

## Noțiuni elementare despre turbocoduri

- acest capitol prezintă unele noțiuni elementare privind structura, codarea și decodarea turbocodurilor, unele considerente privind performanțele (BER vs. SNR) asigurate de aceste coduri.

### 1. Coduri convoluționale

#### 1.1 Tipuri și generare

- în această secțiune vor fi discutate coduri convoluționale de rată  $R_c=1/n$  și constrângere  $K$ .

- constrângerea codului este numărul de biți (informaționali doar pentru codurile nerecursive) necesari la un moment dat pentru generarea biților codați. Constrângerea codului se notează cu  $K$ , iar  $K = v+1$ , unde  $v$  numărul biților anteriori bitului curent care determină biții codați într-o perioadă de tact. Numărul celulelor de memorie necesare codorului este  $K = v + 1$  (o celulă pentru stocarea bitului/biților curenți de intrare în codor) - **vezi relația (4) pag. 1, cap. Modulații codate al cursului TD din anul IV**

- alte rate de codare pot fi obținute, pornind de la acest tip de coduri numite „coduri părinte”, prin procedeul de *puncturare* („*puncturing*”).

- codurile convoluționale sunt coduri liniare, iar biții de control se obțin în urma unor combinații liniare între biții de informație conform unor reguli de generare.

- în practică se utilizează un bloc de biți informaționali de lungime finită, iar în acest caz codurile convoluționale pot fi tratate ca și coduri bloc liniare

- ele pot fi descrise (definite) în mai multe moduri:

- prin matricea generatoare,  $G$ , care indică în mod direct regulile de generare, adică arată modul în care din  $N$  biți informaționali se obțin cei  $N/R_c$  biți codați – **vezi cursul de TD - idem**
- prin matricea de paritate,  $H$ , care indică ecuațiile care trebuiesc satisfăcute de biții codați
- prin diagrama *trellis*, - în care blocurile codate valide sunt indicate de căile posibile în diagramă **vezi Exemplul 6 pag. 5-6 capitol Modulații codate, curs TD an IV**

- matricea generatoare, care este construită pe baza a  $n$  vectori linie,  $g_1, g_2, \dots, g_n$  de dimensiune  $K=v+1$  care indică biții, informaționali sau preprocesați, pe baza cărora se obțin biții de control. Acești vectori pot fi interpretați ca și coeficienții polinoamelor generatoare ale codului

$$G = [g_1^T, g_2^T, \dots, g_n^T] \quad (1)$$

- vectorii sunt, de obicei, reprezentați în octal, de exemplu vectorul generator  $g_1=[1,1,0,1]$  poate fi scris în baza 8 ca fiind  $g_1=(15)_8$ .

- în cazul general, bitul informațional  $d_k$  este preprocesat prin sumarea lui modulo 2 cu biții stocați în celulele indicate de coeficienții nenuli  $g_{0i}$  din vectorul polinomului generator  $g_0$ , cf ecuației (2); în urma acestei preprocesări rezultă bitul  $u_k$  ce este introdus în registrul de deplasare al codorului ca bit de intrare al perioadei de simbol curente  $k$ . Figura 1 prezintă structura generală a unui codor convoluțional de rată  $1/n$ , construit cu un registrul de deplasare cu  $v = K-1$  celule de memorie.

$$u_k = \sum_{i=1}^v g_{0i} \cdot u_i \oplus d_k \quad (2)$$

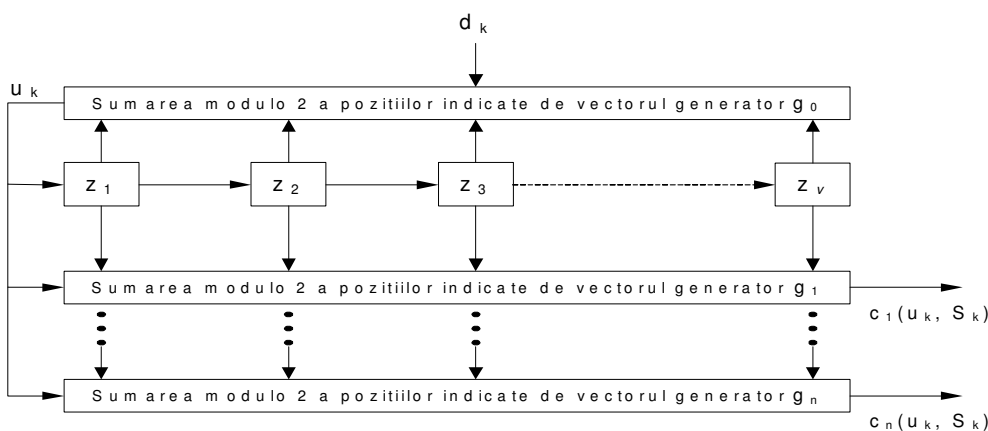


Figura 1.  
Forma generală a unui codor convoluțional cu rata  $1/n$

- această preprocesare este necesară pentru obținerea codurilor convoluționale recursive. - **vezi exemplul din pag. 2 cap. Modulații codate curs TD**

- codorul mai conține  $n$  sumatoare modulo 2; fiecare dintre ele sunează bitul  $u_k$ , obținut în urma preprocesării, cu biții din celulele registrului de deplasare, indicate de vectorii generatori,  $g_1, \dots, g_n$  obținându-se secvența de biți codați  $C(d_k, S_k) = [c_1(d_k, S_k), c_2(d_k, S_k), \dots, c_n(d_k, S_k)]$ , pentru bitul de intrare  $d_k$  și starea codorului  $S_k = [s_1 s_2 \dots s_{v-1}]$ .

$$C(d_k, S_k) = [u_k S_k] \cdot G \quad (3)$$

- numărul de stări în care poate fi codorul este  $2^v$  și este o măsură a complexității codului.

