Teoria Codurilor corectoare de erori

Book · February 2001

CITATION
1

READS
1,288

1 author:

Adrian Constantin Atanasiu
University of Bucharest
66 PUBLICATIONS **322** CITATIONS

SEE PROFILE

Coduri detectoare şi corectoare de erori

Adrian Atanasiu

Editura Universităţii BUCUREŞTI

Prefaţă

Vă uitaţi la televizor – care transmite imagini prin satelit? Vorbiţi la telefon (celular)? Folosiţi Internetul? Ascultaţi muzică de pe DVD-uri?

Aţi admirat fotografii extrem de reuşite ale planetelor Jupiter sau Saturn. Cum au fost obţinute ele

Aţi auzit de alfabetul Morse; v-aţi întrebat de ce pentru ajutor apelul este S.O.S. ?

Sunt multe astfel de domenii intrate în cotidian, care folosesc coduri. De fapt, oricare din exemplele de mai sus se referă la un transfer de informaţie. Ceea ce se solicită (în mod neexplicit dar esenţial) este ca informaţia cerută să ajungă nemodificată la beneficiarul transmisiei. Indiferent prin ce mediu (numit *canal*) se face transmisia (poate fi laser, cablu, unde etc) mesajul expeditorului trebuie să fie identic cu cel al destinatarului. Chiar dacă sunt "zgomote" care pot altera acest conţinut.

Tocmai din cauza acestor posibile perturbaţii de canal, ceea ce se transmite (numit *mesaj de informaţie*) se completează cu elemente redondante (numite *caractere de control*), care nu aduc informaţie suplimentară dar o confirmă pe cea existentă. În acest fel, o modificare posibilă a mesajului (o zgârietură pe DVD, o interferenţă, un ecou de canal) poate fi eliminată la recepţie de către decodor.

Să considerăm de exemplu un mesaj "*abac*" format din patru caractere. Îl vom codifica prin scrierea fiecărui caracter de trei ori. Deci vom transmite "*aaabbbaaaccc*". Dacă destinatarul primeşte "*caabbwaaaccb*", la decodificare va separa textul în grupe de câte trei simboluri "*caa bbw aaa ccb*" şi va păstra din fiecare grup caracterul care apare de cele mai multe ori. Va obţine "*abac*". Este o idee de decodificare folosită cu precădere de aproape toate sistemele, numită *decodificarea cea mai probabilă*.

Evident, pot apare perturbări grupate, care să altereze mai multe caractere consecutive. Cum se operează atunci? Pur şi simplu se foloseşte altă codificare (sau supracodificare, care este o combinaţie de mai multe codificări). S-a ajuns în acest moment să existe clase de coduri (turbo-codurile) capabile să corecteze aproape 75% din simbolurile alterate ale unui mesaj!

Pare fascinant. De fapt, aşa şi este!

Teoria codurilor este un domeniu dezvoltat relativ recent ca aparat matematic (are cam 70 ani vechime). Dacă se face abstracţie de unele confuzii de termeni cu criptografia (care utilizează frecvent termenul de "cod" pentru unele sisteme clasice de criptare) sau de ideea că de fapt şi scrierea sau vorbirea constituie sisteme de codificare ale ideilor, primul cod care merită acest nume în lumina obiectivelor menţionate anterior, este codul Morse apărut în 1840 – de fapt un alfabet care să permită transmiterea mesajelor prin telegraf.

Explozia informaţională din a doua jumătate a secolului XX a condus la apariţia acestei discipline, ale cărei baze teoretice sunt puse în anii 50, are o dezvoltare fulgerătoare până la

sfârşitul anilor 60, stagnează aproape două decenii, după care renaşte din 1985 şi cunoaşte astăzi din nou o creştere explozivă, explicabilă prin necesitatea transmiterii unui volum de informaţie nemaiîntâlnit la o viteză aproape nebănuită până acum 10-15 ani.

Cea mai mare parte a noţiunilor prezentate aici constituie suportul unui curs ţinut o vreme la Facultatea de Matematică a Universităţii Bucureşti (întâi profesorul Silviu Guiaşu în anii 67-75, apoi reluat de autor după 1990). Lucru explicabil, deoarece bazele teoriei codurilor sunt pur matematice (algebră liniară, teoria numerelor, corpuri finite, statistică). Aceasta nu face însă domeniul mai puţin atrăgător, dimpotrivă. Sugerează că pot fi gândite şi alte abordări care să genereze sisteme de codificare noi, cu performanţe superioare. Deja ele apar din domeniul cuantic sau genetic. Ideea în sinte constituie o provocare, un apel al societăţii, al vieţii practice, adresat ştiinţelor fundamentale. Putem să îi răspundem? Aceasta este întrebarea.

Autorul este conştient că nu a putut acoperi şi explica tot ce conţine domeniul în sine. De asemenea, lucrarea poate cuprinde unele erori (nu numai de redactare). Este riscul pe care şi-l asumă orice autor. De aceea, rugămintea către cititor(ul interesat) este de a nu ezita stabilirea unei colaborări. Suntem deschişi şi receptivi la orice mesaj. În ciuda riscurilor unor posibile perturbaţii de canal.

Capitolul 1

Codificare şi decodificare

1.1 Codificare

Definiţia 1.1 *Fiind date două mulţimi finite şi nevide A (alfabetul sursă) şi B (alfabetul cod), o* **codificare** *este o aplicaţie injectivă $\phi : A \longrightarrow B^*$.*

Elementele mulţimii $\phi(A) \subseteq B^*$ se numesc *cuvinte-cod*, iar mulţimea $\phi(A)$ se numeşte *cod*.

Dacă B are numai două simboluri, codificarea ϕ se numeşte *binară*.

Exemplul 1.1 *Fie $A = \{0, 1, \ldots, 9\}$ şi $B = \{0, 1\}$. Printre secvenţele binare de lungime 5, numărul celor care au doi de 1 este $C_5^2 = 10$. Ele pot fi folosite pentru a codifica cifrele din scrierea zecimală (Tabelul 8.1).*

Tabelul 1.1: Codul "doi-din-cinci"

Simbol zecimal	**Cuvânt cod**
1	11000
2	10100
3	01100
4	10010
5	01010
6	00110
7	10001
8	01001
9	00101
0	00011

Mesajul "173" are codul 110001000101100. *De remarcat că între cuvinte - cod nu se lasă nici un spaţiu, deoarece caracterul "spaţiu" poate fi el însuşi un simbol din alfabetul cod. Astfel de exemplu, codul Morse are alfabetul $B = \{., -, spa'tiu\}$.*

Decodificarea se face foarte simplu: se împarte mesajul codificat în grupe de câte cinci caractere şi se vede cifra din Tabelul 8.1 corespunzătoare grupei respective.

Aranjarea cuvintelor cod a fost făcută pentru a realiza şi o decodificare pe baza unei formule. Astfel, dacă $a_0a_1a_2a_3a_4$ este un cuvânt - cod, el corespunde cifrei k rezultată din de algoritmul:

begin
$x := a_1 + 2a_2 + 4a_3 + 7a_4$;
if $x = 11$ **then** $k := 0$ **else** $k := x$;
end.

Definiţia 1.2 *Pentru o codificare $\phi : A \to B^*$, se numeşte "codificare a mesajelor (textului) sursă" aplicaţia $\phi^* : A^* \longrightarrow B^*$ definită recursiv prin:*

- $\phi^*(\epsilon) = \epsilon$ *(ϵ este cuvântul vid);*
- $\phi^*(a\alpha) = \phi(a)\phi^*(\alpha), \quad \forall\, a \in A, \alpha \in A^*.$

Definiţia 1.3 *Codificarea ϕ este "unic decodabilă" dacă ϕ^* este injectivă.*

Codificarea dată în Exemplul 2.1 este – după cum s-a observat – unic decodabilă. Acest lucru nu este totdeauna posibil. Dacă luăm de exemplu codificarea

$$\phi(a) = 10, \quad \phi(b) = 101, \quad \phi(c) = 110,$$

ea nu este unic decodabilă; astfel $\phi^*(ac) = \phi^*(ba) = 10110$.

Definiţia 1.4

1. *O codificare $\phi : A \longrightarrow B^*$ în care toate cuvintele cod au lungi-mea n se numeşte "codificare-bloc de lungime n", iar $\phi(A)$ este un "cod-bloc de lungime n".*
2. *O codificare $\phi : A \longrightarrow B^*$ se numeşte "instantanee" dacă $\phi(A)$ are proprietatea prefixului (dacă $\alpha, \alpha\beta \in \phi(B)$ atunci $\beta = \epsilon$).*

   Mulţimea $\phi(A)$ este numită "cod instantaneu".

Codul definit în Exemplul 2.1 este un cod - bloc de lungime 5.

Codurile bloc sunt eficiente în situaţia când simbolurile sursă au frecvenţe egale de apariţie; în caz contrar, ele devin greoaie şi sunt preferabile codurile instantanee cu lungimi variabile ale cuvintelor cod.

Exemplul 1.2 *Codul Morse, dat în Tabelul 8.2 este un cod instantaneu cu alfabetul cod $B = \{., -, \}$. Deoarece spaţiul este folosit numai la sfârşitul fiecărui cuvânt - cod, procedura de decodificare este simplă: orice cuvânt - cod se află între două spaţii, de la începutul mesajului până la primul spaţiu, sau de la ultimul spaţiu până la sfârşit. Motivul pentru care nu se foloseşte un cod - bloc este simplu: frecvenţele literelor într-o limbă diferă foarte mult.*

Tabelul 1.2: Codul Morse

A . -	F . . - .	K - . -	P . - - .	U . . -
B - . . .	G - - .	L . - . .	Q - - . -	V . . . -
C - . - .	H	M - -	R . - .	W . - -
D - . .	I . .	N - .	S . . .	X - . . -
E .	J . - - -	O - - -	T -	Y - . - -
				Z - - . .

Exemplul 1.3 *Un alt exemplu de cod - bloc este codul hexazecimal:*

0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

Exemplul 1.4 *Să presupunem că vrem să construim un cod binar pentru alfabetul* $\{0,1,2,3\}$ *şi observăm că* 0 *apare în mesajele sursă mai des decât orice alt simbol. Atunci următoarea schemă de codificare pare rezonabilă:*

$$\phi(0) = 0, \qquad \phi(1) = 01, \qquad \phi(2) = 011, \quad \phi(3) = 111.$$

Aici $\phi(0)$ *are lungime minimă, iar algoritmul de decodificare este foarte simplu: se aplică recursiv regula:*

"Fie sufixul 01^k*; el este codificarea numărului* $k \ (mod\ 3)$ $\underbrace{33\ldots3}_{k\ div\ 3}$*"*

Totuşi această codificare nu este instantanee. Într-adevăr, dacă se primeşte un mesaj lung de forma

$$0111111111111111\ldots$$

nu vom şti dacă primul simbol sursă este $0, 1$ *sau* 2 *până nu se termină transmiterea întregului mesaj.*

1.2 Exemple de coduri - bloc importante

Deşi simple, codurile binare sunt de obicei lungi şi deci greu de manipulat. Adesea este mai convenabil să grupăm simbolurile binare formând alfabete mai complexe.

Astfel, grupuri de câte trei simboluri binare conduc la codurile octale. Reprezentarea în octal se indică de obicei prin indicele 8 aşezat la sfârşit. De exemplu,

$$\phi(01) = (01)_8 = 000001$$

În mod similar, prin gruparea a câte patru simboluri binare se obţine codul hexazecimal (Exemplul 2.3).

Un cod foarte important folosit în reprezentarea standard a simbolurilor alfabetice şi numerice este codul *ASCII (***A***merican* **S***tandard* **C***ode for* **I***nformation* **I***nterchange)*[1].

[1] Acest cod a fost prezentat pe larg în cursul de Arhitectura sistemelor de Calcul.

Un ultim cod, folosit de toate editurile de pe glob este *Internaţional Standard Book Number* (ISBN). El este un cod - bloc de lungime 10 (lungimea cuvintelor - cod creşte prin folosirea simbolului e diverse poziţii, dar acest caracter este ignorat la prelucrarea automată). Alfabetul cod este $B = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, X\}$, ($X$ pentru numărul 10).

De exemplu, cartea *S. Lin, P. Costello - Teoria Codurilor* are codul

$$ISBN\ 0 - 13 - 283796 - X$$

Primul număr (0) reprezintă ţara (SUA), 13 reprezintă editura (*Prentice-Hall*), iar următoarele şase cifre sunt asignate de editură ca număr de identificare al cărţii. Ultimul simbol este de control (similar bitului de paritate) şi definit astfel:

Pentru codul ISBN $a_1a_2 \ldots a_{10}$, $\sum_{i=1}^{10} ia_{11-i} = 0\ (mod\ 11)$.

Astfel,în $ISBN$-ul de sus,

$10 \cdot 0 + 9 \cdot 1 + 8 \cdot 3 + 7 \cdot 2 + 6 \cdot 8 + 5 \cdot 3 + 4 \cdot 7 + 3 \cdot 9 + 2 \cdot 6 + 1 \cdot 10 = 187 \equiv 0\ (mod\ 11)$

Unele publicaţii au codul de identificare de trei cifre (de exemplu *Wiley-Inter-Science* are 471); în acest caz numărul pentru fiecare publicaţie are numai cinci simboluri.

Pentru România, codul de ţară este 973.

1.3 Construcţia codurilor instantanee

În acest paragraf ne punem problema construirii unui cod binar instantaneu peste alfabetul sursă $A = \{a_1, \ldots, a_n\}$, $(n \geq 1)$. Deci $B = \{0, 1\}$.

Iniţial se specifică lungimile $d_1, d_2, \ldots, d_n$ ale cuvintelor cod. Fără a micşora generalitatea, putem presupune

$$1 \leq d_1 \leq d_2 \leq \ldots \leq d_n$$

La primul pas se alege un cuvânt - cod binar arbitrar $\phi(a_1)$ de lungime d_1.

În continuare se alege un cuvânt - cod arbitrar $\phi(a_2)$ din mulţimea secvenţelor binare de lungime d_2 care nu au pe $\phi(a_1)$ ca prefix. Aceasta este totdeauna posibil pentru că numărul tuturor secvenţelor binare de lungime d_2 este 2^{d_2}; dintre acestea, numărul celor care au prefixul $\phi(a_1)$ este $2^{d_2-d_1}$.

Cum $2^{d_2} \geq 2^{d_2-d_1} + 1$, există cel puţin o alegere posibilă pentru $\phi(a_2)$ de lungime d_2.

Va trebui să selectăm în continuare un cuvânt de lungime d_3 care nu are ca prefix $\phi(a_1)$ sau $\phi(a_2)$. Deci, din cele 2^{d_3} secvenţe binare posibile trebuiesc eliminate cele $2^{d_3-d_1}$ secvenţe cu prefixul $\phi(a_1)$ şi $2^{d_3-d_2}$ secvenţe cu prefixul $\phi(a_2)$. Aceasta este posibil dacă şi numai dacă

$$2^{d_3} \geq 2^{d_3-d_2} + 2^{d_3-d_1} + 1$$

Împărţind această inegalitate cu 2^{d_3}, se obţine

$$1 \geq 2^{-d_1} + 2^{-d_2} + 2^{-d_3}.$$

În mod analog se poate arăta în general inegalitatea

$$1 \geq 2^{-d_1} + 2^{-d_2} + \ldots + 2^{-d_n}$$

Faptul că acest cod este instantaneu rezultă imediat din construcţie.

Teorema 1.1 *(Kraft) Fiind dat un alfabet sursă A de n simboluri şi un alfabet cod B de k simboluri, se poate construi un cod instantaneu cu lungimile cuvintelor - cod $d_1, d_2, \ldots, d_n$ dacă şi numai dacă este verificată inegalitatea:*

$$k^{-d_1} + k^{-d_2} + \ldots + k^{-d_n} \leq 1.$$

Demonstraţie: "$\Longleftarrow$": Fie $A = \{a_1, a_2, \ldots, a_n\}$ şi putem presupune relaţia $d_1 \leq d_2 \leq \ldots \leq d_n$. Construim prin inducţie codificarea instantanee ϕ, astfel:

- Se alege arbitrar $\phi(a_1) \in B^*,\ |\phi(a_1)| = d_1$ (se notează cu $|\alpha|$ lungimea secvenţei α).
- Presupunem că au fost alese $\phi(a_1), \phi(a_2), \ldots, \phi(a_{s-1})\ \ (s > 1)$. Atunci se va alege un cuvânt arbitrar $\phi(a_s) \in B^*$ care nu are ca prefix nici unul din cuvintele selectate anterior. Aceasta este posibil deoarece numărul cuvintelor cu prefixul $\phi(a_i)$ este $k^{d_s - d_i}$ $(1 \leq i \leq s-1)$; deci alegerea poate fi făcută din

$$k^{d_s} - \sum_{i=1}^{s-1} k^{d_s - d_i} \text{ elemente.}$$

Din inegalitatea lui Kraft avem

$$1 - \sum_{i=1}^{s-1} k^{-d_i} \geq k^{-d_s}$$

care, prin multiplicare cu k^{d_s} conduce la

$$k^{d_s} - \sum_{i=1}^{s-1} k^{d_s - d_i} \geq 1,$$

deci există cel puţin un element de lungime d_s care poate fi ales drept $\phi(a_s)$.

Afirmaţia reciprocă se demonstrează similar (pentru $k = 2$ ea a fost dată anterior).q.e.d.

Teorema 1.2 *(McMillan) Orice codificare unic decodabilă satisface inegalitatea lui Kraft.*

Demonstraţie: Fie ϕ o codificare unic decodabilă. Notăm cu d_i lungimea cuvântului cod $\phi(a_i),\ \ (1 \leq i \leq n)$. Se observă că $\forall\ j,\ (j \geq 1)$ se pot forma k^j cuvinte de lungime j peste alfabetul - cod B cu $|B| = k$. Din proprietatea de unic decodabilitate, numărul mesajelor sursă $\alpha = a_{i_1} a_{i_2} \ldots a_{i_r}$ cu $\phi(\alpha) = j$ nu depăşeşte k^j. Lungimea codului pentru α este $d_{i_1} + d_{i_2} + \ldots + d_{i_r}$; deci numărul tuturor sumelor de forma

$$d_{i_1} + d_{i_2} + \ldots + d_{i_r} = j$$

este cel mult k^j.

Rămâne de demonstrat că numărul $c = \sum_{i=1}^{n} k^{-d_i}$ este cel mult 1. Pentru aceasta, vom arăta că $\forall\ r \geq 1, \quad \dfrac{c^r}{r}$ este mărginit.

Să calculăm puterile lui c:

$$c^2 = \left(\sum_{i=1}^{n} k^{-d_i}\right)\left(\sum_{j=1}^{n} k^{-d_j}\right) = \sum_{i,j=1}^{n} k^{-(d_i+d_j)}$$

şi în general,

$$c^r = \sum_{i_1,i_2,\ldots,i_r=1}^{n} k^{-(d_{i_1}+d_{i_2}+\ldots+d_{i_r})}$$

Această sumă se poate re-ordona grupând toţi termenii de forma k^{-j} unde j satisface egalitatea anterioară. Cel mai mare j posibil este $j = d + d + \ldots + d = rd$, unde $d = max\{d_1, d_2, \ldots, d_n\}$.

Numărul tuturor termenilor de forma k^{-j} din sumă este cel mult k^j. Deci,

$$c^r \leq \sum_{j=1}^{rd} k^j k^{-j} = \sum_{j=1}^{rd} 1 = rd.$$

Deci, $\frac{c^r}{r} \leq d$, de unde va rezulta $c \leq 1$ (pentru $c > 1$ şirul $a_r = \frac{c^r}{r} \to \infty$, deci nu este mărginit). q.e.d.

Corolarul 1.1 *Pentru orice cod unic decodabil există un cod instantaneu care are toate cuvintele - cod de lungimi egale.*

Demonstraţie: Rezultă din demonstraţia teoremei precedente.

Exemplul 1.5 *Să considerăm alfabetul sursă $A = \{a, b, c\}$ şi alfabetul cod $B = \{0, 1\}$; deci $n = |A| = 3$, $k = |B| = 2$. Vrem să construim o codificare instantanee $\phi : A \longrightarrow B$ care are toate cuvintele - cod de lungime d.*

Inegalitatea Kraft va da $2^{-d} + 2^{-d} + 2^{-d} \leq 1$, deci $2^{-d} \leq \frac{1}{3}$. Cel mai mic d care o verifică este $d = 2$. Deci orice mulţime de trei secvenţe binare de lungime 2 va putea fi folosită drept cod. Sunt 4 astfel de mulţimi:

$$\{00, 01, 10\}, \quad \{00, 01, 11\}, \quad \{00, 10, 11\}, \quad \{01, 10, 11\}$$

1.4 Coduri Huffman

Am menţionat anterior faptul că dacă frecvenţa simbolurilor sursă variază, atunci codurile instantanee sunt preferabile codurilor - bloc, deoarece simbolurile care apar mai frecvent vor fi codificate prin cuvinte - cod mai scurte.

Ne punem problema aflării unor codificări cât mai eficiente, în ipoteza că frecvenţele simbolurilor sursă sunt cunoscute exact (de exemplu probabilitatea distribuţiei simbolurilor sursă în mesaje).

Definiţia 1.5 *O sursă de informaţie este o pereche $S = (A, P)$ unde*

- *$A = \{a_1, a_2, \ldots, a_n\}$ este alfabetul sursă(mulţime ordonată);*

- $P = \{P(a_1), P(a_2), \ldots, P(a_n)\}$ *este mulţimea ordonată a probabilităt ilor elementelor lui* A*, deci*
  - $0 \leq P(a_i) \leq 1, (1 \leq i \leq n)$;
  - $\sum_{i=1}^{n} P(a_i) = 1$.

Fie ϕ o codificare a unei surse de informaţie. Dacă se notează cu $d_i = |\phi(a_i)|$, se poate defini *lungimea medie* L a cuvintelor cod prin

$$L(\phi) = \sum_{i=1}^{n} d_i P(a_i).$$

O codificare este eficientă dacă lungimea medie a cuvintelor - cod este cât mai mică.

Definiţia 1.6 *Fiind dată o sursă de informaţie* S *şi un alfabet cod, un cod Huffman este un cod instantaneu cu lungimea medie minimă.*

Lungimea medie minimă a unui cod Huffmann se notează cu $L_{min}(S)$.

Exemplul 1.6 *Să se determine un cod Huffman binar pentru alfabetul sursă* $A = \{a, b, c, d, e, f\}$ *ştiind că* $'a'$ *apare de două ori mai des decât* $'e'$ *şi* $'e'$ *de două ori mai des decât orice consoană.*

Deci, vom avea sursa de informaţie

Simbol	a	b	c	d	e	f
Probabilitate	$0,4$	$0,1$	$0,1$	$0,1$	$0,2$	$0,1$

Putem asigna deci un cuvânt - cod de lungime 1 *lui* $'a'$ *şi unul de lungime doi lui* $'e'$*. Atunci lungimile cuvintelor - cod rămase sunt egale cu* 4*, iar inegalitatea lui Kraft este saturată:*

$$\frac{1}{2} + \frac{1}{2^2} + \frac{4}{2^4} = 1.$$

Un astfel de cod se poate construi efectiv:

$$\begin{array}{lll} \phi(a) = 0 & \phi(c) = 1101 & \phi(e) = 10 \\ \phi(b) = 1100 & \phi(d) = 1110 & \phi(f) = 1111 \end{array}$$

Lungimea sa medie este

$$L = 0,4 + 2 \times 0,2 + 4 \times 4 \times 0,1 = 2,4.$$

Deci, pentru acest exemplu, $L_{min}(S) \leq 2,4$.

1.4.1 Construcţia codurilor Huffman binare

O sursă cu două simboluri are evident un cod Huffman de cuvinte - cod $\{0,1\}$ (şi deci $L_{min}(S) = 1$).

O sursă cu trei simboluri $\{a_1, a_2, a_3\}$ în care a_1 are probabilitate maximă, poate fi redusă la cazul a două simboluri $\{a_1, a_{2,3}\}$ unde $P(a_{2,3}) = P(a_2) + P(a_3)$. Vom construi o codificare Huffman pentru sursa redusă

$$\phi(a_1) = 0, \qquad \phi(a_{2,3}) = 1.$$

după care "spargem" cuvântul cod 1 în două cuvinte: 10 şi 11; în acest fel se obţine o codificare Huffman pentru sursa originală:

a_1	a_2	a_3
0	10	11

În general, fie S o sursă de informaţie cu simbolurile $\{a_1, a_2, \ldots, a_n\}$ ordonate descrescător după probabilităţi, adică:

$$P(a_1) \geq P(a_2) \geq \ldots \geq P(a_n).$$

Construim o sursă redusă S^* cu simbolurile $\{a_1, \ldots, a_{n-2}, a_{n-1,n}\}$ unde $a_{n-1,n}$ este un simbol nou, cu probabilitatea $P(a_{n-1,n}) = P(a_{n-1}) + P(a_n)$.

Dacă nu se poate construi un cod Huffman pentru S^*, se reia procedeul pentru această sursă (reordonând eventual simbolurile după probabilitate); în final se va ajunge la o sursă (pentru două simboluri problema a fost rezolvată) în care se poate construi codul Huffman.

Dacă se poate găsi o codificare Huffman ϕ^* pentru sursa redusă S^*, atunci codul din Tabelul 1.3 este un cod Huffman pentru S (vom demonstra această afirmaţie) .

Tabelul 1.3:

a_1	a_2	$\ldots$	a_{n-2}	a_{n-1}	a_n
$\phi^*(a_1)$	$\phi^*(a_2)$	$\ldots$	$\phi^*(a_{n-2})$	$\phi^*(a_{n-1,n})0$	$\phi^*(a_{n-1,n})1$

Lema 1.1

$$L(\phi) = L(\phi^*) + P(a_{n-1}) + P(a_n).$$

Demonstraţie: Fie $d_1, d_2, \ldots, d_{n-2}, d^*$ lungimile cuvintelor cod corespunzătoare lui ϕ^*. Atunci lungimile cuvintelor cod pentru ϕ sunt $d_1, d_2, \ldots, d_{n-2}, d^*+1, d^*+1$.
Efectuând calculele, se obţine:

$$\begin{aligned} L(\phi) &= \sum_{i=1}^{n-2} d_i P(a_i) + (d^*+1)P(a_{n-1}) + (d^*+1)P(a_n) = \\ &= \sum_{i=1}^{n-2} d_i P(a_i) + d^*[P(a_{n-1}) + P(a_n)] + P(a_{n-1}) + P(a_n) = \\ &= L(\phi^*) + P(a_{n-1}) + P(a_n). \end{aligned}$$

q.e.d.

Teorema 1.3 *Fie ϕ^* o codificare Huffman pentru o sursă de informaţie redusă S^*. Atunci codificarea ϕ definită de Tabelul 1.3 determină un cod Huffman pentru sursa de informaţie S.*

Demonstraţie: Fie $a_1, a_2, \ldots, a_n$ simbolurile sursă, ordonate descrescător după probabilitate. Deoarece teorema este evidentă pentru $P(a_n) = 0$, vom considera doar cazul $P(a_n) > 0$.

Demonstraţia constă din trei paşi:

- *S admite o codificare Huffman ϕ_0 cu lungimile cuvintelor - cod ordonate:*

$$d_1 \leq d_2 \leq \ldots \leq d_n \qquad (d_i = |\phi_0(a_i)|, \quad 1 \leq i \leq n).$$

Pentru a demonstra aceasta, plecăm de la o codificare Huffman arbitrară ϕ pentru S. Dacă există un simbol a_i astfel ca $d_i > d_{i+1}$, notăm cu ϕ' codificarea obţinută din ϕ prin permutarea cuvintelor cod corespunzătoare lui a_i şi a_{i+1}. Codul obţinut prin codificarea ϕ' este evident un cod instantaneu, iar diferenţa dintre lungimile medii $L = L_{min}$ (al lui ϕ) şi L' (al lui ϕ') este:

$$L_{min} - L' = [d_i P(a_i) + d_{i+1} P(a_{i+1})] - [d_{i+1} P(a_i) + d_i P(a_{i+1})] =$$
$$= (d_i - d_{i+1})[P(a_i) - P(a_{i+1})].$$

Această expresie este produsul dintre două numere pozitive, deci $L_{min} \geq L'$, iar din proprietatea de minimalitate rezultă $L_{min} = L'$. Cu alte cuvinte, ϕ' este tot o codificare Huffman. Procedeul continuă până se obţine codificarea ϕ_0 cerută.

- *S admite o codificare Huffman ϕ_1 în care ultimele cuvinte cod, $\phi_1(a_{n-1})$ şi $\phi_1(a_n)$ diferă doar prin ultimul simbol.*

Fie ϕ_0 codificarea Huffman anterioară şi $\tilde{\phi_0}$ codificarea rezultată din ϕ_0 eliminând ultimul simbol din $\phi_0(a_n)$. Lungimea medie a lui $\tilde{\phi_0}$ va fi evident mai mică decât cea a lui ϕ_0 (pentru că $P(a_n) > 0$), deci $\tilde{\phi_0}$ nu poate fi instantanee. Cuvântul - cod $\tilde{\phi_0}(a_i) = \phi_0(a_i), \quad (1 \leq i \leq n-1)$ nu este prefixul nici unui alt cuvânt cod; deci există un i $(i \leq n-1)$ astfel încât $\tilde{\phi_0}(a_n)$ este prefixul lui $\phi_0(a_i)$. Aceasta este posibil numai dacă $d_i = d_n$ şi deci $\phi_0(a_i)$ diferă de $\phi_0(a_n)$ numai prin ultimul simbol. Dacă $i = n-1$, se ia $\phi_1 = \phi_0$. Altfel, se observă că $d_i = d_n$ implică $d_i = d_{i+1} = \ldots = d_n$; deci se pot permuta cuvintele cod definite în ϕ_0 pentru a_i şi a_{n-1}. Codificarea ϕ_1 astfel obţinută are aceeaşi lungime medie ca şi ϕ_0, deci defineşte un cod Huffman.

- Să presupunem că se dă o codificare Huffman ϕ^* pentru sursa redusă S^* şi definim o codificare ϕ pentru S conform Tabelului 1.3. Lungimile lor medii $L(\phi)$, $L(\phi^*)$ verifică relaţia din Lema 1.1.

Să folosim acum codificarea Huffman ϕ_1 construită anterior. Deoarece ultimele două cuvinte - cod diferă numai prin ultimul simbol, ϕ_1 poate fi obţinut dintr-o codificare ϕ_1^* al lui S^* prin "spargerea" ultimului cuvânt - cod. În plus, ϕ_1^* este evident o codificare instantanee.

Prin calcule se ajunge la relaţia

$$L(\phi_1) - L(\phi_1^*) = P(a_{n-1}) + P(a_n).$$

Cum avem şi $L(\phi) - L(\phi^*) = P(a_{n-1}) + P(a_n)$, rezultă

$$L(\phi) = L(\phi_1) - L(\phi_1^*) + L(\phi^*).$$

Acum, $L(\phi^*) = L_{min}(S^*)$, deci $-L(\phi_1^*) + L(\phi^*) \leq 0$.

Rezultă $L(\phi) \leq L(\phi_1) = L_{min}(S)$. Deci, codificarea ϕ defineşte un cod Huffman.

q.e.d.

1.5 Exerciţii

Exerciţiul 1.1 *Care este cea mai mică lungime a unui cod - bloc cu alfabetul sursă* $A = \{a, b, \ldots, z\}$ *şi alfabetul cod* $B = \{., -, spa'tiu\}$ *(ca la codul Morse).*

Dar dacă B *are patru caractere ?*

Exerciţiul 1.2 *Se defineşte codificarea*

$$\begin{array}{lcllcl} 1 & \to & 01 & 4 & \to & 1000 \\ 2 & \to & 011 & 5 & \to & 1100 \\ 3 & \to & 10 & 6 & \to & 0111 \end{array}$$

Este ea unic decodabilă ? Este instantanee ? Se poate găsi un cod instantaneu cu aceleaşi lungimi ale cuvintelor cod ?

Exerciţiul 1.3 *Se defineşte codificarea*

$$\begin{array}{lcllcl} a & \to & 1010 & d & \to & 0001 \\ b & \to & 001 & e & \to & 1101 \\ c & \to & 101 & f & \to & 1011 \end{array}$$

Este ea unic decodabilă ? Dacă nu, găsiţi două mesaje sursă cu acelaşi cod.

Exerciţiul 1.4 *Este unic decodabilă codificarea:*

$$\begin{array}{lcllcllcl} 0 & \to & AA & 4 & \to & ABBAA & 7 & \to & AAAABB \\ 1 & \to & AABAB & 5 & \to & BABBA & 8 & \to & AAAABA \\ 2 & \to & ABBBBB & 6 & \to & BBBAB & 9 & \to & AAAAAB \\ 3 & \to & ABABA & & & & & & \end{array}$$

Exerciţiul 1.5 *Se poate decide unic decodabilitatea codificărilor*

$$\begin{array}{ll} \phi(a) = 001 & \phi(a) = 00 \\ \phi(b) = 1001 & \phi(b) = 10 \\ \phi(c) = 0010 & \phi(c) = 011 \\ \phi(d) = 1110 & \phi(d) = 101 \\ \phi(e) = 1010 & \phi(e) = 111 \\ \phi(f) = 01110 & \phi(f) = 110 \\ \phi(g) = 0101 & \phi(g) = 010 \end{array}$$

folosind inegalitatea lui Kraft ?

Exerciţiul 1.6 *Să se construiască un cod binar instantaneu pentru următorul alfabet sursă cu lungimile corespunzătoare ale cuvintelor cod:*

Simbol	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*
Lungime	2	4	7	7	3	4	7	7	3	4	7	7

Exerciţiul 1.7 *Să se construiască un cod instantaneu pentru următorul alfabet sursă, cu lungimile corespunzătoare ale cuvintelor cod:*

Simbol	1	2	3	4	5	6	7	8	9	0
Lungime	1	3	3	3	3	3	2	2	2	2

care este numărul minim e simboluri - cod necesare ?

Exerciţiul 1.8 *Câte simboluri cod sunt necesare pentru ca următorul alfabet sursă să poată fi codificat într-un cod instantaneu cu lungimile cuvintelor cod date:*

A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*
1	2	2	2	1	2	2	2	1	2	2	2	2	2	1	2

Exerciţiul 1.9 *Demonstraţi că pentru orice cod instantaneu în care inegalitatea Kraft este strictă, este posibil să se adauge un nou simbol sursă şi să se extindă codul dat la un nou cod instantaneu (cu acelaşi alfabet cod). Demonstraţi aceasta pentru codul definit la Exerciţiul 2.7.*

Capitolul 2

Coduri liniare

2.1 Matrice generatoare

Definiţia 2.1 *Fie q un număr prim şi $n \in N^*$ un număr natural nenul. Se numeşte "cod liniar" orice subspaţiu liniar al lui Z_q^n.*
Un subspaţiu k-dimensional ($k \leq n$) al lui Z_q^n se numeşte (n,k)- cod liniar peste Z_q.

Să notăm în general cu $A_{n,k}$ ($A_{n,k} \subseteq Z_q^n$) un (n,k) - cod liniar[1]. Elementele sale se numesc *cuvinte-cod*.

Fie $n, k \in N$, $(k < n)$. O *codificare* este o aplicaţie injectivă $\phi : Z_q^k \longrightarrow Z_q^n$, iar $A_{n,k} = \phi(Z_q^k)$. Elementele lui Z_q^k se numesc *mesaje de informaţie*.

Deci, $\mathbf{x} \in Z_q^k$ este un mesaj de informaţie scris cu caractere din alfabetul Z_q. Dacă transmitem succesiunea $\mathbf{x}$ de semnale printr-un canal de comunicaţie, mesajul este supus diverselor perturbaţii "de canal" care-l modifică. Ideea teoriei codurilor este următoarea: în loc de a transmite numai elementele lui Z_q^k, să "lungim" mesajul informaţional scufundând (prin intermediul aplicaţiei ϕ) Z_q^k într-un spaţiu liniar Z_q^n $(n > k)$ astfel încât cele $n-k$ poziţii noi - numite *poziţii de control* - să asigure redondanţa necesară refacerii mesajului de informaţie iniţial (atunci când a fost alterat parţial).

Astfel, cu preţul lungirii mesajului, se câştigă protecţia faţă de (anumite tipuri de) erori.

Observaţia 2.1

- *Din definiţie rezultă că un cod liniar de lungime n este un set C de cuvinte (secvenţe, şiruri, vectori) de lungime n cu proprietăţile:*
  - $\forall\ \mathbf{a} = (a_1, \ldots, a_n) \in C,\ \mathbf{b} = (b_1, \ldots, b_n) \in C \Longrightarrow \mathbf{a} + \mathbf{b} = (a_1 + b_1, \ldots, a_n + b_n) \in C$;
  - $\forall\ \mathbf{a} = (a_1, \ldots, a_n) \in A,\ t \in Z_q \Longrightarrow t \cdot \mathbf{a} = (ta_1, \ldots, ta_n) \in A$.
- *Orice cod liniar conţine cuvântul cod nul* $\mathbf{0} = (0, 0, \ldots, 0)$.
- $|A_{n,k}| = q^k$.

[1] Dacă nu este necesar sa specificăm dimensiunile n şi k, vom mai folosi şi notaţia C pentru a desemna un cod liniar.

$A_{n,k}$ fiind un spaţiu liniar k-dimensional, admite o bază formată din k vectori cu n elemente fiecare. Fie

$$\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_k}$$

o astfel de bază. Atunci orice cuvânt cod $\mathbf{v} \in A_{n,k}$ are forma

$$\mathbf{v} = \sum_{i=1}^{k} u_i \mathbf{e_i}$$

unde $(u_1, u_2, \ldots, u_k) \in Z_q^k$ este unic determinat.

Cu alte cuvinte, k simboluri de informaţie $u_1, \ldots, u_k \in Z_q$ determină în mod unic un cuvânt - cod $\mathbf{v} \in A_{n,k}$ prin relaţia de sus, şi reciproc. Operaţia de codificare ϕ face această asociere biunivocă:

$$\mathbf{u} = (u_1, \ldots, u_k) \in Z_q^k \Longleftrightarrow \mathbf{v} = \sum_{i=1}^{k} u_i \mathbf{e_i} \in A_{n,k}$$

Fiecare cuvânt - cod într-un cod liniar va avea deci k simboluri de informaţie; celelalte $n - k$ simboluri se numesc *simboluri de control*.

Definiţia 2.2 *Fie $A_{n,k}$ un cod liniar cu baza $\mathbf{e_1}, \ldots, \mathbf{e_k}$. Matricea*

$$G_{k,n} = \begin{pmatrix} \mathbf{e_1} \\ \mathbf{e_2} \\ \ldots \\ \mathbf{e_k} \end{pmatrix}$$

se numeşte "matricea generatoare" a codului.

Deci operaţia de codificare este definită prin matricea generatoare G:

$$\forall\ \mathbf{u} \in Z_q^k, \qquad \phi(\mathbf{u}) = \mathbf{u}G.$$

Exemplul 2.1 *Matricea*

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

generează un $(5, 2)$-cod liniar peste Z_2 $(n = 5,\ k = 2)$. Rangul ei este 2 (primele două coloane formează matricea unitate), deci cele două linii ale lui G sunt cuvinte - cod liniar independente.

Mulţimea mesajelor de informaţie este $Z_2^2 = \{00, 01, 10, 11\}$. Înmulţind fiecare mesaj cu matricea G se obţine spaţiul liniar al cuvintelor - cod

$$A_{5,2} = \{00000, 01101, 10011, 11110\}.$$

Funcţia de codificare este:

$$00 \to 00000, \quad 01 \to 01101, \quad 10 \to 10011, \quad 11 \to 11110.$$

Un cod liniar poate fi generat de mai multe baze posibile. Deci se pot construi mai multe matrici generatoare pentru un (n, k)-cod liniar. Ele se pot transforma una în alta prin operaţiile liniare obişnuite, definite pentru liniile unei matrici. Prin schimbarea matricii generatoare nu se schimbă spaţiul liniar al cuvintelor - cod, ci numai modalitatea de codificare (funcţia ϕ). Cum prin termenul *cod* se înţelege de obicei spaţiul liniar $A_{n,k}$, rezultă că două matrici diferite care se deduc una din alta prin operaţii pe linii, reprezintă acelaşi cod.

Exemplul 2.2 *Reluând Exemplul 2.1, prin adunarea liniei doi la prima linie se obţine matricea*

$$G' = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Ea generează acelaşi spaţiu liniar $A_{5,2}$, dar codificarea diferă:

$$00 \to 00000, \quad 01 \to 01101, \quad 10 \to 11110, \quad 11 \to 10011$$

Exemplul 2.3 *Matricea*

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

generează un $(6, 3)$-cod liniar peste Z_3. Aducând matricea la forma canonică, se obţine

$$G' = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Toate cele $3^3 = 27$ cuvinte ale acestui cod au următoarea proprietate: fiecare simbol este scris de două ori. Din acest motiv, astfel de cod este numit "cod cu repetiţie".

Cea mai convenabilă regulă de codificare constă în scrierea simbolurilor de informaţie şi suplimentarea lor cu $n - k$ simboluri de control. Aceasta corespunde matricii generatoare (unice)

$$G = (I|B)$$

unde I este matricea unitate de ordin k, iar B este o matrice cu k linii şi $n - k$ coloane.

Definiţia 2.3 *Un cod liniar este numit "sistematic" dacă admite o matrice generatoare de forma $G = (I|B)$ unde I este matricea unitate.*

Definiţia 2.4 *Două coduri liniare C şi C' de aceeaşi lungime n se numesc echivalente dacă există o permutare $\pi \in S_n$ astfel încât*

$$(v_1, v_2, \ldots, v_n) \in C \quad \Longleftrightarrow \quad v_{\pi(1)} v_{\pi(2)} \ldots v_{\pi(n)} \in C'$$

(s-a notat cu S_n mulţimea permutărilor de n elemente).

Exemplul 2.4 *Codul din Exemplul 2.1 este un cod sistematic. Din codificare se observă că mesajul de informaţie se află în cuvântul - cod pe primele două poziţii.*

Codul cu repetiţie din Exemplul 2.3 nu este sistematic. Totuşi, folosind permutarea $(1, 4, 6, 2, 5, 3)$ *(se permută simbolurile doi cu patru şi trei cu şase) se ajunge la un cod sistematic generat de matricea*

$$G^* = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Teorema 2.1 *Orice cod liniar este echivalent cu un cod liniar sistematic.*

Demonstraţie: Matricea generatoare G a unui (n, k) - cod liniar $A_{n,k}$ are rangul k; deci ea are k coloane liniar independente.

1. Să presupunem că primele k coloane ale lui G sunt liniar independente. Deci $G = (X|Y)$ unde X este o matrice $k \times k$ inversabilă. Există atunci o succesiune de operaţii de linii care transformă X în matricea unitate.

   Aceeaşi succesiune de operaţii efectuate acum pentru toată matricea G va conduce la matricea $G' = (I|Y')$. Deoarece şi G' este matrice generatoare pentru codul $A_{n,k}$, rezultă că acesta este sistematic.

2. Fie π permutarea care aduce coloanele liniar independente ale matricii G pe primele k poziţii. Se obţine în acest fel o nouă matrice G'. Fie $A'_{n,k}$ codul liniar obţinut prin codificarea cu matricea generatoare G'. Atunci $A_{n,k}$ şi $A'_{n,k}$ sunt echivalente, iar $A'_{n,k}$ este sistematic, conform cazului anterior. q.e.d.

2.2 Matrice de control

Definiţia 2.5 *Fie $A_{n,k}$ un cod liniar generat de matricea $G = G_{k,n}$. Se numeşte "matrice de control" o matrice $H = H_{n-k,n}$ cu proprietatea $GH^T = \mathbf{0}$.*

Observaţia 2.2

- *Din definiţie rezultă că matricea de control H a unui cod liniar C are următoarea proprietate:*

  $$\mathbf{v} \in C \quad \Longleftrightarrow \quad \mathbf{v}H^T = \mathbf{0}$$

  Lăsăm ca exerciţiu demonstrarea acestei echivalenţe.

- *Prin transpunere, relaţia de sus se poate scrie şi $HG^T = \mathbf{0}$. Aceasta înseamnă că şi H este matricea generatoare a unui $(n, n-k)$ - cod liniar peste corpul Z_q, cod pentru care G este matrice de control. Cele două coduri astfel definite se numesc "coduri duale".*

  Cuvintele - cod din cele două coduri duale sunt ortogonale (produsul lor scalar este 0). Într-adevăr, dacă C şi C' sunt două coduri duale generate de matricile G respectiv H, iar $\mathbf{x} \in C, \mathbf{y} \in C'$ sunt cuvinte - cod arbitrare, există $\mathbf{u}, \mathbf{v}$ cu $\mathbf{x} = \mathbf{u}G$, $\mathbf{y} = \mathbf{v}H$. În plus, $\mathbf{x}H^T = \mathbf{0}$, $\mathbf{y}G^T = \mathbf{0}$.

  Atunci, $\mathbf{x}\mathbf{y}^T = \mathbf{u}G\mathbf{y}^T = \mathbf{u}(\mathbf{y}G^T)^T = \mathbf{u}\mathbf{0}^T = \mathbf{0}$.

Exemplul 2.5 $(7,4)$ - *codul liniar binar cu matricea generatoare*

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

are drept matrice de control

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

care la rândul ei este matricea generatoare a unui $(7,3)$ - *cod liniar binar.*

Se verifică imediat relaţia

$$GH^T = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Un cod care coincide cu codul său dual se numeşte *cod auto - dual.*

Teorema 2.2 *Un cod liniar sistematic cu matricea generatoare* $G = (I|B)$ *admite ca matrice de control* $H = (-B^T|I)$.

Demonstraţie: Cele două matrici unitate din scrierea lui G şi H sunt de ordin k respectiv $n-k$. Efectuând calculele, se obţine:

$$GH^T = (I|B)\begin{pmatrix} -B \\ I \end{pmatrix} = -IB + BI = -B + B = \mathbf{0}. \quad q.e.d.$$

Corolarul 2.1 *Matricea de control a unui* (n,k)*-cod liniar are rangul* $n-k$.

Teorema de sus permite un algoritm de calcul extrem de simplu al matricii de control, atunci când se cunoaşte matricea generatoare. Să arătăm aceasta pe un exemplu:

Exemplul 2.6 *Plecând de la matricea generatoare construită în Exemplul 2.4, se poate construi matricea de control (calculele se fac în* Z_3*):*

$$H^* = \begin{pmatrix} -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Aplicând acum permutarea inversă coloanelor lui H^**, se ajunge la matricea de control a codului definit în Exemplul 2.3:*

$$H = \begin{pmatrix} 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 2 & 0 & 0 \end{pmatrix}$$

Relaţia $\mathbf{x}H^T = \mathbf{0}$ pe care o verifică orice cuvânt - cod $\mathbf{x} \in A_{n,k}$ permite să definim un cod şi sub forma unui sistem de ecuaţii.

Astfel, $A_{n,k}$ este un cod peste Z_q dacă şi numai dacă elementele sale sunt soluţii ale sistemului liniar $\mathbf{x}H^T = \mathbf{0}$.

Exemplul 2.7 *Codul cu matricea de control* $H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \end{pmatrix}$

este definit ca mulţimea soluţiilor binare $(x_1, x_2, x_3, x_4, x_5)$ *ale sistemului*

$$x_1 + x_2 + x_4 = 0, \qquad x_1 + x_3 + x_4 + x_5 = 0$$

Sistemul are 8 *soluţii, care formează codul binar*

$$A_{5,2} = \{00000, 00101, 01110, 01011, 11100, 11001, 10010, 10111\}$$

2.3 Sindrom

Fie codul liniar $A_{n,k}$ peste Z_q, cu matricea de control H şi $\mathbf{a} \in Z_q^n$. Se numeşte *sindrom* al cuvântului $\mathbf{a}$ vectorul $\mathbf{z}$ cu $n-k$ componente obţinut prin relaţia

$$\mathbf{z}^T = H\mathbf{a}^T,$$

sau – echivalent – $\mathbf{z} = \mathbf{a}H^T$.

Observaţia 2.3 $\mathbf{z} = \mathbf{0} \iff \mathbf{a} \in A_{n,k}$.

Dacă se transmite printr-un canal de comunicaţie un cuvânt $\mathbf{a}$ pentru care sindromul corespunzător verifică relaţia $\mathbf{z} = \mathbf{a}H^T \neq \mathbf{0}$, înseamnă că *s-a detectat* faptul că în timpul transmisiei au apărut erori.

Teorema 2.3 *În* Z_2*, sindromul recepţionat este egal cu suma coloanelor din matricea de control, corespunzătoare poziţiilor perturbate.*

Demonstraţie: Fie $\mathbf{a} = (a_1, a_2, \ldots, a_n) \in A_{n,k}$ cuvântul-cod transmis. Fără a restrânge generalitatea, să presupunem că au intervenit trei erori, pe poziţiile i, j, k, fiind recepţionat vectorul

$$\mathbf{a}' = (a_1, \ldots, a_{i-1}, a_i', a_{i+1}, \ldots, a_{j-1}, a_j', a_{j+1}, \ldots, a_{k-1}, a_k', a_{k+1}, \ldots, a_n)$$

unde $a_i' \neq a_i,\ a_j' \neq a_j,\ a_k' \neq a_k$. Avem

$$\mathbf{0} = \mathbf{a}H^T = a_1(h_1) + \ldots + a_i(h_i) + \ldots + a_j(h_j) + \ldots + a_k(h_k) + \ldots + a_n(h_n)$$

pentru că $\mathbf{a} \in A_{n,k}$, iar $(h_1), \ldots, (h_n)$ sunt coloanele matricii H.

Avem, de asemenea

$$\mathbf{a}'H^T = a_1(h_1) + \ldots + a_i'(h_i) + \ldots + a_j'(h_j) + \ldots + a_k'(h_k) + \ldots + a_n(h_n).$$

Prin adunarea acestor două egalităţi se obţine:

$\mathbf{z}' = \mathbf{a}'H^T = \mathbf{a}H^T + \mathbf{a}'H^T = (0) + \ldots + (0) + (h_i) + \ldots + (h_j) + \ldots + (h_k) + (0) + \ldots + (0) = (h_i) + (h_j) + (h_k)$. q.e.d.

2.4 Pondere, distanţă Hamming

Pentru orice cuvânt $\mathbf{x} \in Z_q^n$, se numeşte *pondere* numărul $w(\mathbf{x})$ de componente nenule ale lui $\mathbf{x}$. Evident, $0 \leq w(\mathbf{x}) \leq n$.

Pentru două cuvinte $\mathbf{x}, \mathbf{y} \in Z_q^n$, se numeşte *distanţa Hamming* între ele, numărul

$$d(\mathbf{x}, \mathbf{y}) = w(\mathbf{x} - \mathbf{y}).$$

Propoziţia 2.1 *d este o distanţă definită pe Z_q^n.*

Demonstraţie: Se verifică uşor proprietăţile unei distanţe:

1. $d(\mathbf{x}, \mathbf{x}) = 0$;
2. $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$;
3. $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z})$.

Ca o remarcă, deoarece $\mathbf{x} - \mathbf{y} \in Z_q^n$, rezultă că distanţa Hamming dintre două cuvinte este ponderea unui cuvânt din Z_q^n.

Cu ajutorul lui d, mulţimea Z_q^n se poate structura ca spaţiu metric.

Definiţia 2.6 *Se numeşte distanţă (Hamming) a codului liniar $A_{n,k} \subset Z_q^n$ cea mai mică distanţă (Hamming) dintre elementele codului $A_{n,k}$, adică*

$$d = \min_{\mathbf{x},\mathbf{y} \in A_{n,k}, \mathbf{x} \neq \mathbf{y}} d(\mathbf{x}, \mathbf{y})$$

Folosind proprietatea anterioară şi faptul că $\mathbf{0} \in A_{n,k}$, rezultă că

$$d = \min_{\mathbf{x} \in A_{n,k}, \mathbf{x} \neq \mathbf{0}} w(\mathbf{x}).$$

Adesea valoarea d este introdusă printre parametrii generali ai codului, folosind notaţia de (n, k, d)-cod.

Teorema 2.4 *Fie H matricea de control a unui cod liniar $A_{n,k}$. Codul are distanţa minimă d dacă şi numai dacă orice combinaţie liniară de $d-1$ coloane ale lui H este liniar independentă şi există cel puţin o combinaţie liniară de d coloane liniar dependente.*

Demonstraţie: Pentru orice cuvânt $\mathbf{a}$, cu $w(\mathbf{a}) = s$, $\mathbf{a}H^T$ este o combinaţie liniară de s coloane ale lui H. Cum există un cuvânt - cod de pondere minimă egală cu distanţa d a codului, rezultă că avem cel puţin o combinaţie de d coloane liniar dependente ale lui H. O combinaţie de mai puţin de d coloane liniar dependente ar conduce la găsirea unui cuvânt - cod de pondere mai mică decât d, deci contradicţie. q.e.d.

Definiţia 2.7 *Pentru $r > 0$ şi $\mathbf{x} \in Z_q^n$, definim sfera de rază r şi centru $\mathbf{x}$ ca*

$$S_r(\mathbf{x}) = \{\mathbf{y} \mid d(\mathbf{x}, \mathbf{y}) \leq r\}$$

Teorema 2.5 *Fie $A_{n,k} \subseteq Z_q^n$ un cod liniar cu distanţa Hamming d. Dacă $r = \left[\frac{d-1}{2}\right]$, atunci*

$$\forall \mathbf{x}, \mathbf{y} \in A_{n,k} \quad [\mathbf{x} \neq \mathbf{y} \quad \Longrightarrow \quad S_r(\mathbf{x}) \cap S_r(\mathbf{y}) = \emptyset] .$$

Demonstraţie: Presupunem prin absurd că există $\mathbf{z} \in Z_q^n$, $\mathbf{z} \in S_r(\mathbf{x}) \cap S_r(\mathbf{y})$. Atunci $d(\mathbf{x}, \mathbf{z}) \leq r$, $d(\mathbf{y}, \mathbf{z}) \leq r$ deci, conform inegalităţii triunghiului, $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{y}, \mathbf{z}) \leq 2r = 2\left[\frac{d-1}{2}\right] < d$, ceea ce contrazice afirmaţia că d este distanţa minimă a codului $A_{n,k}$. q.e.d.

Evident, în fiecare astfel de sferă există un singur cuvânt - cod: cel aflat în centrul sferei.

Definiţia 2.8 *Un cod liniar $A_{n,k} \subset Z_q^n$ de distanţă minimă d este perfect dacă*

$$Z_q^n = \bigcup_{\mathbf{x} \in A_{n,k}} S_r(\mathbf{x})$$

unde $t = \left[\frac{d-1}{2}\right]$.

2.5 Detectare şi corectare de erori

Fie $A_{n,k} \subset Z_q^n$ un cod liniar de distanţă minimă d şi $t = \left[\frac{d-1}{2}\right]$. Să presupunem că s-a recepţionat cuvântul $\mathbf{z} \in Z_q^n$; va exista cel mult un cuvânt $\mathbf{x} \in A_{n,k}$ astfel încât $\mathbf{z} \in S_t(\mathbf{x})$ (în cazul codurilor perfecte, acest cuvânt există totdeauna).

În cazul (ideal) când $\mathbf{z} = \mathbf{x}$, cuvântul a fost transmis fără erori (sau cu erori nedetectabile).

Dacă $\mathbf{z} \neq \mathbf{x}$, atunci mesajul a fost perturbat (şi avem o detectare de erori); în ipoteza că numărul de erori apărute este minim şi există un $\mathbf{x} \in A_{n,k}$ astfel ca $d(\mathbf{x}, \mathbf{z}) \leq t$, atunci $\mathbf{z}$ provine din cuvântul - cod $\mathbf{x}$ - şi se va transforma în acesta prin corectarea corespunzătoare a erorilor.

În celelalte cazuri, $\mathbf{z}$ sau nu se poate corecta, sau se corectează greşit, în alt cuvânt cod.

Această ultimă situaţie nu apare la codurile perfecte.

Metoda de detectare şi corectare a erorilor descrisă mai sus se numeşte *decodificarea cea mai probabilă*. Pentru marea majoritate a codurilor liniare aceasta este singura metodă utilizată.

Fie $C \subseteq Z_q^n$ un cod liniar; pentru orice cuvânt $\mathbf{e} \in Z_q^n$ formăm mulţimea

$$V_{\mathbf{e}} = \mathbf{e} + C = \{\mathbf{e} + \mathbf{x} \mid \mathbf{x} \in C\}$$

$V_{\mathbf{e}}$ este mulţimea cuvintelor $\mathbf{w} = \mathbf{e} + \mathbf{x}$ care pot fi recepţionate la transmiterea cuvântului - cod $\mathbf{x}$, atunci când a acţionat un vector - eroare $\mathbf{e}$. $\mathbf{e}$ va fi numit *eroare - tip*.

Se verifică imediat că V_e este subspaţiu liniar al lui Z_q^n (a se vedea Observaţia 2.1).

În particular, $C = \mathbf{0} + C = V_{\mathbf{0}}$.

Propoziţia 2.2 *Pentru orice $\mathbf{a} \in V_{\mathbf{e}}$, $V_{\mathbf{a}} = V_{\mathbf{e}}$.*

Demonstraţie: Din $\mathbf{a} \in V_{\mathbf{e}}$, rezultă că există $\mathbf{x} \in C$ cu $\mathbf{a} = \mathbf{e} + \mathbf{x}$. Deoarece $\mathbf{x} + C = C$ (evident, C fiind subspaţiu liniar), avem $V_{\mathbf{a}} = \mathbf{a} + C = \mathbf{e} + \mathbf{x} + C = \mathbf{e} + C = V_{\mathbf{e}}$. q.e.d.

Vom utiliza această propoziţie pentru definirea unei tehnici generale de decodificare.

Dacă vrem să detectăm o anumită eroare - tip $\mathbf{e}$ care modifică cuvintele din C în cuvinte ale subspaţiului $V_{\mathbf{e}}$, atunci vor fi mai uşor de depistat cuvintele din $V_{\mathbf{e}}$ cu ponderea minimă.

Pentru fiecare $V_{\mathbf{e}}$ se alege un cuvânt numit *reprezentantul* lui $V_{\mathbf{e}}$; acesta este un element cu cea mai mică pondere nenulă din $V_{\mathbf{e}}$.

Se construieşte următorul tablou (numit *tablou standard*):

1. Pe prima linie se scriu cuvintele - cod, începând cu $\mathbf{0}$ (reprezentantul lui $C = V_{\mathbf{0}}$);
2. Pe prima coloană se scriu reprezentanţii $\mathbf{0}, \mathbf{e_1}, \mathbf{e_2}, \ldots$;
3. Pe linia cu reprezentantul $\mathbf{e_i}$, sub cuvântul cod $\mathbf{x_j}$ se scrie cuvântul $\mathbf{e_i} + \mathbf{x_j}(\bmod q)$.

Exemplul 2.8 *Fie matricea generatoare* $G = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$ *peste* Z_2. *Ea va codifica* $Z_2^2 = \{00, 01, 10, 11\}$ *în* $A_{4,2} = \{0000, 1001, 0111, 1110\}$.

Calculul mulţimilor $V_{\mathbf{e}}$ *conduce la*

$$\begin{array}{cccccccc} V_{0000} & = & V_{1001} & = & V_{0111} & = & V_{1110} \\ V_{1000} & = & V_{0001} & = & V_{0110} & = & V_{1111} \\ V_{0100} & = & V_{1101} & = & V_{0011} & = & V_{1010} \\ V_{0010} & = & V_{1011} & = & V_{0101} & = & V_{1100} \end{array}$$

Alegem ca reprezentanţi pe $0000, 1000$ *(se poate şi* $0001), 0100, 0010$. *Tabloul standard va fi:*

0000	1001	0111	1110
1000	0001	1111	0110
0100	1101	0011	1010
0010	1011	0101	1100

Pentru destinatar, acest tablou este ca un dicţionar care se utilizează astfel: cuvântul primit $\mathbf{a}$ *se decodifică în cuvântul - cod din capul coloanei pe care se află* $\mathbf{a}$.

De exemplu, dacă se recepţionează 1101, *el se va decodifica în* 1001.

Evident, cuvintele de pe prima coloană se decodifică în ele însele (sunt cuvinte - cod şi nu au fost perturbate de nici o eroare).

Pentru orice cod liniar, un dicţionar complet de tipul celui de mai sus constituie cea mai simplă metodă de decodificare.

Problema apare atunci când într-o mulţime $V_{\mathbf{e}}$ sunt mai multe cuvinte de pondere minimă. Atunci tabela va decodifica corect numai eroarea - tip aleasă ca reprezentant.

Astfel, revenind la Exemplul 2.8, cuvântul recepţionat 1111 se decodifică în 0111 (considerând că a fost alterat primul caracter).

Dacă se ia însă drept reprezentant pe linia a doua 0001 în loc de 1000, a doua linie din tabloul standard este

0001 1000 0110 1111

Atunci, 1111 se decodifică în 1110 (considerînd ultimul caracter ca fiind cel alterat de canalul de transmisie). Care este cuvântul - cod corect transmis ?

Acest lucru nu poate fi decis. Singurul lucru care poate fi făcut este să se aleagă drept reprezentanţi erorile - tip cele mai probabile.

Pentru un (n, k) - cod peste Z_q, un dicţionar complet constă din toate cele q^n cuvinte posibile, lucru destul de dificil deoarece în practică codurile sunt destul de lungi ($n \geq 100$).

De aceea este utilizată o altă manieră de lucru care reduce mult mărimea tabloului standard.

Fie H matricea de control a unui (n, k) - cod liniar $A_{n,k}$; putem considera (Corolarul 2.1) că liniile lui H sunt vectori liniar independenţi.

Teorema 2.6 *Pentru orice $\mathbf{e} \in Z_q^n$, toate cuvintele din $V_\mathbf{e}$ au acelaşi sindrom.*

Demonstraţie: Fie $\mathbf{v} \in A_{n,k}$ arbitrar şi $\mathbf{w} = \mathbf{e} + \mathbf{v}$. Avem

$$\mathbf{s} = \mathbf{w}H^T = (\mathbf{e} + \mathbf{v})H^T = \mathbf{e}H^T + \mathbf{v}H^T = \mathbf{e}H^T + \mathbf{0} = \mathbf{e}H^T.$$

Deci toate cuvintele din $V_\mathbf{e}$ au sindromul egal cu sindromul lui $\mathbf{e}$. q.e.d.

Invers, pentru fiecare sindrom – deci pentru fiecare cuvânt $\mathbf{s}$ de lungime $n-k$, se poate determina un vector - eroare $\mathbf{e}$ având sindromul $\mathbf{s}$.

Mai mult, $\mathbf{s}$ va fi ales astfel încât să aibă pondere minimă (conform decodificării cele mai probabile). Pentru aceasta, se rezolvă sistemul de ecuaţii liniare $H\mathbf{e}^T = \mathbf{s}^T$, care are soluţie, deoarece liniile lui H sunt liniar independente.

Din mulţimea soluţiilor alegem una de pondere minimă, cu ajutorul căreia construim mulţimea $V_\mathbf{e}$ a tuturor cuvintelor de sindrom $\mathbf{s}$.

Pe baza celor de mai sus, se poate folosi următoarea procedură de decodificare:

1. La recepţionarea unui cuvânt $\mathbf{w}$, se calculează sindromul $\mathbf{s}$:
$$\mathbf{s}^T = H\mathbf{w}^T.$$
2. Se află eroarea - tip $\mathbf{e}$ cu sindromul $\mathbf{s}$.
3. Se consideră cuvântul - cod corect ca fiind $\mathbf{v} = \mathbf{w} - \mathbf{e}$.

În acest fel, nu mai este necesar să se reţină tot tabloul standard; este suficient să se ştie doar reprezentanţii subspaţiilor $V_\mathbf{e}$ şi sindromurile corespunzătoare.

Exemplul 2.9 *Reluând codul definit în Exemplul 2.8, el are ca matrice de control*

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

Calculând sindromurile reprezentanţilor, se ajunge la tabloul:

Sindrom	*Reprezentant*
00	0000
01	1000
10	0010
11	0100

care reprezintă o reducere cu 50% *a datelor stocate (comparativ cu tabloul standard). De exemplu, la recepţionarea cuvântului* 1101*, se calculează sindromul*

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

Reprezentantul sindromului 11 *este* 0100*. Efectuând operaţia*

$$1101 - 0100 = 1101 + 0100 = 1001$$

(în Z_2 *scăderea este de fapt adunare) se ajunge la cuvântul transmis, anume* 1001*.*

Exemplul 2.10 *Să considerăm codul liniar* $A_{5,3}$ *peste* Z_3 *definit de matricea de control*

$$H = \begin{pmatrix} 1 & 0 & 2 & 1 & 0 \\ 0 & 1 & 1 & 2 & 2 \end{pmatrix}$$

Sindromurile sunt cuvinte de lungime 2 *peste* Z_3*, deci în total nouă cuvinte. Lista sindromurilor şi a reprezentanţilor este:*

Sindrom	*Reprezentant*
00	00000
10	10000
01	01000
22	10010
11	11000
02	00001
20	20000
12	00010
21	00100

De remarcat că un tablou standard pentru acest cod are dimensiunea 9×27 *şi conţine* 243 *cuvinte; volumul de date s-a redus deci cu* 92%*.*

Să presupunem că s-a trimis cuvântul cod 01221 *şi s-a recepţionat* 01201*.*

Sindromul este

$$\begin{pmatrix} 1 & 0 & 2 & 1 & 0 \\ 0 & 1 & 1 & 2 & 2 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 2 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

deci $\mathbf{e} = 00010$*; decodificarea este*

$$\mathbf{v} = \mathbf{w} - \mathbf{e} = 01201 - 00010 = 01221.$$

Decodificarea a fost corectă deoarece eroarea - tip apărută a fost una din cele selectate prin reprezentanţi.

2.6 Exerciţii

2.1 *Să se construiască coduri liniare care să transforme cuvântul* $(a_1, a_2, \ldots, a_n) \in Z_q^n$ *în:*

1. $(a_1, a_1, a_1, a_2, a_2, a_2, \ldots, a_n, a_n, a_n)$;
2. $(a_1, b_1, a_2, b_2, \ldots, a_n, b_n)$ *unde* $b_1 = a_1, b_2 = a_1 + a_2, b_n = a_1 + \ldots + a_n$;
3. $(a_1, 2a_n, 3a_2, 4a_{n-1}, \ldots)$.

2.2 *Un cod liniar peste* Z_5 *are următoarea matrice generatoare:*

$$G = \begin{pmatrix} 1 & 2 & 3 & 1 & 2 \\ 2 & 2 & 4 & 1 & 0 \\ 1 & 1 & 2 & 2 & 1 \end{pmatrix}$$

Să se afle matricea de control.

2.3 *Aceeaşi problemă pentru* Z_7.

2.4 *Determinaţi dacă următorul cod liniar binar este sistematic sau nu:*

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

Dacă nu, găsiţi un cod sistematic echivalent.

2.5 *Găsiţi matricea generatoare a unui cod liniar binar care are matricea de control:*

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

2.6 *Se dă matricea de control*

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

a unui cod liniar binar.
Să se decodifice mesajele 110110, 010100.

2.7 *Descrieţi dualul* $(6, 3)$ *- codului binar descris de ecuaţiile:*

$$x_1 = x_4, \quad x_2 = x_5, \quad x_3 = x_6.$$

2.8 *Fie A un cod liniar binar obţinut din toate sumele posibile ale cuvintelor* 101011, 011101, 011010.

1. *Aflaţi o matrice de control a codului.*

2. *Aflaţi un tablou standard şi decodificaţi* 111011.

2.9 *Demonstraţi că codul liniar binar descris de ecuaţiile*

$$x_3 = x_1 + x_2, \quad x_4 = x_1, \quad x_5 = x_1 + x_2$$

corectează o eroare. Construiţi tabloul standard.

2.10 *Construiţi o tabelă de sindromuri de decodificare pentru:*

1. *Codul cu repetiţie de lungime* 7*;*

2. *Codul din Exerciţiul 2.7;*

3. *Codul peste* Z_3 *cu matricea generatoare:*

$$G = \begin{pmatrix} 1 & 0 & 0 & 2 & 2 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

2.11 *Fie un cod liniar* $A_{n,k} \subseteq Z_q^n$ *şi tabela de sindromuri de decodificare corespunzătoare. Definim o operaţie de decodificare* $\phi : Z_q^n \to A_{n,k}$ *cu proprietatea:* $\forall \mathbf{v} \in A_{n,k}, \forall \mathbf{e}$ *reprezentant din tabelă,* $\phi(\mathbf{v} + \mathbf{e}) = \mathbf{v}$*. (*$\phi$ *corectează erorile - tip din tabela standard). Să se arate că atunci* ϕ *nu corectează nici o altă eroare - tip: dacă* $\mathbf{e}$ *nu este un reprezentant din tabelă, atunci există* $\mathbf{v} \in A_{n,k}$ *cu* $\phi(\mathbf{v} + \mathbf{e}) \neq \mathbf{v}$*.*

(Altfel spus, decodificarea cu sindromuri este optimală).

Capitolul 3

Coduri liniare - II

3.1 Capacităţi de detectare şi corectare a erorilor

După cum am văzut pâ nă acum, un cod liniar este un spaţiu liniar, codul dual este complementul său ortogonal etc. Ceea ce interesează aici este depistarea unor coduri cu proprietăţi deosebite în detectarea şi corectarea erorilor. Aceste proprietăt i sunt legate în special de distanţa minimă a codului. Vom prezenta pentru început câteva rezultate legate de diverse margini relativ la distanţă, numărul de simboluri de control, informaţie etc.

Teorema 3.1 *Un cod liniar $A_{n,k} \subseteq Z_q^n$ are distanţa minimă d dacă şi numai dacă poate detecta orice combinaţie de maxim $d-1$ erori.*

Demonstraţie: Dacă se transmite un cuvânt $\mathbf{a} \in A_{n,k}$ şi apar t $(0 \leq t < d)$ erori, se va recepţiona cuvântul $\mathbf{a}+\mathbf{e} \in Z_q^n$ cu $w(\mathbf{e}) = t$.

Deoarece $d(\mathbf{a}, \mathbf{a}+\mathbf{e}) = t$, rezultă $\mathbf{a}+\mathbf{e} \notin A_{n,k}$, deci se detectează eroare.

Afirmaţia reciprocă este evidentă. q.e.d.

Teorema 3.2 *Un cod liniar $A_{n,k} \subseteq Z_q^n$ are distanţa minimă d dacă şi numai dacă poate corecta orice combinaţie de $t \leq \left[\dfrac{d-1}{2}\right]$ erori.*

Demonstraţie: Fie t un număr întreg pozitiv; să presupunem că $d \leq 2t$ şi vom arăta că există erori - tip de pondere t (numite şi *pachete de t erori independente*) care nu pot fi corectate de codul $A_{n,k}$.

Aceasta va duce la concluzia că $d \geq 2t+1$, adică $A_{n,k}$ poate corecta orice pachet de $t \leq \left[\dfrac{d-1}{2}\right]$ erori.

Fie $\mathbf{a}, \mathbf{b} \in A_{n,k}$ cu $d(\mathbf{a}, \mathbf{b}) = d$ şi $i_1, i_2, \ldots, i_d$ toţi indicii în care $\mathbf{a}$ diferă de $\mathbf{b}$. Alegând $\dfrac{d}{2} \leq t \leq \dfrac{d+1}{2}$, avem $d \leq 2t$ $\quad$ (t astfel ales este minim).

Să presupunem că se trimite $\mathbf{a}$ şi se recepţionează cuvântul $\mathbf{a}' = (a_1', a_2', \ldots, a_n')$ unde

$$a_i' = \begin{cases} a_i\ (= b_i) & \text{dacă} \quad i \neq i_1, i_2, \ldots, i_d, \\ b_i & \text{dacă} \quad i = i_1, i_3, \ldots, \\ a_i & \text{dacă} \quad i = i_2, i_4, \ldots \end{cases}$$

Atunci, evident $d(\mathbf{a}, \mathbf{a}') = \left[\frac{d+1}{2}\right] = t$ şi $d(\mathbf{a}', \mathbf{b}) = \left[\frac{d}{2}\right] \leq t = d(\mathbf{a}', \mathbf{a})$. Aceasta va duce la decodificarea lui $\mathbf{a}$ în $\mathbf{b}$ - incorect.

Să presupunem acum $d \geq 2t+1$. Atunci codul $A_{n,k}$ poate corecta orice pachet de t erori.

Pentru a arăta aceasta, să presupunem că se trimite un cuvânt - cod $\mathbf{a}$ şi se primeşte un cuvânt $\mathbf{a}'$ cu $d(\mathbf{a}, \mathbf{a}') \leq t$. Pentru orice cuvânt - cod $\mathbf{b}$ (deci cu $d(\mathbf{a}, \mathbf{b}) \geq d \geq 2t+1$) avem, conform inegalităţii triunghiului:

$$d(\mathbf{a}, \mathbf{a}') + d(\mathbf{a}', \mathbf{b}) \geq d(\mathbf{a}, \mathbf{b}) \geq 2t+1$$

de unde rezultă $d(\mathbf{a}', \mathbf{b}) \geq 2t+1-d(\mathbf{a}, \mathbf{a}') \geq 2t+1-t = t+1 > d(\mathbf{a}', \mathbf{a})$.

Deci, în ipoteza celei mai probabile decodificări, $\mathbf{a}'$ se va decodifica în $\mathbf{a}$. q.e.d.

Teorema 3.3 *Distanţa minimă a unui (n, k) - cod liniar verifică relaţia*

$$d \leq n-k+1.$$

Demonstraţie: Vom separa demonstraţia în două părţi:

A: Fie $A_{n,k}$ un cod sistematic. Atunci primele k simboluri din orice cuvânt - cod pot fi alese arbitrar (ele formează mesajul de informaţie, un element arbitrar din Z_q^k). Fie $\mathbf{v} \in A_{n,k}$ cuvântul de forma $\mathbf{v} = 100\ldots0v_{k+1}v_{k+2}\ldots v_n$. Evident, $0 < w(\mathbf{v}) \leq n-k+1$.

Cum d este cea mai mică pondere a unui cuvânt - cod nenul, rezultă inegalitatea cerută.

B: Fie $A_{n,k}$ un cod liniar arbitrar şi $A'_{n,k}$ codul liniar sistematic echivalent. Se observă că cele două coduri au aceiaşi parametri n, k, d. Folosind acum **A**, inegalitatea se obţine din nou. q.e.d.

Teorema 3.4 *În orice cod liniar $A_{n,k} \subseteq Z_q^n$ avem*

$$d \leq \frac{nq^{k-1}(q-1)}{q^k-1} \qquad \textit{(marginea Plotkin)}$$

Demonstraţie: Să considerăm elementele din $A_{n,k}$ aşezate ca linii ale unui tablou. Se obţine un tablou cu q^k linii şi n coloane. Fiecare componentă nenulă din Z_q apare pe fiecare coloană în q^{k-1} linii. Atunci, suma ponderilor tuturor cuvintelor - cod este egală cu $nq^{k-1}(q-1)$ deoarece fiecare componentă nenulă apare de q^{k-1} ori în fiecare coloană şi avem $q-1$ componente nenule distribuite pe n coloane.

Distanţa minimă a codului nu poate să depăşească ponderea medie a cuvintelor codului, adică

$$d \leq \frac{nq^{k-1}(q-1)}{q^k-1}$$

deoarece în $A_{n,k}$ sunt q^k-1 cuvinte nenule. q.e.d.

Teorema 3.5 *Fie $A_{n,k}$ un cod liniar peste Z_q care corectează orice combinaţie de maxim t erori. Într-un asemenea cod sunt necesare cel puţin*

$$n-k \geq log_q[1+C_n^1(q-1)+C_n^2(q-1)^2+\ldots+C_n^t(q-1)^t]$$

poziţii de control (marginea Hamming).

Demonstraţie: Pentru ca $A_{n,k}$ să corecteze orice combinaţie de cel mult t erori, este necesar ca fiecare astfel de eroare - tip să fie un reprezentant în tabloul standard; deci să fie caracterizată printr-un sindrom distinct.

Sunt q^{n-k} sindromuri distincte, deci acesta este numărul maxim de erori care pot fi corectate de cod. Din cele q^{n-k} sindromuri, 1 trebuie să fie pentru 0 erori, $C_n^1(q-1)$ – pentru erori - tip simple (cuvinte $\mathbf{e}$ cu o singură componentă nenulă), $C_n^2(q-1)^2$ – pentru erori duble etc.

Deci, este necesar ca

$$q^{n-k} \geq 1 + C_n^1(q-1) + C_n^2(q-1)^2 + \ldots + C_n^t(q-1)^t.$$

Apoi se logaritmează. q.e.d.

Teorema 3.6 *Dacă*

$$n - k \geq log_q[1 + C_{n-1}^1(q-1) + C_{n-1}^2(q-1)^2 + \ldots + C_{n-1}^{d-2}(q-1)^{d-2}]$$

atunci există un cod liniar $A_{n,k} \subseteq Z_q^n$ cu distanţa minimă d (marginea Varşamov - Gilbert).

Demonstraţie: Pentru ca să existe un cod liniar $A_{n,k}$ peste Z_q cu distanţa minimă d este suficient (Teorema 2.4) ca orice coloană din matricea de control $H_{n-k,n}$ să nu fie combinaţie liniară a altor $d-2$ coloane, în acest fel ne-existând nici o combinaţie liniară între $d-1$ coloane ale lui H.

Această condiţie este echivalentă cu

$$q^{n-k} - 1 \geq C_{n-1}^1(q-1) + C_{n-1}^2(q-1)^2 + \ldots + C_{n-1}^{d-2}(q-1)^{d-2}.$$

Aici, $q^{n-k} - 1$ reprezintă numărul total de coloane distincte nenule care pot apare în matricea H. Semnificaţia termenilor din membrul drept este evidentă; astfel, de exemplu $C_{n-1}^2(q-1)^2$ reprezintă numărul combinaţiilor liniare cu coeficienţi nenuli a două din cele $n-1$ coloane etc.

Apoi se logaritmează. q.e.d.

Fie $A_{n,k} \subseteq Z_q^n$ un cod liniar pentru care $d \geq 2t+1$ (deci cu capacitatea de a corecta orice combinaţie de maxim t erori independente). Reamintim că în prelegerea precedentă am definit pentru orice $\mathbf{x} \in A_{n,k}$ sfera centrată în $\mathbf{x}$ prin

$$S_t(\mathbf{x}) = \{\mathbf{y} \in Z_q^n \mid d(\mathbf{x}, \mathbf{y}) \leq t\}.$$

Mai introducem şi suprafaţa (scoarţa) acestei sfere, definită:

$$A_t(\mathbf{x}) = \{\mathbf{y} \in Z_q^n \mid d(\mathbf{x}, \mathbf{y}) = t\}.$$

Vom nota numărul de elemente ale fiecăreia din cele două mulţimi prin

$$S_t = |S_t(\mathbf{x})|, \qquad A_t = |A_t(\mathbf{x})|$$

(datorită proprietăţii de omogenitate a spaţiilor liniare, valorile S_t şi A_t sunt aceleaşi pentru orice $\mathbf{x} \in Z_q^n$).

Au loc relaţiile evidente:

$$A_t \leq S_t, \qquad S_t = \sum_{i=0}^{t} A_i.$$

Deoarece sferele de rază t centrate în cuvintele codului $A_{n,k}$ sunt disjuncte, avem

$$q^k S_t \leq q^n.$$

Aici, q^k reprezintă numărul de cuvinte - cod, iar q^n - numărul total de cuvinte din Z_q^n.

Din această relaţie se obţine imediat

$$n - k \geq log_q S_t,$$

cunoscută sub numele de *inegalitatea volumului.*

Ea mai poate fi găsită şi sub forma

$$\frac{k}{n} \leq 1 - \frac{1}{n} log_q S_t.$$

Raportul $\frac{k}{n}$ se numeşte *rata de informaţie* şi dă o măsură a cantităţii de informaţie pe care o poartă un cuvânt - cod. O rată de informaţie mică (mai multe simboluri de control) asigură o securitate mai mare a datelor transmise. În schimb, condiţii practice de eficienţă cer o rată de informaţie cât mai mare (mai multă informaţie pe unitatea de mesaj). Aceasta este una din solicitările contradictorii ale teoriei codurilor.

3.2 Modificări ale codurilor liniare

Adesea este imposibil să se utilizeze un cod bun deoarece el nu satisface anumite restricţii tehnice, cum ar fi lungimea sau rata de informaţie. De aceea este util să se aplice anumite prelucrări asupra codurilor, care să nu afecteze proprietăţile principale de detectare şi corectare a erorilor.

Definiţia 3.1 *Numim extensie a unui (n, k) - cod liniar $A_{n,k} \subseteq Z_q^n$, $(n + 1, k)$ - codul liniar $A^*_{n+1,k}$ obţinut din $A_{n,k}$ prin adăugarea la fiecare cuvânt - cod $a_1 a_2 \ldots a_n$ a unui simbol nou a_{n+1}, cu proprietatea $\sum_{i=1}^{n+1} a_i = 0 \ (mod\ q)$.*

Observaţia 3.1

- *În cazul binar, noul caracter a_{n+1} poartă numele bit de paritate.*

  Dacă H este matricea de control a codului $A_{n,k}$, atunci codul extins $A^*_{n+1,k}$ are matricea de control

$$H^* = \left(\begin{array}{ccccc|c} & & & & & 0 \\ & & H & & & \vdots \\ & & & & & 0 \\ \hline 1 & 1 & \ldots & 1 & 1 & 1 \end{array}\right)$$

  De fapt, ultima linie reprezintă ecuaţia $\sum_{i=1}^{n} x_i = (q-1)x_{n+1}$.

- *Dacă un cod liniar binar $A_{n,k}$ are o distanţă minimă impară d, atunci codul extins are distanţa minimă $d+1$. Într-adevăr, fie $\mathbf{a} = a_1a_2\dots a_n \in A_{n,k}$ cu $w(\mathbf{a}) = d$. Cum d este impar, rezultă $\sum_{i=1}^{n} a_i = 1$ (în Z_2), deci $a_{n+1} = 1$.*

  Cuvântul $\mathbf{a}' = a_1a_2\dots a_n1 \in A^*_{n+1,k}$, $w(\mathbf{a}') = d+1$ şi nu se poate construi un alt cuvânt - cod în $A^*_{n+1,k}$ de pondere mai mică.

Definiţia 3.2 *Fie $A_{n,k} \subseteq Z_q^n$.*

1. *"Relaxarea" lui $A_{n,k}$ este un cod liniar $\overline{A}_{n-1,k}$ obţinut prin ştergerea ultimului simbol din cuvintele lui $A_{n,k}$;*
2. *"Completarea" lui $A_{n,k}$ este un cod definit $A^*_{n,k} = A_{n,k} \cup (A_{n,k} + \mathbf{1})$ (unde $\mathbf{1}$ este cuvântul cu toate elementele 1, iar suma se face modulo q);*
3. *"Expurgarea" lui $A_{n,k}$ este codul $A'_{n,k} = \{\mathbf{a} \in A_{n,k} \mid w(\mathbf{a}) \equiv 0 \ (mod\ 2)\}$.*

Observaţia 3.2

- *Relaxarea este operaţia inversă extensiei.*
- *Prin completarea şi expurgarea codurilor liniare se obţin coduri liniare numai în cazul binar. În celelalte cazuri, noile mulţimi rezultate nu sunt spaţii liniare.*

Propoziţia 3.1 *Prin completarea unui cod liniar binar $A_{n,k}$ se obţine un cod liniar binar $A_{n,k+1}$ cu un număr dublu de cuvinte - cod.*

Demonstraţie: Completarea unui cod binar C înseamnă adăugarea la cuvintele - cod ale lui C a tuturor cuvintelor obţinute prin complementare (schimbarea lui 0 în 1 şi a lui 1 în 0).

Fie $G_{k,n}$ matricea generatoare a codului $A_{n,k}$. Se verifică uşor că matricea

$$G'_{k+1,n} = \begin{pmatrix} & G & \\ & & \\ 1 & 1 \ \dots \ 1 & 1 \end{pmatrix}$$

generează $A_{n,k} \cup (\mathbf{1} + A_{n,k})$. Acest cod are $k+1$ poziţii de informaţie şi lungime n. Fiecare din cele două submulţimi are un număr egal de elemente. q.e.d.

Propoziţia 3.2 *Orice cod liniar binar are sau toate cuvintele - cod de pondere pară, sau numărul cuvintelor - cod de pondere pară este egal cu al celor de pondere impară.*

Demonstraţie: Fie C un cod liniar binar cu un cuvânt $\mathbf{v_1}$ de pondere impară. Să presupunem că $\mathbf{v_1}, \mathbf{v_2}, \dots, \mathbf{v_s}$ sunt toate cuvintele lui C; atunci $C = C + \mathbf{v_1}$.

Pentru orice cuvânt - cod $\mathbf{v_i}$ de pondere pară (impară), $\mathbf{v_1} + \mathbf{v_i}$ are pondere impară (pară). Pentru a justifica această afirmaţie, să presupunem că $w(\mathbf{v_1}) = 2p+1$, $w(\mathbf{v_i}) = 2q$ iar $\mathbf{v_i}$ şi $\mathbf{v_1}$ au 1 pe r poziţii comune. Atunci $w(\mathbf{v_i} + \mathbf{v_1}) = w(\mathbf{v_i}) + w(\mathbf{v_1}) - 2r$ (pentru că $1 + 1 = 0) = 2p + 1 + 2q - 2r = 2s + 1$.

Similar se argumentează şi în cazul când $\mathbf{v_i}$ are pondere impară.

Deci adunarea cu $\mathbf{v_1}$ defineşte o corespondenţă biunivocă între cuvintele - cod de pondere pară şi cele de pondere impară, ceea ce completează demonstraţia. q.e.d.

Corolarul 3.1 *Expurgarea unui cod liniar binar C este tot C sau un cod liniar binar având ca elemente jumătate din elementele lui C.*

Matricea generatoare G_{exp} a lui C_{exp} se poate obţine din matricea G a lui C astfel: dacă toate liniile lui G sunt vectori de pondere pară, atunci cele două coduri coincid.

Altfel, fie $G = [\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_s}, \mathbf{e_{s+1}}, \ldots, \mathbf{e_k}]^T$ în care - fără a micşora generalitatea, putem presupune că primele s au pondere impară, iar celelalte $k - s$ au pondere pară. Atunci $G_{exp} = [\mathbf{0}, \mathbf{e_2} + \mathbf{e_1}, \ldots, \mathbf{e_s} + \mathbf{e_1}, \mathbf{e_{s+1}} \ldots, \mathbf{e_k}]^T$. q.e.d.

Exemplul 3.1 *Să construim codul $A_{4,2}$ peste Z_3 de matrice generatoare*

$$G = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}.$$

Ea codifică cele 9 elemente din Z_3^2 în

$$A_{4,2} = \{0000, 0111, 0222, 1010, 1121, 1202, 2020, 2101, 2212\}.$$

Codul liniar relaxat $A_{3,2} = \{000, 011, 022, 101, 112, 120, 202, 210, 221\}$ este generat de matricea

$$G_{rel} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

Construcţia a fost posibilă deoarece prin eliminarea ultimei coloane, liniile rămase sunt tot liniar independente.

Dacă acest lucru nu este realizabil, se caută k cuvinte - cod în $A_{n,k}$ cu proprietatea că după eliminarea ultimei componente, ele sunt liniar independente. Acestea formează liniile noii matrici generatoare.

Codul completat este

0000	0111	0222	1010	1121	1202	2020	2101	2212
1111	1222	1000	2121	2202	2010	0101	0212	0020

De remarcat că el nu este un spaţiu liniar (nu este închis la adunarea din Z_3).

Codul expurgat are cinci elemente: $\{0000, 1010, 1121, 2020, 2212\}$. Nici acesta nu este cod liniar.

Exemplul 3.2 *Să reluăm matricea generatoare din Exemplul 19.4, dar pentru un cod liniar peste Z_2. Codul generat de G este $A_{4,2} = \{0000, 0111, 1010, 1101\}$.*

Toate codurile modificate sunt în acest caz coduri liniare. Astfel

- *Codul relaxat $A_{rel} = \{000, 011, 101, 110\}$ este generat de aceeaşi matrice G_{rel} din Exemplul 19.4.*
- *Codul completat $A_{com} = \{0000, 0111, 1010, 1101, 1111, 1000, 0101, 0010\}$ este un cod liniar generat de matricea*

$$G_{com} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

- *Codul expurgat $A_{exp} = \{0000, 1010\}$ este un cod liniar generat de matricea*

$$G_{ex} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

3.3 Detectarea şi corectarea simultană a erorilor

Să începem cu un exemplu.

Exemplul 3.3 *Fie* $(7,4)$ *- codul liniar binar având matricea de control*

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

El are distanţa minimă $d = 3$*, deci poate detecta* 2 *erori şi poate corecta o eroare (Teoremele 19.1,3.2). Totuşi, codul nu poate realiza simultan ambele deziderate.*

Mai precis, atunci când codul este utilizat pentru corectare de erori, erorile duble scapă nedetectate. Astfel, dacă se trimite 0000000 *şi se recepţionează* 1010000*, sindromul este* 010*. Tabloul standard conduce la corectarea celui de-al doilea bit, şi decodifică (incorect) în cuvântul - cod* 111000*.*

Uneori însă, se solicită în mod explicit un cod capabil să detecteze şi să corecteze erori în acelaşi timp.

Definiţia 3.3 *Un cod* C *de lungime* n *corectează* t *erori şi detectează* s *erori simultan dacă orice cuvânt - cod* $\mathbf{v}$ *are următoarea proprietate:*

$$\forall\, \mathbf{x} \in Z_q^n \ [\, d(\mathbf{x},\mathbf{v}) \le s \quad \Longrightarrow \quad \forall\, \mathbf{a} \in C \setminus \{\mathbf{v}\},\ d(\mathbf{x},\mathbf{a}) > t\,].$$

În această situaţie, detectarea şi corectarea simultană a erorilor se realizează astfel: la recepţionarea unui cuvânt $\mathbf{x} \in Z_q^n$ se caută cel mai apropiat cuvânt - cod $\mathbf{v}$ (în sensul distanţei Hamming). Dacă $d(\mathbf{x},\mathbf{v}) \le t$ atunci cuvântul se corectează în $\mathbf{v}$; altfel, se anunţă că cel puţin s simboluri sunt modificate.

Justificarea acestui procedeu rezultă imediat din definiţie.

Teorema 3.7 *Un cod liniar corectează* t *erori şi detectează* s *erori simultan dacă şi numai dacă*

$$d \ge t + s + 1.$$

Demonstraţie: **A**: Să presupunem $d \ge t + s + 1$.
Fie $\mathbf{v}$ un cuvânt - cod şi $\mathbf{x} \in Z_q^n$ cu $d(\mathbf{v},\mathbf{x}) \le s$. Pentru orice cuvânt - cod $\mathbf{v}'$ ($\mathbf{v}'\mathbf{v}$) avem $d(\mathbf{v},\mathbf{v}') \ge d \ge t + s + 1$.

Folosind inegalitatea triunghiului,

$$d(\mathbf{v},\mathbf{x}) + d(\mathbf{x},\mathbf{v}') \ge d(\mathbf{v},\mathbf{v}') \ge t + s + 1,$$

se deduce

$$d(\mathbf{x},\mathbf{v}') \ge t + s + 1 - d(\mathbf{v},\mathbf{x}) \ge t + s + 1 - s = t + 1.$$

Deci, condiţia din Definiţia 20.3 este îndeplinită.

B: Să presupunem prin absurd $d < t + s + 1$.

Fie $\mathbf{v},\mathbf{v}'$ două cuvinte - cod cu $d(\mathbf{v},\mathbf{v}') = d \le t+s$. Construim cuvântul $\mathbf{x}$ din $\mathbf{v}$ în felul următor: dacă $d \le s$ atunci $\mathbf{x} = \mathbf{v}'$; altfel se înlocuiesc primele s simboluri în care acesta diferă de $\mathbf{v}'$, cu valorile lor din $\mathbf{v}'$. Atunci $d(\mathbf{v},\mathbf{x}) = s$ şi $d(\mathbf{v}',\mathbf{x}) = d - s \le t + s - s = t$, ceea ce contrazice condiţia din Definiţia 20.3. q.e.d.

Exemplul 3.4 *Să considerăm* $(8,4)$ *- codul liniar binar generat de matricea*

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

El are distanţa minimă $d = 4$*, deci - conform Teoremei 3.2 poate corecta maxim o eroare, iar conform Teoremei 3.7 poate corecta o eroare şi detecta simultan două erori.*

Astfel recepţionarea cuvântului 11110010 *conduce la corectarea sa în* 10110010 *(deoarece* $d(11110010, 10110010) = 1$ *şi* 10110010 *este cuvânt - cod).*

În schimb recepţionarea cuvântului 00001111 *anunţă că au apărut cel puţin două erori. În această situaţie, nu mai puţin de patru cuvinte - cod* (00010111, 00101011, 01001101, 10001110) *sunt situate la distanţa* 2 *de cuvântul primit.*

Exemplul 3.5 *Codul binar cu repetiţie de lungime* 7 *(care are* 2 *elemente:* 0000000 *şi* 1111111*) poate realiza una din condiţiile:*

- *Corectează* 3 *erori;*
- *Detectează* 6 *erori;*
- *Corectează* 2 *erori şi detectează* 4 *erori simultan.*

3.4 Probabilitatea nedetectării erorilor

Să ne punem următoarea problemă: care este probabilitatea ca la transmiterea unui cuvânt - cod $\mathbf{a}$ să fie recepţionat alt cuvânt - cod $\mathbf{b}$ ($\mathbf{b} \neq \mathbf{a}$). Altfel spus, care este probabilitatea ca o eroare să scape nedetectată ?

Notând cu $\mathbf{e} = \mathbf{b} - \mathbf{a}$, o eroare este nedetectată dacă şi numai dacă $\mathbf{e}$ este un cuvânt - cod nenul.

Vom considera un canal de transmisie *binar simetric*, adică un canal în care singurele simboluri transmise sunt 0 şi 1, iar probabilitatea p ($0 \leq p \leq 1$) ca la transmiterea lui 0 să fie recepţionat 1 este egală cu probabilitatea ca la transmiterea lui 1 să se recepţioneze 0.

Într-un astfel de canal, dacă $w(\mathbf{e}) = i$ (adică au fost perturbate la transmisie i caractere), probabilitatea de apariţie a erorii - tip $\mathbf{e}$ este $p^i q^{n-i}$, unde $q = 1 - p$.

Notând cu A_i numărul cuvintelor - cod cu ponderea i, probabilitatea P_{ned} a unei erori nedetectabile este suma probabilităţilor $p^i q^{n-i}$, fiecare termen apărând de A_i ori pentru $i = 1, 2, \ldots, n$.

Formal,

$$P_{ned} = \sum_{i=1}^{n} A_i p^i q^{n-i}$$

Cum $A_1 = A_2 = \ldots = A_{d-1} = 0$, suma se reduce la $P_{ned} = \sum_{i=d}^{n} A_i p^i q^{n-i}$.

Exemplul 3.6 *Să considerăm un* $(7,4)$ *- cod liniar binar definit în Exemplul 19.6. Cele* $2^4 = 16$ *cuvinte - cod ale sale sunt: un cuvânt de pondere* 0*, câte şapte cuvinte de pondere* 3 *şi respectiv* 4 *şi un cuvânt de pondere* 7*. Atunci*

$$P_{ned} = 7p^3q^4 + 7p^4q^3 + p^7$$

Dacă folosim acest cod într-un canal binar simetric cu eroare de probabilitate $p = 0.01$*, avem*

$$P_{ned} = 7(0.01)^3(0.99)^4 + 7(0.01)^4(0.99)^3 + (0.01)^7 \approx 7 \times 10^{-6}$$

deci – în medie – apar cam şapte erori nedetectabile la un milion de cuvinte transmise.

Definiţia 3.4 *Polinomul de variabilă* $x \in [0,1]$ *definit*

$$P(x) = \sum_{i=0}^{n} A_i x^i$$

unde A_i *este numărul de cuvinte de pondere* i *din codul liniar* $A_{n,k}$*, se numeşte "numărătorul de ponderi" al codului* $A_{n,k}$*.*

Exemplul 3.7 *Codul definit în Exemplul 19.5 are numărătorul de ponderi*

$$P(x) = 1 + x^2 + 2x^3$$

Propoziţia 3.3 *Fie* $A_{n,k}$ *un cod liniar binar cu numărător de ponderi* $P(x)$*. Probabilitatea apariţiei unei erori nedetectabile la folosirea codului* $A_{n,k}$ *într-un canal binar simetric este*

$$P_{ned} = q^n \left[P\left(\frac{p}{q}\right) - 1 \right].$$

Demonstraţie: Relaţia de definiţie a lui P_{ned} se poate rescrie

$$q^n \sum_{i=1}^{n} A_i p^i q^{-i} = q^n \sum_{i=1}^{n} A_i \left(\frac{p}{q}\right)^i.$$

Deoarece $A_0 = 1$ (singurul cuvânt - cod de pondere 0 este cuvântul - cod $\mathbf{0}$), expresia devine în continuare

$$P_{ned} = q^n \left[\sum_{i=0}^{n} A_i \left(\frac{p}{q}\right)^i - 1 \right] = q^n \left[P\left(\frac{p}{q}\right) - 1 \right]. \qquad \text{q.e.d.}$$

3.5 Identitatea MacWilliams

În acest paragraf vom arăta un rezultat care face posibilă determinarea numără-torului de ponderi $P_{C^\perp}(x)$ al dualului $C^\perp$ unui cod liniar C, direct din numărătorul de ponderi $P_C(x)$ al codului C.

Propoziţia 3.4 *Fie* $\mathbf{x} \in Z_2^n$ *şi* $A_{n,k}$ *un* (n,k) *- cod liniar binar. Atunci are loc egalitatea*

$$\frac{1}{2^k} \sum_{\mathbf{v} \in A} (-1)^{\mathbf{vx}} = \begin{cases} 1 & \text{dacă} \quad \mathbf{x} \in A_{n,k}^\perp \\ 0 & \text{altfel} \end{cases}$$

Demonstraţie: Dacă $\mathbf{x} \in A_{n,k}^{\perp}$ atunci evident, $(-1)^{\mathbf{vx}} = (-1)^0 = 1$ şi suma este egală cu numărul de cuvinte - cod din $A_{n,k}$, care este 2^k.

Să presupunem acum că $\mathbf{x} \notin A_{n,k}^{\perp}$; deci există un cuvânt - cod $\mathbf{v_0} \in A$ cu $\mathbf{v_0 x} = 1$.

Vom arăta că în acest caz, numărul cuvintelor - cod ortogonale pe $\mathbf{x}$ este egal cu cel al cuvintelor - cod ne-ortogonale pe $\mathbf{x}$ (şi deci suma din formulă este 0).

Fie $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_r}$ toate cuvintele - cod ortogonale pe $\mathbf{x}$. Facem afirmaţia că atunci $\mathbf{v_1} + \mathbf{v_0},\ \mathbf{v_2} + \mathbf{v_0},\ \ldots,\ \mathbf{v_r} + \mathbf{v_0}$ sunt toate cuvintele - cod ne-ortogonale pe $\mathbf{x}$.

Într-adevăr:

1. $\forall\, i\ (1 \leq i \leq r) \quad (\mathbf{v_i} + \mathbf{v_0})\mathbf{x} = \mathbf{v_i x} + \mathbf{v_0 x} = 0 + 1 = 1$;
2. Dacă $\mathbf{v} \in A_{n,k}$ verifică relaţia $\mathbf{vx} = 1$, atunci $\mathbf{v} - \mathbf{v_0}$ este un cuvânt cod - ortogonal pe $\mathbf{x}$, deci $\mathbf{v} - \mathbf{v_0} = \mathbf{v_i}$ pentru un anumit i. q.e.d.

Teorema 3.8 *(Identitatea MacWilliams) Pentru orice (n, k) - cod liniar binar C are loc relaţia*

$$P_{C^\perp}(x) = \frac{(1+x)^n}{2^k} P_C\left(\frac{1-x}{1+x}\right).$$

Demonstraţie: Să rescriem numărătorul de ponderi sub o formă puţin diferită:

$$B(x, y) = \sum_{i=0}^{n} A_i x^i y^{n-i}.$$

Evident, deoarece $P(x) = B(x, 1)$ şi $B(x, y) = y^n P\left(\frac{x}{y}\right)$, cele două expresii sunt echivalente.

În notaţia cu polinomul B, identitatea MacWilliams se scrie

$$B_{C^\perp}(x) = \frac{1}{2^k} B_C(y - x, y + x).$$

Cu ajutorul ponderii cuvintelor - cod, numărătorul de ponderi are forma

$$B_C(x, y) = \sum_{i=0}^{n} A_i x^i y^{n-i} = \sum_{\mathbf{a} \in C} x^{w(\mathbf{a})} y^{n - w(\mathbf{a})}$$

unde $\mathbf{a} = (a_1, a_2, \ldots, a_n)$.

Prelucrând membrul drept al identităţii MacWilliams, avem:

$$B_C(y - x, y + x) = \sum_{\mathbf{a} \in C} (y - x)^{w(\mathbf{a})} (y + x)^{n - w(\mathbf{a})} = \sum_{\mathbf{a} \in C} \prod_{i=1}^{n} \left[y + (-1)^{a_i} x\right].$$

În mod analog, membrul stâng se scrie:

$$B_{C^\perp}(x, y) = \sum_{\mathbf{a} \in C^\perp} x^{w(\mathbf{a})} y^{n - w(\mathbf{a})}.$$

Folosind Propoziţia 3.4, el se poate reformula:

$$B_{C^\perp}(x,y) = \sum_{\mathbf{a}\in Z_2^n} \left[\frac{1}{2^k} \sum_{\mathbf{v}\in C} (-1)^{\mathbf{av}} \right] x^{w(\mathbf{a})} y^{n-w(\mathbf{a})}.$$

Expresia din paranteze este 0 pentru toate cuvintele $\mathbf{a}$ care nu sunt în $C^\perp$. Deci

$$B_{C^\perp}(x,y) = \frac{1}{2^k} \sum_{\mathbf{v}\in C} \sum_{\mathbf{a}\in Z_2^n} (-1)^{\mathbf{va}} x^{w(\mathbf{a})} y^{n-w(\mathbf{a})}.$$

Suma interioară se face după toate secvenţele binare $\mathbf{a}$ de lungime n. Vom ordona această sumă după ponderile lui $\mathbf{a}$: pentru $\mathbf{a} = 0$ sumandul este y^n; pentru cuvintele $\mathbf{a}$ de pondere 1 avem: $[(-1)^{a_1} + (-1)^{a_2} + \ldots + (-1)^{a_n}]xy^{n-1}$ etc.
În final, pentru ponderea n avem $[(-1)^{a_1} + \ldots + (-1)^{a_n}]x^n$.

Suma tuturor acestor sumanzi $\sum_{k=0}^{n} [(-1)^{a_{i_1}} + \ldots + (-1)^{a_{i_k}}]x^k y^{n-k}$ este egală cu

$$[y + (-1)^{a_1}x][y + (-1)^{a_2}x] \ldots [y + (-1)^{a_k}x] = \prod_{i=1}^{n} [y + (-1)^{a_i}x].$$

Deci $B_{C^\perp}(x,y) = \frac{1}{2^k} \sum_{\mathbf{a}\in C} \prod_{i=1}^{n} [y + (-1)^{a_i}x] = \frac{1}{2^k} B_C(y-x, y+x)$. q.e.d.

Exemplul 3.8 *Să considerăm codul $A_{4,2}$ din Exemplul 19.5 al cărui numărător de ponderi a fost dat în Exemplul 3.7. Pentru codul dual, numărătorul de ponderi este*

$$P_{A_{4,2}^\perp}(x) = \frac{(1+x)^4}{2^2} P_{A_{4,2}}\left(\frac{1-x}{1+x}\right) = \frac{(1+x)^4}{4}\left[1 + \left(\frac{1-x}{1+x}\right)^2 + 2\left(\frac{1-x}{1+x}\right)^3\right] =$$

$$= \frac{(1+x)^4 + (1+x)^2(1-x)^2 + 2(1+x)(1-x)^3}{4} = \frac{4 + 4x^2 + 8x^3}{4} = 1 + x^2 + 2x^3.$$

Deci cele două coduri au acelaşi numărător de ponderi.

Aceasta nu înseamnă însă că cele două coduri coincid (şi deci codul ar fi auto - dual); a avea acelaşi numărător de ponderi este doar o condiţie necesară, nu şi suficientă pentru ca un cod să coincidă cu dualul său.

Ca o verificare, $A_{4,2} = \{0000, 0111, 1010, 1101\}$, iar $A_{4,2}^\perp = \{0000, 0101, 1011, 1110\}$.

Exemplul 3.9 *Fie codul liniar cu repetiţie de lungime pară n $C = \{00\ldots0, 11\ldots1\}$; numărătorul lui de ponderi este $P_C(x) = 1 + x^n$. Codul dual are numărătorul de ponderi*

$$P_{C^\perp}(x) = \frac{(1+x)^n}{2}\left[1 + \left(\frac{1-x}{1+x}\right)^n\right] = 1 + C_n^2 x^2 + C_n^4 x^4 + \ldots + x^n$$

Rezultă din această formă că dualul codului liniar cu repetiţie este codul liniar binar al tuturor cuvintelor de lungime n şi pondere pară.

3.6 Exerciţii

3.1 *În Z_2^n notăm cu $\overline{\mathbf{x}}$ cuvântul obţinut din $\mathbf{x}$ prin permutarea caracterelor 0 şi 1 între ele. Să se arate că pentru orice $\mathbf{a}, \mathbf{b} \in Z_2^n$:*

1. $\overline{\mathbf{a}} + \overline{\mathbf{b}} = \mathbf{a} + \mathbf{b}$;
2. $\mathbf{a} + \overline{\mathbf{b}} = \overline{\mathbf{a}} + \mathbf{b} = \overline{\mathbf{a} + \mathbf{b}}$;
3. $d(\mathbf{a}, \overline{\mathbf{b}}) = d(\overline{\mathbf{a}}, \mathbf{b}) = w(\overline{\mathbf{a} + \mathbf{b}})$.

3.2 *Descrieţi codurile modificate obţinute din codul liniar binar cu matricea genera-toare*

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

3.3 *Aceeaşi problemă pentru codul peste Z_3 definit prin*

$$G = \begin{pmatrix} 1 & 0 & 0 & 2 & 2 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

3.4 *Fie $A_{n,k}$ un (n, k) - cod liniar binar şi $A'_{n,k-1}$ - codul obţinut din A prin expurgare. Ce relaţie există între matricile de control ale celor două coduri ?*

3.5 *Arătaţi cum poate codul binar cu repetiţie de lungime 7 să corecteze două erori şi să detecteze 4 erori simultan. Câte erori poate detecta dacă corectează o eroare ?*

3.6 *Fie un $(15, 4)$ - cod liniar binar în care fiecare coloană i din matricea generatoare este scrierea binară a lui i sub forma unui vector cu 4 componente. Să se determine distanţa minimă, numărătorul de ponderi şi numărătorul de ponderi al codului dual.*

3.7 *Fie C un $(2k + 1, k)$ - cod binar astfel ca $C^\perp \subset C$. Descrieţi $C^\perp \setminus C$.*

Capitolul 4

Clase de coduri liniare

4.1 Coduri Hamming

Fie H matricea de control a unui cod liniar binar. Dacă se transmite un cuvânt - cod $\mathbf{a}$ şi se recepţionează $\mathbf{a}+\mathbf{e}$ (deci cu eroarea - tip $\mathbf{e}$), atunci sindromul este $\mathbf{e}H^T$. Acest sindrom este egal cu suma coloanelor lui H care corespund poziţiilor afectate de erori (Prelegerea II, Teorema 2.3).

În particular, o eroare care apare pe o poziţie corespunzătoare unei coloane nule din H nu influenţează sindromul. Deci, o astfel de eroare nu este detectată.

Dacă H are două coloane identice şi se întâmplă ca pe poziţiile corespunzătoare lor să apară simultan erori, acestea se anulează reciproc în calculul sindromului - şi deci nu pot fi detectate.

Pe de-altă parte, dacă toate coloanele lui H sunt distincte şi nenule, o eroare singulară pe poziţia s va face ca sindromul să fie egal cu coloana numărul s din H. În acest caz, erorile singulare pot fi detectate şi corectate foarte uşor.

Pe baza acestor observaţii am demonstrat teorema:

Teorema 4.1 *Un cod liniar binar poate corecta o eroare dacă şi numai dacă matricea sa de control are toate coloanele nenule şi distincte.*

Pentru a se putea corecta toate erorile simple, trebuie să existe sindromuri distincte pentru fiecare eroare - tip; deci, conform marginii Hamming (Prelegerea III, Teorema 3.5),

$$2^{n-k} \geq n+1.$$

Pe baza acestor considerente se defineşte codul Hamming binar:

Definiţia 4.1 *Codul liniar binar în care coloanele matricii H sunt reprezentarea binară a numerelor $1, 2, \ldots, 2^r-1$ este numit cod Hamming binar.*

Deci, pentru orice număr natural r ($r \geq 2$) se poate construi un (n,k) - cod liniar binar în care $n = 2^r - 1,\ k = 2^r - r - 1$.

De remarcat că definiţia nu determină pentru fiecare r în mod unic matricea de control a codului Hamming. De obicei se consideră acea matrice H în care coloana i reprezintă scrierea în binar a numărului i. Deoarece codul este echivalent cu un cod sistematic (există coloane pentru $2^0, 2^1, \ldots, 2^{r-1}$), toate celelalte reprezentări au aceleaşi proprietăţi.

Exemplul 4.1 *Pentru* $r = 3$ *avem codul Hamming de lungime* $n = 2^3 - 1 = 7$ *cu* $k = 2^3 - 3 - 1 = 4$ *simboluri de informaţie şi* 3 *simboluri de control. Matricea de control este:*

$$H_{3,7} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

De aici rezultă că el este determinat de soluţiile sistemului liniar:

$$\begin{cases} x_4 + x_5 + x_6 + x_7 = 0 \\ x_2 + x_3 + x_6 + x_7 = 0 \\ x_1 + x_3 + x_5 + x_7 = 0 \end{cases}$$

Să determinăm matricea generatoare a acestui cod. Pentru aceasta, construim întâi codul sistematic echivalent, permutând coloanele pentru a aduce matricea de control la forma eşalonată canonic:

$$H^*_{3,7} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

De aici se obţine matricea generatoare eşalonată canonic:

$$G^*_{4,7} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Aplicând permutarea inversă asupra coloanelor, se obţine matricea generatoare a $(7, 4)$ *- codului Hamming binar:*

$$G_{4,7} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Coloanele corespunzătoare matricii unitate $(3, 5, 6, 7)$*, reprezintă poziţiile simbolurilor de informaţie în fiecare cuvânt - cod.*

Deci simbolurile x_1, x_2, x_4 *sunt simboluri de control. Sistemul de sus poate fi rearanjat pentru a permite calculul simbolurilor de control plecând de la simbolurile de informaţie:*

$$\begin{cases} x_1 = x_3 + x_5 + x_7 \\ x_2 = x_3 + x_6 + x_7 \\ x_4 = x_5 + x_6 + x_7 \end{cases}$$

Toate cuvintele codului sunt:

Informaţie	*Cuvânt cod*	*Informaţie*	*Cuvânt cod*
0000	0000000	0110	0110011
1000	1000011	0101	0101010
0100	0100101	0011	0011001
0010	0010110	1110	1110000
0001	0001111	1101	1101001
1100	1100110	1011	1011010
1010	1010101	0111	0111100
1001	1001100	1111	1111111

Teorema 4.2 *Orice cod Hamming binar are distanţa minimă* 3.

Demonstraţie: Evident, orice două coloane din matricea de control sunt liniar independente. În plus, se pot găsi trei coloane (de exemplu primele trei) a căror sumă să fie **0**. Conform Teoremei 2.4 (Prelegerea II), distanţa minimă a codului este 3. q.e.d.

Deci orice cod Hamming binar poate corecta o eroare sau poate detecta două erori. El nu poate realiza acest lucru simultan (nu verifică condiţia $d \geq s + t + 1$ din Teorema 3.7, Prelegerea III).

Decodificarea se realizează foarte simplu, conform următorului algoritm:

Algoritm A:
Fie **a** vectorul recepţionat.

1. Se calculează sindromul $\mathbf{s} = \mathbf{a}H^T$.
2. Dacă $\mathbf{s} = \mathbf{0}$, nu a apărut nici o eroare (sau eroarea este nedetectabilă), deci $\mathbf{v} = \mathbf{a}$, STOP.
3. Altfel, eroarea este pe poziţia i, unde i este numărul a cărui reprezentare în binar este sindromul **s**.

   Decodificarea este $\mathbf{v} = \mathbf{a} + \mathbf{e_i}$, unde $\mathbf{e_i}$ este vectorul care are 1 pe poziţia i şi 0 în rest.

Exemplul 4.2 *Să considerăm din nou* $(7, 4)$ *- codul Hamming binar din Exemplul 19.4 şi să presupunem că s-a recepţionat cuvântul* $\mathbf{x} = 0011101$. *Calculul sindromului conduce la valoarea*

$$\mathbf{x}H^T = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

care este scrierea în binar a numărului 5. *Deci a intervenit o eroare simplă pe poziţia a cincea. Corectăm această poziţie – schimbând* 1 *cu* 0 *şi se obţine cuvântul - cod* 0011001, *care pe poziţiile* $3, 5, 6, 7$ *conţine mesajul de informaţie:* 1001.

Codul Hamming binar poate fi îmbunătăt it prin extensie. Această operaţie conduce la un $(2^r, 2^r - r - 1)$ - cod liniar, cu toate cuvintele - cod de pondere pară.

Exemplul 4.3 *Prin extensia* $(7, 4)$ *- codului Hamming se obţine codul cu matricea de control*

$$H^* = \left(\begin{array}{ccccccc|c} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right).$$

De remarcat că H^* *are rangul* 4*; în plus, toate liniile acestei matrici sunt cuvinte - cod în codul Hamming binar extins. Deci* H^* *poate fi considerată matrice generatoare a acestui cod. Rezultă că* $(8, 4)$ *- codul Hamming binar extins este auto - dual.*

Un cod Hamming binar extins este soluţia unui sistem liniar de $n-k+1$ ecuaţii, ecuaţia suplimentară $\sum_{i=1}^{n+1} x_i = 0$ fiind *ecuaţia de control a parităţii*.

Propoziţia 4.1 *Un cod Hamming binar extins are* $d = 4$.

Demonstraţie: Fie $\mathbf{a} = a_1 a_2 \dots a_n$ un cuvânt - cod de pondere $d = 3$ din codul Hamming binar. Trecând la codul extins, cuvântul $\mathbf{a}' = a_1 a_2 \dots a_n a_{n+1}$ verifică relaţia suplimentară $\sum_{i=1}^{n+1} a_i = 0$ (reamintim, sumele se fac modulo 2).

Cum $\sum_{i=1}^{n} a_i = 1 \quad (w(\mathbf{a}) = 3)$, rezultă $a_{n+1} = 1$.

Noul cuvânt are evident pondere minimă, şi aceasta este $3 + 1 = 4$. q.e.d.

Codurile Hamming binare extinse corectează o eroare simplă şi detectează 2 erori simultan. Algoritmul prezentat este bazat pe verificarea celor $n-k+1$ ecuaţii de control:

Algoritm B:

1. Dacă nu sunt verificate ecuaţia de control a parităţii şi cel puţin una din primele $n-k$ ecuaţii, înseamnă că a apărut o eroare simplă, care se corectează cu Algoritmul A;
2. Dacă ecuaţia de control a parităţii este verificată dar cel puţin una din primele $n-k$ ecuaţii de control nu se verifică, s-a detectat o eroare dublă;
3. În celelalte situaţii nu au apărut erori (sau eroarea este nedetectabilă).

Faptul că se poate lua totdeauna o decizie, se bazează pe următorul rezultat:

Teorema 4.3 *Codul Hamming binar este perfect.*

Demonstraţie: Reamintim (Prelegerea II, Definiţia 2.8) că un cod binar C este perfect dacă

$$Z_q^n = \bigcup_{\mathbf{x} \in C} S_t(\mathbf{x}), \quad \text{unde} \quad t = \left[\frac{d-1}{2} \right].$$

Scriind această relaţie în funcţie de numărul de elemente din fiecare sferă şi ţinând cont că toate sferele conţin un număr egal de elemente, avem

$$q^k \left[1 + C_n^1 (q-1) + \ldots + C_n^t (q-1)^t \right] = q^n.$$

Pentru cazul codurilor Hamming, $q = 2,\ n = 2^r - 1,\ k = 2^r - r - 1,\ t = 1$, deci totul revine la verificarea egalităţii $2^k(1+n) = 2^n$. q.e.d.

De remarcat că rata de informaţie a codurilor Hamming

$$R = \frac{k}{n} = 1 - \frac{r}{2^r - 1}$$

creşte rapid spre 1.

Evident însă că odată cu această creştere scade protecţia faţă de erori.

4.1.1 Coduri Hamming nebinare

Definiţia 4.2 *Fie q un număr prim, r $(r \geq 2)$ un număr întreg şi $n = \dfrac{q^r - 1}{q - 1}$. Se numeşte cod Hamming nebinar un $(n, n - r)$ - cod liniar peste Z_q, în care matricea de control are orice pereche de două coloane liniar independente (nici o coloană nu este multiplu scalar al altei coloane).*

Mulţimea coloanelor unui astfel de cod formează o mulţime maximală de vectori liniar independenţi doi câte doi.

Exemplul 4.4 *Să considerăm $q = 3, r = 2$. Atunci $n = \dfrac{3^2 - 1}{3 - 1} = 4$. Un $(4, 2)$ - cod Hamming ternar poate fi definit prin de matricea de control*

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 \end{pmatrix}$$

Decodificarea se poate face folosind tabela de sindromuri, în care s-au luat ca reprezentanţi toate combinaţiile posibile de o eroare:

Sindrom	*Reprezentant*	*Sindrom*	*Reprezentant*
01	1000	12	0001
02	2000	20	0200
10	0100	21	0002
11	0010	22	0020

Dacă se recepţionează de exemplu 2222*, calculul sindromului dă* $\mathbf{s} = 02$*. Reprezentantul este* 2000*. Se calculează* $2222 - 2000 = 2222 + 1000 = 0222$ *deci cuvântul - cod transmis a fost* 2021*.*

Cum matricea generatoare a acestui cod este $G = \begin{pmatrix} 2 & 2 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{pmatrix}$ *ultimele două caractere formează mesajul de informaţie; deci s-a codificat mesajul* 22*.*

Despre codurile Hamming nebinare se pot stabili următoarele rezultate:

- Deoarece pentru q şi r fixaţi, codurile Hamming corespunzătoare sunt echivalente, se poate alege o anumită matrice de control. Uzual se foloseşte matricea în care se scriu pe coloane toate elementele din Z_q^r, care satisfac cerinţa ca primul element nenul (de sus în jos) să fie 1.

- Codurile Hamming nebinare au $d = 3$ (evident, din construcţia matricii de control de mai sus). Deci ele pot corecta o eroare.

- Proprietatea de a fi coduri perfecte se păstrează. Într-adevăr, deoarece un cod Hamming conţine $q^k = q^{n-r}$ cuvinte - cod iar $n = \dfrac{q^r - 1}{q - 1}$, egalitatea stabilită în demonstraţia Teoremei 4.3 se scrie $q^{n-r}[1 + n(q - 1)] = q^n$, care se verifică imediat.

Exemplul 4.5 *Să considerăm* $(13, 10)$ *- codul Hamming ternar. Acest cod are o aplicaţie interesantă în problema Pronosportului. După cum se ştie, un buletin Pronosport conţine rezultatele (notate cu* $1, 2, X$*) a* 13 *meciuri. Pentru a avea sigur* 13 *rezultate exacte trebuie completate* 3^{13} *buletine.*

Câte buletine sunt însă necesare pentru a avea sigur 12 *rezultate exacte ?*

La prima vedere s-ar părea că 3^{12}*. Completând însă buletinele cu elementele codului Hamming ternar* $(13, 10)$ *(cu* 0 *în loc de* X*) – care sunt în număr de* 3^{10}*, se atinge scopul dorit. Într-adevăr, acesta fiind un cod perfect corector de o eroare, orice element din* Z_3^{13} *diferă prin cel mult o poziţie de un cuvânt - cod.*

Astfel, numărul buletinelor se reduce de nouă ori.

4.2 Codul Golay

Al doilea cod liniar prezentat are capacitatea de corecţie pentru maxim 3 erori.

Vom construi întâi varianta extinsă a codului, deoarece algoritmul de decodificare este mai simplu şi uşor de aplicat ulterior la codul Golay normal.

4.2.1 Codul Golay binar extins

Acest cod a fost folosit de programul spaţial *Voyager* la începutul anilor ′80 pentru transmiterea fotografiilor planetelor Jupiter şi Saturn.

Să considerăm matricea 12×12 din Figura 4.1:

Tabelul 4.1:

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

Fie matricea 12×24 $G = (I_{12}|B)$. Codul liniar binar generat de G se numeşte *codul Golay extins* şi va fi notat C_{24}.

Observaţia 4.1

- *Matricea B este mai uşor de construit decât pare la prima vedere. Astfel, eliminând ultima linie şi coloană, matricea rămasă – să spunem B_1 – este generată ciclic (spre stânga) de cuvântul binar* 11011100010. *Deci* $B = \begin{pmatrix} B_1 & \mathbf{1}^T \\ \mathbf{1} & 0 \end{pmatrix}$,

  unde $\mathbf{1} = 11111111111$. *Evident, B este simetrică ($B^T = B$).*

- *C_{24} are $n = 24$, $k = 12$ şi $2^{12} = 4096$ cuvinte - cod.*

- *Conform Teoremei 2.2, o matrice de control a codului este $H = (B|I_{12})$.*

Teorema 4.4 *$H = (I_{12}|B)$ este de asemenea matrice de control pentru C_{24}.*

Demonstraţie: Liniile din B au pondere impară (7 sau 11); deci produsul (scalar) al unei linii cu ea însăs i este 1. O verificare simplă arată că produsul primei linii cu oricare altă linie din B este 0. Structura ciclică a lui B_1 asigură că atunci produsul scalar al oricăror două linii este 0.

În concluzie, $BB^T = I_{12}$. Dar $B^T = B$, aşa că putem scrie:

$$GH^T = (I|B)\begin{pmatrix} I \\ B \end{pmatrix} = I^2 + B^2 = I + BB^T = I + I = \mathbf{0}.$$

Vom folosi ambele matrici de control pentru decodificarea codului C_{24}. q.e.d.

Corolarul 4.1
A. *C_{24} admite ca matrice generatoare şi pe $G = (B|I_{12})$.*
B. *Codul Golay extins este auto - dual ($C_{24} = C_{24}^{\perp}$).*

Demonstraţie: Se verifică imediat. q.e.d.

Teorema 4.5 *C_{24} are distanţa minimă $d = 8$.*

Demonstraţie: Vom demonstra afirmaţia în trei paşi.

1. *Ponderea cuvintelor din C_{24} este multiplu de* 4.

   Să observăm că liniile lui G au pondere 8 sau 12. Fie $\mathbf{v} \in C_{24}$, scris ca sumă de două linii din G : $\mathbf{v} = \mathbf{r_i} + \mathbf{r_j}$.

   Cum B are liniile ortogonale, rezultă că şi liniile lui G sunt ortogonale. Deci $\mathbf{r_i}$ şi $\mathbf{r_j}$ au un număr par (să zicem $2x$) de elemente 1 în comun. Atunci $w(\mathbf{v}) = w(\mathbf{r_i}) + w(\mathbf{r_j}) - 2(2x)$, care este multiplu de 4.

   Fie acum $\mathbf{v} \in C_{24}$ reprezentat ca sumă de trei linii din G : $\mathbf{v} = \mathbf{r_i} + \mathbf{r_j} + \mathbf{r_s}$. Notăm $\mathbf{v_1} = \mathbf{r_i} + \mathbf{r_j}$. Deoarece $\mathbf{v_1}, \mathbf{r_s} \in C_{24}$, iar C_{24} este auto - dual (deci oricare două cuvinte - cod sunt ortogonale), rezultă că $\mathbf{v_1}$ şi $\mathbf{r_s}$ au în comun un număr par (să zicem $2y$) de elemente 1.
   Deci $w(\mathbf{v}) = w(\mathbf{v_1}) + w(\mathbf{r_s}) - 2(2y)$ care este multiplu de 4.

   Folosind acum un procedeu de inducţie, cum orice cuvânt - cod este combinaţie liniară de linii din G, ponderea sa va fi multiplu de 4.

2. Primele 11 linii din G sunt cuvinte - cod de pondere 8, deci distanţa minimă a codului C_{24} este 4 sau 8.

3. *C_{24} nu are cuvinte de pondere* 4.

   Să presupunem că există $\mathbf{v} \in C_{24}$ cu $w(\mathbf{v}) = 4$. Cum pentru C_{24} am considerat două matrici generatoare, vor exista două mesaje de informaţie $u_1, u_2 \in Z_2^{12}$ cu $\mathbf{v} = \mathbf{u_1}(I|B)$, $\mathbf{v} = \mathbf{u_2}(B|I)$. Una din cele două jumătăţi din $\mathbf{v}$ are cel puţin doi de 1; deci $w(\mathbf{u_1}) \leq 2$ sau $w(\mathbf{u_2}) \leq 2$.

   Pe de-altă parte, o linie din B are pondere minim 7, deci $\mathbf{v}$ nu este o linie a matricii B. Dacă $\mathbf{v}$ este suma a două linii din B, ea nu poate avea o pondere mai mică de 4; deci $w(\mathbf{v}) = w(\mathbf{u_i}) + w(\mathbf{u_i}B) \geq 2 + 4 > 4$, contradicţie. q.e.d.

4.2.2 Decodificarea codului Golay extins

Conform Teoremei 4.5, un cod Golay extins poate corecta orice combinaţie de maxim 3 erori independente.

În această secţiune vom nota cu $\mathbf{a} \in Z_{24}$ cuvântul recepţionat, cu $\mathbf{v}$ cuvântul - cod cel mai apropiat, şi cu $\mathbf{e}$ eroarea - tip ($\mathbf{v} = \mathbf{a} + \mathbf{e}$).

Deoarece capacitatea de corecţie este de 3 erori, vom considera $w(\mathbf{e}) \leq 3$.

Pentru orice cuvânt $\mathbf{a} \in Z_2^{24}$, vom separa cu o virgulă prima jumătate a cuvântului de cea de-a doua: $\mathbf{a} = [\mathbf{a_1}, \mathbf{a_2}]$ cu $\mathbf{a_1}, \mathbf{a_2} \in Z_{12}^2$. Similar, eroarea - tip va fi notată $\mathbf{e} = [\mathbf{e_1}, \mathbf{e_2}]$. Evident, condiţia $w(\mathbf{e}) \leq 3$ implică $w(\mathbf{e_1}) \leq 1$ sau $w(\mathbf{e_2}) \leq 1$.

Folosind cele două matrici de control posibile, se pot defini două sindromuri pentru $\mathbf{a}$:

$$\mathbf{s_1} = \mathbf{e}\begin{pmatrix} I_{12} \\ B \end{pmatrix} = [\mathbf{e_1}, \mathbf{e_2}]\begin{pmatrix} I_{12} \\ B \end{pmatrix} = \mathbf{e_1} + \mathbf{e_2}B,$$

$$\mathbf{s_2} = \mathbf{e}\begin{pmatrix} B \\ I_{12} \end{pmatrix} = [\mathbf{e_1}, \mathbf{e_2}]\begin{pmatrix} B \\ I_{12} \end{pmatrix} = \mathbf{e_1}B + \mathbf{e_2}.$$

De aici rezultă următoarea observaţie: dacă $w(\mathbf{e_2}) \leq 1$, atunci $\mathbf{s_1}$ este sau un cuvânt de pondere maxim 3 (dacă $w(\mathbf{e_2}) = 0$), sau o linie a lui B cu cel mult doi biţi schimbaţi (dacă $w(\mathbf{e_2}) = 1$).

Similar, dacă $w(\mathbf{e_1}) \leq 1$, atunci $\mathbf{s_2}$ este sau un cuvânt de pondere maxim 3 sau o linie a lui B cu cel mult doi biţi schimbaţi.

Dacă se foloseşte şi faptul că $\mathbf{s_2} = \mathbf{e_1}B + \mathbf{e_2} = (\mathbf{e_1} + \mathbf{e_2}B)B = \mathbf{s_1}B$ (deci se poate folosi doar prima matrice de control), putem defini următorul algoritm de decodificare a codurilor Golay extinse[1]:

[1] S-a notat cu $\mathbf{u_i}$ un cuvânt de lungime 12 cu 1 pe poziţia i şi 0 în rest.

Algoritm C:

1. Se calculează sindromul $\mathbf{s} = \mathbf{a}H^T$, unde $H = (I_{12} \mid B)$;
2. Dacă $w(\mathbf{s}) \leq 3$, atunci $\mathbf{e} = [\mathbf{s}, \mathbf{0}]$, STOP.
3. Dacă există o linie $\mathbf{b_i}$ a lui B cu $w(\mathbf{s} + \mathbf{b_i}) \leq 2$, atunci $\mathbf{e} = [\mathbf{s} + \mathbf{b_i}, \mathbf{u_i}]$, STOP.
4. Dacă $w(\mathbf{s}B) \leq 3$, atunci $\mathbf{e} = [\mathbf{0}, \mathbf{s}B]$, STOP.
5. Dacă există o linie $\mathbf{b_i}$ a lui B cu $w(\mathbf{s}B + \mathbf{b_i}) \leq 2$, atunci $\mathbf{e} = [\mathbf{u_i}, \mathbf{s}B + \mathbf{b_i}]$, STOP.
6. Dacă $\mathbf{e}$ nu a fost determinat încă, se cere retransmiterea.

După determinarea erorii $\mathbf{e}$, cuvântul - cod transmis se determină prin $\mathbf{v} = \mathbf{a} + \mathbf{e}$.

Exemplul 4.6 *Să decodificăm cuvântul* $\mathbf{a} = [101111101111, 010010010010]$.
Sindromul este

$$\mathbf{s} = \mathbf{a}H^T = 101111101111 + 001111101110 = 100000000001.$$

Deoarece $w(\mathbf{s}) = 2 \leq 3$*, se găseşte* $\mathbf{e} = [\mathbf{s}, \mathbf{0}] = [100000000001, 000000000000]$,
deci s-a transmis cuvântul $\mathbf{v} = \mathbf{a} + \mathbf{u} = [001111101110, 010010010010]$.

Deoarece $G = (I_{12} \mid B)$ este în forma eşalonat canonică, mesajul de informaţie (orice cuvânt din Z_2^{12}) apare pe primele 12 poziţii ale cuvântului - cod. Astfel, în exemplul de sus, mesajul de informaţie a fost 001111101110.

Exemplul 4.7 *Se cere decodificarea cuvântului* $\mathbf{a} = [001001001101, 101000101000]$.
Sindromul este

$$\mathbf{s} = \mathbf{a}H^T = 001001001101 + 111000000100 = 110001001001.$$

Deoarece $w(\mathbf{s}) = 5$*, se trece la pasul* 3 *al Algoritmului C şi se calculează:*

$$\begin{aligned} \mathbf{s} + \mathbf{b_1} &= 000110001100 \\ \mathbf{s} + \mathbf{b_2} &= 011111000010 \\ \mathbf{s} + \mathbf{b_3} &= 101101011110 \\ \mathbf{s} + \mathbf{b_4} &= 001001100100 \\ \mathbf{s} + \mathbf{b_5} &= 000000010010 \end{aligned}$$

Deoarece $w(\mathbf{s} + \mathbf{b_5}) \leq 2$*, se determină*

$$\mathbf{e} = [\mathbf{s} + \mathbf{b_5}, \mathbf{e_5}] = [000000010010, 000010000000]$$

şi se decide că s-a transmis cuvântul - cod

$$\mathbf{v} = \mathbf{a} + \mathbf{e} = [001001011111, 101010101000].$$

Exemplul 4.8 *Să decodificăm cuvântul* $\mathbf{a} = [000111000111, 011011010000]$.
Sindromul este

$$\mathbf{s} = \mathbf{a}H^T = \mathbf{e_1} + \mathbf{e_2}B = 000111000111 + 101010101101 = 101101101010$$

care are ponderea 7*. Trecând la pasul* 3 *se găseşte* $w(\mathbf{s} + \mathbf{b_i}) \geq 3$ *pentru toate liniile lui* B*; deci se continuă cu pasul* 4*: al doilea sindrom este* $\mathbf{s}B = 111001111101$ *cu ponderea* 8*. Pasul* 5 *va da:*

$$\begin{aligned} \mathbf{s}B + \mathbf{b_1} &= 001110111000 \\ \mathbf{s}B + \mathbf{b_2} &= 010111110110 \\ \mathbf{s}B + \mathbf{b_3} &= 100101101010 \\ \mathbf{s}B + \mathbf{b_4} &= 000001010000 \end{aligned}$$

S-a ajuns la $w(\mathbf{s}B + \mathbf{b_4}) \leq 2$*, deci se poate determina eroarea:*

$$\mathbf{e} = [\mathbf{u_4}, \mathbf{s}B + \mathbf{b_4}] = [000100000000, 000001010000].$$

Cuvântul - cod transmis a fost

$$\mathbf{v} = \mathbf{a} + \mathbf{e} = [000011000111, 011010000000]$$

4.2.3 Codul Golay binar

Prin relaxarea codului Golay extins (eliminarea ultimului bit din fiecare cuvânt - cod) se ajunge la *Codul Golay binar*.

Fie $\hat{B}$ matricea 12×11 obţinută din B prin eliminarea ultimei coloane. Definim $\hat{G} = (I_{12}|\hat{B})$. Codul liniar binar generat de $\hat{G}$ se numeşte *cod Golay* şi este notat cu C_{23}. Caracteristicile sale sunt:

$$n = 23, \quad k = 12, \quad \text{număr de cuvinte - cod: } 2^{12} = 4096.$$

Evident, extensia lui C_{23} este C_{24}.

Teorema 4.6 *Distanţa minimă a codului Golay este* $d = 7$*.*

Demonstraţie: Demonstraţia se poate face fie direct (similar celei de la Teorema 4.5) fie folosind faptul că C_{23} este relaxarea codului C_{24}, care are distanţa minimă 8. q.e.d.

În consecinţă, un cod Golay va corecta orice combinaţie de maxim 3 erori.

Teorema 4.7 *Codul Golay este perfect.*

Demonstraţie: Se verifică relaţia:

$2^{12}(C_{23}^0 + C_{23}^1 + C_{23}^2 + C_{23}^3) = 2^{12}(1 + 23 + 253 + 1771) = 2^{12}2^{11} = 2^{23}$. q.e.d.

Rezultă că orice cuvânt $\mathbf{a} \in Z_2^{23}$ se află la o distanţă cel mult 3 de un cuvânt - cod. Astfel, dacă se adaugă la sfârşit 0 sau 1, formând $\mathbf{a}0$ respectiv $\mathbf{a}1$ pentru a obţine un cuvânt de pondere impară, acest cuvânt este la distanţă maxim 3 de un cuvânt - cod $\mathbf{c} \in C_{24}$. Se foloseşte Algoritmul C pentru a obţine acest cuvânt - cod, apoi se elimină ultimul caracter din $\mathbf{c}$; se ajunge astfel la cel mai apropiat cuvânt - cod din C_{23} faţă de $\mathbf{a}$.

Algoritmul D:

1. Se formează cuvântul extins de pondere impară $\mathbf{a}0$ sau $\mathbf{a}1$;
2. Se decodifică $\mathbf{a}i$ folosind Algoritmul C şi se obţine $\mathbf{c} \in C_{24}$;
3. Se elimină ultimul caracter din $\mathbf{c}$.

Exemplul 4.9 *Să decodificăm* $\mathbf{a} = [001001001001, 11111110000]$.

Deoarece $\mathbf{a}$ *are pondere impară, se construieşte*

$$\mathbf{a}0 = 001001001001, 111111100000.$$

Sindromul acestui cuvânt este $\mathbf{s_1} = 100010111110$*.*

Pentru că $\mathbf{s_1} = \mathbf{b_6} + \mathbf{e_9} + \mathbf{e_{12}}$*,* $\mathbf{a}0$ *se decodifică în* $[001001000000, 111110100000]$*, aşa că* $\mathbf{a}$ *este decodificat în* $[001001000000, 11111010000]$*.*

4.3 Unicitatea codurilor perfecte binare

Ambele tipuri de coduri liniare prezentate până acum sunt coduri perfecte.

Se observă imediat că pentru corectarea unei simple erori, singurele coduri binare perfecte sunt codurile Hamming.

Vom mai arăta că această singularitate este valabilă şi în cazul codurilor Golay; anume, singurul cod binar perfect corector de 3 erori este codul Golay.

Pentru aceasta sunt necesare două leme:

Lema 4.1 *O condiţie necesară pentru existenţa unui* (n, k) *- cod binar perfect corector de* t *erori este* $\sum_{i=0}^{t} C_n^i = 2^p$ *pentru un anumit* p.

Demonstraţie: Rezultă imediat din relaţia scrisă în demonstraţia Teoremei 4.3, în care se ia $q = 2$. q.e.d.

Lema 4.2 $$\sum_{i=0}^{t} C_n^i = \frac{n+1}{t!} R_t(n)$$
unde t *este număr natural impar,* $R_t(X) \in Z[X]$, $gr(R_t(X)) = t - 1$.

Demonstraţie: Pentru $t = 1$ se verifică imediat.

Presupunem adevărată afirmaţia pentru t şi o demonstrăm pentru $t + 2$. Avem

$$\sum_{i=0}^{t+2} C_n^i = \frac{n+1}{t!} + C_n^{t+1} + C_n^{t+2} = \frac{n+1}{(t+2)!} \left[(t+1)(t+2) R_t(n) + \prod_{i=0}^{t} (n-i) \right]$$

Expresia din paranteza dreaptă este un polinom de gradul $t+1$; notându-l cu $R_{t+2}(n)$, afirmaţia este demonstrată. q.e.d.

Să considerăm acum cazul $t = 3$. Pentru ca să existe un cod binar perfect corector de 3 erori, cu lemele de sus, trebuie ca $(n+1)(n^2 - n + 6) = 3 \cdot 2^s$, sau

$$(n+1)[(n+1)^2 - 3(n+1) + 8] = 3 \cdot 2^s.$$

Considerată ca o ecuaţie în $n+1$, singurele soluţii întregi pozitive sunt de forma $n+1 = 2^r p$ unde $p = 1$ sau $p = 3$. Înlocuind, se ajunge la

$$2^{2r} p^3 - 2^r \cdot 3p^2 + 8p = 2^{s-r} \cdot 3 \tag{1}$$

Pentru $r \leq 3$ şi $p = 1, 3$ verificările se fac imediat. Pentru $r \geq 4$ se ajunge la contradicţie.

Singurele valori care verifică ecuaţia sunt:

$n = 0, 1, 2$	–	nu corespund nici unui cod.
$n = 3$	–	codul trivial cu un singur cuvânt - cod de lungime 3.
$n = 7$	–	codul (trivial) cu repetiţie $\{0000000, 1111111\}$.
$n = 23$	–	codul Golay.

În acest mod am demonstrat teorema:

Teorema 4.8 *Codul binar Golay este singurul cod binar perfect netrivial corector de* 3 *erori.*

Lemele 15.1 şi 4.2 pot fi folosite şi pentru alte valori impare ale lui t. Cercetările nu au condus la alte coduri perfecte binare, dar nici nu s-a demonstrat că nu există nici un cod perfect binar corector de t ($t > 3$ impar) erori. Afirmaţia este valabilă deocamdată pentru $t < 20$.

Un alt caz interesant de studiu este $q = 2,\ t = 2$. Aici se poate da teorema:

Teorema 4.9 *Nu există nici un cod netrivial binar perfect corector de* 2 *erori.*

Demonstraţie: Lema 15.1 conduce la relaţia

$$(2n+1)^2 = 2^{s+3} - 7.$$

Ecuaţia $x^2 + 7 = 2^m$ a fost studiată în multe articole (vezi Math. Rev. 26, #74).

Singurele soluţii sunt $x = 1, 3, 5, 11, 181$ cărora le corespund:

$n = 0, 1$ – fără coduri.
$n = 2$ – codul trivial cu un singur cuvânt.
$n = 5$ – codul cu repetiţie $\{00000, 11111\}$.
$n = 90$.

Acest ultim caz este eliminat de următorul rezultat (Van Lindt - Coding theory, pp. 95):

Dacă există un cod binar perfect corector de t *erori, atunci* $\dfrac{n+1}{t+1}$ *este număr întreg.* q.e.d.

4.4 Exerciţii

4.1 *Fie* $(8, 4)$ *- codul Hamming binar extins. Decodificaţi cuvintele*

$$10101010 \quad 11010110 \quad 11111111.$$

4.2 *Să se demonstreze că toate cuvintele - cod ale codului Hamming binar extins* $(2^r, 2^r - r - 1)$ *au pondere pară.*

4.3 *Construiţi codurile Hamming ternare pentru* $r = 2,\ 3$ *şi determinaţi decodificarea pe baza sindromurilor.*

4.4 *Construiţi* $(5, 3)$ *- codul Hamming peste* Z_4*. Determinaţi toate cuvintele - cod şi tabela de decodificare cu sindromuri.*

Decodificaţi cuvintele: 11223, 32101 2222 1100.

4.5 *Demonstraţi afirmaţiile din Corolarul 15.1.*

4.6 *Arătaţi că* C_{24} *conţine un cuvânt cu toate componentele egale cu* 1 *şi nici un cuvânt de pondere* 20.

Demonstraţi că numărătorul de ponderi al lui C_{24} *este:*

$$1 + 759X^8 + 2576X^{12} + 759X^{16} + X^{24}.$$

4.7 *De ce în Algoritmul D se face o extensie a cuvintelor la cuvinte de pondere impară? Daţi un exemplu pentru a justifica raţionamenul.*

4.8 *În codul Golay extins C_{24} să se decodifice – dacă este posibil – cuvintele:*

$$\begin{array}{ll} [111000000000, 011011011011] & [111111000000, 100011100111] \\ [111111000000, 101011100111] & [111111000000, 111000111000] \\ [111000000000, 110111001101] & [110111001101, 111000000000] \\ [000111000111, 101000101101] & [110000000000, 101100100000] \end{array}$$

4.9 *Să se determine cea mai probabilă eroare - tip pentru cuvinte având sindromurile:*

$$\begin{array}{ll} \mathbf{s_1} = 010010000000, & \mathbf{s_2} = 011111010000 \\ \mathbf{s_1} = 010010100101, & \mathbf{s_2} = 001000110000 \\ \mathbf{s_1} = 111111000101, & \mathbf{s_2} = 111100010111 \\ \mathbf{s_1} = 111111111011, & \mathbf{s_2} = 010010001110 \\ \mathbf{s_1} = 001101110110, & \mathbf{s_2} = 111110101101 \\ \mathbf{s_1} = 010111111001, & \mathbf{s_2} = 100010111111 \end{array}$$

4.10 *Folosind C_{23}, decodificaţi cuvintele:*

$$\begin{array}{ll} [101011100000, 10101011011] & [101010000001, 11011100010] \\ [100101011000, 11100010000] & [011001001001, 01101101111] \end{array}$$

4.11 *Detaliaţi demonstraţia Teoremei 4.6.*

4.12 *Rezolvaţi ecuaţia* (1).

Capitolul 5

Coduri Reed - Muller

Vom introduce o nouă clasă de coduri binare, caracterizate printr-o tehnică de decodificare deosebit de simplă: codurile *Reed - Muller* $(R-M)$. Ele au fost definite de Reed, iar Muller a construit modalitatea de decodificare şi – implicit – de detectare şi corectare a erorilor. Unul din aceste coduri ($\mathcal{RM}(1,5)$) a fost folosit în 1969 de sonda Mariner pentru transmiterea de imagini de pe Lună. Fiecare pixel din imagine avea asignat una din $2^6 = 64$ grade de umbră, iar cei şase biţi de informaţie erau codificaţi într-un cuvânt de lungime 32. Codul $\mathcal{RM}(1,5)$ poate corecta până la 7 erori.

5.1 Definirea prin funcţii booleene

5.1.1 Funcţii şi polinoame booleene

Definiţia 5.1 *O funcţie booleană de* m $(m \geq 1)$ *variabile este o aplicaţie*

$$f : Z_2^m \to Z_2.$$

O modalitate simplă folosită pentru definirea unei funcţii booleene este asocierea unei *tabele de adevăr*: un tablou $(m+1) \times 2^m$ care conţine toate combinaţiile posibile de m valori binare, cărora li se asociază valoarea funcţiei (de asemenea o valoare binară). Prin convenţie, primii m biţi de pe coloana i $(0 \leq i \leq 2^m - 1)$ reprezintă scrierea în baza 2 a numărului i.

Exemplul 5.1 *Următoarea tabelă de adevăr defineşte o funcţie booleană de* 3 *variabile:*

$\mathbf{x_0}$	0	1	0	1	0	1	0	1
$\mathbf{x_1}$	0	0	1	1	0	0	1	1
$\mathbf{x_2}$	0	0	0	0	1	1	1	1
$\mathbf{f}$	0	1	1	0	1	1	1	0

Observăm că o astfel de tabelă defineşte un cuvânt binar de lungime 8*. Afirmaţia este adevărată şi invers: orice cuvânt binar de lungime* 8 *este definit printr-o tabelă de adevăr a unei funcţii booleene de* 3 *variabile. Astfel, se pot identifica funcţiile booleene de* 3 *variabile prin cuvintele binare de lungime* 8*. În tabela de sus, cuvântul* 01101110 *este pus în corespondenţă biunivocă cu funcţia* $\mathbf{f}$*.*

În cele ce urmează, orice cuvânt binar $\mathbf{f} = f_0 f_1 \dots f_{2^m-1}$ de lungime 2^m este conside-rat ca o funcţie booleană de m variabile, unde

$$\begin{aligned} \mathbf{f}(0,0,\dots,0,0) &= f_0, \\ \mathbf{f}(0,0,\dots,0,1) &= f_1, \\ \mathbf{f}(0,0,\dots,1,0) &= f_2, \\ &\vdots \\ \mathbf{f}(1,1,\dots,1,1) &= f_{2^m-1} \end{aligned}$$

În general, $f_i = \mathbf{f}(i_{m-1}, \dots, i_1, i_0)$, unde $i = \sum_{k=0}^{m-1} i_k 2^k$

($i_{m-1} \dots i_1 i_0$ este scrierea în binar a lui i).

