UNIVERSITATEA "POLITEHNICA" DIN TIMIŞOARA
FACULTATEA DE ELECTRONICĂ ŞI TELECOMUNICAŢII
DEPARTAMENTUL DE MASURĂRI ŞI ELECTRONICĂ OPTICĂ

METODE ŞI ALGORITMI DE CUANTIZARE VECTORIALĂ

Teză de doctorat

Conducător ştiinţific:
prof. dr. ing. Sever Crişan

Autor:
ing. Septimiu Mischie

1998

PREFAȚĂ

Prelucrarea numerică a semnalelor constituie o preocupare tradițională în cadrul colectivului Departamentului de Măsurări și Electronică Optică al Facultății de Electronică și Telecomunicații din Timișoara, în acest domeniu susținându-se mai multe teze de doctorat.

Prelucrarea numerică se aplică în prezent pentru foarte multe categorii de semnale dintre care se amintesc: semnalul vocal, semnalul de imagine și semnalele biomedicale.

Prelucrarea semnalului vocal este o preocupare mai veche a autorului. Astfel, tema proiectului de diplomă cu care autorul a obținut titlul de inginer s-a numit "Sistem pentru recunoașterea cuvintelor izolate".

Cuantizarea vectorială este o tehnică utilizată astăzi în majoritatea domeniilor de prelucrare numerică, datorită structurii sale și modului relativ simplu de implementare, având ca scop compresia datelor. Unul dintre domeniile în care cuantizarea vectorială este utilizată frecvent este cel al semnalului vocal.

După conversia analog-numerică realizată cu ajutorul circuitelor specializate, fiecărui eșantion i se atribuie un cod numeric, cu un anumit număr de biți. În scopul micșorării debitului de informație transmis, prin intermediul metodelor de prelucrare numerică trebuie efectuată o compresie, adică trebuie redus numărul de biți alocat fiecărui eșantion, păstrând însă eroarea obținută după reconstituirea semnalului original la valori acceptabile. Mai întâi s-a folosit în acest scop cuantizarea scalară, iar în prezent se folosește aproape exclusiv cuantizarea vectorială. Astfel, prin acest procedeu, se atribuie un cod unui grup de eșantioane sau unui vector. În acest fel se reduce numărul echivalent de biți alocat fiecărui eșantion, realizându-se compresia.

În practică se cuantizează vectorial diferite forme ale parametrilor modelului liniar predictiv al semnalului vocal, între care liniile spectrale de

frecvenţă (parametrii LSF) dau cele mai bune rezultate. Astfel, autorul abordează cuantizarea vectorială a parametrilor LSF, obţinând rezultate deosebite în acest sens.

Teza este structurată în cinci capitole şi două anexe conţinând o listă cu 50 de titluri bibliografice.

Autorul aduce cele mai respectuase mulţumiri domnului profesor dr. ing. Sever Crişan pentru îndrumarea competentă dată în decursul stagiului pentru doctorat, începând cu referatele şi continuând cu teza propriuzisă.

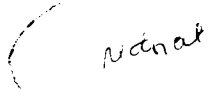
De asemenea, autorul aduce mulţumiri domnului profesor dr. ing. Eugen Pop, pentru importantele indicaţii şi sugestii date în timpul elaborării tezei.

Autorul aduce mulţumiri domnului profesor dr. ing. Liviu Toma, atât pentru sprijinul dat în elaborarea tezei precum şi pentru îndrumarea competentă a întregii activităţi a autorului atât ca student cât şi în calitate de cadru didactic.

Autorul mulţumeşte de asemenea colegului as. ing. Alexandru Iacovliev, pentru ajutorul dat la elaborarea programelor.

În final, autorul mulţumeşte doamnei Magdalena Ciobanu şi domnişoarei Mihaela Şerban, care au avut o contribuţie esenţială la redactarea în această formă a tezei.

Timişoara, 24 septembrie 1998 Autorul



CUPRINS

1. NOȚIUNI INTRODUCTIVE DESPRE CUANTIZAREA VECTORIALĂ

Cuantizarea vectorială, VQ, (vector quantization, în limba engleză) numită de Shanon "cuantizare bloc" [1] a apărut ca urmare a ideii potrivit căreia, sistemele de cuantizare pot obține rezultate mai bune dacă operează asupra unor grupuri de simboluri sau *vectori*, în locul simbolurilor individuale.

Denumirea de cuantizare vectorială s-a impus la începutul anilor 1980, când de fapt s-a trecut de la studiile teoretice, la implementarea unor algoritmi care operează asupra unor semnale reale [1].

Cuantizarea vectorială este o generalizare a cuantizării scalare. Dacă cuantizarea scalară are în general la intrare mărimi analogice, în cazul cuantizării vectoriale se acționează asupra unor semnale sub formă numerică. Scopul principal al utilizării cuantizării vectoriale este compresia datelor.

Un cuantizor vectorial este un sistem care alocă unui vector k-dimensional, $\boldsymbol{x}=(x_1, x_2,...,x_k)$, $\boldsymbol{x}\in\Re^k$, un alt vector de aceeași dimensiune, $\hat{\boldsymbol{x}}=(\hat{x}_1,\hat{x}_2,...,\hat{x}_k), \hat{\boldsymbol{x}}\in\boldsymbol{\ell}$. Mulțimea $\boldsymbol{\ell}$ conține un număr finit de vectori, $\boldsymbol{\ell}=\{\boldsymbol{y}_1, \boldsymbol{y}_2,...,\boldsymbol{y}_N\}$ și se numește *carte de cod* [1] sau *alfabet*, iar vectorii $\boldsymbol{y}_i$, $i=1,...,N$, se numesc *vectori de reproducere* sau *cuvinte de cod*.

Funcția îndeplinită de un cuantizor vectorial este $Q:\Re^k\rightarrow\boldsymbol{\ell}$, $\hat{\boldsymbol{x}}=Q(\boldsymbol{x})$.

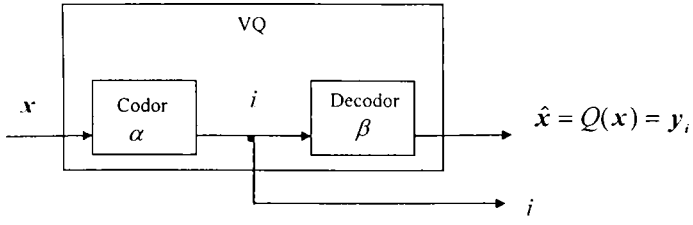


Figura 1.1: *Structura unui cuantizor vectorial.*

[1] Principalele denumiri legate de cuantizarea vectorială, scrise înclinat, au fost preluate de autor din literatură și traduse.

Un cuantizor vectorial poate fi descompus în două structuri distincte: codorul și decodorul (figura 1.1). Codorul, reprezentat prin funcția de codare α, alocă unui vector $\boldsymbol{x}$ un *cod* sau *simbol de canal*, $i \in I = \{1,2,....,N\}$. Decodorul, reprezentat prin funcția de decodare β, alocă codului i, vectorul de reproducere $\hat{\boldsymbol{x}} = \boldsymbol{y}_i$. Astfel,

$$\alpha : \Re^k \rightarrow I \; ; \; \beta : I \rightarrow \ell. \tag{1.1}$$

De multe ori se consideră și codul i ca ieșire a cuantizorului vectorial. Acest cod este de fapt indicele vectorului de reproducere $\boldsymbol{y}_i$ și se reprezintă sub formă binară. Numărul de biți necesar pentru reprezentarea codului este astfel $\log_2 N$. Pentru o utilizare a tuturor combinațiilor binare posibile, N se ia o putere a lui 2. Mărimea $r = \log_2 N$ se numește *rata* unui cuantizor vectorial și se exprimă în biți pe vector. Se observă că această mărime nu depinde de numărul de biți prin care se memorează fiecare componentă a vectorilor de reproducere, ci doar de numărul acestora, N. De asemenea, uneori se folosește și rata, în biți pe componentă, $r' = r/k$.

Oricărui cuantizor vectorial îi este caracteristică o *partiție* a spațiului $\Re^k$ în N regiuni de cuantizare, R_i, $i=1,...,N$. O regiune de cuantizare R_i conține mulțimea vectorilor $\boldsymbol{x}$ care se cuantizează prin vectorul de reproducere $\boldsymbol{y}_i$,

$$R_i = \{\boldsymbol{x} \in \Re^k : Q(\boldsymbol{x}) = \boldsymbol{y}_i\} \tag{1.2}$$

În principiu, codorul trebuie să determine regiunea de cuantizare în care "intră" vectorul de cuantizat, deci trebuie să cunoască partiția. Decodorul conține o tabelă care are la N adrese succesive cei N vectori de reproducere $\boldsymbol{y}_i$. Deci, în momentul primirii codului i, decodorul citește vectorul aflat la adresa i în tabela respectivă.

Regiunile de cuantizare R_i sunt distincte, iar prin reuniunea lor se obține spațiul $\Re^k$, $R_i \cap R_j = \varnothing$ pentru $i \neq j$; $\cup R_i = \Re^k$.

Procesului de cuantizare vectorială al unui vector $\boldsymbol{x}$ prin $\hat{\boldsymbol{x}} = Q(\boldsymbol{x})$ i se asociază o măsură a erorii de cuantizare, $d(\boldsymbol{x}, \hat{\boldsymbol{x}})$, care este o distanță între 2 vectori. Preformanțele unui cuantizor vectorial se apreciază prin valoarea

medie a erorii de cuantizare, $D=E[d(\boldsymbol{X}, \hat{\boldsymbol{X}})]$, unde $\boldsymbol{X}$ reprezintă un vector aleator uniform distribuit în $\Re^k$.

Cu cât eroarea medie este mai mică, cu atât calitatea cuantizorului vectorial este mai bună. În practică, performanțele cuantizorului se apreciază prin media în timp,

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i), \tag{1.3}$$

unde n reprezintă numărul de vectori care se cuantizează.

Cea mai folosită măsură a erorii de cuantizare este eroarea pătratică sau distanța euclidiană,

$$d(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2 = \sum_{i=1}^{k} (x_i - \hat{x}_i)^2. \tag{1.4}$$

O altă măsură a erorii de cuantizare este eroarea pătratică ponderată,

$$d(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \sum_{i=1}^{k} w_i (x_i - \hat{x}_i)^2, \tag{1.5}$$

unde w_i reprezintă ponderea alocată fiecărei componente i, în acest fel putându-se accentua anumite componente ale vectorilor, considerate mai importante.

Un cuantizor vectorial se numește *mărginit*, dacă este definit pe un domeniu mărginit, $B \subset \Re^k$.

Un cuantizor vectorial se numește *regular*, dacă regiunile de cuantizare sunt convexe și, pentru fiecare cod i, vectorul de reproducere $\boldsymbol{y}_i \in R_i$. În plus, dacă regiunile de cuantizare sunt mărginite de segmente și sunt obținute prin intersecția unui număr finit de semispații de forma

$$\{\boldsymbol{x} \in \Re^k : \langle \boldsymbol{u}_v \cdot \boldsymbol{x} \rangle + v_v \geq 0\}, \tag{1.6}$$

cuantizorul vectorial se numește *politop* [1]. $\boldsymbol{u}_v$ (care este un vector k - dimensional) și v_v (scalar) sunt parametrii semispațiului, iar $\langle \boldsymbol{u}_v \cdot \boldsymbol{x} \rangle$ reprezintă un produs scalar. În general, frontierele regiunilor de cuantizare sunt alcătuite din segmente care aparțin unui spațiu de dimensiune mai mică decât k, de obicei $k-1$. În figura 1.2 se prezintă regiunile de cuantizare ale

unui cuantizor vectorial mărginit, pentru k=2. Punctele din interiorul regiunilor reprezintă vectorii de reproducere. Acest cuantizor este regular pentru că regiunile sunt convexe şi politop pentru că sunt mărginite de segmente de dreaptă (din spaţiul k = 1).

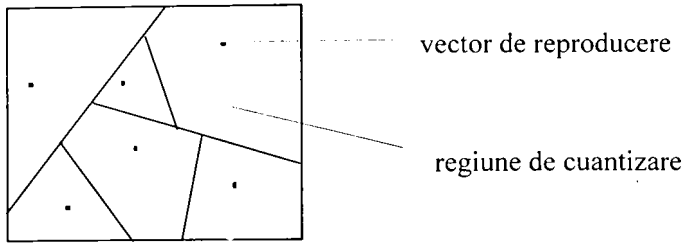


Figura 1.2: *Regiunile de cuantizare şi vectorii de reproducere pentru un cuantizor vectorial.*

La un cuantizor vectorial regular şi politop din spaţiul k = 3, regiunile de cuantizare sunt poliedre, şi sunt mărginite de poligoane (din spaţiul k =2).

Aşa cum s-a mai spus, codorul trebuie să găsească regiunea de cuantizare în care intră vectorul de cuantizat. Pentru aceasta, trebuie cunoscută structura geometrică a partiţiei. Pentru un cuantizor vectorial regular şi politop, regiunile de cuantizare se obţin prin intersecţia unui număr de semispaţii egal cu numărul feţelor fiecărei regiuni. Dacă se notează cu H_ν un semispaţiu,

$$H_\nu = \{\boldsymbol{x} \in \Re^k : \langle \boldsymbol{u}_\nu \cdot \boldsymbol{x} \rangle + v_\nu \geq 0\}, \tag{1.7}$$

atunci o regiune de cuantizare R_i se poate exprima ca

$$R_i = \bigcap_{\nu=1}^{L_i} H_\nu, \tag{1.8}$$

unde L_i reprezintă numărul feţelor regiunii de cuantizare R_i. Frontiera care mărgineşte semispaţiul H_ν se exprimă prin

$$F_v = \{\boldsymbol{x} \in \Re^k : \langle \boldsymbol{u}_v \cdot \boldsymbol{x} \rangle + v_v = 0\}, \tag{1.9}$$

iar feţele regiunilor de cuantizare sunt porţiuni din frontiere de tipul F_v.

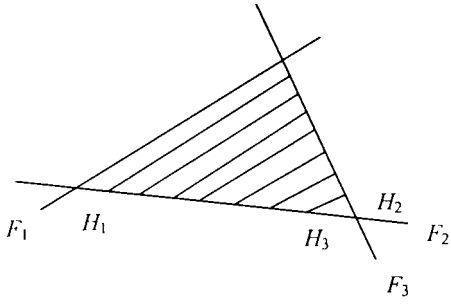


Figura 1.3: *Regiunea de cuantizare a unui cuantizor vectorial obţinută ca intersecţie de semiplane.*

În figura 1.3 se prezintă o regiune de cuantizare pentru un cuantizor vectorial cu $k = 2$, care are 3 feţe. Ea se obţine prin intersecţia a 3 semiplane, H_1, H_2 şi H_3, mărginite de frontierele F_1, F_2, F_3. Pentru a determina dacă un vector $\boldsymbol{x} \in \Re^2$ intră în regiunea de cuantizare din figura 1.3, codorul trebuie să verifice dacă $\langle \boldsymbol{u}_v \cdot \boldsymbol{x} \rangle + v_v \geq 0$ pentru $v = 1, 2, 3$, adică dacă $\boldsymbol{x}$ aparţine fiecăruia din semiplanele H_v, $v = 1, 2, 3$. Deci, codorul trebuie să cunoască ecuaţiile tuturor semiplanelor prin intersecţia cărora se obţine regiunea de cuantizare, iar acest procedeu trebuie repetat pentru toate regiunile de cuantizare. Rezultă un număr mare de operaţii, ţinând cont că numărul de semispaţii care mărginesc regiunile de cuantizare creşte odată cu creşterea dimensiunii k.

Acest mod de tratare a codorului unui cuantizor vectorial are mai mult o valoare teoretică, deoarece în cele mai multe cazuri practice, există un alt mod de realizare a funcţiei de codare, în care partiţia este complet determinată de vectorii de reproducere componenţi ai cărţii de cod şi de măsura erorii de cuantizare.

Un cuantizor vectorial la care orice vector se cuantizează prin cel mai apropiat vector de reproducere, în sensul unei distanţe date, se numeşte de tip Voronoi şi se spune că respectă condiţia celei mai apropiate vecinătăţi. În acest caz regiunile de cuantizare se pot exprima prin

$$R_i=\{\boldsymbol{x} : d(\boldsymbol{x}, \boldsymbol{y}_i) \le d(\boldsymbol{x}, \boldsymbol{y}_j), \text{ pentru orice } j\in I\} \tag{1.10}$$

adică, o regiune de cuantizare R_i conține toți vectorii $\boldsymbol{x}$ care, dacă sunt cuantizați prin vectorul de reproducere $\boldsymbol{y}_i$, produc o eroare de cuantizare mai mică decât dacă s-ar cuantiza prin oricare alt vector de reproducere. Dacă un vector este egal depărtat de doi vectori de reproducere, atunci, convențional, codul este dat de cel mai mic dintre indicii vectorilor egal depărtați.

Codorul acestui tip de cuantizor determină codul vectorului de cuantizat în felul următor: calculează eroarea de cuantizare dintre vectorul respectiv și fiecare din cei N vectori de reproducere, după care determină cea mai mică dintre cele N erori; indicele vectorului de reproducere în raport cu care s-a obținut eroarea minimă reprezintă codul. Un astfel de proces de codare, în care vectorul de cuantizat se compară cu toți vectorii de reproducere se numește codare prin căutare totală.

În cele ce urmează se va arăta că un cuantizor vectorial care respectă condiția celei mai apropiate vecinătăți în sensul distanței euclidiene este regular și politop.

Se definește mulțimea

$$H_{ij} = \left\{\boldsymbol{x}:\left\|\boldsymbol{x}-\boldsymbol{y}_i\right\|^2 \le \left\|\boldsymbol{x}-\boldsymbol{y}_j\right\|^2\right\}, \tag{1.11}$$

care conține toți vectorii $\boldsymbol{x}$ mai apropiați de vectorul de reproducere $\boldsymbol{y}_i$ decât de vectorul de reproducere $\boldsymbol{y}_j$. Rezultă că o regiune de cuantizare R_i se poate exprima ca,

$$R_i = H_{i1} \cap H_{i2} \cap \ldots \cap H_{i\,i-1} \cap H_{i\,i+1} \cap \ldots \cap H_{iN}, \tag{1.12}$$

adică conține toți vectorii $\boldsymbol{x}$ mai apropiați de $\boldsymbol{y}_i$ decât de oricare alt vector de reproducere.

Înlocuind expresia distanței euclidiene (1.4) în (1.11), după reduceri se obține

$$H_{ij} = \left\{\boldsymbol{x}:2\left\langle\left(\boldsymbol{y}_i-\boldsymbol{y}_j\right)\boldsymbol{x}\right\rangle + \left\|\boldsymbol{y}_j\right\|^2 - \left\|\boldsymbol{y}_i\right\|^2 \ge 0\right\}. \tag{1.13}$$

Dacă se notează

$$\boldsymbol{u}_{ij} = 2(\boldsymbol{y}_i - \boldsymbol{y}_j), \tag{1.14}$$

şi

$$v_{ij} = \|\boldsymbol{y}_j\|^2 - \|\boldsymbol{y}_i\|^2 , \quad . \tag{1.15}$$

rezultă că

$$H_{ij} = \{\boldsymbol{x} : \langle \boldsymbol{u}_{ij} \cdot \boldsymbol{x} \rangle + v_{ij} \geq 0\},$$

şi ţinând cont de (1.12) se poate spune că regiunile de cuantizare au aceeaşi formă ca şi la un cuantizor vectorial regular şi politop, iar structura lor depinde de vectorii de reproducere, adică de cartea de cod.

În cazul în care măsura erorii de cuantizare este eroarea pătratică ponderată, relaţia (1.5), concluziile de mai sus nu mai sunt valabile în totalitate. Considerând mulţimea H_{ij} de forma

$$H_{ij} = \{\boldsymbol{x}: d(\boldsymbol{x}, \boldsymbol{y}_i) \leq d(\boldsymbol{x}, \boldsymbol{y}_j \} \tag{1.16}$$

unde pentru distanţa d se foloseşte eroarea pătratică ponderată, după efectuarea calculelor se obţine

$$H_{ij} = \{\boldsymbol{x}: 2\sum_{l=1}^{k} w_l x_l (y_{il} - y_{jl}) + \sum_{l=1}^{k} w_l (y_{jl}^2 - y_{il}^2) \geq 0 \} \tag{1.17}$$

Pentru cazul k=2 se obţine

$$H_{ij} = \{\boldsymbol{x}: 2w_1 x_1 (y_{i1} - y_{j1}) + 2w_2 x_2 (y_{i2} - y_{j2}) + w_1 (y_{j1}^2 - y_{i2}^2) + \\ + w_2 (y_{j2}^2 - y_{i2}^2) \geq 0 \} \tag{1.18}$$

În practică ponderile w_1 şi w_2 depind de componentele vectorului de cuantizat, adică $w_1 = w_1(x_1, x_2)$ şi $w_2 = w_2(x_1, x_2)$. Aceasta înseamnă, conform (1.18), că ecuaţiile frontierelor regiunilor de cuantizare nu conţin numai termeni în x_1 şi x_2, ci pot să conţină şi termeni de forma $(x_1 x_2)^2$, x_1^2 sau x_2^2 în funcţie de forma funcţiilor w_1 şi w_2. **Adică, regiunile de cuantizare nu mai sunt mărginite de drepte, ci de curbe mai complicate (pentru $k \geq 3$ de suprafeţe mai complicate)**. De aici rezultă că un astfel de cuantizor nu mai este regular şi politop. Acest fapt nu afectează însă aplicabilitatea metodelor de cuantizare vectorială folosite de autor. De asemenea, tot din (1.18) rezultă că structura regiunilor de cuantizare este impusă de vectorii de reproducere la fel ca şi în cazul erorii pătratice.

În figura 1.4 se prezintă structura regiunilor de cuantizare pentru un cuantizor de tip Voronoi, cu N=4 vectori de reproducere, k=2, și distanța euclidiană ca măsură a erorii de cuantizare. Sunt figurate și semiplanele de tip H_{ij} prin intersectarea cărora se obțin regiunile de cuantizare. Se observă că pentru fiecare segment de dreaptă (față) care delimitează două regiuni de cuantizare există câte 2 vectori de reproducere echidistanți.

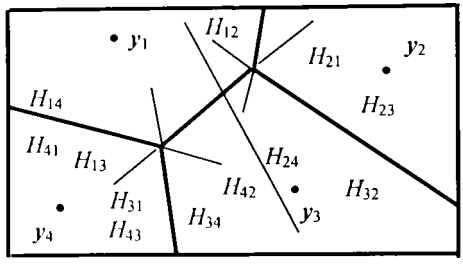


Figura 1.4: *Structura regiunilor de cuantizare pentru un cuantizor vectorial cu* N=4.

Se introduce noțiunea de *centroid* al unei regiuni $R_i \in R^k$: acel vector, y^*, în raport cu care se minimizează eroarea medie de cuantizare a vectorilor X care aparțin acelei regiuni R_i,

$$y^* = \text{cent}\ (R_i) \text{ dacă } E\left[d(X,y^*)\middle|X \in R_i\right] < E\left[d(X,y)\middle|X \in R_i\right], \qquad (1.19)$$

pentru orice $y \in \Re^k$. Dacă măsura erorii de cuantizare este eroarea pătratică, se arată, [1], că centroidul reprezintă chiar media vectorilor din regiunea respectivă,

$$\text{cent}\ (R_i) = E\ (X/X \in R_i) \qquad (1.20)$$

Pentru cazul când regiunea R_i conține un număr finit de vectori relația (1.20) se poate aplica fiind necesară funcția de probabilitate discretă $p_x(x)$. În practică însă, adesea se consideră că fiecare vector din R_i are aceeași probabilitate, egală cu $1/\|R_i\|$, unde $\|R_i\|$ reprezintă numărul de vectori din R_i. În această situație centroidul regiunii R_i este media aritmetică a vectorilor din R_i, adică

$$\text{cent}(R_i) = \frac{1}{\|R_i\|} \sum_{i=1}^{\|R_i\|} \boldsymbol{x}_i \tag{1.21}$$

Un cuantizor vectorial pentru care vectorul de reproducere din fiecare regiune de cuantizare este reprezentat de centroidul acelei regiuni, adică

$$\boldsymbol{y}_i = \text{cent}\ (R_i)\ ,\ i=1,...,N, \tag{1.22}$$

se spune că respectă condiţia centroidului.

În continuare se prezintă câteva noţiuni referitoare la condiţiile pe care trebuie să le îndeplinească un cuantizor vectorial pentru a fi *optim*, precum şi, ca o consecinţă, modul cum se proiectează un cuantizor vectorial.

Un cuantizor vectorial este optim dacă cartea de cod (decodorul) şi partiţia sau regula de codare (codorul) conduc împreună la minimizarea erorii medii de cuantizare D, reprezentată prin media statistică E [$d(\boldsymbol{X}, Q(\boldsymbol{X}))$]. În cazul în care vectorul de intrare, reprezentat de variabila aleatoare $\boldsymbol{X}$, are o distribuţie discretă, eroarea medie de cuantizare se specifică cu ajutorul funcţiei de probabilitate discretă $p_{\boldsymbol{x}}(\boldsymbol{x})$,

$$D = Ed(\boldsymbol{X}, Q(\boldsymbol{X})) = \sum_i d(\boldsymbol{x}_i, Q(\boldsymbol{x}_i)) p_{\boldsymbol{x}}(\boldsymbol{x}_i), \tag{1.23}$$

unde $\{\boldsymbol{x}_i\}$ sunt valorile variabilei aleatoare $\boldsymbol{X}$ care au probabilitatea diferită de 0.

Un cuantizor vectorial se numeşte *local* optimal dacă o mică modificare a vectorilor de reproducere nu poate produce scăderea erorii medii de cuantizare D, şi, respectiv *global* optimal, dacă nu există o altă carte de cod care să producă scăderea erorii medii de cuantizare D. Aşa cum se arată în [1], un cuantizor vectorial care satisface condiţia celei mai apropriate vecinătăţi şi condiţia centroidului, este cel puţin local optimal.

Pe baza celor două condiţii necesare pentru un optim local se realizează proiectarea iterativă a unui cuantizor vectorial.

Pentru aceasta, trebuie să se considere o carte de cod iniţială (în capitolul 3 vor fi date câteva metode de obţinere a unei astfel de cărţi de

cod), o măsură a erorii de cuantizare $d(\cdot,\cdot)$, precum şi o secvenţă de vectori numită de *antrenament*,

$$T=\{\boldsymbol{x}_1, \boldsymbol{x}_2,...,\boldsymbol{x}_n\}, \tag{1.24}$$

în care fiecare vector are probabilitatea egală cu $1/n$.

Algoritmul generalizat al lui Lloyd [1], care va fi descris în continuare, va preciza de câte ori sau până când se va aplica procesul iterativ de optimizare a codorului pentru decodorul dat (adică aplicarea condiţiei celei mai apropriate vecinătăţi) şi invers, a decodorului pentru codorul dat (aplicarea condiţiei centroidului). În acest scop se defineşte o iteraţie Lloyd care cuprinde două etape:

a) partiţionarea secvenţei de antrenament în regiuni de cuantizare, conform condiţiei celei mai apropriate vecinătăţi, în funcţie de o carte de cod $\boldsymbol{C}_m$, pe baza măsurii erorii de cuantizare, $d(\cdot,\cdot)$.

b) aplicarea condiţiei centroidului pentru regiunile de cuantizare obţinute la punctul a), în acest fel obţinându-se cartea de cod

$$\boldsymbol{C}_{m+1}=\{\text{cent}(R_i)\},\ i=1,....,N.$$

Se defineşte funcţia selector, caracteristică unui cuantizor vectorial,

$$S_i(\boldsymbol{x})=\begin{cases}1, daca\ \boldsymbol{x}\in R_i\\ 0, in\ rest\end{cases} \tag{1.25}$$

În acest caz centroidul unei regiuni R_j este [1]

$$cent(R_j)=\frac{\sum_{i=1}^{n}\boldsymbol{x}_i S_j(\boldsymbol{x}_i)}{\sum_{i=1}^{n}S_j(\boldsymbol{x}_i)}\ , \tag{1.26}$$

iar eroarea medie de cuantizare se poate exprima prin

$$D=\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{N}d(\boldsymbol{x}_i,\boldsymbol{y}_j)S_j(\boldsymbol{x}_i)\ , \tag{1.27}$$

unde i reprezintă indicii vectorilor de antrenament, iar j reprezintă indicii vectorilor de reproducere.

Având în vedere că pasul a) al iterației Lloyd îmbunătățește sau lasă neschimbat codorul față de decodorul dat, iar pasul b) îmbunătățește sau lasă neschimbat decodorul față de codorul rezultat la pasul a), rezultă că în urma unei iterații Lloyd, eroarea medie de cuantizare scade sau rămâne neschimbată.

În aceste condiții se prezintă structura algoritmului generalizat al lui Lloyd pentru proiectarea unui cuantizor vectorial optim.

1. Se consideră o carte de cod inițială; m=1.

2. Dându-se cartea de cod $\mathcal{C}_m$, se aplică o iterație Lloyd pentru a obține o variantă îmbunătățită a cărții de cod, notată $\mathcal{C}_{m+1}$.

3. Se calculează eroarea medie D_{m+1} a secvenței de antrenament față de cartea de cod $\mathcal{C}_{m+1}$.

4. Test de ieșire. Dacă raportul $\frac{D_m - D_{m+1}}{D_m}$ este mai mic decât o valoare de prag aleasă convenabil (cât mai mică, de exemplu 0,001), algoritmul se încheie. Dacă raportul este mai mare decât valoarea de prag, parametrul m ia valoarea m+1, și se sare la pasul 2.

Testul de ieșire verifică care este scăderea relativă a erorii la iterația curentă, D_{m+1}, față de cea de la iterația precedentă, D_m.

Algoritmul Lloyd, prin iterația Lloyd, nu modifică foarte mult cartea de cod. Acest fapt sugerează că odată aleasă cartea de cod inițială, algoritmul converge spre cel mai apropiat minim local față de cartea inițială, din spațiul tuturor cărților de cod posibile. Deoarece eroarea medie de cuantizare este o funcție complexă, care are în general mai multe minime locale, rezultă concluzia că algoritmul generalizat Lloyd nu găsește un optim global, ci un optim local.

În continuare se prezintă câteva dintre avantajele utilizării cuantizării vectoriale în raport cu cuantizarea scalară.

Considerând cazul vectorilor bidimensionali, de forma $\boldsymbol{x}=(x_1,x_2)$, prin cuantizarea scalară a celor două componente, cu câte 4 niveluri de reproducere, se obține un cuantizor vectorial echivalent cu 16 regiuni de cuantizare, ca în figura 1.5. Indiferent cum s-ar alege nivelurile de reproducere, forma regiunilor cuantizorului vectorial obținut este totdeauna dreptunghiulară. Prin aplicarea directă a cuantizării vectoriale, regiunile de

cuantizare pot să aibe forme mai complicate, adaptându-se la distribuţia secvenţei de antrenament, rezultând erori mai mici sau compresie.

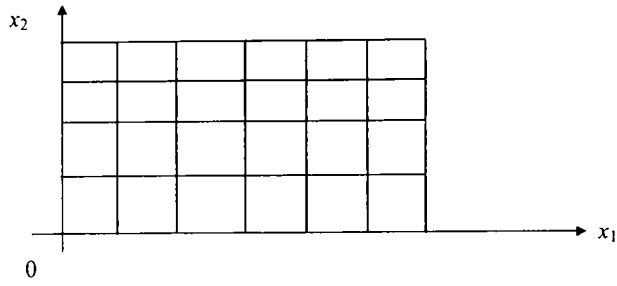


Figura 1.5: *Cuantizor vectorial obţinut prin cuantizarea scalară a componentelor.*

Tabelul 1.1: *Regula de cuantizare scalară.*

Eşantioane de cuantizat posibile	Nivelul de reproducere asociat	Codul (în zecimal)
1,2	1	0
3,4	3	1
5,6	5	2
7,8	7	3
9,10	9	4
11,12	11	5
13,14	13	6
15,16	15	7

Alt avantaj al cuantizării vectoriale rezultă când între componentele vectorilor există o dependenţă. De exemplu, dacă componentele vectorilor sunt eşantioanele unei forme de undă, între eşantioanele alăturate există o corelaţie în sensul că diferenţa între două eşantioane consecutive nu poate fi foarte mare, fiind în general mult mai mică decât gama dinamică a semnalului. Pentru a ilustra avantajul utilizării cuantizării vectoriale într-un astfel de caz, se consideră un semnal ale cărui eşantioane, în urma conversiei analog-numerice, iau valori între 1 şi 16, iar între două

eşantioane consecutive nu poate fi o diferenţă mai mare decât 6 (corelaţia de ordinul 1, între 2 eşantioane consecutive). Aceste eşantioane se pot cuantiza scalar, alocând de exemplu, 8 niveluri de reproducere echidistante: 1,3,5,...,15. Rata cuantizării este deci $\log_2 8=3$ biţi/eşantion. Cuantizarea se face după regula celei mai apropiate vecinătăţi, ca în tabelul 1.1.

Un eşantion egal depărtat de două niveluri se cuantizează prin cel cu indicele mai mic (de exemplu, 2 se cuantizează prin 1, nu prin 3). Grupând eşantioanele în vectori cu dimensiunea 2, se obţin vectorii de reproducere din tabelul 1.2, astfel încât diferenţa între componente să nu fie mai mare de 6.

Componentele vectorilor sunt aceleaşi cu nivelurile de reproducere de la cuantizarea scalară, dar datorită faptului că nu sunt posibile toate combinaţiile de câte două, se obţine un total de 44 vectori de reproducere. Rata cuantizării va fi în acest caz $\log_2 44=5{,}46$ biţi/vector, sau $5{,}46/2=2{,}73$ biţi/eşantion, adică mai mică decât la cuantizarea scalară, deci rezultă o compresie.

Tabelul 1.2: *Vectorii de reproducere* $(x_1 x_2)$.

1	1	3	1	5	1	7	1	9	3	11	5	13	7	15	9		
1	3	3	3	5	3	7	3	9	5	11	7	13	9	15	11		
1	5	3	5	5	5	7	5	9	7	11	9	13	11	15	13		
1	7	3	7	5	7	7	7	9	9	11	11	13	13	15	15		
		3	9	5	9	7	9	9	11	11	13	13	15				
				5	11	7	11	9	13	11	15						
						7	13	9	15								

Datorită faptului că vectorii de reproducere conţin aceleaşi componente ca şi nivelurile de reproducere de la cuantizarea scalară, eroarea medie şi cea maximă sunt egale în cele două cazuri (se foloseşte eroarea pătratică).

Dacă eşantioanele se grupează în vectori cu dimensiunea 3, impunând în plus restricţia ca între componentele 1 şi 3 să nu fie o diferenţă mai mare ca 10 (corelaţia de ordinul 2), se obţin vectorii de reproducere din tabelul 1.3 (doar cei care au prima componentă 1).

Se obţin în total 21 vectori de reproducere. Determinând şi ceilalţi vectori de reproducere posibili, adică cei cu prima componentă 3,5,...,15 se obţin 248 vectori, iar rata este $\log_2 248$=7,95 biţi/vector sau 7,95/3=2,65 biţi/eşantion, deci compresia este mai mare decât în cazul k=2. Dimensiunea vectorilor, k, poate fi crescută până când corelaţia de ordinul k-1 mai este suficient de puternică între componente.

Tabelul 1.3: *Vectorii de reproducere* (x_1 x_2 x_3).

1 1 1	1 3 1	1 5 1	1 7 1
1 1 3	1 3 3	1 5 3	1 7 3
1 1 5	1 3 5	1 5 5	1 7 5
1 1 7	1 3 7	1 5 7	1 7 7
	1 3 9	1 5 9	1 7 9
		1 5 11	1 7 11

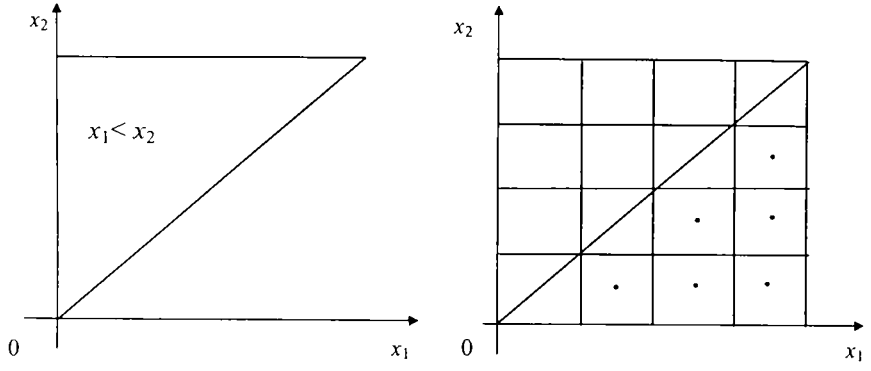
Desigur că în aceste exemple s-au considerat situaţii simple, pur teoretice, dar totuşi sugestive. În practică, pentru semnale reale, se obţin compresii mai puternice.

În concluzie, se poate spune că, datorită faptului că nu toate combinaţiile de 2, 3 sau mai multe componente sunt posibile, cartea de cod le "învaţă", pe baza secvenţei de antrenament, doar pe cele mai plauzibile, în acest fel rezultând un număr echivalent de biţi pe eşantion mai mic.

Pe baza exemplelor prezentate, autorul a demonstrat proprietatea de compresie a cuantizării vectoriale.

O altă situaţie în care autorul arată avantajul cuantizării vectoriale în raport cu cuantizarea scalară este cea în care vectorii au componentele astfel încât $x_1 < x_2 < \ldots < x_k$. Pentru simplitate se consideră cazul k = 2. În figura 1.6 a) se prezintă regiunea în care aceşti vectori sunt uniform distribuiţi. Utilizând cuantizarea scalară pentru componentele x_1 şi x_2 cu câte 4 niveluri (2 biţi/componentă), se obţine un cuantizor vectorial cu 16 regiuni de cuantizare, figura 1.6 b). Se observă că 6 dintre regiuni, cele notate cu puncte în interior, nu sunt în zona în care sunt distribuiţi vectorii. Deci cuantizarea vectorială poate folosi doar 10 regiuni, adică $\log_2 10$=3,32

biţi/vector sau 3,32/2=1,66 biţi/componentă, adică se obţine o compresie. O altă îmbunătăţire s-ar putea face dacă cele 16 regiuni de cuantizare ale cuantizorului vectorial s-ar dispune numai în zona unde sunt distribuiţi vectorii. În acest fel, regiunile de cuantizare ar avea o suprafaţă mai mică, permiţând scăderea erorii de cuantizare, la o rată egală cu cea de la cuantizarea scalară.



a) b)

Figura 1.6: *Cuantizor vectorial pentru* k=2, $x_1 < x_2$.

În finalul acestui capitol se prezintă şi alte caracteristici importante ale unui cuantizor vectorial, pe lângă cele prezentate anterior, adică rata şi eroarea medie de cuantizare.

Complexitatea unui cuantizor vectorial este o mărime care reprezintă numărul de operaţii aritmetice care se fac pentru cuantizarea unui vector. Desigur că operaţia de codare necesită cele mai multe operaţii, decodarea fiind o operaţie de citire de la o adresă. De aceea, în cele mai multe cazuri, complexitatea se referă în exclusivitate la operaţiile de codare. Operaţiile aritmetice care se fac pot fi înmulţiri, adunări, comparaţii. De obicei se consideră numai operaţiile de înmulţire, deoarece acestea consumă mai mult timp. Dar, deoarece actualele procesoare de semnal pot să facă înmulţirile foarte rapid, este mai corect să se considere şi celelalte operaţii.

Astfel, în cazul general în care codarea se face prin căutare totală, în care se codează vectori k-dimensionali, cartea de cod are N elemente, iar măsura erorii de cuantizare este eroarea pătratică, pentru codarea unui vector

se fac Nk înmulţiri (ridicările la pătrat), Nk scăderi (diferenţele între componentele omoloage ale vectorilor) şi $N(k\text{-}1)$ adunări (însumarea celor k diferenţe). În plus mai intervin şi comparaţiile necesare pentru determinarea erorii minime.

Memoria unui cuantizor vectorial reprezintă memoria folosită în procesul de cuantizare a unui vector şi este dată în cazul unei codări prin căutare totală de numărul de locaţii de memorie necesare pentru memorarea vectorilor de reproducere, adică $2Nk$ octeţi, dacă fiecare componentă se memorează prin 2 octeţi.

Când se analizează performanţele unui cuantizor vectorial se iau în considerare aceste caracteristici adică: rata, eroarea medie, complexitatea şi memoria. Desigur că ideal ar fi ca toate aceste caracteristici să aibe valori cât mai mici, dar ele depind una de alta şi de aceea se caută un compromis. De exemplu, se impune o rată şi se doreşte minimizarea erorii, sau se impune o eroare şi se încearcă minimizarea complexităţii sau alte variante. În acest fel se obţin cuantizoare vectoriale numite *suboptimale*. Cartea de cod pentru un astfel de cuantizor are o structură care nu respectă cele două condiţii de optim, permiţând o căutare mai rapidă în vederea determinării codului vectorului de cuantizat. Datorită acestor cauze, vectorul de reproducere găsit nu este cel care produce eroarea de cuantizare minimă posibilă. Avantajul obţinut este însă reducerea complexităţii şi (sau) a memoriei, eroarea crescând însă mai puţin, astfel că, pe ansamblu, analizând toate mărimile caracteristice ale cuantizorului vectorial se poate aprecia că s-a obţinut o îmbunătăţire.

