezért a h hosszt 1 vagy 2 jegyű, 64-es számrendszerben felírt számként kódolják:

$$h = 64m + t \qquad t = 0, 1, \ldots, 63 \qquad m = 0, 1, \ldots, 27$$

Külön kódtáblázat vonatkozik az m, vagyis a kiegészítő kód (make up code, MUC) illetve a t, vagyis a lezáró kód (terminating code, TC) értékeire, külön a fekete és külön a fehér képpontok esetére. Egy futam kódját a színnek megfelelő MUC illetve TC táblázatból kiolvasott kódszavak konkatenációja adja. Azonos színhez tartozó MUC és TC táblázatok prefix tulajdonságúak (együttesen is). Mivel a fekete és fehér futamok mindig váltakozva szerepelnek, ezért a fekete táblázatokban álló kódszavak lehetnek a fehér táblázatokban állók prefixei és fordítva. A sor végét a speciális EOL kódszó jelöli, ez biztosítja az adó és vevő közötti szinkronizációt is.

1984-ben publikálta a CCITT a Group 4 ajánlást, amely kihasználja a függőleges irányú redundanciát is, emellett felülről kompatíbilis a Group 3-mal. Egy soron belül a futamokat nem csak a már megismert módon, a futamhosszak felsorolásával kódolhatjuk, hanem a színátmenetek helyének pozícióival is.

A kétdimenziós kódolás megértéséhez vezessük be az alábbi jelöléseket:

a_0: Az utolsó pixel, amelynek értékét a kódoló és a dekódoló egyaránt ismeri. Egy sor kódolásának megkezdésekor az a_0 egy képzeletbeli fehér képpontra mutat, amely az első aktuális pixel bal oldalán áll.

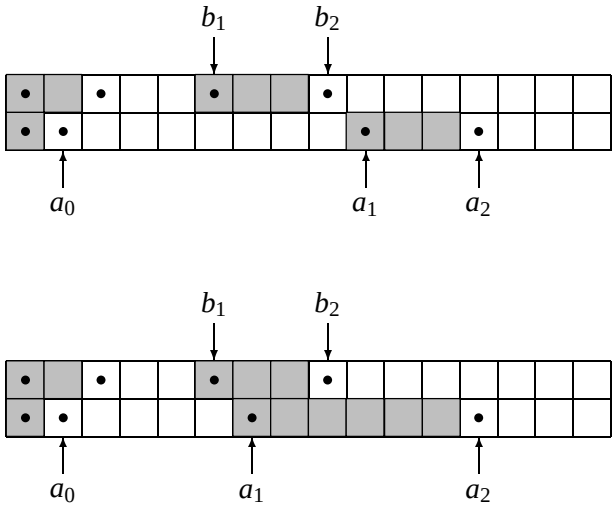
a_1: Az első színátmenetet jelentő képpont a_0 jobb oldalán. a_1 színe ellentétes a_0 színével. a_1 helye csak a kódoló számára ismert.

a_2: Az második színátmenetet jelentő képpont a_0 jobb oldalán. a_2 színe ellentétes a_1 színével, ami azt jelenti, hogy megegyezik a_0 színével.

b_1: Az első színátmenetet jelentő képpont az aktuálisan kódolandó sort megelőző sorban a_0 jobb oldalán, amelynek színe ellentétes a_0 színével. Mivel a megelőző sor és a_0 értéke ismert a kódoló és a dekódoló számára is, ezért b_1 helye is ismert mindkettőjük előtt.

b_2: Az első színátmenetet jelentő képpont az aktuálisan kódolandó sort megelőző sorban b_1 jobb oldalán.

A Group 4 algoritmusa az első sort ugyanúgy kódolja, mint a Group 3 esetében láttuk. A további sorok kódolásához felhasználja az azt megelőzőt, így a másodikhoz az elsőt, a harmadikhoz a másodikat, stb. Az 1.8. ábra azt a helyzetet ábrázolja, amikor éppen a második sornál tartunk a feldolgozásban, és a képponto-



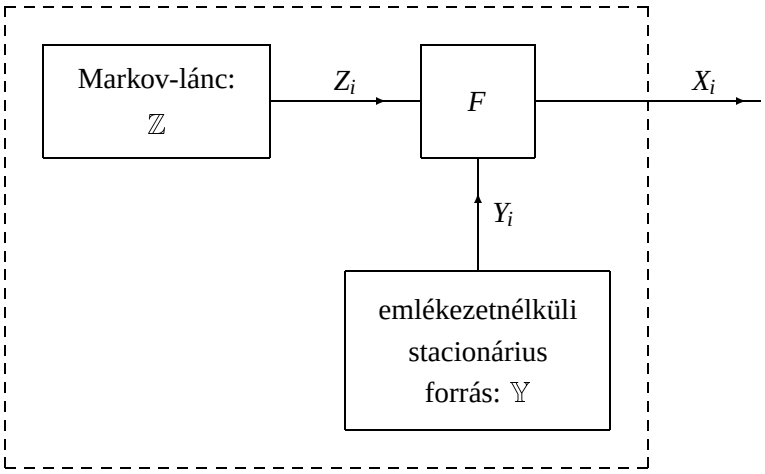
1.8. ábra. Az a_0, a_1, a_2, b_1, b_2 mutatók lehetséges elhelyezkedése.

kat a második pixelig már feldolgoztuk. A színátmenetet jelentő pixeleket ponttal jelöltük. Két esetet kell megkülönböztetnünk:

Ha b_1 és b_2 a_0 és a_1 között fekszik, a kódoláshoz az átadó (pass) módot használjuk. A kódoló egy speciális kódszó kiküldésével értesíti a dekódolót erről. Ebből a dekódoló tudja, hogy az a_0-tól a b_2 alatti pixelig a képpontok színe azonos. Ha ez nem lenne igaz, akkor közben lenne egy színátmenetet jelentő képpont, vagyis a_1 és b_2 viszonyára nem lenne igaz a feltételünk. Ekkor a kódoló és a dekódoló által egyaránt ismert legutolsó képpont a b_2 által mutatott lesz. Így ez lesz az a_0 új helye, a másik négy mutató új pozícióját pedig a már ismert módon jelöljük ki.

Ha a_1 megelőzi b_2-t, ismét két eset lehetséges: Ha a_1 és b_1 távolsága nem nagyobb 3-nál, elküldjük ezt a távolságértéket (ezt függőleges módnak nevezzük). (A kódszó természetesen függ attól, hogy a_1 jobbra vagy balra van-e b_1-től, így itt összesen 7 eset lehetséges azt is beleértve, hogy a_1 éppen b_1 alatt helyezkedik el.) a_0-t a_1-re állítjuk, módosítjuk a másik négy mutatót is, és folytatjuk a kódolást az algoritmus elejétől. Ha a_1 és b_1 távolsága nagy, lényegében visszatérünk az egydimenziós technikához. Egy speciális kódszóval jelezzük a dekódernek, hogy vízszintes módban vagyunk, majd elküldjük a_0 és a_1 illetve a_1 és a_2 távolságát Huffman-kódolva. a_0-t a_2 helyére állítjuk, és aktualizáljuk a másik négy mutatót is. A kódolást az algoritmus elejétől folytatjuk.

A kétdimenziós algoritmus használatával egy sor kódolása a megelőző soron alapul, így elképzelhető, hogy egy sorban bekövetkező hiba kiterjed a többire



1.9. ábra. Markov-forrás.

is. Ezt megelőzendő, a szabvány rögzíti, hogy minden egydimenziós eljárással kódolt sort normál függőleges felbontás esetén legfeljebb 1, nagy felbontás esetén legfeljebb 3 kétdimenziós algoritmussal kódolt sor követhet.

A sorvégeket a Group 3 szabványban megismert EOL szimbólum jelzi, azonban attól függően, hogy vízszintes módban, vagyis az egydimenziós Group 3 szerint kódoltuk a sort, vagy az új kétdimenziós módszerrel, egy 1 illetve 0 bit követi. Ez lehetővé teszi, hogy mindkét módszerrel elvégezve a kódolást az adó a rövidebb hosszúságot eredményezőt választhassa ki, majd a megfelelő EOL jellel jelezze a vevőnek, hogy melyik algoritmust használta.

A Markov-lánc modell általánosításaként vezessük be a Markov-forrást, amely egy stacionárius Markov-lánc emlékezetnélküli sztochasztikus leképezése.

1.7. definíció. *Legyen $\mathbb{Z} = Z_1, Z_2, \ldots$ egy homogén és stacionárius Markov-lánc, és legyen $\mathbb{Y} = Y_1, Y_2, \ldots$ egy emlékezetnélküli, stacionárius forrás, amely független $\mathbb{Z}$-től. Tegyük fel továbbá, hogy adott egy $F(z,y)$ kétváltozós függvény. Ekkor az $X_i = F(Z_i, Y_i)$ leképzéssel definiált $\mathbb{X} = X_1, X_2, \ldots$ forrást* ***Markov-forrás****nak nevezzük.*

Az 1.9. ábra a Markov-forrás származtatását szemlélteti.

Mivel $Z_1, Z_2, \ldots$ és $Y_1, Y_2, \ldots$ stacionáriusak és függetlenek, ezért az $X_1, X_2, \ldots$ is stacionárius lesz, tehát létezik a $H(\mathbb{X}) = \lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, \ldots, X_n)$ forrásentrópia.

Próbáljuk meg kiszámolni $H(\mathbb{X})$-et. A jól ismert összefüggés szerint (amelyet az 1.7. tétel a) egyenletének átrendezésével kapunk)

$$\begin{aligned} H(X_1,\dots,X_n) &= H(Z_1,\dots,Z_n) \\ &\quad +H(X_1,\dots,X_n \mid Z_1,\dots,Z_n) \\ &\quad -H(Z_1,\dots,Z_n \mid X_1,\dots,X_n). \end{aligned} \tag{1.21}$$

Vegyük sorra az egyenlőség jobb oldalán szereplő tagokat:

a) Korábban már láttuk, hogy

$$\lim_{n\to\infty}\frac{1}{n}H(Z_1,Z_2,\dots,Z_n) = H(Z_2 \mid Z_1). \tag{1.22}$$

b) A $H(X_1,\dots,X_n \mid Z_1,\dots,Z_n)$ kiszámításához tekintsük a szóban forgó feltételes valószínűségeket:

$$\begin{aligned} &\mathbf{P}\{X_1=x_1,\dots,X_n=x_n \mid Z_1=z_1,\dots,Z_n=z_n\} = \\ &= \mathbf{P}\{F(Z_1,Y_1)=x_1,\dots,F(Z_n,Y_n)=x_n \mid Z_1=z_1,\dots,Z_n=z_n\} = \\ &= \mathbf{P}\{F(z_1,Y_1)=x_1,\dots,F(z_n,Y_n)=x_n\} = \\ &= \prod_{i=1}^{n}\mathbf{P}\{F(z_i,Y_i)=x_i\} = \\ &= \prod_{i=1}^{n}\mathbf{P}\{F(Z_i,Y_i)=x_i \mid Z_i=z_i\} = \\ &= \prod_{i=1}^{n}\mathbf{P}\{X_i=x_i \mid Z_i=z_i\}, \end{aligned} \tag{1.23}$$

ahol a második és negyedik egyenlőségnél a $(Z_1,\dots,Z_n)$ és az $(Y_1,\dots,Y_n)$ függetlenségét, a harmadik egyenlőségnél pedig az Y_i-k függetlenségét használtuk ki. Az egyszerűség kedvéért a szokásos jelöléseinket felhasználva az (1.23) szerint tehát

$$\log p(x_1,\dots,x_n \mid z_1,\dots,z_n) = \sum_{i=1}^{n}\log p(x_i \mid z_i),$$

és így

$$\begin{aligned} &H(X_1,\dots,X_n \mid Z_1,\dots,Z_n) = \\ &\quad = -\sum_{x_1,\dots,x_n}\sum_{z_1,\dots,z_n} p(x_1,\dots,x_n,z_1,\dots,z_n)\sum_{i=1}^{n}\log p(x_i \mid z_i) = \end{aligned}$$

$$= -\sum_{i=1}^{n} \sum_{x_1,\ldots,x_n} \sum_{z_1,\ldots,z_n} p(x_1,\ldots,x_n,z_1,\ldots,z_n) \log p(x_i \mid z_i) =$$

$$= \sum_{i=1}^{n} H(X_i \mid Z_i). \tag{1.24}$$

Mivel az (X_i, Z_i) párok eloszlása ugyanaz minden i-re, ezért (1.24)-ből

$$\frac{1}{n} H(X_1,\ldots,X_n \mid Z_1,\ldots,Z_n) = H(X_1 \mid Z_1) \tag{1.25}$$

következik.

c) A $H(Z_1,\ldots,Z_n \mid X_1,\ldots,X_n)$ feltételes entrópiát általános esetben nem tudjuk kiszámolni. Nézzünk két speciális esetet, ahol ez a számítás nem okoz nehézséget:

1. Legyen $X_i = Z_i$, vagyis $F(z,y) = z$ minden y-ra, tehát a Markov-forrás kimenete a $\mathbb{Z}$ Markov-lánc állapotai. Ekkor

$$H(Z_1,\ldots,Z_n \mid X_1,\ldots,X_n) = H(Z_1,\ldots,Z_n \mid Z_1,\ldots,Z_n) = 0,$$

és

$$H(X_i \mid Z_i) = H(Z_i \mid Z_i) = 0,$$

tehát (1.22)-ből és (1.25)-ből (1.21) szerint

$$H(\mathbb{X}) = H(Z_2 \mid Z_1),$$

ahogy már korábban láttuk.

2. Legyenek az $F_z(\cdot) = F(z,\cdot)$ függvények értékkészletei minden z-re diszjunktak, vagyis (az X_i-k értékeinek halmazát $\mathcal{X}$-szel, Y_i-k értékeinek halmazát $\mathcal{Y}$-nal jelölve), ha

$$A_z = \{x \in \mathcal{X} : \ x = F(z,y) \text{ valamely } y \in \mathcal{Y}\text{-ra}\},$$

akkor

$$A_{z_1} \cap A_{z_2} = \emptyset, \ \text{ ha } \ z_1 \neq z_2, \ \text{ és } \ \bigcup_{z \in \mathcal{Z}} A_z = \mathcal{X}.$$

Ekkor nyilvánvalóan létezik olyan G függvény, hogy

$$Z_i = G(X_i),$$

hiszen ha $X_i \in A_z$, akkor tudjuk, hogy $Z_i = z$. Tehát a feltételes entrópia 1.7. b) tulajdonsága szerint

$$H(Z_1,\dots,Z_n \mid X_1,\dots,X_n) = \\ = H(G(X_1),\dots,G(X_n) \mid X_1,\dots,X_n) = 0,$$

és így

$$H(\mathbb{X}) = H(Z_2 \mid Z_1) + H(X_1 \mid Z_1).$$

A c) 1. és 2. esetekben tehát kiszámíthatjuk a Markov-forrás entrópiáját ha ismerjük a $\mathbb{Z}$ Markov-lánc állapotátmenetvalószínűség-mátrixát, az Y_1 eloszlását és az F függvényt.

Meg kell jegyezni, hogy az X_i-k feltételes eloszlásainak (1.23) tulajdonsága azt jelenti, hogy az $Y_1, Y_2, \dots$ és az F függvény együttesen egy diszkrét memóriamentes csatornát alkotnak, melynek bemenetei a Z_i-k. A diszkrét memóriamentes csatorna fogalmával a 3. fejezetben foglalkozunk majd.

1.8. Univerzális forráskódolás

Az eddig vizsgált kódok alkalmazásakor az adó és a vevő között átvitelre kerülő bitek két csoportot alkotnak. Először átvisszük a blokk-kódot leíró információt. Ez egy állandó költséget jelent, független az üzenet tényleges hosszától. Majd következnek az üzenet kódszavai. Elméleti vizsgálataink során azzal a feltételezéssel éltünk, hogy a továbbítandó üzenetünk végtelen hosszú. Ilymódon, a kódok aszimptotikus viselkedését tekintve, az állandó költség fajlagosan nullához tart, tehát elhanyagolható. A gyakorlatban azonban véges forrásokkal van dolgunk. Ebben az esetben az állandó költség akár nagyobb is lehet, mint az üzenet kódszavainak összhosszúsága. Ezt elkerülendő, jó lenne, ha rendekezésünkre állna egy olyan technika, amelynek nincs állandó költsége, de aszimptotikusan ugyanolyan jó tömörítési arányt ér el, mint a blokk-kódok. Az állandó költség abból adódik, hogy a kódot a forráson előzetesen elvégzett statisztikai vizsgálatok (a forrásszimbólumok gyakorisága) alapján hozzuk létre, tehát ezek az adatok szükségesek a kód leírásához. Ehelyett járjunk el úgy, hogy menet közben gyűjtünk információt a forrásszimbólumokról, vagyis az aktuális szimbólumot az ezt megelőző szimbólumok alapján kódoljuk. Az ilyen kódokat **adaptív kódok**nak nevezzük, alkalmazásuk során nincs állandó költség. Korábban találkoztunk már ilyen módszerrel az adaptív Huffman-kód kapcsán. A most tárgyalásra kerülő Lempel–Ziv-kódok is ebbe a családba tartoznak.

Az első LZ-algoritmus az 1977-ben publikált LZ77.

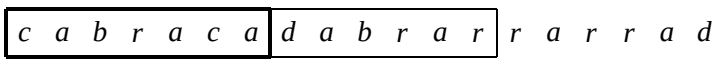
Az LZ77 algoritmus

A kódoló a forrásszimbólumok sorozatát egy h_a hosszú csúszóablakon keresztül vizsgálja. Az ablak két részből áll: egy keresőpufferből, amely a legutóbb kódolt h_k darab forrásszimbólumot tartalmazza, és egy előretekintő pufferből, amely a következő h_e darab kódolandó szimbólumot tartalmazza ($h_a = h_k + h_e$). A kódoló a keresőpufferben megkeresi az előretekintő puffer első szimbólumával megegyező szimbólumokat. Ehhez egy hátrafelé haladó mutatót használ. Megnézi, hogy a megtalált pozíciókkal kezdődően, a keresőpufferben lévő szimbólumok milyen hosszan egyeznek meg az előretekintő puffer szimbólumaival, és a találatok közül azt választja ki, amelytől kezdve a leghosszabb az egyezés. A kódoló ezután elküld egy $\langle t, h, c \rangle$ hármast, ahol t a keresőpufferben megtalált szimbólum távolsága az előretekintő puffertől (offset), h a kereső- és az előretekintő puffer egyező szimbólumainak legnagyobb hosszúsága, c pedig az első, az előretekintő pufferben lévő nem egyező karakter kódszava. Azért küldjük el az első nem egyező karakter kódját is, hogy kezeljük azt az esetet, amikor az előretekintő puffer szimbólumait nem találjuk meg a keresőpufferben. Ilyenkor t és h értéke 0. Egy hármas kódolásához állandó hosszúságú kód használatával $\lceil \log h_k \rceil + \lceil \log h_e \rceil + \lceil \log |\mathcal{X}| \rceil$ bit szükséges, ahol $|\mathcal{X}|$ a forrásábécé mérete. Figyeljük meg, hogy az egyező szimbólumok hosszúságának átviteléhez nem $\lceil \log h_k \rceil$, hanem $\lceil \log h_e \rceil$ bit szükséges. Ennek oka, hogy az egyezés hossza meghaladhatja a keresőpuffer hosszát, vagyis az egyező rész átlóghat az előretekintő pufferbe.

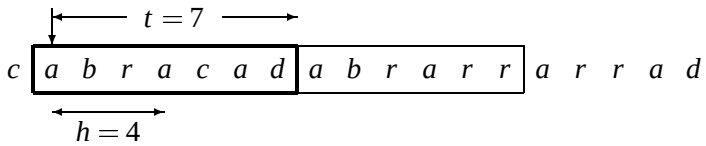
1.10. példa. Legyen a bementünk a következő:

$$\ldots cabracadabrarrarrad \ldots$$

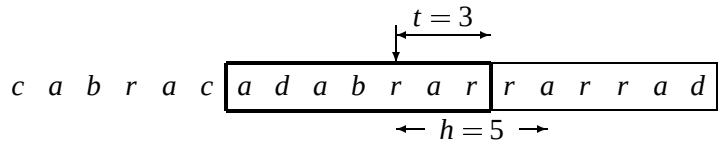
$h_a := 13,\ h_k := 7,\ h_e := 6.$ Tegyük fel, hogy az első néhány karaktert már kódoltuk. Ekkor:



Látható, hogy az előretekintő puffer első karaktere, d, nem található meg a keresőpufferben. Átküldjük a $\langle 0, 0, f(d) \rangle$ hármast, ahol $f(d)$ a d karakter kódját jelöli. Az ablakot eggyel jobbra mozgatjuk, így:



A mutatót a keresőpufferben hátrafelé mozgatva, az előretekintő puffer első szimbólumával (a) egyező karaktert $t = 2$ távolságra találjuk meg. Ekkor az egyezés hosszúsága $h = 1$. Tovább haladva a mutatóval $t = 4$-nél szintén egy 1 hosszú egyezés adódik. Végül $t = 7$-nél találjuk meg a legjobb választást $h = 4$ hosszal. Tehát az *abrar* karaktereket a $\langle 7, 4, f(r)\rangle$ hármassal kódoljuk, és az ablakot 5 pozícióval jobbra toljuk, így:



Az első egyezést $t = 1$-nél $h = 1$ hosszan találjuk. A második egyezés $t = 3$-nál van, hossza első ránézésre $h = 3$. Azonban az egyezés az előretekintő pufferbe is átnyúlik, ezért $h = 5$. Az átküldendő hármas: $\langle 3, 5, f(d)\rangle$. A dekódolás során ez az „átlógás” nem okoz gondot, mert az első három karaktert könnyen megkapjuk a már előzőleg dekódolt karakterekből, a maradék kettőt pedig az előbbi lépés során megkapott 3 karakter segítségével nyerjük.

Láthatjuk, hogy az LZ77 egy rendkívül egyszerű adaptív algoritmus, amely nem igényel előzetes ismeretet vagy feltevést a forrásról. Megmutatható, hogy az eljárás hatékonysága aszimptotikusan ($h_k, h_e \to \infty$) megközelíti az optimális algoritmusét, amely előzetesen ismeri a forráseloszlást, azaz stacionárius és ergodikus forrás esetén az átlagos kódszóhossz konvergál $\frac{H(\mathbb{X})}{\log s}$-hez, ha $h_k, h_e \to \infty$. Bár ez aszimptotikusan igaz, a gyakorlatban az LZ77 számos továbbfejlesztése ismeretes, amelyek célja a hatékonyság növelése. Például a népszerű PKZIP és ARJ tömörítőkben a hármasokat nem fix, hanem változó hosszúságú kóddal kódolják. Egy másik variáció változtatható méretű kereső és előretekintő ablakot használ. Az LZ77 legegyszerűbb módosítása annak kiküszöbölése, amikor egyetlen karaktert kódolunk egy hármassal. Ez egy jelzőbittel oldható meg. Ezzel jelezzük, hogy nem egy hármast, hanem csak egy kódszót küldünk át.

Az LZ77 alkalmazása során a forrásszimbólumok legutóbb kódolt sorozatát használjuk, így azzal a feltételezéssel élünk, hogy a minták egymáshoz közeli intervallumokban visszatérnek (a mozgó ablakon belül). Szélsőséges esetben, ha az ismétlődés hossza éppen eggyel hosszabb a keresőpuffer méreténél, nem tudunk tömöríteni. Az LZ-algoritmus következő, 1978-as veriójánál (LZ78) ezt a problémát egy másfajta, adaptív szótár alapú rendszerrel oldják fel.

Az LZ78 algoritmus

A kódoló és a dekódoló is szótárt épít az előzőleg előfordult sorozatokból.

a kódoló kimenete	szótár index	szótár bejegyzés	a kódoló kimenete	szótár index	szótár bejegyzés
$\langle 0, f(d)\rangle$	1	d	$\langle 4, f(c)\rangle$	10	bac
$\langle 0, f(a)\rangle$	2	a	$\langle 9, f(b)\rangle$	11	$dabb$
$\langle 0, f(b)\rangle$	3	b	$\langle 8, f(d)\rangle$	12	acd
$\langle 3, f(a)\rangle$	4	ba	$\langle 0, f(e)\rangle$	13	e
$\langle 0, f(c)\rangle$	5	c	$\langle 13, f(c)\rangle$	14	ec
$\langle 1, f(a)\rangle$	6	da	$\langle 1, f(e)\rangle$	15	de
$\langle 3, f(b)\rangle$	7	bb	$\langle 14, f(d)\rangle$	16	ecd
$\langle 2, f(c)\rangle$	8	ac	$\langle 13, f(e)\rangle$	17	ee
$\langle 6, f(b)\rangle$	9	dab			

1.10. ábra. Az 1.11. példa LZ78 kódolásának menete.

A kódoló megkeresi a forrásszimbólumok aktuális pozíciójától kezdődő leghosszabb egyezést a szótárban. Átküld egy $\langle i, c\rangle$ párt, ahol i az egyező karaktersorozat szótárbeli indexét jelöli, c pedig az első nem egyező karakter kódja, majd felveszi a szótárba az i indexű egyező karaktersorozat és a c karakter konkatenációjaként kapott sztringet (a következő szabad indexet adja neki). Ha nem talál egyező karaktersorozatot a szótárban, akkor a $\langle 0, c\rangle$ párost küldi át, c itt is az első nem egyező karakter kódja, amely ebben az esetben természetesen az első feldolgozandó szimbólum.

1.11. példa. Kódoljuk a következő sorozatot az LZ78 algoritmussal:

dabbacdabbacdabbacdabbacdeecdeecdee

Kezdetben a szótár üres, ezért az első 3 szimbólumot egyenként felvesszük a szótárba, és a 0 indexszel átküldjük: $\langle 0, f(d)\rangle, \langle 0, f(a)\rangle, \langle 0, f(b)\rangle$. A negyedik szimbólum a b, amely szerepel a szótárban, a következővel együtt (*ba*) viszont már nem, ezért átküldjük a $\langle 3, f(a)\rangle$ párost, amelyből a 3 jelöli a b indexét, $f(a)$ pedig a következő karakter, vagyis az a kódját. A *ba* sorozatot felvesszük a szótárba, indexe 4 lesz. Így folytatjuk az eljárást, az eredményt az 1.10. ábrán látható táblázatban foglaltuk össze. Látható, hogy a szótárbeli bejegyzések egyre hosszabbak, és ha a bemeneti sorozat ismétlődik, akkor előbb-utóbb az egész sztring szerepelni fog a szótárban.

Megmutatható, hogy az LZ78 egy betűre jutó átlagos kódszóhossza konvergál $\frac{H(\mathbb{X})}{\log s}$-hez minden stacionárius és ergodikus forrásra.

Az LZ78 algoritmus egyik hibája, hogy a szótár folyamatosan, korlát nélkül növekszik. A gyakorlatban egy bizonyos határon túl gátat szabunk a növekedésnek: vagy rendszeresen eltávolítjuk a felesleges vagy ritkán használt bejegyzéseket, vagy egy idő után fix szótárasként működik tovább az eljárás.

Az LZW algoritmus

Terry Welch az LZ78 módosításával egy olyan technikát dolgozott ki, amellyel megtakarítható az $\langle i, c\rangle$ párból a c karakterkód átküldése. Ez az ún. LZW algoritmus. A kódoló tehát csak szótárbeli indexeket küld át. Ehhez szükséges, hogy a szótárban már a kiinduló állapotban is szerepeljen az összes egybetűs szimbólum a forrásábécéből. A kódolás során az aktuális pozíciótól kezdve addig olvassuk be a forrásszimbólumokat a p pufferbe, amíg a sorozat szerepel a szótárban. Ha az a karakter az első olyan, amelyre pa nincs benne a szótárban (az egymás után írással a konkatenációt jelöltük), akkor átküldjük a p sorozat indexét, a pa sorozatot felvesszük a szótárba és az a karaktertől kezdve folytatjuk az eljárást.

1.12. példa. Kódoljuk az LZW algoritmussal az előző sorozatunkat:

$$dabbacdabbacdabbacdabbacdeecdeecdee$$

A forrásábécé $\mathcal{X} = \{a, b, c, d, e\}$, kezdetben ez az 5 bejegyzés szerepel a szótárban. A kódoló először veszi a d karaktert. Ez benne van a szótárban, így hozzáilleszti a következő, az a karaktert. A da sorozat már nem szerepel a szótárban, ezért átküldi a d indexét, vagyis a 4-et, felveszi a szótárba a da sorozatot a 6. helyre, és megy tovább az a-val kezdve. Az a szerepel a szótárban, így hozzáveszi a b-t. ab nincs bent, tehát átküldi a indexét, a 2-t, felveszi ab-t, és folytatja az eljárást b-től, stb. Az 1.11. ábrán látható táblázat tartalmazza a kódolás végeztével a szótárban található indexeket és karaktersorozatokat. A kódoló kimenete a következő:

$$4, 1, 2, 2, 1, 3, 6, 8, 10, 12, 9, 11, 7, 16, 4, 5, 5, 11, 21, 23, 5$$

1.13. példa. Dekódoljuk az LZW algoritmussal tömörített, $\mathcal{A} = \{a, b\}$ forrásábécé feletti $abababab\ldots$ karaktersorozatot. A kódoló kimenetéről a dekódoló bemenetére az $1, 2, 3, 5, 4, 7, 6, 9, 8, \ldots$ sorozat jut el. A kiindulási szótár tartalmazza az a és a b bejegyzést. Így az $1, 2$ sorozatot dekódoljuk a illetve b karakterként. A szótárba felvesszük az ab bejegyzést harmadiknak, és a következő sorozat, amely a szótárba kerül majd, a b karakterrel fog kezdődni. A 3 kódszó érkezik a bemeneten, ezt dekódoljuk ab-ként. Először az a betűt illesztjük a készülő új szótárbejegyzés kezdő b betűjéhez. Mivel a ba sorozat nincs a szótárban,

index	bejegyzés
1	*a*
2	*b*
3	*c*
4	*d*
5	*e*
6	*da*
7	*ab*
8	*bb*
9	*ba*
10	*ac*
11	*cd*
12	*dab*
13	*bba*
14	*acd*
15	*dabb*
16	*bac*
17	*cda*
18	*abb*
19	*bacd*
20	*de*
21	*ee*
22	*ec*
23	*cde*
24	*eec*
25	*cdee*

1.11. ábra. Az 1.12. példa szótára.

felvesszük azt. A következő új bejegyzés *a* betűvel fog kezdődni. Mivel az *ab* párnak csak a kezdő *a* betűjét használtuk fel, a maradék *b* betűt hozzáillesztjük a készülő új szótárbejegyzéshez, így *ab*-t kapunk. Ez már szerepel a szótárban, ezért tovább folytatjuk a dekódolást. Az első 4 bejegyzés készen van, míg az 5. éppen készülőfélben, a következő bemenet pedig az 5 kódszó, amely a még nem teljesen kész bejegyzésre hivatkozik. Ennek ellenére tovább tudjuk folytatni a dekódolást! Ha ismernénk az 5. bejegyzést, annak első két karaktere *ab* lenne. Illesszük az *a* karaktert a készülőben lévő új bejegyzéshez. Mivel az így kapott *aba* még nem szerepel a szótárban, ez lesz az 5. bejegyzés. A következő bejegyzés *a* betűvel fog kezdődni, és még megmaradt a *ba* sorozat az előző dekódolásból, stb. Az alábbi táblázat tartalmazza a kódoló és a dekódoló által épített szótárt:

index	bejegyzés
1	*a*
2	*b*
3	*ab*
4	*ba*
5	*aba*
6	*abab*
7	*bab*
8	*baba*
9	*ababa*
⋮	

A Unix COMPRESS parancsa és a GIF (Graphics Interchange Format) képtömörítő eljárás is az LZW algoritmust használja, mégpedig adaptív szótárművelettel. A COMPRESS esetében kezdetben a szótárban 512 bejegyzésnek van hely, ez azt

jelenti, hogy a kódszavak 9 bit hosszúak. Amikor a szótár betelik, méretét megduplázzuk, 1024 bejegyzésre. Ettől kezdve 10 bites kódszavakat viszünk át, és így tovább. A kódszavak lehetséges maximális hosszát a felhasználó állíthatja be 9 és 16 bit között (alapértelmezés: 16 bit). Ezt elérve, a COMPRESS eljárás statikus szótár alapú technikává válik. Ilyenkor a program figyeli a tömörítési arányt. Amennyiben ez egy bizonyos küszöb alá esik, a szótár már nem felel meg céljainknak, ezért a szótárépítő folyamat kezdődik elölről. Így a szótár mindig tükrözi a forrás lokális jellemzőit.

Szintén az LZW algoritmusra épül a CCITT (ma: ITU-T) V.42bis tömörítési szabványa, amely a telefonhálózaton modemekkel történő adatátvitelről szóló V.42 ajánlás kiegészítése. Az algoritmus két üzemmódot definiál. Az egyik a transzparens mód, amikor az adatok tömörítetlen formában kerülnek átvitelre, a másik pedig a tömörített mód. A két üzemmódra azért van szükség, mert lehetséges, hogy az átvitelre kerülő adatokban nincs redundancia, ezért nem tömöríthető az LZW algoritmussal. Ebben az esetben a tömörítő eljárás még hosszabb kimenetet eredményezne, mint a bemenet (ez a helyzet például, ha egy előzőleg már tömörített fájlt akarunk átvinni a telefonvonalon). Tömörített üzemmódban a rendszer LZW algoritmust használ változó méretű szótárral. A szótár kezdeti méretét a kapcsolat létrejöttekor egyezteti az adó- és a vevőberendezés. A V.42bis ajánlás minimum 512 bejegyzés méretű szótárat tartalmaz, de 2048 méretűt tart ideálisnak. Az összes bejegyzés nem használható fel szabadon, mert van 3 kitüntetett szerepű kódszó. Ezek illetve jelentésük a következő: Enter Transparent Mode (üzemmódváltás: ettől kezdve a transzparens mód érvényes), Flush Data (a szótárépítést előről kezdjük), Increment Codeword Size (megduplázzuk a szótár méretét, s ezzel együtt eggyel növeljük a kódszavak méretét is). Az adatátvitel során bekövetkező hibák hatásának csökkentésére az ajánlás meghatározza a maximális sztringméretet, amely szerepelhet a szótárban. Ezt 6 és 250 között az adó- és a vevőberendezés határozza meg a kapcsolat felépítésekor (alapértelmezés: 6).

1.9. Feladatok

1.1. feladat (Egyértelmű dekódolhatóság alternatív definíciója). Nevezzünk egy $f : \mathcal{X} \to \mathcal{Y}^*$ kódot egyértelműen dekódolhatónak, ha az $\mathbf{u} = u_1 \cdots u_k$ és $\mathbf{v} = v_1 \cdots v_k$ üzenetekre (itt $u_1, v_1, \ldots, u_k, v_k \in \mathcal{X}$)

$$f(u_1) f(u_2) \cdots f(u_k) = f(v_1) f(v_2) \cdots f(v_k)$$

esetén $u_i = v_i$ minden i-re. Tehát az 1.1. definícióval ellentétben csak azt követeljük meg, hogy bármely két különböző, azonos hosszúságú üzenet kódja is különbözzön. Bizonyítsa be, hogy a két definíció ekvivalens!

1.2. feladat (Az optimális kód átlagos szóhossza). Mutassa meg, hogy az optimális bináris kód átlagos kódszóhossza tetszőlegesen közel lehet $H(X)+1$-hez. Pontosabban, bármely kis $\varepsilon > 0$ számhoz adjon meg egy olyan eloszlást az $\mathcal{X}$ forrásábécén, hogy az optimális bináris kód átlagos kódszóhosszára

$$\mathbf{E}|f(X)| > H(X) + 1 - \varepsilon$$

teljesüljön.

1.3. feladat (Egyenlőség a Kraft-egyenlőtlenségben). Nevezzünk egy f prefix kódot teljesnek, ha bármely új kódszó hozzáadásával a kód elveszti prefix tulajdonságát. Egy **x** sztringet dekódolhatatlannak nevezünk, ha nem lehet kódszavak egymás után írásával olyan sztringet kapni, amelynek **x** a prefixe. Mutassa meg, hogy a következő három állítás ekvivalens:

a) f teljes,

b) nem létezik f-re nézve dekódolhatatlan sztring,

c) $\sum_{i=1}^{n} s^{-l_i} = 1$, ahol s a kódábécé elemszáma, l_i az i-edik kódszó hossza, és n a kódszavak száma.

1.4. feladat (Rossz kódok). A következő bináris kódok melyike nem lehet semmilyen eloszlás Huffmann-kódja? Mindegyik válaszát indokolja meg, azaz ha nincs ilyen eloszlás, akkor magyarázza meg miért, ha pedig van, akkor adjon meg egy olyat!

a) $0, 10, 111, 101$

b) $00, 010, 011, 10, 110$

c) $1, 000, 001, 010, 011$

1.5. feladat. Legyen az $\mathcal{X} = \{x_1, \ldots, x_n\}$ forrásábécén adott valószínűség-eloszlás olyan, hogy minden egyes elem valószínűsége 2^{-i} alakú, ahol i egy pozitív egész szám. Bizonyítsa be, hogy ilyen esetekben a bináris Shannon–Fano-kód optimális! Mutassa meg, hogy a bináris Huffman-kód átlagos kódszóhossza akkor és csak akkor egyezik meg az entrópiával, ha az eloszlás ilyen alakú!

1.6. feladat. Legyen az $\mathcal{X}$ forrásábécé ötelemű, a következő valószínűségekkel: $0.4; 0.35; 0.1; 0.1; 0.05$. Mennyi az eloszlás entrópiája? Konstruálja meg a bináris Shannon–Fano-kódot erre az eloszlásra, illetve konstruáljon bináris prefix kódot az $l_i = \lceil -\log p(x_i) \rceil$ kódszóhosszakkal az 1.2. lemma bizonyítása szerint (a kód bináris fával való reprezentálásával). Mennyi az átlagos kódszóhossz?

1.7. feladat. Egy pénzérmét addig dobunk fel, amíg írást nem kapunk. Jelölje az X valószínűségi változó a dobások számát. Mennyi az X entrópiája?

1.8. feladat (Egyenletesebb eloszlás entrópiája nagyobb). Mutassa meg, hogy a

$$(p_1, \ldots, p_i, \ldots, p_j \ldots, p_n)$$

eloszlás entrópiája nem lehet nagyobb, mint a

$$(p_1, \ldots, \frac{p_i + p_j}{2}, \ldots, \frac{p_i + p_j}{2} \ldots, p_n)$$

eloszlás entrópiája!

1.9. feladat (Egyenletesebb eloszlás optimális kódja rosszabb).
Tekintsük a

$$\mathbf{p} = (p_1, \ldots, p_i, \ldots, p_j \ldots, p_n)$$

illetve

$$\mathbf{q} = (p_1, \ldots, \frac{p_i + p_j}{2}, \ldots, \frac{p_i + p_j}{2} \ldots, p_n)$$

eloszlásokat. Mutassa meg, hogy a **q** eloszláshoz tartozó optimális (minimális átlagos kódszóhosszúságú) kód átlagos kódszóhossza (**q** szerint) nem lehet kisebb, mint a **p** eloszláshoz tartozó optimális kód átlagos kódszóhossza (**p** szerint)!

1.10. feladat (Információs divergencia). Legyen $\mathbf{p} = (p_1, \ldots, p_n)$ és $\mathbf{q} = (q_1, \ldots, q_n)$ két valószínűség eloszlás, és definiáljuk a két eloszlás közötti „információs távolságot” a

$$D(\mathbf{p} \mid \mathbf{q}) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}$$

kifejezéssel. (A mennyiséget gyakran információs divergenciának, relatív entrópiának, vagy Kullback–Leibler-távolságnak nevezik.) Lássuk be a következő tulajdonságokat:

- Bármely két eloszlásra $D(\mathbf{p} \mid \mathbf{q}) \geq 0$, és egyenlőség pontosan akkor teljesül, ha $\mathbf{p} = \mathbf{q}$.
- $H(\mathbf{p}) = \log n - D(\mathbf{p} \mid \mathbf{u})$, ahol $H(\mathbf{p})$ jelöli a **p** eloszlás entrópiáját, **u** pedig az egyenletes eloszlást az $\{1, \ldots, n\}$ halmazon.

1.11. feladat (Rosszul ismert eloszlás). Tegyük fel, hogy az X valószínűségi változó eloszlása $\mathbf{p} = (p_1, \ldots, p_n)$, de ez az eloszlás nem pontosan ismert, helyette

a $\mathbf{q} = (q_1, \ldots, q_n)$ eloszlás adott, és ennek ismeretében készítünk Shannon–Fano-kódot, melynek kódszóhosszúságai tehát $l_i = \left\lceil \log \frac{1}{q_i} \right\rceil$, $i = 1, \ldots, n$. Mutassa meg, hogy a kapott kód átlagos kódszóhosszára

$$H(\mathbf{p}) + D(\mathbf{p} \mid \mathbf{q}) \leq \sum_{i=1}^{n} p_i l_i < H(\mathbf{p}) + D(\mathbf{p} \mid \mathbf{q}) + 1$$

teljesül! Ez azt jelenti, hogy az ár, amelyet az eloszlás pontatlan ismeretéért fizetünk, körülbelül az információs divergenciával egyezik meg (ami sohasem lehet negatív, tehát semmiképpen sem nyerhetünk!).

1.12. feladat (Shannon–Fano- és Huffman-kód). Legyen az X valószínűségi változó eloszlása $\left(\frac{1}{3}; \frac{1}{3}; \frac{1}{4}; \frac{1}{12}\right)$. Konstruáljon Huffmann-kódot ehhez az eloszláshoz. Mutassa meg, hogy két különböző optimális kód is van, azaz, hogy az $(1;2;3;3)$, és a $(2;2;2;2)$ kódszóhosszúságokkal adott mindkét kód optimális. Vonja le a következtetést, hogy létezik olyan optimális kód, amelynek van a megfelelő Shannon–Fano-kódénál hosszabb kódszava is.

1.13. feladat (Huffmann-kód kódszóhosszai). Tegyük fel, hogy a $(p_1, \ldots, p_n)$ eloszláshoz készítünk optimális bináris prefix kódot, ahol $p_1 > p_2 > \cdots > p_n > 0$. Bizonyítsa be, hogy

- Ha $p_1 > \frac{2}{5}$, akkor a hozzá tartozó kódszó egy hosszúságú;
- Ha $p_1 < \frac{1}{3}$ akkor a hozzá tartozó kódszó legalább kettő hosszúságú.

1.14. feladat (Titkosítás). Legyenek X és Z bináris (0-1 értékű) független valószínűségi változók, úgy, hogy $\mathbf{P}\{X = 1\} = p$ és $\mathbf{P}\{Z = 1\} = \frac{1}{2}$. Legyen $Y = X \oplus Z$, ahol $\oplus$ modulo 2 összeadást jelöl. X felfogható, mint titkosítandó üzenet, Z a titkos kulcs, és Y a rejtjelezett üzenet. Számolja ki a következő mennyiségeket, és magyarázza meg jelentésüket a titkosítás szempontjából: $H(X)$, $H(X \mid Z)$, $H(X \mid Y)$, $H(X \mid Y, Z)$.

1.15. feladat (Egyenlőtlenségek). Legyenek X, Y és Z tetszőleges (véges értékkészletű) valószínűségi változók. Bizonyítsa be a következő egyenlőtlenségeket:

- $H(X, Y \mid Z) \geq H(X \mid Z)$,
- $H(X, Y, Z) - H(X, Y) \leq H(X, Z) - H(X)$.

1.16. feladat. Legyen $\mathbb{X} = X_1, X_2, \ldots$ egy bináris, emlékezetnélküli stacionárius forrás, amelyre $\mathbf{P}\{X_1 = 1\} = 10^{-6}$. Adjuk meg az $\mathbb{X}$ egy olyan változó szóhosszúságú blokk-kódját, melynek betűnkénti átlagos kódszóhossza kisebb, mint $\frac{1}{10}$.

1.17. feladat (Futamhossz kódolás). Legyenek $X_1,\dots,X_n$ bináris valószínűségi változók. Jelölje $\mathbf{R}=(R_1,R_2,\dots)$ az egyes szimbólumok előfordulásainak futamhosszait. Tehát például az 1110010001111 sorozathoz $\mathbf{R}=(3,2,1,3,4)$ tartozik. Hogyan viszonyul egymáshoz $H(X_1,\dots,X_n), H(\mathbf{R})$ és $H(\mathbf{R},X_n)$?

1.18. feladat (Markov-lánc entrópiája). Legyen $\mathbb{X}=X_1,X_2,\dots$ egy bináris, stacionárius Markov-lánc, amelynek állapotátmenetei a következők:

$$\begin{aligned}
\mathbf{P}\{X_2=0\mid X_1=0\} &= p,\\
\mathbf{P}\{X_2=1\mid X_1=0\} &= 1-p,\\
\mathbf{P}\{X_2=0\mid X_1=1\} &= \frac{1-p}{2},\\
\mathbf{P}\{X_2=1\mid X_1=1\} &= \frac{1+p}{2}.
\end{aligned}$$

Mennyi $\mathbf{P}\{X_1=0\}$? Mennyi a forrás entrópiája?

1.19. feladat (Bináris entrópiafüggvény tulajdonságai). Legyen a $[0,1]$ intervallumon értelmezett h függvény (bináris entrópiafüggvény) értéke

$$h(x)=-x\log x-(1-x)\log(1-x),$$

ha $x\in(0,1)$, és $h(0)=h(1)=0$. Mutassuk meg, hogy h rendelkezik a következő tulajdonságokkal:

- szimmetrikus az $\frac{1}{2}$ pontra;
- $[0,1]$ minden pontjában folytonos;
- $\left[0,\frac{1}{2}\right]$-ben szigorúan monoton növekvő;
- szigorúan konkáv.

1.20. feladat (Vissza a jövőbe). Legyen $\dots,X_{-2},X_{-1},X_0,X_1,X_2,\dots$ valószínűségi változók egy stacionárius sorozata. Mutassa meg, hogy

$$H(X_0\mid X_{-1},X_{-2},\dots,X_{-n})=H(X_0\mid X_1,X_2,\dots,X_n),$$

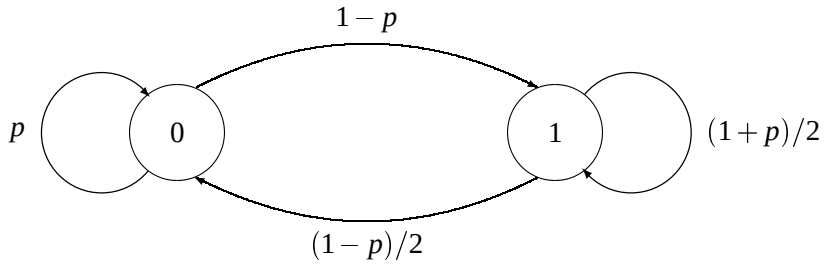
azaz, a jelen feltételes entrópiája a múlttal és jövővel mint feltétellel megegyezik.

1.21. feladat. Legyenek a $\mathbb{Z}$ stacionárius Markov-lánc állapotai a $0,1,\dots,255$ számok, és tegyük fel, hogy az állapotátmenet valószínűségek az alábbi 256×256-os mátrixszal adottak (a mátrix i-edik sorának j-edik oszlopában a $\mathbf{P}\{Z_2=j\mid$

$Z_1 = i\}$ valószínűség található). Mi a Markov-lánc stacionárius eloszlása? Mennyi a forrás entrópiája? Készítsen jó, egyértelműen dekódolható, változó szóhosszúságú blokk-kódot!

$$\begin{bmatrix} 1/2 & 1/4 & 1/4 & 0 & 0 & 0 & \cdots & & & & 0 \\ 0 & 1/2 & 1/4 & 1/4 & 0 & 0 & \cdots & & & & 0 \\ 0 & 0 & 1/2 & 1/4 & 1/4 & 0 & \cdots & & & & 0 \\ \vdots & & & & & & & & & & \vdots \\ 0 & 0 & \cdots & & & & & 0 & 1/2 & 1/4 & 1/4 \\ 1/4 & 0 & 0 & \cdots & & & & & 0 & 1/2 & 1/4 \\ 1/4 & 1/4 & 0 & \cdots & & & & & 0 & 0 & 1/2 \end{bmatrix}$$

1.22. feladat. Az alábbi ábra egy $\mathbb{Z} = Z_1, Z_2, \ldots$ Markov-lánc működését írja le. Tegyük fel, hogy a láncot a stacionárius eloszlásból indítjuk.



Legyen továbbá $Y_1, Y_2, \ldots$ független, azonos eloszlású bináris valószínűségi változók sorozata, ahol $\mathbf{P}(Y_i = 0) = \frac{1}{3}$. Definiáljuk az $\mathbb{X} = X_1, X_2, \ldots$ forrást az $X_i = 2Z_i + Y_i$ egyenlettel. Mennyi az $\mathbb{X}$ forrás entrópiája, feltéve, hogy $Z_1, Z_2, \ldots$ független $Y_1, Y_2, \ldots$-től?

1.23. feladat (Lempel–Ziv). Adjuk meg a 36 darab nullából álló sztring LZ78-kódját!

2. fejezet

Forráskódolás hűségkritériummal

Az eddigi vizsgálataink során megköveteltük, hogy a kódolt üzenet egyértelműen visszaállítható legyen. Ezt a követelményt sok gyakorlati probléma esetén fel kell adnunk, illetve jobb, ha feladjuk. Ebben a fejezetben olyan forráskódolási eljárásokat vizsgálunk, ahol az üzenet tökéletes reprodukciója helyett csak azt várjuk el, hogy a dekódolt üzenet az eredetit valamilyen értelemben hűen — de nem feltétlenül pontosan — adja vissza. Ilyen típusú kódolásokkal a hírközlés gyakorlatában sokszor találkozhatunk. Az emberi beszéd digitális átvitele illetve tárolása esetében például a folytonos jelből mintavételezéssel és kvantálással olyan jelet kapunk, amely már véges értékkészletű. Mégis azt mondhatjuk, hogy ezzel semmit sem vesztettünk, hiszen például a digitális központon keresztülhaladó telefonkapcsolat ugyanolyan jó minőségű (vagy jobb), mintha az analóg/digitális – digitális/analóg átalakítást elhagynánk. A lényeg az, hogy a forrásnak csak számunkra lényeges jellemzőit tartjuk meg, és így — megelégedve a közelítő visszaállítással — úgy kódolhatjuk, hogy a kapott jel továbbítása illetve tárolása már kisebb költséggel megoldható. (Vagyis pl. bináris kódot használva, a forrás kevesebb biten reprezentálható.)

A következőkben tárgyalandó kódok közös jellemzője lesz, hogy ún. blokkból-blokkba kódok, azaz a forrásábécé betűinek állandó hosszú blokkjait állandó hosszú kódszavakkal kódoljuk. Feltesszük, hogy adott az üzenetek és a kódjaik között egy ún. hűségmérték, ami azt méri, hogy egy adott kódszót milyen mértékben tekinthetünk egy adott üzenet reprodukciójának. Vizsgálataink középpontjában az a kérdés áll, hogy kódolással milyen mértékben tömöríthetjük a forrás által kibocsátott jelet, ha azt akarjuk, hogy a kód a forrást adott átlagos hűséggel reprezentálja.

Sajnos torzítást megengedő forráskódolás esetén a tömöríthetőség elvi hatá-

rainak a jellemzése általában nem konstruktív, ugyanakkor a gyakorlati feladatokban mégis kell tömöríteni, tehát megemlítünk néhány gyakorlati eljárást is: kvantálást, mintavételezést, prediktív kódolást, beszéd-, hang-, kép- és videotömörítést. A fejezetet az elvi határok tisztázásával zárjuk.

2.1. Forráskódolás előírt hibavalószínűséggel

A változó szóhosszú forráskódolás egy hátránya azonnal szembetűnik: egy kódszó meghibásodása esetén előfordulhat, hogy az utána levő összes kódszó dekódolását elrontjuk, mivel a kezdetüket rosszul detektáljuk. Ha állandó hosszúságú kódszavakat használunk, akkor ez nem fordulhat elő; bármely kódszó meghibásodása csak az illető kódszó dekódolásánál okoz gondot, hiszen a kódszó kezdetek kijelölése még dekódolás előtt megtörténhet. Azonban — mint rögtön látni fogjuk —, ez a megoldás elveszi a tömörítés lehetőségét. Legyen ugyanis $\mathcal{X}$ az n-elemű forrásábécé, és $\mathcal{Y}$ az s-elemű kódábécé. Ekkor, ha az f kód k-hosszú üzeneteket m-hosszú kódszavakba képez le (tehát $f : \mathcal{X}^k \to \mathcal{Y}^m$), akkor az egyértelmű dekódolhatóság feltétele az

$$n^k \leq s^m \tag{2.1}$$

egyenlőtlenség teljesülése, vagyis ezt a betűnkénti átlagos kódszóhosszal kifejezve

$$L = \frac{m}{k} \geq \frac{\log n}{\log s}, \tag{2.2}$$

mivel a különböző k-hosszú üzeneteknek különböző m-hosszú kódszavakat kell megfeleltetni. A (2.2) feltétel természetesen elégséges is egyértelműen dekódolható $f : \mathcal{X}^k \to \mathcal{Y}^m$ kód létezésére. Mivel a forrás entrópiája jóval kisebb lehet $\log n$-nél (de legfeljebb ennyi) ezért (2.2) megmutatja, hogy az állandó szóhosszú egyértelműen dekódolható kódok esetében az átlagos kódszóhossz általában nem közelítheti tetszőlegesen a forrásentrópiát bármilyen nagy legyen is a k blokkhossz. Valójában csak akkor nem vesztünk semmit, ha a kódolt forrás emlékezetnélküli és stacionárius egyenletes eloszlással.

Feladjuk tehát az egyértelmű dekódolhatóságot, és helyette azt követeljük meg, hogy a k hosszú üzeneteket nagy valószínűséggel tudjuk dekódolni. Mielőtt továbbmennénk bevezetünk egy jelölést:

Ha adott az $\mathbb{X} = X_1, X_2, \ldots$ stacionárius forrás, akkor a k hosszú üzenetek $B \subset \mathcal{X}^k$ halmazának valószínűségén a következőt értjük:

$$\mathbf{P}(B) = \mathbf{P}\{(X_1, \ldots, X_k) \in B\}.$$

A stacionaritás miatt persze bármely k-hosszú blokkot vehetnénk:

$$\mathbf{P}(B) = \mathbf{P}\{(X_n, \ldots, X_{n+k}) \in B\}, \qquad n = 1, 2, \ldots$$

Ha $\mathbf{x} = (x_1, \ldots, x_k) \in \mathcal{X}^k$, és a szokásos

$$p(\mathbf{x}) = \mathbf{P}\{X_1 = x_1, \ldots, X_k = x_k\}$$

jelölést használjuk, akkor

$$\mathbf{P}(B) = \sum_{\mathbf{x} \in B} p(\mathbf{x}).$$

2.1. definíció. *Az $\mathbb{X}$ stacionárius forrás $f : \mathcal{X}^k \to \mathcal{Y}^m$ kódját akkor mondjuk ε-hibával ($0 < \varepsilon < 1$) dekódolhatónak, ha létezik olyan $f' : \mathcal{Y}^m \to \mathcal{X}^k$ dekódoló függvény, hogy a hibás dekódolás valószínűsége kisebb ε-nál, vagyis*

$$\mathbf{P}\{f'(f(X_1, \ldots, X_k)) \neq (X_1, \ldots, X_k)\} \leq \varepsilon.$$

MEGJEGYZÉS:

a) A dekódolás hibáját a következő módon írhatjuk fel:

$$\mathbf{P}\{f'(f(X_1, \ldots, X_k)) \neq (X_1, \ldots, X_k)\} = \sum_{\mathbf{x}:\, f'(f(\mathbf{x})) \neq \mathbf{x}} p(\mathbf{x}).$$

b) Könnyen belátható, hogy az $f : \mathcal{X}^k \to \mathcal{Y}^m$ pontosan akkor dekódolható ε-hibával, ha f az $\mathcal{X}^k$ egy $1-\varepsilon$-nál nagyobb valószínűségű B részhalmazát invertálhatóan képezi le. Ebből következik, hogy akkor és csak akkor létezik $f : \mathcal{X}^k \to \mathcal{Y}^m$ ε-hibával dekódolható kód, ha létezik olyan $B \subset \mathcal{X}^k$, amelyre $\mathbf{P}(B) > 1-\varepsilon$ és $|B| \leq s^m$.

Az előbbi megjegyzés szerint tehát akkor kapunk a forrás k hosszú blokkjait állandó szóhosszúságú, ε-hibával dekódolható minimális kódszóhosszú kódot, ha keresünk egy minimális számosságú $B \subset \mathcal{X}^k$ üzenethalmazt, amelyre $\mathbf{P}(B) > 1-\varepsilon$. Legyen m olyan, hogy

$$s^{m-1} < |B| \leq s^m, \tag{2.3}$$

és így a B-beli üzeneteket kölcsönösen egyértelműen kódolhatjuk az m hosszú kódszavakkal. A többi üzenetet bárhogy (például mindegyiket egyazon $\mathbf{x} \in B$ kódjával) kódolva, egy $f : \mathcal{X}^k \to \mathcal{Y}^m$ ε-hibával dekódolható kódot kapunk ami persze optimális lesz, vagyis ha $g : \mathcal{X}^k \to \mathcal{Y}^{m'}$ is ε-hibával dekódolható, akkor $m \leq m'$.

Ilyen minimális elemszámú halmazt könnyen találhatunk. Indexeljük a k hosszú üzeneteket (összesen $l = n^k$ darabot) valószínűségeik szerint csökkenő sorrendben:

$$p(\mathbf{x}_1) \geq p(\mathbf{x}_2) \geq \cdots \geq p(\mathbf{x}_l),$$

és legyen $N(k,\varepsilon)$ az az index, amelyre

$$\sum_{i=1}^{N(k,\varepsilon)} p(\mathbf{x}_i) > 1-\varepsilon, \qquad \text{de} \qquad \sum_{i=1}^{N(k,\varepsilon)-1} p(\mathbf{x}_i) \le 1-\varepsilon.$$

Ekkor nyilván az első $N(k,\varepsilon)$ üzenet, vagyis a

$$B_{k,\varepsilon} = \bigcup_{i=1}^{N(k,\varepsilon)} \{\mathbf{x}_i\} \tag{2.4}$$

halmaz a kívánt minimális elemszámú, mivel semmilyen $N(k,\varepsilon)$-nál kisebb elemszámú halmaz nem lehet $1-\varepsilon$-nál nagyobb valószínűségű.

Ha tehát az $f : \mathcal{X}^k \to \mathcal{Y}^m$ ε-hibával dekódolható kód, akkor

$$N(k,\varepsilon) \le s^m,$$

tehát

$$L = \frac{m}{k} \ge \frac{\frac{1}{k}\log N(k,\varepsilon)}{\log s}. \tag{2.5}$$

Az L betűnkénti kódszóhossz viselkedésére (a k blokkhossz növelésével) tehát az $\frac{1}{k}\log N(k,\varepsilon)$ viselkedésének vizsgálatával következtethetünk. A rendkívül fontos és meglepő (mivel ε-tól függetlenül igaz)

$$\lim_{k\to\infty} \frac{1}{k}\log N(k,\varepsilon) = H(\mathbb{X})$$

összefüggést fogjuk bebizonyítani a stacionárius források egy jelentős osztályára, az információstabilis forrásokra, majd ennek felhasználásával kimondjuk és bizonyítjuk az előírt hibavalószínűségű forráskódolás tételét, amely formailag nagyon hasonlít a változó szóhosszúságú forráskódolás 1.9. tételéhez.

2.2. definíció. *Az $\mathbb{X} = X_1, X_2, \ldots$ stacionárius forrást **információstabilis**nak nevezzük, ha minden $\delta > 0$-ra*

$$\lim_{k\to\infty} \mathbf{P}\left\{\left|-\frac{1}{k}\log p(X_1,\ldots,X_k) - H(\mathbb{X})\right| > \delta\right\} = 0,$$

vagyis az $Y_k = -\frac{1}{k}\log p(X_1,\ldots,X_k)$, $k = 1,2,\ldots$ valószínűségi változók sorozata valószínűségben (sztochasztikusan) tart $H(\mathbb{X})$-hez, ha $k \to \infty$.

MEGJEGYZÉS:

a) Az $\mathbb{X}$ információstabilitása nagyjából azt jelenti, hogy elég nagy k-ra, a k hosszú sorozatok 1-hez közeli valószínűségű részére, mondjuk az $A \subset \mathcal{X}^k$-ra igaz, hogy

$$-\frac{1}{k}\log p(\mathbf{x}) \approx H(\mathbb{X})$$

illetve

$$p(\mathbf{x}) \approx 2^{-kH(\mathbb{X})},$$

ha $\mathbf{x} \in A$, valamint

$$|A| \approx 2^{kH(\mathbb{X})}.$$

Ezt a kijelentést később pontosítjuk majd, és látni fogjuk, hogy a kódolási tétel lényegében ezen az észrevételen alapul.

b) Nem túl egyszerűen, de be lehet bizonyítani, hogy a stacionárius és ergodikus források információstabilisak. Azt viszont most megmutatjuk, hogy ha $\mathbb{X} = X_1, X_2, \ldots$ emlékezetnélküli és stacionárius, akkor $\mathbb{X}$ információstabilis:

Mivel $X_1, X_2, \ldots, X_k$ függetlenek és azonos eloszlásúak,

$$\begin{aligned} Y_k &= -\frac{1}{k}\log p(X_1, X_2, \ldots, X_k) = \\ &= -\frac{1}{k}\log\left(p(X_1)p(X_2)\cdots p(X_k)\right), \end{aligned}$$

vagyis

$$Y_k = \frac{1}{k}\sum_{i=1}^{k}\left(-\log p(X_i)\right),$$

ahol a $-\log p(X_i)$-k független, azonos eloszlású valószínűségi változók. A nagy számok gyenge törvénye szerint ekkor az $Y_1, \ldots, Y_k, \ldots$ sorozat valószínűségben konvergál az l_{X_i}-k közös várható értékéhez, $H(X_1) = H(\mathbb{X})$-hez, tehát $\mathbb{X}$ információstabilis.

A következő tételen alapul majd az előírt hibavalószínűségű kódolás tétele:

2.1. tétel. *Ha az $\mathbb{X}$ stacionárius forrás információstabilis, akkor*

$$\lim_{k\to\infty}\frac{1}{k}\log N(k,\varepsilon) = H(\mathbb{X})$$

minden $0 < \varepsilon < 1$ esetén.

BIZONYÍTÁS: Legyen a k pozitív egészre és $\delta > 0$ tetszőleges valós számra $A_{k,\delta}$ a következő halmaz:

$$A_{k,\delta} = \left\{ \mathbf{x} \in \mathcal{X}^k : \left| -\frac{1}{k} \log p(\mathbf{x}) - H(\mathbb{X}) \right| \leq \delta \right\},$$

azaz $\mathbf{x} \in A_{k,\delta}$ pontosan akkor, ha

$$2^{-k(H(\mathbb{X})+\delta)} \leq p(\mathbf{x}) \leq 2^{-k(H(\mathbb{X})-\delta)}.$$

Az $A_{k,\delta}$ elemeit szokás tipikus sorozatoknak hívni, mivel egyrészt a valószínűségük közelítőleg $2^{-kH(\mathbb{X})}$, másrészt az $A^c_{k,\delta}$ összvalószínűsége „kicsi”.

Az $A_{k,\delta}$ elemeinek számát a következőképpen becsülhetjük felülről:

$$1 \geq \mathbf{P}(A_{k,\delta}) = \sum_{\mathbf{x} \in A_{k,\delta}} p(\mathbf{x}) \geq \left|A_{k,\delta}\right| \cdot \min_{\mathbf{x} \in A_{k,\delta}} p(\mathbf{x}) \geq \left|A_{k,\delta}\right| \cdot 2^{-k(H(\mathbb{X})+\delta)},$$

vagyis

$$\left|A_{k,\delta}\right| \leq 2^{k(H(\mathbb{X})+\delta)}. \tag{2.6}$$

Mivel az $\mathbb{X}$ információstabilitása pontosan azt jelenti, hogy $\mathbf{P}(A_{k,\delta}) \to 1$, ha $k \to \infty$, minden pozitív δ esetén, ezért k elég nagy értékeire $\mathbf{P}(A_{k,\delta}) > 1 - \varepsilon$ teljesül. Tudjuk viszont, hogy $B_{k,\varepsilon}$ olyan halmaz, hogy nála kisebb elemszámú halmaz valószínűsége nem lehet nagyobb $1 - \varepsilon$-nál, tehát (2.6)-ból

$$N(k,\varepsilon) = |B_{k,\varepsilon}| \leq \left|A_{k,\delta}\right| \leq 2^{k(H(\mathbb{X})+\delta)}$$

következik, ha k elég nagy. Ebből azt kapjuk, hogy

$$\frac{1}{k} \log N(k,\varepsilon) \leq H(\mathbb{X}) + \delta$$

minden elég nagy k-ra, és mivel δ tetszőleges volt, ez azt jelenti, hogy

$$\limsup_{k \to \infty} \frac{1}{k} \log N(k,\varepsilon) \leq H(\mathbb{X}). \tag{2.7}$$

Válasszuk most k-t olyan nagynak, hogy $\mathbf{P}\left(A_{k,\delta}\right) > \frac{1+\varepsilon}{2}$ teljesüljön. (Ezt megtehetjük, hiszen $\frac{1+\varepsilon}{2} < 1$.) Ekkor, mivel $\mathbf{P}\left(B^c_{k,\varepsilon}\right) < \varepsilon$ (B^c a B halmaz komplementere, vagyis $B^c = \mathcal{X}^k \setminus B$), ezért

$$\begin{aligned} \frac{1+\varepsilon}{2} &< \mathbf{P}\left(A_{k,\delta}\right) = \\ &= \mathbf{P}\left(A_{k,\delta} \cap B_{k,\varepsilon}\right) + \mathbf{P}\left(A_{k,\delta} \cap B^c_{k,\varepsilon}\right) < \\ &< \mathbf{P}\left(A_{k,\delta} \cap B_{k,\varepsilon}\right) + \varepsilon, \end{aligned}$$

és innen

$$\mathbf{P}\left(A_{k,\delta}\cap B_{k,\varepsilon}\right) > \frac{1-\varepsilon}{2}.$$

Írhatjuk tehát, hogy

$$\begin{aligned}\frac{1-\varepsilon}{2} &< \mathbf{P}\left(A_{k,\delta}\cap B_{k,\varepsilon}\right) \leq \\ &\leq \left|A_{k,\delta}\cap B_{k,\varepsilon}\right| \cdot \max_{\mathbf{x}\in A_{k,\delta}} p(\mathbf{x}) \leq \\ &\leq |B_{k,\varepsilon}| \cdot 2^{-k(H(\mathbb{X})-\delta)},\end{aligned}$$

vagyis

$$N(k,\varepsilon) > \left(\frac{1-\varepsilon}{2}\right)\cdot 2^{k(H(\mathbb{X})-\delta)}.$$

Ezt átrendezve azt kapjuk, hogy

$$\frac{1}{k}\log N(k,\varepsilon) > H(\mathbb{X}) - \delta + \frac{1}{k}\log\left(\frac{1-\varepsilon}{2}\right),$$

ha k elég nagy. Mivel $\delta > 0$ tetszőleges volt, ebből következik, hogy

$$\liminf_{k\to\infty}\frac{1}{k}\log N(k,\varepsilon) \geq H(\mathbb{X}),$$

amiből (2.7)-tel együtt következik a tétel állítása. ■

Most már mindent tudunk az ε-hibavalószínűségű kódolás tételéhez.

2.2. tétel. *Legyen az $\mathbb{X}$ stacionárius forrás információstabilis. Ekkor, ha az $\mathbb{X}$ forrás k-hosszú blokkjait ε-hibával ($0<\varepsilon<1$) kódoljuk állandó m_k hosszú kódszavakkal, akkor kódok bármely ilyen tulajdonságú sorozatára*

$$\liminf_{k\to\infty}\frac{m_k}{k} \geq \frac{H(\mathbb{X})}{\log s}.$$

Másrészt, tetszőleges $0<\varepsilon<1$ hibavalószínűséghez és pozitív δ-hoz elég nagy k esetén mindig létezik olyan $f:\mathcal{X}^k\to\mathcal{Y}^{m_k}$ ε-hibával dekódolható kód, hogy

$$L = \frac{m_k}{k} < \frac{H(\mathbb{X})}{\log s} + \delta.$$

BIZONYÍTÁS: Ha $f : \mathcal{X}^k \to \mathcal{Y}^{m_k}$ ε-hibával dekódolható kód, akkor (2.5) szerint

$$\frac{m_k}{k} \geq \frac{\frac{1}{k}\log N(k,\varepsilon)}{\log s},$$

és itt a jobb oldal az előző tétel szerint $\frac{H(\mathbb{X})}{\log s}$-hez tart $k \to \infty$ esetén, tehát az első állítást beláttuk.

A második állítás bizonyításához legyen

$$m_k = \left\lceil \frac{\log N(k,\varepsilon)}{\log s} \right\rceil, \tag{2.8}$$

ahol az $\lceil x \rceil$ jelölés az x valós szám felső egész részét jelenti. Ekkor a 2.1. tétel miatt létezik olyan k_0, hogy

$$k_0 > \frac{2}{\delta}, \qquad \text{és} \qquad \frac{1}{k}\log N(k,\varepsilon) < H(\mathbb{X}) + \frac{\delta}{2}\log s, \quad \text{ha} \quad k > k_0. \tag{2.9}$$

Ekkor (2.8) és (2.9) szerint

$$\frac{m_k}{k} \leq \frac{1}{k}\left(\frac{\log N(k,\varepsilon)}{\log s} + 1\right) < \frac{H(\mathbb{X})}{\log s} + \frac{\delta}{2} + \frac{1}{k} < \frac{H(\mathbb{X})}{\log s} + \delta$$

ha $k > k_0$, amivel a második állítást is bebizonyítottuk. ■

A 2.2. tétel tulajdonképpen azt mondja ki, hogy nem lehetséges tetszőlegesen nagy blokkokat kódoló ε-hibával dekódolható olyan kódot konstruálni, melynek betűnkénti átlagos kódszóhossza kisebb $\frac{H(\mathbb{X})}{\log s}$-nél, de ezt az értéket felülről tetszőlegesen megközelíthetjük, ha elég nagy blokkhosszt (k) használunk.

Vizsgáljuk most a problémát más szempontból. Kérdezhetjük azt, hogy ha a betűnkénti átlagos kódszóhosszat állandó értéken tartjuk, mi történik a dekódolás hibájával a k blokkhossz növelésével. Vezessük be az

$$R = \frac{m}{k}\log s$$

jelsebességet, ami megadja, hogy ha az $N = s^m$ darab kódszót binárisan reprezentáljuk, akkor forrásbetűnként átlagosan hány bitet használunk fel. Mivel az m-hosszú kódszavakkal $N = s^m$ kódszót lehet egyértelműen dekódolhatóan kódolni, ezért az R jelsebességű, k-hosszú blokkokat kódoló kódok közül az a legjobb (legkisebb hibával dekódolható), amelyik az $\mathcal{X}^k$ első $N = 2^{kR}$ legnagyobb valószínűségű elemét kódolja egyértelműen (a többi üzenetet tetszőlegesen). Tehát ha az

$\mathcal{X}^k$ elemei valószínűségeik szerint csökkenő sorrendben vannak indexelve, akkor a legjobb ilyen kód hibavalószínűségére

$$P_e(k,R) = \sum_{i \geq 2^{kR}} p(\mathbf{x}_i) \tag{2.10}$$

adódik.

2.3. tétel. *Ha az $\mathbb{X}$ stacionárius forrás információstabilis, akkor a legfeljebb R jelsebességű, k hosszú blokkokat állandó szóhosszon kódoló, legkisebb hibával dekódolható kód hibavalószínűségére igaz a következő:*

$$\lim_{k\to\infty} P_e(k,R) = 0, \quad \text{ha } R > H(\mathbb{X}),$$

és

$$\lim_{k\to\infty} P_e(k,R) = 1, \quad \text{ha } R < H(\mathbb{X}),$$

vagyis $R > H(\mathbb{X})$ esetén elég hosszú blokkokat kódolva a dekódolás hibája tetszőlegesen kicsivé tehető, míg $R < H(\mathbb{X})$ esetén a blokkhosszt növelve a dekódolás hibája 1-hez tart, azaz a kód használhatatlanná válik.

BIZONYÍTÁS: Legyen $R_k = \frac{m_k}{k} \log s$ az $f : \mathcal{X}^k \to \mathcal{Y}^{m_k}$ minimális betűnkénti átlagos kódszóhosszú, ε-hibával dekódolható kód jelsebessége, és legyen $R > H(\mathbb{X})$. A 2.2. tétel második fele szerint

$$\limsup_{k\to\infty} R_k \leq H(\mathbb{X}).$$

Mivel $R > H(\mathbb{X})$, ezért létezik olyan (ε-tól függő) k_0 index, hogy $R_k < R$, ha $k > k_0$. Ebből (2.10) szerint

$$P_e(k,R_k) \geq P_e(k,R)$$

következik, vagyis

$$P_e(k,R) < \varepsilon,$$

ha $k > k_0$, és mivel ε tetszőlegesen kicsi lehet, a tétel első felét beláttuk.

Legyen most $R < H(\mathbb{X})$. Tegyük fel, hogy a tétel második állítása nem teljesül. Ekkor létezik egy olyan $k_i \to \infty$ sorozat, hogy $P_e(k_i,R) < \varepsilon$ $(i = 1,2,\ldots)$ valamilyen $\varepsilon < 1$ pozitív számra. Egy ilyen R jelsebességű, k_i-hosszú blokkokat kódoló kód átlagos kódszóhossza persze

$$L = \frac{m_{k_i}}{k_i} \leq \frac{R}{\log s},$$

tehát

$$\liminf_{i\to\infty} \frac{m_{k_i}}{k_i} \leq \frac{R}{\log s} < \frac{H(\mathbb{X})}{\log s},$$

ami ellentmond a 2.2. tétel első állításának, mivel ezek a kódok ε-hibával dekódolhatóak. ■

Ha $\mathbb{X}$ emlékezetnélküli és stacionárius, akkor a $P_e(k,R)$ hibavalószínűségre a 2.3. tételnél erősebb állítást is be lehet bizonyítani. Nevezetesen azt, hogy ha $R > H(\mathbb{X})$, akkor $P_e(k,R)$ exponenciálisan tart 0-hoz; és $R < H(\mathbb{X})$ esetén pedig $1 - P_e(k,R)$ tart exponenciálisan a 0-hoz, ha $k \to \infty$.

2.2. Kvantálás

A digitális módszereket a hírközlés szinte minden területén alkalmazzák. Minden esetben, amikor az adatok feldolgozása digitálisan történik, a folytonos értékkészletű jelet véges értékkészletűvé kell alakítani. A digitalizálás elvi határait az utolsó szakaszban tisztázzuk, de az egyértelmű, illetve az ε-hibavalószínűségű dekódolhatósággal ellentétben az elmélet nem konstruktív. Az alkalmazott, tehát konstruktív digitalizálás legegyszerűbb módja a skalár (egydimenziós) kvantálás.

Legyen $\mathbb{X} = X_1, X_2, \ldots$ egy stacionárius forrás, ahol az X_i-k valós valószínűségi változók. Az $\mathbb{X}$ egydimenziós kvantáltján egy véges értékkészletű $Q : \mathbb{R} \to \mathbb{R}$ leképezéssel kapott $Q(X_1), Q(X_2), \ldots$ diszkrét valószínűségi változósorozatot (forrást) értünk. A $Q(\cdot)$ függvényt **kvantáló**nak nevezzük. Vegyük észre, hogy a kvantálás az előző fejezet értelmében egy $k = 1$ hosszú blokkokat kódoló forráskód (azaz betűnkénti kód), melynek reprodukciós ábécéje a forrásábécé (a valós számok) egy véges részhalmaza. Természetesen az a célunk, hogy az $\mathbb{X}$-et hűen reprezentáljuk. A kódolás hűségét egy speciális hűségmértékkel, a $D(Q)$ négyzetes torzítással mérjük n hosszú blokkokra:

$$D(Q) = \frac{1}{n}\mathbf{E}\left(\sum_{i=1}^{n} (X_i - Q(X_i))^2\right),$$

ami, mivel X_i-k azonos eloszlásúak, egyenlő a következővel:

$$D(Q) = \mathbf{E}\left((X - Q(X))^2\right), \tag{2.11}$$

ahol X ugyanolyan eloszlású mint az X_i-k. (Természetesen más hűségmértéket is választhattunk volna, de a gyakorlatban ez a legelterjedtebb.)

Legyen a Q kvantáló értékkészlete az $\{x_1, x_2, \ldots, x_N\}$ halmaz, ahol az x_i-k valós számok. Az x_i számokat kvantálási szinteknek nevezzük. Vegyük észre,

hogy Q-t egyértelműen leírják az $\{x_1, x_2, \ldots, x_N\}$ kvantálási szintek és a $\mathcal{B}_i = \{x \in \mathbb{R} : \ Q(x) = x_i\},\ i = 1, \ldots, N$ kvantálási tartományok. A $\mathcal{B}_i$ halmazok persze diszjunktak és egyesítésük kiadja az egész valós egyenest. A kvantáló működése ezért a következőképp írható le:

$$Q(x) = x_i, \quad \text{ha } x \in \mathcal{B}_i.$$

Tegyük most fel, hogy a kvantálót leíró adatok közül most csak az $\{x_1, x_2, \ldots, x_N\}$ kvantálási szinteket ismerjük. Ekkor az ilyen kvantálási szinteket használó kvantálók közül a legkisebb torzítású az a Q kvantáló, amelyre

$$\mathcal{B}_i = \{x : \ |x - x_i| \leq |x - x_j|,\ j = 1, 2, \ldots, N\} \tag{2.12}$$

(legközelebbi szomszéd feltétel), ahol a döntési szabályt úgy tehetjük egyértelművé (vagyis a $\mathcal{B}_i$-ket diszjunktakká), hogy ha egy adott x kettő vagy több $\mathcal{B}_i$-be tartozna, akkor a legkisebb indexűhöz soroljuk. Az adott kvantálási szintekkel Q valóban legkisebb négyzetes torzítású, hiszen ha Q^* egy másik kvantáló ugyanezen kvantálási szintekkel, akkor egy tetszőleges x-re $Q(x) = x_i$ (tehát $x \in \mathcal{B}_i$) és $Q^*(x) = x_j$ valamely $1 \leq i,\ j \leq N$ indexekre, de ekkor (2.12) szerint

$$|x - x_i| \leq |x - x_j|,$$

tehát

$$(x - Q(x))^2 \leq (x - Q^*(x))^2$$

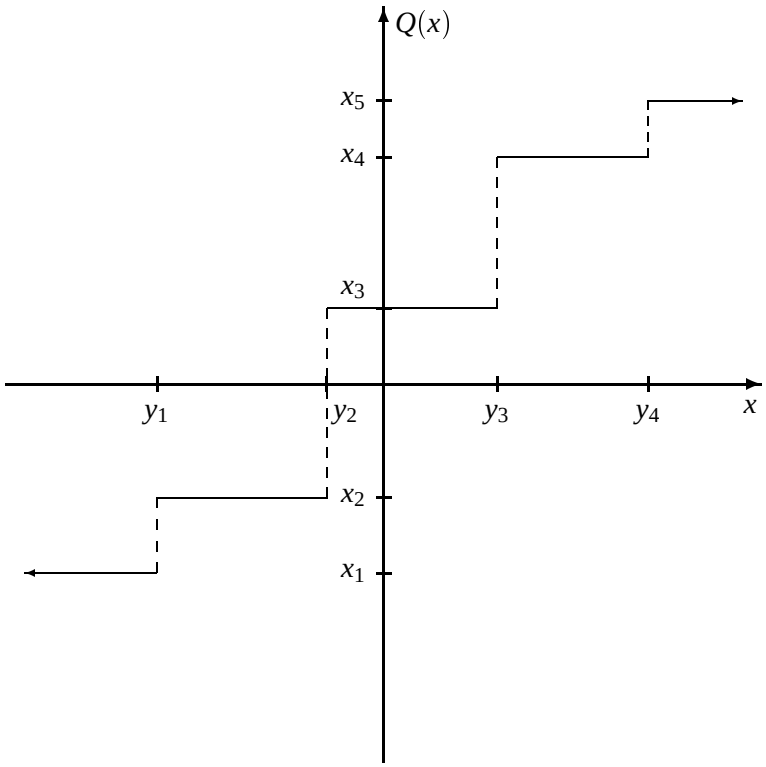
teljesül minden $x \in \mathbb{R}$-re amiből $D(Q) \leq D(Q^*)$ következik. Ezért a következőkben csak a (2.12) szerinti kvantálókkal foglalkozunk. Az ilyen kvantálók $\mathcal{B}_i$ kvantálási tartományai nagyon egyszerűen néznek ki. Az általánosság megszorítása nélkül tegyük fel most, hogy a kvantálási szintek nagyság szerint rendezve vannak, vagyis $x_1 < x_2 < \cdots < x_N$. Ekkor, bevezetve az $y_i = \frac{x_i + x_{i+1}}{2},\ i = 1, \ldots, N-1$ jelölést, a (2.12) szerinti $\mathcal{B}_i$ halmazok a következő intervallumok lesznek:

$$\mathcal{B}_1 = (-\infty, y_1] \qquad \mathcal{B}_i = (y_{i-1}, y_i],\ i = 2, \ldots, N-1, \qquad \mathcal{B}_N = (y_{N-1}, \infty).$$

Egy ilyen, $N = 5$ szintű kvantálót ábrázol a 2.1. ábra.

Az optimális kvantálási szint egy adott tartományhoz annak súlypontja. Ugyanis a Steiner-tétel miatt tetszőleges c konstansra

$$\begin{aligned}
&\mathbf{E}\left((X - c)^2 \mid X \in \mathcal{B}_i\right) = \\
&\quad = \mathbf{E}\left((X - \mathbf{E}(X \mid \mathcal{B}_i))^2 \mid X \in \mathcal{B}_i\right) + \left(\mathbf{E}(X \mid \mathcal{B}_i) - c\right)^2,
\end{aligned}$$



2.1. ábra. Kvantáló $N = 5$ kvantálási szinttel.

tehát az

$$\frac{1}{\int\limits_{\mathcal{B}_i} f(x)\,\mathrm{d}x}\int\limits_{\mathcal{B}_i}(x-x_i)^2 f(x)\,\mathrm{d}x = \mathbf{E}\left((X-x_i)^2 \mid X\in\mathcal{B}_i\right)$$

akkor minimális, ha

$$x_i = \frac{\int\limits_{\mathcal{B}_i} x f(x)\,\mathrm{d}x}{\int\limits_{\mathcal{B}_i} f(x)\,\mathrm{d}x} \qquad = \mathbf{E}(X \mid X\in\mathcal{B}_i),$$

ami a súlypont.

A technikai nehézségek elkerülése végett a további vizsgálataink során feltesszük, hogy a kvantált X valós valószínűségi változó eloszlása abszolút folytonos f sűrűségfüggvénnyel, valamint azt is feltesszük, hogy f a $[-A,A]$ intervallumban folytonos, a $[-A,A]$ intervallumon kívül nulla értékű függvény. Az f-et felhasz-

nálva a kvantáló négyzetes torzítása a következőképp írható fel:

$$D(Q) = \int_{-\infty}^{\infty} (x - Q(x))^2 f(x)\,\mathrm{d}x = \sum_{i=1}^{N} \int_{\mathcal{B}_i} (x - x_i)^2 f(x)\,\mathrm{d}x.$$

A legegyszerűbb kvantáló az **egyenletes kvantáló**. A Q_N N-szintű egyenletes kvantálót úgy kapjuk, hogy az X lehetséges értékeinek halmazát, a $[-A, A]$ intervallumot, N egyenlő nagyságú intervallumra osztjuk (ezek a $\mathcal{B}_i$ intervallumok), és a kvantálási szinteket ezen intervallumok közepén helyezzük el. Formálisan tehát

$$Q_N(x) = -A + (2i-1)\frac{A}{N},$$

ha

$$-A + 2(i-1)\frac{A}{N} < x \le -A + 2i\frac{A}{N}, \qquad i = 1, \ldots, N.$$

A következő tétel megmutatja, hogy az N szintű egyenletes kvantáló négyzetes torzítása nagy N esetén közelítőleg $\frac{1}{12}\left(\frac{2A}{N}\right)^2$.

2.4. tétel. *Ha az X valószínűségi változó f sűrűségfüggvényére a fenti feltételek teljesülnek, akkor az X-et egyenletesen N szinten kvantáló Q_N kvantáló torzítására a*

$$\lim_{N\to\infty} \left(\frac{N}{2A}\right)^2 D(Q_N) = \frac{1}{12}$$

aszimptotikus összefüggés teljesül.

Mivel az N-szintű egyenletes kvantálásnál egy $2A$ hosszú intervallumot osztottunk N egyenlő részre, egy kvantálási intervallum hossza $q_N = \frac{2A}{N}$. A tétel tehát azt állítja, hogy nagy N-ekre $D(Q_N) \approx \frac{q_N^2}{12}$.

BIZONYÍTÁS: Legyenek $y_{N,i} = -A + 2i\frac{A}{N}$, $i = 0, \ldots, N$, $x_{N,i} = -A + (2i-1)\frac{A}{N}$, $i = 1, \ldots, N$, az N-szintű egyenletes kvantáló intervallumainak határai illetve kvantálási szintjei. Így a torzítás

$$D(Q_N) = \sum_{i=1}^{N} \int_{y_{N,i-1}}^{y_{N,i}} (x - x_{N,i})^2 f(x)\,\mathrm{d}x.$$

Definiáljuk az $f_N(x)$ szakaszonként konstans sűrűségfüggvényeket a következőképp:

$$f_N(x) = \frac{1}{q_N} \int_{y_{N,i-1}}^{y_{N,i}} f(z)\,\mathrm{d}z, \qquad \text{ha } x \in (y_{N,i}, y_{N,i-1}].$$

Számítsuk ki az $f_N(x)$ szerinti torzítást, $\bar{D}(Q_N)$-et. Mivel az $f_N(x)$ konstans a $(y_{N,i}, y_{N,i-1}]$ intervallumokon, azt kapjuk, hogy

$$\begin{aligned}
\bar{D}(Q_N) &= \sum_{i=1}^{N} \int_{y_{N,i-1}}^{y_{N,i}} (x - x_{N,i})^2 f_N(x)\,\mathrm{d}x = \\
&= \sum_{i=1}^{N} \frac{1}{q_N} \int_{y_{N,i-1}}^{y_{N,i}} f(z)\,\mathrm{d}z \int_{y_{N,i-1}}^{y_{N,i}} (x - x_{N,i})^2\,\mathrm{d}x = \\
&= \sum_{i=1}^{N} \frac{1}{q_N} \int_{y_{N,i-1}}^{y_{N,i}} f(z)\,\mathrm{d}z \int_{-\frac{q_N}{2}}^{\frac{q_N}{2}} x^2\,\mathrm{d}x = \\
&= \frac{q_N^2}{12} \sum_{i=1}^{N} \int_{y_{N,i-1}}^{y_{N,i}} f(z)\,\mathrm{d}z = \\
&= \frac{q_N^2}{12}. \qquad (2.13)
\end{aligned}$$

A tételt tehát beláttuk, ha be tudjuk bizonyítani, hogy

$$\lim_{N\to\infty} \frac{D(Q_N) - \bar{D}(Q_N)}{D(Q_N)} = \lim_{N\to\infty} \frac{D(Q_N) - \bar{D}(Q_N)}{q_N^2/12} = 0. \qquad (2.14)$$

Ennek bizonyításánál használjuk ki az $f(x)$ folytonosságát. Mivel az $f(x)$ a $[-A, A]$ intervallumban folytonos, ezért itt egyenletesen folytonos, tehát mivel $q_N \to 0$, ha $N \to \infty$, adott $\varepsilon > 0$ esetén elég nagy N-re

$$|f(x) - f(y)| < \varepsilon, \quad \text{ha } x, y \in (y_{N,i-1}, y_{N,i}],\ i = 1, \ldots, N.$$

Ekkora N-ekre tehát $|f(x) - f_N(x)| < \varepsilon$ teljesül.

Ilyen nagy N-ekre írhatjuk tehát, hogy

$$\begin{aligned}
&\frac{12}{q_N^2} |D(Q_N) - \bar{D}(Q_N)| = \\
&\quad = \frac{12}{q_N^2} \left| \sum_{i=1}^{N} \int_{y_{N,i-1}}^{y_{N,i}} (x - x_{N,i})^2 f(x)\,\mathrm{d}x - \sum_{i=1}^{N} \int_{y_{N,i-1}}^{y_{N,i}} (x - x_{N,i})^2 f_N(x)\,\mathrm{d}x \right| \leq \\
&\quad \leq \frac{12}{q_N^2} \sum_{i=1}^{N} \int_{y_{N,i-1}}^{y_{N,i}} (x - x_{N,i})^2 |f(x) - f_N(x)|\,\mathrm{d}x \leq
\end{aligned}$$

$$\leq \frac{12}{q_N^2} N \frac{q_N^3}{12} \varepsilon =$$
$$= 2A\varepsilon, \tag{2.15}$$

ahol a második egyenlőtlenségnél kihasználtuk, hogy $|f(x) - f_N(x)| < \varepsilon$, és hogy $x_{N,i}$-k az $(y_{N,i}, y_{N,i-1}]$ intervallumok közepén vannak. Mivel ε tetszőleges volt, ezért (2.15) bizonyítja (2.14)-et és ezzel a tételt beláttuk. ■

A jelsebesség 2.11. szakasz szerinti definíciójával az N-szintű kvantáló — mint forráskód — jelsebessége $R = \log N$-nek adódik, vagyis egy X_i kvantált $Q(X_i)$ értékét $\log N$ bit felhasználásával továbbíthatjuk illetve tárolhatjuk. Vegyük észre azonban, hogy a $Q(X_1), Q(X_2), \ldots$ valószínűségi változók sorozata egy diszkrét stacionárius forrás. Így tehát a $Q(X_1), Q(X_2), \ldots$ forrásra, mondjuk bináris kódábécét használva, változó szóhoszúságú kódolást alkalmazhatunk, és az 1.9. tétel szerint egy ilyen kód átlagos kódszóhossza a $Q(X_1), Q(X_2), \ldots$ forrás entrópiáját felülről tetszőlegesen megközelítheti. A forrás entrópiája az 1.6. definíció utáni megjegyzés szerint $H(Q(X_1))$, amire tudjuk, hogy $H(Q(X_1)) \leq \log N$, tehát ha a $Q(X_i)$-k nem egyenletes eloszlásúak, akkor további tömörítést érhetünk el. Ebben az esetben tehát a kvantáló entrópiája is érdekel minket, nem csak a torzítása. A következő tétel egy aszimptotikus összefüggést ad az egyenletes kvantáló szintjeinek száma és entrópiája között.

2.5. tétel. *Tegyük fel, hogy az X valószínűségi változó f sűrűségfüggvénye csak a $[-A, A]$ intervallumon belül különbözik nullától, és f folytonos $[-A, A]$-ban. Tegyük fel továbbá, hogy a*

$$H(f) = -\int_{-A}^{A} f(x) \log f(x)\, \mathrm{d}x$$

integrál véges. Ekkor az X N-szintű egyenletes kvantálásának $H(Q_N(X))$ entrópiájára

$$\lim_{N\to\infty} \left(H(Q_N(X)) + \log \frac{2A}{N} \right) = H(f).$$

A tétel tehát azt állítja, hogy a kvantálási szintek N számának növekedésével az egyenletes kvantáló entrópiájára a $H(Q_N(X)) \approx H(f) - \log q_N$ közelítés érvényes.

BIZONYÍTÁS: A bizonyítás során a 2.4. tétel bizonyításában bevezetett jelöléseket fogjuk használni. Mivel f folytonos, ezért a differenciálszámítás Lagrange-féle középértéktételét alkalmazva az $F(x) = \int_{-\infty}^{x} f(z)\, \mathrm{d}z$ deriválható függvényre,

azt kapjuk, hogy léteznek $\xi_{N,i} \in (y_{N,i}, y_{N,i-1}]$ számok, amelyekre

$$\int_{y_{N,i-1}}^{y_{N,i}} f(x)\,\mathrm{d}x = (y_{N,i} - y_{N,i-1}) f(\xi_{N,i}) = q_N f(\xi_{N,i}), \quad i = 1, \ldots, N. \tag{2.16}$$

Ezek szerint tehát

$$\begin{aligned} H(Q_N(X)) &= -\sum_{i=1}^{N} \mathbf{P}\{Q_N(X) = x_i\} \log \mathbf{P}\{Q_N(X) = x_i\} = \\ &= \sum_{i=1}^{N} \left(\int_{y_{N,i-1}}^{y_{N,i}} f(x)\,\mathrm{d}x \right) \log \left(\int_{y_{N,i-1}}^{y_{N,i}} f(x)\,\mathrm{d}x \right) = \\ &= -\sum_{i=1}^{N} q_N f(\xi_{N,i}) \log \left(q_N f(\xi_{N,i}) \right) = \\ &= -\sum_{i=1}^{N} q_N f(\xi_{N,i}) \log q_N - \sum_{i=1}^{N} q_N f(\xi_{N,i}) \log f(\xi_{N,i}). \end{aligned} \tag{2.17}$$

Ekkor egyrészt (2.16) szerint

$$-\sum_{i=1}^{N} q_N f(\xi_{N,i}) \log q_N = (-\log q_N) \sum_{i=1}^{N} \int_{y_{N,i-1}}^{y_{N,i}} f(x)\,\mathrm{d}x = -\log q_N, \tag{2.18}$$

másrészt

$$\lim_{N\to\infty} -\sum_{i=1}^{N} q_N f(\xi_{N,i}) \log f(\xi_{N,i}) = -\int_{-A}^{A} f(x) \log f(x)\,\mathrm{d}x, \tag{2.19}$$

mivel a bal oldal a jobb oldali integrál Riemann közelítő összege. Összevetve (2.17)-et (2.18)-cal és (2.19)-cel, a tétel állítása adódik. ■

Ha a 2.4. és 2.5. tételeket összevetjük, akkor könnyen belátható a

$$\lim_{N\to\infty} \left(H(Q_N(X)) + \log \sqrt{12 D(Q_N)} \right) = H(f)$$

aszimptotikus összefüggés az egyenletes kvantálás torzítása és entrópiája között. Ezt úgy is fogalmazhatjuk, hogy

$$H(Q_N(X)) \approx H(f) - \log \sqrt{12 D(Q_N)}$$

nagy N-ek, vagyis finom (kis lépésközű) kvantálás esetén, ami egy hasznos közelítés lehet egyenletes kvantáló tervezésénél.

A $H(f)$ differenciális entrópia tehát a finom, egyenletes kvantáló kimenetének a tömöríthetőségét méri, azaz valamilyen értelemben az eloszlás terjedelmét. Megmutatjuk, hogy adott σ szórású f sűrűségfüggvények közül a normális eloszlás sűrűségfüggvényére a legnagyobb a $H(f)$ differenciális entrópia. Először megmutatjuk, hogy ha $f(x)$ és $g(x)$ két, olyan folytonos sűrűségfüggvény, amelyre az

$$\int f(z)\log\frac{f(z)}{g(z)}\,\mathrm{d}z$$

integrál létezik és véges, akkor ez az integrál nemnegatív, és 0 akkor és csak akkor, ha $f(x)\equiv g(x)$.

Legyen X egy valószínűségi változó $f(x)$ sűrűségfüggvénnyel, és legyen $Z=\frac{g(X)}{f(X)}$ egy másik valószínűségi változó. Alkalmazzuk erre a Jensen-egyenlőtlenséget a $-\log z$ konvex függvénnyel:

$$\begin{aligned}
\mathbf{E}(-\log(Z)) &\geq -\log(\mathbf{E}Z)\\
-\int\log\left(\frac{g(x)}{f(x)}\right)f(x)\,\mathrm{d}x &\geq -\log\left(\int\frac{g(x)}{f(x)}\,f(x)\,\mathrm{d}x\right)\\
-\int\log\left(\frac{g(x)}{f(x)}\right)f(x)\,\mathrm{d}x &\geq -\log\left(\int g(x)\,\mathrm{d}x\right)\\
\int f(x)\log\frac{f(x)}{g(x)}\,\mathrm{d}x &\geq 0
\end{aligned}$$

Az egyenlőség feltétele szintén a Jensen-egyenlőtlenségből valamint f és g folytonosságából következik.

Legyen most $f(x)$ egy olyan sűrűségfüggvény, amelynek várható értéke 0, szórása σ, és $\varphi(x)$ a 0 várható értékű és σ szórású normális eloszlás sűrűségfüggvénye, azaz

$$\varphi(x)=\frac{1}{\sqrt{2\pi}\sigma}\,\mathrm{e}^{-\frac{x^2}{2\sigma^2}}.$$

Bebizonyítjuk, hogy

$$-\int\varphi(x)\ln\varphi(x)\,\mathrm{d}x\geq-\int f(x)\ln f(x)\,\mathrm{d}x,$$

amiből következik, hogy

$$H(\varphi)\geq H(f).$$

Vegyük észre, hogy

$$\int\varphi(x)\,\mathrm{d}x=\int f(x)\,\mathrm{d}x=1$$

és

$$\int \varphi(x)x^2\,\mathrm{d}x = \int f(x)x^2\,\mathrm{d}x = \sigma^2$$

miatt

$$\begin{aligned}
&-\int \varphi(x)\ln\varphi(x)\,\mathrm{d}x + \int f(x)\ln f(x)\,\mathrm{d}x = \\
&\quad = -\int \varphi(x)\left(\ln\frac{1}{\sqrt{2\pi}\sigma} - \frac{x^2}{2\sigma^2}\right)\mathrm{d}x + \int f(x)\ln f(x)\,\mathrm{d}x = \\
&\quad = -\ln\frac{1}{\sqrt{2\pi}\sigma}\int \varphi(x)\,\mathrm{d}x + \frac{1}{2\sigma^2}\int \varphi(x)x^2\,\mathrm{d}x + \int f(x)\ln f(x)\,\mathrm{d}x = \\
&\quad = -\ln\frac{1}{\sqrt{2\pi}\sigma}\int f(x)\,\mathrm{d}x + \frac{1}{2\sigma^2}\int f(x)x^2\,\mathrm{d}x + \int f(x)\ln f(x)\,\mathrm{d}x = \\
&\quad = -\int f(x)\left(\ln\frac{1}{\sqrt{2\pi}\sigma} - \frac{x^2}{2\sigma^2}\right)\mathrm{d}x + \int f(x)\ln f(x)\,\mathrm{d}x = \\
&\quad = \int f(x)\ln\frac{f(x)}{\varphi(x)}\,\mathrm{d}x,
\end{aligned}$$

ami nemnegatív.

Ebből az is következik, hogy

$$\begin{aligned}
H(f) &\le H(\varphi) = \\
&= -\int \varphi(x)\left(\log\frac{1}{\sqrt{2\pi}\sigma} - \frac{x^2}{2\sigma^2}\log e\right)\mathrm{d}x = \\
&= \log\left(\sqrt{2\pi}\sigma\right) + \frac{1}{2}\log e = \\
&= \frac{1}{2}\log\left(2\pi e\sigma^2\right).
\end{aligned}$$

Hasonló technikával megmutatható, hogy az adott várható értékű és a nemnegatív félegyenesre koncentrált sűrűségfüggvények közül az adott várható értékű, exponenciális sűrűségfüggvénynek a legnagyobb a differenciális entrópiája (2.7. feladat).

A Lloyd–Max-algoritmus

Nem egyenletes kvantáló tervezésénél azt a technikát alkalmazzuk, hogy az átlagos torzítás csökkentéséhez a bemenetet pontosabban közelítjük a nagyobb valószínűségű tartományokban, míg a kis valószínűségű tartományokban rosszabb közelítést engedünk meg, mint egyenletes kvantálás esetén. Ezt úgy tehetjük meg, hogy a nagyobb valószínűségű helyeken a kvantálási intervallumokat kisebbnek

választjuk. Amennyiben az intervallumok száma konstans, ez egyben azt is jelenti, hogy a kisebb valószínűségű helyeken nagyobb intervallumhosszakkal dolgozunk. (Ez hasonlatos ahhoz, hogy veszteségmentes tömörítés esetén az átlagos tömörítési arány javításához a kisebb valószínűséggel előforduló forrásszavakhoz hosszabb kódszavakat rendelünk.)

Egy adott X valószínűségi változóhoz keressük az N szintű, optimális Q kvantálót. Feladatunk az $x_1 < x_2 < \cdots < x_N$ kvantálási szintek és a $Q(X)$ függvény meghatározása úgy, hogy a $D(Q)$ négyzetes torzítás minimális legyen. Bebizonyítható, hogy egy optimális kvantáló kielégíti az alábbi két szükséges feltételt:

1. Legközelebbi szomszéd feltétel:

$$|x - Q(x)| = \min_{1 \le i \le N} |x - x_i| \qquad \forall x \in \mathbb{R},$$

   vagyis minden valós x kvantált $Q(x)$ értéke legalább olyan közel van x-hez, mint bármely másik kvantálási szint.

2. Súlypont feltétel:
   Minden x_j kvantálási szint megegyezik azon X_i minták átlagával, amelyeket erre a szintre kvantálunk ($Q(X_i) = x_j$).

Az 1. és 2. feltételt együtt Lloyd–Max-feltételnek nevezzük, az ezt kielégítő kvantálót pedig Lloyd–Max-kvantálónak. Ha egy kvantáló nem elégíti ki a Lloyd–Max-feltétel bármelyik részét, akkor lehetséges egy olyan kvantálót készíteni, amelynek négyzetes torzítása kisebb, tehát egy nem Lloyd–Max-kvantáló nem lehet optimális, de létezik nem optimális Lloyd–Max-kvantáló is.

2.1. példa. Legyen az X valószínűségi változó az $\{1,2,3,4\}$ halmazon egyenletes eloszlású. Pontosan 3 féle 2-szintű Lloyd–Max-kvantálót alkalmazhatunk erre:

$$Q_1(1) = 1; \qquad Q_1(2) = Q_1(3) = Q_1(4) = 3$$

$$Q_2(4) = 4; \qquad Q_2(1) = Q_2(2) = Q_2(3) = 2$$

$$Q_3(1) = Q_3(2) = 1.5; \qquad Q_3(3) = Q_3(4) = 3.5$$

Q_1 és Q_2 négyzetes torzítása egyaránt 0.5, míg Q_3-é 0.25. Annak ellenére, hogy mindhárom Lloyd–Max-kvantáló, csak Q_3 optimális.

Feltesszük, hogy a kvantálandó X valós valószínűségi változó eloszlása az abszolút folytonos f sűrűségfüggvénnyel adott. Ekkor a Lloyd–Max-feltétel az alábbiak szerint alakul:

1. Legközelebbi szomszéd feltétel:

$$y_i = \frac{x_i + x_{i+1}}{2}, \qquad i = 1, 2, \ldots, N-1,$$

ahol y_{i-1} és y_i annak az intervallumnak a két végpontja, amelyet x_i szintre kvantálunk. (Az nyilvánvaló, hogy minden kvantálási tartomány intervallum.)

2. Súlypont feltétel:

$$x_i = \frac{\int_{y_{i-1}}^{y_i} x f(x)\,\mathrm{d}x}{\int_{y_{i-1}}^{y_i} f(x)\,\mathrm{d}x}, \qquad i = 1, \ldots, N,$$

vagyis minden kvantálási szint a saját kvantálási intervallumának súlypontja.

2.6. tétel (Fleischer). *Legyen az $f(x)$ sűrűségfüggvény pozitív értékű és logaritmikusan konkáv (vagyis $\log f(x)$ legyen konkáv). Ekkor egyetlen N-szintű Lloyd–Max-kvantáló létezik az $f(x)$-re, így ez egyben az egyetlen optimális kvantáló is az $f(x)$-re.*

2.2. példa. Legyen $f(x)$ az egyenletes eloszlás sűrűségfüggvénye $[a,b]$-n. Könnyű ellenőrizni, hogy az N-szintű egyenletes kvantáló az $[a,b]$ intervallumon kielégíti a Lloyd–Max-feltételt az $f(x)$-re. Az egyenletes eloszlás logaritmikusan konkáv, ezért az N-szintű egyenletes kvantáló az egyetlen optimális N-szintű kvantáló az egyenletes eloszlásra.

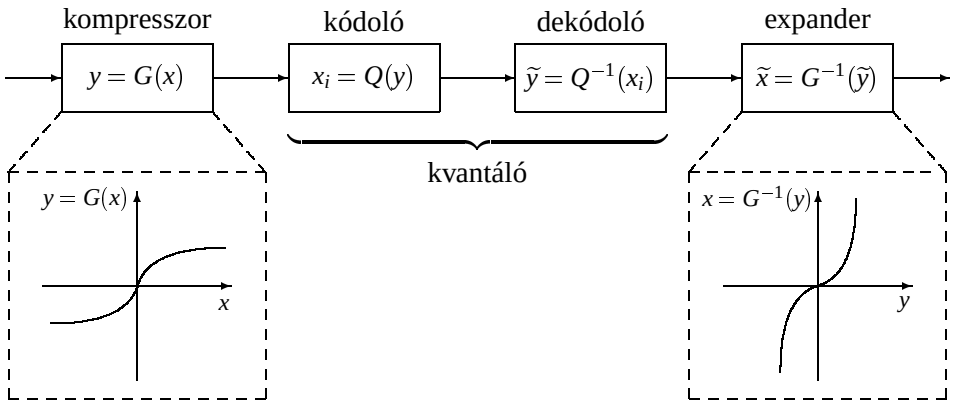
Az algoritmus:
A kvantálót egyértelműen jellemzik az x_i kvantálási szintek és a $B_i = (y_{i-1}, y_i]$ tartományok. Célunk a szintek és az intervallumhatárok javítása lépésről lépésre.

1. Vegyünk fel egy közelítést a kvantálási szintekre.

2. Optimalizáljuk a kvantálót a kvantálási szintek szerint, vagyis határozzuk meg az intervallumhatárokat a legközelebbi szomszéd feltétel kielégítésével.

3. Számítsuk ki, hogy mennyivel csökkent a torzítás, és ha ez egy előre meghatározott küszöbértéknél kisebb, akkor készen vagyunk.

4. Optimalizáljuk a kvantálót az imént kapott intervallumokhoz, vagyis alkalmazzuk a súlypont feltételt, és folytassuk az algoritmust a 2. ponttól.



2.2. ábra. Kompanderes kvantáló.

Kompanderes kvantáló

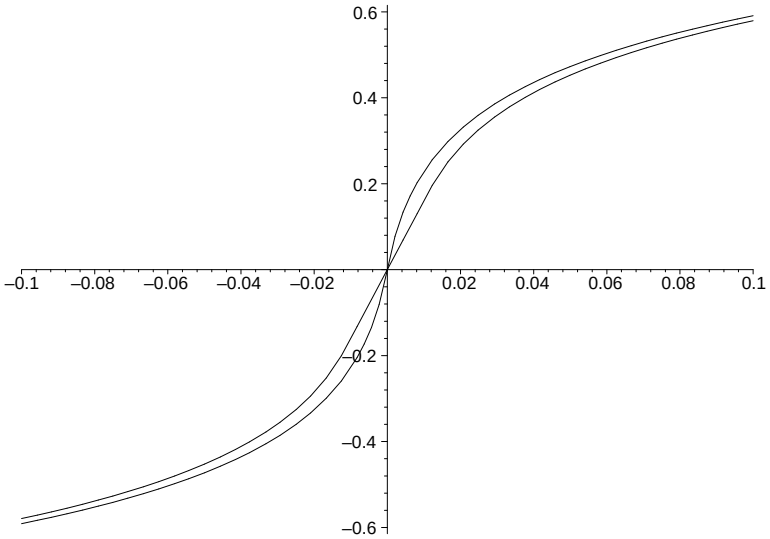
A kvantálásra a gyakorlatban általában olyan alkatrészek érhetők el, melyekkel egyenletes kvantálást valósíthatunk meg. Ebben az esetben a kvantált értéket kell tömöríteni. Gyakran ezt nem akarjuk megtenni, sokkal inkább azt szeretnénk, hogy rögzítve a kvantálási szintek számát, N-et, minimalizáljuk a négyzetes torzítást. Ezt úgy tesszük, hogy a kvantálandó jelet egy monoton növekedő függvénynyel, a kompresszorral a $\left[-\frac{1}{2},\frac{1}{2}\right]$-be képezzük, ott alkalmazunk egy egyenletes kvantálót, és a kvantált értéket a kompresszor inverzével (az expanderrel) visszatranszformáljuk (2.2. ábra). (kompander = kompresszor + expander)

A leggyakrabban alkalmazott nemegyenletes kvantálók a logaritmikus kompresszor karakterisztikát használó eszközök, amelyek a távközlő hálózatokban a beszédkódolást végzik a '60-as évek óta. Azért alkalmaznak logaritmikus kvantálást, mert az emberi beszédben a kis amplitúdójú jelek az érthetőség szempontjából rendkívül fontosak, ezért a lehető legnagyobb pontossággal kell ezeket kvantálni, míg a nagy amplitúdójú jelek esetében a túl nagy jelszintet kell megakadályozni. A beszédkódolásban kétféle kompander használatos: a μ-law és az A-law. Az előbbi az Egyesült Államokban, Kanadában és Japánban elterjedt. Kompreszszor ill. expander függvénye:

$$G_\mu(x) = \frac{\ln(1+\mu|x|)}{\ln(1+\mu)}\operatorname{sgn}x \qquad -1 \le x \le 1$$

$$G_\mu^{-1}(x) = \frac{1}{\mu}\left((1+\mu)^{|x|}-1\right)\operatorname{sgn}x \qquad -1 \le x \le 1$$

A μ paraméter értékét általában 255-nek választják.



2.3. ábra. A-law és μ-law kompresszor függvények.

Az Európában, Afrikában, Ausztráliában és Dél-Amerikában alkalmazott A-law kompandere az alábbi ($A = 87.6$):

$$G_A(x) = \begin{cases} \frac{A|x|}{1+\ln A}\operatorname{sgn} x, & 0 \le |x| < \frac{1}{A} \\ \frac{1+\ln|Ax|}{1+\ln A}\operatorname{sgn} x, & \frac{1}{A} \le |x| \le 1 \end{cases}$$

$$G_A^{-1}(x) = \begin{cases} \frac{|x|\,|1+\ln A|}{A}\operatorname{sgn} x, & 0 \le |x| < \frac{1}{1+\ln A} \\ \frac{e^{|x|(1+\ln A)-1}}{A}\operatorname{sgn} x, & \frac{1}{1+\ln A} \le |x| \le 1 \end{cases}$$

Az alapvető különbség az A-law és a μ-law karakterisztika között, hogy az A-lawnak kicsit szélesebb a dinamikatartománya (értékkészletének terjedelme), míg a μ-lawnak egy kicsit kisebb az alapzaja. A 2.3. ábrán látható a két kompresszor függvény. A különbség csak a 0-hoz közeli tartományban figyelhető meg, ezért a $[-0.1, 0.1]$ intervallumban ábrázoltuk a őket (a vízszintes tengelyhez közelebb haladó görbe az A-law, míg a távolabbi a μ-law).

