CONTRIBUŢII PRIVIND ÎMBUNĂTĂŢIREA TEHNICILOR DE EGALIZARE ALE CANALELOR RADIO

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea *Politehnica* Timişoara
în domeniul INGINERIE ELECTRONICĂ
ŞI TELECOMUNICAŢII
de către

ing. Florin Lucian Morgoş

Conducător ştiinţific:	prof.univ.dr.ing. Miranda Naforniţă
Referenţi ştiinţifici:	prof.univ.dr.ing. Cornelia Gordan
	conf.univ.dr.ing. Romulus Terebeş
	prof.univ.dr.ing. Alexandru Isar

Ziua susţinerii tezei: 10 Ianuarie 2014

Seriile Teze de doctorat ale UPT sunt:

1. Automatică
2. Chimie
3. Energetică
4. Ingineria Chimică
5. Inginerie Civilă
6. Inginerie Electrică
7. Inginerie Electronică şi Telecomunicaţii
8. Inginerie Industrială
9. Inginerie Mecanică
10. Ştiinţa Calculatoarelor
11. Ştiinţa şi Ingineria Materialelor
12. Ingineria sistemelor
13. Inginerie energetică
14. Calculatoare şi tehnologia informaţiei
15. Ingineria materialelor
16. Inginerie şi Management

Universitatea *Politehnica* din Timişoara a iniţiat seriile de mai sus în scopul diseminării expertizei, cunoştinţelor şi rezultatelor cercetărilor întreprinse în cadrul şcolii doctorale a universităţii. Seriile conţin, potrivit H.B.Ex.S Nr. 14 / 14.07.2006, tezele de doctorat susţinute în universitate începând cu 1 octombrie 2006.

România, 300159 Timişoara, Bd. Republicii 9,
tel. 0256 403823, fax. 0256 403221
e-mail: editura@edipol.upt.ro

CUVÂNT ÎNAINTE

Cu prilejul finalizării procesului de elaborare a tezei de doctorat în cadrul Departamentului de Comunicaţii al Universităţii "Politehnica" din Timişoara, aduc mulţumiri, în mod deosebit, conducătorului ştiinţific, D-nei prof. univ. dr. ing. Miranda Naforniţă, pentru sprijinul deplin acordat pe parcursul perioadei de studiu şi pregătire. Graţie profesionalismului său academic şi a cunoştinţelor împărtăşite, domnia sa a avut o contribuţie deosebită în realizarea şi finalizarea acestei lucrări.

Doresc să mulţumesc domnilor colegi din Departamentul de Electronică şi Telecomunicaţii din cadrul Universităţii din Oradea şi, de asemenea, D-nei prof. univ. dr. ing. Cornelia Gordan, coordonatoarea acestui departament, pentru punctele de vedere exprimate, recomandările utile furnizate şi sprijinul moral acordat pe parcursul întregii perioade de pregătire.

Mulţumesc de asemenea referenţilor ştiinţifici din Comisia de doctorat, pentru participarea la susţinerea publică a tezei, pentru bunăvoinţa şi timpul acordat analizării acestei lucrări.

În final, doresc să mulţumesc celor apropiaţi pentru înţelegerea, ajutorul şi încurajările acordate pe parcursul realizării acestei lucrări.

Timisoara, Ianuarie 2014 Morgos Lucian

Morgoș, Florin Lucian

Contribuţii privind îmbunătăţirea tehnicilor de egalizare ale canalelor radio

Teze de doctorat ale UPT, Seria 7, Nr. 67, Editura Politehnica, 2014, 148 pagini, 81 figuri, 6 tabele.

ISSN: 1842-7014

ISBN: 978-606-554-758-2

Cuvinte cheie: egalizor, turbocodor, turboegalizor, estimator.

Rezumat, Scopul principal al tezei este de a propune şi studia îmbunătăţirea tehnicilor de egalizare ale canalelor radio. Întrucât în ultimii ani a crescut exponenţial numărul de utilizatori ai reţelelor fără fir (wireless), creşterea capacităţii acestor sisteme de transmisiuni este o problemă continuă de cercetare. Capacitatea este limitată de zgomotul din canal, de interferenţe, si de modificarea stării canalului ca urmare a mobilităţii utilizatorilor. Interferenţa se poate elimina, reduce sau evita, prin tehnici adecvate de egalizare a canalului sau alte tehnici de prelucrare a semnalului. Pentru a studia importanţa egalizării s-a considerat transmisia unei secvente de date, care a fost codată cu un turbocodor şi apoi modulată BPSK. Acestă secvenţă este transmisă printr-un canal cu zgomot care are răspunsul la impuls. La partea de recepţie, egalizarea canalului şi turbodecodarea sunt realizate prima dată independent, iar apoi se fac împreună cu un turboegalizor. O problema importantă este estimarea răspunsului la impuls al canalului. Sunt prezentate metode de estimare a răspunsului la impuls al canalului precum si rezultatele obţinute în funcţie de corectitudinea estimării. Pentru rafinarea estimării este propusă estimarea iterativă care oferă performanţe superioare faţă de estimarea neiterativă.

CUPRINS

ABREVIERI

ACI - Adjacent Channel Interference
AWGN - Additive White Gaussian Noise
BCJR - Bahl-Cocke-Jelinek-Raviv Algorithm
BDFE - Block Decision-Feedback Equalizer
BER - Bit Error Rate
BPSK - Binary Phase Shift Keying
CCI - Co Channel Interference
CDMA - Code Division Multiple Access
CIR - Channel Impulse Response
CISI - Controlled Inter Symbol Interference
CSI - Channel State Information
DFE - Decision Feedback Detector/Equalizer
DFFT - Discrete Fast Fourier Transform
DFT - Discrete Fourier Transform
DMT - Discrete Multi-Ton
ECC - Error Correcting Codes
FER - Frame Error Rate
FIR - Finite Impulse Response
GMSK - Gaussian Minimum Shift Keying
IC - Interference Canceller
IDFT - Inverse Discrete Fourier Transform
IFFT - Inverse Fast Fourier Transform
ISI - Inter Symbol Interference
LDPC - Low Density Parity Check
LE - Linear Equalizer
LLR - Log-Likelihood Ratio
LMS - Least Mean Square
Log-MAP- Logarithm-MAP
LS - Least-Square
MAI - Multiple Access Interference
MAP - Maximum A Posteriori
MIMO - Multiple Input Multiple Output
ML - Maximum Likelihood
MMSE - Minimum Mean Square Error
MSE - Mean Square Error
OFDM - Orthogonal Frequency Division Multiplex
PLL - Phase Loop Locked
QAM - Quadrature Amplitude Modulation
RFI - Radio Frequency Interference
RLS - Recursive Least Square
RSC - Recursive Systematic Convolutional Code
SINR - Signal to Interference Plus Noise Ratio
SISO - Soft Input Soft Output
SNR - Signal to Noise Ratio

SOVA - Soft Output Viterbi Algorithm

WMF - Whitening Matched Filter
ZF - Zero Forcing
ZF-BLE - Zero Forcing Block Linear Equalizer

LISTA TABELELOR

LISTA FIGURILOR

4.14. (b) Tranziţiile de trellis ale egalizorului, care au fost luate în considerare atunci când calculăm valorile LLR corespunzătoare biţilor $u_m = +1$ la intervalul de trellis m=6

4.15. (a) Tranziţiile de trellis ale decodoarelor de canal cu K=3 RSC şi R=1/2, care au fost luate în considerare atunci când se calculează valorile LLR corespunzătoare bitului $c_{1,4}$ =-1, care este primul bit codat la intervalul de trellis d=4. Trellisul este bazat pe figura 4.8.

4.15. (b) Tranziţiile de trellis ale decodoarelor de canal cu K=3, RSC şi R=1/2, care au fost luate în considerare atunci când se calculează valorile LLR corespunzătoare bitului $c_{1,4}$ =+1, care este primul bit codat la intervalul de trellis d=4. Trellisul este bazat pe figura 4.8.

4.16. (a) Tranziţiile de trellis ale decodoarelor de canal cu K=3 RSC şi R=1/2, care au fost luate în considerare atunci când se calculează valorile LLR corespunzătoare bitului $c_{2,4}$ =-1, care este al doilea bit codat la intervalul de trellis d=4. Trellisul este bazat pe figura 4.8.

4.16. (b) Tranziţiile de trellis ale decodoarelor de canal cu K=3 RSC şi R=1/2, care au fost luate în considerare atunci când se calculează valorile LLR corespunzătoare bitului $c_{2,4}$ = +1, care este al doilea bit codat la intervalul de trellis d=4. Trellisul este bazat pe figura 4.8.

4.17. Schema bloc a sistemului de emisie – receptie cu estimarea lui h

4.18. BER/SNR (k=32)

4.19. BER/SNR (k=128)

4.20. BER/SNR (k=128, $h = [0.302 \quad 0.725 \quad 0.456]$, $\hat{h} = [0.423 \quad 0.567 \quad 0.231]$)

4.21. Schema bloc a sistemului de emisie – receptie cu estimarea iterativă a lui $\hat{h}$

4.22. BER/SNR pentru cazul i) $h = \hat{h}$ (k=128, $h = \hat{h} = [0.302 \quad 0.725 \quad 0.456]$)

4.23. BER/SNR pentru cazul ii) $h \neq \hat{h}$ (k=128, $h = [0.302 \quad 0.725 \quad 0.456]$, $\hat{h} = [0.423 \quad 0.567 \quad 0.231]$)

4.24. BER/SNR (k=128, $h = [0.302 \quad 0.725 \quad 0.456]$, N=8)

4.25. BER/SNR (k=128, $h = [0.302 \quad 0.725 \quad 0.456]$, N=18)

4.26. BER/SNR (k=128, $h = [0.302 \quad 0.725 \quad 0.456]$, N=128)

4.27. Sistem de transmisie utilizând tehnica OFDM

4.28. Implementarea sistemului de transmisie OFDM utilizând IDFT şi DFT

4.29. Implementarea schemei de transmisie OFDM utilizând IFFT şi DFFT cu prefix ciclic.

4.30. BER/SNR (algoritm LS , k=128 , $h = [0.302 \quad 0.725 \quad 0.456]$, N=8, L=8)

4.31. BER/SNR (algoritm LS , k=128, $h = [0.302 \quad 0.725 \quad 0.456]$, N =18, L =8)

4.32. BER/SNR (algoritm LS , k=128, $h = [0.302 \quad 0.725 \quad 0.456]$, N =64, L =8)

4.33. BER/SNR (algoritm MMSE , k=128, $h = [0.302 \quad 0.725 \quad 0.456]$, N =8, L =8)

4.34. BER/SNR (algoritm MMSE , k=128, $h = [0.302 \quad 0.725 \quad 0.456]$, N =18, L =8)

4.35. BER/SNR (algoritm MMSE, k=128, $h = [0.302 \quad 0.725 \quad 0.456]$, N =64, L =8)

1. Introducere

1.1 Limitările canalelor de comunicaţie

În ultimii ani a crescut exponenţial numărul de utilizatori ai reţelelor fără fir (wireless) ceea ce justifica cercetările pentru creşterea capacităţii acestor sisteme de transmisiuni. Capacitatea este limitată de *zgomotul* din canal, de *interferenţe*, si de *modificarea stării* canalului ca urmare a mobilităţii utilizatorilor [86, 87]. Interferenţa între simboluri, ISI (Inter Symbol Interference) este cauzată de banda limitată a canalului, iar interferenţa radio RFI (Radio Frequency Interference) poate fi cauzată de utilizatorii din aceeaşi celulă, CCI (Co Channel Interference) sau din celule adiacente, ACI (Adjacent Channel Interference), precum şi din alte cauze (cuptoare de microunde, platforme de calcul, etc.). Interferenţa se poate elimina, reduce sau evita, prin tehnici adecvate de egalizare a canalului sau alte tehnici de prelucrare a semnalului [90, 85]. Semnalul mai poate fi distorsionat şi datorită propagării pe căi multiple [88, 89].

La începutul anilor 1940, se credea că prin creşterea vitezei de transmitere a informaţiei prin canal creşte şi numărul de erori din informaţia recepţionată. Dar, în 1948, în lucrarea sa, "A Mathematical Theory of Communications" [1] , C. Shannon stabilea că daca rata maximă de transmisie a informaţiei printr-un canal de comunicaţie este mai mică decât capacitatea canalului se poate obţine o probabilitate de eroare oricât de mică, dacă sunt folosite coduri de o anumită lungime:

$$p_e \le 2^{-ne(D)} \tag{1.1}$$

unde p_e este probabilitatea erorii, n este lungimea codului, iar $e(D)$ este exponentul erorii (teorema codării canalelor cu zgomot). C. Shannon a propus utilizarea codurilor aleatoare, împreună cu regula de decodare "cel mai apropiat vecin" (the nearest neighbour), decodare extrem de complexa şi dificil de implementat, dimensiunea tabelelor de asociere (lookup) crescând exponenţial cu n. De atunci au fost create multe alte coduri corectoare de erori, ECC (Error Correcting Codes) cu structuri mult mai simple şi simplu de decodat. Codurile corectoare de erori au îmbunătăţit considerabil calitatea semnalului livrat de receptor, utilizatorului.

Introducerea turbo codurilor, de C. Berrou, A. Glavieux, şi P. Thitimajshina [2] a permis rate de eroare foarte scăzute, la rate de transmisiune apropiate de capacitatea canalului, indicată de C. Shannon, precum şi câştiguri de codare apropiate de limita Shannon. Din 1993 au fost depuse multe eforturi pentru îmbunătăţirea lor şi înţelegerea cauzelor performanţelor lor deosebite. Principiul turbo constă în combinarea a două sau mai multor tehnici şi executarea lor serie, iterativ. El a fost extins şi în alte domenii, primul fiind turbo egalizarea pentru

eliminarea ISI, idee propusă de C. Douillard s.a. [3]. Canalul este asimilat cu un cod convoluţional nerecursiv nesistematic, care se codează cu un codor extern (outer coder), cu un interleaver plasat între cele două. Detecţia turbo e similară cu detecţia iterativă a două coduri convoluţionale concatenate serial; aceeaşi tehnică poate fi aplicată pentru realizarea combinată a detecţiei şi decodării [4]. Egalizarea şi decodarea ieşirii soft a canalului sunt concatenate şi realizate iterativ.

Iteraţiile din procesul de decodare turbo îmbunătăţesc spectaculos rata erorii pe bit BER (Bit Error Rate) şi pe cadru, FER (Frame Error Rate); din simulări rezultă că ISI poate fi eliminat complet pentru un canal gaussian, daca exista un interleaver suficient de lung şi o estimare perfectă a canalului.

Calitatea informaţiei livrată utilizatorului depinde de starea canalului, de disponibilitatea sau nu a informaţiilor despre starea acestuia (CSI-Channel State Information) de modul de propagare, de modulaţia folosită precum şi de tipul de detecţie.

Pentru eliminarea efectelor ISI poate fi folosită egalizarea liniară, LE (Linear Equalizer), sau o prelucrare neliniară DFE (Decision Feedback Detector/Equalizer). Dar metodele optimale de minimizare a BER şi FER sunt neliniare, şi sunt bazate pe estimarea ML (Maximum Likelihood), ca de exemplu algoritmul Viterbi de optimizare a FER [5]. Din estimarea ML derivă şi estimarea MAP (Maximum A Posteriori), ca de exemplu algoritmul BCJR Viterbi de optimizare a BER când există informaţii a priori despre date. În prima lucrare, C. Douillard s.a. au folosit pentru egalizare estimarea ML a secvenţei şi decodarea canalului cu ieşiri soft (algoritmul BCJR pentru detecţia de simbol, respectiv algoritmul SOVA, de complexitate mai redusă pentru decodarea de canal).

Complexitatea algoritmilor bazaţi pe ML sau MAP, creşte exponenţial cu lungimea răspunsului la impuls al canalului, CIR (Channel Impulse Response) şi dimensiunea modulaţiei; de aceea cele mai folosite sunt modulaţiile simple, BPSK (Binary Phase Shift Keying), GMSK (Gaussian Minimum Shift Keying) respectiv CIR cu 6 prize. În consecinţa, primul scop este reducerea complexităţii turbo detectorului. Detectorul ISI contribuie cel mai mult la complexitate, datorită decodării soft a canalului, pentru determinarea informaţiei extrinseci, furnizată apoi ca intrare în turbo egalizor. Multe studii s-au ocupat de scăderea complexităţii turbo detectorului, folosind un estimator suboptimal, SISO (Soft Input Soft Output). O altă metodă pentru reducerea complexităţii este înlocuirea detectorului MAP cu un supresor de ISI, cu rezultate foarte bune pentru canale Rayleigh invariante sau nu în timp şi modulaţii de eficientă spectrală mare. Glavieux [6] a fost printre primii care au aplicat metoda cu supresor ISI, care este de fapt un filtru FIR cu câştiguri adaptive, a cărui complexitate depinde doar liniar de lungimea CIR. A mai propus un turbo egalizor simplu SISO, M-ar, ce reduce drastic ISI, actualizând în fiecare etapă parametrii egalizorului conform criteriului MSE (Mean Square Error) [7].

1.2 Canalul AWGN

Vom descrie un model în banda de bază şi timp discret pentru un sistem digital de comunicaţii şi problema detecţiei, precum şi codarea şi decodarea de canal, pentru transmiterea sigură a informaţiei, la debite şi precizii ridicate. Capacitatea canalului AWGN (Additive White Gaussian Noise) sau rata de maximă a informaţiei care poate fi trimisă prin un canal, este dată de formula stabilită de C.Shannon, pe baza lucrărilor lui H. Nyquist si R.Hartley:

$$C = B\log_2\left(1+\frac{P_S}{P_N}\right) \tag{1.2}$$

unde C [bps], B banda de frecvenţe a canalului [Hz], P_S [W] este puterea semnalului, iar P_N este puterea zgomotului din canal. O altă exprimare este:

$$C_{AWGN} = \log_2\left(1+\frac{\varepsilon_S}{N_0}\right) \tag{1.3}$$

unde raportul $C_{AWGN} = C/B\ \left[bit/s/Hz\right]$ (eficienţa spectrală), care derivă din relaţia (1.2), se foloseşte cu sens de capacitate; la fel, în relaţia (1.3), in loc de raportul P_S/P_N se foloseşte raportul echivalent dintre energia bitului şi densitatea spectrală de putere ε_S/N_0 .

În prezent sunt pe larg studiate proiectarea codurilor şi tehnicile practice de decodare pentru rate apropiate de capacitatea canalului AWGN. La SNR mic, sunt suficiente codurile de rată mică - turbo codurile [2] şi codurile LDPC (Low Density Parity Check), coduri cu controlul parităţii, de densitate mică [8, 9] - care se apropie de limitele teoretice. S-a demonstrat că un cod LDPC poate opera la o valoare a SNR, apropiata de limita Shannon, cu doar 0.0045 dB peste valoarea SNR minimă necesară pentru codurile cu această rată [10]. Pentru atingerea capacităţii, în regimurile cu valori SNR mari, unde sunt necesare coduri cu rată mare, pot fi folosite codurile multinivel, cu mai multe nivele de decodare [11, 12].

1.3 Modele de canale cu interferenţă

Consideram un model de canal în timp discret, în banda de bază, care abstractizează canalul şi ascunde detaliile implementării. Cu acest model putem trata diverse sisteme de comunicaţii şi tipuri de interferenţe, având acelaşi spaţiu al semnalelor. Consideram un vector **X**, $N\ X\ 1$ al datelor echiprobabile, ce trebuie transmise prin canal. Componentele vectorului pot fi purtătoare de diferite frecvente, diferite momente de timp, etc. Interferenţa canalului este considerată liniară, modelată prin înmulţirea lui **X** cu **H**, o matrice $Q\ X\ N$. Deoarece zgomotul (un vector **W** complex) din canal rezultă din suprapunerea mai multor acţiuni independente, teorema limită centrală sugerează că se poate modela ca un zgomot gaussian, alb şi aditiv (AWGN), de medie nulă, cu varianţa N_0 . Vectorul **R**, $Q\ X\ 1$ de la receptor este - figura 1.1:

$$\mathbf{R=HX+W} \tag{1.4}$$

Vom urmări, detecţia la receptor a vectorului **X** transmis, cunoscând **R**, **H** şi statisticile vectorului **W**. Parametrii lui **H** pot fi învăţaţi la receptor prin antrenare (training). Estimarea lui **H** se face prin transmiterea de vectori cunoscuţi la transmiţător şi receptor. Când canalul se modifică în timp **H** poate fi actualizată cu ajutorul deciziilor de detecţie. Daca urmărirea eşuează se reantrenează periodic. În majoritatea cazurilor se consideră cunoscute la receptor **H** şi statisticile vectorului **W**. Sistemele de detecţie dezvoltate sunt aplicabile la orice scenariu pentru care este valabila relaţia (1.4).

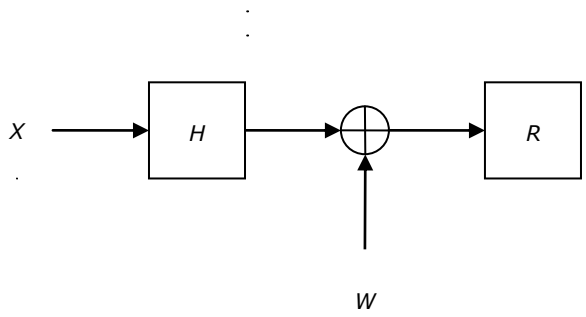


Figura 1.1. Modelul canalului

• Un exemplu de sistem de comunicaţii, în care se aplică modelul de canal (1.4), este scenariul în timp discret a N utilizatori, CDMA sincron, pe legătura uplink - figura 1.3. În acest sistem, utilizatorul i modulează cu simbolul complex x_i o secvenţă de lungime Q, $h_i[k]$ - semnătura (signature), alocată acelui utilizator. Secvenţa modulată este transmisă prin canal, unde suferă o atenuare A_i. Staţia de bază primeşte semnalele de la toţi utilizatorii plus zgomotul, aşa cum se vede din (1.4) , în care coloanele matricei **H** sunt semnăturile utilizatorilor scalate cu factorii de atenuare corespunzători canalului, adică,

$$\begin{bmatrix} r[0] \\ r[1] \\ \vdots \\ r[Q-1] \end{bmatrix} = \begin{bmatrix} A_0h_0[0] & A_1h_1[0] & \cdots & A_{N-1}h_{N-1}[0] \\ A_0h_0[1] & A_1h_1[1] & \cdots & A_{N-1}h_{N-1}[1] \\ \vdots & \vdots & \ddots & \vdots \\ A_0h_0[Q-1] & A_1h_1[Q-1] & \cdots & A_{N-1}h_{N-1}[Q-1] \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix} + \begin{bmatrix} w[0] \\ w[1] \\ \vdots \\ w[Q-1] \end{bmatrix} \quad (1.5)$$

• Un alt exemplu în care se aplică (1.4), este un sistem sincron cu antene multiple, cu N antene de transmisie şi Q antene la recepţie - figura 1.4. Fiecare antenă transmite un alt simbol complex x_i pe canal. Pentru transmisia de bandă îngustă, calea de la antena de emisie i la antena de recepţie j este descrisă printr-un singur coeficient de fadingul plat h_{ji}, care poate fi introdus în matricea **H**. Fiecare antenă de la recepţie primeşte semnale suprapuse de la toate antenele transmiţătoare, cu zgomot alb, deci modelul din (1.4):

$$\begin{bmatrix} r[0] \\ r[1] \\ \vdots \\ r[Q-1] \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & \cdots & h_{0,N-1} \\ h_{10} & h_{11} & \cdots & h_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{Q-1,0} & h_{Q-1,1} & \cdots & h_{Q-1,N-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix} + \begin{bmatrix} w[0] \\ w[1] \\ \vdots \\ w[Q-1] \end{bmatrix} \quad (1.6)$$

• Canalul în timp discret "point-to-point", cu interferenţă inter-simbol (ISI), reprezentat în figura 1.5, poate fi, de asemenea, modelat folosind (1.4). Datele transmise sunt un flux de simboluri complexe x_i, afectate de convoluţia cu

răspunsul la impuls al canalului (cu ISI), h_i , şi de zgomotul aditiv, w_i , obţinând simbolul de la recepţie:

$$r_i = \sum_k h_{i-k} x_k + w_i \tag{1.7}$$

Dacă r_i , x_i , şi w_i sunt aranjate fiecare în format vectorial şi **H=**$\left[h_1,....,h_p\right]$ iar h_{i-k} este versiunea întârziată a lui h_i , atunci se va obţine din nou modelul (1.4). Un exemplu cu un răspuns la impuls de lungime doi este

$$\begin{bmatrix} r_0 \\ r_1 \\ \vdots \\ r_{N-1} \end{bmatrix} = \begin{bmatrix} h_0 & 0 & 0 & \cdots & 0 \\ h_1 & h_0 & 0 & \cdots & 0 \\ 0 & h_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & h_0 & 0 \\ 0 & \cdots & 0 & h_1 & h_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix} + \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{N-1} \end{bmatrix} \tag{1.8}$$

Se observă că matricea **H** este pătratică în acest caz şi de tip Toeplitz.

1.3.1 Alocarea optimala a puterii - water filling

În sistemele de comunicaţii pentru a avea o rată maximă a informaţiei alocarea optima a puterii diferitelor componente ale lui **X** se face aplicând teorema "water pouring/filling". Principiul water filling prevede transmisia începând cu canalele mai puţin zgomotoase si trecând progresiv spre cele mai zgomotoase, pe măsură ce se ocupa primele. Puterea trebuie crescută şi ea, progresiv, pentru menţinerea unui raport semnal/zgomot acceptabil.

Aplicând principiul water filling poate fi maximizată capacitatea canalului MIMO (Multiple Input Multiple Output), care este suma capacităţilor canalelor individuale SISO (Single Input Single Output) paralele. [13]:

$$C = \max_{\{\varepsilon_{s,k}\}\sum_{k=1}^{N}\varepsilon_{s,k}\le\varepsilon,\varepsilon_{s,k}\ge 0} \sum_{k=1}^{N} log_2\left(1+\frac{\varepsilon_{s,k}}{N_0}\lambda_K\right) \tag{1.9}$$

În relaţia de mai sus N_0 este densitatea spectrala de putere, $\lambda_k = |h_k|^2$ reprezintă câştigul subcanalului k , ε_s este energia care trebuie alocată pentru cele N canale, iar $\varepsilon_{s,k}$ este energia transmisă prin subcanalul k

Găsirea acestui maxim se poate face cu ajutorul **funcţiei de tip Lagrange.**

$$L\left(\left\{\varepsilon_{s,k};\lambda\right\}\right) = \sum_{k=1}^{N} log_2\left(1+\frac{\varepsilon_{s,k}}{N_0}\lambda_K\right) + \lambda\left(\sum_{k=1}^{N}\varepsilon_{s,k} - \varepsilon_s\right) \tag{1.10}$$

$$\Rightarrow \frac{\partial L}{\partial \varepsilon_{s,k}} = \frac{\lambda_k}{N_0} \frac{log_2 e}{\left(1 + \frac{\varepsilon_{s,k}}{N_0} \lambda_K\right)} + \lambda = 0 \tag{1.11}$$

$$\Rightarrow \forall k, \left(\frac{N_0}{\lambda_k} + \varepsilon_{s,k}\right) = \mu \Rightarrow \forall k, \left(\frac{N_0}{\lambda_k} + \varepsilon_{s,k}\right) = \mu \ (\mu \text{ este o constantă}) \tag{1.12}$$

Întrucât $\varepsilon_{s,k} \geq 0$

$$\varepsilon_{s,k} = \left(\mu - \frac{N_0}{\lambda_k}\right)^+ \tag{1.13}$$

În care $(x)^+$ este definit ca:

$$(x)^+ = \begin{cases} x & daca \quad x \geq 0 \\ 0 & daca \quad x < 0 \end{cases} \tag{1.14}$$

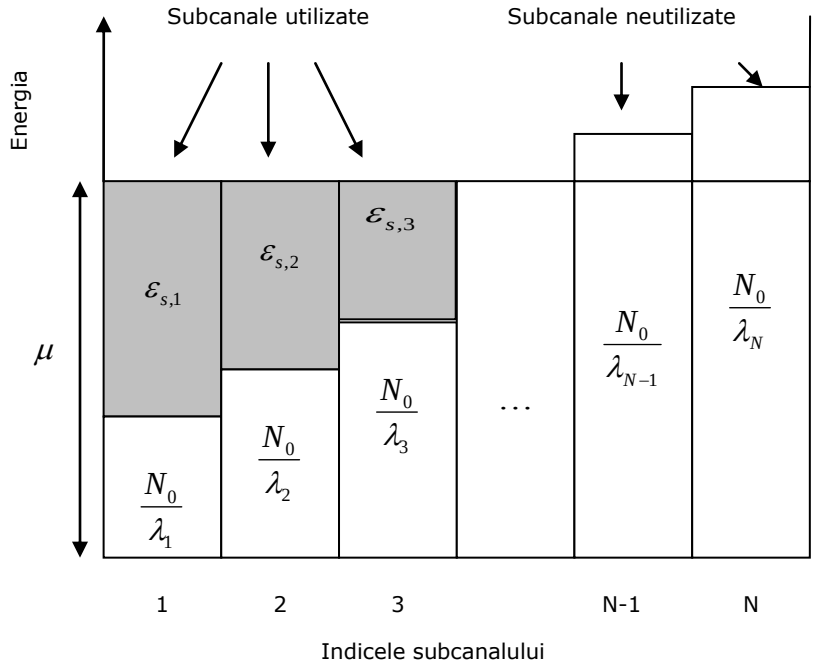


Figura 1.2. Alocarea energiei utilizând algoritmul "water pouring/filling"

Dacă matricea de interferenţă **H** este nontrivială şi cunoscută la emisie, energia transmisă prin subcanalul *k* va fi [13]:

$$\varepsilon_{s,k} = max\left(\mu - \frac{N_0}{\lambda_k}, 0\right) \tag{1.15}$$

unde μ este ales astfel încât energia de emisie medie să fie

$$\varepsilon_s = \frac{1}{N}\sum_{k=0}^{N-1}\varepsilon_{s,k} \tag{1.16}$$

Pentru a maximiza informaţia mutuală, transmiţătorul poate accesa individual aceste subcanale şi să le aloce niveluri variabile de putere.

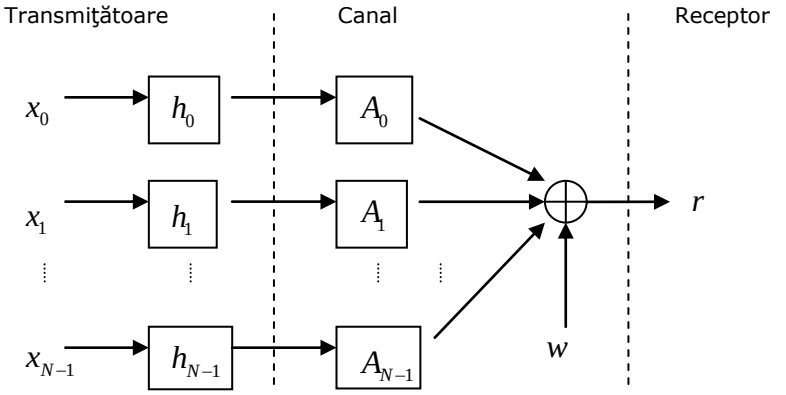


Figura 1.3. Exemplu Uplink pentru sistemul CDMA

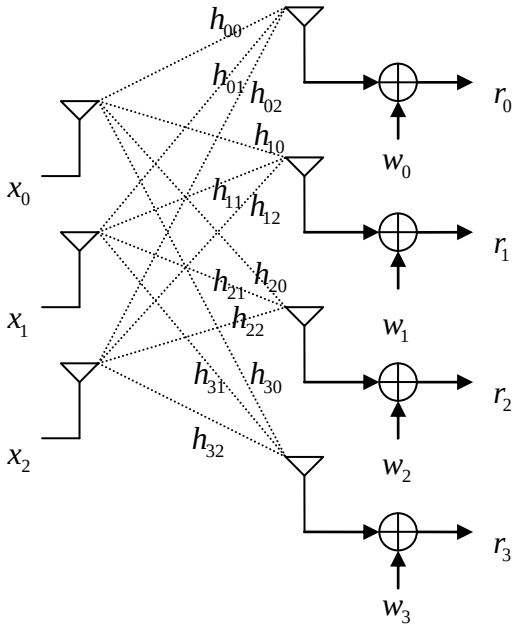


Figura 1.4. Sistem de antene multiple

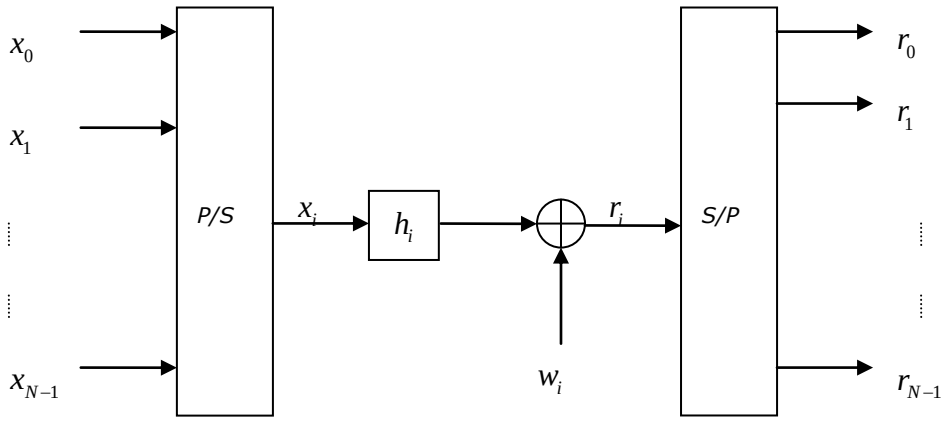


Figura1.5. Canal cu ISI

Dacă *K* este un set de subcanale, iar puterea de transmisie este alocată prin intermediul algoritmului '' water pouring '', atunci capacitatea canalului cu interferenţe (în bit/s/Hz) este:

$$C_{int} = \frac{K}{N} log_2 \left(\frac{\frac{\varepsilon_s}{N_0}\frac{N}{K} + \left\langle \frac{1}{\lambda_k} \right\rangle_{A,K}}{\left\langle \frac{1}{\lambda_k} \right\rangle_{G,K}} \right) \tag{1.17}$$

unde $\left\langle \frac{1}{\lambda_k} \right\rangle_{A,K}$ şi $\left\langle \frac{1}{\lambda_k} \right\rangle_{G,K}$ reprezintă media aritmetică şi respectiv geometrică a rapoartelor $\frac{1}{\lambda_k}$ pentru cele K subcanale şi sunt date de relaţiile:

$$\left\langle \frac{1}{\lambda_k} \right\rangle_{A,K} = \frac{1}{|K|} \sum_{k \in K} \frac{1}{\lambda_k} \tag{1.18}$$

$$log \left\langle \frac{1}{\lambda_k} \right\rangle_{G,K} = \frac{1}{|K|} \sum_{k \in K} log \frac{1}{\lambda_k} \tag{1.19}$$

Azi sunt cunoscute tehnici care fac posibilă utilizarea capacităţii canalelor cu interferenţă aproape la fel ca şi a capacităţilor canalelor fără interferenţă, cu condiţia ca emiţătorul să fie informat cu privire la interferenţe. Un sistem de comunicaţie, care încorporează astfel de tehnici este prezentat în figura 1.6. La emiţător, cunoaşterea informaţiilor despre starea canalului este folosită într-un

"pre-anulator" de interferenţe, care alocă optim puterea de transmisie subcanalelor. Grupul format din emiţător cu pre-anulator şi canal poate fi privit ca un canal fără interferenţe, astfel încât tehnicile de codare si decodare pentru canalele AWGN să poată fi folosite pentru a aproxima capacitatea. Există două clase principale de astfel de tehnici.

O clasă tratează toate subcanalele, ca aparţinând unui singur canal [14, 15, 16], în timp ce cealaltă clasă partiţionează canalul de bază în sub-canale paralele independente, prin care simbolurile sunt transmise ţinând seama de alocarea optimă a puterii conform principiului water filling [17].

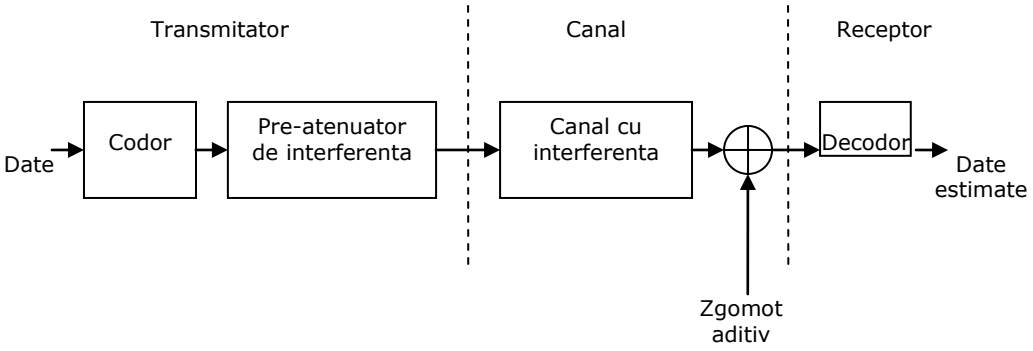


Figura 1.6. Un sistem de comunicare, care pre-anulează interferenţa de canal

Pentru canale cu ISI, un exemplu din prima clasă este precodarea Tomlinson-Harashima, iar un exemplu pentru cea de a doua este discret multi-ton (DMT).

1.3.2 Alocarea puterii fară water-filling

Atunci când transmiţătorul nu are informaţii despre starea canalului, nu se poate aplica principiul water pouring pentru a obţine capacitatea canalului cu interferenţe. În acest caz pentru a obţine o rată maximă de biţi, puterea de transmisie este distribuită uniform în toate subcanalele sistemului.

$$I_{int} = \frac{1}{N}\sum_{k=0}^{N-1} log_2\left(1+\frac{\varepsilon_s \lambda_k}{N_0}\right) \tag{1.20}$$

Această cantitate reprezintă informaţia mutuală a canalului, mai degrabă decât capacitatea canalului, deoarece rata de informaţie nu este optimizată la transmiţător prin intermediul algoritmului water pouring. În acest exemplu, sistemul de comunicaţie descris în figura 1.6 nu mai poate fi folosit, iar interferenţa trebuie să fie tratată la receptor. Receptorul optimal în sensul probabilităţii de eroare foloseşte algoritmii ML (Maximum-Likelihood) sau MAP (Maximum A Posteriori). Receptorul apare ca un singur bloc care efectuează atât detecţia cât şi decodarea, aşa cum se arată în figura 1.7. Deşi optimizarea este comună, complexitatea unui astfel de sistem este de obicei determinată de produsul dintre complexitatea

detectorului optimal pentru sistemul necodificat corespunzător şi decodorul optimal pentru canalul AWGN corespunzător. Astfel, complexitatea unui astfel de sistem este prohibitivă.

O soluţie clasică suboptimală este de a separa problemele de detecţie şi decodare, aşa cum se arată în figura 1.8. Detectorul trebuie să fie proiectat în aşa fel încât grupul format din canal cu interferenţe şi detector să apară ca un canal AWGN. Structura detectorului diferă doar pentru cazul în care datele sunt transmise necodat.

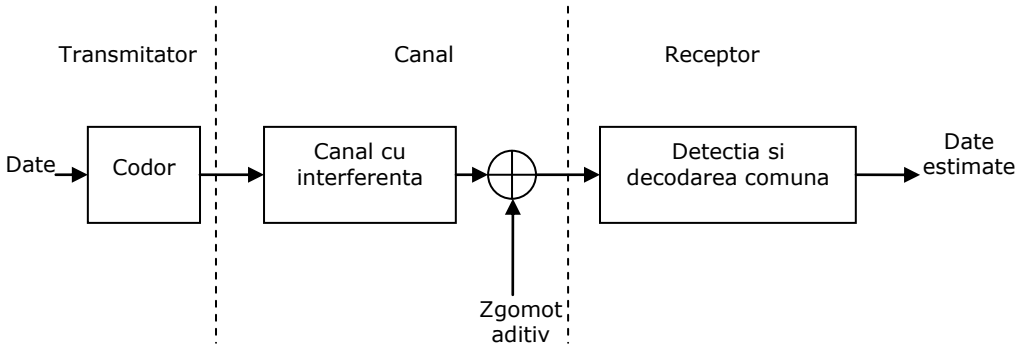


Figura 1.7. Decodarea şi detecţia comună optimală

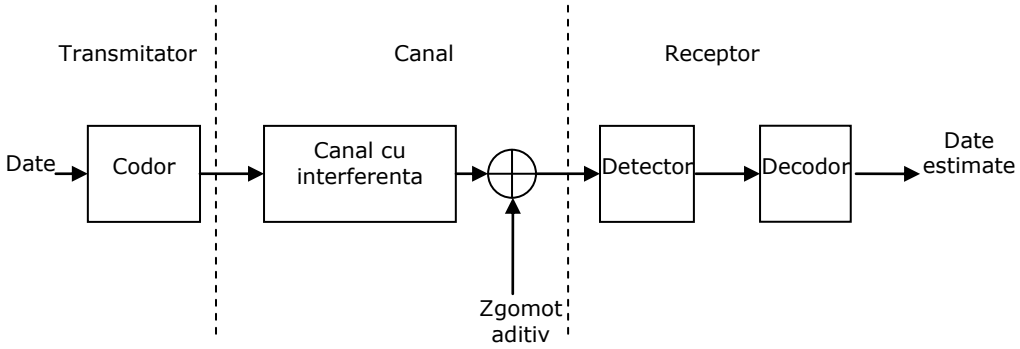


Figura 1.8. Separarea clasică (suboptimală) a detecţiei şi decodării

Cu toate acestea, este mult mai dificil să se realizeze grupul format din canal şi detector ca şi cum ar fi un canal AWGN, decât grupul format din emiţător cu pre-anulator şi canal, deoarece zgomotul de pe canal poate fi amplificat sau consolidat prin detector. Cu toate acestea, presupunând că detectorul este conceput corect, se pot utiliza schemele de codare şi decodare pentru canalele AWGN, astfel încât structura decodorului să fie aceeaşi, atât în sistemele care utilizează canale cu interferenţe, cât şi în cele care utilizează canale fără interferenţe. În acest caz

complexitatea generală este mai degrabă suma complexităţilor componentelor individuale, decât produsul acestora.)

Dintre cele trei scenarii prezentate, în cazul celui din urmă tehnicile de abordare a limitelor teoretice sunt cel mai puţin dezvoltate. Această teză prezintă schemele practice cu complexitate redusă care abordează informaţia mutuală a unui canal cu interferenţă, fără ca la transmiţător să existe informaţii despre canal. Înainte de a prezenta unele detectoare clasice, se vor compara limitele teoretice de transmisie a informaţiei corespunzătoare celor trei scenarii de comunicare amintite.

1.4 Comparaţia dintre ratele maxime de informaţie

Se vor compara în continuare ratele maxime atinse pentru diferitele scenarii de comunicare la aceeaşi SNR recepţionat, aşa cum sunt definite în (1.8). Pentru a putea realiza comparaţia, se va presupune că numărul de coloane din H $\|H\|_F^2 = \sum_{k=1}^{N} \lambda_K$ este normalizat la N, iar energia simbolului ε_S şi varianţa zgomotului N_0 sunt de asemenea fixate. Atunci când puterea de emisie este distribuită în mod egal între subcanale, canalul cu interferenţă nu poate avea o informaţie mutuală mai mare decât cea corespunzătoare canalului AWGN cu o valoare SNR echivalentă. Se poate vedea acest lucru din concavitatea funcţiei log:

$$I_{int} = \frac{1}{N}\sum_{k=1}^{N} log_2\left(1+\frac{\varepsilon_S \lambda_k}{N_0}\right) \le log_2\left(1+\frac{\varepsilon_S}{N_0}\right) = C_{AWGN} \qquad (1.21)$$

În cazul în care se foloseşte normalizarea $\|H\|_F^2$, iar ratele sunt pe două dimensiuni. La valori SNR reduse, putem arăta că $I_{int} = C_{AWGN}$ folosind aproximarea $ln(1+a) \approx a$

$$I_{int} = \frac{1}{N}\sum_{k=1}^{N} log_2\left(1+\frac{\varepsilon_S \lambda_k}{N_0}\right) \approx \frac{1}{N}\sum_{k=1}^{N} \frac{\varepsilon_S \lambda_k}{N_0 \ln 2} \approx \frac{\varepsilon_S}{N_0 \ln 2} \approx log_2\left(1+\frac{\varepsilon_S}{N_0}\right) = C_{AWGN} \qquad (1.22)$$

La valori SNR ridicate, scăderea ratei de la C_{int} la I_{int} datorită neaplicării algoritmului '' water pouring '', este neglijabilă, deoarece alocarea puterii de transmise la subcanale este egală. Expresia " SNR ridicate " este relativă în acest context, deoarece nu numai că puterea de emisie totală trebuie să fie ridicată, dar şi valoarea puterii alocate fiecărui subcanal trebuie să fie de asemenea mare.

Astfel un canal cu mai puţine subcanale necesită o valoare SNR mai mare decât un canal cu mai multe subcanale pentru ca C_{int} şi I_{int} să fie aproximativ egale. Cu toate acestea la valori SNR mici capacitatea C_{int} a unui canal cu interferenţă poate depăşi în mod substanţial atât informaţia mutuală I_{int} fără aplicarea algoritmului '' water pouring '' cât şi capacitatea canalului AWGN C_{AWGN}, chiar dacă s-a făcut normalizarea $\|H\|_F^2$. Rezultă că puterea transmisă este selectiv încărcată pe subcanalele canalului, care pot suporta cele mai mari rate. În figura 1.9

se compară capacitatea canalului AWGN (C_{AWGN}) cu capacitatea canalului cu interferenţe (C_{int}) şi respectiv informaţia mutuală (I_{int}) corespunzătoare canalului cu ISI având răspunsul la impuls de forma:

$$h_i = 0.5\delta_i + 0.707\delta_{i-1} + 0.5\delta_{i-2} \tag{1.23}$$

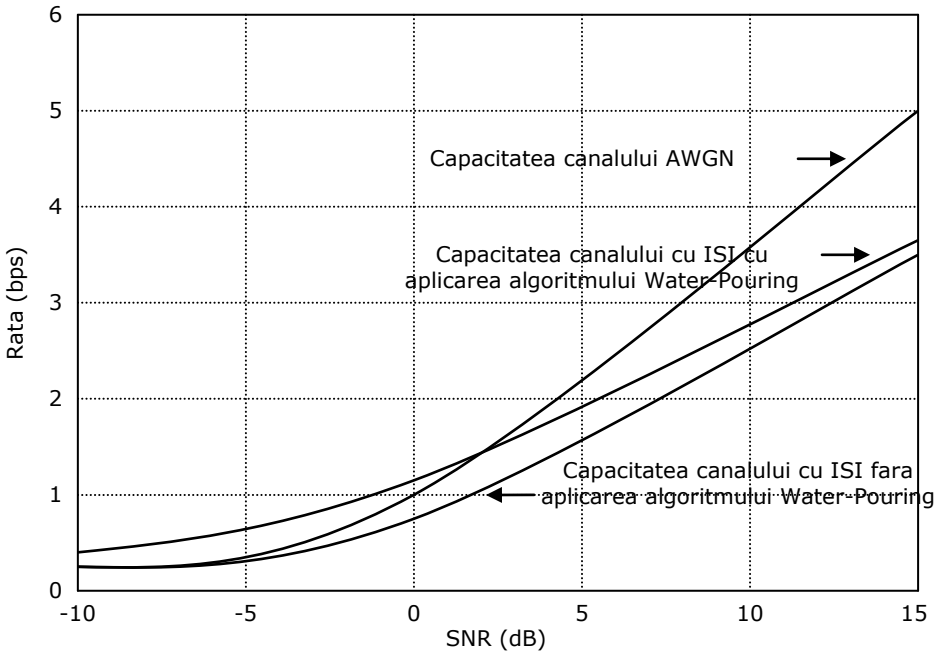


Figura 1.9. Ratele de informaţie pentru canalul cu ISI
având răspunsul la impuls dat de relaţia (1.23)

După cum s-a menţionat mai sus, capacitatea unui canal cu ISI este mai mare decât cea a canalului AWGN pentru valori scăzute ale SNR, deoarece puterea de emisie poate fi încărcată pe frecvenţele favorabile. În cazul în care algoritmul '' water pouring '' nu este aplicat, atunci la valori mici ale SNR informaţia mutuală a canalului cu ISI aleator se apropie de capacitatea canalului AWGN corespunzător. La valori mari ale SNR, capacitatea canalului cu ISI devine mai mică decât capacitatea canalului AWGN, şi, de asemenea, efectul obţinut prin aplicarea algoritmului '' water pouring '' devine neglijabil. Pantele celor trei curbe sunt egale, ceea ce înseamnă că dezavantajul unui canal cu interferenţă constă doar în pierderea ratei fixe.