- vectorul stării viitoare  $S_{k+1}$  se obține din vectorul stării curente  $S_k$  prin deplasare la dreapta cu o poziție și introducerea noului bit pe poziția eliberată. Descrierea modului în care registrul de deplasare trece dintr-o stare în alta poate fi făcută cu ajutorul matricii de tranziție  $T$ , care are dimensiunile  $v \times v$ , construită în felul următor:

$$T = \begin{bmatrix} g_0[2] & 1 & 0 & \dots & 0 \\ g_0[3] & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_0[v] & 0 & 0 & \dots & 1 \\ g_0[v+1] & 0 & 0 & \dots & 0 \end{bmatrix} \quad (4)$$

- vectorul stării viitoare  $S_{k+1}$  se calculează cu relația (5), în care noul bit informațional apare în vectorul  $D_k = [d_k \ 0 \ 0 \ \dots \ 0]$ : În cazul codurilor recursive bitul de intrare  $d_k$  se înlocuiește cu bitul  $u_k$ , obținut prin preprocesare.

$$S_{k+1} = S_k \cdot T + D_k \quad (5)$$

- această formă de descriere a codurilor convoluționale permite generarea și tratarea unitară a trei tipuri de coduri convoluționale prin modificarea polinoamelor (vectorilor) generatoare  $g_0$  și  $g_1$ , și anume: nerecursive sistematice, nerecursive nesistematice și recursive sistematice.

- polinomul  $g_0$  indică recursivitatea sau nerecursivitatea codului ( $g_0 = 1$  pentru coduri nerecursive)

- polinomul  $g_1 = 1$  pentru coduri sistematice și  $g_1 \neq 1$  pentru coduri nesistematice

### 1.1.2. Coduri convoluționale nerecursive sistematice

- În cazul acestui tip de coduri bitul  $u_k = d_k$  bitul de informație, care este transmis în clar, deci vectorii generatori  $g_0$  și  $g_1$  vor avea expresiile date în relația (5), iar la ieșirea  $c_1$  a codorului se va obține valoarea bitului informațional,  $c_1(u_k, S_k) = u_k = d_k$ , vezi relația (2):

$$g_0 = [1, 0, \dots, 0]; \quad g_1 = [1, 0, \dots, 0]; \quad (6)$$

**Exemplu: vezi fig. 7 din referința de mai sus în care se taie legatura  $B_{k-2}$  la XOR- $c_0$**

### 1.1.3 Coduri convoluționale nerecursive nesistematice

- În cazul acestui tip de cod fiecare din cei  $n$  biți codați sunt obținuți prin câte o combinație liniară a biților din registrul de deplasare și a bitului informațional curent, vezi cursul de TD. În acest caz vectorul generator  $g_0$  are forma:

$$g_0 = [1, 0, \dots, 0] \quad \text{Ex: fig. 6 din referința de mai sus} \quad (7)$$

- Astfel ales vectorul  $g_0$ , valoarea bitului  $u_k$  de la intrarea registrului de deplasare este aceeași cu cea a bitului informațional,  $d_k$ , iar biții codați  $C(d_k, S_k)$  se obțin pe baza relației (3).

- în literatura de specialitate se arată că performanțele codurilor convoluționale nesistematice sunt mai bune decât cele ale codurilor sistematice în domeniul de valori mari ale SNR; în domeniul valorilor mici ale SNR codurile sistematice asigură valori mai mici ale BER.

- un dezavantaj major al codurilor nesistematice este faptul că pentru anumite combinații de polinoame generatoare, aceste coduri pot fi coduri catastrofice

### 1.1.4. Coduri convoluționale recursive sistematice

- această clasă de coduri asigură performanțe mai bune decât codurile nesistematice

- există o metodă de echivalare a acestui tip de coduri cu cele nerecursive nesistematice.

- acest tip de coduri presupune reintroducerea biților existenți în registrul de deplasare (memoria codorului) la intrarea acestuia, iar bitul  $d_k$  informațional curent este transmis ca atare, „în clar”.