În capitolul 1 autorul a prezentat sub o formă sistematizată principalele noţiuni şi principii utilizate la cuantizarea vectorială, precum şi unele contribuţii originale.

2. CUANTIZAREA VECTORIALĂ A PARAMETRILOR LSF

2.1 Obţinerea parametrilor LSF. Proprietăţi ale parametrilor LSF

Cuantizarea vectorială se aplică în prezent la semnalul vocal şi la cel de imagine. În cazul prelucrării semnalului vocal, domeniu studiat de către autor, cuantizarea vectorială s-a aplicat la început pentru vectori formaţi din eşantioane consecutive prelevate din semnalul vocal, iar apoi pentru vectori formaţi din diferite forme ale parametrilor obţinuţi prin predicţie liniară (se numesc parametri LPC-linear predictiv coding). De obicei, predicţia liniară se face pentru porţiuni de semnal vocal care conţin 20-25 ms, numite cadre, [10], [11]. În ultimul timp [4], [5], [6] s-a arătat că reprezentarea sub formă de linii spectrale de frecvenţă (LSF) a parametrilor LPC, dă rezultate foarte bune la cuantizarea vectorială.

Parametrii LSF se obţin din parametrii LPC printr-o transformare 1 la 1, adică dintr-un număr de parametri LPC (sub formă de coeficienţi de predicţie) se obţine acelaşi număr de parametri LSF, [43]. Astfel, dacă se face o analiză prin predicţie liniară de ordinul p, se obţin p coeficienţi de predicţie $a_1, a_2,...,a_p$. $A(z)$ este polinomul construit cu aceşti coeficienţi,

$$A(z) = 1 - a_1 z^{-1} - a_2 z^{-2} - \ldots - a_p z^{-p}, \tag{2.1}$$

iar apoi cu $A(z)$ se formează polinoamele

$$P(z) = A(z) + z^{-(p+1)} A(z^{-1}) \tag{2.2}$$

şi

$$Q(z) = A(z) - z^{-(p+1)} A(z^{-1}) \tag{2.3}$$

Pe baza rădăcinilor polinoamelor $P(z)$ şi $Q(z)$ se obţin parametrii LSF. În cazul $p = 10$ (folosit în marea majoritate a cazurilor practice), polinoamele $P(z)$ şi $Q(z)$ au gradul 11. Datorită faptului că au coeficienţi reali, pe lângă rădăcinile reale, -1 pentru $P(z)$ şi +1 pentru $Q(z)$, aceste polinoame au

rădăcini complex conjugate. Rădăcinile complex conjugate ale acestor polinoame au următoarele proprietăţi [4]:

1° Se găsesc pe cercul unitar, deci au modulul egal cu 1.

2° Dacă argumentele, în valoare absolută, ale acestor rădăcini se aranjează în ordine crescătoare, vor fi intercalate.

Dacă rădăcinile lui $P(z)$ se notează: $e^{\pm j\omega_1}, e^{\pm j\omega_3}, \ldots, e^{\pm j\omega_9}$, iar cele ale lui $Q(z)$: $e^{\pm j\omega_2}, e^{\pm j\omega_4}, \ldots, e^{\pm j\omega_{10}}$, proprietatea 2° se poate reprezenta sub forma:

$$\omega_1 < \omega_2 < \omega_3 < \ldots < \omega_{10}. \tag{2.4}$$

Parametrii LSF se obţin prin relaţia:

$$f_i = \frac{\omega_i}{2\pi} f_e,\ i = 1,2,\ldots,10, \tag{2.5}$$

unde f_e reprezintă frecvenţa de eşantionare. Dacă de exemplu, f_e=8kHz, parametrii LSF vor avea valori în intervalul (0, 4000) Hz. Uneori, parametrii LSF se consideră sub formă normalizată ω_i, având valori în intervalul (0,π).

Pentru cuantizarea vectorială, vectorii se formează din cei 10 parametri LSF corespunzători cadrelor, care, fiind în ordine crescătoare, prezintă un avantaj aşa cum s-a arătat în capitolul 1. Dacă în urma cuantizării, parametrii LSF îşi păstrează proprietatea 2°, polinomul $A(z)$ îşi păstrează proprietatea de fază minimă, filtrul de reconstituire al semnalului vocal $H(z)=1/A(z)$ fiind stabil [4]. Polinomul $A(z)$ se reconstituie uşor din parametrii LSF cuantizaţi, deoarece se ştie că pe baza parametrilor cu indice impar se formează $P(z)$, iar din cei cu indice impar $Q(z)$, iar apoi

$$A(z) = (P(z) + Q(z)) / 2. \tag{2.6}$$

Coeficienţii polinomului $A(z)$ astfel obţinut reprezintă coeficienţii de predicţie a_i.

Pentru a determina legătura dintre liniile spectrale de frecvenţă şi densitatea spectrală de putere normalizată, $S(f)$, calculată pe baza coeficienţilor de predicţie (pe scurt, spectrul LPC)

$$S(f) = \frac{1}{\left|A(e^{j2\pi f/f_e})\right|^2}, \tag{2.7}$$

în figura 2.1 se arată forma de variaţie a spectrului LPC pentru un cadru de semnal vocal achiziţionat de autor şi liniile spectrale de frecvenţă aferente, reprezentate prin liniile verticale, acestea având următoarele valori, în Hz: 204, 329, 789, 899, 1488, 1671, 2030, 2604, 2925, 3475.

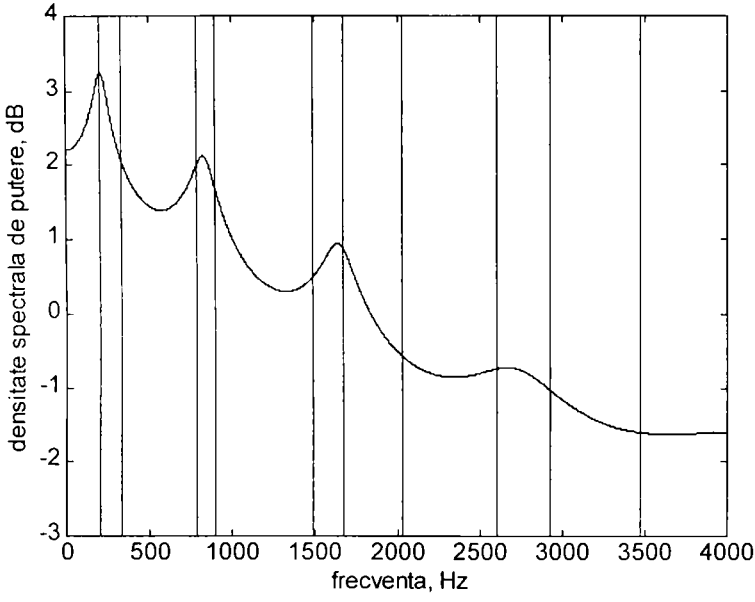


Figura 2.1: *Spectrul* LPC *şi parametrii* LSF *aferenţi.*

Din figura 2.1 se observă că grupuri de 2 sau 3 parametri LSF caracterizează un vârf din spectru (un formant în cazul unui semnal vocal), iar lăţimea vârfului este proporţională cu intervalul de frecvenţă acoperit de parametrii LSF respectivi.

În continuare, autorul dezvoltă, pe baza unor experimente ale căror rezultate se vor justifica teoretic, legătura dintre spectrul LPC şi parametrii LSF, în scopul de a determina o reprezentare simplificată a spectrului LPC, în funcţie de parametrii LSF.

În cazul în care $|z|=1$, valabil pentru rădăcinile polinoamelor $P(z)$ şi $Q(z)$, rezultă:

$$z = \cos\omega + j\sin\omega = e^{j\omega} \tag{2.8 a}$$

şi

$$z^* = \cos\omega - j\sin\omega = e^{-j\omega} = \frac{1}{z} = z^{-1}. \tag{2.8 b}$$

Ştiind că

$$|A(z)|^2 = A(z)\cdot A(z^*),$$

se obţine

$$|A(z)|^2 = A(z)\cdot A(z^{-1}) = \frac{P(z)+Q(z)}{2}\cdot\frac{P(z^{-1})+Q(z^{-1})}{2}$$

Deoarece, pe baza relaţiilor de definiţie ale polinoamelor $P(z)$ şi $Q(z)$, (2.2) şi (2.3), se poate demonstra că

$$P(z)Q(z^{-1}) + P(z^{-1})Q(z) = 0,$$

rezultă că

$$|A(z)|^2 = \frac{|P(z)|^2 + |Q(z)|^2}{4} \tag{2.9}$$

Apoi, pe baza relaţiei (2.7) se obţine:

$$S(f) = \frac{4}{|P(z)|^2 + |Q(z)|^2}, \tag{2.10}$$

unde $z = e^{j\omega} = e^{j\frac{2\pi f}{f_e}}$. Pe de altă parte, ţinând cont de rădăcinile pe care le au, polinoamele $P(z)$ şi $Q(z)$ se pot exprima ca

$$P(z) = z^{-11}(z+1)(z-e^{j\omega_1})(z-e^{-j\omega_1})(z-e^{j\omega_3})\cdot \tag{2.11}$$
$$\cdot(z-e^{-j\omega_3})\cdot\ldots\cdot(z-e^{j\omega_9})(z-e^{-j\omega_9})$$

şi

$$Q(z) = z^{-11}(z-1)(z-e^{j\omega_2})(z-e^{-j\omega_2})(z-e^{j\omega_4})\cdot \tag{2.12}$$
$$\cdot(z-e^{-j\omega_4})\cdot\ldots\cdot(z-e^{j\omega_{10}})(z-e^{-j\omega_{10}})$$

Ştiind că $\omega_i = 2\pi f_i / f_e$, înlocuind $P(z)$ şi $Q(z)$ date de relaţiile (2.11) şi (2.12) în (2.10) se obţine o expresie a spectrului LPC, $S(f)$, în funcţie de parametrii LSF, f_i.

Pentru calculul valorilor $S(f)$ pentru valorile $f_1, f_3, \ldots, f_9$ se observă că, în oricare din aceste puncte, polinomul $P(z)$ se anulează, iar pentru oricare

din valorile $f_2, f_4, \ldots, f_{10}$ se anulează polinomul $Q(z)$. În concluzie, se poate spune că în punctele $f_1, f_3, \ldots, f_9$ spectrul depinde numai de rădăcinile lui $Q(z)$ adică $f_2, f_4, \ldots, f_{10}$, iar în punctele $f_2, f_4, \ldots, f_{10}$, spectrul depinde de rădăcinile lui $P(z)$, adică $f_1, f_3, \ldots, f_9$,

$$S(f_k)=\begin{cases}\dfrac{4}{\left|Q(e^{j2\pi f_k/f_e})\right|^2}, & \text{pentru } k=1,3,\ldots,9\\ \dfrac{4}{\left|P(e^{j2\pi f_k/f_e})\right|^2}, & \text{pentru } k=2,4,\ldots,10\end{cases} \tag{2.13}$$

Înlocuind în relația (2.13) expresiile lui $P(z)$ și $Q(z)$ date de relațiile (2.11) și (2.12), se obține pentru spectrul $S(\omega_k)$ relația:

$$S(\omega_k)=\begin{cases}\dfrac{4}{\left|z^{11}(z-e^{j0})(z-e^{j\omega_2})(z-e^{-j\omega_2})\cdot\ldots\cdot(z-e^{j\omega_{10}})(z-e^{-j\omega_{10}})\right|^2}\\ \text{pentru } k=1,3,\ldots,9\\ \dfrac{4}{\left|z^{11}(z-e^{j\pi})(z-e^{j\omega_1})(z-e^{-j\omega_1})\cdot\ldots\cdot(z-e^{j\omega_9})(z-e^{-j\omega_9})\right|^2}\\ \text{pentru } k=2,4,\ldots,10\end{cases} \tag{2.14}$$

unde $z = e^{j\omega_k}$. Deoarece $f_k = \dfrac{\omega_k}{2\pi} f_e$, $S(f_k)$ și $S(\omega_k)$ sunt echivalente.

Numitorul lui $S(\omega_k)$ exprimat prin (2.14) conține un produs de 11 factori, deoarece $|z^{11}|=1$.

Pentru un caz oarecare, de exemplu $f=f_3$ sau $z=e^{j\omega_3}=e^{j2\pi f_3/f_e}$, în figura 2.2 se arată cum se obțin cei 11 factori cu ajutorul cercului trigonometric. Pe cerc sunt figurate punctele de argument $\omega = \pm\omega_2$, $\omega = \pm\omega_4$, $\omega = \pm\omega_6$, $\omega = \pm\omega_8$, $\omega = \pm\omega_{10}$ precum și punctul de argument 0. Se obțin 11 segmente care unesc punctul $\omega = \omega_3$ cu cele 11 puncte amintite anterior. Ridicând la pătrat fiecare din măsurile celor 11 segmente, se obțin cei 11 factori care constituie numitorul expresiei din (2.14).

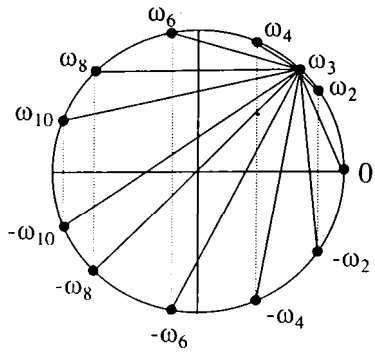


Figura 2.2: *Ilustrativă pentru obținerea celor* 11 *factori din expresia spectrului în punctul* $f=f_3$.

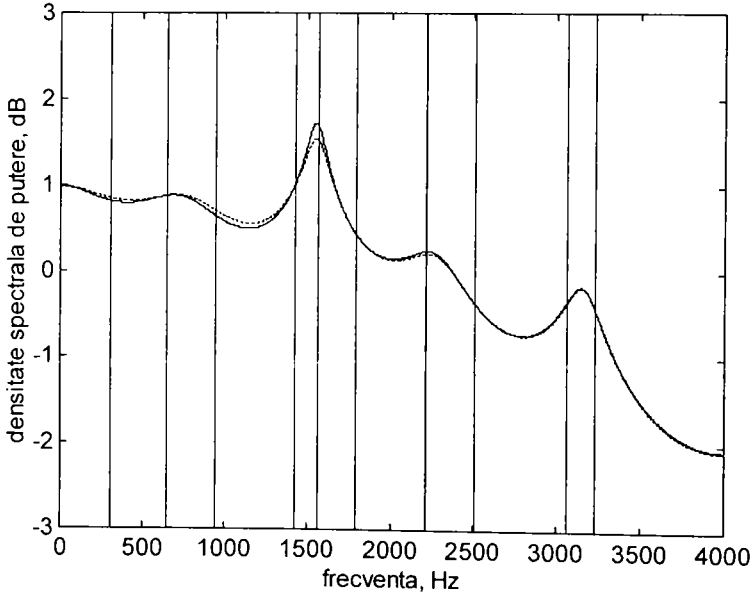


Figura 2.3: *Spectrul inițial (cu linie continuă) și spectrul modificat obținut prin schimbarea valorii parametrului* f_4.

În figura 2.3 se prezintă modificarea produsă în spectrul LPC datorită modificării valorii unui singur parametru LSF. Astfel, graficul desenat cu

linie continuă s-a obţinut pentru următoarele valori ale parametrilor LSF, în Hz (obţinute experimental): 307, 646, 940, 1425, 1558, 1788, 2220, 2515, 3065, 3234. Graficul desenat cu linie întreruptă s-a obţinut pentru cazul când f_4=1395 Hz, ceilalţi 9 parametrii nemodificându-se.

Aşa cum s-a arătat anterior din punct de vedere teoretic, din figură se observă că, în punctele $f=f_2$, $f=f_6$, $f=f_8$, şi $f=f_{10}$, spectrul nu este afectat de modificarea parametrului f_4. Cele mai mari modificări se produc în punctele $f=f_3$ şi $f=f_5$. În punctul $f=f_3$, modificarea spectrului are loc, aşa cum rezultă din analiza figurii 2.2, datorită modificării segmentelor care unesc punctul de argument ω_3 cu punctele de argumente ω_4 şi $-\omega_4$. Este evident că modificarea segmentului ω_3; ω_4 este mult mai mare decât cea a segmentului ω_3;$-\omega_4$. În mod asemănător se poate explica modificarea spectrului în punctul $f=f_5$, care este de asemenea învecinat cu f_4, parametrul modificat.

Tot din figura 2.3 se observă că în punctele $f=f_1$ şi $f=f_7$, modificările spectrului sunt mai mici decât în $f=f_3$ şi $f=f_5$. Deoarece punctele $f=f_1$ şi $f=f_7$ sunt mai depărtate de f_4 , modificarea lui f_4 produce o modificare mai mică a segmentelor ω_1; ω_4 şi ω_7; ω_4 faţă de segmentele ω_3; ω_4 şi ω_5; ω_4 . Şi în acest caz, modificarea segmentelor ω_1; $-\omega_4$ şi ω_7; $-\omega_4$ este mică.

În fine, în punctul $f=f_9$ nu se observă o modificare a spectrului, deoarece acest punct este cel mai îndepărtat de f_4 , segmentele ω_9; ω_4 şi ω_9; $-\omega_4$ având variaţii foarte mici.

În continuare se doreşte obţinerea unei expresii mai simple pentru $S(f_k)$. Pentru aceasta se consideră cazul unui semnal vocal pentru care parametrii LSF sunt egali depărtaţi în intervalul (0,...,4000)Hz, adică:

$$f_k = k \cdot \frac{4000}{p+1} = k \cdot \frac{4000}{11}, k = 1,2,\ldots,10, \tag{2.15}$$

sau

$$\omega_k = k \cdot \frac{\pi}{p+1} = k \cdot \frac{\pi}{11}. \tag{2.16}$$

Se notează

$$\alpha = \omega_{k+1} - \omega_k = \frac{\pi}{p+1} = \frac{\pi}{11}, \tag{2.17}$$

reprezentând diferenţa dintre 2 parametri LSF consecutivi.

În cele ce urmează **se va demonstra că, în acest caz, densitatea spectrală de putere are valoarea 1 la oricare dintre valorile parametrilor LSF**, [47]. Mai întâi se găseşte o altă exprimare pentru termenii de forma $\left|(z - e^{\pm j\omega_i})\right|^2$ situaţi la numitorul expresiilor din relaţia (2.14).

$$\begin{aligned}
&\left|z - e^{j\omega_i}\right|^2 = \left|e^{j\omega} - e^{j\omega_i}\right|^2 = \left|\cos\omega + j\sin\omega - \cos\omega_i - j\sin\omega_i\right|^2 = \\
&= \left|-2\sin\frac{\omega+\omega_i}{2}\sin\frac{\omega-\omega_i}{2} + 2j\sin\frac{\omega-\omega_i}{2}\cos\frac{\omega+\omega_i}{2}\right|^2 = \\
&= 4\sin^2\frac{\omega-\omega_i}{2}
\end{aligned} \tag{2.18}$$

Înlocuind relaţia (2.18) în relaţia (2.14) se obţine următoarea exprimare pentru $S(\omega_k)$

$$S(\omega_k) = \begin{cases} \dfrac{1}{2^{20}\sin^2\dfrac{\omega_k}{2}\prod\limits_{i=1}^{5}\sin^2\dfrac{\omega_k-\omega_{2i}}{2}\sin^2\dfrac{\omega_k+\omega_{2i}}{2}}, \\ \text{pentru } k = 1,3,\ldots,9 \\ \dfrac{1}{2^{20}\cos^2\dfrac{\omega_k}{2}\prod\limits_{i=1}^{5}\sin^2\dfrac{\omega_k-\omega_{2i-1}}{2}\sin^2\dfrac{\omega_k+\omega_{2i-1}}{2}}, \\ \text{pentru } k = 2,4,\ldots,10. \end{cases} \tag{2.19}$$

În continuare, pe baza relaţiei (2.17), $S(\omega_k)$ poate fi exprimată sub forma:

$$S(\omega_k)=\frac{1}{(2^5\sin\frac{\alpha}{2}\sin\frac{3\alpha}{2}\sin\frac{5\alpha}{2}\sin\frac{7\alpha}{2}\sin\frac{9\alpha}{2})^4} \qquad (2.20)$$

pentru $k=1,2,\ldots,10$

adică, din cele 11 segmente reprezentate în figura 2.2, unul are valoarea 2, iar celelalte 10 sunt egale două câte două.

Prin transformări trigonometrice elementare se obţine succesiv:

$$S(\omega_k)=\frac{1}{(2^5\sin\frac{\alpha}{2}\sin\frac{3\alpha}{2}\sin\frac{5\alpha}{2}\cos2\alpha\cdot\cos\alpha)^4}=$$

$$=\frac{1}{(2^5\sin\frac{\alpha}{2}\cos4\alpha\cos3\alpha\cos2\alpha\cos\alpha)^4}= \qquad (2.21)$$

$$=(\frac{\cos\frac{\alpha}{2}}{2^4\sin\alpha\cos\alpha\cos2\alpha\cos4\alpha\cos3\alpha})^4=$$

$$=(\frac{\cos\frac{\alpha}{2}}{2\sin8\alpha\cos3\alpha})^4=(\frac{\cos\frac{\alpha}{2}}{2\sin3\alpha\cos3\alpha})^4=(\frac{\cos\frac{\alpha}{2}}{\sin6\alpha})^4=1$$

adică

$$S(\omega_k)=1,\ \text{pentru}\ \omega=\omega_k,\ k=1,2,\ldots,10 \qquad (2.22)$$

Pentru obţinerea unei formule aproximative pentru $S(\omega_k)$, valabilă pentru orice valori ω_k, se fac nişte aproximări în relaţia (2.21) astfel:

-funcţiile sinus se înlocuiesc cu argumentul, ceea ce înseamnă, referitor la cercul trigonometric din figura 2.2, că cele mai mici 6 segmente se înlocuiesc cu arcele corespunzătoare.

-funcţiile cosinus se înlocuiesc cu valoarea 1, adică cele mai mari 4 segmente, din cercul trigonometric, se înlocuiesc cu valoarea 2.

În acest fel, se obţine:

$$S(\omega_k)=K\cdot\frac{1}{\alpha^{12}}, \qquad (2.23)$$

în care α se stabileşte ţinând cont de următoarele:

1. $S(\omega_k)$ pentru k=2,4,...,10 depinde numai de valorile ω_k, k=1,3,...,9 iar pentru k=1,3,...,9 depinde de valorile ω_k, k=2,4,...,10.
2. $S(\omega_k)$ pentru k=1,2,...,10 depinde în special de valorile ω_k mai apropiate, aşa cum autorul a demonstrat anterior.

Constanta K se stabileşte astfel încât pentru repartiţia uniformă a parametrilor LSF, să se obţină $S(\omega_k)$ =1, pentru k=1,2,...,10.

Astfel, pentru a se obţine cea mai simplă formă de exprimare a $S(\omega_k)$, α se exprimă ca diferenţă faţă de cei 2 parametri LSF învecinaţi, adică:

$$S(f_k) = K \cdot \frac{1}{(f_{k+1} - f_k)^6 (f_k - f_{k-1})^6} \tag{2.24}$$

unde $K = \left[\frac{f_e}{2(p+1)}\right]^{12}$, şi K=5,35x10^{30} pentru f_e=8000Hz şi p=10.

Făcându-se din nou referire la cercul trigonometric din figura 2.2, expresia lui $S(f_k)$ dată prin relaţia (2.24) calculează cele mai mici 2 segmente care au punctul comun în punctul de argument ω_k. Celelalte 4 segmente care au fost înlocuite cu arcele corespunzătoare pentru obţinerea relaţiei (2.23), se consideră că au valori proporţionale cu cele 2 segmente mai mici, constantele de proporţionalitate fiind incluse în constanta K.

O altă relaţie pentru $S(f_k)$ care permite aproximarea cu erori (medii) mai mici, se obţine dacă se utilizează pentru calculul lui α nu numai frecvenţele f_{k-1} şi f_{k+1} ci şi f_{k-3} şi f_{k+3} . În acest fel sunt necesare pentru calculul lui $S(f_k)$, k=1,2,...,10, frecvenţele f_k, k=-2,-1,0,1,...,12,13, aşa cum rezultă din tabelul 2.1.

Tabelul 2.1: *Valorile lui* f_k, k=-2,-1,0,1,...,12,13.

k	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13
f_k	$-f_2$	$-f_1$	0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	$f_e/2$	f_e-f_{10}	f_e-f_9

Pentru mărimea α din relaţia (2.23), se consideră o valoare care depinde de valorile mai mari decât f_k,

$$\alpha = \frac{3(f_{k+1} - f_k) + (f_{k+3} - f_k)}{6} = \frac{f_{k+3} + 3f_{k+1} - 4f_k}{6} \qquad (2.25)$$

pentru k=1,2,...,10,

respectiv o valoare care depinde de valorile mai mici decât f_k,

$$\alpha = \frac{3(f_k - f_{k-1}) + (f_k - f_{k-3})}{6} = \frac{4f_k - 3f_{k-1} - f_{k-3}}{6} \qquad (2.26)$$

pentru k=1,2,...,10.

Utilizarea acestor valori pentru α conduce pe baza relației (2.23) la:

$$S(f_k) = K_1 \cdot \frac{1}{(f_{k+3} + 3f_{k+1} - 4f_k)^6 (4f_k - 3f_{k-1} - f_{k-3})^6} \qquad (2.27)$$

pentru k=1,2,...,10.

unde $K_1 = 6^{12}$ $K = (\frac{3f_e}{p+1})^{12}$ și $K_1 = 1{,}16 \cdot 10^{40}$, pentru p=10 și f_e=8000 Hz.

Adoptarea relațiilor (2.25) și (2.26) pentru α, se justifică prin aceea că trebuie ținut cont cumva că parametrii LSF nu sunt uniform distribuiți. În acest sens, α ia o valoare medie în care intervine și următorul parametru de care depinde $S(f_k)$, adică f_{k+3} și, respectiv f_{k-3}. Dacă se consideră că intervalul $f_{k+3} - f_k$ este egal cu de 3 ori intervalul $f_{k+1} - f_k$, atunci se face o împărțire la 6, pentru a se obține o valoare medie a lui α.

În acest fel autorul a obținut două relații aproximative și simplificate, 2.24 și 2.27, pentru calculul densității spectrale de putere. Aceste relații vor fi folosite așa cum se va vedea în capitolul 3, la calculul erorii pătratice ponderate, folosită la cuantizarea vectorială a parametrilor LSF. Pentru o primă evaluare a acestor relații, în tabelul 2.2 se arată, pentru 2 cadre diferite obținute experimental, cu câte 10 parametri LSF, f_{k1} și f_{k2}, valorile densității spectrale de putere calculate cu cele 2 relații propuse, comparativ cu valoarea exactă calculată cu relația 2.19.

Privind rezultatele din tabelul 2.2, se poate observa, așa cum se anticipa, o diferență mai mică, în raport cu valoarea exactă calculată cu

relaţia (2.19), a valorilor $S(f_k)$ calculate cu relaţia (2.27), şi respectiv o diferenţă mai mare a valorilor calculate cu relaţia (2.24). Trebuie însă subliniat că, la eroarea pătratică ponderată, unde se vor folosi aceste valori, este necesară puterea 1/6 din $S(f_k)$, ceea ce are ca efect apropierea acestor valori. De asemenea interesează variaţia relativă a valorilor $S(f_k)$ alăturate, necontând aşa de mult valoarea lor absolută.

Tabelul 2.2: *Valorile lui $S(f_k)$ pentru 2 cadre de semnal vocal.*

k	1	2	3	4	5	6	7	8	9	10
f_{k1} [Hz]	257	456	716	998	1538	1668	2230	2506	2972	3526
$S(f_{k1})$ cu rel.(2.24)	298	278	34	0,42	44	35	0,38	1,17	0,01	0,01
$S(f_{k1})$ cu rel.(2.27)	189	55	30	1,35	4,75	2,78	0,20	0,59	0,02	0,02
$S(f_{k1})$ cu rel.(2.19)	198	59	49	4,27	8,39	5,33	0,36	0,51	0,04	0,04
f_{k2} [Hz]	190	435	817	988	1567	1815	2214	2591	2910	3418
$S(f_{k2})$ cu rel.(2.24)	524	7,94	68	5,66	0,60	5,68	0,46	1,76	0,29	0,01
$S(f_{k2})$ cu rel.(2.27)	307	13	17	3,38	0,68	2,86	0,35	0,66	0,14	0,01
$S(f_{k2})$ cu rel.(2.19)	205	20	33	9,4	1,67	2,51	0,43	0,53	0,15	0,03

În figura 2.4 se prezintă grafic valorile $S(f_k)$ în punctele corespunzătoare parametrilor LSF, calculate cu (2.24) (cu linie întreruptă) comparativ cu cele calculate cu relaţia exactă (2.19) (cu linie continuă), iar în figura 2.5 se prezintă grafic valorile $S(f_k)$ calculate cu (2.27) (cu linie întreruptă) comparativ cu cele calculate cu relaţia exactă (2.19). Din aceste grafice se observă că valorile $S(f_k)$ calculate cu (2.24) au variaţii mai mari

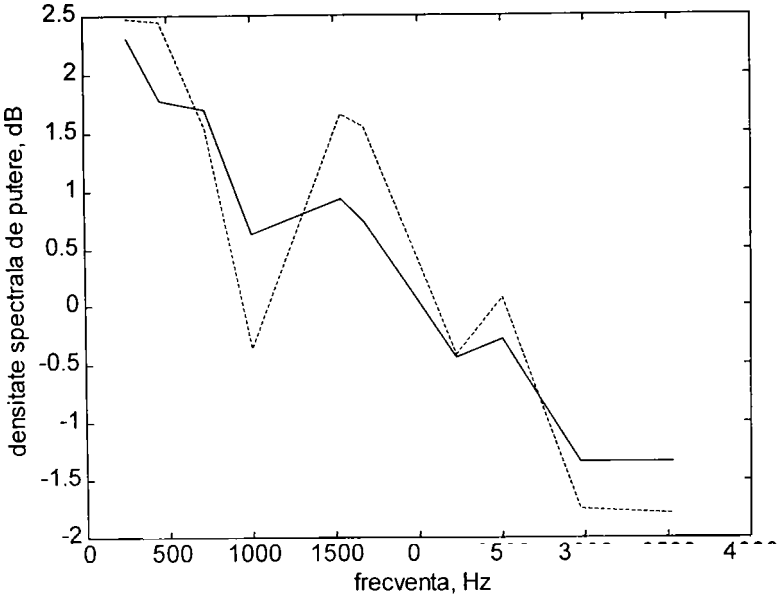


Figura 2.4: *Densitatea spectrală de putere calculată cu relaţiile* (2.24), (*cu linie întreruptă*) *şi* (2.19).

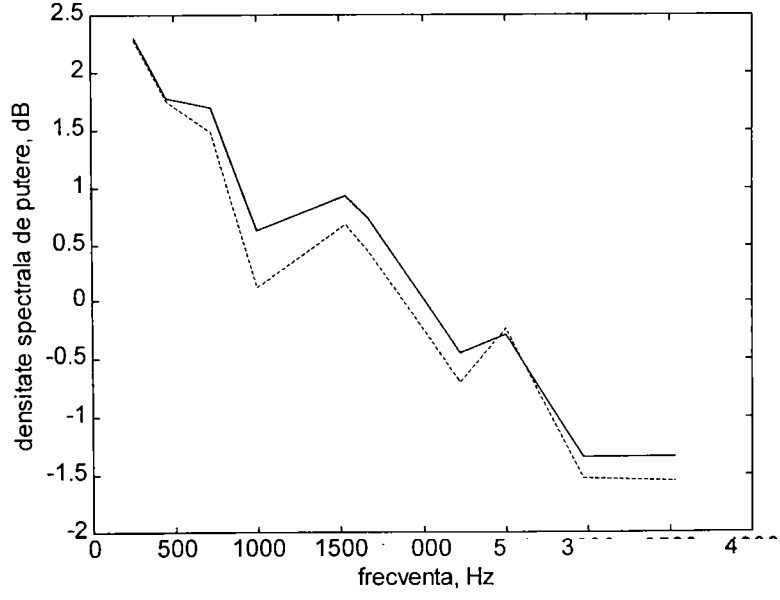


Figura 2.5: *Densitatea spectrală de putere calculată cu relaţiile* (2.27) (*cu linie întreruptă*) *şi* (2.19).

decât cele calculate cu (2.27). Explicaţia este că, dacă cel puţin unul din intervalele $f_{k+1} - f_k$ sau $f_k - f_{k-1}$ sunt mici, aproximarea făcută consideră că şi celelalte intervale sunt la fel de mici, (în realitate acestea sunt mai mari) rezultând valori mai mari pentru $S(f_k)$. Justificarea este similară când cel puţin unul din intervalele menţionate anterior sunt mari.

În concluzie, avantajul principal obţinut prin utilizarea celor două relaţii propuse de autor este reducerea cantităţii de calcul.

2.2 Metode de cuantizare vectorială ale parametrilor LSF

În deschiderea acestui subcapitol se prezintă modul în care se apreciază calitatea unui cuantizor vectorial pentru parametri LSF. Cerinţa primordială în cazul cuantizării unor parametri care caracterizează semnalul vocal este ca, semnalul reconstituit pe baza parametrilor cuantizaţi să nu producă erori sesizabile de către un ascultător oarecare, în raport cu semnalul original. În acest scop s-a introdus *eroarea spectrală*, care este o măsură a pătratului diferenţei ariilor dintre cele două spectre, cel al semnalului original şi cel al semnalului reconstituit din parametrii cuantizaţi, având expresia [4]

$$SD_i^2 = \frac{2}{f_e} \int_0^{f_e/2} \left[10 \log_{10}(S_i(f)) - 10 \log_{10}(\hat{S}_i(f)) \right]^2 df \qquad (2.28)$$

unde i reprezintă indicele cadrului de semnal vocal pentru care s-a făcut analiza prin predicţie liniară, iar $S_i(f)$ şi $\hat{S}_i(f)$ sunt spectrele LPC obţinute din parametrii LPC necuantizaţi, respectiv cuantizaţi. Luându-se în considerare eroarea spectrală pentru toate cadrele, se consideră că se obţine o cuantizare "transparentă", adică nu se introduc erori sesizabile, dacă [4],

1. Eroarea spectrală medie este în jurul valorii de 1 dB.
2. Nu există cadre cu erori spectrale mai mari decât 4 dB.
3. Numărul cadrelor cu erori spectrale în gama 2÷4 dB este mai mic decât 2% din totalul cadrelor.

Pentru a se calcula valoarea medie a erorii spectrale, se cuantizează vectorial un grup de vectori LSF, diferiți de vectorii de antrenament, numiți vectori de test.

Parametrii LSF au fost cuantizați la început scalar, adică fiecare componentă din cele 10 s-a cuantizat individual. S-a constatat că sunt necesari între 32 și 36 de biți [7] pentru realizarea unei cuantizări transparente. Adică, fiecare parametru LSF necesită 3÷4 biți sau 8÷16 niveluri de cuantizare în cartea de cod.

Cuantizarea vectorială a fost aplicată la început pentru parametrii LPC sub formă de coeficienți de predicție, și, utilizându-se 10 biți pentru un cadru s-au obținut performanțe comparabile cu cele ale cuantizării scalare la 24 biți/cadru [7]. Eroarea spectrală medie obținută era însă inacceptabilă (3,35 dB), ceea ce a implicat creșterea în continuare a numărului de biți alocat. Dar acest lucru era practic imposibil deoarece atât complexitatea cât și memoria cresc exponențial cu numărul de biți alocat cărții de cod (vezi capitolul 1 al acestei lucrări). În aceste condiții soluția adoptată în literatură a fost folosirea cuantizoarelor vectoriale suboptimale, care permit reducerea complexității și a memoriei, crescând însă eroarea.

O primă metodă de cuantizare vectorială suboptimală folosită pentru parametrii LSF, care face parte din categoria mai mare a metodelor de tip cod produs, este cuantizarea vectorială prin despicare sau partiționare [4],[43]. Conform acestei metode, vectorul de cuantizat se despică într-un număr de subvectori (de regulă 2 sau 3). De exemplu, în cazul despicării în 2 subvectori, primul subvector conține primele n_1 componente (parametri LSF), iar al doilea următoarele $n_2=k-n_1$ componente. Pentru fiecare subvector se proiectează separat câte o carte de cod, iar fiecare subvector se cuantizează separat cu ajutorul cărții de cod aferente. Versiunea cuantizată a vectorului inițial se obține alăturând versiunile cuantizate ale celor 2 subvectori. Această metodă este suboptimală deoarece împărțind vectorii în subvectori, se pierde avantajul datorat dependenței între cei 10 parametri LSF.

Folosindu-se eroarea pătratică ca măsură a erorii de cuantizare, în [4] se obține o cuantizare transparentă pentru o rată de 26 biți/vector (13 biți

pentru primul subvector având n_1=4 componente şi 13 biţi pentru al doilea subvector, cu n_2=6 componente).

Tot în [4] se foloseşte apoi eroarea pătratică ponderată ca măsură a erorii de cuantizare. Aşa cum s-a arătat în capitolul 1, o eroare pătratică ponderată dă o mai mare importanţă unor componente ale vectorilor în raport cu altele, în sensul că cele k diferenţe la pătrat între componentele similare ale celor 2 vectori între care se calculează distanţa, sunt înmulţite cu coeficienţi diferiţi. În acest fel, componentele cu ponderi mari vor introduce termeni mai mari în eroarea totală. De aceea, pentru vectorul de cuantizat, trebuie să se găsească un vector de reproducere, care are componentele cu pondere mare mult mai apropiate faţă de cele similare ale vectorului de cuantizat, astfel micşorându-se termenii respectivi în eroarea totală. Rezultatul este că, componentele cu pondere mare vor fi cuantizate mai precis decât cele cu pondere mai mică. În cazul cuantizării vectoriale a parametrilor LSF, ponderile trebuie alese astfel încât eroarea spectrală *SD* să fie cât mai mică. Asta înseamnă că porţiunile din spectru corespunzătoare vârfurilor sau zonelor cu amplitudine mare trebuie reproduse mai exact în urma cuantizării. De aici a rezultat ideea ca ponderea fiecăreia din cele 10 componente să fie proporţională cu valoarea spectrului la frecvenţa corespunzătoare parametrului LSF respectiv,

$$w_i = \left[S(f_i)\right]^r, \tag{2.29}$$

iar

$$d(f,\hat{f}) = \sum_{i=1}^{10}\left[w_i(f_i - \hat{f}_i)\right]^2, \tag{2.30}$$

unde r este o constantă determinată experimental, având valori în intervalul [0,15; 0,2]. Ponderile w_i, i=1,...,10 au valori variabile de la un cadru la altul, în funcţie de forma spectrului pentru cadrul respectiv. În plus, pentru a accentua şi mai mult parametrii LSF corespunzători frecvenţelor joase se mai introduce o pondere suplimentară care are aceeaşi valoare indiferent de cadru,

$$c_i = \begin{cases} 1 & \text{pentru } 1 \le i \le 8 \\ 0{,}8 & \text{pentru } i = 9 \\ 0{,}4 & \text{pentru } i = 10 \end{cases}, \tag{2.31}$$

astfel încât distanța devine,

$$d(\boldsymbol{f}, \hat{\boldsymbol{f}}) = \sum_{i=1}^{10} \left[c_i w_i (f_i - \hat{f}_i) \right]^2, \tag{2.32}$$

constantele c_i fiind determinate de asemenea experimental.

Prin utilizarea erorii pătratice ponderate, se obține o cuantizare transparentă la 24 biți/vector, sau cu alte cuvinte, la aceeași rată de cuantizare, scade eroarea spectrală medie, precum și numărul cadrelor cu eroare spectrală mai mare decât 2 dB.

Utilizarea ponderii dată de relația (2.29) presupune o mare cantitate de calcul. Pentru a calcula $S(f_i)$ cu relația (2.19), de exemplu, sunt necesare 11 înmulțiri între numere obținute după aplicarea funcției sinus. În plus mai există ridicarea la puterea r (0,15 ... 0,2) și înmulțirea cu c_i. Însă aceste ponderi se calculează o singură dată la cuantizarea unui vector, și așa cum se va vedea în continuare necesită un număr relativ mic de operații față de celelalte operații necesare pentru cuantizarea unui vector.