Exemplul 5.2 *Există două funcţii booleene constante:*

$$\mathbf{1} = 11\dots11, \qquad \mathbf{0} = 00\dots00.$$

Exemplul 5.3 *Orice variabilă poate fi tratată ca o funcţie booleană. De exemplu,* $\mathbf{x_0}$ *este funcţia booleană care asignează fiecărui* m*-tuplu* $(x_0, x_1, \dots, x_{m-1})$ *valoarea primei coordonate* x_0*. Deci, valoarea este* 0 *pentru toate numerele pare şi* 1 *pentru toate numerele impare:* $\mathbf{x_0} = 0101\dots01$ *(vezi prima linie din Exemplul 20.3 pentru cazul* $m = 3$*).*

În general,

$\mathbf{x_i}$ *este cuvântul binar în care pe poziţia* k $(0 \le k \le 2^m - 1)$ *este* 1 *atunci şi numai atunci când scrierea binară a lui* k *conţine* 1 *pe poziţia* i*.*

Această observaţie rezultă din modul de scriere al tabelelor de adevăr.
De exemplu, $\mathbf{x_1} = 00110011\dots0011$ *şi* $\mathbf{x_{m-1}} = \underbrace{00\dots00}_{2^{m-1}}\underbrace{11\dots11}_{2^{m-1}}$.

Pentru $m = 4$*, cele patru variabile sunt descrise în Tabelul 5.1:*

Tabelul 5.1:

$\mathbf{x_0}$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$\mathbf{x_1}$	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
$\mathbf{x_2}$	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
$\mathbf{x_3}$	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Pe mulţimea funcţiilor booleene se definesc două operaţii:

- **Suma logică** (*sau exclusiv*): $\mathbf{f + g = h}$
  unde $h_i = f_i + g_i \ (mod\ 2),\ 0 \le i \le 2^m - 1.$
- **Produsul logic** (*şi*): $\mathbf{fg = h}$
  unde $h_i = f_i g_i \ (mod\ 2),\ 0 \le i \le 2^m - 1.$

Observaţia 5.1

1. *Produsul logic verifică relaţia* $\mathbf{ff} = \mathbf{f}$.

   Deci în reprezentarea funcţiilor nu vor apare exponenţi mai mari de 1.

2. *Există şi alte operaţii care pot fi exprimate cu ajutorul sumei şi produsului logic. Astfel,*

   - *Negaţia* $\overline{\mathbf{f}} = \mathbf{1} + \mathbf{f}$;
   - $\vee$ *(sau disjunctiv):* $\mathbf{f} \vee \mathbf{g} = \mathbf{f} + \mathbf{g} + \mathbf{fg}$.

Definiţia 5.2 *Un polinom boolean de* m *nedeterminate este o sumă de termeni din mulţimea* $\{\mathbf{0}, \mathbf{1}\} \cup \{\mathbf{x_{i_1} x_{i_2} \ldots x_{i_k}} | 0 \leq i_1 < i_2 < \ldots < i_k \leq m-1,\ k \geq 1\}$.

Funcţia $\mathbf{0}$ este numită *polinom boolean de gradul* -1, funcţia $\mathbf{1}$ este numită *polinom boolean de gradul* 0, iar orice alt polinom boolean are gradul k unde k este numărul maxim de factori dintr-un termen al lui $\mathbf{f}$.

Exemplul 5.4 *Polinomul boolean* $\mathbf{1} + \mathbf{x_0 x_1}$ *de* 3 *nedeterminate are gradul* 2. *El este negaţia polinomului*

$$\mathbf{x_0 x_1} = (01010101)(00110011) = 00010001.$$

Deci $\mathbf{1} + \mathbf{x_0 x_1} = 11101110$.

Acelaşi polinom, considerat ca funcţie de 4 *nedeterminate, este cuvântul*

$$1110111011101110.$$

Exemplul 5.5 *Polinomul* $\mathbf{x_i x_j}$ $(i \neq j)$ *este cuvântul binar în care pe poziţia* k *este* 1 *dacă şi numai dacă reprezentarea binară a lui* k *are* 1 *pe poziţiile* i *şi* j. *Numărul acestor situaţii este* 2^{m-2} *(deoarece pot fi alese arbitrar* $m-2$ *valori din reprezentarea binară a lui* k *pe* m *poziţii). Deci* $w(\mathbf{x_i x_j}) = 2^{m-2}$.

Mai general, se poate arăta prin inducţie

Dacă $i_1, i_2, \ldots, i_r$ sunt valori distincte din $[0, m-1]$, atunci $w(\mathbf{x_{i_1} x_{i_2} \ldots x_{i_r}}) = 2^{m-r}$. $(*)$

Fiecare polinom boolean de m variabile determină un cuvânt binar de lungime 2^m: pentru o singură nedeterminată, se foloseşte Exemplul 14.4, după care se operează adunările şi multiplicările necesare.

Invers, orice cuvânt binar $\mathbf{f} = f_0 f_1 \ldots f_{2^m-1}$ poate fi translatat într-un polinom boolean pe baza următoarei propoziţii:

Propoziţia 5.1 *Dacă* $\mathbf{f}$ *este o funcţie booleană de* m *variabile, atunci:*

$f(x_0, \ldots, x_{m-2}, x_{m-1}) = f(x_0, \ldots, x_{m-2}, 0) + [f(x_0, \ldots, x_{m-2}, 0) + f(x_0, \ldots, x_{m-2}, 1)]\mathbf{x_{m-1}}$.

Demonstraţie: Deoarece x_{m-1} poate lua doar două valori $(0, 1)$, este suficient să verificăm identitatea pentru acestea. Cazul $x_{m-1} = 0$ se verifică banal. Pentru $x_{m-1} = 1$ avem:

$f(x_0, \ldots, x_{m-2}, 0) + [f(x_0, \ldots, x_{m-2}, 0) + f(x_0, \ldots, x_{m-2}, 1)] = f(x_0, \ldots, x_{m-2}, 1)$. q.e.d.

Exemplul 5.6 *Să translatăm* $\mathbf{f} = 01101110$ *într-un polinom boolean de* 3 *variabile. Vom aplica pe etape formula din Propoziţia 20.1:*

$\mathbf{f} = 0110 + [0110 + 1110]\mathbf{x_2} = 0110 + 1000\mathbf{x_2} =$
$= (01 + [01 + 10]\mathbf{x_1}) + (10 + [10 + 00]\mathbf{x_1})\mathbf{x_2} = 01 + 11\mathbf{x_1} + 10\mathbf{x_2} + 10\mathbf{x_1x_2} =$
$= (0 + [0 + 1]\mathbf{x_0}) + (1 + [1 + 1]\mathbf{x_0})\mathbf{x_1} + (1 + [1 + 0]\mathbf{x_0})\mathbf{x_2} + (1 + [1 + 0]\mathbf{x_0})\mathbf{x_1x_2} =$
$= \mathbf{x_0} + \mathbf{x_1} + \mathbf{x_2} + \mathbf{x_0x_2} + \mathbf{x_1x_2} + \mathbf{x_0x_1x_2}.$

Teorema 5.1 *Spaţiul liniar* Z_2^n, $(n = 2^m)$ *are o bază formată din toate monoamele booleene:*

$$\begin{array}{ll} \mathbf{1} & \\ \mathbf{x_i} & (0 \le i \le m-1) \\ \mathbf{x_i x_j} & (0 \le i, j \le m-1, \quad i \ne j) \\ \vdots & \\ \mathbf{x_0 x_1 \ldots x_{m-1}} & \end{array}$$

Demonstraţie: Fiecare cuvânt de lungime $n = 2^m$ este o funcţie booleană de m nedeterminate, care poate fi exprimată printr-un polinom boolean. Monoamele booleene pot fi considerate în Z_2^n şi sunt evident liniar independente. În plus, deoarece pentru fiecare $k = 0, 1, \ldots, m$ există C_m^k monoame de grad k, numărul lor total va fi $\sum_{k=0}^{m} C_m^k = 2^m = n$, adică dimensiunea spaţiului liniar. q.e.d.

5.1.2 Coduri Reed - Muller

Definiţia 5.3 *Se numeşte cod Reed - Muller de lungime* $n = 2^m$ *şi grad* $r \quad (0 \le r \le m)$, *codul liniar* $\mathcal{RM}(r, m)$ *al tuturor cuvintelor binare de lungime* n *care au – ca polinoame booleene – gradul maxim* r.

Exemplul 5.7 $\mathcal{RM}(0, m)$ *este format din toate polinoamele de grad cel mult* 0, *adică* $\mathbf{0}$ *şi* $\mathbf{1}$. *Deci* $\mathcal{RM}(0, m)$ *este codul cu repetiţie de lungime* 2^m.

Exemplul 5.8 $\mathcal{RM}(1, m)$ *are baza* $\mathbf{1}, \mathbf{x_0}, \ldots, \mathbf{x_{m-1}}$; *orice polinom de grad cel mult* 1 *se poate scrie ca sumă de o parte din aceste* $m + 1$ *polinoame (liniar independente). Deci,* $\mathcal{RM}(1, m)$ *este un* $(2^m, m + 1)$ *- cod liniar.*

De exemplu, $\mathcal{RM}(1, 3)$ *are matricea generatoare:*

$$G = \begin{pmatrix} \mathbf{1} \\ \mathbf{x_0} \\ \mathbf{x_1} \\ \mathbf{x_2} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

care generează codul Hamming extins $(8, 4)$.

Similar, $\mathcal{RM}(1, 4)$ *este un* $(16, 5)$ *- cod liniar binar, iar* $\mathcal{RM}(1, 5)$ *este un* $(32, 6)$ *- cod liniar, folosit în* 1969 *de programul Mariner, după cum s-a menţionat anterior.*

Propoziţia 5.2 $\mathcal{RM}(r, m)$ *are* $k = C_m^0 + C_m^1 + \ldots + C_m^r$ *simboluri de informaţie.*

Demonstraţie: Codul $\mathcal{RM}(r,m)$ are ca bază monoamele booleene

$$\{\mathbf{1}\} \cup \{\mathbf{x_{i_1}x_{i_2}}\dots\mathbf{x_{i_s}} | s \le r,\ 0 \le i_1 < i_2 < \dots < i_s < m\}.$$

Numărul acestor monoame este $C_m^0 + C_m^1 + \dots + C_m^r$. Cum ele sunt liniar independente, vor forma liniile matricii generatoare G a codului; ori numărul de linii este egal cu numărul de simboluri de informaţie. q.e.d.

Propoziţia 5.3 *Dualul codului $\mathcal{RM}(r,m)$ este codul $\mathcal{RM}(m-r-1,m)$.*

Demonstraţie: Trebuie arătat că cele două coduri au baze ortogonale şi suma dimensiunilor lor este egală cu suma întregului spaţiu.

A: Fie $\mathbf{v_{i_1}v_{i_2}}\dots\mathbf{v_{i_p}}$ $(p \le r)$ un monom din baza codului $\mathcal{RM}(r,m)$ şi $\mathbf{v_{j_1}v_{j_2}}\dots\mathbf{v_{j_s}}$ $(s \le m-r-1)$ un monom din baza codului $\mathcal{RM}(m-r-1,m)$. Produsul lor are t $(t \le r+m-r-1 = m-1)$ variabile distincte (variabilele comune apar o singură dată), deci ponderea lui este (conform $(*)$) $2^{m-t} \ge 2$. Fiind un număr par, rezultă că produsul scalar (adică suma în binar a termenilor) este 0.

B: Suma dimensiunilor celor două coduri este:

$$\sum_{i=0}^{r} C_m^i + \sum_{i=0}^{m-r-1} C_m^i = \sum_{i=0}^{r} C_m^i + \sum_{i=r+1}^{m} C_m^i = \sum_{i=0}^{m} C_m^i = 2^m = n.\ \text{q.e.d.}$$

Exemplul 5.9 *$\mathcal{RM}(m-2,m)$ este codul Hamming extins de lungime 2^m.*
Într-adevăr, codul său dual este $\mathcal{RM}(m-(m-2)-1,m) = \mathcal{RM}(1,m)$, deci $\mathcal{RM}(m-2,m)$ are matricea de control

$$H = \begin{pmatrix} \mathbf{1} \\ \mathbf{x_0} \\ \vdots \\ \mathbf{x_m} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & \dots & 0 & 1 & 0 & 1 \\ & & & & \vdots & & & & \\ 0 & 0 & 0 & 0 & \dots & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Dacă adunăm prima linie la toate celelalte şi apoi o permutăm cu ultima linie, se obţine altă matrice de control a codului:

$$H \sim \left(\begin{array}{cccccccc|c} 1 & 0 & 1 & 0 & \dots & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & \dots & 1 & 1 & 0 & 0 \\ & & & & \vdots & & & & \\ 1 & 1 & 1 & 1 & \dots & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 \end{array}\right).$$

După eliminarea ultimei linii şi coloane se obţine un $(2^m-1,m)$ - cod ale cărui coloane sunt nenule şi diferite două câte două, adică matricea de control H_0 a unui cod Hamming binar. Deci H este matricea de control a unui cod Hamming extins.

Codificarea mesajelor cu un cod $R-M$ se realizează normal, înmulţind mesajul de informaţie cu matricea generatoare. În acest fel biţii de informaţie devin coeficienţii polinomului boolean corespunzător. De exemplu, în $\mathcal{RM}(1,m)$ codificarea celor $m+1$ caractere de informaţie este:

$$(a_1, a_2, \dots, a_{m+1}) \begin{pmatrix} \mathbf{1} \\ \mathbf{x_0} \\ \vdots \\ \mathbf{x_{m-1}} \end{pmatrix} = a_1\mathbf{1} + a_2\mathbf{x_0} + \dots + a_{m+1}\mathbf{x_{m-1}}.$$

5.2 Definirea recursivă a codurilor R - M

Să introducem o altă modalitate de definire a codurilor Reed - Muller, nu prin polinoame booleene, ci prin construcţie recursivă.

Definiţia 5.4 *Fie $m \geq 0$ un număr natural. Se defineşte codul Reed - Muller $\mathcal{RM}(r,m)$ de ordin r $(0 \leq r \leq m)$ şi lungime $n = 2^m$ astfel:*

- $\mathcal{RM}(0,m) = \{00\ldots0, 11\ldots1\}, \quad \mathcal{RM}(m,m) = Z_2^n$.
- $\mathcal{RM}(p,m) = \{[\mathbf{a}, \mathbf{a}+\mathbf{b}] \mid \mathbf{a} \in \mathcal{RM}(p,m-1),\ \mathbf{b} \in \mathcal{RM}(p-1,m-1)\}, 0 < p < r$.

S-a notat cu $[\mathbf{x}, \mathbf{y}]$ un cuvânt de lungime 2^m scris ca alăturare de două subcuvinte de lungimi egale (2^{m-1}), separate prin virgulă (similar notaţiei de la codurile Golay).

Exemplul 5.10 $\mathcal{RM}(0,0) = \{0,1\}$
$\mathcal{RM}(0,1) = \{00,11\}$, $\mathcal{RM}(1,1) = \{00,01,10,11\} = Z_2^2$
$\mathcal{RM}(0,2) = \{0000,1111\}$, $\mathcal{RM}(2,2) = Z_2^4$
$\mathcal{RM}(1,2) = \{[\mathbf{a}, \mathbf{a}+\mathbf{b}] \mid \mathbf{a} \in \{00,01,10,11\}, \mathbf{b} \in \{00,11\}\} =$
$= \{0000, 0011, 0100, 0111, 1000, 1011, 1100, 1111\}$.

În mod similar se poate da o definiţie recursivă a matricii generatoare $G(r,m)$ pentru codul $\mathcal{RM}(r,m)$.

- $G(0,m) = (11\ldots1)$;
- pentru $0 < p < r < m$, $\quad G(p,m) = \begin{pmatrix} G(p,m-1) & G(p,m-1) \\ \mathbf{0} & G(p-1,m-1) \end{pmatrix}$;
- $G(m,m) = \begin{pmatrix} G(m-1,m) \\ 00\ldots01 \end{pmatrix}$.

Teorema 5.2 *$G(r,m)$ este matrice generatoare pentru codul $\mathcal{RM}(r,m)$.*

Demonstraţie: Se verifică prin inducţie după r. q.e.d.

Exemplul 5.11 *Să considerăm $r = 2$; atunci lungimea este $n = 2^2 = 4$ şi pentru $r = 1, 2$ avem*

$$G(1,2) = \begin{pmatrix} G(1,1) & G(1,1) \\ \mathbf{0} & G(0,1) \end{pmatrix} \qquad G(2,2) = \begin{pmatrix} G(1,2) \\ 0\ 0\ 0\ 1 \end{pmatrix}.$$

Din definiţie, matricile generatoare pentru $\mathcal{RM}(0,1)$ şi $\mathcal{RM}(1,1)$ sunt

$$G(0,1) = (1\ 1), \qquad G(1,1) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \text{ aşa că}$$

$$G(1,2) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \qquad G(2,2) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Propoziţia 5.4 $\mathcal{RM}(r-1,m) \subseteq \mathcal{RM}(r,m)$.

Demonstraţie: Să considerăm iniţial matricea

$$G(1,m) = \begin{pmatrix} G(1,m-1) & G(1,m-1) \\ \mathbf{0} & G(0,m-1) \end{pmatrix}.$$

Pentru că $\mathbf{1}$ este prima linie a lui $G(1,m-1)$, cuvântul $[\mathbf{1},\mathbf{1}]$ formează prima linie a matricii $(G(1,m-1)\ G(1,m-1))$. Deci $\mathcal{RM}(0,m) = \{\mathbf{0},\mathbf{1}\}$ este conţinut în codul $\mathcal{RM}(1,m)$.

În general, deoarece $G(r-1,m-1)$ este submatrice a lui $G(r,m-1)$ şi $G(r-2,m-1)$ este o submatrice a lui $G(r-1,m-1)$, este evident că

$$G(r-1,m) = \begin{pmatrix} G(r-1,m-1) & G(r-1,m-1) \\ \mathbf{0} & G(r-2,m-1) \end{pmatrix}$$

este o submatrice a lui $G(r,m)$, deci $\mathcal{RM}(r-1,m)$ este subcod al lui $\mathcal{RM}(r,m)$. q.e.d.

Teorema 5.3 *$\mathcal{RM}(r,m)$ are distanţa $d = 2^{m-r}$.*

Demonstraţie: Vom folosi o inducţie după r:

Pentru $r = 0$, evident: $\mathcal{RM}(0,m)$ fiind codul cu repetiţie, distanţa sa este $d = n = 2^m$.

La pasul II, deoarece

$\mathcal{RM}(r,m) = \{[\mathbf{x},\mathbf{x}+\mathbf{y}] | \mathbf{x} \in \mathcal{RM}(r,m-1), \mathbf{y} \in \mathcal{RM}(r-1,m-1)\}$ şi $\mathcal{RM}(r-1,m-1) \subseteq \mathcal{RM}(r,m-1)$ (Propoziţia 11.3), rezultă $\mathbf{x}+\mathbf{y} \in \mathcal{RM}(r,m-1)$.

Dacă $\mathbf{x} \neq \mathbf{y}$, conform ipotezei de inducţie $w(\mathbf{x}+\mathbf{y}) \geq 2^{m-1-r}$. Cum şi $w(\mathbf{x}) \geq 2^{m-1-r}$, putem scrie $w([\mathbf{x}+\mathbf{y},\mathbf{x}]) = w(\mathbf{x}+\mathbf{y}) + w(\mathbf{x}) \geq 2^{m-r}$.

Dacă $\mathbf{x} = \mathbf{y}$, atunci $[\mathbf{x},\mathbf{x}+\mathbf{y}] = [\mathbf{y},\mathbf{0}]$; dar $\mathbf{y} \in \mathcal{RM}(r-1,m-1)$ şi deci $w([\mathbf{y},\mathbf{0}]) = w(\mathbf{y}) \geq 2^{m-r}$.

Cum orice linie a matricii generatoare este cuvânt - cod, iar ultima linie are ponderea exact 2^{m-r}, demonstraţia este încheiată. q.e.d.

5.3 Definirea geometrică a codurilor R-M

Codurile Reed - Muller mai pot fi definite şi geometric - prin folosirea spaţiilor afine. Avantajul acestei reprezentări constă în modalitatea mai simplă de aplicare a algoritmilor de decodificare.

Pentru uşurinţa descrierii am construit întâi cazul tri-dimensional. De asemenea, pentru a vedea echivalenţa cu definirea anterioară a codurilor Reed - Muller, vom face permanent legătura cu polinoamele booleene (sau cu funcţiile lor caracteristice).

5.3.1 Cazul 3 -dimensional

Spaţiul euclidian 3 - dimensional binar este mulţimea $\{(a,b,c)|a,b,c \in Z_2\}$. Spre deosebire de spaţiul euclidian obişnuit - unde cele trei coordonate luau valori în $\mathcal{R}$ - aici numărul punctelor este finit: numai 8. Ele pot fi listate, renotându-le astfel:

Punct	Funcţie caracteristică
$\mathbf{p_0} = 000$	00000001
$\mathbf{p_1} = 001$	00000010
$\mathbf{p_2} = 010$	00000100
$\mathbf{p_3} = 011$	00001000
$\mathbf{p_4} = 100$	00010000
$\mathbf{p_5} = 101$	00100000
$\mathbf{p_6} = 110$	01000000
$\mathbf{p_7} = 111$	10000000

Dreptele pot fi definite în geometria euclidiană prin expresii de forma

$$\mathbf{a} + t\mathbf{b} \qquad \mathbf{a}, \mathbf{b} \in Z_2^3,\ \mathbf{b} \neq \mathbf{0}.$$

unde t este un parametru binar ($t \in \{0,1\}$). Deci o dreaptă în spaţiul 3 - dimensional binar are numai 2 puncte: $\mathbf{a},\ \mathbf{a}+\mathbf{b}$. Invers, orice pereche de două puncte distincte $\mathbf{a}, \mathbf{a}'$ formează o dreaptă, anume $\mathbf{a} + t(\mathbf{a}' - \mathbf{a})$. Putem astfel să considerăm dreptele ca fiind totalitatea celor $C_8^2 = 28$ submulţimi de câte două puncte

$$\{\mathbf{p_0}, \mathbf{p_1}\}, \{\mathbf{p_0}, \mathbf{p_2}\}, \ldots, \{\mathbf{p_6}, \mathbf{p_7}\}.$$

În mod similar, *planele* din geometria euclidiană sunt definite

$$\mathbf{a} + t_1\mathbf{b} + t_2\mathbf{c}, \quad \mathbf{a}, \mathbf{b}, \mathbf{c} \in Z_2^3, \quad \mathbf{b}, \mathbf{c} \text{ liniar independente},$$

unde t_1, t_2 sunt parametri binari.

Un plan este format deci din patru puncte: $\mathbf{a}, \mathbf{a}+\mathbf{b}, \mathbf{a}+\mathbf{c}, \mathbf{a}+\mathbf{b}+\mathbf{c}$. Aparent, deşi numărul planelor în geometria euclidiană binară 3 - dimensională ar trebui să fie $C_8^4 = 70$, condiţia de liniar independenţă reduce acest număr la 14:

Plan	Funcţie caracteristică	Polinom boolean
$\{\mathbf{p_1}, \mathbf{p_3}, \mathbf{p_5}, \mathbf{p_7}\}$	10101010	$\mathbf{x_0}$
$\{\mathbf{p_2}, \mathbf{p_3}, \mathbf{p_6}, \mathbf{p_7}\}$	11001100	$\mathbf{x_1}$
$\{\mathbf{p_4}, \mathbf{p_5}, \mathbf{p_6}, \mathbf{p_7}\}$	11110000	$\mathbf{x_2}$
$\{\mathbf{p_0}, \mathbf{p_2}, \mathbf{p_4}, \mathbf{p_6}\}$	01010101	$\mathbf{1} + \mathbf{x_0}$
$\{\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_4}, \mathbf{p_5}\}$	00110011	$\mathbf{1} + \mathbf{x_1}$
$\{\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}\}$	00001111	$\mathbf{1} + \mathbf{x_2}$
$\{\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_5}, \mathbf{p_6}\}$	01100110	$\mathbf{x_0} + \mathbf{x_1}$
$\{\mathbf{p_1}, \mathbf{p_3}, \mathbf{p_4}, \mathbf{p_6}\}$	01011010	$\mathbf{x_0} + \mathbf{x_2}$
$\{\mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}, \mathbf{p_5}\}$	00111100	$\mathbf{x_1} + \mathbf{x_2}$
$\{\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_4}, \mathbf{p_7}\}$	10010110	$\mathbf{x_0} + \mathbf{x_1} + \mathbf{x_2}$
$\{\mathbf{p_0}, \mathbf{p_3}, \mathbf{p_4}, \mathbf{p_7}\}$	10011001	$\mathbf{1} + \mathbf{x_0} + \mathbf{x_1}$
$\{\mathbf{p_0}, \mathbf{p_2}, \mathbf{p_5}, \mathbf{p_7}\}$	10100101	$\mathbf{1} + \mathbf{x_0} + \mathbf{x_2}$
$\{\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_6}, \mathbf{p_7}\}$	11000011	$\mathbf{1} + \mathbf{x_1} + \mathbf{x_2}$
$\{\mathbf{p_0}, \mathbf{p_3}, \mathbf{p_5}, \mathbf{p_6}\}$	01101001	$\mathbf{1} + \mathbf{x_0} + \mathbf{x_1} + \mathbf{x_2}$

Tabelul 5.2

În general, un plan este descris de ecuaţia generală

$$h_0\mathbf{x_0} + h_1\mathbf{x_1} + h_2\mathbf{x_2} = c$$

care defineşte un subspaţiu 2 - dimensional al lui Z_2^3.

Din faptul că orice dreaptă este o intersecţie de două plane, ea poate fi descrisă printr-un sistem de două ecuaţii:

$$h_0\mathbf{x_0} + h_1\mathbf{x_1} + h_2\mathbf{x_2} = c, \qquad h_0'\mathbf{x_0} + h_1'\mathbf{x_1} + h_2'\mathbf{x_2} = c'.$$

Dreptele şi planele sunt exemple de spaţii afine. Un *spaţiu afin* în Z_2^3 este o mulţime de forma

$$C_a = \mathbf{a} + C = \{\mathbf{a} + \mathbf{b} \mid \mathbf{b} \in C\}$$

unde $\mathbf{a} \in Z_2^3$ iar C este un subspaţiu liniar din Z_2^3. Dacă C este un spaţiu de dimensiune s, atunci C_a este un s - *spaţiu afin.*

În particular, dreptele sunt 1 - spaţii afine, iar planele: 2 - spaţii afine. Pentru fiecare punct $\mathbf{p_i}$ avem un 0 - spaţiu afin, şi – în sfârşit – există un 3 - spaţiu afin unic - Z_2^3.

Orice spaţiu afin L poate fi descris de un cuvânt binar $\mathbf{f_L} = f_7 \dots f_1 f_0$ definit prin

$$f_i = \begin{cases} 1 & \text{dacă } \mathbf{p_i} \in L, \quad (0 \le i \le 7) \\ 0 & \text{altfel} \end{cases}$$

Cuvântul $\mathbf{f_L}$ (sau funcţia booleană de trei variabile corespunzătoare) se numeşte *funcţia caracteristică* a spaţiului afin L (tabelele anterioare listează aceste funcţii pentru 0 şi 2 - spaţii afine).

Fiind date două spaţii afine L, L', intersecţia lor $L \cap L'$ este caracterizată de produsul logic $\mathbf{f_L f_{L'}}$.

Exemplul 5.12 *Primele două plane din Tabelul* 5.2 *se intersectează după linia* $\{\mathbf{p_3}, \mathbf{p_7}\}$. *Produsul logic al funcţiilor lor caracteristice este* $\mathbf{x_0 x_1} = 10001000$, *care se poate verifica imediat ca fiind funcţia caracteristică a dreptei* $\{\mathbf{p_3}, \mathbf{p_7}\}$.

5.3.2 Cazul m - dimensional

Vom extinde construcţiile anterioare la un cadru mai general, al geometriei euclidiene m - dimensionale binare. Punctele (elementele lui Z_2^m) pot fi renotate după extensia binară a indicilor; mai clar, vom scrie $Z_2^m = \{\mathbf{p_0}, \mathbf{p_1}, \dots, \mathbf{p_{2^m-1}}\}$ unde $\mathbf{p_i}$ reprezintă scrierea în binar (pe m biţi) a numărului i $(0 \le i \le 2^m - 1)$. Deci
$\mathbf{p_0} = 000 \dots 00, \qquad \mathbf{p_1} = 000 \dots 01, \quad \mathbf{p_2} = 000 \dots 10, \quad \dots \quad , \mathbf{p_{2^m-1}} = 111 \dots 11.$

Definiţia 5.5 *Fie* C *un subspaţiu liniar* r - *dimensional al lui* Z_2^m *şi* $\mathbf{a} \in Z_2^m$. *Mulţimea*

$$C_\mathbf{a} = \mathbf{a} + C = \{\mathbf{a} + \mathbf{b} | \mathbf{b} \in C\}$$

se numeşte r - *spaţiu afin modulo* C *în geometria euclidiană* m - *dimensională.*

Un $(m-1)$ - *spaţiu afin se numeşte "hiperplan".*

Dacă $\mathbf{b_1}, \ldots, \mathbf{b_r}$ este o bază a lui C, r - spaţiul afin $C_{\mathbf{a}}$ se notează

$$\mathbf{a} + t_1\mathbf{b_1} + \ldots + t_r\mathbf{b_r}.$$

El are 2^r puncte (date de variantele de alegere ale parametrilor binari t_i, $1 \leq i \leq r$).

O altă modalitate de notare a r - spaţiilor afine se realizează cu ajutorul sistemelor de ecuaţii liniare: astfel, dacă C este definit ca mulţimea soluţiilor sistemului $H\mathbf{x}^T = \mathbf{0}^T$, atunci $C_{\mathbf{a}}$ este dat de mulţimea soluţiilor sistemului

$$H\mathbf{x}^T = H\mathbf{a}^T.$$

Acest sistem are $m - r$ ecuaţii. În particular, un hiperplan este definit printr-o singură ecuaţie:

$$h_0x_0 + h_1x_1 + \ldots + h_{m-1}x_{m-1} = c.$$

Exemplul 5.13 *Un* 0 *- spaţiu afin cuprinde un singur punct. Există deci* 2^m 0 *- spaţii afine distincte:* $\{\mathbf{p_0}\}, \{\mathbf{p_1}\}, \ldots, \{\mathbf{p_{2^m-1}}\}$.

Similar, orice 1 *- spaţiu afin (sau "dreaptă") este o mulţime formată din două puncte*

$$\mathbf{a} + t\mathbf{b} \equiv \{\mathbf{a}, \mathbf{a} + \mathbf{b}\},$$

şi invers, orice mulţime de două puncte distincte formează un 1 *- spaţiu afin.*

Există deci $C_{2^m}^2$ 1 *- spaţii afine.*

Exemplul 5.14 *Fie* P_i *spaţiul afin definit de ecuaţia* $x_i = 1$. *Deci* P_i *este mulţimea punctelor* $\mathbf{p_k}$ *care au* 1 *pe poziţia* i. *De exemplu* $P_0 = \{\mathbf{p_1}, \mathbf{p_3}, \ldots, \mathbf{p_{2^m-1}}\}$.

Fiecare P_i *este un hiperplan şi - deoarece* $\mathbf{p_{2^i}}$ *are un singur* 1 *pe poziţia* i *şi* 0 *în rest, putem scrie*

$$P_i \equiv \mathbf{p_{2^i}} + C$$

unde C *este spaţiul liniar definit de ecuaţia* $x_i = 0$, *(deci tot un hiperplan).*

Propoziţia 5.5 *Există* $2(2^m - 1)$ *hiperplane.*

Demonstraţie: Deoarece se pot construi $2^m - 1$ ecuaţii cu coeficienţi binari (nu toţi nuli) şi variabile $x_0, x_1, \ldots, x_{m-1}$, Z_2^m are $2^m - 1$ subspaţii C de dimensiune $m - 1$. Fiecare din ele are 2^{m-1} puncte, deci vor exista $\dfrac{2^m}{2^{m-1}} = 2$ spaţii afine modulo C. q.e.d.

Exemplul 5.15 *Pentru* $i \neq j$, *intersecţia* $P_i \cap P_j$ *(mulţimea punctelor care au* 1 *pe poziţiile* i *şi* j*) este un* $(m - 2)$ *- spaţiu afin. Într-adevăr, dacă se ia* $\mathbf{a} = \mathbf{p_{2^i+2^j}}$, *avem*

$$P_i \cap P_j \equiv \mathbf{a} + C,$$

unde C *este determinat de ecuaţiile* $x_i = 0$, $x_j = 0$ *(deci* C *are dimensiunea* $m - 2$*).*

Definiţia 5.6 *Funcţia caracteristică a unui* r *- spaţiu afin* L *este cuvântul binar* $\mathbf{f_L} = f_{2^m-1} \ldots f_1 f_0$ *definit prin*

$$f_i = \begin{cases} 1 & \text{dacă} \quad \mathbf{p_i} \in L, \\ 0 & \text{altfel} \end{cases} \qquad (0 \leq i \leq 2^m - 1)$$

Funcţia caracteristică poate fi interpretată ca un polinom boolean $f_L(x_0, \ldots, x_{m-1})$. Din proprietăţile acestor polinoame rezultă

$$a_0 a_1 \ldots a_{m-1} \in L \qquad \Longleftrightarrow \qquad f_L(a_0, a_1, \ldots a_{m-1}) = 1$$

Observaţia 5.2

1. *Singurul m - spaţiu afin (Z_2^m) are funcţia caracteristică $\mathbf{1} = 11 \ldots 1$.*
2. *Un hiperplan P_i are funcţia $f_{P_i} = x_i$.*
3. *Fie L un hiperplan definit de ecuaţia $h_0 x_0 + \ldots h_{m-1} x_{m-1} = c$. Funcţia sa caracteristică va fi un polinom boolean de gradul 1, anume*

   $$f_L(x_0, \ldots, x_{m-1}) = h_0 x_0 + \ldots + h_{m-1} x_{m-1} + c + 1$$

   Această relaţie rezultă din faptul că un punct $a_0 \ldots a_{m-1}$ este în plan dacă şi numai dacă $h_0 a_0 + \ldots + h_{m-1} a_{m-1} = c$, adică $f_L(a_0, \ldots, a_{m-1}) = 1$.
4. *Pentru două spaţii afine L, L', funcţia caracteristică a intersecţiei $L \cap L'$ este $\mathbf{f_L f_{L'}}$.*

   Astfel, pentru $P_i \cap P_j$ (care este un $(m-2)$ - spaţiu afin) funcţia caracteristică este $\mathbf{x_i x_j}$.

   Mai general, polinomul boolean $\mathbf{x_{i_1} x_{i_2} \ldots x_{i_s}}$ este funcţia caracteristică a unui $(m-s)$ - spaţiu afin.

Teorema 5.4 *Funcţia caracteristică a unui r - spaţiu afin este un polinom boolean de gradul $m - r$.*

Demonstraţie: Un r - spaţiu afin L este definit ca soluţia sistemului de ecuaţii $H\mathbf{x}^T = \mathbf{c}^T$, sau, detaliind,

$$\sum_{j=0}^{m-1} h_{ij} x_j = c_i, \qquad 1 \le i \le m - r.$$

Aceste ecuaţii se pot scrie

$$\sum_{j=0}^{m-1} h_{ij} x_j + c_i + 1 = 1, \qquad 1 \le i \le m - r.$$

Atunci, polinomul boolean de grad $m - r$

$$f(x_0, \ldots, x_{m-1}) = \prod_{i=1}^{m-r} \left(\sum_{j=0}^{m-1} h_{ij} x_j + c_i + 1 \right)$$

poate f considerat funcţia caracteristică a lui L. q.e.d.

Definiţia 5.7 *$\mathcal{RM}(r, m)$ este codul liniar generat de toate funcţiile caracteristice ale spaţiilor afine de dimensiune cel puţin $m - r$ în geometria euclidiană m - dimensională peste Z_2.*

Faptul că aceasta coincide cu definiţia anterioară a codurilor Reed - Muller rezultă din construţia spaţiilor afine: $\mathcal{RM}(r,m)$ conţine toate funcţiile caracteristice ale s - spaţiilor afine, unde $s \geq m-r$. Faptul că aceste funcţii generează tot spaţiul $\mathcal{RM}(r,m)$ rezultă din Observaţia 5.2, punctul 4.

Exemplul 5.16 *Codul $\mathcal{RM}(1,3)$ este generat de funcţiile caracteristice ale tuturor planelor. Orice astfel de funcţie este un polinom de trei variabile de gradul 1.*

Codul $\mathcal{RM}(2,3)$ este generat de funcţiile caracteristice ale tuturor planelor şi liniilor. Cum o linie este intersecţia a două plane, funcţia sa caracteristică este produsul a două polinoame de gradul 1, deci un polinom de gradul 2 (cuvânt - cod din $\mathcal{RM}(2,3)$).

5.4 Exerciţii

5.1 *Ce polinom boolean are ultima linie a tabelei de adevăr:*

10100110 1010011010100110 0101001110011100

5.2 *Determinaţi tabela de adevăr a polinomului boolean $\mathbf{1}+\mathbf{x_0}+\mathbf{x_1x_2}$:*

1. *Ca funcţie de trei variabile;*
2. *Ca funcţie de patru variabile.*

5.3 *Demonstraţi afirmaţia $(*)$.*

5.4 *Găsiţi un $(15,5)$ - cod liniar binar corector de 3 erori (folosiţi un cod $R-M$ relaxat).*

5.5 *Fiind dat codul $\mathcal{RM}(1,3)$, codificaţi toate mesajele de informaţie posibile.*
Acelaşi lucru pentru codul $\mathcal{RM}(2,3)$.

5.6 *Demonstraţi Teorema 5.2*

5.7 *Construiţi matricile generatoare $G(1,3)$, $G(2,3)$, $G(r,4), r=0,1,2$.*

5.8 *Demonstraţi că $G(r,m)$ are $k+1$ linii, unde k este dat de Propoziţia 20.2*

5.9 *Calculaţi numărul de 2 - spaţii afine în geometria euclidiană peste Z_2^m.*

5.10 *Este orice funcţie booleană funcţia caracteristică a unui anumit spaţiu afin ? Caracterizaţi astfel de funcţii.*

5.11 *Orice funcţie caracteristică a unui $(r+1)$ - spaţiu afin aparţine codului dual lui $\mathcal{RM}(r,m)$.*

5.12 *Să se arate că $\mathcal{RM}(2,5)$ este auto - dual. Să se determine toate codurile $\mathcal{RM}$ auto - duale.*

Capitolul 6

Decodificarea codurilor Reed - Muller

Avantajul principal al codurilor Reed - Muller constă în facilitatea decodificării, facilitate bazată pe o tehnică diferită de cea de până acum. Această tehnică, numită *decodificare majoritară* nu apelează la ideea de sindrom, ci corectează direct biţii modificaţi, folosind diverse proprietăţi ale cuvântului recepţionat.

6.1 Decodificarea majoritară

Să prezentăm pe scurt principiile generale ale decodificării majoritare.

Vom începe cu un exemplu foarte simplu:

Exemplul 6.1 *Fie codul binar cu repetiţie de lungime* $2n+1$:

$$C = \{\underbrace{00\ldots0}_{2n+1}, \underbrace{11\ldots1}_{2n+1}\}.$$

Ca sistem de ecuaţii de control se poate lua

$$\begin{cases} x_1 + x_2 & = & 0 \\ x_1 + x_3 & = & 0 \\ & \vdots & \\ x_1 + x_{2n+1} & = & 0 \end{cases}$$

Dacă se recepţionează cuvântul $\mathbf{y} = \mathbf{x} + \mathbf{e}$*, la înlocuirea lui în sistem, acesta devine*

$$y_1 + y_i = e_1 + e_i, \quad (2 \leq i \leq 2n+1)$$

Dacă mai mult de n *din cele* $2n$ *expresii* $y_1 + y_i$ *iau valoarea* 1*, aceasta înseamnă că:*

- *Au apărut mai puţin de* n *erori, printre care şi pe prima poziţie (*$e_1 = 1$*), sau*
- *Au apărut mai mult de* n *erori, dar nu pe prima poziţie (*$e_1 = 0$*).*

Din aceste două variante, prima este mai probabilă. Deci valoarea majoritară pe care o iau cele n *valori* $y_1 + y_i$ *va fi valoarea lui* e_1*.*

Definiţia 6.1 *Fie $A_{n,k}$ un cod liniar. Un set de ecuaţii de control pentru $A_{n,k}$ este ortogonal pe mulţimea de poziţii $P \subset \{1, 2, \ldots, n\}$ dacă şi numai dacă:*

1. *$\forall i \in P$, termenul x_i apare (cu coeficient nenul) în fiecare ecuaţie;*

2. *$\forall i \notin P$, termenul x_i apare cel mult într-o ecuaţie.*

Decodificarea se realizează caracter cu caracter, după următorul procedeu:

Fie $\mathbf{a} = a_1 \ldots a_n$ cuvântul recepţionat şi i $(1 \leq i \leq n)$ o poziţie. Se construieşte mulţimea maximală de ecuaţii de control ortogonale pe poziţia i. Fie e_i valoarea obţinută în majoritatea acestor ecuaţii. Al i-lea caracter decodificat este $a_i + e_i$.

Dacă fiecare ecuaţie de control ortogonală pe poziţia i se scrie

$$a_i = f(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n),$$

atunci decoddificarea majoritară revine la a decodifica pe a_i prin valoarea care apare cel mai frecvent în evaluăriule expresiilor f.

Exemplul 6.2 *Fie dualul $(15, 11)$ - codului Hamming cu matricea de control*

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

Pentru fiecare pereche de coloane distincte din H există o a treia coloană astfel încât suma celor trei coloane este $\mathbf{0}$.

Deci, fiecare cuvânt de pondere 3 al codului Hamming dă o ecuaţie de control cu trei termeni pentru codul dual; în acest fel se obţine pentru fiecare caracter x_i câte un sistem format din 7 ecuaţii ortogonale pe poziţia i. Astfel,

- pentru x_1 ele sunt: $x_1 = x_2 + x_3 = x_4 + x_5 = x_6 + x_7 = x_8 + x_9 = x_{10} + x_{11} = x_{12} + x_{13} = x_{14} + x_{15}$;

- pentru x_2 : $x_2 = x_1 + x_3 = x_4 + x_6 = x_5 + x_7 = x_8 + x_{10} = x_9 + x_{11} = x_{12} + x_{14} = x_{13} + x_{15}$ etc.

Codul dual are distanţa minimă $d = 8$; deci el poate corecta maxim 3 erori.

Dacă în mesajul primit $\mathbf{x}$ apar cel mult 3 erori, atunci un sistem de ecuaţii ortogonal pe x_1 va avea cel mult trei ecuaţii care să dea pentru x_1 o valoare greşită şi cel puţin cinci cu valoarea calculată corect. Valoarea găsită majoritar va fi cea în care se decodifică primul caracter.

Procedeul se reia pentru $x_2, x_3, \ldots$.

Să presupunem de exemplu că s-a recepţionat mesajul 001011110101011. Pentru decodificarea primului caracter vom calcula sumele

$$0 + 1,\ 0 + 1,\ 1 + 1,\ 1 + 0,\ 1 + 0,\ 1 + 0,\ 1 + 1.$$

Se obţin 5 valori de 1 şi două de 0; deci, primul simbol este 1.

Procedând similar pentru fiecare poziţie, se ajunge la cuvântul 10101011010101010.

Această metodă este nu numai uşor de implementat, dar are adesea şi alte completări (gen "bitul de control al parităţii") care fac posibilă corectarea mai multor erori.

În această prelegere vom construi trei algoritmi de decodificare a codurilor $R-M$. Doi din ei sunt bazaţi pe modurile de reprezentare (geometric şi algebric) ale acestor coduri; al treilea este un algoritm care foloseşte definirea recursivă a codurilor $R-M$ şi este prezentat numai pentru cazul $r=1$.

Toţi algoritmii sunt foarte uşor de implementat, utilizarea unuia sau a altuia depinzând doar de criterii particulare în alegerea parametrilor codului.

6.2 Algoritm geometric de decodificare majoritară a codurilor $R-M$

Deoarece distanţa unui cod $\mathcal{RM}(r,m)$ este $d=2^{m-r}$, el va fi capabil să corecteze până la $2^{m-r-1}-1$ erori. În cele ce urmează vom presupune că s-a recepţionat cuvântul $\mathbf{y}=y_{2^m-1}\dots y_1y_0$, în care au fost modificate maxim $2^{m-r-1}-1$ poziţii. Problema este de a determina pentru fiecare i $(0\le i\le 2^m-1)$ dacă bitul y_i trebuie corectat sau nu.
Sau – altfel spus – de a vedea dacă poziţia lui $\mathbf{y}$ corespunzătoare 0 - spaţiului afin $\{\mathbf{p_i}\}$ trebuie sau nu corectată.

Pentru a folosi proprietăţile codurilor Reed - Muller exprimate de spaţiile afine, vom reformula totul în maniera următoare:

Pentru fiecare s - spaţiu afin L $(0\le s\le r+1)$ vom cerceta dacă poziţiile cuvântului recepţionat $\mathbf{y}$ corespunzătoare punctelor lui L (adică acei biţi y_i pentru care $\mathbf{p_i}\in L$) sunt modificaţi sau nu.

Vom da întâi câteva noţiuni şi notaţii ajutătoare pentru a simplifica demonstraţiile.

Definiţia 6.2 *Fie G un grup, C un subgrup al său şi $x\in G$. Se numeşte "subgrup modulo C" al lui G mulţimea*

$$C_x=x+C=\{x+a|a\in C\}.$$

Lema 6.1 *Subgrupurile unui grup modulo un subgrup arbitrar C, au următoarele proprietăţi:*

1. $\forall a\in G,\ \exists x\in G,\ a\in A_x$;
2. *Dacă $x\neq y$ atunci $C_x\cap C_y=\emptyset$ sau $C_x=C_y$;*
3. *Dacă $a,b\in C_x$ atunci $a-b\in C$;*
4. $\forall x,\ card(C_x)=card(C)$.

Demonstraţie: Exerciţiu.

Teorema principală a acestui paragraf este:

Teorema 6.1 *Orice s - spaţiu afin din geometria binară m - dimensională este conţinut în exact $2^{m-s}-1$ $(s+1)$ - spaţii afine distincte. În plus, orice punct din afara lui L se află în unul şi numai unul din aceste $(s+1)$ - spaţii afine.*

Demonstraţie: **I**: Să arătăm întâi că orice s - subspaţiu liniar $C\subseteq Z_2^m$ este conţinut în exact $2^{m-s}-1$ subspaţii distincte de dimensiune $s+1$.

Orice $(s+1)$ - spaţiu care conţine C este de forma

$$\overline{C}=C+t\mathbf{b}\equiv\{\mathbf{a}+t\mathbf{b}\mid\mathbf{a}\in C,\ t=0,1\},$$

unde $\mathbf{b} \notin C$ este un punct arbitrar fixat. Aceasta rezultă imediat din faptul că orice bază a lui C poate fi extinsă la o bază a lui $\overline{C}$.

Pentru două puncte $\mathbf{b}, \mathbf{b}' \notin C$, spaţiile liniare $C + t\mathbf{b}$ şi $C + t\mathbf{b}'$ coincid dacă şi numai dacă $\mathbf{b}$ şi $\mathbf{b}'$ sunt în acelaşi subspaţiu modulo C (Lema 15.1).

Din aceeaşi lemă rezultă că sunt în total $\frac{2^m}{2^s}$ subspaţii modulo C. Unul este chiar C, iar celelalte conţin numai puncte dinafara lui C. Deci – înafară de C – există $2^{m-s} - 1$ spaţii distincte $C + t\mathbf{b}$ pentru $\mathbf{b} \notin C$.

II: Orice s - spaţiu afin $L = \mathbf{a} + C \quad (dimC = s)$ este conţinut în $2^{m-s} - 1$ $(s+1)$ - spaţii afine distincte de forma $\mathbf{a} + \overline{C}$.
Cu **I**, orice spaţiu care conţine L este de forma $\mathbf{b} + \overline{C}$. Deoarece L este de forma $\mathbf{a} + C$, rezultă că $\mathbf{a} \in \mathbf{b} + \overline{C}$, deci $\mathbf{a} - \mathbf{b} \in \overline{C}$, de unde rezultă (Lema 15.1) că punctele $\mathbf{a}, \mathbf{b}$ sunt în acelaşi subspaţiu modulo C.

III: Orice punct $\mathbf{b} \notin L = \mathbf{a} + C$ este într-un $(s+1)$ - spaţiu afin care conţine L, anume $\mathbf{a} + \overline{C}$ unde $\overline{C} = C + t(\mathbf{b} - \mathbf{a})$.

Într-adevăr, alegând $t = 1$ şi $\mathbf{0} \in C$, avem $\quad \mathbf{b} = \mathbf{a} + [\mathbf{0} + (\mathbf{b} - \mathbf{a})]$.

Pentru a verifica că $\overline{C}$ are dimensiunea $s + 1$ este suficient de arătat că $\mathbf{b} - \mathbf{a} \notin C$; dacă prin absurd $\mathbf{b} - \mathbf{a} \in C$, atunci $\mathbf{a} + (\mathbf{b} - \mathbf{a}) = \mathbf{b} \in C$, contradicţie.

În final, ar mai trebui arătat că acest $(s + 1)$ - spaţiu afin care îl conţine pe $\mathbf{b}$ este unic. Orice $(s+1)$ - spaţiu afin care conţine $\mathbf{a} + C$ are forma $\mathbf{a} + \overline{C}$ unde $dim(\overline{C}) = s + 1$.

Dacă $\mathbf{b} \in \mathbf{a} + \overline{C}$, atunci $\mathbf{b} - \mathbf{a} \in \overline{C}$, deci $\overline{C}$ conţine spaţiul liniar $C + (\mathbf{b} - \mathbf{a})$. Cum ambele spaţii au aceeaşi dimensiune $(s + 1)$, ele coincid. q.e.d.

Corolarul 6.1 *Dacă numărul de erori din cuvântul recepţionat* $\mathbf{y}$ *este* $t < 2^{m-r-1}$*, atunci pentru fiecare* s *- spaţiu afin* L $(0 \leq s \leq r)$ *majoritatea* $(s+1)$ *- spaţiilor afine care conţin* L *au aceeaşi paritate a erorilor ca* L*.*

Demonstraţie: Conform Teoremei 20.1, un s-spaţiu afin L este inclus în $2^{m-r} - 1 > 2t$ $(s + 1)$ - spaţii afine L', fiecare L' fiind unic determinat de un punct din afara lui L. Să considerăm toate punctele $\mathbf{p_i} \notin L$ pentru care caracterul y_i este modificat. Lor le corespund cel mult $t \quad (s+1)$ - spaţii afine L'. Toate celelalte spaţii L' rămase au proprietatea că nu conţin nici un punct $\mathbf{p_i}$ din afara lui L pentru care y_i este greşit. Deci aceste spaţii au aceeaşi paritate de erori ca şi L, iar numărul lor este cel puţin $(2^{m-r} - 1) - t > t$, deci majoritar. q.e.d.

Algoritm de decodificare pentru $\mathcal{RM}(r, m)$:

1. (*Iniţializare*): La recepţionarea unui cuvânt $\mathbf{y} \in Z_2^m$ se consideră toate $(r + 1)$ - spaţiile afine L. Un spaţiu L se numeşte *impar* dacă $\mathbf{yf_L} = 1$ (reamintim, $\mathbf{f_L}$ este funcţia caracteristică a spaţiului afin L). În caz contrar L este *par*.
2. (*Inducţie*): Pentru fiecare $s = r, r-1, \ldots, 0$ unde $(s+1)$ - spaţiile afine au fost definite drept pare sau impare, se consideră toate s - spaţiile afine. Un s - spaţiu afin L este par dacă majoritatea $(s+1)$ - spaţiilor afine care conţin pe L sunt pare; altfel L este impar.
3. (*Final*): Pentru $i = 0, 1, \ldots, 2^m - 1$, se corectează y_i dacă şi numai dacă 0 - spaţiul afin $\{\mathbf{p_i}\}$ este impar.

Tabelul 6.1: Primul pas al decodificării lui 11101010

Plan	Paritate	Plan	Paritate
$\{\mathbf{p_1}, \mathbf{p_3}, \mathbf{p_5}, \mathbf{p_7}\}$	par	$\{\mathbf{p_1}, \mathbf{p_3}, \mathbf{p_4}, \mathbf{p_6}\}$	impar
$\{\mathbf{p_2}, \mathbf{p_3}, \mathbf{p_6}, \mathbf{p_7}\}$	impar	$\{\mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}, \mathbf{p_5}\}$	par
$\{\mathbf{p_4}, \mathbf{p_5}, \mathbf{p_6}, \mathbf{p_7}\}$	impar	$\{\mathbf{p_0}, \mathbf{p_3}, \mathbf{p_4}, \mathbf{p_7}\}$	par
$\{\mathbf{p_0}, \mathbf{p_2}, \mathbf{p_4}, \mathbf{p_6}\}$	impar	$\{\mathbf{p_0}, \mathbf{p_2}, \mathbf{p_5}, \mathbf{p_7}\}$	par
$\{\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_4}, \mathbf{p_5}\}$	par	$\{\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_6}, \mathbf{p_7}\}$	impar
$\{\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}\}$	par	$\{\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_4}, \mathbf{p_7}\}$	par
$\{\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_5}, \mathbf{p_6}\}$	impar	$\{\mathbf{p_0}, \mathbf{p_3}, \mathbf{p_5}, \mathbf{p_6}\}$	impar

Tabelul 6.2: Al doilea pas al decodificării lui 11101010

Dreaptă	Paritate	Dreaptă	Paritate	Dreaptă	Paritate
$\{\mathbf{p_0}, \mathbf{p_1}\}$	par	$\{\mathbf{p_1}, \mathbf{p_5}\}$	par	$\{\mathbf{p_3}, \mathbf{p_5}\}$	par
$\{\mathbf{p_0}, \mathbf{p_2}\}$	par	$\{\mathbf{p_1}, \mathbf{p_6}\}$	par	$\{\mathbf{p_3}, \mathbf{p_6}\}$	impar
$\{\mathbf{p_0}, \mathbf{p_3}\}$	par	$\{\mathbf{p_1}, \mathbf{p_7}\}$	par	$\{\mathbf{p_3}, \mathbf{p_7}\}$	par
$\{\mathbf{p_0}, \mathbf{p_4}\}$	par	$\{\mathbf{p_2}, \mathbf{p_3}\}$	par	$\{\mathbf{p_4}, \mathbf{p_5}\}$	par
$\{\mathbf{p_0}, \mathbf{p_5}\}$	par	$\{\mathbf{p_2}, \mathbf{p_4}\}$	par	$\{\mathbf{p_4}, \mathbf{p_6}\}$	impar
$\{\mathbf{p_0}, \mathbf{p_6}\}$	impar	$\{\mathbf{p_2}, \mathbf{p_5}\}$	par	$\{\mathbf{p_4}, \mathbf{p_7}\}$	par
$\{\mathbf{p_0}, \mathbf{p_7}\}$	par	$\{\mathbf{p_2}, \mathbf{p_6}\}$	impar	$\{\mathbf{p_5}, \mathbf{p_6}\}$	impar
$\{\mathbf{p_1}, \mathbf{p_2}\}$	par	$\{\mathbf{p_2}, \mathbf{p_7}\}$	par	$\{\mathbf{p_5}, \mathbf{p_7}\}$	par
$\{\mathbf{p_1}, \mathbf{p_3}\}$	par	$\{\mathbf{p_3}, \mathbf{p_4}\}$	par	$\{\mathbf{p_6}, \mathbf{p_7}\}$	impar
$\{\mathbf{p_1}, \mathbf{p_4}\}$	par				

Exemplul 6.3 *Să considerăm* $\mathcal{RM}(1,3)$ *în care s-a recepţionat cuvântul*

$$\mathbf{y} = 11101010$$

La primul pas trebuie să decidem care plane (2-spaţii afine) sunt pare şi care sunt impare. De exemplu planul $L = \{\mathbf{p_1}, \mathbf{p_3}, \mathbf{p_5}, \mathbf{p_7}\}$ *este par deoarece*

$$\mathbf{yf_L} = 11101010 \cdot 10101010 = 0$$

(vezi Prelegerea V, Paragraful 5.3.1*). Se obţine Tabelul 8.1.*

La pasul următor trebuie să decidem paritatea fiecărei drepte (1-spaţiu afin). Orice dreaptă este inclusă în $2^{3-1} - 1 = 3$ *plane distincte.*

De exemplu, linia $\{\mathbf{p_0}, \mathbf{p_1}\}$ *este conţinută în trei plane:*

$\{\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_4}, \mathbf{p_5}\}$ *(par),* $\{\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}\}$ *(par),* $\{\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_6}, \mathbf{p_7}\}$ *(impar)*

deci, prin majoritate, $\{\mathbf{p_0}, \mathbf{p_1}\}$ *este pară.*

În mod similar se efectuează calculele pentru toate liniile, obţinându-se Tabelul 8.2.

Se poate trece acum la corectarea erorilor. $\mathcal{RM}(1,3)$ *poate corecta maxim* $2^{m-r-1}-1 = 2^{3-1-1} - 1 = 1$ *erori.* $\{\mathbf{p_0}\}$ *este conţinut în şapte linii: şase pare şi una impară, deci* y_0 *este corect etc. Singurul bit care trebuie corectat este* y_6 *deoarece* $\{\mathbf{p_6}\}$ *este în şapte linii, din care şase impare. Cuvântul - cod trimis a fost deci* 10101010*.*

Algoritmul de decodificare se bazează pe următoarea observaţie:

Fie codul $\mathcal{RM}(r,m)$ şi $\mathbf{y} \in Z_2^m$ un cuvânt recepţionat. Un $(r+1)$ - spaţiu afin L este par $\iff$ $\mathbf{yf_L} = 0$.

Într-adevăr, dacă $\mathbf{y}$ este cuvânt - cod, atunci $\mathbf{yf_L} = 0$ (Prelegerea V, Exerciţiul 5.11). Acum, dacă au fost perturbate un număr par de caractere din $\mathbf{y}$, corespunzătoare unor puncte din L, valoarea produsului scalar $\mathbf{yf_L}$ nu se modifică (atenţie ! se lucrează într-un corp de caracteristică 2). Pentru un număr impar de poziţii perturbate, valoarea produsului scalar este 1.

6.3 Algoritm algebric de decodificare majoritară

Înafară de algoritmul prezentat anterior, care se bazează pe reprezentarea geometrică a codurilor $R-M$, se poate da şi o variantă algebrică de decodificare, folosind direct definiţia decodificării majoritare.

În cele ce urmează să considerăm un cod $\mathcal{RM}(r,m)$ fixat. El are $k = 1+C_m^1+\ldots+C_m^r$ simboluri de informaţie şi lungimea $n = 2^m$.

În matricea $U_{m,n} = \begin{pmatrix} \mathbf{x_0} \\ \mathbf{x_1} \\ \vdots \\ \mathbf{x_{m-1}} \end{pmatrix}$ care conţine pe coloane toate elementele spaţiului liniar Z_2^m, vom nota cu $\mathbf{u_i}$ $(0 \le i \le m-1)$ coloanele care reprezintă baza (naturală) a lui Z_2^m.

Deci, orice coloană $\mathbf{w_j}$ $(0 \le j \le n-1)$ din Z_2^m se poate scrie

$$\mathbf{w_j} = \sum_{i=0}^{m-1} t_{ij}\mathbf{u_i}, \qquad t_{ij} \in \{0,1\}.$$

Orice cuvânt $\mathbf{v} \in Z_2^n$ poate fi considerat *indicatorul* unei mulţimi de cuvinte din Z_2^m: acele coloane din $U_{m,n}$ care corespund poziţiilor din $\mathbf{v}$ unde se află valoarea 1.

Exemplul 6.4 *Pentru orice $j = 0,\ldots,n-1$, cuvântul $\mathbf{e_j} = 00\ldots010\ldots0$, cu 1 pe poziţia j corespunde cuvântului de pe coloana j din $U_{m,n}$.*

Cuvântul $\mathbf{1} = 11\ldots1$ corespunde întregului spaţiu Z_2^m.

Propoziţia 6.1

$$\mathbf{e_j} = \prod_{i=0}^{m-1} \left[\mathbf{x_i} + (1+t_{ij})\mathbf{1}\right], \quad \forall j,\ 0 \le j \le 2^m - 1.$$

Demonstraţie: Să arătăm întâi că $\mathbf{x_i} + (1+t_{ij})\mathbf{1}$ reprezintă indicatorul acelor coloane din U care au aceeaşi componentă i ca şi coloana $\mathbf{w_j} = \sum_{i=0}^{m-1} t_{ij}\mathbf{u_i}$.

Într-adevăr:

1. Dacă $t_{ij} = 1$, atunci $\mathbf{x_i}+(1+t_{ij})\mathbf{1} = \mathbf{x_i}+(1+1)\mathbf{1} = \mathbf{x_i}$ care este indicatorul mulţimii coloanelor cu 1 (ca şi $\mathbf{w_j}$) pe linia i.

2. Dacă $t_{ij} = 0$, atunci $\mathbf{x_i} + (1 + t_{ij})\mathbf{1} = \mathbf{x_i} + \mathbf{1}$ care are componentele lui $\mathbf{x_i}$ cu 0 şi 1 permutate, fiind deci indicatorul mulţimii coloanelor care au 0 (ca şi $\mathbf{w_j}$) pe linia i.

Cuvântul $\prod_{i=0}^{m-1} [\mathbf{x_i} + (1 + t_{ij})\mathbf{1}]$ reprezintă indicatorul mulţimii acelor coloane din matricea $U_{m,n}$ care au toate componentele egale cu cele ale coloanei $\mathbf{w_j}$. Cum coloanele matricii U sunt distincte, această mulţime este chiar $\{\mathbf{w_j}\}$, al cărei indicator este $\mathbf{e_j}$. q.e.d.

Fie mesajul de informaţie $a_1a_2 \ldots a_k \in Z_2^k$. Ele se codifică cu ajutorul $(2^m, k)$ - codului Reed - Muller în cuvântul - cod

$$\mathbf{f} = (f_0, f_1, \ldots, f_{n-1}) =$$
$$a_1\mathbf{1} + a_2\mathbf{x_0} + \ldots + a_{m+1}\mathbf{x_{m-1}} + a_{m+2}\mathbf{x_0x_1} + \ldots + a_k\mathbf{x_{m-r}x_{m-r+1}} \ldots \mathbf{x_{m-1}}.$$

Putem enunţa acum rezultatul principal pe baza căruia se construieşte algoritmul algebric de decodificare majoritară pentru codurile $R - M$.

Teorema 6.2 *Pentru fiecare componentă a_s din mesajul de informaţie care se înmulţeşte – la codificare – cu un produs de forma $\mathbf{x_{i_1}x_{i_2}} \ldots \mathbf{x_{i_r}}$, există 2^{m-r} sume disjuncte (cu termeni diferiţi) egale cu a_s, conţinând fiecare 2^r componente ale cuvântului - cod corespunzător $\mathbf{f} = f_0f_1 \ldots f_{n-1}$.*

Demonstraţie: Ţinând cont de Propoziţia 6.1 şi de faptul că $\mathbf{e_0}, \mathbf{e_1}, \ldots, \mathbf{e_{n-1}}$ constituie de asemenea o bază pentru Z_2^n, putem scrie:

$$\mathbf{f} = (f_0, f_1, \ldots, f_{n-1}) = \sum_{j=0}^{n-1} f_j\mathbf{e_j} = \sum_{j=0}^{n-1} f_j \prod_{i=0}^{m-1} [\mathbf{x_i} + (1 + t_{ij})\mathbf{1}].$$

Prin urmare, ca un produs de forma $\mathbf{x_{i_1}} \ldots \mathbf{x_{i_t}}$ să fie în această sumă, trebuie ca $t_{ij} = 0$ pentru $i \notin \{i_1, i_2 \ldots, i_t\}$. Dacă notăm

$$C(i_1, \ldots, i_j) = \{p \mid p = \sum_{i=0}^{m-1} t_{ip}2^i, t_{ip} = 0 \text{ dacă } i \notin \{i_1, \ldots, i_j\}\} =$$
$$= \{p \mid p = t_{i_1p}2^{i_1} + t_{i_2p}2^{i_2} + \ldots + t_{i_jp}2^{i_j},\ t_{i_sp} \in \{0, 1\},\ s = 1, 2, \ldots, j\},$$

vom avea

$$\mathbf{f} = (f_0, f_1, \ldots, f_{n-1}) = \sum_{t=1}^{r} \sum_{i_1<i_2<\ldots<i_t} \left(\sum_{j \in C(i_1,\ldots,i_t)} f_j \right) \mathbf{x_{i_1}x_{i_2}} \ldots \mathbf{x_{i_t}}$$

Pe de-altă parte,

$$\mathbf{f} = (f_0, \ldots, f_{n-1}) = a_1\mathbf{1} + a_2\mathbf{x_0} + \ldots + a_{m+1}\mathbf{x_{m-1}} + a_{m+2}\mathbf{x_0x_1} + \ldots + a_k\mathbf{x_{m-r}} \ldots \mathbf{x_{m-1}}.$$

Prin identificare, rezultă că elementul a_s, coeficient al produsului $\mathbf{x_{i_1}x_{i_2}} \ldots \mathbf{x_{i_r}}$ este

$$a_s = \sum_{j \in C(i_1,\ldots,i_r)} f_j \tag{1}$$

Fie $z \notin \{i_1, \ldots, i_r\}$. Deoarece baza codului $\mathcal{RM}(r,m)$ nu conţine monoame de forma $\mathbf{x_{i_1}} \ldots \mathbf{x_{i_r}} \mathbf{x_{i_{r+1}}}$, vom obtine prin codificare

$$\sum_{j \in C(i_1,\ldots,i_r,z)} f_j = 0. \tag{2}$$

Însă,

$$C(i_1, \ldots, i_r, z) = C(i_1, \ldots, i_r) \cup [C(i_1, \ldots, i_r) + 2^z],$$

unde s-a notat $C + \alpha = \{x + \alpha |\ x \in C\}$. Într-adevăr, putem scrie

$$\begin{gathered} C(i_1, \ldots, i_r, z) = \{j |\ j = t_{i_1 j} 2^{i_1} + \ldots + t_{i_r j} 2^{i_r} + t_{zj} 2^z\} = \\ = \{j |\ j = t_{i_1 j} 2^{i_1} + \ldots + t_{i_r j} 2^{i_r}\} \cup \{j |\ j = t_{i_1 j} 2^{i_1} + \ldots + t_{i_r j} 2^{i_r} + 2^z\} = \\ = C(i_1, \ldots, i_r) \cup [C(i_1, \ldots, i_r) + 2^z], \end{gathered}$$

cele două mulţimi ale reuniunii fiind disjuncte.

Deci relaţia (2) se scrie $\sum_{j \in C(i_1,\ldots,i_r)} f_j + \sum_{j \in C(i_1,\ldots,i_r)+2^z} f_j = 0$ şi – ţinând cont de (1), avem

$$a_s + \sum_{j \in C(i_1,\ldots,i_r)+2^z} f_j = 0 \tag{3}$$

Din (1) şi (3) rezultă următoarea afirmaţie: dacă $m = r + 1$, pentru orice simbol de informaţie a_s înmulţit la codificare cu un produs de forma $\mathbf{x_{i_1}} \ldots \mathbf{x_{i_r}}$ există $2^{m-r} = 2$ sume disjuncte ((1) şi (3)), fiecare de câte 2^r componente ale lui $\mathbf{f}$.

Aceste două relaţii se numesc *relaţii de determinare* a componentei a_s.

Fie acum $z_1, z_2 \notin \{i_1, \ldots, i_r\}$. Deoarece baza codului nu are produse de forma $\mathbf{x_{i_1}} \ldots \mathbf{x_{i_r}} \mathbf{x_{i_{r+1}}} \mathbf{x_{i_{r+2}}}$, vom avea

$$\sum_{j \in C(i_1,\ldots,i_r,z_1,z_2)} f_j = 0. \tag{4}$$

Dar

$$C(i_1, \ldots, i_r, z_1, z_2) = C(i_1, \ldots, i_r) \cup [C(i_1, \ldots, i_r) + 2^{z_1}] \cup [C(i_1, \ldots, i_r) + 2^{z_2}] \cup [C(i_1, \ldots, i_r) + 2^{z_1} + 2^{z_2}]$$

Deci (4) se descompune în

$$\sum_{j \in C(i_1,\ldots,i_r)} f_j + \sum_{j \in C(i_1,\ldots,i_r)+2^{z_1}} f_j + \sum_{j \in C(i_1,\ldots,i_r)+2^{z_2}} f_j + \sum_{j \in C(i_1,\ldots,i_r)+2^{z_1}+2^{z_2}} f_j = 0$$

ceea ce – ţinând cont de (1) şi (3) devine $a_s + a_s + a_s + \sum_{j \in C(i_1,\ldots,i_r)+2^{z_1}+2^{z_2}} f_j = 0.$

deci

$$a_s = \sum_{j \in C(i_1,\ldots,i_r)+2^{t_1}+2^{t_2}} f_j \tag{5}$$

Am arătat astfel că, dacă $m = r + 2$, atunci există $2^{m-r} = 2^2 = 4$ sume ((1), de două ori (3) pentru z_1 respectiv z_2, şi (5)) care dau pe a_s, fiecare sumă conţinând 2^r componente ale lui $\mathbf{f}$. Deci, pentru a_s există în acest caz patru relaţii de determinare.

Teorema rezultă în continuare printr-un procedeu de inducţie finită. q.e.d.

Să vedem cum se foloseşte Teorema 11.1 la decodificarea majoritară a codurilor Reed - Muller:

După cum s-a văzut, pentru fiecare componentă a_s corespunzătoare unui produs de forma $\mathbf{x_{i_1}} \dots \mathbf{x_{i_r}}$ există 2^{m-r} sume disjuncte, conţinând fiecare 2^r componente ale lui $\mathbf{f}$ care – în absenţa erorilor – vor fi toate egale cu a_s.

Deoarece sumele sunt disjuncte (deci termenii lor sunt elemente diferite ale cuvântului - cod), o eroare singulară va afecta o singură relaţie de determinare a lui a_s. Prin urmare, dacă în transmiterea cuvântului - cod $\mathbf{f}$ apar cel mult $2^{m-r-1} - 1$ erori independente, putem determina – prin majoritate – pe a_s, identificându-l cu valoarea a *jumătate plus unu* din relaţiile de determinare corespunzătoare.

După ce se determină în acest mod coeficienţii tuturor produselor $\mathbf{x_{i_1}} \dots \mathbf{x_{i_r}}$, se scade din vectorul recepţionat combinaţia liniară

$$a_p \mathbf{x_0} \dots \mathbf{x_{r-1}} + \dots + a_k \mathbf{x_{m-r}} \dots \mathbf{x_{m-1}}.$$

În acest fel problema se reduce în continuare la decodificarea într-un cod de ordin $r - 1$ şi lungime 2^m; procedeul este similar, fiind determinate aici componentele de informaţie care se codifică prin înmulţire cu produse de forma $\mathbf{x_{j_1}} \dots \mathbf{x_{j_{r-1}}}$ $(j_1 < j_2 < \dots < j_{r-1})$.

Exemplul 6.5 *Fie codul* $\mathcal{RM}(2, 4)$. *El are lungimea* $n = 16$, *ordinul* $r = 2$, $k = 1 + C_4^1 + C_4^2 = 11$ *simboluri de informaţie şi* $n - k = 5$ *simboluri de control. Deci el este un* $(16, 11)$ *- cod liniar, capabil să corecteze cel mult o eroare (pentru că* $2^{4-2-1} - 1 = 1$*).*

Orice mesaj de informaţie $(a_1, a_2, \dots, a_{11}) \in Z_2^{11}$ *se codifică în cuvântul*

$\mathbf{f} = (f_0, \dots, f_{15}) = a_1\mathbf{1} + a_2\mathbf{x_0} + a_3\mathbf{x_1} + a_4\mathbf{x_2} + a_5\mathbf{x_3} + a_6\mathbf{x_0x_1} + a_7\mathbf{x_0x_2} + a_8\mathbf{x_0x_3} + a_9\mathbf{x_1x_2} + a_{10}\mathbf{x_1x_3} + a_{11}\mathbf{x_2x_3}$.

Să stabilim de exemplu relaţiile de determinare pentru a_9, *care se codifică prin înmulţire cu* $\mathbf{x_1x_2}$. *Trebuiesc calculate mulţimile:*

$$\begin{aligned} &C(1,2) = \{j |\ j = t_{1j}2^1 + t_{2j}2^2,\ \ t_{ij} \in \{0,1\}\} = \{0,2,4,6\} \\ &C(1,2) + 2^0 = \{1,3,5,7\} \\ &C(1,2) + 2^3 = \{8,10,12,14\} \\ &C(1,2) + 2^0 + 2^3 = \{9,11,13,15\}. \end{aligned}$$

Deci relaţiile de determinare corespunzătoare lui a_9 *devin*

$$\begin{aligned} a_9 &= f_0 + f_2 + f_4 + f_6 & \qquad a_9 &= f_1 + f_3 + f_5 + f_7 \\ a_9 &= f_8 + f_{10} + f_{12} + f_{14} & \qquad a_9 &= f_9 + f_{11} + f_{13} + f_{15} \end{aligned}$$

Dacă intervine o eroare în timpul transmiterii cuvântului - cod $\mathbf{f}$, *va fi afectată o singură relaţie de determinare, celelalte trei relaţii dând valoarea corectă a lui* a_9.

După ce s-a determinat valorile $a_6, a_7, a_8, a_9, a_{10}, a_{11}$, *se trece la etapa a doua a decodificării, scăzând din cuvântul recepţionat combinaţia*

$$a_6\mathbf{x_0x_1} + a_7\mathbf{x_0x_2} + a_8\mathbf{x_0x_3} + a_9\mathbf{x_1x_2} + a_{10}\mathbf{x_1x_3} + a_{11}\mathbf{x_2x_3}$$

şi aplicând cuvântului obţinut procedeul de corectare a erorilor pentru codul $\mathcal{RM}(1,4)$.

În final, a_1 *se decodifică în codul* $\mathcal{RM}(0,4)$*, ceea ce înseamnă că el este egal cu valoarea majoritară (*0 *sau* (1)*) luată de elementele* $f_0, f_1, \ldots, f_{15}$.

6.4 Decodificarea codurilor $\mathcal{RM}(1,m)$

Pentru cazul $r = 1$ se poate da şi un alt algoritm de decodificare, extrem de eficient. El foloseşte transformarea Hadamard pentru aflarea celui mai apropiat cuvânt - cod (în distanţă Hamming).

Vom da iniţial câteva noţiuni ajutătoare.

Definiţia 6.3 *Fie* $A_{m,n}, B_{p,q}$ *două matrici. Se defineşte produsul Kronecker* $A \times B$ *ca fiind matricea* $C_{mp,nq} = [a_{ij}B]$.

Acest produs este evident asociativ şi necomutativ; el va fi utilizat pe larg şi pentru alte construcţii.

Exemplul 6.6 *Fie* $H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$, $I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. *Atunci*

$$I_2 \times H = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}, \qquad H \times I_2 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}.$$

Se consideră şirul de matrici definit

$$H_m^i = I_{2^{m-i}} \times H \times I_{2^{i-1}}, \quad i = 1, 2, \ldots$$

unde H este matricea (Hadamard) definită în Exemplul 6.6.

Exemplul 6.7 *Fie* $m = 2$. *Atunci*

$$H_2^1 = I_2 \times H \times I_1 = I_2 \times H, \qquad H_2^2 = I_1 \times H \times I_2 = H \times I_2.$$

Exemplul 6.8 *Fie* $m = 3$. *Atunci:*

$$H_3^1 = I_4 \times H \times I_1 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix},$$

$$H_3^2 = I_2 \times H \times I_2 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{pmatrix},$$

$$H_3^3 = I_1 \times H \times I_4 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix}.$$

Modalitatea recursivă de construcţie a codurilor $\mathcal{RM}(r, m)$ (Prelegerea V, secţiunea 5.2) sugerează existenţa unui algoritm similar de decodificare. Acest algoritm există numai pentru $\mathcal{RM}(1, m)$ şi îl prezentăm fără a demonstra corectitudinea lui.

Algoritm de decodificare a codurilor $\mathcal{RM}(1, m)$:

Fie $\mathbf{y} \in Z_2^n$ $(n = 2^m)$ cuvântul recepţionat şi $G(1, m)$ matricea generatoare a codului.

1. Se înlocuieşte 0 cu -1 în $\mathbf{y}$; fie $\mathbf{y}'$ noul cuvânt;
2. Se calculează $\mathbf{y_1} = \mathbf{y}'H_m^1$, $\quad \mathbf{y_i} = \mathbf{y_{i-1}}H_m^i$ $(2 \le i \le m)$;
3. Se determină poziţia j a celei mai mari componente (în valoare absolută) a lui $\mathbf{y_m}$.

   Fie $v(j) \in Z_2^n$ reprezentarea binară a lui j pe m biţi (începând cu biţii cei mai semnificativi). Cuvântul decodificat este:

   $(1, v(j))$ dacă a j - a componentă a lui $\mathbf{y_m}$ este pozitivă,

   $(0, v(j))$ dacă a j - a componentă a lui $\mathbf{y_m}$ este negativă.

Exemplul 6.9 *Fie $m = 3$ şi $G(1, 3)$ matricea generatoare a codului $\mathcal{RM}(1, 3)$. Să presupunem că s-a recepţionat cuvântul* $\mathbf{y} = 10101011$.

Primul pas al algoritmului transformă acest cuvânt în $\mathbf{y}' = (1, -1, 1, -1, 1, -1, 1, 1)$.

Se calculează apoi:

$$\mathbf{y_1} = \mathbf{y}'H_3^1 = (0, 2, 0, 2, 0, 2, 2, 0)$$
$$\mathbf{y_2} = \mathbf{y_1}H_3^2 = (0, 4, 0, 0, 2, 2, -2, 2)$$
$$\mathbf{y_3} = \mathbf{y_2}H_3^3 = (2, 6, -2, 2, -2, 2, 2, -2).$$

Cea mai mare componentă a lui $\mathbf{y_3}$ *este* 6 *pe poziţia* 1. *Cum* $v(1) = 100$ *şi* $6 > 0$, *rezultă că a fost codificat mesajul de informaţie* 1100.

Dacă s-a recepţionat cuvântul $\mathbf{y} = 10001111$, *atunci* $\mathbf{y}' = (1, -1, -1, -1, 1, 1, 1, 1)$ *şi*

$$\mathbf{y_1} = \mathbf{y}'H_3^1 = (0, 2, -2, 0, 2, 0, 2, 0),$$
$$\mathbf{y_2} = \mathbf{y_1}H_3^2 = (-2, 2, 2, 2, 4, 0, 0, 0),$$
$$\mathbf{y_3} = \mathbf{y_2}H_3^3 = (2, 2, 2, 2, -6, 2, 2, 2).$$

Cea mai mare componentă a lui $\mathbf{y_3}$ *(în valoare absolută) este* -6 *aflată pe poziţia* 4. *Deoarece* $v(4) = 001$ *şi* $-6 < 0$, *mesajul transmis este* 0001.

6.5 Exerciţii

6.1 *În codul Hamming* $(15, 11)$ *să se decodifice mesajele*
111100000000000, 011101000111001.

6.2 *Demonstraţi Lema 15.1.*

6.3 *Să se decodifice cuvântul* 01111100 *în* $\mathcal{RM}(1, 3)$. *Verificaţi corectitudinea decodificării.*

6.4 *Să se decodifice* 0111100 *în* $\mathcal{RM}(2, 3)$.

6.5 *Să se scrie toate relaţiile de determinare din codul* $\mathcal{RM}(2, 4)$.

6.6 *Aceeaşi problemă pentru codul* $\mathcal{RM}(1, 3)$.

6.7 *Să se decodifice* 1111111011111111 *în* $\mathcal{RM}(2, 4)$.

6.8 *Arătaţi că orice hiperplan* L *în geometria euclidiană peste* Z_2^m *are drept complement* $Z_2^m \setminus L$ *tot un hiperplan.*

6.9 *În codul* $\mathcal{RM}(1, 3)$*, să se decodifice cuvintele*
01011110, 01100111, 00010100, 11001110.

6.10 *Să se calculeze* H_4^i *pentru* $i = 1, 2, 3, 4$.

6.11 *În codul* $\mathcal{RM}(1, 4)$*, să se decodifice cuvintele:*
1011011001101001, 1111000001011111.

Capitolul 7

Alte clase de coduri elementare

Înafara codurilor Hamming şi Reed - Muller se pot construi şi alte clase de coduri cu proprietăţi remarcabile. Vom prezenta numai câteva astfel de coduri – nu toate liniare – punând în evidenţă la fiecare clasă caracteristicile sale specifice.

7.1 Codurile MacDonald

Fie $A_{n,k}$ un cod liniar peste Z_2 şi G matricea sa generatoare.

Vom nota cu $M_{k,2^k-1}$ o matrice care conţine toate coloanele nenule posibile de k elemente, care se pot construi peste Z_2. Ordinea acestor coloane este arbitrară dar fixată. Coloana i din matricea M se va numi *coloană de tip* i. Pentru uşurinţa notaţiei se admite aserţiunea că tipul de coloană i este reprezentarea în binar a numărului zecimal i.

Deoarece codul $A_{n,k}$ nu este influenţat de operaţiile elementare efectuate asupra coloanelor din matricea generatoare (în particular permutarea de coloane), este suficient să definim acest cod indicând doar numărul coloanelor de fiecare tip care intră în componenţa matricii generatoare.

Se obţine astfel *reprezentarea modulară a codului* $A_{n,k}$:

$$N = (n_1, n_2, \ldots, n_{2^k-1})$$

unde n_i este numărul coloanelor de tip i care apar (sau nu) în matricea G,

$$\sum_{i=1}^{2^k-1} n_i = n.$$

Exemplul 7.1 *Să luăm* $k = 2,\ n = 3$ *şi matricea generatoare* $G = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$.

Avem matricea $M = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$ *(coloanele sunt reprezentările binare ale numerelor* $1, 2, 3$*).*

Reprezentarea modulară a codului generat de matricea G *este atunci* $N = (2, 0, 1)$ *(pentru că în* G *există două coloane de tipul* 1*, nici una de tipul* 2 *şi una de tipul* 3*).*

Să introducem matricea

$$K_{2^k-1,n} = M^T G$$

K are drept linii toate cuvintele nenule ale codului $A_{n,k}$.

Fie de asemenea matricea pătrată simetrică:

$$C_{2^k-1,2^k-1} = M^T M$$

Astfel, pentru Exemplul 19.4, $K = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$, $C = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$.

Teorema 7.1 *O listă a ponderilor cuvintelor codului $A_{n,k}$ se obţine înmulţind (în $\mathcal{R}$) matricea de reprezentare modulară cu matricea C:*

$$W_{2^k-1} = NC \tag{1}$$

Demonstraţie: Să scriem matricea M sub forma unei matrici linie

$$M = (\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_{2^k-1}}),$$

unde $\mathbf{v_i}$ $(1 \le i \le 2^k - 1)$ este coloana de tip i din matricea M.

Matricea generatoare corespunzătoare reprezentării modulare $N = (n_1, n_2, \ldots, n_{2^k-1})$ este

$$G = (\underbrace{\mathbf{v_1}, \ldots, \mathbf{v_1}}_{n_1}, \underbrace{\mathbf{v_2}, \ldots, \mathbf{v_2}}_{n_2}, \ldots, \underbrace{\mathbf{v_{2^k-1}}, \ldots, \mathbf{v_{2^k-1}}}_{n_{2^k-1}})$$

unde $n_1 + n_2 + \ldots + n_{2^k-1} = n$.

Se obţine deci

$$C = \begin{pmatrix} \mathbf{v_1} \\ \mathbf{v_2} \\ \vdots \\ \mathbf{v_{2^k-1}} \end{pmatrix} (\mathbf{v_1}, \ldots, \mathbf{v_{2^k-1}}) = \begin{pmatrix} \mathbf{v_1 v_1} & \mathbf{v_1 v_2} & \ldots & \mathbf{v_1 v_{2^k-1}} \\ \mathbf{v_2 v_1} & \mathbf{v_2 v_2} & \ldots & \mathbf{v_2 v_{2^k-1}} \\ & & \vdots & \\ \mathbf{v_{2^k-1} v_1} & \mathbf{v_{2^k-1} v_2} & \ldots & \mathbf{v_{2^k-1} v_{2^k-1}} \end{pmatrix},$$

unde produsele scalare sunt efectuate în Z_2.

Acum, $W = (w_1, w_2, \ldots, w_{2^k-1}) = (n_1, n_2, \ldots, n_{2^k-1})C$, înmulţire efectuată în $\mathcal{R}$.

Prin identificare rezultă

$$w_i = n_1 \mathbf{v_1 v_i} + n_2 \mathbf{v_2 v_i} + \ldots + n_{2^k-1} \mathbf{v_{2^k-1} v_i}.$$

Deoarece toate produsele scalare $\mathbf{v_j v_i}$ sunt 0 sau 1, w_i este un număr natural. El este chiar ponderea cuvântului cod care ocupă linia i în matricea K.

Într-adevăr, avem

$$K_{2^k-1,n} = \begin{pmatrix} \mathbf{v_1 v_1} & \mathbf{v_1 v_1} & \ldots & \mathbf{v_1 v_{2^k-1}} \\ \mathbf{v_2 v_1} & \mathbf{v_2 v_1} & \ldots & \mathbf{v_2 v_{2^k-1}} \\ & & \vdots & \\ \mathbf{v_{2^k-1} v_1} & \mathbf{v_{2^k-1} v_1} & \ldots & \mathbf{v_{2^k-1} v_{2^k-1}} \end{pmatrix},$$

produsele scalare fiind în Z_2.

Deci ponderea cuvântului situat pe linia i în matricea K este:

$\underbrace{\mathbf{v_i v_1} + \ldots + \mathbf{v_i v_1}}_{n_1} + \ldots + \underbrace{\mathbf{v_i v_{2^k-1}} + \ldots + \mathbf{v_i v_{2^k-1}}}_{n_{2^k-1}} = n_1 \mathbf{v_i v_1} + \ldots + n_{2^k-1} \mathbf{v_i v_{2^k-1}}$. q.e.d.

Ne punem problema de a determina un cod atunci când se dă vectorul ponderilor cuvintelor sale (deci şi ponderea minimă, adică distanţa Hamming).

Formal, din (1) se obţine

$$N = WC^{-1}$$

înmulţire efectuată în $\mathcal{R}$.

Formula este adevărată numai dacă demonstrăm că matricea C este inversabilă.

În acest sens avem:

Teorema 7.2 *Matricea C este nesingulară. Inversa ei se calculează înlocuind în C pe 0 cu -1 şi împărţind toate elementele cu 2^{k-1}.*

Demonstraţie: **I**: Fiecare linie (deci şi coloană) din C conţine 1 pe 2^{k-1} poziţii. Două linii (coloane) distincte din C au în comun 1 în 2^{k-2} poziţii.

Într-adevăr, liniile lui C – la care se adaugă linia nulă – formează un spaţiu liniar cu 2^k cuvinte binare. Să considerăm submulţimea liniilor care au 1 în componentele i şi j. Ele formează un subspaţiu liniar cu $\dfrac{2^k}{2^2} = 2^{k-2}$ elemente deoarece, factorizând cu acest spaţiu se obţin patru clase de resturi, corespunzătoare componentelor (i, j) de forma $(0,0), (0,1), (1,0), (1,1)$.

Pentru a arăta că 1 apare pe fiecare linie din C de 2^{k-1} ori se raţionează similar.

II: În această situaţie, putem scrie

$$C^2 = \begin{pmatrix} 2^{k-1} & 2^{k-2} & \dots & 2^{k-2} \\ 2^{k-2} & 2^{k-1} & \dots & 2^{k-2} \\ & & \vdots & \\ 2^{k-2} & 2^{k-2} & \dots & 2^{k-1} \end{pmatrix} = 2^{k-2}(I + J)$$

unde I este matricea unitate iar J este matricea pătrată cu toate elementele egale cu 1, ambele de ordin 2^k.

Se mai observă că $CJ = 2^{k-1}J$. Deci

$$I = \frac{1}{2^{k-1}}(2C^2 - 2^{k-1}J) = \frac{1}{2^{k-1}}(2C^2 - CJ) = C\frac{2C - J}{2^{k-1}} = CC^{-1}.$$

De aici rezultă că C este inversabilă şi $\quad C^{-1} = \dfrac{1}{2^{k-1}}(2C - J)$. q.e.d.

Exemplul 7.2 *Fie $k = 3$, $n = 5$. Vom avea:*

$$M = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}, \qquad C = M^T M = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix},$$

$$C^{-1} = \frac{1}{4}\begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & -1 & 1 & -1 \\ -1 & 1 & 1 & 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{pmatrix}.$$

Dacă luăm codul binar cu matricea generatoare $G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}$, *el are reprezentarea modulară* $N = (1, 1, 0, 1, 1, 1, 0)$. *Vectorul ponderilor cuvintelor - cod este* $W = (2, 2, 4, 3, 3, 3, 3)$[1]. *Într-adevăr, cuvintele - cod sunt liniile matricii*

$$K = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}.$$

Să determinăm acum reprezentarea modulară a unui cod liniar binar în care toate cuvintele - cod au aceeaşi pondere w (şi deci distanţa minimă va fi $d = w$, iar numărătorul de ponderi $P(X) = 1 + (2^k - 1)X^w$). Vom avea prin ipoteză $W = (\underbrace{w, w, \ldots, w}_{2^k-1})$.

După cum am remarcat în demonstraţia Teoremei 20.2, 1 apare pe fiecare linie din C în 2^{k-1} poziţii. Rezultă (Teorema 20.2) că în C^{-1}, 1 va apare pe fiecare linie (coloană) tot în 2^{k-1} poziţii, în rest fiind -1. Deci

$$N = WC^{-1} = (\underbrace{w2^{-(k-1)}, \ldots, w2^{-(k-1)}}_{2^k-1})$$

Problema enunţată are soluţie dacă w se divide cu 2^{k-1}; în acest caz, reprezentarea modulară este de forma $N = (\underbrace{r, r, \ldots, r}_{2^k-1})$, adică fiecare tip de coloană apare în matricea generatoare de $r = w \cdot 2^{-(k-1)}$ ori.

Plecând de la aceste considerente, McDonald a introdus o clasă de coduri pentru care reprezentarea modulară este de forma 0^i1^j. Mai exact,

n	d	N
$2^k - 1$	2^{k-1}	$(1, 1, 1, \ldots, 1)$
$2^k - 2$	$2^{k-1} - 1$	$(0, 1, 1, \ldots, 1)$
$2^k - 3$	$2^{k-1} - 2$	$(0, 0, 1, \ldots, 1)$
$\ldots$	$\ldots$	$\ldots$
$2^k - 2^u$	$2^{k-1} - 2^{u-1}$	$(\underbrace{0, \ldots, 0}_{2^u-1}, \underbrace{1, \ldots, 1}_{2^k-2^u})$

Aceste coduri au proprietatea că prezintă o distanţă minimă foarte apropiată de marginea Plotkin.

Să luăm de exemplu al treilea cod din tabelul de sus. Reamintim, marginea Plotkin este

$$d \leq \frac{nq^{k-1}(q-1)}{q^k - 1}.$$

În cazul codului McDonald avem $q = 2$, $n = 2^k - 3$, $d = 2^{k-1} - 2$, deci

$$\frac{nq^{k-1}(q-1)}{q^k - 1} - d = \frac{(2^k - 3)2^{k-1}}{2^k - 1} - (2^{k-1} - 2) = \frac{2^k - 2}{2^k - 1} < 1.$$

[1] De aici se poate obţine şi numărătorul de ponderi: $P(X) = 1 + 2X^2 + 4X^3 + X^4$.

7.2 Coduri derivate din matricile Hadamard

În Capitolul anterior am folosit o matrice Hadamard pentru a construi un algoritm de decodificare al codurilor $\mathcal{RM}(1,m)$. De fapt matricile Hadamard pot fi utilizate în mod direct la construcţia de coduri.

Definiţia 7.1 *Se numeşte matrice Hadamard $\mathcal{H}_n$ o matrice pătrată de ordinul n cu elemente 1 şi -1, ale cărei linii sunt ortogonale două câte două.*

Teorema 7.3 *Fiind dată matricea Hadamard $\mathcal{H}_n$, (n par) se poate construi un cod de distanţă minimă $d = \frac{n}{2}$, format din $2n$ cuvinte de lungime n.*

Demonstraţie: În matricea $\mathcal{H}_n$ vom înlocui 1 cu 0 şi -1 cu 1 şi vom nota $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}$ vectorii care formează liniile noii matrici.

Codul căutat este

$$C = \{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}, -\mathbf{v_1}, -\mathbf{v_2}, \ldots, -\mathbf{v_n}\}.$$

Acest cod (care nu este neapărat liniar) are $2n$ cuvinte de lungime n.

Mai trebuie arătat că distanţa sa minimă este $n/2$.

Deoarece $\mathbf{v}$ şi $-\mathbf{v}$ diferă în toate componentele, distanţa Hamming dintre ele este n. Pe de-altă parte, din proprietatea matricilor Hadamard rezultă

$$\mathbf{v_i}(\pm\mathbf{v_j}) = 0,\ \forall i,j \in \{1,\ldots,n\},\ i \neq j.$$

Pentru ca această egalitate să fie adevărată, este necesar ca $\mathbf{v_i}$ şi $\mathbf{v_j}$ să coincidă în jumătate din componente şi să difere în cealaltă jumătate (atenţie, calculele se fac în $\mathcal{R}$, nu în Z_2!). Rezultă că distanţa Hamming dintre cele două cuvinte este $\frac{n}{2}$. Deci $d = \frac{n}{2}$. q.e.d.

Pentru $n = 32$ un astfel de cod a fost folosit de programul spaţial Mariner.

Teorema 7.4 *Dacă $\mathcal{H}_n$ este matrice Hadamard, atunci $\mathcal{H}_{2n} = \begin{pmatrix} \mathcal{H}_n & \mathcal{H}_n \\ \mathcal{H}_n & -\mathcal{H}_n \end{pmatrix}$ este matrice Hadamard.*

Demonstraţie: Evident, $\mathcal{H}_{2n}$ este o matrice $2n \times 2n$ cu elemente ± 1. Mai trebuie arătat că liniile sale sunt ortogonale două câte două.

Avem: $[\mathbf{v_i}, \mathbf{v_i}][\mathbf{v_i}, -\mathbf{v_i}] = \mathbf{v_i}\mathbf{v_i} - \mathbf{v_i}\mathbf{v_i} = n - n = 0$.
De asemenea, pentru $i \neq j$, $\quad [\mathbf{v_i}, \mathbf{v_i}][\mathbf{v_j}, \pm\mathbf{v_j}] = [\mathbf{v_i}, \mathbf{v_j}] \pm [\mathbf{v_i}, \mathbf{v_j}] = 0 \pm 0 = 0$. q.e.d.

Pe baza acestei teoreme, putem construi coduri Hadamard a căror distanţă să fie oricât de mare. Este suficient să găsim o matrice Hadamard iniţială, pe care (folosind Teorema 20.3) să o "dilatăm" ulterior cât este nevoie.

O astfel de matrice Hadamard iniţială a fost dată în Capitolul 6:

$$\mathcal{H}_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Pentru construcţia altor matrici (şi coduri) Hadamard mai generale au fost elaborate diverse tehnici. Una din cele mai cunoscute este cea a lui Paley şi Hall.

Definiţia 7.2 *O matrice pătrată M de ordin n cu 0 pe diagonala principală şi ± 1 înafara ei, astfel ca $MM^T = (n-1)I_n$ se numeşte matrice de conjunctură (conference matrix).*

Exemplul 7.3 *Matricile* $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, $\begin{pmatrix} 0 & 1 & 1 & 1 \\ -1 & 0 & -1 & 1 \\ -1 & 1 & 0 & -1 \\ -1 & -1 & 1 & 0 \end{pmatrix}$

sunt matrici de conjunctură.

Fie q un număr prim impar. În corpul Z_q definim funcţia reziduu χ astfel:

$$\chi(x) = \begin{cases} 0 & \text{dacă } x = 0 \\ 1 & \text{dacă } x \text{ este un pătrat nenul} \\ -1 & \text{în rest} \end{cases}$$

Teorema 7.5 *Matricea Paley $S_q = (s_{ij})$ de ordin q definită prin $s_{ij} = \chi((q+i-j) \bmod q)$, $(i, j \in Z_q)$ are următoarele proprietăţi:*

1. $S_q J_q = J_q S_q = \mathbf{0}$;
2. $S_q S_q^T = qI_q - J_q$;
3. $S_q^T = (-1)^{\frac{q-1}{2}} S_q$.

Toate calculele sunt în Z. S-a notat cu I_n matricea unitate de ordin n şi cu J_n matricea pătrată de ordin n cu toate elementele egale cu 1.

Demonstraţie: Este lăsată ca exerciţiu.

Exemplul 7.4 *Pentru $q = 3$, matricea Paley este* $S_3 = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$,

iar pentru $q = 5$, $S_5 = \begin{pmatrix} 0 & 1 & -1 & -1 & 1 \\ 1 & 0 & 1 & -1 & -1 \\ -1 & 1 & 0 & 1 & -1 \\ -1 & -1 & 1 & 0 & 1 \\ 1 & -1 & -1 & 1 & 0 \end{pmatrix}$.

De remarcat că $S_3^T = -S_3$, $S_5^T = S_5$ (ceea ce verifică şi Teorema 7.5, punctul 3).

Teorema 7.6 *Fie q un număr prim impar astfel încât $q \equiv 3 \pmod 4$ (număr Blum). Atunci există o matrice Hadamard de ordin $q + 1$.*

Demonstraţie: Din ipoteză şi Teorema 7.5, se poate defini matricea Paley S_q. Construim matricea M de ordin $q + 1$ astfel:

$$M = \begin{pmatrix} 0 & 1 & \dots & 1 \\ -1 & & & \\ -1 & & S_q & \\ -1 & & & \end{pmatrix}$$

Se verifică faptul că M este o matrice de conjunctură cu proprietatea $M^T = -M$ ($q \equiv 3 \; mod \; 4$ conduce la relaţia $S^T = -S$).

Fie $H = I_{q+1} + M$. Avem $HH^T = (I_{q+1} + M)(I_{q+1} + M)^T = (I_{q+1} + M)(I_{q+1} + M^T) = I_{q+1} + M + M^T + MM^T = I_{q+1} + qI_{q+1} = (q+1)I_{q+1}$, deci orice două linii distincte ale lui H sunt ortogonale.

Rezultă că H este matrice Hadamard de ordin $q+1$. q.e.d.

Plecând de la matricile Paley se pot construi şi alte coduri Hadamard (modificând puţin Teorema 7.3).

Fie S_n o matrice Paley de ordin n (conform Teoremei 7.5). Un cod Hadamard de ordin n poate fi definit cu $\mathbf{0}, \mathbf{1}$ şi cu liniile matricilor $\frac{1}{2}(S_n + I_n + J_n)$, $\frac{1}{2}(-S_n + I_n + J_n)$. Acesta este un cod care conţine $2n+2$ cuvinte - cod de lungime n şi are distanţa minimă $d = \frac{n-1}{2}$.

Exemplul 7.5 *Pentru $n = 9 = 3^2$, codul Hadamard este format din cele 20 cuvinte care formează linile matricii*

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & J & & & P^2 & & & P & \\ & P & & & J & & & P^2 & \\ & P^2 & & & P & & & J & \\ & I & & & J-P^2 & & & J-P & \\ & J-P & & & I & & & J-P^2 & \\ & J-P^2 & & & J-P & & & I & \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

unde $P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$, *iar I şi J sunt de ordin 3.*

7.3 Coduri produs

Să considerăm un cod de lungime $n = n_1 n_2$. În loc să scriem cuvintele sale sub formă de linii de lungime n, le putem reprezenta ca matrici cu n_1 linii şi n_2 coloane. Un mod simplu de a realiza acest lucru este de a scrie cuvântul cod $\mathbf{a} = a_0 a_1 \ldots a_{n-1}$ ca o matrice $X = [x_{ij}], 0 \leq i \leq n_1 - 1, \; 0 \leq j \leq n_2 - 1$, unde $x_{ij} = a_{in_2+j}$.

Aceasta va fi numită *scrierea canonică.*

Definiţia 7.3 *Fie C_1, C_2 două coduri liniare de lungime n_1 şi respectiv n_2. Se construieşte codul C de lungime $n_1 n_2$ ca mai sus (folosind scrierea canonică). Spunem că $C = C_1 \times C_2$ este codul produs al lui C_1 cu C_2 dacă şi numai dacă C constă din toate cuvintele - cod în care scrierea canonică verifică proprietăţile:*

- *Fiecare coloană a unei matrici este un cuvânt - cod din C_1;*
- *Fiecare linie a unei matrici este un cuvânt - cod din C_2.*

De remarcat că prin permutarea lui C_1 cu C_2 se obţine un cod echivalent.

Teorema 7.7 *Dacă C_1 este un (n_1, k_1) - cod liniar şi C_2 este un (n_2, k_2) - cod liniar, atunci $C_1 \times C_2$ este un $(n_1 n_2, k_1 k_2)$ - cod liniar.*

Demonstraţie: Fie G_i $(i = 1, 2)$ matricile generatoare ale celor două coduri, pe care le presupunem în forma eşalonat canonică; deci $G_i = [I_{k_i} B_{n_i - k_i}]$.

Vom nota cu $\mathbf{g_i^{(1)}}$ $(1 \leq i \leq k_1)$ respectiv $\mathbf{g_i^{(2)}}$ $(1 \leq i \leq k_2)$ liniile celor două matrici.

Definim matricile X_{ij} $(1 \leq i \leq k_1,\ 1 \leq j \leq k_2)$ de dimensiune $n_1 \times n_2$, astfel:

- Primele k_1 linii sunt $\mathbf{0}$ cu excepţia liniei i care este $\mathbf{g_j^{(2)}}$;
- Primele k_2 coloane sunt $\mathbf{0}$ cu excepţia coloanei j care este $\mathbf{g_i^{(1)}}^T$.
- Pentru $k > k_1$, linia k este $g_{ik}^{(1)} \mathbf{g_j^{(2)}}$;
- Pentru $k > k_2$, coloana k este $g_{jk}^{(2)} \mathbf{g_i^{(1)}}^T$.

Schematic, o astfel de matrice arată astfel:

$$X_{ij} = \left(\begin{array}{ccc|ccc} 0 & \dots & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ & 1 & & x & x & x \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & \dots & 0 \\ \hline 0 & x & 0 & & & \\ \dots & x & \dots & & X & \\ 0 & x & 0 & & & \end{array}\right)$$

Evident, X_{ij} reprezintă un cuvânt - cod din $C_1 \times C_2$ şi orice alt cuvânt - cod este o combinaţie liniară formată din aceste matrici. Deci X_{ij} $(1 \leq i \leq k_1,\ 1 \leq j \leq k_2)$ formează o bază a codului $C_1 \times C_2$.

Submatricile formate din primele k_1 linii şi k_2 coloane formează mulţimea mesajelor de informaţie. q.e.d.

Exemplul 7.6 *Să considerăm codurile liniare binare C_1, C_2 generate de matricile* $G_1 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$ *respectiv* $G_2 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$.

Deci $n_1 = n_2 = 3$, $k_1 = k_2 = 2$. Sunt 4 matrici X_{ij}, toate de dimensiune 3×3:

$$X_{11} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} \qquad X_{12} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$X_{21} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} \qquad X_{22} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

Să considerăm mesajul de informaţie 1011 *pe care îl aşezăm sub forma unei matrici* 2×2 : $\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$.

Matricea - cod va fi

$$X_{11} + X_{21} + X_{22} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Se observă că liniile sunt cuvinte - cod din C_2*, iar coloanele sunt cuvinte - cod din* C_1*.*

Propoziţia 7.1 *Matricea generatoare a codului produs* $C_1 \times C_2$ *este* $G_1 \times G_2$ *unde* G_1 *şi* G_2 *sunt matricile generatoare ale celor două coduri, iar "*$\times$*" este produsul Kronecker.*

Demonstraţie: Fie $(n_1, k_1), (n_2, k_2)$ cele două coduri, cu matricile generatoare G_1 respectiv G_2, şi să considerăm matricea $G_1 \times G_2$ obţinută prin produsul Kronecker (Capitolul 6, Definiţia 6.3).

Dacă se ia linia $k_2 i + j$ din această matrice şi se reprezintă sub formă canonică ca o matrice $n_1 \times n_2$, se obţine exact matricea X_{ij} din construcţia Teoremei 7.7. Cum X_{ij} reprezintă o bază pentru codul $C_1 \times C_2$, rezultă că $G_1 \times G_2$ este matricea sa generatoare[2]. q.e.d.

Exemplul 7.7 *Matricea generatoare a codului produs* $A_1 \times A_2$ *din Exemplul 7.6 este:*

$$G = G_1 \times G_2 = \begin{pmatrix} G_2 & \mathbf{0} & G_2 \\ \mathbf{0} & G_2 & G_2 \end{pmatrix} = \left(\begin{array}{ccc|ccc|ccc} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}\right).$$

Se observă imediat că prin "plierea" celor patru linii din această matrice se obţin matricile X_{ij} *din Exemplul 7.6.*

Să considerăm din nou mesajul de informaţie 1011*. Prin înmulţirea cu matricea* G *se ajunge la cuvântul - cod* 101111010 *care - scris ca o matrice* 3×3 *- coincide cu rezultatul din Exemplul 7.6.*

Teorema 7.8 *Distanţa minimă a codului* $C_1 \times C_2$ *este egală cu* $d_1 \cdot d_2$ *(*d_1 *şi* d_2 *fiind distanţele minime ale codurilor* C_1 *respectiv* C_2*).*

Demonstraţie: Dacă C_i are lungimea n_i $(i = 1, 2)$, să considerăm o matrice X reprezentând un cuvânt - cod nenul din $C_1 \times C_2$ (am văzut că asemenea cuvinte există).

Putem găsi deci cel puţin o linie cu minim d_2 elemente nenule. Fiecare astfel de element se află pe o coloană cu minim d_1 elemente nenule; deci matricea X are minim $d_1 \cdot d_2$ elemente nenule. q.e.d.

Exemplul 7.8 *Cele două coduri din Exemplul 7.6 au* $d_1 = d_2 = 2$ *(se găsesc imediat cuvintele - cod şi se observă că ponderea lor nenulă minimă este* 2*). Deci codul produs va avea distanţa minimă* $d = 2 \cdot 2 = 4$*. De remarcat că – deşi* C_1 *şi* C_2 *nu pot corecta erori,* $C_1 \times C_2$ *este capabil să corecteze o eroare.*

Codurile produs - cu toate că au fost definite de la începutul anilor ′60, au devenit foarte importante odată cu folosirea lor în transmisiile telefonice fără fir şi codificarea informaţiei pe CD - uri şi DVD-uri. Vom relua ulterior – pe larg – analiza acestor coduri.

[2]Din acest motiv, în primii ani, codurile produs erau numite şi coduri Kronecker.

7.4 Coduri optimal maximale

Fie mulţimea $C \subseteq Z_q^n$. Se numeşte *diametrul* mulţimii C numărul

$$d(C) = \min \{d(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in C,\ \mathbf{x} \neq \mathbf{y}\}.$$

De remarcat că, dacă C este cod liniar, atunci $d(C)$ este chiar distanţa sa minimă.

Definiţia 7.4 *C este mulţime optimală de distanţă minimă d dacă:*

- $d(C) = d$;
- $\forall \mathbf{x} \in Z_q^n \setminus C,\ d(C \cup \{\mathbf{x}\}) < d$.

Exemplul 7.9 *În Z_2^6, următoarele mulţimi sunt optimale de distanţă minimă 3:*
$A = \{000000, 010101, 101010, 111111\}$
$B = \{000000, 101010, 011001, 000111, 110100, 111111\}$

Definiţia 7.5 *Se numeşte mulţime optimală cardinal maximală de distanţă minimă d o mulţime optimală din Z_q^n având un număr maxim de elemente.*

Vom nota acest număr cu $a(n, d, q)$.

Exemplul 7.10 *Mulţimile date în Exemplul 7.9 sunt optimale dar nu cardinal maximale pentru distanţa 3. O mulţime optimală cardinal maximală din Z_2^6 pentru $d = 3$ este*

$$C = \{000000, 001011, 010101, 011110, 100110, 101101, 110011, 111000\},$$

deci $a(6, 3, 2) = 8$.

Definiţia 7.6 *Un cod liniar C de distanţă minimă d este optimal dacă C este mulţime optimală cardinal maximală de distanţă minimă d.*

Spunem că un astfel de cod satisface *condiţia max - min*, deoarece are un număr maxim de vectori aflaţi la distanţa minimă d.

Nu se cunosc algoritmi de construcţie a unor astfel de coduri. În această secţiune vom da numai câteva condiţii necesare pentru definirea codurilor optimale.

Teorema 7.9 *Într-un cod optimal C de lungime n şi distanţă minimă d,*

$$a(n, d, q) \leq q \cdot a(n-1, d, q).$$

Demonstraţie: Putem presupune – fără a micşora generalitatea – că primul caracter al codului este caracter de informaţie.

Fie K mulţimea cuvintelor din C care au 0 pe prima componentă. Cum C este cod liniar, iar pe prima componentă pot apare q caractere distincte, va rezulta

$$card(K) = \frac{a(n, d, q)}{q}.$$

K este tot un cod liniar cu distanţa minimă d, în care toate cuvintele au 0 pe prima poziţie. Dacă eliminăm această componentă, se obţine un cod din Z_q^{n-1}, tot de distanţă minimă d, cu acelaşi număr de cuvinte - cod.

Acest cod nu este neapărat optimal; deci condiţia va fi

$$a(n-1,d,q) \geq \frac{a(n,d,q)}{q}$$

Mai trebuie făcută observaţia că $a(n-1,n-1,q) = q$ pentru că un cod cu repetiţie este optimal de distanţă minimă egală cu lungimea cuvintelor - cod. q.e.d.

Corolarul 7.1 *Dacă $j \geq d$, atunci $q^{n-j}a(j,d,q) \geq a(n,d,q)$.*

Demonstraţie: Din Teorema 7.9 rezultă inegalităţile:

$$\begin{array}{l} q \cdot a(n-1,d,q) \geq a(n,d,q) \\ q \cdot a(n-2,d,q) \geq a(n-1,d,q) \\ \ldots \\ \underline{q \cdot a(n-i,d,q) \geq a(n-i+1,d,q)} \end{array}$$

Prin înmulţire se obţine $q^i \cdot a(n-i,d,q) \geq a(n,d,q)$, după care se face notaţia $n-i=j$. q.e.d.

Teorema 7.10 *Fie $A_{n,k} \subseteq Z_q^n$ un cod liniar optimal de distanţă minimă d.*

Dacă $n < \dfrac{qd}{q-1}$, atunci numărul k al simbolurilor de informaţie verifică inegalitatea

$$k \leq n - \frac{dq-1}{q-1} + 1 + log_q d.$$

Demonstraţie: Deoarece $A_{n,k}$ este cod liniar, numărul de simboluri de informaţie va fi dat de relaţia $a(n,d,q) = q^k$.

Rescriem inegalitatea lui Plotkin (Capitolul 3) sub forma $d(q^k-1) \leq nq^{k-1}(q-1)$, sau

$$q^{k-1}(dq+n-nq) \leq n.$$

Deoarece – din ipoteză – $dq+n-nq > 0$, avem $q^k \leq \dfrac{dq}{dq+n-nq}$, sau

$$a(n,d,q) \leq \frac{dq}{dq+n-nq}.$$

Fie $\dfrac{dq-1}{q-1} = i + f$ unde $i = \left[\dfrac{dq-1}{q-1}\right]$, $f = \left\{\dfrac{dq-1}{q-1}\right\}$.

Deci $qd-1 = (q-1)i + (q-1)f$ şi avem

$$a(i,d,q) \leq \frac{dq}{dq+i-iq} = \frac{dq}{dq-(q-1)i} = \frac{dq}{1+(q-1)f}.$$

Folosind Corolarul 15.1 şi $i \geq d$ (deoarece $\left[\frac{dq-1}{q-1}\right] = \left[d + \frac{d-1}{q-1}\right] \geq d$), rezultă

$$a(n,d,q) \leq q^{n-i} a(i,d,q) \leq \frac{q^{n-\frac{dq-1}{q-1}+f}}{1+(q-1)f} qd.$$

Dezvoltând în serie Taylor, $q^f \leq 1+(q-1)f \quad (f<1)$, şi deci

$$a(n,d,q) \leq q^{n-\frac{dq-1}{q-1}} qd,$$

sau $q^k \leq q^{n-\frac{dq-1}{q-1}} qd$, de unde – prin logaritmare – se obţine relaţia din enunţ. q.e.d.

7.5 Exerciţii

7.1 *Să se construiască vectorii ponderilor pentru codurile Hamming* $(7,4)$ *şi* $\mathcal{RM}(1,3)$.

7.2 *Folosind vectorul ponderilor, să se construiască numărătorul de ponderi al unui cod liniar.*

7.3 *Să se construiască matricile generatoare şi matricile* K *pentru codurile McDonald cu* $k = 2,3,4$.

7.4 *Să se construiască matricile şi codurile Hadamard pentru* $n = 4$ *şi* $n = 8$.

7.5 *Să se arate că dacă* $\mathcal{H}_n$ *este o matrice Hadamard, atunci* $\mathcal{H}_n \mathcal{H}_n^T = nI_n$.

7.6 *Să se construiască o matrice Hadamard de ordin* 12.

7.7 *Să se arate că matricea Hadamard* $\mathcal{H}_8$ *generează* $(8,4)$ *- codul Hamming binar extins.*

7.8 *Să se construiască codurile produs pentru codul cu repetiţie de lungime* n *(*C_1*) şi codul cu repetiţie de lungime* p *(*C_2*).*

7.9 *Să se construiască matricea generatoare a codului produs* $C \times C$ *unde* C *este codul Hamming* $(7,4)$.

7.10 *Aceeaşi problemă pentru codul* $\mathcal{RM}(1,3)$.

7.11 *Demonstraţi Teorema 7.5.*

7.12 *Folosind matricile Paley, să se construiască codurile Hadamard de ordin* 5, 7 *şi respectiv* 10.

7.13 *Să se arate că un cod binar liniar optimal de distanţă minimă* d *are cel puţin* $2d - 1 - log_2 d$ *simboluri de control.*

Capitolul 8

Circuite liniare şi extensii Galois

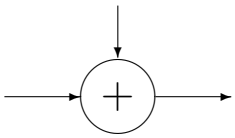
8.1 Circuite liniare pentru operaţii elementare

În această secţiune presupunem că toate calculele se realizează într-un corp de caracteristică q (q număr prim).

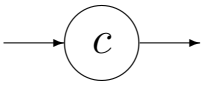
Un *circuit liniar* este un graf orientat, cu trei tipuri de noduri (conectori):

- *Sumator*: este un conector cu cel puţin două intrări şi o ieşire. Efectul este acela de a scoate suma modulo q a elementelor aflate la intrare. Operaţia se execută instantaneu.

  Un sumator (cu 2 intrări) se notează astfel:



- *Multiplicator*: este un conector cu o singură intrare şi o ieşire unde se obţine rezultatul înmulţirii cu $c \in Z_q$ a elementului de la intrare. Operaţia se execută (de asemenea) instantaneu. Notaţie:



- *Element de înmagazinare*: este un registru DFF cu o intrare şi o ieşire; efectul este întârzierea cu un tact a intrării: elementul $a \in Z_q$ intră în momentul k şi iese în momentul $k+1$. Notaţie:

Aceste trei tipuri de conectori se pot grupa arbitrar – folosind eventual şi noduri obişnuite. Utilizarea lor conduce la obţinerea de circuite liniare care pot realiza automat operaţii uzuale cu polinoame: adunări, scăderi, înmulţiri şi împărţiri.

În cele ce urmează, vom identifica un vector cu $n+1$ componente
$$\mathbf{a} = (a_0, a_1, \ldots, a_n)$$
cu polinomul de gradul n cu o variabilă
$$a(X) = a_0 + a_1X + \ldots + a_nX^n$$

În cursul transmisiei mesajului **a**, ordinea de prelucrare (transmisie) se face după puterile descrescătoare ale lui X; deci, sensul de circulaţie a semnalelor va fi:

$$a_0 \quad a_1 \quad \ldots \quad a_n \longrightarrow$$

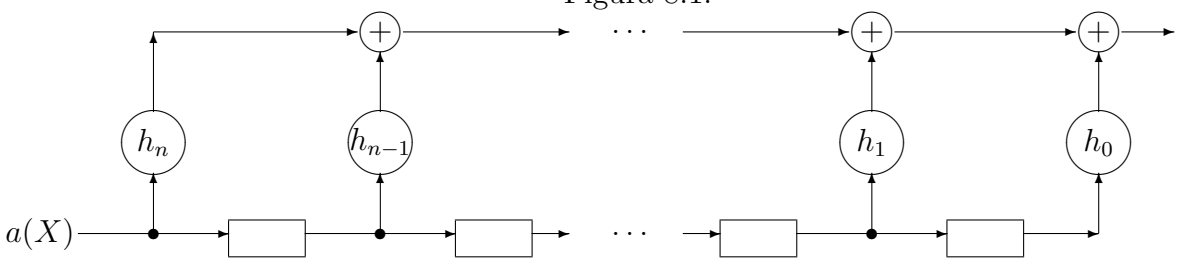
8.1.1 Construcţia unor circuite liniare uzuale

A *Circuit de înmulţire cu un polinom.*

Fie $h(X) = h_0 + h_1X + \ldots + h_nX^n$ un polinom fixat şi $a(X) = a_0 + a_1X + \ldots + a_kX^k$ un polinom arbitrar. Coeficienţii ambelor polinoame pot fi luaţi într-un inel arbitrar (din considerente pur aplicative, acesta este de obicei Z_q, dar construcţiile rămân valabile şi în cazul general).

Un circuit de înmulţire cu $h(X)$ este:

Figura 8.1:



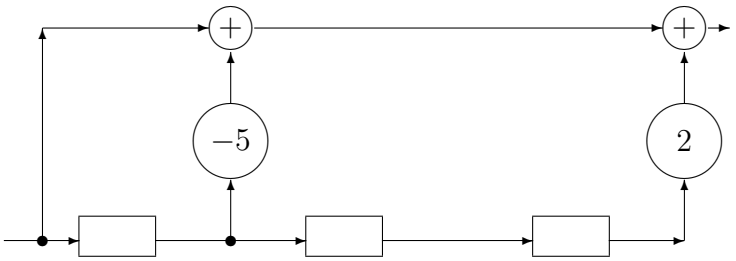
Iniţial elementele de înmagazinare conţin 0; coeficienţii polinomului $a(X)$ intră în stânga jos – după puterile descrescătoare ale lui X – câte unul la un tact, iar coeficienţii produsului ies în dreapta sus. Circuitul funcţionează $n+k+1$ tacţi; după $k+1$ tacţi, la intrare se introduc zerouri în cei n tacţi rămaşi, până ce elementele de înmagazinare conţin din nou numai 0.

De remarcat că un element de înmagazinare funcţionează după principiile unei variabile de memorie (introducerea unei valori face să dispară vechea valoare din locaţie).

Circuitul lucrează după formula de înmulţire obişnuită:

$$a(X)h(X) = a_0h_0 + (a_0h_1 + a_1h_0)X + \ldots + (a_{k-1}h_n + a_kh_{n-1})X^{k+n-1} + a_kh_nX^{n+k}$$

Exemplul 8.1 *Să luăm polinomul* $h(X) = X^3 - 5X^2 + 2 \in Z[X]$ *(vectorul corespunzător este* $\mathbf{h} = (2, 0, -5, 1)$*). Circuitul de înmulţire cu* $h(X)$ *va fi:*



De remarcat că multiplicarea cu 1 *se face prin arc simplu (fără conector), iar multiplicarea cu* 0 *revine la suprimarea arcului. În cazul operaţiilor cu polinoame din* $Z_2[X]$*, toţi multiplicatorii se reduc la aceste două cazuri.*

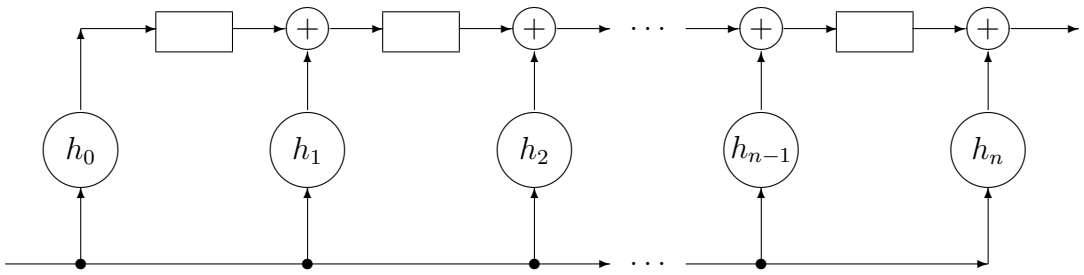
Dacă dorim să înmulţim $h(X)$ *cu polinomul* $a(X) = 2X + 8$ *(vector* $\mathbf{a} = (8, 2)$*), circuitul liniar va funcţiona* 5 *tacţi (până când toate elementele de înmagazinare revin la* 0*); comportarea sa este dată de tabelul:*

Nr. tact	*Intrare*	*Înmagazinare*			*Ieşire*
0	0	0	0	0	0
1	2	0	0	0	2
2	8	2	0	0	−2
3	0	8	2	0	−40
4	0	0	8	2	4
5	0	0	0	8	16
6	0	0	0	0	0

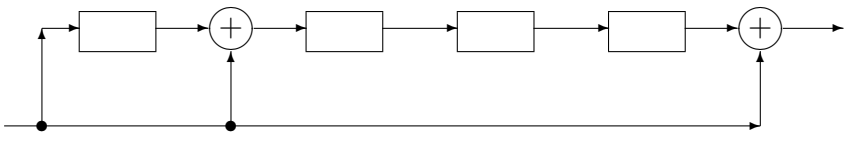
Deci, polinomul produs este $2X^4 - 2X^3 - 40X^2 + 4X + 16$*; sau – în termeni de vectori,* $(16, 4, -40, -2, 2)$.

Aceeaşi operaţie de înmulţire cu un polinom fixat $h(X)$ este realizată şi de circuitul liniar:

Figura 8.2:



Exemplul 8.2 *Circuitul liniar care realizează înmulţirea cu polinomul* $h(X) = 1+X+X^4$ *peste* Z_2 *este*

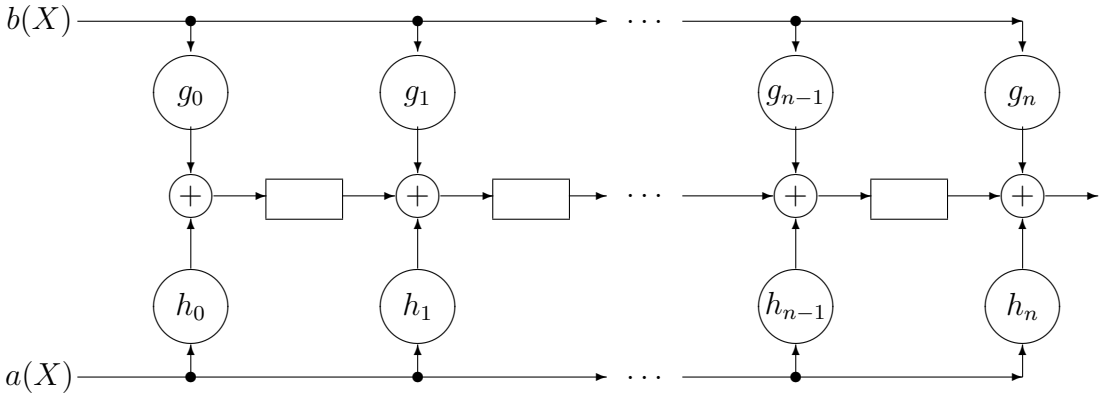


deoarece vectorul corespunzător polinomului $h(X)$ *este* $\mathbf{h} = (1, 1, 0, 0, 1)$.

B. Fie $h(X) = h_0 + h_1X + \ldots + h_nX^n$ şi $g(X) = g_0 + g_1X + \ldots + g_nX^n$ două polinoame fixate. Un circuit liniar care să realizeze operaţia

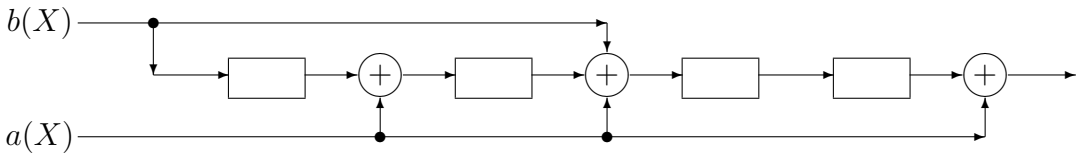
$$a(X)h(X) + b(X)g(X)$$

pentru două polinoame arbitrare $a(X), b(X) \in Z_q[X]$, este:



De remarcat că polinoamele $g(X)$ şi $h(X)$ pot avea grade diferite; în acest caz se alege n ca fiind gradul cel mai mare şi se completează celălalt polinom cu coeficienţi nuli până la gradul n.

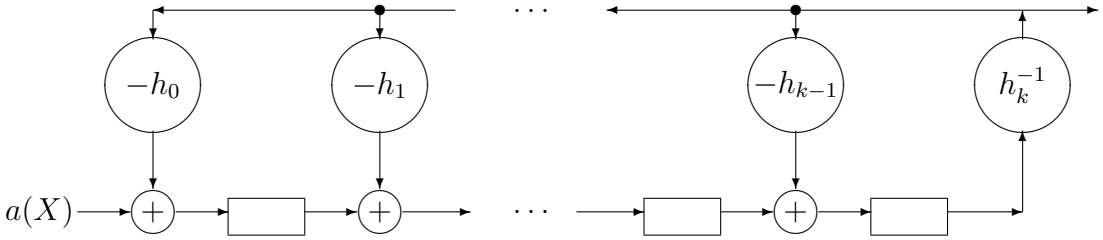
Exemplul 8.3 *Fie polinoamele* $g(X) = 1 + X^2$, $h(X) = X + X^2 + X^4$ *cu coeficienţi în* Z_2*; deci vectorii corespunzători vor fi* $\mathbf{h} = (0, 1, 1, 0, 1)$, $\mathbf{g} = (1, 0, 1, 0, 0)$. *Circuitul liniar care realizează expresia* $a(X)h(X) + b(X)g(X)$ *este:*



C. *Circuit liniar de împărţire cu un polinom.*

Fie polinomul (fixat) $h(X) = h_0 + h_1X + \ldots + h_kX^k$ unde $h_k \neq 0$. Un circuit liniar care realizează împărţirea unui polinom arbitrar $a(X) = a_0 + a_1X + \ldots + a_nX^n$ la $h(X)$ este:

Figura 8.3:

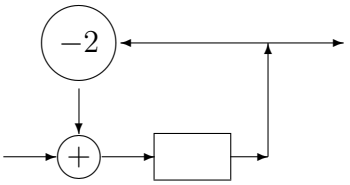


Iniţial, toate cele k elemente de înmagazinare conţin 0. La intrare, timp de $n+1$ tacţi se introduc coeficienţii polinomului $a(X)$ (în ordinea descrescătoare a puterilor). În primii k tacţi, ieşirea va scoate numai 0. Primul coeficient nenul apare la ieşire la momentul $k+1$ şi este $a_n h_k^{-1}$ (operaţii în inelul coeficienţilor celor două polinoame) – coeficientul termenului X^{n-k} din cât.

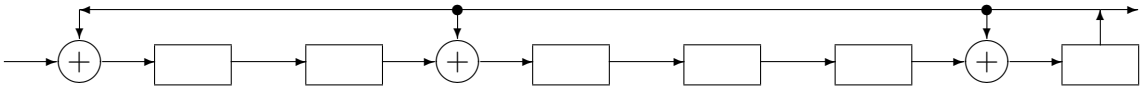
Pentru fiecare coeficient q_j al câtului, polinomul $q_j h(X)$ trebuie scăzut din deîm-părţit; această operaţie se realizează printr-un procedeu de conexiune inversă (feedback).

După momentul $n+1$ (când au intrat toţi coeficienţii polinomului $a(X)$) la ieşire s-a obţinut câtul $q(X)$ al împărţirii lui $a(X)$ la $h(X)$, iar în elementele de înmagazinare se găsesc coeficienţii restului.

Exemplul 8.4 *Un circuit liniar de împărţire (peste Q) la polinomul $h(X) = 2 + X$ este:*



Exemplul 8.5 *Fie polinomul $h(X) = 1+X^2+X^5+X^6 \in Z_2[X]$. Circuitul care realizează împărţirea este:*



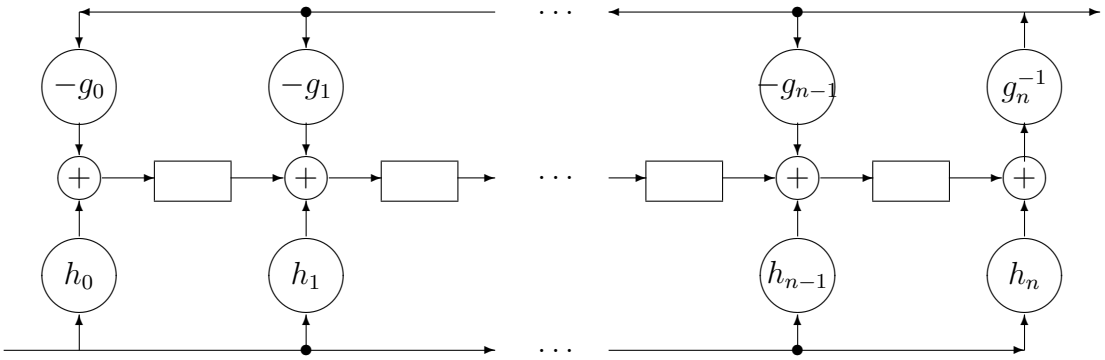
Să luăm polinomul $a(X) = 1 + X^4 + X^5 + X^7 + X^9 + X^{10}$ pe care să-l împărţim în Z_2 la polinomul $h(X)$. Se obţine câtul $X^4 + X$ şi restul $X^5 + X^3 + X + 1$.

Circuitul funcţionează 11 tacţi. Etapele prin care trece circuitul pentru efectuarea acestei împărţiri sunt reprezentate de tabelul următor:

Nr. tact	*Intrare*	*Elemente de înmagazinare*	*Ieşire*
0	–	0 0 0 0 0 0	–
1	1	1 0 0 0 0 0	0
2	1	1 1 0 0 0 0	0
3	0	0 1 1 0 0 0	0
4	1	1 0 1 1 0 0	0
5	0	0 1 0 1 1 0	0
6	1	1 0 1 0 1 1	0
7	1	0 1 1 1 0 0	**1**
8	0	0 0 1 1 1 0	**0**
9	0	0 0 0 1 1 1	**0**
10	0	1 0 1 0 1 0	**1**
11	1	1 1 0 1 0 1	**0**

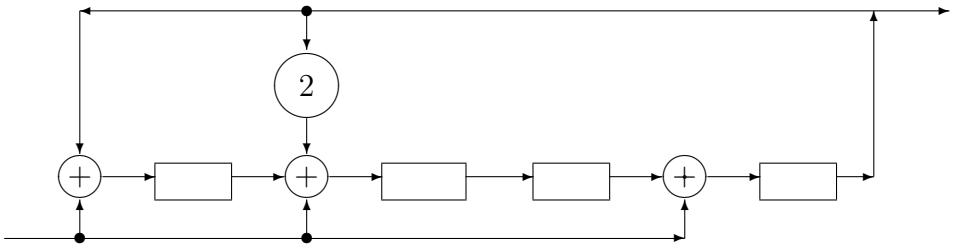
Vectorii $(1, 1, 0, 1, 0, 1)$ - de pe ultima linie şi $(0, 1, 0, 0, 1)$ – scris îngroşat şi în ordine inversă pe ultima coloană – reprezintă restul $1 + X + X^3 + X^5$ respectiv câtul $X + X^4$ împărţirii lui $a(X)$ la $h(X)$.

D. Circuitul liniar care realizează înmulţirea cu polinomul $h(X) = h_0 + h_1X + \ldots + h_nX^n$ urmată de împărţirea la polinomul $g(X) = g_0 + g_1X + \ldots + g_nX^n$ este:



Când gradele celor două polinoame sunt diferite, se construieşte întâi circuitul liniar corespunzător polinomului de grad maxim, după care se completează cu circuitul corespunzător celuilalt polinom.

Exemplul 8.6 *Să luăm polinoamele* $h(X) = 1+X+X^3$, $g(X) = 2+X+X^4$ *cu coeficienţi în* Z_3. *Circuitul liniar care realizează înmulţirea cu* $h(X)$ *şi împărţirea la* $g(X)$ *este*



Deoarece calculele se fac în Z_3, $-2 = 1$, $-1 = 2$, *şi* $1^{-1} = 1$.

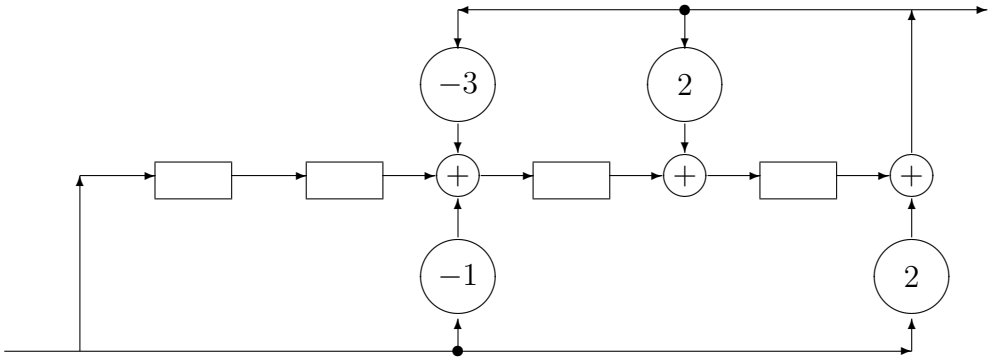
Pentru intrarea $a(X) = 1 + 2X + 2X^2$, *comportarea circuitului este următoarea:*

$-$	0	0	0	0	$-$
2	2	2	0	2	$-$
2	1	2	2	2	2
1	0	0	2	0	2

ceea ce corespunde câtului $2 + 2X$ *şi restului* $2X^2$.

De remarcat că primul coeficient (de grad maxim) al câtului este scos de circuit cu o întârziere de $grad(g(X)) - grad(h(X)) = 1$ *tact.*

Exemplul 8.7 *Un circuit liniar pentru înmulţirea cu* $h(X) = 1-X^2+2X^4$ *şi împărţirea la* $g(X) = 3 - 2X + X^2$ – *ambele din* $Z[X]$ – *este:*



Fie intrarea $a(X) = -2 + X$*; comportarea circuitului este:*

$-$	0	0	0	0	$-$
1	1	0	-7	4	2
-2	-2	1	2	-7	0
0	0	-2	22	-12	-7
0	0	0	34	-2	-12

deci câtul este $2X^3 - 7X - 12$ *iar restul* $34 - 2X$*.*

De remarcat că circuitul este lăsat să funcţioneze $grad(h(X)) - grad(g(X)) = 2$ *tacţi în plus (prin introducerea de zero-uri), pentru golirea elementelor de înmagazinare necontrolate de împărţitor.*

8.2 Extensii Galois

Fie q un număr prim şi $h(X) = h_0 + h_1X + \ldots + h_nX^n \in Z_q[X]$. Vom considera algebra polinoamelor modulo $h(X)$.

Fiind dat un polinom oarecare din $Z_q[X]$, clasa sa de resturi modulo $h(X)$ se găseşte împărţind polinomul respectiv la $h(X)$. Restul împărţirii specifică clasa de resturi respectivă.

Două polinoame cărora le corespunde acelaşi rest la împărţirea cu $h(X)$ sunt echivalente, intrând în aceeaşi clasă de resturi modulo $h(X)$.

Fiecare clasă de resturi modulo $h(X)$ se reprezintă prin polinomul (unic) de grad strict mai mic decât n, care aparţine clasei respective. Toţi aceşti reprezentanţi pot fi consideraţi ca polinoame de gradul $n - 1$, având eventual coeficienţi nuli.

Vom nota cu $\{f(X)\}$ clasa de resturi a polinomului $f(X)$ modulo $h(X)$. Deci $\{f(X)\}$ va corespunde unui vector (distinct) de lungime n, cu componente din Z_q (prin completare eventual cu zerouri).

Operaţiile în algebra claselor de resturi sunt:

$$\{a(X)\} + \{b(X)\} = \{a(X) + b(X)\}$$
$$c\{a(X)\} = \{ca(X)\}, \quad c \in Z_q$$
$$\{a(X)\}\{b(X)\} = \{a(X)b(X)\}$$

toate calculele fiind făcute modulo $h(X)$.

Evident, $\{0\} = \{h(X)\}$, unde $\{0\}$ este polinomul de grad $n-1$ cu toţi coeficienţii nuli (sau vectorul cu n componente zero). Vom nota $\{0\} = \mathbf{0}$.

Pentru un polinom $s(X) \in Z_q[X]$, clasa de resturi modulo $h(X)$ se obţine foarte uşor. Teorema împărţirii cu rest dă:

$$s(X) = a(X)h(X) + r(X)$$

unde $grad(r(X)) < grad(h(X)) = n$. Atunci:

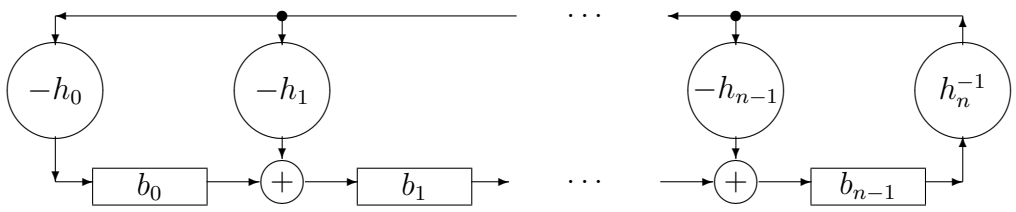
$$\{s(X)\} - \{r(X)\} = \{s(X) - r(X)\} = \{a(X)h(X)\} = \{a(X)\}\{h(X)\} = \mathbf{0},$$

deci $\{s(X)\} = \{r(X)\}$.

Să reluăm circuitul liniar de împărţire definit în Figura 8.3, în care facem următoarele modificări (vezi Figura 8.4):

- Se neglijează intrarea şi ieşirea;
- La momentul iniţial elementele de înmagazinare conţin coeficienţii unui polinom $b(X) = b_0 + b_1X + \ldots + b_{n-1}X^{n-1}$.

Figura 8.4:



La următorul tact, în elementele de înmagazinare se obţin coeficienţii polinomului:

$$b'(X) = -h_0h_n^{-1}b_{n-1} + (b_0 - h_1h_n^{-1}b_{n-1})X + (b_1 - h_2h_n^{-1}b_{n-1})X^2 + \ldots + \\ +(b_{n-2} - h_{n-1}h_n^{-1}b_{n-1})X^{n-1} = \\ = b_0X + b_1X^2 + \ldots + b_{n-1}X^n - b_{n-1}X^n - h_n^{-1}b_{n-1}(h_0 + h_1X + \ldots + h_{n-1}X^{n-1}) = \\ = Xb(X) - h_n^{-1}h_{n-1}g(X).$$

Dacă notăm $\alpha = \{X\}$ ($\{X\}$ este clasa de resturi modulo $h(X)$ a polinomului X), din relaţia de mai sus se obţine:

$$b'(\alpha) = \{b'(X)\} = \{Xb(X)\} = \{X\}\{b(X)\} = \{X\}b(\{X\}) = \alpha b(\alpha).$$

Deci circuitul de împărţire cu polinomul $h(X)$ poate fi utilizat pentru înmulţirea lui $b(\alpha)$ cu α.

Teorema 8.1 *În algebra polinoamelor modulo $h(X)$, $grad(h(X)) = n$, avem:*

1. *$h(\alpha) = \mathbf{0}$ unde $\alpha = \{X\}$;*
2. *h este polinomul de grad minim care are pe α ca rădăcină.*

Demonstraţie:

1. Fie $h(X) = h_0 + h_1X + \ldots + h_nX_n \in Z_q[X], \quad h_n \neq 0.$

   Vom avea $h(\alpha) = h_0 + h_1\alpha + \ldots + h_n\alpha^n = h_0 + h_1\{X\} + \ldots + h_n\{X\}^n =$
   $= h_0 + h_1\{X\} + \ldots + h_n\{X^n\} = \{h_0 + h_1X + \ldots + h_nX^n\} = \{h(X)\} = \mathbf{0}.$

2. Folosind acelaşi raţionament, se obţine pentru orice polinom $f(X)$ cu $grad(f(X)) < n$, că $f(\alpha) = \{f(X)\} \neq \mathbf{0}$, deoarece toate polinoamele ne-identic nule de grad strict mai mic decât n aparţin la clase de resturi distincte. q.e.d.

Să presupunem acum că $h(X) \in Z_q[X]$ este un polinom ireductibil de gradul n. Conform Teoremei 19.1, $\alpha = \{X\}$ este rădăcină a lui $h(X)$. Acest element α poate fi considerat fie ca un polinom de gradul $n-1$ (cu toţi coeficienţii nuli înafară de cel al lui X), fie ca un vector cu n componente $(0, 1, 0, \ldots, 0)$.

Vom nota cu $GF(q^n)$ mulţimea vectorilor din Z_q^n, sau – echivalent:
- mulţimea polinoamelor de grad $n-1$ cu coeficienţi în Z_q, sau
- mulţimea claselor de resturi modulo polinomul $h(X) \in Z_q[X]$ de gradul n, ireductibil peste Z_q.

$GF(q^n)$ se numeşte *extensia Galois de grad* n a lui Z_q.

Spunem că $GF(q^n)$ se obţine prin adăugarea la Z_q a unei rădăcini a polinomului $h(X) \in Z_q[X]$ de grad n, ireductibil peste corpul de bază Z_q.

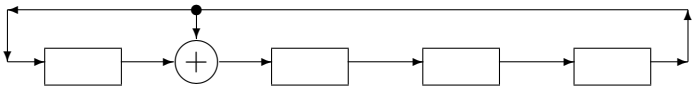
Observaţia 8.1 *Procedeul de trecere de la Z_q la $GF(q^n)$ este similar trecerii de la corpul numerelor reale $\mathcal{R}$ la cel complex $\mathcal{C}$, prin adăugarea rădăcinii $i = \{X\}$ în algebra claselor de resturi modulo polinomul $1 + X^2$, ireductibil peste $\mathcal{R}$. Putem considera mulţimea $\mathcal{C}$ a numerelor complexe fie ca mulţimea polinoamelor de grad $2 - 1 = 1$, $a + bi$ cu $a, b \in \mathcal{R}$, fie ca mulţimea vectorilor (a, b) cu două componente reale.*

Dacă polinomul $h(X) \in Z_q[X]$ este ireductibil peste Z_q şi primitiv, puterile lui $\alpha = \{X\}$ vor genera toate elementele lui $GF(q^n) \setminus \{0\}$ unde $n = grad(h(X))$.

Acest lucru se poate realiza cu circuitul liniar din Figura 8.4, introducând la momentul iniţial în elementele de înmagazinare coeficienţii polinomului $b(X) = 1$ (adică vectorul $(1, 0, \ldots, 0)$).

Conform celor observate mai sus, la momentul următor se obţine $\{Xb(X)\} = \{X\} = \alpha$, după care urmează pe rând $\alpha^2, \alpha^3, \ldots$.
Cum $h(X)$ a fost ales primitiv, el va genera toate elementele nenule din $GF(q^n)$.

Exemplul 8.8 *Fie polinomul $1 + X + X^4$ ireductibil peste Z_2 şi $\alpha = \{X\}$ în algebra claselor de resturi modulo $1 + X + X^4$. Circuitul liniar care va genera toate elementele nenule din $GF(2^4)$ este:*



unde, la momentul iniţial, în elementele de înmagazinare este vectorul $(1, 0, 0, 0)$.

Conform Teoremei 19.1, α este o rădăcină a polinomului $1 + X + X^4$, adică

$$1 + \alpha + \alpha^4 = 0.$$

Lăsând circuitul să funcţioneze, în elementele de înmagazinare se obţin toate elementele lui $GF(2^4)$, pe care le interpretăm fie ca vectori cu patru componente peste Z_2, fie ca polinoame de gradul 3 în α cu coeficienţi în Z_2.

α^0	=	1								1	0	0	0
α^1	=			α						0	1	0	0
α^2	=					α^2				0	0	1	0
α^3	=							α^3		0	0	0	1
α^4	=	1	+	α						1	1	0	0
α^5	=			α	+	α^2				0	1	1	0
α^6	=					α^2	+	α^3		0	0	1	1
α^7	=	1	+	α			+	α^3		1	1	0	1
α^8	=	1			+	α^2				1	0	1	0
α^9	=			α			+	α^3		0	1	0	1
α^{10}	=	1	+	α	+	α^2				1	1	1	0
α^{11}	=			α	+	α^2	+	α^3		0	1	1	1
α^{12}	=	1	+	α	+	α^2	+	α^3		1	1	1	1
α^{13}	=	1			+	α^2	+	α^3		1	0	1	1
α^{14}	=	1					+	α^3		1	0	0	1
α^{15}	=	1								1	0	0	0

Vedem deci că polinomul $1+X+X^4$ *este primitiv, puterile lui* α *epuizând toţi vectorii nenuli din* $GF(2^4) = \{\mathbf{0}, \alpha^0, \alpha, \ldots, \alpha^{14}\}$.

Exemplul 8.9 *Tabelul din exemplul precedent descrie operaţia de înmulţire din extensia galios* $GF(2^4)$ *generată de rădăcina* α *a polinomului primitiv* $1 + X + X^4$. *Dacă s-ar alege* α *ca rădăcină a altui polinom primitiv – de exemplu* $1 + X^3 + X^4$, *se va genera tot* $GF(2^4)$, *dar într-o altă ordine.*

Lăsăm ca exerciţiu această generare.