- pentru obținerea acestor coduri, trebuie ca vectorii generatori  $g_1$  și  $g_0$  să fie identici:

$$g_0(i) = g_1(i) = g(i), \quad \forall i \in \{1, 2, \dots, v+1\} \quad (8)$$

- astfel informația obținută la intrarea registrului de deplasare se obține din sumarea modulo 2 a bitului informațional cu pozițiile din registrul de deplasare indicate de vectorul generator  $g_0$ .
- dacă vectorul generator  $g_1$  satisface condiția (8) atunci la ieșirea  $c_1$  a codorului se obține:

$$c_1(d_k, S_k) = u_k + \sum_{i=1}^v S_k(i) \cdot g_1(i+1) = d_k + \sum_{i=1}^v S_k(i) \cdot g_0(i+1) + \sum_{i=1}^v S_k(i) \cdot g_1(i+1) \quad (9)$$

$$= d_k + \sum_{i=1}^v S_k(i) \cdot g(i+1) + \sum_{i=1}^v S_k(i) \cdot g(i+1) = d_k \quad \text{daca } g_0 = g_1 = g$$

cee ce arată că satisfacerea condiției (8) conduce la un cod sistematic, chiar dacă bitul de intrare este obținut în mod recursiv din bitul informațional curent  $d_k$  și biți a registrului de deplasare.

#### Aducerea codorului în starea zero

- se efectuează la sfârșitul codării unui bloc de date, pentru ca operațiile de codare și decodare să înceapă din aceeași stare inițială a registrului și diagramei trellis
- dacă la codurile nerecursive aducerea codorului în starea zero se putea face simplu prin adăugarea la sfârșitul secvenței informaționale a unei terminații de  $K-1 = v$  biți de „0”, în cazul codurilor recursive aducerea codorului în starea 0 se face în mod diferit, mai exact prin forțarea în „0” a valorii la ieșirea sumatorului corespunzător vectorului generator  $g_0$ , adică  $u_k=0$ .
- pentru aceasta sumatorul se construiește ca în figura 2. Cât timp sunt date la intrare, comutatorul *COM* va fi pe poziția 1; dacă se dorește aducerea codorului în starea 0, comutatorul *COM* va trece în poziția 2, forțând valoarea ieșirii  $u_k$  în 0.

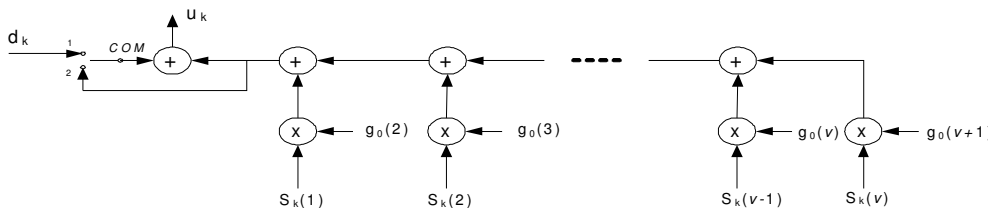


Figura 2. Sumatorul corespunzător vectorului generator  $g_0$

- terminația introdusă la sfârșitul fiecărui bloc de date, pentru a aduce codorul în starea zero și a face ca decodarea unui bloc de date să fie independentă de decodarea blocului anterior, duce la scăderea eficienței spectrale a transmisiei, scădere care poate deveni semnificativă în cazul blocurilor codate având o lungime redusă (număr redus de biți).
- inserarea terminației face ca la decodare să se cunoască întotdeauna stările inițială și finală ale decodorului, care vor fi identice și egale cu zero.
- de asemenea inserarea terminației asigură existența unor informații suplimentare pentru decodarea ultimilor  $K-1 = v$  biți informaționali, la recepție
- o altă abordare care nu necesită inserarea unor biți suplimentari ca terminație, o constituie utilizarea codurilor convoluționale circulare, descrise în literatură.
- în principiu această metodă urmărește determinarea stării codorului  $S_c$ , pentru care codorul plecând din această stare ca stare inițială ( $S_0 = S_c$ ) va ajunge după  $N$  tați tot în această stare, ca stare finală, adică  $S_N = S_c$ ; cu  $N$  s-a notat lungimea blocului de date,
- determinarea stării circulare  $S_c$  la codor se face printr-o codare anterioară a blocului de date inițializând codorul cu starea 0, ceea ce reprezintă un dezavantaj al acestei abordări deoarece pentru codarea unui bloc de informație procesul de codare trebuie aplicat prima dată pentru determinarea stării circulare, iar a doua oară pentru codarea propriu-zisă.
- starea  $S_c$  trebuie transmisă și decodorului de la recepție

### 1.2 Puncturarea codurilor convoluționale

- puncturarea este procesul de eliminare a unor biți codați de la ieșirea unui codor în vederea obținerii unei **rate de codare mai mari**, pornind de la un cod convoluțional „părinte” de rată  $R_m = 1/n$ .
- pentru a obține o rată a codului  $R_c = p/q > 1/n$  trebuie eliminați  $n \cdot p - q$  biți la fiecare  $n \cdot p$  biți codați cu codul „părinte”. - **vezi cazurile particulare prezentate în Ex. 4 din referința menționată**
- biții care urmează să fie eliminați pentru a obține o rată de codare dată sunt indicați de masca de puncturare, vezi Ex.4, sau folosind algoritmi de adaptare de rată („rate-matching”), cum ar fi cel specificat în standardul 3GPP.
- măștile de puncturare sunt structuri bidimensionale, de dimensiune  $n \times p$  care conțin elemente de 1

și 0, unde valoarea 1 indică faptul că bitul respectiv este transmis, iar 0 indică faptul că bitul nu este transmis. Masca de puncturare se aplică la ieșirea codorului pentru fiecare n-p biți codați.