Capitolul 2
Scheme clasice de detecţie

În orice sistem de recepţie pot fi folosite o serie de detectoare. Primul detector tratat este detectorul de plauzibilitate maxima, ML (Maximum Likelihood) care minimizează probabilitatea detecţiei eronate a unui vector/cuvânt de cod şi care poate fi considerat optimal. Celelalte detectoare ce vor fi prezentate au o performanta care se apropie de cea a ML, dar au o complexitate mai redusa. Toate reprezintă un compromis acurateţe/complexitate, dar toate caută soluţia de eliminare a efectului de interferentă a canalului H, astfel încât canalul să poată fi tratat la recepţie ca un canal AWGN. Pentru cazul particular al sistemelor necodate, problema detecţiei poate fi tratată separat, cu un dispozitiv de decizie simbol-cu-simbol.

Înainte de prezentarea detectoarelor este utilă stabilirea metodei de comparare a performanţei datelor. Deşi sistemele practice folosesc transmisii codate, în cele ce urmează compararea performanţei detectoarelor se va face pentru transmisii necodate, deoarece detectoarele tratează de obicei datele ca fiind necodate [18]; astfel, considerând datele ca fiind necodate, se poate separa problema calităţii detectorului de cea a calităţii decodorului. Fiecare componentă a lui X conţine date alese echiprobabil dintr-un set finit, $X \in X^N$. Probabilitatea erorii pe bit, BER (Bit Error Rate) la un sistem codat, este definită ca probabilitatea ca o componentă a lui $\hat{X}$ să nu fie egală cu componenta corespunzătoare din $X \in X^N$, fiind măsura utilizată aici pentru aprecierea performanţei detectorului. Utilizarea acestei metrici este aleasă din câteva motive:

–i) detectorul ML, care minimizează probabilitatea detecţiei eronate a vectorului/cuvântului recepţionat are, la SNR mare, un BER aproape identic cu cel al detectorului care minimizează rata erorii pe bit [5]. Astfel că, asimptotic, detectorul ML oferă o limită inferioară a BER, mai mică decât a oricărui alt detector.

–ii) analiza BER este echivalentă cu calculul distanţei Hamming dintre x şi $\hat{x}$, care dă o măsură a apropierii dintre vectorul detectat şi cel transmis. Deoarece vectorul ML este, în medie, cel mai apropiat de X, putem aprecia cât de apropiate sunt soluţiile oferite de diferite detectoare fată de soluţia optimală.

–iii) folosirea BER este intuitivă, deoarece detectoarele suboptimale încearcă de obicei să decupleze detecţia simbolurilor codate folosind un dispozitiv de decizie simbol cu simbol de distanţă minimă (numit şi slicer),.

–iv) deseori codurile sunt caracterizate de numărul biţilor eronaţi din cuvântul de cod care pot fi corectaţi. La sistemele care realizează decodarea după detecţie, un BER scăzut după detecţie este preferabil pentru o performanţă globală bună a sistemului.

2.1 Detecţia ML

Deoarece toţi vectorii x sunt echiprobabili, detectorul care minimizează probabilitatea detecţiei eronate a vectorului este detectorul ML:

$$\hat{X} = \underbrace{\arg\max}_{X \in X^N} f\left(R \middle| X\right) \tag{2.1}$$

Deoarece zgomotul este independent de X, necorelat şi gaussian, relaţia (2.1) se simplifică la regula distanţei minime - vezi figura 2.1.

$$\hat{X} = \underbrace{\arg\min}_{X \in X^N} \|R - HX\| \tag{2.2}$$

Astfel că, detectorul calculează cel mai plauzibil vector X, bazat pe cunoaşterea lui R şi H, precum şi a distribuţiei W. Mulţimea tuturor vectorilor necodaţi posibili, X^N, poate fi reprezentată într-un spaţiu euclidian N - dimensional ca puncte ale unei latice ortogonale (deplasate) mărginită dintr-un cub N - dimensional. Mulţimea vectorilor posibili este reprezentată în figura 2.2.a, pentru $N = 2$.

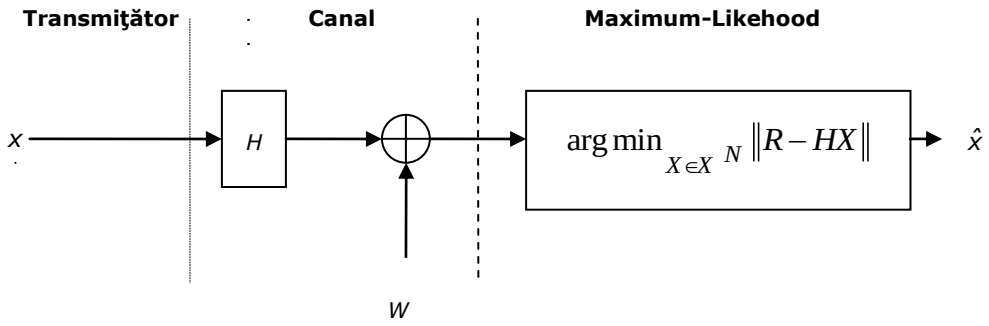
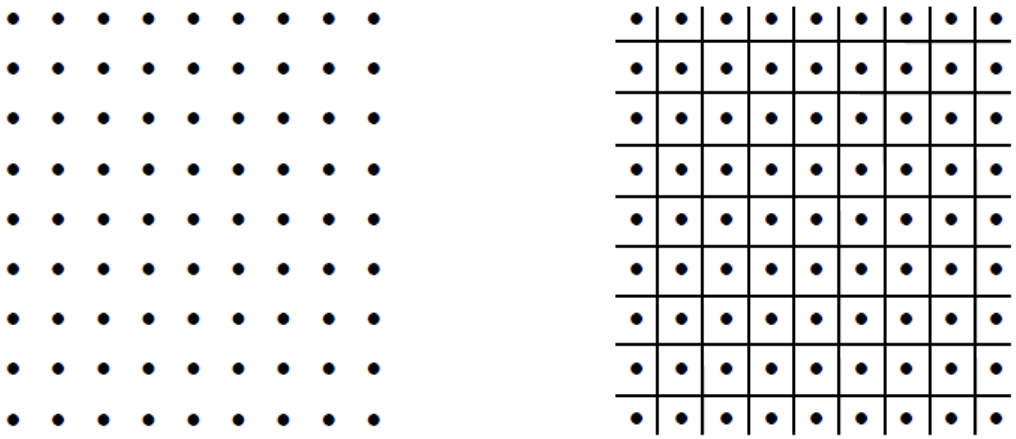


Figura 2.1 Detecţia de plauzibilitate maximă, ML (Maximum Likelihood)



(a) (b)

Figura 2.2 (a)Latice mărginită reprezentând mulţimea vectorilor necodaţi
(b)Regiunile de decizie corespunzătoare pentru canalul AWGN

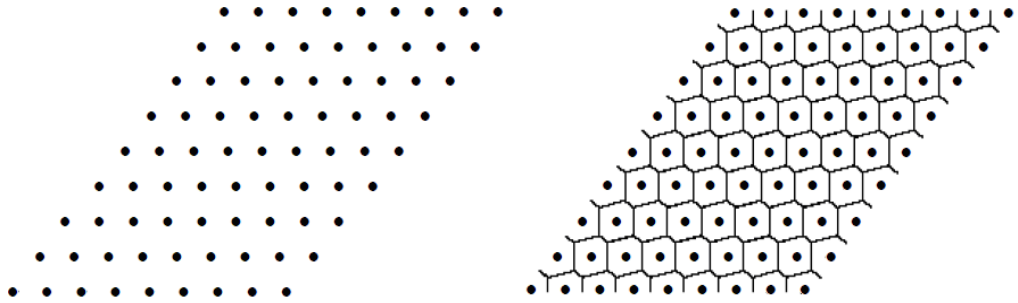
În cazul particular al canalului AWGN, modelul devine $R = X + W$, iar R este versiunea lui X perturbată de zgomot. Regula distanţei minime devine:

$$\hat{X} = \underbrace{arg\,min}_{X \in X \; N} \|R - X\| \tag{2.3}$$

Deoarece fiecare componentă a vectorului necodat X afectează doar componenta corespunzătoare din R, şi deoarece vectorul de zgomot este necorelat, detectorul ML poate fi tratat separat ca o mulţime de optimizări simbol cu simbol, care pot fi rezolvate folosind un dispozitiv de decizie de distanţă minimă (slicer), simbol cu simbol.

$$\hat{x}_i = \underbrace{arg\,min}_{X \in X \; N} \|r_i - x_i\| \quad pentru \quad i = 0,1,\ldots,N-1 \tag{2.4}$$

Regiunile de decizie, corespunzătoare valorilor lui R pentru care trebuie luate decizii la fiecare în parte, sunt prezentate în figura 2.2.b. Capacitatea de a decupla detectorul ML în componente pentru minimizare este marcată de faptul că graniţele oricărei regiuni de decizie formează o grilă ortogonală. Minimizarea fiecăreia din cele N componente ale lui X necesită compararea a $|X|$ diferenţe, deci complexitatea depinde liniar de N.



(a) (b)

Figura 2.3 (a) Latice mărginită reprezentând toţi vectorii posibili Hx pentru un canal cu interferentă. (b) Regiunile de decizie corespunzătoare

În cazul general când există o interferentă liniară, avem $R = HX + W$, iar detectorul ML, (2.2), în general, nu poate fi descompus în N probleme mai mici. Se poate vedea acest lucru ştiind că acţiunea lui **H** asupra mulţimii vectorilor necodaţi $X \in X^N$ este de asociere a punctelor laticei – figura 2.2.a - cu punctele laticei - figura2.3.a - care are generatoarele de-a lungul direcţiilor coloanelor lui **H**. Regiunile de decizie ale lui (2.2), sunt politopuri (figuri geometrice de tip poligon) - figura2.3.b- şi separarea problemelor nu mai este posibilă.

Minimizarea relaţiei (2.2), implica compararea a $|X|^N$ diferenţe, deci complexitatea creşte exponenţial cu N. La canalul ISI, detecţia ML, poate fi făcută folosind decodarea Viterbi [5], cu complexitatea proporţională cu $|X|^L$, L fiind lungimea răspunsului la impuls al canalului. Daca L sau $|X|$ sunt mari, şi detecţia ML este destul de complexă.

Complexitatea detectorului ML este un impediment, deci în practică se caută detectoare mai simple, care aproximează soluţiile relaţiei (2.2).

2.2 Detecţia liniară

Detectoarele liniare – vezi figura 2.4 – preiau vectorul recepţionat R şi îl pre-multiplică cu matricea B^+. Produsul rezultat, $\tilde{X}$, e trimis slicer-ului simbol cu simbol, de distanţă minimă, rezultând $\tilde{x}$. Matricea B poate fi optimizată după criterii diferite, dar cele mai utilizate sunt criteriul cu forţarea zerourilor- zero forcing (ZF) - respectiv MMSE (Minimum Mean Squared Error)de eroare medie pătratică minimă.

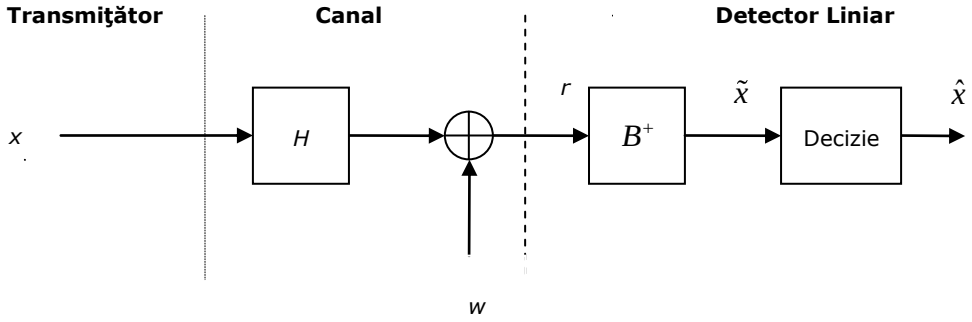


Figura 2.4. Detecţia liniară

Criteriul ZF de anulare a interferenţei, alege B astfel încât să fie complet eliminată interferenţa în $\tilde{X}$, iar la MMSE se alege B astfel încât să fie minimizată varianţa lui $\tilde{X} - X$. Dezavantajul acestor detectoare de complexitate redusă este o rată a erorii pe simbol mai slabă deoarece matricea B^+ creşte puterea componentelor zgomotului din W şi determină corelarea vectorului diferenţă $\tilde{X} - X$. Problema este si mai gravă la criteriul ZF, dar de fapt este prezentă la întreaga clasă de detectoare.

2.2.1 Filtrul adaptat de albire

Estimatorul de date convenţional este format dintr-un banc de filtre adaptate. În cazul zgomotului corelat, estimatorul de date convenţional este extins cu un filtru de pre-albire. Filtrul adaptat de albire, WMF (Whitening Matched Filter) este o combinaţie de filtre de pre-albire şi filtre adaptate urmate de circuite de eşantionare la rata de simbol. Deşi WMF şi MAI tratează ISI ca zgomot, sunt expuse aici, deoarece următoarele pot fi interpretate ca o extensie a WMF, unde zgomotul este decorelat sau prealbit, şi deci semnalul prealbit este livrat filtrelor adaptate ca răspuns al filtrului de pre-albire la secvenţa de date - vezi figura 2.5.

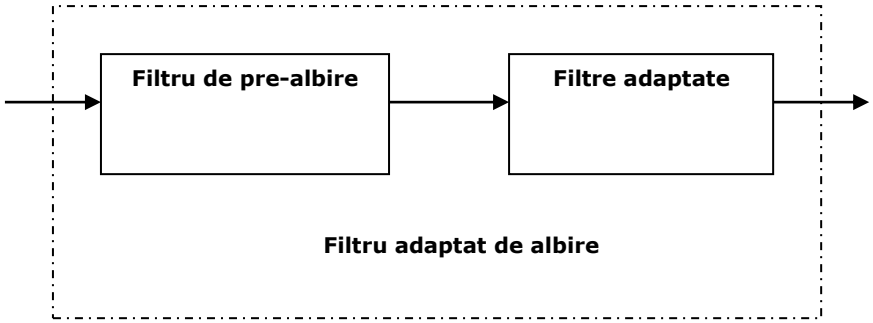


Figura 2. 5 Structura WMF

Albirea zgomotului urmată de filtrarea adaptată se apropie de optim dacă ISI şi componentele MAI sunt neglijabile, care este cazul factorilor mari de împrăştiere şi a secvenţelor ortogonale de tip semnătură, ceea ce implică o eficienţă spectrală redusă.

2.2.2 Egalizorul bloc liniar ZF

Egalizorul bloc liniar ZF (ZF-BLE) conduce la un estimat continuu nedeplasat (fără eroare sistematică). Egalizorul ZF se presupune că elimină total ISI şi MAI, independent de nivelul zgomotului. Egalizorul ZF-BLE este o extensie a WMF. Dacă nu există ISI, ZF-BLE acţionează ca un detector decorelativ – vezi figura 2.6.

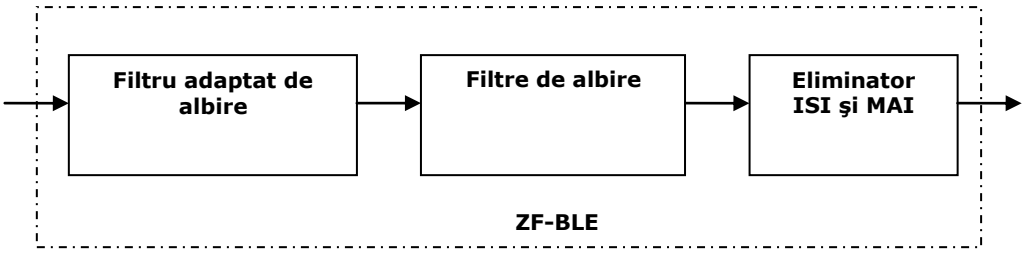


Figura 2.6. Structura ZF-BLE

2.2.3 Egalizorul bloc liniar MMSE

Egalizorul bloc liniar MMSE (MMSE-BLE) minimizează eroarea pătratică şi duce la un estimat continuu, fiind o extensie a ZF-BLE cu un estimator Wiener, care împiedică reducerea performantei ZF-BLE, cauzată de faptul că deciziile nu iau în considerare corelaţia zgomotului. Estimatorul Wiener preia semnalul de la ieşirea ZF-BLE şi produce estimatul MMSE. Cu estimatorul Wiener se decorelează ISI si MAI faţă de zgomot –vezi figura 2.7.

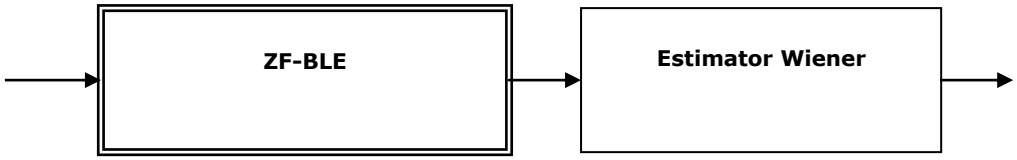


Figura 2.7. Structura MMSE -BLE

Pentru $\sigma^2 \to 0$ MMSE -BLE se apropie de ZF-BLE , iar pentru $\sigma^2 \to \infty$ MMSE -BLE se apropie de WMF.

2.3 Detectorul cu reacţie decizională

Detectorul cu reacţie decizională se bazează pe un detector liniar combinat cu o buclă de reacţie neliniară - figura 2.8. Ca şi mai înainte, vectorul recepţionat R este pre-multiplicat cu B^+ , dar în loc de o decizie simbol cu simbol a slicerului de distanţă minimă pentru întregul vector, deciziile se iau secvenţial, pas cu pas. La început, prima componentă a lui B^+R , notata cu $\tilde{x}_1$ este procesată de slicer care livrează simbolul detectat $\hat{x}_1$. Presupunem că $\hat{x}_1$ este egal cu x_1 , bucla de reacţie este folosită pentru extragerea interferentei cauzata de x_1 din componentele lui B^+R rămase. A doua componentă x_2 a vectorului B^+R cu interferenţa extrasă, este apoi procesată de slicer care generează $\hat{x}_2$. Daca $\hat{x}_2$ este decizia corectă, interferenţa cauzată de x_2 este extrasă din componentele lui B^+R rămase şi procesul continuă până se iau deciziile pentru toate componentele.

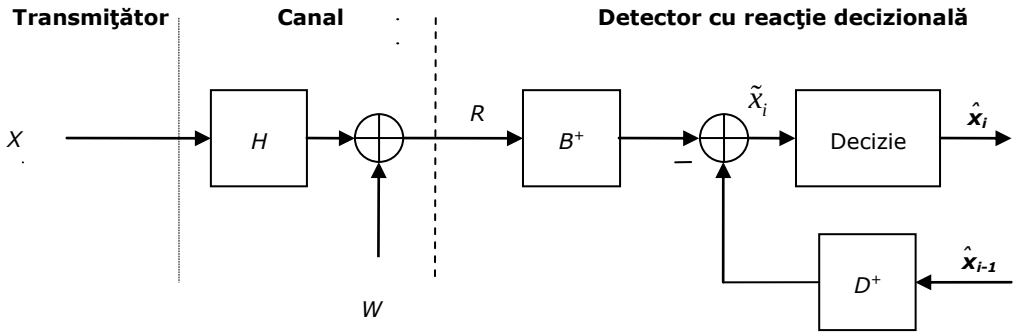


Figura 2.8. Detectorul cu reacţie decizională

Detectoarele liniare ZF şi MMSE sunt foarte răspândite. Deşi detectoarele cu reacţie decizională au performanţe mai bune, au şi probleme. Prima problemă este amplificarea zgomotului, deşi nu e atât de severă ca la detecţia liniară. A doua este că deciziile se iau secvenţial în slicer, deci pot îmbunătăţi doar deciziile viitoare nu şi cele trecute. A treia este propagarea unei decizii eronate, la viitoarele decizii. Acest lucru poate influenţa negativ matricele B şi D folosite în bucla de reacţie, dar acestea sunt optimizate în ipoteza că eroarea nu se propagă. A patra problemă este că detectorul cu reacţie decizională fiind secvenţial, este incompatibil cu canalele ISI împreună cu codarea canalului. Aceste detectoare cu reacţie sunt deci adecvate doar pentru sisteme necodate.

2.3.1 Egalizorul bloc liniar ZF cu reacţie (ZF – BDFE)

În acest tip de egalizor se introduce secvenţa recepţionată modificata [19] , iar deciziile se iau conform unei formule recursive. La transmisii codate ar putea fi generate intrări soft decodorului, folosind estimări cuantizate în bucla de reacţie şi informaţii evaluate continuu înaintea decuantizării pentru decodare, eventual cu anumite informaţii de stare a canalului. Se extrag din noul estimat toate deciziile anterioare, iar dacă deciziile anterioare se apropie de decizia corectă estimarea este convergentă - vezi figura 2.9, de unde se vede că ZF – BDFE este echivalent cu detectorul cu anulare a zgomotului. Dacă estimatele evaluate continuu u_k obţinute prin omiterea pragului de detecţie, sunt readuse prin reacţie la intrare, atunci ZF – BDFE devine ZF – BLE.

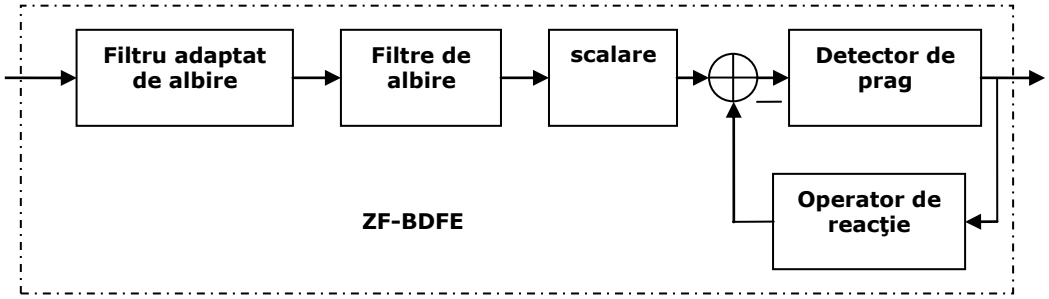


Figura 2.9. Structura egalizorului ZF–BDFE

2.3.2 Egalizorul MMSE bloc cu reacţie decizionala (MMSE-BDFE)

Structura este aceeaşi cu cea a egalizorului ZF–BDFE din figura 2.9, dacă se modifică matricea filtrului de albire. Se foloseşte o descompunere Cholesky, iar extragerea vechilor decizii şi deciziile se fac la fel ca în §2.3.1.

2.4 Detecţia multinivel

Pentru a simetriza problema anulării doar a simbolurilor viitoare în detectoarele cu reacţie decizională, detectorul multinivel/multistage procesează vectorul recepţionat în blocuri de iteraţii. Detectorul multinivel poate fi privit ca un procesor paralel, unde procesorul secvenţial este dat de detectoarele cu reacţie decizională –vezi figura 2.10. În prima iteraţie vectorul recepţionat R este pre-multiplicat cu matricea filtrului adaptat H^+ pentru a produce $\tilde{X}^{(1)}$, care este apoi transmisă slicerului pentru a genera $\hat{X}^{(1)}$ o tentativă cu primul set de decizii asupra tuturor simbolurilor. În a doua iteraţie vectorul recepţionat R este din nou pre-multiplicat cu matricea filtrului adaptat H^+, dar înainte ca rezultatul să fie trimis slicerului , este creată o copie identică a interferentei şi extrasă , presupunând că $\hat{X}^{(1)}$ este setul de decizii corect. Slicerul ia apoi vectorul $\tilde{X}^{(2)}$ şi generează al doilea set de tentativă de decizii $\hat{X}^{(2)}$. Viitoarele tentative $\tilde{X}^{(l)}$ sunt generate la fel, folosind tentativele precedente $\tilde{X}^{(l-1)}$ pentru a extrage interferenţa. După un număr suficient de iteraţii este considerată ca decizie finală, ultima tentativă. Cele două matrice sunt refăcute în fiecare iteraţie, şi optimizate astfel încât să maximizeze SINR (Signal to Interference +noise Ratio) de la intrarea slicerului pentru o tentativă de decizie corectă. În general, categoria detectoarelor multinivel are structura din figura 2.10, doar că matricele pereche se schimbă la fiecare iteraţie. Problema principală la aceste detectoare este că de obicei nu converg spre optimul absolut şi pot apărea cicluri sau divergente. Cauza este ipoteza că optimizarea matricelor se face considerând decizii corecte în fiecare iteraţie, ceea ce nu este întotdeauna adevărat şi poate apărea propagarea erorii.

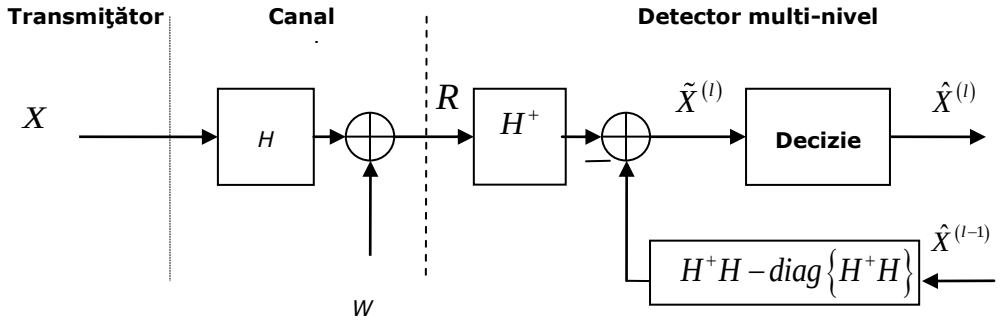


Figura 2.10. Detectorul multinivel

2.4.1 Supresorul interferenţei

Pentru reducerea complexităţii turbo egalizării, detectorul MAP poate fi înlocuit cu succes de un supresor al interferenţei, IC (Interference Canceller). Noul receptor poate evita aproape complet ISI din canale Rayleigh variabile sau nu în timp, pentru modulaţii cu eficienţă spectrală mare. Supresorul interferenţei –vezi figura 2.11- conţine două filtre transversale la intrarea cărora se aduc eşantioanele recepţionate şi datele, estimate în iteraţia precedentă.

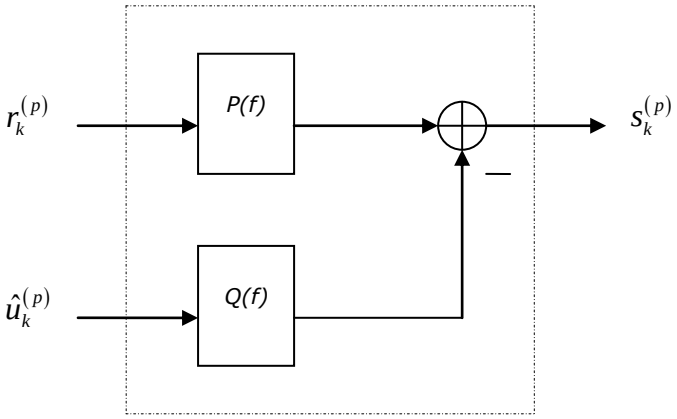


Figura 2.11. Supresorul interferenţei

La fiecare iteraţie supresorul este actualizat conform unui criteriu: IC optim şi IC adaptiv.

2.4.2 Eroarea medie pătratică, MSE

Criteriul erorii medii pătratice, MSE (Mean Square Error) este definit ca: $MSE = E\left\{\left|s_k - u_k\right|^2\right\}$.

Pentru a determina **IC optim**, este necesar ca să fie cunoscute simbolurile $\left(\hat{d}_k = d_k\right)$. Se considera $\left(q_0 = 0\right)$ zero câştigul central al filtrului $Q(f)$, şi atunci filtrul IC optim are funcţia de transfer:

$$P(f)_{optimal} = \beta \frac{H^*(f)}{hh_0} = \sum_{l=0}^{L} p_l \, exp\left(-j2\pi flT\right) \tag{2.5}$$

$$Q(f)_{optimal} = \beta \left(\frac{\left|H(f)\right|^2}{hh_0} - 1 \right) = \sum_{l=0}^{L} q_l \, exp\left(-j2\pi flT\right) \tag{2.6}$$

unde $H(f)$ este funcţia de transfer a canalului, hh_0 funcţia de autocorelaţie a canalului, β este un coeficient de ponderare, egal cu:

$$\beta = \frac{\sigma_u^2 hh_0}{\left(\sigma_u^2 hh_0 + \sigma_w^2\right)} \tag{2.7}$$

iar p_l şi q_l sunt coeficienţii filtrelor $P(f)$ şi $Q(f)$. Astfel ieşirea IC nu are ISI şi este:

$$s_k = \beta\left(u_k + \frac{1}{hh_0}\sum_{l=0}^{L} h_l^* w_{k+l}\right) \tag{2.8}$$

Iar ieşirea MSE este dată de:

$$MSE_{optimal} = \frac{\sigma_u^2 \sigma_w^2}{\left(\sigma_u^2 hh_0 + \sigma_w^2\right)} = 1 - \beta \tag{2.9}$$

Iar SNR la ieşirea IC este:

$$SNR_{optimal} = \frac{\sigma_u^2 hh_0}{\sigma_w^2} \tag{2.10}$$

2.4.3 Supresorul adaptiv al interferenţei

Algoritmii adaptivi, ca LMS (Least Mean Square) sau RLS (Recursive Least Square) pot fi folosiţi pentru actualizarea parametrilor egalizorului. Ei minimizează MSE. Ei pretind o secvenţă de date iniţial, sau periodic, (secvenţa de antrenare), cunoscuta de receptor, pentru a asigura convergenta algoritmului. După asigurarea convergenţei, algoritmii sunt comandaţi de decizii şi minimizează MSE estimat, dat de $MSE = E\left\{\left|s_k - u_k\right|^2\right\}$, unde u_k este decizia tentativă de la ieşirea egalizorului – vezi figura 2.11. Secvenţa Y_k de la ieşirea canalului şi secvenţa simbolurilor estimate $\hat{U}_k$ furnizată de decodorul de canal în iteraţia precedentă, alimentează egalizorul IC. Ieşirea IC este:

$$s_k = P_k^T R_k - Q_k^T \hat{U}_k \tag{2.11}$$

Unde $R_k = \left[r_{k+L_1} \cdots r_k \cdots r_{k-L_1}\right]$ şi $\hat{U}_k = \left[\hat{u}_{k+L_2} \cdots \hat{u}_k \cdots \hat{u}_{k-L_2}\right]$ sunt vectorii eşantion recepţionaţi şi vectorii valorilor medii estimate.

Apoi $P_k = \left[\hat{p}_{-l_1}(k)\cdots\hat{p}_0(k)\cdots\hat{p}_{l_1}(k)\right]$, iar $Q_k = \left[\hat{q}_{-l_2}(k)\cdots\hat{q}_0(k)\cdots\hat{p}_{l_2}(k)\right]$ sunt vectorii parametrilor egalizorului corespunzând filtrelor $P(f)$ şi $Q(f)$. L_1 şi L_2 sunt valori adecvate mai mari sau egale cu L.
Pentru canalul invariant în timp, este folosit algoritmul LMS pentru toate iteraţiile $(p \geq 1)$ şi sunt iniţializate printr-o secvenţă de antrenare la începutul transmisiei.

$$P_{k+1} = P_k - \mu R_k^* \left(s_k - \hat{u}_k\right) \tag{2.12}$$

$$Q_{k+1} = Q_k - \mu \hat{U}_k^* \left(s_k - \hat{u}_k\right) \tag{2.13}$$

Pentru canalul invariabil în timp, este folosit algoritmul RLS pentru toate iteraţiile $(p \geq 1)$ şi date de iniţializare printr-o secvenţă de antrenare periodică. După prima iteraţie $(p = 1)$ simbolurile estimate nu sunt cunoscute , iar IC este doar un filtru transversal. Pentru alte iteraţii $(p > 1)$ din relaţiile (2.5) şi (2.6), coeficienţii egalizorului p_l şi q_l pot fi calculaţi cu:

$$p_l = \frac{\beta h_{l-1}^*}{hh_0} \tag{2.14}$$

$$q_l = \frac{\beta hh_l}{hh_0} \cdot l \neq 0, \quad q_0 = 0 \tag{2.15}$$

Avantajul e numărul redus de prize (taps) de ajustat pentru a permite algoritmului să urmărească fluctuaţiile rapide ale canalului, dar la prima iteratie IC este suboptimal. În plus trebuie integrat în egalizor un circuit cu calare pe fază PLL (Phase Loop Locked).

Capitolul 3
Decodare MAP utilizând algoritmul BCJR

În acest paragraf se descrie procesul de decodare folosind algoritmul BCJR. Structura unui decodor turbo cu concatenare paralelă arată ca în figura 3.1 .

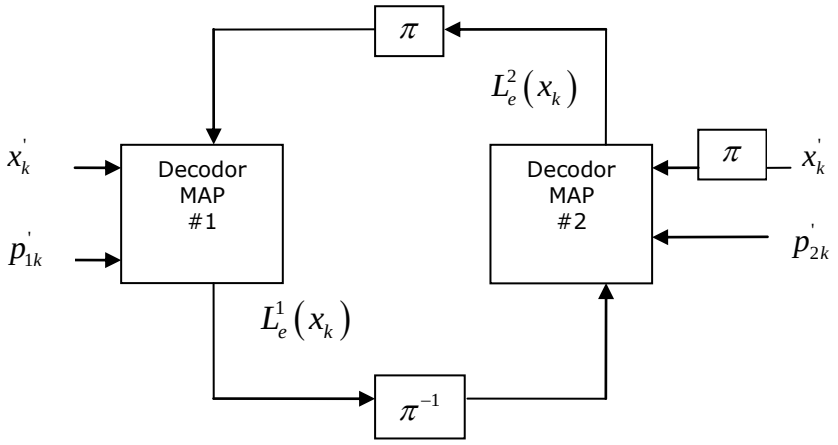


Figura 3.1: Decodorul turbo

Acesta se compune dintr-o pereche de decodoare care lucrează împreună, în scopul de a rafina şi a îmbunătăţi estimarea biţilor originali. Decodorul se bazează pe algoritmul BCJR, de asemenea, numit şi algoritm MAP (Maximum aposteriori probability). Decodarea se bazează pe decizie soft a informaţiei.

După iniţializare decizia soft provenită de la un decodor notată cu L_e şi numită informaţie extrinsecă este utilizată pentru a iniţializa celălalt decodor. Informaţia decodată este din nou adusă la intrarea primului decodor . Aceste iteraţii se repetă până când decizia soft converge către un set stabil de valori. Pentru a calcula valorile estimate ale mesajului se foloseşte ultima informaţie extrinsecă emisă de primul decodor. În figura de mai sus s-a notat cu π^{-1} blocul de întreţesere (interleaver) , iar cu π blocul de de-întreţesere (de-interleaver).

Algoritmul BCJR este un algoritm pentru decodarea MAP a codurilor corectoare de erori bazate pe diagrama de trellis (în principal coduri convoluţionale). Algoritmul este numit după inventatorii săi: Bahl, Cocke, Jelinek şi Raviv. În această secţiune se va descrie algoritmul BCJR folosind [20] şi [21].

Pentru început se vor introduce câteva notaţii :

- Elementele mesajului transmis, care depind de ieşirea codorului $i \in \{1,2\}$, sunt $y_{k,i} = \{x_k, p_k^i\}$, unde x_k este codul sistematic, iar p_k^1 şi p_k^2 reprezintă secvenţele de verificare a parităţii provenite de la codorul RSC 1, respectiv de la codorul RSC 2

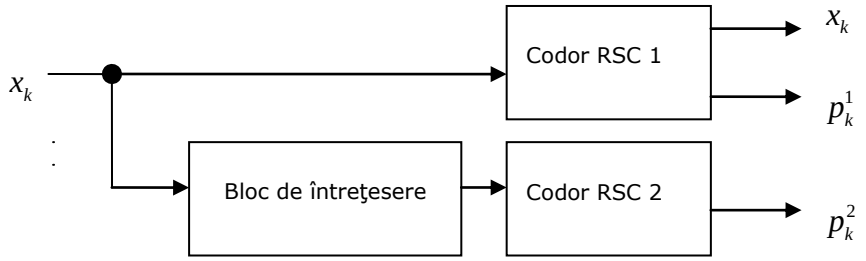


Figura 3.2 Codorul turbo

- Elementele codului recepţionat $y'_{k,i} = \left\{x'_k, p'_{i,k}\right\}$ $i \in \{1,2\}$ şi vectorul codului recepţionat depind de ieşirile codoarelor $Y' = y'_{1,N} = \left\{y'_1, y'_2 \ldots, y'_N\right\}$ unde N este lungimea mesajului.
- Informaţia extrinsecă $Le_{i,j}$ de la decodorul i la decodorul j unde $(i,j) \in \{1,2\}^2$
- Informaţia apriori $L_{a,k}$ la decodorul $k \in \{1,2\}$
- Decizia soft MAP este :

$$L_{map}(x_k) = log\left\{\frac{Pr\left(x_k = +1|Y'\right)}{Pr\left(x_k = -1|Y'\right)}\right\} \tag{3.1}$$

Structura codorului RSC utilizat este de forma:

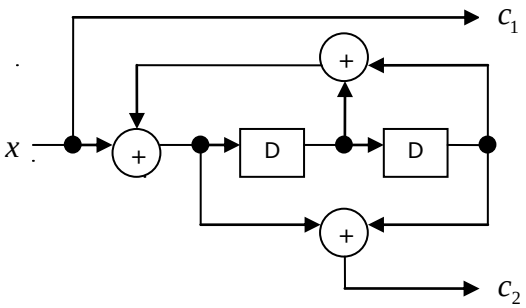


Figura 3.3: Codor convoluţional (G= [111 ; 101])

Diagrama de stare corespunzătoare acestui codor este prezentată în figura 3.4:

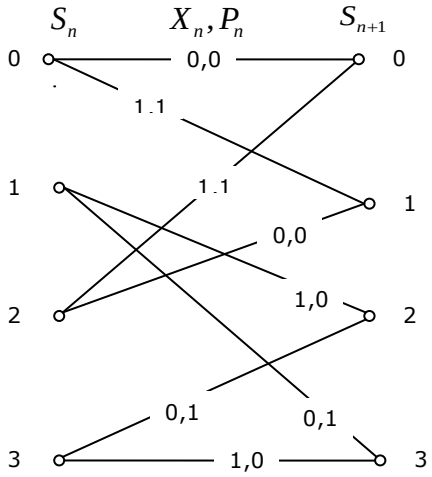


Figura 3.4: Diagrama de stare a codorului

Construcţia diagramei de trellis permite descrierea comportamentului codorului turbo si este un element cheie în procesul de decodare.

Pentru codorul turbo, diagrama de trellis este finalizată atunci când, după secvenţa de intrare, sunt inseraţi m = K-1 biţi suplimentari, unde K este numărul de coloane din matricea generatoare G. Aceşti biţi suplimentari aduc codorul la starea iniţială (terminarea trellisului).

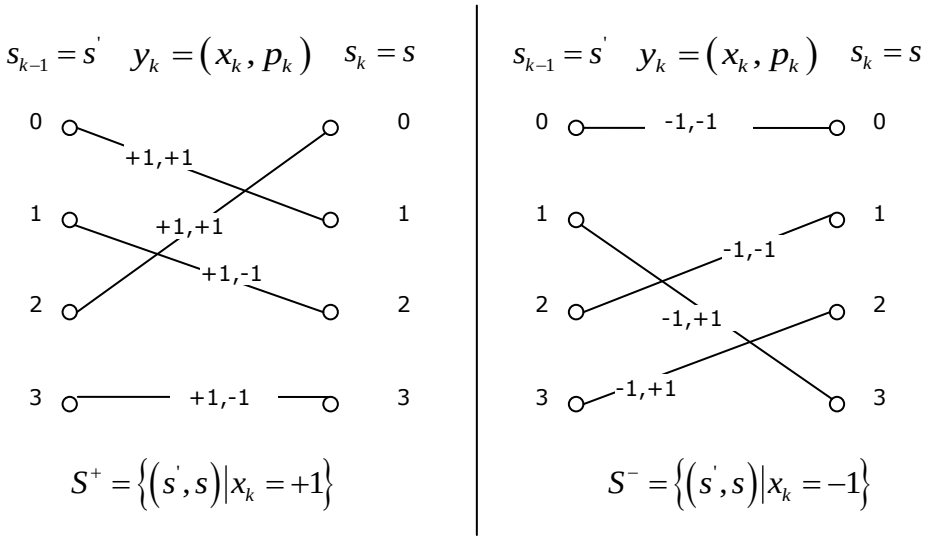


Figura 3.5 Tranziţia stărilor pentru matricea generatoare G = [111; 101] utilizată pentru calcularea coeficienţilor γ, unde S^+ corespunde tranziţiei (s', s) cu $x_k = +1$ în diagrama de trellis

Notaţiile introduse în figurile 3.4 si 3.5 sunt :

- $s_k \in S$ este starea codorului la momentul de timp k,
- S^+ este setul de perechi $\left(s^{'}, s\right)$ care corespund tranziţiei $\left(s_{k-1} = s^{'}\right) \rightarrow \left(s_k = s\right)$ determinat de intrarea de date $x_k = +1$.

La fel, S^- este definit pentru date de intrare $x_k = -1$. Prin utilizarea teoremei lui Baye în realaţia (3.1) , atât la numărător cât şi la numitor se poate simplifica această expresie, eliminând $P\left(y^{'}\right)$. În acest caz se obţine expresia :

$$\left\{\frac{Pr\left(x_k = +1 | y^{'}\right)}{Pr\left(x_k = -1 | y^{'}\right)}\right\} = \frac{\sum_{\left(s^{'},s\right)\in S^+} Pr\left(s_{k-1} = s^{'}, s_k = s, y^{'}\right)}{\sum_{\left(s^{'},s\right)\in S^-} Pr\left(s_{k-1} = s^{'}, s_k = s, y^{'}\right)} \tag{3.2}$$

Dezvoltând această expresie prin introducerea altor funcţii cum ar fi (γ_k, α_k şi β_k) se obţine :

$$Pr\left(s_{k-1} = s^{'}, s_k = s, y^{'}_{1,N}\right) = Pr\left(s_{k-1} = s^{'}, y^{'}_{1,k-1}\right) \cdot$$

$$Pr\left(s_k = s, y^{'}_k | s_{k-1} = s^{'}\right) \cdot Pr\left(y^{'}_{k+1,N} | s_k = s\right) = \alpha_{k-1}\left(s^{'}\right) \cdot \gamma_k\left(s^{'}, s\right) \cdot \beta_k\left(s\right) \tag{3.3}$$

Sensurile acestor funcţii, diferitele etape şi calculele utilizate pentru a obţine aceste expresii sunt prezentate în lucrările [20] [21]. Prin urmare, funcţia $\alpha_k\left(s^{'}\right)$ este probabilitatea de a ajunge la o ramură într-o stare particulară cu secvenţa zgomotoasă $y^{'}_{1,k} = \left\{y^{'}_1, y^{'}_2 \ldots, y^{'}_k\right\}$ care duce la acea stare. Utilizând recursivitatea înainte se poate exprima această funcţie în termenii $\gamma_k\left(s^{'}, s\right)$:

$$\alpha_k\left(s\right) = Pr\left(s_k = s, y^{'}_{1,k}\right) = \sum_{s^{'}} Pr\left(s_{k-1} = s^{'}, y^{'}_{1,k-1}\right) \cdot Pr\left(s_k = s, y^{'}_k | s_{k-1} = s^{'}\right)$$

$$= \sum_{s^{'}} \alpha_{k-1}\left(s^{'}\right) \cdot \gamma_k\left(s^{'}, s\right) \tag{3.4}$$

Prin normalizarea acestei funcţii, se obţine:

$$\alpha_k\left(s\right) = \frac{\sum_{s^{'}} \alpha_{k-1}\left(s^{'}\right) \cdot \gamma_k\left(s^{'}, s\right)}{\sum_s \sum_{s^{'}} \alpha_{k-1}\left(s^{'}\right) \cdot \gamma_k\left(s^{'}, s\right)} \tag{3.5}$$

Funcţia $\beta_k\left(s\right)$ corespunde probabilităţii de a exista o ramură prin intermediul unei anumite stări particulare s şi secvenţa cu zgomot $y^{'}_{k+1,N} = y^{'}_{k+1}, y^{'}_{k+2} \ldots, y^{'}_N$ care termină trellisul.

Utilizând recursivitatea înapoi, funcţia $\beta_k(s)$ se poate exprima cu ajutorul termenilor $\gamma_k(s',s)$.

$$\beta_k(s') = Pr\left(y'_{k+1,N}|s_k = s\right) = \sum_s Pr\left(s_{k+1} = s, y'_{k-1}|s_k = s'\right) \cdot Pr\left(y'_{k+2,N}|s_k = s'\right)$$

$$= \sum_{s'} \gamma_{k+1}(s',s) \cdot \beta_{k+1}(s) \tag{3.6}$$

După normalizare acest coeficient devine :

$$\beta_k(s) = \frac{\sum_{s'} \gamma_{k+1}(s',s) \cdot \beta_{k+1}(s)}{\sum_s \sum_{s'} \gamma_{k+1}(s',s) \cdot \beta_{k+1}(s')} \tag{3.7}$$

Funcţia $\gamma_k(s',s)$ reprezintă probabilitatea de a transmite semnalul cu bitul de paritate y'_k cunoscând starea anterioară. Această funcţie poate fi aproximată ca :

$$exp\left\{\frac{1}{2}\left(x_k L_a(x_k) + x_k L_c x'_k + p_k L_c p'_k\right)\right\} \tag{3.8}$$

Unde L_a este informaţia a-priori emisă de la decodorul anterior, $L_c = \frac{2}{\sigma^2}$ depinde de raportul semnal zgomot (SNR) utilizat. Utilizând de asemenea matricea trellis se exprimă x_k şi p_k pentru o anumită tranziţie de stare. În final x'_k este mesajul cu zgomot, iar p'_k este bitul de paritate care depinde de decodor.

3.1 Implementarea algoritmului BCJR

În această secţiune se va descrie un algoritm de decodare implementat în MATLAB, prezentat şi în [22]. Programul utilizat în [22], a fost parţial modificat de către autorul acestei lucrări. Această modificare s-a realizat pentru a se putea face o comparare între ratele de eroare de bit pentru un sistem care transmite necodat şi un sistem care utilizează codarea turbo şi respectiv decodarea turbo, ambele utilizând modulaţia BPSK [91]. Totodată în urma cestei modificări se poate vizualiza evoluţia BER în funcţie de raportul semnal/zgomot pentru fiecare iteraţie în parte.

==========Iniţializare==========

Decodorul 1

- $\alpha_0^1(s) = \delta_{s,0}$
- $\beta_N^1(s) = \delta_{s,0}$
- $Le_{2,1}^1(x_K) = 0$ pentru k=1,2…..,N

Nu s-a transmis de la D2-> D1, deoarece a fost parcursă doar prima iteraţie.

Decodorul 2

• $\alpha_0^2(s) = \delta_{s,0}$

• $\beta_N^1(s) = \alpha_N^2(s)$ pentru toate s (stabilite după calculul $alpha_N^2(s)$ în prima iteraţie)

• $Le_{2,1}^1(x_K)$ pentru k=1,2…..,N

După prima iteraţie se obţine informaţia extrinsecă de la D1 şi se foloseşte această valoare întreţesută ca informaţie apriori L_a^2 pentru decodorul D2.

==========

========== iteraţia n==========

Decodorul 1

for k=1 :N

•se obţine $y_k' = \left(x_k' p_{1,k}'\right)$

•se calculează prima dată $\gamma_k\left(s', s\right)$ din relaţia (3.8) pentru toate tranziţiile admisibile de stare, unde informaţia apriori este $L_a^1 = \pi^{-1}\left(Le_{2,1}\right)$, informaţia extrinsecă dezântreţesută de la decodorul 2.

• se calculează α_k^1 pentru toate s utilizând relaţia (3.5)

end

for $k = N : -1 : 2$

• se calculează $\beta_{k-1}^1(s)$ pentru toate s utilizând relaţia (3.7)

end

for $k = 1 : N$

• se calculează

$$Le_{1,2}(x_k) = log\left\{\frac{\sum_{s\in S^+}\alpha_{k-1}^1\left(s'\right)\cdot\gamma_k\left(s', s\right)\cdot\beta_k^1(s)}{\sum_{s\in S^-}\alpha_{k-1}^1\left(s'\right)\cdot\gamma_k\left(s', s\right)\cdot\beta_k^1(s)}\right\}$$ utilizând funcţia Trellis din figura 3.5 pentru sumele S^+ şi S^-

end

Decodorul 2

for k=1 :N

• se calculează $y_k' = \left(\pi\left(x_k'\right) p_{2,k}'\right)$

• se calculează $\gamma_k\left(s^{'},s\right)$ din relaţia (3.8) pentru toate tranziţiile admisibile de stare unde informaţia apriori este $L_a^2=\pi\left(Le_{1,2}\right)$, informaţia extrinsecă întreţesută de la decodorul 1, devine informaţie a-priori.