În $GF(2^4)$ *se poate defini şi operaţia de adunare, însumând modulo* 2 *componentele celor doi operanzi. Astfel, dacă reluăm* $GF(2^4)$ *generat în Exemplul 8.8 de rădăcina polinomului* $1 + X + X^4$, *tabela de adunare va fi:*

$+$	$\mathbf{0}$	α^0	α	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}
$\mathbf{0}$	$\mathbf{0}$	α^0	α	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}
α^0	α^0	$\mathbf{0}$	α^4	α^8	α^{14}	α	α^{10}	α^{13}	α^9	α^2	α^7	α^5	α^{12}	α^{11}	α^6	α^3
α	α	α^4	$\mathbf{0}$	α^5	α^9	α^0	α^2	α^{11}	α^{14}	α^{10}	α^3	α^8	α^6	α^{13}	α^{12}	α^7
α^2	α^2	α^8	α^5	$\mathbf{0}$	α^6	α^{10}	α	α^3	α^{12}	α^0	α^{11}	α^4	α^9	α^7	α^{14}	α^{13}
α^3	α^3	α^{14}	α^9	α^6	$\mathbf{0}$	α^7	α^{11}	α^2	α^4	α^{13}	α	α^{12}	α^9	α^{10}	α^8	α^0
α^4	α^4	α	α^0	α^{10}	α^7	$\mathbf{0}$	α^8	α^{12}	α^3	α^5	α^{14}	α^2	α^{13}	α^6	α^{11}	α^9
α^5	α^5	α^{10}	α^2	α	α^{11}	α^8	$\mathbf{0}$	α^9	α^{13}	α^4	α^6	α^0	α^3	α^{14}	α^7	α^{12}
α^6	α^6	α^{13}	α^{11}	α^3	α^2	α^{12}	α^9	$\mathbf{0}$	α^{10}	α^{14}	α^3	α^7	α	α^4	α^0	α^8
α^7	α^7	α^9	α^{14}	α^{12}	α^4	α^3	α^{13}	α^{10}	$\mathbf{0}$	α^{11}	α^0	α^6	α^8	α^2	α^9	α
α^8	α^8	α^2	α^{10}	α^0	α^{13}	α^5	α^4	α^{14}	α^{11}	$\mathbf{0}$	α^{12}	α	α^7	α^9	α^3	α^6
α^9	α^9	α^7	α^3	α^{11}	α	α^{14}	α^6	α^5	α^0	α^{12}	$\mathbf{0}$	α^{13}	α^2	α^8	α^{10}	α^4
α^{10}	α^{10}	α^5	α^8	α^4	α^{12}	α^2	α^0	α^7	α^6	α	α^{13}	$\mathbf{0}$	α^{14}	α^3	α^9	α^{11}
α^{11}	α^{11}	α^{12}	α^6	α^9	α^5	α^{13}	α^3	α	α^8	α^7	α^2	α^{14}	$\mathbf{0}$	α^0	α^4	α^{10}
α^{12}	α^{12}	α^{11}	α^{13}	α^7	α^{10}	α^6	α^{14}	α^4	α^2	α^9	α^8	α^3	α^0	$\mathbf{0}$	α	α^5
α^{13}	α^{13}	α^6	α^{12}	α^{14}	α^8	α^{11}	α^7	α^0	α^5	α^3	α^{10}	α^9	α^4	α	$\mathbf{0}$	α^2
α^{14}	α^{14}	α^3	α^7	α^{13}	α^0	α^9	α^{12}	α^8	α	α^6	α^4	α^{11}	α^{10}	α^5	α^2	$\mathbf{0}$

Fie acum $a(X) = a_0 + a_1X + \ldots + a_{k-1}X^{k-1} \in Z_q[X]$. Valoarea $a(\alpha)$ se obţine folosind circuitul de împărţire dat în Figura 8.3, cu următoarele observaţii:

- Se ignoră ieşirea;
- Coeficienţii lui $a(X)$ se introduc la intrare în ordinea descrescătoare a puterilor;
- Iniţial, elementele de înmagazinare conţin valorile $(0, 0, \ldots, 0)$.

Pentru calculul unei valori de forma $a(\alpha)\alpha^j \quad (j \geq 0$ fixat), se pot construi circuite liniare care să realizeze direct acest produs.

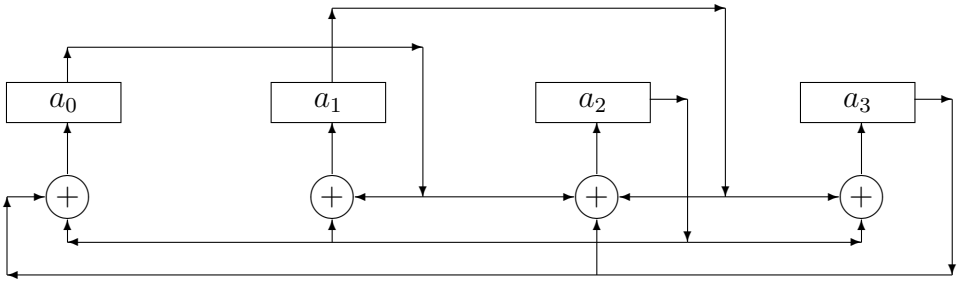
Exemplul 8.10 *Fie $\alpha \in GF(2^4)$ element primitiv, soluţie a ecuaţiei $1 + X + X^4 = 0$. Polinomul $a(X)$ va avea atunci gradul maxim 3. Fie $a(X) = a_0 + a_1X + a_2X^2 + a_3X^3$.*

Să construim un circuit liniar care să efectueze $a(\alpha)\alpha^5$.

Folosind tabelul din Exemplul 8.8, avem:

$(a_0 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3)\alpha^5 = a_0\alpha^5 + a_1\alpha^6 + a_2\alpha^7 + a_3\alpha^8 = a_0(\alpha + \alpha^2) + a_1(\alpha^2 + \alpha^3) + a_2(1 + \alpha + \alpha^2) + a_3(1 + \alpha^2) = (a_2 + a_3) + (a_0 + a_2)\alpha + (a_0 + a_1 + a_3)\alpha^2 + (a_1 + a_2)\alpha^3$

Pentru acest polinom se va construi circuitul liniar:



La momentul iniţial, în elementele de înmagazinare se găsesc coeficienţii lui $a(\alpha)$; după un tact, aici vor fi coeficienţii polinomului $a(\alpha)\alpha^5$.

8.3 Exerciţii

8.1 *Să se realizeze circuite liniare (în ambele modalităţi de construcţie) care să efectueze înmulţiri cu polinoamele:*

$$h(X) = 1 + 2X + 3X^2 + 4X^3 \in Z[X]$$
$$h(X) = 1 + 2X + X^5 \in Z_7[X]$$

8.2 *Să se reprezinte sub formă de tabel comportamentul circuitelor realizate anterior, la înmulţirea cu polinoamele:*

$a(X) = 2 + 5X + X^3$, $a(X) = X - X^3 - 2X^4$, $a(X) = 0$.

8.3 *Se dă vectorul* $\mathbf{h} = (1, 0, 0, -1, 0, 1)$. *Să se realizeze circuitul de înmulţire cu polinomul corespunzător şi să se reprezinte comportamentul acestui circuit la înmulţirea cu polinoamele reprezentate prin vectorii* $\mathbf{a} = (1)$ *şi* $\mathbf{a} = (1, 1, 1, 0, 0, 0, 1)$.

8.4 *Să se realizeze un circuit liniar care realizează produsul* $a(X)g(X)+b(X)h(X)$ *pentru polinoamele:*

$$g(X) = 1 + X + X^3, \quad h(X) = 1 - 5X - 2X^2 + 3X^3;$$
$$g(X) = -4X^3 + X^5 - 2X^8, \quad h(X) = X^2 + 2X^4 + X^6;$$
$$g(X) = X - X^3 + 3X^4, \quad h(X) = 2X^3 + X^5 + X^9.$$

8.5 *Să se facă împărţirea în* $Q[X]$ *la polinomul* $h(X) = X^2 - 2X + 1$ *a polinoamelor* $a(X) = X^5 - 4X^3 - 2X + 5$, $a(X) = X + 7$.

8.6 *Aceeaşi problemă pentru polinoamele* $h(X) = 2X - 1$ *şi respectiv* $a(X) = X^3 - 3$.
(1) *În* $Q[X]$;
(2) *În* $Z_3[X]$.

8.7 *Să realizeze un circuit care să efectueze (în* Q*) înmulţirea cu polinomul* $g(X) = 1 - 2X + X^3$ *şi împărţirea cu polinomul* $h(X) = g(X)$.

8.8 *Aceeaşi problemă pentru polinoamele:*

$$g(X) = X - 2X^3 - X^4, \quad h(X) = 2 + X + X^3 \quad \text{în } Z_5[X];$$
$$g(X) = -1 + X - X^6, \quad h(X) = X^2 - 4X^3 - 2X^5 + 3X^7 \quad \text{în } Z_{11}[X].$$

8.9 *Să se rezolve problema enunţată în Exemplul 18.2.*

8.10 *Să se genereze toate puterile lui* α, *rădăcină a polinomului*
(1) $1 + X^2 + X^5 \in Z_2[X]$
(2) $1 + X + X^3 \in Z_3[X]$.

8.11 *Folosind elementele din Exemplul 9.8, să se construiască circuite de generare pentru* $a(\alpha)\alpha$, $a(\alpha)\alpha^2$ *şi* $a(\alpha)\alpha^7$.

8.12 *Să se construiască circuite de generare în* $GF(2^5)$ *pentru* $a(\alpha)$ *şi* $a(\alpha)\alpha^{10}$, *unde* α *este rădăcină a polinomului* $1 + X^2 + X^5 \in Z_2[X]$.

Capitolul 9

Coduri ciclice

9.1 Relaţii de recurenţă liniară

Teorema 9.1 *Fie q un număr prim şi $f(X) \in Z_q[X]$, $grad(f(X)) = n$. În algebra polinoamelor modulo $f(X)$, fie I un ideal şi $g(X)$ polinomul de grad minim a cărui clasă de resturi aparţine lui I : $\{g(X)\} \in I$. Atunci:*

1. $\{s(X)\} \in I \iff g(X)|s(X)$;
2. $g(X)|f(X)$.

Demonstraţie:
(1) Fie $\{s(X)\} \in I$. Folosind identitatea împărţirii, avem

$$s(X) = b(X)g(X) + r(X), \quad grad(r(X)) < grad(g(X))$$

Trecând la clasele de resturi respective, se obţine $\{s(X)\} = \{b(X)\}\{g(X)\} + \{r(X)\}$. Cum $\{s(X)\} \in I$, rezultă $\{b(X)\}\{g(X)\} \in I$ (deoarece I este ideal), deci $\{r(X)\} \in I$, absurd, deoarece $g(X)$ este polinomul de grad minim a cărui clasă de resturi aparţine idealului I. Singura variantă rămâne $r(X) \equiv 0$, adică $s(X) = b(X)g(X)$.

Reciproca este imediată: din $s(X) = b(X)g(X)$ rezultă $\{s(X)\} = \{b(X)\}\{g(X)\} \in I$.
(2) Prin împărţirea polinomului $f(X)$ cu $g(X)$ se obţine

$$f(X) = c(X)g(X) + r(X) \quad \text{unde} \quad grad(r(X)) < grad(g(X)).$$

Avem $\mathbf{0} = \{f(X)\} = \{c(X)\}\{g(X)\} + \{r(X)\}$ de unde rezultă $\{r(X)\} \in I$, absurd; deci $r(X) \equiv 0$. q.e.d.

Teorema 9.2 *În algebra polinoamelor modulo $f(X)$, pentru orice ideal I există un polinom normat unic $g(X)$ de grad minim, cu $\{g(X)\} \in I$.*

Reciproc, pentru orice divizor normat al lui $f(X)$ există un ideal generat de acel divizor.

$g(X)$ se numeşte *generatorul* idealului I şi scriem $I = (\{g(X)\})$.

Demonstraţie: Fie $h(X)$ un polinom de grad minim cu $\{h(X)\} \in I$; vom nota $g(X) = h_n^{-1}h(X)$ unde h_n este coeficientul termenului de grad maxim din polinomul $h(X)$.

Să presupunem că $g(X)$, $g'(X)$ sunt două astfel de polinoame de grad minim, ale căror clase de resturi sunt în I. Conform Teoremei 20.1, $g(X)$ şi $g'(X)$ se divid unul pe altul

şi – având acelaşi grad – diferă printr-o constantă. Polinoamele fiind normate, această constantă este 1. Deci $g(X) = g'(X)$.

Reciproc, fie $g(X)$ un divizor normat al polinomului $f(X)$ şi $(\{g(X)\})$ idealul generat de el. Un element al acestui ideal este $\{a(X)\} \in (\{g(X)\})$, deci are forma $\{a(X)\} = \{b(X)\}\{g(X)\} = \{b(X)g(X)\}$. q.e.d.

Teorema 9.3 *Fie q un număr prim, $f(X) \in Z_q[X]$ un polinom normat, cu $grad(f(X)) = n$ şi fie $f(X) = g(X)h(X)$. Dacă $grad(h(X)) = k$, atunci în algebra polinoamelor modulo $f(X)$, idealul $(\{g(X)\})$ are dimensiunea k.*

Demonstraţie: Să observăm că vectorii (asociaţi polinoamelor)

$$\{g(X)\},\ \{Xg(X)\},\ \ldots,\ \{X^{k-1}g(X)\}$$

sunt liniar independenţi. Într-adevăr, pentru orice k elemente $c_0, \ldots, c_{k-1} \in Z_q$, nu toate nule, avem

$$c_0\{g(X)\} + c_1\{Xg(X)\} + \ldots + c_{k-1}\{X^{k-1}\} = \{(c_0 + c_1X + \ldots + c_{k-1}X^{k-1})g(X)\} \neq 0$$

deoarece s-a obţinut un polinom de grad cel mult $n - k + k - 1 = n - 1 < n$.

În acelaşi timp, pentru orice $\{s(X)\} \in (\{g(X)\})$ avem $\{s(X)\} = \{a(X)g(X)\} = \{(a_0 + a_1X + \ldots + a_{k-1}X^{k-1})g(X)\} = a_0\{g(X)\} + a_1\{Xg(X)\} + \ldots + a_{k-1}\{X^{k-1}g(X)\}$, unde unii din coeficienţii $a_0, a_1, \ldots, a_{k-1}$ pot fi nuli.

Deci vectorii de sus formează o bază a subspaţiului liniar $(\{g(X)\})$, care are deci dimensiunea k. q.e.d.

Teorema 9.4 *Fie $f(X) \in Z_q[X]$ un polinom normat şi $f(X) = g(X)h(X)$. În algebra polinoamelor modulo $f(X)$,*

$$\{a(X)\} \in (\{g(X)\}) \Longleftrightarrow \{a(X)\} \textit{ este în spaţiul nul al idealului } (\{h(X)\}).$$

Demonstraţie: "$\Longrightarrow$": Fie $\{a(X)\} \in (\{g(X)\})$, deci $a(X) = v(X)g(X)$.

Fie de asemenea $\{b(X)\} \in (\{h(X)\})$, deci $b(X) = w(X)h(X)$. Vom avea $\{a(X)\}\{b(X)\} = \{v(x)w(X)g(X)h(X)\} = \{v(X)w(X)f(X)\} = \{v(X)w(X)\}\{f(X)\} = \mathbf{0}$.

"$\Longleftarrow$": Să considerăm $\{a(X)\}$ în spaţiul nul al idealului $(\{h(X)\})$. Atunci $\{a(X)\}\{h(X)\} = \mathbf{0}$, adică $a(X)h(X) = c(X)f(X) = c(X)g(X)h(X)$ de unde rezultă $a(X) = c(X)g(X)$. Deci $\{a(X)\} \in (\{g(X)\})$. q.e.d.

Definiţia 9.1 *Se numeşte relaţie de recurenţă liniară egalitatea*

$$\sum_{j=0}^{k} h_j a_{i+j} = 0, \tag{1}$$

unde $i \geq 0,\ h_k = 1,\ h_0 \neq 0,\ h_j \in Z_q,\ a_{i+j} \in Z_q$.

Relaţiile de recurenţă liniară se pot scrie şi

$$a_{i+k} = -\sum_{j=0}^{k-1} h_j a_{i+j}, \quad i = 0, 1, \ldots$$

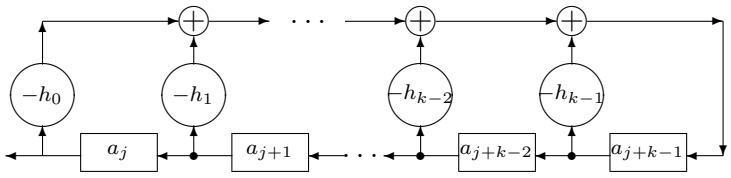
O soluţie a unei astfel de relaţii de recurenţă liniară este orice succesiune infinită de forma $a_0, a_1, \ldots, a_p, \ldots$ care verifică relaţia dată, cu $h_0, h_1, \ldots, h_k$ fixate, $h_0 \neq 0, h_k = 1$. Relaţia va determina succesiv pe a_k din $a_0, \ldots, a_{k-1}$, apoi pe a_{k+1} din $a_1, \ldots, a_k$ ş.a.m.d.

Altfel spus, "condiţiile iniţiale" $a_0, a_1, \ldots, a_{k-1}$ determină o soluţie a relaţiei de recurenţă liniară.

Observaţia 9.1

- *Orice combinaţie liniară de soluţii ale unei relaţii de recurenţă liniară este tot o soluţie.*
- *Soluţiile pentru care condiţiile iniţiale sunt respectiv* $(1, 0, \ldots, 0)$, $(0, 1, \ldots, 0), \ldots,$ $(0, 0, \ldots, 1)$ *determină orice altă soluţie. Deci, spaţiul soluţiilor relaţiei de recurenţă* (1) *are dimensiunea cel mult* k.

Fiind date condiţiile iniţiale (arbitrare) $a_0, a_1, \ldots, a_{k-1}$, soluţia relaţiei de recurenţă liniară (1) corespunzătoare lor se poate obţine folosind circuitul liniar



în care, la momentul iniţial, în elementele de înmagazinare se găsesc $a_0, a_1, \ldots, a_{k-1}$.

Să considerăm acum polinomul (fixat) $h(X) \in Z_q[X]$, $h(X) = h_0 + h_1X + \ldots + h_kX^k$, $h_0 \neq 0, h_k = 1$, şi fie n cel mai mic număr natural pentru care $X^n - 1$ se divide cu $h(X)$. Notăm

$$g(X) = \frac{X^n - 1}{h(X)} \tag{2}$$

Pe baza acestor date construim relaţia de recurenţă liniară

$$a_{i+k} = -\sum_{j=0}^{k-1} h_j a_{i+j},\ i \geq 0 \tag{3}$$

Demonstraţia existenţei unui astfel de număr n se face folosind principiul cutiei. Găsim două numere n_1, n_2 $(n_1 > n_2)$ pentru care polinoamele $X^{n_1} - 1$ şi $X^{n_2} - 1$ dau acelaşi rest la împărţirea cu $h(X)$. Va rezulta că X^{n_2} $(X^{n_1 - n_2} - 1)$ se divide cu $h(X)$. Cum $h(X)$ are termen liber nenul, el va divide pe $X^{n_1 - n_2} - 1$.

Teorema 9.5

1. *Soluţiile relaţiei de recurenţă* (3) *sunt periodice, de perioadă* n.
2. *Mulţimea formată din prima perioadă a fiecărei soluţii, scrisă ca polinom* $a(X) = a_0X^{n-1} + \ldots + a_{n-2}X + a_{n-1}$ *constituie idealul* $(\{g(X)\})$ *în algebra polinoamelor modulo* $X^n - 1$.

Demonstraţie: (a) Vom arăta că oricărui element $\{a(X)\} \in (\{g(X)\})$ cu $a(X) = a_0X^{n-1} + \ldots + a_{n-2}X + a_{n-1}$ îi corespunde o soluţie periodică

$$a_1, a_1, \ldots, a_{n-1}, a_0, a_1, \ldots, a_{n-1}, a_0, \ldots$$

a relaţiei de recurenţă (3). Evident, nu este obligatoriu ca toţi coeficienţii a_i să fie nenuli, gradul real al polinomului $f(X)$ putând fi chiar zero.

Fie $\{a(X)\}\{h(X)\} = \{c(X)\}$. Vom nota $c(X) = c_0 + c_1X + \ldots + c_{n-1}X^{n-1}$. Avem:

- pentru $k \leq p \leq n-1$, $\quad c_p = h_0a_{n-1-p} + h_1a_{n-1-p+1} + \ldots + h_ka_{n-1-p+k}$; (4)

- pentru $0 \leq p < k$, $\quad c_p = h_0a_{n-1-p} + h_1a_{n-1-p+1} + \ldots + h_pa_{n-1} + h_{p+1}a_0 + \ldots + h_ka_{k-p-1}$. (5)

Conform Teoremei 11.6, dacă $\{a(X)\} \in (\{g(X)\})$ atunci $\{a(X)\}\{h(X)\} = \mathbf{0}$, adică $c_p = 0 \quad (0 \leq p \leq n-1)$.

Considerând $(a_i)_i$ ca o succesiune periodică, vom avea evident $a_{i+n} = a_i,\ i = 0, 1, \ldots$. Ţinând cont de aceasta introducând $c_p = 0$ în (4) şi (5) se obţine (3).

(b) Idealul $(\{g(X)\})$ are dimensiunea $k = grad(h(X))$ (Teorema 20.5) şi fiecare element al acestui ideal (polinom sau vector – după cum este scris) va da (conform cu (a)) o soluţie a relaţiei de recurenţă liniară (3). Dar spaţiul soluţiilor relaţiei (3) are dimensiunea cel mult k. Deci idealul $(\{g(X)\})$ va da toate soluţiile relaţiei de recurenţă liniară (3). Mai trebuie arătat că perioada acestor soluţii este chiar n.

Din cele de până acum a rezultat că perioada este cel mult n. Vom arăta că soluţia corespunzătoare clasei de resturi $\{g(X)\}$ are perioada n.

Să presupunem prin absurd că soluţia corespunzătoare lui $\{g(X)\}$ are perioada $m < n$. Dar n este divizibil cu m şi deci coeficienţii polinomului $g(X)$ – considerat ca fiind de grad $n-1$ (prin completare cu coeficienţi nuli) formează $\dfrac{n}{m}$ blocuri care se repetă.

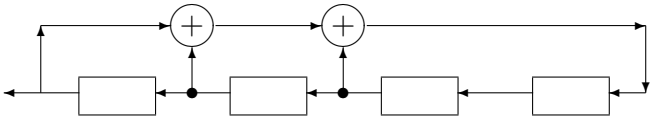
Deci $g(X) = q(X)(1 + X^m + X^{2m} + \ldots + X^{n-m})$ unde $q(X)$ este un polinom de gradul $m-1$. Relaţia se poate scrie şi $g(X) = q(X)\dfrac{X^n - 1}{X^m - 1}$.

Vom avea acum $(X^n - 1)(X^m - 1) = g(X)h(X)(X^m - 1) = q(X)h(X)(X^n - 1)$ adică $X^m - 1 = q(X)h(X)$, ceea ce contrazice condiţia că n este cel mai mic număr pentru care $X^n - 1$ se divide cu $h(X)$. Deci $m = n$. q.e.d.

Exemplul 9.1 *Să considerăm polinomul* $h(X) = 1 + X + X^2 + X^4 \in Z_2[X]$. *Relaţia de recurenţă liniară asociată este*

$$a_{i+4} = a_{i+2} + a_{i+1} + a_i, \quad (i \geq 0).$$

Circuitul liniar corespunzător are forma



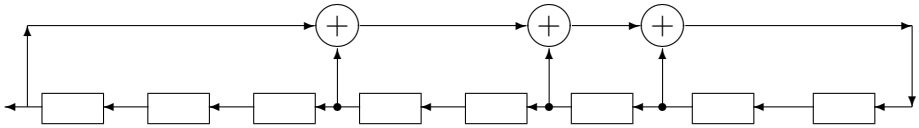
Cel mai mic n *pentru care* $X^n - 1$ *se divide cu* $h(X)$ *este* $n = 7$. *Cum* $g(X) = \dfrac{X^7 - 1}{h(X)} = X^3 + X + 1$, *rezultă că circuitul va genera cuvintele idealului* $(\{g(X)\})$. *Fiecare cuvânt este caracterizat de cele patru valori binare iniţiale din elementele de înmagazinare. Vor fi deci* $2^4 = 16$ *cuvinte de lungime* 7. *Ele corespund tuturor polinoamelor de grad maxim* 6 *din* $Z_2[X]$, *care se divid cu* $g(X)$.

De exemplu, pentru valorile iniţiale $(1, 0, 1, 1)$*, funcţionarea circuitului timp de şapte tacţi este:*

Ieşire				
–	1	0	1	1
1	0	1	1	0
0	1	1	0	0
1	1	0	0	0
1	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	1	0	1	1

În elementele de înmagazinare se regăsesc valorile iniţiale, iar la ieşire s-a obţinut polinomul $f(X) = X^6 + X^4 + X^3 = X^3 g(X)$.

Exemplul 9.2 *Circuitul liniar corespunzător polinomului* $h(X) = 1 + X^3 + X^5 + X^6 + X^8 \in Z_2[X]$ *este*



El dă soluţiile relaţiei de recurenţă liniară $a_i + a_{i+3} + a_{i+5} + a_{i+6} + a_{i+8} = 0$, *care formează idealul generat de polinomul* $g(X) = \dfrac{X^{255} - 1}{h(X)}$ *ideal compus din* $2^8 = 256$ *cuvinte de lungime* 255.

9.2 Definirea codurilor ciclice

Fie n $(n \geq 2)$ un număr natural. Să notăm cu A_n algebra polinoamelor din $Z_q[X]$, modulo $X^n - 1$. După cum s-a convenit, identificăm cuvântul $a_0 a_1 \ldots a_{n-1}$ cu vectorul $(a_0, a_1, \ldots, a_{n-1})$ şi cu polinomul $a(X) = a_0 + a_1 X + \ldots + a_{n-1} X^{n-1} \in Z_q[X]$.

În cadrul dualismului *vector - polinom* vom face o deosebire: anularea produsului a două polinoame nu înseamnă ortogonalitatea vectorilor corespunzători. Pentru această situaţie se foloseşte următoarea propoziţie:

Propoziţia 9.1 *Produsul a două polinoame este zero dacă şi numai dacă toate produsele scalare dintre vectorul unui polinom şi permutările ciclice ale vectorului celuilalt polinom sunt zero.*

Demonstraţie: Fie $a(X), b(X) \in Z_q[X],\ a(X) = \sum_{i=0}^{n-1} a_i X^i,\ b(X) = \sum_{i=0}^{n-1} b_i X^i$.

Atunci $\{c(X)\} = \{a(X)\}\{b(X)\} = \{c_0 + c_1 X + \ldots + c_{n-1} X^{n-1}\}$ unde

$c_j = \sum_{i=0}^{j} a_i b_{j-i} + \sum_{i=j+1}^{n-1} a_i b_{j+n-i} = (a_0, a_1, \ldots, a_{n-1}) \cdot (b_j, b_{j-1}, \ldots b_0, b_{n-1}, b_{n-2}, \ldots, b_{j+1})^T$

şi Propoziţia rezultă din faptul că $c_j = 0\ \forall j$. q.e.d.

Definiţia 9.2 *Un subspaţiu liniar $V_n \subseteq A_n$ se numeşte ciclic dacă*

$$(a_0, a_1, \ldots, a_{n-1}) \in V_n \qquad \Longrightarrow \qquad (a_{n-1}, a_0, \ldots, a_{n-2}) \in V_n.$$

Teorema 9.6 *Un subspaţiu liniar $V_n \subseteq A_n$ este ciclic dacă şi numai dacă este ideal.*

Demonstraţie: Înmulţirea cu clasa de resturi $\{X\}$ înseamnă de fapt permutarea ciclică a componentelor cu o unitate spre dreapta, pentru că:

$$\{X\}\{a_0 + a_1X + \ldots + a_{n-1}X^{n-1}\} = \{a_{n-1} + a_0X + \ldots + a_{n-2}X^{n-1}\}.$$

"$\Longleftarrow$": Dacă $V_n \subseteq A_n$ este ideal, atunci pentru orice $\mathbf{v} \in V_n$ avem $\mathbf{v}' = \{X\}\mathbf{v} \in V_n$ (s-a notat tot cu $\mathbf{v}$ clasa de resturi modulo $X^n - 1$ corespunzătoare polinomului ai cărui coeficienţi sunt componentele vectorului $\mathbf{v}$). Deci V_n este ciclic.

"$\Longrightarrow$": Presupunem că subspaţiul liniar $V_n \subseteq A_n$ este ciclic şi fie $\mathbf{v} \in V_n$. Atunci, din $\{X\}\mathbf{v} \in V_n$ rezultă $\{X^j\}\mathbf{v} \in V_n$, $\forall j = 1, \ldots, n-1$.

Deci $\{c_0 + c_1X + \ldots + c_{n-1}X^{n-1}\}\mathbf{v} \in V_n$, adică V_n este ideal în A_n. q.e.d.

Definiţia 9.3 *Se numeşte cod ciclic un ideal propriu $L \subset A_n$.*

Codurile ciclice au fost introduse de Prange (1957), care a evidenţiat bogăţia de informaţie rezultată din structura lor algebrică.

9.3 Generarea codurilor ciclice

Pentru a construi un cod ciclic se foloseşte structura idealelor în algebra polinoamelor modulo $X^n - 1$. Fie I un ideal propriu în A_n şi $g(X)$ polinomul normat de grad minim cu $\{g(X)\} \in I$. Atunci (Teorema 20.1) $\{a(X)\} \in I$ dacă şi numai dacă $g(X)|a(X)$.

De asemenea, $g(X)|X^n - 1$. Oricărui divizor $g(X)$ al lui $X^n - 1$ ($grad(g(X)) < n$) îi corespunde un ideal propriu în A_n, generat de $g(X)$.

Pentru a defini un cod ciclic este suficient să dăm generatorul său $g(X)$, divizor al lui $X^n - 1$. Fie $g(X) = g_0 + g_1X + \ldots + g_{n-k}X^{n-k}$ ($k < n$).
O bază a idealului ($\{g(X)\}$) se poate alege (Teorema 20.5)

$$\{g(X)\},\ \{Xg(X)\}, \ldots, \{X^{k-1}g(X)\}.$$

Matricea generatoare corespunzătoare acestui cod ciclic va fi

$$G_{k,n} = \begin{pmatrix} g_0 & g_1 & \ldots & g_{n-k} & 0 & 0 & \ldots & 0 \\ 0 & g_0 & \ldots & g_{n-k-1} & g_{n-k} & 0 & \ldots & 0 \\ & & & & \vdots & & & \\ 0 & 0 & \ldots & 0 & g_0 & g_1 & \ldots & g_{n-k} \end{pmatrix}$$

Rezultă că un cod ciclic poate fi organizat ca un (n, k) - cod liniar, unde n este gradul polinomului $X^n - 1$ iar k este gradul polinomului $h(X) = \dfrac{X^n - 1}{g(X)}$.

Exemplul 9.3 *Codul cu repetiţie este un cod ciclic al cărui polinom generator este $g(X) = 1 + X + X^2 + \ldots + X^{n-1}$.*

Exemplul 9.4 *Să considerăm codul ciclic binar de lungime* 7*, cu polinomul generator* $g(X) = 1 + X + X^3$ *(vezi şi Exemplul 9.1). El are matricea generatoare*

$$G = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

Exemplul 9.5 *Fie codul ciclic de lungime* 6 *peste* Z_3*, de polinom generator* $g(X) = 2 + X^2$*. Matricea sa generatoare este*

$$G = \begin{pmatrix} 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 1 \end{pmatrix}.$$

El este deci un $(6, 4)$ *- cod ternar.*

Sunt 14 *coduri ciclice ternare de lungime* 6*, obţinute din factorii polinomului* $X^6 - 1 = (X + 1)^3(X + 2)^3$*. Este un exerciţiu interesant obţinerea matricilor generatoare pentru toate aceste coduri.*

Idealul generat de $\{g(X)\}$ este spaţiul nul al idealului $(\{h(X)\})$ unde $h(X) = \dfrac{X^n - 1}{g(X)}$. Acest ideal $(\{h(X)\})$ se construieşte luând ca bază polinoamele

$$\{h(X)\},\ \{Xh(X)\} \ldots, \{X^{n-k-1}h(X)\}.$$

Ţinând cont de Propoziţia 9.1, putem construi matricea de control H a codului ciclic $(\{g(X)\})$, luând drept linii ale matricii cele $n - k$ polinoame de sus, cu ordinea componentelor inversată.

Exemplul 9.6 *Să reluăm codul din Exemplul 9.4. Deoarece* $X^7 - 1 = (1 + X)(1 + X + X^3)(1+X^2+X^3)$*, pentru acest cod, avem* $h(X) = (1+X)(1+X+X^3) = 1+X^2+X^3+X^4$*.*

Matricea generatoare a codului $(\{h(X)\})$ *este*

$$G = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

deci matricea de control asociată codului din Exemplul 9.4 va fi

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

Deoarece coloanele sale sunt nenule şi distincte două câte două, codul astfel construit este echivalent cu un $(7, 4)$ *cod Hamming binar.*

Pentru a obţine un cod ciclic sistematic, este convenabil să alegem o altă bază pentru idealul $(\{g(X)\})$. Să observăm că pentru $i = n - k, n - k + 1, \ldots, n - 1$, putem scrie

$$X^i = q_i(X)g(X) + r_i(X) \text{ cu } grad(r_i(X)) < grad(g(X)) = n - k.$$

Deci $\{X^i - r_i(X)\} \in (\{g(X)\}), \quad i = n - k, n - k + 1, \ldots, n - 1.$

Această mulţime de vectori este liniar independentă şi conduce la o matrice generatoare de forma

$$G = [-R\ I]$$

unde linia j din matricea R este vectorul coeficienţilor lui $r_j(X)$ pentru $j = n-k, \ldots, n-1$. Codul obţinut este deci sistematic, iar matricea sa de control se scrie imediat:

$$H = [I\ R^T].$$

De remarcat că matricea H^T are pe linii componentele resturilor $r_i(X)$ pentru $i = 0, 1, \ldots, n-1$.

Exemplul 9.7 *Să luăm din nou* $X^7 - 1 = g(X)h(X)$ *peste* Z_2, *unde* $g(X) = 1 + X^2 + X^3$, $h(X) = 1 + X^2 + X^3 + X^4$ *(deci* $n = 7$, $k = 4$*). Avem*

$$\begin{array}{rcrcrcrcr}
X^0 & = & 0g(X) & + & 1 & & & & \\
X^1 & = & 0g(X) & & & + & X & & \\
X^2 & = & 0g(X) & & & & & + & X^2 \\
X^3 & = & g(X) & + & 1 & & & + & X^2 \\
X^4 & = & (1+X)g(X) & + & 1 & + & X & + & X^2 \\
X^5 & = & (1+X+X^2)g(X) & + & 1 & + & X & & \\
X^6 & = & (X+X^2+X^3)g(X) & & & + & X & + & X^2
\end{array}$$

Baza corespunzătoare idealului $(\{g(X)\})$ *este*

$$\{1 + X^2 + X^3\},\ \{1 + X + X^2 + X^4\},\ \{1 + X + X^5\},\ \{X + X^2 + X^6\}$$

care conduce la matricea generatoare a unui cod sistematic:

$$G_{4,7} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} = [-R_{4,3}\ I_4].$$

Matricea de control corespunzătoare se scrie simplu:

$$H_{3,7} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix} = [I_3\ R_{3,4}^T].$$

9.4 Implementarea generării codurilor ciclice

Codurile ciclice sunt coduri a căror implementare este mult facilitată de circuitele liniare. Cu ajutorul acestora se poate realiza automat codificarea, calculul sindromului, detectarea şi corectarea erorilor.

În construcţia practică vom distinge două cazuri:

9.4.1 Circuit cu k elemente de înmagazinare

Este o metodă de generare a codurilor ciclice, avantajoasă dacă sunt mai puţine simboluri de informaţie decât simboluri de control: $k < n - k$.

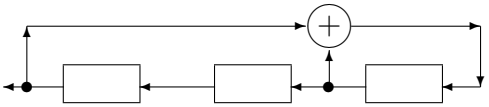
Fie codul ciclic $(\{g(X)\}) \subset A_n$ şi $h(X) = \dfrac{X^n - 1}{g(X)} = h_0 + h_1 X + \ldots + h_k X^k$ cu $h_0 \neq 0,\ h_k = 1$. Conform Teoremei 9.5, idealul $(\{g(X)\})$ este generat de circuitul liniar care construieşte soluţiile relaţiei de recurenţă liniară

$$\sum_{j=0}^{k} h_j a_{i+j} = 0, \quad i = 0, 1, \ldots$$

Mesajul de informaţie care trebuie codificat – conţinând k simboluri pe fiecare bloc – se introduce la momentul iniţial în elementele de înmagazinare ale circuitului, sub formă de "condiţii iniţiale". Lăsând circuitul să funcţioneze, obţinem după n momente cuvântul - cod corespunzător, aparţinând idealului $(\{g(X)\})$ şi având pe primele k poziţii elementele de informaţie.

Exemplul 9.8 *Circuitul liniar construit în Exemplul 9.1 este un circuit de codificare pentru codul generat de polinomul* $g(X) = 1 + X + X^3$. *Exemplul descrie şi un mod de funcţionare pentru cuvântul de informaţie* 1011.

Dacă s-ar lua drept polinom generator $1 + X^2 + X^3 + X^4$, *idealul generat de el este dat de circuitul liniar*



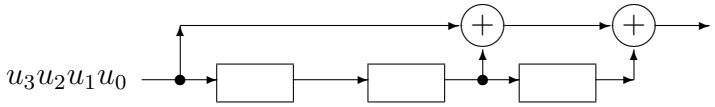
deoarece $h(X) = \dfrac{X^7 - 1}{1 + X^2 + X^3 + X^4} = 1 + X^2 + X^3$.

9.4.2 Circuit cu $n - k$ elemente de înmagazinare

Este o strategie avantajos de utilizat în cazul $n - k < k$.

Dacă interpretăm mesajul de informaţie ca un polinom de gradul $k - 1$, atunci codificarea se poate face utilizând un circuit de înmulţire cu polinomul generator $g(X)$ (vezi Prelegerea anterioară). La decodificare se recapătă mesajul iniţial (dacă nu au apărut erori) folosind un circuit de împărţire cu $g(X)$.

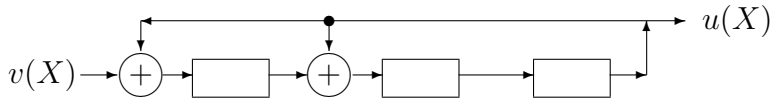
Exemplul 9.9 *Codul ciclic de polinom generator* $g(X) = 1 + X + X^3$ *poate codifica mesajele de informaţie folosind circuitul liniar:*



Astfel, mesajul de informaţie $\mathbf{u} = 1001$ *se codifică în* $\mathbf{v} = 1100101$, *conform calculelor:*

$$v(X) = u(X)g(X) = (1 + X^3)(1 + X + X^3) = 1 + X + X^4 + X^6$$

Pentru decodificare se foloseşte circuitul

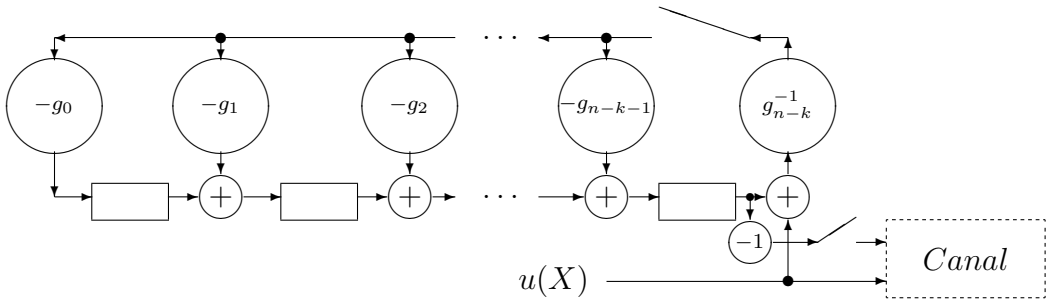


Dezavantajul unei asemenea metode îl constituie faptul că – nefiind un cod sistematic – prin codificare poziţiile de informaţie se pierd. Pentru a obţine un cod sistematic, procedăm în felul următor: mesajul de informaţie este considerat un polinom $u_0(X)$ de gradul $n-1$ având poziţiile de informaţie drept coeficienţii lui $X^{n-k},\ldots,X^{n-1}$, restul coeficienţilor fiind nuli. Atunci avem $u_0(X) = q(X)g(X) + r(X)$ cu $grad(r(X)) < grad(g(X)) = n-k$, de unde

$$\{u_0(X) - r(X)\} \in (\{g(X)\}).$$

$\{u_0(X) - r(X)\}$ reprezintă un cuvânt - cod în care coeficienţii lui $X^{n-k},\ldots,X^{n-1}$ sunt poziţiile de informaţie nealterate, iar coeficienţii lui $r(X)$ cu semnul schimbat (deci coeficienţii lui $X^0, X^1, \ldots, X^{n-k-1}$) sunt caracterele de control.

Circuitul care realizează această codificare este următorul:

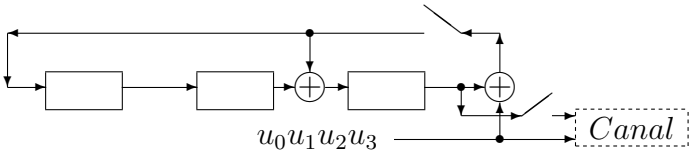


El funcţionează astfel:

- Iniţial cele $n-k$ elemente de înmagazinare conţin 0, întrerupătorul de ieşire este decuplat iar cel de feedback - cuplat.
- Se introduc cele k elemente ale mesajului de informaţie atât în circuit cât şi în canalul de comunicaţie. După k momente, în elementele de înmagazinare avem coeficienţii restului $r(X)$.
- Se decuplează feedbackul şi se cuplează circuitul la canalul de comunicaţie.
- Coeficienţii restului – cu semn schimbat – se transmit în canal imediat după poziţiile de informaţie.

Deoarece provine din circuitul de împărţire cu $g(X)$, acelaşi circuit poate fi folosit şi pentru detectarea erorilor la recepţie. Pentru aceasta, după decuplarea întrerupătorului de ieşire şi cuplarea celui de feedback, se introduce în circuit cuvântul recepţionat. Dacă la momentul n restul nu este nul (cel puţin un element de înmagazinare conţine un element nenul) înseamnă că a apărut cel puţin o eroare în transmisie.

Exemplul 9.10 *Codul ciclic binar de lungime* 7 *generat de* $g(X) = 1 + X^2 + X^3$*, poate fi construit folosind circuitul liniar:*



9.5 Exerciţii

9.1 *Să se determine toate codurile ciclice de lungimi* $n = 5$ *şi* $n = 6$
(a) *peste* Z_2; (b) *peste* Z_3.

9.2 *Construiţi o bază pentru cel mai mic cod ciclic de lungime* n *care conţine cuvântul:*
(a) $\mathbf{v} = 1101000, \quad n = 7;$
(b) $\mathbf{v} = 010101, \quad n = 6;$
(c) $\mathbf{v} = 11011000, \quad n = 8.$

9.3 *Pentru fiecare din cuvintele de mai jos, găsiţi polinomul generator al celui mai mic cod ciclic care conţine cuvântul respectiv:*

010010	01100110	0101100
001000101110000	000010010000000	010111010000000

9.4 *Să se afle polinomul generator al codului ciclic* C *ştiind o bază* S *a sa:*
$S = \{010, 011, 111\}$;
$S = \{1010, 0101, 1111\}$;
$S = \{0101, 1010, 1100\}$;
$S = \{1000, 0100, 0010, 0001\}$;
$S = \{11000, 01111, 11110, 01010\}$.

9.5 *Într-o codificare sistematică, codificaţi mesajul de informaţie* 1101 *într-un cu-vânt - cod al unui cod binar ciclic de lungime* 7 *cu polinomul generator* $g(X) = 1 + X^2 + X^3$.

9.6 *Pentru codul definit mai sus codificaţi mesajele de informaţie date de polinoamele:* $1 + X^2,\ X,\ X + X^2 + X^3$.

Fiind date cuvintele - cod $X^2 + X^4 + X^5,\ 1 + X + X^2 + X^4,\ X^2 + X^3 + X^4 + X^6$, *găsiţi mesajele de informaţie corepunzătoare.*

9.7 *Construiţi circuite liniare pentru codificare şi decodificare ale codului ciclic binar de lungime* 7 *generat de polinomul* $g(X) = (1 + X^2 + X^3)(1 + X)$.

9.8 *Care este lungimea minimă a unui cod ciclic binar de polinom generator* $g(X) = 1 + X^2 + X^3 + X^4$ *?*

Construiţi circuite liniare pentru codificarea şi decodificarea sa.

9.9 *Construiţi un circuit liniar pentru codificarea* $(15, 11)$ *- codului Hamming binar.*

9.10 *Construiţi matricea de control a codului ciclic binar de lungime* n *şi polinom generator* $g(X)$*:*
$n = 6, \quad g(X) = 1 + X^2$;
$n = 8, \quad g(X) = 1 + X^2$;
$n = 9, \quad g(X) = 1 + X^3 + X^6$;
$n = 15, \quad g(X) = 1 + X + X^4$ *(generează codul Hamming);*
$n = 23, \quad g(X) = 1 + X + X^5 + X^6 + X^7 + X^9 + X^{11}$ *(generează codul Golay);*
$n = 15, \quad g(X) = 1 + X^4 + X^6 + X^7 + X^8$.

9.11 *Sunt ciclice codurile liniare binare definite de matricile generatoare definite mai jos ?*

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

9.12 *Arătaţi că dacă polinomul generator al unui cod ciclic binar se divide cu polinomul* $1 + X$*, atunci toate cuvintele sale au pondere pară.*

Reciproca este adevărată ?

9.13 *Daţi o condiţie necesară şi suficientă pentru polinomul generator al unui cod ciclic de lungime impară pentru ca* $11 \ldots 1$ *să fie cuvânt - cod.*

9.14 *Arătaţi că dacă* $g(X) \in Z_q[X]$ *generează un* (n, k) *- cod ciclic, atunci* $g(X^p)$ *generează un* (pn, pk) *- cod ciclic.*

Descrieţi codurile ciclice binare pentru $g(X) = 1 + X$ *şi* $g(X) = 1 + X + X^3$ $(n = 7)$*.*

9.15 *Arătaţi că intersecţia a două coduri ciclice de aceeaşi lungime este tot un cod ciclic.*

Care este polinomul său generator ?

Capitolul 10

Decodificarea codurilor ciclice

10.1 Detectarea şi corectarea erorilor independente

După cum se ştie de la codurile liniare, o modalitate de caracterizare a tipurilor de erori care pot apare în transmisia mesajelor este dată pe baza sindromului. În cazul codurilor ciclice este convenabil să lucrăm cu un *sindrom polinomial*.

Definiţia 10.1 *Fie C un cod ciclic de lungime n, cu polinomul generator $g(X)$. Se numeşte "sindrom polinomial" $s(X)$ al cuvântului* $\mathbf{a}$ *de lungime n, restul împărţirii polinomului corespunzător $a(X)$ la $g(X)$.*

Deci, dacă se transmite cuvântul - cod $q(X)g(X)$ şi se primeşte $a(X)$, sindromul va fi

$$s(X) = a(X) - q(X)g(X), \qquad grad(s(X) < grad(g(X)).$$

Observaţia 10.1 *Fie* $\mathbf{e}$ *vectorul eroare apărut la transmisie.*

- *Deoarece $a(X) - e(X)$ este divizibil cu $g(X)$, rezultă că resturile împărţirii lui $a(X)$ şi $e(X)$ la $g(X)$ coincid. Deci se poate vorbi despre "tipul de eroare"* $\mathbf{e}$.
- *Similar cu raţionamentul folosit la codurile liniare, vor fi considerate sindromuri toate polinoamele din $Z_q[X]$ de grad $< n - k$; pentru fiecare sindrom $s(X)$ putem alege o eroare - tip $e(X)$ de pondere minimă care corespunde acelui sindrom.*

Structura algebrică a codurilor ciclice permite construcţia de metode de decodificare (cu corectarea posibilelor erori aferente) mai eficiente decât algoritmii similari de la codurile liniare.

Vom începe cu prezentarea unui exemplu.

Exemplul 10.1 *Fie C codul Hamming de lungime 7, care are polinomul generator $g(X) = 1 + X + X^3$ (Prelegerea 9, Exemplele 9.4 şi 9.6).*

Dacă se recepţionează cuvântul $\mathbf{a} = 1011001$*, sindromul lui este restul împărţirii lui $1 + X^2 + X^3 + X^6$ la $1 + X + X^3$, adică $s(X) = 1 + X$.*
Se ştie că orice cod Hamming corectează o eroare, deci va fi util să avem un tabel cu sindromurile tuturor erorilor - tip $e(X) = X^i$ $(0 \leq i \leq 6)$.

Eroare - tip	*Sindrom*
0	0
1	1
X	X
X^2	X^2
X^3	$1+X$
X^4	$X+X^2$
X^5	$1+X+X^2$
X^6	$1+X^2$

Pentru că $X+1$ este sindromul lui $e(X)=X^3$, tragem concluzia că a fost perturbat al treilea bit şi vom decodifica

$$a(X)-e(X)=1011001-0001000=1010001.$$

O dificultate în această metodă de decodificare bazată pe sindrom constă în necesitatea de a lista toate erorile - tip corespunzătoare sindromurilor, şi de a le parcurge la recepţia fiecărui cuvânt. O simplificare este însă propusă de Meggitt (1960), bazată pe structura ciclică a codurilor. Vom face corecţia numai pentru ultimul caracter (coeficientul termenului de grad maxim) al cuvântului **a** recepţionat. După aceea se efectuează o permutare ciclică şi se studiază din nou ultimul caracter.

În acest fel, după n permutări ciclice, toate poziţiile au fost corectate.

Metoda prezintă două avantaje majore:

1. Se foloseşte doar lista erorilor - tip de grad $n-1$. (Astfel, în Exemplul 20.3 o singură eroare-tip are gradul 6).

2. Calculul sindromului se efectuează o singură dată – la început – folosind circuitul liniar de împărţire cu $g(X)$. După ce a funcţionat n momente, în elementele de înmagazinare se găseşte sindromul. După aceea se neglijează ieşirea şi se lasă să funcţioneze circuitul. La fiecare pas (Prelegerea 8, Figura 8.4) sindromul se înmulţeşte cu X, această operaţie având ca efect permutarea sa ciclică cu o poziţie.

Propoziţia 10.1 *Dacă un cuvânt $\mathbf{a}=(a_0,a_1,\ldots,a_{n-1})$ are sindromul $s(X)$, atunci permutarea sa ciclică are sindromul $s'(X)$, care este restul împărţirii lui $Xs(X)$ la polinomul generator $g(X)$.*

Demonstraţie: Sindromul $s(X)$ este definit prin identitatea împărţirii

$$a(X)=q(X)g(X)+s(X)$$

unde $q(X)$ este câtul împărţirii lui $a(X)$ la $g(X)$. Deoarece se lucrează în algebra polinoamelor modulo X^n-1, permutarea ciclică a lui $\{a(X)\}$ este $\{a'(X)\}=\{Xa(X)\}$, construit astfel:

$$\begin{aligned}a'(X)&=Xa(X)-a_{n-1}X^n+a_{n-1}=Xa(X)-a_{n-1}(X^n-1)=\\&=Xa(X)-a_{n-1}g(X)h(X)=Xq(X)g(X)+Xs(X)-a_{n-1}g(X)h(X)=\\&=Xs(X)+g(X)[Xq(X)-a_{n-1}h(X)].\end{aligned}$$

Rezultă de aici că resturile împărţirii lui $a'(X)$ şi $Xs(X)$ la polinomul generator $g(X)$ sunt aceleaşi. q.e.d.

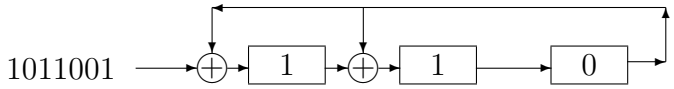
Se poate da acum

Algoritmul de decodificare Meggitt pentru (n, k) - codurile ciclice binare:

1. Se listează toate sindromurile corespunzătoare erorilor-tip **e** reprezentate prin polinoame de grad $n-1$;
2. Cuvântul recepţionat **a** este introdus în circuitul liniar de împărţire cu $g(X)$, care va funcţiona n tacţi;
3. Dacă în elementele de înmagazinare se obţine un polinom care se regăseşte în lista de sindormuri, se modifică cel mai din dreapta bit al cuvântului primit;
4. Se permută ciclic cuvântul primit şi – în acelaşi timp – se lasă să funcţioneze un tact circuitul liniar (ignorând ieşirea), după care se reia Pasul (3).

   Se repetă acest procedeu de $n-1$ ori.

Exemplul 10.2 *Reluând codul Hamming din Exemplul 20.3, algoritmul va lucra astfel:*
- Singurul sindrom din listă este $1+X^2$ *(de vecotr* 101*).*
- Dacă se recepţionează de exemplu $\mathbf{a} = 1011001$*, el se introduce în circuitul liniar*



După 7 *tacţi, în elementele de înmagazinare se obţine sindromul* 110*.*
- 110 *este diferit de* 101 *(singurul sindrom din listă), deci* a_6 *a fost recepţionat corect.*
- Se permută circular **a** *în* 1101100 *şi se lasă circuitul să funcţioneze un tact; elementele de înmagazinare vor conţine acum* 011*.*
- Nici acest sindrom nu este pe listă, deci a_5 *a fost şi el corect ş.a.m.d.*

Paşii de lucru ai algoritmului sunt reprezentaţi în tabloul:

Pas	*Bit*	*Sindrom*
7	a_6	110
8	a_5	011
9	a_4	111
10	a_3	**101**
11	a_2	100
12	a_1	010
13	a_0	001

Deci singurul bit corectat (unde se transformă 1 *în* 0*) este* a_3*. Mesajul primit este* 1010001*.*

Exemplul 10.3 *Să considerăm codul ciclic binar generat de polinomul* $g(X) = 1+X^4+X^6+X^7+X^8$*. El este un* $(15, 7)$ *- cod ciclic de distanţă minimă* $d = 5$ *(după cum vom vedea mai târziu) deci poate corecta cel mult* 2 *erori independente. Lista completă a sindromurilor (deci a tuturor erorilor - tip de pondere* $0, 1, 2$*) are* $C_{15}^0 + C_{15}^1 + C_{15}^2 = 121$ *elemente. Folosind algoritmul Meggitt, numărul lor se reduce la* 15*, anume*

Tip eroare	*Sindrom*
X^{14}	$X^7+X^6+X^5+X^3$
$1+X^{14}$	$X^7+X^6+X^5+X^3+1$
$X+X^{14}$	$X^7+X^6+X^5+X$
X^2+X^{14}	$X^7+X^6+X^5+X^3+X^2$
X^3+X^{14}	$X^7+X^6+X^5$
X^4+X^{14}	$X^7+X^6+X^5+X^4+X^3$
X^5+X^{14}	$X^7+X^6+X^3$
X^6+X^{14}	$X^7+X^5+X^3$
X^7+X^{14}	$X^6+X^5+X^3$
X^8+X^{14}	$X^5+X^4+X^3+1$
X^9+X^{14}	$X^7+X^4+X^3+X+1$
$X^{10}+X^{14}$	X^3+X^2+X
$X^{11}+X^{14}$	$X^7+X^6+X^5+X^4+X^2+X$
$X^{12}+X^{14}$	$X^7+X^6+X^4+X$
$X^{13}+X^{14}$	$X^7+X^4+X^3+X^2$

Deci, la recepţia unui cuvânt $\mathbf{a}$ *de lungime* 15*, vom calcula sindromul şi – dacă acesta se află în tabelul de sus – vom modifica bitul* a_{14}*. Apoi se face o permutare ciclică şi se corectează* a_{13}*, ş.a.m.d.*

De remarcat că în Exemplul 14.4 decodificarea nu este completă: algoritmul Meggitt corectează în total 121 erori - tip. Privit însă ca un cod liniar tabela de decodificare va avea $2^{15}/2^7 = 256$ linii, ceea ce înseamnă că acest cod are posibilitatea de a corecta mult mai multe erori. Cu algoritmul Meggitt se vor corecta toate erorile simple sau duble, ignorând complet posibilitatea existenţei altor tipuri de erori.

Vom încerca în continuare să îmbunătăţim această situaţie.

10.2 Pachete standard de erori

Fie $C \subseteq A_n$ un (n,k) - cod ciclic peste Z_q, capabil să corecteze t erori.

Definiţia 10.2 *Pentru un polinom* $e(X) = X^s e_0(X) \in Z_q[X]$ *cu* $e_0(0) = 1$*, spunem că "lungimea-pachet" a lui* $\mathbf{e}$ *este* $1 + grad(e_0(X))$*.*

Cuvântul $\mathbf{e} \in Z_q^n$ *este un "pachet standard de lungime* j*" dacă gradul minim al polinoamelor* $\{X^i e(X)\}$ $(0 \le i \le n-1)$ *este* $j-1$*.*

Dacă se trimite un cuvânt $\mathbf{v}$ şi se recepţionează $\mathbf{a}$, spunem că erorile au afectat j poziţii consecutive dacă eroarea - tip $\mathbf{e} = \mathbf{a} - \mathbf{v}$ este un pachet standard de lungime j.

Exemplul 10.4 *Fie* $n = 7$ *şi* $\mathbf{e} = 0101100$*. Atunci* $e(X) = X + X^3 + X^4 = X(1 + X^2 + X^3)$*. Efectuând toate permutările sale ciclice, se obţine* $\{X^6 e(X)\} = \{1 + X^2 + X^3\}$ *– corespunzător unui polinom de gradul* 3*; toate celelalte permutări conduc la polinoame de grad mai mic. Deci* $\mathbf{e}$ *este un pachet standard de lungime* 4*.*

Dacă se ia $\mathbf{e} = 1000100$*, polinomul asociat* $(1+X^4)$ *are gradul* 4*; efectuând permutările sale ciclice, se obţine* $\{X^3(1+X^4)\} = \{1+X^3\}$*, de gradul* 3*, deci şi acest* $\mathbf{e}$ *este un pachet standard de lungime* 4*.*

Reamintim că la codurile liniare se construia un tablou standard în care pe fiecare linie se aflau toate cuvintele cu acelaşi sindrom. Codul era corector de t erori independente dacă toate cuvintele din Z_q^n de pondere cel mult t erau aşezate pe linii diferite; aceste cuvinte constituiau reprezentanţii sindromurilor şi apăreau pe prima coloană, fiind considerate erori - tip pentru fiecare linie.

Astfel de tabele se pot construi şi pentru codurile ciclice; aici se iau ca reprezentanţi în fiecare linie, pachetele standard de erori de lungime minimă. În acest fel, un cod liniar corector de t erori (independente) este corector de pachete de j erori consecutive dacă toate cuvintele de lungime - pachet cel mult j sunt reprezentanţi.

Definiţia 10.3 *Un (n,k) - cod ciclic corectează pachetele de j erori consecutive dacă orice pachet standard de lungime cel mult j este selectat ca sindrom.*

Lema 10.1 *Dacă un cod C este corector de t erori independente şi corector de pachete de j erori consecutive, atunci $t \leq j$.*

Demonstraţie: Exerciţiu. q.e.d.

Exemplul 10.5 *Să considerăm toate pachetele standard nenule de lungime cel mult 3 în Z_2^{15}. Fiecare astfel de cuvânt este de forma*
$\{e(X)\} = \{X^i e_0(X)\}$ *cu* $0 \leq i \leq 14$ *şi* $e_0(X) \in \{1, 1+X, 1+X^2, 1+X+X^2\}$.
În total sunt $15 \cdot 4 = 60$ *astfel de pachete de erori.*

Exemplul 10.6 *Fie $g(X) = 1+X+X^2+X^3+X^6$ polinomul generator al unui $(15,9)$ - cod ciclic binar. El nu este un cod corector de 3 erori independente pentru că sunt 576 tipuri de erori de pondere cel mult 3 şi numai $2^{15}/2^9 = 64$ sindromuri. În schimb sunt numai 61 pachete standard de lungime maxim 3 (Exemplul 18.2, la care se adaugă pachetul nul); deci acest cod poate corecta orice pachet de maxim 3 erori consecutive. Faptul că acesta este un cod ciclic corector de pachete de 3 erori consecutive rezultă din calculul sindromurilor polinoamelor $\{X^i e_0(X)\}$ unde $0 \leq i \leq 14$ şi $e_0(X) \in \{1, 1+X, 1+X^2, 1+X+X^2\}$.*

10.2.1 Detectarea pachetelor de erori

Teorema 10.1 *Un (n,k) - cod ciclic detectează orice pachet de maxim $n-k$ erori.*

Demonstraţie: Dacă $\{e(X)\}$ este o eroare - tip de lungime cel mult $n-k$, atunci $e(X) = X^i e_0(X)$ cu $grad(e_0(X)) < n-k$. Fie $g(X)$ polinomul generator al codului, cu $grad(g(X)) = n-k$.

Ştim că $\{e(X)\}$ este cuvânt - cod $\iff$ $g(X)|e(X)$.

Cum $g(X)|X^n-1$, el va fi prim cu X^i; deci $\{e(X)\}$ este cuvânt - cod dacă şi numai dacă $g(X)|e_0(X)$, absurd deoarece $grad(e_0(X)) < n-k = grad(g(X))$. q.e.d.

Teorema 10.2 *Proporţia pachetelor standard de $j > n-k$ erori pe care nu le poate detecta un (n,k) - cod ciclic este*

$$\begin{cases} \dfrac{q^{-(n-k-1)}}{q-1} & \text{dacă} \quad j = n-k+1, \\ q^{-(n-k)} & \text{dacă} \quad j > n-k+1 \end{cases}$$

Demonstraţie: Fie $\{e(X)\} = \{X^s e_0(X)\}$ unde $grad(e_0(X)) = j-1$. Să numărăm câte asemenea pachete standard de erori sunt posibile. Ca prim şi ultim coeficient al lui $e_0(X)$ poate fi orice element din Z_q^* (în număr de $q-1$), iar coeficienţii intermediari pot fi orice elemente din Z_q (în număr de q). Deci sunt $(q-1)^2 q^{j-2}$ pachete standard de erori care încep şi se termină în aceeaşi poziţie.

Un pachet de erori $\{e(X)\}$ nu este detectat de un (n,k) - cod ciclic dacă şi numai dacă $\{e(X)\}$ este cuvânt - cod, adică $e_0(X) = g(X)h(X)$, unde $grad(h(X)) = grad(e_0(X)) - grad(g(X)) = j-1-(n-k)$. Deci polinomul $h(X)$ are $j-n+k$ coeficienţi.

Apar două situaţii:

1. $grad(h(X)) = 0$, adică $j = n-k+1$. Atunci $h(X)$ se reduce la o constantă şi există $q-1$ asemenea polinoame $h(X)$. Deci, raportul dintre numărul pachetelor standard de lungime j nedetectabile şi numărul total al pachetelor standard de lungime j care pot fi localizate este
$$\frac{q-1}{(q-1)^2 q^{j-2}} = \frac{q^{-(n-k-1)}}{q-1}.$$

2. Dacă $grad(h(X)) > 0$, adică $j > n-k+1$, vom avea $(q-1)^2 q^{j-n+k-2}$ polinoame $h(X)$ posibile (pachete standard de lungime j) nedetectabile. Rezultă că raportul de sus este în acest caz
$$\frac{(q-1)q^{j-n+k-2}}{(q-1)^2 q^{j-2}} = q^{-(n-k)}.$$

q.e.d.

Exemplul 10.7 *Codul ciclic definit în Exemplul 18.7 detectează orice pachet standard de maxim* 6 *erori.*

Dintre pachetele standard de 7 *erori posibile care pot apare, doar* $1/2^5$ *nu pot fi detectate, iar dintre pachetele stamdard de* $8-15$ *erori nu pot fi detectate* $1/2^6$.

10.2.2 Corectarea pachetelor de erori

Algoritmul de corectare a pachetelor de erori este o variantă a Algoritmului Meggitt.

Fie $C = (\{g(X)\})$ un (n,k) - cod ciclic binar corector de pachete de maxim j erori consecutive, şi $\mathbf{a} \in Z_q^n$ un cuvânt recepţionat.

1. Se calculează sindromul $\{s(X)\}$;
2. Pentru fiecare $i \geq 0$ se calculează $s_i(X) = X^i s(X) \ mod \ g(X)$, până se ajunge la un p cu $grad(s_p(X)) < j-1$.

   Atunci eroarea standard este $\{e(X)\} = \{X^{n-p} s_p(X)\}$ şi se generează cuvântul - cod $\mathbf{a}+\mathbf{e}$.

Exemplul 10.8 *Conform Exemplului 18.7, polinomul* $g(X) = 1+X+X^2+X^3+X^6$ *generează un* $(15,9)$ *- cod ciclic binar corector de pachete standard de cel mult* 3 *erori. Să folosim algoritmul Meggitt pentru a decodifica cuvântul* $\mathbf{a} = 111100100001010$.

$s(X) = 1 + X + X^2 + X^3 + X^6 + X^{11} + X^{13}\ (mod\ g(X)) = 1 + X^3 + X^4 + X^5.$
$s_1(X) = Xs(X)\ (mod\ g(X)) = 1 + X^2 + X^3 + X^4 + X^5,$
$s_2(X) = X^2s(X)\ (mod\ g(X)) = 1 + X^2 + X^4 + X^5,$
$s_3(X) = X^3s(X)\ (mod\ g(X)) = 1 + X^2 + X^5,$
$s_4(X) = X^4s(X)\ (mod\ g(X)) = 1 + X^2.$

S-a ajuns la $grad(s_4(X)) = 2 \leq 3 - 1$. *Deci eroarea - tip este*

$$\{e(X)\} = \{X^{15-4}s_4(X)\} = X^{11} + X^{13}.$$

Cuvântul - cod transmis a fost
$\mathbf{v} = \mathbf{a} + \mathbf{e} = 111100100001010 + 000000000001010 = 111100100000000.$

10.3 Transpunere

O metodă eficientă de utilizare a capacităt ilor de corectare a pachetelor standard de erori de către codurile ciclice binare este utilizarea de transpuneri (*interleaving*).

În mod natural, mesajele de informaţie $\mathbf{m_1}, \mathbf{m_2}, \ldots$ sunt codificate în cuvintele - cod $\mathbf{c_1}, \mathbf{c_2}, \ldots$ şi transmise prin canal în ordinea codificării. Transmisia se poate realiza însă şi după o rearanjare a caracterelor de informaţie din mai multe cuvinte - cod consecutive.

Formal, prin *transpunere la adâncimea* s se înţelege scrierea a s cuvinte - cod $\mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_s}$ ca linii ale unei matrici

$$X_{s,n} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & \ldots & c_{1n} \\ c_{21} & c_{22} & c_{23} & \ldots & c_{2n} \\ & & \vdots & & \\ c_{s1} & c_{s2} & c_{s3} & \ldots & c_{sn} \end{pmatrix}$$

şi transmiterea a n mesaje de lungime s, formate din coloanele matricii X_{sn}:

$$c_{11}c_{21}\ldots c_{s1}, \quad c_{12}c_{22}\ldots c_{s2}, \quad \ldots, \quad c_{1n}c_{2n}\ldots c_{sn}$$

Exemplul 10.9 *Să considerăm codul liniar generat de matricea* $G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$

şi şase mesaje codificate $\mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_6}$, *unde*

$$\mathbf{c_1} = 100110, \quad \mathbf{c_2} = 010101, \quad \mathbf{c_3} = 111000,$$
$$\mathbf{c_4} = 010101, \quad \mathbf{c_5} = 100110, \quad \mathbf{c_6} = 111000.$$

Deci – la nivel de caracter – secvenţa va arăta:

100110 010101 111000 010101 100110 111000.

Printr-o transpunere la adâncimea 3*, forma mesajelor este:*

101 011 001 110 100 010 011 101 001 110 010 100

iar cu o transpunere la adâncimea 6*:*

101011 011101 001001 110110 100010 010100.

Ce efect are o transpunere la adîncimea s asupra capacităţii de corecţie de pachete standard de erori ale codului ? Răspunsul este dat de

Teorema 10.3 *Fie C un cod binar corector de pachete standard de j erori. Dacă C este transpus la adâncimea s, atunci toate pachetele de cel mult sj erori pot fi corectate, în ipoteza că fiecare cuvânt - cod este afectat de cel mult un pachet standard de j erori.*

Demonstraţie: Dacă primul caracter al cuvântului - cod $\mathbf{c}$ este al i - lea caracter transmis, atunci celelalte caractere apar pe poziţiile $i+s, i+2s, \ldots, i+(n-1)s$. Orice pachet de maxim sj erori va produce un pachet standard de maxim j erori în $\mathbf{c}$, deci $\mathbf{c}$ va putea fi regăsit la recepţie (dacă el nu a mai fost cumva afectat şi de alte pachete de erori). q.e.d.

Restricţia ca fiecare cuvânt - cod să fie afectat de cel mult un pachet de erori impune condiţia ca pachetele de erori să fie separate de perioade în care transmisia este corectă, perioade suficient de lungi pentru evitarea situaţiei ca un bloc de s cuvinte să fie afectat de două pachete distincte de erori. În acest fel, crescând s, creşte şi lungimea pachetului de erori care poate fi corectat, dar creşte şi lungimea perioadelor necesare de transmisie fără erori.

Exemplul 10.10 *Codul din Exemplul 18.5 corectează o eroare. Folosind o transpunere la adâncimea* 3*, el poate corecta orice pachet standard de maxm* 3 *erori consecutive.*

Un dezavantaj al transpunerii este acela că nu se poate face transmisia până nu au fost codificate toate cele s cuvinte, fapt care nu va permite folosirea directă a circuitelor liniare.

Pentru a evita acest neajuns se va folosi un s - *cadru de transpunere întârziată*, care aranjează caracterele codificate, nu sub forma unei matrici X_{sn} ci sub formă de "scară" – ca în tabelul:

$$\begin{array}{cccccccccccc}
c_{1,1} & c_{2,1} & \ldots & c_{s+1,1} & c_{s+2,1} & \ldots & c_{2s+1,1} & c_{2s+2,1} & \ldots & c_{(n-1)s+1,1} & \ldots \\
 & & & c_{1,2} & c_{2,2} & \ldots & c_{s+1,2} & c_{s+2,2} & \ldots & c_{(n-2)s+1,2} & \ldots \\
 & & & & & & c_{1,3} & c_{2,3} & \ldots & c_{(n-3)s+1,3} & \ldots \\
 & & & & & & & & & \vdots & \\
 & & & & & & & & & c_{1,n} & \ldots
\end{array}$$

Deoarece transmisia se face pe coloane, spaţiile libere trebuiesc marcate. Acest lucru se realizează introducând pe aceste poziţii un caracter special $* \notin Z_q$.

Exemplul 10.11 *Să reluăm mesajul codificat $\mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_6}$ din Exemplul 18.5.*

Un 1 *- cadru de transpunere întârziată va fi*

$$\begin{array}{cccccccccccc}
1 & 0 & 1 & 0 & 1 & 1 & \ldots & & & & & \\
* & 0 & 1 & 1 & 1 & 0 & 1 & \ldots & & & & \\
* & * & 0 & 0 & 1 & 0 & 0 & 1 & \ldots & & & \\
* & * & * & 1 & 1 & 0 & 1 & 1 & 0 & \ldots & & \\
* & * & * & * & 1 & 0 & 0 & 0 & 1 & 0 & \ldots & \\
* & * & * & * & * & 0 & 1 & 0 & 1 & 0 & 0 & \ldots
\end{array}$$

Scvenţa de caractere transmise este

$$1 * * * * * 00 * * * * 110 * * * 0101 * * 11111 * 100000 \ldots$$

Dacă se foloseşte un 2 *- cadru de transpunere întârziată, avem*

$$
\begin{array}{cccccccccccccccccc}
1 & 0 & 1 & 0 & 1 & 1 & \ldots & & & & & & & & & & & \\
* & * & 0 & 1 & 1 & 1 & 0 & & 1 & \ldots & & & & & & & & \\
* & * & * & * & 0 & 0 & 1 & & 0 & 0 & & 1 & \ldots & & & & & \\
* & * & * & * & * & * & 1 & & 1 & 0 & & 1 & 1 & & 0 & \ldots & & \\
* & * & * & * & * & * & * & & * & 1 & & 0 & 0 & & 0 & 1 & & 0 & \ldots \\
* & * & * & * & * & * & * & & * & * & & * & 0 & & 1 & 0 & & 1 & 0 & & 0 & \ldots
\end{array}
$$

iar secvenţa de caractere transmise este

$$1 * * * * * 0 * * * * * 10 * * * * 01 * * * * 110 * * * 110 * * * \ldots$$

Este uşor de găsit o teoremă analogă cu Teorema 20.2:

Teorema 10.4 *Fie C un cod capabil să corecteze toate pachetele standard de cel mult j erori consecutive. Dacă C foloseşte un s - cadru de transpunere întârziată, atunci se poate corecta orice pachet standard de maxim $j(sn+1)$ erori, în ipoteza că orice cuvânt - cod este afectat de cel mult un pachet standard de j erori.*

Demonstraţie: Exerciţiu.

Ca o altă facilitate, pentru codificarea unui mesaj se folosesc adesea două coduri. De exemplu, pentru codificarea muzicii pe compact - discuri se utilizează două coduri Reed - Solomon, iar NASA şi Agenţia Spaţială Europeană folosesc două coduri convoluţionale.

Fie A_i două (n_i, k_i) $(i = 1, 2)$ coduri liniare. *Transpunerea încrucişată* a lui A_1 cu A_2 se realizează astfel:

1. Mesajele de informaţie sunt codificate cu A_1 iar cuvintele - cod rezultate sunt transpuse la adâncimea k_2.
2. Coloanele obţinute în acest proces de transpunere (care sunt de lungime k_2) – privite ca mesaje de informaţie – sunt codificate de codul A_2.
3. Cuvintele - cod rezultate sunt transpuse la o adâncime s – fixată sau folosind un s - cadru de transpunere întârziată.

Avantajul principal al celui de al doilea proces de codificare este următorul:

Fie d_1, d_2 distanţele mimime ale celor două coduri. A_2 poate detecta $d_2 - 1$ erori (nu ne punem problema corectării lor). Dacă s-au detectat erori pentru un cuvânt - cod din C_2, atunci toate caracterele acestui cuvânt sunt marcate şi tratate drept simboluri care pot fi incorecte. Se consideră apoi cuvintele - cod din A_1. Dacă $n_1 - d_1 + 1$ caractere dintr-un cuvânt - cod din A_1 sunt corecte, atunci se pot determina unic celelalte $d_1 - 1$ caractere (deoarece este imposibil ca două cuvinte - cod distincte din A_1 să coincidă pe $n_1 - d_1 + 1$ poziţii, ele diferind pe minim d_1 poziţii; afirmaţia este bazată şi pe faptul că poziţiile caracterelor corecte sunt cunoscute).

Deci, dacă un cuvânt - cod din A_1 are cel mult $d_1 - 1$ simboluri marcate şi se ştie că toate caracterele greşite sunt marcate, cuvintele - cod pot fi decodificate corect.

Cât de mare este un pachet de erori pe care îl poate corecta această schemă ? Să presupunem că A_2 a fost transpus la adâncimea s, şi că orice cuvânt - cod (atât din A_1 cât şi din A_2) a fost afectat de cel mult un pachet de erori. Dacă apare un pachet de cel

mult $s(d_2-1)$ erori consecutive, atunci el va afecta maxim d_2-1 simboluri din fiecare cuvânt - cod din A_2. Deci aceste erori vor fi detectate şi conduc la marcarea tuturor caracterele cuvintelor - cod implicate. Dacă $s \leq d_1-1$, atunci orice cuvânt - cod din A_1 are cel mult d_1-1 simboluri marcate.

În ipoteza că nu există decât maxim un pachet de erori care l-a perturbat, caracterele marcate pot fi corectate.

Am arătat astfel:

Teorema 10.5 *Fie o codificare care foloseşte transpunerea încrucişată a (n_1,k_1) - codului A_1 cu (n_2,k_2) - codul A_2, A_2 fiind transpus la adâncimea s ($s \leq d_1-1$). Dacă fiecare cuvânt - cod este afectat de cel mult un pachet standard de erori, atunci toate pachetele standard de erori de lungime maxim $s(d_2-1)$ pot fi corectate. (d_1, d_2 sunt distanţele minime ale celor două coduri).*

Exemplul 10.12 *Fie A_1 şi A_2 codurile binare, definite respectiv de matricile generatoare*

$$G_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \qquad G_2 = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

Deci, $n_1 = 8, k_1 = 4, d_1 = 4, \quad n_2 = 6, k_2 = 3, d_2 = 3$.

Să codificăm mesajele de informaţie $\mathbf{m_1} = 1000$, $\mathbf{m_2} = 1100$, $\mathbf{m_3} = 1010$ folosind o transpunere încrucişată a lui A_1 cu A_2, A_2 fiind transpus la o adâncime $s = 3 = d_1 - 1$.

Codificarea mesajelor de informţie cu codul A_1 dă:

$$\mathbf{c_1} = \mathbf{m_1}G_1 = 10001110, \quad \mathbf{c_2} = \mathbf{m_2}G_1 = 11000011, \quad \mathbf{c_3} = \mathbf{m_3}G_1 = 10100101.$$

Transpunerea acestor cuvinte la adâncimea $k_2 = 3$ conduce la mesajul

$$111\ 010\ 001\ 000\ 100\ 101\ 110\ 011$$

Aceste 8 mesaje de informaţie sunt codificate cu A_2 în:

$$\mathbf{c'_1} = 111000,\ \mathbf{c'_2} = 010101,\ \mathbf{c'_3} = 001011,\ \mathbf{c'_4} = 000000,$$
$$\mathbf{c'_5} = 100110,\ \mathbf{c'_6} = 101101,\ \mathbf{c'_7} = 110011,\ \mathbf{c'_8} = 011110.$$

şi apoi transpuse la adâncimea $s = 3$ ($\mathbf{c'_6}$ şi $\mathbf{c'_7}$ vor fi transpuse cu următorul cuvânt - cod $\mathbf{c'_9}$ obţinut de următoarele trei mesaje de informaţie $\mathbf{m_4}, \mathbf{m_5}, \mathbf{m_6}$).

Secvenţa de caractere transmisă va începe cu

$$100\ 110\ 101\ 010\ 001\ 011\ 011\ 000\ 001\ 011\ 010\ 001\ldots$$

Conform Teoremei 20.5, folosind A_2 pentru a detecta $d_2 - 1 = 2$ erori, şi apoi A_1 pentru a corecta toate caracterele marcate, se pot corecta în ansamblu toate pachetele de cel mult $s(d_2-1) = (d_1-1)(d_2-1) = 6$ erori consecutive.

De exemplu, să presupunem că primii 6 biţi au fost transmişi incorect, deci s-a primit mesajul

$$011\ 001\ 101\ 010\ 001\ 011\ldots.$$

Eliminând efectul transpunerii la adâncimea $s = 3$, se ajunge la cuvintele

$$001000 \quad 100101 \quad 111011$$

(care comparate respectiv cu $\mathbf{c'_1}, \mathbf{c'_2}, \mathbf{c'_3}$ au erori fiecare pe primele două poziţii). Deoarece sindromurile sunt nenule, A_2 va detecta erori în toate cele trei cuvinte, deci toţi cei 18 biţi sunt marcaţi (îi vom înlocui cu $*$).

Presupunând că nu mai sunt alte erori, după un procedeu similar se obţin cuvintele $\mathbf{c'_4}, \ldots, \mathbf{c'_8}$, fără caractere marcate.

Eliminând acum efectul transpunerii la adâncimea $k_2 = 3$, se obţin cuvintele

$$\mathbf{c_1} = ***01110, \quad \mathbf{c_2} = ***00011, \quad \mathbf{c_3} = **100101.$$

Există o modalitate unică de a ajunge la cuvinte - cod din A_1 prin înlocuirea $*$ cu caractere 0 sau 1.

10.4 Exerciţii

10.1 *Demonstraţi Lema 15.1*

10.2 *Arătaţi că dacă un cod ciclic detectează o eroare standard $\mathbf{e}$, atunci detectează toate erorile standard obţinute prin permutări ciclice ale lui $\mathbf{e}$.*

10.3 *Verificaţi că pachetele ciclice standard de erori de lungime 3 din Z_2^{15} au sindromuri diferite pentru codurile din Exemplul 18.7.*

10.4 *Arătaţi că $g(X) = 1+X^2+X^4+X^5$ generează un $(15, 10)$ - cod ciclic binar corector de pachete de maxim 2 erori. Este acesta un cod corector de 2 erori independente ?*

10.5 *Arătaţi că $g(X) = 1 + X^3 + X^4 + X^5 + X^6$ generează un $(15, 9)$ - cod ciclic binar corector de pachete de maxim 3 erori. Este acesta un cod corector de 3 erori independente ?*

10.6 *Arătaţi că $g(X) = 1 + X^4 + X^6 + X^7 + X^8$ generează un $(15, 7)$ - cod ciclic binar corector de 2 erori independente şi de pachete de cel mult 4 erori consecutive.*

10.7 *Fie codul din Exemplul 18.7. Să se decodifice mesajele:*
101101110001000, 001101100010101, 100110101010011,
101101000010111, 000000111110000.

10.8 *Fie $(15, 10)$ - codul ciclic binar generat de $g(X) = 1 + X^2 + X^4 + X^5$.*
Ce lungimi au pachetele standard de erori pe care le poate corecta ?
Decodificaţi mesajele:
010101000010010, 011010010010100, 001101000000100, 000100010100101,
000000011111001.

10.9 *Fie codul ciclic binar generat de polinomul* $g(X) = 1+X+X^2+X^3+X^6$. *Codificaţi mesajele de informaţie* $m_1(X) = 1,\ m_2(X) = X^2,\ m_3(X) = 1 + X,\ m_4(X) = 1 + X^2,\ m_5(X) = X^3,\ m_6(X) = 1.$

Determinaţi şirul de biţi transmişi dacă se foloseşte o transpunere la adâncimea s *unde*

$$(a)\ s = 1; \qquad (b)\ s = 2; \qquad (c)\ s = 3.$$

10.10 *Ce secvenţă de caractere este transmisă dacă se foloseşte un* 0 *- cadru de transpunere întârziată ?*

10.11 *Demonstraţi Teorema 20.3*

10.12 *Folosind codurile definite în Exemplul 18.6, să se codifice următoarele mesaje de informaţie prin transpunerea încrucişată a lui* A_1 *cu* A_2*:*

$(a)\ \mathbf{m_1} = 0110,\ \mathbf{m_2} = 1011,\ \mathbf{m_3} = 1111, \quad s = 2$

$(b)\ \mathbf{m_1} = 0110,\ \mathbf{m_2} = 1011,\ \mathbf{m_3} = 1111, \quad s = 3$

$(c)\ \mathbf{m_1} = 0010,\ \mathbf{m_2} = 1111,\ \mathbf{m_3} = 1010, \quad s = 3$

$(d)\ \mathbf{m_1} = 1000,\ \mathbf{m_2} = 0100,\ \mathbf{m_3} = 0010,\ \mathbf{m_4} = 0001,\ \mathbf{m_5} = 0011,\ \mathbf{m_6} = 0100 \quad s = 3$

10.13 *Folosind codurile din Exemplul 18.6 au fost obţinute prin transpunere încrucişată a lui* A_1 *cu* A_2 *la adâncimea* $s = 3$*, următoarele secvenţe binare:*

(a) 000001001110110001000111000111000111000000000000000000 . . .

(b) 100011001111101010011001111010100110100100011101000100 . . .

Aflaţi meajele de informaţie $\mathbf{m_1}, \mathbf{m_2}, \mathbf{m_3}$.

10.14 *Găsiţi un rezultat analog Teoremei 20.5 dacă pentru* A_2 *se foloseşte* s *- cadrul de transpunere întârziată în loc de* s *- transpunere.*

Capitolul 11

Alte definiţii ale codurilor ciclice

Pentru început vom trece în revistă câteva noţiuni algebrice – fundamentale în definirea unei alte modalităţi de construcţie a codurilor ciclice.

11.1 Elemente primitive în extensii Galois

Definiţia 11.1 *Fie $(F, +, \cdot)$ un corp finit, în care s-au notat cu 0 şi 1 elementele unitate faţă de cele două operaţii. Un element $a \in F$ are ordin n $(n \geq 1)$ dacă $a^n = 1$ şi $a^k \neq 1,\ \forall k,\ 0 < k < n$. Vom scrie $ord(a) = n$.*

Propoziţia 11.1 *Într-un corp finit $(F, +, \cdot)$:*
- *Orice element nenul are un ordin finit;*
- *Dacă $n = ord(a)$ atunci $a, a^2, \ldots, a^n$ sunt distincte;*
- *$a^k = 1$ dacă şi numai dacă $ord(a)|k$ $(ord(a)$ este divizor al lui $k)$.*

Demonstraţie: Fie $a \in F,\ a \neq 0$. Cum F este finit, elementele $a, a^2, a^3, \ldots$ nu pot fi toate distincte. Vom alege cel mai mic n pentru care $\exists\ i\ a^{n+i} = a^i$. Relaţia se poate simplifica (lucrăm într-un corp) cu a^i şi se obţine $a^n = 1$. Din construcţie, $a, a^2, \ldots, a^n$ sunt distincte, deci n este ordinul lui a.

Pentru $k = ni$ avem $a^k = (a^n)^i = 1^i = 1$. Reciproc, să presupunem $a^k = 1$. Teorema împărţirii cu rest dă $k = qn + r,\ \ 0 \leq r < n$. Avem $1 = a^k = a^{qn}a^r = a^r$.

Cum $r < n$ rezultă $r = 0$. q.e.d.

Definiţia 11.2 *Un element $a \in F$ este primitiv dacă $F^* = F \setminus \{0\} = \{1, a, a^2, a^3, \ldots, \}$.*

Observaţia 11.1 *Dacă F are r elemente, atunci un element este primitiv dacă şi numai dacă are ordinul $r - 1$.*

Teorema 11.1 *Orice corp finit are cel puţin un element primitiv.*

Demonstraţie: Dacă F este finit $(card(F) = r)$, se poate găsi un element $a \in F$ de ordin n maxim. Evident $n \leq r - 1$. Mai trebuie arătat că $n \geq r - 1$.

Fie $b \in F^*$ de ordin s. Să-l descompunem pe s în factori primi: $s = p^i q^j \ldots (p, q, \ldots$ numere prime distincte). La rândul lui $n = p^t n'$ unde n' nu este divizibil cu p (iar t poate fi eventual zero).

Să considerăm $s' = s/p^i$ (deci $s = p^i s'$) şi fie $c = a^{p^t} b^{s'} \in F$. Vrem să arătăm că c are ordinul $m = p^i n'$. Pentru aceasta, avem:

$$c^m = a^{p^t m} b^{s'm} = a^{p^t n' p^i} b^{p^i s' n'} = (a^n)^{p^i} (b^s)^{n'} = 1^{p^i} 1^{n'} = 1.$$

Mai rămâne de arătat că $c^{m'} = 1 \Longrightarrow m' \geq m$. Este suficient să verificăm că p^i şi n' sunt divizori ai lui m'; deoarece cele două numere sunt prime intre ele, va rezulta că produsul $m = p^i n'$ este de asemenea divizor al lui m'.

Din $1 = (c^{m'})^{n'} = a^{p^t n' m'} b^{s' m' n'} = (a^n)^{m'} b^{s' m' n'} = b^{s' m' n'}$ rezultă (Propoziţia 11.1) că ordinul $s = p^i s'$ al lui b divide $s' m' n'$; cum $(p', n') = 1$, se deduce $p^i | m'$.

În plus, din $1 = (c^{m'})^{p^i} = a^{p^t p^i m'} b^{p^i s' m'} = a^{p^t p^i m'} (b^s)^{m'} = a^{p^t p^i m'}$ rezultă că ordinul $n = p^t n'$ al lui a divide $p^t p^i m'$; deci $n' | p^i m'$, adică $n' | m'$.

Cum n este cel mai mare ordin ale elementelor din F, avem $ord(e) \leq ord(a)$, sau $m \leq n$, adică $p^i n' \leq p^t n'$; deci $i \leq t$, adică p^i divide $n = p^t n'$.

În mod similar, toţi factorii lui s sunt divizori ai lui n, deci $s | n$.

Am arătat următoarea afirmaţie:

$$\forall\, b \in F^* \quad \Longrightarrow \quad ord(b) | n$$

Deci b este o rădăcină a polinomului $X^n - 1 = 0$, de unde rezultă că polinomul $X^n - 1 = 0$ are $r - 1$ rădăcini, adică $r - 1 \leq n$. q.e.d.

Fie $(F, +, \cdot)$ un corp finit şi $f(X) \in F[X]$ un polinom arbitrar cu coeficienţi în F. Spunem că $a \in F$ este *rădăcină* a lui f dacă $f(a) = 0$.

Se poate demonstra imediat afirmaţia:

Propoziţia 11.2 *Dacă un polinom f are rădăcinile distincte $a_1, a_2, \ldots, a_n$ atunci el este divizibil cu polinomul $(X - a_1)(X - a_2) \ldots (X - a_n)$.*

Exemplul 11.1 *Polinomul $X^3 + 1$ are o rădăcină 1 în Z_2. Deci $X^3 + 1$ se divide cu $X + 1$:*

$$X^3 + 1 = (X + 1)(X^2 + X + 1)$$

Aceasta este o factorizare completă; deoarece $X^2 + X + 1$ nu are rădăcini în Z_2, el nu poate fi descompus în produsul a două polinoame de gradul 1.

Acelaşi polinom $X^3 + 1$ are în Z_3 pe 2 ca rădăcină triplă, deci aici putem scrie:

$$X^3 + 1 = (X + 1)^3.$$

Rezultă că factorizarea unui polinom depinde de corpul în care este definit.

Din Teorema 11.1 şi Propoziţia 20.1 rezultă că dacă $a \in F$ este un element de ordin n, atunci $X^n - 1$ se poate descompune în

$$X^n - 1 = (X - a)(X - a^2) \ldots (X - a^n).$$

Definiţia 11.3 *Un polinom $f(X) \in F[X]$ de grad n este ireductibil în corpul F dacă nu se poate descompune în produsul a două polinoame din $F[X]$ de grad mai mic decât n. În caz contrar, f este un polinom reductibil în F.*

Observaţia 11.2

- *Orice polinom liniar este ireductibil. Pentru polinoame de grad cel puţin* 2*, Propoziţia 20.1 afirmă că:*

  Un polinom ireductibil într-un corp nu are rădăcini în acel corp.

  Este interesant că reciproca nu este adevărată decât pentru polinoamele de grad 2 *şi* 3*. Astfel, polinomul* $f(X) = (X^2 + X + 1)^2 \in Z_2[X]$*, deşi nu are rădăcini în* Z_2*, este reductibil.*

- *În* $\mathcal{R}$ *singurele polinoame ireductibile sunt de gradul* 1 *sau* 2*. Dacă un polinom are grad impar, el are sigur o rădăcină reală, iar dacă este de grad par, atunci are sau cel puţin o rădăcină reală, sau cel puţin două rădăcini complexe de forma* $a \pm ib$*. În acest ultim caz, el se va divide cu* $(x - a)^2 + b^2$*, care este un polinom de gradul* 2 *din* $\mathcal{R}[X]$*.*

- *În corpul complex* $\mathcal{C}$*, teorema fundamentală a algebrei asigură că orice polinom ireductibil are gradul* 1*.*

Definiţia 11.4 *Caracteristica unui corp finit* F *este cel mai mic număr de termeni ai sumei* $S = 1 + 1 + \ldots + 1$ *cu proprietatea* $S = 0$*.*

Propoziţia 11.3 *Caracteristica unui corp finit este număr prim.*

Demonstraţie: Să considerăm elementele $S_i = 1 + 1 + \ldots + 1$ de câte i termeni fiecare. Deoarece nu pot fi o infinitate de valori distincte, fie p minim cu proprietatea
$\exists\, i,\ S_i = S_{i+p}$. Rezultă $S_p = S_{i+p} - S_i = 0$. Deci F are caracteristica p.

Presupunem $p = jt$ cu $1 \leq j < p$. Evident $0 = S_{jt} = S_j + S_j + \ldots + S_j$ (t termeni). Deoarece $j < p$ avem $S_j \neq 0$. Împărţind relaţia cu S_j se obţine $0 = 1 + 1 + \ldots + 1 = S_t$. Deci $t = p$. q.e.d.

Corolarul 11.1 *Orice corp de caracteristică* q *este o extensie a lui* Z_q*.*

Demonstraţie: Exerciţiu.

Corolarul 11.2 $GF(q^r)$ *este un corp de caracteristică* q*.*

Demonstraţie: Reamintim că $GF(q^r)$ conţine toate polinoamele din $Z_q[X]$ de grad cel mult $r - 1$. În particular, polinoamele de grad 0 (constantele) formează un subcorp izomorf cu Z_q care are caracteristica q. q.e.d.

Propoziţia 11.4 *Într-un corp de caracteristică* q *avem* $(a + b)^q = a^q + b^q$*.*

Demonstraţie: Se foloseşte binomul lui Newton, în care $C_q^k = 0,\ \forall\, k,\ 0 < k < q$. q.e.d.

11.2 Polinoame minimale

În cele ce urmează vom restrânge studiul la cazul $F = Z_q$, q număr prim.

Definiţia 11.5 *Fie* β *un element dintr-o extensie a lui* Z_q*. Se numeşte "polinom minimal" al lui* β *polinomul normat* $g(X) \in Z_q[X]$ *de grad minim, cu* $g(\beta) = 0$*.*

Existenţa acestui polinom este asigurată de teorema 4.1 (Prelegerea 4), pe baza căreia s-au definit extensiile Galois.

Exemplul 11.2 *Să considerăm extensia $GF(2^3)$ generată de rădăcina $'al$ a plinomului $1 + X + X^3$. Deoarece α este primitiv, avem $GF(2^3) = \{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$. Polinoamele minimale ale fiecărui element sunt date în tabelul:*

Element	*Polinom minimal*
0	X
1	$1 + X$
$\alpha, \alpha^2, \alpha^4$	$1 + X + X^3$
$\alpha^3, \alpha^5, \alpha^6$	$1 + X^2 + X^3$

Propoziţia 11.5 *Un polinom minimal $g(X) \in Z_q[X]$ este ireductibil peste Z_q.*

Demonstraţie: Dacă ar exista descompunerea $g(X) = u(X)v(X)$ cu $u(X), v(X) \in Z_q[X]$, atunci vom avea $u(\beta) = 0$ sau $v(\beta) = 0$, ceea ce contrazice definiţia polinomului minimal pentru β. q.e.d.

Corolarul 11.3 *Dacă $f(X) \in Z_q[X]$ verifică $f(\beta) = 0$ atunci $g(X)|f(X)$.*

De aici rezultă că un polinom normat peste Z_q care admite pe β ca rădăcină este polinom minimal al lui β. Acest polinom este unic.

Teorema 11.2 *Fie $\beta \in GF(q^k)$. Atunci există un polinom minimal al lui β de grad cel mult k.*

Demonstraţie: După cum rezultă din definirea extensiei Galois şi din Teorema 5.3, $GF(q^k)$ are dimensiunea k. O bază a lui este $\{1\}, \{X\}, \ldots, \{X^{k-1}\}$. Deci cele $k + 1$ elemente $1, \beta, \beta^2, \ldots, \beta^k \in GF(q^k)$ sunt liniar dependente. Relaţia lor de dependenţă conduce la construirea unui polinom de grad maxim k pentru care β este rădăcină. q.e.d.

Teorema 11.3 *Elementele nenule din Z_q sunt date de soluţiile ecuaţiei $X^{q-1} - 1 = 0$.*

Demonstraţie: Elementele din $Z_q \setminus \{0\}$ formează grup multiplicativ. Ordinul fiecărui element divide ordinul grupului, care este $q - 1$ (Teorema 11.1).

Fie $\beta \in Z_q \setminus \{0\}$; subgrupul ciclic generat de β este $1, \beta, \beta^2, \ldots, \beta^{t-1}$ unde $b^t = 1$ şi $t|q - 1$. Deci $\beta^{q-1} = 1$.

În plus, ecuaţia $X^{q-1} - 1 = 0$ are $q - 1$ rădăcini. q.e.d.

Teorema 11.4 $X^m - 1|X^n - 1 \iff m|n$.

Demonstraţie: "$\Longleftarrow$": Fie $n = md$. Cum $Y - 1$ divide pe $Y^d - 1$, dacă se ia $Y = X^m$, vom avea $X^m - 1|X^{md} - 1$.

"$\Longrightarrow$": Să presupunem că $X^m - 1|X^n - 1$ şi fie $n = md + s$, $(s < m)$. Avem $X^n - 1 = X^s(X^{md} - 1) + X^s - 1 = q(X)(X^m - 1) + r(X)$ cu $grad(r(X)) = s < m$.

Pentru a verifica ipoteza, trebuie ca $s = 0$, deci $n = md$. q.e.d.

Teorema 11.5 *Fie $f(X) \in Z_q[X]$ şi β o rădăcină a sa (eventual dintr-o extensie a lui Z_q). Atunci şi β^q este rădăcină a lui $f(X)$.*

Demonstraţie: Scriem $f(X) = a_0 + a_1X + \ldots + a_nX^n$. Cum Z_q este corp de caracteristică q, este adevărată relaţia $(a+b)^q = a^q + b^q$ (Propoziţia 11.4).

De asemenea (Teorema 20.2) $\forall a \in Z_q \setminus \{0\}$ avem $a^{q-1} - 1 = 0$, deci $a^q = a$.

Atunci se poate scrie
$(f(X))^q = a_0^q + a_1^q X^q + \ldots + a_n^q (X^n)^q = a_0 + a_1X^q + \ldots + a_nX^{nq} = f(X^q)$.

În particular, $f(\beta^q) = (f(\beta))^q = 0$. q.e.d.

Teorema 11.6 *Orice polinom normat ireductibil peste Z_q de grad m este un factor al polinomului $X^{q^m} - X$.*

Demonstraţie: Dacă $g(X) = X$, afirmaţia este banală.

Să presupunem că $g(X) \neq X$ şi fie $\beta \in GF(q^m)$ o rădăcină nenulă a sa, deci $g(\beta) = 0$. Cum $g(X)$ este ireductibil, el va fi chiar polinomul minimal al lui β. Conform Teoremei 20.2, β satisface ecuaţia $X^{q^m-1} - 1 = 0$, deci (Corolar 11.3) $X^{q^m-1} - 1$ se divide cu polinomul minimal $g(X)$. q.e.d.

Teorema 11.7 *Orice factor ireductibil $g(X) \in Z_q[X]$ al lui $X^{q^m} - X$ are gradul cel mult m.*

Demonstraţie: Fie $g(X) | X^{q^m} - X$, ireductibil peste Z_q, cu $grad(g(X)) = k$.

Vom considera algebra polinoamelor modulo $g(X)$, în care luăm $\alpha = \{X\}$. Se ştie (Teorema 4.1) că $g(\alpha) = 0$. Un element oarecare din această algebră este de forma

$$\beta = a_0 + a_1\alpha + \ldots + a_{k-1}\alpha^{k-1}.$$

Atunci $\beta^{q^m} = a_0^{q^m} + a_1^{q^m}\alpha^{q^m} + \ldots + a_{k-1}^{q^m}(\alpha^{k-1})^{q^m} = a_0 + a_1\alpha^{q^m} + \ldots + a_{k-1}(\alpha^{q^m})^{k-1} = a_0 + a_1\alpha + \ldots + a_{k-1}\alpha^{k-1} = \beta$
deoarece $GF(q^m)$ are tot caracteristica q ca şi Z_q, iar α – fiind o rădăcină a lui $g(X)$, este rădăcină şi a lui $X^{q^m} - X$, deci $\alpha^{q^m} = \alpha$, de unde rezultă $\alpha^{jq^m} = \alpha^j \; \forall j \geq 0$.

Am obţinut în final faptul că β este o soluţie a ecuaţiei $X^{q^m} - X = 0$. Sunt posibile q^k asemenea elemente β distincte, iar cum ecuaţia are q^m rădăcini, rezultă $q^k \leq q^m$, deci $k \leq m$. q.e.d.

Teorema 11.8 *Fie $\beta \in GF(q^m)$ de polinom minimal $g(X)$, $grad(g(X)) = k$ şi $ord(\beta) = t$. Atunci*

(i) $t | q^k - 1$;
(ii) *k este minim cu proprietatea (i).*

Demonstraţie: Ştim (Teorema 11.6) că $g(X)$ este un factor al lui $X^{q^k} - X$; deci $\beta^{q^k} = \beta$, de unde rezultă $t | q^k - 1$.

Să presupunem acum că există $p < k$ cu $t | q^p - 1$. Atunci $\beta^{q^p-1} = 1$, adică β este o rădăcină a ecuaţiei $X^{q^p} - X = 0$, deci $g(X)$ este un factor al polinomului $X^{q^p} - X$. Conform Teoremei 11.7, rezultă $k \leq p$, contradicţie. q.e.d.

Teorema 11.9 *Fie polinomul $g(X) \in Z_q[X]$, $grad(g(X)) = m$ şi $\beta \in GF(q^m)$ o rădăcină a sa. Atunci $\beta, \beta^q, \beta^{q^2}, \ldots, \beta^{q^{m-1}}$ sunt toate rădăcinile lui $g(X)$.*

Demonstraţie: Deoarece $g(\beta) = 0$, avem – conform Teoremei 20.5 – că $\beta^q, \beta^{q^2}, \ldots, \beta^{q^{m-1}}$ sunt şi ele rădăcini ale lui $g(X)$. De asemenea, conform Teoremei 11.6, β este rădăcină a lui $X^{q^m} - X$, adică $\beta^{q^m} = \beta$.

Mai rămâne de arătat că aceste rădăcini sunt distincte. Presupunem că există $0 \leq i < j < m$ cu $\beta^{q^i} = \beta^{q^j}$. Avem

$$\beta = \beta^{q^m} = \left(\beta^{q^j}\right)^{q^{m-j}} = \left(\beta^{q^i}\right)^{q^{m-j}} = \beta^{q^{m+i-j}}$$

Deci β este rădăcină a polinomului $X^{q^{m+i-j}} - X$ şi – cu Teorema 11.7, $grad(g(X)) = m \leq m + i - j < m$, contradicţie. q.e.d.

Din Teorema 11.9 rezultă că luând extensia $GF(q^m)[X]$, $g(X)$ se poate scrie

$$g(X) = (X - \beta)(X - \beta^q) \ldots (X - \beta^{q^{m-1}}).$$

Teorema 11.10 *Fie $g(X)$ un polinom normat ireductibil peste Z_q, $grad(g(X)) = m$, şi $\beta \in GF(q^m)$ o rădăcină a sa. Atunci toate rădăcinile lui $g(X)$ au acelaşi ordin.*

Demonstraţie: Fie $p = ord(\beta)$, $p' = ord(\beta^{q^j})$, $(1 < j < m)$ care – conform Teoremei 11.9 – este tot rădăcină a lui $g(X)$. Avem:

$$\left(\beta^{q^j}\right)^p = (\beta^p)^{q^j} = 1 \text{ şi deci } p'|p.$$

De asemenea,

$$\beta^{p'} = \left(\beta^{q^m}\right)^{p'} = \left(\left(\beta^{q^j}\right)^{q^{m-j}}\right)^{p'} = \left(\left(\beta^{q^j}\right)^{p'}\right)^{q^{m-j}} = 1 \quad \Longrightarrow \quad p|p'.$$

S-a mai folosit Teorema 11.8, conform căreia p divide $q^m - 1$ dar nu divide nici un număr de forma $q^s - 1$ cu $s < m$. Deci $p = p'$. q.e.d.

11.3 Definirea codurilor ciclice prin rădăcini

O altă metodă de definire a unui cod ciclic constă în listarea tuturor rădăcinilor (eventual într-o extensie a lui Z_q) cuvintelor - cod (considerate ca polinoame).

Definiţia 11.6 *Fie $\alpha_1, \alpha_2, \ldots, \alpha_r$ elemente dintr-o extensie a lui Z_q. Codul ciclic C este format din toate elementele $\{f(X)\}$ din algebra polinoamelor modulo $X^n - 1$, care admit pe α_i, $(1 \leq i \leq r)$ ca rădăcini simple.*

De remarcat că în această definiţie, n este deocamdată nedeterminat.

Conform Corolarului 11.3, $f(X)$ se va divide cu fiecare polinom minimal $m_i(X)$ corespunzător rădăcinii α_i $(1 \leq i \leq r)$. Deci polinomul generator al codului ciclic este

$$g(X) = cmmmc\{m_1(X), m_2(X), \ldots, m_r(X)\}.$$

Propoziţia 11.6 $C = (\{g(X)\})$.

Demonstraţie: Este lăsată ca exerciţiu.

Deoarece polinomul generator $g(X)$ divide pe $X^n - 1$, rezultă că fiecare α_i este rădăcină a lui $X^n - 1$. Deci ordinul $ord(\alpha_i)$ al fiecărei rădăcini α_i $(1 \leq i \leq r)$ va divide pe n.

O modalitate naturală de definire a lui n este atunci

$$n = cmmmc\{ord(\alpha_1), ord(\alpha_2), \ldots, ord(\alpha_r)\}.$$

Un caz particular important este acela în care toate rădăcinile $\alpha_1, \alpha_2, \ldots, \alpha_r$ sunt puteri ale unui anumit element, adică

$$\alpha_i = \alpha^{u_i} \quad (1 \le i \le r).$$

Fie $p = ord(\alpha)$. Atunci polinomul minimal $m_i(X)$ al lui α_i va avea (Teorema 11.9) toate rădăcinile sale printre elementele $\alpha^{u_i}, \alpha^{qu_i}, \alpha^{q^2u_i}, \ldots$.

Numărul factorilor lui $g(X)$ şi gradul fiecărui polinom minimal $m_i(X)$ se vor determina atunci din ordinul p şi din exponenţii $u_i, qu_i, q^2u_i, \ldots$ $(1 \le i \le r)$. Într-adevăr, numărul claselor distincte de resturi modulo p din succesiunea $u_i, qu_i, \ldots$ va da gradul polinomului minimal $m_i(X)$.

Exemplul 11.3 *Fie $\alpha \in GF(2^4)$ primitiv, rădăcină a polinomului $1+X+X^4$, ireductibil peste Z_2. Să definim un cod C care să aibă ca rădăcini $\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6$.*

Fiind primitiv, $ord(\alpha) = 2^4 - 1 = 15$ (vezi şi Prelegerea 8, Exemplul 8.8).

Fie $m_i(X)$ polinomul minimal al rădăcinii α_i, $(1 \le i \le 6)$.

Rădăcinile lui $m_1(X)$ sunt $\alpha, \alpha^2, \alpha^4, \alpha^8$ $(\alpha^{16} = \alpha)$, deci

$$m_1(X) = m_2(X) = m_4(X) = (X-\alpha)(X-\alpha^2)(X-\alpha^4)(X-\alpha^8).$$

Acest polinom este cunoscut, anume $1 + X + X^4$.

Rădăcinile lui $m_3(X)$ sunt $\alpha^3, \alpha^6, \alpha^{12}, \alpha^{24} = \alpha^9$ $(\alpha^{18} = \alpha^3)$, deci $m_3(X)$ este un polinom de gradul 4, egal cu $m_6(X)$.

Notăm $m_3(X) = a_0 + a_1X + a_2X^2 + a_3X^3 + X^4$. Detaliind $m_3(\alpha^3) = 0$, avem

$$a_0\begin{pmatrix}1\\0\\0\\0\end{pmatrix} + a_1\begin{pmatrix}0\\0\\0\\1\end{pmatrix} + a_2\begin{pmatrix}0\\0\\1\\1\end{pmatrix} + a_3\begin{pmatrix}0\\1\\0\\1\end{pmatrix} + \begin{pmatrix}1\\1\\1\\1\end{pmatrix} = \begin{pmatrix}0\\0\\0\\0\end{pmatrix}.$$

de unde se poate construi sistemul de ecuaţii

$$a_0 + 1 = 0, \quad a_3 + 1 = 0, \quad a_2 + 1 = 0, \quad a_1 + a_2 + a_3 + 1 = 0,$$

cu soluţia $a_0 = a_1 = a_2 = a_3 = 1$, deci $m_3(X) = m_6(X) = 1 + X + X^2 + X^3 + X^4$.

Rădăcinile lui $m_5(X)$ sunt α^5, α^{10} $(\alpha^{20} = \alpha^5)$, deci $m_5(X)$ este un polinom de gradul 2 : $m_5(X) = b_0 + b_1X + X^2$. Cum $m_5(\alpha^5) = 0$, se obţine:

$$b_0\begin{pmatrix}1\\0\\0\\0\end{pmatrix} + b_1\begin{pmatrix}0\\1\\1\\0\end{pmatrix} + \begin{pmatrix}1\\1\\1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\0\\0\end{pmatrix}$$

care are soluţia $b_0 = b_1 = 1$, deci $m_5(X) = 1 + X + X^2$.

Polinomul generator al codului C este

$g(X) = cmmmc\{m_1(X), m_2(X), m_3(X), m_4(X), m_5(X), m_6(X)\} =$
$= m_1(X)m_3(X)m_5(X) = (1 + X + X^4)(1 + X + X^2 + X^3 + X^4)(1 + X + X^2) = 1 + X + X^2 + X^4 + X^5 + X^8 + X^{10}$.

De remarcat că era suficient să cerem ca doar $\alpha, \alpha^3, \alpha^5$ să fie rădăcini ale codului.

Fie polinomul $f(X) = a_0 + a_1X + \ldots + a_{n-1}X^{n-1} \in Z_q[X]$ şi α o rădăcină a sa. Atunci $0 = f(\alpha) = a_0 + a_1\alpha + a_2\alpha^2 + \ldots + a_{n-1}\alpha^{n-1}$, sau – altfel scris:

$$(a_0\ a_1\ a_2\ \ldots\ a_{n-1}) \begin{pmatrix} \mathbf{1} \\ \alpha \\ \vdots \\ \alpha^{n-1} \end{pmatrix} = 0.$$

Deci cuvântul - cod corespunzător polinomului $f(X)$ este în spaţiul nul al matricii

$$(\mathbf{1}\ \alpha\ \ldots\ \alpha^{n-1}) \tag{1}$$

Aceasta este şi condiţia de divizibilitate a polinomului $f(X)$ cu polinomul minimal $m(X)$ al lui α. Mulţimea polinoamelor care o satisfac formează idealul generat de $m(X)$. Dimensiunea acestui ideal este (Prelegerea 9) $n - k$ unde $k = grad(m(X))$.

Deci spaţiul liniar generat de matricea (1) are dimensiunea k.

Prin urmare, a cere ca $f(X)$ să admită pe $\alpha_1, \alpha_2, \ldots, \alpha_r \in GF(q^m)$ ca rădăcini este echivalent cu a cere ca vectorul corespunzător să aparţină spaţiului nul al matricii

$$H_{mr,n} = \begin{pmatrix} \mathbf{1} & \alpha_1 & \alpha_1^2 & \ldots & \alpha_1^{n-1} \\ \mathbf{1} & \alpha_2 & \alpha_2^2 & \ldots & \alpha_2^{n-1} \\ & & \vdots & & \\ \mathbf{1} & \alpha_r & \alpha_r^2 & \ldots & \alpha_r^{n-1} \end{pmatrix}$$

Exemplul 11.4 *Să revenim la Exemplul 18.5. Polinomul $\{f(X)\}$ aparţine codului respectiv dacă şi numai dacă vectorul corespunzător aparţine spaţiului nul al matricii*

$$H = \begin{pmatrix} \mathbf{1} & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 & \alpha^7 & \alpha^8 & \alpha^9 & \alpha^{10} & \alpha^{11} & \alpha^{12} & \alpha^{13} & \alpha^{14} \\ \mathbf{1} & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} & \mathbf{1} & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} & \mathbf{1} & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} \\ \mathbf{1} & \alpha^5 & \alpha^{10} & \mathbf{1} & \alpha^5 & \alpha^{10} & \mathbf{1} & \alpha^5 & \alpha^{10} & \mathbf{1} & \alpha^5 & \alpha^{10} & \mathbf{1} & \alpha^5 & \alpha^{10} \end{pmatrix}$$

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

S-a obţinut o matrice 12×15, deşi - conform rezultatelor teoretice, $n = 15$, $n - k = 10$, $k = 5$ şi deci matricea de control a codului ar trebui să aibă dimensiunile $(n-k) \times n = 10 \times 15$. Se observă însă că de fapt ultima linie este nulă, iar următoarele două linii sunt identice.

Dei ultimele două linii se pot ignora, ele neaducând nici o informaţie suplimentară asupra cuvintelor - cod.

11.4 Coduri ciclice ireductibile

Vom mai prezenta şi o a treia construcţie a codurilor ciclice, realizată de Edwin Berlekamp în 1968.

Definiţia 11.7 *Fie polinomul* $Tr(X) = X + X^q + X^{q^2} + \ldots + X^{q^{r-1}} \in GF(q^r)[X]$. *Se numeşte urma lui* $\alpha \in GF(q^r)$, *expresia* $Tr(\alpha)$.

Propoziţia 11.7 Tr *este o aplicaţie* : $GF(q^r) \to Z_q$.

Demonstraţie: Trebuie arătat că $\forall \alpha \in GF(q^r)$ avem $Tr(\alpha) \in Z_q$. Pentru aceasta este suficient să arătăm că $Tr(\alpha)$ verifică ecuaţia $X^q - X = 0$. Folosind faptul că că $GF(q^r)$ este corp de caracteristică q şi că $\forall \alpha \in GF(q^r)$ avem $\alpha^{q^r} = \alpha$, se verifică imediat relaţia $[TR(\alpha)]^q = TR(\alpha)$. q.e.d.

Propoziţia 11.8 Tr *este aplicaţie liniară.*

Demonstraţie: Relaţia $Tr(\alpha + \beta) = Tr(\alpha) + Tr(\beta)$ este uşor de verificat, deoarece se lucrează în corpuri de caracteristică q. q.e.d.

Propoziţia 11.9 *Pentru orice* $a \in Z_q$ *există* q^{r-1} *valori* $\alpha \in GF(q^r)$ *cu* $Tr(\alpha) = a$.

Demonstraţie: Fiecare ecuaţie $Tr(X) = a$ admite maxim q^{r-1} rădăcini, iar numărul de elemente a posibile este q. Mai trebuie arătat că toate cele q ecuaţii au rădăcini distincte. Aceasta se poate deduce foarte simplu folosind derivata formală într-un corp de caracteristică q: toate ecuaţiile au aceeaşi derivată: 1. q.e.d.

Teorema 11.11 *Polinomul* $Tr(X)$ *se poate descompune în* $GF(q^r)$ *în produs de polinoame minimale.*

Demonstraţie: Demonstraţia se bazează pe următorul algoritm:

(1) Fie polinomul $g(X) = Tr(X)$;

(2) Se consideră $\alpha \in GF(q^r)$ cu $g(\alpha) = 0$. Deci $g(X)$ este divizibil cu polinomul minimal $m(X)$ al lui α.

(3) $g(X) := g(X)/m(X)$. Dacă $g(X) = 1$, STOP, altfel se reia pasul (2).

Existenţa rădăcinilor pentru $g(X)$ (pasul (2)) este asigurată de faptul că în $GF(q^r)$ orice polinom ireductibil este polinom minimal. q.e.d.

Exemplul 11.5 *Să considerăm* $GF(2^4)$, *deci* $q = 2$, $r = 4$. *Aici se poate verifica descompunerea*

$$Tr(X) = X + X^2 + X^4 + X^8 = X(1 + X)(1 + X + X^2)(1 + X + X^4),$$

formată din produs de polinoame minimale.

Fie $\alpha \in GF(2^4)$ *rădăcină (primitivă) a ecuaţiei* $1 + X + X^4 = 0$. *Deci toate elementele nenule din* $GF(2^4)$ *se pot scrie ca puteri ale lui* α.

Vom folosi aceasta pentru a lista valorile funcţiei $Tr : GF(2^4) \to Z_2$:

X	$Tr(X)$	X	$Tr(X)$	X	$Tr(X)$	X	$Tr(X)$
0	0	α^3	1	α^7	1	α^{11}	1
1	0	α^4	0	α^8	0	α^{12}	1
α	0	α^5	0	α^9	1	α^{13}	1
α^2	0	α^6	1	α^{10}	0	α^{14}	1

După cum se observă, sunt $8 = 2^3$ *valori* 0 *şi* 8 *valori* 1.

Putem da acum teorema principală care defineşte codurile ciclice folosind operatorul Tr:

Teorema 11.12 *Fie* n *număr natural,* q *număr prim,* k *ordinul lui* q *modulo* n ($q^k \equiv 1 \ mod \ n$) *şi* $\beta \in GF(q^k)$ *element primitiv de ordin* n. *Atunci*

$$C = \{\mathbf{c}_\alpha = (Tr(\alpha), Tr(\alpha\beta), Tr(\alpha\beta^2), \ldots, Tr(\alpha\beta^{n-1}))|\alpha \in GF(q^k)\}$$

este un (n, k) *- cod ciclic peste* Z_q.

Demonstraţie: Din Propoziţia 11.8 rezultă că C este cod liniar.

Se verifică apoi că $\mathbf{c}_{\alpha\beta^{-1}} \in C$ este o permutare ciclică a lui $\mathbf{c}_\alpha$; deci C este cod ciclic. Deoarece β este primitiv, înseamnă că polinomul său minimal $m(X) = h_0 + h_1X + \ldots + h_kX^k$ are gradul k. Dacă $\mathbf{c}_\alpha = (c_0, c_1, \ldots, c_{n-1})$, avem

$$\sum_{i=0}^{k} c_i h_i = Tr(\alpha m(\beta)) = Tr(\mathbf{0}) = 0,$$

care constituie una din cele $n - k$ ecuaţii de control pentru C.

Deoarece $m(X)$ este minimal (deci ireductibil), $h(X) = X^k m(X^{-1})$ este polinomul de control pentru C. Cum gradul lui este k, avem un (n, k) - cod ciclic peste Z_q. q.e.d.

De remarcat că această teoremă defineşte numai codurile ciclice "ireductibile" (care nu conţin subcoduri ciclice netriviale proprii), deci clasa lor este inclusă strict în clasa codurilor ciclice dată de cele două definiţii anterioare (prin ideale şi prin rădăcinile polinomului generator).

Exemplul 11.6 *Să luăm* $n = 15$, $q = 2$, *deci* $k = 4$. *În* $GF(2^4)$ *vom lua* α, *rădăcină a polinomului* $1 + X + X^4$. *Se ştie că el este un element primitiv. Aplicaţia* Tr *este* $Tr(X) = X + X^2 + X^4 + X^8$ *iar codul* C *este format din* 16 *cuvinte de forma*

$$(Tr(\beta), Tr(\beta\alpha), \ldots, Tr(\beta\alpha^{n-1})), \ \forall \beta \in GF(2^4).$$

Pentru $\beta = 0$, $\quad 00\ldots0 \in C$.

Pentru $\beta = 1$, $\quad (Tr(1), Tr(\alpha), \ldots, Tr(\alpha^{n-1})) = 000100110101111$

Celelalte 14 *cuvinte sunt permutările circulare ale acestuia.*

Polinomul de control al codului este $h(X) = X^4\left(1 + \frac{1}{X} + \frac{1}{X^4}\right) = 1 + X^3 + X^4$, *iar polinomul generator*

$$g(X) = \frac{X^{16} - X}{X^4 + X^3 + 1} = X + X^4 + X^5 + X^7 + X^9 + X^{10} + X^{11} + X^{12},$$

al cărui vector corespunzător se obţine în construcţia de sus pentru $\beta = \alpha^2$.

Acesta este un $(15, 4)$ *- cod ciclic binar, ireductibil.*

De remarcat că pentru $n = 15$ *sunt numai două astfel de coduri, celălalt fiind codul dual (rădăcina polinomului* $1 + X^3 + X^4$ *este de asemenea primitivă).*

11.5 Exerciţii

11.1 *Demonstraţi Corolarul 15.1.*

11.2 *Demonstraţi Propoziţia 11.6*

11.3 *Construiţi următoarele corpuri:*
(a) $GF(2^2)$*;*
(b) $GF(2^3)$ *folosind polinomul* $1 + X^2 + X^3$*;*
(c) $GF(2^4)$ *folosind polinomul* $1 + X^3 + X^4$*;*
(d) $GF(2^5)$ *folosind polinomul* $1 + X^2 + X^5$*.*

11.4 *Găsiţi o extensie a lui* Z_2 *în care* $X^9 - 1$ *să se descompună în factori liniari.*

11.5 *Găsiţi toate elementele primitive din* $GF(2^3)$ *şi* $GF(2^4)$*.*

11.6 *Găsiţi toate elementele primitive din* $GF(3^2)$ *şi* $GF(5^2)$*.*

11.7 *Construiţi* $GF(2^4)$ *ca o extensie a lui* $GF(2^2)$*.*

11.8 *Fie* $\alpha \in GF(2^4)$*, rădăcină a polinomului* $1 + X + X^4$*.*
Găsiţi polinoamele minimale al lui $\beta = \alpha^7$ *şi* $\beta = \alpha^{10}$*.*

11.9 *Găsiţi polinoamele minimale al tuturor elementelor din:*
- $GF(2^3)$*, generat cu* $1 + X + X^3$*;*
- $GF(2^5)$*, generat cu* $1 + X^2 + X^5$*;*
- $GF(3^2)$*, generat cu* $2 + X + X^2$*;*
- $GF(5^2)$*, generat cu* $3 + 2X + X^2$*.*

11.10 *Orice element nenul din* $GF(q^m)$ *are un ordin prim cu* q*.*

11.11 *Găsiţi polinomul generator al unui cod ciclic binar de lungime* 15*, de rădăcini* $1, \alpha^5, \alpha^7, \quad \alpha \in GF(2^4)$ *fiind rădăcină a polinomului* $1 + X + X^4$*.*
Construiţi matricea de control a codului.
Arătaţi că $v(X)$ *este polinom - cod dacă şi numai dacă ponderea* $w(\mathbf{v})$ *este pară.*

11.12 *Găsiţi polinomul generator al unui cod ciclic binar de rădăcini* $\alpha^2, \alpha^3, \alpha^6$ *unde* $\alpha \in GF(2^3)$ *este rădăcină a polinomului* $1 + X + X^3$*.*

11.13 *Să se descompună* $Tr(X)$ *în produs de polinoame minimale în corpurile:*
(a) $GF(2^5)$*;* *(b)* $GF(3^2)$*;* *(c)* $GF(3^3)$*;* *(d)* $GF(5^2)$*.*

11.14 *Folosind Teorema 11.12 să se construiască toate codurile ireductibile pentru*
$GF(2^3), \quad GF(2^5), \quad GF(3^2).$

11.15 *Verificaţi că polinomul* $h(X)$ *construit în demonstraţia Teoremei 11.12 este polinomul generator al codului dual.*

Capitolul 12

Coduri BCH

Codurile BCH constituie cea mai relevantă clasă de coduri ciclice. Ele au fost definite în mod independent de Bose şi Chaudhuri pe de-o o parte, Hocquenheim pe de de-altă parte. Numele de BCH (**B**ose - **C**haudhuri - **H**ocquenheim) a fost dat de Peterson, care s-a ocupat de ele în mod special, construind şi algoritmul de decodificare.

12.1 Definirea codurilor BCH

Definiţia 12.1 *Fie q număr prim, m_0, m, d numere naturale şi $\alpha \in GF(q^m)$. Un cod ciclic este cod BCH dacă este definit de rădăcinile polinomului generator*

$$\alpha^{m_0}, \alpha^{m_0+1}, \ldots, \alpha^{m_0+d-2}.$$

În majoritatea cazurilor studiate se consideră $m_0 = 0$ sau $m_0 = 1$.

Teorema 12.1 *Într-un cod BCH lungimea n a cuvintelor - cod este egală cu ordinul e al lui α.*

Demonstraţie: Conform construcţiei codurilor ciclice bazată pe rădăcinile polinomului generator (Prelegerea 11), lungimea n a cuvintelor - cod este cel mai mic multiplu comun al ordinelor rădăcinilor. În cazul codurilor BCH avem

$$(\alpha^{m_0})^n = \alpha^{m_0 n} = 1 \text{ şi } 1 = (\alpha^{m_0+1})^n = \alpha^{m_0+n} = \alpha^{m_0}\alpha^n,$$

deci $\alpha^n = 1$. Prin urmare $e|n$ şi $n \geq e$.

De asemenea, $(\alpha^j)^e = (\alpha^e)^j = 1 \; \forall j = m_0, m_0 + 1, \ldots, m_0 + d - 2$, adică $n \leq e$.

Rezultă $n = e$. q.e.d.

De remarcat că, dacă α este primitiv, atunci $e = q^m - 1$.

Teorema 12.2 *Într-un cod BCH definit de rădăcinile $\alpha^{m_0}, \alpha^{m_0+1}, \ldots, \alpha^{m_0+d-2}$ ($\alpha \in GF(q^m)$), distanţa minimă a codului este cel puţin d (şi deci codul poate corecta orice combinaţie de $t \leq \left[\frac{d-1}{2}\right]$ erori independente).*

Demonstraţie: Un cod BCH este spaţiul nul al matricii

$$H = \begin{pmatrix} 1 & \alpha^{m_0} & (\alpha^{m_0})^2 & \dots & (\alpha^{m_0})^{n-1} \\ 1 & \alpha^{m_0+1} & (\alpha^{m_0+1})^2 & \dots & (\alpha^{m_0+1})^{n-1} \\ & & \vdots & & \\ 1 & \alpha^{m_0+d-2} & (\alpha^{m_0+d-2})^2 & \dots & (\alpha^{m_0+d-2})^{n-1} \end{pmatrix}$$

Să considerăm următorul determinant, obţinut prin alegerea arbitrară a $d-1$ coloane din H:

$$D = \begin{vmatrix} (\alpha^{m_0})^{j_1} & (\alpha^{m_0})^{j_2} & \dots & (\alpha^{m_0})^{j_{d-1}} \\ (\alpha^{m_0+1})^{j_1} & (\alpha^{m_0+1})^{j_2} & \dots & (\alpha^{m_0+1})^{j_{d-1}} \\ & & \vdots & \\ (\alpha^{m_0+d-2})^{j_1} & (\alpha^{m_0+d-2})^{j_2} & \dots & (\alpha^{m_0+d-2})^{j_{d-1}} \end{vmatrix} =$$

$$= \alpha^{m_0(j_1+j_2+\dots+j_{d-1})} \begin{vmatrix} 1 & 1 & \dots & 1 \\ \alpha^{j_1} & \alpha^{j_2} & \dots & \alpha^{j_{d-1}} \\ & & \vdots & \\ (\alpha^{j_1})^{d-2} & (\alpha^{j_2})^{d-2} & \dots & (\alpha^{j_{d-1}})^{d-2} \end{vmatrix} =$$

$$= \alpha^{m_0(j_1+j_2+\dots+j_{d-1})} \prod_{i>k} (\alpha^{j_i} - \alpha^{j_k}) \neq 0$$

pentru că $j_i \neq j_k$ dacă $i \neq k$.

Deci orice submulţime formată cu maxim $d-1$ coloane din H este liniar independentă de unde – conform Teoremei 2.4 (Prelegerea 2) – distanţa minimă a codului este cel puţin d. q.e.d.

Exemplul 12.1 *Dacă se lucrează în binar, puterile pare ale rădăcinilor se pot omite din listă(dacă α^i este rădăcină, atunci şi α^{2i} este rădăcină – conform Teoremei* 11.9*, Prelegerea* 11*). Atunci un cod BCH binar corector de* 3 *erori este definit de rădăcinile β, β^3, β^5, unde β este un element primitiv dintr-o extensie a lui Z_2. De exemplu, dacă $\beta = \alpha$, rădăcină a polinomului $1 + X + X^4$, se obţine codul din Exemplul* 11.3*, Prelegerea* 11*.*

Exemplul 12.2 *Fie $GF(3^2)$ generat de rădăcina α a polinomului ireductibil $g(X) = 2 + X + X^2$. Deoarece $ord(\alpha) = 8$, orice cod generat va avea lungimea* 8*. Să construim codul BCH de rădăcini $\alpha^2, \alpha^3, \alpha^4$.*
Polinoamele minimale sunt:

$m_2(X) = 1 + X^2$ *cu rădăcini* α^2, α^6*;*
$m_3(X) = 2 + X + X^2$ *cu rădăcini* α^3, α
$m_4(X) = 1 + X$*, numai cu rădăcina* α^4*.*

Deci polinomul generator este

$$g(X) = (1 + X^2)(2 + X + X^2)(1 + X) = 2 + X^2 + X^3 + 2X^4 + X^5.$$

Codul este un $(8, 3)$ *- cod ciclic. Deoarece printre rădăcinile sale este şi α, se poate considera $m_0 = 1$, $d = 5$, deci codul poate corecta cel mult două erori independente.*

De remarcat că dacă folosim acest cod drept cod liniar, cum ponderea minimă a cuvintelor sale este 4 *(de exemplu* 02010201 *este cuvânt - cod), distanţa minimă este* 4*, deci codul nu poate corecta decât o singură eroare. În general, tratarea codurilor ciclice sub formă de coduri liniare scade de obicei puterea de corecţie a erorilor independente.*

Numerele m, m_0, d, q constituie parametrii codului BCH. Ideea generală este de a-l mări pe d, pentru a putea corecta cât mai multe combinaţii de erori. Acest lucru este posibil totdeauna, deoarece m şi q sunt alese arbitrar.

Să observăm că pentru $\alpha \in GF(q^m)$ primitiv, toate puterile lui α până la $q^m - 1$ sunt distincte. În cazul binar, avem următorul rezultat:

Teorema 12.3 *Fie m un număr natural oarecare şi $\alpha \in GF(2^m)$ primitiv. Atunci există un cod BCH de lungime $n = 2^m - 1$ care corectează orice combinaţie de $t \leq 2^{m-1} - 1$ erori independente, folosind cel mult mt poziţii de control.*

Demonstraţie: Vom considera codul binar definit de rădăcinile $\alpha, \alpha^2, \ldots, \alpha^{2t}$. Acesta este un cod BCH pentru $q = 2,\ m_0 = 1,\ d = 2t + 1,\ n = 2^m - 1$ şi cu distanţa minimă cel puţin $2t + 1$. Pentru ca rădăcinile menţionate mai sus să fie distincte, trebuie ca $2t \leq n - 1 = 2^m - 1$.

Să observăm că orice putere pară a lui α este rădăcină a unui polinom minimal corespunzţor unei puteri impare anterioare a lui α. Deci este suficient să dăm la început doar rădăcinile $\alpha, \alpha^3, \ldots, \alpha^{2t-1}$ ale cuvintelor codului binar.

Avem – după definiţie: $g(X) = cmmmc\{m_1(X), m_3(X), \ldots, m_{2t-1}(X)\}$.

Pe de - altă parte, $m_i(X) | X^n - 1 = X^{2^m-1} - 1$ şi deci – conform Teoremei 11.7 (Prelegerea 11), $grad(m_i(X)) \leq m$. Din cele două relaţii rezultă $grad(g(X)) \leq mt$.

Cum $grad(g(X))$ dă numărul de simboluri de control, teorema este demonstrată.q.e.d.

12.2 Algoritmul Peterson de decodificare

Fie codul BCH definit de rădăcinile $\alpha^{m_0}, \alpha^{m_0+1}, \ldots, \alpha^{m_0+2t-1}$ $(d = 2t + 1)$ cu $\alpha \in GF(q^m)$.

Un asemenea cod corectează orice combinaţie de maxim t erori independente.

Fie $\{f(X)\}$ un cuvânt - cod transmis şi

$$\{r(X)\} = \{f(X) + e(X)\} = \{f(X)\} + \{e(X)\}$$

secvenţa recepţionată, unde $\{e(X)\}$ este vectorul - eroare.

Avem, pentru $j = m_0, m_0 + 1, \ldots, m_0 + 2t - 1$:

$$S_j = r(\alpha^j) = f(\alpha^j) + e(\alpha^j) = e(\alpha^j).$$

S_j se numesc *componentele sindromului*.

La decodificare, eroarea-tip $\mathbf{e}$ este determinată complet dacă se ştiu:

- Valorile componentelor sale $Y_i \in Z_q$;
- Localizările lor $X_i \in GF(q^m)$.

Un cuvânt din $GH(q^m)$ este un vector cu q^m componente; vom nota poziţiile acestora folosind puterile lui α^i: pe poziţia i se află coeficientul lui α^{i-1} $(1 \leq i \leq q^m)$. Deci α^{i-1} va localiza a i - a componentă dintr-un cuvânt.

Deci, dacă au intervenit t erori, eroarea - tip este complet caracterizată de cunoaşterea celor t perechi nenule (Y_i, X_i) $(1 \leq i \leq t)$.

Putem conveni (evident) că

$$Y_i = 0 \qquad \Longleftrightarrow \qquad X_i = 0.$$

De remarcat că X_i nu reprezintă α^i ci un α^j arbitrar, care va trebui aflat.

Deci, a găsi pe $\{e(X)\}$ revine la a determina $2t$ elemente $X_i, Y_i \ \ (1 \leq i \leq t)$, grupate în t perechi. Vom considera $Y_i = 0 \ \forall i \ (t < i \leq q^m - 1)$. Avem – conform notaţiei polinomiale

$$S_j = e(\alpha^j) = \sum_{i=1}^{t} Y_i X_i^j, \qquad (m_0 \leq j \leq m_0 + 2t - 1)$$

Atunci

$$(S_j)^q = \left(\sum_{i=1}^{t} Y_i X_i^j \right)^q = \sum_{i=1}^{t} Y_i^q X_i^{jq} = \sum_{i=1}^{t} Y_i X_i^{jq} = S_{jq}.$$

Deci, determinarea polinomului - eroare $\{e(X)\}$ se reduce la aflarea soluţiilor $(Y_i, X_i), \ (1 \leq i \leq t)$ ale sistemului de ecuaţii

$$\sum_{i=1}^{t} Y_i X_i^j = S_j, \quad m_0 \leq j \leq m_0 + 2t - 1, \tag{1}$$

Dacă au intervenit $k \ (k < t)$ erori, se completează cele k perechi nenule (Y_i, X_i) cu alte $t - k$ perechi nule $(0, 0)$. De fapt, acest număr k nu este cunoscut apriori; de aceea, se pleacă de la t erori potenţiale (maximul pe care codul BCH îl poate corecta în mod cert), caracterizate de t perechi (Y_i, X_i). Problema care se pune este deci de a-l determina pe k şi cele $2k$ necunoscute $Y_i, X_i \ \ (1 \leq i \leq k)$.

Orice metodă de corectare a erorilor revine la rezolvarea sistemului de ecuaţii (1), care este un sistem neliniar de $2t$ ecuaţii cu $2t$ necunoscute, problemă NP - completă. Să încercăm o abordare mai simplă a acestei probleme.

Fie polinomul

$$(X_1 - X)(X_2 - X) \ldots (X_t - X) = \sigma_t + \sigma_{t-1}(-X) + \ldots + \sigma_1(-X)^{t-1} + (-X)^t.$$

unde σ_j sunt funcţiile simetrice date de relaţiile lui $Viète$.

Dacă în această relaţie se ia $X = X_i \ \ (1 \leq i \leq t)$, se ajunge la

$$X_i^t + (-1)\sigma_1 X_i^{t-1} + \ldots + (-1)^{t-1}\sigma_{t-1} X_i + (-1)^t \sigma_t = 0, \ \ (1 \leq i \leq t).$$

Înmulţim cu $Y_i X_i^j$ şi sumăm după i. Se obţine:

$$\sum_{i=1}^{t} Y_i X_i^{j+t} + (-1)\sigma_1 \sum_{i=1}^{t} Y_i X_i^{j+t-1} + \ldots + (-1)^{t-1}\sigma_{t-1} \sum_{i=1}^{t} Y_i X_i^{j+1} + (-1)^t \sigma_t \sum_{i=1}^{t} Y_i X_i^j = 0,$$
$$j = m_0, m_0 + 1, \ldots, m_0 + t - 1$$

relaţie care se poate scrie

$$S_j(-1)^t \sigma_t + S_{j+1}(-1)^{t-1}\sigma_{t-1} + \ldots + S_{j+t-1}(-1)\sigma_1 + S_{j+t} = 0, \tag{2}$$
$$m_0 \leq j \leq m_0 + t - 1$$

Algoritmul lui Peterson de corectare a $k \ (k \leq t)$ erori într-un cod BCH este:

1. Se rezolvă sistemul (2) în care necunoscutele sunt $(-1)^p\sigma_p,\ (1 \le p \le t)$;
2. Se înlocuiesc valorile aflate în ecuaţia
$$X^t + (-1)\sigma_1 X^{t-1} + \ldots + (-1)^{t-1}\sigma_{t-1}X + (-1)^t\sigma_t = 0$$
şi se află cele t rădăcini $X_1, X_2, \ldots, X_t \in GF(q^m)$ ale sale;
3. Cu aceste valori introduse în sistemul (1), se determină din primele t ecuaţii valorile $Y_1, Y_2, \ldots, Y_t \in Z_q$;
4. Dacă $(Y_i, \alpha^{ji})\ \ (1 \le i \le k)$ sunt cele $k \le t$ valori nenule ale soluţiei obţinute, atunci:
   - Pentru $k = 0$, cuvântul recepţionat este $\{r(X)\}$ (fără erori);
   - Pentru $0 < k \le t$, fie $e(X) = \sum_{i=1}^{k} Y_i X^{j_i}$.

   Cuvântul - cod decodificat este $\{f(X)\} = \{r(X)\} - \{e(X)\}$.

De remarcat că în cazul binar, pasul (3) nu mai este necesar, deoarece
$$Y_i = 1,\ \forall\ i \le k.$$
Pentru corectitudinea algoritmului, trebuie studiate compatibilităţile sistemelor de ecuaţii (1) şi (2).

Teorema 12.4 *Sistemul* $\sum_{i=1}^{t} Y_i X_i^j = S_j\ \ (m_0 \le j \le m_0 + t - 1)$ *de necunoscute* Y_i *este compatibil dacă şi numai dacă au intervenit* t *erori.*

Demonstraţie: Considerat ca sistem în Y, sistemul este liniar; determinantul său principal este
$$\begin{vmatrix} X_1^{m_0} & X_2^{m_0} & \ldots & X_t^{m_0} \\ X_1^{m_0+1} & X_2^{m_0+1} & \ldots & X_t^{m_0+1} \\ X_1^{m_0+2} & X_2^{m_0+2} & \ldots & X_t^{m_0+2} \\ & & \vdots & \\ X_1^{m_0+t-1} & X_2^{m_0+t-1} & \ldots & X_t^{m_0+t-1} \end{vmatrix} = X_1^{m_0} X_2^{m_0} \ldots X_t^{m_0} \begin{vmatrix} 1 & 1 & \ldots & 1 \\ X_1 & X_2 & \ldots & X_t \\ X_1^2 & X_2^2 & \ldots & X_t^2 \\ & & \vdots & \\ X_1^{t-1} & X_2^{t-1} & \ldots & X_t^{t-1} \end{vmatrix}$$
$$= X_1^{m_0} X_2^{m_0} \ldots X_t^{m_0} \prod_{1 \le j < i \le t} (X_i - X_j)$$
Acesta este nenul numai dacă au intervenit t erori, pentru că atunci toţi X_i sunt distincţi şi nenuli. q.e.d.

Teorema 12.5 *Matricea* $M = \begin{pmatrix} S_{m_0} & S_{m_0+1} & \dots & S_{m_0+t-1} \\ S_{m_0+1} & S_{m_0+2} & \dots & S_{m_0+t} \\ & & \vdots & \\ S_{m_0+t-1} & S_{m_0+t} & \dots & S_{m_0+2t-1} \end{pmatrix}$
este nesingulară dacă au intervenit t erori şi este singulară dacă au intervenit mai puţin de t erori.

Demonstraţie: Ţinând cont de forma sistemului (1), este posibilă descompunerea matricii M în produs de 3 matrici $M = ABA^T$ unde:

$$A = \begin{pmatrix} 1 & 1 & \dots & 1 \\ X_1 & X_2 & \dots & X_t \\ X_1^2 & X_2^2 & \dots & X_t^2 \\ & & \vdots & \\ X_1^{t-1} & X_2^{t-1} & \dots & X_t^{t-1} \end{pmatrix} \qquad B = \begin{pmatrix} Y_1X_1^m & 0 & \dots & 0 \\ 0 & Y_2X_2^m & \dots & 0 \\ & & \vdots & \\ 0 & 0 & \dots & Y_tX_t^m \end{pmatrix}.$$

M este nesingulară dacă cele două matrici din descompunere sunt nesingulare. Matricea A este nesingulară dacă şi numai dacă valorile X_i $(1 \le i \le t)$ sunt distincte şi nenule, deci dacă au intervenit t erori. Matricea B este nesingulară dacă toate elementele de pe diagonala sa principală sunt nenule, deci dacă toate perechile (Y_i, X_i) $(1 \le i \le t)$ sunt nenule – adică au apărut t erori. q.e.d.

Aplicarea algoritmului Peterson se face în felul următor:

1. Folosind polinomul recepţionat $\{r(X)\}$, se determină
$$S_j = r(\alpha^j) \quad (m_0 \le j \le m_0 + 2t - 1)$$

2. Se studiază compatibilitatea sistemului
$$S_j(-1)^t\sigma_t + S_{j+1}(-1)^{t-1}\sigma_{t-1} + \ldots + S_{j+t-1}(-1)\sigma_1 + S_{j+t} = 0,$$
$$m_0 \le j \le m_0 + t - 1.$$

   Dacă matricea M a acestui sistem este nesingulară, înseamnă că au apărut t erori. Dacă M este singulară, atunci se observă că $\sigma_t = 0$ (deoarece $\sigma_t = X_1X_2 \ldots X_t$) şi – după ce se înlocuieşte $\sigma_t = 0$ – se obţine un nou sistem liniar format din $t-1$ ecuaţii (neglijând prima sau ultima din ecuaţiile de mai sus) cu $t-1$ necunoscute.

   Se studiază matricea acestui nou sistem. Dacă ea este nesingulară, înseamnă că au apărut $t-1$ erori. Dacă şi aceasta este singulară, cum $\sigma_{t-1} = 0$, se repetă procedeul.

   Primul sistem compatibil (cu matricea nesingulară) care se obţine va da numărul k $(k \le t)$ de erori care au intervenit în transmisie, precum şi valorile
$$(-1)\sigma_1,\ (-1)^2\sigma_2, \ldots, (-1)^k\sigma_k.$$

3. Se introduc valorile determinate mai sus în polinomul
$$P = X^k + (-1)\sigma_1X^{k-1} + \ldots + (-1)^{k-1}\sigma_{k-1}X + (-1)^k\sigma_k$$
   şi se află cele k rădăcini $X_1, X_2, \ldots, X_k \in GF(q^m)$ ale ecuaţiei $P = 0$ (prin diverse metode, eventual chiar prin simpla verificare a ecuaţiei cu toate elementele din $GF(q^m)$). Aceste rădăcini vor da localizările erorilor.

   De remarcat că ordinea de notare a rădăcinilor $X_1, X_2, \ldots, X_t$ este neesenţială.

4. Se introduc aceste soluţii în ecuaţiile sistemului

$$\sum_{i=1}^{k} Y_i X_i^j = S_j, \quad j = m_0, m_0+1, \ldots, m_0+k-1$$

şi se determină valorile erorilor $Y_1, Y_2, \ldots, Y_k$.

5. Se calculează astfel polinomul eroare $e(X) = \sum_{i=1}^{k} Y_i X^{j_i}$, unde $X_i = \alpha^{j_i}$ $(1 \leq i \leq k)$.

Acesta se scade din polinomul recepţionat şi se obţine cuvântul - cod transmis:

$$\{r(X)\} - \{e(X)\} = \{r(X) - e(X)\} = \{f(X)\}$$

Exemplul 12.3 *Fie codul BCH definit de rădăcinile* $\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5\alpha^6$ *unde* $\alpha \in GF(2^4)$ *este element primitiv, rădăcină a polinomului* $1 + X + X^4$ *(Exemplul* 11.3*, Prelegerea* 11 *şi Exemplul 20.3).*

Deoarece $t = 3$*, acest cod este capabil să corecteze maxim* 3 *erori independente.*

Să presupunem că au apărut două erori, pe poziţiile α^3 *şi* α^{10} *(adică pe poziţiile* 4 *şi* 11*) ale cuvântului transmis. Ele sunt reprezentate prin polinomul eroare* $e(X) = X^3 + X^{10}$*.*

Să calculăm sindromul (pentru operaţile folosite, a se vedea Exemplele 8.8 *şi* 8.9*, Prelegerea* 8*):*

$$\begin{aligned}
&S_1 = r(\alpha) = e(\alpha) = \alpha^3 + \alpha^{10} = \alpha^{12}\\
&S_2 = (S_1)^2 = \alpha^{24} = \alpha^9\\
&S_3 = r(\alpha^3) = e(\alpha^3) = \alpha^9 + \alpha^{30} = \alpha^9 + \alpha^0 = \alpha^7\\
&S_4 = (S_2)^2 = \alpha^{18} = \alpha^3\\
&S_5 = r(\alpha^5) = e(\alpha^5) = \alpha^{15} + \alpha^{50} = \alpha^0 + \alpha^5 = \alpha^{10}\\
&S_6 = (S_3)^2 = \alpha^{14}.
\end{aligned}$$

Observăm că într-o situaţie reală, nu se va şti unde sunt localizate aceste erori şi deci – în cazul în care au intervenit două erori pe poziţiile α^3 *şi* α^{10} *(care nu se cunosc încă) – din vectorul recepţionat se obţine direct sindromul*

$$S = (S_1, S_2, S_3, S_4, S_5, S_6) = (\alpha^{12}, \alpha^9, \alpha^7, \alpha^3, \alpha^{10}, \alpha^{14}),$$

fără să ştim din combinaţiile căror localizări ale erorilor provin aceste componente ale sindromului.