A finom, egyenletes kvantáló négyzetes torzítása kiterjeszthető a kompandéres esetre. Jelölje $G(x)$ a kompresszort és $g(x) = G'(x)$ a deriváltját. Legyenek

$$-\frac{1}{2} = y'_0 < y'_1 < \cdots < y'_N = \frac{1}{2}$$

a transzformált intervallumban a kvantálási határok, $y'_i - y'_{i-1} = q' = \frac{1}{N}$, és

$$y_i = G^{-1}(y'_i), \qquad i = 1, 2, \ldots, N-1$$

a valós kvantálási határok. Ekkor az i-edik kvantálási lépcső nagysága

$$q_i = y_i - y_{i-1}.$$

A (2.13)-hoz hasonlóan beláthatjuk, hogy

$$\begin{aligned} D(Q_N) \simeq \bar{D}(Q_N) &= \sum_{i=1}^{N} \int_{y_{i-1}}^{y_i} f(z)\,\mathrm{d}z \frac{q_i^2}{12} = \\ &= \frac{1}{12N^2} \sum_{i=1}^{N} \frac{\int_{y_{i-1}}^{y_i} f(z)\,\mathrm{d}z}{\left(\frac{G(y_i)-G(y_{i-1})}{y_i-y_{i-1}}\right)^2} = \\ &= \frac{1}{12N^2} \sum_{i=1}^{N} \frac{\frac{1}{y_i-y_{i-1}} \int_{y_{i-1}}^{y_i} f(z)\,\mathrm{d}z}{\left(\frac{1}{y_i-y_{i-1}} \int_{y_{i-1}}^{y_i} g(z)\,\mathrm{d}z\right)^2} (y_i - y_{i-1}) = \\ &= \frac{1}{12N^2} \sum_{i=1}^{N} \frac{f(z_i)}{g^2(z_i)} (y_i - y_{i-1}) \end{aligned}$$

valamilyen $y_{i-1} \le z_i \le y_i$-re. A

$$\sum_{i=1}^{N} \frac{f(z_i)}{g^2(z_i)} (y_i - y_{i-1})$$

az

$$\int \frac{f(z)}{g^2(z)}\,\mathrm{d}z$$

egy integrálközelítő-összege. Azt kaptuk, hogy

$$D(Q_N) \simeq \frac{1}{12N^2} \int \frac{f(z)}{g^2(z)}\,\mathrm{d}z.$$

Ismerve az $f(x)$ sűrűségfüggvényt, megkereshetjük az „optimális” kompandert. Ehhez felhasználjuk a Hölder-egyenlőtlenséget.

2.1. lemma (Hölder-egyenlőtlenség). *Legyen $p,q \geq 1$ úgy, hogy $\frac{1}{p}+\frac{1}{q}=1$. Ekkor, ha $h_1(x)$ és $h_2(x)$ két olyan függvény, hogy*

$$\int |h_1(x)|^p \,\mathrm{d}x \qquad \text{és} \qquad \int |h_2(x)|^q \,\mathrm{d}x$$

véges, akkor

$$\left|\int h_1(x)h_2(x)\,\mathrm{d}x\right| \leq \left(\int |h_1(x)|^p \,\mathrm{d}x\right)^{\frac{1}{p}} \cdot \left(\int |h_2(x)|^q \,\mathrm{d}x\right)^{\frac{1}{q}}.$$

Alkalmazzuk a Hölder-egyenlőtlenséget $p=3,\ q=\frac{3}{2},\ h_1(x)=\left(\frac{f(x)}{g^2(x)}\right)^{\frac{1}{3}}$, $h_2(x)=g^{2/3}(x)$ szereposztásban:

$$\begin{aligned}
\int f^{1/3}(x)\,\mathrm{d}x &= \int h_1(x)h_2(x)\,\mathrm{d}x \leq \\
&\leq \left(\int h_1^3(x)\,\mathrm{d}x\right)^{\frac{1}{3}} \left(\int h_2^{3/2}(x)\,\mathrm{d}x\right)^{\frac{2}{3}} = \\
&= \left(\int \frac{f(x)}{g^2(x)}\,\mathrm{d}x\right)^{\frac{1}{3}} \left(\int g(x)\,\mathrm{d}x\right)^{\frac{2}{3}} = \\
&= \left(\int \frac{f(x)}{g^2(x)}\,\mathrm{d}x\right)^{\frac{1}{3}},
\end{aligned}$$

ahol kihasználtuk, hogy $\int\limits_{-\infty}^{\infty} g(z)\,\mathrm{d}z = G(\infty)-G(-\infty)=\frac{1}{2}-\left(-\frac{1}{2}\right)=1$. Azt kaptuk, hogy

$$\int \frac{f(x)}{g^2(x)}\,\mathrm{d}x \geq \left(\int f^{1/3}(x)\,\mathrm{d}x\right)^3,$$

és visszahelyettesítéssel ellenőrizhetjük, hogy ezt az optimumot akkor érhetjük el, ha

$$g^*(x) = c\,f^{1/3}(x) = \frac{f^{1/3}(x)}{\int f^{1/3}(z)\,\mathrm{d}z},$$

következésképp

$$G^*(x) = -\frac{1}{2} + \frac{\int\limits_{-\infty}^{x} f^{1/3}(z)\,\mathrm{d}z}{\int\limits_{-\infty}^{\infty} f^{1/3}(z)\,\mathrm{d}z}.$$

Vektorkvantálás

A forrás kimenetének többdimenziós eloszlását felhasználva kisebb torzítást érhetünk el ugyanakkora bit/minta arány mellett vektorkvantálással. A forrásszimbólumok egyenkénti kvantálása helyett (ahogyan azt skalár kvantáló esetén tettük) szekvenciákat, vektorokat képezünk belőlük, s ezeket együtt kvantáljuk. (Tekintsük például a kétdimenziós esetet. Skalár kvantáló alkalmazása esetén — egymásra merőleges tengelyeket feltételezve — téglalap alakú tartományokat kapunk a síkban, míg vektorkvantálóval bármilyen szabálytalan síkidom alakú (például kör) tartományok kialakíthatók.)

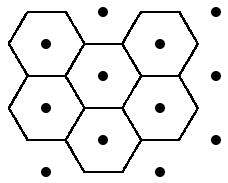
Tegyük fel, hogy a forrásunk emberek magasság- és tömegadatait generálja. Ezek például 100 és 200 cm, illetve 20 és 120 kg közé esnek. Ha összesen 6 biten szeretnénk kvantálni az adatainkat, és skalár kvantálót alkalmazunk, akkor 3-3 bit jut a magasságra és a tömegre egyaránt. Az intervallumokat 8 egyenlő részre osztva, azok közepén kijelölve a szinteket (a szemléletesség kedvéért kétdimenziós koordinátarendszerben ábrázolva az így lehetséges kvantálási pontokat: magasság–testtömeg koordinátájú pontok), egy (22 kg, 197 cm) adatpárt ugyanolyan torzítással kvantálunk, mint egy (70 kg, 180 cm)-est. Holott nyilvánvalóan az előbbi adatpár nem fog előfordulni, míg az utóbbi gyakori lesz. Vektorkvantálás alkalmazásával megtehetjük, hogy például a statisztikai vizsgálatok szerint leggyakoribb magasság–testtömeg egyenes környezetében biztosítunk kis torzítást, míg a gyakorlatilag lehetetlen adatpárok esetében megengedünk nagyobb torzítást is.

Legyen $\mathbf{X}$ egy d-elemű forrásvektor $f(\mathbf{x})$ sűrűségfüggvénnyel. A d-dimenziós vektorkvantáló egy $Q(\mathbf{x})$ függvény, amely az $\mathbf{x} \in \mathbb{R}^d$ bemenetet az $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in \mathbb{R}^d$ vektorok egyikébe képezi le. A kvantálót egyértelműen jellemzi az N kimeneti vektor, és a hozzájuk tartozó $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_N$ tartományok, amelyek $\mathbb{R}^d$ egy partícióját alkotják (diszjunktak, és lefedik $\mathbb{R}^d$-t), tehát $Q(\mathbf{x}) = \mathbf{x}_i$, ha $\mathbf{x} \in \mathcal{B}_i$, $i = 1, 2, \ldots, N$. A torzítás vizsgálatára továbbra is a négyzetes torzítási mértéket fogjuk használni:

$$D(Q) = \frac{1}{d}\mathbf{E}\|\mathbf{X} - Q(\mathbf{X})\|^2 = \frac{1}{d}\sum_{i=1}^{N} \int\limits_{\mathbf{x}\in\mathcal{B}_i} \|\mathbf{x} - \mathbf{x}_i\|^2 f(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

ahol $\|\cdot\|$ az euklidészi norma.

Vektorkvantáló tervezése adott torzítási mértékre ismert bemeneti eloszlás esetén a $\mathcal{B}_i$ tartományok és az $\mathbf{x}_i$ kimeneti vektorok meghatározását jelenti. Míg egydimenziós esetben ez viszonylag egyszerű problémát jelentett, hiszen csak a valós egyenes intervallumai jöhettek számításba, addig most még $\mathcal{B}_i$-k alakjára is végtelen sok lehetőségünk van. A torzítás szempontjából optimális megoldás $\mathcal{B}_i$-k



2.4. ábra. Hatszögminta.

alakjára a kör, a gömb, illetve magasabb dimenziókban a hipergömb lenne. Sajnos azonban ezekkel az alakzatokkal nem lehet hézagmentesen lefedni (csempézni) a teret, pedig minden lehetséges bemeneti vektorhoz pontosan egy kimeneti vektort kell hozzárendelnünk. Kétdimenziós esetben jó közelítést jelent a síkot lefedő szabályos hatszögminta, amely a 2.4. ábrán látható.

Egy optimális vektorkvantáló kielégíti az alábbi két szükséges feltételt, melyek az egydimenziós eset kézenfekvő általánosításai:

1. $\mathbb{R}^d$ partíciója Dirichlet-partíció, vagy más néven tartományai Voronoi-tartományok, azaz

$$\mathcal{B}_i = \{\mathbf{x} : \ \|\mathbf{x}-\mathbf{x}_i\| \leq \|\mathbf{x}-\mathbf{x}_j\|, \quad \forall j \neq i\}.$$

2. A kimeneti vektorok a hozzájuk tartozó tartományok súlypontjai:

$$\mathbf{x}_i = \operatorname*{argmin}_{\mathbf{y}} \int_{\mathcal{B}_i} \|\mathbf{x}-\mathbf{y}\|^2 f(\mathbf{x})\,\mathrm{d}\mathbf{x}.$$

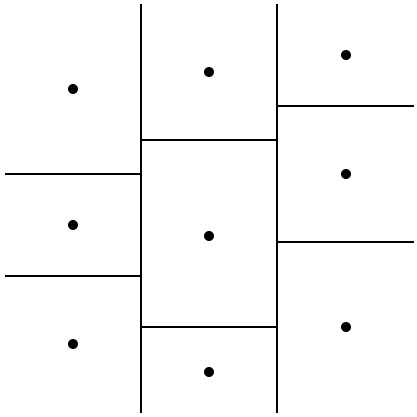
A Lloyd–Max-algoritmus általánosításaként vektorkvantálás esetén a **Linde–Buzo–Gray-algoritmus** használatos:

1. Vegyünk fel egy közelítést a kvantálási vektorokra.

2. Optimalizáljuk a kvantálót a kvantálási vektorok szerint, vagyis határozzuk meg a tartományokat a Voronoi-tartományokra vonatkozó feltétel kielégítésével.

3. Számítsuk ki, hogy mennyivel csökkent a torzítás, és ha ez egy előre meghatározott küszöbértéknél kisebb, akkor készen vagyunk.

4. Optimalizáljuk a kvantálót az imént kapott tartományokhoz, vagyis alkalmazzuk a súlypont feltételt, és folytassuk az algoritmust a 2. ponttól.



2.5. ábra. Fa-struktúrájú vektorkvantáló.

A Linde–Buzo–Gray-algoritmussal tervezett vektorkvantáló kódkönyvének általában nincs megfigyelhető belső struktúrája. Ez a „véletlenszerű” szerkezet megnehezíti ugyan a kvantálási folyamatot, de ez biztosítja a vektorkvantáló közel optimális torzítását. Több megoldás létezik struktúrált, de ugyanakkor kedvező torzítással rendelkező vektorkvantáló tervezésére.

A **fa-struktúrájú** vektorkvantáló alapelve az, hogy az $L = K^d$ kimeneti vektor közül a közel optimálisat egy d mélységű és K-adfokú fa segítségével keressük meg. Egy szint egy bejegyzése a következő szint egy K darab vektort tartalmazó halmazára mutat. Így összesen legfeljebb $d\lceil \log K \rceil$ összehasonlítást kell végrehajtanunk a teljes keresés K^d összehasonlítása helyett. A 2.5. ábra a $d = 2, K = 3$ esetet szemlélteti, ahol az első koordináta szerint alkalmazunk egy skalár kvantálót, majd részintervallumonként külön tervezünk egy skalár kvantálót a második koordinátára.

Néhány alkalmazásban, mint például a beszédfeldolgozásban, a bemeneti jel dinamikatartománya széles skálán változhat. Az ehhez szükséges nagy kódkönyvben való keresés helyett először normáljuk a vektort, majd külön-külön kvantáljuk a normalizált vektort, és a normalizáló faktort. A normalizáló faktor a dinamikatartománybeli helyet, vagyis az energiát határozza meg, míg a normalizált vektor a jel formáját, így ezt a technikát **energia–forma vektorkvantáló**nak nevezzük.

Osztott vektorkvantálás esetén a forrásvektorokat független osztályokba soroljuk térbeli tulajdonságuk alapján. A különböző osztályokhoz különböző vektorkvantálókat tervezünk. Ez a technika előnyös például a képtömörítésben, ahol az éleket tartalmazó illetve nem tartalmazó tartományok két eltérő osztályt alkotnak.

Többszintű vektorkvantálás esetén több menetben dolgozunk. Először egy alacsony bitarányú kvantálóval előállítjuk a bemenet durva közelítését, majd lépésről lépésre mindig az eredeti bemenet és az előző szint által előállított közelítés eltérését (a kvantálási hibát) kvantáljuk.

A képek több nagyobb, közel egyszínű „foltot” tartalmaznak, így jól működik az a megoldás, hogy a képpontok (blokkonkénti) átlagát egy skalár kvantálóval kvantáljuk, majd a képpontokból kivonva ezt az átlagot egy vektorkvantálót alkalmazunk.

2.3. Lineáris szűrés

A részsávos kódolás (subband coding) bevezetéséhez és a mintavételezéshez szükségünk van a gyengén stacionárius folyamat spektrális sűrűségfüggvényének és a gyengén stacionárius folyamatok lineáris szűrésének fogalmára. Legyen $X(t)$ egy 0 várható értékű, $R(\tau)$ kovarianciafüggvényű gyengén stacionárius folyamat, azaz

$$R(\tau) = \mathbf{E}(X(t+\tau)X(t))$$

minden t-re. Ekkor $R(\tau)$ pozitív szemidefinit, tehát tetszőleges $f(t)$ függvényre

$$\iint R(t-s)\,f(t)\,f(s)\,\mathrm{d}t\mathrm{d}s \geq 0.$$

Ezt úgy láthatjuk be, hogy megmutatjuk, hogy a bal oldal

$$\mathbf{E}\left(\int X(t)\,f(t)\,\mathrm{d}t\right)^2,$$

ami nemnegatív.

Tegyük fel, hogy $R(\tau)$-nak létezik a Fourier-transzformáltja:

$$s(\omega) = \int_{-\infty}^{\infty} R(t)\mathrm{e}^{j\omega t}\,\mathrm{d}t.$$

$s(\omega)$-t az $X(t)$ folyamat **spektrális sűrűségfüggvény**ének nevezzük. $R(\tau)$ szimmetriája miatt $s(\omega)$ is szimmetrikus, továbbá $R(\tau)$ pozitív szemidefinit tulajdonságából következik, hogy

$$s(\omega) \geq 0.$$

Legyen $h(t)$ egy négyzetesen integrálható függvény:

$$\int_{-\infty}^{\infty} h(t)^2\,\mathrm{d}t < \infty.$$

Ekkor az $X(t)$ folyamat **lineáris szűrés**én az

$$Y(t) = \int_{-\infty}^{\infty} h(s)X(t-s)\,\mathrm{d}s$$

folyamatot értjük. (A határértéket négyzetes középben vesszük.) Belátható, hogy $Y(t)$ is egy 0 várható értékű, gyengén stacionárius folyamat. $h(t)$-t a szűrő **súlyfüggvény**ének nevezzük. A $h(t)$ Fourier-transzformáltját **átviteli függvény**nek nevezzük:

$$H(\omega) = \int_{-\infty}^{\infty} h(t)\mathrm{e}^{j\omega t}\,\mathrm{d}t.$$

2.3. példa. Legyen $0 < b < B$, és

$$H(\omega) = \begin{cases} 1, & \text{ha } |\omega| \in [b,B] \\ 0, & \text{egyébként} \end{cases}$$

Az ilyen átviteli függvényű szűrőt ideális **sávszűrő**nek nevezzük, mert az

$$X(t) = A\sin(\omega t + \varphi)$$

gyengén stacionárius folyamat szűrtje $X(t)$, ha $|\omega| \in [b,B]$, és azonosan 0 egyébként. (Itt φ a $[0,2\pi]$-ben egyenletes eloszlású.)

Számoljuk ki az $Y(t)$ szűrt folyamat $\widetilde{R}(\tau)$ kovarianciafüggvényét:

$$\begin{aligned}
\widetilde{R}(\tau) &= \mathbf{E}(Y(\tau)Y(0)) = \\
&= \mathbf{E}\left(\int_{-\infty}^{\infty} h(s)X(\tau-s)\,\mathrm{d}s \int_{-\infty}^{\infty} h(t)X(0-t)\,\mathrm{d}t\right) = \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(s)h(t)\mathbf{E}(X(\tau-s)X(-t))\,\mathrm{d}s\mathrm{d}t = \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(s)h(t)R(\tau+t-s)\,\mathrm{d}s\mathrm{d}t = \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(s)h(t)\frac{1}{2\pi}\int_{-\infty}^{\infty} \mathrm{e}^{-j\omega(\tau+t-s)}s(\omega)\,\mathrm{d}\omega\mathrm{d}s\mathrm{d}t = \\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty}\left(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(s)h(t)\mathrm{e}^{-j\omega(t-s)}\,\mathrm{d}s\mathrm{d}t\right)\mathrm{e}^{-j\omega\tau}s(\omega)\,\mathrm{d}\omega =
\end{aligned}$$

$$= \frac{1}{2\pi}\int_{-\infty}^{\infty} |H(\omega)|^2 \mathrm{e}^{-j\omega\tau} s(\omega)\,\mathrm{d}\omega$$

(Az integrálások sorrendjének felcserélhetőségét most nem ellenőriztük.)

Ez alapján kiszámíthatjuk a szűrt folyamat teljesítményét:

$$\mathbf{E}(Y(0)^2) = \widetilde{R}(0) = \frac{1}{2\pi}\int_{-\infty}^{\infty} |H(\omega)|^2 s(\omega)\,\mathrm{d}\omega.$$

A sávszűrő esetén

$$\widetilde{R}(0) = \frac{1}{2\pi}\int_{|\omega|\in[b,B]} s(\omega)\,\mathrm{d}\omega,$$

és ez indokolja a spektrális sűrűségfüggvény elnevezést. Folytonos $s(\omega)$ és kis $B-b$ (tűszűrő) esetén $\widetilde{R}(0) \approx \frac{1}{2\pi}\cdot 2(B-b)s(b)$, a szűrt jel energiája közelítőleg $\frac{1}{\pi}(B-b)s(b)$.

2.4. Mintavételezés

Legyen $X(t)$ egy 0 várható értékű, gyengén stacionárius folyamat $R(\tau)$ kovarianciafüggvénnyel és $s(\omega)$ spektrális sűrűségfüggvénnyel. Az $X(t)$ folyamat most mind időben, mind értékkészletében folytonos, tehát egyáltalán nem nyilvánvaló, hogy mikor lehetséges torzításmentes adattömörítés. Ebben a szakaszban módszert mutatunk az idő „diszkretizálására”, ez a mintavételezés. Legyen $T > 0$ a **mintavételi idő**, az $\{X(kT),\ k = 0, \pm 1, \pm 2, \ldots\}$ diszkrét idejű folyamatot az $X(t)$ folyamat **mintavételezés**ének nevezzük.

Az a kérdés, hogy a mintavételezésből mikor reprodukálható az $X(t)$ tökéletesen.

Választunk egy interpoláló függvényt:

$$f(t) = \frac{\sin(\pi t)}{\pi t}.$$

Nyilván $f(0) = 1$ és $f(k) = 0,\ k = \pm 1, \pm 2, \ldots$.

Definiáljuk a reprodukciót:

$$\widehat{X}(t) = \sum_{k=-\infty}^{\infty} X(kT)\, f\left(\frac{t}{T} - k\right),$$

vagyis a T-vel átskálázott interpoláló függvény $X(kT)$-szeresét eltoljuk a k-adik mintavételi pontba, és ezeket a függvényeket összegezzük. A végtelen összegzésben a határértéket négyzetes középben értjük, azaz

$$\lim_{N\to\infty} \mathbf{E}\left(\widehat{X}(t) - \sum_{k=-N}^{N} X(kT)\, f\left(\frac{t}{T} - k\right)\right)^2 = 0.$$

Nyilván $t = kT$ esetén

$$\widehat{X}(t) = X(t),$$

vagyis a mintavételi időpontokban a reprodukció tökéletes.

2.7. tétel (Mintavételi tétel). *Ha az $X(t)$ folyamat B sávra korlátozott, vagyis*

$$R(0) = \frac{1}{2\pi}\int_{-\infty}^{\infty} s(\omega)\,\mathrm{d}\omega = \frac{1}{2\pi}\int_{-B}^{B} s(\omega)\,\mathrm{d}\omega,$$

és

$$T < \frac{\pi}{B},$$

akkor minden t-re

$$\mathbf{P}\{X(t) = \widehat{X}(t)\} = 1.$$

Ha bevezetjük a $B' = \frac{B}{2\pi}$ frekvenciát, akkor a mintavételi tétel T-re az alábbi feltételt adja:

$$T < \frac{1}{2B'},$$

vagyis a mintavételi frekvencia legalább a B' duplája legyen.

MEGJEGYZÉS: A mintavételezés egy olyan forráskódolási eljárás, amikor a folytonos idejű stacionárius folyamatot a mintáiba kódolunk, és a reprodukció interpolációval történik. Vegyük észre, hogy ez a tökéletes reprodukció csak egy elvi lehetőség, hiszen az interpoláció az összes, tehát végtelen sok mintát használ. Megmutatható, hogy ez az interpoláció megvalósítható ideális sávszűrővel. Ez a sávszűrő annál jobban közelíthető realizálható szűrővel, minél nagyobb a mintavételi frekvencia a sávszélesség duplájánál. Így például a telefóniában a beszédet 3400 Hz-re sávszűrik, és azt 8000 Hz-cel mintavételezik.

A mintavételi tétel bizonyításához előbb egy lemmát bizonyítunk:

2.2. lemma. *Legyen*

$$H_t(\omega) = \sum_k \mathrm{e}^{j\omega(t-k)} f(t-k),$$

ekkor $|\omega| < \pi$ *esetén*

$$H_t(\omega) = 1.$$

BIZONYÍTÁS: Jelölje

$$H(\omega) = \int_{-\infty}^{\infty} \mathrm{e}^{j\omega t} f(t)\,\mathrm{d}t$$

az $f(t)$ interpoláló függvény Fourier-transzformáltját! Először megmutatjuk, hogy

$$H(\omega) = \begin{cases} 1, & \text{ha } |\omega| \le \pi \\ 0, & \text{egyébként} \end{cases},$$

ugyanis

$$\begin{aligned} \frac{1}{2\pi}\int_{-\infty}^{\infty} H(\omega)\mathrm{e}^{-j\omega t}\,\mathrm{d}\omega &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \mathrm{e}^{-j\omega t}\,\mathrm{d}\omega = \\ &= \frac{1}{2\pi}\left[\frac{\mathrm{e}^{-j\omega t}}{-jt}\right]_{-\pi}^{\pi} = \\ &= \frac{\mathrm{e}^{-j\pi t} - \mathrm{e}^{j\pi t}}{-2\pi jt} = \\ &= \frac{\sin(\pi t)}{\pi t} = \\ &= f(t), \end{aligned}$$

tehát $H(\omega)$ inverz Fourier-transzformáltja $f(t)$.

Írjuk fel a $G_t(\omega) = H(\omega)\mathrm{e}^{-j\omega t}$ függvény Fourier-sorát a $[-\pi,\pi]$-ben:

$$G_t(\omega) = \sum_k g_{k,t}\mathrm{e}^{-j\omega k},$$

ahol

$$\begin{aligned} g_{k,t} &= \frac{1}{2\pi}\int_{-\pi}^{\pi} G_t(\omega)\overline{\mathrm{e}^{-j\omega k}}\,\mathrm{d}\omega \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} H(\omega)\mathrm{e}^{-j\omega t}\mathrm{e}^{j\omega k}\,\mathrm{d}\omega \end{aligned}$$

$$\begin{aligned} &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \mathrm{e}^{-j\omega(t-k)}\,\mathrm{d}\omega \\ &= f(t-k), \end{aligned}$$

tehát $|\omega| < \pi$ esetén

$$H(\omega)\mathrm{e}^{-j\omega t} = \sum_k f(t-k)\mathrm{e}^{-j\omega k},$$

következésképp

$$1 = H(\omega) = \sum_k f(t-k)\mathrm{e}^{j\omega(t-k)} = H_t(\omega).$$

■

A 2.7. TÉTEL BIZONYÍTÁSA:
Itt csak egy bizonyításvázlatot adunk, mivel nem ellenőrizzük a különböző felcserélések jogosságát. Megmutatjuk, hogy $\mathbf{E}(X(t)-\widehat{X}(t))^2 = 0$, amiből már következik, hogy $\mathbf{P}\{X(t) = \widehat{X}(t)\} = 1$.

$$\mathbf{E}(X(t)-\widehat{X}(t))^2 = \mathbf{E}(X^2(t)) - 2\mathbf{E}(X(t)\widehat{X}(t)) + \mathbf{E}(\widehat{X}^2(t)) \tag{2.20}$$

Számítsuk ki a három tagot!

$$\mathbf{E}(X^2(t)) = \mathbf{E}(X^2(0)) = R(0) \tag{2.21}$$

A keresztszorzatnál kihasználjuk a sávkorlátozottságot és a 2.2. lemmát:

$$\begin{aligned} \mathbf{E}(X(t)\widehat{X}(t)) &= \mathbf{E}\left(X(t)\sum_k X(kT) f\left(\frac{t}{T}-k\right)\right) = \\ &= \sum_k R(t-kT) f\left(\frac{t}{T}-k\right) = \\ &= \sum_k \frac{1}{2\pi}\int_{-\infty}^{\infty} \mathrm{e}^{-j\omega(t-kT)} s(\omega)\,\mathrm{d}\omega\, f\left(\frac{t}{T}-k\right) = \\ &= \frac{1}{2\pi}\sum_k \int_{-B}^{B} \mathrm{e}^{-j\omega(t-kT)} s(\omega)\,\mathrm{d}\omega\, f\left(\frac{t}{T}-k\right) \end{aligned}$$

Vezessük be a $t' = \frac{t}{T}$ jelölést, ekkor

$$\mathbf{E}(X(t)\widehat{X}(t)) = \frac{1}{2\pi}\sum_k \int_{-B}^{B} \mathrm{e}^{j\omega T(t'-k)} s(\omega)\,\mathrm{d}\omega\, f(t'-k) =$$

$$= \frac{1}{2\pi}\int_{-B}^{B} H_{t'}(\omega T)s(\omega)\,\mathrm{d}\omega =$$

$$= \frac{1}{2\pi}\int_{-B}^{B} s(\omega)\,\mathrm{d}\omega = R(0), \tag{2.22}$$

amennyiben $|\omega|T < \pi$, ami teljesül, ha $BT < \pi$, és ezt feltettük. Nézzük a harmadik tagot!

$$\begin{aligned}
\mathbf{E}(\widehat{X}^2(t)) &= \sum_k\sum_l \mathbf{E}(X(kT)X(lT))f\left(\frac{t}{T}-k\right)f\left(\frac{t}{T}-l\right) = \\
&= \sum_k\sum_l R((k-l)T)f\left(\frac{t}{T}-k\right)f\left(\frac{t}{T}-l\right) = \\
&= \sum_k\sum_l \frac{1}{2\pi}\int_{-\infty}^{\infty} \mathrm{e}^{-j\omega(k-l)T}s(\omega)\,\mathrm{d}\omega\, f\left(\frac{t}{T}-k\right)f\left(\frac{t}{T}-l\right) = \\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty}\left(\sum_k \mathrm{e}^{-j\omega(kT-t)}f\left(\frac{t}{T}-k\right)\right)\cdot \\
&\quad \left(\sum_l \mathrm{e}^{j\omega(lT-t)}f\left(\frac{t}{T}-l\right)\right)s(\omega)\,\mathrm{d}\omega = \\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty} H_{t'}(\omega T)H_{t'}(-\omega T)s(\omega)\,\mathrm{d}\omega = \\
&= \frac{1}{2\pi}\int_{-B}^{B} H_{t'}(\omega T)H_{t'}(-\omega T)s(\omega)\,\mathrm{d}\omega = \\
&= \frac{1}{2\pi}\int_{-B}^{B} s(\omega)\,\mathrm{d}\omega = R(0)
\end{aligned} \tag{2.23}$$

A (2.20), (2.21), (2.22) és (2.23) miatt

$$\mathbf{E}(X(t)-\widehat{X}(t))^2 = R(0)-2R(0)+R(0) = 0. \qquad \blacksquare$$

2.5. Transzformációs kódolás

Ebben a szakaszban olyan technikát mutatunk be, amelynek segítségével a forrás kimenetét összetevőire bonthatjuk, és ezen összetevőket saját, jellemző karakte-

risztikájuk szerint kódoljuk.

A **transzformációs kódolás** a vektorkvantálás egy speciális esete. Az optimális vektorkvantáló nagy számítási bonyolultságú, míg a struktúrált vektorkvantáló nem optimális. A transzformációs kódoló megkísérli a kettőt ötvözni: előbb transzformálja a jelsorozatot, majd a transzformált sorozat komponenseit kvantálja skalár kvantálókkal. Ez így együtt egy struktúrált vektorkvantáló. Ugyanakkor az egyes komponenseket más és más finomságú skalár kvantálóval kvantáljuk, tehát a vektorkvantáló reprodukciós vektorai egy olyan többdimenziós rácsot alkotnak, melyet megkaphatunk egy többdimenziós téglatest eltolásaival.

A transzformációs kódolás lépései a következőek:

Először osszuk fel a kódolandó $\{x_n\}$ jelsorozatunkat k hosszú diszjunkt blokkokra. Minden egyes blokkra alkalmazzunk egy invertálható transzformációt, mely az $\{y_n\}$ sorozatot eredményezi.

Másodszor, kvantáljuk a transzformált sorozatot. Az alkalmazott kvantálási stratégia függ a kívánt bitsebességtől, a transzformált sorozat különböző részeinek eltérő statisztikai tulajdonságaitól és a megengedhető torzítástól.

A harmadik lépésben kódoljuk a kvantált értékeket valamilyen bináris kóddal.

A továbbiakban csak lineáris transzformációval foglalkozunk. Ekkor a transzformált sorozat a következő alakban áll elő:

$$y_n = \sum_{i=0}^{k-1} a_{n,i} x_i$$

Az $\{y_n\}$ sorozat elemeinek eloszlása a sorozaton belül elfoglalt pozíciótól, n-től függ. A transzformált sorozat elemei eloszlásának terjedelmét a $\boldsymbol{\sigma}_n^2$ szórásnégyzettel mérjük. Ez fogja befolyásolni, hogy hogyan kódoljuk a transzformált sorozatot.

Az eredeti sorozat helyreállítható a transzformált sorozatból az inverz transzformációval:

$$x_n = \sum_{i=0}^{k-1} b_{n,i} y_i$$

Ugyanezek mátrixos alakban:

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

$$\mathbf{x} = \mathbf{B}\mathbf{y}$$

ahol $\mathbf{A}$ és $\mathbf{B}$ $k \times k$-as mátrixok, melyek i, j-edik eleme $a_{i,j}$ illetve $b_{i,j}$, és

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}$$

vagyis

$$\mathbf{B} = \mathbf{A}^{-1}.$$

A képtömörítésnél használt kétdimenziós esetben:

$$\mathbf{Y} = \mathbf{A}\mathbf{X}\mathbf{A}^T$$

$$\mathbf{X} = \mathbf{B}\mathbf{Y}\mathbf{B}^T$$

Ezek gyakran ortonormált transzformációk, vagyis a transzformáló mátrix inverze megegyezik a transzponáltjával: $\mathbf{B} = \mathbf{A}^{-1} = \mathbf{A}^T$, így

$$\mathbf{X} = \mathbf{A}^T\mathbf{Y}\mathbf{A}.$$

Az ortonormált transzformáció megőrzi az energiát, vagyis az eredeti és a transzformált sorozat négyzetösszege egyenlő. Például egydimenziós esetre:

$$\sum_{i=0}^{k-1} y_i^2 = \mathbf{y}^T\mathbf{y} = (\mathbf{A}\mathbf{x})^T\mathbf{A}\mathbf{x} = \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{x}^T\mathbf{x} = \sum_{i=0}^{k-1} x_i^2,$$

ahol kihasználtuk az ortonormáltságot, hiszen $\mathbf{A}^T\mathbf{A} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$.

Szerencsés esetben az $\mathbf{y}$ egyes komponenseinek nagyságrendje különböző, tehát az egyes kvantálók különböző számú kvantálási szintet használnak. Ez a **bitallokáció** problémája. Tegyük fel, hogy a k hosszú blokk-kódolására M bitet szánunk úgy, hogy a j-edik kvantáló M_j bitet használhat, azaz kvantálási szintjeinek száma 2^{M_j}. A skalár kvantáló entrópiája a $H(f)$ differenciális entrópiától függ, amely normális eloszlás esetén $\frac{1}{2}\log(2\pi e\sigma^2)$. Az $\mathbf{X}$ lineáris transzformáltjának a komponensei nagy k blokkméret esetén a centrális határeloszlás tétel miatt közelítőleg normális eloszlásúak, tehát indokolt, hogy a j-edik kvantáló

$$M_j = c + \log\boldsymbol{\sigma}(Y_j) \tag{2.24}$$

bitet kapjon. Ekkor

$$\sum_{j=0}^{k-1} (c + \log\boldsymbol{\sigma}(Y_j)) = M,$$

ahonnan a c konstans kiszámolható:

$$c = \frac{M - \sum_{j=0}^{k-1} \log\boldsymbol{\sigma}(Y_j)}{k}.$$

A gyakorlatban három transzformáció terjedt el leginkább. A **diszkrét koszinusz transzformáció** (DCT) $k \times k$-as $\mathbf{A}$ mátrixának első sorában csupa $\frac{1}{\sqrt{k}}$ elem áll, míg az ez alatti elemek az

$$a_{i,j} = \sqrt{\frac{2}{k}}\cos\left(\frac{(2j-1)(i-1)\pi}{2k}\right)$$

képlettel számolhatók. A DCT a legkedveltebb transzformáció, felhasználja a JPEG és az MPEG álló- illetve mozgóképtömörítési szabvány is.

A **diszkrét szinusz transzformáció** (DST) $k \times k$-as mátrixának elemei:

$$a_{i,j} = \sqrt{\frac{2}{k+1}} \sin\left(\frac{\pi i j}{k+1}\right).$$

Az egyszerűbb **diszkrét Walsh–Hadamard-transzformáció** (DWHT) kisebb számításigényű, de általában nem biztosít olyan jó tömörítési hatásfokot, mint a DCT. A transzformáló mátrixot a következő rekurzióval kapjuk:

$$\mathbf{A}_{2^k} = \begin{pmatrix} \mathbf{A}_{2^{k-1}} & \mathbf{A}_{2^{k-1}} \\ \mathbf{A}_{2^{k-1}} & -\mathbf{A}_{2^{k-1}} \end{pmatrix}$$

A kiinduló mátrix pedig:

$$\mathbf{A}_1 = \begin{pmatrix} 1 \end{pmatrix}$$

Így például:

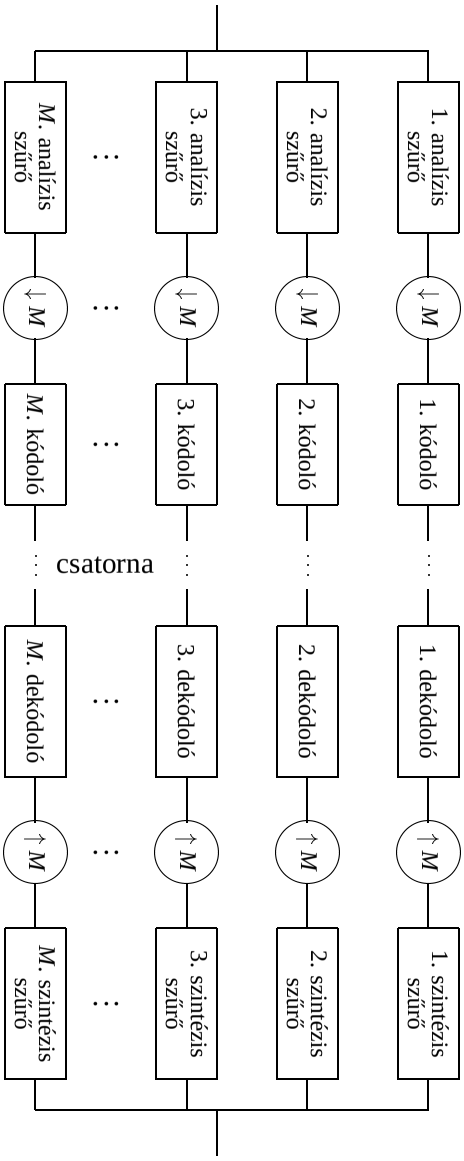
$$\mathbf{A}_2 = \begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_1 \\ \mathbf{A}_1 & -\mathbf{A}_1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

A mátrixok kielégítik az $\mathbf{A}_{2^k}\mathbf{A}_{2^k}^T = 2^k\mathbf{I}$ feltételt, és a transzformáció során ennek a normalizáltjával dolgozunk. A 8×8-as DWHT transzformáló mátrixa így a következő:

$$\mathbf{A} = \frac{1}{\sqrt{8}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}$$

A transzformációs kódolás leggyakrabban használt megvalósítása a **részsávos kódolás**. Alapötlete, hogy a jelet nem a teljes sávszélességében kódoljuk, hanem frekvenciasávonként.

A részsávos kódolás egy gyakorlati megvalósítása látható a 2.6. ábrán. A forrás kimenetét egy szűrőbankon visszük keresztül, amely a jelet az összetevő frekvenciasávokra bontja szét. A szűrők kimenetén kapott frekvenciasávok lehetnek átlapoltak és nem átlapoltak (a teljes sávot egyszeresen lefedők). Ezután mintavételezzük az egyes szűrők kimeneteit. A mintavételi törvény szerint az



2.6. ábra. Részsávos kódoló.

elemi sávszélességek kétszeresével kell ezt megtennünk a visszaállíthatóság érdekében. Tehát a minták számát csökkenthetjük a szűrők kimenetén, hiszen itt kisebb a sávszélesség, mint a szűrőbank bemenetén volt. Ezt decimálásnak vagy alulmintavételezésnek (downsampling) nevezzük. A decimálás mértéke a szűrő bemenetén illetve kimenetén lévő sávszélességek arányától függ. A decimálás után következik a kódolás. A kódolt jelet átküldjük a csatornán, majd annak túloldalán dekódoljuk. Felülmintavételezzük (upsampling), vagyis a minták közé megfelelő számú 0-t illesztünk, így állítva vissza az eredeti másodpercenkénti mintaszámot. Végül egy szűrőbankon vezetjük keresztül a jelet, amely elvégzi a fordított transzformációt, s így a kimeneteit összegezve kapjuk a végleges jelet.

Felmerülhet a kérdés, hogy miért jó ez az egész? Az emberi érzékszervek, akár a hang, akár a (mozgó)kép érzékelés terén frekvenciafüggőek. Ezt a tényt úgy használhatjuk ki, hogy az érzékelés szempontjából fontos frekvenciasávok pontosabb rekonstrukciójára törekszünk, míg a kevésbé fontos sávokban nagyobb torzítást engedünk meg. Másrészről az egyes szűrők kimenetén különböző a szórásnégyzet, és ennek megfelelően rendre különböző számú bitet allokálhatunk a kvantáláshoz a (2.24) egyenlet szerint, és ezzel lényeges tömörítést érhetünk el.

2.6. Prediktív kódolás (DPCM)

A prediktív kódolás alapelvét először egy speciális esetben vezetjük be, amikor a predikció az előző minta („meteorológus prediktor”). Ezt hívjuk **különbségi kódolás**nak.

A különbségi kódolás alkalmazása egy adatforrásra akkor előnyös, ha a szomszédos minták közti eltérés nem túl nagy. Például digitális képek esetén a szomszédos pixelek közötti eltérés viszonylag kicsi, hacsak nem vagyunk egy él közelében.

A következő példa rávilágít, hogy mit nyerhetünk a különbségi kódolással a memóriamentes kódoláshoz képest, vagyis amikor a mintákat egymástól függetlenül kódoljuk.

2.4. példa. Legyen egy 8 bites intenzitású digitalizált kép 8 egymást követő pixelének értéke:

$$147,\ 145,\ 141,\ 146,\ 149,\ 147,\ 143,\ 145.$$

Ha ezeket pontról pontra kódoljuk, egyenként 8 biten, akkor ehhez 64 bit szükséges. Vegyük azonban a kódolás előtt ezen intenzitások különbségét oly módon, hogy az első képpont értékét változatlanul hagyjuk, majd minden további pixel

esetén azt az értéket tároljuk, amelyet az előző képpont intenzitásához hozzáadva a saját intenzitását kapjuk. Esetünkben ez a következő lesz:

$$147,\ -2,\ -4,\ 5,\ 3,\ -2,\ -4,\ 2.$$

Világos, hogy az eredeti sorozat helyreállítható a második sorozatból. A kapott sorozatot egyszerűen vezessük át egy összegzőn.

Először 8 biten átküldjük a dekódernek, hogy a különbségeket hány biten fogjuk ábrázolni (ez függ a legnagyobb megjelenítendő értéktől), majd szintén 8 biten átküldjük az első (változatlan) adatot. Esetünkben a legnagyobb különbségi érték az 5, ezért 4 biten fogjuk átküldeni a differenciákat (1 előjelbit + 3 adatbit). Így összesen $8+8+7\cdot 4=44$ bitet használunk fel, ez 30 %-kal jobb tömörítési arányt jelent.

Sajnos veszteséges tömörítés esetén nem ilyen egyszerű a helyzet: nem igaz, hogy adott hibahatáron belül helyreállítható ily módon az eredeti sorozat. Legyen a forrás kimenete:

$$6.2,\ 9.7,\ 13.2,\ 5.9,\ 8,\ 7.4,\ 4.2,\ 1.8$$

Képezzük a különbségeket:

$$6.2,\ 3.5,\ 3.5,\ -7.3,\ 2.1,\ -0.6,\ -3.2,\ -2.4$$

A veszteséges tömörítéshez használjunk egy 7 szintű kvantálót a következő kimeneti szintekkel: $-6,\ -4,\ -2,\ 0,\ 2,\ 4,\ 6$. A kvantált értékek:

$$6,\ 4,\ 4,\ -6,\ 2,\ 0,\ -4,\ -2$$

Ha most megpróbáljuk az eredeti sorozatot helyreállítani a szokásos módon, a

$$6,\ 10,\ 14,\ 8,\ 10,\ 10,\ 6,\ 4$$

értékeket kapjuk. Az eredeti és a rekonstruált sorozat között az eltérések:

$$0.2,\ -0.3,\ -0.8,\ -2.1,\ -2,\ -2.6$$

Látható, hogy egyre hosszabb sorozatokat vizsgálva a hiba folyamatosan nőhet.

Vizsgáljuk meg ezt az észrevételt általános esetben! Legyen a bemeneti sorozatunk $\{x_n\}$. A különbségi sorozat $\{d_n\}$, ahol $d_n = x_n - x_{n-1}$. A különbségi sorozatot kvantáljuk, s így kapjuk $\{\widehat{d_n}\}$-ot:

$$\widehat{d_n} = Q(d_n) = d_n + q_n$$

ahol q_n a kvantálási hiba. A vevő oldalán a rekonstruált sorozatot, $\{\widehat{x}_n\}$-ot úgy kapjuk, hogy a kvantált különbséget hozzáadjuk az előző értékhez:

$$\widehat{x}_n = \widehat{x}_{n-1} + \widehat{d}_n$$

Tételezzük fel, hogy az adó és a vevő ugyanarról az értékről indul, tehát $x_0 = \widehat{x}_0$. A kvantálás és a rekonstrukció első néhány lépése:

$$\begin{aligned}
d_1 &= x_1 - x_0 \\
\widehat{d}_1 &= Q(d_1) = d_1 + q_1 \\
\widehat{x}_1 &= \widehat{x}_0 + \widehat{d}_1 = x_0 + d_1 + q_1 = x_1 + q_1 \\
d_2 &= x_2 - x_1 \\
\widehat{d}_2 &= Q(d_2) = d_2 + q_2 \\
\widehat{x}_2 &= \widehat{x}_1 + \widehat{d}_2 = x_1 + q_1 + d_2 + q_2 = x_2 + q_1 + q_2.
\end{aligned}$$

Tovább folytatva, az n-edik lépés után:

$$\widehat{x}_n = x_n + \sum_{k=1}^{n} q_k.$$

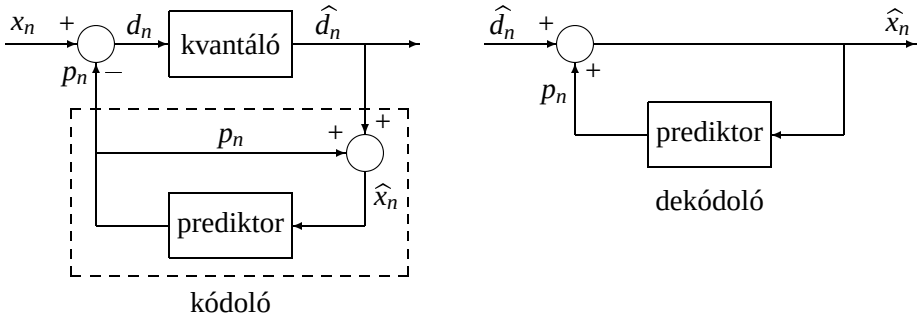
Tehát a kvantálási hiba felhalmozódik.

Vegyük észre, hogy a kódoló és a dekódoló különböző adatokon hajt végre műveleteket! A kódoló az eredeti sorozat különbségeit generálja, míg a dekódoló a kvantált különbségekkel dolgozik. Áthidalhatjuk ezt a problémát, ha az adót és a vevőt rákényszerítjük, hogy azonos adatokon dolgozzanak. A vevő az eredeti $\{x_n\}$ sorozatról csak a rekonstruált $\{\widehat{x}_n\}$ sorozatból szerezhet információt. Ez utóbbi viszont az adónak is rendelkezésére áll, így a különbségképzés műveletét a következőképpen módosíthatjuk:

$$d_n = x_n - \widehat{x}_{n-1}.$$

Ezt felhasználva az előző lépéseink így módosulnak:

$$\begin{aligned}
d_1 &= x_1 - x_0 \\
\widehat{d}_1 &= Q(d_1) = d_1 + q_1 \\
\widehat{x}_1 &= \widehat{x}_0 + \widehat{d}_1 = x_0 + d_1 + q_1 = x_1 + q_1 \\
d_2 &= x_2 - \widehat{x}_1 \\
\widehat{d}_2 &= Q(d_2) = d_2 + q_2 \\
\widehat{x}_2 &= \widehat{x}_1 + \widehat{d}_2 = \widehat{x}_1 + d_2 + q_2 = x_2 + q_2.
\end{aligned}$$



2.7. ábra. DPCM.

Általánosítva:

$$\widehat{x}_n = x_n + q_n,$$

tehát a kvantálási zaj nem akkumulálódik.

Az a célunk, hogy minél kisebb különbségi értékeket kapjunk. Ehhez x_n értékét nem csak $\widehat{x}_{n-1}$ segítségével becsülhetjük, hanem a rekonstruált sorozat előző értékeinek függvényével is. Az ezt megvalósító eszköz a **prediktor**:

$$p_n = f(\widehat{x}_{n-1}, \widehat{x}_{n-2}, \ldots, \widehat{x}_0).$$

Itt a különbségképzés a

$$d_n = x_n - p_n$$

kifejezéssel történik. Az eredő torzítás ebben az esetben sem halmozódik. Az eddig elmondottakat megvalósító rendszer a 2.7. ábrán látható, az eljárás neve **különbségi impulzuskód moduláció** (differential pulse code modulation, **DPCM**). Az eljárást a Bell Laboratóriumban dolgozták ki a '40-es évek végén, és főként a beszédkódolásban terjedt el, így széleskörűen használják a telekommunikációban.

A különbségi kódoló rendszerek, mint a DPCM, azzal érik el céljukat, vagyis a tömörítést, hogy csökkentik a bemeneti sorozat szórását és dinamikatartományát. A szórás csökkentése attól függ, hogy a prediktor mennyire jól tudja becsülni a következő szimbólumot az előzőleg rekonstruáltakból. A 2.7. szakaszban vizsgáljuk a becslés problémakörét.

A bemeneti jelek tulajdonsága változhat az időben. Ehhez tud alkalmazkodni az adaptív DPCM rendszer. Az alkalmazkodás történhet a kódoló bemenetére érkező jel (x_n) alapján, ekkor előre adaptív módszerről beszélünk, vagy a kimenet ($\widehat{x}_n$) alapján, ez a hátra adaptív módszer. Előre adaptív esetben, mivel a dekódernek nem áll rendelkezésére a kódoló bemeneti jele, ezért a rendszer módosított

paramétereit is át kell vinni. Hátra adaptív esetben ilyen probléma nem merül fel, hiszen a kódoló kimenetét megkapja a dekóder.

A kvantálót és a prediktort egyaránt tervezhetjük adaptívra (**ADPCM**), ez utóbbiról a 2.7. szakaszban szólunk. Előre adaptív kvantáló esetén a bemenetet blokkokra bontjuk, minden lépésben kiszámoljuk az aktuális blokkra optimális kvantáló paramétereit, és átküldjük a dekódernek kiegészítő információként.

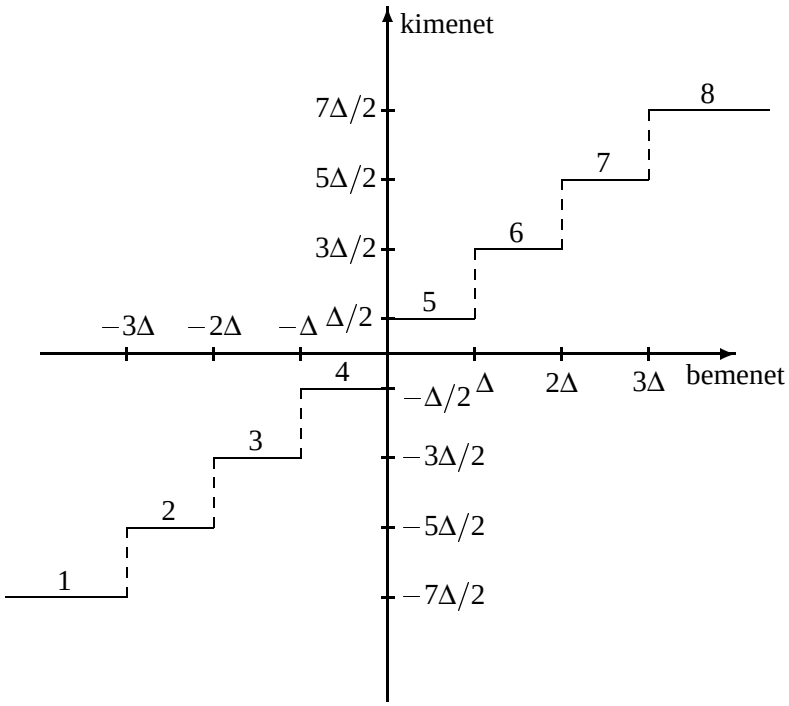
A gyakorlatban inkább a **Jayant-kvantáló** néven ismert hátra adaptív módszert használják. Itt minden egyes kvantálási intervallumhoz egy szorzótényezőt rendelünk, amely a kvantáló belső (origóhoz közeli) tartományainál 1-nél kisebb, míg a külső részeken 1-nél nagyobb. A tényezők az origóra szimmetrikusan helyezkednek el. Annak függvényében, hogy az aktuálisan kvantált érték melyik intervallumba esik, ezen intervallum szorzótényezőjével korrigáljuk a lépésköz értékét. Ha a kvantált érték kicsi, akkor finomítjuk a lépésközt, míg nagy értékek esetén nagyobb lépésközt fogunk alkalmazni.

2.5. példa. A 2.8. ábra egy 8 szintű kvantálót ábrázol. Legyenek az egyes intervallumok szorzótényezői: $M_1 = M_8 = 1.2$, $M_2 = M_7 = 1$, $M_3 = M_6 = 0.9$, $M_4 = M_5 = 0.8$, a kezdeti lépésköz pedig $\Delta_0 = 0.5$. A kvantálandó sorozat: 0.1, -0.2, 0.2, 0.1, 0.2, 0.5, 0.9, 1.5, 1.0, 0.9, ... Az első bemeneti adatot a kezdeti 0.5 lépésközzel az 5. szintre kvantáljuk, a kimenet értéke 0.25 lesz, a hiba 0.15. Így az új lépésköz $\Delta_1 = M_5\Delta_0 = 0.8 \cdot 0.5 = 0.4$ lesz. A következő adat a 4. intervallumba esik, most a lépésköz 0.4, tehát a kimeneten -0.2 jelenik meg, és a lépésköz új értéke $\Delta_2 = M_4\Delta_1 = 0.32$. Így folytatva az eljárást, az eredmény táblázatos formában a 2.9. ábrán látható.

Vegyük észre, hogyan alkalmazkodik a kvantáló az inputhoz. Kezdetben a bemeneti értékek kicsik, ezért a lépésköz folyamatosan csökken, ezzel egyre jobb közelítést biztosítva. Majd nagyobb bemeneti értékek következnek, így a lépésköz is növekszik. Megfigyelhetjük, hogy a hiba értéke viszonylag nagy az átmeneti szakaszban.

A kvantálási lépésközt a konkrét megvalósítások természetesen véges pontossággal ábrázolják, ezért el kell kerülnünk azt a szituációt, hogy a lépésköz folyamatos csökkentésével, az nullává váljon. Be kell vezetnünk egy $\Delta_{\min}$ értéket, amelynél nem választjuk kisebbnek a lépésközt. Hasonló okok miatt egy $\Delta_{\max}$ is szükséges.

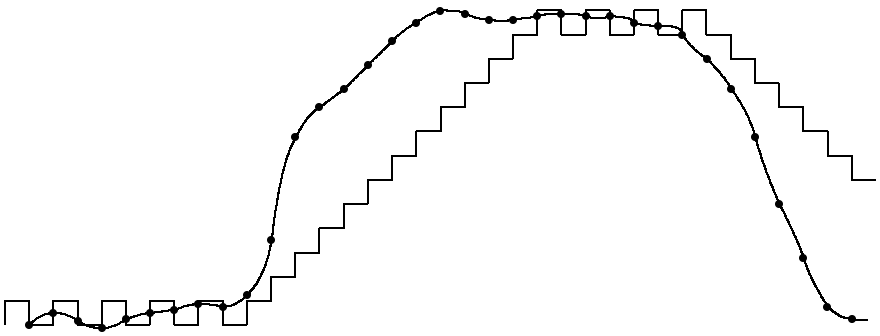
A DPCM egyszerűbb formája, a **delta moduláció** (DM). Ez egy 1 bites (2 szintű) kvantálóval rendelkező DPCM. A 2 szintű kvantálóval csak $\pm\Delta$ kimeneti értékeket állíthatunk elő, vagyis (egy folytonos jel mintavételezésével adódó) két minta közötti eltérést csak ezzel reprezentálhatjuk. Amennyiben egy adott bemeneti sorozatban a minták közötti különbség nagyon eltér Δ-tól, a kimeneten egy állandó torzítás jelentkezik. Ezt gyakoribb mintavételezéssel ellensúlyozhatjuk. A



2.8. ábra. Jayant-kvantáló 8 kvantálási szinttel.

n	Δ_n	bemenet	szint száma	szint értéke	hiba	új lépésköz
0	0.5	0.1	5	0.25	0.15	$\Delta_1 = M_5\Delta_0$
1	0.4	−0.2	4	−0.2	0.0	$\Delta_2 = M_4\Delta_1$
2	0.32	0.2	5	0.16	0.04	$\Delta_3 = M_5\Delta_2$
3	0.256	0.1	5	0.128	0.028	$\Delta_4 = M_5\Delta_3$
4	0.2048	−0.3	3	−0.3072	−0.0072	$\Delta_5 = M_3\Delta_4$
5	0.1843	0.1	5	0.0922	−0.0078	$\Delta_6 = M_5\Delta_5$
6	0.1475	0.2	6	0.2212	0.0212	$\Delta_7 = M_6\Delta_6$
7	0.1328	0.5	8	0.4646	−0.0354	$\Delta_8 = M_8\Delta_7$
8	0.1594	0.9	8	0.5578	−0.3422	$\Delta_9 = M_8\Delta_8$
9	0.1913	1.5	8	0.6696	−0.8304	$\Delta_{10} = M_8\Delta_9$
10	0.2296	1.0	8	0.8036	0.1964	$\Delta_{11} = M_8\Delta_{10}$
11	0.2755	0.9	8	0.9643	0.0643	$\Delta_{12} = M_8\Delta_{11}$

2.9. ábra. A 2.5. példa végeredménye.



2.10. ábra. Lineáris delta moduláció.

DM rendszerekben a legnagyobb előforduló frekvenciának nem csak a kétszeresével, hanem akár a százszorosával is mintavételeznek a torzítás alacsony szinten tartása végett (például jó minőségű A/D átalakítókban, „1 bites átalakítók”).

A fix lépésközű kvantálót tartalmazó DM rendszereket **lineáris delta modulátor**nak nevezzük. A 2.10. ábrán megfigyelhetjük, hogy torzítás két okból jelentkezik. Azokon a részeken, ahol a bemenet hozzávetőleg konstans, a kimenet oszcillál Δ-val (granular regions). Ezt a hibát Δ csökkentésével kompenzálhatjuk. Azokban a tartományokban, ahol a bemenet túl gyorsan növekszik vagy csökken, a kimenet nem tud lépést tartani (slope overload regions), ezért ezen a Δ lépésköz növelésével segíthetünk. Látható, hogy a kétféle torzítás egyidejű javítása éppen ellentétes feltételeket támaszt Δ-ra vonatkozóan, ezért fix lépésközzel nem oldható meg.

Ezen segít a lépésköz adaptív megválasztása. A közel konstans tartományokban kis lépésközt választunk, míg gyors változások esetén növeljük Δ-t. A bemenet lokális tulajdonságához alkalmazkodó rendszerek közül az egyik legkedveltebb a konstans faktorral alkalmazkodó delta modulátor (constans factor adaptive delta modulation, CFDM). A legfontosabb feladat annak megállapítása, hogy éppen milyen jellegű tartományban vagyunk. A közel állandó részeken a kvantáló kimenete majdnem minden mintánál előjelet vált, míg a gyorsan változó intervallumokban a kimenet előjele állandó. A legegyszerűbb esetben csak a megelőző mintáig tekintünk vissza a tartomány jellegének eldöntésére. Jelöljük s_n-nel a delta modulátor n-edik időegységbeli „lépését” és Δ_n-nel az itt érvényes lépésközt ($s_n = \pm\Delta_n$). Ekkor az $n+1$-edik időegységbeli lépésközt a következőképpen kapjuk:

$$\Delta_{n+1} = \begin{cases} M_1\Delta_n, & \text{ha } \operatorname{sgn} s_n = \operatorname{sgn} s_{n-1} \\ M_2\Delta_n, & \text{ha } \operatorname{sgn} s_n \neq \operatorname{sgn} s_{n-1} \end{cases}$$

ahol $1 < M_1 = \frac{1}{M_2} < 2$. A memória növelésével, vagyis például 2 mintára visszatekintve tovább csökkenthető a torzítás:

$$\Delta_{n+1} = \begin{cases} M_1\Delta_n, & \text{ha } \operatorname{sgn} s_{n-2} \neq \operatorname{sgn} s_{n-1} \neq \operatorname{sgn} s_n \\ M_2\Delta_n, & \text{ha } \operatorname{sgn} s_{n-2} = \operatorname{sgn} s_{n-1} \neq \operatorname{sgn} s_n \\ M_3\Delta_n, & \text{ha } \operatorname{sgn} s_{n-2} \neq \operatorname{sgn} s_{n-1} = \operatorname{sgn} s_n \\ M_4\Delta_n, & \text{ha } \operatorname{sgn} s_{n-2} = \operatorname{sgn} s_{n-1} = \operatorname{sgn} s_n \end{cases}$$

ahol pl. $M_1 = 0.4 < M_2 = 0.9 < M_3 = 1.5 < M_4 = 2.0$.

A beszédkódolásban kívánatos, lassabb alkalmazkodást tesz lehetővé a folytonosan változó meredekségű delta moduláció (continuously variable slope delta modulation, CVSD). Ez csökkenti a közel konstans esetben elkövetett hibát, viszont növeli a gyors változások esetén bekövetkező hibát. Az adaptív lépésköz számítására szolgáló képlet:

$$\Delta_n = \beta\Delta_{n-1} + \alpha_n\Delta_0,$$

ahol β egy 1-nél alig kisebb konstans, α_n értéke pedig 1, ha a kvantáló legutóbbi K darab kimeneti értékéből J darabnak azonos volt az előjele, egyébként 0. Tehát egy K hosszú ablakon keresztül figyeljük a bemenetet, és ebből következtetünk annak lokális viselkedésére. Jellemző értékek: $J = K = 3$.

2.7. Lineáris becslés

A prediktív kódoló rendszerek, mint például a DPCM, a különbségi sorozat második momentumának és dinamikatartományának csökkentésével érik el céljukat. A szórás csökkentése azon múlik, hogy a prediktor mennyire jól tudja megjósolni a következő szimbólumot az előzőleg rekonstruált értékek alapján.

Legyen σ_d^2 a különbségi sorozat második momentuma:

$$\sigma_d^2 = \mathbf{E}(X_n - p_n)^2,$$

ahol

$$p_n = f(\widehat{x}_{n-1}, \widehat{x}_{n-2}, \ldots, \widehat{x}_0)$$

a prediktor kimenete. Feladatunk azon $f(\cdot)$ megtalálása, amely minimalizálja σ_d^2-et. Ez azonban nem ilyen egyszerű. Egyrészt $\widehat{x}_n = X_n + q_n$, másrészt q_n függ d_n szórásától, így $f(\cdot)$ megválasztása befolyásolja σ_d^2-et, ami pedig hatással van a rekonstruált $\widehat{x}_n$ sorozatra, ettől pedig függ $f(\cdot)$ megválasztása. A kör bezárult. Ez a **csatolás** (coupling) még egyszerű források esetén is rendkívül bonyolulttá teszi az explicit megoldást. Mivel a gyakorlati források közel sem ideálisak, a probléma számításigénye kézbentarthatatlanná válik a legtöbb alkalmazásban.

Elkerülhetjük ezt a nehézséget finom kvantálás feltételezésével, vagyis feltesszük, hogy a kvantáló lépésköze olyan kicsi, hogy $\widehat{x}_n$ helyett írhatunk X_n-et, ezért

$$p_n = f(X_{n-1}, X_{n-2}, \ldots, X_0).$$

Tehát, ha megtaláltuk $f(\cdot)$-et, akkor alkalmazhatjuk a rekonstruált $\widehat{x}_n$ értékekre, és így közelítőleg megkapjuk p_n-et. A σ_d^2-et minimalizáló függvény a regressziós függvény, vagyis az $\mathbf{E}(X_n \mid X_{n-1}, X_{n-2}, \ldots, X_0)$ feltételes várható érték.

Sajnos a feltételes várható érték megtalálásához szükségünk lenne az n-edrendű feltételes valószínűségekre, amelyek általában nem állnak rendelkezésünkre.

A legjobb megoldás reménytelennek tűnő keresése helyett korlátozzuk vizsgálatainkat a **lineáris prediktorfüggvény**ekre, vagyis

$$p_n := \sum_{i=1}^{N} a_i \widehat{x}_{n-i}.$$

Ezek ugyanis könnyen kiszámíthatóak és tárolhatóak, valamint normális eloszlás esetén lineáris lesz a regressziós függvény. N-et a prediktor rendjének nevezzük. Élve a finom kvantálás feltételezésével, írhatjuk:

$$\sigma_d^2 = \mathbf{E}\left(X_n - \sum_{i=1}^{N} a_i X_{n-i}\right)^2, \tag{2.25}$$

ahol a feladatunk a σ_d^2-et minimalizáló $\{a_i\}$ sorozat megtalálása.

σ_d^2 N változós függvény lokális szélsőértéke létezésének szükséges feltétele, hogy a parciális deriváltak 0-k legyenek. Vegyük (2.25) parciális deriváltjait rendre $a_j,\ j = 1, \ldots, N$ szerint:

$$\begin{aligned}
\frac{\partial}{\partial a_j}\sigma_d^2 &= \mathbf{E}\left(\frac{\partial}{\partial a_j}\left(X_n - \sum_{i=1}^{N} a_i X_{n-i}\right)^2\right) = \\
&= \mathbf{E}\left(-2X_{n-j}\left(X_n - \sum_{i=1}^{N} a_i X_{n-i}\right)\right) = \\
&= -2\mathbf{E}\left(\left(X_n - \sum_{i=1}^{N} a_i X_{n-i}\right) X_{n-j}\right) = 0,
\end{aligned}$$

ahol kihasználtuk a várhatóérték-képzés linearitását. Így $j = 1, \ldots, N$-re N egyenletet kapunk, amelyek N ismeretlent tartalmaznak. Ezeket átrendezve kapjuk:

$$\sum_{i=1}^{N} a_i R(i-j) = R(j), \qquad j = 1, \ldots, N,$$

ahol $R(k)$ a gyengén stacionárius X_n kovarianciafüggvénye:

$$R(k) = \mathbf{E}(X_n X_{n+k}).$$

Írjuk fel a most kapott egyenleteinket a könnyebben kezelhető mátrixos alakban:

$$\mathbf{Ra} = \mathbf{p},$$

ahol

$$\mathbf{R} = \begin{pmatrix} R(0) & R(1) & R(2) & \cdots & R(N-1) \\ R(1) & R(0) & R(1) & \cdots & R(N-2) \\ R(2) & R(1) & R(0) & \cdots & R(N-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R(N-1) & R(N-2) & R(N-3) & \cdots & R(0) \end{pmatrix}$$

$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_N \end{pmatrix} \qquad \mathbf{p} = \begin{pmatrix} R(1) \\ R(2) \\ R(3) \\ \vdots \\ R(N) \end{pmatrix}$$

$\mathbf{R}$ kifejezésében felhasználtuk az $R(-k) = R(k)$ tulajdonságot. Ez az egyenletrendszer az ún. Wiener–Hopf-egyenletrendszer diszkrét alakja. Ha ismerjük az $\{R(k)\}$, $k = 0, 1, \ldots, N$ kovariancia értékeket, akkor meghatározhatjuk a prediktoregyütthatókat:

$$\mathbf{a} = \mathbf{R}^{-1}\mathbf{p}, \tag{2.26}$$

amennyiben $\mathbf{R}^{-1}$ létezik.

A prediktoregyütthatók meghatározására szolgáló (2.26) egyenletet a stacionaritás feltételezésével kaptuk. Ez azonban a gyakorlatban előforduló források esetén nem jogos. A stacionaritás legfeljebb csak lokálisan, egy bizonyos jelhosszon belül igaz. A problémát úgy oldhatjuk meg, ha a prediktort adaptívvá tesszük a jel lokális viselkedéséhez. Ez történhet előre adaptív (a kódoló bemeneti jele alapján) illetve hátra adaptív módszerrel (a kódoló kimenete alapján).

Előre adaptív esetben a bemenetet blokkokra osztjuk. (Beszédkódolás esetén a tipikus blokkhossz 16 ms, ami 8000 minta/másodperc esetén 128 mintát jelent, míg képtömörítés során általában 8×8 pixeles blokkokat használnak.) Ezután kiszámítjuk az empirikus kovariancia együtthatókat minden egyes blokkra, amelyekből a (2.26) segítségével kapjuk a prediktoregyütthatókat. A kovariancia együtthatók kiszámításakor azzal a feltételezéssel élünk, hogy a minták értéke a

blokkon kívül 0. Ezért az l-edik M hosszú blokkra az alábbi közelítést alkalmazzuk:

$$R^{(l)}(k) = \frac{1}{M-|k|} \sum_{i=(l-1)M+1}^{lM-|k|} X_i X_{i+|k|}$$

$\forall k$-ra. ($R^{(l)}(k) = R^{(l)}(-k)$)

Az előre adaptív módszer megköveteli, hogy puffereljük a bemenetet, ezzel késleltetést viszünk a rendszerbe, ami például beszédkódolás esetén nem szerencsés. Ugyanis egyetlen kódoló–dekódoló pár esetén még elviselhető a késleltetés mértéke, azonban egy telefonkapcsolatban számos DPCM kódoló és dekódoló vehet részt (pl. a nagy távolság miatt), s ekkor az eredő késleltetés már jelentős lehet. Sőt, itt is, mint előre adaptív kvantálás esetén, kiegészítő információkat kell átvinni, hiszen a dekódoló nem ismeri a kódoló bemeneti jelét.

Hátra adaptív esetben a prediktor alkalmazkodásához a dekóder számára is rendelkezésre álló jelet, a kódoló kimenetét használjuk. Először az egyszerűség kedvéért vegyünk egy elsőrendű prediktort. Ekkor a jel valódi és becsült értéke közötti eltérés négyzete az n-edik időpillanatban:

$$d_n^2 = (X_n - a_1 \widehat{x}_{n-1})^2.$$

Ezt az a_1-ben másodfokú kifejezést (parabolát) kell minimalizálnunk. Gondoljuk meg, hogy ha a minimumhelytől pozitív irányban helyezkedik el a_1 előző értéke, akkor ott a derivált pozitív, míg az ellenkező irányban negatív. a_1 új értékét közelíthetjük úgy, hogy a régi értékből levonjuk a derivált konstansszorosát (gradiens módszer):

$$a_1^{(n+1)} = a_1^{(n)} - \alpha \frac{\partial d_n^2}{\partial a_1}, \tag{2.27}$$

$$\frac{\partial d_n^2}{\partial a_1} = -2(X_n - a_1 \widehat{x}_{n-1})\widehat{x}_{n-1} = -2 d_n \widehat{x}_{n-1}. \tag{2.28}$$

Helyettesítsük (2.28)-at (2.27)-be:

$$a_1^{(n+1)} = a_1^{(n)} + \alpha^* d_n \widehat{x}_{n-1},$$

ahol $\alpha^* = 2\alpha$. Mivel d_n-et csak a kódoló ismeri, használjuk helyette $\widehat{d}_n = \widehat{x}_n - a_1^{(n)} \widehat{x}_{n-1}$-et:

$$a_1^{(n+1)} = a_1^{(n)} + \alpha^* \widehat{d}_n \widehat{x}_{n-1}.$$

Terjesszük ki ezt az egyenletet az N-edrendű prediktor esetére! Az eltérés hibanégyzete ekkor:

$$d_n^2 = \left(X_n - \sum_{i=1}^{N} a_i \widehat{x}_{n-i} \right)^2.$$

Ennek a_j szerinti parciális deriválásával kapjuk a j-edik prediktoregyüttható számítására szolgáló kifejezést:

$$a_j^{(n+1)} = a_j^{(n)} + \alpha^* \widehat{d_n} \widehat{x}_{n-j}.$$

Vektoros formában:

$$\mathbf{A}^{(n+1)} = \mathbf{A}^{(n)} + \alpha^* \widehat{d_n} \widehat{\mathbf{X}}_{n-1}$$

ahol

$$\widehat{\mathbf{X}}_n = \begin{pmatrix} \widehat{x}_n \\ \widehat{x}_{n-1} \\ \vdots \\ \widehat{x}_{n-N+1} \end{pmatrix},$$

és

$$\widehat{d_n} = \widehat{x}_n - \sum_{i=1}^{N} a_i \widehat{x}_{n-i}.$$

Az ismertetett eljárás a legkisebb átlagos négyzetek (Least Mean Square) módszerének rekurzív változata.

2.8. Beszédtömörítés

A beszédtömörítő eljárások területén erőteljes fejlődés volt megfigyelhető az elmúlt évtizedekben. A minél alacsonyabb bitsebesség mellett minél jobb minőségű hang átvitelének igénye főleg a nyilvános célú távközlés részéről fogalmazódott meg. Kezdetben az eddigiekben megismert általános célú eljárásokat alkalmazták.

Az ismertetésre kerülő megoldásokat a nyilvános távközlőhálózatokban használt **impulzuskód moduláció**hoz (Pulse Code Modulation, PCM) hasonlítjuk. Határozzuk meg ennek bitsebességigényét! A „telefon-minőség” viszonylag kis sávszélességgel is beéri. Az analóg 3.4 kHz sávszélességű beszédjelből (némi rátartással) másodpercenként 8000 mintát veszünk, és 8 bittel kvantáljuk, így $8000 \cdot 8 = 64$ kbps-ot kapunk. Az 1972-ben megjelent **G.711**-es távközlési szabvány logaritmikus PCM kódolást használ, azaz kompanderes kvantálót (mint erről a 2.2. szakaszban szó volt: μ-law, A-law).

A **beszédtömörítés**sel szemben támasztott megnövekedett kommunikációs igény elkerülhetetlenné tette egy új eljárás kifejlesztését, amelynek kisebb a bitsebességigénye, de emellett jó minőségű, jól érthető beszédátvitelt tesz lehetővé. A kívánalmaknak megfelel az 1984-es **G.721** szabvány, amely 32 kbps-os átviteli sebességével megduplázta a kiépített vonalakon folytatható beszélgetések számát. A G.721 adaptív különbségi kódoláson (ADPCM) alapul. Kihasználja az emberi

beszéd különböző időpontokban megfigyelt mintái közötti hosszú- és rövidtávú korrelációt (2–20 ms). A kvantálási lépcsők méretét az előző minták alapján változtatja. A kódoló késleltetése kisebb mint 2 ms, és számos központon keresztülhaladó jel esetén is megfelelő minőséget biztosít.

A fenti eljárások finomításával jobb tömörítési arány érhető el. Ezek a módszerek az emberi hangképzés modelljét használják fel: A tüdőből kiáramló levegő megrezegteti a hangszálakat, majd a gége, garat, száj- és orrüreg illetve a nyelv által képezett változtatható akadályon keresztülhaladva alakul ki végül a hang. A hangszálak után lévő hangképző szerveket összefoglalóan hangképző útnak (vocal tract) nevezzük. A hangszalagok képzik a zöngét (alaphangot), majd a hangképző út modulálja azt. A mesterséges hangképzéshez tehát elő kell állítanunk egy gerjesztő jelet, és ezt kell modulálnunk a hangképző út szerepének megfelelően.

A kódoló a beszédet szegmensekre osztja, és minden szegmensre meghatározza a gerjesztő jelet, és a hangképző utat modellező szűrő paramétereit. A gerjesztő jel olyan, hogy ezen beállított szűrő kimenete leginkább hasonlítson a tömörítendő beszédre. A **szintetizált** beszédtömörítők esetén a gerjesztő jelet nem küldjük át, csak néhány paraméterét, és ebből a dekódoló előállítja a megfelelő gerjesztő jelet, majd ezzel hajtja meg a saját szűrőbankját, amelynek paramétereit a kapott adatoknak megfelelően állította be.

A tömörítendő beszéd minden szegmensét egy sávszűrő bankon vezetjük keresztül, amelyet analízis szűrőnek nevezünk. Az analízis szűrő kimenetének energiáját meghatározott időközönként (általában másodpercenként 50-szer) mintavételezzük, és átküldjük a dekódolónak. (Az emberi beszéd kb. 20 ms-os egységen belül stacionáriusnak tekinthető.) Digitális esetben az energiát közelíthetjük a szűrő kimeneti jelének átlagos négyzetösszegével, míg analóg esetben a jel burkológörbéjének mintavételezésével érhetünk célt. Az energia meghatározásával egyidejűleg azt is el kell dönteni, hogy az adott beszédszegmens zöngés vagy zöngétlen hangot tartalmaz-e. A zöngés hangok álperiodikus viselkedést mutatnak, az alapharmonikus frekvenciáját hangmagasságnak hívjuk. A kódoló a becsült hangmagasság értékét szintén átküldi a dekódolónak. A zöngétlen hangok zajszerű szerkezetet mutatnak. A szintetizált beszédtömörítést tekinthetjük egy olyan vektorkvantálásnak, ahol a kvantálást az egyes szegmensek Fourier-transzformáltjainak a terében végezzük.

A dekódolóban a hangképző szerveket sávszűrő bankkal modellezzük, ez a szintézis szűrő, amely megegyezik az analízis szűrővel. Annak megfelelően, hogy zöngés vagy zöngétlen hangot kell-e visszaállítani, a szintézis szűrő bemeneteként egy, a hangmagasságnak megfelelő periodikus jelgenerátort vagy egy álzajgenerátort használunk. A jel amplitúdóját a becsült energiának megfelelően állítjuk be.

A hangképző út egy változtatható méretű rezonátorüregként fogható fel, ezért számos rezonanciafrekvencia lehetséges, de nem az összes frekvenciaösszetevő egyformán fontos. A rezonanciafrekvenciákat formánsoknak nevezzük. A formáns frekvenciája hangonként eltérő, de férfiak esetén 200–800 Hz, míg nők esetén 250–1000 Hz közötti tartományba esik. Ezt a jellemzőt használja fel a **formáns beszédkódoló**, amely meghatározza a formánsok értékének közelítését (általában 4 formáns megkülönböztetése elegendő), valamint ezek sávszélességét, majd a vevő oldalon a gerjesztő jelet hangolható szűrőkön vezetik keresztül, amelyeket a formánsok frekvenciájára és sávszélességére hangolnak.

Ennyi felvezetés után nézzük meg a konkrét alkalmazásokat!

Lineáris prediktív kódolás (LPC) esetén egy lineáris szűrőt használunk, mely az előző minták alapján próbál optimális becslést adni az aktuális mintára úgy, hogy a négyzetes torzítás minimális legyen:

$$x_n = \sum_{i=1}^{M} a_i x_{n-i} + G\varepsilon_n, \tag{2.29}$$

ahol G a szűrő energiája (gain), ε_n a kvantálandó jel. Az a_i együtthatók optimális meghatározása a 2.7. szakaszban ismertetett módon történhet.

Az LPC-nek két változata van. Az egyszerűbb, nem-szintetizált esetben (ez még „hagyományos” eljárásnak tekinthető abban az értelemben, hogy nem használja fel az emberi beszéd modelljét) közvetlenül a különbségi jel mintavételezett értékét (ε_n) küldjük át a vevőnek. Ekkor (2.29)-ben x_n helyén a 2.6. szakaszban megismert módon a reprodukált $\widehat{x}_n$ értékek állnak.

A fejlettebb módszer a vevőoldalon szintetizált beszédet állít elő, vagyis csak a szűrő $\{a_i\}$ paramétereit és a G energiát kell átküldenünk, és ebből generálja a vevő a saját szűrőjével a beszédet úgy, hogy az ε_n sorozatot helyettesíti egy mesterséges gerjesztő jelsorozattal.

A kódolónak a következő információkat kell átküldenie: zöngés vagy zöngétlen-e a hang, a hangmagasság értéke és a hangképző utat modellező szűrő paraméterei. Az **LPC–10** algoritmus esetén ez a következőképpen néz ki: A zöngés / zöngétlen információ 1 bit, a hangmagasságot 60 lehetséges értékre kvantálják egy logaritmikus karakterisztikájú kompanderes kvantáló segítségével (6 bit), és zöngés esetben 10-edrendű, míg zöngétlen esetben 4-edrendű prediktív szűrőt alkalmaznak, ez 11 (10 a megfelelő együtthatók miatt és 1 az energia számára) illetve 5 előre rögzített adatot jelent. A szűrő nagyon érzékeny az 1-hez közeli együtthatók hibájára. Mivel az első néhány együttható általában 1-hez közeli, ezért a szabvány nemegyenletes kvantálást ír elő a_1 és a_2 számára. Ezt az alábbi módon oldják meg:

$$g_i = \frac{1+a_i}{1-a_i},$$

ezután g_1 és g_2 értékét egy 5 bites egyenletes kvantálóval kvantálják, hasonlóan a_3, a_4-et is. $a_5, \ldots, a_8$-at 4 bites, a_9-et 3 bites, míg a_{10}-et 2 bites egyenletes kvantálóval kvantálják. A zöngétlen esetben fennmaradó 21 bitet hibavédelemre használják. A G energia értékét a minták négyzetösszegéből gyökvonással határozzák meg, és 5 bites, logaritmikus karakterisztikájú kompanderes kvantálóval kvantálják. A szinkronizációhoz szükséges 1 további bittel együtt ez összesen 54 bitet tesz ki. Egy szegmens hossza 22.5 ms, tehát az LPC–10 algoritmus 2.4 kbps átvitelét teszi szükségessé.

A dekódoló zöngés szegmens esetén a szűrő gerjesztőjeleként egy előre tárolt hullámformát használ. A hullámforma 40 minta hosszúságú (a mintavételezés 8 kHz-cel történik), ezért a hangmagasságtól függően csonkolja vagy nullákkal tölti ki. Ha a szegmens zöngétlen, akkor a szűrőt egy álvéletlen generátor jelével hajtja meg (fehér Gauss-zaj). Az LPC–10 kódoló érthető, de nem túl jó minőséget biztosít 2.4 kbps bitsebességen. Az, hogy csak kétféle gerjesztőjelet alkalmaz a szűrő bemeneteként, gépi jellegű hangot eredményez. Ez különösen zajos környezet esetén jelent problémát, ugyanis a kódoló a környezeti zaj miatt a zöngés szegmenseket is zöngétleneknek fogja nyilvánítani.

A természetesen hangzó beszéd előállításának egyik legfontosabb összetevője a gerjesztőjel, ugyanis az emberi fül különösen érzékeny a hangmagasság hibájára. Az LPC gyenge pontját küszöbölik ki a **szinuszos kódoló**k. Ezek gerjesztőjelként szinuszos összetevőket használnak. Egy elemi összetevőt három paraméter határoz meg: az amplitúdó, a frekvencia és a fázis. Ezek közül, ha a jelet egy lineáris szűrőn vezetjük keresztül, a frekvencia nem változik. Mivel a hangképző út modellezésére használt szintézis szűrő lineáris, ezért ezen egy szinuszos jelet átvezetve csak az amplitúdó és a fázis változik. Megállapíthatjuk, hogy a gerjesztő jel leírásához szükséges paraméterek száma megegyezik a szintetizált beszédet reprezentáló paraméterek számával. Így ahelyett, hogy külön-külön kiszámítanánk és átvinnénk a gerjesztő jel és a hangképző út szűrőjének paramétereit és a gerjesztőjelet átvezetnénk a szűrőn, megtehetjük, hogy közvetlenül a beszédet állítjuk elő a vevő oldalon a szinuszos összetevőkből.

A szinuszos kódolók is szegmensekre bontják a beszédet. Amennyiben a vevő oldalon egy-egy szegmensen belül a többitől függetlenül szintetizáljuk a beszédet, úgy az a határokon nem lesz folytonos, ami jelentősen rontja a minőséget. Ezért a szinuszos kódolók különböző interpolációs algoritmusokat használnak a szegmensek közötti átmenetek finomítására.

Azonban ez még nem elég, mert hiába közelítjük jól a hangmagasságot, amíg a hangképző utat modellező szűrő bemenetére periodikus jelként csak egyetlen frekvenciaösszetevőből álló jelet adunk (ahogyan LPC esetén tesszük), addig csak zümmögő orrhangot kapunk. Ezt a problémát oldják meg a **szintézis általi ana-**

lízis alapú kódolók, amelyek a gerjesztő jelet egy optimalizáló hurokban határozzák meg. A kódoló egyben dekódolót is tartalmaz, amely a szintézist végzi. Az így szintetizált jelből kivonja a bemenő jelet, s az előálló hibajelet minimalizálja. A szintézis általi analízis alapú kódolók egyike a **többfrekvenciás lineáris prediktív kódoló** (MultiPulse Linear Predictive Coding, MultiPulse Excitation, MPE) eljárás, amely minden szegmensben egyidejűleg több frekvenciaösszetevőt használ. A lehetséges frekvenciaösszetevő-mintát egy kódkönyvből választja ki olymódon, hogy a valódi és a szintetizált beszédszegmens közötti, az emberi hallás tulajdonságainak megfelelően súlyozott eltérés minimális legyen (ez egy speciális vektorkvantálónak tekinthető). Az MPE 6.4 kbps átviteli sebességgel jó minőségű beszédet produkál. Ennek továbbfejlesztett változata a **kóddal gerjesztett lineáris prediktív kódoló** (Code Excited Linear Prediction, CELP), ahol a lehetséges mintákat tartalmazó (szűk) kódkönyv helyett számos gerjeszetőjelet engedünk meg (egy nagyon nagy méretű sztochasztikusan kitöltött kódkönyvet használunk). Minden szegmens esetén a kódoló megkeresi azt a gerjesztővektort, amelynek alkalmazásával a legtökéletesebb szintetizálás lehetséges. A CELP jó minőségű beszédet biztosít 4.8 kbps-os sebességgel, annak árán, hogy kimerítő keresést kell végezni egy tipikusan 1024 méretű kódkönyvben.

A **GSM** rendszer teljes sebességű kódolása az MPE-hez hasonló módszeren, a szabályos impulzus gerjesztésen (Regular Pulse Excitation, RPE) alapul. MPE esetén egy rövid perióduson belül (5–15 ms) meghatározott számú (M db) impulzust adunk. Egy adott impulzus amplitúdóját és helyét az előző M darab impulzus alapján határozzuk meg. RPE esetén az impulzusok száma szintén rögzített, és az MPE-vel szemben ezek helye sem választható szabadon. Ezáltal szuboptimális megoldáshoz jutunk, de egyben egyszerűsödik is a kódoló. Az RPE-n alapuló rövid idejű szintézis szűrőt a GSM rendszer egy hosszú idejű becslő hurokkal (Long Term Prediction) egészíti ki. (Ez a közepes bitsebességű kódolók, mint pl. az RPE esetén nem létfontosságú, de általában alkalmazzák a minőség növelése érdekében.) 8000 minta/s mintavételezési sebesség mellett 13 kbps-os kódsebességet biztosít, ugyanis 20 ms-onként 260 bitnyi paramétert (tömörített információt) állít elő.

A GSM félsebességű kódolása a CELP családba tartozó **vektorösszeggel gerjesztett lineáris prediktív kódoló** (Vector-Sum Excited Linear Prediction, VSELP) megoldáson alapul. A gerjesztésre a kódkönyvből származó értékek szolgálnak, melyek részben rögzítettek, részben adaptív jellegűek. A zöngés jelleg mértékének vizsgálata alapján a kódoló elrendezése négyféle lehet. A 0 kategória jelenti a zöngétlen, az 1–3 pedig az egyre növekvő mértékben zöngés jellegű beszédszegmenst. A struktúra 5 ms-os ún. alkeretenként változhat. A kódkönyvben kimerítő keresést alkalmaznak az alkeretenkénti gerjesztősorozat meghatáro-

zásához, és a minimális hibajel teljesítményt adó gerjesztő kódot választják ki. Ha a beszéd zöngés, akkor hosszú idejű becslőt is alkalmaznak. (Ez a kis bitsebességű kódolóknál, mint pl. a VSELP nélkülözhetetlen.) A félsebességű kódoló a teljes sebességű változat 260 bitje helyett szegmensenként csak 112 bitet használ, ez 5.6 kbps-nak felel meg.

2.9. Hangtömörítés

Az előző szakaszban az emberi beszéd minél gazdaságosabb, minél kisebb bitsebességet igénylő átvitelére koncentráltunk. Azonban a továbbítandó vagy tárolandó hanginformáció természetesen nemcsak beszéd lehet, hanem például zene is. Az ezen a területen használható módszereket tekintjük át az alábbiakban.

A hangtömörítésben mércének számító CD-minőség azt jelenti, hogy 44100 Hz-cel mintavételezünk. Ez a mintavételi tétel értelmében a legfeljebb 22050 Hz frekvenciájú hangok visszaállítását teszi lehetővé (valójában mintavételezés előtt a 20 kHz-es sávra szűrnek). A mintákat 16 bites kvantálóval kvantáljuk. Ez $44100 \cdot 16 \cdot 2 \approx 1400$ kbps átviteli sebességet igényel (a 2-es szorzó a sztereó átvitel 2 külön csatornája miatt van).

A veszteségmentes tömörítési módok (pl. Huffman-kódolás, LZW) itt nem igazán hatékonyak, az eredmény általában az eredeti méretének 90 %-a körül van.

A modern, **hangtömörítés**re kifejlesztett eljárások az emberi hallás sajátosságait kihasználva érnek el az általános célú módszereknél jobb tömörítési arányt, így amellett, hogy veszteséges eljárások, a veszteség mértéke nem is egyenletes sem a frekvencia-, sem az időtartományban. Mivel itt a közel tökéletes hang visszaállítása a cél, ezért a hangtömörítésben nem megengedett a szintetizálás.

Az emberi hallás tartománya 20 Hz – 20 kHz, a legnagyobb érzékenysége 2 és 4 kHz között van, felbontása (a kvantálási lépcső) frekvenciafüggő. Ebből következően két azonos intenzitású, de különböző frekvenciájú hang különböző hangosságérzetet kelt a hallgatóban, és azokban a tartományokban, amelyekben fülünk kevésbé érzékeny, jobban elviseljük a torzítást. Hallásunkra jellemző két elfedési jelenség. Az egyik a frekvenciatartománybeli elfedés, melynek lényege, hogy egy adott frekvenciájú, nagy intenzitású hang (maszkoló hang) a vele egy időben szóló és közeli frekvencián lévő kisebb intenzitású hangokat elfedi, vagyis azok nem hallhatóak. A másik az időtartománybeli elfedés: egy adott frekvenciájú nagy intenzitású hang a közeli frekvencián lévő kisebb intenzitású hangokat nemcsak akkor fedi el, amikor együtt szólnak, hanem kis ideig még a nagy intenzitású hang bekapcsolása előtt (kb. 2 ms-ig) vagy kikapcsolása után (kb. 15 ms-ig) sem halljuk a kisebb intenzitásúakat.

Mivel az emberi hallás legjobban a frekvenciatartományban modellezhető,

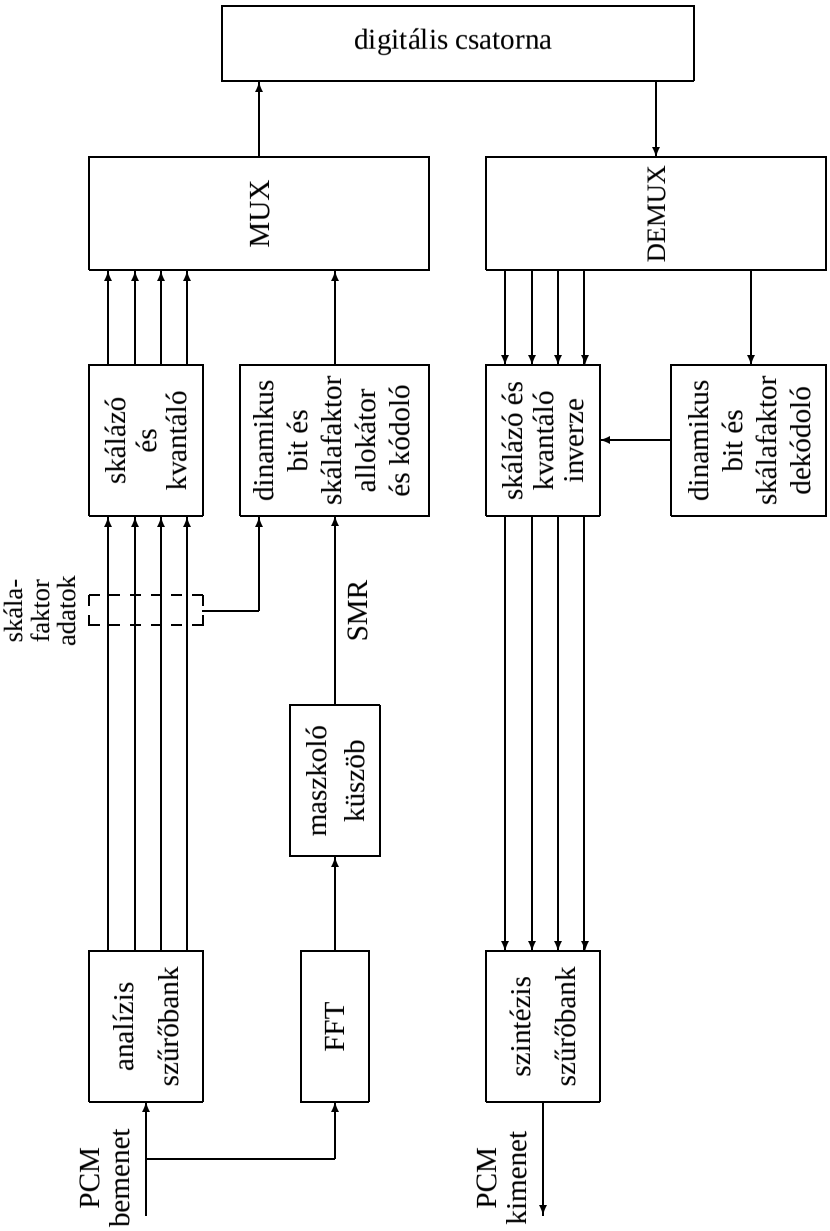
ezért a hangtömörítő eljárások frekvenciatartománybeli analízissel dolgoznak. A fentiek alapján nem minden frekvenciaösszetevőt kell átvinni, és a kódolandókat sem azonos pontossággal kell kvantálni.

Az eredetileg mozgóképtömörítésre kifejlesztett MPEG–1 szabvány hangtömörítő része jól kihasználja az emberi hallás sajátosságait. A hangtömörítésre három hasonló eljárást definiál, és ezeket Layer 1,2,3-nak nevezi.

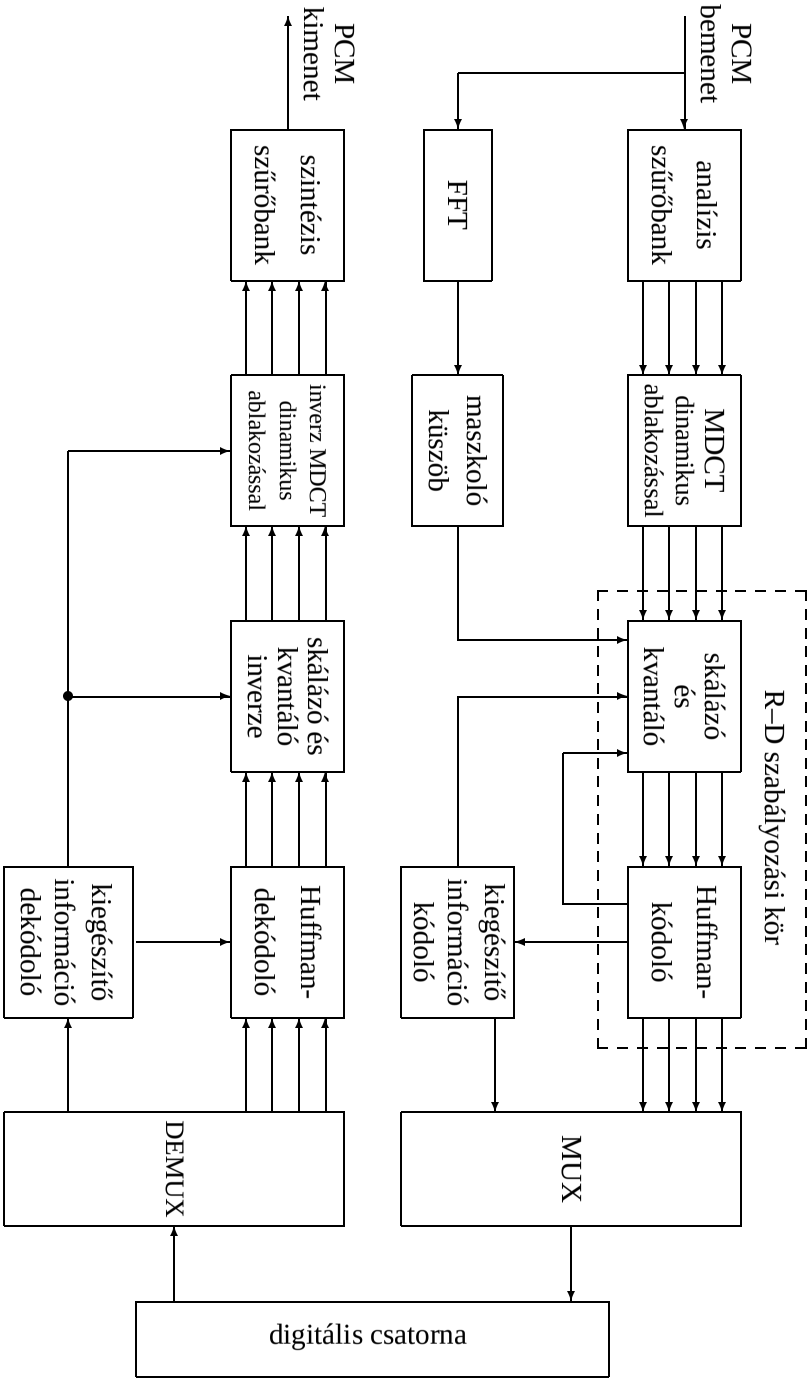
Az egyszerűbb Layer 1 és 2 eljárás analízis szűrőbankja a bemeneti jelet 32 részsávba képezi le, melyek szélessége egyenként a mintavételi frekvencia 64-ed része. A mintavételezés 32, 44.1 vagy 48 kHz-cel történhet. Ezután részsávonként egy transzformációs kódolás következik. A maszkoló és a maszkolt hang(ok) eltérését (Signal-to-Mask Ratio, SMR) Layer 1 esetén 512 pontos, míg Layer 2 esetén 1024 pontos gyors Fourier-transzformációval (FFT) számítja. Mindkét módszer 12 mintát (keret) kódol együtt minden sávban, de a Layer 2 a megelőző és a követő 12 mintát is megvizsgálja az időbeli maszkoláshoz, valamint a skálafaktorokat is hatékonyabban kódolja. Dinamikus bitallokációval kiválasztja a lehetséges 15 kvantáló egyikét minden egyes részsáv számára úgy, hogy az a skálafaktor (a legnagyobb amplitúdójú jel) és az SMR közötti bitmegosztást tekintve optimális legyen. A CD-minőségű hang átviteléhez az MPEG–1 Layer 1 eljárásnak 384 kbit/s-ra, míg a Layer 2-nek 256 kbit/s-ra van szüksége. (2.11. ábra)

A napjainkban oly nagy népszerűségnek örvendő MPEG–1 Layer 3 eljárás a Layer 1,2-höz képest lényeges, a hangminőséget javító eltéréseket tartalmaz. A 32 részsáv mindegyikét további 6 vagy 18 frekvenciaösszetevőre bontja módosított diszkrét koszinusztranszformáció (MDCT) felhasználásával. Így a 18 pontos MDCT alkalmazása $750/18 = 41.67$ Hz frekvenciafelbontást biztosít. A frekvenciakomponensekre ezután egy nemegyenletes kvantálót alkalmaz, majd a kimenetet a lehetséges 18 Huffman-kódtábla egyikével kódolja. Az eljárás tartalmaz egy iterációt a bitsebesség és az elfedési kritérium kielégítésének biztosítására. Ennek az adja meg a lehetőségét, hogy míg rögzített bitsebesség mellett minden keret azonos számú bájtot tartalmaz, addig a Layer 3 esetében meg lehet azt tenni, hogy az egyik keretet kevésbé töltjük fel (ha nincs rá igény), és a maradék helyre a következő keret bitjeit tesszük. Így egy keret adatai adott határokon belül átcsúszhatnak a szomszédos keretekbe. (2.12. ábra)

Az MPEG szabványok csak a dekódolót szabványosítják, a kódolót nem. Persze az adott dekódoló-részegységhez sok esetben létezik optimális kódolórész (pl. IDCT a dekódolóban $\longrightarrow$ DCT a kódolóban, ismertek a kódpontok a dekódolóban $\longrightarrow$ kiszámíthatóak a kvantálási tartományok a kódolóban). Bizonyos esetekben (pl. az emberi hallás modelljének felhasználása vagy a bitallokáció esetén) már nagy a szabadsági fok, ebből következnek a különböző MPEG kódolók közötti minőségi eltérések.



2.11. ábra. Az MPEG–1 Layer 1,2 hangtömörítő eljárás tömbvázlata.



2.12. ábra. Az MPEG–1 Layer 3 hangtömörítő eljárás tömbvázlata.

2.10. Kép- és videotömörítés

Kép- és videotömörítés esetén alkalmazhatunk egyértelműen dekódolható (veszteségmentes), vagy hűségkritériumon alapuló (veszteséges) módszereket. A veszteséges módszerek a veszteségmenteseknél egyes esetekben akár egy nagyságrenddel is jobban tömöríthetnek, észrevehető minőségromlás nélkül. Amennyiben a minőségi megkötések nem nagyon szigorúak, még ennél is többet nyerhetünk a veszteséges módszerek alkalmazásával. Ahhoz, hogy a minőségromlás és a tömörítés mértéke között meg tudjuk találni az egyensúlyt, szükségünk van e két jellemző kvantitatív mérésére.

A tömörítés mértékének megállapításához mérnünk kell a tömörítetlen és a tömörített információ méretét, majd ezek hányadosát kell vennünk. A tömörített információ a legtöbbször bitfolyamként jelenik meg, ennek mérete a benne levő bitek száma. A tömörítetlen információ mennyiségének mérése nehézségekbe ütközhet folytonos értékkészlet vagy értelmezési tartomány esetén. Ilyenkor az információ mennyiségét vehetjük a kódoló bementetén megjelenő jel — ami általában egy bitfolyam — méretének. Célszerűen ez a bitfolyam az eredeti jellel azonos vizuális élményt adó, de már mintavételezett és kvantálással véges értékkészletűvé alakított jel bináris ábrázolása. A tömörítetlen információ mennyiségének becsléséhez kulcsfontosságú a megfelelő mintavételezés és kvantálás kiválasztása. Ezt a legtöbb esetben szabványok határozzák meg. Ilyen szabványokkal találkozhattunk már a hangtömörítésnél is: a CD-minőségű hang a maga 44.1 kHz-es mintavételezésével és 16 bites lineáris kvantálásával pontosan ilyen volt. Képtömörítésnél a felbontást és a színmélységet kell meghatároznunk. Videotömörítés esetén emellett meg kell adnunk a képfrekvenciát is. Képtömöríésnél gyakorlatilag bármilyen felbontás használható, míg videotömörítésnél a nemzetközi televíziós szabványokhoz alkalmazkodva 720×480 (NTSC), 768×576 (PAL) a szokásos felbontás. Az ezekhez tartozó képfrissítési frekvenciák pedig 29.97 Hz illetve 25 Hz. A színtérbeli kvantálást a színmélység, vagyis az egy képpontra jutó bitek száma jellemzi. Általánosan alkalmazott a 8 és a 24 bites színmélység.

A hűségkritériummal való kódolás csakis akkor értelmes, ha azt úgy határozzuk meg, hogy egy adott minőséget garantáljon. Sajnos csak szubjektív értékelésekre támaszkodhatunk egy adott kép vagy video minőségének megítélése során. Egy veszteséges tömörítési módszer értékelése céljából úgy járhatunk el, hogy bizonyos tesztadatokon végrehajtjuk az adott tömörítést, majd a tömörítés inverzét, és az így visszakapott kép vagy video minőségét egy tesztcsoporttal értékeltetjük. Ha a teszt lefolytatása során elég körültekintően járunk el, akkor viszonylag objektív eredményeket kaphatunk az adott módszer minőségéről. Például egy lehetséges eljárás, ha a vizsgálni kívánt tömörítést jól definiált egyéb tömörítő-

sekkel hasonlíttatjuk össze. Hűségkritériumnak tekinthetjük azt, hogy sikerült-e egy adott standard módszernél jobb minőséget elérni. Megfelelően standardizált teszteket ilyen értelemben tekinthetünk hűségkritérumnak.

Az emberi látás

A veszteséges kép és videotömörítési módszereket az emberi látásról meglevő ismereteket felhasználva alakították ki. Az információmennyiség csökkentése érdekében a kép azon részleteit kódoljuk kis pontossággal, amelyekre a szemünk kevéssé érzékeny, és azokat kódoljuk majdnem eredeti minőségben, amelyekre a szemünk igen érzékeny.

A színek és a fényesség érzékeléséért a szemben található fényérzékelő receptorok, a csapok és a pálcikák a felelősek. A pálcikák azonban csak a látás perifériális tartományában illetve igen kis fényességek esetén játszanak jelentős szerepet az érzékelésben, jelenlétüktől ezért most eltekinthetünk. Háromféle különböző spektrális érzékenységű csap található a szemünkben. Ezek a csapok az őket érő fényre lineárisan reagálnak. Spektrális érzékenységüket jelöljük rendre $\bar{s}(\lambda)$-val, $\bar{m}(\lambda)$-val és $\bar{l}(\lambda)$-val. Ezek a függvények a látható spektrumtartományon (360 nm–830 nm) kívül 0 értéket, belül pedig nemnegatív értéket vesznek fel. $\bar{s}(\lambda)$ csúcsa a kisebb hullámhosszak tartományában található, $\bar{m}(\lambda)$ csúcsa a közepes hullámhosszaknál, míg $\bar{l}(\lambda)$ csúcsa a nagyobb hullámhosszaknál. Ezért kapták nevüket az angol short, medium és long szavak kezdőbetűiből.

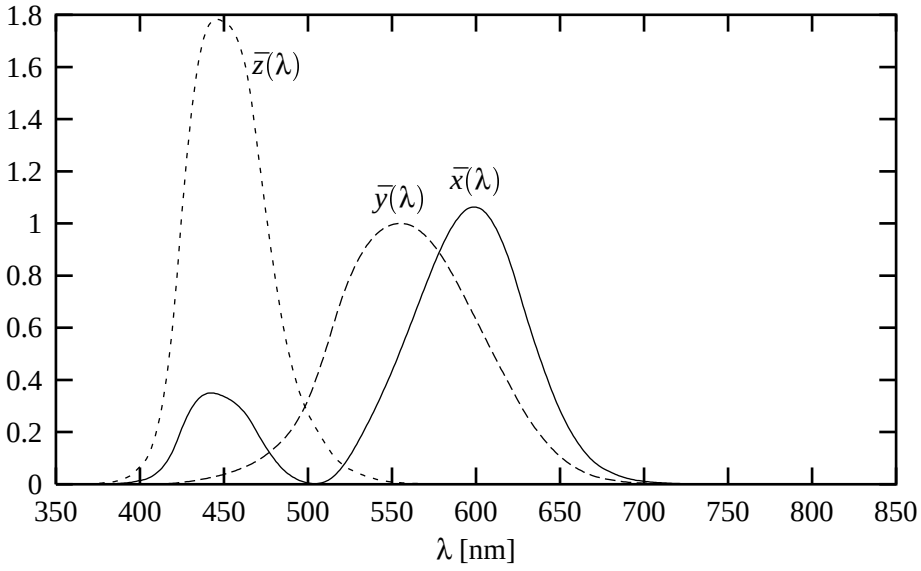
Egy $L(\lambda)$ spektrális energiasűrűséggel rendelkező fénysugár által kiváltott inger az alábbi vektorral jellemezhető:

$$\begin{pmatrix} S \\ M \\ L \end{pmatrix} = \int \begin{pmatrix} \bar{s}(\lambda) \\ \bar{m}(\lambda) \\ \bar{l}(\lambda) \end{pmatrix} L(\lambda)\,\mathrm{d}\lambda$$

Az integrálást itt a látható spektrumtartományon kell elvégezni. Természetesen $L(\lambda)$ semmilyen λ-ra sem lehet negatív, ugyanis a negatív spektrális energiasűrűségnek nincs fizikai értelme. Így nyilvánvalóan S, M és L értéke is nemnegatív.

Ebből az ingerhármasból alakul ki idegrendszerünkben nemlineáris feldolgozás során a fényesség- és színérzet. Fontos következménye ennek, hogy két különböző spektrummal rendelkező fényt nem tudunk megkülönböztetni, ha a hozzájuk tartozó S, M és L értékek megegyeznek. Ezt a jelenséget nevezzük metamer színérzetnek, és az azonos S, M és L ingert kiváltó (vagyis azonos fényesség- és színérzetet keltő, emberi szem által megkülönböztethetetlen) spektrumokat nevezzük metamer színeknek.

Mivel szemünk érzékenysége a fényességre jóval nagyobb, mint a színre, érdemes ezeket az adatokat elkülöníteni egymástól. Ehhez végezzünk az ingerekből



2.13. ábra. Az $\bar{x}(\lambda)$, $\bar{y}(\lambda)$ és $\bar{z}(\lambda)$ spektrális súlyfüggvények.

álló vektoron egy lineáris transzformációt:

$$\begin{pmatrix} X & Y & Z \end{pmatrix}^T = \mathbf{M} \begin{pmatrix} S & M & L \end{pmatrix}^T$$

Ebben a transzformált térben Y a fényességérzetet, míg X és Z a színérzetet írja le. Természetesen egy $L(\lambda)$ spektrumú fény (X,Y,Z) koordinátái egy lépésben (S, M és L kiszámítása nélkül) is számíthatóak:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \int \begin{pmatrix} \bar{x}(\lambda) \\ \bar{y}(\lambda) \\ \bar{z}(\lambda) \end{pmatrix} L(\lambda)\,\mathrm{d}\lambda$$

Az integrálást itt is a látható spektrumra kell elvégeznünk, és az

$$\begin{pmatrix} \bar{x}(\lambda) & \bar{y}(\lambda) & \bar{z}(\lambda) \end{pmatrix}^T = \mathbf{M} \begin{pmatrix} \bar{s}(\lambda) & \bar{m}(\lambda) & \bar{l}(\lambda) \end{pmatrix}^T$$

súlyfüggvényeket kell alkalmaznunk. $\bar{x}(\lambda)$-t, $\bar{y}(\lambda)$-t és $\bar{z}(\lambda)$-t ábrázoltuk a 2.13. ábrán. Fontos tény, hogy **M**-et úgy választották, hogy minden nemnegatív értékkészletű $L(\lambda)$-hoz tartozó X, Y és Z értékek nemnegatívak legyenek.

A gyakorlatban a fényesség leírására Y megfelelő, a színérzetet pedig az $x = \frac{X}{X+Y+Z}$ és $y = \frac{Y}{X+Y+Z}$ koordinátákkal szokták megadni. (Vagyis x és y az adott spektrumhoz tartozó (X,Y,Z) vektor egyenese és az $X+Y+Z=1$ sík döféspontjának X és Y koordinátája.) Ezáltal tehát egy tetszőleges $L(\lambda)$ spektrumhoz

rendeltünk (Y,x,y) koordinátákat. Természetesen egy adott (Y,x,y) koordinátahármas több különböző spektrumhoz is hozzá lett rendelve.

