Noţiuni elementare despre turbocoduri

- acest capitol prezintă unele noţiuni elementare privind structura, codarea şi decodarea turbocodurilor, precum şi unele considerente privind performanţele (BER vs. SNR) asigurate de aceste coduri şi parametrii codurilor care afectează performanţele.

1. Coduri convoluţionale

1.1 Tipuri şi generare

- în această secţiune vor fi discutate coduri convoluţionale de rată $R_c=1/n$ şi constrângere K.

- constrângerea codului este numărul de biţi (informaţionali doar pentru codurile nerecursive) necesari la un moment dat pentru generarea biţilor codaţi. Constrângerea codului se notează cu K, iar K = v+1, unde υ numărul biţilor anteriori bitului curent care determină biţii codaţi într-o perioadă de tact. Nnumărul celulelor de memorie necesare codorului este K = v +1 (o celulă pentru stocarea bitului/biților curenți de intrare în codor) - vezi pag. 1 a cursului TCM din anul IV TD

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

- codurile convoluţionale sunt coduri liniare, iar biţii de control se obţin în urma unor combinaţii liniare între biţii de informaţie conform unor reguli de generare.

- în practică se utilizează o secvenţă de biţi informaţionali de lungime finită, iar în acest caz codurile convoluţionale pot fi tratate ca şi 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 biţi informaţionali se obţin cei N/R_c biţi codaţi – vezi cursul de TD - pag.1 curs TCM
- prin matricea de paritate, H, care indică ecuaţiile care trebuiesc satisfăcute de biţii codaţi
- prin diagrama *trellis,*- în care blocurile codate valide sunt indicate de căile posibile în diagramă pag. 5-6 curs TCM

- matricea generatoare, care este construită pe baza a n vectori linie, g_1, g_2, ..., g_n de dimensiune K= $v+1$ care indică biţii, informaţionali sau preprocesați, pe baza cărora se obţin biţii de control. Aceşti vectori pot fi interpretaţi ca şi coeficienţii polinoamelor generatoare ale codului

$$G = \left[g_1^T, g_2^T, ..., g_n^T \right] \tag{1}$$

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

- figura 1 prezintă structura generală a unui codor convoluţional de rată 1/n, construit cu un registru de deplasare cu v celule de memorie.

- în cazul general, bitul informaţional d_k este preprocesat prin sumarea lui modulo 2 cu biţii stocaţi în celulele indicate de vectorul polinomului generator g_0, cf ecuației (2); în urma acestei preprocesări rezultă bitul u_k ce este introdus în registrul de deplasare ca bit al perioadei de simbol curente k.

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

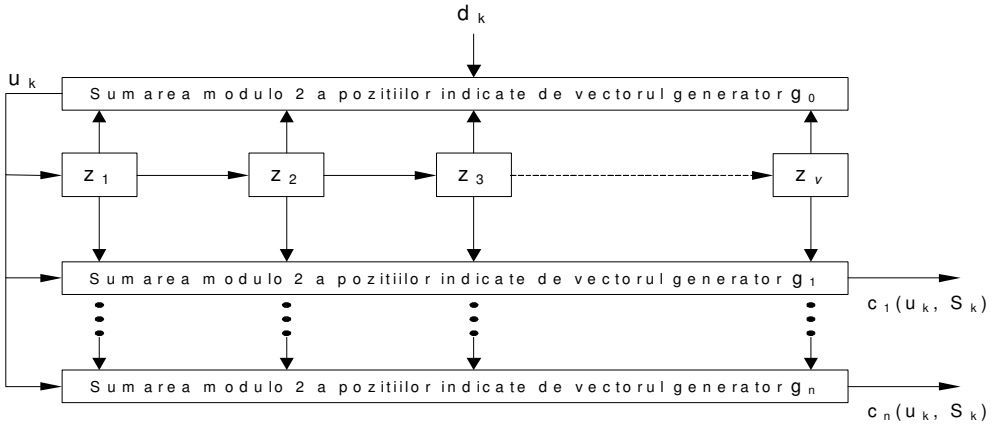


Figura 1. Forma generală a unui codor convoluţional cu rata 1/n

- această preprocesare este necesară pentru obţinerea codurilor convoluţionale recursive. -pag. 2 curs TCM

