UNIVERSITATEA "POLITEHNICA" TIMIŞOARA
FACULTATEA DE ELECTRONICĂ ŞI TELECOMUNICAŢII

TEZĂ DE DOCTORAT

REŢELE NEURONALE LOCAL LINIARE PENTRU COMANDA ADAPTIVĂ A ROBOŢILOR CU REACŢIE VIZUALĂ

ing. ANDREI CIMPONERIU

Conducător ştiinţific
Prof. dr. ing. ANTON POLICEC

Cuprins

Prefață

Această teză de doctorat este o premieră în domeniul Rețelelor Neuronale (RN), la Universitatea "Politehnica" din Timișoara. Tema inițială a tezei a fost "Recunoașterea semnalelor utilizând rețele neuronale", și ea căuta să îmbine preferința autorului spre domeniul RN cu experiența existentă în Facultatea de Electronică și Telecomunicații din Timișoara, în prelucrarea semnalelor. Însă după pregătirea generală în domeniul ales, materializată prin prezentarea primului referat pentru doctorat (Cimponeriu, 1994a), autorul a avut șansa efectuării unui stagiu în Laboratorul de Automatică TROP al Université de Haute-Alsace, Mulhouse, Franța, unde i s-a oferit posibilitatea aplicării cunoștințelor acumulate, în domeniul comenzii adaptive, în particular al comenzii neuronale a unui braț de robot înzestrat cu un sistem vizual. Acest stagiu, repetat după obținerea unor rezultate pozitive, a dus la orientarea tezei spre tematica și conținutul actual, conturate deja în cel de-al doilea referat. (Cimponeriu, 1994b).

Obiectul tezei este comanda cinematică adaptivă a unui braț de robot neredundant, cu 3 grade de libertate. Sistemul vizual, format din două camere CCD fixe, dă pozițiile de interes sub formă de vectori 4-dimensionali. În teză se studiază comanda cu rețele neuronale a brațului, care asigură poziționarea cleștelui robotului la țintă. Sunt studiate în detaliu chestiuni legate de modelarea local liniară a inversei compusei dintre transformarea vizuală și transformarea cinematică inversă a brațului. Abordarea generală, prin pseudoinversa Moore-Penrose, permite generalizarea rezultatelor obținute, pentru mai multe grade de libertate, și pentru reprezentări mai complexe ale pozițiilor în spațiul vizual.

Primul capitol este un studiu bibliografic, ce face prezentarea domeniului tezei. Sunt prezentate, pe scurt, noțiunile esențiale ale domeniului RN, mai exact ale rețelelor cu propagare înainte (*feedforward*). Dintre acestea, sunt prezentate mai în detaliu rețelele de aproximare locală, și mai ales rețelele cu aproximări local liniare, care sunt aprofundate în teză. De asemenea, este prezentat succint domeniul modelării adaptive, în particular al comenzii adaptive pentru roboți cu reacție vizuală, în prezența erorilor de comandă.

Capitolele următoare conțin contribuții ale autorului la domeniul studiat. După prezentarea platformei robotice și a modelării ei, capitolul al doilea realizează o analiză a erorilor admisibile ce asigură convergența poziționării, în prezența erorilor inerente comenzii neuronale. Suportul matematic al analizei convergenței este obținut prin introducerea transformării Z vectoriale, și prin aplicarea ei la sisteme cu reacție de tipul celui din teză. În continuare se face o analiză detaliată a modelului local liniar și local pătratic al transformării cinematice inverse a brațului. Cu toate că este luat în considerare doar brațul de robot, presupunându-se că sistemul vizual este cunoscut cu precizie și nu introduce erori, acest capitol permite desprinderea de concluzii asupra erorilor pătratice maxime admisibile, pentru care poziționarea este convergentă, precum și domeniul de validitate al modelării local liniare.

În capitolul al treilea sunt comparați trei algoritmi de învățare a modelelor local liniare: algoritmul LMS, algoritmul Greville și algoritmul RLS. Primul și cel de-al treilea sunt cunoscuți în filtrarea adaptivă, iar al doilea este obținut de autor pe baza teoremei lui Greville

cunoscute din literatură. Prin intermediul algoritmului Greville, se demonstrează posibilitatea aplicării algoritmului RLS în locul algoritmului LMS în situația de față, în care vectorii de intrare sunt liniar dependenți. Prin aceasta se obțin creșteri spectaculoase în viteza de convergență față de soluțiile prezente în literatură, ce utilizează algoritmul LMS.

Capitolul al patrulea studiază învățarea adaptivă prin aproximare local liniară. Se urmărește obținerea unui minim de informație ce permite realizarea poziționărilor dorite. Spre deosebire de soluțiile din literatură, RN propusă aici nu se dezvoltă decât pentru regiunile de lucru, învățarea unui număr redus de traiectorii fiind posibilă cu doar câțiva neuroni. Prin introducerea adaptării *on-line* a matricii de comandă, pe baza informației disponibile pe durata deplasării, se arată prin simulări că se poate reduce sensibil numărul de aproximări local liniare necesare. Conjugate cu învățarea prin algoritmul RLS, aceste îmbunătățiri fac ca algoritmul de comandă obținut și RN asociată propusă, să aibă o eficiență deosebită, superioară soluțiilor întâlnite în bibliografie.

În capitolul al cincilea este sintetizat conținutul tezei și sunt trecute în revistă contribuțiile originale ale autorului. Sunt, de asemenea, prezentate generalizările și direcțiile de cercetare ce pot fi conturate la sfârșitul perioadei de pregătire a doctoratului.

Teza conține și două anexe. Prima dintre ele prezintă noțiunile cele mai importante de algebră liniară, necesare pentru urmărirea în detaliu a calculelor matriciale efectuate. Ea poate prezenta interes și ca material de fundament în limba română, pentru o abordare actuală a domeniului prelucrării semnalelor. Cea de-a doua anexă conține listingul fișierelor MATLAB mai importante, cu care au fost realizate simulările din capitolul 4.

Pentru trimiterile bibliografice de pe parcursul lucrării, în locul notației utilizate de IEEE, [ ], autorul a preferat notația întâlnită în revistele *Neural Computation* și *Neural Networks*, ca și în unele volume, constând din numele primului sau primilor doi autori și anul publicării. S-a considerat că aceasta dă mai multă informație, fiind mai comodă cititorului, iar autorului îi permite, la o revizie ulterioară, o actualizare mai simplă, fără renumerotarea întregii liste bibliografice.

Mulțumiri

În primul rând, trebuie să mulțumesc conducătorului științific, prof. dr. ing. Anton Policec. Deschiderea dânsului mi-a permis abordarea acestui domeniu nou pe care îl constituie Rețelele Neuronale. Spiritul dânsului de colaborare au făcut posibile stagiile pe care le-am efectuat în laboratorul TROP al Université de Haute-Alsace (UHA) din Mulhouse, Franța, care au concentrat aplicarea RN în domeniul roboticii. Flexibilitatea dânsului și libertatea totală acordată au permis ca această teză să îmi reflecte interesele și modul personal de a aborda tema aleasă.

În al doilea rând, mulțumiri i se cuvin domnului Julien Gresser, profesor la UHA, șeful Laboratorului de Automatică TROP, în a cărui echipă am putut desfășura o activitate de cercetare însumând circa 10 luni. Buna colaborare cu echipa TROP și cu dânsul în special, ca și sprijinul moral și material acordat, se reflectă în câteva lucrări comunicate sau publicate în comun. Bibliografia disponibilă în laboratorul TROP sau procurată la cerere, ca și posibilitatea utilizării serviciului de căutare și schimb bibliografic al UHA, mi-au permis desfășurarea în condiții bune a informării bibliografice aferente tezei. În fine, îi mulțumesc pe această cale domnului Gresser pentru generozitatea sa, care mi-a făcut posibilă participarea la Școala de primăvară de Rețele Neuronale de la Université de Montreal, în aprilie 1996

Doresc să mulțumesc domnului prof. dr. ing. Eugen Pop, al cărui asistent am fost în perioada de început a tezei, și de al cărui ajutor m-am bucurat la fiecare solicitare. Calitățile sale umane și spiritul său pătrunzător au fost și sunt un exemplu căruia îi datorez recunoștință.

Este cazul de a fi menționat aici sprijinul pe care Catedra de Măsurări și Electronică Optică, din fac parte, îl acordă pregătirii doctoratului. În condițiile doctoratului fără frecvență pe care l-am urmat, care a presupus și sarcini didactice, înțelegerea de care a dat dovadă colectivul catedrei pe durata suficient de îndelungată a pregătirii tezei a fost o condiție indispensabilă obținerii de rezultate în cercetare.

Pe parcursul elaborării tezei sau a unor articole cuprinse în ea, m-am bucurat de ajutorul lui Andrei Török, căruia îi aduc mulțumiri. Deschiderea unui matematician către tehnică nu poate decât să favorizeze și procesul reciproc, atât de necesar în obținerea de performanțe în stadiul actual de dezvoltare al electronicii, în care rolul tranzistorului este luat de procesorul de semnal, iar în loc de cositor și sacâz se folosesc matrici și norme. În același context, doresc să-i mulțumească lui Aldo De Sabata care, cu prilejul publicării unui articol în Buletinul Științific al Politehnicii din Timișoara, mi-a demonstrat că efectuarea câtorva iterații între recenzent și autor au ca efect convergența într-o lucrare de o calitate superioară.

Pe durata stagiilor la Mulhouse, ca și a școlii de la Montreal, am profitat de prietenia lui Hubert Kihl și a lui Jean-Philippe Urban. De asemenea, aici este locul să menționez colegii de cabinet, ale căror discuții și încurajări au venit să completeze latura exclusiv intelectuală solicitată de lectura unor articole sau de conceperea și analizarea unor simulări pe calculator.

În sfârșit, în același cadru personal, sper ca această lucrare să fie la înălțimea așteptărilor părinților mei, care prin interes și răbdare au participat la elaborarea ei.

Timișoara, 20.12.1996

Andrei Cimponeriu

Notații și tipuri de caractere

caractere italice : cuvinte în limba engleză sau prescurtări uzuale provenind din limba latină; exemple: *feedforward*; *winner-takes-all*; *e.g.*- *exempli gratia*, de exemplu; *et al.*- *et alii*, și alții.

caractere Courier : simboluri și programe utilizate în simulările pe calculator; exemple: `vtarg, Jc, th1`

caractere Arial : definiții, *e.g.* **Neuronul**, **Rețeaua neuronală** (§ 1.1.1)

mulțimi : $\mathcal{A}$
$\mathcal{A}^{\perp}$ complementul ortogonal al lui $\mathcal{A}$
$\mathbf{R}$ ($\mathbf{C}$) mulțimea numerelor reale (complexe)
$\mathbf{R}^n$ mulțimea numerelor reale n-dimensionale

matrici : $\mathbf{A}$
$\tilde{\mathbf{A}}$ o aproximare a matricii $\mathbf{A}$
$\hat{\mathbf{A}}$ estimarea matricii $\mathbf{A}$ obținută printr-un procedeu iterativ
$\mathbf{A}^T$ transpusa lui $\mathbf{A}$
$\mathbf{A}^{m \times n}$ matrice cu m linii, n coloane
$\mathbf{I}$ matricea unitate, cu dimensiunea cerută de context

vectori : $\mathbf{a}, \theta$

funcții : f, g

scalari : a, α

Prescurtări des utilizate :

ART	*Adaptive Resonance Theory*, teoria rezonanței adaptive (§ 1.2.1)
BP	*Backpropagation*, algoritmul propagării înapoi a erorilor (§ 1.1.3)
CMAC	*Cerebellar Model Articulation Controller*, controlerul articulațiilor (bazat pe) modelul cerebelului (§1.2.3)
LLM	*Local Linear Mappings*, funcții (aproximări) local liniare (§ 1.2.3)
LMS	*Least Mean Squares*, (algoritmul erorii) pătrate medii minime (§ 3.1)
RBF	*Radial Basis Functions*, rețele cu funcțiile bazei radiale (§1.2.2)
RLS	*Recursive Least Squares*, (algoritmul erorii) pătrate minime recursive recursive (§ 3.4)
RN	rețele neuronale (§ 1.1.1)
SOM	*Self-Organizing Map*, harta autoorganizantă a lui Kohonen (§ 1.2.1)

1. Comandă adaptivă cu rețele neuronale

1.1. Rețele neuronale - prezentare generală [1]

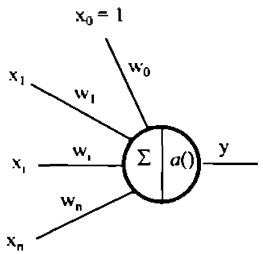
1.1.1. Introducere, definiții, clasificare

Rețelele neuronale (RN) constituie un domeniu actual de cercetare, destul de eterogen, cuprinzând specialiști în biologie, matematică aplicată, statistică, știința calculatoarelor, științele inginerești și în special robotică, tehnologi de componente electronice, fizicieni. Există reviste ce le sunt dedicate exclusiv, precum *Neural Computation* (MIT Press, bilunară, apare din 1988). *Neural Networks* (Pergamon Press, 9 numere/an, 1987), *IEEE Transactions on Neural Networks* (IEEE, bilunară, 1989). Articole pe această temă apar și în *Neural Processing Letters*, *Biological Cybernetics*, *Physical Review*, *Pattern Recognition*, *IEEE Transactions on Systems, Man and Cybernetics*, *IEEE Trasactions on Acoustics, Speech and Signal Processing*, *IEEE ASSP Magazine*, *IEEE Transactions on Computers*. Modele și aplicații pot fi găsite și în *Journal of Systems Engineering* (Springer,1991), *Applied Intelligence* (Kluwer, 1993), *Neural Computing and Applications* (Springer, 1993). În ultimii ani au început să apară monografii sau manuale, o selecție personală fiind volumele editate sau scrise de Rumelhart și McClelland (1987), Hecht-Nielsen (1989), Hertz, Krogh și Palmer (1991), Cichocki și Unbehauen (1992), Haykin (1994), Ripley (1996). Toate acestea au în comun modele adaptive, descoperite sau inspirate din biologie, dar unde matematica, în special prin capitole ale analizei numerice, precum optimizarea, interpolarea sau statistica, au un cuvânt greu de spus. Aplicațiile cele mai importante sunt modelarea adaptivă, inclusiv predicția seriilor de timp financiare sau comanda adaptivă a roboților, recunoașterea formelor (*pattern recognition*) prin recunoașterea vorbirii sau a scrisului, și optimizarea. Oferind soluții caracterizate în primul rând prin adaptabilitate și doar în al doilea rând prin precizie, RN se aplică în general cu succes (care este adesea superior tehnicilor convenționale) în probleme imprecise prin însăși natura lor, caracterizate prin date zgomotoase, indescriptibilitate analitică sau nestaționaritate. Iar tratarea acestor probleme, adesea empirică, dar uneori riguros matematică, a dus deja la rezultate interesante în teoria aproximării (*e.g.* Hornik, Sinchcombe și White, 1990) sau în teoria învățării (Vapnik, 1995), ca și la realizări tehnice performante în domeniile de aplicații menționate. Circuite integrate neuronale, analogice sau numerice au apărut deja și continuă să apară, mai ales în laboratoarele de cercetare, dar și pe piață (Graf, 1996). Caracterizate prin paralelism masiv, acestea sunt rapide și au un consum mic. Ele sunt în general mai ieftine decât rudele lor de uz general, dar deocamdată au dezavantajul de a fi dedicate unei anumite aplicații.

În continuare se prezintă terminologia de bază a domeniului rețelelor neuronale.

[1] Acest subcapitol reia, într-o formă revizuită, chestiuni prezentate în (Cimponeriu, 1994a).

Neuronul (artificial) sau elementul de calcul al rețelei neuronale (artificiale) este cel mai adesea întâlnit sub forma din fig.1.1.1, de inspirație biologică. Semnalele de intrare x_i sunt însumate prin ponderile w_i, ce modelează sinapsele. Suma obținută este aplicată apoi unei funcții de activare $a(\ )$, ce poate fi ca în figura 1.1.2. Valoarea acesteia dă semnalul de ieșire al neuronului, ce poate fi semnal de intrare pentru mulți alții. Termenul w_0x_0, cu $x_0 = 1$, permite deplasarea pe caracteristica funcției de activare. Poate fi remarcată generalitatea acestui element de calcul: dacă funcția de activare este liniară se obțin circuite liniare, un caz particular fiind filtrele transversale. Aproximativ același lucru se obține pentru funcția de activare sigmoidală (în S), dacă argumentul este suficient de mic. Dacă argumentul funcției sigmoidale este mare sau dacă se consideră funcția de activare treaptă, se obțin elemente binare, adecvate logicii binare sau clasificării. Funcția sigmoidală a fost găsită tocmai pentru aplicații de clasificare, ca aproximare diferențiabilă a funcției treaptă, permițând ajustarea ponderilor prin metoda gradientului (§ 1.1.3).



$$y = a\left(\mathbf{w}^T\mathbf{x}\right) = a\left(\sum_{i=0}^{n} w_i\, x_i\right),$$
$$\mathbf{x} = \begin{bmatrix} x_0 & \cdots & x_i & \cdots & x_n \end{bmatrix}^T \qquad (1.1.1)$$
$$\mathbf{w} = \begin{bmatrix} w_0 & \cdots & w_i & \cdots & w_n \end{bmatrix}^T$$

Figura.1.1.1. Neuron artificial.

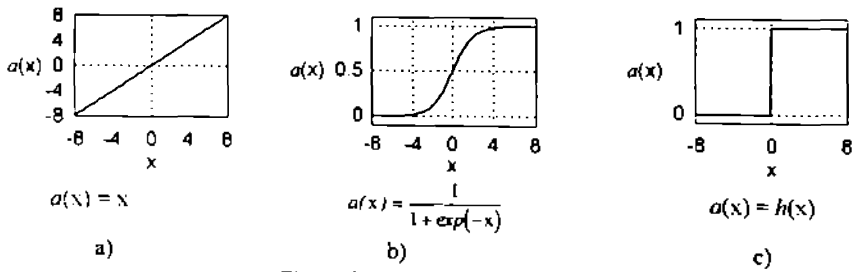


Figura 1.1.2. Funcții de activare
a) activare liniară, b) activare sigmoidală, c) activare în treaptă

Rețeaua neuronală (artificială) cuprinde mai multe elemente de calcul de acest tip, organizate în principiu oricum. Practic, cel mai des întâlnit tip de rețea este cel din figura 1.1.3. În această structură, de obicei elementele de pe stratul ascuns sunt sigmoidale, iar elementele de ieșire sunt liniare. Elementele de intrare se reprezintă pentru păstrarea tradiției; ele nu au rol funcțional, fiind simple repetoare. Numărul intrărilor și ieșirilor este dat de problema vizată. Numărul elementelor ascunse se ia suficient de mare pentru obținerea unei familii suficient de largi de funcții realizabile (§ 1.1.2) pentru interpolare, regresie sau clasificare, dar suficient de mic pentru ca parametrii ajustabili să poată fi estimați într-o măsură suficientă pe baza unei mulțimi de date de dimensiune acceptabilă (§ 1.1.4).

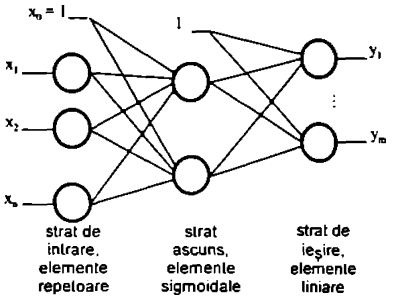


Figura 1.1.3. Reţea neuronală cu un strat ascuns

Se observă că elementele de calcul din figurile 1.1.2 a şi b au ca trăsătură comună faptul că ieşirea lor se modifică pentru toate valorile argumentelor; aproximările realizate prin RN ce conţin astfel de elemente sunt aproximări globale. Există şi elemente cu suport local, care intervin doar pentru valori ale argumentului situate într-o anumită zonă din domeniul de definiţie. Pentru aceasta se utilizează funcţii de activare locale ca în figura 1.1.4, cel mai adesea gaussiene (1.1.2).

$$a(\mathbf{x}) = exp\left(-\|\mathbf{w}_i - \mathbf{x}\|^2\right) \tag{1.1.2}$$

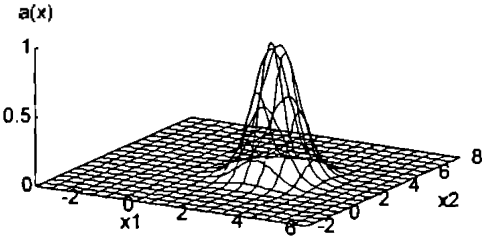


Figura 1.1.4. Funcţie de activare locală

De cele mai multe ori ponderile sau structura reţelei pot fi modificate pentru a se obţine un anumit comportament, dorit, al reţelei. Acest proces este denumit antrenare. Ea poate fi supravegheată (supervizată), ca în figura 1.1.5. Antrenarea supervizată se poate face dacă se dispune de vectori $\boldsymbol{d}(\mathbf{x})$ reprezentând ieşirile dorite pentru vectorii de intrare $\mathbf{x}$. Modificarea ponderilor se face pe baza unei funcţii de eroare $E(\boldsymbol{y}, \boldsymbol{d})$ ce măsoară distanţa între ieşirea obţinută $\boldsymbol{y}(\mathbf{x})$ şi $\boldsymbol{d}(\mathbf{x})$, printr-un algoritm ce minimizează funcţia E. Cel mai adesea se utilizează eroarea medie pătratică. Mulţimea perechilor de vectori $\mathbf{x}$, $\boldsymbol{d}(\mathbf{x})$ formează setul de antrenare. De obicei reţeaua antrenată este verificată pe date ce nu au fost prezentate la antrenare, ce formează setul de testare.

Dar antrenarea poate fi şi nesupravegheată (nesupervizată). In acest caz adaptarea se face doar pe baza semnalelor de intrare $\mathbf{x}$, astfel încât RN să determine trăsăturile importante ale acestora (analiza componentelor principale ale matricii de corelaţie a vectorilor

x), sau să realizeze împărțirea spațiului de intrare în anumite zone (cuantizare vectorială). Prin faptul că nu ia în considerare ieșirile, antrenarea nesupervizată este mult mai rapidă decât cea supervizată, dar uneori insuficientă. Există și rețele hibride, ce combină cele două tipuri de învățare.

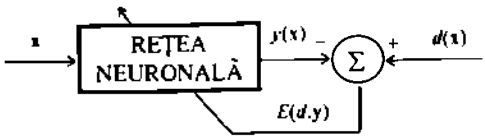


Figura 1.1.5. Antrenarea supervizată

RN sunt așadar sisteme adaptabile, ce prelucrează un semnal de intrare (vectorial) **x** și furnizează un semnal de ieșire (vectorial) **y**, funcție de arhitectura rețelei și de valorile tuturor parametrilor ajustabili din rețea. RN pot fi **statice** sau **dinamice**, după cum se poate sau nu considera că **x** și **y** sunt statici sau sunt funcții de timp. Cele dinamice pot fi cu timp continuu, când neuronii sunt descriși de ecuații diferențiale, ca în cazul implementării analogice, sau cu timp discret, pentru implementări numerice, când sunt descrise prin ecuații cu diferențe. Dacă la intrările rețelei se aplică și semnale de la ieșire, rețeaua este **recursivă**. Ea este **nerecursivă**, sau cu **cuplaj înainte** *(feedforward)* dacă nu există legături ieșire-intrare iar semnalele de intrare se propagă într-o singură direcție.

Vor face obiectul lucrării de față doar rețelele nerecursive statice, *feedforward*, cu antrenare supervizată.

1.1.2. Aproximarea funcțiilor cu rețele neuronale

Rețelele neuronale realizează funcții vectoriale neliniare. Prin adaptarea ponderilor sau structurii, funcția **y** calculată de o RN se poate apropia de o funcție dorită ***d***, descrisă prin perechi de vectori **x**, ***d***(**x**). Se întâmplă aceasta pentru orice funcție ***d*** ?

O proprietate foarte importantă a rețelelor feedforward este posibilitatea aproximării oricât de bune a funcțiilor continue definite pe mulțimi compacte (Hecht-Nielsen, 1990; Hornik *et al.*, 1989; Hornik, 1991; Funahashi, 1989; Blum și Li, 1991). Mai mult, s-a demonstrat că pot fi aproximate oricât de bine atât funcțiile cât și derivatele lor (Gallant și White 1992).

Considerarea funcțiilor definite pe intervale închise din $\mathbf{R}^n$ nu este restrictivă din punct de vedere tehnic, când semnalele sunt mărginite prin tensiunile de alimentare.

O funcție vectorială de variabilă vectorială,

$$f:[0,\ 1]^n \subset \mathbf{R}^n \rightarrow [0,\ 1]^m \subset \mathbf{R}^m$$

este echivalentă cu m funcții scalare de variabilă vectorială $f_j : [0,1]^n \rightarrow [0,1]$, $j = 1...m$. Proprietatea enunțată poate fi deci demonstrată pentru funcții scalare. Există mai multe demonstrații, ce urmează căi diferite, prezentate în (Cimponeriu, 1994a). Aici vor fi prezentate

pe scurt doar demonstrația lui Blum și Li, intuitivă, și cea a lui Hornik, Stinchcombe și White, foarte generală.

Blum și Li (1991) demonstrează că rețelele *feedforward* pot aproxima orice funcție continuă direct, prin superpoziția de "funcții simple", constante pe porțiuni. Neuronii folosiți sunt cu prag. Dacă funcția de aproximat f este unidimensională, se obține reprezentarea intuitivă a integralei Duhamel, aproximată (oricât de bine) printr-o sumă Riemann, ca în figura 1.1.6, care de fapt reprezintă o sumă Darboux inferioară. Pentru cazul n-dimensional, autorii obțin o rețea cu două straturi ascunse construind prin funcții treaptă celulele n-dimensionale pe care funcția aproximată e constantă. Fiecare interval obținut este activat doar când vectorul de intrare $\mathbf{x}$ se află în interiorul lui, ponderile de ieșire având valoarea funcției pe intervalul respectiv. Autorii dau și o estimare (grosieră) a numărului de neuroni necesari în rețea: de exemplu, dacă $f \in C^1[0,1]^n$ și $\|f'\| \le k$, iar eroarea maximă de aproximare este ε, atunci $\|f(\mathbf{x})-f(\mathbf{y})\| \le k \|\mathbf{x}-\mathbf{y}\|$ și se poate lua $m \ge k/\varepsilon$, rezultând pe primul strat m^n și pe al doilea $2n(m+1)$ elemente cu prag.

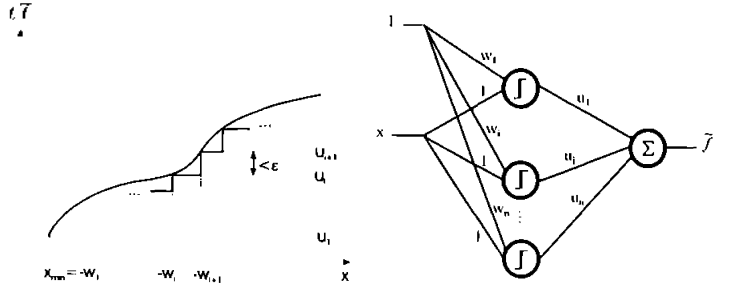


Figura 1.1.6. Aproximarea funcțiilor scalare prin funcții constante pe porțiuni
a) modul de aproximare, b) rețeaua corespunzătoare

O demonstrație întrucâtva asemănătoare este cea a lui Cardaliaguet și Evrard (1992), ce utilizează funcții în formă de clopot, construite astfel încât reprezentarea pentru N puncte ale funcției este exactă; pentru $N \to \infty$ aproximarea converge uniform spre funcția dorită.

Cea mai elegantă, generală și des citată demonstrație a posibilității de aproximare universală a rețelelor feedforward este cea dată de Hornik, Stinchcombe și White (1989). Principala lor teoremă este :

> Fie funcția de activare G orice funcție continuă neconstantă de la $\mathbf{R}$ la $\mathbf{R}$. Atunci mulțimea $\Sigma^r(G)$ a funcțiilor realizate de o rețea având un strat ascuns cu activări G și o ieșire liniară, vectorii ponderilor spre stratul ascuns dați în A_j și ponderile dinspre el spre ieșire β_j,
>
> $$\Sigma^r(G) = \left\{ \tilde{f}:\mathbf{R}\to\mathbf{R},\ \tilde{f}(x) = \sum_{j=1}^{n} \beta_j G(A_j(x)),\ j = 1,2,K \right\},\ \text{cu}$$
>
> $$x \in \mathbf{R},\ \beta_j \in \mathbf{R}, A_j \in \mathcal{A}^r = \left\{ A(x) = \mathbf{w}^T\mathbf{x} + b \mid \mathbf{w} \in \mathbf{R}^r, \mathbf{x} \in \mathbf{R}^r, b \in \mathbf{R} \right\},$$
>
> este uniform densă pe mulțimi compacte în $\mathcal{C}^r = \{\mathbf{R}^r \to \mathbf{R}, f \text{ continuă}\}$. Anterior ei reamintesc că o mulțime $\mathcal{S}$ din $\mathcal{C}^r$ se spune că este uniform densă pe compacte în $\mathcal{C}^r$ dacă pentru orice submulțime compactă $K \subset \mathbf{R}^r$, $\mathcal{S}$ este ρ_K-densă în $\mathcal{C}^r$, adică

pentru $(\forall)\varepsilon>0$ și $(\forall)f\in \mathcal{C}^r$, $(\exists)\tilde{f}\in\mathcal{F}$ astfel încât $\rho_K(f,\tilde{f})<\varepsilon$, cu $\rho_K(f,g) \equiv \sup_{x\in K}|f(x)-g(x)|$, $f,g\in \mathcal{C}^r$. Cu alte cuvinte, există în $\Sigma^r(G)$ funcții care aproximează uniform, oricât de bine într-un interval închis, în metrica din $\mathbf{R}$, orice funcție continuă definită pe $\mathbf{R}^n$.

Teorema nu este constructivă. Ei obțin și rezultate mai generale, pentru rețele $\Sigma\Pi$, în care elementele de pe stratul ascuns pot face produse între combinații liniare între intrări, și pentru funcții de aproximat măsurabile Borel (*e.g.* Spătaru, 1990), ce cuprind atât funcțiile continue cât și pe cele constante pe porțiuni sau discrete. Demonstrațiile lor se bazează pe teorema Stone-Weierstrass, adică pe structura de algebră a claselor de funcții Σ^r, $\Sigma\Pi^r$. Pentru rețelele $\Sigma\Pi$ sunt permise și funcții de activare discontinue.

Hornik, Stinchcombe și White (1990) au demonstrat că rețelele feedforward cu un singur strat ascuns pot aproxima, pe lângă funcțiile de mai multe variabile, și derivatele lor. Acest rezultat este necesar în unele aplicații precum robotica (Gallant și White 1992).

Hornik (1991) a arătat că aceste rezultate sunt valabile pentru funcții de activare oarecare, mărginite și neconstante, concluzionând că proprietățile acestui tip de RN de a fi aproximatori universali rezidă nu din alegerea funcției de activare, ci din arhitectura lor.

1.1.3. Algoritmul *Backpropagation* (propagarea înapoi a erorilor)

Pentru ca o RN să poată realiza un anumit rol funcțional prin aproximarea unei anumite funcții, ponderile ei trebuie ajustate, pe baza setului de antrenare. Cel mai des utilizat algoritm de modificare a ponderilor rețelelor *feedforward* este algoritmul propagării înapoi a erorilor, *Error Backpropagation*, sau doar *Backpropagation* (Rumelhart *et al.*, 1986). Ea mai este cunoscută și ca regula delta generalizată sau regula perceptronului cu mai multe straturi. Ea a fost descrisă în (Cimponeriu, 1994a), și este prezentată în limba română și în lucrarea (Todoran *et al.*, 1994). Algoritmul constă în modificarea ponderilor rețelei proporțional cu inversul gradientului funcției de eroare, care este eroarea medie pătratică dintre ieșirea dorită $d(\mathbf{x})$ și cea calculată de rețea $y(\mathbf{x})$ pentru intrarea $\mathbf{x}$. Pentru a se putea calcula gradientul, este necesar ca funcțiile de activare din rețea să fie diferențiabile, ca sigmoida din figura 1.1.2 b. Ea are avantajul de a necesita doar calcule locale, ce presupun, pentru un anumit element, cunoașterea doar a ponderilor și a valorilor funcțiilor ce îl leagă de restul rețelei, și nu ale tuturor elementelor din rețea. Algoritmul este simplu, iar rezultatele obținute în aplicații au dus la resuscitarea interesului pentru acest domeniu, după o perioadă de relativ declin, între 1967 și 1982 (Hecht-Nielsen, 1990).

Dezavantajele algoritmului *Backpropagation* (BP) sunt viteza scăzută de convergență spre minim și posibilitatea convergenței într-un minim local. Există numeroase studii și îmbunătățiri ale algoritmului BP, prezentate în (Cimponeriu, 1994a). O trecere în revistă a algoritmilor de antrenare a RN este făcută și de (Smagt, 1994), iar o lucrare excelentă dedicată optimizării, metoda gradientului fiind cea mai simplă metodă de găsire a minimului unei funcții, este (Gill *et al.*, 1981).

Dacă toate elementele din rețea sunt liniare, algoritmul BP devine mai simplu regulă Widrow-Hoff sau regulă delta (Widrow și Hoff, 1960), sau algoritmul LMS, cunoscut în filtrarea adaptivă (Goodwin & Sin, 1984; Widrow & Stearns, 1985; Haykin, 1991). Față de

rețelele neliniare. funcția de eroare a rețelelor liniare este un hiperparaboloid ce nu are minime locale. Însă RN liniare pot realiza doar funcții liniare. o clasă mult mai săracă decât cea a funcțiilor realizabile prin RN neliniare (§ 1.1.2).

1.1.4. Generalizarea

În limbajul cotidian prin generalizare se înțelege extragerea trăsăturilor comune din mai multe exemple particulare, sau extinderea trăsăturilor comune unui grup de obiecte. tuturor obiectelor sau fenomenelor din clasa respectivă (DEX). În RN. generalizarea este măsura a cât de bine se comportă rețeaua în problema reală. odată antrenarea încheiată. datorită faptului că învățarea nu se face pe o infinitate de vectori. care să cuprindă și cazurile reale pe care rețeaua va funcționa după antrenare. Un termen tehnic apropiat poate fi cel de interpolare.

În practică antrenarea și verificarea corectitudinii funcționării rețelei se fac pe mulțimi (seturi) distincte. Erorile ce se obțin pe aceste mulțimi. în funcție de numărul de cicluri de antrenare. se prezintă ca în figura 1.1.7. (Hecht-Nielsen. 1990). Adesea criteriul de oprire a antrenării este obținerea minimului erorii pe setul de testare.

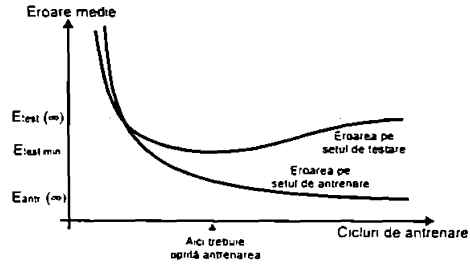


Figura 1.1.7. Erorile la antrenare și la funcționarea reală.

În figura 1.1.7. efectuarea antrenării prin parcurgerea setului de antrenare de un număr foarte mare de ori are ca efect scăderea erorii până la o valoare finală, minimă : rețeaua s-a adaptat maxim posibil. pentru numărul dat de parametri ai săi. la această mulțime. Pentru setul de testare. pe care nu s-a făcut antrenarea. adaptarea prea exactă la setul de antrenare devine dezavantajoasă. de la un anumit moment la continuarea antrenării eroarea începînd să crească. Aceasta este supraantrenarea. iar explicația ei constă în aceea că rețeaua. având mulți parametri ajustabili. poate fi făcută să aproximeze foarte bine valorile din mulțimea de antrenare. pentru valori ale lui x pe care nu s-a facut antrenarea. comportarea să fie mai puțin bună. Figura 1.1.8 prezintă rezultatul unei simulări[2] pentru o rețea având 40 de elemente ascunse. antrenată pe 21 de puncte. Cu toate că după antrenare funcția dorită d(x) este învățată corect în punctele x din setul de antrenare. pentru celelalte valori x erorile de interpolare sunt mari. Ar fi trebuit ca rețeaua să aibă mai puține elemente ascunse. sau ca punctele de antrenare să fie mai numeroase.