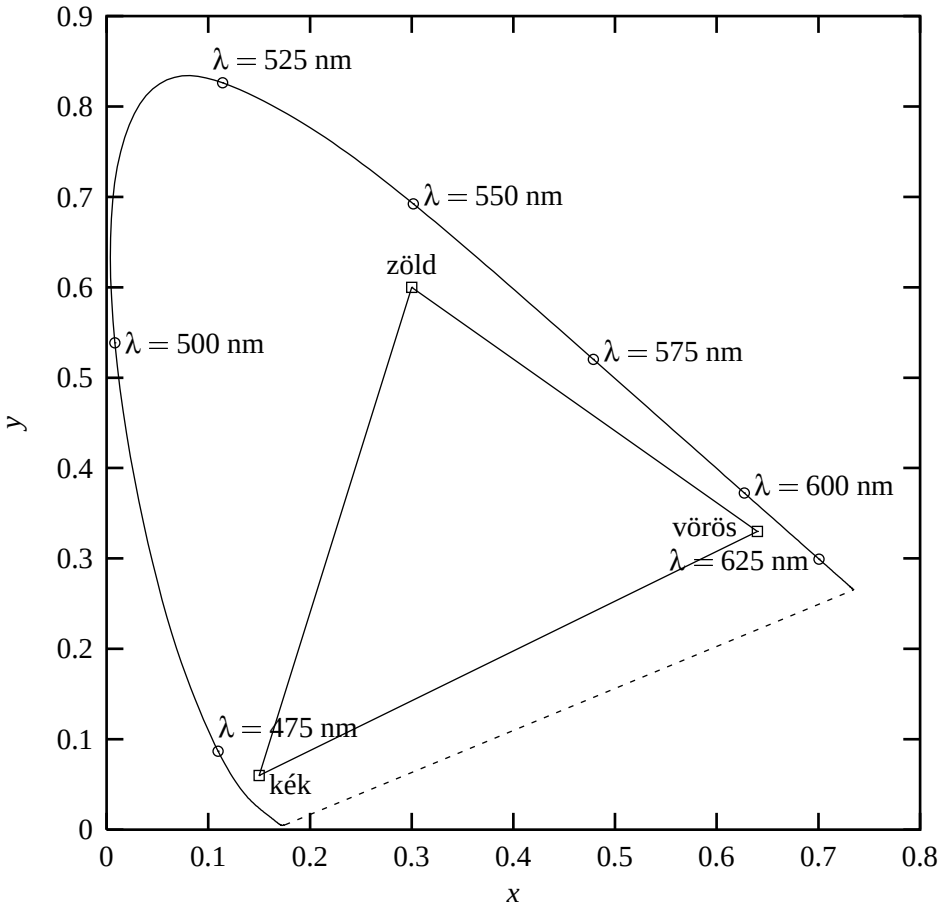
Ha csak a színérzettel foglalkozunk, elegendő az (x,y) koordinátapárossal számolni. Ennek az (x,y) színkoordináta-rendszernek az az előnye, hogy két tetszőleges spektrumból additívan kikeverhető spektrumok a két kiindulási spektrum által meghatározott szakaszon fekszenek. Matematikailag, ha $L_1(\lambda)$ spektrumhoz az (x_1,y_1) pont tartozik, és $L_2(\lambda)$-hoz pedig (x_2,y_2), akkor tetszőleges nemnegatív ν-re és μ-re az $L(\lambda)=\mu L_1(\lambda)+\nu L_2(\lambda)$ spektrumhoz tartozó (x,y) pont az (x_1,y_1)-et és (x_2,y_2)-t összekötő szakaszon fekszik. Sőt, általánosságban is kimondható, hogy az $L_i(\lambda)$ spektrumok nemnegatív együtthatós lineárkombinációihoz $(\sum_i \mu_i L_i(\lambda), \mu \geq 0)$ rendelt (x,y) koordináták az $L_i(\lambda)$ spektrumokhoz rendelt (x_i,y_i) pontok konvex burkán belül fekszenek.

Az egyetlen spektrális összetevőt tartalmazó $(L(\lambda)=c\,\delta(\lambda-\lambda_0))$ spektrumokat monokromatikus spektrumoknak nevezzük. A monokromatikus spektrumokhoz tartozó pontok egy görbét adnak az (x,y) színkoordináta-rendszerben. A görbe és a két végpontját összekötő szakasz által határolt területet nevezzük színpatkónak. Ezt tüntettük fel a 2.14. ábrán.

Mivel tetszőleges spektrum felírható monokromatikus spektrumok integráljaként, a fentiek értelmében tetszőleges valódi spektrumhoz tartozó (x,y) koordináták a színpatkó területén helyezkednek el. (Ugyanis ez a tartomány a monokromatikus spektrumok pontjainak konvex burka.) Másképpen fogalmazva: nemnegatív $L(\lambda)$-ból számolt (x,y) értékek nem eshetnek a színpatkón kívülre.

A ma használt képernyők legnagyobb része három különböző színű foszfor által kibocsátott fény additív keverésével dolgozik. A foszforokat különböző mértékben gerjesztve különböző fényesség- és színérzetek kelthetőek. Ha a fenti (x,y) koordinátarendszerben bejelöljük a felhasznált foszforok által kibocsátott fény spektrumához tartozó pontokat, akkor egy háromszöget kapunk, mely a képernyő által kelthető színérzetek tartományát határolja. Az ezen tartományon kívül eső (x,y) koordinátákhoz tartozó színérzetek keltésére az adott képernyő nem alkalmas. Példaként a 2.14. ábrán feltűntettük a HDTV szabványban szereplő foszforoknak megfelelő pontokat, és a velük megjeleníthető színek tartományát.

Ha a fent bemutatott módon működő képernyővel jelenítünk meg, érdemes a fényességeket és színeket nem (Y,x,y) koordinátákkal, hanem olyan koordinátákkal megadni, amelyek azt mutatják, hogy a három alapszínt milyen arányban kell keverni a kívánt színhatás eléréséhez. Mivel a megjelenítők nemlineáris karakterisztikával rendelkeznek, a megfelelő nemlineáris előtorzítással (gammakorrekció) kapott (R',G',B') értékeket szokták használni. Ezek értéktartománya a 0 és 255 közötti egész számokból áll. Ilyen színkezelés esetén 2^{24} különböző színt tudunk reprodukálni, ami a legtöbb esetben kielégítő. Az (R',G',B') színko-



2.14. ábra. Az (x,y) színkoordináta-rendszer és a HDTV képernyőkben használt foszforok által megjeleníthető tartomány.

ordináták csak a három alapszín által határolt háromszögön belüli színeket tudják leírni, de ha úgyis csak ezt a tartományt tudjuk megjeleníteni, akkor nincs is értelme az ezen tartományon kívüli színek leírásának.

Egyes esetekben el kell választanunk a fényesség- és a színinformációt. Erre természetesen alkalmasak lennének az (Y,x,y) koordináták, viszont ezek valós értékűek, nehézkes velük számolni és nehéz őket hatékonyan tömöríteni. További hátrány, hogy bár Y a fényességérzetet jellemzi, szemünk nem egyenletesen érzékeny az Y koordinátában: például az $Y = 0.1$ és az $Y = 0.2$ közötti fényességkülönbséget jóval nagyobbnak látjuk, mint az $Y = 0.7$ és $Y = 0.8$ közötti különb-

séget. Az (R',G',B') értékekből affin transzformációval kaphatóak az (Y',C_b,C_r) színkoordináták. Y'-t **luminanciá**nak nevezzük, és a fényességérzetet írja le, C_b-t és C_r-t **krominanciá**nak nevezzük, ezek a színérzetet írják le.

$$\begin{pmatrix} Y' \\ C_b \\ C_r \end{pmatrix} = \begin{pmatrix} 16 \\ 128 \\ 128 \end{pmatrix} + \frac{1}{256} \begin{pmatrix} 65.738 & 129.057 & 25.064 \\ -37.945 & -74.494 & 112.439 \\ 112.439 & -94.154 & -18.285 \end{pmatrix} \begin{pmatrix} R', G', B' \end{pmatrix}$$

Ennek a színkoordináta-rendszernek az Y' együtthatójában már többé-kevésbé állandó a szemünk érzékenysége. További előny, hogy ezek az értékek is 0 és 255 közötti egész számok, és belőlük az (R',G',B') számértékek egyszerűen előállíthatóak. Hátrányuk viszont, hogy csak a teljes színpatkó egy háromszög alakú tartományát tudjuk velük reprezentálni.

Most foglaljuk össze, hogy az emberi színérzékelésről szerzett információinkat hogyan használhatjuk fel kép- és videotömörítés esetén. Bizonyos esetekben, amikor a cél nem a színérzet tárolása, hanem egyéb későbbi feldolgozás (például csillagászati felvételek esetén), akkor szükséges lehet a beérkező fény spektrumának mintavételezett formáját tárolni. Ha azonban csak a vizuális élmény visszaadása a célunk, akkor a metamer színérzet miatt képpontonként elegendő mindössze három valós számértéket (pl. (Y,x,y)) tárolni. Ha ezen felül azt is tudjuk, hogy a képet a későbbiekben vörös, zöld és kék foszfort alkalmazó megjelenítőn akarjuk rekonstruálni, vagy legalábbis megelégszünk az ezen megjelenítők által előállítható színtartománnyal és 2^{24} szín megkülönböztetésével, akkor elegendő csak három nyolcbites értéket (pl. (R',G',B')) tárolni. (Nyomdatechnika esetén például előfordulhat, hogy ez a tárolás nem kielégítő.) Veszteséges tömörítés esetén felhasználhatjuk azt is, hogy a fényességre jobban érzékeny a szemünk, mint a színre. Ilyenkor fogjuk az (Y',C_b,C_r) számhármast használni, és nagyobb torzítást engedünk meg C_b és C_r értékére, mint Y' értékére.

Szemünknek sok egyéb olyan tulajdonsága is van, amit kép- és videotömörítéskor kihasználhatunk. Megfigyelhetjük, hogy a nagyobb térbeli frekvenciájú összetevőkre kevésbé érzékeny a szemünk mint az alacsonyabb frekvenciájúakra. Például egy halvány sűrű mintázatot sokkal kevésbé veszünk észre, mint egy ugyanolyan halvány, de ritkás mintát. Tömörítésnél, ha a kép kétdimenziós Fourier-transzformáltját vesszük, akkor ebben a magasabb frekvenciás összetevők torzítása nem okoz észrevehető hibát, míg az alacsonyabb frekvenciájú összetevők torzítása igen feltűnő a képen. Pontosabban, a kétdimenziós Fourier-transzformált frekvenciatartományban a szem érzékenysége a két frekvencia összegének növekedésével monoton csökken.

Kísérleti tény továbbá, hogy a térbeli és időbeli frekvenciák érzékelése összefügg: egy rövid ideig látott képen csak az igen kicsi térbeli frekvenciájú összetevőket figyeljük meg, a finomabb mintázatokat csak hosszabb ideig látott képen

vagyunk képesek érzékelni. Veszteséges videotömörítésnél ennek megfelelően megengedjük, hogy egy hirtelen megjelenő képrészletnek az első néhány képkockán csak az alacsonyabb térbeli frekvenciás összetevői jelenjenek meg. A magasabb frekvenciás összetevőket elég csak ezután megjeleníteni.

Graphics Interchange Format (GIF)

A GIF formátumot a Compuserve Information Service foglalta szabványba. Grafikus képek **veszteségmentes** tömörítésére lehet használni. Az algoritmus LZW alapokon működik. Lényegében egy dinamikusan növekvő szótárral dolgozó megoldás.

A GIF formátum a képpontok színét úgynevezett indexelt formában tárolja. Amikor minden képpont színét a fentebb bemutatott színkoordináta-rendszerek valamelyikében adjuk meg, **folytonos tónusú tárolás**ról beszélünk. Megtehetjük azonban, hogy a képen használt színek valamilyen leírását egy palettába gyűjtjük ki, és a képpontok megadásakor a színekre a palettabeli indexükkel hivatkozunk. Ezt nevezzük **indexelt tárolás**nak.

Ilyen tárolással igen nagy tömörítést érhetünk el. Gondoljuk meg, hogy ha egy képet az (R', G', B') színkoordinátákkal, koordinátánként 8–8 bitet felhasználva folytonos tónusúan tárolunk, akkor minden képponthoz 24 bitet kell megadnunk. Ha ugyanezt a képet egy 256 színű palettával indexelten tároljuk, akkor a palettán kívül képpontonként csak egy 8 bites indexet kell megadnunk. Ez — amennyiben a képméret nem túl kicsi — körülbelül $1:3$ tömörítést jelent.

Természetesen bármilyen kép tárolható indexelt formában, legfeljebb igen nagy méretű paletta szükséges. Az indexelt tárolás lényege viszont épp a viszonylag kis méretű paletta alkalmazása. A képek folytonos tónusú tárolással kerülnek a tömörítőalgoritmusok bemenetére, gondoljunk például szkennelt fényképre vagy digitalizált videojelre. (Ez alól csak a számítógépes grafikák alkotnak néha kivételt, ugyanis ezek lehet, hogy azonnal indexelt formában készülnek.) Ezért szükséges lehet, hogy egy folytonos tónusú képet indexeltté alakítsunk. Ennek során minél kisebb palettaméretet szeretnénk elérni, annál többet kell az eredeti kép színei közül másikkal helyettesíteni. Felfoghatjuk ezt úgy is, hogy az indexelt tárolású formává való átalakítás egy vektorkvantálás a színtérben. Az optimális (tehát a lehető legkevésbé látható torzulást okozó) paletta (vagyis az optimális kvantálási vektorok) kiválasztására speciális — az emberi színérzékelés tulajdonságait kihasználó — algoritmusok állnak rendelkezésre.

A GIF formátumban a paletta mérete általában 256, de bármely kisebb 2 hatvány használata is megengedett. A palettában a színek megadása az (R', G', B') színkoordinátákkal történik. A paletta tömörítésének miként jével nem foglalko-

zunk. A képpontokat a GIF tömörítés sorfolytonosan tapogatja le, és a sorozatot LZW algoritmussal tömöríti.

A tömörített kép első bájtjának értéke a képpontonkénti bitszám (b). Ez határozza meg a palettaméretet: 2^b. Az LZW szótár kezdeti mérete 2^{b+1}, tehát minden kódszó $b+1$ bites. Az első néhány képpont szempontjából (amíg nincs a képpontok sorozatában ismétlődés) ez azt jelenti, hogy képpontonként egy plusz bit képződik, hisz a b bites képpont $b+1$ biten lesz tárolva. Közben azonban épül az LZW szótár, és ha a képpontértékek sorozatában valahol ismétlődés van, azt az LZW azonnal kihasználja. Ha a 2^{b+1} méretű szótár betelt (ezt a pillanatot természetesen a kitömörítő algoritmus is meg tudja állapítani), a szótár méretét az algoritmus megduplázza, és ettől kezdve $b+2$ bites kódszavakkal dolgozik. Ha ez a szótár is betelik, akkor ismét duplázódik a méret, amíg el nem éri a 4096 bejegyzést. Innentől már nem növekszik tovább a szótár, hanem az eddig elkészült szótárat fogja a továbbiakban statikus szótárként használni az algoritmus.

A GIF formátum nagyon elterjedt, leginkább kis ikonok, ábrák tömörítésére alkalmazzák, ugyanis ezeket képes igen jól tömöríteni. Ennek egyik oka, hogy az ikonok és ábrák sok esetben nagy kiterjedésű, egyszínű területeket tartalmaznak, és ezek a sorfolytonos letapogatással hosszú egyszínű sorozatokként jelennek meg. A hosszú sorozatok az LZW szótárban egy kódszóval reprezentálhatók, így igen hatékony a tömörítés. Ugyancsak jellegzetessége az ilyen képeknek, hogy ismétlődő részleteket tartalmaznak, és ezek megintcsak kedveznek az LZW algoritmusnak. Azonban például fotók esetén az egyszínű területek igen ritkák, és méretük is viszonylag kicsi. Ennek eredményeképpen a GIF tömörítési aránya ilyen képnél ($1:1.2-1.7$) elmarad sok más, viszonylag egyszerű módszer mögött. (Például, ha predikcióként az előző képpont értékét használjuk, és a kapott különbségi sorozatot adaptív aritmetikai kódolóval (tehát gyakorlatilag emlékezetnélküli forrás entrópiájának megfelelő bitsebességgel) tömörítjük, akkor a tömörítési arány $1:1.3-2.2$ lesz.)

A különbség másik oka a színkezelésben keresendő. Az ikonok és ábrák általában viszonylag kevés színt tartalmaznak, ezért igen előnyös az indexelt tárolásuk. Egy olyan tömörítés, amely nem tud indexelt képeket tömöríteni, egy ilyen ábrát kénytelen lenne folytonos tónusúvá alakítani a tömörítés előtt. Így kétszer-háromszor akkora adatmennyiséget kellene, hogy tömörítsen, mint a GIF, amely közvetlenül az indexelt formával képes dolgozni. Sok módszer azért nem tud közvetlenül indexelt képeken dolgozni, mert a szomszédos képpontok értékeinek kis különbségére alapoz. Mivel folytonos tónusú tárolásnál a szomszédos képpontokban tárolt értékek a színkoordináták, ez a feltételezés helyes, és általában teljesül is. Indexelt tárolás esetén azonban a képpontokban tárolt értékek a palettaindexek, és ezek tetszőlegesen távoliak lehetnek (csak a nekik megfelelő palettabeli szín-

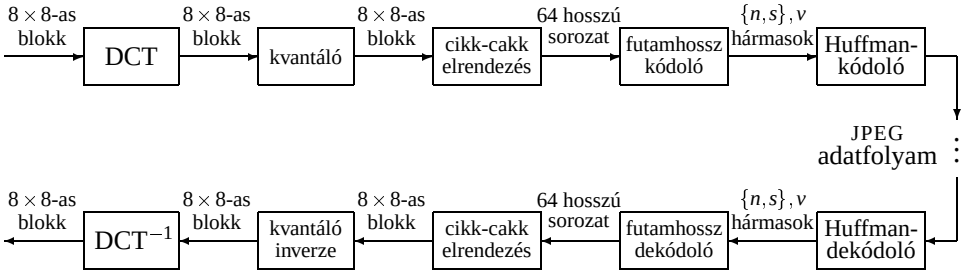
koordináták értékei vannak egymáshoz közel). Fokozottan jelentkezik ez akkor, ha valamilyen palettaoptimalizáló eljárást alkalmaztunk. Eléggé jól látható, hogy a GIF-nek miért nem probléma ez: a GIF ismétlődő mintázatokat keres, és ilyen szempontból teljesen mindegy, hogy a közeli képpontok értékei megközelítőleg egyenlőek-e. Mindössze az számít, hogy vannak-e ismétlődések. Így már érthető, hogy miért is olyan hatásos módszer a GIF számítógépes ábrák tömörítésénél.

Joint Photographic Experts Group (JPEG)

A JPEG a képtömörítés civil világában szinte egyeduralkodó a veszteséges tömörítések területén. Az egész szabványcsomagot a Joint Photographic Experts Group (ISO/IEC JTC 1 / SC 29 / WG 1) dolgozta ki folytonos tónusú képek tömörítésére. Valójában a JPEG szabványcsomag támogat veszteséges és veszteségmentes képtömörítést is, bár tény, hogy a gyakorlatban szinte kizárólag az elsőt használják. Mindezek ellenére a prediktív kódolás jó példája a veszteségmentes JPEG, ezért az alábbiakban ezt is bemutatjuk, továbbá részletesen tárgyaljuk a veszteséges JPEG egy egyszerű változatát, a baseline JPEG-et.

A **veszteségmentes** JPEG képtömörítés alapja a predikciós kódolás. Predikcióra az éppen tömörítésre kerülő képpont környezetét használjuk fel. Mivel a képpontok letapogatása itt is sorfolytonosan történik, az adott képponttól balra vagy felfelé eső képpontok értékei azok, amelyek a dekódoló rendelkezésére állnak az adott képpont értékének visszaállításakor, így a kódoló is csak ezeket használja fel a predikcióhoz. Nyolcféle predikciós séma létezik, és ezek közül egy adott kép tömörítéséhez bármelyik használható, de egy képen belül végig ugyanaz a séma érvényes. Jelölje $X_{i,j}$ az (i,j) koorinátájú képpont értékét, és $\widehat{X}_{i,j}$ az (i,j) koordinátájú képpont becsült értékét. Az (i,j) koordinátájú képpont becsült értéke az egyes predikciós sémákban:

$$
\begin{array}{ll}
0 & \widehat{X}_{i,j} = 0 \\
1 & \widehat{X}_{i,j} = X_{i-1,j} \\
2 & \widehat{X}_{i,j} = X_{i,j-1} \\
3 & \widehat{X}_{i,j} = X_{i-1,j-1} \\
4 & \widehat{X}_{i,j} = X_{i,j-1} + X_{i-1,j} - X_{i-1,j-1} \\
5 & \widehat{X}_{i,j} = X_{i,j-1} + \dfrac{X_{i-1,j} - X_{i-1,j-1}}{2} \\
6 & \widehat{X}_{i,j} = X_{i-1,j} + \dfrac{X_{i,j-1} - X_{i-1,j-1}}{2} \\
7 & \widehat{X}_{i,j} = \dfrac{X_{i,j-1} + X_{i-1,j}}{2}
\end{array}
$$



2.15. ábra. Veszteséges baseline JPEG tömörítés.

Látható, hogy az első lehetőség maga a predikciómentes tömörítés. Megfigyelhető, hogy az összes többi predikciós séma mellett az egyszínű területeken a predikciós hiba 0, azaz $\widehat{X}_{i,j} = X_{i,j}$.

A predikciós hibát adaptív aritmetikai kódolással ajánlott tömöríteni. Javítható a tömörítés mértéke, ha a képet blokkokra osztjuk, és minden blokkra külön adjuk meg a használni kívánt predikciós sémát.

A **veszteséges** baseline JPEG tömörítés algoritmusa a 2.15. ábrán látható. A JPEG a képeket először képsíkokra bontja: minden képsík egy-egy színkoordináta értékeit tartalmazza. Színkoordinátának az előzőekben bemutatott (Y', C_b, C_r) hármast használjuk, mindegyiket 8-8 biten tárolva. Így egy képpontra összesen 24 bit jut. A JPEG tömörítő algoritmus számára a képsíkok gyakorlatilag egymástól független képekként jelennek meg, külön-külön vannak kódolva.

Megengedett, hogy a képsíkok felbontása eltérjen. Mindössze annyi a kikötés, hogy a felbontások aránya racionális legyen. A kép kitömörítésekor a felbontások legkisebb közös többszörösének megfelelő felbontású képet állítunk vissza, ahol az egyes képsíkok képpontbeli értékeit lineáris interpolációval állítjuk elő a tárolt felbontással megegyező felbontásban rekonstruált képsík értékeiből. A különböző felbontású képsíkok alkalmazásának lehetőségét — az emberi látás tulajdonságairól tanultaknak megfelelően — úgy szokás kihasználni, hogy a C_b és C_r képsíkok vízszintesen és függőlegesen is felére csökkentett felbontással vannak tárolva. Például 400×400 pontos kép esetén az Y' képsík felbontása 400×400, míg a C_b és C_r képsíkok felbontása 200×200. Kitömörítésnél visszaállítjuk a 400×400-as Y' képsíkot, és a két 200×200-as C_b és C_r képsíkot. Ezután a C_b és C_r képsíkokat lineáris interpolációval a kétszeresükre nagyítjuk, majd az így kapott két 400×400-as képsíkot és a 400×400-as Y képsíkot „vetítjük egymásra”, hogy az eredeti képet kapjuk.

Az algoritmus első lépése a képsíkok 8×8-as **négyzetekre bontás**a. Ha a képméret nem osztható 8-cal, akkor a jobb szélső oszlop és/vagy a legalsó sor

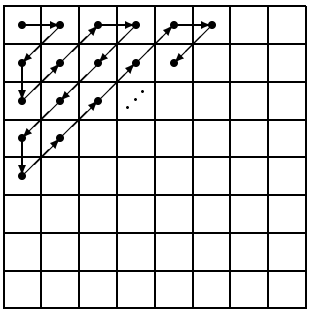
többszörözésével azt 8-cal oszthatóvá tesszük. (Az így keletkező felesleges szélek levágása a dekódoló feladata — ezt a képméret alapján könnyű megtenni, a képméret pedig a JPEG fejlécben van tárolva.) Más tömörítési módszerek, ha arra kényszerülnek, hogy a bemenő adatsort meghosszabbítsák — valamilyen kötött blokkméret miatt —, akkor azt általában 0-kkal egészítik ki. Érdekes a JPEG megoldása erre az esetre. Az a furcsa, hogy egy igen információdús adatdarabbal történik a kiegészítés: a kép egy részletével. Sok egyéb algoritmus esetén a 0-kkal való feltöltés a 0 sorozat jó tömöríthetősége miatt ajánlott. Esetünkben azonban a tömörítés későbbi lépései arra építenek, hogy egy kép alapvetően homogén, és csak ott kell valamit lekódolni, ahol a kép változik. Ennek megfelelően a legnagyobb nehézséget az éles határok kódolása okozza — ezek kódolása teszi ki a tömörített adatfolyam legnagyobb részét, ezek térnek el legjobban a „minden egyszínű” megközelítéstől. A JPEG kódolás annyira jól használja ki a kép redundanciáját, hogy több bitbe kerülne egy 0 sorral vagy oszloppal való kiegészítés által behozott új él lekódolása, mint az utosó sor már amúgy is feltérképezett szerkezetének megismétlése. Ezért döntöttek a szabványban ezen megoldás mellett.

A 8×8-as négyzeteket kétdimenziós **DCT transzformáció**nak vetjük alá. Így ugyancsak 8×8-as négyzeteket kapunk, csak azok most már nem egész értékekből, hanem valós számokból állnak. Ezeket a frekvenciatartománybeli valós együtthatókat a következő lépés, a kvantálás fogja ismét egészekké alakítani.

A **kvantálás** egyenletes, viszont a 8×8-as négyzet minden egyes elemére más lépésközzel hajtjuk végre. A 8×8-as DCT transzformált négyzet együtthatói különböző frekvenciájú felharmonikusoknak felelnek meg. A kis frekvenciás együtthatók a négyzet bal felső részében vannak. Mint tudjuk, a szem ezekre a kisebb frekvenciához tartozó értékekre sokkal érzékenyebb, mint a nagyobb frekvenciához tartozókra, ezért ezeket finomabb lépésközzel fogjuk kvantálni. A szabvány nem köti meg, hogy milyen kvantálási lépésközöket használjunk. A használt lépésközöket táblázatba szokás foglalni, ahol egy táblázatelem a 8×8-as négyzet megfelelő együtthatójának kvantálási lépésközét tartalmazza. Bár a szabvány nem teszi kötelezővé a használatát, de ajánl egy kvantálási táblát:

$*$	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

A 0 frekvenciához tartozó együttható (neve DC komponens) a DCT transzformá-



2.16. ábra. Cikk-cakk elrendezés.

ció során a bal felső sarokba került. Kihasználva az egymás melletti képrészek viszonylag hasonló színét, a DC komponenst nem kvantáljuk, ehelyett az egymás utáni 8×8-as blokkok DC komponenseinek különbségét vesszük. Ezért nem tartozik a DC komponenshez érték a kvantálási táblában.

Látható, hogy a nagyobb frekvenciás együtthatókra nagyobb lépésközt ad meg a táblázat. Ennek eredményeképpen a nagyobb frekvenciás együtthatók kvantált értékei majdnem mind 0-k lesznek. Ezeket a 0-kat egy későbbi lépés (a futamhossz kódolás) során jól tudjuk majd tömöríteni. A kvantálási tábla változtatása tehát lehetőséget ad a tömörítés mértékének és minőségének befolyásolására: a tömörítést növelni lehet a kvantálás lépésközének növelésével, a kép minőségének rontása árán.

A kvantált együtthatókat ezután az úgynevezett **cikk-cakk elrendezés** szerint sorbarendezzük (2.16. ábra). Így az első helyre a különbségi DC együttható kerül, azt követi a vízszintes alapharmonikus, majd a függőleges alapharmonikus, majd pedig az egyre növekvő frekvenciás felharmonikusok következnek. Köszönhetően annak, hogy a kvantálás során a táblázat magasabb frekvenciákhoz tartozó együtthatói nagyrészt kinullázódtak, a kapott sorozat a vége felé túlnyomórészt 0-kból áll. Ezt használjuk ki a következő lépésben a futamhossz kódolással.

A **futamhossz kódolási** lépésben a cikk-cakk elrendezéssel kapott, javarészt 0-kból álló sorozatot részekre bontjuk úgy, hogy minden részsorozat eleje tetszőleges számú 0-ból álljon, és ezt a végén egyetlen nem 0 elem zárja. Minden így kapott részsorozathoz egy $(\{n,s\},v)$ számhármast rendelünk. n az adott részsorozat elején levő 0-futam hossza, s jelzi, hogy a részsorozat végén található nem 0 értéket hány biten kódoljuk, v pedig ezen nem 0 érték bináris ábrázolása. Az s változóra a tömörítés hatékonyságának növelése miatt van szükség, mert ugyan előfordulnak néha nagy abszolút értékű elemek, de ezek ritkák, így nem érdemes minden elemet azonos bithosszon kódolni.

Így például a $23,0,0,0,0,-7,0,-19,4,0,0,0,\ldots$ sorozat első futamhossz-kódhármasa: $(\{0,6\},010111)$, hiszen az első elem, a 23 előtt nincs 0, így $n=0$. 23 bináris számként 6 biten kódolható előjelesen, így $s=6$. A további kódszavak: $(\{4,4\},1000)$; $(\{1,6\},101100)$; $(\{0,4\},0100)$;... (Az előjeles bináris számokat egyes komplemens ábrázolásban tüntettük fel. Megfelelő technikával ezek 1 bittel rövidebben is kódolhatóak.)

Az így kapott $(\{n,s\},v)$ hármasok $\{n,s\}$ elemeit statikus Huffman-kóddal tömörítjük tovább, a v értékeket pedig egyszerűen elválasztójel nélkül egymás után írjuk. A statikus Huffman-kód azt jelenti, hogy nem az $\{n,s\}$ párok adott képben levő gyakorisága alapján választjuk a kódszavakat, hanem azok a JPEG szabványban le vannak rögzítve. (A statikus kódot az $\{n,s\}$ párok sok képre vett átlagos gyakoriságai alapján tervezték meg.) Alkalmazhatnánk a Huffman-kódolást magukra az $(\{n,s\},v)$ hármasokra is, de akkor a kód annyira sok kódszóból állna, hogy kezelhetetlen lenne, nagyon lassú lenne vele dolgozni. A v értékek leválasztása jó döntés, mert ezzel a kód mérete kezelhető lesz, viszont a hatékonysága csak alig romlik: a v értékek bitjei a tapasztalat szerint gyakorlatilag függetlenek és egyenletes eloszlásúak, így nem tömöríthetőek, hiába is vonnánk be őket a Huffman-kódba.

A fenti példán bemutatva ezt az utolsó lépést, a tömörített képbe a $\{0,6\}$, $\{4,4\}$, $\{1,6\}$ és $\{0,4\}$ Huffman-kódja, és a 01011110001011000100 sorozat kerül.

Bár a baseline JPEG (a szabvány alapkódolása) nem engedi meg, de más, kiterjesztett módokban használhatunk aritmetikai kódolót is az $\{n,s\}$ párok kódolására. Ilyenkor általában a legjobb eredményeket adaptív aritmetikai kódoló használatával lehet elérni.

A JPEG egyszerűsége ellenére meglepően alacsony bitsebességeket tud elérni. A képpontonkénti 2 bites tömörítés egy 24 bites színmélységű kép esetén az eredetitől megkülönböztethetetlen képet eredményez — legalábbis az emberi szem számára nem látható a különbség. Igen kis, 1 bit/képpontos bitsebesség esetén kezd el zavaróvá válni a torzítás, és 0.5 bit/képpont alatt a kép nem élvezhető. A felismerhetőség még 0.086 bit/képpontos bitsebesség mellett is biztosítható. Természetesen ilyen alacsony bitsebességek esetén különböző kiterjesztések is szükségesek a szabványhoz. Mindenesetre ezek az eredmények nagyon jónak számítanak. A JPEG alacsony bitsebességek esetén legelőször „kockásodni” kezd. Ezt az alacsonyfrekvenciás együtthatók értékének kvantálás miatti torzulása okozza.

Moving Picture Experts Group (MPEG)

A Moving Picture Experts Group (MPEG) a JPEG-hez hasonlóan az ISO egyik munkacsoportja (ISO/IEC JTC 1 / SC 29 / WG 11). Feladatuk olyan video-

tömörítési szabványok kidolgozása volt, melyek széleskörű ipari konszenzuson alapulnak. Az ilyen jellegű szabványokra a digitális konvergencia időszakában mind a tartalom előállítóknak, mind a felhasználóknak, mind a közöttük álló szolgáltatóknak szükségük van. Az MPEG egy öt szabványból álló szabványcsomag kidolgozását tűzte ki célul, mely a veszteséges videotömörítés (beleértve a hangtömörítést is), a digitális televíziózás és a multimédiás alkalmazások széles körét fedi le. Az öt szabvány: MPEG-1, MPEG-2, MPEG-4, MPEG-7 és MPEG-21.

A CD-olvasók első generációja a hifi minőségű tömörítetlen zenéhez szükséges, 1.4 Mbit/s lejátszási sebességre volt képes. Az MPEG-1 szabvány kialakításakor azt tűzték ki célul, hogy a fenti sebességű, egyszeres ($1\times$) CD-meghajtók olvasási sebességén használható formátumot hozzanak létre. Ez nehéz feladat volt, hiszen egy 8 biten mintavételezett video (NTSC vagy PAL) tömörítetlenül nagyságrendileg 200 Mbit/s átviteli sebességet igényel. A szükséges tömörítés tehát nagyjából $1 : 140$ kell hogy legyen, sőt még a hangcsatorná(k)nak és a hibajavító kódnak is helyet kell szorítani. A nagy tömörítési igény mellett a „véletlen elérés” lehetőségét is teljesíteni kellett, vagyis azt, hogy a tárolt video bármelyik részét (akár a közepét is) gyorsan el lehessen érni. Amennyiben prediktív algoritmussal tömörítünk, problémát okozhat ennek a kritériumnak a megvalósítása, hiszen ilyenkor csak a kezdeti, biztos ponttól elindulva lehetséges a rekonstrukció. Viszont a prediktív algoritmusok kizárása szóba sem jöhetett, hiszen ezek nélkül remény sincs ilyen tömörítés elérésére. Az MPEG által választott megoldás a rövid, de egymástól független prediktív blokkok alkalmazása volt. Az MPEG-1 — bár egy kimagaslóan jó videotömörítési algoritmus — az $1\times$-es CD-ROM-ok sebességének megfelelő, igen erős tömörítést csak viszonylag gyenge képminőség mellett tudja megvalósítani. Azonban 1992-ben, amikor a szabvány megjelent, az ipar ezzel a minőséggel is megelégedett. A szabvány sikeres alkalmazásai többek között a CD-I és a Video-CD technológiák.

Az MPEG-2 szabványt a digitális televíziózáshoz fejlesztették ki. Ehhez az MPEG-1 segítségével elérhetőnél jobb minőségre volt szükség, támogatja például a szabvány a HDTV video tömörítését is. A HDTV technológia, a High Definition TeleVision a szokványos televízióminőségnél nagyobb felbontású és színmélységű, CD-minőségű hanggal kísért televíziós szabvány. Az MPEG-2 által igényelt bitsebesség (a minőség függvényében) 1–40 Mbit/s között alakul. Jellegzetes értékek a 4–6 Mbit/s hagyományos (PAL) videojel esetén, és a 12–20 Mbit/s HDTV videojel esetén. Digitális televíziózás során gondoskodni kell a tömörítésen kívül a jelátvitelről, multiplexelésről is. Az MPEG-2 szabvány egyik sikeres alkalmazása, az Európában szabványosított DVB (Digital Video Broadcasting) technológia. Ez a digitális műsorszórást teszi lehetővé, hagyományos földi (8 MHz sávszélességű) televíziócsatornában QPSK vagy COFDM modulációval, illetve

műholdas (36 MHz sávszélességű) csatornában BPSK modulációval. Mindkét esetben 3–6 televízióműsor multiplex átvitelére van lehetőség. Az MPEG-2 további sikeres alkalmazása a VoD (Video on Demand), és a számítástechnikában egyre népszerűbb DVD is.

Az MPEG-4 a multimédia alkalmazások szabványa. Közös technológiai alapot nyújt a műsorszórásos, az interaktív és a beszélgetés-jellegű szolgáltatásokhoz. Sokrétű interaktivitást tesz lehetővé a hagyományos lejátszás–megállítás–előre/visszatekerés mellett. Képes szintetikus (pl. számítógép-animáció) és természetes forrásból származó (pl. videokamera által rögzített) információtípusok együttes kezelésére. A bitsebességek igen széles skáláját támogatja, egészen 10 kbit/s-tól kezdve. Többek között lehetséges MPEG-4-gyel egy videokonferencia 64 kbit/s-os tömörítése és a professzionális HDTV tömörítése is 40 Mbit/s sebességgel. Bár a szabvány már teljesen kész, egyes bővítésein jelenleg is dolgoznak. Az MPEG-4 egyik sikeres alkalmazása a DivX.

Az MPEG-7 a többi MPEG szabvány által kódolt információtípusok katalogizálására, címkézésére fog lehetőséget adni, még fejlesztés alatt áll. Az MPEG-21 szabvány egy egységes multimédia keretrendszer lesz.

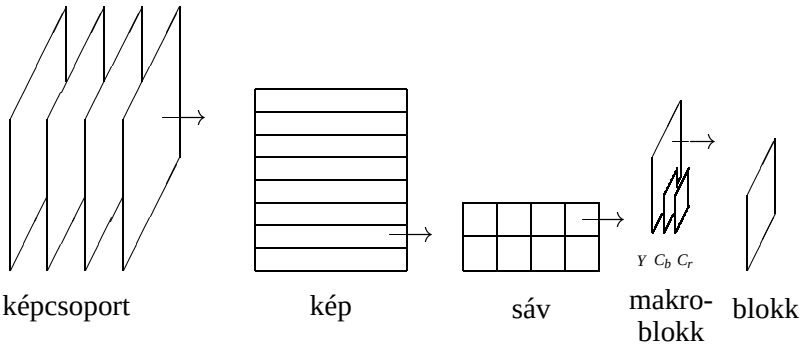
A továbbiakban az MPEG-1 szabvány videotömörítési részének vázát fogjuk áttekinteni. Fontos megjegyezni, hogy az MPEG-1 emellett foglalkozik a hangtömörítéssel, valamint a video- és hangfolyamok multiplexelésével is. A hangtömörítési részről a 2.9. szakaszban már volt szó.

A szabvány kidolgozása során kiemelt figyelmet fordítottak az implementáció szabadságára. A videotömörítési részében az MPEG-1 szabvány csak a tömörített bitfolyam szintaxisát és értelmezését írja le. A kódoló tetszőleges algoritmussal dolgozhat, de természetesen csak szintaktikailag helyes bitfolyamot állíthat elő. Nagyon sok múlik azon — mind képminőség, mind pedig bitsebesség tekintetében — hogy egy kódoló mennyire jól tudja kihasználni az MPEG-1 szabvány adta lehetőségeket. A dekódoló implementálása terén a bitfolyam kötött értelmezése miatt nincs nagy szabadság. A szabvány ad is egy referencia dekódert, bár ennek használata nem kötelező.

A 2.17. ábrán a bitfolyam szabvány által definiált szintaktikai felépítése látható. A legkülső elem a **videoszekvencia** (ezt nem tüntettük fel az ábrán). Egy videoszekvenciának adott képmérete, képsebessége és egyéb jellemző paraméterei vannak. A videoszekvencia fejléce ezek leírását tartalmazza.

A videoszekvencia belsejében **képcsoportok** (Group Of Pictures, GOP) vannak, amelyek néhány, közelebbről meg nem határozott számú kép egymásutánjából állnak. Ennek jelentőségét később fogjuk megérteni, most csak annyit, hogy a képcsoportok az egymástól függetlenül kódolt prediktív egységek.

A képcsoporton belül **képek** helyezkednek el. A kép háromféle lehet, I, P



2.17. ábra. Az MPEG bitfolyam szintaktikai felépítése.

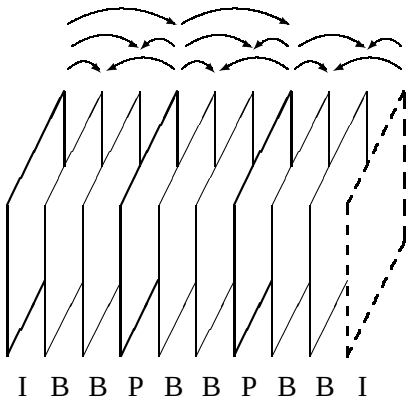
vagy B. Ez az alkalmazott predikció fajtájával van összefüggésben. Ezt is később tárgyaljuk.

A képek **sávok**ra vannak bontva. A sávok elején egy különleges, máshol elő nem forduló bitminta található, mely a szinkronizációra ad lehetőséget adatátviteli hibák esetén.

A sávok **makroblokkok**ra vannak bontva. Egy makroblokk a kép egy 16×16 képpont méretű részét írja le.

Egy makroblokk 6 **blokk**ból áll. Mindegyik blokk egy 8×8 elemű mátrix. Ennek az a magyarázata, hogy a képek az (Y', C_b, C_r) színkoordináta-rendszer szerint vannak tárolva. A krominancia-együtthatókat (C_b és C_r) vízszintesen és függőlegesen is egy kettes faktorral csökkentett felbontásban tároljuk, az emberi szem érzékenységéről leírtakkal összhangban. Így tehát egy 16×16 képpont méretű rész leírásához szükség van egy 16×16-os luminancia (Y') mátrixra, és két 8×8-as krominancia (C_b, C_r) mártixra. A 16×16-os luminancia mátrixot 4 darab 8×8-as mátrixra bontjuk, így jön ki a makroblokkonkénti 6 blokk.

A tömörítési algoritmus lelke a mozgásbecslés. Ez tulajdonképpen egy predikciós kódolás, ahol a video adatfolyam egymás utáni képeinek hasonlóságát használjuk ki. A predikciót makroblokk szinten végezzük: minden makroblokkhoz kikeressük az időben megelőző és az időben következő képeken található, hozzájuk leginkább hasonló részleteket. A megelőző és a következő képeken a hasonló részek helyét az úgynevezett mozgásvektorral határozzuk meg. A makroblokknak csak ezen referencia-képrészektől való eltérését (vagyis a predikció hibáját) kódoljuk. Ezt a hibát vágjuk a fent bemutatott módon 6 blokkra, és ezután a blokkokat a JPEG módszernél megismert DCT $\longrightarrow$ kvantálás $\longrightarrow$ futamhossz kódolás $\longrightarrow$ Huffman-kódolás sémának megfelelően tömörítjük tovább. Minden makroblokk kódolásához tehát két mozgásvektort, és 6 JPEG kódolású blokkot adunk meg.



2.18. ábra. Az MPEG különböző predikciótípusai.

Valójában az eljárás kicsit árnyaltabb a fentinél. Egy makroblokk tömörítése során ugyanis négyféle módon járhatunk el. Tömöríthetjük predikció nélkül, vagy predikcióval hátulról, elölről, és mindkét irányból. Amennyiben nem alkalmaztunk predikciót, úgy az eredeti makroblokk 6 blokkját kell JPEG-szerűen kódolnunk. Amennyiben egyirányú predikciót alkalmaztunk (időben megelőző vagy következő kép egy részét jelöltük ki), akkor a makroblokkból levonjuk a kijelölt részt, és a különbséget tömörítjük. Amennyiben kétirányú predikciót hajtottunk végre (időben megelőző és következő kép egy-egy részletét is felhasználjuk), akkor a makroblokkból a JPEG tömörítés előtt a két kijelölt rész átlagát vonjuk le.

Az elölről és a kétirányból alkalmazott predikciónál gondoskodni kell az egyértelmű dekódolhatóságról. Ezt az MPEG-1 szabvány azzal oldja meg, hogy kizárja, hogy a képek körkörösen egymásra hivatkozzanak. Ennek biztosítására minden egyes képben meg van kötve, hogy a benne szereplő makroblokkok kódolása során milyen típusú predikciót használhatunk. Ezt mutatjuk be a 2.18. ábrán.

I típusú képben egyáltalán nem alkalmazhatunk predikciót, azaz egy I kép egy önálló kép, JPEG típusú tömörítéssel. Így egy I típusú képet tehát minden egyéb kép nélkül tudunk dekódolni.

A P típusú képek makroblokkjai lehetnek predikció nélkül kódolva, vagy pedig predikcióval hátulról, a hozzájuk legközelebb levő I vagy P típusú képből. (Tehát a makroblokkok kódolásának további finomsága, hogy a predikció nem az időben közvetlenül megelőző kép alapján történik, hanem a közvetlenül megelőző I vagy P kép alapján.) Egy P típusú kép dekódolásához ezek szerint legfeljebb az előtte levő utolsó I képig kell visszaszaladnunk, és innentől kezdve P képről P képre tudunk lépkedni.

A harmadik fajta kép, a B kép bármilyen kódolású makroblokkokat tartalmazhat, de a makroblokkok csak a legközeleb eső megelőző vagy következő P vagy I képekre hivatkozhatnak. Sem B képekre, sem a legközelebbi I vagy P képnél távolabbi képekre nem lehet hivatkozni. A lényegi különbség a B és P képek között az, hogy B képekénél jövőbeli P vagy I képre is hivatkozhatunk.

Természetesen, a szabvány koncepciójának megfelelően a képek sorrendje nem kötött, akárhogyan jöhetnek egymás után a P, I és B képek, ahogy a kódoló akarja. A fenti szabályok betartásával lehetséges a dekódolás, mert egy I típusú kép magában dekódolható, egy P típusú az előtte levő, hozzá legközelebbi I képtől indulva dekódolható, míg egy B kép az őt körülvevő I vagy P képek alapján dekódolható — tehát minden kép dekódolható.

Mindemellett hogy nem kötelező, kialakultak szokványos képsorrendek. A legsűrűbben egy I képre két P kép épül, és közöttük két-két B kép van. A 2.18. ábrán egy ilyen képsorrend látható.

A képek sorrendje a bitfolyamban nem egyezik meg a képek idősorrendjével. A bitfolyambeli sorrend úgy lett meghatározva, hogy ha sorban olvassuk be a bitfolyamból a képeket, akkor egy kép beolvasásának időpontjában az összes szükséges információ rendelkezésre álljon a dekódoláshoz. Ezt úgy oljda meg a szabvány, hogy hátraküldi a B képeket az utánuk következő I vagy P kép mögé. Ezáltal a B képek időben előre mutató referenciája a bitfolyamban hátramutató referencia lesz. Természetesen átrendezéskor az I és P képek a helyükön maradnak, és a B képek egymáshoz viszonyított sorrendjén sem változtatunk. Tehát, ha az időbeli sorrendet az indexekkel jelöljük, akkor a

$$\ldots B_{-2}B_{-1}I_0B_1B_2P_3B_4B_5P_6B_7B_8I_9B_{10}B_{11}P_{12}\ldots$$

sorozatból az átrendezés után az

$$\ldots I_0B_{-2}B_{-1}P_3B_1B_2P_6B_4B_5I_9B_7B_8P_{12}B_{10}B_{11}\ldots$$

sorozat lesz.

Egy képcsoport mindig egy I képpel kezdődik, méghozzá a bitfolyambeli sorrend szerint. Emiatt egy képcsoport bitfolyamban első képe önmagában dekódolható, vagyis nem szükséges régebben dekódolt képek ismerete a dekódolásához. A képcsoport további képei is mind dekódolhatóak a bitfolyambeli sorrendben. (Természetesen ezek dekódolása során szükségünk lesz az ezen képcsoportból előttük dekódolt képekre.) Kivételt képeznek az első P képet megelőző B képek, ugyanis ezek a megelőző képcsoport utolsó P vagy I képére is hivatkozhatnak. (Ne felejtsük el, hogy a bitfolyambeli sorrendben definiáljuk a képcsoportot!) Ezek a B képek a megelőző képcsoport dekódolása nélkül nem dekódolhatóak. Emiatt

az első néhány nem dekódolható B kép miatt hívják nyíltnak az ilyen képcsoportot. (A szabvány definiál egy zárt képcsoportot is, természetesen ehhez más képsorrend szükséges.) Ezáltal a képcsoport a bevezetőben említett viszonylag rövid prediktív egységet testesíti meg, hiszen (nyílt csoport esetén néhány B kép kivételével) a képcsoportok egymástól függetlenül dekódolhatóak.

Az MPEG videokódolás, amennyiben ezt nem szabályozzuk külön, változó bitsebességet fog eredményezni. Gondoljunk például arra, hogy ahol egy teljesen új kép jelenik meg a videofolyamban (vágás), ott az első kép makroblokkjait nem tudjuk predikcióval tömöríteni, és így gyakorlatilag egy teljes képet kell JPEG-gel kódolnunk. Ez igen sok bitet igényelhet. Ott viszont ahol egy állókép van a videofolyamban, szinte nem is lesz predikciós maradék, vagyis nagyon rövid lesz a kód. Az átvitelhez használt csatorna bitsebességét (MPEG-1 esetén a CD lejátszó 1.4 Mbit/s sebessége, MPEG-2 esetén pl. az egy televízióadásra jutó adatfolyam 5 Mbit/s sebessége) a kódolás során természetesen nem léphetjük át. Ezen a problémán pufferek felhasználásával valamennyit lehet segíteni, de meg kell oldani, hogy a dekódoló puffere sohase ürüljön ki, és sohase csorduljon túl, továbbá a kódoló is csak a csatorna átviteli képességének megfelelő mennyiségű adatot hozzon létre. Ennek eléréséhez a kódoló állandóan figyeli a saját pufferét, és nyomon követi a dekódolóét is. (Ez utóbbihoz egyes esetekben a dekódoló szimulációjára kényszerül.) Ha például a kódoló puffere kezd megtelni — ezt a képsorozat információdússága okozhatja, — csökkentenie kell a képenként átvitt bitek mennyiségét. Ezt a JPEG kvantálási lépcső megemelésével érheti el, természetesen a minőség rovására. Durvább megoldás, ha bizonyos makroblokk predikciós maradékát egyszerűen elhagyja a kódból, és csak a mozgásvektorokat adja meg. Még durvább lehetőségek is rendelkezésére állnak a kódolónak. Ilyen például a bemenetén érkező kép információtartalmának korlátozása, például a képfrekvencia vagy a felbontás csökkentésével.

2.11. Forráskódolás betűnkénti hűségkritériummal

Ebben a szakaszban áttekintjük a veszteséges (torzítást megengedő) forráskódolás elvi határait. A vizsgálat során kulcsszerepet játszik a kölcsönös információ.

2.3. definíció. *Az X és Y diszkrét valószínűségi változók* ***kölcsönös információ****ján az*

$$I(X;Y) = H(X) + H(Y) - H(X,Y)$$

mennyiséget értjük.

A definíció mutatja, hogy a kölcsönös információ szimmetrikus, és

$$I(X;Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X) = I(Y;X).$$

Két valószínűségi vektorváltozó kölcsönös információja a fenti definícióból értelemszerűen következik.

2.8. tétel (A kölcsönös információ tulajdonságai).

a)

$$\begin{aligned} I(X;Y) &= \sum_{x,y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} = \\ &= \sum_{x,y} p(x,y)\log\frac{p(x\mid y)}{p(x)} = \\ &= \sum_{x,y} p(x,y)\log\frac{p(y\mid x)}{p(y)}. \end{aligned}$$

b)

$$I(X;Y)\geq 0.$$

c)

$$I(X;Y)\leq H(X),$$

$$I(X;Y)\leq H(Y).$$

d) Az X és Y bármely g és f függvényére

$$I(X;Y)\geq I(g(X);f(Y)).$$

BIZONYÍTÁS: Az a), b) és c) tulajdonságok a definícióból és a (feltételes) entrópia tulajdonságaiból közvetlenül adódnak. A d) tulajdonság a feltételes entrópia 1.7. d) tulajdonságából következik:

$$\begin{aligned} I(X;Y) &= H(X)-H(X\mid Y)\geq \\ &\geq H(X)-H(X\mid f(Y)) = \\ &= I(X;f(Y)) = \\ &= H(f(Y))-H(f(Y)\mid X)\geq \\ &\geq H(f(Y))-H(f(Y)\mid g(X)) = \\ &= I(g(X);f(Y)). \end{aligned}$$

■

Tegyük fel, hogy egy $\mathbb{X}$ információforrás k hosszú blokkját, $X_1X_2\ldots X_k$-t egy kódolással egy $Y_1Y_2\ldots Y_k$ k-hosszú blokkal reprezentáljuk, ahol az X_i-k és Y_i-k a

véges $\mathcal{X}$ illetve $\mathcal{Y}$ halmazokból veszik értékeiket. Legyen $\mathcal{X}\times\mathcal{Y}$-on adva egy d : $\mathcal{X}\times\mathcal{Y}\to\mathbb{R}^+$ nemnegatív függvény. A d függvényt **torzítási mérték**nek nevezzük, mert minden $x\in\mathcal{X}$, $y\in\mathcal{Y}$ párra $d(x,y)$ azt a torzítást méri, amit az okoz, hogy az x forrásbetűt y-nal reprezentáljuk. Az $\mathbf{x}=x_1\dots x_k$ és $\mathbf{y}=y_1\dots y_k$ $(x_i\in\mathcal{X},y_i\in\mathcal{Y})$ blokkok közti torzítást a

$$d(\mathbf{x},\mathbf{y})=\frac{1}{k}\sum_{i=1}^{k}d(x_i,y_i)$$

nemnegatív mennyiség méri. Ezt nevezzük **betűnkénti torzítás**nak. Ha az $\mathbb{X}$ forrás $\mathbf{X}=(X_1,\dots,X_k)$ blokkját az $\mathbf{Y}=(Y_1,\dots,Y_k)$ blokk reprezentálja, akkor a köztük levő $\mathbf{E}(d)$ átlagos betűnkénti torzítást az

$$\mathbf{E}(d)=\mathbf{E}(d(\mathbf{X},\mathbf{Y}))=\sum_{\mathbf{x},\mathbf{y}}p(\mathbf{x},\mathbf{y})d(\mathbf{x},\mathbf{y})$$

várható érték méri, ahol $p(\mathbf{x},\mathbf{y})=\mathbf{P}\{\mathbf{X}=\mathbf{x},\mathbf{Y}=\mathbf{y}\}$ és az összegzés az összes $\mathbf{x}\in\mathcal{X}^k$ és $\mathbf{y}\in\mathcal{Y}^k$ vektorra kiterjed.

2.6. példa. Legyen $\mathcal{X}=\mathcal{Y}$, vagyis a forrás- és a reprodukciós ábécé ugyanaz, és legyen d az ún. Hamming-torzítás:

$$d(x,y)=\begin{cases}0, & \text{ha } x=y\\ 1, & \text{ha } x\neq y\end{cases}$$

vagyis, ha a reprodukció tökéletes, akkor a torzítás nulla, egyébként pedig egy. Ekkor

$$\mathbf{E}(d(\mathbf{X},\mathbf{Y}))=\mathbf{E}\left(\frac{1}{k}\sum_{i=1}^{k}d(X_i,Y_i)\right)=\frac{1}{k}\sum_{i=1}^{k}\mathbf{E}(d(X_i,Y_i)).$$

A d definíciója szerint

$$\mathbf{E}(d(X_i,Y_i))=\mathbf{P}\{X_i\neq Y_i\},$$

tehát

$$\mathbf{E}(d(\mathbf{X},\mathbf{Y}))=\frac{1}{k}\sum_{i=1}^{k}\mathbf{P}\{X_i\neq Y_i\},$$

vagyis $\mathbf{E}(d)$ az egyes pozíciókban történő karaktertévesztések valószínűségeinek átlaga. $\mathbf{E}(d)$-t nem szabad összetévesztenünk a 2.1. szakaszban vizsgált blokkhiba-valószínűséggel.

2.7. példa. Legyen $\mathcal{X}=\mathcal{Y}=\mathbb{R}$, és $d(x,y)=(x-y)^2$, ekkor

$$\mathbf{E}\,(d(\mathbf{X},\mathbf{Y}))=\mathbf{E}\|\mathbf{X}-\mathbf{Y}\|^2,$$

vagyis a kvantálásnál használt négyzetes torzítást kapjuk.

Most, hogy a betűnkénti torzítás fogalmát bevezettük, megadhatunk egy ilyen hűségkritériumot. Mondhatjuk például azt, hogy az $X_1 \dots X_k$ blokk reprodukciójaként elfogadjuk $Y_1 \dots Y_k$ blokkot, ha egy adott $\delta > 0$ számra

$$\mathbf{E}\left(\frac{1}{k}\sum_{i=1}^{k} d(X_i, Y_i)\right) < \delta.$$

Ezt nevezzük **betűnkénti hűségkritérium**nak.

A 2.6. példa szerint ez pl. azt jelentheti, hogy egy $0 < \delta < 1$ számra biztosítani szeretnénk, hogy a karakterek meghibásodásának átlagos valószínűsége δ-nál kisebb legyen.

A forrás adott hűségű kódolására forráskódokat használunk. Az $\mathbb{X}$ forrás k-hosszú blokkjait kódoló forráskódja egy $g : \mathcal{X}^k \to \mathcal{Y}^k$ leképezés, amely tehát minden $\mathbf{x} \in \mathcal{X}^k$ üzenethez egy $\mathbf{y} \in \mathcal{Y}^k$ reprezentánst rendel. Az előbbiek szerint a forráskód betűnkénti átlagos torzítása a

$$D(g) = \mathbf{E}(d(\mathbf{X}, g(\mathbf{X}))) = \sum_{\mathbf{x}} p(\mathbf{x}) d(\mathbf{x}, g(\mathbf{x}))$$

mennyiség. Ha a g forráskód értékkészleteként M darab vektort használ az $\mathcal{Y}^k$ vektorai közül, akkor a g **jelsebesség**e az

$$R = \frac{\log M}{k}$$

szám lesz, hiszen a g lehetséges M értékét $\log M$ biten, forrásbetűnként $\frac{\log M}{k} = R$ biten lehet kódolni. A forráskódolás célja az, hogy a forrást adott hűségű, minél kisebb jelsebességű kóddal kódoljuk, hiszen a jelsebesség az átvitel illetve tárolás „költségét” jelenti. Ha a g értékkészlete a $\mathcal{C} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_M\}$, ($\mathbf{y}_i \in \mathcal{Y}^k$, $i = 1, \dots, M$) halmaz, akkor a legkisebb torzítás eléréséhez g a következő kell legyen:

$$g(\mathbf{x}) = \mathbf{y}_i, \quad \text{ha} \quad d(\mathbf{x}, \mathbf{y}_i) \le d(\mathbf{x}, \mathbf{y}_j), \quad j = 1, 2, \dots, M,$$

vagyis az $\mathbf{x} \in \mathcal{X}^k$ kódja az az $\mathbf{y} \in \mathcal{C}$ amely a legjobban „hasonlít” $\mathbf{x}$-re. A g leképezés tehát egy vektorkvantáló. Ha a d hűségkritériumot egyfajta távolságnak tekintjük, akkor azt mondhatjuk, hogy $g(\mathbf{x})$ az $\mathbf{x}$ legközelebbi szomszédja a $\mathcal{C}$ halmazból. Jó kód keresése ezek szerint a megfelelő $\mathcal{C}$ halmaz keresésével ekvivalens. A továbbiakban a forráskódok teljesítőképességének elvi határait vizsgáljuk emlékezetnélküli stacionárius forrásra.

A következőkben bevezetjük az emlékezetnélküli stacionárius forrás R-D (rate-distortion) függvényét, amelyről kiderül, hogy megadja azt a legkisebb jelsebességet, amit egy legfeljebb δ torzítású forráskóddal el lehet érni.

2.4. definíció. *Legyen adva az $\mathcal{Y}$ reprodukciós ábécé és a d torzítási mérték. Ekkor az $\mathbb{X}$ emlékezetnélküli stacionárius forrás $\delta \geq 0$ számokra értelmezett* ***R-D függvény****e a következő:*

$$R(\delta) = \min\{I(X;Y) : \mathbf{E}(d(X,Y)) \leq \delta\},$$

ahol a minimumot az összes olyan (X,Y) valószínűségi változópár fölött kell venni, ahol X eloszlása megegyezik az X_i-k (közös) eloszlásával és Y az $\mathcal{Y}$ halmazból veszi értékeit. Ha nincs olyan Y, amellyel $\mathbf{E}(d(X,Y)) \leq \delta$, akkor legyen $R(\delta) = \infty$.

MEGJEGYZÉS:

a) Vegyük észre, hogy ha a $\delta_{\min} \geq 0$ számot a

$$\delta_{\min} = \sum_x p(x) \min_y d(x,y)$$

kifejezéssel definiáljuk, akkor bármely Y-ra

$$\mathbf{E}(d(X,Y)) \geq \delta_{\min}$$

hiszen

$$\begin{aligned} \mathbf{E}(d(X,Y)) &= \sum_{x,y} p(x,y)d(x,y) \geq \\ &\geq \sum_{x,y} p(x,y) \min_y d(x,y) = \\ &= \sum_x p(x) \min_y d(x,y). \end{aligned}$$

Ezek szerint $R(\delta) < \infty$ akkor és csak akkor, ha $\delta \geq \delta_{\min}$.

b) Mivel az X eloszlása rögzített, az (X,Y) pár eloszlását a $p(y \mid x) = \mathbf{P}\{Y = y \mid X = x\}$ felételes eloszlások határozzák meg $p(y,x) = p(x)p(y \mid x)$ szerint. Figyelembe véve a kölcsönös információ definícióját, az $R(\delta)$ definíciója, a $p(y) = \sum_{x' \in \mathcal{X}} p(x')p(y \mid x')$ egyenlőséget felhasználva a következőképpen írható át:

$$R(\delta) = \min \left\{ \sum_{\substack{x \in \mathcal{X} \\ y \in \mathcal{Y}}} p(x)p(y \mid x) \log \frac{p(x)p(y|x)}{p(x) \sum\limits_{x' \in \mathcal{X}} p(x')p(y \mid x')} \right\},$$

ahol a minimumot az összes olyan $p(y \mid x)$ feltételes eloszlás fölött vesszük, amelyekre $\sum_{x,y} p(x)p(y \mid x)d(x,y) \leq \delta$.

A következőkben megvizsgáljuk az $R(\delta)$ függvény néhány alapvető tulajdonságát.

2.3. lemma. *$R(\delta)$ a δ monoton fogyó és konvex függvénye, ha $\delta \geq \delta_{min}$.*

BIZONYÍTÁS: Az $R(\delta)$ definíciójából közvetlenül látszik, hogy $R(\delta)$ monoton fogy, hiszen ha $\delta' < \delta''$ akkor a feltételes eloszlások halmaza, amely felett a minimumot vesszük, δ' esetében szűkebb, mint δ'' esetében, tehát $R(\delta') \geq R(\delta'')$.

Legyen most $0 < \lambda < 1$ és $\delta_1, \delta_2 \geq \delta_{\min}$. Azt kell belátnunk, hogy

$$R(\lambda\delta_1 + (1-\lambda)\delta_2) \leq \lambda R(\delta_1) + (1-\lambda)R(\delta_2).$$

Legyenek $p_1(y \mid x)$ és $p_2(y \mid x)$ azon feltételes eloszlások, amelyekre az $R(\delta)$ definíciójában $I(X;Y)$ eléri minimumát, $R(\delta_1)$-et illetve $R(\delta_2)$-t. Jelöljük a $p_1(y \mid x)$ és $p_2(y \mid x)$ feltételes eloszlások által meghatározott valószínűségi változókat Y_1-gyel illetve Y_2-vel. Ekkor

$$I(X;Y_i) = R(\delta_i),$$

és

$$\mathbf{E}(d(X;Y_i)) \leq \delta_i, \qquad i = 1,2.$$

Definiáljunk egy új $p(y \mid x)$ feltételes eloszlást a következőképp:

$$p(y \mid x) = \lambda p_1(y \mid x) + (1-\lambda)p_2(y \mid x), \qquad x \in \mathcal{X},\ y \in \mathcal{Y}.$$

Ha Y az a valószínűségi változó, melynek feltételes eloszlása X-re $p(y \mid x)$, akkor könnyen belátható, hogy minden $y \in \mathcal{Y}$-ra

$$\begin{aligned} p(y) &= \mathbf{P}\{Y = y\} = \\ &= \lambda\mathbf{P}\{Y_1 = y\} + (1-\lambda)\mathbf{P}\{Y_2 = y\} = \\ &= \lambda p_1(y) + (1-\lambda)p_2(y), \end{aligned}$$

valamint

$$\begin{aligned} \mathbf{E}(d(X;Y)) &= \sum_{x,y} p(x)\left(\lambda p_1(y \mid x) + (1-\lambda)p_2(y \mid x)\right)d(x,y) = \\ &= \lambda\sum_{x,y} p(x)p_1(y \mid x)d(x,y) + (1-\lambda)\sum_{x,y} p(x)p_2(y \mid x)d(x,y) = \\ &= \lambda\mathbf{E}(d(X;Y_1)) + (1-\lambda)\mathbf{E}(d(X;Y_2)) \leq \\ &\leq \lambda\delta_1 + (1-\lambda)\delta_2. \end{aligned}$$

Mivel tehát $\mathbf{E}(d(X;Y)) \leq \lambda\delta_1 + (1-\lambda)\delta_2$ teljesül, az $R(\delta)$ definíciójából következik, hogy

$$I(X;Y) \geq R(\lambda\delta_1 + (1-\lambda)\delta_2). \tag{2.30}$$

Másrészt, a Jensen-egyenlőtlenség 1.2. b) következményéből
$p_i(x,y) = p(x)\,p_i(y \mid x),\ i = 1,2$ jelöléssel

$$(\lambda p_1(x,y) + (1-\lambda)p_2(x,y)) \log \frac{\lambda p_1(x,y) + (1-\lambda)p_2(x,y)}{p(x)[\lambda p_1(y) + (1-\lambda)p_2(y)]} \leq$$

$$\leq \lambda p_1(x,y) \log \frac{p_1(x,y)}{p(x)p_1(y)} + (1-\lambda)p_2(x,y)\frac{p_2(x,y)}{p(x)p_2(y)}. \qquad (2.31)$$

Ha (2.31) mindkét oldalát minden x-re és y-ra összegezzük, akkor azt kapjuk, hogy

$$I(X;Y) \leq \lambda I(X;Y_1) + (1-\lambda)I(X;Y_2) = \lambda R(\delta_1) + (1-\lambda)R(\delta_2),$$

amiből (2.30)-cal együtt a lemma állítása adódik. ■

2.4. lemma. *Amennyiben az $\mathbb{X}$ emlékezetnélküli stacionárius forrás $\mathbf{X} = (X_1, X_2, \ldots, X_k)$ blokkja és az $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_k)$ valószínűségi vektorváltozók kielégítik az*

$$\mathbf{E}\left(\frac{1}{k}\sum_{i=1}^{k} d(X_i, Y_i)\right) \leq \delta$$

hűségkritériumot, akkor

$$I(\mathbf{X};\mathbf{Y}) \geq kR(\delta).$$

BIZONYÍTÁS: A feltételes entrópia 1.7. e) és c) tulajdonságai szerint

$$H(\mathbf{X} \mid \mathbf{Y}) = \sum_{i=1}^{k} H(X_i \mid \mathbf{Y}, X_1, \ldots, X_{i-1}) \leq \sum_{i=1}^{k} H(X_i \mid Y_i).$$

Ezt és az X_i-k függetlenségét felhasználva azt kapjuk, hogy

$$\begin{aligned} I(\mathbf{X};\mathbf{Y}) &= H(\mathbf{X}) - H(\mathbf{X} \mid \mathbf{Y}) = \\ &= \sum_{i=1}^{k} H(X_i) - H(\mathbf{X} \mid \mathbf{Y}) \geq \\ &\geq \sum_{i=1}^{k} (H(X_i) - H(X_i \mid Y_i)) \\ &= \sum_{i=1}^{k} I(X_i;Y_i). \end{aligned} \qquad (2.32)$$

Ha $\delta_i = \mathbf{E}(d(X_i, Y_i))$, akkor a tétel feltétele szerint

$$\frac{1}{k}\sum_{i=1}^{k}\delta_i \leq \delta,$$

tehát (2.32) felhasználásával, mivel $R(\delta)$ monoton fogyó és konvex

$$\frac{1}{k}I(\mathbf{X};\mathbf{Y}) \geq \frac{1}{k}\sum_{i=1}^{k}I(X_i;Y_i) \geq \frac{1}{k}\sum_{i=1}^{k}R(\delta_i) \geq R\left(\frac{1}{k}\sum_{i=1}^{k}\delta_i\right) \geq R(\delta),$$

amivel a lemmát beláttuk. ■

Most bebizonyítjuk a forráskódolási tétel megfordítását, ami azt mondja ki, hogy egy δ-nál kisebb torzítású forráskód jelsebessége nem lehet kisebb $R(\delta)$-nál.

2.9. tétel (A forráskódolási tétel megfordítása). *Ha az $\mathbb{X}$ emlékezetnélküli és stacionárius forrás $g : \mathcal{X}^k \to \mathcal{Y}^k$ forráskódja, amely M különböző kódszót használ, valamely $\delta \geq \delta_{min}$ számra kielégíti a*

$$D(g) \leq \delta$$

hűségkritériumot, akkor a kód $R = \frac{\log M}{k}$ jelsebességére

$$R \geq R(\delta).$$

BIZONYÍTÁS: A kölcsönös információ 2.8. c) és az entrópia 1.6. a) tulajdonsága szerint

$$I(\mathbf{X};g(\mathbf{X})) \leq H(g(\mathbf{X})) \leq \log M.$$

Másrészt a 2.4. lemmából

$$I(\mathbf{X};g(\mathbf{X})) \geq kR(\delta),$$

tehát

$$\log M \geq kR(\delta),$$

amivel az állítást bebizonyítottuk. ■

A következő tétel, ami a fejezet főtétele, megmutatja, hogy $R(\delta)$ valóban a δ torzítással elérhető jelsebességet jelenti.

2.10. tétel (Forráskódolási tétel emlékezetnélküli stacionárius forrásokra). *Legyen $R(\delta)$ az $\mathbb{X}$ emlékezetnélküli stacionárius forrás R-D függvénye egy adott reprodukciós ábécére és d torzítási mértékre. Ekkor, ha $\delta \geq \delta_{min}$, akkor minden $\delta' > \delta$ és $R' > R(\delta)$ esetén elég nagy k-ra létezik egy, az $\mathbb{X}$ forrás k-hosszú blokkjait kódoló g forráskód M darab kódszóval, amelyre*

$$M \leq 2^{kR'}$$

és

$$D(g) < \delta'.$$

A tétel tehát azt mutatja, hogy elég hosszú blokkokat kódolva a kódsebesség alsó határa, $R(\delta)$, tetszőlegesen közelíthető δ-hoz tetszőlegesen közeli torzítású forráskóddal. A tételt nem bizonyítjuk, mivel a bizonyítás nem túl egyszerű. Annyit meg kell jegyezni, hogy a tétel bizonyítása egy, az információelméletben gyakran és sikeresen alkalmazott technikán, a véletlen kódoláson alapul. A véletlen kódolással a csatornakódolási tétel bizonyításakor találkozunk majd, amely bizonyítás meglehetősen hasonlít a forráskódolási tétel bizonyításához.

2.8. példa. Tekintsük Hamming-torzítást, ahol a forrásábécé és a reprodukciós ábécé megegyeznek, és

$$d(x,y) = \begin{cases} 0, & \text{ha } x = y, \\ 1, & \text{ha } x \neq y \end{cases}.$$

Legyen $\mathbb{X}$ emlékezetnélküli és stacionárius $\mathbf{P}\{X_i = x\} = \mathbf{P}\{X = x\}$ adott eloszlással. Számítsuk ki az $R(0)$ értékét!

Mivel

$$\mathbf{E}(d(X,Y)) = \mathbf{P}\{X \neq Y\},$$

ezért az $\mathbf{E}(d(X,Y)) \leq \delta$ feltétel $\delta = 0$ esetében a $\mathbf{P}\{X = Y\} = 1$ feltételt jelenti, vagyis ekkor Y 1-valószínűséggel meghatározza X-et. Ekkor viszont

$$I(X;Y) = H(X) - H(X \mid Y) = H(X),$$

vagyis

$$R(0) = H(X) = H(\mathbb{X}).$$

Ebben az esetben a forráskódolási tétel azt mondja ki, hogy tetszőleges kis δ pozitív számra találhatunk egy $g : \mathcal{X}^k \to \mathcal{X}^k$ forráskódot, hogy

$$\frac{1}{k} \sum_{i=1}^{k} \mathbf{P}\{X_i \neq Y_i\} < \delta,$$

ahol Y_i-vel a $g(X_1, \ldots, X_k)$ i-edik koordinátáját jelöltük, és a kód jelsebessége a forrás entrópiájához nagyon közeli szám, azaz körülbelül $2^{kH(\mathbb{X})}$ kódszót használ.

2.12. Feladatok

2.1. feladat (Fix szóhosszúságú kód). Legyen $\mathcal{X}$ egy stacionárius, memóriamentes bináris forrás, $\mathbf{P}\{X_1 = 1\} = 0.005$, $\mathbf{P}\{X_1 = 0\} = 0.995$ eloszlással. Azokhoz a 100 hosszúságú blokkokhoz rendelünk kódszavakat, amelyek legfeljebb három egyest tartalmaznak. Ha minden kódszó azonos hosszúságú, akkor mi az a minimális hosszúság, amellyel ez a kód megvalósítható? Mekkora a nem kódolt blokkok összvalószínűsége pontosan, és milyen becslést kapunk erre a számra a Csebisev-egyenlőtlenségből?

2.2. feladat (Normális eloszlás egy bites kvantálása). Legyen X nulla várható értékű, σ szórású normális eloszlású valószínűségi változó. Határozza meg az optimális egy bites (két szintű) kvantálót! Mennyi a torzítás?

2.3. feladat (Exponenciális eloszlás kvantálása). Legyen X valós valószínűségi változó, melynek sűrűségfüggvénye

$$f(x) = \begin{cases} c\,\mathrm{e}^{-\frac{1}{2}x}, & \text{ha } x \in [0,2] \\ 0, & \text{ha } x \notin [0,2] \end{cases},$$

ahol c olyan konstans, hogy f valószínűségi sűrűségfüggvény.

a) Kvantáljuk X-et egy $[0,2]$-re illeszkedő 4 bites (16 szintű) egyenletes Q_1 kvantálóval. A tanult közelítéseket használva számolja ki a kvantáló négyzetes torzítását és a $H(Q_1(X))$ entrópiát!

b) Legyen most

$$f(x) = \begin{cases} \frac{1}{2}\mathrm{e}^{-\frac{1}{2}x}, & \text{ha } x \geq 0 \\ 0, & \text{ha } x < 0 \end{cases}.$$

Kvantáljuk X-et a következőképpen:

$$Q(x) = \begin{cases} Q_1(x), & \text{ha } x \in [0,2] \\ 2, & \text{ha } x > 2 \end{cases}.$$

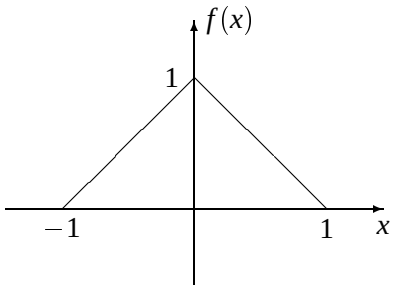
Az a) rész eredményét felhasználva számolja ki a $H(Q(X))$ entrópiát!
(Segítség: Legyen $Z = 0$, ha $X \in [0,2]$, és $Z = 1$, ha $X > 2$. Ekkor $H(Q(X)) = H(Q(X), Z) = H(Z) + H(Q(X) \mid Z)$.)

2.4. feladat. Az ábrán látható $f(x)$ sűrűségfüggvényű X valószínűségi változót 2 bites egyenletes kvantálóval kvantáljuk, mely illeszkedik a $[-1,1]$ intervallumra.

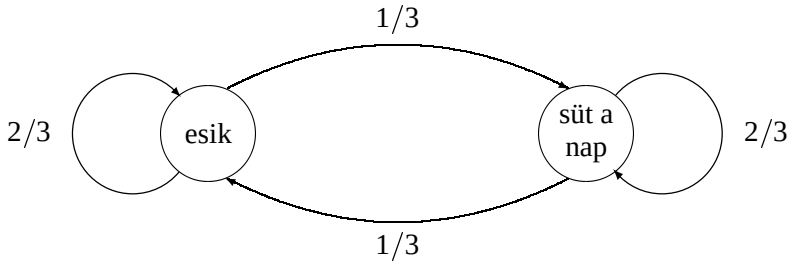
a) Számolja ki *pontosan* a kvantáló négyzetes torzítását és kimenetének entrópiáját!

b) Számolja ki a fenti két mennyiséget a tanult közelítéseket felhasználva! Mennyire egyeznek a pontos és közelítő értékek?

(Segítség: $\int_0^x t \ln t \, \mathrm{d}t = x^2 \left(\frac{\ln x}{2} - \frac{1}{4}\right)$.)



2.5. feladat. A napsütéses és esős napok stacionárius Markov-lánc szerint követik egymást az ábrán látható átmenetvalószínűségekkel. Az esős napokon az eső mennyisége exponenciális eloszlású, $f(x) = \mathrm{e}^{-x}$ $(x > 0)$ sűrűségfüggvénnyel (centiméterben mérve). Az időjárásjelentésben az eső mennyiségét 1 milliméter felbontásban egyenletesen kvantálva mondják be. Mennyi (közelítőleg) a csapadékmérések sorozatának mint forrásnak az entrópiája?



2.6. feladat (A Lloyd–Max-algoritmus nem optimális). Mutasson példát arra, hogy a Lloyd–Max kvantálótervező algoritmus nem mindig a minimális torzítású kvantálóhoz konvergál. Azaz adjon meg egy olyan „rossz” sűrűségfüggvényt és kiindulási kvantálót, hogy az algoritmus biztosan ne az optimumhoz konvergáljon.

2.7. feladat (Maximális differenciális entrópia). Mutassa meg, hogy

- az adott várható értékű, nemnegatív valószínűségi változók között az exponenciális eloszlásúnak maximális a differenciális entrópiája;
- az $[a, b]$ intervallumon kívül nulla sűrűségfüggvényű valószínűségi változók között az $[a, b]$-n egyenletes eloszlásúnak maximális a differenciális entrópiája.

2.8. feladat (Maximális entrópia). Tekintsük a természetes számok halmazára koncentrálódó, m várható értékű diszkrét eloszlásokat. Mutassa meg, hogy a geometriai eloszlásnak maximális az entrópiája.

2.9. feladat (Differenciális entrópia). Legyen X abszolút folytonos eloszlású valószínűségi változó, és jelöljük $H(X)$-szel a differenciális entrópiáját. Mutassa meg, hogy bármely $a > 0$ számra

$$H(aX) = H(X) + \log a.$$

3. fejezet

Csatornakódolás

Ebben a fejezetben a zajos csatornán történő megbízható információátvitel problémájával foglalkozunk.

Tekintsük az 1.1. ábrán látható hírközlési modellt. Célunk a forrás által kiválasztott üzenetet a nyelőbe eljuttatni. Felhívjuk a figyelmet arra, hogy valóságos hírközlési problémák vizsgálatánál nem mindig egyértelmű, hogy hol kell a kódolót és a csatornát, illetve a csatornát és a dekódolót szétválasztani, azaz, hogy az esetleg jelenlévő modulátort és a demodulátort a csatorna avagy a kódoló, illetve dekódoló részének tekintsük-e. Ebben a fejezetben, mint látni fogjuk, csak az ún. diszkrét csatornával foglalkozunk, vagyis a modulációt és a demodulációt a csatorna részének tekintjük. Ez annak a tervezői szemléletnek felel meg, amely szerint a modulátort és demodulátort nem akarjuk, vagy nem tudjuk megváltoztatni.

Alapvetően két problémával kell szembenéznünk. Az egyik az, hogy az átvivő közeg — a csatorna — bemeneti ábécéje nem feltétlenül egyezik meg a forrásábécével, a másik pedig az, hogy az átvivő közeg zajos, azaz a csatorna bemenetére adott szimbólum nem feltétlenül egyezik meg a csatorna kimenetén megjelenővel. Ezen nehézségek legyőzésének eszköze a csatornakódolás, amelynek elvi lehetőségeit a csatornakódolási tétel mutatja meg.

A zajos csatornát jól modellezi az a feltevés, hogy a csatorna bemenetére adott üzenet egyes szimbólumai bizonyos valószínűséggel meghibásodhatnak.

A csatornakódolás alapvető problémája az, hogy hogyan lehet egy megbízhatatlan eszközön (zajos csatornán) üzenetet nagy megbízhatósággal átküldeni úgy, hogy minél jobban kihasználjuk a csatornát. Megmutatjuk, hogy a kihasználtságnak van egy elvi határa, ez a csatornakapacitás. Mivel a dekódolásnak egy fontos komponense a döntés, ezért a fejezetet az optimális döntéssel kezdjük.

3.1. Bayes-döntés

A gyakorlatban gyakran előforduló feladat, hogy egy A paraméter értékét nem tudjuk közvetlenül megmérni, csupán egy megfigyelt X mennyiség értékéből szeretnénk A-ra következtetni.

A számunkra érdekes esetek matematikai modellje a következőképpen fogalmazható meg:

Legyen X valószínűségi változó, amely értékeit az $\mathcal{X}$ halmazból veszi fel valamilyen eloszlás szerint. ($\mathcal{X}$ lehet véges vagy végtelen halmaz, például $\mathcal{X} = \mathbb{R}^d$ esetén X értéke egy d-dimenziós vektor, vagy $\mathcal{X} = \{0,1\}^d$ esetén d-hosszúságú bináris vektor.) A véletlen A paraméter az $\mathcal{A}$ halmazból veszi fel értékét. Azt, hogy X megfigyeléséből A értékét próbáljuk meghatározni, egy $G : \mathcal{X} \to \mathcal{A}$ függvénnyel írhatjuk le. $G(X)$-et következtetésnek nevezzük. Ezen belül két esetet különböztethetünk meg:

- ha G értékkészlete véges halmaz, akkor $G(X)$-et **döntés**nek,
- ha G értékkészlete végtelen halmaz, akkor $G(X)$-et **becslés**nek

nevezzük.

Természetesen következtetésünket valahogy minősíteni szeretnénk, ezért definiáljuk az ún. **költségfüggvény**t: $C : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$, itt $C(A, G(X))$ értéke adja meg a következtetés jóságát abban az értelemben, hogy minél kisebb a C értéke, annál jobbnak tekintjük a következtetést. Ezért C-t szokás jósági kritériumnak, hasonlósági kritériumnak, vagy megbízhatósági kritériumnak is nevezni. Természetesen $C(A, G(X))$ maga is valószínűségi változó, hiszen értéke függ X és A értékétől.

A költségfüggvény csupán X és A egy-egy konkrét értéke esetén minősíti a következtetést, magát a G következtetésfüggvényt ennek várható értékével jellemezhetjük.

Az $R(G) = \mathbf{E}(C(A, G(X)))$ (csak G-től függő) mennyiséget **globális kockázat**nak nevezzük.

Tekintsük a következő szituációt:

- legyen az X megfigyelés valószínűségi változó, amely értékeit az $\mathcal{X}$ halmazból veszi fel, és nézzük először azt a speciális esetet, amikor $\mathcal{X}$ diszkrét, tehát véges vagy megszámlálhatóan végtelen halmaz;
- az A paraméter szintén valószínűségi változó, értékeit az $\mathcal{A} = \{a_1, \ldots, a_s\}$ véges halmazból veszi fel (ekkor természetesen a $G(\cdot)$ következtetésfüggvény értékkészlete is az $\mathcal{A}$ halmaz), legyen továbbá az A eloszlása a következő: $q_i = \mathbf{P}\{A = a_i\}$.

3.1. definíció. *A $H_i = \{A = a_i\}$ eseményt i-edik hipotézisnek, míg a q_i valószínűségeket **a priori valószínűség**eknek szokás nevezni.*

3.1. példa. Tekintsünk csak egy speciális költségfüggvényt:

$$C(a_i, a_j) = C_{ij} = 1 - \delta_{ij} = \begin{cases} 1, & \text{ha } i \neq j \\ 0 & \text{egyébként} \end{cases} .$$

Vizsgáljuk meg, hogy alakul ekkor a globális kockázat értéke:

$$\begin{aligned} R(G) &= \mathbf{E}(C(A, G(X))) = \\ &= \sum_{i=1}^{s} \sum_{j=1}^{s} \mathbf{P}\{A = a_i, G(X) = a_j\} C(a_i, a_j) = \\ &= \sum_{i=1}^{s} \sum_{j=1}^{s} \mathbf{P}\{A = a_i, G(X) = a_j\} (1 - \delta_{ij}) = \\ &= 1 - \sum_{i=1}^{s} \mathbf{P}\{A = a_i, G(X) = a_i\} = \\ &= 1 - \mathbf{P}\{A = G(X)\} = \\ &= \mathbf{P}\{A \neq G(X)\}. \end{aligned}$$

Láthatjuk tehát, hogy a költségfüggvény fenti választása esetén a globális kockázat éppen megegyezik a **hibavalószínűség**gel. A továbbiakban csak a hibavalószínűséggel foglalkozunk.

3.2. definíció. *A $\mathbf{P}\{A = a_i \mid X = x\}$ feltételes valószínűségeket **a posteriori valószínűség**eknek nevezzük, és $P_i(x)$-szel jelöljük őket.*
*Nevezzük **döntési tartomány**oknak az $\mathcal{X}$ halmaz azon D_j részhalmazait, amelyek bármely elemének megfigyelésekor az a_j-re döntünk:*

$$D_j = \{x : \ G(x) = a_j\}, \qquad j = 1, 2, \ldots, s.$$

Láthatjuk, hogy a G döntésfüggvényt teljesen meghatározzák a D_j döntési tartományok, vagyis a döntési tartományok megadásával magát a döntést is megadtuk.

Válasszuk a j-edik döntési tartományt, D_j^*-ot olyannak, hogy minden $x \in D_j^*$ esetén $P_j(x) \geq P_i(x)$ teljesüljön minden $i \neq j$-re, azaz az x pont akkor eleme a j-edik döntési tartománynak, ha az x megfigyelés esetén az $\{A = a_j\}$ hipotézis feltételes valószínűsége a legnagyobb. Választhatjuk a D_j^* tartományokat páronként diszjunktaknak például úgy, hogy amennyiben valamely x-re a maximum nem egyértelmű, azaz, ha ebben a pontban több P_j függvény is maximális, akkor

x-et az ezek közül legkisebb indexűhöz tartozó D_j^* tartományba soroljuk. Ekkor a döntésfüggvény:

$$G^*(x) = a_i, \qquad \text{ha } x \in D_i^*.$$

Ezt a döntést **Bayes-döntés**nek vagy maximum a posteriori döntésnek nevezzük.

3.1. tétel. *Az így definiált G^* döntésfüggvény optimális, vagyis ennek a döntésnek legkisebb a hibavalószínűsége.*

BIZONYÍTÁS: Legyen $G(x)$ egy tetszőleges döntésfüggvény $\{D_i\}$ döntési tartományokkal. Ekkor

$$\begin{aligned}
R(G) &= \mathbf{P}(A \neq G(X)) = \\
&= 1 - \mathbf{P}(A = G(X)) = \\
&= 1 - \sum_{i=1}^{s} \mathbf{P}(A = a_i, G(X) = a_i) = \\
&= 1 - \sum_{i=1}^{s} \sum_{x} \mathbf{P}(A = a_i, G(X) = a_i \mid X = x)\mathbf{P}(X = x) = \\
&= 1 - \sum_{i=1}^{s} \sum_{x} I_{\{G(x)=a_i\}} \mathbf{P}(A = a_i \mid X = x)\mathbf{P}(X = x) = \\
&= 1 - \sum_{i=1}^{s} \sum_{x} I_{\{x \in D_i\}} P_i(x)\mathbf{P}(X = x) \geq \\
&\geq 1 - \sum_{i=1}^{s} \sum_{x} I_{\{x \in D_i\}} \max_j P_j(x)\mathbf{P}(X = x) = \\
&= 1 - \sum_{x} \left(\max_j P_j(x) \right) \mathbf{P}(X = x) = \\
&= 1 - \sum_{i=1}^{s} \sum_{x} I_{\{x \in D_i^*\}} \left(\max_j P_j(x) \right) \mathbf{P}(X = x) = \\
&= 1 - \sum_{i=1}^{s} \sum_{x} I_{\{x \in D_i^*\}} P_i(x)\mathbf{P}(X = x) = \\
&= \mathbf{P}(A \neq G^*(x)) = \\
&= R(G^*).
\end{aligned}$$

■

Az a posteriori valószínűségek a következő alakba írhatók:

$$\begin{aligned}
P_i(x) &= \mathbf{P}\{A = a_i \mid X = x\} = \\
&= \frac{\mathbf{P}\{A = a_i, X = x\}}{\mathbf{P}\{X = x\}} =
\end{aligned}$$

$$= \frac{\mathbf{P}\{A=a_i\}\mathbf{P}\{X=x \mid A=a_i\}}{\mathbf{P}\{X=x\}} =$$
$$= \frac{q_i p_i(x)}{\mathbf{P}\{X=x\}},$$

ahol $p_i(x)$-szel az X feltételes eloszlását jelöljük az $\{A=a_i\}$ feltétel mellett. Látszik, hogy a $P_i(x)$ a posteriori valószínűség ugyanarra az i indexre veszi fel a maximumát, amelyre $q_i p_i(x)$. Amennyiben az a priori valószínűségeloszlás egyenletes, azaz $q_i = \frac{1}{s}$ minden i-re, akkor a Bayes-döntés a következőképpen alakul:

$$x \in D_j^*, \text{ ha } p_j(x) = \max_i p_i(x).$$

Az ilyen típusú döntést **maximum-likelihood döntés**nek nevezzük.

MEGJEGYZÉS: Maximum-likelihood döntést általában olyan esetekben hozunk, amikor az a priori valószínűségek nem ismertek. Ilyenkor az $\mathcal{A}$ halmaz azon elemére döntünk, amellyel mint feltétellel a mért értéknek legnagyobb a feltételes valószínűsége.

3.2. példa (Dekódolás bináris szimmetrikus csatorna kimenetén). Ebben az esetben az $\mathbf{A} = (A_1, \ldots, A_n)$ valószínűségi változó értékei bináris, n hosszúságú kódszavak. Jelöljünk egy kódszót $\mathbf{c}_i$-vel, ahol $\mathbf{c}_i = (c_{i_1}, c_{i_2}, \ldots, c_{i_n})$, vagyis az i-edik kódszó j-edik bitje c_{i_j}. A kódszavakat egy zajos, bináris szimmetrikus csatornán továbbítjuk, melynek kimenetén az $\mathbf{X} = X_1, \ldots, X_n$ valószínűségi változó (szintén n hosszúságú bináris sorozat) jelenik meg. Az $\mathbf{X}$ eloszlását abban az esetben, amikor a továbbított kódszó $\mathbf{c}_i$ volt, a csatorna p átmenetvalószínűsége határozza meg a következőképpen:

$$\begin{aligned}
&\mathbf{P}\{\mathbf{X} = x_1, x_2, \ldots, x_n \mid \mathbf{A} = \mathbf{c}_i\} = \\
&\quad = \mathbf{P}\{X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n \mid A_1 = c_{i_1}, \ldots, A_n = c_{i_n}\} = \\
&\quad = \mathbf{P}\{X_1 = x_1 \mid A_1 = c_{i_1}\}\mathbf{P}\{X_2 = x_2 \mid A_2 = c_{i_2}\} \cdots \\
&\qquad \mathbf{P}\{X_n = x_n \mid A_n = c_{i_n}\}.
\end{aligned}$$

Ez az egyenlőség definiálja az emlékezetnélküli csatornát, vagyis a j-edik kimeneti szimbólum eloszlása csak a j-edik bemeneti bit értékétől függ. Az, hogy a csatorna átmenetvalószínűsége p $(0 < p < 1/2)$, éppen azt jelenti, hogy

$$\begin{aligned}
\mathbf{P}\{X_k = x_k \mid A_k = c_{i_k}\} &= \begin{cases} p, & \text{ha } x_k \neq c_{i_k} \\ 1-p, & \text{ha } x_k = c_{i_k} \end{cases} = \\
&= p^{I_{\{x_k \neq c_{i_k}\}}} (1-p)^{1-I_{\{x_k \neq c_{i_k}\}}} = \\
&= \left(\frac{p}{1-p}\right)^{I_{\{x_k \neq c_{i_k}\}}} (1-p).
\end{aligned}$$

Ezért tehát

$$\begin{aligned}\mathbf{P}\{\mathbf{X}=\mathbf{x}\mid\mathbf{A}=\mathbf{c}_i\} &= \prod_{k=1}^{n}\left[\left(\frac{p}{1-p}\right)^{I_{\{x_k\neq c_{i_k}\}}}(1-p)\right]= \\ &= \left(\frac{p}{1-p}\right)^{\sum_{k=1}^{n} I_{\{x_k\neq c_{i_k}\}}}(1-p)^n.\end{aligned}$$

Amennyiben a csatorna $\mathbf{X}$ kimenete ismeretében a csatorna bemenetére adott kódszót maximum-likelihood döntés segítségével akarjuk meghatározni, azt a $\mathbf{c}_i$-t választjuk, amelyre $p_i(\mathbf{x})=\mathbf{P}\{\mathbf{X}=\mathbf{x}\mid\mathbf{A}=\mathbf{c}_i\}$ maximális, azaz amelyre $\sum_{k=1}^{n} I_{\{x_k\neq c_{i_k}\}}$ minimális (hiszen $\frac{p}{1-p}<1$).

Vegyük észre, hogy $\sum_{k=1}^{n} I_{\{x_k\neq c_{i_k}\}}$ éppen az $\mathbf{x}$ és a $\mathbf{c}_i$ Hamming-távolsága, ami azt jelenti, hogy bináris szimmetrikus csatorna esetén a maximum-likelihood dekódolás éppen a csatorna kimenetén vett sorozattól minimális Hamming-távolságra levő kódszó választását jelenti. (Két egyenlő hosszú bináris sorozat Hamming-távolságán azon bitek számát értjük, amelyekben a két sorozat különbözik.)

Amennyiben az X megfigyelés értéke valós, és $f(x)$ az eloszlást meghatározó sűrűségfüggvény, továbbá $f_i(x)=f(x\mid A=a_i)$ az A paraméter a_i értékéhez tartozó feltételes sűrűségfüggvény, akkor bebizonyítható, hogy az a posteriori valószínűségfüggvények a következő alakba írhatók:

$$P_i(x)=\frac{q_i f_i(x)}{f(x)}.$$

Látszik, hogy ez az eset formailag teljesen analóg a diszkrét esettel, ezt a következő példával szeretnénk érzékeltetni:

Tegyük fel, hogy minden egyes $x\in\mathbb{R}$ számra csak azt tudjuk megfigyelni, hogy az X valószínűségi változó $\Delta/2$ sugarú környezetébe esik-e, azaz, hogy bekövetkezett-e a $B_\Delta=X\in(x-\Delta/2,x+\Delta/2)$ esemény. Tegyük fel továbbá, hogy $f,f_1,\ldots,f_s$ folytonos függvények. Ekkor a B_Δ-hoz tartozó a posteriori valószínűségek:

$$\begin{aligned}P_i^\Delta(x) &= \mathbf{P}\{A=a_i\mid B_\Delta\}= \\ &= \frac{\mathbf{P}\{A=a_i,B_\Delta\}}{\mathbf{P}\{B_\Delta\}}= \\ &= \frac{\mathbf{P}\{A=a_i\}\mathbf{P}\{B_\Delta\mid A=a_i\}}{\mathbf{P}\{B_\Delta\}}=\end{aligned}$$

$$= \frac{q_i \int_{x-\Delta/2}^{x+\Delta/2} f_i(z)\,\mathrm{d}z}{\int_{x-\Delta/2}^{x+\Delta/2} f(z)\,\mathrm{d}z} = \frac{q_i \frac{1}{\Delta} \int_{x-\Delta/2}^{x+\Delta/2} f_i(z)\,\mathrm{d}z}{\frac{1}{\Delta} \int_{x-\Delta/2}^{x+\Delta/2} f(z)\,\mathrm{d}z}.$$

Az f_i-k folytonossága miatt alkalmazhatjuk az integrálszámítás középértéktételét, amiből

$$\lim_{\Delta\to 0} P_i^{\Delta}(x) = \frac{q_i f_i(x)}{f(x)}.$$

Látszik tehát, hogy egyre kisebb Δ esetén aszimptotikusan a maximum a posteriori döntés a következő alakú:

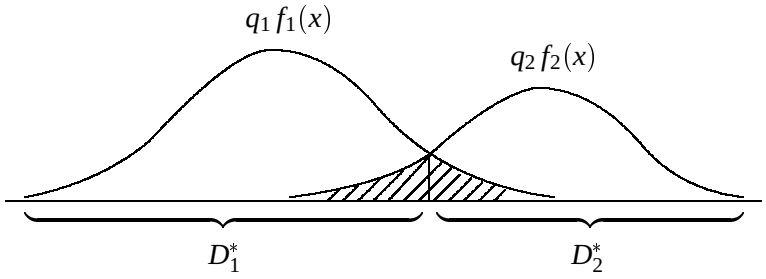
$$G^*(x) = \max_i q_i f_i(x),$$

ekkor

$$D_i^* = \{x:\ G^*(x) = q_i f_i(x)\}.$$

Megmutatjuk, hogy ez tényleg optimális döntés.

$$\begin{aligned}
R(G) &= 1 - \sum_{i=1}^{s} \mathbf{P}\{A = a_i, G(\mathbf{X}) = a_i\} = \\
&= 1 - \sum_{i=1}^{s} \mathbf{P}\{\mathbf{X} \in D_i \mid A = a_i\}\mathbf{P}\{A = a_i\} = \\
&= 1 - \sum_{i=1}^{s} \int_{D_i} f_i(\mathbf{x})\,\mathrm{d}\mathbf{x}\, q_i = \\
&= 1 - \sum_{i=1}^{s} \int_{D_i} q_i f_i(\mathbf{x})\,\mathrm{d}\mathbf{x} \geq \\
&\geq 1 - \sum_{i=1}^{s} \int_{D_i} \max_j \{q_j f_j(\mathbf{x})\}\,\mathrm{d}\mathbf{x} = \\
&= 1 - \int_{\mathbb{R}^d} \max_j \{q_j f_j(\mathbf{x})\}\,\mathrm{d}\mathbf{x} =
\end{aligned}$$



3.1. ábra. A Bayes-döntés hibavalószínűsége.

$$
\begin{aligned}
&= 1 - \sum_{i=1}^{s} \int_{D_i^*} \max_j \{q_j f_j(\mathbf{x})\}\, \mathrm{d}\mathbf{x} = \\
&= 1 - \sum_{i=1}^{s} \int_{D_i^*} q_i f_i(\mathbf{x})\, \mathrm{d}\mathbf{x} = R(G^*).
\end{aligned}
$$

A fenti levezetés alapján kiszámíthatjuk a Bayes-döntés hibavalószínűségét:

$$R(G^*) = 1 - \int_{\mathbb{R}^d} \max_j \{q_j f_j(\mathbf{x})\}\, \mathrm{d}\mathbf{x}.$$

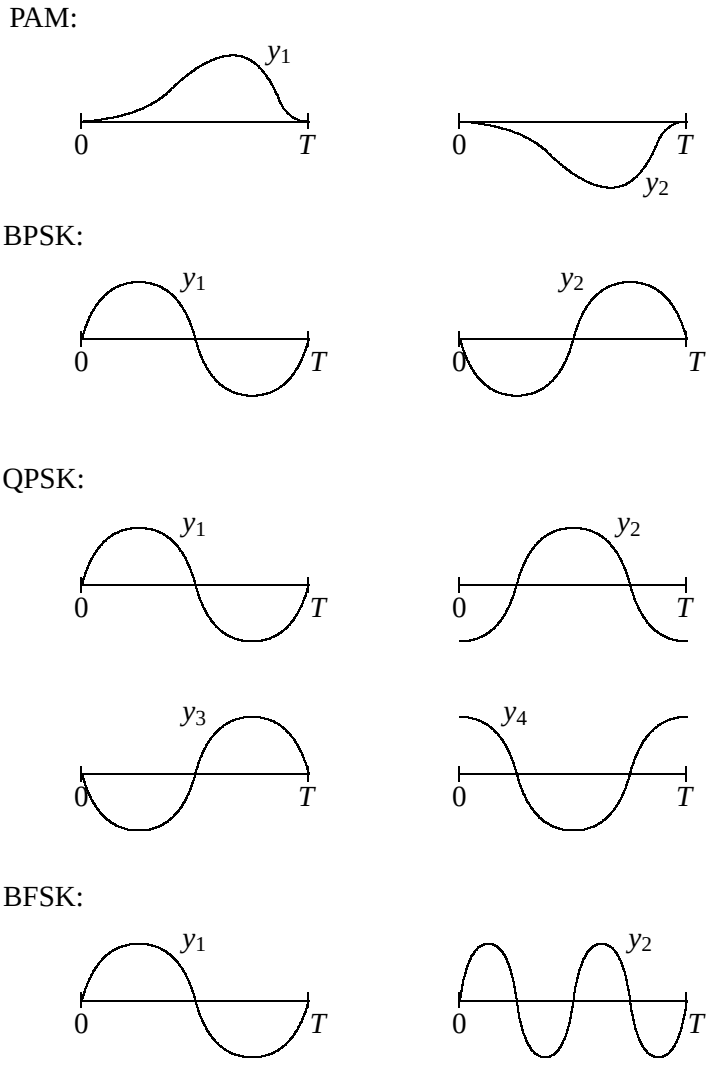
$s = 2$ esetén

$$R(G^*) = \int_{\mathbb{R}^d} \min\{q_1 f_1(\mathbf{x}), q_2 f_2(\mathbf{x})\}\, \mathrm{d}\mathbf{x},$$

amelyet a 3.1. ábra szemléltet egy dimenzióban.

3.2. Optimális detektálás analóg csatorna kimenetén

A következő szituációt vizsgáljuk: legyen a $Z_1, Z_2, \ldots$ stacionárius forrás olyan, hogy Z_i értékeit az $\mathcal{A} = \{a_1, a_2, \ldots, a_s\}$ halmazból veszi, és ezeket az értékeket (üzeneteket) egy analóg csatornán kell továbbítanunk. Ezt az analóg (fizikai) csatornát úgy modellezzük, hogy az idő τ hosszúságú szeletekre van osztva, és az i-edik szeletben a Z_i betű átvitelére kerül sor úgy, hogy a lehetséges $a_1, \ldots, a_s$ betűknek megfeleltetünk $[0, \tau]$-n értelmezett $y_1(t), \ldots, y_s(t)$ függvényeket (jelalakokat), és ha $Z_i = a$, akkor az $[(i-1)\tau, i\tau]$ intervallumon az $y_a(t - (i-1)\tau)$ jelet küldjük el. Ezt a technikát **digitális moduláció**nak nevezzük (3.2. ábra).



3.2. ábra. Digitális moduláció.

3.3. példa. PAM (impulzus amplitúdómoduláció, pulse amplitude modulation):

$$y_i(t) = a_i \cdot y(t), \quad i = 1,2.$$

3.4. példa. BPSK (bináris fázismoduláció, binary phase shift keying):

$$y_i(t) = \sin(t + (i-1)\pi), \quad i = 1,2.$$

3.5. példa. QPSK (kvadratikus fázismoduláció, quadratic phase shift keying):

$$y_i(t) = \sin\left(t + (i-1)\frac{\pi}{2}\right), \quad i = 1,2,3,4.$$

3.6. példa. BFSK (bináris frekvenciamoduláció, binary frequency shift keying):

$$y_i(t) = \sin\left((\omega_0 + (i-1)\Delta)t\right), \quad i = 1,2.$$

A csatorna bemenetére adott jel tehát a következő:

$$U(t) = \sum_{i=-\infty}^{\infty} y_{Z_i}(t-(i-1)\tau).$$

Feltételezzük, hogy az analóg csatorna additív zajos, ami azt jelenti, hogy a kimenete

$$V(t) = U(t) + N(t),$$

ahol az $N(t)$ egy 0 várható értékű stacionárius sztochasztikus folyamat. Feltesszük, hogy a csatorna kimenetén ismerjük a τ hosszú időszeletek határait, azaz a szinkronizálást már elvégeztük. Az egyszerűség kedvéért tekintsük a döntést Z_1-re, azaz $V(t)$-t megfigyeljük a $[0,\tau]$ intervallumon, abból $\frac{\tau}{d}$ időközönként mintát veszünk, és a minták alapján döntünk. Ekkor

$$\begin{aligned}
\mathbf{X} &= \left(V\left(\frac{\tau}{d}\right), V\left(\frac{2\tau}{d}\right), \ldots, V(\tau)\right) = \\
&= \left(y_{Z_1}\left(\frac{\tau}{d}\right), y_{Z_1}\left(\frac{2\tau}{d}\right), \ldots, y_{Z_1}(\tau)\right) + \\
&\quad \left(N\left(\frac{\tau}{d}\right), N\left(\frac{2\tau}{d}\right), \ldots, N(\tau)\right) = \\
&= \mathbf{y}_{Z_1} + \mathbf{N},
\end{aligned}$$

ahol $\mathbf{y}_i$ jelöli az $y_i(t)$ mintáinak és $\mathbf{N}$ az $N(t)$ mintáinak a vektorát. Ha $f(\mathbf{x})$ jelöli az $\mathbf{N}$ sűrűségfüggvényét, akkor az $\mathbf{y}_i + \mathbf{N}$ sűrűségfüggvénye

$$f_i(\mathbf{x}) = f(\mathbf{x} - \mathbf{y}_i),$$

tehát a Bayes-döntéshez a

$$q_i f_i(\mathbf{x}) = q_i f(\mathbf{x} - \mathbf{y}_i)$$

értékeket kell i-ben maximalizálni. Nézzük a továbbiakban azt a speciális esetet, amikor $s=2,\ q_1=q_2=\frac{1}{2}$, és az $N(t)$ egy 0 várható értékű stacionárius Gauss-folyamat. Ekkor az $N(t)$ folyamat $\mathbf{N}$ mintáinak vektora normális eloszlású $\mathbf{0}$ várható értékkel és $\underline{\underline{K}}$ kovarianciamátrixszal, azaz

$$f(\mathbf{x}) = \frac{1}{\left(\sqrt{2\pi}\right)^d \sqrt{\det \underline{\underline{K}}}}\, \mathrm{e}^{-\frac{1}{2}\left(\mathbf{x},\underline{\underline{K}}^{-1}\mathbf{x}\right)},$$

tehát

$$f_i(\mathbf{x}) = \frac{1}{\left(\sqrt{2\pi}\right)^d \sqrt{\det \underline{\underline{K}}}}\, \mathrm{e}^{-\frac{1}{2}\left(\mathbf{x}-\mathbf{y}_i,\underline{\underline{K}}^{-1}(\mathbf{x}-\mathbf{y}_i)\right)}.$$

(a,b) alakban a skalárszorzatot jelöltük.

A hibavalószínűséget a Bayes-döntés minimalizálja, azaz

$$\begin{aligned}
D_1^* &= \{\mathbf{x}:\ q_1 f_1(\mathbf{x}) \geq q_2 f_2(\mathbf{x})\} = \\
&= \{\mathbf{x}:\ f(\mathbf{x}-\mathbf{y}_1) \geq f(\mathbf{x}-\mathbf{y}_2)\} = \\
&= \{\mathbf{x}:\ \ln f(\mathbf{x}-\mathbf{y}_1) \geq \ln f(\mathbf{x}-\mathbf{y}_2)\} = \\
&= \{\mathbf{x}:\ \left(\mathbf{x}-\mathbf{y}_1,\underline{\underline{K}}^{-1}(\mathbf{x}-\mathbf{y}_1)\right) \leq \left(\mathbf{x}-\mathbf{y}_2,\underline{\underline{K}}^{-1}(\mathbf{x}-\mathbf{y}_2)\right)\} = \\
&= \{\mathbf{x}:\ \left(\mathbf{x},\underline{\underline{K}}^{-1}\mathbf{x}\right) - \left(\mathbf{y}_1,\underline{\underline{K}}^{-1}\mathbf{x}\right) - \left(\mathbf{x},\underline{\underline{K}}^{-1}\mathbf{y}_1\right) + \left(\mathbf{y}_1,\underline{\underline{K}}^{-1}\mathbf{y}_1\right) \leq \\
&\qquad \left(\mathbf{x},\underline{\underline{K}}^{-1}\mathbf{x}\right) - \left(\mathbf{y}_2,\underline{\underline{K}}^{-1}\mathbf{x}\right) - \left(\mathbf{x},\underline{\underline{K}}^{-1}\mathbf{y}_2\right) + \left(\mathbf{y}_2,\underline{\underline{K}}^{-1}\mathbf{y}_2\right)\} =
\end{aligned}$$

$\underline{\underline{K}}^{-1}$ szimmetriájának felhasználásával:

$$\begin{aligned}
&= \{\mathbf{x}:\ -2\left(\mathbf{x},\underline{\underline{K}}^{-1}\mathbf{y}_1\right) + \left(\mathbf{y}_1,\underline{\underline{K}}^{-1}\mathbf{y}_1\right) \leq \\
&\qquad -2\left(\mathbf{x},\underline{\underline{K}}^{-1}\mathbf{y}_2\right) + \left(\mathbf{y}_2,\underline{\underline{K}}^{-1}\mathbf{y}_2\right)\} = \\
&= \{\mathbf{x}:\ (\mathbf{x},\mathbf{v}) \leq C_1\},
\end{aligned}$$

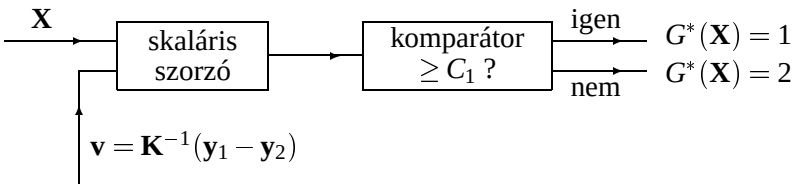
ahol

$$\mathbf{v} = \underline{\underline{K}}^{-1}(\mathbf{y}_2-\mathbf{y}_1)$$

és

$$C_1 = \frac{1}{2}\left(\left(\mathbf{y}_2,\underline{\underline{K}}^{-1}\mathbf{y}_2\right) - \left(\mathbf{y}_1,\underline{\underline{K}}^{-1}\mathbf{y}_1\right)\right).$$

Látszik tehát, hogy ebben az esetben a Bayes-döntés rendkívül egyszerűen megvalósítható, amennyiben ismerjük az additív Gauss-zaj kovarianciafüggvényét. Az is látszik, hogy a döntési tartományok határa egy affin hipersík (eltolt altér). A 3.3. ábrán látható optimális detektálást végrehajtó eszközt a híradástechnikában **diszkrét korrelációs detektor**nak nevezik.



3.3. ábra. Diszkrét korrelációs detektor.

3.3. Csatorna, csatornakapacitás

A diszkrét csatorna leírható a bemeneti ábécével, a kimeneti ábécével, illetve az átmenetvalószínűségekkel. Az $\mathcal{U}$ bemeneti, és a $\mathcal{V}$ kimeneti ábécék legalább kételemű véges halmazok. A csatorna egyenletes időközönként a bemenetére adott u szimbólumra (amely a bemeneti ábécé eleme) a kimenetén a kimeneti ábécé egy v elemét adja válaszul. A csatorna viselkedését a bemenőjelek $U_1, U_2, \ldots, U_n, \ldots$, illetve a kimenőjelek $V_1, V_2, \ldots, V_n$ sorozata írja le; ezen valószínűségi változók együttes eloszlása:

$$\begin{aligned}\mathbf{P}\{V_1 = v_1, \ldots, V_n = v_n, U_1 = u_1, \ldots, U_n = u_n\} = \\ = \mathbf{P}\{U_1 = u_1, \ldots, U_n = u_n\} \cdot \\ \mathbf{P}\{V_1 = v_1, \ldots, V_n = v_n \mid U_1 = u_1, \ldots, U_n = u_n\},\end{aligned}$$

ahol $\mathbf{P}\{U_1 = u_1, \ldots, U_n = u_n\}$-et a forrás eloszlása és a kódoló együttesen határozzák meg, míg a $\mathbf{P}\{V_1 = v_1, \ldots, V_n = v_n \mid U_1 = u_1, \ldots, U_n = u_n\}$ feltételes valószínűségek ($n = 1, 2, \ldots$) a csatornát leíró ún. átmenetvalószínűségek.

MEGJEGYZÉS: Azt mindig feltesszük, hogy a csatorna nem szinkronizációhibás, vagyis, hogy a kimenetén pontosan annyi szimbólum jelenik meg, amennyi a bemenetére került, tehát szimbólum nem vész el, és nem is keletkezik fölöslegesen.