• se calculează α_k^2 pentru toate s utilizând relaţia (3.5)

end

for $k=N:-1:2$

• se calculează $\beta_{k-1}^2(s)$ pentru toate s utilizând relaţia (3.7)

end

for (pentru) $k=1:N$

• se calculează

$$Le_{2,1}(x_k)=log\left\{\frac{\sum_{s\in S^+}\alpha_{k-1}^2\left(s^{'}\right)\cdot\gamma_k\left(s^{'},s\right)\cdot\beta_k^2(s)}{\sum_{s\in S^-}\alpha_{k-1}^2\left(s^{'}\right)\cdot\gamma_k\left(s^{'},s\right)\cdot\beta_k^2(s)}\right\}$$

end

==========

========După ultima iteraţie ========

$k=1:N$

• se calculează L_{map1} emise de decodorul 1

$$L_{map}(x_k)=L_a(x_k)+L_c\left(x_k^{'}\right)+Le_{1,2}$$

Folosind valorile obţinute la ultima iteraţie

• dacă $L_{map}(x_k)>0$

decide $x_k=+1$

altfel

$x_k=-1$

End

3.2 Rezultate obţinute

În figura 3.6 s-a reprezentat BER în funcţie de SNR pentru o secvenţă de date transmisă necodat. S-a generat o secvenţă de biţi care a fost modulată BPSK utilizând diferite valori pentru SNR. Pentru fiecare valoare a SNR s-a calculat BER cu relaţia (3.9) şi s-a reprezentat grafic (în figura 3.6 curba roşie)

$$BER\_sim = \frac{număr\ de\ erori}{număr\ de\ biţi\ transmişi} \tag{3.9}$$

Aceste valori "BER_sim" sunt comparate cu valorile "BER_teoretic" , acestea din urmă fiind calculate pentru modulaţia BPSK cu relaţia (3.10) şi apoi reprezentate grafic. (în figura 3.6 curba albastră)

$$BER\_teoretic = \frac{1}{2} erfc\left(\sqrt{SNR}\right) \qquad \text{.....(3.10)}$$

unde

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_{x}^{\infty} e^{-x} dx \tag{3.11}$$

În figura 3.6 se observă că cele două curbe aproape se suprapun. Această comparaţie s-a făcut pentru a vedea daca se poate utiliza relaţia (3.10) în programul care face turbo codare/ decodare pentru a trasa caracteristica BER în funcţie de SNR pentru secvenţa necodată (în figurile 3.11 şi 3.12 curba albastra). Astfel s-a încercat să se facă o comparaţie (în figurile 3.11 şi 3.12) între valoarea BER pentru secvenţa codată utilizând un anumit număr de iteraţii şi valoarea BER pentru aceeaşi secvenţă, dar necodată.

Aceasta ar fi echivalentă cu transmiterea simultană, pentru aceleaşi valori SNR, atât a secvenţei de biţi necodată cât şi a secvenţei codată cu un turbocodor, ambele modulate BPSK.

La receptor, pentru secvenţa codată, BER-ul se calculează după decodare (utilizând un anumit număr de iteraţii) cu relaţia (3.9), iar pentru secvenţa necodată cu relaţia (3.10).

În figurile 3.7, 3.8, 3.9 şi 3.10 sunt prezentate rezultatele pe care le-am obţinut în urma simulării ratei erorii de bit (BER) în funcţie de numărul de iteraţii pentru diferite valori ale lungimii secvenţei de biţi transmise şi a raportului semnal zgomot (SNR). Din aceste patru figuri se observă că BER-ul scade odată cu creşterea atât a numărului de iteraţii cât şi a numărului de biţi din mesajul transmis. La această concluzie se ajunge din rezultatele pe care le-am obţinut, urmărind rata erorii de bit în funcţie de raportul semnal zgomot (BER/SNR), acestea fiind prezentate în figurile 3.11 şi 3.12

Totodată în figurile 3.11 şi 3.12 se observă că după prima iteraţie, valoarea BER este mai mare pentru secvenţa codată decât pentru secvenţa necodată. În cazul de faţă se impune ca în procesul de decodare să existe minim două iteraţii pentru a obţine o performanţă mai bună faţă de cazul transmisiei necodate.

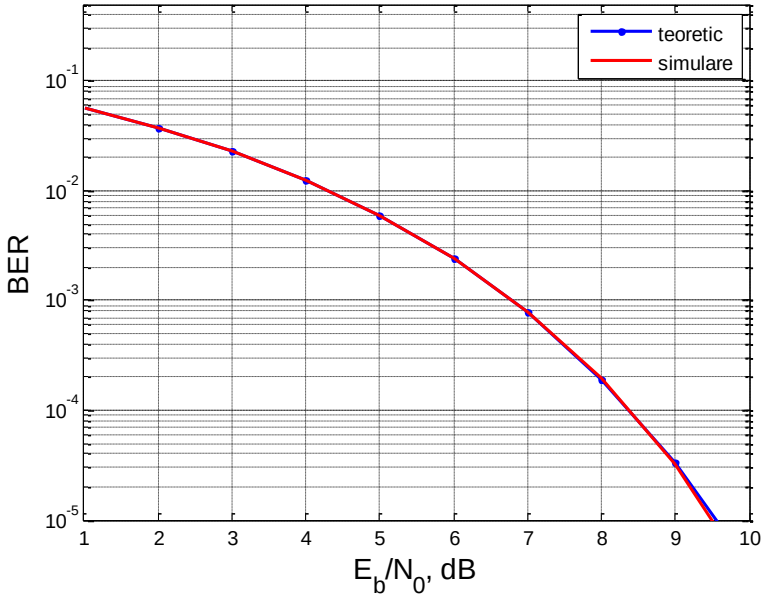


Figura 3.6 Probabilitatea erorii pe bit pentru o modulaţie BPSK (numărul biţilor transmişi este 10^6)

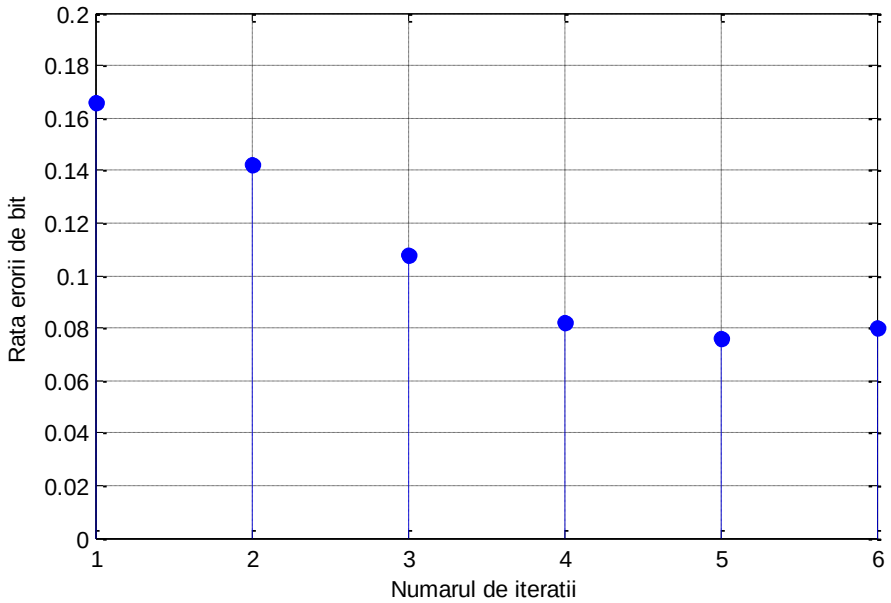


Figura 3.7 Rata erorii de bit in funcţie de numărul de iteraţii ($E_b / N_0 = 0,2$, numărul de biţi=100)

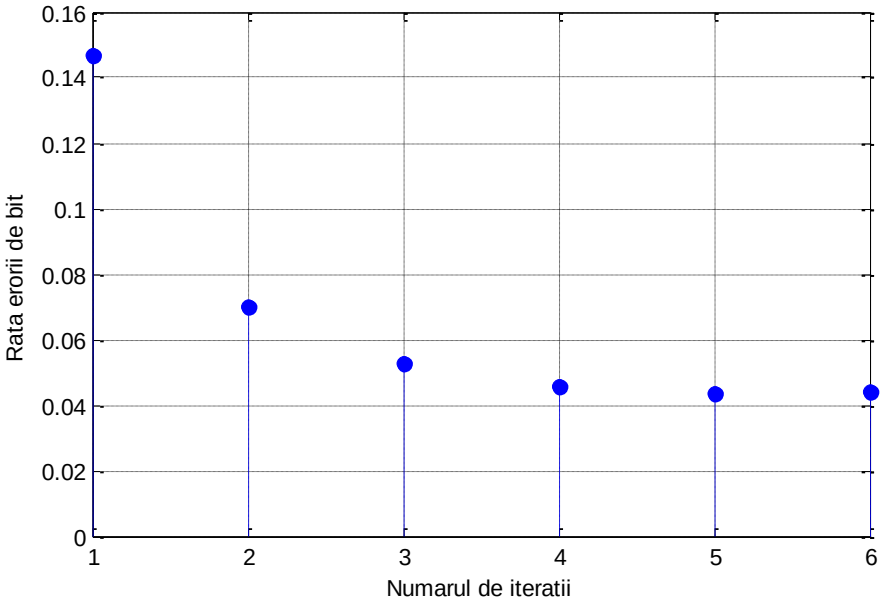


Figura 3.8 Rata erorii de bit în funcţie de numărul de iteraţii
($E_b / N_0 = 0,5$, numărul de biţi=100)

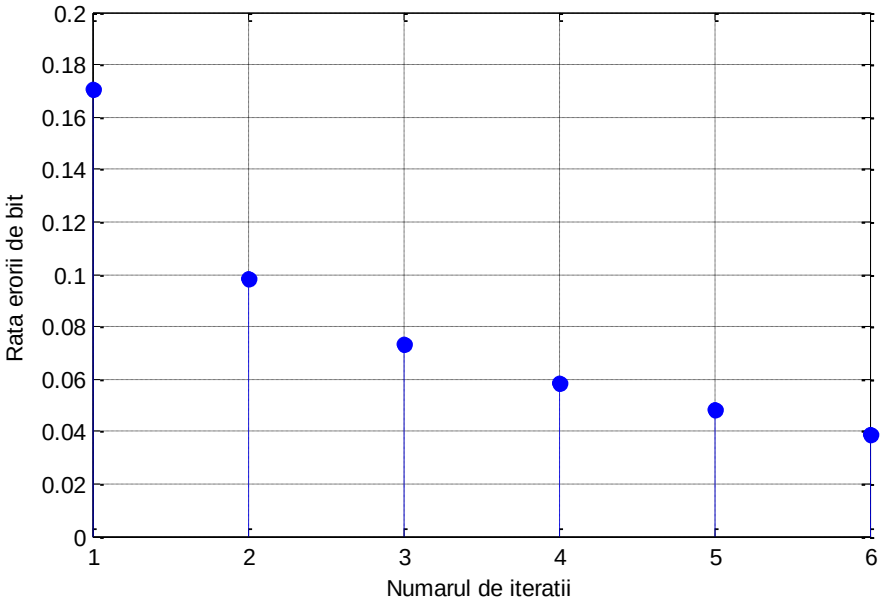


Figura 3.9 Rata erorii pe bit în funcţie de numărul de iteraţii
($E_b / N_0 = 0,2$, numărul de biţi=10000)

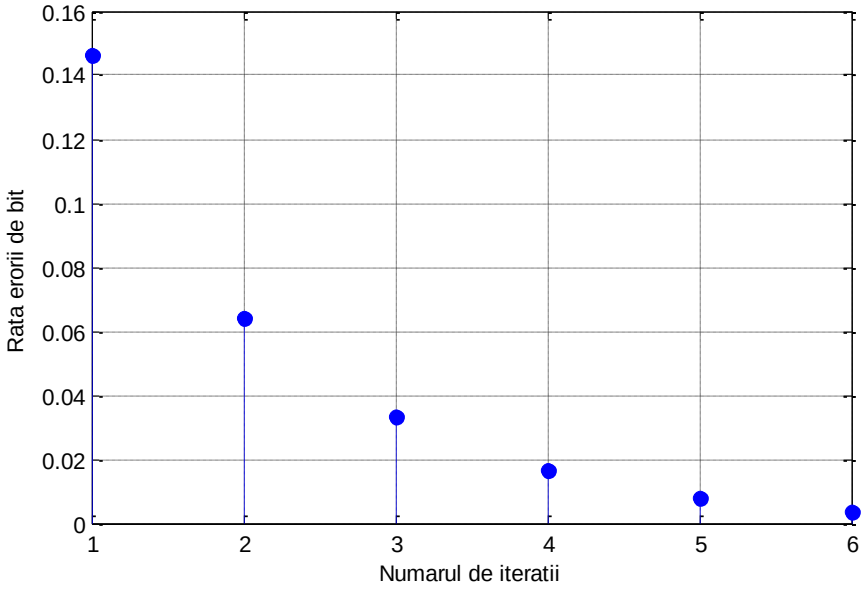


Figura 3.10 Rata erorii pe bit în funcţie de numărul de iteraţii
($E_b / N_0 = 0,5$, numărul de biţi=10000)

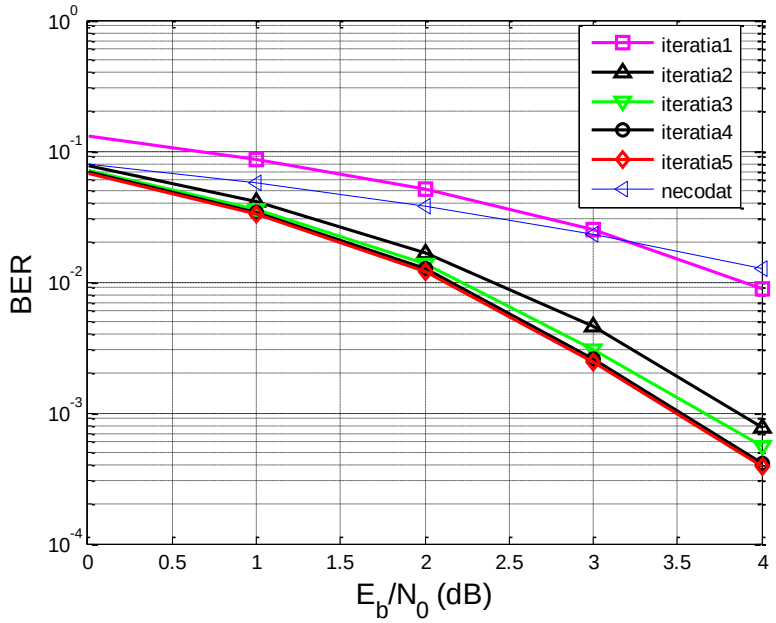


Figura 3.11 BER/SNR (numărul de biţi=10)

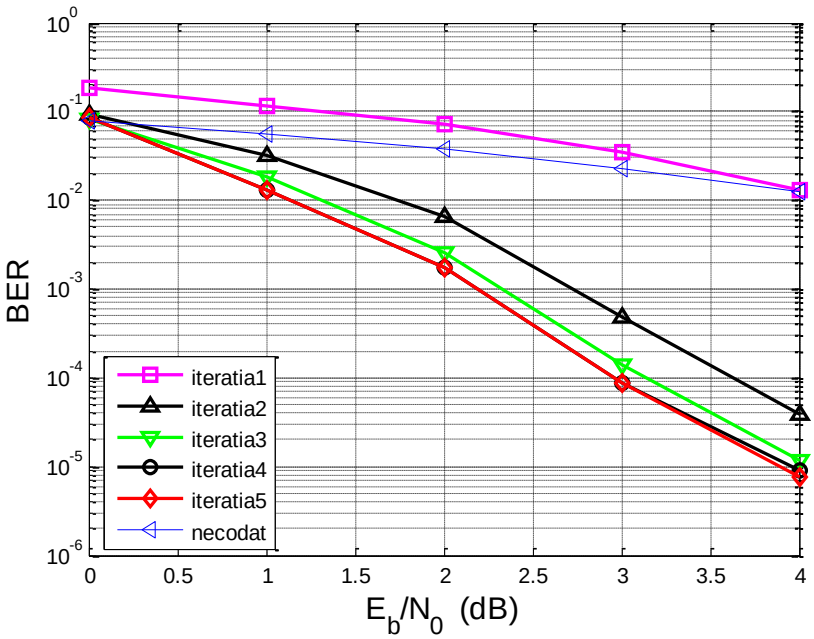


Figura 3.12 BER/SNR (numărul de biţi=100)

Totodată iteraţia cinci nu aduce nici o îmbunătăţire secvenţei estimate de la ieşirea decodorului turbo.În exemplul prezentat răspunsul la impuls h al canalului era considerat ca fiind unitar (nu există propagare multicale).

În cazul în care există propagare multicale în faţa decodorului trebuie introdus un egalizor. Schema bloc a sistemului de transmisie - recepţie este prezentată în figura 3.13. În cazul de faţă am implementat schema bloc din figura 3.13 cu programul Matlab pentru două tipuri de egalizoare şi anume: MMSE şi ZF. Înainte de a face implementarea am făcut o analiză comparativă a performanţelor celor două tipuri de egalizoare în cazul unei transmisii BPSK, considerând estimarea răspunsului la impuls al canalului ca fiind perfectă. Lungimea secvenţei de biţi s-a considerat $k = 10^5$, iar valoarea lui $h = [0.213 \quad 0.812 \quad 0.312]$. În urma simulării am obţinut figura 3.14.

Din figura 3.14 se observă că BER-ul, în cazul egalizării MMSE, este mai mic decât în cazul egalizării ZF pentru aceeaşi valoare a SNR, (ceea ce indică o performanţă mai ridicată a egalizării MMSE faţă de egalizarea ZF).

Revenind la schema bloc din figura 3.13, când comutatoarele sunt în poziţia 1 se transmite o secvenţă de învăţare , s_N , care este cunoscută de estimatorul de canal. Pe baza secvenţei recepţionate $\tilde{s}_N$ şi utilizând unul din algoritmii de estimare cum ar fi MMSE sau LS, se poate determina valoarea estimată a lui h ($\hat{h}$). Când comutatoarele sunt în poziţia 2 se face transmisia datelor.

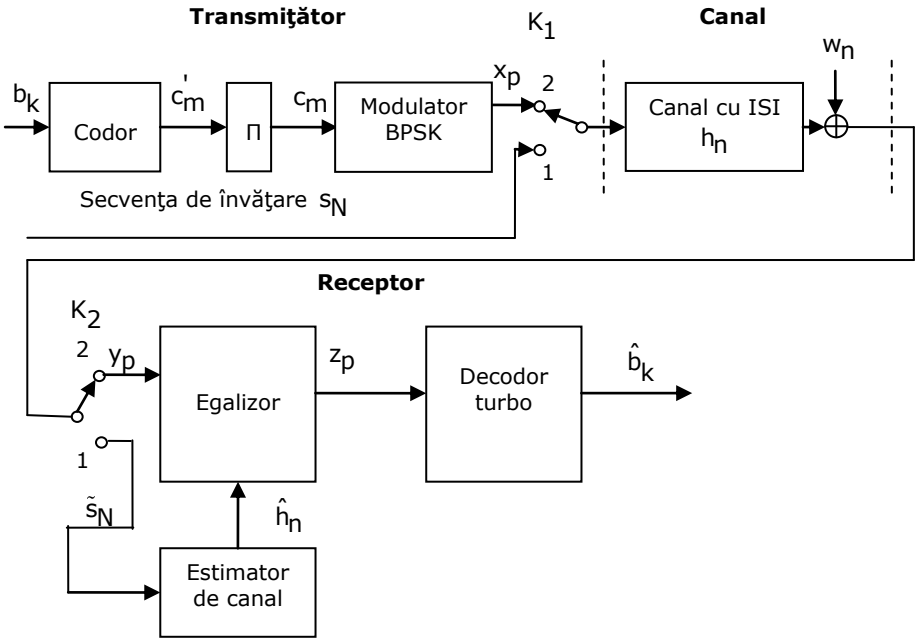


Figura 3.13 Schema bloc a sistemului de transmisie – recepţie care realizează la recepţie estimarea şi egalizarea canalului

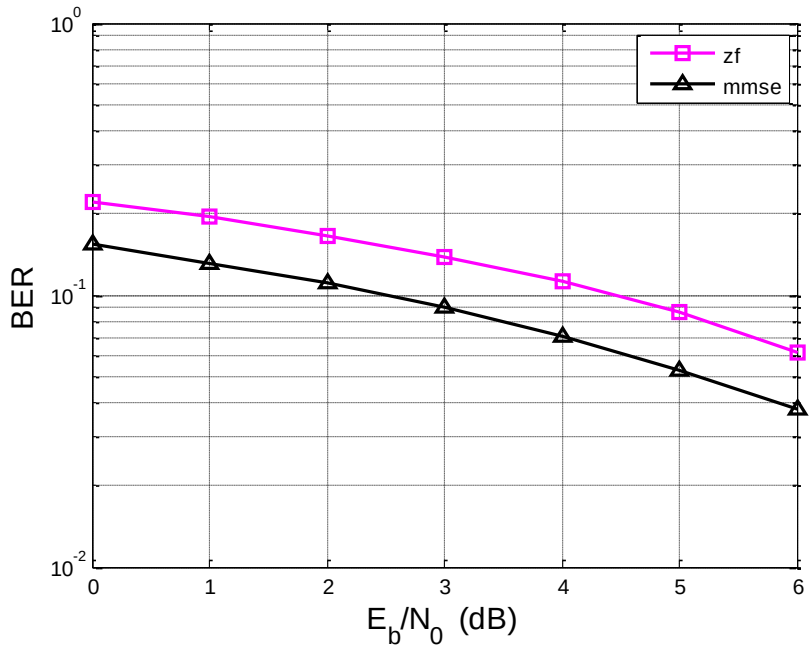


Figura 3.14. BER/SNR în cazul egalizării MMSE şi ZF

$(k = 10^5, h = \hat{h} = [0.213 \quad 0.812 \quad 0.312])$

În cazul de faţă am folosit algoritmul de estimare LS (least-square). Metoda de estimare LS , implică găsirea lui $\hat{h}$ care minimizează eroarea pătratică utilizând următoarea relaţie:

$$\hat{h} = \arg\min_{h} \|\tilde{s} - Sh\|^2 \tag{3.12}$$

Presupunând că zgomotul este alb şi Gaussian, soluţia ecuaţiei de mai sus este:

$$\hat{h}_{LS} = \left(S^T \cdot S\right)^{-1} \cdot S^T \cdot \tilde{s} \tag{3.13}$$

unde S este matricea Toeplitz corespunzătoare secvenţei de învăţare transmise $s[N]$, iar S^T reprezintă transpusa matricei S.

În figurile 3.15 şi 3.16 este prezentată analiza comparativă a performanţelor dintre egalizarea cu egalizor MMSE şi egalizor ZF pentru două valori ale lungimii secvenţei de învăţare.(N=8, respectiv 16). Valoarea răspunsului la impuls al canalului s-a considerat ca fiind $h = [0.2 \quad 0.9 \quad 0.3]$, iar lungimea secvenţei de date care se transmite $k = 128$.

Odată cu creşterea lungimii secvenţei de învăţare, estimarea canalului devine mai bună, dar egalizarea MMSE oferă performante mai ridicate faţă de egalizarea ZF.

O îmbunătăţire a estimării şi implicit a decodării turbo poate fi obţinută prin utilizarea schemei bloc din figura 3.17.

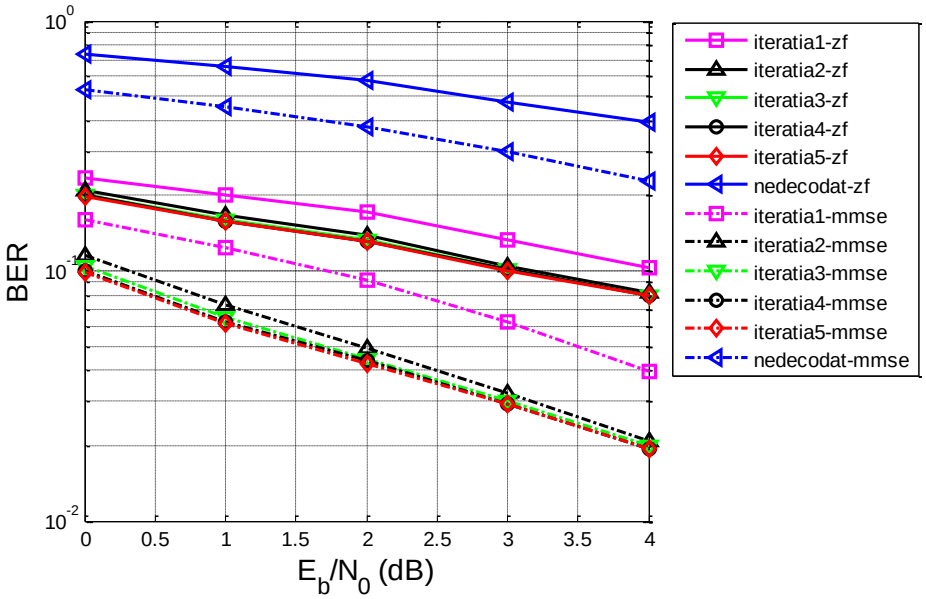


Figura 3.15. BER/SNR ($h = [0.2 \quad 0.9 \quad 0.3]$, $N = 8$, $k = 128$)

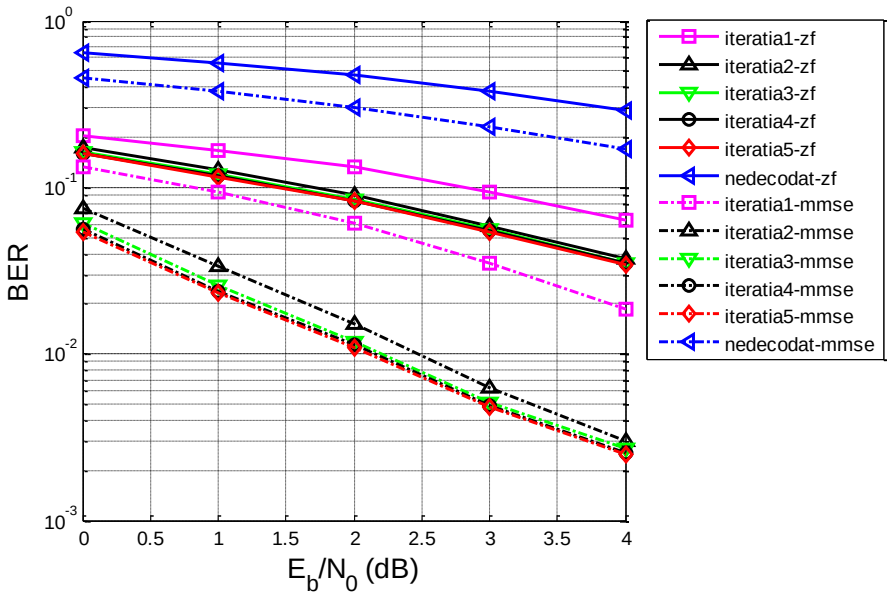


Figura 3.16. BER/SNR ($h = [0.2 \quad 0.9 \quad 0.3]$, $N = 16$, $k = 128$)

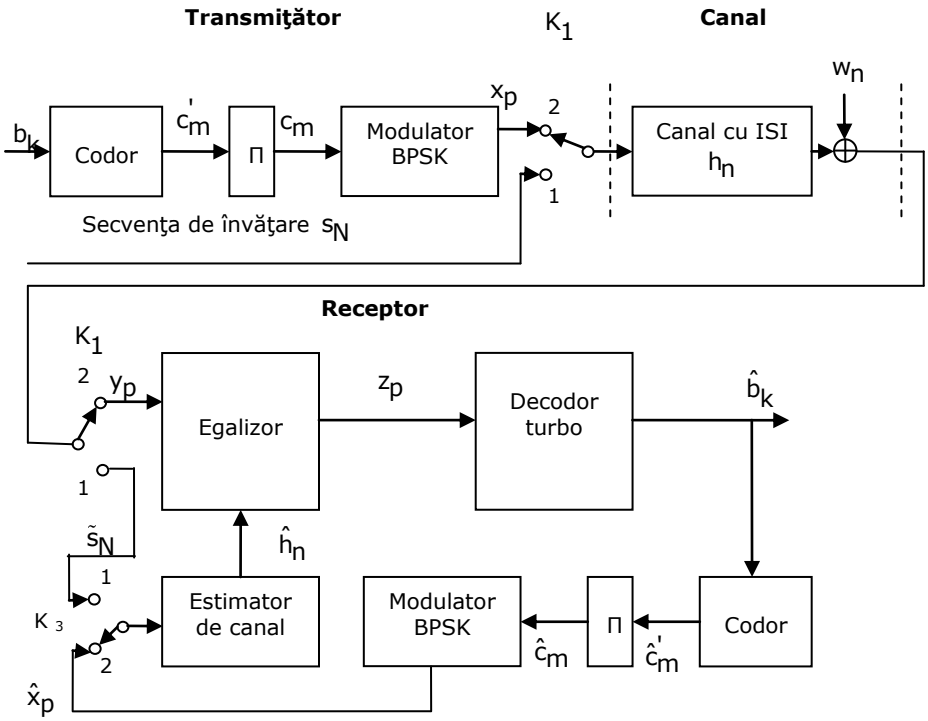


Figura 3.17 Schema bloc a sistemului de transmisie – recepţie care realizează la recepţie estimarea iterativă şi egalizarea canalului

Schema din figura 3.17 funcţionează la fel ca şi schema din figura 3.12, doar că estimarea canalului este făcută iterativ. După un anumit număr de iteraţii impus decodorului turbo, secvenţa estimată de ieşirea acestuia $\hat{b}_k$ este codată, întreţesută iar apoi modulată BPSK şi adusă la estimatorul de canal. Acesta, pe baza secvenţei recepţionate y_p şi a secvenţei estimate transmise $\hat{x}_p$, recalculează cu ajutorul algoritmului de estimare LS valoarea lui $\hat{h}_n$ utilizând relaţia:

$$\hat{h}_{LS} = \left(\hat{X}^T \cdot \hat{X}\right)^{-1} \cdot \hat{X}^T \cdot y \tag{3.14}$$

unde $\hat{X}$ este matricea Toeplitz corespunzătoare secvenţei $\hat{x}[p]$, care intră în estimatorul LS, iar $\hat{X}^T$ reprezintă transpusa matricei $\hat{X}$. Dacă presupunem că:

$$\hat{x}[p] = \left[\hat{x}[1], \hat{x}[2].....\hat{x}[p]\right], \tag{3.15}$$

iar semnalul recepţionat

$$y[p] = \left[y[1], y[2].....y[p]\right]^T \tag{3.16}$$

şi valoarea estimată a lui h

$$\hat{h}_{LS} = \left[\hat{h}[1], \hat{h}[2].....\hat{h}[n]\right]^T \tag{3.17}$$

atunci matricea $\hat{X}$ arată în felul următor:

$$\hat{X} = \begin{bmatrix} \hat{x}[1] & 0 & \cdots & 0 \\ \hat{x}[2] & \hat{x}[1] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \hat{x}[p] & \hat{x}[p-1] & \cdots & \hat{x}[p-n+1] \end{bmatrix} \tag{3.18}$$

În continuare am făcut o analiza comparativă a performanţelor obţinute cu schemele din figurile 3.13 şi 3.17, unde schema din figura 3.13 face estimarea o singură dată, iar schema din figura 3.17 face estimarea iterativ. Pentru această analiză, în ambele scheme (din figura 3.13 şi 3.17) am utilizat egalizorul MMSE, deoarece oferă performanţe mai ridicate decât egalizorul ZF.

3.2.1 Cazul estimării perfecte a canalului

În acest exemplu s-a presupus că estimarea iniţială făcută este perfectă. Pentru aceasta s-a considerat valoarea răspunsului la impuls al canalului ca fiind $h = [0.2 \quad 0.9 \quad 0.3]$, lungimea secvenţei de date care se transmite $k = 128$. În urma simulării am obţinut figura 3.18

În figura 3.18 s-a notat cu "mmse" rezultatele obţinute cu schema din figura 3.13 iar cu "mmseLS" rezultatele obţinute cu schema din figura 3.17. În acest caz se observă că rezultatele obţinute cu schema din figura 3.13 sunt mai bune

decât cele obţinute cu schema din figura 3.17. Rezultă că nu este utilă estimarea iterativă a lui h, dacă acesta a fost estimat corect înaintea procesului de egalizare şi decodare.

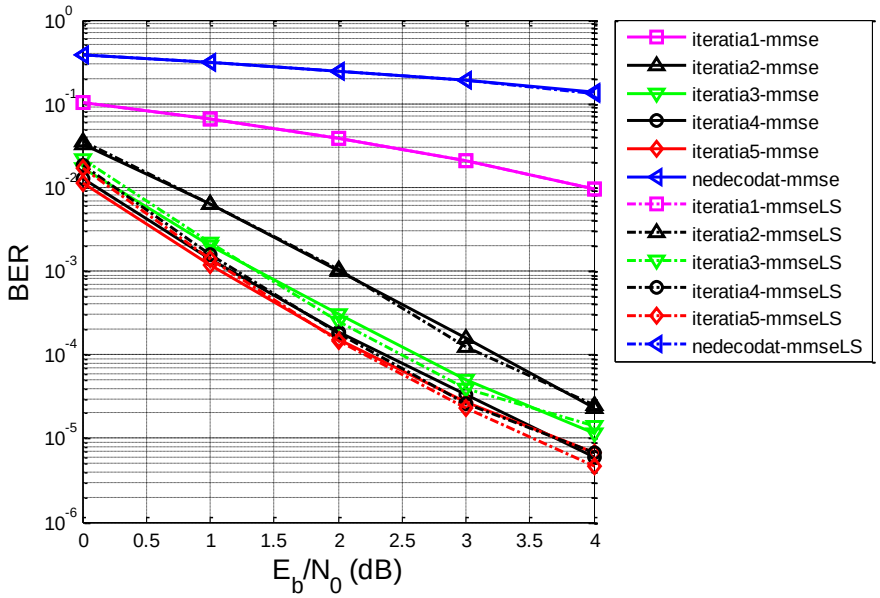


Figura 3.18. BER/SNR ($h = \hat{h} = [0.2 \quad 0.9 \quad 0.3]$, $k = 128$)

3.2.2 Cazul estimării cu algoritmul LS

În acest caz valoarea răspunsului la impuls al canalului s-a considerat aceeaşi $h = [0.2 \quad 0.9 \quad 0.3]$, lungimea secvenţei de date care se transmite $k = 128$, iar lungimea secvenţei de învăţare s-a considerat: i)N=8 , respectiv ii)N=16

După cum se observă din cele două figuri (3.19 şi 3.20), odată cu creşterea secvenţei de învăţare, estimarea devine mai bună, iar BER-ul mai mic. Spre exemplu pentru cazul i) $N = 8$ (figura 3.19) , la o valoare a $SNR = 4dB$, la iteraţia cinci ,BER-ul este cuprins între 10^{-2} şi 10^{-3}, iar în cazul ii) $N = 16$ (figura 3.20), pentru aceeaşi valoare a SNR, la aceeaşi iteraţie, BER-ul este cuprins între 10^{-3} şi 10^{-4}. Totodată se observă că rezultatele obţinute cu estimarea iterativă a lui h sunt mai bune decât cu estimarea acestuia o singură dată.

Dezavantajul schemei din figura 3.17 este numărul mare de iteraţii efectuat de decodorul turbo. În cazul de faţă după primele cinci iteraţii se face o nouă estimare a lui h, şi procesul de decodare se reia cu încă cinci iteraţii.

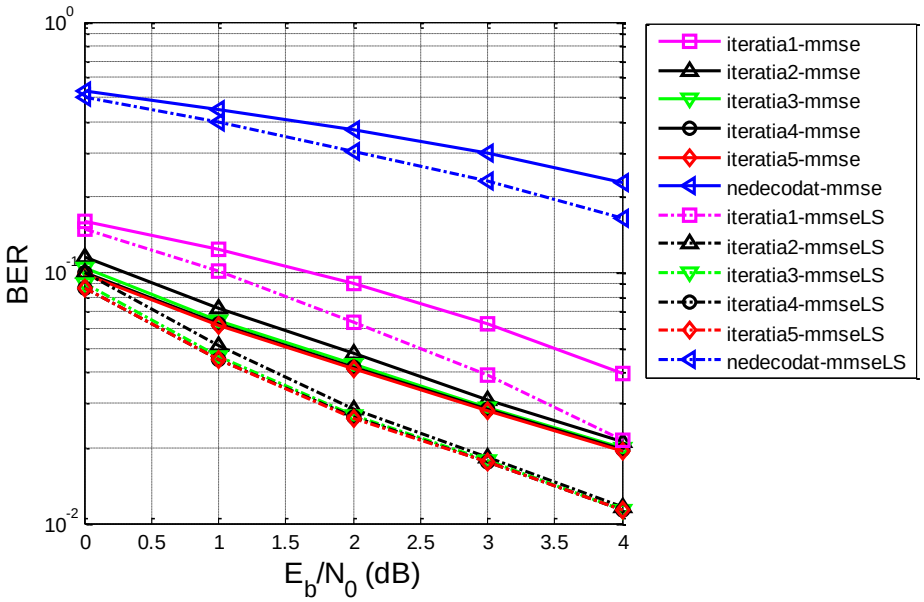


Figura 3.19. BER/SNR ($h = \left[0.2 \quad 0.9 \quad 0.3\right], k = 128, N = 8$)

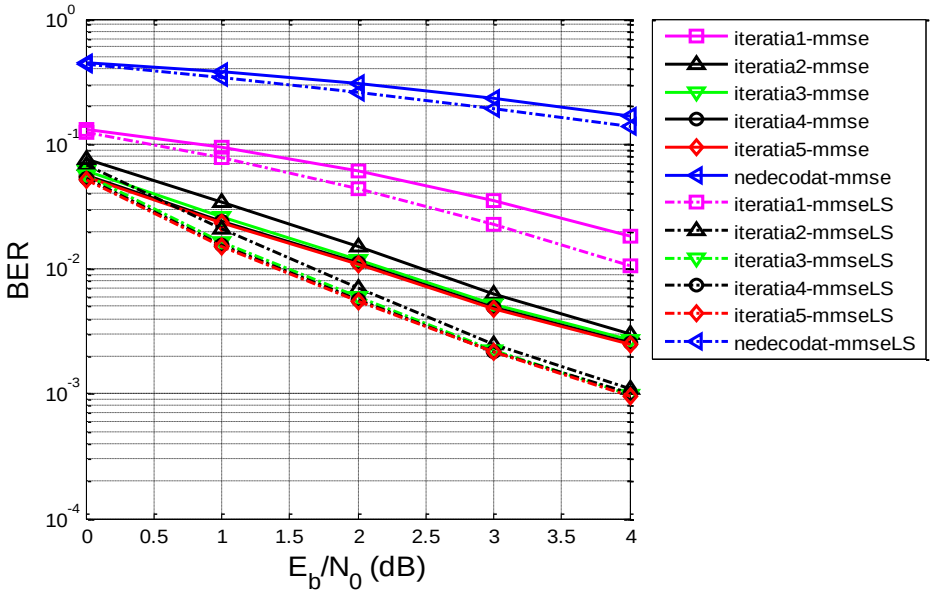


Figura 3.20. BER/SNR ($h = \left[0.2 \quad 0.9 \quad 0.3\right], k = 128, N = 16$)

Dacă s-ar fi făcut de R ori recalcularea lui $\hat{h}$, iar I să fie numărul de iteraţii efectuate de decodor în cadrul unui proces de decodare, atunci numărul total de iteraţii efectuate de decodorul turbo este $(R+1)\cdot I$. În simulările pe care le-am făcut pentru schema din figura 3.17 recalcularea lui $\hat{h}$ a fost făcută o singură dată

($R = 1$) , iar numărul de iteraţii efectuate de decodorul turbo în cadrul unui proces de decodare este cinci $I = 5$. Rezultă că în acest caz numărul minim de iteraţii efectuate de decodorul turbo din figura 3.17 este zece.

În cazul transmisiei multicale am obţinut rezultate mai bune în ce priveşte rata erorii pe bit (BER mai mic la aceeaşi valoare a SNR) prin utilizarea unui turboegalizor la recepţie.

Aceste rezultate precum şi descrierea principiului de funcţionare al turboegalizorului sunt prezentate în capitolul patru.

Capitolul 4
Turbo egalizarea

Turbo egalizarea a fost propusă pentru prima dată de Catherine Douillard s.a. în 1995 [3], pentru un sistem cu modulatie BPSK (Binary Phase Shift Keying) şi coduri convolutionale concatenate serie - figura 4.1.

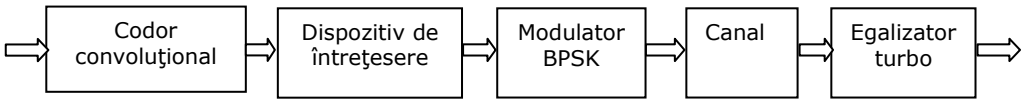


Figura 4.1 Sistem serial format din codor convoluţional, dispozitiv de întreţesere, modulator BPSK şi un turbo egalizor, care efectuează egalizare, demodulare şi decodare iterativă.

Egalizorul turbo propus de autori –figura 4.2 – reduce efectele ISI, când răspunsul la impuls al canalului este perfect cunoscut. Egalizarea şi decodarea nu se fac ca operatii independente; pentru de a depăşi problemele create de selectivitatea în frecvenţă a canalului, o performanţă mai bună se poate obţine cu un egalizor turbo, care efectuează iterativ atât egalizarea cât şi decodarea, considerând canalul discret cu memorie. Turbo-egalizarea se bazează pe decodarea turbo iterativă, cu două decodoare SISO, structură propusă de C. Berrou s.a. în [2, 23].

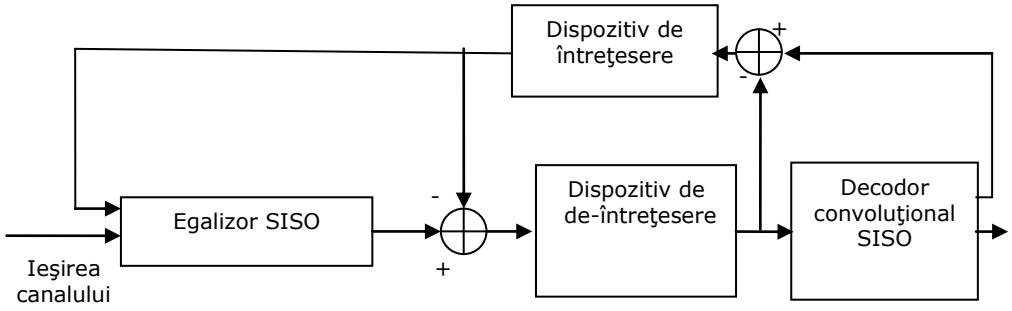


Figura 4.2: Structura originală a egalizorului turbo introdusă de C.Douillard

Informaţia a priori asociată unui bit v_m, cunoscută înainte de egalizare sau decodare, de la altă sursă decat secvenţa recepţionată sau constrângerile codului, mai este numită şi *informaţie intrinsecă*. *Informaţia extrinsecă* asociată bitului v_m este furnizată de egalizor sau decodor din secvenţa recepţionată şi informaţiile a priori ale tuturor celorlati biţi, cu excepţia informaţiei a priori referitoare la bitul v_m. Informaţia extrinsecă referitoare la un anumit bit există din diferite cauze. Când se utilizează un codor corector de erori, cu memorie, fiecare bit de intrare influenţează o secvenţa codată lungă. Deoarece un codor convoluţional convenţional

are un răspuns la impuls de durata infinită [24], fiecare bit de intrare influenţează un număr infinit de biţi codaţi de ieşire, teoretic. Practic, intervalul este redus la de cinci ori constrângerea codului, cu aproximaţie. În codarea turbo cu coduri RSC, este folosit de obicei un interleaver cu memorie mare, care extinde numărul de biţi codaţi influenţati de un bit de intrare, deoarece răspunsul la impuls al codorului turbo este de obicei prelungit. Acest lucru justifică puterea mare de corectie şi interleaverul lung. Deoarece un bit de intrare influenţează multi biţi din secvenţa codată, chiar şi când încrederea este redusă într-o decizie particulară asupra unui bit, informaţia extrinsecă referitoare la acesta a fost distribuită pe un număr mare de biţi codaţi. Cu informaţia extrinsecă, decodorul turbo iterativ poate spori încrederea în decizia de bit, iniţial redusă. Informaţia intrinsecă şi extrinsecă legate de un bit trebuie tratate separat de decodor şi astfel răman necorelate, pentru ca decodorul să fie capabil sa îmbunătăţească fiecare estimat din iteraţiile consecutive ale decodarii turbo. În afară de memoria codorului turbo există alte mecanisme generatoare de informaţii extrinseci datorate memoriei lor, ca memoria canalului ce induce dispersia în timp, respectiv CISI (Controlled Intersymbol Interference) din modulatoarele GMSK.

Informaţia a-posteriori asociată unui bit este informaţia dată de algoritmul SISO luând în considerare toate informaţiile disponibile despre biţii u_k. Egalizorul turbo constă dintr-un egalizor SISO şi un decodor SISO. Egalizorul SISO - figura 4.2 - generează probabilitatea a-posteriori, obţinută din secvenţa recepţionată din canal (în care unii biţi pot fi eronaţi) şi probabilitatea a-priori furnizată de decodorul SISO. Dar la prima iteraţie de egalizare turbo, decodorul nu furnizează nici o informaţie a-priori; probabilitatea a-priori este setată la ½ - biţii transmişi se presupun echiprobabili.

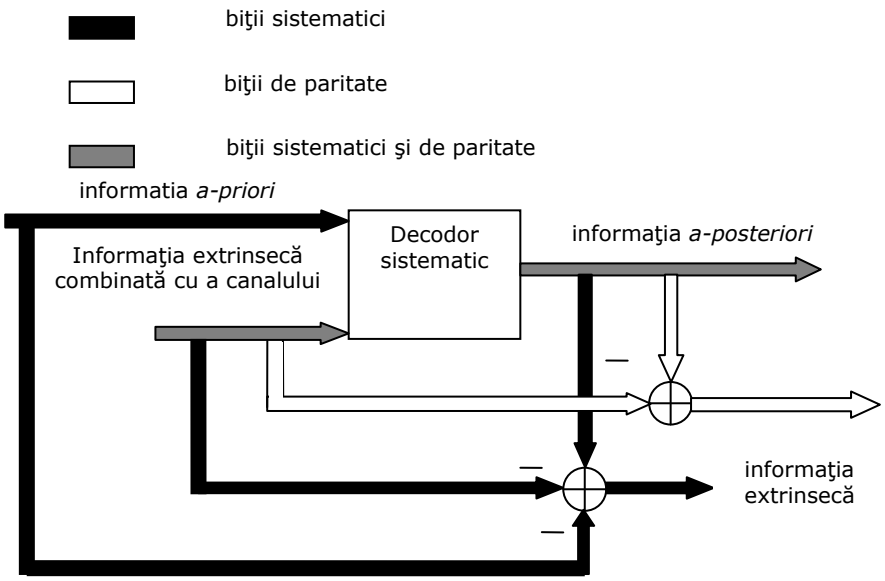


Figura 4.3 Schema decodorului utilizat în turboegalizare

Înainte de a pasa decodorului SISO informaţia a-posteriori generată de egalizorul SISO, trebuie eliminată contribuţia decodorului (ca informaţie a-priori), de la iteraţia anterioară, pentru a aduce la intrarea decodorului doar informaţiile extrinseci combinate cu ale canalului.

Se minimizează astfel si corelatia dintre informaţiile apriori furnizate de decodor şi cele aposteriori generate de egalizor. Informaţiile extrinseci combinate cu ale canalului sunt inerent legate, nu pot fi separate, fiind induse de mecanisme cu memorie.

Eliminarea informaţiilor a-priori este necesară pentru ca decodorul să nu-si recepţioneze propria informaţie - caz în care ar apărea o reactie pozitiva care ar distruge estimarea curentă a decodorului referitoare la biţii codaţi, adică informaţia extrinseca.

Informaţiile combinate, extrinseci si ale canalului, sunt de-întreţesute (de de-interleaverul de canal) şi trimise decodorului SISO care va calcula probabilitatea a-posteriori a biţilor codaţi. Paşii acestia diferă de cei de la turbo decodare. Decodoarele componente dintr-un decodor turbo produc doar probabilitatea a-posteriori a biţilor sursă, şi nu a biţilor codaţi pentru canal - vezi figura 4.4.