[2] demolm2.m scrisă de Beale (1992) pentru Neural Networks Toolbox. MATLAB™ 4.2.

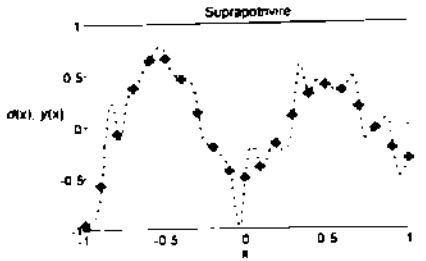


Figura 1.1.8. Suprapotrivire : funcția de aproximat (linia continuă) este aproximată foarte bine pe setul de antrenare (+), dar este aproximată slab pe setul de testare (linia întreruptă).

Uzual, setul de antrenare cuprinde 90% din datele disponibile, iar cel de testare, 10%. Utilizarea unei mulțimi separate pentru testare poate fi însă considerată o risipă dacă se dispune de puține date. Acest neajuns poate fi înlăturat, cu prețul unor calcule suplimentare, prin metoda validării de S ori (*S-fold validation*; Weiss și Kulikowski, 1991, citat de Leonard *et al.*, 1992; Ripley, 1996). Aceasta constă în împărțirea datelor disponibile în S mulțimi disjuncte, utilizarea a S-1 mulțimi pentru antrenare și a celei rămase pentru testare. Procesul se repetă de S ori, păstrându-se pentru testare de fiecare dată o altă submulțime, iar erorile obținute se însumează. Se folosesc astfel toate datele disponibile pentru antrenare, dar se efectuează de S ori mai multe calcule.

Se prezintă în continuare noțiunile esențiale legate de generalizare. Fie notațiile :

- $\mathbf{x}_i$ - vectorii de intrare pentru rețea, $\mathbf{x}_i \in \mathcal{X}$, i=1...*l*
- $y_i \in \mathcal{Y}$ i=1...*l* - ieșirile dorite corespunzătoare care, fără a restrânge generalitatea, vor fi scalare
- *l* - numărul de perechi de vectori de antrenare
- *n* - numărul de parametri, sau dimensiunea vectorului pondere al rețelei
- $f_n(\mathbf{x}_i, \mathbf{w})$ - valoarea funcției calculate de rețea pentru intrarea $\mathbf{x}_i$. Se consideră că f_n aparține unei familii $\mathcal{F}_n$, dată de arhitectura considerată în fiecare caz particular, și având ca parametri componentele vectorului pondere **w**.
- $\frac{1}{2}(y_i - f_n(\mathbf{x}_i, \mathbf{w}))^2$ - eroarea ieșirii, apreciată ca eroare pătratică.
- E[ ] - operatorul de mediere.

Selecția vectorilor de intrare $\mathbf{x}_i$ se poate considera că se face, în domeniul de definiție, într-un mod aleator. Datorită procesului de măsurare sau a însuși fenomenului (fizic, economic) ce le generează, adesea și ieșirile dorite y_i sunt zgomotoase. Tratarea va fi deci una statistică (*e.g.* White 1989; Vapnik 1995; Niyogi și Girosi, 1996).

Problema este următoarea : se dorește minimizarea a ceea ce literatura denumește funcționala riscului (Vapnik 1992, 1995, 1996) adică media funcției de eroare pe o densitate de repartiție a datelor $P(\mathbf{x},y)$ necunoscută.

$$R(f_n) = \mathrm{E}\left[\left(y - f_n(\mathbf{x}, \mathbf{w})\right)^2\right] = \int_{\mathcal{X},\mathcal{Y}} \left(y - f_n(\mathbf{x}, \mathbf{w})\right)^2 P(\mathbf{x}, y)\, d\mathbf{x}\, dy . \qquad (1.1.3)$$

Dacă se notează cu $f_0(\mathbf{x})$ funcția de regresie sau media condiționată.

$$f_0(\mathbf{x}) = \mathrm{E}\left[y \mid \mathbf{x}\right] = \int y\, P(y \mid \mathbf{x})\, dy \qquad (1.1.4)$$

funcționala riscului se poate descompune în (White, 1989; Niyogi și Girosi, 1996) :

$$R(f_n) = \mathrm{E}\left[\left(f_0(\mathbf{x}) - f_n(\mathbf{x},\mathbf{w})\right)^2\right] + \mathrm{E}\left[\left(y - f_0(\mathbf{x})\right)^2\right]. \qquad (1.1.5)$$

Primul termen al ecuației (1.1.5) arată că funcția $f_n(\mathbf{x},\mathbf{w})$ care minimizează funcționala riscului este funcția de regresie. Al doilea termen este limitarea intrinsecă dată de varianța (zgomotul) răspunsurilor dorite. Se poate observa că :

1. S-ar putea ca familia $\mathcal{F}_n$ să nu conțină funcția $f_0(\mathbf{x})$. Dacă s-ar cunoaște densitatea de probabilitate $P(\mathbf{x},y)$, s-ar putea determina parametrul optim $\mathbf{w}^*$ al familiei de funcții $f_n(\mathbf{x},\mathbf{w})$,

$$\mathbf{w}^* = arg\,\min_{\mathbf{w}} R\left(f_n(\mathbf{x},\mathbf{w})\right) \qquad (1.1.6)$$

iar distanța dintre $f_0(\mathbf{x})$ și $f_n(\mathbf{x},\mathbf{w}^*)$ ar fi

$$\mathrm{E}\left[\left(f_0(\mathbf{x}) - f_n(\mathbf{x},\mathbf{w}^*)\right)^2\right] = R\left(f_n(\mathbf{x},\mathbf{w}^*)\right) - R\left(f_0(\mathbf{x})\right). \qquad (1.1.7)$$

Aceasta este deplasarea estimatorului sau **eroarea de aproximare**. Cazul în care familia $\mathcal{F}_n$ e fixată și nu se face decât o estimare a parametrului optim este **estimarea parametrică** a funcției $f_0(\mathbf{x})$.

Există algoritmi de învățare ce modifică structura RN, precum algoritmul corelației cascadate (Fahlman și Lebiere, 1990; prezentat succint în Cimponeriu, 1994a) care, pentru reducerea erorii de aproximare, adaugă elemente suplimentare. S-a arătat în § 1.1.2 că, dacă există suficient de multe elemente ascunse, rețelele pot aproxima oricât de bine funcții continue. Aceasta înseamnă că printr-o creștere suficientă a numărului de elemente, eroarea de aproximare poate fi făcută oricât de mică. Estimarea funcției f_0 printr-o funcție cu un număr nespecificat, oricât de mare de parametri este o **estimare neparametrică**.

2. Se dorește așadar obținerea funcției $f_0(\mathbf{x})$, dată de $P(\mathbf{x},y)$, necunoscută. Se dispune însă doar de eșantioanele independente, identic distribuite $(\mathbf{x}_i, y_i)$ obținute conform $P(\mathbf{x},y)$, putându-se minimiza **riscul empiric** :

$$R_{emp,l}(f_n) = \frac{1}{l}\sum_{i=1}^{l}\left(y_i - f_n(\mathbf{x}_i,\mathbf{w})\right)^2, \qquad (1.1.8)$$

obținându-se vectorul parametrilor optimi

$$\mathbf{w}_l^* = arg\,\min_{\mathbf{w}} R_{emp,l}(\mathbf{w}). \qquad (1.1.9)$$

Eroarea

$$R_{emp,l}(\mathbf{w}_l^*) - R(\mathbf{w}) \qquad (1.1.10)$$

este **eroarea de estimare**, sau eroarea datorată varianței estimatului $\mathbf{w}_l^*$. La o mărime dată l a setului de antrenare disponibil, cu cât numărul de parametri de estimat n este mai mare, cu atât varianța este mai mare.

Cu toate că, în sens strict, eroarea de generalizare este dată doar de eroarea de estimare, în literatură se consideră că eroarea de generalizare are drept componente atât eroarea de estimare, cât și pe cea de aproximare (*e.g.* Niyogi și Girosi, 1996). Cu cât rețeaua are mai mulți parametri, cu atât eroarea de aproximare este mai mică dar, pentru un număr dat de vectori de antrenare, eroarea de estimare crește. Compromisul alegerii unui n potrivit pentru minimizarea erorii de generalizare reprezintă așa-numita dilemă deplasare-varianță (Geman *et al.* 1992).

Considerațiile prezentate au avut drept scop să evidențieze faptul că dimensiunea rețelei și numărul datelor disponibile pentru care se obțin performanțe corecte sunt strâns legate. White (1989) recomandă ca raportul dintre numărul vectorilor de antrenare și al

parametrilor rețelei să fie orientativ 10 dacă problema nu are un caracter determinist, și 100 pentru un caracter aleator mai pronunțat. Există deja estimări teoretice pentru anumite clase de funcții și de aproximanți, problema fiind de mare actualitate (Vapnik 1995, Niyogi și Girosi, 1996 și referințele lor).

Pentru obținerea unei bune generalizări se folosesc două metode. Prima este evitarea supraantrenării pe setul de antrenare prin impunerea de constrângeri suplimentare, numite regularizatoare sau stabilizatoare, care de regulă impun ca soluția să fie o funcție cât mai netedă (*e.g.* Ripley, 1996). A doua este construcția de rețele generale, și reducerea apoi a numărului de parametri prin curățare (*pruning*; *e.g.* Reed 1993, Ripley 1996). Prin aceasta se elimină ponderile ce au o influență redusă asupra funcției calculate de rețea.

Mai există o tendință, aceea de a construi rețele dedicate problemei de rezolvat, ce exploatează periodicități existente sau invarianțe necesare. Exemple sunt Neocognitronul lui Fukushima (1988) și LeNet al lui LeCun *et al.* (1989), rețele pentru recunoașterea scrisului, ca și rețeaua pentru recunoașterea vorbirii TDNN (*Time-Delay Neural Network*; Waibel *et al.* 1989). Acestea utilizează structuri localizate repetitive, pentru a obține invarianță în raport cu poziția sau momentul de apariție a trăsăturilor de interes. Neocognitronul și TDNN sunt prezentate mai detaliat în (Cimponeriu, 1994a). Un alt exemplu este rețeaua obținută de Ploix *et al.* (1994) pentru modelarea unei coloane de distilare având 102 variabile de stare, în care se exploatează faptul că cele 50 de talgere ale coloanei sunt identice. Rețeaua globală, având peste 1000 de neuroni, are o structură modulară: modulele corespunzătoare fiecărui talger fiind identice, numărul ponderilor antrenabile este foarte mic.

Dacă există cunoștințe despre soluția dorită, acestea pot fi furnizate rețelei și sub formă de vectori suplimentari de antrenare, meniți să le scoată în evidență. Este ceea ce Abu-Mostafa (1995) numește "indicații" (*hints*).

În anumite cazuri, când datele de antrenare sunt foarte puține, se pot obține rezultate mai bune prin "strâmtare" (*shrinking*), când se alege deliberat o familie de aproximanți mai săracă (în speță, o funcție de clasificare pătratică este înlocuită cu una liniară), dar pentru care parametrii pot fi estimați cu o precizie mai bună (Ripley 1996).

1.2. Rețele de aproximare locală

Deși în § 1.1. s-a făcut o prezentare generală a RN, accentul a căzut pe rețelele cu propagare înainte (*feedforward*) ce conțin neuroni cu funcție de activare liniară sau sigmoidală. Pentru acestea, aproximarea realizabilă este globală : fiecare element intervine, în mai mare sau mai mică măsură, în orice punct al domeniului de definiție al funcției de aproximat. Totodată, modificarea unei ponderi de pe primul strat, de exemplu, are ca efect schimbarea valorilor propagate în întreaga rețea. Aceasta este una din cauzele importante ale duratei mari la antrenarea BP.

Există și rețele în care aproximarea se face local, prin elemente de calcul realizând funcții a căror valoare intervine doar în zone locale ale domeniului de definiție al funcției de aproximat. Un exemplu este funcția de activare gaussiană (1.1.2). Rețelele de aproximare locală au posibilitatea de a găsi un compromis între numărul vectorilor de intrare, repartizați neuniform, și numărul de parametri ai aproximantului. Prin aceasta, performanțele lor de generalizare pot fi foarte bune (Bottou și Vapnik, 1992). Metodele locale sunt recomandate și de Wettscereck și Dietterich (1994) pentru învățarea on-line și pentru aplicațiile unde regiuni diferite ale spațiului de intrare sunt acoperite de vectori ce rezolvă subprobleme diferite.

Se prezintă în continuare, pe scurt, principalele tipuri de rețele de aproximare locală.

1.2.1. Rețele competitive

Este vorba de o familie de rețele destinate în principal clasificării, ce au ca element comun desemnarea unui element câștigător. Ponderile $\mathbf{w}_i$ asociate fiecărui element formează vectori ce reprezintă anumite valori (prototipuri) ale vectorului de intrare $\mathbf{x}$. La prezentarea unui $\mathbf{x}$ are loc o competiție pentru a se determina vectorul $\mathbf{w}_i$ cel mai apropiat de $\mathbf{x}$, conform unei funcții de distanță. Câștigătorului și eventual elementelor din vecinătatea sa i se adaptează ponderile în sensul apropierii de $\mathbf{x}$, iar ponderile celorlalte elemente fie rămân neschimbate, fie sunt îndepărtate de $\mathbf{x}$. Acest tip de rețele a apărut și s-a dezvoltat în jurul încercării de a se modela percepția vizuală de către Grossberg, von der Malsburg și Kohonen (Hecht-Nielsen, 1990), și al unor algoritmi de recunoaștere nesupervizată a formelor, precum algoritmul *k*-mediilor și al celor mai apropiați K vecini (*e.g.* Lippmann,1989).

1. Rețeaua competitivă simplă și cuantizarea vectorială

Se urmărește împărțirea pe categorii a vectorilor sau gruparea (*clustering*) lor, prin tehnica câștigătorul-ia-totul (*winner-takes-all*). Intrările rețelei sunt binare, dar funcționarea ei este foarte apropiată de cuantizarea vectorială (Hertz *et al.*, 1991; Gersho și Gray, 1992). În cuantizarea vectorială fiecărui vector aplicat la intrare i se asociază eticheta unui vector prototip, cel mai apropiat de vectorul de intrare, dintr-o mulțime de prototipuri disponibile. Prin transmiterea sau memorarea doar a indicelui prototipului (și eventual a tabelei prototipurilor, dacă acestea nu sunt fixe), se poate obține o compresie de date semnificativă.

Rețeaua competitivă simplă (Hertz *et al.*, 1991) constă dintr-un strat de intrare și unul de ieșire. Pentru fiecare vector de intrare $\mathbf{x}$ se calculează distanțele la prototipurile existente $\mathbf{w}_i$ și este desemnat câștigător prototipul $\mathbf{w}_{i^*}$ cel mai apropiat de $\mathbf{x}$:

$$\|\mathbf{w}_{i^*} - \mathbf{x}\| \le \|\mathbf{w}_i - \mathbf{x}\|, \ \forall i, \tag{1.2.1}$$

De obicei se utilizează norma euclidiană, și se obține o partiționare (pavare, *tesselation*) Dirichlet a spațiului de intrare, frontierele claselor corespunzând prototipurilor fiind suprafețe liniare pe porțiuni, (hiper)plane mediatoare ale dreptelor unind prototipuri vecine. Vectorii prototip sunt denumiți și noduri.

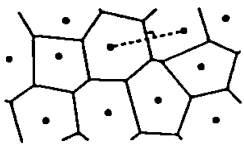


Figura 1.2.1. Pavarea Dirichlet sau poligoanele Voronoi
date de vectorii prototip (•) dintr-o regiune bidimensională.

În urma competiției este activată doar ieșirea corespunzătoare vectorului pondere $\mathbf{w}_{i^*}$.

$$y_i = \begin{cases} 1 \text{ pentru } i = i^* \\ 0 \text{ pentru } i \neq i^* \end{cases} \tag{1.2.2}$$

Adaptarea rețelei se face modificând vectorul pondere doar pentru câștigătorul i^*, pentru a-l apropia de $\mathbf{x}$:

$$\begin{aligned} \Delta \mathbf{w}_{i^*} &= \eta(\mathbf{x} - \mathbf{w}_{i^*}) \\ \Delta \mathbf{w}_i &= 0, \text{ pentru } i \neq i^* \end{aligned} \tag{1.2.3}$$

Cu ieșirile (1.2.2), (1.2.3) se poate scrie și ca (1.2.4) :

$$\Delta w_{ij} = \eta\, y_i (x_j - w_{ij}). \tag{1.2.4}$$

Se poate considera că ponderea elementului câștigător urmărește prin filtrare trece-jos de constantă de timp $1/\eta$ vectorii din raza sa de acțiune. În urma acestei adaptări, prototipurile se distribuie în zonele în care sunt grupați vectorii de intrare, ca în figura 2. Algoritmul este foarte simplu, dar, în privința minimizării erorii medii pătratice dintre $\mathbf{w}_i$ și vectorii $\mathbf{x}$ din clasa respectivă, rezultatele sunt mai slabe decât ale altor algoritmi de grupare (Ripley, 1996).

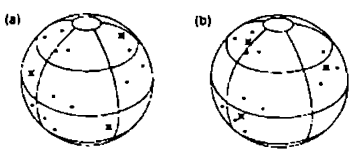


Figura 1.2.2. Efectul învățării competitive simple.
Vectorii de intrare (•) și prototipurile (x), înainte (a) și după (b) învățare

O problemă a rețelelor competitive o constituie elementele moarte (*dead units*), prea depărtate de vreun vector de intrare pentru a câștiga vreodată (Hertz *et al.*, 1991). Există însă mai multe moduri în care ea poate fi evitată, prezentate în (Cimponeriu, 1994a).

Convergență. Selectarea câștigătorului prin funcția maxim face ca o eventuală funcție de energie sau eroare asociată rețelei, care să fie minimizată prin antrenare, să fie

diferențiabilă doar pe porțiuni, dificil de mânuit și cu minime locale. În cazul actualizării incrementale, dacă modificarea se face după fiecare vector de intrare, pot apărea modificări continue în clasificare, adică rețeaua nu converge (Hertz *et al.*, 1991). Stabilitatea și convergența pot fi demonstrate doar în cazul vectorilor suficient de împrăștiați. În practică, atât pentru demonstrarea teoremelor prin metode de aproximare stohastică (*e.g.* Duflo, 1990), cât și pentru clasificarea datelor reale, parametrul de învățare η se descreşte pe parcursul antrenării : la început o valoare mare (0.8) favorizează deplasarea pe domenii întinse, la sfârșit scăderea valorii (până la 0.1 sau mai puțin) duce la o ajustare fină și la stabilizare.

Adaptarea parametrilor rețelei se face fără a fi impuse ieșirile dorite : învățarea este nesupervizată.

O variantă supervizată este cuantizarea vectorială cu învățare (*Learning Vector Quantization*, LVQ) introdusă de Kohonen și colaboratorii săi (1988; 1992), când există un set de eșantioane etichetate, câteva pentru fiecare clasă. Desemnarea câștigătorului este tot (1.2.1), dar modificarea ponderilor se face conform (1.2.5),

$$\Delta \mathbf{w}_{i^*} = \begin{cases} +\eta(t)\,(\mathbf{x} - \mathbf{w}_{i^*}) \text{ dacă clasa e corectă} \\ -\eta(t)\,(\mathbf{x} - \mathbf{w}_{i^*}) \text{ dacă clasa e incorectă} \end{cases} \qquad (1.2.5)$$

$$\Delta \mathbf{w}_i = 0 \text{ pentru } i \neq i^*$$

unde $\eta(t)$ se ia inițial mai mic decât 0,1 (tipic 0.03), și este redus apoi liniar pe durata numărului ales de iterații.

În aceeași bibliografie sunt date și variante îmbunătățite sau optimizate ale procedurii (1.2.5). Algoritmii propuși au un pronunțat iz empiric, dar rezultatele obținute sunt uneori foarte bune.

2. Stratul Kohonen

Des întâlnită în literatură este rețeaua sau harta autoorganizantă (SOM, *Self-Organizing Map*) obținută de Kohonen (*e.g.* 1989) pornind de la modelul organizării neocortexului. Ea este alcătuită dintr-un strat de elemente cu învățare competitivă. Ele sunt dispuse conform unei matrici bidimensionale, fiecare element fiind conectat cu cei patru vecini ai săi. Față de învățarea competitivă standard, după desemnarea câștigătorului (1.2.1) - (1.2.2), se modifică ponderile pentru mai multe elemente, situate în vecinătatea câștigătorului, conform (1.2.6) :

$$\Delta \mathbf{w}_i = \eta(t)\,\lambda(i,i^*)\,(\mathbf{x} - \mathbf{w}_i) \qquad (1.2.6)$$

Funcția de vecinătate $\lambda(i,i^*)$ este 1 pentru $i=i^*$, și scade cu creșterea distanței $d(\mathbf{w}_i, \mathbf{w}_{i^*})$. Adesea și ea este micșorată pe parcursul iterațiilor t. Tipic se utilizează

$$\lambda(i,i^*) = exp\left(-\frac{d(\mathbf{w}_i,\mathbf{w}_{i^*})}{2\sigma^2}\right). \qquad (1.2.7)$$

cu $d(\mathbf{w}_i, \mathbf{w}_{i^*})$ dat de suma diferențelor indicilor lui $\mathbf{w}_i$ și $\mathbf{w}_{i^*}$ în matricea bidimensională, σ ales astfel încât să fie acoperită o fracțiune semnificativă a întregii hărți, și

$$\eta \sim t^{-\alpha}, \quad 0 < \alpha \leq 1. \qquad (1.2.8)$$

Se obține o transformare topologică ce transformă relația de vecinătate din spațiul de intrare al vectorilor $\mathbf{x}$, dată de norma $\|\cdot\|$ (1.2.1) în vecinătate spațială în matricea bidimensională a neuronilor, dată de distanța $d(\,)$ din (1.2.7). Demonstrarea matematică a afirmației există pentru cazul unidimensional (Kohonen, 1989), dar nu și pentru altele mai generale (Hecht-

Nielsen, 1990). Un exemplu de astfel de transformare este harta tonotopică, în care valoarea frecvenței semnalului de intrare se localizează în unul din elementele unei matrici unidimensionale, astfel încât elementele vecine corespund unor frecvențe apropiate (Kohonen, 1989). Alt exemplu este harta bidimensională ce învață prin autoorganizare transformarea cinematică $(\theta,\phi)\rightarrow(x,y)$ a unui mecanism, braț de robot cu două articulații (Hecht-Nielsen, 1990).

3. Teoria rezonanței adaptive

Învățarea competitivă simplă nu garantează stabilitatea categoriilor formate care, chiar pentru un set finit de vectori de antrenare, se pot modifica nesfârșit. Fenomenul se poate înlătura prin reducerea treptată spre zero a parametrului învățării η: rețeaua îngheață, nu mai oscilează, dar își pierde plasticitatea, adică abilitatea de a reacționa la date noi. Nu este ușor a le avea pe amândouă: aceasta este dilema stabilitate - plasticitate a lui Grossberg.

O altă problemă este numărul de elemente de ieșire disponibile: dacă acesta e fixat, la un moment dat nu vor mai exista elemente disponibile pentru noi vectori de intrare. Dacă însă există o sursă cu capacitate suficient de mare de elemente de ieșire, trebuie evitată și folosirea a prea multe elemente, întrucât se pierde capacitatea de clasificare și proprietățile de generalizare se înrăutățesc.

Carpenter și Grossberg (1987a, 1987b, citate de Hertz *et al.*, 1991; Ripley, 1996) au dezvoltat rețelele numite ART1 și ART2, care furnizează elemente noi de calcul doar când acestea sunt necesare. Construcția rețelelor ART se bazează pe teoria rezonanței adaptive (*Adaptive Resonance Theory*): când intrarea și prototipul memorat sunt suficient de similare, ele intră în rezonanță; dacă un vector de intrare nu rezonează cu nici unul din prototipurile existente, este formată o nouă categorie. Dacă s-au epuizat elementele disponibile, stimulul nu primește nici un răspuns. Înțelesul lui "suficient de similare" depinde de un parametru de vigilență sau de precizie ρ, cu $0<\rho\leq1$: dacă ρ este mare, condiția de similaritate este foarte selectivă și se obțin multe categorii divizate fin; un ρ mic dă o împărțire grosieră. ρ poate fi crescut pe parcursul antrenării. ART1 este proiectată pentru intrări și ponderi binare, iar ART2 pentru intrări continue. Modelele lor detaliate sunt complexe; o descriere simplificată a rețelei ART1 este prezentată în (Cimponeriu, 1994a). Pentru scopul acestei lucrări, este de reținut doar ideea posibilității adăugării de noi elemente, până la un număr maxim, în caz că clasificarea efectuată nu este suficient de bună.

1.2.2. Rețeaua cu funcțiile bazei radiale

Rețeaua cu funcțiile bazei radiale (*Radial Basis Functions*, RBF), dezvoltată în mai multe lucrări, cea mai cunoscută fiind (Moody și Darken, 1989), și prezentată în toate articolele generale sau monografiile suficient de recente (Hush și Horne, 1993; Haykin, 1994; Ripley, 1996) este cea din figura 1.2.3. Ea are două straturi, elementele de ieșire formând o combinație liniară (1.2.10) de funcții gaussiene (1.2.9), realizate de elementele stratului ascuns. Aceste rețele mai sunt denumite rețele cu câmpuri receptive localizate. Neuroni

biologici cu această proprietate sunt întâlniți în multe părți ale sistemului nervos : celulele receptoare ale cochleei prezintă un răspuns localizat în frecvență: celulele cortexului somato-senzorial răspund selectiv la stimuli provenind din regiuni localizate ale corpului; în cortexul vizual există celule cu selectivitate la stimuli localizați atât ca poziție pe retină cât și ca orientare a obiectului (Moody și Darken, 1989). O hartă cu răspunsuri localizate, funcție de frecvență și de abaterea Doppler a frecvenței se găsește și pe creierul liliacului (Suga, 1990). Pe de altă parte, metode locale pentru estimarea neparametrică a densității de probabilitate, precum ferestrele Parzen, sau utilizarea funcțiilor discriminant locale numite "de potențial", utilizate în recunoașterea formelor (Duda și Hart, 1973) sunt foarte asemănătoare, dacă nu identice, cu funcțiile RBF.

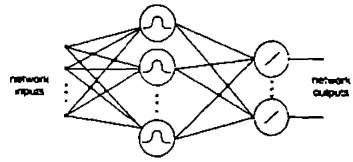


Figura 1.2.3. Rețeaua RBF.

$$u_j = exp\left(-\frac{(\mathbf{x} - \mathbf{w}_j)^T (\mathbf{x} - \mathbf{w}_j)}{2\sigma_j^2} \right) \quad j = 1 \ldots H \tag{1.2.9}$$

$$y_i = \mathbf{v}_i^T \mathbf{u} = \sum_{j=1}^{H} v_{ij} u_j \quad i = 1 \ldots M \tag{1.2.10}$$

unde

$\mathbf{x}$ este vectorul de intrare,
$\mathbf{w}_j$-vectorul ponderi al elementului ascuns j,
σ_j-lățimea ferestrei gaussiene j,
u-vectorul activităților stratului ascuns,
$\mathbf{v}_i$-vectorul ponderi al ieșirii i,
H, M-dimensiunile stratului ascuns și respectiv de ieșire.

Funcțiile (1.2.9) au o simetrie radială : răspunsul obținut nu depinde de direcția lui $\mathbf{x}$, ci doar de distanța sa față de $\mathbf{w}_j$ în raport cu σ_j. Există și variante ale funcției (1.2.9). Uneori acestea se normalizează, pentru ca suma lor să fie unitară (Moody și Darken, 1989). Alteori, în calculul distanței dintre $\mathbf{x}$ și $\mathbf{w}_j$ se renunță la simetria radială, preferându-se hiperelipsoizi ce țin seama de împrăștierea anizotropă a datelor (Lee și Kil, 1991).

În locul funcției gaussiene din (1) pot fi folosite aproximări apropiate ca formă, dar mai ușor de calculat, precum cea a lui Platt (1991) :

$$u_j = \begin{cases} \left(1 - \dfrac{z_i}{q\sigma_j^2}\right)^2 & \text{dacă } z_i < q\sigma_j^2, \\ 0 & \text{altfel} \end{cases} \tag{1.2.11}$$

unde $z_i = (\mathbf{x} - \mathbf{w}_j)^T (\mathbf{x} - \mathbf{w}_j)$,

iar q=2.67 este ales empiric pentru cea mai bună potrivire cu funcția gaussiană. Ripley (1996) prezintă și alte funcții de activare.

Aproximarea funcțiilor

S-a demonstrat (*e.g.* Hartman *et al.* 1990, citat de Hush și Horne, 1993) că rețeaua RBF poate aproxima oricât de bine orice funcție continuă. În fapt, o parte a demonstrațiilor proprietăților de aproximare și de clasificare universală a rețelelor feedforward cu activare sigmoidală (v. § 1.1.2) se bazează pe construirea unor volume de percepție localizată, obținute cu prețul unui strat suplimentar, aici realizate direct. Aproximarea uniformă pe mulțimi compacte a funcțiilor continue prin funcții RBF gaussiene poate fi demonstrată și cu ajutorul teoremei Weierstass-Stone (Girosi și Poggio, 1990, citat de Ripley, 1996).

Învățarea

Dacă centrele gaussienelor se iau a fi chiar vectorii de antrenare, se obține estimarea neparametrică a ferestrelor Parzen de nucleu gaussian (Specht, 1990). Aceasta duce însă la o rețea nepractic de mare. De obicei se preferă un număr mai mic de vectori care să reprezinte centrele funcțiilor gaussiene. Moody și Darken (1989) au obținut centrele $\mathbf{w}_i$ și lățimile σ_i în două moduri, fie prin adaptare prin învățare supervizată, fie prin metode de grupare din recunoașterea formelor (*e.g.* Duda și Hart, 1973; Ripley, 1996). Antrenarea supervizată dă rezultate bune ca precizie, dar nu impune restricții arhitecturale rețelei. În particular, lățimile pot lua valori mari și se pierde proprietatea de reprezentare locală. De asemenea, învățarea elementelor gaussiene nu este apreciabil mai rapidă decât a rețelei BP. Gruparea vectorilor de intrare este mai avantajoasă : ea garantează obținerea unor reprezentări într-adevăr locale și, conținând doar optimizări liniare, este mult mai rapidă. Centrele celor k gaussiene presupuse a fi necesare sunt determinate prin algoritmul k-mediilor, ce găsește k vectori - valori medii empirice locale peste distribuția vectorilor de intrare. Lățimile σ_j se determină euristic, astfel încât ariile de percepție să se suprapună parțial, de exemplu luând valoarea medie a distanței dintre centrul $\mathbf{w}_j$ și vectorii $\mathbf{x}$ care contribuie la obținerea lui :

$$\sigma_i^2 = \frac{1}{M_j}\sum_{i=1}^{M_j}(\mathbf{x}-\mathbf{w}_i)^T(\mathbf{x}-\mathbf{w}_i) \ . \qquad (1.2.12)$$

Ponderile de pe ultimul strat, liniar, sunt determinate fără dificultăți, prin metode de optimizare liniară. Uneori se adaugă și o pondere de deplasare (*bias*), care dă valoarea de ieșire când nici un element gaussian nu e activ.

Rețelele RBF sunt adesea mult superioare rețelelor BP, atât ca eficiență de calcul, cât și ca precizie. Pe lângă variantele sugerate de Moody și Darken, există și mai multe îmbunătățiri, mai ales în sensul construcției adaptive a stratului ascuns.

Lee și Kil (1991) cresc treptat numărul elementelor gaussiene, micșorându-le simultan raza, până la obținerea preciziei impuse. Weymaere și Martens (1991) le cresc și ei numărul, până la un maxim predefinit.

Platt (1991) adaugă treptat elemente, dar într-un mod supervizat. Pentru un vector nou de intrare $\mathbf{x}$, dacă eroarea rețelei depășește un prag ε și dacă $\mathbf{x}$ este suficient de depărtat de ceilalți $\mathbf{w}_j$ memorați,

$$\|\mathbf{x}-\mathbf{w}_j\| > \delta(t), \qquad (1.2.13)$$

el alocă un element nou cu $\mathbf{w} = \mathbf{x}$ și $\sigma = \kappa \left\| \mathbf{x} - \mathbf{w}_{\text{cel mai apropiat}} \right\|$. Distanța $\delta(t)$ este o scală de rezoluție ce variază cu iterația t conform expresiei

$$\delta(t) = max\left\{ \delta_{max}\, exp\left(-\frac{t}{\tau}\right),\ \delta_{min} \right\}, \tag{1.2.14}$$

unde τ este o constantă de timp. La început rețeaua crează o reprezentare grosieră a funcției de învățat, care apoi se rafinează, dar fără ca nodurile să fie mai apropiate decât δ_{min}. Eventual se pot adapta toți parametrii rețelei prin algoritmul LMS (v. § 3.1), obținându-se aceeași precizie cu rețele mai mici. Pentru predicția unei serii de timp, valorile parametrilor au fost : δ_{max} = 0.7, δ_{min} = 0.07 atins după 100 de iterații, κ = 0,87, ε = 0,02 (optimizare pentru eroare) sau 0.05 (optimizare pentru dimensiunea rețelei). Față de rețeaua RBF standard, pentru o eroare similară, creșterea numărului de elemente cu numărul de vectori dein setul de antrenare este mult mai puțin pronunțată. Numărul de parametri este aproximativ același cu al unei rețele BP cu aceeași eroare, dar timpul de calcul este mult mai mic.

Fritzke (1994 a, b) adaugă elemente gaussiene nu pe baza erorii unui sigur eșantion, ci a erorii acumulate pe mai multe date, având grijă să păstreze relațiile de vecinătate cu elementele deja existente. Rețeaua lui învață într-un număr de iterații cu două ordine de mărime mai mic decât rețeaua BP, și pare să aibă și performanțe mai bune de generalizare.

Pentru a nu crește prea mult dimensiunea rețelei, dacă s-a ajuns la un număr prestabilit de elemente gaussiene, Molina și Narajan (1996) realocă un element vechi dar puțin eficace.

Se poate menționa faptul că rețeaua RBF poate fi obținută impunând unei structuri generale de aproximare, aceea a rețelei cu un strat ascuns având funcții de activare nespecificate și elemente de ieșire liniare (modele aditive, Ripley, 1996) criterii de regularizare (netezire), precum un conținut cât mai redus de putere la frecvențe înalte al aproximantului (Poggio și Girosi, 1990a, Girosi *et al.*, 1995). Însă faptul de a fi un aproximator cât mai neted este tocmai ceea ce Wong (1991) reproșează rețelelor RBF. El își explică performantele bune ce le sunt cvasiunanim atribuite, printr-o frecvență prea joasă de eșantionare a datelor.

Cu câteva restricții, rețelele RBF sunt echivalente funcțional cu sistemele de inferență *fuzzy* (Roger Jang și Sun, 1993; Hunt *et al.*, 1996).