- decodarea se face folosind decodorul codului părinte, prin inserarea elementului neutru al operației de calcul a metricii folosite de decodor în pozițiile biților codați care au fost eliminați la emisie.

- performanțele codurilor convoluționale de rată  $R_c$  obținute prin puncturare, sunt similare cu cele ale codurilor convoluționale nepuncturate de rată  $R_c$ , pentru o aceeași constrângere  $K$ . Acest lucru a dus la o largă răspândire a acestui tip de coduri, datorită faptului că adaptarea de rată se face foarte ușor, iar decodarea se face folosind doar decodorul codului părinte pentru o gamă largă de rate obținute prin puncturare. **Cum se modifică ratele codurilor obținute prin puncturare sau prescurtare?**

### 1.3. Decodarea codurilor convoluționale utilizând algoritmul MAP

**Notă: înainte de parcurgerea acestui subcapitol de recomandă studierea anexei de la pag. 8**

- algoritmul MAP utilizat pentru decodarea codurilor convoluționale recursive este o variantă modificată, de către C. Berrou et al., a unui algoritm clasic denumit BCRJ.

- spre deosebire de algoritmul Viterbi, vezi capitolul TCM din cursul TD, care determină cea mai plauzibilă secvență de biți transmisă folosind valoarea acumulată a metricii utilizate, (fiind de tip Maximum Likelihood - ML), algoritmul MAP (Maximum A Posteriori) determină cea mai plauzibilă valoare logică pentru fiecare bit recepționat. O descriere detaliată a algoritmului MAP folosit pentru decodarea codurilor convoluționale este prezentată în Pietrobon, S. S., "Implementation and Performance of a Turbo/MAP Decoder," Int'l. J. Satellite Communications, vol. 15, Jan-Feb 1998, pp.23-46.

- se consideră un codor convoluțional recursiv cu constrângerea  $K = v+1$ .

- se notează cu  $S_k = \{s_1, s_2, \dots, s_v\}$  starea registrului de deplasare al codorului în perioada de simbol  $k$ , stare ce depinde doar de valoarea celulelor de memorie ale codorului. Bitul informațional de la intrarea codorului în momentul  $k$  se notează cu  $d_k$  și corespunde tranziției dintre stările  $S_k$  și  $S_{k+1}$ . Pentru fiecare bloc de de biți, starea inițială și starea finală ale codorului sunt forțate la zero.

- folosind un cod RSC (Recursive Systematic Convolutional) de rată 0.5 se codează o secvență de date de  $N$  biți, din care  $N-v$  biți de cod corespunzători biților informaționali, iar ultimii  $v$  biți sunt astfel aleși astfel încât să forțeze starea codorului în zero.

- la ieșirea codorului, pentru fiecare bit informațional  $d_k$  se obține câte un bit de control  $c_k$ .

- se consideră că secvența codată obținută este transmisă în formă bipolară pe un canal de tip AWGN; astfel secvența recepționată este  $R_1^N = \{R_1, \dots, R_N\}$ , unde fiecare element este de forma  $R_k = \{x_k, y_k\}$  descrisă de (10), iar  $n_{d,k}$  și  $n_{c,k}$  reprezintă valorile zgomotului care afectează ce doi biți, fiind două variabile aleatoare cu distribuție normală de deviație standard  $\sigma^2/2$  și medie nulă.

$$x_k = (2 \cdot d_k - 1) + n_{d,k}; \quad y_k = (2 \cdot c_k - 1) + n_{c,k}; \quad (10)$$

- raportul de plauzibilitate logaritmice,  $LLR_k$  (11), calculat la recepție pentru bitul de date emis la momentul  $k$ , se scrie ca fiind logaritmul natural al raportului dintre probabilitatea *aposteriori* ca bitul emis să fie „1” și probabilitatea *aposteriori* ca bitul emis să fie „0”; ambele trebuie calculate în ipotezele că la emisie starea codorului era  $m$ , iar secvența recepționată este  $R_1^N$ , vezi (10)

$$LLR_k = \ln \left( \frac{p(d_k = 1 | R_1^N)}{p(d_k = 0 | R_1^N)} \right) = \ln \left( \frac{\sum_m p(d_k = 1, S_k = m | R_1^N)}{\sum_m p(d_k = 0, S_k = m | R_1^N)} \right) = \ln \left( \frac{\sum_m \lambda_k(1, m)}{\sum_m \lambda_k(0, m)} \right) \quad (11)$$

unde  $m \in \{0, 1, \dots, 2^v - 1\}$  reprezintă stările pe care le poate lua codorul.