- codorul mai conţine n sumatoare modulo 2; fiecare dintre ele sumează bitul u_k, obţinut în urma preprocesării, cu biţii din celulele registrului de depasare, indicate de vectorii generatori, g_1, ..., g_n obţinându-se secvenţa de biţi codaţi $C(d_k,S_k)=[c_1(d_k,S_k), c_2(d_k,S_k), ..., c_n(d_k,S_k)]$, pentru bitul de intrare d_k şi starea codorului $S_k=[s_1\ s_2\ ...\ s_{v-1}]$.

$$C(d_k, S_k) = [u_k\ \ S_k] \cdot G \tag{3}$$

- numărul de stări în care poate fi codorul este 2^v şi este o măsură a complexităţii codului.

- vectorul stării viitoare S_{k+1} se obţine din vectorul stării curente S_k prin deplasare la dreapta cu o poziţie şi introducera noului bit pe poziţia eliberată. Descrierea modului în care registrul de deplasare trece dintr-o stare în alta poate fi facută cu ajutorul matricii de tranziţie T, care are dimensiunile υ x υ, construită în felul următor:

$$T = \begin{bmatrix} g_0[2] & 1 & 0 & \cdots & 0 \\ g_0[3] & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_0[v] & 0 & 0 & \cdots & 1 \\ g_0[v+1] & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{4}$$

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

$$S_{k+1} = S_k \cdot T + D_k \tag{5}$$

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

- În cazul acestui tip de coduri bitul de informaţie este transmis în clar, deci vectorii generatori g_0 şi g_1 vor avea expresiile date în relaţia (5), iar la ieşirea c_1 a codorului se va obţine valoarea bitului informaţional, $c_1(u_k,S_k) = u_k = d_k$., vezi relaţia (2):

$$g_0 = [1,0,...,0]; \quad g_1 = [1,0,...,0]; \quad \text{fig. 6 curs TCM din care se taie legatura } B_{k-2} \text{ la XOR-}c_0 \tag{6}$$

1.1.3 Coduri convoluţionale nerecursive nesistematice

- În cazul acestui tip de cod fiecare din cei n biţi codaţi sunt obţinuţi prin câte o combinaţie liniară a biţilor din registrul de deplasare şi a bitului informaţional curent, vezi cursul de TD. În acest caz vectorul generator g_0 are forma:

$$g_0 = [1,0,...,0] \qquad \text{fig. 6 curs TCM, an IV TD} \tag{7}$$

- Astfel ales vectorul g_0, valoarea bitului u_k de la intrarea registrului de deplasare este aceeaşi cu cea a bitului informaţional, d_k, iar biţii codaţi $C(d_k, S_k)$ se obţin pe baza relaţiei (3).

- în literatura de specialitate se arată că performanţele codurilor convoluţionale 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 combinaţii de polinoame generatoare, aceste coduri pot fi coduri catastrofice

1.1.4. Coduri convoluţionale recursive sistematice

- această clasă de coduri asigură performanţe 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 biţilor existenţi în registrul de deplasare (memoria co-dorului) la intrarea acestuia, iar bitul d_k informaţional curent este transmis ca atare, „în clar”.

- pentru obţinerea acestor coduri, trebuie ca vectorii generatori g_1 şi g_0 să fie identici:

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

- astfel informaţia obţinută la intrarea registrului de deplasare se obţine din sumarea modulo 2 a bitului informaţional cu poziţiile din registrul de deplasare indicate de vectorul generator g_0.
- dacă vectorul generator g_1 satisface condiţia (8) atunci la ieşirea c_1 a codorului se obţine:

$$\begin{aligned} c_1(d_k, S_k) &= u_k + \sum_{i=1}^{v} S_k(i) \cdot g_1(i+1) = d_k + \sum_{i=1}^{v} S_k(i) \cdot g_0(i+1) + \sum_{i=1}^{v} S_k(i) \cdot g_1(i+1) \\ &= d_k + \sum_{i=1}^{v} S_k(i) \cdot g(i+1) + \sum_{i=1}^{v} S_k(i) \cdot g(i+1) = d_k \quad daca\ g_0 = g_1 = g \end{aligned} \tag{9}$$