Considerând că termenii $(c_i w_i)^2$, i=1,...,10 se calculează anterior procesului de calcul al distanțelor față de vectorii de reproducere, expresia complexității este

$$C = \sum_{i=1}^{s} 2 n_i N_i, \tag{2.33}$$

unde s este numărul de subvectori în care se împarte un vector, $n_1+n_2+...+n_s=10$, iar N_i este numărul de vectori de reproducere pentru fiecare subvector. Factorul 2 apare datorită modului în care se calculează termenii din expresia (2.32): $(c_i w_i)^2 \cdot (f_i - \hat{f}_i) \cdot (f_i - \hat{f}_i)$, care necesită 2 înmulțiri. De exemplu, pentru n_1=4, n_2=6, $N_1=N_2=2^{12}$=4096 (rata este 12+12=24 biți/vector), complexitatea are valoarea

$$C=2(4\text{x}4096+6\text{x}4096)=81920 \text{ înmulțiri.}$$

În cazul în care s-ar fi făcut o căutare totală în cartea de cod de 24 biți, complexitatea ar fi avut valoarea

$C'=2x10x2^{24}=3,355x10^8$ înmulţiri,

deci se obţine o reducere foarte mare a numărului de operaţii.

Memoria necesară memorării vectorilor de reproducere este

$$Mem=\sum_{i=1}^{s}2n_iN_i\,, \tag{2.34}$$

unde factorul 2 semnifică faptul că fiecare componentă a vectorilor se memorează prin 2 octeţi. Pentru cazul de mai sus, se obţine

$Mem=2(4x4096+6x4096)=81920$ octeţi.

În cazul unei codări prin căutare totală, memoria necesară are valoarea

$Mem'=2x10x2^{24}=3,355x10^8$ octeţi,

deci în cazul cuantizării prin despicare se obţine şi o reducere a memoriei.

În [15] se propune o nouă pondere, de forma

$$w_i=\frac{1}{f_i-f_{i-1}}+\frac{1}{f_{i+1}-f_i},i=1,\ldots,10\,, \tag{2.35}$$

unde $f_0=0$ *şi* $f_{11}=4000$ *Hz*, iar distanţa între doi vectori se calculează prin

$$d(\boldsymbol{f},\hat{\boldsymbol{f}})=\sum_{i=1}^{10}w_i(f_i-\hat{f}_i)^2\,. \tag{2.36}$$

Expresia (2.35), fără a fi demonstrată, se justifică prin faptul că ponderea la frecvenţa f_i trebuie să fie proporţională cu spectrul la frecvenţa f_i, şi aşa cum s-a arătat în capitolul 2.1, spectrul la frecvenţa f_i este invers proporţional cu diferenţa între f_i şi f_{i-1}, respectiv, f_{i+1} şi f_i. Expresia lui w_i dată prin (2.35) este mai simplă decât cea dată prin (2.29), dar se pare că duce la rezultate ceva mai slabe (în literatură nu se compară performanţele obţinute cu cele două ponderi, pentru aceeaşi bază de date).

O altă metodă de cuantizare vectorială a parametrilor LSF, de asemenea suboptimală şi făcând parte din categoria metodelor de tip cod produs, este cuantizarea vectorială în mai multe stadii (multi stage VQ, MSVQ), [4], [5]. Conform acestei metode, în primul stadiu se face o cuantizare mai "brută" a vectorului de cuantizat (adică se foloseşte o carte de cod cu un număr relativ mic de vectori de reproducere). În fiecare din următoarele stadii se cuantizează vectorul eroare de cuantizare al stadiului precedent, considerat ca diferenţă între $\boldsymbol{x}$ şi $Q(\boldsymbol{x})$. Versiunea cuantizată a

vectorului de intrare se obţine însumând ieşirile cuantizoarelor din fiecare stadiu.

În figura 2.6 se prezintă structura unui cuantizor vectorial în 3 stadii. VQ_1 reprezintă cuantizorul vectorial al primului stadiu. $Q(\boldsymbol{x})=\boldsymbol{y}_{1i}$ reprezintă ieşirea lui VQ_1, $\boldsymbol{y}_{1i}$ fiind unul din cei N_1 vectori de reproducere corespunzători stadiului 1. Vectorul de intrare în stadiul 2 este $\boldsymbol{x}-\boldsymbol{y}_{1i}$, iar $Q(\boldsymbol{x}-\boldsymbol{y}_{1i})=\boldsymbol{y}_{2j}$. În fine, vectorul de intrare în stadiul 3 este $\boldsymbol{x}-\boldsymbol{y}_{1i}-\boldsymbol{y}_{2j}$ şi $Q(\boldsymbol{x}-\boldsymbol{y}_{1i}-\boldsymbol{y}_{2j})=\boldsymbol{y}_{3l}$. Cărţile de cod pentru stadiile 2 şi 3 au N_2 şi respectiv N_3 vectori de reproducere. În acest caz, după cele 3 stadii, cuantizatul vectorului de intrare, $\hat{\boldsymbol{x}}$, este dat prin

$$\hat{\boldsymbol{x}} = \boldsymbol{y}_{1i} + \boldsymbol{y}_{2j} + \boldsymbol{y}_{3l}. \tag{2.37}$$

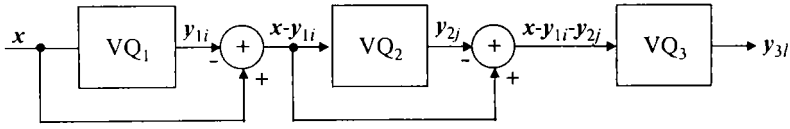


Figura 2.6: *Cuantizor vectorial în 3 stadii.*

Bineînţeles că în practică numărul de stadii poate să fie mai mare decât 3. În faza de proiectare, pentru fiecare stadiu se proiectează separat câte o carte de cod, având ca secvenţă de antrenament secvenţa vectorilor eroare de cuantizare ai stadiului precedent. Acest procedeu este suboptimal pentru că nu ţine cont de stadiile următoare. Şi procedura de codare (căutare) numită secvenţială, (adică codarea se face găsind vectorul de reproducere în primul stadiu, în al doilea ş.a.m.d.) este suboptimală, deoarece nu se face o comparare a vectorului de intrare cu toţi cei N_1 x N_2 x...x N_K vectori de reproducere posibili, unde K reprezintă numărul stadiilor.

Trebuie remarcat că numărul total de biţi prin care se codează un vector, $\log_2 N_1 + \log_2 N_2 + ... + \log_2 N_K$, are o valoare apropiată de numărul de biţi care s-ar obţine în cazul proiectării unei singure cărţi de cod [4], [5], iar în toate stadiile vectorii utilizaţi au aceeaşi dimensiune ca şi vectorii iniţiali, adică k.

O modalitate de a optimiza procesul de căutare la MSVQ, foloseşte un algoritm de căutare mai vechi [1], numit algoritmul M, care se combină cu procesul de căutare secvenţială specific MSVQ [5]. Acest algoritm numit şi MSVQ-M decurge în modul următor: în urma calculării celor N_1 distanţe între vectorul de cuantizat şi vectorii de reproducere ai stadiului 1, se reţin M vectori de reproducere faţă de care se realizează cele mai mici distanţe; în stadiul 2 se calculează distanţele dintre cei M vectori eroare de cuantizare ai stadiului 1, calculaţi ca diferenţă între vectorul de intrare $\boldsymbol{x}$ şi cei M vectori de reproducere care au produs cele mai mici distanţe, şi fiecare din cei N_2 vectori de reproducere ai stadiului 2; din cele MN_2 distanţe, se reţin din nou cele mai mici M. În acest fel se crează M "căi" de parcurgere a cuantizorului, de la stadiul 1 spre stadiul 2. Procesul continuă în acest mod, în fiecare stadiu se reţin M vectori de reproducere care dau cele mai mici distanţe.

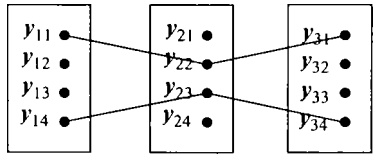


Figura 2.7: *Explicativă pentru algoritmul* MSVQ-M.

În figura 2.7 se arată un caz simplu de MSVQ-M, cu K=3 stadii, M=2 şi N_1=N_2=N_3=4. În stadiul 1, vectorii de reproducere $\boldsymbol{y}_{11}$ şi $\boldsymbol{y}_{14}$ dau cele mai mici 2 distanţe faţă de vectorul de cuantizat $\boldsymbol{x}$. În stadiul 2, din cele 8 distanţe posibile, 4 între $\boldsymbol{x}$-$\boldsymbol{y}_{11}$ şi $\boldsymbol{y}_{21}$,..., $\boldsymbol{y}_{24}$ şi 4 între $\boldsymbol{x}$-$\boldsymbol{y}_{14}$ şi $\boldsymbol{y}_{21}$,..., $\boldsymbol{y}_{24}$, cele mai mici 2 distanţe se obţin între $\boldsymbol{x}$-$\boldsymbol{y}_{11}$ şi $\boldsymbol{y}_{22}$, respectiv între $\boldsymbol{x}$-$\boldsymbol{y}_{14}$ şi $\boldsymbol{y}_{23}$. În stadiul 3, dintre cele 8 distanţe posibile, 4 între $\boldsymbol{x}$-$\boldsymbol{y}_{11}$-$\boldsymbol{y}_{22}$ şi $\boldsymbol{y}_{31}$,..., $\boldsymbol{y}_{34}$ şi 4 între $\boldsymbol{x}$-$\boldsymbol{y}_{14}$-$\boldsymbol{y}_{23}$ şi $\boldsymbol{y}_{31}$, ... $\boldsymbol{y}_{34}$, cele mai mici 2 se obţin între $\boldsymbol{x}$-$\boldsymbol{y}_{11}$-$\boldsymbol{y}_{22}$ şi $\boldsymbol{y}_{31}$, respectiv între $\boldsymbol{x}$-$\boldsymbol{y}_{14}$-$\boldsymbol{y}_{23}$ şi $\boldsymbol{y}_{34}$. Astfel, în acest caz, cele 2 căi care străbat cuantizorul sunt (1,2,1) şi (4,3,4), reprezentând indicii vectorilor de reproducere din cărţile de cod ale stadiilor 1, 2 şi 3. În final, din cele M (2 în cazul de faţă) căi minime se alege calea în raport cu care vectorul de

cuantizat $\boldsymbol{x}$ produce eroarea minimă. Vectorul cuantizat $\hat{\boldsymbol{x}}$ se obţine însumând vectorii de reproducere corespunzători căii minime.

Pe lângă algoritmul MSVQ-M care asigură o optimizare a procesului de căutare, tot în [5], se prezintă o metodă îmbunătăţită de proiectare a cărţilor de cod pentru cele K stadii. Astfel, se proiectează iterativ, cartea de cod pentru fiecare stadiu, începând cu stadiul 1. Pentru a proiecta cartea de cod pentru un stadiu i, i=1, ... ,K, se consideră ca secvenţă de antrenament diferenţa între vectorul $\boldsymbol{x}$ şi vectorii de reproducere corespunzători tuturor stadiilor, inclusiv cele care urmează după stadiul i; vectorii de reproducere ai stadiilor diferite de i se consideră stabiliţi, iar vectorii de reproducere ai stadiului i se optimizează iterativ prin algoritmul Lloyd.

Cumulat, cele două îmbunătăţiri aduse algoritmului MSVQ, adică utilizarea algoritmului M în procesul de căutare şi proiectarea îmbunătăţită a cărţilor de cod pentru cele K stadii, conduc la scăderea erorii spectrale medii, SD. În tabelul 2.3 se prezintă (din [5]), comparativ, valorile erorii spectrale pentru diferite valori ale parametrului M, ale numărului de stadii K, şi ale numărului de vectori de reproducere pentru fiecare stadiu N_1, N_2, ...,N_K. În aceste cazuri, N_1=N_2= ... =N_K=N.

Tabelul 2.3: *Erori spectrale pentru* MSVQ-M.

M	K	N	SD[dB]
1	4	64	1,2
2	4	64	1,08
4	4	64	1,02
1	3	256	1,11
2	3	256	1,01
4	3	256	0,98

Se observă că eroarea spectrală scade odată cu creşterea parametrului M şi, de asemenea scade odată cu scăderea numărului de stadii, respectiv cu creşterea numărului de vectori de reproducere pe stadiu. În ambele variante, cu 4, respectiv cu 3 stadii, numărul total de biţi este 24 (4 stadii x 6

biţi/stadiu; 3 stadii x 8 biţi/stadiu). Cazul $M=1$ este echivalent cu situaţia algoritmului MSVQ necombinat cu algoritmul M.

În continuare se face evaluarea complexităţii şi a memoriei pentru algoritmul MSVQ-M.

Astfel, complexitatea se calculează prin relaţia

$$C = 2k N_1 + 2k M \sum_{i=2}^{K} N_i , \tag{2.38}$$

considerând doar înmulţirile şi eroarea pătratică ponderată ca distanţă între vectori. În plus, algoritmul MSVQ-M necesită mai multe operaţii de comparare.

Memoria folosită se calculează prin

$$Mem = \sum_{i=1}^{K} 2k N_i \tag{2.39}$$

Tabelul 2.4: *Caracteristici pentru cuantizorul vectorial prin despicare.*

s	n_i	N_i	C [înmulţiri]	*Mem* [octeţi]	*SD* [dB]
3	3;3;4	256;256;256	5120	5120	1,17
2	5;5	4096;4096	81920	81920	1,03

Tabelul 2.5: *Caracteristici pentru* MSVQ-M.

K	N	parametrul M	C [înmulţiri]	*Mem* [octeţi]	*SD* [dB]
4	64	1	5120	5120	1,21
		2	8960		1,09
		4	16640		1,04
3	256	1	15360	15360	1,10
		2	25600		1,02
		4	46080		0,95
2	4096	1	163840	163840	0,92
		2	245760		0,90
		4	409600		0,88

În tabelele 2.4 şi 2.5 se prezintă comparativ pentru diferite variante de cuantizoare vectoriale prin despicare şi, respectiv, MSVQ-M, complexitatea, memoria şi eroarea spectrală (valorile erorii spectrale au fost luate din [4] şi [5], iar complexitatea şi memoria au fost calculate de autor conform relaţiilor prezentate). De notat că în toate cazurile rata cuantizoarelor este 24 biţi/vector.

Pentru a compara cele două metode trebuie analizate complexitatea, memoria şi eroarea spectrală, având în vedere că rata este aceeaşi. Deşi cele două metode nu au fost implementate pe aceeaşi bază de date, ceea ce implică doar o comparare orientativă a erorilor spectrale în cele două cazuri, se poate spune că metoda MSVQ-M este superioară pe ansamblu metodei de cuantizare vectorială prin despicare. Această superioritate se datorează în special metodei de proiectare îmbunătăţită a cărţii de cod, deoarece în [4] metoda de cuantizare vectorială prin despicare se compară cu o metodă MSVQ clasică (M=1 şi proiectare a cărţilor de cod separat pentru fiecare stadiu), rezultând că această din urmă metodă este inferioară.

În încheierea analizei comparative a metodelor de cuantizare vectorială prin despicare şi MSVQ-M trebuie remarcat următorul fapt legat de complexitate: dacă s-ar considera şi operaţiile de comparare, valorile din tabelele 2.4 şi 2.5 ar creşte cu cantităţi mai mari pentru MSVQ-M, deoarece această metodă necesită mai multe comparaţii pentru determinarea celor mai mici M distanţe, pe când la cuantizarea vectorială prin despicare se determină numai cea mai mică distanţă.

Pe aceste două metode, cuantizarea vectorială prin despicare şi MSVQ-M (sau MSVQ), se bazează majoritatea celorlalte metode de cuantizare a parametrilor LSF prezente în literatură, metode care încearcă îmbunătăţirea performanţelor: reducerea erorii spectrale, reducerea complexităţii şi a memoriei folosite, protecţia împotriva erorilor care apar la propagarea pe canalul de transmisiune.

În continuare se enumeră pe scurt câteva din ideile desprinse din lucrările din literatură care tratează aspectele prezentate mai sus.

În [16] se prezintă o metodă de adaptare a cărţilor de cod pentru un cuantizor vectorial prin despicare, posibilă datorită proprietăţii de ordonare a componentelor vectorilor LSF. Cuantizarea primului subvector (n_1=3) al

vectorului de intrare prin $(\hat{f}_1, \hat{f}_2, \hat{f}_3)$, impune condiția ca toate elementele cărții de cod pentru subvectorul 2 să verifice condiția $\hat{f}_{4i} > \hat{f}_3, i = 1, \ldots, N_2$, care în mod normal nu poate fi îndeplinită în totalitate. Prin procedura de adaptare descrisă, toți subvectorii din cartea de cod cu N_2 elemente se modifică, astfel încât se restrânge spațiul ocupat de această carte de cod. Se obține o micșorare a regiunilor Voronoi rezultând posibilitatea reducerii erorii de cuantizare. Aceeași adaptare se face și pentru cartea de cod pentru subvectorul 3. În urma experimentărilor rezultă o reducere a numărului de cadre care au erori spectrale între 2 și 4 dB, dar crește complexitatea, datorită adaptării celor două cărți de cod.

O altă metodă adaptivă, aplicată la un MSVQ este prezentată în [17]. Cartea de cod a primului stadiu se adaptează în felul următor: vectorul de reproducere care a fost utilizat cel mai puțin în ultimul timp se elimină din cartea de cod și se înlocuiește cu cuantizatul vectorului curent (suma ieșirilor cuantizoarelor celor două stadii); apoi se trece la cuantizarea vectorului următor. În acest fel se obține o cuantizare transparentă pentru 22 biți/cadru. Complexitatea înregistrează însă o creștere datorită faptului că după fiecare cuantizare, trebuie determinat vectorul de reproducere care a fost folosit mai puțin în ultimul timp.

În [8] se prezintă un cuantizor vectorial de tip latice. O latice reprezintă o mulțime de puncte uniform distribuite într-un spațiu restrâns din $\Re^n$. Un cuantizor vectorial de tip latice are o carte de cod ai cărei vectori de reproducere constituie o latice. În procesul de cuantizare al unui vector, se cuantizează la început prin căutare totală subvectorul format din frecvențele f_3 și f_7 numite puncte "ancoră", deoarece determină intervalele în care se află ceilalți parametri LSF: f_1 și f_2 în intervalele $(0, \hat{f}_3)$, f_4, f_5, f_6 în intervalul $(\hat{f}_3, \hat{f}_7)$, iar f_8, f_9 și f_{10} în intervalul $(\hat{f}_7, 4000)$. Apoi, se definesc în felul următor 3 subvectori, având dimensiunile 2, 3, și, respectiv, 3.

$$\boldsymbol{x}_1 = (x_{11}, x_{12}), \begin{cases} x_{11} = \dfrac{f_1}{\hat{f}_3} \\ x_{12} = \dfrac{\hat{f}_3 - f_2}{\hat{f}_3} \end{cases} \tag{2.40a}$$

$$\boldsymbol{x}_2 = (x_{21}, x_{22}, x_{23}), \begin{cases} x_{21} = \dfrac{f_4 - \hat{f}_3}{\hat{f}_7 - \hat{f}_3} \\ x_{22} = \dfrac{f_5 - \hat{f}_4}{\hat{f}_7 - \hat{f}_3} \\ x_{23} = \dfrac{\hat{f}_7 - f_6}{\hat{f}_7 - \hat{f}_3} \end{cases} \tag{2.40b}$$

$$\boldsymbol{x}_3 = (x_{31}, x_{32}, x_{33}), \begin{cases} x_{31} = \dfrac{f_8 - \hat{f}_7}{4000 - \hat{f}_7} \\ x_{32} = \dfrac{f_9 - \hat{f}_8}{4000 - \hat{f}_7} \\ x_{33} = \dfrac{4000 - f_{10}}{4000 - \hat{f}_7} \end{cases} \tag{2.40c}$$

Se demonstrează că distribuțiile acestor subvectori sunt de forma unui triunghi dreptunghic isoscel, pentru $\boldsymbol{x}_1$, respectiv sub formă de piramidă regulată pentru $\boldsymbol{x}_2$ și $\boldsymbol{x}_3$. Cărțile de cod pentru cei 3 subvectori conțin numere întregi uniform distribuite (latice), astfel încât să acopere zonele ocupate de distribuțiile subvectorilor. Suma acestor numere întregi (2 numere pentru $\boldsymbol{x}_1$, respectiv câte 3 pentru $\boldsymbol{x}_2$ și $\boldsymbol{x}_3$) este sau pară, sau impară, fiind mai mică sau egală cu un număr fixat inițial. De exemplu, pentru vectorul $\boldsymbol{x}_1$, se arată în tabelul 2.6 vectorii de reproducere pentru cazul când suma componentelor este pară, mai mică sau egală cu 6. În total, sunt 16 vectori, care acoperă în plan un triunghi dreptunghic isoscel.

Înainte de cuantizare, subvectorii $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ și $\boldsymbol{x}_3$ se înmulțesc cu constante alese convenabil pe baza secvenței de antrenament, astfel încât să se încadreze în zonele acoperite de cartea de cod. Evident că acești subvectori vor avea după înmulțire componente neîntregi. Găsirea vectorului de reproducere se face în acest caz foarte simplu, necesitând doar două înmulțiri cu matrici și rotunjirea la valorile întregi cele mai apropiate (procedeul descris pe larg în [8]).

Tabelul 2.6: *Vectorii de reproducere* (x_{11}, x_{12}) *pentru un cuantizor vectorial de tip latice.*

0,6	0,4	0,2	0,0
1,5	1,3	1,1	
2,4	2,2	2,0	
3,3	3,1		
4,2	4,0		
5,1			
6,0			

Această metodă are o complexitate foarte mică, nu necesită practic memorie decât doar pentru cartea de cod a subvectorului (f_3, f_7), dar cuantizarea transparentă se obține pentru o rată mai mare, de 28 biți/vector.

În [9] se prezintă un concept nou de abordare a cuantizării vectoriale. Ideea de bază este reprezentată de existența unei relații între codul unui vector de reproducere (reprezentarea binară care se transmite la decodor) și valoarea sa (cele k componente),

$$y_j = T b_j , \tag{2.41}$$

unde y_j, j=0, ...,N-1 reprezintă vectorii de reproducere, b_j reprezintă codurile corespunzătoare, iar T este o matrice. Codul b_j are forma, numită cod bloc

$$b_j = (+1, i_j(1), i_j(2), \ldots, i_j(k), c_j(1), \ldots, c_j(r))^T , \tag{2.42}$$

preluată din domeniul teoriei codării. Biții $i_j(l)$, l=1, ... ,k sunt biții de informație care corespund reprezentării binare a lui j, în care simbolul 0 se înlocuiește cu 1, iar 1 cu -1, iar $c_j(l)$, l=1, ... ,r sunt biții de paritate care se obțin prin produse între biții de informație. Deoarece lungimea codului b_j este $k+r+1$, matricea T are dimensiunea $(k+r+1)$xk. Pentru un cuantizor care folosește o relație de forma (2.41), în procesul de proiectare trebuie determinată matricea T și de asemenea trebuie specificat modul cum se obțin biții de paritate.

Avantajul esențial al acestei metode este, așa cum se demonstrează pe larg în [9], robustețea față de erorile care apar la propagarea pe canal.

Memoria folosită este cu atât mai mică cu cât numărul r de biţi de paritate este mai mic, deoarece trebuie memorată matricea T, pe baza căreia se calculează vectorii de reproducere. Calcularea vectorilor de reproducere, pe de altă parte produce creşterea complexităţii, însă având în vedere că în codul b_j sunt numai 1 şi -1, înmulţirea dată de (2.41) se transformă în adunări şi scăderi.

Pe baza celor prezentate despre codul bloc, atât proiectarea cât şi căutarea celui mai apropiat vector de reproducere se fac având la bază metodele de cuantizare vectorială prin despicare şi MSVQ. Ca element de noutate, la cuantizarea vectorială prin despicare (n_1=5, n_2=5) se reţin L subvectori de reproducere care produc cele mai mici erori L din cele 2 cărţi de cod, alegându-se vectorul care produce cea mai mică eroare în raport cu vectorul de cuantizat, din cele L^2 posibilităţi.

Având în vedere succinta trecere în revistă a metodelor de cuantizare vectorială a parametrilor LSF, autorul se va orienta înspre găsirea unei metode de cuantizare vectorială rapidă (de complexitate redusă) care însă să nu sacrifice performanţele, adică să nu producă creşterea erorii spectrale.

În capitolul 2, s-a prezentat reprezentarea unui semnal vocal prin parametri LSF, arătându-se unele contribuţii ale autorului în acest sens. De asemenea sunt prezentate şi principalele metode de cuantizare vectorială ale acestor parametrii.

3. PROIECTAREA CARȚII DE COD A UNUI CUANTIZOR VECTORIAL

3.1 Generalități

Proiectarea cărții de cod a unui cuantizor vectorial, adică determinarea vectorilor de reproducere, se face anterior procesului de cuantizare propriuzisă (codare și decodare), pe baza unei secvențe de antrenament, care cuprinde un mare număr de vectori. Vectorii de reproducere trebuie să fie reprezentativi pentru informația conținută în secvența de antrenament. Apoi, cuantizarea se aplică pentru alți vectori, diferiți de cei de antrenament. Cu cât secvența de antrenament conține un număr mai mare de vectori, cu caracteristici mai diferite, cu atât vectorii de cuantizat vor fi "aproximați" mai exact cu ajutorul vectorilor de reproducere din cartea de cod.

În cele mai multe cazuri, deci și pentru cuantizarea vectorială a parametrilor LSF, cartea de cod se proiectează cu ajutorul algoritmului generalizat al lui Lloyd, prezentat în capitolul 1. O problemă importantă o reprezintă alegerea cărții de cod inițiale, care urmează să fie îmbunătățită. O posibilitate ar fi folosirea ca și carte de cod inițială a unei cărți aleatoare care să conțină N vectori de antrenament, aleși la întâmplare. Altă posibilitate ar fi alegerea a N vectori egal depărtați unul de altul în secvența de antrenament, adică $\boldsymbol{x}_1, \boldsymbol{x}_{r+1}, \boldsymbol{x}_{2r+1}, \ldots, \boldsymbol{x}_{(N-1)r+1}$, unde $r = n/N$, n fiind lungimea secvenței de antrenament. În [4] și [11] se folosește o metodă care aplică algoritmul Lloyd succesiv pentru cărți de cod având 2,4,8,...,N vectori de reproducere. Astfel, mai întâi se determină centroidul întregii secvențe de antrenament, care apoi se "despică" în doi vectori (unul dintre acești vectori se obține prin modificarea cu o cantitate mică a fiecărei componente a centroidului, iar celălalt poate fi chiar centroidul). În continuare, de fiecare dată, dublarea numărului de vectori de reproducere se face prin astfel de despicări, apoi se optimizează cartea de cod cu acel număr de vectori de reproducere, până la obținerea numărului dorit de vectori de reproducere, N.

(A nu se confunda această metodă de proiectare a cărții de cod în care se fac "despicări", adică dublări ale numărului de vectori, cu metoda de codare prin despicare, în care un vector se împarte în subvectori de dimensiune mai mică în vederea reducerii complexității și memoriei în procesul de codare).

În [14], Equitz prezintă o metodă de proiectare a unei cărți de cod, în care vectorii de antrenament se grupează în *mănunchiuri*, fiecare mănunchi conținând vectori cât mai apropiați din spațiul k-dimensional, centroizii acestor mănunchiuri reprezentând în final vectorii de reproducere. În principiu acest algoritm se desfășoară în felul următor: la început, fiecare vector de antrenament reprezintă un mănunchi, deci există n mănunchiuri; apoi se unesc împreună cei mai apropiați doi vectori de antrenament, într-un singur mănunchi, astfel numărul de mănunchiuri devenind n-1. Procesul continuă în acest mod, de fiecare dată se unesc cele mai apropiate două mănunchiuri, până când numărul de mănunchiuri devine egal cu N, mărimea dorită a cărții de cod. Acest procedeu numit PNN (pairwise nearest neighbor - împerecherea inteligentă a celor mai apropiați vectori), a fost aplicat în [14] pentru cuantizarea vectorială a imaginilor, și are avantajul unui timp de realizare mult mai scurt decât algoritmul Lloyd, fără a crește eroarea de cuantizare.

Autorul aplică algoritmul PNN pentru proiectarea cărții de cod pentru vectori cu parametri LSF, adaptându-l în funcție de particularitățile acestui tip de vectori.

3.2 Utilizarea algoritmului PNN pentru cuantizarea vectorială a parametrilor LSF

Varianta principială a algoritmului prezentată anterior necesită un volum mare de calcul, deoarece secvența de antrenament conține foarte mulți vectori. În [14] Equitz propune o metodă rapidă de efectuare a acestui algoritm, bazată pe următoarea idee: căutarea în vederea găsirii celor mai apropiate două mănunchiuri să nu se facă în întreaga secvență de antrenament, ci în zone mai mici care cuprind vectori apropiați; pentru

aceasta, mai întâi se împarte întreaga secvență de antrenament în zone care conțin cel mult un număr prestabilit de vectori, b, apropiați în spațiu, aceste zone numindu-se *buchete*; apoi, în fiecare buchet se găsesc cele mai apropiate două mănunchiuri (la început fiecare mănunchi conține un vector); în final, doar într-o fracțiune, de obicei jumătate, din aceste buchete perechea de mănunchiuri mai apropiate se unește într-un singur mănunchi; buchetele în care se fac uniri sunt acelea în care distanțele între cele două mănunchiuri mai apropiate sunt cele mai mici; apoi, din nou se inspectează buchetele în vederea găsirii celor mai apropiate două mănunchiuri, urmată de unirea mănunchiurilor, până când numărul de mănunchiuri devine egal cu N.

3.2.1 Împățirea secvenței de antrenament în buchete

Pentru proiectarea cărții de cod autorul a folosit o secvență de antrenament conținând n=2790 vectori cu parametri LSF, având dimensiunea k=10, proveniți de la un singur vorbitor. După captarea cu ajutorul unui microfon, semnalul vocal a fost amplificat prin intermediul unui amplificator de tip MT 60, având amplificarea de 60 dB. Apoi s-a făcut achiziția semnalului prin intermediul plăcii de tip DS 1101 conținând un procesor de semnal TMS 320 E14. Achiziția s-a făcut cu frecvența de eșantionare f_e=8 kHz, fiind urmată de o filtrare numerică de tip trece jos, la 4 kHz și o analiză prin predicție liniară de ordinul 10, efectuată prin metoda autocorelației. Durata unui cadru de semnal pentru care s-a făcut predicția liniară a fost de 25 ms, conținând 200 de eșantioane. Programul pentru dialogul între calculatorul gazdă și sistemul DS 1101 în vederea achiziției, ca și cel pentru predicția liniară au fost scrise în limbajul C. Pe baza vectorilor cu coeficienți de predicție, s-au obținut vectorii cu parametri LSF, pe baza unui program în MATLAB.

Proiectarea cărții de cod s-a făcut în vederea realizării unui cuantizor vectorial de tipul prin despicare. Vectorii au fost despicați în 3 subvectori, având dimensiunile 3, 3 și, respectiv, 4. Astfel, pentru fiecare din cei 3 subvectori s-a proiectat câte o carte de cod, conținând câte 256 vectori de reproducere, adică rata cuantizorului a fost de 8x3=24 biți/vector. În continuare se descrie proiectarea completă a cărții de cod pentru un singur

subvector, prezentându-se la sfârşit rezultatele obţinute pentru toţi cei 3 subvectori.

Aşa cum s-a amintit în descrierea făcută în introducerea acestui paragraf, mai întâi s-a făcut o împărţire a secvenţei de antrenament în buchete. Împărţirea s-a făcut separat pentru cele 3 secvenţe de antrenament rezultate prin considerarea componentelor de la 1 la 3, de la 4 la 6, şi respectiv de la 7 la 10, ale secvenţei iniţiale cu dimensiunea 10. Împărţirea în buchete s-a făcut conform arborilor k-dimensionali [14]. Un astfel de arbore conţine mai multe niveluri (figura 3.1). Primul nivel conţine un singur nod, numit rădăcină, în care vectorii corespunzători secvenţei de antrenament se împart conform unui anumit criteriu în 2 subsecvenţe distincte. Aceste 2 subsecvenţe devin secvenţele asociate celor 2 noduri de pe nivelul 2, unde la rândul lor, cele două secvenţe se împart, rezultând 4 subsecvenţe. Procesul continuă în acest mod. Subsecvenţele care au un număr de vectori mai mic sau egal cu un număr prestabilit, notat b, devin buchete, sau noduri terminale, împărţirea continuând pentru celelalte subsecvenţe, până când toate se vor transforma în buchete.

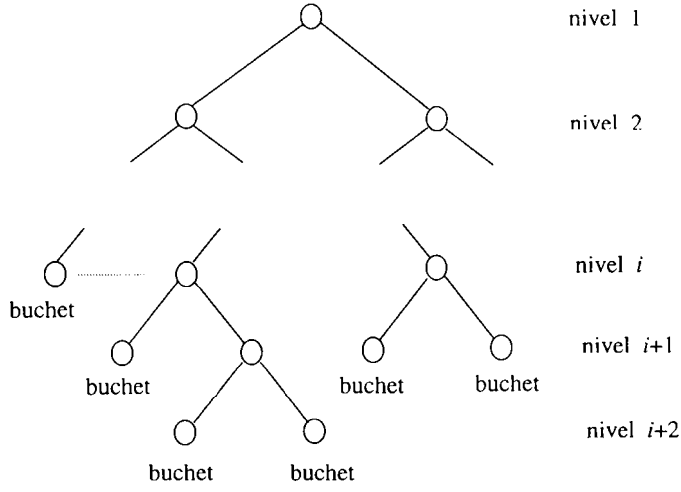


Figura 3.1: *Structura unui arbore pentru împărţirea în buchete.*

Conform celor expuse este evident că odată cu apariţia buchetelor, numărul de noduri de pe un nivel nu se va mai dubla faţă de nivelul

precedent, astfel încât arborele va avea o structură neregulată, cu ramuri inegale.

Foarte important este criteriul după care se face împărţirea secvenţelor în subsecvenţe în fiecare din noduri. Criteriul ales de autor, acelaşi ca în [14], cuprinde următoarele etape:

- mai întâi, pentru fiecare componentă a vectorilor asociaţi unui nod se calculează dispersia conform relaţiei

$$\sigma_i^2 = \frac{\sum_{j=1}^{n'} (x_{ij} - m_i)^2}{n'}, i = 1,\ldots,k, \tag{3.1}$$

unde k reprezintă dimensiunea subvectorilor (3, 3 sau 4), x_{ij} reprezintă componenta i a vectorului j, m_i este valoarea medie a componentei i,

$$m_i = \frac{\sum_{j=1}^{n'} x_{ij}}{n'}, i = 1,\ldots,k, \tag{3.2}$$

iar n' reprezintă numărul de vectori de antrenament asociaţi nodului respectiv.

- în continuare se determină care din cele k componente are cea mai mare dispersie; fie c această componentă, $c \in \{1,\ldots,k\}$, valoarea medie m_c a componentei respective reprezintă valoarea de prag asociată nodului respectiv.

-în vederea împărţirii secvenţei asociate unui nod în două subsecvenţe distincte, vectorii a căror componentă x_{cj} este mai mică decât m_c se asociază primei subsecvenţe iar cei cu componenta x_{cj} mai mare sau egală cu m_c se asociază celei de-a doua subsecvenţe ($j = 1,\ldots,n'$). (Se notează cu n_{s1} şi n_{s2} numărul de vectori din cele 2 subsecvenţe, $n_{s1}+n_{s2} = n'$). De remarcat că valorile de prag m_c şi componentele c asociate nodurilor, pot să difere de la nod la nod, chiar dacă nodurile sunt pe acelaşi nivel.

În acest caz autorul a ales numărul maxim de vectori conţinuţi de un buchet, b=10. Programul care face împărţirea secvenţei de antrenament în buchete a fost scris în limbajul C. În figura 3.2 se prezintă organigrama acestui program, care este valabilă evident pentru toţi cei 3 subvectori în care s-a împărţit vectorul cu dimensiunea k=10.

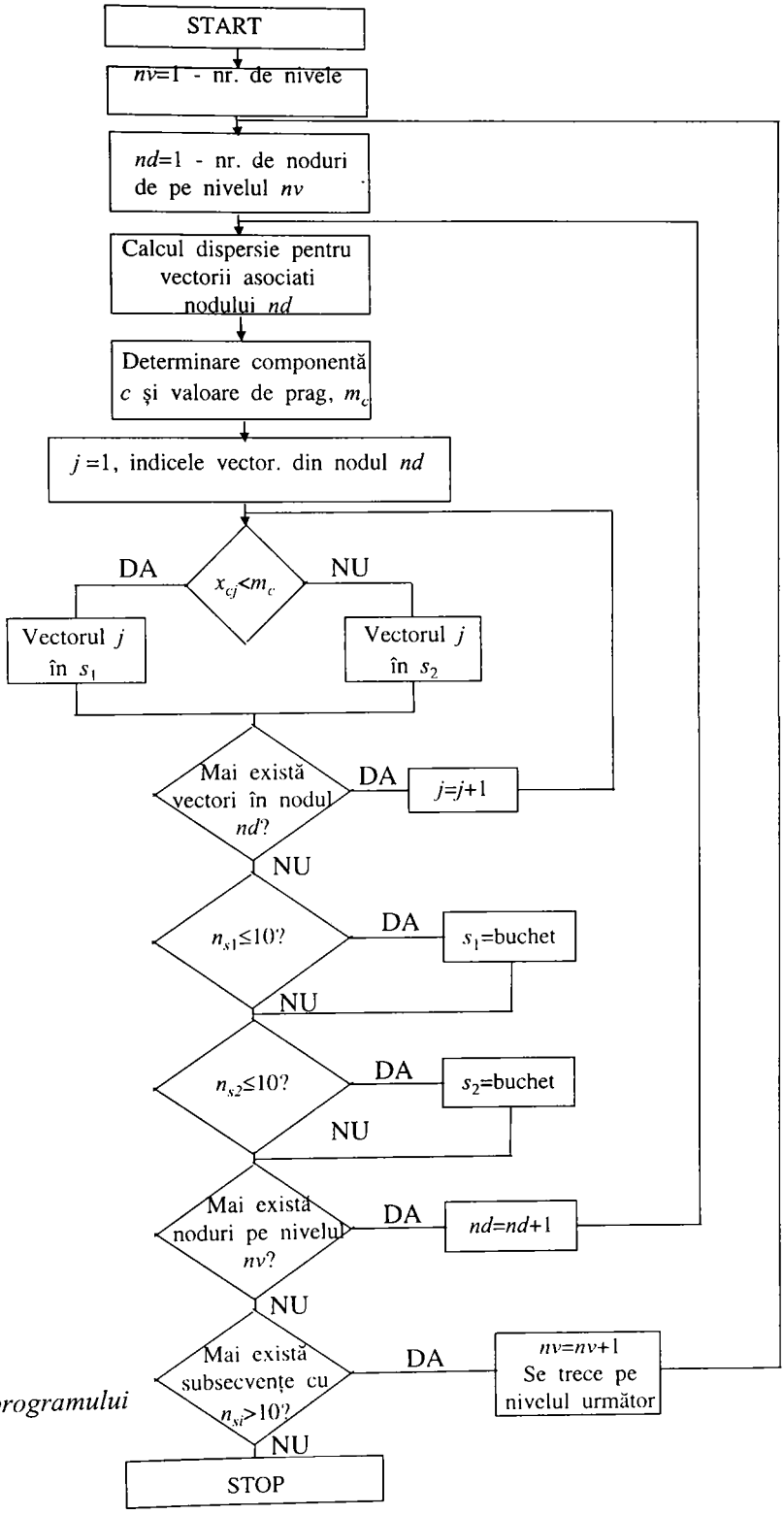


Figura 3.2: *Organigrama programului de împărţire a secvenţei de antrenament în buchete.*

Referitor la programul a cărui organigramă s-a prezentat anterior, se impun următoarele observaţii:

- vectorii de antrenament asociaţi nodului rădăcină (întreaga secvenţă de antrenament) se depun într-o memorie, M_1.

- pentru fiecare nod al arborelui, vectorii asociaţi se împart în două subsecvenţe, notate s_1 şi s_2, conform criteriului prezentat; dacă o subsecvenţă are 10 vectori sau mai puţini, ea devine buchet, şi vectorii corespunzători se depun într-o memorie, notată M_2, iar dacă are mai mult de 10 vectori va constitui o secvenţă asociată cu un nod de pe nivelul următor; în acest caz vectorii corespunzători se depun în altă memorie notată M_3.

- după parcurgerea tuturor nodurilor de pe un nivel, vectorii corespunzători subsecvenţelor care nu au devenit buchete, vor constitui vectorii secvenţelor asociate nodurilor de pe următorul nivel, şi se vor transfera din memoria M_3 în memoria M_1, peste vectorii care au costituit secvenţele asociate nodurilor de pe nivelul curent, care nu mai sunt necesari; în memoria M_2, după fiecare nivel se vor acumula vectorii componenţi ai buchetelor.