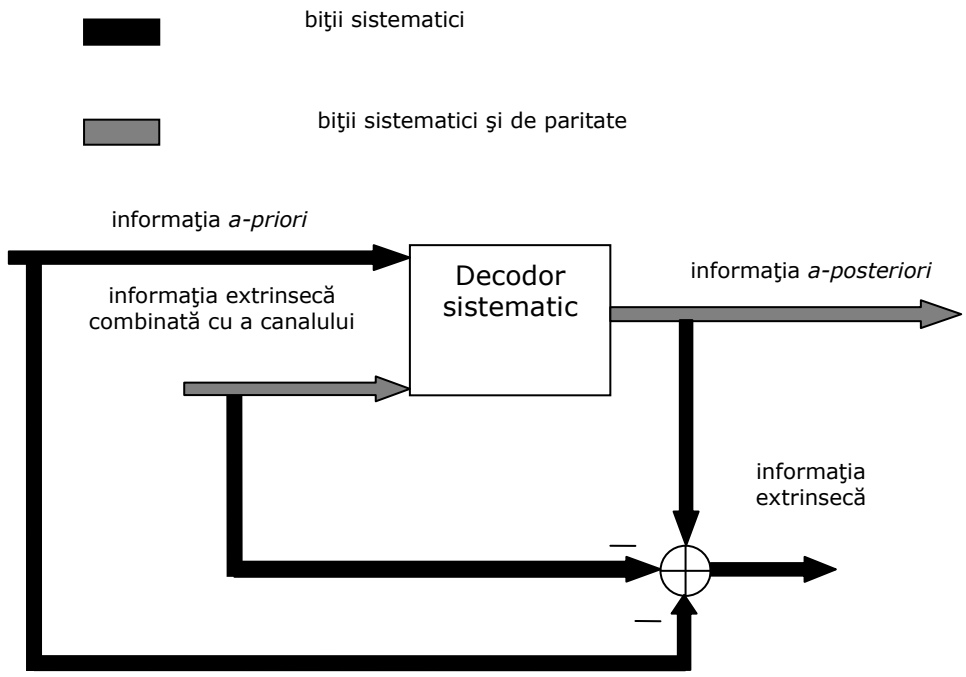


Figura 4.4 Schema decodorului utilizat în turbodecodare

Decodorul din turbo egalizor calculează valorile LLR (Log-Likelihood Ratio) a-posteriori pentru biţii de paritate şi cei sistematici -figura 4.3. Pentru a obţine informaţiile extrinseci, informaţiile extrinseci combinate cu ale canalului sunt scoase din informaţiile a-posteriori provenite din decodor -figura 4.2- înainte ca acestea din urmă să fie întreţesute. Acest lucru se face prin scăderea valorilor LLR ale biţilor sistematici care conţin informaţiile extrinseci combinate cu ale canalului şi a valorilor

LLR a-priori din valorile LLR a-posteriori a biţilor sistematici -figura 4.3. Prin această operaţie rezultă doar LLR extrinseci corespunzătoare biţilor sistematici.

Pentru obţinerea valorilor LLR extrinseci ale biţilor de paritate - figura 4.3 - se scad numai valorile LLR ale biţilor de paritate care conţin informaţiile extrinseci combinate cu ale canalului din valorile LLR a-posteriori corespunzătoare biţilor de paritate. Scopul scăderii acestor valori LLR de intrare din valorile LLR a-posteriori este de a nu lăsa egalizorul să recepţioneze informaţii bazate pe propriile decizii, care au fost generate în iteraţia anterioară de egalizare turbo.

Informaţia extrinsecă calculată este apoi adusă ca informaţie de intrare a-priori egalizorului în următoarea etapă a procesului de egalizare de canal. Aceasta constituie prima iteraţie de egalizare turbo. Procesul iterativ se repetă până când criteriile de terminare cerute sunt îndeplinite [25]. În acest stadiu, informaţia a-posteriori a biţilor sursă, care a fost generată de decodor, este utilizată la estimarea biţilor transmişi.

Structura unui egalizor turbo având două decodoare componente este prezentată în Figura 4.5. Pentru simplitate, s-a omis interleaverul şi s-au notat doar poziţiile interleaverului , unde π_c reprezintă interleaverul de canal iar π_t turbo interleaverul. Indicele -1 este folosit pentru de-interleaver.

4.1 Principiul egalizării turbo utilizând unul sau mai multe decodoare

În această secţiune se va descrie principiul egalizării turbo pentru un receptor care lucrează în banda de bază compus dintr-un egalizor şi un număr N_d de decodoare componente folosind figura 4.5.

Această schemă este o extensie a sistemului iniţial de egalizare turbo [3], ilustrat în figura 4.2. Pentru simplitate, s-a omis interleaverul de canal π_c şi turbo interleaverul π_t şi s-au marcat numai poziţiile lor. Puterea „-1” reprezinta o de-întreţesere. De obicei, pentru codurile turbo sunt Nd = 2 decodoare componente, în timp ce pentru decodarea convoluţională non-iterativă este un singur decodor (Nd = 1). Regula pentru egalizatoarele turbo este că informaţiile de intrare pentru un anumit bloc în iteraţia curentă nu trebuie să conţină informaţia obţinută de acest bloc în iteraţia anterioară. În caz contrar informaţia utilizată în iteraţii consecutive ar fi dependentă de cea de dinainte.

Egalizorul şi decodorul din figura 4.5 folosesc un algoritm SISO, cum ar fi algoritmul MAP optimal [26], algoritmul Log-MAP [27] sau Soft Output Viterbi Algorithm (SOVA) [28, 29, 30], care dă informaţii a-posteriori.

Aşa cum este definită anterior, informaţia aposteriori cu privire la bit este informaţia pe care blocul SISO o generează, luând în considerare toate sursele disponibile de informaţii despre acel bit.

Când se foloseşte algoritmul MAP sau algoritmul Log-MAP, exprimăm informaţia a posteriori din punct de vedere al valorilor LLR [31]. Valoarea LLR a unui bit v_m , L^{v_m} , este definită ca logaritmul natural al raportului probabilităţilor ca biţii sa ia una din cele două valori posibile de 1 şi -1:

$$L^{v_m} \triangleq ln\left(\frac{P(v_m = +1)}{P(v_m = -1)}\right) \tag{4.1}$$

Valoarea LLR a egalizorului şi decodorului utilizând notaţia vectoriala [32].

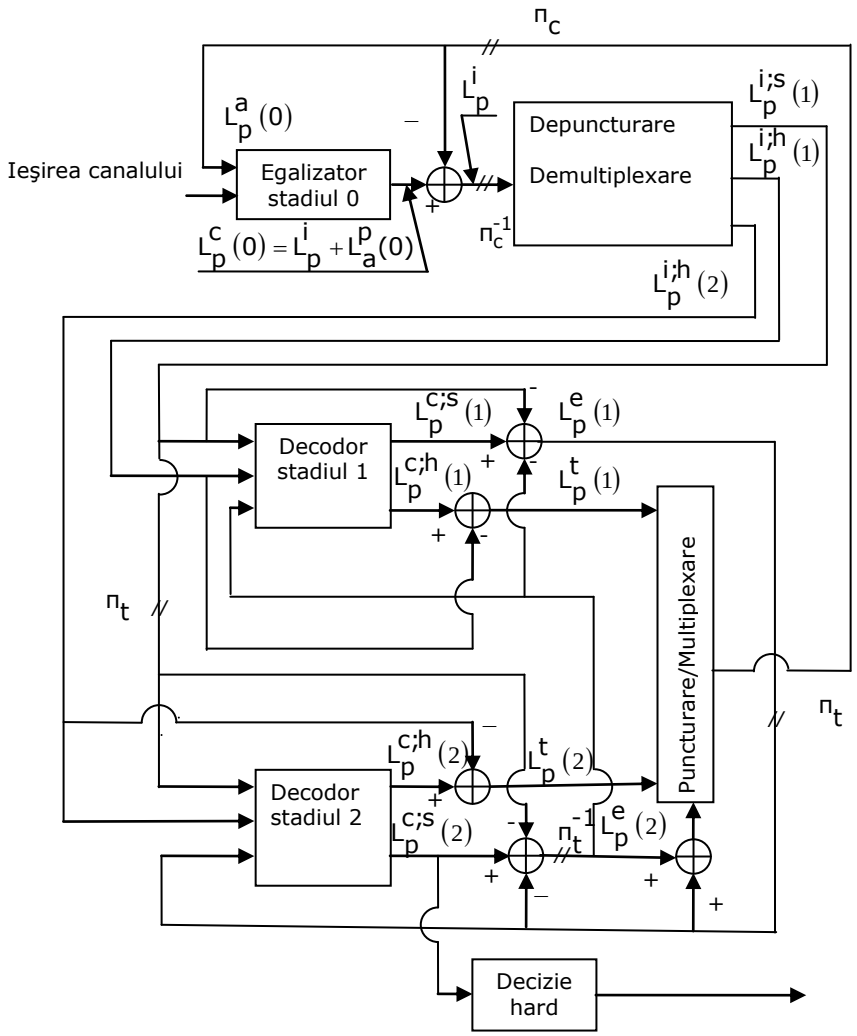


Figura 4.5 Structura unui egalizor turbo având două decodoare componente

Exponentul asociat indică natura LLR-ului şi anume:

L^c - LLR a-posteriori compus constând din sursa (informaţie) şi din bitul de paritate (informaţia a-posteriori)

$L^{c;s}$: informatia a-posteriori a bitului sursă

$L^{c;h}$: informatia a-posteriori a bitului de paritate

L^i : informaţia extrinsecă combinată cu a canalului, pentru biţii sursă şi biţii de paritate.

$L^{i;s}$: informaţia extrinsecă combinată cu a canalului pentru biţii sursă.

$L^{i;h}$ informaţia extrinsecă combinată cu a canalului pentru biţii de paritate

L^s : informaţia extrinsecă corespunzătoare bitului sursă.

L^t :informaţia extrinsecă a bitului de paritate.

$L^{a;s}$ informaţia a-priori a bitului sursă

$L^{a;h}$ informaţia a-priori a bitului de paritate

În plus, indicele este utilizat pentru a reprezenta numărul iteraţiei, în timp ce argumentul din paranteze () este indicele care reprezintă starea receptorului.

La egalizor, care este notat în nivelul 0 în figura 4.5, valoarea ieşirii LLR-aposteriori, $L_p^c(0)$, la iteraţia p este dată de suma valorii LLR a informaţiei a-priori , $L_p^a(0)$, şi de valoarea LLR a informaţia extrinsecă combinată cu a canalului, $L_p^i(0)$, rezultând:

$$L_p^c(0) = L_p^i + L_p^a(0) \tag{4.2}$$

La începutul detecţiei iterative nu există nici o informaţie apriori despre bitul care va fi decodat şi, prin urmare se utilizează $L_p^a(0) = 0$, ceea ce indică o probabilitate egală de unu şi zero. Cu toate acestea, în timpul iteraţiei următoare, estimarea cu privire la acest bit poate fi adusă înapoi la intrarea egalizorului (figura 4.5), cu scopul de a ajuta iteraţiile sale ulterioare.

Este imposibil să se separe informaţiile de la ieşirea canalului şi informaţiile extrinseci la ieşirea de egalizorului, care se notează cu L_p^i, deoarece răspunsul la impuls al canalului poate fi privit ca acela a unui cod nesistematic [33] care "întinde" efectele biţilor de intrare ai canalului în timp din cauza operaţiei de convoluţie asociat.

Luând în considerare stadiul 0 Figura 4.6, care este o ilustrare detaliată a egalizorului, cu accent pe informaţiile de intrare şi ieşire, se remarcă faptul că valorile $L_p^c(0)$ (informaţia compusă a-posteriori), $L_p^a(0)$ (apriori) şi L_p^i (LLR pentru informaţia extrinsecă combinată cu a canalului) reflectă fiabilitatea nu numai a biţilor sursă ci şi a biţilor de paritate.

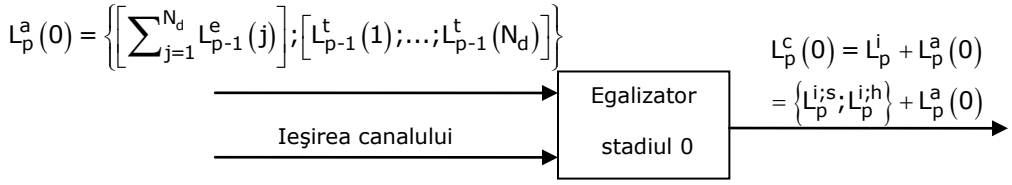


Figura 4.6 Schema egalizorului SISO cu accent pe informaţia de intrare şi de ieşire a acestuia la iteraţia p

Termenul L_p^i poate fi scris ca un vector, având în vedere ca:

$$L_p^i \in \left\{ \underbrace{L_p^{i;s}}_{\text{bitul sursa}} ; \underbrace{L_p^{i;h}}_{\text{biţii de paritate}} \right\} \tag{4.3}$$

În stadiul 0 vectorul $L_p^a(0)$ corespunzător informaţiei apriori se obţine de la un decodor N_d aflat in stadiul 1 sau 2 (Figura 4.5), şi conţine informaţiile apriori a biţilor codificaţi. De aceea, la fel ca şi vectorul L_p^i , $L_p^a(0)$ este, de asemenea, un vector format din informaţia apriori pentru biţii sursă şi biţii de paritate, şi prin urmare, este dat de relaţia:

$$L_p^a(0) \in \left\{ \underbrace{L_p^{a;s}(0)}_{\substack{\text{bitul}\\ \text{sursa}}} ; \underbrace{L_p^{a;h}(0)}_{\substack{\text{biţii de}\\ \text{paritate}}} \right\} \in \left\{ \underbrace{\sum_{j=1}^{N_d} L_{p-1}^e(j)}_{\substack{\text{bitul}\\ \text{sursa}}} ; \underbrace{L_{p-1}^t(1); \ldots L_{p-1}^t(N_d)}_{\substack{\text{biţii de}\\ \text{paritate}}} \right\}$$

unde $L_{p-1}^e(j)$ şi $L_{p-1}^t(j)$ sunt valorile LLR extrinseci a bitului sursă şi respectiv de paritate ale decodorului in starea j din iteraţia anterioara $p-1$, în timp ce N_d este numărul decodoarelor componente (figura 4.5).

Folosind relaţiile 4.2 şi 4.3, valoarea LLR a-posteriori a biţilor codificaţi la ieşirea egalizorului $L_p^c(0)$ (din figura 4.5), poate fi exprimat ca :

$$L_p^c(0) \in \left\{ \underbrace{L_p^{c;s}(0)}_{\substack{\text{bitul}\\ \text{sursa}}} ; \underbrace{L_p^{c;h}(0)}_{\substack{\text{biţii de}\\ \text{paritate}}} \right\} \in \left\{ \underbrace{L_p^{i;s} + \sum_{j=1}^{N_d} L_{p-1}^e(j)}_{\substack{\text{bitul}\\ \text{sursa}}} ; \underbrace{\left[L_p^{i;h} + L_{p-1}^t(1); \ldots; L_p^{i;h} + L_{p-1}^t(N_d) \right]}_{\substack{\text{biţii de}\\ \text{paritate}}} \right\}$$

Etapa b a decodorului din figura 4.5 este ilustrată, mai în detaliu în figura 4.7, cu accent pe informaţiile de intrare şi de ieşire .

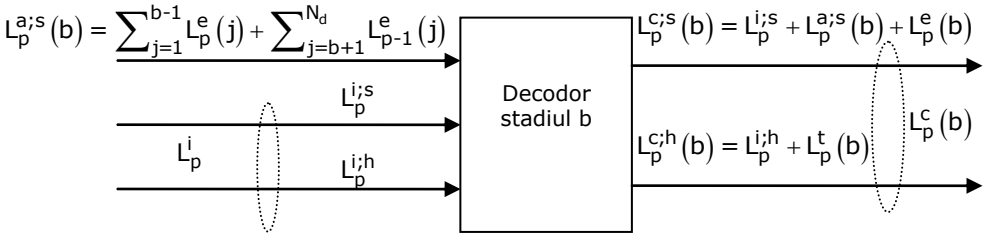


Figura 4.7 Schema decodorului SISO cu accent pe informaţia de intrare şi de ieşire a acestuia la iteraţia p

Acesta recepţionează valoarea LLR a informaţiei a-priori ($L_p^{a;s}$) dată de suma ($\sum_{j=1}^{b-1} L_p^e(j) + \sum_{j=b+1}^{N_d} L_{p-1}^e(j)$) informaţilor extrinseci de la toate celelalte decodoare, (cu excepţia etapei b a decodorului) şi valoarea LLR a informaţiei extrinseci combinate cu a canalului (a biţilor sursă $L_p^{i;s}$ şi a biţilor de paritate $L_p^{i;h}$).

Valoarea LLR a sursei mărite de informaţii a-priori a decodorului din starea b la iteraţia p poate fi exprimat ca:

$$L_p^{a;s}(b) = \sum_{j=1}^{b-1} L_p^e(j) + \sum_{j=b+1}^{N_d} L_{p-1}^e(j) \tag{4.4}$$

Din relaţia 4.4 se observă că informaţiile a-priori $L_p^{a;s}(b)$ a biţilor sursă conţin informaţia extrinsecă de la decodoare din etapele anterioare şi anume pentru $j < b$, în iteraţia curentă p şi de la decodoare din etapele ulterioare şi anume pentru $j > b$ din iteraţia anterioară $p-1$, dar nu include nici o informaţie extrinsecă de la stadiul b.

Spre deosebire de egalizor, care recepţionează atât informaţiile a-priori ale sursei cât şi ale biţilor de paritate, decodoarele N_d acceptă numai valorile LLR apriori ale biţilor sursă.Acest lucru se datorează faptului că bitul sursă este singura informaţie comună pentru toate decodoarele, fie în ordinea întreţesută sau a secvenţei iniţiale, în timp ce biţii de paritate sunt exclusiv pentru un anumit decodor, şi prin urmare aceştia nu sunt în măsură să contribuie în operaţia de decodare a celorlalte decodoare.

De exemplu, la etapa b decodorul sistematic, la ieşirile din etapele 1 şi 2 (din figura 4.5 şi mai în detaliu în figura 4.7) LLR-ul compozitului a-posteriori $L_p^c(b)$ este format din doi vectori LLR, şi anume informaţia a-posteriori a bitului sursă $L_p^{c;s}(b)$ şi informaţia a posteriori a bitului de paritate $L_p^{c;h}(b)$:

$$L_p^c(b) \in \left\{ L_p^{c;s}(b); L_p^{c;h}(b) \right\} \tag{4.5}$$

Valoarea LLR a sursei compuse a-posteriori, $L_p^{c;s}(b)$, care este prima componentă în relaţia 4.5 poate fi exprimată ca suma dintre valoarea informaţiei a-priori mărita $L_p^{a;s}(b)$, a valorii informaţiilor extrinseci $L_p^e(b)$ produse de stadiul decodorului şi de valoarea LLR a informaţiei extrinseci combinate cu a canalului a bitului sursă din egalizor $L_p^{i;s}(b)$:

$$L_p^{c;s}(b) = L_p^{a;s}(b) + L_p^e(b) + L_p^{i;s} \tag{4.6}$$

care pot fi rescrise cu ajutorul relaţiilor 4.4 şi 4.6 ca:

$$\begin{aligned} L_p^{c;s}(b) &= L_p^{a;s}(b) + L_p^e(b) + L_p^{i;s} \\ &= \sum_{j=1}^{b-1} L_p^e(j) + \sum_{j=b+1}^{N_d} L_{p-1}^e(j) + L_p^e(b) + L_p^{i;s} \\ &= \sum_{j=1}^{b} L_p^e(j) + \sum_{j=b+1}^{N_d} L_{p-1}^e(j) + L_p^{i;s} \end{aligned} \tag{4.7}$$

în timp ce LLR-ul de paritate a-posteriori $L_p^{c;h}(b)$ din relaţia 4.5 poate fi exprimat ca:

$$L_p^{c;h}(b) = L_p^{i;h}(b) + L_p^{t}(b) \tag{4.8}$$

Termenul $L_p^{i;h}(b)$ este informaţia extrinsecă combinată cu a canalului al bitului de paritate, iar $L_p^{t}(b)$ este LLR-ul extrinsec corespunzător bitului de paritate.

În general, în practica turbo codării sunt utilizate doar două codoare componente. De aceea, înlocuind $N_d = 2$ în relaţiile 4.2, 4.7 şi 4.8, putem determina LLR-urile a-posteriori ale egalizorului şi decodoarelor, întrucât relaţiile 4.2 şi 4.4 pot fi folosite pentru a determina corespondenţa între intrarea a-priori la egalizor şi intrările apriori la decodoare. Figura 4.5 ilustrează structura egalizorului turbo, care are principiile discutate mai sus. În cele ce urmează se vor descrie principalele componente ale figurii 4.5, şi anume: egalizorul şi decodoarele.

4.2 Turbo Egalizor cu intrare şi ieşire soft

În [35] este prezentat modul în care algoritmul Log-MAP poate fi folosit ca un egalizor GMSK cu ieşire soft, împreună cu decodarea turbo. Aici iteraţiile au fost efectuate numai în turbo decodor. În acest caz, egalizorul nu a recepţionat nici o valoare a-priori de la o sursă independentă. Atunci când se pune în aplicare egalizarea turbo, egalizorul SISO va primi nu numai ieşirile canalului, ci şi informaţia apriori a decodorului SISO. Se va putea folosi algoritmul Log-MAP care este descris în [35] pentru egalizorul GMSK cu ieşire soft , dar va trebui să se ia în considere în plus la intrarea egalizorului turbo şi informaţiile a-priori de la decodorul SISO. În mod explicit, termenul de $P(u_m)$ care reprezintă probabilitatea a-priori a biţilor de cod necesară atunci când se evaluează $\Gamma_m(\acute{k},k)$ în relaţia (4.10) nu mai este $1/2$, deoarece probabilităţile bitului u_m de a fi +1, respectiv -1 nu mai sunt egale. În schimb, probabilitatea $P(u_m)$ este completată cu ajutorul informaţiilor a-priori de la decodor (decodoare), după cum s-a văzut în relaţia 4.2.

De obicei, pentru a decoda codurile convoluţionale, este folosit algoritmul Viterbi [8,9], algoritm de estimare a secvenţei cu probabilitate maximă. Cu toate acestea, în egalizarea turbo este nevoie de ieşiri soft, care reprezintă LLR-urile biţilor codaţi individual. În următoarul paragraf, se va arăta modul în care sunt determinate valorile LLR ale acestor biţi codaţi.

4.3 Decodor cu intrare şi ieşire soft pentru egalizarea turbo

În egalizarea turbo blocul decodor SISO acceptă intrări soft din egalizorul SISO şi generează LLR nu numai pentru biţii sursă, ci şi pentru biţii codificaţi. De aceea, decodorul trebuie să aibă algoritmi SISO, cum ar fi algoritmul MAP, Log-Map sau SOVA. În această prezentare se va folosi algoritmul Log-MAP deoarece cu acesta se obţin performanţe identice ca şi cu algoritmul MAP optimal, şi totodată implică complexitate redusă.

Cu algoritmul Log-MAP LLR-urile biţilor sursă şi cele ale biţilor codificaţi care sunt daţi de decodorul turbo şi respectiv de turbo egalizor, pot fi calculate în două etape principale. Înainte de a evidenţia aceşti paşi se vor defini notaţiile utilizate. În decodor tranziţiile de trellis sunt notate cu indicele d, iar intervalele de tranziţie ale egalizorului sunt reprezentate cu m. Notaţia v_d reprezintă bitul sursă, în timp ce $c_{l,d}$ reprezintă al l -lea bit la intervalul de trellis d al decodorului . Referitor la intervale de tranziţie ale egalizorului, la rata intervalelor de trellis corespunzătoare decodorului $R = k / n$, se observă că numărul intervalelor de egalizare ale egalizorului este $\frac{n}{K_b}$, în care K_b este numărul de biţi folosiţi pentru a reprezenta un simbol.

De exemplu, un interval de tranziţie al decodorului la o rată a decodorului $R = 1/2$ este echivalent cu $\frac{n=2}{k_b=1} = 2$ intervale de egalizare BPSK. De aceea, biţii u_m şi u_{m+1} recepţionaţi la egalizorul BPSK corespund biţilor $c_{1,d}$ şi, $c_{2,d}$ la intervalul d al decodorului. Având definite notaţiile utilizate, pentru a determina valorile LLR ale biţilor sursă şi ale biţilor codificaţi, se va discuta în continuare despre cele două etape principale utilizate.

În primul rând sunt calculate valorile recursivităţii înainte şi înapoi, $A_m(k)$ şi respectiv $B_m(k)$, precum şi tranziţia metrică $\Gamma_m(\acute{k},k)$, care sunt omologii logaritmici de domeniu ai lui $a_m(k)$ $\beta_m(k)$ şi $\gamma_m(\acute{k},k)$. Exemplul pentru calculul acestor valori este prezentat în algoritmul Log MAP din [35], rezultând pentru calculul valorii LLR relaţia (4.9) , iar pentru $\Gamma_m(\acute{k},k)$ relaţia (4.10) . Pentru decodor indicele tranziţiei de trellis m se înlocuieşte cu d .

$$L\left(u_m|y\right) = ln\left(\frac{\sum\limits_{(\acute{k},k)\Rightarrow u_m=+1} P(\acute{k}\wedge k\wedge y)}{\sum\limits_{(\acute{k},k)\Rightarrow u_m=-1} P(\acute{k}\wedge k\wedge y)}\right) = ln\left(\frac{\sum\limits_{(\acute{k},k)\Rightarrow u_m=+1} exp\left(A_{m-1}(\acute{k}) + \Gamma_m(\acute{k},k) + B_m(k)\right)}{\sum\limits_{(\acute{k},k)\Rightarrow u_m=-1} exp\left(A_{m-1}(\acute{k}) + \Gamma_m(\acute{k},k) + B_m(k)\right)}\right) \tag{4.9}$$

$$\Gamma_m(\acute{k},k) = -\frac{1}{2\sigma^2}(y_m - \hat{y}_m)^2 + ln\left(P\left(u_m\right)\right) \tag{4.10}$$

În al doilea rând, pentru a obţine valorile LLR nu numai pentru biţii sursă v_d ,ci şi pentru biţii codificaţi $c_{l,d}$, trebuie modificată relaţia 4.9 pentru decodorul SISO. În mod explicit, pentru rata $\frac{k}{n}$, codorul convoluţional are un număr n de valori LLR generate la fiecare interval de trellis , în timp ce egalizorul binar produce doar o singură valoare LLR la fiecare interval.

Din relaţia 4.9 se observă că la intervalul de trellis al egalizorului m , numărătorul este suma probabilităţilor de tranziţie asociate cu căile cauzate de bitul codat $u_m = +1$, în timp ce numitorul este suma probabilităţilor corespunzătoare tranziţiilor generate de $u_m = -1$. Aceste probabilităţi de tranziţie pot fi scrise ca o sumă de termeni exponenţiali ai căror argumente este suma dintre $A_{m-1}(\acute{k})$, $B_m(k)$, şi $\Gamma_m(\acute{k},k)$. De aceea, la fiecare interval de trellis m al egalizorului, valoarea LLR obţinută este pentru bitul codificat u_m . Cu toate acestea, pentru rata $R = \frac{k}{n}$ a decodorului, este nevoie de valorile LLR pentru un număr n de biţi codificaţi. Prin urmare, pentru a determina valoarea LLR pentru fiecare bit codat $c_{l,d}$ (pentru $l = 1...n$), trebuie să se ia în considerare prima dată ramurile şi valoarea probabilităţilor corespunzătoare (care este însumarea termenului $exp((A_{d-1}(k) + \Gamma_d(\acute{k},k) + B_d(k)))$) pentru care se obţine bitul codat $c_{l,d} = +1$ şi ulterior ramurile care rezultă când bitul codificat este $c_{l,d} = -1$. Prin urmare, relaţia 4.9 este modificată pentru calculul valorilor LLR a biţilor codaţi $c_{l,d}$, sub forma de mai jos:

$$L\left(c_{l,d}\middle|L_p^i\right) = ln\left(\frac{\sum_{(\acute{k},k)\Rightarrow c_{l,d}=+1} exp\left(A_{d-1}(\acute{k}) + \Gamma_d(\acute{k},k) + B_d\left(k\right)\right)}{\sum_{(\acute{k},k)\Rightarrow c_{l,d}=-1} exp\left(A_{d-1}(\acute{k}) + \Gamma_d(\acute{k},k) + B_d\left(k\right)\right)}\right)$$

$$= ln\left(\sum_{(\acute{k},k)\Rightarrow c_{l,d}=+1} exp\left(A_{d-1}(\acute{k}) + \Gamma_d(\acute{k},k) + B_d\left(k\right)\right)\right)$$

$$-ln\left(\sum_{(\acute{k},k)\Rightarrow c_{l,d}=-1} exp\left(A_{d-1}(\acute{k})+\Gamma_d(\acute{k},k)+B_d(k)\right)\right)$$

unde $L\left(c_{l,d}\middle|L_p^i\right)$ este valoarea LLR a bitului codat l la intervalul de trellis d, având în vedere faptul că decodorul a recepţionat valoarea LLR extrinsecă combinată cu a canalului, notată L_p^i.

Se va considera acum exemplul trellisului unui decodor convoluţional SISO pentru o rată R = 1/2, lungimea constrângerii codorului K = 3 RSC, aşa cum este ilustrat în figura 4.8. În acest exemplu în locul unui cod convoluţional nesistematic a fost ales codul RSC. Deoarece rata codului este R = 1/2, cuvântul de cod la ieşire este format din $k=2$ biţi codaţi, $c_{1,d}$ şi $c_{2,d}$, la fiecare interval de trellis d al decodorului. Cuvântul de cod de la ieşire pentru fiecare tranziţie este descris ca $(c_{1,d},c_{2,d})$ în figura 4.8 şi este dat de polinoamele generatoare G_0 şi G_1, care pot fi reprezentate în forma octală ca 7 şi respectiv 5. Pentru a determina valoarea LLR pentru $c_{1,d}$, trebuie să se ia în considerare toate căile care dau bitul codat $c_{1,d}=+1$.

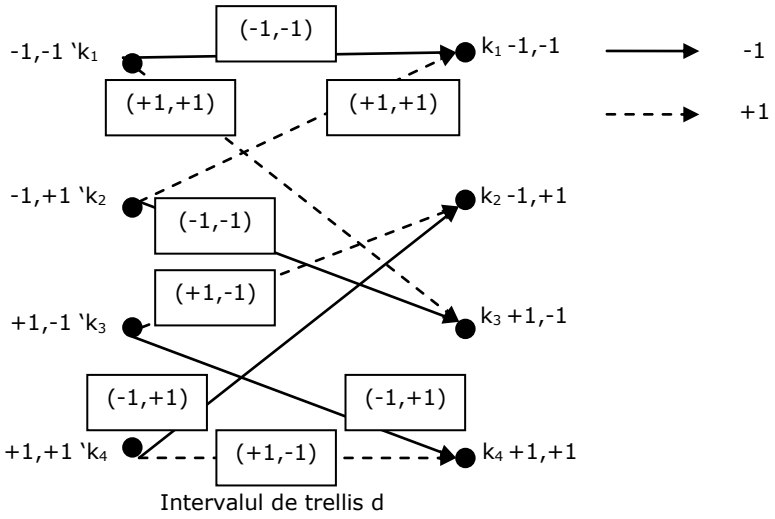


Figura 4.8: Trellisul pentru un decodor convoluţional (K =3 RSC) la intervalul de trellis d al decodorului, unde reprezentarea în octal a polinoamelor generatoare G_0 şi G_1 este de 7 şi respectiv 5.

În figura 4.8 se observă că există patru ramuri care îndeplinesc această cerinţă, şi anume ramurile de stare $\acute{k}_1$ la k_3, $\acute{k}_2$ la k_1, $\acute{k}_3$ la k_2 şi. $\acute{k}_4$ la k_4.

Ţinând cont de relaţia 4.9, se observă că probabilitatea tranziţiei din starea `κ la κ atunci când fost obţinută valoarea L_i^p (valoarea LLR extrinsecă combinată cu a canalului) este dată de relaţia:

$$P(\acute{k} \wedge k \wedge L_p^i) = exp\left(A_{d-1}(\acute{k}) + \Gamma_d(\acute{k}, k) + B_d(k) \right) \tag{4.11}$$

Prin urmare, probabilităţile ramurii individuale asociate bitului codificat $c_{1,d} = +1$ vor fi:

$$\begin{aligned}
P(\acute{k}_1 \wedge k_3 \wedge L_p^i) &= exp\left(A_{d-1}(\acute{k}_1) + \Gamma_d(\acute{k}_1, k_3) + B_d(k_3) \right) = exp\left(x(\acute{k}_1, k_3) \right) \\
P(\acute{k}_2 \wedge k_1 \wedge L_p^i) &= exp\left(A_{d-1}(\acute{k}_2) + \Gamma_d(\acute{k}_2, k_1) + B_d(k_1) \right) = exp\left(x(\acute{k}_2, k_1) \right) \\
P(\acute{k}_3 \wedge k_2 \wedge L_p^i) &= exp\left(A_{d-1}(\acute{k}_3) + \Gamma_d(\acute{k}_3, k_2) + B_d(k_2) \right) = exp\left(x(\acute{k}_3, k_2) \right) \\
P(\acute{k}_4 \wedge k_4 \wedge L_p^i) &= exp\left(A_{d-1}(\acute{k}_4) + \Gamma_d(\acute{k}_4, k_4) + B_d(k_4) \right) = exp\left(x(\acute{k}_4, k_4) \right)
\end{aligned} \tag{4.12}$$

unde $x(\acute{k}_i, k_j)$ reprezintă suma $A_{d-1}(\acute{k}_i) + \Gamma_d(\acute{k}_i, k_j) + B_d(k_j)$. Expresia lui $\Gamma_d(\acute{k}, k)$ pentru decodor poate fi simplificată din relaţia 4.10, observându-se că termenul $\Gamma_d(\acute{k}, k)$ este legat de pătratul distanţei euclidiene dintre eşantionul semnalului recepţionat y_m şi semnalul aşteptat $\hat{y}_m$, care este ulterior normalizat cu $2\sigma^2$. Expresia 4.10 poate fi extinsă la:

$$\Gamma_m(\acute{k}, k) = -\frac{1}{2\sigma^2}(y_m^2 - 2y_m \cdot \hat{y}_m + \hat{y}_m^2) + ln\left(P(u_m)\right) \tag{4.13}$$

Semnale $\hat{y}_m$ aşteptate la intrarea decodorului sunt biţii codificaţi de valori +1, şi respectiv -1. În consecinţă, pătratul semnalului aşteptat $\hat{y}_m$ este întotdeauna unitar, şi prin urmare poate fi ignorat. În plus, întrucât aceeaşi valoare y_m este luată în considerare pentru fiecare tranziţie de trellis, aceasta poate fi neglijată în timpul procesului, potrivit modelului asociat, lăsând metrica decodorului $\Gamma_d(\acute{k}, k)$, să fie dependentă de termenul cross-corelaţiei $\left(2 \cdot y_m \cdot \hat{y}_m\right) / 2\sigma^2$. La intrarea decodorului este recepţionat L_i^p conţinând contribuţia lui $2\sigma^2$ [33], în timp ce la nivel local semnalele aşteptate generate sunt asociate cu tranziţiile de trellis din cauza biţilor codaţi $c_{l,d}$, pentru $l = 1.....n$, unde n este numărul de biţilor

de cod. Prin urmare, metrica decodorului $\Gamma_d(\acute{k},k)$ la intervalul de trellis d al decodorului poate fi exprimat ca [33]:

$$\Gamma_m(\acute{k},k)=\sum_{l=1}^{n=2}\left(\frac{1}{2}L_p^i\cdot c_{l,d}\right)+ln\left(P\left(v_d\right)\right) \tag{4.14}$$

unde $P\left(v_d\right)$ este probabilitatea a-priori a bitului sursa v_d . După calcularea lui $\Gamma_d(\acute{k},k)$, pot fi determinaţi termenii $A_{d-1}(\acute{k})$ şi $B_d(k)$. Prin urmare, probabilitatea ca $c_{l,d}=+1$ să fi fost transmis la intervalul d ştiind că a fost recepţionat L_i^p este determinată de suma probabilităţilor ramurii individuale asociate lui $c_{l,d}=+1$ şi anume:

$$P(\acute{k}_1\wedge k_3\wedge L_p^i),\quad P(\acute{k}_2\wedge k_1\wedge L_p^i),\quad P(\acute{k}_3\wedge k_2\wedge L_p^i)\quad şi\quad P(\acute{k}_4\wedge k_4\wedge L_p^i)$$

Similar, trebuie luate în considerare toate căile din figura 4.8, care dau $c_{l,d}=-1$, şi anume ramurile de la starea $\acute{k}_1$ la k_1, $\acute{k}_2$ la k_3 , $\acute{k}_3$ la k_4 şi. $\acute{k}_4$ la k_2 . Cu aceasta observaţie, şi cu ajutorul figurii 4.8 se calculează probabilităţile ramurii individuale asociate cu bitul codificat $c_{l,d}=-1$, după cum urmează:

$$\begin{aligned}
P(\acute{k}_1\wedge k_1\wedge L_p^i)&=exp\left(A_{d-1}(\acute{k}_1)+\Gamma_d(\acute{k}_1,k_1)+B_d\left(k_1\right)\right)=exp\left(x(\acute{k}_1,k_1)\right)\\
P(\acute{k}_2\wedge k_3\wedge L_p^i)&=exp\left(A_{d-1}(\acute{k}_2)+\Gamma_d(\acute{k}_2,k_3)+B_d\left(k_3\right)\right)=exp\left(x(\acute{k}_2,k_3)\right)\\
P(\acute{k}_3\wedge k_4\wedge L_p^i)&=exp\left(A_{d-1}(\acute{k}_3)+\Gamma_d(\acute{k}_3,k_4)+B_d\left(k_4\right)\right)=exp\left(x(\acute{k}_3,k_4)\right)\\
P(\acute{k}_4\wedge k_2\wedge L_p^i)&=exp\left(A_{d-1}(\acute{k}_4)+\Gamma_d(\acute{k}_4,k_2)+B_d\left(k_2\right)\right)=exp\left(x(\acute{k}_4,k_2)\right)
\end{aligned} \tag{4.15}$$

Ca şi mai înainte, se poate calcula probabilitatea ca $c_{l,d}=-1$ să fi fost transmis la intervalul de trellis d, dat fiind faptul că a fost recepţionat L_i^p prin luarea sumei probabilităţilor ramurii individuale a:

$$P(\acute{k}_1\wedge k_1\wedge L_p^i),\quad P(\acute{k}_2\wedge k_3\wedge L_p^i),\quad P(\acute{k}_3\wedge k_4\wedge L_p^i)\quad şi\quad P(\acute{k}_4\wedge k_2\wedge L_p^i)$$

Cu ajutorul relaţiilor 4.4 şi folosind relaţiile 4.11, 4.12 şi 4.15, se poate exprima valoarea LLR pentru primul bit codat $c_{l,d}$ ca:

$$L\left(c_{I,d}\middle|L_p^i\right)=ln\left(\frac{\sum_{(\acute{k},k)\Rightarrow c_{I,d}=+1}P(\acute{k}\wedge k\wedge L_p^i)}{\sum_{(\acute{k},k)\Rightarrow c_{I,d}=-1}P(\acute{k}\wedge k\wedge L_p^i)}\right)$$

$$=ln\left(\frac{P(\acute{k}_1\wedge k_3\wedge L_p^i)+P(\acute{k}_2\wedge k_1\wedge L_p^i)+P(\acute{k}_3\wedge k_2\wedge L_p^i)+P(\acute{k}_4\wedge k_4\wedge L_p^i)}{P(\acute{k}_1\wedge k_1\wedge L_p^i)+P(\acute{k}_2\wedge k_3\wedge L_p^i)+P(\acute{k}_3\wedge k_4\wedge L_p^i)+P(\acute{k}_4\wedge k_2\wedge L_p^i)}\right)$$

$$=ln\left(exp\left(x(\acute{k}_1,k_3)\right)+exp\left(x(\acute{k}_2,k_1)\right)+exp\left(x(\acute{k}_3,k_2)\right)+exp\left(x(\acute{k}_4,k_4)\right)\right)$$

$$-ln\left(exp\left(x(\acute{k}_1,k_1)\right)+exp\left(x(\acute{k}_2,k_3)\right)+exp\left(x(\acute{k}_3,k_4)\right)+exp\left(x(\acute{k}_4,k_2)\right)\right) \quad (4.16)$$

Întrucât termenul $L(c_{1,d}\middle|L_i^p)$ poate fi exprimat ca logaritmul natural al unei sume de exponenţiale, o relaţie generalizată care foloseşte logaritmul Jacobian (4.17)

$$ln\left(\sum_{k=1}^{V}e^{x_k}\right)=J\left(x_V,J\left(x_{V-1},\ldots,J\left(x_3,J\left(x_2,x_1\right)\right)\right)\right) \quad (4.17)$$

poate fi utilizată, pentru a calcula valoarea LLR, a biţilor codaţi $c_{1,d}$,la intervalul de trellis d. Prin utilizarea aceleaşi abordări ca şi mai sus, se poate demonstra că valoarea LLR a bitului codat $c_{2,d}$ este dată de:

$$L\left(c_{2,d}\middle|L_p^i\right)=ln\left(\frac{\sum_{(\acute{k},k)\Rightarrow c_{2,d}=+1}P(\acute{k}\wedge k\wedge L_p^i)}{\sum_{(\acute{k},k)\Rightarrow c_{2,d}=-1}P(\acute{k}\wedge k\wedge L_p^i)}\right) \quad (4.18)$$

$$=ln\left(\frac{P(\acute{k}_1\wedge k_3\wedge L_p^i)+P(\acute{k}_2\wedge k_1\wedge L_p^i)+P(\acute{k}_3\wedge k_4\wedge L_p^i)+P(\acute{k}_4\wedge k_2\wedge L_p^i)}{P(\acute{k}_1\wedge k_1\wedge L_p^i)+P(\acute{k}_2\wedge k_3\wedge L_p^i)+P(\acute{k}_3\wedge k_2\wedge L_p^i)+P(\acute{k}_4\wedge k_4\wedge L_p^i)}\right)$$

$$=ln\left(exp\left(x(\acute{k}_1,k_3)\right)+exp\left(x(\acute{k}_2,k_1)\right)+exp\left(x(\acute{k}_3,k_4)\right)+exp\left(x(\acute{k}_4,k_2)\right)\right)$$

$$-ln\left(exp\left(x(k_1',k_1)\right)+exp\left(x(k_2',k_3)\right)+exp\left(x(k_3',k_2)\right)+exp\left(x(k_4',k_4)\right)\right)$$

Este apoi utilizată relaţia 4.17 (relaţia generalizată care foloseşte logaritmul Jacobian) pentru a calcula suma termenilor exponenţiali în relaţiile 4.16 şi 4.18, pentru a obţine valorile LLR $L(c_{2,d}|L_p^i)$ ale bitului codat $c_{2,d}$, la intervalul d de trellis.

4.4 Exemplu de egalizare turbo

În scopul de a evidenţia principiul de egalizare, se consideră un sistem simplu care utilizează o codare convoluţională şi o modulaţie BPSK. Acesta transmite pe trei căi şi utilizează la receptor un egalizor turbo, aşa cum se arată în figura 4.10. Simbolul spaţiat în canalul static este detaliat în figura 4.9. Codorul foloseşte polinoame generatoare octale $G_0 = 7$ şi $G_1 = 5$ la o rată R = 0,5, lungimea constrângerii fiind K = 3, RSC. Structura de trellis corespunzătoare decodorului şi tranziţiile sunt ilustrate în figura 4.8. Această schemă a egalizorului turbo, conţinând un egalizor de canal bazat pe trellis şi un decodor de canal, a fost prima dată introdusă în figura 4.2. Întrucât dispersia maximă a răspunsului la impuls al canalului este două perioade de bit ($\tau_d = 2$), trellisul bazat pe egalizarea BPSK necesită patru stări ($2^{\tau_d=2} = 4$), aşa cum se vede în figura 4.11. Fiecare dintre aceste patru stări poate fi privită ca starea unui registru de deplasare care posedă două elemente de memorie. Prin urmare aceste stări pot fi reprezentate de doi biţi. Dacă se ia în considerare starea 1 din figura 4.11 se observă că aceasta corespunde biţilor 1 şi 0 în registrul deplasare, unde bitul 0 este primul bit. Pentru un bit de intrare de 1, bitul 0 este deplasat, în timp ce bitul de intrare 1 este memorat. În consecinţă, noua stare în care s-a ajuns este starea 3, întrucât registrul de deplasare conţine biţii 1 şi 1. Aplicarea aceluiaşi raţionament, pentru oricare dintre toate celelalte tranziţii de stare legitime produce structura de trellis din figura 4.11.

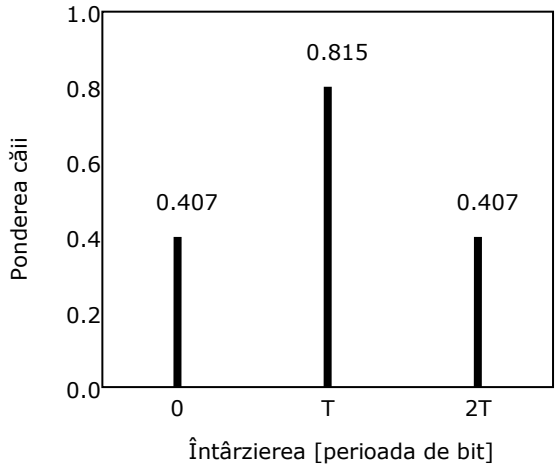


Figura 4.9 Simbolul spaţiat în canalul static cu trei căi

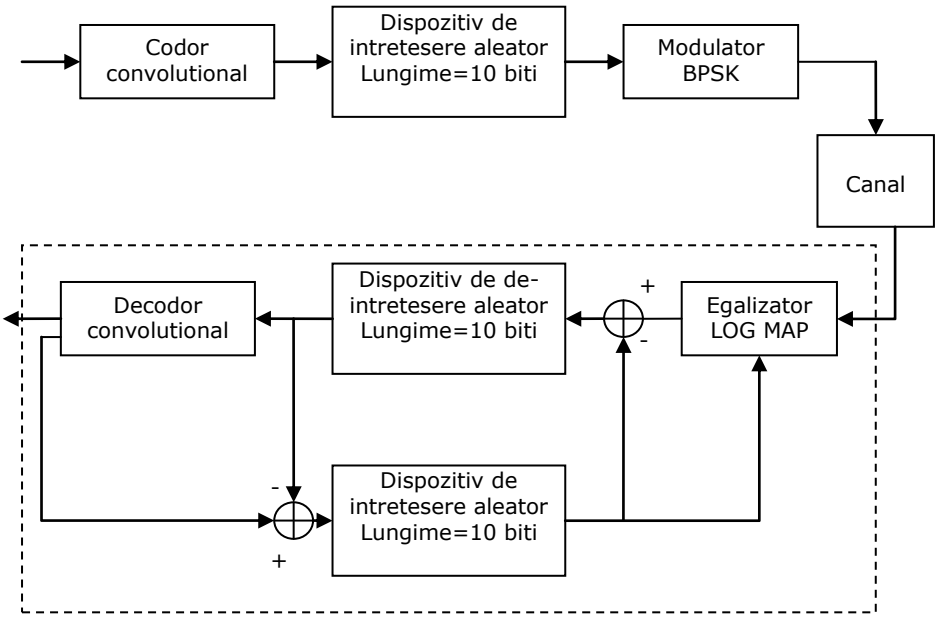


Figura 4.10: Schema unui sistem utilizând codarea convoluţională, modulaţia BPSK şi schema egalizorului turbo din figura 4.2.

În acest exemplu, secvenţa transmisă este formată din 10 biţi de date şi încă 3 biţi terminali, cu interleaver aleator de 10 biţi.

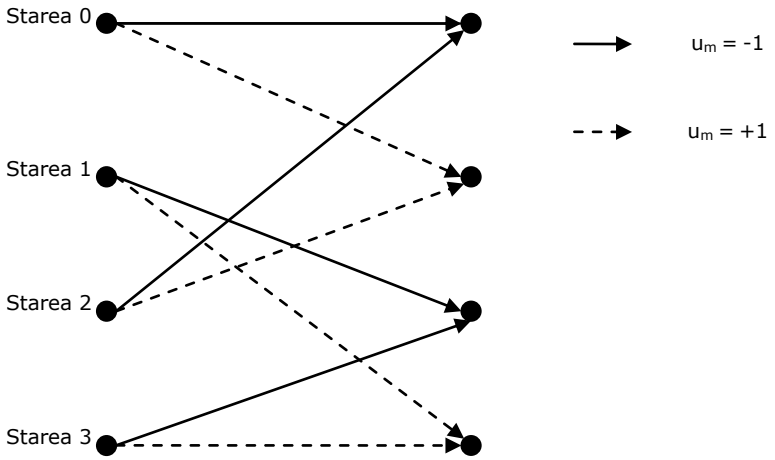


Figura 4.11. Reprezentarea tranziţiilor legitime pe trellis ale egalizorului

În continuare se va descrie funcţionarea egalizorului turbo, începând cu generarea de către egalizor a valorii LLR a-posteriori, care apoi este prelucrată pentru a extrage informaţiile extrinseci combinate cu ale canalului pentru a fi

trimise la decodor. Ulterior, se arată modul în care aceste informaţii extrinseci combinate cu ale canalului sunt utilizate de către decodor pentru a genera valoarea LLR a-posteriori, care, ca şi în egalizor este procesată pentru a obţine informaţia extrinsecă, înainte ca aceasta să treacă înapoi la egalizor. Pentru simplitate, toate valorile cu virgulă au fost rotunjite la o zecimală. Egalizarea canalului bazată pe trellis foloseşte algoritmul Log-MAP. Pentru a calcula valoarea LLR a bitului u_m, valorile $\Gamma_d(\acute{k},k)$, $A_{m-1}(\acute{k})$ şi $B_m(k)$ trebuie să fie evaluate şi utilizate în comun. Acest lucru va fi demonstrat numeric în următorul exemplu. În acest exemplu, cei cinci biţi sursă şi corespunzător cei zece biţi codaţi convoluţional sunt specificaţi în Tabelul 4.1.