cee ce arată că satisfacerea condiţiei (8) conduce la un cod sistematic, chiar dacă bitul de intrare este obţinut în mod recursiv din bitul informaţional curent d_k şi biţi 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 sfârşitul secvenţei informaţionale a unei terminaţii de K-1= *v* biţi de „0” vezi Fig. 9 curs TCM, în cazul codurilor recursive aducerea codorului în starea 0 se face în mod diferit.
- pentru a aduce codorul în starea 0, valoarea la ieşirea sumatorului corespunzător vectorului generator g_0 trebuie forţată în 0, adică u_k=0.
- pentru aceasta sumatorul se construieşte ca în figura 2. Cât timp sunt date la intrare, comutatorul *COM* va fi pe poziţia 1; dacă se doreşte aducerea codorului în starea 0, comutatorul *COM* va trece în poziţia 2, forţând valoarea ieşirii u_k în 0.

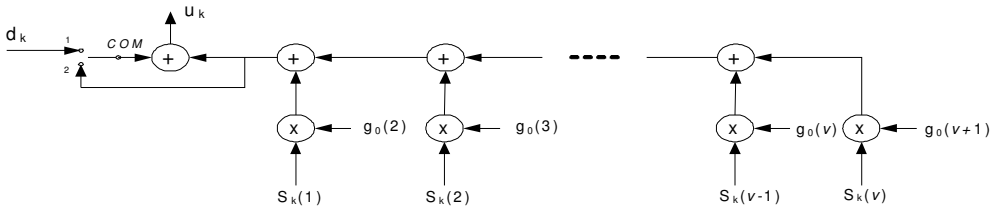


Figura 2. Sumatorul corespunzător vectorului generator g_0

- terminaţia introdusă la sfârşitul fiecărui bloc de date, pentru a aduce codorul în starea zero şi a face ca decodarea unui bloc de date să fie independentă de decodarea blocului anterior, duce la scăderea eficienţei spectrale a transmisiei, scădere care poate deveni semnficativă în cazul blocurilor codate având o lungime redusă (număr redus de biţi).
- inserarea terminaţiei face ca la decodare să se cunoască întotdeauna stările iniţială şi finală ale decodorului, care vor fi identice şi egale cu zero.
- de asemenea inserarea terminației asigură existența unor informații suplimentare pentru decodarea ultimilor K-1 = υ biți informaționali, la recepție
- o altă abordare care nu necesită inserarea unor biţi suplimentari ca terminaţie, o constituie utilizarea codurilor convoluţionale circulare, descrise în literatură.
- în principiu această metodă urmăreşte determinarea stării codorului S_c, pentru care codorul plecând din această stare ca stare iniţială (S_0= S_c) va ajunge după N tacţi 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 iniţializând codorul cu starea 0, ceea ce reprezintă un dezavataj al acestei abordări deoarece pentru codarea unui bloc de informaţie, procesul de codare trebuie aplicat de două ori, 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 convoluţionale

- puncturarea este procesul de eliminare a unor biţi codaţi de la ieşirea unui codor în vederea obţinerii unei **rate de codare mai mari,** pornind de la un cod convoluţional „părinte” de rată R_m= 1/n.
- pentru a obţine o rată a codului R_c= p/q >1/n trebuie eliminaţi $n \cdot p - q$ biţi la fiecare $n \cdot p$ biţi codaţi cu codul „părinte”.
- alegerea biţilor care urmează să fie eliminaţi pentru a obţine o rată de codare dată se face pe baza

unei măşti de puncturare, vezi curs TD, sau folosind algoritmi de adaptare de rată („rate-matching"), cum ar fi cel specificat în standardul 3GPP.
- măştile de puncturare sunt structuri bidimensionale, de dimensiune *nxp* care conţin elemente de 1 şi 0, unde valoarea 1 ne indică faptul că bitul de pe poziţia respectivă este păstrat (transmsmis) iar 0 indică faptul că bitul este eliminat (nu este transmis). Masca de puncturare se aplică la ieşirea codorului pentru fiecare n·p biţi codaţi.
- unele măşti de puncturare sunt prezentate în pag. 2 a capitolului TCM din cursul de TD.
- decodarea se face folosind decodorul codului părinte, prin inserarea elementului neutru al operaţiei de calcul a metricii folosite de decodor în poziţile biţilor codaţi care au fost eliminaţi la emisie.
- performanţele codurilor convoluţionale de rată R_c obţinute prin puncturare, sunt similare cu cele ale codurilor convoluţionale nepuncturate de rată R_c, pentru o aceeaşi constrângere K. Acest lucru a dus la o largă răspândire a acestui tip de coduri, datorită faptului că adaptarea de rată se face foarte uşor, iar decodarea se face folosind doar decodorul codului părinte pentru o gamă largă de rate obţinute prin puncturare. - de făcut comparaţia cu prescurtarea codurilor bloc