În acest fel autorul a asigurat o bună organizare a datelor în memorie, în scopul folosirii unei memorii cât mai mici. Astfel, fără efectuarea acestei alocări dinamice a memoriei, zona de memorie ocupată de M_1 şi M_3, ar fi necesară pentru fiecare nivel în parte. Pentru un nivel sunt necesari 2x2790x4x2≅45000 octeţi. (S-a considerat cazul cel mai defavorabil, pentru dimensiunea k=4 a vectorilor, fiecare componentă fiind memorată prin 2 octeţi). Deoarece arborele a avut 8...10 niveluri, rezultă că ar fi fost necesară o memorie de circa 400 kocteţi.

În tabelul 3.1 se prezintă, separat, pentru subvectorii care conţin parametrii LSF de la 1 la 3, de la 4 la 6, respectiv de la 7 la 10, numărul de subsecvenţe şi numărul de buchete rezultate pentru fiecare nivel, precum şi numărul total de buchete obţinute, N_B.

Numărul de vectori pe buchete obţinut a fost cuprins între 4 şi 10, rezultând o medie de aproximativ 7 vectori pe buchet, aşa cum rezultă din tabelul 3.1.

Privind blocurile din organigrama programului de împărţire a secvenţei de antrenament în buchete, se constată că se fac înmulţiri doar la

determinarea dispersiei. In rest se fac doar comparări sau transferuri de date între diferite zone de memorie. O expresie analitică a complexității nu se poate obține, deoarece de la un anumit nivel încep să apară buchetele, ceea ce reduce numărul de vectori din secvențele asociate nodurilor, neputându-se estima pe fiecare nivel câte buchete apar. Din tabelul 3.1, rezultă că

Tabelul 3.1: *Rezultatele algoritmului de împărțire a secvenței de antrenament în buchete.*

Subvectorul 1, 2, 3,			Subvectorul 4, 5, 6,		Subvectorul 7, 8, 9,10	
Nivelul	Subsec-vențe	Buchete	Subsecvențe	Buchete	Subsecvențe	Buchete
1	2	0	2	0	2	0
2	4	0	4	0	4	0
3	8	0	8	0	8	0
4	16	0	16	0	16	0
5	32	0	32	0	32	0
6	63	1	64	0	64	0
7	119	7	123	5	121	7
8	140	98	139	107	119	123
9	7	273	17	261	38	200
10	0	14	0	34	4	72
11	-	-	-	-	0	8
N_B		393		407		410

începând cu nivelurile 6 sau 7 apar primele buchete, pe nivelul 8 încă se mai împarte un număr considerabil de secvențe, în timp ce pe ultimele niveluri numărul buchetelor crește brusc, astfel scăzând numărul de secvențe care se împart. Pentru a determina o limită superioară a complexității se consideră acoperitor că pe primele 8 niveluri nu se obțin buchete (adică subsecvențele au mai mult de 10 vectori), ceea ce înseamnă că pe fiecare din aceste niveluri se calculează dispersia pentru 2790 vectori de antrenament (pentru 3 componente). În acest fel complexitatea este

C=2790 x 3 x 8= 66960 înmulțiri

Desigur că această complexitate crește odată cu lungimea secvenței de antrenament, când în plus va crește și numărul de niveluri.

Pentru simplitate, complexitatea s-a calculat doar pentru subvectorii cu componentele 1, 2 și 3. In mod similar complexitatea se poate calcula și pentru subvectorii cu componentele 4, 5 și 6, și respectiv pentru cei cu componentele 7, 8, 9, 10, fiind proporțională cu dimensiunea k.

3.2.2 Algoritmul PNN rapid. Obținerea vectorilor de reproducere

Împărțirea secvenței de antrenament în regiuni mici de vectori sau buchete, permite o căutare mai rapidă în vederea grupării vectorilor în mănunchiuri. Două mănunchiuri se unesc într-unul singur, dacă după unire introduc cea mai mică eroare de cuantizare, dintre toate perechile de mănunchiuri posibile. Situația este mai simplă la început, când fiecare mănunchi conține un singur vector. Odată cu prima unire de mănunchiuri, există un mănunchi cu 2 vectori, iar în continuare după alte uniri, numărul de vectori pe mănunchi crește. În [14] se deduce expresia erorii de cuantizare a vectorilor din două mănunchiuri C_i și C_j care se unesc, dacă acești vectori sunt cuantizați prin centroidul mănunchiului rezultat după unire, notat C_{ij} astfel

$$E_{ij} = \sum_{x \in C_{ij}} \left| \boldsymbol{x} - \bar{\boldsymbol{x}}_{ij} \right|^2 = E_i + E_j + \frac{n_i n_j}{n_i + n_j} \left| \bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}_j \right|^2, \tag{3.3}$$

unde operatorul $||^2$ reprezintă norma euclidiană a vectorului, n_i și n_j reprezintă numărul vectorilor din cele două mănunchiuri C_i și C_j, $\bar{\boldsymbol{x}}_i$ și $\bar{\boldsymbol{x}}_j$ reprezintă centroizii (media) vectorilor din mănunchiurile C_i și C_j, $\bar{\boldsymbol{x}}_{ij}$ reprezintă centroidul vectorilor din mănunchiul C_{ij},

$$\bar{\boldsymbol{x}}_{ij} = \frac{n_i \bar{\boldsymbol{x}}_i + n_j \bar{\boldsymbol{x}}_j}{n_i + n_j}, \tag{3.4}$$

iar E_i și E_j erorile corespunzătoare mănunchiurilor C_i și C_j,

$$E_i = \sum_{x \in C_i} \left| \boldsymbol{x} - \bar{\boldsymbol{x}}_i \right|^2. \tag{3.5}$$

Din relaţia (3.3) se observă că pentru a calcula eroarea E_{ij} nu este necesară cunoaşterea explicită a vectorilor care compun mănunchiurile care se unesc, ci numai a numărului lor, a centroizilor lor, precum şi a erorilor de cuantizare care se fac la cuantizarea vectorilor componenţi prin centroidul fiecărui mănunchi.

Pentru o mai clară înţelegere a modului în care se alege perechea de mănunchiuri care se va uni, se consideră un caz simplu. Se presupune că există 3 mănunchiuri, cu n_1, n_2, n_3 vectori fiecare, iar erorile de cuantizare ale vectorilor din fiecare mănunchi prin centroidul mănunchiului sunt E_1, E_2, E_3. Aplicând relaţia (3.3), erorile totale de cuantizare ale vectorilor din cele 3 mănunchiuri, în urma celor 3 uniri posibile sunt:

- când se unesc mănunchiurile 1 şi 2,

$$E = E_{12} + E_3 = E_1 + E_2 + \frac{n_1 n_2}{n_1 + n_2}\left|\bar{x}_1 - \bar{x}_2\right|^2 + E_3, \tag{3.6}$$

- când se unesc mănunchiurile 1 şi 3,

$$E = E_{13} + E_2 = E_1 + E_3 + \frac{n_1 n_3}{n_1 + n_3}\left|\bar{x}_1 - \bar{x}_3\right|^2 + E_2, \tag{3.7}$$

- când se unesc mănunchiurile 2 şi 3,

$$E = E_1 + E_{23} = E_1 + E_2 + E_3 + \frac{n_2 n_3}{n_2 + n_3}\left|\bar{x}_2 - \bar{x}_3\right|^2. \tag{3.8}$$

Privind relaţiile de mai sus, se constată că termenul $E_1+E_2+E_3$ apare în toate cele 3 cazuri, ceea ce înseamnă că se vor uni mănunchiurile pentru care termenul $\frac{n_i n_j}{n_i + n_j}\left|\bar{x}_i - \bar{x}_j\right|^2$ este minim.

În acest fel, la începutul algoritmului PNN rapid, în fiecare buchet se găseşte perechea de mănunchiuri care produce eroarea minimă prin unire, numită "candidata" buchetului respectiv. Apoi într-o fracţiune din buchete (de exemplu în 50%) se fac efectiv unirile, şi anume, în acele buchete în care creşterile erorii prin unire, (adică $\frac{n_i n_j}{n_i + n_j}\left|\bar{x}_i - \bar{x}_j\right|^2$) sunt cele mai mici. Motivul pentru care nu se fac uniri în toate buchetele este următorul: există buchete care conţin mănunchiuri îndepărtate, prin a căror unire eroarea totală ar creşte mult; de aceea, deoarece în buchetele în care se fac uniri se

modifică parametrii n_i şi $\bar{x}_i$, se reia procesul de căutare a candidatelor în fiecare buchet, şi astfel se pot face uniri şi în buchete în care anterior s-au mai făcut, dar care produc creşteri mai mici ale erorii totale. Algoritmul PNN rapid continuă în acest fel, reluându-se procesul de căutare a candidatelor şi efectuându-se uniri. Astfel, numărul total de mănunchiuri existente în toţi bucheţii, notat N_{TM}, va scădea, algoritmul încheindu-se când numărul N_{TM} va fi egal cu mărimea dorită a cărţii de cod, N. Vectorii de reproducere vor fi centroizii mănunchiurilor obţinute.

O problemă deosebit de importantă, nediscutată în [14] este dată de raportul între mărimea dorită a cărţii de cod N, care uzual poate avea valorile 256, 512, 1024, 2048 sau 4096 şi valoarea numărului de buchete N_B obţinut. În cazul de faţă s-a văzut că N_B are valori în jurul lui 400, pentru cei 3 subvectori. Deoarece N=256 (valoare identică pentru toate cele 3 cărţi de cod, la cuantizorul prin despicare), se observă că $N<N_B$, ceea ce ar însemna că, dacă s-ar face uniri până când fiecare buchet va rămâne cu un mănunchi, numărul de vectori de reproducere posibili ar fi egal cu N_B şi nu s-ar mai putea reduce. În plus, pentru a se ajunge în această situaţie s-ar încălca principiul de a se face uniri doar în buchetele în care unirile provoacă creşteri mai mici ale erorii de cuantizare. O primă soluţie ar fi scăderea numărului de buchete, impunând ca b, numărul maxim de vectori pentru un buchet să aibe valori mai mari. Aceasta ar duce însă la creşterea volumului de calcul necesar pentru găsirea candidatei în fiecare buchet (care este proporţional cu C_b^2). **Autorul propune în acest caz o variantă modificată a algoritmului PNN rapid:** se fac uniri de fiecare dată doar în jumătate dintre buchete; după un număr de astfel de iteraţii, în unele buchete, care conţin vectori mai apropiaţi, se fac mai multe uniri între mănunchiuri decât în altele, numărul de mănunchiuri în aceste buchete scăzând; fiecare astfel de buchet, în situaţia când ajunge să conţină 2 mănunchiuri, se uneşte cu cel mai apropiat buchet, determinat pe baza distanţei dintre centroizii generali ai buchetelor (media vectorilor din buchetul respectiv); astfel, numărul de buchete scade, crescând numărul de mănunchiuri în buchetele care se unesc; procesul continuă până când numărul total de mănunchiuri din toate buchetele, N_{TM}, devine egal cu N.

În figura 3.3 se prezintă organigrama programului care implementează algoritmul PNN rapid modificat, scris în limbajul C.

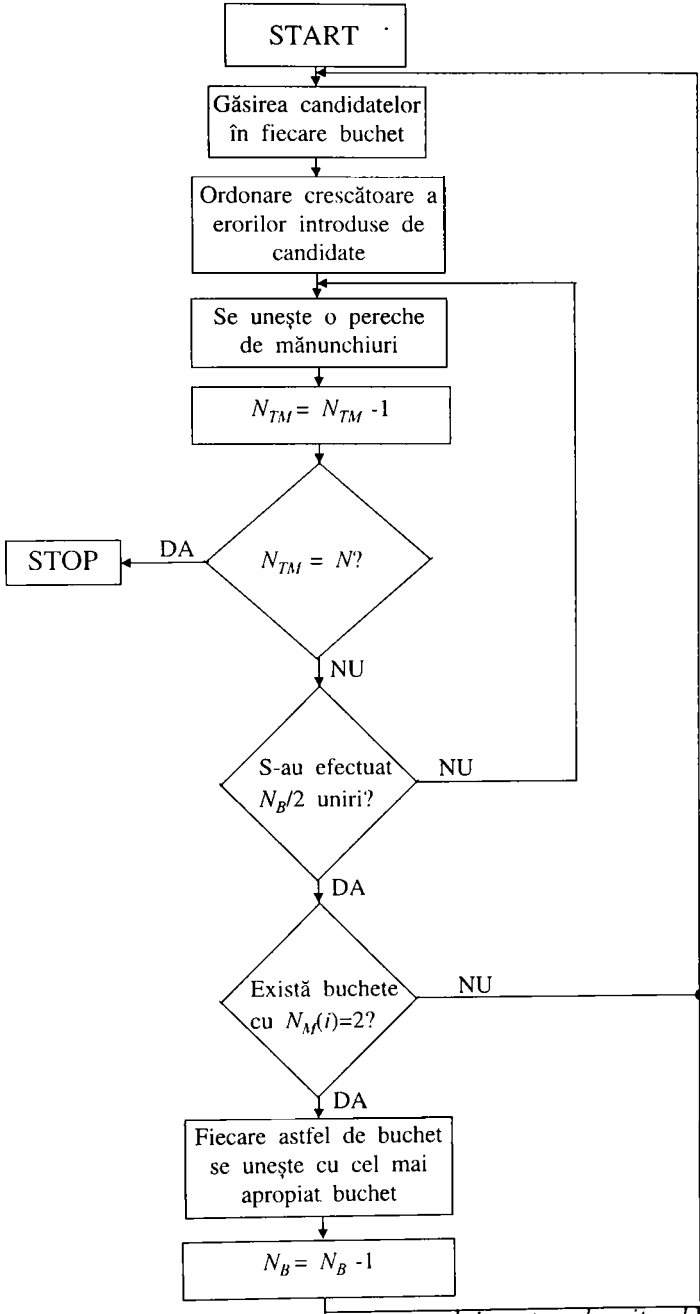


Figura 3.3: *Organigrama programului pentru algoritmul* PNN.

În legătură cu algoritmul PNN rapid descris prin organigrama din figura 3.3 (programul complet este prezentat în Anexa 2), se impun următoarele observații:

- la început, fiecare din cei n vectori de antrenament constituie câte un mănunchi, deci numărul total de mănunchiuri este $N_{TM}=n$.

- găsirea perechii de mănunchiuri candidată pentru fiecare buchet, presupune calculul ultimului termen al relației (3.3) pentru fiecare pereche de mănunchiuri.

- pentru a se face uniri între mănunchiuri doar în jumătate din numărul de buchete, se face o ordonare crescătoare a erorilor introduse de fiecare dintre perechile candidate, făcându-se uniri în primele $N_B/2$ buchete din tabelul obținut în urma ordonării.

- pentru un buchet i, care are numărul de mănunchiuri $N_M(i)=2$, se calculează distanța de la centroidul său la centroizii tuturor celorlalte buchete, în scopul găsirii celui mai apropiat buchet.

- unind două buchete, scade numărul de buchete, N_B, ceea ce înseamnă că scade numărul de uniri între mănunchiuri ($N_B/2$) care se vor face la următoarea parcurgere a buchetelor; iar tot prin unirea a două buchete crește numărul de mănunchiuri $N_M(i)$ al buchetului rezultant.

În urma rulării programului, separat pentru cele 3 structuri de buchete obținute pentru fiecare din cei 3 subvectori s-au obținut următoarele rezultate:

- în toate cele 3 cazuri, bucla mai mare a programului, cea în care se face determinarea candidatelor, urmată de uniri între mănunchiuri și eventual între buchete, se parcurge de 20 de ori.

- de asemenea, după 4 astfel de bucle, apar primele buchete cu $N_M(i)=2$, deci se fac uniri între buchete.

- în final, rămân 65, 62, și, respectiv 68 de buchete, pentru cele 3 tipuri de subvectori; fiecare din cele 3 grupe de buchete conțin în total 256 de mănunchiuri, ale căror centroizi reprezintă vectorii de reproducere.

Pe baza rezultatelor obținute la rularea programului se face o evaluare a complexității, considerând operațiile de înmulțire. Referitor la organigrama din figura 3.3 se identifică blocurile în care se fac înmulțiri. Aceste blocuri sunt:

- a) blocul pentru găsirea candidatelor în fiecare buchet, unde ultimul termen al relației 3.3 se calculează pentru fiecare pereche de două mănunchiuri a buchetului cercetat.

- b) blocul în care se face unirea a 2 mănunchiuri, unde se aplică relația 3.4 pentru determinarea centroidului mănunchiului rezultant.

- c) blocul în care fiecare buchet cu numărul de mănunchiuri $N_M(i)=2$ se unește cu cel mai apropiat buchet, în care în primul rând se calculează distanțele de la buchetul respectiv la toate buchetele existente, și în al doilea rând, trebuie determinat centroidul general al buchetului rezultat prin unire, pe baza relației (3.4).

Evaluarea complexității în cazul determinării candidatelor în buchete, este ceva mai dificilă deoarece numărul de mănunchiuri este în continuă scădere și de aceea se va face o evaluare aproximativă. Astfel, la primele 3 parcurgeri ale buclei mari, numărul de buchete este 393, iar numărul de mănunchiuri este succesiv 2790, 2590, 2390, ceea ce înseamnă că fiecare buchet are în medie aproximativ 7, apoi 6,5, iar la a 3-a parcurgere a buclei, 6 mănunchiuri. Aceasta înseamnă că ultimul termen din relația 3.3. se calculează de aproximativ $393(C_7^2 + C_7^2 + C_6^2) = 22400$ ori. În următoarele 17 parcurgeri ale buclei, numărul de buchete scade de la 393 la 65, iar numărul de mănunchiuri de la 2190 la 256. Se poate considera cu aproximație că fiecare buchet are în medie 5 mănunchiuri. Știind că s-au făcut în acest timp 2190 - 256 = 1934 uniri, înseamnă că s-au cercetat un număr dublu de buchete, adică 3868. Rezultă că ultimul termen al relației (3.3) s-a calculat de 3868 x C_5^2 = 38680 ori. Rezultă că în total, în cele 20 de parcurgeri ale buclei, acest termen s-a calculat de aproximativ 61000 ori, iar deoarece necesită 6 înmulțiri, determinarea candidatelor din buchete necesită 36,6 x 10^4 înmulțiri.

Deoarece inițial există N_{TM}=2790 mănunchiuri, iar în final rămân N_{TM} =N=256 mănunchiuri, înseamnă că se fac 2790-256=2534 uniri între mănunchiuri. Pentru k=3, relația 3.4 implică 6 înmulțiri și o împărțire, adică, în total unirile între mănunchiuri necesită 2534 x 7=17738 înmulțiri.

Referitor la determinarea celui mai apropiat buchet față de un buchet cu 2 mănunchiuri, din rezultatele obținute în urma rulării programului rezultă că s-au făcut aproximativ 20 de uniri între buchete, în medie, la fiecare din

cele 17 parcurgeri ale buclei. Ştiind că în total, în această perioadă s-au cercetat 3868 buchete înseamnă că distanţa între centroizii generali a 2 buchete s-a calculat în total de 20 x 3868 = 77360 ori. Ştiind că o astfel de distanţă necesită 3 înmulţiri (pentru k=3) rezultă un total de aproximativ 23 x 10^4 înmulţiri.

Deoarece, la început sunt N_B=393 buchete şi în final rămân 65, pentru k=3, rezultă că s-au făcut 393-65=328 uniri între buchete, adică 328 x 7=2296 înmulţiri, pentru calcularea centroidului general.

Considerând în final, toate înmulţirile din cele 3 blocuri rezultă un număr total de aproximativ 60 x 10^4 înmulţiri. Bineânţeles că aceste înmulţiri reprezintă operaţiile necesare numai în cazul subvectorilor cu componentele 1, 2 şi 3. Pentru subvectorii cu componentele 4, 5, 6 numărul de înmulţiri este cam de acelaşi ordin, iar pentru subvectorii cu componentele 7, 8, 9 şi 10 este ceva mai mare datorită dimensiunii k=4 a vectorilor.

Dacă se adaugă şi înmulţirile din cadrul algoritmului de împărţire în buchete, se obţine un număr total de aproximativ 67 x 10^4 înmulţiri.

Aşa cum s-a arătat anterior, o metodă alternativă pentru situaţia când $N_B > N$ ar fi mărirea numărului maxim de vectori pe buchet. De exemplu, pentru un număr maxim de 40 de vectori pe buchet, ar putea rezulta 100 de bucheţi cu 28 de vectori fiecare, în medie. În acest caz s-ar face câte 50 de uniri între mănunchiurile candidate, necesitând aproximativ 50 de parcurgeri ale buclei mari (fără să se facă uniri între buchete). Pentru determinarea candidatelor ar trebui să se calculeze aproximativ 50 x $(C_{28}^2 + C_{28}^2 + C_{27}^2 + C_{27}^2 + \ldots + C_3^2 + C_3^2) = 73 \times 10^4$ de distanţe între mănunchiuri (numărul mediu de mănunchiuri pe buchet scade cu un pas de 0,5 de la 28 până la aproximativ 2,5). Fără a mai considera numărul de înmulţiri necesar pentru calculul unei distanţe, este evident că în acest caz complexitatea este mult mai mare. Chiar dacă complexitatea nu este un factor foarte important având în vedere că aceşti algoritmi se desfăşoară o singură dată, înaintea codării, valoarea mai mică obţinută constituie un argument pentru ideea propusă.

Pentru o primă apreciere a calităţii cărţii de cod proiectate prin algoritmul PNN rapid, în tabelul 3.2 se arată eroarea medie, sub formă de eroare pătratică, în Hz^2, obţinută prin cuantizarea, prin căutare totală a celor

2790 vectori de antrenament cu cei 256 de vectori de reproducere, precum și aceeași eroare pentru cuantizarea a 310 vectori de test, diferiți de cei de antrenament, dar rostiți de același vorbitor (separat pentru cele trei tipuri de subvectori). Programele care implementează cuantizarea au fost scrise în limbajul C.

Tabelul 3.2: *Eroarea obținută la un cuantizor vectorial folosind cartea de cod* PNN.

Subvectori		$f_1 \cdots f_3$	$f_4 \cdots f_6$	$f_7 \cdots f_{10}$
Eroarea medie în Hz^2 pentru:	Vectori de antrenament	523	1641	2171
	Vectori de test	705	2211	2823

Se observă că pentru vectorii de test eroarea este ceva mai mare, lucru normal, deoarece ei nu au fost cuprinși în procesul de proiectare. De exemplu, pentru subvectorul cu componentele f_4, f_5, f_6, o eroare medie pătratică de 2211 Hz^2 reprezintă o diferență de aproximativ 27 Hz, în medie, între componentele cuantizate și cele originale (în general aceste componente au valori cuprinse între 1000 și 2500 Hz).

Autorul a proiectat, de asemenea, o carte de cod în care, prin despicări succesive și aplicări ale algoritmului generalizat al lui Lloyd se obțin cărțile de cod cu dimensiunile 2, 4, 8,..., 256, la fel cum s-a procedat în [4] (algoritm descris pe scurt și în paragraful 3.1). In tabelul 3.3 se prezintă rezultatele obținute în acest caz.

Tabelul 3.3: *Eroarea obținută la un cuantizor vectorial folosind cartea de cod Lloyd, prin despicări succesive.*

Subvectori		$f_1 \cdots f_3$	$f_4 \cdots f_6$	$f_7 \cdots f_{10}$
Eroarea medie în Hz^2 pentru:	Vectori de antrenament	573	1600	2390
	Vectori de test	769	2240	3178

Se observă că rezultatele obținute în acest caz sunt ceva mai slabe decât cele obținute pentru o carte de cod proiectată cu algoritmul PNN rapid. În plus, complexitatea algoritmului de proiectare este mult mai mare.

Au fost necesare în medie câte 12 iteraţii Lloyd pentru fiecare dimensiune a cărţii de cod. De exemplu, pentru fiecare din cele 3 cărţi de cod s-au calculat $2790x12x(2+4+8+...+256)=17x10^6$ distanţe k-dimensionale (adică $51x10^6$ înmulţiri pentru k=3).

În final, s-a aplicat algoritmul generalizat al lui Lloyd considerând ca şi carte iniţială, pentru îmbunătăţit, cartea de cod obţinută prin algoritmul PNN rapid. Convergenţa a fost mult mai rapidă fiind necesare 6-7 iteraţii. Astfel, pe lângă cele $67x10^4$ înmulţiri necesare pentru obţinerea cărţii iniţiale, s-au mai efectuat $2790 \times 6 \times 256 \times 3 \cong 12.8 \cdot 10^6$ înmulţiri, adică un total de $13,4x10^6$ înmulţiri, ceea ce reprezintă aproximativ 25% din înmulţirile necesare proiectării cărţii de cod folosind despicări succesive şi algoritmul generalizat al lui Lloyd. In tabelul 3.4 se prezintă erorile obţinute în acest caz.

Tabelul 3.4: *Eroarea obţinută la un cuantizor vectorial folosind cartea de cod PNN îmbunătăţită prin algoritmul Lloyd.*

Subvectori		$f_1,...,f_3$	$f_4,...,f_6$	$f_7,...,f_{10}$
Eroarea medie în Hz^2 pentru:	Vectori de antrenament	476	1471	1939
	Vectori de test	723	2133	2866

În comparaţie cu rezultatele din tabelul 3.2 se observă o îmbunătăţire a performanţelor, pentru secvenţa de antrenament, ceea ce era obligatoriu, în timp ce pentru secvenţa de test eroarea scade doar pentru subvectorii $f_4,...,f_6$, crescând puţin pentru subvectorii $f_1,...f_3$ şi $f_7,...f_{10}$. Acest rezultat este explicabil, deoarece cartea de cod a fost îmbunătăţită pentru secvenţa de antrenament, secvenţa de test diferind de aceasta.

În concluzie, aceste rezultate confirmă cele obţinute în [14] şi anume că proiectarea unei cărţi de cod prin algoritmul PNN rapid are o complexitate mult mai mică decât algoritmul clasic al lui Lloyd, având în plus şi o eroare mai mică. **Meritul autorului este că a introdus o variantă modificată a algoritmului PNN rapid, pe care a aplicat-o pentru vectori de tip LSF, adaptând-o în funcţie de caracteristicile acestui tip de vectori.**

3.3 Analiza calităţii unui cuantizor vectorial în funcţie de diferitele forme ale erorii pătratice ponderate

Aşa cum s-a arătat în paragraful 2.2, calitatea unui cuantizor vectorial pentru parametri LSF se apreciază prin eroarea spectrală medie *SD*. Eroarea spectrală care se face la cuantizarea unui vector i, căruia îi corespunde o densitate spectrală de putere $S_i(f)$, prin alt vector căruia îi corespunde densitatea spectrală de putere $\hat{S}_i(f)$ este

$$SD_i^2 = \frac{2}{f_e}\int_0^{f_e/2}\left[10\log_{10}(S_i(f)) - 10\log_{10}(\hat{S}_i(f))\right]^2 df \tag{3.9}$$

Pentru calculul lui SD_i^2, s-a descompus aria reprezentată de această integrală în porţiuni foarte mici, cu lăţimea de 1 Hz, s-au calculat ariile acestor porţiuni, însumându-se apoi în final aceste arii. Programul prin care se calculează această mărime a fost scris de autor în limbajul C.

Eroarea spectrală medie pentru cazul când cuantizarea s-a făcut folosind eroarea pătratică s-a obţinut la valoarea *SD*=0,933 dB, 5 cadre din cele 310 având erori spectrale mai mari decât 2 dB, întrunind cerinţele necesare pentru o cuantizare transparentă (vezi paragraful 2.2).

În scopul reducerii erorii spectrale medii şi a numărului de cadre cu eroare mai mare decât 2 dB s-a folosit apoi eroarea pătratică ponderată,

$$d(f,\hat{f}) = \sum_{i=1}^{10}\left[w_i(f_i - \hat{f}_i)\right]^2, \tag{3.10}$$

unde ponderea w_i s-a calculat cu relaţia

$$w_i = \left[S(f_i)\right]^r, r = 1/6, \tag{3.11}$$

iar $S(f_i)$ s-a calculat cu relaţia exactă (2.19).

Folosirea erorii pătratice ponderate are ca efect cuantizarea mai precisă a componentelor f_i la care densitatea spectrală de putere are valori mai mari, în scopul reconstituirii mai precise a porţiunilor din spectru cu amplitudine mai mare.

Pentru verificarea experimentală a acestei ipoteze autorul a considerat un vector cu parametri LSF (din secvența de antrenament), pentru care a calculat ponderile w_i aferente, tabelul 3.5. Apoi s-au făcut diverse modificări unora dintre acești parametri, calculându-se în fiecare caz eroarea spectrală între vectorul inițial și cel cu parametrii modificați, ca în tabelul 3.6.

Tabelul 3.5: *Componentele* LSF *și ponderile aferente.*

f_i	201	423	800	923	1477	1658	2156	2528	2996	3512
W_i	2,73	1,78	2,20	1,73	1,28	1,25	0,80	0,78	0,58	0,58

Tabelul 3.6: *Erorile spectrale în funcție de modificările parametrilor* LSF.

Modificarea făcută	*SD*
$f_i \leftarrow f_i$+20 Hz i=1,2,3	0,62 dB
$f_i \leftarrow f_i$+20 Hz i=8,9,10	0,36 dB
$f_1 \leftarrow f_1$+20 Hz	0,42 dB
$f_2 \leftarrow f_2$+20 Hz	0,27 dB
$f_3 \leftarrow f_3$+20 Hz	0,52 dB
$f_4 \leftarrow f_4$+20 Hz	0,25 dB
$f_5 \leftarrow f_5$+20 Hz	0,25 dB
$f_6 \leftarrow f_6$+20 Hz	0,29 dB

Rezultatele din tabelul 3.6 confirmă în linii mari aceste ipoteze, deoarece se observă că eroarea spectrală este cu atât mai mare cu cât ponderile parametrilor modificați sunt mai mari.

În scopul evidențierii efectului global al erorii pătratice ponderate asupra întregii secvențe de test, autorul a realizat experimentul descris în continuare. S-a considerat mai întâi, pentru subvectorul (f_1, f_2, f_3), cartea de cod obținută prin algoritmul PNN rapid. Cei 310 vectori de test s-au cuantizat folosind atât eroarea pătratică ponderată (prescurtat e.p.p.), cât și cea pătratică (e.p.), prin intermediul cărții de cod amintite. O mare parte din vectori (circa 80%) și-au găsit același vector de reproducere prin ambele

tipuri de distanţe, ceilalţi 20% găsindu-şi vectori de reproducere diferiţi. În tabelul 3.7 se arată, pentru 2 vectori de cuantizat din aceasta ultimă categorie, vectorii de reproducere găsiţi în cele 2 cazuri, precum şi ponderile w_i (s-au considerat numai subvectorii cu componentele f_1, f_2, f_3).

Tabelul 3.7: *Rezultatele cuantizării folosind e.p. şi e.p.p.*

f_i	212	340	558	208	297	665
w_i	3,09	2,80	1,97	3,16	2,51	1,40
e.p	198	323	554	233	296	673
e.p.p.	205	355	574	197	298	698

După cum se vede din datele prezentate pentru cei 2 vectori, componentele 1, în primul rând, şi 2, care au ponderi mari, sunt cuantizate mai precis când se foloseşte eroarea pătratică ponderată. În schimb, componenta 3, este cuantizată mai puţin precis în cazul acesta. Chiar dacă în cazul erorii pătratice ponderate, eroarea de cuantizare sub formă pătratică este mai mare, eroarea spectrală scade, de exemplu, pentru primul vector, de la 0,396 dB la 0,354 dB, iar pentru al doilea, de la 0,817 dB la 0,595 dB. În ambele cazuri s-au considerat componentele $f_4,...,f_{10}$ de la vectorul cuantizat, identice cu cele ale vectorului de cuantizat.

Referitor la modul de exprimare a erorii pătratice ponderate, autorul a observat că, indiferent dacă ponderea w_i este la puterea a 2-a, ca în relaţia (3.10) sau la puterea întâi, eroarea *SD* are valori foarte apropiate, astfel încât, pentru simplitatea calculelor s-a folosit w la puterea întâi.

În continuare s-a efectuat cuantizarea întregii secvenţe de test folosind eroarea pătratică ponderată, pentru fiecare din cei 3 subvectori folosindu-se cartea de cod obţinută prin algoritmul PNN rapid. Eroarea spectrală medie obţinută în acest caz a fost de 0,903 dB, obţinându-se o scădere de 0,03 dB faţă de cea obţinută când s-a folosit eroarea pătratică, cu aceleaşi 3 cărţi de cod. Numărul de vectori cu eroare spectrală mai mare ca 2 dB a fost de 3.

În scopul îmbunătăţirii şi mai mult a performanţelor, cele 3 cărţi de cod s-au proiectat pentru eroarea pătratică ponderată. Astfel, plecând de la

cele 3 cărţi de cod obţinute prin algoritmul PNN rapid, s-a aplicat algoritmul generalizat Lloyd, folosind însă eroarea pătratică ponderată pentru cuantizarea secvenţelor de antrenament.

Cu ajutorul celor 3 cărţi de cod obţinute în acest fel s-a făcut din nou cuantizarea secvenţei de test, obţinându-se o reducere suplimentară a erorii spectrale cu 0,01 dB, un singur vector mai având eroarea spectrală peste 2 dB.

În continuare, pe baza relaţiilor obţinute pentru densitatea spectrală de putere, în paragraful 2.1, autorul a introdus 2 noi ponderi. Având în vedere forma expresiilor (2.24) şi (2.27) pentru $S(f_i)$, constanta r din expresia (3.11) a lui w_i, care în literatură se spune că poate lua valori în intervalul 0,1 ... 0,2, s-a ales valoarea 1/6=0,1666 (cum dealtfel s-a şi utilizat deja în acest paragraf). Astfel, pentru w_i rezultă relaţiile

$$w_i = \frac{K'}{(f_{i+1} - f_i)(f_i - f_{i-1})}, \quad i = 1, \ldots, 10, \tag{3.12}$$

unde $f_0 = 0$ şi $f_{11} = \frac{f_e}{2} = 4000Hz$, şi

$$w_i = \frac{K_1'}{(f_{i+3} + 3f_{i+1} - 4f_i)(4f_i - 3f_{i-1} - f_{i-3})}, i = 1, \ldots, 10. \tag{3.13}$$

Pentru relaţia (3.13) valorile f_i au fost definite în tabelul 2.1, iar constantele K' şi K'_1 sunt rădăcinile de ordinul 6 din constantele K şi K_1 prezentate în paragraful 2.1, adică $K'=1{,}32 \times 10^5$ şi $K'_1=4{,}76 \times 10^6$.

Cu aceste două ponderi s-au făcut aceleaşi etape de prelucrare ca şi cu ponderea în care $S(f_i)$ s-a calculat cu relaţia exactă (2.19). De asemenea s-a utilizat şi ponderea

$$w_i = \frac{1}{f_{i+1} - f_i} + \frac{1}{f_i - f_{i-1}}, \tag{3.14}$$

dată empiric în [13] de Laroia şi colectivul, prezentată şi în paragraful 2.2.

În tabelul 3.8 se prezintă rezultatele obţinute (eroarea spectrală medie SD şi numărul de cadre cu eroarea spectrală mai mare decât 2 dB, n_m), pentru diferite ponderi utilizate, specificându-se dacă s-a utilizat cartea de cod proiectată pentru eroarea pătratică ponderată (e.p.p.) sau doar cea pentru eroarea pătratică (e.p.).

Tabelul 3.8: *Rezultatele obținute prin cuantizarea vectorială cu eroarea pătratică ponderată.*

Ponderea	Cartea de cod	*SD* [dB]	n_m
(3.11)	e.p.	0,903	3
	e.p.p.	0,893	1
(3.12)	e.p.	0,898	4
	e.p.p.	0,888	2
(3.13)	e.p.	0,903	3
	e.p.p.	0,894	2
(3.14)	e.p.p.	0,899	2
Fără pondere	e.p.	0,933	5

Din tabelul 3.8 se pot trage mai multe concluzii.

- în primul rând se observă că reducerea mai puternică a erorii spectrale medii se obține prin utilizarea erorii pătratice ponderate, chiar dacă cartea de cod nu este proiectată pentru această eroare, iar proiectarea cărții de cod pentru eroarea pătratică ponderată produce o scădere mai mică a erorii spectrale medii.

- cu cele două ponderi propuse de autor se obțin rezultate apropiate față de cele obținute cu relațiile propuse în literatură (prin relațiile (3.11) și (3.14)).

- ponderea propusă de autor prin relația (3.12) dă rezultate ceva mai bune pentru *SD* decât pondrerea calculată pe baza relației exacte a densității spectrale de putere $S(f_i)$, adică cea dată de relația (3.11); acest lucru pare neașteptat ținând cont că în paragraful 2.1, tabelul 2.2, valorile obținute pentru $S(f_i)$ difereau de cele calculate cu formula exactă.

- o explicație a concluziei anterioare poate fi dată prin faptul că $S(f_i)$ corespunzător ponderii (3.12) are variații mari în raport cu valorile obținute prin relația exactă, ceea ce înseamnă că se cuantizează și mai precis componentele f_i cărora le corespund vârfuri în spectre.

- pe de altă parte, tot cu ponderea dată prin (3.12) se obțin rezultate mai slabe în ce privește vectorii cu eroare spectrală mai mare decât 2 dB

În final se poate spune că ponderile propuse de autor dau rezultate similare cu ponderile propuse în literatură, pentru o comparare mai exactă fiind necesară poate o bază de date mai mare, atât ca vectori de antrenament cât şi ca vectori de test. Principalul avantaj al acestor ponderi este însă simplitatea relaţiilor, ele necesitând practic o înmulţire şi o împărţire, în timp ce, pentru ponderea dată de relaţia (3.11), este necesară calcularea exactă a $S(f_i)$, urmată de ridicarea la puterea r. De exemplu, prin relaţia (2.19) sunt necesare pentru calculul lui $S(f_i)$ 11 înmulţiri între numere obţinute prin aplicarea funcţiei *sinus*. În ce priveşte ponderea dată prin relaţia (3.14), chiar dacă este simplă, ea nu este demonstrată, ci dată empiric [15].

În capitolul 3 s-a realizat proiectarea printr-o metodă nouă a cărţii de cod pentru realizarea unui cuantizor vectorial prin despicare şi testarea cărţii de cod obţinute pentru diferite forme ale erorii pătratice ponderate, unele propuse de autor.

4. CUANTIZAREA VECTORIALA A PARAMETRILOR LSF PRIN METODA COMBINATĂ ARBORE–CAUTARE TOTALĂ

Aşa cum s-a arătat la sfârşitul capitolului 2, autorul s-a orientat înspre obţinerea unei metode de cuantizare vectorială a parametrilor LSF de complexitate redusă (metodă rapidă).

Cea mai rapidă metodă de cuantizare vectorială (aşa cum s-a spus, procesul de codare este cel care determină "rapiditatea", decodarea fiind totdeauna la fel de simplă) cunoscută în literatură este metoda de tip arbore, (TSVQ - tree structured VQ) [1], [2], care, în principiu necesită k operaţii (teste) pentru determinarea celor k biţi din codul vectorului. Aşa cum se va arăta, această metodă conduce însă la creşterea erorii de cuantizare, faţă de situaţia când codarea s-ar fi făcut prin căutare totală. În [2] se prezintă o altă metodă rapidă, tot de tip arbore, care se bazează pe o structurare a cărţii de cod, având ca efect o creştere a complexităţii faţă de TSVQ, fără a produce însă creşterea erorii de cuantizare faţă de căutarea totală. Ambele metode se folosesc în literatură, însă doar pentru vectorii formaţi din eşantioane de semnal vocal. **Autorul preia principiul celei de-a doua metode şi îl adaptează astfel încât să fie folosit la cuantizarea parametrilor LSF, rezultând cea mai rapidă metodă de cuantizare vectorială a parametrilor LSF, fără a produce creşterea erorii de cuantizare.**

4.1 Cuantizarea vectorială de tip arbore

În acest paragraf autorul demonstrează de ce un cuantizor vectorial de tip arbore (TSVQ) este suboptimal faţă de un cuantizor vectorial de tip Voronoi (în care codarea se face prin căutare totală).

Structura generală a unui TSVQ (partea de codare) este prezentată în figura 4.1. Arborele conţine d niveluri, 2^{d-1} noduri interne şi 2^d noduri terminale. Fiecărui nod terminal îi corespunde un cod sau un vector de

reproducere. Evident că numărul de noduri terminale, 2^d este egal cu N, mărimea cărții de cod, de unde $d=\log_2 N$.

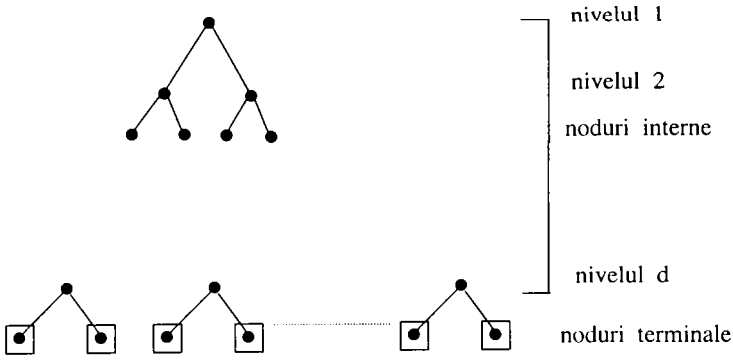


Figura 4.1: *Codorul unui cuantizor de tip arbore.*

În procesul de codare, vectorul de cuantizat parcurge o "cale" prin arbore, de la nodul rădăcină, situat pe primul nivel, până la un nod terminal, în funcție de testele binare făcute în nodurile interne. Prin testul făcut pe fiecare nivel se determină câte un bit al codului.