Să considerăm sistemul de ecuaţii

$$\left\{\begin{array}{lcl} S_1\sigma_3 + S_2\sigma_2 + S_3\sigma_1 &=& S_4\\ S_2\sigma_3 + S_3\sigma_2 + S_4\sigma_1 &=& S_5\\ S_2\sigma_3 + S_4\sigma_2 + S_5\sigma_1 &=& S_6 \end{array}\right. \quad \textit{adică} \quad \left\{\begin{array}{lcl} \alpha^{12}\sigma_3 + \alpha^9\sigma_2 + \alpha^7\sigma_1 &=& \alpha^3\\ \alpha^9\sigma_3 + \alpha^7\sigma_2 + \alpha^7\sigma_1 &=& \alpha^{10}\\ \alpha^7\sigma_3 + \alpha^3\sigma_2 + \alpha^{10}\sigma_1 &=& \alpha^{14} \end{array}\right.$$

Înmulţim cele trei ecuaţii ale sistemului respectiv cu

$$\alpha^{-7} = \alpha^8,\ \alpha^{-3} = \alpha^{12},\ \alpha^{-10} = \alpha^5$$

şi obţinem:

$$\left\{\begin{array}{lcl} \alpha^5\sigma_3 + \alpha^2\sigma_2 + \sigma_1 &=& \alpha^{11}\\ \alpha^6\sigma_3 + \alpha^4\sigma_2 + \sigma_1 &=& \alpha^7\\ \alpha^{12}\sigma_3 + \alpha^8\sigma_2 + \sigma_1 &=& \alpha^4 \end{array}\right.$$

Adunăm prima ecuaţie la ultimele două şi atunci

$$\left\{\begin{array}{lcl} \alpha^9\sigma_3 + \alpha^{10}\sigma_2 &=& \alpha^8\\ \alpha^{14}\sigma_3 + \alpha^0\sigma_2 &=& \alpha^{13} \end{array}\right.$$

Deoarece a doua ecuaţie se obţine din prima prin înmulţire cu α^5, ele nu sunt independente; deci au apărut mai puţin de 3 erori. Facem $\sigma_3 = 0$ şi a doua ecuaţie va da $\sigma_2 = \alpha^{13}$. Din prima ecuaţie a sistemului se ajunge la $\alpha^0 + \sigma_1 = \alpha^{11}$, adică $\sigma_1 = \alpha^{12}$.

Deci, în acest caz $k = 2$, $\sigma_1 = \alpha^{12}$, $\sigma_2 = \alpha^{13}$. Se ajunge la ecuaţia $X^2 + \sigma_1 X + \sigma_2 = X^2 + \alpha^{12}X + \alpha^{13} = 0$, care are ca rădăcini $X_1 = \alpha^3$, $X_2 = \alpha^{10}$.

Am redescoperit astfel cele două localizări ale erorilor. Codul fiind binar, corectarea se face imediat: se modifică (din 1 în 0 sau din 0 în 1) biţii de pe poziţiile 4 respectiv 11 din cuvântul recepţionat.

Exemplul 12.4 *Fie codul BCH definit de rădăcinile $\alpha, \alpha^2, \alpha^3, \alpha^4$ unde $\alpha \in GF(2^4)$ este rădăcina polinomului $1 + X + X^4$. Acest cod are $n = 15$, $m_0 = 1$, $m_0 + 2t - 1 = 4$, deci $t = 2$; codul poate corecta în mod sigur cel mult două erori independente.*

Să presupunem că s-a recepţionat cuvântul

$$\mathbf{r} = 100100110000100.$$

Să determinăm cuvântul - cod transmis.

Polinomul corespunzător este $r(X) = 1 + X^3 + X^6 + X^7 + X^{12}$, iar componentele sindromului sunt

$$S_1 = r(\alpha) = \alpha^0, \qquad S_2 = r(\alpha^2) = (S_1)^2 = \alpha^0,$$
$$S_3 = r(\alpha^3) = \alpha^4, \qquad S_4 = r(\alpha^4) = (S_2)^2 = \alpha^0.$$

Trebuie rezolvat sistemul

$$\left\{ \begin{array}{rcl} S_1\sigma_2 + S_2\sigma_1 & = & S_3 \\ S_2\sigma_2 + S_3\sigma_1 & = & S_4 \end{array} \right. \quad \textit{adică} \quad \left\{ \begin{array}{rcl} \alpha^0\sigma_2 + \alpha^0\sigma_1 & = & \alpha^4 \\ \alpha^0\sigma_2 + \alpha^4\sigma_1 & = & \alpha^0 \end{array} \right. ,$$

de unde se obţine $\sigma_1 = \alpha^0$, $\sigma_2 = \alpha$.

Polinomul de localizare a erorilor este $X^2 + \sigma_1 X + \sigma_2 = X^2 + X + \alpha$, cu rădăcinile $X_1 = \alpha^7$, $X_2 = \alpha^9$. Au apărut deci două erori, pe poziţiile 8 şi 10. Pentru că suntem în cazul binar, $Y_1 = Y_2 = 1$; vectorul - eroare obţinut este $e(X) = X^7 + X^9$, deci cuvântul - cod transmis este

$$\{f(X)\} = \{r(X)\} - \{e(X)\} = \{r(X) + e(X)\} = 1 + X + X^6 + X^9 + X^{12}$$

adică $\mathbf{f} = 100100100100100$.

12.3 Localizarea erorilor la codurile BCH binare

După cum s-a observat, în cazul codurilor BCH binare este suficient să se localizeze eroarea; corectarea se face imediat prin complementarea biţilor aflaţi pe poziţiile respective. După determinarea valorilor $\sigma_1, \sigma_2, \ldots, \sigma_t$, algoritmul Peterson nu explicitează modul de rezolvare a ecuaţiei de localizare al erorii, care uneori poate fi dificil de prelucrat.

Un procedeu eficient de determinare a poziţiilor erorilor pentru codurile BCH binare a fost realizat de Chien.

Fie polinomul (de localizare a erorii):

$$S(X) = (1 - X_1X)(1 - X_2X)\ldots(1 - X_tX) = \sigma_0 + \sigma_1 X + \sigma_2 X^2 + \ldots + \sigma_t X^t$$

unde

$$\sigma_0 = 1$$
$$\sigma_1 = X_1 + X_2 + \ldots + X_t$$
$$\ldots$$
$$\sigma_t = X_1 X_2 \ldots X_t$$

(operaţiile fiind efectuate în binar, semnul $-$ este asimilat cu $+$).

Rădăcinile acestui polinom sunt inversele localizărilor erorilor X_i^{-1} $(1 \leq i \leq t)$.

Procedeul lui Chien de corectare a erorilor – pentru codurile BCH binare – este următorul:

Fie $\mathbf{r} = r_0 r_1 \ldots r_{n-1}$ cuvântul recepţionat; componentele lui se vor decodifica pas - cu - pas (pe măsură ce sunt recepţionate).

Să considerăm primul bit primit r_{n-1} şi să cercetăm dacă el este corect sau nu. Aceasta este echivalent cu a verifica dacă α^{n-1} este o localizare a erorii, sau dacă $\alpha^{-(n-1)} = \alpha$ este o rădăcină a polinomului $S(X)$ de localizare a erorii.

Dacă $\sigma_1\alpha + \sigma_2\alpha^2 + \ldots + \sigma_t\alpha^t = \mathbf{1}$ adică $S(\alpha) = \mathbf{0}$, atunci a intervenit o eroare pe poziţia α^{n-1} şi deci componenta r_{n-1} trebuie corectată.

Dacă $\sigma_1\alpha + \sigma_2\alpha^2 + \ldots + \sigma_t\alpha^t \neq \mathbf{1}$ adică $S(\alpha) \neq \mathbf{0}$, atunci r_{n-1} s-a recepţionat corect.

Procedeul se repetă: în general componenta recepţionată r_{n-s} este corectă dacă $\alpha^{-(n-s)} = \alpha^s$ nu este rădăcină a polinomului $S(X)$, adică

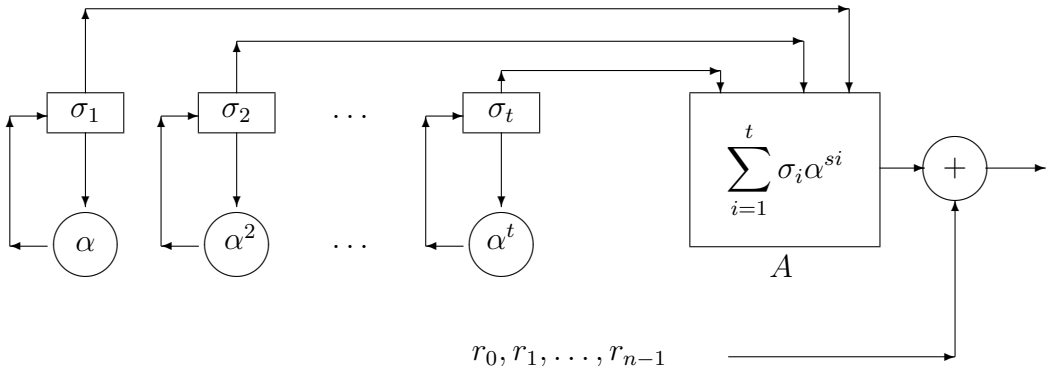
$$\sigma_1\alpha^s + \sigma_2\alpha^{2s} + \ldots + \sigma_t\alpha^{ts} \neq \mathbf{1}.$$

În caz contrar, r_{n-s} va fi corectată în $r_{n-s} + 1$ $(mod\ 2)$.

Algoritmul lui Chien este deci:

1. $s := 0$;
2. **while** $s \leq n$ **do**
   (a) $s := s + 1$;
   (b) Se calculează $P := \sigma_1\alpha^s + \sigma_2\alpha^{2s} + \ldots + \sigma_t\alpha^{ts}$;
   (c) Se decodifică $a_{n-s} := r_{n-s} + v$ unde $v = \begin{cases} 0 & \text{dacă } P \neq \mathbf{1} \\ 1 & \text{dacă } P = \mathbf{1} \end{cases}$

Circuitul liniar care implementează acest algoritm este:

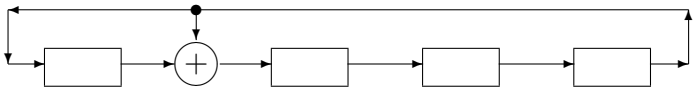


Cele t circuite din stânga realizează la fiecare tact multiplicarea cu α^i.

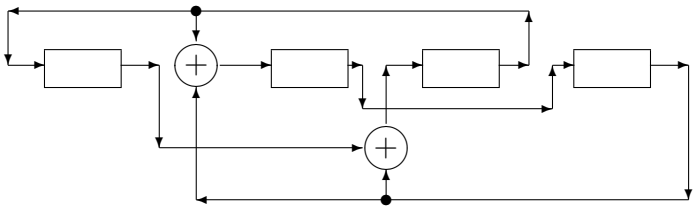
La momentul iniţial blocul A calculează suma $\sigma_1\alpha + \sigma_2\alpha^2 + \ldots + \sigma_t\alpha^t$ dând la ieşire valoarea 1 (scalar) dacă această sumă este egală cu $\mathbf{1} = (1, 0, \ldots, 0)$, şi 0 (scalar) dacă suma anterioară dă orice element din $GF(2^m)$ diferit de $\mathbf{1}$.
În funcţie de acest rezultat, componenta r_{n-1} este schimbată, respectiv păstrată.

La momentul următor, blocul A calculează suma $\sigma_1\alpha^2 + \sigma_2\alpha^4 + \ldots + \sigma_t\alpha^{2t}$. Dacă această sumă este $\mathbf{1}$, α^2 este rădăcină a polinomului de localizare a erorii $S(X)$, deci $\alpha^{-2} = \alpha^{n-2}$ este o localizare a erorii. În caz contrar, suma este diferită de $\mathbf{1}$. La ieşirea din blocul A se obţine scalarul 1 respectiv 0, componenta r_{n-2} fiind modificată respectiv păstrată.

Exemplul 12.5 *Fie $\alpha \in GF(2^4)$ primitiv, rădăcina polinomului $1 + X + X^4$. Să presupunem că se ştiu σ_1 şi σ_2. Circuitul care multiplică cu α este (Prelegerea 8, Figura 8.4) circuitul de împărţire la $m(X) = 1 + X + X^4$:*

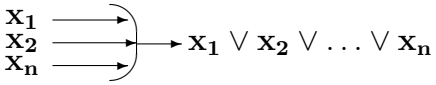


Circuitul de înmulţire cu α^2 este – conform egalităţii
$\alpha^2(a_1+a_2\alpha+a_3\alpha^2+a_4\alpha^3) = a_1\alpha^2+a_2\alpha^3+a_3\alpha^4+a_4\alpha^5 = a_3+(a_3+a_4)\alpha+(a_1+a_4)\alpha^2+a_2\alpha^3$
următorul (vezi şi Exemplul 8.10, Prelegerea 8):



Iniţial în elementele de înmagazinare ale celor două circuite se află σ_1 respectiv σ_2.

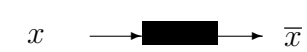
Pentru cazul binar vom nota cu



poarta OR care dă valoarea (scalar) 0 dacă şi numai dacă

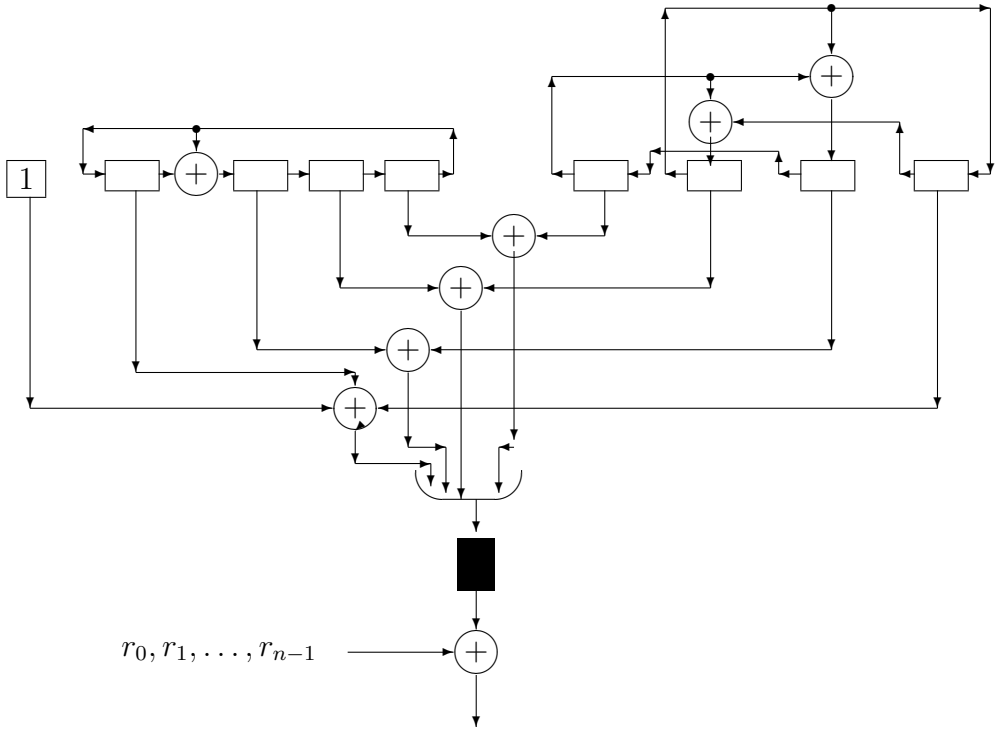
$$\mathbf{x_1} = \mathbf{x_2} = \ldots = \mathbf{x_n} = \mathbf{0},$$

şi cu



paorta NOT.

Atunci, circuitul de implementare a procedeului Chien de localizare a erorii este:



În acest circuit, cele patru elemente de înmagazinare din stânga formează circuitul de înmulţire cu α (unde iniţial se introduce σ_1), iar cele patru elemente din dreapta formează circuitul de înmulţire cu α^2, aranjat în ordine inversă (unde σ_2 se introduce iniţial de la dreapta spre stânga).

De asemenea, există un element de înmagazinare care conţine bitul 1, *element pe care îl furnizează la fiecare tact.*

Iniţial, circuitul funcţionează odată la momentul 0 *fără a introduce nimic de pe canal (pentru "amorsarea" în elementele de înmagazinare a coeficienţilor pentru $\sigma_1\alpha$ respectiv $\sigma_2\alpha^2$).*

La momentul următor se începe procesul efectiv de decodificare. Dacă r_{n-1} a fost perturbat, atunci α^{n-1} este localizarea erorii, adică α este rădăcină a polinomului de localizare a erorii şi avem $\mathbf{1} + \sigma_1\alpha + \sigma_2\alpha^2 = \mathbf{0}$. În acest caz, la intrarea în "sumatorul" OR este vectorul $\mathbf{0} = (0, 0, 0, 0)$, iar la ieşire scalarul 0. *După negare se obţine scalarul* 1, *care se adună (pentru corecţie) la r_{n-1}.*

Dacă r_{n-1} este corect, la intrarea în sumator se obţine o valoare nenulă, deci ieşirea va fi 1, *care prin negare este* 0 *şi r_{n-1} rămâne nemodificat.*

12.4 Exerciţii

12.1 *Să se verifice valabilitatea descompunerii $M = ABA^T$ din demonstraţia Teoremei 20.2*

12.2 *Fie $\alpha \in GF(2^m)$ $(m > 2)$ primitiv. Să se arate că polinoamele minimale $m_1(X)$ şi $m_3(X)$ au acelaşi grad.*

12.3 *Construiţi un cod BCH binar corector de* 2 *erori de lungime $n = 11$.*

12.4 *Construiţi un cod BCH ternar de lungime $n = 26$ şi $d = 5$.*

12.5 *Să se determine polinomul generator şi cel de control pentru codul BCH binar corector de* 3 *erori de lungime*

$(a)\ n = 15 \qquad (b)\ n = 31.$

Codificaţi mesajul de informaţie $111\ldots1$.

12.6 *Caracterizaţi codurile BCH care conţin toate elementele extensiei Galois* $GF(q^m)$.

12.7 *Dacă* $g(X) = 1 + X^4 + X^6 + X^7 + X^8$ *este polinomul generator al unui cod BCH binar de lungime* 15*, să se cerceteze dacă următoarele secvenţe sunt cuvinte - cod:*

(a) 011001011000010, (b) 000111010000110,
(c) 011100000010001, (d) 111111111111111.

12.8 *Aflaţi rădăcinile polinoamelor cu coeficienţi în* $GF(2^4)$ *(*α *este rădăcina polinomului* $1 + X + X^4$*):*

$X^2 + \alpha^4 X + \alpha^{13}$, $X^2 + \alpha^7 X + \alpha^2$, $X^2\alpha^2 X + \alpha^5$,
$X^2 + \alpha^6$, $X^2 + \alpha^2 X$, $X^2 + X + \alpha^8$.

12.9 *Definim codul BCH de polinom generator* $g(X) = m_1(X)m_3(X)$ *peste* $GF(2^4)$ *folosind elementul primitiv* α*, rădăcină a polinomului* $1 + X + X^4$*. Se recepţionează cuvintele* $\mathbf{r}$ *şi se calculează sindromurile* $\mathbf{r}H$*. Ştiind aceste sindromuri, localizaţi erorile (dacă se poate):*

$a)$ 01000101, $b)$ 11101000, $c)$ 11001101, $d)$ 01000000,
$e)$ 00000100, $f)$ 10100100, $g)$ 00111101, $h)$ 00000000.

12.10 *Pentru codul BCH definit în exerciţiul anterior, decodificaţi mesajele:*

$a)$ 110000000000000, $b)$ 000010000100001, $c)$ 010001010100000,
$d)$ 110011100111000, $e)$ 110011100100000, $f)$ 111000000000001,
$g)$ 101110000000000, $h)$ 101010010110001, $i)$ 010000100000000,
$j)$ 010101001011000, $k)$ 110111011101100, $m)$ 101110000001000,
$n)$ 111001011000000, $p)$ 000111010000110.

12.11 *În codul BCH binar de lungime* $n = 15$ *corector de* 3 *erori (vezi Exemplele* 11.1 *şi* 11.8*), să se decodifice mesajele:*

(a) 001000100000000, (b) 101011001000000.

12.12 *În codul BCH peste* Z_3 *de lungime* $n = 8$*, corector de* 2 *erori, să se decodifice mesajele*

(a) 00100000 (b) 12121212 (c) 11211122
(d) 02020202 (e) 10201020 (f) 22110011.

12.13 *Fie* $\alpha \in GF(2^5)$ *primitiv, rădăcină a polinomului* $1 + X^2 + X^5$*. Se defineşte un cod BCH binar de lungime* $n = 31$ *şi* $d = 5$*.*

1. *Să se construiască polinomul generator al codului:* $g(X) = m_1(X)m_3(X)$.
2. *Să se decodifice mesajul* 10010110111100001101010101111111.

Capitolul 13

Coduri Reed - Solomon

13.1 Definirea codurilor RS

O clasă foarte interesantă de coduri ciclice a fost definită în 1960 de Reed şi Solomon. Numite în articolul iniţial *coduri polinomiale*, Peterson arată un an mai târziu că ele sunt de fapt un caz particular de coduri BCH.

Construcţ ia dată de Reed - Solomon a fost următoarea:

Fie $m \geq 2$ şi $\alpha \in GF(q^m)$ primitiv (deci $ord(\alpha) = n = q^m - 1$).

Dacă $\mathbf{a_0 a_1} \ldots \mathbf{a_{k-1}} \in GF(q^m)^k$ este secvenţa de informaţie, se defineşte polinomul $h(X) = \mathbf{a_0} + \mathbf{a_1}X + \ldots + \mathbf{a_{k-1}}X^{k-1} \in GF(q^m)[X]$. Codul polinomial va fi format de toate cuvintele de forma $h(\mathbf{1})h(\alpha)\ldots h(\alpha^{n-1})$.

Teorema 13.1 *Un cod polinomial este un cod BCH peste $GF(q^m)$, definit de rădăcinile $\alpha, \alpha^2, \ldots, \alpha^{n-k}$.*

Demonstraţie: În algebra polinoamelor modulo $X^n - 1$ ($n = q^m - 1$) să considerăm clasa $\{f(X)\}$ asociată cuvântului - cod $h(q)h(\alpha)\ldots h(\alpha^{n-1})$:
$f(X) = h(\mathbf{1}) + h(\alpha)X + \ldots + h(\alpha^{n-1})X^{n-1} = \mathbf{a_0}(1 + X + X^2 + \ldots + X^{n-1}) +$
$\mathbf{a_1}(1 + \alpha X + \alpha^2 X^2 + \ldots + \alpha^{n-1}X^{n-1}) + \mathbf{a_2}(1 + \alpha^2 X + \alpha^4 X^2 + \ldots + (\alpha^2 X)^{n-1}) + \ldots +$
$\mathbf{a_{k-1}}(1 + \alpha^{k-1}X + \ldots + (\alpha^{k-1}X)^{n-1})$.

Se ştie că toate elementele nenule din $GF(q^m)$ sunt rădăcinile polinomului
$X^{q^m-1} - 1 = X^n - 1 = (X-1)(1 + X + X^2 + \ldots + X^{n-1})$, deci toate elementele din $GF(q^m) \setminus \{\mathbf{0}, \mathbf{1}\}$[1] sunt rădăcinile polinomului $1 + X + X^2 + \ldots + X^{n-1}$.

Prin calcul se obţine $f(\mathbf{1}) = \mathbf{a_0}$, $f(\alpha^{-1}) = -\mathbf{a_1}$, $f(\alpha^{-2}) = -\mathbf{a_2}, \ldots, f(\alpha^{-(k-1)}) = -\mathbf{a_{k-1}}$ şi $f(\alpha^{-j}) = \mathbf{0}$ dacă $n - 1 \geq j \geq k$.

Dar cum $\alpha^n = 1$, avem $\alpha^{-j} = \alpha^{n-j}$ şi deci $f(\alpha^i) = \mathbf{0}$, $\quad (1 \leq i \leq n-k)$. q.e.d.

În continuare vom numi aceste coduri *Reed - Solomon* pentru care vom folosi următoarea definiţie:

Definiţia 13.1 *Fie $\alpha \in GF(q^m)$ primitiv. Se numeşte cod Reed - Solomon (RS) codul BCH de polinom generator $g(X) \in GF(q^m)[X]$ definit prin*

$$g(X) = (X - \alpha^{m_0})(X - \alpha^{m_0+1})\ldots(X - \alpha^{m_0+d-2}).$$

[1] Reamintim, $\mathbf{1} = (1, 0, \ldots, 0)$, $\mathbf{0} = (0, \ldots, 0)$.

Deci, diferenţa faţă de codurile BCH generale este aceea că un cod RS este definit direct pe extensie, nu pe corpul de bază. Astfel, la un cod BCH cuvintele - cod sunt din Z_q^n, pe când la un cod RS, ele sunt din $GF(q^m)^n$. Cum şi $GF(q^m)$ este corp, toate proprietăţile codurilor BCH se păstrează şi în cazul codurilor RS.

Deoarece au fost studiate (şi utilizate) numai codurile Reed - Solomon de caracteristică 2, vom considera în continuare $q = 2$.

Exemplul 13.1 *Fie $GF(2^3)$ generat de rădăcina α a polinomului $1 + X + X^3$, şi să considerăm codul RS definit de polinomul $g(X) = (\alpha + X)(\alpha^2 + X) = \alpha^3 + \alpha^4 X + X^2$. Este un cod ciclic de lungime $n = 7$, $k = 5$ şi $d = 3$. Folosind definiţia din Prelegerea 9.3, se poate construi matricea generatoare a codului:*

$$G = \begin{pmatrix} \alpha^3 & \alpha^4 & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \alpha^3 & \alpha^4 & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \alpha^3 & \alpha^4 & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \alpha^3 & \alpha^4 & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \alpha^3 & \alpha^4 & \mathbf{1} \end{pmatrix}$$

Codul are 8^5 cuvinte, obţinute prin înmulţirea cu $g(X)$ a tuturor polinoamelor din $GF(2^3)[X]$ de grad cel mult 4 sau – echivalent – din înmulţirea cu matricea G a tuturor cuvintelor din $GF(2^3)^5$.

De exemplu, codificarea mesajului de informaţie $\mathbf{a} = \mathbf{1}\alpha\mathbf{00}\alpha^3$, sau $a(X) = \mathbf{1} + \alpha X + \alpha^3 X^4$ este $a(X)g(X) = \mathbf{1} + \alpha + (\alpha + \alpha^2)X^2 + \alpha X^3 + (1 + \alpha^2)X^4 + X^5 + (1 + \alpha)X^6 = \alpha^3 + \alpha^4 X^2 + \alpha X^3 + \alpha^6 X^4 + X^5 + \alpha^3 X^6$ (respectiv $\alpha^3\mathbf{0}\alpha^4\alpha\alpha^6\mathbf{1}\alpha^3$).

O matrice de control foarte simplă se obţine folosind (conform Prelegerii 11.3) rădăcinile α şi α^2 ale cuvintelor - cod (reamintim, $\alpha^7 = 1$):

$$H = \begin{pmatrix} \mathbf{1} & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \mathbf{1} & \alpha^2 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 & \alpha^5 \end{pmatrix}.$$

Teorema 13.2 *Distanţa minimă a unui cod RS este $d = n - k + 1$.*

Demonstraţie: Să notăm – temporar – cu d_H distanţa minimă a unui cod. Conform Teoremei 3.3 (Prelegrea 3), $d_H \leq n-k+1$. Pe de-altă parte, într-un cod RS, $grad(g(X)) = n-k = d-1$ deci $d = n-k+1$. Din cele două afirmaţii rezultă $d_H \leq d$. Pe de-altă parte, din Teorema 12.2, cum codul RS este un caz particular de cod BCH, rezultă $d_H \geq d$.q.e.d.

Din această teoremă rezultă că un cod RS are *distanţa maximă separabilă* (se mai spune că este un cod DMS).

13.2 Coduri RS scurte

Deoarece $\alpha \in GF(2^m)$ este primitiv, toate cuvintele unui cod RS au lungimea $n = 2^m - 1$ şi $k = n - d + 1 = 2^m - d$ simboluri de informaţie. Uneori însă sunt necesare coduri RS ale căror caracteristici să nu fie de această formă. Ele se obţin astfel:

Fie C un (n, k) - cod RS de distanţă minimă d şi s $(1 \leq s < 2^m - d)$ un număr întreg. Se construieşte codul RS *scurt* $C(s)$ format din toate cuvintele lui C care au $\mathbf{0}$ pe ultimele s poziţii, după care apoi aceste s poziţii se ignoră.

Exemplul 13.2 *Fie codul C dat de $g(X) = \alpha + X$, unde α este rădăcina polinomului $1 + X + X^2$ şi generează $GF(2^2)$. C are deci $n = 3,\ k = 2,\ d = 2$ şi 16 cuvinte - cod.*

Codul $C(1)$ se obţine luând toate cuvintele lui C care au $\mathbf{0}$ pe ultima poziţie (acestea sunt $\mathbf{000}, \alpha\mathbf{10}, \alpha^2\alpha\mathbf{0}, \mathbf{1}\alpha\mathbf{0}$), din care se elimină ultimul caracter.

Deci $C(1) = \{\mathbf{00}, \alpha\mathbf{1}, \alpha^2\alpha, \mathbf{1}\alpha\}$.

Atenţie*: Ultima poziţie nu înseamnă ultimul bit. Fiecare element din cuvânt are aici două poziţii binare. Astfel, cele patru cuvinte - cod alese din C au reprezentarea binară*

$$000000,\ 011000,\ 110100,\ 100100$$

iar codul scurt, reprezentat în binar este $\hat{C}(1) = \{0000, 0110, 1101, 1001\}$.

Din definiţie,

$$C(s) = \{\mathbf{b} \in C \mid grad(b(X)) < n - s\}.$$

Dacă $g(X)$ este polinomul generator al codului Reed - Solomon C, atunci $C(s)$ conţine toate polinoamele de forma $b(X)$ unde $b(X) = a(X)g(X),\ grad(a(X)) < k-s = 2^m-d-s$ (deoarece $grad(g(X)) = d - 1 = n - k$).

De aici rezultă că matricea generatoare a unui cod RS scurt $C(s)$ este

$$G = \begin{pmatrix} g(X) \\ Xg(X) \\ \vdots \\ X^{k-s-1}g(X) \end{pmatrix}$$

care se obţine din matricea generatoare a codului C, prin eliminarea ultimelor s linii. Numărul de simboluri de informaţie este egal cu numărul de linii din G, deci

$$k_s = k - s = 2^m - d - s.$$

Să notăm acum cu d respectiv d_s distanţele minime ale celor două coduri. Se observă că $\forall \mathbf{c_1}, \mathbf{c_2} \in C(s),\quad \mathbf{c'_1} = \mathbf{c_1}\mathbf{00}\ldots\mathbf{0},\ \mathbf{c'_2} = \mathbf{c_2}\mathbf{00}\ldots\mathbf{0} \in C$ şi deci $d(\mathbf{c_1}, \mathbf{c_2}) = d(\mathbf{c'_1}, \mathbf{c'_2})$, adică $d_s \geq d$.

Pe de-altă parte (Teorema 3.3, Prelegerea 3),

$d_s \leq n_s - k_s + 1 = 2^m - 1 - s - (2^m - d - s) + 1 = d$, deci $d_s = d$.

Din cele arătate mai sus putem enunţa teorema:

Teorema 13.3 *Fie C un (n, k) - cod RS cu distanţa minimă d şi $C(s)$ codul său scurt, cu parametrii n_s, k_s, d_s. Atunci*

$$n_s = 2^m - 1 - s, \quad k_s = 2^m - d - s, \quad d_s = d$$

şi $C(s)$ este un cod DMS.

Remarca 13.1 *Alte coduri scurte se pot obţine eliminând orice combinaţie fixă de s elemente din cuvintele unui cod RS. Se poate vedea imediat că proprietăţile arătate mai sus nu se schimbă.*

Exemplul 13.3 *Plecând de la codul RS din Exemplul 13.1, codul scurt $C(2)$ va avea ca matrice generatoare matricea G fără ultimele două linii şi coloane, iar ca parametri $n_2 = 5,\ k_2 = 3,\ d_2 = 3$. Matricea sa generatoare este*

$$G = \begin{pmatrix} \alpha^3 & \alpha^4 & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \alpha^3 & \alpha^4 & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \alpha^3 & \alpha^4 & \mathbf{1} \end{pmatrix}$$

13.3 Decodificarea codurilor RS

Fiind cazuri particulare de coduri BCH, codurile RS se pot decodifica folosind direct algoritmul Peterson (Prelegerea 12). Să reluăm detaliat pe un exemplu aplicarea acestui algoritm.

Exemplul 13.4 *Fie* $g(X) = (1+X)(\alpha+X)(\alpha^2+X)(\alpha^3+X)$ *unde* $\alpha \in GF(2^3)$ *este rădăcina (primitivă) a polinomului* $1+X+X^3$. *Codul are distanţa minimă* $d = 5$, *deci poate corecta maxim* 2 *erori independente. Mai ştim că:*

$$\alpha^3 = 1+\alpha, \quad \alpha^4 = \alpha+\alpha^2, \quad \alpha^5 = 1+\alpha+\alpha^2, \quad \alpha^6 = 1+\alpha^2.$$

Să presupunem că s-a recepţionat cuvântul $\mathbf{r} = \alpha^6\alpha\alpha^5\alpha^2\mathbf{10}\alpha^2$. *Pentru decodificare se aplică următorii paşi:*

(1)*: Se calculează cele patru sindromuri:*

$S_0 = v(\alpha^0) = \alpha^6+\alpha+\alpha^5+\alpha^2+\mathbf{1}+\mathbf{0}+\alpha^2 = \mathbf{1}+\alpha^2+\alpha+\mathbf{1}+\alpha+\alpha^2+\alpha^2+\mathbf{1}+\alpha^2 = \mathbf{1}$;
$S_1 = v(\alpha) = \alpha^6+\alpha^2+\alpha^7+\alpha^5+\alpha^4+\mathbf{0}+\alpha^8 = \alpha^3$;
$S_2 = v(\alpha^2) = \alpha^6+\alpha^3+\alpha^9+\alpha^8+\alpha^8+\mathbf{0}+\alpha^{14} = \alpha^3$;
$S_3 = v(\alpha^3) = \alpha^6+\alpha^4+\alpha^{11}+\alpha^{11}\alpha^{12}+\mathbf{0}+\alpha^{20} = \mathbf{1}$.

(2) *Deoarece rangul matricii* $\begin{pmatrix} S_0 & S_1 \\ S_1 & S_2 \end{pmatrix} = \begin{pmatrix} \mathbf{1} & \alpha^3 \\ \alpha^3 & \alpha^3 \end{pmatrix}$ *este* 2, *cuvântul* $\mathbf{r}$ *are două erori; cum distanţa minimă este* 5, *ele pot fi corectate.*

(3) *Se rezolvă sistemul liniar*

$$\begin{cases} S_0\sigma_2 + S_1\sigma_1 = S_2 \\ S_1\sigma_2 + S_2\sigma_1 = S_3 \end{cases} \text{ adică } \begin{cases} \sigma_2 + \alpha^3\sigma_1 = \alpha^3 \\ \alpha^3\sigma_2 + \alpha^3\sigma_1 = \mathbf{1} \end{cases}$$

Adunând cele două ecuaţii, se obţine $(\mathbf{1}+\alpha^3)\sigma_2 = \mathbf{1}+\alpha^3$, *deci* $\sigma_2 = \mathbf{1}$.

Înlocuind în prima ecuaţie, avem $\alpha^3\sigma_1 = \alpha^3+\mathbf{1} = \alpha$, *de unde, prin înmulţire cu* α^4 *se ajunge la* $\sigma_1 = \alpha^5$.

(4) *Polinomul de localizare a erorii este* $\sigma(X) = X^2+\sigma_1 X+\sigma_2 = X^2+\alpha^5 X+\mathbf{1}$. *Prin încercări se obţine* $\sigma(\alpha) = \sigma(\alpha^6) = \mathbf{0}$, *deci cele două erori sunt pe poziţiile* $X_1 = \alpha$, $X_2 = \alpha^6$.

(5) *Se rezolvă sistemul liniar* $\begin{pmatrix} X_1^0 & X_2^0 \\ X_1 & X_2 \end{pmatrix} \cdot \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} = \begin{pmatrix} S_0 \\ S_1 \end{pmatrix}$, *adică:*

$$\begin{cases} Y_1 + Y_2 = \mathbf{1} \\ \alpha Y_1 + \alpha^6 Y_2 = \alpha^3 \end{cases}$$

Înmulţind a doua ecuaţie cu α *şi adunând la prima, se obţine* $\alpha^6 Y_1 = \alpha^5$, *deci* $Y_1 = \alpha^6$. *După înlocuire în prima ecuaţie, se determină şi* $Y_2 = \alpha^2$.

(6) *Rezumând, au apărut erorile* α^6 *pe poziţia* 1 *şi* α^2 *pe poziţia* 6. *Deci eroarea este* $\mathbf{e} = \mathbf{0}\alpha^6\mathbf{0000}\alpha^2$, *iar cuvântul - cod trimis a fost*

$$\mathbf{c} = \mathbf{r}+\mathbf{e} = \alpha^6\alpha^5\alpha^5\alpha^2\mathbf{100}.$$

13.4 Altă construcţie a codurilor RS

Plecând de la construcţia iniţială, se poate realiza şi o altă definiţie pentru codurile RS. Ea se bazează pe o modalitate diferită de reprezentare a cuvintelor din Z_q^n, anume considerând polinoamele drept funcţii.

Definiţia 13.2 *Fie $S \subseteq GF(2^m)$. Două polinoame $p(X), q(X)$ reprezintă aceeaşi funcţie $: S \to GF(2^m)$ dacă şi numai dacă $\forall \beta \in S,\ p(\beta) = q(\beta)$.*

Exemplul 13.5 *Fie $S = GF(2^3)$, construit folosind rădăcina primitivă α a polinomului $1 + X + X^3$. Definim funcţia $f : S \to Z_2$ prin*

X	$\mathbf{0}$	$\mathbf{1}$	α	α^2	α^3	α^4	α^5	α^6
$f(X)$	0	0	1	1	0	1	0	0

Atunci $f(x)$ se poate reprezenta prin

$$\mathbf{v_f} = f(\mathbf{0})f(\mathbf{1})f(\alpha)f(\alpha^2)f(\alpha^3)f(\alpha^4)f(\alpha^5)f(\alpha^6) = 00110100.$$

Exemplul 13.6 *Fie $S = GF(2^3)$ şi funcţia $f : S \to S$ definită $\mathbf{v_f} = \alpha^4\mathbf{01}\alpha^2\mathbf{1}\alpha\mathbf{00}$.*

Considerând f drept polinom, folosim metoda coeficienţilor nedeterminaţi pentru a obţine reprezentarea

$$f(X) = \alpha^4 + \alpha^2 X + \alpha^3 X^2 + X^3.$$

Mulţimea $A = \{f(X) \in GF(2^m)[X] \mid grad(f(X)) \leq k-1\}$ se poate structura evident ca un spaţiu liniar de dimensiune k, cu baza $\{\mathbf{1}, X, X^2, \ldots, X^{k-1}\}$. Deoarece toate aceste polinoame pot fi considerate funcţii definite pe o anumită mulţime $S \subseteq GF(2^m)$, vom numi *A spaţiul funcţiilor pe S*.

De remarcat că în această fază S nu este definit explicit.

Teorema 13.4 *Fie mulţimea S, $card(S) = n$ şi $k \leq n$. Mulţimea tuturor funcţiilor $: S \to GF(2^m)$ reprezentate prin polinoame de grad $\leq k-1$ formează un spaţiu liniar de funcţii, cu baza $\{\mathbf{1}, X, \ldots, X^{k-1}\}$.*

Demonstraţie: Evident, orice polinom de grad cel mult $k-1$ este generat de $\{\mathbf{1}, X, \ldots, X^{k-1}\}$. Tot ce trebuie arătat este că fiecare funcţie $f : S \to GF(2^m)$ are o reprezentare unică sub formă de polinom.

Să presupunem (prin absurd) că $p(X)$ şi $q(X)$ sunt egale ca funcţii pe S. Deci $\forall \alpha \in S,\ p(\alpha) = q(\alpha)$. Dar atunci $p(X) - q(X)$ este un polinom de grad $< k$ care are n ($n \geq k$) rădăcini, deci $p(X) - q(X) \equiv 0$, adică $p(X) \equiv q(X)$. q.e.d.

Exemplul 13.7 *Fie $S = GF(2^3)$ construit folosind rădăcina α a ecuaţiei $1+X+X^3 = 0$, şi să considerăm toate polinoamele din $GF(2^3)[X]$, de grad cel mult doi. O bază pentru acest spaţiu de funcţii este $\{\mathbf{1}, X, X^2\}$, cu reprezentarea corespunzătoare:*

$$\begin{array}{lcl} \mathbf{1} & - & \mathbf{11111111} \\ X & - & \mathbf{01}\alpha\alpha^2\alpha^3\alpha^4\alpha^5\alpha^6 \\ X^2 & - & \mathbf{01}\alpha^2\alpha^4\alpha^6\alpha\alpha^3\alpha^5 \end{array}$$

Atunci, orice polinom $f(X) = \mathbf{a_0} + \mathbf{a_1}X + \mathbf{a_2}X^2 \in GF(2^3)[X]$*, considerat ca o funcţie, poate fi reprezentat sub formă de vector prin*

$$\mathbf{v_f} = (\mathbf{a_0}\ \mathbf{a_1}\ \mathbf{a_2}) \begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \mathbf{0} & \mathbf{1} & \alpha^2 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 & \alpha^5 \end{pmatrix}$$

Sunt $8^3 = 512$ *astfel de polinoame.*

Teorema 13.5 *Spaţiul funcţiilor pe* $S \subseteq GF(2^m)$ $(card(S) = n)$ *al tuturor polinoamelor din* $GF(2^m)[X]$ *de grad* $\leq k-1$ *formează un* (n, k) *- cod liniar DMS.*

Demonstraţie: Alegem o mulţime cu n elemente $S \subseteq GF(2^m)$ şi considerăm spaţiul liniar al funcţiilor tuturor polinoamelor $p : S \to GF(2^m)$ de grad cel mult $k-1$. Lungimea fiecărui cuvânt - cod este n, iar dimensiunea spaţiului este k (Teorema 20.1). Mai rămâne de arătat că acest cod este DMS (are distanţă maximă separabilă), adică $d = n - k + 1$. Pentru aceasta, să observăm că orice polinom $p(X)$ are maxim $k-1$ rădăcini distincte; deci reprezentarea $\mathbf{v_p}$ are cel mult $k-1$ zero-uri, adică $w(\mathbf{v_p}) \geq n - k + 1$. În particular, $d = min\{w(\mathbf{v_p})\} \geq n - k + 1$.

Pe de altă parte, conform Teoremei 3.3, $d \leq n - k + 1$, deci $d = n - k + 1$. q.e.d.

Fie n un divizor al lui $2^m - 1$ şi $S = \{\alpha \in GF(2^m) | \alpha^n = 1\}$ mulţimea rădăcinilor de ordinul n ale unităţii din $GF(2^m)$. Evident, S conţine toate rădăcinile din $GF(2^m)$ ale ecuaţiei $X^n + 1 = 0$ şi – fiind un corp finit – are cel puţin un element primitiv (Teorema 11.1).

Teorema 13.6 *Fie* $p(X), q(X) \in GF(2^m)[X]$ *şi* $S \subseteq GF(2^m)$ *mulţimea rădăcinilor de ordinul* n *ale unităţii. Atunci* $p(X)$ *şi* $q(X)$ *reprezintă aceeaşi funcţie* $f : S \to GF(2^m)$ *dacă şi numai dacă* $p(X) \equiv q(X)\ (mod\ X^n + 1)$.

Demonstraţie: "$\Longleftarrow$": Fie $q(X) = h(X)(X^n+1)+p(X)$. Atunci pentru orice $\alpha \in S$ avem $q(\alpha) = h(\alpha)(1 + \alpha^n) + p(\alpha)$. Dar $1 + \alpha^n = 0$, deci $q(\alpha) = p(\alpha)$.

"'$\Longrightarrow$: Fie $\alpha \in S$ primitiv. Deci orice element din S are forma α^i $(0 \leq i < n)$. Dacă $q(\alpha^i) = p(\alpha^i)$, atunci α^i $(0 \leq i \leq n)$ sunt rădăcini ale polinomului $p(X) - q(X)$. Putem scrie atunci

$$p(X) - q(X) = h(X) \prod_{i=0}^{n-1} (1 + \alpha^i) = h(X)(1 + X^n).$$

q.e.d.

Teorema 13.7 *Fie* S *mulţimea rădăcinilor de ordinul* n *ale unităt ii din* $GF(2^m)$*. Atunci spaţiul funcţiilor tuturor polinoamelor din* $GF(2^m)[X]$ *de grad* $\leq k-1$ *pe* S *formează un* (n, k) *- cod ciclic peste* $GF(2^m)$.

Demonstraţie: Să notăm cu C acest spaţiu şi fie S ca în ipoteză, generat de α primitiv. Atunci, dacă $p(X) \in GF(2^m)[X]$ este un polinom de grad cel mult $k-1$, avem $\mathbf{v_p} = p(\mathbf{1})p(\alpha)\dots p(\alpha^{n-1}) \in C$.

Fie $q(X) = p(\alpha X)$. Cum $grad(q(X)) \leq k-1$, rezultă $\mathbf{v_q} \in C$. Dar

$$q(\mathbf{1})q(\alpha)\dots q(\alpha^{n-1}) = p(\alpha)p(\alpha^2)\dots p(\alpha^{n-1})p(\mathbf{1}) \in C,$$

care ce reprezintă – conform definiţiei – un cod ciclic. q.e.d.

Exemplul 13.8 *Să considerăm $S = GF(2^3)$ spaţiul rădăcinilor de ordin 7 ale unităţii, generat de α, rădăcina ecuaţiei $1 + X + X^3 = 0$. Fie de exemplu $p(X) = \alpha^4 + \alpha^2 X + \alpha^3 X^2 + X^3 = (\mathbf{1} + X)(\alpha^5 + X)(\alpha^6 + X)$, care are reprezentarea $\mathbf{01}\alpha^2\mathbf{1}\alpha\mathbf{00}$.*

Permutarea ei ciclică este $\mathbf{1}\alpha^2\mathbf{1}\alpha\mathbf{000}$, care corespunde funcţiei $p(\alpha X) = \alpha^4 + \alpha^3 X + \alpha^5 X^2 + \alpha^3 X^3 = (\alpha^4 + X)(\alpha^5 + X)(\alpha^6 + X)\alpha^3$.

Definiţia 13.3 *Fie polinomul $V(X) = \mathbf{V_0} + \mathbf{V_1}X + \ldots + \mathbf{V_{n-1}}X^{n-1} \in GF(2^m)[X]$ şi $\alpha \in GF(2^m)$ o rădăcină primitivă de ordinul n a unităţii. Spunem că $v(X) = \mathbf{v_0} + \mathbf{v_1}X + \ldots + \mathbf{v_{n-1}}X^{n-1}$ este "transformata" lui $V(X)$ dacă*

$$V(\alpha^j) = \sum_{i=0}^{n-1} \mathbf{V_i}\alpha^{ji} = \mathbf{v_j}, \quad j = 0, 1, \ldots, n-1.$$

Altfel spus, dacă se notează cu $M_{n,n} = (\alpha^{ij})_{1 \le i,j \le n}$, atunci

$$(\mathbf{V_0}\ \mathbf{V_1}\ \ldots\ \mathbf{V_{n-1}})M = (\mathbf{v_0}\ \mathbf{v_1}\ \ldots\ \mathbf{v_{n-1}}).$$

Matricea M este numită *transformata Fourier finită*. Ea este nesingulară, deci se poate scrie $(\mathbf{V_0}\ \mathbf{V_1}\ \ldots\ \mathbf{V_{n-1}}) = (\mathbf{v_0}\ \mathbf{v_1}\ \ldots\ \mathbf{v_{n-1}})M^{-1}$ sau

$$\mathbf{V_i} = \sum_{i=1}^{n-1} \mathbf{v_j}\alpha^{-ij} = v(\alpha^{-i})$$

Vom demonstra aceasta formulă în mod direct, fără a folosi inversa matricii M:

Teorema 13.8 *Fie $\alpha \in GF(2^m)$ o rădăcină primitivă de ordin n a unităţii. Dacă $\mathbf{v_j} = V(\alpha^j)$ pentru $V(X) = \mathbf{V_0} + \mathbf{V_1}X + \ldots + \mathbf{V_{n-1}}X^{n-1} \in GF(2^m)[X]$, atunci $\mathbf{V_i} = v(\alpha^{-i})$ unde $v(X) = \mathbf{v_0} + \mathbf{v_1}X + \ldots + \mathbf{v_{n-1}}X^{n-1}$.*

Demonstraţie: Deoarece α este primitiv, vom avea

$$\sum_{j=0}^{n-1} \alpha^{(k-i)j} = \begin{cases} n\ (mod\ 2) & \text{pentru} \quad k = i \\ 0 & \text{pentru} \quad k \neq i \end{cases}$$

$$\text{Atunci } v(\alpha^{-i}) = \sum_{j=0}^{n-1} \mathbf{v_j}\alpha^{-ij} = \sum_{j=0}^{n-1} \left(\sum_{k=0}^{n-1} \mathbf{V_k}\alpha^{kj} \right) \alpha^{-ij} = \sum_{k=0}^{n-1} \mathbf{V_k} \left(\sum_{j=0}^{n-1} \alpha^{(k-i)j} \right) = \mathbf{V_i}.$$

q.e.d.

Deci, fiind dat un vector de valori $(\mathbf{v_0}, \mathbf{v_1}, \ldots, \mathbf{v_{n-1}})$, se pot regăsi coeficienţii polinomului $V(X) = \mathbf{V_0} + \mathbf{V_1}X + \ldots + \mathbf{V_{n-1}}X^{n-1}$, lucru esenţial în definirea unui algoritm de decodificare.

Teorema 13.9 *Fie $S \subseteq GF(2^m)$ mulţimea rădăcinilor de ordinul n ale unităţii din $GF(2^m)$, generat de rădăcina primitivă α. Spaţiul liniar al tuturor funcţiilor polinomiale de grad $< n - d + 1$ pe S este un cod ciclic DMS cu polinomul generator*

$$g(X) = (\alpha + X)(\alpha^2 + X)\ldots(\alpha^{d-1} + X).$$

Demonstraţie: Funcţia polinomială $V(X)$ a cărei transformată corespunde lui $v(X) = a(X)g(X)$ este $V(X) = \sum_{i=0}^{n-1} v(\alpha^{n-i})X^i$. Deoarece $g(\alpha^p) = 0$, $(1 \le p \le d-1)$, vom

avea $v(\alpha^{n-i}) = 0$ pentru $i = n-d+1, n-d+2, \ldots, n-1$. Rezultă că pentru orice i, $(n-d+1 \leq i \leq n-1)$ coeficientul lui X^i din $V(X)$ este zero, deci $V(X)$ are gradul $< n-d+1$. q.e.d.

Dacă se ia $n = 2^m - 1$, se ajunge la o altă modalitate de construcţie a codurilor RS, folosind o altă matrice generatoare (deci o altă dispunere a biţilor de informaţie).

Exemplul 13.9 *Fie α o rădăcină a polinomului $1+X+X^3$, cu care se construieşte $GF(2^3)$. Considerăm codul RS de polinom generator $g(X) = (\mathbf{1}+X)(\alpha+X)(\alpha^2+X)(\alpha^3+X) = \alpha^6+\alpha^5X+\alpha^5X^2+\alpha^2X^3+X^4$ (care corespunde reprezentării $\alpha^6\alpha^5\alpha^5\alpha^2\mathbf{100}$).*

Transformata lui $g(X)$ este polinomul $G(X) = \sum_{i=0}^{6} g(\alpha^{7-i})X^i$.

Pentru că $g(\alpha^0)g(\alpha)\ldots g(\alpha^6) = \mathbf{00001}\alpha\alpha^4$, avem

$$G(X) = g(\mathbf{1}) + g(\alpha^{7-1})X + g(\alpha^{7-2})X^2 + g(\alpha^{7-3})X^3 = \alpha^4X + \alpha X^2 + X^3.$$

Cum $d-1 = 3$, avem $d = 4$ deci $gr(G(X)) < n-d+1 = 4$.

Se verifică uşor că $G(X)$ este o funcţie polinomială cu reprezentarea

$$G(\alpha^0)G(\alpha)\ldots G(\alpha^6) = \alpha^6\alpha^5\alpha^5\alpha^2\mathbf{100}.$$

Putem gândi acest cod RS ca spaţiul tuturor polinoamelor de grade $1, 2$ şi 3, adică

$$\{Xp(X) \mid p(X) \in GF(2^3)[X],\ grad(p(X)) < 3\}.$$

Baza pentru acest spaţiu liniar este $\{X, X^2, X^3\}$, iar matricea generatoare

$$\begin{pmatrix} \mathbf{1} & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \mathbf{1} & \alpha^2 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 & \alpha^5 \\ \mathbf{1} & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha & \alpha^4 \end{pmatrix}$$

Deci $G(X) = \alpha^4X + \alpha X^2 + X^3$ va avea reprezentarea

$$(\alpha^4\ \alpha\ \mathbf{1}) \begin{pmatrix} \mathbf{1} & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \mathbf{1} & \alpha^2 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 & \alpha^5 \\ \mathbf{1} & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha & \alpha^4 \end{pmatrix} = (\alpha^6\ \alpha^5\ \alpha^5\ \alpha^2\ \mathbf{1}\ \mathbf{0}\ \mathbf{0}).$$

13.4.1 Algoritmul de decodificare Berlekamp - Massey

Fie $g(X)$ polinomul generator al unui cod RS, $c(X) = a(X)g(X)$ un polinom - cod transmis, şi $v(X) = c(X) + e(X)$ polinomul recepţionat. Notăm $V(X), C(X)$ şi $E(X)$ transformatele lui $v(X), c(X)$ respectiv $e(X)$. Cum transformata Fourier finită este o aplicaţie liniară, se verifică imediat $V(X) = C(X) + E(X)$.

Deoarece $g(X)$ are $d-1$ rădăcini consecutive α^i, $(m_0 \leq i \leq m_0+d-2)$, sindromul polinomial $s(X)$ va avea $s(\alpha^{n-i}) = e(\alpha^{n-i}) = E_i$ pentru $i = n-m_0-d+1, \ldots, n-m_0$. Deci sindromul va putea determina $d-1$ coeficienţi ai transformatei $E(X)$.

Pentru ceilalţi coeficienţi va trebui să reluăm polinomul de localizare a erorilor $\sigma(X)$ din algoritmul Peterson de decodificare a codurilor BCH.

$\sigma(X)$ are proprietatea că $\sigma(\alpha^k) = 0 \Longleftrightarrow e_k \neq 0$.

Deoarece $E(\alpha^k) = e_k$, vom avea $\sigma(\alpha^k)E(\alpha^k) = 0\ \forall k$, deci – lucrând în algebra polinoamelor modulo $1+X^n$:

$$\{\sigma(X)\}\{E(X)\} = 0$$

şi

$$\{\sigma(X)\}\{E(X)\} = \left\{ \left(\sum_{i=0}^{t} \sigma_i X^i \right) \left(\sum_{j=0}^{n-1} \mathbf{E_j} X^j \right) \right\},$$

unde $t = [(d-1)/2]$. Calculând în ambele relaţii coeficientul lui X^{t+j}, se obţine:

$$\sigma_t \mathbf{E_j} + \sigma_{t-1} \mathbf{E_{j+1}} + \ldots + \sigma_0 \mathbf{E_{j+t}} = 0.$$

Pentru că ştim deja $d-1$ valori consecutive ale lui $\mathbf{E_j}$, putem determina recursiv coeficienţii σ_i, pe care îi folosim la generarea tuturor valorilor $\mathbf{E_j}$.

În [7] se defineşte o altă formă a acestui algoritm.

Exemplul 13.10 *Să reluăm codul RS definit în Exemplul 14.4, în care considerăm că s-a recepţionat cuvântul* $\mathbf{v} = \alpha^6\alpha\alpha^5\alpha^2\mathbf{10}\alpha^2$.

Deoarece $d = 5$, *avem* $t \leq 2$ *şi*

$$\mathbf{E_0} = v(\alpha^0) = \mathbf{1},\ \mathbf{E_6} = v(\alpha) = \alpha^3,\ \mathbf{E_5} = v(\alpha^2) = \alpha^3,\ \mathbf{E_4} = v(\alpha^3) = \mathbf{1}.$$

Polinomul de localizare a erorii este $\sigma(X) = X^2 + \sigma_1 X + \sigma_0$.

Determinăm σ_0, σ_1 *ca în Exemplul 14.4 şi găsim* $\sigma_0 = \mathbf{1},\ \sigma_1 = \alpha^5$. *Se poate determina astfel relaţia de recurenţă:*

$$\mathbf{E_k} = \alpha^5 \mathbf{E_{k+1}} + \mathbf{E_{k+2}}.$$

Deoarece $(\mathbf{E_0}, \mathbf{E_6}, \mathbf{E_5}, \mathbf{E_4}) = (\mathbf{1}, \alpha^3, \alpha^3, \mathbf{1})$, *vom avea:*

$$\mathbf{E_3} = \alpha^5\mathbf{E_4} + \mathbf{E_5} = \alpha^5 + \alpha^3 = \alpha^2,$$
$$\mathbf{E_2} = \alpha^5\mathbf{E_3} + \mathbf{E_4} = \alpha^5 \cdot \alpha^2 + \mathbf{1} = \mathbf{0},$$
$$\mathbf{E_1} = \alpha^5\mathbf{E_2} + \mathbf{E_3} = \mathbf{0} + \alpha^2 = \alpha^2.$$

Am determinat transformata lui $e(X)$, *adică* $E(X) = \mathbf{1} + \alpha^2 X + \alpha^2 X^3 + X^4 + \alpha^3 X^5 + \alpha^3 X^6$. *De aici rezultă* $\mathbf{e} = (E(\alpha^0), E(\alpha), E(\alpha^2), \ldots, E(\alpha^6)) = \mathbf{0}\alpha^6\mathbf{0000}\alpha^2$, *deci cuvântul decodificat este*

$$\mathbf{c} = \mathbf{v} + \mathbf{e} = \alpha^6\alpha^5\alpha^5\alpha^2\mathbf{100}.$$

Pe de-altă parte, dacă s-ar fi folosit matricea generatoare din Exemplul 17.3, nu ar mai fi fost necesar să se determine vectorul eroare $\mathbf{e}$, *ci doar să se calculeze toate valorile lui* $v(\alpha^k)$, $(0 \leq k \leq 6)$, *din care se obţine direct mesajul transmis; mai precis, se află transformata lui* $v(X)$, *care se adună la transformata lui* $e(X)$*:*

$\mathbf{v_0v_1} \ldots \mathbf{v_6} = v(\alpha^0)v(\alpha^6)v(\alpha^5)\ldots v(\alpha) = \mathbf{1}\alpha\alpha\alpha^6\mathbf{1}\alpha^3\alpha^3$, $\quad \mathbf{E_0E_1} \ldots \mathbf{E_6} = \mathbf{1}\alpha^2\mathbf{0}\alpha^2\mathbf{1}\alpha^3\alpha^3$,

şi deci

$$\mathbf{C_0C_1} \ldots \mathbf{C_6} = \mathbf{V_0V_1} \ldots \mathbf{V_6} + \mathbf{E_0E_1} \ldots \mathbf{E_6} = \mathbf{0}\alpha^4\alpha\mathbf{1000}.$$

S-a obţinut $C(X) = \alpha^4 X + \alpha X^2 + X^3$. *Deci biţii de informaţie sunt* $(\alpha^4, \alpha, \mathbf{1})$.

De aici se obţine forma cuvântului-cod transmis: $\mathbf{c} = \alpha^6\alpha^5\alpha^5\alpha^2\mathbf{100}$.

13.5 Ştergeri

O *ştergere* este o eroare detectată, pentru corectarea ei trebuind determinată doar valoarea erorii. *Numărul locaţiei de ştergere* este poziţia elementului perturbat. Acest număr este în general cunoscut (din citirea fizică a mesajului primit sau din structura codului).

Astfel, fie C un cod RS peste $GF(2^m)$ şi să considerăm codul $\hat{C}$ format din reprezentarea binară a cuvintelor codului C (fiecare element din $GF(2^m)$ se scrie ca un cuvânt binar de lungime m). Se defineşte şi codul $\hat{C}'$ obţinut prin adăugarea câte unui bit de paritate la fiecare simbol din componenţa cuvintelor codului $\hat{C}$.

Exemplul 13.11 *Să considerăm codul RS din Exemplul 20.3. Pentru a forma* $\hat{C}'$, *se înlocuieşte fiecare element* $\mathbf{0}, \mathbf{1}, \alpha, \alpha^2$ *respectiv cu* $000, 101, 011, 110$ *(al treilea bit este bitul de paritate). Astfel, pentru* $\alpha\mathbf{10} \in A$ *avem* $011101000 \in \hat{C}'$.

Deoarece fiecare element dintr-un cuvânt - cod $\mathbf{c} \in C$ este reprezentat printr-un cuvânt binar de lungime $m+1$, cuvântul - cod corespunzător $\hat{\mathbf{c}}' \in \hat{C}'$ va avea lungimea $(2^m - 1)(m+1)$. Se observă că fiecare element nenul din $\mathbf{c}$ este înlocuit cu un cuvânt de pondere pară. Deci, $\hat{\mathbf{c}}'$ va fi format din $2^m - 1$ cuvinte binare de lungimi $m+1$, fiecare de pondere pară.

În acest fel, dacă în cuvântul recepţionat $\hat{\mathbf{v}}'$ există un grup de pondere impară, ştim că a apărut o eroare printre cei $m+1$ biţi corespunzători. Considerat din nou ca un cuvânt cu elemente din $GF(2^m)$, ştim că acest cuvânt are un element perturbat; nu cunoaştem valoarea erorii, dar ştim poziţia sa. Avem deci o "ştergere".

Exemplul 13.12 *Reluând construcţia din exemplul anterior, să presupunem că s-a primit* 011100000. *Deci, au apărut erori în al doilea grup de* 3 *biţi (acesta are pondere impară), aşa că* α^1 *este un numărul locaţiei de ştergere. Ştiind acest număr, putem înlocui al doilea element cu* $\mathbf{0}$ *(uşurează calculul sindromului), şi vom încerca decodificarea lui* $\mathbf{v} = \alpha\mathbf{00}$ *în cel mai apropiat cuvânt - cod din A, ştiind că poziţia erorii este* $X_1 = \alpha$.

Teorema 13.10 *Fie C un cod RS peste* $GF(2^m)$ *de distanţa minimă d, şi* $\mathbf{v}$ *un cuvânt recepţionat care are s ştergeri şi t erori care nu sunt ştergeri. Atunci* $\mathbf{v}$ *poate fi decodificat corect dacă* $2t + s \leq d - 1$.

Demonstraţie: De remarcat că în Teorema 3.7 (pentruiu coduri liniare), o condiţie mai slabă ($t + s \leq d - 1$) asigură corectarea celor t erori şi detectarea (doar) a s ştersături. Această teoremă realizează corectarea completă a cuvântului.

Fie B mulţimea poziţiilor ştersăturilor şi A mulţimea poziţiilor erorilor; deci $A \setminus B$ este mulţimea poziţiilor erorilor care nu sunt ştersături. Definim $\sigma_B(X) = \prod_{i \in B}(\alpha^i + X)$ ca fiind polinomul de localizare a ştersăturilor. Atunci polinomul de localizare a erorilor se poate scrie

$$\sigma_A(X) = \sigma_B(X)\sigma_{A\setminus B}(X).$$

Aflarea poziţiilor erorilor revine la determinarea rădăcinilor polinomului $\sigma_{A\setminus B}(X)$. Pentru aceasta se poate folosi algoritmul Peterson de decodificare a codurilor BCH (Prelegerea 12), cu câteva schimbări, corespunzătoare sindromurilor "modificate".

Astfel, fie
$\sigma_B(X) = B_0 + B_1X + \ldots + B_{s-1}X^{s-1} + X^s$ şi $\sigma_{A\setminus B}(X) = A_0 + A_1X + \ldots + A_{t-1}X^{t-1} + X^t$.

Similar algoritmului Peterson, vom înmulţi ambii membrii ai egalităţii
$\sigma_B(X)\sigma_{A\setminus B}(X) = \sigma_A(X)$ cu $Y_iX_i^j$ $(m_0 \leq j \leq m_0 + d - 2)$ (reamintim, X_i corespund locaţiilor erorilor, iar Y_i sunt valorile acestor erori).

Apoi se înlocuieşte $X = X_i$ şi se face suma după $i = 1, \ldots, t + s$. Se obţine

$$S_j'A_0 + S_{j+1}'A_1 + \ldots + S_{j+t-1}'A_{t-1} + S_{j+t}' = 0 \quad (m_0 \leq j \leq m_0 + d - 2) \tag{1}$$

unde S_k' sunt sindromurile modificate, de forma

$$S_k' = B_0S_k + B_1S_{k+1} + \ldots + B_{s-1}S_{k+s-1} + S_{k+s}. \tag{2}$$

Pentru că se ştiu valorile lui S_j pentru $j = m_0, \ldots, m_0 + d - 2$ iar $B_0, B_1, \ldots, B_{s-1}$ sunt cunoscute, se pot determina S_j' pentru $m_0 \leq j \leq d - 1 - s$. Deoarece $2t + s \leq d - 1$, deci $2t \leq d - 1 - s$, se poate rezolva sistemul (1) scris matricial

$$\begin{pmatrix} {S_{m_0}}' & {S_{m_0+1}}' & \dots & {S_{m_0+t-1}}' \\ {S_{m_0+1}}' & {S_{m_0+2}}' & \dots & {S_{m_0+t}}' \\ & & \vdots & \\ {S_{m_0+t-1}}' & {S_{m_0+t}}' & \dots & S_{m_0+2t-1} \end{pmatrix} \begin{pmatrix} A_0 \\ A_1 \\ \vdots \\ A_{t-1} \end{pmatrix} = \begin{pmatrix} {S_{m_0+t}}' \\ {S_{m_0+t+1}}' \\ \vdots \\ S_{m_0+2t} \end{pmatrix} \tag{3}$$

cu t necunoscute $A_0, A_1, \dots, A_{t-1}$.

După determinarea polinomului de localizare a erorilor, algoritmul Peterson se poate aplica în continuare nemodificat. q.e.d.

Fie C un cod RS peste $GF(2^m)$, de polinom generator $g(X) = (\alpha^{m_0} + X) \dots (\alpha^{m_0+d-2} + X)$. Se transmite cuvântul - cod $\mathbf{c}$ şi se recepţionează $\mathbf{v}$ cu s ştergeri localizate în elementele mulţimii $B = \{X_1, \dots, X_s\}$. Fie $\sigma_B(X) = B_0 + B_1X + \dots + B_{s-1}X^{s-1} + X^s$ polinomul de localizare al ştergerilor.

Polinomul de localizare a erorilor $\sigma_A(X) = \sigma_{A\setminus B}(X)\sigma_B(X)$ poate fi determinat aflând $\sigma_{A\setminus B}(X) = A_0 + A_1X + \dots + A_{t-1}X^{t-1} + X^t$ astfel:

(a) Se calculează $S_j = v(\alpha^j)$ pentru $j = m_0, m_0 + 1, \dots, m_0 + d - 2$;
(b) Se determină ${S_j}'(m_0 \le j \le m_0 + d - 2 - s)$ cu relaţiile (2);
(c) Se rezolvă sistemul liniar (3) de necunoscute A_i.

Exemplul 13.13 *Fie C codul RS peste $GF(2^4)$ construit de rădăcina α a ecuaţiei $1 + X + X^4 = 0$, definit cu polinomul generator $g(X) = (1 + X)(\alpha + X) \dots (\alpha^4 + X)$ şi $\hat{C}'$ codul binar asociat în care sunt codificate mesajele. S-a primit mesajul*

$$\hat{\mathbf{v}}' = 11101\ 11001\ 00101\ 00110\ 10010\ 0 \dots 0,$$

deci $\mathbf{v} = \alpha^{10}\mathbf{0}\alpha^2\mathbf{0}\alpha^6\alpha^{14}\mathbf{0} \dots \mathbf{0}$.

Singura ştergere localizată este α^1, deci $\sigma_B(X) = \alpha + X$.

Trebuie să determinăm polinomul de localizare a erorilor.

Din $v(X) = \alpha^{10} + \alpha^2X^2 + \alpha^6X^4 + \alpha^{14}X^5$ calculăm sindromurile

$$S_0 = \alpha^5,\ S_1 = \mathbf{0},\ S_2 = \alpha^3,\ S_3 = \alpha^4,\ S_4 = \alpha^3.$$

Deoarece $B_0 = \alpha$ şi $s = 1$, relaţia de recurenţă pentru sindromurile modificate este ${S_j}' = B_0S_j + S_{j+1} = \alpha S_j + S_{j+1} \quad (0 \le j \le 3)$.

Se determină astfel ${S_0}' = \alpha^6,\ {S_1}' = \alpha^3,\ {S_2}' = \mathbf{0},\ {S_3}' = \alpha^{11}$.

Deoarece $d = 5$ avem $t = 2$, deci sistemul (3) este

$$\begin{pmatrix} {S_0}' & {S_1}' \\ {S_1}' & {S_2}' \end{pmatrix} \begin{pmatrix} A_0 \\ A_1 \end{pmatrix} = \begin{pmatrix} {S_2}' \\ {S_3}' \end{pmatrix} \quad \text{sau} \quad \begin{pmatrix} \alpha^6 & \alpha^3 \\ \alpha^3 & \mathbf{0} \end{pmatrix} \begin{pmatrix} A_0 \\ A_1 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \alpha^{11} \end{pmatrix}$$

cu soluţia $A_0 = \alpha^8,\ A_1 = \alpha^{11}$.

S-a obţinut polinomul $\sigma_{A\setminus B}(X) = \alpha^8 + \alpha^{11}X + X^2$. Deci,

$\sigma_A(X) = \sigma_B(X)\sigma_{A\setminus B}(X) = (\alpha + X)(\alpha^8 + \alpha^{11}X + X^2) = (\alpha + X)(\alpha^3 + X)(\alpha^5 + X)$.

Rezultă că erorile sunt localizate pe poziţiile $X_1 = \alpha,\ X_2 = \alpha^3,\ X_3 = \alpha^5$.

Mai departe se lucrează pentru determinarea valorilor erorilor, folosind algoritmul Peterson şi sindromurile originale. Trebuie rezolvat sistemul:

$$\begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \alpha & \alpha^3 & \alpha^5 \\ \alpha^2 & \alpha^6 & \alpha^{10} \end{pmatrix} \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = \begin{pmatrix} \alpha^5 \\ \mathbf{0} \\ \alpha^3 \end{pmatrix}$$

cu soluţia $Y_1 = \alpha^{12},\ Y_2 = \mathbf{1},\ Y_3 = \alpha^3$.

Deci $v(X)$ se decodifică în $c(X) = v(X) + e(X) = (\alpha^{10} + \alpha^2X^2 + \alpha^6X^4 + \alpha^{14}X^5) + (\alpha^{12}X + X^3 + \alpha^3X^5)$, adică $\mathbf{c} = \alpha^{10}\alpha^{12}\alpha^2\mathbf{1}\alpha^6\mathbf{10} \dots \mathbf{0}$, sau – în $\hat{C}'$:

$$11101\ 11110\ 00101\ 10001\ 00110\ 10001\ 0 \dots 0.$$

13.6 Exerciţii

13.1 *Construiţi matricea generatoare şi listaţi toate cuvintele - cod ale codului RS din Exemplul 20.3. Scrieţi şi forma binară a acestor cuvinte.*

13.2 *Fie codul definit de* $g(X) = (1+X)(\alpha+X)(\alpha^2+X)(\alpha^3+X)$*, bazat pe extensia Galois* $GF(2^3)$ *generată de* α*, rădăcină a polinomului* $1+X+X^3$*.*

1. *Determinaţi* n, k, d *şi numărul de cuvinte - cod.*

2. *Construiţi o matrice generatoare şi o matrice de control a codului.*

3. *Codificaţi mesajele de informaţie* $\mathbf{10}\alpha^2$, $\mathbf{111}$, $\alpha^2\alpha^4\alpha^6$.

13.3 *Construiţi coduri RS cu următorii parametri:*
(a) $m = 2,\ d = 3,\ m_0 = 2$*;*
(b) $m = 3,\ d = 3,\ m_0 = 2$*;*
(c) $m = 3,\ d = 5,\ m_0 = 0$*;*
(d) $m = 4,\ d = 5,\ m_0 = 0$*;*
(e) $m = 5,\ d = 7,\ m_0 = 0$.

13.4 *Pentru fiecare cod determinat la exerciţiul anterior determinaţi* n, k *şi numărul de cuvinte - cod.*

13.5 *Arătaţi că un* $(2^m - 1, k)$ *- cod RS are* 2^{mk} *cuvinte - cod.*

13.6 *Fie codul RS de polinom generator* $g(X) = (1+X)(\alpha+X)(\alpha^2+X)(\alpha^3+X)(\alpha^4+X)(\alpha^5+X) \in GF(2^4)[X]$ *unde* α *este rădăcina polinomului* $1+X+X^4$*. Decodificaţi mesajele:*

(a) $\mathbf{0}\alpha^3\alpha\alpha^5\alpha^3\alpha^2\alpha^6\alpha^{10}\alpha\mathbf{000000}$
(b) $\mathbf{1}\alpha^4\alpha^2\alpha\mathbf{0010}\alpha\alpha^5\alpha^3\alpha^2\mathbf{0}\alpha^{10}\alpha$
(c) $\alpha\mathbf{0}\alpha^7\mathbf{0}\alpha^{12}\alpha^3\alpha^3\mathbf{10000000}$

13.7 *Fie codul RS generat de* $g(X) = (\alpha+X)(\alpha^2+X)(\alpha^3+X)(\alpha^4+X)$ *unde* $\alpha \in GF(2^4)$ *este rădăcina polinomului* $1+X+X^4$*. Decodificaţi mesajele:*

(a) $\mathbf{001}\alpha^8\mathbf{00}\alpha^5\mathbf{00000000}$
(b) $\mathbf{0}\alpha^{10}\mathbf{0}\alpha^6\alpha^{13}\mathbf{0}\alpha^8\alpha^{11}\alpha^3\alpha^5\mathbf{00000}$
(c) $\alpha^4\mathbf{0100}\alpha^2\alpha^5\alpha^{12}al^{14}\mathbf{000000}$

13.8 *Plecând de la codul RS din exerciţiul precedent, se formează codul scurt* $C(4)$ *de lungime* $n = 11$ *(şi* $k = 7$*). Decodificaţi mesajele:*

(a) $\mathbf{001}\alpha^8\mathbf{00}\alpha^5\mathbf{0000}$
(b) $\mathbf{0}\alpha^{10}\mathbf{0}\alpha^6\alpha^{13}\mathbf{0}\alpha^8\alpha^{11}\alpha^3\alpha^5\mathbf{0}$
(c) $\alpha^4\mathbf{0100}\alpha^2\alpha^5\alpha^{12}\alpha^{14}\mathbf{00}$

13.9 *Arătaţi că transformata Fourier finită* M *este inversabilă.*

13.10 *Fiind dat $GF(2^3)$ construit cu polinomul $1+X+X^3$, găsiţi matricea generatoare a unui cod DMS de lungime 7, pentru spaţiul funcţiilor definit pe $S = GF(2^3) \setminus \{0\}$, cu baza:*

(a) $\{X, X^2, X^3\}$; *(b)* $\{\mathbf{1}, X, X^2, X^3, X^4\}$; *(c)* $\{X, X^2, X^3\}$.

Arătaţi că toate aceste coduri sunt ciclice. Determinaţi polinomul generator pentru fiecare cod.

13.11 *Fiind dat $GF(2^3)$ construit cu $1+X+X^3$, determinaţi pentru fiecare $G(X)$ reprezentarea $\mathbf{v_g}$:*

$$(a) \quad G(X) = X + \alpha X^3, \qquad (b) \quad G(X) = 1 + X^2 + X^4.$$

13.12 *În aceleaşi condiţii din exerciţiul precedent, determinaţi $G(X)$ ştiind valorile lui $\mathbf{v_g}$:*

$$(a) \quad \mathbf{v_g} = \alpha^3\alpha\alpha^4\mathbf{0}\alpha^6\alpha^5\alpha^2; \qquad (b) \quad \mathbf{v_g} = \alpha^4\alpha^2\alpha\alpha^3\mathbf{0}\alpha^6\mathbf{1}.$$

13.13 *Folosind codul definit în Exemplul 18.7, decodificaţi mesajele:*

(a) 11101 11110 11010 00111 11110 10100 10110 10100 0 . . . 0

(b) 11000 00000 01010 11111 11011 00000 10001 00101 0 . . . 0

(c) 00000 10000 10000 10000 00101 10100 11101 0 . . . 0.

Capitolul 14

Alte clase de coduri ciclice binare

14.1 Coduri Hamming ciclice

Să considerăm spaţiul nul al matricii

$$H = (\mathbf{1}\ \alpha\ \alpha^2\ \ldots\ \alpha^{n-1}),$$

unde $\alpha \in GF(2^p)$ este primitiv, deci $n = ord(\alpha) = 2^p - 1$.

Cum α poate fi considerat ca un vector coloană cu p componente, H este o matrice $p \times (2^p - 1)$ în care toate coloanele sunt distincte. Acesta este după cum se ştie (Prelegerea 3) un cod Hamming $(2^p - 1, 2^p - p - 1)$ - construit însă ca un cod ciclic.

Exemplul 14.1 *Fie $\alpha \in GF(2^3)$, α rădăcină a polinomului ireductibil $1 + X + X^3$. Am văzut că $ord(\alpha) = 7$, deci α este primitiv. Construim matricea*

$$H = (\mathbf{1}\ \alpha\ \alpha^2\ \alpha^3\ \alpha^4\ \alpha^5\ \alpha^6) = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

Ea este matricea de control pentru un $(7, 4)$ - cod Hamming binar.

Deci codul studiat în Exemplul 3.1 (Prelegerea 3) poate fi tratat şi ca un cod ciclic de polinom generator $h(X) = 1 + X + X^3$.

Un cod Hamming binar ciclic se defineşte cerând ca orice polinom - cod $\{f(X)\}$ să admită ca rădăcină un element primitiv $\alpha \in GF(2^p)$. Generatorul unui asemenea cod este chiar polinomul minimal al lui α : $g(X) = m(X)$.

Deoarece şi α^2 este rădăcină a polinoamelor - cod (Teorema 7.9), codul Hamming binar ciclic este un caz particular de cod BCH cu $d = 3$, capabil să corecteze o eroare.

Codificarea se realizează folosind un circuit liniar cu $p = n-k$ elemente de înmagazinare (vezi paragraful 5.4.2, Prelegerea 5), corespunzător polinomului minimal $m(X)$. Calculul sindromului cuvântului recepţionat $\{r(X)\}$ – adică $\mathbf{r}H^T = r(\alpha)$ se realizează folosind acelaşi circuit de împărţire cu $m(X)$. Acest sindrom - în cazul în care este nenul, deci a intervenit o eroare simplă – este una din coloanele matricii H, adică una din valorile $\mathbf{1}, \alpha, \alpha^2, \ldots, \alpha^{n-1}$.

Pentru a afla care componentă a fost afectată de eroare, se poate folosi acelaşi circuit în felul următor: neglijăm intrarea şi lăsăm circuitul să funcţioneze, având iniţial în elementele de înmagazinare sindromul $r(\alpha)$, care este de forma

$$\alpha^j \ (0 < j \leq 2^p - 2).$$

La fiecare tact, circuitul înmulţeşte conţinutul elementelor de înmagazinare cu α; deci, după $n - j = 2^p - 1 - j$ momente, în elementele de înmagazinare se obţine

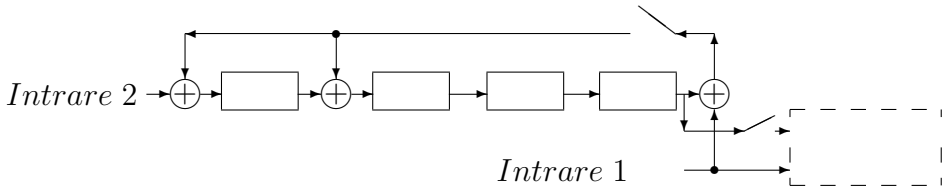
$$\alpha^{j+2^p-1-j} = \alpha^{2^p-1} = \alpha^n = \alpha^0 = \mathbf{1} = (1, 0, \ldots, 0).$$

Cum componentele cuvântului recepţionat sunt scrise în ordinea descrescătoare a puterilor (prima componentă este coeficientul termenului de rang maxim), înseamnă că va trebui să corectăm componenta $n - j$ din cuvânt, înlocuind 0 cu 1 sau 1 cu 0.

Exemplul 14.2 *Fie $\alpha \in GF(2^4)$, rădăcină a polinomului $1 + X + X^4$. Codul Hamming binar $(15, 11)$ obţinut folosind ca rădăcină pe α – deci de polinom generator $g(X) = 1 + X + X^4$, are matricea de control*

$$H = (\mathbf{1}\ \alpha\ \ldots\ \alpha^{14}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Circuitul care realizează (în funcţie de necesităţi) codificarea, calculul sindromului şi corectarea erorilor simple este (vezi şi paragraful 5.4.2):



Codificarea se obţine introducând la Intrarea 1 mesajul de informaţie de lungime k, mesaj considerat ca un polinom de gradul $n - 1$ cu primii coeficienţi (corespunzători termenilor de rang mare) elementele mesajului informaţional şi cu ultimii $n - k$ coeficienţi nuli.

La momentul iniţial în elementele de înmagazinare se află numai 0, circuitul feedback este închis, iar circuitul spre canal este deschis. De asemenea, Intrarea 2 este ignorată.

Simultan cu transmiterea pe canal, elementele de informaţie se introduc şi în circuitul liniar. După k momente, aici se află poziţiile de control.

În acest moment se decuplează feedback-ul, se cuplează legătura spre canal şi se ignoră cele două intrări. Timp de $n - k$ tacţi, elementele de control înmagazinate sunt trimise pe canal, imediat după mesajul de informaţie.

Pentru calculul sindromului: se neglijează canalul, se închide feedback-ul, iar în elementele de înmagazinare se găseşte iniţial $(0, 0, \ldots, 0)$. Cuvântul recepţionat este introdus prin Intrarea 2. După n tacţi, aici se va găsi $r(\alpha)$.

Pentru corectarea erorii: se neglijează cele două intrări, se închide feedback-ul şi se lasă să funcţioneze circuitul până când în elementele de înmagazinare se obţine vectorul $(1, 0, \ldots, 0)$. Numărul de tacţi scurşi dă indicele componentei din mesajul recepţionat, care trebuie corectată prin complementare.

Să considerăm spaţiul nul al matricii

$$H = \begin{pmatrix} \mathbf{1} & \alpha & \alpha^2 & \dots & \alpha^{2^p-2} \\ 1 & 1 & 1 & \dots & 1 \end{pmatrix},$$

unde $\alpha \in GF(2^p)$ este primitiv. H are $p+1$ linii şi $n = 2^p - 1$ coloane (pe prima coloană, primele p poziţii formează vectorul $\mathbf{1} = \alpha^0$, iar pe a $p+1$-a poziţie se află scalarul $1 \in Z_2$). Codul obţinut este codul Hamming binar extins, capabil să corecteze o eroare şi să detecteze două erori (vezi şi Algoritm B, Prelegerea 3).

Un asemenea cod se poate defini cerând ca orice cuvânt - cod $\{f(X)\}$ să aibă ca rădăcini α şi $\mathbf{1}$. Generatorul unui astfel de cod ciclic este $g(X) = (\mathbf{1} + X)m(X)$, unde $m(X)$ este polinomul minimal al lui α.

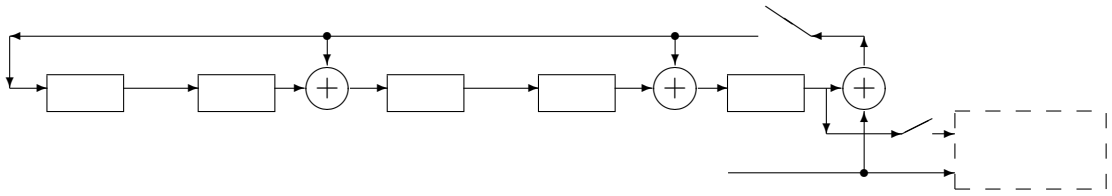
Codificarea în acest caz se face – ca mai sus – folosind circuitul cu $n - k + 1 = p + 1$ elemente de înmagazinare. Pentru calculul sindromului cuvântului recepţionat $\{r(X)\}$ se folosesc însă practic două circuite: unul de împărţire la $m(X)$ (pentru calculul lui $r(\alpha)$) şi altul de împărţire la $\mathbf{1} + X$ (pentru $r(\mathbf{1})$). Discuţia este următoarea:

- Dacă $r(\alpha) = r(\mathbf{1}) = \mathbf{0}$, atunci nu a intervenit nici o eroare (sau avem o eroare nedetectabilă);
- Dacă $r(\alpha) \neq \mathbf{0},\ r(\mathbf{1}) \neq \mathbf{0}$, avem o eroare simplă care se corectează cu circuitul de împărţire cu $m(X)$, după procedeul descris în Exemplul 20.3;
- Dacă $r(\alpha) \neq \mathbf{0},\ r(\mathbf{1}) = \mathbf{0}$, atunci au fost detectate două erori.

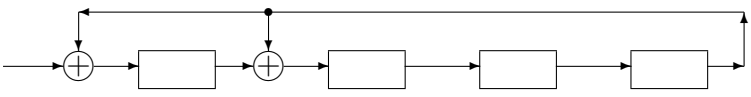
Exemplul 14.3 *Să extindem codul Hamming binar din Exemplul 20.3. Vom avea polinomul generator*

$$g(X) = (1 + X)(1 + X + X^4) = 1 + X^2 + X^4 + X^5$$

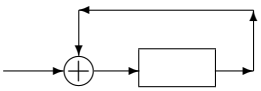
Circuitul liniar care codifică mesajele de informaţie (de lungime $k = 10$) este:



Sindromul se calculează cu ajutorul circuitului liniar de împărţire cu polinomul $m(X) = 1 + X + X^4$:



iar $r(\mathbf{1})$ se obţine din circuitul de împărţire la $1 + X$:



Circuitul liniar construit pentru $m(X)$ efectuează şi corectarea erorilor simple.

14.2 Coduri Hamming ciclice nebinare

Am studiat codurile Hamming atât sub forma liniară, cât şi ca un caz particular de coduri Reed - Muller sau coduri ciclice. În această ultimă situaţie, ele se pot implementa cu ajutorul circuitelor liniare şi – fapt important – admit o variantă de generalizare naturală la cazul nebinar.

Definiţia 14.1 *Fie $\alpha \in GF(q^p)$ un element primitiv. Se defineşte codul Hamming ciclic C astfel:*

$$\{f(X)\} \in C \quad \Longleftrightarrow \quad f(X) \text{ are rădăcina } \beta = \alpha^{q-1}.$$

sau – echivalent – C are matricea de control $H = (\mathbf{1}\ \beta\ \beta^2 \dots \beta^{n-1})$ unde $n = ord(\beta)$.

În cazul $q = 2$ se obţine definiţia codului Hamming binar.

Din

$$\beta^n = \mathbf{1} = (\alpha)^{q^p-1} = \left(\alpha^{q-1}\right)^{\frac{q^p-1}{q-1}} \text{ rezultă } n = \frac{q^p-1}{q-1}.$$

Se observă imediat că H are un număr maxim de coloane distincte şi nenule.

Propoziţia 14.1 $(n, q-1) = 1$ *dacă şi numai dacă* $(p, q-1) = 1$.

Demonstraţie: Din $q-1 | q^j - 1$ $(1 \le j \le p)$ rezultă $q^j = (q-1)s_j + 1$. Deci

$$n = \frac{q^p-1}{q-1} = q^{p-1} + q^{p-2} + \ldots + q + 1 = (q-1)(s_{p-1} + \ldots + s_1) + p$$

şi atunci

$$(p, q-1) = 1 \quad \Longleftrightarrow \quad (n, q-1) = 1. \qquad \text{q.e.d.}$$

Teorema 14.1 *Spaţiul nul al matricii $H = (\mathbf{1}\ \beta\ \beta^2 \dots \beta^{n-1})$ unde $\beta = \alpha^{q-1}$, $n = \dfrac{q^p-1}{q-1}$ şi $\alpha \in GF(q^p)$ primitiv, are distanţa minimă $d \ge 3$ dacă şi numai dacă $(n, q-1) = 1$.*

Demonstraţie: "$\Longleftarrow$": Să presupunem că există două coloane liniar dependente în matricea H. Deci $\beta^i = a\beta^j$ cu $a \in Z_q$, adică $\beta^{i-j} = a$.

Elementele nenule din Z_q sunt rădăcinile ecuaţiei $X^{q-1} - 1 = 0$ (Teorema 7.3). Mai mult, ele sunt chiar primele $q-1$ puteri ale lui α^n. Într-adevăr, pentru orice $s = 0, 1, \ldots, q-2$ avem

$$(\alpha^{ns})^{q-1} = \left(\alpha^{q^p-1}\right)^s = \mathbf{1}^s = \mathbf{1}.$$

Deci pentru orice $a \in Z_q \setminus \{0\}$ există s $(0 \le s \le q-1)$ cu $a = \alpha^{ns}$. Se poate scrie atunci $\beta^{i-j} = \alpha^{ns}$, adică $\alpha^{(i-j)(q-1)} = \alpha^{ns} \quad \Rightarrow \quad (i-j)(q-1) = ns$.

Dar $s < q-1$ iar $(n, q-1) = 1$, deci $i = j$, de unde rezultă $a = 1$, $\beta^i = \beta^j$, imposibil pentru că toate coloanele lui H sunt distincte. Rezultă că orice două coloane din H sunt liniar independente, adică (Teorema 2.4) $d \ge 3$.

"$\Longrightarrow$": Dacă $q-1$ şi n nu sunt prime între ele, atunci ecuaţia anterioară admite o soluţie cu $i \ne j$; deci există două coloane distincte liniar dependente β^i şi β^j în matricea H, ceea ce contrazice faptul că $d \ge 3$. q.e.d.

Pentru operaţia de codificare se foloseşte circuitul de împărţire cu polinomul minimal $m(X)$ al lui β (care aici este chiar polinomul generator al codului Hamming ciclic C). Acelaşi circuit va calcula sindromul $r(\beta)$ al polinomului recepţionat $\{r(X)\}$.

În cazul în care a intervenit o eroare, $r(\beta)$ este de forma $b\beta^j$ (eroarea b a apărut pe poziţia β^j). Funcţionarea secvenţială a circuitului liniar (ignorând intrarea) revine la înmulţirea succesivă cu β. Deci, după $n-j$ tacţi, în elementele de înmagazinare se obţine vectorul $(b\ 0\ \ldots\ 0)$. Atunci, din componenta $n-j$ a cuvântului recepţionat se va scădea eroarea b. De observat că – deoarece coloanele lui H sunt nenule şi distincte – primul tact în care se obţine un vector cu toate componentele nule înafară de prima, este $n-j$.

Exemplul 14.4 *Să considerăm $p = q = 3$ (astfel, $(m, q-1) = 1$). Folosind rădăcina α a polinomului primitiv $1 + 2X + X^3$ vom genera extensia $GF(3^3)$. Fie $\beta = \alpha^{q-1} = \alpha^2$. Deoarece $\alpha^{3^3-1} = \alpha^{26} = 1$, avem $ord(\beta) = 13$, deci $n = 13$. Pentru a construi polinomul minimal $m(X)$ al lui β, să observăm (Secţiunea 7.3) că el are ca rădăcini α^2, α^6 şi α^{18} ($\alpha^{54} = \alpha^2$). Deci $m(X) = a + bX + cX^2 + X^3$. Punând condiţia $m(\alpha^2) = a + b\alpha^2 + c\alpha^4 + \alpha^6 = \mathbf{0}$, obţinem ecuaţia matricială:*

$$a\begin{pmatrix}1\\0\\0\end{pmatrix} + b\begin{pmatrix}0\\0\\1\end{pmatrix} + c\begin{pmatrix}0\\2\\1\end{pmatrix} + \begin{pmatrix}1\\1\\1\end{pmatrix} = \begin{pmatrix}0\\0\\0\end{pmatrix}$$

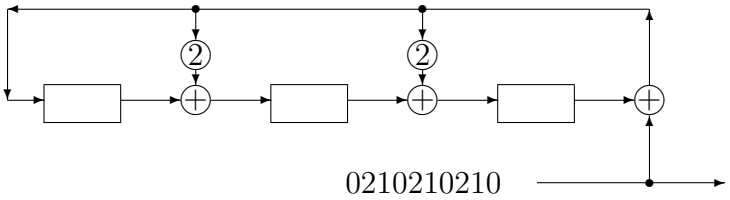
cu soluţia $a = 2,\ b = c = 1$. Deci $m(X) = 2 + X + X^2 + X^3$.

Codul Hamming ciclic de polinom generator $g(X) = m(X) = 2 + X + X^2 + X^3$ are matricea de control

$$H = (\mathbf{1}\ \alpha^2\ \alpha^4\ \ldots\ \alpha^{24}) = \begin{pmatrix} 1 & 0 & 0 & 1 & 2 & 0 & 2 & 0 & 1 & 1 & 1 & 2 & 1 \\ 0 & 0 & 2 & 1 & 0 & 1 & 0 & 2 & 2 & 2 & 1 & 2 & 1 \\ 0 & 1 & 1 & 1 & 2 & 1 & 1 & 0 & 0 & 1 & 2 & 0 & 2 \end{pmatrix}.$$

Am definit astfel $(13, 10)$ - codul Hamming ternar, evocat în Exemplul 3.5 relativ la Problema Pronosportului.

Să considerăm mesajul de informaţie 0120120120. Pentru operaţia de codificare se va folosi circuitul (pentru detalii, a se revedea secţiunea 5.4.2):



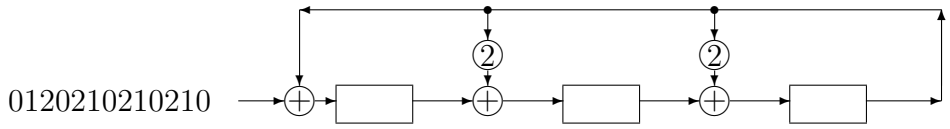
(s-a ţinut cont că se lucrează în Z_3, deci $-1 = 2,\ -2 = 1$). În primii 10 tacţi de funcţionare, biţii de informaţie sunt trimişi pe canal şi – simultan – intră în circuitul liniar, a cărui funcţionare este (pe prima coloană se află informaţia, iar pe celelalte trei – conţinuturile elementelor de înmagazinare):

0	0	0	0		2	2	1	2
1	1	2	2		0	2	0	2
2	1	0	1		1	0	2	0
0	1	0	2		2	2	1	0
1	0	1	0		0	0	2	1

În elementele de înmagazinare au rămas caracterele de control 021. Ele sunt trimise prin canal la tacţii $11 - 13$ de la dreapta spre stânga, cu semn schimbat: în ordine $-1 =$

$2,\ -2 = 1,\ -0 = 0$. *Deci cuvântul - cod transmis a fost* $\mathbf{a} = 0120120120210$. *Se verifică imediat că* $\mathbf{a}H^T = \mathbf{0}$.

La recepţie, primele 10 *caractere sunt cele de informaţie. Pentru a verifica dacă au apărut sau nu erori, se foloseşte acelaşi circuit, dar cu altă intrare:*



Funcţionarea acestui circuit liniar timp de 13 *tacţi, este dată de tabelul:*

	0	0	0		0	1	2	0
0	0	0	0		1	1	1	2
1	1	0	0		2	1	2	2
2	2	1	0		0	2	2	0
0	0	2	1		2	2	2	2
1	2	2	1		1	0	0	0
2	0	1	1		0	0	0	0

Sindromul $a(\alpha^2)$ *obţinut este* 000, *deci nu a apărut nici o eroare.*

Dacă s-ar fi recepţionat însă cuvântul $\mathbf{a}' = 0120120100210$, *după funcţionarea circuitului timp de* 13 *tacţi s-ar fi ajuns în elementele de înmagazinare la* 220. *Cum acesta este nenul, circuitul se lasă să funcţioneze în continuare, ignorând intrarea, până se ajunge la un sindrom de forma* $b00$ *cu* $b \neq 0$. *Mai exact, funcţionarea circuitului va fi (pe prima coloană se numără tacţii):*

0	2	2	0		5	1	0	1
1	0	2	2		6	1	0	2
2	2	1	0		7	2	2	1
3	0	2	1		8	1	1	1
4	1	2	1		9	1	0	0

Deci, după 9 *tacţi s-a ajuns la* 100. *Corecţia se face astfel: din componenta a* 9 *- a a cuvântului recepţionat se scade* $b = 1$, *obţinându-se* $0 - 1 = 2\ (mod\ 3)$.

Observaţia 14.1 *Definiţia codurilor Hamming ciclice nebinare nu este echivalentă cu cea a codurilor Hamming nebinare dată în Secţiunea* 3.1.1. *De exemplu, codul (dat în Exemplul* 3.4*) cu matricea de control*

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 \end{pmatrix}$$

este un $(4, 2)$ *- cod Hamming ternar. El conţine* 9 *cuvinte*

$$\{(0000,\ 1012,\ 2021,\ 0111,\ 1120,\ 2102,\ 0222,\ 1201,\ 2210)\},$$

care – evident – nu formează un cod ciclic.

Deci între cele două definiţii ale codurilor Hamming nebinare (liniare şi ciclice) există doar o relaţie de incluziune strictă (orice cod Hamming ciclic este un cod Hamming liniar).

14.3 Coduri Goppa

Această clasă de coduri a fost introdusă de Goppa în 1970. Deşi în general sunt coduri neciclice, ele constituie o generalizare a codurilor BCH şi formează o punte între teoria codurilor şi criptografie. Din acest motive, prezentarea lor este deosebit de utilă.

Fie q un număr prim şi $g(X) \in Z_q[X]$. Dacă $a \in Z_q$ este un element cu proprietatea $g(a) \neq 0$, vom nota

$$\frac{1}{X-a} = -\frac{g(X)-g(a)}{X-a} g(a)^{-1}.$$

Propoziţia 14.2 *În algebra polinoamelor din $Z_q[X]$ modulo $g(X)$, polinomul $\frac{1}{X-a}$ este inversul lui $X-a$.*

Demonstraţie: Faptul că $\frac{1}{X-a}$ este polinom se verifică imediat. Pe de-altă parte,

$$(X-a)\frac{1}{X-a} = [g(a)-g(X)]g(a)^{-1} = 1 - g(a)^{-1}g(X) \equiv 1 \ (mod\ g(X)).$$ q.e.d.

Definiţia 14.2 *Un cod Goppa de lungime n peste Z_q este determinat prin:*

1. *Un polinom $g(X) \in Z_q[X]$ de grad t $(t \geq 2)$ definit peste o extensie $GF(q^m)$ a lui Z_q;*
2. $\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n} \in GF(q^m)$ *cu* $g(\mathbf{a_i}) \neq \mathbf{0} \quad (1 \leq i \leq n)$.

Codul conţine toate cuvintele $\mathbf{v} = v_1 v_2 \ldots v_n \in Z_q^n$ *pentru care* $\sum_{i=1}^{n} \frac{v_i}{X - \mathbf{a_i}} = \mathbf{0}$.

Teorema 14.2 *Codul Goppa conţine numai cuvintele $\mathbf{v} = v_1 v_2 \ldots v_n$ pentru care*

$$\sum_{i=1}^{n} v_i g(\mathbf{a_i})^{-1} \mathbf{a_i}^s = \mathbf{0} \quad (0 \leq s \leq t-1)$$

unde $t = grad(g(X))$.

Demonstraţie: Să explicităm polinomul $\frac{1}{X-a}$. Dacă $g(X) = g_0 + g_1 X + \ldots + g_t X^t$ $(g_t \neq 0)$, atunci pentru orice $a \in GF(q^m)$ cu $g(a) \neq 0$, are loc descompunerea

$\frac{g(X)-g(a)}{X-a} = g_t X^{t-1} + (ag_t + g_{t-1})X^{t-2} + (a^2 g_t + a g_{t-1} + g_{t-2})X^{t-3} + \ldots + (a^{t-1} g_t + a^{t-2} g_{t-1} + \ldots + g_1)$.

Deci coeficientul lui X^{p-1} $(1 \leq p \leq t)$ în polinomul $\frac{1}{X-a}$ este $-g(a)^{-1} \sum_{j=p}^{t} a^{t-j} g_j$.

Ecuaţia caracteristică a codurilor Goppa $\sum_{i=1}^{n} \frac{v_i}{X - a_i} = \mathbf{0}$ implică faptul că toţi coeficienţii lui X^{p-1} $(1 \leq p \leq t)$ sunt nuli. Deci

$$\sum_{i=1}^{n} v_i g(\mathbf{a_i})^{-1} \sum_{j=p}^{t} \mathbf{a_i}^{t-j} g_j = \mathbf{0}, \quad (1 \leq p \leq t).$$

Cazul $p = t$ conduce la $g_t \sum_{i=1}^{n} v_i g(\mathbf{a_i})^{-1} = \mathbf{0}$ şi – deoarece $g_t \neq 0$ – se obţine $\sum_{i=1}^{n} v_i g(\mathbf{a_i})^{-1} = \mathbf{0}$.

Din cazul $p = t-1$ se obţine $g_t \sum_{i=1}^{n} v_i g(\mathbf{a_i})^{-1}\mathbf{a_i} + g_{t-1} \sum_{i=1}^{n} v_i g(\mathbf{a_i})^{-1} = \mathbf{0}$. Cum a doua sumă este zero, iar $g_t \neq 0$, va rezulta $\sum_{i=1}^{n} v_i g(\mathbf{a_i})^{-1}\mathbf{a_i} = \mathbf{0}$ ş.a.m.d.

Am obţinut echivalenţa

$$\sum_{i=1}^{n} \frac{v_i}{X - a_i} = \mathbf{0} \quad \Longleftrightarrow \quad \sum_{i=1}^{n} v_i g(\mathbf{a_i})^{-1}\mathbf{a_i}^s = \mathbf{0} \quad (0 \leq s \leq t-1)$$

q.e.d.

Corolarul 14.1 *Codul Goppa determinat de un polinom $g(X)$ de gradul t are matricea de control*

$$H = \begin{pmatrix} \frac{1}{g(a_1)} & \frac{1}{g(a_2)} & \frac{1}{g(a_3)} & \cdots & \frac{1}{g(a_n)} \\ \frac{a_1}{g(a_1)} & \frac{a_2}{g(a_2)} & \frac{a_3}{g(a_3)} & \cdots & \frac{a_n}{g(a_n)} \\ & & \vdots & & \\ \frac{a^{t-1}}{g(a_1)} & \frac{a_2^{t-1}}{g(a_2)} & \frac{a_3^{t-1}}{g(a_3)} & \cdots & \frac{a_n^{t-1}}{g(a_n)} \end{pmatrix}.$$

Demonstraţie: Este lăsată ca exerciţiu.

Exemplul 14.5 *Polinomul $1 + X + X^3 \in Z_2[X]$ este ireductibil. Fie α o rădăcină a sa, cu ajutorul căreia se generează extensia $GF(2^3)$. Dacă alegem $g(X) = 1 + X + X^2$ şi drept elemente $\mathbf{a_i}$ cele 8 componente ale lui $GF(2^3)$, vom obţine un cod Goppa de lungime 8, cu matricea de control*

$$H = \begin{pmatrix} \frac{1}{g(0)} & \frac{1}{g(\mathbf{1})} & \frac{1}{g(\alpha)} & \cdots & \frac{1}{g(\alpha^6)} \\ \mathbf{0} & \frac{1}{g(\mathbf{1})} & \frac{\alpha}{g(\alpha)} & \cdots & \frac{\alpha^6}{g(\alpha^6)} \end{pmatrix} = \begin{pmatrix} \mathbf{1} & \mathbf{1} & \alpha^2 & \alpha^4 & \alpha^2 & \alpha & \alpha & \alpha^4 \\ \mathbf{0} & \mathbf{1} & \alpha^3 & \alpha^6 & \alpha^5 & \alpha^5 & \alpha^6 & \alpha^3 \end{pmatrix} =$$

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

Acest cod este format din numai patru cuvinte:

$$\{00000000, \quad 00111111, \quad 11001011, \quad 11110100\}.$$

El are distanţa minimă 5; s-a obţinut deci un $(8, 2)$ - cod binar corector de două erori. De remarcat că acesta nu este un cod ciclic.

Observaţia 14.2 *Să considerăm un cod BCH definit prin rădăcinile*

$$\alpha^{m_0}, \alpha^{m_0+1}, \ldots, \alpha^{m_0+d-2}$$

unde $\alpha \in GF(q^m)$ are ordinul n. Acest cod se poate scrie ca un cod Goppa astfel: se ia polinomul $g(X) = X^{m_0+d-2}$ şi elementele $\alpha^0, \alpha^{-1}, \ldots, \alpha^{-(n-1)}$. Vom avea pentru orice $i = 0, 1, \ldots, d-2$ şi $p = 0, 1, \ldots, n-1$:

$$\frac{(\alpha^{-p})^i}{(\alpha^{-p})^{m_0+d-2}} = (\alpha^p)^{m_0+d-2-i} = (\alpha^{m_0+d-2-i})^p$$

Pe baza aceasta, codul Goppa rezultat are matricea de control

$$H = \begin{pmatrix} \mathbf{1} & \alpha^{m_0+d-2} & (\alpha^{m_0+d-2})^2 & \ldots & (\alpha^{m_0+d-2})^{n-1} \\ \mathbf{1} & \alpha^{m_0+d-3} & (\alpha^{m_0+d-3})^2 & \ldots & (\alpha^{m_0+d-3})^{n-1} \\ & & \vdots & & \\ \mathbf{1} & \alpha^{m_0} & (\alpha^{m_0})^2 & \ldots & (\alpha^{m_0})^{n-1} \end{pmatrix}$$

care coincide cu matricea de control a codului BCH considerat, în care s-a efectuat o inversare a liniilor.

Se poate arăta (Van Lint) că singurele coduri Goppa ciclice sunt codurile BCH.

Propoziţia 14.3 *Codul Goppa binar de lungime n, determinat de un polinom de grad t peste $GF(2^m)$ are $k \geq n - mt$ simboluri de informaţie şi distanţa minimă $\geq t + 1$.*

Demonstraţie: Matricea de control a codului Goppa are t linii în $GF(2^m)$, deci mt linii scrise în binar. Rezultă că numărul $n - k$ de simboluri de control este cel mult mt : $n - k \leq mt$.

Pe de-altă parte, se arată uşor că orice combinaţie liniară de t coloane din H este liniar independentă. q.e.d.

Definiţia 14.3 *Fie $g(X)$ un polinom ireductibil peste $GF(q^m)$. Codul Goppa determinat de $g(X)$ şi de toate elementele lui $GF(q^m)$ se numeşte ireductibil.*

Ideea de "ireductibil" este similară codurilor ciclice ireductibile: un cod Goppa ireductibil nu are subcoduri Goppa proprii.

Codul prezentat în Exemplul 17.3 este un cod Goppa ireductibil. Despre codurile Goppa ireductibile se poate prezenta următorul rezultat:

Teorema 14.3 *Un cod Goppa binar ireductibil, determinat de un polinom de gradul t poate corecta cel puţin t erori independente.*

Pentru demonstraţie va trebui să facem o construcţie suplimentară.

Definiţia 14.4 *Derivata formală a polinomului $a(X) = a_0 + a_1X + \ldots + a_nX^n$ este definită prin $a'(X) = a_1 + 2a_2X + 3a_3X^2 + \ldots + na_nX^{n-1}$.*

Lema 14.1 *Pentru derivata formală sunt adevărate egalităt ile:*

1. $(a(X) + b(X))' = a'(X) + b'(X)$;

2. $(a(X)b(X))' = a'(X)b(X) + a(X)b'(X)$

Demonstraţie: Exerciţiu.

Lema 14.2 *Într-un corp de caracteristică* 2 *derivata formală a unui polinom este un pătrat perfect.*

Demonstraţie: Deoarece pentru n par, $(X^n)' = nX^{n-1} = 0$, derivata formală a unui polinom va avea numai puteri pare. Cum fiecare coeficient se poate scrie ca un pătrat perfect (se lucrează într-un corp de caracteristică 2), avem $f'(X) = f_0^2 + f_2^2X^2 + f_4^2X^4 + \ldots$ şi, dacă se ia $g(X) = f_0 + f_2X + f_4X^2 + \ldots$, avem $f'(X) = g^2(X)$. q.e.d.

Să revenim acum la *Demonstraţia Teoremei 20.2*: Fie $g(X) \in Z_2[X]$ un polinom ireductibil de grad t ($t \geq 2$) peste $GF(2^m) = \{\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n}\}$ ($n = 2^m$), α o rădăcină a lui g ($g(\alpha) = 0$) şi $\mathbf{v} = v_1v_2\ldots v_n$ un cuvânt al codului Goppa, $w(\mathbf{v}) = s$. Notăm $v_{i_1}, v_{i_2}, \ldots, v_{i_s}$ biţii egali cu 1 din $\mathbf{v}$.

Deoarece $\mathbf{v}$ este cuvânt - cod, avem $\displaystyle\sum_{i=1}^{n} \frac{v_i}{X - \mathbf{a_i}} = \sum_{k=1}^{s} \frac{1}{X - \mathbf{a_{i_k}}} = \mathbf{0}$.

Polinomul $f(X) = (X - \mathbf{a_{i_1}})(X - \mathbf{a_{i_2}})\ldots(X - \mathbf{a_{i_s}})$ nu este divizibil cu $g(X)$ (deoarece $g(X)$ este ireductibil de grad t). Rezultă că, în algebra claselor de resturi $Z_2[X]/g(X)$, $f(\alpha) \neq 0$ (Teorema 4.1) deci admite invers. În consecinţă, există un polinom $r(X)$ cu $\{f(\alpha)\}\{r(\alpha)\} = 1$.

Să calculăm derivata formală $f'(X) = \displaystyle\sum_{p=1}^{s} \frac{f(X)}{X - \mathbf{a_{i_k}}}$. Făcând $X = \alpha$ şi înmulţind relaţia cu $r(\alpha)$, obţinem

$$\{f'(\alpha)\}\{r(\alpha)\} = \sum_{p=1}^{s} \frac{\{f(\alpha)\}\{r(\alpha)\}}{\alpha - \mathbf{a_{i_p}}} = \sum_{p=1}^{s} \frac{1}{\alpha - \mathbf{a_{i_p}}} = 0 \pmod{g(X)}.$$

Aceasta înseamnă că $f'(X)r(X)$ este divizibil cu $g(X)$. Cum $g(X)$ este ireductibil şi nu divide $r(X)$ (deoarece $r(\alpha) \neq 0$), rezultă $g(X)|f'(X)$.

Deoarece suntem într-un corp de caracteristică 2, conform Lemei 15.1 derivata formală este un pătrat perfect: $f'(X) = h^2(X)$. Din faptul că $g(X)$ este divizor ireductibil al lui $h^2(X)$ rezultă $g(X)|h(X)$.

Deci $g^2(X)|h^2(X) = f'(X)$.

Se ştie că $grad(f'(X)) = s - 1$ şi $s \geq 1$; deci $grad(f'(X)) \geq grad(h^2(X))$, adică

$$s - 1 \geq 2t.$$

Cum ponderea oricărui cuvânt - cod este $\geq 2t+1$, rezultă că distanţa minimă a codului este cel puţin $2t + 1$, deci codul corectează sigur t erori independente. q.e.d.

Exemplul 14.6 *Să considerăm* $GF(2^3)$ *generat de o rădăcină a lui* $g(X) = 1 + X + X^3$. *Deoarece* $g(X)$ *are* 3 *rădăcini în* $GF(2^3)$, *el nu va avea nici o rădăcină în corpul* $GF(2^5)$ *(*$GF(2^5)$ *nu este o extensie a lui* $GF(2^3)$*), deci aici* $g(X)$ *este ireductibil.*

Codul Goppa ireductibil corespunzător are $n = 32$, $k \geq 32 - 5 \cdot 3 = 17$, $d \geq 7$*; deci este un* $(32, 17)$ *- cod corector de* 3 *erori.*

Decodificarea codurilor Goppa se face în manieră similară decodificării codurilor BCH. Astfel, fie C un cod Goppa definit de $g(X) \in Z_q[X]$ ($grad(g(X)) = t$), şi $\mathbf{a_1}, \ldots, \mathbf{a_n} \in GF(q^m)$; considerăm $\mathbf{v} = v_0 v_1 \ldots v_{n-1}$ un cuvânt - cod şi $\mathbf{r} = r_0 r_1 \ldots r_{n-1}$ cuvântul recepţionat după transmiterea lui $\mathbf{v}$.

Vom nota cu $\mathbf{e} = \mathbf{r} - \mathbf{v} = e_0 e_1 \ldots e_{n-1}$ secvenţa - eroare.

Pentru $t/2 \leq w(\mathbf{e}) \leq t$ se poate folosi un algoritm din cei uzuali. În practică însă $t = grad(g(X))$ este mare şi există algoritmi eficienţi de decodificare pentru cazul $w(\mathbf{e}) < t/2$. Un astfel de algoritm vom construi mai jos.

Fie $M = \{i \mid e_i \neq 0\}$, $card(M) = j < t/2$; considerăm sindromul polinomial

$$\{s(X)\} = \sum_{i=0}^{n-1} \frac{e_i}{X - \mathbf{a_i}} \pmod{g(X)}.$$

De remarcat că $\{s(X)\}$ se poate determina prin calcul la primirea cuvântului $\mathbf{r}$. Se definesc polinoamele $\sigma(X)$ (de localizare a erorii) şi $\omega(X)$ prin

$$\sigma(X) = \prod_{i \in M} (X - \mathbf{a_i}), \qquad \omega(X) = \sum_{i \in M} e_i \prod_{k \in M \setminus \{i\}} (X - \mathbf{a_k}).$$

Din definiţie rezultă că $\sigma(X)$ şi $\omega(X)$ sunt prime între ele. În plus, $grad(\sigma(X)) = j$, $grad(\omega(X)) < j$. Între ele există relaţia

$$\{s(X)\}\{\sigma(X)\} = \sum_{i=0}^{n-1} \frac{e_i}{X - \mathbf{a_i}} \prod_{i \in M} (X - \mathbf{a_i}) \equiv \{\omega(X)\} \pmod{g(X)}. \tag{1}$$

$\sigma(X)$ şi $\omega(X)$ sunt polinoamele normate de grad minim care verifică relaţiile de sus. Într-adevăr, să presupunem că există $\sigma_1(X)$ un polinom nenul normat de grad $< j$ şi $\omega_1(X)$ polinomul corespunzător, astfel ca

$$\{s(X)\}\{\sigma_1(X)\} \equiv \{\omega_1(X)\} \pmod{g(X)}$$

Fie $d(X) = cmmdc(s(X), g(X))$. Din relaţiile

$s(X)\sigma(X) = a(X)g(X) + \omega(X)$, $\quad s(X)\sigma_1(X) = b(X)g(X) + \omega_1(X)$,

rezultă că $d(X)$ va divide şi polinoamele $\omega(X)$, $\omega_1(X)$. Deci,

$$\left\{\frac{s(X)}{d(X)}\right\}\{\sigma(X)\} - \left\{\frac{\omega(X)}{d(X)}\right\} \equiv 0,$$

$$\left\{\frac{s(X)}{d(X)}\right\}\{\sigma_1(X)\} - \left\{\frac{\omega_1(X)}{d(X)}\right\} \equiv 0 \qquad \left(mod\ \frac{g(X)}{d(X)}\right)$$

sau

$$\left\{\frac{s(X)}{d(X)}\right\} \frac{\{\omega_1(X)\sigma(X) - \omega(X)\sigma_1(X)\}}{\{d(X)\}} \equiv 0 \pmod{g(X)}.$$

Deci $\{\omega_1(X)\sigma(X) - \omega(X)\sigma_1(X)\} \equiv 0 \pmod{g(X)}$. Deoarece polinomul din membrul stâng are grad mai mic decât t, rezultă că membrul stâng este 0. Apoi, cum $cmmdc(\sigma(X), \omega(X)) = 1$, deducem $\sigma(X) | \sigma_1(X)$ – ceea ce contrazice faptul că $grad(\sigma_1(X)) < grad(\sigma(X))$ şi $\{\sigma_1(X)\} \not\equiv 0$.

Rezumând, procedeul de decodificare rapidă a mesajelor la codurile Goppa are loc astfel:

1. Din mesajul recepţionat $\mathbf{r}$ se calculează sindromul $s(X)$;
2. Se determină polinomul normat $\sigma(X)$ de grad minim $< t/2$ care verifică relaţia (1) (un algoritm eficient a fost dat de Berlekamp).

   Din rezolvarea ecuaţiei $\sigma(X) = 0$ se află locaţiile erorilor;
3. Se determină e_i $(i \in M)$ din relaţiile $\omega(a_i) = e_i \prod_{j \in M \setminus \{i\}} (\mathbf{a_i} - \mathbf{a_j})$.

Pe baza acestor valori se construieşte secvenţa - eroare $\mathbf{e}$;
4. Cuvântul decodificat este $\mathbf{v} = \mathbf{r} - \mathbf{e}$.

14.4 Exerciţii

14.1 *Să se construiască matricea generatoare a* $(13, 10)$ *- codului Hamming ternar din Exemplul 14.4.*

14.2 *În codul din Exemplul 14.4 să se codifice mesajele de informaţie:*
(a) 1122002211 (b) 0000111122 (c) 1020012210
(d) 2200211101 (e) 2222222222 (f) 0000000000

14.3 *Să se construiască toate codurile Hamming peste* Z_3 *şi* Z_5.

14.4 *Demonstraţi Corolarul 15.1*

14.5 *Demontraţi Lema 14.1*

14.6 *Pentru* $g(X) = X^{2t}$ *arătaţi că* $\dfrac{1}{X - a} = \sum_{j=1}^{2t} a^{-j} X^{j-1}$.

14.7 *Găsiţi distanţa minimă a codului Goppa binar determinat de* $g(X) = 1 + X^2$ *şi de elementele* $\mathbf{0}, \alpha, \alpha^2 \in GF(2^2)$.

14.8 *Polinomul generator este* $g(X) = 1 + X^2$ *iar parametrii sunt cele* 15 *elemente nenule ale extensiei* $GF(2^4)$*, generată de rădăcina polinomului* $1 + X + X^4$*. Analizaţi codul Goppa binar astfel construit.*

14.9 *Găsiţi o matrice de control a* $(16, 8)$ *- codului Goppa binar ireductibil, dat de* $g(X) = \alpha^3 + X + X^2$*, unde* $\alpha \in GF(2^4)$ *este primitiv.*

14.10 *Să se construiască codul Goppa ireductibil peste* $GF(2^4)$*, cu* $g(X) = 1 + X + X^3$*. Să se determine matricea de control şi matricea generatoare.*

Care este distanţa minimă a codului ?

14.11 *Folosind codul construit anterior, să se decodifice (folosind algoritmi de coduri liniare) cuvintele:*

0110110010011110; 0111011001000000; 1011110101100100;
1111111111111111; 1011011111001011; 1010010110000010.

14.12 *Pentru acelaşi cod Goppa binar ireductibil, să se decodifice cuvintele*

0101100101111001; 1100101100110100; 0001100010010110

folosind algoritmul rapid specific.

14.13 *Verificaţi că polinomul* $1 + X + X^2$ *este ireductibil în* $GF(2^5)$*. Determinaţi parametrii şi matricea de control a codului Goppa ireductibil corespunzător.*

Capitolul 15

Coduri Preparata

În 1968 Preparata a introdus o clasă de coduri neliniare corectoare de două erori, care s-au dovedit foarte interesante prin proprietăţile şi prin legăturile pe care le au cu diverse alte coduri deja studiate. Ele sunt o combinaţie între codurile Reed - Muller şi codurile BCH corectoare de două erori. Definiţia pe care o vom folosi aparţine lui Baker şi permite o abordare mai simplă a algoritmilor de codificare/decodificare.

15.1 Definirea codurilor Preparata extinse

Să revenim la modalitatea de marcare a poziţiilor cuvintelor de lungime 2^r folosind elementele lui $GF(2^r)$, aşa cum s-a definit în algoritmul de decodificare Peterson pentru codurile BCH. Deci:

Fie $r \geq 2$ un număr întreg, $\alpha \in GF(2^r)$ primitiv, şi $U \subseteq GF(2^r)$; se notează $\chi(U)$ cuvântul de lungime 2^r definit prin

1 pe poziţia i dacă $\alpha^i \in U$ $(0 \leq i \leq 2^r - 2)$,
1 pe poziţia $2^r - 1$ dacă $\mathbf{0} \in U$,
0 în rest.

Exemplul 15.1 *Fie $\alpha \in GF(2^3)$ primitiv. Atunci*

$$\begin{aligned} \chi(\{\mathbf{0}\}) &= 00000001 \\ \chi(\{\alpha^2, \alpha^5, \alpha^6\}) &= 00100110 \\ \chi(\emptyset) &= 00000000. \end{aligned}$$

Se definesc în mod natural operaţiile:

$$U + \beta = \{u + \beta | u \in U\}, \quad \beta U = \{\beta u | u \in U\},$$
$$U \Delta V = \{u | u \in U \cup V, u \notin U \cap V\},\ U, V \subset GF(2^r),\ \beta \in GF(2^r).$$

Se verifică uşor relaţiile

$$\chi(U) + \chi(V) = \chi(U \Delta V), \quad w(\chi(U)) = card(U). \tag{1}$$

Exemplul 15.2 *Fie $GF(2^3)$ construit folosind rădăcina (primitivă) α a polinomului $1 + X + X^3$, şi mulţimile $U = \{\alpha^2, \alpha^5, \alpha^6\}$, $V = \{\alpha^5, \mathbf{0}\}$. Atunci*

$$U + \alpha^2 = \{\alpha^2 + \alpha^2, \alpha^5 + \alpha^2, \alpha^6 + \alpha^2\} = \{\mathbf{0}, \alpha^3, \alpha^0\},$$
$$\alpha^2 U = \{\alpha^2\alpha^2, \alpha^2\alpha^5, \alpha^2\alpha^6\} = \{\alpha^4, \alpha^0, \alpha\},$$

$\chi(U) + \chi(V) = 00100110 + 00000101 = 00100011 = \chi(\{\alpha^2, \alpha^6, \mathbf{0}\}) = \chi(U \Delta V)$.

Definiţia 15.1 *Fie $r > 2$ un număr impar. Codul extins Preparata $P(r)$ este mul-ţimea cuvintelor de forma $\chi(U)\chi(V)$ unde $U, V \subseteq GF(2^r)$ verifică condiţiile*

1. *$card(U)$, $card(V)$ sunt numere pare;*
2. $\sum_{u\in U} u = \sum_{v\in V} v$;
3. $\sum_{u\in U} u^3 + \left(\sum_{u\in U} u\right)^3 = \sum_{v\in V} v^3$.

Pentru uşurinţă vom nota cuvintele - cod cu $[\chi(U), \chi(V)]$ (vezi şi codul Golay).

Exemplul 15.3 *În ipotezele din Exemplul 19.5, să considerăm $U = \{\alpha, \alpha^2, \alpha^5, \mathbf{0}\}$, $V = \{\alpha^0, \alpha, \alpha^2, \alpha^3, \alpha^6, \mathbf{0}\}$. Prima condiţie din definiţie este verificată. Pentru a doua condiţie avem:*
$\sum_{u\in U} u = \alpha + \alpha^2 + \alpha^5 + \mathbf{0} = 010 + 001 + 111 + 000 = 100 = \alpha^0$,
$\sum_{v\in V} v = \alpha^0 + \alpha + \alpha^2 + \alpha^3 + \alpha^6 + \mathbf{0} = 100 + 010 + 001 + 110 + 101 + 000 = 100 = \alpha^0$.
A treia condiţie este şi ea verificată deoarece
$\sum_{u\in U} u^3 = \alpha^3 + \alpha^6 + \alpha + \mathbf{0} = 110 + 101 + 010 + 000 = 001 = \alpha^2$,
$\sum_{v\in V} v^3 = \alpha^0 + \alpha^3 + \alpha^6 + \alpha^2 + \alpha^4 + \mathbf{0} = 100 + 110 + 101 + 001 + 011 + 000 = 101 = \alpha^6$
şi $\alpha^2 + (\alpha^0)^3 = \alpha^6$.
Deci $[\chi(U), \chi(V)] = 01100101\ 11110011$ este cuvânt - cod în $P(3)$.

Observaţii:

- Deoarece $card(\chi(U)) = card(\chi(V)) = 2^r$, $P(r)$ este un cod de lungime 2^{r+1}.
- Faptul că $\mathbf{0}$ este sau nu un element al mulţimilor U sau V nu afectează cu nimic condiţiile 2 şi 3 din Definiţia 15.1. Acest element se introduce în cele două mulţimi numai pentru a asigura prima condiţie din definiţie. Rezultă că biţii de pe poziţia $2^r - 1$ din $\chi(U)$ şi $\chi(V)$ sunt biţi de paritate (ceea ce justifică termenul de cod Preparata extins).
- Exponentul $'3'$ din Definiţia 15.1 nu este esenţial. Dacă el se înlocuieşte cu $s = 2^t + 1$ astfel ca aplicaţiile $f, g : GF(2^r) \longrightarrow GF(2^r)$, $f(x) = x^s$, $g(x) = x^{s-2}$ să fie bijective, toate proprietăţile se păstrează.

  Aceste coduri se numesc *Coduri Preparata generalizate*.

Lema 15.1 *Fie $[\chi(U), \chi(V)], [\chi(A), \chi(B)] \in P(r)$ şi $\beta = \sum_{u\in U} u$.*

Atunci $[\chi(U\Delta A + \beta), \chi(V\Delta B)] \in P(r)$.

Demonstraţie: Vom arăta că noul cuvânt construit verifică condiţiile din Definiţia 15.1.

1. Deoarece $card(U), card(V), card(A), card(B)$ sunt pare, se arată uşor că:
   $card(V \Delta B) = card(V) + card(B) - 2 \cdot card(V \cap B)$,
   $card(U \Delta A + \beta) = card(U \Delta A) = card(U) + card(A) - 2 \cdot card(U \cap A)$.

2. Să observăm întâi că dacă $I, J \subseteq GF(2^r)$, avem $\sum_{x \in I \Delta J} x = \sum_{x \in I} x + \sum_{x \in J} x$ deoarece orice element $\alpha^i \in I \cap J$ apare de două ori în membrul drept şi niciodată în membrul stâng, iar $\alpha^i + \alpha^i = 0$. Deci:
   $\sum_{x \in U \Delta A + \beta} x = \sum_{y \in U \Delta A} (y + \beta) = \sum_{y \in U \Delta A} y + \beta \cdot card(U \Delta A) = \sum_{y \in U} y + \sum_{y \in A} + \mathbf{0}$ (deoarece $card(U \Delta A)$ este par) $= \sum_{y \in V} y + \sum_{y \in B} y = \sum_{y \in V \Delta B} y$.

3. $$\sum_{x \in U \Delta A + \beta} x^3 + \left(\sum_{x \in U \Delta A + \beta} x \right)^3 = \sum_{y \in U \Delta A} (y + \beta)^3 + \left(\sum_{y \in V \Delta B} y \right)^3 =$$
   $$= \sum_{y \in U} (y + \beta)^3 + \sum_{y \in A} (y + \beta)^3 + \left(\sum_{y \in V} y + \sum_{y \in B} y \right)^3 = \sum_{y \in U} y^3 + \beta \sum_{y \in U} y^2 + \beta^2 \sum_{y \in U} y +$$
   $$\beta^3 card(U) + \sum_{y \in A} y^3 + \beta \sum_{y \in A} y^2 + \beta^2 \sum_{y \in A} y + \beta^3 card(A) +$$
   $$\left(\sum_{y \in V} y \right)^3 + \left(\sum_{y \in V} y \right)^2 \left(\sum_{y \in B} y \right) + \left(\sum_{y \in V} y \right) \left(\sum_{y \in B} y \right)^2 + \left(\sum_{y \in B} y \right)^3 .$$
   Dar $\beta = \sum_{y \in U} y = \sum_{y \in V} y$, $\left(\sum_{y \in V} y \right)^2 = \sum_{y \in V} y^2$, iar $card(U)$, $card(V)$ sunt pare. Deci expresia de sus se reduce la
   $$\sum_{y \in V} y^3 + \sum_{y \in B} y^3 = \sum_{y \in V \Delta B} y^3.$$

Definiţia 15.2 *Un cod C este invariant la distantă dacă pentru orice $c_1, c_2 \in C$, are loc relaţia*

$$\forall i \; 0 \leq i \leq n, \quad card(\{\alpha | d(\alpha, c_1) = i\}) = card(\{\alpha | d(\alpha, c_2) = i\}).$$

Exemple simple de coduri invariante la distanţă sunt codurile liniare.

Ca o consecinţă imediată, pentru un cod invariant la distanţă care conţine cuvântul nul, distanţa minimă este ponderea minimă a unui cuvânt - cod nenul.

Vom arăta în continuare că un cod Preparata are această proprietate.

Lema 15.2 *$P(r)$ este invariant la distanţă.*

Demonstraţie: Fie $[\chi(U), \chi(V)], [\chi(A), \chi(B)] \in P(r)$ arbitrare aflate la distanţa i unul de altul. Conform Lemei 15.1, $[\chi(U \Delta U + \beta), \chi(V \Delta V)]$, $[\chi(U \Delta A + \beta), \chi(V \Delta B)]$ sunt de asemenea cuvinte - cod şi se verifică uşor că ele se află la distanţa i. Deoarece $U \Delta U = \emptyset$ rezultă $[\chi(U \Delta U + \beta), \chi(V \Delta V)] = \mathbf{0}$, deci $[\chi(U \Delta A) + \beta), \chi(V \Delta B)]$ are pondere i. $\square$

Lema 15.3 *Fie* $[\chi(U), \chi(V)] \in P(r)$. *Atunci* $P(r)$ *mai conţine următoarele cuvinte:*
(i). $[\chi(V), \chi(U)]$;
(ii). $[\chi(U+\beta), \chi(V+\beta)]$ *pentru orice* $\beta \in GF(2^r)$;
(iii). $[\chi(\beta U), \chi(\beta V)]$ *pentru orice* $\beta \in GF(2^r)$, $\beta \neq \mathbf{0}$.

Demonstraţie: Se rezolvă similar demonstraţiei de la Lema 15.1. □

Exemplul 15.4 *Conform Exemplului 17.3,* $[\chi(U), \chi(V)] \in P(3)$, *unde*
$U = \{\alpha, \alpha^2, \alpha^5, \mathbf{0}\}$, $V = \{\alpha^0, \alpha, \alpha^2, \alpha^3, \alpha^6, \mathbf{0}\}$.
Conform Lemei 15.3 în care se ia $\beta = \alpha^3$, *vom mai găsi drept cuvinte - cod şi pe*
$[\chi(V), \chi(U)] = 11110011\ \ 01100101$,
$[\chi(U+\beta), \chi(V+\beta)] = [\chi(\{\alpha^0, \alpha^5, \alpha^2, \alpha^3\}), \chi(\{\alpha, \alpha^0, \alpha^5, \mathbf{0}, \alpha^4, \alpha^3\})] =$
$= 10110100\ \ 11011101$,
$[\chi(\beta U), \chi(\beta V)] = [\chi(\{\alpha^4, \alpha^5, \alpha, \mathbf{0}\}), \chi(\{\alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^2, \mathbf{0}\})] =$
$= 01001101\ \ 00111111$.

Putem folosi Lema 15.3 pentru a simplifica problema determinării distanţei mini-me a lui $P(r)$. Pentru aceasta însă, este nevoie de un rezultat suplimentar (care justifică condiţia ca r să fie impar).

Lema 15.4 *Fie* $\alpha \in GF(2^r)$ *primitiv. Atunci* α^3 *este primitiv dacă* r *este impar şi nu este primitiv dacă* r *este par.*

Demonstraţie: Se ştie că α^i este primitiv dacă şi numai dacă $(i, 2^r - 1) = 1$. Avem $2^r - 1 = (3-1)^r - 1 = \mathcal{M}3 + (-1)^r - 1$; deci pentru r impar $2^r - 1 = \mathcal{M}3 - 2 = \mathcal{M}3 + 1$, ceea ce înseamnă că α^3 este primitiv. Similar, pentru r par, $2^r - 1 = \mathcal{M}3$, deci α^3 nu este primitiv. □

Corolarul 15.1 *Dacă* $r > 2$ *este un număr impar, atunci pentru orice* $x \in GF(2^r) \setminus \{\mathbf{0}\}$ *există* y *unic (numit rădăcina cubică a lui* x*) astfel ca* $y^3 = x$.

Teorema 15.1 $P(r)$ *are distanţa minimă* $t = 6$.

Demonstraţie: Deoarece $P(r)$ este invariant la distanţă, el va conţine un cuvânt - cod $[\chi(U), \chi(V)]$ de pondere t. Deci

$$t = w(\chi(U)) + w(\chi(V)) = card(U) + card(V).$$

Rezultă că t este număr par; mai trebuie verificat că $t \neq 2$, $t \neq 4$ şi există un cuvânt - cod de pondere 6.

Să presupunem $t = 2$; atunci $card(U) = 2$, $card(V) = 0$ (Lema 15.3 asigură că totul se poate reduce la acest caz). Folosind tot Lema 15.3, (ii), putem presupune că $U = \{\mathbf{0}, \mathbf{x}\}$ cu $\mathbf{x} \neq \mathbf{0}$. Dar atunci $\sum_{u \in U} u = \mathbf{0} + \mathbf{x} = \mathbf{x}$, ceea ce duce la o contrazicere a definiţiei codurilor Preparata, pentru că $V = \emptyset$.

Să presupunem $t = 4$. Cu Lema 15.3, (i) aceasta înseamnă $card(U) = 4$, $card(V) = 0$ sau $card(U) = card(V) = 2$. În primul caz avem $U = \{\mathbf{0}, \mathbf{x}, \mathbf{y}, \mathbf{z}\}$ cu $\mathbf{x}, \mathbf{y}, \mathbf{z} \neq \mathbf{0}$ şi distincte. Ultima condiţie a definiţiei codurilor Preparata este:
$\mathbf{0^3} + \mathbf{x^3} + \mathbf{y^3} + \mathbf{z^3} + (\mathbf{0} + \mathbf{x} + \mathbf{y} + \mathbf{z})^{\mathbf{3}} = \mathbf{0}$ sau $(\mathbf{x} + \mathbf{y})(\mathbf{x} + \mathbf{z})(\mathbf{y} + \mathbf{z}) = \mathbf{0}$
ceea ce este imposibil, cele trei numere fiind distincte şi nenule.

În a doua variantă, se poate considera $U = \{\mathbf{0}, \mathbf{x}\}$, $V = \{\mathbf{y}, \mathbf{z}\}$, toate cele patru elemente fiind distincte. Din definiţie rezultă

$$\mathbf{0^3} + \mathbf{x^3} + (\mathbf{0} + \mathbf{x})^\mathbf{3} = \mathbf{y^3} + \mathbf{z^3} \text{ sau } \mathbf{y^3} + \mathbf{z^3} = \mathbf{0}.$$

Folosind acum Corolarul 15.1, din $\mathbf{y^3} = \mathbf{z^3}$ rezultă $\mathbf{y} = \mathbf{z}$, contradicţie.

Să construim acum un cuvânt - cod de pondere $t = 6$. Fie $\mathbf{x}, \mathbf{y}, \mathbf{z} \in GF(2^r)$ elemente distincte nenule. Există atunci (Corolarul 15.1) un element $\mathbf{v} \in GF(2^r)$ unic cu $\mathbf{v^3} = \mathbf{x^3} + \mathbf{y^3} + \mathbf{z^3}$. Dacă $\mathbf{v} = \mathbf{x}$, atunci $\mathbf{v^3} = \mathbf{x^3}$ deci $\mathbf{y^3} = \mathbf{z^3}$, ceea ce duce la contradicţia $\mathbf{y} = \mathbf{z}$. Deci $\mathbf{v}$ este distinct de $\mathbf{x}, \mathbf{y}, \mathbf{z}$.

Definim $\mathbf{u} = \mathbf{v} + \mathbf{x} + \mathbf{y} + \mathbf{z}$. Se observă că $\mathbf{u} \neq \mathbf{0}$ (altfel

$$\mathbf{v^3} + (\mathbf{x} + \mathbf{y} + \mathbf{z})^\mathbf{3} = (\mathbf{x} + \mathbf{y})(\mathbf{x} + \mathbf{z})(\mathbf{y} + \mathbf{z}) \neq \mathbf{0}$$

deci - cu Corolarul 15.1 - $\mathbf{v} \neq \mathbf{x} + \mathbf{y} + \mathbf{z}$). Fie acum $U = \{\mathbf{0}, \mathbf{u}\}$, $V = \{\mathbf{v}, \mathbf{x}, \mathbf{y}, \mathbf{z}\}$. Din construcţie, cum toate elementele sunt distincte, $[\chi(U), \chi(V)]$ este un cuvânt de pondere 6 şi se verifică uşor că este cuvânt - cod. □

Exemplul 15.5 *Să considerăm $GF(2^3)$ construit anterior. Fie elementele* $\mathbf{x} = \alpha$, $\mathbf{y} = \alpha^3$, $\mathbf{z} = \alpha^5$. *Definim*

$\mathbf{v^3} = \mathbf{x^3} + \mathbf{y^3} + \mathbf{z^3} = \alpha^3 + \alpha^9 + \alpha^{15} = 110 + 001 + 100 = \alpha^4 = (\alpha^6)^3$

(deoarece $\alpha^7 = \mathbf{1}$), deci se poate lua $\mathbf{v} = \alpha^6$. Mai departe,

$\mathbf{u} = \mathbf{v} + \mathbf{x} + \mathbf{y} + \mathbf{z} = \alpha^6 + \alpha + \alpha^3 + \alpha^5 = \alpha^4$. *Acum,* $U = \{\mathbf{0}, \mathbf{u}\} = \{\mathbf{0}, \alpha^4\}$, $V = \{\mathbf{v}, \mathbf{x}, \mathbf{y}, \mathbf{z}\} = \{\alpha^6, \alpha, \alpha^3, \alpha^5\}$ *şi se obţine* $[\chi(U), \chi(V)] =$ 00001001 01010110 *care este un cuvânt cod din $P(3)$ de pondere* 6.

Teorema 15.2 *Codul Preparata nu este cod liniar.*

Demonstraţie: După cum se ştie, $[\chi(U), \chi(V)] + [\chi(A), \chi(B)] = [\chi(U \Delta A), \chi(V \Delta B)]$. Din demonstraţia Teoremei 20.5 am văzut cum se pot construi cuvintele - cod $[\chi(U), \chi(V)]$, $[\chi(A), \chi(B)] \in P(r)$ cu $U = \{\mathbf{0}, \mathbf{u_1}\}$, $V = \{\mathbf{x_1}, \mathbf{y_1}, \mathbf{z_1}, \mathbf{v_1}\}$,
$A = \{\mathbf{0}, \mathbf{u_2}\}$, $B = \{\mathbf{x_2}, \mathbf{y_2}, \mathbf{z_2}, \mathbf{v_2}\}$. Conform Lemei 15.1,

$\mathbf{c} = [\chi(U \Delta A + \mathbf{u_1}), \chi(V \Delta B)] \in P(r)$. Facem următoarele observaţii:

$card(U \Delta A + \mathbf{u_1}) \leq 2$;

$d(\mathbf{c}, [\chi(U \Delta A), \chi(V \Delta B)]) \leq 2 \cdot card(U \Delta A + \mathbf{u_1}) \leq 4$;

$P(r)$ are distanţa minimă 6.

Din ele rezultă că $[\chi(U), \chi(V)] + [\chi(A), \chi(B)] \notin P(r)$.

Deci $P(r)$ nu este cod liniar. □

Ca o consecinţă a acestei teoreme, nu se poate da nici o dimensiune pentru $P(r)$, deci se pare că nu se poate determina numărul de cuvinte - cod. Vom reuşi totuşi acest lucru ca o consecinţă a schemei de codificare.

15.2 Codificarea codurilor Preparata extinse

Fie $\alpha \in GF(2^r)$ primitiv şi să considerăm codul BCH de polinom generator $g(X) = m_1(X)m_3(X)$ unde $m_i(X)$ este polinomul minimal al lui α^i $(i = 1, 3)$. După cum se ştie, $grad(m_1(X)) = grad(m_3(X)) = r$, deci $grad(g(X)) = 2r$. Codul BCH astfel definit poate corecta maxim 2 erori şi are matricea de control (transpusă):

$$H = \begin{pmatrix} \alpha^0 & \alpha^0 \\ \alpha^1 & \alpha^3 \\ \alpha^2 & \alpha^6 \\ & \vdots \\ \alpha^{2^r-2} & \alpha^{3(2^r-2)} \end{pmatrix}$$

Deoarece $g(X)$ generează un cod ciclic, rezultă că orice submatrice a lui H formată din $2r$ linii consecutive este nesingulară, deci inversabilă. Se defineşte matricea A ca submatrice a lui H formată din ultimele $2r$ linii, şi H' matricea rămasă prin ştergerea lui A.

Exemplul 15.6 *În extensia $GF(2^3)$ generată de rădăcina α a polinomului $1 + X + X^3$ vom avea*

$$A = \begin{pmatrix} \alpha & \alpha^3 \\ \alpha^2 & \alpha^6 \\ \alpha^3 & \alpha^2 \\ \alpha^4 & \alpha^5 \\ \alpha^5 & \alpha \\ \alpha^6 & \alpha^4 \end{pmatrix} = \begin{pmatrix} 010 & 110 \\ 001 & 101 \\ 110 & 001 \\ 011 & 111 \\ 111 & 010 \\ 101 & 011 \end{pmatrix} \quad cu \quad A^{-1} = \begin{pmatrix} 001 & 011 \\ 111 & 010 \\ 011 & 101 \\ 110 & 100 \\ 101 & 110 \\ 111 & 001 \end{pmatrix}.$$

Pentru $GF(2^5)$ generată de rădăcina α a polinomului $1 + X^2 + X^5$ avem

$$A = \begin{pmatrix} \alpha^{21} & \alpha^{63} \\ \alpha^{22} & \alpha^{66} \\ \vdots & \vdots \\ \alpha^{30} & \alpha^{90} \end{pmatrix} = \begin{pmatrix} 00011 & 01000 \\ 10101 & 00001 \\ 11110 & 00101 \\ 01111 & 10001 \\ 10011 & 00111 \\ 11101 & 11011 \\ 11010 & 01100 \\ 01101 & 10101 \\ 10010 & 10011 \\ 01001 & 01101 \end{pmatrix} \quad cu \quad A^{-1} = \begin{pmatrix} 00111 & 00010 \\ 00011 & 10001 \\ 10011 & 00011 \\ 11011 & 01010 \\ 01101 & 10101 \\ 10101 & 11001 \\ 00110 & 11111 \\ 11001 & 01110 \\ 11000 & 00111 \\ 10001 & 10100 \end{pmatrix}$$

Fie $\mathbf{a} = [\mathbf{a_L}, \mathbf{a_R}]$ un cuvânt binar oarecare de lungime $2^{r+1} - 2r - 2$, unde $|\mathbf{a_L}| = 2^r - 1$, $|\mathbf{a_R}| = 2^r - 2r - 1$. În notaţie polinomială putem scrie

$$\mathbf{a_L}H = [a_L(\alpha), a_L(\alpha^3)], \quad \mathbf{a_R}H' = [a_R(\alpha), a_R(\alpha^3)].$$

Se mai defineşte

$$\mathbf{v_R} = [a_L(\alpha) + a_R(\alpha), a_L(\alpha^3) + (a_L(\alpha))^3 + a_R(\alpha^3)]A^{-1}. \tag{2}$$

Teorema 15.3 *Fie r un număr impar. Pentru orice cuvânt binar $\mathbf{a}$ de lungime $2^{r+1} - 2r - 2$, dacă $\chi(U) = [\mathbf{a_L}, p_L]$, $\chi(V) = [\mathbf{a_R}, \mathbf{v_R}, p_R]$ unde p_L, p_R sunt biţii de control pentru $\mathbf{a_L}$ respectiv $[\mathbf{a_R}, \mathbf{v_R}]$, atunci $[\chi(U), \chi(V)] \in P(r)$.*

Demonstraţie: $[\mathbf{a_R}, \mathbf{v_R}]H = \mathbf{a_R}H' + \mathbf{v_R}A = [a_R(\alpha), a_R(\alpha^3)] + [a_L(\alpha) + a_R(\alpha), a_L(\alpha^3) + (a_L(\alpha))^3 + a_R(\alpha^3)] = [a_L(\alpha), a_L(\alpha^3) + (a_L(\alpha))^3]$.

Dar $[\mathbf{a_R}, \mathbf{v_R}] = [\sum_{v \in V} v, \sum_{v \in V} v^3]$, $a_L(\alpha) = \sum_{u \in U} u$, $a_L(\alpha^3) + (a_L(\alpha))^3 = \sum_{u \in U} u^3 + \left(\sum_{u \in U} u\right)^3$, ceea ce verifică condiţiile din Definiţia 15.1.

Deci $[\chi(U), \chi(V)]$ este cuvânt - cod în $P(r)$. □

Corolarul 15.2 *$P(r)$ are $2^{2^{r+1}-2r-2}$ cuvinte - cod.*

Demonstraţie: În Teorema 20.3 există $2^{2^{r+1}-2r-2}$ alegeri posibile pentru $\mathbf{a}$, toate conducând la cuvinte - cod distincte. □

Putem da acum un algoritm de codificare a meajelor de informaţie în codurile Preparata.

Fie $\mathbf{a_L}, \mathbf{a_R}$ cuvinte de lungimi $2^r - 1$ respectiv $2^r - 2r - 1$.
Fie $\mathbf{v_R}$ definit de (2).
Se construiesc p_L, p_R conform Teoremei 20.3.
Atunci mesajul de informaţie $\mathbf{a} = [\mathbf{a_L}, \mathbf{a_R}]$ se codifică
în $[\mathbf{a_L}, \mathbf{p_L}, \mathbf{a_R}, \mathbf{v_R}, \mathbf{p_R}]$.