1.3. Decodarea codurilor convoluţionale utilizând algoritmul MAP

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

- algoritmul MAP utilizat pentru decodarea codurilor convoluţionale recursive este o variantă modificată, de către C. Berrou et all., a unui algoritm clasic denumit BCRJ.
- spre deosebire de algoritmul Viterbi, vezi capitolul TCM din cursul TD, care determină cea mai plauzibilă secvenţă de biţi 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 recepţionat. O descriere detaliată a algoritmului MAP folosit pentru decodarea codurilor convoluţionale 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 convoluţional recursiv cu constrângerea K = *v+1*.
- se notează cu $S_k=\{s_1, s_2, ..., s_v\}$ starea registrului de deplasare al codorului în perioada de simbol *k*, stare ce depinde doar de valoarea celulelor de memorie ale codorului. Bitul informaţional de la intrarea codorului, în momentul *k* se notează cu d_k şi corespunde tranziţiei dintre stările S_k şi S_{k+1}. Pentru fiecare bloc de de biţi, starea iniţială şi starea finală ale codorului sunt forţate la zero.
- folosind un cod RSC (Recursive Systematic Convolutional) de rată 0.5 se codează o secvenţă de date de *N* biţi, din care *N-v* biţi de cod corespunzători biţilor informaţionali, iar ultimii v biţi sunt astfel aleşi astfel încât să forţeze starea codorului în zero.
- la ieşirea codorului, pentru fiecare bit informaţional, d_k se obţine câte un bit de control c_k.
- se consideră că secvenţa codată obţinută este transmisă în formă bipolară pe un canal de tip AWGN; astfel secvenţa recepţionată este $R_1^N = \{R_1,...,R_N\}$, unde fiecare element este de forma $R_k=\{x_k, y_k\}$ descrisă de (10), iar $n_{d,k}$ şi $n_{c,k}$ reprezintă valorile zgomotului care afectează ce doi biţi, fiind două variabile aleatoare cu distribuţie normală de deviaţie standard σ^2 şi medie nulă.

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

- raportul de plauzibilitate logaritmic, LLR_k (11), calculat la recepţie pentru bitul de date emis la momentul *k*, se scrie ca fiind logaritmul natural al raportului dintre probabilitatea *aposteriori* ca bitul emis sa fie "1" şi probabilitatea *aposteriori* ca bitul emis să fie „0"; ambele trebuie calculate în ipotezele că la emisie starea codorului era *m*, iar secvenţa receptionată este R_1^N, vezi (10)

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

unde $m \in \{0,1,...,2^v - 1\}$ 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 recepţionat secvenţa R_1^N, se poate scrie ca un raport de două probabilităţi, care în urma aplicării succesive a teoremei lui Bayes privitoare la probabilităţi condiţionate, vezi (30) din anexa de la pag.8 care este reluată aici, are expresia (12), în care $p(R_1^N)$ este probabilitatea de recepţionare a secvenţei R_1^N pentru orice d_k şi orice stare *m.*

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

$$\lambda_k(i,m) = p\left(d_k = i, S_k = m \mid R_{k-1}^N, R_k^N, R_{k+1}^N\right)$$
$$= \frac{p\left(R_{k-1}^N \mid S_k = m, d_k = i, R_k^N\right) \cdot p\left(R_{k+1}^N \mid S_k = m, d_k = i, R_k\right) \cdot p\left(S_k = m, d_k = i, R_k^N\right)}{p\left(R_k^N\right)} \quad (12)$$

- primul factor al numărătorului ecuaţiei (12), este probabilitatea de a se recepţiona secvenţa R_{k-1}^N, condiţionată de S_k= m, d_k= i şi R_k^N, denumită *metrică de stare înainte*; acesta se poate simplifica datorită faptului că secvenţa recepţionată până la momentul k nu este condiţionată decât de starea curentă a codorului, S_k.