Pentru a proiecta un astfel de cuantizor vectorial trebuie specificat modul cum se fac testele în nodurile interne, precum și vectorii de reproducere corespunzători nodurilor terminale.

În figura 4.2 se arată un exemplu de TSVQ, pentru $d=2$, în spațiul bidimensional ($k=2$), arătându-se arborele, figura 4.2a) și regiunile de cuantizare, figura 4.2 b). Distanța folosită în procesul de codare este distanța euclidiană.

Pe nivelul 1, întreg spațiul $\Re^2$ este împărțit în 2 regiuni de cuantizare distincte. Testul care se face în singurul nod de pe nivelul 1 determină de care parte a frontierei reprezentată prin dreapta A este situat vectorul de cuantizat, $\boldsymbol{x}$, sau, echivalent, de care din cei doi vectori de reproducere notați $\boldsymbol{y}_1$ și $\boldsymbol{y}_2$ este mai apropiat. Cei doi vectori de reproducere sunt egal depărtați de dreapta A. Pe nivelul 2, cele două regiuni de cuantizare corespunzătoare nivelului 1, se împart fiecare, separat, în câte două regiuni de cuantizare distincte, delimitate de dreptele B și C. Vectorii de reproducere $\boldsymbol{y}_{11}$ și $\boldsymbol{y}_{12}$ sunt egal depărtați de dreapta B, iar $\boldsymbol{y}_{21}$ și $\boldsymbol{y}_{22}$ sunt

egal depărtaţi de dreapta C. Deoarece proiectarea cuantizorului (obţinerea vectorilor de reproducere) se face separat în cele două regiuni corespunzătoare nivelului 1, vectorii y_{11} şi y_{21}, respectiv y_{12} şi y_{22} nu mai

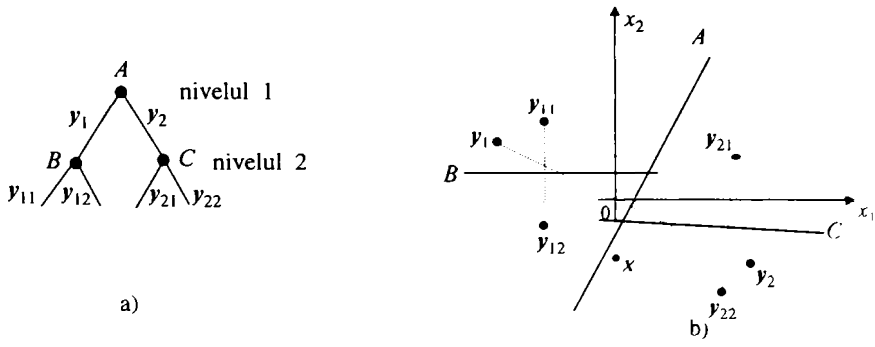


Figura 4.2: *Structura arborelui şi a regiunilor de cuantizare pentru un* TSVQ *cu* N=4 *şi* k=2.

sunt egal depărtaţi de feţele comune ale regiunilor respective ca la un cuantizor vectorial de tip Voronoi. Astfel, în procesul de cuantizare, vectorul x, după ce pe nivelul 1 a fost atribuit regiunii corespunzătoare lui y_2, în urma testului de pe nivelul 2 se atribuie regiunii corespunzătoare lui y_{22}, deşi se vede că este mai apropiat de y_{12}. În acest fel, vectorul de cuantizat x nu-şi mai găseşte cel mai apropiat vector de reproducere, producând creşterea erorii de cuantizare. În acest exemplu simplu de TSVQ, cartea de cod are N=4 vectori de reproducere. Evident că în practică numărul de vectori de reproducere este mult mai mare.

Ceea ce este caracteristic acestui tip de cuantizor vectorial, explicând rapiditatea procesului de codare, este faptul că fiecare test elimină jumătate din vectorii de reproducere care ar putea reprezenta versiunea cuantizată a vectorului de cuantizat.

Considerând tot cazul k=2, dreptele după care se fac testele în noduri pot să fie înclinate faţă de axele sistemului de coordonate, (A şi C în figura 4.2), sau pot să fie perpendiculare pe aceste axe (B). În primul caz testul implică înlocuirea componentelor vectorului de cuantizat în ecuaţia dreptei care separă cele două semiplane şi efectuarea unei comparări. Dacă toate

dreptele de test sunt perpendiculare, se compară una dintre componentele vectorului de cuantizat cu o valoare, corespunzătoare uneia din dreptele perpendiculare. Evident că în al doilea caz testul este mai simplu, iar regiunile de cuantizare rezultate sunt de formă dreptunghiulară. Oricum, în ambele cazuri, regiunile de cuantizare diferă de regiunile obținute în cazul unui cuantizor vectorial de tip Voronoi, unde fiecare față a frontierei unei regiuni este egal depărtată de doi vectori de reproducere, așa cum se arată în figura 4.3.

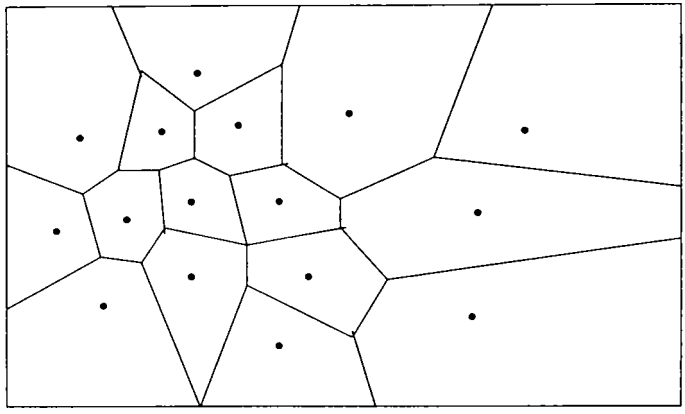


Figura 4.3: *Regiunile de cuantizare la un cuantizor vectorial de tip Voronoi.*

Pe baza celor prezentate rezultă suboptimalitatea unui TSVQ, deoarece regiunile de cuantizare diferă de cele optimale și, în plus, nu se găsește totdeauna cel mai apropiat vector de reproducere din cei disponibili. Această problemă a suboptimalității este tratată de autor și în [48].

4.2 Principiul cuantizării vectoriale prin metoda combinată arbore-căutare totală

În [2] se descrie o metodă de cuantizare vectorială care combină metoda de tip arbore cu metoda de căutare totală. Efectul este obținerea aceleiași erori de cuantizare ca și la căutarea totală cu prețul creșterii complexității față de metoda de tip arbore.

La această metodă de cuantizare, codarea se face tot după o structură de tip arbore, care conține d niveluri, 2^d-1 noduri interne și 2^d noduri terminale. Deosebirea esențială față de metoda TSVQ este că fiecărui nod terminal îi corespund mai mulți vectori de reproducere. După parcurgerea arborelui, cu ajutorul testelor binare efectuate asupra vectorului de cuantizat, se ajunge într-un nod teminal, unde se face o căutare totală printre vectorii de reproducere corespunzători acelui nod terminal, pentru a se găsi codul sau cuantizatul vectorului de cuantizat. Pentru un astfel de cuantizor, numărul de niveluri d este mai mic decât $\log_2 N$.

Pentru a se proiecta un astfel de cuantizor vectorial, trebuie să fie disponibili vectorii de reproducere ai unui cuantizor de tip Voronoi, optimizat conform algoritmului generalizat al lui Lloyd. Acești vectori de reproducere se vor regăsi printre vectorii corespunzători nodurilor terminale.

Deși procesul de codare, până la ajungerea într-un nod terminal, se face identic cu cel de la TSVQ, există o deosebire majoră: dacă la TSVQ fiecare test elimina jumătate din vectorii de reproducere candidați, în acest caz se elimină mai puțin de jumătate din acești vectori. Mai precis, există unii vectori care rămân printre vectorii posibil de a reprezenta cuantizatul vectorului de intrare, indiferent de rezultatul testului binar. Sau, în urma testului binar, vectorii de reproducere disponibili se împart în două grupe care au și elemenete comune.

În continuare autorul prezintă printr-un exemplu din spațiu $\Re^2$, pentru un arbore cu un singur nivel, modul de determinare a vectorilor care se elimină și care rămân, în funcție de testul efectuat.

În figura 4.4 se arată regiunile de cuantizare și vectorii de reproducere, pentru un cuantizor de tip Voronoi, cu N=16. Testul se face cu ajutorul unei drepte, notată A, perpendiculară pe axa $0x_1$, caracterizată prin $x_1=a$. Se presupune că vectorul de cuantizat este $\boldsymbol{x}=(x_1, x_2)$. Deoarece $x_1>a$, trebuie reținuți în primul rând toți vectorii de reproducere care au componenta $x_1>a$, adică cei notați 1, 7, 12, 13, 14, 15, 16. Regiunile unora dintre acești vectori intersectează dreapta A (vectorii 1, 7, 16). Este posibil să existe vectori de reproducere care au $x_1< a$, dar regiunile lor de cuantizare conțin și puncte cu $x_1>a$, adică intersectează dreapta A. Deoarece în aceste puncte

se pot găsi posibili vectori de cuantizat, cazul vectorului x, trebuie reținuți și astfel de vectori (vectorii 10 și 11). Ceilalți vectori de reproducere, adică 2, 3, 4, 5, 6, 8 și 9, se elimină.

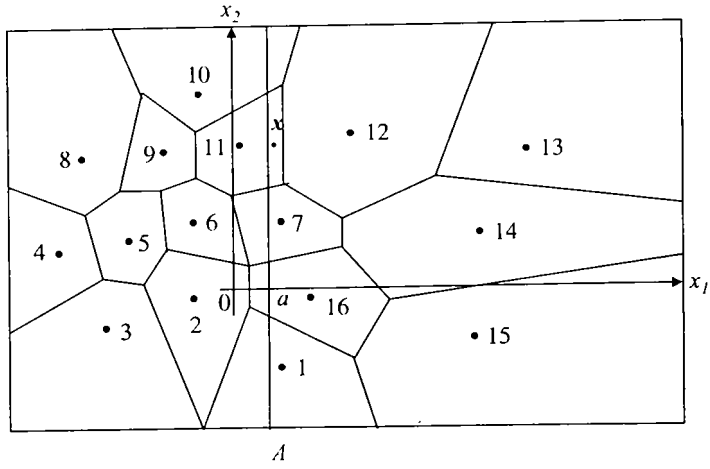


Figura 4.4: *Regiunile și vectorii de reproducere pentru un cuantizor de tip Voronoi cu* $N=16$.

Apoi se face o căutare totală printre toți vectorii de reproducere reținuți. În acest fel, vectorul de cuantizat își va găsi cu siguranță cel mai apropiat vector de reproducere din cartea de cod a cuantizorului de tip Voronoi, adică cel notat cu 11.

În concluzie, indiferent de poziția vectorului de cuantizat față de dreapta de test, se reține vectorii de reproducere ale căror regiuni intersectează dreapta de test, împreună cu vectorii de reproducere situați de aceeași parte cu vectorul de cuantizat față de dreapta de test, și ale căror regiuni de cuantizare nu intersectează această dreaptă.

Desigur că procesul de codare poate continua pe mai multe niveluri. Pentru aceasta, regiunea situată la stânga dreptei A, împreună cu vectorii de reproducere reținuți când vectorul de cuantizat are $x'_1<a$ se atribuie pentru analiză primului nod de pe nivelul 2, iar regiunea situată de cealaltă parte a dreptei A împreună cu vectorii de reproducere reținuți când $x'_1>a$ se atribuie celuilalt nod de pe nivelul 2. În aceste noduri se reiau separat procesele executate în nodul rădăcină (stabilirea dreptei de test și a vectorilor de

reproducere reţinuţi în cele 2 cazuri). Procesul continuă în acest mod, până când numărul vectorilor de reproducere reţinuţi scade suficient de mult, pentru a permite o căutare totală cu o complexitate cât mai mică.

În continuarea acestei lucrări, această metodă va fi referită ca metoda de cuantizare vectorială *combinată arbore-căutare totală.*

Creşterea în complexitate faţă de TSVQ înregistrată la această metodă se datorează căutării care se face în nodurile terminale. Însă, dacă proiectarea este realizată corespunzător, adică vectorii de reproducere care se reţin, respectiv, se elimină sunt corect determinaţi, atunci eroarea medie de cuantizare este aceeaşi cu cea care s-ar obţine printr-o căutare totală realizată asupra celor N vectori de reproducere ai cuantizor de tip Voronoi.

4.3 Structurarea cărţii de cod a unui cuantizor vectorial în vederea aplicării metodei combinate arbore-căuatre totală

4.3.1 Descrierea generală a procesului de structurare

Pentru aplicarea metodei combinate arbore-căutare totală trebuie să fie disponibilă o carte de cod optimă, specifică unui cuantizor de tip Voronoi. Această carte de cod trebuie prelucrată (structurată) în vederea determinării pentru fiecare nod intern al arborelui a componentei vectorului de cuantizat care se va testa (şi a valorii cu care se compară componenta respectivă) precum şi a vectorilor de reproducere asociaţi fiecărui nod terminal.

Se notează cu $m = 2^d$ numărul nodurilor terminale ale arborelui, respectiv cu n_i, $i=1,...,m$, numărul de vectori de reproducere asociat cu fiecare din aceste noduri, iar p_i, $i=1,...,m$, reprezintă probabilitatea ca un vector de cuantizat să fie atribuit în urma testelor, nodului terminal i. În aceste condiţii numărul de distanţe calculate pentru codarea unui vector, efectuate într-unul din nodurile terminale este

$$n_{dist} = p_1 n_1 + p_2 n_2 + \ldots + p_m n_m \tag{4.1}$$

Se consideră că în fiecare nod intern, pentru efectuarea testului, componenta j a vectorului de cuantizat se compară cu o valoare, notată v_j,

adică dreapta în raport cu care se face testul este perpendiculară pe una din axele sistemului de coordonate (în cazul k=2). Pentru o proiectare optimă, în fiecare nod, perechea (j, v_j) trebuie aleasă astfel încât numărul de distanțe n_{dist} să fie minim. Componenta j poate lua valori întregi din mulțimea {1, 2, ..., k}, iar v_j aparține intervalului în care componenta respectivă poate lua valori conform secvenței de antrenament. În acest fel, pentru fiecare din cele 2^d-1 noduri interne ale arborelui trebuie testate toate perechile posibile (j, v_j), în vederea găsirii unui minim pentru n_{dist}.

Dacă, de exemplu v_j ia, în medie 50 de valori în intervalul specificat, într-un nod ar exista $50k$ perechi (j, v_j) posibile. În total, expresia n_{dist} poate avea $(50\ k)^{m-1}$ valori posibile. Dacă k=3 și m=64, se obțin 150^{63} cazuri posibile. Această operație este deosebit de complexă și de aceea, așa cum se propune și în [2], se realizează o optimizare locală. Adică, în fiecare nod, independent de alte noduri, se determină perechea (j, v_j) pentru care, la cuantizarea unui vector $\boldsymbol{x}$, s-ar calcula un număr minim de distanțe, în funcție de vectorii de reproducere reținuți în urma testului binar efectuat.

În acest sens se consideră un nod intern al arborelui notat i. În urma divizării realizată de dreapta de test în nodul de pe nivelul precedent din care se ajunge în acest nod intern, se obțin două regiuni dreptunghiulare. Una dintre aceste regiuni, cea atribuită nodului i, notată R_i, este intersectată de regiunile de cuantizare a N_i vectori de reproducere (acești N_i vectori de reproducere reprezintă vectorii reținuți în cazul când vectorul de cuantizat este plasat în regiunea care apoi s-a atribuit nodului i).

În figura 4.5 se arată această configurație pentru cazul k=2. În acest caz, în perechea (j, v_j) față de care se face testul, j poate lua valoarea 1 sau 2, iar $v_j \in (a_j, b_j)$.

Se consideră $\boldsymbol{x}=(x_1, x_2)$ un vector de cuantizat (nearătat în figura 4.5). În acest caz p_s este probabilitatea ca un astfel de vector să aibe componenta $x_j \le v_j$, iar p_d este probabilitatea ca vectorul respectiv să aibe $x_j > v_j$, j=1,2. Rezultă că $p_s+p_d=1$. n_s este numărul regiunilor de cuantizare care conțin vectori pentru care $x_j \le v_j$, iar n_d este numărul regiunilor de cuantizare care conțin vectori pentru care $x_j > v_j$, j =1,2. Evident că $n_s+n_d>N_i$. În aceste condiții, dintre toate perechile posibile (j, v_j), trebuie aleasă aceea pentru

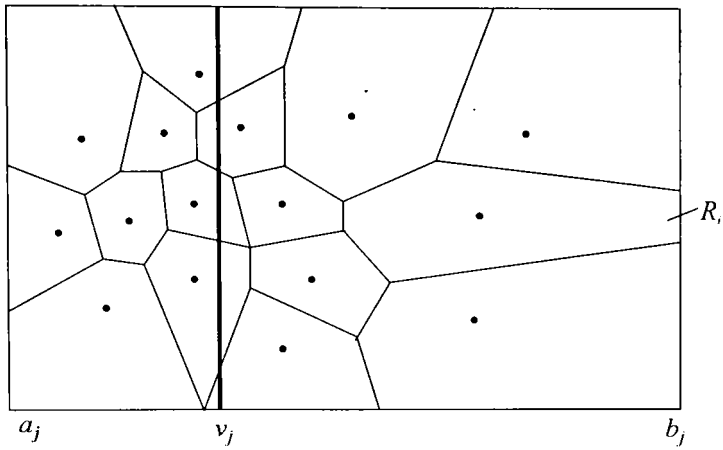


Figura 4.5: *Regiunea R_i și regiunile de cuantizare care o intersectează.*

care expresia $n'_{dist}=p_s n_s+p_d n_d$ este minimă. Adică, dacă un vector de cuantizat ajunge în nodul i în urma testelor efectuate pe nivelurile precedente, după testul efectuat în nodul i, își va găsi cel mai apropiat vector de reproducere în urma unui număr minim de distanțe calculate. (Se presupune că nodul i se află pe nivelul d, ultimul al arborelui, iar în acest caz n_s și n_d reprezintă numărul de vectori de reproducere asociați cu cele două noduri terminale la care se ajunge din nodul i).

Acest procedeu, de determinare a perechii (j, v_j) optime se aplică pentru toate nodurile de pe un anumit nivel al arborelui. Dacă valorile n_s și n_d din aceste noduri sunt suficient de mici, structurarea se încheie. În caz contrar se trece pe nivelul următor. Valorile n_s și n_d ale acestor noduri vor fi valorile n_1, n_2, ..., n_m asociate nodurilor terminale. Bineînțeles că pentru fiecare nod terminal trebuie să se cunoască nu numai valoarea n_i, ci și care sunt cei n_i vectori de reproducere asociați nodului (din cei N vectori de reproducere ai cărții de cod).

Probabilitățile p_s și p_d se determină pe baza vectorilor din secvența de antrenament. Fiecărei regiuni de tipul R_i îi sunt atribuiți pe lângă cei N_i vectori de reproducere, și N_{ai} vectori de antrenament situați în acea regiune. În funcție de perechea (j, v_j), acești vectori de antrenament se împart în 2 grupe distincte, adică cei cu probabilitatea p_s și cei cu probabilitatea p_d. După stabilirea perechii (j, v_j) conform criteriului de minim

enunţat, cele două grupe de vectori de antrenament rezultate (cele care au probabilităţile p_s şi, respectiv, p_d), vor reprezenta vectorii de antrenament atribuiţi celor 2 regiuni corespunzătoare celor 2 noduri de pe nivelul următor.

Dacă determinarea probabilităţilor p_s şi p_d nu ridică probleme, fiind simplă dacă se cunoaşte secvenţa de antrenament, determinarea lui n_s şi n_d pentru diferite perechi (j, v_j) este mai dificilă. Sau, mai precis, ceea ce este mai dificil este determinarea acelor regiuni de cuantizare care intersectează dreapta (planul, pentru $k>2$) în raport cu care se face testul, adică $x_j=v_j$. Vectorii de reproducere corespunzători acestor regiuni de cuantizare, se vor găsi în ambele grupe de vectori de reproducere care rezultă în urma efectuării testului.

În [2] se propune pentru determinarea parametrilor n_s şi n_d o metodă numită metoda proiecţiilor. Conform acestei metode se determină pentru orice valoare v_j a oricăreia din cele k componente, care sunt valorile lui n_s şi n_d. Aceasta presupune următorii paşi.

1. Se generează nişte vectori $(x_1,..., x_k)$ uniform distribuiţi în porţiunea (mărginită) din $\Re^k$ în care vectorii de cuantizat pot lua valori, adică $x_j \in (a_j, b_j)$, $j=1, ..., k$. Aceşti vectori se cuantizează prin intermediul cărţii de cod cu N vectori de reproducere ai cuantizorului Voronoi, proiectată conform algoritmului Lloyd.

2. În urma cuantizării, vectorii uniform distribuiţi sunt repartizaţi în cele N regiuni de cuantizare. Pe baza acestor vectori se determină proiecţiile fiecărui regiuni pe cele k componente. Astfel, pentru o regiune R_i, $i=1, ..., N$, proiecţiile după componenta j, $j=1, ..., k$ sunt

$$P^i_{j\,\min} = \min_{x \in R_i}(x_j : Q(\boldsymbol{x}) = \boldsymbol{y}_i) \tag{4.2}$$

şi respectiv

$$P^i_{j\,\max} = \max_{x \in R_i}(x_j : Q(\boldsymbol{x}) = \boldsymbol{y}_i) \tag{4.3}$$

unde vectorii $\boldsymbol{x}$ de forma $\boldsymbol{x}=(x_1, x_2, ..., x_k)$ reprezintă vectorii atribuiţi regiunii R_i în urma procesului de cuantizare. În figura 4.6 se arată proiecţiile

pentru regiunea R_3, pentru cazul k=2. Evident că pentru o determinare cât mai exactă a acestor proiecții, vectorii uniform distribuiți trebuie să fie cât mai apropiați între ei, pentru a acoperi tot spațiul $\Re^2$.

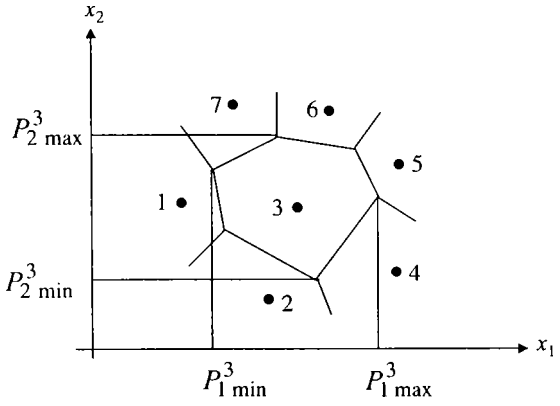


Figura 4.6: *Proiecțiile regiunii R_3 pe cele două axe ale sistemului de coordonate.*

3. Fiind disponibile proiecțiile tuturor celor N regiuni după cele k componente, se pot determina parametrii n_s și n_d pentru orice valoare $v_j \in (a_j, b_j)$, j =1, ..., k, în modul următor: n_s reprezintă numărul regiunilor de cuantizare pentru care $P^i{}_{j\min} < v_j$, iar n_d reprezintă numărul regiunilor de cuantizare pentru care $P^i{}_{j\max} > v_j$; parametrul i, i=1,...,N.

După cum rezultă din prezentarea pașilor parcurși în cadrul metodei proiecțiilor, exceptând pasul 1, predomină operațiile de comparare. Astfel, dacă se notează cu N_{mediu} numărul mediu de vectori uniform distribuiți atribuiți fiecărei regiuni de cuantizare, numărul total de operații (comparații) care se fac la pasul 2 este

$$n_{c2} = N(2N_{mediu})k\,, \tag{4.4}$$

unde factorul 2 apare datorită faptului că se fac N_{mediu} comparații pentru determinarea proiecției minime și, respectiv, același număr de comparații pentru determinarea proiecției maxime. Pentru determinarea numărului de operații care se fac în cadrul pasului 3, se notează cu n_{mediu} numărul mediu

al valorilor v_j pentru care se dorește determinarea parametrilor n_s și n_d, pentru fiecare din cele k componente. Astfel numărul total de operații este

$$n_{c3} = (n_{mediu} 2N)k \,, \tag{4.5}$$

unde factorul 2 apare pentru că se fac comparații cu cele N proiecții minime, pentru determinarea lui n_s și apoi cu cele N proiecții maxime, pentru determinarea lui n_d.

4.3.2 Determinarea parametrilor n_s și n_d prin metoda intersecțiilor

Autorul propune o altă metodă pentru determinarea parametrilor n_s și n_d. Înainte de descrierea metodei generale, pentru determinarea lui n_s și n_d pentru toate valorile v_j ale tuturor componentelor, se descrie aplicarea metodei pentru o singură valoare v_j a unei componente. Astfel, se generează vectorii de forma $x=(x_1,..., v_j,..., x_k)$, unde componenta j are o valoare fixă v_j, iar celelalte $k-1$ componente iau valori uniform distribuite în gama în care aceste componente iau valori conform secvenței de antrenament. Apoi, acești vectori se cuantizează prin intermediul celor N vectori de reproducere ai cărții de cod a cuantizorului Voronoi proiectată prin algoritmul Lloyd. Regiunile de cuantizare ale vectorilor de reproducere care au reprezentat cel puțin o dată versiunea cuantizată a vreunui vector de forma $\boldsymbol{x}$, intersectează dreapta (planul, pentru $k>2$) $x_j=v_j$ și prin urmare acești vectori vor face parte din ambele grupe de vectori de reproducere. Vectorii de reproducere care nu au reprezentat niciodată cuantizatul vreunui vector de forma $\boldsymbol{x}$ vor face parte din prima grupă (cea care conține n_s vectori) dacă $x_j \leq v_j$, sau din a doua grupă (care conține n_d vectori) dacă $x_j > v_j$).

În figura 4.7 se exemplifică metoda propusă de autor, pentru cazul $k=2$. Se generează vectori de forma (v_1, x_2), unde v_1 este valoarea pentru care se dorește determinarea parametrilor n_s și n_d, iar x_2 ia valori egal depărtate în gama admisibilă. Acești vectori sunt notați prin "*" și sunt cuantizați pe baza căutării totale, în ordine, prin vectorii de reproducere notați 1, 1, 1, 16, 7, 11, 11 și 10. Acești vectori de reproducere vor face parte din ambele grupe, adică grupa care conține n_s vectori, și, respectiv,

grupa care conţine n_d vectori. Apoi, dintre vectorii de reproducere care nu au reprezentat niciodată versiunea cuantizată a vectorilor de forma (v_1, x_2), cei notaţi 2, 3, 4, 5, 6, 8 şi 9 vor face parte din grupa cu n_s vectori (deci n_s=12) iar cei notaţi 12, 13, 14 şi 15 vor face parte din grupa cu n_d vectori (deci n_d=9).

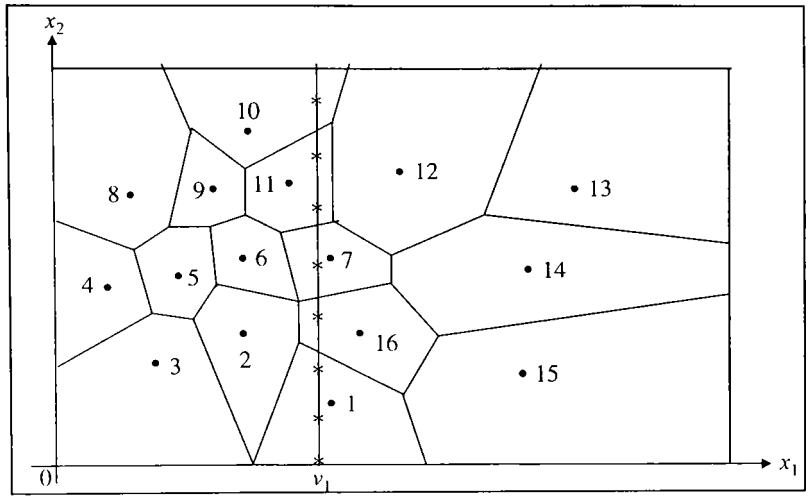


Figura 4.7: *Explicativă pentru determinarea lui n_s şi n_d pentru k=2.*

Pe baza acestui exemplu, în care s-a arătat, cum se determină n_s şi n_d pentru valoarea v_1 a componentei 1, se va arăta prin generalizare, cum se obţin valorile n_s şi n_d corespunzătoare tuturor valorilor v_j ale celor k componente ale vectorilor. **Această metodă propusă de autor va fi referită în continuare ca metoda *intersecţiilor*.**

La fel ca şi la metoda proiecţiilor, mai întâi se generează vectori de forma (x_1,...,x_k), uniform distribuiţi în porţiunea mărginită din $\Re^k$ în care vectorii de cuantizat pot lua valori, adică $x_j \in (a_j, b_j)$. În figura 4.8 se arată aceşti vectori pentru cazul k=2.

Astfel, x_1 ia valori în intervalul (a_1, b_1), egal decalate între ele cu pasul Δ_1, iar x_2 ia valori în intervalul (a_2, b_2), egal decalate cu pasul Δ_2. În practică se poate ca $\Delta_1=\Delta_2$.

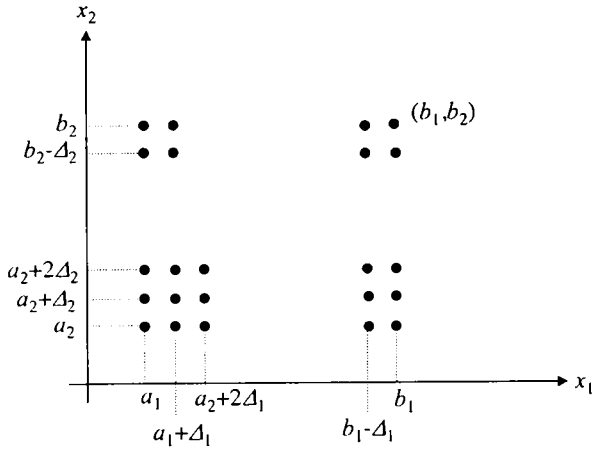


Figura 4.8: *Dispunerea vectorilor uniform distribuiţi în cazul* k=2.

Considerând vectorii care au $x_1=a_1$, adică

$$\begin{cases} a_1, a_2 \\ a_1, a_2 + \Delta_2 \\ \vdots \\ a_1, b_2 \end{cases} \tag{4.6}$$

prin cuantizarea acestora prin intermediul celor N vectori de reproducere, se pot determina parametrii n_s şi n_d pentru $v_1=a_1$. Apoi, prin cuantizarea vectorilor care au $x_1=a_1+\Delta_1$, adică

$$\begin{cases} a_1 + \Delta_1, a_2 \\ a_1 + \Delta_1, a_2 + \Delta_2 \\ \vdots \\ a_1 + \Delta_1, b_2 \end{cases} \tag{4.7}$$

se determină parametrii n_s şi n_d pentru $v_1=a_1+\Delta_1$. În continuare, în acest fel, x_1 creşte cu Δ_1 în timp ce x_2 ia valori în toată gama admisibilă, până când se ajunge la vectorii cu $x_1=b_1$, adică

$$\begin{cases} b_1, a_2 \\ b_1, a_2 + \Delta_2 \\ \vdots \\ b_1, b_2 \end{cases} \tag{4.8}$$

Prin cuantizarea acestor vectori se determină parametrii n_s și n_d pentru $v_1=b_1$. În acest fel s-au cuantizat toți vectorii din figura 4.8. Din figură se poate observa că în același timp s-au generat și cuantizat vectorii care au $x_2=a_2$, $x_2=a_2+\Delta_2$, $x_2=a_2+2\Delta_2$, ..., $x_2=b_2$, de fiecare dată x_1 luând valori în întreaga gamă admisibilă. Astfel se pot determina parametrii n_s și n_d pentru $v_2=a_2$, pentru $v_2=a_2+\Delta_2$ și așa mai departe, pentru $v_2=b_2$. În concluzie, prin cuantizarea vectorilor uniform distribuiți în porțiunea mărginită din $\Re^2$, se determină parametrii n_s și n_d pentru valorile ambelor componente (k=2), valori distribuite uniform în gama admisibilă.

Acest rezultat se poate explica și în felul următor: considerând că se cuantizează un vector (x'_1, x'_2) oarecare din porțiunea mărginită, înainte sau după cuantizarea acestui vector s-au cuantizat sau se vor mai cuantiza vectori care au $x_1=x'_1$ și x_2=variabil, sau x_1=variabil și $x_2=x'_2$; pe baza rezultatelor obținute prin cuantizarea lui (x'_1, x'_2) și a celorlalți vectori amintiți, se determină regiunile de cuantizare care intersectează dreptele $x_1=x'_1$ și $x_2=x'_2$ și apoi a parametrilor n_s și n_d pentru $v_1=x'_1$ și $v_2=x'_2$.

Această idee se poate aplica în același mod și pentru valori ale lui k mai mari decât 2. Astfel, pentru vectori de forma $(x_1, x_2,...,x_k)$, fiecare componentă se poate scrie sub forma

$$x_i = a_i + q\Delta_i; q = 0,1,2,\ldots; i = 1,2,\ldots,k. \tag{4.9}$$

Considerând că fiecare componentă din cele k poate avea n_{mediu} valori egal depărtate unele de altele, se poate scrie că

$$b_i = a_i + (n_{mediu} - 1)\Delta_i; i = 1,2,\ldots,k. \tag{4.10}$$

În acest fel, se dorește determinarea parametrilor n_s și n_d pentru kn_{mediu} valori (câte n_{mediu} valori pentru fiecare din cele k componente).

Pentru determinarea acestor kn_{mediu} perechi de parametri, se creează kn_{mediu} tablouri (zone de memorie), fiecare cu N locații. După cuantizarea fiecărui vector uniform distribuit, $(x_1, x_2, ..., x_k)$, se vor face înscrieri în k din cele kn_{mediu} tablouri, la locația ii, unde ii reprezintă codul obținut prin

codarea vectorului $(x_1, x_2, ..., x_k)$, *ii* putând lua orice valoare de la 1 până la N. Aceste înscrieri se vor face în tablourile corespunzătoare valorilor de forma $a_i+q\Delta_i$ pe care le are fiecare din cele k componente x_i ale vectorului respectiv.

De exemplu, dacă inițial toate locațiile conțin valoarea 0, se poate înscrie valoarea 1, la locația *ii*, în fiecare din cele k tablouri.

Pe parcursul cuantizării fiecăruia din cei $(n_{mediu})^k$ vectori uniformi distribuiți, fiecare din cele kn_{mediu} valori se va regăsi de n_{mediu} ori printre componentele vectorilor uniformi, producând înscrierea valorii 1 la locația *ii* din tabloul propriu. (Se poate scrie valoarea 1 de mai multe ori în aceeași locație, deoarece mai mulți vectori, pentru care una dintre componente este identică, se pot cuantiza prin același vector de reproducere).

După încheierea acestui proces de cuantizare și înscriere în memorie, va urma un proces de analiză a acestor tablouri, în scopul determinării parametrilor n_s și n_d pentru fiecare din cele kn_{mediu} valori.

Se va începe cu componenta i=1, primul tablou fiind cel corespunzător valorii a_1. Se vor analiza toate locațiile j, j =1,...,N, în felul următor (se subliniază că în aceste locații s-au făcut înscrieri după cuantizarea tuturor vectorilor care au $x_1=a_1$):

- dacă la o locație j se găsește valoarea 1, înseamnă că regiunea de cuantizare a acelui vector de reproducere intersectează dreapta $x_1=a_1$, adică vectorul de reproducere corespunzător $\boldsymbol{y}_j$, va face parte din ambele grupe.

- dacă la o locație j se găsește valoarea 0, înseamnă că regiunea de cuantizare a acelui vector de reproducere nu intersectează dreapta $x_1=a_1$; dacă $y_{j1}\le a_1$, vectorul de reproducere $\boldsymbol{y}_j$ va face parte din grupa cu n_s elemente, iar dacă $y_{j1}>a_1$, din grupa cu n_d elemente (y_{j1} este prima componentă a vectorului de reproducere k-dimensional $\boldsymbol{y}_j$).

Apoi procesul continuă asemănător pentru celelalte n_{mediu}-1 valori corespunzătoare pentru i=1, după care se repetă aceleași operații pentru i=2, 3, ..., k.

Pentru a se face o comparație a metodei intersecțiilor, propusă de autor, cu metoda proiecțiilor, prezentă în literatură, din punct de vedere al numărului de operații, trebuie remarcat că în ambele metode se generează

același număr de vectori uniform distribuiți, astfel încât pentru cuantizarea acestora se face același număr de operații. Deosebirile se vor arăta în continuare. La metoda intersecțiilor, după cuantizarea fiecărui vector, se fac k înscrieri în memorie, deci în total se fac

$$m' = N_{mediu} N k \tag{4.11}$$

înscrieri, iar apoi, în cadrul procesului de analiză a tablourilor din memorie se fac

$$m'' = n_{mediu}(N + 0{,}7N)k \tag{4.12}$$

comparații. Termenul $0{,}7N$ se justifică prin aceea că, în general (așa cum se va arăta ulterior), aproximativ 70% din regiunile de cuantizare nu intersectează dreapta $x_i=a_i+q\Delta_i$, deci trebuie făcută încă o comparație pentru a se determina în care grupă intră vectorii corespunzători acelor regiuni de cuantizare.

Revăzând relațiile (4.4) și (4.5) rezultă că la metoda proiecțiilor se fac $n_{c2}+n_{c3}$ comparații,

$$n_{c2} + n_{c3} = 2NN_{mediu}\, k + 2Nn_{mediu}\, k. \tag{4.13}$$

Pentru metoda intersecțiilor se consideră de asemenea comparațiile (deci nu se consideră înscrierile în memorie), astfel că, dacă se compară relațiile (4.12) și (4.13) se observă s-a obținut o evidentă reducere a numărului de operații matematice.

Pentru un caz practic, N=256 k=3, n_{mediu}=50, se obține

$$N_{mediu} = \frac{(n_{mediu})^k}{N} = \frac{50^3}{256} \approx 500.$$

Rezultă $n_{c2}+n_{c3}$=2x256x500x3+2x256x50x3≈845000, respectiv m''=50x3x(256+0,7x256)≈65250, adică o reducere de aproximativ 12 ori a numărului de operații matematice pentru metoda intersecțiilor.

O altă deosebire între cele două metode este în ce privește valorile componentelor pentru care se determină parametrii n_s și n_d. La metoda proiecțiilor, acești parametri se pot determina pentru valori independente de componentele vectorilor uniform distribuiți cuantizați, în timp ce la metoda intersecțiilor parametrii se determină numai pentru valori identice cu componentele vectorilor uniform distribuiți cuantizați. Acest fapt nu constituie însă un dezavantaj pentru metoda intersecțiilor, deoarece în ambele cazuri numărul vectorilor uniform distribuiți trebuie să fie suficient de mare pentru a asigura precizia necesară.

4.3.3 Aplicarea metodei intersecțiilor pentru structurarea unei cărți de cod la un cuantizor vectorial pentru parametri LSF

În capitolul 3 autorul a realizat cuantizarea vectorială a parametrilor LSF folosind metoda prin despicare. În acest sens au fost proiectate 3 cărți de cod, fiecare cu N=256 vectori de reproducere, câte una pentru fiecare din cei 3 subvectori (care au avut dimensiunile 3, 3 și 4). În procesul de codare, pentru fiecare subvector s-a făcut o căutare totală în cartea de cod aferentă. În scopul reducerii complexității, autorul dorește realizarea codării pentru fiecare din cei 3 subvectori prin metoda combinată arbore-căutare totală. Pentru aceasta, trebuie structurate cele trei cărți de cod, autorul realizând acest lucru folosind metoda intersecțiilor.

În tabelul 4.1 se prezintă gama de variație a fiecăruia din cei 10 parametri LSF, așa cum rezultă prin calcul pe baza secvenței de antrenament de 2790 vectori.

Tabelul 4.1: *Gama de variație a parametrilor* LSF.

Parametrul LSF	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
Gama de variație	141	223	429	751	925	1117	1684	2289	2558	2890
$(f_{min}; f_{max})$ [Hz]	576	905	1425	1768	2086	2444	2765	3101	3382	3721

Conform acestor intervale de variaţie, numărul vectorilor uniform distribuiţi care trebuie generaţi şi cuantizaţi ar fi foarte mare. Bazându-se pe două observaţii, autorul arată că nu trebuie generaţi toţi vectorii posibili.