Az egyszerűség kedvéért vezessük be a következő jelöléseket:

$$\begin{aligned} p(\mathbf{v}, \mathbf{u}) &= \mathbf{P}\{\mathbf{V} = \mathbf{v}, \mathbf{U} = \mathbf{u}\} = \\ &= \mathbf{P}\{V_1 = v_1, \ldots, V_n = v_n, U_1 = u_1, \ldots, U_n = u_n\} \\ p(\mathbf{u}) &= \mathbf{P}\{\mathbf{U} = \mathbf{u}\} = \\ &= \mathbf{P}\{U_1 = u_1, \ldots, U_n = u_n\} \\ p(\mathbf{v}) &= \mathbf{P}\{\mathbf{V} = \mathbf{v}\} = \\ &= \mathbf{P}\{V_1 = v_1, \ldots, V_n = v_n\} \\ p(\mathbf{v} \mid \mathbf{u}) &= \mathbf{P}\{\mathbf{V} = \mathbf{v} \mid \mathbf{U} = \mathbf{u}\} = \\ &= \mathbf{P}\{V_1 = v_1, \ldots, V_n = v_n \mid U_1 = u_1, \ldots, U_n = u_n\}, \end{aligned}$$

illetve

$$\begin{aligned} p(v_i,u_i) &= \mathbf{P}\{V_i=v_i, U_i=u_i\} \\ p(u_i) &= \mathbf{P}\{U_i=u_i\} \\ p(v_i) &= \mathbf{P}\{V_i=v_i\} \\ p(v_i \mid u_i) &= \mathbf{P}\{V_i=v_i \mid U_i=u_i\} \end{aligned}$$

Most definiáljuk a diszkrét memóriamentes csatorna fogalmát:

3.3. definíció. *Egy csatornát akkor nevezünk* ***diszkrét memóriamentes csatornának****, ha a* $p(\mathbf{v} \mid \mathbf{u})$ *átmenetvalószínűségek minden n-re,* $\mathbf{u} \in \mathcal{U}^n$*-re és* $\mathbf{v} \in \mathcal{V}^n$*-re kielégítik a következő egyenlőséget (*$\mathcal{U}^n$ *és* $\mathcal{V}^n$ *az* $\mathcal{U}$ *ill.* $\mathcal{V}$ *elemeiből alkotható n-hosszúságú sorozatok halmazát jelöli):*

$$p(\mathbf{v} \mid \mathbf{u}) = \prod_{i=1}^{n} p(v_i \mid u_i),$$

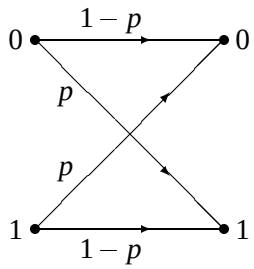
azaz a csatorna kimeneti szimbólumának eloszlása mindig csak az aktuális bemenettől függ.

Nézzünk meg példaként néhány fontos diszkrét memóriamentes csatornát.

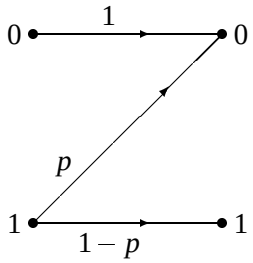
3.7. példa (Bináris szimmetrikus csatorna (BSC)). Ezzel a csatornatípussal már foglalkoztunk a 3.2. példa kapcsán. Itt mind az $\mathcal{U}$ bemeneti, mind a $\mathcal{V}$ kimeneti ábécé kételemű ($\mathcal{U}=\mathcal{V}=\{0,1\}$), az átmenetvalószínűségek pedig:

$$\begin{aligned} \mathbf{P}\{V=1 \mid U=0\} &= \mathbf{P}\{V=0 \mid U=1\} = p \\ \mathbf{P}\{V=1 \mid U=1\} &= \mathbf{P}\{V=0 \mid U=0\} = 1-p. \end{aligned}$$

Az átmenetvalószínűségek a 3.4. ábrán látható módon ábrázolhatóak.



3.4. ábra. Bináris szimmetrikus csatorna.



3.5. ábra. Bináris z-csatorna.

3.8. példa (Bináris z-csatorna). Ennek a csatornatípusnak a bemeneti és kimeneti ábécéje szintén kételemű (bináris), azonban ez a csatorna nem szimmetrikus, a bemenetére adott „0” szimbólumot egy valószínűséggel hiba nélkül továbbítja, csupán az „1” szimbólum továbbításánál hibázhat p valószínűséggel, azaz

$$\begin{aligned}\mathbf{P}\{V=0 \mid U=0\} &= 1,\\ \mathbf{P}\{V=0 \mid U=1\} &= p,\\ \mathbf{P}\{V=1 \mid U=1\} &= 1-p\end{aligned}$$

(lásd 3.5. ábra).

3.9. példa (Bináris törléses csatorna). A bináris törléses csatorna bemeneti ábécéje $\mathcal{U}=\{0,1\}$, míg a kimeneti ábécé tartalmaz egy további speciális szimbólumot is: $\mathcal{V}=\{0,1,*\}$. Hibázáskor mindig a „$*$” szimbólum jelenik meg a csatorna kimenetén:

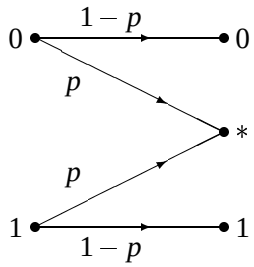
$$\begin{aligned}\mathbf{P}\{V=* \mid U=0\} &= \mathbf{P}\{V=* \mid U=1\} = p\\ \mathbf{P}\{V=1 \mid U=1\} &= \mathbf{P}\{V=0 \mid U=0\} = 1-p\end{aligned}$$

(lásd 3.6. ábra).

A diszkrét memóriamentes csatornák jellemzésére vezessük be a csatornakapacitás fogalmát:

3.4. definíció. *A $p(v_i \mid u_i)$ $(v_i \in \mathcal{V}, u_i \in \mathcal{U})$ átmenetvalószínűségekkel leírt diszkrét memóriamentes csatorna C* ***csatornakapacitása*** *a következő:*

$$C = \max\{I(U;V)\},$$



3.6. ábra. Bináris törléses csatorna.

ahol a maximumot az összes olyan (U,V) valószínűségi változópár halmazán kell képezni, ahol U a csatorna $\mathcal{U}$ bemeneti ábécéjén, V pedig a $\mathcal{V}$ kimeneti ábécén veszi fel értékeit, továbbá együttes eloszlásuk kielégíti a $\mathbf{P}\{V = v_i \mid U = u_i\} = p(v_i \mid u_i)$ egyenlőséget, azaz V tekinthető a csatorna által U-ra adott válasznak. Ekkor

$$\begin{aligned} C &= \max\left\{\sum_{u\in\mathcal{U},v\in\mathcal{V}} p(u,v)\log\frac{p(u,v)}{p(u)p(v)}\right\} = \\ &= \max\left\{\sum_{u\in\mathcal{U},v\in\mathcal{V}} p(u)p(v\mid u)\log\frac{p(v\mid u)}{\sum\limits_{u'\in\mathcal{U}} p(u')p(v\mid u')}\right\}. \end{aligned}$$

Láthatjuk tehát, hogy a maximumot elég a csatorna bemeneti ábécéjén definiálható $p(u)$ $(u\in\mathcal{U})$ eloszlásokon képezni.

MEGJEGYZÉS: A csatornakapacitás definíciójában azért írhattunk maximumot szuprémum helyett, mert az $I(U;V)$ egy folytonos függvény a $p(u)$ $(u\in\mathcal{U})$ eloszlások zárt, korlátos halmazán, ezért biztosan van olyan bemeneti eloszlás, hogy a bemenet és a kimenet közötti kölcsönös információ eléri a csatornakapacitást.

A csatornakapacitás intuitív jelentése tehát a „csatornán maximálisan átvihető információ mennyisége”. Tulajdonképpen ennek az állításnak a pontos megfogalmazását adja a későbbiekben tárgyalandó csatornakódolási tétel.

Példaként számoljuk ki a 3.7. példában szereplő bináris szimmetrikus csatorna kapacitását:

3.10. példa. Esetünkben a bemenet eloszlása egyetlen α paraméterrel jellemezhető:

$$\mathbf{P}\{U=0\}=\alpha; \qquad \mathbf{P}\{U=1\}=1-\alpha.$$

Ekkor a csatornakapacitás:

$$C = \max_{\alpha} I(U;V) = \max_{\alpha} (H(V) - H(V \mid U)).$$

Könnyen látható, hogy

$$H(V \mid U) = h(p) = -p\log p - (1-p)\log(1-p)$$

független U eloszlásától, ezért elég

$$H(V) = h(\alpha(1-p) + (1-\alpha)p)$$

maximumát keresni. Tudjuk, hogy a $h(x)$ függvény az $x = \frac{1}{2}$ helyen veszi fel a maximumát, ezért $H(V)$ akkor maximális, ha $\alpha(1-p) + (1-\alpha)p = \frac{1}{2}$, azaz ha $\alpha = \frac{1}{2}$, tehát egyenletes a bemenet eloszlása. Ekkor tehát

$$C = \log 2 - h(p) = 1 - h(p).$$

Látszik, hogy a csatornakapacitás $p = 0$ és $p = 1$ esetén $C = 1$, $p = \frac{1}{2}$ esetén $C = 0$, továbbá, hogy a csatornakapacitás a p átmenetvalószínűség alulról konvex függvénye.

3.4. Csatornakódolási tétel

Definiáljuk most az 1.1. ábra hírközlési modelljének többi blokkját is.

3.5. definíció. ***Csatornakód****nak nevezzük az $\mathcal{U}^n$ egy M elemű $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_M\}$ részhalmazát, amelynek elemei a $\mathbf{c}_i = c_{i_1}, \ldots, c_{i_n}$ kódszavak ($c_{i_j} \in \mathcal{U},\ i = 1, \ldots, M$, $j = 1, \ldots, n$).*

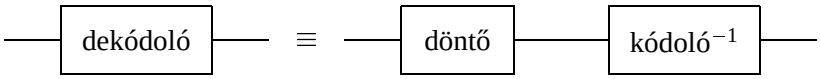
3.6. definíció. *A **kódoló** egy invertálható $f : \mathcal{Y}^k \to \{\mathbf{c}_1, \ldots, \mathbf{c}_M\}$ függvény, amely a forrás k hosszúságú blokkjaihoz kódszavakat, vagyis a csatorna bemeneti ábécéjének elemeiből alkotott n hosszúságú sorozatokat rendel:*

$$f(Y_1, \ldots, Y_k) = (U_1, \ldots, U_n).$$

3.7. definíció. *A **dekódoló** két független részre bontható, **döntő**re és a **kódoló inverzé**re (lásd 3.7. ábra).*

A döntőt a $g : \mathcal{V}^n \to \{\mathbf{c}_1, \ldots, \mathbf{c}_M\}$ leképezés adja meg, azaz a csatorna kimenetén megjelenő n hosszúságú blokkokhoz rendel hozzá kódszavakat. A döntést definiálhatjuk a $D_1, \ldots, D_M \subset \mathcal{V}^n$ döntési tartományokkal is a következőképpen:

$$D_m = \{\mathbf{v} \in \mathcal{V}^n :\ g(\mathbf{v}) = \mathbf{c}_m\}, \qquad m = 1, 2, \ldots, M.$$



3.7. ábra. A dekódoló felépítése.

A kódoló inverze egyszerűen az f kódolófüggvény f^{-1} inverzét jelenti, ez a döntő kimenetén adódó kódszóhoz hozzárendeli azt az üzenetet (azaz a forrás k-hosszúságú blokkját), amelynek kódja éppen ez a kódszó.

Most tekintsük át a csatorna megbízhatóságának mérőszámait.

3.8. definíció. *Tegyük fel, hogy M számú üzenet valamelyikét akarjuk továbbítani (például bináris forrás k hosszúságú blokkjai esetén $M = 2^k$). Jelöljük ezeket $\mathbf{y}_1, \ldots, \mathbf{y}_M$-mel. Adott kódoló és dekódoló esetén annak a valószínűsége, hogy a dekódolás hibás, feltéve, hogy az $\mathbf{y}_m$ m-edik üzenetet továbbítottuk:*

$$P_{e,m} = \sum_{f^{-1}(\mathbf{y}') \neq \mathbf{y}_m} p(\mathbf{y}' \mid \mathbf{y}_m) = \sum_{\mathbf{y}' \notin D_m} p(\mathbf{y}' \mid \mathbf{y}_m).$$

Így, ha $p(\mathbf{y}_m)$ az m-edik közlemény előfordulásának valószínűsége, akkor a hibás dekódolás valószínűsége:

$$P_e = \sum_{m=1}^{M} p(\mathbf{y}_m) P_{e,m} = \sum_{m=1}^{M} \sum_{\mathbf{y}' \notin D_m} p(\mathbf{y}', \mathbf{y}_m).$$

A P_e hibavalószínűség tehát függ a közlemények eloszlásától. Mi olyan kódolást szeretnénk azonban, amely ettől függetlenül jó, ezért vezessük be a

$$\bar{P_e} = \frac{1}{M} \sum_{m=1}^{M} P_{e,m}$$

átlagos hiba kifejezését. $\bar{P_e}$ értéke független az üzenetek eloszlásától.

MEGJEGYZÉS: Ha P_e minimalizálása a célunk, akkor mint a 3.1. szakaszban láttuk, azt a Bayes-döntés megteszi. Ez azt jelenti, hogy a dekódoló azt az $\mathbf{y}_m$ közleményt választja ki, amelyre az $\mathbf{y}_m$ feltételes valószínűsége maximális a csatorna kimenetén megjelenő $\mathbf{y}'$ sorozattal mint feltétellel. $\bar{P_e}$ minimalizálása megfelel a hibavalószínűség minimalizálásának egyenletes a priori eloszlás esetén. Ekkor az optimális döntést maximum-likelihood döntésnek nevezzük. Ebben az esetben a kódoló–dekódoló párt a kódszavak $\{\mathbf{c}_1, \ldots, \mathbf{c}_M\}$ halmazával teljesen megadtuk.

Tekintsünk egy rendkívül egyszerű példát annak illusztrálására, hogy csatornakódolással hogyan növelhetjük meg az átvitel biztonságát.

3.11. példa. Tekintsük a 3.7. példában ismertetett p hibavalószínűségű bináris szimmetrikus csatornát. Kódolás nélkül a hibázás valószínűsége nyilván:

$$P_e = P_{e,1} = P_{e,2} = \bar{P_e} = p.$$

Vegyük most a következő kódot: $C = \{000, 111\}$, vagyis $M = 2$, $n = 3$, $k = 1$, illetve a kódolási szabály: $f(0) = 000$; $f(1) = 111$. Ha a dekódolási szabály a többségi döntés, azaz akkor és csak akkor dekódolunk „1”-est, ha a vett három bit között legalább két „1”-es van, akkor könnyű látni, hogy a hibavalószínűségek:

$$P_e{}' = P_{e,1}{}' = P_{e,2}{}' = \bar{P_e'} = 3p^2 - 2p^3,$$

amely $p \ll \frac{1}{2}$ esetén sokkal kisebb, mint a kódolás nélkül elérhető hibavalószínűség.

Az előző példában láthattuk, hogy a hibavalószínűség csökkenéséért árat kellett fizetnünk: az üzenetek leadásához háromszor olyan hosszú időre volt szükségünk. Definiáljuk tehát a jelsebesség fogalmát.

3.9. definíció. ***Jelsebesség****nek vagy* ***kódolási sebesség****nek nevezzük az* $R = \frac{\log M}{n}$ *értéket.*

A jelsebesség mértékegysége bit/csatornahasználat, ami a csatorna kihasználtságát méri. Azt jelenti, hogy egy csatornahasználattal hány bit üzenetet viszünk át.

3.12. példa. Bináris forrás k hosszúságú blokkjainak kódolása esetén a jelsebesség $R = \frac{k}{n}$.

Olyan kódot szeretnénk készíteni, hogy egyidejűleg $\bar{P_e}$ kicsi, az R jelsebesség pedig minél nagyobb legyen. Az a kérdés, hogy megfelelő kódolással és dekódolással milyen maximális jelsebesség érhető el, ha az átvitel hűségével szemben támasztott követelmény: $\bar{P_e} < \varepsilon$, ahol $\varepsilon > 0$ adott hibakorlát. A következőkben tárgyalandó csatornakódolási tétel, illetve annak megfordítása azt a meglepő tényt mondja ki, hogy tetszőleges ε-ra ez a maximálisan elérhető jelsebesség független ε-tól, és megegyezik a csatornakapacitással.

Először azt a kérdést vizsgáljuk, hogy adott $R = \frac{\log M}{n}$ jelsebesség mellett milyen kicsi lehet a dekódolás hibavalószínűsége. Jelölje az $\mathbf{Y}$ valószínűségi változó a továbbítandó üzenetet, mely M különböző értéket vehet fel. A kódoló után az $f(\mathbf{Y}) = \mathbf{C} = (C_1, \ldots, C_n)$ kódjelsorozat jelenik meg — ez a csatorna bemenete —, míg a csatorna kimenete a $\mathbf{V} = (V_1, \ldots, V_n)$ jelsorozat. A dekódolás eredménye

az $f^{-1}(g(\mathbf{V})) = \widetilde{\mathbf{Y}}$ visszaállított üzenet, ahol g a döntőfüggvény. A dekódolás hibájának valószínűsége: $\mathbf{P}\{\mathbf{Y} \neq \widetilde{\mathbf{Y}}\} = P_e$.

A csatornakódolási tétel alábbi megfordítása a $\bar{P}_e$ átlagos hibavalószínűségre ad alsó becslést.

3.2. tétel (A csatornakódolási tétel gyenge megfordítása). *A C kapacitású diszkrét memóriamentes csatorna minden R jelsebességű és n szóhosszú kódjára*

$$\bar{P}_e \geq 1 - \frac{C}{R} - \frac{1}{nR}.$$

3.1. következmény. *Ha $R > C$, akkor $\liminf_{n\to\infty} \bar{P}_e > 0$, azaz ha n elég nagy, akkor nem készíthető olyan kód, amelyre a dekódolás hibakorlátja tetszőlegesen kicsi.*

Mielőtt rátérnénk a bizonyításra, egy önmagában is érdekes lemmát látunk be.

3.1. lemma (Fano-egyenlőtlenség). *Tegyük fel, hogy az X és az Y valószínűségi változók értéküket ugyanabból az M elemű halmazból veszik fel. Ha $P_e = \sum_x \sum_{y\neq x} p(x,y)$ annak a valószínűsége, hogy $X \neq Y$, akkor*

$$H(X \mid Y) \leq P_e \log(M-1) + h(P_e),$$

ahol $h(x)$ a bináris entrópiafüggvény.

BIZONYÍTÁS: Bontsuk fel a feltételes entrópia kifejezését két részre:

$$\begin{aligned} H(X \mid Y) &= \sum_x \sum_y p(x,y) \log \frac{p(y)}{p(x,y)} = \\ &= \sum_x \sum_{y\neq x} p(x,y) \log \frac{p(y)}{p(x,y)} + \sum_y p(y,y) \log \frac{p(y)}{p(y,y)}. \end{aligned}$$

Először az első tagot vizsgáljuk. Mivel

$$\sum_x \sum_{y\neq x} p(y) = \sum_x (1 - p(x)) = M - 1,$$

a Jensen-egyenlőtlenség 1.2. következményéből következik, hogy

$$\sum_x \sum_{y\neq x} p(x,y) \log \frac{p(y)}{p(x,y)} \leq P_e \log \frac{M-1}{P_e} = P_e \log(M-1) + P_e \log \frac{1}{P_e}.$$

A második tag: mivel $\sum_y p(y,y) = 1 - P_e$, ezért szintén az 1.2. következményből:

$$\sum_y p(y,y) \log \frac{p(y)}{p(y,y)} \le (1-P_e) \log \frac{1}{1-P_e},$$

amivel az állítást beláttuk. ■

A 3.2. TÉTEL BIZONYÍTÁSA: A kölcsönös információ tulajdonságaiból (2.8. d)-ből) következik, hogy az $\mathbf{Y}$ üzenet és a dekódolt $\widetilde{\mathbf{Y}}$ kölcsönös információja:

$$I(\mathbf{Y};\widetilde{\mathbf{Y}}) = I(f(\mathbf{Y}); f^{-1}(g(\mathbf{V}))) \le I(f(\mathbf{Y});\mathbf{V}) = I(\mathbf{C};\mathbf{V}), \tag{3.1}$$

ugyanis az f függvény kölcsönösen egyértelmű.

Legyen az $\mathbf{U} = \mathbf{C}$ eloszlása:

$$\mathbf{P}\{\mathbf{U} = \mathbf{u}\} = p(\mathbf{u}), \qquad \mathbf{u} = (u_1, \ldots, u_n) \in \mathcal{U}^n.$$

Ekkor, mivel a csatorna memóriamentes, azaz

$$p(\mathbf{v} \mid \mathbf{u}) = \prod_{i=1}^{n} p(v_i \mid u_i),$$

ezért

$$\begin{aligned} H(\mathbf{V} \mid \mathbf{U}) &= \sum_{\mathbf{u}} \sum_{\mathbf{v}} p(\mathbf{u}) p(\mathbf{v} \mid \mathbf{u}) \log \frac{1}{p(\mathbf{v} \mid \mathbf{u})} = \\ &= \sum_{\mathbf{u}} \sum_{\mathbf{v}} p(\mathbf{u}) p(\mathbf{v} \mid \mathbf{u}) \sum_{i=1}^{n} \log \frac{1}{p(v_i \mid u_i)} = \\ &= \sum_{i=1}^{n} \sum_{\mathbf{u}} \sum_{\mathbf{v}} p(\mathbf{u},\mathbf{v}) \log \frac{1}{p(v_i \mid u_i)} = \\ &= \sum_{i=1}^{n} H(V_i \mid U_i). \end{aligned}$$

Másrészt pedig tudjuk, hogy

$$H(\mathbf{V}) \le \sum_{i=1}^{n} H(V_i),$$

tehát

$$I(\mathbf{U};\mathbf{V}) \le \sum_{i=1}^{n} (H(V_i) - H(V_i \mid U_i)) = \sum_{i=1}^{n} I(U_i;V_i) \le nC, \tag{3.2}$$

ahol a második egyenlőtlenség a csatornakapacitás definíciójából következik.

Tegyük fel, hogy az $\mathbf{Y}$ valószínűségi változó egyenletes eloszlású, vagyis minden üzenet egyformán $\frac{1}{M}$ valószínűséggel kerül átvitelre. Ebben az esetben $H(\mathbf{Y}) = \log M$, és $P_e = \bar{P_e}$, tehát a Fano-egyenlőtlenségből, (3.1)-ből és (3.2)-ből következik, hogy

$$\bar{P_e}\log(M-1) + h(\bar{P_e}) \geq H(\mathbf{Y} \mid \widetilde{\mathbf{Y}}) = \log M - I(\mathbf{Y}, \widetilde{\mathbf{Y}}) \geq \log M - nC.$$

Felhasználva, hogy $\log(M-1) < \log M = nR$, illetve, hogy $h(\bar{P_e}) \leq 1$,

$$\bar{P_e}R + \frac{1}{n} \geq R - C,$$

és ezzel az állítást beláttuk. ■

MEGJEGYZÉS: A tétel fő következménye az, hogy $R > C$ esetén $\liminf_{n\to\infty} \bar{P_e} > 0$. Igaz azonban az ennél sokkal erősebb $\liminf_{n\to\infty} \bar{P_e} = 1$ állítás is, amelynek bizonyítását itt nem ismertetjük. Ezért nevezik a 3.2. tételt a kódolási tétel gyenge megfordításának.

A következő tételt, amelyet először C.E. Shannon mondott ki, az információelmélet egyik alaptételének is szokás nevezni.

3.3. tétel (Csatornakódolási tétel). *Tekintsünk egy C kapacitású diszkrét memóriamentes csatornát. Ekkor bármely $r < C$ és $\varepsilon > 0$ számhoz létezik olyan $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_M\}$ csatornakód — amelynek kódszavai n hosszúságúak —, hogy*

a) $\bar{P_e} < \varepsilon$

b) $M > 2^{rn}$, *azaz a* $\frac{\log M}{n}$ *jelsebesség nagyobb, mint* r.

BIZONYÍTÁS: Legyen $\Omega = \mathcal{U}^n \times \mathcal{V}^n$, azaz az összes olyan $(\mathbf{u}, \mathbf{v})$ pár halmaza, ahol $\mathbf{u} = (u_1, \ldots, u_n)$ és $\mathbf{v} = (v_1, \ldots, v_n)$ a csatorna bemeneti, illetve kimeneti ábécéjének elemeiből alkotott n-hosszúságú sorozatok. Definiáljunk az Ω halmazon egy valószínűségeloszlást a következőképpen:

$$p(\mathbf{u}, \mathbf{v}) = p(\mathbf{u})p(\mathbf{v} \mid \mathbf{u}),$$

ahol $p(\mathbf{v} \mid \mathbf{u}) = \prod_{i=1}^{n} p(v_i \mid u_i)$, ahol a $p(v_i \mid u_i)$-k a csatorna átmenetvalószínűségei, továbbá $p(\mathbf{u}) = \prod_{i=1}^{n} p(u_i)$, ahol $p(u)$ $(u \in \mathcal{U})$ éppen az a bemeneti eloszlás, amelyre $I(U;V) = C$ (lásd a 3.4. definíció utáni megjegyzést).

Legyen most R olyan, hogy $r < R < C$. Definiáljuk a $T \subseteq \Omega$ halmazt a következőképpen:

$$T = \{(\mathbf{u}, \mathbf{v}) : \ i(\mathbf{u}, \mathbf{v}) \geq nR\},$$

ahol

$$i(\mathbf{u}, \mathbf{v}) = \log \frac{p(\mathbf{v} \mid \mathbf{u})}{p(\mathbf{v})}$$

az $\mathbf{u}$ és $\mathbf{v}$ elemi kölcsönös információja.

A T halmazra gondolhatunk úgy, mint azon $(\mathbf{u}, \mathbf{v})$ párok halmazára, amelyekben $\mathbf{u}$ és $\mathbf{v}$ bizonyos értelemben „közel vannak egymáshoz", hiszen nagy az elemi kölcsönös információjuk.

Az optimális maximum-likelihood döntés helyett egy másik döntést használunk. A maximum-likelihood döntésnél $p(\mathbf{v} \mid \mathbf{c}_i)$-t maximalizáltuk. Ennél a döntésnél csak akkor döntünk, ha

$$\max_i \frac{p(\mathbf{v} \mid \mathbf{c}_i)}{p(\mathbf{v})} \geq 2^{nR}.$$

Most tegyük fel, hogy adott egy tetszőleges $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_M\}$ kódunk. A T halmaz segítségével — a kód jóságától függetlenül — állapítsuk meg a dekódolási szabályt (azaz a döntőt, lásd a 3.7. definíciót) a következőképpen:

Amennyiben a csatorna kimenetén a $\mathbf{v}$ sorozat jelenik meg, tekintsük a bemeneti sorozatok következő $S(\mathbf{v}) \subset \mathcal{U}^n$ halmazát:

$$S(\mathbf{v}) = \{\mathbf{u} : \ (\mathbf{u}, \mathbf{v}) \in T\}$$

(amely tehát a $\mathbf{v}$-hez „közel eső" bemeneti sorozatokat tartalmazza). Amennyiben $S(\mathbf{v})$ pontosan egy $\mathbf{c}_i$ kódszót tartalmaz, akkor $\mathbf{c}_i$-re döntünk, azaz $g(\mathbf{v}) = \mathbf{c}_i$. Különben, azaz ha $S(\mathbf{v})$ több kódszót tartalmaz, vagy ha egyet sem, akkor tetszőleges kódszóra döntünk, azaz szándékosan hibázunk.

Láthatjuk, hogy a fenti dekódolási szabály mellett, amennyiben a $\mathbf{c}_i$ kódszó került a csatorna bemenetére, kétféleképpen történhet hiba: ha a kimeneten megjelenő $\mathbf{v}$ sorozatra $\mathbf{c}_i \notin S(\mathbf{v})$, vagy ha valamely $i \neq j$-re $\mathbf{c}_j \in S(\mathbf{v})$.

Ezért a feltételes hibavalószínűség felülről becsülhető:

$$P_{e,i} \leq \mathbf{P}\{\mathbf{c}_i \notin S(\mathbf{V}) \mid \mathbf{U} = \mathbf{c}_i\} + \sum_{\substack{j=1 \\ j \neq i}}^{M} \mathbf{P}\{\mathbf{c}_j \in S(\mathbf{V}) \mid \mathbf{U} = \mathbf{c}_i\}.$$

A kifejezés jobb oldalát átalakítva:

$$\begin{aligned} P_{e,i} &\leq \sum_{\mathbf{v}} (1 - I_T(\mathbf{c}_i, \mathbf{v}))\, p(\mathbf{v} \mid \mathbf{c}_i) + \sum_{j \neq i} \sum_{\mathbf{v}} I_T(\mathbf{c}_j, \mathbf{v}) p(\mathbf{v} \mid \mathbf{c}_i) = \\ &= Q_i(\mathbf{c}_1, \ldots, \mathbf{c}_M), \end{aligned}$$

ahol

$$I_T(\mathbf{u},\mathbf{v}) = \begin{cases} 1, & \text{ha } (\mathbf{u},\mathbf{v}) \in T \\ 0, & \text{ha } (\mathbf{u},\mathbf{v}) \notin T \end{cases},$$

azaz a $P_{e,i}$ $(i = 1,\ldots,M)$ feltételes hibavalószínűségek felülről becsülhetők a kódszavak igen bonyolult Q_i függvényeivel. Olyan kódot szeretnénk találni (vagy legalább a létezését bizonyítani), amelyre $Q_i(\mathbf{c}_1,\ldots,\mathbf{c}_M)$ minden i-re kicsi, ekkor ugyanis $\overline{P_e}$ is kicsi. Sajnos a Q_i függvények száma és bonyolultsága olyan nagy lehet, hogy hagyományos minimalizálási módszerekkel reménytelen „jó" kódot találni. Szerencsére azonban létezik egy meglepő technika, amellyel jó kódot készíteni ugyan nem tudunk, azonban a létezését be tudjuk látni. Ez a technika az ún. véletlen kódolás. Ennek lényege röviden az, hogy a kódszavakat véletlenszerűen sorsoljuk, és az így készített kódra vizsgáljuk a $Q_i(\mathbf{c}_1,\ldots,\mathbf{c}_M)$ valószínűségi változókat. Meglepő, de kiderül, hogy a Q_i valószínűségi változók várható értéke minden i-re nullához tart, ha $M = \lceil 2^{rn} \rceil$ és $n \to \infty$. Ebből pedig már könnyű lesz belátni a kívánt tulajdonságú kód létezését.

Az első kérdés nyilván az, hogy milyen eloszlás szerint sorsoljuk ki az egyes $\mathbf{c}_i = (c_{i1},\ldots,c_{in})$ $(i = 1,\ldots,M)$ kódszavakat: Legyen egy $\mathcal{C} = \{\mathbf{c}_1,\ldots,\mathbf{c}_M\}$ kód valószínűsége:

$$p(\mathbf{c}_1,\ldots,\mathbf{c}_M) = \prod_{i=1}^{M} p(\mathbf{c}_i) = \prod_{i=1}^{M}\prod_{j=1}^{n} p(c_{ij}),$$

ahol $p(\cdot)$ az az eloszlás a bemeneti ábécén, amelyre a be- és kimenet kölcsönös információja eléri a csatornakapacitást.

Képezzük most a $Q_i(\mathbf{C}_1,\ldots,\mathbf{C}_M)$ valószínűségi változó várható értékét a fenti eloszlás szerint sorsolt kódra:

$$\begin{aligned} \mathbf{E}(Q_i) &= \mathbf{E}\left[\sum_{\mathbf{v}} (1 - I_T(\mathbf{C}_i,\mathbf{v}))\, p(\mathbf{v} \mid \mathbf{C}_i)\right] + \\ &\quad \sum_{j\neq i} \mathbf{E}\left[\sum_{\mathbf{v}} I_T(\mathbf{C}_j,\mathbf{v}) p(\mathbf{v} \mid \mathbf{C}_i)\right] = \\ &= E_1 + \sum_{j\neq i} E_{2,j}. \end{aligned}$$

Először E_1-et vizsgáljuk:

$$\begin{aligned} E_1 &= \sum_{\mathbf{c}_1,\ldots,\mathbf{c}_M} p(\mathbf{c}_1,\ldots,\mathbf{c}_M) \sum_{\mathbf{v}} (1 - I_T(\mathbf{c}_i,\mathbf{v}))\, p(\mathbf{v} \mid \mathbf{c}_i) = \\ &= \sum_{\mathbf{c}_i,\mathbf{v}} p(\mathbf{c}_i)\, p(\mathbf{v} \mid \mathbf{c}_i)\, (1 - I_T(\mathbf{c}_i,\mathbf{v})) = \\ &= \sum_{\mathbf{u},\mathbf{v}} p(\mathbf{u},\mathbf{v})\, (1 - I_T(\mathbf{u},\mathbf{v})) = \\ &= \mathbf{P}\{(\mathbf{U},\mathbf{V}) \notin T\}, \end{aligned}$$

tehát $E_1 = \mathbf{P}\{i(\mathbf{U},\mathbf{V}) < nR\}$, viszont

$$i(\mathbf{u},\mathbf{v}) = \log\frac{p(\mathbf{v}\mid\mathbf{u})}{p(\mathbf{v})} = \sum_{k=1}^{n}\log\frac{p(v_k\mid u_k)}{p(v_k)} = \sum_{k=1}^{n} i(u_k,v_k).$$

(Az itt szereplő eloszlások definíciója alapján gondoljuk át, miért igaz a $p(\mathbf{v}) = \prod_{k=1}^{n} p(v_k)$ egyenlőség.)

Láthatjuk, hogy az $i(\mathbf{U},\mathbf{V})$ valószínűségi változó n darab független, azonos eloszlású valószínűségi változó összege, amelyek várható értéke: $\mathbf{E}(i(U_k,V_k)) = I(U,V) = C$.

Mivel $R < C$, a nagy számok gyenge törvényéből adódik, hogy

$$\lim_{n\to\infty}\mathbf{P}\left\{\frac{1}{n}\sum_{k=1}^{n} i(U_k,V_k) < R\right\} =$$
$$= \lim_{n\to\infty}\mathbf{P}\left\{\frac{1}{n}\sum_{k=1}^{n}\left(i(U_k,V_k) - \mathbf{E}i(U_k,V_k)\right) < R - C\right\} = 0,$$

tehát

$$\lim_{n\to\infty}\mathbf{P}\{i(\mathbf{U},\mathbf{V}) < nR\} = 0, \tag{3.3}$$

azaz E_1 nullához tart, ahogy n tart a végtelenhez. Most megpróbáljuk $E_{2,j}$-t felülről becsülni.

$$\begin{aligned} E_{2,j} &= \sum_{\mathbf{c}_1,\ldots,\mathbf{c}_M} p(\mathbf{c}_1,\ldots,\mathbf{c}_M)\sum_{\mathbf{v}} I_T(\mathbf{c}_j,\mathbf{v})p(\mathbf{v}\mid\mathbf{c}_i) = \\ &= \sum_{\mathbf{c}_j,\mathbf{v}} p(\mathbf{c}_j)I_T(\mathbf{c}_j,\mathbf{v})\sum_{\mathbf{c}_i} p(\mathbf{c}_i)p(\mathbf{v}\mid\mathbf{c}_i) = \\ &= \sum_{\mathbf{c}_j,\mathbf{v}} p(\mathbf{c}_j)p(\mathbf{v})I_T(\mathbf{c}_j,\mathbf{v}), \end{aligned}$$

azaz

$$E_{2,j} = \sum_{(\mathbf{u},\mathbf{v})\in T} p(\mathbf{u})p(\mathbf{v}).$$

A T halmaz definíciója szerint, ha $(\mathbf{u},\mathbf{v}) \in T$, akkor

$$p(\mathbf{u})p(\mathbf{v}) \le p(\mathbf{u},\mathbf{v})2^{-nR},$$

így

$$E_{2,j} \le 2^{-nR}\sum_{(\mathbf{u},\mathbf{v})\in T} p(\mathbf{u},\mathbf{v}) \le 2^{-nR}.$$

Összegezve az eredményeket:

$$\mathbf{E}(Q_i) = E_1 + \sum_{i \neq j} E_{2,j} \leq \mathbf{P}\{i(\mathbf{U},\mathbf{V}) < nR\} + M2^{-nR}. \tag{3.4}$$

Megjegyezzük, hogy $\mathbf{E}(Q_i)$ nem függ i-től. Válasszuk most a kódszavak számát $M = \lceil 2^{nr} \rceil$-re. Ekkor $M2^{-nR} \leq 2 \cdot 2^{-n(R-r)}$. Mivel $r < R < C$, ezért (3.3) miatt a (3.4) kifejezés jobb oldala tetszőlegesen kicsivé tehető, azaz található olyan nagy n kódszóhossz, hogy az $M = \lceil 2^{nr} \rceil$ méretű kódokra minden Q_i várható értéke: $\mathbf{E}(Q_i) < \varepsilon$ $(i = 1, \ldots, M)$.

Azt tehát beláttuk, hogy a $\overline{P_e} = \frac{1}{M} \sum_{i=1}^{M} P_{e,i}$ átlagos hibavalószínűség (amely persze a véletlen kódválasztás miatt szintén valószínűségi változó) várható értéke:

$$\mathbf{E}(\overline{P_e}) < \varepsilon,$$

amiből nyilvánvalóan következik, hogy létezik olyan konkrét kód, a véletlen kód egy realizációja, amelyre $\overline{P_e} < \varepsilon$. ■

MEGJEGYZÉS: A csatornakódolási tétel imént bizonyított formája nem mond semmit arról, hogy $R < C$ esetén a $\overline{P_e}$ hibavalószínűség milyen sebességgel tart nullához, azaz, hogy egy adott $\varepsilon > 0$-hoz milyen szóhosszúságú kóddal érhető el ε-nál kisebb hibavalószínűség. Itt nem bizonyítjuk, csak megemlítjük, hogy létezik olyan eredmény, amely az exponenciális konvergenciasebességet mondja ki, pontosabban:

$$\overline{P_e} \leq \mathrm{e}^{-E(R)n},$$

ahol $E(R)$ alulról konvex függvény, és $R < C$ esetén $E(R) > 0$.

3.5. Feladatok

3.1. feladat (Paritásbit használata visszacsatolásos csatornán).
A gyakorlatban rendkívül gyakran használt kódolási módszer az, hogy az üzenet bináris sztringet egy bittel meghosszabbítjuk, és ez a bit éppen a sztring bitjeinek bináris összege (paritásbit). Ez a kód természetesen hibajavításra nem alkalmas, de ha páratlan számú hiba történik az átvitel során, akkor azt a dekódoló észreveszi, és ha van visszacsatolás a csatornán, akkor az üzenet ismételt küldését kérheti. A legegyszerűbb eset a következő: Tekintsünk egy bináris szimmetrikus csatornát p hibavalószínűséggel. Csakúgy, mint az előző példában, az üzenet a 0 és 1 szimbólumok valamelyike. A kódolási szabály szerint, ha az üzenet 0, akkor a (00) kódszót küldjük át a csatornán, míg 1 esetén az (11) kódszót. Dekódoláskor, ha a két vett bit megegyezik, akkor annak megfelelően döntünk, ha pedig nem, akkor újabb küldést igénylünk.

a) Bizonyítsa be, hogy ekkor a dekódolás hibavalószínűsége

$$P_e = P_{e,1} = P_{e,2} = \frac{p^2}{1-2p+2p^2},$$

továbbá, hogy egy üzenetbit továbbításához átlagosan

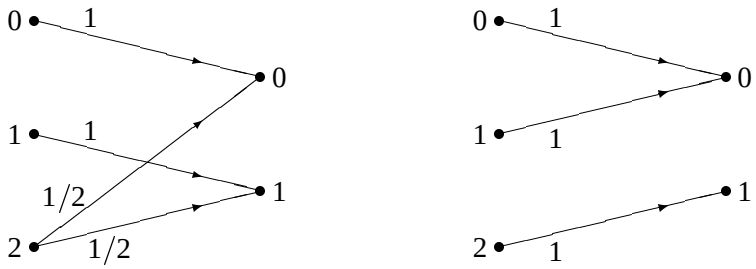
$$\frac{2}{1-2p+2p^2}$$

bitet kell a csatornán átküldeni.

b) Hasonlítsa össze ezeket a számokat az előző példa eredményével! (Meglepő, de bebizonyítható, hogy a visszacsatolás nem növeli a csatornakapacitást, azaz, visszacsatolás nélkül is mindig elérhető ugyanaz a jelsebesség és hibavalószínűség, legfeljebb hosszabb blokkokat kell használni.)

3.2. feladat.

a) Mennyi az ábrán látható két csatorna kapacitása?



b) Tegyük fel most, hogy a két csatornát egyszerre használjuk, azaz a bemeneti 0,1 vagy 2 szimbólumot mindkét csatornán továbbítjuk. Így egy olyan csatornához jutottunk, amelynek három lehetséges bemenete, és négy kimenete van. Mennyi az új csatorna kapacitása? Mutassa meg, hogy mindhárom csatorna esetén van olyan $R = C$ jelsebességű kód, amely nulla hibavalószínűséggel dekódolható!

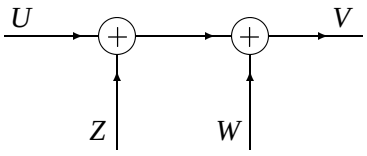
3.3. feladat (mod 11 csatorna). Legyen egy diszkrét memóriamentes csatorna bemenete $U \in \{0,1,\ldots,10\}$. A csatorna kimenetét, V-t a $V = U + Z \pmod{11}$ egyenlet határozza meg, ahol Z független U-tól és

$$\mathbf{P}\{Z=1\} = \mathbf{P}\{Z=2\} = \mathbf{P}\{Z=3\} = \frac{1}{3}.$$

a) Határozza meg a csatorna kapacitását!

b) Adjon maximális jelsebességű, egy hosszú blokkokat kódoló csatornakódot, melyet 0 hibavalószínűséggel lehet dekódolni! Mekkora az így elérhető legnagyobb jelsebesség?

3.4. feladat. Az ábrán látható blokksémán U, Z és W független bináris valószínűségi változók, $\oplus$ pedig modulo 2 összeadást jelöl. Az elrendezés meghatároz egy bináris csatornát, melynek bemenete U, kimenete V.



a) Mennyi a csatorna kapacitása, ha $\mathbf{P}\{Z=1\}=p$ és $\mathbf{P}\{W=1\}=q$?

b) Ha az első $\oplus$ műveletet kicseréljük bináris „vagy” műveletre, a másodikat bináris „és” műveletre, akkor $p=\frac{1}{3}$ és $q=\frac{3}{4}$ esetén mi a csatorna kapacitása?

3.5. feladat (BSC-k kaszkádja). Kapcsoljunk egymás után k darab bináris szimmetrikus csatornát, melyek mindegyikének hibavalószínűsége p. Mutassa meg, hogy a kapott csatorna ekvivalens egy bináris szimmetrikus csatornával, amelynek hibavalószínűsége $\frac{1}{2}(1-(1-2p)^k)$, azaz a csatorna kapacitása nullához tart, ahogy k tart végtelenhez. (Az információelmélet megszületésekor az jelentette az egyik legnagyobb meglepetést, hogy ha az egyes csatornák közé kódolókat és dekódolókat teszünk, akkor az egyes elemi csatornák kapacitásának megfelelő jelsebesség mellett tetszőlegesen kis hibavalószínűség érhető el. Más szóval, az átvitel tulajdonságai megjavíthatóak az átvivő közegbe iktatott kódolók és dekódolók segítségével, anélkül, hogy a közeg fizikai tulajdonságait megváltoztatnánk.)

3.6. feladat (Törléses csatorna kapacitása). Bizonyítsa be, hogy a p hibavalószínűségű bináris törléses csatorna kapacitása $C=1-p$.

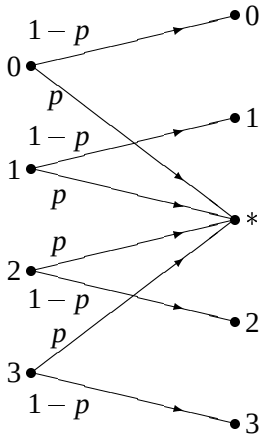
3.7. feladat (Gyengén szimmetrikus csatornák kapacitása). A csatornamátrix a $p(v \mid u)$ átmenetvalószínűségeket tartalmazza ($u \in \mathcal{U}, v \in \mathcal{V}$), mégpedig úgy, hogy az i-edik sor j-edik oszlopában a $p(v_j \mid u_i)$ valószínűség áll. Egy diszkrét memóriamentes csatornát gyengén szimmetrikusnak nevezünk, ha csatornamátrixának minden sora ugyanannak a $\mathbf{p}$ valószínűségi vektornak egy permutációja, továbbá az oszlopokban álló valószínűségek összege állandó. Mutassa meg, hogy egy ilyen csatorna kapacitása

$$C=\log|\mathcal{V}|-H(\mathbf{p}).$$

3.8. feladat.

a) Mennyi az ábrán látható csatorna kapacitása?

b) Az előző csatornánál legyen $p = \frac{1}{4}$. Tegyük fel, hogy a csatornát összesen 30-szor használhatjuk. Közelítőleg hány bites bináris üzenetek átvitelét valósíthatjuk így meg, ha az üzenet meghibásodásának valószínűségét kis szinten akarjuk tartani?



3.9. feladat (Kapacitás fölötti jelsebesség). Tegyük fel, hogy egyenletes eloszlással véletlenszerűen sorsolt n hosszúságú bináris sztringeket kódolás nélkül továbbítunk egy p hibavalószínűségű bináris szimmetrikus csatornán. Világos, hogy ekkor a jelsebesség nagyobb, mint a csatornakapacitás. Mekkora a dekódolás hibavalószínűsége? Mekkorára érdemes az n blokkhosszt választani?

3.10. feladat (MAP és ML dekódolás). Vegyünk egy bináris szimmetrikus csatornát p hibavalószínűséggel. Tegyük fel, hogy két üzenet egyikét akarjuk továbbítani, és n hosszúságú ismétléses kódot alkalmazunk, azaz az első üzenet esetén az n darab nullából, míg a második esetén az n darab egyesből álló kódszót küldjük át a csatornán. Tudjuk, hogy a maximum-likelihood dekódolás azt a kódszót választja ki, amelynek a csatorna kimenetén vett sorozattól mért Hamming-távolsága kisebb. Tudjuk, hogy a legkisebb hibavalószínűségű dekódolás a maximum a posteriori dekódolás. Tegyük most fel, hogy a két üzenet küldésének valószínűsége q, illetve $1-q$. Állapítsa meg a maximum a posteriori dekódolás szabályát! Például $p = \frac{1}{3}$ és $q = 0.1$ esetén határozza meg a dekódolási szabályt n több értékére, pl. $n = 3, 10, 100$ esetén.

3.11. feladat (ML dekódolás törléses csatornán). Állapítsa meg a maximum-likelihood dekódolás szabályát bináris törléses csatornán!

4. fejezet

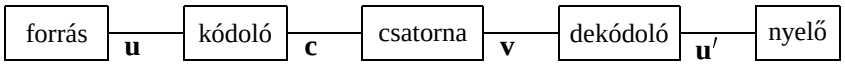
Hibajavító kódolás

A hibajavító kódok és a titkosítás egészen a ’80-as évek elejéig kívül rekedt a polgári alkalmazások körén. A kódelmélet esetén ennek elsősorban technológiai okai voltak, ezeket az akadályokat a mikroelektronika tömeges elterjedése söpörte el, míg a titkosítás azért törhetett be a civil területre, mert megszületett az utóbbi évek információtechnikájának egyik legeredetibb ötlete, a nyilvános kulcsú titkosítás.

A következő két fejezet célja éppen az, hogy ismertesse azon eljárások elméleti alapjait, melyek már elterjedtek vagy elterjedésük megkezdődött a távközlési és informatikai tömegszolgáltatásokban.

Ez a rész igyekszik összefoglalni a hibajavító kódok elméletéből azokat az eredményeket, melyek egyrészt viszonylag egyszerű matematikai alapokkal már megérthetők, másrészt a hírközlési illetve informatikai tömegszolgáltatásokban alapvető fontosságúak.

A hibajavító kódok története az információelmélet kialakulásával kezdődött, nevezetesen, amikor Shannon bebizonyította a csatornakódolási tételt, vagyis megtalálta a zajos csatorna megbízható használatának az elvi korlátját, a csatornakapacitást, és az alkalmazott bizonyítás nem adott eljárást a csatornakapacitást közelítő hatékony kódok konstrukciójára. Az ’50-es években fedezték fel az alapvető blokk-kódokat (Hamming, BCH, Reed–Solomon, stb.), míg a ’60-as évekre tehető a konvolúciós kódok és a Viterbi-algoritmus bevezetése. Egészen a ’70-es évek második feléig a hibajavító kódolás fő felhasználási területei a katonai hírközlés és a nagymegbízhatóságú irányító rendszerek voltak, csak a mikroprocesszorok elterjedésével nyílt meg az út a tömeges polgári alkalmazások felé.



4.1. ábra. Hírközlési rendszer blokkdiagramja.

4.1. Kódolási alapfogalmak

A hibajavító kódolás alapvető módszereit a 4.1. ábrán látható egyszerű hírközlési struktúra kapcsán vizsgáljuk.

Az $\mathbf{u}$ és $\mathbf{u}'$ vektorok koordinátái egy F halmazból veszik értékeiket, mely halmazt **forrásábécé**nek nevezzük. A kódoló a k hosszú $\mathbf{u}$ vektort (az **üzenet**et) egy n hosszú $\mathbf{c}$ vektorba (a **kódszó**ba) képezi le. A $\mathbf{c}$ koordinátái egy Q halmazból veszik értékeiket. A Q-t **kódábécé**nek vagy csatorna bemeneti ábécének fogjuk hívni. A csatorna kimenete $\mathbf{v}$, szintén egy n hosszú vektor, melynek koordinátái szintén Q-beliek.

Egy

$$\mathbf{c} = (c_1, \ldots, c_n)$$

bemeneti és

$$\mathbf{v} = (v_1, \ldots, v_n)$$

kimeneti sorozat esetén azt mondjuk, hogy az m-edik időpontban a csatorna hibázott, ha $c_m \neq v_m$. Jelölje $d(\mathbf{c}, \mathbf{v})$ azon i pozíciók számát, ahol $c_i \neq v_i$.

$d(\mathbf{c}, \mathbf{v})$ neve a $\mathbf{c}, \mathbf{v}$ sorozatok **Hamming-távolság**a, és azt mondjuk, hogy a $\mathbf{c}$ sorozat küldésekor és a $\mathbf{v}$ sorozat vételekor a hibák száma $t = d(\mathbf{c}, \mathbf{v})$. Ezt az esetet nevezzük **egyszerű hibázás**nak, amikor a hiba helye és értéke egyaránt ismeretlen. $d(\mathbf{c}, \mathbf{v})$ valóban távolság, hiszen

$$d(\mathbf{c}, \mathbf{v}) \geq 0,$$

$$d(\mathbf{c}, \mathbf{v}) = d(\mathbf{v}, \mathbf{c}),$$

és igaz a háromszög-egyenlőtlenség:

$$d(\mathbf{c}, \mathbf{v}) \leq d(\mathbf{c}, \mathbf{w}) + d(\mathbf{w}, \mathbf{v}).$$

Kód (blokk-kód) alatt a Q^n halmaz egy C részhalmazát értjük, azaz C minden eleme egy n hosszú vektor, melynek koordinátái Q-beliek. C elemeit kódszavaknak nevezzük. A **kódolás** egy invertálható függvény, mely k hosszú F-beli sorozatot — üzenetet — képez le egy kódszóba, formalizálva:

$$f : F^k \to C,$$

és minden különböző $\mathbf{u}$, $\mathbf{u}'$-re $f(\mathbf{u})$, $f(\mathbf{u}')$ is különböző.

Dekódolás alatt két függvény egymásutánját értjük. Az egyik a csatorna kimenetének n hosszú szegmensét képezi le C-be, azaz igyekszik eltalálni a küldött kódszót, a másik pedig az f függvény inverze, tehát

$$g : Q^n \to C, \qquad f^{-1} : C \to F^k.$$

Mivel f egyértelműen meghatározza f^{-1}-et, ezért dekódolás alatt a későbbiekben csak a g függvényt értjük. A g dekódoló függvényként az algebrai hibajavító kódok elméletében speciális függvényt választunk, nevezetesen a $\mathbf{v}$ vektorhoz megkeressük azt a $\mathbf{c}' \in C$ kódszót, mely Hamming-távolság szerint hozzá a legközelebb van, vagy ha több ilyen van, akkor az egyiket, tehát teljesül, hogy ha $\mathbf{c}' = g(\mathbf{v})$, akkor

$$d(\mathbf{c}', \mathbf{v}) = \min_{\mathbf{c} \in C} d(\mathbf{c}, \mathbf{v}).$$

Az előző fejezetben megmutattuk, hogy bináris szimmetrikus csatorna esetén ez a dekódolás optimális. Más csatornánál ez a dekódolás már nem feltétlenül optimális.

A dekódolás feladata ezek után arra a messze nem triviális feladatra szűkül, hogy egy $\mathbf{v}$ vett szóhoz hogyan keressük meg a hozzá legközelebbi $\mathbf{c}'$ kódszót anélkül, hogy minden $d(\mathbf{c}, \mathbf{v})$-t kiszámítanánk. Ha mégis kiszámítjuk ezeket a távolságokat, és minden $\mathbf{v}$-hez megkeressük a hozzá legközelebbi $\mathbf{c}$ kódszót, majd a neki megfelelő üzenetet, akkor elvben azt eltárolhatjuk, és így egy táblázathoz jutunk, melynek címét $\mathbf{v}$ adja, tartalma pedig a $\mathbf{v}$-nek megfelelő dekódolt üzenet. Ez a **táblázatos dekódolás**nak a legegyszerűbb, de legpazarlóbb esete, hiszen a táblázat q^n darab üzenetből áll, ahol q a Q kódábécé elemszáma.

4.1. példa. Tekintsük azt a feladatot, amikor a forrás a következő négy lehetséges üzenetet bocsátja ki: alma, körte, szilva, cseresznye, melyeket rendre 00, 01, 10, 11 sorozatokkal forráskódolunk. A 2 hosszú forrásszegmensekhez az f kódoló a következő, 5 hosszú kódszavakat rendeli hozzá:

u_1u_2		$c_1c_2c_3c_4c_5$	
0 0	$\to$	0 0 0 0 0	$\mathbf{c}_1$
0 1	$\to$	0 1 1 0 1	$\mathbf{c}_2$
1 0	$\to$	1 0 1 1 0	$\mathbf{c}_3$
1 1	$\to$	1 1 0 1 1	$\mathbf{c}_4$

A g és az f^{-1} függvényt a következő táblázat foglalja össze:

$v_1\,v_2\,v_3\,v_4\,v_5$		$c'_1\,c'_2\,c'_3\,c'_4\,c'_5$		$u'_1\,u'_2$
0 0 0 0 0	$\rightarrow$	0 0 0 0 0	$\rightarrow$	0 0
1 0 0 0 0				
0 1 0 0 0				
0 0 1 0 0				
0 0 0 1 0				
0 0 0 0 1				
0 1 1 0 1	$\rightarrow$	0 1 1 0 1	$\rightarrow$	0 1
1 1 1 0 1				
0 0 1 0 1				
0 1 0 0 1				
0 1 1 1 1				
0 1 1 0 0				
1 0 1 1 0	$\rightarrow$	1 0 1 1 0	$\rightarrow$	1 0
0 0 1 1 0				
1 1 1 1 0				
1 0 0 1 0				
1 0 1 0 0				
1 0 1 1 1				
1 1 0 1 1	$\rightarrow$	1 1 0 1 1	$\rightarrow$	1 1
0 1 0 1 1				
1 0 0 1 1				
1 1 1 1 1				
1 1 0 0 1				
1 1 0 1 0				
0 0 0 1 1	$\rightarrow$	0 0 0 0 0	$\rightarrow$	0 0
0 1 0 1 0				
1 0 0 0 1				
1 1 0 0 0				
0 0 1 1 1	$\rightarrow$	0 1 1 0 1	$\rightarrow$	0 1
0 1 1 1 0				
1 0 1 0 1				
1 1 1 0 0				

Az első 24 kimeneti szó dekódolásakor nincs probléma, hiszen minden 6 szóból álló csoport minden szava olyan, hogy vagy kódszó (az első helyen álló), vagy a kódszó egy bitjének a megváltoztatásával (egy hibával) képződött. A 25–28. szavak mindegyike $\mathbf{c}_1$-től és $\mathbf{c}_4$-től 2, míg $\mathbf{c}_2$-től és $\mathbf{c}_3$-tól 3 távolságra van. Itt önkényesen a dekódolás eredménye $\mathbf{c}_1$ (jobb érvünk nincs, mint az, hogy jobban szeretjük az almát, mint a cseresznyét). A 29–32. szavak mindegyike $\mathbf{c}_1$-től és $\mathbf{c}_4$-től 3, míg $\mathbf{c}_2$-től és $\mathbf{c}_3$-tól 2 távolságra van. Itt (szintén önkényesen) a dekódolás eredménye $\mathbf{c}_2$.

4.2. példa. A 4.1. példában szereplő üzenetekhez rendeljünk 3 hosszú kódszavakat:

u_1u_2		$c_1\,c_2\,c_3$	
0 0	$\to$	0 0 0	$\mathbf{c}_1$
0 1	$\to$	0 1 1	$\mathbf{c}_2$
1 0	$\to$	1 0 1	$\mathbf{c}_3$
1 1	$\to$	1 1 0	$\mathbf{c}_4$

azaz az u_1u_2 bitet kiegészítjük egy paritásbittel. Ez a kód már sokkal szegényesebb, mert ha a $\mathbf{v}$ nem kódszó, akkor 3 kódszóból nyerhető 1 bit megváltoztatásával, azaz 1 hibával.

A későbbiekben kiderül, hogy a kódoló f függvény leglényegesebb tulajdonsága a C kód egy paramétere, amit **kódtávolság**nak nevezünk, és $d_{\min}$-nel jelölünk:

$$d_{\min} = \min_{\substack{\mathbf{c}\neq\mathbf{c}'\\ \mathbf{c},\mathbf{c}'\in C}} d(\mathbf{c},\mathbf{c}').$$

A 4.1. példában $d_{\min} = 3$, míg a 4.2. példában $d_{\min} = 2$.

A **hibajelzés** a hibakorlátozó kódolás azon feladata, amikor a vevőben csupán detektálni akarjuk a hibázás tényét, azaz azt kérdezzük, hogy van-e hiba. Nyilván egy $\mathbf{v}$ vett szó esetén akkor tudjuk a hibázást észrevenni, ha $\mathbf{v}$ nem kódszó, amire garancia, hogy ha $\mathbf{c}$ küldött kódszó esetén

$$d_{\min} > d(\mathbf{v},\mathbf{c}),$$

azaz a hibák számára

$$d_{\min} > t,$$

tehát egy $d_{\min}$ kódtávolságú kód minden, legfeljebb $d_{\min} - 1$ számú hibát jelezni tud.

Mivel a 4.1. példában $d_{\min} = 3$, ezért ez a kód 2 hibát tud jelezni, míg a 4.2. példa kódja $d_{\min} = 2$ miatt 1-et.

Hibajavítás esetén azt kérdezzük, hogy ha t a hibák száma, akkor mi biztosítja, hogy a $\mathbf{v}$ vett szóból a $\mathbf{c}$ küldött kódszó egyértelműen visszaállítható legyen, azaz minden más $\mathbf{c}'$ kódszóra

$$d(\mathbf{v},\mathbf{c}') > d(\mathbf{v},\mathbf{c}) \tag{4.1}$$

legyen. Mivel a Hamming-távolság valóban távolság, ezért teljesíti a háromszög-egyenlőtlenséget, azaz

$$d(\mathbf{v},\mathbf{c}') \geq d(\mathbf{c},\mathbf{c}') - d(\mathbf{v},\mathbf{c}), \tag{4.2}$$

tehát (4.1) úgy biztosítható, hogy

$$d(\mathbf{c},\mathbf{c}') - d(\mathbf{v},\mathbf{c}) > d(\mathbf{v},\mathbf{c}),$$

ugyanis, ha ez utóbbi teljesül, akkor (4.1) is teljesül, azaz minden $\mathbf{c}' \neq \mathbf{c}$-re

$$d(\mathbf{c},\mathbf{c}') > 2d(\mathbf{v},\mathbf{c}),$$

vagyis

$$\frac{d_{\min}}{2} > d(\mathbf{v},\mathbf{c}).$$

Összefoglalva: **egyszerű hibázás** esetén $\left\lfloor \frac{d_{\min}-1}{2} \right\rfloor$ hiba javítható.

A 4.1. példa kódja 1 hibát tud javítani, mégpedig a dekódolási tábla első 24 kimeneti szava esetén, míg a 4.2. példa kódjának hibajavító képessége 0.

Gyakran fordul elő olyan hibázás, amikor tudjuk, hogy egy pozícióban hiba lehet, vagyis tudjuk, hogy más pozíciókban nincs hiba, tehát a hiba helyét ismerjük, csak a hiba értékét nem. Az ilyen hibát **törléses hibá**nak nevezzük. Egyszerűen belátható, hogy minden $d_{\min} - 1$ törléses hiba javítható, ugyanis a legrosszabb esetben sem fordulhat elő, hogy két $\mathbf{c},\mathbf{c}'$ kódszó ugyanazon, de legfeljebb $d_{\min} - 1$ pozíciójának törlésével ugyanazt a szót kapnánk.

A 4.1. példa kódja 2 törléses hibát tud javítani, míg a 4.2. példa kódja 1-et.

Nyilván adott n kódszóhosszúság és $d_{\min}$ kódtávolság esetén nem lehet akármilyen nagy méretű kódot konstruálni:

4.1. tétel (Singleton-korlát). *Egy M kódszóból álló, n hosszú és d_{min} kódtávolságú kódra*

$$M \leq q^{n-d_{min}+1}.$$

BIZONYÍTÁS: Legyen k egy természetes szám, melyre

$$q^{k-1} < M \leq q^k.$$

Mivel a $k-1$ hosszú különböző sorozatok száma q^{k-1}, ezért $q^{k-1} < M$ miatt létezik két kódszó $\mathbf{c}$ és $\mathbf{c}'$, melyek az első $k-1$ koordinátában megegyeznek. Ezekre

$$d(\mathbf{c},\mathbf{c}') \leq n-k+1,$$

következésképpen

$$d_{\min} \leq n-k+1,$$

azaz

$$M \leq q^k \leq q^{n-d_{\min}+1}. \qquad \blacksquare$$

Jellegzetes esetben $M = q^k$, vagyis a kódoló k hosszú forrásszegmensekhez rendel n hosszú vektorokat. Azt mondjuk ilyenkor, hogy a kódunk (n,k) paraméterű. Ebben az esetben a Singleton-korlát alakja

$$d_{\min} \leq n-k+1.$$

4.1. definíció. *Azon kódot, melyre a Singleton-korlátban = áll,* ***maximális távolságú*** *vagy* ***MDS*** *(maximum distance separable) kódnak nevezzük.*

A 4.1. példában $k=2,\ n=5,\ d_{\min}=3$, tehát ez a kód nem MDS kód.

4.2. tétel (Hamming-korlát). *Ha egy (n,k) paraméterű kód t hibát tud javítani, akkor*

$$\sum_{i=0}^{t} \binom{n}{i} (q-1)^i \leq q^{n-k}. \tag{4.3}$$

BIZONYÍTÁS: Egy kódszó közepű gömb álljon azokból a vektorokból, melyek a kódszóból legfeljebb t hibával keletkeznek. $(q-1)^i$ darab olyan vektor van, amely az adott kódszótól valamely fix i pozícióban tér el, ezért egy ilyen gömb $\sum_{i=0}^{t} \binom{n}{i}(q-1)^i$ vektort tartalmaz. A kód akkor tud t hibát javítani, ha a q^k darab kódszó körüli gömbök diszjunktak. Ekkor viszont az összes gömbben levő vektorok száma kisebb vagy egyenlő, mint q^n, tehát

$$q^k \sum_{i=0}^{t} \binom{n}{i} (q-1)^i \leq q^n. \qquad \blacksquare$$

Emlékeztetünk arra, hogy egy kód $t = \left\lfloor \frac{d_{\min}-1}{2} \right\rfloor$ hibát tud javítani.

A 4.2. tétel nem mondja azt, hogy ha $n,\ k,\ t$ kielégíti a (4.3)-at, akkor létezik ilyen paraméterű kód. Azt viszont állítja, hogy ha (4.3) nem teljesül, akkor ilyen kód nincs, tehát (4.3) a kód létezésének szükséges feltétele.

4.2. definíció. *Az olyan kódokat, ahol a Hamming-korlátban = áll,* ***perfekt kód****oknak nevezzük.*

Általában a Hamming-korlátot bináris ($q = 2$) esetben idézik, amikor az a következő egyszerű alakú:

$$\sum_{i=0}^{t} \binom{n}{i} \leq 2^{n-k}.$$

A következő szakaszban mutatunk egy kódot, a bináris Hamming-kódot ($t = 1$), mely perfekt kód, azaz

$$1 + n = 2^{n-k}.$$

4.2. Bináris lineáris kódok, bináris Hamming-kód

Ebben a szakaszban kódok egy fontos csoportjával ismerkedünk meg. A továbbiakban a kódjainkban szereplő kódszavakat alkotó szimbólumok legyenek 0 vagy 1 értékűek, az összeadás és a szorzás pedig a bináris összeadás és a bináris szorzás, azaz a modulo 2 összeadás és a modulo 2 szorzás.

Vezessük be a lineáris kód fogalmát:

4.3. definíció. *Egy C kód* ***lineáris****, ha a C halmaza lineáris tér, azaz ha minden $\mathbf{c}, \mathbf{c}' \in C$-re $\mathbf{c} + \mathbf{c}' \in C$.*

A lineáris kód definíciójából következik, hogy a $\mathbf{0}$ vektor eleme minden lineáris kódnak, vagyis minden lineáris kód esetén a $\mathbf{0}$ kódszó. Egyszerűen belátható, hogy a 4.1. és 4.2. példa kódja lineáris.

A lineáris kódok jelentőségét az adja, hogy az egyes üzenetekhez tartozó kódszavak viszonylag egyszerűen generálhatók, és ugyancsak egyszerű módszer található a vett kódszavak hibamentességének vizsgálatára, vagyis a hibadetektálásra, és a hibák javítása sem bonyolult. A következőkben e módszereket fogjuk bemutatni.

Jelentsen C továbbra is egy lineáris kódot, a kódszóhossz legyen n. Ekkor C az n hosszúságú bináris koordinátájú vektorok terének egy altere; „kódszó” helyett gyakran „vektor”-t fogunk mondani.

A valós vektortérben megszokott lineáris függetlenség és bázis fogalmak itt is teljesen hasonlóan értelmezhetők, vagyis

4.4. definíció. *A $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_j \in C$ vektorok lineárisan függetlenek, ha $\alpha_i \in \{0, 1\}$ mellett*

$$\sum_{i=1}^{j} \alpha_i \mathbf{g}_i = 0$$

csak úgy állhat elő, ha $\alpha_i = 0$ minden $i = 1, 2, \ldots, j$-re.

4.5. definíció. *A* $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_k \in C$ *vektorok a* C *lineáris tér egy bázisát alkotják, ha lineárisan függetlenek, továbbá igaz az, hogy minden* $\mathbf{c} \in C$ *vektor előállítható*

$$\mathbf{c} = \sum_{i=1}^{k} u_i \mathbf{g}_i \tag{4.4}$$

alakban, ahol $u_i \in \{0,1\}$ *minden* $i = 1,2,\ldots,k$*-ra.*

Az utóbbi definícióban a bázist alkotó vektorok lineáris függetlenségéből következik, hogy a kódszavak fenti típusú előállítása egyértelmű is, ha ugyanis létezne két különböző előállítás valamely $\mathbf{c} \in C$-re, tehát

$$\mathbf{c} = \sum_{i=1}^{k} u_i \mathbf{g}_i$$

és

$$\mathbf{c} = \sum_{i=1}^{k} y_i \mathbf{g}_i,$$

ahol nem áll fönn $u_i = y_i$ minden i-re, akkor a két egyenletet kivonva egymásból a nullvektornak egy nem triviális előállítását kapnánk a bázisvektorokkal, ami ellentmondana azok lineáris függetlenségének.

A (4.4) egyenlőség fölírható mátrixalakban:

$$\mathbf{c} = \mathbf{u}\mathbf{G}, \tag{4.5}$$

ahol $\mathbf{u} = (u_1, u_2, \ldots, u_k)$, $\mathbf{G}$ pedig a bázisvektorokból mint sorvektorokból álló mátrix. A (4.5) egyenlettel tehát egy k-dimenziós és egy n-dimenziós vektort rendelünk össze lineáris transzformációval, mégpedig kölcsönösen egyértelmű módon. Azt fogjuk mondani, hogy az $\mathbf{u}$ üzenethez a $\mathbf{c}$ kódszó tartozik.

A k-dimenziós $\mathbf{u}$ vektorokkal 2^k-féle üzenetet fejezhetünk ki, s ezeket kódoljuk a C kóddal. C elemei azonban n-dimenziós vektorok, és n nem kisebb k-nál, hiszen k az n-dimenziós vektorok C alterének dimenziószáma. A $k = n$ esetnek nincs most jelentősége, ha k kisebb, mint n, akkor viszont világos, hogy nem minden vektort kell felhasználni kódszónak, vagyis kódunk redundáns lesz, s ezt a redundanciát tudjuk hibajavításra felhasználni.

Az üzenetekhez a kódszavakat a $\mathbf{G}$ mátrix segítségével rendeljük hozzá, vagyis a $\mathbf{G}$ mátrix jelöli ki az n-dimenziós vektortérnek a kódot jelentő C alterét, a kódot $\mathbf{G}$ „generálja”.

4.6. definíció. *A fenti tulajdonságú* $\mathbf{G}$ *mátrixot a* C *kód* ***generátormátrix****ának nevezzük.*

Vegyük észre, hogy ha nem törődünk azzal, hogy melyik kódszó melyik üzenethez tartozik, csak a kódszavak halmazát tekintjük, akkor **G** nem egyértelmű, vagyis több mátrix is generálhatja ugyanazt a C kódszóhalmazt. A következő definíció egy megfeleltetést definiál az üzenetek és a kódszavak között.

4.7. definíció. *Egy (n,k) paraméterű lineáris kód **szisztematikus**, ha minden kódszavára igaz, hogy annak utolsó $n-k$ szimbólumát elhagyva éppen a neki megfelelő k hosszúságú üzenetet kapjuk, más szavakkal a k hosszú üzenetet egészítjük ki $n-k$ karakterrel.*

A 4.1. szakaszban már leszögeztük, hogy dekódolás alatt csak az esetleges hibák kijavítását értjük, aminek eredményeképp egy kódszót kapunk. Az üzenetvektor visszanyeréséhez még el kell ugyan végezni a kódolás inverz műveletét, ez azonban rendszerint triviális lépés, szisztematikus kód esetén például csak el kell hagyni a kódszó egy részét (a végét).

Szisztematikus kód esetén a generátormátrix is egyértelmű, mégpedig

$$\mathbf{G} = (\mathbf{I}_k, \mathbf{B}) \tag{4.6}$$

alakú, ahol $\mathbf{I}_k$ a $k \times k$ méretű egységmátrix, **B** pedig $k \times (n-k)$ méretű mátrix. Az **u** üzenethez tartozó **c** kódszó szerkezete tehát:

$$\mathbf{c} = (u_1, u_2, \ldots, u_k, c_{k+1}, c_{k+2}, \ldots, c_n).$$

A **c** első k koordinátájából álló szegmensét **üzenetszegmens**nek, az utolsó $n-k$ koordinátájából állót **paritásszegmens**nek nevezzük.

A lineáris kódok további tulajdonságai elvezetnek az ígért egyszerű hibadetektáláshoz illetve hibajavításhoz.

4.8. definíció. *Ha egy $n-k$ sorból és n oszlopból álló* **H** *mátrixra*

$$\mathbf{H}\mathbf{c}^T = \mathbf{0}$$

akkor és csak akkor, ha $\mathbf{c} \in C$*, akkor* **H***-t a C kód **paritásellenőrző mátrix**ának nevezzük. (Röviden paritásmátrixot fogunk mondani.)*

H segítségével tehát meg tudjuk állapítani, hogy egy vett szó valóban kódszó-e.

4.3. tétel. *Ha* **G** *és* **H** *ugyanazon C lineáris kód generátormátrixa illetve paritásmátrixa, akkor*

$$\mathbf{H}\mathbf{G}^T = \mathbf{0}.$$

Minden lineáris kódnak van paritásmátrixa.

BIZONYÍTÁS: Jelölje Q^k a k hosszú bináris sorozatok halmazát. Ekkor minden $\mathbf{u} \in Q^k$-hoz létezik $\mathbf{c} \in C$, amire $\mathbf{c} = \mathbf{uG}$. Ugyanakkor $\mathbf{c} \in C$ miatt $\mathbf{Hc}^T = \mathbf{0}$, azaz

$$\mathbf{Hc}^T = \mathbf{H}(\mathbf{uG})^T = \mathbf{HG}^T\mathbf{u}^T = \mathbf{0}.$$

Az utolsó egyenlőség pedig csak úgy állhat fönn minden $\mathbf{u} \in Q^k$-ra, ha $\mathbf{HG}^T = \mathbf{0}$, amint állítottuk. (Megjegyezzük, hogy amennyiben $\mathbf{H}$ kielégíti a $\mathbf{HG}^T = \mathbf{0}$ egyenletet, akkor $\mathbf{H}$ még nem biztos, hogy paritásmátrix.) A tétel második felét először szisztematikus esetben látjuk be. Legyen

$$\mathbf{G} = (\mathbf{I}_k, \mathbf{B})$$

alakú, keressük $\mathbf{H}$-t

$$\mathbf{H} = (\mathbf{A}, \mathbf{I}_{n-k})$$

alakban. A tétel első fele miatt

$$\mathbf{HG}^T = (\mathbf{A}, \mathbf{I}_{n-k})(\mathbf{I}_k, \mathbf{B})^T = \mathbf{A} + \mathbf{B}^T = \mathbf{0}.$$

Azaz

$$\mathbf{A} = -\mathbf{B}^T$$

kell teljesüljön. (Bináris esetben $-\mathbf{B}^T = \mathbf{B}^T$.) Meg kell mutatni, hogy az így kapott $\mathbf{H}$ mátrix valóban paritásmátrix. Azt tudjuk, hogy minden $\mathbf{c}$ kódszóra $\mathbf{Hc}^T = \mathbf{0}$. A másik irány hiányzik: ha egy $\mathbf{c} = (c_1, c_2, \ldots, c_k, c_{k+1}, \ldots, c_n)$ vektorra $\mathbf{Hc}^T = \mathbf{0}$, akkor $\mathbf{c}$ kódszó, de ez is egyszerű:

$$\mathbf{0} = \mathbf{Hc}^T = (\mathbf{A}, \mathbf{I}_{n-k})\mathbf{c}^T = (-\mathbf{B}^T, \mathbf{I}_{n-k})\mathbf{c}^T$$

miatt

$$-(c_1, c_2, \ldots, c_k)\mathbf{B} + (c_{k+1}, \ldots, c_n) = \mathbf{0},$$

tehát

$$(c_{k+1}, \ldots, c_n) = (c_1, c_2, \ldots, c_k)\mathbf{B},$$

következésképp

$$(c_1, c_2, \ldots, c_k, c_{k+1}, \ldots, c_n) = (c_1, c_2, \ldots, c_k)(\mathbf{I}_k, \mathbf{B}),$$

ezért $\mathbf{c}$ tényleg kódszó.

Beláthatjuk, hogy egy tetszőleges (nem feltétlenül szisztematikus) kódnak van paritásmátrixa, hiszen ezt még csak szisztematikus esetben tudjuk. Az igazolás fő gondolata arra épül, hogy minden lineáris kód általános értelemben szisztematikus. Egy lineáris kódot általános értelemben szisztematikusnak nevezünk, ha

létezik egy **G** generátormátrixa és $i_1, i_2, \ldots, i_k$ egész számok úgy, hogy **G** i_j-edik oszlopa a j-edik helyen 1, a többi helyen 0. Nyilván egy ilyen tulajdonságú **G** mátrix oszlopcseréinek egy sorozatával szisztematikus mátrixot kapunk, ahol már elő tudjuk állítani a paritásmátrixot, és az előző oszlopcserék inverzével az eredeti kód egy paritásmátrixát kapjuk. Általános értelemben szisztematikus generátormátrixhoz úgy juthatunk, ha bizonyos sorok nem 0-szorosát hozzáadjuk egy másik sorhoz, úgy, ahogy azt a Gauss-eliminációnál megszoktuk. Ezek a transzformációk azért jogosak, mert a **G** mátrix sorai egy bázist alkotnak, és a transzformáció után is egy bázist alkotnak, mivel a kód lineáris. ■

4.3. példa. Adjuk meg a 4.1. példa kódjának szisztematikus generátormátrixát, ha van, és a paritásmátrixot! Ezt úgy kapjuk meg, ha **G** első sora $\mathbf{c}_3$, míg a második $\mathbf{c}_2$, mivel ekkor az első 2×2-es részmátrix egységmátrix:

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

A fentiek alapján a paritásmátrix:

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

Ugyanígy kapjuk a 4.2. példa szisztematikus generátormátrixát és paritásmátrixát:

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix},$$

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}.$$

A következőkben a súly fogalmát definiáljuk, megmutatjuk, hogy lineáris kódoknál a minimális súly a kódtávolsággal egyenlő, majd további három tételt mondunk ki velük kapcsolatban. (Emlékeztetünk, hogy két kódszó távolsága azon koordinátáik száma, ahol a két kódszó különbözik.)

4.9. definíció. *Egy* **c** *vektor* ***súly****a a koordinátái között levő nem nulla elemek száma, jelölése* $w(\mathbf{c})$.

4.10. definíció. *Egy C kód* ***minimális súly****án a*

$$w_{min} = \min_{\substack{\mathbf{c} \in C \\ \mathbf{c} \neq \mathbf{0}}} w(\mathbf{c})$$

számot értjük.

4.4. tétel. *Ha C lineáris kód, akkor a kódtávolsága megegyezik a minimális súlyával, azaz*

$$d_{min} = w_{min}.$$

BIZONYÍTÁS:

$$d_{\min} = \min_{\mathbf{c}\neq\mathbf{c}'} d(\mathbf{c},\mathbf{c}') = \min_{\mathbf{c}\neq\mathbf{c}'} w(\mathbf{c}-\mathbf{c}') = \min_{\mathbf{c}''\neq\mathbf{0}} w(\mathbf{c}'') = w_{\min},$$

ahol az utolsó előtti egyenlőség felírásakor a C kód linearitását használtuk ki, ebből következik ugyanis, hogy $\mathbf{c}'' = \mathbf{c} - \mathbf{c}'$ is kódszó, továbbá, az is, hogy minden kódszó előáll ilyen különbség alakjában. (Utóbbi ahhoz szükséges, hogy a minimum képzésekor valóban minden $\mathbf{c}'' \in C$-t figyelembe vehessünk.) ■

A 4.4. tétel jelentősége abban áll, hogy segítségével a $d_{\min}$ definíció alapján történő kiszámításához szükséges $\frac{|C|\,(|C|-1)}{2}$ műveletet a $w_{\min}$ kiszámításához szükséges $|C|-1$ műveletre redukálhatjuk. ($|C|$-vel a C elemszámát jelöltük.)

A következőkben azt mutatjuk meg, hogyan használható a $\mathbf{H}$ mátrix a dekódolás során.

4.11. definíció. *Az $\mathbf{s} = \mathbf{e}\mathbf{H}^T$ mennyiséget **szindróma**nak nevezzük.*

Legyen az adott kódszó $\mathbf{c}$, a vett szó $\mathbf{v}$. Az $\mathbf{e} = \mathbf{v} - \mathbf{c}$ vektort **hibavektor**nak nevezzük. Vegyük észre, hogy

$$\mathbf{H}\mathbf{v}^T = \mathbf{H}(\mathbf{c}+\mathbf{e})^T = \mathbf{H}\mathbf{c}^T + \mathbf{H}\mathbf{e}^T = \mathbf{H}\mathbf{e}^T,$$

vagyis $\mathbf{H}\mathbf{v}^T$ értéke csak a hibavektortól függ, az adott kódszótól nem. A szindróma tehát a hibavektor egy lineáris leképezése.

A dekódolás leggyakoribb módja a **szindróma dekódolás**. A fentiek alapján a dekódolás a következőképpen mehet végbe: a vett $\mathbf{v}$ szóból kiszámítjuk az $\mathbf{s}^T = \mathbf{H}\mathbf{v}^T = \mathbf{H}\mathbf{e}^T$ szindrómát, ennek alapján megbecsüljük a hibavektort, s ezt $\mathbf{v}$-ből levonva megkapjuk a kódszóra vonatkozó becslésünket.

A szindrómának hibamintára történő leképezési módját táblázatba szokás foglalni, az ún. **standard elrendezési táblázat**ba.

Valamely $\mathbf{e}$ hibaminta által generált mellékosztály az $\mathbf{e}+\mathbf{c},\ \mathbf{c} \in C(n,k)$ vektorok halmaza. Adott mellékosztály elemeihez azonos szindróma tartozik. Az $\mathbf{e} = \mathbf{0}$ zérus hibavektorhoz tartozó mellékosztály a $C(n,k)$ kóddal azonos. Ha egy $\mathbf{e}$ hibaminta $\mathbf{e} = \mathbf{e}' + \mathbf{c}$ alakban írható fel, akkor a két hibaminta ($\mathbf{e}$ és $\mathbf{e}'$) azonos mellékosztályt generál. Azonos mellékosztályba tartozó hibaminták közül

válasszuk ki a legkisebb súlyút, s azt mellékosztály-vezetőnek nevezzük. Ennek megfelelően a standard elrendezési táblázat az alábbi struktúrájú:

szindróma	mellékosztály-vezető			
$\mathbf{s}^{(0)}$	$\mathbf{e}^{(0)}=\mathbf{0}$	$\mathbf{c}^{(1)}$		$\mathbf{c}^{(q^k-1)}$
$\mathbf{s}^{(1)}$	$\mathbf{e}^{(1)}$	$\mathbf{c}^{(1)}+\mathbf{e}^{(1)}$		$\mathbf{c}^{(q^k-1)}+\mathbf{e}^{(1)}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\mathbf{s}^{(q^{n-k}-1)}$	$\mathbf{e}^{(q^{n-k}-1)}$	$\mathbf{c}^{(1)}+\mathbf{e}^{(q^{n-k}-1)}$		$\mathbf{c}^{(q^k-1)}+\mathbf{e}^{(q^{n-k}-1)}$

mellékosztály elemek

A $w(\mathbf{e}^{(i+1)}) \geq w(\mathbf{e}^{(i)})$, $\mathbf{e}^{(0)}=\mathbf{0}$, $i=0,1,\ldots,q^{n-k}-2$ a szokásos sorrend. Könnyen látható, hogy a táblázat elemei különbözőek. Egy soron belül ez nyilvánvaló. Különböző sorokat tekintve tegyük fel, hogy $\mathbf{e}^{(i)}+\mathbf{c}^{(j)}=\mathbf{e}^{(k)}+\mathbf{c}^{(m)}$, ahol $i>k$. Mivel ebből $\mathbf{e}^{(i)}=\mathbf{e}^{(k)}+\mathbf{c}^{(m)}-\mathbf{c}^{(j)}=\mathbf{e}^{(k)}+\mathbf{c}^{(n)}$ következik, ahol $\mathbf{c}^{(j)},\mathbf{c}^{(m)},\mathbf{c}^{(n)}\in C(n,k)$, ezért $\mathbf{e}^{(i)}$-nek is az $\mathbf{e}^{(k)}$ mellékosztály-vezetőjű sorban kell lennie, ami ellentétes kiindulási feltételünkkel.

Az $\mathbf{e}^{(i)}$, $i=1,2,\ldots,q^{n-k}-1$ mellékosztály-vezetőket **javítható hibamintá**knak nevezzük, ugyanis ha a $\mathbf{v}$ vett szó szindrómája $\mathbf{s}^{(i)}$, akkor a $\widehat{\mathbf{c}}=\mathbf{v}-\mathbf{e}^{(i)}$ kódszóra döntünk. A szindróma dekódolásnak ezt az — elsősorban elvi — módját **táblázatos dekódolás**nak nevezzük (look up table decoding).

A szindrómát használó táblázatos dekódoló tárja a vizsgált bináris esetben 2^{n-k} darab hibavektort tartalmaz, és a táblázat elemeit a szindróma segítségével címezzük.

4.4. példa. Adjuk meg a javítható hibamintákat a

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

mátrixszal adott kód esetére.

A dekódolási táblázat a következő:

szindróma	javítható hibaminták
000	00000
001	10000
010	01000
011	00110
100	00100
101	00001
110	01100
111	00010

Tehát a standard elrendezés fenti táblázata alapján történő szindróma dekódolással az egyszeres hibák és a $00110,\ 01100$ két hibát tartalmazó hibaminták javíthatók.

Illusztrációként egy klasszikusnak számító kódot mutatunk be, mely **bináris Hamming-kód** néven ismeretes.

Olyan kódot keresünk, mely egy hibát tud javítani, vagyis ha **c**-t adjuk, és **v**-t vesszük, akkor $1 \geq d(\mathbf{c}, \mathbf{v})$ esetén biztosan meg tudjuk mondani **v** ismeretében **c**-t. Legyen a kódunk lineáris és bináris.

A hibajavítás céljára r bitet kívánunk felhasználni, vagyis az n kódszóhossz és a k üzenethossz különbségét r-nek rögzítjük. Ezen adott r és 1 hibát javító képesség mellett szeretnénk a lehető legnagyobb k-t elérni, hogy minél több üzenetünk lehessen. Legyen a majdani kód paritásmátrixa:

$$\mathbf{H} = (\mathbf{a}_1^T, \mathbf{a}_2^T, \ldots, \mathbf{a}_n^T).$$

Legfeljebb egy hiba esetén az **e** hibavektor vagy **0**, vagy egységvektor, tehát pontosan 1 koordinátája 1-es. Ekkor az $\mathbf{s} = \mathbf{e}\mathbf{H}^T$ szindróma vagy **0**, vagy valamelyik $\mathbf{a}_i$-vel egyenlő. Akkor és csak akkor tudjuk tehát **e**-t (és így **c**-t is) egyértelműen megállapítani, ha az $\mathbf{a}_i$-k mind különbözők, és egyik sem **0**. Mivel **H** sorainak a száma $n-k=r$, az $\mathbf{a}_i$-k legfeljebb 2^r-1 félék lehetnek, ha egyikük sem **0**. Mivel minél nagyobb k-ra, és így a rögzített r révén minél nagyobb $k+r=n$-re törekszünk, mind a 2^r-1 lehetőséget használni fogjuk. Ezzel lényegében megadtuk **H**-t, hiszen megmondtuk, hogy **H** az összes lehetséges r hosszúságú nemnulla oszlopvektorból álló mátrix. Ez pedig már definiálja a kódszóhalmazt a

$$\mathbf{H}\mathbf{c}^T = \mathbf{0}$$

egyenletrendszer összes megoldásvektorának halmazaként. (**H** oszlopvektorainak sorrendjén csak az fog múlni, hogy melyik üzenethez melyik kódszót rendeljük,

tehát például, hogy szisztematikus lesz-e a kód. V.ö. a 4.7. definíció után mondottakkal.) Az így kapott kódot nevezzük bináris Hamming-kódnak, mely tehát k hosszú üzenethez n hosszú kódszót rendel, ahol n és k között fennáll az

$$n = 2^{n-k} - 1 \tag{4.7}$$

összefüggés. Ilyen tulajdonságú számpárok a következők:

$n =$	$k =$
3	1
7	4
15	11
31	26
63	57
127	120

A (4.7)-ből és a 4.2. tételből következik, hogy az (n,k) paraméterű bináris Hamming-kód perfekt kód, tehát

4.5. tétel. *Nincs olyan egy hibát javító bináris kód, amely egy Hamming-kóddal azonos szóhosszúságú, és a hozzá tartozó kódszavak száma nagyobb, mint a megfelelő Hamming-kód kódszavainak száma.*

4.5. példa. A $(7,4)$ paraméterű Hamming-kód paritásmátrixa

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

A generátormátrixa ebből könnyen kiszámítható a már szerepelt $\mathbf{A} = -\mathbf{B}^T$ összefüggés alapján:

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

A $(7,4)$-es Hamming-kódot egy páratlan paritásúra kiegészítő paritásbittel kapunk egy $(8,4)$ paraméterű, továbbra is egy hibát javító kódot, amelyet a Teletextben használnak [17].

A közvetlen műholdas műsorszórás (Direct Broadcasting Satellite, DBS) digitalizált hangját is hibajavító kóddal védik. Így a D2-MAC/PACKET szabványa [3] szerint az egyik változatban a 14 bites hangminta felső 11 bitjét egy $(16,11)$ paraméterű kóddal kódolják, ami a $(15,11)$ paraméterű Hamming-kód kiegészítése

egy páratlan paritásbittel. A másik változatban a 10 bites hangminta felső 6 bitjét kódolják egy $(11,6)$-os kóddal. Megjegyezzük még, hogy a csomagolt, kódolt beszédmintákat egy olyan csomagfejjel látják el, melyet 2 hibát javító $(71,57)$ ill. $(94,80)$ paraméterű BCH-kóddal védenek, míg a legfontosabb adatokat, az úgynevezett szolgáltatásazonosítást egy három hibát javító $(23,12)$ paraméterű Golay-kóddal kódolják. A BCH-kódokat a 4.9. szakaszban definiáljuk részletesebb tulajdonságaik nélkül, míg a Golay-kódok tárgyalása újabb algebrai apparátus bevezetését igényelné.

4.3. Véges test

Hatékony hibajavító kódok konstrukciójához szükséges, hogy a nembináris $Q = G$ kódábécé struktúrált legyen, mely például úgy lehetséges, hogy műveleteket vezetünk be G-n.

4.12. definíció. *Egy G halmazt **test**nek nevezünk, ha értelmezve van tetszőleges két eleme között két művelet, amelyeket összeadásnak illetve szorzásnak nevezünk, $+$ illetve $*$ szimbólumokkal jelöljük, és G rendelkezik a következő tulajdonságokkal:*

1. *G az összeadásra nézve kommutatív csoport, azaz*
   a) *Minden $\alpha,\beta \in G$ esetén $\alpha+\beta \in G$, tehát G az összeadásra nézve zárt.*
   b) *Minden $\alpha,\beta,\gamma \in G$ esetén $\alpha+(\beta+\gamma) = (\alpha+\beta)+\gamma$ (asszociativitás).*
   c) *Létezik egy 0-val jelölt eleme G-nek úgy, hogy minden $\alpha \in G$-re $0+\alpha = \alpha+0 = \alpha$. 0-t nullelemnek nevezzük.*
   d) *Minden $\alpha \in G$-hez létezik $\beta \in G$ úgy, hogy $\alpha+\beta = 0$. β-t az α additív inverzének nevezzük és $-\alpha$-val jelöljük.*
   e) *Minden $\alpha,\beta \in G$-re $\alpha+\beta = \beta+\alpha$ (kommutativitás).*
2. *$G \setminus \{0\}$ a szorzásra nézve kommutatív csoport, azaz*
   a) *Minden $\alpha,\beta \in G \setminus \{0\}$ esetén $\alpha\cdot\beta \in G \setminus \{0\}$ (zártság).*
   b) *Minden $\alpha,\beta,\gamma \in G \setminus \{0\}$ esetén $(\alpha\cdot\beta)\cdot\gamma = \alpha\cdot(\beta\cdot\gamma)$ (asszociativitás).*
   c) *Létezik egy 1-gyel jelölt eleme $G \setminus \{0\}$-nak úgy, hogy $1\cdot\alpha = \alpha\cdot 1 = \alpha$. 1-et egységelemnek nevezzük.*
   d) *Minden $\alpha \in G \setminus \{0\}$ esetén létezik $\beta \in G \setminus \{0\}$ úgy, hogy $\alpha\cdot\beta = \beta\cdot\alpha = 1$. β-t az α multiplikatív inverzének nevezzük, és α^{-1}-gyel jelöljük.*
   e) *Minden $\alpha,\beta \in G \setminus \{0\}$-ra $\alpha\cdot\beta = \beta\cdot\alpha$ (kommutativitás).*

3. *Minden* $\alpha, \beta, \gamma \in G$*-re* $\alpha \cdot 0 = 0 \cdot \alpha = 0$ *és* $\alpha \cdot (\beta + \gamma) = (\alpha \cdot \beta) + (\alpha \cdot \gamma)$ *(disztributivitás).*

Egyszerű konvenciókkal egy G testben definiálható a kivonás és az osztás a következő módon: $\alpha - \beta$ *alatt az* α*-nak és a* β *additív inverzének összegét értjük, azaz* $\alpha + (-\beta)$*-t.* α/β *alatt az* α*-nak és a* β *multiplikatív inverzének a szorzatát értjük, azaz* $\alpha \cdot \beta^{-1}$*-et, amennyiben* β *nem 0.*

Példák testre:

1. Valós számok halmaza a valós összeadással és szorzással.
2. Racionális számok halmaza a valós összeadással és szorzással.
3. Komplex számok halmaza a komplex összeadással és szorzással.
4. $\{0, 1\}$ a bináris összeadással és szorzással.

Egy q elemszámú G testet **véges test**nek nevezünk és GF(q)-val jelöljük.

Sajnos egy GF(q) esetén q nem lehet bármilyen. Ez azért fontos, mert a kódábécé a későbbiekben GF(q) lesz. Bizonyítás nélkül közöljük a következő tételt:

4.6. tétel. *Egy GF(q) esetén* $q = p^m$ *alakú, ahol p prímszám, tehát q vagy prímszám, vagy prímhatvány.*

Lényeges különbség van a prím és a prímhatvány méretű véges testek aritmetikája között. Most a prímszám méretű véges testek aritmetikáját tárgyaljuk, a másikat csak a polinomok ismeretében tudjuk egyszerűen bevezetni.

4.7. tétel. *A* $G = \{0, 1, \ldots, p-1\}$ *halmaz a* ***modulo*** *p* ***aritmetiká****val egy p prímszám esetén véges test, azaz a testműveletek*

$$a + b = a + b \mod p,$$

$$a \cdot b = a \cdot b \mod p,$$

ahol $+$ *illetve* $\cdot$ *jelöli a valós összeadást illetve szorzást.*

BIZONYÍTÁS: Ellenőrizzük az 1., 2., 3. pontokat a test definíciójában! 1. a)–e)-ig ez egyszerű, csupán azt jegyezzük meg, hogy

$$-a = p - a \mod p,$$

ahol $-$ a valós kivonást is jelöli. 2. a), b), d) és 3. megint triviális. 2. a)-nál azt kell megmutatni, hogy $a, b \neq 0$ esetén $a \cdot b \neq 0$. Ellenkező esetben $a \cdot b = Np$ alakú

lenne, tehát p osztaná $a \cdot b$-t, azaz vagy a-t vagy b-t, ami nem lehet, mert mindkettő 1 és $p-1$ közötti szám. 2. d)-hez azt kell belátni, hogy minden $a \neq 0$-hoz létezik b, hogy $a \cdot b = 1$. Az $1 \cdot a, 2 \cdot a, 3 \cdot a, \ldots, (p-1) \cdot a$ számok mind különbözők, mivel ellenkező esetben $j \cdot a = i \cdot a$ lenne, azaz $(j-i) \cdot a = 0$ lenne, ami 2. a)-nak ellentmond. A fent említett $p-1$ darab szám halmaza tehát megegyezik a $\{1, 2, \ldots, p-1\}$ halmazzal, ezért létezik b, melyre $b \cdot a = 1$. ■

Lássuk most a GF(q) néhány egyszerű tulajdonságát!

4.1. lemma. *Minden* $0 \neq a \in$ *GF(q)-ra*

$$a^{q-1} = 1.$$

BIZONYÍTÁS: Legyenek $a_1, a_2, \ldots, a_{q-1}$ a GF(q) nem 0 elemei, ekkor $a \neq 0$ esetén

$$a \cdot a_1, a \cdot a_2, \ldots, a \cdot a_{q-1}$$

is mind különbözők és nem 0-k, tehát

$$\{a_1, a_2, \ldots, a_{q-1}\} = \{a \cdot a_1, a \cdot a_2, \ldots, a \cdot a_{q-1}\},$$

következésképp

$$a_1 \cdot a_2 \cdot \ldots \cdot a_{q-1} = a^{q-1} \cdot a_1 \cdot a_2 \cdot \ldots \cdot a_{q-1},$$

ahonnan az állítás a bal oldal inverzével való szorzással következik. ■

4.2. lemma. *Minden* $0 \neq a \in$ *GF(q)-ra létezik egy legkisebb m természetes szám, amit az a* ***elem rendjé****nek nevezünk, melyre*

$$a^m = 1,$$

és az $a, a^2, \ldots, a^m$ *elemek mind különbözők. m osztója* $q-1$*-nek.*

BIZONYÍTÁS: Tekintsük az $a^1, a^2, a^3, \ldots$ sorozatot! Mivel a sorozat minden eleme GF(q)-beli, ezért GF(q) végessége miatt létezik olyan $i > j$, melyre

$$a^i = a^j,$$

azaz

$$a^{i-j} = 1,$$

tehát létezik olyan m' természetes szám, melyre $a^{m'} = 1$. Legyen m ezek közül a legkisebb. Ha az m' természetes szám olyan, hogy

$$a^{m'} = 1$$

és $m' = k \cdot m + l$ alakú, ahol $m > l \geq 0$, akkor

$$1 = a^{m'} = a^{km+l} = (a^m)^k \cdot a^l = a^l,$$

tehát l csak 0 lehet, ezért m' az m többszöröse, következésképp a 4.1. lemma miatt $q-1$ is m többszöröse. Be kell még látni, hogy $a, a^2, \ldots, a^m$ különbözők. Ennek bizonyításához tegyük fel, hogy létezik $m \geq i > j \geq 1$, hogy

$$a^i = a^j,$$

tehát

$$a^{i-j} = 1,$$

ami nem lehet, mert $m > i - j$. ■

4.13. definíció. *Egy $\alpha \in$ GF(q)-t a GF(q)* ***primitív elem****ének nevezünk, ha α rendje $q-1$.*

A 4.5. szakaszban bizonyítjuk, hogy

4.8. tétel. *Minden GF(q)-ban létezik primitív elem.*

4.6. példa. GF(7).

elem $(\neq 0)$	hatványai	rendje
1	1	1
2	2, 4, 1	3
3	3, 2, 6, 4, 5, 1	6 (primitív elem)
4	4, 2, 1	3
5	5, 4, 6, 2, 3, 1	6 (primitív elem)
6	6, 1	2

A prímitív elem egyrészt igen fontos hatékony kódok konstrukciójakor, másrészt GF(q)-beli szorzások és osztások elvégzésekor. Ha α a GF(q) egy primitív eleme, akkor bevezethetjük egy $a \in$ GF(q) testelem α alapú logaritmusát az

$$\alpha^{\log a} = a$$

egyenlet (egyértelmű) megoldásával, ahol $a \neq 0$. Ha a, b a GF(q) nem 0 elemei, akkor

$$a \cdot b = \alpha^{\log a} \cdot \alpha^{\log b} = \alpha^{\log a + \log b},$$

tehát egy α alapú logaritmustábla és egy inverzlogaritmus-tábla segítségével a szorzás (illetve az osztás) visszavezethető valós összeadásra (illetve kivonásra).

4.4. Lineáris kódok, nembináris Hamming-kód

Ebben a szakaszban kódok egy fontos csoportjával ismerkedünk meg, melyek a 4.2. szakaszban megismert bináris lineáris kódok kiterjesztései nembináris esetre.

A továbbiakban a kódjainkban szereplő kódszavakat alkotó szimbólumokat vegyük GF(q)-ból, a lehetséges szimbólumok tehát a $0,1,2,\ldots,q-1$ számoknak feleltethetők meg.

4.14. definíció. *Egy C kód **lineáris**, ha a C halmaz lineáris tér GF(q) fölött, azaz ha minden $\mathbf{c},\mathbf{c}' \in C$-re*

$$\mathbf{c}+\mathbf{c}' \in C$$

illetve $\beta \in$ GF(q) esetén

$$\beta\mathbf{c} \in C.$$

A 4.2. szakaszhoz hasonló módon belátható, hogy tetszőleges C lineáris kódhoz létezik egy k lineárisan független sorból és n oszlopból álló $\mathbf{G}$ mátrix, melyre

$$\mathbf{c} = \mathbf{u}\mathbf{G}, \tag{4.8}$$

ahol a k hosszú $\mathbf{u}$ üzenethez a $\mathbf{c}$ kódszó tartozik, és a $\mathbf{G}$ mátrixot a $\mathbf{C}$ kód **generátormátrix**ának nevezzük.

A bináris esethez hasonlóan a C lineáris kódhoz egy $n-k$ sorból és n oszlopból álló $\mathbf{H}$ mátrixot **paritásmátrix**nak nevezünk, amennyiben

$$\mathbf{H}\mathbf{c}^T = 0$$

akkor és csak akkor teljesül, ha $\mathbf{c} \in C$.

A bináris eset másolataként kaphatjuk, hogy

4.9. tétel. *Minden C lineáris kódnak van paritásellenőrző mátrixa.*

Példaként bemutatjuk a **nembináris Hamming-kód**ot. Ismét 1 hibát javító kódot akarunk konstruálni. A bináris esetben a hiba javításához elég volt ismerni a hiba helyét, amihez elégséges volt, ha a $\mathbf{H}$ paritásmátrix minden oszlopa különböző. Nembináris esetben nemcsak a hiba helyét, hanem a hiba értékét is meg kell állapítani, ezért a $\mathbf{H}$ mátrix oszlopait úgy választjuk, hogy azok nem 0-k, mind különbözők legyenek, és az első nem 0 elem minden oszlopban 1 értékű legyen. Ekkor, ha egy hiba esetén az az i-edik helyen fordul elő és értéke e_i, akkor a szindróma $\mathbf{s} = e_i\mathbf{a}_i$, ahol $\mathbf{a}_i^T$ a $\mathbf{H}$ i-edik oszlopa. Tehát a hiba értéke, e_i éppen a szindróma első nem 0 értéke, míg $\mathbf{a}_i = \frac{\mathbf{s}}{e_i}$, amiből az i visszakereshető.

Ha **H** tartalmazza az összes lehetséges, a fenti módon megengedett oszlopvektort, akkor

$$n = \frac{q^{n-k}-1}{q-1},$$

azaz

$$1+n(q-1) = q^{n-k},$$

másrészt a Hamming-korlát miatt

$$1+n(q-1) \leq q^{n-k},$$

tehát

4.10. tétel. *A maximális hosszúságú nembináris Hamming-kód perfekt kód.*

A nembináris Hamming-kódok közül különösen érdekes az az eset, amikor a kód szisztematikus és a paritásszegmens hossza 2, azaz $n-k=2$. Legyen α a GF(q) egy nem 0 eleme, melynek rendje $m \geq 2$. Válasszunk $n \leq (m+2)$-t és $k=(n-2)$-t. Ekkor a paritásmátrix:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 & 0 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-3} & 0 & 1 \end{pmatrix}.$$

Ez egy $(n, n-2)$ paraméterű nembináris Hamming-kód paritásmátrixa.

A 4.3. tétel alkalmazásával nyerjük a kód generátormátrixát:

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & -1 & -1 \\ 0 & 1 & 0 & 0 & \cdots & 0 & -1 & -\alpha \\ 0 & 0 & 1 & 0 & \cdots & 0 & -1 & -\alpha^2 \\ \vdots & & & & \ddots & & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & -1 & -\alpha^{n-3} \end{pmatrix}.$$

Mivel ez a kód 1 hibát tud javítani, ezért $d_{\min} \geq 3$, de a Singleton-korlát miatt $d_{\min} \leq n-k+1 = 3$, ezért

4.11. tétel. *Az $(n, n-2)$ paraméterű nembináris Hamming-kód MDS kód.*

4.7. példa. Írjuk fel a GF(7) feletti, $(8,6)$ paraméterű Hamming-kód generátormátrixát és paritásmátrixát! GF(7)-ben a 3 primitív elem (lásd a 4.6. példát), tehát

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 3 & 2 & 6 & 4 & 5 & 0 & 1 \end{pmatrix}$$

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & -3 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & -2 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & -6 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & -4 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & -5 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 6 & 6 \\ 0 & 1 & 0 & 0 & 0 & 0 & 6 & 4 \\ 0 & 0 & 1 & 0 & 0 & 0 & 6 & 5 \\ 0 & 0 & 0 & 1 & 0 & 0 & 6 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 6 & 3 \\ 0 & 0 & 0 & 0 & 0 & 1 & 6 & 2 \end{pmatrix}.$$

4.5. Véges test feletti polinomok

GF(q) feletti vektorok reprezentálására, és vektorok közötti szorzás kényelmes bevezetésére egy célszerű eszköz a polinomreprezentáció:

4.15. definíció. $a(x) = a_0 + a_1x + \ldots + a_mx^m$ ***GF(q) feletti*** *m-edfokú* ***polinom,*** *ha*

$$a_i \in \mathrm{GF}(q), \qquad i = 0, \ldots, m, \quad a_m \neq 0,$$
$$x \in \mathrm{GF}(q).$$

A polinom m fokszámát $\deg a(x)$ *jelöli. (Az* $a(x) \equiv 0$ *polinom fokszáma definíció szerint legyen* $-\infty$*.)*

4.16. definíció. $a(x) = b(x)$*, ha* $a_i = b_i$ *minden i-re.*

Műveletek polinomok között:

1. Polinomok összeadása: $c(x) = a(x) + b(x)$ tagonként történik GF(q) feletti műveletekkel: $c_i = a_i + b_i$. Nyilvánvalóan

$$\deg c(x) \leq \max\{\deg a(x), \deg b(x)\}.$$

2. Polinomok szorzása: $c(x) = a(x)b(x)$ minden tagot minden taggal szorzunk, majd az azonos fokú tagokat csoportosítjuk (az összeadások és szorzások GF(q) felettiek):

$$c_i = \sum_{j=0}^{\min\{i, \deg a(x)\}} a_j \cdot b_{i-j}.$$

Nyilván

$$\deg c(x) = \deg a(x) + \deg b(x)$$

4.8. példa. Ha GF(2) felett $a(x) = 1 + x$ és $b(x) = 1 + x + x^3$, akkor $a(x) + b(x) = x^3$ és $a(x)b(x) = 1 + x^2 + x^3 + x^4$.

4.12. tétel (Euklidészi osztás polinomokra). *Adott $a(x)$ és $d(x) \neq 0$ esetén egyértelműen létezik olyan $q(x)$, $r(x)$ úgy, hogy*

$$a(x) = q(x)d(x) + r(x),$$

és $\deg r(x) < \deg d(x)$.

4.17. definíció. *$r(x)$-et az $a(x)$-nek $d(x)$-re vonatkozó maradékának nevezzük. Jelölés:* $r(x) = a(x) \mod d(x)$.

4.18. definíció. *$d(x)$ osztja $a(x)$-et, ha $a(x) \mod d(x) = 0$. Ezt a továbbiakban $d(x) \mid a(x)$ formában fogjuk jelölni.*

A 4.12. TÉTEL BIZONYÍTÁSA: Az egész számok euklidészi osztási algoritmusának analógiájára megadunk egy rekurzív algoritmust. Legyen $a(x)$ m-edfokú és x^m együtthatója a_m, $d(x)$ k-adfokú és x^k együtthatója d_k. Ha $m \geq k$, akkor legyen

$$r_1(x) = a(x) - a_m d_k^{-1} x^{m-k} d(x)$$

és

$$q_1(x) = a_m d_k^{-1} x^{m-k},$$

akkor

$$a(x) = q_1(x)d(x) + r_1(x)$$

és

$$\deg(r_1(x)) < \deg(a(x)).$$

Ha most $a(x)$-nek van egy

$$a(x) = q_i(x)d(x) + r_i(x)$$

előállítása, ahol $\deg(r_i(x)) \leq \deg(d(x))$, akkor az előző módon $r_i(x)$ előáll

$$r_i(x) = q_{i+1}(x)d(x) + r_{i+1}(x)$$

alakban, ahol $\deg(r_{i+1}(x)) < \deg(r_i(x))$, tehát

$$a(x) = (q_{i+1}(x) + q_i(x))d(x) + r_{i+1}(x),$$

és $\deg(r_i(x))$ monoton fogy, ezért legfeljebb $m-k$ lépésben a tétel által megkívánt felbontást kapjuk. Az egyértelműség igazolásához tegyük fel, hogy van két különböző felbontás:

$$a(x) = q(x)d(x) + r(x)$$

$$a(x) = q'(x)d(x) + r'(x),$$

és $q(x) \neq q'(x)$, tehát

$$(q'(x) - q(x))d(x) = r(x) - r'(x).$$

Mivel $q(x) \neq q'(x)$, ezért

$$\begin{aligned} \deg(r(x) - r'(x)) &= \deg((q'(x) - q(x))d(x)) = \\ &= \deg(q'(x) - q(x)) + \deg(d(x)) \geq \\ &\geq \deg(d(x)), \end{aligned}$$

másrészt

$$\deg(r(x) - r'(x)) < \deg(d(x)),$$

ami ellentmondás. ■

4.19. definíció. *$b \in GF(q)$ gyöke az $a(x)$ polinomnak, ha $a(b) = 0$.*

4.13. tétel. *Ha c az $a(x)$ polinom gyöke, akkor az előáll*

$$a(x) = b(x)(x - c)$$

alakban.

BIZONYÍTÁS: Alkalmazzuk a 4.12. tételt $d(x) = x - c$ esetén, akkor

$$a(x) = b(x)(x - c) + e.$$

Mivel c gyök, ezért

$$0 = a(c) = b(c)(c - c) + e = e.$$ ■

4.14. tétel. *Egy k-adfokú polinomnak legfeljebb k gyöke lehet.*

BIZONYÍTÁS: A 4.13. tétel miatt a $b(x)$ polinom fokszáma eggyel kisebb, mint az $a(x)$ polinom fokszáma, tehát ezt a faktorizációt legfeljebb k-szor lehet megismételni. ■

Ezek után be tudjuk már bizonyítani a 4.8. tételt, mely azt állította, hogy minden testben van primitív elem. Először lássunk egy segédtételt!

4.3. lemma. *Ha $\alpha, \beta \in GF(q)$ elemek rendje n illetve m relatív prímek, akkor az $\alpha\beta$ elem rendje mn.*

BIZONYÍTÁS: m és n legnagyobb közös osztóját jelölje (m,n), akkor a feltétel miatt $(m,n)=1$. Legyen k az $\alpha\beta$ rendje. $(\alpha\beta)^{mn}=(\alpha^n)^m(\beta^m)^n=1$, tehát k osztja mn-t, tehát

$$k \le mn.$$

Ugyanakkor $(\alpha\beta)^k=1$, ezért $\alpha^k=\beta^{-k}$, így $\alpha^{mk}=\beta^{-mk}=1$, tehát a 4.2. lemma bizonyítása alapján α rendje n osztja mk-t, ezért $(m,n)=1$ miatt n osztja k-t. Hasonló módon $\beta^{nk}=\alpha^{-nk}=1$, ezért β rendje m osztja nk-t, ezért $(m,n)=1$ miatt m osztja k-t. Ismét $(m,n)=1$ miatt tehát mn is osztja k-t, ezért

$$mn \le k.$$

■

A 4.8. TÉTEL BIZONYÍTÁSA: Azt kell belátni, hogy minden GF(q)-nak van primitív eleme. A bizonyítás konstruktív. Ha $q-1$ prím, akkor az osztói az 1 és a $q-1$, tehát a lehetséges rendek 1 vagy $q-1$, mégpedig az 1 elem rendje 1, a többinek pedig $q-1$, tehát a GF(q) minden 0-tól és 1-től különböző eleme primitív elem. Ha $q-1$ nem prím, akkor a prímtényezős felbontása legyen

$$q-1=\prod_{i=1}^{s} p_i^{\nu_i}.$$

Mivel a GF(q) nem 0 elemeinek száma $q-1$ és az $x^{\frac{q-1}{p_i}}-1$ polinomnak legfeljebb $\frac{q-1}{p_i}$ gyöke van, és $\frac{q-1}{p_i}<q-1$, ezért létezik $a_i\neq 0$ úgy, hogy $a_i^{\frac{q-1}{p_i}}\neq 1$. Legyen

$$b_i=a_i^{\frac{q-1}{p_i^{\nu_i}}} \qquad \text{és} \qquad b=\prod_{i=1}^{s} b_i.$$

Megmutatjuk, hogy b primitív elem. Ehhez először azt látjuk be, hogy b_i rendje $p_i^{\nu_i}$. Nyilván $b_i^{p_i^{\nu_i}}=a_i^{q-1}=1$, tehát b_i rendje a $p_i^{\nu_i}$ osztója, azaz $p_i^{n_i}$ alakú, ahol $n_i\le\nu_i$. Ha $n_i<\nu_i$ lenne, akkor

$$b_i^{p_i^{n_i}}=b_i^{p_i^{\nu_i-(\nu_i-n_i)}}=1,$$

ezért a $p_i^{\nu_i-(\nu_i-n_i)}$ minden r_i többszörösére $b_i^{r_i}=1$, tehát

$$b_i^{p_i^{\nu_i-1}}=1$$

lenne, ami nem lehet, mert

$$b_i^{p_i^{\nu_i-1}}=a_i^{\frac{q-1}{p_i}}\neq 1,$$

következésképp b_i rendje $p_i^{\nu_i}$. Alkalmazzuk a 4.3. lemmát $(s-1)$-szer, mivel $b_1, b_2, \ldots, b_s$ olyan elemek, melyek rendjei relatív prímek, akkor b rendje a rendek szorzata, azaz

$$\prod_{i=1}^{s} p_i^{\nu_i} = q-1,$$

tehát b primitív elem. ■

4.6. Reed–Solomon-kód

Ebben a szakaszban a lineáris kódok egyik leggyakrabban használt osztályával, a **Reed–Solomon-kód**okkal, azok különböző konstrukcióival ismerkedünk meg.