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

- similar, cel de-al doilea factor al numărătorului ecuaţiei (12), denumit *metrică de stare înapoi,* poate fi scris sub forma (14), în care f(i,m) reprezintă starea viitoare a decodorului dacă starea curentă este m iar bitul informaţional ia valoarea i („0" sau „1"):

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

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

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

- înlocuind ecuaţiile (13), (14) şi (15) în ecuaţia (12) se obţine:

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

- folosind (16), relaţia de calcul a raportului de plauzibilitate logaritmic (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 şi de tranziţie

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

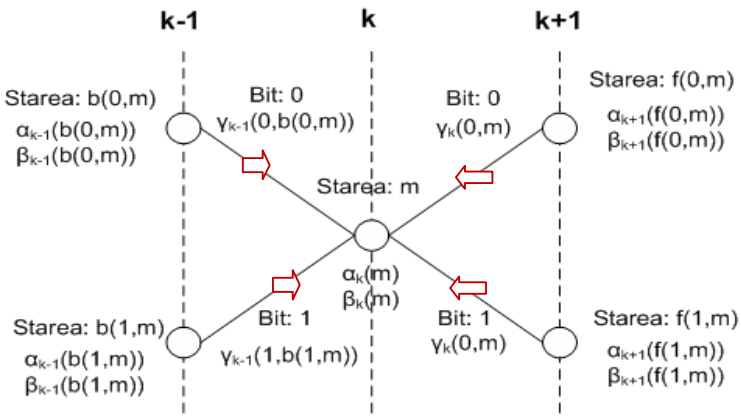
- pornind de la relaţia (13) şi folosind o abordare recursivă, valoarea *metricii înainte* pentru starea *m* are expresia (18), unde *b(i,m)* reprezintă starea din care se ajunge în starea *m* cu bitul de intrare *i*; *b(i,m)* se poate determina cu matricea *T* - rel (4)

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

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

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

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



- metrica de stare înapoi trebuie calculată abia după recepţionarea tuturor fazorilor corespunzători unui bloc codat; această operaţie introduce întârziere („latency")

- figura 3 ilustrează grafic modul de calcul al metricilor de stare şi a celor de tranziţie.

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, şi va stoca aceste valori;
- valorile metricilor de tranziţie şi va stoca aceste valori.

- După ce a fost recepţionat tot blocul codat, algoritmul va determina metricile de stare înapoi, pornind de la celălalt capăt al secvenţei de fazori recepţionaţi.
- datorită faptului că starea iniţială şi finală a codorului sunt zero metricile de stare se iniţializează în felul următor:

$$\alpha_1(m)=\begin{cases}1, m=0\\ 0, \hat{i}n\ rest\end{cases}\qquad \beta_{N+1}(m)=\begin{cases}1, m=0\\ 0, \hat{i}n\ rest\end{cases} \tag{20}$$

- Metricile de tranziţie pot fi calculate folosind de două ori succesiv teorema lui Bayes privitoare la probabilităţi 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, şi că probabilitatea de a ajunge în starea m dacă bitul de intrare d_k = i este $1/2^{\nu}$, în felul următor:

$$p(A;B)=p(A/B)p(A)=p(B/A)p(B) \tag{31}$$

$$\gamma_k(i,m)=p\left(S_k=m,d_k=i,R_k\right)=p\left(R_k\mid S_k=m,d_k=i\right)\cdot p\left(S_k=m\mid d_k=i\right)\cdot p\left(d_k=i\right)$$
$$=p\left(R_k\mid S_k=m,d_k=i\right)\cdot\frac{1}{2^{\nu}}\cdot\frac{1}{2}=p\left(x_k\mid S_k=m,d_k=i\right)\cdot p\left(y_k\mid S_k=m,d_k=i\right)\cdot\frac{1}{2^{\nu+1}};\qquad R_k=(x_k,\ y_k) \tag{21}$$