Prima observaţie este că, perechea (j, v_j) (componenta şi valoarea cu care se compară aceasta) conform căreia vectorii de reproducere asociaţi unui nod al arborelui se împart în cele două grupe, trebuie aleasă astfel încât parametrii n_s şi n_d care se obţin să aibe valori apropiate. Această restricţie a fost impusă deoarece se doreşte ca arborele după care se va face codarea să aibe ramuri egale, sau, numărul de noduri să se dubleze pe fiecare nivel faţă de nivelul precedent. Dacă n_s şi n_d ar avea valori mai diferite, după câteva niveluri, unele grupe vor avea un număr foarte mic de vectori de reproducere asociaţi, făcând imposibilă continuarea procesului de împărţire. În acest fel unele niveluri vor conţine atât noduri terminale cât şi noduri interne, unde procesul de împărţire în câte două grupe va continua. Acest lucru nu este de dorit, deoarece procesul de codare ar fi mai neregulat, în funcţie de calea urmată prin arbore, urmând să se ajungă într-un nod terminal după un număr diferit de comparaţii. Aşa cum va rezulta şi din exemplele următoare, pentru ca n_s şi n_d să aibe valori apropiate, valoarea v_j trebuie să fie centrală faţă de intervalul respectiv de variaţie al parametrului LSF.

A doua observaţie susţine că expresia $n'_{dist}=p_s n_s+p_d n_d$, care trebuie minimizată la alegerea perechii (j, v_j), are valori minime în zona centrală a intervalului în care fiecare componentă ia valori. Acest fapt rezultă din datele prezentate în tabelul 4.2, unde se arată valoarile calculate ale expresiei $n'_{dist}=p_s n_s+p_d n_d$ pentru componentele f_1, f_2, f_3, în funcţie de câteva valori, egal depărtate, din intervalul de variaţie al fiecărei componente. În toate cele trei cazuri s-a considerat întreaga carte de cod, cu 256 vectori de reproducere, adică s-au împărţit vectorii de reproducere asociaţi nodului rădăcină.

Din tabelul 4.2 rezultă mai multe concluzii:

- valoarea expresiei n'_{dist} care trebuie minimizată are variaţii mai mari de la o componentă la alta decât în cadrul aceleiaşi componente, în funcţie de diferitele valori ale lui f_i, $i=1, 2, 3$.

Tabelul 4.2: *Valori ale expresiei* n'_{dist}.

Valori f_1[Hz]	p_s / p_d	n_s / n_d	n'_{dist}	Valori f_2[Hz]	p_s / p_d	n_s / n_d	n'_{dist}	Valori f_3[Hz]	p_s / p_d	n_s / n_d	n'_{dist}
248	0,48 0,51	153 209	181,7	410	0,41 0,58	92 203	156,8	740	0,34 0,63	79 210	164,6
250	0,50 0,49	157 205	180,9	420	0,44 0,55	96 197	152,2	750	0,37 0,62	85 207	161,6
252	0,52 0,47	158 204	180	430	0,46 0,53	99 190	147,8	760	0,40 0,59	97 206	162,4
254	0,53 0,46	160 202	179	440	0,48 0,51	103 187	146	770	0,43 0,56	100 201	157
256	0,55 0,44	163 200	179	450	0,50 0,49	107 180	143	780	0,46 0,53	105 195	153
258	0,56 0,43	164 198	178	460	0,52 0,47	110 177	141	790	0,49 0,50	109 192	150
260	0,58 0,41	166 196	178	470	0,54 0,45	114 177	142	800	0,52 0,47	121 188	152
262	0,59 0,40	169 196	180	480	0,56 0,43	120 173	143	810	0,55 0,44	126 185	152
264	0,60 0,39	169 194	178	490	0,57 0,42	129 168	145	820	0,58 0,41	130 173	147
266	0,61 0,38	172 192	179	500	0,59 0,40	132 162	144	830	0,60 0,39	135 169	148,3
268	0,62 0,37	174 188	179	510	0,60 0,39	133 155	141	840	0,63 0,36	144 165	151,7
270	0,64 0,35	175 186	178	520	0,62 0,37	138 152	143	850	0,66 0,33	154 159	155,7
272	0,65 0,34	178 180	178,8	530	0,63 0,36	140 152	144,4	860	0,68 0,31	159 152	156,8
274	0,66 0,33	179 178	178,7	540	0,65 0,34	145 149	146,4	870	0,70 0,29	163 141	156,6
276	0,66 0,33	181 176	179,3	550	0,66 0,33	149 142	146,7	880	0,73 0,26	171 133	161

- intervalele în care vectorii de antrenament au ambele probabilități semnificative (adică p_s sau p_d mai mari decât 0,25÷0,30), sunt mult mai mici decât intervalele în care aceeași vectori iau valori (tabelul 4.1)

- în situația în care $p_s \cong p_d$, n_s și n_d sunt mult diferite și invers, când $n_s \cong n_d$, p_s și p_d sunt mult diferite.

- valoarea expresiei n'_{dist} are variații în general mici la variațiile specificate în tabelul 4.2 pentru parametrii f_i (2Hz pentru f_1, 10 Hz pentru f_2 și f_3)

Pe baza celor două observații precum și a concluziilor rezultate din analiza rezultatelor prezentate în tabelul 4.2, autorul a adaptat algoritmul general de structurare a cărții de cod în vederea codării prin metoda arbore-căutare totală pentru a fi aplicat vectorilor care conțin parametri LSF.

Înainte de prezentarea structurii complete a acestui algoritm, se prezintă modul de determinare a parametrilor n_s și n_d pentru un nod, care se va aplica apoi în orice nod al arborelui.

Mai întâi s-a determinat valoarea v_j pentru fiecare din cele 3 (sau 4) componente ale vectorilor. Ideea de bază rezultată din cele două observații anterioare este că trebuie generați și cuantizați numai acei vectori uniform distribuiți pentru care parametrii n_s și n_d au valori apropiate.

De exemplu, în cazul vectorilor de forma (f_1, f_2, f_3), pentru determinarea valorii v_3, trebuie generați vectori în care f_3 ia câteva valori egal depărtate într-un interval limitat, iar f_1 și f_2 iau valori egal depărtate în toată gama admisibilă conform tabelului 4.1. Intervalul limitat este acela pentru care parametrii n_s și n_d au valori apropiate. În final, dintre toate valorile lui f_3, v_3 va fi aceea pentru care valoarea expresiei n'_{dist} este minimă.

Foarte important a fost să se determine care este valoarea lui f_3 de la care începe intervalul limitat. În acest scop s-a început de la o valoare a lui f_3 situată în prima parte a gamei de variație din tabelul 4.1, determinându-se p_s și p_d, adică probabilitatea ca un vector de antrenament să aibe componenta a 3-a mai mică sau mai mare decât valoarea respectivă a lui f_3. Acest procedeu se repetă, crescând fin valoarea lui f_3 până când se obține o valoare a lui p_s de 0,4. Începând cu această valoare a lui f_3 se generează vectorii uniform distribuiți pentru care se calculează valorile expresiei n'_{dist}. Pentru determinarea valorii inițiale a lui f_3, s-a folosit calculul probabilităților p_s și p_d deoarece acesta necesită mai puține operații decât calculul lui n_s și n_d. Valoarea p_s=0,4 s-a ales ținând cont de rezultatele prezentate în tabelul 4.2, și anume că, începând de la această probabilitate,

parametrii n_s şi n_d încep să aibe valori comparabile. Desigur, valorile lui n_s şi n_d pentru p_s=0,4 diferă de la componentă la componentă (aşa cum se vede şi în tabelul 4.2) precum şi de la un nod al arborelui la altul. Aceasta a fost însă o soluţie de compromis care să permită generarea, în ansamblu, a cât mai puţini vectori uniformi. Legătura între p_s şi, respectiv, n_s şi n_d este doar o constatare experimentală, având în vedere că p_s şi p_d se referă la vectorii de antrenament, iar n_s şi n_d la vectorii de reproducere, care însă acoperă aceeaşi zonă a spaţiului k-dimensional.

Începând de la valoarea lui f_3 pentru care p_3=0,4, se calculează parametrii n_s şi n_d, valoarea lui f_3 crescând apoi cu câte 10 Hz, până când între parametrii n_s şi n_d există relaţia n_s/n_d=0,9. Începând de la această valoare, se face o modificare mai fină a lui f_3, şi anume cu 5 Hz, pentru ca raportul n_s/n_d să crească mai lent, rămânând cât mai apropiat de 1. Se mai generează în total vectori uniformi în care f_3 ia 3 valori decalate cu 5 Hz, începând de la aceea pentru care n_s/n_d=0,9. Dintre aceste 3 valori, valoarea pentru care expresia n'_{dist} are valoarea minimă este valoarea v_3.

În continuare acest procedeu se repetă şi pentru determinarea valorilor v_2 şi v_1, ale componentelor f_2 şi f_1, din cadrul vectorului (f_1, f_2, f_3). Deoarece f_1 are o gamă dinamică mai mică, după determinarea valorii pentru care p_s=0,4, f_1 se creşte cu 4 Hz, iar din momentul în care n_s/n_d=0,9, cu 2 Hz. În final, fiecare componentă având o valoare "candidată", se determină pentru care din aceste componente expresia n'_{dist} are valoarea minimă.

În concluzie, faţă de forma generală a metodei intersecţiilor, care necesită generarea şi cuantizarea unui număr foarte mare de vectori, autorul a propus o variantă simplificată, în scopul reducerii numărului de vectori generaţi şi cuantizaţi, simplificarea fiind posibilă datorită modului în care sunt distribuiţi parametrii LSF. Dacă la forma generală a metodei, parametrii n_s şi n_d pentru oricare din valorile posibile ale tuturor celor k componente se determinau după generarea o singură dată a tuturor vectorilor uniform distribuiţi, în varianta simplificată, se generează pentru fiecare componentă în parte alţi vectori, aşa cum se arată în figura 4.9, pentru cazul k=2.

În figura 4.9 se arată vectorii generaţi pentru determinarea parametrilor n_s şi n_d, pentru componenta 1, în care valorile fixe ale lui f_1 (3 valori), sunt decalate cu valoarea Δ'_1, iar componenta f_2 ia valori decalate cu valoarea Δ_2,

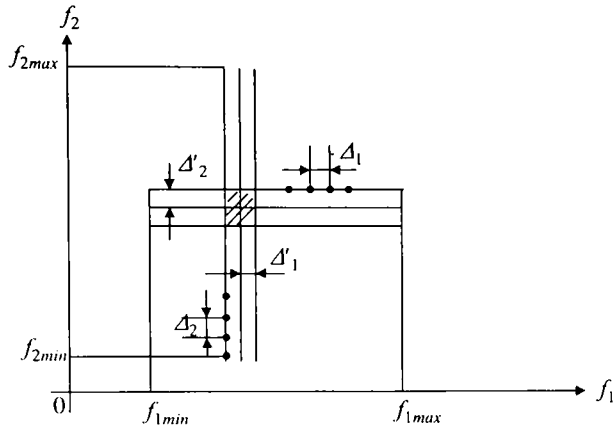


Figura 4.9: *Vectorii uniform distribuiți în varianta simplificată a metodei intersecțiilor.*

în intervalul (f_{2min}, f_{2max}). În mod asemănător se arată vectorii folosiți pentru determinarea parametrilor n_s și n_d pentru componenta 2. Tot din figura 4.9 se observă că unii vectori sunt generați de două ori (cei din porțiunea hașurată), dar aceștia sunt în număr foarte mic. Pe ansamblu, numărul vectorilor generați în varianta simplificată a metodei intersecțiilor este mult mai mic, față de forma generală, când ar fi trebuit să se genereze vectori care să acopere dreptunghiul (f_{1min}, f_{1max}), (f_{2min}, f_{2max}).

De asemenea, se mai impun câteva precizări asupra valorilor pașilor, Δ_1, Δ_2, respectiv, Δ'_1 și Δ'_2. În forma generală a metodei intersecțiilor Δ_1 era implicit egal cu Δ'_1, iar Δ_2 egal cu Δ'_2 (vezi figura 4.8). În forma simplificată a metodei $\Delta_1 \neq \Delta'_1$, deoarece Δ_1 reprezintă pasul cu care se modifică f_1, pentru vectorii care au f_2 constant, acești vectori fiind necesari pentru determinarea regiunilor de cuantizare care intersectează dreapta f_2=constant. Δ'_1 reprezintă pasul cu care se modifică f_1 într-un interval îngust, dintre valorile respective ale lui f_1, determinându-se cea pentru care mărimea n'_{dist} are valoare minimă, notată v_1. În mod similar și Δ_2 este diferit de Δ'_2.

O mare importanță pentru realizarea cât mai corectă a procesului de structurare a cărții de cod o are modul în care iau valori componentele care se modifică în toată gama admisibilă (de exemplu f_1 și f_2, când f_3 este fix). Mai exact, în descrierea generală făcută anterior s-a spus că acele

componente vor fi distanţate unele de altele cu un pas, Δ_i. De valoarea acestui pas depinde cât de corect se vor determina regiunile care intersectează dreapta $x_j=v_j$.

În figura 4.10 se arată un exemplu, din spaţiul k=2 care întăreşte această afirmaţie. Astfel, sunt arătaţi 4 vectori de reproducere, notaţi 1, 2, 3, 4 şi se doreşte să se determine care dintre regiunile de cuantizare ale acestor vectori intersectează dreapta $f_1=v_1$. Pentru aceasta se generează 4 vectori în care prima componentă are valoarea v_1, iar a doua componentă are valori egal depărtate cu Δ_1,

$$\begin{cases}(v_1, a_1)\\(v_1, a_1 + \Delta_2)\\(v_1, a_1 + 2\Delta_2)\\(v_1, a_1 + 3\Delta_2).\end{cases} \tag{4.14}$$

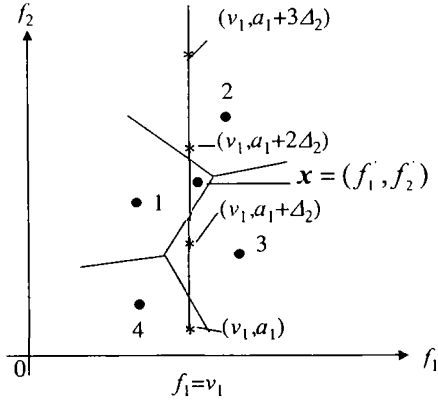


Figura 4.10: *Explicativă pentru determinarea intersecţiilor cu dreapta* $f_1=v_1$.

Din figură se observă că, deşi dreapta $f_1=v_1$ intersectează regiunea de cuantizare a vectorului 1, nici unul din cei 4 vectori generaţi nu se cuantizează prin vectorul de reproducere 1. Cauza pentru care apare această inexactitate este valoarea prea mare a pasului Δ_2, deoarece, din figură se observă cu uşurinţă că o valoare mai mică a pasului Δ_2 ar permite determinarea şi a vectorului de reproducere 1 printre cei ale căror regiuni

intersectează dreapta $f_1=v_1$. În aceste condiții, la cuantizarea vectorului $x=(f_1', f_2')$, considerând că testul se face în raport cu valoarea v_1 a componentei 1, acest vector va fi comparat cu vectorii de reproducere ale căror regiuni intersectează dreapta $f_1=v_1$ (adică 4, 3 și 2) sau cu cei care au componenta $f_1>v_1$, dar regiunile lor nu intersectează dreapta $f_1=v_1$ (nu este cazul). Prin urmare, vectorul $x=(f_1', f_2')$ va fi cuantizat prin vectorul de reproducere 2 și nu prin 1, din a cărui regiune de cuantizare face parte, producându-se o creștere a erorii de cuantizare.

Modul cel mai simplu și totodată cel mai sigur de alegere a pasului Δ_i este stabilirea acestuia la valoarea de 1 Hz, valoarea minimă prin care pot diferi între ele componentele vectorilor. În această situație inconvenientul ar fi din nou numărul mare de vectori uniform distribuiți care trebuie generați și cuantizați, prelungind mult timpul necesar structurării cărții de cod. Pentru a se vedea dacă, în practică, pasul poate avea o valoare mai mare decât 1 Hz, se poate proceda în modul următor: se determină numărul regiunilor de cuantizare care intersectează o dreaptă $f_i=v_i$, în situația în care pasul Δ_j , $j\neq i$ are valoarea de 1 Hz; apoi se crește progresiv acest pas (2 Hz, 3 Hz,...,) până când numărul regiunilor care intersectează dreapta scade. Cea mai mare valoare a pasului pentru care numărul de regiuni rămâne același ca pentru pasul de 1 Hz, este cea optimă. Desigur că această metodă este pur experimentală, neputându-se justifica teoretic valoarea care se obține.

O altă soluție, propusă de asemenea de autor, își propune să obțină o valoare pentru pasul Δ_i, estimând dimensiunile regiunilor de cuantizare. În acest sens se determină pentru fiecare vector de reproducere component al cărții de cod, distanța (euclidiană) până la cel mai apropiat vector de reproducere, notată d_{min}. Apoi se calculează media aritmetică a acestor distanțe pentru cei N vectori ai cărții de cod respective, notată d_{med}. Mărimea obținută are semnificația unei distanțe medii între oricare 2 vectori de reproducere învecinați. În tabelul 4.3 se arată aceste distanțe pentru cele 3 cărți de cod (cele care conțin vectorii cu componentele $f_1,...,f_3,f_4,...,f_6$ și $f_7,...f_{10}$), precum și numărul vectorilor pentru care distanța până la cel mai

apropiat vector de reproducere este mai mică decât o anumită valoare, inferioară lui d_{med}.

Tabelul 4.3: *Valori ale parametrului* d_{med} .

Cartea de cod	$f_1 \dots f_3$	$f_4 \dots f_6$	$f_7 \dots f_{10}$
d_{med}	47,1	75,7	83,9
Număr de vectori	18 cu d_{min}< 30	8 cu d_{min}< 50	7 cu d_{min}< 40

Dacă vectorii de reproducere ar fi uniform distribuiţi, regiunile de cuantizare ar avea forma unor pătrate (pentru k=2) sau cuburi (pentru k=3), cu laturile egale chiar cu distanţa d_{med}. În acest caz, o valoare a lui Δ_i egală cu d_{med} ar permite determinarea corectă a regiunilor care intersectează dreapta f_i=v_i. Deoarece regiunile de cuantizare au forme mult mai complicate, este evident că pasul Δ_i trebuie ales mai mic decât d_{med}. Pe baza datelor din tabelul 4.3, inclusiv numărul vectorilor cu distanţa d mult mai mică decât d_{med}, autorul propune ca pasul Δ_i să aibe o valoare cuprinsă în intervalul $(0{,}3 \dots 0{,}5)d_{med}$.

Rezultatele obţinute experimental pentru câteva valori ale pasului Δ_i, care vor confirma valorile propuse de autor, se vor arăta ulterior, după prezentarea organigramei algoritmului de structurare şi a organigramei celui de codare.

Înainte de prezentarea acestor organigrame se introduc unele notaţii necesare pentru buna înţelegere (unele notaţii au fost deja prezentate).

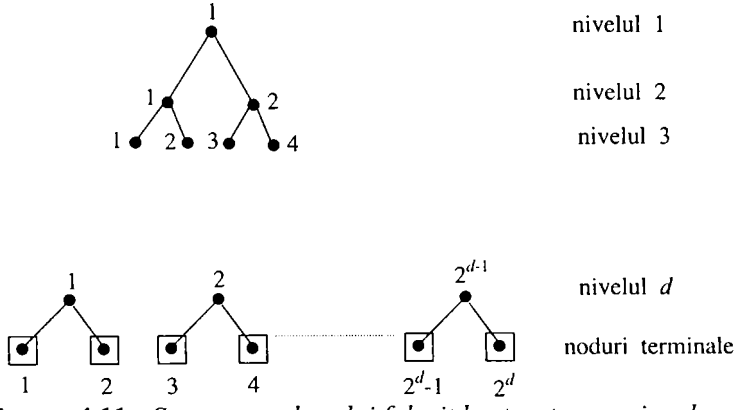
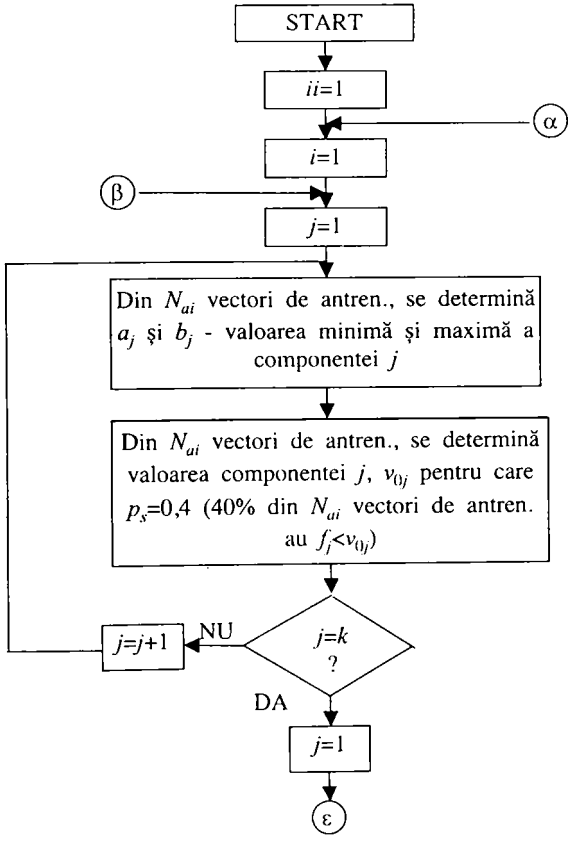
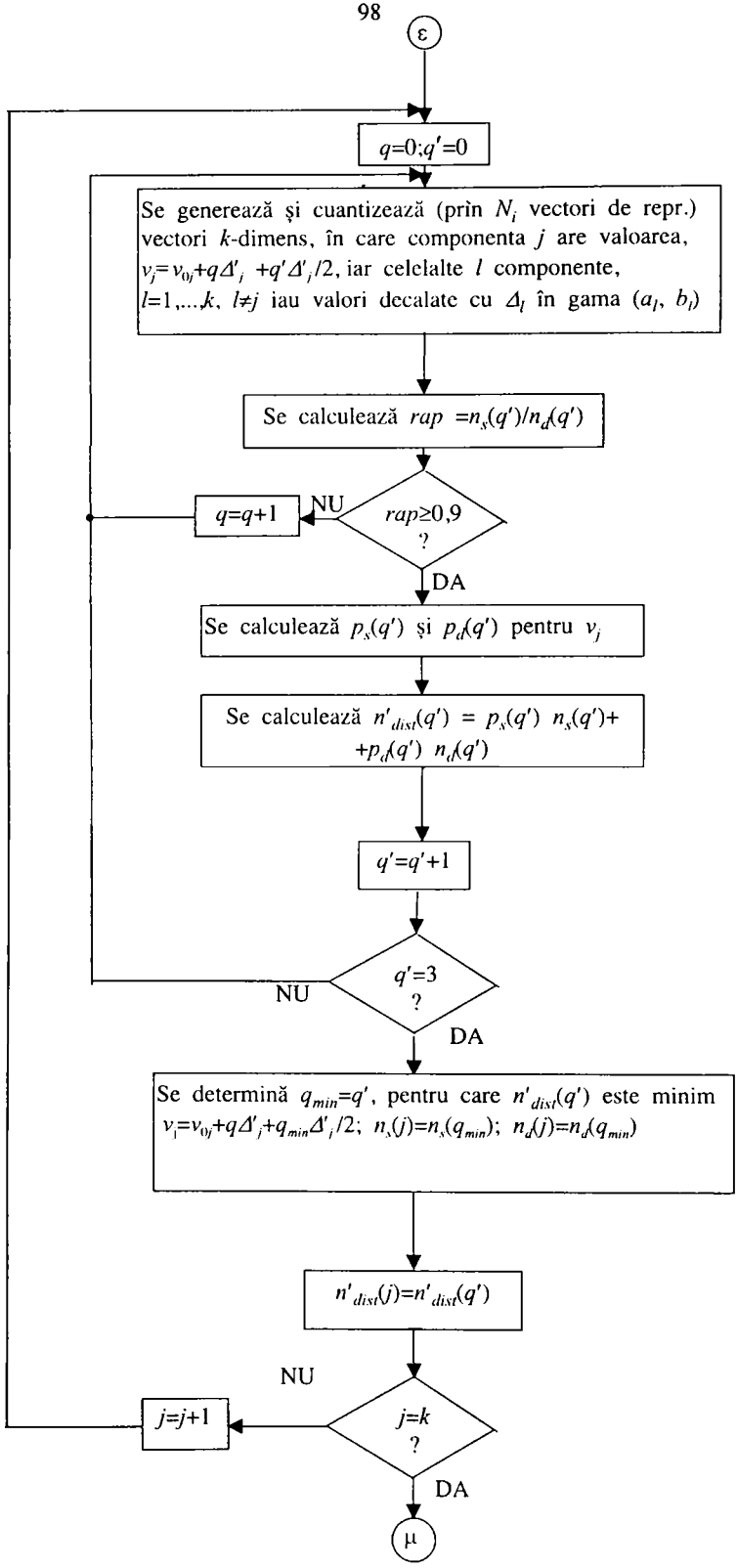


Figura 4.11: *Structura arborelui folosit la structurare şi codare.*

Din figura 4.11 se vede că arborele după care se face atât structurarea cât şi codarea conţine d niveluri, nivelul curent fiind notat cu ii, ii=1,..., d. Pe fiecare nivel există 2^{ii-1} noduri interne, nodul curent fiind notat i, i=1, ..., 2^{ii-1}. Arborele mai conţine $2(2^{d-1})=2^d=m$ noduri terminale. Fiecărui nod intern îi sunt asociaţi N_i vectori de reproducere şi N_{ai} vectori de antrenament (nodului 1 de pe nivelul 1 îi sunt asociaţi $N_1=N$ vectori de reproducere, adică toţi vectorii de reproducere din cartea de cod, şi $N_{a1}=n$ vectori de antrenament, adică toţi vectorii din secvenţa de antrenament). În general dintr-un nod i, situat pe nivelul ii, se ajunge în nodurile $2i$-1 şi $2i$ de pe nivelul următor, ii+1. Vectorii de antrenament k dimensionali sunt de forma $(f_1, f_2, ..., f_k)$, k=3 sau k=4.

În figura 4.12 se prezintă organigrama programului (vezi Anexa 2) care implementează algoritmul de structurare a cărţii de cod. Arborele are d=6 niveluri, adică o valoare care permite aşa cum se va arăta, o reducere suficientă a complexităţii, nodurile terminale având asociate valori suficient de mici pentru parametrii n_i, numărul vectorilor de reproducere asociaţi.




ε
q=0;q'=0
Se generează și cuantizează (prîn N_i vectori de repr.) vectori k-dimens, în care componenta j are valoarea, $v_j=v_{0j}+q\Delta'_j+q'\Delta'_j/2$, iar celelalte l componente, $l=1,...,k$, $l\neq j$ iau valori decalate cu Δ_l în gama (a_l, b_l)
Se calculează $rap=n_s(q')/n_d(q')$
$rap\geq 0,9$?
NU
q=q+1
DA
Se calculează $p_s(q')$ și $p_d(q')$ pentru v_j
Se calculează $n'_{dist}(q') = p_s(q')\ n_s(q')+ +p_d(q')\ n_d(q')$
q'=q'+1
q'=3 ?
NU
DA
Se determină $q_{min}=q'$, pentru care $n'_{dist}(q')$ este minim $v_j=v_{0j}+q\Delta'_j+q_{min}\Delta'_j/2$; $n_s(j)=n_s(q_{min})$; $n_d(j)=n_d(q_{min})$
$n'_{dist}(j)=n'_{dist}(q')$
NU
j=k ?
j=j+1
DA
μ

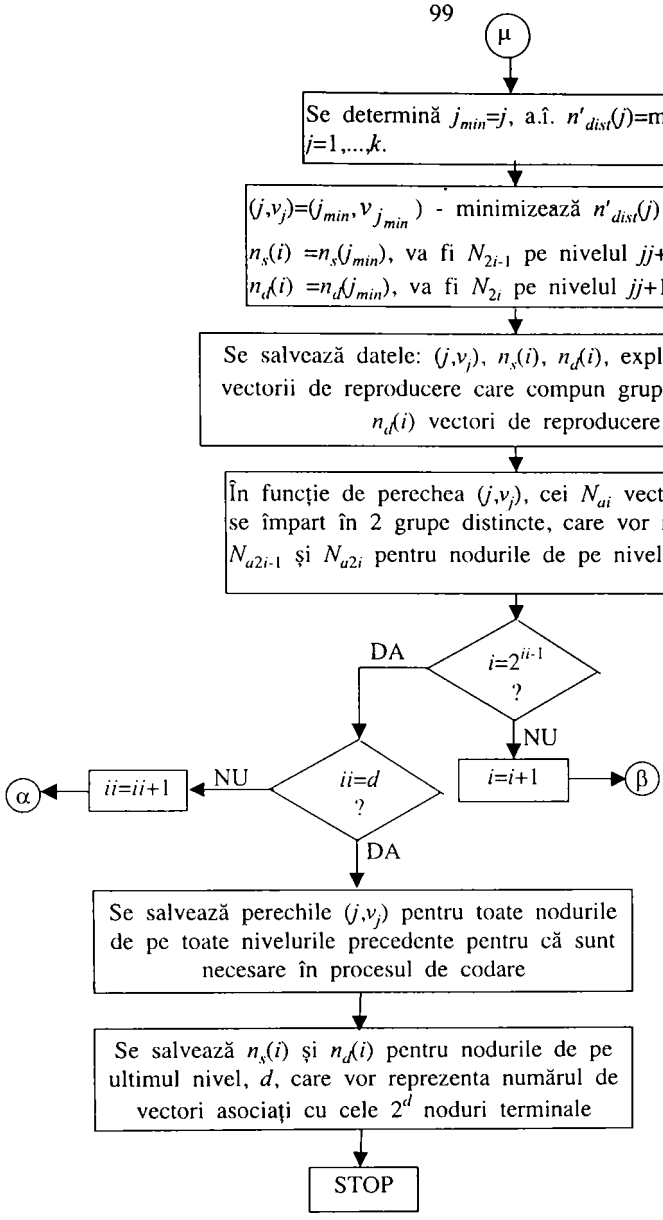


Figura 4.12: *Organigrama programului de structurare a cărții de cod.*

Programul care implementează acest algoritm, scris în limbajul C, a fost rulat separat pentru cele trei cărți de cod.

După determinarea perechii (j, v_j) în toate nodurile de pe un nivel, algoritmul se poate încheia, dacă s-au parcurs toate nivelurile dorite. De

aceea în momentul respectiv, trebuie să fie disponibile toate datele necesare care trebuie folosite în procesul de codare, în vederea memorării. Acestea sunt:

- perechile (j, v_j) pentru toate nodurile interne, memorate în tabloul comp[*ii*][*i*] pentru componenta j, respectiv, în tabloul val[*ii*][*i*] pentru valoarea v_j.

- numărul de vectori de reproducere n_i, asociat cu fiecare din cele m noduri terminale, care provin din parametri $n_s(i)$ și $n_d(i)$ ale celor $m/2$ noduri de pe nivelul d, precum și indicii (între 1 și N) acestor vectori.

Pe parcursul rulării programului nu sunt păstrați decât parametrii $n_s(i)$ și $n_d(i)$ și indicii care corespund nivelului precedent, cei de pe celelalte niveluri nemaifiind necesari. În acest fel se face o economie de memorie.

Pentru o mai clară imagine asupra rezultatelor procesului de structurare, în tabelul 4.4 se arată valorile parametrilor n_s, n_d precum și numărul de regiuni care intersectează un plan oarecare f_i=constant, i=1,2,3, notat n_{com}, care se obțin în nodul rădăcină (N_i=256).

Tabelul 4.4: *Parametrii n_s, n_d, n_{com} obținuți în nodul rădăcină.*

Parametrul LSF	f_1=265	f_2=520	f_3=845
n_s	171	138	148
n_d	186	152	163
n_{com}	56	36	102

Din tabelul 4.4 rezultă prin diferență față de 256 și numărul regiunilor de cuantizare care nu intersectează un aumit plan, adică 200, 220 și, respectiv 154. În funcție de aceste date și de alte date obținute în cursul rulării programelor dar neprezentate aici, s-a ales o valoare aproximativă de 0,7N (N=256) pentru numărul regiunilor de cuantizare care nu intersectează un aumit plan, această valoare fiind necesară la determinarea numărului de operații care se fac la metoda intersecțiilor.

Așa cum s-a mai spus, pasul Δ'_j are valoarea 4 Hz pentru f_1 și, respectiv, 10 Hz pentru celelalte componente. Pasul Δ_l, l=1,...,k are diverse

valori în intervalul (0,3 ... 0,5) d_{mediu}. După efectuarea codării se vor discuta rezultatele obţinute în funcţie de diversele valori ale pasului Δ_l.

În continuare, în tabelul 4.5 se arată pentru exemplificare, valorile celor 3 componente ale vectorilor (f_1, f_2, f_3) distribuiţi uniform, în situaţia când f_3 are o valoare fixă, de exemplu f_3=500 Hz. Prin cuantizarea acestor vectori s-au determinat regiunile de cuantizare care intersectează planul f_3=constant, (egal cu 500 Hz).

Tabelul 4.5: *Valori ale vectorilor uniform distibuiţi.*

141	223	500
141	243	500
141	483	500
161	223	500
161	483	500
241	243	500
241	263	500
241	483	500
461	483	500
481	483	500

Componentele f_1 şi f_2 au valorile iniţiale 141 şi respectiv 223 (aceste valori corespund limitelor inferioare ale lui f_1 şi f_2, tabelul 4.1). La început f_1 rămâne la valoarea de 141, iar f_2 creşte cu pasul Δ_2=20, până la o valoare apropiată de f_3, dar mai mică (483 faţă de 500). În continuare, f_1 creşte cu

pasul Δ_1=20 la valoarea de 161, în timp ce f_2 creşte în acelaşi fel. Când f_1 depăşeşte valoarea iniţială a lui f_2 (adică 223), a patra linie în tabelul 4.5, f_2 va avea o valoare iniţială mai mare, astfel încât să fie îndeplinită condiţia $f_1<f_2$. În continuare, odată cu creşterea lui f_1, f_2 va avea o gamă dinamică din ce în ce mai mică.

În acest fel se generează şi ceilalţi vectori de tipul (f_1, f_2, f_3), precum şi vectorii (f_4, f_5, f_6) şi, respectiv, (f_7, f_8, f_9, f_{10}). Condiţia $f_{i-1}< f_i< f_{i+1}$, i=2, i=5, şi, respectiv, i=8,9 trebuie îndeplinită la generarea tuturor vectorilor uniform distribuiţi.

4.4 Algoritmul de codare al cuantizării vectoriale prin metoda combinată arbore-căutare totală

Pe baza rezultatelor obţinute în procesul de structurare pentru cele 3 cărţi de cod s-a efectuat algoritmul de codare prin metoda combinată arbore-căutare totală, pentru o secvenţă de 310 vectori de test.

În figura 4.13 se prezintă organigrama programului care implementează acest algoritm, scris în limbajul C. Organigrama programului este descrisă pentru subvectorii cu componentele f_1, f_2, f_3, ea fiind similară şi pentru ceilalţi subvectori.

În fiecare nod al arborelui, i, de pe un nivel ii, se citesc mai întâi elementele perechii (j, v_j), iar apoi se face testul. În funcţie de rezultatul testului, se află valoarea nodului i de pe nivelul următor. Nodul i obţinut în urma testului de pe ultimul nivel d, va reprezenta nodul terminal. Apoi se va face o căutare totală printre cei n_i vectori de reproducere asociaţi nodului terminal i. Distanţa minimă obţinută (considerată ca distanţă euclidiană) reprezintă de fapt eroarea de cuantizare, indicele vectorului de reproducere faţă de care s-a obţinut distanţa minimă reprezintă codul vectorului de cuantizat, iar vectorul de reproducere reprezintă versiunea cuantizată a vectorului de cuantizat.

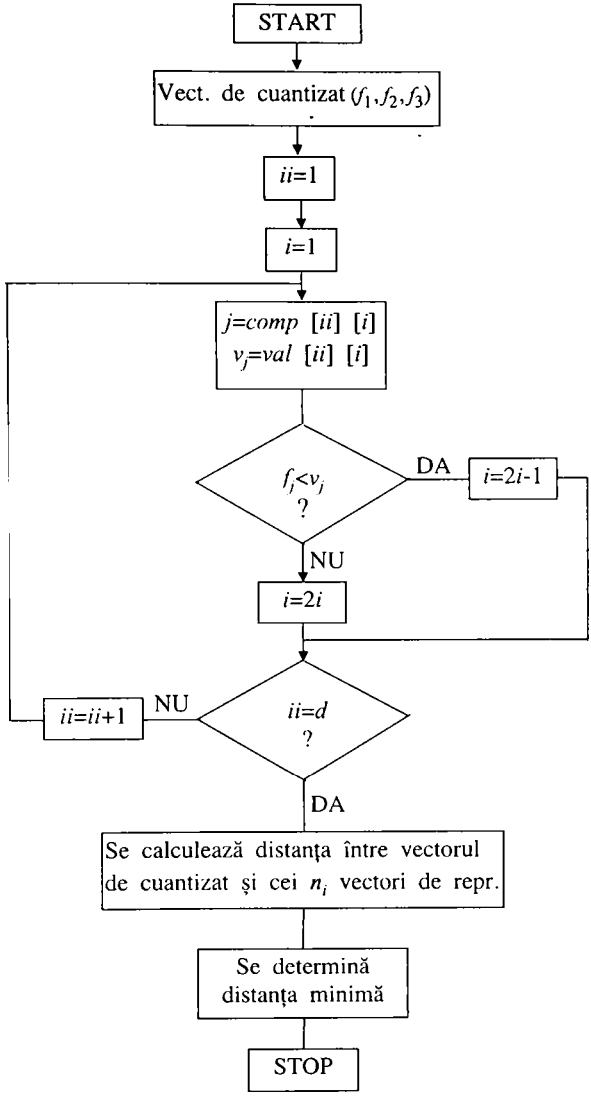


Figura 4.13: *Organigrama programului de codare prin metoda combinată arbore-căutare totală.*

În tabelul 4.6 se prezintă eroarea medie de cuantizare, notată $E_{a\text{-}ct}$, pentru diferite valori ale pasului Δ_i, i=1,...,k, folosite în partea de structurare a cărţii de cod, pentru fiecare din cele 3 tipuri de subvectori, comparativ cu eroarea medie notată E_{ct}, obţinută la cuantizarea acelorași vectori prin

căutare totală. În toate cele 3 cazuri, $\Delta_i=\Delta$, i=1,2,3, apoi i=4,5,6 și, respectiv, i=7,8,9,10.

Tabelul 4.6: *Erori medii obținute în funcție de valori ale pasului Δ.*

Subvectorii	f_1,f_2,f_3	f_4,f_5,f_6	f_7,f_8,f_9,f_{10}
E_{cl} [Hz^2]	705,56	2211,51	2823,75
$E_{a\text{-}cl}$ [Hz^2]	705,56 pt. Δ=20 Hz	2211,51 pt. Δ=35 Hz	2823,75 pt. Δ=30 Hz
$E_{a\text{-}cl}$ [Hz^2]	706,91 pt. Δ=25 Hz	2212,05 pt. Δ=40 Hz	2235,39 pt. Δ=35 Hz

Din tabel se observă că pentru fiecare tip de subvector, o anumită valoare a pasului Δ permite obținerea aceleiași erori medii ca și la căutarea totală. Creșterea valorii pasului Δ peste valoarea respectivă provoacă creșterea erorii medii de cuantizare. Astfel, în cazul subvectorilor f_1, f_2, f_3, la 2 vectori din cei 310 s-a produs creșterea erorii de cuantizare, de la valoarea 1533 Hz^2 la 1634 Hz^2 (pentru primul) și respectiv de la 1509 Hz^2 la 1890 Hz^2 (pentru al doilea). În cazul subvectorilor f_4, f_5, f_6, la un vector a crescut eroarea de la 13556 Hz^2 la 13702 Hz^2, iar în cazul subvectorilor f_7, f_8, f_9, f_{10} eroarea a crescut de la 1846 Hz^2 la 3461 Hz^2 pentru un vector și, respectiv de la 7246 Hz^2 la 8294 Hz^2 pentru un alt vector.

Valorile pașilor Δ din tabelul 4.6 pentru care erorile obținute prin cele 2 metode sunt egale reprezintă niște valori de prag maxime care permit acest lucru. Desigur că pentru alți vectori de test este posibil să existe vectori de tipul x din figura 4.10, care produc creșterea erorii medii. Oricum, datorită valorilor mici ale pașilor Δ, acești vectori au probabilități minime de apariție.

Un alt experiment care confirmă în mare măsură valoarea propusă pentru pasul Δ constă în determinarea regiunilor de cuantizare care intersectează un plan în spațiul k=3 sau k=4, (adică vectorii care au f_i=constant), prin modificarea valorilor f_j, $j\neq i$, cu valori ale pasului Δ_j din ce în ce mai mici.