4.1. konstrukció. *Legyenek $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ a GF(q) különböző elemei ($n \leq q$), és $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1})$ ($u_i \in$ GF(q)) a k hosszúságú üzenetszegmens, amelyhez az*

$$u(x) = u_0 + u_1 x + \ldots + u_{k-1} x^{k-1}$$

üzenetpolinomot rendeljük. Ekkor a Reed–Solomon-kódnak az **u** *üzenethez tartozó n hosszú* **c** *kódszavát a következő módon állítjuk elő:*

$$\begin{aligned} c_0 &= u(\alpha_0) \\ c_1 &= u(\alpha_1) \\ c_2 &= u(\alpha_2) \\ &\vdots \\ c_{n-1} &= u(\alpha_{n-1}). \end{aligned}$$

Egyszerűen belátható, hogy a Reed–Solomon-kód lineáris, és a generátormátrixa

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ \alpha_0 & \alpha_1 & \alpha_2 & \cdots & \alpha_{n-1} \\ \vdots & & & \ddots & \vdots \\ \alpha_0^{k-1} & \alpha_1^{k-1} & \alpha_2^{k-1} & \cdots & \alpha_{n-1}^{k-1} \end{pmatrix}.$$

4.15. tétel. *Az (n,k) paraméterű Reed–Solomon-kód kódtávolsága*

$$d_{min} = n-k+1,$$

vagyis a Reed–Solomon-kód maximális távolságú.

BIZONYÍTÁS:

$$\begin{aligned} w(\mathbf{c}) &= |\{\mathbf{c} \text{ nem 0 koordinátái}\}| = \\ &= n - |\{\mathbf{c} \text{ 0 koordinátái}\}| \geq \\ &\geq n - |\{u(x) \text{ gyökei}\}| \geq \\ &\geq n - (k-1), \end{aligned}$$

tehát

$$w_{\min} \geq n - k + 1.$$

Ugyanakkor a 4.1. tétel és a 4.4. tétel miatt

$$n - k + 1 \geq d_{\min} = w_{\min},$$

következésképp az állítást bebizonyítottuk. ■

Az (n,k) paraméterű Reed–Solomon-kód tehát $n-k$ hibát tud jelezni, $\left\lfloor \frac{n-k}{2} \right\rfloor$ egyszerű hibát javítani és $n-k$ törléses hibát javítani. Ez utóbbi azt is jelenti, hogy az **u** ismeretlenre vonatkozó

$$\mathbf{uG} = \mathbf{c}$$

n darab egyenletből bármelyik $n-k$ egyenlet elhagyásával egy egyértelműen megoldható egyenletrendszer marad, tehát a **G** mátrix minden $k \times k$-s négyzetes részmátrixa invertálható. Ez utóbbi állítás igazolható úgy is, hogy felismerjük, miszerint **G** minden $k \times k$-s négyzetes részmátrixa Vandermonde típusú.

4.2. konstrukció. *Legyen α a GF(q) egy nem 0 eleme, melynek rendje m, $m \geq n$ és a 4.1. konstrukcióban legyen $\alpha_0 = 1, \alpha_1 = \alpha, \ldots, \alpha_{n-1} = \alpha^{n-1}$. Ekkor a generátormátrix:*

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & & & \ddots & \vdots \\ 1 & \alpha^{k-1} & \alpha^{2(k-1)} & \cdots & \alpha^{(k-1)(n-1)} \end{pmatrix}$$

4.3. konstrukció. *A $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ vektorhoz rendeljük hozzá $c(x)$ polinomot a szokásos módon:*

$$c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}.$$

Ha az α elem rendje m, és $n \leq m$, akkor a kód definíciója

$$C = \{\mathbf{c} : \ c(\alpha^i) = 0, \ i = 1, 2, \ldots, n-k\}.$$

Egyszerűen belátható, hogy a C kódnak ez a megadása ekvivalens a következővel:

$$C = \{\mathbf{c} : \ \mathbf{H}\mathbf{c}^T = 0\},$$

ahol

$$\mathbf{H} = \begin{pmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & & & \ddots & \vdots \\ 1 & \alpha^{n-k} & \alpha^{2(n-k)} & \cdots & \alpha^{(n-k)(n-1)} \end{pmatrix}.$$

Bebizonyítjuk, hogy ez a kód maximális távolságú, és $n = m$ esetén ez a kód azonos a 4.2. konstrukcióban leírt Reed–Solomon-kóddal. (Az irodalomban $n < m$ esetén a 4.2. konstrukció kódját rövidített Reed–Solomon-kódnak nevezik.) Először azt mutatjuk meg, hogy a C kód maximális távolságú. Ennek érdekében előbb a paritásmátrixok egy hasznos tulajdonságát bizonyítjuk:

4.4. lemma. *Ha* $\mathbf{H}$ *egy lineáris C kód paritásmátrixa, akkor* $\mathbf{H}$ *azon oszlopainak minimális száma, melyek lineárisan függők, d_{min}.*

BIZONYÍTÁS: Írjuk fel $\mathbf{H}$ oszlopait!

$$\mathbf{H} = (\mathbf{a}_0^T, \mathbf{a}_1^T, \ldots, \mathbf{a}_{n-1}^T),$$

akkor a paritásegyenlet $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ jelöléssel:

$$c_0\mathbf{a}_0 + c_1\mathbf{a}_1 + \ldots + c_{n-1}\mathbf{a}^{n-1} = \mathbf{0}.$$

Ezt az egyenletet csak olyan nemnulla $\mathbf{c}$ vektorok (kódszavak) elégíthetik ki, melyek súlya legalább $w_{\min}$, ilyen súlyú viszont van, tehát a lineárisan függő oszlopok minimális száma $w_{\min}$, amely a 4.4. tétel miatt éppen $d_{\min}$.

■

Erre alapozva már szinte készen vagyunk. Ha a 4.2. konstrukcióban a $\mathbf{G}$ képletében k helyett $(n-k)$-t írunk, akkor annak minden $(n-k) \times (n-k)$-s részmátrixa invertálható. Ha egy ilyen részmátrixot összehasonlítunk a $\mathbf{H}$ ugyanezen oszlopokból álló részmátrixával, akkor vegyük észre, hogy az egymásnak megfelelő oszlopok egy nem 0 elem faktorban különböznek, tehát a $\mathbf{H}$ minden

$(n-k)\times(n-k)$-s négyzetes részmátrixa invertálható, ezért minden $n-k$ oszlopa lineárisan független. A 4.4. lemma miatt tehát

$$n-k<d_{\min},$$

azaz

$$n-k+1\le d_{\min},$$

ezért ismét a Singleton-korlátot alkalmazva készen vagyunk.

Eddig azt mutattuk meg, hogy $\mathbf{H}$ egy (n,k) paraméterű MDS kód paritásmátrixa. Mivel mind a 4.2. mind a 4.3. konstrukció (n,k) paraméterű, ezért a kettő azonosságához elég megmutatni, hogy a 4.3. tartalmazza a 4.2.-t. A 4.2. konstrukció egy kódszavának i-edik koordinátája

$$c_i=\sum_{j=0}^{k-1}u_j\alpha^{ij},\qquad 0\le i\le n-1.$$

Megmutatjuk, hogy az így megadott kódszó kielégíti a 4.3. konstrukció paritásegyenletét, azaz

$$\sum_{i=0}^{n-1}c_i\alpha^{il}=0,\qquad 1\le l\le n-k.$$

A $\mathbf{c}$ definícióját felhasználva

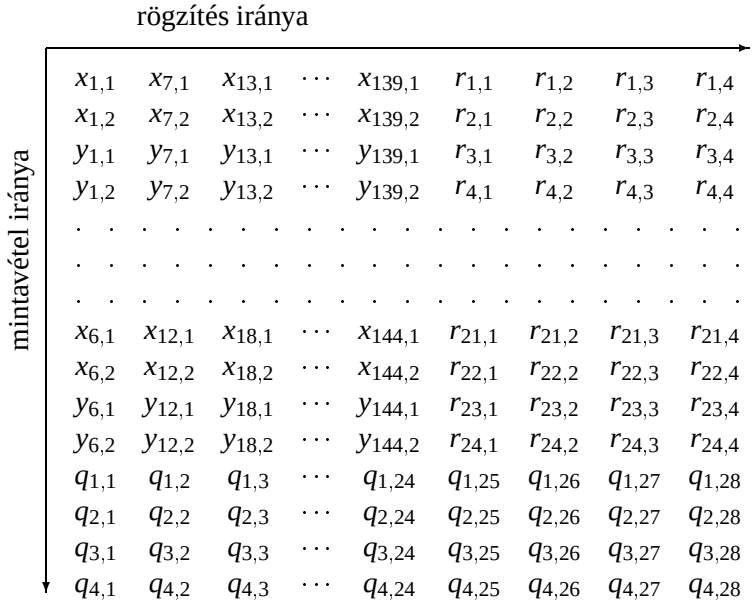
$$\sum_{i=0}^{n-1}c_i\alpha^{il}=\sum_{i=0}^{n-1}\sum_{j=0}^{k-1}u_j\alpha^{ij}\alpha^{il}=\sum_{j=0}^{k-1}u_j\sum_{i=0}^{n-1}\alpha^{i(j+l)}.$$

Mivel $0\le j\le k-1$ és $1\le l\le n-k$, ezért $1\le j+l\le n-1$, tehát $\alpha^{j+l}\neq 1$ és

$$\sum_{i=0}^{n-1}\alpha^{i(j+l)}=\frac{\alpha^{n(j+l)}-1}{\alpha^{j+l}-1}=\frac{1^{j+l}-1}{\alpha^{j+l}-1}=0,$$

következésképp $\mathbf{c}$ kielégíti a paritásegyenletet. ■

Példaként a digitális hangrögzítésben (CD és DAT) alkalmazott Reed–Solomon-kódot említjük [2, 35, 16, 24, 7]. A kódolási eljárás lényegét közelítőleg a következő módon lehet összefoglalni: a 44.1 kHz-cel mintavételezett és 16 bitbe kvantált mintákat két bájtban ábrázoljuk, és egy mátrixba írjuk be oszlopfolytonosan.



$$
\begin{array}{cccccccccc}
x_{1,1} & x_{7,1} & x_{13,1} & \cdots & x_{139,1} & r_{1,1} & r_{1,2} & r_{1,3} & r_{1,4} \\
x_{1,2} & x_{7,2} & x_{13,2} & \cdots & x_{139,2} & r_{2,1} & r_{2,2} & r_{2,3} & r_{2,4} \\
y_{1,1} & y_{7,1} & y_{13,1} & \cdots & y_{139,1} & r_{3,1} & r_{3,2} & r_{3,3} & r_{3,4} \\
y_{1,2} & y_{7,2} & y_{13,2} & \cdots & y_{139,2} & r_{4,1} & r_{4,2} & r_{4,3} & r_{4,4} \\
\vdots & & & & & & & & \vdots \\
x_{6,1} & x_{12,1} & x_{18,1} & \cdots & x_{144,1} & r_{21,1} & r_{21,2} & r_{21,3} & r_{21,4} \\
x_{6,2} & x_{12,2} & x_{18,2} & \cdots & x_{144,2} & r_{22,1} & r_{22,2} & r_{22,3} & r_{22,4} \\
y_{6,1} & y_{12,1} & y_{18,1} & \cdots & y_{144,1} & r_{23,1} & r_{23,2} & r_{23,3} & r_{23,4} \\
y_{6,2} & y_{12,2} & y_{18,2} & \cdots & y_{144,2} & r_{24,1} & r_{24,2} & r_{24,3} & r_{24,4} \\
q_{1,1} & q_{1,2} & q_{1,3} & \cdots & q_{1,24} & q_{1,25} & q_{1,26} & q_{1,27} & q_{1,28} \\
q_{2,1} & q_{2,2} & q_{2,3} & \cdots & q_{2,24} & q_{2,25} & q_{2,26} & q_{2,27} & q_{2,28} \\
q_{3,1} & q_{3,2} & q_{3,3} & \cdots & q_{3,24} & q_{3,25} & q_{3,26} & q_{3,27} & q_{3,28} \\
q_{4,1} & q_{4,2} & q_{4,3} & \cdots & q_{4,24} & q_{4,25} & q_{4,26} & q_{4,27} & q_{4,28}
\end{array}
$$

Nevezetesen egy 24×24-es mátrix oszlopai egymásután következő 6 mintavételi időpontban vett két minta (bal és jobb hangcsatorna) $2 \times 2 = 4$ bájtját tartalmazzák. Ha $x_{i,1}, x_{i,2}$ jelöli a jobb csatorna mintáját az i-edik időpillanatban, és $y_{i,1}, y_{i,2}$ a bal csatornáét, akkor a fenti ábra mutatja a minták beírását a táblázatba. A kapott 24×24-es mátrix minden oszlopát kódoljuk egy $(28, 24)$ paraméterű, GF(2^8) feletti szisztematikus Reed–Solomon kóddal. A j-edik oszlop paritásbájtjait jelöltük $q_{1,j}, q_{2,j}, q_{3,j}, q_{4,j}$-vel. Ennek a kódnak a kódtávolsága 5, tehát 4 hibát tud jelezni, 2 egyszerű hibát tud javítani és 4 törléses hibát tud javítani. A digitális lemezen előforduló hibák jól modellezhetők egy kétállapotú csatornával. Az egyik állapotot nevezzük JÓ állapotnak, melyben átlagosan 10000–20000 bitideig tartózkodik, és ekkor a hibák előfordulása független egymástól és valószínűsége kb. 10^{-4}. A másik állapotot nevezzük ROSSZ állapotnak, amiben 30–40 bitideig tartózkodik, és ekkor gyakorlatilag használhatatlan a vétel. Ekkor azt mondjuk, hogy a hibázás csomós (burst-ös). Az ilyen csatornák kódolására találták ki a kódátfűzés (interleaving) technikát, amikor az előbbi mátrixot sorfolytonosan olvassák ki, de előtte minden sort kódolnak ugyanazzal a $(28, 24)$ paraméterű Reed–Solomon-kóddal. A j-edik sor paritásbájtjait jelöli $r_{j,1}, r_{j,2}, r_{j,3}, r_{j,4}$. Ennek előnye az, hogy a fizikailag összefüggő, csomós hiba hatását több kódszóra osztja szét.

A Sony és a Philips megegyezett a fentihez hasonló (kicsit bonyolultabb) kódolásban azért, hogy a tömeges digitális hanglemezgyártás elindulhasson. A verseny nyitott viszont a lejátszó készülékben, vagyis a dekódolás terén. A különböző dekódolások igaziból a következő egyszerű eljárás finomításai: számítsuk ki

soronként a szindrómát! Ha a szindróma 0, akkor azzal a sorral készen vagyunk. Ha 1 hiba volt, akkor azt kijavítjuk. Ha 2 hiba volt, akkor azt kijavítjuk, és az oszloponkénti javításhoz ezeket a hibahelyeket megjegyezzük, azaz mesterségesen törléses hibákat generálunk. Minden egyéb esetben az egész sort törléses hibaként regisztráljuk. Ezek után oszloponként javítunk, ha ott legfeljebb két törléses hiba volt (emlékeztetünk, hogy 4 törléses hibát képes a rendszer javítani). Ha a hibák száma nagyobb, mint 2, akkor a környező hibátlan mintákból interpolálunk. Látható, hogy a hibajavítás nem használja ki a Reed–Solomon-kód hibajavítási lehetőségeit, aminek elsősorban technológiai okai vannak, mivel a dekódolás bonyolultsága a javítandó hibák számának négyzetével arányos, és itt igen gyorsan kell dekódolni (a forrás sebessége $2 \cdot 44100 \cdot 16 = 1.4112$ Mbit/sec)

4.7. Aritmetika GF(p^m)-ben

A 4.3. szakaszban említettük, hogy lényeges különbség van a prím illetve prímhatvány méretű testek aritmetikája között. Prím méretű testben a modulo aritmetika megfelelt. Prímhatvány méret esetén sajnos a modulo aritmetika nem teljesíti a testaxiómákat, például egy 4 elemű halmazban $2 \cdot 2 \mod 4 = 0$, tehát két nem 0 elem szorzata 0 lenne, ami sérti a 2. a) axiómát. A GF(p^m) feletti aritmetika konstrukciója azért alapvető fontosságú, mert manapság a hibajavító kódokat tömegesen alkalmazzuk számítástechnikai környezetben, ahol a természetes ábécé a GF(2^8), vagyis a bájt. Az előző szakasz végén szereplő digitális hangtechnikai példában is GF(2^8) volt az ábécé.

A GF(p^m)-beli elemek legyenek a $0, 1, \ldots, p^m - 1$ számok, melyeknek m hosszú vektorokat feleltetünk meg, ahol a koordináták GF(p)-beliek. Ezt megfogalmazhatjuk például úgy is, hogy a $0, 1, \ldots, p^m - 1$ számokat p-s számrendszerben írjuk fel. Ezek után a GF(p^m)-beli aritmetikát m hosszú vektorok közötti műveletekkel definiáljuk. A két művelet közül az összeadás az egyszerűbb: két vektor összegén a koordinátánkénti GF(p)-beli összeget értjük, vagyis a koordinátánkénti mod p összeget. A szorzás egy kicsit bonyolultabb. A két m hosszú vektort legfeljebb $(m-1)$-edfokú polinom formájában reprezentáljuk, és összeszorozzuk. Az eredmény fokszáma meghaladhatja $(m-1)$-et, ezért itt egy speciális polinom szerinti maradékot képezünk. Ezt a speciális polinomot irreducíbilis polinomnak nevezzük, és ez a polinom ugyanolyan szerepet játszik, mint a prímszám a GF(p)-beli aritmetikában.

4.20. definíció. *A GF(p) feletti, nem nulladfokú $P(x)$ polinomot* ***irreducíbilis polinom****nak nevezzük, ha nem bontható fel két, nála alacsonyabb fokú GF(p) feletti polinom szorzatára, azaz nincs GF(p) feletti $a_1(x), a_2(x)$ polinom, melyekre*

$$P(x) = a_1(x) \cdot a_2(x)$$

és

$$0 < \deg(a_i(x)) < \deg(P(x)), \qquad i = 1,2.$$

Bizonyítás nélkül megjegyezzük, minden véges testben található tetszőleges fokszámú irreducíbilis polinom. Példát mutatunk viszont arra, hogy hogyan lehet GF(2) feletti irreducíbilis polinomokat generálni. A definícióból következik, hogy minden elsőfokú polinom (x és $x+1$) irreducíbilis. Ha találunk olyan másodfokú polinomot, mely különbözik az x^2, az $x(x+1)$ és az $(x+1)^2$ mindegyikétől, akkor találtunk irreducíbilis másodfokú polinomot. Egy ilyen van: x^2+x+1. Más testben és nagyobb fokszám esetén ennél hatékonyabb konstrukciókat érdemes használni, de bináris esetben így is találhatók irreducíbilis polinomok, amelyeket táblázatban foglalunk össze:

fokszám	irreducíbilis polinom
2	x^2+x+1
3	x^3+x+1
4	x^4+x+1
5	x^5+x^2+1
6	x^6+x+1
7	x^7+x^3+1
8	$x^8+x^4+x^3+x^2+1$
9	x^9+x^4+1

4.16. tétel. *Legyen p egy prím, m egy természetes szám, $P(x)$ egy GF(p) feletti m-edfokú irreducíbilis polinom és $Q = \{0,1,\ldots,p^m-1\}$. Egy $a \in Q$-nak és $b \in Q$-nak kölcsönösen egyértelműen feleltessünk meg GF(p) feletti, legfeljebb $(m-1)$-edfokú polinomot. $a+b$ definíció szerint az a $c \in Q$, melynek megfelelő $c(x)$ polinomra*

$$c(x) = a(x) + b(x).$$

$a \cdot b$ az a $d \in Q$, melynek megfelelő $d(x)$ polinomra

$$d(x) = \{a(x) \cdot b(x)\} \mod P(x).$$

Ezzel az aritmetikával Q egy GF(p^m).

4.9. példa. Készítsük el a GF(2^2)-beli aritmetikát! Tudjuk, hogy a $P(x) = x^2 + x + 1$ egy másodfokú irreducíbilis polinom. A kölcsönösen egyértelmű megfeleltetéseket egy táblázatba foglaljuk össze:

testelemek	$m = 2$ hosszú vektorok	polinomok
0	00	0
1	01	1
2	10	x
3	11	$x+1$

Az összeadást egyszerűen a 2 hosszú vektorok koordinátánkénti bináris összegével kapjuk. Nézzünk a szorzásra példát! $2 \cdot 3$-at úgy számoljuk ki, hogy a 2-nek és a 3-nak megfelelő polinomot összeszorozzuk, és vesszük a $P(x)$ szerinti maradékot:

$$x(x+1) = 1 \pmod{x^2+x+1},$$

amely megfelel az 1 testelemnek. Az összeadó és a szorzó tábla ennek megfelelően a bináris vektorokra:

$+$	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

$\cdot$	00	01	10	11
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

majd testelemekre

$+$	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

$\cdot$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

A 4.3. szakaszban már említettük, hogy primitív elem alapú logaritmustábla és inverzlogaritmus-tábla segítségével egyszerűsíthető a szorzás. A fenti szorzótábla segítségével ellenőrizhető, hogy mind a 2, mind a 3 primitív elem. Válasszuk a 2-t! Ekkor

logaritmustábla	
testelem	log
1	3
2	1
3	2

inverzlogaritmus-tábla	
egészek	invlog
1	2
2	3
3	1

Ezt felhasználva $2\cdot 3 = 2^{\log(2\cdot 3)} = 2^{\log 2+\log 3} = 2^{1+2} = 2^3 = 1$ (itt $+$ jel a $\bmod q-1$ összeadást jelöli).

Hátra van még a 4.16. tétel bizonyítása. Ehhez először egy segédtételt bizonyítunk:

4.5. lemma. *Ha $P(x)$ egy GF(p) feletti irreducíbilis polinom, és osztja $a(x)b(x)$-et, akkor legalább az egyik tényezőt osztja.*

BIZONYÍTÁS: Egy $p(x)$ polinomot **főpolinom**nak nevezünk, ha a legmagasabb fokú tagjának az együtthatója 1. Az $a(x)$ és $b(x)$ polinomok legkisebb közös többszöröse legyen az az $[a(x),b(x)]$ minimális, de pozitív fokszámú főpolinom, amelynek $a(x)$ és $b(x)$ is osztója. Egyszerűen belátható, hogy a legkisebb közös többszörös minden közös többszörösnek osztója. A lemma bizonyításához tegyük fel, hogy $P(x)$ az $a(x)$-nek nem osztója, akkor egyrészt $a(x)\not\equiv 0$, másrészt a $P(x)a(x)$ egy közös többszörös, mely felírható

$$P(x)a(x) = q(x)[P(x),a(x)] = q(x)d(x)a(x)$$

alakban, ahol $\deg(d(x)) > 0$, azaz

$$P(x) = q(x)d(x),$$

tehát $q(x)$ a $P(x)$ osztója. Mivel $P(x)$ irreducíbilis, és $\deg(d(x)) > 0$, ezért $\deg(q(x)) = 0$, azaz $q(x) = q_0$ alakú. Mivel $a(x)b(x)$ az $a(x)$-nek és $P(x)$-nek is többszöröse, ezért felírható

$$a(x)b(x) = h(x)[P(x),a(x)] = h(x)q_0^{-1}P(x)a(x)$$

alakban, azaz $b(x) = h(x)q_0^{-1}P(x)$, tehát $P(x)$ osztja $b(x)$-et. ■

A 4.16. TÉTEL BIZONYÍTÁSA: A testaxiómákat egy kivételével egyszerű igazolni, ha észrevesszük, hogy a nullelem az azonosan 0 polinom, és az egységelem az azonosan 1 polinom. Egyedül az igényel részletesebb meggondolást, hogy minden nemnulla $a(x)$ polinomnak van multiplikatív inverze, azaz létezik $b(x)$ polinom, melyre

$$a(x)b(x) = 1 \mod P(x).$$

Tekintsük a következő két halmazt:

$$A = \{f(x) : \deg(f(x)) \le m-1\},$$

$$B = \{a(x)f(x) \mod P(x) : \deg(f(x)) \le m-1\}.$$

Nyilván B az A részhalmaza, hiszen A egy $f(x)$ eleméhez a B egy elemét rendeltük az $a(x)f(x) \mod P(x)$ leképezéssel. A fenti $b(x)$ létezését belátjuk, ha megmutatjuk, hogy $A = B$, azaz az $a(x)f(x) \mod P(x)$ leképezésünk invertálható. Tegyük fel, hogy nem az, azaz létezik két különböző $f_1(x) \in A$ és $f_2(x) \in A$ úgy, hogy

$$a(x)f_1(x) \mod P(x) = a(x)f_2(x) \mod P(x)$$

tehát $P(x)$ osztja $a(x)f_1(x) - a(x)f_2(x) = a(x)(f_1(x) - f_2(x))$-et. A 4.5. lemma miatt $P(x)$ vagy $a(x)$-et vagy $(f_1(x) - f_2(x))$-et osztja. $P(x)$ nem oszthatja $a(x)$-et, mert az utóbbi nem 0, és a fokszáma kisebb, mint a $P(x)$ fokszáma. Mivel $f_1(x) - f_2(x)$ fokszáma is kisebb, mint $P(x)$-é, ezért $P(x)$ csak akkor osztja $(f_1(x) - f_2(x))$-et, ha $f_1(x) - f_2(x) = 0$, ami ellentmond annak a feltételünknek, hogy $f_1(x)$ és $f_2(x)$ különböznek. ■

4.8. Ciklikus kódok

4.21. definíció. *Egy*

$$\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$$

vektor ciklikus eltoltja a

$$S\mathbf{c} = (c_{n-1}, c_0, \ldots, c_{n-2}).$$

*S-et a **ciklikus eltolás** operátorának nevezzük.*

4.22. definíció. *A C kódot **ciklikus**nak nevezzük, ha bármely kódszó ciklikus eltoltja is kódszó.*

4.10. példa. Legyen C a

000
101
110
011
111

vektorok halmaza. Egyszerűen belátható, hogy C ciklikus. Megjegyezzük, hogy a ciklikusságból nem következik a linearitás, például a 2. és az 5. kódszó összege 010, ami nem eleme a kódnak.

A ciklikus eltolás operátorát kényelmesebben tudjuk kezelni, ha a kódszavakat mint vektorokat a már megszokott polinomos formában reprezentáljuk.

4.23. definíció. *Rendeljünk polinomot az egyes kódszavakhoz a következő módon:*

$$\mathbf{c} = (c_0, c_1, \ldots, c_{n-1}) \;\mapsto\; c(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1},$$

ekkor a $\mathbf{c}$ *kódszónak megfeleltetett* $c(x)$ *polinomot* ***kódszópolinom****nak vagy röviden kódpolinomnak nevezzük. A kódszópolinomok halmazát* $C(x)$*-szel jelöljük.*

4.6. lemma. *Legyen* $c'(x)$ *a* $\mathbf{c}$ *kódszó* $S\mathbf{c}$ *eltoltjához rendelt kódszópolinom, ekkor*

$$c'(x) = [xc(x)] \mod (x^n - 1).$$

BIZONYÍTÁS: Legyen $c(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1}$, ekkor

$$xc(x) = c_{n-1}(x^n - 1) + c'(x),$$

ahonnan az állítás következik, mert $\deg(c'(x)) \leq n-1$. ■

4.17. tétel. *Minden* (n,k) *paraméterű, ciklikus, lineáris* C *kódban a nem azonosan nulla kódszópolinomok között egyértelműen létezik egy minimális fokszámú* $g(x)$ *főpolinom.* $g(x)$ *fokszáma* $n-k$*, és egy* $\mathbf{c} \in C$ *akkor és csak akkor, ha* $g(x) \mid c(x)$*, azaz létezik egy* $u(x)$ *polinom úgy, hogy* $c(x) = g(x)u(x)$*.*

BIZONYÍTÁS: A nem nulladfokú kódszópolinomok között biztosan létezik legkisebb fokú. Legyen egy ilyen

$$q(x) = c_0 + c_1 x + \cdots + c_r x^r.$$

Mivel a kód lineáris, ezért a

$$g(x) = c_r^{-1} q(x)$$

is kódpolinom és minimális fokszámú főpolinom. Be kell látnunk az egyértelműséget. Tegyük fel, hogy van kettő: $g'(x)$ és $g''(x)$, akkor a linearitás miatt a különbségük is kódszópolinom, és r-nél kisebb fokszámú, ami lehetetlen. A ciklikusság miatt $g(x), xg(x), x^2g(x), \ldots, x^{n-r-1}g(x)$ is kódszópolinom, ezért ismét a linearitás miatt tetszőleges $(u_0, u_1, \ldots, u_{n-r-1})$-re

$$(u_0 + u_1 x + \cdots + u_{n-r-1}x^{n-r-1})g(x)$$

is kódszópolinom, azaz

$$u(x) = u_0 + u_1 x, \ldots, u_{n-r-1}x^{n-r-1}$$

jelölés esetén $u(x)g(x)$ kódszópolinom. A másik irányú állítás az, hogy ha $c(x)$ egy tetszőleges kódszópolinom, akkor $g(x)$ osztja $c(x)$ polinomot. Az euklidészi

osztási tétel miatt ugyanis $c(x) = q(x)g(x) + r(x)$, ahol $\deg(r(x)) < \deg(g(x))$. Ez csak úgy lehet, ha $r(x) = 0$, mivel $r(x) = c(x) - q(x)g(x)$ is kódszópolinom és $g(x)$-nél kisebb fokszámú. Így $r(x)$ csak 0 lehet. Ebből viszont következik, hogy minden kódszópolinom előáll $(u_0 + u_1 x, \ldots, u_{n-r-1}x^{n-r-1})g(x)$ alakban. Másrészt egy (n,k) paraméterű kódban q^k darab kódszó van, ahonnan következik, hogy

$$r = n - k.$$

■

4.24. definíció. *$g(x)$-et a kód **generátorpolinom**jának nevezzük.*

4.18. tétel. *Minden ciklikus, lineáris kód $g(x)$ generátorpolinomjára*

$$g(x) \mid x^n - 1.$$

Másrészről, ha egy $g(x)$ főpolinomra $g(x) \mid x^n - 1$, akkor létezik egy lineáris ciklikus kód, melynek $g(x)$ a generátorpolinomja.

BIZONYÍTÁS: Miután $x^{k-1}g(x)$ $(n-1)$-edfokú főpolinom, annak eltoltja $x^k g(x) - (x^n - 1)$. Mindkettő kódszópolinom, tehát ez utóbbi is osztható $g(x)$-szel, másrészt $x^k g(x)$ is osztható $g(x)$-szel, ezért a kettő különbsége is osztható $g(x)$-szel, ami viszont éppen $x^n - 1$. Az állítás második felét is egyszerű belátni. Legyen $g(x)$ fokszáma $n-k$, akkor a kódszópolinomok

$$C(x) = \{c(x) = a(x)g(x);\ \deg(a(x)) \leq k-1\}$$

halmaza egy (n,k) paraméterű lineáris kódot definiál. Meg kell mutatni, hogy ez ciklikus, azaz $c(x) = a(x)g(x)$, $\deg(a(x)) \leq k-1$ esetén eltoltja is kódszópolinom, azaz létezik $a'(x)$, $\deg(a'(x)) \leq k-1$, úgy hogy $xc(x) = a'(x)g(x) \mod (x^n - 1)$. Ha $\deg(a(x)) < k-1$, akkor legyen $a'(x) = xa(x)$. Ha $\deg(a(x)) = k-1$ és legmagasabb fokú tagjának együtthatója a_{k-1}, akkor $xc(x)$ előáll

$$xc(x) = xa(x)g(x) = a_{k-1}(x^n - 1) + b(x)$$

alakban, ahol $\deg(b(x)) \leq n-1$. Ha $g(x)$ osztja $x^n - 1$-et, akkor osztja $b(x)$-et is, tehát az $a'(x)g(x)$ alakú, ezért

$$xc(x) = a'(x)g(x) \mod x^n - 1.$$

■

A ciklikus kódok előnyös tulajdonságai egyrészt a generálási lehetőségek sokféleségében, másrészt egyszerű dekódolási eljárásokban jelentkeznek. Egy lineáris ciklikus kódot lehet például a generátorpolinom és az üzenetpolinom szorzásával generálni. Ez a módszer megfogalmazható egy olyan **G** generátormátrix

segítségével is, amelyhez legegyszerűbben úgy juthatunk el, ha **G** sorai a $g(x)$-nek megfelelő vektor eltoltjai:

$$\mathbf{G} = \begin{pmatrix} g_0 & g_1 & g_2 & \cdots & g_{n-k-1} & 1 & 0 & \cdots & & 0 \\ 0 & g_0 & g_1 & \cdots & g_{n-k-2} & g_{n-k-1} & 1 & \cdots & & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & g_0 & g_1 & g_2 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & g_0 & g_1 & \cdots & g_{n-k-1} & 1 \end{pmatrix},$$

kihasználva, hogy $g(x)$ főpolinom, így $g_{n-k} = 1$.

Egy másik generálási módszer kapcsán azt is megmutatjuk, hogy

4.19. tétel. *Minden lineáris ciklikus kód generálható szisztematikusan.*

BIZONYÍTÁS: Vegyük észre, hogy $g_0 \neq 0$, mert ha 0 lenne, akkor a **g**-t balra eltolva **g**-nél 1-gyel kisebb fokszámú kódszópolinomot kapnánk, ami lehetetlen. $g_0 \neq 0$ miatt viszont a Gauss-eliminációt balról jobbra végrehajtva szisztematikus generátormátrixot kapunk. ■

A **szisztematikus generálás** egy praktikus módszere a következő:
Legyen $u(x)$ egy legfeljebb $(k-1)$-edfokú üzenetpolinom és

$$c(x) = u(x)x^{n-k} - [u(x)x^{n-k}] \mod g(x),$$

akkor $c(x)$ kódszópolinom, mivel $c(x) = 0 \mod g(x)$. Ezen generálás szisztematikus: a $c(x)$-et definiáló egyenlőség jobb oldalának első tagja adja az üzenetszegmenst, míg a második tagja a paritásszegmenst.

4.11. példa. Tekintsük a GF(2) feletti $g(x) = 1 + x + x^3$ polinomot! Mivel

$$x^7 - 1 = (1+x)(1+x+x^3)(1+x^2+x^3),$$

ezért $g(x)$ osztja (x^7-1)-et, tehát $g(x)$ egy $(7,4)$ paraméterű bináris, lineáris, ciklikus kód generátorpolinomja. Egy generátormátrixához jutunk a $g(x)$ eltoltjaival:

$$\mathbf{G} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

A második módszer szerinti szisztematikus generátormátrixhoz úgy jutunk el, ha kiszámítjuk az $[x^{3+i}] \mod (1+x+x^3)$ maradékokat $i = 0,1,2,3$-ra, melyek

$$\begin{aligned}
x^3 &= 1+x \mod (1+x+x^3)\\
x^4 &= x(1+x+x^3)-x(1+x) =\\
&= x(1+x) =\\
&= x+x^2 \mod (1+x+x^3)\\
x^5 &= x^2(1+x+x^3)-x^2(1+x) =\\
&= x^2(1+x) =\\
&= 1+x+x^2 \mod (1+x+x^3)\\
x^6 &= x^3(1+x+x^3)-x^3(1+x) =\\
&= x^3+x^4 =\\
&= 1+x+x+x^2 =\\
&= 1+x^2 \mod (1+x+x^3)
\end{aligned}$$

Ezek alapján

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1\\ 0 & 1 & 0 & 0 & 1 & 1 & 1\\ 0 & 0 & 1 & 0 & 1 & 1 & 0\\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix},$$

amely a már jól ismert $(7,4)$ paraméterű Hamming-kód szisztematikus generátormátrixa.

A paritásmátrixnak is van polinomos megfelelője:

4.25. definíció. *Egy $g(x)$ generátorpolinomú lineáris, ciklikus kód esetén a*

$$h(x) = \frac{x^n-1}{g(x)}$$

*polinomot **paritásellenőrző polinom**nak nevezzük.*

4.20. tétel. *Egy lineáris, ciklikus kódra $c(x)$ akkor és csak akkor kódszópolinom, ha*

$$c(x)h(x) = 0 \mod (x^n-1)$$

és

$$\deg(c(x)) \leq n-1.$$

BIZONYÍTÁS: Ha $c(x)$ kódpolinom, akkor $c(x) = u(x)g(x)$ alakú, tehát

$$c(x)h(x) = u(x)g(x)h(x) = u(x)(x^n - 1),$$

tehát $c(x)h(x) = 0 \mod (x^n - 1)$. Ha viszont $c(x)h(x) = 0 \mod (x^n - 1)$, akkor $c(x)h(x) = a(x)(x^n - 1)$ alakú, ahonnan

$$c(x) = a(x)(x^n - 1)/h(x) = a(x)g(x),$$

tehát $c(x)$ kódpolinom. ■

4.21. tétel. *A Reed-Solomon-kódok 4.3. konstrukciójában legyen az n kódszóhossz egyenlő az ott szereplő* α *elem m rendjével. Ekkor a kód ciklikus, és generátorpolinomja*

$$g(x) = \prod_{i=1}^{n-k} (x - \alpha^i),$$

továbbá paritásellenőrző polinomja

$$h(x) = \prod_{i=n-k+1}^{n} (x - \alpha^i),$$

tehát a nem rövidített Reed–Solomon-kód ciklikus.

BIZONYÍTÁS: A 4.3. konstrukciót a paritásmátrixával adtuk meg, és a kódszavak kielégítették a paritásegyenleteket:

$$\sum_{j=0}^{n-1} c_j \alpha^{ij} = 0, \qquad i = 1, \ldots, n-k,$$

tehát $\alpha^1, \ldots, \alpha^{n-k}$ gyöke a $c(x)$ polinomnak, vagyis $c(x)$ felírható

$$c(x) = \prod_{i=1}^{n-k} (x - \alpha^i) u(x)$$

alakban. A $g(x)$ tételbeli definíciójával az tényleg egy ciklikus kód generátorpolinomja, ha megmutatjuk, hogy osztja az $(x^n - 1)$-et. Ez utóbbi azért igaz, mert $g(x)$ gyökei az $(x^n - 1)$-nek is gyökei, ugyanis

$$(\alpha^i)^n = (\alpha^i)^m = \alpha^{im} = (\alpha^m)^i = 1,$$

ezért

$$x^n - 1 = \prod_{i=1}^{n} (x - \alpha^i),$$

ahonnan a $h(x)$-re vonatkozó állítás is következik. ■

A ciklikus kódok gyakorlata a leghosszabb múlttal a hibajelzés területén rendelkezik, amikor a kód bináris, a kódokat a szabványok generátorpolinomjuk segítségével adják meg és generálásuk a 4.19. tétel bizonyításában leírt módon, a generátorpolinom szerinti maradékos osztással, szisztematikusan történik. Ezeket a kódokat **CRC** kódoknak hívják (Cyclic Redundancy Check). A hibajelzést legtöbbször zajos, visszacsatolásos csatornáknál használják, amikor a vevő hiba detektálása esetén értesíti az adót, amely ezután az adást ugyanazzal a kóddal vagy egy jobbal megismétli. Ezt az eljárást **ARQ**-nak nevezzük (Automatic Repeat reQuest).

A CCITT 16 paritásbitet tartalmazó szabványában a CRC generátorpolinomja

$$g_1(x) = x^{16} + x^{12} + x^5 + 1.$$

Ezt a generátorpolinomot alkalmazzák pl. az SNC 2653 (Polinomial Generator Checker), INTEL 82586 (Local Communication Controller), INTEL 8274 (Multi-Protocol Serial Controller), Signetics 2652 (Multi-Protocol Communications Circuit) integrált áramkörökben. A két utóbbiban még választhatjuk a

$$g_2(x) = x^{16} + x^{15} + x^2 + 1$$

polinomot is. Az INTEL 82586-os Ethernet chip tartalmaz egy 32 bites generátorpolinomot is:

$$g_3(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1.$$

A 4.18. tétel értelmében ezek a polinomok akkor generátorpolinomjai ciklikus kódoknak, ha osztják az $x^n - 1$ polinomot, tehát csak bizonyos kódszóhosszakra ciklikus kódok. Ugyanakkor erre nem figyelmeztetik a felhasználót. Ez azért nem okoz problémát, mert tetszőleges üzenethossz esetén azért jó kódot kapunk, ugyanis az vagy eleve ciklikus, vagy egy ciklikus kód rövidítése. Legyen C egy (n,k) paraméterű szisztematikus lineáris kód és $k' < k$. Egy

$$\mathbf{u}' = (u'_0, u'_1, \ldots, u'_{k'})$$

üzenethez rendeljük a k hosszú

$$\mathbf{u} = (0, 0, \ldots, 0, u'_0, u'_1, \ldots, u'_{k'})$$

üzenetet, ahhoz a

$$\mathbf{c} = (0, 0, \ldots, 0, u'_0, u'_1, \ldots, u'_{k'}, c_{k+1}, c_{k+2}, \ldots, c_n)$$

kódszót és ahhoz a rövidített kódszót:

$$\mathbf{c}' = (u'_0, u'_1, \ldots, u'_{k'}, c_{k+1}, c_{k+2}, \ldots, c_n).$$

Az ilyen $\mathbf{c}'$ kódszavak C' halmazát nevezzük a C kód rövidített kódjának. Nyilván

$$n - n' = k - k'$$

és C' kódtávolsága legalább akkora, mint a C kódé.

Ez utóbbi miatt elég összefoglalni a CRC kódok alapvető tulajdonságait akkor, amikor az ciklikus kód, azaz az n kódszóhosszra a generátorpolinom osztja $x^n - 1$-et.

Ezek után legyen n az a legkisebb természetes szám, melyre $g_1(x) \mid x^n - 1$, és jelölje C az $(n, n-16)$ paraméterű, ciklikus, lineáris kódot, melynek a generátorpolinomja $g_1(x)$, ekkor

1. tulajdonság: $n = 2^{15} - 1 = 32767$.
2. tulajdonság: C jelez minden legfeljebb 3 súlyú hibát.
3. tulajdonság: C jelez minden páratlan súlyú hibát.
4. tulajdonság: C jelez minden olyan hibát, ahol a hibahelyek maximumának és minimumának a távolsága kisebb, mint 16. (Ez utóbbit úgy szokás mondani, hogy a kód jelez minden legfeljebb 16 hosszú hibacsomót.)

BIZONYÍTÁS: Az 1. bizonyítása egyrészt történhet további matematikai apparátus bevezetésével, másrészt számítógépes ellenőrzéssel.

Ha a **v** vett szónak, az **e** hibamintának és a küldött **c** kódszónak rendre a $v(x)$, $e(x)$ és $c(x)$ polinom felel meg, akkor

$$v(x) = c(x) + e(x)$$

miatt C egy olyan **e** hibát tud jelezni, melyre $g_1(x)$ nem osztja $e(x)$-et. Mivel $g_1(1) = 0$, ezért

$$g_1(x) = (x-1)g'(x)$$

alakú. Ha **e**-ben páratlan sok 1 van, akkor $e(1) = 1$, tehát $x-1$ nem osztja $e(x)$-et, ezért $g_1(x)$ sem osztja $e(x)$-et, és ezzel a 3. tulajdonságot beláttuk.

Mivel $g_1(x)$ kódszópolinom és a súlya 4, ezért a C kódtávolsága legfeljebb 4, tehát C legfeljebb 3 hibát jelezhet. A 3. tulajdonság miatt az 1 és a 3 súlyút jelzi, tehát elég megvizsgálni a 2 súlyú hibákat. Ha **e** 2 súlyú, akkor

$$e(x) = x^j + x^i$$

alakú, ahol $0 \leq i < j \leq n-1$. $g_1(x) \mid e(x)$, ha

$$x^i(x^{j-i}+1) = g_1(x)z(x)$$

alakú. Írjuk fel $z(x)$-et

$$z(x) = x^m z'(x)$$

alakban, ahol x nem osztja $z'(x)$-et, azaz

$$x^i(x^{j-i}+1) = x^m g_1(x)z'(x).$$

Ha $i > m$ lenne, akkor

$$x^{i-m}(x^{j-i}+1) = g_1(x)z'(x),$$

tehát $x \mid g_1(x)z'(x)$, ami lehetetlen, mert x nem osztja $z'(x)$-et és $g_1(0) = 1$ miatt x nem osztja $g_1(x)$-et. Ha $i \leq m$ lenne, akkor

$$x^{j-i}+1 = x^{j-i}-1 = g_1(x)z'(x)x^{m-i},$$

ami szintén lehetetlen, mert n volt az a legkisebb egész, melyre $g_1(x) \mid x^n - 1$, márpedig $j-i < n$, ezért $g_1(x)$ nem oszthatja $(x^{j-i}-1)$-et. Ezzel a 2. tulajdonságot is beláttuk.

A 4. tulajdonsághoz legyen

$$e(x) = x^i e'(x),$$

ahol $i \geq 0$ és $\deg(e'(x)) < 16$. Ha $g_1(x) \mid e(x)$, akkor

$$x^i e'(x) = g_1(x)z'(x)x^m$$

alakú, ahol x nem osztja $z'(x)$-et. $i \leq m$ nem lehet, mert ekkor

$$e'(x) = g_1(x)z'(x)x^{m-i},$$

tehát $g_1(x)$ a nála kisebb fokszámú $e'(x)$-et osztaná. $i > m$ sem lehet, mert ekkor

$$x^{i-m}e'(x) = g_1(x)z'(x),$$

azaz x osztaná $g_1(x)z'(x)$-et, ami szintén lehetetlen a 2. tulajdonság bizonyításában elmondottak miatt.

Az 1., 2., 3. és 4. tulajdonság mindegyike áll a $g_2(x)$ polinom esetén is. A 2., 3. és 4. tulajdonságot be lehet látni minden $g(x) = x^{16} + x^j + x^i + 1$ alakú generátorpolinomra, ahol $16 > j > i > 0$. Nyilván fontos, hogy az az n, melyre $g(x)$ egy

n hosszú ciklikus kód generátorpolinomja, elegendően nagy legyen, tehát a gyakorlatban választott üzenethossz a kód egy rövidítésének üzenethossza legyen. Belátható, hogy az ilyen $g(x)$-ek közül a $g_1(x)$ és a $g_2(x)$ esetén a legnagyobb az n. ■

A 4.2. szakasz végén említettük, hogy a műholdas műsorszórás szolgáltatásazonosítását egy $(23,12)$ paraméterű Golay-kóddal védik. Ennek a kódnak a kódtávolsága 7, ezért 3 hibát tud javítani. Könnyen ellenőrizhető, hogy a kód paramétereire a Hamming-korlátban az $=$ teljesül:

$$\sum_{i=0}^{3}\binom{23}{i}=2^{11},$$

tehát ez a kód perfekt. Eredetileg Golay a kódot szisztematikus generátormátrixával adta meg:

$$\mathbf{G}=\begin{pmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&1&1&0&0&0&1&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&1&1&0&0&0&1\\
0&0&1&0&0&0&0&0&0&0&0&0&1&0&1&1&0&1&1&1&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&1&1&0&1&1&1&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&1&1&0&1&1&1&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&1&1&0&1&1&1\\
0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&1&0&1&1&0&1&1\\
0&0&0&0&0&0&0&1&0&0&0&0&1&1&0&0&0&1&0&1&1&0&1\\
0&0&0&0&0&0&0&0&1&0&0&0&1&1&1&0&0&0&1&0&1&1&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&1&1&1&0&0&0&1&0&1&1\\
0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&1&1&0&0&0&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1
\end{pmatrix}$$

Kiderült, hogy ez a kód ciklikus is, és a generátorpolinomja

$$g(x)=x^{11}+x^{10}+x^6+x^5+x^4+x^2+1.$$

Ugyanilyen paraméterű kódot kapunk a

$$g'(x)=x^{11}+x^9+x^7+x^6+x^5+x+1$$

generátorpolinommal. Ezek valóban 23 hosszú ciklikus kódok generátorpolinomjai, ugyanis

$$(x-1)g(x)g'(x)=x^{23}-1.$$

Mivel a $g(x)$ együtthatói közül 7 darab 1, ezért a minimális súly nem lehet 7-nél nagyobb. Megmutatható, hogy pontosan 7:

4.22. tétel. *A* $(23,12)$ *paraméterű Golay-kód egy 3 hibát javító perfekt, lineáris, ciklikus kód.*

Ciklikus kódok shiftregiszteres generálásai

Mind a ciklikus kódok generálására, mind a paritásellenőrzésre hatékony eszköz a lineárisan előrecsatolt illetve a lineárisan visszacsatolt **shiftregiszter**.

Mint láttuk, egy (n,k) paraméterű ciklikus kód nemszisztematikusan generálható $g(x) = g_0 + g_1x + \cdots + g_{n-k}x^{n-k}$ generátorpolinomja felhasználásával, amelynél egy $u(x) = u_0 + u_1x + \cdots + u_{k-1}x^{k-1}$ üzenethez a

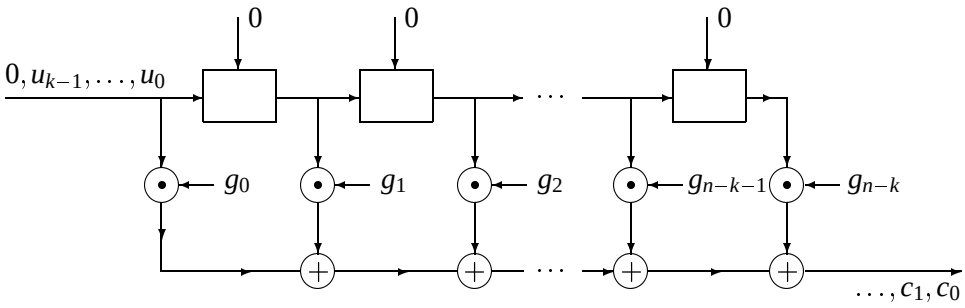
$$c(x) = u(x)g(x) \tag{4.9}$$

polinomszorzással generáljuk a $c(x) = c_0 + c_1x + \cdots + c_{n-1}x^{n-1}$ kódszót. A (4.9) alakú polinomszorzás a 4.2. ábra szerinti lineárisan előrecsatolt shiftregiszteres elrendezéssel realizálható.

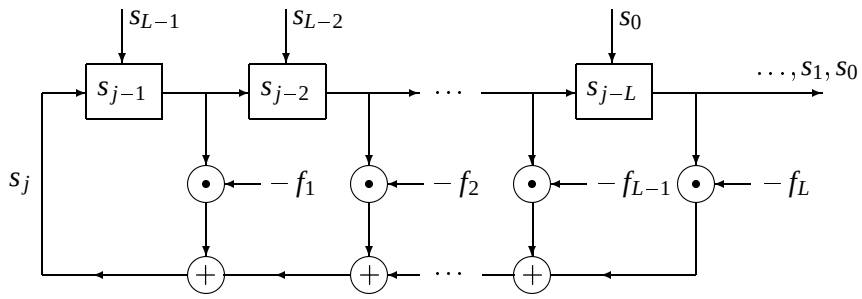
Könnyen látható, hogy ha a zéró kezdő értékű regiszterbe n lépésben beléptetjük a zéróelemekkel n hosszúra kiegészített $u_0, u_1, \ldots, u_{k-1}, 0, \ldots, 0$ sorozatot, akkor a 4.2. ábra szerinti generátor kimenetén megjelenő karakterek lépésenként a következők:

$$\begin{array}{rrcl} 0. & c_0 & = & u_0g_0 \\ 1. & c_1 & = & u_0g_1 + u_1g_0 \\ & & \vdots & \\ n-1. & c_{n-1} & = & u_{k-1}g_{n-k}, \end{array} \tag{4.10}$$

azaz valóban a (4.9) polinomszorzást hajtjuk végre.



4.2. ábra. Nemszisztematikus generálás előrecsatolással.



4.3. ábra. LFSR.

A szisztematikus generálás realizálásának hatékony alapeleme a 4.3. ábra szerinti lineárisan visszacsatolt shiftregiszter (LFSR, Linear Feedback Shift Register).

A shiftregiszter L hosszú, és a kezdő érték $s_0, s_1, \ldots, s_{L-1}$. A további, $s_j,\ j \geq L$ elemek a

$$s_j = -\sum_{i=1}^{L} f_i s_{j-i}, \qquad j \geq L \tag{4.11}$$

rekurzív alakban állnak elő. A (4.11) összefüggést gyakran célszerűbb az

$$\sum_{i=0}^{L} f_i s_{j-i} = 0, \qquad j \geq L \tag{4.12}$$

alakba átírni, ahol $f_0 = 1$. Az LFSR-hez rendelt

$$f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_L x^L \tag{4.13}$$

polinomot az **LFSR karakterisztikus polinomjá**nak nevezzük. Az $\langle f(x), L \rangle$ jelölést használjuk az $f(x)$ karakterisztikus polinomú és L regiszterhosszú LFSR jelölésére.

Egy (n,k) paraméterű ciklikus kód egyértelműen megadható $h(x) = h_0 + h_1 x + \cdots + h_k x^k$ paritásellenőrző polinomjával. Vezessük be a $h(x)$ polinom

$$\widehat{h}(x) = h_k + h_{k-1} x + \cdots + h_0 x^k$$

reciprok polinomját. Megmutatjuk, hogy ciklikus kódot szisztematikusan generálhatunk $\langle \widehat{h}(x), k \rangle$ LFSR segítségével, ha a kezdő értéket a kódolandó üzenetnek választjuk. A $c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$ szisztematikus kódszóban a

$c_{n-1} = u_0,\ c_{n-2} = u_1,\ \ldots,\ c_{n-k} = u_{k-1}$ pozíciókba helyezzük el az üzenetkaraktereket.

A 4.20. tétel szerint egy lineáris, ciklikus kódra $c(x)$ akkor és csak akkor kódszópolinom, ha

$$c(x)h(x) = 0 \mod (x^n - 1), \tag{4.14}$$

amelynek alapján a $c(x)h(x)$ polinom $n-1, n-2, \ldots, k$ fokszámú tagjainak együtthatójára az alábbi összefüggéseket írhatjuk fel:

$$\begin{aligned} h_0c_{n-1} + h_1c_{n-2} + \cdots + h_kc_{n-k-1} &= 0 \\ h_0c_{n-2} + h_1c_{n-3} + \cdots + h_kc_{n-k-2} &= 0 \\ &\vdots \\ h_0c_k + h_1c_{k-1} + \cdots + h_kc_0 &= 0 \end{aligned} \tag{4.15}$$

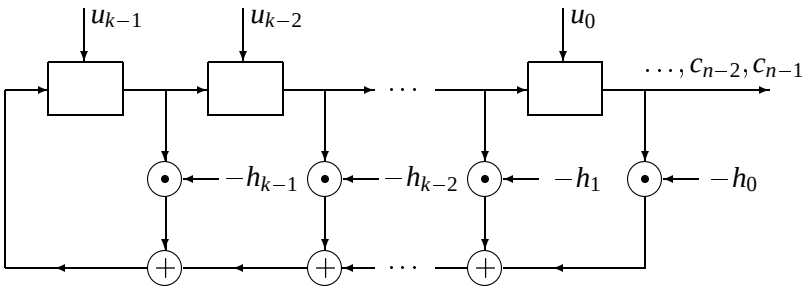
Mivel a $g(x)$ generátorpolinom főpolinom, ezért a $h(x)g(x) = x^n - 1$ összefüggésből következik, hogy $h(x)$ is főpolinom, azaz $h_k = 1$. Ennek figyelembevételével a (4.15) egyenletek a következő rekurzív alakba írhatók:

$$c_{n-k-j} = -\sum_{i=0}^{k-1} h_i c_{n-i-j}, \tag{4.16}$$

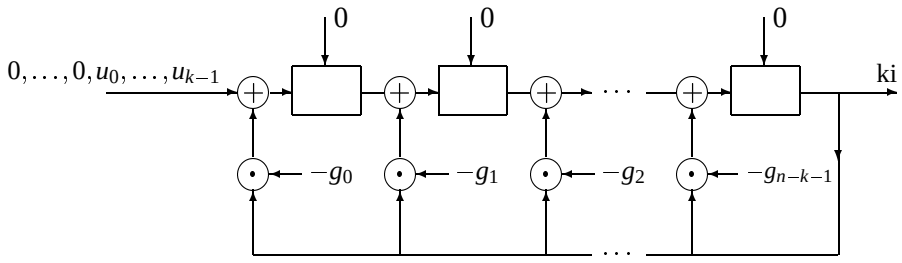
$j = 1, 2, \ldots, n-k$, azaz $c_{n-1} = u_0,\ c_{n-2} = u_1,\ \ldots, c_{n-k} = u_{k-1}$ ismeretében generáljuk a $c_{n-k-1}, c_{n-k-2}, \ldots, c_0$ paritásszegmenst.

A 4.4. ábrán látható a paritásellenőrző polinomra épülő LFSR kódgenerálás, melynek megfeleltetése:

$$f(x) = \widehat{h}(x); \qquad L = k; \qquad s_0 = u_0,\ s_1 = u_2,\ \ldots,\ s_{k-1} = u_{k-1}.$$



4.4. ábra. Kódgenerálás paritásellenőrző polinommal.



4.5. ábra. Osztás $g(x)$ polinommal.

A paritásellenőrző polinommal történő kódgenerálásnál az LFSR regiszterhossza k karakter. Így ezt az eljárást nem célszerű alkalmazni azon tipikus esetekben, amikor nagyobb az üzenethossz a paritásszegmens $n-k$ méretéhez képest. Ekkor célszerűbb a generátorpolinom alapú shiftregiszteres generálás. Tudjuk, hogy egy (n,k) paraméterű $g(x)$ generátorpolinomú ciklikus kód $u(x)$ üzenethez tartozó $c(x)$ kódszava szisztematikusan generálható az alábbi összefüggéssel (lásd a 4.19. tétel bizonyítása után írtakat):

$$c(x) = u(x)x^{n-k} - [u(x)x^{n-k}] \mod g(x). \tag{4.17}$$

A (4.17) összefüggésben egy $u(x)x^{n-k}$ polinomnak a $g(x)$ polinomra vett maradékát kell képezni. Figyelembe véve, hogy $g(x)$-et főpolinomnak választottuk, ezt a polinomosztást végzi el az alábbi **visszacsatolt shiftregiszteres osztó** elrendezés.

Az osztó elrendezés által végzett shiftelés és az éppen kilépő elemmel az osztópolinom nemzérus együtthatóinak megfelelő pozíciókban történő súlyozott levonás nyilván a jól ismert polinomosztás egy lépésének a végrehajtását jelenti. Tehát a 4.5. ábrán látható elrendezés euklidészi osztást végez. Zéró kezdő állapotból indítva, és az osztó bemenetére időben egymás után a $d_{n-1}, d_{n-2}, \ldots, d_0$ elemeket léptetve a

$$d(x) = q(x) \cdot g(x) + r(x) \tag{4.18}$$

euklidészi osztásnak megfelelően a kimenetén a $q(x)$ hányados, a regiszterében pedig az $r(x)$ maradék áll elő. Így a bemenetén az $n-k$ 0-ával kiegészített k üzenetkaraktert beléptetve, n lépés után a regiszter az $[u(x)x^{n-k}] \mod g(x)$ maradékot tartalmazza.

Nyilván az osztó elrendezés használható CRC generálására illetve ellenőrzésére is, mivel ekkor is egy (n,k) paraméterű szisztematikus ciklikus kód kódszavait állítjuk elő. CRC generáláskor a fentiekben elmondott módon paritásszeg-

menst állítunk elő, míg ellenőrzéskor hibátlan szó esetén zéró osztási maradékot kapunk az osztó regiszterében.

4.9. BCH-kód

4.26. definíció. *Az n kódszóhosszú, $n = q^m - 1$, $GF(q)$ feletti kódot t hibát javító* ***BCH-kód****nak nevezzük, ha a $g(x)$ generátorpolinomjának gyökei az $\alpha^i \in \mathrm{GF}(q^m)$, $i = 1, 2, \ldots, 2t$ testelemek. (BCH = Bose–Chaudhuri–Hocquenghem)*

4.23. tétel. *Ha az n kódszóhosszú, $GF(q)$ feletti C ciklikus kód generátorpolinomjának az $\alpha \in \mathrm{GF}(q^m)$ elem $d-1$ egymás utáni (különböző) hatványa gyöke, azaz valamely $i_0 \geq 0, d > 1$ esetén*

$$g(\alpha^{i_0}) = g(\alpha^{i_0+1}) = \cdots = g(\alpha^{i_0+d-2}) = 0,$$

akkor a kód minimális távolsága legalább d.

BIZONYÍTÁS: Amennyiben $c(x)$ C-beli kódszó, akkor $c(\alpha^{i_0}) = c(\alpha^{i_0+1}) = \cdots = c(\alpha^{i_0+d-2}) = 0$. A paritásellenőrző mátrixot a következő tömör formában írhatjuk fel:

$$\mathbf{H} = \begin{pmatrix} 1 & \alpha^{i_0} & \alpha^{2i_0} & \cdots & \alpha^{(n-1)i_0} \\ 1 & \alpha^{i_0+1} & \alpha^{2(i_0+1)} & \cdots & \alpha^{(n-1)(i_0+1)} \\ \vdots & & & \ddots & \vdots \\ 1 & \alpha^{i_0+d-2} & \alpha^{2(i_0+d-2)} & \cdots & \alpha^{(n-1)(i_0+d-2)} \end{pmatrix}$$

hiszen például $c(\alpha^{i_0}) = c_0 + c_1\alpha^{i_0} + c_2\alpha^{2i_0} + \cdots + c_{n-1}\alpha^{(n-1)i_0}$. Az állítás bizonyításához azt kell megmutatni, hogy a $\mathbf{H}$ mátrix bármely $d-1$ vagy kevesebb oszlopa $\mathrm{GF}(q^m)$ feletti vektorok lineárisan független halmaza, ekkor ugyanis legalább d súlyú az a $\mathbf{c}$ vektor, amelyre $\mathbf{H}\mathbf{c}^T = 0$, és így a kód teljesíti az előírt minimális távolságot.

Az indirekt bizonyításhoz tegyük fel, hogy van egy $\widehat{\mathbf{c}}$ vektor, amelynek súlya $w \leq d-1$, az $i \in \{a_1, a_2, \ldots, a_w\}$ pozíciókban tartalmazza nemzérus karaktereit és $\mathbf{H}\widehat{\mathbf{c}}^T = 0$. Ekkor az

$$\begin{pmatrix} \alpha^{a_1 i_0} & \cdots & \alpha^{a_w i_0} \\ \alpha^{a_1(i_0+1)} & \cdots & \alpha^{a_w(i_0+1)} \\ \vdots & \ddots & \vdots \\ \alpha^{a_1(i_0+w-1)} & \cdots & \alpha^{a_w(i_0+w-1)} \end{pmatrix} \begin{pmatrix} c_{a_1} \\ c_{a_2} \\ \vdots \\ c_{a_w} \end{pmatrix} = \mathbf{0}$$

egyenletben a mátrix determinánsa 0. De ezen mátrix determinánsa az ún. Vandermonde-determináns $\alpha^{(a_1+\cdots+a_w)i_0}$-szorosa:

$$\det\begin{pmatrix} 1 & \cdots & 1 \\ \alpha^{a_1} & \cdots & \alpha^{a_w} \\ \vdots & \ddots & \vdots \\ \alpha^{a_1(w-1)} & \cdots & \alpha^{a_w(w-1)} \end{pmatrix} = \prod_{j=1}^{w-1}\prod_{i=j+1}^{w}(\alpha^{a_i}-\alpha^{a_j})$$

A Vandermonde-determináns viszont nemzérus, mivel $\alpha^i \neq \alpha^j,\ i \neq j,\ i,j \in \{0,1,\ldots,n-1\}$ esetén, hiszen α rendje n. ■

Így, ha $\alpha, \alpha^2, \ldots, \alpha^{2t}$ a $g(x)$ gyökei, akkor $d \geq 2t+1$, tehát a kód t hibát képes javítani ($i_0 = 1$). Ez támasztja alá a BCH generátorpolinom fentebb mondott választását.

Ha $q = 2$, akkor bináris BCH-kódot kapunk, míg $m = 1$ esetben Reed–Solomon-kódra jutunk. A $t = 1, q = 2$ választással kapjuk a Hamming-kódokat. Mivel a GF(q) feletti polinomok között az $\alpha \in$ GF(q) testelem az $x-\alpha$ elsőfokú polinom gyöke, így a t hibát javító Reed–Solomon-kód generátorpolinomja lehet például a

$$g(x) = (x-1)(x-\alpha)(x-\alpha^2)\cdots(x-\alpha^{2t-1})$$

polinom ($i_0 = 0$), ahol α rendje n.

A $g(x)$ generátorpolinom konstrukciója

Mint láttuk, egy n kódszóhosszú GF(q) feletti lineáris ciklikus kód $g(x)$ generátorpolinomja az x^n-1 polinom osztója. Az állítás megfordítása is igaz: az x^n-1 polinom tetszőleges — nemtriviális — GF(q) feletti osztója egy $C(n,k)$, $\deg(g(x)) = n-k$ paraméterű GF(q) feletti lineáris ciklikus kód generátorpolinomja lehet. A linearitás nyilvánvaló, hiszen $a(x)g(x)+b(x)g(x) = (a(x)+b(x))g(x)$. A ciklikussághoz elég azt belátni, hogy egy ciklikus léptetés is kódszóra vezet. Ehhez tekintsük a következő két esetet: $\deg(a(x)) < k-1$ illetve $\deg(a(x)) = k-1$. Az első eset egyszerű, mivel $\deg(xa(x)g(x)) < n$. A második esetben az igazolás

$$xa(x)g(x) = a_{k-1}(x^n-1)+r(x)$$

maradékos osztás egyenlőségen alapul, ahol azt kell belátni az $r(x)$ maradékra, hogy $r(x) = a'(x)g(x)$ alakú valamely $a'(x)$ polinomra. Ez azonban egyszerűen következik a $g(x) \mid x^n-1$ feltétel felhasználásával.

A $g(x)$ generátorpolinomot tehát az x^n-1 polinom GF(q) feletti irreducíbilis faktorjaiból konstruáljuk, ahol legyen

$$x^n-1 = f_1(x)\,f_2(x)\cdots f_s(x) \tag{4.19}$$

a GF(q) feletti irreducíbilis faktorokra bontás. Ennek alapján az s különböző faktor összes lehetséges nemtriviális kombinációjával 2^s-2 különböző generátorpolinom — és hozzájuk tartozó n kódszóhosszú GF(q) feletti ciklikus kód — állítható elő.

Tekintsük az $n=q^m-1$ kódszóhosszakat, azaz úgynevezett primitív szóhosszú kódokat konstruálunk. A konstrukcióval kapcsolatos GF(q) testet szokás az együtthatók testének („kis testnek"), míg a GF(q^m) testet a gyökök testének („nagy testnek") nevezni. A testelméleti alapokból ismeretes, hogy tetszőleges GF(Q) test esetén

$$x^{Q-1}-1=\prod_j (x-\beta_j), \tag{4.20}$$

ahol a β_j gyökök a GF(Q) test különböző nemzérus elemei. A (4.19), (4.20) formulák alapján a primitív szóhosszú kódok esetén a β_j testelemek mindegyike egy és csak egy $f_i(x)$ faktor gyöke. Innen már az is látható, hogy ha a β_j testelem az $f_i(x)$ gyöke, akkor $f_i(x)$ irreducíbilis polinom egyben minimális fokszámú GF(q) feletti polinom ezen tulajdonsággal. Ha feltesszük ugyanis, hogy nem $f_i(x)$, hanem egy tőle különböző $k(x)$ GF(q) feletti polinom lenne minimális fokszámú polinom nevezett tulajdonsággal, az ellentmondásra vezetne. Ezt a szokásos trükkel igazolhatjuk: az $f_i(x)=q(x)k(x)+r(x)$ maradékos osztás egyenlőségbe helyettesítsük a β_j gyököt, s vegyük figyelembe, hogy $\deg(r(x))<\deg(k(x))$ valamint azt, hogy $f_i(x)$ irreducíbilis. Az $f_i(x)$ polinomot a β_j GF(q^m)-beli testelem GF(q) feletti minimálpolinomjának nevezzük.

4.12. példa. Legkisebb példaként tekintsük a $q=2, m=2$ esetet (azaz GF(2) a kis test, GF(4) a nagy test). Ekkor a (4.19), (4.20) formuláknak megfelelő felbontások a következő egyszerű alakot öltik:

$$x^3-1=f_1(x)f_2(x)=(x-1)(x^2+x+1)=(x-1)((x-\alpha)(x-\alpha^2)),$$

ahol α a GF(4) primitív eleme. Az $1\in$ GF(4) egységelem GF(2) feletti minimálpolinomja nyilván $f_1(x)=x-1$, míg α és α^2 GF(4) testelemek közös GF(2) feletti minimálpolinomja $f_1(x)=x^2+x+1$.

Visszatekintve a BCH-kód 4.26. definíciójára, az eddigiek alapján láthatjuk, hogy t hibát javító BCH-kód generátorpolinomját úgy konstruálhatjuk meg, hogy megkeressük az $\alpha^i\in$ GF(q^m) gyökök különböző, GF(q) feletti minimálpolinomjait, s azokat összeszorozzuk.

Ezek után már „csak" egy tisztán algebrai technikai kérdésről van szó, arról, hogy hogyan konstruáljunk minimálpolinomot. Az alábbiakban röviden megmutatjuk a technikát (részletesebben lásd például [10] 5.3. fejezet).

Ha egy $f(x)$ polinom egy β GF(q^m)-beli testelem GF(q) feletti minimálpolinomja, akkor $f(x)$ gyökhalmaza a β elem úgynevezett konjugáltjainak

$$\left\{\beta, \beta^q, \beta^{q^2}, \ldots, \beta^{q^{\nu-1}}\right\}$$

halmaza, amely halmaz elemeit β egymás után, q-adik hatványra emelésével képezzük a GF(q^m) testben, továbbá ahol ν az a legkisebb hatványkitevő, amelyre

$$\beta^{q^\nu} = \beta.$$

Mindezek alapján gyöktényezős (nagy test feletti linearizált) alakban meg tudjuk már adni a minimálpolinomokat. Ebből az alakból az explicit alakú polinomokat a lineáris faktorok összeszorzásával kapjuk a nagy testbeli aritmetika szerint, amelynek során az együtthatók végülis kis testbeli együtthatókká egyszerűsödnek.

A 4.12. példa esetén α konjugáltjainak halmaza kételemű, tagjai α és α^2 ($\alpha^4 = \alpha$).

Az eddigiek alapján anélkül, hogy megkonstruálnánk magát a generátorpolinomot, azt már meg tudjuk mondani, hogy mekkora lesz a generátorpolinom fokszáma, következésképp hogy mekkora a kód k paramétere illetve $R = \frac{k}{n}$ kódsebessége. Ehhez nem kell mást tennünk, mint konjugált halmazokra bontani a nagy test nemzérus elemeinek $q^m - 1$ méretű halmazát. Mivel a nemzérus testelemek a primitív elem hatványaiként megkaphatók, a konjugált halmazok meghatározásánál elegendő a halmaz elemei helyett a megfelelő primitív elem kitevőjét megadni.

4.13. példa. GF(16), $q = 2, m = 4$:

$$\begin{array}{l}\{0\}\\ \{1,2,4,8\}\\ \{3,6,12,9\}\\ \{5,10\}\\ \{7,14,13,11\}\end{array}$$

ahol — primitív elem kitevőiről lévén szó — modulo 15 számolunk. Ennek alapján például egy $t = 1$ paraméterű kód generátorpolinomjához tartozó konjugált (kitevő) halmaz $\{1,2,4,8\}$, azaz $C(15,11)$ a kapcsolatos bináris Hamming-kód paramétere. Hasonlóan, ha egy $n = 15$ kódszóhosszú $t = 2$ hibát javító bináris BCH-kódot szeretnénk generálni, akkor $\{1,2,4,8\}$ és $\{3,6,12,9\}$ kitevőhalmazoknak megfelelő gyökök szerepelnek a generátorpolinomban, tehát a kód sebessége $R = \frac{7}{15} = 0.466$ lesz.

A BCH kódok előállíthatók mint a Reed-Solomon kódok részkódjai. Ehhez kapcsolódik az alábbi tétel:

4.24. tétel. *Legyen C a GF(q^m) felett egy (n,k) paraméterű, d_{min} kódtávolságú Reed–Solomon-kód. Legyen C' a C-nek egy részhalmaza, mely C azon kódszavaiból áll, melyek koordinátái GF(q)-beliek. C' egy GF(q) feletti (n,k') paraméterű, d'_{min} kódtávolságú, lineáris kód, melyre*

$$k' \leq k$$

és

$$d'_{min} \geq d_{min}.$$

A 4.24. tétel bizonyítását az olvasóra bízzuk. A linearitás triviális, másrészt C'-ben nyilván nincs több lineárisan független kódszó, mint C-ben, harmadrészt egy részkód kódtávolsága nem lehet kisebb, mint az eredeti kódé. A kapott részkód 4.26. definíció szerinti BCH-kód $t \geq (n-k)/2$ paraméterrel.

4.10. Kódkombinációk

A standard kódkonstrukciók során kapott kódok paraméterei nem mindig illeszkednek közvetlenül az adott alkalmazásban megkövetelt értékekhez. Hatékony, ugyanakkor egyszerű módszerek léteznek arra, hogy változtassuk a kódszóhossz, üzenethossz, kódtávolság paraméterek értékét az eredeti konstrukcióhoz képest. Az alábbiakban ezen módszereket tekintjük át röviden.

Kódátfűzés és a csomós hibák javítása

Adott $C(n,k)$ kód m-szeres **átfűzés**ével egy $C^m = C(mn,mk)$ kódot kapunk, olyan módon, hogy a C kód $\mathbf{c}^{(i)}$, $i = 1,\ldots,m$ m darab kódszavát egy $m \times n$ dimenziós mátrixba rendezzük soronként, s a C^m átfűzéses kód $\mathbf{c}$ kódszavát ezen mátrix oszlapainak sorrendben való kiolvasásával képezzük. Azaz a kódszavakat (komponens szavakat) fésű módon egymásba toljuk:

$$\mathbf{c} = \left(c_0^{(1)}, c_0^{(2)}, \ldots, c_0^{(m)}, c_1^{(1)}, c_1^{(2)}, \ldots, c_1^{(m)}, \ldots, c_{n-1}^{(1)}, c_{n-1}^{(2)}, \ldots, c_{n-1}^{(m)}\right) \qquad (4.21)$$

Lineáris kódot átfűzve nyilván lineáris kódot kapunk. Az is könnyen látható, hogy ha d a C kód kódtávolsága, akkor a C^m átfűzéses kód távolsága is d marad. Lineáris C kódot tekintve legyen $\mathbf{c}^{(i)}$, $i = 1,\ldots,m$ sorozat egyik kódszavának súlya d, míg a többi kódszó legyen a zérus kódszó. Általános esetben tekintsük C kódbeli kódszavak $\mathbf{c}^{(1,i)}$, $i = 1,\ldots,m$, $\mathbf{c}^{(2,i)}$, $i = 1,\ldots,m$ két sorozatát, ahol $\mathbf{c}^{(1,1)}$ és $\mathbf{c}^{(2,1)}$ távolsága d, míg $\mathbf{c}^{(1,i)} = \mathbf{c}^{(2,i)}$, $i = 2,\ldots,m$. (A CD példájában $m = 2, n = 28, k = 24$.)

Ciklikus kódot átfűzve ciklikus kódot kapunk. Legyen S az egyszeri ciklikus jobbra léptetés operátora. Könnyen ellenőrizhető, hogy a (4.21) szerinti $\mathbf{c}$ kódszó $S\mathbf{c}$ ciklikus eltolása az $S\mathbf{c}^{(m)}, \mathbf{c}^{(1)}, \mathbf{c}^{(2)}, \ldots, \mathbf{c}^{(m-1)}$ sorozat átfűzésének felel meg, s mivel $S\mathbf{c}^{(m)} \in C$, ezért $S\mathbf{c} \in C^m$ is fennáll. A C és a C^m kódok generátorpolinomja között egyszerű kapcsolat áll fenn:

4.25. tétel. *Ha $g(x)$ a C kód generátorpolinomja, akkor $g(x^m)$ a C^m kód generátorpolinomja.*

BIZONYÍTÁS: Mivel $g(x)$ — mint generátorpolinom — a C kód egyben legkisebb fokszámú nemzérus főpolinom kódszava, ezért a $c^{(1)}(x) = g(x), c^{(2)}(x) = 0, \ldots, c^{(m)}(x) = 0$ sorozat átfűzésével kapható kódszó polinom alakban $g(x^m)$, a C^m kód legkisebb fokszámú nemzérus főpolinom kódszava, azaz generátorpolinomja. ■

Mint láttuk, a kódtávolság nem változik átfűzés során, ami azt jelenti, hogy a C^m átfűzéses kód szokásos képességei (véletlen hibák javítása, törlésjavítás, detekciós képesség) romlanak az átfűzéssel, hiszen ezen képességek m-szeres kódszóhosszon érvényesek. Ha valaki itt arra gondolna, hogy például az egyes komponensszavak javítóképessége nem változott, s így a teljes javító képesség a komponensek m-szeresének tűnik, az ott hibázik, hogy t javítóképesség azt jelenti, hogy tetszőleges t pozícióban eshet hiba, nem pedig azt, hogy az a komponens szavaknak megfelelően kerül „szétosztásra”. Ezen a ponton felmerül a természetes kérdés: egyáltalán mire jó akkor az átfűzés? A válasz: hibacsomók javítására.

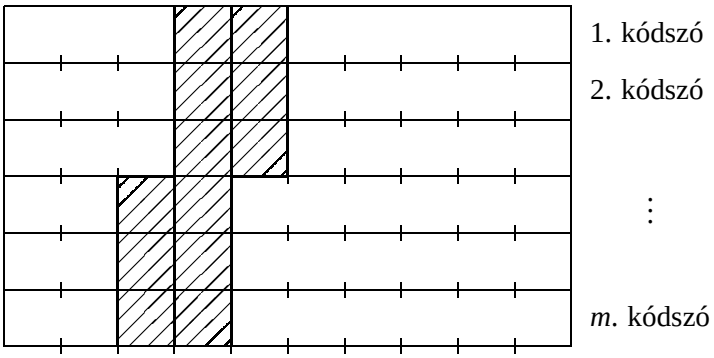
A hibavektor egy l hosszúságú szegmense **hibacsomó** l hosszal, ha a szegmens első és utolsó karaktere nem zérus. Egy kód l hosszúságú hibacsomót javító, ha minden legfeljebb l hosszúságú hibacsomó javítható.

4.26. tétel. *A C^m átfűzéses kód $m \cdot t$ hosszúságú hibacsomót javító, ahol t a C kód javítóképessége.*

BIZONYÍTÁS: A (4.21) szerinti $\mathbf{c}$ kódszóban egy legfeljebb $m \cdot t$ hosszúságú hibacsomónak megfelelő hibázás a komponens szavakban legfeljebb t számú hibát okozhat, amit azok javítani képesek. (A $t = 2$ esetet szemlélteti a 4.6. ábra.) ■

Egy tetszőleges lineáris $C(n,k)$ kód n,k paramétere alapján a kód l **hibacsomójavító képesség**ére az alábbi egyszerű korlát adható:

4.27. tétel. *Egy $C(n,k)$ lineáris kód l hibacsomójavító képességére fennáll, hogy $l \leq \left\lfloor \frac{n-k}{2} \right\rfloor$.*



4.6. ábra. Kódátfűzés $t = 2$ esetén.

BIZONYÍTÁS: Tekintsünk egy tetszőleges nemzérus kódszót, s abban a leghosszabb, nemzérussal kezdődő, s nemzérussal végződő részsorozatot (a továbbiakban csomórészsorozat). Tegyük fel, hogy az összes nemzérus kódszót tekintve a legrövidebb ilyen részsorozat hossza $b+1$. Ekkor egy adott kódszópozícióban kezdődő, legfeljebb b hosszú hibacsomóknak megfelelő hibavektorok a standard elrendezési táblázatban különböző sorokba (mellékosztályokba) kell, hogy essenek, ellenkező esetben két ilyen hibavektor különbsége kódszó lenne, ami ellentmondásra vezetne. Mivel a táblázat sorainak száma q^{n-k}, továbbá a különböző legfeljebb b hosszú hibacsomók száma q^b, ezért $q^b \leq q^{n-k}$, ahonnan $b \leq n-k$ adódik. Tehát van olyan kódszó, amelyben a leghosszabb csomórészsorozat hossza legfeljebb $n-k+1$. Ha ezen kódszó ezen csomórészsorozatát szétvágjuk két rövidebb csomórészsorozatra, akkor az azoknak megfelelő hibavektorok azonos mellékosztályba kell, hogy essenek, hiszen ezen vektorok összege kódszó, következésképpen csak egyikük választható mellékosztály-vezetőnek, vagyis javítható hibamintának. Innen már következik, hogy garantálhatóan legfeljebb az $\lfloor \frac{n-k}{2} \rfloor$ hosszú hibacsomók javíthatók. ■

A tételbeli korlát Reiger-korlát néven ismert. Azokat a hibacsomó javító kódokat, amelyre $l = \lfloor \frac{n-k}{2} \rfloor$ fennáll, **Reiger-optimális**nak hívjuk.

MEGJEGYZÉS: Egy MDS tulajdonságú lineáris kód Reiger-optimális.

4.14. példa. A $g(x) = (x+1)(x^3+x+1) = x^4+x^3+x^2+1$ generátorpolinomú $C(7,3,4)$, $GF(2)$ kód hibacsomójavító képessége $l = 2$, azaz Reiger-optimális. Ennek egy praktikus ellenőrzési módja az, hogy megmutatjuk, hogy az x^i, $i = 0,\ldots,6$ és $x^i + x^{i+1}$, $i = 0,\ldots,5$ polinomok osztási maradékai különbözőek a $g(x)$ generátorpolinomra.

$$\left[\begin{array}{cccc|ccc}
c_0^{(0)} & c_1^{(0)} & \cdots & c_{k_1-1}^{(0)} & c_{k_1}^{(0)} & \cdots & c_{n_1-1}^{(0)} \\
c_0^{(1)} & c_1^{(1)} & \cdots & c_{k_1-1}^{(1)} & c_{k_1}^{(1)} & \cdots & c_{n_1-1}^{(1)} \\
\vdots & & & & & & \vdots \\
c_0^{(k_2-1)} & c_1^{(k_2-1)} & \cdots & c_{k_1-1}^{(k_2-1)} & c_{k_1}^{(k_2-1)} & \cdots & c_{n_1-1}^{(k_2-1)} \\
\hline
\vdots & & & & & & \vdots \\
c_0^{(n_2-1)} & c_1^{(n_2-1)} & \cdots & c_{k_1-1}^{(n_2-1)} & c_{k_1}^{(n_2-1)} & \cdots & c_{n_1-1}^{(n_2-1)}
\end{array}\right]$$

4.7. ábra. A szorzat-kódszó képzése.

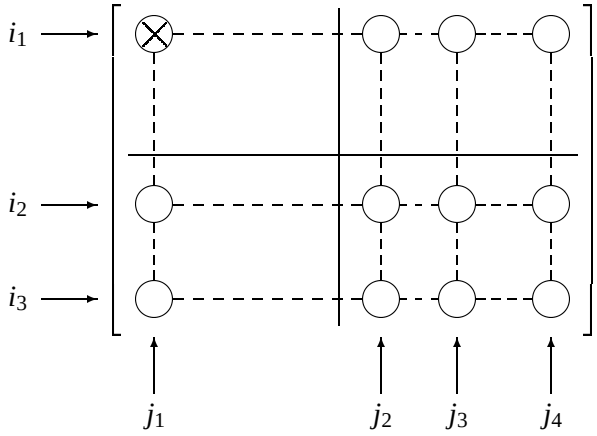
Szorzatkód

Egy $C_1(n_1,k_1,d_1)$ és egy $C_2(n_2,k_2,d_2)$ lineáris kód (komponenskódok) felhasználásával $C_1\times C_2(n_1\cdot n_2,k_1\cdot k_2,d_1\cdot d_2)$ **szorzatkód**ot készíthetünk, amelynek kódszavai $n_1\times n_2$ dimenziós mátrixok, ahol a mátrix sorai C_1 kódbeli, oszlopai C_2 kódbeli kódszavak. Szisztematikus komponenskódok esetén a szorzatkódbeli mátrix-kódszó bal felső $k_1\times k_2$ dimenziós minorja tartalmazza az üzenetet. A mátrix-kódszavakat soronként kiolvasva kapjuk a szorzatkód — soros — kódszavát. A kapott $C_1\times C_2(n_1\cdot n_2,k_1\cdot k_2)$ kód lineáris.

A mátrix-kódszó képzése a következőképp történik. Az első k_1 oszlopot a $C_2(n_2,k_2)$ kód alapján szisztematikus kódolással kapjuk, kiegészítve a k_2 hosszú üzenetszegmenst n_2-k_2 hosszú paritásszegmenssel (4.7. ábra). Az első k_2 sort a $C_1(n_1,k_1)$ kód alapján szisztematikus kódolással kapjuk, kiegészítve a k_1 hosszú üzenetszegmenst n_1-k_1 hosszú paritásszegmenssel. A mátrix jobb alsó sarkába kerül a paritások paritása, amit — mint azt hamarosan belátjuk — képezhetjük akár az első k_1 oszlop, akár az első k_2 sor paritásai alapján szisztematikus kódolással a C_2 illetve C_1 kódbeli szavakkal. A fenti módon képezett szorzatkódot — amelynek sorai illetve oszlopai az alapkódok kódszavai — kanonikus elrendezésűnek nevezzük.

A paritások paritásai képzésével kapcsolatos alábbi gondolatmenetünket illusztrálja a 4.8. ábra.

Képezzük azt a $C_1\times C_2$ kódbeli kódszót, amelynek üzenetmátrixa csak az (i_1,j_1) koordinátájú helyen tartalmaz nullától különböző elemet. Ehhez az üzenethez képezzük a C_2 illetve C_1 kódolás szerint a $C_1\times C_2$ kódbeli kódszó i_1-edik sorát és j_1-edik oszlopát. A kétdimenziós paritásszegmens jobb felső illetve bal alsó részmátrixa az i_1-edik sor illetve a j_1-edik oszlop kivételével csak 0 elemeket tartalmaz. Innen már egyszerűen látszik, hogy a jobb alsó részmátrixot megkaphat-



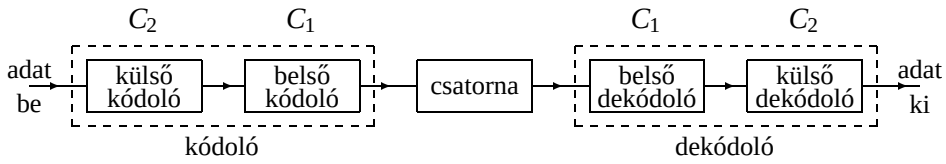
4.8. ábra. A paritások paritásainak képzése.

juk, akár az (i_2, j_1) illetve (i_3, j_1) elemekből C_2 kód szerinti, akár $(i_1, j_2), (i_1, j_3)$ illetve (i_1, j_4) elemekből C_1 kód szerinti kódolással. S miután a $C_1 \times C_2$ kód lineáris, ezért tetszőleges üzenetmátrixú szorzatkód kódszót a 4.8. ábrán is illusztrált elem-kódszavakból koordinátánként vett összeadással képezhetjük.

Annak igazolása, hogy a szorzatkód kódtávolsága a komponens kódok távolságainak szorzata, a minimális nemzérus súlyú, azaz $d_1 \cdot d_2$ súlyú kódszó előállításával történhet. Válasszunk ehhez egy-egy minimális súlyú kódszót a C_1 illetve C_2 kódból, amelyeket jelöljön $\mathbf{c}'$ illetve $\mathbf{c}''$, ekkor egy minimális súlyú mátrix-kódszó az i-edik sorában a $\mathbf{c}''$ kódszót tartalmazza, ha $\mathbf{c}'$ i-edik komponense 1, egyébként a csupa zérus kódszó kerül a sorba. Az, hogy a kapott kódszó minimális súlyú, onnan látható, hogy ha nem minimális súlyú $\mathbf{c}''$ kódszót helyeznénk el valamelyik sorba, akkor több nemzérus oszlopot kellene elhelyezni a mátrixban a C_1 kódszavai közül és viszont.

4.15. példa. Az egyik legismertebb és egyben legegyszerűbb konstrukciójú hibajavító kód a **kétdimenziós paritáskód**. Ez egy $C \times C$ szorzatkód, ahol a C komponenskód $(n, n-1, 2)$ paraméterű egy paritásbittel rendelkező, egy hibát jelző bináris kód. A kapott szorzatkód kódtávolsága 4, azaz egyszerű paritásbites konstrukcióval 1 hiba javítására vagy 3 hiba jelzésére alkalmas kódot kaptunk.

A mátrix-kódszóban a komponens kódszavak eredeti elrendezése a kanonikus elrendezés. Ha a C_1, C_2 komponenskódok ciklikusak, akkor egy másfajta mátrixbeli elrendezéssel, az úgynevezett ciklikus elrendezéssel elérhetjük, hogy a soronkénti kiolvasással kapott kódszó ciklikus legyen. Ha a $C_1(n_1, k_1, d_1)$ és



4.9. ábra. Kaszkád kódoló.

$C_2(n_2,k_2,d_2)$ komponenskódok n_1 és n_2 szóhosszai relatív prímek, akkor a kanonikus mátrixbeli pozíciókat úgy rendezzük át, hogy az $n_1 \times n_2$ dimenziós mátrix $(u,v), 0 \le u < n_1, 0 \le v < n_2$ koordinátájú eleme az $(u',v'), 0 \le u' < n_1, 0 \le v' < n_2$ pozícióba kerüljön, ahol $u' = u \cdot n_2 + v \mod n_1$, $v' = u \cdot n_2 + v \mod n_2$. A pozíciók ezen megfeleltetése a kínai maradéktétel (lásd a következő fejezetben) miatt egyértelmű a relatív prím szóhosszak esetén.

Kaszkád kódok

Vegyünk egy $C_1(n_1,k_1,d_1)$ GF(q) feletti és egy $C_2(N_2,K_2,D_2)$ GF(q^{k_1}) feletti lineáris kódot, amelyből az alábbi módon generálhatjuk a szisztematikus, $C(n_1N_2,k_1K_2,d)$ paraméterű GF(q) feletti **kaszkád kód** kódszavait. A k_1K_2 hosszú üzenetet osszuk fel K_2, egyenként k_1 hosszú szegmensre. A C_2 kód egy k_1 hosszú üzenetszegmenst egy üzenetkarakternek vesz, és K_2 ilyen karakter alkot számára egy üzenetszegmenst, amelyből N_2 karakter hosszúságú kódszót képez $N_2 - K_2$ paritáskarakternek az üzenethez való illesztésével. A C_2-beli kódszó elkészülte után a kódszó mindegyik koordinátáját a C_1 kód kódolója újra k_1 hosszúságú üzenetként értelmezi, és $n_1 - k_1$ paritáskarakterrel kiegészíti. Így kapjuk az n_1N_2 hosszú kódszót, ami a kaszkád kód adott k_1K_2 hosszú üzenethez tartozó kódszava. A kaszkád kód kódtávolsága $d \ge d_1D_2$.

A C_1 kódot **belső**, a C_2 kódot **külső kód**nak is nevezik. A kód az elnevezését onnan kapta, hogy a külső kód kódolójának és a belső kód kódolójának a kaszkádba kötése képezi a generált kód kódolóját (4.9. ábra).

A dekódolás során először a C_1 kódszavakat dekódoljuk, majd értelemszerűen, a C_1 kódszavai paritásszegmensének törlése után a C_2 kódszó dekódolását végezzük el.

A kaszkád kódok igen alkalmasak az együttes csomós és véletlen hibák javítására, ahol a csomós hibákat a C_2 kód, a véletlen hibákat a C_1 kód javítja elsősorban. A C_2 kód egy karakterének tetszőleges meghibásodása legfeljebb k_1 méretű q-áris hibaszámnak felel meg. Ugyanakkor ritka egyedi hibák javítása C_1-beli kódszavakban könnyen elvégezhető, míg ezen egyedi hibák C_2-beli karakterszintű javítása „pazarlás” lenne.

4.11. Kódmódosítások

Rövidített kód

Egy $C(n,k)$ kód **rövidítés**ével egy $C(n-i,k-i)$, $1 \leq i < k$ kódot kapunk olyan módon, hogy a $C(n,k)$ szisztematikus kód kódszavai közül csak azokat hagyjuk meg, amelyek az első i karakterén zérust tartalmazó üzenetekhez rendeltek. Ekkor a $C(n,k)$ kód i karakterrel történő rövidítéséről beszélünk. Mivel a rövidített kód kódszavai a $C(n,k)$ kód kódszavai is egyben, ezért a rövidített kód minimális távolsága legalább akkora, mint az eredeti kódé volt. Praktikusan természetesen a kódoló és a dekódoló úgy van kiképezve, hogy az első i zérus karaktert nem is továbbítjuk, s a dekóder zérusnak tekinti azokat. A kódrövidítés elsődleges célja a kódhossznak az alkalmazásbeli paraméterekhez való igazítása.

Paritásbittel bővítés

A paritáskarakterrel történő kiegészítés után a $C(n,k)$ bináris lineáris alapkódból egy $\widehat{C}(n+1,k)$ lineáris kódot kapunk, amelynek a minimális távolsága az alapkód d minimális távolságával azonos, ha d páros, illetve $d+1$ lesz, ha d páratlan. A $\mathbf{H}_{\widehat{C}}$ paritásmátrix $\mathbf{H}_C$ ismeretében az alábbi alakú:

$$\mathbf{H}_{\widehat{C}} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ & & & & 0 \\ & & \mathbf{H}_C & & \vdots \\ & & & & 0 \\ & & & & 0 \end{pmatrix} \tag{4.22}$$

4.16. példa. Adjuk meg a $C(7,4)$ Hamming-kód $\widehat{C}(8,4)$ paritásbittel bővített kódjának paritásmátrixát. Az x^3+x+1 generátorpolinomú Hamming-kód bővítésével a (4.22) képlet alapján

$$\mathbf{H}_{\widehat{C}} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

A kapott $\widehat{C}(8,4)$ kód nyilván nem ciklikus, például a (10001101) a $\widehat{C}$ kódszava, de (11000110) már nem az. A bővített kód minimális távolsága 4, ezért 3 véletlen hiba detektálására alkalmas.

4.12. Reed–Solomon-kódok dekódolása

Tekintsünk egy (n,k) paraméterű GF(q) feletti lineáris kódot. Egy $\mathbf{c}$ kódszó küldésekor és $\mathbf{v}$ szó vételekor az $\mathbf{e} = \mathbf{v} - \mathbf{c}$ hibavektorra és $\mathbf{s}$ szindrómára

$$\mathbf{s}^T = \mathbf{H}\mathbf{v}^T = \mathbf{H}(\mathbf{c}+\mathbf{e})^T = \mathbf{H}\mathbf{c}^T + \mathbf{H}\mathbf{e}^T = \mathbf{H}\mathbf{e}^T$$

teljesül.

Ezen egyenlőség alapján az n hosszú $\mathbf{e}$ hibavektor rekonstrukciójához az $n-k$ hosszú $\mathbf{s}$ szindróma áll rendelkezésre. A vett szóhoz legközelebbi kódszóba javításkor egy lineáris egyenletrendszer egy speciális tulajdonságú, mégpedig a legkisebb súlyú megoldását keressük. A minimális súlyú $\mathbf{e} = (e_0, e_1, \ldots, e_{n-1})$ hibavektor rekonstrukciójához a következő ismeretlenek felderítésére van szükség:

- a hibák száma: t,
- a hibák helye: $0 \le i_1 < i_2 < \cdots < i_t \le n-1$,
- a hibák értéke: $e_{i_1}, e_{i_2}, \ldots, e_{i_t}$.

Ha csak törléses hibánk van, akkor ismerjük a hibák számát és a hibahelyeket, csupán a hibaértékeket kell megtalálni. A Reed–Solomon-kódok 4.1. és 4.2. konstrukciója kapcsán éppen a törléses hibák lehetséges maximális számának kiderítésével bizonyítottuk, hogy a Reed–Solomon-kód maximális távolságú. Ezt úgy tettük, hogy az

$$\mathbf{u}\mathbf{G} = \mathbf{c}$$

egyenletrendszer azon részrendszerét tudtuk megoldani, amely a nem törölt pozíciók halmazának felelt meg. Ilymódon persze közvetlenül az $\mathbf{u}$ üzenetvektort kaptuk meg. Annak érdekében, hogy az általános hibajavítási feladat harmadik komponensére adott megoldás alapelvét megértsük, vizsgáljuk meg a törléses hibák javítását, mégpedig a hibaértékek megállapítását a Reed–Solomon-kódok 4.3. konstrukciójakor, amikor a kódot a

$$\mathbf{H}\mathbf{c}^T = 0$$

paritásegyenlet megoldásainak a halmaza adta, ahol

$$\mathbf{H} = \begin{pmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & & & \ddots & \vdots \\ 1 & \alpha^{n-k} & \alpha^{2(n-k)} & \cdots & \alpha^{(n-k)(n-1)} \end{pmatrix}.$$

A vett szó törléses pozícióiba írjunk 0-t, és az így kapott **v** szó esetén számítsuk ki az $\mathbf{s}^T = \mathbf{H}\mathbf{v}^T$ szindrómát. Ekkor az **e** hibavektort az

$$\mathbf{s}^T = \mathbf{H}\mathbf{e}^T$$

egyenlet megoldása adja. Vezessük be a következő jelöléseket:

$$\mathbf{s} = (s_1, s_2, \ldots, s_{n-k})$$

$$X_j = \alpha^{i_j}$$

$$Y_j = e_{i_j}$$

$(j = 1, 2, \ldots, t)$. X_j-t a j-edik **hibahely lokátor**ának nevezzük, mert α szerinti logaritmusa a hibahely. Az $\mathbf{s}^T = \mathbf{H}\mathbf{e}^T$ egyenletrendszer l-edik egyenlete

$$\sum_{i=0}^{n-1} e_i \alpha^{li} = \sum_{j=1}^{t} e_{i_j} \alpha^{l i_j} = s_l, \qquad l = 1, 2, \ldots, n-k,$$

vagy az új jelöléseinkkel

$$\sum_{j=1}^{t} Y_j X_j^l = s_l, \qquad l = 1, 2, \ldots, n-k.$$

Tudjuk, hogy **H** minden $(n-k) \times (n-k)$ méretű négyzetes részmátrixa invertálható (lásd a 4.15. tétel utáni megjegyzést), sőt $t \leq n-k$ esetén annak „bal felső” $t \times t$-es részmátrixa is invertálható, tehát $t \leq n-k$ esetén az egyenletrendszerünknek egyértelmű megoldása van, amennyiben $X_1, X_2, \ldots, X_t$ különbözők.

4.7. lemma. *Legyen*

$$\mathbf{A}_t = \begin{pmatrix} X_1 & X_2 & \cdots & X_t \\ X_1^2 & X_2^2 & \cdots & X_t^2 \\ \vdots & \vdots & \ddots & \vdots \\ X_1^t & X_2^t & \cdots & X_t^t \end{pmatrix},$$

$\mathbf{S}_t = (s_1, s_2, \ldots, s_t)$ *és* $\mathbf{Y}_t = (Y_1, Y_2, \ldots, Y_t)$. *Ha* $X_1, X_2, \ldots, X_t$ *mind különbözőek, és* $t \leq n-k$, *akkor* $\mathbf{A}_t$ *invertálható, azaz az*

$$\mathbf{A}_t \mathbf{Y}_t^T = \mathbf{S}_t^T$$

egyenletnek van egyértelmű megoldása. Ha $X_1, X_2, \ldots, X_t$ *nem mind különbözőek, akkor* $\mathbf{A}_t$ *nem invertálható.*

BIZONYÍTÁS: A lemma első felét az előzőekben igazoltuk. A második fele azért igaz, mert ekkor az $\mathbf{A}_t$-nek van legalább két egyforma oszlopa. ■

Tekintsük ezek után az általános hibajavítási feladatot! Ekkor is ugyanabból az egyenletből indulunk ki:

$$\sum_{j=1}^{t} Y_j X_j^l = s_l, \qquad l = 1,2,\ldots,n-k.$$

Most az ismeretlenek:

- t,
- $X_1, X_2, \ldots, X_t$,
- $Y_1, Y_2, \ldots, Y_t$,

tehát egy $2t+1$ ismeretlent tartalmazó, $n-k$ egyenletből álló, nemlineáris egyenlettel van dolgunk. Ennek a megoldása elsőre nem tűnik egyszerű feladatnak. Megmutatjuk, hogy ez visszavezethető két lineáris egyenletrendszer megoldására, melyek közül a másodikat a csak törléses hiba kapcsán az előbb már megismertük. Az első egyenletrendszernek kell produkálnia t-t és a hibahelyeket.

Vezessük be a **hibahelypolinom**ot:

$$L(x) = \prod_{i=1}^{t} (1 - xX_i),$$

melynek együtthatói $1, L_1, \ldots, L_t$, azaz $L(x) = 1 + L_1 x + \cdots + L_t x^t$. $L(x)$-et azért nevezzük hibahelypolinomnak, mert a gyökei a hibalokátorok inverzei: X_1^{-1}, $X_2^{-1}, \ldots, X_t^{-1}$. Ha megtaláltuk az $L(x)$ polinomot, akkor annak t darab gyöke is meghatározható (például az összes nem 0 elem behelyettesítésével), majd azok inverzei adják a hibalokátorokat.

Mivel X_j^{-1} az $L(x)$ gyöke, ezért minden l-re és j-re

$$Y_j X_j^{l+t} L(X_j^{-1}) = 0,$$

tehát

$$\sum_{j=1}^{t} Y_j X_j^{l+t} L(X_j^{-1}) = 0,$$

azaz

$$\sum_{j=1}^{t} Y_j \left(X_j^{l+t} + L_1 X_j^{l+t-1} + L_2 X_j^{l+t-2} + \cdots + L_t X_j^l \right) = 0,$$

tehát

$$\sum_{j=1}^{t} Y_j X_j^{l+t} + L_1 \sum_{j=1}^{t} Y_j X_j^{l+t-1} + L_2 \sum_{j=1}^{t} Y_j X_j^{l+t-2} + \cdots + L_t \sum_{j=1}^{t} Y_j X_j^{l} = 0,$$

így

$$L_1 s_{l+t-1} + L_2 s_{l+t-2} + \cdots + L_t s_l = -s_{l+t}$$

$(l = 1, \ldots, t)$. Ha bevezetjük az

$$\mathbf{U}_t = \begin{pmatrix} s_1 & s_2 & \cdots & s_t \\ s_2 & s_3 & \cdots & s_{t+1} \\ \vdots & \vdots & \ddots & \vdots \\ s_t & s_{t+1} & \cdots & s_{2t-1} \end{pmatrix},$$

$$\mathbf{L}_t = (L_t, L_{t-1}, \ldots, L_1),$$

$$\mathbf{V}_t = (-s_{t+1}, -s_{t+2}, \ldots, -s_{2t})$$

jelöléseket, akkor a következő lineáris egyenletrendszerre jutunk:

$$\mathbf{U}_t \mathbf{L}_t^T = \mathbf{V}_t^T.$$

4.8. lemma. $\mathbf{U}_r$ *invertálható, ha* $r = t$. $\mathbf{U}_r$ *nem invertálható, ha* $r > t$.

BIZONYÍTÁS: Vezessünk be két mátrixot: az egyik a Vandermonde-mátrix $B_{ij} = X_j^{i-1}$ elemekkel

$$\mathbf{B}_r = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ X_1^1 & X_2^1 & \cdots & X_r^1 \\ \vdots & \vdots & \ddots & \vdots \\ X_1^{r-1} & X_2^{r-1} & \cdots & X_r^{r-1} \end{pmatrix},$$

ahol $r > t$ esetén $X_r = 0$. A másik egy diagonálmátrix $D_{ij} = Y_i X_i \delta_{ij}$ elemekkel

$$\mathbf{D}_r = \begin{pmatrix} Y_1 X_1 & 0 & \cdots & 0 \\ 0 & Y_2 X_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Y_r X_r \end{pmatrix},$$

ahol δ_{ij} a Kronecker-szimbólum. Akkor $\mathbf{B}_r \mathbf{D}_r \mathbf{B}_r^T$ elemei

$$(\mathbf{B}_r \mathbf{D}_r \mathbf{B}_r^T)_{ij} = \sum_{l=1}^{r} X_l^{i-1} \sum_{k=1}^{r} Y_l X_l \delta_{lk} X_k^{j-1} =$$

$$\begin{aligned} &= \sum_{l=1}^{r} X_l^{i-1} Y_l X_l X_l^{j-1} = \\ &= \sum_{l=1}^{r} Y_l X_l^{i+j-1} = \\ &= s_{i+j-1} = \\ &= (\mathbf{U}_r)_{ij}, \end{aligned}$$

tehát

$$\mathbf{B}_r \mathbf{D}_r \mathbf{B}_r^T = \mathbf{U}_r,$$

ezért

$$\det(\mathbf{B}_r)\det(\mathbf{D}_r)\det(\mathbf{B}_r^T) = \det(\mathbf{U}_r).$$

Ha $r > t$, akkor $\det(\mathbf{D}_r) = 0$, ezért $\det(\mathbf{U}_r) = 0$. Ha $r = t$, akkor $\det(\mathbf{D}_r) \neq 0$, másrészt $\det(\mathbf{B}_r) \neq 0$ a 4.7. lemma miatt, ugyanis $X_1, X_2, \ldots, X_t$ különbözők, és $\det(\mathbf{A}_r) = \det(\mathbf{B}_r) \cdot X_1 \cdot X_2 \cdot \ldots \cdot X_t$. Ekkor viszont $\det(\mathbf{U}_r) \neq 0$, tehát $\mathbf{U}_t$ invertálható. ■

4.28. tétel (Peterson–Gorenstein–Zierler-dekódoló). *Ha a Reed–Solomon-kódok 4.3. konstrukciója esetén a hibák t számára $t \leq \left\lfloor \frac{n-k}{2} \right\rfloor$, akkor a következő algoritmus helyesen dekódol:*

1. *Számítsuk ki az $s_1, s_2, \ldots, s_{n-k}$ szindrómákat!*
2. *Keressük meg azt a legnagyobb r-et, melyre $\mathbf{U}_r$ invertálható! Ez lesz a t.*
3. *Oldjuk meg az*
$$\mathbf{U}_t \mathbf{L}_t^T = \mathbf{V}_t^T$$
egyenletet!
4. *Keressük meg az $\mathbf{L}_t$ által definiált $L(x)$ hibahelypolinom gyökeit, majd azok inverzeit: $X_1, X_2, \ldots, X_t$.*
5. *Oldjuk meg az*
$$\mathbf{A}_t \mathbf{Y}_t^T = \mathbf{S}_t^T$$
egyenletet!
6. *Számítsuk ki a hibahelyeket:*
$$i_j = \log X_j$$
és a hibaértékeket:
$$e_{i_j} = Y_j, \qquad j = 1, 2, \ldots, t.$$

A 4.28. tétellel nem azt akarjuk mondani, hogy nagy t esetén így kell dekódolni. Nagy t esetén nem ajánlatos a két mátrixinverziót végrehajtani. Ezek helyettesíthetők hatékonyabb eljárásokkal. Az első helyettesítésére (2. és 3. lépés) példa a Berlekamp–Massey-algoritmus, míg a másodikéra (5. lépés) a Forney-algoritmus (lásd [10]).

Kis t-re viszont a Peterson–Gorenstein–Zierler-dekódoló nemcsak egy elvi, hanem egy gyakorlati eljárás is. Nézzük végig illusztrációként a 2 hibát javító $(n, n-4)$ paraméterű Reed–Solomon-kód esetét, amelynek digitális hangtechnikai alkalmazását a 4.6. szakaszban említettük.

1. Számítsuk ki az s_1, s_2, s_3, s_4 szindrómákat!

2. Ha mind 0, akkor $t = 0$, és készen vagyunk. Mivel ez a kód 2 hibát tud javítani ezért $t \leq 2$. Ha $\det(\mathbf{U}_2) \neq 0$, akkor $t = 2$. Megjegyezzük, hogy

$$\det(\mathbf{U}_2) = \det\begin{pmatrix} s_1 & s_2 \\ s_2 & s_3 \end{pmatrix} = s_1 s_3 - s_2^2.$$

   Ha $\det(\mathbf{U}_2) = 0$, akkor $\det(\mathbf{U}_1) \neq 0$, mivel feltételünk miatt t legfeljebb 2, de sem nem 0, sem nem 2, tehát $t = 1$. (Ha $\det(\mathbf{U}_1) = s_1 = 0$, akkor ez azt jelenti, hogy feltételünk nem teljesül, azaz 2-nél több hiba volt.)

$t = 1$ *eset:*

3. Oldjuk meg az

$$s_1 L_1 = -s_2$$

   egyenletet: $L_1 = -s_2 \cdot s_1^{-1}$

4. Keressük meg az $L(x) = 1 - s_2 \cdot s_1^{-1} x$ hibahelypolinom gyökének inverzét: $X_1 = s_2 \cdot s_1^{-1}$

5. Oldjuk meg az

$$Y_1 X_1 = s_1$$

   egyenletet: $Y_1 = s_1 X_1^{-1} = s_1^2 s_2^{-1}$.

6. Számítsuk ki a hibahelyet

$$i_1 = \log X_1$$

   és a hibaértéket

$$e_{i_1} = Y_1.$$

$t = 2$ *eset:*

3. Oldjuk meg az
$$\begin{pmatrix} s_1 & s_2 \\ s_2 & s_3 \end{pmatrix} \begin{pmatrix} L_2 \\ L_1 \end{pmatrix} = \begin{pmatrix} -s_3 \\ -s_4 \end{pmatrix}$$
egyenletet! Ezt megtehetjük például a Cramer-szabállyal:
$$L_2 = \det \begin{pmatrix} -s_3 & s_2 \\ -s_4 & s_3 \end{pmatrix} \cdot \det \begin{pmatrix} s_1 & s_2 \\ s_2 & s_3 \end{pmatrix}^{-1} = (s_2 s_4 - s_3^2)(s_1 s_3 - s_2^2)^{-1}$$
$$L_1 = \det \begin{pmatrix} s_1 & -s_3 \\ s_2 & -s_4 \end{pmatrix} \cdot \det \begin{pmatrix} s_1 & s_2 \\ s_2 & s_3 \end{pmatrix}^{-1} = (s_2 s_3 - s_1 s_4)(s_1 s_3 - s_2^2)^{-1}.$$

4. Keressük meg az $L(x) = 1 + L_1 x + L_2 x^2$ hibahelypolinom két gyökét, majd azok inverzeit: X_1, X_2.

5. Oldjuk meg az
$$Y_1 X_1 + Y_2 X_2 = s_1$$
$$Y_1 X_1^2 + Y_2 X_2^2 = s_2$$
egyenletet! (Itt ismét a Cramer-szabály egy lehetséges módszer.)