- Dacă considerăm canalul de tip AWGN de medie nulă şi deviaţie standard σ^2, atunci metrica de tranziţie se poate exprima, pornind de la relaţia (21), în felul următor:

$$\gamma_k\left(i_k,m\right)=\frac{1}{2^{\nu+1}\cdot\sigma\cdot\sqrt{2\cdot\pi}}\cdot e^{\left(-\frac{\left(x_k-\left(2\cdot i_k-1\right)\right)^2}{\sigma^2}\right)}\cdot\frac{1}{\sigma\cdot\sqrt{2\cdot\pi}}\cdot e^{\left(-\frac{\left(y_k-\left(2\cdot c\left(i_k,m\right)-1\right)\right)^2}{\sigma^2}\right)} \tag{22}$$
$$=\frac{1}{2^{\nu+1}\cdot\sigma^2\cdot 2\cdot\pi}\cdot e^{\frac{2}{\sigma^2}\cdot\left(x_k\cdot i'+y_k\cdot c'\left(i,m\right)\right)}$$

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

- Constantele $e^{-\frac{\left(x_k^2+y_k^2\right)}{2\sigma^2}}$ şi $e^{-\frac{\left(i_k^2+c_k^2\right)}{2\sigma^2}}$, obţinute prin dezvolatarea pătratelor, se neglijează pentru că nu afectează semnul LLR-ului care e esenţial la decizia valorii bitului respectiv

- Constanta $\frac{1}{2^{\nu+1}\cdot\sigma^2\cdot 2\cdot\pi}$ poate fi ignorată din acelaşi motiv

- Expresia *LLR* se obţine înlocuind relaţia (22) în ecuaţia (17) şi rezultă:

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

- Primul termen al relaţiei (23) reprezinta *informaţia intrinsecă*, adică informaţia care se obţine despre acest bit din fazorul recepţionat (a cărui valoare este efectată de canalul de transmisie)

- Al doilea termen al relaţiei (23) reprezintă *informaţia extrinsecă*, $L_e(x_k)$, adică informaţia suplimentară care se poate obţine, din semnalele corespunzătoare celorlaţi biţi ai blocului codat şi din intercondiţionările dintre aceştia (impuse de modul de construcţie a codului), despre un anume bit informaţional, în acest caz bitul *k*.

- $L_e(x_k)$ este informaţie "soft" deoarece poate lua un număr mult mai mare de valori (teoretic ea e o variabilă continuă şi nu discretă) decât numătul de valori pe care îl poate lua bitul (care e o informaţie "hard").

- Această informaţie este importantă în cazul codurilor concatenate deoarece permite transferul de informaţie soft între decodoare, adică a efectului decodării cu fiecare cod asupra LLR-ului bitului respectiv sau, în cazul decodoarelor iterative, unde informaţia „soft"este utilizată pentru actualizarea LLR-ului bitului respectiv la fiecare iteraţie.
- Pe baza raportului de plauzibilitate logaritmic (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:

$$\tilde{d}_k = \begin{cases} 0; \, daca \; LLR_k < 0 \\ 1; \, daca \; LLR_k > 0 \end{cases} \tag{24}$$

1.3.2 Algoritmul Max-Log-MAP
- După cum reiese din descrierea de mai sus, algoritmul MAP este foarte complex din punct de vedere computaţional datorită numărului mare al operaţiilor de înmulţire ce trebuie efectuate.
- Pentru eliminarea acestui neajuns s-a elaborat o variantă care operează cu logaritmii metricilor de stare şi tranziţie şi efectuează operaţiile în domeniul logaritmic, în care înmulţirile se "transformă" în adunări.
- Astfel, prin aplicarea logaritmului natural relaţiilor (18) şi (19), relaţiile de calcul ale metricilor înainte şi înapoi devin (25) şi, respectiv (26), în care $\tilde{\alpha}, \tilde{\beta}, \tilde{\gamma}$ reprezintă varianta logaritmată a metricilor de stare şi tranziţie. Se foloseşte identitatea $U = \exp(\ln U)$

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

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

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

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

$$\begin{aligned} \tilde{\alpha}_k(m) &\approx \max\left(\tilde{\alpha}_{k-1}\left(b(0,m)\right) + \tilde{\gamma}_{k-1}\left(0, b(0,m)\right), \tilde{\alpha}_{k-1}\left(b(1,m)\right) + \tilde{\gamma}_{k-1}\left(1, b(1,m)\right)\right) \\ \tilde{\beta}_k(m) &\approx \max\left(\tilde{\beta}_{k+1}\left(f(0,m)\right) + \tilde{\gamma}_k(m), \tilde{\beta}_{k+1}\left(f(1,m)\right) + \tilde{\gamma}_k(m)\right) \end{aligned} \tag{27}$$