Exemplul 15.7 *Să considerăm $r = 3$, $\mathbf{a_L} = 0110010$, $\mathbf{a_R} = 1$. Deci*

$$a_L(\alpha) = \alpha + \alpha^2 + \alpha^5 = \alpha^0, \quad a_R(\alpha) = \alpha^0,$$
$$a_L(\alpha^3) = \alpha^3 + \alpha^6 + \alpha^{15} = \alpha^2, \quad a_R(\alpha^3) = \alpha^0.$$

Din (2) *avem* $\mathbf{v_R} = [\alpha^0 + \alpha^0, \alpha^2 + \alpha^0 + \alpha^0]A^{-1} = [000\ 001]A^{-1} = 111001$ *unde* A^{-1} *este matricea din Exemplul 18.5. Atunci vom codifica* $\mathbf{a} = [0110010\ 1]$ *în* $\mathbf{c} = [\mathbf{a_L}, p_L, \mathbf{a_R}, \mathbf{v_R}, p_R] =$ $[01100010\ 1\ 1\ 111001\ 1]$.

Deci $\mathbf{c} = [\chi(U), \chi(V)]$ *unde* $\chi(U) = 01100101$, $\chi(V) = 11110011$, *care coincide cu secvenţa - cod construită în Exemplul 17.3.*

15.3 Decodificarea codurilor Preparata extinse

Cum $P(r)$ are distanţa minimă 6, el poate corecta maxim 2 erori.

Fie $\mathbf{b}$ un cuvânt recepţionat; îl scriem $\mathbf{b} = [\mathbf{b_L}, p_L, \mathbf{b_R}, p_R]$ unde $\mathbf{b_L}$, $\mathbf{b_R}$ sunt ambele de lungime $2^r - 1$ iar p_L, p_R sunt biţi de control a parităţii. Se calculează

$$\mathbf{b_L}H = [b_L(\alpha), b_L(\alpha^3)], \quad \mathbf{b_R}H = [b_R(\alpha), b_R(\alpha^3)].$$

Apar mai multe cazuri, în funcţie de poziţiile unde apar erori.

1. Dacă erorile apar pe poziţiile de control, atunci

   $$b_L(\alpha) = b_R(\alpha), \quad b_L(\alpha^3) + (b_L(\alpha))^3 = b_R(\alpha^3)$$

   (conform Definiţiei 15.1), ceea ce se verifică uşor.

2. Dacă există o eroare pe poziţia i în $\mathbf{b_R}$ şi cel mult o eroare pe poziţiile de control atunci

   $b_L(\alpha) = b_R(\alpha) + \alpha^i$, $\quad b_L(\alpha^3) + (b_L(\alpha))^3 = b_R(\alpha^3) + \alpha^{3i}$, deci

   $$(b_L(\alpha) + b_R(\alpha))^3 = b_L(\alpha^3) + (b_L(\alpha))^3 + b_R(\alpha^3).$$

   Dacă ultima relaţie este verificată, atunci se scrie $\alpha^i = b_L(\alpha) + b_R(\alpha)$ şi se schimbă bitul i din $\mathbf{b_R}$ (plus cel mult un bit de control).

3. Dacă există o eroare pe poziţia i din $\mathbf{b_L}$ şi cel mult o eroare pe poziţiile de control, similar cu cazul anterior (lucru posibil pe baza Lemei 15.3) se verifică relaţia

   $$(b_R(\alpha) + b_L(\alpha))^3 = b_R(\alpha^3) + (b_R(\alpha))^3 + b_L(\alpha^3);$$

   în caz afirmativ, se calculează $\alpha^i = b_R(\alpha) + b_L(\alpha)$ şi se schimbă bitul i din $\mathbf{b_L}$ (plus cel mult un bit de control).

4. Dacă apar două erori în $\mathbf{b_R}$ pe poziţiile i şi j, atunci:

   $b_L(\alpha) = b_R(\alpha) + \alpha^i + \alpha^j, \quad b_L(\alpha^3) + (b_L(\alpha))^3 = b_R(\alpha^3) + \alpha^{3i} + \alpha^{3j},$

   deci se determină $\alpha^i + \alpha^j$ şi $\alpha^{3i} + \alpha^{3j}$. Valorile $i,\ j$ sunt determinate cu un algoritm similar celui de la codurile BCH.

5. Dacă apar două erori în $\mathbf{b_L}$, invocând din nou Lema 15.3 putem aplica analiza de la cazul anterior.

6. Dacă apar două erori: una pe poziţia i în $\mathbf{b_L}$ şi una pe poziţia j în $\mathbf{b_R}$, atunci

   $b_L(\alpha) + \alpha^i = b_R(\alpha) + \alpha^j, \quad b_L(\alpha^3) + \alpha^{3i} + (b_L(\alpha) + \alpha^i)^3 = b_R(\alpha^3) + \alpha^{3j}.$

   Acest sistem de necunoscute α^i şi α^j se rezolvă astfel: din prima ecuaţie se scoate $\alpha^j = b_L(\alpha) + \alpha^i + b_R(\alpha)$ şi se înlocuieşte în a doua ecuaţie, ceea ce dă

   $b_L(\alpha^3) + \alpha^{3i} + (b_L(\alpha) + \alpha^i)^3 = b_R(\alpha^3) + (b_L(\alpha) + \alpha^i)^3 + (b_L(\alpha) + \alpha^i)^2 b_R(\alpha) + (b_L(\alpha) + \alpha^i) b_R(\alpha)^2 + b_R(\alpha)^3.$

   După simplificări se ajunge la

   $\alpha^{3i} + \alpha^{2i} b_R(\alpha) + \alpha^i (b_R(\alpha))^2 + (b_R(\alpha))^3 = b_L(\alpha^3) + b_R(\alpha^3) + b_L(\alpha)^2 b_R(\alpha) + b_L(\alpha) b_R(\alpha)^2,$

   sau

   $(\alpha^i + b_R(\alpha))^3 = (b_L(\alpha^3) + b_R(\alpha^3)) + (b_L(\alpha) + b_R(\alpha))^3 + b_L(\alpha)^3 + b_R(\alpha)^3.$

   Notând membrul drept al acestei expresii cu Δ, obţinem soluţia

   $$\alpha^i = b_R(\alpha) + \Delta^{1/3}, \quad \alpha^j = b_L(\alpha) + \Delta^{1/3}.$$

Deci locaţiile erorilor se pot determina în toate situaţiile posibile. Biţii de paritate pentru fiecare jumătate de cuvânt dau posibilitatea de a decide ce caz se aplică lui $\mathbf{b}$. Putem da acum algoritmul de decodificare pentru codurile Preparata $P(r)$.

Fie $\mathbf{b} = [\mathbf{b_L}, p_L, \mathbf{b_R}, p_R]$ cuvântul recepţionat.

0. Se calculează $L_1 = b_L(\alpha),\ L_3 = b_L(\alpha^3),\ R_1 = b_R(\alpha),\ R_3 = b_R(\alpha^3)$;

1. Dacă $L_1 + R_1 = 0$ şi $L_3 + L_1^3 + R_3 = 0$, atunci singurele erori posibile au apărut pe poziţiile de control.

2. Dacă $(L_1+R_1)^3+L_3+L_1^3+R_3 = 0$, calculăm $\alpha^i = L_1+R_1$. Se corectează poziţia i din $\mathbf{b_R}$ şi cel mult un bit de control al parităţii; se cere retransmisia dacă ambii biţi de paritate trebuie modificaţi.

3. Dacă $(L_1+R_1)^3+R_3+R_1^3+L_3 = 0$, calculăm $\alpha^i = L_1+R_1$. Se corectează poziţia i din $\mathbf{b_L}$ şi cel mult un bit de control al parităţii; se cere retransmisia dacă ambii biţi de paritate trebuie modificaţi.

4. Dacă ambele jumătăţi ale lui $\mathbf{b}$ sunt de pondere pară şi
$$X^2 + (L_1+R_1)X + \frac{L_3 + L_1^3 + R_3 + (L_1+R_1)^3}{L_1+R_1} = (X+\alpha^i)(X+\alpha^j),$$
se corectează poziţiile i şi j din $\mathbf{b_L}$.

5. Dacă ambele jumătăţi ale lui $\mathbf{b}$ sunt de pondere pară şi
$$X^2 + (L_1+R_1)X + \frac{R_3 + R_1^3 + R_3 + (L_1+R_1)^3}{L_1+R_1} = (X+\alpha^i)(X+\alpha^j),$$
se corectează poziţiile i şi j din $\mathbf{b_R}$.

6. Dacă ambele jumătăţi ale lui $\mathbf{b}$ sunt de pondere impară, se calculează
$$\alpha^i = R_1 + (L_1^3 + R_1^3 + (L_1+R_1)^3 + L_3 + R_3)^{1/3}$$
$$\alpha^j = L_1 + (L_1^3 + R_1^3 + (L_1+R_1)^3 + L_3 + R_3)^{1/3}$$
Se corectează poziţia i din $\mathbf{b_L}$ şi poziţia j din $\mathbf{b_R}$.

7. Dacă nu a apărut nici una din situaţiile anterioare, se cere retransmisia cuvântului.

Exemplul 15.8 *Să decodificăm următoarele cuvinte primite, despre care se presupune că au fost codificate folosind $P(3)$, unde $GF(2^3)$ s-a generat cu rădăcina α a polinomului $1+X+X^3$:*

I: $\mathbf{b} = 10010011\ \ 11100111$

(0) $[L_1, L_3] = \mathbf{b_L}H = 111\ 110, \quad [R_1, R_3] = \mathbf{b_R}H = 101\ 110$

(1) $L_1 + R_1 = 111 + 101 = \alpha \neq 0$

(2) $(L_1+R_1)^3 + L_3 + L_1^3 + R_3 = \alpha^3 + \alpha^3 + \alpha^{15} + \alpha^3 = \alpha^0 \neq 0$

(3) $(L_1+R_1)^3 + R_3 + R_1^3 + L_1 = \alpha^3 + \alpha^3 + \alpha^{18} + \alpha^3 = \alpha^6 \neq 0$

(4) $X^2 + \alpha X + \dfrac{\alpha^3 + \alpha^{15} + \alpha^3 + \alpha^3}{\alpha} = X^2 + \alpha X + \alpha^6 = (X+\alpha^2)(X+\alpha^4)$

Deci $\mathbf{b}$ *se decodifică în* 10010011 11001111.

II: $\mathbf{b} = 10100100\ \ 10001001$

(0) $[L_1, L_3] = \mathbf{b_L}H = [010, 011], \quad [R_1, R_3] = \mathbf{b_R}H = [111, 011]$

(1) $L_1 + R_1 = 010 + 111 = \alpha^6 \neq 0$

(2) $(L_1+R_1)^3 + L_3 + L_1^3 + R_3 = \alpha^{18} + \alpha^4 + \alpha^3 + \alpha^4 = \alpha^6 \neq 0$

(3) $(L_1+R_1)^3 + R_3 + R_1^3 + L_3 = \alpha^{18} + \alpha^4 + \alpha^{15} + \alpha^4 = \alpha^2 \neq 0$

Ambele jumătăţi ale lui $\mathbf{b}$ *au pondere impară, deci*

(6) $\alpha^i = \alpha^5 + (\alpha^3+\alpha^{15}+\alpha^{18}+\alpha^4+\alpha^4)^{1/3} = \alpha^5 + (\alpha^5)^{1/3} = \alpha^5 + (\alpha^{12})^{1/3} = \alpha^5 + \alpha^4 = \alpha^0$.

S-a obţinut $i = 0$. *Se poate determina apoi imediat*

$\alpha^j = \alpha + \alpha^4 = \alpha^2$, *deci* $j = 2$.

Cuvântul $\mathbf{b}$ *se decodifică în* 00100100 10101001.

III: $\mathbf{b} = 10001000\ \ 11101001$

(0) $[l_1, L_3] = \mathbf{b_L}H = [111, 011], \quad [R_1, R_3] = \mathbf{b_R}H = [100, 000]$

(1) $L_1 + R_1 = 111 + 100 = \alpha^4 \neq 0$

(2) $(L_1 + R_1)^3 + L_3 + L_1^3 + R_3 = \alpha^{12} + \alpha^4 + \alpha^{15} + \mathbf{0} = \alpha^3 \neq 0$

(3) $(L_1 + R_1)^3 + R_3 + R_1^3 + L_3 = \alpha^{12} + \mathbf{0} + \alpha^0 + \alpha^4 = \mathbf{0}$

Calculăm $\alpha^i = L_1 + R_1 = \alpha^4$, *deci* $i = 4$. *Dar modificarea bitului* 4 *în* $\mathbf{b_L}$ *cere ca ambii biţi de control să fie modificaţi; deci se cere retransmisia cuvântului.*

Pentru codul Preparata obţinut prin relaxarea lui $P(r)$, se poate aplica aceeaşi strategie de decodificare folosită la codul Golay; cum acesta are $d = 5$, poate corecta de asemenea cel mult 2 erori independente.

15.4 Coduri înrudite cu codul Preparata

15.4.1 Codul NHordstrom - Robinson

Să plecăm de la codul Golay binar extins C_{24} ale cărui cuvinte le scriem sub forma $\mathbf{a} = [\mathbf{a_1}, \mathbf{a_2}]$ unde $|\mathbf{a_1}| = 8,\ |\mathbf{a_2}| = 16$. Considerăm submulţimea $GNR \subset C_{24}$ ale cărei elemente verifică condiţiile:

1. Dacă $\mathbf{a} \in GNR$ atunci $w(\mathbf{a_1}) \in \{0, 2\}$.
2. Dacă $\mathbf{a}, \mathbf{b} \in GNR$ atunci $d(\mathbf{a_1}, \mathbf{b_1}) \leq 2$.

Evident, GNR va avea 256 cuvinte. Pe baza lui se construieşte codul

$$NR = \{\mathbf{a_2} | \mathbf{a} = [\mathbf{a_1}, \mathbf{a_2}] \in GNR\}.$$

Acesta este un cod neliniar de lungime $n = 16$ şi distanţă minimă $d = 6$. Numit *codul NHostrom - Robinson*, el coincide cu codul Preparata $P(3)$.

Un alt cod derivat din acesta este codul *Nadler*, construit astfel: se păstrează din NR doar cuvintele - cod care coincid pe ultimele două poziţii. Apoi la toate aceste cuvinte se şterg ultimele trei caractere (o dublă scurtare urmată de o relaxare). Codul astfel obţinut este unic, are $n = 13,\ d = 5$ şi 64 cuvinte - cod.

15.4.2 Codurile Kerdock

Definiţia 15.3 *Fie* $r \geq 3$, r *impar şi* $U, V \subset GF(2^r)$, $card(U), card(V)$ *pare.*

Un cod Kerdock $K(r)$ *este format din toate cuvintele de forma* $[U, V]$ *care verifică condiţiile:*

1. $\forall s\ (1 \leq s \leq 2^r - 2)\ (1 \leq w_2(s) < r - 2)\ \Rightarrow \quad \sum_{u \in U} u^s = \sum_{v \in V} v^s = \mathbf{0}$;

2. $\forall s\ (\leq s \leq 2^r - 2)\ (w_2(s) = r - 2)\ \Rightarrow \sum_{u \in U} u^s = \sum_{v \in V} v^s = \left(\sum_{u \in U} u^{-1} + \sum_{v \in V} v^{-1}\right)^{-s}$

unde $w_2(s)$ *reprezintă ponderea reprezentării în binar a lui* s.

Se consideră prin convenţie $\forall s,\ \mathbf{0^{-s}} = \mathbf{0}$.

Codul Kerdok $K(r)$ (definit în 1972) este de fapt un subcod al codului Reed - Muller $\mathcal{RM}(2, r+1)$; el este format din perechi de translatări de cuvinte din $\mathcal{RM}(1, r)$ selectate în aşa fel ca să maximizeze distanţa minimă dintre două translatări.

Este interesant că dintre toate codurile cu aceeaşi distanţă minimă, codul Kerdok are numărul maxim de cuvinte - cod.

Nu vom detalia demonstraţii referitoare la proprietăţile acestor coduri. Pentru informaţii suplimentare se pot folosi [9] şi [4], cu menţiunea că justificările pleacă de la o modalitate diferită de definire a codurilor $K(r)$ şi $P(r)$. Rezultatele sunt însă deosebit de interesante. Astfel:

Teorema 15.4 *$K(r)$ este invariant la distanţă.*

Lema 15.5 $K(3) = P(3)$.

Teorema 15.5 *Codurile $P(r)$ şi $K(r)$ sunt duale.*

Ca o consecinţă imediată, $P(3)$ este un cod auto-dual.

15.5 Exerciţii

15.1 *Demonstraţi relaţiile* (1).

15.2 *Demonstraţi Lema 15.3.*

15.3 *Demonstraţi Corolarul 15.1.*

15.4 *Aplicaţi Lema 15.3 cuvântului - cod definit în Exemplul 18.7, folosind*

$$\beta = \alpha^0, \qquad \beta = \alpha, \qquad \beta = \alpha^6$$

15.5 *De ce $[\chi(\beta U), \chi(\beta V)]$ nu este cuvânt - cod pentru $\beta = \mathbf{0}$?*

15.6 *Arătaţi că cele trei cuvinte obţinute în Exemplul 18.7 satisfac condiţiile defini-ţiei codurilor Preparata.*

15.7 *Fie $GF(2^3)$ generat de rădăcina α a polinomului $1 + X + X^3 = 0$. Folosind următoarele cuvinte $\mathbf{x}, \mathbf{y}, \mathbf{z} \in GF(2^3)$ construiţi cuvinte - cod de pondere 6 din $P(3)$:*

$$\mathbf{x} = \alpha \quad \mathbf{y} = \alpha^2, \quad \mathbf{z} = \alpha^3,$$
$$\mathbf{x} = \alpha \quad \mathbf{y} = \alpha^4, \quad \mathbf{z} = \alpha^6,$$
$$\mathbf{x} = \alpha^0 \quad \mathbf{y} = \alpha^3, \quad \mathbf{z} = \alpha^6,$$

15.8 *Fie $GF(2^3)$ construit cu $1 + X + X^3$, iar A^{-1} din Exemplul 18.5. Codificaţi următoarele mesaje folosind $P(3)$:*

$$\mathbf{a_L} = 1010100, \quad \mathbf{a_R} = 1;$$
$$\mathbf{a_L} = 1010100, \quad \mathbf{a_R} = 0;$$
$$\mathbf{a_L} = 1111111, \quad \mathbf{a_R} = 1;$$
$$\mathbf{a_L} = 1111111, \quad \mathbf{a_R} = 0;$$
$$\mathbf{a_L} = 0000000, \quad \mathbf{a_R} = 1;$$

15.9 *Fie* $GF(2^5)$ *construit cu* $1 + X^2 + X^5$, *iar* A^{-1} *din Exemplul 18.5. Codificaţi următoarele mesaje folosind* $P(5)$*:*

$$\mathbf{a_L} = 10100\ldots0, \quad \mathbf{a_R} = 000001000100\ldots0;$$
$$\mathbf{a_L} = 10100\ldots0, \quad \mathbf{a_R} = 00\ldots0;$$
$$\mathbf{a_L} = 10100\ldots0, \quad \mathbf{a_R} = 111100\ldots0;$$
$$\mathbf{a_L} = 0000\ldots0, \quad \mathbf{a_R} = 100\ldots0;$$

Care este lungimea lui $\mathbf{a_L}$ *? dar a lui* $\mathbf{a_R}$ *?*

15.10 *În aceleaşi condiţii din Exemplul 18.2, să se decodifice cuvintele:*

10000001	11101000	00011010	01000010	00100101	10100100
01010110	00011110	11101000	10001001	10011001	01010101
01000111	11001000	10101101	11010000	11101110	01010101
10111011	01101010	01011101	11101101	10011100	10100100
01101101	10011000	10101010	10111011	10100101	00010001

15.11 *Decodificaţi cuvintele primite, care au fost codificate cu* $P(5)$*, unde* $GF(2^5)$ *a fost construit folosind polinomul* $1 + X^2 + X^5$*.*

11000110001000000000000001000010 00011110000000000000000110010000
10100000001000000000000000000000 00000100010000000100010101011100

15.12 *Fie* $\mathbf{b}$ *un cuvânt recepţionat în care* $w(\mathbf{b_L})$ *este impar iar* $w(\mathbf{b_R})$ *este par. Poate fi decodificat* $\mathbf{b}$ *la pasul* 2 *al algoritmului într-un cuvânt - cod situat la distanţă cel mult* 2 *?*

Capitolul 16

Coduri convoluţionale

În cazul codurilor prezentate până acum, la codificarea unui mesaj de informaţie se obţine un cuvânt - cod de lungime n, ale cărui caractere nu depind decât de cele k elemente de informaţie curente. Avem de-a face cu un proces de codificare fără memorie, iar aceste coduri sunt numite şi *coduri - bloc*.

P. Elias a introdus în 1965 o clasă de coduri, remarcabile prin siguranţa sporită pe care o asigură în transmiterea informaţiei: **codurile convoluţionale**. În cazul lor, codificarea unui mesaj de lungime k se face ţinând cont şi de codificările mesajelor de informaţie anterioare.

De remarcat că NASA şi Agenţia Spaţială Europeană folosesc o combinaţie între aceste coduri şi codurile Reed - Solomon. Fiecare mesaj de informaţie este codificat iniţial cu un cod RS, apoi cu un cod convoluţional.

16.1 Coduri liniare şi coduri convoluţionale

După cum am văzut, un (n, k) - cod liniar codifică un mesaj de informaţie $\mathbf{u} \in Z_q^k$ într-un cuvânt - cod $\mathbf{v} = \mathbf{u}G$ de lungime n ($\mathbf{v} \in Z_q^n$). Un cod liniar poate fi privit însă şi ca o aplicaţie care transformă mesaje sursă (de orice lungime $i \cdot k,\ i = 1, 2, \ldots$) în mesaje - cod de lungime $i \cdot n$.

Exemplul 16.1 *Codul generat de matricea*

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

defineşte o funcţie astfel:

$$a_0a_1a_2a_3a_4a_5\ldots \quad \longmapsto \quad a_0a_1a_2x_0a_3a_4a_5x_1\ldots$$

unde

$$x_0 = a_0 + a_1 + a_2, \quad x_1 = a_3 + a_4 + a_5, \ldots$$

O astfel de funcţie poate fi exprimată foarte simplu prin polinoame. Astfel, fie $a(X) = a_0+a_1X+a_2X^2+\ldots$ polinomul reprezentând mesajul sursă, şi $u(X) = u_0+u_1X+u_2X^2+\ldots$ mesajul codificat. Atunci un cod liniar este o aplicaţie C definită $C(a(X)) = u(X)$. O astfel de aplicaţie are următoarele proprietăţi:

- C este liniară:
$$C(a(X) + a'(X)) = C(a(X)) + C(a'(X));$$
$$C(ta(X)) = tC(a(X)).$$

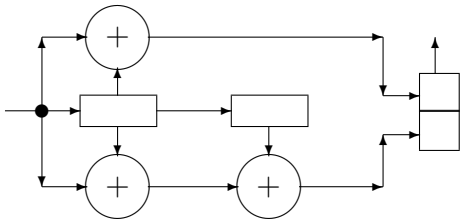
- C este invariantă în timp: întârzierea mesajului sursă cu k tacţi are ca efect decalarea răspunsului cu n tacţi:
$$C(X^k a(X)) = X^n C(a(X)).$$

- C nu are memorie: codificarea unui mesaj sursă de lungime k nu depinde de mesajele sursă precedente.

Aceste trei proprietăt i caracterizează complet codurile liniare. Dacă se renunţă la ultima proprietate, se obţine o nouă clasă de coduri, numite *coduri convoluţionale.*

Definiţia 16.1 *Fie q un număr prim. Prin (n, k) - cod convoluţional se înţelege o aplicaţie liniară $C : Z_q[X] \longrightarrow Z_q[X]$ cu proprietatea (de invarianţă de timp)*
$$C(X^k a(X)) = X^n C(a(X)).$$

Exemplul 16.2 *Să considerăm circuitul liniar:*



El reprezintă un $(2, 1)$ - cod convoluţional binar. Este liniar (datorită circuitului), invariant în timp (o întârziere de 1 tact la intrare provoacă o întârziere de 2 tacţi la ieşire).

De exemplu, pentru intrarea 1 se obţine 11 la primul tact, 11 la al doilea tact şi 01 la al treilea (după care urmează $0000\ldots$). Deci
$$C(1) = 111101 = 1 + X + X^2 + X^3 + X^5.$$
Datorită invarianţei în timp, avem
$$C(X) = C(01) = 00111101, \qquad C(X^2) = C(001) = 0000111101, \ldots$$
(s-a ţinut cont de dualitatea de notare polinom - cuvânt).

Datorită liniarităt ii, răspunsul $C(1)$ caracterizează complet codul. De exemplu, mesajul de intrare 101 se codifică prin

$C(101) = C(1+X^2) = C(1)+C(X^2) = C(1)+C(001) = 11110100\ldots+0000111101\ldots =$ $1111101101 = 1 + X + X^2 + X^3 + X^4 + X^6 + X^7 + X^9$.

16.2 Coduri $(n, 1)$ - convoluţionale

În cazul particular $k = 1$, codul este complet determinat de ieşirea pentru mesajul de intrare 1. Notăm
$$C(1) = g_0(X)$$
polinomul generator al acestei subclase de coduri. Din invarianţa în timp rezultă
$$C(X^i) = X^{ni} g_0(X), \quad \forall i \geq 1.$$

Deci, pentru orice polinom $a(X) = a_0 + a_1X + \ldots + a_pX^p$ avem

$C(a(X)) = a_0C(1) + a_1C(X) + a_2C(X^2) + \ldots + a_pC(X^p) = a_0g_0(X) + a_1X^ng_0(X) + a_2X^{2n}g_0(X) + \ldots + a_pX^{np}g_0(X) = a(X^n)g_0(X)$.

Aceasta conduce la ideea că un $(n, 1)$ - cod convoluţional este generat de polinomul $g_0(X)$ conform regulii

$$a(X) \mapsto a(X^n)g_0(X).$$

Invers, orice polinom $g_0(X) \in Z_q[X]$ defineşte un $(n, 1)$ - cod convoluţional.

Observaţie: Ideea de polinom generator este similară celei de la codurile ciclice, unde codificarea mesajului de informaţie $a(X)$ revenea la înmulţirea cu polinomul generator. La codurile convoluţionale nu există însă restricţie referitor la gradul lui $a(X)$ şi – în plus – la înmulţire se foloseşte $a(X^n)$.

Există o modalitate simplă de construcţie a unui circuit liniar, generator al unui astfel de cod. Vom descrie aceasta pentru cazul binar.

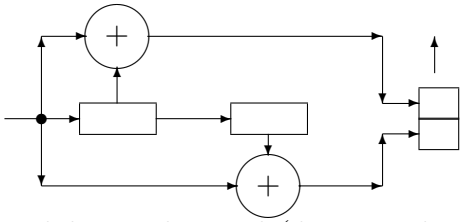
Polinomul $g_0(X)$ se descompune în sume de n termeni consecutivi, fiecare sumă descriind modul de conectare al elementelor de înmagazinare. Astfel, fie

$g_0(X) = (b_0 + b_1X + \ldots + b_{n-1}X^{n-1}) + (b_nX^n + b_{n+1}X^{n+1} + \ldots + b_{2n-1}X^{2n-1}) + \ldots + (b_{mn}X^{mn} + b_{mn+1}X^{mn+1} + \ldots + b_{mn+n-1}X^{mn+n-1})$

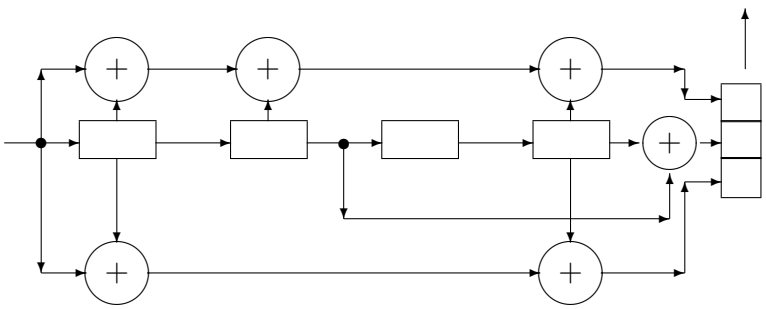
Numărul de $m+1$ paranteze indică faptul că sunt necesare m elemente de înmagazinare (prima paranteză se referă la intrare). A i - a secvenţă de termeni descrie ieşirea celui de-al i - lea element de înmagazinare ($b_p = 1$ semnifică o legătură directă cu al p - lea element al ieşirii).

În plus, elementele de înmagazinare sunt legate între ele secvenţial.

Exemplul 16.3 *Să considerăm* $(2, 1)$ *- codul convoluţional de polinom generator* $g_0(X) = 1+X+X^2+X^5 = (1+X)+(X^2+0)+(0+X^5)$. *Prima paranteză,* $1+X$, *arată că ambii biţi de ieşire sunt legaţi de intrare,* X^2+0 *indică faptul că primul element de înmagazinare este legat direct doar de primul bit de ieşire,* $0+X^5$ *arată că al doilea element de înmagazinare este legat numai de al doilea bit de ieşire. În acest fel se obţine codificatorul descris de circuitul liniar:*



Exemplul 16.4 *Fie* $(3, 1)$ *- codul convoluţional (deci cu* 3 *biţi de ieşire) generat de polinomul* $g_0(X) = 1 + X^2 + X^3 + X^5 + X^6 + X^7 + X^{12} + X^{13} + X^{14}$. *Pentru a construi circuitul liniar, rescriem acest polinom astfel:* $g_0(X) = (1 + 0 + X^2) + (X^3 + 0 + X^5) + (X^6 + X^7 + 0) + (X^{12} + X^{13} + X^{14})$. *Se obţine*



Similar codurilor ciclice, plecând de la polinomul generator, se poate construi matricea generatoare a codurilor $(n,1)$ - convoluţionale. Aceasta este

$$G = \begin{pmatrix} g_0(X) \\ X^n g_0(X) \\ X^{2n} g_0(X) \\ \vdots \end{pmatrix}$$

Numărul liniilor (şi deci şi al coloanelor) lui G nu este fixat, el depinzând de lungimea mesajului sursă. Astfel, pentru un mesaj de lungime i, matricea G are i linii şi $1 + grad(g_0(X)) + (i-1)\cdot n$ coloane.

Exemplul 16.5 *Fie* $(3,1)$ *- codul convoluţional binar de polinom generator* $g_0(X) = 1 + X + X^3 + X^4 + X^5 + X^8$. *Pentru un mesaj sursă de lungime* 3, *matricea* G *va avea dimensiunea* $3'times15$*:*

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

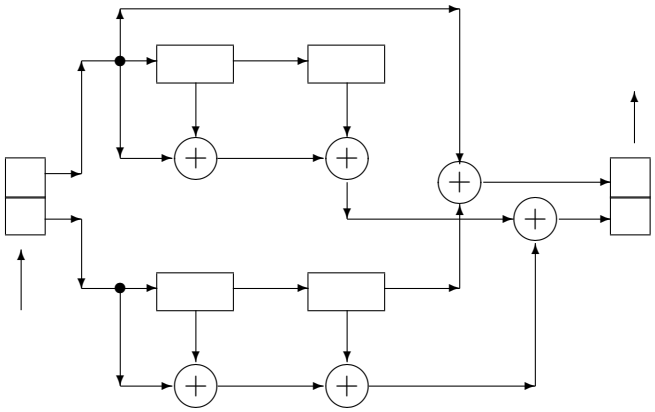
(spaţiile libere sunt completate cu 0*). Astfel, codificarea mesajului* 101 *este*

$$101 \cdot G = 110001000110001.$$

16.3 Coduri (n,k) - convoluţionale

Să dezvoltăm o descriere a codurilor (n,k) - convoluţionale, similară celei date anterior pentru cazul $k = 1$. Vom începe cu un exemplu:

Exemplul 16.6 *Fie circuitele liniare a două* $(2,1)$ *- coduri convoluţionale, care au ieşirea comună:*



La fiecare tact intrarea este formată din doi biţi care formează intrările în cele două circuite liniare. O întârziere cu doi tacţi la intrare generează o întârziere de doi tacţi la ieşire; deci, acesta este un $(2,2)$ *- cod convoluţional.*

Pentru intrarea 10 *ieşirea este* 110101 *(funcţionează doar circuitul superior), iar pentru* 01 *ieşirea este* 010111 *(numai pe baza circuitului inferior). Formal,*

$$C(1) = 1 + X + X^3 + X^5 = g_0(X), \qquad C(X) = X + X^3 + X^4 + X^5 = g_1(X).$$

Folosind invarianţa în timp se pot determina şi alte ieşiri:

$$C(X^{2i}) = X^{2i} g_0(X), \quad C(X^{2i+1}) = X^{2i} g_1(X) \quad \forall i \geq 1.$$

Definiţia 16.2 *Pentru un (n, k) - cod convoluţional, polinoamele*

$$g_i(X) = C(X^i), \quad 0 \leq i \leq k-1$$

se numesc polinoame generatoare.

Teorema 16.1 *Un (n, k) - cod convoluţional de polinoame generatoare $g_i(X)$, $(0 \leq i \leq k-1)$ este determinat de funcţia de codificare*

$$a(X) \mapsto \sum_{i=0}^{k-1} a^{(i)}(X^n) g_i(X) \tag{1}$$

unde pentru secvenţa $\mathbf{a} = a_0 a_1 a_2 \ldots$ s-a notat $\mathbf{a}^{(i)} = a_i a_{i+k} a_{i+2k} \ldots$ $(0 \leq i \leq k-1)$.

Reciproc, fiind date polinoamele $g_i(X)$ $i = 0, 1, \ldots, k-1$ şi un număr n, formula (1) determină un (n, k) - cod convoluţional.

Demonstraţie: Vom începe prin a arăta ultima afirmaţie. Aplicaţia

$$a(X) \mapsto a^{(i)}(X^n) g_i(X)$$

este liniară (ca o compunere de transformări liniare) şi invariantă în timp (deci defineşte un (n, k) - cod convoluţional). Ultima aserţiune rezultă din observaţia că pentru orice secvenţă $\mathbf{a}$,

$$\left(X^k a(X)\right)^{(i)} \mapsto X^n a^{(i)}(X).$$

Deci ieşirea la $X^k a(X)$ prin această aplicaţie este $X^n \sum_{i=0}^{k-1} a^{(i)}(X^n) g_i(X)$.

Să arătăm acum că un (n, k) - cod convoluţional definit de aplicaţia C este identic cu cel dat de $C'(a(X)) = \sum_{i=0}^{k-1} a^{(i)}(X^n) g_i(X)$ unde $g_i(X) = C(X^i)$ sunt polinoamele generatoare. Deoarece C şi C' sunt ambele liniare, este suficient să arătăm că $C(X^r) = C'(X^r)$ pentru orice $r \geq 0$; atunci vom avea

$$C(a_0 + a_1 X + a_2 X^2 + \ldots) = \sum_{r \geq 0} a_r C(X^r) = \sum_{r \geq 0} a_r C'(X^r) = C'(a_0 + a_1 X + \ldots).$$

Deoarece ambele aplicaţii sunt (n, k) invariante în timp, ne putem restrânge la $0 \leq r \leq k-1$. Pentru $\mathbf{a} = 00 \ldots 0100 \ldots$ (cu 1 pe poziţia r), este evident că $\mathbf{a}^{(r)} = 100 \ldots$ şi $\mathbf{a}^{(i)} = 000 \ldots$ pentru $i \neq r$. Deci $C'(X^r) = g_r(X) = C(X^r)$. □

Analog codurilor $(n, 1)$ - convoluţionale, şi în acest caz se poate defini matricea generatoare, care este o reprezentare matricială a relaţiei (1). Liniile matricii sunt $g_0(X), g_1(X), \ldots, g_{k-1}(X), X$ considerate ca polinoame de grad infinit. Pentru mesajul $\mathbf{a} = a_0 a_1 \ldots a_r$ codificarea este $\mathbf{a}G = a_0 g_0(X) + \ldots + a_{k-1} g_{k-1}(X) + X^n[a_k g_0(X) + \ldots + a_{2k-1} g_{k-1}(X)] + X^{2n}[a_{2k} g_0(X) + \ldots + a_{3k-1} g_{k-1}(X)] + \ldots = (a_0 + a_k X^n + a_{2k} X^{2n} + \ldots) g_0(X) + \ldots + (a_{k-1} + a_{2k-1} X^n + \ldots) g_{k-1}(X) = \sum_{i=0}^{k-1} a^{(i)}(X^n) g_i(X)$.

Exemplul 16.7 *Matricea generatoare a $(2, 2)$ - codului convoluţional descris în Exemplul 18.6 este*

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 & & & & & \\ 0 & 1 & 0 & 1 & 1 & 1 & & & & & \\ & & 1 & 1 & 0 & 1 & 0 & 1 & & & \\ & & 0 & 1 & 0 & 1 & 1 & 1 & & & \\ & & & & 1 & 1 & 0 & 1 & 0 & 1 & \dots \\ & & & & 0 & 1 & 0 & 1 & 1 & 1 & \dots \\ & & & & & \vdots & & & & & \end{pmatrix}$$

(locurile goale sunt completate cu 0*).*

Pentru intrarea 101 *ieşirea este*

$$(1\,0\,1)\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} = (1\,1\,1\,0\,0\,0\,0\,1),\ \textit{deci}$$

$$C(1+X^3) = 1 + X + X^3 + X^8.$$

S-a prezentat anterior modul de construcţie al unui circuit liniar pentru $(n, 1)$ - coduri convoluţionale, plecând de la polinomul generator $g(X)$. În mod similar se determină circuitul liniar de codificare pentru un (n, k) - cod convoluţional:

1. Pentru $i = 0, 1, \dots, k-1$ se construieşte circuitul liniar al $(n, 1)$ - codului dat de $a(X) \mapsto a(X^n)g_i(X)$;

2. Se foloseşte un buffer comun de ieşire (de n simboluri), în care ajung rezultatele însumate ale celor k circuite;

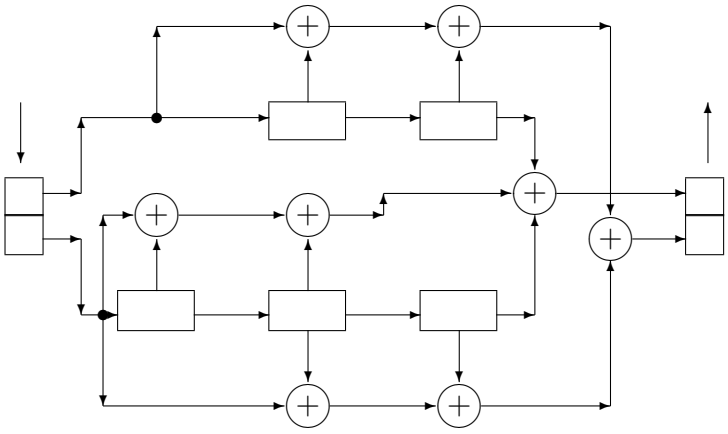
3. Secvenţa de intrare este segmentată în grupe de câte k simboluri, fiecare din ele constituind intrarea în câte un circuit liniar.

Exemplul 16.8 *Să construim circuitul liniar pentru* $(2, 2)$ *- codul definit de polinoamele generatoare*

$$g_0(X) = X + X^3 + X^4 + X^5, \quad g_1(X) = 1 + X + X^2 + X^4 + X^5 + X^6 + X^7.$$

Scriem $g_0(X) = (0+X)+(0+X^3)+(X^4+X^5)$ *deci circuitul său liniar are* 2 *elemente de înmagazinare. Pentru* $g_1(X) = (1 + X) + (X^2 + 0) + (X^4 + X^5) + (X^6 + X^7)$ *sunt necesare trei elemente de înmagazinare.*

Reprezentarea sa grafică este:



Definiţia 16.3 *Un (n,k) - cod convoluţional are memorie N dacă*

$$\forall i\ (0 \le i \le k-1), \quad grad(g_i(X)) \le Nn,$$

iar N este minim cu această proprietate. Numărul Nn se numeşte lungimea restrânsă a codului.

După cum s-a văzut, un cod convoluţional de memorie N poate fi implementat cu o combinaţie de $N-1$ elemente de înmagazinare. N este numărul de simboluri de ieşire care pot fi modificate la schimbarea unui singur caracter de intrare.

16.4 Coduri convoluţionale sistematice

Un (n,k) - cod convoluţional sistematic are proprietatea că la fiecare ieşire de n caractere, pe primele k poziţii se găsesc simbolurile de informaţie. Vom da o construcţie directă a acestor coduri, plecând tot de la similitudinea cu codurile - bloc.

Fie n, k, N $(k < n)$ numere naturale nenule. Considerăm sistemul de $k(n-k)$ secvenţe cu N componente peste Z_q:

$$g(i,j) = g_0(i,j)g_1(i,j)\dots g_{N-1}(i,j), \quad 1 \le i \le k, \quad 1 \le j \le n-k.$$

Aceste secvenţe se numesc *subgeneratori*.

Considerăm matricea

$$Q_0 = \begin{pmatrix} g_\infty(1) \\ g_\infty(2) \\ \vdots \\ g_\infty(k) \end{pmatrix} = (I_k P_0 O_k P_1 O_k P_2 O_k \dots P_{N-2} O_k P_{N-1} O_k O_k \dots)$$

unde I_k este matricea unitate, O_k este matricea nulă (ambele de ordin k),

$$P_t = \begin{pmatrix} g_t(1,1) & g_t(1,2) & \dots & g_t(1,n-k) \\ g_t(2,1) & g_t(2,2) & \dots & g_t(2,n-k) \\ & & \vdots & \\ g_t(k,1) & g_t(k,2) & \dots & g_t(k,n-k) \end{pmatrix}, \qquad 0 \le t \le N-1$$

iar

$$g_\infty(i) = 0\dots010\dots g_0(i,1)\dots g_0(i,n-k)0\dots0g_1(i,1)\dots g_1(i,n-k)0\dots0$$
$$0\dots0g_{N-1}(i,1)\dots g_{N-1}(i,n-k)0\dots \qquad (1 \le i \le k).$$

Cuvintele obţinute din primele Nn componente ale lui $g_\infty(i)$ $(1 \le i \le k)$:

$$g(i) = 0\dots010\dots0g_0(i,1)\dots g_0(i,n-k)0\dots0g_{N-1}(i,1)\dots g_{N-1}(i,n-k)$$

se numesc *generatori*.

Fie l un număr natural. Se defineşte operatorul

$$D^l g_\infty(i) = \underbrace{0 \ldots 0}_{ln} g_\infty(i)$$

care translatează $g_\infty(i)$ cu ln poziţii spre dreapta.

În sfârşit, fie

$$G_\infty = \begin{pmatrix} Q_0 \\ DQ_0 \\ D^2Q_0 \\ \vdots \end{pmatrix} = \begin{pmatrix} g_\infty(1) \\ \vdots \\ g_\infty(k) \\ Dg_\infty(1) \\ \vdots \\ Dg_\infty(k) \\ D^2g_\infty(1) \\ \vdots \end{pmatrix} =$$

$$= \begin{pmatrix} I_kP_0 & O_kP_1 & O_kP_2 & \ldots & O_kP_{N-1} & & \ldots \\ & I_kP_0 & O_kP_1 & \ldots & O_kP_{N-2} & O_kP_{N-1} & \ldots \\ & & I_kP_0 & \ldots & O_kP_{N-3} & O_kP_{N-2} & O_kP_{N-1} \\ & & & & \vdots & & \end{pmatrix}$$

matricea generatoare a unui (n, k) - cod convoluţional sistematic.

Observaţie: Evident, un (n, k) - cod convoluţional sistematic este un (n, k) - cod convoluţional. Invers, fiind dat un (n, k) - cod convoluţional, el se poate transforma în unul sistematic prin adunarea la fiecare polinom generator $g_i(X) = \sum_{p=0}^{mn+n-1} b_{i,p}X^i$ a polinomului $g_i'(X) = \sum_{p=0}^{m}\sum_{j=0}^{k-1}(k - b_{i,np+j})X^{np+j} + b_iX^i \quad (0 \le i \le k-1)$.

Deşi are - teoretic - un număr infinit de linii şi coloane, fiind în formă canonică, pentru această reprezentare se poate construi imediat matricea de control:

$$H_\infty = \begin{pmatrix} -P_0^T & I_{n-k} & & & & & \ldots \\ -P_1^T & O_{n-k} & -P_0^T & I_{n-k} & & & \ldots \\ -P_2^T & O_{n-k} & -P_1^T & O_{n-k} & -P_0^T & I_{n-k} & \ldots \\ & & & \vdots & & & \\ -P_{N-1}^T & O_{n-k} & -P_{N-2}^T & O_{n-k} & -P_{N-3}^T & O_{n-k} & \ldots \\ & & -P_{N-1}^T & O_{n-k} & -P_{N-2}^T & O_{n-k} & \ldots \\ & & & & -P_{N-1}^T & O_{n-k} & \ldots \\ & & & \vdots & & & \end{pmatrix}$$

a cărei transpusă este

$$H_\infty^T = \begin{pmatrix} -P_0 & -P_1 & -P_2 & \ldots & -P_{N-1} & & \ldots \\ I_{n-k} & O_{n-k} & O_{n-k} & \ldots & O_{n-k} & & \ldots \\ & -P_0 & -P_1 & \ldots & -P_{N-2} & -P_{N-1} & \ldots \\ & I_{n-k} & O_{n-k} & \ldots & O_{n-k} & O_{n-k} & \ldots \\ & & & \vdots & & & \end{pmatrix}$$

Se verifică imediat egalitatea $G_\infty H_\infty^T = O_\infty$.

Să considerăm un mesaj de informaţie $\mathbf{a}$, sub forma unei succesiuni (teoretic) infinite. El va fi descompus iniţial în blocuri de lungime k:

$$\mathbf{a} = \underbrace{a_0(1)\dots a_0(k)}_{\mathbf{a_0}}\underbrace{a_1(1)\dots a_1(k)}_{\mathbf{a_1}}\dots\underbrace{a_p(1)\dots a_p(k)}_{\mathbf{a_p}}\dots$$

Mesajul codificat se obţine similar codurilor liniare, prin înmulţirea lui $\mathbf{a}$ cu matricea generatoare: $\mathbf{c} = \mathbf{a}G_\infty$; textul rezultat va fi descompus în blocuri de lungime n:

$$\mathbf{c} = \mathbf{a}G_\infty = \underbrace{c_0(1)\dots c_0(n)}_{\mathbf{c_0}}\underbrace{c_1(1)\dots c_1(n)}_{\mathbf{c_1}}\dots\underbrace{c_p(1)\dots c_p(n)}_{\mathbf{c_p}}\dots$$

Ţinând cont de forma canonică a matricii G_∞, rezultă următoarele relaţii:

$$c_p(i) = a_p(i), \quad (1 \le i \le k)$$
$$c_p(k+j) = \sum_{i=1}^{k}\sum_{t=0}^{N-1} a_{p-t}(i)g_t(i,j), \quad (1 \le j \le n-k). \tag{2}$$

Aceeaşi formulă de codificare se poate scrie şi în alt mod:

$\mathbf{c} = \mathbf{a}G_\infty = \sum_{i=1}^{k} a_0(i)g_\infty(i) + \sum_{i=1}^{k} a_1(i)Dg_\infty(i) + \sum_{i=1}^{k} a_2(i)D^2g_\infty(i) + \dots$, deci

$$\mathbf{c} = \sum_{i=1}^{k}\sum_{t=0}^{\infty} a_t(i)D^t g_\infty(i) \tag{3}$$

Exemplul 16.9 *Să considerăm un* $(2,1)$ *- cod convoluţional binar cu* $N = 4$ *şi subgenerator* $g(1,1) = 1101$. *Generatorul va fi atunci* $g(1) = 110100001$ *iar matricea*

$$G_\infty = \begin{pmatrix} 11 & 01 & 00 & 01 & & \dots \\ & 11 & 01 & 00 & 01 & \dots \\ & & 11 & 01 & 00 & 01 & \dots \\ & & & \vdots & & & \end{pmatrix}$$

Dacă dorim să se transmită secvenţa de informaţie $\mathbf{a} = 10011\dots$, *ea va fi codificată în* $\mathbf{c} = \mathbf{a}G_\infty = 1101001010\dots$

Exemplul 16.10 *Fie* $(3,2)$ *- codul convoluţional binar cu* $N = 3$ *şi subgeneratori* $g(1,1) = 101$, $g(2,1) = 110$. *Calculând generatorii, se obţine*

$$g(1) = 101000001, \quad g(2) = 011001000 \text{ şi deci}$$

$$G_\infty = \begin{pmatrix} 101 & 000 & 001 & & & \dots \\ 011 & 001 & 000 & & & \dots \\ & 101 & 000 & 001 & & \dots \\ & 011 & 001 & 000 & & \dots \\ & & 101 & 000 & 001 & \dots \\ & & 011 & 001 & 000 & \dots \\ & & & \vdots & & & \end{pmatrix}$$

Fie $\mathbf{a} = 110010\dots$ *o secvenţă de informaţie. Rezultatul codificării ei este* $\mathbf{c} = \mathbf{a}G_\infty$ $= 110001100\dots$

16.5 Arborescenţa codurilor convoluţionale

Cuvintele - cod dintr-un cod convoluţional sistematic se pot reprezenta folosind un graf arborescent infinit, cu noduri situate la o distanţă de n poziţii şi cu q^k descendenţi din fiecare nod. Codificarea unui mesaj revine la parcurgerea unui drum în acest arbore.

Pentru simplificare, să considerăm un $(n, 1)$ - cod convoluţional sistematic binar (generalizarea este imediată) de lungime restrânsă N. El este complet caracterizat prin $n-1$ subgeneratori

$$g(1,j) = g_0(1,j)g_1(1,j)\dots g_{N-1}(1,j), \qquad (1 \le j \le n-1).$$

Codul are un singur generator, anume

$$g(1) = 1g_0(1,1)g_0(1,2)\dots g_0(1,n-1)0g_1(1,1)g_1(1,2)\dots g_1(1,n-1)0\dots \\ \dots 0g_{N-1}(1,1)\dots g_{N-1}(1,n-1)$$

Vom folosi notaţiile

$$\mathbf{g_0} = 1g_0(1,1)\dots g_0(1,n-1),$$
$$\mathbf{g_p} = 0g_p(1,1)\dots g_p(1,n-1), \qquad (1 \le p \le N-1).$$

Atunci $g(1) = \mathbf{g_0 g_1 \dots g_{N-1}}$, iar $\mathbf{g_p}$ este a p - a ramificaţie a lui $g(1)$. Matricea generatoare a codului va fi - cu aceste notaţii:

$$G_\infty = \begin{pmatrix} g_\infty(1) \\ Dg_\infty(1) \\ D^2g_\infty(1) \\ \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{g_0} & \mathbf{g_1} & \mathbf{g_2} & \cdots & \mathbf{g_{N-1}} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{g_0} & \mathbf{g_1} & \cdots & \mathbf{g_{N-2}} & \mathbf{g_{N-1}} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{g_0} & \cdots & \mathbf{g_{N-3}} & \mathbf{g_{N-2}} & \mathbf{g_{N-1}} & \cdots \\ & & & & \vdots & & & \end{pmatrix}$$

unde $\mathbf{0}$ este un bloc de n zerouri.

Fie $\mathbf{a} = a_0a_1a_2\dots$ secvenţa de informaţie şi $\mathbf{c} = \mathbf{c_0c_1c_2}\dots$ cuvântul - cod corespunzător, unde $\mathbf{c_p} = c_p(1)c_p(2)\dots c_p(n)$. Are loc egalitatea

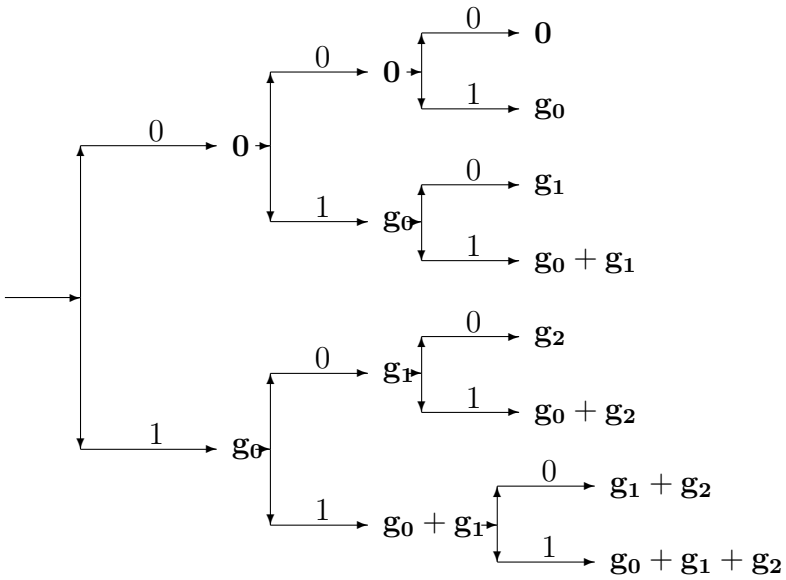
$$\mathbf{c} = \mathbf{c_0c_1c_2}\dots = \mathbf{a}G_\infty = a_0g_\infty(1) + a_1Dg_\infty(1) + a_2D^2g_\infty(1) + \dots \qquad (4)$$

De aici rezultă că $\mathbf{c_0} = \mathbf{0}$ dacă $a_0 = 0$ şi $\mathbf{c_0} = \mathbf{g_0}$ dacă $a_0 = 1$. Cu alte cuvinte, codul poate fi partiţionat în două submulţimi de mărime egală: o submulţime S_0 conţine toate secvenţele care corespund lui $a_0 = 0$ (deci toate cuvintele - cod din S_0 au acelaşi prefix $\mathbf{0}$); cealaltă submulţime S_1 conţine toate cuvintele - cod corespunzătoare lui $a_0 = 1$ (deci toate cuvintele - cod din S_1 au prefixul $\mathbf{g_0}$).

La rândul lui, S_0 se împarte în două submulţimi egale S_{00} şi S_{01}. S_{00} corespunde secvenţelor - cod pentru care $a_0 = a_1 = 0$ (deci toate cuvintele - cod din S_{00} încep cu $\mathbf{00}$). S_{01} conţine secvenţele cu $a_0 = 0,\ a_1 = 1$, deci cu prefixul $\mathbf{0g_0}$. Similar, S_1 se partiţionează în S_{10} şi S_{11}, S_{10} fiind pentru $a_0 = 1,\ a_1 = 0$ (deci cuvintele - cod din S_{10} au prefixul $\mathbf{g_0g_1}$); secvenţele din S_{11} încep cu prefixul $\mathbf{g_0(g_0 + g_1)}$.

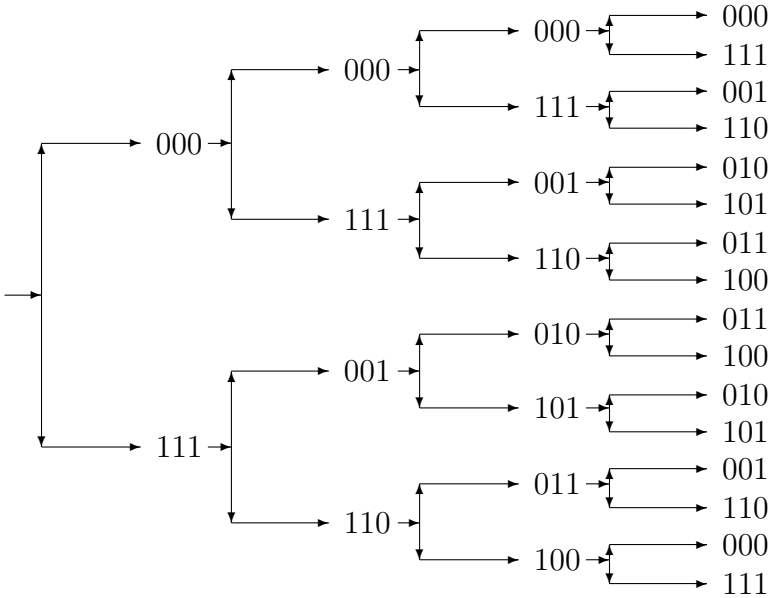
Acest procedeu continuă indefinit.

Deci cuvintele - cod pot fi aranjate într-un arbore cu noduri de n caractere şi câte două ramificaţii din fiecare nod. O ramificaţie este marcată de un cod bloc de n caractere, corespunzătoare unui caracter de informaţie particular, iar întreaga secvenţă corespunde unui drum prin arbore.



Operaţia de codificare poate fi privită atunci ca un proces în care este trasat un drum particular în arbore, conform instrucţiunilor date de mesajul de informaţie.

Exemplul 16.11 *Să considerăm* $(3,1)$ *- codul convoluţional binar cu* $N = 4$*, gene-rat de* $g(1,1) = 1011$, $g(1,2) = 1101$. *Generatorul codului este* $g(1) = 111\ 001\ 010\ 011$*, iar arborele de codificare*



Dacă vom considera de exemplu mesajul de informaţie $\mathbf{a} = 1001\ldots$*, secvenţa - cod generată va fi* $\mathbf{c} = 111\ 001\ 010\ 100\ \ldots$.

Această construcţie se poate extinde la un (n,k) - cod convoluţional sistematic binar. Deoarece la momentul 0 există 2^k secvenţe posibile de lungime k, toate cuvintele - cod se pot partiţiona în 2^k submulţimi $S_0, S_1, \ldots, S_{2^k-1}$. Toate secvenţele din S_i încep cu acelaşi bloc, corespunzător aceluiaşi mesaj de informaţie. Fiecare S_i se împarte la rândul lui în 2^k submulţimi, după tipul celui de-al doilea bloc de la tactul 1. Procesul continuă în mod similar până la epuizarea mesajului de informaţie.

16.6 Exerciţii

16.1 *Construiţi un codificator pentru* $(2,1)$ *- codul convoluţional binar de polinom generator* $g_0(X) = 1 + X + X^3 + X^4 + X^6 + X^7 + X^9$. *Cât este* N *? Codificaţi mesajele* 110, 0110, 1010.

16.2 *Construiţi o matrice generatoare pentru codul definit anterior.*

16.3 *Construiţi un codificator pentru* $(2,1)$ *- codul convoluţional ternar de polinom generator* $g_0(X) = 1 + 2X + 2X^3$.

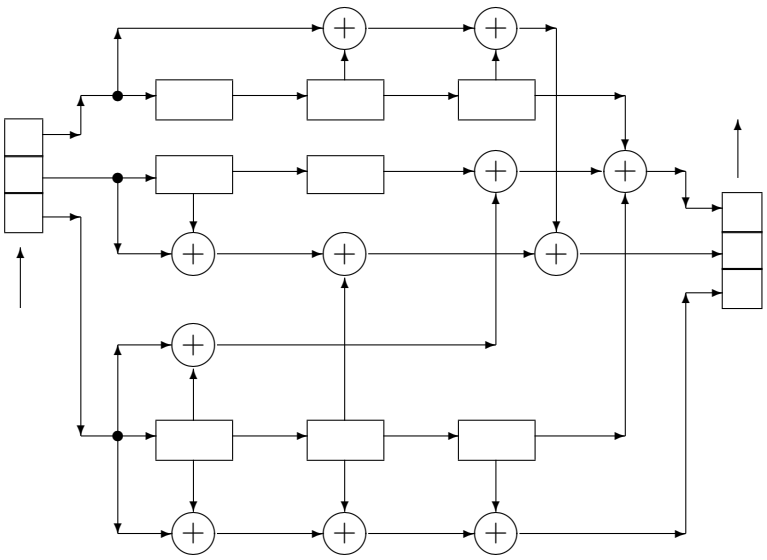
Să se determine o matrice generatoare.

Să se codifice mesajele 102, 200, 0120.

16.4 *Construiţi un codificator pentru un* $(3,2)$ *- cod convoluţional binar cu* $N = 2$.

16.5 *Codificatorul din Exemplul 16.8 are* 5 *elemente de înmagazinare. Construiţi un codificator pentru acelaşi cod, care utilizează doar* 4 *elemente de înmagazinare.*

16.6 *Construiţi polinoamele generatoare ale codului convoluţional dat mai jos:*



16.7 *Construiţi o matrice generatoare pentru codul din exerciţiul anterior.*

Codificaţi mesajul 11001.

16.8 *Construiţi un codificator pentru* $(4,3)$ *- codul binar, de polinoame generatoare* $g_0(X) = 1 + X + X^3 + X^5 + X^6$, $g_1(X) = X + X^2 + X^3 + X^4$, $g_2(X) = 1 + X^2 + X^3 + X^5 + X^6$.

16.9 *Să se construiască matricea generatoare şi de control pentru* $(3,2)$ *- codul convoluţional sistematic ternar de subgeneratori* $g(1,1) = 1200$, $g(2,1) = 2011$.

Să se codifice mesajele 11220, 1012, 2222.

16.10 *Aceeaşi problemă pentru* $(4,2)$ *- codul binar cu*

$g(1,1) = 001$, $g(1,2) = 110$, $g(2,1) = 101$, $g(2,2) = 111$.

Să se codifice mesajele 1100, 011, 001100.

Să se construiască reprezentarea arborescentă.

16.11 *Să se demonstreze formulele* (2).

Capitolul 17

Decodificarea codurilor convoluţionale

17.1 Capacitatea de corectare a erorilor

Algoritmii de decodificare pentru codurile convoluţionale folosesc aceeaşi idee de la codurile liniare: cuvântul primit este decodificat în cel mai apropiat cuvânt - cod, în sensul distanţei Hamming.

Ideea este de a nu decodifica deodată tot cuvântul primit, ci pas cu pas, la fiecare tact fiind decodificate n caractere (conţinutul unui buffer de ieşire) în k simboluri (mărimea bufferului de intrare). Distanţa Hamming (notată cu d_H) pentru un (n, k) - cod convoluţional se defineşte

$$d_H(a(X), b(X)) = d(a_0a_1 \dots a_{n-1}, b_0b_1 \dots b_{n-1}).$$

Observaţie: În cele ce urmează vom continua să identificăm secvenţele de $p+1$ caractere $\mathbf{a} = a_0a_1 \dots a_p$ cu polinoamele de gradul p $a(X) = a_0 + a_1X + \ldots + a_pX^p$. Se va folosi adesea chiar notaţia (neambiguă) $a(X) = \mathbf{a}$.

Fie deci $a(X)$ mesajul de informaţie; el se codifică în $v(X) = C(a(X))$. Să presupunem că se recepţionează $u(X)$. La primul tact se determină cuvântul - cod $w(X)$ cu $d_H(u(X), w(X))$ minim (cel mai apropiat cuvânt - cod), iar $u_0u_1 \dots u_{n-1}$ se transformă în $w_0w_1 \dots w_{n-1}$. Dacă decodificarea este corectă, atunci s-au determinat primele n caractere şi se pot găsi imediat ca-racterele de informaţie $a_0a_1 \dots a_{k-1}$ (conţinutul primei intrări). Se defineşte apoi

$$u(X) := u(X) - C(a_0 + a_1X + \ldots + a_{k-1}X^{k-1})$$

(care aduce 0 pe primele n caractere ale cuvântului recepţionat), se ignoră primele n caractere şi procedeul continuă inductiv.

Motivaţia constă în faptul că acum nu se lucrează cu $v(X) = C(a(X))$ ci cu

$v(X) - C(a_0 + a_1X + \ldots a_{k-1}X^{k-1}) = C(a(X) - a_0 - a_1X - \ldots - a_{k-1}X^{k-1}) = C(a_kX^k + a_{k+1}X^{k+1} + \ldots) = C(X^k(a_k + a_{k+1}X + a_{k+2}X^2 + \ldots)) = X^nC(a_k + a_{k+1}X + a_{k+2}X^2 + \ldots)$.

Exemplul 17.1 *Să considerăm* $(2, 1)$ *- codul convoluţional binar de polinom gene-rator* $g_0(X) = 1 + X + X^3$. *Cuvintele lui cod sunt*

$$\begin{array}{rl} \mathbf{0} & 00000000\ldots \\ g_0(X) & 11010000\ldots \\ X^2g_0(X) & 00110100\ldots \\ (1+X^2)g_0(X) & 11100100\ldots \\ \ldots & \ldots \end{array}$$

Dacă se primeşte - de exemplu - $\mathbf{u} = 11110001$*, vom determina cel mai apropiat cuvânt - cod, care este* $(1+X^2+X^4)g_0(X)$*, adică* 11101001*. Primul grup de* $n = 2$ *biţi este* 11*, deci primul bit de informaţie* $(k = 1)$ *este* 1*. Fie acum*

$$\mathbf{u} := \mathbf{u} - C(1) = 11110001 - 11010000 = 00100001.$$

Se ignoră primele două simboluri din noul $\mathbf{u}$ *şi se determină cel mai apropiat cuvânt - cod de* 100001*; acesta este* $000\ldots0$*; deci al doilea grup cadru este* 00 *- adică al doilea simbol de informaţie este* 0*.*

Refacem $\mathbf{u} := \mathbf{u} - X^2C(0) = 00100001$*. Se şterg primele patru simboluri din* $\mathbf{u}$ *şi se caută cel mai apropiat cuvânt - cod de* 0001*. Acesta este din nou* $00\ldots$*. Deci* $\mathbf{u}$ *se corectează în*

$$C(1+0X+0X^2) = 11000000.$$

Definiţia 17.1 *Pentru un* (n,k) *- cod convoluţional se defineşte distanţa liberă prin*

$$d_{lib} = min\{d_H(C(a(X)), C(a'(X)))|a_0a_1\ldots a_{k-1} \neq a_0{}'a_1{}'a_{k-1}{}'.$$

Similar observaţiei de la codurile liniare, d_{lib} este cea mai mică pondere a unui cuvânt - cod $C(a'(X))$ cu proprietatea $a_0{}'a_1{}'\ldots a_{k-1}{}' \neq \mathbf{0}$. (s-a notat $a'(X) = a(X) - a'(X)$ unde $a(X), a'(X)$ sunt două mesaje de informaţie arbitrare distincte).

Exemplul 17.2 *Reluând codul construit în Exemplul 20.3, acesta are* $d_{lib} = 3$*, deoarece* $g_0(X)$ *are ponderea* 3 *şi orice cuvânt* $C(a_0 + a_1X + \ldots)$ *cu* $a_0 \neq 0$ *are ponderea minim* 3*.*

Propoziţia 17.1 *Un cod convoluţional corectează* t *erori dacă şi numai dacă* $d_{lib} > 2t$*.*

Demonstraţie: Să presupunem $d_{lib} > 2t$. La recepţia unui cuvânt $v(X)$ cu cel mult t erori, există un cuvânt - cod $u(X) = C(a(X))$ aflat la o distanţă Hamming minimă de acesta: $d_H(u(X), v(X)) = s$. Dacă $u'(X) = C(a'(X))$ este cuvântul - cod trimis prin canal (recepţionat drept $v(X)$), din ipoteză $d_H(v(X), u'(X)) \leq t$. Deci, cu inegalitatea triunghiului, $d(u(X), u'(X)) \leq s + t$. Cum s este cea mai mică distanţă Hamming, avem $s \leq t$, deci $d(C(a(X)), C(a'(X))) \leq s+t \leq 2t < d_{lib}$. Din definiţia distanţei libere rezultă că primele k simboluri din $a(X)$ au drept cel mai apropiat grup de k simboluri generat de cod grupul primelor k simboluri din $a'(X)$, ceea ce permite decodificarea şi corectarea erorilor.

Pentru următoarele grupuri, procedeul decurge analog.

Invers, să presupunem $d_{lib} \leq 2t$ şi fie $u(X) = C(a(X))$, $u'(X) = C(a'(X))$ cu $d_H(u(X), u'(X)) = d_{lib}$ şi $u_0\ldots u_{k-1} \neq u_0{}'u_{k-1}{}'$. Fie $i_1, i_2, \ldots i_{d_{lib}}$ toţi indicii i pentru care $u_i \neq u_i'$. Construim următorul cuvânt $v(X)$:

$$v_i = \begin{cases} u_i & \text{dacă } u_i = u_i{}' sau i = i_{2s+1} \\ u_i{}' & \text{dacă } i = i_{2s} \end{cases}$$

Avem $d_H(u(X), v(X)) \leq d_H(u'(X), v(X)) \leq t$. Presupunem că a fost trimis $u'(X)$ şi recepţionat $v(X)$. Atunci eroarea, care a modificat cel mult t simboluri - poate conduce la o decodificare incorectă, deoarece $u(X)$ este cuvântul cod cel mai apropiat.

Deci, în această ipoteză, codul nu va corecta t erori. □

Din analiza de sus rezultă că un parametru important pentru un cod convolu-ţional este distanţa liberă, care specifică numărul maxim de erori care pot fi corectate. Din nefericire ([1]), nu se cunoaşte nici o metodă analitică de construcţie de coduri convoluţionale bune, în sensul unei maximizări a d_{lib} pentru n şi k date. Folosind calculatorul, au fost găsite însă o serie de astfel de coduri bune. Listăm câteva din ele pentru cazul binar şi $k = 1$:

n	d_{lib}	$g_0(X)$
2	5	$110111 = 1 + X + X^3 + X^4 + X^5$
2	6	11110111
2	7	1101011011
2	8	110111011011
2	10	11011111001011
2	10	111011110111
3	8	111011111
3	10	111011101111
3	12	111011101011111
4	10	111101111111
4	13	1111011110011111

17.2 Decodifocarea codurilor sistematice

Păstrând similitudinea cu codurile liniare, să folosim la codurile sistematice ideea de decodificare bazată pe calculul sindromului.

Fie un (n, k) - cod convoluţional sistematic definit prin matricea generatoare G_∞ şi cea de control H_∞, construite pe baza subgeneratorilor de lungime N $g(i, j), 1 \leq i \leq k,\ 1 \leq j \leq n - k$ (a se vedea prelegerea anterioară). Din relaţiile $G_\infty H_\infty^T = \mathbf{0}$ şi $\mathbf{c} = \mathbf{a}G_\infty$ se obţine $\mathbf{c}H_\infty^T = \mathbf{0}$.

Să presupunem că s-a recepţionat secvenţa (infinită) $\mathbf{r} = \mathbf{c} + \mathbf{e}$ unde $\mathbf{e}$ este o secvenţă eroare. Ca şi la codurile liniare, sindromul va fi

$$\mathbf{s} = \mathbf{r}H_\infty^T = \mathbf{c}H_\infty^T + \mathbf{e}H_\infty^T = \mathbf{e}H_\infty^T.$$

Secvenţa recepţionată se poate structura în blocuri de n caractere:

$$\mathbf{r} = \underbrace{r_0(1)\dots r_0(n)}_{\mathbf{r_0}}\underbrace{r_1(1)\dots r_1(n)}_{\mathbf{r_1}}\dots\underbrace{r_p(1)\dots r_p(n)}_{\mathbf{r_p}}\dots$$

Similar se segmentează sindromul în blocuri de lungime $n - k$:

$$\mathbf{s} = \underbrace{s_0(1)\dots s_0(n-k)}_{\mathbf{s_0}}\underbrace{s_1(1)\dots s_1(n-k)}_{\mathbf{s_1}}\dots\underbrace{s_p(1)\dots s_p(n-k)}_{\mathbf{s_p}}\dots$$

Vom avea

$$s_p(j) = r_p(k+j) - \sum_{i=1}^{k} r_p(i)g_0(i,j) - \sum_{i=1}^{k} r_{p-1}(i)g_1(i,j) - \dots - \sum_{i=1}^{k} r_{p-N+1}(i)g_{N-1}(i,j),$$
$$1 \leq j \leq n-k.$$

Această relaţie se poate scrie şi $s_p(j) = r_p(k+j) - A_p(j)$, unde

$A_p(j) = \sum_{t=0}^{N-1}\sum_{i=1}^{k} r_{p-t}(i)g_t(i,j)$ este rezultatul codificării secvenţei $r_p(1)\dots r_p(k)$ folo-sind matricea G_∞, în funcţie de $r_{p-1}(1), \dots, r_{p-1}(k), \dots, r_{p-N+1}(1)\dots, r_{p-N+1}(k)$.

Din aceste relaţii, se poate deduce o formulă pentru calculul elementelor sindromului, în funcţie de caracterele secvenţei - eroare:

$$s_p(j) = e_p(k+j) - \sum_{i=1}^{k} e_p(i)g_0(i,j) - \sum_{i=1}^{k} e_{p-1}(i)g_1(i,j) - \ldots - \sum_{i=1}^{k} e_{p-N+1}(i)g_{N-1}(i,j),$$

$$1 \le j \le n-k,$$

unde $\mathbf{e} = \underbrace{e_0(1)\ldots e_0(n)}_{\mathbf{e_0}}\underbrace{e_1(1)\ldots e_1(n)}_{\mathbf{e_1}}\ldots\underbrace{e_p(1)\ldots e_p(n)}_{\mathbf{e_p}}\ldots$

este secvenţa de eroare - tip, segmentată în blocuri de lungime n. Deci, pentru orice $j = 1, 2, \ldots, n-k$, putem scrie:

$$s_0(j) = e_0(k+j) - \sum_{i=1}^{k} e_0(i)g_0(i,j),$$

$$s_1(j) = e_1(k+j) - \sum_{i=1}^{k} e_1(i)g_0(i,j) - \sum_{i=1}^{k} e_0(i)g_1(i,j),$$

$$\vdots$$

$$s_{N-1}(j) = e_{N-1}(k+j) - \sum_{p=0}^{N-1}\sum_{i=1}^{k} e_p(i)g_{N-p-1}(i,j),$$

$$s_N(j) = e_N(j) - \sum_{p=0}^{N-1}\sum_{i=1}^{k} e_{p+1}(i)g_{N-p-1}(i,j),$$

$$\vdots$$

$$s_{N+m}(j) = e_{N+m}(k+j) - \sum_{p=1}^{N-1}\sum_{i=1}^{k} e_{p+m-1}(i)g_{N-p-1}(i,j)$$

$$\vdots$$

De remarcat că secvenţa $\mathbf{e_0}$ afectează numai primele N componente ale sindromului: $\mathbf{s_0}, \mathbf{s_1}, \ldots, \mathbf{s_{N-1}}$, deoarece apare numai în primele $N(n-k)$ ecuaţii din sistemul de sus.

Pentru corectarea erorilor, se determină iniţial secvenţa $\mathbf{e_0}$; după aceasta, sindromul se recalculează scăzând din el componentele lui $\mathbf{e_0}$. Această operaţie se numeşte *rearanjarea sindromului*. În paralel, $\mathbf{e_0}$ se foloseşte la corectarea primelor n caractere recepţionate; operaţia se realizează identic cu cea de la codurile liniare.

În continuare, se va folosi un nou sindrom, anume:

$s_0'(j) = 0$

$$s_m'(j) = e_m(k+j) - \sum_{p=0}^{m-1}\sum_{i=1}^{k} e_{m-p}(i)g_p(i,j), \qquad 1 \le m \le N-1, \qquad 1 \le j \le n-k$$

$$s_{N+m}'(j) = s_{N+m}(j), \qquad \forall m \ge 0.$$

De remarcat că secvenţa $\mathbf{e_1}$ figurează numai în expresiile lui $\mathbf{s_1}$", $\mathbf{s_2}$", ..., $\mathbf{s_N}$" unde $\mathbf{s_i}" = s_i"(1)\ldots s_i"(n-k)$, $1 \le i \le N$. Deci aceste componente apar în $N(n-k)$ ecuaţii. După determinarea lui $\mathbf{e_1}$ se corectează blocul $\mathbf{r_1}$ din secvenţa recepţionată, apoi se recalculează sindromul, ş.a.m.d.

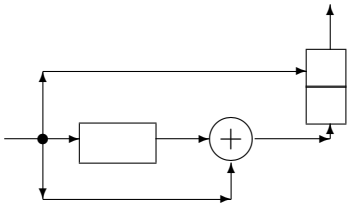
17.3 Algoritmul Viterbi

Cel mai cunoscut algoritm de decodificare pentru codurile convoluţionale aparţine lui Viterbi şi a fost folosit pe scară largă în comunicaţiile spaţiale. Astfel - ca să dăm numai un exemplu, staţia Voyager în misiunea sa din 1974 spre Marte, Saturn şi Jupiter, a folosit pentru transmisii un $(2,1)$ - cod convoluţional binar de polinom generator $g_0(X) = 1 + X + X^2 + X^5 + X^6 + X^7 + X^8 + X^{10} + X^{11} + X^{12}$.

În descrierea algoritmului Viterbi vom folosi o reprezentare în *reţea* a codurilor convoluţionale, care reia sub o formă finită reprezentarea arborescentă. Din nou, ne vom mărgini la construcţii pentru $(n,1)$ - coduri convoluţionale binare, extensia la cazul general fiind imediată.

O *diagramă reţea* este un graf orientat infinit, care are ca noduri stările unui circuit liniar la fiecare moment (tact). Un nod marcat care corespunde unei stări S la momentul i, este legat direct cu alte două noduri, corespunzătoare stărilor circuitului la momentul $i+1$: S_0 (pentru intrarea 0) respectiv starea S_1 (pentru intrarea 1). Grafic, arcul $S \longrightarrow S_0$ va fi totdeauna trasat sub arcul $S \longrightarrow S_1$. Fiecare arc este marcat cu secvenţa de ieşire a circuitului pentru intrarea corespunzătoare.

Exemplul 17.3 $(2,1)$ - *codul convoluţional definit de circuitul liniar*

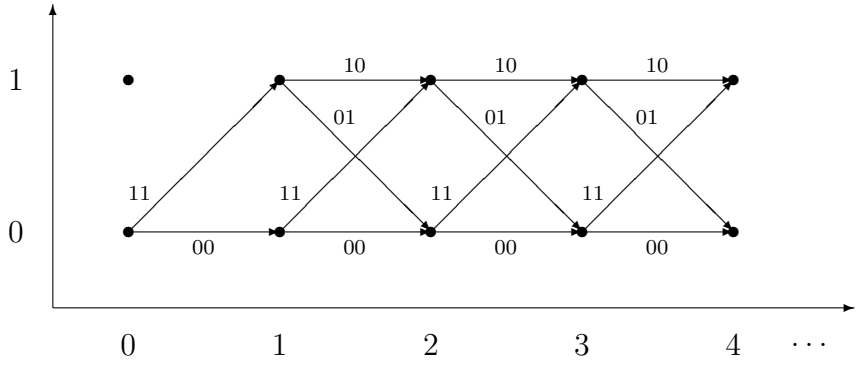


are două stări posibile (după conţinutul elementului de înmagazinare): 0 *şi* 1.

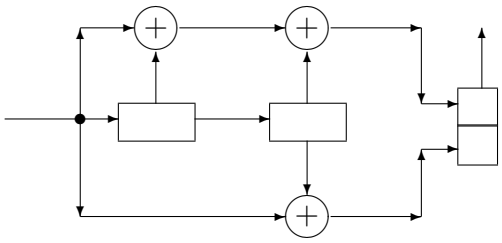
Pentru intrarea i $(i = 0, 1)$ *ieşirea este secvenţa*

ii	*dacă starea este* 0,
$i(1-i)$	*dacă starea este* 1.

Diagrama reţea a codului este:



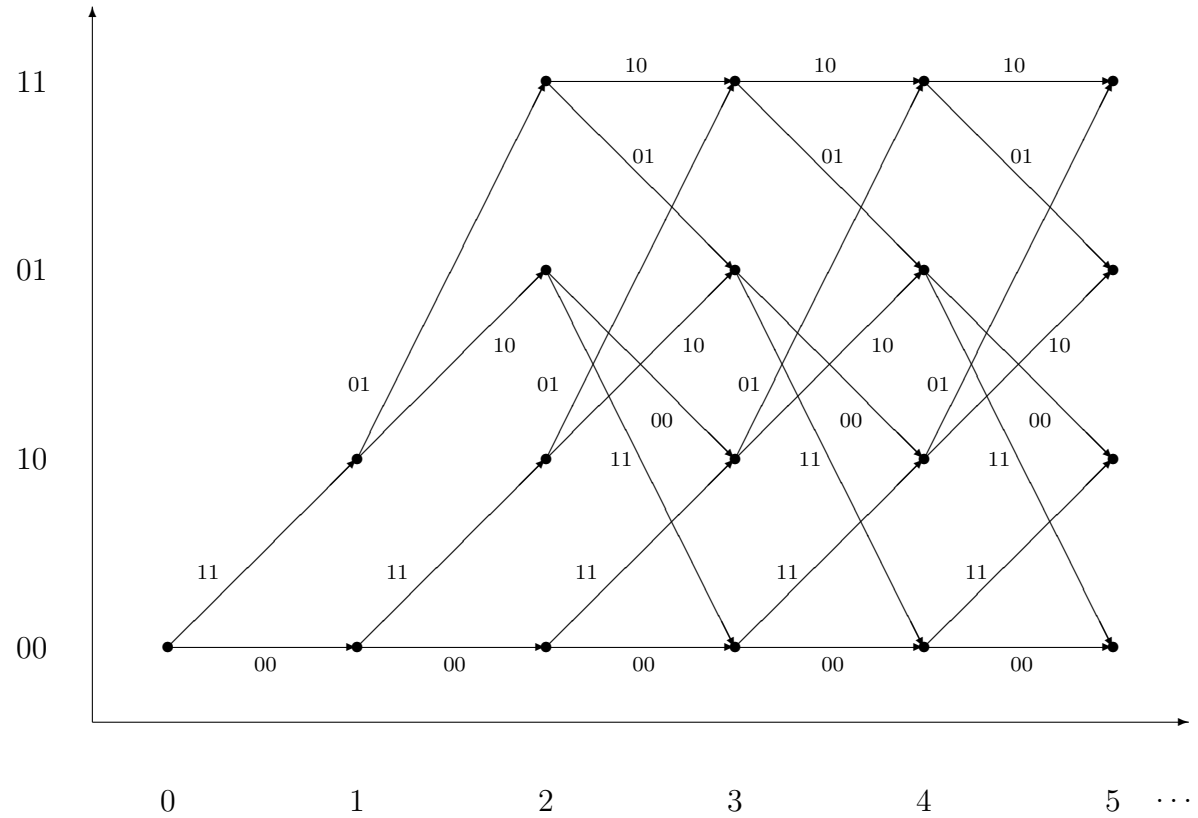
Exemplul 17.4 $(2,1)$ - *codul convoluţional (de memorie* $N = 2$*) reprezentat prin circuitul liniar*



are patru stări: 00, 01, 10 *şi* 11 *(după conţinuturile elementelor de înmagazinare).*

Din starea $S = 00$ *se ajunge la* $S_0 = 00$ *dacă intrarea este* 0*, sau în* $S_1 = 10$ *dacă intrarea este* 1*. Ieşirile corespunzătoare sunt* 00 *respectiv* 11*.*

Analog, dacă $S = 10$*, atunci* $S_0 = 01$ *(intrare* 0 *ieşire* 10*) şi* $S_1 = 11$ *(intrare* 1 *ieşire* 01*), etc. Se ajunge la diagrama reţea următoare:*



Orice drum prin reţea, care pleacă din momentul 0 *conduce - prin citirea marcajelor arcelor - la un cuvânt - cod. De exemplu, pentru intrarea* 100110*, codul generat este* $C(100110) = 111000011001$ *(reamintim, spre dreapta pleacă totdeauna două arce, iar arcul pentru intrarea* 0 *este situat sub arcul pentru intrarea* 1*).*

Invers, orice cuvânt - cod reprezintă marcajul unui drum în reţea.

Să construim acum algoritmul de decodificare Viterbi.

La recepţia unui cuvânt $\mathbf{v} = \mathbf{v_0 v_1 v_2} \ldots$ vom căuta construcţia unui drum în reţea, cât mai apropiat de $\mathbf{v}$.

Pentru fiecare moment i şi stare S vom lista drumurile *active* prin reţea până la starea S la momentul i. Un drum este *activ* dacă *discrepanţa* sa este minimă (prin *discrepanţă* se înţelege distanţa Hamming dintre cuvântul generat de drumul prin arbore şi cuvântul de aceeaşi lungime primit la intrare). Vom adnota fiecare stare la momentul i cu discrepanţa drumului său activ. Pentru calculul ei se poate folosi o formulă recursivă definită astfel:

Fie $d(S \to S")$ distanţa Hamming dintre marcajul arcului $S \to S"$ şi secvenţa de n caractere primită, iar $di(S)$ discrepanţa stării S. Atunci

$$di(S) = \min_{S \to S"} \{di(S) + d(S \to S")\} \tag{1}$$

Algoritmul va avea o durată de funcţionare finită numită *fereastră de decodificare*, pe care o notăm cu b (deci va funcţiona b tacţi).

Algoritmul Viterbi:
Intrare: Secvenţa $\mathbf{v_0 v_1} \ldots$, $\mathbf{v_p} = v_{p,0} v_{p,1} \ldots v_{p,n-1}$;
Algoritm:

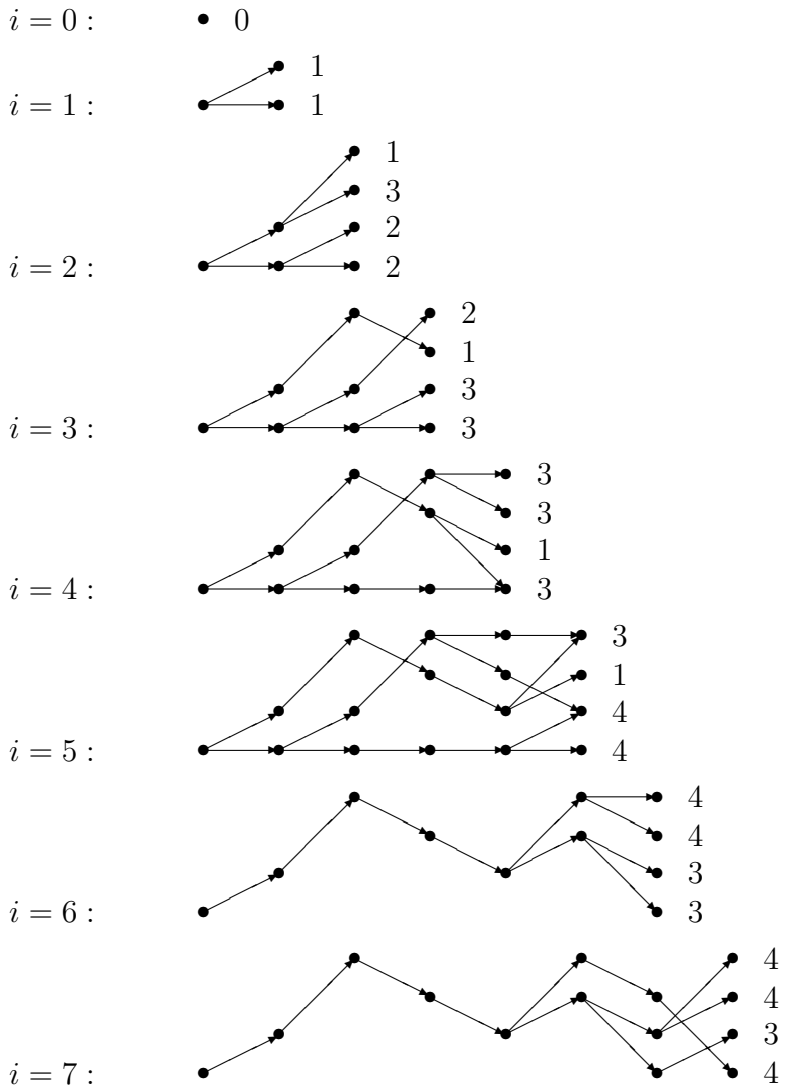
1. Fie $p := 0$ şi S_0 starea iniţială a reţelei (starea cu toate elementele de înmagazinare 0);
2. Se marchează $di(S_0) = 0$;
3. **for** $i := 1$ **to** b **do**

   Pentru toate stările S accesibile la momentul $p+i$ se determină $di(S)$ folosind formula (1), şi se listează toate drumurile active care duc la S.
4. Dacă toate drumurile active la momentul $p + b$ încep cu acelaşi arc $S_0 \to S_1$, secvenţa de intrare $\mathbf{v_p}$ se corectează în marcajul $\mathbf{u_p}$ al acestui prim arc (şi se decodifică în primul caracter al lui S_1). Altfel, eroarea nu este corectabilă, STOP.
5. $p := p + 1$, $S_0 := S_1$, salt la pasul 3.

Exemplul 17.5 *Să reluăm codul descris în Exemplul 18.7, împreună cu diagrama sa de reţea. Presupunem că s-a primit secvenţa* $\mathbf{v} = 10010100101100000 \ldots$ *Deci*

$\mathbf{v_0} = 10$, $\mathbf{v_1} = 01$, $\mathbf{v_2} = 01$, $\mathbf{v_3} = 00$, $\mathbf{v_4} = 10$, $\mathbf{v_5} = 11$, $\mathbf{v_6} = 00 \ldots$.

Vom lucra cu o fereastră de decodificare $b = 7$. *Detaliem grafic numai procedura de decodificare a primului bloc (cazul* $p = 0$*).*



Se ajunge la concluzia că toate arcele active au acelaşi început: arcul marcat cu 11*. Deci primul bloc* $\mathbf{v_0} = 10$ *se decodifică în* 11*, după care se ia ca nod iniţial* $S_0 = 10$ *şi procesul se repetă.*

De remarcat că pentru o fereastră de decodificare $b = 5$*, eroarea nu s-ar fi putut corecta (primul arc nu este unic).*

În această prezentare a algoritmului Viterbi, apare o problemă: cât de mare este fereastra b ? Cât de departe mergem prin diagrama reţea până să fim siguri că totuşi este posibilă decodificarea şi corectarea erorilor ?

Pentru aceasta să refacem sub o formă finită reprezentarea diagramelor - reţea.

Din definiţia dată în Prelegerea 16 se observă că dacă $g_0(X), g_1(X), \ldots,$ $g_{n-1}(X)$ sunt polinoamele generatoare şi $a(X)$ un mesaj de informaţie, codificarea poate fi scrisă ca o secvenţă de n componente (infinite)

$$c(X) = (a(X)g_0(X), a(X)g_1(X), \ldots, a(X)g_{n-1}(X))$$

ale căror caractere se transmit în paralel, în ordinea crescătoare a puterilor lui X.

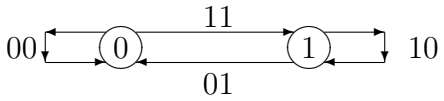
Exemplul 17.6 *Fie* $(2, 1)$*- codul generat de* $g_0(X) = 1 + X + X^3$ *şi* $g_1(X) = 1 + X$*. Mesajul* $a(X) = 1 + X^2$ *este codificat în*

$c(X) = ((1+X^2)(1+X+X^3), (1+X^2)(1+X)) = (1+X+X^2+X^5, 1+X+X^2+X^3)$.
Deci cuvântul cod este $\mathbf{c} = 11\ 11\ 11\ 01\ 00\ 10\ 00\ \ldots$.

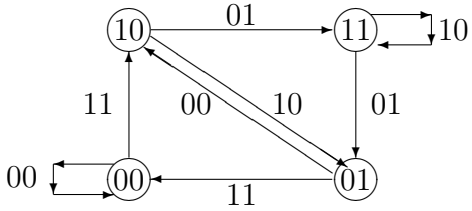
Vom construi un translator (*diagramă de translatare*) asociat unui $(n, 1)$ - cod convoluţional astfel:

Stările translatorului sunt N - tupluri binare, fiecare stare reprezentând o situaţie posibilă a celor N elemente de înmagazinare. O stare $s_1 \ldots s_{N-1}s_N$ este legată direct prin arcul marcat $x_1 \ldots x_n$ de starea $is_1 \ldots s_{N-1}$, $(i = 0, 1)$ dacă în circuitul liniar corespunzător codului, intrarea i conduce la modificarea conţinuturilor elementelor de înmagazinare, din $(s_1, \ldots, s_N)$ în $(i, s_1, \ldots, s_{N-1})$ şi are ca efect ieşirea $x_1 \ldots x_n$.

Exemplul 17.7 *Reţeaua codului definit în Exemplul 17.3 poate fi reprezentată sub formă de diagramă de translatare astfel:*



Pentru codul din Exemplul 18.7, reprezentarea finită este:



De remarcat că pe fiecare arc este suficient să marcăm doar secvenţa de ieşire; caracterul de intrare este reprezentat de primul caracter al stării în care intră arcul respectiv.

Definiţia 17.2 *Se defineşte lungimea unui drum ca fiind numărul de arce care formează acel drum.*

Ponderea unui drum este suma ponderilor arcelor care formează drumul.

Reamintim, *ponderea unui cuvânt* este numărul de caractere nenule din cuvântul respectiv.

Conform Definiţiei 17.1, distanţa liberă a unui cod este ponderea nenulă minimă a unui cuvânt cod - deci a unei secvenţe care marchează un drum prin diagrama de translatare.

Propoziţia 17.2 *Distanţa liberă a unui* $(n, 1)$ *- cod convoluţional este egală cu ponderea nenulă minimă a unui ciclu care trece prin starea iniţială* $00 \ldots 0$ *a diagramei de translatare a codului.*

Demonstraţie: Este imediată.

Fie $S_0 = 00 \ldots 0$ starea iniţială. Pentru orice t, $1 \leq t \leq \left[\dfrac{d_{lib} - 1}{2}\right]$, se defineşte $d(t)$ ca fiind cel mai mic număr întreg pozitiv p cu proprietatea că orice drum $S_0 \to S_1 \to \ldots S_{p-1}$, $(S_0 \not\equiv S_1)$ are ponderea mai mare de $2t$. Pentru determinarea algoritmică a lui $d(t)$ se poate adapta imediat un algoritm similar din teoria grafurilor.

Deoarece codul este liniar, valoarea lui $d(t)$ se păstrează pentru orice stare a diagramei de translatare.

Acum, în condiţiile că pot apare maxim t erori, algoritmul Viterbi va funcţiona corect pentru o fereastră $b = d(t)$ (datorită Propoziţiei 17.1).

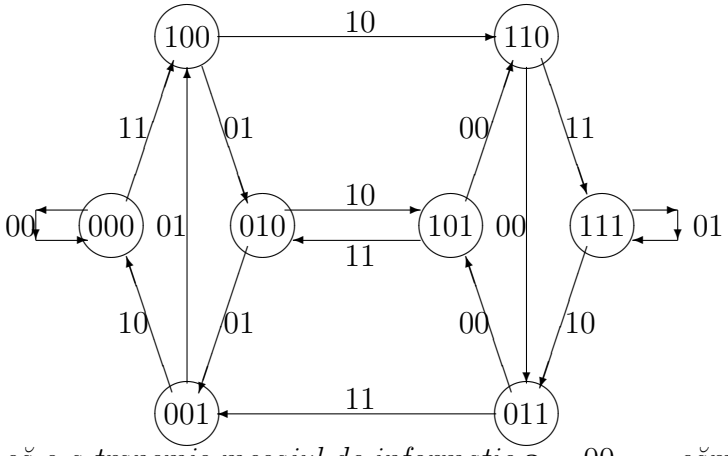
Reluarea algoritmului Viterbi la fiecare pas cu fereastra maximă $d(t)$ este însă destul de costisitoare ca timp; de aceea, practic, se foloseşte o fereastră mobilă $b \leq d(t)$ şi algoritmul trece la faza de decodificare atunci când toate drumurile active selectate au acelaşi prim arc. Dacă s-a ajuns la $b = t(b)$ şi nu s-a selectat un prim arc comun, algoritmul eşuează.

17.4 Coduri convoluţionale catastrofice

Dacă este definit neglijent, un cod convoluţional poate conduce la următoarea situa-ţie, complet nefericită: un mesaj în care a fost perturbat un număr mic de caractere generează prin decodificare o infinitate de erori.

Pentru a studia acest caz, considerăm din nou numai codurile $(n, 1)$ - convoluţio-nale binare. Pentru a vedea în ce constă problema, să luăm următorul exemplu:

Exemplul 17.8 *Fie* $(2, 1)$ *- codul convoluţional binar, cu polinoamele generatoare* $g_0(X) = 1 + X^3, g_1(X) = 1 + X + X^2$. *Diagrama sa de translatare este*



Să presupunem că s-a transmis mesajul de informaţie $\mathbf{a} = 00\ldots$, *căruia îi corespunde cuvântul - cod nul* $\mathbf{c} = 0000\ldots$. *La recepţie s-a primit un cuvânt cu primele trei simboluri greşite:* $\mathbf{v} = 111000\ldots$. *Decodificarea sa nu ridică nici o problemă, acesta fiind de asemenea cuvânt - cod, care marchează drumul prin stările* $000-100-110-011-101-110-\ldots$.

Deci, în ipoteza generală din Teoria Codurilor (cea a decodificării cele mai proba-bile), nu au apărut erori, şi mesajul decodificat este $\mathbf{a}$" $= 110110110\ldots$, *care diferă de* $\mathbf{a}$ *într-o infinitate de poziţii.*

În acest mod, modificarea doar a primelor trei caractere a dus la situaţia în care aceste erori se propagă într-o secvenţă infinită.

Se observă din acest exemplu că tot necazul provine din faptul că diagrama conţine două cicluri distincte cu ieşirea 0: $000 - 000$ şi $110 - 011 - 101 - 110$.

Definiţia 17.3 *Un cod convoluţional se numeşte catastrofic dacă diagrama sa de translatare conţine două cicluri distincte de pondere zero.*

Observaţie: Orice cod convoluţional conţine un ciclu de pondere zero, anume bucla care pleacă şi intră în starea iniţială. Reamintim, ponderea unui drum este suma ponderilor arcelor sale.

Teorema 17.1 *Un* $(n,1)$ *- cod convoluţional este catastrofic dacă şi numai dacă* $cmmdc(g_0(X),\ldots,g_{n-1}$ 1.

Demonstraţie: Teorema rămâne valabilă dacă o demonstrăm pentru cazul $n=2$. Fie deci un $(2,1)$ - cod convoluţional, de polinoame generatoare $g_0(X),\ g_1(X)$. Pentru a marca un ciclu, un cuvânt de intrare trebuie să fie de forma $b(X)+X^r a(X)\cdot(1+X^p+X^{2p}+\ldots)$, cu $grad(a(X))<p$. Pentru a elimina din discuţie bucla din starea S_0, vom considera $a(X)\not\equiv 0$. Să presupunem că în diagrama de translatare a codului există un ciclu de lungime p şi pondere 0. Atunci $a(X)\cdot g_0(X)$ şi $a(X)\cdot g_1(X)$ trebuie să se dividă cu $1+X^p$, deci $a(X)\cdot g'(X)$ (unde $g'(X)=cmmdc(g_0(X),g_1(X))$), se divide cu $1+X^p$.

Dacă $g'(X))=1$, se ajunge la o contradicţie (deoarece $a(X)$ are grad mai mic decât p şi nu este polinomul identic nul). Deci $grad(g'(X))\geq 1$. În acest caz, toate mesajele de informaţie $b(X)+a(X)\left[g'(X)\right]^t(1+X^p+X^{2p}+\ldots),\ (t\geq 0)$ se codifică identic, ceea ce va conduce la imposibilitatea decodificării.

Reciproca se demonstrează similar. □

17.5 Exerciţii

17.1 *Determinaţi* d_{lib} *pentru fiecare din codurile următoare:*

(a) $\quad g_0(X)=1+X^2,\quad g_1(X)=1+X+X^2$*;*

(b) $\quad g_0(X)=1+X+X^2+X^3,\quad g_1(X)=1+X^2+X^3$*;*

(c) $\quad g_0(X)=1+X^3+X^4,\quad g_1(X)=1+X+X^2+X^4$*.*

17.2 *Folosind* $(2,2)$ *- codul convoluţional definit în Exemplul 18.7, şi fereastra de decodificare* $b=7$*, decodificaţi (cât este posibil) secvenţele*

$$\mathbf{v}=01000001000\ldots$$
$$\mathbf{v}=11000110010010001110010\ldots$$

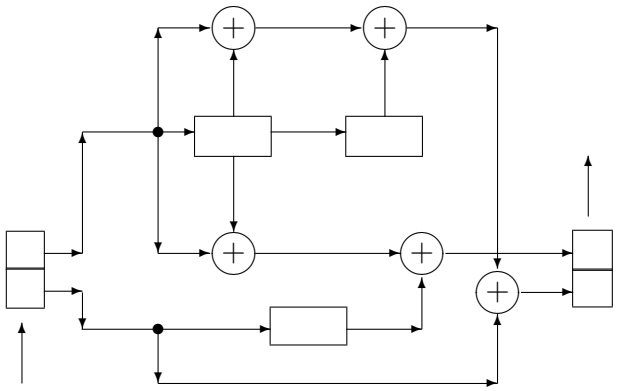
17.3 *Trasaţi diagrama reţea a* $(2,1)$ *- codului convoluţional generat de* $g_0(X)=1=X^2+X^3+X^4$*. Folosiţi algoritmul Viterbi pentru decodificarea secvenţei* $\mathbf{v}=1000001000001000\ldots$

17.4 *Fie* $(2,1)$ *- codul generat de* $g_0(X)=1+X+X^2+X^3,\quad g_1(X)=1+X^2+X^3$*. Decodificaţi primele* 4 *caractere de informaţie ale cuvântului recepţionat* $\mathbf{c}=11000000\ldots$ *pentru*

$$b=2,\qquad b=3,\qquad b=4.$$

17.5 *Generalizaţi conceptul de diagramă reţea la* (n,k) *- coduri.*

Trasaţi reţeaua $(2,2)$ *- codului definit de circuitul liniar:*



17.6 *Generalizaţi noţiunea de diagramă de translatare pentru* (n, k) *coduri binare. Construiţi diagrama de translatare a codului definit în exerciţiul anterior.*

17.7 *Demonstraţi Propoziţia 17.2,*

17.8 *Pentru codurile definite în Exerciţiul 17.1, determinaţi diagramele de translatare şi* $d(t)$ *pentru* $1 \leq t \leq \left[\frac{d_{lib}-1}{2}\right]$.

17.9 *Ce se întâmplă dacă se încearcă să se determine* $d(t)$ *pentru* $t > \left[\frac{d_{lib}-1}{2}\right]$ *?*

17.10 *Construiţi un algoritm pentru determinarea lui* $d(t)$ *în cazul unui* $(n, 1)$ *- cod convoluţional (*$1 \leq t \leq \left[\frac{d_{lib}-1}{2}\right]$ *dat).*

17.11 *Pentru fiecare din codurile următoare, decideţi dacă sunt catastrofice sau nu. În caz afirmativ, determinaţi ciclurile de pondere* 0.

(a) $g_0(X) = 1 + X, \quad g_1(X) = 1 + X + X^2 + X^3$;
(b) $g_0(X) = 1 + X + X^4, \quad g_1(X) = 1 + X^2 + X^4$;
(c) $g_0(X) = 1 + X + X^2, \quad g_1(X) = 1 + X + X^3 + X^4$.

Capitolul 18

Coduri - tablou

18.1 Definirea codurilor tablou

În Capitolul 3 au fost definite codurile - produs. Deşi construcţ ia prezentată era cunoscută din anii ′60, dezvoltarea acestei clase de coduri a fost explozivă abia după trei decenii, odată cu introducerea pe piaţă a telefoniei mobile. Acum s-a arătat că orice cod - bloc poate fi generat finit recursiv, folosind o variantă de coduri - produs.

Să considerăm o clasă particulară de coduri - produs binare, în care $n_1 = k_1 + 1,\ n_2 = k_2 + 1$ (deci singurul simbol de control este bitul de control al parităt ii). Acest cod este generat de produsul Kronecker a două matrici de forma

$$G_{p,p+1} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 1 \\ 0 & 1 & 0 & \dots & 0 & 1 \\ & & & \vdots & & \\ 0 & 0 & 0 & \dots & 1 & 1 \end{pmatrix} \tag{1}$$

unde $p = k_1, k_2$.

Un astfel de $(n_1 \cdot n_2, k_1 \cdot k_2)$ - cod are distanţ a $d = d_1 \cdot d_2 = 2 \cdot 2 = 4$, deci corectează o eroare şi detectează două erori. Chiar dacă în scrierea liniară a cuvintelor - cod ultimul simbol de control se poate elimina (el este de fapt suma modulo 2 a biţ ilor de control de pe ultima linie şi coloană) distanţ a se reduce la 3, deci capacitatea de corectare a unei erori rămâne.

Exemplul 18.1 *Fie un* $(3, 2)$ *- cod binar sistematic, generat de matricea* $G = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$. *Codul produs* $(3, 2)(3, 2) = (9, 4)$ *va avea ca matrice generatoare*

$$G' = G'timesG = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Dacă $\mathbf{a} = 1001$ *este un mesaj de informaţ ie, el se codifică în* $\mathbf{x} = \mathbf{a}G' = 101011110$*, sau – folosind forma matricială:*

$$\mathbf{x} = \begin{matrix} 1 & 0 & \underline{1} \\ 0 & 1 & \underline{1} \\ \underline{1} & \underline{1} & \underline{0} \end{matrix} = 10\underline{1}01\underline{1110}$$

caracterele de control fiind subliniate, iar cuvântul - cod este transmis linie cu linie.

18.2 Structura de reţ ea a codurilor tablou

În cadrul codurilor convoluţ ionale s-a definit o structură de reţ ea pentru facilitarea codificării, decodificării, detectării şi corectării de erori (Prelegerea 17). Această structură se poate extinde cu mici modificări şi la codurile liniare.

Fie C un (n, k) - cod liniar care se poate reprezenta printr-un cod - tablou. Lui i se asociază un graf orientat cu arce marcate. Vârfurile grafului sunt partiţ ionate în coloane $V_0, V_1, \ldots, V_{N_c-1}$ $(N_c \leq n+1)$. O coloană V_i conţ ine mulţ imea stărilor la care se poate ajunge la momentul i. Fiecare arc U pleacă dintr-o stare $q_j \in V_i$ şi ajunge la o stare $q_m \in V_{i+1}$ $(i \geq 0)$; în acest caz, q_m este *succesorul* lui q_j, iar q_j este *predecesorul* lui q_m. Arcul este marcat cu o pereche $\delta(U)/l(U)$, unde $\delta(U) \in Z_q^*$ este un mesaj de informaţ ie, iar $l(U) \in Z_q^*$ este mesajul - cod corespunzător (de obicei ele se reduc la câte un singur caracter).

Proprietăt i:

(a) $V_0 = \{q_0\}$, $V_{N_c-1} = \{q_{N_c-1}\}$; deci, orice reţ ea pleacă dintr-o stare unică q_0 (*rădăcina*) şi ajunge de asemenea într-o stare unică q_{N_c-1} (*ţ inta*);

(b) Orice stare este accesibilă din rădăcină prin cel puţ in un drum;

(c) Din orice stare se poate ajunge la ţ intă prin cel puţ in un drum.

Numim *drum prin reţ ea* un drum de la rădăcină la ţ intă. Printr-o reţ ea se pot cataloga toate cuvintele unui cod liniar reprezentabil sub formă de cod - tablou. Orice cuvânt - cod corespunde unui drum unic prin reţ ea. Deci o astfel de reţ ea va conţ ine q^k drumuri distincte. Fiecare astfel de drum are două marcaje:

(i) concatenarea marcajelor de informaţ ie $\delta(U)$ ale fiecărui arc U al drumului; această secvenţă formează mesajul de informaţ ie care se codifică.

(ii) concatenarea marcajelor $l(U)$ ale fiecărui arc U al drumului; ele formează cuvântul care codifică secvenţ a de informaţ ie.

În acest fel, operaţ ia de codificare sau decodificare (fără corectare de erori) se reduce pentru fiecare mesaj de informaţ ie (cod) la determinarea acelui drum prin reţ ea, marcat de mesajul respectiv.

Să considerăm numai $(n_1 \cdot n_2, k_1 \cdot k_2)$ - coduri - tablou binare (generalizarea la cazul nebinar se face fără probleme deosebite). Pentru aceste coduri se pot construi evident mai multe structuri de reţ ea; vom considera aici numai reţ ele cu un număr minim de stări pe fiecare coloană. Construcţ ia unei astfel de reţ ele va necesita

$$N_s = 2^{min\{k_1, k_2\}}$$

stări (reamintim, aici $k_i = n_i - 1,\ i = 1, 2$). Procedura de generare ([8]) a reţ elei este:

1. Fie $n_1 = min\{n_1, n_2\}$ (pentru cazul invers se procedează similar);
2. Se determină numărul N_s de stări şi numărul N_c de coloane prin relaţ iile
$$N_s = 2^{n_1-1}, \qquad N_c = n_2 + 1;$$
3. Pentru fiecare coloană V_p $(0 \leq p \leq N_c - 1)$ se notează elementele (stările) sale prin $(A)_p = (a_1 a_2 \dots a_{k_1})_p$, unde $a_i \in Z_2$;
4. Rădăcina reţ elei este $(00\dots0)_0$, iar scopul este $(00\dots0)_{N_c-1}$;
5. Pentru orice pereche de stări $(A, B) \in V_p' times V_{p+1}$ $(0 \leq p < N_c - 2)$, se marchează arcul corespunzător cu $\delta_p(A, B)/l_p(A, B)$ definite $\delta_p(A, B) = (A)_p + (B)_{p+1}$, $l_p(A, B) = \delta_p(A, B)c_1$, unde c_1 este bitul de paritate al secvenţ ei $\delta_p(A, B)$ iar suma se realizează în Z_2.
6. Arcele care leagă stările $A \in V_{N_c-2}$ de ţ intă au $\delta_{N_c-2}(A, B) = \epsilon, l_{N_c-2}(A, B) = (A)_{N_c-2}c_1$ unde c_1 este bitul de paritate al secvenţ ei $(A)_{N_c-2}$ care defineşte starea A.

Există $2^{k_1 \cdot k_2}$ drumuri distincte prin reţ ea, fiecare corespunzând unui cuvânt - cod unic. Procedura poate fi extinsă uşor la codurile produs cu mai multe simboluri de control.

Exemplul 18.2 *Să construim reţ eaua asociată codului tablou* $(3, 2)(3, 2)$ *(deci* $n_1 = n_2 = 3$, $k_1 = k_2 = 2$, $d = 4$*). Ea va avea* $N_s = 2^{n_1-1} = 2^{3-1} = 4$ *stări distincte şi* $N_c = n_2 + 1 = 3 + 1 = 4$ *coloane.*

Fiecare stare va fi de forma $(a_1 a_2)_p$ *cu* $p = 0, 1, 2, 3$, $a_1, a_2 \in \{0, 1\}$.

$(00)_0$ *şi* $(00)_3$ *vor fi rădăcina respectiv ţ inta reţ elei.*

Marcajele arcelor sunt de forma $\delta_p(A, B)/l_p(A, B)$*, obţ inute prin toate combina-ţ iile posibile ale lui* $(a_1 a_2)_p$ *şi* $(a_1 a_2)_{p+1}$.

Astfel, pentru $p = 0$ *avem:*

$\delta_0(00, 00) = (00)_0 + (00)_1 = 00$ $\qquad$ $\delta_0(00, 01) = (00)_0 + (01)_1 = 01$

$\delta_0(00, 01) = (00)_0 + (10)_1 = 10$ $\qquad$ $\delta_0(00, 11) = (00)_0 + (11)_1 = 11.$

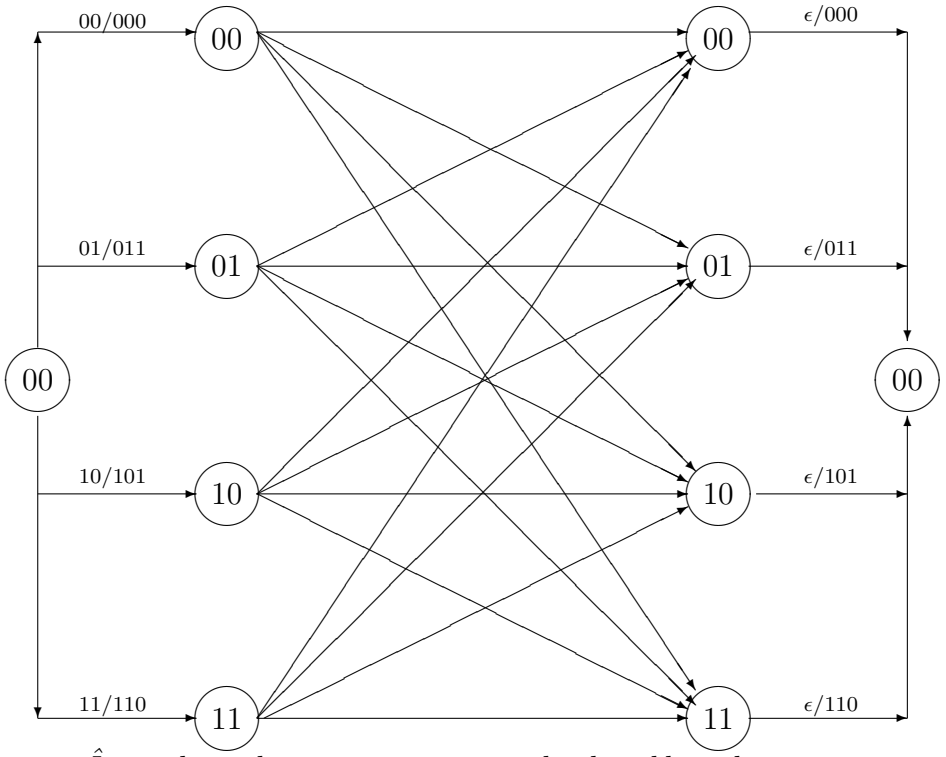
iar fiecare arc va fi marcat cu o pereche de forma
$$\delta_0/l_0 = a_1 a_2/a_1 a_2 c_1$$
unde $c_1 = a_1 + a_2$.

Pe nivelul $p = 1$*, arcele sunt marcate prin perechi din tabelul următor:*

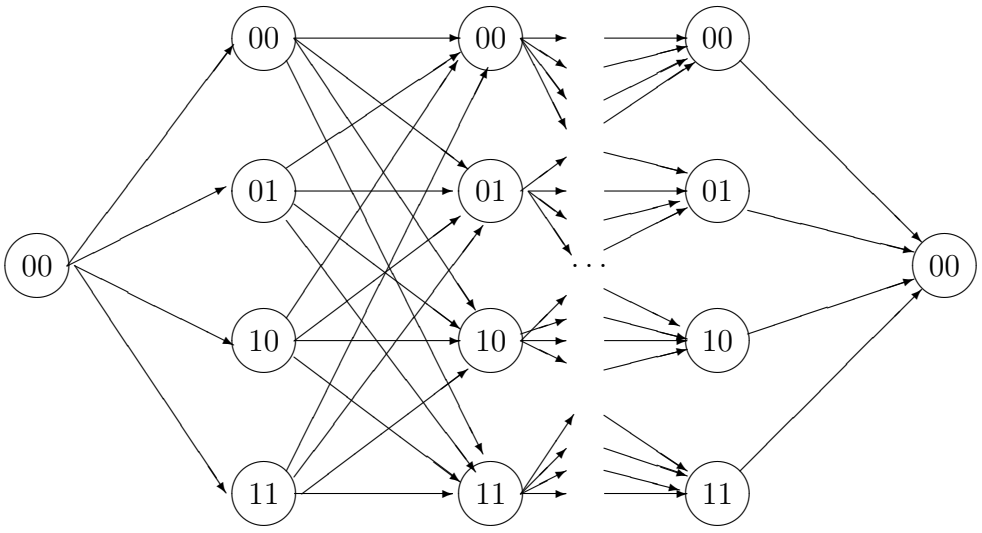
$\delta_1(A, B)$	00	01	10	11
00	00/000	01/011	10/101	11/110
01	01/011	00/000	11/110	10/101
10	10/101	11/110	00/000	01/011
11	11/110	10/101	01/011	00/000

La ultimul nivel, pentru $p = N_c - 2 = 2$ *vom avea* $\delta_2(A, 00) = \epsilon$ *şi* $l_2(00, 00) = 000$, $l_2(01, 00) = 011$, $l_2(10, 00) = 101$, $l_2(11, 00) = 110$ *(pe ultimul arc sunt numai simboluri de control).*

Reţ eaua se poate reprezenta grafic astfel:



Exemplul 18.3 *În mod similar se construiesc codurile tablou obţ inute prin produsul a două* $(3,2)(n_2,k_2)$ *- coduri bloc. Toate au acelaşi număr de stări* $N_s = 2^2 = 4$*, doar adâncimea (numărul de coloane) variind în funcţ ie de* n_2 *(*$N_c = n_2 + 1$*). Reprezentarea grafică a reţ elei este*



marcajul nodurilor fiind realizat similar cu cel din Exemplul 18.2.

18.3 Decodificarea codurilor tablou

Operaţ ia de codificare se realizează foarte simplu: fiecare mesaj de informaţ ie de $k_1 \cdot k_2$ caractere, se scrie ca o secvenţă de $k_2 + 1$ grupuri, primele k_2 grupuri având câte k_1 simboluri, iar ultimul ϵ. Această secvenţă trasează prin reţ ea un drum unic

$$(00\ldots0) - A_1 - \ldots - A_{N_c-2} - (00\ldots0)$$

de la rădăcină la ţ intă. Ceea ce se obţ ine prin concatenarea ieşirilor

$$l_0(00\ldots0, A_1)l_1(A_1, A_2)\ldots l_{N_c-2}(A_{N_c-2}, 00\ldots0)$$

va fi un cuvânt - cod al $(n_1 \cdot n_2, k_1 \cdot k_2)$ - codului - tablou reprezentat de reţ ea.

Pentru decodificare se poate folosi o variantă a algoritmului Viterbi, cu mici modificări, necesare trecerii de la codurile convoluţ ionale la coduri bloc.

1. Cuvântul recepţ ionat este împărţ it în n_2 subcuvinte, fiecare având n_1 valori digitizate (vezi Exemplul 18.4) ale alfabetului - cod.
2. La fiecare nivel p, pentru fiecare arc (A_p, B_{p+1}) se calculează *metrica arcului*, care este distanţ a euclidiană dintre al $p+1$-lea subcuvânt recepţ ionat şi $l_p(A_p, B_{p+1})$. Distanţ a euclidiană dintre $\mathbf{x} = x_1x_2\ldots x_n$ şi $\mathbf{y} = y_1y_2\ldots y_n$ este
$$d_E(\mathbf{x}, \mathbf{y}) = (x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2.$$
3. Pentru fiecare nod B al reţ elei se determină *metrica nodului* B, astfel:
   (a) Metrica nodului rădăcină este 0;
   (b) Pentru un nod $B_{p+1} \in V_{p+1}$ $(p \geq 0)$, metrica lui B este cea mai mică valoare a sumei dintre metrica unui nod $A_p \in V_p$ şi $d_E((A)_p, (B)_{p+1})$, minimul fiind luat după toate nodurile din V_p.
4. Arcul selectat în calculul metricii nodului se introduce într-un *drum posibil*, iar toate celelalte arce de pe nivelul respectiv sunt eliminate.
5. După ce s-a calculat metrica nodului ţ intă, se consideră drumul posibil corespunzător şi decodificarea se face prin concatenarea sub-mesajelor de informaţ ie care marchează arcele acestui drum.

Exemplul 18.4 *Să reluăm codul - tablou definit în Exemplul 18.2. Presupunem că s-a transmis cuvântul cod* $\mathbf{a} = 011101110$ *şi s-a recepţionat mesajul digitizat*

$$\mathbf{b}" = 0.2\ \ 0.4\ \ 0.4\ \ 0.3\ \ 0.1\ \ 0.9\ \ 1.0\ \ 0.9\ \ 0.6.$$

Presupunând că orice simbol digitizat din intervalul $[0.0, 0.5)$ *reprezintă caracterul binar* 0*, iar* $[0.5, 1.0]$ *reprezintă* 1*, putem considera că s-a primit secvenţ a* $\mathbf{b} = 000001111$ *deci au apărut* 4 *erori. Într-o decodificare obişnuită a codurilor liniare, deoarece* $d = 4$*, erorile nu pot fi corectate. Folosind structura de reţ ea însă, acest lucru este realizabil. Să detaliem paşii decodificării cuvântului* $\mathbf{b}"$*.*

– *Metrica nodului rădăcină* $(00)_0$ *este* 0*.*

– *Prima subsecvenţă de* $n_1 = 3$ *caractere primite este* 0.2 0.4 0.4*.*

– *Se calculează metricile celor patru arce care pleacă din rădăcină:*

$$d_E((00)_0, (00)_1) = (0.2 - 0.0)^2 + (0.4 - 0.0)^2 + (0.4 - 0.0)^2 = 0.36;$$
$$d_E((00)_0, (01)_1) = (0.2 - 0.0)^2 + (0.4 - 1.0)^2 + (0.4 - 1.0)^2 = 0.76;$$
$$d_E((00)_0, (10)_1) = (0.2 - 1.0)^2 + (0.4 - 0.0)^2 + (0.4 - 1.0)^2 = 1.16;$$
$$d_E((00)_0, (11)_1) = (0.2 - 1.0)^2 + (0.4 - 1.0)^2 + (0.4 - 0.0)^2 = 1.16.$$

– *Metricile celor patru noduri de pe nivelul* 1 *sunt chiar aceste valori.*

– *Următorul submesaj primit este* 0.3 0.1 0.9. *Pentru nodurile de pe nivelul* 2, *valorile calculate ale metricilor (metrica nodului predecesor plus metrica arcului) sunt:*

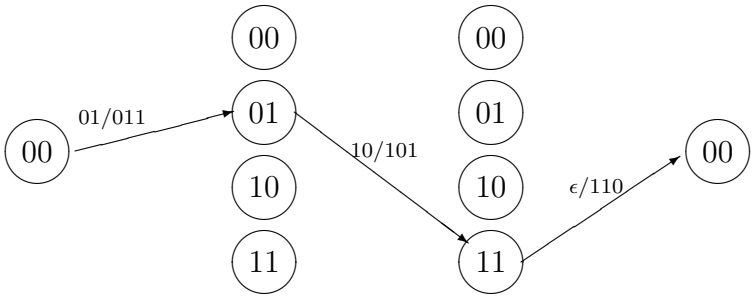
	$(00)_2$	$(01)_2$	$(10)_2$	$(11)_2$
$(00)_1$	**1.27**	**1.27**	**0.87**	2.42
$(01)_1$	1.67	1.67	2.87	**1.37**
$(10)_1$	1.67	3.27	2.07	2.07
$(11)_1$	3.27	1.67	2.07	2.07

– *Minimul pe fiecare coloană ale acestor valori (scris îngroşat) reprezintă metrica nodului respectiv. Deci nodurile* $(00)_2$ *şi* $(01)_2$ *au metrica* 1.27, $(10)_2$ *are metrica* 0.87, *iar* $(11)_2$ *are metrica* 1.37.

– *Ultima subsecvenţă de trei caractere digitizate este* 1.0 0.9 0.6. *Valorile calculate ale metricilor (metrica nodului predecesor plus metrica arcului) drumurilor care ajung în nodul ţ intă sunt:*

	$(00)_3$
$(00)_2$	3.44
$(01)_2$	2.44
$(10)_2$	1.84
$(11)_2$	**1.74**

Deci metrica nodului ţintă este 1.74. *Refăcând acum traiectoria în sens invers, se obţ ine drumul posibil* $(00)_0 - (01)_1 - (11)_2 - (00)_3$, *reprezentat în figura*



Prin concatenarea marcajelor arcelor de pe acest drum, rezultă cuvântul - cod 011101110 *(care coincide cu mesajul transmis* **a***) şi decodificarea lui - mesajul de informaţ ie* 0110.

18.4 Coduri tablou generalizate (GAC)

Definiţ ia codurilor - tablou se poate extinde pentru a cuprinde şi alte clase de coduri liniare (Hamming, Reed - Muller) sau ciclice (BCH, Reed - Solomon); ceea ce se obţ ine este un *cod - tablou generalizat* (GAC - Generalized Array Codes).

Un GAC este o sumă binară de două sau trei coduri - tablou, completate eventual cu linii nule. Procedura de construcţ ie a unui astfel de (n, k) - cod C este:

A. Fie C_1 un cod - produs prin înmulţ irea Kronecker $G_{A_1}'timesG_{A_2}$, unde:

(i) G_{A_1} este o matrice $k_1'times(k_1 + 1)$ de tipul (1);

(ii) $G_{A_2} = (I_{k_2}|J_{k_2,n_2-k_2})$ unde I_{k_2} este matricea unitate iar J este o matrice cu toate elementele 1;

(iii) $n = (k_1 + 1) \cdot n_2,\ k_A = k_1 \cdot k_2$.

B. Fie C_2 un cod - produs prin înmulţ irea Kronecker $G_{B_1}'timesG_{B_2}$, unde:

(i) G_{B_1} este o matrice $1'timesn_{B_1}$ cu toate elementele 1;

(ii) $G_{B_2} = (G_3|G_4)$ este o matrice $k_B'timesn_{B_2}$ unde G_3 este o matrice cu coloane 0 sau 1, iar G_4 este o matrice de tipul (1) cu k_B linii. Coloanele lui G_{B_2} pot fi eventual permutate.

(iii) $n = n_{B_1} \cdot n_{B_2}$.

C. Fie C_3 un cod - produs prin înmulţ irea Kronecker $G_{C_1}'timesG_{C_2}$, unde:

(i) $G_{C_1} = (00\ldots01)$ are dimensiunea $1'timesn_{C_1}$;

(ii) $G_{C_2} = (11\ldots1)$ are dimensiunea $1'timesn_{C_2}$;

(iii) $n = n_{C_1} \cdot n_{C_2}$.

D. Codul C este $C_1 + C_2$ sau $C_1 + C_2 + C_3$ unde suma se efectuează în Z_2; el are lungimea n, $k = k_A + k_B$ respectiv $k = k_A + k_B + 1$ simboluri de informaţ ie şi este generat de matricea

$$\begin{pmatrix} G_{A_1}'timesG_{A_2} \\ G_{B_1}'timesG_{B_2} \end{pmatrix} \qquad \text{sau} \qquad \begin{pmatrix} G_{A_1}'timesG_{A_2} \\ G_{B_1}'timesG_{B_2} \\ G_{C_1}'timesG_{C_2} \end{pmatrix}$$

În exemplele următoare vom prezenta sub formă de coduri GAC unele din cele mai cunoscute coduri - bloc (liniare sau ciclice).

Exemplul 18.5 *Să considerăm*

$$G_{A_1} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \quad G_{A_2} = (1\ 1)$$

$$G_{B_1} = (1\ 1\ 1\ 1), \quad G_{B_2} = (0\ 1).$$

Dimensiunile codului sunt $n = 4 \cdot 2 = 8,\ k = 3 + 1 = 4$. *Matricea generatoare va fi*

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Aceasta corespunde unui cod Reed - Muller $\mathcal{RM}(1,3)$. *Dacă se ignoră ultima coloană (de control), se obţ ine un* $(7,4)$ *- cod Hamming binar.*

Aranjarea sub formă de tablou a cuvintelor - cod din cele două coduri - produs componente este următoarea:

Fie $\mathbf{a} = a_1a_2a_3a_4$ *un mesaj de informaţ ie. Atunci cuvântul - cod* $\mathbf{x}$ *este dat prin:*

$$\mathbf{x_1} = \begin{matrix} a_1 & p_1 \\ a_2 & p_2 \\ a_3 & p_3 \\ p_4 & p_5 \end{matrix} \qquad \mathbf{x_2} = \begin{matrix} 0 & a_4 \\ 0 & a_4 \\ 0 & a_4 \\ 0 & a_4 \end{matrix},$$

$$\mathbf{x} = \mathbf{x_1} + \mathbf{x_2} = \begin{matrix} a_1 & p_1 + a_4 \\ a_2 & p_2 + a_4 \\ a_3 & p_3 + a_4 \\ p_4 & p_4 + a_4 \end{matrix} = (a_1, a_4 + p_1, a_2, p_2 + a_4, a_3, p_3 + a_4, p_4, p_4 + a_4),$$

unde $p_i = a_i\ (1 \le i \le 3)$ *şi* $p_4 = a_1 + a_2 + a_3$ *sunt simboluri de control.*

Exemplul 18.6 *Codul* $\mathcal{RM}(1,4)$ *poate fi generat cu un cod GAC astfel:*

$$G_{A_1} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \qquad G_{A_2} = (1\ 1\ 1\ 1)$$

$$G_{B_1} = (1\ 1\ 1\ 1) \qquad G_{B_2} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

Deci $n = 4 \cdot 4,\ k = 3 + 2 = 5$ *şi*

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Eliminând ultimul simbol de control (ultima coloană din matricea G*) se ajunge la un* $(15, 5)$ *- cod binar cu* $d = 7$*, care este un cod* BCH*.*

Fie $\mathbf{a} = a_1a_2a_3a_4a_5$ *un mesaj de informaţ ie. Reprezentarea tabelară a cuvintelor - cod este:*

$$\mathbf{x_1} = \begin{matrix} a_1 & p_1 & p_1 & p_1 \\ a_2 & p_2 & p_2 & p_2 \\ a_3 & p_3 & p_3 & p_3 \\ p_4 & p_4 & p_4 & p_4 \end{matrix} \qquad \mathbf{x_2} = \begin{matrix} 0 & a_4 & a_5 & a_4 + a_5 \\ 0 & a_4 & a_5 & a_4 + a_5 \\ 0 & a_4 & a_5 & a_4 + a_5 \\ 0 & a_4 & a_5 & a_4 + a_5 \end{matrix},$$

$$\mathbf{x} = \mathbf{x_1} + \mathbf{x_2} = \begin{matrix} a_1 & p_1 + a_4 & p_1 + a_5 & p_1 + a_4 + a_5 \\ a_2 & p_2 + a_4 & p_2 + a_5 & p_2 + a_4 + a_5 \\ a_3 & p_3 + a_4 & p_3 + a_5 & p_3 + a_4 + a_5 \\ p_4 & p_4 + a_4 & p_4 + a_5 & p_4 + a_4 + a_5 \end{matrix},$$

unde $p_i = a_i\ (1 \leq i \leq 3),\ p_4 = a_1 + a_2 + a_3$ *sunt simboluri de control.*

Exemplul 18.7 *Să plecăm de la următoarele matrici de bază:*

$$G_{A_1} = G_{A_2} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

$$G_{B_1} = G_{C_2} = (1\ 1\ 1\ 1), \quad G_{B_2} = G_{C_1} = (0\ 0\ 0\ 1).$$

Se obţ ine un cod $C = C_1 + C_2 + C_3$ *cu* $n = 4 \cdot 4 = 16,\ k = 3 \cdot 3 + 1 + 1 = 11$ *şi matrice generatoare*

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

El are $d = 4$ *şi este echivalent cu un cod* $\mathcal{RM}(2, 4)$*. Dacă se elimină ultimul simbol de control, se obţ ine un* $(15, 11)$ *- cod Hamming binar.*

Fie $\mathbf{a} = a_1a_2 \ldots a_{11}$ *un mesaj de informaţ ie. Cuvântul - cod* $\mathbf{x}$ *este format din:*

$$\mathbf{x_1} = \begin{matrix} a_1 & a_2 & a_3 & p_1 \\ a_4 & a_5 & a_6 & p_2 \\ a_7 & a_8 & a_9 & p_3 \\ p_4 & p_5 & p_6 & p_7 \end{matrix}, \ \mathbf{x_2} = \begin{matrix} 0 & 0 & 0 & a_{10} \\ 0 & 0 & 0 & a_{10} \\ 0 & 0 & 0 & a_{10} \\ 0 & 0 & 0 & a_{10} \end{matrix}, \ \mathbf{x_3} = \begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ a_{11} & a_{11} & a_{11} & a_{11} \end{matrix}$$

$$\mathbf{x} = \mathbf{x_1} + \mathbf{x_2} + \mathbf{x_3} = \begin{matrix} a_1 & a_2 & a_3 & p_1 + a_{10} \\ a_4 & a_5 & a_6 & p_2 + a_{10} \\ a_7 & a_8 & a_9 & p_3 + a_{10} \\ p_4 + a_{11} & p_5 + a_{11} & p_6 + a_{11} & p_7 + a_{10} + a_{11} \end{matrix}$$

Exemplul 18.8 *Pentru* $(24, 12)$ *- codul Golay binar extins se poate da următoarea reprezentare GAC:*

$$G_{A_1} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \quad G_{A_2} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$G_{B_1} = (1\ 1\ 1) \quad G_{B_2} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$G_{C_1} = (0\ 0\ 1) \quad G_{C_2} = (1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)$.

Fie $\mathbf{a} = a_1 a_2 \dots a_{12}$ *un mesaj de informaţ ie.*

Pentru obţ inerea cuvântului - cod, avem:

$$\mathbf{x_1} = \begin{matrix} a_1 & a_2 & a_3 & a_4 & p_1 & p_2 & p_3 & p_4 \\ a_5 & a_6 & a_7 & a_8 & p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} & p_{13} & p_{14} & p_{15} & p_{16} \end{matrix}$$

$$\mathbf{x_2} = \begin{matrix} c_1 & a_9 & c_2 & a_{10} & c_3 & c_4 & a_{11} & c_5 \\ c_1 & a_9 & c_2 & a_{10} & c_3 & c_4 & a_{11} & c_5 \\ c_1 & a_9 & c_2 & a_{10} & c_3 & c_4 & a_{11} & c_5 \end{matrix}$$

$$\mathbf{x_3} = \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{12} & a_{12} & a_{12} & a_{12} & a_{12} & a_{12} & a_{12} & a_{12} \end{matrix}$$

deci

$$\mathbf{x} = \mathbf{x_1} + \mathbf{x_2} + \mathbf{x_3} = \begin{matrix} a_1 + c_1 & a_2 + a_9 & \dots & a_{11} + p_3 & c_5 + p_4 \\ a_5 + c_1 & a_6 + a_9 & \dots & a_{11} + p_7 & c_5 + p_8 \\ a_{12} + p_9 + c_1 & a_{12} + p_{10} + a_9 & \dots & a_{12} + a_{11} + p_{15} & a_{12} + p_{16} + c_5 \end{matrix}$$

De remarcat maniera mult mai simplă de codificare, ea reducându-se la adunări de valori binare (pentru simbolurile de control) şi adunări de matrici binare.

18.5 Decodificarea codurilor *GAC*

Pentru decodificare se poate folosi tot algoritmul lui Viterbi aplicat unei structuri de reţ ea sub care se pot reprezenta aceste coduri. Structura de reţ ea se construieşte similar cu cea a codurilor - tablou, cu unele mici modificări datorate definiţ iei cuvintelor - cod, ca sume de două sau trei alte secvenţ e.

Vom considera reprezentarea cuvintelor unui cod *GAC* sub formă de tablou cu n_1 linii şi n_2 coloane. Fie k_i numărul de simboluri noi de informaţ ie care pot apare pe linia i a

cuvintelor - cod şi $t = \max\limits_{1 \le i \le n_1} \{k_i\}$.

1. Numărul de coloane (adâncimea reţ elei) este $N_c = n_1 + 1$, iar numărul de stări distincte este $N_s = 2^t$.
2. Fiecare stare de pe coloana p se notează $(a_1 a_2 \dots a_t)_p$ unde $a_i \in Z_2$ $(1 \le i \le t)$.
3. Rădăcina reţ elei este $(00\dots0)_0$ iar ţ inta $(00\dots0)_{n_1}$.
4. Fiecare arc (A_p, B_{p+1}) de pe nivelul p este marcat cu o pereche $\delta_p(A,B)/l_p(A,B)$ unde $\delta_p(A,B)$ este secvenţ a de simboluri noi de informaţ ie de pe linia p (eventual ϵ), iar $l_p(A,B)$ este linia p a cuvântului - cod.
5. Dacă A_p şi B_{p+1} sunt legate printr-un arc şi $|\delta_p(A,B)| = s$, atunci $B_{p+1} = A_p + \delta_p(A,B)0^{t-s}$.
6. Dacă construcţ ia codului GAC a folosit şi codul C_3, atunci fiecare nod de pe coloana n_2 este legat de ţ intă prin două arce, marcajul celui de-al doilea arc fiind complementara marcajului primului arc (pe acest nivel $\delta_{n_2}(A, 00\dots0) = \epsilon$).

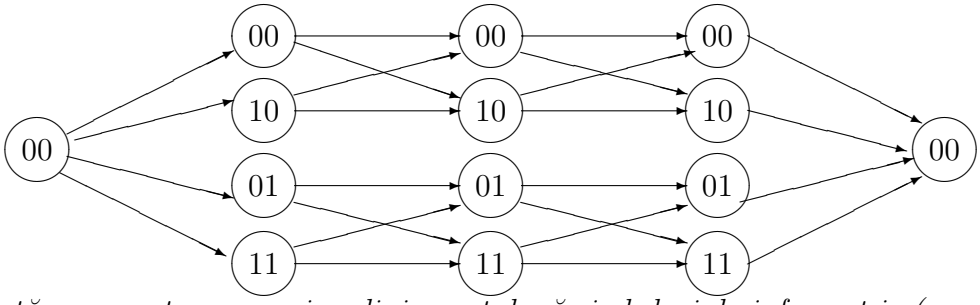
În această construcţ ie de reţ ea sunt $\prod\limits_{p=1}^{n_1} 2^{k_i}$ drumuri distincte, fiecare din ele corespunzând unui cuvânt - cod.

Operaţ iile de codificare/decodificare (inclusiv aplicarea algoritmului Viterbi a-daptat) sunt identice cu cele definite la codurile - tablou.

Exemplul 18.9 *Reluăm codul definit în Exemplul 18.5. Pentru el se poate construi o reţ ea în felul următor:*

Sunt necesare $N_c = n_1 + 1 = 4 + 1 = 5$ *coloane. Pentru numărul de stări, să studiem forma de tablou a cuvintelor cod:*

$$\mathbf{x} = \begin{matrix} a_1 & p_1 + a_4 \\ a_2 & p_2 + a_4 \\ a_3 & p_3 + a_4 \\ p_4 & p_4 + a_4 \end{matrix}$$



În această reprezentare, pe prima linie sunt două simboluri de informaţ ie (a_1 *şi* a_4*), pe a doua şi a treia câte unul (*a_2 *respectiv* a_3*,* a_4 *nefiind simbol nou), iar pe a patra linie, nici*

unul. Deci numărul de stări al reţ elei este $N_s = 2^2 = 4$*, fiecare stare fiind reprezentată printr-o secvenţă de* $t = 2$ *caractere binare (*$00, 01, 10, 11$*).*

Funcţ ia δ *este definită prin*

$\delta_0(00, B) = a_1a_4, \quad \delta_1(A, B) = a_2, \quad \delta_2(A, B) = a_3, \quad \delta_3(A, 00) = \epsilon.$

Nodurile şi arcele sunt marcate cu valorile date de tabelele următoare:

$\delta_0(A,B)/l_0(A,B)$	00	10	01	11
00	00/00	10/11	01/01	11/10

$\delta_1(A,B)/l_1(A,B)$	00	10	01	11
00	0/00	1/11	–	–
10	1/11	0/00	–	–
01	–	–	0/01	1/10
11	–	–	1/10	0/01

$\delta_2(A,B)/l_2(A,B)$	00	10	01	11
00	0/00	1/11	–	–
10	1/11	0/00	–	–
01	–	–	0/01	1/10
11	–	–	1/10	0/01

$\delta_3(A,B)/l_3(A,B)$	00
00	$\epsilon/00$
10	$\epsilon/11$
01	$\epsilon/01$
11	$\epsilon/10$

pentru codul $\mathcal{RM}(1, 3)$*,*

$\delta_3(A,B)/l_3(A,B)$	00
00	$\epsilon/0$
10	$\epsilon/1$
01	$\epsilon/0$
11	$\epsilon/1$

pentru $(7, 4)$ *- codul Hamming*

Exemplul 18.10 *Reprezentarea prin reţ ea a codurilor* $\mathcal{RM}(1, 4)$ *şi* $(15, 5)$ *– BCH construite sub formă de GAC în Exemplul 18.6 se realizează astfel:*

Adâncimea reţ elei (numărul de coloane) este $N_c = 4 + 1 = 5$*. Numărul maxim de simboluri de informaţ ie noi se află pe prima linie a cuvintelor - cod* (a_1, a_4, a_5)*; deci reţ eaua are* $N_s = 2^3 = 8$ *stări distincte.*

Funcţ ia δ *este definită*

$\delta_0(000, B) = a_1a_4a_5, \ \delta_1(A, B) = a_2, \ \delta_2(A, B) = a_3, \ \delta_3(A, 000) = \epsilon.$

iar funcţ ia l*:*

$l_0(000, B) = a_1(a_1 + a_4)(a_1 + a_5)(a_1 + a_4 + a_5)$

$l_1(A, B) = a_2(a_2 + a_4)(a_2 + a_5)(a_2 + a_4 + a_5)$

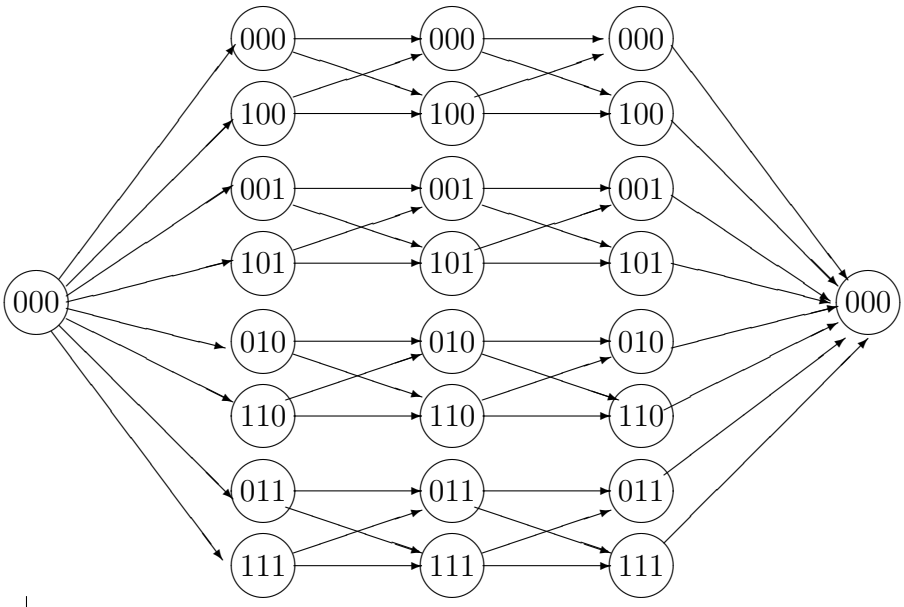
$l_2(A, B) = a_3(a_3 + a_4)(a_3 + a_5)(a_3 + a_4 + a_5)$

$l_3(A, 000) = (a_1 + a_2 + a_3)(a_1 + a_2 + a_3 + a_4)(a_1 + a_2 + a_3 + a_5)(a_1 + a_2 + a_3 + a_4 + a_5)$

pentru codul $\mathcal{RM}(1, 4)$*. Pentru* $(15, 5)$ *- codul BCH se elimină ultimul bit de control.*

Nodurile şi arcele sunt etichetate conform tabelelor:

δ_0/l_0	000	100	001	101	010	110	011	111
000	000/0000	100/1111	001/0011	101/1100	010/0101	110/1010	011/0110	111/1001



δ_i/l_i	000	100	001	101	010	110	011	111
000	0/0000	1/1111	—	—	—	—	—	—
100	1/1111	0/0000	—	—	—	—	—	—
001	—	—	0/0011	1/1100	—	—	—	—
101	—	—	1/1100	0/0011	—	—	—	—
010	—	—	—	—	0/0101	1/1010	—	—
110	—	—	—	—	1/1010	0/0101	—	—
011	—	—	—	—	—	—	0/0110	1/1001
111	—	—	—	—	—	—	1/1001	0/0110

(i=1,2)

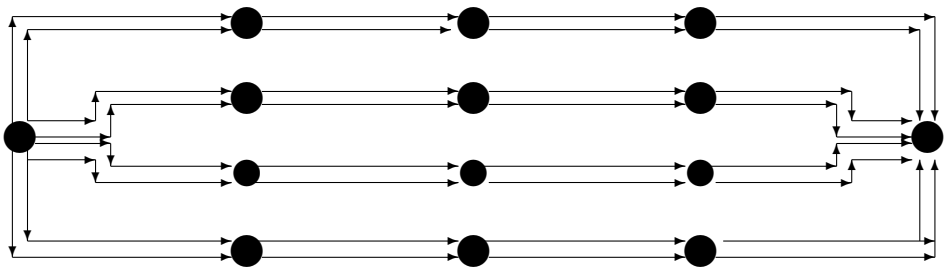
δ_3/l_3	0000
000	ϵ/0000
100	ϵ/1111
001	ϵ/0011
101	ϵ/1100
010	ϵ/0101
110	ϵ/1010
011	ϵ/0110
111	ϵ/1001

respectiv

δ_3/l_3	0000
000	ϵ/000
100	ϵ/111
001	ϵ/001
101	ϵ/110
010	ϵ/010
110	ϵ/101
011	ϵ/011
111	ϵ/100

.

De remarcat că această reţ ea se poate reprezenta folosind numai 4 *stări în loc de* 8*:*



Cele două arce care leagă două noduri sunt marcate cu etichete complementare. Excepţ ie fac arcele de pe primul nivel unde în definiţ ia funcţ iei δ_0 *este complementat numai primul bit* (a_1)*; ceilalţ i doi biţ i de informaţ ie* a_4a_5 *sunt identici pentru fiecare pereche de arce şi formează marcajele nodurilor succesor de pe nivelul* 1*:* $00, 01, 10, 11$.

Capitolul 19

Alte reprezentări de coduri - bloc

19.1 *RLL* - coduri

Sistemele de comunicare actuale (telefonice, de înregistrare etc), datorită vitezei mari de lucru, diferă substanţ ial de canalele clasice (binar simetrice). Pentru ele se folosesc filtre de bandă capabile să elimine fenomenul de interferenţă care apare. Numite *canale cu răspuns parţ ial*, ele corectează aceste interferenţ e folosind algoritmi de tip Viterbi, incorporaţ i prin construcţ ie. Reuşita decodificării corecte a cuvintelor recepţ ionate este asigurată de supra-codificări ale mesajului sursă. În prezent sunt folosite trei supra-coduri care asigură transmiterea fără interferenţ e (în anumite limite) a mesajului: acestea sunt codurile cu lungime limitată şi codurile balansate pentru codurile-bloc, turbo-codurile pentru ocdurile convoluţ ionale.

Definiţia 19.1 *Un cod cu lungime limitată (RLL - Run Length Limited Codes), sau* (d^0, k^0) *- cod, este un cod - bloc cu proprietatea că orice două simboluri* 1 *consecutive sunt separate prin p zerouri, unde* $d^0 \leq p \leq k^0$.

Parametrul d^0 este folosit pentru controlul interferenţ ei dintre simboluri (feno-men care apare la viteze de transmisie mari, apropiate de capacitatea de saturaţ ie a canalului); k^0 impune numărul maxim de zerouri care pot apare, pentru păstrarea unei rate de informaţ ie cât mai convenabile.

Fie C un (n, k) - cod bloc care poate fi reprezentat sub formă de cod tablou $(n_1 \cdot n_2, k_1 \cdot k_2)$, şi $\mathbf{c}_{1,n}$ un cuvânt binar *de comutaţ ie*. Codul $C + \mathbf{c} = \{\mathbf{a} + \mathbf{c} | \mathbf{a} \in A\}$ va fi tot un (n, k) - cod, cu restricţ ii *RLL*.