- probabilitatea să se fi transmis bitul  $d_k = i$ , ( $i = „0”$  sau „1”), iar starea codorului să fie  $m$ , dacă s-a recepționat secvența  $R_1^N$ , se poate scrie ca un raport de două probabilități, care în urma aplicării succesive a teoremei lui Bayes privitoare la probabilități condiționate, vezi (30) din anexă care este reluată aici, are expresia (12), în care  $p(R_1^N)$  este probabilitatea de recepționare a secvenței  $R_1^N$  pentru orice  $d_k$  și orice stare  $m$ .

$$p(A/B) = p(B/A) \cdot p(A) / p(B) \quad (30)$$

$$\begin{aligned} \lambda_k(i, m) &= p(d_k = i, S_k = m | R_{k-1}^N, R_k^N, R_{k+1}^N) \\ &= \frac{p(R_{k-1}^N | S_k = m, d_k = i, R_k^N) \cdot p(R_{k+1}^N | S_k = m, d_k = i, R_k^N) \cdot p(S_k = m, d_k = i, R_k^N)}{p(R_k^N)} \end{aligned} \quad (12)$$

- primul factor al numărătorului ecuației (12), este probabilitatea de a se recepționa secvența  $R_{k-1}^N$ , condiționată de  $S_k = m$ ,  $d_k = i$  și  $R_k^N$ , denumită *metrică de stare înainte*,  $\alpha(m)$ ; acesta se poate simplifica datorită faptului că secvența recepționată până la momentul  $k$  nu este condiționată decât de starea curentă a codorului,  $S_k$ .

$$p\left(R_1^{k-1} \mid S_k = m, \overbrace{d_k = i, R_k^N}^{\text{irelevant}}\right) = p\left(R_1^{k-1} \mid S_k = m\right) \stackrel{\Delta}{=} \alpha(m) \quad (13)$$

- similar, cel de-al doilea factor al numărătorului ecuației (12), denumit *metrică de stare înapoi*  $\beta_{k+1}(f(i,m))$ , poate fi scris sub forma (14), în care  $f(i,m)$  reprezintă starea viitoare a decodorului dacă starea curentă este  $m$  iar bitul informațional ia valoarea  $i$  („0” sau „1”):

$$p\left(R_{k+1}^N \mid S_k = m, d_k = i, \overbrace{R_k^N}^{\text{irelevant}}\right) = p\left(R_{k+1}^N \mid S_{k+1} = f(i,m)\right) \stackrel{\Delta}{=} \beta_{k+1}(f(i,m)) \quad (14)$$

- ultimul factor, numit *metrică de tranziție*, se notează astfel:

$$p\left(S_k = m, d_k = i, R_k^N\right) \stackrel{\Delta}{=} \gamma_k(i,m) \quad (15)$$

- înlocuind ecuațiile (13), (14) și (15) în ecuația (12) se obține:

$$\lambda_k(i,m) = \frac{\alpha_k(m) \cdot \gamma_k(i,m) \cdot \beta_{k+1}(i,m)}{p(R_1^N)} \quad (16)$$

- folosind (16), relația de calcul a raportului de plauzibilitate logaritmice (11) devine:

$$LLR_k = \ln \left( \frac{\sum_m \alpha_k(m) \cdot \gamma_k(1,m) \cdot \beta_{k+1}(1,m)}{\sum_m \alpha_k(m) \cdot \gamma_k(0,m) \cdot \beta_{k+1}(0,m)} \right) \quad (17)$$

### 1.3.1 Calculul metricilor de stare și de tranziție

- în literatură este descris modul în care metricile de stare pot fi calculate recursiv.

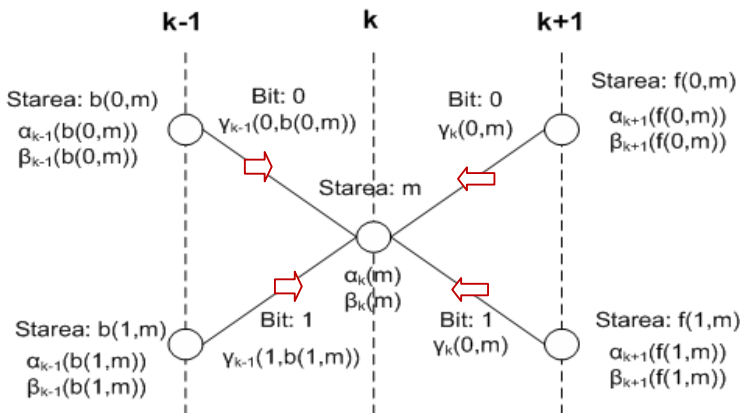
- pornind de la relația (13) și folosind o abordare recursivă, valoarea *metricii înainte* pentru starea  $m$  are expresia (18), unde  $b(i,m)$  reprezintă starea din care se ajunge în starea  $m$  cu bitul de intrare  $i$ ;  $b(i,m)$  se poate determina cu matricea  $T$ , **vezi relațiile (4) și (5)**

$$\alpha_k(m) = p\left(R_1^{k-1} \mid S_k = m\right) = \sum_i \alpha_{k-1}(b(i,m)) \cdot \gamma_{k-1}(i,b(i,m)) \quad (18)$$

- similar valoarea *metricii înapoi* se poate determina cu (19):  $f(i,m)$  se poate determina cu  $T$ , **(4), (5)**

$$\beta_k(m) = p\left(R_k^N \mid S_k = m\right) = \sum_i \beta_{k+1}(f(i,m)) \cdot \gamma_k(i,m) \quad (19)$$

- **metrica de stare înainte se poate calcula pe măsură ce sosesc fazorii  $R_k$**



- **metrica de stare înapoi trebuie calculată abia după recepționarea tuturor fazorilor corespunzători unui bloc codat;** această operație introduce întârziere („latency”)

- figura 3 ilustrează grafic modul de calcul al metricilor de stare și a celor de tranziție.