1.2.3. Rețele de aproximare locală constantă și liniară

Aproximarea cea mai simplă a funcțiilor se poate face prin funcții local constante, sau prin funcții liniare pe porțiuni, corespunzând aproximării Taylor de ordinul 0, respectiv 1:

$$f(\mathbf{x}) \cong f(\mathbf{w}_i) \text{ pentru } \mathbf{x} \cong \mathbf{w}_i. \tag{1.2.15}$$

$$f(\mathbf{x}) \cong f(\mathbf{w}_i) - J(\mathbf{w}_i)\,(\mathbf{w}_i - \mathbf{x}) \text{ pentru } \mathbf{x} \cong \mathbf{w}_i. \tag{1.2.16}$$

unde

$$J = \left[J_{jk}\right] = \left[\frac{\partial f_j}{\partial x_k}\right] \tag{1.2.17}$$

este matricea jacobiană a funcției vectoriale de variabilă vectorială $f(\mathbf{x})$. Așa cum s-a arătat în § 1.2. Blum și Li (1991) demonstrează capacitatea RN de a fi aproximatori universali prin

funcții simple, constante pe porțiuni. Rețeaua lor are două straturi ascunse, primul servind la obținerea domeniilor corespunzătoare constantelor, date de stratul de ieșire.

Cel mai des folosită pentru aproximări locale constante sau liniare este rețeaua Kohonen. Cherkassky și Najafi (1991) învață prin algoritmul Kohonen valorile discrete simultan pentru intrări și pentru ieșiri, prin ceea ce ei denumesc "asociere topologică restrânsă" (*constrained topological mapping*). Aproximarea constantă pe porțiuni este dată de valorile discrete corespunzătoare ieșirilor. Dacă se dorește o aproximare liniară, ea se obține prin interpolare liniară între valorile corespunzătoare nodurilor ce definesc poligonul de număr minim de laturi din spațiul de intrare, ce conține intrarea curentă.

Constatând că distribuirea nesupervizată a nodurilor din spațiul de intrare este ineficientă, Najafi și Cherkassky (1994) propun ca plasarea acestora să se facă și în funcție de estimata derivatei a doua a funcției de aproximat. Însă ei nici nu menționează modul de estimare al derivatei a doua, nici nu iau în considerare cazul în care funcția de aproximat este vectorială. Această situație poate fi descompusă în aproximarea independentă a mai multor funcții scalare, sau tratată unitar. Prima soluție este mai simplă, dar ridică problema nodurilor diferite ale intrărilor pentru fiecare din funcțiile scalare de ieșire. A doua evită această problemă și păstrează avantajul cuantizării vectoriale pentru ieșiri, dar ridică alte probleme, precum evaluarea celei de-a doua derivate a funcției de aproximat, care este matricea hessiană.

Un caz particular de aproximare prin funcții constante pe porțiuni este **rețeaua CMAC**, ce modelează obținerea comenzilor musculare în cerebel (*Cerebellar Model Articulation Controller*, Albus 1975). Vectorul de intrare adresează o zonă de memorie cuprinzând l locații unde sunt stocate valori a căror sumă dă valoarea ieșirii. Un vector de intrare vecin va adresa tot l locații de memorie, din care cel mult m coincid cu cele ale primului, dând un răspuns apropiat, iar restul diferă, furnizând diferența rămasă. Dacă funcția de învățat este netedă, ceea ce se învață pentru un vector este valabil și pentru vectorii apropiați, obținându-se o adaptare rapidă. Cu cât doi vectori de intrare sunt mai depărtați, cu atât numărul locațiilor de memorie comune este mai mic. Se poate considera că pentru fiecare componentă a vectorului de intrare se dispune de m grile de eșantionare de pas p, decalate cu $d = p/m$, iar răspunsul este suma valorilor memorate pe fiecare din grile. Pentru vectori multidimensionali poate fi necesară o memorie prea mare. Pentru a se obține o memorie de dimensiune rezonabilă, se aplică tehnica *hash-coding*, utilizată pentru memorarea matricilor rare, prin care adrese de dimensiuni mari sunt redirectate spre zone mai mici de memorie. Redirectarea se face prin calcul, sau pe baza unui generator de numere aleatoare. Aceasta face ca să poată apărea coliziuni, când două adrese inițial diferite ajung la aceeași locație de memorie. Efectul este o eroare care poate fi micșorată prin creșterea suficientă a memoriei alocate. În final, pentru fiecare $\mathbf{x}$ sunt adresate mai multe ponderi w_i, iar ieșirea rețelei este suma lor :

$$y(\mathbf{x}) = \sum_{i}^{l} w_i \,. \tag{1.2.18}$$

Ponderile adresate se adaptează conform relației :

$$\Delta w_i = \frac{1}{l}\left(f(\mathbf{x}) - y(\mathbf{x})\right). \tag{1.2.19}$$

Wong și Sideris (1992) arată că algoritmul de învățare este convergent, iar valorile prezentate pot fi învățate cu o precizie oricât de bună dacă doi vectori de intrare nu au asociate exact aceleași locații de memorie. Cotter și Guillerm (1992) și Brown *et al.* (1993) demonstrează că rețeaua CMAC nu poate învăța funcții care au oscilații locale prea mari. Practic, rețeaua CMAC este folosită pentru învățarea rapidă a unor traiectorii repetitive (*e.g.* Miller, 1989, 1994; Lin și Song, 1993), când interesează învățarea unei funcții continue, pentru o mulțime restrânsă de puncte, apropiate.

Funcțiile prin care rețeaua CMAC aproximează funcția dorită sunt constante pe porțiuni, iar trecerea de la un domeniu de aproximare la altul se face prin salturi. Aceasta corespunde asocierii domeniilor de aproximare sau câmpurilor receptive a unei funcții dreptunghiulare, binare 0/1, ce înmulțește valorile asociate. O extindere a rețelei CMAC (Lane *et al.*, 1992) se obține dacă aceste funcții se iau a fi continue, liniare pe porțiuni, sau funcții *spline* liniare. O aproximare netedă, cu care se poate obține și continuitatea derivatelor 1 și 2, se poate obține prin funcții *spline* cubice (polinoame cubice pe porțiuni; Micula, 1978).

Moody (1989) a realizat o arhitectură multirezoluție de rețele CMAC, în care grilele de eșantionare au pași diferiți. Prima care este antrenată este rețeaua care dă aproximația cea mai grosieră. După aceea este antrenată rețeaua care dă detalii mai fine, apoi cea care dă detalii și mai fine, ș.a.m.d. Rețeaua obținută, utilizând funcții *spline*, are, pentru o aplicație de predicție a unei serii de timp, o durată de antrenare mai mică cu două ordine de mărime decât o rețea *backpropagation*, fiind potrivită pentru aplicații în timp real. Fără a o plasa în contextul rețelelor neuronale, învățarea transformărilor inverse prin funcții *spline* o studiază și Heiss (1994).

Dar extensia naturală a aproximării prin funcții constante pe porțiuni este aproximarea liniară pe porțiuni. Pellionisz și Llinas (1979, citat de Atkeson, 1989) par a fi primii care învață iterativ o transformare prin **modele local liniare** (LLM, *Local Linear Mappings*), memorate într-un tabel sub forma vectorilor proprii și valorilor proprii ai matricii jacobiene. Ei aplică această metodă și transformării cinematice a unui braț de robot (Pellionisz și Llinas, 1980, citat de Atkeson, 1989).

Ritter, Martinetz și Schulten (1989, 1992) învață și ei aproximări liniare pe porțiuni ale transformării cinematice și dinamice a unui braț manipulator. Rețeaua lor este o extindere a rețelei Kohonen : spațiul de intrare al vectorilor $\mathbf{x}$ este discretizat prin vectorii $\mathbf{w}_i$ prin algoritmul lui Kohonen. Ei învață transformarea compusă dintre o transformare vizuală efectuată de două camere ce observă un braț de robot, și transformarea cinematică a brațului (v. § 2.2) : fiecărui element câștigător 4-dimensional $\mathbf{w}_i$ al stratului Kohonen, ei îi asociază un vector $\mathbf{u}_i$ având 15 componente : 3 pentru $f(\mathbf{w}_i)$ și 12 pentru $J(\mathbf{w}_i)$, matrice 3x4. Vectorii $f(\mathbf{w}_i)$ și matricile $J(\mathbf{w}_i)$ sunt învățate prin algoritmul LMS, necesitând câteva zeci de mii de iterații.

O variantă a rețelei Kohonen extinse (cu modele local liniare) este "gazul neuronal" (*neural gas*) al lui Martinetz *et al.* (1993). Aceștia modifică modul de adaptare a ponderilor pe durata autoorganizării, astfel încât să fie minimizată eroarea medie de distorsiune (Zador, 1982, citat de Martinetz *et al.*, 1993). Această rețea, ce utilizează de asemenea LLM, dă rezultate mai bune, pentru predicția seriilor de timp, decât rețelele BP și RBF (Martinetz *et al.*, 1993). Hesselroth *et al.* (1994) utilizează același tip de rețea pentru comanda unui braț pneumatic a cărei dependență a poziției cu presiunea este neliniară și cu histerezis, obținând o precizie de poziționare de un pixel.

Kuhn *et al.* (1995) utilizează o rețea cu aproximări LLM pentru a comanda un braț cu reacție vizuală. Rețeaua lor, asemănătoare rețelei lui Ritter *et al.* (1992), conține 700 de elemente care sunt întâi autoorganizate prin algoritmul Kohonen, și apoi adaptate prin algoritmul LMS, convergența fiind obținută după exersarea a circa 200 de traiectorii aleatoare în întreg spațiul de lucru. Același tip de rețea este utilizat în (Gresser *et al.*, 1996).

Fritzke (1995) construiește o rețea LLM în mod supervizat, inserând elemente acolo unde eroarea ieșirii, acumulată pentru câteva eșantioane, este mare. Noul element LLM, $\mathbf{w}_{nou}$ este inserat între vectorul de intrare $\mathbf{x}$ și cel mai apropiat nod deja existent, $\mathbf{w}_{j^*}$, fiind caracterizat prin relațiile (1.2.18).

$$\begin{aligned} \mathbf{w}_{nou} &= \tfrac{1}{2}\left(\mathbf{x} + \mathbf{w}_{j^*}\right) \\ f\left(\mathbf{w}_{nou}\right) &= \tfrac{1}{2}\left(f(\mathbf{x}) + f\left(\mathbf{w}_{j^*}\right)\right) \\ \boldsymbol{J}\left(\mathbf{w}_{nou}\right) &= \tfrac{1}{2}\left(\boldsymbol{J}(\mathbf{x}) + \boldsymbol{J}\left(\mathbf{w}_{j^*}\right)\right) \end{aligned} \qquad (1.2.20)$$

Fritzke nu specifică modul în care se învață matricile $\boldsymbol{J}$; se poate presupune implicit că este vorba de algoritmul LMS. Pentru o problemă de clasificare în două clase, el obține convergența în 60 de cicluri de antrenare (cuprinzând fiecare 528 de vectori). Pentru aceeași problemă, o rețea BP necesită 3000 de cicluri.

Rețeaua LLM cu arhitectură în cascadă a lui Littmann și Ritter (1992; 1996) poate fi considerată că combină aproximarea local liniară cu abordarea multirezoluție. Rețeaua poate aloca noi elemente, asemănător rețelei corelației cascadate a lui Fahlman și Lebiere (§ 1.1.4): dacă eroarea medie pătratică dintre ieșirea calculată și cea dorită nu este suficient de mică, se adaugă un nou strat de elemente LLM ce au ca intrări vectorul de intrare și ieșirile straturilor adăugate precedent. Performanțele obținute cu această structură sunt bune, atât pentru o aplicație de clasificare, cât și pentru una de aproximare - predicția unei serii de timp. O concluzie interesantă este că pentru același număr de elemente LLM, generalizarea este mai bună când elementele sunt în cascadă și nu într-un singur strat. De asemenea, ei dau ideea de a utiliza, pentru straturile succesive, intrări suplimentare, de context.

O altă clasă de aproximanți este cea în care răspunsul este obținut nu ca urmare a competiției între elemente, ci a cooperării. În aceste rețele ieșirea este dată de suma mai multor elemente cu aproximare local liniară, eventual ponderată în funcție de distanțele dintre vectorul de intrare și nodurile corespunzătoare. Acestea sunt folosite mai ales pentru aproximarea funcției de densitate de probabilitate condiționată de clasă, în aplicații de clasificare, fiind numite "amestecuri de experți" (Jacobs și Jordan, 1993; Jordan și Jacobs, 1994; Alpaydin și Jordan, 1996; Ripley, 1996). Se observă că rețeaua CMAC este combinație competiție-cooperare. Pentru aplicații de regresie, Schaal și Atkeson (1995) ponderează "experți locali" liniari, care sunt de fapt aproximări liniare valabile pe întreg domeniul funcției, prin funcții gaussiene, care conferă caracterul de localizare.

Extensia naturală a modelării local liniare este modelarea local pătratică. Cu ajutorul ei, și utilizând în plus regresia ponderată local, în care sunt luate în considerare mai mult datele apropiate decât cele depărtate de poziția curentă, Atkeson (1991) obține, în învățarea dinamicii unui braț de robot cu două grade de libertate, rezultate superioare ca precizie rețelelor CMAC și *feedforward* sigmoidale. Totuși, prin preocupările sale recente (Schaal și Atkeson, 1995), Atkeson pare să indice că aproximările local liniare sunt mai avantajoase celor local pătratice.

1.3. Modelare adaptivă şi comandă adaptivă cu reţele neuronale

1.3.1. Modelare adaptivă şi comandă adaptivă[1]

Cea mai întâlnită clasă de aplicaţii ale RN este modelarea adaptivă. Prin modelare adaptivă se înţelege antrenarea şi apoi utilizarea unei RN ca o "cutie neagră", în locul unui anumit sistem fizic, pe care ea îl modelează. Ca sisteme adaptive, RN au aplicaţii îndeosebi acolo unde caracteristicile mediului sau sistemelor de interes fie nu sunt cunoscute, fie sunt greu exprimabile analitic.

Există două tipuri de modelare adaptivă: directă şi inversă, ca în figura 1.3.1.

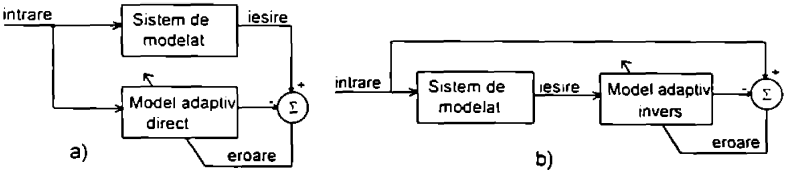


Figura 1.3.1. Modelare adaptivă a) directă, şi b) inversă

În modelarea adaptivă directă eroarea dintre ieşirea sistemului de modelat şi cea a modelului serveşte la modificarea parametrilor modelului adaptiv, conform unei legi de adaptare (învăţare) alese, astfel încât să se minimizeze o funcţie de eroare aleasă. Modelul adaptiv învaţă să aproximeze sistemul de modelat. În modelarea inversă compusa funcţiilor realizate de sistemul de modelat şi de modelul adaptiv învaţă să aproximeze unitatea, astfel încât modelul adaptiv învaţă o aproximare a inversei funcţiei sistemului modelat.

În modelarea adaptivă inversă interesează găsirea modelului invers al sistemului necunoscut, astfel încât compusa acestora să fie o identitate sau o întârziere. Un exemplu de astfel de aplicaţie este egalizarea adaptivă a canalelor telefonice. De observat că funcţia inversă este obţinută implicit, evitându-se obţinerea expresiei sale analitice. Problema devine nebanală în cazul în care inversa nu e unică, ca în cazul braţelor de robot redundante, şi poate fi rezolvată prin impunerea de constrângeri suplimentare, ce determină unicitatea soluţiei (e.g. Bailleul, 1986, citat de Lee şi Kil, 1994). Dezvoltările din capitolele următoare se încadrează în modelarea adaptivă inversă, cu soluţie unică.

Un caz particular al modelării adaptive este comanda adaptivă, în care locul regulatorului tradiţional este luat de un model adaptiv, ca în figura 1.3.2 (Widrow şi Stearns, 1985).

[1] Acest paragraf reia § 1.2 din (Cimponeriu, 1994b). O bună introducere în comanda adaptivă, ce consideră cazul modelării liniare, este (Widrow şi Stearns, 1985). O lucrare de referinţă este şi (Goodwin şi Sin, 1984). O introducere mai avansată este (Jordan, 1994).

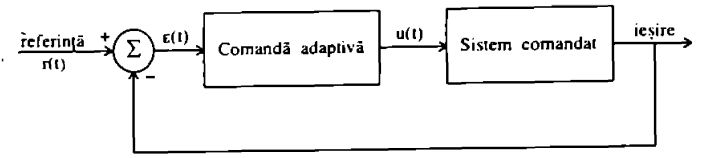


Figura 1.3.2. Comandă adaptivă

Comanda adaptivă se poate realiza fie cu modelul adaptiv direct, fie cu cel invers. Primul caz este prezentat în figura 1.3.3. Se presupune că nu este disponibil un model analitic al sistemului comandat. Modelul direct este învățat ca în figura 1.3.1 a. Parametrii modelului (neuronal) obținut servesc pentru calculul înainte al semnalului de comandă.

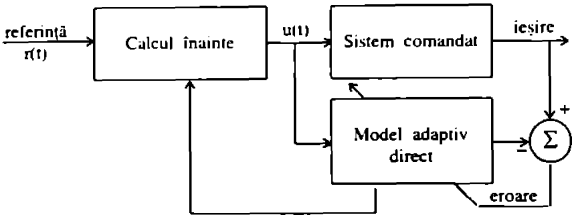


Figura 1.3.3. Comandă adaptivă cu modelul adaptiv direct

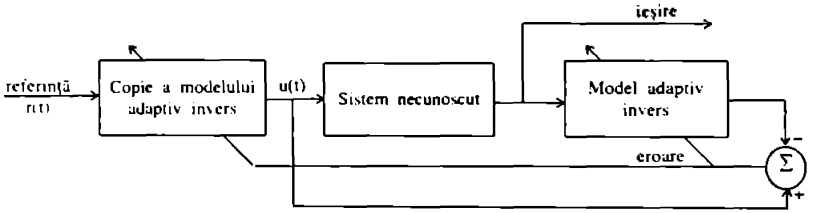


Figura 1.3.4. Comandă adaptivă cu modelul adaptiv invers

Comanda adaptivă ce utilizează modelul invers este prezentat în figura 1.3.4. Modelul invers este obținut ca în figura 1.3.1 b. Între intrare și ieșire se compune, în ordine inversă, modelul invers și sistemul modelat, obținându-se deci, aproximativ, identitatea, deci ieșirea aproximează referința. Învățarea modelului invers se face, cel mai adesea, *off-line*, ca etapă separată înaintea utilizării, de regulă pe un calculator mai puternic. Dacă învățarea modelului invers este suficient de rapidă, se poate face chiar o învățare *on-line*. Acesta este cel mai des întâlnit tip de modelare adaptivă. Avantajul său principal este evitarea necesității de a se dispune de un model analitic al funcției inverse a procesului comandat.

Un caz particular de comandă adaptivă îl constituie comanda robotică. Acesta este unul din domeniile în care rețelele neuronale au cunoscut un succes deosebit : există volume ce îi sunt dedicate exclusiv, fie ca și culegeri de articole prezentate la întâlniri de lucru (Miller *et al.*, 1990; Bekey și Golberg 1993), fie ca volume unitare (Ritter *et al.*, 1992). Proprietatea rețelelor neuronale de a învăța funcții pe baza unor eșantioane poate fi utilizată pentru

învăţarea unor modele cinematice sau dinamice, a comenzii necesare unui sistem pentru urmărirea unei anumite traiectorii[2], sau a modelelor unor sarcini particulare.

Pentru învăţarea modelelor cinematice sau dinamice pe întreg spaţiul de lucru se folosesc mai ales reţeaua Kohonen extinsă (Ritter *et al.*, 1992; Walter şi Schulten, 1993), reţelele RBF şi reţeaua BP (Bassi şi Bekey, 1989; Yeung şi Bekey, 1993). Gorinevski şi Connoly (1994) prezintă o comparaţie între reţele BP, RBF şi Kohonen extinsă, ca şi o bibliografie mai detaliată. Se folosesc şi reţele BP locale, pe care Jacobs şi Jordan (1993) le ponderează potrivit unui model probabilist. Sunt de asemenea utilizate reţelele de aproximare locală liniară, prezentate în § 1.2.3.

Pentru modelarea unor sarcini particulare, în care trebuie învăţate doar câteva traiectorii posibile, reţeaua cea mai utilizată este reţeaua CMAC (*e.g.* Miller, 1989, 1994; Lin şi Song, 1993), avantajoasă prin capacitatea ei de învăţare rapidă.

O trecere în revistă cuprinzătoare a metodelor de învăţare a modelelor cinematice şi dinamice ale braţelor de robot, precum şi chestiuni mai generale, din care multe continuă să fie actuale, este făcută de Atkeson (1989).

1.3.2. Comandă adaptivă cu reacţie vizuală

Una dintre căile de obţinere a informaţiei este cea vizuală. Relativ complexe, sistemele robotice cu reacţie vizuală prezintă avantajul perceperii mediului de lucru. Aceasta conferă o adaptabilitate deosebită, necesară, spre exemplu, manipulatoarelor ce deservesc o bandă de asamblare caracterizată de variabilitatea pieselor şi a poziţiei lor, sau deplasării într-un mediu necunoscut.

O lucrare ce conţine rezultate recente în comanda roboţilor cu reacţie vizuală este (Hashimoto, 1994). Din cuprinzătoarea trecere în revistă a lui Corke din acest volum, se prezintă, pe scurt, conceptele şi rezultatele cele mai importante din comanda vizuală a manipulatoarelor robotice.

Modul cel mai simplu de realizare a unui sistem robotic înzestrat cu un sistem vizual este prin comandă directă (*feedforward*) sau în buclă deschisă, "privire" şi apoi "deplasare", ca în figura 1.3.5. Precizia de poziţionare depinde direct de precizia captorului video şi a blocului de comandă.

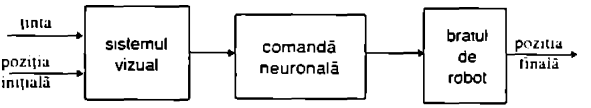


Figura 1.3.5. Poziţionare prin comandă directă

O alternativă mai avantajoasă este utilizarea unei bucle de reacţie. Cum se va arăta cantitativ în capitolul următor, aceasta permite obţinerea unei poziţionări precise şi în prezenţa erorilor în lanţul de comandă. Termenul utilizat actual în literatură pentru comanda cu reacţie vizuală este *visual servoing*, utilizat în sensul de "sistem cu reacţie cu captare vizuală". Pe

[2] Arie de aplicaţie cunoscută sub numele de comandă cu învăţare iterativă (*iterative learning control*; Moore, 1992)

lângă acesta, mai vechiul *visual feedback*, reacție vizuală, reprezintă abordarea în care se alternează între captarea imaginii și deplasare. În lucrarea de față se preferă cel de-al doilea termen, datorită mai marii apropieri de principiu cu comanda cinematică realizată, și a conciziunii.

Problema constă în a comanda poziționarea (poziția și orientarea; *pose*) a organului efector al robotului, utilizând informația vizuală sub forma unor trăsături (*features*) extrase din imagine. Pentru obținerea acestora o cameră video este, cu excepția poziționării în plan, insuficientă. De obicei se utilizează două camere, sau imagini consecutive obținute de la o singură cameră, aflată în mișcare. Comanda cu reacție vizuală poate fi bazată pe poziție (*position-based*) sau pe imagine (*image-based*), ca în figurile 1.3.6 și 1.3.7.

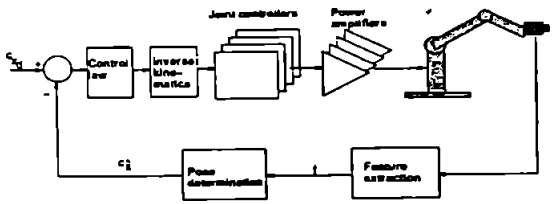


Figura 1.3.6. Buclă cu reacție vizuală având comanda bazată pe poziție

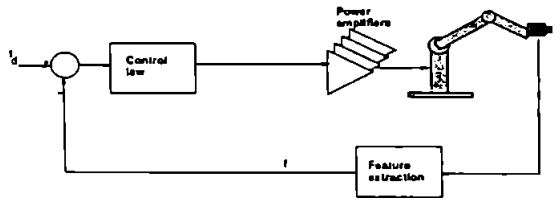


Figura 1.3.7. Buclă cu reacție vizuală având comanda bazată pe imagine

În primul caz comanda este calculată cunoscându-se poziționarea efectorului robotului în spațiul cartezian, obținută din imagini. În cea de-a doua, propusă de Weiss *et al.* (1987, citat de Hashimoto și Kimura, 1994, care prezintă și alte trimiteri bibliografice) și studiată în numeroase alte lucrări, precum (Feddema *et al.*, 1994; Bien *et al.*, 1994; Khadraoui *et al.*, 1995) se evită etapa calculării poziționării, comanda fiind dată direct pe baza trăsăturilor detectate în imagini. Este propusă inclusiv specificarea sarcinilor și planificarea în spațiul vizual (Jägersand și Nelson, 1994, 1995). Prima metodă presupune modelarea și calibrarea cât mai exactă a sistemului vizual. În ultima vreme se preferă însă evitarea calibrării sistemului vizual, făcându-se o comandă bazată pe reacție vizuală necalibrată (*uncalibrated visual servoing*). Aceasta duce la cea de-a doua metodă, care este mai avantajoasă și prin calculele mai reduse. Ea utilizează modelul liniarizat local al transformării compuse a ansamblului manipulator – sistem vizual, prin matricea jacobiană vizuo-motoare (*visual motor Jacobian*). Astfel de sisteme, numite și "cu coordonare mână-ochi" (*hand-eye coordination*), sunt realizate prin algoritmi generali de învățare, dar mai ales prin tehnici de rețele neuronale (Corke, 1994).

Comanda poate îmbina informații provenind de la o cameră video fixă și de la traductoarele de poziție ale brațului. Castaño și Hutchkinson (1994) comandă cele două grade

de libertate din planul camerei prin reacţie vizuală, iar gradul de libertate rămas, perpendicular pe cameră, prin reacţie provenind de la traductoarele articulaţiilor braţului.

De cele mai multe ori există două camere video, fixe. Dar reacţia vizuală poate fi realizată şi prin intermediul unei singure camere, mobile, plasate pe braţul de robot. Miller (1989) comandă un braţ ce deserveşte o bandă rulantă printr-o reţea CMAC cu 12 intrări, 4 fiind poziţiile unghiulare ale articulaţiilor, 4 parametri ai imaginii pe camera fixată pe braţ a obiectului urmărit, şi 4 pentru variaţiile parametrilor imagine. Tot cu o cameră fixată pe braţ, van der Smagt *et al.* (1995) reuşesc poziţionarea braţului de robot deasupra unei ţinte observate vizual. În altă lucrare, van der Smagt (1995) a comandat un braţ flexibil acţionat pneumatic, realizat după modelul muşchilor scheletici umani (*rubbertuator*). Faţă de o comandă PID clasică, reacţia vizuală permite rezolvarea problemei de stabilire exactă a parametrilor, a întârzierii care apare între traiectoria dorită şi cea realizată, şi a histerezei, datorate flexibilităţii care face ca modelul dinamic al braţului să fie complex şi puternic neliniar. Camera montată pe braţ, care rămâne orizontală, serveşte atât pentru sesizarea poziţionării corecte în plan orizontal şi a distanţei rămase până la obiect, prin aria proiecţiei perspective realizate (§ 2.1). Ea este însă folosită şi pentru măsurarea vitezei şi acceleraţiei braţului, pentru obţinerea unei mişcări line.

Un sistem cuprinzând două camere montate pe braţul de robot este abordat în (Maru *et al.*, 1993, citat de Hager *et al.*, 1994.)

Câteva din aplicaţiile comenzii robotice cu reacţie vizuală, menţionate în (Corke, 1994), care dă şi trimiterile bibliografice, sunt : deservirea benzilor rulante; mişcarea ghidată vizual; asamblarea, sudura, eventual pe maşina aflată în mişcare; aplicaţii militare precum urmărirea automată a rachetelor; ghidarea automată a autovehiculelor; maşini ce joacă ping-pong, jonglează cu popice, ţin în echilibru un obiect (pendul invers); prinderea de obiecte aflate în zbor pe Pământ sau în spaţiu; maşini de cules fructe; cuplarea unui conector al rampei de lansare a navetei spaţiale.

1.3.3. Comandă cu reacţie vizuală în prezenţa erorilor

Prin erorile de aproximare şi de estimare (§ 1.1.4), modelele adaptive sunt doar aproximaţii ale sistemelor modelate. Aceasta face ca ele să fie mai puţin adecvate unei comenzi directe dacă în aplicaţia vizată este necesară o precizie ridicată. Erorile pot fi datorate braţului de robot, spre exemplu urmare a jocului sau rigidităţii imperfecte, sau sistemului vizual, prin modelarea imperfectă a camerei, prin prezenţa aberaţiilor optice, etc. Un alt exemplu este eroarea de cuantizare introdusă de camerele CCD, a căror modelare şi studiu teoretic nu au fost întâlnite în lucrările consultate. Dar ceea ce deranjează în practică este cunoaşterea imperfectă a parametrilor modelelor optice şi a poziţionării camerelor. Comanda cu reacţie vizuală ce utilizează un sistem vizual necalibrat face de asemenea obiectul unor lucrări, şi poate fi încadrată în comandă în prezenţa erorilor.

Dacă însă este posibilă realizarea unei bucle de reacţie, se pot obţine precizii ridicate şi în prezenţa erorilor (Bassi şi Bekey, 1989; Arimoto şi Miyazaki, 1983, citat de Feddema *et al.*, 1994; Samson *et al.*, 1991). Prin captarea poziţiei braţului de robot în mediul de lucru sau în raport cu ţinta dorită, reacţia vizuală este adecvată în mod particular realizării unei bucle de reacţie pentru comanda roboţilor în prezenţa erorilor.

Creșterea preciziei de poziționare prin reacție poate fi efectuată explicit sau implicit. În primul caz sistemul este reprezentat ca un sistem cu reacție și, în mod natural, bucla este parcursă de un număr de ori suficient pentru atingerea preciziei de poziționare dorite. Este cazul buclei din (Bassi și Bekey, 1989; Hollinghurst și Cipolla, 1993; Hager *et al.*, 1994; Jägersand și Nelson, 1994). În al doilea caz, se consideră că, după o poziționare relativ imprecisă, pentru a se ajunge la țintă mai trebuie efectuată o corecție, care corespunde, de fapt, unei parcurgeri a buclei de reacție. Și într-un caz și în celălalt, este necesar un model de variații, care este fie matricea jacobiană a transformării globale captor+efector, obținându-se o comandă cinematică în poziție sau în viteză, fie, în cazul unei comenzi dinamice, aproximarea liniară dintre cupluri și accelerații. În această lucrare se tratează cazul mai simplu, al comenzii cinematice. Însă nici pentru acesta, literatura nu este prea bogată în prezentarea domeniului erorilor jacobienei, pentru care bucla de reacție este stabilă sau convergentă.

Bassi și Bekey (1989) par să fie primii care demonstrează, prin simulare, că se poate obține o poziționare oricât de precisă chiar atunci când comanda se face cu erori, dacă se utilizează o reacție negativă iterativă. Ei comandă dinamic un braț cu două grade de libertate, prin metode neuronale. Prezența erorilor în comanda directă face necesară utilizarea unei bucle de reacție. Întrucât aceasta presupune înmulțirea cu o matrice, este introdusă "rețeaua dependentă de context" (*context dependent neural network*, Yeung și Bekey, 1993; v. Cimponeriu, 1994a), care învață, în locul funcției produs, doar matricea înmulțitor. Aceasta va fi și metoda adoptată în lucrarea de față. Ei constată că, în prezența erorilor, eroarea de poziționare scade exponențial cu scăderea perioadei de eșantionare a buclei, ceea ce corespunde creșterii numărului de iterații. Însă nu este explicitată legătura dintre viteza de convergență spre zero a erorii de poziționare și eroarea de comandă, și nici nu se specifică eroarea de comandă maximă pentru care brațul poate fi poziționat.

Hollinghurst și Cipolla (1993) realizează o comandă cu reacție vizuală în care calibrarea sistemului optic nu este necesară, ea făcându-se automat, prin urmărirea unui număr de puncte ale unui contur de referință, plan. Ei utilizează o buclă de comandă integrală, termenul de reacție fiind diferența dintre vectorii vizuali ce descriu poziția și orientarea țintei și respectiv a efectorului robotului. Manipulatorul se mișcă în pași discreți. Poziția carteziană a punctelor de interes este calculată aproximând transformarea realizată de camere prin proiecția perspectivă slabă (*weak perspective*), valabilă aproximativ pentru o distanță cameră-obiect suficient de mare. Ei constată experimental o imunitate remarcabilă la translații și rotații, ca și la modificări ale distanței focale ale camerelor, chiar după autocalibrare. Reacția vizuală face ca sistemul lor să fie imun și la distorsiuni neliniare ale cinematicii robotului. Ei menționează doar calitativ aceste proprietăți.

Jägersand și Nelson (1994) investighează experimental proprietățile comenzii unor brațe de robot cu 3 sau mai multe grade de libertate. Legat de problematica acestei teze, ei constată experimental că reacția vizuală elimină problema limitării preciziei de poziționare datorită erorilor modelului cinematic al brațului. Este indicat ca unghiul dintre cele două camere să fie mai mare de 30°. Repetabilitatea poziționării este de câteva ori mai bună în cazul utilizării reacției vizuale, decât în cazul reacției făcute prin unghiurile articulațiilor. Precizia poziționării crește cu numărul de trăsături vizuale utilizate. Ei modelează local liniar sistemul braț – camere video prin jacobiana transformării compuse, pe care o învață prin algoritmul LMS normalizat (v. § 3.1). În Jägersand (1996) este introdusă "metoda regiunii de încredere" (*trust region method*), care adaptează mărimea deplasării la fiecare iterație, astfel încât eroarea

ce se face prin modelarea local liniară a unei transformări neliniare să nu depășească un prag ales. Eroarea de modelare este definită prin

$$d = \frac{\Delta \mathbf{y}_{masurat}}{\mathbf{J}\delta}, \qquad (1.3.1)$$

unde numărătorul reprezintă deplasarea vizuală obținută, iar numitorul se poate presupune că este deplasarea cerută. Nu este analizată teoretic dimensiunea regiunii de încredere. Nu se fac nici aprecieri asupra erorilor de comandă, pentru care stabilitatea sau convergența poziționării se păstrează.

Există totuși unele rezultate teoretice în această direcție. Sunt cunoscute condiții generale ce asigură stabilitatea globală asimptotică a buclei de reacție, în prezența perturbațiilor, a variației parametrilor sau a erorilor de modelare. Lucrarea de referință (Goodwin și Sin, 1984) prezintă câteva considerații generale de imunitate la perturbații, și o analiză succintă a senzitivității parametrilor, în cazul general al sistemelor liniare cu timp discret, fără însă a detalia subiectul domeniului erorilor admisibile. Pentru sisteme cu timp continuu, problema stabilității este abordată în monografia (Samson *et al.*, 1991), dedicată urmăririi traiectoriilor impuse în spațiul articulațiilor.

Hager și colegii săi (1994) au obținut unele rezultate teoretice și experimentale privind erorile admisibile la poziționarea camerelor video pentru un braț de robot cu reacție vizuală. Camerele lor sunt amplasate paralel, pentru a se obține doar trei coordonate imagine diferite, astfel încât matricea jacobiană a transformării optice să fie pătrată și inversabilă. Pentru o comandă proporțională, ei modelează analitic eroarea de orientare a camerelor și a distanței dintre camere, și obțin regiuni de stabilitate pentru modificări ale orientărilor de până la 15°. Orientarea camerelor spre interior față de poziția estimată tinde să destabilizeze sistemul mai repede decât orientarea spre exterior. Distanța dintre camere acționează ca un factor de scală, și nu afectează stabilitatea asimptotică a buclei. Calibrarea sistemului se realizează printr-o serie de mișcări cunoscute. Nu sunt studiate erorile admisibile la modelarea robotului.