Intervalul de trellis al decodorului (d)	Biţii sursă	Biţii codaţi	Biţii codaţi întreţesuţi
1	+1	+1, +1	+1, -1
2	+1	+1, -1	+1, -1
3	-1	-1, -1	+1, +1
4	-1	-1, +1	-1, +1
5	+1	+1, -1	-1, -1

Tabelul 4.1: Biţii sursă transmişi şi biţii codificaţi corespunzători produşi de un codor convoluţional recursiv la o rată R=1/2 si K=3 folosind polinoamele generatoare octale $G_0 = 7$ şi $G_1 = 5$

Ulterior, aceşti biţi codificaţi sunt rearanjaţi cu ajutorul unui bloc de întreţesere numit interleaver de canal de mărime 2 × 5, rezultând noua secvenţă, detaliată în tabelul 4.1. Aceşti biţi întreţesuţi sunt modulaţi BPSK înainte de a fi transmişi prin canalul static cu trei căi din figura 4.9.

Zgomotul termic aditiv, alb, gaussian afectează semnalul recepţionat. Semnalul este eşantionat şi stocat într-un buffer, până când toţi cei zece biţi codaţi şi ultimii trei biţi (biţii terminali) ajung la receptor. În această etapă, se poate începe să se calculeze metrica tranziţiei $\Gamma_m(\acute{k},k)$ pentru toate intervalele de trellis. Figura 4.12 ilustrează tranziţiile în primele trei intervale de trellis. Începând de la starea 0 la intervalul de trellis $m = 1$, valoarea iniţială $A_0(0)$ este de 0.0, în timp ce valorile $\Gamma_1(0,0)$ şi $\Gamma_1(0,1)$ sunt calculate folosind următoarea relaţie (4.19):

$$\Gamma_m(\acute{s},s) = -\frac{1}{2\sigma^2}(y_m - \hat{y}_m)^2 + ln\left(P\left(u_m\right)\right) \qquad (4.19)$$

În relaţia 4.19 y_m, $\hat{y}_m$, $P\left(u_m\right)$ reprezintă eşantionul semnalului depreciat recepţionat din canal, semnalul recepţionat estimat şi respectiv informaţia a-priori a biţilor codificaţi. În prima iteraţie, informaţia a-priori nu este trimisă înapoi de decodorul de canal la egalizor. Prin urmare, termenul $P\left(u_m\right)$ este 1/2, indicând faptul că biţii 1 şi -1 au probabilitate egală de a se produce. Termenul $A_m(k)$ din relaţia 4.9 se calculează cu ajutorul relaţiei de mai jos (4.20), fiind dependent de $A_{m-1}(\acute{k})$ şi $\Gamma_m(\acute{k},k)$:

$$A_m(k) = ln\left(\sum_{toţi\ \acute{k}} exp\left[A_{m-1}(\acute{k}) + \Gamma_m(\acute{k},k) \right] \right) \tag{4.20}$$

În felul acesta pot fi determinate noile valori pentru $A_1(0)$ şi $A_1(1)$. Considerăm tranziţia de la starea 0 la starea 0. Din moment ce valorile termenilor $\Gamma_1(0,0) = -0.3$ şi $A_0(0) = 0.0$, termenul $A_1(0)$ devine:

$$A_{m=1}(k=0) = ln\left(exp\left[A_{m=1-1}(\acute{k}=0) + \Gamma_{m=1}(\acute{k}=0, k=0) \right] \right) \tag{4.21}$$

$$A_1(0) = 0.0 + (-0.3) = -0.3$$

în timp ce $A_1(1)$ este:

$$A_{m=1}(k=1) = ln\left(exp\left[A_{m=1-1}(\acute{k}=0) + \Gamma_{m=1}(\acute{k}=0, k=1) \right] \right) \tag{4.22}$$

$$A_1(1) = 0.0 + (-0.5) = -0.5$$

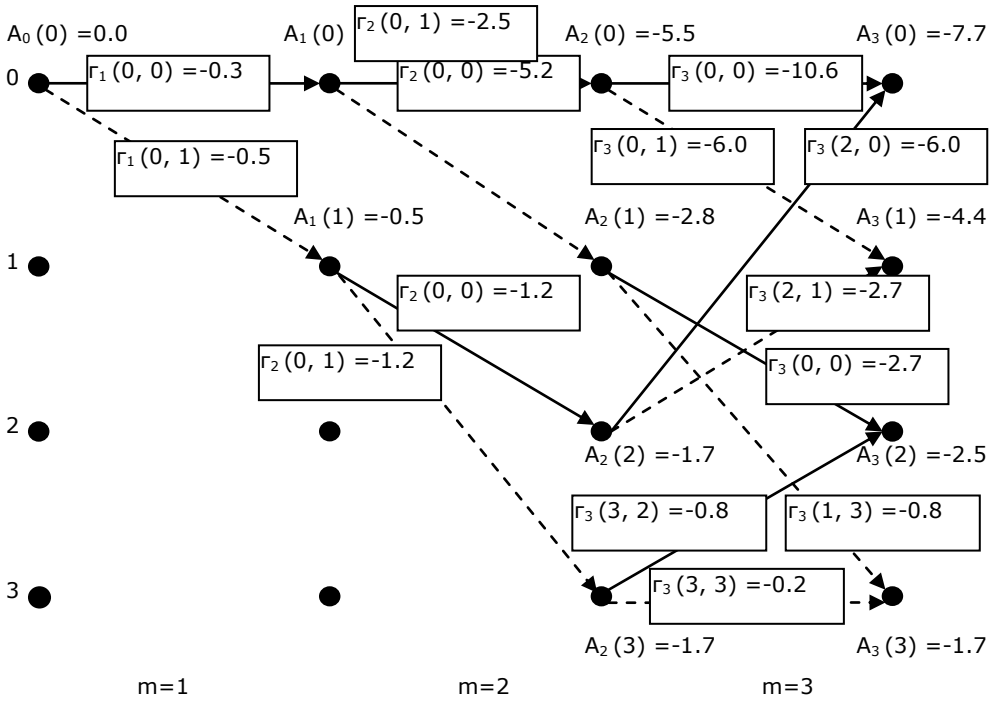


Figura 4.12 Calcularea termenului $A_m(k)$ pentru egalizorul turbo din figura 4.10 prin recursivitate înainte utilizând modelul de canal din figura 4.9 şi o serie de valori $\Gamma_m(\acute{k},k)$ care au fost calculate.

La următorul interval de trellis, pentru $m = 2$, se repetă acelaşi proces de actualizare. Cu toate acestea, în al treilea interval există mai mult de o tranziţie prin care se ajunge la o stare particulară. Dacă se ia în considerare starea 2, în cazul în care există, fuzionează tranziţiile de la starea 1 şi starea 3. În această situaţie, noua valoare a lui $A_3(2)$ pentru starea 2 este:

$$A_{m=3}(k=2) = ln\left(exp\left[A_{m=2}(\acute{k}=1) + \Gamma_{m=3}(\acute{k}=1, k=2) \right] \right.$$
$$\left. + exp\left[A_{m=2}(\acute{k}=3) + \Gamma_{m=3}(\acute{k}=3, k=2) \right] \right)$$

$$A_3(2) = ln\left(exp(-2,8-2,7) + exp(-1,7-0,8)\right)$$
$$= ln\left(exp(-5,5) + exp(-2,5)\right)$$

care pot fi calculate folosind logaritmul Jacobian [27] şi un tabel, pentru a evita transformarea logaritmului natural şi calculele exponenţiale, obţinându-se:

$$ln\left(exp(-5,5) + exp(-2,5)\right) = max\left(-5,5; -2,5\right) + \underbrace{ln\left(1 + exp\left(-\left|-5,5-(-2,5)\right|\right)\right)}_{\text{din tabel } f_{diff}=\left|-5,5-(-2,5)\right|=3}$$
$$= -2,5 + 0,05 = -2,45$$

Intervale pentru f_{diff}	$\ln(1 + \exp(-f_{diff}))$
$f_{diff} > 3.70$	0.00
$3.70 \geq f_{diff} > 2.25$	0.05
$2.25 \geq f_{diff} > 1.50$	0.15
$1.50 \geq f_{diff} > 1.05$	0.25
$1.05 \geq f_{diff} > 0.70$	0.35
$0.70 \geq f_{diff} > 0.43$	0.45
$0.43 \geq f_{diff} > 0.20$	0.55
$f_{diff} \leq 0.20$	0.65

Tabelul 4.2 Valorile logaritmului pentru diferite valori ale f_{dif}

Din tabelul 4.2 pentru $f_{diff} = 3$ se obţine 0.05 [27]. Aceeaşi operaţie se realizează pentru toate celelalte stări din întregul trellis, până când toate valorile A_m sunt obţinute. Tabelul 4.2 este utilizat cu scopul de a evita logaritmul natural şi calcule exponenţiale. Valoarea f_{diff} este valoarea absolută a diferenţei dintre argumentele funcţiei exponenţiale.

Calcularea termenului $B_m(k)$ de recursivitate înapoi, implică o abordare similară. Punctul de plecare este la sfârşitul trellisului, la $m = 13$. Iniţial, valorile $B_{m=13}(k)$ la intervalul de trellis $m = 13$ sunt stabilite la 0.0 - figura 4.13. De asemenea, se observă că restul de trei biţi terminali cu valoarea -1 au fost inseraţi in secvenţa transmisă la intervale de trellis $m = 11$, $m = 12$ şi $m = 13$, singurele tranziţii legitime la aceste intervale sunt datorate bitului de -1. Prin urmare,

tranziţiile corespunzătoare transmisiei unui bit +1 sunt nelegitime şi, prin urmare, nu sunt luate în considerare.

Pentru a calcula valorile $B_{m=12}(k)$, se utilizează relaţia de mai jos:

$$B_{m-1}(\acute{k}) = ln\left(\sum_{toţi\ k} exp\left[B_m(k) + \Gamma_m(\acute{k},k)\right]\right) \tag{4.23}$$

De exemplu, pentru a determina valoarea $B_{m-1=12}(k)\ (\acute{k}=0)$ la $m=13$ sunt necesare numai valorile $B_{m=13}(k)\ (\acute{k}=0)$ şi $\Gamma_{m=13}(k)\ (\acute{k}=0,k=0)$, obţinându-se:

$$B_{m-1=12}(\acute{k}=0) = ln\left(exp\left[B_{m=13}(k=0) + \Gamma_{m=13}(\acute{k}=0,k=0)\right]\right)$$
$$= ln\left(exp\left(0,0+57,5\right)\right) = 57,5$$

Acest lucru se datorează faptului că există o singură cale legitimă, cauzată de bitul terminal -1. După cum se vede în figura 4.13, acest proces de recursivitate înapoi se repetă cu scopul de a determina valorile $B_{m-1=11}(\acute{k})$ corespunzătoare stărilor $\acute{k}=0.....3$ la intervalul $m=12$, cu toate că din cauza ultimilor trei biţi -1 numai $B_{m-1=11}(\acute{k}=0)$ şi $B_{m-1=11}(\acute{k}=2)$ trebuie să fie calculate. În această etapă, nu se utilizează relaţia 4.17, deoarece există o singură cale generată de bitul -1, care poate fi luată în considerare, la stările $(\acute{k}=0)$ şi $(\acute{k}=2)$.

În consecinţă, termenul $B_{m-1}(\acute{k})$, în relaţia 4.23 este exprimat ca logaritmul natural al unui singur termen exponenţial. La intervalul de trellis $m=10$ sunt, de asemenea, luate în considerare căile asociate cu bitul +1 ca în figura 4.13, deoarece aceste tranziţii nu mai sunt determinate de biţii terminali -1.

Având în vedere starea 0, se observă că există două tranziţii care părăsesc această stare în timpul intervalului de trellis $m=10$, unde prima ajunge la starea 0, iar cealaltă la starea 1. Prin urmare, probabilităţile acestor tranziţii care pornesc din starea 0 în intervalul $m=10$ sunt însumate rezultând:

$$B_{m-1=9}(\acute{k}=0) = ln\left(exp\left[B_{m=10}(k=0) + \Gamma_{m=10}(\acute{k}=0,k=0)\right]\right.$$
$$\left. + exp\left[B_{m=10}(k=1) + \Gamma_{m=10}(\acute{k}=0,k=1)\right]\right.$$
$$= ln\left(exp\left(170,7+\left(-4,6\right)\right) + exp\left(162,9+\left(-1,8\right)\right)\right)$$

$$= max(166,1;161,1) + \underbrace{ln\left(1+exp\left(-|166,1-161,1|\right)\right)}_{\text{din tabel } f_{diff}=|166,1-161,1|=5}$$

$$= 166,1+0,00 = 166,1$$

unde din nou a fost folosită relaţia generalizată 4.17, pentru a simplifica calculul logaritmului natural, al unei sume de exponenţiale, la o operaţie care implică un proces de maximizare şi o operaţie cu tabelul 4.2 [27].

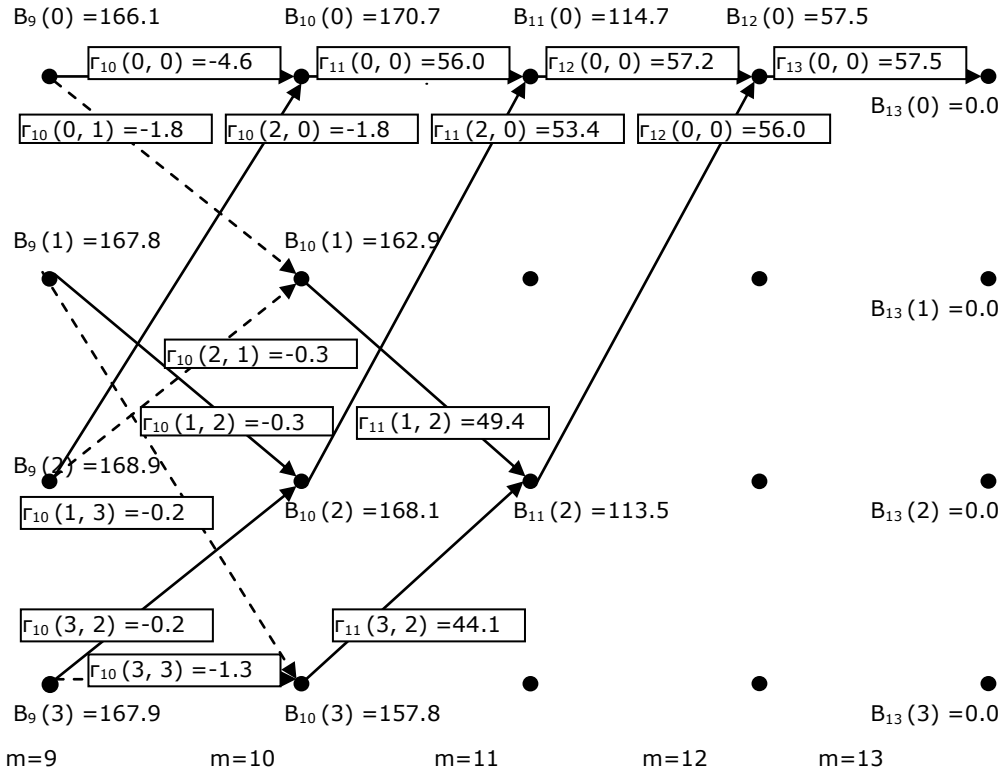


Figura 4.13. Calcularea termenului $B_m(\acute{k})$ pentru egalizorul turbo din figura 4.10 prin recursivitate înapoi utilizând modelul de canal din figura 4.9 şi o serie de valori $\Gamma_m(\acute{k},k)$ care au fost calculate.

Prin repetarea recursivităţii înapoi de mai sus pot fi determinate toate valorile $B_m(k)$ pentru stările $k = 0.....3$ şi intervale $m = 0.....13$, dintre care unele au fost calculate pentru a fi ilustrate, în figura 4.13.

Având determinate valorile $A_m(k)$, $B_m(k)$ pentru toate stările şi valorile $\Gamma(\acute{k},k)$ asociate cu toate tranziţiile, poate fi calculată valoarea LLR corespunzătoare bitului u_m utilizând relaţia 4.9.

Acest lucru presupune identificarea tranziţiilor de trellis cauzate de bitul $u_m = -1$ şi însumarea probabilităţilor de tranziţie a fiecărei căi. Deoarece, conform relaţiei de mai jos 4.24

$$L(u_m|y) \triangleq ln\left(\frac{P(u_m = +1|y)}{P(u_m = -1|y)}\right) \tag{4.24}$$

probabilităţile asociate pot fi exprimate ca o sumă de termeni exponenţiali, valoarea LLR fiind logaritmul natural al unei sume de valori exponenţiale. Prin urmare, pentru a reduce complexitatea asociată cu calcul exponenţialelor şi logaritmilor, pot fi din nou utilizate relaţia generalizată 4.17 şi tabelul 4.2 .

Figura 4.14 (a) ilustrează setul de tranziţii la intervalul de trellis $m = 6$ cauzate de $u_m = -1$, în timp ce figura 4.14 (b) descrie setul de tranziţii corespunzătoare unui bit $u_m = +1$, care au fost extrase din trellisul generic, după cum se vede şi în relaţiile 4.12.

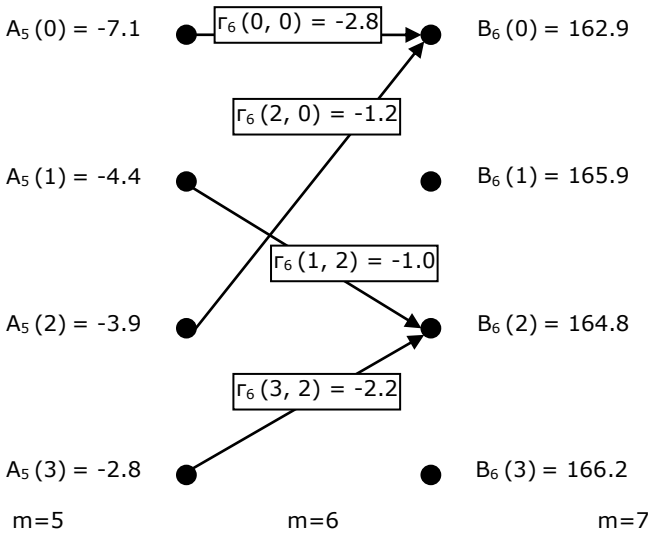
Considerând acest interval de trellis ca un exemplu, logaritmul natural al probabilităţii de tranziţie asociate cu bitul $u_m = -1$ este dat de logaritmul natural al numitorului în relaţia (4.24), rezultând:

$$ln\left[P(u_m = -1|y)\right] = ln[exp\left(A_5(0) + \Gamma_6(0,0) + B_6(0)\right) + exp\left(A_5(1) + \Gamma_6(1,2) + B_6(2)\right)$$

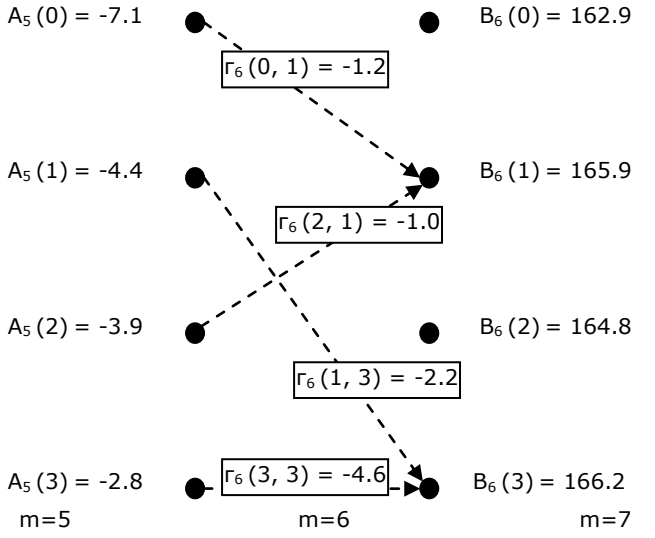
$$+ exp\left(A_5(2) + \Gamma_6(2,0) + B_6(0)\right) + exp\left(A_5(3) + \Gamma_6(3,2) + B_6(2)\right) \tag{4.25}$$

Deoarece relaţia (4.25) este logaritmul natural al sumei de patru termeni exponenţiali, relaţia generalizată care foloseşte logaritmul Jacobian (4.17) este aplică recursiv, rezultând:

$$\begin{aligned} ln\left[P(u_m = -1|y)\right] &= ln[exp\left(-7,1 + (-2,8) + 162,9\right) + exp\left(-4,4 + (-1,0) + 164,8\right) \\ &\quad + exp\left(-3,9 + (-1,2) + 162,9\right) + exp\left(-2,8 + (-2,2) + 164,8\right)] \\ &\approx ln[exp\left(\overbrace{\underbrace{max(153,0;159,4) + 0,0}_{TAB}}^{=159,4}\right) + exp\left(-3,9 + (-1,2) + 162,9\right) \\ &\quad + exp\left(-2,8 + (-2,2) + 164,8\right)] \\ &\approx ln\left[exp\left(\overbrace{\underbrace{max(159,4;157,8) + 0,15}_{TAB}}^{=159,55}\right) + exp\left(-2,8 + (-2,2) + 164,8\right)\right] \\ &\approx max(159,55;159,8) + \underbrace{0,55}_{TAB} \approx 160,35 \end{aligned} \tag{4.26}$$



(a) Tranziţiile corespunzătoare biţilor $u_m = -1$



(b) Tranziţiile corespunzătoare biţilor $u_m = +1$

Figura 4.14 Tranziţiile de trellis ale egalizorului, care au fost luate în considerare atunci când calculăm valorile LLR corespunzătoare biţilor -1 şi +1 la intervalul de trellis m=6

În mod similar sunt grupate împreună căile corespunzătoare bitului $u_m = +1$, iar cu figura 4.14 (b) se va obţine logaritmul natural al probabilităţii de a recepţiona bitul $u_m = +1$ având în vedere faptul că a fost transmis y:

$$\ln\left[P\left(u_m=+1|y\right)\right]=\ln[\exp\left(-7,1+\left(-1,2\right)+165,9\right)+\exp\left(-4,4+\left(-2,2\right)+166,2\right)$$
$$+\exp\left(-3,9+\left(-1,0\right)+165,9\right)+\exp\left(-2,8+\left(-4,6\right)+166,2\right)]$$
$$\approx \ln[\exp\left(\overbrace{\max\left(157,6;159,6\right)+\underbrace{0,15}_{TAB}}^{=159,75}\right)+\exp\left(-3,9+\left(-1,0\right)+165,9\right)$$
$$+\exp\left(-2,8+\left(-4,6\right)+166,2\right)]$$
$$\approx \ln\left[\exp\left(\overbrace{\max\left(159,75;161,0\right)+\underbrace{0,25}_{TAB}}^{=161,25}\right)+\exp\left(-2,8+\left(-4,6\right)+166,2\right)\right]$$
$$\approx \max\left(161,25;158,8\right)+\underbrace{0,05}_{TAB}\approx 161,3 \qquad (4.27)$$

Conform relaţiei 4.24

$$L(u_m|y)\ln\left(\frac{P\left(u_m=+1|y\right)}{P\left(u_m=-1|y\right)}\right)=\ln P\left(u_m=+1|y\right)-\ln P\left(u_m=-1|y\right)$$

Prin urmare, cu ajutorul rezultatelor obţinute din relaţiile 4.26 şi 4.27 valoarea LLR a posteriori a bitului u_m la intervalul de trellis $m=6$ este:
161,3 – 160,35 = 0.95.

Prin aplicarea aceluiaşi raţionament la toate stările de trellis, pot fi determinate valorile LLR a posteriori a tuturor celor 13 biţi u_m.

În tabelul 4.3 se prezintă biţii codaţi şi transmişi pe canal, care au fost întreţesuţi, obţinând secvenţa (TXD-C). Sunt de asemenea prezentate în tabel valorile LLR a-priori (APRI), valorile LLR a-posteriori (APOS), valorile LLR extrinseci combinate cu ale canalului (EXT) şi biţi de decizie (DCSN) din prima iteraţie a egalizării turbo.

	Intervalul de trellis *m* al egalizorului												
	1	2	3	4	5	6	7	8	9	10	11	12	13
APRI	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
APOS	2,7	2,1	-0,5	1,1	-0,1	1,0	-0,6	0,2	-0,4	-6,5	-123	-199,9	-116,2
EXT	2,7	2,1	-0,5	1,1	-0,1	1,0	-0,6	0,2	-0,4	-6,5	-123	-199,9	-116,2
TXD-C	1	-1	1	-1	1	1	-1	1	-1	-1	-1	-1	-1
DCSN	1	**1**	**-1**	**1**	**-1**	1	-1	1	-1	-1	-1	-1	-1

Tabelul 4.3. Diferite metrici de egalizare extrase din simulări după prima iteraţie

În prima iteraţie egalizorul nu recepţionează informaţie a-priori şi prin urmare valorile din tabel sunt 0,0. Din valorile LLR a-posteriori calculate, bitul detectat este determinat cu ajutorul unui prag de zero, adică se va lua o decizie de +1 atunci când valoarea LLR este ≥ 0 şi -1 când valoarea LLR este <0. Valoarea APOS de 1.0 la m = 6 a fost determinată prin rotunjirea valorii de 0,95 calculate anterior. În tabelul 4.3 se observă că au fost făcute patru erori, care sunt afişate cu bold. În această etapă, s-a arătat cum este determinată valoarea LLR corespunzătoare bitului u_m de egalizorul Log-MAP, folosind exemplul unui sistem simplu BPSK codat convoluţional. În etapa următoare, valoarea LLR generată de egalizor (EXT) este trimisă la decodorul de canal.

După cum s-a menţionat anterior, o regulă esenţială pentru egalizarea turbo este că informaţiile de intrare pentru un bloc particular în iteraţia curentă nu trebuie să includă informaţiile care au contribuit la acel bloc particular în iteraţia anterioară, deoarece informaţiile utilizate în iteraţii consecutive ar fi dependente unele de altele şi prin urmare, nu s-ar reuşi obţinerea unui câştig în urma iteraţiei.

De aceea valoarea LLR a-posteriori, care conţine informaţiile extrinseci combinate cu ale canalului, plus informaţiile a-priori furnizate de decodorul de canal, trebuie să fie prelucrată pentru a elimina contribuţia anterioară a decodorului (sub forma valorii LLR a-priori) înainte ca aceasta să treacă la decodor. Acest lucru este realizat prin stocarea contribuţiei precedente a decodorului şi scăzând-o din valoarea LLR a-posteriori produsă de egalizor, după cum se vede în figura 4.10.

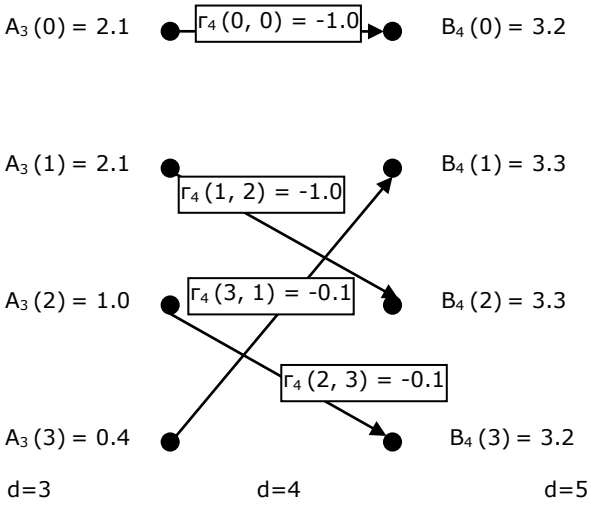
Deoarece în prima iteraţie nu există nici o informaţie a-priori generată de decodor, contribuţiile APRI corespunzătoare din tabelul 4.3 sunt zero şi, prin urmare, valoarea LLR a-posteriori conţinând informaţiile extrinseci combinate cu ale canalului, pot fi transmise direct la decodor. Se observă de asemenea, că egalizorul produce valori LLR, şi pentru ultimii trei biţi (cei trei biţi terminali -1 de la emisie). Deşi aceste valori LLR, ale ultimilor trei biţi, nu sunt cerute de decodor, acestea sunt folosite în procesul de egalizare.

Având descrisă funcţionarea egalizorului Log-MAP, în continuare se va descrie funcţionarea decodorului Log-MAP, în contextul egalizării turbo. După cum se vede în Figura 4.5, decodorul primeşte valoarea LLR extrinsecă combinată cu a canalului $L_p^i = \left\{ L_p^{i;s}, L_p^{i;h} \right\}$, care a fost de-întreţesută. Aici, rolul deinterleaverului este de a minimiza corelaţia între informaţiile de intrare ale egalizorului şi intrarea decodorului. De asemenea, ajută la transmiterea informaţiilor în ordinea corectă la decodor, deoarece anterior interleaverul de canal π_c din figura 4.5 rearanjează biţii în ordinea cerută de egalizor. În acest exemplu se utilizează un codor K = 3 RSC, utilizând polinoame generatore octale $G_0 = 7$ şi $G_1 = 5$ la o rata R = 1/2 . Deoarece lungimea constrângerii este K=3, trellisul decodorului conţine $2^{K-1} = 2^2 = 4$ stări, aşa cum este ilustrat în figura 4.8

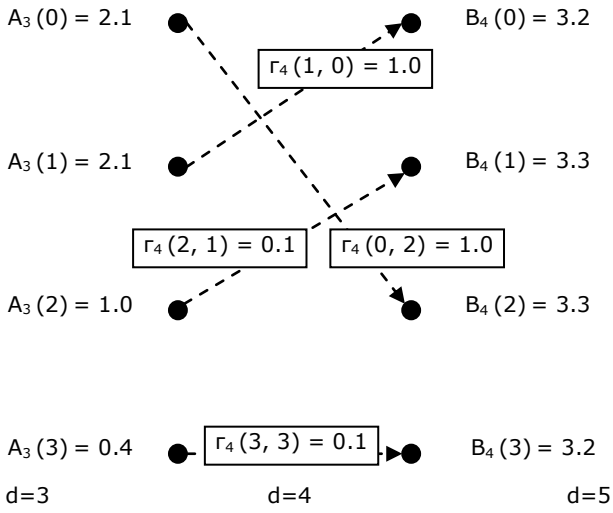
Ca şi în egalizorul Log-MAP, decodorul Log-MAP calculează, valorile $A_m(k)$, $B_m(k)$ şi $\Gamma_m(\acute{k},k)$, pentru toate stările şi respectiv toate tranziţiile de trellis. Calculul lui $\Gamma_m(\acute{k},k)$ care se face folosind relaţia (4.14) poate fi simplificat, deoarece biţii codaţi recepţionaţi şi pătratul biţilor codificaţi aşteptaţi sunt întotdeauna constante pentru toate tranziţiile de trellis.

În comparaţie cu intervalul de trellis al egalizorului care se notează cu indice m, la decodor se va folosi d. Odată ce aceşti parametri sunt calculaţi, sunt

determinate valorile LLR atât pentru biţii sursă cât şi pentru biţii codaţi, faţă de decodarea turbo unde calculul se face doar pentru valorile LLR ale biţilor sursă. De aceea, la fiecare rată $R = \frac{1}{2}$ a trellisului decodorului sunt produse valorile LLR a celor doi biţi codificaţi. În continuare se va arăta cum este determinată valoarea LLR a fiecărui bit codat.



(a) Tranziţiile corespunzătoare biţilor codaţi $c_{1,4} = -1$



(b) Tranziţiile corespunzătoare biţilor codaţi $c_{1,4} = +1$

Figura 4.15 . Tranziţiile de trellis ale decodoarelor de canal cu K=3 RSC şi R=1/2, care au fost luate în considerare atunci când se calculează valorile LLR corespunzătoare bitului $c_{1,4}$, care este primul bit codat la intervalul de trellis d=4. (Trellisul este bazat pe figura 4.8)

În figura 4.15(a) este prezentată o parte din trellisul decodorului la intervalul de trellis $d = 4$, pentru setul ramurilor corespunzătoare primului bit codat $c_{1,4} = -1$, în timp ce în figura 4.15(b) se ilustrează grupul tranziţiilor de trellis cauzate de $c_{1,4} = +1$. Atunci când este utilizat un codor sistematic, valoarea LLR a primilor biţi codificaţi $c_{1,d}$ constituie valoarea LLR a biţilor sursă.

Similar cu abordarea în cazul egalizorului Log-MAP, sunt determinate valorile $ln\left[P\left(c_{1,4} = +1 \middle| L_p^i\right)\right]$ şi $ln\left[P\left(c_{1,4} = -1 \middle| L_p^i\right)\right]$, pentru a calcula valoarea LLR pentru $c_{1,4}$. Din figura 4.15(a) şi folosind relaţia 4.4, se obţine termenul $ln\left[P\left(c_{1,4} = -1 \middle| L_p^i\right)\right]$, care este dat de:

$$ln\left[P\left(c_{1,4} = -1 \middle| L_p^i\right)\right] = ln[exp(2,1 + (-1,0) + 3,2) + exp(2,1 + (-1,0) + 3,3)$$
$$+ exp(1,0 + (-0,1) + 3,2) + exp(0,4 + (-0,1) + 3,3)]$$
$$\approx ln[exp(\overbrace{max(4,3;4,4) + \underbrace{0,65}_{TAB}}^{=5,05}) + exp(1,0 + (-0,1) + 3,2)$$
$$+ exp(0,4 + (-0,1) + 3,3)]$$
$$\approx ln[exp(\overbrace{max(5,05;4,1) + \underbrace{0,35}_{TAB}}^{=5,4}) + exp(0,4 + (-0,1) + 3,3)]$$
$$\approx max(5,4;3,6) + \underbrace{0,15}_{TAB} \approx 5,55$$

în timp ce $ln\left[P\left(c_{1,4} = +1 \middle| L_p^i\right)\right]$ poate fi scris ca:

$$ln\left[P\left(c_{2,4} = +1 \middle| L_p^i\right)\right] = ln[exp(2,1 + 1,0 + 3,3) + exp(2,1 + 1,0 + 3,2)$$
$$+ exp(1,0 + (-0,1) + 3,2) + exp(0,4 + (-0,1) + 3,3)]$$
$$\approx ln[exp(\overbrace{max(6,4;6,3) + \underbrace{0,65}_{TAB}}^{=7,05}) + exp(1,0 + (-0,1) + 3,2)$$
$$+ exp(0,4 + (-0,1) + 3,3)]$$

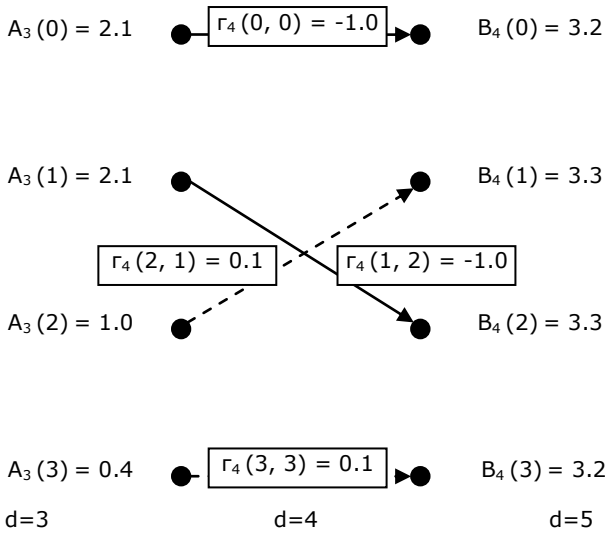
$$\approx ln[exp(\overbrace{max(7,05;4,1)+\underbrace{0,05}_{TAB}}^{=7,1})+exp(0,4+(-0,1)+3,3)]$$
$$\approx max(7,1;3,6)+\underbrace{0,05}_{TAB}\approx 7,15$$

Din relaţiile de mai sus, valoarea LLR pentru bitului $c_{1,4}$ la intervalul de trellis $d=4$ este:

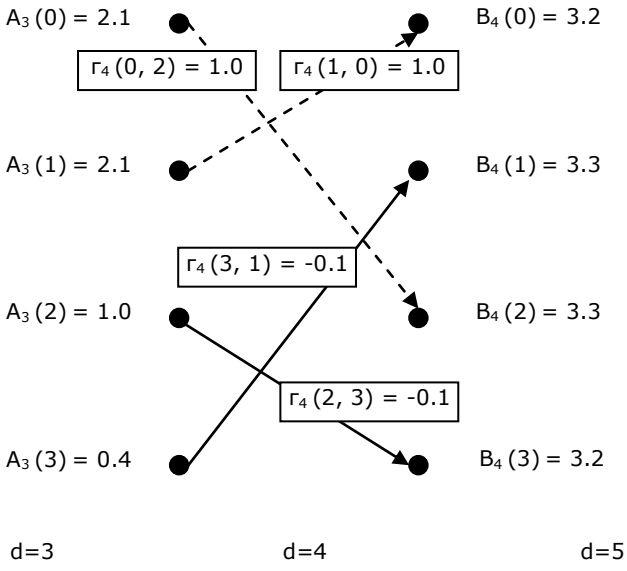
$$ln\left[P\left(c_{1,4}=+1\middle|L_p^i\right)\right]-ln\left[P\left(c_{1,4}=-1\middle|L_p^i\right)\right]=7,15-5,55=1,6$$

Urmând aceeaşi abordare poate fi calculată valoarea LLR pentru cel de-al doilea bit codat $c_{2,4}$ la intervalul $d=4$. În mod explicit, trebuie să se calculeze $ln\left[P\left(c_{2,4}=-1\middle|L_p^i\right)\right]$ şi $ln\left[P\left(c_{2,4}=+1\middle|L_p^i\right)\right]$ prin luarea în considerare a unui set diferit de tranziţii, care au fost determinate de biţi codaţi $c_{2,4}=+1$ şi respectiv $c_{2,4}=-1$. (aşa cum se vede din diagrama de trellis din figura 4.8). Având în vedere setul de tranziţii corespunzător bitului $c_{2,4}=-1$ prezentat în figura 4.16(a), termenul $ln\left[P\left(c_{2,4}=-1\middle|L_p^i\right)\right]$ este:

$$ln\left[P\left(c_{2,4}=-1\middle|L_p^i\right)\right]=ln[exp(2,1+(-1,0)+3,2)+exp(2,1+(-1,0)+3,3)$$
$$+exp(1,0+0,1+3,3)+exp(0,4+0,1+3,2)]$$
$$\approx ln[exp(\overbrace{max(4,3;4,4)+\underbrace{0,65}_{TAB}}^{=5,15})+exp(1,0+0,1+3,3)$$
$$+exp(0,4+0,1+3,2)]$$
$$\approx ln[exp(\overbrace{max(5,15;4,4)+\underbrace{0,25}_{TAB}}^{=5,4})+exp(0,4+0,1+3,2)]$$
$$\approx max(5,4;3,7)+\underbrace{0,15}_{TAB}\approx 5,55$$



(a) Tranziţiile corespunzătoare biţilor codaţi $c_{2,4} = -1$



(b) Tranziţiile corespunzătoare biţilor codaţi $c_{2,4} = +1$

Figura 4.16. Tranziţiile de trellis ale decodoarelor de canal cu K=3 RSC şi R=1/2, care au fost luate în considerare atunci când se calculează valorile LLR corespunzătoare bitului $c_{2,4}$, care este al doilea bit codat la intervalul de trellis d=4. (Trellisul este bazat pe figura 4.8)

Luând în considerare toate tranziţiile de trellis posibile corespunzătoare bitului $c_{2,4}=+1$, $ln\left[P\left(c_{2,4}=+1\middle|L_p^i\right)\right]$ poate fi calculat în felul următor:

$$ln\left[P\left(c_{2,4}=+1\middle|L_p^i\right)\right]=ln[exp(2,1+1,0+3,3)+exp(2,1+1,0+3,2)$$
$$+exp(1,0+(-0,1)+3,2)+exp(0,4+(-0,1)+3,3)]$$
$$\approx ln[exp(\overbrace{max(6,4;6,3)+\underbrace{0,65}_{TAB}}^{=7,05})+exp(1,0+(-0,1)+3,2)$$
$$+exp(0,4+(-0,1)+3,3)]$$
$$\approx ln[exp(\overbrace{max(7,05;4,1)+\underbrace{0,05}_{TAB}}^{=7,1})+exp(0,4+(-0,1)+3,3)]$$
$$\approx max(7,1;3,6)+\underbrace{0,05}_{TAB}\approx 7,15$$

Rezultă că valoarea LLR a-posteriori pentru bitul $c_{2,4}=1$ la interval $d=4$ este $7,15-5,55=1,6$. Pentru a determina valorile LLR ale biţilor codificaţi, această operaţiune se aplică la toate intervalele tranziţiei de trellis din trellis.

În ultima iteraţie, se calculează doar valoarea LLR a posteriori a bitului sursă în loc de doi biţi codificaţi, deoarece trebuie să fie determinaţi numai biţii sursă transmişi. În exemplul prezentat decizia privind biţii sursă, bazată pe valorile LLR a-posteriori ale acestora, a fost făcută chiar după prima iteraţie, după cum vede în tabelul 4.4. Această decizie a fost invocată pentru determinarea performanţelor de decodare a codului convoluţional utilizând algoritmul Log-MAP, care este echivalent cu performanţa egalizării turbo după o iteraţie de egalizare turbo. Cu toate acestea, în general, biţii detectaţi sunt determinaţi numai în iteraţia finală.

Metricile numeroase care caracterizează decodorul convoluţional din figura 4.10 au fost extrase din simulări, după prima iteraţie de egalizare turbo şi prezentate în tabelul 4.4 . Prima dată se va lua în considerare informaţiile primite de decodorul Log-MAP, urmată de informaţiile de la ieşirea decodorului.

Se observă în tabelul 4.4 că decodorul convoluţional nu recepţionează niciodată informaţiile a-priori (APRI-S) cu privire la bitul sursă, cu excepţia cazului în aceste informaţii pot fi extrase dintr-o altă sursă independentă, cum ar fi un alt decodor convoluţional.

LLR-ul extrinsec (EXT-C) este calculat prin scăderea LLR-ului corespunzător informaţiei extrinseci combinate cu a canalului (RXD-LLR) din valoarea LLR a-posteriori a biţilor codaţi (APOS-C). Pentru a determina numărul de biţi eronaţi, biţii de decizie (DCSN) obţinuţi de decodor ca informaţie a-posteriori (APOS-S) sunt

comparaţi cu biţi sursă transmişi (TXD-S). Se observă faptul că în prima iteraţie au fost obţinute două erori.

Prin urmare, în mod similar ca la egalizor, şi la decodor valoarea LLR a-priori a bitului sursă (numit APRI-S) este, de asemenea, iniţializat la 0, indicând faptul că biţii $+1$ şi -1 au probabilitate egală de apariţie.

	Intervalul de trellis *d* al decodorului				
	1	2	3	4	5
APRI-S	0,0	0,0	0,0	0,0	0,0
RXD-LLR	2,7; -0,5	-0,1; -0,6	-0,4; 2,1	1,1; 0,8	0,2; -6,5
APOS-C	2,5; 2,5	0,2; -0,3	0,1; 2,3	1,6; 1,6	0,2; -6,5
EXT-C	-0,2; 3,0	0,3; 0,3	0,5; 0,2	0,6; 0,8	0,0; 0,0
APOS-S	2,5	0,2	0,1	1,6	0,2
TXD-S	1	1	-1	-1	1
DCSN	1	1	**1**	**1**	1

Tabelul 4.4. Metricile decodorului de canal extrase din simulări după prima iteraţie de turbo egalizare

Cu toate acestea, spre deosebire de egalizorul Log-MAP, în absenţa unor decodoare suplimentare, valorile APRI-S rămân la 0 în iteraţiile succesive de egalizare turbo, în timp ce în cazul egalizorului turbo valorile LLR a-priori îmbunătăţesc operaţiile egalizorului turbo în timpul celei de-a doua iteraţii

După cum s-a menţionat anterior, decodorul recepţionează, de asemenea valoarea LLR asociată cu informaţiile extrinseci combinate cu ale canalului, (numite EXT în Tabelul 4.3), care au fost de-întreţesute rezultând RXD-LLR (din tabelul 4.4).

Ulterior, cu ajutorul acestor valori LLR recepţionate, adică APRI-S, precum şi RXD-LLR, şi prin utilizarea principiilor subliniate mai sus, decodorul Log-MAP este acum gata să determine valorile LLR aposteriori a biţilor sursă şi a biţilor codificaţi, şi anume (APOS-S) şi respectiv (APOS-C), (din tabelul 4.4). Se ştie că informaţiile de intrare pentru un bloc particular în iteraţia curentă nu trebuie să includă informaţii care au contribuit la acel bloc particular din iteraţia precedentă, deoarece trebuie asigurată o corelaţie minimă între aceste valori.

Prin urmare, după cum se vede în figura 4.10, valorile LLR RXD-LLR extrinseci combinate cu ale canalului se scad din APOS-C, pentru a obţine valorile LLR extrinseci (EXT-C tabelul 4.5). Valorile EXT-C sunt ulterior întreţesute şi trimise la egalizorul Log-MAP în iteraţia următoare ca valori LLR apriori. (APRI din tabelul 4.5).

	Intervalul de trellis *m* al egalizorului												
	1	2	3	4	5	6	7	8	9	10	11	12	13
APRI	-0,2	0,2	2,9	0,5	0,3	0,8	0,3	0,1	0,5	-0,1	-115,1	-115,1	-115,1
APOS	2	1,2	2,2	0,1	0,1	1,3	-0,6	-0,1	-0,1	-6,7	-123,1	-199,9	-116,2
EXT	2,2	1	-0.7	-0,4	-0,2	0,5	-0,9	-0,2	-0,6	-6,6	-8	-4,8	-1,1
TXD-C	1	-1	1	-1	1	1	-1	1	-1	-1	-1	-1	-1
DCSN	1	**1**	1	**1**	1	1	-1	**-1**	-1	-1	-1	-1	-1

Tabelul 4.5 Diferite metrici de egalizare extrase din simulări după a doua iteraţie

Aceste informaţii suplimentare vor fi utilizate de către egalizor în a doua iteraţie, pentru a îmbunătăţi fiabilitatea valorii LLR a-posteriori. Valorile LLR a-posteriori a biţilor sursă, (notate APOS-S în tabelul 4.4) sunt utilizate pentru a detecta biţii sursă transmişi (notaţi cu DCSN în tabelul 4.4).

Prin compararea DCSN cu biţii sursă care au fost efectiv transmişi (TXD-S), se observă că au apărut două erori (care au fost subliniate cu bold, în tabelul 4.4) în prima iteraţie. În continuare se va arăta că schimbul iterativ de informaţii între decodor şi egalizor îmbunătăţeşte performanţa decodării, astfel încât egalizorul turbo surclasează egalizarea şi decodarea independentă.

Notaţiile TXD-C, APRI, APOS, EXT şi DCSN reprezintă biţi codificaţi transmişi, care au a fost întreţesuţi, LLR-ul a-priori, LLR-ul a-posteriori, LLR extrinsec şi biţii detectaţi sau de decizie .În a doua iteraţie egalizorul primeşte informaţiile a-priori furnizate de decodorul convoluţional, îmbunătăţindu-se astfel fiabilitatea LLR-urilor a-posteriori calculate, generând doar trei erori, afişate cu bold.

	Intervalul de trellis *d* al decodorului				
	1	2	3	4	5
APRI-S	0,0	0,0	0,0	0,0	0,0
RXD-LLR	2,2; -0,7	-0,2; -0,9	-0,5; 1,0	-0,4; 0,5	-0,1; -6,7
APOS-C	1,4; 1,4	0,3; -0,6	0,2; +0,7	-0,2; 0,4	0,1; -6,7
EXT-C	-0,8; 2,1	0,5; -0,3	0,3; -0,3	0,2; -0,1	0,2; 0,0
APOS-S	1,4	0,3	-0,2	-0,2	0,1
TXD-S	1	1	-1	-1	1
DCSN	1	1	-1	-1	1

Tabelul 4.6 Metricile decodorului de canal extrase din simulări după a doua iteraţie de turbo egalizare.(Nu se recepţionează informaţiile a-priori APRI-S).

Ca şi în prima iteraţie, LLR-ul extrinsec EXT-C este calculat prin scăderea LLR-ului corespunzător informaţiilor extrinseci combinate cu ale canalului RXD-LLR din LLR-ul a-posteriori APOS-C al biţilor codaţi generat de decodor.

În a doua iteraţie, egalizorul primeşte informaţia apriori a biţilor sursă APRI-S de la ieşirea decodorului, corespunzătoare primei iteraţii, aşa cum se prezintă în tabelul 4.5. Se ştie că în prima iteraţie de egalizare turbo valoarea LLR a-priori prezentată în linia de sus a tabelului 4.3 a fost iniţializată la 0, indicând faptul că biţii $+1$ şi -1 au avut probabilitate egală de a fi transmişi. Aceste informaţii suplimentare ajută egalizorul în furnizarea de valori LLR a-posteriori, APOS, mai exacte. Fiabilitatea îmbunătăţită este reflectată în numărul redus de erori, în comparaţie cu rezultatele primei iteraţii (văzute în tabelul 4.3), indicând o reducere de la patru la trei erori.

Ca şi înainte, valorile LLR corespunzătoare informaţiilor a-priori sunt scăzute din valorile LLR a-posteriori, la ieşirea egalizorului Log-MAP din figura 4.10, obţinându-se valorile LLR extrinseci combinate cu ale canalului notate EXT, care ulterior sunt de-întreţesute. La decodor valoarea LLR extrinsecă combinată cu a canalului recepţionată este procesată folosind algoritmul Log-MAP şi, după cum se vede în tabelul 4.6, decodorul corectează două erori apărute în iteraţie anterioară (prezentate în tabelul 4.4).