Figura 3. Reprezentarea grafică a calculului metricilor de stare

- algoritmul MAP va determina în primă fază:

- valorile metricilor de *stare înainte* pentru fiecare stare posibilă a codorului în fiecare perioadă de simbol pentru cele două valori ale bitului, și va stoca aceste valori;
- valorile metricilor de tranziție și va stoca aceste valori.

- după ce a fost recepționat tot blocul codat, algoritmul va determina metricile de stare înapoi, pornind de la celălalt capăt al secvenței de fazori recepționați.
- datorită faptului că stările inițială și finală ale codorului sunt zero metricile de stare se inițializează în felul următor:

$$\alpha_1(m) = \begin{cases} 1, m=0 \\ 0, \text{în rest} \end{cases} \quad \beta_{N+1}(m) = \begin{cases} 1, m=0 \\ 0, \text{în rest} \end{cases} \quad (20)$$

- metricile de tranziție pot fi calculate folosind de două ori succesiv teorema lui Bayes privitoare la probabilități compuse (vezi (31) din anexa de la pag.8, reluată mai jos), considerând că valorile logice ale bitului generat de sursă sunt echiprobabile,  $p(0) = p(1) = 0.5$ , și că probabilitatea codorului de a ajunge în starea  $m$  dacă bitul de intrare  $d_k = i$  este  $1/2^v$ , în felul următor:

$$p(A;B) = p(B/A)p(B) = p(A/B)p(A) \quad (31)$$

$$\begin{aligned} \gamma_k(i, m) &= p(S_k = m, d_k = i, R_k) = p(R_k | S_k = m, d_k = i) \cdot p(S_k = m | d_k = i) \cdot p(d_k = i) \\ &= p(R_k | S_k = m, d_k = i) \cdot \frac{1}{2^v} \cdot \frac{1}{2} = p(x_k | S_k = m, d_k = i) \cdot p(y_k | S_k = m, d_k = i) \cdot \frac{1}{2^{v+1}}; \quad R_k = (x_k, y_k) \end{aligned} \quad (21)$$

- în ipoteza că  $T_c \gg T_s$ ;  $B_c \gg B_s$  (vezi Parametrii canalelor radio) avem un canal cu fading plat și lent variabil, adică unul canalul de tip AWGN cu zgomot de medie nulă și deviație standard  $\sigma^2$ .
- metrica de tranziție se poate exprima, pornind de la (21), în felul următor:

$$\begin{aligned} \gamma_k(i_k, m) &= \frac{1}{2^{v+1} \cdot \sigma \cdot \sqrt{2 \cdot \pi}} \cdot e^{-\left(\frac{x_k - (2 \cdot i_k - 1)}{\sigma^2}\right)^2} \cdot \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot e^{-\left(\frac{y_k - (2 \cdot c(i_k, m) - 1)}{\sigma^2}\right)^2} \\ &= \frac{1}{2^{v+1} \cdot \sigma^2 \cdot 2 \cdot \pi} \cdot e^{-\frac{2}{\sigma^2} \cdot (x_k \cdot i' + y_k \cdot c'(i, m))} \end{aligned} \quad (22)$$

unde  $c(i_k, m)$  reprezintă bitul de control obținut pentru bitul de informație  $i_k$  dacă codorul se găsea în starea  $m$ , iar cu  $i'$  și  $c'(i_k, m)$  s-au notat formele bipolare ale bitului de informație  $i_k$  și respectiv ale bitului de control  $c(i_k, m)$ .  $x_k, y_k$  sunt nivelele recepționate.

- Constantele  $e^{-\frac{(x_k^2 + y_k^2)}{2\sigma^2}}$  și  $e^{-\frac{(i_k^2 + c_k^2)}{2\sigma^2}}$ , obținute prin dezvoltarea pătratelor, se neglijează pentru că nu afectează semnul LLR-ului care e esențial la decizia valorii bitului respectiv
- Constanta  $\frac{1}{2^{v+1} \cdot \sigma^2 \cdot 2 \cdot \pi}$  poate fi ignorată din același motiv
- Expresia LLR se obține înlocuind relația (22) în ecuația (17) și rezultă:

$$LLR_k = + \frac{2}{\sigma^2} \cdot x_k \cdot i' + \ln \underbrace{\left( \frac{\sum_m \alpha_k(m) \cdot e^{-\frac{2}{\sigma^2} \cdot y_k \cdot c'(1, m)} \cdot \beta_{k+1}(1, m)}{\sum_m \alpha_k(m) \cdot e^{-\frac{2}{\sigma^2} \cdot y_k \cdot c(0, m)} \cdot \beta_{k+1}(0, m)} \right)}_{L_e(x_k)} \quad (23)$$

- Primul termen al relației (23) reprezintă *informația intrinsecă*, adică informația care se obține despre acest bit din fazorul recepționat (a cărui valoare este efectată de canalul de transmisie)
- Al doilea termen al relației (23) reprezintă *informația extrinsecă*,  $L_e(x_k)$ , adică informația suplimentară care se poate obține, din semnalele corespunzătoare celorlați biți ai blocului codat și din intercondiționările dintre aceștia (impuse de modul de construcție a codului), despre un anumit bit informațional, în acest caz bitul  $k$ .
- $L_e(x_k)$  este informație "soft" deoarece poate lua un număr mult mai mare de valori (teoretic ea e o variabilă continuă și nu discretă) decât numărul de valori pe care îl poate lua bitul (care e o informație "hard").
- Această informație este importantă în cazul codurilor concatenate deoarece permite transferul de informație soft între decodare, adică a efectului decodării cu fiecare cod asupra LLR-ului bitului



respectiv sau, în cazul decodoarelor iterative, unde informația „soft” este utilizată pentru actualizarea LLR-ului bitului respectiv la fiecare iterație.