Comanda vizuală a roboților face și obiectul tezei (Yoshimi, 1995), dar care nu se apropie prea mult de cele dezvoltate în lucrarea de față.

2. Cerinţe de precizie pentru convergenţă în comanda cinematică iterativă cu reţele neuronale

Acest capitol studiază posibilitatea comenzii cinematice a unui braţ de robot cu reacţie vizuală în prezenţa erorilor inerente oricărei modelări, în particular a modelării adaptive prin reţele neuronale. Pentru început, se prezintă braţul de robot pe care se concretizează calculele, şi sistemul vizual cu care se doreşte a se comanda braţul. Sunt apoi prezentate două modalităţi de comandă cinematică derivate din literatură. Este analizată partea comună iterativă a algoritmilor şi condiţia generală de convergenţă asimptotică. Analiza este apoi particularizată pentru braţul de robot. Rezultatele sunt obţinute astfel încât să fie aplicabile în realizarea concretă a unei comenzi neuronale.

2.1 Braţul de robot şi sistemul vizual, descriere şi modelare

Studiul acestei teze se concretizează pe o platformă robotică ce cuprinde un braţ manipulator şi un sistem vizual alcătuit din două camere CCD. Platforma face obiectul preocupărilor colectivului laboratorului de automatică TROP al Université de Haute-Alsace, Mulhouse, Franţa.

2.1.1. Braţul de robot ERICC

Braţul de robot, de tipul ERICC (Barras Provence), este reprezentat în figura 2.1.1. Dimensiunile, ca şi domeniile de variaţie a unghiurilor sunt cele din tabelul 2.1.1.

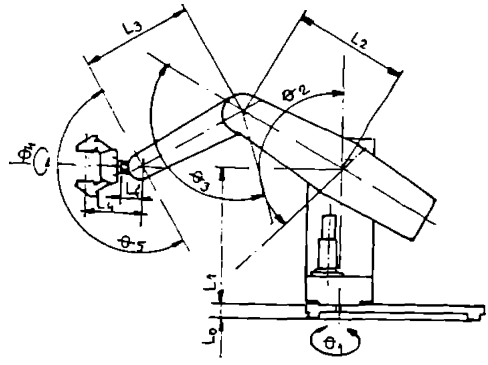


Figura 2.1.1 Braţul manipulator ERICC

Un model simplificat al braţului este reprezentat în figura 2.1.2 şi descris de ecuaţiile (2.1.1). Referinţele unghiurilor sunt cele specificate de constructor.

Brațul ERICC dispune de regulatoare PID pe fiecare axă și poate fi comandat absolut, prin valoarea unghiurilor corespunzând poziției care se dorește a fi obținută, sau incremental, prin variații ale unghiurilor.

Notație	Dimensiune [mm]	Valoare minimă [°]	Valoare maximă [°]
L_1	340		
L_2	280		
L_3	317.5		
L_4	150		
θ_1		-135	135
θ_2		-45	90
θ_3		-45	90
θ_4		-180	180
θ_5		th3-90	th3 + 90

Tabelul 2.1.1. Date constructive ale manipulatorului ERICC

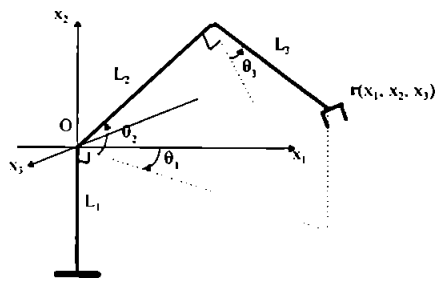


Figura 2.1.2. Reprezentarea simplificată a brațului de robot

$$
\begin{aligned}
&x_1 = [L_2 \cos\theta_2 + L_3 \sin(\theta_2 + \theta_3)]\cos\theta_1 \\
&x_2 = L_2 \sin\theta_2 - L_3 \cos(\theta_2 + \theta_3) \\
&x_3 = [L_2 \cos\theta_2 + L_3 \sin(\theta_2 + \theta_3)]\sin\theta_1 \\
&L_1 = 340 \text{ mm};\ L_2 = 280 \text{ mm};\ L_3 = 467 \text{ mm}; \\
&-135° \le \theta_1 \le 135°; -45° \le \theta_2 \le 90°; -45° \le \theta_3 < 90°; \\
&d \ge 140 \text{ mm}
\end{aligned}
\qquad (2.1.1)
$$

2.1.2. Sistemul vizual stereoscopic

Sistemul vizual este, în intenția laboratorului TROP, antropomorf. Se dorește ca cele două camere CCD să urmărească ținta, printr-un sistem de servomotoare, ca în figura 2.1.3. Câteva din caracteristicile celor două camere sunt cele din tabelul 2.1.2. (Micam X).

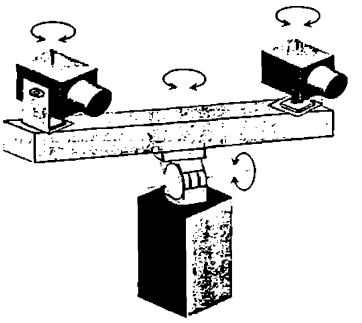


Figura 2.1.3. Sistemul vizual vizat de laboratorul TROP

pixeli	500 (H) x 582 (V)
dimensiune celulă	12.7 (H) x 8.3 (V) μm
zona sensibilă	6,4 (H) x 4,8 (V) mm
distanţa captor - lentilă D	17.5 mm

Tabelul 2.1.2 Caracteristici ale camerelor CCD

Cele două camere vor fi notate cu B şi C. Amplasarea lor în raport cu robotul poate fi modelată ca în figura 2.1.4.

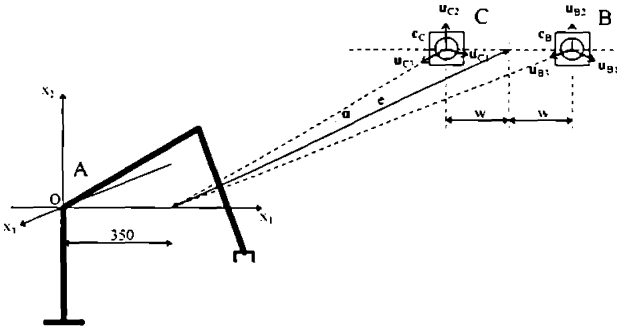


Figura 2.1.4. Dispunerea celor două camere

În această lucrare se va considera dispunerea camerelor dată de :

$$w = 500, \quad e = 5000 \ [mm]. \tag{2.1.2}$$

Se ştie (*e.g.* Craig, 1986) că la trecerea de la un sistem de coordonate A la un sistem de coordonate B, un punct oarecare $\mathbf{r}$ suferă o translaţie şi o rotaţie :

$$^{A}\mathbf{r} = {}^{A}\mathbf{o}_{B} + {}^{A}\mathbf{R}_{B}\,{}^{B}\mathbf{r},$$

$$^{A}\mathbf{R}_{B} = \left[{}^{A}\hat{\mathbf{x}}_{1,B} \quad {}^{A}\hat{\mathbf{x}}_{2,B} \quad {}^{A}\hat{\mathbf{x}}_{3,B}\right]$$

unde

$^{A}\mathbf{r}$ este $\mathbf{r}$ exprimat în coordonatele referenţialului A.

$^{A}\mathbf{o}_{B}$ este originea referenţialului B în referenţialul A.

$^{A}\mathbf{R}_{B}$ este matricea de rotaţie a referenţialului B faţă de A,

$^{A}\hat{\mathbf{x}}_{1,B}$ este versorul axei x_1 a referenţialului B exprimat în A.

În particular, A reprezintă sistemul de coordonate obiect, legat de robot, iar B este sistemul de coordonate legat de camera B. Pentru ambele camere, conform figurii 2.1.4 şi alegerii (2.1.2), se poate scrie:

$$ {}^{A}\mathbf{o}_{B} = \begin{pmatrix} 350 - w \\ 0 \\ -e \end{pmatrix}, \quad {}^{A}\mathbf{R}_{B} = \begin{bmatrix} \begin{pmatrix} \cos\alpha \\ 0 \\ -\sin\alpha \end{pmatrix} & \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} & \begin{pmatrix} \sin\alpha \\ 0 \\ \cos\alpha \end{pmatrix} \end{bmatrix}, \tag{2.1.3} $$

$$ {}^{A}\mathbf{o}_{C} = \begin{pmatrix} 350 + w \\ 0 \\ -e \end{pmatrix}, \quad {}^{A}\mathbf{R}_{C} = \begin{bmatrix} \begin{pmatrix} \cos\alpha \\ 0 \\ \sin\alpha \end{pmatrix} & \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} & \begin{pmatrix} -\sin\alpha \\ 0 \\ \cos\alpha \end{pmatrix} \end{bmatrix}, \tag{2.1.4} $$

cu $\alpha = \operatorname{arctg}\left(\frac{w}{e}\right)$, $w = 500$, $e = 5000$ [mm].

În continuare se tratează transformarea realizată de camera B. Pentru camera C se procedează similar.

În referenţialul camerei B punctul **r** este :

$$ {}^{B}\mathbf{r} = {}^{A}\mathbf{R}_{B}^{-1}\left({}^{A}\mathbf{r} - {}^{A}\mathbf{o}_{B}\right) = {}^{A}\mathbf{R}_{B}^{T}\left({}^{A}\mathbf{r} - {}^{A}\mathbf{o}_{B}\right), \tag{2.1.5} $$

întrucât ${}^{A}\mathbf{R}_{B}$ este ortogonală.

Fiecare din cele două camere are o lentilă ce realizează pe captor o proiecţie perspectivă. Cel mai simplu model al proiecţiei perspective, în aproximarea unei lentile cu o diafragmă centrală punctiformă aflată la distanţa D de captor, este cel din figura 2.1.5 (Breton, 1994).

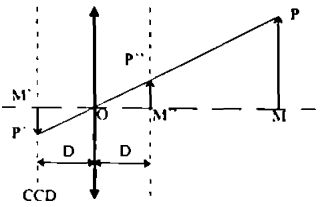


Figura 2.1.5. Aproximarea unei lentile cu diafragmă centrală punctiformă

Într-o formă intuitivă, proiecţia vizuală, obţinută ca în figura 2.1.6, unde parametrii dispunerii sunt (2.1.3')-(2.1.4'), poate fi exprimată prin ecuaţiile (2.1.6) (Breton, 1994; Hager *et al.*, 1994), unde prin h_B şi v_B s-a notat coordonata pe orizontală respectiv verticală pe ecranul camerei B.

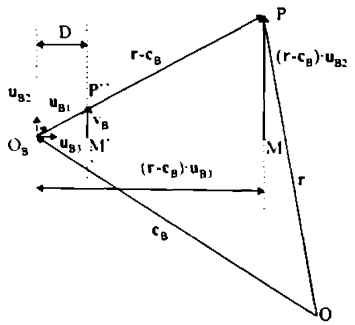


Figura 2.1.6. Proiecţia perspectivă realizată pe ecranul unei camere :
O - originea sistemului de referinţă;
r - vectorul vizualizat;
$\mathbf{c}_B$-centrul lentilei B;
$\mathbf{u}_{B1}$, $\mathbf{u}_{B2}$, $\mathbf{u}_{B3}$ - orientarea camerei B;

$$\begin{cases} \mathbf{c}_B = [350 + w \quad 0 \quad 3000]^T \\ \mathbf{u}_{B1} = [cos(\alpha) \quad 0 \quad sin(\alpha)]^T \\ \mathbf{u}_{B2} = [0 \quad 1 \quad 0]^T \\ \mathbf{u}_{B3} = [-sin(\alpha) \quad 0 \quad cos(\alpha)]^T \end{cases} \tag{2.1.3'}$$

$$\begin{cases} \mathbf{c}_C = [350 - w \quad 0 \quad 3000]^T \\ \mathbf{u}_{C1} = [cos(\alpha) \quad 0 \quad -sin(\alpha)]^T \\ \mathbf{u}_{C2} = [0 \quad 1 \quad 0]^T \\ \mathbf{u}_{C3} = [sin(\alpha) \quad 0 \quad cos(\alpha)]^T \end{cases} \tag{2.1.4'}$$

cu $\alpha = \text{arctg}\left(\frac{w}{e}\right)$, $w = 500$, $e = 5000$ [mm]

$$\begin{cases} h_B = D\dfrac{(\mathbf{r} - \mathbf{c}_B)\cdot \mathbf{u}_{B1}}{(\mathbf{r} - \mathbf{c}_B)\cdot \mathbf{u}_{B3}} & h_C = D\dfrac{(\mathbf{r} - \mathbf{c}_C)\cdot \mathbf{u}_{C1}}{(\mathbf{r} - \mathbf{c}_C)\cdot \mathbf{u}_{C3}} \\ v_B = D\dfrac{(\mathbf{r} - \mathbf{c}_B)\cdot \mathbf{u}_{B2}}{(\mathbf{r} - \mathbf{c}_B)\cdot \mathbf{u}_{B3}} & v_C = D\dfrac{(\mathbf{r} - \mathbf{c}_C)\cdot \mathbf{u}_{C2}}{(\mathbf{r} - \mathbf{c}_C)\cdot \mathbf{u}_{C3}} \end{cases}, \tag{2.1.6}$$

unde · este produsul scalar.

Se preferă însă forma matricială, mai eficientă în simularea MATLAB. Întrucât se consideră ecranul CCD plasat perpendicular pe ${}_B\hat{x}_3$, la distanţa D de origine, în modelul diafragmei punctuale vectorul vizualizat ${}^B\mathbf{r}$, originea $\mathbf{o}_B$ (centrul de proiecţie) şi vectorul imagine $\mathbf{ri}_B = (h_B \quad v_B \quad D)^T$ sunt coliniari. Ecuaţia de proiecţie, cunoscută în literatura de fotogrammetrie drept ecuaţia de coliniaritate (Lu *et al.*, 1994), poate fi scrisă ca:

$$\mathbf{ri}_B = \begin{pmatrix} h_B \\ v_B \\ D \end{pmatrix} = \frac{D}{{}^B\mathbf{r}\ {}^B\hat{\mathbf{x}}_{3,B}^T}\ {}^B\mathbf{r} = \frac{D}{{}^A\hat{\mathbf{x}}_{3,B}^T\left({}^A\mathbf{r} - {}^A\mathbf{o}_B\right)}\ {}^A\mathbf{R}_B^T\left({}^A\mathbf{r} - {}^A\mathbf{o}_B\right). \tag{2.1.7}$$

În fine, camera CCD are receptorii de dimensiuni q_h=12.7 μm, q_v=8.3 μm (tabelul 2.1.2). Dacă se consideră ecranul CCD centrat pe 0, indicii pixelilor corespunzători sunt

$$\mathbf{p}_B = \begin{pmatrix} ph_B \\ pv_B \end{pmatrix} = \begin{pmatrix} \frac{h_B}{q_h} \\ \frac{v_B}{q_v} \end{pmatrix} = \begin{bmatrix} q_h^{-1} & 0 \\ 0 & q_v^{-1} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{ri}_B. \tag{2.1.8}$$

unde $\mathbf{I}_{2x3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ selectează doar componentele h şi v.

În funcţionarea reală, funcţiile ph şi pv realizează şi o conversie analog-numerică, ce poate fi modelizată prin funcţia parte întreagă. Dar în lucrarea de faţă acest efect se neglijează, el putând face obiectul altui studiu.

Se obţine astfel vectorul $\mathbf{p}_{4x1}$ al celor 2x2 coordonate ale pixelilor :

$$\mathbf{p} = \begin{pmatrix} \mathbf{p}_B \\ \mathbf{p}_C \end{pmatrix} = D\ diag\left(q_h^{-1}, q_v^{-1}, q_h^{-1}, q_v^{-1}\right) \begin{bmatrix} \mathbf{I}_{2x3} & \mathbf{0}_{2x3} \\ \mathbf{0}_{2x3} & \mathbf{I}_{2x3} \end{bmatrix} \begin{pmatrix} \dfrac{{}^A\mathbf{R}_B^T\left({}^A\mathbf{r} - {}^A\mathbf{o}_B\right)}{{}^A\hat{\mathbf{x}}_{3,B}^T\left({}^A\mathbf{r} - {}^A\mathbf{o}_B\right)} \\ \dfrac{{}^A\mathbf{R}_C^T\left({}^A\mathbf{r} - {}^A\mathbf{o}_C\right)}{{}^A\hat{\mathbf{x}}_{3,C}^T\left({}^A\mathbf{r} - {}^A\mathbf{o}_C\right)} \end{pmatrix}, \qquad (2.1.9)$$

unde $\mathbf{0}_{2x3}$ este matricea nulă de dimensiune 2x3.

Sistemul vizual realizează o transformare din spaţiul cartezian tridimensional în cele două perechi de coordonate pe ecranele camerelor CCD, într-un spaţiu de 4-dimensional.

Deşi pentru scopul acestei lucrări este suficientă transformarea vizuală (2.1.9), pentru posibilitatea verificării rezultatelor din capitolul 4 al tezei este utilă şi obţinerea jacobienei acestei transformări. Derivând (2.1.7) în raport cu ${}^A\mathbf{r}$, se obţine matricea jacobiană a transformării ${}^B\mathbf{ri}({}^A\mathbf{r})$,

$$\mathbf{Jri}_B = D\ \frac{{}^A\mathbf{R}_B^T}{{}^A\hat{\mathbf{x}}_{1,B}^T\left({}^A\mathbf{r} - {}^A\mathbf{o}_B\right)} \left(\mathbf{I}_3 - \frac{\left({}^A\mathbf{r} - {}^A\mathbf{o}_B\right){}^A\hat{\mathbf{x}}_{3,B}^T}{{}^A\hat{\mathbf{x}}_{3,B}^T\left({}^A\mathbf{r} - {}^A\mathbf{o}_B\right)} \right), \qquad (2.1.10)$$

unde $\mathbf{I}_3$ este matricea unitate 3x3.

Analog se obţine matricea $\mathbf{Jri}_C$ pentru camera C.

Matricea jacobiană a transformării vizuale (2.1.9) este aşadar :

$$\mathbf{Jv} = D\ diag\left(q_h^{-1}, q_v^{-1}, q_h^{-1}, q_v^{-1}\right) \begin{bmatrix} \mathbf{I}_{2x3} & \mathbf{0}_{2x3} \\ \mathbf{0}_{2x3} & \mathbf{I}_{2x3} \end{bmatrix} \begin{pmatrix} \mathbf{Jri}_B \\ \mathbf{Jri}_C \end{pmatrix}. \qquad (2.1.11)$$

2.2. Comandă cinematică iterativă cu reacţie vizuală

Se consideră sistemul din figura 2.2.1. Camerele recunosc ţinta **t** şi efectorul braţului de robot **r**, şi transmit blocului de comandă (neuronală) cele 4 coordonate vizuale ale vectorilor respectivi. Acesta din urmă trebuie să transmită robotului comanda sau şirul de comenzi care să suprapună pe **r** peste **t**.

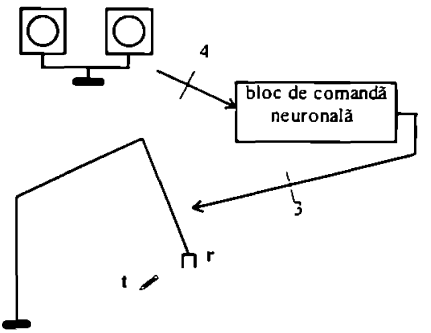


Figura 2.2.1. Braţ de robot cu reacţie vizuală şi comandă neuronală

2.2.1. Două scheme de comandă cinematică

Sistemul cu reacţie din figura 2.2.1 poate fi realizat, în principiu, ca în figura 2.2.2.

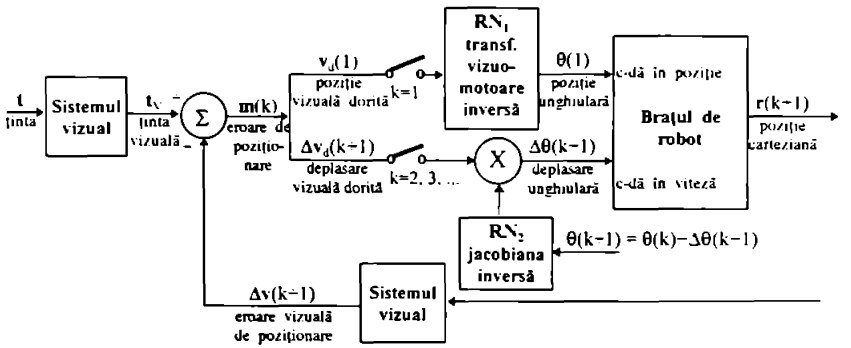


Figura 2.2.2. Comanda cinematică iterativă globală şi locală bazată pe imagine

În această schemă se poate încadra şi soluţia propusă de Ritter *et al.* (1992). Ei utilizează o reţea Kohonen extinsă (v. § 1.2.3), care dă variante discretizate, conform cuantizării vectoriale efectuate, pe de o parte ale transformării vizuo-motoare (compusa transformării vizuale şi a transformării cinematice a braţului de robot) inverse, iar pe de alta, ale inversei matricii

jacobiene a transformării vizuo-motoare, $\tilde{J}^{-1}$. După o primă poziţionare dată de primii doi termeni ai dezvoltării în serie Taylor ai transformării cinematice inverse, dacă eroarea de poziţionare nu e suficient de mică, se mai face o iteraţie cu $\tilde{J}^{-1}$ şi eroarea scade suficient. De menţionat că autorii citaţi nu consideră explicit că această procedură corespunde unui sistem cu reacţie negativă.

În această lucrare se consideră că poziţionarea iniţială se face conform doar unei valori aproximative a transformării cinematice inverse dată de o primă reţea, RN_1, şi că cea de-a doua reţea, RN_2 dă valoarea $\tilde{J}^{-1}$ cu erori mai mari, astfel încât sunt necesare mai multe iteraţii. De asemenea, se presupune că $\tilde{J}^{-1}$ este dat de valoarea exactă a lui θ curent, şi nu de varianta sa cuantizată. Totuşi, această din urmă posibilitate este avută în vedere în calculele ce vor urma. Funcţionarea acestei scheme de comandă este ilustrată în figura 2.2.3.

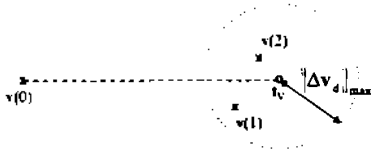


Figura 2.2.3. Ilustrarea funcţionării algoritmului corespunzător figurii 2.2.2.

Să presupunem că poziţia iniţială pe ecranele CCD este $\mathbf{v}_0$, şi că distanţa la ţinta vizuală $\mathbf{t}_V$ este prea mare pentru ca să poată fi utilizată o aproximare liniară. La prima iteraţie comanda este dată de NN_1. Comanda nu este destul de precisă pentru ca cleştele robotului **r** să se suprapună peste ţinta **t**, sau ca **v**, imaginea lui **r**, să se suprapună peste $\mathbf{t}_V$. Dar eroarea comenzii este suficient de mică pentru ca poziţia atinsă **v**(1) să fie în interiorul cercului de rază $\|\Delta v_d\|_{max}$ cu centrul în $\mathbf{t}_V$. Acesta asigură validitatea modelului liniar, care acum poate fi aplicat. Eroarea de poziţionare rămasă **m** (de la *misplacement*) dă deplasarea dorită $\Delta \mathbf{v}_d$: $\mathbf{m}(1)=\Delta \mathbf{v}_d(1)=\mathbf{t}_V-\mathbf{r}(1)$, care este multiplicată cu ieşirile reţelei NN_2. Comanda rezultantă $\Delta\theta$ este trimisă robotului, care se deplasează în **r**(2), poziţie văzută ca **v**(2). Dacă NN_2 ar furniza valoarea exactă pentru $\mathbf{J}^{-1}$ şi dacă **m** ar fi foarte mic, atunci **v**(2) ar deveni egal cu $\mathbf{t}_V$, ceea ce ar însemna **r**(2)=**t**. Întrucât avem valori mai mari pentru **m**, ceea ce face ca modelul local liniar să fie doar aproximativ valabil, iar valoarea $\tilde{J}^{-1}$ este doar o aproximare a valorii exacte J^{-1}, aceasta nu se întâmplă. Însă cerem ca **v**(2) să fie de q<1 ori mai aproape de **t** decât **v**(1). Eroarea rămasă **m**(2) dă deplasarea dorită $\Delta \mathbf{v}_d(2)$, cu care, prin înmulţire cu $\tilde{J}^{-1}$, se obţine o nouă comandă şi braţul se apropie de ţintă. Această buclă se repetă în acelaşi mod până când este obţinută o eroare de poziţionare suficient de mică. NN_2 trebuie doar să aibă erori suficient de mici, astfel încât eroarea de poziţionare să diminueze de q ori la fiecare iteraţie, asigurând convergenţa asimptotică a lui **m** spre **0**. Condiţia q mai mic dar apropiat de 1 este cea mai puţin restrictivă posibilă pentru NN_2. Robotul trebuie să poată accepta două tipuri de comenzi: în poziţie şi în viteză.

O variantă a schemei din figura 2.2.2 este reprezentată în figura 2.2.4. Aceasta este o comandă incrementală de poziţie, bazată pe principiul optimului dinamic (*dynamic optimality principle*), conform căruia se urmăreşte traiectoria curentă optimă (Bassi şi Bekey, 1989), care urmăreşte traiectoria optimă din poziţia curentă spre ţintă şi este reprezentat în figura 2.2.5. Blocul programator stabileşte ţinte curente corespunzând unor mici deplasări în direcţia ţintei

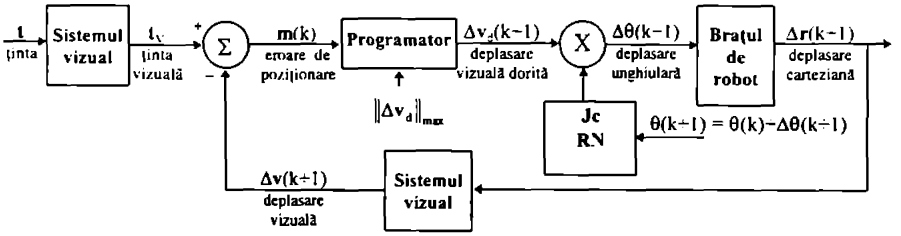


Figura 2.2.4. Comandă cinematică iterativă local liniară bazată pe imagine

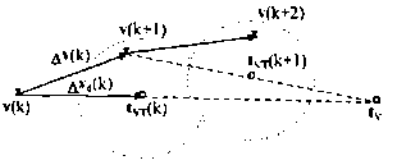


Figura 2.2.5. Principiul optimului dinamic, pentru comanda cinematică iterativă local liniară, cu $q \cong 0{,}5$.

vizuale, la o distanță de cel mult $\|\Delta v_d\|_{max}$. Cerința ca mărimea deplasării din timpul unei iterații să nu depășească valoarea maximă $\|\Delta v_d\|_{max}$ este impusă pentru asigurarea validității modelului local liniar. Să presupunem că imaginea țintei este t_V, iar poziția vizuală la iterația k este v(k). Dacă între ele nu există obstacole, programatorul S stabilește o țintă temporară $t_{VT}(k)$ aflată în direcția lui t_V, la o distanță mai mică sau egală cu $\|\Delta v_d\|_{max}$, obținând deplasarea dorită $\Delta v_d(k)=v(k)-t_{VT}(k)$. Datorită erorii de comandă, imaginea brațului nu devine $t_{VT}(k)$, ci v(k+1). Însă erorile blocului de comandă sunt suficient de mici ca să asigure ca distanța rămasă între v(k+1) și $t_{VT}(k)$, $\Delta v_r(k)=v(k+1)-t_{VT}(k)$, să scadă de q ori față de distanța dintre v(k) și $t_{VT}(k)$. Cu alte cuvinte,

$$\frac{\|\Delta v_r(k)\|}{\|\Delta v_d(k)\|} = q < q_{max} \le 1 \qquad (2.2.1)$$

Aceasta înseamnă că următoarea poziție v(k+1) se va situa în interiorul cercului centrat în $t_{VT}(k)$ de rază $q\|\Delta v_d(k)\|$, mai aproape de $t_{VT}(k)$. O a doua țintă temporară $t_{VT}(k+1)$ este aleasă, în direcția lui t, la o distanță maximă $\|\Delta v_d\|_{max}$. Noua poziție v(k+2) va fi în interiorul cercului cu centrul în $t_{VT}(k+1)$, mai aproape de $t_{VT}(k+1)$, și așa mai departe. Întrucât la fiecare iterație, t_{VT} este tot mai aproape de t_V, distanța dintre v(k) și t_V descrește. *A fortiori*, ultimii pași sunt toți în aceeași vecinătate de rază $\|\Delta v_d\|_{max}$ a țintei, iar eroarea descrește exponențial spre zero, ca în algoritmul corespunzător figurilor 2.2.2, 2.2.3.

Se poate menționa că, dacă în locul unei poziționări finale în ținta t, se dorește urmărirea unei traiectorii date, se pot utiliza drept ținte temporare puncte intermediare (*way points*) de pe traiectoria dorită (Jägersand și Nelson, 1994).

Se consideră că pozițiile succesive ale brațului sunt discrete, atinse în momente egale cu multiplii unei perioade de eșantionare T_e. Dacă nu se așteaptă oprirea brațului la fiecare

pas, comanda unor mici deplasări unghiulare $\Delta\theta$ în intervale $\Delta t=T_e$ devine o comandă în viteză în timp discret.

2.2.2. Modelul iterativ general pentru întreaga buclă de reacţie

Ambii algoritmi prezentaţi în paragraful precedent au ca parte centrală aceeaşi buclă iterativă, reprezentată în figura 2.2.6.

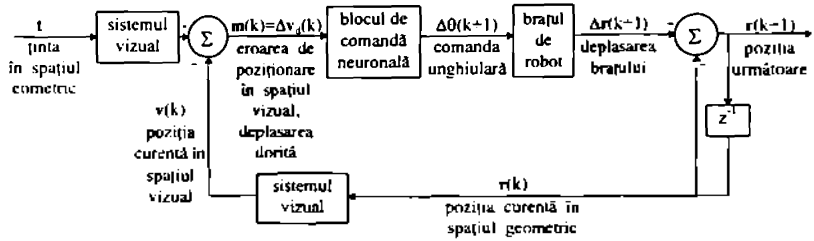


Figura 2.2.6. Modelul iterativ general pentru bucla de reacţie

Bucla reprezintă funcţionarea ambilor algoritmi, identică atunci când v(k) este la o distanţă de ţintă mai mică decât $|\Delta v_d|_{max}$. Aceasta se obţine implicit pentru figurile 2.2.2-2.2.3. Pentru schema din figurile 2.2.4-2.2.5, dacă această distanţă este mai mare, se poate considera că ţinta **t** este ţinta temporară stabilită de programator, modificată la fiecare iteraţie, şi că de fiecare dată se face o singură parcurgere a buclei.

2.2.3. Liniarizarea buclei de reacţie

Transformările efectuate de sistemul vizual şi de braţul de robot sunt neliniare. Dacă însă se consideră că deplasările sunt mici, ambele pot fi liniarizate pe porţiuni. Aceasta înseamnă că transformările devin înmulţiri cu matricile jacobiene respective. Devine astfel posibilă o analiză relativ simplă folosind unele rezultate din algebra liniară. Ecuaţiile modelului liniar pe porţiuni sunt (2.2.2) - (2.2.4):

- braţul de robot:

$$\mathbf{r}(\theta + \Delta\theta) = \mathbf{r}(\theta) + \mathbf{Jr}\ \ \Delta\theta = \mathbf{r}(\theta) + \Delta\mathbf{r}\,, \qquad (2.2.2)$$

unde **Jr** este matricea jacobiană a transformării cinematice a braţului.

- sistemul vizual:

$$\mathbf{v}(\mathbf{r} + \Delta\mathbf{r}) = \mathbf{v}(\mathbf{r}) + \mathbf{Jv}\ \ \Delta\mathbf{r} = \mathbf{v}(\mathbf{r}) + \Delta\mathbf{v}\,, \qquad (2.2.3)$$

unde **Jv** este matricea jacobiană a proiecţiei vizuale.

- blocul de comandă:

$$\Delta\theta = \tilde{\mathbf{J}}\mathbf{c}\ \ \Delta\mathbf{v}_d \qquad (2.2.4)$$

unde $\tilde{\mathbf{J}}\mathbf{c}$ este matricea ale cărei elemente sunt date de ieşirile RN.

Datorită liniarității se poate considera că transformarea sistemului vizual se aplică direct diferenței dintre vectorii geometrici ai țintei și poziției curente, obținându-se sistemul din figura 2.2.7.

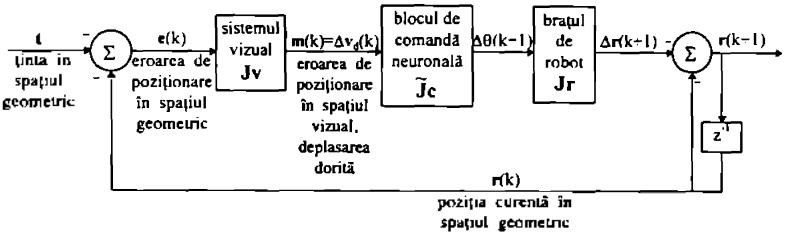


Figura 2.2.7. Modelul iterativ liniarizat

Ecuațiile ce descriu modelul din figura 2.2.7 sunt :

$$\begin{aligned} &\mathbf{t} - \mathbf{r}(k) = \mathbf{e}(k) \\ &\mathbf{m}(k) = \mathbf{Jv}\,\mathbf{e}(k) \\ &\Delta\theta(k+1) = \tilde{\mathbf{Jc}}\,\mathbf{m}(k) \\ &\Delta r(k+1) = \mathbf{Jr}\,\Delta\theta(k+1) \\ &\mathbf{r}(k+1) = \mathbf{r}(k) + \Delta r(k) \end{aligned} \qquad (2.2.5)$$

În consecință, poziția curentă evoluează conform (2.2.6). Dacă ținta e constantă, eroarea de poziționare variază conform (2.2.7).

$$\mathbf{r}(k+1) = \left(\mathbf{I} - \mathbf{Jr}\,\tilde{\mathbf{Jc}}\,\mathbf{Jv}\right)\,\mathbf{r}(k) + \mathbf{Jr}\,\tilde{\mathbf{Jc}}\,\mathbf{Jv}\,\mathbf{t} \qquad (2.2.6)$$

$$\mathbf{m}(k+1) = \left(\mathbf{I} - \mathbf{Jr}\,\tilde{\mathbf{Jc}}\,\mathbf{Jv}\right)\mathbf{m}(k) \qquad (2.2.7)$$

2.2.4. Comanda ideală

Pentru o comandă ideală $\tilde{\mathbf{Jc}} = \mathbf{Jc}$, eroarea se anulează într-un pas:

$$\mathbf{m}(k+1) = \mathbf{0} \Rightarrow \mathbf{Jr}\,\mathbf{Jc}\,\mathbf{Jv} = \mathbf{I} \qquad (2.2.8)$$

Matricile ce apar în ecuația (2.2.8) au dimensiuni diferite: $\mathbf{Jr}^{3x3}$, $\mathbf{Jc}^{3x4}$, $\mathbf{Jv}^{4x3}$, $\mathbf{I}^{3x3}$.

Ecuația (2.2.8) are soluții (Penrose, 1955, citat de Kohonen, 1984), soluția de normă euclidiană minimă fiind (Kohonen, 1984):

$$\mathbf{Jc} = \mathbf{Jr}^{+}\,\mathbf{I}\,\mathbf{Jv}^{+} = \mathbf{Jr}^{+}\,\mathbf{Jv}^{+}, \qquad (2.2.9)$$

unde $\mathbf{A}^{+}$ este pseudoinversa Moore-Penrose a matricii A. Definiția pseudoinversei și unele proprietăți ale ei pot fi găsite în anexa I.