Dificultatea acestei construcţ ii rezidă în modul de alegere a vectorului $\mathbf{c}$. De aceea, vom detalia procedura de codificare pentru obţ inerea de coduri *RLL* cu $d^0 = 0$:

1. Se defineşte codul - tablou $(n_1 \cdot n_2, k_1 \cdot k_2)$ care trebuie transformat.
2. Se defineşte cuvântul de comutaţ ie $\mathbf{c} = \mathbf{c_1 c_2} \ldots \mathbf{c_{n_2}}$, unde $\mathbf{c_i} = c_{i_1} c_{i_2} \ldots c_{i_{n_1}}$ $(1 \leq i \leq n_2)$ este secvenţ a binară de comutaţ ie pentru linia i.
3. Pentru linia i (care este un cuvânt - cod cu un singur simbol de control), cuvântul de comutaţ ie verifică următoarea condiţ ie:
   Dacă n_1 este par, $\mathbf{c_i}$ are un număr impar de 1,
   altfel, $\mathbf{c_i}$ are un număr par de 1.

Exemplul 19.1 *Fie* $(16, 9)$ *- codul tablou, având cuvintele - cod de forma*

$$\mathbf{a} = \begin{matrix} a_1 & a_2 & a_3 & p_1 \\ a_4 & a_5 & a_6 & p_2 \\ a_7 & a_8 & a_9 & p_3 \\ p_4 & p_5 & p_6 & p_7 \end{matrix}$$

(este codul C_1 construit în Exemplul 18.7). Deoarece $n_1 = 4$, cuvântul de comutaţ ie al fiecărei linii conţ ine un număr impar de 1. Vor exista opt variante posibile pentru $\mathbf{c_i}$:

$$0001 \quad 0010 \quad 0100 \quad 1000 \quad 1110 \quad 1101 \quad 1011 \quad 0111,$$

deci se pot defini $N_0 = 8^4$ cuvinte de comutaţ ie distincte $\mathbf{c}$.

Dacă se ia de exemplu $\mathbf{c} = 0100110110001110$, codul RLL definit pe baza lui va avea cuvintele - cod

$$\mathbf{a} = \begin{matrix} a_1 & \overline{a_2} & a_3 & p_1 \\ \overline{a_4} & \overline{a_5} & a_6 & \overline{p_2} \\ \overline{a_7} & a_8 & a_9 & p_3 \\ \overline{p_4} & \overline{p_5} & \overline{p_6} & p_7 \end{matrix}$$

unde s-a notat $\overline{x} = 1 - x$.

Tehnica descrisă mai sus se poate extinde la coduri GAC, conducând la coduri cu proprietăt i RLL optime ([8]). Generalizarea este următoarea:

1. Dacă n_1 este par, $\mathbf{c_i}$ $(1 \leq i \leq n_2)$ conţ ine zero sau un număr impar de 1;
2. Dacă n_2 este impar, $\mathbf{c_i}$ $(1 \leq i \leq n_2)$ conţ ine un număr par de 1;
3. Dacă $n = n_1 \cdot n_2$ este par, cuvântul de comutaţ ie $\mathbf{c}$ va conţ ine un număr par de 1;
4. $\mathbf{c_i}$ nu este cuvânt - cod al codului care formează liniile; dacă aceasta nu se poate evita, atunci $\mathbf{c_{i+1}}$ se alege de asemenea dintre cuvintele aceluiaşi cod.

Exemplul 19.2 *Fie un $(8, 4)$ - cod binar care codifică fiecare mesaj de informaţ ie (a_1, a_2, a_3, a_4) în*

$$\mathbf{a} = \begin{matrix} a_1 & a_1 + a_4 \\ a_2 & a_2 + a_4 \\ a_3 & a_3 + a_4 \\ p_1 & p_1 + a_4 \end{matrix}$$

Deoarece n şi n_1 sunt pare, alegem $\mathbf{c_i}$ $(1 \leq i \leq 4)$ cu număr impar de 1, astfle încât numărul total de 1 din $\mathbf{c}$ să fie par. Putem lua de exemplu

$$\mathbf{c_1} = 10, \quad \mathbf{c_2} = 00, \quad \mathbf{c_3} = 01, \quad \mathbf{c_4} = 00$$

Deci secvenţ a generală de comutaţ ie este $\mathbf{c} = 10000100$ şi cuvintele - cod cu restricţ ia RLL sunt

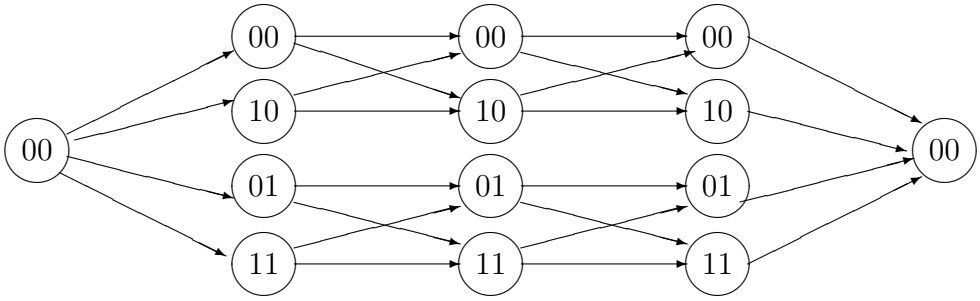
$$\mathbf{a} = \begin{matrix} \overline{a_1} & a_1 + a_4 \\ a_2 & a_2 + a_4 \\ a_3 & \overline{a_3 + a_4} \\ p_1 & p_1 + a_4 \end{matrix}$$

Studiind toate cele 16 cuvinte - cod, se găsesc trei cuvinte cod care încep cu trei caractere 1 şi trei cuvinte - cod care se termină cu trei caractere 1. Deci $k^0 = 6$ şi se poate demonstra că aceasta este valoarea minimă care se poate obţ ine pentru k^0 în cazul $(8, 4)$ - codurilor.

Dacă se relaxează codul prin eliminarea ultimului simbol de control, se ajunge la un $(7, 4)$ - cod Hamming binar, cu $k^0 = 7$.

Pentru decodificarea codurilor *RLL* se poate folosi structura de reţ ea definită în cazul codurilor *GAC*, cu o singură modificare: la marcajul arcelor, se înlocuieşte $l_p(A,B) := l(A,B) + \mathbf{c_p}$.

Exemplul 19.3 *Să reluăm* $(8,4)$ *- codul GAC de mai sus, a cărui reţ ea este descrisă în Exemplul* 18.5. *Folosind secvenţ a de comutaţ ie* $\mathbf{c} = 00010001$*, se obţ ine un cod RLL cu aceeaşi structură de reţ ea:*



iar marcajele arcelor:

δ_0/l_0	00	10	01	11
00	00/00	10/11	01/01	11/10

δ_1/l_1	00	10	01	11
00	0/01	1/10	–	–
10	1/10	0/01	–	–
01	–	–	0/01	1/11
11	–	–	1/11	0/00

δ_2/l_2	00	10	01	11
00	0/00	1/11	–	–
10	1/11	0/00	–	–
01	–	–	0/01	1/10
11	–	–	1/10	0/01

δ_3/l_3	00
00	ϵ/01
10	ϵ/10
01	ϵ/00
11	ϵ/11

(pentru $(7,4)$ *- codul Hamming binar se ignoră ultimul bit din* l_3*).*

De remarcat că faţă de Exemplul 18.5 *s-a modificat numai marcajul nodurilor de pe nivelele* 1 *şi* 3*).*

19.2 Coduri balansate

Codurile balansate reprezintă o clasă specială de coduri *RLL* în care fiecare cuvânt - cod are un număr egal de 0 şi 1. Avantajele pe care le oferă această condiţ ie (siguranţ a la transmiterea de date la viteze mari, capacitatea de auto-control al tacţ ilor, posibilităt i de detectare şi corectare a erorilor comparabile cu codurile - bloc) o recomandă ca o măsură eficientă de codificare suplimentară.

Algoritm **B**:
Fie $\mathbf{a} = a_1 a_2 \dots a_k$ un mesaj de informaţ ie. Definim următorul cuvânt - cod tablou balansat asociat lui **a**:
1. Se alege n_1 astfel ca $2n_1 - 2 \leq k$ şi se notează $n = 4 \cdot n_1$.
2. Se construieşte un cuvânt - cod tablou cu primele $2n_1 - 2$ caractere de informaţ ie:

$$\mathbf{c} = \begin{array}{ccccccc} a_1 & a_3 & a_5 & \dots & a_{2n_1-3} & p_1 \\ \overline{a_1} & \overline{a_3} & \overline{a_5} & \dots & \overline{a_{2n_1-3}} & \overline{p_1} \\ a_2 & a_4 & a_6 & \dots & a_{2n_1-2} & p_2 \\ \overline{a_2} & \overline{a_4} & \overline{a_6} & \dots & \overline{a_{2n_1-2}} & \overline{p_2} \end{array}$$

Cuvintele de această formă, citite pe coloană, formează un $(n, 2n_1 - 2)$ - cod tablou balansat C, având $d = 4$. C este folosit pentru definirea caracterelor de control p_1 şi p_2.
3. Se definesc funcţ iile $f_1, f_2 : Z_2 \to Z_2^4$ prin
$f_1(x, y) = (x, \overline{x}, y, \overline{y})$
$f_2(x, y) = (x \wedge y, x \wedge \overline{y}, \overline{x} \wedge y, \overline{x} \wedge \overline{y})$
unde $\wedge$ este operatorul boolean obişnuit. Evident, coloanele codului C sunt codificate folosind f_1.
4. Se construieşte cuvântul - cod **u** folosind ultimele $k - 2n_1 + 2$ caractere de informaţ ie $\mathbf{x} = a_{2n_1-1} a_{2n_1} \dots a_k$ astfel:

1. Dacă $\mathbf{x} = 0 \dots 00$, atunci $\mathbf{u} = f_1(a_1, a_2) f_1(a_3, a_4) \dots f(p_1, p_2)$;
2. Dacă $\mathbf{x} = 0 \dots 01$, atunci $\mathbf{u} = f_1(a_1, a_2) f_1(a_3, a_4) \dots \overline{f_1(p_1, p_2)}$;
3. În rest, $\mathbf{u} = g(a_1, a_2) g(a_3, a_4) \dots g(p_1, p_2)$, unde

   (i) g este f_2 sau $\overline{f_2}$; fiecare din ele apare de un număr egal de ori (câte $n_1/2$ aparit ii);

   (ii) Dacă se notează f_2 cu 0 şi $\overline{f_2}$ cu 1, lui **u** i se asociază o secvenţă α de n_1 caractere binare, cu proprietatea că pentru primele $2n_1 - 2$ caractere de informaţ ie fixate, aplicaţ ia $\mathbf{x} \mapsto \alpha$ este izotonă faţă de relaţ ia de ordine lexicografică.

5. Cuvântul - cod final se obţ ine prin scrierea pe fiecare linie a unei valori $f_i(x, y)$ din **u**. El aparţ ine astfel unui cod $GAC \quad A$ de dimensiuni $n_1' times 4$.

Sunt multe modalităt i de transformare a unui cod - bloc binar într-un cod balansat; de exemplu dublarea fiecărui cuvânt - cod **a** cu o secvenţă identică (**aa**) sau cu caractere complementate ($\mathbf{a\overline{a}}$) conduc la coduri balansate. Construcţ ia unui cod balansat realizată prin Algoritmul **B** asigură posibilitatea construcţ iei unei reţ ele de decodificare în maniera celei din Prelegerea 18.

În construcţ ia dată de Algoritmul **B** mai trebuie rezolvată problema alegerii lui n_1. Aceasta se realizează folosind următoarea teoremă:

Teorema 19.1 *Fiind dat k, n_1 este de forma $n_1 = 2s$ unde s este cel mai mic număr pozitiv care verifică inegalitatea*

$$2^{k+2-4s} \leq 2 + C_{2s}^{s}. \qquad (1)$$

Demonstraţ ie: Condiţ ia ca n_1 să fie par rezultă din cerinţ a ca în $\mathbf{u}$ jumătate din cele n_1 funcţ ii g să fie f_2 şi jumătate să fie $\overline{f_2}$. Fie deci $n_1 = 2s$.

Din condiţ ia $2n_1 - 2 \leq k$ rezultă $k + 2 - 4p \leq 0$.

Deoarece $\mathbf{x}$ este format din $k+2-4s$ caractere binare, sunt 2^{k+2-4s} secvenţ e $\mathbf{x}$ posibile. Din acestea, $2^{k+2-4s} - 2$ sunt construite folosind numai funcţ ia f_2, deci trebuie să existe pentru ele cel puţ in tot atâtea secvenţ e α. Deoarece numărul secvenţ elor binare α este C_{2s}^{s}, rezultă inegalitatea din enunţ.

Condiţ ia de minim asigură unicitatea codificării. □

De remarcat că prin această metodă, operaţ ia de balansare nu păstrează proprietatea de cod liniar sau ciclic.

Exemplul 19.4 *Să construim un cod GAC balansat care să codifice $k = 3$ simboluri de informaţ ie. Folosind inegalitatea* (1)*, se obţ ine $s = 1$, deci $n_1 = 2$, $n = 8$.*

Codul cadru, care defineşte simbolurile de control, este un $(8, 2)$ *cod bloc având cuvintele - cod de forma (reamintim, reprezentarea desfăs urată se face pe coloane):*

$$\mathbf{c} = \begin{matrix} a_1 & p_1 \\ \overline{a_1} & \overline{p_1} \\ a_2 & p_2 \\ \overline{a_2} & \overline{p_2} \end{matrix} = a_1\ \overline{a_1}\ a_2\ \overline{a_2}\ p_1\ \overline{p_1}\ p_2\ \overline{p_2}$$

unde $p_1 = a_1$, $p_2 = a_2$. Operaţ ia de codificare este reprezentată de tabloul:

$000 \to 01010101$	$001 \to 01011010$
$010 \to 01100110$	$011 \to 01101001$
$100 \to 10011001$	$101 \to 10010110$
$110 \to 10101010$	$111 \to 10100101$

Exemplul 19.5 *Pentru $k = 4$ simboluri de informaţ ie se obţ ine de asemenea $s = 1$ şi acelaşi* $(8, 2)$ *- cod cadru, cu două simboluri de control. Cele două caractere de informaţ ie rămase, a_3a_4 asigură codificarea în felul următor:*

$$\begin{aligned} \alpha = 00 \quad &\Longrightarrow \quad \mathbf{u} = f(a_1, a_2)\underline{f(p_1, p_2)} \\ \alpha = 10 \quad &\Longrightarrow \quad \mathbf{u} = f(a_1, a_2)\overline{f(p_1, p_2)} \\ \alpha = 01 \quad &\Longrightarrow \quad \mathbf{u} = \underline{f_2(a_1, a_2)}f_2(p_1, p_2) \\ \alpha = 11 \quad &\Longrightarrow \quad \mathbf{u} = \overline{f_2(a_1, a_2)}f_2(p_1, p_2) \end{aligned}$$

Se obţ ine în acest fel un $(8, 4)$ *- cod balansat. Din studiul celor* 16 *cuvinte - cod rezultă că el are $d = 2$.*

Exemplul 19.6 *Următorul cod balansat a fost studiat de Blaum ([3]), odată cu introducerea metodei de balansare prezentată mai sus. Din $k = 9$, singura valoare posibilă este $s = 2$, deci $n_1 = 4$, $n = 16$. Cuvintele - cod cadru au forma*

$$\mathbf{c} = \begin{matrix} a_1 & a_3 & a_5 & p_1 \\ \overline{a_1} & \overline{a_3} & \overline{a_5} & \overline{p_1} \\ a_2 & a_4 & a_6 & p_2 \\ \overline{a_2} & \overline{a_4} & \overline{a_6} & \overline{p_2} \end{matrix}$$

unde $p_1 = a_1 + a_3 + a_5,\ p_2 = a_2 + a_4 + a_6$.

Procedura de codificare este dată de tabelul următor:

$a_7a_8a_9$	$u_1u_2u_3u_4$	$u_5u_6u_7u_8$	$u_9u_{10}u_{11}u_{12}$	$u_{13}u_{14}u_{15}u_{16}$
000	$f_1(a_1,a_2)$	$f_1(a_3,a_4)$	$f_1(a_5,a_6)$	$\overline{f_1(p_1,p_2)}$
001	$f_1(a_1,a_2)$	$f_1(a_3,a_4)$	$\overline{f_1(a_5,a_6)}$	$\overline{f_1(p_1,p_2)}$
010	$f_2(a_1,a_2)$	$\overline{f_2(a_3,a_4)}$	$\overline{f_2(a_5,a_6)}$	$\overline{f_2(p_1,p_2)}$
011	$f_2(a_1,a_2)$	$\overline{f_2(a_3,a_4)}$	$\overline{f_2(a_5,a_6)}$	$f_2(p_1,p_2)$
100	$\overline{f_2(a_1,a_2)}$	$\overline{f_2(a_3,a_4)}$	$\overline{f_2(a_5,a_6)}$	$\overline{f_2(p_1,p_2)}$
101	$\overline{f_2(a_1,a_2)}$	$f_2(a_3,a_4)$	$\overline{f_2(a_5,a_6)}$	$\overline{f_2(p_1,p_2)}$
110	$\overline{f_2(a_1,a_2)}$	$\overline{f_2(a_3,a_4)}$	$\overline{f_2(a_5,a_6)}$	$f_2(p_1,p_2)$
111	$\overline{f_2(a_1,a_2)}$	$\overline{f_2(a_3,a_4)}$	$f_2(a_5,a_6)$	$f_2(p_1,p_2)$

S-a obţ inut astfel un $(16, 9)$ *- cod balansat cu* $d = 4$.

Să codificăm mesajul de informaţ ie $\mathbf{a} = 001101010$. *Pentru determinarea biţ ilor de control se construieşte cuvântul*

$$\mathbf{c} = \begin{matrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{matrix}$$

Deoarece $a_7 = 0,\ a_8 = 1,\ a_9 = 0$, *codul va fi generat folosind a treia linie din tabelul de sus:*

$$\mathbf{u} = \begin{matrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{matrix} = 0001\ 1000\ 1101\ 1011.$$

Alte exemple de mesaje de informaţ ie şi codificările lor:

000000000	0101010101010101
111111001	1010101010100101
000000101	1110000100011110
111111011	1000011110000111

Pentru realizarea unei structuri de reţ ea a codurilor balansate, dificultatea constă în faptul că aceste coduri nu sunt liniare, deci algoritmii utilizaţ i până acum nu funcţ ionează. Se va utiliza următorul algoritm, specific codurilor balansate definite în această secţ iune:

1. Pe baza algoritmilor cunoscuţ i se construieşte structura de reţ ea R_b a codului - bloc cadru C_b; fie k_0 numărul ei de coloane (egal cu numărul de coloane din reprezentarea în tablou a cuvintelor - codului C_b).

2. Se copiază reţ eaua R_b de $2^p - 1$ ori, unde $p = k - 2n_1 + 2$ este dat de pasul 3 al algoritmului **B**; fiecare copie corespunde unei generări posibile a unui cuvânt - cod balansat.

   Toate subreţ elele au o singură rădăcină şi o singură ţ intă.

3. În fiecare din cele 2^p subreţ ele:

   (a) Arcul $((00\ldots0)_0, B_1)$ de pe nivelul 0 se marchează cu δ_0/l_0 unde $\delta_0(00\ldots0, B) = a_{2n_1-1}\ldots a_k$, iar $l_0(00\ldots0, B) = g_1(a_1, a_2)$.

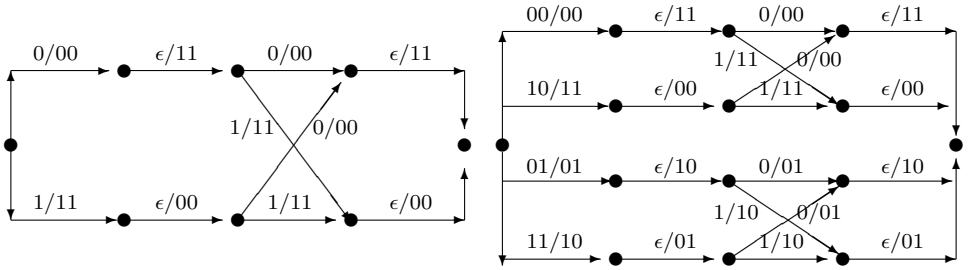
   (b) Arcul spre ţ intă are $\delta_{k_0}(A, 00\ldots0) = \epsilon$ şi $l_{k_0}(A, 00\ldots0) = g(p_1, p_2)$. (notaţ iile sunt cele din algoritmul **B**).

4. În fiecare subreţ ea se defineşte valoarea δ_j/l_j $(1 \le j < k_0)$ a arcului (A_j, B_{j+1}) astfel: $\delta_j(A, B)$ este valoarea nemodificată din R_b, iar $l_j(A, B) = g(a_{2j+1}, a_{2j+2})$.

Exemplul 19.7 *Să reluăm* $(8, 3)$ *- codul balansat din Exemplul 19.4. După ce se construieşte reţ eaua* $(8, 2)$ *- codului cadru, deoarece există doar un bit de informaţ ie suplimentar avem* $p = 1$ *deci reţ eaua se mai copiază de* $2^p - 1 = 1$ *ori.*

Sunt două maniere de abordare:

A. Dacă transmiterea se face pe linii, apare o construcţ ie conformă cu cea a reţ elelor definite pentru codurile GAC. Reţ eaua corespunzătoare codului cadru este prezentată în (a)*, iar cea pentru codul balansat, în* (b)*:*

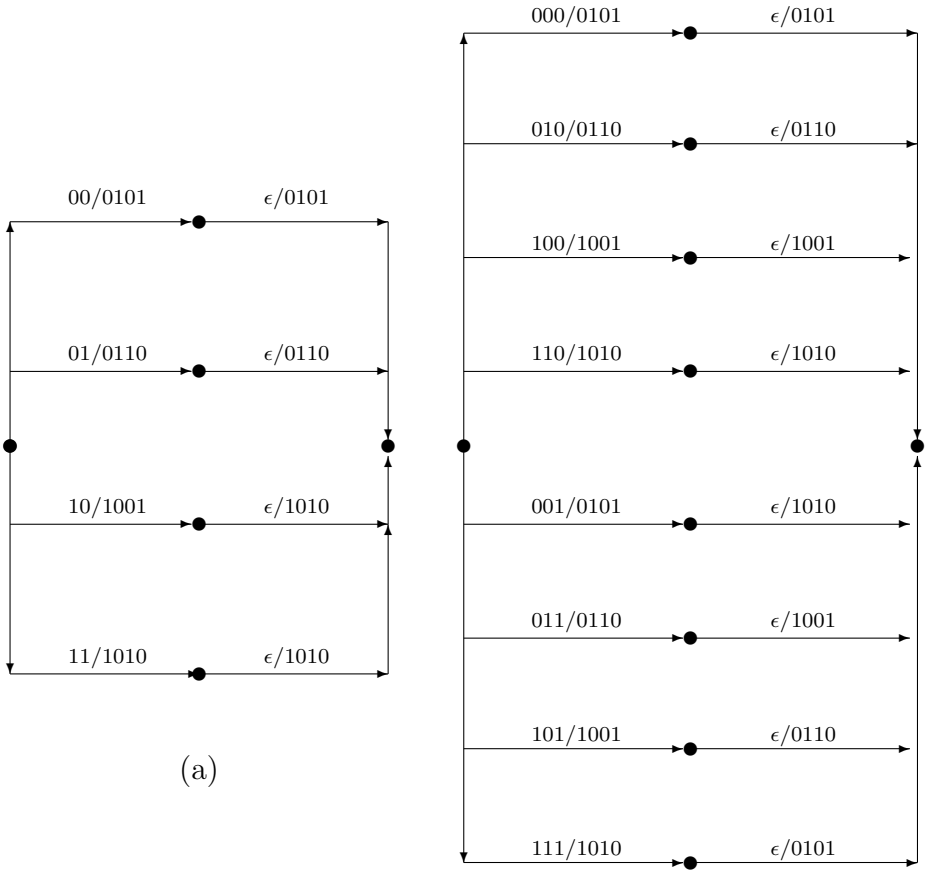


(a) *(b)*

B. Dacă transmiterea datelor se face pe coloane (specific codurilor balansate), reţ eaua codului cadru este cea prezentată în Figura 19.1, poziţ ia (a)*; din ea se obţ ine reţ eaua* (b) *a codului balansat.*

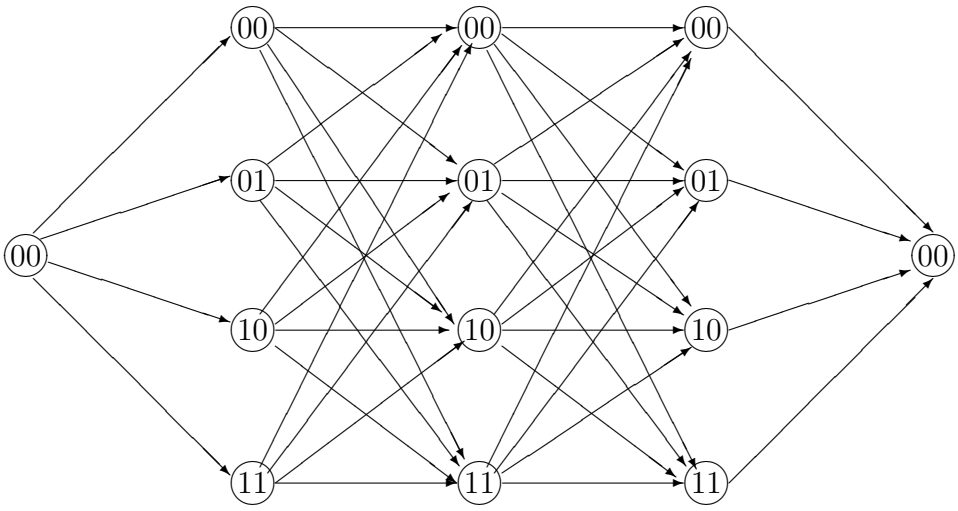
Exemplul 19.8 *Să construim reţ eaua de (de)codificare pentru* $(16, 9)$ *- codul balansat definit în Exemplul 19.6. Conform procedurii, sunt necesare* 8 *subreţ ele, câte una pentru*

Figura 19.1:



fiecare linie din tabelul de codificare. Deoarece cuvintele - cod sunt citite pe coloane, fiecare subreţ ea are câte 5 *coloane şi* 4 *stări.*

Pentru $a_7 = a_8 = a_9 = 0$ *(prima linie din tabel) subreţ eaua construită plecînd de la* $(16, 6)$ *- codul cadru din Exemplul 19.6 are structura:*



unde marcajul arcelor este dat de tabela:

p	δ_p/l_p	00	01	10	11
0	00	00/0101	01/0110	10/1001	11/1010
1	00	00/0101	01/0110	10/1001	11/1010
	01	01/0110	00/0101	11/1010	10/1001
	10	10/1001	11/1010	00/0101	01/0110
	11	11/1010	10/1001	00/0101	00/0101
2	00	00/0101	01/0110	10/1001	11/1010
	01	01/0110	00/0101	11/1010	10/1001
	10	10/1001	11/1010	00/0101	01/0110
	11	11/1010	10/1001	01/0110	00/0101
3	00	ϵ/0101	–	–	–
	01	ϵ/0110	–	–	–
	10	ϵ/1001	–	–	–
	11	ϵ/1010	–	–	–

Pentru $a_7 = a_8 = 0,\ a_9 = 1$ *(a doua linie din tabel) se construieşte aceeaşi subreţ ea, în care valorile funcţ iei* l_3 *din marcajul arcelor ultimului nivel se complementează. Prin compunerea acestor două subreţ ele se obţ ine* $(16, 7)$ *- codul balansat cu* $d = 4$.

Celelalte 6 *subreţ ele se definesc similar. Reţ eaua finală a* $(16, 9)$ *- codului balansat cu* $d = 4$ *obţ inută din compunerea lor va avea* 32 *stări şi* 5 *nivele.*

19.3 Exerciţ ii

19.1 *Construiţ i un* $(3,2)(3,2)$ *– RLL cod tablou şi calculaţ i parametrii săi* $(d^0,\ k^0)$.

19.2 *Construiţ i un* $(12,8)$ *- cod GAC cu restricţ ii RLL şi calculaţ i pentru el* $(d^0,\ k^0)$.

19.3 *Construiţ i un GAC cod balansat pentru* $k = 10$.

19.4 *Aceeaşi problemă pentru* $k = 13$.

19.5 *Definiţ i un vector de modificare care transformă codul tablou* $(3,2)(3,2)$ *într-un cod RLL. Construiţ i reţ eaua acesui cod.*

19.6 *Aceeaşi problemă pentru* $(12,8)$ *- codul GAC cu* $d = 3$.

19.7 *Construiţ i un* $(12,5)$ *cod balansat cu* $d = 4$. *Trasaţ i reţ eaua acestui cod.*

Capitolul 20

Alte rezultate din teoria codurilor

20.1 Coduri aritmetice

Construcţiile oferite de teoria codurilor pot fi utilizate şi în alte domenii decât în cele clasice, de transmitere şi recepţie corectă a mesajelor. O aplicaţie legată de operaţiile aritmetice pe calculator este prezentată de van Lindt în [11].

Definiţia 20.1 *Fie $r \geq 2$ un număr întreg fixat. r - ponderea aritmetică unui număr întreg x este definită prin*

$$w_{(r)}(x) = \begin{cases} 0 & \text{dacă } x = 0 \\ t & \text{altfel} \end{cases}$$

$$\text{unde } t = min \left\{ p \left| \sum_{i=1}^{p} a_i \cdot r^{n(i)} = x,\ |a_i| < r,\ n(i) \geq 0 \right. \right\}.$$

În practică se folosesc des cazurile $r = 2, 8, 10, 16$.

Exemplul 20.1
$w_{(10)}(5) = 1$,
$w_{(10)}(199) = 2$ *(deoarece* $199 = 200 - 1 = 2 \cdot 10^2 - 1 \cdot 10^0$),
$w_{(2)}(-9) = 2$,
$w_{(2)}(246) = 3$ *(deoarece* $246 = 256 - 8 - 2 = 2^8 - 2^3 - 2^1$).

Propoziţia 20.1
1. $w_{(r)}(-x) = w_{(r)}(x)$;
2. $w_{(r)}(x + y) \leq w_{(r)}(x) + w_{(r)}(y)$.

Demonstraţie: Este lăsată ca exerciţiu. □

În cele ce urmează vom considera r ca o valoare fixată. Se numeşte *distantă aritmetică* valoarea

$$d(x, y) = w_{(r)}(x - y).$$

Propoziţia 20.2
1. *d este o distantă;*
2. *d este invariantă la translatare.*
3. *$d(x, y) \leq d_H(x, y)$ unde d_H este distanţa Hamming a două secvenţe numerice reprezentate în baza r.*

Demonstraţie: (1) Faptul că d este o distantă se verifică imediat din Definiţia 20.1 şi Propoziţia 20.1.

(2) Deoarece $d(x+z, y+z) = w_{(r)}(x+z-y-z) = w_{(r)}(x-y) = d(x,y)$ rezultă că distanţa aritmetică este invariantă la translatare (proprietate pe care distanţa Hamming nu o are).

(3) Este lăsat ca exerciţiu. Precizăm că cele două numere reprezentate în baza r pot fi aduse la un număr egal de cifre completând eventual numărul mai scurt cu 0-uri (nesemnificative) în fată. □

Definiţia 20.2 *Fie a, b numere întregi pozitive. Se numeşte AN - cod mulţimea finită $C_{a,b} = \{a \cdot n | 0 \leq n < b\}$.*

Ideea folosirii AN - codurilor în aritmetica calculatorului este următoa-rea: să presupunem că trebuie calculată suma $n_1 + n_2$ (n_1, n_2 pozitive şi mici comparativ cu b); fie S suma lor. Cele două numere se codifică în $a \cdot n_1, a \cdot n_2 \in C_{a,b}$. Dacă $S \equiv 0 \ (mod\ a)$ atunci suma a fost efectuată corect şi ea este $S\ div\ a$. Dacă nu, înseamnă că au apărut erori de calcul şi se ia ca rezultat acel număr n_3 cu proprietatea că $d(S, an_3)$ este minimă.

Pentru a corecta orice combinaţie de maxim t erori este necesar ca C să aibă distanţa minimă $\geq 2t+1$, deci orice număr din C are ponderea minim $2t+1$.

Teorema 20.1 *Fie a un număr întreg fixat şi $s = min\{w_{(r)}(a \cdot n) | n \neq 0\}$. Atunci $s \leq 2$.*

Demonstraţie: Cazurile $a = 0$ şi $a = 1$ sunt banale. Pentru $a < 0$ se va lucra cu $-a$. Deci rămâne de studiat numai cazul $a > 1$. Vom folosi Teorema lui Fermat:

$$\text{Dacă } (a, r) = 1 \text{ atunci } r^{\phi(a)} \equiv 1 \ (mod\ a).$$

(reamintim, $\phi(a)$ este simbolul Euler).

Deci $a \mid r^{\phi(a)} - 1$. Cum $w_{(r)}(r^{\phi(a)} - 1) = 2$, afirmaţia este demonstrată pentru cazul când a şi r sunt prime între ele.

Să presupunem acum că $(a, r) = d > 1$. Vom avea $a = a_1 d,\ r = r_1 d$ cu $(a_1, r_1) = 1$. Atunci (notând $n = \phi(a_1)$), putem scrie $r^n - 1 = d^n r_1^n - 1 = d^n(r_1^n - 1) + (d^n - 1)$. Se ştie că $a_1 \mid r_1^n - 1$. Dacă $(a_1, d) = 1$ atunci $a_1 \mid d^n - 1$, deci $a_1 \mid r^n - 1$, sau $a \mid d \cdot r^n - d$ şi cum $d < r$, teorema este demonstrată.

Dacă $(a_1, d) > 1$, raţionamentul se reia şi el se va termina după un număr finit de paşi (deoarece $a_1 < a,\ d < r$). □

20.1.1 AN - coduri ciclice

Rezultatul menţionat în Teorema 20.1 conduce la o dificultate de alegere. O aritmetică a calculatorului eficientă recomandă alegerea unei valori mari pentru b. Pe de-altă parte Teorema 20.1 arată că în acest caz riscul de a avea r - ponderea aritmetică minimă cel mult 2 (deci de a nu putea corecta nici o eroare) este mare.

Problema se elimină dacă vom considera codurile AN *modulare*. Fie a, b numere prime şi $m = a \cdot b$. Definim $C_{a,b}$ ca subgrup al lui Z_m. Aceasta conduce la o altă definiţie pentru distanţa dintre două numere. Pentru determinarea ei, să luăm elementele lui Z_m ca vârfuri într-un graf Γ_m; $a, b \in Z_m$ sunt legate printr-un arc dacă şi numai dacă

$$\exists\ c, j\ (0 < c < r,\ j \geq 0) \quad a - b \equiv \pm c \cdot r^j \ (mod\ m).$$

Definiţia 20.3 *Distanţa modulară $d_m(x, y)$ dintre două numere $a, b \in Z_m$ este lungimea drumului minim dintre a şi b în graful Γ_m.*

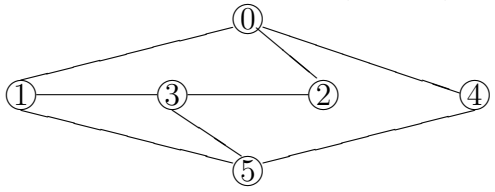
Ponderea modularăindexpondere modulară a lui $a \in Z_m$ este $w_m(a) = d_m(a, 0)$.

Propoziţia 20.3 *Pentru $a \in Z_m$*

$$w_m(a) = min\{w_{(r)}(x) |\ x \in \mathcal{Z}, y \equiv x\ (mod\ m)\}.$$

Demonstraţie: Rezultă imediat din Definiţia 20.3. □

Exemplul 20.2 *Să considerăm $r = 2,\ a = 2,\ b = 3$ deci $m = 6$. Graful Γ_6 va fi*



deci tabelul distanţelor între elementele lui Z_6 este:

d	0	1	2	3	4	5
0	0	1	1	2	1	2
1	1	0	2	1	2	1
2	1	2	0	1	2	2
3	2	1	1	0	2	1
4	1	2	2	2	0	1
5	2	1	2	1	1	0

De remarcat că alegerea lui m trebuie făcută cu grijă. De exemplu, dacă se ia $r = 3,\ m = 35$ vom avea $d_{35}(0, 12) = 1$ deoarece $12 = 3^{11}\ (mod\ 35)$. Practic însă, lucrând cu numere din Z_{35}, nu vom putea corecta erori pe poziţia corespunzătoare lui 3^{11}.

De aceea, aritmetica pe calculator consideră doar situaţia $m = r^n - 1\ (n \geq 2)$ care elimină astfel de situaţii. Orice număr întreg nenul x admite o reprezentare unică

$$x \equiv \sum_{i=0}^{n-1} c_i \cdot r^i \quad mod\ (r^n - 1),$$

cu $0 \leq c_i < r$ nu toţi nuli.

Deci Z_{r^n-1} poate fi interpretat ca fiind mulţimea $GF(r^n) \setminus \{0\}$ cuvintelor nenule de lungime n peste alfabetul $\{0, 1, \ldots, r-1\}$.

Pentru $a \cdot b = m = r^n - 1$ distanţa modulară devine distanţa obişnuită în Z_m şi $C_{a,b}$ are un comportament similar unui cod liniar.

Definiţia 20.4 *Un AN - cod ciclic de lungime n şi bază r este un subgrup multiplicativ $C \subseteq Z_{r^n-1}$.*

Evident, un astfel de subgrup este ideal principal în inelul Z_{r^n-1}, deci există $a, b \in \mathcal{Z}$ astfel încât $a \cdot b = r^n - 1$ şi $C = C_{a,b}$, adică:

$$C = \{a \cdot k | k \in \mathcal{Z},\ 0 \leq k < b\}.$$

Dacă $x \in C_{a,b}$, atunci $r \cdot x\ (mod\ r^n - 1)$ este tot în $C_{a,b}$ deoarece acesta este grup multiplicativ; pe de-altă parte noul număr este o permutare ciclică a lui x (ambele fiind scrise în baza r). Numărul b poate fi asimilat polinomului de control al unui cod ciclic.

Exemplul 20.3 *Fie* $r = 2,\ n = 11$. *Atunci* $m = 2^{11} - 1 = 2047$. *Să considerăm* $a = 23,\ b = 89$. *Se obţine AN - codul ciclic format din* 89 *multipli ai lui* 23 *(până la* 2047*). Există* 22 *modalităţi de a semnala o eroare, fiecare din ele corespunzând unui număr de forma* $\pm 2^j\ (0 \leq j \leq 11)$. *Acestea sunt exact toate numerele din* $Z_{23} \setminus \{0\}$. *Deci orice număr întreg din intervalul* $[0, 2047]$ *are distanţa modulară* 0 *sau* 1 *de exact un cuvânt - cod. Acest AN - cod ciclic este perfect şi poate fi considerat o generalizare a codurilor Hamming.*

În [11] se face afirmaţia că nu există AN - coduri perfecte corectoare de o eroare pentru $r = 10$ *sau* $r = 2^k,\ (k > 1)$.

20.1.2 *AN* - coduri corectoare de mai multe erori

Pentru a construi *AN* - coduri capabile să corecteze erori multiple, va trebui definită o modalitate mai simplă de determinare a ponderii aritmetice sau modulare a numerelor întregi.

Conform Definiţiei 20.1, orice număr întreg x se poate scrie sub forma

$$x = \sum_{i=1}^{w_{(r)}} a_i \cdot r^{n(i)}$$

cu $a_i,\ n(i)$ numere întregi, $|a_i| < r,\ n(i) \geq 0\ (0 \leq i \leq w_{(r)})$. Această reprezentare are defectul că nu este unică. De exemplu, pentru $r = 10$, numărul 99 se poate reprezenta în două moduri diferite:

$$99 = 9 \cdot 10 + 9 \cdot 10^0, \qquad 99 = 1 \cdot 10^2 - 1 \cdot 10^0.$$

Se poate obţine – prin impunerea de restricţii asupra coeficienţilor – o reprezenta-re unică a numerelor întregi.

Definiţia 20.5 *Fie* $b, c \in \mathcal{Z}\ ,\ |b| < r,\ |c| < r$. *Perechea* (b, c) *se numeşte admisibilă dacă este adevărată una din relaţiile:*

(1) $b \cdot c = 0$*;*
(2) $b \cdot c > 0$ *şi* $|b + c| < r$*;*
(3) $b \cdot c < 0$ *şi* $|b| > |c|$.

De remarcat că ambele perechi $(b, c),\ (c, b)$ sunt admisibile numai în cazurile (1) sau (2). Cazul (3) nu permite comutativitatea relaţiei de admisibilitate.

Exemplul 20.4 *Pentru* $r = 2$ *este posibil numai cazul* (1). *Deci o reprezentare* $x = \sum_{i=0}^{\infty} c_i \cdot 2^i$ *în care toate perechile* (c_{i+1}, c_i) *sunt admisibile, nu are doi coeficienţi consecutivi nenuli.*

Definiţia 20.6 *O reprezentare* $x = \sum_{i=0}^{\infty} c_i r^i$ *cu* $c_i \in \mathcal{Z}$, $|c_i| < r$ *şi* $\exists n_x$ *cu* $c_i = 0\ \forall i > n_x$ *se numeşte NAF (non-adiacent form) dacă pentru orice* $i \geq 0$, *perechea* (c_{i+1}, c_i) *este admisibilă.*

Exemplul 20.5 *Pentru* $r = 10$ *putem scrie:*
$96 = -4 \cdot 10^0 + 0 \cdot 10^1 + 1 \cdot 10^2$ *(cazul* (1)*),*

$11 = 1 \cdot 10^0 + 1 \cdot 10^1$ *(cazul* (2)*),*
$38 = -2 \cdot 10^0 + 4 \cdot 10^1$ *(cazul* (3)*).*

De remarcat că reprezentarea lui $96 = 6 \cdot 10^0 + 9 \cdot 10^1$ *nu este în forma* NAF *deoarece perechea* $(9, 6)$ *nu este admisibilă. La fel pentru reprezentările celorlaltor numere.*

Teorema 20.2 *Orice număr întreg* x *are o reprezentare* NAF *unică în baza* r*. Dacă aceasta este*

$$x = \sum_{i=0}^{\infty} c_i \cdot r^i,$$

atunci $w_{(r)}(x) = card\,(\{i | i \geq 0,\ c_i \neq 0\})$.

Demonstraţie: Fie $\sum_{i=0}^{\infty} b_i \cdot r^i,\ (|b_i| < r)$ o reprezentare a lui x în baza r, şi i cel mai mic număr cu proprietatea că perechea (b_{i+1}, b_i) nu este admisibilă. Putem presupune că $b_i > 0$ (altfel se va lucra cu $-x$). Vom înlocui b_i cu $b'_i = b_i - r$ şi b_{i+1} cu $b'_{i+1} = b_{i+1} + 1$ (dacă $b_{i+1} + 1 = r$, atunci $b'_{i+1} = 0$ şi facem deplasarea obişnuită de la adunare).

Dacă $b_{i+1} > 0$, atunci avem sau $b'_{i+1} = 0$, sau $b'_i \cdot b'_{i+1} < 0$ şi $b'_{i+1} = b_{i+1} + 1 > r - b_i = |b'_i|$ (deoarece perechea (b_{i+1}, b_i) nu era admisibilă).

Dacă $b_{i+1} < 0$, atunci sau $b'_{i+1} = 0$, sau $b'_i \cdot b'_{i+1} > 0$ şi $|b'_i + b'_{i+}| = r - b_i - b_{i+1} < r$ deoarece $-b_{i+1} \leq b_i$ (perechea (b_{i+1}, b_i) nu era admisibilă).

Deci, (b'_{i+1}, b'_i) este admisibilă, şi se verifică similar dacă (b'_i, b_{i-1}) este admisibilă. Procedeul continuă până se ajunge la $i = 0$.

Să arătăm acum că reprezentarea NAF este unică. Presupunem că există $x \in \mathcal{Z}$ cu două astfel de reprezentări:

$$x = \sum_{i=0}^{\infty} c_i \cdot r^i = \sum_{i=0}^{\infty} c'_i \cdot r^i.$$

Considerăm – fără a micşora generalitatea – că $c_0 \neq c'_0$ şi $c_0 > 0$; deci $c'_0 = c_0 - r$. Atunci pentru c'_1 sunt posibile trei valori: $c_1 + 1,\ c_1 + 1 \pm r$. Dacă $c'_1 = c_1 + 1 - r$, atunci $c_1 \geq 0$, deci $c_0 + c_1 \leq r - 1$. Deoarece $c'_0 \cdot c'_1 > 0$, avem $-c'_0 - c'_1 < r$, de unde $r - c_0 + r - c_1 - 1 < r$, deci $c_0 + c_1 > r - 1$, contradicţie. Celelalte două cazuri se tratează similar. □

Reprezentarea NAF a unui număr x se poate afla direct şi pe baza teoremei:

Teorema 20.3 *Fie* $x \in \mathcal{Z}, x \geq 0$*. Considerăm reprezentările în baza* r *a numerelor*

$$(r+1) \cdot x = \sum_{i=0}^{\infty} a_i \cdot r^i, \quad x = \sum_{i=0}^{\infty} b_i \cdot r^i,$$

unde $a_i, b_i \in \{0, 1, \ldots, r-1\}$*. Atunci reprezentarea* NAF *pentru* x *este*

$$x = \sum_{i=0}^{\infty} (a_{i+1} - b_{i+1}) \cdot r^i.$$

Demonstraţie: Ştim că pentru două numere naturale nenule a, r, $[a/r]$ reprezintă câtul împărţirii celor două numere, iar $a - [a/r] \cdot r$ - restul.

Din reprezentarea din enun,¬rezultă că fiecare coeficient a_i se determină prin adunarea coeficienţilor corespunzători ai numerelor x şi $r \cdot x$, scrise în baza r. Să definim secvenţa numerică $\alpha_i,\ i \geq 0$ astfel:

$$\alpha_0 = 0, \quad \alpha_i = \left[\frac{\alpha_{i-1} + b_{i-1} + b_i}{r}\right].$$

Atunci, conform observaţiei de la începutul demonstraţiei, $a_i = \alpha_{i-1} + b_{i-1} + b_i - \alpha_i \cdot r$. Dacă notăm $c_i := a_i - b_i$, avem $c_i = \alpha_{i-1} + b_{i-1} - \alpha_i \cdot r$.

Mai rămâne de verificat faptul că (c_{i+1}, c_i) este o pereche admisibilă. Relaţia $|c_{i+1} + c_i| < r$ rezultă imediat din definiţia lui α_i. Să presupunem $c_i > 0,\ c_{i+1} < 0$; atunci $\alpha_i = 0$. Vom avea $c_i = \alpha_{i-1} + b_{i-1},\ c_{i+1} = b_i - r$ şi condiţia $|c_{i+1}| > |c_i|$ este echivalentă cu $\alpha_{i-1} + b_{i-1} + b_i < r$, adică $\alpha_i = 0$. Celălalt caz se arată analog. □

Exemplul 20.6 *Pentru a găsi reprezentarea NAF a numărului 98 în baza $r = 10$, avem*

$$\begin{aligned} 98 &= 8 \cdot 10^0 + 9 \cdot 10^1 + 0 \cdot 10^2 + 0 \cdot 10^3 + \ldots \\ 980 + 98 = 1078 &= 8 \cdot 10^0 + 7 \cdot 10^1 + 0 \cdot 10^2 + 1 \cdot 10^3 \end{aligned}$$

Deci, $98 = (7-9) \cdot 10^0 + (0-0) \cdot 10^1 + (1-0) \cdot 10^2 = -2 + 1 \cdot 10^2$.

Similar situaţiei din paragraful anterior, să considerăm acum cazul reprezentării modulare. Vom lua deci $m = r^n - 1,\ (n \geq 2)$.

Definiţia 20.7 *O reprezentare*

$$x \equiv \sum_{i=0}^{n-1} c_i \cdot r^i \ (mod\ m)$$

cu $c_i \in \mathcal{Z}, |c_i| < r$ se numeşte $CNAF$ (cyclic NAF) pentru x dacă $\forall i\ (0 \leq i \leq n-1)$, (c_{i+1}, c_i) este admisibilă (se consideră $c_n = c_0$).

Din Teoremele 20.2 şi 20.3 rezultă un rezultat similar pentru reprezentă- rile $CNAF$:

Teorema 20.4 *Orice număr întreg x admite o reprezentare $CNAF$ modulo m. Această reprezentare este unică, exceptând cazul*

$$(r+1) \cdot x \equiv 0 \not\equiv x \ (mod\ m),$$

când sunt posibile două reprezentări.

Dacă $x \equiv \sum_{i=0}^{n-1} c_i \cdot r^i \ (mod\ m)$, *atunci*

$$w_m(x) = card(\{i | 0 \leq i < n,\ c_i \neq 0\}).$$

Demonstraţie: Construcţia este identică cu cea din demonstraţia Teoremei 20.3. Unicitatea se arată similar cu cea din demonstraţia Teoremei 20.2. Singura excepţie este cazul $b_{i+1} \equiv b_i\ (mod\ m)$. În acest caz sunt posibile două reprezentări:

$x \equiv b_0 + b_1 \cdot r + \ldots b_{n-1} \cdot r^{n-1}$ şi $x \equiv -b_1 - b_2 \cdot r - \ldots - b_0 \cdot r^{n-1}$,

ambele modulo m. □

20.1.3 Coduri Mandelbaum - Barrows

O clasă de AN coduri a fost definită de Mandelbaum şi Barrows, generalizată ulterior de van Lindt ([11]).

Iniţial este necesar un rezultat referitor la ponderea modulară în AN coduri ciclice, a cărui demonstrare se află în [11], pag. $127 - 128$:

Teorema 20.5 *Fie $C \subset \mathcal{Z}/(r^n-1)$ un AN cod ciclic cu generator a şi $b = (r^n-1)/a = card(C)$, cu proprietatea că $\forall x \in C$ are o reprezentare $CNAF$ unică. Atunci*

$$\sum_{x \in C} w_m(x) = n\left(\left[\frac{r \cdot b}{r+1}\right] - \left[\frac{b}{r+1}\right]\right).$$

Teorema 20.6 *Fie b un număr prim care nu divide r, cu proprietatea că grupul multiplicativ Z_b este generat de r şi -1. Fie n un număr întreg pozitiv astfel ca $r^n \equiv 1 \ (mod\ b)$ şi $a = (r^n-1)/b$. Atunci codul $C \subset \mathcal{Z}/(r^n-1)$ generat de a este un cod echidistant cu distanţa*

$$\frac{n}{b-1}\left(\left[\frac{r \cdot b}{r+1}\right] - \left[\frac{b}{r+1}\right]\right).$$

Demonstraţie: Fie $x \in C \setminus \{0\}$. Atunci $x = a \cdot n \ (mod\ r^n-1)$ cu $n \neq 0 \ (mod\ b)$. Din ipoteză rezultă că există j întreg cu $n \pm r^j \ (mod\ b)$. Deci $w_m(x) = w_m(\pm r^j \cdot a) = w_m(a)$. Aceasta arată că C este echidistant. Valoarea distanţei rezultă din Teorema 20.5. □

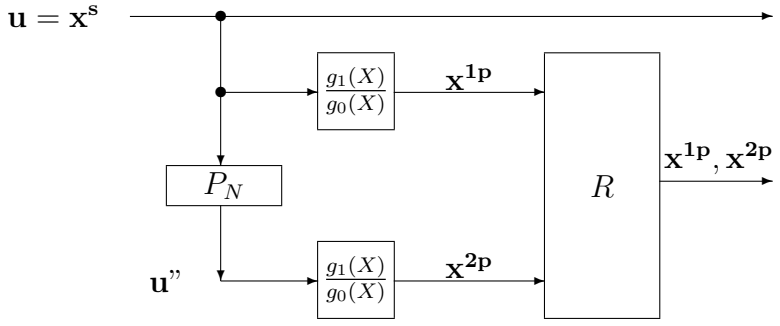
20.2 Turbo - coduri

În ultimii ani, viteza tot mai mare de transmisie a datelor - aproape de capacitatea maximă a canalelor de comunicaţie – a condus la modalităţi noi de codificare. Turbo codurile sunt prezentate prima dată în 1993 de Berrou, Glavier şi Thitimajshima şi combină sub formă de reţea (minim) două coduri convoluţionale. Modalitatea de decodificare este total deosebită de algoritmii cunoscuţi până acum (se folosesc capacităţile statistice de performantă ale canalelor de transmisie). Ele asigură o rată de corectare a erorilor mult mai ridicată decât la codurile clasice; de aceea turbo - codurile asigură transmisiile de pe staţiile lansate după anul 1993, precum şi canalele de satelit.

20.2.1 Structura unui turbo - cod

Un turbo - cod standard este reprezentat de Figura 20.1:

Figura 20.1: Turbo - codificator standard



El foloseşte două $(2,2)$ - coduri convoluţionale (care – fără a micşora generalitatea – au fost considerate identice), separate printr-un bloc P_N de permutare de N caractere (N fiind o constantă fixată, dependentă de structura canalului de transmisie). Mesajul

de informaţie este spart în blocuri $u \in Z_q$ de lungime N; dacă se ignoră mecanismul de relaxare R, rata codificatorului este $1/3$ (N simboluri de informaţie se transformă în cuvinte - cod de lungime $3N$). Cele 3 ieşiri $(\mathbf{x^s}, \mathbf{x^{1p}}, \mathbf{x^{2p}})$ sunt apoi transmise pe coloană, ca un cod GAC.

Codificatorul

Pentru codul convoluţional, matricea generatoare poate fi considerată (într-o variantă simplificată, bazată pe structura de reţea a mesajelor) $G = (g_0(X)\ g_1(X))$. Codificatorul unui turbo - cod va folosi ca matrice generatoare o formă echivalentă recursivă:

$$G_R^T = \begin{pmatrix} 1 & \dfrac{g_1(X)}{g_0(X)} \end{pmatrix}.$$

Din acest motiv, un codificator convoluţional pentru turbo - coduri este numit *Codificator Sistematic Recursiv* (RSE).

Un mesaj de informaţie $u(X)$, este codificat de codul convoluţional în $u(X)G = (u(X)g_0(X)\ u(X)g_1((X))$. RSE va realiza aceeaşi ieşire pentru mesajul $u'(X) = u(X)g_0(X)$ (se verifică imediat relaţia $u(X)g_0(X)G_R = u(X)G$). Vom numi totuşi cuvânt - cod perechea de polinoame $u(X)G$ (deşi se mai efectuează o operaţie de înmulţire pentru obţinerea mesajului $u'(X)$).

Se observă că pentru un RSE, cuvântul cod are pondere finită dacă şi numai dacă mesajul de intrare se divide cu $g_0(X)$.

Corolarul 20.1 *Un mesaj sursă* $\mathbf{u}$" *cu* $w(\mathbf{u}") = 1$ *se codifică într-un cuvânt - cod de pondere infinită.*

Demonstraţie: Evident, deoarece $u'(X) = X^p$ nu se divide cu $g_0(X)$. □

Corolarul 20.2 *Pentru orice polinom netrivial* $g_0(X) \in Z_q[X]$ *există o infinitate de mesaje sursă de pondere* 2 *care se codifică în cuvinte - cod de pondere finită.*

Demonstraţie: Pentru $g_0(X) \in Z_q[X]$, $g_0(X) \neq X^p$, există un n minim cu proprietatea $g_0(X)|X^n - 1$ (n este lungimea secvenţei pseudo-aleatoare generată de $g_0(X)$ - a se vedea Relaţii de recurentă liniară, Prelegerea 8).

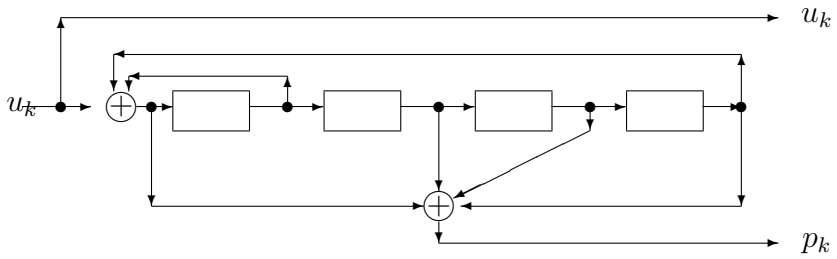
Orice secvență $\mathbf{u}$" de forma $u"(X) = aX^i(X^n - 1)$, $a \in Z_q \setminus \{0\}$ are pondere 2 şi este divizibilă cu $g_0(X)$, deci codificarea prin RSE va genera un polinom cu un număr finit de termeni. □

Exemplul 20.7 *Fie* $g_0(X) = 1 + X + X^4$, $g_1(X) = 1 + X^2 + X^3 + X^4$ *polinoame din* $Z_2[X]$. *În mod uzual se foloseşte notaţia în octal; deci, cum* $g_0 = 11001_2 = 31_8$, $g_1 = 10111_2 = 27_8$, *vom avea* $(g_0\ g_1) = (31, 27)$.

Matricea generatoare este

$$G_R = \begin{pmatrix} 1 & \dfrac{1 + X^2 + X^3 + X^4}{1 + X + X^4} \end{pmatrix},$$

iar un circuit liniar care realizează acest RSE are forma:



(s-a notat cu u_k simbolul de informaţie curent, iar cu p_k simbolul de control corespunzător).

Cum $g_0(X)$ este primitiv, lungimea secvenţelor este $2^4 - 1 = 15$. De exemplu, mesajul sursă $u(X) = 1 + X^{15}$ se codifică în $(1 + X^{15}, 1 + X + X^2 + X^3 + X^5 + X^7 + X^8 + X^{11}) =$ $(1000000000000001, 1111010110010000)$.
$u(X) = X^7(1 + X^{15})$ va genera acelaşi cuvânt - cod, cu o întârziere de şapte tacţi.

Permutatorul

P_N este un bloc de permutare. Cele N caractere care constituie intrarea în primul codificator RSE sunt rearanjate înainte de a intra în al doilea codificator. Din considerente practice este preferabil ca permutarea folosită să nu păstreze nici o ordine anterioară a simbolurilor (deşi acest lucru este uneori dificil, mai ales pentru cuvinte de pondere mică). De asemenea, N trebuie să fie suficient de mare (în practică se foloseşte $N \geq 1000$). Aceste două cerinţe – uzuale în criptografie – sunt necesare în obţinerea de performanţe ridicate la decodificare.

Mecanismul de relaxare

Dacă pentru transmiterea de imagini din spaţiu sunt folosite coduri cu rate mici de informaţie (fiecărui bit îi corespund cel puţin 3 caractere cod), în alte situaţii (comunicări prin satelit de exemplu) sunt preferabile rate mari (cel puţin 1/2). Rolul mecanismului de relaxare (vezi Capitolul 2) este de a reduce periodic anumite caractere pentru a scurta lungimea cuvintelor - cod. De obicei se elimină biţi de control; astfel, pentru a obţine o rată de informaţie 1/2 se pot elimina toţi biţii de control pari de la începutul codului şi toţi biţii de control impari de la sfârşit.

20.2.2 Decodificarea turbo - codurilor

Din construcţie rezultă că un turbo codificator este liniar, deoarece toate componentele sale sunt liniare. Codificările RSE sunt implementate prin circuite liniare, permutatorul este liniar deoarece poate fi modelat printr-o matrice de permutare. La fel, mecanismul de relaxare nu afectează liniaritatea, deoarece din toate cuvintele - cod se şterg simbolurile de pe aceleaşi poziţii. Importanţa liniarităţii constă în faptul că se poate lua ca referintă cuvântul - cod nul. De asemenea, toate construcţiile le facem pentru cazul binar, cu simbolurile ± 1. Decodificatorul va lucra după principiul obişnuit al decodificării cele mai probabile.

Din păcate, utilizarea algoritmului Viterbi nu este posibilă din cauza operaţiei de permutare folosită în decodificare. Totuşi, pentru subsecven-ţe stricte, un astfel de algoritm poate da rezultate.

Primul algoritm de decodificare pentru turbo - coduri a fost propus de Berrou în 1993 ([12]), bazat pe ideile din [2]. Numit $BCJR$, el foloseşte o decodificare caracter-cu-caracter

(spre deosebire de Viterbi care decodifica pe secvenţe bloc de câte n caractere).

Vom folosi următoarele notaţii:

- E_i - notarea codificatorului RSE i $(1 \leq i \leq 2)$;
- D_i - notarea decodificatorului i $(1 \leq i \leq 2)$;
- m - capacitatea de memorie (buffer) a codificatorului;
- S - mulţimea celor 2^m stări ale codificatorului;
- $\mathbf{x^s} = x_1^s x_2^s \dots x_N^s = u_1 u_2 \dots u_N$ secvenţa de informaţie care se codifică;
- $\mathbf{x^p} = x_1^p x_2^p \dots x_N^p$ cuvântul de control generat de codificator;
- $y_k = y_k^s y_k^p$ o recepţie (posibil perturbată) a lui $x_k^s x_k^p$;
- $\mathbf{y_a^b} = y_a y_{a+1} \dots y_b$;
- $\mathbf{y_1^N} = y_1 y_2 \dots y_N$ cuvântul recepţionat.

Algoritmul $BCJR$ (iniţial şi modificat)

O primă versiune a algoritmului se bazează pe decodificarea cea mai probabilă *aposteriori* (MAP - maximul aposteriori). Se realizează decodificarea

$$u_k = \begin{cases} +1 & \text{dacă } P(u_k = +1|\mathbf{y}) > P(u_k = -1|\mathbf{y}) \\ -1 & \text{altfel.} \end{cases}$$

Formal, $\hat{u}_k = sign[L(u_k)]$, unde $L(u_k)$ este logaritmul raportului probabilităţilor de potrivire aposteriori, definit prin relaţia

$$L(u_k) = log\left(\frac{P(u_k = +1|\mathbf{y})}{P(u_k = -1|\mathbf{y})}\right).$$

Dacă se ţine cont de faptul că se codifică în reţea, aceasta se scrie:

$$L(u_k) = log\left(\frac{\sum\limits_{S^+} p(s_{k-1} = s', s_k = s, \mathbf{y})/p(\mathbf{y})}{\sum\limits_{S^-} p(s_{k-1} = s', s_k = s, \mathbf{y})/p(\mathbf{y})}\right) \tag{1}$$

unde

$s_k \in S$ este starea codificatorului la momentul k,

S^+ este mulţimea perechilor (ordonate) (s', s) corespunzătoare tuturor stărilor de tranziţie $(s_{k-1} = s') \to (s_k = s)$ generate de $u_k = +1$,

S^- este definită similar pentru $u_k = -1$.

Este posibil să simplificăm cu $p(\mathbf{y})$ în (1); deci este necesară doar o formulă pentru calculul lui $p(s', s, \mathbf{y})$. În [2], este construită o variantă sub forma

$$p(s", s, \mathbf{y}) = \alpha_{k-1}(s") \cdot \gamma_k(s", s) \cdot \beta_k(s) \tag{2}$$

unde:

- $\gamma_k(s", s) = p(s_k = s, y_k | s_{k-1} = s")$;

- $\alpha_k(s) = p(s_k = s, y_1^k)$ este calculat recursiv cu formula

$$\alpha_k(s) = \sum_{s" \in S} \alpha_{k-1}(s") \cdot \gamma_k(s", s)$$

cu condiţiile iniţiale $\alpha_0(0) = 1,\ \alpha_0(s \neq 0) = 0$

(codificatorul pleacă din starea 0).

- $\beta_k(s) = p(y_{k+1}^N | s_k = s)$ are formula recursivă de calcul

$$\beta_{k-1}(s") = \sum_{s \in S} \beta_k(s) \cdot \gamma_k(s", s)$$

şi condiţiile $\beta_N(0) = 1,\ \beta_N(s \neq 0) = 0$

(după N biţi de intrare codificatorul trebuie să ajungă la starea 0; restricţia se realizează alegând corespunzător ultimii m biţi, numiţi *biţi de încheiere*).

Aplicarea acestei variante de decodificare la turbo - coduri (reamintim, algoritmul $BCJR$ iniţial a fost definit în 1974) are un neajuns: simplificarea cu $p(\mathbf{y})$ conduce la algoritmi numerici instabili. De aceea, Berrou ([12]) face o modificare a algoritmului, în felul următor:

Se definesc probabilităţile (modificate)

$$\tilde{\alpha}_k(s) = \alpha_k(s)/p(y_1^k), \quad \tilde{\beta}_k(s) = \beta_k(s)/p(y_{k+1}^N | y_1^N).$$

Prin împărţirea relaţiei (2) cu $p(\mathbf{y})/p(y_k) = p(y_1^{k-1}) \cdot p(y_{k+1}^N | y_1^k)$ se ajunge la

$$p(s', s|\mathbf{y}) \cdot p(y_k) = \tilde{\alpha}_{k-1}(s') \cdot \gamma_k(s', s) \cdot \tilde{\beta}_k(s).$$

Pentru că $p(y_1^k) = \sum_{s \in S} \alpha_k(s)$, valorile $\tilde{\alpha}_k(s)$ se pot determina din $\alpha_k(s)$ pe baza formulei

$$\tilde{\alpha}_k(s) = \frac{\alpha_k(s)}{\sum_{s \in S} \alpha_k(s)},$$

sau – folosind definiţia recursivă a lui $\alpha_k(s)$:

$$\tilde{\alpha}_k(s) = \frac{\sum_{s' \in S} \alpha_{k-1}(s') \cdot \gamma_k(s', s)}{\sum_{s \in S} \sum_{s' \in S} \alpha_{k-1}(s') \cdot \gamma_k(s', s)} = \frac{\sum_{s' \in S} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k(s', s)}{\sum_{s \in S} \sum_{s' \in S} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k(s', s)}$$

ultimul rezultat fiind obţinut prin împărţirea numărătorului şi numitorului cu $p(y_1^{k-1})$.

Definirea recursivă a lui $\tilde{\beta}_k(s)$ se obţine plecând de la

$$p(y_k^N | y_1^{k-1}) = p(y_1^k) \cdot \frac{p(y_{k+1}^N | y_1^k)}{p(y_1^{k-1}} = \sum_{s \in S} \sum_{s' \in S} \alpha_{k-1}(s') \cdot \gamma_k(s', s) \cdot \frac{p(y_{k+1}^N | y_1^k)}{p(y_1^{k-1})} = \sum_{s \in S} \sum_{s' \in S} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k(s', s) \cdot p(y_{k+1}^N | y_1^k)$$

şi folosind definiţia recursivă a lui $\beta_{k-1}(s)$, se ajunge la relaţia

$$\tilde{\beta}_{k-1}(s') = \frac{\sum_{s \in S} \tilde{\beta}_k(s) \cdot \gamma_k(s', s)}{\sum_{s \in S} \sum_{s' \in S} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k(s', s)}.$$

În final, algoritmul $BCJR$ modificat va folosi valoarea $L(u_k)$ dată de relaţia

$$L(u_k) = log\left(\frac{\sum\limits_{S^+}\tilde{\alpha}_{k-1}(s')\cdot\gamma_k(s',s)\cdot\tilde{\beta}_k(s)}{\sum\limits_{S^-}\tilde{\alpha}_{k-1}(s')\cdot\gamma_k(s',s)\cdot\tilde{\beta}_k(s)}\right). \quad (3)$$

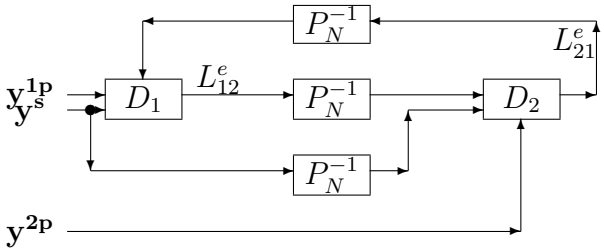
Condiţiile la limită pentru $\tilde{\alpha}_k(s)$ şi $\tilde{\beta}_k(s)$ sunt cele de la $\alpha_k(s)$ respectiv $\beta_k(s)$.

O altă versiune a algoritmului foloseşte informaţia *apriori*. Pentru acesta avem

$$L(u_k) = log\left(\frac{P(\mathbf{y}|u_k=+1)}{P(\mathbf{y}|u_k=-1)}\right) + log\left(\frac{P(u_k=+1)}{P(u_k=-1)}\right).$$

Deoarece în mod normal $P(u_k=+1)=P(u_k=-1)$, al doilea termen al sumei este zero în decodificatoarele uzuale. Pentru un turbo - decodor care lucrează recursiv, D_1 primeşte informaţie suplimentară de la D_2, care serveşte ca informaţie apriori. Similar, D_2 primeşte de la D_1 informaţie suplimentară, ş.a.m.d. Ideea constă în faptul că D_2 poate da lui D_1 informaţii despre u_k la care acesta nu are acces (de exemplu caracterele de control generate de E_2); acelaşi lucru îl realizează D_1 pentru D_2.

Un circuit de decodificare bazat pe algoritmul $BCJR$ este:



S-a notat cu P_N^{-1} inversa matricii de permutare P_N din circuitul de codificare. L_{12}^e este informaţia suplimentară transmisă de la D_1 la D_2, iar L_{21}^e este cea transmisă de la D_2 la D_1. Deciziile finale de decodificare pot veni atât de la D_1 cât şi de la D_2.

Mai rămâne de văzut cum se poate obţine această informaţie suplimentară care circulă între cei doi decodificatori recursivi.

Definiţia lui $\gamma_k(s',s)$ se poate rescrie

$$\gamma_k(s',s) = P(s|s')p(y_k|s',s) = P(u_k)\cdot p(y_k|u_k)$$

unde evenimentul u_k corespunde tranziţiei $s's$. Dacă se notează:

$$L^e(u_k) = log\left(\frac{P(u_k=+1)}{P(u_k=-1)}\right),$$

$P_+ = P(u_k=+1)$, $\quad P_- = P(u_k=-1)$, avem

$$\left(\frac{\sqrt{P_-/P_+}}{1+P_-/P_+}\right)\cdot\sqrt{P_+/P_-} = P_+ \text{ dacă } u_k=+1,$$

$$\left(\frac{\sqrt{P_-/P_+}}{1+P_-/P_+}\right)\cdot\sqrt{P_-/P_+} = P_- \text{ dacă } u_k=-1.$$

În această situaţie se obţine

$P(u_k) = \left(\frac{exp[-L^e(u_k)/2]}{1+exp[-L^e(u_k)]}\right)\cdot exp[u_k\cdot L^e(u_k)/2] = A_k\cdot exp[u_k\cdot L^e(u_k)/2]$, şi (reamintim, $y_k = y_k^s y_k^p$, $x_k = x_k^s x_k^p = u_k x_k^p$)

$$p(y_k|u_k) \propto exp\left[-\frac{(y_k^s-u_k)^2}{2\sigma^2} - \frac{(y_k^p-x_k^p)^2}{2\sigma^2}\right] =$$

$$= exp\left[-\frac{(y_k^s)^2 + u_k^2 + (y_k^p)^2 + (x_k^p)^2}{2\sigma^2}\right] \cdot exp\left[\frac{u_k \cdot y_k^s + x_k^p \cdot y_k^p}{\sigma^2}\right] =$$
$$= B_k \cdot exp\left[\frac{y_k^s \cdot u_k + y_k^p \cdot x_k^p}{\sigma^2}\right]$$

deci

$$\gamma_k(s', s) \propto A_k \cdot B_k \cdot exp\,[u_k \cdot L^e(u_k)/2] \cdot exp\left[\frac{u_k \cdot y_k^s + x_k^p \cdot y_k^p}{\sigma^2}\right] \tag{4}$$

Deoarece $\gamma_k(s', s)$ apare în (3) la numărător (unde $u_k = +1$) şi numitor (unde $u_k = -1$), factorul $A_k \cdot B_k$ se va reduce fiind independent de u_k. De asemenea, particularităţile de canal la transmisia lui ± 1 dau relaţia $\sigma^2 = N_0/(2E_c)$ unde E_c este energia de canal per bit. Din (4) se obţine

$$\gamma_k(s', s) \sim exp\,[u_k \cdot (L^e(u_k) + L_c \cdot y_k^s)/2 + L_c \cdot y_k^p \cdot x_k^p/2] =$$
$$= exp\,[u_k \cdot (L^e(u_k) + L_c \cdot y_k^s)/2] \cdot \gamma_k^e(s', s),$$

unde $L_c = \dfrac{4E_c}{N_0}$ şi $\gamma_k^e(s', s) = exp\,[L_c \cdot y_k^p \cdot x_k^p/2]$.

Combinând această relaţie cu (3) se ajunge la

$$L(u_k) = log\left(\frac{\sum\limits_{S^+} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k^e(s', s) \cdot \tilde{\beta}_k(s) \cdot C_k}{\sum\limits_{S^-} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k^e(s', s) \cdot \tilde{\beta}_k(s) \cdot C_k}\right) =$$
$$= L_c y_k^s + L^e(u_k) + log\left(\frac{\sum\limits_{S^+} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k^e(s', s) \cdot \tilde{\beta}_k(s)}{\sum\limits_{S^-} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k^e(s', s) \cdot \tilde{\beta}_k(s)}\right) \tag{5}$$

unde s-a notat $C_k = exp\,[u_k \cdot (L^e(u_k) + L_c \cdot y_k^s)/2]$.

A doua egalitate rezultă deoarece $C_k(u_k = +1)$ şi $C_k(u_k = -1)$ pot fi scoase ca factor. Primul termen al sumei (5) este numit *valoare de canal*, al doilea reprezintă informaţia apriori despre u_k (furnizată de un decodificator anterior), iar al treilea termen conţine informaţia suplimentară care se trimite la decodificatorul următor. Deci – de exemplu – la orice iteraţie, D_1 calculează

$$L_1(u_k) = L_c \cdot y_k^s + L_{21}^e(u_k) + L_{12}^e(u_k)$$

unde $L_{21}^e(u_k)$ este informaţia suplimentară venită de la D_2 şi $L_{12}^e(u_k)$ este al treilea termen din (5), folosit ca o informaţie suplimentară trecută de la D_1 spre D_2.

Algoritmul $BCJR$ se poate formaliza, în unele ipoteze implicite. Astfel:

- se presupune că cei doi codificatori lucrează corect, adică ultimii m biţi din mesajul de informaţie de lungime N se codifică astfel ca la sfârşit E_1 să ajungă la starea zero.

- decodificatorii deţin toată informaţia referitoare la reţeaua de codificare; astfel, ei au tabele complete cu simbolurile de informaţie şi de control pentru toate tranziţiile de stări $s's$, matricile de permutare şi inversele lor.

Algoritmul *BCJR*:
1. (Iniţializare):

$\mathbf{D_1}$: $\tilde{\alpha}_0^{(1)}(s) := \begin{cases} 1 & \text{pentru } s = 0 \\ 0 & \text{pentru } s \neq 0 \end{cases}$ $\tilde{\beta}_N^{(1)}(s) := \begin{cases} 1 & \text{pentru } s = 0 \\ 0 & \text{pentru } s \neq 0 \end{cases}$

$L_{21}^e(u_k) := 0,\ k = 1, 2, \ldots, N$

$\mathbf{D_2}$: $\tilde{\alpha}_0^{(2)}(s) := \begin{cases} 1 & \text{pentru } s = 0 \\ 0 & \text{pentru } s \neq 0 \end{cases}$, $\tilde{\beta}_N^{(2)}(s) := \tilde{\alpha}_N^{(2)}(s),\ \forall s.$

$L_{12}^e(u_k)$ se determină din D_1 după prima trecere, deci nu se iniţializează.

2. (A n-a iteraţie):

$\mathbf{D_1}$: pentru $k = 1, 2, \ldots, N$

- Se determină $y_k := y_k^s y_k^{1p}$ unde y_k^{1p} sunt biţii de control recepţionaţi pentru E_1 (posibil perturbaţi de canal);
- Se determină $\gamma_k(s', s)$ pentru toate tranziţiile posibile $s's$;
- Se determină $\tilde{\alpha}_k^{(1)}(s),\ \forall s.$

pentru $k = N, N-1, \ldots, 2$ se determină $\tilde{\beta}_{k-1}^{(1)}(s),\ \forall s$;

pentru $k = 1, 2, \ldots, N$ se determină $L_{12}^e(u_k)$ cu valorile de probabilităţi asociate lui D_1.

$\mathbf{D_2}$: pentru $k = 1, 2, \ldots, N$

- Se determină $y_k := y_{P_N[k]}^s y_k^{2p}$;
- Se determină $\gamma_k(s', s)$ pentru toate tranziţiile posibile $s's$;
- Se determină $\tilde{\alpha}_k^{(2)}(s),\ \forall s$;

pentru $k = N, N-1, \ldots, 2$ se determină $\tilde{\beta}_{k-1}^{(2)}(s),\ \forall s$;

pentru $k = 1, 2, \ldots N$ se determină $L_{21}^e(u_k)$ cu valorile de probabilităti asociate lui D_2;

3. (După ultima iteraţie):

Pentru $k = 1, 2, \ldots, N$

- Se determină $L_1(u_k) := L_c \cdot y_k^s + L_{21}^e(u_{P_N^{-1}[k]}) + L_{12}^e(u_k)$;
- Dacă $L_1(u_k) > 0$ atunci $u_k := +1$ altfel $u_k := -1$;

4. Stop.

20.3 Exerciţii

20.1 *Demonstraţi Propoziţia 20.1.*

20.2 *Demonstraţi Propoziţia 20.2,* (3).

20.3 *Calculaţi* $w_{(2)},\ w_{(10)}$ *şi* $w_{(16)}$ *pentru numerele* 100, 32412, 999, 1024.

20.4 *Fie* $x \in \mathcal{Z}$. *Un cod Booth este o reprezentare* $x = \sum_{i=0}^{\infty} c_i 3^i$ *unde* $c_i \in \{-1, 0, 1\}$.

1. *Să se reprezinte în codul Booth numerele* 23, 455, 81, -6493*;*
2. *Să se arate că pentru orice număr întreg, codul Booth este unic.*

20.5 *Determinaţi AN - codurile ciclice din* Z_{2^3-1} *şi* Z_{3^3-1}.
Stabiliţi valorile a şi b pentru fiecare din ele.

20.6 *Generalizaţi Exemplul 20.3. Găsiţi un AN - cod ciclic perfect corector de o eroare pentru* $r = 3$.

20.7 *Scrieţi în forma NAF pentru* $r = 2$, $r = 10$ *şi* $r = 7$ *numerele*
$$-15,\ 32075,\ 5665,\ -992.$$

20.8 *Completaţi demonstraţia Teoremei 20.2, verificând unicitatea re-prezentării NAF în cazurile* $c_1' = c_1 + 1$ *şi* $c_1' = c_1 + 1 + r$.

20.9 *În definiţia reprezentării NAF a numărului întreg* x *(cu completarea* $n_0 = -1$*), să se arate că*
$$n_x \leq k \quad \Longleftrightarrow \quad |x| < \frac{r^{k+2}}{r+1}.$$

20.10 *Considerăm reprezentarea ternară modulo* $3^6 - 1$. *Să se determine forma CNAF pentru numărul* 455.

20.11 *Determinaţi cuvintele - cod din codul Mandelbaum - Barrows cu* $b = 11$, $r = 3$, $n = 5$.

20.12 *Fie* $g_0(X) = 1 + X + X^3$, $g_1(X) = 1 + X^2 + X^4 + X^5$ *din* $Z_2[X]$. *Construiţi circuitul liniar pentru codificatorul RSE.*

Codificaţi mesajele de informaţie $1 + X^2 + X^3, \quad 1 + X^7, \quad X + X^4 + X^5$.

20.13 *Aceeaşi problemă pentru polinoamele* $g_0(X) = 1+X^3+X^4$, $g_1(X) = X+X^3+X^6$.

20.14 *Să se construiască un turbo-codificator folosind codificatoarele RSE din Exemplul 20.7,* $N = 3$, *matricea de comutaţie* $P_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ *şi fără mecanism de relaxare.*

Să se codifice mesajul de informaţie 100 011 101.

Bibliografie

[1] J. Adamek - *Foundations of Coding*, Wiley - Interscience, 1991;

[2] L. Bahl, J. Cocke, F. Jelinek, J. Raviv - *Optimal decoding of linear codes for minimizing symbol error rate*, IEEE Inf. Theory, pp. 284-287, Martie 1974;

[3] M. Blaum - *A* $(16, 9, 6, 5, 4)$ *error correcting dc-free block code*, IEEE Transactions on Information Theory, vol. 34, 1988, pp. 38-141;

[4] C. Carlet - *Codes de Reed - Muller; Codes de Kerdok et de Preparata* (teză de doctorat), PARIS VI, 1990;

[5] G. Cullmann - *Coduri detectoare şi corectoare de erori*, Editura Tehnică, 1972;

[6] S. Guiaşu - *Teoria Codurilor*, Tipografia Universităţii Bucureşti, 1976;

[7] D.G. Hoffman, D.A. Leonard, C.C. Lindner, K.T. Phelps, C.A. Rodger, J.R. Wall - *Coding Theory; The Essentials*, Marcel Dekker, Inc, 1991;

[8] B. Honary, G. Markarian - *Trellis Decoding of Block Codes, A Practical Approach*, Kluwer Academic Publ., 1997;

[9] A. M. Kerdok - *A class of low-rate non linear codes*, Information and control, 20 (1972), 182-187;

[10] J.H. van Lindt - *Coding Theory*, Springer Verlag, 1971;

[11] J.H. van Lindt - *Introduction to Coding Theory*, Springer Verlag, 1982;

[12] W.E.Ryan - *A Turbo Code Tutorial*, IEEE Trans. comm. pp. 1261-1271, Oct. 1996;

Cuprins

Cuprins

10. Alte clase de coduri ciclice
    (a) Coduri Hamming ciclice
    (b) Coduri Hamming ciclice nebinare
    (c) Coduri Goppa
    (d) Sistemul de criptare McElliece
11. Coduri Preparata
    (a) Definirea codurilor Preparata extinse
    (b) Codificarea codurilor Preparata extinse
    (c) Decodificarea codurilor Preparata extinse
    (d) Coduri înrudite cu codul Preparata
12. Coduri convoluţionale
    (a) Coduri liniare şi coduri convoluţionale
    (b) Codificarea codurilor convoluţionale
    (c) Decodificarea codurilor convoluţionale
    (d) Algoritmul Viterbi
13. Coduri reţea
    (a) Structura de reţea a codurilor bloc
    (b) Tehnici de codificare şi decodificare în reţea a codurilor
    (c) Aplicaţii la telefonia mobilă a codurilor reţea
14. Alte clase de coduri
    (a) Coduri aritmetice
    (b) Turbo - coduri

Coduri detectoare si corectoare de erori. Coduri Hamming

În domeniul compresiei de date (considerata în teoria codarii ca si "**codarea sursei**") codarea s-a facut în scopul reducerii dimensiunii reprezentarii, presupunând un canal de comunicatie ideal, fara pierderi. În cele ce urmeaza vom aborda pe scurt o problema diferita, si anume codarea în scopul detectiei si eventual corectiei erorilor ce pot apare pe un canal de comunicatie real, cu erori (considerata în teoria codarii ca si "**codarea canalului**").

În acest al doilea caz este evident ca, pentru a obtine detectie si corectie de erori, trebuie sa încarcam suplimentar fluxul cu biti dedicati acestui lucru, având de a face cu o crestere a dimensiunii reprezentarii (trebuie platit un pret...). În timp ce la compresia datelor se elimina cât mai mult posibil redundanta, codurile corectoare adauga acel nivel de redundanta necesar pentru a transmite datele în mod eficient si cu fidelitate pe un canal cu zgomot (cu erori).

1. Distanta Hamming

Pe lânga distantele prezentate într-un material anterior, o distanta de un interes deosebit în problema noastra este distanta Hamming, numita astfel dupa Richard Hamming, cel care a introdus-o în 1950. **Distanta Hamming** între doi vectori de dimensiuni egale este data de **numarul de pozitii în care acestia difera**. Ea masoara astfel numarul de schimbari care trebuie facute într-un vector pentru a îl obtine pe celalalt, sau reformulat numarul de *erori* care transforma un vector în celalalt.

Exemple:

vector 1	**codare**	**126359**	**01101011**
vector 2	**notate**	**226389**	**01001110**
distanta Hamming	3	2	3

Desi definirea este generala, în cele ce urmeaza von considera doar cazul vectorilor cu elemente binare, fiind vorba de fluxuri de biti transmise pe canalul de comunicatie. În acest caz distanta este data de numarul de 1 din rezultatul obtinut prin XOR.

Pentru început vom considera doar cazul simplu al **erorilor singulare** - adica avem un singur bit eronat.

2. Un exemplu simplu de detectie de erori singulare

Sa consideram urmatoarea solutie simpla de codare – fiecare bit este codat prin repetarea sa de doua ori. Introducem astfel în mod evident o redundanta care însa ne va permite sa detectam erori singulare.

Daca la receptie obtinem coduri 00 respectiv 11 am receptionat corect 0 respectiv 1 iar daca obtinem 01 sau 10 am detectat o eroare singulara (fara a putea decide însa nimic în sensul corectiei ei).

Exemplu:

mesaj initial:	0.1.0.0.1.0.1.1.0.1
mesaj codat:	00.11.00.00.11.00.11.11.00.11
mesaj receptionat cu eroare:	00.11.00.00.**10**.00.11.11.00.11 => **detectie** de eroare

Remarcam faptul ca între oricare doua ”cuvinte de cod” valide avem o distanta Hamming 2.

3. Un exemplu simplu de corectie de erori singulare

Sa consideram urmatoarea solutie simpla de codare – fiecare bit este codat prin repetarea sa de trei ori. Introducem astfel în mod evident o redundanta si mai mare care însa ne va permite sa detectam si sa corectam erori singulare.

Daca la receptie obtinem coduri 000 respectiv 111 am receptionat corect 0 respectiv 1. Daca obtinem 001 sau 010 sau 100 am detectat o eroare singulara si putem **corecta** spre 000 (deci am receptionat 0) iar daca obtinem 011 sau 101 sau 110 am detectat o eroare singulara si putem **corecta** spre 111 (deci am receptionat 1).

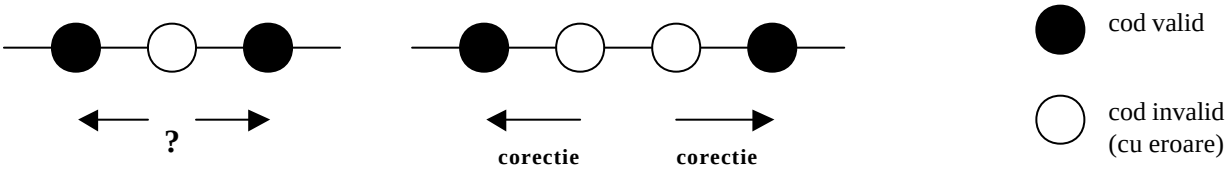
Exemplu:

mesaj initial:	0.1.0.0.1.0.1.1.0.1
mesaj codat:	000.111.000.000.111.000.111.111.000.111
mesaj receptionat cu eroare:	000.**101**.000.000.111.000.111.111.**001**.111
mesaj corectat:	000.**111**.000.000.111.000.111.111.**000**.111

=> **detectie** si corectie de erori singulare

Remarcam faptul ca între oricare doua ”cuvinte de cod” valide avem o distanta Hamming 3.

Din exemplele anterioare constatam ca, daca distanta Hamming între oricare cuvinte de cod valide este 2, putem detecta erori singulare dar nu le putem corecta - nu stim care cod valid aflat la distanta 1 este cel corect. Daca distanta Hamming între oricare cuvinte de cod valide este 3 putem detecta erori singulare si putem corecta înspre codul valid aflat la distanta 1 (remarcam faptul ca, în acest caz, putem detecta chiar erori duble ! – fara însa a le putea corecta).



Daca scopul este doar de a detecta erori se pot folosi si alte solutii cum ar fi simpli biti de paritate, sume de control, coduri ciclice de tip CRC, functii hash criptografice, etc.

4. Codul Hamming (7,4)

Unul din cele mai cunoscute coduri detectoare si corectare de erori singulare este codul numit Hamming (7,4). Acesta notatie indica faptul ca avem un cod de 7 biti din care 4 sunt biti de date independenti (restul fiind biti redundanti, reprezentând paritatea a diferite combinatii a bitilor de date). Codul contine deci 4 biti de date d1, d2, d3, d4 si 3 biti de paritate p1, p2, p3. Bitii de paritate sunt calculati astfel:

p1 sumeaza d1, d2, d4
p2 sumeaza d1, d3, d4
p3 sumeaza d2, d3, d4

iar organizarea bitilor în cuvântul de cod este urmatoarea:

p1	p2	d1	p3	d2	d3	d4

De obicei se definesc doua matrici în legatura cu acest cod: matricea generatoare de cod G si matricea de detectie a paritatii H

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

Se constata în G ca liniile 1,2 si 4 calculeaza sumele aferente paritatilor respective iar liniile 3,5,6,7 simplu copiaza bitii de date. În H cele 3 linii calculeaza paritatile corespunzatoare.

La codare: se determina cuvântul de cod calculând paritatile corespunzatoare (se poate utiliza atât paritatea para cât si paritatea impara - în exemplele urmatoare vom folosi paritatea impara).

La decodare: se calculeaza paritatile corespunzatoare si se verifica cu cele corecte (în fapt se sumeaza si cu paritatile corecte si se verifica sa rezulte 0).

Exemplu de codare-decodare pentru codul Hamming (7,4)

La codare: fie cuvântul de cod 1001. Calculam:

p1 = 1+0+1 = 0

p2 = 1+0+1 = 0

p3 = 0+0+1 = 1

rezultând codul Hamming **0 0 1 1 0 0 1** (am reprezentat subliniat bitii de paritate).

La decodare (cazul 1): presupunem ca am receptionat 0011001 (deci fara erori)

Reconstituim bitii de date: 1 0 0 1

Verificam paritatile:

p1 + d1 + d2 + d4 = 0 + 1 + 0 + 1 = 0

p2 + d1 + d3 + d4 = 0 + 1 + 0 + 1 = 0

p3 + d2 + d3 + d4 = 1 + 0 + 0 + 1 = 0

Cum toate aceste valori sunt 0 rezulta ca nu am avut eroare.

La decodare (cazul 2): presupunem ca am receptionat 0011011 (deci cu o eroare în zona de date)

Reconstituim bitii de date: 1 0 1 1

Verificam paritatile:

p1 + d1 + d2 + d4 = 0 + 1 + 0 + 1 = 0

p2 + d1 + d3 + d4 = 0 + 1 + 1 + 1 = 1

p3 + d2 + d3 + d4 = 1 + 0 + 1 + 1 = 1

Cum aceste valori nu sunt toate 0 rezulta ca am avut eroare (deci am realizat **detectie** de eroare). În plus codul format din pozitiile eronate (aferent p3p2p1) având valoarea 110 indica bitul 6 ca fiind eronat deci al treilea bit de date trebuie corectat (deci am realizat **corectie** de eroare). Deci, bitii de date corectati sunt 1 0 0 1.

Pozitionarea initiala aparent ciudata a bitilor de paritate în cadrul codului Hamming este necesara pentru a obtine simplu, direct indicele bitului eronat.

La decodare (cazul 3): presupunem ca am receptionat 0111001 (deci cu o eroare în zona de paritate)

Reconstituim bitii de date: 1 0 0 1

Verificam paritatile:

p1 + d1 + d2 + d4 = 0 + 1 + 0 + 1 = 0

p2 + d1 + d3 + d4 = 1 + 1 + 0 + 1 = 1

p3 + d2 + d3 + d4 = 1 + 0 + 0 + 1 = 0

Cum aceste valori nu sunt toate 0 rezulta ca am avut eroare (deci am realizat **detectie** de eroare). În plus codul format din pozitiile eronate (aferent p3p2p1) având valoarea 010 indica bitul 2 ca fiind eronat deci al doilea bit de paritate era eronat iar bitii de date erau corecti (deci am realizat **corectie** de eroare).

Codul Hamming (7,4) descris anterior are distanta Hamming între oricare cuvintele de cod valide minim 3 si deci poate **detecta si corecta erori singulare** sau poate doar detecta erori duble (fara a putea corecta nici un fel de erori).

Calculele anterioare se pot face si în forma matriciala considerând la codare înmultirea matricii G cu vectorul coloana al bitilor de date iar la verificarea paritatii înmultirea matricii H cu vectorul coloana al codului Hamming.

5. Coduri Hamming generale

Codul Hamming descris anterior a fost generalizat pentru orice numar de biti. Algoritmul general pentru proiectarea unui cod Hamming **corector de erori singulare** (”single error correcting” - SEC) este urmatorul:

- Numerotam bitii începând cu 1: 1, 2, 3, 4, etc.
- Fiecare pozitie care este putere a lui 2 reprezinta **bit de paritate** (pozitiile care au un singur bit în reprezentarea binara)
- Celelalte pozitii reprezinta **biti de date** (pozitiile care au mai mult de un bit pe 1 în reprezentarea binara).
- Fiecare bit de paritate acopera biti astfel: bitul k de paritate acopera acele pozitii care au bitul k cel mai semnificativ pe 1.

Desi modul de calcul al paritatii (pare sau impare) nu este esential, de regula însa se foloseste paritatea impara (paritatea e 1 daca numarul de biti de 1 este impar).

Algoritmul general poate fi înteles mai bine din figura urmatoare:

pozitie		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	...
cod		p1	p2	d1	p3	d2	d3	d4	p4	d5	d6	d7	d8	d9	d10	d11	p5	d12	d13	...
componenta biti paritate	p1	X		X		X		X		X		X		X		X		X		...
	p2		X	X			X	X			X	X			X	X			X	...
	p3				X	X	X	X					X	X	X	X				...
	p4								X	X	X	X	X	X	X	X				...
	p5																X	X	X	...

S-au reprezentat doar 5 biti de paritate si 13 biti de date având de a face cu codul care se noteaza Hamming(18,13).

Daca avem m biti de paritate putem acoperi pâna la $2^m - 1$ biti. Daca scadem cei m biti de paritate ramân $2^m - m - 1$ biti care pot fi folositi pentru date. În functie de valoare lui m avem urmatoarele coduri Hamming:

Biti de paritate	Total biti	Biti date	Nume cod	Eficienta utilizare biti date
2	3	1	Hamming(3,1)	1/3 = 0.333
3	7	4	Hamming(7,4)	4/7 = 0.571
4	15	11	Hamming(15,11)	11/15 = 0.733
5	31	26	Hamming(31,26)	26/31 = 0.839
m	2^m-1	2^m-m-1	Hamming(2^m-1, 2^m-m-1)	$1 - m / (2^m-1)$

Codurile Hamming descrise au distanta Hamming minima 3 deci permit:

- detectie si corectie a erorilor simple
- detectie a erorilor duble daca se renunta la corectie.

Uneori se adauga un **bit suplimentar de paritate** a întregului cod facând distanta minima 4. În acest caz se poate distinge între erorile simple (care se pot corecta) si erorile duble care doar se

detecteaza. Codurile astfel rezultate se numesc SECDED ("single error correction, double error detection".

De un interes deosebit este codul (72,64) care reprezinta o versiune trunchiata a codului cu 7 biti de paritate (m=7) adica Hamming (127,120) varianta cu bit aditional de paritate. Acesta este folosit pe circuitele de memorie de 72 biti pentru detectia erorilor duble si corectia erorilor singulare. Codul respectiv are o eficienta de utilizare a bitilor de date de 64/72 = 0.8888 identica cu situatia simpla a utilizarii unui bit de paritate pe fiecare byte, situatie corespunzatoare unui cod de tip (9,8). În plus fata de codul (9,8) care permite doar detectia erorilor singulare, codul (72,64) este de tip SECDEC adica permite detectia erorilor duble si corectia erorilor simple.

Capitolul 5

Coduri grup - coduri Hamming

5.1 Breviar teoretic

Dacă în capitolul precedent s-a pus problema codării surselor pentru eficientiezarea unei transmisiuni ce se presupunea a nu fi perturbată de erori, de această dată ne adresăm unei transmisiuni în condiţii de zgomot, când mesajul transmis este modificat de erori. Cerinţa este să se genereze coduri capabile să detecteze şi corecteze erorile apărute pe parcurs.

Operaţii cu elemente ale mulţimii $\{0,1\}$ Simbolurile ce intră în discuţie nu pot lua decât valori de 0 sau 1. Operaţiile obişnuite în acest caz se desfăşoară conform tabelelor:

$+$	0	1
0	0	1
1	1	0

Tabela 5.1: Adunarea elementelor mulţimii $\{0,1\}$.

$\cdot$	0	1
0	0	0
1	0	1

Tabela 5.2: Înmulţirea elementelor mulţimii $\{0,1\}$.

Distanţa Hamming Distanţa Hamming între două cuvinte este egală cu numărul poziţiilor în care cele două cuvinte diferă. De exemplu:

- distanţa Hamming între 0010101 şi 1010101 este 1 întrucât diferă doar simbolul de pe prima poziţie.
- distanţa Hamming între 0010101 şi 0000000 este 3 întrucât diferă simbolurile de pe poziţile 3, 5 şi 7.

Erori. Detecţie şi corecţie Dacă v este un cuvânt (vector de simboluri) de cod valid, cuvănt care este transmis pe un canal cu perturbaţii, iar ε este vectorul perturbator, atunci $v' = v + \varepsilon$ este cuvântul recepţionat. Dacă perturbarea a fost cu o singură eroare, atunci cuvântul eroare ε va avea un singur 1, pe pozţia modificată şi în rest 0.

Distanţa Hamming între cuvântul transmis şi cuvântul recepţionat este 1 (şi este egală cu numărul erorilor introduse. Dacă toate combinaţiile posibile cu k biţi sunt considerate cuvinte, atunci distantă minimă între cuvinte este 1.

În acest caz când toate combinaţiile posibile ale simbolurilor de informaţie sunt cuvinte de cod, iar în timpul transmisiunii apare o eroare atunci cuvântul recepţionat va fi tot un cuvânt cu sens deşi este greşit. De aceea dacă se doreşte detecţia erorilor cuvintele de cod trebuie spaţiate (crescută distantă între ele). Acest lucru se realizează prin adăugarea simbolurilor de control. Acestea sunt în număr de m, iar lungimea unui cuvânt de cod este: $n = k + m$. Simbolurile de control vor fi combinaţii ale simbolurilor de informaţie care vor fi transmise astfel de mai multe ori, crescându-se redundanţa.

Pentru detecţie a e_d erori, distanţa minimă între două cuvinte de cod trebuie să fie $d_{min} = e_d + 1$; în acest mod, orice cuvânt cu sens, ce ulterior va fi eronat, va fi plasat la o distanţă maximă e_d şi va conduce la un cuvânt fără sens.

Pentru corecţie distanţa trebuie mărită. Dacă se doreşte un cod cabapil să corecteze e_c erori atunci distanţa minimă trebuie să fie $d_{min} = 2e_c + 1$; în acest fel, fiind dat un cuvânt fără sens, se poate identifica şi cuvântul cu sens din care a provenit.

Identificarea erorii se face cu ajutorul simbolurilor de control. Dacă un cod este capabil să corecteze 1 eroare, aceasta poate fi pe oricare din cele n poziţii ale cuvântului de cod. Pentru 2 erori, acestea pot fi în orice combinătie de n luate câte 2. Generalizând, dacă sunt e_c erori atunci cazurile posibile sunt $\sum_{i=1}^{e_c} C_n^i$. Având m simboluri de control numărul cuvintelor construibile cu acestea sunt 2^m. Dacă reţinem o poziţie pentru cuvântul corect iar restul sunt folosite pentru identificarea erorilor atunci putem scrie relaţia cunoscută ca "marginea Hamming":

$$2^m - 1 \geq \sum_{i=1}^{e_c} C_n^i \tag{5.1}$$

Codarea Codarea presupune construcţia cuvântului de cod pornind de la simbolurile de informaţie. Există două variante de codare:

1. Codarea $v = iG$, unde v este vectorul asociat cuvântului de cod, i este vectorul asociat cuvântului de informaţie, iar G este matricea generatoare. Aceasta are k linii şi n coloane. Dacă $\mathbf{g_1}, \mathbf{g_2}, \ldots \mathbf{g_k}$ sunt liniile matricei G, iar $i_1, i_2, \ldots i_k$ sunt simbolurile de informaţie atunci relaţia de codare se poate rescrie astfel:

   $$v = i_1\mathbf{g_1} + i_2\mathbf{g_2} + \ldots i_k\mathbf{g_k} \tag{5.2}$$

   De aici rezultă că cuvintele de cod sunt toate combinaţiile liniare ale liniilor matricei generatoare.

2. Codarea $Hv^T = 0$, unde H este o matrice de m linii şi n coloane ce este denumită matrice de control. Pentru aflarea relaţiilor de codare, în acest caz, se vor plasa simboluri de control pe poziţiile corespunzătoare coloanelor matricei H având un singur 1, şi se va rezolva sistemul rezultant, având drept necunoscute simbolurile de control.

Decodarea. Corectîa erorilor Decodarea se face pe baza relaţiei $Hv^T = 0$. Dacă v este un cuvânt cu sens (cuvânt nealterat de erori) atunci înmulţindul cu cu H se va obţine 0. Dacă rezultatul este nenul atunci sunt erori. În acest caz, dacă circuitul funcţionează în regim:

1. detecţie – atunci se va semnala existenţa unor erori.
2. corecţie – atunci se vor corecta erorile

Pentru corecţia erorii se calculează sindromul s. Dacă cuvântul recepţionat este $v' = v + \varepsilon$ atunci

$$H(v')^T = Hv^T + H\varepsilon^T = 0 + Hv^T = s \tag{5.3}$$

Sindromul s identifică eroarea. Dacă de exemplu există o singură eroare, pe poziţia 2 atunci sindromul va fi egal cu coloana a 2 a matricei H.

Odată identificate pozţia erorilor (sau cu alte cuvinte identificat vectorul eroare) corecţia se obţine adunând 1 simbolurilor corecpunzătoare recepţionate. Adică $v = v' + \varepsilon$.

5.2 Probleme rezolvate

1. [8] *Un cod grup are matricea de control*:

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

   (a) *Să se determine numărul de simboluri de informaţie şi numărul de simboluri de control. Să se determine proprietăţile de corecţie / detecţie ale acestui cod. Acest cod este perfect?*

   (b) *Să se calculeze matricea generatoare a codului.*

   (c) *Să se deducă relaţiile de codare.*

   (d) *Să se realizeze codarea atât cu matrice G cât şi c matricea H.*

   (e) *Cuvântul* [11111] *este cuvânt de cod?. Să se explice funcţionare decodorului în cazul in care se recepţionează acest cuvânt.*

   Rezolvare:

   (a) Parametrii codului : se ştie că matricea H are m (numărul de simboluri de control) linii şi n (lungimea cuvântului de cod) coloane:

$$m = 3, n = 5 \Longrightarrow k = 2$$

   Având k simboluri de informaţie numarul maxim de mesaje care se pot coda cu acest cod sunt: $2^k = 4$ mesaje ale sursei Numărul de erori corectabile este de dat de marginea Hamming (aceasta este o condiţie necesară nu şi suficientă):

$$2^m - 1 \geq \sum_{i=1}^{e_c} C_n^i$$

Membrul stâng este: $2^m - 1 = 7$.
Membrul drept este:

$$\hat{\text{I}}\text{n cazul unei erori } n = 5 < 2^{m-1}$$

$$\hat{\text{I}}\text{n cazul a 2 erori } C_n^1 + C_n^2 > 7$$

$$\Longrightarrow \text{codul e corector de o eroare}$$

Un cod corector de e_c erori poate detecta $e_d = 2e_c$ erori.Un cod capabil să corecteze e_c erori are distanţa minimă $d_min = 2e_c + 1 = e_d + 1$.
Codul nu este perfect (nu se obţine egalitate in marginea Hamming). Un cod perfect are exact numarul de corectori necesari pentru a detecta orice variantă de eroare.

(b) Matricea de control a codului este scrisă în forma canonică: $H = [I_m Q]$. Matricea generatore (in formă canonică) se poate obtine ca: $G = [Q^t I_k]$.;

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

$$\Longrightarrow Q^t = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \Longrightarrow G = [Q^t I_k] = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

(c) Structura unui cuvânt de cod sistematic presupune o separare a biţilor de control de cei de informaţie. Făcând convenţie că biţi de control corespund coloanelor din H care au un singur 1 succesiunea simbolurilo într-un cuvânt de cod este: $v = [c_1 c_2 c_3 i_1 i_2]$. De fapt forma canonica a matricelor de control si generatoare impune un cod sistematic.Relaţiile de codare presupun aflarea modului în care se formează biţii de control din biţii de informaţie. Vom considera ca punct de pornire relaţia $Hv^T = 0$

$$\Longleftrightarrow \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} c_1 + i_1 + i_2 \\ c_2 + i_2 \\ c_3 + i_1 \end{bmatrix} = 0$$

In baza 2 scăderea cu $-x$ este echivalentă cu adunarea cu x. Adică:

$$c_1 = i_1 + i_2$$
$$c_2 = i_2$$
$$c_3 = i_1$$

Schema codorului este prezentată în figura 5.1;

(d) Pentru a calcula cuvintele de cod avem doua variante: să calculăm relaţiile de codare folosind matricea H sau să calculam cuvintele de cod folosind direct matricea G:

$$v = iG$$

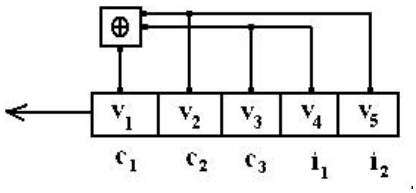


Figura 5.1: Codorul. Schema logică urmareşte relaţiile de codare. Fiecare simbol este stocat într-o celulă a unui registru de deplasare; la fiecare tact, o celulă comunică valoarea celulei din stânga sa. După formarea cuvântului de cod intrările (biţi de infomaţie) sunt blocate si conţinutul este vărsat la ieşire sub forma unui tren de impulsuri.

Un cuvânt de infomaţie conîne două simboluri: $i = \left[\alpha\beta\right]$. Matricea generatoare poate scrisă pe linii $G = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}$. Atunci relaţia de codare devine:

$$V = iG = \left[\alpha\beta\right] \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} = \alpha g_1 + \beta g_2$$

In acest caz cuvintele de cod sunt:

α	β	$v_{\alpha\beta} = \alpha g_1 + \beta g_2$
0	0	0 0 0 0 0
0	1	1 1 0 0 1
1	0	1 0 1 1 0
1	1	0 1 1 1 1

Tabela 5.3: Cuvintele de cod

Se poate observa că ponderea minima a unui cuvânt de cod este 3.

(e) Cuvântul recepţionat este: $v' = [11111]$ Consideăm corectorul:

$$Hv'^{T} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = z = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \neq 0$$

Fiindcă corectorul este nenul există erori. Atunci se poate spune:

- dacă decodorul functionează în regim de corecţie este capabil să corecteze o eroare (orice variantă de eroare). Eroare este pe poziţia 1.Poziţia erorii se determină prin identificare coloanei din matricea de control H egală cu corectorul calculat z. Acelaşi lucru se constată si dacă se compara cuvantul eronat cu cuvintele cu sens determinate în tabelul 5.3. În acest caz cuvăntul cu sens este $v = [01111]$. Dacă

cuvântul recepţionat conţine două erori atunci corectorul dă informaţii greşite. Să consideram alt exemplu :

$$v = [10110] \qquad \varepsilon = [00011] \qquad v' = v + \varepsilon = 10101$$

Corectorul calculat este:

$$Hv'^T = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Adică eroare este pe poziţia 3 si deci cuvântul cu sens este v=[10001]. Ceea ce nu este adevarat.*Un cod corector de o eroare nu poate corecta două erori*

- dacă decodorul functionează in regim de detecţie, fiindcă corectorul este nenul înseamană că există erori. În regim de detecţie nu se poate spune nimic despre numărul si poziţia erorilor. Dacă există mai multe erori decât codul poate detecta, se obin aberaţii. De exemplu pentru un cuvânt eroare $\varepsilon = [11001]$ şi pentru cuvântul cu sens $v = [11001]$ se obţine alt cuvănt cu sens şi, deci nu se detectează nimic. Adică *un cod detector de 2 erori nu poate detecta trei erori.*

2. [8] *Cele* 6 *simboluri generate de o sursă sunt transmise pe un canal binar cu perturbaţii folosind un cod Hamming grup corector de o eroare.*

   (a) *Să se determine numărul de simboluri de informaţie, de control si lungimea cuvintelor de cod. Codul este perfect?*

   (b) *Să se scrie matricile de control şi generatoare a codului? Codul este sistematic?*

   (c) *Să se scrie cuvintele de cod şi să se determine ponderea minimă a acestora.*

   (d) *Să se explice ce se întâmplă dacă într-un cuvânt recepţionat apar două erori, pe poziţiile 1 şi 2.*

Rezolvare:

(a) Cele 6 simboluri pot fi reprezentate folosin k biţi de informaţie:

$$2^k \geq 6 \Longrightarrow k = 3$$

Marginea Hamming pentru un cod corector de $e_c = 1$ erori este:

$$2^m - 1 \geq n = m + k = m + 3$$

$$2^m - 4 \geq m$$

$$\Longrightarrow m = 3$$

$$n = k + m = 6$$

Dat fiind ca nu îşi atinge margine (nu avem egalitate) codul nu e perfect.

(b) Matricea de control a unui cod Hamming se obţine codând pe fiecare coloana indicele ei în baza 2:

$$H = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Dacă facem convenţia că plasăm simbolurile de control pe poziţiile corespunzătoare colanele matricei de control care conţin un singur unu, cuvântul de cod este de forma:

$$v = [c_1 c_2 i_1 c_3 i_2 i_3]$$

În acest caz, codul nu e sistematic.

$$Hv^T = 0$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ i_1 \\ c_3 \\ i_2 \\ i_3 \end{bmatrix} = \begin{bmatrix} c_3 + i_2 + i_3 \\ c_2 + i_1 + i_3 \\ c_1 + i_1 + i_2 \end{bmatrix} = 0$$

$$c_1 = i_1 + i_2$$

$$c_2 = i_1 + i_3$$

$$c_3 = i_2 + i_2$$

Se poate arăta foarte uşor că pe linii G are cuvinte de cod.

$$v = iG = \begin{bmatrix} i_1 & i_2 & i_3 \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} = i_1 g_1 + i_2 g_2 + i_3 g_3$$

$$i_1 = 0; i_2 = 0; i_3 = 1 \Longrightarrow v = g_3 = [010101]$$

$$i_1 = 0; i_2 = 1; i_3 = 0 \Longrightarrow v = g_2 = [100110]$$

$$i_1 = 1; i_2 = 0; i_3 = 0 \Longrightarrow v = g_1 = [111000]$$

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

Alte cuvinte de cod sunt:

$$i_1 = 0, i_2 = 0, i_3 = 0 \Longrightarrow v = [000000]$$
$$i_1 = 0, i_2 = 0, i_3 = 0 \Longrightarrow v = g_1 + g_2 + g_3 = [101011]$$
$$i_1 = 1, i_2 = 1, i_3 = 0 \Longrightarrow v = g_1 + g_2 = [110011]$$
$$i_1 = 0, i_2 = 1, i_3 = 1 \Longrightarrow v = g_2 + g_3 = [011110]$$
$$i_1 = 1, i_2 = 0, i_3 = 1 \Longrightarrow v = g_1 + g_3 = [101101]$$

(c)

$$\varepsilon = [110000]$$
$$H\varepsilon^T = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \neq 0$$

Dat fiind faptul ca corectorul este nenul, se poate spune că:

- Dacă functionarea este in regim de corecţie se hotărăşte că eroarea este pe poziţia 3. Greşeala se datorează apariţiei a doua erori când codul poate corecta numai una. Pentru a corecta două erori este necesar ca suma a oricare doua coloane a lui H sa aibă rezultat diferit. Aici:$h_1 + h_2 = h_3$.
  Dacă corectorul obţinut era $z = [111]^T$, care este diferit de orice coloana a lui H, înseamnă că au fost două erori: fie pe poziţiile 1 şi 6, fie pe 2 şi 5. Acest fenomen se datorează faptului că codul nu e perfect.
- Dacă functionarea este in regim de detecţie se depistează apariţia unor erori.

5.3 Probleme propuse

1. [7] Se consideră o sursă de informaţie având un alfabet de dimensiune $Q = 15$ simboluri echiprobabile.

   (a) Să se determine parametrii k, m, n ai unui cod bloc Hamming corector de erori singulare.

   (b) Să se scrie matricea H de control a codului.

   (c) Să se precizeze structura cuvântului de cod.

   (d) Codul este sistematic? De ce?

   (e) Să se efectueze codarea utilizând matricea H de control a codului.

   (f) Să se scrie matricea G generatoare a codului.

   (g) Să se verifice prin calcul direct relaţia de ortogonalitate între matricile H si G.

   (h) Să se efectueze codarea utilizând matricea G generatoare a codului.

   (i) Să se deseneze schema codorului şi să se explice funcţionarea sa.

   (j) Să se scrie toate cuvintele de cod.

   (k) Să se efectueze codarea Hamming sistematică extinsă cu matricea H_{ext} a vectorului informaţional având nenule doar primul şi ultimul simbol.

(l) Să se scrie matricea generatoare de cod sistematic extins G_{ext}.

(m) Se consideră cuvântul de cod extins având simbolul de control a parităţii eronat. Să se scrie vectorul eroare extins. Să se calculeze vectorul corector (extins).

(n) Să se scrie vectorul eroare pentru eroare dublă, de simboluri informaţionale consecutive în partea centrală a zonei informaţionale a cuvântului de cod sistematic.

(o) Să se scrie corectorul extins pentru eroarea dublă de mai sus şi să se explice utilizarea sa.

2. [7] Un număr de 20 simboluri se transmit pe un canal cu perturbaţii utilizând cod Hamming grup corector de o eroare.

(a) Să se determine numărul simbolurilor de informaţie k, al celor de control m si lungimea n a fiecărui cuvânt de cod.

(b) Să se scrie matricea de control a codului H.

(c) Să se scrie formele canonice ale matricei de control.

(d) Să se scrie formele canonice ale matricei generatoare.

(e) Să se deducă matricea generatoare.

(f) Să se scrie toate cuvintele de cod.

(g) Să se stabilească expresia corectorului pentru cazul că se eronează poziţia 4 din cuvântul de cod.

(h) Să se explice ce se întâmplă dacă într-un cuvânt de cod se eronează poziţiile 2 şi 7.

(i) Să se stabilească schema codorului.

3. [8] Se dă matricea de control a unui cod grup:

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

(a) Să se determine numărul de simboluri de control, numărul de simboluri de informaţie, lungimea cuvintelor de cod, numărul de simboluri ce pot fi transmise cu acest cod şi numărul de erori ce pot fi corectate. Codul este perfect? Codul este sistematic?

(b) Să se precizeze structura cuvintelor de cod şi să se scrie ecuaţiile de codare.

(c) Să se determine matricea generatoare a codului.

(d) Să se calculeze corectorul şi să se explice decizia luată la decodare dacă se recepţionează un cuvânt eronat pe poziţiile 2 si 3?

4. [4] Fie matricea de control

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

   (a) Arătaţi că, prin transformări elementare, această matrice poate fi adusă la forma $H' = [I_3 Q]$.

   (b) Arătaţi că respectivele transformări pot fi astfel alese încât proprietăţile de detecţie şi corecţie a erorilor să rămână aceleaşi.

   (c) Să se determine simbolurile de control în funcţie de cele de informaţie atât pentru matricea H cât şi pentru matricea H'.

5. [4] Considerând matricea

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

   să se determine matricea generatoare $G = [PI_k]$ şi să se realizeze codarea după aceasta.

6. [4] Se consideră un cod cu $n = 6$ şi $k = 3$ a cărui matrice de control este:

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

   (a) Codul este sau nu perfect?

   (b) Ce decizie de ia pentru un corector cu valoarea $z^t = [010]$? Dar pentru $z^t = [111]$

Bibliografie

[1] Mihai Ciuc. “Note de seminar”.

[2] A. T. Murgan, I. Spânu, I. Gavăt, I. Sztojanov, V. E. Neagoe, şi A. Vlad. *Teoria Transmisiunii Informatîei - probleme*. Editura Didactică şi Pedagogică, Bucureşti, România, 1983.

[3] Alexandru Spătaru. *Teoria Transmisiunii Informattţiei*. Editura Didactică şi Pedagogică, Bucureşti, România, 1983.

[4] Alexandru Spătaru. *Fondements de la theorie de la transmisssion de línformation*. Presses polytechniques romandes, Lausanne, Elveţia, 1987.

[5] Rodica Stoian. “Note de seminar”.

[6] Dan Alexandru Stoichescu. “Note de seminar”.

[7] Eugen Vasile. “Note de seminar”.

[8] Constantin Vertan. “Note de seminar”.

2

Coduri detectoare/corectoare de erori. Criptarea informaţiei

1. Prezentare teoretică

În cadrul acestei lucrări de laborator se vor prezenta algotimii CRC şi Reed-Solomon folosiţi la detectarea şi corectarea erorilor care pot apărea într-o transmisie de date. Algoritmii RSA şi IDEA prezentaţi sunt uzual folosiţi pentru criptarea informaţiei şi se bazează pe chei publice. Implementările hardware ale altor algoritmi de criptare, care se bazează pe metode tradiţionale (de exemplu algoritmul de criptare DES), pot fi studiate la http://www.csit-sun.pub.ro/resources.

Sume de control

Scopul unei tehnici de detecţie a erorilor este acela de a pune la dispoziţia receptorului unui mesaj, transmis printr-un canal cu zgomote (pasibil de introducere de erori), o metodă de a determina dacă mesajul a fost corupt sau nu. Pentru a face posibil acest lucru, emiţătorul construieşte o valoare numită sumă de control care este o funcţie de mesaj şi o anexează acestuia. Receptorul poate să folosească aceeaşi funcţie pentru a calcula suma de control pentru mesajul primit, iar apoi să o compare cu suma de control anexată (concatenată mesajului) pentru a vedea dacă mesajul a fost receptat corect.

Exemplu Să se aleagă o funcţie care are ca rezultat (sumă de control) suma octeţilor din mesaj modulo 256:

$$f(x) = \sum(octeti \quad mesaj) \bmod 256 \qquad (1)$$

Considerând toate valorile în zecimal, se obţine:

mesaj	:	7 24 3
mesaj cu suma de control	:	7 24 3 34
mesaj după transmisie	:	7 28 3 38

Al doilea octet al mesajului a suferit o modificare în timpul transmisiei, de la 24 la 28. Cu toate acestea, receptorul poate determina prezenţa unei erori comparând suma de control transmisă (34) cu cea calculată (38 = 7 + 28 + 3).

Dacă însăşi suma de control este coruptă, un mesaj transmis corect poate fi (incorect) interpretat drept unul eronat. Acesta nu este însă un eşec periculos. Un eşec periculos are loc atunci când atât mesajul cât şi suma de control se modifică astfel încât rezultă într-o transmisie consistentă intern (interpretată ca neavând erori).

Din păcate, această posibilitate nu poate fi evitată şi cel mai bun lucru care se poate realiza este de a minimiza probabilitatea ei de apariţie prin creşterea cantităţii de informaţie din suma de control (de exemplu, lărgind dimensiunea ei la doi octeţi în loc de unul).

Coduri CRC

Ideea de bază pentru algoritmii CRC este de a trata mesajul drept un număr reprezentat în binar, de a-l împărţi la un alt număr binar fixat şi de a considera restul drept sumă de control. La primirea mesajului, receptorul poate efectua aceeaşi împărţire şi poate compara restul cu suma de control primită (restul transmis).

Exemplu Considerând că mesajul care trebuie transmis este alcătuit din 2 octeţi (6, 23), el este reprezentat în baza 16 ca numărul 0617 şi în baza 2 ca 0000_0110_0001_0111. Se presupune folosirea unei sume de control de l octet şi a unui împărţitor constant 1001. Atunci suma de control va fi restul împărţirii 0000_0110-0001_0111 : 1001 = ... 0000010101101, rest 0010. Mesajul transmis de fapt va fi: 06172, unde 0617 este mesajul iniţial (informaţia utilă), iar 2 este suma de control (restul).

Aritmetica binară fără transport

Toate calculele executate în cadrul algoritmilor CRC sunt realizate în binar, fără transport. Deseori se foloseşte denumirea de aritmetică polinomială, dar în continuare se va folosi denumirea de aritmetică CRC deoarece la implementarea cu polinoame s-a renunţat.

Adunarea a două numere în aritmetica CRC, aşa cum se poate observa în figura 1, este asemănătoare cu adunarea binară obişnuită, însă nu există transport. Aceasta înseamnă că fiecare pereche de biţi corespondenţi determină bitul corespondent din rezultat, fără nici o referinţă la alt bit din altă poziţie (aşa cum se poate observa din exemplul prezentat în figura 2 a).).

Definiţia operaţiei de scădere este identică cu operaţia de adunare şi poate fi observată în figura 1, iar un exemplu este prezentat în figura 2 b).

Se poate concluziona că atât adunarea cât şi scăderea în aritmetica CRC sunt echivalente cu operaţia SAU EXCLUSIV (XOR), iar operaţia XOR este propria sa inversă. Acest fapt reduce operaţiile primului nivel de putere (adunare, scădere) la una singură, care este propria sa inversă (o proprietate foarte convenabilă a acestei aritmetici).

a	b	a + b
0	0	0
0	1	1
1	0	1
1	1	0

a	b	a - b
0	0	0
0	1	1
1	0	1
1	1	0

Figura 1: Definirea operaţiilor de adunare/scădere.

Pe baza adunării, se poate defini şi înmulţirea, care se realizează natural, fiind suma dintre primul număr deplasat corespunzător şi cel de-al doilea număr (se foloseşte adunarea CRC). Un exemplu pentru această operaţie este prezentat în figura 2 c).

Pentru realizarea operaţiei de împărţire, este nevoie să se cunoască când *un număr este cuprins în altul.* De aceea, se va considera următoarea definiţie: *X* este mai mare decât sau egal cu *Y* dacă poziţia celui mai semnificativ bit l al lui X este mai mare sau aceeaşi cu poziţia celui mai semnificativ bit l al lui Y. Un exemplu complet este prezentat în figura 2 d).

Transmisia - recepţia datelor folosind CRC

Aşa cum s-a arătat până acum, calculul CRC este de fapt o simplă împărţire. Pentru realizarea unui calcul CRC este nevoie de un divizor, denumit în limbaj matematic *polinom generator.* Lungimea polinomului uzuală este de 16 sau 32 de biţi, CRC-16, CRC-32, şi aceste dimensiuni sunt folosite în calculatoarele digitale moderne. *Lungimea unui polinom - W-* este de fapt poziţia celui mai semnificativ bit l (lungimea polinomului 10011 este 4).

La transmiţător, înainte de calculul CRC, se adaugă *W* biţi cu valoarea 0 la sfârşitul mesajului care va fi împărţit folosind aritmetica CRC la polinom, astfel încât toţi biţii mesajului să participe la calculul CRC. Un exemplu este prezentat în figura 2 d). Împărţirea produce un cât, care nu va fi ignorat şi un rest, care este suma de control

calculată (CRC-ul). În mod uzual CRC-ul este apoi adăugat mesajului, iar rezultatul este trimis către receptor, în acest caz se transmite 11010110111110.

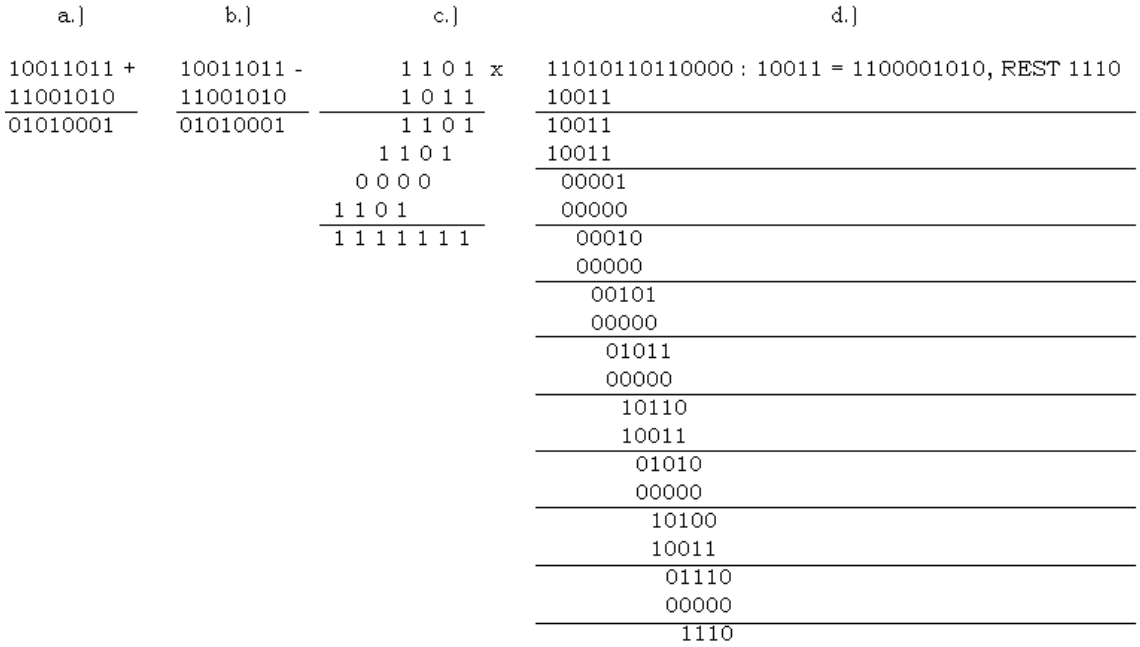


Figura 2: Exemplificarea operaţiilor binare fără transport

Receptorul calculează suma de control pentru întreg mesajul primit (fără adăugare de zerouri) şi compară restul cu 0. Realizarea acestei operaţii este motivată de faptul că mesajul transmis *T* este multiplu de polinomul folosit drept divizor.

Implementarea directă

CRC-ul se poate calcula utilizând noţiunile teoretice prezentate până acum. Algoritmul, implementarea Verilog precum şi rezultatele simulării pot fi observate în figura 3.

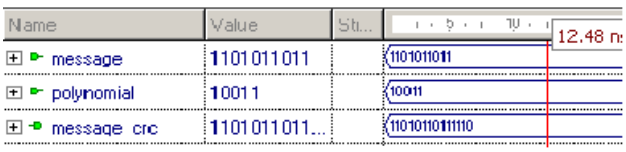
Implementarea bazată pe tabelă

Acest algoritm este o variantă îmbunătăţită a algoritmului anterior, el fiind foarte eficient deoarece implică doar o deplasare, o operaţie SAU, o operaţie SAU EXCLUSIV şi un acces la memorie pentru fiecare octet. Algoritmul precum şi rezultatele implementării sale în Verilog sunt prezentate în figura 4.

```
module crc_simple(message, polynomial, message_crc);
        /* Parametrii: */
        parameter crc_width = 4;  /* lungimea CRC-ului; */
        parameter msg_width = 10; /* lungimea mesajului; */
        /*Intrari, iesiri si resurse fizice interne*/
        /* -> masajul initial: */
        input  [msg_width - 1 : 0] message;
        wire   [msg_width - 1 : 0] message;
        /* -> polinomul generator: */
        input  [crc_width : 0] polynomial;
        wire   [crc_width : 0] polynomial;
        /* -> mesajul cu CRC-ul adaugat: */
        output [msg_width + crc_width - 1 : 0] message_crc;
        reg    [msg_width + crc_width - 1 : 0] message_crc;
        reg    [crc_width - 1 : 0] crc; /* the CRC register; */
        reg    msb_crc;/*cel mai semnificativ bit al registrului CRC; */
        /* resurse logice: */
        integer count; /* numarator; */
        /* Calculeaza CRC-ul si atasare: */
        always @(message | polynomial) begin
                /* Initializare: */
                message_crc = message;
                message_crc = message_crc << crc_width;
                crc = 'b0;
                /* Calculeaza CRC-ul: */
                for (count = msg_width + crc_width - 1; count >= 0; count = count - 1)
                 begin
                        msb_crc = crc[crc_width - 1];
                        crc = crc << 1;
                        crc[0] = message_crc[count];
                        if (msb_crc == 1) begin
                                crc = crc ^ polynomial;
                        end
                end
                /* Adauga CRC-ul mesajului: */
                message_crc[crc_width - 1 : 0] = crc;
        end
endmodule
```

Algoritm:

1. încarcă registrul cu biți 0
2. adaugă la sfârşitul mesajului W biți 0.
3. cât timp (mesajul mai are biți) deplasează registrul stânga cu un bit, introducând următorul bit din mesaj în poziția 0
4. dacă (un bit 1 a fost scos din registru) registru = registru XOR polinom
5. registrul conține restul



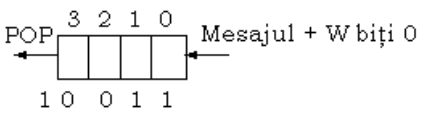


Figura 3: Implementare directă

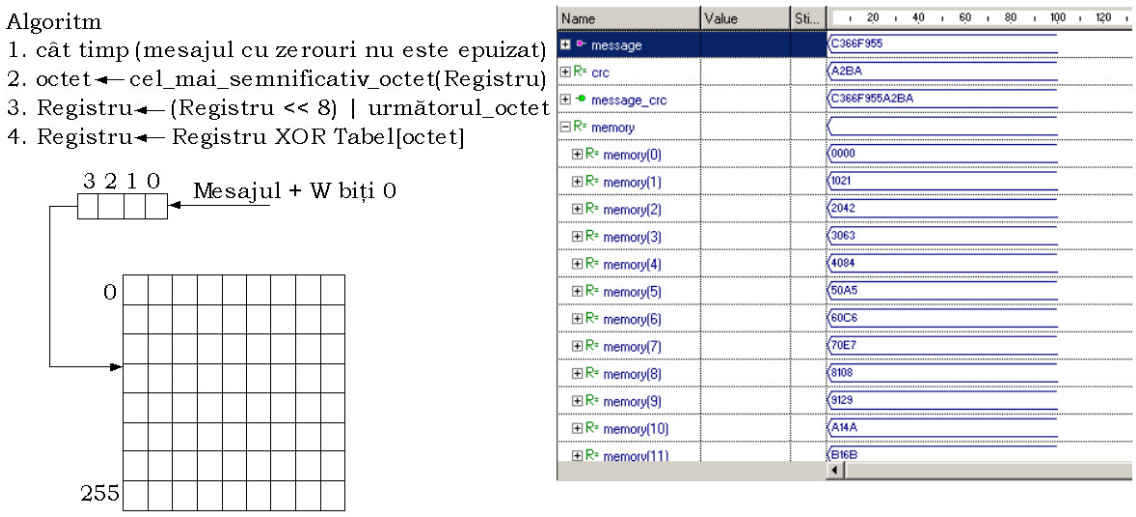


Figura 4: Implementarea bazată pe tabelă

Coduri Reed-Solomon

Codurile Reed-Solomon (RS) sunt coduri corectoare de erori în bloc inventate în 1960 de Irving Reed şi Gustave Solomon. Aceste coduri au început să fie utilizate începând cu 1990, atunci când progresele tehnologice au făcut posibilă trimiterea datelor în cantităţi mari si la viteze ridicate. Actualmente aceste coduri sunt utilizate într-o gamă largă de echipamente electronice cum sunt:

- dispozitivele pentru stocarea datelor (CD, DVD, hard-disk);
- telefoanele mobile;
- echipamentele folosite în comunicaţiile prin satelit;
- televiziunea digitală;
- modemurile de mare viteză (ADSL, xDSL).

Realizarea unei transmisii folosind codurile RS presupune ca, codificatorul RS să preia un bloc de date şi să adauge o informaţie suplimentară caracteristică. Una dintre caracteristicile importante ale codului RS constă în faptul că acest cod va codifica grupuri de simboluri de date.

Decodificatorul RS procesează fiecare bloc şi încearcă să corecteze erorile apărute şi să recupereze datele trimise original.

Un cod RS este specificat ca *RS(n, k)* cu simboluri de *s* biţi. Această descriere semnifică faptul că, codificatorul preia *k* simboluri de paritate astfel încât să rezulte un cuvânt de cod de *n* simboli. Sunt *n - k* simboluri de paritate, de câte *s* biţi fiecare. Un decodificator RS poate corecta până la *t* simboluri ce conţin erori, cu $2t = n - k$.

Un cod RS este obţinut împărţind mesajul original în blocuri de lungime fixă. Fiecare bloc este apoi împărţit în simboluri de *m* biţi. Fiecare simbol are lungime fixă (între 3 si 8 biţi). Natura liniară a acestui cod asigură faptul că fiecare cuvânt de *m* biţi este valid pentru codificare astfel încât se pot transmite date binare sau text.

Exemplu Un cod des folosit este RS(255, 233) cu simboluri de 8 biţi. Fiecare cuvânt de cod conţine 255 de simboluri din care 233 sunt de date şi 22 sunt de paritate. Pentru acest cod se pot stabili următoarele relaţii: $n = 255$, k — 233, $s = 8$, $t = 16$.

Codurile RS pot fi scurtate dacă la codificator se fac anumiţi biţi zero, nu se transmit dar sunt adăugaţi la decodificator. Spre exemplu, codul RS(255, 233) poate fi scurtat la (200, 168). Operaţiile realizate de codificator sunt următoarele:

- se preia un bloc de 168 de biţi de date;
- se adaugă virtual 55 de biţi de zero creând astfel un cod (255, 233);
- se transmit doar 168 biţi de date şi 32 biţi de paritate.

Un decodificator RS poate corecta un număr de *t* erori şi până la *2t* ştersături. La decodificarea unui cuvânt RS pot apărea următoarele variante:

- dacă $2s + r < 2t$ atunci codul original transmis poate fi corectat în întregime;
- decodificatorul indică faptul că nu poate reface codul original;
- decodificatorul va genera un cuvânt decodat cu erori şi nu va fi semnalat acest lucru.

Arhitectura decodorului poate fi urmărită în figura 5.

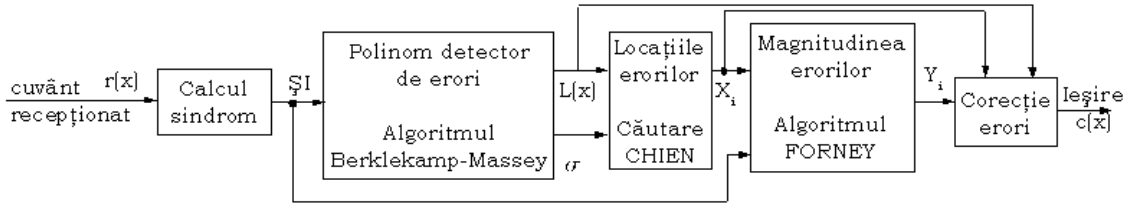


Figura 5: Arhitectura unui decodificator RS.

Algoritmul de criptare RSA

Algoritmul RSA este un sistem criptografic ce utilizează chei publice şi a fost creat de un grup de cercetători de la MIT (Massachusetts Institute of Technology) cu scopul de a asigura securitatea datelor schimbate prin intermediul Internet-ului.

Metodele tradiţionale de criptare (spre exemplu algoritmul DES - implementările hardware şi JAVA precum şi simulările acestor implementări pot fi vizualizate la http://www.csit-sun.pub.ro) folosesc un număr de $\frac{n \cdot (n-1)}{2}$ chei, în timp ce algoritmii bazaţi pe chei publice utilizează un număr de cel mult *n* chei publice.

O altă deosebire constă în faptul că în sistemele tradiţionale de criptare, cheia de criptare trebuie ţinută secretă deoarece ea trebuie utilizată în cadrul procesului de decriptare. În cazul criptării cu chei publice, cheia de criptare/decriptare nu mai este trimisă receptorului, deci canalul de comunicaţie dintre transmiţător şi receptor poate să nu fie securizat.

Utilizarea algoritmului RSA implică crearea a două chei de către transmiţător: *una publică* şi una *privată.* Cheia publică este trimisă oricărui destinatar la care trebuie trimis mesajul criptat. Cheia privată sau secretă este utilizată pentru decriptarea mesajului criptat cu ajutorul cheii publice.

Modalitatea de realizare a unei comunicaţii criptate cu ajutorul algoritmului RSA este prezentată în figura 6.

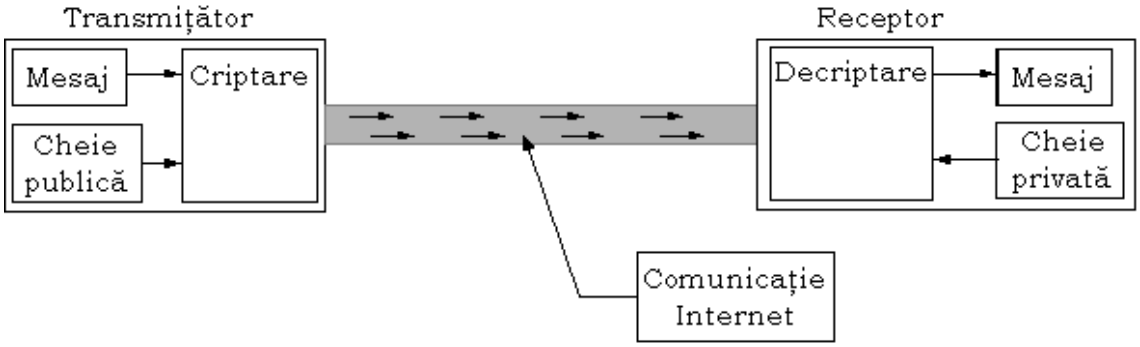


Figura 6: Arhitectura unui decodor RS.

Transmisia folosind algoritmul RSA necesită parcurgerea a două etape importante:

1. Generarea cheilor - se generează două chei una publică şi una privată. Pentru aceasta trebuie parcurşi următorii paşi:

1. se aleg două numere prime p şi q cu aceeaşi magnitudine (lungime) şi se generează numărul $n = p \cdot q$;

2. se determină $\Phi = (p-1) \cdot (q-1)$;

3. se alege e ca fiind un număr prim în raport cu Φ, deci cel mai mare divizor comun (notat gcd(e, Φ)) al celor două numere trebuie să fie 1. În implementările practice valoarea lui e este aleasă ca fiind un număr prim Fermat (3, 5, 17, 65537,...);

4. se determină valoarea d care reprezintă inversiunea modulară a lui e şi Φ:

$$d = rest\left(e - \frac{1}{\Phi}\right)$$

Cheia publică este alcătuită din perechea *(n, e),* cât timp cheia privată este formată din perechea *(n, d).* Implementarea hardware a celui mai mare divizor comun se realizează cu ajutorul algoritmului lui Euclid.

```
Algoritm EuclidExtins(a, b)
     if b = 0 then
          return (a, 1, 0)
     else
          (d', x', y') = EuclidExtins(b, rest(a/b))
     return (d', y', x' - a/b · y')
```

2. Transmisia informaţiei - În cadrul acestei etape, atât transmiţătorul cât şi receptorul trebuie să execute câteva operaţii distincte. Transmiţătorul realizează următoarele operaţii:
   a. obţine cheia publică *(n, e)* de la receptor;
   b. converteşte mesajul într-o mulţime de întregi pozitivi;
   c. calculează textul criptat conform relaţiei: $c = m^e \ mod$ n;
   d. transmite mesajul c la receptor.

   Receptorul realizează următoarele operaţii:
   a. utilizează cheia privată *(n, d)* pentru a calcula $m = c^d \ mod \ n$;
   b. extrage textul din colecţia de numere întregi *m.*

Algoritmul de criptarea IDEA

IDEA este un algoritm bazat pe chei publice care criptează blocuri de câte 64 de biţi folosind o cheie de criptare de lungime 128 de biţi. Criptarea şi decriptarea presupun utilizarea aceluiaşi algoritm. Implementarea acestui algoritm impune utilizarea a trei operaţii: *XOR, adunarea modulo 65536* şi *înmulţirea modulo 65537* care operează pe sub-blocuri de dimensiune 16 biţi.

Funcţionarea algoritmului constă în parcurgerea a opt paşi. Blocul de date de dimensiune 64 de biţi este împărţit în 4 părţi X_0, X_1, X_2 şi X_3, fiecare parte având dimensiunea de 16 biţi. În fiecare pas, între cele 4 sub-blocuri se realizează o operaţie *XOR,* de adunare sau de înmulţire, împreună cu 6 subchei de dimensiune 16 biţi fiecare.

Între paşii 2 şi 3, sub-blocurile sunt interschimbate, iar în final cele 4 sub-blocuri sunt combinate împreună cu 4 subchei pentru a forma ieşirea. În cadrul fiecărui pas al algoritmului se execută următoarea succesiune de operaţii:

- se înmulţeşte X_0 cu prima subcheie;
- se adună X_1 la a doua subcheie;
- se adună X_2 la a treia subcheie;
- se înmulţeşte X_3 cu a patra subcheie;
- XOR între rezultatele paşilor l şi 3;
- XOR între rezultatele paşilor 2 şi 4;
- se înmulţeşte rezultatul pasului 5 cu subcheia numărul 5;
- se adună rezultatele obţinute în cadrul paşilor 6 şi 7;
- se înmulţeşte rezultatul de la pasul 8 cu subcheia numărul 6;
- se adună rezultatele obţinute la paşii 7 şi 9;
- XOR între rezultatele paşilor l şi 9;
- XOR între rezultatele paşilor 3 şi 9;
- XOR între rezultatele paşilor 2 şi 10;

- XOR între rezultatele paşilor 4 şi 10.

Cele patru rezultate sunt sub-blocurile obţinute în urma paşilor 11, 12, 13 si 14. Se inter-schimbă cele două sub-blocuri din mijloc şi astfel se obţine intrarea pentru următorul pas. Excepţie face ultimul pas în care nu se mai execută interschimbarea celor două sub-blocuri din mijloc. După pasul opt se execută următoarea secvenţă de operaţii pentru a determina rezultatul final:

- se înmulţeşte X_0 cu prima subcheie;
- se adună X_1 la a doua subcheie;
- se adună X_2 la a treia subcheie;
- se înmulţeşte X_3 cu a patra subcheie.

În final cele patru sub-blocuri se vor concatena pentru a forma blocul criptat de lungime 64 de biţi.

Algoritmul utilizează 52 de subchei: 6 subchei pentru fiecare pas şi 4 subchei pentru pasul final. Generarea subcheilor porneşte de la cheia de lungime 128 de biţi care se împarte în opt subchei. Acestea reprezintă primele opt subchei utilizate în algoritm. La pasul următor cheia este deplasată la stânga 25 de poziţii şi apoi împărţită în opt părţi. Acest proces de generare a subcheilor este continuat până se generează toate cele 52 de subchei necesare funcţionării algoritmului.

Schema generală a algoritmului de criptare ***IDEA*** este prezentată în figura 7.

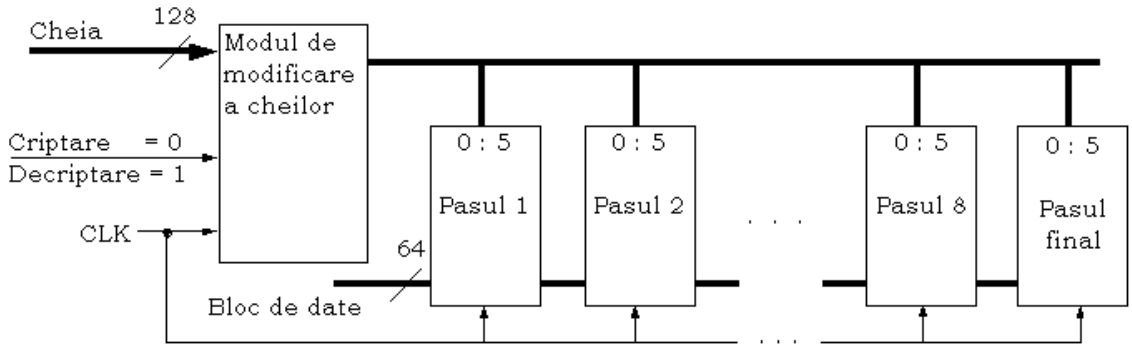


Figura 7: Schema generală a algoritmului de criptare cu chei publice IDEA.

2. Desfăşurarea lucrării

Se va proiecta în Verilog utilizând Xilinx WebPACK ISE 10.1 şi se va simula un circuit, care implementează algoritmul IDEA. Se va folosi schema generală prezentată în figura 7.

3. Probleme propuse

1. Să se proiecteze în Verilog utilizând Xilinx WebPACK ISE 10.1 şi să se simuleze un circuit, care implementează algoritmul CRC bazat pe tabelă.

2. Să se proiecteze în Verilog utilizând Xilinx WebPACK ISE 10.1 şi să se simuleze un circuit, care implementează algoritmul de criptare RSA.

Indicaţii

- Este bine să se calculeze o tabelă de conversie pentru fiecare dintre cele 256 valori de intrare posibile. Pentru a cripta mesajul va fi necesar doar accesul la o memorie locală care memorează tabela determinată. La decriptare se va utiliza acelaşi artificiu.

- Pentru a putea implementa în hardware expresia $m^e \ mod \ n$ se va utiliza următorul algoritm:

```
res = m;
for (i = 2; i<= e; i = i + 1) begin
  res = res * m;
  if (res > m) begin
        res = res % n;
  end
end
cypher(m, n, e) = res;
```

TEORIA INFORMAŢIEI ŞI A CODĂRII

Îndrumător de lucrări la laborator

Horia BALTA Maria KOVACI

2009

Cuprins

L1. Algoritmi pentru codarea sursei şi compresie

1.1. Codarea binară a surselor de informaţie

Lucrarea de faţă îşi propune explicitarea algoritmilor de calcul pentru a coda binar surse de informaţie întâlnite în practică.

Codarea binară a surselor de informaţie are dublu rol, de mărire a eficienţei sursei de informaţie (şi implicit de micşorare a costului transmisiei prin scăderea timpului transmisiei); de adaptare a sursei de informaţie la canalul transmisiune (canal binar, în majoritate).

Din punct de vedere al codării, la o sursă de informaţie interesează numărul de simboluri N şi probabilităţile lor de apariţie p_i, $i = \overline{1, N}$. Doar pe utilizator îl interesează ce reprezintă fiecare mesaj în parte (literă de alfabet, nivel al intensităţii pixelilor dintr-o imagine, valori ale unor mărimi fizice ce variază arbitrar[1]).