- Pe baza raportului de plauzibilitate logaritmice (LLR) furnizat de decodor se face decizia hard a bitului estimat  $\tilde{d}_k$  pe baza următoarei forme a criteriului de decizie al lui Bayes:

$$\tilde{d}_k = \begin{cases} 0; & \text{daca } LLR_k < 0 \\ 1; & \text{daca } LLR_k > 0 \end{cases} \quad (24)$$

### 1.3.2 Algoritmul Max-Log-MAP

- După cum reiese din descrierea de mai sus, algoritmul MAP este foarte complex din punct de vedere computațional datorită numărului mare al operațiilor de înmulțire ce trebuie efectuate.

- Pentru eliminarea acestui neajuns s-a elaborat o variantă care operează cu logaritmiile metricilor de stare și tranziție; prin efectuarea operațiilor în domeniul logaritmice, înmulțirile se “transformă” în adunări.

- Astfel, prin aplicarea logaritmului natural relațiilor (18) și (19), relațiile de calcul ale metricilor înainte și înapoi devin (25) și, respectiv (26), în care  $\tilde{\alpha}, \tilde{\beta}, \tilde{\gamma}$  reprezintă varianta logaritmată a metricilor de stare și tranziție. Se folosește identitatea  $U = \exp(\ln U)$

$$\begin{aligned} \tilde{\alpha}_k(m) &\triangleq \ln(\alpha_k(m)) = \ln\left(\sum_i \alpha_{k-1}(b(i,m)) \cdot \gamma_{k-1}(i,b(i,m))\right) \\ &= \ln \sum_i e^{\ln(\alpha_{k-1}(b(i,m))) + \ln(\gamma_{k-1}(i,b(i,m)))} = \ln \sum_i e^{\tilde{\alpha}_{k-1}(b(i,m)) + \tilde{\gamma}_{k-1}(i,b(i,m))} \end{aligned} \quad (25)$$

$$\begin{aligned} \tilde{\beta}_k(m) &\triangleq \ln(\beta_k(m)) = \ln \sum_i \beta_{k+1}(f(i,m)) \cdot \gamma_k(m) \\ &= \ln \sum_i e^{\ln(\beta_{k+1}(f(i,m))) + \ln(\gamma_k(m))} = \ln \sum_i e^{\tilde{\beta}_{k+1}(f(i,m)) + \tilde{\gamma}_k(m)} \end{aligned} \quad (26)$$

- Calculul acestor metrici se poate simplifica și mai mult folosind aproximarea logaritmului Jacobian, vezi (32) din anexa de pe pag. 8:

$$\ln(e^a + e^b) = \max(a, b) + \ln(1 + e^{-|a-b|}) \approx \max(a, b) \quad (32)$$

$$\tilde{\alpha}_k(m) \approx \max(\tilde{\alpha}_{k-1}(b(0,m)) + \tilde{\gamma}_{k-1}(0,b(0,m)), \tilde{\alpha}_{k-1}(b(1,m)) + \tilde{\gamma}_{k-1}(1,b(1,m))) \quad (27)$$

$$\tilde{\beta}_k(m) \approx \max(\tilde{\beta}_{k+1}(f(0,m)) + \tilde{\gamma}_k(m), \tilde{\beta}_{k+1}(f(1,m)) + \tilde{\gamma}_k(m))$$

- Logaritmand relația (22) metrica de tranziție se calculează în felul următor:

$$\tilde{\gamma}_k(i,m) \triangleq \ln(\gamma_k(i,m)) = \frac{2}{\sigma} \cdot (x_k \cdot i' + y_k \cdot c'(i,m)) \quad (28)$$

- Datorită logaritmării metricilor de stare, valorile cu care acestea trebuie inițializate date în (20) vor fi înlocuite cu :

$$\tilde{\alpha}_1(m) = \begin{cases} 0, & m = 0 \\ -\infty, & \text{în rest} \end{cases} \quad \tilde{\beta}_{N+1}(m) = \begin{cases} 0, & m = 0 \\ -\infty, & \text{în rest} \end{cases} \quad (29)$$

- Prin utilizarea aproximării logaritmului Jacobian valorile metricilor de stare obținute vor fi afectate de eroarea de aproximare, care se propagă în tot blocul codat datorită modului recursiv de calcul. Acest fapt conduce la o degradare a performanțelor decodurului (scăderea câștigului codării, adică a capacității de corecție) care însă nu este semnificativă pentru valori medii și mari ale SNR, la care  $|a-b| \gg 0$ .