6. Számítsuk ki a hibahelyeket
$$i_1 = \log X_1, \qquad i_2 = \log X_2$$
és a hibaértékeket
$$e_{i_1} = Y_1, \qquad e_{i_2} = Y_2.$$

4.13. Reed–Solomon-kódok spektrális tulajdonságai

Legyen GF(q)=GF(p^m), ahol p prím. Bevezetve a $[\mathrm{GF}(q)]^n$ jelölést a GF(q) ($q = p^m$, p prím) feletti n hosszúságú vektorok halmazára, ahol $n \mid q-1$, definiáljuk a $[\mathrm{GF}(q)]^n$ halmaz önmagára történő leképezését az alábbi módon:

4.27. definíció. *A* $\mathbf{c} \in [\mathrm{GF}(q)]^n$ *vektornak egy* $\mathbf{C} \in [\mathrm{GF}(q)]^n$ *vektorra történő leképezése az alábbi:*

$$C_j = \sum_{i=0}^{n-1} \alpha^{ij} c_i, \qquad j = 0, 1, \ldots, n-1, \tag{4.23}$$

ahol α *a GF(q) egy n-edrendű eleme.*

A (4.23) transzformációt GF(q)-beli **Fourier-transzformáció**nak nevezzük, így a **c** időtartománybeli vektor frekvenciatartománybeli megfelelője (**spektrum**a) **C**. A (4.23) transzformáció rendelkezik a komplex számok feletti vektorokon képzett transzformáció tulajdonságaival. (Komplex vektorok esetén α megfelelője a komplex n-edik egységgyök.)

1. tulajdonság: A (4.23) leképezés kölcsönösen egyértelmű, azaz invertálható, és az inverz leképezés a következő:

$$c_i = f(n) \sum_{j=0}^{n-1} \alpha^{-ij} C_j, \qquad i = 0, 1, \ldots, n-1, \tag{4.24}$$

ahol $f(n) = (n \mod p)^{-1}$, és a -1-edik hatvány az ($n \mod p$) GF(p)-beli inverzét jelöli.

Bináris test esetén ($p = 2$) n páratlan, így ekkor $f(n) = 1$ eredményre jutunk. Ha α primitív elem, akkor $n = p^m - 1$, tehát $f(n) = (-1 \mod p)^{-1} = (p-1)^{-1} \mod p$. Mivel $n \mid p^m - 1$, ezért $p \nmid n$, azaz $n \neq 0 \mod p$, és így létezik multiplikatív inverze $\mod p$.

Mielőtt magát a tulajdonságot belátnánk, igazoljuk az alábbi lemmát.

4.9. lemma. *Ha α n-edrendű elem GF(q)-ban, akkor:*

$$\sum_{j=0}^{n-1} \alpha^{rj} = \begin{cases} 0, & \text{ha } r \neq 0 \mod n \\ n \mod p, & \text{ha } r = 0 \mod n \end{cases}. \tag{4.25}$$

BIZONYÍTÁS: Helyettesítsük az

$$x^n - 1 = (x-1)(x^{n-1} + x^{n-2} + \cdots + x + 1) \tag{4.26}$$

azonosság mindkét oldalába α^r-t. A bal oldalon mindig 0-t kapunk, mivel α rendje n. A jobb oldalon $r \neq 0 \mod n$ esetén $\alpha^r - 1 \neq 0$, így a (4.26) szorzat csak akkor lehet nulla, ha a második tényezője 0, azaz ha

$$\sum_{j=0}^{n-1} \alpha^{rj} = 0$$

míg $r = 0 \mod n$ esetén $\alpha^r = 1$, így az első tényező nulla, és ekkor a második tényező

$$\sum_{j=0}^{n-1} \alpha^{rj} = \sum_{j=0}^{n-1} 1 = n \mod p.$$

■

A 4.9. lemma felhasználásával most már egyszerűen belátható az 1. tulajdonság. Nevezetesen

$$\begin{aligned}\sum_{j=0}^{n-1}\alpha^{-ij}C_j &= \sum_{j=0}^{n-1}\alpha^{-ij}\sum_{k=0}^{n-1}\alpha^{kj}c_k = \\ &= \sum_{k=0}^{n-1}c_k\sum_{j=0}^{n-1}\alpha^{(k-i)j} = \\ &= f(n)^{-1}c_i. \end{aligned} \tag{4.27}$$

2. tulajdonság: Érvényes a **konvolúciós tétel** alábbi megfelelője: ha az $\mathbf{e},\mathbf{f},\mathbf{g}\in[\mathrm{GF}(q)]^n$ vektorokra

$$e_i = f_i g_i, \qquad i=0,1,\ldots,n-1 \tag{4.28}$$

fennáll, akkor a spektrumaikra

$$E_j = f(n)\sum_{k=0}^{n-1}F_{(j-k)\bmod n}G_k. \tag{4.29}$$

BIZONYÍTÁS: (4.28) figyelembevételével képezzük **e** transzformáltját:

$$\begin{aligned}E_j &= \sum_{i=0}^{n-1}\alpha^{ij}f_i g_i = \\ &= \sum_{i=0}^{n-1}\alpha^{ij}f_i f(n)\sum_{k=0}^{n-1}\alpha^{-ik}G_k = \\ &= f(n)\sum_{k=0}^{n-1}G_k\sum_{i=0}^{n-1}\alpha^{(j-k)i}f_i = \\ &= f(n)\sum_{k=0}^{n-1}G_k F_{(j-k)\bmod n}.\end{aligned}$$

Ezzel az állítást beláttuk. ■

A (4.29) szerinti konvolúciót **ciklikus konvolúció**nak nevezzük és $\mathbf{E}=\mathbf{F}*\mathbf{G}$ módon jelöljük. A fenti levezetést fordított sorrendben elvégezve láthatjuk, hogy a 2. tulajdonság megfordítva is igaz, azaz (4.29)-ből (4.28) is következik.

A továbbiakban az egyszerűség kedvéért $q=2^m$ elemszámú testet (azaz 2 karakterisztikájút) tételezünk fel, s ekkor $f(n)=1$. A $\mathbf{c}=(c_0,c_1,\ldots,c_{n-1})$ vektorhoz rendelt $c(x)=c_0+c_1x+\cdots+c_{n-1}x^{n-1}$ polinomhoz tartozó, **C** vektorral képzett $C(x)=C_0+C_1x+\cdots+C_{n-1}x^{n-1}$ polinomot **spektrumpolinom**nak nevezzük.

4.29. tétel. *A $c(x)$ polinomnak α^i akkor és csak akkor gyöke, ha $C_i = 0$ illetve a $C(x)$ polinomnak α^{-i} akkor és csak akkor gyöke, ha $c_i = 0$.*

BIZONYÍTÁS: A (4.23), (4.24) egyenletek felhasználásával

$$c(\alpha^i) = c_0 + c_1\alpha^i + \cdots + c_{n-1}\alpha^{(n-1)i} = C_i$$

$$C(\alpha^{-i}) = C_0 + C_1\alpha^{-i} + \cdots + C_{n-1}\alpha^{-(n-1)i} = c_i$$

ahonnan az állítás közvetlenül adódik. ■

Legyen az (n,k) paraméterű GF(q) feletti RS-kód generátorpolinomja $g(x)$, ahol $n \mid q-1$. Ekkor a $c(x)$ kódszó $c(x) = u(x)g(x)$ alakban állítható elő, ahol $\deg u(x) \leq k-1$.

A továbbiakban egy $r(x) = r_0 + r_1x + \cdots + r_jx^j$ j-edfokú polinomhoz rendeljünk egy n hosszú $\mathbf{r}$ vektort, amely legyen a következő: $\mathbf{r} = (r_0, r_1, \ldots, r_j, 0, \ldots, 0)$. Ezt és a polinomszorzás szabályát figyelembe véve egyszerűen belátható, hogy $\mathbf{c}, \mathbf{g}$ és $\mathbf{u}$ vektorokkal

$$c_i = \sum_{j=0}^{i} g_{i-j}u_j = \sum_{j=0}^{n-1} g_{(i-j) \bmod n}u_j \tag{4.30}$$

$i = 0, 1, \ldots, n-1$. (4.30)-ban felismerve a $\mathbf{c} = \mathbf{g} * \mathbf{u}$ konvolúciót, a konvolúciós tétel alapján a $\mathbf{C}, \mathbf{G}, \mathbf{U}$ spektrumokra a

$$C_j = G_j \cdot U_j \tag{4.31}$$

összefüggés érvényes.

Legyen $g(x) = (x-1)(x-\alpha)\cdots(x-\alpha^{n-k-1})$ az RS-kód generátorpolinomja, ahol α a GF(q) egy n-edrendű eleme. Ekkor $1, \alpha, \ldots, \alpha^{n-k-1}$ minden $c(x)$-nek gyöke, ezért tetszőleges $\mathbf{u}$ üzenethez tartozó $\mathbf{c}$ kódszó $\mathbf{C}$ spektrumában a 4.29. tétel felhasználásával

$$C_0 = C_1 = \cdots = C_{n-k-1} = 0 \tag{4.32}$$

zérus spektrumkomponensek adódnak. RS-kódot generálhatunk olyan ravasz módon, hogy eleve garantáljuk azt, hogy a kódszóspektrumok teljesítsék a (4.32) tulajdonságot, s a spektrum további komponenseit az üzenetektől függően különbözőkre választjuk. A tényleges időtartománybeli kódszavakat ezen mesterségesen összeállított spektrumok inverz Fourier-transzformáltjainak választjuk. Így, ha egy

$$\mathbf{C} = (\underbrace{0, 0, \ldots, 0}_{n-k}, u_0, u_1, \ldots, u_{k-1}) \tag{4.33}$$

4.10. ábra. Transzformációs kódolás.

vektort spektrumtartománybeli vektorként fogjuk fel, akkor az időtartománybeli **c** megfelelője egy kódszó. Ezen kódolást **transzformációs kódolás**nak nevezzük. A kódolás ekkor tehát a 4.10. ábrán látható blokkvázlat szerinti.

A Reed–Solomon-kódok dekódolásának több módszere ismeretes. A Peterson–Gorenstein–Zierler-algoritmus GF(q) feletti mátrixok invertálását igényli (lásd a 4.12. szakaszt). Ez az eljárás néhány hibát javító kód esetén számításigény szempontjából még elfogadható. A koncepció szempontjából jóval bonyolultabb Berlekamp–Massey-algoritmus (Ber–Mas) iteratív eljárást ad, ahol az iteráció lépései azonosak és egyszerűek. Növekvő hibajavítóképesség esetén egyre kedvezőbb a számításigény a mátrixinvertáláshoz képest. Az alábbiakban az ún. transzformációs-kódolási–dekódolási technikát mutatjuk be.

Tegyük fel, hogy a $\mathbf{v} = \mathbf{c} + \mathbf{e}$ vett szóban az $i_1, i_2, \ldots, i_t$ pozíciókban t számú hiba keletkezett, azaz a hibapolinom:

$$e(x) = e_{i_1} x^{i_1} + \cdots + e_{i_t} x^{i_t}. \tag{4.34}$$

Az (n,k) paraméterű RS-kód $2t \leq n-k$ teljesülése esetén képes a hibák kijavítására. A továbbiakban a t jelölést az aktuális, míg a $t_{\max}$ jelölést a maximálisan javítható hibák számára használjuk. Az egyes hibákat a helyük és értékük definiálja (azaz i_j és e_{i_j}). A dekódolás alábbiakban leírt módszerében először megkeressük a hibahelyeket — felépítve a hibahelypolinomot — majd ennek felhasználásával generáljuk a hibavektort, s ennek kivonásával elvégezzük a vett szó javítását.

Az

$$L(x) = \prod_{j=1}^{t} (1 - \alpha^{i_j} x) = 1 + L_1 x + \cdots + L_t x^t \tag{4.35}$$

hibahelypolinom együtthatóit a rájuk mint ismeretlenekre vonatkozó megfelelő számú lineáris egyenletből álló egyenletrendszer megoldásaként kapjuk. Ezen egyenletrendszer levezetését ismertetjük először.

1. lépés: Szindrómaszámítás

A $\mathbf{v} = \mathbf{c} + \mathbf{e}$ vektor Fourier-transzformáltját képezve a

$$\mathbf{V} = \mathbf{C} + \mathbf{E} \tag{4.36}$$

spektrumot kapjuk, ahonnan (4.32) felhasználásával

$$V_j = E_j, \qquad j = 0,1,\ldots,n-k-1 \tag{4.37}$$

adódik, azaz a **V** spektrum első $n-k$ komponense szindróma.

2. lépés: A hibahelypolinom kiszámítása

Az $L(x)$ hibahelypolinom definíciójából következően $L(\alpha^{-i_j}) = 0$, $j = 1,2,\ldots,t$. Innen következik, hogy az $\mathbf{L} = (1, L_1, L_2, \ldots, L_t, 0, \ldots, 0)$ n-hosszra kiegészített vektor **l** inverz Fourier-transzformáltjában a hibahelyeknek megfelelő pozíciójú elemek is nullelemek. Ezen jelölésekkel az

$$l_i = 0 \longleftrightarrow e_i \neq 0 \tag{4.38}$$

kölcsönösen egyértelmű megfeleltetést kapjuk. Innen következik, hogy

$$l_i \cdot e_i = 0, \qquad i = 0,1,\ldots,n-1, \tag{4.39}$$

amit a konvolúciós tétel felhasználásával az

$$\mathbf{L} * \mathbf{E} = 0 \tag{4.40}$$

ekvivalens alakba írhatunk. A (4.40) egyenlet ekvivalens a

$$\sum_{j=0}^{t} L_j E_{(i-j) \bmod n} = 0, \qquad i = 0,1,\ldots,n-1 \tag{4.41}$$

egyenletrendszerrel. Mivel maximálisan $t_{\max}$ hiba javítására készülünk fel, azaz amikor $t \leq t_{\max}$, ezért helyettesítsük a t változót a $t_{\max}$ fix értékkel a (4.41) egyenletrendszerbe, $L_{t+1} = \cdots = L_{t_{\max}} = 0$ definíció mellett, majd a (4.41) egyenletrendszerből az alábbi $t_{\max}$ egyenletet emeljük ki

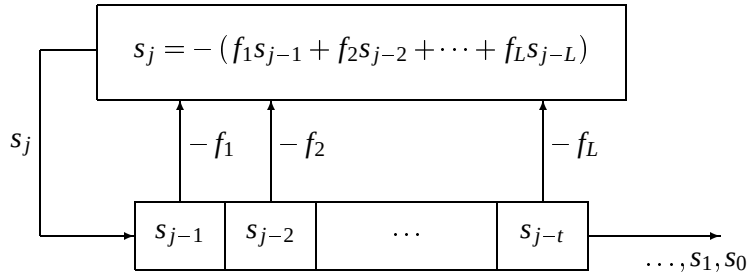
$$\sum_{j=0}^{t_{\max}} L_j E_{i-j} = 0, \qquad i = t, t+1, \ldots, 2t-1. \tag{4.42}$$

A feladat ezek után a (4.42) egyenletrendszer megoldása azzal a megkötéssel, hogy az $L(x)$ megoldás minimális fokszámú legyen.

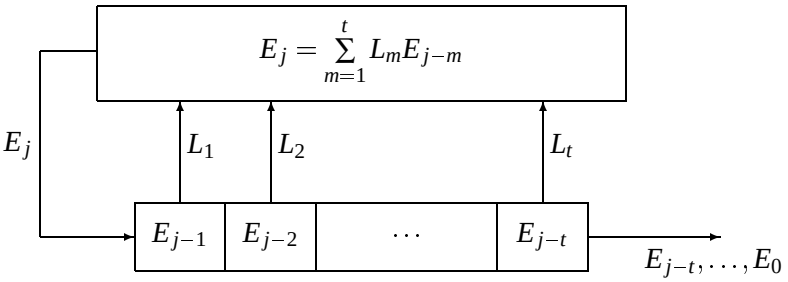
A (4.42) alakú egyenletrendszerből kiindulva szokásos egyenletrendszer megoldással (mátrixinvertálással) dolgozik a Peterson–Gorenstein–Zierler- (PGZ) dekóder.

A feladat azonban más úton is megoldható. A (4.41) egyenletrendszer ekvivalens módon megadható egy lineárisan visszacsatolt shiftregiszteres leírással.

Ehhez felelevenítjük a 4.8. szakasz végén bevezetett a lineárisan visszacsatolt shiftregisztereket (LFSR) a 4.11. ábra segítségével.



4.11. ábra. LFSR.



4.12. ábra. A szindrómákat generáló LFSR.

Az ottani jelöléseknek megfelelően $\langle f(x), L\rangle$ jelöli az LFSR-t. A mostani feladatunkban $\langle L(x), t\rangle$ LFSR generálja a szindrómák $E_0, E_1, E_2, \ldots, E_{n-1}, E_0, E_1, \ldots$ periodikus sorozatát, ha az LFSR-t az $E_0, E_1, \ldots, E_{t-1}$ kezdeti értékből indítjuk. Ez a generálás tehát a (4.41) egyenletrendszernek az

$$E_j = -(L_1 E_{j-1} + L_2 E_{j-2} + \cdots + L_t E_{j-t}) \tag{4.43}$$

rekurzióba történő átírását jelenti, ahol j tetszőleges egész szám és $E_j = E_{j \bmod n}$. A (4.43) lineáris rekurziónak megfelelő LFSR generálást szemléltetjük a 4.12. ábrán.

Az $\langle L(x), t\rangle$ LFSR egyben egyértelmű, minimális regiszterhosszú az összes olyan LFSR-ek közül, amelyek generálják a periodikusan kiterjesztett szindrómasorozatot. Ez abból látható be, hogy a (4.43) rekurzió alapján felírható

$$\begin{pmatrix} E_0 & E_1 & \cdots & E_{t-1} \\ E_1 & E_2 & \cdots & E_t \\ \vdots & & \ddots & \vdots \\ E_{t-1} & E_t & \cdots & E_{2t-2} \end{pmatrix} \begin{pmatrix} L_t \\ L_{t-1} \\ \vdots \\ L_1 \end{pmatrix} = \begin{pmatrix} -E_t \\ -E_{t+1} \\ \vdots \\ -E_{2t-1} \end{pmatrix} \tag{4.44}$$

egyenletrendszerben a bal oldali, szindrómákat tartalmazó mátrix invertálható, s az egyenletrendszernek a tényleges hibavektorhoz tartozó hibahelypolinom egy megoldása. Azon egyenletrendszereknek, amelyeket úgy kapunk, hogy a (4.44) egyenletrendszerben a szindrómák indexét $j = 0, 1, \ldots, 2(t_{\max} - t)$ értékkel növeljük, szintén egyértelmű, minimális megoldása a tényleges hibahelypolinom, ha $t \leq t_{\max}$ fennáll.

A dekódolás során csak az $E_i (= V_i)$, $i = 0, 1, \ldots, 2t_{\max} - 1$ szindrómák adottak. A shiftregiszter-szintézis feladat azon minimális t' fokszámú $\langle L'(x), t' \rangle$ LFSR meghatározása, amely az $E_0, E_1, \ldots, E_{t'-1}$ kezdőállapotból elindítva generálja a további $E_{t'}, E_{t'+1}, \ldots, E_{2t_{\max}-1}$ elemeket is.

A fentiekben láttuk, hogy amennyiben $t \leq t_{\max}$, akkor a szintézis az $L'(x) = L(x)$, $t' = t$ megoldásra vezet. A minimális fokszámra törekvés megfelel a PGZ-dekódolásnál a szindrómákból alkotott $t_{\max} \times t_{\max}$ méretű mátrix legnagyobb invertálható főminorja megkeresésének. A Berlekamp–Massey shiftregiszter szintézis eljárásnál a mátrixinvertálás helyett egy jóval kisebb számításigényű, iteratív algoritmussal állítjuk elő a hibahelypolinomot.

3. lépés: A rekurzív kiterjesztés

Az $E_{2t_{\max}}, \ldots, E_{n-1}$ komponenseket a (4.30) alapján az

$$E_i = \sum_{k=1}^{t'} L'_j E_{i-j}, \qquad i = 2t_{\max}, \ldots, n-1 \tag{4.45}$$

rekurzív kiterjesztéssel állítjuk elő a 2. lépésben meghatározott $L'(x)$ valamint $E_0, \ldots, E_{2t_{\max}-1}$ felhasználásával.

Ezután a

$$\mathbf{C} = \mathbf{V} - \mathbf{E} \tag{4.46}$$

kivonással kapjuk meg **C**-t, amelynek szegmense az **u** üzenet.

4.14. A konvolúciós kódolás alapfogalmai

A blokk- illetve a konvolúciós kódok a hibajavító kódok két nagy osztályát alkotják. Az egyszerűbb matematikai leírhatóság miatt a bevezető jellegű tankönyvek elsősorban a blokk-kódokat részletezik, főleg azért, mert azokon jól bemutathatók a hibajavítás alapfogalmai és lépései, másrészt pedig azért, mert látványos algebrai struktúrák (pl. Galois-testek elméletének alapelemei) és optimális csatornakód-kontrukciók léteznek a blokk-kódok körében.

A konvolúciós kódok leírása nem ilyen egyszerű, és a kódoptimalizálás is elsősorban számítógépes keresésen, s nem konstrukciós tételeken alapszik. A praktikus paramétertartományok esetére ismeretes a jó kódok listája.

A blokk-kódok algebrai dekódolási algoritmusai olyan vett blokkokat képesek csak feldolgozni, amelyek elemei ugyanazon halmazból valók (tipikusan GF(2)- vagy GF(q)-beliek), mint amelyből a kódszavak veszik elemeiket, legfeljebb a törlés szimbólum engedett meg ezen felül a vett blokkokban. A konvolúciós kódok esetén a csatorna kimeneti ábécé ennél lényegesen bővebb halmaz lehet. A konvolúciós kódok ún. soft-dekóderei (pl. Viterbi-dekódolás, szekvenciális dekódolás) megengedik, hogy a csatorna kimeneti ábécé elemei tetszőleges valós számok (mérési minták) legyenek.

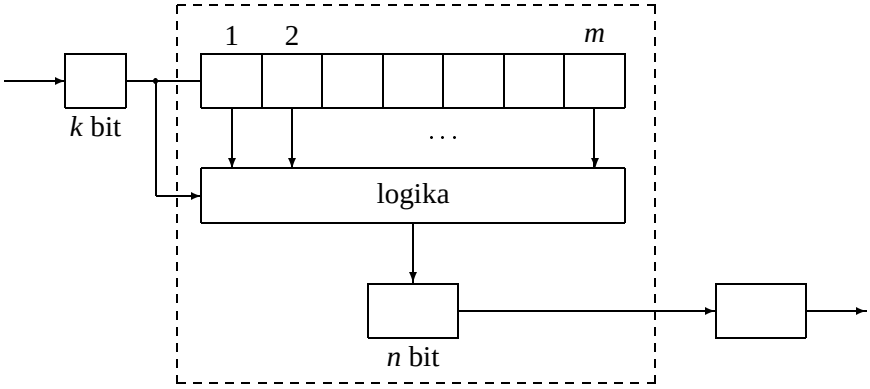
Létezik optimális (maximum-likelihood, ML), implementálható (számításigényében gazdaságos) dekódolási algoritmus a konvolúciós kódokra, az ún. Viterbi-dekódolás. Természetesen az ML dekódolás elvileg a blokk-kódok esetén is elvégezhető, de a blokk-kódok esetén az exponenciális (adott kódoló mellett egy rövidített kód üzenethosszával exponenciálisan növekvő) bonyolultságú kimerítő keresésnél kisebb komplexitású ML dekóder nem ismeretes még speciális kódosztályokra sem. Ezzel szemben a Viterbi-dekódolás komplexitása adott kódoló mellett az üzenethosszal közel lineárisan növekvő.

A konvolúciós kódok alkalmazásával lehetővé válik a moduláció és a csatornakódolás illetve a demoduláció és a csatornadekódolás együttes tervezése. Így a konvolúciós kód struktúrák napjainkban nemcsak a hibajavító kódolásban, hanem például korszerű modulációs–demodulációs eljárásokban, diszperzív csatornán történő megbízható adatátvitelben (csatornakiegyenlítés) is felhasználásra kerülnek.

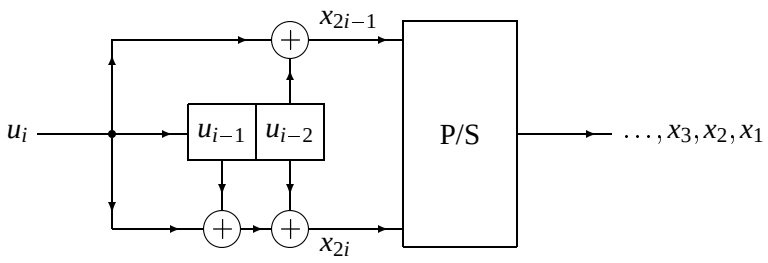
A léptetőregiszteres kódoló

A forrás bitfolyamát k bites szegmensekre, ún. **üzenetkeret**ekre bontjuk. A kódoló m üzenetkeretet tárol léptetőregiszterében, azaz egy időegység alatt egy új üzenetkeretet léptetünk a regiszterbe, a legrégebbi tárolt keret kilép abból, s azt eldobjuk. Az időegység kezdetén a bemenetre érkezett új keret valamint a tárolt m keret alapján a kódoló kiszámít egy **kódszókeret**et, amely n bit hosszúságú. A kódszókeretet kiléptetjük a kódolóból (4.13. ábra).

Az $R = \frac{k}{n}$ arányt **kódsebesség**nek nevezzük. A bitekben kifejezett regiszterhossz + 1 ($= (m+1)k$) a léptetőregiszteres kódoló **kényszerhossz**a. Egy, a kódolóhoz érkező üzenetbit maximum $(m+1)k$ forrásbitnek megfelelő hosszon befolyásolhatja a kódoló outputját. Az $(m+1)n$ mennyiség neve **blokkhossz**, mely megmutatja, hogy egy forrásbit maximálisan hány csatornabitnek megfelelő hosszon befolyásolja a kódoló outputját. Az ismertetett konstrukció eredménye egy (n,k) **fa-kód**; a léptetőregiszteres kódoló bemenetére vezetve az összes lehetséges félig végtelen hosszú input bitsorozatot, a kimenetén megjelenik az összes



4.13. ábra. Léptetőregiszteres kódoló.



4.14. ábra. Konvolúciós kódoló.

különböző kódszó, amelyek úgyszintén félig végtelen sorozatok. Ha a fa-kód egyben véges kényszerhosszú is, azt **trellis-kód**nak nevezzük. Egy kód időinvariáns, amennyiben ha két input bitsorozat között csak T üzenetkeret időeltolás különbség van, akkor a nekik megfelelő output bitsorozatok (kódszavak) között T kódszókeret különbség van. Az időinvariáns trellis-kód a csúszó blokk-kód.

4.28. definíció. *Egy (n,k) fa-kódot, ha lineáris, időinvariáns és véges kényszerhosszú, (N,K)* ***konvolúciós kód****nak hívunk, ahol $K=(m+1)k$, $N=(m+1)n$.*

A 4.14. ábra konvolúciós kódolója esetén $m=2, k=1, n=2, R=\frac{1}{2}$, s a két lineáris kombináció:

$$\begin{aligned} x_{2i-1} &= u_i \oplus u_{i-2} \\ x_{2i} &= u_i \oplus u_{i-1} \oplus u_{i-2} \end{aligned}$$

$i = 1, 2, \ldots$. Az adott példa esetére elmondva, a konvolúciós kódolás elnevezés onnan származik, hogy az $(u_1, u_2, \ldots)$ üzenetbitsorozatot az $(1, 0, 1, 0, 0, 0, \ldots)$ illetve az $(1, 1, 1, 0, 0, 0, \ldots)$ sorozatokkal történő mod 2 konvolúció adja a páratlan illetve a páros pozíciójú csatornabiteket a kódoló kimenetén.

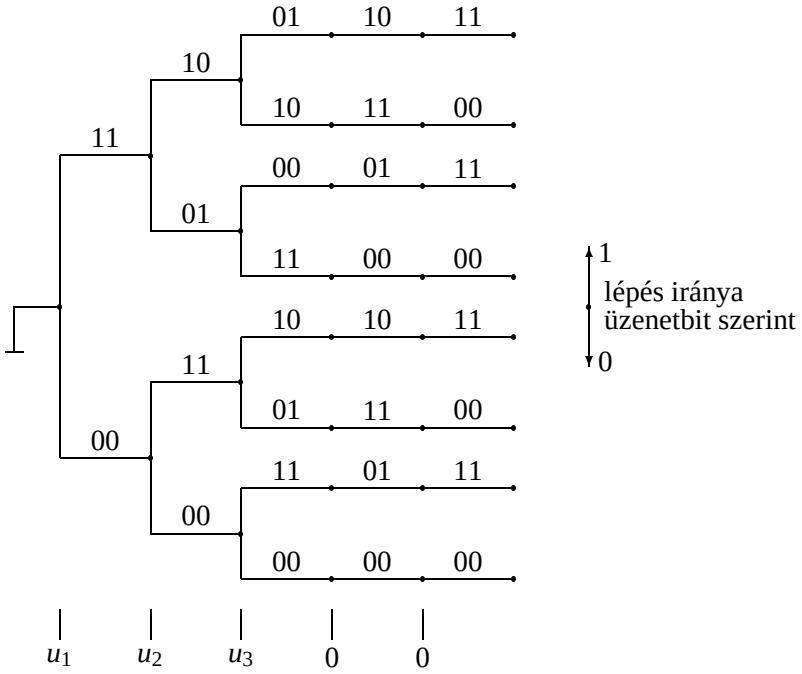
A konvolúciós kódolóval blokk-kódolást is végezhetünk az alábbi módon. Tegyük fel, hogy az $\mathbf{u} = (u_1, u_2, \ldots, u_L)$ üzenetet szeretnénk kódolni egy kódszóba. Egészítsük ki ezen üzenetet $m \cdot k$ zéró bittel, azaz képezzük az $\mathbf{u}' = (u_1, u_2, \ldots, u_L, 0, \ldots, 0)$ üzenetet. Ha zéró állapotból indítjuk a kódolót, akkor $\mathbf{u}'$ üzenetenként a zéró állapotba tér vissza (térítjük vissza), azaz L bites üzenetblokkokat blokk-kódolunk. Az R' névleges sebességű konvolúciós kódoló esetén ekkor

$$R = \frac{L}{L + m \cdot k} R' \tag{4.47}$$

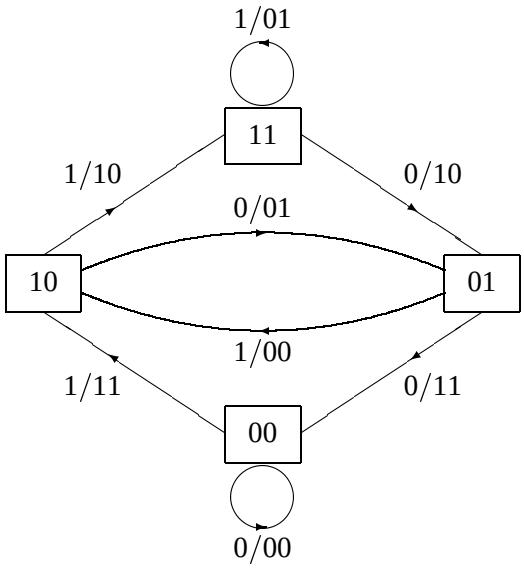
a valós kódolási sebesség. A példabeli kódolás esetét tekintve pl. egy $L = 3$ bites üzenetblokkot $2 \cdot (3 + 2) = 10$ bites kódszóba kódolunk.

A konvolúciós kódok ábrázolásának egyik módja a **bináris fa reprezentáció**. Példabeli kódolónk esetére láthatjuk ezt a reprezentációt a 4.15. ábrán.

A fa csomópontjaiból két irányba léphetünk a kódolandó üzenetbitnek meg-



4.15. ábra. A kód bináris fa reprezentációja.



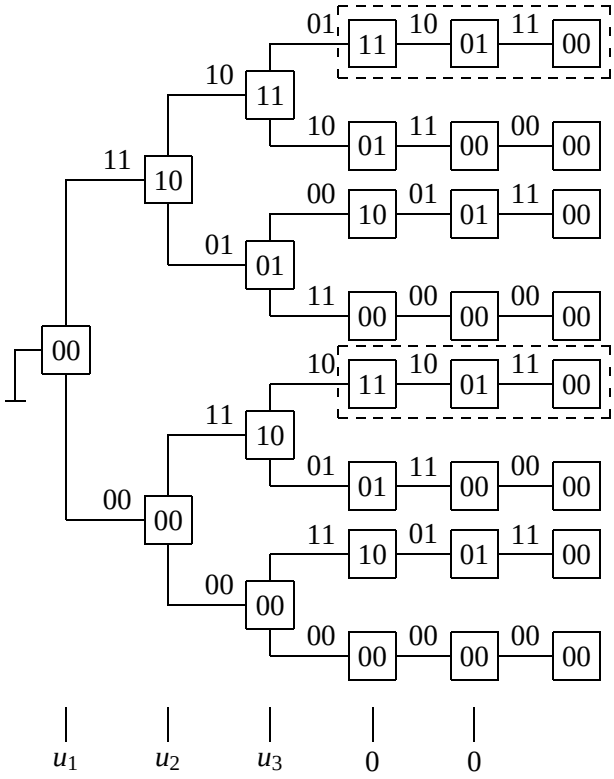
4.16. ábra. A kódoló állapotátmenet-gráfja.

felelően. A fa éleit azon bitpárral (általában bit *n*-essel) címkéztük fel, amely a kódoló kimenetén megjelenik az aktuális üzenetbit belépése hatására. A gyökértől a fa élei mentén a fa leveleiig vezető utak egy-egy kódszónak felelnek meg. Nyilván minden konvolúciós kódolónak megfeleltethető egy fenti bináris fa. A dekódolás szempontjából azonban szerencsésebb az ún. **trellis reprezentáció**.

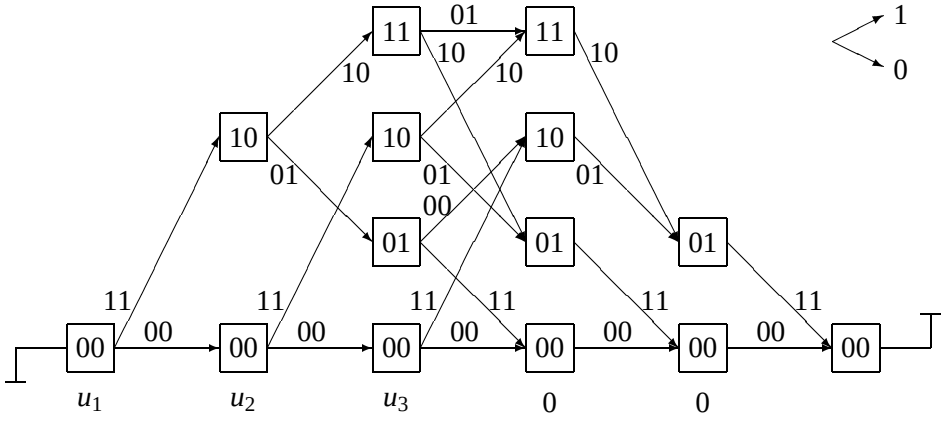
A trellis ábrázolást megkönnyíti, ha elkészítjük a **kódoló állapotátmenet-gráf**ját, amelyet a példánk esetére a 4.16. ábrán láthatunk.

Mindegyik állapotból két irányított él vezet egy másik állapotba. Az élen elhelyezett i/jk jelölésben i az üzenetbit, jk pedig az ezen állapotátmenet során a kimeneten megjelenő bitpár. Így például az 10 állapotból az 1 üzenetbit belépésekor az 11 állapotba megy át a kódoló, miközben a kimeneten az 10 bitpár jelenik meg. Ezek után tekintsük újra a bináris fa reprezentációt, s bővítsük ki ezt az ábrázolást úgy, hogy a csomópontokat állapotoknak feleltetjük meg, ahogy ezt a 4.17. ábrán láthatjuk a példa esetére.

A 4.17. ábrát alaposabban szemügyre véve láthatjuk, hogy ha a fa azonos mélységében az azonos állapotoknak megfelelő csomópontokat egy állapot-csomópontnak tekintjük, akkor láthatjuk, hogy csak négyféle és nem nyolcféle eset áll elő. Például a 4.17. ábrabeli fának a szaggatott vonallal körülhatárolt két része összevonható. Ha összevonjuk ennek megfelelően az azonos állapot-csomópontokat egy csomópontba, akkor kapjuk a trellis reprezentációt, amit a példa esetére a 4.18. ábrán láthatunk.



4.17. ábra. A kibővített fa ábrázolás.



4.18. ábra. A kód trellis ábrázolása.

A trellisben a csomópontokat összekötő élek címkéje az állapotváltozás során keletkező kimenet. Az egymást követő élek utat alkotnak, amelyek mindegyike az azonos kezdő (00 állapot) csomópontból indul, s azonos (00 állapot) végcsomópontba fut be. Az utak egy-egy kódszónak felelnek meg. A kezdő csomóponttól i él távolságra levő csomópontokról azt mondjuk, hogy i mélységben vannak. Így pl. egy mélységben kettő, kettő mélységben négy csomópontot tartalmaz a trellis.

Azon kódokat, amelyek egy trellis ábrázolással leírhatók, trellis kódoknak nevezzük (ekkor nem kell felcímkézve lenniük a csomópontoknak). A konvolúciós kódolóval tehát olyan speciális blokk-kódolást végezhetünk, amely blokk-kód bináris fa-kód, továbbá olyan speciális fa-kód, amely trellis-kód is egyben.

Polinomok alkalmazása a konvolúciós kódok esetén is segíti a tömör leírást. Egy konvolúciós kódoló léptetőregiszterének n számú lineáris előrecsatolását tartalmazza. Az egyes előrecsatolásokat a megcsapolási pozícióknak megfelelően polinomokkal írhatjuk le. A 4.14. ábra szerinti kódoló esetén ez a $g_{11}(x) = x^2+1$, $g_{12}(x) = x^2+x+1$ bináris polinomokat jelenti. (Szokásos még az ún. oktális megadási mód, amelynél a polinom együtthatók hármas csoportjait oktális számokba képezzük. Például a fenti kódoló esetén $[5,7]$ a kódoló oktális megadása.) Általános konvolúciós kód esetén $g_{ij}(x)$ az üzenetkeret i-edik bitje és a kódszókeret j-edik bitje közötti kapcsolatot írja le.

A $g_{ij}(x)$ generátorpolinomokat mátrixba rendezve kapjuk a

$$\mathbf{G}(x) = [g_{ij}(x)]$$

$k \times n$ méretű generátorpolinom-mátrixot.

Ha $d_i(x)$ jelöli az egymás utáni üzenetkeretek i-edik bitje félig végtelen sorozatának megfelelő polinomot, s hasonlóan $c_j(x)$ a kódszókeret j-edik bitjei félig végtelen sorozatának megfelelő polinomot, akkor a

$$\mathbf{d}(x) = [d_1(x), d_2(x), \ldots, d_k(x)]$$

$$\mathbf{c}(x) = [c_1(x), c_2(x), \ldots, c_n(x)]$$

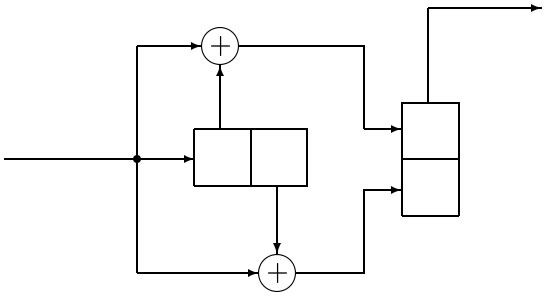
jelölésekkel a

$$\mathbf{c}(x) = \mathbf{d}(x)\mathbf{G}(x)$$

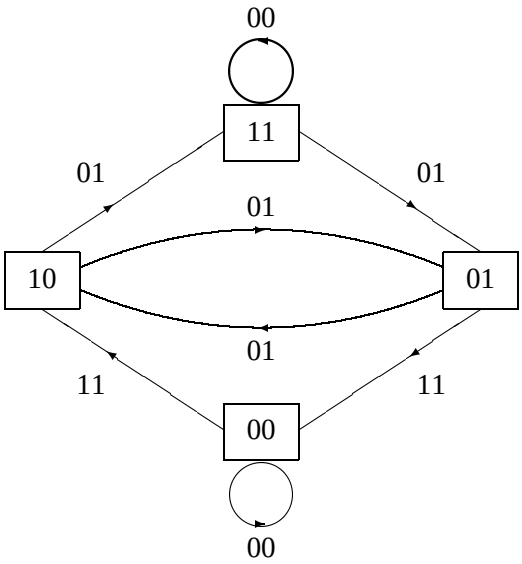
generálási szabály adódik.

4.29. definíció. *Egy konvolúciós kód* ***katasztrofális****, ha tetszőlegesen nagy Hamming-súlyú input sorozat esetén korlátos Hamming-súlyú marad az output.*

Ezen definíció alapján könnyen belátható az alábbi tétel.



4.19. ábra. Katasztrofális kódoló ($g_{11}(x) = x+1$ és $g_{12}(x) = x^2+1$).



4.20. ábra. A katasztrofális kódoló állapotátmenet-gráfja.

4.30. tétel. *Egy kód katasztrofális tulajdonságának szükséges és elégséges feltétele az, hogy az állapotgráfjában létezzen egy hurok, amelyet alkotó valamennyi élen a kódszókeretek zérus súlyúak.*

A 4.19. ábrán egy katasztrofális kódolót látunk. Ennek állapotátmenet-gráfja a 4.20. ábrán látható, ahol vastag vonallal jelöltük a zéró súlyú hurkot.

$k = 1$ paraméterű kódok esetén az alábbi tétel egy még praktikusabb ellenőrzésre ad lehetőséget:

4.31. tétel. *$\frac{1}{n}$ sebességű kód esetén a nem katasztrofális tulajdonság fennállásának szükséges és elégséges feltétele, hogy*

$$\text{lnko}[g_{11}(x), g_{12}(x), \ldots, g_{1n}(x)] = 1$$

legyen.

MEGJEGYZÉS: „Jó” konvolúciós kód konstruálása tehát azonos feladat relatív prím polinomok egy „jó” halmazának konstruálásával. Sajnos erre a feladatra nem állnak jelenleg még rendelkezésre olyan általános konstrukciók, mint a blokkkódok esetén.

BIZONYÍTÁS: Tekintsünk egy $\frac{1}{n}$ sebességű ($k = 1$) nemkatasztrofális kódot. Tetszőleges kódtól megköveteljük, hogy létezzen a kódolás inverze. Az euklidészi algoritmus alapján léteznek (és elő is állíthatók) azok az $a_1(x), a_2(x), \ldots, a_n(x)$ polinomok, amelyekkel

$$a_1(x)g_{11}(x) + a_2(x)g_{12}(x) + \cdots + a_n(x)g_{1n}(x) = 1$$

adódik. Ha $d(x)$ az adatpolinom (mivel $k = 1$, ezért egy polinomról van szó), akkor

$$c_j(x) = d(x)g_{1j}(x), \qquad j = 1, 2, \ldots, n$$

a kódszókeretek j-edik bitjét leíró polinom, tehát a $d(x)$ adat könnyen ellenőrizhetően a

$$d(x) = a_1(x)c_1(x) + a_2(x)c_2(x) + \cdots + a_n(x)c_n(x)$$

művelettel nyerhető vissza.

A kódoló és inverze a 4.21. ábrán látható. Véges súlyú $c_j(x)$ polinomok véges súlyú $a_j(x)$ polinommal való szorzata is véges súlyú. ■

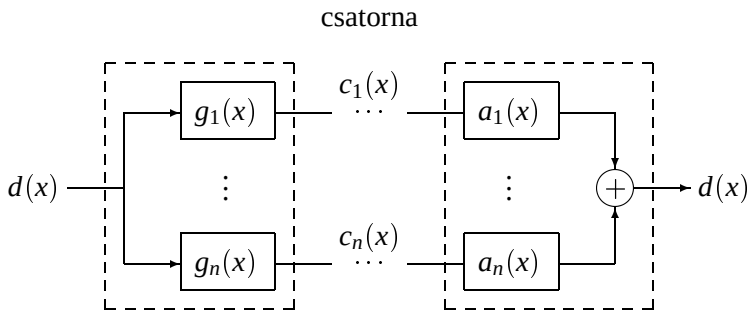
Az r-edik minimális távolság egy konvolúciós kód esetén a legkisebb Hamming-távolság a kódoló output sorozatok első r kódszókeret hosszú (rn bit) szegmense között. Jelölése: d_r^*. Nyilván

$$d_1^* \leq d_2^* \leq d_3^* \leq \cdots$$

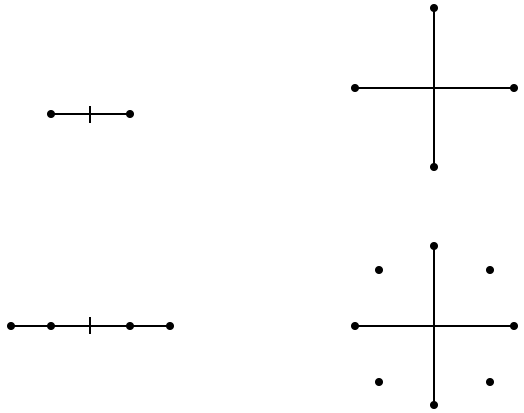
A $d_1^*, d_2^*, d_3^*, \ldots$ sorozat a konvolúciós kód **távolságprofil**ja.

$$d_\infty = \max_r d_r^*$$

a **szabad távolság**.



4.21. ábra. Kódoló és inverze.



4.22. ábra. Jelkészlet példák: 2,4 szintű AM, 4,8 szintű PSK.

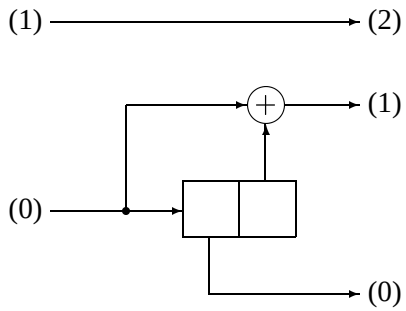
Konvolúciós kódok komplex ábécé felett (Ungerboeck-kódok)

Az (n,k) konvolúciós kódoló a k bites üzenetkeret inputokat egy-egy kódolási lépésben 2^n méretű komplex számhalmaz egy elemére képezi le. A komplex számhalmazt jelkészletnek hívjuk. Tipikus példáit láthatjuk a 4.22. ábrán.

Két kódszó távolságát euklidészi távolságban mérjük:

$$d_E(\mathbf{c}_1,\mathbf{c}_2)^2 = \sum_{i=0}^{\infty} |c_{1i} - c_{2i}|^2$$

ahol $\mathbf{c}_1$ és $\mathbf{c}_2$ komplex elemű konvolúciós kódszavak. Ennek megfelelően euklidészi távolságokban adódnak a kódtávolság jellemzők. A kódoló gaussi csatornán történő alkalmazásakor ez a kódteljesítmény legalkalmasabb mértéke. Ugyanis



4.23. ábra. Bináris $(3,2)$ konvolúciós kódoló.

annak P_e valószínűsége, hogy hibás kódszóra döntünk

$$P_e \geq N_d \Phi\left(-\frac{d}{2\sigma}\right),$$

ahol Φ a standard normális eloszlásfüggvény, σ a gaussi zajminta szórása, N_d pedig a zérus kódszótól d minimális euklidészi távolságra lévő kódszavak száma.

A **kódolási nyereség**et

$$G = 20\log_{10}\left(\frac{d_\infty}{d_{\text{ref}}}\right) \ [\text{dB}]$$

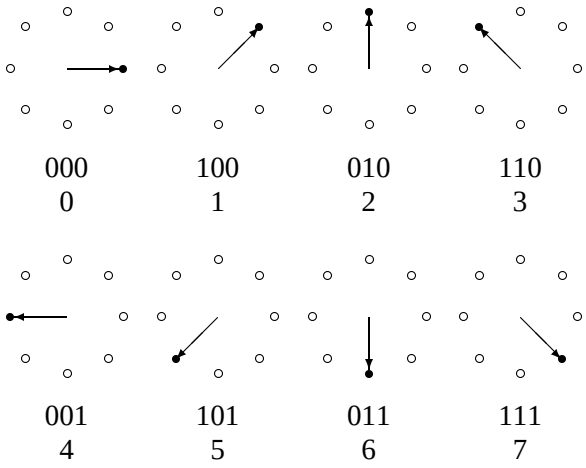
definiálja, ahol a d_{ref} a kódolatlan esetben adódó minimális euklidészi távolság.

Egy példát láthatunk a 4.23. ábrán. A $(3,2)$ konvolúciós kódoló egy lépésben a bemenetére érkező bitpárt output bithármasba, majd azt a 8PSK jelkészlet egy elemébe képezi.

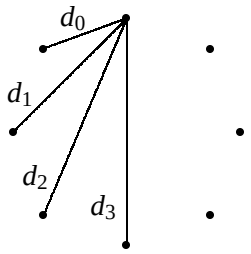
$$\mathbf{G} = \begin{pmatrix} x & x^2+1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

A 3 bites output 8PSK jelkészletbe (45°-os szöggel forgatott komplex egységvektorok) történő leképezése a 4.24. ábra szerinti (a bithármasok bitsorrendje sorban a $0,1,2$ outputpozícióknak felel meg). A jelkészlet elemei közötti euklidészi távolságokat a 4.25. ábrán láthatjuk, ahol $d_0 = \sqrt{2-\sqrt{2}}$, $d_1 = \sqrt{2}$, $d_2 = \sqrt{2+\sqrt{2}}$, $d_3 = 2$.

A kódoló trellisének egy részlete látható a 4.26. ábrán, ahol az ágakon a 8PSK jelek bináris indexét, illetve az input bitpárt tüntettük fel. Egy csomópontból két él vezet minden utána következő csomópontba. A vastag vonallal kiemelt hurok



4.24. ábra. A 3 bites output 8PSK jelkészletbe történő leképezése.



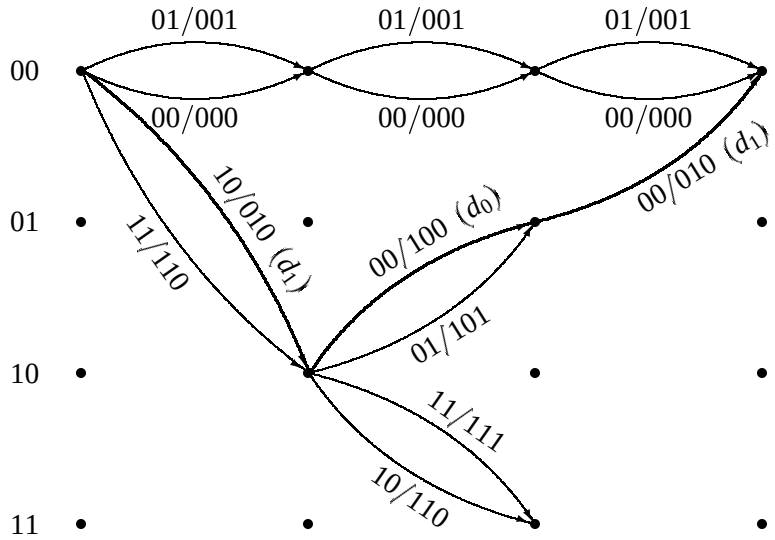
4.25. ábra. A jelkészlet elemei közötti euklidészi távolságok.

a csupa zérus kódszótól minimális euklidészi távolságra eltérő kódszónak felel meg.

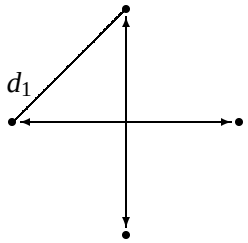
Egyszerű vizsgálattal megállapítható, hogy ezen hurok a 01 00 00 input bitpárok (üzenetkeretek) hatására jön létre, s a kódszó Hamming-súlya 3. A hurokra írt 8PSK jelek távolságát a 0 jeltől képezve kapjuk, hogy az euklidészi metrikában mért kódtávolság

$$d_\infty = \sqrt{d_1^2 + d_0^2 + d_1^2} = \sqrt{2 + 2 - \sqrt{2} + 2} = \sqrt{6 - \sqrt{2}} = 2.141.$$

Ahhoz, hogy a d_{ref} referencia euklidészi távolságot megállapíthassuk, tekintsük a kódolatlan esetet. Ha nincs kódolás, akkor a 2 bites üzenetkeretet közvetlenül 4PSK jelkészlet elemeibe transzformáljuk, amely jelkészletet a 4.27. ábrán láthatjuk. Az ábra alapján nyilvánvaló, hogy ekkor a minimális euklidészi távolság a 00



4.26. ábra. A kódoló trellis reprezentációja.



4.27. ábra. 4PSK.

00 00 ... zérus sorozathoz képest a 01 00 00 ... sorozatra adódik, azaz

$$d_{\text{ref}} = d_1 = \sqrt{2}.$$

Tehát a kódolási nyereség $G = 20\log_{10}\left(\frac{2.141}{\sqrt{2}}\right) = 3.6$ dB. Az egyszerű kódot tekintve ez igen jó eredmény. Az Ungerboeck-kódok jelentőségét még könnyebben láthatjuk, ha észrevesszük, hogy a kódolatlan 4PSK adó-vevőbe a 4PSK modulátor helyébe mindennemű konverzió nélkül beillesztve az Ungerboeck komplex kódoló illetve dekódoló blokkot, minden egyéb fizikai feltétel (azonos forrássebesség, adási teljesítmény, sávszélesség) változatlanul maradása mellett figyelemre-

méltó kódolási nyereséget érhetünk el, illetve a kódolási nyereséget kisebb jel/zaj igénybe konvertálhatjuk, azaz kisebb adási teljesítmény igényünk lesz.

A CCITT V.32bis modemszabványa is Ungerboeck-kódot használ, s ezzel az átviteli sebességet 14.4 kbps-ra növeli a korábbi technikához képest. (Pl. az 1964-ben megjelent V.21-es szabvány csak 200 bps-ot tudott biztosítani.)

4.15. A konvolúciós kódok Viterbi-dekódolása

A Viterbi-dekódolás a trellis kódok **maximum-likelihood** (ML) **dekódolás**ára optimalizált algoritmus.

A dekódolás algoritmusának egyszerűbb szemléltetése kedvéért tekintsünk egy p paraméterű bináris szimmetrikus emlékezetnélküli csatornát (BSC), ahol $0 < p < \frac{1}{2}$. Ezen csatornára $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ input vektor és $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ output vektor jelölés esetén

$$\mathbf{P}\{\mathbf{y} \mid \mathbf{x}\} = (1-p)^N \left(\frac{p}{1-p}\right)^{d(\mathbf{x},\mathbf{y})} \tag{4.48}$$

ahol $d(\mathbf{x},\mathbf{y})$ a vektorok Hamming-távolsága. Mivel $0 < p < \frac{1}{2}$ ekvivalens a $0 < \frac{p}{1-p} < 1$ állítással, ezért $\mathbf{y}$ vétele esetén az ML dekódolás ekvivalens a minimális Hamming-távolságban levő kódszó (input vektor) megkeresésével, azaz

$$\max_{\mathbf{x}_i} \mathbf{P}\{\mathbf{y} \mid \mathbf{x}_i\} \iff \min_{\mathbf{x}_i} d(\mathbf{y}, \mathbf{x}_i). \tag{4.49}$$

A (4.49)-ben jelzett művelet távolságmérést és minimumkeresést jelent, ahol L az üzenethossz. Ennél azonban sokkal kisebb számításigénnyel is megoldható a feladat. Tudjuk, hogy a kód trellisének útjai az egyes kódszavaknak, s az ezen utakat alkotó élek a kódszavak n bites szegmenseinek felelnek meg. Ennek megfelelően, ha a trellis éleit súlyozzuk az $\mathbf{y}$ és $\mathbf{x}_i$ megfelelő n bites szegmensének a Hamming-távolságával, akkor az $\mathbf{x}_i$-nek megfelelő út összsúlya megegyezik a $d(\mathbf{y}, \mathbf{x}_i)$ Hamming-távolsággal.

Ekkor tehát dekódolási metrikaként a Hamming-távolságot használjuk. Általában diszkrét emlékezetnélküli csatorna (DMC) esetén

$$\log \mathbf{P}\{\mathbf{y} \mid \mathbf{x}\} = \sum_{j=1}^{N} \log \mathbf{P}\{y_j \mid x_j\}, \tag{4.50}$$

s ekkor a j-edik mélységben levő ágat $\log \mathbf{P}\{y_j \mid x_j\}$ metrikával súlyozva ismét élenként additív metrikához jutunk, azaz az utak súlya az alkotó élek súlyának összege. Ezen fogalmak bevezetése után megadjuk a Viterbi-algoritmust. Az

egyértelműség kedvéért tegyük fel még, hogy a maximális súlyú utat keressük. A feladat tehát egy speciális irányított gráfban maximális súlyú út keresése. A Viterbi-algoritmus nem más, mint egy dinamikus programozási eljárás ügyes alkalmazása a konvolúciós kódok dekódolására.

A Viterbi-algoritmus

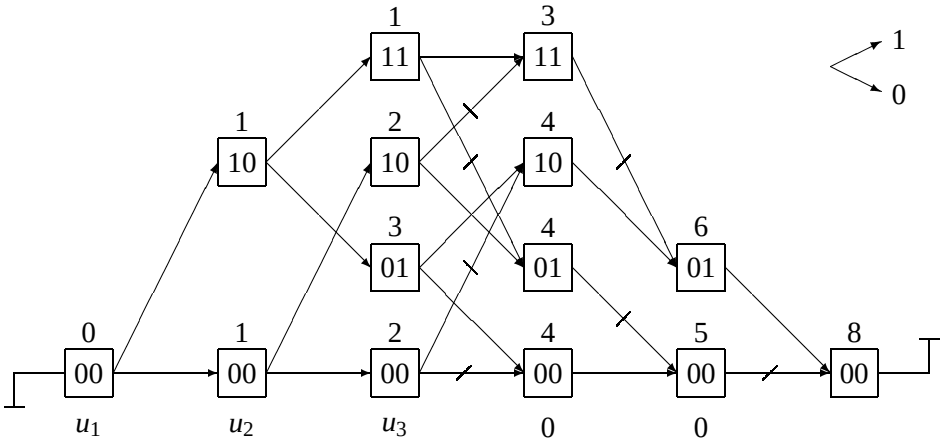
Tegyük fel, hogy a trellis *i*-edik mélységében ismerjük minden e mélységbeli csomópontba befutó azon utat, amelynek súlya ezen csomópontig maximális. Ezen részutakat az *i* mélységbeli **túlélő**knek nevezzük. Az *i*-edik mélységben tároljuk ezen túlélőket, valamint a túlélők súlyát.

Az $i+1$ mélységbeli túlélőket úgy határozzuk meg, hogy meghosszabbítjuk az *i* mélységbeli túlélőket a lehetséges folytatás élekkel, s az egyes $i+1$ mélységbeli csomópontokba a különböző *i* mélységbeli csomópontokból befutó egy-egy éllel meghosszabbított túlélőkből az lesz az új túlélő, amelyiknek nagyobb az egy ág súlyával megnövelt súlya.

Mivel a trellis útjai egy végcsomópontba futnak, ezért legvégül egyetlen túlélő marad, a maximális súlyú út.

4.17. példa. A 4.18. ábra trellise esetére legyen $\mathbf{y} = (0,1,0,1,0,1,0,1,1,1)$ a vett kódszó, ekkor a 4.28. ábrán követhetjük végig a dekódolás menetét.

A trellis csomópontjai felett jelöltük az adott csomópontba befutó minimális Hamming-távolságú részút (azaz a túlélő) súlyát, továbbá áthúzással jelöltük a túlélési versenyből kiesett részutak megfelelő éleit. A csomópontok felett megadott



4.28. ábra. Viterbi dekódolási példa.

számértékek a túlélő részutak súlyát jelentik. Figyelmesen végigkövetve az algoritmus lépéseit, eredményül az $(1,0,1,0,0)$ két 0-val meghosszabbított dekódolt üzenetet kapjuk.

4.16. A Viterbi-dekódolás bithibaaránya diszkrét, emlékezetnélküli csatornán

A következő részben egy módszert mutatunk a konvolúciós kódok ML dekódolása esetén elérhető dekódolási bithibaarány számítására DMC esetén.

A lineáris blokk-kódokkal kapcsolatos tanulmányokból ismeretes, hogy adott csatorna esetén a minimális kódszótávolság (röviden minimális távolság) nagysága döntően befolyásolja a kódszó-dekódolási hibaarányt. Hasonló a helyzet a konvolúciós kódokkal kapcsolatosan is, ezért első lépésként a kódszótávolság tulajdonságokat elevenítjük fel röviden.

A kódolás linearitása miatt itt is elegendő a csupa 0 kódszótól mérni a Hamming-távolságokat, amit a következőképpen láthatunk be. Legyen $\mathbf{u}$ és $\mathbf{u}'$ két különböző kódolandó üzenet, s jelölje $f(\mathbf{u})$ illetve $f(\mathbf{u}')$ a megfelelő kódszót, amelyet konvolúciós kódolóval a fentiekben részletesen elemzett blokk-kódolás üzemmódban nyertünk. Ekkor a

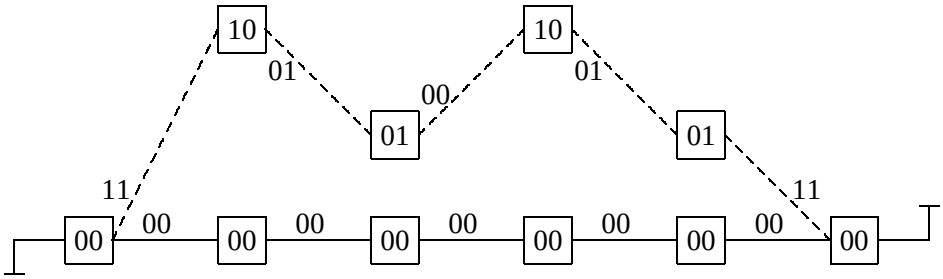
$$d(f(\mathbf{u}),f(\mathbf{u}')) = d(f(\mathbf{u})\oplus f(\mathbf{u}),f(\mathbf{u})\oplus f(\mathbf{u}')) = d(0,f(\mathbf{u}\oplus\mathbf{u}')) \qquad (4.51)$$

egyenlőségláncot kapjuk, ahol f linearitását használtuk fel. (4.51) alapján tehát $f(\mathbf{u})$ és $f(\mathbf{u}')$ kódszavak távolsága megegyezik egy nemzérus $f(\mathbf{u}\oplus\mathbf{u}')$ kódszó csupa 0 kódszótól mért távolságával. Következésképpen a lehetséges kódszótávolságok megegyeznek a nemzérus üzenethez tartozó kódszavak csupa zérus kódszótól mért távolságával, s ez az amit be akartunk látni.

Tegyük fel tehát, hogy a csupa 0 kódszót adjuk a csatornába. Kódoló példánk folytatásaként a 4.29. ábrán láthatjuk a csupa 0 kódszót, s egy nemzérus üzenethez tartozó kódszót.

A 4.29. ábra nemzérus kódszava $d = 6$ Hamming-távolságra van a zérus kódszótól, továbbá a neki megfelelő $(1,0,1,0,0)$ üzenet $i = 2$ nemzérus üzenetbitet tartalmaz. Ha tehát a zérus kódszó helyett hibásan ezt a kódszót dekódolnánk, akkor az 2 bithibát okozna.

A konvolúciós kódoknál — a blokk-kódoktól eltérően — az egy kódszóba kódolandó üzenethossz, s így a kódszóhossz, elvileg szabadon választható paraméter. A dekódolás hibaanalíziséhez szükségünk lesz a kódszavak súlyeloszlására. Tudjuk, hogy a kódszavaknak utak felelnek meg a kód trellisében. Vezessük be az $a(d,i)$ jelölést a zéró kódszónak megfelelő útból (zéró út) az 1. csomópontban leágazó azon utak darabszámára, amelyek d Hamming-távolságra vannak és i súlyú



4.29. ábra. Zérus kódszóból leágazó nemzérus kódszó.

üzenethez tartoznak. Bár véges méretű (L bit) üzenetblokkot kódolunk egy lépésben, technikailag egyszerűbb az $a(d,i)$ eloszlás számítása ha végtelen hosszú üzenetblokkok kódolását vizsgáljuk.

4.18. példa. A 4.14. ábra kódolója esetén $a(d,i) = 0$, ha $d < 5$, továbbá $a(5,1) = 1$, $a(6,2) = 2$, ...

Vezessük be az $\{a(d,i) : i,d = 1,2,\ldots\}$ kétindexes sorozathoz rendelt formális hatványsort:

$$T(D,I) = \sum_{i=1}^{\infty} \sum_{d=1}^{\infty} a(d,i) I^i D^d. \tag{4.52}$$

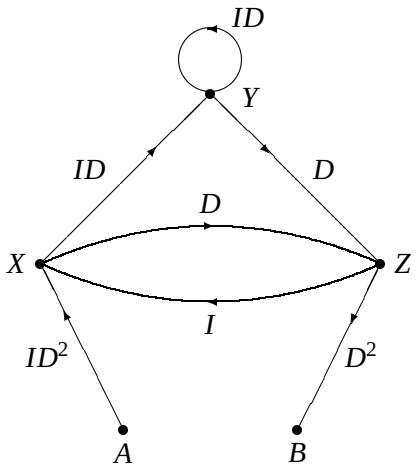
(A formális hatványsorok tulajdonságaival kapcsolatosan utalunk pl. [34] 10.3. pontjára).

4.19. példa ($T(D,I)$ meghatározásának egy módszere).
A 4.16. ábrán látható állapotdiagramot a 4.30. ábrán látható folyamatgráffá alakíthatjuk.

A leágazás-folyamatgráf A,B jelű csomópontja a 00 állapotnak, míg az X,Y,Z jelű csomópontok a három nemzéró állapotnak felelnek meg. Így a leágazás-folyamatgráf A bemeneti pontja a zéró útból történő leágazásnak, míg a B kimeneti pontja az abba való első visszatérésnek felel meg. A gráf éleinek címkéje $I^j D^k$, ha ezen él $j = 0$ vagy $j = 1$ üzenetbit hatására jött létre, illetve k Hamming-súlyú szelete a megfelelő útnak. Egy úthoz az élei címkéinek szorzatát rendeljük. A leágazás-folyamatgráf A bemeneti és B kimeneti pontja közötti utakra így kiadódó értékek összege a keresett $T(D,I)$ sort adja.

Példabeli kódolónk esetén az

$$X = ID^2A + IZ,$$



4.30. ábra. A konvolúciós kódoló leágazás-folyamatgráfja.

$$Y = IDX + IDY,$$
$$Z = DX + DY,$$
$$B = D^2 Z$$

lineáris egyenletrendszerből

$$B = T(D,I) \cdot A$$

felhasználásával, $T(D,I)$ megkapható:

$$T(D,I) = \frac{ID^5}{1-2ID} = ID^5(1+2ID+4I^2D^2+\cdots). \tag{4.53}$$

Innen kiolvasható a keresett súlyeloszlás:

$$a(d,i) = \begin{cases} 2^{i-1}, & \text{ha } d = i+4,\ i = 1,2,3,\ldots \\ 0, & \text{egyébként} \end{cases}. \tag{4.54}$$

Működjön a kódoló blokk-kódolóként véges üzenethosszal, használjuk a Viterbi-dekódolást. A dekódolás eredménye egy a leadott zéró úttól esetleg eltérő út, amely bizonyos csomópontokban leágazhat a zéró útból, majd egy későbbi csomópontban visszatérhet bele, s esetleg újra elágazhat belőle.

A dekódolt út szakaszokra bontható: egyes szakaszain együttfut a csupa zéró úttal, majd leágazik, s visszatér abba. A leágazástól a visszatérésig tartó részutat huroknak nevezzük. A fentiekben megállapítottuk a hurkok $a(d,i)$ eloszlását.

Jelölje a W_j valószínűségi változó az üzenetbit-hibák számát a dekódolt út j-edik mélységben induló hurokjában ($j = 0, 1, \ldots, L-1$). Definíció szerint legyen $W_j = 0$, ha nincs leágazás a j-edik pontban, vagy ha egy j-t megelőző pontban indult egy hurok, s még nem tért vissza a zéró útba.

Ha L jelöli az üzenethosszt, akkor W_j definíciója alapján nyilván az $\eta = \sum_{j=0}^{L-1} W_j$ összeg az üzenetbithibák számát adó valószínűségi változó. Az $\frac{\eta}{L}$ valószínűségi változó adja az egy dekódolt bitre eső bithibaarányt. Ennek várható értékével definiáljuk a P_b átlagos bithibaarányt, azaz

$$P_b = \mathbf{E}\left(\sum_{j=1}^{L} \frac{W_j}{L}\right). \tag{4.55}$$

DMC esetén nem nehéz felső becslést adni annak a valószínűségére, hogy az átküldött zéró kódszó helyett egy attól d Hamming-távolságra levő kódszóra döntünk ML dekódolás esetén. Bevezetve ezen (pár-) hibázás esemény valószínűségére a $P_{\text{e,pár}}$ valószínűséget, arra egy

$$P_{\text{e,pár}} \leq (2^{-w})^d \tag{4.56}$$

alakú felső becslés adható, ahol $w \geq 0$ az adott csatorna paramétereitől függő mennyiség (pl. w a Bhattacharyya-távolság). Ez a következőképp látható be. Legyen $\mathbf{x}_1$ a leadott kódszó, $\mathbf{x}_2$ egy tőle d távolságra levő kódszó, valamint legyen $\mathbf{y}$ a vett szó. A vételi oldalon természetesen nem tudjuk, hogy $\mathbf{x}_1$ volt a leadott kódszó, s így $\mathbf{y}$ ismeretében az $\mathbf{x}_1$ illetve $\mathbf{x}_2$ kódszavak közül a valószínűbben leadottra szeretnénk dönteni. Vezessük be a $\mathcal{D}_1, \mathcal{D}_2$ ML döntési tartományokat, ahol

$$\begin{aligned} &\mathbf{y} \in \mathcal{D}_1, \quad \text{ha} \quad \mathbf{P}\{\mathbf{y} \mid \mathbf{x}_1\} \geq \mathbf{P}\{\mathbf{y} \mid \mathbf{x}_2\}, \\ &\mathbf{y} \in \mathcal{D}_2, \quad \text{ha} \quad \mathbf{P}\{\mathbf{y} \mid \mathbf{x}_2\} \geq \mathbf{P}\{\mathbf{y} \mid \mathbf{x}_1\}. \end{aligned} \tag{4.57}$$

Hibát akkor követünk el, ha $\mathbf{y} \in \mathcal{D}_2$. Ennek megfelelően a hibavalószínűség

$$P_{\text{e,pár}} = \sum_{\mathbf{y} \in \mathcal{D}_2} \mathbf{P}\{\mathbf{y} \mid \mathbf{x}_1\}. \tag{4.58}$$

(4.57) és (4.58) alapján a következő felső becslést adhatjuk:

$$\begin{aligned} P_{\text{e,pár}} &\leq \sum_{\mathbf{y} \in \mathcal{D}_2} \mathbf{P}\{\mathbf{y} \mid \mathbf{x}_1\} \sqrt{\frac{\mathbf{P}\{\mathbf{y} \mid \mathbf{x}_2\}}{\mathbf{P}\{\mathbf{y} \mid \mathbf{x}_1\}}} \leq \\ &\leq \sum_{\mathbf{y}} \sqrt{\mathbf{P}\{\mathbf{y} \mid \mathbf{x}_1\}\mathbf{P}\{\mathbf{y} \mid \mathbf{x}_2\}} = \end{aligned}$$

$$= \prod_{n=1}^{N} \left(\sum_{y} \sqrt{\mathbf{P}\{y \mid x_{1n}\}\mathbf{P}\{y \mid x_{2n}\}} \right) =$$

$$= \left(\sum_{y} \sqrt{\mathbf{P}\{y \mid 0\}\mathbf{P}\{y \mid 1\}} \right)^{d}, \tag{4.59}$$

ahol az utolsó lépésben kihasználtuk, hogy azon n-ekre, amelyekre $x_{1n} = x_{2n}$, a szumma 1-gyel egyenlő. Bevezetve a

$$w = -\log_2 \left(\sum_{y} \sqrt{\mathbf{P}\{y \mid 0\}\mathbf{P}\{y \mid 1\}} \right) \tag{4.60}$$

jelölést, a bizonyítandó (4.56) felső becslésre jutunk.

Vezessük be az $a_j(d,i)$ jelölést a vizsgált véges üzenethosszra a zéró kódszónak megfelelő útból (zéró út) a j-edik csomópontban leágazó azon hurkok darabszámára, amelyek d Hamming-távolságra vannak és i súlyú üzenethez tartoznak, s amelyek legkésőbb az L-edik mélységben visszatérnek. Nyilván $a_j(d,i) \leq a(d,i)$ tetszőleges d, i és j esetén.

Az eddigiek alapján az

$$\mathbf{E}(W_j) \leq \sum_{i} i \cdot \sum_{d} a_j(d,i)(2^{-w})^d \tag{4.61}$$

felső becslést kapjuk, ahonnan $a_j(d,i) \leq a(d,i)$ felhasználásával a

$$P_b \leq \sum_{i} \sum_{d} i \cdot a(d,i)(2^{-w})^d \tag{4.62}$$

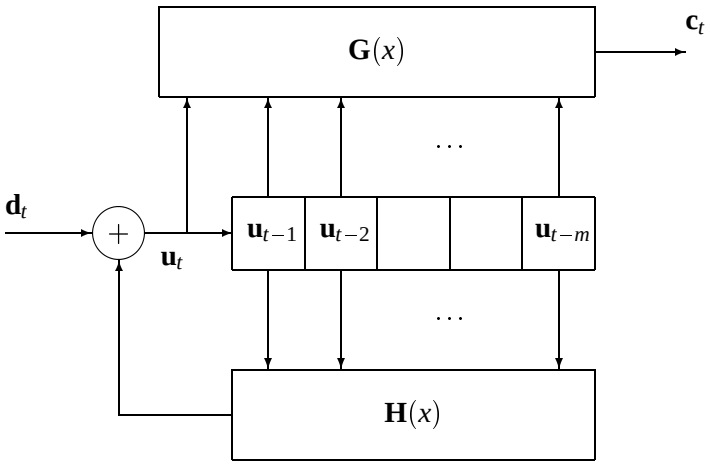
eredményt kaphatjuk. A (4.52) és (4.62) összefüggések egybevetésével a

$$P_b \leq \left. \frac{\partial T(D,I)}{\partial I} \right|_{I=1, D=2^{-w}} \tag{4.63}$$

praktikus végformulához jutunk.

4.17. Rekurzív konvolúciós kódolás

A 4.14. szakaszban láttuk, hogy a konvolúciós kódokat generálhatjuk előrecsatolt shiftregiszterek segítségével. Ha a regiszterek tartalmát valamilyen lineáris logikán keresztül visszacsatoljuk a kódoló bemenetére, egy általánosabb kódosztályhoz jutunk, a **rekurzív (visszacsatolt) konvolúciós kód**okhoz. Ezek a kódok



4.31. ábra. Visszacsatolt konvolúciós kódoló.

szintén lineáris időinvariáns fa-kódok, ám a kényszerhossz a legtöbb esetben végtelen: egy darab 1-es bit végtelen hosszú ideig keringhet a kódolóban, így egy egy súlyú bemenet végtelen súlyú kimenetet generálhat. Ez a tulajdonság igen hasznos lehet bizonyos kódkonstrukcióknál, például a 4.18. szakaszban tárgyalt turbó kódoknál. Az előrecsatolt konvolúciós kódokhoz hasonlóan továbbra is csak bináris kódokkal foglalkozunk.

A visszacsatolt kódoló struktúrája a 4.31. ábrán látható. Itt

$$\mathbf{d}_t = [d_{t,1}, \ldots, d_{t,k}]$$

jelöli a kódolóba a t időpontban belépő k bites üzenetkeretet,

$$\mathbf{c}_t = [c_{t,1}, \ldots, c_{t,n}]$$

az onnan kilépő n bites kódszókeret, végül pedig

$$\mathbf{u}_t = [u_{t,1}, \ldots, u_{t,k}]$$

a shiftregiszterbe a t időpontban belépő szintén k bites úgynevezett állapotkeretet.

A kódolót két generátorpolinom-mátrixszal írhatjuk le, hiszen a shiftregiszterben tárolt állapotkeretek nem csak a kimenetet, hanem a shiftregiszterbe belépő új állapotkeretet is befolyásolják (az előrecsatolt esettel ellentétben a regiszter nem az üzenetkereteket, hanem azoknak egy módosított változatát, az állapotkereteket tartalmazza). Vegyük észre, hogy a bemenet közvetlenül nem befolyásolja a

kimenetet, hiszen az első kicsatolás az első összeadó után van. (Ez az analízist megkönnyítő elrendezés azonban nem jelent megszorítást, hiszen az előrecsatoló súlyok alkalmas megváltoztatásával ez a kicsatolás az összegzés elé hozható.) Az előrecsatolást a $k \times n$ méretű

$$\mathbf{G}(x) = [g_{i,j}(x)]$$

mátrix határozza meg, ahol $g_{i,j}(x) = g_{i,j,0} + g_{i,j,1}x + \cdots + g_{i,j,m}x^m$ írja le az állapotkeretek i-edik és a kódszókeretek j-edik bitje közötti összefüggést, ahol m — az előrecsatolt konvolúciós kódoknál használt jelölésekhez hasonlóan — a kódoló regisztereinek száma. Az állapotkeretek i-edik és az új állapotkeret j-edik bitje közötti kapcsolatot a $h_{i,j}(x) = h_{i,j,0} + h_{i,j,1}x + \cdots + h_{i,j,m}x^m$ polinom írja le, ahol $h_{i,j,0}$, a konstans tag együtthatója, az üzenetkeret i-edik bitjének hatását mutatja az új állapotkeret j-edik bitjére. A $h_{i,j}(x)$ polinomokat mátrixba rendezve kapjuk a visszacsatolást meghatározó

$$\mathbf{H}(x) = [h_{i,j}(x)]$$

$k \times k$ méretű négyzetes mátrixot. A továbbiakban azzal a tipikus esettel foglalkozunk, amikor

$$h_{i,j,0} = \delta_{i,j} = \begin{cases} 1, & \text{ha } i = j \\ 0, & \text{ha } i \neq j \end{cases} .$$

Ennek szemléltetésére a 4.31. ábrán a bemenetet közvetlenül nem is vezettük be a visszacsatolást leíró $\mathbf{H}(x)$ blokkba.

4.20. példa. A 4.32. ábrán egy egy bemenetű egy kimenetű visszacsatolt konvolúciós kódoló általános sémája látható.

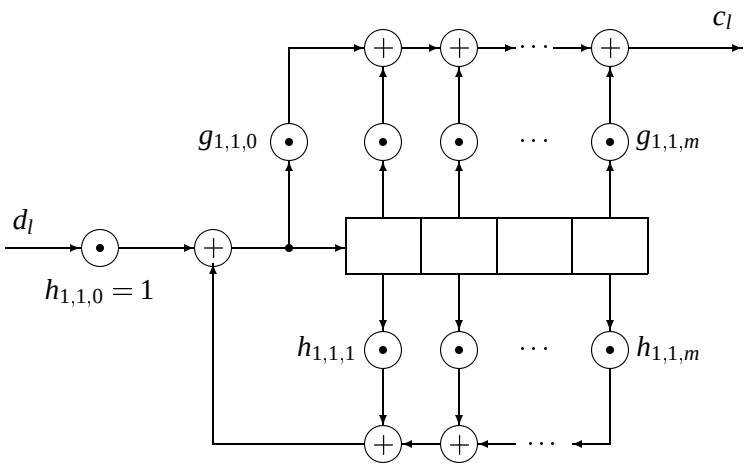
A rekurzív konvolúciós kódok az előrecsatolt konvolúciós kódokkal teljesen analóg módon kezelhetőek. Ugyanúgy elkészíthetjük a kódoló állapotgráfját, trellisét, és a Viterbi-dekódolást is változtatás nélkül alkalmazhatjuk. Mindebből következik, hogy a Viterbi-dekódolás bithibaarányára korábban adott felső becslés is érvényben marad.

A továbbiakban meghatározzuk az előrecsatolt konvolúciós kódoknál már megismert

$$\mathbf{c}(x) = [c_1(x), \ldots, c_n(x)]$$

kódszópolinom-vektort a

$$\mathbf{d}(x) = [d_1(x), \ldots, d_k(x)]$$



4.32. ábra. Visszacsatolt konvolúciós kódoló, $k=1$, $n=1$.

üzenetpolinom-vektor segítségével, ahol

$$c_j(x) = c_{0,j} + c_{1,j}x + c_{2,j}(x) + \dots \qquad j = 1,\dots,n$$

és

$$d_j(x) = d_{0,j} + d_{1,j}x + d_{2,j}(x) + \dots \qquad j = 1,\dots,k.$$

A levezetéshez szükségünk lesz még az

$$\mathbf{u}(x) = [u_1(x),\dots,u_k(x)]$$

állapotpolinom-vektorra, melynek elemei az

$$u_j(x) = u_{0,j} + u_{1,j}x + u_{2,j}(x) + \dots \qquad j = 1,\dots,k$$

polinomok. Ha a kódolót a csupa nulla állapotból indítjuk, akkor definíció szerint legyen $u_{t,j} = 0$, ha $t = -1,\dots,-m$ és $j = 1,\dots,k$. Ekkor a visszacsatolásokat figyelembe véve az

$$u_{t,j} = \sum_{l=1}^{m}\sum_{i=1}^{k} h_{i,j,l}u_{t-l,i} + d_{t,j} \tag{4.64}$$

alakú egyenletekhez jutunk minden $t = 0,1,\dots$ és $j = 1,\dots,k$ értékre. Szorozzuk meg a (4.64) egyenletet x^t-nel, és a kapott egyenleteket összegezzük minden

nemnegatív t-re. Ekkor

$$\begin{aligned} u_j(x) &= \sum_{t=0}^{\infty} u_{t,j}x^t = \\ &= \sum_{t=0}^{\infty}\sum_{l=1}^{m}\sum_{i=1}^{k} h_{i,j,l}u_{t-l,i}x^t + \sum_{t=0}^{\infty} d_{t,j}x^t = \\ &= \sum_{t=0}^{\infty}\sum_{l=1}^{m}\sum_{i=1}^{k} h_{i,j,l}x^l u_{t-l,i}x^{t-l} + \sum_{t=0}^{\infty} d_{t,j}x^t = \\ &= \sum_{l=1}^{m}\sum_{i=1}^{k} h_{i,j,l}x^l \sum_{t=0}^{\infty} u_{t-l,i}x^{t-l} + \sum_{t=0}^{\infty} d_{t,j}x^t = \\ &= \sum_{i=1}^{k}\sum_{l=1}^{m} h_{i,j,l}x^l u_i(x) + d_j(x). \end{aligned}$$

Figyelembe véve, hogy bináris kódokkal dolgozunk és $h_{i,j,0} = \delta_{i,j}$, a

$$d_j(x) = \sum_{i=1}^{k} h_{i,j}(x)u_i(x) = [u_1(x),\ldots,u_k(x)][h_{1,j}(x),\ldots,h_{k,j}(x)]^T$$

egyenletekhez jutunk, $j = 1,\ldots,k$. Innen pedig a

$$\mathbf{d}(x) = \mathbf{u}(x)\mathbf{H}(x)$$

kifejezéshez jutunk, amiből

$$\mathbf{u}(x) = \mathbf{d}(x)\mathbf{H}^{-1}(x)$$

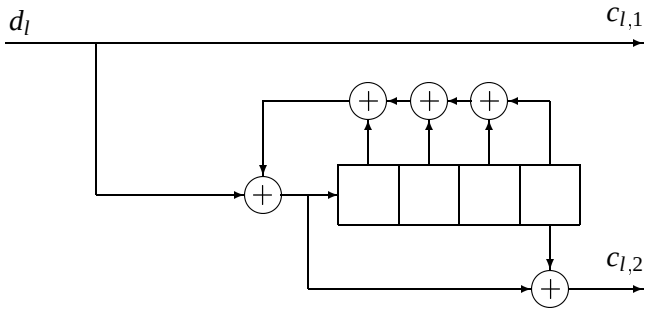
adódik, ahol $\mathbf{H}^{-1}(x)$ a $\mathbf{H}(x)$ inverze. Ekkor, mivel a kimenet megkapható egy $\mathbf{u}(x)$ bemenetű $\mathbf{G}(x)$ előrecsatolású konvolúciós kódoló segítségével, az előrecsatolt konvolúciós kódoknál bizonyítottak alapján a kódszópolinom-vektor a

$$\mathbf{c}(x) = \mathbf{d}(x)\mathbf{H}^{-1}(x)\mathbf{G}(x) \tag{4.65}$$

formában írható. Az igen fontos $k = 1$ speciális esetben

$$\mathbf{c}(x) = \frac{d_1(x)}{h_{1,1}(x)}[g_{1,1}(x),\ldots,g_{1,n}(x)] \tag{4.66}$$

adódik. Azon speciális esetben, amikor nincs visszacsatolás, azaz a kódolónk ténylegesen egy előrecsatolt konvolúciós kódoló, $\mathbf{H}(x)$ éppen a $k \times k$ méretű egységmátrix, és így (4.65) a korábban már megismert $\mathbf{c}(x) = \mathbf{d}(x)\mathbf{G}(x)$ összefüggésre egyszerűsödik.



4.33. ábra. Rekurzív szisztematikus konvolúciós kódoló.

4.21. példa. A 4.33. ábrán látható rekurzív szisztematikus konvolúciós kódoló esetén $g_{1,2}(x) = 1 + x^4$ és $h_{1,1}(x) = 1 + x + x^2 + x^3 + x^4$. Vegyük észre, hogy az első kimenet kicsatolása az első összeadó előtt történik, és nyilván $c_1(x) = d_1(x)$ (ez egyébként némi átrendezés után megfelel a $g_{1,1}(x) = h_{1,1}(x)$ esetnek). Ekkor

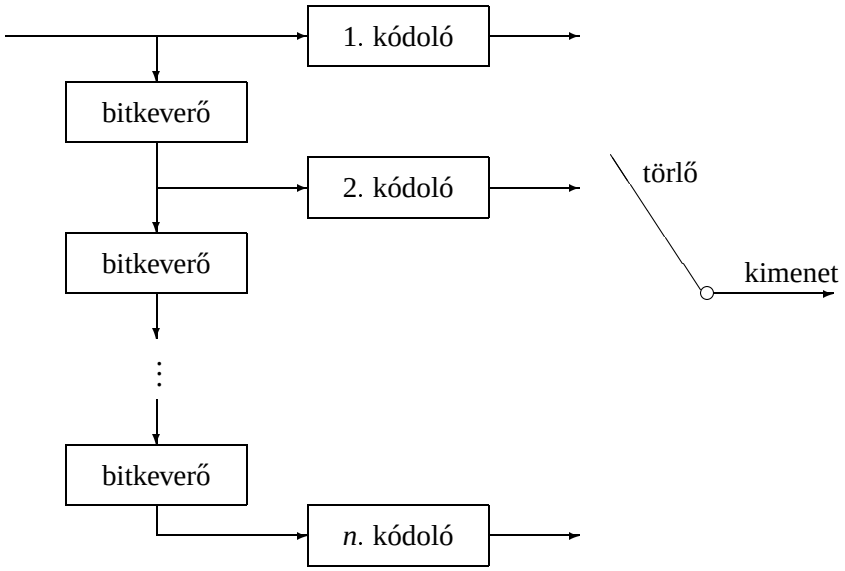
$$\begin{aligned} \mathbf{c}(x) &= [c_1(x), c_2(x)] = \\ &= d_1(x)\left(1, \frac{g_{1,2}(x)}{h_{1,1}(x)}\right) = \\ &= d_1(x)\left(1, \frac{1+x^4}{1+x+x^2+x^3+x^4}\right). \end{aligned} \tag{4.67}$$

4.18. Turbó kódok

A turbó kódok a kódelméleti kutatás legújabb vonalát jelentik. 1993-as felfedezésükkel (Berrou, Glavieux és Thitimajshima [9]) megjelentek az első olyan praktikus kódok, melyek jelsebessége megközelíti a csatornakapacitást. Jelen pillanatban úgy tűnik, hogy széles körű elterjedésüknek csak a kódolás és dekódolás során fellépő nagy késleltetés, illetve a viszonylag nagyobb komplexitás szab határt (bár ez utóbbi tekintetben igen biztató eredmények is megjelentek már, például [29]). Az űrtávközlésben napjainkra a turbó kódokat alkalmazzák leggyakrabban (például EN 301 790 szabvány), és alkalmazásukat az UMTS szabvány (4.19. szakasz) is javasolja.

Az eljárás alapgondolata az, hogy a 4.34. ábra szerint bitkeverők közbeiktatásával párhuzamosan kapcsolunk több kódolót, az úgynevezett komponenskódolókat, és ezek kódszavait összefésülve kapjuk a tényleges kódszót. A gyakorlatban a komponenskódok leggyakrabban visszacsatolt konvolúciós kódok.

A dekódolás során a komponenskódokat külön-külön dekódoljuk, azonban a dekódolás során felhasználjuk a többi komponenskód dekódolása során kapott

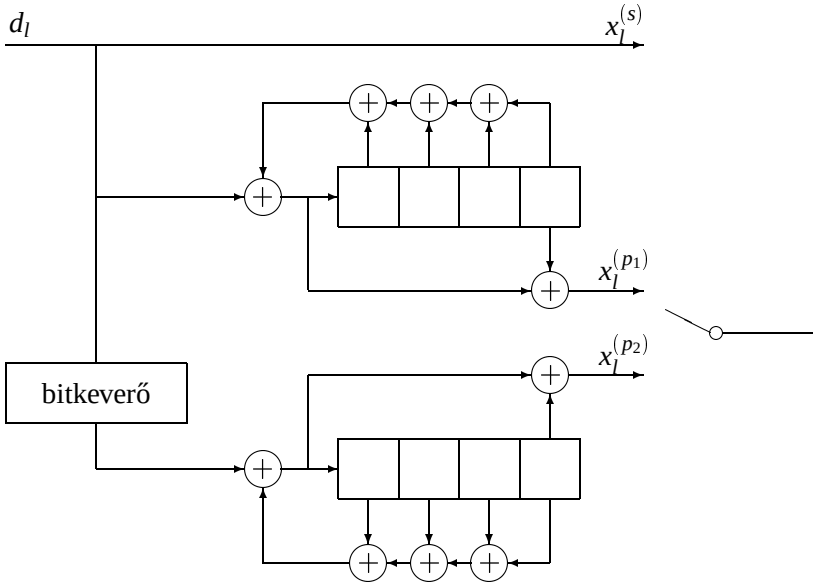


4.34. ábra. A turbó kódoló általános struktúrája.

eredményeket. A tapasztalatok szerint az így elérhető hibavalószínűség nagyságrendekkel kisebb lehet, mint ha az egyes komponenskódokat külön-külön használnánk. n kódoló összekapcsolása esetén egy $1/n$ sebességű kódot kapunk. Ahhoz, hogy a kódsebességet az alkalmazások igényei szerint alakíthassuk, az egyes komponenskódszavak bizonyos bitjeit egy előre meghatározott minta szerint töröljük, azaz nem továbbítjuk. A vevőben a törölt bitek pozíciója ismert, így a dekódolás során ezek törléses hibáknak tekinthetőek, és ennek megelelően javíthatóak. A gyakorlati dekódolási eljárások során azonban a törölt biteket a vevőben leggyakrabban nullákkal helyettesítjük.

4.22. példa. A 4.35. ábrán egy standard turbó kódoló struktúrája látható. A két komponenskód azonos, a 4.21. példa $1/2$ sebességű rekurzív szisztematikus konvolúciós kódja. Mivel a két kódban a szisztematikus rész a sorrendtől eltekintve azonos (hiszen a bemeneti bitsorozatot a 2. kódoló számára permutáltuk), ezt csak egyszer továbbítjuk: így összességében egy $1/3$ sebességű kódhoz jutunk (bizonyos paritásbitek törlésével a jelsebesség tetszőlegesen megközelítheti az 1-et).

A továbbiakban ezen a példán magyarázzuk el a turbó kódok működését; ennek általánosítása több illetve más típusú komponenskódokra nem okozhat problémát. Először is fontos megjegyezni, hogy bár a turbó kódok magját konvolúciós



4.35. ábra. Turbó kódoló.

kódolók alkotják, ezek a kódok ténylegesen blokk-kódok, ahol egy blokk mérete a bitkeverő hossza. Mind a kódolást, mind a dekódolást blokkonként külön végezzük el. A blokkméretet a továbbiakban K-val jelöljük, tipikus értéke 2^{10}–2^{16} között van.

Az érkező $d_1, d_2, \ldots, d_K$ adatsorozatot közvetlenül továbbítjuk a kimenetre, így kapjuk az $x_1^{(s)}, x_2^{(s)}, \ldots, x_K^{(s)}$ sorozatot (az s felső index a szisztematikus szóra utal). Az első komponens kódoló a $d_1, d_2, \ldots, d_K$ bemenetre az $x_1^{(p_1)}, x_2^{(p_1)}, \ldots, x_K^{(p_1)}$ paritásbit-sorozatot generálja. A második kódoló bemenetére azonban — a bitkeverő közbeiktatása miatt — az eredeti adatsorozat egy permutált változata, a $d'_1, d'_2, \ldots, d'_K$ sorozat kerül, és ehhez generálja a második kódoló az $x_1^{(p_2)}, x_2^{(p_2)}, \ldots, x_K^{(p_2)}$ paritásbit-sorozatot. Így végül a csatornába az $x_1^{(s)}, x_1^{(p_1)}, x_1^{(p_2)}, x_2^{(s)}, x_2^{(p_1)}, x_2^{(p_2)}, \ldots, x_K^{(s)}, x_K^{(p_1)}, x_K^{(p_2)}$ bitsorozat kerül. Törlés alkalmazása esetén a paritásbit-sorozatok előre meghatározott elemei nem kerülnek továbbításra.

A dekódolás megkönnyítése érdekében az első komponenskódolót blokk-kódolás üzemmódban használjuk; visszacsatolt konvolúciós kódolók esetén is elérhető, hogy a kódolót tetszőleges állapotból elindítva m új bit beléptetésével a kódoló a 0 állapotba kerüljön. A különbség csupán annyi, hogy míg az előrecsatolt esetben ez mindig m darab 0 beléptetésével történt, addig a 4.22. példában

ehhez a 0110 állapotból kiindulva rendre a $0,0,1,0$ karaktereket kell a bemenetre adni. Mivel a két kódoló párhuzamosan működik, általában nem érhető el, hogy a második kódoló is egy előre meghatározott állapotban álljon meg.

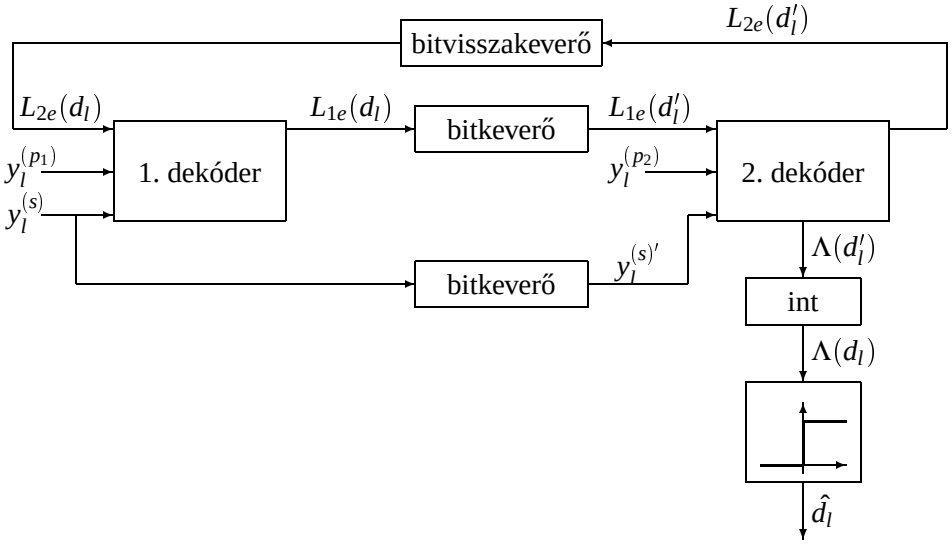
A kódoló utolsó — eddig még nem vizsgált — komponenese a bitkeverő. A hagyományosan használatos strukturált bitkeverőkkel ellentétben (ilyen például a blokk-bitkeverő, amikor egy négyzetes mátrixba sorfolytonosan írjuk az adatokat és oszlopfolytonosan olvassuk ki) a turbó kódok esetében fontos, hogy a permutáció minden látható struktúrát nélkülözzön: ez biztosítja, hogy a kód a csatornakódolási tétel bizonyítása során megismert nagy blokkhosszú véletlen kód jellegű legyen. Ezért leggyakrabban álvéletlen bitkeverőket használnak.

A turbó kódok dekódolása

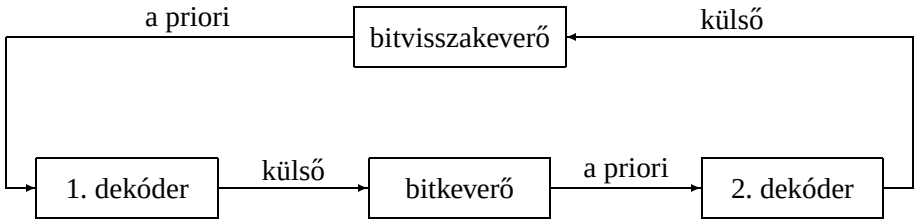
A bitkeverő jelenléte miatt a turbó kódok ML dekódolása praktikus (nagy) blokkhosszak esetén gyakorlatilag megvalósíthatatlan a nagy komplexitásból adódó számítási kapacitásigény miatt, ezért a dekódolás során egy szuboptimális iteratív eljárást, a BCJR algoritmust szokás alkalmazni (léteznek más dekódolási módszerek is, illetve az itt bemutatott módszer javított változatai, melyekre itt nem térünk ki). Az eljárás lényegét Bahl, Cocke, Jelinek és Raviv dolgozták ki 1974-ben [5], és a turbó kódok megjelenésével kis módosítással újra az érdeklődés középpontjába került. Az iterációs lépések során a dekódoló az egyes bitek eloszlására becslést ad, és a következő iterációban a dekódolást ezen a priori eloszlást feltételezve végezzük el. Egy iteráció két lépésből áll, melynek során előbb az első, majd a második komponenskód dekódolását végezzük el, a dekódolt eloszlást a fentiek szerint továbbadva az első dekóderből a másodiknak, majd a másodikból az elsőnek, és így tovább.

A dekóder blokkdiagrammját a 4.36. ábra mutatja, ahol $y_l^{(s)}, y_l^{(p_1)}, y_l^{(p_2)}$ a csatorna kimenetén az $x_l^{(s)}, x_l^{(p_1)}, x_l^{(p_2)}$ bementek hatására megjelenő bitek, $L_{1e}(d_l)$ és $L_{2e}(d_l)$ az egyes komponenskódok dekódolása során nyert ún. külső információ, melyet a dekódolás következő lépésében a priori információként használunk fel. Figyeljük meg, hogyan biztosítja a bitek megfelelő sorrendjét az első és második dekóder között lévő bitkeverő illetve a második és első dekóder között elhelyezkedő bitvisszakeverő, amely az előző bitkeverő inverz leképzését valósítja meg, azaz a permutált bitsorozatot visszaállítja az eredeti sorrendbe.

A dekódolás során először az $y_l^{(s)}$, $y_l^{(p_1)}$ értékek és az $L_{2e}(d_l) = \log \frac{\mathbf{P}\{d_l=1\}}{\mathbf{P}\{d_l=0\}}$ a priori információ alapján ($l = 1,\ldots,K$) egy új $L_{1e}(d_l)$ becslést készít a $\log \frac{\mathbf{P}\{d_l=1\}}{\mathbf{P}\{d_l=0\}}$ kifejezésre. A 2. dekóder ezt az információt, a megfelelő sorrendbe rakott $y_l^{(s)'}$ szisztematikus biteket és az $y_l^{(p_2)}$ paritásbiteket felhasználva új $L_{2e}(d_l)$ becslést



4.36. ábra. A turbó dekóder blokkdiagrammja.



4.37. ábra. Információáramlás a két dekóder között.

készít. Az eljárást addig ismételjük, amíg az $L_{1e}(d_l)$ és $L_{2e}(d_l)$ értékek csak kicsit változnak meg. (Másik lehetséges megállási feltétel, ha az iterációk számát előre rögzítjük.) A dekóderek közötti információáramlást jól szemlélteti a 4.37. ábra. Ezután a 2. dekóder az $L_{1e}(d_l)$ apriori információt feltételezve kiszámítja a tényleges

$$\Lambda(d_l) = \log \frac{\mathbf{P}\{d_l = 1 | y_l^{(s)'}, y_l^{(p_2)}, l = 1, \ldots, K\}}{\mathbf{P}\{d_l = 0 | y_l^{(s)'}, y_l^{(p_2)}, l = 1, \ldots, K\}}$$

értéket. Ha $\Lambda(d_l) > 0$, akkor értelemszerűen $d_l = 1$-re döntünk, egyébként $d_l = 0$-ra. (A $\Lambda(d_l)$ és $L_{2e}(d_l)$ közötti különbségre itt nem térünk ki. Az eltérés célja az iteratív dekódolás stabilitásának biztosítása.)

A turbó kódok teljesítményanalízise

Ebben a részben megmutatjuk, hogyan lehet egyszerűen becsülni a turbó kódok ML dekódolásának hibavalószínűségét, és hogy milyen jellegű teljesítménygörbét várhatunk el a turbó kódoktól. Mivel a Gauss-csatorna gyakorlati szempontból sokkal fontosabb a bináris csatornáknál, ebben a részben erre koncentrálunk.

A bithibavalószínűség vizsgálatakor a konvolúciós kódoknál már megismert unió-korlátot fogjuk alkalmazni. Mivel a turbó kódok lineáris kódok, a korábban megszokottak szerint azt feltételezzük, hogy a csupa 0 üzenetet küldtük át, és ehhez vizsgáljuk a hibázás valószínűségét. A bithiba valószínűség abban az esetben, ha az n-edik kódszóra döntünk a 0 helyett

$$P_b(n \mid 0) = w_n/K\,\mathbf{P}_{e,\text{pár},n},$$

ahol w_n az n-edik kódszóhoz tartozó nem 0 üzenetbitek száma, azaz az n-edik kódszó üzenetsúlya, $\mathbf{P}_{e,\text{pár},n}$ pedig annak a valószínűsége, hogy az n-edik kódszóra hibázunk. Ha az n-edik kódszó súlya d_n, akkor — a Viterbi-dekódolásnál látottak alapján — p paraméterű bináris szimmetrikus csatorna esetén ($p < 1/2$)

$$\mathbf{P}_{e,\text{pár},n} \le (2\sqrt{p(1-p)})^{d_n},$$

Gauss-csatorna esetén pedig megmutatható [19], hogy

$$\mathbf{P}_{e,\text{pár},n} = Q\left(\sqrt{\frac{2d_nR_cE_b}{N_0}}\right),$$

ahol $Q(x) = \Phi(-x)$ a standard normális eloszlás farokeloszlása, R_c a kód sebessége, E_b az egy információs bit átviteléhez felhasznált energia, $N_0/2$ pedig a csatorna kétoldali spektrális sűrűségfüggvénye (az E_b/N_0 értéket jel–zaj viszonynak nevezik). Ekkor a bithibavalószínűség

$$\begin{aligned} P_b &\le P_b(\text{valamilyen } n\text{-re hibázunk}) \\ &\le \sum_{n=1}^{2^K} P_b(n \mid 0) \le \begin{cases} \sum_{n=1}^{2^K} \frac{w_n}{K}(2\sqrt{p(1-p)})^{d_n} & \text{BSC} \\ \sum_{n=1}^{2^K} \frac{w_n}{K} Q\left(\sqrt{\frac{2d_nR_cE_b}{N_0}}\right) & \text{Gauss} \end{cases} \end{aligned} \tag{4.68}$$

csatorna esetén.

Jelöljük a d súlyú kódszavak számát M_d-vel, és az ilyen kódszavak üzenetsúlyának összegét W_d-vel. Definiáljuk a kódszavankénti átlagos üzenetsúlyt a következő módon:

$$\tilde{w} = \frac{W_d}{M_d}.$$

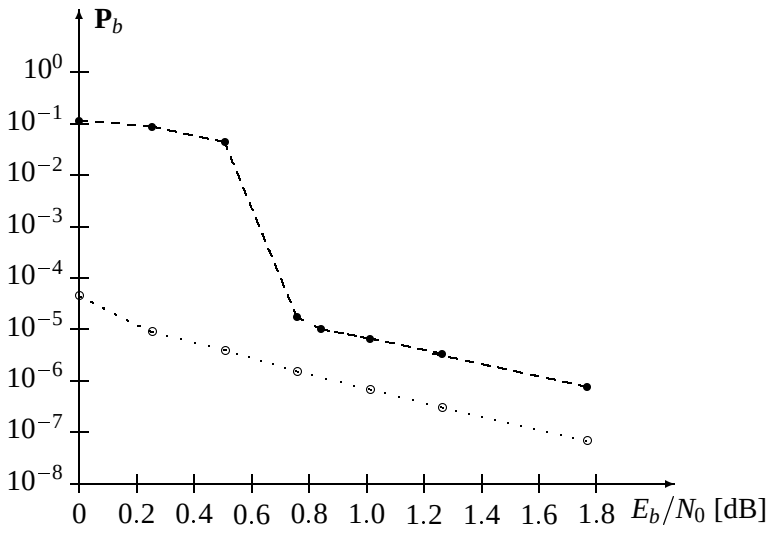
Mivel egy K hosszú bemenet K/R_c hosszú kimenetet eredményez, ekkor Gauss csatorna esetén a bithibavalószínűségre vonatkozó felső korlát a következő formában írható:

$$P_b \leq \sum_{d=d_{free}}^{K/R_c} \frac{M_d \tilde{w}_d}{K} Q\left(\sqrt{\frac{2dR_cE_b}{N_0}}\right) \approx$$
$$\approx \frac{M_{free}\tilde{w}_{free}}{K} Q\left(\sqrt{\frac{2d_{free}R_cE_b}{N_0}}\right), \tag{4.69}$$

ahol d_{free} a turbó kód minimális vagy más néven szabad távolsága, M_{free} a minimális súlyú kódszavak száma, és $\tilde{w}_{free}$ ezek átlagos információs súlya. Mivel a normális eloszlás farokeloszlása igen gyorsan csökken, a képletbeli közelítés nagy E_b/N_0 érték esetén (azaz kis zajú csatornán) igen pontos. Az így kapott közelítést szokás *szabad távolság aszimptotá*nak is nevezni.

4.23. példa. A 4.22. példában megfelelő bitkeverőt alkalmazva $K = 65536$ esetén $d_{free} = 6$, $\tilde{w}_{free} = 2$, $M_{free} = 3$ adódik.

A turbó kódok esetében a szabad távolság általában elég kicsi. Azonban kis jel–zaj viszony és nagy K esetén a minimális súlyú kódszavak száma jobban befolyásolja a teljesítményt, mint a minimális kódtávolság. Pontosan ez jellemzi a



4.38. ábra. A dekódolás hibavalószínűsége és a szabad távolság aszimptota a 4.22. példa esetén Gauss csatornán; álvéletlen bitkeverő, $K = 65536$, $R_c = 1/2$.

turbó kódokat: igaz ugyan, hogy a minimális kódtávolság kicsi, ám a viszonylag kis súlyú kódszavak száma nagyságrendekkel kisebb lehet, mint a hagyományos hibavédő kódok esetén, amelyek a minimális távolság maximalizálására törekednek (lásd a 4.23. példát). Ily módon a turbó kódok távolságprofilja sokkal jobban hasonlít az „optimális” véletlen csatornakódok távolságprofiljára. Ennek következtében a turbó kódok igen jól viselkednek nagyon zajos csatorna esetében, ám ahogy a jel–zaj viszony nő, a hibavalószínűség-görbe hirtelen ellaposodik, és a kis minimális távolság miatt nagy jel–zaj viszony esetén már nagyobb hibát kapunk, mint mondjuk egy jól választott hasonló komplexitású konvolúciós kód esetén (lásd 4.38. ábra).

4.19. Kódok konstrukciója többszörös hozzáférésű kódosztásos csatornákhoz (CDMA)

Tegyük fel, hogy T számú potenciális felhasználó egy közös digitális csatornán kíván kommunikálni. A rendszerben maximálisan M felhasználó lehet egyszerre aktív. Az éppen aktív felhasználók halmaza a potenciális felhasználók halmazának egy véletlen részhalmaza. Általános esetben egy éppen aktivizálódó felhasználó tetszőleges időpillanatban léphet a csatornába, azaz nem tételezhetünk fel időkoordinációt a különböző felhasználók között. Hasonlóan nincs frekvenciakoordináció sem, azaz a teljes rendelkezésre álló sávszélességet használhatják a felhasználók. Következésképp az aktív felhasználók jelei ütközhetnek a csatornában, amely az átvitt üzenetek — legalábbis részbeni — sérülését okozza.

A klasszikus többhozzáféréses esetekben idő- vagy frekvenciaosztást alkalmaznak, s így ortogonális részcsatornákra bontják (időben illetve frekvenciában) a közös csatornát. A mi esetünkben egy más „dimenzióban” végezzük el a különböző felhasználók egyidejű üzeneteinek „ortogonalizálását”. A módszer a kódosztás alkalmazása. Az egyes felhasználóknak saját kódot adunk, amelyekkel megkülönböztetjük őket egymástól. Nyilvánvaló, hogy a kódokat nem lehet akárhogyan megválasztani. A kódokra „ültetett” üzenetek, amelyek az egyes felhasználóktól érkeznek, ütköznek a csatornában (pl. összeadódnak), így a venni kívánt üzenet hasonló struktúrájú jelekkel keveredve (ún. rendszerzajjal) érkezik a vevőbe. A többhozzáféréses kódosztásos rendszerben a kódolás célja eltérő a klasszikus esetekhez képest, ahol elsősorban hibakontroll (javítás, detekció) céljából alkalmaztunk kódolást. A mostani esetben a kódolással címzést is végzünk, azaz megjelöljük vele azt a felhasználót, aki az üzenetet küldi. Más szemszögből nézve viszont, az adott esetben is zajjal, a rendszerzajjal szemben kívánjuk védeni az üzenetet, ilyen értelemben a kódválasztás (adott M felhasználószám mellett) az üzenetek detektálási hibavalószínűségét befolyásolja. A kódszóválasztás módját

azonban nem csak a megkívánt detektálási hibavalószínűség szabja meg, hanem egy legalább ilyen fontos kritérium, a megfelelő szinkronizálhatóság. Említettük, hogy az egyes felhasználók időben függetlenül kezdhetnek kommunikálni, így a vevő első feladata — digitális összeköttetésről lévén szó — bizonyos szinkronizmus (a kódszószinkron) megteremtése önmaga és adója között. E kettős követelménynek egyféle ciklikusan ortogonalitást mutató kódszóhalmaz — a multiplexáló kódszavak halmazának (T kódszó) — kiválasztása tehet eleget.

A '90-es évek kezdetére számos jelentős polgári kommunikációs alkalmazásban is megjelentek a **kódosztásos többszörös hozzáférésű** (CDMA, Code Division Multiple Access) rendszerek (lásd pl. [46]). A **közvetlen sorozatú** (Direct Sequence, DS), a lassú vagy gyors **frekvenciaugratásos** (Frequency Hopping, FH), valamint az **időugratásos** (Time Hopping, TH) rendszerek képezik a **CDMA** rendszerek alaptípusait. A CDMA lényege az, hogy nem tételezi fel az idő- vagy frekvenciatartomány központi szinkronizálását (egy központot).