Relația (2.2.9) pune în evidență posibilitatea separării blocului de comandă în două componente: una care furnizează pseudoinversa matricii jacobiene a sistemului vizual $\mathbf{Jv}^{+}$, și alta corespunzătoare brațului de robot $\mathbf{Jr}^{+}$. Evoluția erorii de poziționare (2.2.7) poate fi scrisă atunci ca:

$$\mathbf{m}(k+1) = \left(\mathbf{I} - \left(\mathbf{Jr}\,\mathbf{Jr}^{+}\right)\left(\mathbf{Jv}^{+}\,\mathbf{Jv}\right)\right)\mathbf{m}(k). \qquad (2.2.10)$$

Pentru a se obține anularea erorii de poziționare într-o iterație, $\mathbf{m}(k+1)=0$, pentru orice $\mathbf{m}(k)$, trebuie ca:

$$\mathbf{I} - \left(\mathbf{Jr}\,\mathbf{Jr}^{+}\right)\left(\mathbf{Jv}^{+}\,\mathbf{Jv}\right) = \mathbf{0}. \tag{2.2.11}$$

Produsul $\mathbf{Jr}\,\mathbf{Jr}^{+}$ este matricea proiecție ortogonală pe spațiul liniar generat de coloanele lui **Jr**, iar $\mathbf{Jv}^{+}\,\mathbf{Jv}$ este proiecția ortogonală pe spațiul liniar generat de liniile lui **Jv** (v. anexa 1). Ambele produse au dimensiunea 3x3, ca și matricea unitate din (2.2.11). Este astfel necesar ca:

$$\mathbf{Jr}\,\mathbf{Jr}^{+} = \mathbf{I}. \tag{2.2.12}$$

$$\mathbf{Jv}^{+}\,\mathbf{Jv} = \mathbf{I}. \tag{2.2.13}$$

Semnificația acestor condiții este aceea că liniile matricii $\mathbf{Jr}^{3x3}$ și respectiv coloanele matricii $\mathbf{Jv}^{4x3}$ trebuie să fie liniar independente.

Două observații de ordin practic : Condițiile (2.2.12) și (2.2.13) cer ca rangul matricilor **Jr** și **Jv** să fie 3, dimensiunea spațiului geometric. Aceasta înseamnă că cleștele robotului trebuie să se poată deplasa în orice direcție din spațiul geometric tridimensional, iar sistemul vizual trebuie să poată observa deplasarea după orice direcție, sau să nu existe nici o direcție după care deplasarea nu s-ar reflecta într-o modificare a coordonatelor pe măcar una din cele două camere.

În fine, mai trebuie remarcat faptul că, pentru modelul considerat aici al brațului ERICC (2.1.1), matricea $\mathbf{Jr}^{3x3}$ este pătrată și nesingulară; în acest caz pseudoinversa este chiar inversa matricii.

$$\mathbf{Jr}^{+} = \mathbf{Jr}^{-1}. \tag{2.2.14}$$

2.3. Transformata Z a buclei şi condiţii de stabilitate asimptotică [1]

Acest paragraf studiază condiţiile de stabilitate şi convergenţă asimptotică a buclei de reacţie utilizând extensia transformatei Z pentru sisteme vectoriale. Suplinind o lacună în literatură, aceasta este abordată într-un cadru general, ce vizează sistemele cu timp discret multidimensionale. Ca aplicaţie se studiază răspunsul tranzitoriu şi eroarea de regim staţionar pentru un sistem multidimensional cu reacţie negativă. În final se obţin condiţiile de stabilitate şi convergenţă asimptotică pentru bucla cu reacţie vizuală.

2.3.1. Introducere şi cadrul problemei

Se urmăreşte definirea unei transformate Z pentru sisteme multidimensionale având intrările/ starea/ ieşirile mărimi vectoriale, şi utilizarea ei pentru obţinerea regimului tranzitoriu, a erorii de regim permanent şi a condiţiilor de stabilitate pentru astfel de sisteme.

Sistemele multidimensionale cu timp discret apar în literatură mai ales sub forma bidimensională a sistemelor dedicate prelucrării imaginilor (*e.g.* Pop *et al.*, 1989) sau ca încercări de generalizare a acestora (Dudgeon şi Mersereau, 1984). Transformata Z a unei secvenţe bidimensionale este definită ca :

$$\left[x(n_1,n_2)\right] = X(z_1,z_2) \overset{\Delta}{=} \sum_{n_1=0}^{\infty}\sum_{n_2=0}^{\infty} x(n_1,n_2)\, z_1^{-n_1} z_2^{-n_2} \qquad (2.3.1)$$

Ea derivă din transformata Fourier pentru semnale bidimensionale, ce urmăreşte punerea în evidenţă a proprietăţilor de periodicitate ale semnalelor după diferite direcţii.

Dacă secvenţa consideratã este un şir de vectori n_1-dimensionali, această definiţie ridică cel puţin două obiecţii :

- periodicitatea după n_1 nu are sens,
- transformata şirului de vectori devine un scalar, în timp ce s-ar putea dori ca ea să fie un vector cu aceeaşi dimensiune.

Sistemele multidimensionale sunt studiate şi în manualele de automatică. În modernul lor curs, Rivoire şi Ferrier (1992) aplică ecuaţiilor de stare a sistemelor multidimensionale o extensie a transformatei Z. Călin *et al.* (1984) definesc extensia vectorială implicit. Goodwin şi Sin (1984) prezintă rezultate de convergenţă asimptotică pentru sisteme vectoriale cu ajutorul matricilor polinomiale şi al proprietăţilor lor. Ei însă nu menţionează explicit aplicabilitatea transformatei Z pentru şiruri vectoriale. Nici una din lucrările menţionate nu îi studiază convergenţa şi condiţiile de existenţă a inversei $[zI-F]^{-1}$.

[1] Paragrafele 2.3.1-2.3.4 reproduc articolul (Cimponeriu şi Policec, 1995), cu câteva completări ce au fost necesare după obţinerea lucrării (Goodwin şi Sin, 1984).

Sistemele liniare multidimensionale cu timp discret pot fi descrise prin ecuaţiile matriciale :

$$\mathbf{x}(k+1) = \mathbf{F}\,\mathbf{x}(k) + \mathbf{G}\,\mathbf{u}(k), \text{ ecuaţia de stare cu diferenţe,} \qquad (2.3.2)$$

$$\mathbf{y}(k) = \mathbf{P}\,\mathbf{x}(k) + \mathbf{Q}\,\mathbf{u}(k), \text{ ecuaţia de ieşire,} \qquad (2.3.3)$$

cu $\mathbf{x}(k_0)$, starea iniţială cunoscută, şi matricile **F**, **G**, **P**, **Q** de dimensiuni corespunzătoare vectorilor de intrare **u**, de stare **x** şi de iesire **y**.

Aceste ecuaţii generale cuprind drept caz particular şi bucla de reacţie cu un pol nul, care s-a dorit a fi studiată, reprezentată în figura 2.3.1 şi descrisă de ecuaţia (2.3.4) :

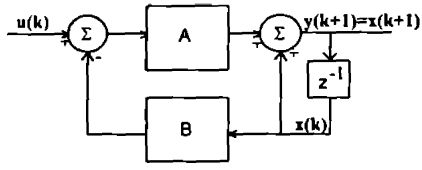


Figura 2.3.1. Buclă de reacţie multidimensională

$$\begin{aligned} &\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{u}(k), \\ &\mathbf{y}(k) = \mathbf{x}(k), \\ &\mathbf{F} = \mathbf{I} - \mathbf{AB},\ \ \mathbf{G} = \mathbf{A} \end{aligned} \qquad (2.3.4)$$

unde intrările, stările şi ieşirile sunt vectori $\mathbf{u}_{m\times 1}$, $\mathbf{x}_{n\times 1}$, $\mathbf{y}_{n\times 1}$.

Se ştie că pentru un şir de scalari complecşi $s(k\in\mathbf{N}) \in \mathbf{C}$, transformata Z (unilaterală) este funcţia complexă de argument complex $S(z) = \mathcal{Z}[s]$, cu

$$S(z) = \sum_{k=0}^{\infty} s(k) z^{-k}. \qquad (2.3.5)$$

S(z) există doar dacă seria (2.3.5) este convergentă, adică dacă z aparţine unui domeniu de convergenţă de forma $|z|>R$.

Exemplu: Pentru şirul $s(k)=a^k$, $a\in\mathbf{C}$, transformata Z este

$$\mathcal{Z}[s] = S(z) = \lim_{K\to\infty} \sum_{k=0}^{K} a^k z^{-k} = \frac{z}{z-a}, \text{ dacă } |z|>|a|.$$

2.3.2. Definiţia transformatei Z pentru sisteme vectoriale

Pentru un şir de vectori m-dimensionali complecşi $\mathbf{v}(k\in\mathbf{N}) = [v_1(k) \ldots v_i(k)] \ldots v_m(k)]^T \in \mathbf{C}^m$, se defineşte transformata Z (monolaterală) drept funcţia vectorială m-dimensională de argument complex V(z), $\mathbf{V}(z) = \mathcal{Z}[\mathbf{v}]$ cu (2.3.6):

$$\mathcal{Z}[\mathbf{v}] = \mathbf{V}(z) = \left[V_1(z) \ldots V_i(z) \ldots V_m(z)\right]^T, \qquad (2.3.6)$$

$$V_i(z) = \sum_{k=0}^{\infty} v_i(k) z^{-k}, \quad i = \overline{1,m}$$

$$\mathcal{Z}[\mathbf{v}] = \mathbf{V}(z) = \sum_{k=0}^{\infty} \mathbf{v}(k)\, z^{-k} \qquad (2.3.6')$$

Cu alte cuvinte, transformata Z a unui şir de vectori este vectorul transformatelor Z scalare ale componentelor. V(z) este convergent dacă toate componentele sale converg, adică dacă $|z| \geq$ *max* R_i, cu R_i - raza de convergenţă a componentei v_i.

Observaţie: Condiţia de convergenţă a seriei $\sum_{k=0}^{\infty} v_i(k)\, z^{-k}$ poate fi obţinută şi studiind norma termenului general.

În condiţiile convergenţei, toate proprietăţile transformării Z scalare se păstrează pentru cazul vectorial. De exemplu:

p1) Liniaritatea.:

$$\mathbf{A}\,\mathbf{v}(k) \xleftrightarrow{\mathcal{Z}} \mathbf{A}\,\mathbf{V}(z) \qquad (2.3.7)$$

p2) Avansul semnalului cu un pas :

$$\begin{aligned} &\mathbf{v}(k) \xleftrightarrow{\mathcal{Z}} \mathbf{V}(z) \\ &\mathbf{v}(k+1) \xleftrightarrow{\mathcal{Z}} z\big(\mathbf{V}(z) - \mathbf{v}(0)\big) \end{aligned} \qquad (2.3.8)$$

În (2.3.2), partea recurentă a lui x este $\mathbf{x}(k) = \mathbf{F}^k\,\mathbf{x}(0)$, un şir vectorial exponenţial. Dacă se aplică ecuaţiilor (2.3.2), (2.3.3) rezultatele paragrafului următor şi proprietăţile (2.3.7) şi (2.3.8), se obţin :

$$z\big(\mathbf{X}(z) - \mathbf{x}(0)\big) = \mathbf{F}\,\mathbf{X}(z) + \mathbf{G}\,\mathbf{U}(z),$$

$$\mathbf{X}(z) = (z\mathbf{I} - \mathbf{F})^{-1}\big[\mathbf{G}\,\mathbf{U}(z) + z\mathbf{x}(0)\big], \text{ ecuaţia de stare.} \qquad (2.3.9)$$

$$\mathbf{Y}(z) = \mathbf{P}\,\mathbf{X}(z) + \mathbf{Q}\,\mathbf{U}(z), \text{ ecuaţia de ieşire.} \qquad (2.3.10)$$

Domeniul de convergenţă al lui z este intersecţia domeniilor de convergenţă ale transformatelor $\mathcal{Z}[\mathbf{F}\,\mathbf{x}]$, $\mathcal{Z}[\mathbf{P}\,\mathbf{x}]$, $\mathcal{Z}[\mathbf{u}]$, $\mathcal{Z}[\mathbf{G}\,\mathbf{u}]$, $\mathcal{Z}[\mathbf{Q}\,\mathbf{u}]$. Existenţa inversei $(z\mathbf{I} - \mathbf{F})^{-1}$ este asigurată prin condiţiile precedente, după cum se arată în paragraful următor.

2.3.3. Şirul exponenţial

Fie şirul vectorial exponenţial :

$\mathbf{s}(k) = \mathbf{A}^k\,\mathbf{v}$, cu vectorul oarecare $\mathbf{v} \in \mathbb{C}^m$, şi matricea pătrată oarecare

$$\mathbf{A} \in \mathbb{C}^{m \times m}. \qquad (2.3.11)$$

Atunci:

$$\mathcal{Z}[\mathbf{s}] = \mathbf{S}(z) = \lim_{K \to \infty} \sum_{k=0}^{K-1} z^{-k}\mathbf{A}^k\,\mathbf{v}. \qquad (2.3.12)$$

Avem că :

$$\left(\mathbf{I} - z^{-1}\mathbf{A}\right)\left(\sum_{k=0}^{K-1} z^{-k}\mathbf{A}^k\right)\mathbf{v} = \left(\mathbf{I} - z^{-K}\mathbf{A}^K\right)\mathbf{v}.$$

$\mathbf{S}(z)$ există dacă $z^{-K}\,\mathbf{A}^K$ converge şi dacă $z\mathbf{I}-\mathbf{A}$ este inversabilă :

$$S(z) = z(zI - A)^{-1}\left(I - \lim_{K\to\infty} z^{-K}A^K\right)v. \qquad (2.3.13)$$

Propoziţie : S(z) este convergent pentru orice v dacă şi numai dacă

$$|z| > \max_i |\lambda_i|, \qquad (2.3.14)$$

unde $\lambda_{i,A}$ sau λ_i, dacă nu este posibilă nici o confuzie, sunt valorile proprii ale matricii **A**.

Demonstraţie :

Se ştie că dacă matricea $A\in C^{mxm}$ are valorile proprii λ_i distincte (de multiplicitate 1), atunci există o matrice nesingulară $X\in C^{mxm}$ cu proprietatea $X^{-1}AX = \Lambda = diag(\lambda_1,...,\lambda_m)$ (*e.g.* Golub şi Van Loan, 1989). Puterea A^K este în acest caz

$$A^K = X\,\Lambda^K\,X^{-1} = X\,diag\left(\lambda_1^K,...,\lambda_m^K\right)X^{-1}, \qquad (2.3.15)$$

iar condiţia de convergenţă din (2.3.13) devine :

$$\lim_{K\to\infty} z^{-K}A^K v = X\,diag\left(\lim_{K\to\infty} z^{-K}\lambda_1^K,...,\lim_{K\to\infty} z^{-K}\lambda_m^K\right)X^{-1}v \qquad (2.3.16)$$

Convergenţa este deci obţinută dacă şi numai dacă

$$\lim_{K\to\infty} z^{-K}A^K v = 0,\ (\forall)v \Leftrightarrow \lim_{K\to\infty} z^{-K}\lambda_i^K = 0,\ i = \overline{1,m} \qquad (2.3.17)$$

$$\Leftrightarrow |z| > |\lambda_i|,\ i = \overline{1,m} \qquad (2.3.18)$$

Dacă matricea $A\in C^{mxm}$ are valori proprii multiple, acelaşi rezultat se poate obţine utilizând descompunerea Jordan pentru **A** şi calculând puterile blocurilor Jordan, sau utilizând caracterizarea Jordan a definiţiei generale a funcţiilor de matrice (Golub şi Van Loan, 1989), particularizate pentru funcţia putere.

În domeniul de convergenţă zI-A este inversabilă : $det(zI - A) = 0 \Rightarrow z \in \lambda(A)$, z este o valoare proprie a lui A; dar (2.3.18) nu permite acest lucru.

Transformata Z a lui (2.3.11) este deci

$$S(z) = z(zI - A)^{-1}v. \qquad (2.3.19)$$

Observaţii :

1. O condiţie mai slabă dar mai uşor de aplicat, pentru ca S(z) să fie convergentă pentru orice v este

$$\|z^{-1}A\|_p < 1,\quad |z| > \|A\|_p, \qquad (2.3.20)$$

unde $\|A\|_p$ este norma p a matricii **A** indusă de norma vectorială p (Anexa I). Aceasta este echivalentă cu a cere ca $v_k = \left\|\left(z^{-1}A\right)^k v\right\|_p$ să descrească la fiecare iteraţie,

$$\left\|\left(z^{-1}A\right)^k v\right\|_p \le \left\|\left(z^{-1}A\right)^{k-1} v\right\|_p,\ (\forall)\ k \ge 1. \qquad (2.3.21)$$

De obicei p este 1 (suma modulelor componentelor), 2 (norma euclidiană) sau ∞ (componenta maximă).

Demonstraţie :

Dacă $\left\|\left(z^{-1}A\right)\right\|_p = \alpha < 1$, atunci pentru orice vector v,

$$\left\|\left(z^{-1}\mathbf{A}\right)^{K}\mathbf{v}\right\|_p = \left\|\left(z^{-1}\mathbf{A}\right)\left(z^{-1}\mathbf{A}\right)^{K-1}\mathbf{v}\right\|_p \le \left\|\left(z^{-1}\mathbf{A}\right)\right\|_p \left\|\left(z^{-1}\mathbf{A}\right)^{K-1}\mathbf{v}\right\|_p =$$
$$= \alpha \left\|\left(z^{-1}\mathbf{A}\right)^{K-1}\mathbf{v}\right\|_p \le \ldots \le \alpha^K \|\mathbf{v}\|_p \xrightarrow{K\to\infty} 0 \qquad (2.3.22)$$

astfel încât $\mathbf{v}_k$ descreşte cu k şi $|z| > \|\mathbf{A}\|_p$ este suficientă pentru convergenţă.

Ca urmare a definiţiei normei matriciale p,

$$\|\mathbf{A}\|_p = \sup_{\mathbf{x}\neq 0} \frac{\|\mathbf{A}\mathbf{x}\|_p}{\|\mathbf{x}\|_p},$$

se poate obţine egalitate în (2.3.21) atunci când

$$\left\|\mathbf{A}^k\mathbf{v}\right\|_p = \|\mathbf{A}\|_p \left\|\mathbf{A}^{k-1}\mathbf{v}\right\|_p, \qquad (2.3.23)$$

astfel încât (2.3.20) şi (2.3.21) sunt echivalente.

Dacă p = 2 şi A este hermitică, atunci $\|\mathbf{A}\|_2 = \sqrt{|\lambda|_{max.\,\mathbf{A}^*\mathbf{A}}} = |\lambda|_{max.\,\mathbf{A}}$, iar cele două condiţii de convergenţă (2.3.14) şi (2.3.20) dau acelaşi rezultat.

2. Pentru p = 2, este interesant de observat că dacă **v** este specificat, $\left\|\left(z^{-1}\mathbf{A}\right)^k\mathbf{v}\right\|_2$ poate descreşte chiar pentru $|z| < \|\mathbf{A}\|_2$. De exemplu, dacă **v** are direcţia unui vector propriu $\mathbf{q}_j$ cc corespunde unei valori proprii $|\lambda_j|$ mai mică decât $\|\mathbf{A}\|_2$, atunci pentru $|\lambda_j| < |z| < \|\mathbf{A}\|_2$, $\left(z^{-1}\mathbf{A}\right)^K\mathbf{v}\|_2$ descreşte, iar (2.3.12) converge. Dacă totuşi **v** poate lua orice direcţie, sau dacă **v** are (chiar puţin) zgomot în jurul direcţiei alese a lui $\mathbf{q}_j$, atunci trebuie satisfăcută relaţia (2.3.20), cu p = 2.

2.3.4. Regimul tranzitoriu şi eroarea de regim staţionar pentru bucla de reacţie

Cu ajutorul transformatei Z vectoriale se obţin răspunsul tranzitoriu, eroarea de regim staţionar şi condiţiile de stabilitate pentru sistemul din figura 2.3.1. Transformata Z a ecuaţiei (2.3.4) este :

$$\mathbf{X}(z) = \mathbf{Y}(z) = (z\mathbf{I} - \mathbf{F})^{-1}\mathbf{G}\,\mathbf{U}(z) + z(z\mathbf{I} - \mathbf{F})^{-1}\mathbf{y}(0),$$
$$\mathbf{F} = \mathbf{I} - \mathbf{AB}, \quad \mathbf{G} = \mathbf{A} \qquad (2.3.24)$$

1. Fie

$$\mathbf{u}(k) = \delta(k)\cdot\mathbf{t}, \quad \delta(k) = \begin{cases} 0, & k \neq 0 \\ 1, & k = 0 \end{cases}, \text{ cu } \mathbf{t} \text{ - un vector ţintă oarecare.}$$

Atunci

$$\mathbf{U}(z) = \mathcal{Z}[\delta]\cdot\mathbf{t} = 1\cdot\mathbf{t}, \ (\forall)z.$$

Rezultă

$$\mathbf{Y}(z)_{\mathbf{u}(k)=\delta(k)\cdot\mathbf{t}} = (z\mathbf{I} - \mathbf{F})^{-1}\mathbf{G}\,\mathbf{t} + z(z\mathbf{I} - \mathbf{F})^{-1}\mathbf{y}(0). \qquad (2.3.25)$$

Ţinând seama de (2.3.8) şi identificând în (2.3.11) şi (2.3.19), se obţine răspunsul tranzitoriu al buclei de reacţie :

$$\mathbf{y}(k)_{\mathbf{u}(k)=\delta(k)\cdot\mathbf{t}} = \mathbf{F}^{k-1}\mathbf{G}\,\mathbf{t} + \mathbf{F}^{k}\,\mathbf{y}(0). \qquad (2.3.26)$$

Condiţia necesară şi suficientă de stabilitate a buclei de reacţie este :

$$\|\mathbf{y}(k)\|_2 < \infty,\ k \to \infty,\ (\forall)\,\mathbf{t}, \mathbf{y}(0) \ \Leftrightarrow\ \max_i |\lambda_{i,\mathbf{F}}| \le 1\ , \qquad (2.3.27)$$

Condiţia ca $\mathbf{x} = \mathbf{y} = \mathbf{0}$ să fie o stare global asimptotic stabilă este :

$$\|\mathbf{y}(k) - \mathbf{0}\|_2 \xrightarrow{k\to\infty} 0,\ (\forall)\,\mathbf{t}, \mathbf{y}(0) \ \Leftrightarrow\ \max_i |\lambda_{i,\mathbf{F}}| < 1\ , \qquad (2.3.28)$$

Afirmaţiile pot fi demonstrate calculând puterea matricii **F** din (2.3.25), cu ajutorul relaţiilor (2.3.15)-(2.3.18), în care se ia z=1. Rezultatul (2.3.28) este dat şi de (Goodwin şi Sin, 1984).

Se poate dori ca $\|\mathbf{y}(k)\|_p$ să descrească la fiecare iteraţie. Atunci, conform relaţiei (2.3.21) cu z = 1 :

$$\|\mathbf{y}(k+1)\|_p < \|\mathbf{y}(k)\|_p,\ (\forall)\,\mathbf{t}, \mathbf{y}(0) \ \Leftrightarrow\ \|\mathbf{F}\|_p < 1,\ \|\mathbf{I} - \mathbf{AB}\|_p < 1\ . \qquad (2.3.29)$$

Aceasta este o condiţie suficientă de stabilitate. Este mai restrictivă decât (2.3.25), dar mai uşor de aplicat. Demonstraţia ei este (2.3.22), luând z = 1.

2. Fie

$$\mathbf{u}(k) = \sigma(k)\cdot\mathbf{t},\ \ \sigma(k) = \begin{cases} 0, & k < 0 \\ 1, & k \ge 0 \end{cases},\ \text{cu } \mathbf{t} \text{ - un vector ţintă oarecare.}$$

Atunci

$$U(z) = \mathcal{Z}[\sigma]\cdot\mathbf{t} = \frac{z}{z-1}\mathbf{t}, \text{cu } |z| > 1.$$

Rezultă

$$\mathbf{Y}(z)_{\mathbf{u}(k)=\sigma(k)\cdot\mathbf{t}} = (z\mathbf{I} - \mathbf{F})^{-1}\mathbf{G}\,\frac{z}{z-1}\mathbf{t} + z(z\mathbf{I} - \mathbf{F})^{-1}\mathbf{y}(0). \qquad (2.3.30)$$

Poziţia atinsă asimptotic e dată de teorema valorii finale (Pop et al., 1989; Rivoire şi Ferrier, 1992) :

$$\mathbf{y}(\infty) = \lim_{z\to 1}(z-1)\mathbf{Y}(z) = (\mathbf{I} - \mathbf{F})^{-1}\mathbf{G}\,\mathbf{t} = (\mathbf{AB})^{-1}\mathbf{A}\,\mathbf{t}, \qquad (2.3.31)$$

dacă matricea **AB** este inversabilă. Dacă **A** şi **B** sunt inversabile, atunci :

$$\mathbf{y}(\infty) = \mathbf{B}^{-1}\mathbf{A}^{-1}\mathbf{A}\mathbf{t} = \mathbf{B}^{-1}\mathbf{t}. \qquad (2.3.32)$$

În sfârşit, dacă sistemul are reacţie unitară $\mathbf{B} = \mathbf{I}$, atunci eroarea de regim permanent este nulă:

$$\mathbf{y}(\infty) - \mathbf{t} = \mathbf{0}. \qquad (2.3.33)$$

Aceasta este versiunea discretă multidimensională a faptului binecunoscut în teoria sistemelor analogice, că un pol nul (o integrare) pe calea directă a unui sistem cu reacţie negativă are ca efect o eroare nulă în regim permanent.

Din condiţia necesară şi suficientă de convergenţă a transformatei Z (2.3.18), şi condiţia z=1 din (2.3.31), se obţine că condiţia $\max_i |\lambda_{i,\mathbf{F}}| < 1$ (2.3.28) este necesară şi suficientă pentru ca starea $\mathbf{B}^{-1}\mathbf{t}$ (2.3.32) să fie global asimptotic stabilă pentru orice **t** :

$$\|\mathbf{y}(k) - \mathbf{B}^{-1}\mathbf{t}\|_2 \xrightarrow{k\to\infty} 0,\ (\forall)\,\mathbf{t}, \mathbf{y}(0) \ \Leftrightarrow\ \max_i |\lambda_{i,\mathbf{F}}| < 1\ . \qquad (2.3.34)$$

Prin aceea că consideră intrarea nenulă, această relaţie generalizează rezultatul (2.3.28) şi pe cel dat de Goodwin şi Sin (1984).

2.3.5. Erorile maxime admisibile pentru bucla cu reacţie vizuală

Identificând componentele din figurile 2.2.7 şi 2.3.1, $\mathbf{A} = \mathbf{Jr}\,\tilde{\mathbf{J}}\mathbf{c}\,\mathbf{Jv}$, $\mathbf{B} = \mathbf{I}$ şi $\mathbf{F} = \mathbf{I} - \mathbf{Jr}\,\tilde{\mathbf{J}}\mathbf{c}\,\mathbf{Jv}$, iar din (2.3.34) se obţine condiţia necesară şi suficientă pentru stabilitatea buclei cu reacţie vizuală.

$$\max_i \lambda\left(\mathbf{I} - \mathbf{Jr}\,\tilde{\mathbf{J}}\mathbf{c}\,\mathbf{Jv}\right) < 1 \ . \tag{2.3.35}$$

Condiţia mai restrictivă, suficientă pentru stabilitate (2.3.29) devine :

$$\left\|\mathbf{I} - \mathbf{Jr}\,\tilde{\mathbf{J}}\mathbf{c}\,\mathbf{Jv}\right\|_2 < 1 \ . \tag{2.3.36}$$

Ea asigură ca după fiecare iteraţie norma erorii de poziţionare $\mathbf{m}(k+1)$ (2.3.6),

$$\left\|\mathbf{m}(k+1)\right\|_2 = \left\|\left(\mathbf{I} - \mathbf{Jr}\,\tilde{\mathbf{J}}\mathbf{c}\,\mathbf{Jv}\right)\,\mathbf{m}(k)\right\|_2 \le \left\|\mathbf{I} - \mathbf{Jr}\,\tilde{\mathbf{J}}\mathbf{c}\,\mathbf{Jv}\right\|_2 \left\|\mathbf{m}(k)\right\|_2 , \tag{2.3.37}$$

să scadă faţă de norma lui $\mathbf{m}(k)$.

$$\left\|\mathbf{I} - \mathbf{Jr}\,\tilde{\mathbf{J}}\mathbf{c}\,\mathbf{Jvv}\right\|_2 < 1 \Rightarrow \left\|\mathbf{m}(k+1)\right\|_2 < \left\|\mathbf{m}(k)\right\|_2 . \tag{2.3.38}$$

În (2.3.38) implicaţia are loc şi în sens contrar : în (2.3.37) egalitatea poate fi atinsă.

Cele două condiţii (2.3.35) şi (2.3.36) sunt echivalente dacă matricea $\mathbf{F}$ este simetrică. Dacă $\mathbf{F}$ are componente aleatoare, atunci acesta este şi cazul cel mai defavorabil, cum se va arăta şi în § 2.4.3.

Conform relaţiei (2.3.33), eroarea de regim permanent este nulă, chiar în prezenţa erorilor datorate faptului că comanda braţului de robot se face prin înmulţirea cu aproximarea $\tilde{\mathbf{J}}\mathbf{c}$, şi nu cu matricea fără erori $\mathbf{Jc}$.

2.4. Cerinţe de precizie pentru convergenţă la comanda braţului de robot [1]

În acest paragraf se studiază erorile admisibile ale părţii blocului de comandă ce furnizează doar $\mathbf{Jr}^{-1}$ din modelul considerat în § 2.2. Altfel spus, se consideră sistemul vizual cunoscut cu exactitate, astfel că prima componentă a blocului de comandă (cea neluată în considerare aici, dar care va face obiectul § 2.5), care furnizează inversa transformării vizuale $\mathbf{Jv}^{-}$ este şi ea disponibilă. Aşadar se presupun cunoscute poziţia ţintei, a braţului manipulator şi eroarea de poziţionare direct în spaţiul geometric. Modelul general din figura 2.2.6 devine cel din figura 2.4.1.

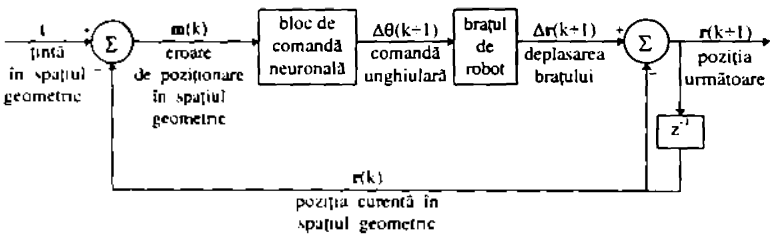


Figura 2.4.1. Modelul iterativ general cuprinzând doar braţul de robot

2.4.1. Modelul local liniar şi eroarea maximă pentru $\mathbf{Jr}^{-1}$

Liniarizând braţul ca în (2.4.1), se obţine modelul valabil pentru deplasări mici din figura 2.4.2. Pentru a se face distincţie faţă de cazul din §§ 2.2.3-2.2.4, în care s-a considerat şi transformarea vizuală, matricea de comandă ideală este notată $\mathbf{Jcr}$, iar aproximarea ei, furnizată de RN, $\tilde{\mathbf{J}}\mathbf{cr}$. Din figura 2.4.2 se obţin ecuaţiile (2.4.2).

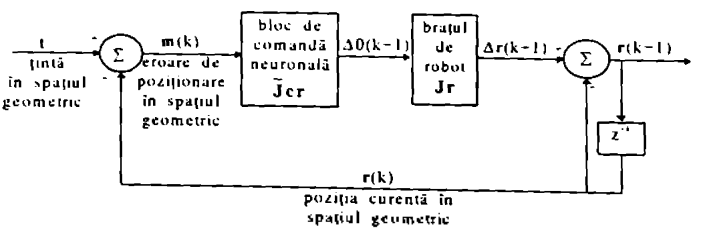


Figura 2.4.2. Modelul iterativ local liniar cuprinzând doar braţul de robot

$$\mathbf{r}(\theta + \Delta\theta) = \mathbf{r}(\theta) + \mathbf{Jr}\,\Delta\theta = \mathbf{r}(\theta) + \Delta\mathbf{r}$$
$$\Delta\theta = \tilde{\mathbf{J}}\mathbf{cr}\,\Delta\mathbf{v} \qquad (2.4.1)$$

[1] Într-o formă mai puţin elaborată, §§ 2.4.1-2.4.3 fac obiectul lucrării (Cimponeriu şi Gresser, 1995). Cu conţinutul actual, întregul subcapitol constituie lucrarea (Cimponeriu şi Gresser, 1996).

$$\begin{aligned}
&\mathbf{t} - \mathbf{r}(k) = \mathbf{m}(k)\\
&\Delta\theta(k+1) = \tilde{\mathbf{J}}\mathbf{cr}\,\mathbf{m}(k)\\
&\Delta\mathbf{r}(k+1) = \mathbf{Jr}\,\Delta\theta(k+1)\\
&\mathbf{r}(k+1) = \mathbf{r}(k) + \Delta\mathbf{r}(k+1)
\end{aligned} \tag{2.4.2}$$

Poziţia cleştelui manipulatorului evoluează conform (2.4.3), iar eroarea faţă de ţintă, ca (2.4.4).

$$\mathbf{r}(k+1) = \left(\mathbf{I} - \mathbf{Jr}\,\tilde{\mathbf{J}}\mathbf{cr}\right)\mathbf{r}(k) + \mathbf{Jr}\,\tilde{\mathbf{J}}\mathbf{cr}\,\mathbf{t} \tag{2.4.3}$$

$$\mathbf{m}(k+1) = \left(\mathbf{I} - \mathbf{Jr}\,\tilde{\mathbf{J}}\mathbf{cr}\right)\mathbf{m}(k) \tag{2.4.4}$$

Bucla de reacţie poate fi analizată cu ajutorul transformatei Z vectoriale (§ 2.3). S-a arătat că condiţia necesară şi suficientă de stabilitate este (2.4.5),

$$max_i\left|\lambda_{i,\mathbf{F}}\right| < 1, \tag{2.4.5}$$

unde $\mathbf{F} = \mathbf{I} - \mathbf{Jr}\,\tilde{\mathbf{J}}\mathbf{cr}$ eroarea faţă de matricea identitate a câştigului buclei, iar $\lambda_{i,\mathbf{F}}$ este a i-a valoare proprie a matricii $\mathbf{F}$. De asemenea, în condiţiile stabilităţii, eroarea de regim staţionar a buclei este nulă.