- Logaritmând relaţia (22) metrica de tranziţie se calculează în felul următor:

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

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

$$\tilde{\alpha}_1(m) = \begin{cases} 0, m = 0 \\ -\infty, \hat{i}n \; rest \end{cases} \qquad \tilde{\beta}_{N+1}(m) = \begin{cases} 0, m = 0 \\ -\infty, \hat{i}n \; rest \end{cases} \tag{29}$$

- Prin utilizarea aproximării logaritmului Jacobian valorile metricilor de stare obţinute 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 performanţelor decodorului (scăderea câştigului codării, adică a capacităţii de corecţie) care însă nu este semnificativă pentru valori medii şi mari ale SNR, la care $|a-b| >> 0$.
-deşi performanţele decodorului se înrăutăţesc puţin datorită aproximării utilizate, reducerea complexităţii implementării este substanţială.

Anexă
Noțiuni și relații uzuale

1. Teorema lui Bayes privitoare la probabilităţi condiţionate: $$p(A/B)=\frac{p(B/A)\cdot p(A)}{p(B)} \quad (30)$$

2. Teorema lui Bayes privitoare la probabilităţi 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 aposteriori probability
P(.) - probabilitatea unui eveniment (.)
p(.) - probability density function (pdf) a variabilei
$P_{XY/R}(x,y/r)$ – probabilitatea aposteriori (APP) - (x,y) coordonate recepționate, r fazor emis
$P_{R/XY}(r/x,y)$ – funcţia de plauzibilitate (likelihood function) - r fazor emis, (x,y) coordonate rec.

5. Relaţia dintre probabilitatea aposteriori şi funcţia de plauzibilitate
- Fie x ε X, y ε Y, variabile aleatoare discrete şi $r \in R$ o variabilă continuă (vectorul observaţiilor).
-Variabilele X şi 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 secvenţe pe baza plauzibilităţii maxime (MLD)
– Determină cea mai plauzibilă cale (secvenţă de simboluri) dintre secvenţele posibile ale unei diagrame trellis date.
- Algoritmul lui Viterbi (VA) este un algoritm eficient pentru implementarea MLD.
- Chiar dacă foloseşte variabile “soft”(set de probabilităţi sau distanţe) ca informaţii de intrare, VA furnizează la ieşire valori discrete (biţi), “hard outputs”.
- Există şi 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, ţinând cont de condiţionările impuse de diagrama trellis sau de ecuaţiile 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, şi astfel şi algoritmul BCRJ furnizează la ieşure valori discrete (biţi, simboluri) – “hard outputs”; şi acest algoritm are o variantă care furnizează „soft output”, adică probabilităţi sau LLR-uri.

8. Definirea estimatelor condiţionate (joint) şi a estimatelor simbolurilor
- Estimata condiţionată (joint) ML: $$(\hat{x},\hat{y}) = \arg\max_{x\in X, y\in Y} p(r/x,y) \quad (34)$$

- Estimata condiţionată (joint) MAP estimate:
$$(\hat{x},\hat{y}) = \arg\max_{x\in X, y\in Y} P(x,y/r) = \arg\max_{x\in X, y\in Y} p(r/x,y)\cdot P(x)\cdot P(y)/p_R(r) \quad (35)$$
- Estimata ML a unui simbol: $$(\hat{y}) = \arg\max_{y\in Y} p(r/y) = \arg\max_{y\in Y} \sum_{x\in X} p(r/x,y)\cdot P(x) \quad (36)$$
- Estimata MAP a unui simbol (această abordare se folosește pentru operația de „soft-demapping”):
$$(\hat{y}) = \arg\max_{y\in Y} P(y/r) = \arg\max_{y\in Y} \sum_{x\in X} p(r/x,y)\cdot P(x)\cdot P(y) \quad (37)$$
- 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 >=<0 este echivalentă cu comparația tanh(LLR/2) >=< 0 (vezi relația (38))
$$LLR = \ln\frac{P_0}{P_1} \Rightarrow e^{LLR} = \frac{P_0}{P_1} \Rightarrow \tanh(\frac{LLR}{2}) = \frac{e^{LLR}-1}{e^{LLLR}+1} = P_0 - P_1 \quad (38)$$
- Relația care aproximează tanh(x/2) este mai ușor de implementat decât cea care aproximează ln(y)