Tabelul 4.7: *Valori ale numărului de regiuni care intersectează un anumit plan.*

pas \ f_3	792	842	847	852
20	47	55	57	54
15	47	55	57	54
10	48	55	56	57
5	49	56	57	58
1	50	57	57	59

Astfel, în tabelul 4.7, se prezintă numărul regiunilor de cuantizare care intersectează planul specificat prin f_3=constant (s-au ales pentru f_3 valorile 792, 842, 847 și 852 Hz). Pentru determinarea acestor regiuni, componentele f_1 și f_2 au luat valori în toată gama admisibilă, distanțate între ele mai întâi prin valoarea Δ=20 Hz, adică acea valoare a pasului care, conform tabelului 4.6 permite obținerea unei erori identice cu cea obținută în cazul în care codarea s-a făcut prin căutare totală. Apoi, acest pas a fost redus treptat, până la valoarea minimă de 1 Hz. Se observă că, față de valoarea de 20 Hz, numărul regiunilor care intersectează planul crește odată cu micșorarea pasului, ceea ce înseamnă că, mai există, pentru fiecare valoare f_3, de la 2 până la 5 regiuni de cuantizare care nu sunt determinate ca făcând parte din cele care sunt comune celor două grupe. Acest lucru ar putea avea ca efect creșterea erorii medii de cuantizare, dacă în zonele respective sunt vectori de cuantizat. Având în vedere numărul relativ mic al acestor regiuni, creșterea posibilă a erorii medii de cuantizare este relativ mică.

Cea mai importantă consecință a aplicării cuantizării vectoriale prin metoda arbore-căutare totală este reducerea complexității de calcul în procesul de codare. Pentru o evaluare simplă și rapidă a complexității trebuie cunoscute valorile numărului de vectori de reproducere, n_i, asociate cu nodurile terminale. Valoarea medie a numărului de vectori, pentru fiecare din cele trei tipuri de subvectori (în fiecare caz, pentru cele 64 noduri terminale) reprezintă cu bună aproximare numărul de distanțe care se calculează în procesul de codare. Pentru o determinare mai exactă ar trebui aplicată relația (4.1) în funcție de probabilitățile vectorilor de test de a intra în unul din nodurile terminale.

Astfel, s-au obţinut următoarele valori pentru numărul mediu de vectori asociaţi nodurilor terminale: pentru subvectorii (f_1, f_2, f_3), $n_{m1,3}=15,60$; pentru subvectorii (f_4, f_5, f_6), $n_{m\,4,6}=16,32$; iar pentru (f_7, f_8, f_9, f_{10}), $n_{m\,7,10}=25,9$. Se observă că, odată cu creşterea dimensiunii vectorilor, valoarea medie a numărului de vectori asociaţi cu nodurile terminale creşte, sau numărul regiunilor de cuantizare care intersectează un plan creşte odată cu dimensiunea spaţiului. În acest fel, pentru codarea unui vector complexitatea are expresia

$$C = 3n_{m1,3} + 3n_{m4,6} + 4n_{m7,10} \tag{4.15}$$

adică

$$C = 15,60\text{x}3 + 16,32\text{x}3 + 25,9\text{x}4 \cong 200 \text{ înmulţiri}$$

Dacă pentru subvectorii (f_7, f_8, f_9, f_{10}) s-a făcut o structurare şi respectiv o codare pe 7 niveluri, numărul mediu de vectori asociaţi celor 128 noduri terminale a scăzut la valoarea de 17,57, ceea ce a avut ca efect scăderea numărului de înmulţiri la valoarea de 166.

Memoria necesară pentru un cuantizor vectorial de tipul arbore-căutare totală conţine spaţiul necesar pentru memorarea vectorilor de reproducere ale celor 3 cărţi de cod, precum şi un spaţiu pentru memorarea informaţiei obţinute în urma procesului de structurare: în primul rând indicii vectorilor de reproducere asociaţi cu fiecare dun cele 2^d noduri terminale (precum şi numărul de vectori n_i pentru fiecare din aceste noduri), iar apoi perechile pentru testare (j, v_j) din fiecare din cele 2^d-1 noduri interne. Astfel, memoria totală este

$$Mem = kN2 + 2^d(n_{m1,3} + n_{m4,6} + n_{m7,10}) + 3 \cdot 2^d \cdot 1 + 3 \cdot \\ \cdot (2^d - 1)(2 + 1) \tag{4.16}$$

adică

$$Mem = 10\text{x}256\text{x}2 + 2^6(15,60 + 16,32 + 25,9) + 3\text{x}2^6 + \\ + 3\text{x}(2^6 - 1)\text{x}3 \cong 9580 \; octeti$$

S-a considerat că fiecare componentă a vectorilor de reproducere se memorează pe 2 octeţi, la fel ca şi valoarea v_j, restul mărimilor necesitând câte un octet.

În cazul în care pentru subvectorii (f_7, f_8, f_9, f_{10}) s-a făcut o structurare pe 7 nivele memoria devine

$$Mem = k \cdot N \cdot 2 + 2^d (n_{m1,3} + n_{m4,6}) + 2^{d+1} n_{m7,10} + 2 \cdot 2^d \cdot 1 + 2^{d+1} \qquad (4.17)$$
$$+ 2(2^d - 1) \cdot (2 + 1) + (2^{d+1} - 1)(2 + 1)$$

adică,

M=10x256x2+2^6(15,6+16,3)+2^7x17,5+2x2^6+2^7+2(2^6-1)x3+3x(2^7-1)x3=10400

octeţi, deci, în acest fel, memoria a înregistrat o uşoară creştere.

O comparaţie a acestor rezultate cu cele obţinute prin alte metode din literatură se va face după prezentarea structurării şi codării folosind eroarea pătratică ponderată.

4.5 Realizarea structurării cărţii de cod şi a codării pentru cazul erorii pătratice ponderate

Structurarea celor 3 cărţi de cod în vederea aplicării cuantizării vectoriale prin metoda combinată arbore căutare-totală s-a efectuat pentru situaţia în care măsura erorii de cuantizare a fost eroarea pătratică. Dar aşa cum s-a arătat în paragraful 2.2, pentru scăderea erorii spectrale *SD*, pentru o anumită rată a cuantizării, se foloseşte eroarea pătratică ponderată, autorul propunând două variante de ponderi w_i. De aceea, structurarea trebuie să fie aplicabilă şi în cazul erorii pătratice ponderate. Acest lucru însă ridică anumite probleme suplimentare.

Cartea de cod pentru eroarea pătratică ponderată s-a obţinut, aşa cum s-a arătat, plecând de la o carte de cod obţinută prin algoritmul PNN, cu dimensiunea N=256, căreia i s-a aplicat algoritmul Lloyd, în care secvenţa de antrenament a fost cuantizată pe baza erorii pătratice ponderată. De exemplu, pentru cartea de cod corespunzătoare componentelor f_1, f_2, f_3, folosind eroarea pătratică ponderată dată prin (3.12), au fost necesare şi componentele f_4 ale vectorilor de antrenament respectivi. Adică, structura

finală a acestei cărți de cod pentru f_1, f_2, f_3 depinde și de componentele f_4 ale vectorilor de antrenament.

Considerând că cei N vectori de reproducere sunt dați (pentru k=3), deci structura regiunilor de cuantizare este fixă, depinzând de vectorii de reproducere, se poate ajunge la un paradox: un vector oarecare, care are anumite valori pentru f_1, f_2 și f_3 poate să aparțină unei regiuni de cuantizare pentru o valoare a lui f_4, iar dacă f_4 are o altă valoare, poate aparține altei regiuni de cuantizare, deoarece f_4 intervine în ponderea w_3. Adică, dacă s-ar prezenta într-un desen vectorii de reproducere corespunzători pentru k=3 împreună cu regiunile lor de cuantizare, orice punct (vector) din spațiul k=3 poate aparține la două sau chiar mai multe regiuni de cuantizare. Explicația acestui paradox este că, vectorilor de reproducere cu k=3 trebuie să li se asocieze, pentru determinarea formei regiunilor de cuantizare, vectorii de antrenament cu k=4, pentru că și a 4-a componentă contribuie la găsirea vectorului de reproducere asociat unui anumit vector cu k=3. Indicele vectorului de reproducere găsit codează însă numai primele 3 componente ale vectorului respectiv, a 4-a componentă făcând parte din următorul subvector care conține componentele f_4, f_5, f_6.

Desigur că aspectele amintite nu au importanță la un cuantizor vectorial prin despicare în care codarea se face prin căutare totală în interiorul fiecăreia din cele 3 cărți de cod. În cazul în care codarea se face prin metoda combinată arbore-căutare totală, în procesul de structurare a cărții de cod se pune problema determinării regiunilor de cuantizare de care aparțin vectorii uniform distribuiți caracterizați prin f_i=constant. Pentru k=3, acest lucru înseamnă determinarea regiunilor de cuantizare care intersectează planul caracterizat prin f_i=constant (i poate lua una dintre valorile 1, 2 sau 3). În cazul folosirii erorii pătratice fără pondere, acest lucru se făcea simplu prin cuantizarea vectorilor uniform distribuiți respectivi. În cazul erorii pătratice ponderate, de exemplu cea în care ponderea este dată prin (3.12), la cuantizarea vectorilor uniform distribuiți intervine ponderea w_3 care depinde și de f_4. Aceasta înseamnă că acestor vectori trebuie să li se precizeze și componenta f_4. Teoretic, f_4 ar trebui să ia valori cât mai multe în intervalul în care această componentă poate lua valori conform secvenței de antrenament, tabelul 4.1. Aceasta ar conduce practic la generarea unor

vectori uniform distribuiți cu dimensiunea 4 ceea ce ar mări foarte mult durata de execuție a programului care implementează structurarea cărții de cod. În plus, pentru subvectorul cu componentele f_4, f_5 și f_6 trebuie generați vectori uniformi care conțin și componentele f_3 și f_7 care intervin în ponderile w_4 și w_6. În mod similar, pentru subvectorul cu componentele f_7, f_8, f_9 și f_{10} trebuie generată și componenta f_6.

Practic însă, s-a constatat că nu este nevoie să se genereze pentru f_4 valori în toată gama admisibilă. Pentru a justifica această afirmație, s-a făcut experimentul descris în continuare. S-a generat mai întâi vectorul având parametrii: f_1=141 Hz, f_2=223 Hz, f_3=792 Hz (primul vector din tabelul 4.5), iar pentru f_4 s-au considerat valorile: $f_4=f_3+100xi$[Hz], i=1, 2,..., 9, acoperindu-se toată gama în care poate lua valori f_4. Acest vector a fost cuantizat, pentru fiecare din cele 9 valori ale lui f_4. Procesul s-a repetat cu f_2 având succesiv valorile 243 Hz, 263 Hz, ..., 543 Hz (decalate cu câte 20 Hz), în scopul determinării regiunilor de cuantizare care intersectează planul f_3=792 Hz. În urma cuantizării s-au constatat următoarele:

- pentru diferitele valori ale lui f_4 (când f_2=constant) s-au obținut cel mult 3 coduri (în cele mai multe cazuri 1 cod sau 2) distincte.
- începând de la valoarea a 4-a a lui f_4, codul obținut nu s-a mai modificat, obținându-se același cod până la 9-a valoare.
- în cazul în care pentru vectorul cu componentele f_1, f_2, f_3 fixate s-au obținut mai multe coduri (maximum 3) pentru diferitele valori ale lui f_4, aceste coduri s-au regăsit printre codurile obținute când f_2 a avut valori cu 20 Hz mai mici sau mai mari.

Acest experiment a fost reluat și pentru alte valori ale lui f_1, f_2, f_3, obținându-se rezultate asemănătoare. Considerând distanța dintre un vector $\boldsymbol{f}=(f_1, f_2, f_3)$ și vectorul de reproducere $\boldsymbol{y}_i=(y_{i1}, y_{i2}, y_{i3})$,

$$d(\boldsymbol{f}, \boldsymbol{y}_i) = w_1(f_1 - y_{i1})^2 + w_2(f_2 - y_{i2})^2 + w_3(f_3 - y_{i3})^2, \tag{4.18}$$

unde

$$w_3 = \frac{1{,}32 \cdot 10^5}{(f_3 - f_2)(f_4 - f_3)},$$

rezultatele exprimentale obținute se explică prin faptul că odată cu creșterea lui f_4 ponderea w_3 are valori tot mai mici, nemaicontând în raport cu

ponderile w_1 şi w_2 care vor fi esenţiale în stabilirea vectorului y_i pentru care distanţa $d(f, y_i)$ este minimă, iar w_3 este singurul termen în care intervine f_4.

S-a demonstrat că aceste concluzii sunt valabile şi pentru subvectorii (f_4, f_5, f_6) respectiv (f_7, f_8, f_9, f_{10}).

Pe baza acestor rezultate, autorul a procedat în felul următor pentru realizarea structurării:

- pentru cartea de cod cu componentele f_1, f_2, f_3, pentru fiecare vector uniform s-au considerat 3 valori pentru f_4, adică $f_4=f_3+100xi$[Hz], i=1,2,3.

- pentru cartea de cod cu componentele f_4, f_5, f_6 pentru fiecare vector uniform s-au considerat 3 valori pentru f_3, $f_3=f_4-100xi$[Hz], i=1,2,3 şi 3 valori pentru $f_7, f_7=f_6+150xi$[Hz], i=1,2,3.

- pentru cartea de cod cu componentele f_7, f_8, f_9, f_{10} pentru fiecare vector uniform s-au considerat 3 valori pentru $f_6, f_6=f_7-150xi$ [Hz], i=1,2,3.

Pasul de 150 Hz pentru f_6 şi f_7 se justifică prin faptul că aceste componente au o gamă dinamică mai mare.

După efectuarea procesului de structurare, s-a efectuat cuantizarea folosind metoda combinată arbore căutare-totală, pentru aceeaşi secvenţă de 310 vectori de test. S-a obţinut pentru fiecare din cei 3 subvectori aceeaşi eroare medie (sub formă pătratică ponderată) ca şi în cazul utilizării cuantizării prin căutare totală.

Deosebirea care apare în urma structurării cărţilor de cod în această situaţie, când măsura erorii de cuantizare este eroarea pătratică ponderată este creşterea numărului de vectori de reproducere asociaţi nodurilor terminale. Astfel, aşa cum s-a arătat anterior, la cuantizarea unui vector (f_1, f_2, f_3), intervine şi componenta f_4 care ia 3 valori distincte, putându-se obţine 2 sau 3 coduri diferite. În acest fel creşte numărul regiunilor de cuantizare care intersectează planul f_3=constant, adică creşte numărul vectorilor de reproducere comuni celor două grupe care rezultă după efectuarea testului în oricare din nodurile arborelui. Afirmaţia anterioară este valabilă şi pentru ceilalţi doi subvectori.

Astfel, la structurarea cărţii de cod corespunzătoare componentelor (f_1, f_2, f_3) s-a obţinut un număr mediu de vectori pe nod terminal $n_{m1,3}$=17,43, pentru cea cu componentele (f_4, f_5, f_6), $n_{m4,6}$=19,5, iar pentru cea cu componentele (f_7, f_8, f_9, f_{10}), $n_{m7,10}$=28,5.

În această situație, ținând cont că datorită ponderii se mai face o înmulțire pe lângă cea necesară pentru ridicarea la pătrat a diferenței între componenta vectorului de cuantizat și cea a vectorului de reproducere, complexitatea este

$$C = 2(3 \cdot 17{,}43 + 3 \cdot 19{,}5 + 4 \cdot 28{,}5) = 450 \text{ înmulțiri}$$

Memoria folosită în acest caz este

$$Mem = 10 \cdot 256 \cdot 2 + 2^6(17{,}43 + 19{,}5 + 28{,}5) + 3 \cdot 2^6 + 3 \cdot (2^6 - 1) \cdot 3 = 10066 \text{ octeți}$$

În tabelul 4.8 se reprezintă complexitatea, memoria și eroarea spectrală pentru cazul unui cuantizor prin despicare în care codarea s-a făcut prin căutare totală, (VQ desp. c.t.) așa cum au fost prezentate în paragraful 2.2 și, respectiv, pentru un cuantizor prin despicare în care codarea s-a realizat prin metoda combinată arbore căutare-totală, (VQ desp. a.-c.t.). În ambele cazuri, despicarea a fost de tipul (3,3,4), cu o rată de 24 biți/vector, și se reprezintă rezultatele pentru distanța de tip eroare pătratică (e.p.), respectiv eroare pătratică ponderată (e.p.p.), în care ponderea este dată de relația (3.12), propusă de autor.

Tabelul 4.8: *Valori ale caracteristicilor unui cuantizor vectorial în funcție de metoda de codare și eroarea folosită.*

VQ desp. c.t	Complexitate	Memorie	Er. Spectrală
e.p.	2560	5120	0,933 dB
e.p.p.	5120	5120	0,888 dB
VQ desp. a.-c.t.	Complexitate	Memorie	Er. Spectrală
e.p.	200	9580	0,933 dB
e.p.p.	450	10066	0,888 dB

Din tabelul 4.8 se observă că, atât în cazul folosirii erorii pătratice, cât și pentru eroarea pătratică ponderată, realizarea codării prin metoda combinată arbore-căutare totală produce scăderea complexității de aproximativ 11-12 ori, în timp ce memoria necesară crește de doar aproape 2 ori, în raport cu căutarea totală. Și de asemenea, ceea ce este esențial, folosirea acestei metode de codare nu produce creșterea erorii spectrale, indiferent de tipul măsurii erorii folosite. **Rezultă că avantajul acestei metode propuse**

de autor este reducerea drastică a complexităţii de calcul, fără creşterea erorii spectrale, cu preţul unei uşoare creşteri a memoriei utilizate.

În capitolul 4, autorul a introdus şi implementat codarea prin metoda combinată arbore-căutare totală ca alternativă la codarea prin căutare totală, în cadrul cuantizării vectoriale prin despicare prezentate în capitolul 3. În urma aplicării acestei metode se obţine o complexitate deosebit de scăzută, fără afectarea erorii de cuantizare în comparaţie cu căutarea totală.

5. CONCLUZII ŞI CONTRIBUŢII ORIGINALE

Teza abordează domeniul cuantizării vectoriale şi cuprinde contribuţiile autorului referitoare la metode şi algoritmi, cu aplicaţii la cuantizarea vectorială a semnalului vocal reprezentat prin linii spectrale de frecvenţă (parametri LSF).

În capitolul 1 autorul prezintă sub o formă sistematizată principalele noţiuni şi principii utilizate la cuantizarea vectorială, precum şi unele contribuţii originale.

1.1 Autorul demonstrează că în cazul în care măsura erorii de cuantizare este eroarea pătratică ponderată, regiunile de cuantizare nu mai sunt mărginite de drepte sau plane, ci de curbe sau suprafeţe mai complicate, astfel încât cuantizorul vectorial nu mai este regular şi politop. Acest fapt nu afectează însă aplicabilitatea metodelor de cuantizare vectorială folosite de autor.

1.2 Proprietatea cunoscută de compresie a cuantizării vectoriale, este demonstrată de autor pe baza a două exemple simple, pentru vectori având dimensiunile 2 şi, respectiv, 3. Această compresie este rezultatul corelaţiei între componentele vectorilor. De asemenea, se demonstrează că se produce o scădere a erorii de cuantizare la aceeaşi rată a cuantizării, faţă de cuantizarea scalară, în cazul în care între componentele vectorilor există o corelaţie suplimentară în sensul că ele sunt în ordine crescătoare.

În capitolul 2 se prezintă reprezentarea unui semnal vocal prin parametri LSF, arătându-se unele contribuţii ale autorului în acest sens. De asemenea, sunt prezentate şi principalele metode de cuantizare vectorială ale acestor parametri.

2.1 Autorul deduce o relaţie care exprimă densitatea spectrală de putere în funcţie de parametrii LSF. Apoi, se demonstrează, teoretic şi experimental, că densitatea spectrală de putere pentru o frecvenţă corespunzătoare unei linii spectrale, depinde în special de parametrii LSF

învecinaţi. Pe această bază se deduc două relaţii de calcul aproximativ şi simplificat al densităţii spectrale de putere. Cele două relaţii, verificate experimental, vor fi folosite pentru calculul ponderilor în cadrul erorii pătratice ponderate.

2.2. Pentru o cuantizare "transparentă" a parametrilor LSF, în literatură se arată că este necesară o rată de circa 25 biţi/vector. Acest lucru implică un mare număr de vectori de reproducere, adică o zonă de memorie mare pentru memorarea lor, precum şi un mare număr de operaţii pentru cuantizarea unui vector. De aceea, în practică sunt utilizate metode suboptimale care permit o reducere semnificativă a memoriei şi a complexităţii, cu preţul unei uşoare creşteri a erorii de cuantizare. În urma cercetării bibliografiei s-a constatat că principalele metode folosite în acest sens sunt cuantizarea vectorială prin despicare şi cuantizarea vectorială în mai multe stadii. Aproape toate celelalte metode folosesc ca bază una din cele două metode, iar prin diverse soluţii se încearcă reducerea erorii de cuantizare (sub formă de eroare spectrală), a complexităţii sau a memoriei pentru o anumită rată a cuantizării.

Capitolul 3 este dedicat proiectării printr-o metodă nouă a cărţii de cod pentru realizarea unui cuantizor vectorial prin despicare şi testării cărţii de cod obţinute pentru diferite forme ale erorii pătratice ponderate.

3.1 Faţă de soluţia clasică a proiectării cărţii de cod folosind algoritmul generalizat Lloyd pentru cărţi de cod având succesiv mărimile 2, 4, 8, ..., N, autorul foloseşte pentru vectori cu parametri LSF algoritmul PNN (pairwise nearest neighbor - împerecherea inteligentă a celor mai apropiaţi vectori), utilizat în literatură numai la cuantizarea imaginilor.

3.2 În scopul reducerii numărului de operaţii necesare în algoritmul PNN, autorul utilizează o metodă originală de împărţire a secvenţei de antrenament în buchete, urmată de grupări între mănunchiuri, iar în final se face o alăturare a unora dintre buchete, astfel încât să se obţină numărul dorit, N, de mănunchiuri. În acest fel, autorul introduce o variantă modificată a algoritmului PNN şi o aplică pentru vectori cu parametri LSF.

Pentru verificarea experimentală a metodei propuse pentru proiectarea cărţii de cod s-a folosit o secvenţă de antrenament de 2790 vectori LSF.

S-au obţinut 3 cărţi de cod cu dimensiunile vectorilor 3, 3 şi, respectiv 4, conform despicării realizate. Fiecare carte de cod, are N=256 vectori, realizându-se o rată de 24 biţi/vector.

3.3 Comparativ cu cartea de cod proiectată folosind algoritmul Lloyd pentru cărţi de cod având succesiv mărimile 2, 4, 8, ..., 256, proiectarea unei cărţi prin algoritmul PNN modificat necesită mult mai puţine operaţii (aproximativ 25%). Cuantizând atât secvenţa de antrenament cât şi o secvenţă de test de 310 vectori, diferită de cea de antrenament şi folosind cartea de cod PNN s-au obţinut erori cu 5% până la 10% mai mici faţă de cele obţinute prin cartea de cod proiectată prin algoritmul Lloyd. Rezultă că prin aplicarea algoritmului PNN modificat se obţine atât reducerea volumului de calcul cât şi scăderea erorii de cuantizare.

3.4 Aplicarea algoritmului Lloyd celor 3 cărţi obţinute prin algoritmul PNN modificat conduce la o convergenţă mai rapidă (6 iteraţii Lloyd faţă de 12) decât în cazul aplicării pentru cărţile de cod cu mărimile 2, 4, 8, ..., 256. La cuantizarea secvenţei de antrenament eroarea a scăzut cu 5% până la 10%, în timp ce la cuantizarea secvenţei de test a rămas aproximativ în aceleaşi limite. Toate aceste rezultate experimentale dovedesc calitatea algoritmului PNN modificat.

3.5 Deoarece programele care implementează proiectarea celor 3 cărţi de cod PNN utilizează un volum mare de date, autorul a folosit o alocare dinamică a memoriei, în scopul reducerii capacităţii de memorie utilizate. Astfel, s-a folosit o memorie de circa 45 kocteţi faţă de 400 kocteţi cât s-ar fi folosit fără alocarea dinamică.

3.6 Autorul verifică experimental prin câteva exemple de vectori că folosirea erorii pătratice ponderate produce cuantizarea mai precisă, comparativ cu eroarea pătratică, a componentelor cu pondere mare ale vectorilor LSF. De asemenea, printr-un alt experiment, se arată că eroarea spectrală este cu atât mai mică cu cât componentele cu ponderi mari sunt cuantizate mai precis, chiar dacă cele cu pondere mică sunt cuantizate mai puţin precis.

3.7 Se cuantizează o secvenţă de test de 310 vectori, folosind mai întâi ca şi măsură a distanţei între doi vectori eroarea pătratică, iar apoi

eroarea pătratică ponderată, în care ponderea are o valoare calculată ca rădăcină de ordinul 6 din valoarea exactă a densității spectrale de putere. În cazul erorii pătratice ponderate, se obține, așa cum era de așteptat, o reducere a erorii spectrale medii (de la 0,93 dB la 0,90 dB).

3.8 Pe baza relațiilor simplificate și aproximative pentru densitatea spectrală de putere deduse în capitolul 2, autorul introduce 2 noi relații de calcul pentru pondere. Se obțin valori foarte apropiate pentru eroarea spectrală medie, în raport cu cea obținută cu relația exactă a ponderii, cu avantajul unui număr mult mai mic de operații.

În capitolul 4, autorul introduce și implementează codarea prin metoda combinată arbore-căutare totală ca alternativă la codarea prin căutare totală, în cadrul cuantizării vectoriale prin despicare prezentate în capitolul 3. În urma aplicării acestei metode se obține o complexitate deosebit de scăzută, fără afectarea erorii de cuantizare în comparație cu căutarea totală.

4.1 Este prezentată această metodă, aplicată conform bibliografiei numai pentru vectori constituiți din eșantioane de semnal vocal, pe care autorul o va aplica pentru vectori LSF. Metoda constă în realizarea unei succesiuni de teste binare într-o structură de tip arbore până se ajunge într-unul din nodurile terminale. În nodul terminal respectiv se face o căutare totală printre vectorii de reproducere asociați acelui nod. Vectorii de reproducere asociați nodurilor terminale fac parte din cartea de cod optimă pentru secvența de antrenament așa cum a fost proiectată în capitolul precedent.

4.2 Pentru a se aplica codarea prin metoda combinată arbore-căutare totală este necesar să se realizeze o structurare a cărții de cod. Aceasta înseamnă că pentru fiecare nod intern al arborelui trebuie determinată componenta j care se testează (j poate lua orice valoare între 1 și k, unde k este dimensiunea vectorilor) și valoarea cu care se face compararea. De asemenea, pentru nodurile terminale trebuie determinați vectorii de reproducere asociați. Pentru a se realiza structurarea în acest fel, este esențială determinarea regiunilor de cuantizare situate de o parte și de alta a unei drepte sau plan din spațiul k-dimensional. În acest scop, autorul propune o nouă metodă, numită a intersecțiilor, care necesită un volum de

calcul mult mai mic (de aproximativ 12 ori) decât metoda proiecţiilor, prezentă în literatură. Pe baza acestei metode se poate face o structurare pentru orice vectori, nu numai pentru vectori LSF. În principiu metoda intersecţiilor permite realizarea structurării pe baza informaţiei obţinute prin cuantizarea unor vectori uniform distribuiţi, obţinuţi prin modificarea pe rând a componentelor cu un anumit pas.

4.3 Autorul realizează structurarea pentru vectori LSF, pe baza caracteristicilor acestor vectori concepând o formă simplificată a metodei intersecţiilor, pentru micşorarea numărului de vectori generaţi, în scopul micşorării suplimentare a volumului de calcul. Pasul cu care se modifică componenta variabilă a vectorilor uniform distribuiţi s-a stabilit la valoarea de 0,3...0,5 din distanţa medie dintre vectorii de reproducere învecinaţi. Această valoare a fost confirmată pe baza experimentelor. Astfel, efectuând cuantizarea conform metodei descrise la începutul acestui capitol, s-a obţinut aceeaşi eroare medie ca şi în situaţia în care s-a făcut prin căutare totală, cu avantajul reducerii foarte mari a complexităţii. Un alt experiment care confirmă valoarea determinată de autor pentru pasul de modificare a componentei variabile a vectorilor, este realizat prin generarea şi cuantizarea unor vectori pentru care pasul are valori din ce în ce mai mici, până la 1 Hz.

4.5 Acelaşi algoritm de structurare a cărţii de cod se face apoi utilizând eroarea pătratică ponderată pentru calculul distanţei între 2 vectori, în care ponderea are una din cele două relaţii introduse de autor. Deoarece structurarea se face separat pentru cele 3 tipuri de subvectori, iar în calculul erorii ponderate intervin şi componente diferite de ale subvectorului respectiv (de exemplu pentru structurarea cărţii de cod pentru subvectorii f_4, f_5, f_6 intervin şi componentele f_3 şi f_7), autorul face o analiză privind structurarea regiunilor de cuantizare în acest caz. De aici rezultă valori corespunzătoare pentru componentele care nu aparţin subvectorului pentru care se face structurarea, dar intervin în calculul ponderii. Şi în acest caz eroarea medie de cuantizare se obţine egală cu cea obţinută în situaţia în care codarea s-a făcut prin căutare totală (folosind eroarea pătratică ponderată), din nou obţinându-se o reducere foarte mare a complexităţii.

4.6 În final, autorul face o comparație între cele două variante de codare, prin metoda combinată arbore-căutare totală, respectiv prin căutare totală, folosite în cadrul cuantizării vectoriale prin despicare. În cazul primei variante, introduse de autor, complexitatea se reduce de circa 11-12 ori, obținându-se aceeași eroare față de a doua variantă, cu prețul creșterii capacității de memorie utilizate de aproximativ doar 2 ori.

Teza are un caracter unitar conținând toate elementele necesare implementării unui cuantizor vectorial: definirea principalelor noțiuni legate de cuantizarea vectorială, prezentarea parametrilor ce urmează să fie cuantizați, proiectarea cărții de cod și realizarea procesului propriuzis de cuantizare.

ANEXA 1. Notaţii şi prescurtări folosite în cadrul tezei

VQ, cuantizarea vectorială, (vector quantization în limba engleză)
Q, funcţia de cuantizare
$x=(x_1,x_2,...,x_k)$, vector cu dimensiunea k, unde x_i, $i=1,...,k$ reprezintă componentele vectorului
α, funcţia de codare
β, funcţia de decodare
$\mathcal{C}$, cartea de cod
I, mulţimea codurilor
N, mărimea cărţii de cod
$\Re^k$ spaţiul cu dimensiunea k
R_i, regiune de cuantizare
F_i, frontiera unei regiuni de cuantizare
$d(\cdot,\cdot)$, distanţa între doi vectori
H_{ij}, mulţimea (semispaţiul) care conţine toţi vectorii mai apropiaţi de vectorul de reproducere i decât de vectorul de reproducere j
w_i, funcţia pondere
n, lungimea secvenţei de antrenament
Mem, memoria unui cuantizor vectorial
C, complexitatea unui cuantizor vectorial
LSF, linii spectrale de frecvenţă
f_i, frecvenţele liniilor spectrale
parametrii LPC, parametrii obţinuţi prin codare prin predicţie liniară (linear predictiv coding în limba engleză)
$S(f)$, densitatea spectrală de putere
p, ordinul modelului liniar predictiv
a_i, $i=1,...,p$, coeficienţii de predicţie
$A(z)$, polinomul coeficienţilor de predicţie
SD, eroarea spectrală, (spectral distorsion în limba engleză)
MSVQ, cuantizor vectorial în mai multe stadii (multistage vector quantizer, în limba engleză)
TSVQ, cuantizor vectorial de tip arbore (tree structured vector quantizer, în limba engleză)
K, numărul de stadii la algoritmul MSVQ
d, numărul de niveluri al unei structuri de tip arbore
M, parametru specific în cadrul algoritmului MSVQ-M
algoritmul PNN, împerecherea inteligentă a celor mai apropiate vecinătăţi (pairwise nearest neighbor, în limba engleză)

s_1, s_2, subsecvenţele rezultate în urma împărţirii efectuate într-un nod al arborelui pentru obţinerea buchetelor

n_{s1}, n_{s2}, numărul vectorilor de antrenament din subsecvenţele s_1 şi s_2

C_i, mănunchi

C_{ij}, mănunchiul obţinut prin unirea mănunchirilor C_i şi C_j

$\bar{x}_i$, centroidul mănunchiului C_i

E_i, eroarea de cuantizare a vectorilor din mănunchiul C_i prin centroidul mănunchiului

$\bar{x}_i$, centroidul mănunchiului C_i

N_B, numărul de buchete

N_{TM}, numărul total de mănunchiuri

m, numărul de noduri terminale în cadrul arborelui de la cuantizarea vectorială de tip arbore-căutare totală

(j,v_j), componenta şi valoarea cu care se compară componenta respectivă

n_s, n_d, numărul regiunilor de cuantizare situate la stânga, respectiv la dreapta faţă de un plan

N_i, numărul vectorilor de reproducere asociaţi cu nodul i de pe un nivel al arborelui utilizat pentru structurarea cărţii de cod

N_{ai}, numărul vectorilor de antrenament asociaţi cu nodul i de pe un nivel al arborelui utilizat pentru structurarea cărţii de cod

N_{mediu}, numărul mediu de vectori de reproducere dintr-o regiune de cuantizare

n_{mediu}, numărul mediu de valori ale unei componente a unui vector pentru care se determină parametrii n_s şi n_d

Δ_i, valoarea pasului cu care se modifică o componentă LSF în cadrul metodei intersecţiilor

ANEXA 2. Programe realizate de autor şi folosite în cadrul tezei

Din multitudinea de programe realizate de autor în cadrul elaborării acestei teze, în această anexă se prezintă două dintre ele, considerate mai reprezentative

1.Programul care implementează algoritmul de structurare a cărţii de cod

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<math.h>
#include<process.h>
FILE *fp1,*fp2,*fp3,*fp4,*fp5,*fp6;
int niv=6;/*nr de nivele*/
double ii,jj; int j7,j8,de7,de8;
int bu,nll[128],nrr[128],de1,de2;float pl[3],pr[3],plf[3],prf[3];int nl1[128],nr1[128];
int ll[128],rr[128],nv;
int N1=1024,n=2790,n1=256,ind,j1,y,p,s,ip,s1,l,id,in,t,k,S1,r,r1,c,c1,coord;
unsigned int cod[350],q[256],jmax,a2min,nr,sum=0,st[3][220],dr[3][220],com[3][150],comun1[128];
unsigned int stf[3][220],drf[3][220],commf[3][200];
unsigned int comf[128][150];
float rap,aa[3],x,x1,y1,min[128],minim[250],max,prag[7][128],a[3],b[3],d[256];long int i,j,i1;int
g1=0,g2=0,g3=0;
long int dim=3;/*dimens.vectorilor*/
int coordon[7][128],lim,depl,depla,out,val;float pas;
float df0,df1,df2,p1=0.51,p2=0.49,min0,min1,min2,max0,max1,max2,E[4],Emin;
int sfa[128]; int sfv[128];int comun[128],omun[128];
float huge *ptr1;float huge *ptr2;int huge *ptr3;int huge *ptr4;int huge *ptr5;int huge *ptr6; int huge
*ptr7;int huge *ptr8;
void main (void) {
for(i=0;i<3;i++)     /*initializ.cu 0 a probabil.*/
   pl[i]=pr[i]=0; clrscr();
ptr1=(float huge*)farmalloc(3700);
      if(ptr1==0)
      {printf("nu se poate aloca memorie 1");
      exit(1);
      }
if(!(fp1=fopen("c:\\home\\mischie\\car3.dat","r+")))
   {puts("nu se poate deschide fis1");
   exit(1);}
for(i=0;i<dim*n1;i++)
   {fscanf(fp1,"%f",&x);
    *(ptr1+i)=x;}            /*vectori de reproducere*/
fclose(fp1);
for(i=212*dim;i<233*dim;i++)
```

```
   printf(" %5.1f ",*(ptr1+i));
printf("\n");
ptr2=(float huge*)farmalloc(37050);
     if(ptr2==0)
     {printf("nu se poate aloca memorie 2");
     exit(1);
     }
if(!(fp2=fopen("c:\\home\\mischie\\antr3.dat","r+")))
   {puts("nu se poate deschide fis1");
   exit(1);}
for(i=0;i<dim*n;i++)
   {fscanf(fp2,"%f",&x);
    *(ptr2+i)=x;}            /*vectori de antrenam.*/
fclose(fp2);
printf("\n");
for(i=2780;i<2790;i++)
   printf(" %5.1f ",*(ptr2+i));
printf("\n");
ptr3=(int huge*)farmalloc(10000); /*memorie pt.indicii vectorilor de antren.*/
     if(ptr3==0)
     {printf("nu se poate aloca memorie 3");
     exit(1);
     }
ptr4=(int huge*)farmalloc(5500); /*memorie pt.indicii vector.de repr.din celule*/
     if(ptr4==0)
     {printf("nu se poate aloca memorie 4");
     exit(1);
     }
ptr5=(int huge*)farmalloc(5000); /*memorie pt.indicii vectorilor de antren.*/
     if(ptr5==0)              /*din grupele -st.- rezultate dupa impartire*/
     {printf("nu se poate aloca memorie 5");
     exit(1);
     }
ptr6=(int huge*)farmalloc(5000); /*memorie pt.indicii vector.de antr.*/
     if(ptr6==0)              /*din grupele -dr.-rezultate dupa impartire*/
     {printf("nu se poate aloca memorie 6");
     exit(1);
     }
ptr7=(int huge*)farmalloc(5000); /*memorie pt.indicii vectorilor de repr.*/
     if(ptr7==0)      /*din grupa-st-,de eroare min.rezultate dupa impartire*/
     {printf("nu se poate aloca memorie 7");
     exit(1);
     }
ptr8=(int huge*)farmalloc(5000); /*memorie pt.indicii vector.de antr.*/
     if(ptr8==0)      /*din grupa-dr-,de eroare min.rezultate dupa impartire*/
     {printf("nu se poate aloca memorie 8");
     exit(1);
     }
```

```
    /*initializari*/
    sfa[0]=2790;
    sfv[0]=256;
    for(i=0;i<n;i++)
        *(ptr3+i)=i;
    for(i=0;i<n1;i++)
        *(ptr4+i)=i;
for(ii=0;ii<niv;ii++){    /*nr de nivele de despicare */
j7=0;j8=0;
for(jj=0;jj<pow(2,ii);jj++)    /*nr de celule care se despica pe niv.ii*/
{ depl=0;
 if(jj==0) depl=0;
        else{for(i=0;i<jj;i++)
            depl=depl+sfa[i];}
 b[0]=190;b[1]=270;b[2]=520; /*valorile de la care se pleaca*/
 max0=0;min0=4000;
 for(ip=0;;ip++)
 {  pl[0]=0;pr[0]=0;
   for(i=0;i<sfa[jj];i++)            /*determinare probabilitati*/
  {in=*(ptr3+depl+i);
   if(*(ptr2+dim*in+0)<b[0]+1*ip) pl[0]=pl[0]+1;
        else pr[0]=pr[0]+1;
    if(ip==0){ if(*(ptr2+dim*in+0)<min0) min0=*(ptr2+dim*in+0);
                if(*(ptr2+dim*in+0)>max0) max0=*(ptr2+dim*in+0);
              }
   }
  pl[0]=(float)pl[0]/sfa[jj];pr[0]=(float)pr[0]/sfa[jj];
  if((pl[0])>0.5) break;
 }
 b[0]=b[0]+ip*1;
 printf("\n%d %f %f %5.1f",ip,pl[0],pr[0],b[0]);
 max1=0;min1=4000;
  for(ip=0;;ip++)
 {  pl[1]=0;pr[1]=0;
     for(i=0;i<sfa[jj];i++)
        { in=*(ptr3+depl+i);
          if(*(ptr2+dim*in+1)<b[1]+2*ip) pl[1]=pl[1]+1;
            else pr[1]=pr[1]+1;
            if(ip==0){ if(*(ptr2+dim*in+1)<min1) min1=*(ptr2+dim*in+1);
                       if(*(ptr2+dim*in+1)>max1) max1=*(ptr2+dim*in+1);
                     }

        }
   pl[1]=(float)pl[1]/sfa[jj];pr[1]=(float)pr[1]/sfa[jj];
   if((pl[1])>0.5)  break;
 }
    b[1]=b[1]+ip*2;
    printf("\n%d %f %f %5.1f",ip,pl[1],pr[1],b[1]);
```