Prin efectuarea egalizării şi decodării iterative, se poate observa că influenta caracterului dispersiv al canalului poate fi eliminata apropape complet. În acest exemplu, erau suficiente două iteraţii de turbo egalizare, spre deosebire de exemplul din tabelul 4.4, care este echivalent cu situaţia în care egalizarea şi

decodarea sunt realizate independent. Pentru a îmbunătăţi şi mai mult performanţa sistemului, pot fi efectuate mai multe iteraţii de turbo egalizare. Odată cu creşterea numărului de iteraţii se îmbunătăţeşte performanţa.

4.5 Rezultate obţinute

4.5.1 Estimarea canalului la transmisia BPSK

Pentru estimarea canalului la transmisia BPSK am utilizat următoarea schema bloc de turbo egalizare din figura 4.17:

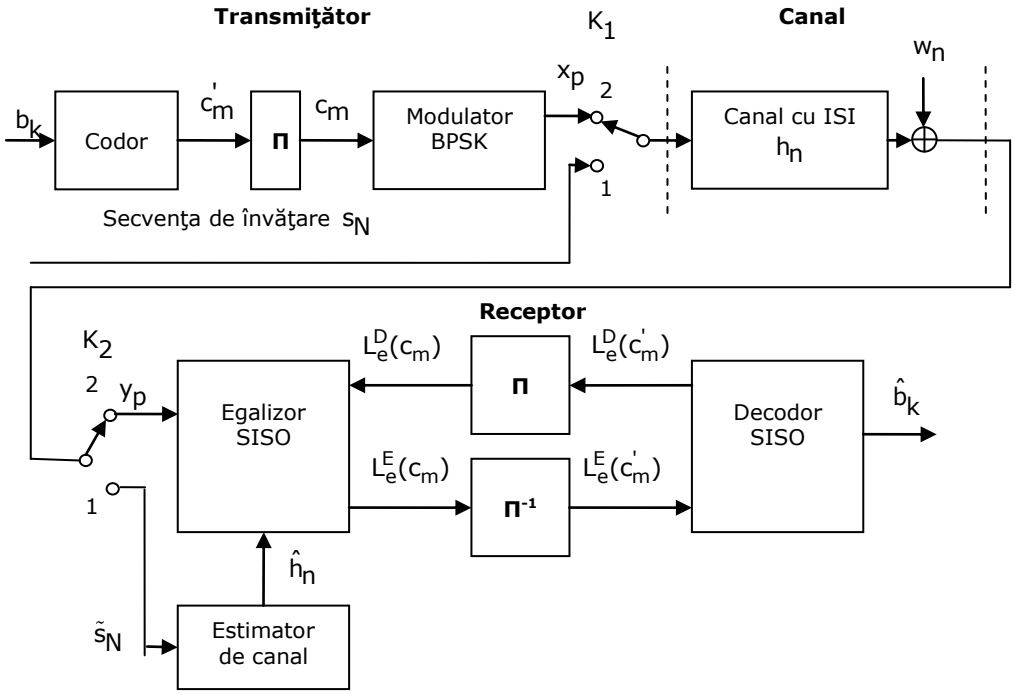


Figura 4.17. Schema bloc a sistemului de emisie – receptie cu estimarea lui h

Se observă din schemă, când comutatoarele K_1 şi K_2 sunt în poziţia 1, se generează o secvenţă de învăţare s_N , cunoscută de estimatorul de canal de la receptor. Această secvenţă s_N , este transmisă prin canal şi suferă modificări datorită canalului şi zgomotului. Pe baza secvenţei recepţionate $\tilde{s}_N$ şi a secvenţei cunoscute de la emisie (s_N), se poate determina, $\hat{h}_n$ valoarea estimată a răspunsului la impuls al canalului, cu ajutorul algoritmilor de estimare. După determinarea lui $\hat{h}_n$ se face prelucrarea semnalului la receptor.

4.5.1.1 Estimare perfectă a canalului

Mai întâi am presupus că $\hat{h}_n = h_n$, adică $\hat{h}_n$, estimatul răspunsului la impuls al canalului, este acelaşi cu h_n răspunsul la impuls al canalului (estimare perfectă a lui h_n). Este importantă cunoaşterea lungimii lui h_n , (n=3) trei in cazul nostru, deoarece în funcţie de aceasta se construieşte egalizorul SISO. Am realizat două simulări folosind mediul de programare Matlab:

i) La prima simulare, am considerat valorile: h=[0.302 0.725 0.456], lungimea secvenţei $b = [b_1, b_2, \ldots\ldots b_k]$ este 32 ($k = 32$), iar numărul de iteraţii efectuate de turbo egalizor este 5. În urma simulării se obţin curbele BER din figura 4.18.

ii) Pentru a doua simulare, se modifică doar lungimea secvenţei $b = [b_1, b_2, \ldots\ldots b_k]$ de 128 ($k = 128$), pastrând numărul de iteraţii efectuate de turbo egalizor şi valoarea lui h ca şi în cazul i). Rezultatele obţinute sunt prezentate în figura 4.19.

Din figurile 4.18 şi 4.19 rezultă că:

- rata erorii de bit este cu atât mai mică cu cât k, lungimea secvenţei de date transmise este mai mare, astfel ca la o crestere a lui k de patru ori, BER scade cu aproape două ordine de mărime.
- sunt suficiente patru iteraţii în procesul de turbo egalizare, deoarece iteraţia cinci nu aduce îmbunătăţiri substanţiale ale BER.

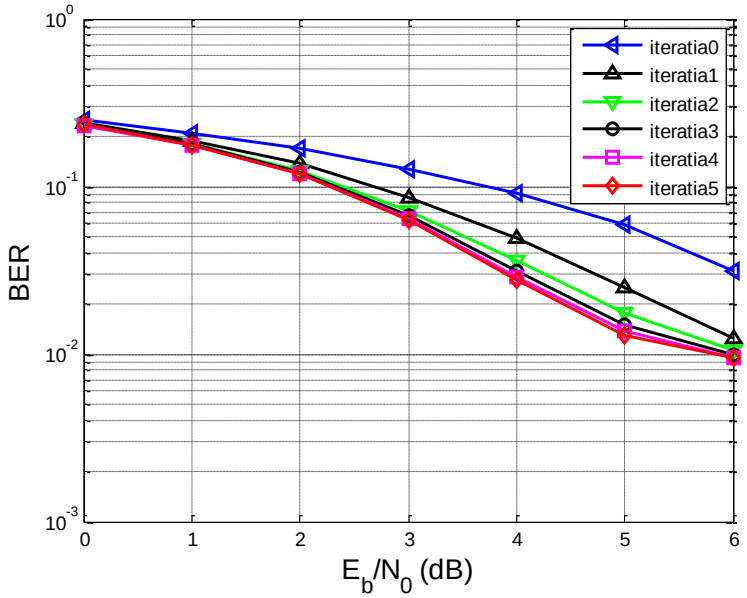


Figura 4.18. BER/SNR (k=32)

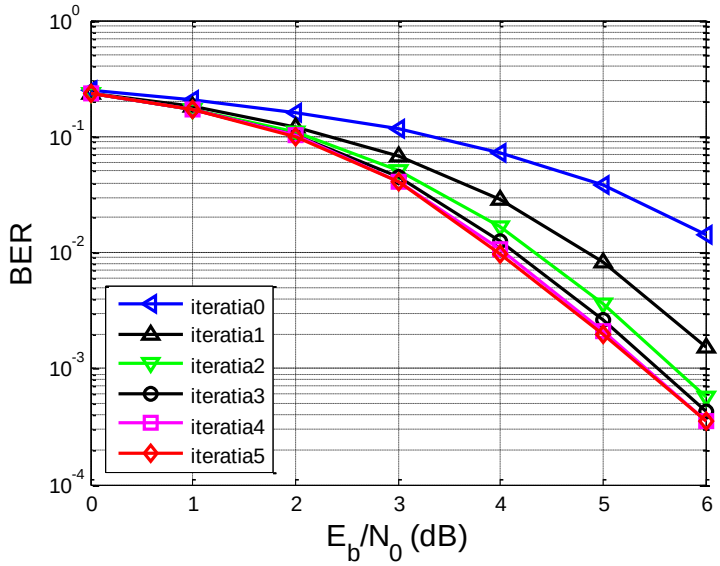


Figura 4.19. BER/SNR (k=128)

4.5.1.2 Estimare eronată a canalului

În continuare, am urmărit BER în procesul de turbo egalizare, considerând estimarea lui h_n ($\hat{h}_n$) ca fiind eronată.

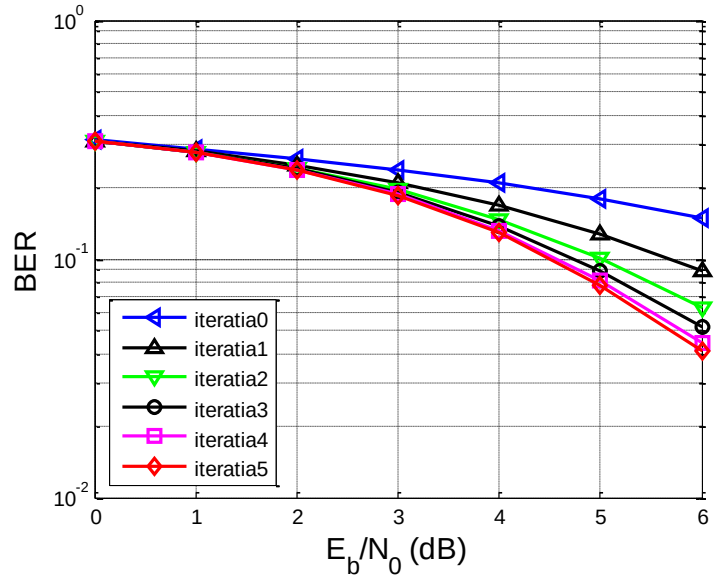


Figura 4.20. BER/SNR, (k=128, $h = [0.302 \quad 0.725 \quad 0.456]$, $\hat{h} = [0.423 \quad 0.567 \quad 0.231]$)

Pentru aceasta am presupus că: $h = [0.302 \quad 0.725 \quad 0.456]$, iar $\hat{h} = [0.423 \quad 0.567 \quad 0.231]$. În cazul estimării perfecte a canalului (figura 4.19) pentru o lungime a secvenţei de date, k=128, la o valoare a SNR de 6dB ($E_b / N_0 = 6dB$), BER la iteraţia 5 este cuprins între 10^{-3} şi 10^{-4} . În cazul cazul estimării eronate a canalului (figura 4.20) pentru aceleaşi date- lungimea secvenţei de date (k=128), SNR=6dB- BER-ul la iteraţia 5 este cuprins între 10^{-1} şi 10^{-2}. Se observă că în cazul unei estimări greşite a lui h_n rezultatele sunt mult mai slabe, BER modificându-se cu două ordine de mărime

Apoi, am propus o schemă, prin care se face o estimare iterativă a lui $\hat{h}$ - vezi figura 4.21.

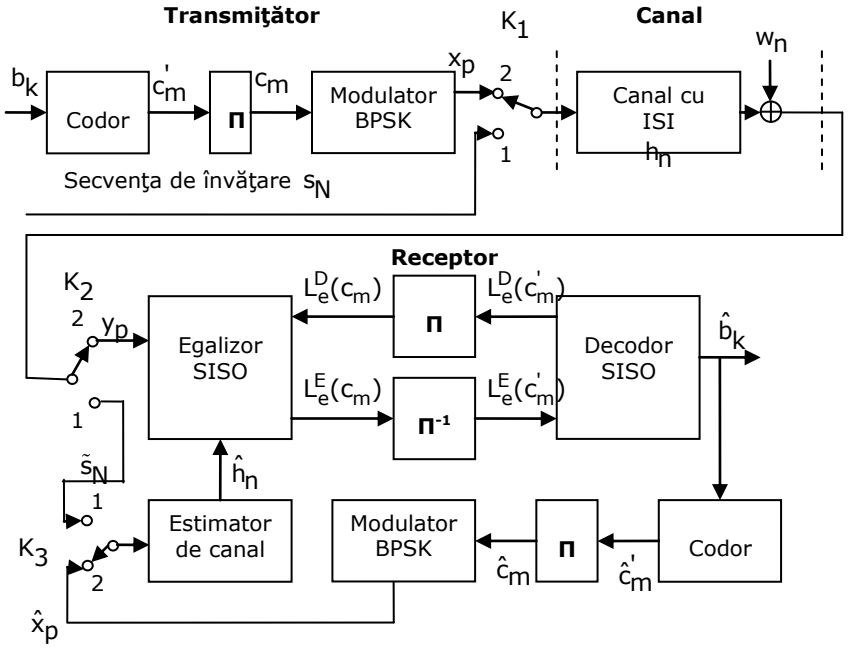


Figura 4.21. Schema bloc a sistemului de emisie – receptie cu estimarea iterativă a lui $\hat{h}$

Biţii recepţionaţi $\hat{b}_k$, în cadrul fiecărei iteraţii, sunt din nou codaţi, întreţesuţi, iar apoi modulaţi (la fel ca si la partea de emisie) şi aduşi la intrarea blocului de estimare, care în funcţie de semnalul recepţionat y , face o nouă estimare a lui h_n . Valoarea recalculată a lui h_n este transmisă egalizorului SISO. În cadrul secvenţei de învăţare, comutatoarele sunt în poziţia 1, iar în cadrul transmisiei sunt în poziţia 2. În cazul de faţă estimatorul de canal poate utiliza acelaşi algoritm de estimare, atât pentru estimarea iniţială (când recepţionează $\tilde{s}_N$) cât şi pentru estimările ulterioare (când recepţionează $\hat{x}_p$); sau poate utiliza

algoritmi diferiţi; de exemplu LS (least-square) sau MMSE (Minimum Mean-Square Error).

Pentru analiza comparativă a performanţelor celor două scheme, din figura 4.17 şi figura 4.21 voi presupune că algoritmul care face estimarea iniţială este necunoscut, dar cunosc valorile lui h_n, respectiv $\hat{h}_n$. Dacă în schema din figura 4.17, $\hat{h}_n$ rămâne constant pe durata celor cinci iteraţii, în schema din figura 4.21, $\hat{h}_n$ este recalculat la fiecare iteraţie cu algoritmul LS (Least-Square) .

Metoda de estimare LS -descrisă în capitolul 3- implică găsirea lui $\hat{h}$ utilizând următoarea relaţie:

$$\hat{h}_{LS} = \left(S^T \cdot S\right)^{-1} \cdot S^T \cdot \tilde{s} \tag{4.28}$$

unde S este matricea Toeplitz corespunzătoare secvenţei de învăţare transmise $s[N]$, iar S^T reprezintă transpusa matricei S.

Dacă estimarea lui h se face în funcţie de semnalul estimat care s-a transmis $\hat{x}[p]$, cum este în cazul schemei din figura 4.21, relaţia 4.28 devine:

$$\hat{h}_{LS} = \left(\hat{X}^T \cdot \hat{X}\right)^{-1} \cdot \hat{X}^T \cdot y \tag{4.29}$$

i) În primul caz se consideră că: $h = \hat{h} = [0.302 \quad 0.725 \quad 0.456]$, rezultatele simulării acestei transmisii fiind prezentate în figura 4.22.

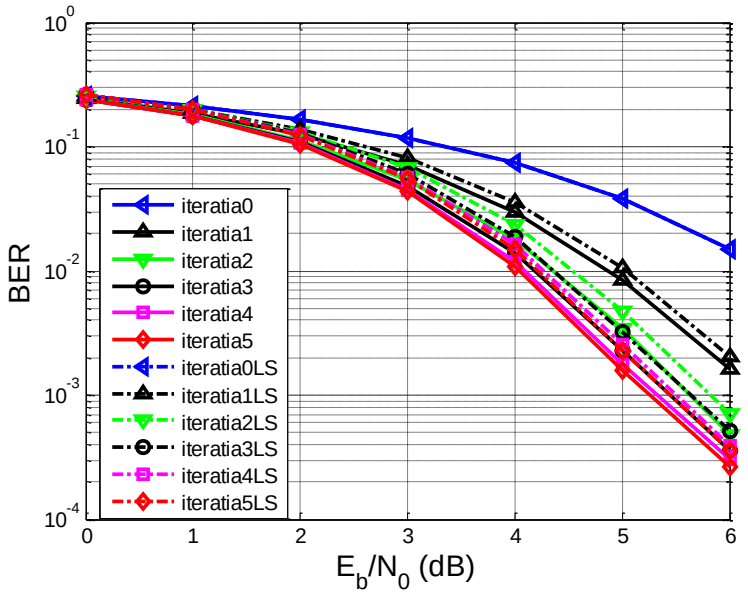


Figura 4.22. BER/SNR pentru cazul i) $h = \hat{h} = [0.302 \quad 0.725 \quad 0.456]$), k=128

ii) În al doilea caz, $h \neq \hat{h}$, şi anume $h = [0.302 \quad 0.725 \quad 0.456]$ şi $\hat{h} = [0.423 \quad 0.567 \quad 0.231]$, rezultatele simulării fiind prezentate în figura 4.23.

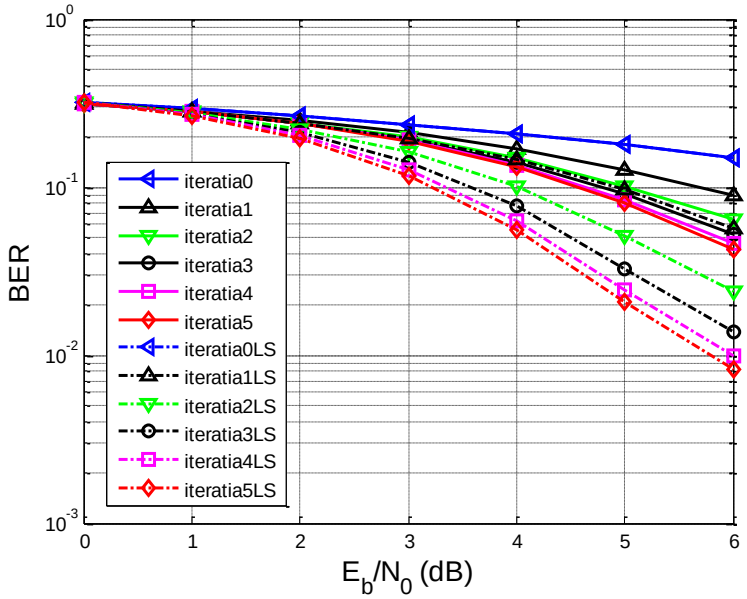


Figura 4.23. BER/SNR pentru cazul ii) $h \neq \hat{h}$
($h = [0.302 \quad 0.725 \quad 0.456]$, $\hat{h} = [0.423 \quad 0.567 \quad 0.231]$), k=128

Din rezultatele expuse în figurile 4.22 şi 4.23, se vede că în primul caz, când $h = \hat{h}$ se recomandă schema din figura 4.17, iar pentru al doilea caz când ($h \neq \hat{h}$), se recomandă utilizarea schemei din figura 4.21.

În continuare se urmăreşte dependenta BER/SNR pentru cele două scheme din figura 4.17 şi figura 4.21 atunci când estimarea lui h_n este făcută cu algoritmul LS. Secvenţa de învăţare s_N este modulată BPSK, iar apoi este transmisă prin canal, obţinându-se la ieşire $\tilde{s}_N$.

Pentru schema din figura 4.21 am considerat că estimările succesive sunt realizate utilizând acelaşi algoritm (LS). Pentru acest exemplu am realizat trei simulări, pentru trei lungimi diferite ale secvenţei de învăţare (N =8, 18; 128).

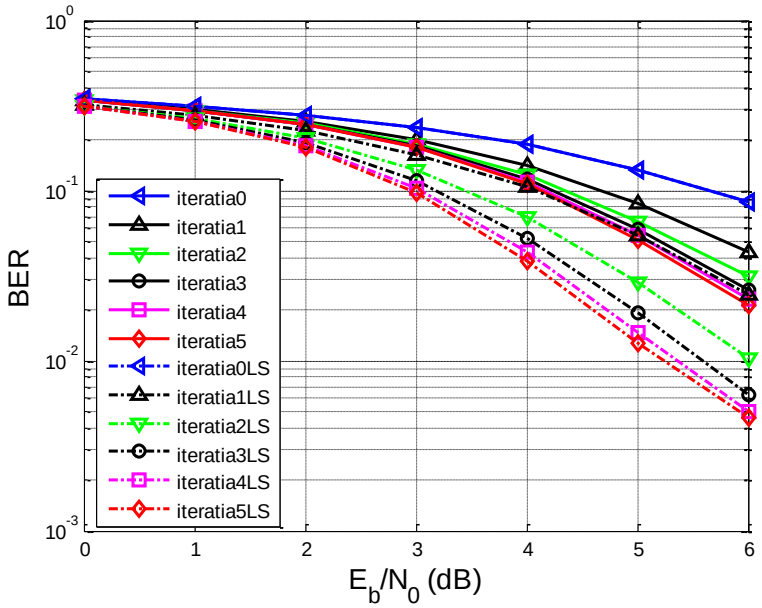


Figura 4.24. BER/SNR
(k=128, $h=\left[0.302 \quad 0.725 \quad 0.456\right]$, N=8)

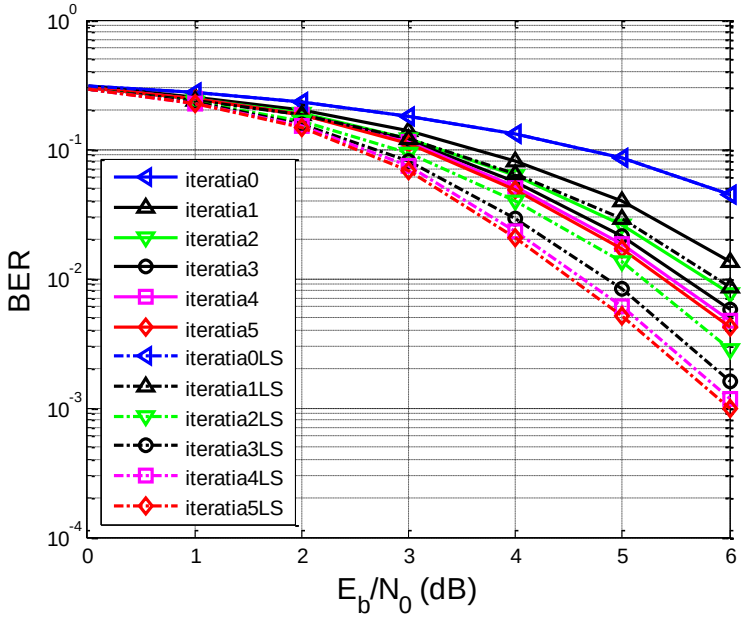


Figura 4.25. BER/SNR (k=128, $h=\left[0.302 \quad 0.725 \quad 0.456\right]$, N=18)

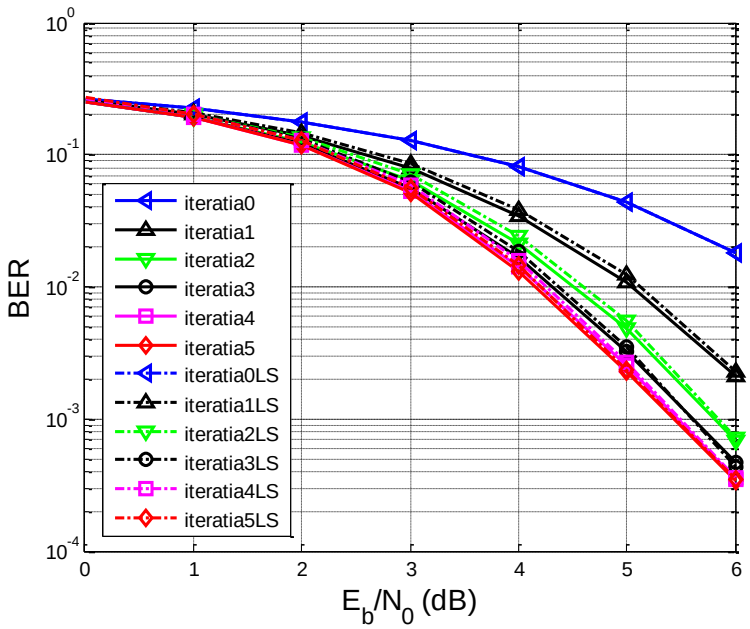


Figura 4.26. BER/SNR (k=128, $h = \left[0.302 \quad 0.725 \quad 0.456\right]$, N=128)

Din graficele obţinute în primele două simulări (figurile 4.24 şi 4.25) se poate observa că rezultatele obţinute cu schema din figura 4.21– care face o estimare iterativă a lui $\hat{h}$ – sunt mult mai bune decât cele obţinute cu schema din figura 4.17– care face estimarea lui $\hat{h}$ o singură dată. În figura 4.26 se observă că rezultatele obţinute cu schema din figura 4.17 sunt mai bune decât cele obţinute cu schema din figura 4.21, ceea ce se datorează unei estimări iniţiale bune a lui h.

Rezultatele din figura 4.26 au fost obţinute utilizând o secvenţă de învăţare de lungime mare (N=128), determinând o scădere a ratei de transmisie. Odată cu creşterea lungimii secvenţei de învăţare, performanţele cresc atât pentru schema din figura 4.17 cât si pentru cea din figura 4.21, dar pentru a menţine rata de transmisie la valori ridicate este de preferat ca lungimea secvenţei de învăţare să fie cât mai mică. De aceea în cazul estimării LS este de preferat utilizarea schemei din figura 4.21.

4.5.2. Estimarea canalului la transmisia OFDM

Se presupune că secvenţa de învăţare s_N este modulată BPSK şi transmisă prin canal utilizând tehnica de transmisie OFDM. Schema bloc în cazul transmisiei OFDM este prezentată în figura 4.27 [1] .

Am implementat în Matlab funcţionarea sistemului de transmisie din figura 4.27 utilizând transformata Fourier discretă (DFT) şi transformata Fourier discretă inversă (IDFT). Această implementare este prezentată în figura 4.28.

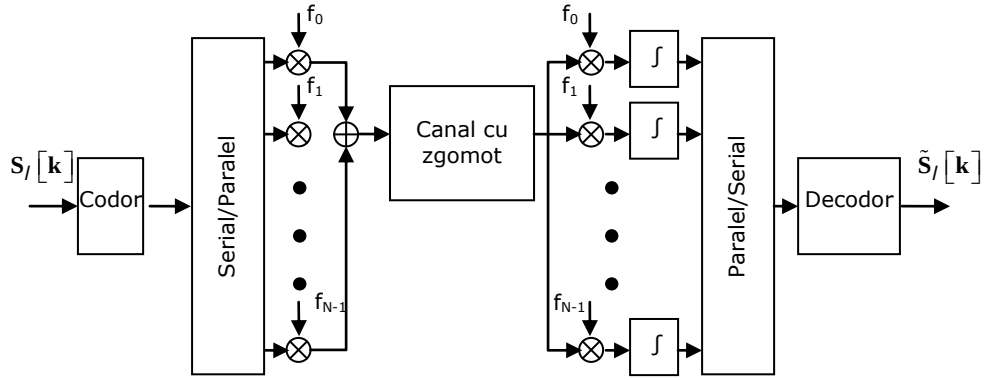


Figura 4.27 Sistem de transmisie utilizând tehnica OFDM

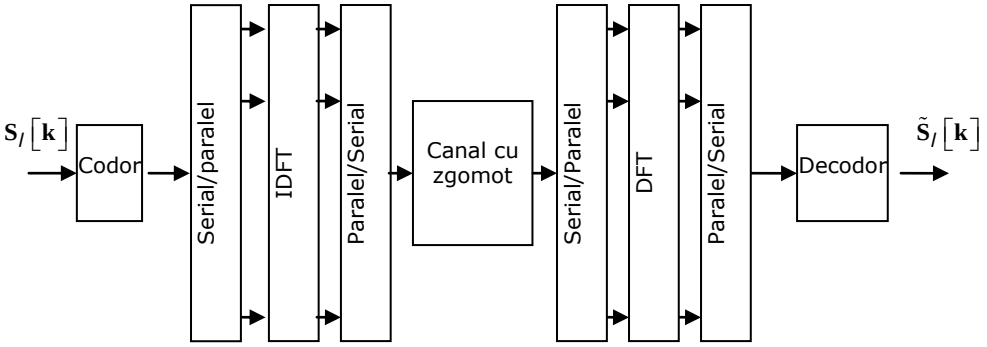


Figura 4.28. Implementarea sistemului de transmisie OFDM utilizând IDFT şi DFT

Pentru estimarea lui h, se utilizează schema bloc din figura 4.29. Deoarece secvenţa de învăţare este cunoscută la recepţie, ea nu mai trebuie să fie codată, respectiv decodată. Astfel se poate renunţa la blocurile de codare şi decodare.

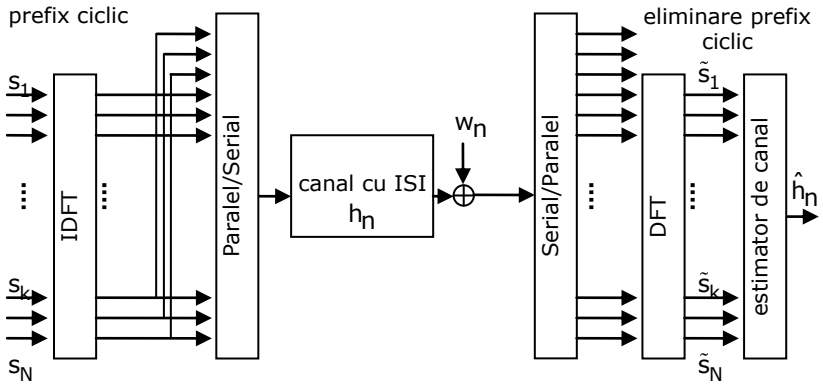


Figura 4.29 Implementarea schemei de transmisie OFDM utilizând IFFT şi DFFT cu prefix ciclic.

Simbolurile de învăţare s_N sunt modulate pe N subpurtătoare cu ajutorul transformatei Fourier discrete inverse (IDFT) , iar ultimele valori L sunt copiate şi puse ca un preambul (prefix ciclic) pentru a forma simbolul OFDM. Prefixul ciclic separă două blocuri OFDM succesive şi facilitează egalizarea canalului la receptor. Totodată permite sincronizarea demodulatorului, elimină interferenţa dintre două simboluri OFDM successive precum şi interferenţa inter-canal.

Acest vector de date este transmis serial peste un canal discret, în timp, lungimea răspunsului la impuls al canalului fiind mai mică decât L. La receptor, prefixul ciclic este eliminat. Semnalul este demodulat cu ajutorul transformatei Fourier discrete (DFT). În acest caz valoarea estimată a lui h, utilizând algoritmul LS, se calculează cu relaţia:

$$\hat{h}_{ls} = S^{-1}\tilde{s} = h + \tilde{n} \tag{4.30}$$

unde $\tilde{n} = S^{-1}n$ este un vector cu variabile independente de zgomotul Gaussian, având matricea de covarianţă: $R_{\tilde{n}\tilde{n}} = \sigma_n^2 \left(SS^H\right)^{-1}$.

Estimatorul liniar MMSE se calculează cu relaţia:

$$\hat{h} = W_X \hat{h}_{ls} \tag{4.31}$$

În relaţia de mai sus W_X reprezintă matricea de ponderare dată de relaţia:

$$W_X = R_{hh}\left(R_{hh} + \sigma_n^2 \left(SS^H\right)^{-1}\right)^{-1} \tag{4.32}$$

iar $R_{hh} = E\left\{hh^H\right\}$ reprezintă matricea de auto-covarianţă a lui h.

Matricea de ponderare W_X este de dimensiune NxN şi depinde de datele transmise S. Utilizând raţionamentul din [36] unde se consideră S ca fiind stochastic cu punctele constelaţiei distribuite uniform şi independente, matricea de auto-covarianţă devine $R_{\tilde{n}\tilde{n}} = \frac{\beta}{SNR}$, relaţie în care β reprezintă factorul de constelaţie $\beta = E\left\{|s_k|^2\right\} E\left\{|s_k|^{-2}\right\}$ ($\beta = 17/9$ pentru 16 QAM, $\beta = 1$ pentru BPSK [3] [4]) .

SNR este raportul semnal zgomot pe simbol, dat de relaţia:

$$SNR = E\left\{|s_k|^2\right\} / \sigma_n^2 \tag{4.33}$$

În continuare am comparat rezultatele obţinute cu schemele din figurile 4.17 şi 4.21.

4.5.2.1 Rata erorii de bit (BER) în funcţie de raportul semnal zgomot utilizând algoritmul de estimare LS

Lungimea secvenţei de date transmise s-a păstrat 128, modificându-se doar lungimea secvenţei de învăţare (N =8, 18; 128). De asemenea prefixul ciclic este fixat la 8 (L=8) . Pe grafic notaţia "LS" reprezintă rezultatele obţinute cu schema din figura 4.17 utilizând algoritmul LS, iar "LS-LS" reprezintă rezultatele obţinute cu schema din figura 4.21 utilizând algoritmul LS atât în estimarea iniţială cât şi în estimările corespunzătoare fiecărei iteraţii.

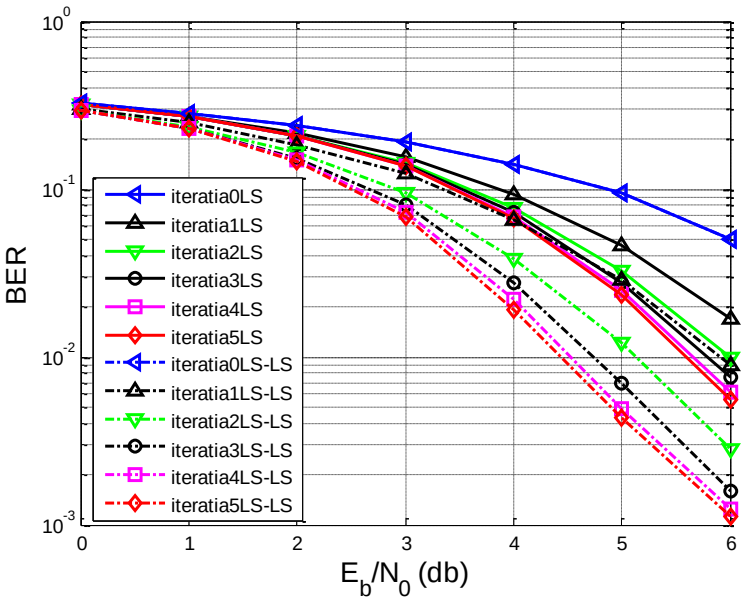


Figura 4.30. BER/SNR
(algoritm LS , k=128 , $h=[0.302 \quad 0.725 \quad 0.456]$, N=8, L=8)

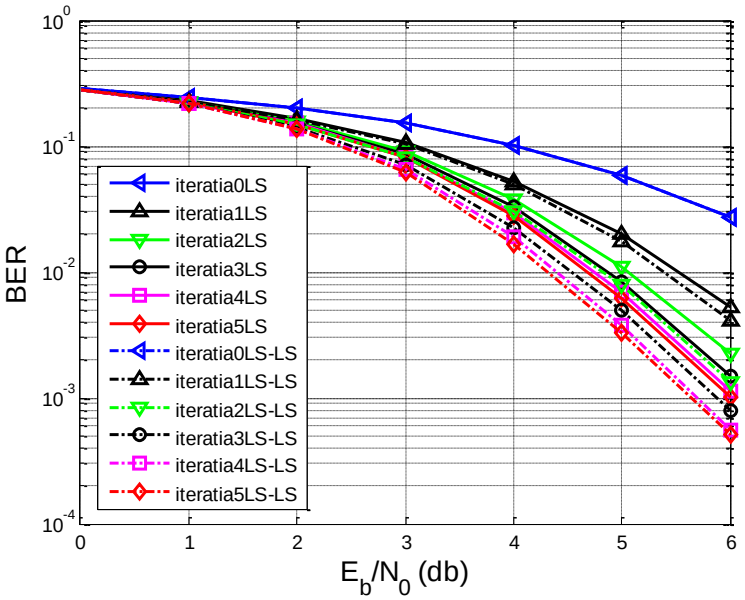


Figura 4.31. BER/SNR
(algoritm LS , k=128 , $h=[0.302 \quad 0.725 \quad 0.456]$, N=18, L=8)

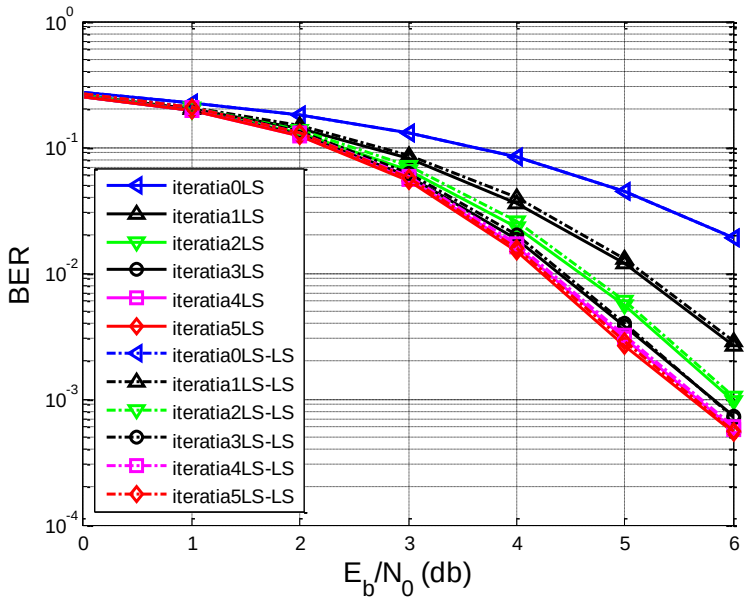


Figura 4.32. BER/SNR
(algoritm LS , k=128 , $h = [0.302 \quad 0.725 \quad 0.456]$, N=64, L=8)

Din analiza graficelor prezentate în figurile 4.30, 4.31 şi 4.32 rezultă şi acum, că estimarea devine mai precisă odată cu creşterea lungimii secvenţei de învăţare. Pentru o secvenţă de învăţare de lungime mare poate fi utilizată schema din figura 4.17, iar pentru o secvenţă de lungime mică schema din figura 4.21.

4.5.2.2 Rata erorii de bit (BER) în funcţie de raportul semnal zgomot utilizând algoritmul de estimare MMSE

Pe graficele din figurile 4.33, 4.34 şi 4.35 notaţia "MMSE" reprezintă rezultatele obţinute cu schema din figura 4.17 utilizând algoritmul MMSE, iar "MMSE -LS" reprezintă rezultatele obţinute cu schema din figura 4.21 utilizând pentru estimarea iniţială (iteraţia 0) , algoritmul MMSE şi algoritmul LS pentru estimările din următoarele iteraţii.

Se observă că utilizând algoritmul MMSE în schema din figura 4.17, se obţin rezultate mai bune decât cu algoritmul LS pentru secvenţele de învăţare de lungime mai mică. Totuşi pentru secvenţe de învăţare de lungime mică (când estimarea lui h este mai puţin bună) se recomandă schema din figura 4.21

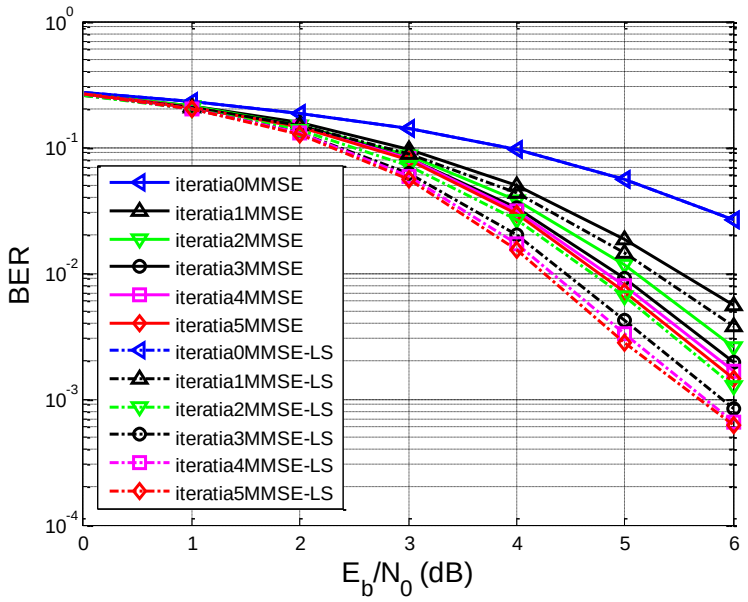


Figura 4.33. BER/SNR
(algoritm MMSE , k=128 , $h=\left[0.302 \quad 0.725 \quad 0.456\right]$, N=8, L=8)

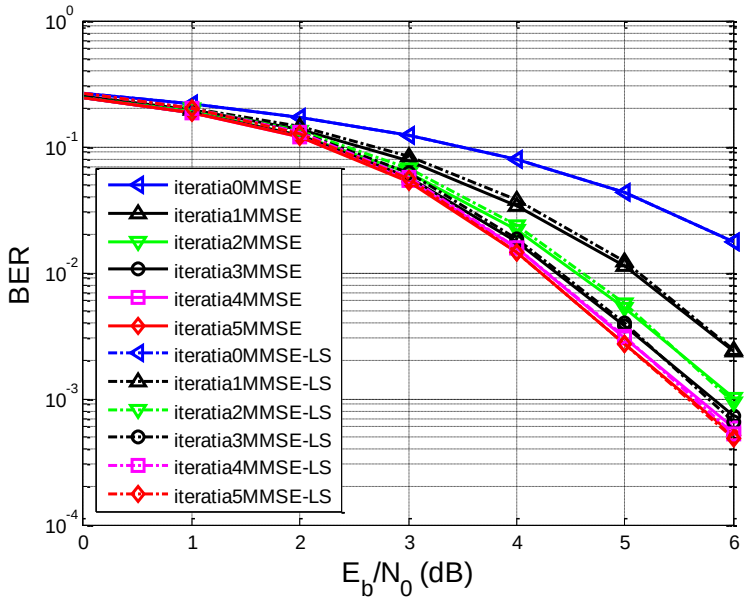


Figura 4.34. BER/SNR
(algoritm MMSE , k=128 , $h=\left[0.302 \quad 0.725 \quad 0.456\right]$, N=18, L=8)

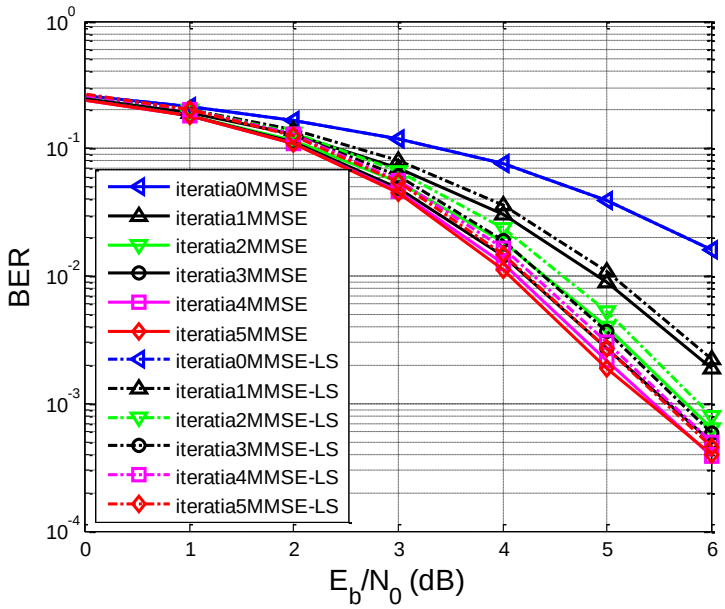


Figura 4.35. BER/SNR
(algoritm MMSE, k=128, $h=[0.302 \quad 0.725 \quad 0.456]$,N=64, L=8)

4.5.2.3 Comparaţia rezultatelor utilizând algoritmii LS si MMSE cu schema din figura 4.17

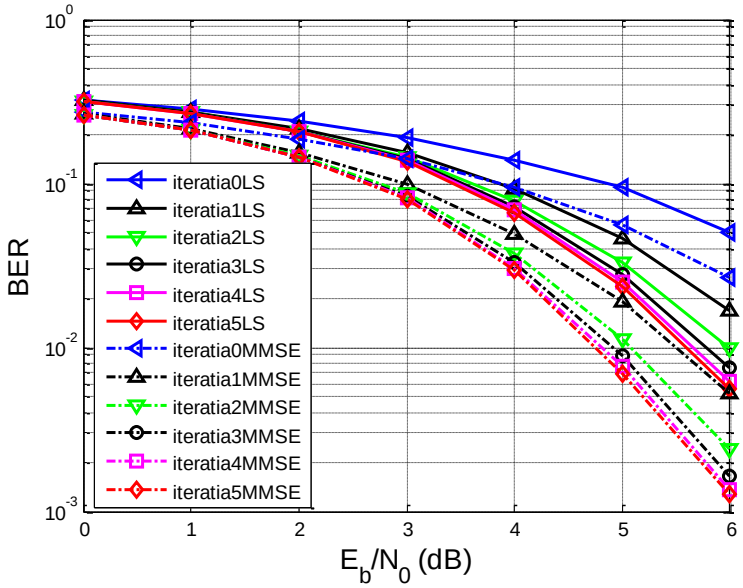


Figura 4.36. BER/SNR (k=128, $h=[0.302 \quad 0.725 \quad 0.456]$, N=8, L=8)

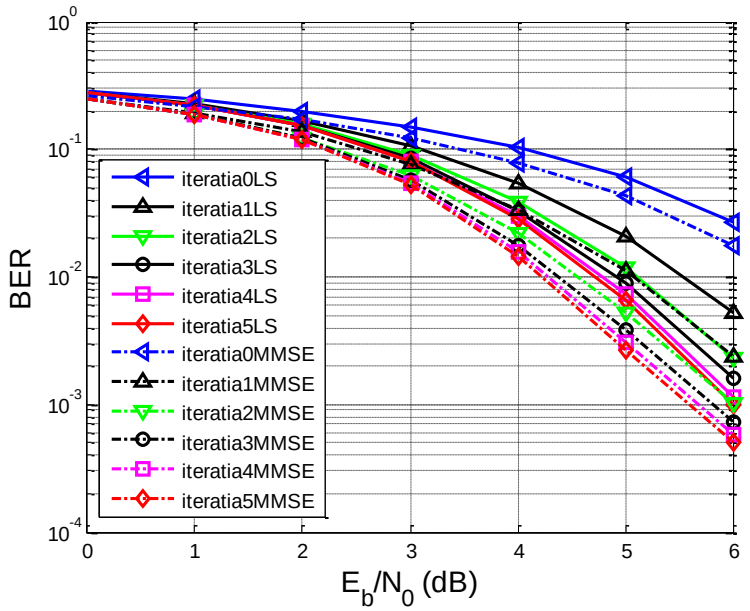


Figura 4.37. BER/SNR (k=128, $h = [0.302 \quad 0.725 \quad 0.456]$, N=18, L=8)

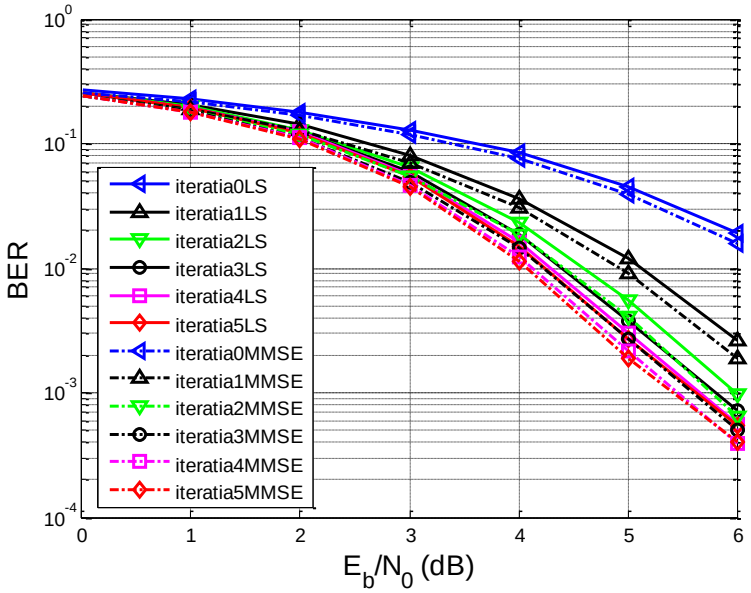


Figura 4.38. BER/SNR (k=128, $h = [0.302 \quad 0.725 \quad 0.456]$, N=64, L=8)

Din rezultatele prezentate în figurile 4.36, 4.37 şi 4.38 am constatat că pentru secvenţe de învăţare de lungime mică rata erorii de bit este mai mică în cazul estimării MMSE decât în cazul estimării LS. Pentru secvenţe de învăţare de

lungime mare performanţele de estimare corespunzătoare celor doi algoritmi sunt apropiate.