Codarea este operaţia prin care fiecărui simbol al sursei, s_i, i se alocă o succesiune binară unică:

$$s_i \longleftarrow r_i^1,\ r_i^2,\ \ldots r_i^{li} = c_i \qquad (1.1.1)$$

unde c_i reprezintă cuvântul de cod <u>asociat</u> simbolului (mesajului) sursei, iar r_i^j, cu $i = \overline{1, N}$, biţii componenţi ai cuvântului de cod (pot lua valori binare ”1” sau ”0”). Suportul fizic al valorilor binare este, de asemenea, mai puţin important pentru codare. Acest suport poate fi un semnal electric, optic, acustic, etc.

În expresia (1.1.1) l_i reprezintă lungimea cuvântului de cod c_i (număr de simboluri binare). Lungimea medie a cuvintelor de cod este dată de relaţia:

$$L = \sum_{i=1}^{N} p_i \cdot l_i \qquad (1.1.2)$$

Ieşirea codorului, pentru canalul de transmisie, reprezintă o sursă de informaţie numită secundară. Această sursă poate genera simbolurile 0 şi 1. Entropia (informaţia medie pe simbol), entropia maximă, precum şi eficienţa sursei primare, sunt date de relaţiile:

$$\begin{aligned} H(S) &= \sum_{i=1}^{N} p_i \cdot \log_2 \frac{1}{p_i}\ ; \\ H_{max}(S) &= \log_2 N\ ; \\ \eta_S &= \frac{H(S)}{H_{max}(S)}\ . \end{aligned} \qquad (1.1.3)$$

Entropia sursei secundare este egală cu informaţia medie pe un cuvânt raportată la lungimea medie a cuvântului:

$$H(X) = \frac{H(S)}{L} \qquad (1.1.4)$$

Atribuirea literelor de alfabet r_i^j pentru a forma cuvântul de cod c_i trebuie să conducă la îndeplinirea condiţiilor:

- codul să fie instantaneu (nici un cuvânt nu este prefixul altuia);

[1] Deşi, aparent, astfel de surse de informaţie nu au un număr finit de simboluri, prin eşantionare şi cuantizare, orice sursă de informaţie poate fi redusă la una cu număr finit de simboluri.

- codarea să conducă la o eficienţă maxim posibilă. Această condiţie se realizează folosind cuvinte de lungime variabilă, mai mare pentru simboluri de probabilitate de emisie (a sursei primare) mai mică. Codarea cu cuvinte de lungime variabilă este mai complicată, dar conduce la o sursă secundară cu eficienţă mai bună şi la o ieftinire a transmisiei, lungimea medie a cuvintelor de cod fiind mai mică. Codarea cu cuvinte de lungime constantă este mai simplă dar cuvintele având aceeaşi lungime oferă maleabilitate la prelucrări ulterioare.

Codarea cu cuvinte de lungime constantă se mai numeşte şi codare cu cod bloc. Cuvintele codului bloc sunt secvenţe binare diferite, de aceeaşi lungime L. Ştiind că numărul de secvenţe diferite ce pot fi constituite cu L cifre binare este 2^L, rezultă că numărul simbolurilor sursei trebuie să fie mai mic decât 2^L, sau:

$$L \geq \log_2 N = H_{max}(S) > L - 1 \qquad (1.1.5)$$

Inecuaţia a doua din (1.1.5) se datorează criteriului de eficienţă minim posibilă (sub restricţia de cod bloc).

Realizarea condiţiilor mai sus precizate se face prin codare după algoritmul Huffman (static). Acesta presupune:

1. Ordonarea probabilităţilor sursei (primare) în ordine descrescătoare;
2. Simbolurile cu probabilităţile cele mai mici (ultimele două[1] în ordonare) se reunesc formând un nou simbol cu probabilitatea sumei celor două, după care se reordonează probabilităţile, obţinându-se o sursă cu N-1 simboluri;
3. Procesul continuă până când rămân 2 simboluri. Acestora li se atribuie valorile 0 şi 1;
4. Mergând pe cale inversă, se disociază simbolurile compuse în simboluri din care s-au compus, atribuind arbitrar, la fiecare disociere, valorile 0 şi 1 pentru a găsi cuvintele de cod pentru cei doi componenţi. De reţinut că doar un singur simbol se disociază la fiecare pas, lungimea componenţilor săi crescând cu 1, celelalte simboluri păstrându-şi lungimea şi structura cuvântului de cod.

***Observaţie**: -atribuirea simbolurilor binare celor două simboluri compuse este arbitrară, codurile obţinute vor fi diferite, dar de aceeaşi eficienţă. De asemenea, ordinea simbolurilor de egală probabilitate poate fi aleasă arbitrar, diferitele moduri de atribuire conducând la coduri diferite, dar de aceeaşi eficienţă. Însă, în vederea posibilităţii de a confrunta rezultatele, se vor respecta regulile: -„0" se atribuie totdeauna simbolului de jos (ultimul în ordonare); -ordinea simbolurilor egal probabile este ordinea în care s-au citit, cu simbolul compus totdeauna ultimul.*

Programul pe calculator, asociat acestei lucrări, permite codarea surselor de informaţie prin algoritm Huffman static.

Precizarea sursei de informaţie se poate face simplu, introducând valorile probabilităţilor sursei şi numărul lor. Probabilităţile trebuie să îndeplinească condiţia:

$$\sum_{i=1}^{N} p_i = 1 \quad ; \quad 0 \leq p_i \leq 1 ; \quad \forall\ i = \overline{1, N} \qquad (1.1.6)$$

Probabilităţile pot fi introduse indirect, prin numere întregi, pozitive k_i, calculatorul urmând să facă transformarea:

$$p_i = \frac{k_i}{\sum_{k=1}^{N} k_i} \qquad (1.1.7)$$

[1] Algoritmul se poate aplica şi pentru obţinerea codurilor nebinare (având un alfabet cu m simboluri). În acest caz se grupează câte m simboluri.

ceea ce asigură îndeplinirea relaţiei (1.1.6).

Sursa de informaţie poate fi şi un text scris (în caractere ASCII), cu cel puţin 3 caractere diferite. Calculatorul va înţelege că simbolurile sursei (caracterele prezente în text) au probabilităţi precizate prin ponderea lor în text (număr de prezenţe ale caracterului respectiv raportat la întregul număr de caractere ale textului).

Programul permite şi o a treia sursă de informaţie. Simbolurile acesteia reprezintă intervale de lungime egală, şi în număr finit, de pe axă reală. În acest caz datele vor consta în valorile eşantioanelor unui semnal discret (Fig.1.1.1) , mărginit în timp cu suportul finit precizat ca dată de intrare M. Tot ca dată de intrare calculatorul va cere şi q-cuanta. Având aceste mărimi q (cuanta), M (numărul de eşantioane) şi b(i) (valorile eşantioanelor), pentru găsirea simbolurilor sursei şi implicit a numărului lor – N, calculatorul va proceda astfel:

- va căuta eşantioanele de valoare minimă şi maximă : min; max;
- va calcula $N = [\frac{\max - \min}{2}] + 1$ unde [ ] semnifică partea întreagă;

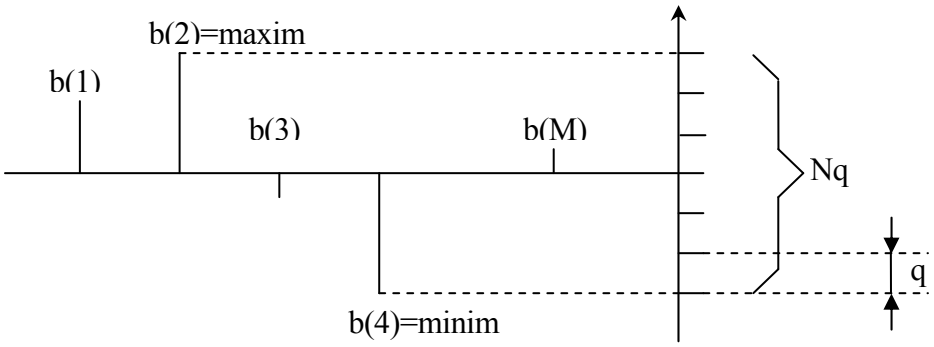


Fig.1.1.1 Sursă de informaţie „semnal discret”

- va calcula $N = [\frac{\max - \min}{q}] + 1$ unde [ ] semnifică partea întreagă;
- va atribui sursei de informaţie primare simbolurile: [min, min + q); [min+q, min+2q); ... ;[min+(N-1)q,min+Nq)
- va calcula probabilitatea simbolurilor ca fiind egale cu fracţiunea din numărul eşantioanelor M, ce au valori cuprinse în intervalele ce definesc simbolurile. Este posibil desigur să rămână simboluri cu probabilitate 0.

În toate cele 3 cazuri se vor calcula şi se vor putea vizualiza:

a) probabilităţile sursei primare;
b) datele asociate sursei şi codării:
   - entropia sursei primare H(S);
   - entropia maximă a sursei primare $H_{max}(S)$;
   - eficienţa sursei primare η_S;
   - entropia sursei secundare (eficienţa codării) H(X);
   - lungimea medie a cuvintelor de cod L;
   - redundanţa sursei primare $R(S)=H_{max}(S)-H(S)$;
   - redundanţa relativă a sursei primare $\rho_S = 1-\eta_S$;
   - redundanţa sursei secundare $\rho_X = 1-H(X)$.
c) cuvintele de cod;
d) graful codării;

e) schema algoritmului Huffman (numai pentru cazul codării cu cuvinte de lungime variabilă);
f) în cazul sursei „semnal discret" se pot vedea şi graficul semnalului precum şi valorile eşantioanelor.

Desfăşurarea lucrării

a). Rulaţi programul cobsinf.exe selectând opţiunea „Demonstraţie". Selectaţi pe rând cele trei feluri de surse de informaţie (tablou, text şi semnal discret), şi pentru fiecare sursă solicitaţi o codare bloc şi o codare prin algoritmul Huffman static. Urmăriţi aplicarea algoritmului.
b). Alcătuiţi surse de informaţie de forma celor prezentate şi codaţi-le bloc şi prin algoritmul Huffman static, calculând în fiecare caz entropia sursei, eficienţa sursei, lungimea medie a cuvintelor codului obţinut, precum şi eficienţa codării. Porniţi pentru început cu surse tablou mai simple, apoi măriţi numărul de simboluri. Utilizaţi ulterior şi surse „text" sau „semnal discret".
c) La sfârşitul lucrării de laborator se va efectua test asupra cunoştinţelor acumulate. Durata acestuia este aproximativ constantă şi independentă de numărul de studenţi ce efectuează simultan testul (calculatorul poate testa până la 4 studenţi simultan – în sensul că afişează pe rând date pentru cei $n \leq 4$ studenţi şi apoi întreabă pe rând, în aceeaşi ordine studenţii, timpii de afişare şi de chestionare rezervaţi fiecărui student sunt constanţi şi nu influenţează pe cei rezervaţi altuia). Rezultatul testului se va concretiza printr-o notă, calculată automat de calculator.

1.2. Algoritmul Huffman dinamic

Compresia este procesul de minimizare a spaţiului ocupat sau a timpului necesar transmiterii unei anumite cantităţi de informaţie.

Metodele de compresie pot fi împărţite în:

- Metode de compresie cu pierdere;
- Metode de compresie fără pierdere.

Metodele de compresie cu pierdere de informaţie sunt folosite în special în transmiterea semnalului audio şi video, unde pierderea de informaţie are ca rezultat o scădere a calităţii sunetului, respectiv imaginii.

Compresia de date fără pierdere, prezentă în programele de arhivare, în sistemele de transmisiune a datelor, a evoluat de-a lungul timpului pornind de la algoritmi simpli (suprimarea zerourilor, codarea pe şiruri) şi ajungând la algoritmii complecşi folosiţi în prezent.

Metodele de compresie fără pierderi au la bază ideea că în general cantitatea de informaţie prelucrată (transmisă, depozitată) conţine o anumită redundanţă ce se datorează:

- Distribuţiei caracterelor (unele caractere au o frecvenţă de apariţie mult mai mare decât altele);
- Repetării consecutive a unor caractere;
- Distribuţiei grupurilor de caractere (unele grupuri sunt mult mai frecvente decât altele şi în plus există grupuri care nu apar deloc);
- Distribuţiei poziţiei (unele caractere sau grupuri ocupă poziţii preferenţiale, predictibile în anumite blocuri de date).

Având în vedere toate aceste tipuri se poate înţelege de ce o anumită tehnică de compresie poate da un rezultat bun pentru un anumit tip de surse, pentru altele însă rezultatul fiind dezastruos. În studiul compresiei se urmăreşte obţinerea unui algoritm care să ofere o compresie cât mai bună pentru tipuri de surse cât mai diferite.

Aprecierea cantitativă a compresiei realizate se face utilizând *factorul de compresie*, definit ca:

$$F_c = \frac{n_u}{n_c} \qquad (1.2.1)$$

unde n_u este lungimea în biţi a mesajului iniţial şi n_c lungimea de compresie.

Algoritmul Huffman dinamic

Algoritmii de tip Huffman static au dezavantajul că necesită cunoaşterea prealabilă a statisticii sursei. Acest dezavantaj poate fi înlăturat utilizând un algoritm dinamic.

Algoritmul Huffman dinamic este un algoritm de compresie. Mesajele au fost deja codate în prealabil, dar neeficient, cu un cod bloc, de lungime *n* biţi/cuvânt. Ideea de bază în această codare este folosirea pentru codarea unui simbol s_{i+1} din mesaj a unui graf de codare, ce este un arbore care creşte, dintr-un punct iniţial (numit sursă) şi care este construit pe baza primilor *i* simboluri din mesaj. După transmiterea simbolului s_{i+1} se va revizui arborele de codare în vederea codării simbolului s_{i+2}.

Codarea presupune la fiecare pas transmiterea codului aferent simbolului de intrare şi modificarea corespunzătoare a grafului şi a codului. Dacă în mesajul transmis este un simbol nou, adică un simbol care n-a mai apărut în mesaj, se transmite mai întâi codul frunză goală şi apoi cei *n* biţi aferenţi simbolului nou apărut. Frunza goală are semnificaţia că urmează un nou simbol. Frunza goală se codifică ca un mesaj oarecare, dar ponderea sa întodeauna va rămâne nulă.

Exemplu:
În continuare se prezintă evoluţia arborelui (grafului) asociat codului în timpul codării mesajului "M_i = aaabccc":

Obs: *Semnificaţia notaţiilor este:*

-pentru nod: x | p |

- x: număr de ordine (creşte de la stânga spre dreapta şi de jos în sus pe linii);
- p: ponderea cumulată din nodurile aferente.

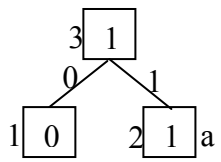
- pentru frunză (nod terminal): x | p | y

- x: număr de ordine;
- p: pondere (număr de apariţii ale respectivului simbol);
- y: simbolul.

Frunza goală, notată cu "0", este de pondere 0.

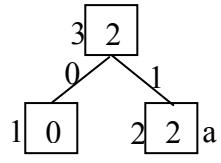
- pentru ramură: - spre stânga ≡ codare cu zero;
- spre dreapta ≡ codare (atribuire) cu unu.

1. M_i = "a"



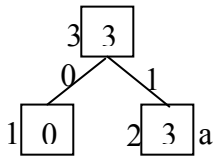
Codurile sunt: 0 - 0
a - 1
Mesajul transmis este: "a"

2. M_i = "aa"



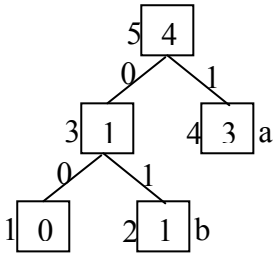
Codurile sunt: 0 - 0
a - 1
Mesajul transmis este: "a1"

3. M_i = “aaa”



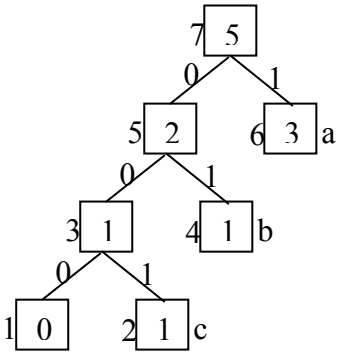
Codurile sunt: 0 - 0
a - 1
Mesajul transmis este: “a11”

4. M_i = “aaab”



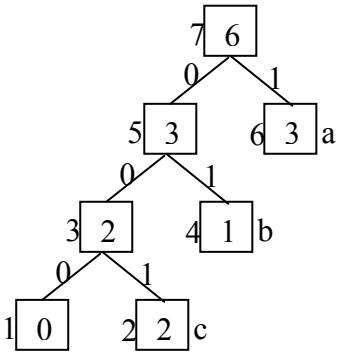
Codurile sunt: 0 - 00
a - 1
b - 01
Mesajul transmis este: “a110b”

5. M_i = “aaabc”



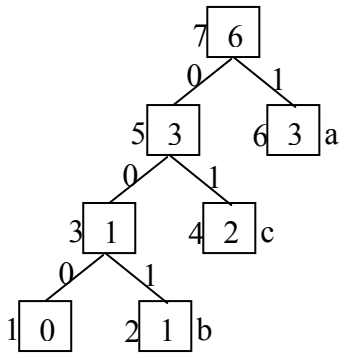
Codurile sunt: 0 - 000
a - 1
b - 01
c - 001
Mesajul transmis este: “a110b00c”

6. M_i = “aaabcc”



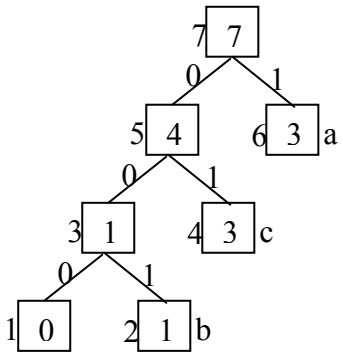
Codurile sunt: 0 - 000
a - 1
b - 01
c - 001
Mesajul transmis este: “a110b00c001”

Deoarece ponderea nodului “c” este mai mare decât ponderea nodului “b” se va face interschimbarea între noduri, realizându-se o rearanjare a arborelui:



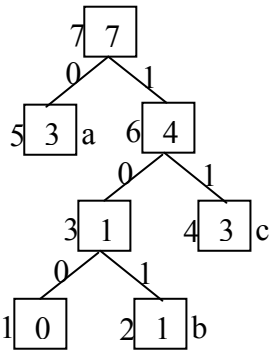
Codurile sunt: 0 - 000
a - 1
b - 001
c - 01
Mesajul transmis este: “a110b00c001”

7. M_i = “aaabccc”



Codurile sunt: 0 - 000
a - 1
b - 001
c - 01
Mesajul transmis este: “a110b00c00101”

În urma rearanjării arborelui se obţine:



Codurile sunt: 0 - 100
a - 0
b - 101
c - 11
Mesajul transmis este: “a110b00c00101”

Dacă se presupune că simbolurile mesajului de la intrare au fost codate în prealabil cu un cod bloc, cu n=8 biţi/cuvânt vom avea lungimea mesajului iniţial:
Considerând caracterele în mesajul iniţial codate pe 8 biţi, lungimea acestuia este:

$$n_u = 7 \cdot 8 = 56 \text{ biţi} \qquad (1.2.2)$$

Mesajul comprimat (a110b00c00101) are:

$$n_c = 8 \cdot 3 + 10 = 34 \text{ biţi} \qquad (1.2.3)$$

Rezultă un factor de compresie:

$$F = \frac{n_u}{n_c} \cong 1,7$$

Desfăşurarea lucrării:

a) Se lansează executabilul dHuffman.exe din directorul Huffman Dinamic, după care se introduce parola TTI. Rulaţi programul, selectând opţiunea Demonstraţie, pentru codare şi decodare.
b) Se verifică, cu programul, exemplul luat în această lucrare, selectând pe rând compresia, respectiv decompresia prezentate.
c) Pentru compresie şi decompresie se alege câte un exemplu oarecare asemănător cu cel prezentat în lucrare. Se realizează compresia/decompresia, după care, cu programul, se verifică rezultatele obţinute.
d) La sfârşitul lucrării de laborator se va efectua un test asupra cunoştinţelor acumulate. Rezultatul testului se va concretiza printr-o notă, calculată automat de calculator.

1.3. Algoritmi de compresie de tip LZ

Cu toate că algoritmii de compresie Huffman, static sau dinamic, sunt cei mai performanţi (din punct de vedere al factorului de compresie) relativ la sursele fără memorie, această concluzie nu este valabilă şi pentru sursele cu memorie, surse frecvent întâlnite în practică.

Algoritmii de compresie de tip LZ fac parte din categoria tehnicilor de dicţionar adaptive. Ele servesc (în special) la compresia fişierelor de tip text, fişiere ce se caracterizează prin repetarea frecventă a unor subşiruri.

Ideea de bază în algoritmii LZ este înlocuirea unor subşiruri ale mesajului cu cuvinte de cod, astfel încât, la o nouă apariţie a unui subşir să se transmită doar codul asociat lui.

Algoritmul LZ –77

Mesajul de intrare este trecut printr-un buffer de lungime N. Bufferul este divizat în două blocuri distincte numite Lempel şi Ziv, ca în Fig. 1.3.1.

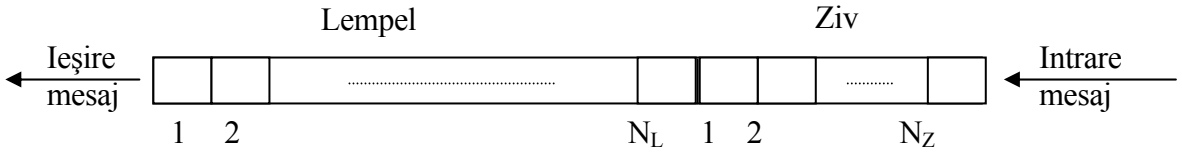


Fig. 1.3.1. Buffer–fereastră utilizat în algoritmul LZ–77

Mesajul de ieşire constă într-o succesiune de triplete de forma „P, L, A”. Compresia decurge astfel:

-se încarcă primele N_Z caractere din mesajul de intrare în blocul Ziv;

-se transmite la ieşire tripletul „0, 0, x”, unde x reprezintă primul caracter (literă) din mesajul de intrare (aflat pe poziţia 1 în blocul Ziv);

se deplasează cu o poziţie mesajul de intrare în blocul Ziv, astfel încât x ajunge acum în poziţia N_L. Din acest moment:

-se caută în blocul Lempel un subşir, S', care este identic cu cel mai lung subşir, S, care începe în poziţia 1 a blocului Ziv şi este conţinut în întregime în blocul Ziv. Subşirul S' trebuie neapărat să înceapă în blocul Lempel, dar poate să sew sfârşească în blocul Ziv. Dacă un astfel de subşir, S', există, atunci se va transmite tripletul „P, L, A”, unde P = poziţia de la care începe subşirul S'; L = lungimea subşirurilor S şi S'; A = litera ce succede subşirul S în mesajul de intrare.

Dacă nu există nici un subşir S' care să coincidă cu cel mai scurt subşir S, atunci se va transmite tripletul „0, 0, x”, unde x are aceeaşi semnificaţie ca şi mai sus;

-se deplasează mesajul de intrare până când, după caz, A sau x ajung pe poziţia N_L şi procesul de căutare se reia până când tot mesajul a trecut prin blocul Ziv.

Decompresia presupune, în principal, aceleaşi etape. Utilizând tripleţii recepţionaţi, se adaugă subşiruri în blocul Ziv mesajului deja decomprimat, subşiruri aflate deja în blocul Lempel urmând ca, mai apoi, toate să fie deplasate în blocul Lempel.

Observaţie: *este posibil să se renunţe la cel de-al doilea zero din tripletul „0, 0, x", el fiind redundant.*

Cei trei componenţi ai tripletului „P, L, A" sunt codaţi binar bloc, separat. Astfel, codul pentru P are lungimea:

$$k_P = \log_2 (N_L + 1) \qquad (1.3.1)$$

(trebuie codat şi 0 din situaţia „0, 0, x" sau „0,x"), iar pentru L:

$$k_L = \log_2 (N_Z - 1) \qquad (1.3.2)$$

considerând doar varianta „0, x". Caracterele de tipul „A" pot fi codate, de exemplu, ASCII.

Din relaţiile (1.3.1) şi (1.3.2) rezultă că, pentru o bună compresie, N_L+1 şi N_Z-1 trebuie să fie puteri întregi ale lui 2. În relaţia (1.3.2) s-a luat N_Z-1 şi nu doar N_Z deoarece, în situaţia extremă a lungimii maxime pentru subşirul S, N_Z va fi egal cu lungimea subşirului S plus unu; acest „unu" rezervă literei A un loc în blocul Ziv.

Exemplu *Fie parametrii N_L =7 şi N_Z =4, iar mesajul de intrare „aaababacacab…". Aplicarea algoritmului LZ –77 asupra acestui mesaj de intrare este ilustrată în Fig. 1.3.2 a).*
Mesajul comprimat este, aşadar: „0a72b63c73b…", sau în binar:
„000a11110b11011c11111b…". Fig. 1.3.2. b) prezintă un exemplu de decompresie.

1	2	3	4	5	6	7	1	2	3	4	P	L	A
							a	a	a	b	0		a
						a	a	a	b	a	7	2	b
			a	a	a	b	a	b	a	c	6	3	c
a	a	b	a	b	a	c	a	c	a	b	7	3	b

a) compresia mesajului „aaababacacab…"

1	2	3	4	5	6	7	1	2	3	4	P	L	A
							a				0		a
						a	a	a	a	b	7	3	b
		a	a	a	a	b	a	a	c		3	2	c
a	a	a	b	a	a	c	b	a	c		4	2	c
b	a	a	c	b	a	c	a	c	b	c	3	3	c

b) decompresia mesajului „0aa73b32c42c33c"

Fig. 1.3.2. Exemplu de aplicare a algoritmul LZ–77

Algoritmul LZ–78

Spre deosebire de varianta anterioară, la LZ–78 blocul Lempel este un dicţionar în continuă creştere. De asemenea, nu există teoretic nici o limitare a blocului Ziv.

Algoritmul LZ–78 este, în esenţă, următorul:

-se iniţializează dicţionarul (blocul Lempel) cu un şir nul;

-se încarcă mesajul de intrare în blocul Ziv;

-se transmite în clar primul caracter şi se introduce şi în dicţionar la poziţia 1. Cele două poziţii ale dicţionarului (notate cu 0 şi 1) sunt, în acest moment codate prin „0" şi „1". Din acest moment:

-se caută în dicţionar un subşir, S', identic cu cel mai lung subşir, S, din blocul Ziv. Binenţeles, S începe din prima poziţie a blocului Ziv. Dacă există un astfel de subşir, S', atunci se transmite codul

aferent urmat de caracterul A ce urmează lui S în mesajul de intrare. În plus, dicţionarul se îmbogăţeşte cu o poziţie, poziţie unde se trece subşirul SA. Dacă nici măcar primul caracter, x, din Ziv nu se regăseşte ăn dicţionar, atunci se transmite 0x, unde 0 semnifică subşirul din prima poziţie, codat prin cuvântul de cod „00…0”.

Observaţie: *Atunci când dicţionarul ajunge la un număr de elemente egal cu 2^k, tuturor codurilor elementelor anterioare li se adaugă ca şi prefix „0”, iar ultimul subşir va fi codat prin „10…0”.*

Exemplu: *În Fig. 1.3.3 a) se prezintă un exemplu de compresie utilizând algoritmul LZ–78. Mesajului de intrare este cel din exemplul anterior, „aaababacacab…”. Mesajul comprimat rezultat este „a1a0b1b1c6a3”, sau în binar „a1a0b01c110a011”. În Fig. 1.3.3 b) se prezintă un exemplu de decompresie, utilizând acelaşi algoritm. Mesajul comprimat (de intrare) este „a1b3b5a3a”. Rezultatul decompresiei este „aababcabcaaba”.*

Blocul Lempel (dicţionarul)

0	
1	a
2	aa
3	b
4	ab
5	c
6	ac
7	aca
8	

Blocul Ziv	Mesajul de ieşire
a a a b a b a c a c a b	a
a a b a b a c a c a b	1a
b a b a c a c a b	0b
a b a c a c a b	1b
a c a c a b	1c
a c a b	6a
b	3

b) compresia mesajului „aaababacacab…”;

Blocul Lempel (dicţionarul)

0	
1	a
2	b
3	ab
4	c
5	abc
6	abca
7	

Mesaj decomprimat	Mesaj comprimat
a	a
a a b	1b
a a b a b c	3b
a a b a b c a b c a	5a
a a b a b c a b c a a b a3a	

b) decompresia mesajului „a1b3b5a3a”

Fig. 1.3.3 Exemplu de aplicare a algoritmul LZ–78.

Decompresia presupune următorul algoritm:
-se citeşte primul caracter (codul ASCII aferent); acest caracter se introduce în dicţionar la poziţia 1 şi se generează şi la ieşire; în continuare procedura este:
-se citeşta codul poziţiei la care se găseşte subşirul transmis. Acest cod conţine:

$$k = \sup(\log_2 n) \quad \text{biţi} \tag{1.3.3}$$

unde sup(x) reprezintă aproximarea prin adaos a lui x, iar n dimensiunea momentană a dicţionarului;
-se citeşte din dicţionar subşirul S aflat la poziţia citită anterior şi se generează la ieşire;

-se citeşte din mesajul recepţionat următorul caracter, A (în cod ASCII), şi se generează şi acesta la ieşire. Dacă A este un nou caracter, atunci se introduce în dicţionar la o nouă poziţie;
-se introduce în dicţionar subşirul SA la următoarea poziţie;
Procesul se repetă până la epuizarea mesajului recepţionat.

Algoritmul LZW (Lempel–Ziv–Welch)

Modificarea esenţială adusă de algoritmul LZW este faptul că atât compresorul cât şi decompresorul cunosc simbolurile (caracterele) componente ale mesajului. Cu alte cuvinte, înainte de începerea propriu-zisă a compresiei (decompresiei) dicţionarul este iniţializat cu toate caracterele posibil a fi emise.(În exemplul următor, ilustrat în Figura 3.4a, dicţionarul este iniţializat cu simbolurile a–00, b–01, c–10.) În acest fel nu se mai trimite informaţie despre următorul caracter la fiecare pas. O altă modificare o constituie existenţa unui prefix, P, care se memorează de la un pas la următorul.

Algoritmul compresiei este:
-se iniţializează dicţionarul (aşa cum s-a descris anterior);
-se citeşte primul caracter şi se memorează ca şi prefix, P. Din acest moment:
-se citeşte următorul caracter, notat E. Se caută în dicţionar subşirul PE. Dacă acest subşir există, atunci se trece la pasul următor, fără a emite nimic şi fără a adăuga nimic în dicţionar. Singura modificare este că acum PE devine prefix pentru pasul următor. Dacă subşirul PE nu există în dicţionar, atunci se execută următoarele operaţii:
-se trimite la ieşire codul aferent subşirului P;
-se trece pe post de prefix caracterul E (E→P);
-se adaugă în dicţionar subşirul PE la următoarea poziţie liberă.
Procesul continuă în acest fel până la epuizarea întregului mesaj de intrare.

Observaţie: *La fel ca şi la LZ-78, pe vreme ce dicţionarul creşte trebuie modificat corespunzător codul binar loc asociat.*

Algoritmul decompresiei este asemănător compresiei. Diferenţa este că simbolul E nu se citeşte direct de la intrare (ca şi la compresie) ci doar după ce s-a decodat codul recepţionat. Mesajul decomprimat se poate citi fie de la E (exceptând primul simbol, care se citeşte de la P), fie prin compunerea şirului decodat.

Exemplu: *Fig. 1.3.4. prezintă câte un exemplu de compresie şi decompresie utilizând algoritmul LZW. Pentru compresie s-a ales ca mesaj de intrare: „aaababacacab”, acelaşi ca şi în exemplele anterioare. Mesajul comprimat este: „031052081”, sau în binar „001100100010101000001000001”. Pentru decompresie s-a ales mesajul „013205”, car în binar se exprimă: „0001011010000101”. Mesajul decomprimat este: „ababcaabc”.*

<u>Desfăşurarea lucrării:</u>

a) Rulaţi programul pe calculator, utilizând opţiunea demonstrativă, urmărind aplicarea celor trei algoritmi la compresie şi decompresie. Aflaţi pentru fiecare exemplu şi factorul de compresie.
b). Alcătuiţi, utilizând trei sau patru litere, mesaje de circa 10 litere lungime. Comprimaţi mesajele independent de program şi verificaţi rezultatele cu ajutorul programului. Pentru decompresie utilizaţi mesajele comprimate de colegi, fără a cunoaşte originalul. Confruntaţi rezultatele decomprimării cu cele originale. Calculaţi în fiecare caz factorul de compresie, considerând că orice literă (caracter) se scrie în binar pe 8 biţi.
c) La sfârşitul lucrării de laborator se va efectua test asupra cunoştinţelor acumulate, prin intermediul programului pe calculator. Testul constă din: 1–o compresie şi 2–o decompresie, a unui mesaj ales aleatoriu de către calculator printr-unul dintre cei trei algoritmi (de asemenea ales aleatoriu de program) şi 3 –cinci întrebări teoretice fiecare cu un răspuns corect din cinci propuse, având ca temă algoritmii de compresie.

Prefix	Mesaj de intrare	Şir căutat în dicţionar	Şir transmis	Cod transmis	Dicţionarul a 0=00 b 1=01 c 2=10	
P	E	PE			Şir adăugat în dicţionar	Cod adăugat în dicţionar
	a					
a	a	aa	a	0=00	aa	3=11
a	a	aa				
aa	b	aab	aa	3=11	aab	4=100
b	a	ba	b	1=001	ba	5=101
a	b	ab	a	0=000	ab	6=110
b	a	ba				
ba	c	bac	ba	5=101	bac	7=111
c	a	ca	c	2=010	ca	8=1000
a	c	ac	a	0=0000	ac	9=1001
c	a	ca				
ca	b	cab	ca	8=1000	cab	10=1010
b		b	b	1=0001		

a) compresia mesajului „aaababacacab”;

Prefix	Mesaj de intrare	Şir căutat în dicţionar	Şir recepţionat decodat	Cod recepţionat	Dicţionarul a 0=00 b 1=01 c 2=10	
P	E	PE			Şir adăugat în dicţionar	Cod adăugat în dicţionar
			a	0		
a	b	ab	b	1	ab	3=11
b	a	ba	ab	3	ba	4=100
a	b	ab				
ab	c	abc	c	2	abc	5=101
c	a	ca	a	0	ca	6=110
a	a	aa	abc	5	aa	7=111
a	b	ab				
ab	c	abc				

b) decompresia mesajului „013205”;
Fig. 1.3.4. Exemplu de aplicare a algoritmului LZW

L2. Coduri simple corectoare de o eroare

2.1. Codul Hamming

2.1.1. Cod Hamming corector de o eroare

Codurile Hamming constiruie prima clasă de coduri bloc liniare corectoare de erori şi au fost propuse de R. Hamming în 1950.

Codul Hamming este un cod protector. Scopul codării este acela de a adapta sursa de informaţie la canalul de transmisie. În cazul de faţă sursa este deja codată şi se face o codare pentru protecţia informaţiei. Acest lucru se realizează prin mărirea redundanţei (prin creşterea suportului binar; biţilor de informaţie adăugânduse biţii de control). Biţii de control au ca scop de a crea legături, relaţii între biţii de informaţie. Aceste relaţii folosesc codorului pentru a calcula biţii de control şi folosesc decodorului pentru a verifica corectitudinea transmisiei.

Codul Hamming este un cod nesistematic (simbolurile de control sunt intercalate printre simbolurile informaţionale, situându-se pe poziţii puteri ale lui 2).

2.1.1.1. Codarea codului Hamminmg corector de o eroare

Particularitatea codului liniar Hamming corector de o eroare constă în forma matricii de control, H, care are fiecare coloană structurată prin reprezentarea binară a numărului zecimal de ordine al coloanei respective. Din acest motiv, corectorul Z, calculat cu relaţia $Z = H \cdot V^T$, decodat din binar în zecimal, indică numărul de ordine zecimal al bitului eronat din cuvântul recepţionat.

Distanţa minimă a acestui cod este:

$$d \geq 2e_c + 1 = 3,$$

unde e_c reprezintă numărul de erori corectabile.

Se consideră codul cu parametrii:

- n=7, numărul de simboluri în cuvânt;
- m=3, numărul de simboluri de control în cuvânt;
- k=4, numărul de simboluri de informaţie în cuvânt.

Secvenţa de informaţie este:

$$i = i_3 i_5 i_6 i_7$$

şi secvenţa de control:

$$C = C_1 C_2 C_4$$

astfel încât cuvântul de cod rezultat va fi:

$$V = C_1 C_2 i_3 C_4 i_5 i_6 i_7$$

Codarea înseamnă calculul simbolurilor de control C_1, C_2, şi C_4 când se dau simbolurile de informaţie i_3, i_5, i_6, i_7.

Relaţia de codare:

$$H \cdot V^T = 0 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ i_3 \\ C_4 \\ i_5 \\ i_6 \\ i_7 \end{bmatrix} = 0 \qquad (2.1.1)$$

unde *H* reprezintă matricea de control, ce este de dimensiune *m×n*.

Sau în formă explicită:

$$\begin{aligned} C_1 &= i_3 + i_5 + i_7 \\ C_2 &= i_3 + i_6 + i_7 \\ C_4 &= i_5 + i_6 + i_7 \end{aligned} \qquad (2.1.2)$$

Suma făcându-se modulo doi.

Exemplul 1:

Pentru n=7, k=4, m=3 şi secvenţa de informaţie i=1010, găsiţi cuvântul V de cod.

Rezolvare:

Se observă că există 4 simboluri de informaţie: i_3, i_5, i_6, i_7. Cuvântul de cod,V, se poate scrie sub formă literară: $V = C_1C_2i_3C_4i_5i_6i_7$. Avem aşadar 7 simboluri ce alcătuiesc cuvântul de cod şi 3 biţi de control. Rezultă că matricea de control, H, va conţine 3 linii şi 7 coloane.
Din relaţia de codare (2.1.1), rezultă relaţiile de calcul a biţilor de control:

$$\begin{aligned} C_1 &= i_3 + i_5 + i_7 = 1+0+0 = 1 \\ C_2 &= i_3 + i_6 + i_7 = 1+1+0 = 0 \\ C_4 &= i_5 + i_6 + i_7 = 0+1+0 = 1 \end{aligned}$$

Rezultă cuvântul de cod:

$$V=1011010$$

2.1.1.2. Decodarea codului Hamminmg corector de o eroare

Având cuvântul recepţionat V', compus din 7 imboluri:

$$V' = C_1'C_2'i_3'C_4'i_5'i_6'i_7'$$

se poate calcula corectorul codului Hamming cu relaţia:

$$z = H \cdot V'^{T} = \begin{bmatrix} z_4 \\ z_2 \\ z_1 \end{bmatrix} \qquad (2.1.3)$$

Matricea de control:

$$H_{[m,n]} = \begin{bmatrix} h_1 & \cdots & h_i & \cdots & h_n \end{bmatrix} \qquad (2.1.4)$$

unde, pentru acest caz se observă că m=3 şi n=7. Coloana h_i exprimă în cod binar natural numărul de ordine al coloanei respective, cu bitul cel mai puţin semnificativ în linia *m*.

Din relaţia (2.1.3) rezultă:

$$\begin{aligned} z_4 &= C_4' + i_5' + i_6' + i_7' \\ z_2 &= C_2' + i_3' + i_6' + i_7' \\ z_1 &= C_1' + i_3' + i_5' + i_7' \end{aligned} \qquad (2.1.5)$$

Decodarea zecimală a corectorului z, indică poziţia erorii, dacă este una singură, în cuvântul V'. Va fi eronat simbolul cu indicele r, dat de relaţia:

$$r = 4z_4 + 2z_2 + z_1 \qquad (2.1.6)$$

Cuvântul recepţuionat se mai poate scrie ca fiind:

$$V' = V + \varepsilon,$$

unde ε reprezintă cuvântul eroare.
Relaţia (2.1.3) va deveni:

$$z = H \cdot V'^{T} = H \cdot (V + \varepsilon)^{T} = H \cdot \varepsilon^{T} = h_r$$

unde poziţia erorii se calculează cu relaţia (2.1.6).

Obs: *În cazul în care există două erori, sunt cazuri în care nu numai că nu se corectează nici o eroare, dar se mai eronează un al treilea simbol, rezultând la recepţie trei simboluri eronate.*

Exemplul 2:

Decodaţi cuvântul recepţionat V'=1001001

Rezolvare:

Cuvântul de cod recepţionat se poate scrie ca fiind:

$$V' = C_1' C_2' i_3' C_4' i_5' i_6' i_7'$$

$$V' = 1\ 0\ 0\ 1\ 0\ 0\ 1$$

Relaţia (2.1.5) va deveni:

$$z_4 = C_4' + i_5' + i_6' + i_7' = 1 + 0 + 0 + 1 = 0$$
$$z_2 = C_2' + i_3' + i_6' + i_7' = 0 + 0 + 0 + 1 = 1$$
$$z_1 = C_1' + i_3' + i_5' + i_7' = 1 + 0 + 0 + 1 = 0$$

Aşadar, poziţia eronată este:

$$r = 4z_4 + 2z_2 + z_1 = 4 \cdot 0 + 2 \cdot 1 + 1 \cdot 0 = 2$$

Rezultă cuvântul eroare:

$$\varepsilon = 0100000$$

Se poate scrie:

$$V = V' + \varepsilon = 1001001 + 0100000 = 1101001$$

Rezultă secvenţa de informaţie:

$$i = i_3 i_5 i_6 i_7 = 0001$$

2.1.2. Cod Hamming corector de o eroare şi detector de două erori

Condiţia necesară şi suficientă pentru ca un cod să poată simultan corecta maxim *t* erori şi a detecta maxim *e* erori este ca distanţa minimă a codului să fie:

$$d \geq t + e + 1 = 1 + 2 + 1 = 4, \qquad e > t$$

2.1.2.1. Codarea codului Hamminmg corector de o eroare şi detector de două erori

Pentru a înlătura dezavantajul codului Hamming corector de o eroare (acela de a erona suplimentar, la depăşirea capacităţii de corecţie a codului, t>1) şi de al face mai util în aplicaţii

practice, codul a fost modificat în sensul creşterii distanţei minime de la d=3 la d=4, ceea ce permite detecţia erorilor duble.

Creşterea distanţei de cod de la 3 la 4 s-a făcut prin adăugarea unui simbol de control suplimentar, numit simbol de control al parităţii, C_0, structura cuvîntului de cod devenind:

$$V = C_0C_1C_2i_3C_4i_5i_6i_7$$

Matricea de control se modifică şi are structura:

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

Relaţia de codare va fi:

$$H' \cdot V^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ i_3 \\ C_4 \\ i_5 \\ i_6 \\ i_7 \end{bmatrix} = 0 \qquad (2.1.7)$$

Sau în formă explicită:

$$\begin{aligned} C_1 &= i_3 + i_5 + i_7 \\ C_2 &= i_3 + i_6 + i_7 \\ C_4 &= i_5 + i_6 + i_7 \\ C_0 &= C_1 + C_2 + i_3 + C_4 + i_5 + i_6 + i_7 \end{aligned} \qquad (2.1.8)$$

Exemplul 3:

Fie secvenţa de informaţie i=1010, găsiţi cuvântul V de cod.

Rezolvare:

Se observă că există 4 simboluri de informaţie: i_3, i_5, i_6, i_7. Cuvântul de cod, V, se poate scrie sub formă literară: $V = C_0C_1C_2i_3C_4i_5i_6i_7$. Avem aşadar 8 simboluri ce alcătuiesc cuvântul de cod şi 4 biţi de control. Rezultă că matricea de control, H', va conţine 4 linii şi 8 coloane.
Din relaţia de codare (2.1.8), rezultă relaţiile de calcul a biţilor de control:

$$\begin{aligned} C_1 &= i_3 + i_5 + i_7 = 1+0+0 = 1 \\ C_2 &= i_3 + i_6 + i_7 = 1+1+0 = 0 \\ C_4 &= i_5 + i_6 + i_7 = 0+1+0 = 1 \\ C_0 &= C_1 + C_2 + i_3 + C_4 + i_5 + i_6 + i_7 = 1+0+1+1+0+1+0 = 0 \end{aligned}$$

Rezultă cuvântul de cod:

V=01011010

2.1.2.2. Decodarea codului Hamminmg corector de o eroare şi detector de două erori

Având cuvântul recepţionat V', compus din 8 imboluri:

$$V' = C_0'C_1'C_2'i_3'C_4'i_5'i_6'i_7'$$

se poate calcula corectorul codului cu relaţia:

$$Z = H' \cdot V'^{T} = \begin{bmatrix} z \\ z_0 \end{bmatrix} = \begin{bmatrix} z_4 \\ z_2 \\ z_1 \\ z_0 \end{bmatrix} \qquad (2.1.9)$$

Unde:

- z, reprezintă corectorul de la codul Hamming corector de o eroare;
- z_0, reprezintă simbolul binar, 0 sau 1, cu ajutorul căruia se pot detecta erori pare (z_0=0).

Există patru cazuri:

Cazul 1:

$$\begin{cases} z = 0 \\ z_0 = 0 \end{cases}$$

nu există erori sau nu există erori detectabile prin cod.

Cazul 2:

$$\begin{cases} z \neq 1 \\ z_0 = 0 \end{cases}$$

se face detecţia erorilor duble.

Cazul 3:

$$\begin{cases} z = 0 \\ z_0 = 1 \end{cases}$$

simbolul C_0 este eronat.

Cazul 4:

$$\begin{cases} z \neq 0 \\ z_0 = 1 \end{cases}$$

există o eroare corectabilă.

Va fi eronat simbolul cu indicele r-1, unde r este dat de relaţia:

$$r = 4z_4 + 2z_2 + z_1 + z_0 \qquad (2.1.10)$$

Exemplul 4:

Decodaţi cuvântul recepţionat V'=01001010

Rezolvare:

Cuvântul de cod recepţionat se poate scrie ca fiind:

$$V' = C_0'C_1'C_2'i_3'C_4'i_5'i_6'i_7'$$
$$V' = 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0$$

Relaţia (2.1.5) va deveni:

$$z_4 = C_4^{'} + i_5^{'} + i_6^{'} + i_7^{'} = 1+0+1+0 = 0$$
$$z_2 = C_2^{'} + i_3^{'} + i_6^{'} + i_7^{'} = 0+0+1+0 = 1$$
$$z_1 = C_1^{'} + i_3^{'} + i_5^{'} + i_7^{'} = 1+0+0+0 = 1$$
$$z_0 = C_0^{'} + C_1^{'} + C_2^{'} + i_3^{'} + C_4^{'} + i_5^{'} + i_6^{'} + i_7^{'} = 0+1+0+0+1+0+1+0 = 1$$

Rezultă că avem situaţia cazului 4, când există o eroare corectabilă.
Aşadar, rezultă r:

$$r = 4z_4 + 2z_2 + z_1 + z_0 = 4\cdot 0 + 2\cdot 1 + 1\cdot 1 + 1 = 4$$

Deci va fi eronat simbolul cu indicele r-1, adică 4-1=3, i_3.
Rezultă cuvântul eroare:

$$\varepsilon = 00010000$$

Se poate scrie:

$$V = V^{'} + \varepsilon = 01001010 + 00010000 = 01011010$$

Rezultă secvenţa de informaţie:

$$i = i_3 i_5 i_6 i_7 = 1010$$

Desfăşurarea lucrării:
a) Se lansează executabilul Hamming.exe din directorul Hamming, după care se introduce parola TTI. Rulaţi programul, selectând opţiunea Demonstraţie, pentru toate cele patru variante prezentate.
b) Se verifică, cu programul, toate exemplele luate în această lucrare, selectând pe rând codurile prezentate.
c) Pentru fiecare cod în parte, se alege câte un exemplu de codare/decodare, asemănător cu cele prezentate în lucrare, cu observaţia ca secvenţa de informaţie să conţină 11 biţi de informaţie, la codare, respectiv cuvîntul recepţionat să conţină 15/16 biţi, la decodare. Se realizează codarea/decodare, după care, cu programul, se verifică rezultatele obţinute.
d) La sfârşitul lucrării de laborator se va efectua un test asupra cunoştinţelor acumulate. Rezultatul testului se va concretiza printr-o notă, calculată automat de calculator.

2.2. Codul ciclic

Codurile ciclice sunt utilizate pentru protejarea informaţiei împotriva perturbaţiilor.

Codurile ciclice sunt coduri bloc (toate cuvintele au aceeaşi lungime, codarea şi decodarea unui bloc este independentă de a celorlalte).

Denumirea de "ciclice" provine de la proprietatea pe care o au cuvintele de cod şi anume dacă v_i = 1001001 este cuvânt de cod atunci şi v_j = 0010011 şi v_k = 1100100 sunt cuvinte de cod . Adică orice permutare ciclică a unui cuvânt de cod este un alt cuvânt de cod .

Parametrii codului sunt n (biţi) lungimea cuvântului de cod, k numărul de biţi de informaţie per cuvânt, m numărul de biţi de control per cuvânt şi polinomul generator g(x). Deşi poate fi facută şi o codare nesistematică vom considera codul sistematic cei k biţi de informaţie fiind situaţi pe poziţiile cele mai semnificative, iar cei m biţi de control pe poziţiile mai puţin semnificative.
Fiecărui cuvânt de cod i se poate ataşa un polinom de grad maxim n-1:

$$v = v_{n-1}v_{n-2}v_{n-3}.....v_1v_0,$$

coeficienţii v_i fiind coeficienţi binari (din câmpul binar {0,1}).
Polinomul asociat cuvântului va fi :

$$v(x) = v_{n-1}x^{n-1} + v_{n-2}x^{n-2} + v_{n-3}x^{n-3} + ...+ v_1x + v_0 \qquad (2.2.1)$$

Ponderea unui cuvânt reprezintă numărul de 1 din cadrul cuvântului.
Polinomul de informaţie este :

$$i(x) = i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + i_{k-3}x^{k-3} + ...+ i_1x + i_0 \qquad (2.2.2)$$

de grad k-1.

Pentru orice cuvânt de cod este valabilă relaţia :

$$\text{rest}\frac{v(x)}{g(x)} = 0 \qquad (2.2.3)$$

Deci cuvintele de cod se aleg astfel încât să fie multiplii ai polinomului g(x) numit polinomul generator al carui grad va fi deci m = n-k, ($g_m = 1$):

$$g(x) = g_mx^m + g_{m-1}x^{m-1} + g_{m-2}x^{m-2} + ...+ g_1x + g_0 \qquad (2.2.4)$$

Operaţiile se fac modulo $(x^n + 1)$.
Dacă, codul este corector de o singură eroare atunci:

$$n = 2^m - 1. \qquad (2.2.5)$$

Codurile ciclice corectoare de o eroare, având distanţa de cod (distanţa Hamming minimă între cuvintele codului) $d_{Hmin} = 3$, sunt capabile să corecteze o eroare sau să detecteze două.

Codarea codului ciclic corector de o eroare

Codarea va fi prezentată în două moduri nesistematică sau sistematică.

Cu relaţia $v(x) = i(x)\cdot g(x)$ se poate face codarea nesistematică, dar pentru că nu este utilă în practică nu va fi luată în considerare .

În continuare va fi deci prezentată codarea sistematică unde corespondenţa dintre i(x) şi v(x) este dată prin relaţia:

$$v(x) = i(x)\cdot x^m + \text{rest}\frac{i(x)\cdot x^m}{g(x)} \qquad (2.2.6)$$

unde $\text{rest}\frac{i(x)\cdot x^m}{g(x)}$ semnifică restul împărţirii polinomului $i(x)\cdot x^m$ la g(x).

Operaţia de adunare din ecuaţia (2.2.6) este sumă modulo 2 (SAU exclusiv), iar coeficienţii polinoamelor sunt din câmpul binar {0,1}.
Matematic codarea poate fi facută polinomial sau matricial.

a) Polinomial

Codarea va fi prezentată printr-un exemplu, putând fi uşor generalizată.

Fie $g(x) = x^3 + x + 1$. Deci m = 3 şi cu relaţia $2^m - 1 = n$ rezultă că n = 7 de unde k = n – m = 4. Vom avea ca atare 4 biţi de informaţie $i = v_{n-1}v_{n-2}v_{n-3}v_{n-4}$ cu polinomul asociat :

$$i(x) = v_{n-1}x^3 + v_{n-2}x^2 + v_{n-3}x + v_{n-4} \qquad (2.2.7)$$

şi deci $i(x)\cdot x^m = (v_{n-1}x^3 + v_{n-2}x^2 + v_{n-3}x^3 + v_{n-1}x + v_{n-4})\cdot x^3 = v_{n-1}x^6 + v_{n-2}x^5 + v_{n-3}x^4 + v_{n-4}x^3$

Biţii de control sunt coeficienţii restului împărţirii lui $i(x)\cdot x^m/g(x)$.

$$\begin{array}{l|l} v_{n-1}x^6 + v_{n-2}x^5 + v_{n-3}x^4 + v_{n-4}x^3 & \underline{x^3 + x + 1} \\ \underline{v_{n-1}x^6 \quad + v_{n-1}x^4 + v_{n-1}x^3} & v_{n-1}x^3 + v_{n-2}x^2 + (v_{n-3} + v_{n-1})x + (v_{n-4} + v_{n-2} + v_{n-1}) \end{array}$$

$$/\ \ v_{n-2}x^5 + (v_{n-3} + v_{n-1})x^4 + (v_{n-4} + v_{n-1})x^3$$
$$\underline{v_{n-2}x^5 \qquad\qquad + v_{n-2}x^3 \qquad + v_{n-2}x^2}$$

$$/\ \ (v_{n-3} + v_{n-1})x^4 + (v_{n-4} + v_{n-2} + v_{n-1})x^3 + v_{n-2}x^2$$
$$\underline{(v_{n-3} + v_{n-1})x^4 \qquad\qquad + (v_{n-3} + v_{n-1})x^2 + (v_{n-3} + v_{n-1})x}$$

$$/\ \ (v_{n-4} + v_{n-2} + v_{n-1})x^3 + (v_{n-3} + v_{n-2} + v_{n-1})x^2 + (v_{n-3} + v_{n-1})x$$
$$\underline{(v_{n-4} + v_{n-2} + v_{n-1})x^3 \qquad\qquad + (v_{n-4} + v_{n-2} + v_{n-1})x + v_{n-4} + v_{n-2} + v_{n-1}}$$

$$(v_{n-3} + v_{n-2} + v_{n-1})x^2 + (v_{n-4} + v_{n-3} + v_{n-2})x + v_{n-4} + v_{n-2} + v_{n-1}$$

$$\Rightarrow r(x) = (v_{n-3} + v_{n-2} + v_{n-1})x^2 + (v_{n-4} + v_{n-3} + v_{n-2})x + v_{n-4} + v_{n-2} + v_{n-1}, \qquad (2.2.8)$$

unde r(x) este restul împărţirii .
Conform relaţiei (2.2.6) se obţine :

$$v(x) = v_{n-1}x^6 + v_{n-2}x^5 + v_{n-3}x^4 + v_{n-4}x^3 + (v_{n-3} + v_{n-2} + v_{n-1})x^2 + (v_{n-4} + v_{n-3} + v_{n-2})x + v_{n-4} + v_{n-2} + v_{n-1}$$

Pentru că n = 7 şi v(x) este de forma:

$$v(x) = v_6x^6 + v_5x^5 + v_4x^4 + v_3x^3 + v_2x^2 + v_1x + v_0 \qquad (2.2.9)$$

şi ţinând cont de relaţia (2.2.8) simbolurile de control vor fi date de:

$$\begin{aligned} v_2 &= v_4 + v_5 + v_6 \\ v_1 &= v_3 + v_4 + v_5 \\ v_0 &= v_3 + v_5 + v_6 \end{aligned} \qquad (2.2.10)$$

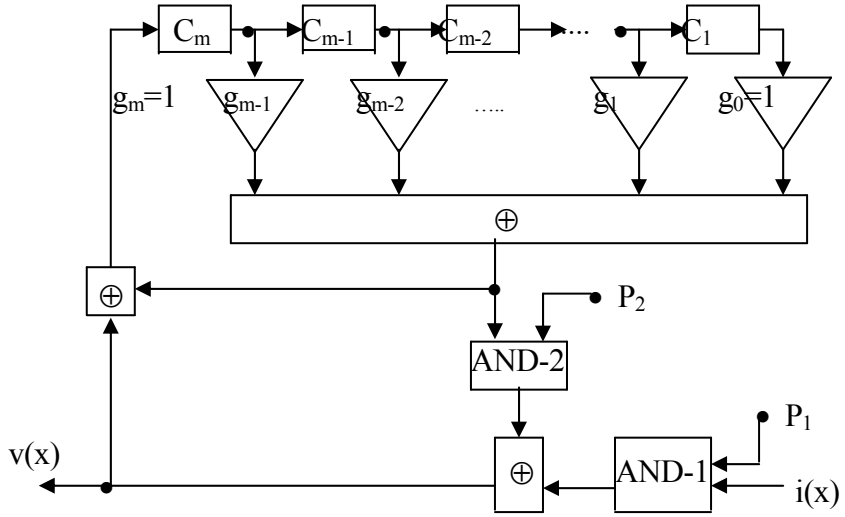
Aşadar cuvântul de cod va fi $v = v_6v_5v_4v_3v_2v_1v_0$ unde primii 4 sunt biţii de informaţie, iar ultimii 3 cei de control.

b) Codor ciclic cu RDR (codarea matricială)

În Fig. 2.2.1 este prezentat un codor ciclic care implementează relaţia de codare (2.2.6). Secvenţa de informaţie, i(x), intră în codor în primele k tacte, primul bit fiind cel mai semnificativ şi, de asemenea, este conectată şi la ieşire. Pentru aceasta, întrerupătorul 1, format din poarta AND-1 este închis iar întrerupătorul 2, format din poarta AND-2 este deschis (vezi semnalele de comandă P_1 şi P_2).

În următoarele m tacte întrerupătorul 1 este deschis iar întrerupătorul 2 este închis, astfel că secvenţa r, generată de RDR, este livrată la ieşirea v.

a) Codor ciclic RDR ;



b) Semnalele de validare a porţilor „AND”

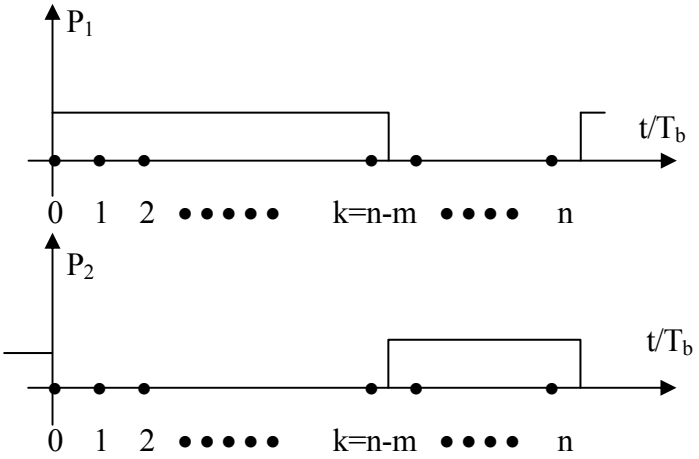


Fig. 2.2.1 Codor ciclic cu RDR şi semnalele de comandă

Obs. *–Se observă că, în ultimele k tacte, la intrările sumatorului (care adună intrarea RDR-lui cu reacţia sa) se regăseşte acelaşi semnal, astfel că, în aceste k tacte, în celula C_k se va introduce o secvenţă nulă care „curăţă” registrul pentru un nou cuvânt de cod.*

<u>Din funcţionarea registrului de deplasare cu reacţie rezultă relaţia:</u>

$$S_j = TS_{j-1} + v_{n-j}U \tag{2.2.11}$$

unde S reprezintă starea registrului, U este o matrice coloană, iar T este matricea caracteristică a registrului de deplasare cu reacţie. Ele sunt de forma:

$$S=\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}, \qquad U=\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, \qquad T=\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ g_0 & g_1 & g_2 & \cdots & g_{m-1} \end{bmatrix}. \tag{2.2.12}$$

Ţinând cont de observaţia făcută mai sus rezultă că $S_n = 0$.
Considerând acelaş exemplu

Matricea caracteristică, T, este:

$$T = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \tag{2.2.13}$$

Utilizând relaţiile (2.2.11) şi (2.2.13) pentru cazul n = 7 vom avea:

$$S_7 = v_6T^6U + v_5T^5U + v_4T^4U + v_3T^3U + v_2T^2U + v_1TU + v_0U = 0 \tag{2.2.14}$$

Efectuând calculele rezultă:

$$T \cdot U = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix},\ T^2 \cdot U = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix},\ T^3 \cdot U = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix},\ T^4 \cdot U = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix},$$
$$T^5 \cdot U = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix},\ T^6 \cdot U = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{2.2.15}$$

Introducând (2.2.15) în (2.2.14) şi efectuând calculele obţinem:

$$v_2 = v_4 + v_5 + v_6$$
$$v_1 = v_3 + v_4 + v_5$$
$$v_0 = v_4 + v_3 + v_2 = v_4 + v_3 + v_4 + v_5 + v_6 = v_3 + v_5 + v_6$$

relaţii identice cu relaţiile (2.2.10) deci cele două proceduri de calcul ne-au dus la aceleaşi rezultate .

Cuvântul de cod va fi $v = v_6v_5v_4v_3v_2v_1v_0$.

Decodarea codului ciclic corector de o eroare

Decodarea codului ciclic cuprinde verificarea corectitudinii cuvântului recepţionat:

$$w(x) = v'_{n-1}x^{n-1} + v'_{n-2}x^{n-2} + ... + v'_1x + v'_0, \tag{2.2.16}$$

corectarea sau detectarea erorilor, după destinaţia codului şi apoi selecţia biţilor de informaţie.

Verificarea presupune calculul restului împărţirii lui w(x) la g(x). Dacă restul este 0 se decide: cuvânt corect recepţionat. Dacă nu, atunci se decide: cuvânt eronat.(Prima decizie poate fi falsă, a doua nu!) Pentru codurile detectoare decodarea ia sfârşit aici. Pentru codurile corectoare analiza restului permite determinarea poziţiei erorilor. Acest lucru presupune existenţa unei corespondenţe biunivoce între resturile posibile şi cuvintele eroare corectabile de către codul dat.

Dacă codul este corector de o eroare, atunci decodarea decurge astfel:

1°- se calculează corectorul z_n cu formula:

$$z = \sum_{j=0}^{n-1} v'_j T^j U = \sum_{j=0}^{n-1} \varepsilon_j T^j U \tag{2.2.17}$$

unde ε_j reprezintă coeficienţii polinomului eroare:

$$\varepsilon(x) = w(x) + v(x) \tag{2.2.18}$$

2°- dacă z = 0 se decide că w(x) este corect w(x) = v(x)

- dacă $z \neq 0$ atunci $z = T^rU$, unde r este indicele coeficientului eronat. Comparăm z cu T^jU şi găsim r. Corecţia presupune schimbarea valorii lui v'_r.

Dacă codul ciclic corectează mai multe erori, atunci decodarea se poate face pe seama corespondenţei amintite anterior.

a) Decodarea polinomială

Metoda constă în împărţirea lui w(x) la g(x). Va rezulta un polinom rest, r(x). Dacă acesta este diferit de 0, cuvântul recepţionat este eronat şi prin identificarea restului r(x) cu valorile din tabelul cuvinte eroare – corectori (vezi Anexe) se determină poziţia erorii şi se realizează corecţia.

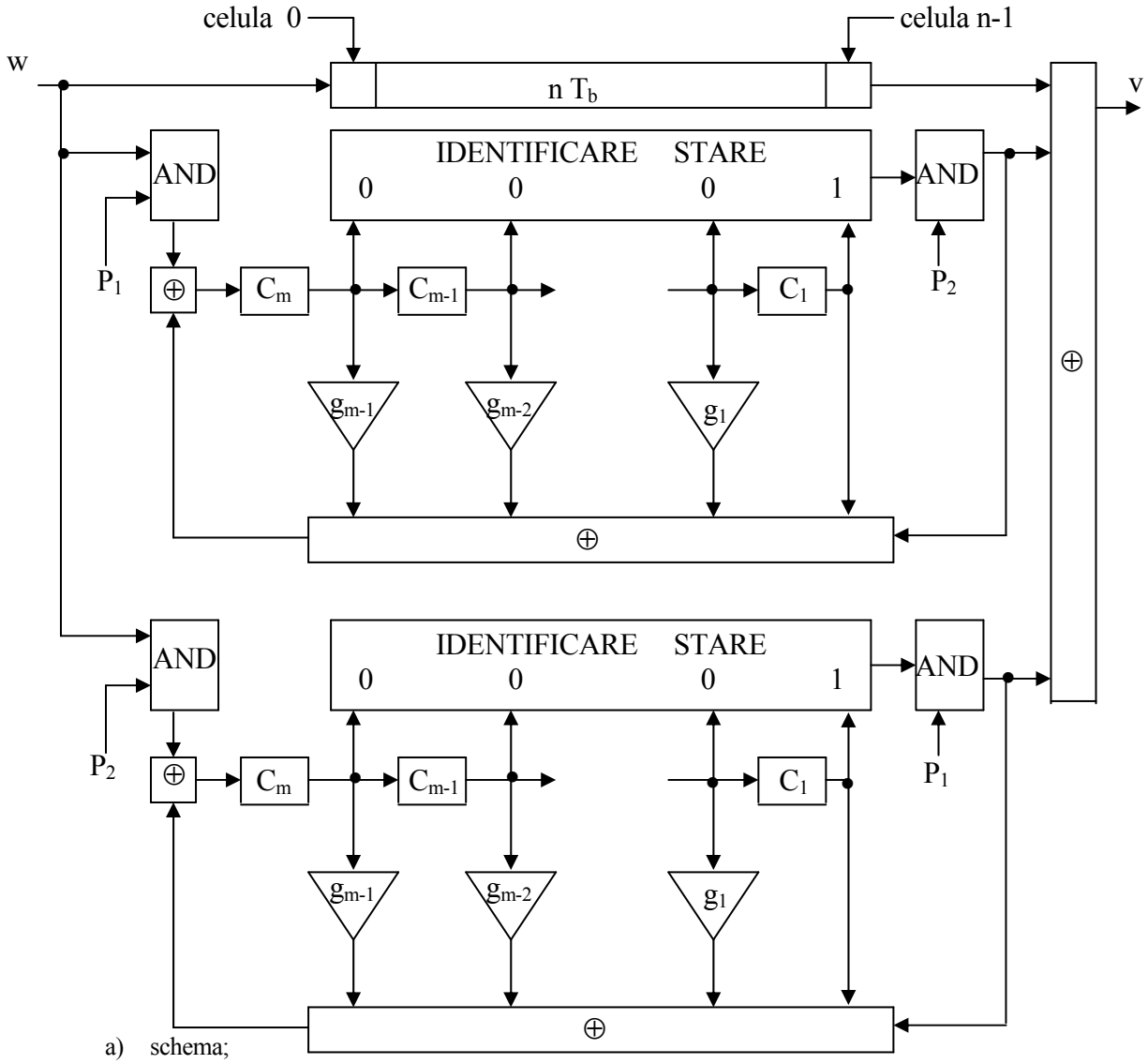
Spre exemplu fie : $w(x) = x^6 + x^5 + x^3 + x^2 + 1$. Calculăm rest[$w(x) / g(x)$]:

$$
\begin{array}{l|l}
x^6 + x^5 + x^3 + x^2 + 1 & x^3 + x + 1 \\
\underline{x^6 + x^4 + x^3} & \underline{x^3 + x^2 + x + 1} \\
x^5 + x^4 + x^2 + 1 & \\
\underline{x^5 + x^3 + x^2} & \\
x^4 + x^3 + 1 & \\
\underline{x^4 + x^2 + x} & \\
x^3 + x^2 + x + 1 & \\
\underline{x^3 + x + 1} & \\
x^2 &
\end{array}
$$

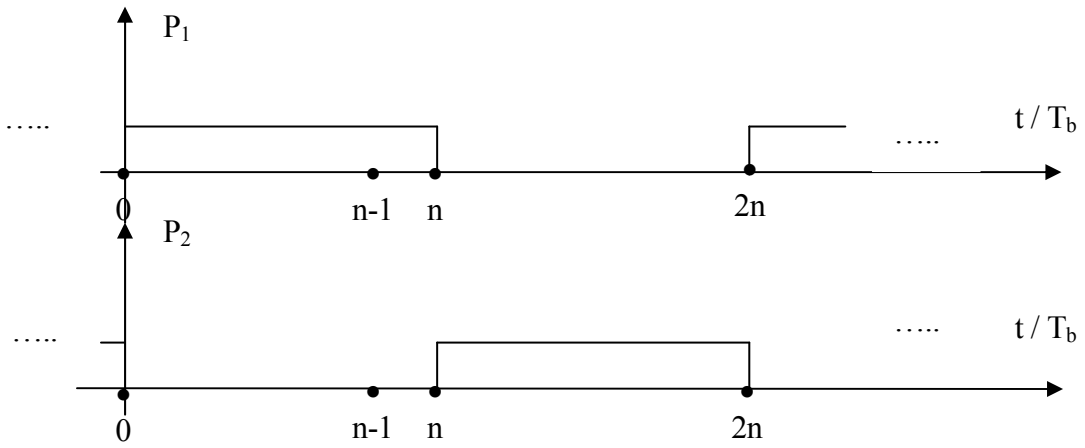
deci conform tabelului din anexă este eronat w_2. Schimbând valoarea bitului w_2 rezultă :

$$w(x) = x^6 + x^5 + x^3 + 1.$$

b) Decodor ciclic cu RDR (decodarea matricială)



a) schema;



b)Semnalele de validare a porţilor AND ;

Fig. 2.2.2. Decodor ciclic corector de o eroare

În Fig. 2.2.2. este prezentată structura unui decodor ciclic corector de o eroare. Biţii cuvântului recepţionat vor intra în schemă unul după altul în ordine, începând cu bitul v'_{n-1}.

Schema de decodare conţine un registru de deplasare (blocul „nT_b" este un registru de întârziere cu n celule) cu rol de memorie, care păstrează cuvântul recepţionat care intră pas cu pas şi două subscheme cu RDR pentru corecţie care funcţionează în contratimp. Procedura de decodare durează 2n tacte. În primele n tacte în memorie intră cuvântul 1 care prin poarta P_1 intră şi în primul decodor, ieşirea acestuia fiind 0 pentru că P_2 este închis. În următoarele n tacte cuvântul 2 va intra prin poarta P_2 în al doilea decodor care are ieşirea 0 deoarece P_1 este închis, în acest timp cuvântul 1 părăseşte memoria trecând prin sumator şi este corectat de primul decodor care identifică starea 00.....1 şi care are acces la sumator prin P_2.

După n = 7 tacte starea registrului va fi:

$$S'_7 = v'_6T^6U + v'_5T^5U + v'_4T^4U + v'_3T^3U + v'_2T^2U + v'_1TU + v'_0U \qquad (2.2.19)$$

care nu va mai fi 0 decât în cazul în care nu avem eroare.

Dacă eroarea afectează bitul v_r starea registrului de deplasare cu reacţie după intrarea întregului cuvânt este:

$$S'_7 = v'_6T^6U + v'_5T^5U + ... + v'_rT^rU + ... + v'_1TU + v'_0U \qquad (2.2.20)$$

Deoarece la emisie $S_7 = 0$ şi $v'_j = v_j$ pentru $j \neq r$ şi $v'_r = v_r + 1$ rezultă:

$$S'_7 + S_7 = S'_7 = T^rU = z \qquad (2.2.21)$$

Pe durata următoarelor n = 7 tacte RDR din prima subschemă va evolua numai sub acţiunea semnalului de reacţie şi după n-r-1 tacte se va ajunge în starea:

$$S'_{7+n-r-1} = T^{n-r-1}\, S'_7 = T^{n-1}U = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad (2.2.22)$$

În acelaş timp printr-o deplasare cu n-r-1 tacte eroarea care se afla în celula r va ajunge în celula n-1. Detectorul va sesiza starea $S'_{7+n-r-1}$ fixă şi prin poarta P_2 care este închisă va emite semnalul de corecţie (blocul IDENTIFICĂ generează un 1^L) care se însumează cu bitul eronat aflat în celula n-1, astfel la ieşirea sumatorului final se va obţine cuvântul corectat.

Desfăşurarea lucrării:

a) Rulaţi programul de pe calculator alegând butonul “Demonstraţie” de la codul ciclic şi se urmăreşte o codare şi o decodare.

b) Alegeţivă un polinom generator şi secvenţa de informaţie şi efectuaţi o codare, apoi alegând polinomul generator şi biţii recepţionaţi o decodare. Verificaţi apoi rezultatul cu ajutorul calculatorului.

c) La sfârşitul lucrării de laborator se efectuează testul de verificare a cunoştinţelor acumulate prin intermediul calculatorului. Testul constă în a răspunde la 5 întrebări teoretice, fiecare având un răspuns corect din cele cinci propuse, şi efectuarea unei codări şi a unei decodări pornind de la polinomul generator şi secvenţa de informaţie respectiv secvenţa recepţionată generate aleator de calculator.

L3. Coduri ciclice corectoare de erori multiple

3.1 Codul BCH

Codurile BCH fac parte din categoria codurilor ciclice.
Cuvintele de cod BCH vor avea deci structura ca şi cele de cod ciclic:

$$v = v_{n-1}\, v_{n-2} \ldots v_1\, v_0 \qquad v_j \in GF(2^q, p(x)) \quad j = 0 \div n-1 \tag{3.1.1}$$

unde coeficienţii u_j sunt coeficienţi binari (fac parte din câmpul binar {0, 1}).
Polinomul corespunzător acestui cuvânt fiind:

$$v(x) = v_{n-1}x^{n-1} + v_{n-2}x^{n-2} + \ldots + v_1x + v_0 \tag{3.1.2}$$

adică un polinom de grad n - 1.

Polinomul de informaţie este:

$$i(x) = i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + \ldots + i_1x + i_0 \tag{3.1.3}$$

care este un polinom de gradul k - 1.

Cuvintele de cod se aleg astfel încât să fie multiplii ai polinomului generator g(x) şi deci acesta va trebui sa fie un polinom de gradul m = n – k:

$$g(x) = g_mx^m + g_{m-1}x^{m-1} + \ldots + g_1x + g_0 \tag{3.1.4}$$

Coeficienţii polinomului g(x) sunt de asemenea coeficienţi binari.
Deoarece polinomul trebuie să aibă gradul m rezultă că $g_m = 1$, iar g_0 trebuie să fie şi el egal cu 1 pentru că altfel putem da factor pe x şi gradul polinomului va fi mai mic decât k.

Polinomul g(x) este deci de forma:

$$g(x) = x^m + g_{m-1}x^{m-1} + \ldots + g_1x + 1 \tag{3.1.5}$$

Cuvintele de cod sunt elemente ale câmpului Galois $GF(2^q)$ generat de un polinom primitiv p(x) de grad q (sunt clase de resturi modulo p(x)), câmp obţinut aşa cum este prezentat în Anexa.

Codarea BCH

Codarea codurilor BCH poate fi făcută în două moduri:

a) Cu relaţia

$$v(x) = i(x)\cdot g(x) \tag{3.1.6}$$

care conduce la obţinerea unui cod nesistematic (relaţie mai puţin utilizată).

b) Pentru obţinerea unei structuri sistematice, la care informaţia să se găsească nemodificată pe poziţiile cele mai semnificative se foloseşte relaţia:

$$v(x) = i(x)\cdot x^m + \text{rest}\,(i(x)\cdot x^m/g(x)) \tag{3.1.7}$$

Această relaţie se mai poate scrie:

$$v(x) = q(x)\cdot g(x) = v_{n-1}x^{n-1} + v_{n-2}x^{n-2} + \ldots + v_1 x + v_0 \qquad (3.1.8)$$

deci se obţine un cuvânt de cod ciclic care este multiplu a lui g(x) şi este format din simbolurile de informaţie aflate pe poziţiile cele mai semnificative (primele k) şi m simboluri de control determinate de restul împărţirii lui $i(x)\cdot x^m$ la g(x).
Relaţia (3.1.8) poate fi adusă sub forma $H\cdot v^T = 0$.

$$\begin{bmatrix} h_0 & h_1 & \ldots & h_{m-1} & h_m & 0 & \ldots & 0 & 0 \\ 0 & h_0 & \ldots & h_{m-2} & h_{m-1} & h_m & \ldots & 0 & 0 \\ \vdots & & & & & & & & \\ 0 & 0 & \ldots & 0 & 0 & h_0 & \ldots & h_{m-1} & h_m \end{bmatrix} \begin{bmatrix} v_{n-1} \\ \vdots \\ v_1 \\ v_0 \end{bmatrix} = 0 \qquad (3.1.9)$$

Acesta este de forma $H\cdot v^T = 0$ deci îl putem determina pe H prin identificare ca fiind:

$$H = \begin{bmatrix} h_0 & h_1 & \ldots & h_{m-1} & h_m & 0 & \ldots & 0 & 0 \\ 0 & h_0 & \ldots & h_{m-2} & h_{m-1} & h_m & \ldots & 0 & 0 \\ \vdots & & & & & & & & \\ 0 & 0 & \ldots & 0 & 0 & h_0 & \ldots & h_{m-1} & h_m \end{bmatrix} \qquad (3.1.10)$$

Pentru a corecta t erori independente g(x) trebuie să îndeplinească anumite condiţii. Ştiim că v(x) trebuie să fie multiplu al lui g(x) şi că g(x) trebuie să fie divizor a lui $x^n + 1 = 0$.
Pentru aceasta alegem un număr de r rădăcini ale lui $x^n + 1 = 0$ pe care le notăm cu $\beta_i = \alpha^i \in GF(2^q)$. Aceste rădăcini β_i sunt elemente primitive ale extensiei de ordinul q al câmpului Galois binar $GF(2^q)$. Acest ordin poate fi determinat cu relaţia:

$$n = 2^q - 1 \qquad (3.1.11)$$

de aici rezultând q şi extensia în care se lucrează $GF(2^q)$.
Pe g(x) îl vom determina ca cel mai mic multiplu comun al polinoamelor minimale a rădăcinilor β_i.

$$g(x) = c.m.m.m.c.\{m_1(x),\ldots m_r(x)\} \qquad (3.1.12)$$

Iar dacă toate aceste r polinoame sunt relativ prime între ele atunci:

$$g(x) = \prod_{i=1}^{r} m_i(x) \qquad (3.1.13)$$

Expresia polinomului minimal care este definit ca polinomul $m_i(x)$ ireductibil de grad minim pentru care $m_i(\beta_i) = 0$ este:

$$m_i(x) = (x+\beta_i)\left(x+\beta_i^{2}\right)\ldots\left(x+\beta_i^{2^{q-1}}\right) \qquad (3.1.14)$$

Deşi în calculul lui $m_i(x)$ folosim coeficienţi $GF(2^q)$ coeficienţii lui $m_i(x)$ vor fi binari, fapt care iese în evidenţă şi din calculul acestor coeficienţi în cazul exemplului 1 prezentat mai jos.
Pentru corecţia unui număr de maxim t erori se impune alegerea unui număr de r = 2t rădăcini $\beta_i \in GF(2^q)$ ceea ce determină obţinerea unui cuvânt de cod având distanţa minimă $d \geq 2t + 1$.

În cazul codurilor ciclice binare, dacă α este o rădăcină şi α^2, $\alpha^{2\cdot 2}$, ... $\alpha^{2\cdot(q-1)}$ sunt tot rădăcini şi deci pentru a forma polinomul generator este suficient să luăm rădăcinile impare:

$$\beta_1 = \alpha,\ \beta_3 = \alpha^3,\ \dots,\ \beta_{2t-1} = \alpha^{2t-1}.$$

Structura matricii de control H în cazul codurilor BCH se determină deci impunând ca v(x) să aibă rădăcini pe $\beta_1 = \alpha,\ \beta_3 = \alpha^3,\ \dots,\ \beta_{2t-1} = \alpha^{2t-1}$.
Faptul că β_i este rădăcină a cuvântului de cod v(x) se exprimă prin $v(\beta_i) = 0$ sau mai explicit cu relaţia:

$$v(\beta_i) = v_{n-1}\beta_i^{n-1} + \dots + v_1\beta_i^1 + v_0\beta_i^0 = 0 \quad \text{cu } i = 1, 3, \dots, 2t-1 \tag{3.1.15}$$

Pentru fiecare i putem interpreta această relaţie ca pe un produs scalar al cuvântului format prin puterile lui α^i sau (β_i) şi cuvântul de cod:

$$\begin{bmatrix} \alpha^{i\cdot(n-1)} & \cdots & \alpha^{i\cdot 1} & \alpha^{i\cdot 0} \end{bmatrix} \begin{bmatrix} v_{n-1} \\ \vdots \\ v_1 \\ v_0 \end{bmatrix} = 0 \tag{3.1.16}$$

Înlocuind cu valorile corespunzătoare pentru i (i = 1, 3, ..., 2t-1) obţinem:

$$\begin{bmatrix} \alpha^{n-1} & \cdots & \alpha^1 & \alpha^0 \\ \alpha^{3\cdot(n-1)} & \cdots & \alpha^3 & \alpha^0 \\ \vdots & & & \\ \alpha^{(2t-1)\cdot(n-1)} & \cdots & \alpha^{2t-1} & \alpha^0 \end{bmatrix} \begin{bmatrix} v_{n-1} \\ \vdots \\ v_1 \\ v_0 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{3.1.17}$$

Deci matricea de control H în cazul codurilor BCH este de forma:

$$H = \begin{bmatrix} \alpha^{n-1} & \cdots & \alpha^2 & \alpha^1 & \alpha^0 \\ \alpha^{3\cdot(n-1)} & \cdots & \alpha^6 & \alpha^3 & \alpha^0 \\ \vdots & & & & \\ \alpha^{(2t-1)\cdot(n-1)} & \cdots & \alpha^{(2t-1)\cdot 2} & \alpha^{2t-1} & \alpha^0 \end{bmatrix} \tag{3.1.18}$$

Această matrice este formată din elementele GF(2^q) şi are t linii şi n coloane. Fiecare element poate fi exprimat printr-un cuvânt binar din q biţi şi deci vom putea scrie matricea H şi sub forma binară dispunând vertical cuvintele binare care înlocuiesc puterile lui α. Forma binară a matricii H va avea astfel q·t linii şi tot n coloane.

Exemplul 1:

Proiectarea unui cod BCH de lungime n = 15 şi corectând 2 erori cu determinarea polinomului generator şi a relaţiilor de codare.

$$n = 2^q - 1 = 15 \Rightarrow q = 4 \Rightarrow \text{extensia GF}(2^4)$$

Elementele lui GF(2^q) sunt clase de resturi modulo p(x) un polinom primitiv de gradul q = 4.

Un polinom p(x) este primitiv dacă este ireductibil şi binomul $x^n + 1$ pe care p(x) îl divide are cel puţin gradul $n = 2^q - 1$. Fie acest polinom primitiv $p(x) = x^4 + x + 1$. Putem să constatăm că $x^4 + x + 1$ divide pe $x^{15} + 1$ ($n = 2^q - 1 = 2^4 - 1 = 15$), dar nu divide nici un $x^n + 1$ cu $1 \le n < 15$ deci este primitiv. Se ştie că pentru orice câmp Galois $\alpha^n = 1$, deci la noi $\alpha^{15} = 1$.
Câmpul GF(2^4) generat de $p(x) = x^4 + x + 1$ obţinut printr-o rădăcină primitivă α a lui $x^4 + x + 1$ cu relaţia $\alpha^4 = \alpha + 1$ este cel dat în Anexa:

Pentru t = 2 se aleg rădăcinile $\beta_1 = \alpha$ si $\beta_3 = \alpha^3$, suntem în cazul binar şi alegem doar rădăcinile impare până la ordinul 2t-1 = 4-1 = 3).
Polinoamele minimale vor fi conform relaţiei (3.1.14):

$$m_1(x) = (x + \alpha)\cdot(x + \alpha^2)\cdot(x + \alpha^4)\cdot(x + \alpha^8) = x^4 + x^3(1 + \alpha^2 + \alpha^2 + 1 + \alpha + \alpha) + x^2(1+ \alpha + \alpha^2 + 1+ \alpha + \alpha^2 + \alpha^3 + \alpha^2 + \alpha^3 +\alpha + \alpha^3 + \alpha^3 + \alpha + \alpha^2) + x(1 + \alpha^3 + \alpha + \alpha^2 + \alpha^3 + 1 + \alpha^2 + \alpha^3 + 1 + \alpha + \alpha^3) + \alpha^{15} = x^4 + x + 1$$

$$m_3(x) = (x + \alpha^3)\cdot(x + \alpha^6)\cdot(x + \alpha^{12})\cdot(x + \alpha^{24}) = x^4 + x^3(\alpha^{24} + \alpha^{12} + \alpha^6 + \alpha^3) + x^2(\alpha^{36} + \alpha^{30} + \alpha^{18} + \alpha^{27} + \alpha^{15} + \alpha^9) + x(\alpha^{42} + \alpha^{39} + \alpha^{33} + \alpha^{21}) + \alpha^{45} = x^4 + x^3 + x^2 + x + 1$$

În aceste două relaţii puterile lui α mai mari decât 15 s-au exprimat în funcţie de α^{15}, iar suma s-a efectuat modulo doi.
Cele două polinoame fiind prime între ele şi folosind relaţia (3.1.13), g(x) se obţine:

$$g(x) = m_1(x)\cdot m_3(x) = (x^4 + x + 1)\cdot(x^4 + x^3 + x^2 + x + 1) = x^8 + x^7 + x^6 + x^5 + x^4 + x^5 + x^4 + x^3 + x^2 + x + x^4 + x^3 + x^2 + x + 1 = x^8 + x^7 + x^6 + x^4 + 1$$

$\Rightarrow$ m = 8 simboluri de control $\Rightarrow$ k = 15 - 8 = 7 simboluri de informaţie, adică g = 111010001. Cuvântul de cod va fi:

$$v = v_{14}v_{13}v_{12}v_{11}v_{10}v_9v_8v_7v_6v_5v_4v_3v_2v_1v_0$$

unde primii 7 sunt biţii de informaţie şi ultimii 8 cei de control.
Matricea de control H va avea 2 linii (t = 2) şi 15 coloane (n = 15):

$$H = \begin{bmatrix} \alpha^{14} & \alpha^{13} & \alpha^{12} & \alpha^{11} & \alpha^{10} & \alpha^9 & \alpha^8 & \alpha^7 & \alpha^6 & \alpha^5 & \alpha^4 & \alpha^3 & \alpha^2 & \alpha^1 & 1 \\ \alpha^{42} & \alpha^{39} & \alpha^{36} & \alpha^{33} & \alpha^{30} & \alpha^{27} & \alpha^{24} & \alpha^{21} & \alpha^{18} & \alpha^{15} & \alpha^{12} & \alpha^9 & \alpha^6 & \alpha^3 & 1 \end{bmatrix}$$

Luând puterile lui α modulo 15 şi ţinând cont că $\alpha^{15} = 1$ cu ajutorul tabelului câmpului GF(2^4) obţinem exprimarea binară a lui H:

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

adică q·t = 4·2 = 8 linii şi n = 15 coloane.

Folosind relaţia (3.1.9) se pot obţine relaţiile de codare.
Pentru o secvenţă informaţională i = 0000001 cuvântul de cod sistematic se poate determina cu relaţia (3.1.7):

$i(x) = 1$
$x^m\cdot i(x) = x^8$
$i(x)\cdot x^m/g(x) = x^8 / x^8 + x^7 + x^6 + x^4 + 1 \Rightarrow$
$\Rightarrow rest(i(x)\cdot x^m/g(x)) = x^7 + x^6 + x^4 + 1$
$\Rightarrow v(x) = x^8 + x^7 + x^6 + x^4 + 1$
adică v = 000000111010001

Decodarea BCH

În cazul codurilor ciclice binare pentru a putea corecta erori este suficient să determinăm poziţia acestora din expresia sindromului. Modul în care poziţia erorilor poate fi găsită cu ajutorul sindromului rezultă din prezentarea următoare.
Un cod ciclic corector de t erori are cuvinte de cod care au un număr r = 2t de rădăcini $\beta_i \in GF(2^q)$, $\beta_i=\alpha^i$:

$$v(\beta_i)= v(\alpha^i) = 0 \qquad (3.1.19)$$

La recepţie trebuie să verificăm legea de codare dată de relaţia (3.1.15). În cazul în care avem erori de tip aditiv putem exprima această lege sub forma:

$$r(\beta_i) = u(\beta_i) + e(\beta_i) = e(\beta_i) = e(\alpha^i) = S_i \qquad (3.1.20)$$

unde $e(\beta_i)$ este eroarea şi S_i este sindromul, iar i = 1, 3, ..., 2t-1 în cazul codurilor binare.

Pentru a vedea cum putem găsi poziţia erorii cu ajutorul sindromului luăm un mic exemplu:

Presupunem un cuvânt oarecare care are erori pe două poziţii de exemplu 1 şi 4 deci $e(x)=x^4+x$. Ţinând cont de relaţia (3.1.20) vom avea sindromul:

$$S_i = e(\alpha^i) = (\alpha^i)^1 + (\alpha^i)^4 = (\alpha^1)^i + (\alpha^4)^i = X_1^i + X_2^i$$

Expresia care indică poziţia erorii s-a notat cu X_k - **locatorul erorii**:

$$X_k = \alpha^j \text{ , } k = 1, \ldots t \text{ şi } j = 0, \ldots n-1 \qquad (3.1.21)$$

Deci avem atâţia locatori câte erori (în număr de t), iar erorile pot apărea pe orice poziţie în cadrul cuvântului (de la 0 la n-1).
Deci putem exprima sindromul S_i sub forma:

$$S_i = \sum_{k=1}^{t} X_k^i \qquad (3.1.22)$$

Această relaţie ne indică un sistem de ecuaţii neliniare care are ca necunoscute pe X_k (adică locatorii).

Pentru realizarea decodării va fi prezentat algoritmul Peterson cu căutare Chien care constă în:
1° Calcularea sindromului erorii

$$S_i = r\left(\alpha^i\right) = \sum_{k=1}^{t} X_k^i \quad \text{cu} \quad i=1, 3, \ldots 2t-1$$

2° Cu ajutorul locatorilor putem forma un polinom al erorilor care are ca rădăcini locatorii:

$$\sigma(x) = \prod_{k=1}^{t} (x + X_k) = x^t + \sigma_1 x^{t-1} + \ldots + \sigma_t \qquad (3.1.23)$$

Dacă punem condiţia ca X_k să fie rădăcină a lui $\sigma(x)$ obţinem:

$$X_k^t + \sigma_1 X_k^{t-1} + \ldots + \sigma_t = 0 \qquad (3.1.24)$$

Dacă înmulţim (3.1.24) cu X_k^i şi însumăm în funcţie de k:

$$\sum_{k=1}^{t} X_k^{t+i} + \sigma_1 \sum_{k=1}^{t} X_k^{t+i-1} + ... + \sigma_t \sum_{k=1}^{t} X_k^{i} = 0 \qquad (3.1.25)$$

Ţinând cont de (3.1.22) relaţia (3.1.25) devine:

$$S_{t+i} + \sigma_1 S_{t+i-1} + ... + \sigma_t S_i = 0 \qquad (3.1.26)$$

Această ecuaţie este un sistem liniar de t ecuaţii cu t necunoscute care se poate rezolva aplicând regula lui Cramer.

Pentru codurile BCH folosind şi relaţia $S_{2k} = S_k^2$ expresiile coeficienţilor σ_i în funcţie de S_i pentru un număr de 1, 2 sau 3 erori sunt:

Tabelul 3.1.1

t	σ_i
1	$\sigma_1 = S_1$
2	$\sigma_1 = S_1$ $\sigma_2 = \dfrac{S_3 + S_1^3}{S_1}$
3	$\sigma_1 = S_1$ $\sigma_2 = \dfrac{S_1^2 S_3 + S_5}{S_1^3 + S_3}$ $\sigma_3 = S_1^3 + S_3 + S_1 \sigma_2$

3° Căutarea poziţiei eronate.

Cunoscând σ_i putem determina poziţia erorii.

Dacă împărţim relaţia (3.1.24) cu X_k^t rezultă relaţia:

$$\sum_{i=1}^{t} \sigma_i X_k^{-i} = 1 \qquad (3.1.27)$$

i dându-ne poziţia eronată.

Numărul maxim de erori corectabile este t eroarea putându-se găsi pe oricare din cele n poziţii ale cuvântului.

În cazul unei căutări Chien se începe căutarea simbolului eronat de la r_{n-1}. Pentru o poziţie oarecare n-j ţinând cont de (3.1.21) vom avea $X_k = \alpha^{n-j}$

$$\begin{aligned} &\sum_{i=1}^{t} \sigma_i \alpha^{-i(n-j)} = 1 \Rightarrow \sum_{i=1}^{t} \sigma_i \alpha^{-i \cdot n} \cdot \alpha^{i \cdot j} = 1 \\ &\Rightarrow \sum_{i=1}^{t} \sigma_i \left(\frac{1}{\alpha^n} \right)^i \cdot \alpha^{i \cdot j} = 1 \end{aligned} \qquad (3.1.28)$$

Deoarece $\alpha^n = 1$ obţinem ecuaţia de căutare Chien:

$$\sum_{i=1}^{t} \sigma_i \alpha^{i \cdot j} = 1 \quad \text{cu} \quad j = 1,...n \tag{3.1.29}$$

lui j = 1 corespunzându-i simbolul r_{n-1}, iar lui j = n simbolul r_0.
Deci conform ecuaţiei (3.1.29) eroarea apare când suma respectivă este egală cu 1.

Exemplul 2:

Presupunem că recepţionăm un cuvânt BCH cu n = 15 şi k = 7 şi că avem t = 2 erori.

$$r = 110101011010011$$
$$\Rightarrow r(x) = x^{14} + x^{13} + x^{11} + x^9 + x^7 + x^6 + x^4 + x^1 + 1$$

Calculăm sindromul. Pentru t = 2 avem de calculat până la S_3 inclusiv (2t-1 = 2·2-1 = 3):

$$S_1 = r(\alpha) = \alpha^{14} + \alpha^{13} + \alpha^{11} + \alpha^9 + \alpha^7 + \alpha^6 + \alpha^4 + \alpha + 1=$$
$$= 1 + \alpha^3 + 1 + \alpha^2 + \alpha^3 + \alpha + \alpha^2 + \alpha^3 + \alpha + \alpha^3 + 1 + \alpha +$$
$$+ \alpha^3 + \alpha^2 + \alpha^3 + 1 + \alpha + \alpha + 1 = \alpha^2 + \alpha + 1 = \alpha^{10}$$

$$S_3 = r(\alpha^3) = \alpha^{42} + \alpha^{39} + \alpha^{33} + \alpha^{27} + \alpha^{21} + \alpha^{18} + \alpha^{12} + \alpha^3$$
$$+ 1= \alpha^{12} + \alpha^9 + \alpha^3 + \alpha^{12} + \alpha^6 + \alpha^3 + \alpha^{12} + \alpha^3 + 1 = \alpha^9 +$$
$$+ \alpha^6 + \alpha^{12} + \alpha^3 + 1 = \alpha + \alpha^3 + \alpha^2 + \alpha^3 + 1 + \alpha + \alpha^2 + \alpha^3 +$$
$$+ \alpha^3 + 1 = 0$$

Din tabelul 3.1.1 avem pentru t = 2:

$$\sigma_1 = S_1 = \alpha^{10}$$
$$\sigma_2 = \frac{S_3 + S_1^3}{S_1} = \alpha^{20} = \alpha^5$$

În calculele anterioare am utilizat valorile pentru puterile lui α din tabelul 1 şi puterile care depăşesc α^{15} le-am exprimat in funcţie de aceasta ţinând cont că $\alpha^{15} = 1$.
Folosim relaţia de căutare (3.1.29) şi avem:

$j = 1 \Rightarrow \sigma_1\alpha^{1\cdot1} + \sigma_2\alpha^{2\cdot1} = \alpha^{10}\cdot\alpha + \alpha^5\cdot\alpha^2 = \alpha + \alpha^2 + \alpha^3 + 1 + \alpha + \alpha^3 = 1 + \alpha^2 = \alpha^8$
$j = 2 \Rightarrow \sigma_1\alpha^{1\cdot2} + \sigma_2\alpha^{2\cdot2} = \alpha^{10}\cdot\alpha^2 + \alpha^5\cdot\alpha^4 = \alpha^8$
$j = 3 \Rightarrow \sigma_1\alpha^{1\cdot3} + \sigma_2\alpha^{2\cdot3} = \alpha^{10}\cdot\alpha^3 + \alpha^5\cdot\alpha^6 = \alpha^4$
$j = 4 \Rightarrow \sigma_1\alpha^{1\cdot4} + \sigma_2\alpha^{2\cdot4} = \alpha^{10}\cdot\alpha^4 + \alpha^5\cdot\alpha^8 = \alpha^2$
$j = 5 \Rightarrow \sigma_1\alpha^{1\cdot5} + \sigma_2\alpha^{2\cdot5} = \alpha^{10}\cdot\alpha^5 + \alpha^5\cdot\alpha^{10} = 0$
$j = 6 \Rightarrow \sigma_1\alpha^{1\cdot6} + \sigma_2\alpha^{2\cdot6} = \alpha^{10}\cdot\alpha^6 + \alpha^5\cdot\alpha^{12} = \alpha^5$
$j = 7 \Rightarrow \sigma_1\alpha^{1\cdot7} + \sigma_2\alpha^{2\cdot7} = \alpha^{10}\cdot\alpha^7 + \alpha^5\cdot\alpha^{14} = \alpha^{10}$
$j = 8 \Rightarrow \sigma_1\alpha^{1\cdot8} + \sigma_2\alpha^{2\cdot8} = \alpha^{10}\cdot\alpha^8 + \alpha^5\cdot\alpha^{16} = \alpha^2$
$j = 9 \Rightarrow \sigma_1\alpha^{1\cdot9} + \sigma_2\alpha^{2\cdot9} = \alpha^{10}\cdot\alpha^9 + \alpha^5\cdot\alpha^{18} = \alpha^5$
$j = 10 \Rightarrow \sigma_1\alpha^{1\cdot10} + \sigma_2\alpha^{2\cdot10} = \alpha^{10}\cdot\alpha^{10} + \alpha^5\cdot\alpha^{20} = 1$ deci este eronat simbolul n - j = 15 – 10 = 5 (r_5)
$j = 11 \Rightarrow \sigma_1\alpha^{1\cdot11} + \sigma_2\alpha^{2\cdot11} = \alpha^{10}\cdot\alpha^{11} + \alpha^5\cdot\alpha^{22} = \alpha^4$
$j = 12 \Rightarrow \sigma_1\alpha^{1\cdot12} + \sigma_2\alpha^{2\cdot12} = \alpha^{10}\cdot\alpha^{12} + \alpha^5\cdot\alpha^{24} = \alpha$
$j = 13 \Rightarrow \sigma_1\alpha^{1\cdot13} + \sigma_2\alpha^{2\cdot13} = \alpha^{10}\cdot\alpha^{13} + \alpha^5\cdot\alpha^{26} = \alpha^{10}$
$j = 14 \Rightarrow \sigma_1\alpha^{1\cdot14} + \sigma_2\alpha^{2\cdot14} = \alpha^{10}\cdot\alpha^{14} + \alpha^5\cdot\alpha^{28} = \alpha$
$j = 15 \Rightarrow \sigma_1\alpha^{1\cdot15} + \sigma_2\alpha^{2\cdot15} = \alpha^{10}\cdot\alpha^{15} + \alpha^5\cdot\alpha^{30} = 1$ deci este eronat simbolul n - j = 15 – 15 = 0 (r_0)

Cuvântul decodat va fi deci 11010101***1***1001***0*** .
Secvenţa de informaţie de la ieşirea decodorului va fi i = 1101010.

Desfăşurarea lucrării:
a) Rulaţi programul pe calculator, utilizând opţiunea demonstrativă atât pentru algoritmul de codare cât şi pentru cel de decodare. Se urmăreşte pas cu pas algoritmul de codare existând posibilitatea de a alege un cod corector de t = 1, 2, sau 3 erori. Urmăriţi cu atenţie la fiecare pas mesajele calculatorului care va vor ghida în rularea corectă a programului. De asemenea se urmăreşte pas cu pas algoritmul de decodare.
b) Realizaţi o codare a unui cuvânt în cazul unui cod corector de t = 1, 2, sau 3 erori alegând şirul de biţi de informaţie de lungimea corespunzătoare. Pentru cazul t = 2 efectuaţi apoi decodarea unui cuvântului de cod. Verificaţi apoi calculele cu ajutorul programului de pe calculator.
c) La sfârşitul lucrării de laborator se va efectua cu ajutorul programului un test asupra cunoştinţelor acumulate. Testul cuprinde 5 întrebări teoretice fiecare cu un răspuns corect din cinci propuse, o codare şi o decodare a unui mesaj pe care calculatorul îl generează în mod aleator.

3.2 Codul Reed-Solomon

Codurile Reed-Solomon (RS) fac parte din categoria codurilor ciclice, însă sunt coduri nebinare. Spre deosebire de celelalte coduri ciclice, alfabetul codului RS nu este câmpul binar {0, 1}, ci un câmp finit de ordin superior, numit câmp Galois şi care va fi descris în Anexa. În acest fel, cuvintele codului RS nu sunt secvenţe (succesiuni) de biţi, ci de caractere. Aceste caractere pot fi reprezentate, la rândul lor, prin secvenţe binare, însă sunt indivizibile din punct de vedere al codării şi decodării Reed-Solomon.

Structural, cuvintele de cod RS au aceeaşi alcătuire ca şi cele de cod ciclic:

$$v = v_{n-1}v_{n-2} \dots v_1v_0 \qquad v_j \in GF(2^q, p(x)) \quad j = 0 \div n-1 \tag{3.2.1}$$

unde: v – cuvântul de cod, format din n caractere;
$v_{n-1}v_{n-2} \dots v_k$ – caracterele de informaţie, în număr de k;
$v_{k-1}v_{k-2} \dots v_0$ – caracterele de control, în număr de m;
q – ordinul câmpului;
p(x) – polinomul generator al câmpului GF.
Relaţia de codare are aceeaşi formă ca şi la codurile ciclice:

$$v(x) = i(x)\cdot x^m + \text{rest}\,(i(x)\cdot x^m / g(x)) \tag{3.2.2}$$

unde g(x) este polinomul generator al codului, al cărui construcţie este prezentată mai jos, iar

$$i(x) = v_{n-1}\cdot x^{k-1} + v_{n-2}\cdot x^{k-2} + \dots + v_{k+1}\cdot x + v_k \tag{3.2.3}$$

este polinomul de informaţie.

Prin relaţia de codare (3.2.2) se obţine polinomul ataşat cuvântului de cod, polinom ai cărui coeficienţi sunt tocmai caracterele ce alcătuiesc cuvântul de cod dat de (3.2.1). Relaţia (3.2.2) indică, deasemenea, că v(x) este un multiplu al lui g(x).

Codul RS, având parametrii n, k şi m, construit după relaţia (3.2.2), este capabil să corecteze un număr e_c de caractere eronate, unde:

$$2\cdot e_c = m = n-k \tag{3.2.4}$$

La decodare, spre deosebire de codurile ciclice, într-un cuvânt de cod RS recepţionat, în vederea corecţiei, este necesară atât localizarea erorii, cât şi stabilirea valorii ei.

Polinomul generator, g(x), al codului

Pentru a se corecta t erori dintr-un cuvânt este necesar a se preciza poziţia fiecăreia precum şi valoarea ei. Dacă ne referim la un cuvânt de lungime n, unde:

$$n = 2^q - 1 \tag{3.2.5}$$

atunci informaţia necesară pentru a preciza un caracter eronat între cele n este:

$$i_{p1} = -\log_2(1/n) \tag{3.2.6}$$

Pentru a include şi varianta "cuvânt fără eroare", considerăm că în acest caz se "eronează" un al n+1–lea caracter, fictiv, astfel încât informaţia necesară pentru a preciza poziţia unui caracter eronat (dintre n+1 caractere) este:

$$i_{p1} = \log_2(n+1) \tag{3.2.7}$$

Această informaţie poate fi conţinută de un caracter din $GF(2^q)$. Deasemenea şi valoarea erorii, ε, poate să fie orice caracter din $GF(2^q)$:

$$w = v + \varepsilon \tag{3.2.8}$$

unde: w –caracterul recepţionat $\in GF(2^q)$;

v –caracterul emis $\in GF(2^q)$;

ε -valoarea erorii $\in GF(2^q)\backslash\{0\}$.

Incluzând şi cazul "eronare a caracterului fictiv", rezultă că ε poate lua orice valoare din $GF(2^q)$, adică 2^q valori posibile. Informaţia necesară pentru a preciza valoarea ei este identică cu cea dată de (3.2.7).

În concluzie, pentru fiecare eroare ce se doreşte a fi corectată este necesară o informaţie egală cu 2q biţi, adică două caractere din $GF(2^q)$. La t erori sunt necesare 2t caractere (cantitate de informaţie).

Obs. *În fapt condiţia anterioară este una suficientă, cea necesară implică mai puţină informaţie deoarece nu se poate erona un caracter de două ori. De exemplu în cazul a două erori:*

$$i_{p2} = -\log_2 \frac{1}{C_{n+1}^2} = \log_2 \frac{(n+1)\cdot n}{2} = \log_2(n+1) + \log_2 n - 1 \tag{3.2.9}$$

iar pentru n suficient de mare $i_{p2} \approx 2i_{p1} - 1$.

Cele 2t caractere de informaţie necesare soluţionării problemei corecţiei se află din 2t ecuaţii, care înseamnă tot atâtea legături (proprietăţi) pentru cuvântul receptionat. Aceste 2t proprietăţi pentru cuvintele de cod RS sunt generate prin relaţia de codare (3.2.2). Prin această relaţie v(x) devine multiplul lui g(x), ceea ce înseamnă că rădăcinile lui g vor fi şi rădăcini pentru v. Rezultă necesitatea ca g să aibă 2t rădăcini. Aceste rădăcini pot fi oricare dintre cele 2^q elemente ale câmpului $GF(2^q)$. Vom alege pentru g rădăcinile $\alpha, \alpha^2, \alpha^3, \ldots \alpha^{2t}$, datorită simplităţii şi simetriei:

$$g(x) = (x+\alpha)(x+\alpha^2)\ldots(x+\alpha^{2t}) = x^m + g_{m-1}x^{m-1} + \ldots + g_1 x + g_0 \tag{3.2.10}$$

Aşadar cuvântul de cod RS, v, rezultat prin codarea cu ajutorul relaţiei (3.2.2), în care g este dat de (3.2.10), are proprietatea că elementele $\alpha, \alpha^2, \alpha^3, \ldots \alpha^{2t}$ sunt rădăcini pentru polinomul ataşat, v(x).

Codarea codului Reed– Solomon

Codarea se poate face utilizând relaţia (3.2.2).
Spre exemplu în cazul codului RS cu n = 7, k = 5, t = 1 având polinomul generator:

$$g(x) = (x + \alpha)(x + \alpha^2) = x^2 + \alpha^4 x + \alpha^3 = x^2 + 5x + 4 \qquad (3.2.11)$$

Ecuaţia împărţirii este:

$$i(x)\cdot x^2 = (2x^4 + 5x^3 + x^2 + x + 5)(x^2 + 5x + 4) + x + 1 \qquad (3.2.12)$$

unde: $-i(x)\cdot x^2$ este deîmpărţitul ($i = [2\ 1\ 7\ 5\ 4] \Rightarrow i(x) = 2x^4 + x^3 + 7x^2 + 5x + 4$);
$-2x^4 + 5x^3 + x^2 + x + 5$ este câtul;
$-g(x) = x^2 + 5x + 4$ este împărţitorul, iar
$- x + 1$ este restul.

Împărţirea polinomului $i(x)\cdot x^m = i(x)\cdot x^2$ la g(x), cerută pentru codare de relaţia (3.2.2), este prezentată mai jos:

$$\begin{array}{l|l} 2x^6 + x^5 + 7x^4 + 5x^3 + 4x^2 & x^2 + 5x + 4 \\ \underline{2x^6 + 6x^5 + 5x^4} & \hline 2x^4 + 5x^3 + x^2 + x + 5 \\ / \quad 5x^5 + 4x^4 + 5x^3 + 4x^2 & \\ \quad \underline{5x^5 + 2x^4 + x^3} & \\ \quad / \quad 2x^4 + 6x^3 + 4x^2 & \\ \quad\quad \underline{x^4 + 5x^3 + 4x^2} & \\ \quad\quad / \quad x^3 & \\ \quad\quad\quad \underline{x^3 + 5x^2 + 4x} & \\ \quad\quad\quad / \quad 5x^2 + 4x & \\ \quad\quad\quad\quad \underline{5x^2 + 2x + 1} & \\ \quad\quad\quad\quad / \quad\quad x + 1 & \end{array}$$

Cuvântul de cod, conform relaţiei (3.2.2) rezultă:

$$v(x) = i(x)\cdot x^2 + x + 1 = 2x^6 + x^5 + 7x^4 + 5x^3 + 4x^2 + x + 1 \qquad (3.2.13)$$

Decodarea codului Reed-Solomon

Decodarea codului RS poate fi făcută atât în timp cât şi în frecvenţă

Decodarea codului RS-corector de erori multiple

Decodarea va fi sistematizată sub forma unor paşi algoritmici. La fiecare pas se va prezenta operaţia necesară a fi executată, scopul urmărit cât şi argumentările necesare. Fie aşadar codul RS corector de t erori.

Pasul I

Conform relaţiei (3.2.10) g(x) este un polinom de grad $k = 2t > 2$. Prin construcţie, cuvintele de cod RS au proprietatea că polinoamele ataşate lor sunt divizibile cu g(x), adică elementele câmpului $GF(2^2)$ $\alpha, \alpha^2, \ldots, \alpha^{2t}$ sunt rădăcini atât pentru g(x) cât şi pentru orice cuvânt de cod v(x):

$$\begin{cases} g(\alpha^j) = 0 \\ v(\alpha^j) = 0 \qquad j = 1 \div 2t \end{cases} \qquad (3.2.14)$$

Aceste proprietăţi constituie şi punctul de plecare în decodare. Presupunând că w este un cuvânt recepţionat:

$$w = v + \varepsilon \quad \text{sau} \quad w(x) = v(x) + \varepsilon(x) \tag{3.2.15}$$

vom calcula 2t coeficienţi, numiţi coeficienţi sindrom, Sj, în forma:

$$S_j = w(\alpha^j) = v(\alpha^j) + \varepsilon(\alpha^j) = \varepsilon(\alpha^j) \qquad j=1\div 2t \tag{3.2.16}$$

Evident că dacă nu există erori $S_j = 0$. În acest caz se trece la pasul VI. Evident concluzia poate fi eronată. Un exemplu în argumentarea acestei afirmaţii este situaţia: ε = cuvânt de cod. Dar în acest caz numărul erorilor depăşeşte puterea de corecţie de t erori.

Pasul II

Dacă există erori în limitele corectabile (numărul erorilor este mai mic sau egal cu t) atunci există coeficienţi sindrom diferiţi de zero. Fie cuvântul eroare în forma:

$$\varepsilon(x) = \sum_{i=1}^{t} {}^{r_i} \cdot x^{k_i} \tag{3.2.17}$$

unde:

$$Y_i = \alpha^{r_i} \tag{3.2.18}$$

reprezintă valoarea erorii $r_i \in \{0, 1, 2, \ldots n-1\}$, iar:

$$X_i = \alpha^{k_i} \tag{3.2.19}$$

reprezintă locatorul erorii $k_i \in \{0, 1, 2, \ldots, n-1\}$. Cu aceste notaţii coeficienţii sindrom au expresiile:

$$S_j = \sum_{i=1}^{t} Y_i \cdot \left(\alpha^j\right)^{k_i} = \sum_{i=1}^{t} Y_i \cdot X_i^j, \qquad j = 1\div 2t \tag{3.2.20}$$

Ecuaţiile (3.2.20) reprezintă un sistem de 2t ecuaţii cu 2t necunoscute: t locatori ai erorilor X_i şi t valori pentru respectivele erori Y_i.

Rezolvarea acestui sistem de ecuaţii se va face în mai multe etape. La pasul prezent se vor calcula coeficienţii polinomului σ(x) ai cărui rădăcini sunt locatorii erorilor:

$$\sigma(x) = \sum_{i=1}^{t} (x + X_i) = x^t + \sigma_{\ 1} \cdot x^{t-1} + \ldots + \sigma_{t-1} \cdot x + \sigma_t \tag{3.2.21}$$

Pentru că Xi , $1 \le i \le t$ este o rădăcină a lui σ(x) putem scrie:

$$X_i^t + \sigma_1 \cdot X_i^{t-1} + \ldots + \sigma_{t-1} \cdot X_i + \sigma_t = 0, \quad i = 1\div t \tag{3.2.22}$$

Înmulţind ecuaţiile (3.2.22) pe rând cu $X_i^k Y_i$ şi sumându-le obţinem ecuaţia:

$$\sum_{i=1}^{t} Y_i \cdot X_i^{t+k} + \sigma_1 \cdot \sum_{i=1}^{t} Y_i X_i^{t+k-1} + \ldots + \sigma_{t-1} \cdot \sum_{i=1}^{t} Y_i \cdot X_i^{k+1} + \\ + \sigma_t \cdot \sum_{i=1}^{t} Y_i \cdot X_i^k = 0 \tag{3.2.23}$$

sau, ţinând cont de (3.2.20) pentru k luând valorile 1, 2, ..., t:

$$S_{t+k} + \sigma_1 \cdot S_{t+k-1} + \ldots + \sigma_{t-1} \cdot S_{k-1} + \sigma_t \cdot S_k = 0 , \quad k = 1 \div t \tag{3.2.24}$$

Ecuaţiile (3.2.24) reprezintă un sistem de t ecuaţii cu t necunoscute (coeficienţii σ_i) a cărui rezolvare constituie obiectivul acestui pas algoritmic. Ecuaţiile (3.2.24) pot fi puse sub forma compactă:

$$A_s \cdot \sigma = B_s \tag{3.2.25}$$

unde:

$$A_s = \begin{bmatrix} S_t & S_{t-1} & \ldots & S_1 \\ S_{t+1} & S_t & \ldots & S_2 \\ \ldots & \ldots & \ldots & \ldots \\ S_{2t-1} & S_{2t-2} & \ldots & S_t \end{bmatrix}; \quad \sigma = \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \ldots \\ \sigma_t \end{bmatrix}; \quad B_s = \begin{bmatrix} S_{t+1} \\ S_{t+2} \\ \ldots \\ S_{2t} \end{bmatrix} \tag{3.2.26}$$

Calculând inversa matricii A_s găsim soluţia sistemului (3.2.24) în forma:

$$\sigma = A_s^{-1} \cdot B_s \tag{3.2.27}$$

Obs: *Toate calculele trebuiesc făcute în câmpul GF(2²), atât coeficienţii sindrom, S_j, cât şi coeficienţii σ fiind elemente ale respectivului câmp.*

În rezolvarea ecuaţiei (3.2.25) pot apărea trei situaţii:

1° rangul matricii As este e < t şi este egal cu al matricii [AsBs]. În acest caz numărul de erori este e şi din rezolvarea ec (3.2.25) rezultă un număr e de coeficienţi σ_l nenuli Rezolvarea ecuaţiei (3.2.25) presupune restrângerea sistemului (10.24) la un număr e < t de ecuaţii cu e necunoscute, rezolvabil.

2° rangul matricii As este t. În acest caz există As^{-1} iar ecuaţia (3.2.25) are soluţie dată prin relaţia (3.2.27).Se vor găsi t erori în acest caz.

3° rangul matricii As este e` <t şi este mai mic decât al matricii [AsBs]. O astfel de situaţie este posibil să apară dacă numărul erorilor depăşeşte t. În acest caz se semnalează prezenţa erorilor în număr necorectabil. Funcţie de aplicaţie se va abandona cuvântul în cauză sau se va cere retransmisia sa.

Pasul III

Ecuaţia (3.2.22) se poate rescrie în forma:

$$\sum_{j=1}^{t} \sigma_j \cdot X_i^{-j} = 1 \tag{3.2.28}$$

Ştiind că X_i este de forma $X_i = \alpha^{k_i}$ unde k_i indică rangul pe care îl ocupă eroarea (ex: k_i = n-1 este prima poziţie) se vor putea afla locatorii erorilor printr-o operaţie de căutare:

$$\sum_{j=1}^{t} \sigma_j \cdot \alpha^{k \cdot j} = 1 \quad ? \qquad k = 1, 2, \ldots, n \tag{3.2.29}$$

Acei k pentru care (3.2.29) este o identitate, indică prezenţa erorii pe poziţia:

$$r = n - k \tag{3.2.30}$$

Obs: *Înlocuind pe:*

$$X_r = \alpha^r \qquad (3.2.31)$$

în (3.2.28) şi utilizând identitatea $\alpha^n = 1$ *obţinem:*

$$\sum_{j=1}^{t} \sigma_j \cdot \alpha^{-j \cdot r} = \sum_{j=1}^{t} \sigma_j \cdot \alpha^{n \cdot j} \alpha^{-j \cdot r} = \sum_{j=1}^{t} \sigma_j \cdot \alpha^{(n-j) \cdot r} = \sum_{j=1}^{t} \sigma_j \cdot \alpha^{k \cdot jr} = 1$$

Pasul IV

Dispunând de poziţiile erorilor dispunem implicit de numărul lor. Reţinem că problema are soluţie doar dacă e < t. Cunoscând aşadar e locatori ai erorilor în forma $X_i = \alpha^{k_i}$, i = 1÷e, din sistemul de ecuaţii (3.2.20) se reţin e ecuaţii în vederea aflării valorilor Y_i pentru cele e erori. Acest sistem este compatibil unic determinat. Rezolvarea sa conduce la aflarea celor e valori necesare Y_i.

Pasul V

Cunoscând atât poziţiile erorilor $X_i = \alpha^i$, i = 1÷e, cât şi valorile lor $Y_i = \alpha^{r_i}$, i=1÷e putem face corecţia caracterelor eronate:

$$v_{k_i} = w_{k_i} + Y_i, \qquad i = 1 \div e \qquad (3.2.32)$$

Pasul VI

Se face selecţia caracterelor de informaţie şi livrarea lor la ieşire.

Desfăşurarea lucrării:

a) Rulaţi programul pe calculator, utilizând opţiunea demonstrativă atât pentru algoritmul de codare cât şi pentru cel de decodare. Se urmăreşte pas cu pas algoritmul de codare existând posibilitatea de a alege un cod corector de t = 1, sau 2 erori. Urmăriţi cu atenţie la fiecare pas mesajele calculatorului care va vor ghida în rularea corectă a programului. De asemenea se urmăreşte pas cu pas algoritmul de decodare.

b) Realizaţi o codare a unui cuvânt în cazul unui cod corector de t = 1, sau 2 erori alegând şirul de caractere de informaţie de lungimea corespunzătoare. Pentru cazul t = 1 şi 2 efectuaţi apoi decodarea unui cuvântului de cod. Verificaţi apoi calculele cu ajutorul programului de pe calculator.

L4. Coduri convoluţionale

Codurile convoluţionale au fost introduse în 1954 de P. Elias şi reprezintă o clasă de coduri corectoare având o mare aplicabilitate practică.

În cazul codurilor convoluţionale, fiecare bloc de *n* simboluri binare de la ieşirea codorului depinde atât de blocul de *k* simboluri binare prezent la intrarea sa, la momentul considerat, cât şi de *m* blocuri precedente. În consecinţă codurile convoluţionale introduc un efect de memorie de ordinul *m*. Cantitatea $K=m+1$ se numeşte lungimea de constrângere a codului.

Codorul este constituit dintr-un sistem de *m* registre de întârziere, fiecare având o capacitate de *k* biţi, care memorează cele *m* blocuri de *k* simboluri de informaţie, dintr-o mulţime de funcţii liniare, ce generează blocurile de *n* simboluri de la ieşire şi dintr-un convertor paralel-serie. Mărimea $R=k/n$ se numeşte randamentul codului.

Un cod convoluţional de randament *R* este o aplicaţie de la mulţimea matricilor (binare) cu un număr de *k* linii şi număr infinit de coloane către mulţimea matricilor (binare) cu un număr de *n* linii şi număr infinit de coloane, unde $n > k$:

$$\mathbf{C} : \mathbf{M}_{k\times\infty} \rightarrow \mathbf{M}_{n\times\infty} \tag{4.1}$$

Astfel, prin transformarea $\mathbf{C}$, fiecărei matrici $\mathbf{I} \in \mathbf{M}_{k\times\infty}$, de forma :

$$\mathbf{I} = \begin{bmatrix} i_{01} & i_{11} & i_{21} & \cdots & i_{j1} & \cdots \\ i_{02} & i_{12} & i_{22} & \cdots & i_{j2} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ i_{0k} & i_{1k} & i_{2k} & \cdots & i_{jk} & \cdots \end{bmatrix}, \qquad i_{js} \in \{0,1\} \;\; \forall\, \mathrm{j} = \overline{0,\infty}, \;\; \mathrm{s} = \overline{1,\mathrm{k}} \tag{4.2}$$

i se ataşează o matrice $\mathbf{V} \in \mathbf{M}_{n\times\infty}$, de forma:

$$\mathbf{V} = \begin{bmatrix} a_{01} & a_{11} & a_{21} & \cdots & a_{j1} & \cdots \\ a_{02} & a_{12} & a_{22} & \cdots & a_{j2} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{0k} & a_{1k} & a_{2k} & \cdots & a_{jk} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{0n} & a_{1n} & a_{2n} & \cdots & a_{jn} & \cdots \end{bmatrix}, \quad a_{js} \in \{0,1\} \;\; \forall\, \mathrm{j} = \overline{0,\infty}, \;\; \mathrm{s} = \overline{1,\mathrm{n}} \tag{4.3}$$

Matricea **I** conţine practic biţii de informaţie, în ordinea $i_{01}\, i_{02} \ldots \mathrm{i}_{0k}\, \mathrm{i}_{11} \ldots$, iar matricea **V** secvenţa codată : $a_{01}\, a_{02} \ldots \mathrm{a}_{0n}\, \mathrm{a}_{11} \ldots$. Dacă $a_{js} \equiv i_{js}$ pentru orice *j* pozitiv şi pentru orice $s = 1 \div k$, atunci codul se numeşte sistematic. Făcând apel la o reprezentare polinomială, matricile **I** şi **V** se pot scrie ca :

$$\mathbf{I}(\mathrm{D}) = \begin{bmatrix} \sum_{s=0}^{\infty} \mathrm{i}_{s1} \mathrm{D}^s \\ \sum_{s=0}^{\infty} \mathrm{i}_{s2} \mathrm{D}^s \\ \vdots \\ \sum_{s=0}^{\infty} \mathrm{i}_{sk} \mathrm{D}^s \end{bmatrix}, \quad \mathbf{V}(\mathrm{D}) = \begin{bmatrix} \sum_{s=0}^{\infty} \mathrm{a}_{s1} \mathrm{D}^s \\ \sum_{s=0}^{\infty} \mathrm{a}_{s2} \mathrm{D}^s \\ \vdots \\ \sum_{s=0}^{\infty} \mathrm{a}_{sk} \mathrm{D}^s \\ \vdots \\ \sum_{s=0}^{\infty} \mathrm{a}_{sn} \mathrm{D}^s \end{bmatrix} \tag{4.4}$$

Cu aceste notaţii, relaţia de codare, poate fi scrisă astfel :

$$\mathbf{V}(\mathrm{D}) = \mathbf{G}(\mathrm{D}) \cdot \mathbf{I}(\mathrm{D}) \tag{4.5}$$

unde **G**(D) se numeşte matricea generatoare a codului şi are forma:

$$\mathbf{G}(\mathrm{D}) = \begin{bmatrix} g_{11}(D) & g_{12}(D) & \cdots & g_{1k}(D) \\ \cdots & \cdots & \cdots & \cdots \\ g_{n1}(D) & g_{n1}(D) & \cdots & g_{nk}(D) \end{bmatrix} \tag{4.6}$$

În funcţie de felul polinoamelor gebneratoare $g_{js}(D)$, codurile convoluţionale pot fi clasificate ca şi:

a) Coduri *sistematice* sau *nesistematice*. Dacă primele k linii din **G**(D) formează matricea unitate de ordinul k, I_k, atunci codurile sunt sistematice. În acest caz biţii din **I** se vor regăsi printre biţii din **V**. Astfel, dacă cele k simboluri de informaţie, prezente sunt efectiv emise, adică se găsesc în mod explicit în blocul de n simboluri de la ieşirea codorului pe primele k poziţii, codul se numeşte cod sistematic. Pentru codurile nesistematice, biţii din **V** sunt combinaţii liniare ale biţilor din **I**, neexistând biţi de informaţie şi de control ca şi în cazul precedent.

b) Coduri *recursive* sau *nerecursive*. Dacă toate polinoamele generatoare care compun **G**(D) sunt finite, atunci codul rezultat este nerecursiv. În caz contrar polinoamele generatoare $g_{js}(D)$ pot fi scrise sub forma:

$$g_{js}(D) = \frac{a_{js}(D)}{b_{js}(D)} \tag{4.7}$$

unde polinoamele a_{js} şi b_{js} sunt finite. Deci, dacă există cel puţin un polinom $b_{js}(D) \neq 1$, atunci codul este recursiv.

Lungimea de constrângere este unul dintre parametri importanţi ai codurilor convoluţionale. O altă definiţie a sa este dată în relaţia de mai jos:

$$K = 1 + \max_{j,s} grad\{a_{j,s}(D), b_{j,s}(D)\} \tag{4.8}$$

Pentru a ilustra codurile convoluţionale nerecursive şi nesistematice, în Fig. 4.1. se prezintă un exemplu de codor convoluţional de randament R=1/2 şi de lungime de constrângere K=(m+1)=3. Intrarea sa este constituită din blocurile de k=1 simbol şi ieşirea sa de blocurile de n=2 simboluri.

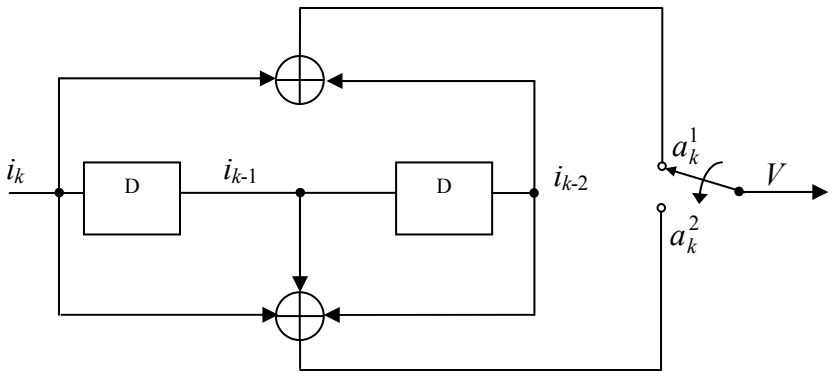


Fig. 4.1. Exemplu de codor convoluţional nerecursiv, nesistematic (R = 1/2, K = 3).

Relaţiile de calcul ale secvenţelor de la ieşire sunt:

$$a_k^1 = \sum_{j=0}^{2} i_{k-j} g_j^1 \ , \ a_k^2 = \sum_{j=0}^{2} i_{k-j} g_j^2 \tag{4.9}$$

Cele două secvenţe generatoare sunt:

$$\begin{aligned} g^1 &= \left[g_0^1, g_1^1, g_2^1\right] = [1,0,1] \\ g^2 &= \left[g_0^2, g_1^2, g_2^2\right] = [1,1,1] \end{aligned} \tag{4.10}$$

Observăm că ieşirile codorului fiind egale cu o combinaţie liniară a simbolurilor de informaţie, codul este liniar. Codurile convoluţionale sunt de asemenea definite pornind de la polinoamele lor generatoare exprimate în funcţie de variabila D (întârziere) echivalentă cu variabila Z^{-1} a transformatei Z. Considerând tot exemplul din Fig. 4.1, polinoamele generatoare ale acestui cod au expresiile:

$$\begin{aligned} g^1 &\rightarrow G^1(D) = g_0^1 + g_1^1 D + g_2^1 D^2 \\ g^2 &\rightarrow G^2(D) = g_0^2 + g_1^2 D + g_2^2 D^2 \end{aligned} \tag{2.11}$$

şi:

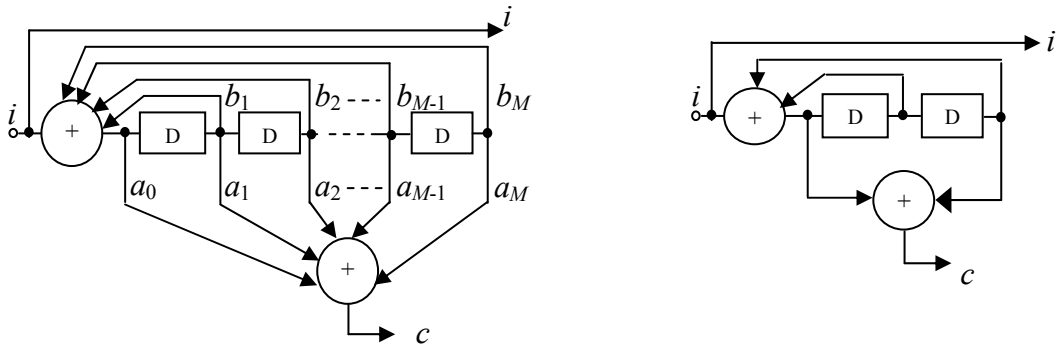
$$\begin{aligned} G^1(D) &= 1 + D^2 \\ G^2(D) &= 1 + D + D^2 \end{aligned} \tag{2.12}$$

rezultând matricea generatoare $G=[1+D^2,\ 1+D+D^2]$.

În general polinoamele generatoare ale codorului se exprimă în octal şi astfel, pentru cazul din Fig. 4.1, avem:

$$\begin{aligned} G^1 &= [1,0,1] = 5(\text{în octal}) \\ G^2 &= [1,1,1] = 7(\text{în octal}) \end{aligned} \tag{2.13}$$

În Fig.4.2 a) se prezintă schema generală a unui codor recursiv sistematic, RSC (*Recursive Systematic Code*), iar în figura Fig. 4.2 b) un caz particular al acesteia.



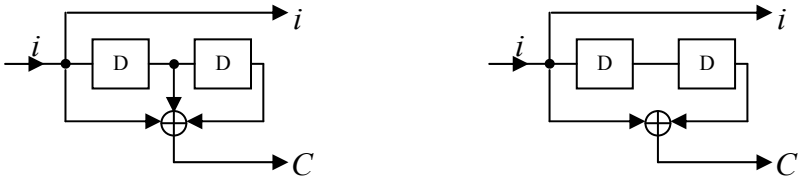
a) b)

Fig. 4.2. Cod convoluţional recursiv sistematic: a) schema generală, b) exemplu (R=1/2, K=3).

În exemplul considerat matricea generatoare este de forma:

$$G = \left[1, \frac{1+D^2}{1+D+D^2}\right] \tag{2.14}$$

În figura următoare sunt prezentate două exemple de codoare sistematice şi nerecursive, NRSC (*Non-Recursive Systematic Code*), considerându-se randamentul R=1/2 şi lungimea de constrângere K=3:



a) b)

Fig. 4.3 Coduri convoluţionale sistematic şi nerecursiv, R=1/2, K=3, G=1+D+D^2.

Diagrame de stări

Diagrama de stări este o reprezentare a funcţionării unui codor convoluţional, în care timpul nu apare în mod explicit.

În Fig.4.4 s-a reprezentat diagrama de stări asociată codorului convoluţional din Fig. 4.1. Diagrama de stări permite evaluarea funcţiei de transfer a codorului, care va fi utilizată pentru calculul performanţelor codului.

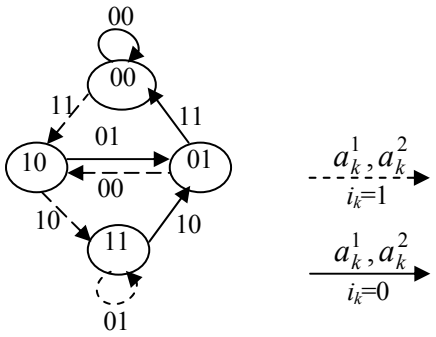


Fig. 4.4: Diagrama de stări a codorului din Fig. 4.1.

Diagramele de stări corespunzătoare codurilor convoluţionale din Fig.4.2 b) şi Fig.4.3 a) şi b) sunt prezentate în Fig.4.5.

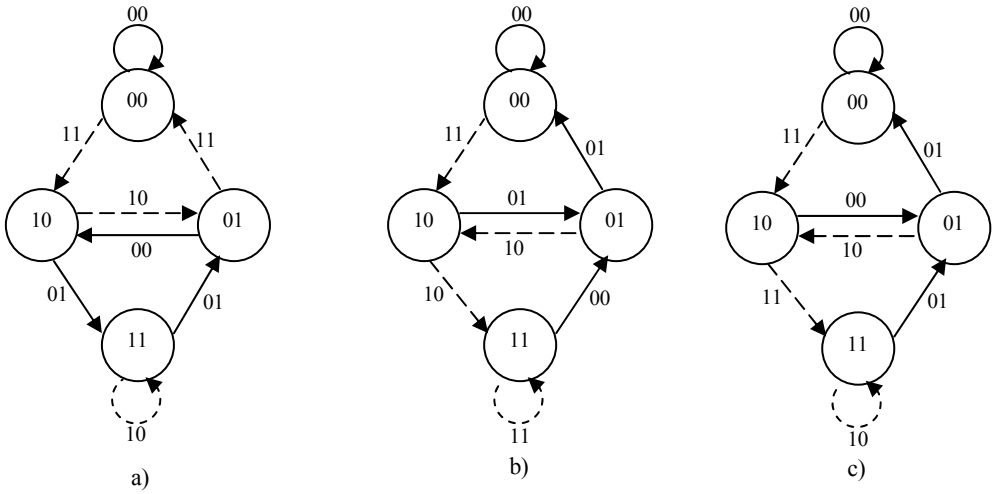


Fig. 4.5 Diagramele de stare pentru codoarele convoluţionale: a) RSC; b) şi c) NRSC.

Diagrama trellis

Fiecare bloc de $n = 2$ simboluri de la ieşirea acestui codor depinde de blocul de $k = 1$ simbol prezent la intrarea sa dar şi de $m = 2$ blocuri de k simboluri conţinute în memoria sa. Aceste $mk = 2$ simboluri definesc starea codorului. Se notează cu $S_0 = (00)$, $S_1 = (01)$, $S_2 = (10)$ şi $S_3 = (11)$, cele patru stări posibile ale codorului din Fig. 4.1. Oricare ar fi starea iniţială a codorului, după $m+1=3$ întârzieri la intrarea codorului, toate stările au fost atinse.

Funcţionarea codorului poate fi explicată ţinând seama doar de stările sale şi de tranziţiile dintre acestea, numite ramuri (ramificaţii, braţe). Diagrama *trellis* astfel obţinută este reprezentată în Fig. 4.6, pentru codorul convoluţional din Fig. 4.1, presupunând ipoteza că starea sa iniţială era $S_0 = (00)$.

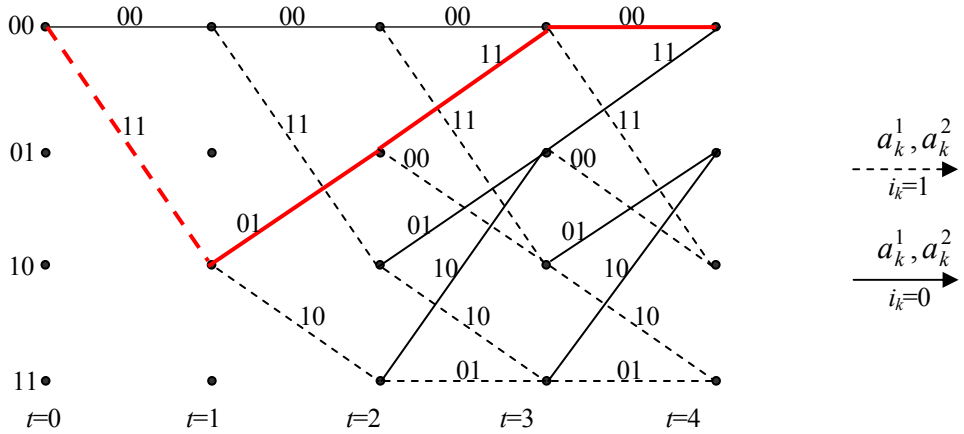


Fig. 4.6. *Trellis*-ul codorului convoluţional din Fig. 4.1.

Ramurile reprezentate prin linii punctate corespund prezenţei unui simbol de informaţie egal cu 1, la intrarea codorului, şi ramurile reprezentate prin linii pline, unui simbol de informaţie egal cu 0. Fiecărei ramuri i s-a asociat valoarea cuplului binar disponibil la ieşirea codorului.

După $m+1$ întârzieri, oricare ar fi starea iniţială a codorului, *trellis*-ul se repetă. Din fiecare nod pleacă 2^k ramuri (în cazul de faţă sunt două ramuri) şi în fiecare nod converg 2^k ramuri.

Pornind de la starea $S_0=(00)$ în momentul $t=0$, de exemplu, vedem că există patru căi care permit atingerea stării $S_0=(00)$ în momentul $t=4$.

00 00 00 00 → calea 1
00 11 01 11 → calea 2
11 10 10 11 → calea 3
11 01 11 00 → calea 4

Codarea codului convoluţional

Folosind o codare convoluţională, se codează următoarea secvenţă de informaţie: i=100111, considerând polinomul generator al codului: $g(D)=1+D^2$.
În acest scop se foloseşte Fig. 4.7.

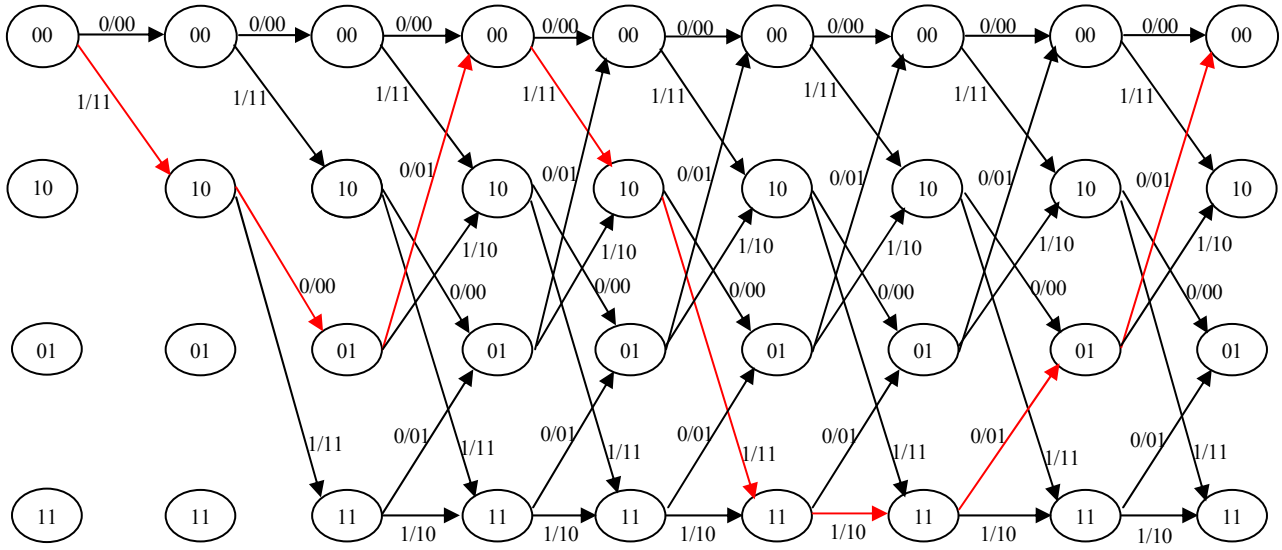


Fig. 4.7. Utilizarea diagramei trellis în cazul codării secvenţei de informaţie i=100111, considerând diagrama de stări din Fig. 4.5. c).

Cuvântul de cod obţinut este v=11 00 01 11 11 10 01 01. Trellis-ul codorului convoluţional se închide la zero, astfel, datorită acestui fapt au apărut două grupe suplimentare de biţi.

Calea pe trellis care ne generează cuvântul de cod, v, este cea marcată cu roşu.

Decodarea codului convoluţional, algoritmul Viterbi cu decizie hard

Pentru prezentarea acestui algoritm se consideră un canal binar simetric (fără memorie), intrarea decodorului fiind alcătuită dintr-o secvenţă de simboluri binare.

În fiecare moment două ramuri, aparţinând la două căi diferite, converg spre fiecare nod al *trellis*-lui (Fig.4.6). Din aceste două căi una este mai probabilă, altfel spus, se găseşte la cea mai mică distanţă Hamming faţă de secvenţa recepţionată, decât cealaltă cale. Distanţa fiind o funcţională aditivă, în fiecare nod se păstrează calea cea mai probabilă numită cale supravieţuitoare. Dacă se obţin două căi cu aceeaşi distanţă Hamming, doar o singură cale este pastrată, alegându-se în mod arbitrar una din cele două căi posibile.

În figura următoare se prezintă un exemplu de decodare pentru codorul reprezentat în Fig. 4.3. b), cu diagrama de stări reprezentată în Fig. 4.5. c). Calea rezultantă este marcată cu culoare roşie.

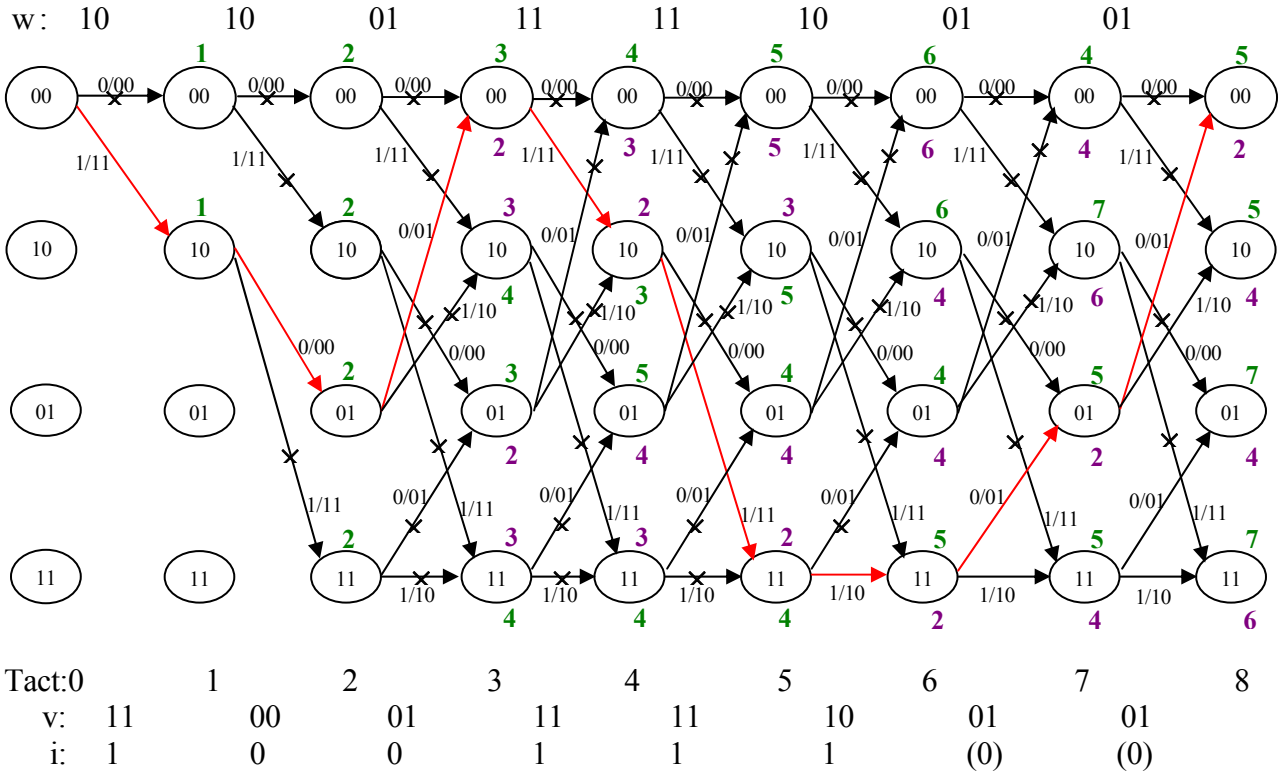


Fig. 4.8. Utilizarea diagramei trellis în cazul decodării secvenţei w=A7E5$_H$, considerând diagrama de stări din Fig. 4.5. c).

Desfăşurarea lucrării:

a) Rulaţi programul pe calculator, utilizând opţiunea demonstrativă atât pentru algoritmul de codare cât şi pentru cel de decodare.

b) Realizaţi o codare şi o decodare, folosind algoritmul de decodare Viterbi. Verificaţi apoi calculele cu ajutorul programului de pe calculator.

Bibliografie

[1] P. Elias, *"Error-free Coding"*, IRE Trans. Inform. Theory, vol IT-4, pp.29-37, 1954,

[2] Chien, R, "*Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem codes",* IEEE Transactions on Information Theory, Volume 10, Issue 4, Page(s): 357 - 363, Oct 1964,

[3] A. Glavieux, M. Joindot, *"Communications numériques. Introduction"*, Masson, Paris, 1996,

[4] M. E. Borda, *"Teoria transmiterii informaţiei"*, Editura Dacia, Cluj-Napoca, 1999,

[5] G. Wade, *"Coding Techniques-An Introduction to Compresion and Error Control"*, Creative Print and Design, Ebbw Vale, Geat Britain, 2000,

[6] J. Proakis, *"Digital Communications"*, 4th Ed., McGraw Hill, New York, 2000,

[7] H. Baltă, M. Kovaci, *"A Comparasion Between Weight Spectrum of Different Convolutional Code Types"*, conferinţă Oradea, 2004.

TEORIA INFORMAŢIEI ŞI A CODĂRII

Îndrumător de lucrări la laborator

Horia BALTA Maria KOVACI

2009

Cuprins

L1. Algoritmi pentru codarea sursei şi compresie

1.1. Codarea binară a surselor de informaţie

Lucrarea de faţă îşi propune explicitarea algoritmilor de calcul pentru a coda binar surse de informaţie întâlnite în practică.

Codarea binară a surselor de informaţie are dublu rol, de mărire a eficienţei sursei de informaţie (şi implicit de micşorare a costului transmisiei prin scăderea timpului transmisiei); de adaptare a sursei de informaţie la canalul transmisiune (canal binar, în majoritate).