Comanda ideală care anulează eroarea de poziţionare într-un pas este (2.4.6),

$$\mathbf{Jcr} = \mathbf{Jr}^{-1}. \tag{2.4.6}$$

Însă ieşirile reţelei neuronale furnizează doar o aproximaţie a matricii $\mathbf{Jr}^{-1}$, $\tilde{\mathbf{J}}\mathbf{r}^{-1}$. Abaterile faţă de valorile exacte $\mathbf{Jr}^{-1}$ pot fi modelate printr-o eroare aditivă $\mathbf{Er}$:

$$\tilde{\mathbf{J}}\mathbf{cr} = \tilde{\mathbf{J}}\mathbf{r}^{-1} = \mathbf{Jcr} + \mathbf{Er} = \mathbf{Jr}^{-1} + \mathbf{Er}. \tag{2.4.7}$$

În acest caz evoluţia erorii de poziţionare este (2.4.8),

$$\mathbf{m}(k+1) = \left(\mathbf{I} - \mathbf{Jr}\left(\mathbf{Jr}^{-1} + \mathbf{Er}\right)\right)\mathbf{m}(k) = -\mathbf{Jr}\,\mathbf{Er}\,\mathbf{m}(k). \tag{2.4.8}$$

iar norma 2 (norma vectorială euclidiană) $\|\mathbf{m}\|_2$ se modifică conform (2.4.9), unde $\|\|_2$ este şi norma matricială 2 indusă de norma vectorială 2, şi unde au fost folosite proprietăţi ale normei matriciale (v. anexa I).

$$\begin{aligned}
\|\mathbf{m}(k+1)\|_2 &= \|\mathbf{Jr}\,\mathbf{Er}\,\mathbf{m}(k)\|_2 \le \|\mathbf{Jr}\,\mathbf{Er}\|_2 \|\mathbf{m}(k)\|_2\\
&\le \|\mathbf{Jr}\|_2 \|\mathbf{Er}\|_2 \|\mathbf{m}(k)\|_2 \le \|\mathbf{Jr}\|_2 \|\mathbf{Er}\|_F \|\mathbf{m}(k)\|_2
\end{aligned}. \tag{2.4.9}$$

Întrucât de obicei antrenarea RN este făcută prin minimizarea sumei pătratelor erorilor (*mean squared error*) mSE($\tilde{\mathbf{J}}\mathbf{r}^{-1}$) a celor $m \cdot n$ ieşiri ale matricii $\tilde{\mathbf{J}}\mathbf{r}^{-1\,m\times n}$, în (2.4.9) s-a introdus şi norma matricială euclidiană sau Frobenius (2.4.10),

$$\left\|\mathbf{Er}^{m\times n}\right\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\left|\mathrm{Er}_{ij}\right|^2} = \sqrt{mn\ \mathrm{mSE}\left(\tilde{\mathbf{J}}\mathbf{r}^{-1}\right)}. \tag{2.4.10}$$

Condiţia (2.4.5) garantează doar convergenţa asimptotică spre zero. Se doreşte însă ca eroarea de poziţionare să descrească la fiecare pas de cel puţin q_{max} ori.

$$q = \frac{\|\mathbf{m}(k+1)\|_2}{\|\mathbf{m}(k)\|_2} \le \|\mathbf{Jr}\|_2 \|\mathbf{Er}\|_F < q_{max} \le 1. \tag{2.4.11}$$

Astfel, dacă se impune q_{max} , rata minimă a descreşterii geometrice a mărimii erorii de poziţionare, se poate obţine eroarea medie pătratică a lui $\tilde{\mathbf{J}}\mathbf{r}^{-1}$:

$$\mathrm{mSE}\left(\tilde{\mathbf{J}}\mathbf{r}^{-1}\right) \le \frac{q_{\max}}{mn\left\|\mathbf{Jr}^{-1}\right\|_2} . \qquad (2.4.12)$$

Aici media este luată peste cele *mn* elemente ale matricii. Întrucât $\tilde{\mathbf{J}}\mathbf{r}^{-1}$ este o funcție de θ, pentru întregul domeniu de variație al lui θ se obține MSE, maximul al lui mSE :

$$\mathrm{MSE}\left(\tilde{\mathbf{J}}\mathbf{r}^{-1}\right) \le \min_{\theta} \frac{q_{\max}^{2}}{mn\left\|\mathbf{Jr}\right\|_2^{2}} . \qquad (2.4.13)$$

Acesta este un criteriu aplicabil practic pentru terminarea antrenării RN. Dacă, de exemplu, q_{max}=1/2, atunci mărimea erorii de poziționare va fi cel puțin înjumătățită la fiecare iterație.

Dar raportul normelor erorilor de poziționare q (2.4.11) va fi de fapt mai mic decât q_{max}:

$$\frac{\left\|\mathbf{m}(k+1)\right\|_2}{\left\|\mathbf{m}(k)\right\|_2} = q \le q_{\max} ,$$

iar convergența spre zero va fi mai rapidă. Pe de altă parte, dacă nu se poate obține egalitate în (2.4.14), atunci (2.4.13) este prea restrictivă, întrucât se dorește ca q, și nu q_{max}, să fie subunitar. Este deci interesant de văzut dacă și când se poate obține egalitate între q și q_{max} - datorită majorărilor succesive din (2.4.9).

În primul rând, $\left\|\mathbf{Jr}\,\mathbf{Er}\,\mathbf{m}(k)\right\|_2 = \left\|\mathbf{Jr}\,\mathbf{Er}\right\|_2 \left\|\mathbf{m}(k)\right\|_2$ se poate obține când $\mathbf{m}(k)$ are aceeași direcție cu cel mai mare vector propriu (cel corespunzând razei spectrale, cea mai mare valoare proprie) a matricii **Jr Er**. Aceasta e direcția pentru care convergența e cea mai lentă, dată de q_{max}.

În al doilea rând, $\left\|\mathbf{Jr}\,\mathbf{Er}\right\|_2 \le \left\|\mathbf{Jr}\right\|_2 \left\|\mathbf{Er}\right\|_2$ poate da egalitate când cel mai mare vector propriu (la stânga) al matricii **Er** este coliniar cu cel mai mare vector propriu (la dreapta) al lui **Jr**. Datorită caracterului presupus aleator al elementelor lui **Er**, această inegalitate (ca și următoarea) poate fi văzută ca S=ρD, unde $\rho = \rho(\theta)$ este o variabilă aleatoare luând valori între 0 și 1. Pentru a avea o idee despre distribuția lui ρ, figura 2.4.3 prezintă rezultatele unei simulări Monte Carlo.

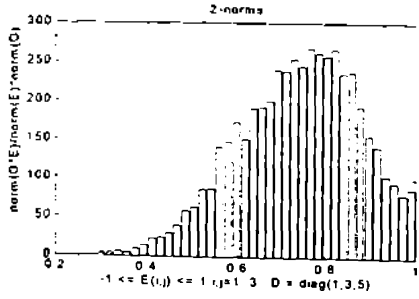


Figura 2.4.3. Histograma pentru $\dfrac{\left\|\mathbf{DE}\right\|_2}{\left\|\mathbf{D}\right\|_2\left\|\mathbf{E}\right\|_2}$ într-o simulare cu 5000 de încercări cu o matrice aleatoare $\mathbf{E}^{3\times3}$, $-1 \le e_{ij} \le 1$ și $\mathbf{D} = diag(1.3.5)$, $\left\|\mathbf{D}\right\|_2 = 5$.

Dacă valorile proprii ale lui **Jr** ar fi cele ale matricii **D** din simulare, aceasta l-ar da pe ρ. Valoarea sa maximă ρ_{max} este 1. Valoarea sa medie ρ_{med} este aproximativ 0.74. Valoarea

$\frac{\rho_{med}}{\rho_{max}} \cong 0{,}74$ arată că în medie valoarea lui q este de 0,74 de ori mai mică decât q_{max}. În medie, q_{max} este de 0,74 de ori mai mic decât ar fi dacă această inegalitate nu ar fi prezentă.

În al treilea rând, pentru $\|\mathbf{Er}\|_2 \le \|\mathbf{Er}\|_F$ se poate obţine egalitate când elementele lui **Er** sunt egale. Figura 2.4.4 arată rezultatele unei simulări Monte Carlo. Explicaţia este aceeaşi ca pentru figura precedentă. Valoarea medie este aproximativ 0,84, dar 1 (egalitate) se obţine mai puţin frecvent ca în cazul precedent.

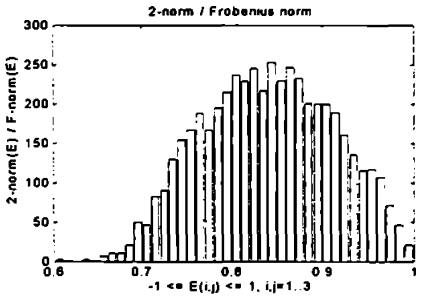


Figura 2.4.4. Histograma pentru $\frac{\|\mathbf{E}\|_2}{\|\mathbf{E}\|_F}$ într-o simulare cu 5000 de încercări cu o matrice aleatoare $\mathbf{E}^{3x3}$, $-1 \le e_{ij} \le 1$.

În fine, relaţia (2.4.13) mai face o majorare, care dacă nu este respectată, pot exista locuri θ în care algoritmul să diveargă. Majorarea depinde de funcţia $\|\mathbf{Jr}(\theta)\|_2$, dependentă de aplicaţie. Graficul ei va fi prezentat în § 2.4.3.

O ultimă observaţie pentru acest paragraf : condiţia (2.4.11), ca mărimea erorii de poziţionare să scadă la fiecare iteraţie, este o condiţie suficientă de stabilitate pentru bucla de reacţie. Ea este mai restrictivă decât condiţia necesară şi suficientă de stabilitate (2.4.5). Totuşi, dacă matricea erorii **Er** este simetrică, rezultatele condiţiilor (2.4.5) şi (2.4.11) sunt identice. Figura 2.4.5 prezintă rezultatul unei simulări Monte Carlo pentru raportul $\frac{\|\mathbf{E}\|_2}{max|\lambda(\mathbf{E})|}$.

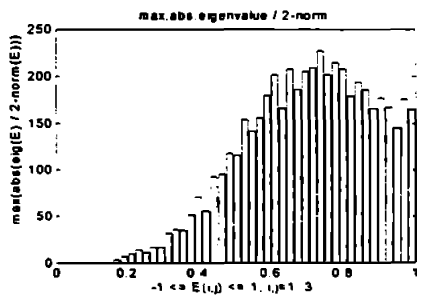


Figura 2.4.5. Histograma pentru $\frac{\|\mathbf{E}\|_2}{max|\lambda(\mathbf{E})|}$ într-o simulare cu 5000 de încercări cu o matrice aleatoare $\mathbf{E}^{3x3}$, $-1 \le e_{ij} \le 1$.

Valoarea medie a raportului este aproximativ 0,70. Aceasta înseamnă că în medie q_{max} ar putea fi de 1/0.70 de ori mai mare decât 1, iar stabilitatea încă să fie obţinută. Dar uneori $q_{max}=1$ este cea mai mare valoare posibilă pentru care, pentru o anumită direcţie a lui **m** şi pentru o anumită **Er**, bucla nu diverge.

2.4.2. Modelul local pătratic: eroarea maximă de poziţionare

Modelul local liniar este valabil doar dacă deplasarea cleştelui manipulatorului, respectiv eroarea de poziţionare, nu este mare; în caz contrar apar erori suplimentare de neliniaritate. Un model pătratic al braţului, ca o aproximare mai precisă a transformării cinematice neliniare, poate da o limită a mărimii erorii de poziţionare care încă garantează convergenţa algoritmului iterativ.

În locul modelului liniar (2.4.1), se consideră că fiecare componentă x_i a vectorului de poziţie **r** este dată de seria Taylor de ordinul 2:

$$x_i(\theta+\Delta\theta) \cong x_i(\theta) + \nabla_i^T \Delta\theta + \frac{1}{2}\Delta\theta^T \mathbf{H}_i \Delta\theta , \qquad (2.4.14)$$

unde ∇_i este gradientul lui x_i sau a i-a linie a matricii jacobiene **Jr**, $\nabla_i^T = \mathbf{Jr}(i,:)$, iar $\mathbf{Hr}_i$ este matricea hessiană a lui $x_i(\theta)$, $\mathbf{Hr}_i = \left[hr^i_{jk}\right] = \left[\partial^2 x_i / \partial\theta_j \partial\theta_k\right]$. Pentru vectorul de poziţie $\mathbf{r} = \left[\cdots \quad x_i \quad \cdots\right]^T$, (2.4.14) devine (2.4.15),

$$\mathbf{r}(\theta+\Delta\theta) - \mathbf{r}(\theta) = \Delta r \cong$$

$$\cong \mathbf{Jr}\,\Delta\theta + \frac{1}{2}\begin{pmatrix} \vdots \\ \Delta\theta^T \mathbf{Hr}_i \\ \vdots \end{pmatrix}\Delta\theta = \Delta\mathbf{r}_l + \Delta\mathbf{r}_q , \qquad (2.4.15)$$

unde $\Delta\mathbf{r}_l$ este partea liniară a modelului, iar $\Delta\mathbf{r}_q$ este cea pătratică. În consecinţă, evoluţia erorii de poziţionare (2.4.4) devine (2.4.16),

$$\mathbf{m}(k+1) = \left\{ I - \left[\mathbf{Jr} + \frac{1}{2}\begin{pmatrix} \vdots \\ \Delta\theta^T \mathbf{Hr}_i \\ \vdots \end{pmatrix} \right] \widetilde{\mathbf{Jr}}^{-1} \right\} \mathbf{m}(k) = \mathbf{m}_l(k+1) + \mathbf{m}_q(k+1) \qquad (2.4.16)$$

unde partea liniară a erorii $\mathbf{m}_l(k+1)$ a fost studiată în paragraful precedent. Termenul $\mathbf{m}_q(k+1)$ este partea pătratică a erorii. Cele două componente satisfac inegalitatea triunghiului (2.4.17) :

$$\|\mathbf{m}\| \le \|\mathbf{m}_l\| + \|\mathbf{m}_q\| . \qquad (2.4.17)$$

În continuare, pentru studiul componentei pătratice se consideră $\widetilde{\mathbf{Jr}}^{-1} \cong \mathbf{Jr}^{-1}$. $\Delta\theta$ poate reprezenta variaţia unghiulară care îl dă pe Δr, dar poate fi şi diferenţa între poziţia curentă position θ şi o poziţie vecină θ_0 pentru care este cunoscută valoarea lui $\mathbf{Jr}^{-1}$, ca în cazul în care se utilizează pentru comandă reţele Kohonen extinse. Se obţin ecuaţiile (2.4.18) şi (2.4.19) :

$$\|\mathbf{m}_q(k+1)\|_2 = \frac{1}{2}\left\| \begin{pmatrix} \vdots \\ \Delta\theta^T \mathbf{Hr}_i \mathbf{Jr}^{-1} \\ \vdots \end{pmatrix} \mathbf{m}(k+1) \right\|_2$$

$$\le \frac{1}{2}\left\| \begin{pmatrix} \vdots \\ \Delta\theta^T \mathbf{Hr}_i \mathbf{Jr}^{-1} \\ \vdots \end{pmatrix} \right\|_2 \|\mathbf{m}(k+1)\|_2 \qquad (2.4.18)$$

$$\le \frac{1}{2}\left\| \begin{pmatrix} \vdots \\ \Delta\theta^T \mathbf{Hr}_i \mathbf{Jr}^{-1} \\ \vdots \end{pmatrix} \right\|_F \|\mathbf{m}(k+1)\|_2$$

$$\left\| \begin{pmatrix} \vdots \\ \Delta\theta^T \mathbf{Hr}_i \mathbf{J}^{-1} \\ \vdots \end{pmatrix} \right\|_F = \sqrt{\Delta\theta^T \mathbf{Mr}\, \Delta\theta} \le \|\Delta\theta\| \sqrt{\|\mathbf{Mr}\|_2}\,, \qquad (2.4.19)$$

$$\mathbf{Mr} = \sum_i \left(\mathbf{Mr}_i \mathbf{Mr}_i^T \right), \quad \mathbf{Mr}_i = \mathbf{Hr}_i \mathbf{Jr}^{-1}$$

Din nou, în (2.4.18) există o direcţie cea mai puţin favorabilă pentru $\mathbf{m}(k)$, şi o creştere a normei F faţă de norma l_2, dependentă de aplicaţie. În (2.4.19) există o direcţie cea mai puţin favorabilă pentru $\Delta\theta$.

Dacă se caută eroarea m în cooordonate carteziene în locul erorii unghiulare $\Delta\theta = \mathbf{Jr}^{-1}\mathbf{m}$, atunci (2.4.19) devine (2.4.20).

$$\left\| \begin{pmatrix} \vdots \\ \Delta\theta^T \mathbf{Hr}_i \mathbf{J}^{-1} \\ \vdots \end{pmatrix} \right\|_F = \sqrt{\mathbf{m}^T \mathbf{Mr}'\, \mathbf{m}} \le \|\mathbf{m}\| \sqrt{\|\mathbf{Mr}'\|_2}\,, \qquad (2.4.20)$$

$$\mathbf{Mr}' = \sum_i \left(\mathbf{Mr}'_i \mathbf{Mr}'^T_i \right), \quad \mathbf{Mr}'_i = \mathbf{Jr}^{T^{-1}} \mathbf{Hr}_i \mathbf{Jr}^{-1}$$

Dacă în condiţia de convergenţă (2.4.12) $\mathbf{m}(k+1)$ este înlocuit cu $\mathbf{m}_l(k+1)$ şi $q_{max,l}$ mărgineşte partea liniară a erorii, se poate impune o valoare $q_{max,q}$ pentru raportul q_q de scădere a componentei pătratice :

$$q_q = \frac{\|\mathbf{m}_q(k+1)\|_2}{\|\mathbf{m}(k)\|_2} \le \frac{1}{2}\|\Delta\theta\| \sqrt{\|\mathbf{Mr}'\|_2} < q_{max,q} \le 1 - q_{max,l}. \qquad (2.4.21)$$

$$q_q = \frac{\|\mathbf{m}_q(k+1)\|_2}{\|\mathbf{m}(k)\|_2} \le \frac{1}{2}\|\mathbf{m}\| \sqrt{\|\mathbf{Mr}'\|_2} < q_{max,q} \le 1 - q_{max,l}. \qquad (2.4.21')$$

Pentru eroarea totală (2.4.16), conform (2.4.17), rata minimă a descreşterii erorii este $q_{max,l}+q_{max,q}$. Aici există din nou un caz cel mai defavorabil, anume cel când $\mathbf{m}_l(k+1)$ şi $\mathbf{m}_q(k+1)$ sunt coliniare; altfel, mărimea erorii de poziţionare scade mai repede.

Este interesant de observat că, potrivit modelului local pătratic, valoarea maximă a raportului q_q (2.4.21), (2.4.21') creşte liniar cu $\|\Delta\theta\|$ sau cu $\|\mathbf{m}\|$.

Matricile ce apar în (2.4.20) sunt funcţii de θ. Deci mărimea maximă a lui Δθ sau m ce se obţine este o funcţie de θ,

$$\|\Delta\theta\| \le \|\Delta\theta\|_{\max,\theta} = \frac{2q_{\max,q}}{\sqrt{\|\mathbf{M}^{-1}\|_{12}}}, \tag{2.4.22}$$

$$\|\mathbf{m}\| \le \|\mathbf{m}\|_{\max,\theta} = \frac{2q_{\max,q}}{\sqrt{\|\mathbf{M}^{-1}\|_{12}}}. \tag{2.4.22'}$$

Pentru asigurarea convergenței pe întregul domeniu al lui θ este necesar ca

$$\|\Delta\theta\| \le \|\Delta\theta\|_{\max} = \min_{\theta} \|\Delta\theta\|_{\max,\theta}, \tag{2.4.23}$$

$$\|\mathbf{m}\| \le \|\mathbf{m}\|_{\max} = \min_{\theta} \|\mathbf{m}\|_{\max,\theta}. \tag{2.4.23'}$$

Dacă pentru comanda neuronală se preferă o soluție ce utilizează valori cuantizate pentru θ, atunci $\|\Delta\theta\|_{\max}$ limitează suma dintre deplasarea unghiulară și eroarea de cuantizare. Cazul cel mai defavorabil este când cele două sunt coliniare. De exemplu, dacă se alege ca amândouă să aibe o valoare maximă de $\frac{1}{2}\|\Delta\theta\|_{\max}$, atunci convergența este asigurată. O altă posibilitate este de a se considera că se fac pași foarte mici la fiecare iterație; în acest caz eroarea maximă de cuantizare poate practic egala $\|\Delta\theta\|_{\max}$.

2.4.3. Rezultate pentru brațul ERICC și rețele Backpropagation

Pentru brațul manipulator ERICC prezentat în § 2.1.1, calculele presupuse de secțiunile precedente pot fi făcute fără dificultate. Tabelul 2.4.1 prezintă eroarea pătratică maximă $\mathrm{MSE}\left(\tilde{J}r^{-1}\right)$ (2.4.13) și lungimea maximă a erorii de poziționare $\|\Delta\theta\|_{\max}$ (2.4.23), $\|\mathbf{m}\|_{\max}$ (2.4.23') ce garantează convergența pentru θ aparținând domeniului din (2.1.1), cu restricțiile suplimentare $-45° \le \theta_3 \le 60°$ și $d \ge 200$ mm. Pentru $\|\mathbf{m}\|_{\max}$ valoarea de 0,064 se obține prin normalizarea lui 48 mm cu $L_2+L_3=747$ mm (2.1.1), pentru motive ce se vor vedea mai jos. Dacă θ_3 ia valori vecine lui 90° sau dacă cleștele se apropie de axul vertical, transformarea jacobiană inversă devine mai neliniară și se obțin valori mai mici pentru lungimea maximă a erorii.

MSE [rad²]		$\|\Delta\theta\|_{\max}$ [°]	$\|\mathbf{m}\|_{\max}$	
$q_{\max,l}=1$	$q_{\max,l}=1/2$	$q_{\max,q}=1/2$	$q_{\max,q}=1/2$	
0.084	0.021	14.7	48 [mm]	0.064

Tabelul 2.4.1. Erorile maxime ce garantează convergența pentru $-45° \le \theta_3 \le 60°$ și $d \ge 200$ mm.

Dacă sunt luate în considerare și erorile modelului liniar și ale celui pătratic și $q_{\max,l}=1/2$, atunci $\mathrm{MSE}\left(\tilde{J}r^{-1}\right)=0.021$. Utilitatea lui poate fi constatată în figura 2.4.9. Dacă deplasarea la fiecare iterație e foarte mică și nu există eroare de cuantizare pentru θ, $q_{\max,l}$ poate fi crescut la 1 și MSE poate fi 0.084.

Graficul erorii pătratice admisibile $\mathrm{mSE}\left(\tilde{J}r^{-1}(\theta)\right)$ ce rezultă din (2.4.11) pentru $q_{\max,l}=1$, $\theta_1 = 0$, $-45° \le \theta_2 \le 90°$ și $-45° \le \theta_3 \le 75°$ este prezentat în figura 2.4.6.

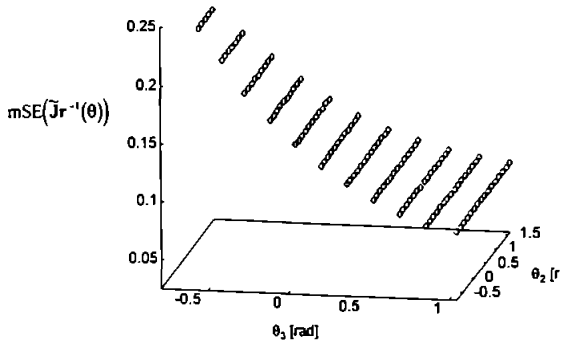


Figura 2.4.6. $\mathrm{mSE}\left(\tilde{\mathbf{J}}\mathbf{r}^{-1}(\theta)\right)$ în planul $\theta_1 = 0$

În tabelul 2.4.1. mărimea vectorului variație unghiulară pentru care este valabilă aproximația liniară. este $\|\Delta\theta\|_{max}$ =14,7°, adică circa 8,5° pentru fiecare din componentele care în cel mai defavorabil caz sunt egale. Dacă limitele domeniului lui (θ_1, θ_2) sunt restrânse la o zonă mai centrală. se obțin valori mai mari. Figurile 2.4.7 și 2.4.8 prezintă valorile $\|\Delta\theta\|_{max}$ [°] (2.4.23) și respectiv ale $\|\mathbf{m}\|_{max}$ (2.4.23') normalizate. în funcție de θ. pe domeniul $\theta_1 = 0$. $-45° \le \theta_2 \le 90°$ și $-45° \le \theta_3 \le 75°$, pentru $q_{max,q} \approx 1/2$. În centru $\|\Delta\theta\|_{max}$ poate depăși 30°. iar $\|\mathbf{m}\|_{max}$ poate depăși 0,3 adică 22 cm.

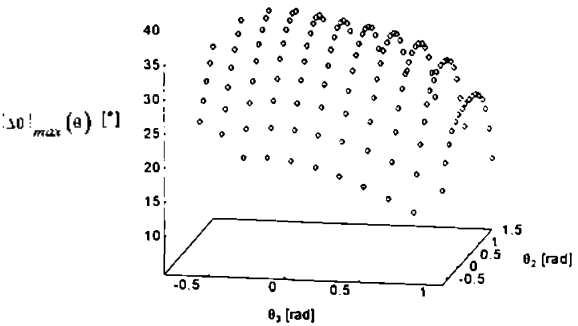


Figura 2.4.7. $\|\Delta\theta\|_{max}(\theta)$ în planul $\theta_1 = 0$.

De asemenea. au fost antrenate două rețele Backpropagation. utilizând compilatorul de rețele neuronale Aspirin/MIGRAINES 6, disponibil gratuit pe Internet[2]. Prima din rețele corespunde lui RN_1 din figura 2.2.2, care, în condițiile acestui subcapitol, are de aproximat transformarea cinematică inversă a manipulatorului. Cea de-a doua corespunde lui RN_2 din figura 2.2.4. și ea învață inversa matricii jacobiene 3x3 a manipulatorului. considerată ca o funcție cu 9 ieșiri. Urmând recomandările din (Gorinevsky și Konnoly, 1994). rețelele au avut

[2] și scris pentru sistemul de operare UNIX (Leighton. 1992). Tot disponibil gratuit pe Internet este și programul Gnuplot (Williams }i Kelley. 1993). executabil sub UNIX și sub DOS sau Windows. cu care au fost trasate graficele experimentale prezentate în continuare. pornind de la fișiere de date ASCII.

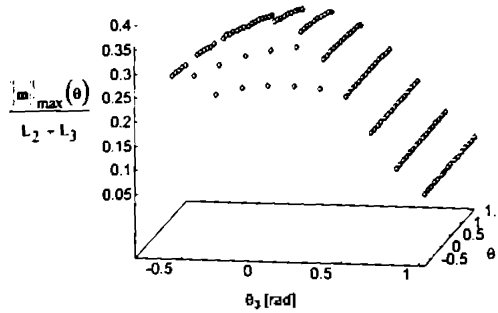


Figura 2.4.8. $\|\mathbf{m}\|_{max}(\theta)$ normalizat cu L_2+L_3, în planul $\theta_1 = 0$.

două straturi, deşi restricţiile din (2.1.1) fac atât transformarea cinematică cât şi jacobiana sa inversabile. şi un strat ar fi în principiu suficient (Hornik et al., 1989; Sontag, 1990). În (2.2.1), pentru x_2 , L_1 a fost luat 0 şi, pentru ca datele să fie de ordinul de mărime [-1, 1], toţi x_i au fost normalizaţi cu (L_2+L_3)=747 mm. Domeniile unghiurilor au fost $-60° \le \theta_1 \le 60°$, $-45° \le \theta_2 \le 90°$, $-45° \le \theta_3 \le 60°$. Antrenarea reţelelor a fost *Backpropagation* simplă cu inerţie. Figurile 2.4.9 şi 2.4.10 prezintă graficele antrenării pentru reţele cu 3-10-10-3, respectiv 3-25-25-9 neuroni.

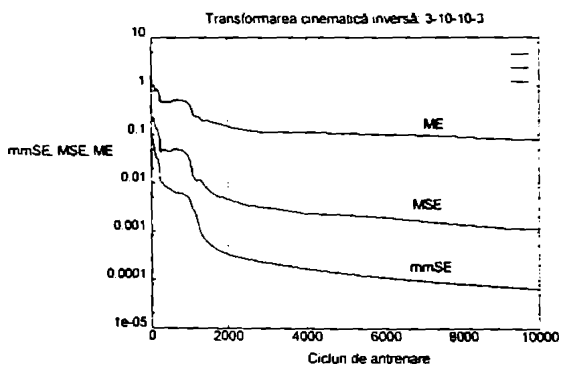


Figura 2.4.9. Graficul antrenării unei reţele BP 3-10-10-3 pentru aproximarea transformării cinematice inverse. Este reprezentată evoluţia erorii pătratice medii medii mmSE, a erorii pătratice medii maxime MSE şi a erorii maxime ME. Erorile pătratice sunt reprezentate înjumătăţite.

În ambele figuri există diferenţe de circa un ordin de mărime între mmSE, eroarea pătratică medie (peste ieşirile reţelei) medie (peste setul de antrenare), şi MSE, eroarea pătratică medie (peste ieşirile reţelei) maximă (peste setul de antrenare). De exemplu, în figura 2.4.8 după 10.000 de cicluri valorile lor sunt $6 \cdot 10^{-4}$ respectiv $6,7 \cdot 10^{-3}$. Prima este cea minimizată pe parcursul antrenării, în timp ce cea de-a doua este cea care indică momentul când antrenarea poate fi oprită. ME, eroarea maximă l_∞ în raport cu toate ieşirile şi întreg setul de antrenare, poate părea mare comparată cu MSE, dar diferenţa se explică prin aceea că ultima este ridicată

la pătrat şi mediată adică, pentru RN_2, divizată cu 9. În plus, programul de simulare dă erorile pătratice înjumătăţite.

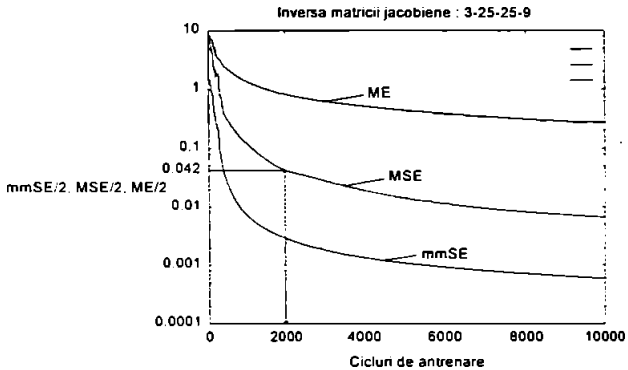


Figura 2.4.10. Graficul antrenării unei reţele BP 3-25-25-9 pentru aproximarea inversei matricii jacobiene. Erorile pătratice sunt reprezentate înjumătăţite.

Pentru RN_1, conform tabelului 2.4.1, eroarea maximă de poziţionare este de 14,7° ~ 0.256 rad. ~ $6.6 \cdot 10^{-2}$ rad^2. Această valoare se împarte la 3, numărul ieşirilor, pentru a se obţine eroarea pătratică medie, şi la 2, pentru că pe grafice erorile pătratice sunt înjumătăţite; şi se obţine circa $1.1 \cdot 10^{-2}$. Pe figura 2.4.9, conform curbei MSE antrenarea poate fi oprită după circa 1300 de cicluri. Pentru RN_2, cu $q_{max,i}$=1, MSE/2=0.084/2=0.042 : în figura 2.4.10 antrenarea poate fi oprită puţin înainte de 2000 de cicluri. Garantarea convergenţei pentru $q_{max,i}$=1/2 necesită circa 6500 de cicluri de antrenare.

În figura 2.4.11 se poate vedea cum aproximează reţeaua antrenată RN_2 inversa matricii jacobiene în planul θ_1=0.

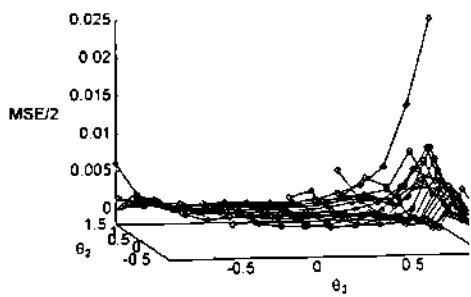


Figura 2.4.11. Distribuţia erorilor de aproximare după antrenarea reţelei NN_2, în planul θ_1=0.

Eroarea pătratică maximă (înjumătăţită) în acest plan este de 0,0244, şi se obţine la marginea domeniului de antrenare, pentru $\theta_2=\pi/2$ şi $\theta_3=\pi/3$. Eroarea pătratică medie este sensibil mai mică; asigurarea convergenţei în tot domeniul este mult mai restrictivă decât asigurarea convergenţei în medie, care se obţine dacă oprirea antrenării se face conform unui prag impus mediei pe setul de antrenare a erorii pătratice a ieşirilor reţelei.

2.4.4. Rezultate de simulare

Întrucât la sfârşitul efectuării acestui studiu nu era încă disponibilă legătura dintre calculatorul pe care poate fi rulat compilatorul de reţele neuronale Aspirin şi braţul ERICC, pentru validarea modelelor şi calculelor prezentate a fost aleasă simularea. Aceasta a fost făcută înlocuind robotul prin funcţiile matematice ce îl descriu, (2.1.1).

A fost realizat algoritmul corespunzător comenzii cinematice iterative local liniare corespunzător figurilor 2.2.3 şi 2.2.4, care este mai general decât algoritmul corespunzând figurilor 2.2.2 şi 2.2.3. Rezultatul unei simulări este prezentat în figura 2.4.12 a-c.

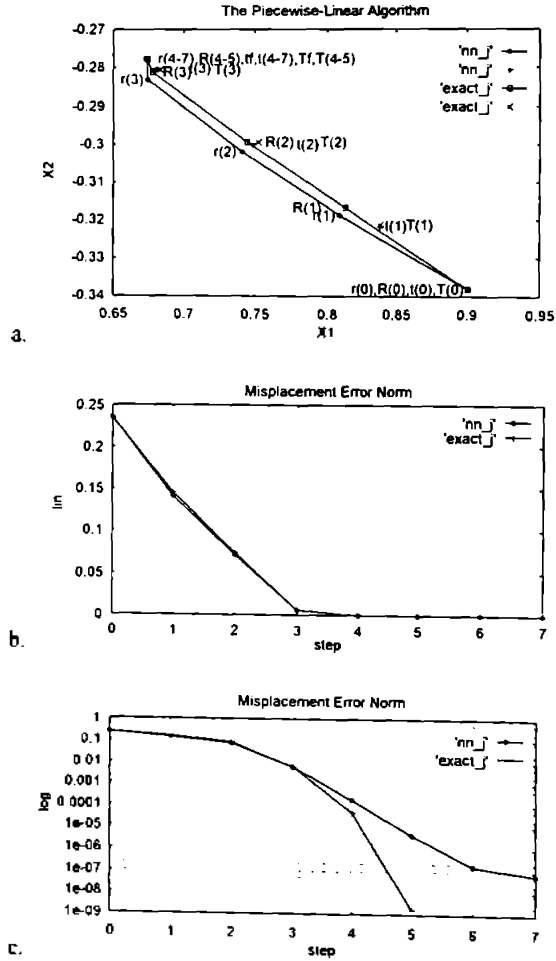


Figura 2.4.12. Simularea algoritmului de comandă cinematică iterativă local liniară, pentru o comandă exactă (curbele "exact_j") şi obţinută prin RN (curbele "nn_j") : a) deplasarea spre ţintă: b. c) Variaţia mărimii erorii de poziţionare : b) scară liniară, c) scară logaritmică

În figura 2.4.12 a) poziţia iniţială este **r**(0), dată de θ(0)=(0, 0, 1), iar ţinta finală este **tf**, θf=(0, 0,64, 0), valori alese pentru ca erorile să fie observabile. Sunt prezentate două cazuri, cel al comenzii date de RN (curbele notate "nn_j"), şi cel al comenzii ideale, obţinute prin inversarea matricii jacobiene a robotului descris de (2.1.1) (curbele "exact_j"). Programatorul traiectoriei stabileşte ţinta temporară **t**(1), iar cleştele robotului, comandat de RN, este deplasat în **r**(1). Distanţa dintre poziţia curentă şi ţinta temporară următoare este impusă, $\|\mathbf{m}\|_{max}$=0,064, valoare luată din tabelul 2.4.1. Dacă comanda este exactă, ţinta temporară este **T**(1)=**t**(1), iar robotul se deplasează în **R**(1). Se observă că **r**(1)≠**R**(1)≠**T**(1). **R**(1) diferă de **T**(1) datorită erorii aproximării liniare, iar **r**(1) diferă de **R**(1) datorită erorilor suplimentare ale RN.