4.5.2.4 Comparaţie între rezultatele obţinute cu schema din figura 4.21 utilizând algoritmii LS si MMSE

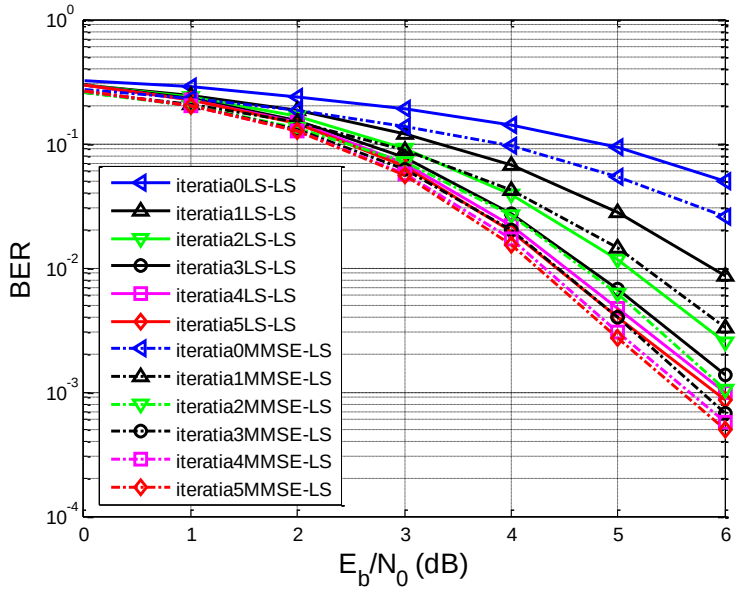


Figura 4.39. BER/SNR (k=128 , $h=\left[0.302 \quad 0.725 \quad 0.456\right]$, N=8, L=8)

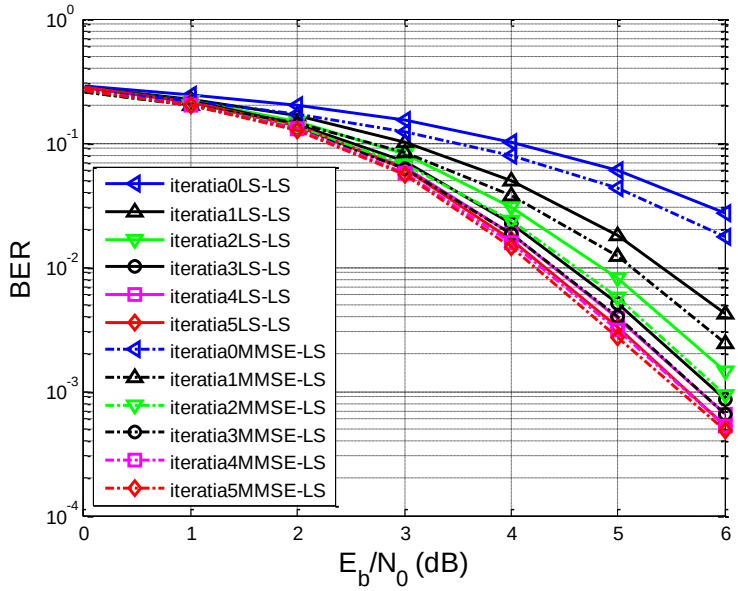


Figura 4.40 BER/SNR (k=128 , $h=\left[0.302 \quad 0.725 \quad 0.456\right]$, N=18, L=8)

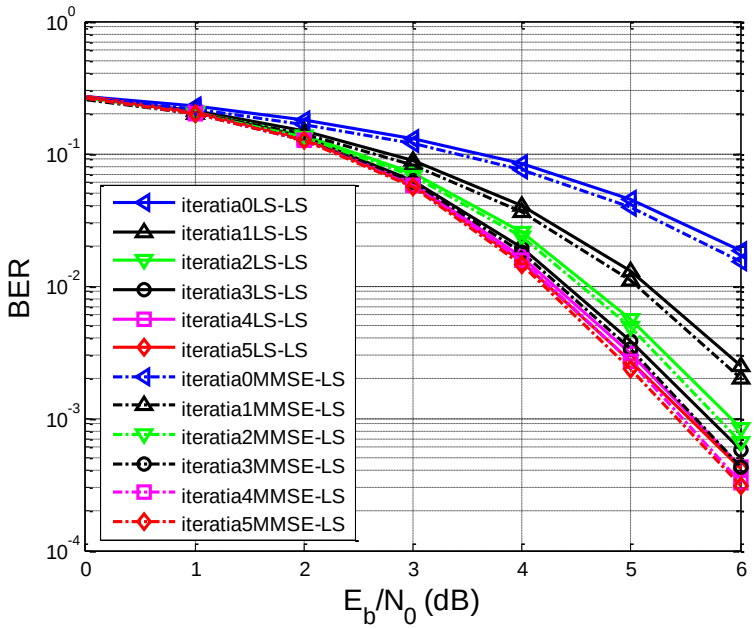


Figura 4.41 BER/SNR (k=128 , $h = [0.302 \quad 0.725 \quad 0.456]$, N=64, L=8)

În urma rezultatelor obţinute cu ajutorul schemei din figura 4.21, prezentate în figurile 4.39, 4.40 şi 4.41, am constatat că odată cu creşterea secvenţei de învăţare performanţele celor doi algoritmi devin apropiate. Pentru secvenţe de învăţare de lungime mică se obţin performanţe mai bune utilizând algoritmul MMSE decât algoritmul LS.

Capitolul 5
Contribuţii şi concluzii

Problematica abordată în această lucrare privind îmbunătăţirea tehnicilor de egalizare ale canalelor radio constituie un subiect de cercetare de mare actualitate.

Creşterea numărului de utilizatori ai reţelelor wireless implică utilizarea eficientă a capacităţii canalului. Această capacitate este limitată printre altele şi de interferenţe, care se pot elimina, reduce sau evita prin tehnici adecvate de egalizare a canalului.

În capitolul 1 am făcut o scurtă prezentare a două tipuri de canale de comunicaţie; canalul AWGN şi canalul cu interferenţă, urmărind capacitatea de transmisie a acestora. Am prezentat, de asemenea, două clase principale de tehnici care fac posibilă utilizarea capacităţii canalelor cu interferenţă aproape la fel ca a canalelor fără interferenţă, cu condiţia ca emiţătorul să fie informat cu privire la interferenţe. O clasă tratează toate subcanalele, ca aparţinând unui singur canal, în timp ce cealaltă clasă partiţionează canalul de bază în sub-canale paralele independente, prin care simbolurile sunt transmise ţinând seama de alocarea optimă a puterii conform principiului water filling.

În capitolul 2 am prezentat câteva scheme clasice de detecţie. Am comparat performanţa detectoarelor pentru transmisii necodate, deoarece detectoarele tratează de obicei datele ca fiind necodate deşi sistemele practice folosesc transmisii codate. Măsura pe care am utilizat-o pentru aprecierea performanţei detectoarelor este probabilitatea erorii pe bit (BER). Primul detector tratat este detectorul de plauzibilitate maxima, ML (Maximum Likelihood) care minimizează probabilitatea detecţiei eronate a unui vector/cuvânt de cod şi care poate fi considerat ca fiind optimal. Detectorul ML oferă o limită inferioară a BER, mai mică decât a oricărui alt detector. Celelalte detectoare prezentate cum ar fi: detectoarele liniare, detectoarele cu reacţie decizională, detectoarele multinivel au o performanţă care se apropie de cea a ML, dar au o complexitate mai redusă.

În capitolul 3 am făcut o descriere atât a principiului de funcţionare a codorului şi decodorului turbo cât şi a implementării acestora în Matlab utilizând algoritmul BCJR.

În urma implementării programului în Matlab am făcut o analiză a ratei erorii pe bit (BER) în funcţie de lungimea secvenţei transmise, numărul de iteraţii efectuate de decodorul turbo precum şi de raportul semnal zgomot (SNR). În urma analizei am constatat că BER scade odată cu creşterea lungimii secvenţei transmise, a numărului de iteraţii precum şi a raportului semnal zgomot. Totodată, am observat că după aproximativ patru iteraţii BER rămâne constant pentru aceeaşi valoare a SNR, rezultând faptul că în procesul de decodare turbo sunt suficiente maxim patru iteraţii de decodare turbo.

Deoarece la recepţie pentru a putea face egalizarea, este foarte importantă cunoaşterea sau estimarea răspunsului la impuls (h) al canalului, am făcut analiza pentru trei cazuri şi anume: i) estimarea perfectă a răspunsului la impuls (h) al canalului, ii) estimarea eronată a lui h, precum şi iii) estimarea lui h cu ajutorul algoritmului de estimare LS.

Procesul de egalizare l-am realizat atât cu egalizorul ZF cât şi cu egalizorul MMSE. Înainte de a introduce cele două egalizoare în schema de decodare turbo

(figura 3.13) am făcut o analiză a performantelor acestora (BER/SNR). În urma rezultatelor am constatat că egalizarea realizată cu egalizorul MMSE oferă performanţe mai bune decât aceea realizată cu egalizorul ZF (figura 3.14).

Pentru estimarea lui h am generat secvenţe de învăţare de lungimi diferite care au fost modulate BPSK şi transmise prin canal, iar pe baza secvenţei de învăţare recepţionate şi cu ajutorul algoritmului de estimare LS, am determinat valoarea estimată $\hat{h}$.

Am observat că valoarea estimată $\hat{h}$ este cu atât mai bună cu cât lungimea secvenţei de învăţare este mai mare.

Şi în urma introducerii acestor egalizoare în schema de decodare turbo (figura 3.13), am obţinut rezultate mai bune în ceea ce priveşte rata erorii de bit in funcţie de raportul semnal zgomot cu schema în care egalizorul era MMSE

În cazul utilizării egalizorului MMSE, pentru o secvenţă de învăţare N=8 şi o lungime a secvenţei de biţi k=128, la o valoare a SNR=4dB, BER este cuprins între 10^{-1} şi 10^{-2} (figura 3.15), iar pentru o secvenţă de învăţare N=16, BER este cuprins între 10^{-2} şi 10^{-3} (figura 3.16).

Utilizând egalizorul ZF pentru aceeaşi valoare a secvenţei transmise (k=128) şi pentru cele două valori ale lungimii secvenţei de învăţare (N=8 şi N=16), valoarea BER scade foarte puţin (figura 3.15 şi 3.16) în cazul creşterii secvenţei de învăţare (cu aproximativ 0.04).

Deoarece creşterea secvenţei de învăţare duce la o scădere a ratei de transmisie, am menţinut secvenţa de învăţare la o valoare mică N=8 şi am realizat o estimare iterativă a lui $\hat{h}$ (figura 3.17) pe baza secvenţei de date recepţionate şi a secvenţei estimate transmise utilizând algoritmul LS. Secvenţa estimată transmisă s-a obţinut prin codarea, întreţeserea şi modularea secvenţei de biţi estimate de la ieşirea decodorului turbo.

În acest caz am utilizat doar egalizorul MMSE deoarece oferă performanţe mai bune (BER/SNR) decât egalizorul ZF. Totodată am comparat performanţele celor două scheme (schema din figura 3.13 şi 3.17) în cazul în care egalizorul este MMSE.

În cazul utilizării estimării iterative, pentru o secvenţă de învăţare N=8 şi o lungime a secvenţei de biţi k=128, la o valoare a SNR=4dB, BER este cuprins între 10^{-1} şi 10^{-2} (figura 3.19). Dacă se măreşte lungimea secvenţei de învăţare, N=16, la aceeaşi valoare a SNR (SNR=4dB), BER este cuprins între 10^{-3} şi 10^{-4} (figura 3.20).

În urma simulărilor am observat că estimarea iterativă a lui $\hat{h}$ duce la o scădere a BER odată cu creşterea numărului de estimări. Creşterea numărului de estimări duce la o creştere a numărului de iteraţii efectuate de decodorul turbo. Dacă se face de R ori recalcularea lui $\hat{h}$, iar I este numărul de iteraţii efectuate de decodor în cadrul unui proces de decodare, atunci numărul total de iteraţii efectuate de decodorul turbo este $(R+1)\cdot I$. Deci dezavantajul este numărul mare de iteraţii.

În capitolul 4 am făcut o descriere a principiului de funcţionare a egalizorului turbo. Egalizorul şi decodorul din schemele utilizate sunt de tip SISO şi ambele utilizează algoritmul MAP. În cazul în care canalul este multicale, am observat că este mult mai avantajos utilizarea egalizării turbo, decât utilizarea separată atât a egalizării cât şi a decodării turbo. În cazul utilizării egalizării turbo informaţiile extrinseci sunt readuse la intrarea egalizorului ca informaţii a priori. Astfel egalizorul şi decodorul lucrând împreună vor genera în iteraţia următoare date

mai exacte (mai rafinate). Astfel pentru a obţine aceleaşi performanţe (BER/SNR) ca şi în cazul utilizării separate a egalizării şi decodării turbo, numărul de iteraţii în procesul de egalizare turbo este mai mic.

Când se proiectează egalizorul SISO este important să se ştie lungimea răspunsului la impuls al canalului şi valoarea estimată a acestuia. În cadrul acestei lucrări am ales lungimea răspunsului la impuls al canalului ca fiind trei.

Pentru analiza funcţionării circuitului în funcţie de valoarea estimată a lui h am considerat trei cazuri:

i) estimarea perfectă a canalului,

ii) estimarea eronată a canalului

iii) estimarea canalului cu ajutorul algoritmilor de estimare LS şi MMSE.

Pentru toate cele trei cazuri am propus două scheme bloc:

- prima schemă face calcularea lui $\hat{h}$ o singură dată (figura 4.17),

- a doua schemă face calcularea lui $\hat{h}$ iterativ (4.21). În cazul celei de-a doua scheme, după fiecare iteraţie de egalizare turbo, secvenţa estimată de la ieşirea decodorului este codată întreţesută şi modulată BPSK, iar apoi adusă la intrarea estimatorului de canal. Estimatorul de canal recalculează valoarea lui $\hat{h}$ utilizând algoritmul LS pe baza secvenţei estimate transmise şi a secvenţei recepţionate la ieşirea din canal.

În cazul i) estimarea iterativă nu aduce nici o îmbunătăţire a BER la aceeaşi valoare a SNR (figura 4.22).

Pentru a vedea dependenţa BER în funcţie de lungimea secvenţei de date transmise am utilizat schema bloc din figura 4.17. Am luat două valori pentru lungimea secvenţei de date (k=32 şi k=128).Pentru o lungime a secvenţei de date k=32 la o valoare a SNR=6dB, BER este cuprins între 10^{-2} şi 10^{-3} (figura 4.18), iar pentru k=128, între 10^{-3} şi 10^{-4} (figura 4.19). În urma rezultatelor obţinute rezultă că odată cu creşterea lungimii secvenţei de date transmise BER scade.

Pentru al doilea caz, ii) estimarea iterativă oferă performanţe mai bune decât în cazul în care aceasta se face o sigură dată.

În cazul estimării eronate a canalului (figura 4.20), utilizând schema din figura 4.17, pentru o lungime a secvenţei de date k=128 , SNR=6dB, BER la iteraţia 5 este cuprins între 10^{-1} şi 10^{-2} . În cazul estimării iterative (schema din figura 4.21), pentru aceeaşi lungime a secvenţei de date, la o valoare SNR=6dB, la iteraţia 5 BER este cuprins între 10^{-2} şi 10^{-3} (figura 4.23). Totodată am observat că în cazul unei estimări eronate a lui h rezultatele sunt mult mai slabe pentru ambele scheme.

În cazul iii) estimarea lui h am realizat-o cu algoritmul LS pentru transmisia secvenţei de învăţare utilizând modulaţia BPSK şi cu doi algoritmi LS şi MMSE utilizând tehnica de transmisie OFDM.

În cazul utilizării modulaţiei BPSK pentru secvenţa de învăţare, prima dată am comparat rezultatele obţinute cu cele două scheme (figura 4.17 şi 4.21), când estimarea la ambele scheme se face cu algoritmul LS. Pentru schema din figura 4.21 am considerat că estimările succesive sunt realizate utilizând acelaşi algoritm (LS).

Din rezultatele obţinute prezentate în figurile 4.24, 4.25 şi 4.26 am observat că odată cu creşterea secvenţei de învăţare, BER la aceeaşi valoare a SNR scade, deci estimarea lui h este mai bună.

Totodată din figurile 4.24 şi 4.25 unde secvenţele de învăţare sunt $N = 8$ şi respectiv $N = 18$, se poate observa că rezultatele obţinute cu schema din figura

4.21– care face o estimare iterativă a lui $\hat{h}$ - sunt mult mai bune decât cele obţinute cu schema din figura 4.17– care face estimarea lui $\hat{h}$ o singură dată. Odată cu creşterea lungimii secvenţei de învăţare (N=128) estimarea devine mai bună, astfel că rezultatele obţinute cu schema din figura 4.17 sunt mai bune (figura 4.26) decât cele obţinute cu schema din figura 4.21. Pentru a menţine rata de transmisie la valori ridicate este de preferat ca lungimea secvenţei de învăţare sa fie cât mai mică. De aceea în cazul estimării LS este de preferat utilizarea schemei din figura 4.21.

În cazul transmisiei secvenţei de învăţare utilizând tehnica OFDM, am făcut o comparaţie a rezultatelor BER/SNR pentru fiecare schemă în parte, cât şi între cele două scheme, pentru cei doi algoritmi menţionaţi LS şi MMSE. Astfel am analizat patru cazuri, şi anume:

1. Comparaţie între rezultatele obţinute cu schemele din figurile 4.17 şi 4.21 utilizând algoritmul LS atât în estimarea iniţială cât şi în estimările ulterioare.

În acest caz din analiza graficelor prezentate în figurile 4.30, 4.31 şi 4.32 rezultă că şi de această dată, odată cu creşterea lungimii secvenţei de învăţare, estimarea devine mai precisă. Pentru o secvenţă de învăţare de lungime mică (N=8 şi N=18) poate fi utilizată schema din figura 4.21, iar pentru o secvenţă de lungime mare (N=64) , caz în care estimarea este destul de precisă, poate fi utilizată schema din figura 4.17

2. Comparaţie între rezultatele obţinute cu schemele din figurile 4.17 şi 4.21 utilizând algoritmul MMSE în estimarea iniţială şi algoritmul LS în estimările ulterioare.

Utilizând algoritmul MMSE în schema din figura 4.17, se obţin rezultate mai bune (figura 4.33) decât cu algoritmul LS (figura 4.30) pentru secvenţele de învăţare de lungime mai mică (N=8). Totuşi pentru secvenţe de învăţare de lungime mică (când estimarea lui h este mai puţin bună) se recomandă schema din figura 4.21.

3. Comparaţie între rezultatele obţinute cu schema din figura 4.17 utilizând algoritmii LS si MMSE

Din rezultatele prezentate în figurile 4.36, 4.37 şi 4.38, am constatat că în cazul estimării MMSE rata erorii de bit este mai mică decât în cazul estimării LS, pentru secvenţe de învăţare de lungime mică (N=8 şi N=18). Pentru secvenţe de învăţare de lungime mare (N=64) performanţele de estimare corespunzătoare celor doi algoritmi sunt apropiate

4.Comparaţie între rezultatele obţinute cu schema din figura 4.21 utilizând algoritmii LS si MMSE.

În urma rezultatelor obţinute cu ajutorul schemei din figura 4.21, prezentate în figurile 4.39, 4.40 şi 4.41, am constatat că odată cu creşterea secvenţei de învăţare performanţele celor doi algoritmi devin apropiate. Pentru secvenţe de învăţare de lungime mică se obţin performanţe mai bune utilizând algoritmul MMSE decât algoritmul LS.

O concluzie finală ar fi aceea că schemele în care estimarea este făcută iterativ, oferă performante mai bune comparativ cu estimarea făcută o singură dată pe parcursul unui ciclu de egalizare, excepţie face cazul estimării perfecte unde rezultatele sunt aproximativ egale. Întrucât estimarea canalului nu este niciodată perfectă, se poate utiliza schema bloc în care estimarea este făcută iterativ.

Dintre cei doi algoritmi utilizaţi, am obţinut rezultate mai bune privind estimarea şi implicit BER cu algoritmul MMSE. În ce priveşte transmisia secvenţei de învăţare cu tehnica OFDM, aceasta oferă atât o viteză mai mare cât şi o estimare mai bună faţă de transmisia care utilizează modulaţia BPSK pentru aceeaşi lungime a secvenţei de biţi.

Bibliografie citată şi consultată

[1] C. Shannon, "A Mathematical Theory of Communication," Bell System Technical Journal, pp. 379-423 and 623-656, July and October 1948.

[2] C. Berrou, A. Glavieux, and P. Thitimajshina, "Near Shannon limit error correcting coding and decoding: Turbo codes," in IEEE int. conf. on Commun., pp. 1064-1070, 1993.

[3] C. Douillard and al. "Iterative correction of inter-symbol interference: Turbo equalization," European Trans. on Commun., vol. 6, pp. 507-511, Sept.-Oct. 1995.

[4] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: performances, analysis, design and iterative decoding," TDA Progress Rep. 42-126, Aug.1996.

[5] G. Forney, "Maximum-Likelihood sequence estimation of digital sequences in the presence of inter-symbol interference," IEEE Trans. on Information Theory, vol. 18, pp. 363-378, May 1972.

[6] A. Glavieux, C. Laot, and J. Labat, "Turbo equalization over a frequency selective channel," Proc. Int. Symposium on Turbo codes & related topics, Brest, France, pp. 96-102, Sept. 1997.

[7] A. Glavieux, C. Laot, and J. Labat, "Turbo equalization: Adaptive equalization and channel decoding jointly optimized," IEEE journal on selected areas in Commun., vol. 19, No 9, pp. 1744-1752, Sept. 2001.

[8] R. G. Gallager, "Low-density parity-check codes," IRE Trans. Inform. Theory, vol. IT-8, pp. 21–28, Jan. 1962.

[9] R. G. Gallager, Low-Density Parity-Check Codes. Cambridge, MA: MIT Press, 1963.

[10] S.-Y. Chung, G. D. Forney, Jr., T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," IEEE Commun. Letters, vol. 5, pp. 58–60, Feb. 2001.

[11] H. Imai and S. Hirakawa, "A new multilevel coding method using error correcting codes," IEEE Trans.Inform.The ory, vol. 23, pp. 371–377, May 1977

[12] J. Leech and N. J. A. Sloane, "Sphere packing and error-correcting codes," Canad.J.Math., vol. 23, pp. 718–745, 1971.

[13] Yong Soo Cho, Jaekwon Kim, Won Young Yang, Chung G. Kang "MIMO-OFDM Wireless Communications with MATLAB," October 2010.

[14] G. Caire and S. Shamai, "On the achievable throughput of a multiantenna Gaussian broad-cast channel," IEEE Trans.Inform.The ory, vol. 49, pp. 1691–1706, July 2003.

[15] H. Harashima and H. Miyakawa, "A method of code conversion for a digital communication channel with intersymbol interference," IEEE Trans.Commun. , vol. COM-20, pp. 774–780, Aug. 1972.

[16] M. Tomlinson, "New automatic equalizer employing modulo arithmetic," Electron. Lett., vol. 7, pp. 138–139, Mar. 1971.

[17] A. Ruiz, J. M. Cioffi, and S. Kasturia, "Discrete multiple tone modulation with coset coding for the spectrally shaped channel," IEEE Trans.Commun. vol. 40, pp. 1012–1029, June 1992.

[18] Alaa Ghait "Methodes pour l'Estimation de canal, Egalisation et Codage pour le Traitement Iteratif en présence d'interférences" Paris 2006.

[19] A. Klein, G. Kaleh, and P.W. Baier, "Zero forcing and minimum mean square error equalization for multi-user detection in code-division multiple access channels," IEEE trans. on vehicular technology, vol. 45, No. 2, pp. 276-287, May 1996.

[20] Bilal Riaz "EXIT Chart Analysis for Compressive Turbo Codes",Department of Electrical and Computer Engineering McGill University Montréal,Canada, 2009.

[21] W. E. Ryan, "A Turbo Code Tutorial", New Mexico State University.

[22] Turbo codes Encoding/Decoding & EXIT chart-Walid Boumerdassi, Etienne Collange & Team Space Busterss Georgia Tech Atlanta FALL 2010.

[23] C. Berrou and A. Glavieux, „Near optimum error correcting coding and decoding: Turbo codes", IEEE Transactions on Communications, vol. 44, pp. 1261.1271, October 1996.

[24] R. Steele and L. Hanzo, „Mobile Radio Communications: Second and Third Generation Cellular and WATM Systems", New York, USA, IEEE Press - John Wiley & Sons, 2nd ed., 1999.

[25] G. Bauch, H. Khorram, and J. Hagenauer, „Iterative equalization and decoding in mobile communications systems", European Personal Mobile Communications Conference, Bonn, Germany, pp. 301.312, 30 September.2 October 1997.

[26] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, „Optimal Decoding of Linear Codes for Minimising Symbol Error Rate", IEEE Transactions on Information Theory, vol. 20, pp. 284.287, March 1974.

[27] P. Robertson, E. Villebrun, and P. H¨oher, „A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain", Proceedings of the International Conference on Communications, Seattle, USA, pp. 1009.1013, June 1995.

[28] J. Hagenauer and P. Hoher, „A Viterbi algorithm with soft-decision outputs and its applications”, IEEE Globecom, pp. 1680.1686, 1989.

[29] J. Hagenauer, „Source-controlled channel decoding”, IEEE Transactions on Communications, vol. 43, pp. 2449. 2457, September 1995.

[30] C. Berrou, P. Adde, E. Angui, and S. Faudeil, „A low complexity soft-output Viterbi decoder architecture”, Proceedings of the International Conference on Communications, pp. 737.740, May 1993.

[31] P. Robertson, „Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes”, IEEE Globecom, pp. 1298.1303, 1994.

[32] M. Gertsman and J. Lodge, „Symbol-by-symbol MAP demodulation of CPM and PSK signals on Rayleigh flat-fading channels”, IEEE Transactions on Communications, vol. 45, pp. 788.799, July 1997.

[33] G. Bauch and V. Franz, „Iterative equalisation and decoding for the GSM-system”, Proceedings of IEEE Vehicular Technology Conference (VTC.98), Ottawa, Canada, pp. 2262.2266, IEEE, 18.21 May 1998.

[34] S. Benedetto and G. Montorsi, „Unveiling turbo codes: Some results on parallel concatenated coding schemes”, IEEE Transactions on Information Theory, vol. 42, pp. 409.428, March 1996.

[35] L. Hanzo, T. H. Liew, B. L. Yeap, „Turbo Coding, Turbo Equalisation and Space-Time Coding for Transmission over Wireless Channels”, Edit. John Wiley&Sons, 2002.

[36] Jan-Jaap van de Beek „Synchronization and Channel Estimation in OFDM Systems”
September 1998.

[37] Stkphane Le Goff, Alain Glavieux and Claude Berrou “Turbo-codes and high spectral efficiency modulation” IEEE International Conference. vol.2 , pp. 645 – 649, 01 May 1994.

[38] Jakob Dahl Andersen, “Turbo Codes Extended with Outer BCH Code”, Electronics Letters, vol. 32 No. 22, Oct. 1996.

[39] J. Dahl Andersen and V. V. Zyablov, “Interleaver Design for Turbo Coding”, Proc. Int. Symposium on Turbo Codes, Brest, Sept. 1997.

[40] Jakob Dahl Andersen, “Selection of Component Codes for Turbo Coding based on Convergence Properties”, Annales des Telecommunication, Special issue on iterated
decoding, June 1999.

[41] S. Benedetto and G. Montorsi, “Performance Evaluation of Turbo-codes”, Electronics Letters, vol. 31, No. 3, Feb. 1995.

[42] R. J. McEliece, E. R. Rodemich and J.-F. Cheng, "The Turbo Decision Algorithm", Presented at the 33rd Allerton Conference on Communication, Control and Computing, Oct. 1995.

[43] L. C. Perez, J. Seghers and D. J. Costello, Jr, "A Distance Spectrum Interpretation of Turbo Codes", IEEE Trans. on Inform. Theory, Vol. 42, No. 6, Nov. 1996.

[44] Steven S. Pietrobon, "Implementation and Performance of a Serial MAP Decoder for use in an Iterative Turbo Decoder", Proc. Int. Symposium on Information Theory, Whistler, Canada, Sept. 1995.

[45] G. Ungerboeck, "Channel coding with multi- level/phase signals", IEEE Trans. Inf. Theory, pp. 55-67, Jan. 1982.

[46] M. Eyuboglu, G. D. Forney, P. Dong, G. Long, "Advanced modulation techniques for V.Fast", Eur. Trans. on Telecom., pp. 243-256, May 1993.

[47] S. Benedetto and G.Montorsi, "Design of parallel con-catenated codes", IEEE Trans. Comm., pp. 591-600, May 1996.

[47] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes", IEEE Trans. Inf. Theory, pp. 429-445, Mar. 1996.

[49] D. Arnold and G. Meyerhans, "The realization of the turbo-coding system", Semester Project Report, Swiss Fed. Inst. of Tech., Zurich, Switzerland, July, 1995.

[50] A. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes", IEEE JSAC, pp. 260-264, Feb. 1998.

[51] D. Divsalar and F. Pollara, "Turbo codes for PCS applications", Proc. 1995 Int. Conf. Comm., pp. 54-59.

[52] A. Aaron and B. Girod, "Compression with side information using turbo codes," in Proc. IEEE Data Compr. Conf., pp. 252-261, Snowbird, UT, April 2002.

[53] A. Ashikhmin, G. Kramer, and S. ten Brink, "Code rate and the area under extrinsic information transfer curves," in Proc. IEEE Int. Symp. Information Theory (ISIT 2002), p. 115, Lausanne, Switzerland, July 2002.

[54] J. Bajcsy, "Course notes ECSE 686B – Codes and Graphs," Department of Electrical and Computer Engineering, McGill University, Winter 2006.

[55] J. Bajcsy, C. V. Chong, J. A. Hunziker, D. A. Garr, and H. Kobayashi, "Iterative decoding in some existing systems," IEEE J. Select. Areas Commun., vol. 19, no. 5, pp. 883 – 890, May 2001.

[56] S. ten Brink, "Design of Concatenated Coding Schemes based on Iterative Decoding Convergence," Ph.D. dissertation, University of Stuttgart, Germany, 2001.

[57] S. ten Brink, "Convergence behaviour of iteratively decoded parallel concatenated codes," IEEE Trans. Commun., vol. 40, pp. 1727-1737, October 2001.

[58] T. M. Cover, J. A. Thomas, "Elements of Information Theory", New York: Wiley-Interscience, 1991.

[59] I. Deslauriers and J. Bajcsy, "Serial Turbo Coding for Data Compression and the Slepian-Wolf Problem", in Proc. of the IEEE Information Theory Workshop, pp. 296-299, Paris, France, March 2003.

[60] D. Divsalar, S. Dolinar, F. Pollara, and R. J. McEliece, "Transfer function bounds on the performance of turbo codes," TDA Progress Report 42–122, pp. 44 – 55, August 1995.

[61] N. Dütsch, "Code optimization for lossless turbo source coding," in Proc. IST Mobile & Wireless Communications Summit, paper 197, Dresden, Germany, June 2005.

[62] V. C. Gaudet and P. G. Gulak, "A 13.3-Mb/s 0.35-um CMOS analog Turbo decoder IC with a configurable interleaver," IEEE J. Solid-State Circuits, vol. 38, no. 11, pp. 2010-2015, November 2003.

[63] J. Haganeuer and M. Winklhofer, "The Analog Decoder," in Proc. of IEEE Int. Symp. On Information Theory (ISIT 98), p. 145, Boston, USA, August 1998.

[64] J. Haganeuer, A. Barros, and A. Schaefer, "Lossless turbo source coding with decremental redundancy," in Proc. Fifth Int. ITG Conference on Source and Channel Codes, pp. 333-339, Erlangen, Germany, January 2004.

[65] J. Haghighat, W. Hamouda, and M. Soleymani, "Design of lossless turbo source encoders," IEEE Signal Proc. Letters, vol. 13, no. 8, pp. 453-456, August 2006.

[66] M. S. C. Ho, S. S. Pietrobon, and T. C. Giles. "Interleavers for punctured turbo codes," in Proc. IEEE Asia-Pacific Conf. on Commun., pp. 520 – 524, Singapore, November 1998.

[67] Y. Kim, B. J. Jeong, J. Chung, C. Hwang, J. S. Ryu, K. Kim, and Y. K. Kim, "Beyond 3G: Vision, requirements and enabling technologies," IEEE Commun. Mag., vol. 41, no. 3, pp. 120 – 124, March 2003.

[68] L. -N. Lee, A. R. Hammons, F. -W. Sun, and M. Eroz, "Application and Standardization of Turbo Codes in Third Generation High Speed Wireless Data Services," IEEE Trans. Vehicular Technology, vol. 49, no. 6, pp. 2198-2207, November 2000.

[69] H. Leib, "Course Notes ECSE 620B – Information Theory and Coding," Department of Electrical and Computer Engineering, McGill University, Winter 2008.

[70] A. D. Liveris, Z. Xiong, and C. N. Georghiades, "Compression of binary sources with side information at the decoder using LDPC codes," IEEE Communication Letters, vol. 6, no. 10, pp. 440-442, October 2002.

[71] A. D. Liveris, Z. Xiong, and C. N. Georghiades, "Distributed compression of binary sources using convolutional parallel and serial concatenated convolutional codes," in Proc. of IEEE Data Compr. Conf., pp. 193-202, Snowbird, UT, March 2003.

[72] P. Mitran and J. Bajcsy, "Turbo source coding: A noise-robust approach to data compression," in Proc. of IEEE Data Compr. Conf., p.465, Snowbird, UT, Aprilie 2002.

[73] P. Mitran, "Design and Applications of Turbo Source Codes," M. Eng. Thesis, McGill University, Montreal, QC, Canada, 2002.

[74] G. Montorsi and S. Benedetto, "Design of fixed-point iterative decoders for concatenated codes with interleavers," IEEE J. Select. Areas Commun., vol. 19, pp. 871–882, May 2001.

[75] J. Proakis, Digital Communications, New York: McGraw Hill, 1989.

[76] B. Riaz, "Performance Analysis of Turbo Compressive Codes," B. Eng. Thesis, McGill University, Montreal, QC, Canada, December 2006.

[77] D. Slepian and J. K. Wolf, "Noiseless Coding of Correlated Information Sources," IEEE Transactions on Information Theory, pp. 471-480, July 1973.

[78] M. R. Soleymani, Y. Gao, and U. Vilaipornsawai, Turbo Coding for Satellite and Wireless Communications, Norwell, MA: Kluwer Academic Publishers, 2002.

[79] G. Titericz, R. D. Souza, J. Garcia-Frias, and G. I. Shamir, "Comparing Different Transmission Strategies Using Turbo Codes for Nonuniform Memoryless Sources," in Proc. of IEEE Int. Conf. on Communications, pp. 2722-2727, Glasgow, Scotland, June 2007.

[80] Z. Tu, J. Li, and R. S. Blum, "Compression of a binary source with side information using parallel concatenated convolutional codes," in Proc. IEEE Global Commun. Conf., vol. 1, pp. 46-50, Dallas, TX, November 2004.

[81] M. Tüchler, "Convergence prediction for iterative decoding of threefold concatenated systems," in Proc. IEEE Global Commun. Conf., pp. 1358 –1362, Taipei, Taiwan, November 2002.

[82] A. D. Wyner, "On source coding with side information at the decoder," IEEE Trans. Inform. Theory, vol. IT-21, pp. 294–300, May 1975.

[83] Y. Zhao and J. Garcia-Frias, "Data compression of correlated non-binary sources using punctured turbo codes," in Proc. IEEE Data Compr. Conf., pp. 242-251, Snowbird, UT, April 2002.

[84] Y. Zhao and J. Garcia-Frias, "Joint estimation and compression of correlated nonbinary sources using punctured turbo codes," IEEE Trans. Communications, vol. 53, no. 3, pp. 385-390, March 2005.

[85] C.Gordan, **L.Morgoş**, R.Reiz: „Detection and estimation of linear FM signals", International Symposium on Signals, Circuits & Systems, ISSCS 2005, 14-15.07.2005, Iaşi, pp. 705-708, ISBN 0-7803-9029-6, Indexată ISI

[86] **L.Morgos**, S.Popa, C.Gordan: "Rayleigh Fading Channels in Digital Communication Systems", The fourth International PhD Student's workshop, IWCIT 2005, Ostrava, Czech Republic, 4th International PhD Student's workshop, IWCIT 2005, Ostrava, Czech Republic, pp. 271-274, ISBN 80-248-0906-0;

[87] Popa, **L.Morgoş**, N. Draghiciu, S. Castrase- "Fading types in mobile radio communication systems" 8th International Conference on Engineering of Modern Electric Systems 26-28 May 2005, Oradea – România. Analele Universităţii din Oradea, Fascicula Electrotehnică, Secţiunea Electronică pag. 112-115, ISSN 1454-9239.

[88] S.Popa, S. Castrase, N. Drăghiciu, **L.Morgoş** „Fading Channels Modelation in Digital Communication Systems", 8 International Workshop CPEE Wilasky, 14-16.10.2007 Poland, pag.41-44, ISSN 1731-6103.

[89] **L.Morgos**, Nistor Daniel Trip " Considerations about the Modeling of Software Defined Radio for Mobile Communications Networks" Journal of Electrical and Electronics Engineering 2009, Oradea –România., ISSN 1844-6035.

[90] S. Popa, **L. Morgoş**, A. Şchiop „Multicarrier modulation techniques (MCM) used to combat frecvency-selective fading", National Symposium of Theoretical Electrical Engineering Bucureşti, Romania 12.10.2007, pag.330-335, ISBN978-973-718-899-1.

[91] **L.Morgos** "Turbo Decoding Using BCJR Algorithm" Journal of Electrical and Electronics Engineering Vol.6 Nr1 2013, Oradea – România. pag.87, ISSN 1844-6035.

Anexa 1

```
   %Program care utilizeaza turbocodarea la emisie, iar la receptie o
%egalizare si o turbodecodare.
%La receptie se face o estimare iterativa utilizand algoritmul
%LS a secventei transmise,si o egalizare MMSE
clear all;
close all;
EbN0_db_vect= [0:1:4];% Diferite valori pentru raportul semnal zgomot (SNR)care
vor fi utilizate pentru
%a vizualiza evolutia ratei erorii pe bit (BER)

% polinomul generator
g=[1 1 1;1 0 1];
%  parameterii polinomului generator
[n,K] = size(g);
% numarul starilor este 2^m
m = K-1;
%Rata codului
R=1/K;
marime_mesaj=128;% numarul bitilor generati aleator de bloc
marime_bloc = marime_mesaj+m;%pentru a termina trellisul este nevoie de m biti
%Determinarea pozitiei bitilor dupa ce trec prin interlaver   :poz
[random, poz] = sort(rand(1,marime_bloc));
i_ber=0; %Indice pentru  BER
ber=zeros(1,length(EbN0_db_vect));
nr_max_de_erori=500;

for i_ber=1:length(EbN0_db_vect)
   EbN0_db=EbN0_db_vect(i_ber);

   Nr_erori_iteratia_1_mmse=0;
   Nr_erori_iteratia_2_mmse=0;
   Nr_erori_iteratia_3_mmse=0;
   Nr_erori_iteratia_4_mmse=0;
   Nr_erori_iteratia_5_mmse=0;
   nr_erori_mmse=0;
   Nr_erori_iteratia_1_mmseLS=0;
   Nr_erori_iteratia_2_mmseLS=0;
   Nr_erori_iteratia_3_mmseLS=0;
   Nr_erori_iteratia_4_mmseLS=0;
   Nr_erori_iteratia_5_mmseLS=0;
   nr_erori_mmseLS=0;
   numar_secvente_generate=0;
   while  Nr_erori_iteratia_5_mmseLS<nr_max_de_erori

      MesajIn = round(rand(1, marime_mesaj));
```

```
     %---------------------- Codor -------------------------------
          IesireTurboCodor = TurboCodor(g, MesajIn, poz);
    bitiicodati= zeros(1,3*length(IesireTurboCodor(1,:)));
      for i=1:marime_bloc
         bitiicodati(3*i-2:3*i) = (IesireTurboCodor(:,i)');
      end
    bitiicodati;
     % modulare BPSK
    Simb_codate = 2*bitiicodati - 1;
    N=length(bitiicodati);
    %---------------------- Canal multicale------------
    h = [0.2 0.9 0.3];
    [iesire_canal] = canal(h,N,Simb_codate,EbN0_db);
    EbN0=10^(EbN0_db/10);
    sigmaB_2= 1/(2*R*EbN0);
   %---------------------- estimare--------------------
   N1=16; %lungime secventa invatate
   secv_inv=round(rand(1, N1));
   secv_inv_cod=2*secv_inv-1;
   [iesire_canal_secv_inv] = canal(h,N1,secv_inv_cod,EbN0_db);
   [h_est]=estimare0_LS(secv_inv,iesire_canal_secv_inv);
   %---------------------- egalizare--------------------
   [biti_estimati_zf,biti_estimati_mmse]
=egalizare_zf_mmse(iesire_canal,h_est,N,EbN0_db);
   nr_erori_mmse1 = size(find([bitiicodati- biti_estimati_mmse]),2);
   nr_erori_mmse = nr_erori_mmse + nr_erori_mmse1;
   Simb_transmise_turbodec_mmse=2*biti_estimati_mmse-1;
    % Turbo-decodor
   %-------------------------------------------------------
      Lc=2/sigmaB_2;

    In_dec_mmse = [ Simb_transmise_turbodec_mmse(1:3:end);
    Simb_transmise_turbodec_mmse(2:3:end);
    Simb_transmise_turbodec_mmse(3:3:end)];
    extrinsec2_mmse = zeros(marime_bloc,1);% valoarea extrinseca de la
decodorul 2
    apriori1_mmse = zeros(marime_bloc,1); % valoarea apriorii la primul decodor

 %_____________________________________________________________
     %iteratia 1

[Le1_it1_mmse,Le2_it1_mmse,nr_erori1_mmse,biti_estimati1_mmse]=iteratie(g,M
esajIn,In_dec_mmse,Lc, marime_bloc, apriori1_mmse,extrinsec2_mmse,poz);
      Nr_erori_iteratia_1_mmse=Nr_erori_iteratia_1_mmse+nr_erori1_mmse;
  %_____________________________________________________________
     %iteratia 2

[Le1_it2_mmse,Le2_it2_mmse,nr_erori2_mmse,biti_estimati2_mmse]=iteratie(g,M
esajIn,In_dec_mmse,Lc, marime_bloc, Le1_it1_mmse,Le2_it1_mmse,poz);
      Nr_erori_iteratia_2_mmse=Nr_erori_iteratia_2_mmse+nr_erori2_mmse;
  %_____________________________________________________________
```

```
      %iteratia 3

[Le1_it3_mmse,Le2_it3_mmse,nr_erori3_mmse,biti_estimati3_mmse]=iteratie(g,M
esajIn,In_dec_mmse,Lc, marime_bloc, Le1_it2_mmse,Le2_it2_mmse,poz);
      Nr_erori_iteratia_3_mmse=Nr_erori_iteratia_3_mmse+nr_erori3_mmse;
   %_____________________________________________________________
      %iteratia 4

[Le1_it4_mmse,Le2_it4_mmse,nr_erori4_mmse,biti_estimati4_mmse]=iteratie(g,M
esajIn,In_dec_mmse,Lc, marime_bloc, Le1_it3_mmse,Le2_it3_mmse,poz);
      Nr_erori_iteratia_4_mmse=Nr_erori_iteratia_4_mmse+nr_erori4_mmse;
      %_____________________________________________________________
      %iteratia 5

[Le1_it5_mmse,Le2_it5_mmse,nr_erori5_mmse,biti_estimati5_mmse]=iteratie(g,M
esajIn,In_dec_mmse,Lc, marime_bloc, Le1_it4_mmse,Le2_it4_mmse,poz);
      Nr_erori_iteratia_5_mmse=Nr_erori_iteratia_5_mmse+nr_erori5_mmse;


      %---------------------- egalizare------------------
   [h_estLS]=estimare_LS(marime_bloc,biti_estimati5_mmse,iesire_canal,g,poz);

   [biti_estimati_zfLS,biti_estimati_mmseLS]
=egalizare_zf_mmse(iesire_canal,h_estLS,N,EbN0_db);

nr_erori_mmse1LS = size(find([bitiicodati- biti_estimati_mmseLS]),2);
   nr_erori_mmseLS = nr_erori_mmseLS + nr_erori_mmse1LS;
    Simb_transmise_turbodec_mmseLS=2*biti_estimati_mmseLS-1;

      % Turbo-decodor
      %-----------------------------------------------------
      Lc=2/sigmaB_2;

      In_dec_mmseLS = [ Simb_transmise_turbodec_mmseLS(1:3:end);
         Simb_transmise_turbodec_mmseLS(2:3:end);
         Simb_transmise_turbodec_mmseLS(3:3:end)];

       extrinsec2_mmseLS = zeros(marime_bloc,1);% valoarea extrinseca de la
decodorul 2
       apriori1_mmseLS = zeros(marime_bloc,1); % valoarea apriorii la primul
decodor
        %_____________________________________________________________
      %iteratia 1

[Le1_it1_mmseLS,Le2_it1_mmseLS,nr_erori1_mmseLS,biti_estimati1_mmseLS]=ite
ratie(g,MesajIn,In_dec_mmseLS,Lc, marime_bloc,
apriori1_mmseLS,extrinsec2_mmseLS,poz);

Nr_erori_iteratia_1_mmseLS=Nr_erori_iteratia_1_mmseLS+nr_erori1_mmseLS;
      %_____________________________________________________________
      %iteratia 2
```

```
[Le1_it2_mmseLS,Le2_it2_mmseLS,nr_erori2_mmseLS,biti_estimati2_mmseLS]=ite
ratie(g,MesajIn,In_dec_mmseLS,Lc, marime_bloc,
Le1_it1_mmseLS,Le2_it1_mmseLS,poz);

Nr_erori_iteratia_2_mmseLS=Nr_erori_iteratia_2_mmseLS+nr_erori2_mmseLS;
      %_____________________________________________________________
      %iteratia 3

[Le1_it3_mmseLS,Le2_it3_mmseLS,nr_erori3_mmseLS,biti_estimati3_mmseLS]=ite
ratie(g,MesajIn,In_dec_mmseLS,Lc, marime_bloc,
Le1_it2_mmseLS,Le2_it2_mmseLS,poz);

Nr_erori_iteratia_3_mmseLS=Nr_erori_iteratia_3_mmseLS+nr_erori3_mmseLS;
      %_____________________________________________________________
      %iteratia 4

[Le1_it4_mmseLS,Le2_it4_mmseLS,nr_erori4_mmseLS,biti_estimati4_mmseLS]=ite
ratie(g,MesajIn,In_dec_mmseLS,Lc, marime_bloc,
Le1_it3_mmseLS,Le2_it3_mmseLS,poz);

Nr_erori_iteratia_4_mmseLS=Nr_erori_iteratia_4_mmseLS+nr_erori4_mmseLS;
      %_____________________________________________________________
      %iteratia 5

[Le1_it5_mmseLS,Le2_it5_mmseLS,nr_erori5_mmseLS,biti_estimati5_mmseLS]=ite
ratie(g,MesajIn,In_dec_mmseLS,Lc, marime_bloc,
Le1_it4_mmseLS,Le2_it4_mmseLS,poz);

Nr_erori_iteratia_5_mmseLS=Nr_erori_iteratia_5_mmseLS+nr_erori5_mmseLS;

      numar_secvente_generate=numar_secvente_generate+1;  % numarul de
secvente generate
   end
        numar_biti_generati=numar_secvente_generate*marime_mesaj;
        ber1_mmse(i_ber)=Nr_erori_iteratia_1_mmse/numar_biti_generati;
        ber2_mmse(i_ber)=Nr_erori_iteratia_2_mmse/numar_biti_generati;
        ber3_mmse(i_ber)=Nr_erori_iteratia_3_mmse/numar_biti_generati;
        ber4_mmse(i_ber)=Nr_erori_iteratia_4_mmse/numar_biti_generati;
        ber5_mmse(i_ber)=Nr_erori_iteratia_5_mmse/numar_biti_generati;
        ber6_mmse(i_ber)=nr_erori_mmse/numar_biti_generati;
        ber1_mmseLS(i_ber)=Nr_erori_iteratia_1_mmseLS/numar_biti_generati;
        ber2_mmseLS(i_ber)=Nr_erori_iteratia_2_mmseLS/numar_biti_generati;
        ber3_mmseLS(i_ber)=Nr_erori_iteratia_3_mmseLS/numar_biti_generati;
        ber4_mmseLS(i_ber)=Nr_erori_iteratia_4_mmseLS/numar_biti_generati;
        ber5_mmseLS(i_ber)=Nr_erori_iteratia_5_mmseLS/numar_biti_generati;
        ber6_mmseLS(i_ber)=nr_erori_mmseLS/numar_biti_generati;
end
figure;
```

```
    semilogy(EbN0_db_vect,ber1_mmse,'ms-',EbN0_db_vect,ber2_mmse,'k^-
',EbN0_db_vect,ber3_mmse,'gv-',EbN0_db_vect,ber4_mmse,'ok-
',EbN0_db_vect,ber5_mmse,'dr-',EbN0_db_vect, ber6_mmse,'b<-','linewidth',2);
    xlabel('Eb/No, dB','FontSize',14)
    ylabel('BER','FontSize',14)
    hold on
   semilogy(EbN0_db_vect,ber1_mmseLS,'ms-.',EbN0_db_vect,ber2_mmseLS,'k^-
.',EbN0_db_vect,ber3_mmseLS,'gv-.',EbN0_db_vect,ber4_mmseLS,'ok-
.',EbN0_db_vect,ber5_mmseLS,'dr-.',EbN0_db_vect, ber6_mmseLS,'b<-
.','linewidth',2);
    grid on
    hold off
    legend('iteratia1-mmse','iteratia2-mmse','iteratia3-mmse','iteratia4-mmse',
'iteratia5-mmse','nedecodat-mmse','iteratia1-mmseLS','iteratia2-
mmseLS','iteratia3-mmseLS','iteratia4-mmseLS', 'iteratia5-mmseLS','nedecodat-
mmseLS')
```