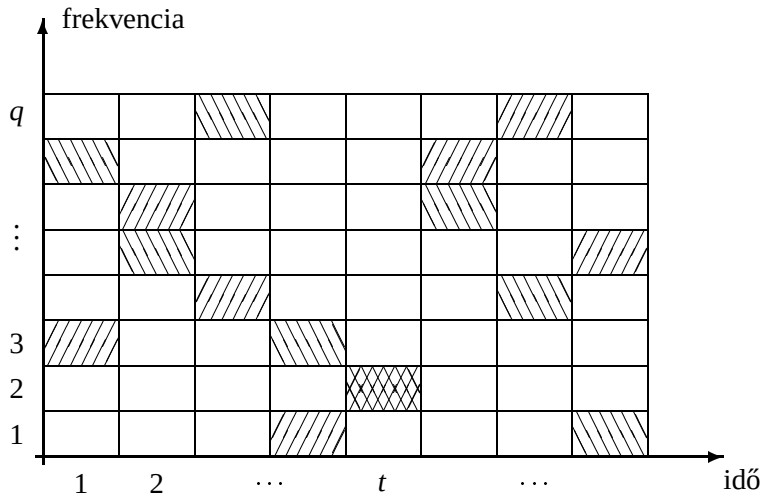
Lassú frekvenciaugratásos (SFH) többszörös hozzáférésű rendszer példáját tekintve a kód T kódszót — itteni terminológiában frekvenciaugratás-sorozatot — tartalmaz. Minden felhasználó számára kiosztunk egy frekvenciaugratás-sorozatot. A felhasználható frekvenciák száma q, ahol tipikusan $T \gg q$. Feltételezzük, hogy az aktív felhasználók információs egységeiket (csomag) ezen sorozatoknak megfelelő frekvenciákon továbbítják. Tegyük fel, hogy az aktív felhasználók ugratássorozat-aszinkronok, de csomag-szinkronok, azaz ha egynél több felhasználó ad azonos időszeletben azonos frekvencián, akkor az ütköző csomagok teljes hosszban ütköznek, s egymást „megsemmisíthetik". Ezt szemlélteti a 4.39. ábra $M = 2$ esetre, ahol a t-edik időszeletben a 2. frekvencián állt elő ütközés.

Lassú frekvenciaugratásos többszörös hozzáférésre példa a GSM, ahol a bázisállomás a híváskor kioszt egy ugrássorozatot.

Minimalizálni kell az elkerülhetetlen csatornabeli ütközések hatását, s ehhez az ütközések számát minimalizáló, T elemű ugratássorozat-halmazt generálunk. Ennek elemeit osztjuk ki egy-egy felhasználónak. Jelölje C^* az n hosszúságú, F ábécé feletti multiplexáló kódszavak halmazát:

$$C^* = \left\{ \mathbf{c}^{(1)}, \mathbf{c}^{(2)}, \ldots, \mathbf{c}^{(T)} \right\}, \qquad \mathbf{c}^{(i)} \in F^n.$$

A C^* kód tehát a többszörös hozzáférésű kódolást végzi, vagy másképp kifejezve a multiplexálást. A multiplexáló kód a „hátán" hordozza az üzenetcsomagokat, amely csomagok minimális számban történő sérülését kívánja a kódolás garantálni az aszinkron többszörös hozzáférésű csatornán való továbbításukkor. Egy további kódolást, törlés- illetve hibajavító kódolást is alkalmazva, lehetőség van arra, hogy a mégis megsérült csomagokat kijavíthassuk a vevőben. A feladat ebben a megközelítésben tehát alapvetően a C^* kód tervezése, mivel a hibakontroll



4.39. ábra. Ütközéses FH csatorna.

kódolás fázis klasszikus, ismert lépés.

A C^* kód hatékonyságát kódszavai ciklikus távolságának minimumával mérjük, amit kódosztásos rendszertervezői szótárból származó ekvivalens korrelációs fogalmakkal is leírhatunk. E célból ciklikus auto- valamint keresztkorrelációs mennyiségeket vezetünk be:

$$\lambda_a = \max_{\mathbf{c}\in C^*} \max_{0<\tau<n} \left| \sum_{t=0}^{n-1} f(\mathbf{c}_t, \mathbf{c}_{t+\tau}) \right|$$

a C^* kód autokorrelációs paramétere, valamint

$$\lambda_c = \max_{\substack{\mathbf{c},\widehat{\mathbf{c}}\in C^* \\ \mathbf{c}\neq\widehat{\mathbf{c}}}} \max_{0\leq\tau<n} \left| \sum_{t=0}^{n-1} f(\mathbf{c}_t, \widehat{\mathbf{c}}_{t+\tau}) \right|$$

a C^* kód keresztkorrelációs paramétere, ahol az indexbeli összeadás modulo n értendő. Az f leképezés választása az F karakterábécétől függően a következő a tipikus rendszerekben: bináris DS rendszerekben, amikor $F = \{+1, -1\}$:

$$f(u, v) = u \cdot v,$$

ahol $u, v \in \{1, -1\}$, a szokásos skalárszorzás korrelációt kapjuk, ha pedig $F = \{1, 2, \ldots, q\}$, q-áris ábécé, akkor az

$$f(u, v) = \begin{cases} 1, & \text{ha } u = v \\ 0, & \text{egyébként} \end{cases}$$

definícióval a **Hamming-korrelációt** kapjuk. Két n karakter hosszú sorozat közti Hamming-korreláció azon karakterpozíciók darabszáma, amelyben a két sorozat azonos karaktereket tartalmaz (azaz a Hamming-távolság „komplemense”). FH rendszerekben $q > 2$.

A C^* kód konstruálása során a

$$\lambda_{\max} = \max(\lambda_a, \lambda_c)$$

értéket szeretnénk minél kisebbre beállítani.

Rendszertervezői szemszögből tekintve kicsi λ_c érték azért szükséges, hogy minimalizáljuk a különböző aktív felhasználók csatornabeli jelei közti interferenciát (rendszerzaj). Hasonlóan a kis λ_a értékre törekvés a vevő kódszinkronizációs képességeit kívánja növelni. A $\lambda_{\max}$ paraméter minimalizálása szimultán szeretné optimalizálni a λ_a és λ_c értékeket.

Eddigi jelöléseinknek megfelelően

$$(n, T, \lambda_{\max}, F)$$

paraméternégyessel jellemezhetjük a C^* kódot, ahol $T = |C^*|$. Két kód egybevetését azonos n szóhossz és F ábécé mellett a T potenciális felhasználószám és a $\lambda_{\max}$ korrelációs paraméter alapján végezhetjük. Ha csak F rögzített, akkor a $\frac{T}{n}$ és $\frac{\lambda_{\max}}{n}$ szóhosszra vett relatív paraméterértékeket vethetjük egybe.

4.24. példa. Tegyük fel, hogy q különböző frekvencia áll rendelkezésre, és csatorna időszeletenként bármelyiket választhatjuk a csatornába adott jel számára, tehát q-FSK modulációt végezhetünk. Ez azt jelenti, hogy nembináris kódolást alkalmazva GF(q) felett dolgozhatunk a csatornakód elkészítésekor, ahol q valamely prímszám vagy prímhatvány.

Minden forrásnak két különböző típusú információt kell a csatornába juttatnia. Az egyik a forrás (felhasználó) azonosítója (címe), a másik az üzenet.

Ezen információk kódolásának egyik módja az alábbi. A forrás bináris sorozatából képezzünk q-szintű karaktersorozatot, s egy kódszóba kódolandó üzenet legyen egy karakter, azaz GF(q) egy eleme. Egy megfelelő kód ezen üzenet kódolására egy $(q-1, 1)$ paraméterű GF(q) feletti RS-kód. Ekkor a $d = n - k + 1$ összefüggés alapján $d = q - 1$ adódik a minimális kódszótávolságra, azaz tetszőlegesen kiválasztott két kódszó minden koordinátában különbözik egymástól. A kód generátorpolinomja lehet a

$$g(x) = (x-1)(x-\alpha)\cdots(x-\alpha^{q-3}),$$

ahol α a GF(q) primitív eleme. Könnyen belátható az

$$(x-\alpha^{q-2})g(x) = x^{q-1} - 1$$

felhasználásával, hogy

$$g(x) = x^{q-2} + \alpha^{q-2}x^{q-3} + \alpha^{q-3}x^{q-4} + \cdots + \alpha^2 x + \alpha,$$

ha $q = 2^l$ alakú. Így a kódszavakat az $(1, \alpha^{q-2}, \alpha^{q-3}, \ldots, \alpha)$ vagy — ekvivalens kódot használva — az $\mathbf{A} = (1, \alpha, \alpha^2, \ldots, \alpha^{q-2})$ GF(q) feletti vektornak az üzenettel (GF(q) elemmel) történő szorzása generálja. A különböző üzenetekhez rendelt kódszavak az $\mathbf{A}$ vektor elemeinek ciklikus eltolásával generálhatók. Innen is látható, hogy a kódszavak egymástól azonos, $q-1$ távolságra vannak.

Ennél a fajta kódolásnál minden felhasználó azonos kódot használ az üzenetek kódolására, tehát ez multiplexálásra nem jó. Egy újabb kódolással a forrás azonosítójával is megjelöljük az egyes források fenti módon kódolt üzeneteit.

Használjuk az i-edik forrás azonosítójául az

$$\alpha^{i-1}\mathbf{1} = (\alpha^{i-1}, \alpha^{i-1}, \ldots, \alpha^{i-1})$$

$n \leq q-1$ hosszú vektort, $i = 1, 2, \ldots, q-1$, és az azonosítónak az üzenetkódszóra történő „rákódolását” a

$$\mathbf{c}^{(i,\beta)} = \beta\mathbf{A} + \alpha^{i-1}\mathbf{1} \tag{4.70}$$

formula felhasználásával végezzük, ahol $\beta \in$ GF(q) a pillanatnyi üzenet. Maximálisan $q-1$ felhasználó lehet egy rendszerben, hiszen ennyi azonosítót oszthatunk ki. A rendszerben kódszószinkront feltételezve az i- illetve j-edik felhasználó $\mathbf{c}^{(i,\beta)}$ illetve $\mathbf{c}^{(j,\gamma)}$ kódszavai legfeljebb egy koordinátájukon egyezhetnek meg, azaz a különböző felhasználók kódszavainak minimális távolsága $q-2$. Aszinkron esetben, amikor a kódszavak átfedésben vannak, a $\mathbf{c}^{(i,\beta)}$ kódszó a vele átfedődő $\mathbf{c}^{(j,\gamma)}$ és $\mathbf{c}^{(j,\delta)}$ kódszófelekkel összesen legfeljebb két koordinátáján ütközhet.

A kódszavak ábrázolására idő $\times$ frekvencia mátrix használatos, amely egy $(q-1, q)$ méretű bináris mátrix az adott esetben, s azon eleme 1 értékű, amely koordinátáknak megfelelő cellát átvitelre felhasználunk.

Általában nem küldik át a teljes kódszót, hanem néhány koordinátát törölve rövidítik azt. A $\mathbf{c}^{(i,\beta)}$ kódszó kifejezését tekintve, láthatóan nem vesztünk információt az (α^{i-1}, β)-ra nézve ezzel a rövidítéssel. A kódszó rövidítésével viszont csökkentjük a többi forrás kódszavai okozta rendszerzajtól való megkülönböztethetőségét. Adott bit/sec sebességű forrás csatornakódolása ilyen kis sebességű kóddal, a frekvenciatartományban nagy sávszélességnövelést jelent. (Emiatt ezen kódosztásos rendszerekre a **szórt spektrumú** (spread-spectrum) elnevezést is használják.)

A fentiekben leírt kódolásra vegyünk egy kisméretű illusztratív számpéldát, ahol legyen $q = 8$. Legyen $\alpha \in$ GF(8) a primitív elem, amelyre $\alpha^3 + \alpha + 1 = 0$.

Alkalmazzuk az $i \leftrightarrow \alpha^{i-1}$ megfeleltetést a nemzérus testelemekre, ekkor (4.70) alapján a 3 karakterrel rövidített kódszavak a következők:

$$(0,0,0,0) \qquad (1,2,3,4) \qquad (2,3,4,5) \qquad (3,4,5,6)$$

$$(4,5,6,7) \qquad (5,6,7,1) \qquad (6,7,1,2) \qquad (7,1,2,3)$$

Például, ha a harmadik forrás az 5 üzenetet kívánja átvinni, akkor a

$$\mathbf{c}^{(3,5)} = (5,6,7,1) + (3,3,3,3) = (2,4,1,7)$$

kódszót küldi el.

A dekódolás a kódolás inverz művelete. Helyezzük el a források pillanatnyilag küldött kódszavait az idő $\times$ frekvencia mátrixba. Legyen pl. $M = 3$ és $\mathbf{c}^{(1,2)} = (4,7,2,6)$, $\mathbf{c}^{(2,7)} = (6,4,0,5)$, $\mathbf{c}^{(3,5)} = (2,4,1,7)$ az egyes kódszavak, ekkor a dekódolás inputja az alábbi bináris vett mátrix:

$$\begin{array}{l} \text{frekvencia} \uparrow \\ \\ 7 \\ 6 \\ 5 \\ 4 \\ 3 \\ 2 \\ 1 \\ 0 \end{array} \quad \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \quad \longrightarrow$$

idő

Az idő $\times$ frekvencia mátrix nemzérus elemeit oszlopaikon belül permutáljuk. Először a nemzérus elemek sorindexéből (mint GF(8) testelemből) kivonjuk a dekódolni kívánt üzenet forrásának azonosítóját, majd az eredményül kapott mátrix nemzérus elemeinek sorindexét oszloponként, rendre az $1, \alpha^{-1}, \alpha^{-2}, \ldots, \alpha^{-(q-j-2)}$ elemekkel szorozzuk. Ezen műveletekkel valóban a kódolás inverzét végeztük el. A műveletek eredményeképpen kapott mátrix azon sorában, amely a venni kívánt felhasználó pillanatnyi β üzenetének felel meg, csupa 1 áll.

Számpéldánkat folytatva az idő $\times$ frekvencia mátrix a dekódolás során az

alábbi alakra transzformálódik az első felhasználó üzenetének dekódolása során:

$$\begin{array}{cc} 7 & \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix} \\ 6 & \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix} \\ 5 & \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} \\ 4 & \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} \\ 3 & \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix} \\ 2 & \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix} \\ 1 & \begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix} \\ 0 & \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix} \\ & \\ & \begin{matrix} 0 & 1 & 2 & 3 \end{matrix} \end{array}$$

Valóban $(1,1,1,1)$ áll a 2. sorban, amely a $\beta = \alpha$ üzenetnek felel meg. A dekóder feladata tehát megállapítani, hogy melyik sor tartalmaz csupa 1-est.

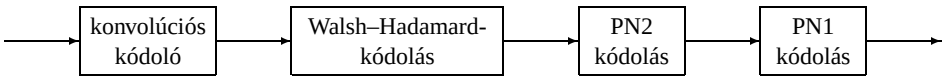
Mivel egyszerre több felhasználó is küld a csatornába, ezért előfordulhat, hogy egynél több sor is csupa 1-esből áll, s ekkor a dekódoló véletlenszerű választásra van utalva ezen sorok között, azaz dekódolási hiba keletkezhet. Mivel az adott kódolásnál a különböző felhasználók kódszavai a mátrix egy adott sorába csak egy 1-est adhatnak, ezért μ karakteres rövidítés esetén ha $M \leq q - \mu - 2$, akkor hibamentes átvitel biztosítható elvileg. Számpéldánkban ez az $M \leq 3$ feltételt jelenti.

Példánkban kódszószinkron esetet tekintettünk hallgatólagosan, míg a gyakorlati eset a kódszavak aszinkronizmusa. A fenti kódkonstrukció ebben az esetben is jól használható, de aszinkronizmus esetén növekszik a kódszótévesztés valószínűsége a kódszóátlapolódás miatt.

Celluláris mobilis CDMA rádiótelefónia példája

Az IS-95 CDMA rendszer:
A CDMA elvének legkiteljesedettebb polgári alkalmazását jelenti a celluláris mobilis CDMA rádiótelefónia. A CDMA megoldás igazi előnyei celluláris rádiótelefónia alkalmazásban a hálózati funkciók alaposabb tanulmányozása és megértése kapcsán derülnek ki, amelynek részleteire (helyszűke miatt) itt nem tudunk kitérni. A fizikai szintű vonatkozások, mint a moduláció, kódolás, szinkronizáció, megoldásaik paraméter-értékeiben ugyan lehetnek speciálisak a CDMA rendszerekben, de alapvetően nem különülnek el a vezetéknélküli digitális kommunikációs rendszerek alapvető problémáitól, megoldásaitól.

Ugyanakkor a celluláris CDMA hálózati szintű koncepciójának alapvető speciális elemei vannak, s nagyrészt ezeken múlik a celluláris CDMA jelentős kapa-



4.40. ábra. A celluláris CDMA kódolás tömbvázlata.

citáselőnye a konvencionális rendszerekhez képest.

A celluláris CDMA összetett és hatékony kódolási rendszere egyidejűleg megoldja a

a) különböző egyidejű, azonos frekvenciasávban kommunikáló jelek kódolással történő multiplexálását (CDMA),

b) a zajos, fadinges (véletlenszerű jelcsillapodásos) csatornán történő átvitel minőségét biztosító hibakontroll kódolást,

c) privát (kommerciális szinten védett) kommunikáció biztosítását.

A CDMA CAI [1] specifikálja a CDMA jelalakok tervezését, köztük a frekvenciaosztást, az álvéletlen sorozatokkal történő kódosztást, és az ortogonális jelmultiplexálási technikát.

A mobil $\longrightarrow$ bázisállomás (uplink) és a bázisállomás $\longrightarrow$ mobil (downlink) csatornában hasonló jelképzési elvek érvényesülnek. Vannak azonban rendszertechnikai különbségek, ezért a jelképzés tömbvázlat jellegű áttekintését külön végezzük a két kommunikációs irányra.

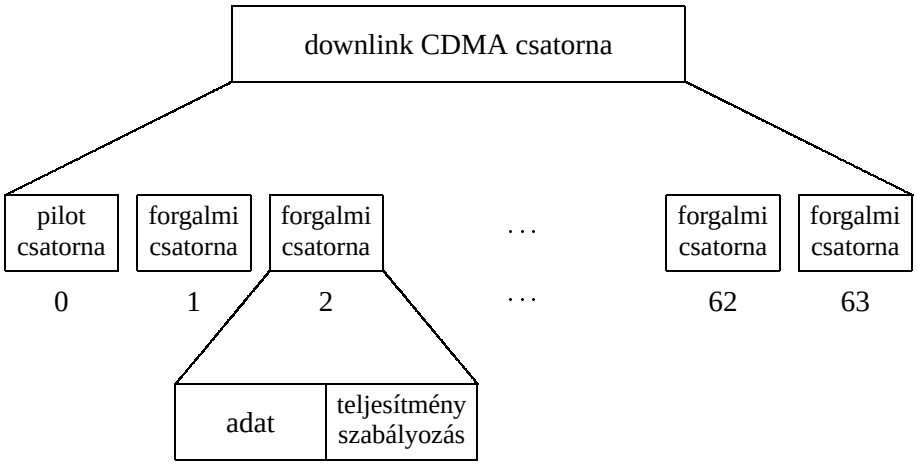
Bázisállomás $\longrightarrow$ mobil csatorna:

A kódolási rendszer elemei:

- konvolúciós kódoló (hibajavítás)
- Walsh–Hadamard-kódolás (multiplexálás egy szektoron belül)
- PN2 kódolás (privát kommunikáció)
- PN1 kódolás (multiplexálás különböző szektorok, cellák között)

A tömbvázlatot a 4.40. ábrán láthatjuk. Egy cella egy szektrorán belüli aktív mobilok megkülönböztetése (jeleik kódmultiplexálása) bináris ortogonális kódok (Walsh–Hadamard-kód) alkalmazásával történik. A tipikus kódszóhossz 64, az ortogonális kódszavak száma 64. Ezzel 64 csatorna képezhető egy szektoron belül, amelyből egy a pilotcsatorna (jelzésátvitel) céljára van lefoglalva.

A koherensen (azaz vivőfázis-helyesen), kódszinkronban működő egy szektoron belüli downlink jeleket az ortogonális kódok alkalmazása hatékonyan szeparálja.



4.41. ábra. Ortogonális kódmultiplexálásos csatornaképzés a downlink csatornában.

A k paraméterű **Walsh–Hadamard-kód** kódszavainak halmaza a 2.5. szakaszban már megismert egyszerű rekurzív konstrukcióval állítható elő. A generált $C(2^k, k, 2^{k-1})$ kód szavait az $\mathbf{A}_{2^k}$ mátrix soraiból képezik. Illusztráció kedvéért az alábbiakban megadjuk a 8 elemű Walsh–Hadamard-kód elemeit ($+1$ amplitúdó $\rightarrow$ 0 bit, -1 amplitúdó $\rightarrow$ 1 bit a megfeltetés):

1	2	3	4	5	6	7	8	
1	1	1	1	1	1	1	1	1. kódszó
1	-1	1	-1	1	-1	1	-1	2. kódszó
1	1	-1	-1	1	1	-1	-1	3. kódszó
1	-1	-1	1	1	-1	-1	1	4. kódszó
1	1	1	1	-1	-1	-1	-1	5. kódszó
1	-1	1	-1	-1	1	-1	1	6. kódszó
1	1	-1	-1	-1	-1	1	1	7. kódszó
1	-1	-1	1	-1	1	1	-1	8. kódszó

Bármely két kódszó skaláris szorzata zérust eredményez, mivel fele koordinátán egyező, felén ellenkező előjelűek az elemek.

Álvéletlen (pszeudorandom, PN1) sorozatokkal történő kódolás segítségével különböztetik meg a különböző szektorok és cellák mobiljainak jeleit a downlink csatornában, azaz ezen az úton történik a többhozzáféréses fázisugratásos kódmultiplexálás. A Walsh–Hadamard-kód fizikai sebessége megegyezik az úgynevezett chip-sebességgel, azaz a PN1 kódolás ütemének megfelelő.

A különböző cellák és szektorok saját PN1 kódjai egy alapkódfázis különböző időtolásaival állnak elő, míg egy cella egy szektorában az összes aktív mobil PN1 kódja azonos időfázisú. Az időtolás megoldásnak a működőképessége a PN1 kódok autokorrelációs tulajdonságán alapszik, miszerint ha az időtolás az alapkód két származtatottja között legalább egy chip-idő (chip = elemi jel, a PN1 kód egy eleme), akkor elegendő hosszú időablakban korrelációt képezve, a korreláció értéke az időablak hosszával a zérushoz tart.

A PN1 kódokat lineárisan visszacsatolt shiftregiszteres (LFSR) generátorokkal állítják elő. A generált sorozat periódushossza 32768 chip-idő. A PN1 chipsebesség 1.2288 MHz, amivel pontosan 128 chip esik a 9600 bit/s sebességű forrás minden bitjére. A kvadratúra moduláció két ágán két különböző generátorú PN sorozatokkal kódolnak, azaz négyfázisú PN1 modulációt alkalmaznak.

A rendszer használ egy igen nagy periódusú (periódushossz = $2^{42} - 1$) PN2 sorozatot is, amelyet mobilcímhez választott egyedi fázistolással rendelnek az egyes felhasználókhoz. Mivel minden egyes eltoláshoz egy cím tartozhat, ez igen nagy felhasználószám melletti megkülönböztetést tesz lehetővé. A chip-sebességgel futó PN2 sorozat chip-időbeli XOR-ral one-time-pad jellegű, kommerciális körülmények közötti bizalmasító (confidential) mértékű védelmet nyújt.

A downlink jelalaktervezés nagyban támaszkodik a bázisállomástól sugárzott pilotjel jelenlétére, ami lehetővé teszi a koherens vételt. A viszonylag nagyszintű pilotjel igen pontos követést tesz lehetővé a mobilok szinkronizáló egységei számára. A pilot csatornabeli jelképzés módja olyan, mintha egy külön adatcsatorna lenne, amelyben csupa 0 „adat” kerül továbbításra, azaz csupa 0 Walsh–Hadamard-kódszó (64 db 1) ismétlése „ül” a PN1 kód kvadratúra párján.

A mobil szinkronizál a PN1 kódra, keresve azt a fázishelyzetet, ahol a maximális autokorreláció adódik a vett és az általa generált sorozat között. Végigkeresi a teljes periódust (32768 chip), s a legerősebb autokorrelációs értéknek megfelelő fázishoz tartozó bázisállomást tekinti legközelebbi bázisállomásnak. Miután megtörtént ezen szinkronizálás a legközelebbi bázisállomás pilotjele PN1 kódjára, a következő lépés a vivőszinkronizálás a pilotjel felhasználásával. Ezen szinkront tartva a megfelelő Walsh–Hadamard-kódszó alkalmazásával a forgalmi csatornákon (max. 63) a mobil alkalmas a PN1 + PN2 + Walsh–Hadamard-kódolás lefejtése után a hibakontroll konvolúciós kód dekódolására.

Mobil $\longrightarrow$ bázisállomás csatorna:

Az uplink csatornában a Walsh–Hadamard-kód alkalmazása másképpen történik, mint a downlink csatornában. Nevezetesen a downlink csatornában a bázisállomás fixen egy-egy ortogonális csatorna képzéséhez használja fel a 64 kódszóhosszú, 64 kódszóból álló Walsh–Hadamard-kód egy-egy kódszavát, az uplink csatornában viszont az üzenetbitek 6 bites csoportjai folyamatosan címzik meg a

megfelelő Walsh–Hadamard-kódszót, azaz az nem fix, hanem a bitfolyamtól függően változó. A módszer a szuperortogonális konvolúciós kódolás. Úgy is fogalmazhatunk, hogy ezzel 64 szintű ortogonális jelkészletű modulációt valósítunk meg. Ezzel a módszerrel egy igen jó minőségű csatorna alakítható ki fadinges körülmények között, alacsony jel/zaj viszony mellett anélkül, hogy pilot referencia jelre kellene támaszkodni. (Erre a kódolóra a tömbvázlat további elemeinek rövid ismertetése után részletesen visszatérünk.)

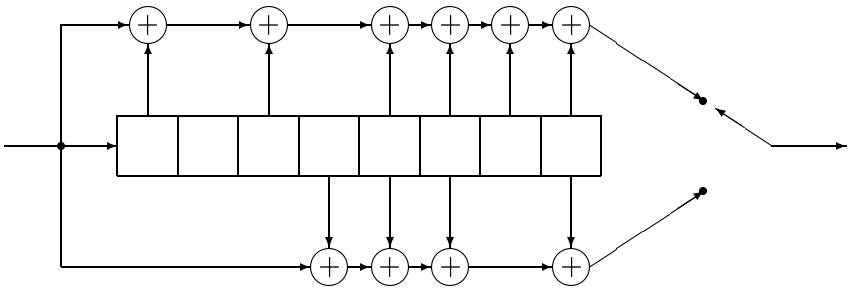
Az uplink csatornában ugyancsak alkalmazásra kerülnek a PN1 és PN2 álvéletlen kódok.

A kódolási tömbvázlatunk ezzel még nem teljes, mivel a CDMA kódrendszer igen erőteljesen támaszkodik a kódátfűzés alkalmazására is. Ha a vételi jelteljesítmény időben változik, ezzel együtt változik a demodulátor bemenetén a bitenergia, következésképpen a jel/zaj viszony, ezáltal a demodulációs (nyers) bithibaarány is. Javuláshoz vezethet megfelelő diverzity alkalmazása. (Diverzity rendszerben ugyanazt az információt két különböző csatornán viszik át. Működésének alapja az, hogy kisebb annak a valószínűsége, hogy egyidejűleg mindkét csatornán rossz átviteli függvény alakul ki, mint annak, hogy a kettő közül csak az egyiken.) A CDMA rendszerben idődiverzityt valósítanak meg kódátfűzés felhasználásával. Blokkátfűzés esetén ez azt jelenti, hogy $I \times J$ méretű blokkba gyűjtjük az együtt átfűzendő $I \cdot J$ szimbólumot (chip, bit, szó, stb. egységet), soronként egymásután töltve fel a blokkot. Ezután oszloponként olvassuk ki a csatornába a blokk elemeit. A vevőben ennek inverzét elvégezve az eredeti sorrendet hozzuk létre.

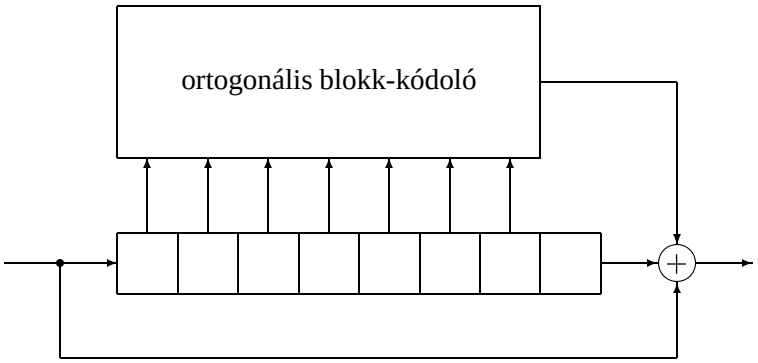
Ezáltal a csatornában időben szomszédos szimbólumok az átfűzés visszafejtése után J szimbólumidő távolságra kerülnek. Ennek az a haszna, hogy J szimbólumidőnél kisebb átlagos fadingidőtartamok esetén fadinghatás szempontjából független energiákkal kerülnek szomszédságba a szimbólumok, azaz létrejön az idődiverzity.

A 4.42. ábrán látható a $K = 9$ kényszerhosszú (regiszterhossz + 1), $\frac{1}{2}$ kódsebességű, 753 és 561 generátorú (oktális ábrázolás), $d_\infty = 12$ szabad távolságú bináris konvolúciós kód generátora, amely a downlink csatorna hibajavító kódolója.

A megcsapolások helyeit az oktális ábrázolású $G_1 = 753$ és $G_2 = 561$ generátorok adják. Ez azt jelenti, hogy bináris ábrázolásban 110 101 111 (357) valamint 100 011 101 (165) 1-eseinek megfelelő pozíciókban csapoljuk meg a léptetőregisztert. (A rendszerben használatos még $\frac{1}{3}$ sebességű kódoló is, amely $d_\infty = 18$ szabad távolságú, $K = 9$ kényszerhosszú, $G_1 = 711, G_2 = 663, G_3 = 557$ megcsapolás generátorokkal rendelkezik. Az említett $\frac{1}{2}$ illetve $\frac{1}{3}$ sebességű kódok a lehető legnagyobb szabad távolsággal rendelkeznek az adott kényszerhossz mellett.)



4.42. ábra. $\frac{1}{2}$ kódsebességű konvolúciós kódoló.



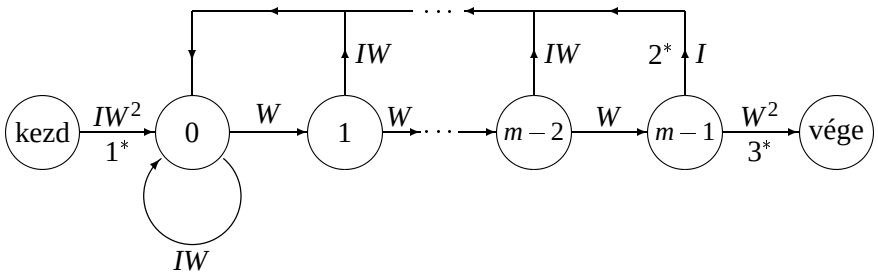
4.43. ábra. A szuperortogonális konvolúciós kódoló.

A 4.43. ábrán láthatjuk a **szuperortogonális konvolúciós kódoló** blokkvázlatát.

Az m bites léptetőregisztert $m-1$ belső pontján megcsapoljuk, s ez az $m-1$ bites blokk címez meg egy Walsh–Hadamard-kódszót, amelynek kódszóhossza 2^{m-1}. A 4.43. ábrán ezt a kódszókiválasztást végzi az ortogonális blokk-kódoló. A szuperortogonális kód tehát egy bináris konvolúciós kód és egy ortogonális blokk-kód hibridje. A szuperortogonális konvolúciós kódoló inputján egy bit belépésekor kiszámítódik a shiftregiszter állapotának megfelelő ortogonális kódszó, s ehhez hozzáadódik a bemeneti (b_1) és a shiftregiszter aktuális kimeneti (b_2) bitjének 2^{m-1}-szeres ismétlésével kapott szó, azaz

$$\text{kimeneti szó} = \text{ortogonális kódszó} \oplus (b_1, b_1, \ldots, b_1) \oplus (b_2, b_2, \ldots, b_2).$$

Ha tehát $b_1 \neq b_2$, akkor a kimeneten az ortogonális kódszó bitenkénti invertáltja jelenik meg.



4.44. ábra. A redukált állapotátmenet-gráf.

A kódoló kicsi sebességű, azaz egy bit belépésekor a kimeneten 2^{m-1} bites szó jelenik meg, $R = \frac{1}{2^{m-1}}$.

A szuperortogonális kód figyelemre méltó tulajdonsága egyrészt viszonylag könnyű analizálhatósága, másrészt kiváló hatékonysága (mindennek ára a kis sebesség). Az ortogonális szavak párjai pontosan fele bitjükben különböznek, tehát a zérus súlyú vektortól pontosan $v = 2^{m-2}$ bitben különböznek. A redukált állapotátmenet-gráf látható a 4.44. ábrán, amely jól követhető a kódoló blokksémája alapján. A kezdeti és végállapot a regiszter csupa 0 állapotának felel meg (hasonlóan ahhoz, ahogy a 4.30. ábra kapcsán eljártunk). A kezdeti állapotból egy 1 bit beléptetésével lépünk ki. A kezdeti és végállapot közötti i-edik állapot ($i = 0, 1, \ldots, m-1$) annak felel meg, hogy a regiszterben milyen hosszú csupa 0 bitből álló sorozat található az input oldali cellákat tekintve.

A 4.18. példában megismert transzfer függvény számításának egy gyakorlati példáját mutatjuk be ennek kapcsán. Az ott megismert jelölésekkel a blokksémában a

$$W = D^v = D^{2^{m-2}} \quad \left(= D^{1/(2R)}\right)$$

jelölést használtuk, továbbá I szorzó azon az élen áll, amely az 1 adatbit kódolóba léptetésével generálódott, míg az I szorzó hiánya W előtt azt jelenti, hogy az él 0 adatbit kódolóba léptetésével generálódott.

$*$-gal jelöltük meg a három, külön magyarázatot igénylő élet. Az 1^* él 1 adatbit beléptetésekor keletkezik az azelőtt zérus állapotú kódolóban, azaz a zérus útból leágazó első él. A kódoló vázlata alapján jól látható, hogy ekkor ennek a bitnek a hatásaként a csupa 1 kódszó jelenik meg a kimeneten, azaz amelynek súlya $2v\ (= 2^{m-1})$ (a W^2 jelölés pontosan ezt fejezi ki, mivel W definíciója szerint a súly a kitevőben szerepel). Hasonló a magyarázat a 3^* él esetén is, csak ott a hatást a kilépő 1 bit okozza. A 2^* él esetén a belépő és kilépő bit 1, mod 2 összegük 0, s az ennek megfelelő csupa 0 szó ($v = 0$ súlyú szó) jelenik meg a kimeneten.

A fenti redukált folyamatgráf alapján felírható transzfer függvény a következő:

$$T(W,I) = \frac{IW^{m+3}}{1 - I(W + W^2 + \cdots + W^{m-2} + 2W^{m-1})}.$$

Innen a bithibaarány összefüggés

$$P_b < \left.\frac{\mathrm{d}T(W,I)}{\mathrm{d}I}\right|_{I=1,D=Z} = \frac{Z^{(m+3)v}}{(1-2Z^v)^2}\left(\frac{1-Z^v}{1-Z^{(m-1)v}}\right)^2,$$

ahol Z az egy bináris csatornaszimbólum dekódolási hibavalószínűségére vonatkozó Bhattacharyya felső becslés:

$$Z = \int_{-\infty}^{\infty} \sqrt{p_0(y)p_1(y)}\,\mathrm{d}y.$$

Emlékezetnélküli csatorna esetén, ha két csatornabeli szó m csatornabitben különbözik, akkor annak valószínűsége, hogy téves lesz a döntés, Z^m-mel felülről becsülhető. Gaussi csatorna esetén

$$Z = \exp\left(\frac{-E_s}{N_0}\right),$$

ahol E_s a csatornabitre jutó energia, azaz $E_s = RE_b$.

Az UMTS rendszer:
Az UMTS (Universal Mobile Telecommunications System) a harmadik generációs nyilvános mobil távközlő rendszercsalád (IMT2000, International Mobile Telecommunications) európai tagja [23]. Fontosabb jellemzőit 1998-ra rögzítették, és azóta évről évre ún. ReleaseXXXX formában teszik közzé a szabványosítás aktuális állapotát.

Az UTRA (UMTS Terrestrial Radio Interface) az UMTS földfelszíni rádiós interfész csatornahozzáférési technológiája a szélessávú DS-CDMA. A rádiós interfész támogatja mind a frekvenciaosztásos (FDD, Frequency Division Duplex), mind az időosztásos (TDD, Time Division Duplex) üzemmódot. FDD esetén a felhasználótól a cellaközpont irányában (uplink), illetve a cellaközponttól a felhasználók irányában (downlink) a kommunikáció két külön frekvenciasávban folyik. TDD esetén azonos frekvenciasávban időben kerül a két irány szétválasztásra.

A rendszer beszédátvitelt és nagysebességű adatátvitelt biztosít, széles sebességtartományban nyújt szolgáltatást a felhasználónak. A felajánlott bitsebességek a 16–2048 kbit/sec tartományban 2 hatványainak megfelelő értékek lehetnek. Különböző szolgáltatási minőségeket (QoS, Quality of Service) tud felajánlani a

rendszer, mely le kívánja fedni a klasszikus mobil szolgáltatásokat, de az épületen belüli adat- és beszédkommunikációt is. Mobil felhasználók esetén a makrocellában 500 km/h sebességig 144 kbit/sec adatsebességet, mikrocellában 120 km/h sebességig 384 kbit/sec adatsebességet, pikocellában 10 km/h sebességig 2048 kbit/sec adatsebességet tud felajánlani a rendszer.

Az UMTS kódolási alrendszere komplex és variábilis, igazodik a különböző szolgáltatásokhoz (sebesség, QoS). Az UMTS rendszerben használt csatornakódok két alapvető funkciója a többhozzáféréses kódosztásos csatornaképzés illetve a „klasszikus" hibakontroll (hibajavítás, hibadetekció) kódolás. Az UMTS kódolási rendszere sok rokon vonást mutat az IS-95 CDMA rendszerével, de alapvetően annál gazdagabb, részletezőbb a sokféle szolgáltatás kapcsán.

Hibajavító kódok céljára konvolúciós kódot, Reed–Solomon-kódot, turbó kódot használnak. A kódok sebessége a szolgáltatáshoz illeszkedően különböző lehet. A tipikusan csomós hibájú kommunikációs csatornákon kódátfűzés (interleaving) alkalmazása alapvető kódkombinációs technika. A különféle átviteli sebességekhez, adott kerethosszak mellett különböző kódsebességek szükségesek. A sebességek ezen illesztésének fontos kódmódosító technikája a kódszóbitelhagyás (puncturing). Konvolúciós kódok tipikus paraméterei: $R = 1/2, R = 1/3$ sebességek, valamint $K = 9$ kényszerhossz. A beszédcsatorna (általában valósidejű vagy interaktív szolgáltatások csatornája) hibajavítása esetén, ahol a késleltetésmegszorítás miatt rövidebbek a kódszóhosszak, kevesebb komponens átfűzése engedhető csak meg, a szerencsére tipikusan nagyobb megengedhető dekódolási bithibaarányok (BER, Bit Error Rate) rovására. Nem valósidejű szolgáltatások esetén, mint például adatátvitel, az alacsony dekódolási bithibaarány a kritikus szempont, ezért nagyobb kódátfűzés, általában nagyobb blokkméretek, hatékonyabb javítás alkalmazható.

Példaként tekintsük egy kissebességű beszédcsatorna, illetve egy nagysebességű adatcsatorna hibakontroll kódolásának lépéseit és paramétereit. FDD módban az úgynevezett DCH (Dedicated Channel) radio unit előállítása a következőképp történik: a 8 kbit/sec sebességű G.729 beszédkódoló 80 bitet állít elő 10 msec hosszú időkeretben. Ehhez a 80 bithez 16 bit CRC kerül kiszámításra, amely kódolási lépés során 96 bites, hibadetekcióra alkamas kódszót nyerünk. Az $R = 1/3, K = 9$ paraméterű konvolúciós kódoló 8 bites shiftregiszterének nullázása (blokk-kódoló mód) céljára 8 bitet (tail) illesztünk a 96 bites szóhoz, amivel 104 bites kódolandó üzenet kapunk, majd ennek konvolúciós kódolásával $3 \cdot 104 = 312$ bites kódszóhosszra jutunk. Ezen 312 bites kódszavak kerülhetnek ezután átfűzésre.

TDD módban a 2048 kbit/sec sebességű adatcsatorna hibakontroll kódolásának lépései a következőe: 10 msec hosszú időkeretben ezen sebesség mellett

20480 adatbit keletkezik. $R = 200/210$ sebességű Reed–Solomon-kódolással $20480 \cdot 200/210 = 21504$ bites kódszót kapunk. Ehhez X számú jelzésbitet, valamint $16 \cdot 8 = 128$ nullázó bitet (tail) illesztünk, amivel $21504 + X + 128$ bites szót nyerünk. 16 komponensre szétválasztva ezen szót, s $R = 2/3$ sebességű konvolúciós kódolást alkalmazva $(21504 + X + 128) \cdot 3/2 = 32245 + 3X/2 + 192$ bites kószót kapunk. Ebből kódszóbitelhagyás művelettel Burst 1 képzése esetén $3900 + 3X/2$ bitet, Burst 2 képzése esetén $3744 + 3X/2$ bitet elhagyva, kapjuk a végső szóhosszakat, amelyek 28548 bit (Burst 1), illetve 28704 bit (Burst 2) méretet jelentenek.

Hasonlóan, mint az IS-95 CDMA esetén, a kódosztásos többhozzáférése csatornaképzés kódolási eljárásai egyrészt a saját cellából, illetve a környező cellákból származó egyidejű, azonos idő- és frekvenciacsatornában található interferáló jelek (MAI, Multiple Access Interference) csillapításra (scrambling codes), másrészt ortogonális csatornák képzésére szolgálnak (channelization codes). Az utóbbi, a csatornaképző kódok — az IS-95 CDMA rendszernél elmondottakhoz hasonlóan — Walsh–Hadamard-kódra épülnek, amelyből azonban az UMTS esetén ún. Orthogonal Variable Spreading Factor (OVSF) kódcsaládot képeznek. Ezen kódokat egy ún. kódfából származtatják, ahol a gyökértől a levelek felé haladva egyre hosszabb kódok találhatóak. Jentőségük, hogy nemcsak az azonos, hanem a különböző hosszúságú kódok is ortogonálisak egymásra, ezáltal kiválóan alkalmasak a különböző adatsebességű csatornák megkülönböztetésére, miközben az eredő bitsebességet konstanssá teszik.

Downlink irányban pedig az egyazon cellában tartózkodó felhasználók jeleinek ortogonalizálása oldható meg OSVF kóddal. Mivel az OSVF kódok halmazának mérete nagy felhasználói populációkhoz kicsi, ezért azok cellánként újrafelhasználásra kerülnek. Az OSVF kódok kódszószinkron esetén nyújtják az ortogonalitás tulajdonságát, aszinkron esetben azonban igen kedvezőtlen a korrelációs viselkedésük. Következésképpen a szomszédos bázisállomások által sugárzott OSVF kódok minden további intézkedés nélkül elviselhetetlen zavarjelet generálnának. Megoldásul az IS-95 CDMA PN1 kódjának az itteni terminológia szerinti scrambling kódok szolgálnak. Downlink irányban $2^{41} - 1$ szóhosszú, hosszú Gold-kódok különböző szeleteit osztják ki a különbözo celláknak, amely szeletek kedvező aszinkron keresztkorrelációs tulajdonsággal rendelkeznek.

Uplink irányban minden felhasználó használhatja ugyanazt a csatornaképző kódot. Ezen irányban mind a $2^{41} - 1$ szóhosszú Gold-kódok szeleteit, mind a 256 bites szóhosszú Kasami nagy halmaz (VL-Kasami, Very Large set) szavait használják. Ezen rövidebb szóhosszú kód alkalmazása jelentősen egyszerűsíti a többfelhasználós detekció (multiuser detection) algoritmus implementálását a bázisállomáson.

4.20. Feladatok

Lineáris blokk-kódok

4.1. feladat. Egy $\{0,1,2\}$ kódábécéjű GF(3) feletti lineáris kód generátormátrixa:

$$\mathbf{G} = \begin{pmatrix} 1021 \\ 0122 \end{pmatrix}$$

Adja meg a kódszavakat, valamint a d minimális távolságot!

4.2. feladat. Egy lineáris bináris kód paritásellenőrző mátrixa $\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$. Adja meg a kód következő paramétereit: n, k, d, kódszavak száma!

4.3. feladat. Egy lineáris bináris blokk-kód generátormátrixa:

$$\mathbf{G} = \begin{pmatrix} 10110 \\ 01101 \end{pmatrix}$$

Adja meg

a) a kód paramétereit: n, k, d,

b) standard elrendezési táblázatát,

c) szindróma dekódolási táblázatát,

d) a kódszó dekódolási hibavalószínűséget emlékezetnélküli BSC(p) esetére!

4.4. feladat. Egy lineáris bináris kód paritásellenőrző mátrixa:

$$\mathbf{H} = \begin{pmatrix} 11100 \\ 10010 \\ 11001 \end{pmatrix}$$

Adja meg a szindróma dekódolási táblázatot!

4.5. feladat. Egy lineáris bináris kód generátormátrixa az alábbi:

$$\mathbf{G} = \begin{pmatrix} 101011 \\ 011101 \\ 011010 \end{pmatrix}$$

Adja meg:

a) egy ekvivalens szisztematikus kód generátor- és paritásellenőrző mátrixát,

b) a duális kód kódszavait (duális kód = a paritásellenőrző mátrix mint generátormátrix által generált kód).

4.6. feladat. Adja meg a GF(4) feletti $C(4,2)$ paraméterű,

$$\mathbf{G} = \begin{pmatrix} 1022 \\ 0112 \end{pmatrix}$$

generátormátrixú kód szindróma dekódolási táblázatát! ($0 \leftrightarrow 0,\ 1 \leftrightarrow 1,\ 2 \leftrightarrow x,\ 3 \leftrightarrow x+1$)

4.7. feladat. Definiáljon egy $(5,3)$ paraméterű GF(4) feletti kódot a generátormátrixa, amely

$$\mathbf{G} = \begin{pmatrix} 10011 \\ 01012 \\ 00113 \end{pmatrix}.$$

a) Mennyi a kód minimális távolsága?

b) Perfekt-e a kód?

c) Mi lehetett az átküldött kódszó, ha a vett szó $\begin{pmatrix} 1 & ? & 1 & 3 & ? \end{pmatrix}$?

A kód tisztán $0,1$ elemeket tartalmazó kódszavakat is tartalmaz.

d) Adja meg a bináris kódszavakat!

e) Igazolja, hogy ezen bináris kódszavak részkódot alkotnak az eredeti kódban!

f) Adja meg ezen részkód (n,k,d) paraméterhármasát!

g) Adja meg a részkód generátormátrixát!

4.8. feladat. Egy GF(5) feletti $(5,3)$ paraméterű lineáris kód kódszavai között vannak a $(0,1,0,1,2)$, $(1,0,0,1,4)$, $(0,0,1,1,3)$ szavak is. Adja meg a kód hibajavító képességét!

4.9. feladat. Adja meg egy $(21,18)$ paraméterű GF(4) feletti, egy hibát javító kód paritásmátrixát és generátormátrixát.

4.10. feladat. Adja meg a legkisebb kódszóhosszú GF(3) feletti, 1 hibát javító, $k = 2$ üzenethosszú szisztematikus kódot paritásellenőrző mátrixával!

4.11. feladat. Létezik-e $C(n,k)$, $n-k=2$, GF(3) feletti 1 hibát javító kód? Ha igen, adja meg szisztematikus mátrixaival!

4.12. feladat. Adja meg a $C(n,k=1)$ paraméterű bináris kód C' duális kódját (duális kód = paritásellenőrző mátrix mint generátormátrix által generált kód), s annak paraméterhármasát! Adja meg C' szavait $n=4$ esetén!

4.13. feladat. Létezhet-e olyan GF(q) feletti lineáris blokk-kód, amelynek generátormátrixa egyben a kód paritásellenőrző mátrixa is? Ha válasza igen, mutasson példát rá, mind $q=2$, mind pedig $q>2$ esetben.

4.14. feladat. Valaki azt állítja, hogy ha egy $C(n=2m-1,k)$ lineáris bináris kódnak a csupa 1 kódszó eleme, akkor pontosan eggyel kevesebb páros paritású nemzérus kódszava van, mint páratlan paritású. Igaza van-e?

4.15. feladat. Legyen $C(n,k)$ egy lineáris bináris blokk-kód, amelynek generátormátrixában nincsen csupa zérus oszlop. Igaz-e, hogy az összes kódszó egyeseinek összesített darabszáma $n \cdot 2^{k-1}$?

4.16. feladat. Egy GF(q) feletti lineáris blokk-kód paritásellenőrző mátrixa

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{q-2} \end{pmatrix},$$

ahol α a test primitív eleme. Adja meg a kódtávolságot!

4.17. feladat. A legegyszerűbb konstrukciójú hibajavításra már alkalmas nemtriviális kód a kétdimenziós bináris paritáskód. (Az üzenetet mátrixba rendezzük, soronként és oszloponként paritásbittel egészítjük ki, majd a jobb alsó sarokba írjuk a paritások paritását.) Mennyi a minimális távolság?

4.18. feladat. Igazolja, hogy a kétdimenziós bináris paritáskód jobb alsó paritáselemét (azaz a paritások paritását) képezhetjük akár a sorparitások paritásaként, akár az oszlopparitások paritásaként, azaz mindkét esetben azonos eredményre jutunk.

4.19. feladat. Egy bináris blokk-kódot a 2.5. szakaszban megismert Walsh–Hadamard-mátrix segítségével állítunk elő. A generált $C^{(2^r)}$ kód szavai az $\mathbf{A}_{2^r}$ mátrix soraiból, valamint azok komplemenseiből álljanak.

a) Lineáris-e a kód?

b) Adja meg a $C^{(2^r)}$ kód n,k,d paramétereit r függvényében!

4.20. feladat. Konstruáljon egy GF(q), $q = 2^m$ feletti $(q+1, q-1)$ paraméterű 1 hibát javító lineáris blokk-kódot.

a) Adja meg a szisztematikus paritásellenőrző mátrixot!

b) Perfekt-e a kód?

c) Adja meg $q = 4$ esetre a generátormátrixot! ($0 \leftrightarrow 0$, $1 \leftrightarrow 1$, $2 \leftrightarrow x$, $3 \leftrightarrow x+1$)

4.21. feladat. Igaz-e, hogy tetszőleges $C(n,k,d=3)$ paraméterű lineáris kódot egy paritásszimbólummal kiegészítve $C'(n+1,k,d=4)$ paraméterű kódot kapunk?

Kalkulus

4.22. feladat. Egy egyenlő oldalú háromszöggel háromféle elemi transzformációt végezhetünk:

- e: helybenhagyás
- t: tengelyes tükrözés
- f: középpont körüli 120°-os forgatás

A szorzás művelet két elemi transzformáció között legyen azok egymás utáni alkalmazása. Adja meg ezen transzformáció csoport műveleti tábláját, részcsoportjait!

4.23. feladat. Mutassa meg, hogy az egészek ($+, -$, zéró) halmaza nem csoport a kivonás műveletre!

4.24. feladat. Legyen S egy véges halmaz, valamint G az S részhalmazainak halmaza.

a) Mutassa meg, hogy G a halmazunió művelettel nem alkot csoportot!

b) Mutassa meg, hogy G az $A \Delta B = (A-B) \cup (B-A)$ szimmetrikus differencia művelettel csoportot alkot!

c) Mutassa meg, hogy a „Δ” és „$\cap$” (halmaz metszet) műveletekkel G gyűrűt alkot! (A $+$ és $*$ bináris műveletekkel bíró R halmazt gyűrűnek nevezzük, ha R az összeadásra nézve Abel-csoport, a szorzásra nézve félcsoport, és a műveletek között érvényesek a disztributivitási szabályok.)

4.25. feladat. Adja meg a mod 7 szorzócsoport egy mellékosztály dekompozícióját!

4.26. feladat. Adjon példát egységelem nélküli gyűrűre!

4.27. feladat. Mutassa meg, hogy a $p(x) = x^2 - 1$ polinomnak több, mint két gyöke van a mod 15 gyűrűben! Nem ellentmondás ez?

4.28. feladat. Tekintse az $S = \{0,1,2,3\}$ halmazt az alábbi műveleti táblák szerinti „+” és „$*$” műveletekkel:

$+$	**0**	**1**	**2**	**3**
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$*$	**0**	**1**	**2**	**3**
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Testet kapunk-e?

4.29. feladat. Konstruálja meg GF(4) műveleti tábláit!

4.30. feladat. Konstruálja meg GF(8) műveleti tábláit:

a) Az $x^3 + x + 1$ bináris irreducíbilis polinom felhasználásával!

b) Ismételje meg a konstrukciót az $x^3 + x^2 + 1$ bináris irreducíbilis polinom felhasználásával, s mutassa meg hogy a két test izomorf (az elemek átnevezésével azonos műveleti táblákhoz jutunk)!

4.31. feladat. Legyen adva GF(4) a következő műveleti táblákkal:

$+$	**0**	**1**	**2**	**3**
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

$*$	**0**	**1**	**2**	**3**
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

a) Oldja meg az alábbi GF(4) feletti egyenletrendszert:

$$\begin{aligned} 2x+y &= 3 \\ x+2y &= 3 \end{aligned}$$

b) Számítsa ki az alábbi GF(4) feletti mátrix determinánsát!

$$\det \begin{pmatrix} 2 & 1 & 2 \\ 1 & 1 & 2 \\ 1 & 0 & 1 \end{pmatrix} = ?$$

4.32. feladat. Adja meg az $x+1 \in$ GF(8) polinom alakban megadott testelem inverzét, ha x^3+x^2+1 az aritmetika generáló polinom!

4.33. feladat. Adja meg a GF(8) test elemeinek bináris minimálpolinomját!

4.34. feladat.

a) Mutassa meg, hogy a $p(x) = x^3+x^2+2$ GF(3) feletti polinom irreducíbilis!

b) Adja meg GF(27) elemeinek rendjét!

c) Mi az x polinom által reprezentált elem rendje GF(27)-ben, ha $p(x)$ az aritmetika generáló polinom?

4.35. feladat. Konstruálja meg GF(9) műveleti tábláit!

4.36. feladat. A GF(16) test összeadó- és szorzótábláját többféleképpen is megkonstruálhatjuk:

a) GF(2) feletti 4-edfokú irreducíbilis polinommal,

b) GF(4) feletti 2-edfokú irreducíbilis polinommal.

Kövessük a b) utat!

4.37. feladat. A $p(x) = x^{20}+x^3+1$ GF(2) feletti polinommal GF(2^{20}) konstruálható, ahol az x polinom által reprezentált elem primitív:

a) Adja meg ezen test résztesteinek méretét!

b) Mely résztesteknek nincsen további részteste a GF(2) testen kívül?

4.38. feladat. Igaz-e, hogy $(x-3)(x+2)(x+8) \mid (x^{16}+10x^6)$ GF(11) felett?

4.39. feladat. Legyen α primitív elem a GF(16) testben, amelynek bináris minimálpolinomja x^4+x+1. Adja meg az α minimál polinomját GF(4) felett!

4.40. feladat. Adja meg az x^8-1 polinom GF(3) feletti irreducíbilis polinomokra történő faktorizációját! Mennyi különböző, $n=8$ hosszúságú, GF(3) feletti ciklikus kód van?

4.41. feladat. $f(x)=3x^4+2x+4$ egy GF(5) feletti polinom. Adjuk meg az $[f(x)]^{25}$ polinomot explicit alakban!

4.42. feladat.

a) Hány különböző másodfokú x^2+ax+b alakú (fő)polinom van GF(16) felett?

b) Hány különböző $(x-\beta)(x-\gamma)$, $\beta,\gamma\neq 0$ alakú polinom van GF(16) felett?

c) Adja meg a GF(16) feletti irreducíbilis másodfokú főpolinomok számát!

4.43. feladat. Igaz-e, hogy egy $\alpha\in$ GF(q) primitív elem konjugáltja is primitív eleme a testnek, ahol $q=2^m$?

4.44. feladat. Legyen β egy GF(q) test nemzérus eleme. Mi az a legkisebb n érték, amelyre $\beta+\beta^2+\beta^3+\cdots+\beta^n=0$ teljesül?

4.45. feladat. Legyenek S és T különböző kétdimenziós alterek egy háromdimenziós vektortérben. Mutassa meg, hogy S és T metszete egydimenziós altér!

4.46. feladat. Származzanak az x,y és z vektorok egy GF(p), p prím, feletti lineáris blokk-kód egy generátomátrixa sorvektorainak halmazából. Lehetnek az $x+y,y+z,x+z$ vektorok ugyanezen kód valamely más generátormátrixának sorai közül valók,

a) ha $p=2$,

b) ha $p>2$?

4.47. feladat. Legyen $\alpha\in$ GF(q), $q=2^m$, egy n-edrendű elem, továbbá

$$\begin{aligned}\mathbf{a} &= \left(1,\alpha^3,\alpha^6,\ldots,\alpha^{3i},\ldots,\alpha^{3(n-1)}\right)\\ \mathbf{b} &= \left(1,\alpha^5,\alpha^{10},\ldots,\alpha^{5i},\ldots,\alpha^{5(n-1)}\right).\end{aligned}$$

Ortogonálisak-e ezen vektorok?

4.48. feladat. Tekintse a $p(x) = x^3 + 1$ és $q(x) = x^4 + x^3 + x^2 + 1$ bináris polinomokat.

a) Számolja ki a legnagyobb közös osztót: $\text{lnko}(p(x), q(x))$-et!

b) Adjon meg $A(x), B(x)$ polinompárt, amelyre $\text{lnko}(p(x), q(x)) = A(x)p(x) + B(x)q(x)$ fennáll.

4.49. feladat.

a) Számolja ki a legnagyobb közös osztót: $\text{lnko}(1753, 308)$-at!

b) Adjon meg A, B egészeket, amelyre $\text{lnko}(1753, 308) = 1753A + 308B$.

4.50. feladat. Kalkulátor használata nélkül adja meg $8^{64} + 7^{311} \mod 3$ értékét!

Ciklikus kódok

4.51. feladat. Valaki azt állítja, hogy egy 1 hibát javító bináris ciklikus kód egyik szava 0001111. Lehetséges ez?

4.52. feladat. Tekintsük a $g(x) = x^3 + x^2 + 1$ generátorpolinomú, $n = 7$ kódszóhosszú bináris Hamming-kódot.

a) Adja meg a kód $h(x)$ paritásellenőrző polinomját és szisztematikus alakú generátormátrixát!

b) Tekintsük a kód nem páros súlyú szavainak halmazát. Adja meg ezen részkód méretét, valamint minimális távolságát!

c) Tekintsük a nem páratlan súlyú szavainak halmazát. Lineáris, illetve ciklikus-e ez a halmaz, s mik a paraméterei?

4.53. feladat. A $g(x) = x^3 + x + 1$ bináris polinom egy 7 kódszóhosszú Hamming-kód generátorpolinomja. Adja meg

a) a kódszavak halmazát,

b) a minimális távolságot,

c) a paritásellenőrző polinomot!

4.54. feladat. Egy $n = 7$ kódszóhosszú bináris ciklikus blokk-kód generátorpolinomja $g(x) = x - 1$. Adja meg a

a) lehetséges kódszósúlyokat, és a k, d paramétereket,

b) paritásellenőrző polinomot,

c) szisztematikus paritásmátrixot!

4.55. feladat. Egy $n = 7$ hosszú bináris ciklikus kód generátorpolinomja $g(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$. Adja meg a kódszavak halmazát!

4.56. feladat.

a) Hány különböző 7 kódszóhosszú bináris ciklikus kód van?

b) Adja meg (n, k, d) paramétereivel és $g(x)$ generátorpolinomjával az összes lehetséges bináris $n = 7$ kódszóhosszú ciklikus kódot!

4.57. feladat. Egy C' kódot úgy származtatunk, hogy egy $g(x)$ generátorpolinomú $C(n,k)$, GF(q) feletti Reed–Solomon-kód kódszavait tükrözzük, azaz elemeit fordított sorrendben tekintjük ($c'_i = c_{n-1-i},\ i = 0, 1, \ldots, n-1$).

a) Ciklikus-e C'?

b) Ha az a) kérdésre a válasz igen, akkor adja meg a C' kód $g'(x)$ generátorpolinomját $g(x)$ alapján, továbbá annak gyökeit, ha $g(x)$ gyökei $\alpha_1, \alpha_2, \ldots, \alpha_{n-k}$.

4.58. feladat. Egy $C(n,k),\ n = 2^m - 1$ bináris ciklikus kód $g(x)$ generátorpolinomját osztja az $x + 1$ polinom. Eleme-e a kódnak a csupa 1 szó?

4.59. feladat. A $g(x) = x^3 + x^2 + 1$ polinom egy $n = 7$ kódszóhosszú bináris Hamming-kód generátorpolinomja. Adja meg a kód $h(x)$ paritásellenőrző polinomjára épülő, szisztematikus kódolást nyújtó visszacsatolt shiftregiszteres kódoló eszköz blokksémáját!

4.60. feladat. Egy $C_1(n, k_1, d_1)$ illetve egy $C_2(n, k_2, d_2)$ ciklikus kód $h_1(x)$ illetve $h_2(x)$ paritásellenőrző polinomja közötti kapcsolat $h_1(x) \mid h_2(x)$. Mi a kapcsolat:

a) C_1 és C_2 között,

b) d_1 és d_2 között?

4.61. feladat. Egy $C_1(n_1,k_1)$ ciklikus kód egy $C_2(n_2,k_2)$ ciklikus kód részkódja. Mi az algebrai kapcsolat a megfelelő

a) $h_1(x), h_2(x)$ paritásellenőrző polinomok között,

b) $g_1(x), g_2(x)$ generátorpolinomok között?

4.62. feladat. Mennyi különböző, $n = 8$ kódszóhosszú GF(3) feletti ciklikus kód van?

4.63. feladat. Egy $n = 15$ kódszóhosszú C bináris Hamming-kód $\mathbf{H}$ paritásellenőrző mátrixának oszlopai az $1,2,\ldots,15$ egészek bináris alakjai. Adjon meg egy $1 \to i_1,\ 2 \to i_2,\ \ldots,\ 15 \to i_{15}$ permutációt, hogy a keletkező C^* kód már ciklikus legyen!

4.64. feladat. Igazolja, hogy egy $h(x)$ paritásellenőrző polinomú ciklikus kód paritásellenőrző mátrixának sorait a $(0,0,\ldots,h_k,h_{k-1},\ldots,h_0)$ vektor ciklikus eltolásaival nyerhetjük.

4.65. feladat. Képezzük a CRC-t a $g(x) = x^5 + x^3 + x^2 + 1$ generátorpolinommal. Jelez-e hibát a detektor, ha a vett szó 0000 0001 0011 1011, ahol a jobb oldali bit a zéró helyiértékű?

4.66. feladat. A következőket állítja valaki:

a) Egy $C(n,k)$ ciklikus lineáris kód $h(x)$ paritásellenőrző polinomját használhatom egy n kódszóhosszú C' kód generátorpolinomjaként.

b) A C' kód minimális kódtávolsága elérheti a $k+2$ értéket is.

Igazak-e az állítások?

Kódkorlátok

4.67. feladat. Konstruálható-e $n = 11, k = 5$ paraméterű $t = 2$ hibát javító bináris kód?

4.68. feladat. Valaki azt állítja, hogy olyan kódot tervezett, amely 7 redundanciakarakterrel meghosszabbítja az üzenetblokkot, és 4 véletlen hibát képes javítani a kódszóban. Lehetséges ez?

4.69. feladat. Létezik-e $C(n,k)$, $n-k=2$, GF(3) feletti egy hibát javító kód? Ha igen adjon példát, megadva a szisztematikus paritásellenőrző mátrixát és a kódszavait!

4.70. feladat. Perfekt-e a $\{(1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1), (0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0)\}$ kódszavakat tartalmazó kód?

4.71. feladat. Perfekt-e egy $C(11,6)$ paraméterű GF(3) feletti 2 hibát javító kód?

4.72. feladat. Igazoljuk, hogy egy MDS-kód duálisa is MDS tulajdonságú!

RS-kódok

4.73. feladat. Tekintsünk egy GF(11) feletti Reed–Solomon-kódot $g(x) = (x-2)(x-4)(x-8)(x-5)$ generátorpolinommal. Adja meg a kód következő jellemzőit: minimális távolság (d), hibajavító képesség (t_c), hibadetektáló képesség (t_d), törlésjavító képesség (t_e)!

4.74. feladat. Adja meg egy GF(11) feletti három hibát javító, primitív szóhosszú Reed–Solomon-kód paramétereit, generátorpolinomját, paritásellenőrző polinomját!

4.75. feladat. Tekintsünk egy GF(13) feletti 2 hibát javító primitív szóhosszú Reed–Solomon-kódot.

a) Adja meg a generátorpolinomjával!

b) Állítsa elő az $u(x) = 2x+1$ üzenethez tartozó kódszót transzformációs kódolással!

4.76. feladat. Eleme-e az $(1,1,\ldots,1)$ csupa 1 vektor egy GF(q) feletti $C(n,k)$, $n=q-1$ Reed–Solomon-kódnak, ahol a generátorpolinom gyökei az α primitív elem $0,1,2,\ldots,n-k-1$ hatványai?

4.77. feladat. Egy t hibát javító GF(q) feletti Reed–Solomon-kód generátorpolinomjának gyöke az α primitív elem $2,\ldots,2t$-edik hatványa. Lehetséges-e, hogy a kódszavak elemeinek (koordinátáinak) összege a test zéró eleme legyen?

4.78. feladat. Legyen $\alpha^{(r)} = \left(1, \alpha^r, \alpha^{2r}, \ldots, \alpha^{(n-1)r}\right)$, $r=0,1,\ldots,n-1$, ahol $\alpha \in$ GF(q) egy n-edrendű elem. Igaz-e, hogy ha egy $C(n,k)$ kód $\mathbf{G}$ generátormátrixának sorai rendre $\alpha^{(r)}$, $r=0,1,\ldots,k-1$, akkor $\mathbf{H}$ mátrixának sorai lehetnek rendre az $\alpha^{(r)}$, $r=1,\ldots,n-k$ vektorok?

4.79. feladat. Legyen $\alpha^{(r)} = \left(1, \alpha^r, \alpha^{2r}, \ldots, \alpha^{(n-1)r}\right)$, $r = 0, 1, \ldots, n-1$, ahol GF(q) egy n-edrendű elem. Van-e olyan $C(n,k)$ kód, amelyre a **G** és **H** mátrixainak sorai rendre ugyanazok az $\alpha^{(r)}$ alakú vektorok?

4.80. feladat. Legyen $g(x) = x^3 + x + 1$ egy $C(7,4)$ bináris Hamming-kód generátorpolinomja. Mutassuk meg, hogy C lineáris részkódja egy $C'(7,5)$ GF(8) feletti Reed–Solomon-kódnak!

4.81. feladat. Igaz-e a következő állítás? Egy $C(n,k,d)$ GF(q) feletti Reed–Solomon-kód kódszavaiból kiemelve bármely, rögzített k méretű koordinátahalmaz által meghatározott részvektorokat, azok különbözők a különböző kódszavakra.

BCH-kódok

4.82. feladat. Adjuk meg egy egy hibát javító $n = 8$ primitív szóhosszú, GF(3) feletti kód

a) üzenethossz (k) paraméterét,

b) generátorpolinomját!

(Segítség: $f(x) = x^2 + x + 2$ egy GF(3) feletti irreducíbilis polinom).

4.83. feladat. Adjon meg egy bináris $n = 15$ kódszóhosszú, 2 hiba javítására alkalmas kódot bináris generátorpolinomjának explicit megadásával!

4.84. feladat. $n = 31$ kódszóhosszú $t = 2$ hiba javítására alkalmas bináris BCH blokk-kódot tervezünk. Adja meg a keletkező kód

a) k paraméterét,

b) generátorpolinomját!

(Segítség: $f(x) = x^5 + x^2 + 1$ egy primitív irreducíbilis bináris polinom).

4.85. feladat. Adja meg egy GF(3) feletti $n = 91$ primitív kódszóhosszú $t = 7$ hibajavító képességű BCH-kód üzenethossz paraméterét!

4.86. feladat. Adja meg a GF(8) feletti $(9,7)$ paraméterű 1 hibát javító BCH-kód generátorpolinomját gyöktényezős alakban!

Konvolúciós kód

4.87. feladat. Mekkora a d_∞ minimális távolsága (szabad távolsága) a

$$G(x) = \left(x^2+x+1, \quad x^5+x^2+1\right)$$

generátorpolinom-mátrixú konvolúciós kódnak?

4.88. feladat. Adjon meg egy $t = 2$ véletlen hiba javítására alkalmas bináris $R = \frac{1}{2}$ sebességű konvolúciós kódot generátorpolinom-mátrixával!

4.89. feladat. Egy $\frac{1}{2}$ sebességű bináris konvolúciós kód generátorpolinomjai: 17, 15 (együtthatók oktális ábrázolásban).

a) Adja meg a kódoló vázlatát!

b) Katasztrofális tulajdonságú-e a kód?

c) Mekkora a szabad távolság (d_∞)?

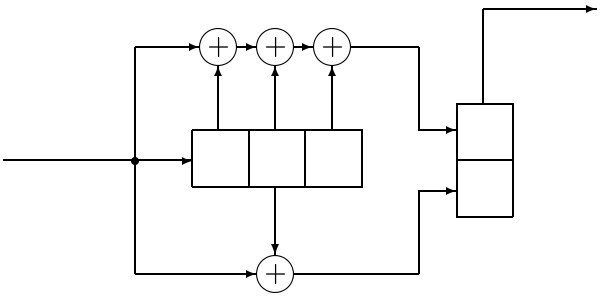
4.90. feladat. Adja meg a következő, generátorpolinom-mátrixával megadott $R = \frac{2}{3}$ sebességű bináris konvolúciós kód kódoló blokksémáját, valamint d_∞ szabad távolságát!

$$G(x) = \begin{pmatrix} x & x^2+1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

4.91. feladat. Mekkora a sebessége, s a minimális távolsága a $G(x) = \left(1, \ x^2+1\right)$ generátorpolinom-mátrixú konvolúciós kódnak?

4.92. feladat. Tekintsük a $G(x) = \left(x^2+1, \quad x^3+x+1\right)$ generátorpolinom-mátrixú konvolúciós kódot. Adja meg az állapotátmenet-gráfját, s a szabad távolságát (d_∞)!

4.93. feladat. Katasztrofális tulajdonságú-e az alábbi blokksámával megadott kódoló?



Kódkombinációk, kódmódosítások

4.94. feladat. Adja meg a $g(x) = x^3 + x + 1$ generátorpolinomú $C(7,4)$ kód nem páratlan súlyú szavai C' részkódjának halmazát, s ezen részkód paramétereit!

4.95. feladat. Perfekt marad-e a $C(n,k)$ bináris Hamming-kód, ha kódrövidítést hajtunk végre, amelynek mértéke

a) 1 bit,

b) 2 bit?

4.96. feladat. A $C(7,4)$ bináris Hamming-kód kódszavait paritáskarakterrel bővítjük páros paritásúra egészítve ki a kódszavakat. Adja meg a kapott kód paramétereit!

4.97. feladat. A $(7,4)$ bináris Hamming-kódból kiindulva konstruáljon bináris kódot, amelynek 8 kódszava van, 7 a szóhossza és alkalmas 3 hiba detektálására.

4.98. feladat. A $g(x) = x^3 + x^2 + 1$ generátorpolinomú szisztematikus Hamming-kódon 3 bites kódrövidítést hajtunk végre.

a) Adja meg a rövidített kód (n,k) paramétereit!

b) Adja meg a kódszavakat és a kód d paraméterét!

4.99. feladat. Adja meg a 8 bitnyi kódrövidítéssel kapható kód paramétereit és kódszavait, ha a $g(x) = x^4 + x + 1$ generátorpolinomú bináris Hamming-kódot rövidítettük.

4.100. feladat. $C(n,k)$, $n = 91$ kódszóhosszú 7 hibát javító bináris blokk-kódot szeretnénk konstruálni. Egy megfelelő primitív szóhosszú $C^*(N,K)$ bináris BCH-kód rövidítésével oldjuk meg a feladatot. Adja meg az N,K,k paramétereket!

4.101. feladat. Legyen $C(255,251)$ egy GF(256) feletti Reed–Solomon-kód, amelynek generátorpolinomja $g(x) = (x-1)(x-\alpha)(x-\alpha^2)(x-\alpha^3)$, $\alpha \in$ GF(256) primitív elem.

a) Igazolja, hogy a bináris kódszavak C bináris ciklikus C' alterét alkotják!

b) Mi ezen C' altér generátorpolinomja és mik C' paraméterei?

4.102. feladat. Egy bináris lineáris $C(n,k,d)$ kód nem tartalmazza a csupa egyesekből $(\mathbf{1})$ álló kódszót. Mit mondhatunk a $C' = C \oplus \mathbf{1}$ kódról, ahol $\oplus$ a koordinátánkénti mod 2 összeadás?

a) Lineáris-e?

b) Mik a paraméterei: n', k', d' ?

Mit mondhatunk a $C'' = C \cup C'$ kódról, ahol $\cup$ a halmazegyesítés:

c) Lineáris-e?

d) Mik a paraméterei: n'', k'', d'' ?

4.103. feladat. Egy GF(q) feletti $n = q-1$ szóhosszú C Reed–Solomon-kód generátorpolinomjának gyökei $\alpha, \alpha^2, \ldots, \alpha^{d-1}$, $\alpha \in \mathrm{GF}(q)$. A kódot egy „paritás” karakterrel bővítjük, olyan módon, hogy a kódszó karaktereinek testbeli aritmetika szerinti összege lesz az $n+1$-edik karakter. MDS tulajdonságú marad-e a kapott q szóhosszú kód?

4.104. feladat. Egy kommunikációs csatornán nagyon ritkán maximum 8 bit hosszú hibacsomók keletkeznek. A következő beállítható paraméterű kódolási elemekben gondolkozunk:

a) bináris Hamming-kódoló,

b) bájt karakter alapú Reed–Solomon-kódoló,

valamint alkalmazhatjuk a kódátfűzés technikát is. A cél minimális redundancia mellett elvégezni a javítást. Milyen konstrukciót alkalmazzunk?

Hibajavító dekódolás

4.105. feladat. Adja meg egy GF(q) feletti, egy hibát javító BCH-kód PGZ-dekódolás szerinti általános dekódolási algoritmusát a szindrómaegyenletek közvetlen megoldásával!

4.106. feladat. Egy GF(11) feletti lineáris blokk-kód paritásellenőrző mátrixa

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^9 \end{pmatrix},$$

ahol $\alpha = 2$ a test primitív eleme.

a) Adja meg a kód paramétereit!

b) Adja meg a dekódolás menetét az $e(x) = 5x^3$ hibapolinom esetére!

4.107. feladat. Adja meg egy GF(q) feletti két hibát javító bináris BCH-kód általános dekódolási algoritmusát a szindrómaegyenletek közvetlen megoldásával!

4.108. feladat. Valaki azt állítja, hogy nem feltétlenül kell egy GF(2^m) feletti $C(n = 2^m - 1, k)$ bináris kód generátorpolinomjában 4 ciklikusan egymás utáni gyöknek lennie ahhoz, hogy a kód $t = 2$ hibát javíthasson. Szerinte az is megfelelő, ha a $g(x)$ generátorpolinom olyan, hogy $g(\alpha) = g(\alpha^{-1}) = 0$, ahol α a GF(2^m) primitív eleme. Igaza van-e?
(Segítség: A szindrómaegyenletek megoldhatóságnak közvetlen vizsgálatával ellenőrizze az állítást.)

4.109. feladat. Valaki a következő gondolatmenetet mondja a társának:
Én úgy tudom, hogy egy $C(n,k)$ lineáris bináris blokk-kód szindróma dekódolási táblázata mindig $2^{(n-k)}$ javítható hibamintát tartalmaz, tehát végülis a minimális távolságnak nincs jelentősége, s mindegyik (n,k) paraméterű kód egyformán hasznos (hasznos = emlékezetnélküli BSC csatornán használva azonos kódszó-dekódolási hibavalószínűséget kapunk). Tömör érveléssel tegyen igazságot! Mutasson egy egyszerű példát is!

4.110. feladat. A és B beszélgetnek:

A: A BCH-kódok PGZ-szindróma dekódolási algoritmusa egy maximum t súlyú hibavektort tud korrigálni. Tehát, ha több hiba esett, mint t, és a vett szó minden kódszótól távolabb van, mint t, akkor a dekódolt szó nem lehet kódszó, azaz újbóli szindrómaszámítással lehetőségünk van ellenőrizni a dekódolás helyességét.

B: Itt valami csalás van, hiszen szindróma dekódoláskor a dekódolt szó szindrómáját mindig zérusra korrigáljuk, mivel a vett szóból levonjuk a vett szó szindrómájának megfelelő szindrómájú hibavektort.

Hol az igazság?

4.111. feladat. Egy $C(n,k)$ blokk-kód dekódolását szindróma dekódolási táblázat alapján végezzük. Az alábbi két állítás közül melyik az igaz és miért?

a) A dekóder kimenetén az aktuális hibázástól függetlenül mindig valamilyen (esetleg hibás) kódszó jelenik meg.

b) A dekóder a táblázat alapján egyszerűen levon a vett szóból egy hibamintát, így súlyosabb hibázás esetén előfordulhat, hogy nem egy kódszó jelenik meg a kimeneten.

4.112. feladat. Egy egy hibát javító GF(7) feletti $g(x) = (x-\alpha)(x-\alpha^2)$, $\alpha = 3$, generátorpolinomú Reed–Solomon-kód egy kódszava hibásan, de még javíthatóan érkezett meg a vételi oldali dekódolóhoz. A vett szó $(v_0, v_1, v_2, v_3, v_4, v_5) = (6, 0, 5, 2, 4, 0)$. Mi lehetett az átküldött kódszó?

4.113. feladat. Egy GF(11) feletti $C(10, 6)$ 2 hibát javító Reed–Solomon-kód generátorpolinomja $g(x) = (x-2)(x-4)(x-8)(x-5)$, $\alpha = 2$. Egy vett szóra a dekóder a következő szindrómákat számította ki a gyököknek megfelelő sorrendben: $3, 6, 1, 2$. Adja meg a hibapolinomot!

4.114. feladat. Egy GF(7) feletti $C(6, 2)$ Reed–Solomon-kód generátorpolinomja $g(x) = (x-3)(x-2)(x-6)(x-4)$, $\alpha = 3$. Egy vett szóra a dekóder a következő szindrómákat számította ki a gyököknek megfelelő sorrendben: $5, 5, 4, 2$. Adja meg a

a) hibahelypolinomot,

b) hibapolinomot!

4.115. feladat. Egy $C(15, 5)$ paraméterű $t = 3$ hibát javító bináris BCH-kód gyökei a következő GF(16) testelemek: $\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6$. A GF(16) aritmetikát az $f(x) = x^4 + x + 1$ polinommal generáljuk. A dekóder a következő szindrómákat számította a fenti gyökök sorrendjében: $\alpha^{12}, \alpha^9, 0, \alpha^3, 1, 0$. Adja meg a

a) hibahelypolinomot,

b) hibapolinomot!

4.116. feladat. Egy GF(11) feletti $g(x) = (x-2)(x-4)$ generátorpolinomú primitív szóhosszú Reed–Solomon-kód dekóderéhez érkezett vett szóból 2 karakter törlődött:

0.	1.					7.		9.
(8	2	0	0	2	0	0	?	0	?)

Mik a törlődött karakterek?

4.117. feladat. Van GF(256) aritmetikában gyorsan számoló egységünk. A csatorna hibázása olyan, hogy ritkán, legfeljebb 16 bit hosszú hibacsomók keletkezhetnek. Milyen kódot javasol a javításra, ha maximalizálni szeretnénk a kódolási sebességet?

4.118. feladat. Egy üzenetforrás kimenetén 8 különböző karakter jelenhet meg. Ha ezeket a karaktereket az átviteli csatornán továbbítjuk, akkor alkalmanként egy karakter meghibásodik, de két hibás karakter közti távolság legalább 10 karakter. Blokk-kódos hibajavítást szeretnénk alkalmazni azzal a megkötéssel, hogy a relatív redundancia nem haladhatja meg a 30 %-ot. Javasoljon megoldást a hibajavításra, adja meg az alkalmazandó kódot!

Hibadetekció

4.119. feladat. Valaki azt állítja, hogy tetszőleges m fokszámú, bináris $f(x)$ polinom, amelynek konstans tagja 1, alkalmas arra, hogy CRC generátorpolinomként detektáljunk vele tetszőleges, legfeljebb m bit hosszúságú hibacsomagot. (Pl. 1xxx1 egy 5 hosszú hibacsomag.) Igaza van-e?

4.120. feladat. Egy hibadetekciós protokollban a $000\ldots0011$ (két utolsó bitjén 1-et tartalmazó) üzenetcsomaghoz a $g(x) = x^{16} + x^{12} + x^5 + 1$ szabványos generátorpolinommal ciklikus redundancia ellenőrző összeget (CRC) alkalmazunk. Adja meg az ellenőrzőösszeggel ellátott blokkot!

Hibavalószínűség

4.121. feladat. Egy bináris kód generátormátrixa

$$\mathbf{G} = \begin{pmatrix} 11001 \\ 01110 \end{pmatrix}.$$

A kódot emlékezetnélküli BSC(p) csatornán hibadetekcióra illetve hibajavításra használjuk. A hibajavítás algoritmusa a legközelebbi kódszóra döntés. Adja meg mindkét alkalmazás esetén a hibázás valószínűségét! (A forrás azonos valószínűséggel sorsolja az üzeneteket.)

4.122. feladat. Egy $C(n,k)$ paraméterű GF(q) feletti, t hibát javító perfekt kódot hibajavításra használunk szimmetrikus emlékezetnélküli csatornán, ahol a hibázás valószínűsége p. Adja meg egy kódszó téves dekódolásának valószínűségét!

4.123. feladat. Emlékezetnélküli szimmetrikus q-áris ($0,1,\ldots,q$ input ábécé, $0,1,\ldots,q$ output ábécé) csatornán kommunikálunk, ahol a hibázás valószínűsége p ($\mathbf{P}(i \mid j) = p$, $i \neq j$). Egy (n,k) paraméterű GF(q) feletti t hibát javító perfekt kódot használunk csatornakódként. Adja meg egy kódszó téves dekódolásának valószínűségét!

4.124. feladat. Tekintsük a $g(x) = x^4 + x + 1$ generátorpolinomú bináris Hamming-kódot. A kódot hibajavításra használjuk. Adja meg egy kódszó téves dekódolásának valószínűségét!

4.125. feladat. A $(8,4)$ paraméterű paritásbittel bővített bináris Hamming-kódot hibadetekcióra használjuk p hibázási valószínűségű emlékezetnélküli bináris szimmetrikus csatornán. Adja meg a detekció mulasztás valószínűségét!

4.126. feladat. A $C(n, k = 1)$ paraméterű bináris ismétléses kódot

a) tisztán törléses csatornán

b) véletlen bithibázásos csatornán

használjuk, ahol a törlés illetve hibázás valószínűsége p. Adja meg mindkét esetben egy kódszó téves dekódolásának valószínűségét!

4.127. feladat. 1 bitnyi üzenetet úgy viszünk át a csatornán, hogy ismételjük azt, azaz a $(00\ldots0)$ illetve az $(11\ldots1)$ szavak valamelyikét küldjük át. Legyen $p = 0.01$ a bithibázás valószínűsége az emlékeznélküli bináris csatornában. Menynyivel javul a téves dekódolás valószínűsége, ha $n = 3$ hosszú szavak helyett $n = 5$ hosszúakat használunk?

4.128. feladat. Az alábbi méretű kétdimenziós paritáskódot paritásellenőrzésre használjuk (u: üzenetbit, p: paritásbit)

$$\begin{pmatrix} u & u & p \\ u & u & p \\ p & p & p \end{pmatrix}.$$

Adja meg a hibadetekció elmulasztásának valószínűségét emlékezetnélküli BSC(p) csatorna esetén!

4.129. feladat. Egy bináris, tisztán törléses emlékezetnélküli csatornán $p = 0.05$ a törlés és $1 - p = 0.95$ a hibátlan továbbítás valószínűsége. Félbájtos (4 bit) egységekben továbbítjuk a forrás információját, amelyet 4 bit redundanciával kiegészítünk kódszóvá. Hasonlítsuk össze az alábbi két kódolási eljárást a kódolási hatékonyság szempontjából:

a) a redundancia nem más mint az üzenet félbájt megismétlése,

b) a $(8,4)$ paraméterű, paritásbittel kiegészített Hamming-kódot használjuk.

4.130. feladat. Tisztán törléses hibát okozó emlékezetnélküli bináris csatornán $p = 0.05$ a törlés valószínűsége. 4 bites üzeneteinket paritásbittel bővített $(7,4)$ Hamming-kóddal továbbítjuk. Elfogadhatóan választottuk-e a kódot, ha üzeneteinket legalább 0.999 valószínűséggel szeretnénk a vevőben helyesen rekonstruálni?

4.131. feladat. 18 bájt méretű üzenetcsomagjainkat 2 bájt méretű CRC-vel védjük egy $p = 0.001$ bithibázás valószínűségű emlékezetnélküli BSC csatornán.

a) Tegyük fel, hogy zajmentes nyugtázócsatorna áll rendelkezésre, s a hibadetekció tökéletes! Mennyi a csomagismétlések átlagos száma?

b) Mekkora ugyanez a szám, ha a nyugtázó csatorna is ugyanilyen mértékben hibázhat, ahol az 1 bájt méretű nyugta szintén 2 bájt méretű CRC-vel védett. Az adó csak akkor nem ismétel, ha hibátlan nyugta érkezik. Mennyi a csomagismétlések átlagos száma?

4.132. feladat. p hibázási valószínűségű emlékezetnélküli csatornán N bájt méretű, T időtartamú csomagokat továbbítunk. A hibakontroll CRC alapú hibadetekció. Az adó addig nem küldi el a következő csomagot, amíg az utoljára elküldött csomag sikeresen át nem jutott a csatornán, s erről a visszairányú csatornán nyugtát nem kapott. Tegyük fel, hogy a nyugtázás szintén T időt vesz igénybe. Mekkora a csomagkésleltetés várható értéke, ha a visszairányú csatorna hibamentes?

4.133. feladat. Additív gaussi csatornán kommunikálunk, s modemünk E_b/N_0 jel/zaj viszony mellett $p = \mathrm{e}^{-E_b/N_0}$ valószínűséggel hibázik. $(7,4)$ paraméterű Hamming-kóddal szeretnénk hibát javítani. $P_w = 10^{-6}$ a megengedhető kódszóhiba valószínűség.

a) Mekkora jel/zaj viszony mellett érhető ez el, ha nem használunk kódolást?

b) Tegyük fel, hogy 1 hibát javít dekóderünk, s több mint 1 hiba esetén érintetlenül hagyja a vett szót. Mennyivel javul a jel/zaj viszony nyereség kódolás esetén?

5. fejezet

Kriptográfia

A kriptológia a titkos illetve védett kommunikáció tudománya, amelynek két ága a kriptográfia és a kriptoanalízis. A **kriptográfia** azon algoritmikus módszerekkel foglalkozik, amelyek biztosítják az üzenetek (tárolt információk) titkosságát, védettségét vagy hitelességét. A **kriptoanalízis** a titok — általában illetéktelen — megfejtésére („feltörésére") tartalmaz eljárásokat.

A kriptológia mint titkosírás és annak fejtése nagy hagyományokkal rendelkezik, amely történeti előzményekkel kapcsolatosan [15]-ban találhatunk példákat.

A kriptográfiával kapcsolatos nyilvános kutatások a '70-es évek közepétől egyre növekvő iramban folynak. Ennek fő oka az algoritmikus adatvédelem iránt a privát (nem katonai vagy diplomáciai) szektorban megnyilvánuló igény, amelynek hátterében az áll, hogy napjainkban — különösen a fejlett hírközléssel rendelkező országokban — „érzékeny információk", amelyek jogtalan megszerzése súlyos anyagi és erkölcsi károk okozására adhat lehetőséget, nagy mennyiségben kerülnek átvitelre nyilvános távközlési csatornákon, vagy tárolódnak fizikailag átlagosan védett memóriákban. Gondoljunk például pénzügyi információkra, egészségügyi, életrajzi vagy személyi adatokra.

Titkos információk védelmi rendszerének az algoritmikus módszerek általában csak egyik pillérét jelentik, amelyek megfelelő fizikai valamint ügyviteli (információkezelés rendszabályai) eljárások alkalmazásával együtt biztosítják a valóságos rendszerekben az információvédelmet.

Az alábbiakban a kriptográfia elemeit taglaljuk, annak alapjaiba kívánunk betekintést nyújtani.

5.1. Alapfogalmak

A digitális információforrás kimeneti adatfolyamából, az ún. nyílt adatból blokkokat készítünk, amelyre az $\mathbf{x} = (x_1, x_2, \ldots, x_M)$ jelölést alkalmazzuk, s amelyet **nyílt üzenet**nek (plaintext) hívunk. A titkosító kódoló egy egy-egy értelmű leképezéssel ebből az $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ **titkosított** (rejtett) **üzenet**et (ciphertext) állítja elő:

$$\mathbf{y} = E_{\mathbf{k}}(\mathbf{x}), \tag{5.1}$$

ahol $E_{\mathbf{k}}$, $\mathbf{k} = (k_1, k_2, \ldots, k_K)$ paraméterű invertálható kódoló transzformáció. A $\mathbf{k}$ vektor a titkosítás **kulcs**a, az az információ, amely egyértelműen meghatározza az aktuális titkosító transzformációt.

A titkosító dekódoló az inverz transzformáció, $D_{\mathbf{k}}$ felhasználásával

$$\mathbf{x} = D_{\mathbf{k}}(\mathbf{y}) \tag{5.2}$$

módon reprodukálja a nyílt üzenetet.

Tekintsünk először néhány egyszerű példát.

5.1. példa. Egy titkosító kódoló kimenetén az alábbi karaktersorozat jelenik meg az angol ábécé betűiből:

AELTJEKELHMLMTQECHECPBMTBEQSKCDJ.

Betűnkénti helyettesítést alkalmaztunk az alábbi helyettesítő tábla felhasználásával:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
C I P H E R A B D F G J K L M N O Q S T U V W X Y Z ’

ahol egy felső sorbeli karaktert az alatta állóra cseréltünk ki. A nyílt és a rejtett üzenetblokk egy karakter hosszúságú, azaz $M = N = 1$. A helyettesítő táblát úgy konstruáltuk, hogy a második sorát egy értelmes szóval kezdtük (CIPHER), majd sorrendben felsoroltuk az ábécé fennmaradó karaktereit. A kulcs ez esetben az értelmes kezdőszó. A dekódolást a kulcs (és az annak megfelelő helyettesítő tábla) ismeretében elvégezhetjük, s az alábbi nyílt adatfolyamot kapjuk:

GENTLEMEN DO NOT READ EACH OTHERS MAIL.

5.2. példa. A fenti példabeli nyílt szöveget bontsuk 8 karakteres blokkokra, majd kódoljunk blokkonként az alábbi permutáció felhasználásával:

$$\Pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 4 & 3 & 7 & 1 & 8 & 2 & 5 & 6 \end{pmatrix},$$

azaz az 1. pozícióban levő karaktert a 4. pozícióba, a 2.-ban levőt a 3.-ba, stb. helyezzük át. Ezzel az alábbi rejtett szöveget kapjuk:

TNMGEELENORNEDOTAEOATDCHSRIHLEMA

A dekódolás során az inverz permutálást végezzük el blokkonként. A kulcs ez esetben az aktuális permutáció. A helyettesítés mellett ezen ún. transzpozíció (vagy keverés) eljárás a másik alapmódszer blokk-kódolók készítésében, általában a kettőt kombinálva alkalmazzák.

5.3. példa (Caesar-titkosító). A már Julius Caesar által is használt titkosító elve egyszerű. Az angol ábécé betűit feleltessük meg számoknak az $A = 0, B = 1, \ldots, Z = 25$ helyettesítéssel. Az $\mathbf{x} = (x_1, x_2, \ldots, x_M)$ nyílt szöveget az $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ rejtett szövegbe karakterenkénti — a megfelelő egészeken végrehajtott —

$$y_i = x_i + k_i \pmod{26}, \tag{5.3}$$

$i = 1, 2, \ldots, M$ modulo összeadással kódoljuk, ahol a $\mathbf{k} = (k_1, k_2, \ldots, k_K)$ kulcs is a $\{0, 1, \ldots, 25\}$ halmazból veszi az elemeit.

Számpéldaként legyen $\mathbf{k} = (3, 3, \ldots, 3)$ konstans elemű vektor, továbbá $M = 7$, s ekkor az $\mathbf{x} = (\mathrm{CAESARX})$ nyílt üzenetet az $\mathbf{y} = (\mathrm{FDHVDUA})$ rejtett üzenetbe kódoljuk. A dekódolás az

$$x_i = y_i - k_i \pmod{26},$$

$i = 1, 2, \ldots, M$ inverz művelettel történik.

5.4. példa. Az 5.3. példa nyílt szövegének megfeleltetett decimális értékek binárisba alakításával az alábbi bináris nyílt szöveget kapjuk:

$$\mathbf{x} = (00010, 00000, 00100, 10010, 00000, 10001, 10111)$$

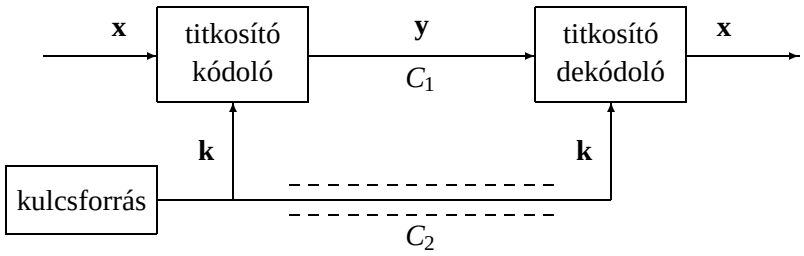
Legyen a kulcs egy a nyílt szöveggel azonos hosszúságú pénzfeldobás sorozat (fej=1, írás=0):

$$\mathbf{k} = (10011, 10011, 00100, 10111, 10010, 00001, 01110)$$

és képezzük a rejtett szöveget az $\mathbf{x}$ és $\mathbf{k}$ bitenkénti modulo 2 összeadásával:

$$\mathbf{y} = (10001, 10011, 00000, 00101, 10010, 10000, 11001)$$

A dekódolás a kódolás inverze, azaz $x_i = y_i - k_i = (y_i + k_i) \pmod{2}$.



5.1. ábra. A konvencionális titkosító rendszer elvi vázlata.

Az az eljárás, amikor **k** kulcsot minden egyes **x** kódolásakor függetlenül sorsoljuk, a **bináris véletlen átkulcsolás** (one-time-pad, ill. Vernam-titkosító néven is ismert), amely, mint látni fogjuk, elvileg tökéletes titkosító eljárás.

Az eddigiekben bevezetett titkosítási módszer az ún. **rejtett kulcsú**, más néven konvencionális vagy egykulcsos blokk-kódolás. A **konvencionális titkosító rendszer** blokkvázlatszerű modelljét láthatjuk az 5.1. ábrán.

A rejtett üzenetet a C_1 nyilvános csatornán (public channel), míg a kulcsot a C_2 titkos csatornán (secret channel) továbbítjuk a dekódoló oldalra, tehát a kódolás és a dekódolás azonos titkos kulcsot használ. A csatorna nyilvánosságát illetve titkosságát egy támadóval szembeni korlátozott illetve tökéletes védettsége jelenti. Például közhasználatú hírközlési csatornáink nyilvános, míg egy megbízható futár általi kézbesítés titkos csatornának felel meg.

A **támadó** (behatoló) célja lehet az **x** nyílt üzenet megállapítása vagy a **k** kulcs megszerzése. Alapfeltételezés a támadó ismereteivel kapcsolatosan, hogy az aktuális **k** kulcs kivételével a kódolás, dekódolás alkalmazott algoritmusát teljes részletességgel ismeri. (Titkosítási algoritmuson a titkosító kódolás és dekódolás számítási eljárását értjük.) Ez azt is jelenti, hogy egy titkosítási algoritmus által nyújtott védettség nem haladhatja meg a kulcsa védettségének mértékét. A kulcs az az információ, amely gyorsan és kívánatosan gyakran cserélhető, míg a titkosító rendszer többi elemét hosszabb ideig nem kell változtatni. A támadó által alkalmazható módszerek közül csak az algoritmikus támadási módszereket tekintjük. (Elképzelhetők további módszerek, pl. betörés, lefizetés alkalmazása, elektromágneses kisugárzás mérése is egy elszánt támadó részéről.)

Az algoritmikus típusú támadásnak passzív és aktív módját különböztetjük meg, amelyet a nyilvános csatornán hajt végre a támadó.

A **passzív módszer** az ún. lehallgatás, amikor a támadó a nyilvános csatornán áramló üzenetek sorozatának birtokába jut. A támadó célja az, hogy a megfigyelt rejtjelezett kapcsolatból kinyert információk felhasználásával algoritmikus támadást indítson az aktuális kulcs megállapítására. Ezt az eljárást szokás rejtjel-

fejtésnek nevezni, amelynek eszköztárát a kriptoanalízis módszerei alkotják. A rejtjelfejtő általában széleskörű háttérinformációra is alapozhat, mint pl. a nyílt szöveg nyelve, formátummegkötései, statisztikai tulajdonságai, tipikus szavai, kifejezései (gondoljunk pl. egy beszélt nyelven megfogalmazott táviratra, vagy egy számítógépes programra). Ezen háttérinformációk az éppen lehallgatott üzenetekkel együtt képezik azt az információbázist, amelyekre alapozva egy rejtjelfejtési művelet sikeres lehet. Így anélkül, hogy bárki az aktuális nyílt szövegnek akár egy részletét is elárulná a támadónak, az nagy valószínűséggel tudhat egy rejtett szöveghez tartozó nyílt szöveget, vagy annak egy részletét. Az is lehetséges egy számítógépes rendszerben, hogy a behatoló képes a rendszer egyik nem rejtjelezési védelmi vonala mögé kerülni (átlépve ügyvitel-kezelési rendszabályokat, fizikai védelmet), abból a célból, hogy kikényszerítsen általa választott nyílt szöveghez tartozó rejtett szöveget.

A passzív típusú támadásokat azok növekvő ereje szerinti sorrendben a következő kategóriákba sorolhatjuk:

a) **Rejtett szövegű támadás** (ciphertext only attack): támadás azonos kulccsal kódolt rejtett üzenetek birtokában, azaz a támadó rendelkezésére áll egy $E_{\mathbf{k}}(\mathbf{x}_1), E_{\mathbf{k}}(\mathbf{x}_2), \ldots, E_{\mathbf{k}}(\mathbf{x}_L)$ sorozat, ahol $\mathbf{k}$ az aktuális kulcs.

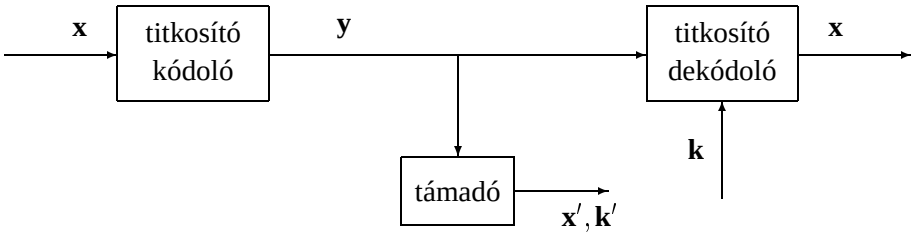
b) **Ismert nyílt szövegű támadás** (known plaintext attack): a támadó rendelkezésére álló $(\mathbf{x}_1, E_{\mathbf{k}}(\mathbf{x}_1)), (\mathbf{x}_2, E_{\mathbf{k}}(\mathbf{x}_2)), \ldots, (\mathbf{x}_L, E_{\mathbf{k}}(\mathbf{x}_L))$ nyílt és rejtett üzenetpárok birtokában történő támadás.

c) **Választható nyílt szövegű támadás** (chosen plaintext attack): támadás abban az esetben, amikor a támadó szabadon választhatja meg azt a nyílt üzenetet, amelynek rejtett párját látni szeretné. Képzeljük például azt, hogy egy banki információs rendszerben valaki egy tranzakciót kezdeményez, azaz generáltat egy $\mathbf{x}_1$-et, és látja annak titkosított változatát $E_{\mathbf{k}}(\mathbf{x}_1)$-et is.

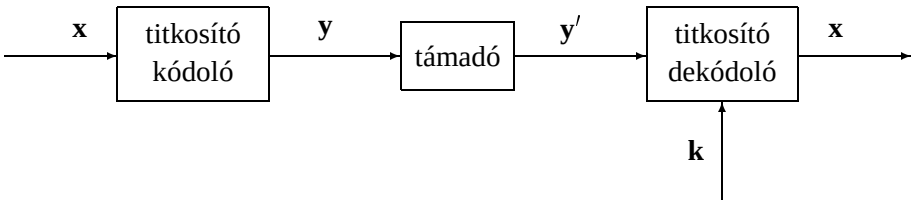
d) **Választható szövegű támadás** (chosen text attack): támadás abban az esetben, amikor a támadó szabadon választhatja meg akár a nyílt üzenetet, akár a rejtett szöveget, amelynek párját látni szeretné.

Aktív módszer a rejtett üzenetek csatornából történő kivonása, kicserélése, amelyek célja a hozzávetőlegesen ismert tartalmú rejtett üzeneteknek a támadó szempontjából kedvező módosítása (**üzenetmódosítás**). Egy másik aktív módszer az, amikor a támadó megpróbálja egy legális felhasználó szerepét eljátszani azért, hogy valamely másik legális rendszerelemtől információt csaljon ki (**megszemélyesítés**).

A kétféle támadási módszert illusztráljuk az 5.2. és 5.3. ábrán.



5.2. ábra. Passzív támadás.



5.3. ábra. Aktív támadás.

Egy üzenet **titkosság**a (privacy) azt jelenti, hogy csak a legális (kívánatos) partner számára rekonstruálható annak nyílt tartalma, míg egy üzenet **hitelesség**e (authenticity) azt jelenti, hogy azt olyan személy generálta, aki a kulcs legális birtokában van. Lényegében azt mondhatjuk, hogy a titkosság az üzenet tartalmával, míg a hitelesség az üzenetet küldő személlyel kapcsolatos. Szemléletes példával illusztrálva: ha a postás felbontja a levelet, akkor annak tartalma már nem titok, de ha a levél tartalmát nem módosítja, s úgy kézbesíti, az üzenet hiteles marad.

Azt mondjuk, hogy a támadó feltörte a titkosító algoritmust, ha „gyorsan” meg tudja állapítani egy lehallgatott üzenet nyílt tartalmát, függetlenül attól, hogy éppen melyik kulcsot alkalmazzák. A gyorsaság olyan időintervallumot jelent, amelyen belül a támadó sikeresen használhatja céljaira a megszerzett információt.

A **titkosítási algoritmus**ok célja a passzív támadások sikerének megakadályozása. Az aktív típusú támadásokat algoritmikus eszközökkel nem akadályozhatjuk meg, de megfelelő kriptoprotokollok alkalmazásával a támadást észlelhetjük. A **protokoll**ok általában véve egy előre meghatározott üzenetcsere folyamatot jelentenek, amelyet kettő vagy több partner bonyolít le kooperatívan valamely feladat végrehajtására. A kriptoprotokollok elemként alkalmazzák a titkosítási algoritmust, s biztosítják a kapcsolat védett felépülését, a kommunikáció alatti aktív támadások észlelhetőségét, összességében garantálják a partnerek és üze-

netfolyamataik hitelességének ellenőrizhetőségét. A titkosítási algoritmusok és a protokollok a kriptográfia tudományának két fő ágát jelentik.

A konvencionális titkosítási algoritmusok mellett a '70-es évek végétől kezdődően rohamosan fejlődik a **nyilvános kulcsú titkosítás**i algoritmusok családja, amelyeket a továbbiakban még részletezni fogunk. Megemlítjük továbbá, hogy a blokktitkosítás mellett létezik az ún. kulcsfolyamatos titkosítás (stream ciphers), amelyet ezen bevezető ismeretanyagban nem részletezünk [40].

5.2. A konvencionális titkosítók analízise

Az információelméleti eszközöket alkalmazó alábbi analízisben azt tételezzük fel, hogy csak rejtett üzenetek állnak az analízis rendelkezésére, vagyis a rejtett szövegű támadást vizsgáljuk.

Jelölje $\mathbf{X}$ és $\mathbf{K}$ a nyílt üzenet és a kulcs valószínűségi változókat, amelyek egy realizációja $\mathbf{x}$ és $\mathbf{k}$ az aktuálisan kódolandó nyílt üzenet és az alkalmazott kulcs. Feltesszük, hogy $\mathbf{X}$ és $\mathbf{K}$ független valószínűségi változók. Az (5.1) leképezésnek megfelelően $\mathbf{Y} = E_{\mathbf{K}}(\mathbf{X})$ a rejtett üzenet valószínűségi változó.

5.1. definíció. ***Tökéletes titkosítás****ról akkor beszélünk, ha*

$$I(\mathbf{X};\mathbf{Y}) = 0, \tag{5.4}$$

azaz, ha $\mathbf{X}$ *és* $\mathbf{Y}$ *kölcsönös információja nulla, amely ekvivalens az* $\mathbf{X}$ *és* $\mathbf{Y}$ *valószínűségi változók függetlenségével, és azzal, hogy a passzív támadó egy olyan csatorna kimenetét látja, amelynek csatornakapacitása nulla (a csatorna zaja csak a* $\mathbf{k}$ *kulcstól függ).*

5.1. tétel. *Tökéletes titkosítás létezik.*

BIZONYÍTÁS: Tekintsük az 5.4. példa algoritmusát:

$$\mathbf{Y} = \mathbf{X} + \mathbf{K}, \tag{5.5}$$

ahol $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_N)$, $\mathbf{X} = (X_1, X_2, \ldots, X_N)$, $\mathbf{K} = (K_1, K_2, \ldots, K_N)$ bináris vektorok és koordinátánként modulo 2 összeadást végzünk. Legyen $\mathbf{K}$ egyenletes eloszlású a bináris N-dimenziós vektorok halmazán, ekkor

$$\begin{aligned}
\mathbf{P}\{\mathbf{Y} = \mathbf{y} \mid \mathbf{X} = \mathbf{x}\} &= \mathbf{P}\{\mathbf{X} + \mathbf{K} = \mathbf{y} \mid \mathbf{X} = \mathbf{x}\} = \\
&= \mathbf{P}\{\mathbf{K} = \mathbf{y} - \mathbf{x} \mid \mathbf{X} = \mathbf{x}\} = \\
&= \mathbf{P}\{\mathbf{K} = \mathbf{y} - \mathbf{x}\} = \\
&= \frac{1}{2^N},
\end{aligned}$$

ahol felhasználtuk **X** és **K** függetlenségét. Innen

$$\begin{aligned}\mathbf{P}\{\mathbf{Y}=\mathbf{y}\} &= \sum_{\mathbf{x}}\mathbf{P}\{\mathbf{Y}=\mathbf{y}\mid\mathbf{X}=\mathbf{x}\}\mathbf{P}\{\mathbf{X}=\mathbf{x}\}= \\ &= \frac{1}{2^N}= \\ &= \mathbf{P}\{\mathbf{Y}=\mathbf{y}\mid\mathbf{X}=\mathbf{x}\}\end{aligned}$$

adódik, azaz **Y** és **X** függetlenek. ■

Érdemes észrevenni, hogy **X** és **Y** függetlenségéhez nem kellett semmit feltételezni az **X** eloszlásáról, továbbá, hogy a rejtett üzenet valószínűségi változó is egyenletes eloszlású lesz. Azonban ahhoz, hogy a szóban forgó (Vernam) titkosító realizálható legyen, minden egyes üzenet kódolásához új kulcs szükséges. Így az adási és vételi oldalon nagy mennyiségű kulcsot kellene tárolni vagy egy védett csatornán továbbítani. Pontosabban a szükséges kulcsbitszám azonos az átküldendő nyílt adatfolyam bitjeinek számával.

Az alábbi tételből világossá válik, hogy ez a hátrány minden tökéletes titkosítóra fennáll.

5.2. tétel. *Tetszőleges tökéletes titkosító algoritmus esetén*

$$H(\mathbf{K})\geq H(\mathbf{X}). \tag{5.6}$$

BIZONYÍTÁS: (5.4) felhasználásával azt kapjuk, hogy

$$H(\mathbf{X})=H(\mathbf{X}\mid\mathbf{Y})+I(\mathbf{X};\mathbf{Y})=H(\mathbf{X}\mid\mathbf{Y}),$$

továbbá

$$\begin{aligned}H(\mathbf{X}\mid\mathbf{Y}) &\leq H(\mathbf{X},\mathbf{K}\mid\mathbf{Y})= \\ &= H(\mathbf{K}\mid\mathbf{Y})+H(\mathbf{X}\mid\mathbf{Y},\mathbf{K})= \\ &= H(\mathbf{K}\mid\mathbf{Y})\leq \\ &\leq H(\mathbf{K}),\end{aligned}$$

ahol felhasználtuk az $\mathbf{X}=D_{\mathbf{K}}(\mathbf{Y})$ alapján fennálló

$$H(\mathbf{X}\mid\mathbf{Y},\mathbf{K})=0$$

egyenlőséget. ■

Az 5.2. tétel felhasználásával bináris esetben a

$$H(\mathbf{X})\leq H(\mathbf{K})=H(K_1,K_2,\ldots,K_N)\leq|\mathbf{K}|, \tag{5.7}$$

azaz a $H(\mathbf{X}) \leq |\mathbf{K}|$ egyenlőtlenséget kapjuk. Tehát bináris tökéletes titkosító algoritmus esetén legalább annyi bináris digitből kell állnia a kulcsnak, amennyi információs bitet hordoz a nyílt üzenet.

5.2. definíció. *Minimális tökéletes titkosító algoritmusról beszélünk, ha*

$$|\mathbf{K}| = \lceil H(\mathbf{X}) \rceil, \tag{5.8}$$

ahol $\lceil v \rceil$ *a* v *valós szám felső egész részét jelöli.*

Az (5.7) és (5.8) formulák alapján látható, hogy a kulcsméret csökkentése érdekében adattömörítés szükséges, vagyis az, hogy a $\frac{H(\mathbf{X})}{M}$ arány minél közelebb essen az 1-hez. Ideális tömörítést és Vernam-titkosítást alkalmazva minimális tökéletes titkosítót készíthetünk, hiszen ekkor

$$|\mathbf{K}| = M = N \simeq H(\mathbf{X}) \tag{5.9}$$

állna fenn. Ezért kell az üzenetet először tömöríteni, és utána titkosítani.

Shannon vezette be a gyakorlati titkosság és a feltétel nélküli titkosság fogalmát.

5.3. definíció. *Egy titkosítási algoritmust* ***gyakorlati titkosság****ot nyújtónak nevezünk, ha feltörése irreálisan nagy számítási / tárolási kapacitást kíván.*

5.4. definíció. *A* ***feltétel nélküli titkosság*** *azt jelenti, hogy a megszerezhető információ (pl. rejtett üzenetek sorozata rejtett üzenetű támadást tekintve) mennyisége elvileg sem elegendő a feltöréshez bármekkora számítási kapacitás is állna rendelkezésünkre.*

Az 5.1. definíció szerinti tökéletes titkosító feltétel nélküli titkosságot biztosít. Adott algoritmusok számításigényének elméleti meghatározását egy rohamosan fejlődő tudományág, az algoritmusok komplexitáselmélete tűzte ki célul. Ezen eredmények alapján egyes nyilvános kulcsú titkosító algoritmusok feltörhetőségét bebizonyították.

Az alábbiakban bevezetésre kerülő nyilvános kulcsú titkosítási algoritmusok gyakorlati titkosságot kívánnak nyújtani. Az algoritmusok alapja tipikusan egy közismerten igen nehéz matematikai probléma, a tárgyalandó RSA-algoritmus például az igen nagy ($\approx 10^{300}$) egészek prímfaktorokra bontásának nagy számításigényére alapoz.