În figura 2.4.12 b se vede că la primele 3 iteraţii distanţa de la cleşte la ţintă scade aproximativ liniar, cu $\|\mathbf{t}(k)-\mathbf{r}(k)\| \cong \|\mathbf{m}\|_{max}$ la fiecare pas. Cele două curbe nu diferă mult între ele. În absenţa erorilor, ambele ar fi drepte, şi la fiecare pas eroarea ar scădea cu $\|\mathbf{m}\|_{max}$. În continuare, pentru k≥4, distanţa rămasă este mai mică decât $\|\mathbf{m}\|_{max}$, şi ţintele **t**(k), respectiv **T**(k) coincid cu ţinta finală.

Se pot trage concluzii suplimentare reprezentând erorile pe scară logaritmică, ca în figura 2.4.12 c; valoarea foarte mică a erorii finale a fost aleasă tocmai pentru ca să poată fi făcute observaţiile care urmează. Pentru curba "nn_j" panta este aproximativ constantă, descreşterea exponenţială capătând un aspect liniar pe scara logaritmică. Aceasta înseamnă că la fiecare pas eroarea rămasă scade constant, de circa q ori. La iteraţia a 7-a, q ia o valoare mai mică, prin schimbarea direcţiei de deplasare a braţului. Însă pe curba "exact_j" eroarea scade din ce mai repede. Explicaţia este că, pe măsură ce eroarea este mai mică, modelul local liniar este mai precis. Dacă nu ar exista erori, ţinta ar fi atinsă într-un pas, corespunzând, în coordonate logaritmice, unei pante verticale a segmentului de dreaptă. Aşadar, abaterea de la verticală a pantelor segmentelor de dreaptă ale curbei "exact_j" reprezintă erorile de modelare local liniară, iar pantele segmentelor curbei "nn_j" reprezintă erorile de aproximare ale RN.

În continuare s-a urmărit variaţia distanţei faţă de o ţintă (temporară) aflată la distanţa $\|\mathbf{m}(0)\|$, la efectuarea unui pas. Figurile 2.4.13 a şi b prezintă mărimea erorii de poziţionare $\|\mathbf{m}(k)\|$, k=0.1 înainte şi după efectuarea pasului, pentru mai multe valori ale erorii iniţiale $\|\mathbf{m}(0)\|$ şi câte 100 de direcţii aleatoare ale ţintei. Valorile $\|\mathbf{m}\|$ sunt normalizate cu L_2+L_3=747 mm; de exemplu, $\|\mathbf{m}\|$=0,03 corespunde unei distanţe de 0.03·747=22 mm. Ca în figura precedentă, pantele q ale segmentelor de dreaptă nu sunt infinite, pe scara logaritmică. Aceasta înseamnă că există erori. Ele depind de poziţie şi de mărimea şi direcţia deplasării. La deplasări mici, $\|\mathbf{m}(0)\|$=0,003, pentru θ(0) = (0, 0, 0) $q \in [0{,}005, 0{,}05]$, erorile sunt mai mici decât pentru θ(0) = (0, π/2, π/3), unde $q \in [0{,}013; 0{,}2]$. Aceasta corespunde unei aproximări mai bune de către RN a jacobienei în primul punct, situat în centrul domeniului pentru care s-a făcut antrenarea, faţă de al doilea, ales ca fiind punctul de eroare maximă a reţelei în planul θ_1=0, conform figurii 2.4.11. Faptul este calitativ în concordanţă cu rezultatele teoretice. De asemenea, conform consideraţiilor din § 2.4.1, dependenţa în raport cu direcţia de deplasare era previzibilă. Însă valorile obţinute în § 2.4.1 sunt pesimiste: reţeaua a fost antrenată, pentru $q_{max.\ impus}$=1. până la MSE=0,084, eroare ce se constată că se face pentru θ = (π/3, π/2, π/3). O simulare de acelaşi gen cu cele din figura 2.4.13 dă la deplasări mici $q \le q_{max.\ real}$ = 0,2, valoare

inferioară lui $q_{max,\ impus}$. Prin $q_{max,\ real}$ s-a notat valoarea maximă a lui q ce rezultă din

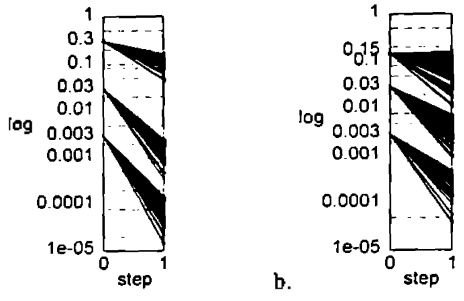


Figura 2.4.13. Variația erorii de poziționare $\|\mathbf{m}(k)\|$ pentru un pas, k = 0 și 1, pornind din a) $\theta(0)=(0, 0, 0)$, b) $\theta(0)=(0, \pi/2, \pi/3)$.

simulare, iar prin $q_{max,\ impus}$, valoarea impusă, corespunzând lui q_{max} (2.4.11)-(2.4.12). Cazul în care în toate majorările detaliate în § 2.4.1 sunt prezente simultan direcțiile cele mai defavorabile, nu apare în această simulare, obținută pentru o anumită distribuție a elementelor matricii erorii de aproximare a RN.

Pentru a se constata dacă rezultatul din § 2.4.1 este într-adevăr prea restrictiv, a fost realizată o altă simulare, în care erorile matricii J^{-1} au valori aleatoare, dar satisfăcând condiția mSE=0.084. Rezultatul este prezentat în figura 2.4.14. Media lui $q_{max,\ real}$ este

$$q_{max,\ real,\ mediu} \cong 0{,}57. \qquad (2.4.24)$$

Aceasta înseamnă că pentru mai multe rețele antrenate, presupunând că erorile lor sunt independente statistic, în medie $q_{max,\ real}$ va fi 0,57 din valoarea aleasă $q_{max,\ impus}=1$. Dar, cu o anumită probabilitate (destul de mică), raportul $q_{max,\ real}$ este mai mare de 0,8, adică relativ apropiat de 1. Este astfel posibil, deși nu foarte probabil, ca pentru o altă antrenare a RN, să se obțină o pantă maximă $q_{max,\ real}$ mai apropiată de valoarea lui $q_{max,\ impus}$.

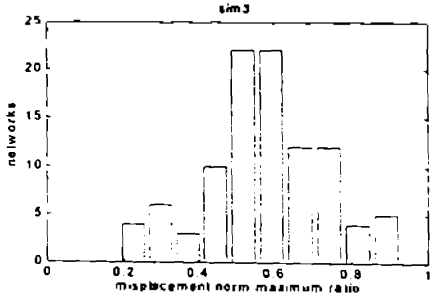


Figura 2.4.14. Maximul obținut prin simulare al raportului mărimilor erorii de poziționare, $q_{max,\ real}$, pentru 100 de rețele având erori aleatoare satisfăcând mSE=0.084. Maximul a fost detectat dintre 200 de direcții aleatoare pornind din $\theta(0) = (0, \pi/2, \pi/3)$.

Revenim la figura 2.4.13. Pentru o mărime a erorii de poziționare sau o mărime a deplasării mai mare, $\|\mathbf{m}(0)\|$=0,03, în ambele puncte a) și b) încep să apară erorile de neliniaritate : pantele maxime $q_{max.\ real}$ sunt cu ceva mai mari decât pentru $\|\mathbf{m}(0)\|$=0,003. Pentru deplasări mari, $\|\mathbf{m}(0)\|$=0,3 pornind din θ(0)=(0, π/2, π/3), există direcții pentru care $q_{max.\ real} > 1$, distanța la țintă crește.

În final a fost studiată prin simulare dependența lui $q_{max.\ real}$ cu $\|\mathbf{m}\|$. Rezultatele sunt prezentate în figura 2.4.15. Pe figura 2.4.15 a) se vede că pentru valori mici ale mărimii erorii de poziționare, $\|\mathbf{m}\| \le 0{,}005$, $q_{max.\ real}$ este aproximativ constant, cu valoarea dată de eroarea RN. Acesta este domeniul pentru care modelul liniar este valabil. Valorile ce se obțin de pe grafic sunt în concordanță cu cele obținute în figura 2.4.13 pentru $\|\mathbf{m}\| = 0{,}003$. La valori mai mari ale lui $\|\mathbf{m}\|$, pentru unele puncte încep să se facă simțite erorile de neliniaritate. Pentru $\|\mathbf{m}\| \ge 0.02$ acestea domină. Până la $\|\mathbf{m}\| \equiv 0.3$, panta curbelor reprezentate pe scara logaritmică este unitară: aceasta se poate vedea trecând la coordonate liniare. figura b), unde unde creșterile în domeniul respectiv sunt într-adevăr liniare. Faptul este în concordanță cu modelul pătratic din § 2.4.2. unde s-a obținut că q_q crește liniar cu mărimea erorii de poziționare (2.4.21'). Modelul pătratic este așadar valabil pe un domeniu neașteptat de mare, $\|\mathbf{m}\| \le 0.3$, adică pentru deplasări de lungimi ce pot atinge circa o treime din lungimea brațelor L_2+L_3. În fine, de pe această porțiune a curbelor se poate obține $\|\mathbf{m}\|_{max}$ în funcție de $q_{max.q}$. Pentru 1/2, valoarea aleasă în tabelul 2.4.1, valoarea minimă dintre cele trei curbe reprezentate se obține pentru pentru θ(0) = (0, π/2, π/3) și pentru θ(0) = (0, π/2, π/3). $\|\mathbf{m}\|_{max} \equiv$ 0.065. Concordanța cu valoarea din tabelul 2.4.1, calculată, este excelentă. De asemenea, valoarea $\|\mathbf{m}\|_{max}$ ce se obține pentru θ(0) = (0, 0. 0) concordă cu cea care rezultă din figura 2.4.7.

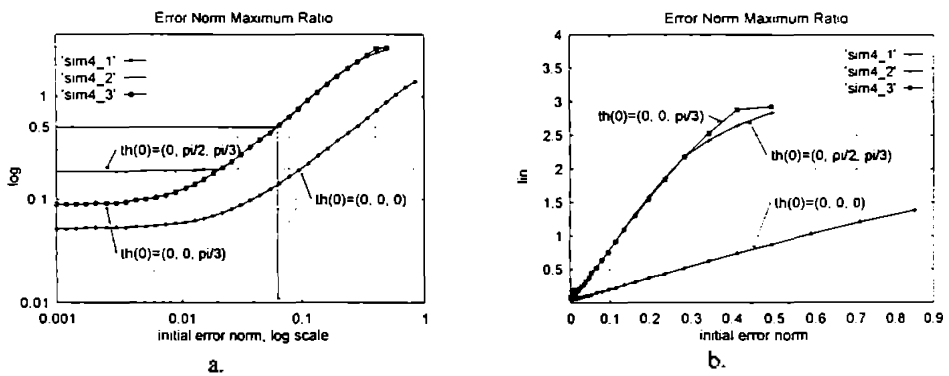


Figura 2.4.15. Raportul $q_{max.\ real}$ la efectuarea unei iterații.
în funcție de mărimea erorii inițiale de poziționare:
a) scări liniare. b) scări logaritmice.
Maximul a fost detectat pentru câte 2000 de direcții aleatoare.

2.4.5. Concluzii

Concluziile ce se desprind din paragrafele teoretice ale acestui subcapitol sunt :

1. Comportarea în prezenţa erorilor depinde de direcţia de deplasare şi de poziţie.

2. Antrenarea RN minimizează în general eroarea medie pătratică; dacă este necesar un alt criteriu de eroare, care aici a fost norma matricială 2, apar majorări suplimentare, făcând mai restrictivă limita de eroare.

3. Domeniul de valabilitate al modelării local liniare poate fi studiat printr-un model local pătratic. Rezultatele ce se obţin pot fi utile pentru proiectarea reţelelor de cuantizare vectorială precum reţeaua Kohonen extinsă, sau reţeaua CMAC (v. § 1.2.1, 1.2.3).

Aplicarea rezultatelor teoretice braţului de robot considerat a mai adus concluzia:

4. Erorile maxime admisibile pentru inversa jacobienei robotului depind de poziţie. Braţul ERICC poate fi aproximat liniar pe domenii mai extinse în centrul domeniului său de lucru.

Rezultatele teoretice au fost verificate prin simulări. Acestea aduc încă câteva concluzii:

5. În practică, pentru o anumită antrenare a RN, eroarea maximă admisibilă pentru $\mathbf{Jr}^{-1}$ poate părea prea restrictivă. Însă ea garantează descreşterea erorii de poziţionare pentru orice repartiţie între ieşiri a erorilor RN, pentru orice poziţie şi pentru orice direcţie a deplasării. Conform (2.4.24), se poate considera că în practică eroarea pătratică maximă nu ar trebui să depăşească de mai mult de $(1/q_{max,\,real,\,mediu})^2 \cong 3$ ori valoarea calculată.

6. Concordanţa rezultatelor de simulare cu modelul local pătratic calculat este excelentă. Domeniul de valabilitate al acestuia este surprinzător de mare: pentru braţul ERICC acesta poate atinge, în centrul domeniului de lucru, până 1/3 din lungimea segmentelor deplasabile ale braţului. Prin aceasta, modelul local pătratic poate fi utilizat şi ca un model în sine.

7. Se poate menţiona diferenţa apreciabilă dintre eroarea pătratică maximă şi eroarea pătratică medie. După antrenarea reţelei, erori pătratice mari apar în doar câteva puncte. Asigurarea convergenţei poziţionării pentru ţinte situate în aceste puncte este considerabil mai dificilă decât asigurarea convergenţei în medie, obţinută dacă antrenarea reţelei este oprită conform mediei pe setul de antrenare a erorii pătratice. În acest din urmă caz există riscul ca, chiar dacă în general poziţionarea se face cu succes, pentru anumite poziţii ale ţintei, ea să divearge.

8. Cea mai importantă concluzie ce poate fi desprinsă este utilitatea unui algoritm adaptiv, pentru care dimensiunea domeniilor de aproximare liniară să fie obţinută adaptiv, în funcţie de braţul disponibil, de poziţiile de interes, şi poate chiar de direcţiile deplasării de efectuat.

3. Învăţare adaptivă pentru modele local liniare

Capitolul precedent a prezentat comanda cinematică a braţului de robot utilizând modele local liniare, calculând erorile admisibile maxime pentru care robotul se apropie de ţintă la fiecare iteraţie. Capitolul de faţă studiază învăţarea acestor modele local liniare. Liniarizarea locală face posibilă utilizarea metodelor de învăţare din filtrarea adaptivă.

Problema corespunde modelării adaptive inverse (v. § 1.3). Anume, ea este aceea a învăţării matricii ce caracterizează blocul de comandă $\mathbf{Jc} \in \mathbf{R}^{3x4}$ din § 2.3. Pentru generalitate, ea va fi înlocuită cu matricea $\mathbf{A} \in \mathbf{R}^{mxn}$, cu m şi n oarecare. Ca atare, deplasările din spaţiul "vizual" Δv, ce sunt intrările pentru blocul **Jc**, îşi păstrează notaţia, dar devin vectori coloană reali cu n componente, iar ieşirile $\Delta\theta$, reprezentând deplasări unghiulare, sunt vectori coloană reali cu m componente. Pe baza perechilor cunoscute succesiv $\Delta\theta(k)$, $\Delta v(k)$, $k = 1, 2, ..., K$, se obţin estimate succesive ale lui **A**, $\hat{\mathbf{A}}(k)$. La iteraţia k ieşirea se calculează cu estimarea precedentă a lui $\hat{\mathbf{A}}$ şi se obţine eroarea denumită uneori apriori (Bellanger, 1987) :

$$\mathbf{e}(k) = \Delta\theta(k) - \hat{\mathbf{A}}(k-1)\Delta v(k). \qquad (3.0.1)$$

Printr-un algoritm iterativ de adaptare se obţine, pe baza ei, următoarea estimare $\hat{\mathbf{A}}(k+1)$, putându-se calcula şi eroarea aposteriori :

$$\mathbf{e}'(k) = \Delta\theta(k) - \hat{\mathbf{A}}(k)\Delta v(k). \qquad (3.0.2)$$

În acest capitol se urmăreşte obţinerea unui algoritm de adaptare iterativ cât mai performant şi compararea lui cu cele deja existente.

3.1. O analiză a algoritmului LMS

Atât în RN, cât şi în filtarea adaptivă, este binecunoscută metoda adaptării după gradient, denumită şi "coborârea cea mai abruptă" (*steepest descent*). În RN ea dă naştere algoritmilor de adaptare a ponderilor cunoscuţi sub numele "regula delta", şi "algoritmul *Backpropagation*" (v. § 1.1.3). Ea este dealtfel o metodă utilizabilă pentru minimizarea unor funcţii oarecare (Press et al, 1988). Corespunzător acestei metode, cel mai des utilizat este algoritmul LMS (*Least Mean Squares*), care înlocuieşte gradientul prin estimarea sa dată de vectorii de intrare şi ieşire prezentaţi la iteraţia curentă.

Ritter, Martinetz şi Schulten (1992) folosesc algoritmul LMS pentru învăţarea de modele locale liniare în comanda cinematică a unui braţ de robot cu reacţie vizuală. La iteraţia k, ei îl adaptează pe $\hat{\mathbf{A}}$ astfel :

- se comandă deplasarea braţului cu $\Delta\theta(k)$,
- se observă $\Delta v(k)$ şi se calculează eroarea de comandă (3.0.1),
- se adaptează matricea de comandă prin :

$$\Delta\hat{\mathbf{A}}_{LMS}(k) = \eta\mathbf{e}(k)\Delta v(k)^T, \qquad (3.1.1)$$

obţinându-se noua matrice

$$\hat{\mathbf{A}}_{LMS}(k) = \hat{\mathbf{A}}_{LMS}(k-1) + \Delta\hat{\mathbf{A}}_{LMS}(k) = \hat{\mathbf{A}}_{LMS}(k-1) + \eta\mathbf{e}(k)\Delta v(k)^T, \qquad (3.1.2)$$

unde η este parametrul învățării. Ei dau și valoarea optimă a acestuia, fără a o justifica :

$$\eta_{opt}(k) = \frac{1}{\|\Delta v(k)\|^2} \tag{3.1.3}$$

norma vectorială fiind cea euclidiană.

Explicația acestei valori este simplă : Pentru $\eta = \eta_{opt}$ matricea nou obținută $\hat{A}(k)$ anulează eroarea aposteriori $e'(k)$,

$$e'(k) = \Delta\theta(k) - \left\{\hat{A}_{LMS}(k-1) + \eta\left[\Delta\theta(k) - \hat{A}_{LMS}(k-1)\,\Delta v(k)\right]\Delta v(k)^T\right\}\Delta v(k) =$$

$$= \left[\Delta\theta(k) - \hat{A}_{LMS}(k-1)\,\Delta v(k)\right]\left[1 - \eta\,\Delta v(k)^T\Delta v(k)\right] \tag{3.1.4}$$

$$e'(k) = 0\ ,\ (\forall)\,\Delta v, \Delta\theta \ \Leftrightarrow\ \eta = \eta_{opt}(k) = \frac{1}{\Delta v(k)^T\Delta v(k)} = \frac{1}{\|\Delta v(k)\|^2}\ .$$

În acest caz, modificarea $\Delta\hat{A}_{LMS}(k)$ devine $\Delta\hat{A}_{LMS,opt}(k)$,

$$\Delta\hat{A}_{LMS,opt}(k) = \eta_{opt}\,e(k)\Delta v(k)^T = \frac{1}{\|\Delta v(k)\|^2}e(k)\Delta v(k)^T\ , \tag{3.1.5}$$

corespunzând unei valori $\hat{A}_{LMS,opt}(k)$

$$\hat{A}_{LMS,opt}(k) = \hat{A}_{LMS,opt}(k-1) + \Delta\hat{A}_{LMS,opt}(k). \tag{3.1.6}$$

În cele ce urmează, dacă nu se menționează explicit, se consideră că adaptarea se face întotdeauna după optim, iar indicele $_{opt}$ se omite.

Se consideră că datele disponibile $\Delta v(k)$ și $\Delta\theta(k)$, $(\forall)$ k, nu conțin erori și nici zgomot, și reprezintă un sistem liniar. În acest caz se pot constata, pentru algoritmul LMS optimizat, următoarele proprietăți :

1. Dacă următoarea deplasare este coliniară cu precedenta, $\Delta v(k+1) = \alpha\Delta v(k)$, și deci $\Delta\theta(k+1) = \alpha\Delta\theta(k)$, atunci eroarea apriori $e(k+1)$ este nulă; direcția $\Delta v(k)$ a fost învățată corect:

$$e(k+1) = \Delta\theta(k+1) - \hat{A}_{LMS}(k)\,\Delta v(k+1) =$$
$$= \alpha\left[\Delta\theta(k) - \hat{A}_{LMS}(k)\,\Delta v(k)\right] = \alpha e'(k) = 0 \tag{3.1.7}$$

2. Dacă următoarea deplasare este ortogonală pe precedenta, $\Delta v(k+1) \perp \Delta v(k)$, efectul adaptării precedente $\Delta A_{LMS}(k)$ este nul :

$$\hat{A}_{LMS}(k)\Delta v(k+1) = \left[\hat{A}_{LMS}(k-1) + \Delta\hat{A}_{LMS}(k)\right]\Delta v(k+1) =$$
$$= \hat{A}_{LMS}(k-1)\Delta v(k+1) + \eta e(k)\Delta v(k)^T\Delta v(k+1) = \tag{3.1.8}$$
$$= \hat{A}_{LMS}(k-1)\Delta v(k+1) \quad \text{d.d.} \quad \Delta v(k) \perp \Delta v(k+1)\ .$$

Aceasta înseamnă că, dacă $\Delta v(k+1)$ are o direcție diferită de $\Delta v(k)$, $\Delta v(k+1) = \alpha\Delta v(k) + \beta\Delta v(k)^{\perp}$ eroarea apriori la iterația k+1, dată de componenta ortogonală $\beta\Delta v(k)^{\perp}$, este aceeași ca în lipsa adaptării $\Delta A_{LMS}(k)$.

3. Dacă a k+1-a corecție, $\Delta A_{LMS}(k+1)$, este dată de un vector $\Delta v(k+1)$ neortogonal pe $\Delta v(k)$ și de direcție neînvățată deja, $e(k+1) \neq 0$, ea strică parțial ce s-a învățat la iterația k :

$$\hat{A}_{LMS}(k+1)\,\Delta v(k) = \left[\hat{A}_{LMS}(k) + \Delta\hat{A}_{LMS}(k+1)\right]\Delta v(k) =$$
$$= \hat{A}_{LMS}(k)\Delta v(k) + \eta_{opt}(k+1)e(k+1)\Delta v(k+1)^T \Delta v(k) = \qquad (3.1.9)$$
$$= \Delta\theta(k) + \frac{1}{\Delta v(k+1)^2}\, e(k+1)\Delta v(k+1)^T \Delta v(k) \neq \Delta\theta(k)$$

Aceste observații demonstrează că învățarea cea mai eficace a unui sistem liniar prin algoritmul LMS se face prezentând perechi intrare-ieșire pentru care vectorii de intrare sunt ortogonali. Faptul a fost observat și de Goodwin și Sin (1984) și de Polycarpou și Ioannou (1991) Aceștia din urmă îl utilizează pentru obținerea pseudoinverselor cu rang plin (*full rank*) pe linii sau coloane ´r *rang*($\mathbf{A}^{mxn}$) = *min*(m.n), în r pași, printr-un algoritm de tip "neuronal", LMS. ce ortogonalizează vectorii prezentați. Esența demonstrației lor este algoritmul proiecției ortogonalizate din (Goodwin și Sin, 1984). Algoritmul proiecției ortogonalizate este o variantă a algoritmului LMS, ce ortogonalizează vectorii de intrare prin înmulțirea lor cu o matrice proiecție care este construită din vectorii prezentați anterior. La rândul ei, demonstrația algoritmului proiecției ortogonalizate se bazează pe ortogonalitate și este tehnică, făcută prin inducție. Nu se fac mențiuni asupra semnificațiilor componentelor obținute sau ale matricilor proiecție construite iterativ.

3.2. Pseudoinversa Moore-Penrose și algoritmul Greville

Se pune deci problema calculării matricii $\mathbf{A} \in \mathbf{R}^{m \times n}$, ce reprezintă un sistem liniar satisfăcând la :

$$\Delta\theta(k) = \mathbf{A}\,\Delta v(k), \quad k = 1, 2, \ldots, K \tag{3.2.1}$$

pentru care se consideră că se cunosc K perechi de vectori intrare-ieșire.
Egalitățile (3.2.1) pot fi puse într-o formă compactă, matricială :

$$\begin{bmatrix}\Delta\theta(1) & \cdots & \Delta\theta(k) & \cdots & \Delta\theta(K)\end{bmatrix} = \mathbf{A}\begin{bmatrix}\Delta v(1) & \cdots & \Delta v(k) & \cdots & \Delta v(K)\end{bmatrix}$$
$$\Delta\Theta(K) = \mathbf{A}\,\Delta\mathbf{V}(K) \tag{3.2.2}$$

unde $\Delta\Theta(K) \in \mathbf{R}^{m \times K}$ și $\Delta\mathbf{V}(K) \in \mathbf{R}^{n \times K}$ sunt matricile conținând vectorii de ieșire, respectiv de intrare. După prezentarea a doar k vectori din cei K disponibili, se cunosc $\Delta\Theta(k)$ și $\Delta\mathbf{V}(k)$.

Fie :

r - dimensiunea subspațiului căruia îi aparțin vectorii Δv, $r \le n$.

r_k - rangul matricii $\Delta\mathbf{V}(k)$, $r_k \le min(r, k)$.

Se pot distinge mai multe cazuri, după cum se consideră că relațiile (3.2.1) sunt sau nu exacte, și după n, r și r_k .

1. Dacă sistemul comandat este liniar, datele $\Delta\theta(k)$, $\Delta v(k)$ sunt cunoscute cu exactitate, și există suficienți vectori, $r_K \ge r$, atunci ecuațiile (3.2.3) au o soluție exactă, anume (Kohonen, 1984) :

$$\mathbf{A}_+(K) = \Delta\Theta(K)\,\Delta\mathbf{V}^+(K) \tag{3.2.3}$$

unde $\Delta\mathbf{V}^+$ este pseudoinversa Moore-Penrose a matricii $\Delta\mathbf{V}$; (v. Anexa I). Dependența lui $\mathbf{A}_+$ de numărul datelor disponibile K nu va fi menționată explicit decât când va fi cazul. De remarcat că sunt suficienți k=r vectori, liniar independenți; ceilalți $\Delta\theta(k)$, $\Delta v(k)$, cu $k \ge r_k$ sunt combinații liniare ale celorlalți.

Dacă în plus K = r = m = n , atunci soluția ia forma particulară

$$\mathbf{A}_{-1} = \Delta\Theta\,\Delta\mathbf{V}^{-1}$$

unde $\Delta\mathbf{V}^{-1}$ este inversa matricii $\Delta\mathbf{V}$.

Dacă sistemul comandat este neliniar, sau datele sunt zgomotoase, atunci ecuațiile (3.2.1) sunt satisfăcute doar cu aproximație, iar soluția este, de asemenea, aproximativă. Se poate obține doar o soluție aproximativă, dată de (3.2.2) și dacă sistemul este liniar și datele exacte, dar insuficiente, $K \le r$.

2. Dacă se consideră că sistemul **A** este liniar, și că sunt zgomotoase doar valorile $\Delta\theta(k)$, atunci cea mai bună aproximare disponibilă pentru **A** este dată de (3.2.3). Aproximarea este în sensul celor mai mici pătrate. Aceasta înseamnă

$$\left\|\Delta\Theta - \hat{\mathbf{A}}_+\Delta\mathbf{V}\right\|_E = \min_{\hat{\mathbf{A}}}\left\|\Delta\Theta - \hat{\mathbf{A}}\Delta\mathbf{V}\right\|_E, \tag{3.2.4}$$

unde $\|\cdot\|_E$ este norma matricială euclidiană sau Frobenius (v. anexa I). În fapt, soluțiile pot fi multiple, $\mathbf{A}_+$ fiind soluția de normă euclidiană minimă, unică.

Acesta este cazul acoperit de tehnicile de filtrare adaptivă, unde ieșirea este, de obicei, scalară, m = 1.

3. Dacă se consideră că sistemul **A** este liniar, dar sunt zgomotoase şi observaţiile $\Delta\theta(k)$ şi $\Delta v(k)$, sau doar cele din urmă, atunci problema devine a celor mai mici pătrate totale (*total least squares*; Golub şi Van Loan, 1989; Van Huffel şi Vanderwalle, 1991).

4. Dacă funcţia de comandă $\Delta\theta = \Delta\theta(\Delta v)$ este neliniară, dar interesează o aproximaţie liniară (eventual locală) a acesteia, iar datele $\Delta\theta(k)$ şi $\Delta v(k)$ sunt exacte, atunci soluţia aproximativă optimă este dată tot de (3.2.3). Aproximaţia este obţinută tot în sensul erorii pătratice. Acesta este cazul tratat aici.

Soluţia pentru date fără zgomot

Se studiază soluţia $\mathbf{A}_-$ (3.2.3) pentru cazul când datele sunt exacte. Fie acestea generate prin

$$\Delta\Theta(K) = \mathbf{A}_0\,\Delta\mathbf{V}(K). \tag{3.2.5}$$

Atunci

$$\mathbf{A}_-(K) = \Delta\Theta(K)\,\Delta\mathbf{V}^+(K) = \mathbf{A}_0\,\Delta\mathbf{V}(K)\,\Delta\mathbf{V}^+(K) = \mathbf{A}_0\,\mathbf{P}_{\Delta\mathbf{V}(K)}, \tag{3.2.6}$$

unde $\mathbf{P}_{\Delta\mathbf{V}(K)} = \Delta\mathbf{V}\,\Delta\mathbf{V}^+$ este matricea proiecţie pe subspaţiul coloanelor lui $\Delta\mathbf{V}$, $\mathcal{R}(\Delta\mathbf{V}(K))$ (v. anexa I). Aceasta înseamnă că vectorii linie ai soluţiei $\mathbf{A}_-$ sunt vectorii linie ai matricii $\mathbf{A}_0$, proiectaţi în subspaţiul vectorilor de intrare disponibili $\mathcal{R}(\Delta\mathbf{V}(K))$. Eroarea soluţiei este :

$$\mathbf{A}_0 - \mathbf{A}_- = \mathbf{A}_0\left(\mathbf{I} - \mathbf{P}_{\Delta\mathbf{V}(K)}\right) = \mathbf{A}_0\,\mathbf{P}^{\perp}_{\Delta\mathbf{V}(K)}. \tag{3.2.7}$$

unde $\mathbf{P}^{\perp}_{\Delta\mathbf{V}(K)}$ este matricea proiecţie în $\mathcal{R}(\Delta\mathbf{V}(K)^{\perp})$, subspaţiul ortogonal al lui $\mathcal{R}(\Delta\mathbf{V}(K))$. Dacă

$$r_K = dim\,\mathcal{R}(\Delta\mathbf{V}(K)) < n,$$

atunci, chiar cu date exacte, nu se poate obţine soluţia exactă $\mathbf{A}_0$ pe baza vectorilor de intrare $\Delta\mathbf{V}(K)$.

Eroarea ieşirilor, după învăţare, va fi:

$$\begin{aligned}\mathbf{A}_0\,\Delta\mathbf{V} - \mathbf{A}_-(K)\,\Delta\mathbf{V}(K) &= \mathbf{A}_0\left(\mathbf{I} - \Delta\mathbf{V}(K)\,\Delta\mathbf{V}^+(K)\right)\Delta\mathbf{V}(K) = \\ &= \mathbf{A}_0\left(\Delta\mathbf{V}(K) - \Delta\mathbf{V}(K)\right) = \mathbf{0}\,.\end{aligned} \tag{3.2.8}$$

A fost folosită proprietatea pseudoinversei $\mathbf{X}\,\mathbf{X}^+\,\mathbf{X} = \mathbf{X}$ (anexa I, 9.5.1). Aşadar, pentru date fără zgomot, chiar dacă soluţia $\mathbf{A}^+$ este eronată, erorile ieşirilor sunt nule.

Soluţia pentru date zgomotoase

Dacă datele disponibile sunt zgomotoase (3.2.9).

$$\begin{aligned}\Delta\theta(k) &= \mathbf{A}_0\,\Delta v(k) + \mathbf{n}(k), \quad k = \overline{1,K} \\ \Delta\Theta(K) &= \mathbf{A}_0\,\Delta\mathbf{V}(K) + \mathbf{N}(K)\end{aligned} \tag{3.2.9}$$

soluţia devine (3.2.10), iar eroarea ei este (3.2.11).

$$\mathbf{A}_-(K) = \Delta\Theta(K)\,\Delta\mathbf{V}^+(K) = \mathbf{A}_0\,\Delta\mathbf{V}(K)\,\Delta\mathbf{V}^+(K) + \mathbf{N}(K)\,\Delta\mathbf{V}^+(K) \tag{3.2.10}$$

$$\mathbf{A}_0 - \mathbf{A}_-(K) = \mathbf{A}_0\left(\mathbf{I} - \mathbf{P}_{\Delta\mathbf{V}(K)}\right) = \mathbf{A}_0\,\mathbf{P}^{\perp}_{\Delta\mathbf{V}(K)} - \mathbf{N}(K)\,\Delta\mathbf{V}^+(K) \tag{3.2.11}$$

Dacă intrările și zgomotul sunt necorelate și zgomotul are medie nulă, atunci soluția medie este (3.2.12).

$$E\left[\mathbf{A}_0 - \mathbf{A}_-(K)\right] = \mathbf{A}_0\, E\left[\mathbf{P}^{\perp}_{\Delta v(K)}\right] - E\left[\mathbf{N}(K)\right] E\left[\Delta \mathbf{V}^{+}(K)\right] = \mathbf{A}_0\, E\left[\mathbf{P}_{\Delta V(K)^T}\right] \tag{3.2.12}$$

Dacă vectorii de intrare sunt liniar dependenți, atunci $E\left[\mathbf{P}^{\perp}_{\Delta V(K)}\right] \neq \mathbf{0}$, soluția medie este deplasată. Conform (3.2.4), erorile de ieșire (3.2.13) au suma pătratelor, sau norma euclidiană a matricii (3.2.14), minimă.

$$\mathbf{e}'(K) = \Delta\theta(k) - \mathbf{A}_-(K)\,\Delta v(k) \quad k = \overline{1,K} \tag{3.2.13}$$

$$\mathbf{E}'(K) = \left[\mathbf{e}'(k)_{k=1,K}\right] = \Delta\Theta(K) - \mathbf{A}_+(K)\,\Delta V(K) = \mathbf{N}(K)\,\mathbf{P}^{\perp}_{\Delta V^T(K)} \tag{3.2.14}$$

În general, prezintă interes calculul iterativ al estimărilor lui $\mathbf{A}$, și deci al pseudoinversei ΔV^{+}, pe măsura acumulării vectorilor $\Delta\theta(k)$ și $\Delta v(k)$, k=1,2.... .