Funcţii utilizate in cadrul programului:

1. canal.m

```
function [y] = canal(h,N,s,Eb_N0_dB)

K=3;
L  = length(h);
iesire_canal = conv(s,h);
n = 1/sqrt(2)*[randn(1,N+length(h)-1) + j*randn(1,N+length(h)-1)];% zgomot
y = iesire_canal + 10^(-Eb_N0_dB/20)*n;
end
```

2. egalizare_zf_mmse.m

```
function [biti_est_zf,biti_est_mmse] =egalizare_zf_mmse(y,h,N,Eb_N0_dB);


L  = length(h);
K=3;

% egalizare ZF
H = toeplitz([h([2:end]) zeros(1,2*K+1-L+1)], [ h([2:-1:1]) zeros(1,2*K+1-
L+1) ]);
d  = zeros(1,2*K+1);
d(K+1) = 1;
c_zf  = [inv(H)*d.'].';
y_zf = conv(y,c_zf);
y_zf = y_zf(K+2:end);
y_zf = conv(y_zf,ones(1,1)); % convolutie
y_esant_zf = y_zf(1:1:N);  % esantionare
```

```
% egalizare MMSE
autocor_h = conv(h,fliplr(h));
H = toeplitz([autocor_h([3:end]) zeros(1,2*K+1-L)], [ autocor_h([3:end])
zeros(1,2*K+1-L) ]);
H = H + 1/2*10^(-Eb_N0_dB/10)*eye(2*K+1);
d  = zeros(1,2*K+1);
d([-1:1]+K+1) = fliplr(h);
c_mmse  = [inv(H)*d.'].';
y_mmse = conv(y,c_mmse);
y_mmse = y_mmse(K+2:end);
y_mmse = conv(y_mmse,ones(1,1)); % convolutie
y_esant_mmse = y_mmse(1:1:N);  % esantionare
% decizie
biti_est_zf = real(y_esant_zf)>0;
biti_est_mmse = real(y_esant_mmse)>0;
```

3. estimare0_LS.m

```
  function [h_est]=estimare0_LS(secventa_test,y)
Nr=3;%lungimea lui h
% se face modularea
secv_test_mod_BPSK=2*secventa_test-1;
x=secv_test_mod_BPSK;

Xc=zeros(1,length(x)+ Nr-1);
for i=1:length(x)
   Xc(i)=x(i);
end
Xr=zeros(1,Nr);
Xr(1)=x(1);
Xr=Xr';
X = toeplitz(Xc,Xr);
 h_est = inv(X.'*X) * X.' * y';
 h_est=real(h_est');
```

4. estimare_LS.m

```
function [h_est]=estimare_LS(marime_bloc,secventa_rec,y,g,poz)
 %---------------------- Codor -----------------------------------------
 IesireTurboCodorLS = TurboCodor(g, secventa_rec, poz);
 bitiicodatiLS= zeros(1,3*length(IesireTurboCodorLS(1,:)));
    for i=1:marime_bloc
 bitiicodatiLS(3*i-2:3*i) = (IesireTurboCodorLS(:,i)');
   end
    bitiicodatiLS;
   Simb_codateLS = 2*bitiicodatiLS - 1;
    N=length(bitiicodatiLS);
 Nr=3;%lungimea lui h
x=Simb_codateLS;
Xc=zeros(1,length(x)+ Nr-1);
```

```
for i=1:length(x)
   Xc(i)=x(i);
end
Xr=zeros(1,Nr);
Xr(1)=x(1);
Xr=Xr';
X = toeplitz(Xc,Xr);
 h_est = inv(X.'*X) * X.' * y';
 h_est=real(h_est');
```

5. iteraţie.m

```
function[Le1,Le2,nr_erori,X_obt]=iteratie(g,MesajIn,In_dec_cu_zgomot,Lc,marime_
bloc,apriori1,extrinsec2,poz)
  % Primul decodor
  apriori1(poz) = extrinsec2;
  %-----------Iesirea primului decoder
 [LMAP1 extrinsec1] = TurboDecodor1(g, In_dec_cu_zgomot,Lc, marime_bloc,
apriori1);
  %-----------Stocarea  valorii extrinsec1 in Le1
  Le1 = extrinsec1;
 % Al doilea decodor
  apriori2 = extrinsec1(poz);
  %-----------Iesirea celui de-al doile decodor
[LMAP2 extrinsec2] = TurboDecodor2(g, In_dec_cu_zgomot,Lc, marime_bloc,
apriori2, poz);
%-----------Stocarea  valorii extrinsec2 in Le2
Le2 = extrinsec2;
X_obt=(LMAP1(1:end-2)>0)';
 dif=MesajIn-X_obt;
nr_erori=sum(abs(dif));
```

6. trellis.m

```
 function [matricea_trellis, matricea_paritate] = trellis(cod_g);
[n,K] = size(cod_g);
m = K - 1;
num_starilor = 2^m;
matricea_trellis= zeros(num_starilor);
matricea_paritate= zeros(num_starilor);
for s=1: num_starilor
cont_zec_s=s-1 ;
i=1;
while  cont_zec_s >=0 & i<=m
      cont_bin_s(i) = rem( cont_zec_s,2); %restul impartirii la 2
      cont_zec_s  = (cont_zec_s-  cont_bin_s(i))/2;
       i=i+1;
end
cont_bin_s=cont_bin_s(m:-1:1);
```

```
  % urmatoarea stare cand intrarea este zero
   a = rem( cod_g(1,:)*[0 cont_bin_s ]', 2 );
     v = cod_g(2,1)*a;
   for j = 1:K-1
         v = xor(v, cod_g(2,j+1)*cont_bin_s(j));    %suma modulo 2
   end;
starea0 = [a  cont_bin_s(1:m-1)];
d=2.^(m-1:-1:0);
j=starea0*d'+1;
matricea_trellis(s,j)= -1;
matricea_paritate(s,j)= 2*v-1;
% urmatoarea stare cand intrarea este 1
a = rem( cod_g(1,:)*[1 cont_bin_s]', 2 );
   v = cod_g(2,1)*a;
   for j = 1:K-1
     v = xor(v, cod_g(2,j+1)*cont_bin_s(j));
   end;
starea1 = [a cont_bin_s(1:m-1)];
d=2.^(m-1:-1:0);
j=starea1*d'+1;
matricea_trellis(s,j)= 1;
matricea_paritate(s,j)= 2*v-1;
end
```

7. TurboCodor.m

```
function IesireTurboCodor = TurboCodor(g, MesajIn, poz);
% parametri polinomului generator
[n,K] = size(g);
% numarul starilor este m
m = K-1;
% seteaza starile la zero
stare = zeros(1,m);
[b,lungime]=size(MesajIn);
iesirecodor11=[];
iesirecodor12=[];
iesirecodor22=[];
 for x=1:lungime+m
       if x<=lungime
       mes=MesajIn(1,x);
    else
       mes=rem(g(1,2:K)*(stare)',2);
       end
    intrare=rem(g(1,:)*([mes stare])',2);
    for i=1:n
       iesire(i) = g(i,1)*intrare;
       for j = 2:K
          temp=g(i,j)*stare(j-1);
          iesire(i) = xor(iesire(i),temp);
       end;
    end
```

```
   stare = [intrare, stare(1:m-1)];
   iesirecodor11=[iesirecodor11,mes];
   iesirecodor12=[iesirecodor12,iesire(i)];

end
% codor 2
for x=1:lungime+m
   mesajintretesut=iesirecodor11(1,poz(1,x));
      intrare=rem(g(1,:)*([mesajintretesut stare])',2);
   for i=1:n
      iesire(i) = g(i,1)*intrare;
      for j = 2:K
         temp=g(i,j)*stare(j-1);
         iesire(i) = xor(iesire(i),temp);
      end;
         if i==1
         temp1=iesire(i);
      end
   end
   stare = [intrare, stare(1:m-1)];
   % concateneaza noul element
   iesirecodor22=[iesirecodor22,iesire(2)];
end
IesireTurboCodor = [ iesirecodor11;
                iesirecodor12;
                iesirecodor22 ];
```

8. TurboDecodor1.m

```
function [LMAP LE] = TurboDecodor1(g, In_dec_cu_zgomot,Lc, marime_bloc,
apriori)
x=In_dec_cu_zgomot(1,:);
p1=In_dec_cu_zgomot(2,:);
% parametrii generatorului polinomial
[n,K] = size(g);
% dimensiunea memoriei
m = K-1;
% numarul starilor
nr_stari=2^m;
[matricea_trelis, matricea_paritate]= trellis(g);
LMAP=zeros(marime_bloc,1);
LE=zeros(marime_bloc,1);
%Initializarea  Decodorului
alpha= zeros(nr_stari,marime_bloc+1);
alpha(1,1)=1;
beta= zeros(nr_stari,marime_bloc+1);
beta(1,end)=1;
gamma(1,1:marime_bloc)= struct('matricea_gamma',zeros(size(matricea_trelis)));
for k=1:marime_bloc
```

```
   gamma(k).matricea_gamma= exp(1/2*(matricea_trelis.*(apriori(k) + Lc*x(k)')
+ Lc*matricea_paritate.*p1(k)'));
   gamma(k).matricea_gamma= gamma(k).matricea_gamma.*(matricea_trelis>0 |
matricea_trelis<0);
end
%Calculul lui alpha1
for k=1:marime_bloc
   alpha(:,k+1)= (gamma(k).matricea_gamma)*(alpha(:,k));
   alpha(:,k+1)=alpha(:,k+1)/sum(alpha(:,k+1));
end
%Calcularea lui beta1
for k=marime_bloc:-1:1
   beta(:,k)=(gamma(k).matricea_gamma)'*(beta(:,k+1));
  beta(:,k)=beta(:,k)/sum(beta(:,k));
end

for k=1:marime_bloc

sumSplus=alpha(:,k)'*(gamma(k).matricea_gamma.*(matricea_trelis>0))'*beta(:,k
+1);

sumSminus=alpha(:,k)'*(gamma(k).matricea_gamma.*(matricea_trelis<0))'*beta(
:,k+1);
   LMAP(k)=log(sumSplus/sumSminus);
end

LE=LMAP-apriori-Lc*x';

end
```

9. TurboDecodor2.m

```
function [LMAP LE] = TurboDecodor2(g, In_dec_cu_zgomot,Lc, marime_bloc, apriori,
poz);
x=In_dec_cu_zgomot(1,poz);
%p1=In_dec_cu_zgomot(2,:);
p2=In_dec_cu_zgomot(3,:);
% parametrii polinomului generator
[n,K] = size(g);
% marimea memoriei
m = K-1;
% numarul starilor
nr_stari=2^m;
[matricea_trelis, matricea_paritate]= trellis(g);
LMAP=zeros(marime_bloc,1);
LE=zeros(marime_bloc,1);
%Initializare decodor  MAP2
alpha= zeros(nr_stari,marime_bloc+1);
alpha(1,1)=1;
beta= zeros(nr_stari,marime_bloc+1);
beta(:,end)=1/(nr_stari)*ones(nr_stari,1);
```

```
gamma(1,1:marime_bloc)= struct('matricea_gamma',zeros(size(matricea_trelis)));
gammaE(1,1:marime_bloc)=
struct('matricea_gamma',zeros(size(matricea_trelis)));

for k=1:marime_bloc
   gamma(k).matricea_gamma= exp(1/2*(matricea_trelis.*(apriori(k) + Lc*x(k)')
+ Lc*matricea_paritate.*p2(k)));
   gamma(k).matricea_gamma= gamma(k).matricea_gamma.*(matricea_trelis>0 |
matricea_trelis<0);

end

for k=1:marime_bloc
   alpha(:,k+1)= (gamma(k).matricea_gamma)*(alpha(:,k));
   alpha(:,k+1)=alpha(:,k+1)/sum(alpha(:,k+1));
end
for k=marime_bloc:-1:1
   beta(:,k)= (gamma(k).matricea_gamma)'*(beta(:,k+1));
  beta(:,k)=beta(:,k)/sum(beta(:,k));
end

for k=1:marime_bloc

sumSplus=alpha(:,k)'*(gamma(k).matricea_gamma.*(matricea_trelis>0))'*beta(:,k
+1);

sumSminus=alpha(:,k)'*(gamma(k).matricea_gamma.*(matricea_trelis<0))'*beta(
:,k+1);
   LMAP(k)=log(sumSplus/sumSminus);
end
LE=LMAP-apriori-Lc*x';
end
```

Anexa 2

```
%Egalizor turbo cu estimare LS
%(atat estimarea initiala cat si iterativa)

% ber.m
clear all;
close all;
clc;
Eb=2; %energia bitului
EbN0_db =[0:6]; % SNR
EbN0=10.^(EbN0_db/10);
No=Eb./EbN0;
varianta=No./2;
h=[0.302 0.725 0.456];
lungimea_secventei_de_date =128;
nr_max_de_erori=100;

for i=1:length(EbN0_db)
    Nr_erori_iteratia_0=0;
    Nr_erori_iteratia_1=0;
    Nr_erori_iteratia_2=0;
    Nr_erori_iteratia_3=0;
    Nr_erori_iteratia_4=0;
    Nr_erori_iteratia_5=0;
    Nr_erori_iteratia_1LS=0;
    Nr_erori_iteratia_2LS=0;
    Nr_erori_iteratia_3LS=0;
    Nr_erori_iteratia_4LS=0;
    Nr_erori_iteratia_5LS=0;

    numar_secvente_generate=0;
    while  Nr_erori_iteratia_5LS<nr_max_de_erori
%_____________________________________________________
    % iteratia 0
    [bitii_transmisi,simboluri_codate,Lapos_dec_a0,Lapos_dec_b0,...
    Lex_deint_apriori_dec0,y,numar_erori0,rata0,y1,h_est]=...
    iteratia_0(varianta(i),lungimea_secventei_de_date,h);
    Nr_erori_iteratia_0=Nr_erori_iteratia_0+numar_erori0;
    %_____________________________________________________
    %o iteratie
    [Lapos_dec_a1,Lapos_dec_b1,Lex_deint_apriori_dec1,...
    numar_erori1,rata1]=iteratia_N(varianta(i),...
    Lapos_dec_a0,Lex_deint_apriori_dec0,y,bitii_transmisi,h_est);
    Nr_erori_iteratia_1=Nr_erori_iteratia_1+numar_erori1;
    %_____________________________________________________
    %doua iteratii
    [Lapos_dec_a2,Lapos_dec_b2,Lex_deint_apriori_dec2,...
```

```
numar_erori2,rata2]=iteratia_N(varianta(i),...
Lapos_dec_a1,Lex_deint_apriori_dec1,y,bitii_transmisi,h_est);
Nr_erori_iteratia_2=Nr_erori_iteratia_2+numar_erori2 ;
%_____________________________________________________________
%trei iteratii
[Lapos_dec_a3,Lapos_dec_b3,Lex_deint_apriori_dec3,...
numar_erori3,rata3]=iteratia_N(varianta(i),...
Lapos_dec_a2,Lex_deint_apriori_dec2,y,bitii_transmisi,h_est);
Nr_erori_iteratia_3=Nr_erori_iteratia_3+numar_erori3;
%_____________________________________________________________
%patru iteratii
[Lapos_dec_a4,Lapos_dec_b4,Lex_deint_apriori_dec4,...
numar_erori4,rata4]=iteratia_N(varianta(i),...
Lapos_dec_a3,Lex_deint_apriori_dec3,y,bitii_transmisi,h_est);
Nr_erori_iteratia_4=Nr_erori_iteratia_4+numar_erori4;
%_____________________________________________________________
%cinci iteratii
[Lapos_dec_a5,Lapos_dec_b5,Lex_deint_apriori_dec5,...
numar_erori5,rata5]=iteratia_N(varianta(i),...
Lapos_dec_a4,Lex_deint_apriori_dec4,y,bitii_transmisi,h_est);
Nr_erori_iteratia_5=Nr_erori_iteratia_5+numar_erori5;

%cu estimare LS
%_____________________________________________________________
% iteratia 0
h_est0LS=estimare_LS(Lapos_dec_b0,y1)
%_____________________________________________________________
%o iteratie
[Lapos_dec_a1LS,Lapos_dec_b1LS,Lex_deint_apriori_dec1LS,...
numar_erori1LS,rata1LS]=iteratia_N(varianta(i),...
Lapos_dec_a0,Lex_deint_apriori_dec0,y,bitii_transmisi,h_est0LS);
Nr_erori_iteratia_1LS=Nr_erori_iteratia_1LS+numar_erori1LS;
h_est1LS=estimare_LS(Lapos_dec_b1LS,y1)
%_____________________________________________________________
%doua iteratii
[Lapos_dec_a2LS,Lapos_dec_b2LS,Lex_deint_apriori_dec2LS,...
numar_erori2LS,rata2LS]=iteratia_N(varianta(i),...
Lapos_dec_a1LS,Lex_deint_apriori_dec1LS,...
y,bitii_transmisi,h_est1LS);
Nr_erori_iteratia_2LS=Nr_erori_iteratia_2LS+numar_erori2LS ;
h_est2LS=estimare_LS(Lapos_dec_b2LS,y1)
%_____________________________________________________________
%trei iteratii
[Lapos_dec_a3LS,Lapos_dec_b3LS,Lex_deint_apriori_dec3LS,...
numar_erori3LS,rata3LS]=iteratia_N(varianta(i),...
Lapos_dec_a2LS,Lex_deint_apriori_dec2LS,...
y,bitii_transmisi,h_est2LS);
Nr_erori_iteratia_3LS=Nr_erori_iteratia_3LS+numar_erori3LS;
h_est3LS=estimare_LS(Lapos_dec_b3LS,y1)
%_____________________________________________________________
%patru iteratii
```

```
        [Lapos_dec_a4LS,Lapos_dec_b4LS,Lex_deint_apriori_dec4LS,...
        numar_erori4LS,rata4LS]=iteratia_N(varianta(i),...
        Lapos_dec_a3LS,Lex_deint_apriori_dec3LS,...
        y,bitii_transmisi,h_est3LS);
        Nr_erori_iteratia_4LS=Nr_erori_iteratia_4LS+numar_erori4LS;
        h_est4LS=estimare_LS(Lapos_dec_b4LS,y1)
        %_____________________________________________________________
        %cinci iteratii
        [Lapos_dec_a5LS,Lapos_dec_b5LS,Lex_deint_apriori_dec5LS,...
        numar_erori5LS,rata5LS]=iteratia_N(varianta(i),...
        Lapos_dec_a4LS,Lex_deint_apriori_dec4LS,...
        y,bitii_transmisi,h_est4LS);
        Nr_erori_iteratia_5LS=Nr_erori_iteratia_5LS+numar_erori5LS;
        h_est5LS=estimare_LS(Lapos_dec_b5LS,y1)
        numar_secvente_generate=numar_secvente_generate+1;
        end
      numar_biti_generati=numar_secvente_generate...
                    *lungimea_secventei_de_date;

        ber0(i)=Nr_erori_iteratia_0/numar_biti_generati;
        ber1(i)=Nr_erori_iteratia_1/numar_biti_generati;
        ber2(i)=Nr_erori_iteratia_2/numar_biti_generati;
        ber3(i)=Nr_erori_iteratia_3/numar_biti_generati;
        ber4(i)=Nr_erori_iteratia_4/numar_biti_generati;
        ber5(i)=Nr_erori_iteratia_5/numar_biti_generati;
        ber1LS(i)=Nr_erori_iteratia_1LS/numar_biti_generati;
        ber2LS(i)=Nr_erori_iteratia_2LS/numar_biti_generati;
        ber3LS(i)=Nr_erori_iteratia_3LS/numar_biti_generati;
        ber4LS(i)=Nr_erori_iteratia_4LS/numar_biti_generati;
        ber5LS(i)=Nr_erori_iteratia_5LS/numar_biti_generati;
end

      semilogy(EbN0_db,ber0,'b<-',EbN0_db,ber1,'k^-',...
         EbN0_db,ber2,'gv-',EbN0_db,ber3,'ok-',EbN0_db,...
         ber4,'ms-',EbN0_db,ber5,'dr-',EbN0_db,ber0,...
         'b<-.',EbN0_db,ber1LS,'k^-.',EbN0_db,ber2LS,...
         'gv-.',EbN0_db,ber3LS,'ok-.',EbN0_db,ber4LS,...
         'ms-.',EbN0_db,ber5LS,'dr-.','linewidth',2);

      grid on;
      legend('iteratia0','iteratia1','iteratia2',...
         'iteratia3','iteratia4','iteratia5',...
         'iteratia0LS','iteratia1LS','iteratia2LS',...
         'iteratia3LS','iteratia4LS','iteratia5LS');
      xlabel('E_b/N_0','FontSize',14);
      ylabel('BER','FontSize',14);
```

Functii utilizate în cadrul programului ber.m

1. calc_erorii.m

```
%calculul erorii de bit
function   [numar_erori,rata]=calc_erorii(Secventa_rec,bitii_transmisi)

for i=1:length(Secventa_rec)
  if Secventa_rec(i)>=0
    bit_dec(i)=1;
  elseif Secventa_rec(i)<0
    bit_dec(i)=0;
  end
end
 [numar_erori,rata]=biterr(bit_dec,bitii_transmisi);
```

2. canal.m

```
%face modularea BPSK a semnalului codat care
%apoi este trecut printr-un canal avand raspunsul h
%la semnal se mai adauga si zgomotul rezultand semnalul y
function [y,y1]=canal(simb_cod_int,h,varianta,amplitudinea)
%simb_cod_intr-sunt simbolurile de la iesirea codorului
%h-raspunsul canalului
Nr=length(h);
N=length(simb_cod_int);
Secventa=simb_cod_int;
Secventa_cu_zero=[zeros(1,Nr-1) Secventa];
Secveta_mod_BPSK=2*Secventa-1; %moduleaza BPSK  Secventa
Secventa_cu_zero_mod_BPSK=2*Secventa_cu_zero-1;
Prod_conv=amplitudinea*conv(h,Secventa_cu_zero_mod_BPSK);
y=zeros(1,length(Secventa_cu_zero_mod_BPSK)-(Nr-1));
%generarea zgomotului
randn('state',0);
n=randn(length(Prod_conv),1)*sqrt(varianta);
y1=Prod_conv+n';
for i=Nr:length(Secventa_cu_zero_mod_BPSK)
  v(i-(Nr-1))=y1(i);
end
y=v';
```

3. canal_fara_zg.m

```
%aceasta functie este utilizata pentru a determina
%matricea iesire la egalizor

function [y]=canal_fara_zg(secventa_test,h_est,amplitudinea)
```

```
%secventa_test-sunt simbolurile generate
%pentru determinarea tranzitiilor
Nr=length(h_est);
N=length(secventa_test);
Secventa=secventa_test;
Secventa_cu_zero=[zeros(1,Nr-1) Secventa];
Secveta_mod_BPSK=2*Secventa-1;
Secventa_cu_zero_mod_BPSK=2*Secventa_cu_zero-1;
Prod_conv=amplitudinea*conv(h_est,Secventa_cu_zero_mod_BPSK);
y=zeros(1,length(Secventa_cu_zero_mod_BPSK)-(Nr-1));
 for i=Nr:length(Secventa_cu_zero_mod_BPSK)
  v(i-(Nr-1))=Prod_conv(i);
end
y=v'
```

4. codare.m

```
%codarea de canal
function iesire=codare(K,g1,g2,Secventa_intrare)
%Codare cod convolutional
clear iesire;
clear intrare;
n=2;
[trel]=trellis(K,g1,g2);
size=K-1;
intrare=[Secventa_intrare zeros(1,size)]  ;
iesire=zeros(1,n*length(intrare));

if intrare(1)==0
   a=fliplr(de2bi(trel.iesire(1,1)));
   if isempty(a)==1
      a=[0 0];
   end
   if length(a)<n
      a=[zeros(1,2-length(a)) a];
   end
   stare=trel.state(1,1);
   elseif intrare(1)==1
   a=fliplr(de2bi(trel.iesire(1,2)));
    f isempty(a)==1
      a=[0 0];
   end
   if length(a)<n
      a=[zeros(1,2-length(a)) a];
   end
   stare=trel.state(1,2);
end
iesire=[a];
for i=2:length(intrare)
   if intrare(i)==0
```

```
      a=fliplr(de2bi(trel.iesire(stare+1,1)));
      if isempty(a)==1
        a=[0 0];
      end
      if length(a)<n
        a=[zeros(1,2-length(a)) a];
      end
      stare=trel.state(stare+1,1);

    elseif intrare(i)==1
      a=fliplr(de2bi(trel.iesire(stare+1,2)));
      if isempty(a)==1
        a=[0 0];
      end
      if length(a)<n
        a=[zeros(1,2-length(a)) a];
      end

      stare=trel.state(stare+1,2);
    end
    iesire=[iesire a];
  end
```

5. conversie.m

```
function [C0,C1]=conversie(Secventa)
 C0=zeros(1,length(Secventa));
for i=1:length(Secventa)
   C0(i)=exp(-Secventa(i))/(1+exp(-Secventa(i)));
end
 C1=zeros(1,length(Secventa));
for i=1:length(Secventa)
   C1(i)=1/(1+exp(-Secventa(i)));
end
```

6. decod_canal.m

```
% decodor de canal
%Lapos_dec_b-este informatia a-posteriori generata de decodor
%din care rezulta bitii sursa(sistenatici) estimati
%Lapos_dec_a-este informatia a-posteriori provenita
% de la decodor care se va utiliza la determinarea informatiei
% a-priori pentru egalizor
  function [Lapos_dec_b,Lapos_dec_a] =...
    decod_canal(C1,C0,lungimea_secventei_receptionate)
lungimea_secventei_transmise=...
    length(lungimea_secventei_receptionate)/2;
gamma=zeros(4,4,lungimea_secventei_transmise);
 l=1;
for i=1:lungimea_secventei_transmise
   gamma(:,:,i)=.5*[C0(l)*C0(l+1),C1(l)*C1(l+1),0,0;
```

```
        0,0,C1(l)*C0(l+1),C0(l)*C1(l+1);
        C1(l)*C1(l+1),C0(l)*C0(l+1),0,0;
        0,0,C0(l)*C1(l+1),C1(l)*C0(l+1)];
    l=l+2;
end
nr_stari=4;            %Numarul de stari
A1=[0 1 0 0;0 0 1 0;1 0 0 0;0 0 0 1];
A0=[1 0 0 0;0 0 0 1;0 1 0 0;0 0 1 0];
B1_d1=zeros(4,4,lungimea_secventei_transmise);
B0_d0=zeros(4,4,lungimea_secventei_transmise);
numarator_d1=zeros(1,lungimea_secventei_transmise);
numitor_d1=zeros(1,lungimea_secventei_transmise);

%Initializare vector F utilizand  recursivitatea înainte
F=zeros(nr_stari,1,lungimea_secventei_transmise+1);
F(:,1,1)=[1;0;0;0] ;
C(1)=1;
%F calculat recursiv
for i=1:lungimea_secventei_transmise
    F(:,1,i+1)=gamma(:,:,i)'*F(:,1,i);
    C(i+1)=sum(F(:,1,i+1));
    F(:,1,i+1)=F(:,1,i+1)./C(i+1);
end
 %Initializare vector G utilizand  recursivitatea înapoi
G=zeros(nr_stari,1,lungimea_secventei_transmise+1);
G(:,1,lungimea_secventei_transmise+1)=[1;1;1;1];
%calcul vector G
for i=lungimea_secventei_transmise:-1:1
    G(:,1,i)=gamma(:,:,i)*G(:,1,i+1);
    h(i)=sum(G(:,1,i));
    G(:,1,i)=G(:,1,i)./h(i);
end
for n=1:lungimea_secventei_transmise
    B1_d1(:,:,n)=A1.*gamma(:,:,n);
    B0_d0(:,:,n)=A0.*gamma(:,:,n);
end
LLR_d1=zeros(1,lungimea_secventei_transmise);
for l=1:lungimea_secventei_transmise
    numarator_d1(l)=F(:,:,l)'*B1_d1(:,:,l)*G(:,:,l+1);
    numitor_d1(l)=F(:,:,l)'*B0_d0(:,:,l)*G(:,:,l+1);
    LLR_d1(l)=log((numarator_d1(l))/(numitor_d1(l)));
end

for i=1:lungimea_secventei_transmise
    if LLR_d1(i)>=0
        d1(i)=1;
    else
        d1(i)=-1;
    end
end
%nr_stari=4;            %Numarul de stari
```

```
A1=[0 1 0 0;0 0 0 1;1 0 0 0;0 0 1 0];
A0=[1 0 0 0;0 0 1 0;0 1 0 0;0 0 0 1];

B1_d1=zeros(4,4,lungimea_secventei_transmise);
B0_d0=zeros(4,4,lungimea_secventei_transmise);
numarator_d2=zeros(1,lungimea_secventei_transmise);
numitor_d2=zeros(1,lungimea_secventei_transmise);
for n=1:lungimea_secventei_transmise
   B1_d1(:,:,n)=A1.*gamma(:,:,n);
   B0_d0(:,:,n)=A0.*gamma(:,:,n);
end
LLR_d2=zeros(1,lungimea_secventei_transmise);
for l=1:lungimea_secventei_transmise
   numarator_d2(l)=F(:,:,l)'*B1_d1(:,:,l)*G(:,:,l+1);
   numitor_d2(l)=F(:,:,l)'*B0_d0(:,:,l)*G(:,:,l+1);
   LLR_d2(l)=log((numarator_d2(l))/(numitor_d2(l)));
end
for i=1:lungimea_secventei_transmise
   if LLR_d2(i)>=0
      d2(i)=1;
   else
      d2(i)=-1;
   end
end
concatenare=0;
for n=1:lungimea_secventei_transmise
   concatenare=[concatenare LLR_d1(n) LLR_d2(n)];
end
Lapos_dec_a=concatenare...
(2:length(lungimea_secventei_receptionate)+1);
%LLR-ul generat la iesirea decodorului
%nr_stari=4;
A1=[0 1 0 0;0 0 0 1;0 1 0 0;0 0 0 1];
A0=[1 0 0 0;0 0 1 0;1 0 0 0;0 0 1 0];

B1_d1=zeros(4,4,lungimea_secventei_transmise);
B0_d0=zeros(4,4,lungimea_secventei_transmise);
numarator_db=zeros(1,lungimea_secventei_transmise);
numitor_db=zeros(1,lungimea_secventei_transmise);
for n=1:lungimea_secventei_transmise
   B1_d1(:,:,n)=A1.*gamma(:,:,n);
   B0_d0(:,:,n)=A0.*gamma(:,:,n);
end
Lapos_dec_b=zeros(1,lungimea_secventei_transmise);
 for l=1:lungimea_secventei_transmise
   numarator_db(l)=F(:,:,l)'*B1_d1(:,:,l)*G(:,:,l+1);
   numitor_db(l)=F(:,:,l)'*B0_d0(:,:,l)*G(:,:,l+1);
   Lapos_dec_b(l)=log((numarator_db(l))/(numitor_db(l)));
end
```

7. deintretesere.m

```
function deintretesere=...
    deintretesere(Secventa,lungimea_secventei_de_date)
a=0;
matricea_de_deintretesere=reshape(Secventa,sqrt...
(2*lungimea_secventei_de_date),sqrt...
(2*lungimea_secventei_de_date));
[nr_linii,nr_coloane]=size(matricea_de_deintretesere);
for i=1:nr_linii
   a=[a matricea_de_deintretesere(i,:)];
end
 deintretesere=a(2:(1+2*lungimea_secventei_de_date));
```

8. det_prag.m

```
%detectie prag
function [biti_obtinuti]=det_prag(Secventa_rec)

for i=1:length(Secventa_rec)
   if Secventa_rec(i)>=0
      bit_dec(i)=1;
   elseif Secventa_rec(i)<0
      bit_dec(i)=0;
   end
end
biti_obtinuti= bit_dec;
```

9. egalizor_it0.m

```
%egalizor MAP iteratia 0

function  [Lex,date_estimate]=egalizor_it0(y,varianta,h_est)
%iesire=[1->1,1->2,   0      0;
%       0      0    2->3  2->4;    % tranzitiile posibile
%       3->1,3->2,   0      0;
%       0      0    4->3  4->4];

%starile sunt 00->1;   01->2 ;  10->3   11->4

 Secventa=[0 1 0 0 1 1 0 1 1 1 0 0 0];%este secventa pe baza careia se
construieste matricea iesire

 %succesiunea starilor este urmatoarea(iau grupuri de cate doua):
 %   1->2->3->1->2->4->3->2->4->4->3->1->1
 %se observa ca avem toate combinatiile necesare pentru matricea iesire
 %aceasta 'secventa' se trece prin functie canal
 %este important ca zgomotul prin canal sa fie 0 cand construiesc aceasta
matrice
 amplitudinea=1;
```

```
[y_iesire]=canal_fara_zg(Secventa,h_est,amplitudinea);

 iesire=zeros(4,4);
 iesire(1,1)=y_iesire(1);
 iesire(1,2)=y_iesire(2);
 iesire(2,3)=y_iesire(3);
 iesire(2,4)=y_iesire(6);
 iesire(3,1)=y_iesire(4);
 iesire(3,2)=y_iesire(8);
 iesire(4,3)=y_iesire(7);
 iesire(4,4)=y_iesire(10);
nr_stari=4;  %Numarul starilor
G=zeros(4,4,length(y));

A1=[0 1 0 0;0 0 0 1;0 1 0 0;0 0 0 1];
A0=[1 0 0 0;0 0 1 0;1 0 0 0;0 0 1 0];

pozitie=[1 1 0 0;0 0 1 1;1 1 0 0;0 0 1 1];
gamma=zeros(4,4,length(y));
B1=zeros(4,4,length(y));
B0=zeros(4,4,length(y));
numarator=zeros(1,length(y));
numitor=zeros(1,length(y));

%calculul gamma
for k=1:length(y)
  for i=1:4
    for j=1:4
      argument=-(y(k)-iesire(i,j))^2;

       G(i,j,k)=exp(argument/(2*varianta))*pozitie(i,j);

        end
    end
  end
gamma=G;

N=length(y)+1;

%recursivitate inainte ; initializare vector f
f=zeros(nr_stari,1,N);
f(:,1,1)=[1;0;0;0] ;
c=zeros(1,N);
c(1)=1;
%Calculul recursiv înainte a vectorului f
for i=1:(N-1)
  f(:,1,i+1)=gamma(:,:,i)'*f(:,1,i);
  c(i+1)=sum(f(:,1,i+1));
  f(:,1,i+1)=f(:,1,i+1)./c(i+1);
end
%recursivitate inapoi; initializare vector b
```

```
b=zeros(nr_stari,1,N);
b(:,1,N)=[1;1;1;1];
%Calculul vectorului b
for i=(N-1):-1:1
   b(:,1,i)=gamma(:,:,i)*b(:,1,i+1);
   h(i)=sum(b(:,1,i));
   b(:,1,i)=b(:,1,i)./h(i);
end
for n=1:length(y)
B1(:,:,n)=A1.*gamma(:,:,n);
B0(:,:,n)=A0.*gamma(:,:,n);
end
  date_estimate=zeros(1,N-1);
 for k=1:(N-1)
       numarator(k)=f(:,:,k)'*B1(:,:,k)*b(:,:,k+1);
        numitor(k)=f(:,:,k)'*B0(:,:,k)*b(:,:,k+1);
      Lex(k)=log((numarator(k))/(numitor(k)));
end
for i=1:length(y)
   if Lex(i)>=0
      date_estimate(i)=1;
   elseif Lex(i)<0
      date_estimate(i)=-1;
   end
end
```

10. egalizor_itN.m

```
% egalizor MAP iteratia N
%Lex-informatia extrinseca la iesirea din egalizor
function  [Lex,secventa_estimata]=...
    egalizor_itN(y,varianta,C1egalizor,C0egalizor,h_est)

 Secventa=[0 1 0 0 1 1 0 1 1 1 0 0 0];%este secventa pe baza
 %careia se construieste matricea iesire

 amplitudinea=1;
[y_iesire]=canal_fara_zg(Secventa,h_est,amplitudinea);
 iesire=zeros(4,4);
 iesire(1,1)=y_iesire(1);
 iesire(1,2)=y_iesire(2);
 iesire(2,3)=y_iesire(3);
 iesire(2,4)=y_iesire(6);
 iesire(3,1)=y_iesire(4);
 iesire(3,2)=y_iesire(8);
 iesire(4,3)=y_iesire(7);
 iesire(4,4)=y_iesire(10);
numarul_starilor=4;
P=zeros(4,4,length(y));
```

```
A1=[0 1 0 0;0 0 0 1;0 1 0 0;0 0 0 1];
A0=[1 0 0 0;0 0 1 0;1 0 0 0;0 0 1 0];

gamma=zeros(numarul_starilor,numarul_starilor,length(y));
B1=zeros(numarul_starilor,numarul_starilor,length(y));
B0=zeros(numarul_starilor,numarul_starilor,length(y));
numarator=zeros(1,length(y));
numitor=zeros(1,length(y));
inf_apriori_egal=...
    zeros(numarul_starilor,numarul_starilor,length(y));

for l=1:length(y)
    inf_apriori_egal(:,:,l)=[C0egalizor(l),C1egalizor(l),...
        0,0;0,0,C0egalizor(l),C1egalizor(l);C0egalizor(l)...
        ,C1egalizor(l),0,0;0,0,C0egalizor(l),C1egalizor(l)];
end
%calcul gamma
for l=1:length(y)
   for i=1:4
      for j=1:4
         arg=-(y(l)-iesire(i,j))^2;
         P(i,j,l)=inf_apriori_egal(i,j,l)...
             *exp(arg/(2*varianta))*(1/sqrt(2*pi*varianta));
          end
       end
   end
gamma=P;

N=length(y)+1;
 %recursivitate inainte; initializare vector f
f=zeros(numarul_starilor,1,N);
f(:,1,1)=[1;0;0;0] ;
c=zeros(1,N);
c(1)=1;
%Calculul vector f
for i=1:(N-1)
   f(:,1,i+1)=gamma(:,:,i)'*f(:,1,i);
   c(i+1)=sum(f(:,1,i+1));
   f(:,1,i+1)=f(:,1,i+1)./c(i+1);
end
%recursivitate inapoi; initializare vector b
b=zeros(numarul_starilor,1,N);
b(:,1,N)=[1;1;1;1];
%Calculul recursiv înapoi a vectorului b
for i=(N-1):-1:1
   b(:,1,i)=gamma(:,:,i)*b(:,1,i+1);
   h(i)=sum(b(:,1,i));
   b(:,1,i)=b(:,1,i)./h(i);
 end
b;
for n=1:length(y)
```

```
B1(:,:,n)=A1.*gamma(:,:,n);
B0(:,:,n)=A0.*gamma(:,:,n);
end
  date=zeros(1,N-1);
 for l=1:(N-1)
        numarator(l)=f(:,:,l)'*B1(:,:,l)*b(:,:,l+1);
        numitor(l)=f(:,:,l)'*B0(:,:,l)*b(:,:,l+1);
      Lex(l)=log((numarator(l))/(numitor(l)));
end
  for i=1:length(y)
   if Lex(i)>=0
      secventa_estimata(i)=1;
   elseif Lex(i)<0
      secventa_estimata(i)=-1;
   end
end
```

11. estimare0_LS.m

```
% estimarea initiala
function [h_est]=estimare0_LS(secventa_test,y)
 Nr=3;
secv_test_cu_zero=[zeros(1,Nr-1) secventa_test];
% se face modularea
secv_test_cu_zero_mod_BPSK=2*secv_test_cu_zero-1;
% in estimarea Ls aceasta este notata cu x
x=secv_test_cu_zero_mod_BPSK;
%----------------------------------------
%construirea matricei X

Xc=zeros(1,length(x)+ Nr-1);
for i=1:length(x)
    Xc(i)=x(i);
end
Xr=zeros(1,Nr);
Xr(1)=x(1);
Xr=Xr';
X = toeplitz(Xc,Xr);
%----------------------------------------
  h_est = inv(X.'*X) * X.' * y';
 h_est=h_est';
```

12. estimare_LS.m

```
%se utilizeaza pentru estimare iterativa
function [h_est]=estimare_LS(Secventa_rec,y)
%y -reprezinta secventa obtinuta la iesirea din canal
%secventa receptionata trebuie trecuta
%printr-un slicer (functia det prag)pt a obtine  bitii 1 si 0.
```

```
[Secventa_rec]=det_prag(Secventa_rec);
%aceasta secventa trebuie codata
lungimea_secventei=length(Secventa_rec);
K=3;
g1=[1 1 1];g2=[1 0 1];
simboluri_rec_codate=codare(K,g1,g2,Secventa_rec);
simboluri_rec_codate=...
   simboluri_rec_codate(1:2*lungimea_secventei);
%se face o intretesere

simboluri_codate_intretesute=...
   intretesere(simboluri_rec_codate,lungimea_secventei);
Nr=3;
Secventa_cu_zero_rec=[zeros(1,Nr-1) simboluri_codate_intretesute];
% se face modularea
Secventa_cu_zero_rec_mod_BPSK=2*Secventa_cu_zero_rec-1;
% in estimarea Ls aceasta este notata cu x
x=Secventa_cu_zero_rec_mod_BPSK;
%----------------------------------------
%construirea matricei X

Xc=zeros(1,length(x)+ Nr-1);
for i=1:length(x)
   Xc(i)=x(i);
end
Xr=zeros(1,Nr);
Xr(1)=x(1);
Xr=Xr';
X = toeplitz(Xc,Xr);
%----------------------------------------
 h_est = inv(X.'*X) * X.' * y';
 h_est=h_est';
```

13. intretesere.m

```
function intretesere=intretesere...
   (Secventa,lungimea_secventei_de_date)
a=0;
matricea_de_intretesere=...
reshape(Secventa,sqrt(2*lungimea_secventei_de_date),...
sqrt(2*lungimea_secventei_de_date));

[nr_linii,nr_coloane]=size(matricea_de_intretesere);
for i=1:nr_linii
  a=[a matricea_de_intretesere(i,:)];
end
intretesere=a(2:(1+2*lungimea_secventei_de_date));
```

14. **trellis.m**

```
%matricea trellis
function trel = trellis(K,g1,g2)

memorie_trelis=K-1;
num_stari_trelis=2^(K-1);
trel.state=zeros(num_stari_trelis,2);
trel.iesire=zeros(num_stari_trelis,2);
s=zeros(1,memorie_trelis);
 %implementarea matricii de stare
 %prima linie a matricii de stare
for j=memorie_trelis:2
      s(j)=s(j-1);
   end
   s(1)=0;
   trel.state(1,1)=bi2de(s);
   s(1)=1;
   trel.state(1,2)=bi2de(s);
%urmatoarele linii ale matricii
for i=1:num_stari_trelis-1
   s=de2bi(i);
   for j=memorie_trelis:2
      s(j)=s(j-1);
   end
   s(1)=0;
   trel.state(i+1,1)=bi2de(s);
   s(1)=1;
   trel.state(i+1,2)=bi2de(s);
end

%implemantarea  primei coloane a matricii de iesire
W=zeros(1,K);
for w=1:num_stari_trelis ;
   iesireg1=0;
   iesireg2=0;
  H=de2bi(w-1);
   a=length(H);
   if a<K
      x=memorie_trelis-a;
      G=zeros(1,x);
      H=[H G];
   end
a=isempty(H);
if a==1
   H=zeros(1,memorie_trelis);
   end
   W=[0 H];
   iesire1=bitand(W,g1);
   iesire2=bitand(W,g2);
for j=1:2
```

```
            iesireg1=bitxor(iesireg1,iesire1(j+1));
            iesireg2=bitxor(iesireg2,iesire2(j+1));
          end
          z=[iesireg2 iesireg1];
          trel.iesire(w,1)=bi2de(z);
 end
 trel.iesire;
 %implemantarea  coloanei a doua a matricii de iesire
W=zeros(1,K);
for w=1:num_stari_trelis
   iesireg1=1;
   iesireg2=1;
  H=de2bi(w-1);
   a=length(H);
   if a<K
      x=memorie_trelis-a;
      G=zeros(1,x);
      H=[H G];
   end
a=isempty(H);
if a==1
   H=zeros(1,memorie_trelis);
   end
   W=[1 H];
   iesire1=bitand(W,g1);
   iesire2=bitand(W,g2);
for j=1:2
   iesireg1=bitxor(iesireg1,iesire1(j+1));
   iesireg2=bitxor(iesireg2,iesire2(j+1));
end
z=[iesireg2 iesireg1];
trel.iesire(w,2)=bi2de(z);

end
 trel.iesire;
```

15. iteratia_0.m

```
function [bitii_transmisi,simboluri_codate,...
    Lapos_dec_a0,Lapos_dec_b0, Lex_deint,y,...
    numar_erori,rata,y1,h_est]=...
    iteratia_0(varianta,lungimea_secventei,h)

 K=3;
g1=[1 1 1];g2=[1 0 1];
bitii_transmisi = randint(1,lungimea_secventei);
%codificarea de canal; codare convolutionala
simboluri_codate=codare(K,g1,g2,bitii_transmisi);
simboluri_codate=simboluri_codate(1:2*lungimea_secventei);
```

```
%intretesere
simboluri_codate_intretesute=...
    intretesere(simboluri_codate,lungimea_secventei);

secventa_test=randint(1,18);%sau alte valori
%Dupa care aceste date trec prin canal otinandu-se la iesire y

amplitudinea=1; %amplitudinea semnalului
[y,y1]=canal(simboluri_codate_intretesute,...
    h,varianta,amplitudinea);
[y_test,y1_test]=canal(secventa_test,h,...
    varianta,amplitudinea); %secventa test prin canal

[h_est]=estimare0_LS(secventa_test,y1_test);

%Algoritmul MAP prin egalizor

[Lex,date_estimate]=egalizor_it0(y,varianta,h_est);
% informatia extrinseca provenita de la egalizor

%deintretesere
Lex_deint=deintretesere(Lex,lungimea_secventei);

%Genereaza informatia a-priori în decodor
[C0,C1]=conversie(Lex_deint);

%decodarea de canal------ Algoritmul MAP
[Lapos_dec_b0,Lapos_dec_a0]=decod_canal(C1,C0,Lex_deint);
%Lapos_dec_b0-este informatia a-posteriori generata
%de decodor din care rezulta bitii sursa(sistenatici) estimati
%Lapos_dec_a0-este informatia extinseca provenita de la decodor
% care se va utiliza la determinarea informatiei
% a-priori pentru egalizor

[numar_erori,rata]=calc_erorii(Lapos_dec_b0,bitii_transmisi);
```

16. iteratia_N.m

```
function [Lapos_dec_a,Lapos_dec_b,Lex_deint_apriori_dec,...
    numar_erori,rata]=iteratia_N(varianta,...
    Lapos_dec_a_anterior,Lex_deint,y,bitii_transmisi,h_est)

   %Informatiile extrinseci la iesirea din decodor
   Lextrinsec_decodor=Lapos_dec_a_anterior-Lex_deint;
   %informatiile extrinseci combinate cu ale canalului
    %(Lex_deint) sunt scoase din informatiile a-posteriori
    %provenite din decodor
```

```
%Dupa intretesere
lungimea_secventei = length(bitii_transmisi);
Lapriori_egalizor=intretesere...
   (Lextrinsec_decodor,lungimea_secventei);
%informatia apriori care merge la egalizor

%Informatii extrinseci la intrarea in egalizor
[C0egalizor,C1egalizor]=conversie(Lapriori_egalizor);

%Prin egalizor ------ algoritmul MAP
[Lextrinseci_egalizor0,secventa_estimata]=...
   egalizor_itN(y,varianta,C1egalizor,...
   C0egalizor,h_est);
%Se obtine la iesirea egalizorului informatii extrinseci

Lextrinseci_egalizor=...
   Lextrinseci_egalizor0-Lapriori_egalizor; %Înainte de
%a transmite decodorului SISO informatia a-posteriori
%generata de egalizorul SISO, trebuie eliminata contributia
%decodorului (ca informatie a-priori),de
% la iteratia anterioara,pentru a aduce la intrarea
% decodorului doar informatiile extrinseci combinate
% cu ale canalului.

%deintretesere
Lex_deint_apriori_dec=...
   deintretesere(Lextrinseci_egalizor,lungimea_secventei);

%Obtinerea informatiilor la intrarea in decodor
[C0,C1]=conversie( Lex_deint_apriori_dec);


%decodor canal  algoritm MAP
[Lapos_dec_b,Lapos_dec_a]=...
   decod_canal(C1,C0,Lex_deint_apriori_dec);

[numar_erori,rata]=calc_erorii(Lapos_dec_b,bitii_transmisi);
```