-deși performanțele decodurului se înrăutățesc puțin datorită aproximării utilizate, reducerea complexității implementării este substanțială.

## Anexă

### Noțiuni și relații uzuale

1. Teorema lui Bayes privitoare la probabilități condiționate:

$$p(A/B) = \frac{p(B/A) \cdot p(A)}{p(B)} \quad (30)$$

2. Teorema lui Bayes privitoare la probabilități compuse:

$$p(A;B) = p(A/B) \cdot p(B) = p(B/A) \cdot p(A) \quad (31)$$

3. Aproximarea logaritmului Jacobian:

$$\ln(e^a + e^b) = \max(a, b) + \ln(1 + e^{-|a-b|}) \approx \max(a, b) \quad (32)$$

4. Abrevieri

ML – maximum likelihood (plauzibilitate maximă)

MAP- maximum a posteriori probability

P(.) - probabilitatea unui eveniment (.)

p(.) - probability density function (pdf) a variabilei

$P_{XY/R}(x, y/r)$  – probabilitatea a posteriori (APP) - (x,y) coordonate recepționate, r fazor emis

$P_{R/XY}(r/x, y)$  – funcția de plauzibilitate (likelihood function) - r fazor emis, (x,y) coordonate rec.

5. Relația dintre probabilitatea a posteriori și funcția de plauzibilitate

- Fie  $x \in X$ ,  $y \in Y$ , variabile aleatoare discrete (de exemplu coordonatele fazorului) și  $r \in R$  o variabilă continuă (vectorul observațiilor, de exemplu fazorul recepționat).

- Variabilele X și Y se presupun independente statistic.

- Avem identitatea (33) unde  $P_Z(z)$  e probabilitate valorii  $z (= x, y, r)$  in mulțimea  $Z (=X, Y, R)$ :

$$P_{XY/R}(x, y/r) \cdot p_R(r) = p_{R/XY}(r/x, y) \cdot P_X(x) \cdot P_Y(y) \quad (33)$$

6. Decodarea unei secvențe pe baza plauzibilității maxime (MLD)

– Determină cea mai plauzibilă cale (secvență de simboluri) dintre secvențele posibile permise de condiționările codorului (de ex. descrise de diagrama trellis).

- Algoritmul lui Viterbi (VA) este un algoritm eficient pentru implementarea MLD.

- Chiar dacă folosește variabile “soft”(set de probabilități sau distanțe) ca informații de intrare, VA furnizează la ieșire valori discrete (biți), “hard outputs”.

- Există și o variantă a VA care furnizează la ieșire mărimi continue (valorile raportului de plauzibilitate – LR sau LLR). Această variantă este denumită Soft-Output-VA (SOVA)

7. Decodarea simbolului pe baza APP maximă (MAP)

- Determină cel mai plauzibil simbol, ținând cont de condiționările impuse de diagrama trellis sau de ecuațiile de control

- Algoritmul BCJR (Bahl, Cocke, Jelinek, Raviv-1974) este utilizat pentru determinarea eficientă a APP-ului fiecărui simbol (bit).

- În final se selectează simbolul (bitul) cel mai probabil, și astfel și algoritmul BCRJ furnizează la ieșire valori discrete (biți, simboluri) – “hard outputs”; și acest algoritm are o variantă care furnizează „soft output”, adică probabilități sau LLR-uri.

8. Definierea estimatelor condiționate (joint) și a estimatelor simbolurilor

- Estimata condiționată (joint) ML:  $(\hat{x}, \hat{y}) = \arg \max_{x \in X, y \in Y} p(r/x, y)$  (34)

- Estimata condiționată (joint) MAP estimate, **vezi și (30)**:

$$(\hat{x}, \hat{y}) = \arg \max_{x \in X, y \in Y} P(x, y/r) = \arg \max_{x \in X, y \in Y} p(r/x, y) \cdot P(x) \cdot P(y) / p_R(r) \quad (35)$$

- Estimata ML a unui simbol, v. (33):  $(\hat{y}) = \arg \max_{y \in Y} p(r/y) = \arg \max_{y \in Y} \sum_{x \in X} p(r/x, y) \cdot P(x)$  (36)

- Estimata MAP a unui simbol (această abordare se folosește pentru operația de „soft-demapping”), **vezi și (30)**:

$$(\hat{y}) = \arg \max_{y \in Y} P(y/r) = \arg \max_{y \in Y} \sum_{x \in X} p(r/x, y) \cdot P(x) \cdot P(y) \quad (37)$$

- Logaritmul raportului de plauzibilitate al unui bit (log-likelihood ratio – LLR) permite aplicarea criteriului lui Bayes prin comparația cu 0 (decizie pe bază de semn)

- Comparația  $LLR \geq 0$  este echivalentă cu comparația  $\tanh(LLR/2) \geq 0$  (vezi relația (38))

$$LLR = \ln \frac{P_0}{P_1} \Rightarrow e^{LLR} = \frac{P_0}{P_1} \Rightarrow \tanh\left(\frac{LLR}{2}\right) = \frac{e^{LLR} - 1}{e^{LLR} + 1} = P_0 - P_1 \quad (38)$$

- Relația care aproximează  $\tanh(x/2)$  este mai ușor de implementat decât cea care aproximează  $\ln(y)$