Din punct de vedere al codării, la o sursă de informaţie interesează numărul de simboluri N şi probabilităţile lor de apariţie p_i, $i = \overline{1, N}$. Doar pe utilizator îl interesează ce reprezintă fiecare mesaj în parte (literă de alfabet, nivel al intensităţii pixelilor dintr-o imagine, valori ale unor mărimi fizice ce variază arbitrar[1]).

Codarea este operaţia prin care fiecărui simbol al sursei, s_i, i se alocă o succesiune binară unică:

$$s_i \longleftarrow r_i^1,\ r_i^2,\ \ldots r_i^{li} = c_i \qquad (1.1.1)$$

unde c_i reprezintă cuvântul de cod <u>asociat</u> simbolului (mesajului) sursei, iar r_i^j, cu $i = \overline{1, N}$, biţii componenţi ai cuvântului de cod (pot lua valori binare ”1” sau ”0”). Suportul fizic al valorilor binare este, de asemenea, mai puţin important pentru codare. Acest suport poate fi un semnal electric, optic, acustic, etc.

În expresia (1.1.1) l_i reprezintă lungimea cuvântului de cod c_i (număr de simboluri binare). Lungimea medie a cuvintelor de cod este dată de relaţia:

$$L = \sum_{i=1}^{N} p_i \cdot l_i \qquad (1.1.2)$$

Ieşirea codorului, pentru canalul de transmisie, reprezintă o sursă de informaţie numită secundară. Această sursă poate genera simbolurile 0 şi 1. Entropia (informaţia medie pe simbol), entropia maximă, precum şi eficienţa sursei primare, sunt date de relaţiile:

$$\begin{aligned} H(S) &= \sum_{i=1}^{N} p_i \cdot \log_2 \frac{1}{p_i}\ ; \\ H_{max}(S) &= \log_2 N\ ; \\ \eta_S &= \frac{H(S)}{H_{max}(S)}\ . \end{aligned} \qquad (1.1.3)$$

Entropia sursei secundare este egală cu informaţia medie pe un cuvânt raportată la lungimea medie a cuvântului:

$$H(X) = \frac{H(S)}{L} \qquad (1.1.4)$$

Atribuirea literelor de alfabet r_i^j pentru a forma cuvântul de cod c_i trebuie să conducă la îndeplinirea condiţiilor:

- codul să fie instantaneu (nici un cuvânt nu este prefixul altuia);

[1] Deşi, aparent, astfel de surse de informaţie nu au un număr finit de simboluri, prin eşantionare şi cuantizare, orice sursă de informaţie poate fi redusă la una cu număr finit de simboluri.

- codarea să conducă la o eficienţă maxim posibilă. Această condiţie se realizează folosind cuvinte de lungime variabilă, mai mare pentru simboluri de probabilitate de emisie (a sursei primare) mai mică. Codarea cu cuvinte de lungime variabilă este mai complicată, dar conduce la o sursă secundară cu eficienţă mai bună şi la o ieftinire a transmisiei, lungimea medie a cuvintelor de cod fiind mai mică. Codarea cu cuvinte de lungime constantă este mai simplă dar cuvintele având aceeaşi lungime oferă maleabilitate la prelucrări ulterioare.

Codarea cu cuvinte de lungime constantă se mai numeşte şi codare cu cod bloc. Cuvintele codului bloc sunt secvenţe binare diferite, de aceeaşi lungime L. Ştiind că numărul de secvenţe diferite ce pot fi constituite cu L cifre binare este 2^L, rezultă că numărul simbolurilor sursei trebuie să fie mai mic decât 2^L, sau:

$$L \geq \log_2 N = H_{max}(S) > L - 1 \qquad (1.1.5)$$

Inecuaţia a doua din (1.1.5) se datorează criteriului de eficienţă minim posibilă (sub restricţia de cod bloc).

Realizarea condiţiilor mai sus precizate se face prin codare după algoritmul Huffman (static). Acesta presupune:

1. Ordonarea probabilităţilor sursei (primare) în ordine descrescătoare;
2. Simbolurile cu probabilităţile cele mai mici (ultimele două[1] în ordonare) se reunesc formând un nou simbol cu probabilitatea sumei celor două, după care se reordonează probabilităţile, obţinându-se o sursă cu N-1 simboluri;
3. Procesul continuă până când rămân 2 simboluri. Acestora li se atribuie valorile 0 şi 1;
4. Mergând pe cale inversă, se disociază simbolurile compuse în simboluri din care s-au compus, atribuind arbitrar, la fiecare disociere, valorile 0 şi 1 pentru a găsi cuvintele de cod pentru cei doi componenţi. De reţinut că doar un singur simbol se disociază la fiecare pas, lungimea componenţilor săi crescând cu 1, celelalte simboluri păstrându-şi lungimea şi structura cuvântului de cod.

***Observaţie**: -atribuirea simbolurilor binare celor două simboluri compuse este arbitrară, codurile obţinute vor fi diferite, dar de aceeaşi eficienţă. De asemenea, ordinea simbolurilor de egală probabilitate poate fi aleasă arbitrar, diferitele moduri de atribuire conducând la coduri diferite, dar de aceeaşi eficienţă. Însă, în vederea posibilităţii de a confrunta rezultatele, se vor respecta regulile: -„0" se atribuie totdeauna simbolului de jos (ultimul în ordonare); -ordinea simbolurilor egal probabile este ordinea în care s-au citit, cu simbolul compus totdeauna ultimul.*

Programul pe calculator, asociat acestei lucrări, permite codarea surselor de informaţie prin algoritm Huffman static.

Precizarea sursei de informaţie se poate face simplu, introducând valorile probabilităţilor sursei şi numărul lor. Probabilităţile trebuie să îndeplinească condiţia:

$$\sum_{i=1}^{N} p_i = 1 \quad ; \quad 0 \leq p_i \leq 1 ; \quad \forall\ i = \overline{1, N} \qquad (1.1.6)$$

Probabilităţile pot fi introduse indirect, prin numere întregi, pozitive k_i, calculatorul urmând să facă transformarea:

$$p_i = \frac{k_i}{\sum_{k=1}^{N} k_i} \qquad (1.1.7)$$

[1] Algoritmul se poate aplica şi pentru obţinerea codurilor nebinare (având un alfabet cu m simboluri). În acest caz se grupează câte m simboluri.

ceea ce asigură îndeplinirea relaţiei (1.1.6).

Sursa de informaţie poate fi şi un text scris (în caractere ASCII), cu cel puţin 3 caractere diferite. Calculatorul va înţelege că simbolurile sursei (caracterele prezente în text) au probabilităţi precizate prin ponderea lor în text (număr de prezenţe ale caracterului respectiv raportat la întregul număr de caractere ale textului).

Programul permite şi o a treia sursă de informaţie. Simbolurile acesteia reprezintă intervale de lungime egală, şi în număr finit, de pe axă reală. În acest caz datele vor consta în valorile eşantioanelor unui semnal discret (Fig.1.1.1) , mărginit în timp cu suportul finit precizat ca dată de intrare M. Tot ca dată de intrare calculatorul va cere şi q-cuanta. Având aceste mărimi q (cuanta), M (numărul de eşantioane) şi b(i) (valorile eşantioanelor), pentru găsirea simbolurilor sursei şi implicit a numărului lor – N, calculatorul va proceda astfel:

- va căuta eşantioanele de valoare minimă şi maximă : min; max;
- va calcula $N = [\frac{\max - \min}{2}] + 1$ unde [ ] semnifică partea întreagă;

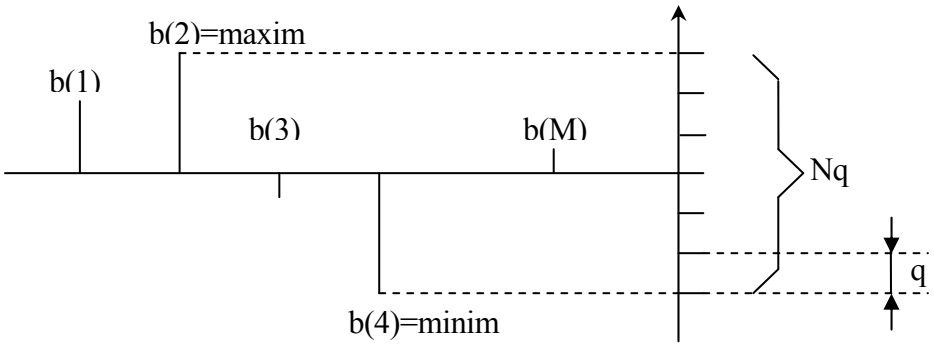


Fig.1.1.1 Sursă de informaţie „semnal discret”

- va calcula $N = [\frac{\max - \min}{q}] + 1$ unde [ ] semnifică partea întreagă;
- va atribui sursei de informaţie primare simbolurile: [min, min + q); [min+q, min+2q); ... ;[min+(N-1)q,min+Nq)
- va calcula probabilitatea simbolurilor ca fiind egale cu fracţiunea din numărul eşantioanelor M, ce au valori cuprinse în intervalele ce definesc simbolurile. Este posibil desigur să rămână simboluri cu probabilitate 0.

În toate cele 3 cazuri se vor calcula şi se vor putea vizualiza:

a) probabilităţile sursei primare;
b) datele asociate sursei şi codării:
   - entropia sursei primare H(S);
   - entropia maximă a sursei primare $H_{max}(S)$;
   - eficienţa sursei primare η_S;
   - entropia sursei secundare (eficienţa codării) H(X);
   - lungimea medie a cuvintelor de cod L;
   - redundanţa sursei primare $R(S)=H_{max}(S)-H(S)$;
   - redundanţa relativă a sursei primare $\rho_S = 1-\eta_S$;
   - redundanţa sursei secundare $\rho_X = 1-H(X)$.
c) cuvintele de cod;
d) graful codării;

e) schema algoritmului Huffman (numai pentru cazul codării cu cuvinte de lungime variabilă);
f) în cazul sursei „semnal discret" se pot vedea şi graficul semnalului precum şi valorile eşantioanelor.

Desfăşurarea lucrării

a). Rulaţi programul cobsinf.exe selectând opţiunea „Demonstraţie". Selectaţi pe rând cele trei feluri de surse de informaţie (tablou, text şi semnal discret), şi pentru fiecare sursă solicitaţi o codare bloc şi o codare prin algoritmul Huffman static. Urmăriţi aplicarea algoritmului.
b). Alcătuiţi surse de informaţie de forma celor prezentate şi codaţi-le bloc şi prin algoritmul Huffman static, calculând în fiecare caz entropia sursei, eficienţa sursei, lungimea medie a cuvintelor codului obţinut, precum şi eficienţa codării. Porniţi pentru început cu surse tablou mai simple, apoi măriţi numărul de simboluri. Utilizaţi ulterior şi surse „text" sau „semnal discret".
c) La sfârşitul lucrării de laborator se va efectua test asupra cunoştinţelor acumulate. Durata acestuia este aproximativ constantă şi independentă de numărul de studenţi ce efectuează simultan testul (calculatorul poate testa până la 4 studenţi simultan – în sensul că afişează pe rând date pentru cei $n \leq 4$ studenţi şi apoi întreabă pe rând, în aceeaşi ordine studenţii, timpii de afişare şi de chestionare rezervaţi fiecărui student sunt constanţi şi nu influenţează pe cei rezervaţi altuia). Rezultatul testului se va concretiza printr-o notă, calculată automat de calculator.

1.2. Algoritmul Huffman dinamic

Compresia este procesul de minimizare a spaţiului ocupat sau a timpului necesar transmiterii unei anumite cantităţi de informaţie.

Metodele de compresie pot fi împărţite în:

- Metode de compresie cu pierdere;
- Metode de compresie fără pierdere.

Metodele de compresie cu pierdere de informaţie sunt folosite în special în transmiterea semnalului audio şi video, unde pierderea de informaţie are ca rezultat o scădere a calităţii sunetului, respectiv imaginii.

Compresia de date fără pierdere, prezentă în programele de arhivare, în sistemele de transmisiune a datelor, a evoluat de-a lungul timpului pornind de la algoritmi simpli (suprimarea zerourilor, codarea pe şiruri) şi ajungând la algoritmii complecşi folosiţi în prezent.

Metodele de compresie fără pierderi au la bază ideea că în general cantitatea de informaţie prelucrată (transmisă, depozitată) conţine o anumită redundanţă ce se datorează:

- Distribuţiei caracterelor (unele caractere au o frecvenţă de apariţie mult mai mare decât altele);
- Repetării consecutive a unor caractere;
- Distribuţiei grupurilor de caractere (unele grupuri sunt mult mai frecvente decât altele şi în plus există grupuri care nu apar deloc);
- Distribuţiei poziţiei (unele caractere sau grupuri ocupă poziţii preferenţiale, predictibile în anumite blocuri de date).

Având în vedere toate aceste tipuri se poate înţelege de ce o anumită tehnică de compresie poate da un rezultat bun pentru un anumit tip de surse, pentru altele însă rezultatul fiind dezastruos. În studiul compresiei se urmăreşte obţinerea unui algoritm care să ofere o compresie cât mai bună pentru tipuri de surse cât mai diferite.

Aprecierea cantitativă a compresiei realizate se face utilizând *factorul de compresie*, definit ca:

$$F_c = \frac{n_u}{n_c} \qquad (1.2.1)$$

unde n_u este lungimea în biţi a mesajului iniţial şi n_c lungimea de compresie.

Algoritmul Huffman dinamic

Algoritmii de tip Huffman static au dezavantajul că necesită cunoaşterea prealabilă a statisticii sursei. Acest dezavantaj poate fi înlăturat utilizând un algoritm dinamic.

Algoritmul Huffman dinamic este un algoritm de compresie. Mesajele au fost deja codate în prealabil, dar neeficient, cu un cod bloc, de lungime *n* biţi/cuvânt. Ideea de bază în această codare este folosirea pentru codarea unui simbol s_{i+1} din mesaj a unui graf de codare, ce este un arbore care creşte, dintr-un punct iniţial (numit sursă) şi care este construit pe baza primilor *i* simboluri din mesaj. După transmiterea simbolului s_{i+1} se va revizui arborele de codare în vederea codării simbolului s_{i+2}.

Codarea presupune la fiecare pas transmiterea codului aferent simbolului de intrare şi modificarea corespunzătoare a grafului şi a codului. Dacă în mesajul transmis este un simbol nou, adică un simbol care n-a mai apărut în mesaj, se transmite mai întâi codul frunză goală şi apoi cei *n* biţi aferenţi simbolului nou apărut. Frunza goală are semnificaţia că urmează un nou simbol. Frunza goală se codifică ca un mesaj oarecare, dar ponderea sa întodeauna va rămâne nulă.

Exemplu:
În continuare se prezintă evoluţia arborelui (grafului) asociat codului în timpul codării mesajului "M_i = aaabccc":

Obs: *Semnificaţia notaţiilor este:*

-pentru nod: x | p |

- x: număr de ordine (creşte de la stânga spre dreapta şi de jos în sus pe linii);
- p: ponderea cumulată din nodurile aferente.

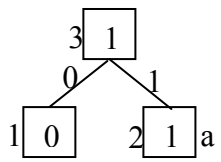
- pentru frunză (nod terminal): x | p | y

- x: număr de ordine;
- p: pondere (număr de apariţii ale respectivului simbol);
- y: simbolul.

Frunza goală, notată cu "0", este de pondere 0.

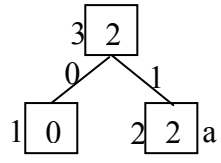
- pentru ramură: - spre stânga ≡ codare cu zero;
- spre dreapta ≡ codare (atribuire) cu unu.

1. M_i = "a"



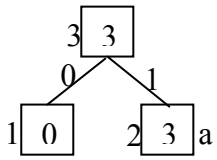
Codurile sunt: 0 - 0
a - 1
Mesajul transmis este: "a"

2. M_i = "aa"



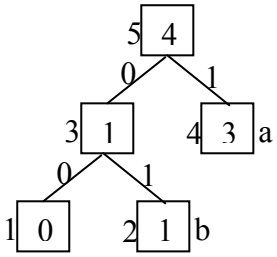
Codurile sunt: 0 - 0
a - 1
Mesajul transmis este: "a1"

3. M_i = “aaa”



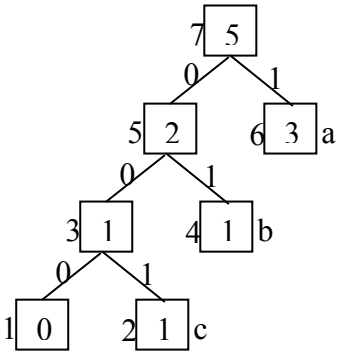
Codurile sunt: 0 - 0
a - 1
Mesajul transmis este: “a11”

4. M_i = “aaab”



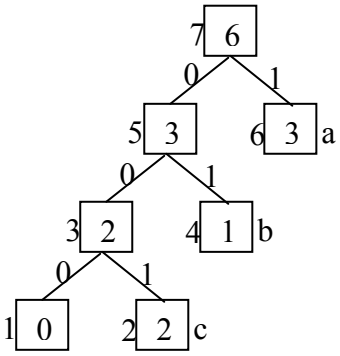
Codurile sunt: 0 - 00
a - 1
b - 01
Mesajul transmis este: “a110b”

5. M_i = “aaabc”



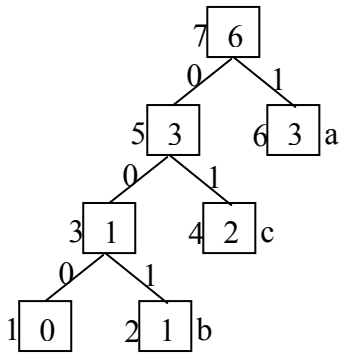
Codurile sunt: 0 - 000
a - 1
b - 01
c - 001
Mesajul transmis este: “a110b00c”

6. M_i = “aaabcc”



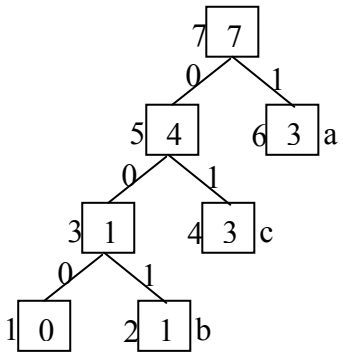
Codurile sunt: 0 - 000
a - 1
b - 01
c - 001
Mesajul transmis este: “a110b00c001”

Deoarece ponderea nodului “c” este mai mare decât ponderea nodului “b” se va face interschimbarea între noduri, realizându-se o rearanjare a arborelui:



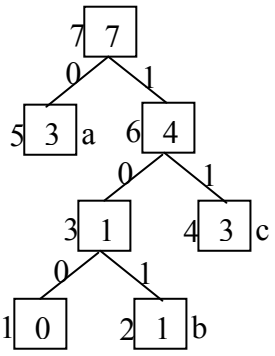
Codurile sunt: 0 - 000
a - 1
b - 001
c - 01
Mesajul transmis este: “a110b00c001”

7. M_i = “aaabccc”



Codurile sunt: 0 - 000
a - 1
b - 001
c - 01
Mesajul transmis este: “a110b00c00101”

În urma rearanjării arborelui se obţine:



Codurile sunt: 0 - 100
a - 0
b - 101
c - 11
Mesajul transmis este: “a110b00c00101”

Dacă se presupune că simbolurile mesajului de la intrare au fost codate în prealabil cu un cod bloc, cu n=8 biţi/cuvânt vom avea lungimea mesajului iniţial:
Considerând caracterele în mesajul iniţial codate pe 8 biţi, lungimea acestuia este:

$$n_u = 7 \cdot 8 = 56 \text{ biţi} \qquad (1.2.2)$$

Mesajul comprimat (a110b00c00101) are:

$$n_c = 8 \cdot 3 + 10 = 34 \text{ biţi} \qquad (1.2.3)$$

Rezultă un factor de compresie:

$$F = \frac{n_u}{n_c} \cong 1,7$$

Desfăşurarea lucrării:

a) Se lansează executabilul dHuffman.exe din directorul Huffman Dinamic, după care se introduce parola TTI. Rulaţi programul, selectând opţiunea Demonstraţie, pentru codare şi decodare.
b) Se verifică, cu programul, exemplul luat în această lucrare, selectând pe rând compresia, respectiv decompresia prezentate.
c) Pentru compresie şi decompresie se alege câte un exemplu oarecare asemănător cu cel prezentat în lucrare. Se realizează compresia/decompresia, după care, cu programul, se verifică rezultatele obţinute.
d) La sfârşitul lucrării de laborator se va efectua un test asupra cunoştinţelor acumulate. Rezultatul testului se va concretiza printr-o notă, calculată automat de calculator.

1.3. Algoritmi de compresie de tip LZ

Cu toate că algoritmii de compresie Huffman, static sau dinamic, sunt cei mai performanţi (din punct de vedere al factorului de compresie) relativ la sursele fără memorie, această concluzie nu este valabilă şi pentru sursele cu memorie, surse frecvent întâlnite în practică.

Algoritmii de compresie de tip LZ fac parte din categoria tehnicilor de dicţionar adaptive. Ele servesc (în special) la compresia fişierelor de tip text, fişiere ce se caracterizează prin repetarea frecventă a unor subşiruri.

Ideea de bază în algoritmii LZ este înlocuirea unor subşiruri ale mesajului cu cuvinte de cod, astfel încât, la o nouă apariţie a unui subşir să se transmită doar codul asociat lui.

Algoritmul LZ –77

Mesajul de intrare este trecut printr-un buffer de lungime N. Bufferul este divizat în două blocuri distincte numite Lempel şi Ziv, ca în Fig. 1.3.1.

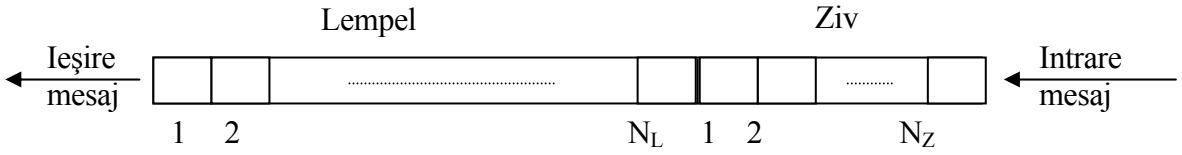


Fig. 1.3.1. Buffer–fereastră utilizat în algoritmul LZ–77

Mesajul de ieşire constă într-o succesiune de triplete de forma „P, L, A”. Compresia decurge astfel:

-se încarcă primele N_Z caractere din mesajul de intrare în blocul Ziv;

-se transmite la ieşire tripletul „0, 0, x”, unde x reprezintă primul caracter (literă) din mesajul de intrare (aflat pe poziţia 1 în blocul Ziv);

se deplasează cu o poziţie mesajul de intrare în blocul Ziv, astfel încât x ajunge acum în poziţia N_L. Din acest moment:

-se caută în blocul Lempel un subşir, S', care este identic cu cel mai lung subşir, S, care începe în poziţia 1 a blocului Ziv şi este conţinut în întregime în blocul Ziv. Subşirul S' trebuie neapărat să înceapă în blocul Lempel, dar poate să sew sfârşească în blocul Ziv. Dacă un astfel de subşir, S', există, atunci se va transmite tripletul „P, L, A”, unde P = poziţia de la care începe subşirul S'; L = lungimea subşirurilor S şi S'; A = litera ce succede subşirul S în mesajul de intrare.

Dacă nu există nici un subşir S' care să coincidă cu cel mai scurt subşir S, atunci se va transmite tripletul „0, 0, x”, unde x are aceeaşi semnificaţie ca şi mai sus;

-se deplasează mesajul de intrare până când, după caz, A sau x ajung pe poziţia N_L şi procesul de căutare se reia până când tot mesajul a trecut prin blocul Ziv.

Decompresia presupune, în principal, aceleaşi etape. Utilizând tripleţii recepţionaţi, se adaugă subşiruri în blocul Ziv mesajului deja decomprimat, subşiruri aflate deja în blocul Lempel urmând ca, mai apoi, toate să fie deplasate în blocul Lempel.

Observaţie: *este posibil să se renunţe la cel de-al doilea zero din tripletul „0, 0, x", el fiind redundant.*

Cei trei componenţi ai tripletului „P, L, A" sunt codaţi binar bloc, separat. Astfel, codul pentru P are lungimea:

$$k_P = \log_2 (N_L + 1) \qquad (1.3.1)$$

(trebuie codat şi 0 din situaţia „0, 0, x" sau „0,x"), iar pentru L:

$$k_L = \log_2 (N_Z - 1) \qquad (1.3.2)$$

considerând doar varianta „0, x". Caracterele de tipul „A" pot fi codate, de exemplu, ASCII.

Din relaţiile (1.3.1) şi (1.3.2) rezultă că, pentru o bună compresie, N_L+1 şi N_Z-1 trebuie să fie puteri întregi ale lui 2. În relaţia (1.3.2) s-a luat N_Z-1 şi nu doar N_Z deoarece, în situaţia extremă a lungimii maxime pentru subşirul S, N_Z va fi egal cu lungimea subşirului S plus unu; acest „unu" rezervă literei A un loc în blocul Ziv.

Exemplu *Fie parametrii N_L =7 şi N_Z =4, iar mesajul de intrare „aaababacacab…". Aplicarea algoritmului LZ –77 asupra acestui mesaj de intrare este ilustrată în Fig. 1.3.2 a).*
Mesajul comprimat este, aşadar: „0a72b63c73b…", sau în binar:
„000a11110b11011c11111b…". Fig. 1.3.2. b) prezintă un exemplu de decompresie.

1	2	3	4	5	6	7	1	2	3	4	P	L	A
							a	a	a	b	0		a
						a	a	a	b	a	7	2	b
			a	a	a	b	a	b	a	c	6	3	c
a	a	b	a	b	a	c	a	c	a	b	7	3	b

a) compresia mesajului „aaababacacab…"

1	2	3	4	5	6	7	1	2	3	4	P	L	A
							a				0		a
						a	a	a	a	b	7	3	b
		a	a	a	a	b	a	a	c		3	2	c
a	a	a	b	a	a	c	b	a	c		4	2	c
b	a	a	c	b	a	c	a	c	b	c	3	3	c

b) decompresia mesajului „0aa73b32c42c33c"

Fig. 1.3.2. Exemplu de aplicare a algoritmul LZ–77

Algoritmul LZ–78

Spre deosebire de varianta anterioară, la LZ–78 blocul Lempel este un dicţionar în continuă creştere. De asemenea, nu există teoretic nici o limitare a blocului Ziv.

Algoritmul LZ–78 este, în esenţă, următorul:

-se iniţializează dicţionarul (blocul Lempel) cu un şir nul;

-se încarcă mesajul de intrare în blocul Ziv;

-se transmite în clar primul caracter şi se introduce şi în dicţionar la poziţia 1. Cele două poziţii ale dicţionarului (notate cu 0 şi 1) sunt, în acest moment codate prin „0" şi „1". Din acest moment:

-se caută în dicţionar un subşir, S', identic cu cel mai lung subşir, S, din blocul Ziv. Binenţeles, S începe din prima poziţie a blocului Ziv. Dacă există un astfel de subşir, S', atunci se transmite codul

aferent urmat de caracterul A ce urmează lui S în mesajul de intrare. În plus, dicţionarul se îmbogăţeşte cu o poziţie, poziţie unde se trece subşirul SA. Dacă nici măcar primul caracter, x, din Ziv nu se regăseşte ăn dicţionar, atunci se transmite 0x, unde 0 semnifică subşirul din prima poziţie, codat prin cuvântul de cod „00…0”.

Observaţie: *Atunci când dicţionarul ajunge la un număr de elemente egal cu 2^k, tuturor codurilor elementelor anterioare li se adaugă ca şi prefix „0”, iar ultimul subşir va fi codat prin „10…0”.*

Exemplu: *În Fig. 1.3.3 a) se prezintă un exemplu de compresie utilizând algoritmul LZ–78. Mesajului de intrare este cel din exemplul anterior, „aaababacacab…”. Mesajul comprimat rezultat este „a1a0b1b1c6a3”, sau în binar „a1a0b01c110a011”. În Fig. 1.3.3 b) se prezintă un exemplu de decompresie, utilizând acelaşi algoritm. Mesajul comprimat (de intrare) este „a1b3b5a3a”. Rezultatul decompresiei este „aababcabcaaba”.*

Blocul Lempel (dicţionarul)

0	
1	a
2	aa
3	b
4	ab
5	c
6	ac
7	aca
8	

Blocul Ziv	Mesajul de ieşire
a a a b a b a c a c a b	a
a a b a b a c a c a b	1a
b a b a c a c a b	0b
a b a c a c a b	1b
a c a c a b	1c
a c a b	6a
b	3

b) compresia mesajului „aaababacacab…”;

Blocul Lempel (dicţionarul)

0	
1	a
2	b
3	ab
4	c
5	abc
6	abca
7	

Mesaj decomprimat	Mesaj comprimat
a	a
a a b	1b
a a b a b c	3b
a a b a b c a b c a	5a
a a b a b c a b c a a b a3a	

b) decompresia mesajului „a1b3b5a3a”

Fig. 1.3.3 Exemplu de aplicare a algoritmul LZ–78.

Decompresia presupune următorul algoritm:
-se citeşte primul caracter (codul ASCII aferent); acest caracter se introduce în dicţionar la poziţia 1 şi se generează şi la ieşire; în continuare procedura este:
-se citeşta codul poziţiei la care se găseşte subşirul transmis. Acest cod conţine:

$$k = \sup(\log_2 n) \quad \text{biţi} \tag{1.3.3}$$

unde sup(x) reprezintă aproximarea prin adaos a lui x, iar n dimensiunea momentană a dicţionarului;
-se citeşte din dicţionar subşirul S aflat la poziţia citită anterior şi se generează la ieşire;

-se citeşte din mesajul recepţionat următorul caracter, A (în cod ASCII), şi se generează şi acesta la ieşire. Dacă A este un nou caracter, atunci se introduce în dicţionar la o nouă poziţie;
-se introduce în dicţionar subşirul SA la următoarea poziţie;
Procesul se repetă până la epuizarea mesajului recepţionat.

Algoritmul LZW (Lempel–Ziv–Welch)

Modificarea esenţială adusă de algoritmul LZW este faptul că atât compresorul cât şi decompresorul cunosc simbolurile (caracterele) componente ale mesajului. Cu alte cuvinte, înainte de începerea propriu-zisă a compresiei (decompresiei) dicţionarul este iniţializat cu toate caracterele posibil a fi emise.(În exemplul următor, ilustrat în Figura 3.4a, dicţionarul este iniţializat cu simbolurile a–00, b–01, c–10.) În acest fel nu se mai trimite informaţie despre următorul caracter la fiecare pas. O altă modificare o constituie existenţa unui prefix, P, care se memorează de la un pas la următorul.

Algoritmul compresiei este:
-se iniţializează dicţionarul (aşa cum s-a descris anterior);
-se citeşte primul caracter şi se memorează ca şi prefix, P. Din acest moment:
-se citeşte următorul caracter, notat E. Se caută în dicţionar subşirul PE. Dacă acest subşir există, atunci se trece la pasul următor, fără a emite nimic şi fără a adăuga nimic în dicţionar. Singura modificare este că acum PE devine prefix pentru pasul următor. Dacă subşirul PE nu există în dicţionar, atunci se execută următoarele operaţii:
-se trimite la ieşire codul aferent subşirului P;
-se trece pe post de prefix caracterul E (E→P);
-se adaugă în dicţionar subşirul PE la următoarea poziţie liberă.
Procesul continuă în acest fel până la epuizarea întregului mesaj de intrare.

Observaţie: *La fel ca şi la LZ-78, pe vreme ce dicţionarul creşte trebuie modificat corespunzător codul binar loc asociat.*

Algoritmul decompresiei este asemănător compresiei. Diferenţa este că simbolul E nu se citeşte direct de la intrare (ca şi la compresie) ci doar după ce s-a decodat codul recepţionat. Mesajul decomprimat se poate citi fie de la E (exceptând primul simbol, care se citeşte de la P), fie prin compunerea şirului decodat.

Exemplu: *Fig. 1.3.4. prezintă câte un exemplu de compresie şi decompresie utilizând algoritmul LZW. Pentru compresie s-a ales ca mesaj de intrare: „aaababacacab”, acelaşi ca şi în exemplele anterioare. Mesajul comprimat este: „031052081”, sau în binar „001100100010101000001000001”. Pentru decompresie s-a ales mesajul „013205”, car în binar se exprimă: „0001011010000101”. Mesajul decomprimat este: „ababcaabc”.*

<u>Desfăşurarea lucrării:</u>

a) Rulaţi programul pe calculator, utilizând opţiunea demonstrativă, urmărind aplicarea celor trei algoritmi la compresie şi decompresie. Aflaţi pentru fiecare exemplu şi factorul de compresie.
b). Alcătuiţi, utilizând trei sau patru litere, mesaje de circa 10 litere lungime. Comprimaţi mesajele independent de program şi verificaţi rezultatele cu ajutorul programului. Pentru decompresie utilizaţi mesajele comprimate de colegi, fără a cunoaşte originalul. Confruntaţi rezultatele decomprimării cu cele originale. Calculaţi în fiecare caz factorul de compresie, considerând că orice literă (caracter) se scrie în binar pe 8 biţi.
c) La sfârşitul lucrării de laborator se va efectua test asupra cunoştinţelor acumulate, prin intermediul programului pe calculator. Testul constă din: 1–o compresie şi 2–o decompresie, a unui mesaj ales aleatoriu de către calculator printr-unul dintre cei trei algoritmi (de asemenea ales aleatoriu de program) şi 3 –cinci întrebări teoretice fiecare cu un răspuns corect din cinci propuse, având ca temă algoritmii de compresie.

Prefix	Mesaj de intrare	Şir căutat în dicţionar	Şir transmis	Cod transmis	Dicţionarul a 0=00 b 1=01 c 2=10	
P	E	PE			Şir adăugat în dicţionar	Cod adăugat în dicţionar
	a					
a	a	aa	a	0=00	aa	3=11
a	a	aa				
aa	b	aab	aa	3=11	aab	4=100
b	a	ba	b	1=001	ba	5=101
a	b	ab	a	0=000	ab	6=110
b	a	ba				
ba	c	bac	ba	5=101	bac	7=111
c	a	ca	c	2=010	ca	8=1000
a	c	ac	a	0=0000	ac	9=1001
c	a	ca				
ca	b	cab	ca	8=1000	cab	10=1010
b		b	b	1=0001		

a) compresia mesajului „aaababacacab”;

Prefix	Mesaj de intrare	Şir căutat în dicţionar	Şir recepţionat decodat	Cod recepţionat	Dicţionarul a 0=00 b 1=01 c 2=10	
P	E	PE			Şir adăugat în dicţionar	Cod adăugat în dicţionar
			a	0		
a	b	ab	b	1	ab	3=11
b	a	ba	ab	3	ba	4=100
a	b	ab				
ab	c	abc	c	2	abc	5=101
c	a	ca	a	0	ca	6=110
a	a	aa	abc	5	aa	7=111
a	b	ab				
ab	c	abc				

b) decompresia mesajului „013205”;
Fig. 1.3.4. Exemplu de aplicare a algoritmului LZW

L2. Coduri simple corectoare de o eroare

2.1. Codul Hamming

2.1.1. Cod Hamming corector de o eroare

Codurile Hamming constiruie prima clasă de coduri bloc liniare corectoare de erori şi au fost propuse de R. Hamming în 1950.

Codul Hamming este un cod protector. Scopul codării este acela de a adapta sursa de informaţie la canalul de transmisie. În cazul de faţă sursa este deja codată şi se face o codare pentru protecţia informaţiei. Acest lucru se realizează prin mărirea redundanţei (prin creşterea suportului binar; biţilor de informaţie adăugânduse biţii de control). Biţii de control au ca scop de a crea legături, relaţii între biţii de informaţie. Aceste relaţii folosesc codorului pentru a calcula biţii de control şi folosesc decodorului pentru a verifica corectitudinea transmisiei.

Codul Hamming este un cod nesistematic (simbolurile de control sunt intercalate printre simbolurile informaţionale, situându-se pe poziţii puteri ale lui 2).

2.1.1.1. Codarea codului Hamminmg corector de o eroare

Particularitatea codului liniar Hamming corector de o eroare constă în forma matricii de control, H, care are fiecare coloană structurată prin reprezentarea binară a numărului zecimal de ordine al coloanei respective. Din acest motiv, corectorul Z, calculat cu relaţia $Z = H \cdot V^T$, decodat din binar în zecimal, indică numărul de ordine zecimal al bitului eronat din cuvântul recepţionat.

Distanţa minimă a acestui cod este:

$$d \geq 2e_c + 1 = 3,$$

unde e_c reprezintă numărul de erori corectabile.

Se consideră codul cu parametrii:

- n=7, numărul de simboluri în cuvânt;
- m=3, numărul de simboluri de control în cuvânt;
- k=4, numărul de simboluri de informaţie în cuvânt.

Secvenţa de informaţie este:

$$i = i_3 i_5 i_6 i_7$$

şi secvenţa de control:

$$C = C_1 C_2 C_4$$

astfel încât cuvântul de cod rezultat va fi:

$$V = C_1 C_2 i_3 C_4 i_5 i_6 i_7$$

Codarea înseamnă calculul simbolurilor de control C_1, C_2, şi C_4 când se dau simbolurile de informaţie i_3, i_5, i_6, i_7.

Relaţia de codare:

$$H \cdot V^T = 0 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ i_3 \\ C_4 \\ i_5 \\ i_6 \\ i_7 \end{bmatrix} = 0 \qquad (2.1.1)$$

unde *H* reprezintă matricea de control, ce este de dimensiune *m×n*.

Sau în formă explicită:

$$\begin{aligned} C_1 &= i_3 + i_5 + i_7 \\ C_2 &= i_3 + i_6 + i_7 \\ C_4 &= i_5 + i_6 + i_7 \end{aligned} \qquad (2.1.2)$$

Suma făcându-se modulo doi.

Exemplul 1:

Pentru n=7, k=4, m=3 şi secvenţa de informaţie i=1010, găsiţi cuvântul V de cod.

Rezolvare:

Se observă că există 4 simboluri de informaţie: i_3, i_5, i_6, i_7. Cuvântul de cod,V, se poate scrie sub formă literară: $V = C_1C_2i_3C_4i_5i_6i_7$. Avem aşadar 7 simboluri ce alcătuiesc cuvântul de cod şi 3 biţi de control. Rezultă că matricea de control, H, va conţine 3 linii şi 7 coloane.
Din relaţia de codare (2.1.1), rezultă relaţiile de calcul a biţilor de control:

$$\begin{aligned} C_1 &= i_3 + i_5 + i_7 = 1+0+0 = 1 \\ C_2 &= i_3 + i_6 + i_7 = 1+1+0 = 0 \\ C_4 &= i_5 + i_6 + i_7 = 0+1+0 = 1 \end{aligned}$$

Rezultă cuvântul de cod:

$$V=1011010$$

2.1.1.2. Decodarea codului Hamminmg corector de o eroare

Având cuvântul recepţionat V', compus din 7 imboluri:

$$V' = C_1'C_2'i_3'C_4'i_5'i_6'i_7'$$

se poate calcula corectorul codului Hamming cu relaţia:

$$z = H \cdot V'^{T} = \begin{bmatrix} z_4 \\ z_2 \\ z_1 \end{bmatrix} \qquad (2.1.3)$$

Matricea de control:

$$H_{[m,n]} = \begin{bmatrix} h_1 & \cdots & h_i & \cdots & h_n \end{bmatrix} \qquad (2.1.4)$$

unde, pentru acest caz se observă că m=3 şi n=7. Coloana h_i exprimă în cod binar natural numărul de ordine al coloanei respective, cu bitul cel mai puţin semnificativ în linia *m*.

Din relaţia (2.1.3) rezultă:

$$\begin{aligned} z_4 &= C_4' + i_5' + i_6' + i_7' \\ z_2 &= C_2' + i_3' + i_6' + i_7' \\ z_1 &= C_1' + i_3' + i_5' + i_7' \end{aligned} \qquad (2.1.5)$$

Decodarea zecimală a corectorului z, indică poziţia erorii, dacă este una singură, în cuvântul V'. Va fi eronat simbolul cu indicele r, dat de relaţia:

$$r = 4z_4 + 2z_2 + z_1 \qquad (2.1.6)$$

Cuvântul recepţuionat se mai poate scrie ca fiind:

$$V' = V + \varepsilon,$$

unde ε reprezintă cuvântul eroare.
Relaţia (2.1.3) va deveni:

$$z = H \cdot V'^{T} = H \cdot (V + \varepsilon)^{T} = H \cdot \varepsilon^{T} = h_r$$

unde poziţia erorii se calculează cu relaţia (2.1.6).

Obs: *În cazul în care există două erori, sunt cazuri în care nu numai că nu se corectează nici o eroare, dar se mai eronează un al treilea simbol, rezultând la recepţie trei simboluri eronate.*

Exemplul 2:

Decodaţi cuvântul recepţionat V'=1001001

Rezolvare:

Cuvântul de cod recepţionat se poate scrie ca fiind:

$$V' = C_1' C_2' i_3' C_4' i_5' i_6' i_7'$$

$$V' = 1\ 0\ 0\ 1\ 0\ 0\ 1$$

Relaţia (2.1.5) va deveni:

$$z_4 = C_4' + i_5' + i_6' + i_7' = 1 + 0 + 0 + 1 = 0$$
$$z_2 = C_2' + i_3' + i_6' + i_7' = 0 + 0 + 0 + 1 = 1$$
$$z_1 = C_1' + i_3' + i_5' + i_7' = 1 + 0 + 0 + 1 = 0$$

Aşadar, poziţia eronată este:

$$r = 4z_4 + 2z_2 + z_1 = 4 \cdot 0 + 2 \cdot 1 + 1 \cdot 0 = 2$$

Rezultă cuvântul eroare:

$$\varepsilon = 0100000$$

Se poate scrie:

$$V = V' + \varepsilon = 1001001 + 0100000 = 1101001$$

Rezultă secvenţa de informaţie:

$$i = i_3 i_5 i_6 i_7 = 0001$$

2.1.2. Cod Hamming corector de o eroare şi detector de două erori

Condiţia necesară şi suficientă pentru ca un cod să poată simultan corecta maxim *t* erori şi a detecta maxim *e* erori este ca distanţa minimă a codului să fie:

$$d \geq t + e + 1 = 1 + 2 + 1 = 4, \qquad e > t$$

2.1.2.1. Codarea codului Hamminmg corector de o eroare şi detector de două erori

Pentru a înlătura dezavantajul codului Hamming corector de o eroare (acela de a erona suplimentar, la depăşirea capacităţii de corecţie a codului, t>1) şi de al face mai util în aplicaţii

practice, codul a fost modificat în sensul creşterii distanţei minime de la d=3 la d=4, ceea ce permite detecţia erorilor duble.

Creşterea distanţei de cod de la 3 la 4 s-a făcut prin adăugarea unui simbol de control suplimentar, numit simbol de control al parităţii, C_0, structura cuvîntului de cod devenind:

$$V = C_0C_1C_2i_3C_4i_5i_6i_7$$

Matricea de control se modifică şi are structura:

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

Relaţia de codare va fi:

$$H' \cdot V^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ i_3 \\ C_4 \\ i_5 \\ i_6 \\ i_7 \end{bmatrix} = 0 \qquad (2.1.7)$$

Sau în formă explicită:

$$\begin{aligned} C_1 &= i_3 + i_5 + i_7 \\ C_2 &= i_3 + i_6 + i_7 \\ C_4 &= i_5 + i_6 + i_7 \\ C_0 &= C_1 + C_2 + i_3 + C_4 + i_5 + i_6 + i_7 \end{aligned} \qquad (2.1.8)$$

Exemplul 3:

Fie secvenţa de informaţie i=1010, găsiţi cuvântul V de cod.

Rezolvare:

Se observă că există 4 simboluri de informaţie: i_3, i_5, i_6, i_7. Cuvântul de cod, V, se poate scrie sub formă literară: $V = C_0C_1C_2i_3C_4i_5i_6i_7$. Avem aşadar 8 simboluri ce alcătuiesc cuvântul de cod şi 4 biţi de control. Rezultă că matricea de control, H', va conţine 4 linii şi 8 coloane.
Din relaţia de codare (2.1.8), rezultă relaţiile de calcul a biţilor de control:

$$\begin{aligned} C_1 &= i_3 + i_5 + i_7 = 1+0+0 = 1 \\ C_2 &= i_3 + i_6 + i_7 = 1+1+0 = 0 \\ C_4 &= i_5 + i_6 + i_7 = 0+1+0 = 1 \\ C_0 &= C_1 + C_2 + i_3 + C_4 + i_5 + i_6 + i_7 = 1+0+1+1+0+1+0 = 0 \end{aligned}$$

Rezultă cuvântul de cod:

V=01011010

2.1.2.2. Decodarea codului Hamminmg corector de o eroare şi detector de două erori

Având cuvântul recepţionat V', compus din 8 imboluri:

$$V' = C_0'C_1'C_2'i_3'C_4'i_5'i_6'i_7'$$

se poate calcula corectorul codului cu relaţia:

$$Z = H' \cdot V'^{T} = \begin{bmatrix} z \\ z_0 \end{bmatrix} = \begin{bmatrix} z_4 \\ z_2 \\ z_1 \\ z_0 \end{bmatrix} \qquad (2.1.9)$$

Unde:

- z, reprezintă corectorul de la codul Hamming corector de o eroare;
- z_0, reprezintă simbolul binar, 0 sau 1, cu ajutorul căruia se pot detecta erori pare (z_0=0).

Există patru cazuri:

Cazul 1:

$$\begin{cases} z = 0 \\ z_0 = 0 \end{cases}$$

nu există erori sau nu există erori detectabile prin cod.

Cazul 2:

$$\begin{cases} z \neq 1 \\ z_0 = 0 \end{cases}$$

se face detecţia erorilor duble.

Cazul 3:

$$\begin{cases} z = 0 \\ z_0 = 1 \end{cases}$$

simbolul C_0 este eronat.

Cazul 4:

$$\begin{cases} z \neq 0 \\ z_0 = 1 \end{cases}$$

există o eroare corectabilă.

Va fi eronat simbolul cu indicele r-1, unde r este dat de relaţia:

$$r = 4z_4 + 2z_2 + z_1 + z_0 \qquad (2.1.10)$$

Exemplul 4:

Decodaţi cuvântul recepţionat V'=01001010

Rezolvare:

Cuvântul de cod recepţionat se poate scrie ca fiind:

$$V' = C_0'C_1'C_2'i_3'C_4'i_5'i_6'i_7'$$
$$V' = 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0$$

Relaţia (2.1.5) va deveni:

$$z_4 = C_4^{'} + i_5^{'} + i_6^{'} + i_7^{'} = 1+0+1+0 = 0$$
$$z_2 = C_2^{'} + i_3^{'} + i_6^{'} + i_7^{'} = 0+0+1+0 = 1$$
$$z_1 = C_1^{'} + i_3^{'} + i_5^{'} + i_7^{'} = 1+0+0+0 = 1$$
$$z_0 = C_0^{'} + C_1^{'} + C_2^{'} + i_3^{'} + C_4^{'} + i_5^{'} + i_6^{'} + i_7^{'} = 0+1+0+0+1+0+1+0 = 1$$

Rezultă că avem situaţia cazului 4, când există o eroare corectabilă.
Aşadar, rezultă r:

$$r = 4z_4 + 2z_2 + z_1 + z_0 = 4\cdot 0 + 2\cdot 1 + 1\cdot 1 + 1 = 4$$

Deci va fi eronat simbolul cu indicele r-1, adică 4-1=3, i_3.
Rezultă cuvântul eroare:

$$\varepsilon = 00010000$$

Se poate scrie:

$$V = V^{'} + \varepsilon = 01001010 + 00010000 = 01011010$$

Rezultă secvenţa de informaţie:

$$i = i_3 i_5 i_6 i_7 = 1010$$

Desfăşurarea lucrării:
a) Se lansează executabilul Hamming.exe din directorul Hamming, după care se introduce parola TTI. Rulaţi programul, selectând opţiunea Demonstraţie, pentru toate cele patru variante prezentate.
b) Se verifică, cu programul, toate exemplele luate în această lucrare, selectând pe rând codurile prezentate.
c) Pentru fiecare cod în parte, se alege câte un exemplu de codare/decodare, asemănător cu cele prezentate în lucrare, cu observaţia ca secvenţa de informaţie să conţină 11 biţi de informaţie, la codare, respectiv cuvîntul recepţionat să conţină 15/16 biţi, la decodare. Se realizează codarea/decodare, după care, cu programul, se verifică rezultatele obţinute.
d) La sfârşitul lucrării de laborator se va efectua un test asupra cunoştinţelor acumulate. Rezultatul testului se va concretiza printr-o notă, calculată automat de calculator.

2.2. Codul ciclic

Codurile ciclice sunt utilizate pentru protejarea informaţiei împotriva perturbaţiilor.

Codurile ciclice sunt coduri bloc (toate cuvintele au aceeaşi lungime, codarea şi decodarea unui bloc este independentă de a celorlalte).

Denumirea de "ciclice" provine de la proprietatea pe care o au cuvintele de cod şi anume dacă v_i = 1001001 este cuvânt de cod atunci şi v_j = 0010011 şi v_k = 1100100 sunt cuvinte de cod . Adică orice permutare ciclică a unui cuvânt de cod este un alt cuvânt de cod .

Parametrii codului sunt n (biţi) lungimea cuvântului de cod, k numărul de biţi de informaţie per cuvânt, m numărul de biţi de control per cuvânt şi polinomul generator g(x). Deşi poate fi facută şi o codare nesistematică vom considera codul sistematic cei k biţi de informaţie fiind situaţi pe poziţiile cele mai semnificative, iar cei m biţi de control pe poziţiile mai puţin semnificative.
Fiecărui cuvânt de cod i se poate ataşa un polinom de grad maxim n-1:

$$v = v_{n-1}v_{n-2}v_{n-3}.....v_1v_0,$$

coeficienţii v_i fiind coeficienţi binari (din câmpul binar {0,1}).
Polinomul asociat cuvântului va fi :

$$v(x) = v_{n-1}x^{n-1} + v_{n-2}x^{n-2} + v_{n-3}x^{n-3} + ...+ v_1x + v_0 \qquad (2.2.1)$$

Ponderea unui cuvânt reprezintă numărul de 1 din cadrul cuvântului.
Polinomul de informaţie este :

$$i(x) = i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + i_{k-3}x^{k-3} + ...+ i_1x + i_0 \qquad (2.2.2)$$

de grad k-1.

Pentru orice cuvânt de cod este valabilă relaţia :

$$\text{rest}\frac{v(x)}{g(x)} = 0 \qquad (2.2.3)$$

Deci cuvintele de cod se aleg astfel încât să fie multiplii ai polinomului g(x) numit polinomul generator al carui grad va fi deci m = n-k, ($g_m = 1$):

$$g(x) = g_mx^m + g_{m-1}x^{m-1} + g_{m-2}x^{m-2} + ...+ g_1x + g_0 \qquad (2.2.4)$$

Operaţiile se fac modulo $(x^n + 1)$.
Dacă, codul este corector de o singură eroare atunci:

$$n = 2^m - 1. \qquad (2.2.5)$$

Codurile ciclice corectoare de o eroare, având distanţa de cod (distanţa Hamming minimă între cuvintele codului) $d_{Hmin} = 3$, sunt capabile să corecteze o eroare sau să detecteze două.

Codarea codului ciclic corector de o eroare

Codarea va fi prezentată în două moduri nesistematică sau sistematică.

Cu relaţia $v(x) = i(x)\cdot g(x)$ se poate face codarea nesistematică, dar pentru că nu este utilă în practică nu va fi luată în considerare .

În continuare va fi deci prezentată codarea sistematică unde corespondenţa dintre i(x) şi v(x) este dată prin relaţia:

$$v(x) = i(x)\cdot x^m + \text{rest}\frac{i(x)\cdot x^m}{g(x)} \qquad (2.2.6)$$

unde $\text{rest}\frac{i(x)\cdot x^m}{g(x)}$ semnifică restul împărţirii polinomului $i(x)\cdot x^m$ la g(x).

Operaţia de adunare din ecuaţia (2.2.6) este sumă modulo 2 (SAU exclusiv), iar coeficienţii polinoamelor sunt din câmpul binar {0,1}.
Matematic codarea poate fi facută polinomial sau matricial.

a) Polinomial

Codarea va fi prezentată printr-un exemplu, putând fi uşor generalizată.

Fie $g(x) = x^3 + x + 1$. Deci m = 3 şi cu relaţia $2^m - 1 = n$ rezultă că n = 7 de unde k = n – m = 4. Vom avea ca atare 4 biţi de informaţie $i = v_{n-1}v_{n-2}v_{n-3}v_{n-4}$ cu polinomul asociat :

$$i(x) = v_{n-1}x^3 + v_{n-2}x^2 + v_{n-3}x + v_{n-4} \qquad (2.2.7)$$

şi deci $i(x)\cdot x^m = (v_{n-1}x^3 + v_{n-2}x^2 + v_{n-3}x^3 + v_{n-1}x + v_{n-4})\cdot x^3 = v_{n-1}x^6 + v_{n-2}x^5 + v_{n-3}x^4 + v_{n-4}x^3$

Biţii de control sunt coeficienţii restului împărţirii lui $i(x)\cdot x^m/g(x)$.

$$\begin{array}{l|l} v_{n-1}x^6 + v_{n-2}x^5 + v_{n-3}x^4 + v_{n-4}x^3 & \underline{x^3 + x + 1} \\ \underline{v_{n-1}x^6 \quad + v_{n-1}x^4 + v_{n-1}x^3} & v_{n-1}x^3 + v_{n-2}x^2 + (v_{n-3} + v_{n-1})x + (v_{n-4} + v_{n-2} + v_{n-1}) \end{array}$$

$$/\ \ v_{n-2}x^5 + (v_{n-3} + v_{n-1})x^4 + (v_{n-4} + v_{n-1})x^3$$
$$\underline{v_{n-2}x^5 \qquad\qquad + v_{n-2}x^3 \qquad + v_{n-2}x^2}$$

$$/\ \ (v_{n-3} + v_{n-1})x^4 + (v_{n-4} + v_{n-2} + v_{n-1})x^3 + v_{n-2}x^2$$
$$\underline{(v_{n-3} + v_{n-1})x^4 \qquad\qquad + (v_{n-3} + v_{n-1})x^2 + (v_{n-3} + v_{n-1})x}$$

$$/\ \ (v_{n-4} + v_{n-2} + v_{n-1})x^3 + (v_{n-3} + v_{n-2} + v_{n-1})x^2 + (v_{n-3} + v_{n-1})x$$
$$\underline{(v_{n-4} + v_{n-2} + v_{n-1})x^3 \qquad\qquad + (v_{n-4} + v_{n-2} + v_{n-1})x + v_{n-4} + v_{n-2} + v_{n-1}}$$

$$(v_{n-3} + v_{n-2} + v_{n-1})x^2 + (v_{n-4} + v_{n-3} + v_{n-2})x + v_{n-4} + v_{n-2} + v_{n-1}$$

$$\Rightarrow r(x) = (v_{n-3} + v_{n-2} + v_{n-1})x^2 + (v_{n-4} + v_{n-3} + v_{n-2})x + v_{n-4} + v_{n-2} + v_{n-1}, \qquad (2.2.8)$$

unde r(x) este restul împărţirii .
Conform relaţiei (2.2.6) se obţine :

$$v(x) = v_{n-1}x^6 + v_{n-2}x^5 + v_{n-3}x^4 + v_{n-4}x^3 + (v_{n-3} + v_{n-2} + v_{n-1})x^2 + (v_{n-4} + v_{n-3} + v_{n-2})x + v_{n-4} + v_{n-2} + v_{n-1}$$

Pentru că n = 7 şi v(x) este de forma:

$$v(x) = v_6x^6 + v_5x^5 + v_4x^4 + v_3x^3 + v_2x^2 + v_1x + v_0 \qquad (2.2.9)$$

şi ţinând cont de relaţia (2.2.8) simbolurile de control vor fi date de:

$$\begin{aligned} v_2 &= v_4 + v_5 + v_6 \\ v_1 &= v_3 + v_4 + v_5 \\ v_0 &= v_3 + v_5 + v_6 \end{aligned} \qquad (2.2.10)$$

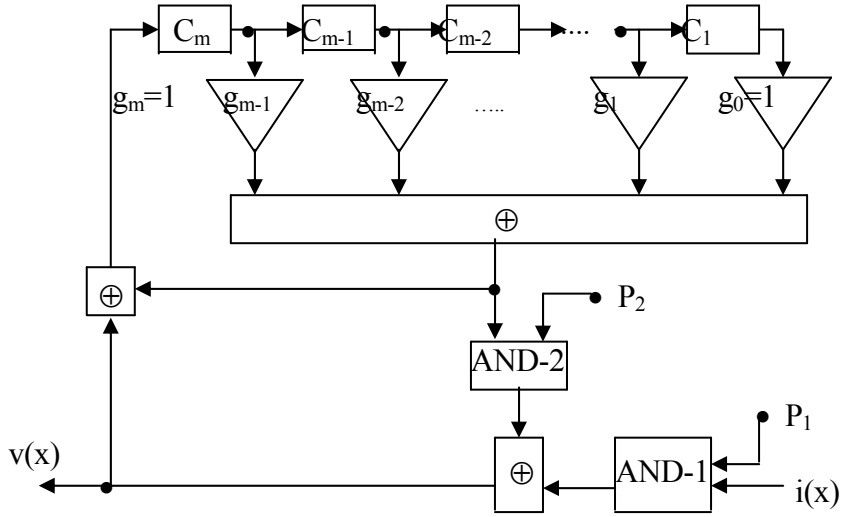
Aşadar cuvântul de cod va fi $v = v_6v_5v_4v_3v_2v_1v_0$ unde primii 4 sunt biţii de informaţie, iar ultimii 3 cei de control.

b) Codor ciclic cu RDR (codarea matricială)

În Fig. 2.2.1 este prezentat un codor ciclic care implementează relaţia de codare (2.2.6). Secvenţa de informaţie, i(x), intră în codor în primele k tacte, primul bit fiind cel mai semnificativ şi, de asemenea, este conectată şi la ieşire. Pentru aceasta, întrerupătorul 1, format din poarta AND-1 este închis iar întrerupătorul 2, format din poarta AND-2 este deschis (vezi semnalele de comandă P_1 şi P_2).

În următoarele m tacte întrerupătorul 1 este deschis iar întrerupătorul 2 este închis, astfel că secvenţa r, generată de RDR, este livrată la ieşirea v.

a) Codor ciclic RDR ;



b) Semnalele de validare a porţilor „AND”

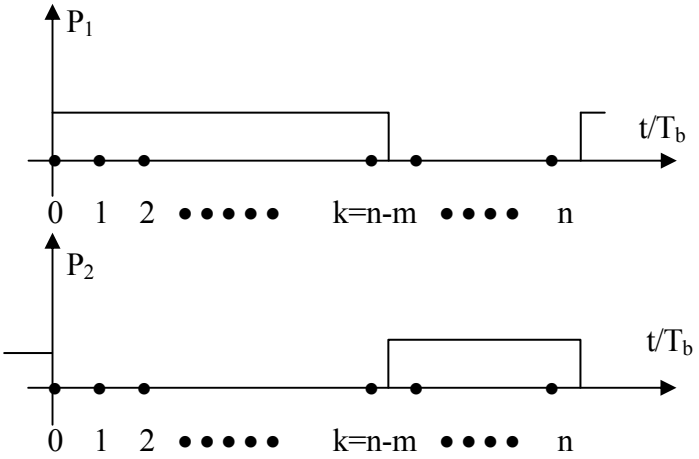


Fig. 2.2.1 Codor ciclic cu RDR şi semnalele de comandă

Obs. *–Se observă că, în ultimele k tacte, la intrările sumatorului (care adună intrarea RDR-lui cu reacţia sa) se regăseşte acelaşi semnal, astfel că, în aceste k tacte, în celula C_k se va introduce o secvenţă nulă care „curăţă” registrul pentru un nou cuvânt de cod.*

<u>Din funcţionarea registrului de deplasare cu reacţie rezultă relaţia:</u>

$$S_j = TS_{j-1} + v_{n-j}U \tag{2.2.11}$$

unde S reprezintă starea registrului, U este o matrice coloană, iar T este matricea caracteristică a registrului de deplasare cu reacţie. Ele sunt de forma:

$$S=\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}, \qquad U=\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, \qquad T=\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ g_0 & g_1 & g_2 & \cdots & g_{m-1} \end{bmatrix}. \tag{2.2.12}$$

Ţinând cont de observaţia făcută mai sus rezultă că $S_n = 0$.
Considerând acelaş exemplu

Matricea caracteristică, T, este:

$$T = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \tag{2.2.13}$$

Utilizând relaţiile (2.2.11) şi (2.2.13) pentru cazul n = 7 vom avea:

$$S_7 = v_6T^6U + v_5T^5U + v_4T^4U + v_3T^3U + v_2T^2U + v_1TU + v_0U = 0 \tag{2.2.14}$$

Efectuând calculele rezultă:

$$T \cdot U = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix},\ T^2 \cdot U = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix},\ T^3 \cdot U = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix},\ T^4 \cdot U = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix},$$
$$T^5 \cdot U = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix},\ T^6 \cdot U = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{2.2.15}$$

Introducând (2.2.15) în (2.2.14) şi efectuând calculele obţinem:

$$v_2 = v_4 + v_5 + v_6$$
$$v_1 = v_3 + v_4 + v_5$$
$$v_0 = v_4 + v_3 + v_2 = v_4 + v_3 + v_4 + v_5 + v_6 = v_3 + v_5 + v_6$$

relaţii identice cu relaţiile (2.2.10) deci cele două proceduri de calcul ne-au dus la aceleaşi rezultate .

Cuvântul de cod va fi $v = v_6v_5v_4v_3v_2v_1v_0$.

Decodarea codului ciclic corector de o eroare

Decodarea codului ciclic cuprinde verificarea corectitudinii cuvântului recepţionat:

$$w(x) = v'_{n-1}x^{n-1} + v'_{n-2}x^{n-2} + ... + v'_1x + v'_0, \tag{2.2.16}$$

corectarea sau detectarea erorilor, după destinaţia codului şi apoi selecţia biţilor de informaţie.

Verificarea presupune calculul restului împărţirii lui w(x) la g(x). Dacă restul este 0 se decide: cuvânt corect recepţionat. Dacă nu, atunci se decide: cuvânt eronat.(Prima decizie poate fi falsă, a doua nu!) Pentru codurile detectoare decodarea ia sfârşit aici. Pentru codurile corectoare analiza restului permite determinarea poziţiei erorilor. Acest lucru presupune existenţa unei corespondenţe biunivoce între resturile posibile şi cuvintele eroare corectabile de către codul dat.

Dacă codul este corector de o eroare, atunci decodarea decurge astfel:

1°- se calculează corectorul z_n cu formula:

$$z = \sum_{j=0}^{n-1} v'_j T^j U = \sum_{j=0}^{n-1} \varepsilon_j T^j U \tag{2.2.17}$$

unde ε_j reprezintă coeficienţii polinomului eroare:

$$\varepsilon(x) = w(x) + v(x) \tag{2.2.18}$$

2°- dacă z = 0 se decide că w(x) este corect w(x) = v(x)

- dacă $z \neq 0$ atunci $z = T^rU$, unde r este indicele coeficientului eronat. Comparăm z cu T^jU şi găsim r. Corecţia presupune schimbarea valorii lui v'_r.

Dacă codul ciclic corectează mai multe erori, atunci decodarea se poate face pe seama corespondenţei amintite anterior.

a) Decodarea polinomială

Metoda constă în împărţirea lui w(x) la g(x). Va rezulta un polinom rest, r(x). Dacă acesta este diferit de 0, cuvântul recepţionat este eronat şi prin identificarea restului r(x) cu valorile din tabelul cuvinte eroare – corectori (vezi Anexe) se determină poziţia erorii şi se realizează corecţia.

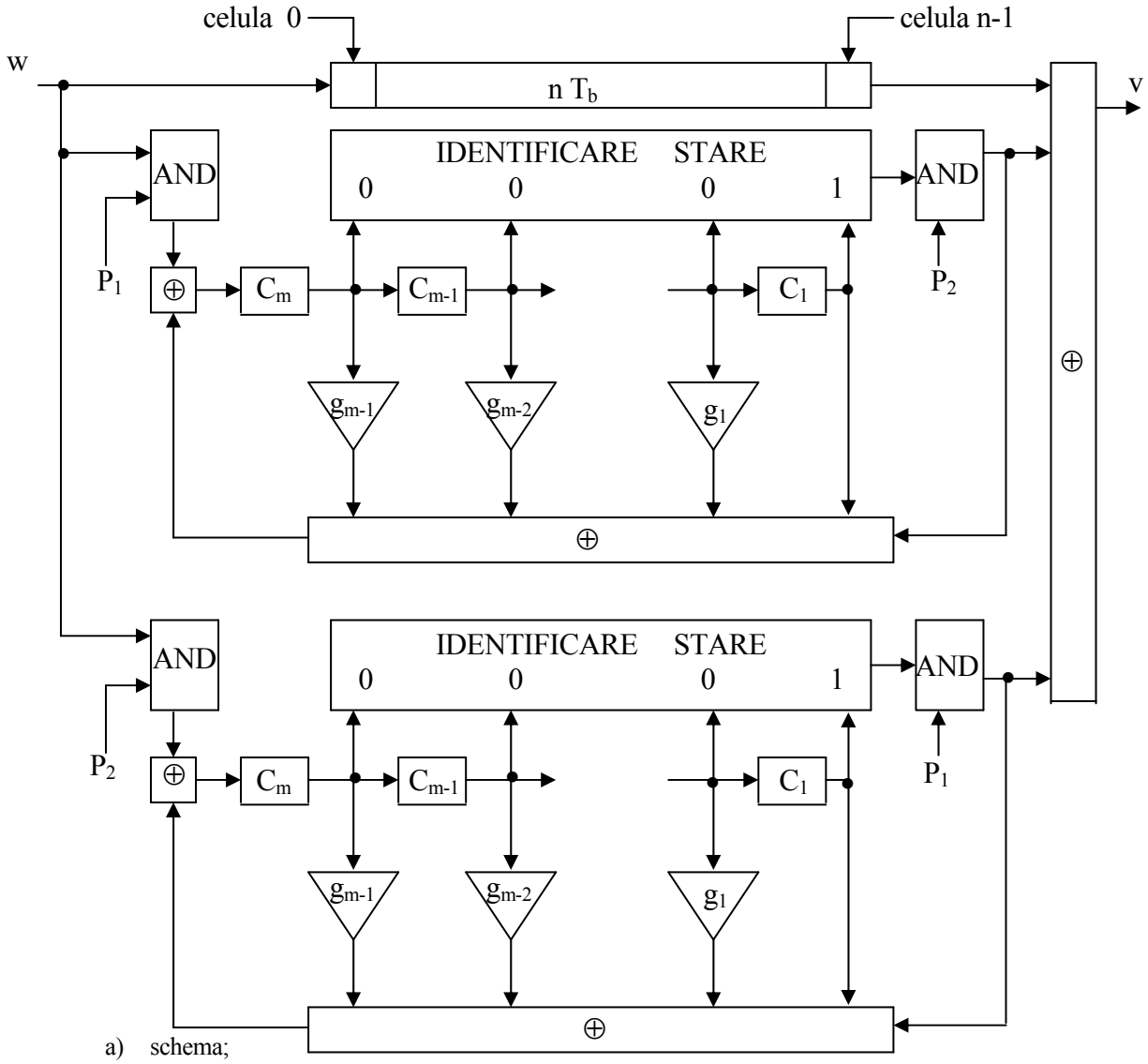
Spre exemplu fie : $w(x) = x^6 + x^5 + x^3 + x^2 + 1$. Calculăm rest[$w(x) / g(x)$]:

$$
\begin{array}{l|l}
x^6 + x^5 + x^3 + x^2 + 1 & x^3 + x + 1 \\
\underline{x^6 + x^4 + x^3} & \underline{x^3 + x^2 + x + 1} \\
x^5 + x^4 + x^2 + 1 & \\
\underline{x^5 + x^3 + x^2} & \\
x^4 + x^3 + 1 & \\
\underline{x^4 + x^2 + x} & \\
x^3 + x^2 + x + 1 & \\
\underline{x^3 + x + 1} & \\
x^2 &
\end{array}
$$

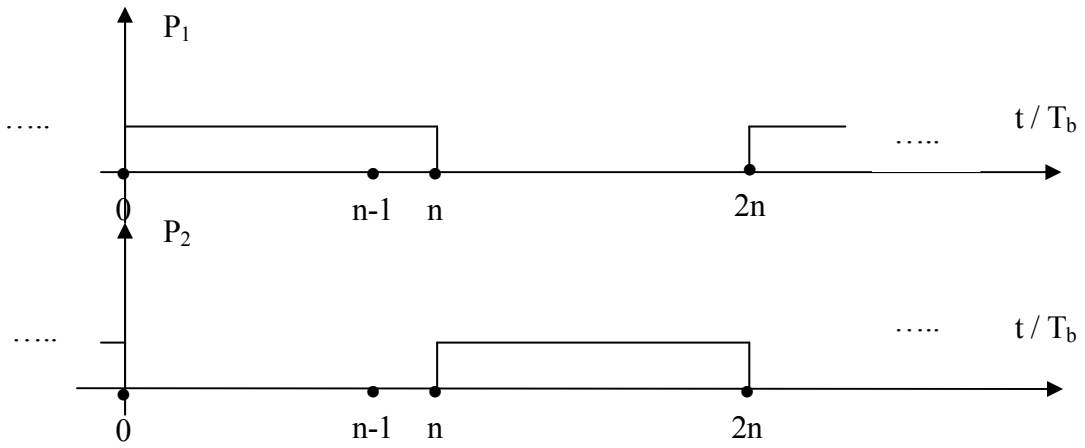
deci conform tabelului din anexă este eronat w_2. Schimbând valoarea bitului w_2 rezultă :

$$w(x) = x^6 + x^5 + x^3 + 1.$$

b) Decodor ciclic cu RDR (decodarea matricială)



a) schema;



b)Semnalele de validare a porţilor AND ;

Fig. 2.2.2. Decodor ciclic corector de o eroare

În Fig. 2.2.2. este prezentată structura unui decodor ciclic corector de o eroare. Biţii cuvântului recepţionat vor intra în schemă unul după altul în ordine, începând cu bitul v'_{n-1}.

Schema de decodare conţine un registru de deplasare (blocul „nT_b" este un registru de întârziere cu n celule) cu rol de memorie, care păstrează cuvântul recepţionat care intră pas cu pas şi două subscheme cu RDR pentru corecţie care funcţionează în contratimp. Procedura de decodare durează 2n tacte. În primele n tacte în memorie intră cuvântul 1 care prin poarta P_1 intră şi în primul decodor, ieşirea acestuia fiind 0 pentru că P_2 este închis. În următoarele n tacte cuvântul 2 va intra prin poarta P_2 în al doilea decodor care are ieşirea 0 deoarece P_1 este închis, în acest timp cuvântul 1 părăseşte memoria trecând prin sumator şi este corectat de primul decodor care identifică starea 00.....1 şi care are acces la sumator prin P_2.

După n = 7 tacte starea registrului va fi:

$$S'_7 = v'_6T^6U + v'_5T^5U + v'_4T^4U + v'_3T^3U + v'_2T^2U + v'_1TU + v'_0U \qquad (2.2.19)$$

care nu va mai fi 0 decât în cazul în care nu avem eroare.

Dacă eroarea afectează bitul v_r starea registrului de deplasare cu reacţie după intrarea întregului cuvânt este:

$$S'_7 = v'_6T^6U + v'_5T^5U + ... + v'_rT^rU + ... + v'_1TU + v'_0U \qquad (2.2.20)$$

Deoarece la emisie $S_7 = 0$ şi $v'_j = v_j$ pentru $j \neq r$ şi $v'_r = v_r + 1$ rezultă:

$$S'_7 + S_7 = S'_7 = T^rU = z \qquad (2.2.21)$$

Pe durata următoarelor n = 7 tacte RDR din prima subschemă va evolua numai sub acţiunea semnalului de reacţie şi după n-r-1 tacte se va ajunge în starea:

$$S'_{7+n-r-1} = T^{n-r-1}\, S'_7 = T^{n-1}U = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad (2.2.22)$$

În acelaş timp printr-o deplasare cu n-r-1 tacte eroarea care se afla în celula r va ajunge în celula n-1. Detectorul va sesiza starea $S'_{7+n-r-1}$ fixă şi prin poarta P_2 care este închisă va emite semnalul de corecţie (blocul IDENTIFICĂ generează un 1^L) care se însumează cu bitul eronat aflat în celula n-1, astfel la ieşirea sumatorului final se va obţine cuvântul corectat.

Desfăşurarea lucrării:

a) Rulaţi programul de pe calculator alegând butonul “Demonstraţie” de la codul ciclic şi se urmăreşte o codare şi o decodare.

b) Alegeţivă un polinom generator şi secvenţa de informaţie şi efectuaţi o codare, apoi alegând polinomul generator şi biţii recepţionaţi o decodare. Verificaţi apoi rezultatul cu ajutorul calculatorului.

c) La sfârşitul lucrării de laborator se efectuează testul de verificare a cunoştinţelor acumulate prin intermediul calculatorului. Testul constă în a răspunde la 5 întrebări teoretice, fiecare având un răspuns corect din cele cinci propuse, şi efectuarea unei codări şi a unei decodări pornind de la polinomul generator şi secvenţa de informaţie respectiv secvenţa recepţionată generate aleator de calculator.

L3. Coduri ciclice corectoare de erori multiple

3.1 Codul BCH

Codurile BCH fac parte din categoria codurilor ciclice.
Cuvintele de cod BCH vor avea deci structura ca şi cele de cod ciclic:

$$v = v_{n-1}\, v_{n-2} \ldots v_1\, v_0 \qquad v_j \in GF(2^q, p(x)) \quad j = 0 \div n-1 \tag{3.1.1}$$

unde coeficienţii u_j sunt coeficienţi binari (fac parte din câmpul binar {0, 1}).
Polinomul corespunzător acestui cuvânt fiind:

$$v(x) = v_{n-1}x^{n-1} + v_{n-2}x^{n-2} + \ldots + v_1x + v_0 \tag{3.1.2}$$

adică un polinom de grad n - 1.

Polinomul de informaţie este:

$$i(x) = i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + \ldots + i_1x + i_0 \tag{3.1.3}$$

care este un polinom de gradul k - 1.

Cuvintele de cod se aleg astfel încât să fie multiplii ai polinomului generator g(x) şi deci acesta va trebui sa fie un polinom de gradul m = n – k:

$$g(x) = g_mx^m + g_{m-1}x^{m-1} + \ldots + g_1x + g_0 \tag{3.1.4}$$

Coeficienţii polinomului g(x) sunt de asemenea coeficienţi binari.
Deoarece polinomul trebuie să aibă gradul m rezultă că $g_m = 1$, iar g_0 trebuie să fie şi el egal cu 1 pentru că altfel putem da factor pe x şi gradul polinomului va fi mai mic decât k.

Polinomul g(x) este deci de forma:

$$g(x) = x^m + g_{m-1}x^{m-1} + \ldots + g_1x + 1 \tag{3.1.5}$$

Cuvintele de cod sunt elemente ale câmpului Galois $GF(2^q)$ generat de un polinom primitiv p(x) de grad q (sunt clase de resturi modulo p(x)), câmp obţinut aşa cum este prezentat în Anexa.

Codarea BCH

Codarea codurilor BCH poate fi făcută în două moduri:

a) Cu relaţia

$$v(x) = i(x)\cdot g(x) \tag{3.1.6}$$

care conduce la obţinerea unui cod nesistematic (relaţie mai puţin utilizată).

b) Pentru obţinerea unei structuri sistematice, la care informaţia să se găsească nemodificată pe poziţiile cele mai semnificative se foloseşte relaţia:

$$v(x) = i(x)\cdot x^m + \text{rest}\,(i(x)\cdot x^m/g(x)) \tag{3.1.7}$$

Această relaţie se mai poate scrie:

$$v(x) = q(x)\cdot g(x) = v_{n-1}x^{n-1} + v_{n-2}x^{n-2} + \ldots + v_1 x + v_0 \qquad (3.1.8)$$

deci se obţine un cuvânt de cod ciclic care este multiplu a lui g(x) şi este format din simbolurile de informaţie aflate pe poziţiile cele mai semnificative (primele k) şi m simboluri de control determinate de restul împărţirii lui $i(x)\cdot x^m$ la g(x).
Relaţia (3.1.8) poate fi adusă sub forma $H\cdot v^T = 0$.

$$\begin{bmatrix} h_0 & h_1 & \ldots & h_{m-1} & h_m & 0 & \ldots & 0 & 0 \\ 0 & h_0 & \ldots & h_{m-2} & h_{m-1} & h_m & \ldots & 0 & 0 \\ \vdots & & & & & & & & \\ 0 & 0 & \ldots & 0 & 0 & h_0 & \ldots & h_{m-1} & h_m \end{bmatrix} \begin{bmatrix} v_{n-1} \\ \vdots \\ v_1 \\ v_0 \end{bmatrix} = 0 \qquad (3.1.9)$$

Acesta este de forma $H\cdot v^T = 0$ deci îl putem determina pe H prin identificare ca fiind:

$$H = \begin{bmatrix} h_0 & h_1 & \ldots & h_{m-1} & h_m & 0 & \ldots & 0 & 0 \\ 0 & h_0 & \ldots & h_{m-2} & h_{m-1} & h_m & \ldots & 0 & 0 \\ \vdots & & & & & & & & \\ 0 & 0 & \ldots & 0 & 0 & h_0 & \ldots & h_{m-1} & h_m \end{bmatrix} \qquad (3.1.10)$$

Pentru a corecta t erori independente g(x) trebuie să îndeplinească anumite condiţii. Ştiim că v(x) trebuie să fie multiplu al lui g(x) şi că g(x) trebuie să fie divizor a lui $x^n + 1 = 0$.
Pentru aceasta alegem un număr de r rădăcini ale lui $x^n + 1 = 0$ pe care le notăm cu $\beta_i = \alpha^i \in GF(2^q)$. Aceste rădăcini β_i sunt elemente primitive ale extensiei de ordinul q al câmpului Galois binar $GF(2^q)$. Acest ordin poate fi determinat cu relaţia:

$$n = 2^q - 1 \qquad (3.1.11)$$

de aici rezultând q şi extensia în care se lucrează $GF(2^q)$.
Pe g(x) îl vom determina ca cel mai mic multiplu comun al polinoamelor minimale a rădăcinilor β_i.

$$g(x) = c.m.m.m.c.\{m_1(x),\ldots m_r(x)\} \qquad (3.1.12)$$

Iar dacă toate aceste r polinoame sunt relativ prime între ele atunci:

$$g(x) = \prod_{i=1}^{r} m_i(x) \qquad (3.1.13)$$

Expresia polinomului minimal care este definit ca polinomul $m_i(x)$ ireductibil de grad minim pentru care $m_i(\beta_i) = 0$ este:

$$m_i(x) = (x+\beta_i)\left(x+\beta_i^{2}\right)\ldots\left(x+\beta_i^{2^{q-1}}\right) \qquad (3.1.14)$$

Deşi în calculul lui $m_i(x)$ folosim coeficienţi $GF(2^q)$ coeficienţii lui $m_i(x)$ vor fi binari, fapt care iese în evidenţă şi din calculul acestor coeficienţi în cazul exemplului 1 prezentat mai jos.
Pentru corecţia unui număr de maxim t erori se impune alegerea unui număr de r = 2t rădăcini $\beta_i \in GF(2^q)$ ceea ce determină obţinerea unui cuvânt de cod având distanţa minimă $d \geq 2t + 1$.

În cazul codurilor ciclice binare, dacă α este o rădăcină şi α^2, $\alpha^{2\cdot 2}$, ... $\alpha^{2\cdot(q-1)}$ sunt tot rădăcini şi deci pentru a forma polinomul generator este suficient să luăm rădăcinile impare:

$$\beta_1 = \alpha,\ \beta_3 = \alpha^3,\ \dots,\ \beta_{2t-1} = \alpha^{2t-1}.$$

Structura matricii de control H în cazul codurilor BCH se determină deci impunând ca v(x) să aibă rădăcini pe $\beta_1 = \alpha,\ \beta_3 = \alpha^3,\ \dots,\ \beta_{2t-1} = \alpha^{2t-1}$.
Faptul că β_i este rădăcină a cuvântului de cod v(x) se exprimă prin $v(\beta_i) = 0$ sau mai explicit cu relaţia:

$$v(\beta_i) = v_{n-1}\beta_i^{n-1} + \dots + v_1\beta_i^1 + v_0\beta_i^0 = 0 \quad \text{cu } i = 1, 3, \dots, 2t-1 \tag{3.1.15}$$

Pentru fiecare i putem interpreta această relaţie ca pe un produs scalar al cuvântului format prin puterile lui α^i sau (β_i) şi cuvântul de cod:

$$\begin{bmatrix} \alpha^{i\cdot(n-1)} & \cdots & \alpha^{i\cdot 1} & \alpha^{i\cdot 0} \end{bmatrix} \begin{bmatrix} v_{n-1} \\ \vdots \\ v_1 \\ v_0 \end{bmatrix} = 0 \tag{3.1.16}$$

Înlocuind cu valorile corespunzătoare pentru i (i = 1, 3, ..., 2t-1) obţinem:

$$\begin{bmatrix} \alpha^{n-1} & \cdots & \alpha^1 & \alpha^0 \\ \alpha^{3\cdot(n-1)} & \cdots & \alpha^3 & \alpha^0 \\ \vdots & & & \\ \alpha^{(2t-1)\cdot(n-1)} & \cdots & \alpha^{2t-1} & \alpha^0 \end{bmatrix} \begin{bmatrix} v_{n-1} \\ \vdots \\ v_1 \\ v_0 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{3.1.17}$$

Deci matricea de control H în cazul codurilor BCH este de forma:

$$H = \begin{bmatrix} \alpha^{n-1} & \cdots & \alpha^2 & \alpha^1 & \alpha^0 \\ \alpha^{3\cdot(n-1)} & \cdots & \alpha^6 & \alpha^3 & \alpha^0 \\ \vdots & & & & \\ \alpha^{(2t-1)\cdot(n-1)} & \cdots & \alpha^{(2t-1)\cdot 2} & \alpha^{2t-1} & \alpha^0 \end{bmatrix} \tag{3.1.18}$$

Această matrice este formată din elementele GF(2^q) şi are t linii şi n coloane. Fiecare element poate fi exprimat printr-un cuvânt binar din q biţi şi deci vom putea scrie matricea H şi sub forma binară dispunând vertical cuvintele binare care înlocuiesc puterile lui α. Forma binară a matricii H va avea astfel q·t linii şi tot n coloane.

Exemplul 1:

Proiectarea unui cod BCH de lungime n = 15 şi corectând 2 erori cu determinarea polinomului generator şi a relaţiilor de codare.

$$n = 2^q - 1 = 15 \Rightarrow q = 4 \Rightarrow \text{extensia GF}(2^4)$$

Elementele lui GF(2^q) sunt clase de resturi modulo p(x) un polinom primitiv de gradul q = 4.

Un polinom p(x) este primitiv dacă este ireductibil şi binomul $x^n + 1$ pe care p(x) îl divide are cel puţin gradul $n = 2^q - 1$. Fie acest polinom primitiv $p(x) = x^4 + x + 1$. Putem să constatăm că $x^4 + x + 1$ divide pe $x^{15} + 1$ ($n = 2^q - 1 = 2^4 - 1 = 15$), dar nu divide nici un $x^n + 1$ cu $1 \le n < 15$ deci este primitiv. Se ştie că pentru orice câmp Galois $\alpha^n = 1$, deci la noi $\alpha^{15} = 1$.
Câmpul GF(2^4) generat de $p(x) = x^4 + x + 1$ obţinut printr-o rădăcină primitivă α a lui $x^4 + x + 1$ cu relaţia $\alpha^4 = \alpha + 1$ este cel dat în Anexa:

Pentru t = 2 se aleg rădăcinile $\beta_1 = \alpha$ si $\beta_3 = \alpha^3$, suntem în cazul binar şi alegem doar rădăcinile impare până la ordinul 2t-1 = 4-1 = 3).
Polinoamele minimale vor fi conform relaţiei (3.1.14):

$$m_1(x) = (x + \alpha)\cdot(x + \alpha^2)\cdot(x + \alpha^4)\cdot(x + \alpha^8) = x^4 + x^3(1 + \alpha^2 + \alpha^2 + 1 + \alpha + \alpha) + x^2(1+ \alpha + \alpha^2 + 1+ \alpha + \alpha^2 + \alpha^3 + \alpha^2 + \alpha^3 +\alpha + \alpha^3 + \alpha^3 + \alpha + \alpha^2) + x(1 + \alpha^3 + \alpha + \alpha^2 + \alpha^3 + 1 + \alpha^2 + \alpha^3 + 1 + \alpha + \alpha^3) + \alpha^{15} = x^4 + x + 1$$

$$m_3(x) = (x + \alpha^3)\cdot(x + \alpha^6)\cdot(x + \alpha^{12})\cdot(x + \alpha^{24}) = x^4 + x^3(\alpha^{24} + \alpha^{12} + \alpha^6 + \alpha^3) + x^2(\alpha^{36} + \alpha^{30} + \alpha^{18} + \alpha^{27} + \alpha^{15} + \alpha^9) + x(\alpha^{42} + \alpha^{39} + \alpha^{33} + \alpha^{21}) + \alpha^{45} = x^4 + x^3 + x^2 + x + 1$$

În aceste două relaţii puterile lui α mai mari decât 15 s-au exprimat în funcţie de α^{15}, iar suma s-a efectuat modulo doi.
Cele două polinoame fiind prime între ele şi folosind relaţia (3.1.13), g(x) se obţine:

$$g(x) = m_1(x)\cdot m_3(x) = (x^4 + x + 1)\cdot(x^4 + x^3 + x^2 + x + 1) = x^8 + x^7 + x^6 + x^5 + x^4 + x^5 + x^4 + x^3 + x^2 + x + x^4 + x^3 + x^2 + x + 1 = x^8 + x^7 + x^6 + x^4 + 1$$

$\Rightarrow$ m = 8 simboluri de control $\Rightarrow$ k = 15 - 8 = 7 simboluri de informaţie, adică g = 111010001. Cuvântul de cod va fi:

$$v = v_{14}v_{13}v_{12}v_{11}v_{10}v_9v_8v_7v_6v_5v_4v_3v_2v_1v_0$$

unde primii 7 sunt biţii de informaţie şi ultimii 8 cei de control.
Matricea de control H va avea 2 linii (t = 2) şi 15 coloane (n = 15):

$$H = \begin{bmatrix} \alpha^{14} & \alpha^{13} & \alpha^{12} & \alpha^{11} & \alpha^{10} & \alpha^9 & \alpha^8 & \alpha^7 & \alpha^6 & \alpha^5 & \alpha^4 & \alpha^3 & \alpha^2 & \alpha^1 & 1 \\ \alpha^{42} & \alpha^{39} & \alpha^{36} & \alpha^{33} & \alpha^{30} & \alpha^{27} & \alpha^{24} & \alpha^{21} & \alpha^{18} & \alpha^{15} & \alpha^{12} & \alpha^9 & \alpha^6 & \alpha^3 & 1 \end{bmatrix}$$

Luând puterile lui α modulo 15 şi ţinând cont că $\alpha^{15} = 1$ cu ajutorul tabelului câmpului GF(2^4) obţinem exprimarea binară a lui H:

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

adică q·t = 4·2 = 8 linii şi n = 15 coloane.

Folosind relaţia (3.1.9) se pot obţine relaţiile de codare.
Pentru o secvenţă informaţională i = 0000001 cuvântul de cod sistematic se poate determina cu relaţia (3.1.7):

$i(x) = 1$
$x^m\cdot i(x) = x^8$
$i(x)\cdot x^m/g(x) = x^8 / x^8 + x^7 + x^6 + x^4 + 1 \Rightarrow$
$\Rightarrow rest(i(x)\cdot x^m/g(x)) = x^7 + x^6 + x^4 + 1$
$\Rightarrow v(x) = x^8 + x^7 + x^6 + x^4 + 1$
adică v = 000000111010001

Decodarea BCH

În cazul codurilor ciclice binare pentru a putea corecta erori este suficient să determinăm poziţia acestora din expresia sindromului. Modul în care poziţia erorilor poate fi găsită cu ajutorul sindromului rezultă din prezentarea următoare.
Un cod ciclic corector de t erori are cuvinte de cod care au un număr r = 2t de rădăcini $\beta_i \in GF(2^q)$, $\beta_i=\alpha^i$:

$$v(\beta_i)= v(\alpha^i) = 0 \qquad (3.1.19)$$

La recepţie trebuie să verificăm legea de codare dată de relaţia (3.1.15). În cazul în care avem erori de tip aditiv putem exprima această lege sub forma:

$$r(\beta_i) = u(\beta_i) + e(\beta_i) = e(\beta_i) = e(\alpha^i) = S_i \qquad (3.1.20)$$

unde $e(\beta_i)$ este eroarea şi S_i este sindromul, iar i = 1, 3, ..., 2t-1 în cazul codurilor binare.

Pentru a vedea cum putem găsi poziţia erorii cu ajutorul sindromului luăm un mic exemplu:

Presupunem un cuvânt oarecare care are erori pe două poziţii de exemplu 1 şi 4 deci $e(x)=x^4+x$. Ţinând cont de relaţia (3.1.20) vom avea sindromul:

$$S_i = e(\alpha^i) = (\alpha^i)^1 + (\alpha^i)^4 = (\alpha^1)^i + (\alpha^4)^i = X_1^i + X_2^i$$

Expresia care indică poziţia erorii s-a notat cu X_k - **locatorul erorii**:

$$X_k = \alpha^j \text{ , } k = 1, \ldots t \text{ şi } j = 0, \ldots n-1 \qquad (3.1.21)$$

Deci avem atâţia locatori câte erori (în număr de t), iar erorile pot apărea pe orice poziţie în cadrul cuvântului (de la 0 la n-1).
Deci putem exprima sindromul S_i sub forma:

$$S_i = \sum_{k=1}^{t} X_k^i \qquad (3.1.22)$$

Această relaţie ne indică un sistem de ecuaţii neliniare care are ca necunoscute pe X_k (adică locatorii).

Pentru realizarea decodării va fi prezentat algoritmul Peterson cu căutare Chien care constă în:
1° Calcularea sindromului erorii

$$S_i = r\left(\alpha^i\right) = \sum_{k=1}^{t} X_k^i \quad \text{cu} \quad i=1, 3, \ldots 2t-1$$

2° Cu ajutorul locatorilor putem forma un polinom al erorilor care are ca rădăcini locatorii:

$$\sigma(x) = \prod_{k=1}^{t} (x + X_k) = x^t + \sigma_1 x^{t-1} + \ldots + \sigma_t \qquad (3.1.23)$$

Dacă punem condiţia ca X_k să fie rădăcină a lui $\sigma(x)$ obţinem:

$$X_k^t + \sigma_1 X_k^{t-1} + \ldots + \sigma_t = 0 \qquad (3.1.24)$$

Dacă înmulţim (3.1.24) cu X_k^i şi însumăm în funcţie de k:

$$\sum_{k=1}^{t} X_k^{t+i} + \sigma_1 \sum_{k=1}^{t} X_k^{t+i-1} + ... + \sigma_t \sum_{k=1}^{t} X_k^{i} = 0 \qquad (3.1.25)$$

Ţinând cont de (3.1.22) relaţia (3.1.25) devine:

$$S_{t+i} + \sigma_1 S_{t+i-1} + ... + \sigma_t S_i = 0 \qquad (3.1.26)$$

Această ecuaţie este un sistem liniar de t ecuaţii cu t necunoscute care se poate rezolva aplicând regula lui Cramer.

Pentru codurile BCH folosind şi relaţia $S_{2k} = S_k^2$ expresiile coeficienţilor σ_i în funcţie de S_i pentru un număr de 1, 2 sau 3 erori sunt:

Tabelul 3.1.1

t	σ_i
1	$\sigma_1 = S_1$
2	$\sigma_1 = S_1$ $\sigma_2 = \dfrac{S_3 + S_1^3}{S_1}$
3	$\sigma_1 = S_1$ $\sigma_2 = \dfrac{S_1^2 S_3 + S_5}{S_1^3 + S_3}$ $\sigma_3 = S_1^3 + S_3 + S_1 \sigma_2$

3° Căutarea poziţiei eronate.

Cunoscând σ_i putem determina poziţia erorii.

Dacă împărţim relaţia (3.1.24) cu X_k^t rezultă relaţia:

$$\sum_{i=1}^{t} \sigma_i X_k^{-i} = 1 \qquad (3.1.27)$$

i dându-ne poziţia eronată.

Numărul maxim de erori corectabile este t eroarea putându-se găsi pe oricare din cele n poziţii ale cuvântului.

În cazul unei căutări Chien se începe căutarea simbolului eronat de la r_{n-1}. Pentru o poziţie oarecare n-j ţinând cont de (3.1.21) vom avea $X_k = \alpha^{n-j}$

$$\begin{aligned} &\sum_{i=1}^{t} \sigma_i \alpha^{-i(n-j)} = 1 \Rightarrow \sum_{i=1}^{t} \sigma_i \alpha^{-i \cdot n} \cdot \alpha^{i \cdot j} = 1 \\ &\Rightarrow \sum_{i=1}^{t} \sigma_i \left(\frac{1}{\alpha^n} \right)^i \cdot \alpha^{i \cdot j} = 1 \end{aligned} \qquad (3.1.28)$$

Deoarece $\alpha^n = 1$ obţinem ecuaţia de căutare Chien:

$$\sum_{i=1}^{t} \sigma_i \alpha^{i \cdot j} = 1 \quad \text{cu} \quad j = 1,...n \tag{3.1.29}$$

lui j = 1 corespunzându-i simbolul r_{n-1}, iar lui j = n simbolul r_0.
Deci conform ecuaţiei (3.1.29) eroarea apare când suma respectivă este egală cu 1.

Exemplul 2:

Presupunem că recepţionăm un cuvânt BCH cu n = 15 şi k = 7 şi că avem t = 2 erori.

$$r = 110101011010011$$
$$\Rightarrow r(x) = x^{14} + x^{13} + x^{11} + x^9 + x^7 + x^6 + x^4 + x^1 + 1$$

Calculăm sindromul. Pentru t = 2 avem de calculat până la S_3 inclusiv (2t-1 = 2·2-1 = 3):

$$S_1 = r(\alpha) = \alpha^{14} + \alpha^{13} + \alpha^{11} + \alpha^9 + \alpha^7 + \alpha^6 + \alpha^4 + \alpha + 1=$$
$$= 1 + \alpha^3 + 1 + \alpha^2 + \alpha^3 + \alpha + \alpha^2 + \alpha^3 + \alpha + \alpha^3 + 1 + \alpha +$$
$$+ \alpha^3 + \alpha^2 + \alpha^3 + 1 + \alpha + \alpha + 1 = \alpha^2 + \alpha + 1 = \alpha^{10}$$

$$S_3 = r(\alpha^3) = \alpha^{42} + \alpha^{39} + \alpha^{33} + \alpha^{27} + \alpha^{21} + \alpha^{18} + \alpha^{12} + \alpha^3$$
$$+ 1= \alpha^{12} + \alpha^9 + \alpha^3 + \alpha^{12} + \alpha^6 + \alpha^3 + \alpha^{12} + \alpha^3 + 1 = \alpha^9 +$$
$$+ \alpha^6 + \alpha^{12} + \alpha^3 + 1 = \alpha + \alpha^3 + \alpha^2 + \alpha^3 + 1 + \alpha + \alpha^2 + \alpha^3 +$$
$$+ \alpha^3 + 1 = 0$$

Din tabelul 3.1.1 avem pentru t = 2:

$$\sigma_1 = S_1 = \alpha^{10}$$
$$\sigma_2 = \frac{S_3 + S_1^3}{S_1} = \alpha^{20} = \alpha^5$$

În calculele anterioare am utilizat valorile pentru puterile lui α din tabelul 1 şi puterile care depăşesc α^{15} le-am exprimat in funcţie de aceasta ţinând cont că $\alpha^{15} = 1$.
Folosim relaţia de căutare (3.1.29) şi avem:

$j = 1 \Rightarrow \sigma_1\alpha^{1\cdot1} + \sigma_2\alpha^{2\cdot1} = \alpha^{10}\cdot\alpha + \alpha^5\cdot\alpha^2 = \alpha + \alpha^2 + \alpha^3 + 1 + \alpha + \alpha^3 = 1 + \alpha^2 = \alpha^8$
$j = 2 \Rightarrow \sigma_1\alpha^{1\cdot2} + \sigma_2\alpha^{2\cdot2} = \alpha^{10}\cdot\alpha^2 + \alpha^5\cdot\alpha^4 = \alpha^8$
$j = 3 \Rightarrow \sigma_1\alpha^{1\cdot3} + \sigma_2\alpha^{2\cdot3} = \alpha^{10}\cdot\alpha^3 + \alpha^5\cdot\alpha^6 = \alpha^4$
$j = 4 \Rightarrow \sigma_1\alpha^{1\cdot4} + \sigma_2\alpha^{2\cdot4} = \alpha^{10}\cdot\alpha^4 + \alpha^5\cdot\alpha^8 = \alpha^2$
$j = 5 \Rightarrow \sigma_1\alpha^{1\cdot5} + \sigma_2\alpha^{2\cdot5} = \alpha^{10}\cdot\alpha^5 + \alpha^5\cdot\alpha^{10} = 0$
$j = 6 \Rightarrow \sigma_1\alpha^{1\cdot6} + \sigma_2\alpha^{2\cdot6} = \alpha^{10}\cdot\alpha^6 + \alpha^5\cdot\alpha^{12} = \alpha^5$
$j = 7 \Rightarrow \sigma_1\alpha^{1\cdot7} + \sigma_2\alpha^{2\cdot7} = \alpha^{10}\cdot\alpha^7 + \alpha^5\cdot\alpha^{14} = \alpha^{10}$
$j = 8 \Rightarrow \sigma_1\alpha^{1\cdot8} + \sigma_2\alpha^{2\cdot8} = \alpha^{10}\cdot\alpha^8 + \alpha^5\cdot\alpha^{16} = \alpha^2$
$j = 9 \Rightarrow \sigma_1\alpha^{1\cdot9} + \sigma_2\alpha^{2\cdot9} = \alpha^{10}\cdot\alpha^9 + \alpha^5\cdot\alpha^{18} = \alpha^5$
$j = 10 \Rightarrow \sigma_1\alpha^{1\cdot10} + \sigma_2\alpha^{2\cdot10} = \alpha^{10}\cdot\alpha^{10} + \alpha^5\cdot\alpha^{20} = 1$ deci este eronat simbolul n - j = 15 – 10 = 5 (r_5)
$j = 11 \Rightarrow \sigma_1\alpha^{1\cdot11} + \sigma_2\alpha^{2\cdot11} = \alpha^{10}\cdot\alpha^{11} + \alpha^5\cdot\alpha^{22} = \alpha^4$
$j = 12 \Rightarrow \sigma_1\alpha^{1\cdot12} + \sigma_2\alpha^{2\cdot12} = \alpha^{10}\cdot\alpha^{12} + \alpha^5\cdot\alpha^{24} = \alpha$
$j = 13 \Rightarrow \sigma_1\alpha^{1\cdot13} + \sigma_2\alpha^{2\cdot13} = \alpha^{10}\cdot\alpha^{13} + \alpha^5\cdot\alpha^{26} = \alpha^{10}$
$j = 14 \Rightarrow \sigma_1\alpha^{1\cdot14} + \sigma_2\alpha^{2\cdot14} = \alpha^{10}\cdot\alpha^{14} + \alpha^5\cdot\alpha^{28} = \alpha$
$j = 15 \Rightarrow \sigma_1\alpha^{1\cdot15} + \sigma_2\alpha^{2\cdot15} = \alpha^{10}\cdot\alpha^{15} + \alpha^5\cdot\alpha^{30} = 1$ deci este eronat simbolul n - j = 15 – 15 = 0 (r_0)

Cuvântul decodat va fi deci 11010101***1***1001***0*** .
Secvenţa de informaţie de la ieşirea decodorului va fi i = 1101010.

Desfăşurarea lucrării:
a) Rulaţi programul pe calculator, utilizând opţiunea demonstrativă atât pentru algoritmul de codare cât şi pentru cel de decodare. Se urmăreşte pas cu pas algoritmul de codare existând posibilitatea de a alege un cod corector de t = 1, 2, sau 3 erori. Urmăriţi cu atenţie la fiecare pas mesajele calculatorului care va vor ghida în rularea corectă a programului. De asemenea se urmăreşte pas cu pas algoritmul de decodare.
b) Realizaţi o codare a unui cuvânt în cazul unui cod corector de t = 1, 2, sau 3 erori alegând şirul de biţi de informaţie de lungimea corespunzătoare. Pentru cazul t = 2 efectuaţi apoi decodarea unui cuvântului de cod. Verificaţi apoi calculele cu ajutorul programului de pe calculator.
c) La sfârşitul lucrării de laborator se va efectua cu ajutorul programului un test asupra cunoştinţelor acumulate. Testul cuprinde 5 întrebări teoretice fiecare cu un răspuns corect din cinci propuse, o codare şi o decodare a unui mesaj pe care calculatorul îl generează în mod aleator.

3.2 Codul Reed-Solomon

Codurile Reed-Solomon (RS) fac parte din categoria codurilor ciclice, însă sunt coduri nebinare. Spre deosebire de celelalte coduri ciclice, alfabetul codului RS nu este câmpul binar {0, 1}, ci un câmp finit de ordin superior, numit câmp Galois şi care va fi descris în Anexa. În acest fel, cuvintele codului RS nu sunt secvenţe (succesiuni) de biţi, ci de caractere. Aceste caractere pot fi reprezentate, la rândul lor, prin secvenţe binare, însă sunt indivizibile din punct de vedere al codării şi decodării Reed-Solomon.

Structural, cuvintele de cod RS au aceeaşi alcătuire ca şi cele de cod ciclic:

$$v = v_{n-1}v_{n-2} \dots v_1v_0 \qquad v_j \in GF(2^q, p(x)) \quad j = 0 \div n-1 \tag{3.2.1}$$

unde: v – cuvântul de cod, format din n caractere;
$v_{n-1}v_{n-2} \dots v_k$ – caracterele de informaţie, în număr de k;
$v_{k-1}v_{k-2} \dots v_0$ – caracterele de control, în număr de m;
q – ordinul câmpului;
p(x) – polinomul generator al câmpului GF.
Relaţia de codare are aceeaşi formă ca şi la codurile ciclice:

$$v(x) = i(x)\cdot x^m + \text{rest}\,(i(x)\cdot x^m / g(x)) \tag{3.2.2}$$

unde g(x) este polinomul generator al codului, al cărui construcţie este prezentată mai jos, iar

$$i(x) = v_{n-1}\cdot x^{k-1} + v_{n-2}\cdot x^{k-2} + \dots + v_{k+1}\cdot x + v_k \tag{3.2.3}$$

este polinomul de informaţie.

Prin relaţia de codare (3.2.2) se obţine polinomul ataşat cuvântului de cod, polinom ai cărui coeficienţi sunt tocmai caracterele ce alcătuiesc cuvântul de cod dat de (3.2.1). Relaţia (3.2.2) indică, deasemenea, că v(x) este un multiplu al lui g(x).

Codul RS, având parametrii n, k şi m, construit după relaţia (3.2.2), este capabil să corecteze un număr e_c de caractere eronate, unde:

$$2\cdot e_c = m = n-k \tag{3.2.4}$$

La decodare, spre deosebire de codurile ciclice, într-un cuvânt de cod RS recepţionat, în vederea corecţiei, este necesară atât localizarea erorii, cât şi stabilirea valorii ei.

Polinomul generator, g(x), al codului

Pentru a se corecta t erori dintr-un cuvânt este necesar a se preciza poziţia fiecăreia precum şi valoarea ei. Dacă ne referim la un cuvânt de lungime n, unde:

$$n = 2^q - 1 \tag{3.2.5}$$

atunci informaţia necesară pentru a preciza un caracter eronat între cele n este:

$$i_{p1} = -\log_2(1/n) \tag{3.2.6}$$

Pentru a include şi varianta "cuvânt fără eroare", considerăm că în acest caz se "eronează" un al n+1–lea caracter, fictiv, astfel încât informaţia necesară pentru a preciza poziţia unui caracter eronat (dintre n+1 caractere) este:

$$i_{p1} = \log_2(n+1) \tag{3.2.7}$$

Această informaţie poate fi conţinută de un caracter din $GF(2^q)$. Deasemenea şi valoarea erorii, ε, poate să fie orice caracter din $GF(2^q)$:

$$w = v + \varepsilon \tag{3.2.8}$$

unde: w –caracterul recepţionat $\in GF(2^q)$;

v –caracterul emis $\in GF(2^q)$;

ε -valoarea erorii $\in GF(2^q)\backslash\{0\}$.

Incluzând şi cazul "eronare a caracterului fictiv", rezultă că ε poate lua orice valoare din $GF(2^q)$, adică 2^q valori posibile. Informaţia necesară pentru a preciza valoarea ei este identică cu cea dată de (3.2.7).

În concluzie, pentru fiecare eroare ce se doreşte a fi corectată este necesară o informaţie egală cu 2q biţi, adică două caractere din $GF(2^q)$. La t erori sunt necesare 2t caractere (cantitate de informaţie).

Obs. *În fapt condiţia anterioară este una suficientă, cea necesară implică mai puţină informaţie deoarece nu se poate erona un caracter de două ori. De exemplu în cazul a două erori:*

$$i_{p2} = -\log_2 \frac{1}{C_{n+1}^2} = \log_2 \frac{(n+1)\cdot n}{2} = \log_2(n+1) + \log_2 n - 1 \tag{3.2.9}$$

iar pentru n suficient de mare $i_{p2} \approx 2i_{p1} - 1$.

Cele 2t caractere de informaţie necesare soluţionării problemei corecţiei se află din 2t ecuaţii, care înseamnă tot atâtea legături (proprietăţi) pentru cuvântul receptionat. Aceste 2t proprietăţi pentru cuvintele de cod RS sunt generate prin relaţia de codare (3.2.2). Prin această relaţie v(x) devine multiplul lui g(x), ceea ce înseamnă că rădăcinile lui g vor fi şi rădăcini pentru v. Rezultă necesitatea ca g să aibă 2t rădăcini. Aceste rădăcini pot fi oricare dintre cele 2^q elemente ale câmpului $GF(2^q)$. Vom alege pentru g rădăcinile $\alpha, \alpha^2, \alpha^3, \ldots \alpha^{2t}$, datorită simplităţii şi simetriei:

$$g(x) = (x+\alpha)(x+\alpha^2)\ldots(x+\alpha^{2t}) = x^m + g_{m-1}x^{m-1} + \ldots + g_1 x + g_0 \tag{3.2.10}$$

Aşadar cuvântul de cod RS, v, rezultat prin codarea cu ajutorul relaţiei (3.2.2), în care g este dat de (3.2.10), are proprietatea că elementele $\alpha, \alpha^2, \alpha^3, \ldots \alpha^{2t}$ sunt rădăcini pentru polinomul ataşat, v(x).

Codarea codului Reed– Solomon

Codarea se poate face utilizând relaţia (3.2.2).
Spre exemplu în cazul codului RS cu n = 7, k = 5, t = 1 având polinomul generator:

$$g(x) = (x + \alpha)(x + \alpha^2) = x^2 + \alpha^4 x + \alpha^3 = x^2 + 5x + 4 \qquad (3.2.11)$$

Ecuaţia împărţirii este:

$$i(x)\cdot x^2 = (2x^4 + 5x^3 + x^2 + x + 5)(x^2 + 5x + 4) + x + 1 \qquad (3.2.12)$$

unde: $-i(x)\cdot x^2$ este deîmpărţitul ($i = [2\ 1\ 7\ 5\ 4] \Rightarrow i(x) = 2x^4 + x^3 + 7x^2 + 5x + 4$);
$-2x^4 + 5x^3 + x^2 + x + 5$ este câtul;
$-g(x) = x^2 + 5x + 4$ este împărţitorul, iar
$- x + 1$ este restul.

Împărţirea polinomului $i(x)\cdot x^m = i(x)\cdot x^2$ la g(x), cerută pentru codare de relaţia (3.2.2), este prezentată mai jos:

$$\begin{array}{l|l} 2x^6 + x^5 + 7x^4 + 5x^3 + 4x^2 & x^2 + 5x + 4 \\ \underline{2x^6 + 6x^5 + 5x^4} & \hline 2x^4 + 5x^3 + x^2 + x + 5 \\ / \quad 5x^5 + 4x^4 + 5x^3 + 4x^2 & \\ \quad \underline{5x^5 + 2x^4 + x^3} & \\ \quad / \quad 2x^4 + 6x^3 + 4x^2 & \\ \quad\quad \underline{x^4 + 5x^3 + 4x^2} & \\ \quad\quad / \quad x^3 & \\ \quad\quad\quad \underline{x^3 + 5x^2 + 4x} & \\ \quad\quad\quad / \quad 5x^2 + 4x & \\ \quad\quad\quad\quad \underline{5x^2 + 2x + 1} & \\ \quad\quad\quad\quad / \quad\quad x + 1 & \end{array}$$

Cuvântul de cod, conform relaţiei (3.2.2) rezultă:

$$v(x) = i(x)\cdot x^2 + x + 1 = 2x^6 + x^5 + 7x^4 + 5x^3 + 4x^2 + x + 1 \qquad (3.2.13)$$

Decodarea codului Reed-Solomon

Decodarea codului RS poate fi făcută atât în timp cât şi în frecvenţă

Decodarea codului RS-corector de erori multiple

Decodarea va fi sistematizată sub forma unor paşi algoritmici. La fiecare pas se va prezenta operaţia necesară a fi executată, scopul urmărit cât şi argumentările necesare. Fie aşadar codul RS corector de t erori.

Pasul I

Conform relaţiei (3.2.10) g(x) este un polinom de grad $k = 2t > 2$. Prin construcţie, cuvintele de cod RS au proprietatea că polinoamele ataşate lor sunt divizibile cu g(x), adică elementele câmpului $GF(2^2)$ $\alpha, \alpha^2, \ldots, \alpha^{2t}$ sunt rădăcini atât pentru g(x) cât şi pentru orice cuvânt de cod v(x):

$$\begin{cases} g(\alpha^j) = 0 \\ v(\alpha^j) = 0 \qquad j = 1 \div 2t \end{cases} \qquad (3.2.14)$$

Aceste proprietăţi constituie şi punctul de plecare în decodare. Presupunând că w este un cuvânt recepţionat:

$$w = v + \varepsilon \quad \text{sau} \quad w(x) = v(x) + \varepsilon(x) \tag{3.2.15}$$

vom calcula 2t coeficienţi, numiţi coeficienţi sindrom, Sj, în forma:

$$S_j = w(\alpha^j) = v(\alpha^j) + \varepsilon(\alpha^j) = \varepsilon(\alpha^j) \qquad j=1\div 2t \tag{3.2.16}$$

Evident că dacă nu există erori $S_j = 0$. În acest caz se trece la pasul VI. Evident concluzia poate fi eronată. Un exemplu în argumentarea acestei afirmaţii este situaţia: ε = cuvânt de cod. Dar în acest caz numărul erorilor depăşeşte puterea de corecţie de t erori.

Pasul II

Dacă există erori în limitele corectabile (numărul erorilor este mai mic sau egal cu t) atunci există coeficienţi sindrom diferiţi de zero. Fie cuvântul eroare în forma:

$$\varepsilon(x) = \sum_{i=1}^{t} {}^{r_i} \cdot x^{k_i} \tag{3.2.17}$$

unde:

$$Y_i = \alpha^{r_i} \tag{3.2.18}$$

reprezintă valoarea erorii $r_i \in \{0, 1, 2, \ldots n-1\}$, iar:

$$X_i = \alpha^{k_i} \tag{3.2.19}$$

reprezintă locatorul erorii $k_i \in \{0, 1, 2, \ldots, n-1\}$. Cu aceste notaţii coeficienţii sindrom au expresiile:

$$S_j = \sum_{i=1}^{t} Y_i \cdot \left(\alpha^j\right)^{k_i} = \sum_{i=1}^{t} Y_i \cdot X_i^j, \qquad j = 1\div 2t \tag{3.2.20}$$

Ecuaţiile (3.2.20) reprezintă un sistem de 2t ecuaţii cu 2t necunoscute: t locatori ai erorilor X_i şi t valori pentru respectivele erori Y_i.

Rezolvarea acestui sistem de ecuaţii se va face în mai multe etape. La pasul prezent se vor calcula coeficienţii polinomului σ(x) ai cărui rădăcini sunt locatorii erorilor:

$$\sigma(x) = \sum_{i=1}^{t} (x + X_i) = x^t + \sigma_{\ 1} \cdot x^{t-1} + \ldots + \sigma_{t-1} \cdot x + \sigma_t \tag{3.2.21}$$

Pentru că Xi , $1 \le i \le t$ este o rădăcină a lui σ(x) putem scrie:

$$X_i^t + \sigma_1 \cdot X_i^{t-1} + \ldots + \sigma_{t-1} \cdot X_i + \sigma_t = 0, \quad i = 1\div t \tag{3.2.22}$$

Înmulţind ecuaţiile (3.2.22) pe rând cu $X_i^k Y_i$ şi sumându-le obţinem ecuaţia:

$$\sum_{i=1}^{t} Y_i \cdot X_i^{t+k} + \sigma_1 \cdot \sum_{i=1}^{t} Y_i X_i^{t+k-1} + \ldots + \sigma_{t-1} \cdot \sum_{i=1}^{t} Y_i \cdot X_i^{k+1} + \\ + \sigma_t \cdot \sum_{i=1}^{t} Y_i \cdot X_i^k = 0 \tag{3.2.23}$$

sau, ţinând cont de (3.2.20) pentru k luând valorile 1, 2, ..., t:

$$S_{t+k} + \sigma_1 \cdot S_{t+k-1} + \ldots + \sigma_{t-1} \cdot S_{k-1} + \sigma_t \cdot S_k = 0 , \quad k = 1 \div t \tag{3.2.24}$$

Ecuaţiile (3.2.24) reprezintă un sistem de t ecuaţii cu t necunoscute (coeficienţii σ_i) a cărui rezolvare constituie obiectivul acestui pas algoritmic. Ecuaţiile (3.2.24) pot fi puse sub forma compactă:

$$A_s \cdot \sigma = B_s \tag{3.2.25}$$

unde:

$$A_s = \begin{bmatrix} S_t & S_{t-1} & \ldots & S_1 \\ S_{t+1} & S_t & \ldots & S_2 \\ \ldots & \ldots & \ldots & \ldots \\ S_{2t-1} & S_{2t-2} & \ldots & S_t \end{bmatrix}; \quad \sigma = \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \ldots \\ \sigma_t \end{bmatrix}; \quad B_s = \begin{bmatrix} S_{t+1} \\ S_{t+2} \\ \ldots \\ S_{2t} \end{bmatrix} \tag{3.2.26}$$

Calculând inversa matricii A_s găsim soluţia sistemului (3.2.24) în forma:

$$\sigma = A_s^{-1} \cdot B_s \tag{3.2.27}$$

Obs: *Toate calculele trebuiesc făcute în câmpul GF(2²), atât coeficienţii sindrom, S_j, cât şi coeficienţii σ fiind elemente ale respectivului câmp.*

În rezolvarea ecuaţiei (3.2.25) pot apărea trei situaţii:

1° rangul matricii As este e < t şi este egal cu al matricii [AsBs]. În acest caz numărul de erori este e şi din rezolvarea ec (3.2.25) rezultă un număr e de coeficienţi σ_l nenuli Rezolvarea ecuaţiei (3.2.25) presupune restrângerea sistemului (10.24) la un număr e < t de ecuaţii cu e necunoscute, rezolvabil.

2° rangul matricii As este t. În acest caz există As^{-1} iar ecuaţia (3.2.25) are soluţie dată prin relaţia (3.2.27).Se vor găsi t erori în acest caz.

3° rangul matricii As este e` <t şi este mai mic decât al matricii [AsBs]. O astfel de situaţie este posibil să apară dacă numărul erorilor depăşeşte t. În acest caz se semnalează prezenţa erorilor în număr necorectabil. Funcţie de aplicaţie se va abandona cuvântul în cauză sau se va cere retransmisia sa.

Pasul III

Ecuaţia (3.2.22) se poate rescrie în forma:

$$\sum_{j=1}^{t} \sigma_j \cdot X_i^{-j} = 1 \tag{3.2.28}$$

Ştiind că X_i este de forma $X_i = \alpha^{k_i}$ unde k_i indică rangul pe care îl ocupă eroarea (ex: k_i = n-1 este prima poziţie) se vor putea afla locatorii erorilor printr-o operaţie de căutare:

$$\sum_{j=1}^{t} \sigma_j \cdot \alpha^{k \cdot j} = 1 \quad ? \qquad k = 1, 2, \ldots, n \tag{3.2.29}$$

Acei k pentru care (3.2.29) este o identitate, indică prezenţa erorii pe poziţia:

$$r = n - k \tag{3.2.30}$$

Obs: *Înlocuind pe:*

$$X_r = \alpha^r \qquad (3.2.31)$$

în (3.2.28) şi utilizând identitatea $\alpha^n = 1$ *obţinem:*

$$\sum_{j=1}^{t} \sigma_j \cdot \alpha^{-j \cdot r} = \sum_{j=1}^{t} \sigma_j \cdot \alpha^{n \cdot j} \alpha^{-j \cdot r} = \sum_{j=1}^{t} \sigma_j \cdot \alpha^{(n-j) \cdot r} = \sum_{j=1}^{t} \sigma_j \cdot \alpha^{k \cdot jr} = 1$$

Pasul IV

Dispunând de poziţiile erorilor dispunem implicit de numărul lor. Reţinem că problema are soluţie doar dacă e < t. Cunoscând aşadar e locatori ai erorilor în forma $X_i = \alpha^{k_i}$, i = 1÷e, din sistemul de ecuaţii (3.2.20) se reţin e ecuaţii în vederea aflării valorilor Y_i pentru cele e erori. Acest sistem este compatibil unic determinat. Rezolvarea sa conduce la aflarea celor e valori necesare Y_i.

Pasul V

Cunoscând atât poziţiile erorilor $X_i = \alpha^i$, i = 1÷e, cât şi valorile lor $Y_i = \alpha^{r_i}$, i=1÷e putem face corecţia caracterelor eronate:

$$v_{k_i} = w_{k_i} + Y_i, \qquad i = 1 \div e \qquad (3.2.32)$$

Pasul VI

Se face selecţia caracterelor de informaţie şi livrarea lor la ieşire.

Desfăşurarea lucrării:

a) Rulaţi programul pe calculator, utilizând opţiunea demonstrativă atât pentru algoritmul de codare cât şi pentru cel de decodare. Se urmăreşte pas cu pas algoritmul de codare existând posibilitatea de a alege un cod corector de t = 1, sau 2 erori. Urmăriţi cu atenţie la fiecare pas mesajele calculatorului care va vor ghida în rularea corectă a programului. De asemenea se urmăreşte pas cu pas algoritmul de decodare.

b) Realizaţi o codare a unui cuvânt în cazul unui cod corector de t = 1, sau 2 erori alegând şirul de caractere de informaţie de lungimea corespunzătoare. Pentru cazul t = 1 şi 2 efectuaţi apoi decodarea unui cuvântului de cod. Verificaţi apoi calculele cu ajutorul programului de pe calculator.

L4. Coduri convoluţionale

Codurile convoluţionale au fost introduse în 1954 de P. Elias şi reprezintă o clasă de coduri corectoare având o mare aplicabilitate practică.

În cazul codurilor convoluţionale, fiecare bloc de *n* simboluri binare de la ieşirea codorului depinde atât de blocul de *k* simboluri binare prezent la intrarea sa, la momentul considerat, cât şi de *m* blocuri precedente. În consecinţă codurile convoluţionale introduc un efect de memorie de ordinul *m*. Cantitatea $K=m+1$ se numeşte lungimea de constrângere a codului.

Codorul este constituit dintr-un sistem de *m* registre de întârziere, fiecare având o capacitate de *k* biţi, care memorează cele *m* blocuri de *k* simboluri de informaţie, dintr-o mulţime de funcţii liniare, ce generează blocurile de *n* simboluri de la ieşire şi dintr-un convertor paralel-serie. Mărimea $R=k/n$ se numeşte randamentul codului.

Un cod convoluţional de randament *R* este o aplicaţie de la mulţimea matricilor (binare) cu un număr de *k* linii şi număr infinit de coloane către mulţimea matricilor (binare) cu un număr de *n* linii şi număr infinit de coloane, unde $n > k$:

$$\mathbf{C} : \mathbf{M}_{k\times\infty} \rightarrow \mathbf{M}_{n\times\infty} \tag{4.1}$$

Astfel, prin transformarea $\mathbf{C}$, fiecărei matrici $\mathbf{I} \in \mathbf{M}_{k\times\infty}$, de forma :

$$\mathbf{I} = \begin{bmatrix} i_{01} & i_{11} & i_{21} & \cdots & i_{j1} & \cdots \\ i_{02} & i_{12} & i_{22} & \cdots & i_{j2} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ i_{0k} & i_{1k} & i_{2k} & \cdots & i_{jk} & \cdots \end{bmatrix}, \qquad i_{js} \in \{0,1\} \;\; \forall\, \mathrm{j} = \overline{0,\infty}, \;\; \mathrm{s} = \overline{1,\mathrm{k}} \tag{4.2}$$

i se ataşează o matrice $\mathbf{V} \in \mathbf{M}_{n\times\infty}$, de forma:

$$\mathbf{V} = \begin{bmatrix} a_{01} & a_{11} & a_{21} & \cdots & a_{j1} & \cdots \\ a_{02} & a_{12} & a_{22} & \cdots & a_{j2} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{0k} & a_{1k} & a_{2k} & \cdots & a_{jk} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{0n} & a_{1n} & a_{2n} & \cdots & a_{jn} & \cdots \end{bmatrix}, \quad a_{js} \in \{0,1\} \;\; \forall\, \mathrm{j} = \overline{0,\infty}, \;\; \mathrm{s} = \overline{1,\mathrm{n}} \tag{4.3}$$

Matricea **I** conţine practic biţii de informaţie, în ordinea $i_{01}\, i_{02} \ldots \mathrm{i}_{0k}\, \mathrm{i}_{11} \ldots$, iar matricea **V** secvenţa codată : $a_{01}\, a_{02} \ldots \mathrm{a}_{0n}\, \mathrm{a}_{11} \ldots$. Dacă $a_{js} \equiv i_{js}$ pentru orice *j* pozitiv şi pentru orice $s = 1 \div k$, atunci codul se numeşte sistematic. Făcând apel la o reprezentare polinomială, matricile **I** şi **V** se pot scrie ca :

$$\mathbf{I}(\mathrm{D}) = \begin{bmatrix} \sum_{s=0}^{\infty} \mathrm{i}_{s1} \mathrm{D}^s \\ \sum_{s=0}^{\infty} \mathrm{i}_{s2} \mathrm{D}^s \\ \vdots \\ \sum_{s=0}^{\infty} \mathrm{i}_{sk} \mathrm{D}^s \end{bmatrix}, \quad \mathbf{V}(\mathrm{D}) = \begin{bmatrix} \sum_{s=0}^{\infty} \mathrm{a}_{s1} \mathrm{D}^s \\ \sum_{s=0}^{\infty} \mathrm{a}_{s2} \mathrm{D}^s \\ \vdots \\ \sum_{s=0}^{\infty} \mathrm{a}_{sk} \mathrm{D}^s \\ \vdots \\ \sum_{s=0}^{\infty} \mathrm{a}_{sn} \mathrm{D}^s \end{bmatrix} \tag{4.4}$$

Cu aceste notaţii, relaţia de codare, poate fi scrisă astfel :

$$\mathbf{V}(\mathrm{D}) = \mathbf{G}(\mathrm{D}) \cdot \mathbf{I}(\mathrm{D}) \tag{4.5}$$

unde **G**(D) se numeşte matricea generatoare a codului şi are forma:

$$\mathbf{G}(\mathrm{D}) = \begin{bmatrix} g_{11}(D) & g_{12}(D) & \cdots & g_{1k}(D) \\ \cdots & \cdots & \cdots & \cdots \\ g_{n1}(D) & g_{n1}(D) & \cdots & g_{nk}(D) \end{bmatrix} \tag{4.6}$$

În funcţie de felul polinoamelor gebneratoare $g_{js}(D)$, codurile convoluţionale pot fi clasificate ca şi:

a) Coduri *sistematice* sau *nesistematice*. Dacă primele k linii din **G**(D) formează matricea unitate de ordinul k, I_k, atunci codurile sunt sistematice. În acest caz biţii din **I** se vor regăsi printre biţii din **V**. Astfel, dacă cele k simboluri de informaţie, prezente sunt efectiv emise, adică se găsesc în mod explicit în blocul de n simboluri de la ieşirea codorului pe primele k poziţii, codul se numeşte cod sistematic. Pentru codurile nesistematice, biţii din **V** sunt combinaţii liniare ale biţilor din **I**, neexistând biţi de informaţie şi de control ca şi în cazul precedent.

b) Coduri *recursive* sau *nerecursive*. Dacă toate polinoamele generatoare care compun **G**(D) sunt finite, atunci codul rezultat este nerecursiv. În caz contrar polinoamele generatoare $g_{js}(D)$ pot fi scrise sub forma:

$$g_{js}(D) = \frac{a_{js}(D)}{b_{js}(D)} \tag{4.7}$$

unde polinoamele a_{js} şi b_{js} sunt finite. Deci, dacă există cel puţin un polinom $b_{js}(D) \neq 1$, atunci codul este recursiv.

Lungimea de constrângere este unul dintre parametri importanţi ai codurilor convoluţionale. O altă definiţie a sa este dată în relaţia de mai jos:

$$K = 1 + \max_{j,s} grad\{a_{j,s}(D), b_{j,s}(D)\} \tag{4.8}$$

Pentru a ilustra codurile convoluţionale nerecursive şi nesistematice, în Fig. 4.1. se prezintă un exemplu de codor convoluţional de randament R=1/2 şi de lungime de constrângere K=(m+1)=3. Intrarea sa este constituită din blocurile de k=1 simbol şi ieşirea sa de blocurile de n=2 simboluri.

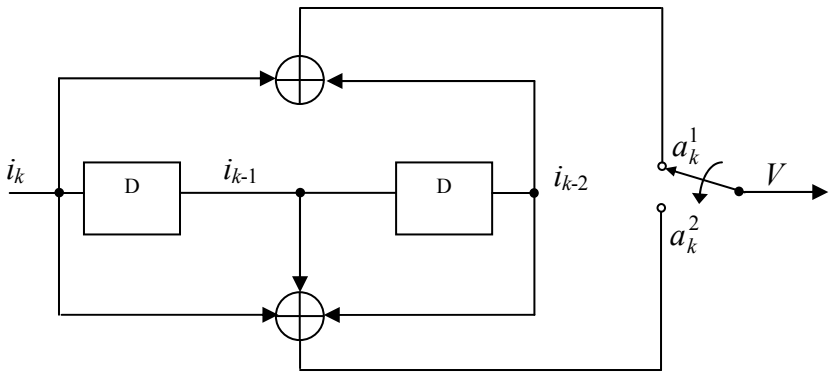


Fig. 4.1. Exemplu de codor convoluţional nerecursiv, nesistematic (R = 1/2, K = 3).

Relaţiile de calcul ale secvenţelor de la ieşire sunt:

$$a_k^1 = \sum_{j=0}^{2} i_{k-j} g_j^1 \ , \ a_k^2 = \sum_{j=0}^{2} i_{k-j} g_j^2 \tag{4.9}$$

Cele două secvenţe generatoare sunt:

$$\begin{aligned} g^1 &= \left[g_0^1, g_1^1, g_2^1\right] = [1,0,1] \\ g^2 &= \left[g_0^2, g_1^2, g_2^2\right] = [1,1,1] \end{aligned} \tag{4.10}$$

Observăm că ieşirile codorului fiind egale cu o combinaţie liniară a simbolurilor de informaţie, codul este liniar. Codurile convoluţionale sunt de asemenea definite pornind de la polinoamele lor generatoare exprimate în funcţie de variabila D (întârziere) echivalentă cu variabila Z^{-1} a transformatei Z. Considerând tot exemplul din Fig. 4.1, polinoamele generatoare ale acestui cod au expresiile:

$$\begin{aligned} g^1 &\rightarrow G^1(D) = g_0^1 + g_1^1 D + g_2^1 D^2 \\ g^2 &\rightarrow G^2(D) = g_0^2 + g_1^2 D + g_2^2 D^2 \end{aligned} \tag{2.11}$$

şi:

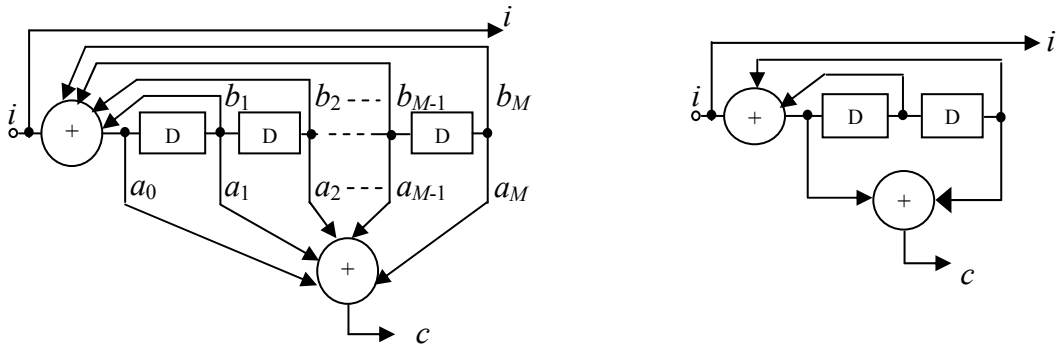
$$\begin{aligned} G^1(D) &= 1 + D^2 \\ G^2(D) &= 1 + D + D^2 \end{aligned} \tag{2.12}$$

rezultând matricea generatoare $G=[1+D^2,\ 1+D+D^2]$.

În general polinoamele generatoare ale codorului se exprimă în octal şi astfel, pentru cazul din Fig. 4.1, avem:

$$\begin{aligned} G^1 &= [1,0,1] = 5(\text{în octal}) \\ G^2 &= [1,1,1] = 7(\text{în octal}) \end{aligned} \tag{2.13}$$

În Fig.4.2 a) se prezintă schema generală a unui codor recursiv sistematic, RSC (*Recursive Systematic Code*), iar în figura Fig. 4.2 b) un caz particular al acesteia.



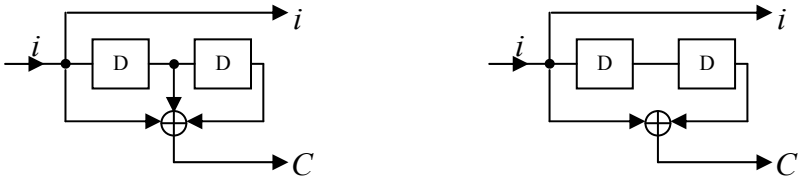
a) b)

Fig. 4.2. Cod convoluţional recursiv sistematic: a) schema generală, b) exemplu (R=1/2, K=3).

În exemplul considerat matricea generatoare este de forma:

$$G = \left[1, \frac{1+D^2}{1+D+D^2}\right] \tag{2.14}$$

În figura următoare sunt prezentate două exemple de codoare sistematice şi nerecursive, NRSC (*Non-Recursive Systematic Code*), considerându-se randamentul R=1/2 şi lungimea de constrângere K=3:



a) b)

Fig. 4.3 Coduri convoluţionale sistematic şi nerecursiv, R=1/2, K=3, G=1+D+D^2.

Diagrame de stări

Diagrama de stări este o reprezentare a funcţionării unui codor convoluţional, în care timpul nu apare în mod explicit.

În Fig.4.4 s-a reprezentat diagrama de stări asociată codorului convoluţional din Fig. 4.1. Diagrama de stări permite evaluarea funcţiei de transfer a codorului, care va fi utilizată pentru calculul performanţelor codului.

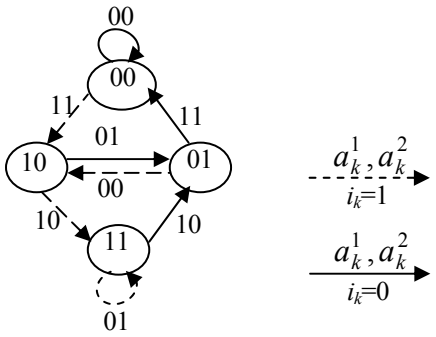


Fig. 4.4: Diagrama de stări a codorului din Fig. 4.1.

Diagramele de stări corespunzătoare codurilor convoluţionale din Fig.4.2 b) şi Fig.4.3 a) şi b) sunt prezentate în Fig.4.5.

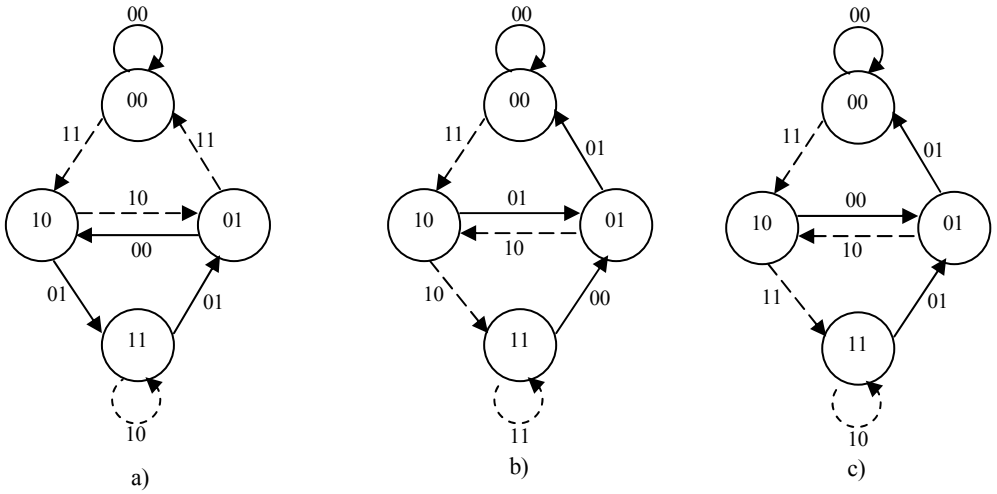


Fig. 4.5 Diagramele de stare pentru codoarele convoluţionale: a) RSC; b) şi c) NRSC.

Diagrama trellis

Fiecare bloc de $n = 2$ simboluri de la ieşirea acestui codor depinde de blocul de $k = 1$ simbol prezent la intrarea sa dar şi de $m = 2$ blocuri de k simboluri conţinute în memoria sa. Aceste $mk = 2$ simboluri definesc starea codorului. Se notează cu $S_0 = (00)$, $S_1 = (01)$, $S_2 = (10)$ şi $S_3 = (11)$, cele patru stări posibile ale codorului din Fig. 4.1. Oricare ar fi starea iniţială a codorului, după $m+1=3$ întârzieri la intrarea codorului, toate stările au fost atinse.

Funcţionarea codorului poate fi explicată ţinând seama doar de stările sale şi de tranziţiile dintre acestea, numite ramuri (ramificaţii, braţe). Diagrama *trellis* astfel obţinută este reprezentată în Fig. 4.6, pentru codorul convoluţional din Fig. 4.1, presupunând ipoteza că starea sa iniţială era $S_0 = (00)$.

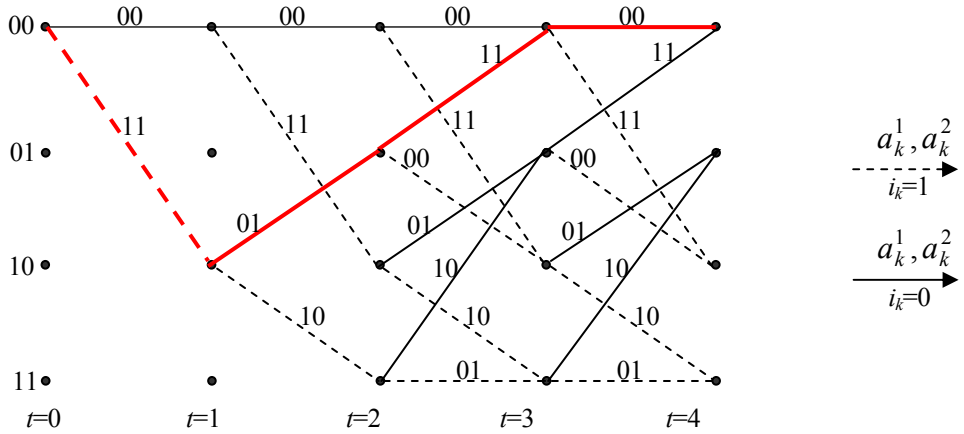


Fig. 4.6. *Trellis*-ul codorului convoluţional din Fig. 4.1.

Ramurile reprezentate prin linii punctate corespund prezenţei unui simbol de informaţie egal cu 1, la intrarea codorului, şi ramurile reprezentate prin linii pline, unui simbol de informaţie egal cu 0. Fiecărei ramuri i s-a asociat valoarea cuplului binar disponibil la ieşirea codorului.

După $m+1$ întârzieri, oricare ar fi starea iniţială a codorului, *trellis*-ul se repetă. Din fiecare nod pleacă 2^k ramuri (în cazul de faţă sunt două ramuri) şi în fiecare nod converg 2^k ramuri.

Pornind de la starea $S_0=(00)$ în momentul $t=0$, de exemplu, vedem că există patru căi care permit atingerea stării $S_0=(00)$ în momentul $t=4$.

00 00 00 00 → calea 1
00 11 01 11 → calea 2
11 10 10 11 → calea 3
11 01 11 00 → calea 4

Codarea codului convoluţional

Folosind o codare convoluţională, se codează următoarea secvenţă de informaţie: i=100111, considerând polinomul generator al codului: $g(D)=1+D^2$.
În acest scop se foloseşte Fig. 4.7.

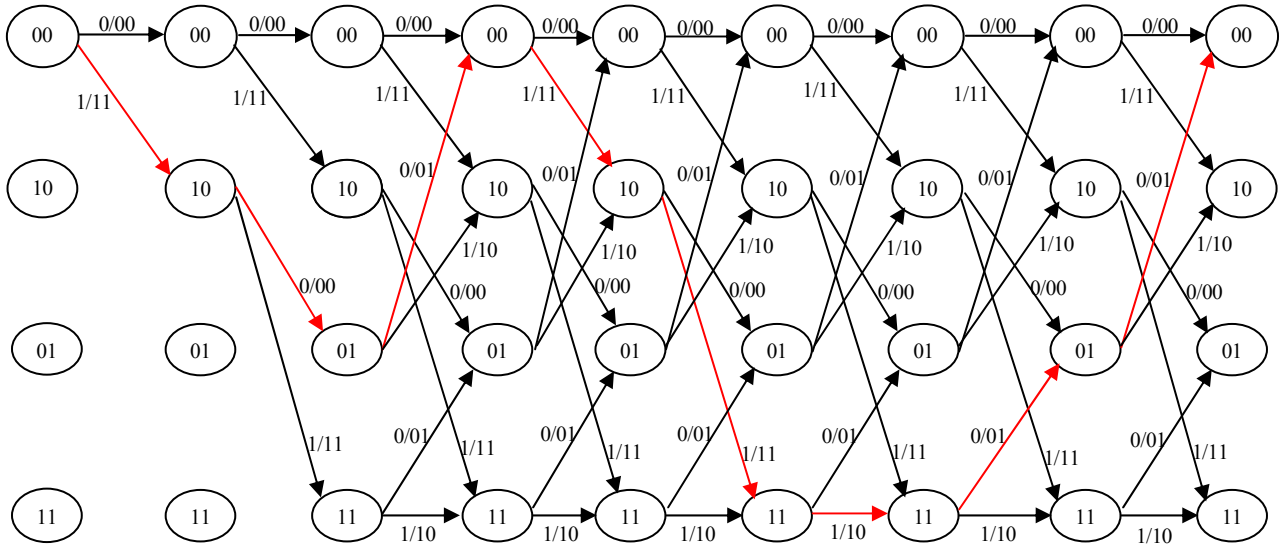


Fig. 4.7. Utilizarea diagramei trellis în cazul codării secvenţei de informaţie i=100111, considerând diagrama de stări din Fig. 4.5. c).

Cuvântul de cod obţinut este v=11 00 01 11 11 10 01 01. Trellis-ul codorului convoluţional se închide la zero, astfel, datorită acestui fapt au apărut două grupe suplimentare de biţi.

Calea pe trellis care ne generează cuvântul de cod, v, este cea marcată cu roşu.

Decodarea codului convoluţional, algoritmul Viterbi cu decizie hard

Pentru prezentarea acestui algoritm se consideră un canal binar simetric (fără memorie), intrarea decodorului fiind alcătuită dintr-o secvenţă de simboluri binare.

În fiecare moment două ramuri, aparţinând la două căi diferite, converg spre fiecare nod al *trellis*-lui (Fig.4.6). Din aceste două căi una este mai probabilă, altfel spus, se găseşte la cea mai mică distanţă Hamming faţă de secvenţa recepţionată, decât cealaltă cale. Distanţa fiind o funcţională aditivă, în fiecare nod se păstrează calea cea mai probabilă numită cale supravieţuitoare. Dacă se obţin două căi cu aceeaşi distanţă Hamming, doar o singură cale este pastrată, alegându-se în mod arbitrar una din cele două căi posibile.

În figura următoare se prezintă un exemplu de decodare pentru codorul reprezentat în Fig. 4.3. b), cu diagrama de stări reprezentată în Fig. 4.5. c). Calea rezultantă este marcată cu culoare roşie.

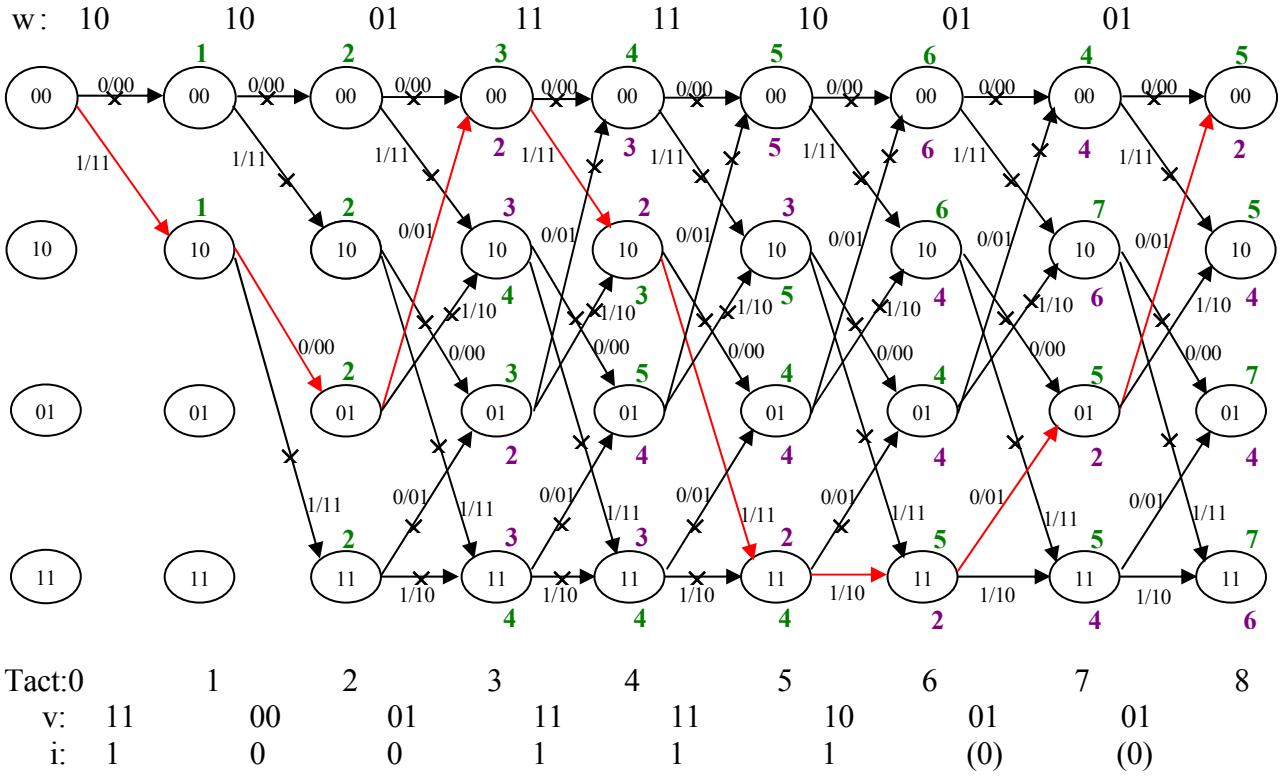


Fig. 4.8. Utilizarea diagramei trellis în cazul decodării secvenţei w=A7E5$_H$, considerând diagrama de stări din Fig. 4.5. c).

Desfăşurarea lucrării:

a) Rulaţi programul pe calculator, utilizând opţiunea demonstrativă atât pentru algoritmul de codare cât şi pentru cel de decodare.

b) Realizaţi o codare şi o decodare, folosind algoritmul de decodare Viterbi. Verificaţi apoi calculele cu ajutorul programului de pe calculator.

Bibliografie

[1] P. Elias, *"Error-free Coding"*, IRE Trans. Inform. Theory, vol IT-4, pp.29-37, 1954,

[2] Chien, R, "*Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem codes",* IEEE Transactions on Information Theory, Volume 10, Issue 4, Page(s): 357 - 363, Oct 1964,

[3] A. Glavieux, M. Joindot, *"Communications numériques. Introduction"*, Masson, Paris, 1996,

[4] M. E. Borda, *"Teoria transmiterii informaţiei"*, Editura Dacia, Cluj-Napoca, 1999,

[5] G. Wade, *"Coding Techniques-An Introduction to Compresion and Error Control"*, Creative Print and Design, Ebbw Vale, Geat Britain, 2000,

[6] J. Proakis, *"Digital Communications"*, 4th Ed., McGraw Hill, New York, 2000,

[7] H. Baltă, M. Kovaci, *"A Comparasion Between Weight Spectrum of Different Convolutional Code Types"*, conferinţă Oradea, 2004.