5.3. Nyilvános kulcsú titkosítás

A **nyilvános kulcsú titkosítás** alapgondolata, hogy gyakorlati titkosságot tud nyújtani anélkül, hogy a titkos üzenetküldést megelőzően a kommunikáló partnerek bármiféle titkos kulcsot cseréltek volna egymással, mint ahogy ez az előzetes kooperáció előfeltétel a konvencionális titkosítás esetén. Az A nyilvános kulcsú titkosító két kulccsal dolgozik, egy nyilvános ($\mathbf{k}_A^p$) és egy titkos ($\mathbf{k}_A^s$) kulccsal. A nyilvános kulcsot a kódoláshoz, a titkos kulcsot a dekódoláshoz használjuk. Ha A és B titkosítók felhasználásával történik titkos üzenetváltás, akkor B az

$$\mathbf{y} = E_{\mathbf{k}_A^p}(\mathbf{x})$$

rejtett üzenetet küldi A-nak, amit $\mathbf{k}_A^p$ nyilvánossága miatt megtehet, s ebből a rejtett üzenetből

$$\mathbf{x} = D_{\mathbf{k}_A^s}(\mathbf{y})$$

dekódolással nyeri vissza A a nyílt üzenetet. A kódolás a nyilvános kulcs ismeretében „könnyű” feladat, míg a dekódolás a rejtett kulcs ismeretének hiányában gyakorlatilag nem végrehajtható („nehéz feladat”).

Egy $A, B, C, \ldots$ titkosítókat (felhasználókat) tartalmazó rendszerben egy nyilvános kulcstárba tesszük le a $\mathbf{k}_A^p, \mathbf{k}_B^p, \ldots$ nyilvános kulcsokat, ahonnan bármelyik felhasználó kiolvashatja annak a felhasználónak a nyilvános kulcsát, akinek rejtett üzenetet kíván küldeni.

Az egyes felhasználók helyben generálhatják a $\left(\mathbf{k}_V^p, \mathbf{k}_V^s\right)$ kulcspárt, s ebből a nyilvános részt közzéteszik, míg a másikat titokban tartják. Nagyon fontos észrevennünk azt a kitételt, miszerint a nyilvános kulcstárból csak olvasni szabad, s védeni kell azt a nyilvános kulcsokkal történő manipulációktól (pl. cserétől). Ez azt jelenti, hogy ugyan a kommunikációt megelőzően nem kell titkos kulcscserét végezni, hiszen a nyilvános kulcstárból olvasás egy nyílt előzetes információcserének felel meg, de megmarad az a feladat, hogy ezen nyílt előzetes információ hitelességét biztosítani kell.

Hitelesítési feladatban jól alkalmazható a D dekódoló transzformáció. Tegyük fel, hogy A az $\mathbf{x}$ üzenetet kívánja B-nek elküldeni olyan módon, hogy egyúttal „aláírását” is elhelyezze a rejtett üzenetben. Ezt úgy teheti meg, hogy az $\mathbf{y} = E_B(D_A(\mathbf{x}))$ alakú üzenetet küldi el. $\mathbf{y}$ alapján a $D_A(\mathbf{x})$ tartalmat csak B tudja dekódolni, s nyilván $D_A(\mathbf{x})$ az a leképezés, amely egyértelműen kapcsolódik a küldő személyéhez, A-hoz, és a küldött üzenethez, $\mathbf{x}$-hez. A dekódoló transzformációnak a nyílt üzenetre történő alkalmazásával digitális aláírást generálhatunk. Hasonló aláírások hitelesíthetik a nyilvános kulcstár elemeit. Ezen alkalmazásokra a kriptográfiai protokollok kapcsán még visszatérünk.

A. Shamirtól származik az a rendkívül érdekes eljárás, amelynek a felhasználásával mindennemű előzetes kulcscsere nélkül titkos üzenetváltás történhet partnerek között, feltéve az esetleges behatolóról, hogy passzív, azaz a csatornán folyó üzenetváltás lehallgatására képes csak. Ez az eljárás elvi továbblépést jelent a nyilvános kulcsú algoritmusokkal szemben is, hiszen még nyilvános információt sem kell előzetesen a partnerek tudomására hozni.

A felhasználók mindegyike egy titkos kulccsal rendelkezzen, s tegyük fel, hogy a rendszerben alkalmazott $E_{\mathbf{k}}(\mathbf{x})$ kódoló transzformáció kommutatív, azaz tetszőleges $\mathbf{x}$ nyílt üzenet, $\mathbf{k}_A$ és $\mathbf{k}_B$ kulcspár esetén

$$E_{\mathbf{k}_A}\left(E_{\mathbf{k}_B}(\mathbf{x})\right) = E_{\mathbf{k}_B}\left(E_{\mathbf{k}_A}(\mathbf{x})\right), \tag{5.10}$$

azaz a kétszeres kódolás eredménye független legyen a kulcsválasztás sorrendjétől. Ekkor az ún. „háromlépéses” eljárás alkalmazásával A felhasználó B-nek a következőképp küldheti el $\mathbf{x}$ üzenetét:

1. $A \to B$: $\quad \mathbf{y}_1 = E_{\mathbf{k}_A}(\mathbf{x})$
2. $B \to A$: $\quad \mathbf{y}_2 = E_{\mathbf{k}_B}\left(E_{\mathbf{k}_A}(\mathbf{x})\right) = E_{\mathbf{k}_A}\left(E_{\mathbf{k}_B}(\mathbf{x})\right)$
3. $A \to B$: $\quad \mathbf{y}_3 = D_{\mathbf{k}_A}\left(E_{\mathbf{k}_B}\left(E_{\mathbf{k}_A}(\mathbf{x})\right)\right) = E_{\mathbf{k}_B}(\mathbf{x})$

Ez a természetes ötlet még szemléletesebben is leírható. Nevezetesen képzeljük el, hogy egy lelakatolható ládába helyezi el A az $\mathbf{x}$ üzenetet, s lelakatolja $\mathbf{k}_A$ kulcsával (1. lépés), majd elküldi B-nek. B nem próbálkozik a nyitás számára is lehetetlen feladatával, hanem inkább még a saját $\mathbf{k}_B$ kulcsával is lelakatolja a ládát (2. lépés), majd visszaküldi azt A-nak. A leveszi a saját lakatját, s a ládát, amelyen már csak B lakatja maradt, visszaküldi B-nek (3. lépés), aki ezután már könnyen kinyithatja azt.

A fentiekből úgy tűnhet, hogy megtaláltuk a tökéletes megoldást, hiszen valóban nem kellett előzetes kulcscsere a partnerek között (sem nyilvános, sem titkos). De mint említettük, ezen protokoll praktikus alkalmazhatóságát az gátolja, hogy teljesen ki van szolgáltatva az aktív támadásnak, hiszen az üzenetküldés nyilvános hálózaton történik. A fenti szemléletes leírásmód szóhasználatával gondoljunk csak arra, hogy pl. a postás, akivel a ládát kívántuk B-hez eljuttatni a saját lakatját teszi a ládára a 2. lépésben, s így adja azt vissza A-nak.

Az (5.10) kommutativitási tulajdonságnak eleget tesz az egyszerű

$$E_{\mathbf{k}}(\mathbf{x}) = \mathbf{x} + \mathbf{k} \pmod 2$$

bitenkénti modulo 2 vektorösszeadást alkalmazó kódolás, ugyanis

$$(\mathbf{x} + \mathbf{k}_A) + \mathbf{k}_B = [(\mathbf{x} + \mathbf{k}_B) + \mathbf{k}_A] \pmod 2,$$

mivel a mod 2 összeadás asszociatív.

Könnyen beláthatjuk azonban, hogy ezen kódolás alkalmazása a háromlépéses eljárásban még a legegyszerűbb passzív támadásnak, a pusztán rejtett szövegre alapozó támadásnak sem áll ellen. Ugyanis az eljárás 1., 2., 3. lépése során a támadó megfigyelve az

$$\mathbf{y}_1 = \mathbf{x} + \mathbf{k}_A, \qquad \mathbf{y}_2 = \mathbf{x} + \mathbf{k}_A + \mathbf{k}_B, \qquad \mathbf{y}_3 = \mathbf{x} + \mathbf{k}_B$$

rejtett üzeneteket, és képezve azok mod 2 összegét,

$$\mathbf{y}_1 + \mathbf{y}_2 + \mathbf{y}_3 = \mathbf{x},$$

az $\mathbf{x}$ nyílt üzenethez jutunk, sőt a kulcsok is kinyerhetők: $\mathbf{k}_B = \mathbf{y}_1 + \mathbf{y}_2$ és $\mathbf{k}_A = \mathbf{y}_2 + \mathbf{y}_3$. Vegyük észre, hogy az alkalmazott kódolás nem véletlen átkulcsolás, hiszen kétszer is alkalmaztuk ugyanazon kulcsot.

Kommutatív egy $x^e \mod n$ alakú kódolás (modulo hatványozás) is, ahol x, e, n természetes számok, azaz

$$(x^{e_1})^{e_2} = (x^{e_2})^{e_1} \mod n.$$

Ilyen típusú műveletet használ az 5.4. szakaszban részletesen kifejtett RSA-kódolás.

Elemi számelméleti eredmények

Ahhoz, hogy megérthessük a Rivest–Shamir–Adleman (RSA) algoritmus működését, néhány elemi számelméleti eredmény felidézése szükséges.

5.3. tétel (A maradékos osztás tétele). *Tetszőleges a és b, $b > 0$ egészekre egyértelműen létezik q és r egész, hogy*

$$a = bq + r, \qquad 0 \le r < b.$$

BIZONYÍTÁS: A létezés könnyen látható, hiszen a $qb \le a < (q+1)b$ egyenlőtlenségnek eleget tevő egészet választhatjuk, s ekkor $r = a - qb$. Az egyértelműség igazolásához tegyük fel, hogy q, r páron kívül létezik q', r' pár is, amelyre $a = q'b + r'$, $0 \le r' < b$. Mivel ekkor $qb + r = q'b + r'$ állna fenn, ezért $r - r' = b(q' - q)$ átrendezéséből ellentmondásra jutunk, hiszen $0 \le r, r' < b$ miatt $r - r' < b$, s ugyanakkor teljesülnie kell a $b \mid r - r'$ oszthatóságnak, ami csak $r = r'$ esetén lehetséges. Az $r = r'$ egyenlőségből viszont $q = q'$ is következik. ■

5.5. definíció. *Az a számot a b és c szám közös osztójának nevezzük, ha $a \mid b$ és $a \mid c$ teljesül. Ha b és c közül legalább az egyik nem nulla, akkor a közös osztóik legnagyobbikát b és c legnagyobb közös osztójának nevezzük és (b,c)-vel jelöljük.*

5.6. definíció. *Azt mondjuk, hogy a és b relatív prímek, ha $(a,b) = 1$.*

5.4. tétel (Az euklidészi algoritmus). *Adott b és $c > 0$ egészekre egymás után többször alkalmazzuk a maradékos osztást, s ezzel az egyenletek következő sorozatát kapjuk:*

$$\begin{aligned}
b &= cq_1 + r_1 && 0 < r_1 < c \\
c &= r_1q_2 + r_2 && 0 < r_2 < r_1 \\
r_1 &= r_2q_3 + r_3 && 0 < r_3 < r_2 \\
&\vdots \\
r_{n-2} &= r_{n-1}q_n + r_n && 0 < r_n < r_{n-1} \\
r_{n-1} &= r_nq_{n+1} + 0 && 0 = r_{n+1}
\end{aligned}$$

A b és c számok legnagyobb közös osztója r_n, az osztási eljárás legutolsó nem nulla maradéka, azaz $(b,c) = r_n$.

BIZONYÍTÁS:

a) $r_1, r_2, \ldots$ pozitív egészek szigorúan monoton csökkenő sorozata, ezért léteznie kell olyan egésznek, amelyre $r_{n+1} = 0$.

b) A $b = cq_1 + r_1$ egyenletre tekintve láthatjuk, hogy ha $x \mid b$ és $x \mid c$ fennáll, akkor $x \mid r_1$ is fenn kell álljon, s viszont, ha $y \mid c$ és $y \mid r_1$, akkor ebből $y \mid b$ is következik. Ez pedig azt jelenti, hogy b, c közös osztóinak halmaza megegyezik c, r_1 közös osztóinak halmazával, ezért $(b,c) = (c,r_1)$ is igaz. Az euklidészi algoritmus további egyenleteit is tekintve a

$$(b,c) = (c,r_1) = (r_1,r_2) = \cdots = (r_{n-1},r_n) = (r_n,0) = r_n$$

egyenlőségi láncot kapjuk, tehát $(b,c) = r_n$ valóban igaz. ■

5.1. következmény. *Tetszőleges b és c egészekre, amelyek közül legalább az egyik nem nulla, léteznek s és t egészek, hogy*

$$(b,c) = sb + tc. \tag{5.11}$$

BIZONYÍTÁS: Az euklidészi algoritmus egyenleteiből egymásba helyettesítésekkel a következő egyenletsort kaphatjuk:

$$\begin{aligned} r_1 &= b - q_1 c = b + (-q_1)c \\ r_2 &= c - q_2 r_1 = (-q_2)b + (1 + q_1 q_2)c \\ &\vdots \\ r_n &= sb + tc, \end{aligned}$$

tehát az összes r_i, így r_n is felírható a b és c számok lineáris kombinációjaként. ■

5.5. példa. Határozzuk meg a 8387 és 1243 legnagyobb közös osztóját!
Az euklidészi algoritmus alapján:

$$\begin{aligned} 8387 &= 1243 \cdot 6 + 929 \\ 1243 &= 929 \cdot 1 + 314 \\ 929 &= 314 \cdot 2 + 301 \\ 314 &= 301 \cdot 1 + 13 \\ 301 &= 13 \cdot 23 + 2 \\ 13 &= 2 \cdot 6 + 1 \\ 2 &= 1 \cdot 2 + 0, \end{aligned}$$

így $(8387, 1243) = 1$, azaz relatív prímek. Ezen számolást felhasználva az (5.11) előállítás a következőképp kapható $b = 8347, c = 1243$ jelölésekkel:

$$\begin{aligned} 929 &= b - 6c \\ 314 &= -b + 7c \\ 301 &= 3b - 20c \\ 13 &= -4b + 27c \\ 2 &= 95b - 641c \\ 1 &= -574b + 3873c, \end{aligned}$$

azaz $(8387, 1243) = -574 \cdot 8387 + 3873 \cdot 1243$.

5.7. definíció. *Ha az m nem nulla egész osztja az $a - b$ különbséget, akkor azt mondjuk, hogy az a kongruens b-vel modulo m, és ezt a következőképpen jelöljük:* $a = b \pmod{m}$.

5.8. definíció. *Azt mondjuk, hogy b modulo m inverze a-nak ha*

$$ab = 1 \pmod{m}.$$

5.6. példa. Az 5.5. példa alapján a 3873 az 1243 modulo 8387 inverze, hiszen $3873 \cdot 1243 = 1 \mod 8387$.

5.5. tétel. *Az a modulo m inverze akkor és csak akkor létezik, ha $(a,m) = 1$. Ha létezik inverz, akkor az egyértelmű az m-nél kisebb pozitív egészek között.*

BIZONYÍTÁS: Ha létezik b, amelyre $ab = 1 \pmod{m}$, akkor az ezzel ekvivalens $ab - qm = 1$ alakból $(a,m) \mid 1$, azaz $(a,m) = 1$ következik. Megfordítva, ha $(a,m) = 1$, akkor az (5.11) felhasználásával $sa + tm = 1$ kapható, ahol $b = s$ választással $ba = -tm + 1$, azaz $ba = 1 \pmod{m}$ következik.

Az egyértelműséget a következőképp bizonyíthatjuk. Ha a $b' \neq b$, $0 < b$, $b' < m$ egészekre $ab = ab' = 1 \pmod{m}$, akkor $a(b - b') = 0 \pmod{m}$, ami azt jelentené, hogy $m \mid a(b - b')$. De mivel $(a,m) = 1$, ezért ebből $m \mid b - b'$ következik, ami viszont $|b - b'| < m$ miatt lehetetlen. ■

5.6. tétel (Fermat-tétel). *Ha a c egész nem osztható a p prímmel, akkor*

$$c^{p-1} = 1 \pmod{p}. \tag{5.12}$$

BIZONYÍTÁS: Tetszőleges c egészre a $c, 2c, 3c, \ldots, (p-1)c$ számok különböznek modulo p. Ha ugyanis $ic = jc \pmod{p}$, $i \neq j$, $0 < i$, $j < p$ fennállna, akkor az $(i - j)c = qp$ és $|i - j| < p$, $p \nmid c$ összefüggésekből ellentmondásra jutnánk. Így tehát a $c, 2c, \ldots, (p-1)c$ számok modulo p az $1, 2, \ldots, p-1$ számok egy permutációját adják. A kongruencia definíciójából következik, hogy ha $a_i = b_i \pmod{p}$, $i = 1, 2, \ldots, n$, akkor $a_1 a_2 \ldots a_n = b_1 b_2 \ldots b_n \pmod{p}$ is fennáll. Tehát $1 \cdot 2 \cdot 3 \cdot \ldots \cdot (p-1) = c \cdot 2c \cdot 3c \cdot \ldots \cdot (p-1)c$, azaz

$$\prod_{i=1}^{p-1} i = c^{p-1} \prod_{i=1}^{p-1} i \pmod{p}$$

fennáll, ahonnan

$$1 = c^{p-1} \pmod{p}.$$ ■

5.7. tétel (Fermat-tétel általánosítása). *Ha p_1 és p_2 különböző prímek, és $(a, p_1 p_2) = 1$, akkor*

$$a^{(p_1-1)(p_2-1)} = 1 \pmod{p_1 p_2}. \tag{5.13}$$

BIZONYÍTÁS: Az $(a, p_1p_2) = 1$ a $p_1 \nmid a$ és $p_2 \nmid a$ állításokat jelenti, így mivel p_1 és p_2 prím, $p_1 \nmid a^{p_2-1}$ és $p_2 \nmid a^{p_1-1}$ is teljesül. A Fermat-tétel felhasználásával tehát

$$\left(a^{(p_2-1)}\right)^{(p_1-1)} = 1 \pmod{p_1}$$

$$\left(a^{(p_1-1)}\right)^{(p_2-1)} = 1 \pmod{p_2}$$

következnek. De ha valamely c egészre $c = 1 \pmod{p_1}$, $c = 1 \pmod{p_2}$, akkor az ekvivalens $p_1 \mid c-1$ és $p_2 \mid c-1$ állításokból $p_1p_2 \mid c-1$ következik. A $c = a^{(p_1-1)(p_2-1)}$ megfeleltetéssel a tétel bizonyítását kaptuk. ■

Az 5.6. és 5.7. tételek az alábbi tétel speciális esetei:

5.8. tétel. *Ha $m = p_1p_2\cdots p_n$, ahol $p_1, p_2, \ldots, p_n$ különböző prímek és $(a, m) = 1$, akkor*

$$a^{\phi(m)} = 1 \pmod{m}, \tag{5.14}$$

ahol $\phi(m) = (p_1-1)(p_2-1)\cdots(p_n-1)$ az Euler-függvény. (Ez a tétel az Euler-tétel speciális esete.)

A továbbiakban csak az 5.7. tétel általánosságára lesz szükségünk, ezért az 5.8. tétel bizonyítását (lásd [34]), amely az 5.7. tételéhez hasonlóan történhet, elhagyjuk, s csak a $\phi(p_1p_2) = (p_1-1)(p_2-1)$ rövidített jelölést használjuk. Az eddigiekben bevezetett számelméleti alapok birtokában megérthetjük az RSA-algoritmust.

5.4. RSA-algoritmus

Az algoritmus lépései a következők:

1. Válasszunk véletlenszerűen két nagy prímszámot p_1-et és p_2-t.
2. Kiszámítva az $m = p_1p_2$ és $\phi(m) = (p_1-1)(p_2-1)$ paramétereket, válasszunk véletlenszerűen egy e, $1 \le e < \phi(m)$ egészet úgy, hogy

   $$(\phi(m), e) = 1 \tag{5.15}$$

   teljesüljön.
3. Számítsuk ki e inverzét modulo $\phi(m)$:

   $$d = e^{-1} \pmod{\phi(m)} \tag{5.16}$$

   (az 5.5. tétel miatt létezik).

4. Az m, e egészeket nyilvánosságra hozzuk, míg a d, p_1, p_2 értékeit titokban tartjuk (azaz $\mathbf{k}^p = (m, e)$, $\mathbf{k}^s = (d, p_1, p_2)$).

5. A titkosító kódolás az $1 \leq x < m$ nyílt üzenetre egy $1 \leq y < m$ rejtett üzenetet ad, ahol
$$y = x^e \pmod{m}, \tag{5.17}$$
míg a dekódolás az
$$x = y^d \pmod{m} \tag{5.18}$$
inverz művelet.

(Az x és y skalár jelölést alkalmazzuk a nyílt ill. a rejtett üzenetre az RSA esetén, kihangsúlyozandó, hogy nemnegatív egész számokként végezzük velük a mod m műveleteket.)

5.1. lemma. *Az (5.17) és (5.18) formulákkal megadott műveletek egymás inverzei.*

BIZONYÍTÁS: Mivel (5.16) a $de = q\phi(m) + 1$ állítással ekvivalens valamely q egészre, ezért az
$$y^d = (x^e)^d = x^{ed} = x^{q\phi(m)+1} \pmod{m}$$
kongruencia láncolatot kapjuk. Külön vizsgáljuk az $(x, m) = 1$ és $(x, m) > 1$ eseteket.

a) $(x, m) = 1$ esetén (5.13) felhasználásával $x^{\phi(m)} = 1 \pmod{m}$, így
$$y^d = x^{q\phi(m)+1} = x(x^{\phi(m)})^q = x \pmod{m}$$
azaz ez esetben a kívánt eredményt kaptuk.

b) $(x, m) > 1$ esetén mivel $m = p_1 p_2$ és $x < m$, ezért vagy $p_1 \mid x$ vagy $p_2 \mid x$. Az általánosság korlátozása nélkül feltehetjük, hogy $p_1 \mid x$, ezért $x = wp_1$ felbontás kapható, ahol $(w, m) = 1$.

Ennek felhasználásával
$$x^{q\phi(m)+1} = w^{q\phi(m)+1} p_1^{q\phi(m)+1} \pmod{m}. \tag{5.19}$$
Mivel $(w, m) = 1$, ezért (5.19) jobb oldali első tényezőjére az 5.7. tétel felhasználásával
$$w^{q\phi(m)+1} = w \pmod{m} \tag{5.20}$$

adódik. A Fermat-tétel felhasználásával:

$$p_1^{q\phi(m)+1} = p_1\left(p_1^{q\phi(m)}\right) = p_1\left(p_1^{(p_2-1)}\right)^{q(p_1-1)} = p_1 \pmod{p_2} \quad (5.21)$$

illetve nyilvánvalóan:

$$p_1^{q\phi(m)+1} = 0 = p_1 \pmod{p_1} \quad (5.22)$$

Így mivel $p_1 \mid p_1^{q\phi(m)+1} - p_1$ és (5.21) alapján $p_2 \mid p_1^{q\phi(m)+1} - p_1$ teljesülnek a $p_1 \neq p_2$ prímekre, ezért $p_1p_2 \mid p_1^{q\phi(m)+1} - p_1$ is fennáll. Azaz

$$p_1^{q\phi(m)+1} = p_1 \pmod{m}. \quad (5.23)$$

Az (5.20) és (5.23) eredményeket (5.19)-ben felhasználva

$$x^{q\phi(m)+1} = wp_1 = x \pmod{m}$$

adódik, amit bizonyítani szerettünk volna. ■

Az alábbiakban egy számpéldával szemléltetjük az RSA-algoritmus paraméterszámítását (1–4. lépések), majd a kódolás és a dekódolás műveletét, amelyben a számok nagyságrendjét a lépések szemléltetéséhez kicsire választottuk.

5.7. példa. Legyen $p_1 = 73$, $p_2 = 151$, így $m = 73 \cdot 151 = 11023$, $\phi(m) = (73-1)(151-1) = 10800 = 2^4 \cdot 3 \cdot 5^2 \cdot 9$. Az e paramétert választhatjuk például 11-re, mivel $(10880, 11) = 1$. Az e inverzét modulo $\phi(m)$ az (5.11) alak segítségével állíthatjuk elő, azaz az 5.5. példában látott számításmenetet követve kaphatjuk meg az $es + \phi(m)t = 1$ előállítást:

$$\begin{aligned} 10800 &= 11 \cdot 981 + 9 \\ 11 &= 9 \cdot 1 + 2 \\ 9 &= 2 \cdot 4 + 1 \\ 2 &= 1 \cdot 2 + 0, \end{aligned}$$

ahonnan

$$\begin{aligned} 9 &= 10800 - 981 \cdot 11 \\ 2 &= 11 - 9 = 11 - (10800 - 981 \cdot 11) = -10800 + 982 \cdot 11 \\ 1 &= 9 - 2 \cdot 4 = 10800 - 981 \cdot 11 - 4 \cdot (-10800 + 982 \cdot 11) = \\ &= 5 \cdot 10800 - 4909 \cdot 11, \end{aligned}$$

így $-4909 \cdot 11 = 1 \pmod{10800}$, ezért $d = 10800 - 4909 = 5891 \pmod{10800}$, tehát $d = 5891$.

Nyilvánosságra hozzuk az $e = 11, m = 11023$ egészeket, s titokban tartjuk a $p_1 = 73, p_2 = 151, d = 5891$ egészeket.

Tegyük fel, hogy az $x = 17$ nyílt üzenetet kívánjuk kódolni, ekkor

$$y = 17^{11} \pmod{11023},$$

ahonnan $y = 1782$. A dekódolás az

$$x = 1782^{5891} \pmod{11023} \tag{5.24}$$

művelettel történik.

Az (5.24) modulo hatvány kiszámítását egyszerűsíti a gyorshatványozás („négyzetre emelés és szorzás”) módszere. A kitevőt az

$$5891 = 2^0 + 2^1 + 2^8 + 2^9 + 2^{10} + 2^{12}$$

összegre bonthatjuk a bináris ábrázolása alapján. Ennek alapján az (5.24) hatványt az alábbi alakba célszerű átírni:

$$\begin{aligned} &1782^{2^{12}} \cdot 1782^{2^{10}} \cdot 1782^{2^9} \cdot 1782^{2^8} \cdot 1782^{2^1} \cdot 1782 = \\ &\quad = ((\ldots(1782)^2)^2 1782)^2 1782)^2 1782)^2)^2)^2)^2)^2)^2 1782)^2 1782 \end{aligned}$$

A kiértékelést a legbelső modulo négyzetre emeléssel kezdjük, azaz az első néhány lépés:

$$\begin{aligned} 1782^2 &= 900 \pmod{11023} \\ 900^2 &= 5321 \pmod{11023} \\ (5321 \cdot 1782) = (2242)^2 &= 76 \pmod{11023}, \\ &\vdots \end{aligned}$$

s az utolsó lépés eredményéül a 17-et kapjuk vissza. Így ahelyett, hogy (5.24) mechanikus kiszámításához, azaz a

$$((\ldots(1789 \cdot 1789) \cdot 1789) \cdot 1789) \cdot \ldots) \cdot 1789$$

szorzáshoz szükséges 5890 darab modulo szorzást végeztük volna el, megúsztuk 17 darab modulo szorzással. (Könnyen ellenőrizhető, hogy ezzel a módszerrel legfeljebb a kitevő kettes alapú logaritmusának kétszerese számú modulo szorzásra van szükség.)

Az egyirányú függvény

Mai ismereteink szerint e és m nyilvános adatok birtokában az
$f : \{1,2,\ldots,m-1\} \to \{1,2,\ldots,m-1\}$,

$$f(x) = x^e \pmod{m} \tag{5.25}$$

inverzének kiszámítása (azaz egy y értékhez egy, az $y = f(x) \pmod{m}$ összefüggésnek megfelelő x meghatározása) számításigényét tekintve gyakorlatilag megoldhatatlan feladat, ha a p_1 és p_2 prímeket elegendően nagyra választjuk. Erős sejtés, hogy ezen inverzképzés nehézsége ekvivalens az m prímfaktorokra bontásának nehézségével. További sejtés, hogy egész számok faktorizálására szolgáló algoritmusok felhasználásával ma számításigényét tekintve megoldhatatlan feladat a $\log_{10} m \simeq 300$ nagyságrendű egészek felbontása. Mai ismeretek és technológia alapján többszáz év nagyságrendű idő lenne szükséges a számításhoz. Ezen időtartambecslést természetesen elsősorban úgy kell tekintenünk mint egy irreálisan nagy értéket, amely szemlélteti a gyakorlatilag biztos védettség eléréséhez szükséges erős túlméretezést a kriptográfiai kódtervezésben.

A gyakorlati titkosságra visszatérve, elvileg nyilván mód lenne e, m nyilvános adatok birtokában az összes $\{$nyílt üzenet, rejtett üzenet$\}$ pár előzetes kiszámítására, amelyeket tárolva és a rejtett üzenetek valamilyen sorrendjében felsorolva, tetszőleges rejtett üzenet vételekor kiolvashatnánk a nyílt párját. A feltétel nélküli titkosság így nyilván nem állhat fenn, hiszen az megengedné akár az összes ilyen pár tárolását és mégsem juthatnánk közelebb a titok (kulcs) megfejtéséhez.

Az (5.25) modulo hatványozás a fentebb már említett megfelelő paraméternagyságrendek mellett egy példája az úgynevezett egyirányú függvényeknek.

5.9. definíció. *Az invertálható f függvényt **egyirányú**nak nevezzük, ha értelmezési tartományának tetszőleges x elemére $f(x)$ értéket könnyű kiszámítani, míg gyakorlatilag irreális feladat tetszőleges y értékkészletbeli elemhez az $y = f(x)$-nek megfelelő x kiszámítása.*

Az RSA-kódolás esetén könnyű feladatnak számít az (5.25) kódolás, míg a dekódolás csak a titkos dekódoló kulcs, d ismeretében könnyű. Azon egyirányú függvényeket, amelyeket egy információ birtokában könnyű, de annak hiányában gyakorlatilag lehetetlen invertálni csapda típusú egyirányú függvényeknek nevezzük. Ezzel a szóhasználattal élve csapda típusú egyirányú függvény birtokában elvileg tervezhetünk nyilvános kulcsú titkosító algoritmust. Megjegyezzük, hogy a '70-es évektől kezdődően rengeteg erőfeszítés történt ilyen függvények konstruálására, amelyek közül eddig csak a modulo hatványozáson alapuló RSA-kódolás maradt feltöretlen.

Prímek előállítása

Visszatérve az RSA-algoritmus első lépésére, nézzük meg röviden a véletlen prímválasztás kérdését. Bizonyítás nélkül utalunk a számelmélet Csebisev-tételére [30], amely kimondja, hogy $\Pi(n)$-re, az n pozitív egésznél kisebb prímek számára a következő becslés adható, ha n elég nagy:

$$\Pi(n) \simeq \frac{n}{\ln n} \tag{5.26}$$

Említettük, hogy ma teljesen biztonságos az $m = p_1 p_2 \simeq 10^{300}$ nagyságrend, amelyhez $p_i \simeq 10^{150}$ $(\simeq 2^{500})$, $i = 1,2$ nagyságrendű prímeket választottunk. Így annak a valószínűsége, hogy egy véletlenszerűen választott $2^{501} < s < 2^{502}$ (502 bites) szám prím, az (5.26) alapján közelíthető:

$$\frac{\Pi\left(2^{502}\right) - \Pi\left(2^{501}\right)}{2^{502} - 2^{501}} \simeq \frac{2^{501}\frac{1}{501\ln 2}}{2^{501}} \simeq \frac{1}{350},$$

s mivel az adott számtartomány fele páros szám, amelyek nem prímek, elég csak a páratlanok közül válogatnunk, s ezzel megduplázzuk a találási valószínűséget $\simeq \frac{1}{175}$-re. Így tehát átlagosan 175 próbálkozásonként jutunk prímszámhoz a 10^{150} nagyságrendű számok között.

Felmerül a kérdés, hogyan dönthetjük el, hogy a véletlenül kisorsolt egész szám prím-e vagy sem. Véletlenül sorsolt számok közül a prímek valószínűségi alapon történő kiszűrésére több algoritmus is létezik, amelyeknek az alapelvét kívánjuk itt elmondani.

Tegyük fel, hogy az s egészről szeretnénk eldönteni, hogy prím-e. Válasszunk egy úgynevezett bázist, egy b egészet a $2 \leq b < s$ tartományban. A Fermat-tétel alapján tudjuk, hogy ha s prím, akkor

$$b^{s-1} = 1 \pmod{s}. \tag{5.27}$$

Így, ha

$$b^{s-1} \neq 1 \pmod{s}, \tag{5.28}$$

akkor s biztosan nem prím, és b az s összetettségének ún. Fermat-tanúja. Azonban ha (5.27) igaz, akkor csak annyit mondhatunk, hogy lehetséges, hogy s prím.

Ha a fenti tesztet r-szer egymás után megismételjük, és a függetlenül sorsolt $2 \leq b_1, \ldots, b_r < s$ bázisok mindegyikére (5.27) teljesül, akkor az s egészet kis tévedési valószínűséggel prímnek nyilvánítjuk. Az ellenőrzések r számát nyilván úgy szeretnénk beállítani, hogy a tévedés valószínűsége (nem prím elfogadása) minél kisebb legyen [38, 31].

Közös modulus választásának hibája

Tegyük fel, hogy az RSA-kódolást alkalmazzuk egy sokfelhasználós rendszerben, és egy p_1, p_2 prímpárt választunk véletlen szelekcióval a teljes rendszer számára a kulcskiosztó központban. Ekkor az $m = p_1 \cdot p_2$ modulus is közös a rendszerben. Az egyes felhasználóknak a központ függetlenül sorsolja az $e_A, e_B, e_C, \ldots$ nyilvános kódoló paramétereket, majd kiszámolja a megfelelő $d_A, d_B, d_C, \ldots$ titkos dekódoló paramétereket. A nyilvános paramétereket nyilvános, olvasható kulcstárban helyezzük el. Amikor egy új felhasználó a rendszerhez kíván csatlakozni, ezen igényével — a rendszeren kívül — egy kulcskiosztó központhoz fordul, ahol egyrészt bejegyzésre kerül, másrészt megkapja a titkos kulcsát. A közös modulus (m) választásának előnye lehetne, hogy a nyilvános kulcstárban kevesebb adatot kell tárolni (azaz nem kell a különböző $m_A, m_B, m_C, \ldots$ modulusokat tárolni), továbbá elképzelhető, hogy azonos modulus az aritmetika hardver megvalósításában is egyszerűsítést eredményez a felhasználói eszközben. Az alábbiakban egy példán keresztül megmutatjuk, hogy közös modulus választásával súlyos rendszertervezési hibát követnénk el, amelyre egy példát mutatunk. További példa található [31] irodalomban.

Relatív prím kódoló kulcspár esete
Abban az esetben, ha A és B felhasználók e_A és e_B kódoló kulcsa relatív prím és azonos nyílt tartalmú üzenet érkezik mindkét felhasználóhoz (pl. egy C felhasználótól egy körlevél), akkor ezen üzenetet a támadó dekódolni képes anélkül, hogy ez egyben az RSA-kódolás megtörését (azaz m faktorizálását) jelentené.

Legyenek $y_A = x^{e_A} \pmod m$ és $y_B = x^{e_B} \pmod m$ a támadó által megfigyelt, azonos nyílt üzenethez tartozó rejtett üzenetek. Ha $(e_A, e_B) = 1$, akkor az 5.4. tétel következménye alapján léteznek olyan t, s egészek, hogy $t \cdot e_A + s \cdot e_B = 1$ fennáll. Mivel $e_A > 0$ és $e_B > 0$, ezért t és s egyike negatív. Legyen $t < 0$, azaz $t = -1 \cdot |t|$. Feltehetjük, hogy $(y_A, m) = (y_B, m) = 1$, hiszen ellenkező esetben y_A és m (illetve y_B és m) legnagyobb közös osztója a p_1 vagy p_2, aminek megismerése a titkosító transzformáció (RSA-kód) feltörését jelentené. Ha pedig $(y_A, m) = 1$, akkor $y_A \pmod m$ inverze (y_A^{-1}) létezik, ahonnan

$$\left(y_A^{-1}\right)^{|t|} \cdot (y_B)^s = (x^{e_A})^t \cdot (x^{e_B})^s = x^{te_A + se_B} = x \pmod m,$$

azaz anélkül, hogy feltörte volna a támadó az RSA-kódot, képes volt kiszámítani a rejtett üzenet nyílt tartalmát.

A fentiekben elmondott algoritmikus támadás konklúziója az, hogy közös modulus választását mindenképpen el kell kerülni RSA-kódra alapuló rendszerekben.

A kicsi kódoló kulcsok problémája

Az RSA-algoritmus paramétereinek megválasztásánál a kódoló e kitevőjére az $1 \leq e < \phi(m)$, és $(\phi(m), e) = 1$ megkötéseket tettük. Kisebb számításigényű a kódolás, ha e értékét kicsire választjuk, és ezt megtehetjük, mivel ezen választás — a mai ismeretek szerint — nem könnyíti meg az RSA-algoritmus feltörhetőségét. Ha azonban egy sokfelhasználós rendszerben kicsire (például tíznél kisebbre) választjuk valamely felhasználó e paraméterét, az bizonyos körülmények között nyílt üzenet kiszámítására adhat lehetőséget a támadó számára anélkül, hogy feltörné a titkosító kódot.

Itt felelevenítjük a kínai maradéktételt, amely a kódoláselméletben széleskörben alkalmazható.

5.9. tétel (kínai maradéktétel). *Ha az $m_1, m_2, \ldots, m_r$ pozitív egészek páronként relatív prímek, és $a_1, a_2, \ldots, a_r$ tetszőleges egész számok, akkor az*

$$x = a_i \pmod{m_i}, \qquad i = 1, 2, \ldots, r$$

kongruenciarendszernek van közös megoldása. Bármely két megoldás kongruens modulo $m_1 \cdot m_2 \cdots m_r$.

BIZONYÍTÁS: Ha $m = m_1 \cdot m_2 \cdots m_r$, akkor $\frac{m}{m_j}$ egész és $\left(\frac{m}{m_j}, m_j\right) = 1$. Ezért az 5.5. tétel értelmében léteznek olyan b_j egészek, hogy $\frac{m}{m_j} b_j = 1 \pmod{m_j}$. Mivel $m_i \mid \frac{m}{m_j}$, ezért $\frac{m}{m_j} b_j = 0 \pmod{m_i}$, ha $i \neq j$. Ha az x_1 egészet a következőképpen definiáljuk:

$$x_1 = \sum_{j=1}^{r} \frac{m}{m_j} b_j a_j, \tag{5.29}$$

azt kapjuk, hogy

$$x_1 = \sum_{j=1}^{r} \frac{m}{m_j} b_j a_j = \frac{m}{m_i} b_i a_i = a_i \pmod{m_i},$$

azaz x_1 közös megoldása a kongruenciáknak. Ha x_1 és x_2 mindketten megoldásai az $x = a_i \pmod{m_i}$, $i = 1, 2, \ldots, r$ kongruenciáknak, akkor nyilván $x_1 = x_2 \pmod{m_i}$, $i = 1, 2, \ldots, r$. Azaz $m_i \mid (x_1 - x_2)$, $i = 1, 2, \ldots, r$, amiből az $(m_i, m_j) = 1$, $i \neq j$ feltétel miatt $m \mid (x_1 - x_2)$ következik, így $x_1 = x_2 \pmod{m}$. ■

Ha egy sokfelhasználós rendszerben kicsi az a tartomány, amelyből az e kitevőket választjuk, akkor előfordulhat, hogy több felhasználó ugyanazt az e kitevőt kapja, de természetesen függetlenül választják a prímpárt, így különböző

dekódoló kulcsok és modulusok tartoznak hozzájuk (pl. okulva a közös modulus választásának következményeiből). Tegyük fel ekkor, hogy egy felhasználó az $(e,m_1),(e,m_2),\ldots,(e,m_r)$ nyilvános kódolási paraméterekkel rendelkező r ($r \geq e$) számú különböző felhasználónak ugyanazt az x üzenetet küldi, s a támadó rendelkezésére állnak a megfelelő:

$$\begin{aligned} y_1 &= x^e \pmod{m_1} \\ y_2 &= x^e \pmod{m_2} \\ &\vdots \\ y_r &= x^e \pmod{m_r} \end{aligned} \tag{5.30}$$

rejtett üzenetek. Ha $m_1, m_2, \ldots, m_r$ relatív prímek, akkor a kínai maradéktétel felhasználásával kiszámíthatja az $x^e \pmod{m_1 m_2 \cdots m_r}$ hatványt. Mivel $x < \min\{m_i\}$ miatt

$$0 < x^e < m_1 m_2 \cdots m_r,$$

ezért magát az x^e hatványt (és nem csak modulo ekvivalensét) ismeri, így innen már x értékét is kiszámíthatja. Ha azonban az üzenetküldő az üzenet küldési időpontját is elhelyezi az üzenetben (időpecsét), akkor az azonos x üzenetet a $t_1, t_2, \ldots, t_r$ küldési időpontok módosítják, s különböző kódolandó nyílt üzenetekké válnak.

5.5. Kriptográfiai protokollok

Algoritmikus szempontból egy titkosító rendszer két fő komponenst tartalmaz: egyrészt a titkosító kódoló és dekódoló transzformációkat, másrészt **kriptográfiai protokoll**okat. A protokoll algoritmikus lépések sorozata két vagy több résztvevő partner között valamely feladat végrehajtása céljából.

Szükségesek olyan szabályok, amelyek biztosítják, hogy a titkosító transzformációk egy adott alkalmazásban a megkívánt titkosságot vagy hitelességet nyújtsák. A transzformáció többnyire egy kulcsot használ, de a transzformációt végrehajtó algoritmus nem gondoskodik e kulcs védett célbajuttatásáról (kulcskiosztás), a tárolás ideje alatti algoritmikus védelméről (pl. hitelességének biztosítása). Aktív támadások ellen egy titkosító transzformáció önmagában nem véd, így megfelelő szabályokkal kell gondoskodni az üzenetek támadó általi manipulációjának (blokkok nyilvános csatornából történő kivonása, vagy helyettesítése) felfedhetőségéről. Protokollok felhasználásával történik a kommunikáló partner hitelességének megállapítása, az illetéktelen megszemélyesítés felfedése is. A legerősebb titkosító transzformáció sem nyújt védettséget egy hibásan tervezett protokollkörnyezetben.

A kriptográfiai protokollok az algoritmusok igen széles családját foglalják össze. A gyakorlatban leginkább alkalmazott, alapvető protokollok a következő csoportokba sorolhatók:

- partnerhitelesítés
- kulcskiosztás
- üzenetintegritás
- digitális aláírás
- titokmegosztás

Megjegyezzük, hogy számos további speciális célú, gyakorlatban kevésbé használatos protokoll is ismert.

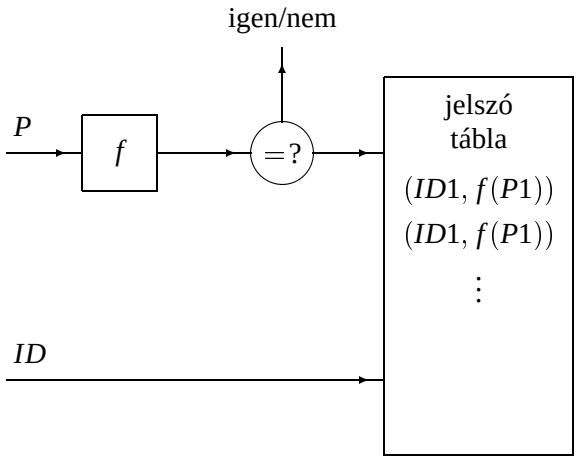
Partnerhitelesítés

A **jelszavas partnerhitelesítés** a legelterjedtebb, régóta ismert azonosítási eljárás. Hosszú idő telt el Ali Baba jelszavától a banki PIN kódokig. A jelszó egy titok, amit a felhasználó megoszt azzal az „erőforrással”, amihez alkalmanként hozzá szeretne férni. A jelszavas rendszerek szokásos problémái: a nem megfelelő jelszóválasztás, a jelszó nyílt alakban történő továbbítása a rendszerbe jutás pontjától (pl. terminál klaviatúra) az ellenőrzés pontjáig (pl. gazdagép), a jelszavak nem eléggé védett tárolása (mind a felhasználó oldalán, mind pedig a jelszófájl tekintetében).

A jelszóellenőrzés folyamatát láthatjuk az 5.4. ábrán. A jelszavak gazdagép oldali védelmén javít azok egyirányú függvénnyel történő leképezése. Ekkor a jelszófájl az egyirányú leképezés eredményét tárolja az egyes felhasználókra, a felhasználói azonosítókkal (ID) együtt. A jelszó (P) továbbra is nyílt alakban érkezik az ellenőrzés helyére, ahol előbb kiszámításra kerül annak egyirányú függvényes leképezése ($f(P)$), majd ennek eredménye kerül összevetésre a táblázat bejegyzésével.

Ezzel a megoldással tehát a jelszófájl illetéktelen olvasása önmagában nem jelent veszélyt, hiszen az egyirányú leképezés gyakorlatilag invertálhatatlan tulajdonsága biztosítja, hogy jelszóhoz a támadó nem férhet hozzá. Ez utóbbi kijelentés azonban csak feltételek mellett igaz: a jelszóméret legyen elég nagy a teljes kipróbálás megakadályozásához, a jelszavak a teljes jelszótérből kerüljenek kiválasztásra.

A nem megfelelő jelszóválasztás azt jelenti, hogy a felhasználók nem véletlenszerűen választanak az adott hosszúságú alfanumerikus karaktersorozatok közül, hanem pl. értelmes szavakat, jellemző dátumokat, ezek triviális kombinációit



5.4. ábra. Egyirányú jelszóellenőrzés.

használják. Ezért a szótár alapú támadások jelentős százalékban sikerre vezethetnek.

Az egyszer használatos jelszó
Ha a jelszót kapcsolatfelvételenként változtatnánk, nyilván semmi értelme nem lenne a jelszó megismerésére irányuló támadásnak, feltéve, hogy a korábbi jelszavakból nem számítható ki a következő jelszó. Ekkor tehát az aktuális jelszót nyíltan is átküldhetjük. Egy **változó jelszavas protokoll** az alábbi:

$$\begin{array}{lll}
\text{Ini1.} & A: & r \text{ generálása} \\
\text{Ini2.} & A \to B: & ID_A, n, y = f^n(r) \\
1. & A \to B: & P_1 = f^{n-1}(r) \\
1.1 & B: & y = f(P_1)? \\
2. & A \to B: & P_2 = f^{n-2}(r) \\
2.1 & B: & y = f^2(P_2)? \\
\vdots & & \\
i. & A \to B: & P_i = f^{n-i}(r) \\
i.1 & B: & y = f^i(P_i)?
\end{array} \tag{5.31}$$

A protokoll inicializálásakor egy r titkos véletlen elemet A felhasználó az f egyirányú függvény n-szeri alkalmazásával leképez, amelynek y eredményét küldi át B-nek, aki ezt tárolja. Az i-edik bejelentkezéskor használandó P_i jelszó y inverze az f^i egyirányú függvényre vonatkozólag. Ezt az inverzet csak A képes kiszámítani, de a számítás helyességét (miután f nyilvános) bárki képes ellenőrizni

az f i-szeri alkalmazásával. Az inverz leképezés praktikus futásidejű algoritmusát természetesen maga A sem képes előállítani, azonban erre nincs is szüksége, mivel az r véletlen elem f függvénnyel történő $n-i$-szeres leképezése ugyanerre vezet. A két félnek szinkront kell tartani a jelszósorszám vonatkozásában. Ha azonban valamely hiba következtében az átvitel során elveszne egy jelszó, akkor A eggyel továbblép a jelszósorszámban, és új jelszót küld, mellette jelezve az egylépéses szinkronhibát.

Partnerhitelesítés nyilvános kulcsú függvények felhasználával
Tekintsük a következő, **„kihívás és válaszvárás”** (challenge & response) típusú partnerazonosítási protokollt, amelyben B kívánja A-t azonosítani:

$$\begin{array}{lll} 1. & B \to A: & R \\ 2. & A \to B: & y = D_A(R) \\ 3. & B: & R = E_A(y)? \end{array} \tag{5.32}$$

A protokoll biztonságosnak tűnik, hiszen A nem játszhat vissza korábban felvett 2. lépésbeli üzenetet, továbbá csak A képes y előállítására a titkos dekódoló kulcsának használatával. A protokoll tehát jónak tűnik, pedig nem feltétlenül az, s ez a véletlen elemre adott dekódolási lépéssel kapcsolatos:

Tegyük fel, hogy a leggyakoribb nyilvános kulcsú rejtjelező algoritmust, az RSA-t kívánjuk használni. Tegyük fel továbbá, hogy C lehallgatott egy korábbi, A számára küldött $y = E_A(x) = x^e \mod n$ rejtjelezett blokkot, ahol e az A nyilvános kulcsa az RSA-algoritmusnak megfelelően. C az x üzenetet szeretné megtudni, amit közvetlenül kiszámítani nem tud, hiszen $x = y^d \mod n$, és a d kitevő az A titkos kulcsa.

C a következő ravasz módon jár el. Választ egy R véletlen természetes számot, ahol $R < n$ és $(R,n) = 1$, majd a következő előkészítő számításokat végzi el:

$$v = R^e \mod n$$

$$w = vy \mod n$$

$$t = R^{-1} \mod n$$

Ezután C elküldi w-t A-nak aláírásra. Itt kapcsolódunk vissza az (5.32) protokollhoz, ugyanis legyen az (5.32) protokollbeli véletlen elem w, amit C küld A-nak „kihívásként” az 1. lépésben. Az (5.32) protokoll 2. lépésében A egy dekódolási lépést hajt végre, azaz visszaküldi a B-t megszemélyesítő támadónak az

$$u = w^d \mod n$$

értéket, ami már lehetővé teszi C számára, hogy megtudja mi is volt az x üzenet, ugyanis:

$$tu = R^{-1}w^d = R^{-1}v^d y^d = v^{-d}v^d y^d = y^d = x \mod n,$$

ahol felhasználtuk, hogy $R = v^d \mod n$.

A következő, módosított változat már biztonságosabb, ahol továbbra is B kívánja A-t azonosítani:

$$\begin{array}{lll} 1. & B \to A: & R_2 \\ 2. & A \to B: & D_A(R_1), \quad A \to B:\ z = D_A(R_1 \oplus R_2) \\ 3. & B: & R_1 \oplus R_2 = E_A(z)? \end{array} \tag{5.33}$$

C nem tud A-tól egy általa előállított R_2 blokkra dekódolást kérni, mivel A csak egy általa módosított $R_1 \oplus R_2$ blokkra alkalmazza a dekódoló transzformációt. Másfelől C nem tudja A-t megszemélyesíteni, mivel ehhez a friss R_2 véletlen blokkot kellene neki R_1 véletlen blokkon keresztül úgy manipulálnia, hogy a 2. lépésben korábbi üzenetek visszajátszásával csalhasson. Erre azonban nem képes, mivel az R_1 blokkot is dekódolt formában kell küldenie.

Ezen módosítás után C már nem képes megszemélyesíteni A-t, mivel nem tudja végrehajtani az 2. lépést, illetve ha az a 2. lépésben egy dedódolás nélkül egy véletlen elemet küld át B-nek, akkor nem tudja a 3. lépést elvégezni. Ha A is azonosítani kívánja B-t, akkor ugyanezen protokollt lejátsszák fordított szereposztással.

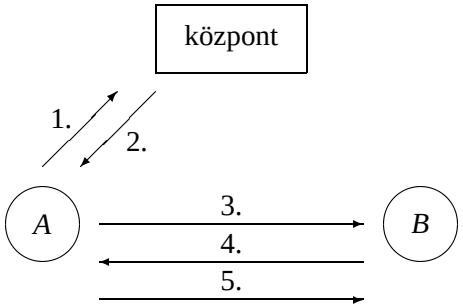
Elkerülhető a fenti támadás olyan módon is, hogy rejtjelezésre és azonosításra más kulcskészletet használunk.

Kulcskiosztás

A konvencionális kódolók közös titkos kulcsot használnak. A jó kulcs valódi véletlen bináris vektor, amelyet nyilván nem lehet szinkronban generálni. Tehát az azt generáló féltől védetten át kell juttatni a partnerhez. A védelem jelenthet fizikai védelmet (például kulcsszállító hardverben „kézben” visszük át a kulcsot a partnerhez), illetve tisztán algoritmikus védelmet, amikor a kulcsot rejtjelezve nyilvános kommunikációs csatornán továbbítjuk. Ezen szakaszban a tisztán algoritmikus módszerekkel foglalkozunk. A konvencionális kódoláson alapuló rejtjelezés ma is alapvető módszer, miután a nyilvános kulcsú, rejtjelezés céljára is alkalmas algoritmusok kódolási sebessége sok alkalmazásban nem elegendően nagy.

Kulcskiosztás konvencionális algoritmussal

A rendszer $A, B, C, \ldots$ felhasználói konvencionális titkosítással kívánnak üzene-



5.5. ábra. Kulcskiosztás központ felhasználásával.

teket váltani egymással. Feltesszük, hogy a felhasználók kötődnek egy-egy terminálhoz és közülük tetszőleges pár tud egymással kommunikálni. A rendszer rendelkezik egy kulcskiosztó központtal, amely lehet egy, a terminálok által elérhető számítógép. A további magyarázatot segíti az 5.5. szemléltető ábra.

A protokoll a következő:

$$\begin{array}{lll} 1. & A \to K_p: & ID_A, ID_B, R_1 \\ 2. & K_p \to A: & E_{TK_A}(R_1, ID_B, DK, E_{TK_B}(DK, ID_A)) \\ 3. & A \to B: & E_{TK_B}(DK, ID_A) \\ 4. & B \to A: & E_{DK}(R_2) \\ 5. & A \to B: & E_{DK}(R_2 - 1) \end{array} \tag{5.34}$$

A protokoll kétféle kulcsot használ: a felhasználói terminálkulcsokat (TK_A, $TK_B, \ldots$) és a kapcsolatkulcsot (DK).

Az R_1 és R_2 véletlen elemek használatának oka a visszajátszásos támadás (replay attack) megakadályozása. Egy ilyen támadásban a C támadó egy korábbi, rögzített üzenetet próbál újrahasználni.

Ha tehát a 2. lépésben A felé az üzenet nem a központtól, hanem C-től származna, ezt A rögtön észrevenné, hiszen nem az 1. lépésben általa elküldött véletlen elemet tartalmazná a 2. lépésben megkapott üzenet.

Ha pedig C küldené el A nevében az 1. lépésbeli üzenetet, akkor — nem ismervén a TK_A kulcsot — nem tudná dekódolni a 2. lépésben vett üzenetből a DK kapcsolatkulcsot.

Ha C a B-t próbálná megszemélyesíteni, nyilván nem tudná dekódolni a 3. lépésbeli üzenetet.

Ha C az A-t próbálná megszemélyesíteni, s a 3. lépéssel indítva egy régebbi 3. lépésbeli üzenetet kíván elküldeni B felé, akkor miután a DK kulcsot ebből

dekódolni nem tudta, nem lesz képes a 4. lépésbeli „friss kihívás” 5. lépésbeli megválaszolására.

A 3–4–5. lépések után B azt tudja, hogy olyan féllel áll szemben, aki ismer egy „valamikor” A-nak küldött DK kapcsolatkulcsot. A protokoll nem gondoskodik arról, hogy A meggyőződjön B azonosságáról, hiszen R_2 véletlen blokk lévén, az nem ellenőrizhető, hogy a 4. lépésbeli rejtett üzenet valóban egy véletlen blokk rejtjelezésével állt elő. A protokoll hátránya még, hogy megbízható központ létére támaszkodik, ami nemcsak annak többlet infrastrukturális kiadását jelenti, de annak a veszélyét is, hogy a központ üzemképtelensége vagy sok kérés miatti leterheltsége az egész rendszer üzemképtelenségéhez vezet.

A protokoll leginkább kifogásolt „gyenge” pontja az, hogy régi, a C támadó által időközben megismert DK kulcs felhasználható támadásra. A támadást C a protokoll 3. lépésétől kezdi, s láthatóan B szemében A felet sikeresen megszemélyesítheti (vegyük észre, hogy az $E_{TK_B}(DK, ID_A)$ korábbi lehallgatott üzenetet kompletten használhatja fel). Ezen támadással szemben megerősíthető a protokoll, ha a központ $E_{TK_B}(DK, ID_A)$ helyett egy időpecséttel ellátott $E_{TK_B}(DK, ID_A, T)$ protokoll-elemet állít elő. Az időpecsét azonban az üzenetcsere gyakoriságoknak megfelelő pontos órát tételez fel, amely biztonságos is abban az értelemben, hogy egy C támadó azt nem képes állítani.

A rendszer üzembe helyezése előtt megfelelő védelmi rendszabályok betartása mellett kerülnek elhelyezésre a mester és terminálkulcsok. A központ mesterkulcsával kódolva átlagosan védett tároló közegen (pl. diszk) tárolhatja az egyes terminálok kulcsait ($E_{MK}(TK_A), E_{MK}(TK_B), \ldots$). A mester- és terminálkulcsokat kulcs titkosítására, a kapcsolatkulcsot a nyílt adatfolyam titkosítására használják. A legnagyobb védettséget a mesterkulcs igényli, viszonylag a legkisebbet a kapcsolatkulcs kapja, azaz a mesterkulcs helyezkedik el a kulcshierarchia tetején. A kapcsolatkulcsok kerülnek leggyakrabban felhasználásra. A hierarchiában alsóbb szintű kulcs kompromittálódása felsőbb szintre nem hat, fordítva viszont igen, hiszen a terminálkulcs megszerzésével dekódolhatjuk az oda érkező kapcsolatkulcsot.

Az (5.34) protokoll véletlen elemeket használ a régebben felvett protokollrészletek visszajátszásával történő támadási kísérletek megakadályozására. Alkalmazhatnánk időpecséteket is védelemül. Az (5.34) kulcscsere protokoll időpecséteket alkalmazó alábbi változata a kulcscsere és azonosítás kettős feladatot kívánja megoldani. A javított protokoll az alábbi lépésekből áll:

$$\begin{array}{lll} 1. & A \to K_p: & ID_A, ID_B \\ 2. & K_p \to A: & E_{TK_A}(T, L, DK, ID_B), E_{TK_B}(T, L, DK, ID_A) \\ 3. & A \to B: & E_{DK}(T', ID_A), E_{TK_B}(T, L, DK, ID_A) \\ 4. & B \to A: & E_{DK}(T'+1) \end{array} \tag{5.35}$$

A fenti lépésekben T a DK kulcs előállításának időpontja, L pedig a DK kulcs élettartama. A 2. lépésben megkapott kódolt üzenetből A megállapíthatja, hogy az órája szerinti időpont belül van-e a $[T, T+L]$ intervallumon. Ugyanezt megteszi B is a 3. lépés után. A 3. lépésbeli $E_{DK}(T', ID_A)$ protokoll-elem küldése egyrészt azért szükséges, hogy B meggyőződhessen arról, hogy ezen lépésbeli átvitelt valóban A végzi (vegyük észre, hogy az 1–2. lépésig egy C támadó is eljuthat), másrészt A T' friss elemmel kihívást is küld egyúttal B felé. Az utolsó lépés az A felet győzi meg arról, hogy partnere valóban B.

Az ilyen, órát feltételező protokollok ki vannak téve annak a veszélynek, hogy az órák pontos együttfutása (szinkronizáltsága) valamilyen rendszerbeli hiba vagy szándékos beavatkozás miatt megszűnik. Ha a küldő órája siet a vevő órájához képest, egy C támadó, lehallgatva az üzenetet, azt egy visszajátszó támadásban felhasználhatja.

Kulcskiosztás nyilvános kulcsú algoritmussal

A **„támadó középen" támadás** a kulcscsere vonatkozásában azt jelenti, hogy a támadó A és B legális felek közé áll, azok kezdődő kulcscsere protokolljába megpróbál bekapcsolódni oly módon, hogy észrevétlenül rávegye a legális feleket egy általa is ismert kulcs használatára. Ezen támadásnál tehát nem a támadó kezdeményezi a kulcscserét, hanem szinkronban belép A és B beszélgetésébe.

Tekintsük az alábbi protokollt:

$$\begin{array}{lll} 1. & A \to B: & ID_A, k_A^p, C_A \\ 2. & B \to A: & ID_B, k_B^p, C_B \\ 3. & A \to B: & E_B(R_1) \\ 4. & B \to A: & E_A(R_2) \\ 5. & A, B: & k = F(R_1, R_2) \end{array} \tag{5.36}$$

ahol C_A és C_B a k_A^p és k_B^p nyilvános kulcsok tanúsítványai (certificate), azaz az adott nyilvános kulcsokra egy központ által előállított digitális aláírások. Ezen aláírást a központ $\{ID_A, k_A^p\}$ együttesére adja. A központ nyilvános kulcsát a rendszer minden résztvevője ismeri. Az F leképezés szerepe annak biztosítása, hogy a k kulcs létrehozásában mindkét fél „egyforma mértékben" vehessen részt, így például választható az R_1, R_2 véletlen bináris vektorok koordinátánkénti modulo 2 összeadása.

Ha a protokoll 1. és 2. lépésében nem kerülne átküldésre a hiteles kulcstanúsítvány (C_A, C_B), akkor a nyilvános kulcsokat egy középen álló támadó sikerrel manipulálhatná általa is ismert kapcsolatkulcs megbeszélésére.

Kommutatív egyirányú függvényre is építhető kulcscsere protokoll. Az egyik legismertebb kommutatív egyirányú függvény a diszkrét hatványozás. Ha csak

passzív támadásra kell felkészülni, akkor bárminemű kulcscsere illetve központ segítsége nélkül képes két fél — A és B — arra, hogy közös véletlen elemet, azaz közös kapcsolatkulcsot „megbeszéljen”.

Válasszunk egy „nagyméretű” véges testet, jelölje ezt GF(q), jelölje g ennek egy — nem titkos — primitív elemét. A protokoll a következő:

$$\begin{array}{lll} 1. & A \to B: & g^{R_1} \\ 2. & B \to A: & g^{R_2} \\ 3. & A: & (g^{R_2})^{R_1} \\ & B: & (g^{R_1})^{R_2} \end{array} \tag{5.37}$$

Mivel a hatványozás kommutatív művelet, ezért a $k = (g^{R_2})^{R_1} = (g^{R_1})^{R_2}$ közös kulcsban állapodhat meg a két fél. Az átviteli csatornában hallgatózó C támadó nem képes az R_1 illetve R_2 véletlen elemeket megállapítani, mivel a diszkrét logaritmusképzés nehéz feladat. A C támadó ugyanakkor képes a „támadó középen” aktív támadásra. Az A féllel a fenti protokoll szerint megbeszél egy k_A kulcsot, B féllel egy k_B kulcsot, majd összekapcsolja őket olyan módon, hogy konvertálja a rejtett üzeneteket az egyik kulcsról a másikra az üzenet irányának megfelelően.

Üzenethitelesítés

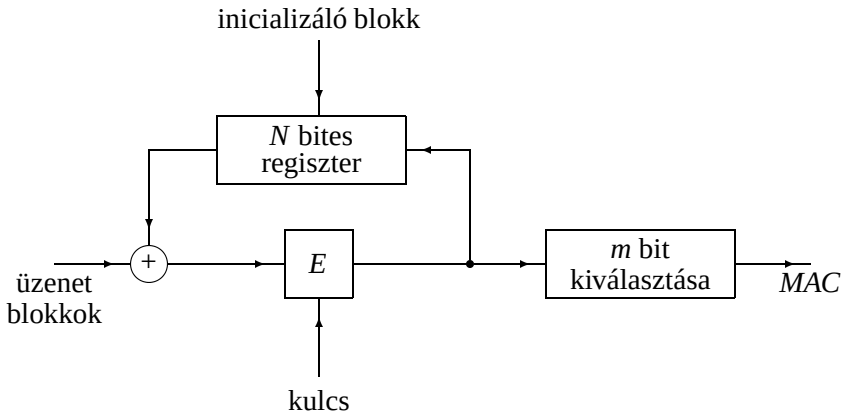
Az üzenethitelesítés feladata az, hogy a vételi oldalon detektálhatóvá tegyük azon eseményeket, amelyek során az átviteli úton az üzenet valamilyen módosulást szenvedett el. Az alábbiakban a legfontosabb módszerek közül a

- kriptográfiai ellenőrző összeg (MAC)
- rejtjelezés
- digitális aláírás
- hash függvény

alapú protokollok alapelveit mutatjuk be.

Kriptográfiai ellenőrző összeg (MAC) alapú üzenethitelesítés
Konvencionális blokk-kódolót alkalmazunk. Tegyük fel, hogy az üzenet r számú blokkra bontható: $x_1, \ldots, x_r$. Ha az utolsó blokk töredék lenne, egészítsük ki zéró bitekkel teljes blokkra. Blokk-kódolónkat CBC (rejtett blokk láncolás) módban használva kriptográfiai ellenőrző összeget (MAC, Message Authentication Code) állítunk elő, s ezt a nyílt alakú üzenethez fűzve továbbítjuk a csatornán, azaz a továbbított hitelesített üzenet:

$$[x_1, \ldots, x_r, MAC(x_1, \ldots, x_r)]$$



5.6. ábra. MAC generálás.

Az eljárást szemléltetjük az 5.6. ábrán. Formálisan $MAC = g(y_r)$, ahol y_r az

$$y_i = E_k(x_i \oplus y_{i-1}), \qquad i = 1, \ldots, r$$

rekurzió r-edik lépésbeli eredménye, továbbá $y_0 = I$ (inicializáló blokk). A g függvény y_r N bitjéből m bitet választ ki, s az eredmény a kriptográfiai ellenőrző összeg. A vételi oldal megismétli az MAC számítást, s az eredményt összehasonlítja a vett MAC-vel. Egyezés esetén fogadja el hitelesnek a vett üzenetet.

Az m értékét akkorára kell választani, hogy elhanyagolható legyen annak a valószínűsége, hogy a C támadó az $[x_1, \ldots, x_r, MAC(x_1, \ldots, x_r)]$ hitelesített üzenetet $[x'_1, \ldots, x'_r, MAC(x'_1, \ldots, x'_r)]$ hiteles formátumú üzenetre cserélje, ahol $[x'_1, \ldots, x'_r]$ a támadó céljainak megfelelő csaló üzenet. Mivel a titkos k kulcsot nem ismeri, azaz MAC számításra nem képes, ezért támadása akkor lehet csak sikeres, ha kisorsolva egy $\mathbf{v}$ m bites vektort, véletlenül $\mathbf{v} = MAC(x'_1, \ldots, x'_r)$ egyenlőség fennáll. Az inicializáló blokk lehet nyilvános az adott CBC módú alkalmazásban. Ennek megfelelően választhatjuk például a csupa zéró blokkot.

Konvencionális rejtjelezéssel történő üzenethitelesítés

Ha konvencionális rejtjelezést alkalmazunk egy nyílt üzenetre, és rejtjelezett formában küldjük el a vételi oldalra, nyilván csak a titkos kulcs birtokában levő, jogosult vevő képes annak tartalmát olvasni. Ezzel tehát a hitelesítési feladatot is megoldottuk, ha a nyílt üzenet redundáns, valamely strukturával vagy formátummal, amelyet a vevő kriptográfián kívüli „eszközzel” ellenőriz. Ha valamely nyelven írott szöveg a nyílt üzenet, akkor ez az „eszköz” az adott nyelven olvasni

tudó ember olvasási képessége. Ha azonban a nyílt üzenet strukturálatlan folyam (legalábbis a rendelkezésre álló „eszközeink” számára), akkor önmagában a rejtjelezés nem ad lehetőséget üzenethitelesítésre. A strukturálttá tétel egyik legegyszerűbb módja valamely lineáris ellenőrző összeg (MDC, Manipulation Detection Code) alkalmazása, s a nyílt szöveg ezen összeggel történő kiegészítése. Ez lehet például az adatátvitelben jólismert ciklikus redundancia karakter (CRC) képzése.

Összegezve: a rejtjelezéssel történő hitelesítés ötlete az, hogy a dekódolt üzenet struktúrája megtörjön egy aktív támadás esetén. (Pl. írott szöveg vagy részletei véletlenszerűvé válnak, beszéd részletei zajjá válnak, stb.)

Sajnos ezzel még nem oldottuk meg megnyugtatóan a feladatot. A gondot az jelenti, hogy a nyílt üzenet tipikusan sokszorta hosszabb, mint a konvencionális blokk-kódoló blokkmérete, azaz tördelni kell a nyílt üzenetet, s blokkonként kódolni. Ezt azonban nem mindegy hogyan tesszük, ha üzenetmódosító támadásra is gondolnunk kell. Nyilván veszélyes lehet az, ha az $x_1, \ldots, x_r$ nyílt üzenet blokkokat külön-külön rejtjük (**ECB**, Electronic Code Book), mert az eredményül kapott $y_1, \ldots, y_r$ rejtett blokkok sorozatának egyes elemei észrevétlenül kihagyhatók, korábbira kicserélhetők, duplikálhatók lehetnek a nyílt üzenet struktúrájától függően. Ezért alkalmazzuk a láncolás módszerét, így a rejtett blokk láncolás (**CBC**, Cipher Block Chaining) módot. Az 5.6. ábrának megfelelő elrendezést tekintsük ismét, elhagyva a kimeneti m bitet kiválasztó elemet. Az I inicializáló blokkot azonban ezesetben körültekintően kell kezelni, hogy egy támadási lehetőséget megakadályozzunk. A láncolás első lépése ugyanis

$$y_1 = E_k(x_1 \oplus I),$$

következésképpen ha C támadó az I blokkot tetszőleges I' blokkra tudja módosítani, akkor tetszés szerinti x_1' blokkra módosítható az x_1 első nyílt blokk, miközben $x_1' \oplus I' = x_1 \oplus I$ fennáll, azaz az y_1 első rejtett szöveg blokk nem változik. S mindezt a kulcs ismerete nélkül teheti meg. Hogy ezt a támadását véghezvigye, el kell érnie, hogy a legális vételi oldalon I' inicializáló blokkot alkalmazzanak a dekódolás során. Ha tehát a vételi oldali dekódoló eszköz nem tulajdonít nagy jelentőséget az I blokk módosítás elleni védelmének, akkor ezzel potenciális támadásra ad lehetőséget. Tehát elvileg nem baj, ha az inicializáló blokkok tára „bárki” által olvasható, de fontos, hogy ne legyen illegálisan írható, azaz hiteles maradjon.

Digitális aláírással történő üzenethitelesítés

A konvencionális rejtjelezők alkalmazásával történő üzenethitelesítés (MAC illetve CBC módú rejtjelezés) csak a két fél számára ad lehetőséget a hitelesség megállapítására. Így nyilván hiába mutat fel egy bíróság előtt például B fél egy

konvencionális rejtjelezésű üzenetet azt állítva, hogy azt *A* küldte, akivel közös titkos kulcsuk volt. A bíróság számára ez nem bizonyíték, hiszen *B* maga is előállíthatta azt. Van azonban olyan módszer, amivel detektálhatóvá tehetünk üzenetmódosító támadást, s ugyanakkor egy harmadik fél felé is bizonyítékul szolgálhat. A módszer a digitális aláírást használja, amelyre a következő pontban térünk vissza.

Üzenethitelesítés titkos kulcs nélkül
Meglepő módon, egyirányú lenyomatkészítő függvény, azaz hash függvény felhasználásával titkos kulcs nélkül is oldhatunk meg üzenethitelesítési feladatot. Ekkor a titkos kulcsból illetve a nyílt üzenetből képzett kriptográfiai ellenőrzőösszeg (MAC) helyett csak a nyílt üzenet hash függvényes lenyomatát használjuk ellenőrző összegként. A kriptográfiai hash függvény lényegében egy olyan egyirányú leképezés, amelynél nehéz feladat azonos lenyomatra (hashképre) vezető ősképeket találni. Azaz ha X jelöli a nyílt üzenetet,

$$X, \mathrm{Hash}(X)$$

kerül átküldésre *A*-tól *B*-hez, ahol Hash egy nyilvánosságra hozott függvény. Érthetetlennek tűnik az állítás, hiszen bárki előállíthat $X', \mathrm{Hash}(X')$ párt tetszőleges, általa választott X' üzenethez. Mégis használható lehet a módszer a gyakorlatban, ha ezen algoritmikus eszközökön túl, *A* és *B* között telefonösszeköttetés is rendelkezésre áll. Ekkor ugyanis *B* felhívja *A*-t, akinek hangját ismeri, azt kéri, hogy az elküldött hash érték elegendő számú — hexadecimális — karakterét olvassa be a telefonba.

Digitális aláírás

Míg az üzenet- és partnerhitelesítés protokollok a kommunikáció időtartamára és a partnerek számára nyújtanak hitelesítési lehetőséget, addig a digitális aláírás az üzenetváltás után és harmadik személy számára is nyújt hitelességellenőrzési lehetőséget. A digitális aláírás protokollok a következő feladatot oldják meg:

a) az aláírás generálása (az üzenetet küldő végzi)

b) az aláírás ellenőrzése (az üzenetvevő által)

c) hitelességgel kapcsolatos vitás kérdések harmadik személy (pl. bíróság) előtti tisztázása.

A c) pontban említett vita tárgya lehet: az aláíró szeretne letagadni egy korábban általa küldött üzenetet, mert tartalma már kedvezőtlen számára. Vád tárgya

lehet az is, hogy a címzett saját céljainak megfelelően módosította a küldött üzenetet.

A digitális aláírás utánozni kívánja a valódi kézjegy tulajdonságait, nevezetesen:

a) legyen könnyen generálható,

b) ne legyen egyik „okmányról" a másikra áthelyezhető (hamisítható), azaz csak a tulajdonosa generálhassa,

c) „bárki" képes legyen ellenőrizni annak hitelességét.

A digitális aláírásnak a fenti közös tulajdonságok mellett van egy igen fontos sajátossága, mégpedig az, hogy nem az üzenet anyagi hordozójához (pl. papír) tartozik, hanem tartalmilag kapcsolódik az aktuális üzenethez, azaz üzenetfüggő.

Nyilvános kulcsú titkosító algoritmus felhasználásával egyszerűen készíthetünk digitális aláírást. Tegyük fel, hogy A az x üzenetet kívánja B-nek elküldeni olyan módon, hogy egyúttal aláírását is elhelyezze a rejtett üzenetben. Ezt elérheti, ha a titkos kulcsát alkalmazva egy dekódolási lépést hajt végre (tegyük fel egyelőre, hogy az x üzenet hossza nem nagyobb, mint a dekódoló transzformáció input mérete). Nyilván

$$D_A(x)$$

függvénye mind az x üzenetnek, mind pedig a titkos dekódoló kulcsnak, tehát csak A képes előállítani azt. Az üzenetvevő B fél ismerve A nyilvános kulcsát képes x visszaállítására egy

$$x = E_A(D_A(x))$$

kódolási lépéssel. Abban az esetben, ha x egy B által is ismert formátummal (általánosabban redundanciával) rendelkező üzenet, akkor $D_A(x)$ önmagában az aláírt üzenet, s nem csak az aláírás. Ugyanis csak A képes olyan z, dekódoló output méretű blokkot előállítani, amelyre $E_A(z)$ nem egy véletlenül választott blokk lesz, hanem olyan, amely megfelelő formátummal is rendelkezik. Ha azonban nem tételezhetjük fel, hogy B formátum ellenőrzést végez (pl. a B oldali szoftver erre nem készült fel), akkor egyszerűbb, ha a $D_A(x)$-et csak aláírásnak tekintjük, amit az x üzenethez csatolva küldünk el, azaz ekkor

$$[x, D_A(x)]$$

az aláírással hitelesített üzenet. B fél ekkor az $E_A(D_A(x))$ kódolási lépés eredményét veti össze a nyíltan is megérkezett x üzenettel.

Ezzel teljesítjük a következő — aláírással szembeni — elvárásokat:

a) az aláírás hitelessége biztonsággal ellenőrizhető: B az A nyilvános kulcsát használó kódolási lépéssel bizonyossággal megállapíthatja, hogy a küldő A volt-e

b) az aláírás nem hamisítható: csak A ismeri a szükséges titkos kulcsot

c) az aláírás nem vihető át egy másik dokumentumra: az aláírás függvénye az adott dokumentumnak

d) az aláírt dokumentum már nem változtatható meg: ha megváltoztatják a dokumentumot, ahhoz nem illeszkedik már az aláírás

e) az aláírás letagadhatatlan: B-nek nincs szüksége A-ra, hogy egy harmadik fél számára bebizonyítsa, miszerint A küldte az aláírt dokumentumot.

Mindezen elvi tökéletességek ellenére két ponton tovább érdemes finomítani az aláírás protokollt. Ezek a pontok a

- lenyomatkészítés,
- időpecsét alkalmazása.

Célszerű a dokumentum méretétől függetleníteni az aláírás méretét, s egy alkalmas nyilvános egyirányú dimenziószűkítő függvény (hash függvény) felhasználásával nem az eredeti dokumentumra, hanem annak lenyomatára adni az aláírást. Ekkor az aláírt üzenet

$$[x, D_A(\mathrm{Hash}(x))]$$

alakú. Egy támadó x megfigyelésével a hash függvény nyilvánossága miatt ki tudja számolni $\mathrm{Hash}(x)$ értékét. Sikeres támadáshoz azonban arra van szüksége, hogy egy olyan x' üzenetet találjon, amivel csaló célját elérheti, s ugyanakkor $\mathrm{Hash}(x) = \mathrm{Hash}(x')$, mert ezesetben $[x', D_A(\mathrm{Hash}(x))]$ is hiteles üzenet B szemében. Csakhogy egy elfogadható hash függvény garantálja, hogy gyakorlatilag nem találhatunk azonos hash értékre vezető x' üzenetet. A támadó feladatát még csak tovább nehezíti, hogy egy ilyen azonos hash értékre vezető üzenetnek ráadásul értelmes, sőt céljainak megfelelő csaló tartalmúnak kell lennie.

Időpecsét az aláírásban és a letagadásvédelem
Tegyük fel, hogy A aláírásával hitelesítetten elküldött B-nek egy szerződést, majd egy idő múlva — a körülmények számára kedvezőtlenné válása miatt — szeretné, ha letagadhatná az aláírt szerződés elküldését. Elhíreszteli, hogy már előzőleg kompromittálódott a titkos kulcsa, ezért nem vállal felelősséget a nevében aláírt szerződésért.

Ahhoz, hogy A ne tudja letagadni, hogy ő írta alá a dokumentumot, nyilván nem elég az időpont, de egy harmadik megbízható személy részvételével a probléma megoldható. Legyen ez a személy G. Egyszerűbb jelölés kedvéért az alábbiakban $S_A(x)$ jelölje az A által digitálisan aláírt x dokumentumot, továbbá $V_A(X)$ jelölje az A általi aláírással elátott $X = S_A(x)$ dokumentum ellenőrzését az aláírás hitelessége szempontjából. Tekintsük a következő protokollt:

$$\begin{array}{lll} 1. & A \to G: & U = S_A(I, S_A(x)) \\ 2. & G: & V_A(U) \\ 3. & G \to A: & W = S_G(T, I, S_A(x)) \\ & G \to B: & W = S_G(T, I, S_A(x)) \\ 4. & A: & V_G(W) \\ 5. & B: & V_G(W), I, V_A(S_A(x)) \end{array} \tag{5.38}$$

Az 1. lépésben A aláírja az x dokumentumot, az aláírt dokumentumot ($S_A(x)$) kiegészíti azonosító fejléccel (I), s újra aláírja az eredményt, amellyel U-t kapja. Az azonosító információ minimálisan azt tartalmazza, hogy A B-számára szándékozik aláírt dokumentumot küldeni. A 2. lépésben G ellenőrzi az U által hordozott külső aláírást, valamint az azonosító I információt. G az aláírt x dokumentumot és az I azonosító információt kiegészíti egy időpecséttel (T), majd az eredményt aláírja, s a 3. lépésben elküldi azt mind A-nak mind pedig B-nek. A 4. lépésben A ellenőrzi a G-től érkezett üzenetet, s ha nem ő küldte előzőleg az I azonosítójú, $S_A(x)$ aláírt dokumentumot, akkor azonnal jelzi, hogy kompromittálódott titkos kulcsával visszaéltek. Az 5. lépésben B ellenőrzi G aláírását, az I azonosító információt, majd pedig A aláírását.

Az I azonosító használata némi magyarázatra szorul. Ugyanis úgy gondolhatnánk, hogy a dokumentum (x) elvárhatóan tartalmazza a két fél azonosítóját. Azonban nem feltétlen köthetjük meg, hogy a dokumentumában ki hova tegyen azonosítót, s egy számítógépes, automatikus alírásellenőrző rögzített pozíciójú adatmezőkkel nyilván egyszerűbben dolgozik. Továbbá az 5. lépésben B az I információból tudja meg, hogy A küld számára aláírt dokumentumot, s A nyilvános kulcsával kell a $V_A(S_A(x))$ verifikációs lépést végrehajtania.

Másolható-e a digitális aláírás?
Tekintsünk egy RSA-kódolást használó titkosítást. Egy x nyílt üzenethez csak az A felhasználó tudja előállítani a $D_A(x)$ transzformáltat. Tegyük fel, hogy a sokfelhasználós rendszerünkben egy „közjegyzőt” alkalmazunk, amely a saját titkos k_{KJ}^S kulcsának felhasználásával egy hozzá benyújtott r nyílt üzenethez a $D_{KJ}(r)$ aláírást generálja. Egy támadó egy t üzenetére szeretne aláírást kapni, amelynek azonban a tartalma olyan, amit nyíltan, a felfedődés veszélye nélkül nem mutat-

hat be a közjegyzőnek. Kérdés, hogy van-e mód arra, hogy mégis generáltasson aláírást a közjegyzővel a t üzenetére. Járjon el a támadó a következőképpen:

1. Tetszőlegesen választott x üzenethez az E_{KJ} nyilvános transzformáció ismeretében meghatározza az

   $$y = E_{KJ}(x)$$

   rejtett üzenetet.

2. Az aláírandó t üzenetet módosítja a tartalmában már nem veszélyes t'-re:

   $$t' = y \cdot t \pmod{m_{KJ}}$$

3. A t' üzenetre kér aláírást, azaz a közjegyzővel előállíttatja az

   $$s' = D_{KJ}(t')$$

   aláírást.

4. Az s' aláírás a következő alakba írható:

   $$s' = (t')^d = (y \cdot t)^d = y^d \cdot t^d = x \cdot t^d \pmod{m_{KJ}}$$

   ahonnan

   $$s'' = s' \cdot x^{-1} = t^d \pmod{m_{KJ}}$$

   szorzással előállíthatja a kívánt aláírást. (Mivel x-et a támadó választja, ezért invertálhatóra választhatja.)

A támadó tehát tetszőleges üzenetre képes generálni aláírást anélkül, hogy arról tudna az aláíró. Ez a támadási mód azonban mégis csak elvileg lehetséges. Egy aláírandó üzenetről (dokumentumról) megkövetelhetjük, hogy formátum-megkötéseknek tegyen eleget, azaz tartalmazzon speciális információkat kötött pozíciókban, így pl. felhasználó- és dokumentumazonosítót, dátumot, ahogyan az egy szokásos dokumentumnál is szükséges. Ezt a formátumot ellenőrizze a közjegyző (közjegyző program) is aláírás előtt. Ha ezen formátum-megkötéssel és ellenőrzéssel kiegészítjük az egyszerű aláírásprotokollunkat, akkor a fentebb elmondott támadási módszer már nem vezethet sikerre. Ugyanis a t üzenetet nem tudjuk modulo szorzással egy formátumnak megfelelő t' üzenetbe átvinni (2. lépés). A protokoll tehát olyan megszorítással működteti a dekódoló transzformációt, hogy az értelmezési tartományának csak egy részhalmazából vehetünk elemeket. (Megjegyezzük, hogy ezen támadás algoritmikus ötlete rokon az (5.32) protokoll elleni támadásával.)

Titok megosztása

Mint láttuk, a titkosító transzformációk mindegyike esetén szükséges valamilyen titkos információ, a titkos kulcs, amelyet már nem véd újabb titkosító transzformáció. Ehhez ugyanis újra valamilyen titkos információ kellene s.í.t.. Így ezt a titkos információt másfajta védelemre kell bízni. Leheteséges pl. valamilyen fizikai védelem alá helyezés (memorizálás, felnyitás-biztos dobozba helyezés, stb.). Egy másik módszer úgy igyekszik „feldarabolni" a titkos információt N személy között, hogy abból tetszőlegesen választott K személy együttesen rekonstruálni tudja a titkot, de K-nál kevesebb személy sohasem legyen erre képes, ahol $K < N$. Ez a megoldás nyilván rekonstruálhatóvá tenné a titkot még akkor is, ha annak legfeljebb $N-K$ darabja megsemmisülne.

Kézenfekvő lenne a binárisan ábrázolt titok valahány, mondjuk T szeletének szétosztása. Ez azonban nem helyes megoldás. Ha ugyanis a szeletek számával egyező a személyek száma ($N = T$), akkor egy rész megsemmisülése is a titok elvesztéséhez vezethet, másrészt az összes személy szükséges a rekonstrukcióhoz. Ha viszont több személynek is adjuk ugyanazt a szeletet ($N > T$), akkor nem választhatunk tetszőleges $K = T$ személyt a rekonstrukcióhoz. Továbbá, jelentős információval rendelkezik T-hez közeli számú személy együttese, hiszen közvetlenül a titkos bináris információ részeit kapták meg.

Egy kicsit javíthatunk a helyzeten a következőképpen. Címezzük meg a lehetséges titkok S halmazának elemeit a $0,1,\ldots,q-1$ számokkal. Válasszunk $N-1$ alkalommal véletlenszerűen egy-egy elemet a $\{0,1,2,\ldots,q-1\}$ halmazból, előállítva $r_1, r_2, \ldots, r_{N-1}$ teljesen független, egyenletes eloszlású valószínűségi változót. Jelölje $s \in S$ az aktuális titkot, s az N személynek szétosztandó N számú információ legyen $r_1, r_2, \ldots, r_{N-1}, r_N$, ahol

$$r_N = s - (r_1 + r_2 + \ldots + r_{N-1}) \mod q.$$

Ekkor ugyan továbbra is az összes személy szükséges a titok rekonstruálásához, viszont N-nél kevesebb részlet ismerete nem nyújt információt a titokra vonatkozóan. Vegyük azt is észre, hogy ekkor a titok-darabok mindegyikének azonos a mérete a titokéval.

Ha ezen véletlenítési ötletet ötvözzük a Reed–Solomon-kódolással, akkor egy kívánt megoldáshoz jutunk. Nevezetesen tekintsünk egy (N,K) paraméterű RS-kódot GF(q) felett, ahol q legyen prímhatvány (ha q eredetileg nem ilyen lenne, bővítsük ki fiktív elemekkel az S halmazt). Az RS-kódok ismert tulajdonsága szerint a kódszó legalább K elemének ismeretében egyértelműen dekódolható, amit a hibajavító kódolás elméletében úgy fogalmazunk, hogy $\leq N-K$ törlés javítására alkalmasak. Ezt a tulajdonságot a titok-szétosztásban a következőképpen kamatoztathatjuk:

Válasszunk $K-1$ elemet véletlenszerűen GF(q)-ból, ezeket jelölje $r_1, r_2, \ldots, r_{K-1}$, továbbá legyen $r_0 = s$. Az

$$\mathbf{r} = (r_0, r_1, \ldots, r_{K-1})$$

vektort tekintsük az RS-kóddal kódolandó üzenetnek, azaz ezt egy $\mathbf{G}$, GF(q) feletti $K \times N$ dimenziós generátormátrixszal leképezzük egy

$$\mathbf{c} = (c_0, c_1, \ldots, c_{N-1})$$

kódszóba a

$$\mathbf{c} = \mathbf{rG}$$

lineáris transzformációval. A $\mathbf{c}$ kódszó elemeit osszuk szét az N személy között. A $\mathbf{G}$ generátormátrixot választhatjuk a következőképpen:

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{N-1} \\ \vdots & & & \ddots & \vdots \\ 1 & \alpha^{K-1} & \alpha^{2(K-1)} & \cdots & \alpha^{(N-1)(K-1)} \end{pmatrix}$$

ahol $\alpha \in \mathrm{GF}(q)$ N-edrendű primitív elem. Bevezetve a

$$D(x) = r_0 + r_1 x + \cdots + r_{K-1} x^{K-1}$$

GF(q) feletti polinomot, a c kódszó elemei a

$$c_i = D(\alpha^i), \qquad i = 0, 1, \ldots, N-1$$

alakban is előállíthatók. Ennek a konstrukciónak van egy érdekes interpretációja:

Egyetlen olyan $K-1$ fokszámú $y = D(x)$ polinom létezik (lehet valós számtest vagy véges test feletti), amely adott $(y_1, x_1), (y_2, x_2), \ldots, (y_K, x_K)$ K számú ponton keresztül fektethető, vagyis amelyre

$$y_i = D(x_i), \qquad 1 \le i \le K.$$

Ha tehát N darab, egy $(K-1)$ fokszámú görbén fekvő pontot választunk, akkor bármely legalább K pontot tartalmazó részhalmazból a görbe (polinom) rekonstruálható, amelynek a nulladfokú tagja a titok.

5.6. Feladatok

5.1. feladat. Egy egyszerű lineáris rejtjelezőt konstruálunk: $y = ax + b \mod r$ lineáris transzformációval rejtjelezünk, ahol $0 < a, b < s$, a és b a kulcs részei, x a nyílt szöveg, y a rejtett szöveg, továbbá r az ábécé mérete.

a) Adja meg a kulcstér méretét, ha $s = 26$!

b) Támadóként a kulcsot szeretnénk megfejteni. Nagyon egyszerű strukturájú a rejtjelező. A forrás jól tömörített, véletlen forrásként modellezhető. Rejtett szövegű támadásban is gondolkodhatunk?

c) Milyen információt kellene birtokolnunk ilyen forrás esetén a sikeres támadáshoz?

d) Mit mondhatunk azon esetben, ha a forrás írott szöveg, s imerjük a karaktergyakoriságot?

5.2. feladat. Az $y = ax + b \mod N$ betűnkénti lineáris rejtjelezés transzformációt tekintjük, ahol N az ábécé mérete.

a) Hány ilyen transzformáció van, ha $N = 30$?

b) x a transzformáció fixpontja, ha $y = x$ teljesül. Legyen $a \neq 1$. Mutassuk meg, hogy ha N prímszám, akkor pontosan egy ilyen fixpont van!

c) Adjunk meg olyan N értéket, amelyre nincs fixpontja a transzformációnak!

5.3. feladat. Tegyük fel, hogy $\mathbf{y} = \mathbf{Ax} + \mathbf{b}$ lineáris transzformációval rejtjelezünk, ahol $\mathbf{A}$ $n \times n$-es bináris mátrix, $\mathbf{x}, \mathbf{y}, \mathbf{b}$ n hosszú bináris vektorok, továbbá $\mathbf{A}$ és $\mathbf{b}$ a kulcs részei, $\mathbf{x}$ a nyílt szöveg, $\mathbf{y}$ a rejtett szöveg. Az algoritmikus támadó célja a kulcselemek meghatározása.

a) Végrehajtható-e a támadás, ha a támadó

   - $y_1, y_2, \ldots$ rejtett szövegeket
   - $(x_1, y_1), (x_2, y_2), \ldots$ nyílt–rejtett szöveg párokat

   tud megszerezni? Adja meg a támadás módját, ha ilyen van, valamint a sikeres támadáshoz szükséges információ mennyiségét!

b) Korlátozhatjuk-e a támadás sikerét azzal, hogy maximáljuk egy kulcs felhasználásának számát (egy kulcsot szeretnénk minél többször használni)?

5.4. feladat. Tekintsük egy szimmetrikus rejtjelező páros gráf reprezentációját, azaz amelyben egy x nyílt üzenetet egy y rejtjeles üzenettel él köt össze, ha valamely k kulcsra $E_k(x) = y$. Igazolja, hogy a tökéletes rejtjelezés páros gráfjában a csomópontpárokat azonos számú él köti össze.

5.5. feladat. One-time-pad rejtjelezést tekintünk:

a) Pénzfeldobás-sorozattal generálunk bináris véletlen folyamot, ahol a pénz egyik felére gyakrabban esik, s a 0 kimenetel valószínűsége $p > 0.5$. Használhatjuk-e ezt kulcsfolyamként egy one-time-pad rejtjelezőben?

b) A rejtjelezést hogyan befolyásolja a kódolandó nyílt szöveg redundanciájának mértéke. (Pl. erősen strukturált szövegeket „nehezebb-e rejteni” ezen a módon?) Formálisan indokoljon!

5.6. feladat. Valaki azt állítja, hogy egy RSA-algoritmus biztonságát nem veszélyezteti, ha az e nyilvános kulcs és $m = p_1 \cdot p_2$ modulus mellett a $\phi(m)$ értékét is nyilvánosságra hozzuk. Igaz ez?

5.7. feladat. Egy játék RSA-algoritmus esetén $p_1 = 23$, $p_2 = 11$ prímeket választottuk. Adja meg a lehető legkisebb kódoló kulcsot, s az ehhez tartozó dekódoló kulcsot, majd kódolja az $x = 5$ üzenetet!

5.8. feladat. Tegyük fel, hogy DES (Data Encryption Standard) rejtjelezést használtunk 64 bites üzenet blokkok rejtjelezésére, amelyek 8 bites karakterekből állnak, s a 8. bit páros paritás. Elvben végrehajtható-e kulcskereséses támadás csak rejtett szövegek megfigyelésére alapozva? Hány rejtjeles blokkot kellene megfigyelni ehhez? (A DES 64 bites nyílt szövegblokkot azonos méretű rejtett szövegblokkba kódol, amelyhez 56 kulcsbitet használ.)

5.9. feladat. Tekintsünk egy RSA-rejtjelzőt $e = 3$ nyilvános kulccsal. Legyen a blokkhossz 128 bájt. Tegyük fel, hogy rövidek az üzeneteink, hosszuk nem nagyobb mint 40 bájt, s a nagyobb helyiértékek felé nullákkal egész blokkokra egészítjük ki azokat. Valaki azt állítja, hogy a rejtett blokkokat lehallgatva fejti az üzeneteket. Milyen tanulságot vonna le?

5.10. feladat. Ha két, RSA-rejtjelezéssel kommunikáló pár között az üzenetek tere, azaz a különböző lehetséges üzenetek száma kicsi halmaz, az támadásra ad lehetőséget. Tegyük fel, hogy a támadó, ismerve a kommunikáló partnerek közti szokásos információcserék halmazát, de nem ismerve az RSA dekódoló kulcsot, támadásra szánja el magát. Adja meg a támadás menetét, s javasoljon — algoritmikus — védekezési módot!

5.11. feladat. RSA-algoritmus számára történő prímválasztásnál egyik követelmény, hogy a két prím bitmérete legyen lehetőleg közel azonos. Például 1024 bites modulus esetén 512 bites prímeket választunk. Ugyanakkor, ha a $p-q$ differencia nem elegendően nagy, akkor faktorizálhatóvá válhat a modulus. Adjon egy faktorizálási algoritmus!
(Segítség: Tekintsük a következő felbontást: $m = pq = (t+s)(t-s) = t^2 - s^2$ alakból, ahol $p = t+s, q = t-s$, látható, hogy $s = (p-q)/2$ viszonylag kicsi volta miatt $t \approx \sqrt{m}$.)

5.12. feladat. Arra, hogy az RSA-algoritmus fejtésének nehézsége a modulus faktorizálásával ekvivalens feladat, csak erős sejtés létezik. Ugyanakkor, ha rögzített, (pl. $e = 2$) kódoló kitevőt használunk, akkor már ugyanez a sejtés igazolható. Igazoljuk tehát, hogy ekkor a megfigyelt rejtett szöveg alapján a nyílt pár megfejtésének bonyolultsága a modulus faktorizációjával ekivivalens feladat.
(Segítség: Tekintsük az $x^2 = c \mod m,\ 0 < c < m,\ m = pq$ egyenletet, ahol p és q prímszámok. Ha van megoldása az egyenletnek, akkor négy megoldása van, s a megoldások $\{b_1, m-b_1, b_2, m-b_2\},\ 0 < b_1, b_2 < n$. p és q ismeretében a négy megoldást az $x^2 = c \mod p, x^2 = c \mod q$ egyenletpár megoldásait felhasználva a kínai maradéktétel segítségével kaphatjuk meg. Ha viszont ezen faktorok nem ismertek, az egyenlet megoldása nehéz.)

5.13. feladat. Szeretnénk megtudni A felhasználó $m = pq$ RSA-modulusa titkos faktorjait. Felelőtlenül jár-e el A, ha „baráti" kérésünkre egy b számnak megmondja egy $\mod m$ szerinti négyzetgyökét.

5.14. feladat. Kriptoanalistaként támadjon egy RSA-rejtjelezőt, amelyről a nyilvános $m = 4003997$ modulus, valamint az $e = 379$ kulcs mellett megtudja a $\phi(m) = 3999996$ értéket is.

a) Számítsa ki a d dekódoló kulcsot!

b) Adja meg az $m = p_1 p_2$ prímfaktorait!

5.15. feladat. Az RSA-algoritmus az $m = (p-1)(q-1)$ modulust alkalmazza a kulcsgenerálásnál. Használhatnánk-e az $m' = \text{lkkt}(p-1, q-1)$ modulust e helyett?

5.16. feladat. Jelszavas védelem során a *salting* azt jelenti, hogy r számú véletlen bittel meghosszabbítják a jelszavakat a jelszóverifikáció során, s aztán alkalmazzák az egyirányú leképezést, majd a leképezés eredményének a tárolt jelszótábla elemmel történő egybevetését. A salt bitfüzér nyíltan kerül tárolásra a jelszótáblában, s a különböző felhasználók saját salt bitekkel rendelkeznek. Felmerül a

kérdés, hogy mi a salt bitek jelentősége, ha azok nyíltan tárolódnak, azaz a jelszótábla „szótáras” támadója is olvashatja azokat?

5.17. feladat. Tegyük fel, hogy véletlenszerűen választunk a jelszótér elemei közül jelszavakat. Legyen m a jelszó hossza, c a karakterábécé mérete, t a verifikációs leképezés iterációinak száma, továbbá r jelölje egy iteráció futási idejét. Legfeljebb 7 hosszú jelszót szereténk használni. A karakterábécé vonatkozásában három lehetőségben gondolkodunk:

- $c = 26$ méretű kisbetűs
- $c = 36$ méretű kisbetűs, alfanumerikus
- $c = 62$ méretű kis- és nagybetűs, alfanumerikus.

Legyen $t = 25$ az iterációk száma, továbbá $r = 4$ mikrosec. Adja meg azon választási lehetőségeket, amely mellett a jelszótér kimerítő végigkeresésén alapuló támadás számítási ideje legalább 100 nap.
Ha a hosszmegkötést nem kellene figyelembe venni, s a jelszavakat a szótár két tetszőleges szavának egymás mellé illesztésével képeznénk, egy 250000 tételes szótárméret elegendő lenne-e?

5.18. feladat. A háromlépéses, előzetes kulcscserét nem igénylő rejtett üzenettovábbító kriptográfiai protokollban a kulcsbitek üzenetbitenkénti $\bmod 2$ hozzáadásával rejtjeleznek a felek. Helyesen cselekszenek-e?

5.19. feladat. Tegyük fel, hogy egy űrjármű (A) leszálláshoz készülődik egy távoli bolygó űrállomásán (B), s ehhez először azonosítania kell magát. A feltételek:

- B ismeri A jelszavát, ezen kívül más közös titkuk nincs.
- A $\bmod 2$ összeadásnál bonyolultabb műveletet nem tud végezni.
- A lesugárzott jeleket egy az űrállomás környéki támadó (C) is lehallgathatja, mivel nem lehet jól koncentrálni a sugárzást. Ugyanakkor a bolygón levő űrállomás képes úgy továbbítani a jeleket, hogy azok a bolygó felszínén nem vehetők. Javasoljon egy protokollt a biztonságos azonosításra!

5.20. feladat. A klaviatúra 26 kisbetűs karakteréből, a 10 számkarakterből, valamint az üreshely (space) karakterből álló 37 betűs ábécéből képezünk titkos, 7 karakterből álló jelszót. 3 személy között kívánjuk szétosztani a titkot úgy, hogy legalább 2 személy jelenléte esetén lehessen csak azt rekonstruálni. Legyen a titok *4qt76ff*. Adjon meg egy megosztási és rekonstrukciós algoritmust, s ossza szét az adott titkot!

5.21. feladat. A véletlen csatornahibák detekciójára szolgáló CRC-kódok nem alkalmasak szándékos bitmanipulációk detektálására. Mutassa ezt meg a következő konkrét példán. Legyen „pay to john 1000$” a továbbítandó nyílt szöveg (ASCII kódot használjon, s vegye figyelembe a szóközöket is). Manipulálja a nevet „jack”-re. A következő szabványos CRC-polinomot használja az „integritásvédelem”: $p(x) = 1 + x^2 + x^{15} + x^{16}$!

5.22. feladat. Tekintsen egy additív kulcsfolyamos rejtjelezést. Vizsgálja meg, hogy ezen rejtjelezés biztosít-e adatintegritás-védelmet!

5.23. feladat. Egy cég informatikai központja szoftverek egy-egy példányát szétosztja a kihelyezett egységei informatikai részlegeinek. Szeretné a szoftverek sértetlenségét biztosítani, s alkalmanként (például hetente) szeretné ellenőrizni azok helyességét. A feladat megoldásához azonban nem kíván titkos kulcshoz kapcsolódó eljárásokat alkalmazni, például azért mert a korrekt kulcsgondozás kényes és költséges feladat, s erre nem kíván felvonulni, erőforrásokat lekötni. Lehetséges-e megoldás ilyen feltételek mellett?

5.24. feladat. Az egyik legfontosabb MAC hash függvény generálás blokk rejtjelező kódok CBC módját használja. Az $m = [M_1, M_2, \ldots, M_r]$ üzenetre k kulcs mellett adott MAC legyen az alábbi:

$$\mathrm{MAC}_k(m) = E_k(E_k(\cdots E_k(M_1) \oplus M_2) \oplus \cdots M_{r-1}) \oplus \cdots M_r),$$

ahol az $E_k : \{0,1\}^n \to \{0,1\}^n$ egy titkos kulcsú rejtjelező transzformáció.
Mutassa meg, hogy a fenti definíció szerinti MAC hashing nem teljesíti az egyirányúság követelményét egy k kulcsot ismerő fél számára!
(Segítség: Mutassa meg, hogy n-bites üzenetblokkok tetszőleges véges sorozatát — a k kulcs ismeretében — kiegészíthetjük úgy egy további blokkal, hogy előre megadott MAC álljon elő!)

5.25. feladat. Egy MAC választott szövegű támadás ellen védett, ha egy támadó rendelkezésére áll $\mathrm{MAC}_k(m_i)$ az általa választott $m_1, m_2, \ldots, m_j$ üzenetekre, mégsem képes ennek alapján kiszámítani egy $\mathrm{MAC}_k(m)$ lenyomatot egy új $m \neq m_i$ üzenetre. Az 5.24. feladatbeli definíció szerinti MAC-t tekintsük. Mutassa meg, hogy az MAC nem védett választott szövegű támadás ellen!
(Segítség: Mutassa meg, hogy két n bites — azaz 1 blokk méretű — üzenetből konstruálható megfelelő $2n$ bites üzenet!)
Mutassa meg továbbá, hogy az MAC nem védett választott szövegű támadás ellen akkor sem, ha úgy kívánjuk megerősíteni, hogy az MAC számítás előtt kiegészítjük az üzenetet egy olyan blokkal, amely az üzenet hosszát tartalmazza! A

támadó képességéről azt teszzük fel, hogy üzeneteire MAC lenyomatot kaphat, de a k kulcsot nem ismeri.

5.26. feladat. Tekintsük az alábbi integritásvédelmi kódolást, ahol a rejtjelezés és MAC kombinációját használjuk:

$$E_k(m\|\mathrm{MAC}_{k'}(m)).$$

Továbbá $E_k(x)$ CBC módú rejtjelezés, $a\|b$ az a és b bináris vektorok egybefűzése. A rejtjelezés, illetve az MAC számítás tekintetében, az egyik funkcióra $[IV,k]$ míg a másikra $[IV',k']$ [inicializáló vektor, kulcs] pár kerül alkalmazásra. Van-e veszélye annak, ha $IV = IV', k = k'$ „egyszerűsítő" választással élünk? Mi a tanulság?

5.27. feladat. Egy véletlen bitfolyam generátor kimenetén ugyan megmaradt az egymás utáni bitek statisztikai függetlensége, de $\frac{1}{2}$-nél nagyobb, $p = 0.75$ valószínűséggel kapunk 0-t a kimeneten. Milyen mértékben korrigálhatjuk a bitfolyam statisztikát azzal, hogy 8 bites szeletekre bontjuk a folyamot és az egyes szeletek 8 bitjéből modulo 2 összeadással egy-egy bitet generálunk?

5.28. feladat. Külföldön dolgozunk, s szeretnénk rejtetten párbeszédet folytatni az otthoni barátunkkal (például számítógépes modemkapcsolat útján), azonban tilos rejtjelezni a határon átlépő üzeneteket. Hitelesítő protokollok használata ugyanakkor nem tiltott (amikor is a nyílt szöveg nyílt marad).

a) Van megoldás?

b) Ha megtalálta, adjon egy példát egy rövid párbeszéd védelmére!

5.29. feladat. Szimmetrikus kulcsú rejtjelező transzformáción alapuló egyirányú transzformáció, amelyet PIN kódolására használnak, tipikusan a következő általános alakú:

$$AP = f(KP, PIN, ID)$$

AP: nyilvános hitelesítő paraméter (authentication parameter),
KP: titkos kulcs paraméter (key parameter),
PIN: titkos személy-azonosítószám (personal identification number),
ID: nyilvános azonosító (identification data),
f egy egyirányú függvény. A legismertebb ilyen függvény az $E_k(\cdot)$ DES transzformáción alapul:

$$AP = E_{KP\oplus PIN}(ID)$$

A verifikációs táblázat tartalma a következő:

$$\begin{array}{cc} ID_1 & AP_1 \\ ID_2 & AP_2 \\ \vdots & \vdots \\ ID_n & AP_n \end{array}$$

A táblázat olvasható, mivel f egyirányú leképezés, és AP ismerete egy adott ID-hez nem segít a PIN kiszámításában.

Mutassuk meg, hogy ugyanakkor a következő leképezés

$$AP = E_{KP \oplus PIN}(ID \oplus PIN)$$

nem egyirányú abban az értelemben, hogy nem „nehéz feladat” megadni olyan KP', PIN' párt amelyre

$$f(KP', PIN', ID) = f(KP, PIN, ID).$$

Irodalomjegyzék

[1] *CDMA Digital Common Air Interface Standard, Revision 1.0 released Oct.1., 1990*.

[2] *All about the Compact Disc System, Compact Disc Digital Audio*. Sony, 1981.

[3] *Specification of the D2-MAC/PACKET System EBU/SPB 352/B.* 1985.

[4] Ash, R.B. *Information Theory*. Interscience Publishers, 1965.

[5] Bahl, L.R., Cocke, J., Jelinek, F., Raviv, J. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Transactions on Information Theory*, IT-20:248–287, 1974.

[6] Berlekamp, E.R. *Algebraic Coding Theory*. McGraw Hill, 1968.

[7] Berlekamp, E.R. The technology of error-correcting codes. *Proceedings of the IEEE*, 68, May, 1980.

[8] Berlekamp, E.R., Peile, R.E., Pope, S.P. The application of error control to communications. *IEEE Communications Magazine*, 25, Apr, 1987.

[9] Berrou, C., Glavieux, A., Thitimajshima, P. Near Shannon limit error-correcting coding and decoding: Turbo-codes(1). *Proceedings of the IEEE International Conference on Communication*, pages 1064–1070, 1993.

[10] Blahut, R.E. *Theory and Practice of Error Control Codes*. Addison-Wesley, 1983.

[11] Csibi S. (szerk.) *Információ közlése és feldolgozása*. Tankönyvkiadó, Budapest, 1986.

[12] Csiszár I., Fritz J. *Információelmélet*. ELTE TTK jegyzet, Budapest, 1986.

[13] Csiszár I., Körner J. *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Akadémiai Kiadó, Budapest, 1981.

[14] Diffie, W. Hellman, M.E. Privacy and authentication: An introduction to cryptography. *Proceedings of the IEEE*, 67:397–427, Mar, 1979.

[15] Diffie, W. Hellman, M.E. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, Nov, 1976.

[16] Doi, T. *Error Correction for Digital Audio Recordings, No. 1991 AES 73. Convention*. Eindhoven, 1983.

[17] Ferenczy P. *Video- és hangrendszerek*. Műszaki Könyvkiadó, Budapest, 1986.

[18] Gallager, R.G. *Information Theory and Reliable Communication*. Wiley, 1968.

[19] Géher K. (szerk.) *Híradástechnika*. Műszaki Könyvkiadó, Budapest, 1993.

[20] Gibson, J.D., Berger, T., Lookabaugh, T., Lindbergh, D., Baker, R.L. *Digital Compression for Multimedia (Principles and Standards)*. Morgan Kaufmann Publishers, San Francisco, 1998.

[21] Gordos G., Takács Gy. *Digitális beszédfeldolgozás*. Műszaki Könyvkiadó, Budapest, 1983.

[22] Györfi L., Vajda I. *A hibajavító kódolás és a nyilvános kulcsú titkosítás elemei*. Műegyetemi Kiadó, Budapest, 1990.

[23] Hanzo, L., Steele, R. *Mobile Radio Communications*. Wiley, 1999.

[24] Ishida, Y., Ishida, M., Nakagawa, K., Osuga, Y., Yanabe, J. *On the development of a car use rotary-head digital audio tape recorder, No. 2318 AES 80. Convention*. Montreux, 1986.

[25] Linder T., Lugosi G. *Bevezetés az információelméletbe*. Műegyetemi Kiadó, Budapest, 1993.

[26] MacWilliams, F.J., Sloane, N.J.A. *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[27] Massey, J.L. *Applied Digital Information Theory*. Course Notes, ETH Zürich, 1984.

[28] Massey, J.L. Introduction to contemporary cryptology. *Proceedings of the IEEE*, 76:533–548, May, 1988.

[29] Massey, P.C., D.J. Costello. Jr. New Developments in Asymmetric Turbo Codes. *Proceedings of the 2nd International Symposium on Turbo Codes and Related Topics*, pages 93–100, 2000.

[30] McEliece, R.J. *The Theory of Information and Coding*. Addison-Wesley, 1977.

[31] Moore, J.H. Protocol failures in cryptosystems. *Proceedings of the IEEE*, 76:594–602, 1988.

[32] Németh J. *Adatvédelem számítógépes és hírközlő rendszerekben*. Számalk, Budapest, 1984.

[33] Nemetz T., Vajda I. *Bevezetés az algoritmikus adatvédelembe*. Akadémiai Kiadó, Budapest, 1991.

[34] Niven, I., Zuckerman, H.S. *Bevezetés a számelméletbe*. Műszaki Könyvkiadó, Budapest, 1978.

[35] Odaka, K., Furuya, T., Taki, A. *LSlos for Digital Signal Processing to be used in CD Players, No. 1860 AES 71. Convention*. Montreux, 1982.

[36] Peterson, W.W. *Error Correcting Codes*. MIT Press – Wiley, Cambridge, 1961.

[37] Pursley, M.B. Performance evaluation for phase-coded spread-spectrum multiple-access communications – Part I-II. *IEEE Transactions on Communications*, 25, Aug, 1977.

[38] Rabin, M.O. Probabilistic algorithm for preliminary testing. *Journal of Number Theory*, 12:128–138, 1980.

[39] Rivest, R.L., Shamir, A., Adleman, L. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21:120–126, Feb, 1978.

[40] Rueppel, R.A. *Analysis and Design of Stream Ciphers*. Springer-Verlag Berlin, Heidelberg, 1986.

[41] Sarwate, D.V., Pursley, M.B. Crosscorrelation properties of pseudorandom and related sequences. *Proceedings of the IEEE*, 68, May, 1978.

[42] Sayood, K. *Introduction to Data Compression*. Morgan Kaufmann Publishers, San Francisco, 1996.

[43] Shannon, C.E. A mathematical theory of communication. *Bell System Technical Journal*, 1948.

[44] Shannon, C.E. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, Oct, 1949.

[45] Vajda I. *Hibajavító kódolás és műszaki alkalmazásai*. BME Mérnöki Továbbképző Intézet, Budapest, 1982.

[46] Viterbi, A.J. *CDMA Principles of Spread Spectrum Communication*. Addison–Wesley, 1995.

Tárgymutató