```
min2=4000;max2=0;
   for(ip=0;;ip++)
 {  pl[2]=0;pr[2]=0;
     for(i=0;i<sfa[jj];i++)
         { in=*(ptr3+depl+i);
           if(*(ptr2+dim*in+2)<b[2]+1*ip) pl[2]=pl[2]+1;
             else pr[2]=pr[2]+1;
             if(ip==0){ if(*(ptr2+dim*in+2)<min2) min2=*(ptr2+dim*in+2);
                        if(*(ptr2+dim*in+2)>max2) max2=*(ptr2+dim*in+2);
                       }
         }
    pl[2]=(float)pl[2]/sfa[jj];pr[2]=(float)pr[2]/sfa[jj];
   if((pl[2])>0.5) break;
  }
  b[2]=b[2]+ip*1;
  printf("\n%d %f %f %5.1f",ip,pl[2],pr[2],b[2]);
for(j=0;j<3;j++)   /*coordonata */
   printf(" %f %f;",pl[j],pr[j]);
/*   getch();              */
printf("\n");
printf(" %5.1f %5.1f %5.1f %5.1f %5.1f %5.1f ",min0,max0,min1,max1,min2,max2);
/*getch();  */
for(i=0;i<256;i++)
    q[i]=0;
out=0;pas=10;
df2=20;df1=20;df0=20;
for(bu=0;;)
{nv=0;
 a[0]=min0;a[1]=min1;a[2]=b[2]+10*bu+out*pas;      /*coordonata 2*/
 for(i=0;i<n1;i++)
  q[i]=0;
  depl=0;
    if(jj==0) depl=0;
            else{for(id=0;id<jj;id++)
                depl=depl+sfv[id];   }
 for(j1=0;;j1++)
 {printf("\n%5.1f %5.1f %5.1f",a[0],a[1],a[2]);
  for(j=0;;j++)
  {nv=nv+1;
   for(i=0;i<sfv[jj];i++)
     {d[i]=0;
     in=*(ptr4+depl+i);
     for(k=0;k<3;k++)
     {x=*(ptr1+in*3+k);              /*calcul distanta*/
      d[i]=d[i]+(x-a[k])*(x-a[k]);} }
  minim[j]=d[0];coord=0;                    /*determin.minim*/
  for(i=1;i<sfv[jj];i++)
     {if(d[i]<minim[j]) {minim[j]=d[i];coord=i;}
```

```
    printf("");}
   cod[j]=coord;
   if(coord==202)                                          .
   { printf("\n acum   %5.1f %5.1f %5.1f",a[0],a[1],a[2]);  getch();}
   if(minim[j]<10000) q[coord]=q[coord]+1;
   a[1]=a[1]+df1; if(a[1]>max1+df1) {jmax=j;break;}
                  if(a[1]>a[2]) {jmax=j;break;}
}
printf("\n  %d",j1);
printf("\n  %2.1f",jj);
/*for(i=0;i<256;i++)
  printf("%d; ",q[i]);          */
nr=0;
for(i=0;i<sfv[jj];i++)
  if(q[i]==0) nr=nr+1;
  printf("\nnr=%d",nr);
  printf("\n%5.1f %5.1f %5.1f",a[0],a[1],a[2]);
/*  getch(); */
a[0]=a[0]+df0;a[1]=min1;if(a[0]>min1){a[1]=a[0]+10;}  if(a[0]>max0+df0) break; }
printf("\n");
for(j=0;j<jmax;j++)
  {printf("%6.1f %d; ",minim[j],cod[j]);
    }
   printf("\n");
for(i=0;i<sfv[jj];i++)
  printf("%d; ",q[i]);
nr=0;
s=0;r=0;c=0;
for(i=0;i<sfv[jj];i++)
 { in=*(ptr4+depl+i);
  if(q[i]==0){ nr=nr+1;
        if(*(ptr1+dim*in+2)<a[2]) {st[out][s]=in;
                                   s=s+1;}
        else {dr[out][r]=in;
              r=r+1;}   }
    else {com[out][c]=in;c=c+1;}
    sum=sum+q[i];}
    printf("\n nr=%d",nr);
    printf("\n sum=%d\n",sum);
for(i=0;i<s;i++)
 printf("%d ",st[out][i]);
printf("\n");
for(i=0;i<r;i++)
 printf("%d ",dr[out][i]);
printf("\n");
for(i=0;i<c;i++)
 printf("%d ",com[out][i]);
ll[out]=s+c;rr[out]=r+c;omun[out]=c;
```

```
rap=(float)ll[out]/rr[out];
if(rap>0.9) {  pas=5;
                depla=0;        /*recalculare probabilitati 2 */
          if(jj==0) depla=0;
                 else{for(i=0;i<jj;i++)
                      depla=depla+sfa[i];}
              pl[out]=0;pr[out]=0;
              for(i=0;i<sfa[jj];i++)
                 {in=*(ptr3+depla+i);
              if(*(ptr2+dim*in+2)<a[2]) pl[out]=pl[out]+1;
                 else pr[out]=pr[out]+1;
                    }
  pl[out]=(float)pl[out]/sfa[jj];pr[out]=(float)pr[out]/sfa[jj];

             E[out]=ll[out]*pl[out]+rr[out]*pr[out];
             out=out+1;}
     else    {bu=bu+1;}
             if(out==3) break;
  }/* sfirsit coord 2*/
Emin=E[0]; val=0;
for(i=1;i<out;i++)
 if(E[i]<Emin) {Emin=E[i];val=i; }
  b[2]=a[2]-(out-1-val)*pas;
  nll[2]=ll[val];nrr[2]=rr[val];comun[2]=omun[val];
for(i=0;i<nll[2]-comun[2];i++)
  stf[2][i]=st[val][i];
for(i=0;i<nrr[2]-comun[2];i++)
  drf[2][i]=dr[val][i];
for(i=0;i<comun[2];i++)
  commf[2][i]=com[val][i];
plf[2]=pl[val];prf[2]=pr[val];
out=0;pas=10;
for(bu=0;;)
{a[0]=min0;a[1]=b[1]+bu*10+out*pas;a[2]=a[1]+10;if(a[2]<min2) a[2]=min2; /*coordonata 1*/
for(i=0;i<256;i++)  /*!!!*/
q[i]=0;
for(j1=0;;j1++)
{printf("\n%5.1f %5.1f %5.1f",a[0],a[1],a[2]);
 for(j=0;;j++)
 {for(i=0;i<sfv[jj];i++)
   {d[i]=0;
   in=*(ptr4+depl+i);
   for(k=0;k<3;k++)
   {x=*(ptr1+in*dim+k);            /*calcul distanta*/
    d[i]=d[i]+(x-a[k])*(x-a[k]);} }
 minim[j]=d[0];coord=0;                  /*determin.minim*/
 for(i=1;i<sfv[jj];i++)
    {if(d[i]<minim[j]) {minim[j]=d[i];coord=i;}
```

```
    printf("");}
   cod[j]=coord;if(minim[j]<10000) q[coord]=q[coord]+1;
   a[2]=a[2]+df2; if(a[2]>max2+df2) {jmax=j;break;}
 }
 printf("\n    %d\n",j1);
 printf("\n    %2.1f\n",jj);
 /*for(i=0;i<256;i++)
   printf("%d; ",q[i]);           */
 nr=0;
 for(i=0;i<sfv[jj];i++)
   {if(q[i]==0) nr=nr+1;
   }
   printf("\nnr=%d",nr);
   printf("\n%5.1f %5.1f %5.1f",a[0],a[1],a[2]);
/*   getch(); */           /*570!!!*/
 a[0]=a[0]+df0;a[2]=a[1]+10;if(a[2]<min2) a[2]=min2;if(a[0]>a[1]) break;
                              if(a[0]>max0+df0) break; }
 printf("\n");
 nr=0;
 s=0;r=0;c=0;
 for(i=0;i<sfv[jj];i++)
  {in=*(ptr4+depl+i);
   if(q[i]==0){ nr=nr+1;
          if(*(ptr1+dim*in+1)<a[1]) {st[out][s]=in;
                                        s=s+1;}
          else {dr[out][r]=in;
              r=r+1;}    }
     else {com[out][c]=in;
           c=c+1;}
     sum=sum+q[i];}
     printf("\n nr=%d",nr);
     ll[out]=c+s;rr[out]=c+r;omun[out]=c;
     printf("\n");
     rap=(float)ll[out]/rr[out];
 if(rap>0.9) { pas=5;
                depla=0;        /*recalculare probabilitati 1 */
             if(jj==0) depla=0;
                    else{for(i=0;i<jj;i++)
                         depla=depla+sfa[i];}
               pl[out]=0;pr[out]=0;
               for(i=0;i<sfa[jj];i++)
                  {in=*(ptr3+depla+i);
               if(*(ptr2+dim*in+1)<a[1]) pl[out]=pl[out]+1;
                  else pr[out]=pr[out]+1;
                      }
    pl[out]=(float)pl[out]/sfa[jj];pr[out]=(float)pr[out]/sfa[jj];

              E[out]=ll[out]*pl[out]+rr[out]*pr[out];
```

```
            out=out+1;}
          else{bu=bu+1;}
            if(out==3) break;
  }/* sfirsit coord 0*/
Emin=E[0]; val=0;
for(i=1;i<out;i++)
 if(E[i]<Emin) {Emin=E[i];val=i;  }
  b[1]=a[1]-(out-1-val)*pas;
  nll[1]=ll[val];nrr[1]=rr[val];comun[1]=omun[val];
for(i=0;i<nll[1]-comun[1];i++)
  stf[1][i]=st[val][i];
for(i=0;i<nrr[1]-comun[1];i++)
  drf[1][i]=dr[val][i];
for(i=0;i<comun[1];i++)
  commf[1][i]=com[val][i];
plf[1]=pl[val];prf[1]=pr[val];

out=0;pas=4;
for(bu=0;;)
{a[0]=b[0]+bu*4+out*pas;a[1]=a[0]+10;a[2]=min2;   /*coordonata 0*/
for(i=0;i<n1;i++)
q[i]=0;
for(j1=0;;j1++)
{printf("\n%5.1f %5.1f %5.1f",a[0],a[1],a[2]);
 for(j=0;;j++)
 {for(i=0;i<sfv[jj];i++)
   {d[i]=0;
   in=*(ptr4+depl+i);
   for(k=0;k<3;k++)
   {x=*(ptr1+in*dim+k);            /*calcul distanta*/
    d[i]=d[i]+(x-a[k])*(x-a[k]);} }
 minim[j]=d[0];coord=0;                    /*determin.minim*/
 for(i=1;i<sfv[jj];i++)
    {if(d[i]<minim[j]) {minim[j]=d[i];coord=i;}
     printf("");}
   cod[j]=coord;if(minim[j]<10000) q[coord]=q[coord]+1;
   a[2]=a[2]+df2; if(a[2]>max2+df2) {jmax=j;break;}
 }
 printf("\n   %d\n",j1);
 printf("\n   %2.1f\n",jj);
 /*for(i=0;i<256;i++)
   printf("%d; ",q[i]);          */
 nr=0;
 for(i=0;i<sfv[jj];i++)
   {if(q[i]==0) nr=nr+1;
   }
   printf("\nnr=%d",nr);
   printf("\n%5.1f %5.1f %5.1f",a[0],a[1],a[2]);
```

```
 getch();
a[1]=a[1]+df1;a[2]=min2;  if(a[1]>min2) a[2]=a[1]+20;if(a[1]>max1+df1) break; }
printf("\n");
 nr=0;
s=0;r=0;c=0;
for(i=0;i<sfv[jj];i++)
 {in=*(ptr4+depl+i);
  if(q[i]==0){ nr=nr+1;
        if(*(ptr1+3*in+0)<a[0]) {st[out][s]=in;
                                 s=s+1;}
        else {dr[out][r]=in;
              r=r+1;}    }
    else {com[out][c]=in;
          c=c+1;}
    sum=sum+q[i];}
    printf("\n nr=%d",nr);
    ll[out]=c+s;rr[out]=c+r;omun[out]=c;
    printf("\n");
    rap=(float)ll[out]/rr[out];
  if(rap>0.9) {pas=2;
               depla=0;
           if(jj==0) depla=0;
                 else{for(i=0;i<jj;i++)
                      depla=depla+sfa[i];}
              pl[out]=0;pr[out]=0;   /*recalculare probabilitati 0 */
              for(i=0;i<sfa[jj];i++)
                 {in=*(ptr3+depla+i);
              if(*(ptr2+dim*in+0)<a[0]) pl[out]=pl[out]+1;
                 else pr[out]=pr[out]+1;
                     }
   pl[out]=(float)pl[out]/sfa[jj];pr[out]=(float)pr[out]/sfa[jj];

             E[out]=ll[out]*pl[out]+rr[out]*pr[out];
             out=out+1;}
          else{bu=bu+1;}
             if(out==3) break;
   }
 Emin=E[0]; val=0;
 for(i=1;i<out;i++)
  if(E[i]<Emin) {Emin=E[i];val=i;}
    b[0]=a[0]-(out-1-val)*pas;
    nll[0]=ll[val];nrr[0]=rr[val];comun[0]=omun[val];
 for(i=0;i<nll[0]-comun[0];i++)
    stf[0][i]=st[val][i];
 for(i=0;i<nrr[0]-comun[0];i++)
    drf[0][i]=dr[val][i];
 for(i=0;i<comun[0];i++)
    commf[0][i]=com[val][i];
```

```
plf[0]=pl[val];prf[0]=pr[val];


for(j=0;j<3;j++)
  printf(" %d %d;",nll[j],nrr[j]);
 printf("\n");
/* getch(); */
 for(i=0;i<3;i++) /* coordonata */
    {aa[i]=(float)plf[i]*nll[i]+prf[i]*nrr[i];
     printf("%5.1f ;",aa[i]);}
 min[jj]=aa[0];
 coordon[ii][jj]=0;
 for(i=1;i<3;i++)
  if(aa[i]<min[jj]) {coordon[ii][jj]=i;min[jj]=aa[i];}
  printf("\n%d %5.1f",coordon[ii][jj],min[jj]);
  printf("");
  lim=coordon[ii][jj];   /*valoarea coordonatei care da minimul*/
  prag[ii][jj]=b[lim];
  nl1[jj]=nll[lim];nr1[jj]=nrr[lim];comun1[jj]=comun[lim];
  de7=j7;
  for(i=0;i<nl1[jj]-comun1[jj];i++)   /*elem.grupei care dau minimul*/
    {*(ptr7+de7+i)=stf[lim][i];
         j7=j7+1;printf("");}
  de8=j8;
  for(i=0;i<nr1[jj]-comun1[jj];i++)
    {*(ptr8+de8+i)=drf[lim][i];
         j8=j8+1;printf("");}
  for(i=0;i<comun1[jj];i++)
    {comf[jj][i]=commf[lim][i];
     printf("");}
  } /*sfirsit jj*/
for(jj=0;jj<pow(2,ii);jj++)
 {sfv[2*jj]=nl1[jj];  /*nr.de elem.din cele doua celule obtin.din */
  sfv[2*jj+1]=nr1[jj];}        /*impart.grupei jj*/
printf("\n");

for(jj=0;jj<pow(2,ii);jj++)
 {printf("%d %5.1f  ",coordon[ii][jj],prag[ii][jj]);
  printf("%d %d ",sfv[2*jj],sfv[2*jj+1]);}
printf("\n");
/*getch(); */
de7=0;de8=0;
for(jj=0;jj<pow(2,ii);jj++)
 {
 if(jj==0)  depl=0;

 for(i=0;i<nl1[jj]-comun1[jj];i++)   /*indicii vector.din prima celula*/
   *(ptr4+depl+i)=*(ptr7+de7+i);
```

```
    de7=de7+nl1[jj]-comun1[jj];
for(i=0;i<comun1[jj];i++)
  *(ptr4+depl+nl1[jj]-comun1[jj]+i)=comf[jj][i];

for(i=0;i<nr1[jj]-comun1[jj];i++)   /*indicii vector.din celula doi*/
  *(ptr4+depl+sfv[2*jj]+i)=*(ptr8+de8+i);
   de8=de8+nr1[jj]-comun1[jj];
for(i=0;i<comun1[jj];i++)
  *(ptr4+depl+sfv[2*jj]+nr1[jj]-comun1[jj]+i)=comf[jj][i];

for(i=0;i<sfv[2*jj];i++)
  printf("%d ",*(ptr4+depl+i));
  printf("\n   ");
for(i=0;i<sfv[2*jj+1];i++)
  printf("%d ",*(ptr4+depl+sfv[2*jj]+i));

  depl=depl+sfv[2*jj]+sfv[2*jj+1]; /*actualiz.deplasament*/
  printf("\n");
    }    /*sfirsit jj*/
printf("\n");
de1=0;de2=0;     /*initial.cu 0 a deplas.in ptr5 si ptr6*/
for(jj=0;jj<pow(2,ii);jj++)     /*impartire vect.de antren in celule*/
{lim=coordon[ii][jj];
 depl=0;s=0;r=0;
if(jj==0) depl=0;
        else{for(i=0;i<jj;i++)
            depl=depl+sfa[i];}
for(i=0;i<sfa[jj];i++)
{in=*(ptr3+depl+i);
 if(*(ptr2+dim*in+lim)<prag[ii][jj]) {*(ptr5+de1+s)=in;
                                     s=s+1;}
        else {*(ptr6+de2+r)=in;
            r=r+1; }
 }
 nll[jj]=s;nrr[jj]=r;
 de1=de1+s;de2=de2+r;
 } /* sfirsit jj*/
for(jj=0;jj<pow(2,ii);jj++)    /*stabil.num.de vect.de antren.din cele 2 celule*/
  {sfa[2*jj]=nll[jj];sfa[2*jj+1]=nrr[jj];
   printf("\n%d %d ",sfa[2*jj],sfa[2*jj+1]);   }
/*    getch();  */
printf("\n");
for(jj=0;jj<pow(2,ii);jj++)
 {if(jj==0)  {depl=0;de1=0;de2=0;}

   for(i=0;i<nll[jj];i++)   /*indicii vector.din prima celula*/
    {*(ptr3+depl+i)=*(ptr5+de1+i);
     printf("");}
```

```
    for(i=0;i<nrr[jj];i++)   /*indicii vector.din celula doi*/
      {*(ptr3+depl+nll[jj]+i)=*(ptr6+de2+i);
      printf("");}
    for(i=0;i<30;i++)
      printf("%d ",*(ptr3+depl+i));
    printf("\n");
    for(i=0;i<30;i++)
      printf("%d ",*(ptr3+depl+sfa[2*jj]+i));
    printf("\n");
   depl=depl+sfa[2*jj]+sfa[2*jj+1];
   de1=de1+nll[jj];de2=de2+nrr[jj];
  }/*sfirsit jj */
} /*sfirsit ii*/
if(!(fp3=fopen("c:\\home\\mischie\\coor36.dat","w+")))
    {puts("nu se poate deschide fis3");
    exit(1);}
for(ii=0;ii<niv;ii++)
  for(jj=0;jj<pow(2,ii);jj++)
  fprintf(fp3,"%d ",coordon[ii][jj]);
fclose(fp3);
if(!(fp4=fopen("c:\\home\\mischie\\pra36.dat","w+")))
    {puts("nu se poate deschide fis4");
    exit(1);}
for(ii=0;ii<niv;ii++)
  for(jj=0;jj<pow(2,ii);jj++)
  fprintf(fp4,"%5.1f ",prag[ii][jj]);
fclose(fp4);
if(!(fp5=fopen("c:\\home\\mischie\\sf36.dat","w+")))
    {puts("nu se poate deschide fis5");
    exit(1);}
  for(jj=0;jj<pow(2,niv-1);jj++)
  {fprintf(fp5,"%d ",sfv[2*jj]);
   fprintf(fp5,"%d ",sfv[2*jj+1]);}
fclose(fp5);
if(!(fp6=fopen("c:\\home\\mischie\\indic36.dat","w+")))
    {puts("nu se poate deschide fis6");
    exit(1);}
  s=0;
  for(jj=0;jj<pow(2,niv-1);jj++)
  s=s+sfv[2*jj]+sfv[2*jj+1];
  for(i=0;i<s;i++)
  fprintf(fp6,"%d ",*(ptr4+i));
fclose(fp6);
for(jj=0;jj<pow(2,niv);jj++)
  printf("%4.1f ",min[jj]);
printf("");

}
```

2. Program care implementează algoritmul PNN rapid

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<stdlib.h>
FILE *fp1,*fp2,*fp3;
int NB=410,NB1,NC[420],NC1[420];float x;/*nr.de bucheti,nr.de cluster/buch,centroizi ai clusterului
*/
int NV=2790; /*nr de vectori(clusteri)*/
float CG[420][10];  /*centroid general*/
int dim=4,sum=0; /* dimens. vectorilor */
float min,x1,y,S[420][10],S1[11][12],S2[420],S3[420],dis[420];
int u,u1,umin,i,ii,s,j,i1,j1,k,l,l[420],J[420],n[420][10],loc[420],prag,suma=0,Nuniri;
float huge *ptr;
float tabl(float huge*ptr,int i,int j,int k)
        {int sss=0,ss;
        for(ss=0;ss<i;ss++)
          sss=sss+NC1[ss]+2;
          return(*(ptr+sss*dim+j*dim+k));
          }
float dist(int i,int j,int k)
        {float d=0;int t;
        for(t=0;t<dim;t++)
          d=d+(tabl(ptr,i,j,t)-tabl(ptr,i,k,t))*(tabl(ptr,i,j,t)-tabl(ptr,i,k,t));
          return d;
        }
int adresa(int i,int j,int k)
         {int sss=0,ss,adr;
         for(ss=0;ss<i;ss++)
           sss=sss+NC1[ss]+2;
           adr=sss*dim+j*dim+k;
           return adr;
           }
void main(void) {
clrscr();
if(!(fp1=fopen("c:\\home\\mischie\\nvect710.dat","r+")))
{puts("\n  nu se poate deschide fis1");
exit(1);}
for(i=0;i<NB;i++)
  fscanf(fp1,"%d",&NC[i]);
for(i=0;i<NB;i++)
  NC1[i]=NC[i];
printf("%d %d %d %d %d %d %d",NC[0],NC[1],NC[2],NC[3],NC[4],NC[5],NC[6]);
fclose(fp1);
for(i=0;i<NB;i++)
```

```
suma=suma+NC[i];
printf("\n%d \n",suma);
ptr=(float huge*)farmalloc(150000);  /*pt.citirea buchetilor*/
  if(ptr==0)
    {printf("\nnu se poate aloca memorie");
     exit(1);
     }

if(!(fp2=fopen("c:\\home\\mischie\\buche710.dat","r+")))
{puts("nu se poate deschide fis2");
exit(1);}

for(i=0;i<NB;i++)
  {sum=sum+NC[i-1];
   for(j=0;j<NC[i];j++)
     for(k=0;k<dim;k++)
          {fscanf(fp2,"%f",&x);
           *(ptr+(sum+j)*dim+k)=x;}
   sum=sum+2;
  }
fclose(fp2);
printf("\n sum=%d\n",sum);
for(i=50;i<150;i++)
 printf("%5.1f ",*(ptr+i));
getch();
NB1=NB;/*nr de buch.folosit pt a determ. 1/2 din nr.real de bucheti*/
printf("\n%5.1f %5.1f %5.1f %5.1f",tabl(ptr,0,0,0),tabl(ptr,0,1,1),tabl(ptr,1,0,3),tabl(ptr,1,0,4));
getch();
for(i=0;i<NB;i++)    /*centroizii generali se initializeaza cu 0 */
  for(k=0;k<dim;k++)
    CG[i][k]=0;
for(i=0;i<NB;i++)    /*se calculeaza centroizii generali*/
  for(k=0;k<dim;k++)
    {for(j=0;j<NC[i];j++)
     CG[i][k]=CG[i][k]+tabl(ptr,i,j,k);
     CG[i][k]=CG[i][k]/NC[i];
     printf("");
    }
for(i=0;i<NB;i++)          /*initializare cu 0 a erorilor pe cluster*/
  for(j=0;j<NC[i];j++)     /*si cu 1 a numerelor de vectori pe cluster*/
          {S[i][j]=0;
          n[i][j]=1;}
for(ii=0;;ii++)
{for(i=0;i<NB;i++)    /*i=nr.curent al buchetului in care se face cercetarea*/
  {printf("\n i=%d",i);
  if(NC[i]==0)  S2[i]=S3[i];
  else
  {for(j=0;j<NC[i];j++)       /*calcul distante S1[j][k]*/
```

```
    for(k=j+1;k<NC[i];k++)
        {S1[j][k]=/*(float)n[i][j]*S[i][j]+(float)n[i][k]*S[i][k]+*/(float)n[i][j]*(float)n[i][k]/((float)n[i][
j]+(float)n[i][k])*dist(i,j,k);
                printf(""); }

   min=S1[0][1];I[i]=0;J[i]=1;          /*determinare minim S1[j][k]*/
   for(j=0;j<NC[i];j++)
     for(k=j+1;k<NC[i];k++)
         if(S1[j][k]<min) {min=S1[j][k]; I[i]=j;J[i]=k;
                          }
         S2[i]=min;
   }
   }
   for(i=0;i<NB;i++)
   printf(" %d %d   ",I[i],J[i]);
   for(i=0;i<NB;i++)
   printf(" %5.4f",S2[i]);
   printf("\n");
   getch();
   /*Ordonare crescatoare a erorilor S2[i]*/
   for(i=0;i<NB;i++)
   S3[i]=S2[i];   /*de fapt se ordoneaza S3==S2,ca sa nu se piarda S2*/
   for(j=0;;j++)
   {y=0;
   for(i=0;i<NB-1;i++)
     {if(S2[i]>S2[i+1])
         {x1=S2[i];
         S2[i]=S2[i+1];
         S2[i+1]=x1;
         y=y+1;}
     }
     if (y==0) break;
   }
   printf("\n");
   for(i=0;i<NB;i++)
   printf("  %5.3f",S2[i]);
   /*determinare loc*/
   for(j=0;j<NB;j++)
    for(i=0;i<NB;i++)
        if (S3[i]==S2[j]) {loc[i]=j;
                          }
    for(i=0;i<NB;i++)
     printf(" %d ",loc[i]);
     printf("\n");

   /*uniri in primii NB1/2 bucheti d.p.d.v. al erorilor minime,*/
   prag=(int)NB1/2;Nuniri=0;          /* incep cu buchetul aflat pe locul 0,*/
```

```
 for(j=0;j<NB;j++)       /* pina la locul NB1/2,fara buch.cu NC[i]=0 */
  {if(Nuniri==prag) break;
   for(i=0;i<NB;i++)
    if(loc[i]==j)

        if(NC[i]>0)       /*unirile se fac numai daca NC[i]>0*/
        {
        NC[i]=NC[i]-1;
        NV=NV-1;
        Nuniri=Nuniri+1;
         i1=I[i];j1=J[i];
         S[i][i1]=S3[i];  /*noua val.a erorii in clusterul obtin.prin unire*/
         for(k=0;k<dim;k++)
         {*(ptr+adresa(i,i1,k))=((float)n[i][i1]*tabl(ptr,i,i1,k)+(float)n[i][j1]*tabl(ptr,i,j1,k))/((float)n[i][i
1]+(float)n[i][j1]);
         printf("");}
         /*noua valoare a centroidului */
        n[i][i1]=n[i][i1]+n[i][j1]; /*noul nr.de vect./cluster     */
        for(s=j1;s<NC[i];s++)
        {n[i][s]=n[i][s+1]; /* decalari in sus ale nr.de vect/clust,erorii si  */
         S[i][s]=S[i][s+1]; /* centroidului datorita disparitiei clusterului j1  */
           for(k=0;k<dim;k++)
           *(ptr+adresa(i,s,k))=tabl(ptr,i,s+1,k);
        }
        break;
     }

    if(NV==256) break;
   }
 if(NV==256) break;
 suma=0;
 for(i=0;i<NB;i++)
 {suma=suma+NC[i];
  printf(" %d %d;",NC1[i],NC[i]);}
  printf("\n%d \n",suma);
 for(i=0;i<NB;i++)
   if(NC[i]>0)
    if(NC[i]==2)
        {printf("\n buchetul %d se alatura altui buchet",i);
         NC[i]=0;NB1=NB1-1;
         for(u=0;u<NB;u++)   /*calcul dist.intre CG[i] si CG[u]*/
          if(NC[u]>0)  /*buch. u sa nu fi fost deja alaturat!*/
          {dis[u]=0;
           for(k=0;k<dim;k++)
            {x=CG[i][k];y=CG[u][k];
             dis[u]=dis[u]+(x-y)*(x-y);}
          }
         for(u=0;u<NB;u++)   /*prima dist.calcul.este primul minim*/
```

```
          if(NC[u]>0)
            if(NC[u]<=NC1[u]+0) /*sa nu se alature unui buch. care a mai"absorbit*/
             {min=dis[u];umin=u;break;}   /*  bucheti"  */
        for(u1=u+1;u1<NB;u1++)   /*se gaseste dist.minima*/
          if(NC[u1]>0)
            if(NC[u1]<=NC1[u1]+0) /*la fel ca mai sus!*/
             if(dis[u1]<min)  {min=dis[u1];umin=u1;}
        printf("\numin=%d,pt.care NC1=%d, NC=%d ",umin,NC1[umin],NC[umin]);
/*      getch();          */
        for(k=0;k<dim;k++)   /*noul centr.gen.in buchetul umin*/
          {CG[umin][k]=(NC1[i]*CG[i][k]+NC1[umin]*CG[umin][k])/(NC1[i]+NC1[umin]);
           printf("");}
        for(k=0;k<dim;k++) /*mutarea celor 2 centr.din buch.i in buch.umin*/
          {*(ptr+adresa(umin,NC[umin],k))=tabl(ptr,i,0,k);
           *(ptr+adresa(umin,NC[umin]+1,k))=tabl(ptr,i,1,k);} /*Aici!!*/
        printf("\n");
        for(k=0;k<dim;k++)
         printf(" %5.1f",tabl(ptr,umin,NC[umin],k));/*centroid pt.clust.1*/
        printf("\n");          /*dupa mutarea in noul buch.umin*/
        for(k=0;k<dim;k++)
         printf(" %5.1f",tabl(ptr,umin,NC[umin]+1,k)); /*la fel pt.cl.2*/
        printf("\n");
        for(k=0;k<dim;k++)
         printf(" %5.1f",tabl(ptr,i,0,k));  /*centr.pt.cl.1 inainte de mutare*/
         printf("\n");
        for(k=0;k<dim;k++)              /*la fel,pt.cl.2*/
         printf(" %5.1f",tabl(ptr,i,1,k));
         k=NC[umin];
        S[umin][k]=S[i][0]; /*erorile celor 2 clust.se atas.in buch umin*/
        S[umin][k+1]=S[i][1];      /*Aici!!!*/
        n[umin][k]=n[i][0]; /*nr.de vectori/cluster din buch.i se transf.*/
        n[umin][k+1]=n[i][1]; /*Aici!!!*/          /* in buchetul umin*/
        NC[umin]=NC[umin]+2;/*nr de clust.in buch.obtinut prin adugarea buch.i*/
        }  /*sfirs.unirii clusterilor*/
}
printf("\n %d \n",ii);
suma=0;
for(i=0;i<NB;i++)
{printf(" %d ",NC[i]);
suma=suma+NC[i];    }
printf("\n%d\n ",suma);
 for(i=0;i<NB;i++)
  {for(j=0;j<NC[i];j++)
   {for(k=0;k<dim;k++)
    printf("%5.5f ",tabl(ptr,i,j,k));
    printf("\n");}
    printf("\n i=%d",i);/*getch();*/  }
    if(!(fp3=fopen("c:\\home\\mischie\\car710.dat","w+")))
```

```
{puts("nu se poate deschide fis3");
exit(1);}
   for(i=0;i<NB;i++)
     for(j=0;j<NC[i];j++)
      for(k=0;k<dim;k++)
           fprintf(fp3,"%5.2f ",tabl(ptr,i,j,k));
fclose(fp2);

}
```

BIBLIOGRAFIE

[1] A. Gersho, R.M. Gray, "Vector Quantization and Signal Compresion", Kluwer Academic Publishers, London 1993.

[2] V. Ramasubramanian, K. Paliwal, "Fast *K*-Dimensional Tree Algorithms for Nearest Neighbor Search with Application to Vector Quantization Encoding", IEEE Trans. on Signal Processing, pp.518-531, March 1992.

[3] N. Moayeri, D. Neuhoff, "Time-Memory Tradeoffs in Vector Quantizer Codebook Searching Based on Decision Trees", IEEE Trans. on Speech and Audio Processing, pp.490-506,October 1994.

[4] K. Paliwal, B. Atal, "Efficient Vector Quantization of LPC Parameters at 24 Bits/Frame", IEEE Trans. on Speech and Audio Processing, pp.3-14, January 1993.

[5] W. Le Blanc, B. Bhattacharya, S. Mahmoud, V. Cuperman, "Efficient Search and Design Procedures for Robust Multi-Stage VQ of LPC Parameters for 4 kb/s Speech Coding", IEEE Trans. on Speech and Audio Processing, pp.373-385, October 1994.

[6] W. Gardner, B. Rao, "Theoretical Analysis of the High-Rate Vector Quantization of LPC Parameters", IEEE Trans. on Speech and Audio Processing, pp.367-381, September 1995.

[7] F. Song, B. Huang, "Optimal Quantization on LSP Parameters", IEEE Trans. on Speech and Audio Processing, pp.15-24, January 1993.

[8] M. Xie, J. Adoul, "Algebric Vector Quantization of LSP Parameters with Low Storage and Computational Complexity", IEEE Trans. on Speech and Audio Processing, pp.234-239, May 1996.

[9] R. Hagen, "Robust LPC Spectrum Quantization- Vector Quantization by a Linear Mapping of a Block Code", IEEE Trans. on Speech and Audio Processing, pp.266-280, July 1996.

[10] A. Buzo, A. Gray, R.M. Gray, J. Markel, "Speech Coding Based Upon Vector Quantization", IEEE Trans. on Acoustics Speech and Signal Processing, pp.562-574, October 1980.

[11] Y. Linde, A. Buzo, R. M.Gray, "An Algorithm for Vector Quantizer design", IEEE Trans. Communications, pp.84-95, January 1980.

[12] B.Juang, D. Gray, A. Gray, "Distorsion performance of vector quantization for LPC voice coding", IEEE Trans. on Acoustics Speech and Signal Processing, pp.294-303, April 1982.

[13] R. Ramachandran, M. Sondhi, N. Seshadri, B. Atal, "A Two Codebook Format for Robust Quantization of Line Spectral Frequencies", IEEE Trans. on Speech and Audio Processing, pp.157-168, May 1995.

[14] W. Equitz, "A new Vector Quantization Clustering Algorithm", IEEE Trans. on Acoustics Speech and Signal Processing, pp.1568-1575, October 1989.

[15] R. Laroia, N. Phamdo, N. Farvardin, "Robust and Efficient Quantization of Speech LSP Parameters using Structured Vector Quantizers", presented in ICASSP '91 Toronto, Canada, April 1991.

[16] K. Malone, T. Fischer, "Enumeration and Trellis-Searched Coding Schemes for Speech LSP Parameters", IEEE Trans. on Speech and Audio Processing, pp.304-315, July 1993.

[17] M. Ferrer-Ballester, A. Figueiras-Vidal, "Efficient Adaptive Vector Quantization of LPC Parameters", IEEE Trans. on Speech and Audio Processing, pp.314-317, July 1995.

[18] M. Balacrishnan, W. Perlman, L. Lu, "Variable-Rate Tree Structured Vector Quantizers", IEEE Trans. on Information Theory, pp.917-930, July 1995.

[19] N. Moayeri, D. Neuhoff, W. Stark, "Fine Coarse Vector Quantization", IEEE Trans. on Signal Processing, pp.1503-1515, July 1991.

[20]D. Lee, S. Baeck, K. Sung, "Modified *K*-means Algorithm for Vector Quantizer Design", IEEE Signal Processing Letters, pp.2-4, January 1997.

[21] Y. Hussain, N. Farvardin, "Variable-Rate Finite-State Vector Quantization and Application to Speech and Image Coding", IEEE Trans. on Speech and Audio Processing, pp.25-38, January 1993.

[22] F. Jean, H. Wang, "Transparent Quantization of Speech LSP Parameters Based on KLT and 2-D-Prediction", IEEE Trans. on Speech and Audio Processing, pp.60-65, January 1996.

[23] P. Laurent, "Expression of Spectral Distorsion using Line Spectrum Frequencies", IEEE Trans. on Speech and Audio Processing, pp.481-483, September 1997.

[24] J. Pan, T. Fischer, "Vector Quantization of Speech Line Spectrum Pair Parameters and Reflection Coefficients", IEEE Trans. on Speech and Audio Processing, pp.106-115, March 1998.

[25] K. Law, C.Chan, "Split Dimension Vector Quantization of Parcor Coefficients for Low Bit Rate Speech Coding", IEEE Trans. on Speech and Audio Processing, pp.443-446, July 1994.

[26] A. Gersho, V. Cuperman, "Vector Quantization. A pattern matching technique for speech coding", IEEE Communication Magazine, pp 15-21, December 1983.

[27] R. M.Gray, "Vector Quantization", IEEE ASSP Magazine, pp.4-29, April 1984.

[28]L. Rabiner, R. Schafer, "Digital Processing of Speech Signal", Prentice-Hall, New Jersey, 1978.

[29] A. Oppenheim, R. Schafer, "Digital Signal Processing", Pretince Hall, New Jersey, 1975.

[30] C. Marven, G. Ewers, "Digital Signal Processing", Texas Instruments, 1993.

[31] P. Swasek, "Quantization", University of Rhode Island, 1985.

[32] B. Atal, V. Cuperman, A. Gersho, "Speech and Audio Coding for Wireless and Network Applications", Kluwer Academic Publishers, London 1993.

[33] P. Cosman, K. Oehler, E. Riskin, R. Gray, "Using Vector Quantization for Image Processing", IEEE Proceedings, pp.1326-1341, September 1993.

[34] C. Wu, J. Chen, "A Novel Two Level Method for the Computation of the LSP Frequencies Using a Decimation-in-Degree Algorithm", IEEE Trans. on Speech and Audio Processing, pp.106-115, March 1997.

[35] C. Nasar, M. Soleymani, "Codebook Design for Trellis Quantization Using Simulated Annealing", IEEE Trans. on Speech and Audio Processing, pp.400-405, October 1993.

[36] W. Chan, A. Gersho, "High Fidelity Audio Coding with Generalized Product Code VQ", Speech and Audio Coding for Wireless and Network Applications, Kluwer Academic Publishers, pp.153-160,London 1993

[37] N. Phamdo, N. Farvardin, T. Moriya, "Combined Source-Channel Coding of LSP Parameters Using Multi-Stage Vector Quantization", Speech and Audio Coding for Wireless and Network Applications, Kluwer Academic Publishers, pp181-190, London 1993

[38] S. Wang, E. Paksoy, A Gersho, "Product Code Vector Quantization of LPC Parameters", Speech and Audio Coding for Wireless and Network Applications", Kluwer Academic Publishers, pp.252-258, London 1993

[39] E. Pop, V. Tiponuț, I. Naforniță, A. Mihăescu, L. Toma, "Metode în Prelucrarea Numerică a Semnalelor", vol. I și II, Ed. Facla Timișoara, 1986.

[40] A. Millea, "Măsurări electrice. Principii și metode", Ed. Tenhică București, 1980.

[41] L. Toma, A. Stoian, "Predicția liniară", Catedra de Electronică Aplicată, I.P. Timișoara, 1987.

[42] S. Mischie, L. Toma, "Programs for Serial Communication between a M80-UC Microsystem and an IBM Personal Computer", ICATE '91, pp.E29, Craiova 1991.

[43] S. Mischie, L. Toma, "Quantization of LPC Parameters", Proceedings of the Symposium on Electronics and Telecommunications, Timișoara, vol. III, pp.145-150, 1994.

[44] S. Mischie, A. Iacovliev, "An Algorithm for Vector Quantizer Design using Pairwise Nearest Neighbor", Proceedings of the Symposium on Electronics and Telecommunications, Timișoara, vol. II, pp.99-104, 1996.

[45] A. Iacovliev, S. Mischie, "Determination of Prediction Coefficients of Speech with Digital Signal Processor (DSP)", Proceedings of the Symposium on Electronics and Telecommunications, Timișoara, vol. II, pp.176-179, 1996.

[46] S. Mischie, "Structurarea cărții de cod a unui cuantizor vectorial de tip Voronoi în vederea cuantizării vectoriale prin metoda combinată arbore-căutare totală", Buletinul Științific al Universității "Politehnica" din Timișoara, Tom 43/57 Electrotehnică, Electronică, Comunicații, 1998.

[47] S. Mischie, D. Stoiciu, L. Toma, "Efficient Computation of LPC Power Spectral Density at Speech LSF's", Buletinul Științific al Universității "Politehnica" din Timișoara, Tom 43/57 Electrotehnică, Electronică, Comunicații, 1998.

[48] S. Mischie, A. Iacovliev, "Performance Evaluation of Unstructured, Tree-Structured and Tree Searched-Unstructured Vector Quantizers for LSF's", Proceedings of the Symposium on Electronics and Telecommunications, Timișoara, 1998.

[49] S. Mischie, "Calculul erorilor în procesul de cuantizare", referatul nr.1 pentru doctorat, 1994.

[50] S. Mischie, "Algoritmi de cuantizare vectorială", referatul nr.2 pentru doctorat, 1996.