Matricile datelor $\Delta\Theta(k)$ și $\Delta V(k)$, conținând câte k vectori, se construiesc din $\Delta\Theta(k-1)$ și $\Delta V(k-1)$, prin adăugarea lui $\Delta\theta(k)$ respectiv $\Delta v(k)$:

$$\Delta\Theta(k) = \left[\Delta\Theta(k-1) \quad \Delta\theta(k)\right] \tag{3.2.15}$$

$$\Delta V(k) = \left[\Delta V(k-1) \quad \Delta v(k)\right] \tag{3.2.16}$$

Pseudoinversa matricii $\Delta V(k+1)$ poate fi calculată direct, de exemplu prin descompunerea după valori singulare. Dar ea poate fi calculată și recursiv, pe baza partiționării (3.2.16), prin teorema lui Greville (v. anexa I; Greville, 1960; Boullion și Odell, 1971; Albert, 1972) :

$$\Delta V^{+}(k) = \begin{bmatrix} \Delta V^{+}(k-1)\left(\mathbf{I} - \Delta v(k)\,\mathbf{p}^T(k)\right) \\ \mathbf{p}^T(k) \end{bmatrix}. \tag{3.2.17}$$

unde

$$\mathbf{p}(k) = \begin{cases} \dfrac{\left(\mathbf{I} - \Delta V(k-1)\,\Delta V^{+}(k-1)\right)\Delta v(k)}{\left\|\left(\mathbf{I} - \Delta V(k-1)\,\Delta V^{+}(k-1)\right)\Delta v(k)\right\|^2}, & \text{dacă numărătorul este} \neq 0 \\ \dfrac{\Delta V^{+T}(k-1)\,\Delta V^{+}(k-1)\Delta v(k)}{1 + \left\|\Delta V^{+}(k-1)\Delta v(k)\right\|^2} & \text{în caz contrar.} \end{cases} \tag{3.2.18}$$

Pentru k=1.

$\Delta V(1) = \Delta v(1)$.

$$\Delta V^{+}(1) = \Delta v^{+}(1) = \mathbf{p}^T(1) = \frac{\Delta v^T(1)}{\left\|\Delta v(1)\right\|^2}. \tag{3.2.19}$$

Scriind matricile $\Delta\Theta(k)$ și $\Delta V(k)$ sub formă partiționată (3.2.15), (3.2.16), calculul estimatei $\hat{\mathbf{A}}_-(k)$ dat de (3.2.3) se poate pune sub o formă recursivă, denumit în continuare algoritmul Greville :

$$\hat{\mathbf{A}}_{-}(k) = \Delta\Theta(k)\Delta V^{-}(k) =$$
$$= \left[\Delta\Theta(k-1) \quad \Delta\theta(k)\right]\begin{bmatrix}\Delta V^{+}(k-1)\left(\mathbf{I} - \Delta v(k)\,\mathbf{p}^{T}(k)\right)\\ \mathbf{p}^{T}(k)\end{bmatrix} = \tag{3.2.20}$$
$$= \Delta\Theta(k-1)\Delta V^{-}(k-1)\left(\mathbf{I} - \Delta v(k)\,\mathbf{p}^{T}(k)\right) + \Delta\theta(k)\mathbf{p}^{T}(k) =$$
$$= \hat{\mathbf{A}}_{-}(k-1)\left(\mathbf{I} - \Delta v(k)\,\mathbf{p}^{T}(k)\right) + \Delta\theta(k)\mathbf{p}^{T}(k)$$

cu $\mathbf{p}(k)$ dat de (3.2.18).

După cum se va vedea mai jos, ecuațiile (3.1.2) pot fi rezolvate iterativ în sensul celor mai mici pătrate și prin algoritmul RLS (*Recursive Least Squares*). Acesta poate fi însă aplicat doar pentru vectori Δv având componentele liniar independente; dacă una din componentele vectorilor de intrare este combinație liniară de celelalte, atunci matricea lor de corelație nu este inversabilă și algoritmul RLS nu poate fi aplicat. Aceasta se întâmplă implicit atunci când vectorii $\Delta v(k)$, $k=1...K$ aparțin unui subspațiu liniar de dimensiune r mai mică decât n, dimensiunea vectorilor Δv: acesta este cazul aplicației de față, datorat vederii stereoscopice. Avantajul major al pseudoinversei este acela că permite rezolvarea acestor ecuații și în acest caz.

În paragrafele următoare vor fi folosite următoarele noțiuni și notații (v. și anexa I) :

Spațiul euclidian $\mathbf{R}^n$ căruia îi aparțin vectorii $\Delta v(k)$ $k = 1,2,...,K$ poate fi scris, pentru fiecare k, ca suma directă dintre subspațiul generat de vectorii $\Delta v(i)$ $i = 1,2,...,k-1$ sau coloanele matricii $\Delta V(k-1)$, $span\,\Delta V(k-1) = \mathcal{R}(\Delta V(k-1))$, și complementul său ortogonal, care este și subspațiul nul al matricii $\Delta V^{T}(k-1)$. $\mathcal{R}(\Delta V(k-1))^{\perp} = \mathcal{N}(\Delta V^{T}(k-1))$,

$$\mathbf{R}^{n} = \mathcal{R}(\Delta V(k-1)) \oplus \mathcal{R}(\Delta V(k-1))^{\perp}. \tag{3.2.21}$$

Atunci $\Delta v(k)$ poate fi descompus în mod unic astfel :

$$\Delta v(k) = \Delta v^{\parallel}(k) + \Delta v^{\perp}(k),$$
$$\Delta v^{\parallel}(k) \in \mathcal{R}(\Delta V(k-1)), \tag{3.2.22}$$
$$\Delta v^{\perp}(k) \in \mathcal{R}(\Delta V(k-1))^{\perp},$$

cu

$$\Delta v^{\parallel}(k) = \Delta V(k-1)\,\Delta V^{-}(k-1)\,\Delta v(k) = \mathbf{P}_{\Delta V(k-1)}\,\Delta v(k)$$
$$\Delta v^{\perp}(k) = \left[\mathbf{I} - \Delta V(k-1)\,\Delta V^{-}(k-1)\right]\Delta v(k) = \mathbf{P}^{\perp}_{\Delta V(k-1)}\,\Delta v(k) \tag{3.2.23}$$

unde $\mathbf{P}_{\Delta V(k-1)}$ și $\mathbf{P}^{\perp}_{\Delta V(k-1)}$ sunt matricile proiecție ortogonală pe subspațiile $\mathcal{R}(\Delta V(k-1))$ și respectiv $\mathcal{R}(\Delta V(k-1))^{\perp}$.

Așadar $\Delta v^{\parallel}(k)$ este componenta lui $\Delta v(k)$ din subspațiul generat de vectorii deja învățați, $\Delta v(j)$, $j = \overline{1,\,k-1}$, iar $\Delta v^{\perp}(k)$ este componenta lui $\Delta v(k)$ ortogonală pe toți vectorii deja învățați, reprezentând o direcție încă necunoscută.

Observație importantă. În (3.2.12) decizia e luată conform mărimii vectorului (3.2.24),

$$\left(\mathbf{I} - \Delta\mathbf{V}(k-1)\,\Delta\mathbf{V}^{+}(k-1)\right)\Delta v(k) = \mathbf{P}^{\perp}(k-1)\Delta v(k) = \Delta v^{\perp}(k), \tag{3.2.24}$$

iar (3.2.18) poate fi rescris ca :

$$\mathbf{p}(k) = \begin{cases} \dfrac{\Delta v^{\perp}(k)}{\left\|\Delta v^{\perp}(k)\right\|^{2}} & \text{pentru } \left\|\Delta v^{\perp}(k)\right\| \neq 0 \\ \dfrac{\Delta\mathbf{V}^{+T}(k-1)\,\Delta\mathbf{V}^{+}(k-1)\,\Delta v^{\|}(k)}{1+\Delta v^{\|T}(k)\,\Delta\mathbf{V}^{+T}(k-1)\,\Delta\mathbf{V}^{+}(k-1)\,\Delta v^{\|}(k)} & \text{pentru } \left\|\Delta v^{\perp}(k)\right\| = 0 \end{cases} . \tag{3.2.25}$$

În ceea ce privește implementarea algoritmului, având în vedere că, datorită erorilor de cuantizare și de calcul, egalitatea exactă (cu zero) nu poate fi obținută, decizia în (3.2.18) se va face după cum norma lui $\Delta v^{\perp}(k)$ este sau nu mai mică decât un prag corelat cu precizia de calcul a calculatorului.

3.3. Comparaţie între algoritmul Greville şi algoritmul LMS.

3.3.1. Comparaţie teoretică

La algoritmul Greville, actualizarea matricii $\hat{\mathbf{A}}_+(k)$ (3.2.15) poate fi rescrisă astfel :

$$\hat{\mathbf{A}}_+(k) = \hat{\mathbf{A}}_+(k-1) + \left[\Delta\theta(k) - \hat{\mathbf{A}}_+(k-1)\Delta\mathbf{v}(k)\right]\mathbf{p}^T(k) = \\ = \hat{\mathbf{A}}_+(k-1) + \mathbf{e}(k)\mathbf{p}^T(k) \tag{3.3.1}$$

cu e(k) eroarea de comandă (3.1.1). Există o asemănare cu algoritmul LMS.

Dacă $\Delta\mathbf{v}^\perp(k)$ (3.2.16) este nenulă, adică dacă $\Delta\mathbf{v}(k)$ aduce informaţii despre o direcţie neînvăţată încă, atunci în (3.2.18) are loc :

$$\mathbf{p}(k) = \frac{\Delta\mathbf{v}^\perp(k)}{\left\|\Delta\mathbf{v}^\perp(k)\right\|^2}, \tag{3.3.2}$$

iar (3.3.1) devine

$$\hat{\mathbf{A}}_+(k) = \hat{\mathbf{A}}_+(k-1) + \mathbf{e}(k)\frac{\Delta\mathbf{v}^\perp(k)}{\left\|\Delta\mathbf{v}^\perp(k)\right\|^2}. \tag{3.3.3}$$

S-a obţinut algoritmul LMS optimizat (3.1.6), în care adaptarea la iteraţia k se face doar pe componenta vectorului $\Delta\mathbf{v}(k)$ ortogonală pe ceilalţi vectori deja prezentaţi. Dacă în plus, în algoritmul LMS optimizat se prezintă doar vectori $\Delta\mathbf{v}(j)$ succesiv ortogonali, atunci conform (3.1.8) eroarea pe direcţiile deja învăţate $\Delta\mathbf{v}(j)$ nu va creşte iar e(k) în (3.3.3) şi (3.1.6) va fi aceeaşi.

Algoritmul Greville coincide aşadar cu algoritmul LMS optimizat, dacă acestuia din urmă i se furnizează vectori $\Delta\mathbf{v}$ ortogonali. Dacă subspaţiul vectorilor $\Delta\mathbf{v}$ este r-dimensional şi datele nu sunt zgomotoase, se poate ca după exact r vectori $\Delta\mathbf{v}$, $\mathbf{A}$ să fie cunoscut cu exactitate, $\mathbf{e} = \mathbf{0}$. În acest caz, soluţiile $\hat{\mathbf{A}}_{LMS}$ (3.1.6) respectiv $\hat{\mathbf{A}}_+$ (3.3.3) au aceeaşi valoare, şi nu se mai modifică la prezentarea de noi vectori. Pentru algoritmul LMS cei r vectori trebuie să fie reciproc ortogonali şi primii prezentaţi. Pentru algoritmul Greville este suficient ca cei r vectori să fie liniar independenţi, şi ei nu trebuie să fie primii r. Dacă datele $\Delta\theta$ sunt zgomotoase algoritmul Greville dă eroarea pătratică minimă în raport cu datele deja prezentate. În schimb, utilizarea algoritmului LMS şi a unui η mic poate aduce avantajul unei medieri (*e.g.* Widrow şi Stearns, 1985).

Dacă vectorii prezentaţi $\Delta\mathbf{v}$ prezentaţi algoritmului LMS nu sunt ortogonali, atunci învăţarea se face mai încet, conform celor prezentate în § 3.1.

3.3.2. Rezultate de simulare şi concluzie

Consideraţiile teoretice prezentate au fost verificate prin simulare. Datele $\Delta\theta$ sunt date conform modelului studiat mai înainte.

$$\Delta\theta(k) = A\Delta v + n(k), \qquad (3.3.4)$$

$A \in \mathbf{R}^{4\times3}$. A fost aleasă matricea A = [1.0 0.3 0.4 0.2;0.2 1.0 -0.5 0.2;0.6 -0.2 0.5 0.7] (notație MATLAB, pe linii). Vectorii Δv au componente aleatoare distribuite uniform în intervalul (-1, 1). Componentele erorii de măsurare **n** sunt zgomote gaussiene, de medie nulă și dispersie σ. În toate simulările ce urmează sunt reprezentate eroarea apriori, dinaintea adaptării k a ieșirii $e(k)$, și eroarea aposteriori, de după adaptarea k a soluției, $\|A - \hat{A}(k)\|_{:2}$. S-a ales o reprezentare logaritmică pe y pentru a se putea observa o gamă mai largă de erori. Pentru a se putea observa în detaliu comportarea la primele iterații, în condițiile unui număr total de 100 de iterații, s-a ales și pe x o reprezentare logaritmică.

În figura 3.3.1 se prezintă o comparație între adaptarea prin algoritmul LMS optimizat și adaptarea prin algoritmul Greville în absența zgomotului, σ = 0. Calculul se face cu o eroare relativă de rotunjire de *eps* = $2^{-52} \cong 2{,}22\cdot10^{-16}$. Rezultatele simulărilor sunt în concordanță cu previziunile teoretice. Pentru algoritmul Greville numărul iterațiilor necesare pentru învățare este 4, dimensiunea vectorilor Δv, care acoperă întregul $\mathbf{R}^4$. Deși nu a fost reprezentată grafic, se poate menționa că pentru algoritmul LMS optimizat eroarea aposteriori $e'(k)$ este de fiecare dată nulă (în limitele preciziei de calcul). Același lucru se întâmplă și în cazul algoritmului Greville. Cu toate acestea, adaptarea prin algoritmul Greville este net mai rapidă decât cea obținută prin LMS.

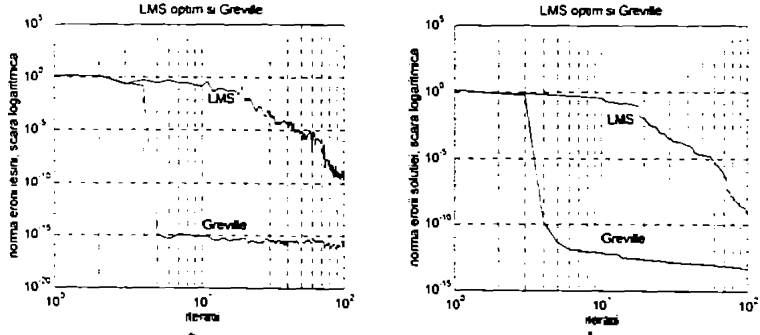


Figura 3.3.1. Comparație între învățarea prin algoritmul LMS optim și prin algoritmul Greville, în absența zgomotului. Precizia de calcul este $2{,}2\cdot10^{-16}$.
a) eroarea apriori a ieșirii b) eroarea soluției

Figura 3.3.2 prezintă rezultatul unei simulări în care datele de intrare sunt perturbate de un zgomot aditiv (3.3.4), cu σ = 0.01. Sunt reprezentate mediile pe 100 de rulări, fiecare rulare presupunând învățarea matricii **A** pe baza a 100 de perechi Δv-$\Delta\theta$. Matricea **A** este aceeași de mai sus. Se constată că și în prezența zgomotului adaptarea Greville este mult mai eficientă decât adaptarea LMS, chiar dacă valorile proprii ale matricii de învățat **A** nu sunt foarte dispersate (*e.g.* Widrow și Stearns, 1985), `svd(A)'=1.4557 1.1967 0.4570`. Pentru ambii algoritmi, prima adaptare este identică, vectorul $\Delta v(1)$ fiind ortogonal pe subspațiul general de vectorii precedenți, egal cu **{0}**.

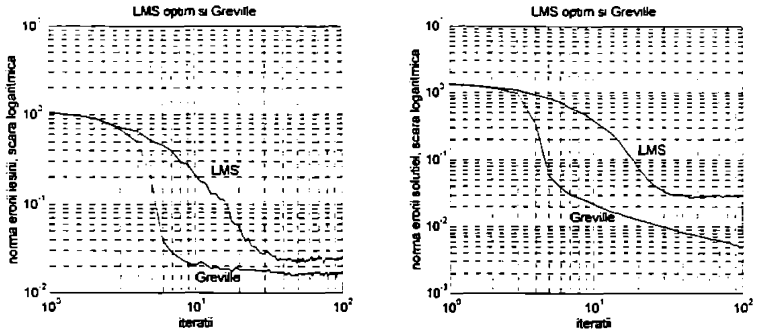


Figura 3.3.2. Comparaţie între învăţarea prin algoritmul LMS optim şi prin algoritmul Greville. Date zgomotoase, $\sigma = 10^{-2}$.
a) eroarea apriori a ieşirii. b) eroarea soluţiei.

Întrucât nu aduc informaţii suplimentare, nu sunt prezentate şi graficele erorilor aposteriori . Acestea sunt în concordanţă cu consideraţiile teoretice : chiar dacă datele sunt zgomotoase, pentru primele 4 iteraţii erorile aposteriori sunt neglijabile pentru ambii algoritmi, iar după aceea se constată că eroarea aposteriori a ieşirii pentru LMS este practic nulă, în timp ce pentru algoritmul Greville ea are valori de acelaşi ordin de mărime cu eroarea apriori.

3.4. Comparație între algoritmul Greville și algoritmul RLS[1]

Eroare pătratică minimă în cazul datelor $\Delta\theta$ zgomotoase, ca și calculul iterativ al parametrilor filtrului oferă și algoritmul de adaptare RLS (*Recursive Least Squares*) din filtrarea adaptivă. În continuare se face o comparație între algoritmul RLS și adaptarea ce folosește algoritmul Greville.

În filtrarea adaptivă ieșirea este de obicei scalară. $\mathbf{A}=\mathbf{a}\in\mathbf{R}^{n\times 1}$. Cu notațiile introduse în § 3.1, se caută $\hat{\mathbf{A}}_{RLS}(k)=\hat{\mathbf{a}}^{T}_{RLS}(k)$ pentru care eroarea pătratică

$$E(k)=\sum_{p=1}^{k}\lambda^{k-p}e^{2}(p)=\sum_{p=1}^{k}\lambda^{k-p}\left[\Delta\theta(p)-\hat{\mathbf{a}}^{T}_{RLS}(k)\Delta\mathbf{v}(p)\right]^{2} \qquad (3.4.1)$$

este minimă. Parametrul $\lambda\in(0,\ 1]$ servește, în cazul semnalelor nestaționare, la luarea în considerare doar a eșantioanelor din ultimul timp. Pentru semnale staționare se ia $\lambda=1$; acesta e și cazul considerat în continuare. Totuși, pentru generalitate, λ se păstrează.

Algoritmul RLS se obține, pe scurt, în modul următor : (Haykin, 1991; Bellanger, 1987)

Soluția de eroare pătratică minimă pentru (3.4.1) este :

$$\hat{\mathbf{a}}_{RLS}(k)=\hat{\mathbf{R}}^{-1}(k)\,\mathbf{r}_{\nu\theta}(k), \qquad (3.4.2)$$

unde $\hat{\mathbf{R}}$ este matricea de corelație deterministă a intrării, iar $\hat{\mathbf{r}}_{\nu\theta}$ este vectorul de intercorelație intrare-ieșire :

$$\hat{\mathbf{R}}(k)=\sum_{p=1}^{k}\lambda^{k-p}\Delta\mathbf{v}(p)\,\Delta\mathbf{v}^{T}(p),\quad \hat{\mathbf{r}}_{\nu\theta}(k)=\sum_{p=1}^{k}\lambda^{k-p}\Delta\mathbf{v}(p)\,\Delta\theta(p). \qquad (3.4.3)$$

Au loc relațiile de recurență :

$$\hat{\mathbf{R}}(k)=\lambda\hat{\mathbf{R}}(k-1)+\Delta\mathbf{v}(k)\Delta\mathbf{v}^{T}(k),\quad \hat{\mathbf{r}}_{\nu\theta}(k)=\lambda\hat{\mathbf{r}}_{\nu\theta}(k)+\Delta\mathbf{v}(k)\Delta\theta(k)\ . \qquad (3.4.4)$$

Folosind lema inversării matriciale,

$$\left(\mathbf{M}+\mathbf{BCB}^{T}\right)^{-1}=\mathbf{M}^{-1}-\mathbf{M}^{-1}\mathbf{B}\left(\mathbf{B}^{T}\mathbf{M}^{-1}\mathbf{B}+\mathbf{C}^{-1}\right)^{-1}\mathbf{B}^{T}\mathbf{M}^{-1} \qquad (3.4.5)$$

inversa $\hat{\mathbf{R}}^{-1}(k)$ poate fi calculată recursiv prin :

$$\hat{\mathbf{R}}^{-1}(k)=\frac{1}{\lambda}\left[\hat{\mathbf{R}}^{-1}(k-1)-\frac{\hat{\mathbf{R}}^{-1}(k-1)\Delta\mathbf{v}(k)\Delta\mathbf{v}^{T}(k)\hat{\mathbf{R}}^{-1}(k-1)}{\lambda+\Delta\mathbf{v}^{T}(k)\hat{\mathbf{R}}^{-1}(k-1)\Delta\mathbf{v}(k)}\right]. \qquad (3.4.6)$$

Dacă se definește câștigul de adaptare $G(k)$,

$$G(k)=\frac{\hat{\mathbf{R}}^{-1}(k-1)\Delta\mathbf{v}(k)}{\lambda+\Delta\mathbf{v}^{T}(k)\hat{\mathbf{R}}^{-1}(k-1)\Delta\mathbf{v}(k)}\ , \qquad (3.4.7)$$

relația de recurență (3.4.6) devine

$$\hat{\mathbf{R}}^{-1}(k)=\frac{1}{\lambda}\left[\hat{\mathbf{R}}^{-1}(k-1)-G(k)\Delta\mathbf{v}^{T}(k)\hat{\mathbf{R}}^{-1}(k-1)\right]. \qquad (3.4.8)$$

Prin rearanjarea relației (3.4.7) și ținându-se cont de (3.3.10) și (3.4.8), se obține

$$G(k)=\hat{\mathbf{R}}^{-1}(k)\Delta\mathbf{v}(k). \qquad (3.4.9)$$

După câteva calcule, adaptarea lui $\hat{\mathbf{A}}$ se poate pune sub forma recursivă :

[1] Acest subcapitol dă conținutul articolului (Cimponeriu, 1996).

$$\hat{\mathbf{a}}_{RLS}(k) = \hat{\mathbf{a}}_{RLS}(k-1) + \mathbf{G}(k)\left[\Delta\theta(k) - \hat{\mathbf{a}}_{RLS}(k-1)\Delta v(k)\right] = \\ = \hat{\mathbf{a}}_{RLS}(k-1) + \mathbf{G}(k)\,e(k). \tag{3.4.10}$$

Pentru inițializarea algoritmului trebuie asigurată inversabilitatea matricii $\hat{\mathbf{R}}$. Pentru aceasta, se ia (Haykin, 1991)

$$\mathbf{R}(0) = \delta\mathbf{I}, \tag{3.4.11}$$

unde δ este o constantă mică pozitivă. Se obține

$$\hat{\mathbf{R}}(k) = \lambda^k \delta\,\mathbf{I} + \sum_{p=1}^{k} \lambda^{k-p}\Delta v(p)\,\Delta v^T(p). \tag{3.4.12}$$

Se poate arăta că, cu această valoare, algoritmul RLS rămîne neschimbat (Haykin, 1991).

De asemenea, se ia

$$\hat{\mathbf{a}}_{RLS}(0) = \mathbf{0}.$$

În funcționarea algoritmului RLS se presupune că $\hat{\mathbf{R}}$ este inversabilă. Aceasta înseamnă că r, rangul matricii $\hat{\mathbf{R}}$, trebuie să fie egal cu n, dimensiunea vectorilor de intrare Δv. În cazul de față interesează situația când $r < n$, vectorii Δv aparțin unui subspațiu de dimensiune inferioară lui n. De asemenea, algoritmul Greville este valabil pentru sisteme cu ieșiri multidimensionale, în timp ce algoritmul RLS este obținut pentru sisteme cu ieșire scalară. Pe de altă parte, algoritmul RLS oferă unele avantaje în raport cu algoritmul Greville :

- Algoritmul RLS este mai simplu și necesită memorarea doar a matricii $\hat{\mathbf{R}}^{-1\,nxn}(k)$, față de matricile ΔV^{Kxn}, $\Delta V^{-\,nxK}$ și $\Delta\theta^{nxK}$, unde K, numărul vectorilor prezentați, poate fi mare.

- Există algoritmi de calcul rapid ai algoritmului RLS (Cioffi și Kailath, 1984; Michaut, 1992).

Aceste avantaje de ordin practic motivează o analiză mai detaliată a algoritmului RLS, pentru a se vedea dacă el nu poate fi extins, înlocuind algoritmul Greville.

3.4.1. Comparație teoretică

Dacă se ține seama că $\hat{\mathbf{A}}_{RLS}(k) = \hat{\mathbf{a}}^T_{RLS}(k)$ și că eroarea ieșirii este scalară, $e(k) = e^T(k)$, relația (3.4.10), poate fi rescrisă ca :

$$\hat{\mathbf{A}}_{RLS}(k) = \hat{\mathbf{A}}_{RLS}(k-1) + e(k)\mathbf{G}^T(k), \tag{3.4.10'}$$

dând forma multicanal a algoritmului RLS. Aceasta este similară cu adaptarea prin algoritmul Greville (3.3.1) :

$$\hat{\mathbf{A}}_{-}(k) = \hat{\mathbf{A}}_{-}(k-1) + \mathbf{e}(k)\mathbf{p}^T(k). \tag{3.3.1}$$

Se studiază relația dintre $\mathbf{G}(k)$ și $\mathbf{p}(k)$. Se începe cu câteva chestiuni preliminare :

În primul rând, în cele ce urmează va fi foarte avantajoasă utilizarea descompunerii după valori singulare (SVD, *Singular Value Decomposition*; v. anexa I) a matricilor $\Delta V(k-1)$ și $\Delta V^*(k-1)$:

$$\Delta \mathbf{V}(k-1)_{n\times k-1} = \mathbf{U}_{n\times n}\begin{bmatrix} diag(\sigma_i)_{i=1..r} & \mathbf{0}_{r\times k-1-r} \\ \mathbf{0}_{n-r\times r} & \mathbf{0}_{n-r\times k-1-r} \end{bmatrix}\mathbf{S}^{T}_{k-1\times k-1}, \tag{3.4.13}$$

$$\Delta \mathbf{V}^{+}(k-1)_{k-1\times n} = \mathbf{S}_{k-1\times k-1}\begin{bmatrix} diag\left(\sigma_i^{-1}\right)_{i=1..r} & \mathbf{0}_{r\times n-r} \\ \mathbf{0}_{k-1-r\times r} & \mathbf{0}_{k-1-r\times n-r} \end{bmatrix}\mathbf{U}^{T}_{n\times n}, \tag{3.4.14}$$

unde indicii reprezintă dimensiunile matricilor, $r \le min\,(n, k-1)$ este rangul lui $\Delta\mathbf{V}(k-1)$, iar $\mathbf{U}$, $\mathbf{S}$ sunt matrici ortogonale. Matricile proiecţii ortogonale din (3.2.23) devin :

$$\mathbf{P}_{\Delta V(k-1)} = \mathbf{U}_{n\times n}\begin{bmatrix} \mathbf{I}_r & \mathbf{0}_{r\times n-r} \\ \mathbf{0}_{n-r\times r} & \mathbf{0}_{n-r\times n-r} \end{bmatrix}\mathbf{U}^{T}_{n\times n}, \tag{3.4.15}$$

$$\mathbf{P}^{\perp}_{\Delta V(k-1)} = \mathbf{U}_{n\times n}\begin{bmatrix} \mathbf{0}_{r\times r} & \mathbf{0}_{r\times n-r} \\ \mathbf{0}_{n-r\times r} & \mathbf{I}_{n-r} \end{bmatrix}\mathbf{U}^{T}_{n\times n}. \tag{3.4.16}$$

În al doilea rând, se observă că, pentru $\lambda=1$ iniţializarea algoritmului RLS (3.4.11) face ca în locul estimatei matricii de corelaţie $\hat{\mathbf{R}}(k-1)$ (3.4.3) să fie calculată $\hat{\mathbf{R}}_\delta(k-1)$ (3.4.17):

$$\hat{\mathbf{R}}_\delta(k-1) = \delta\mathbf{I} + \hat{\mathbf{R}}(k-1) = \delta\mathbf{I} + \Delta\mathbf{V}(k-1)\,\Delta\mathbf{V}^{T}(k-1). \tag{3.4.17}$$

Utilizând SVD, se obţine :

$$\hat{\mathbf{R}}_\delta^{-1}(k-1) = \mathbf{U}_{n\times n}\begin{bmatrix} diag\left(\left(\sigma_i^{2}+\delta\right)^{-1}\right)_{i=1..r} & \mathbf{0}_{r\times n-r} \\ \mathbf{0}_{n-r\times r} & \delta^{-1}\mathbf{I}_{n-r} \end{bmatrix}\mathbf{U}^{T}_{n\times n}. \tag{3.4.18}$$

În utilizarea clasică a algoritmului RLS matricea de corelaţie a intrării este nesingulară, $r = n$, $\left(\Delta\mathbf{V}^{T}(k-1)\right) = 0$,

$$\hat{\mathbf{R}}^{-1}(k-1) = \left[\Delta\mathbf{V}^{T}(k-1)\,\Delta\mathbf{V}(k-1)\right]^{-1} = \mathbf{U}\left[diag\left(\sigma_i^{-2}\right)_{i=1..n}\right]\mathbf{U}^{T}, \tag{3.4.19}$$

iar iniţializarea (3.4.11) dă

$$\hat{\mathbf{R}}_\delta^{-1}(k-1) = \mathbf{U}\left[diag\left(\left(\sigma_i^{2}+\delta\right)^{-1}\right)_{i=1..n}\right]\mathbf{U}^{T}. \tag{3.4.20}$$

Cu (3.4.18), (3.4.14) şi (3.4.15) şi folosind proprietatea de idempotenţă a matricilor proiecţie, se poate scrie :

$$\begin{aligned} \hat{\mathbf{R}}_\delta^{-1}(k-1)\,\Delta\mathbf{v}(k) &= \hat{\mathbf{R}}_\delta^{-1}(k-1)\left(\mathbf{P}_{\Delta V(k-1)} + \mathbf{P}^{\perp}_{\Delta V(k-1)}\right)\Delta\mathbf{v}(k) \\ &= \hat{\mathbf{R}}_\delta^{-1}(k-1)\left(\mathbf{P}^{2}_{\Delta V(k-1)} + \mathbf{P}^{\perp 2}_{\Delta V(k-1)}\right)\mathbf{x}(k) \\ &= \mathbf{U}\Sigma_\delta\mathbf{U}^{T}\mathbf{P}_{\Delta V(k-1)}\,\Delta\mathbf{v}(k) + \delta^{-1}\mathbf{P}^{\perp}_{\Delta V(k-1)}\Delta\mathbf{v}(k) \\ &= \mathbf{U}\Sigma_\delta\mathbf{U}^{T}\,\Delta\mathbf{v}^{\parallel}(k) + \delta^{-1}\Delta\mathbf{v}^{\perp}(k). \end{aligned} \tag{3.4.21}$$

unde

$$\Sigma_\delta = \begin{bmatrix} diag\left(\left(\sigma_i^{2}+\delta\right)^{-1}\right)_{i=1..r} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{3.4.22}$$

Prin aceasta, pentru $\lambda=1$, câştigul $\mathbf{G}(k)$ al algoritmului RLS (3.4.7) devine, în urma iniţializării,

$$\mathbf{G}_\delta(k) = \frac{\mathbf{U}\,\Sigma_\delta\mathbf{U}^{T}\,\Delta\mathbf{v}^{\parallel}(k) + \delta^{-1}\Delta\mathbf{v}^{\perp}(k)}{1 + \Delta\mathbf{v}^{\parallel T}(k)\mathbf{U}\,\Sigma_\delta\mathbf{U}^{T}\,\Delta\mathbf{v}^{\parallel}(k) + \delta^{-1}\left\|\Delta\mathbf{v}^{\perp}(k)\right\|_2^{2}}. \tag{3.4.23}$$

Comparaţia între $\mathbf{p}(k)$ (3.2.18), (3.2.25) şi $\mathbf{G}(k)$ (3.4.7), (3.4.23), $\lambda=1$, poate fi acum făcută cu uşurinţă:

Dacă $\Delta\mathbf{v}^{\perp}(k)\neq\mathbf{0}$, atunci

$$\mathbf{G}(k)=\lim_{\delta\to 0}\mathbf{G}_{\delta}(k)=\frac{\Delta\mathbf{v}^{\perp}(k)}{\left\|\Delta\mathbf{v}^{\perp}(k)\right\|_2^2}=\mathbf{p}(k).$$

Dacă $\Delta\mathbf{v}^{\perp}(k)=\mathbf{0}$, atunci $\mathbf{p}(k)$ este, folosind (3.4.14) :

$$\mathbf{p}(k)=\frac{\mathbf{U}\begin{bmatrix}diag\left(\sigma_i^{-2}\right)_{i=1,r} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\mathbf{U}^{T}\Delta\mathbf{v}^{\|}(k)}{1+\Delta\mathbf{v}^{\|}(k)\,\mathbf{U}\begin{bmatrix}diag\left(\sigma_i^{-2}\right)_{i=1,r} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\mathbf{U}^{T}\Delta\mathbf{v}^{\|}(k)},\qquad(3.4.24)$$

şi deci $\mathbf{G}(k)=\lim_{\delta\to 0}\mathbf{G}_{\delta}(k)=\mathbf{p}(k)$.

Aceasta duce la concluzia teoretică : Algoritmul RLS este valabil şi când componentele vectorilor de intrare $\Delta\mathbf{v}$ sunt dependente liniar iar estimata matricii lor de corelaţie $\hat{\mathbf{R}}$ (3.4.3) poate fi inversată doar sub forma (3.4.12), dacă $\delta\to 0$.

3.4.2. Rezultate de simulare şi concluzie

Comparaţia teoretică între algoritmul RLS şi algoritmul Greville a fost verificată prin simulări, efectuate în aceleaşi condiţii cu cele din § 3.3.2. Simulările au urmărit învăţarea aceleiaşi matrici, $\mathbf{A}$ = [1.0 0.3 0.4 0.2;0.2 1.0 -0.5 0.2;0.6 -0.2 0.5 0.7]. Pentru fiecare simulare sunt prezentate eroarea apriori a ieşirilor, $\|\mathbf{e}(k)\|$, şi eroarea aposteriori a soluţiei, $\|\mathbf{A}-\hat{\mathbf{A}}(k)\|_2$.

În prima simulare, prezentată în figura 3.4.1, s-a făcut învăţarea matricii $\mathbf{A}$ în absenţa zgomotului. În figura 3.4.1 parametrul iniţializării RLS (3.4.11) este $\delta = 10^{-16}$. În limita preciziei de calcul, erorile sunt identice. Punctele ce nu apar pe curba RLS corespund unei erori nule, nereprezentabile pe scara logaritmică.

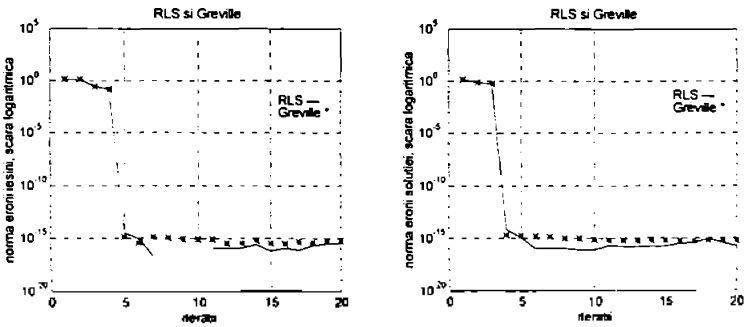


Figura 3.4.1. Învăţarea unei matrici prin algoritmul Greville şi prin algoritmul RLS. cu $\delta = 10^{-16}$. a) norma erorii ieşirii b) norma erorii soluţiei

Eroarea datorată inițializării algoritmului RLS printr-un δ insuficient de mic, $\delta = 10^{-5}$, poate fi observată în figura 3.4.2.

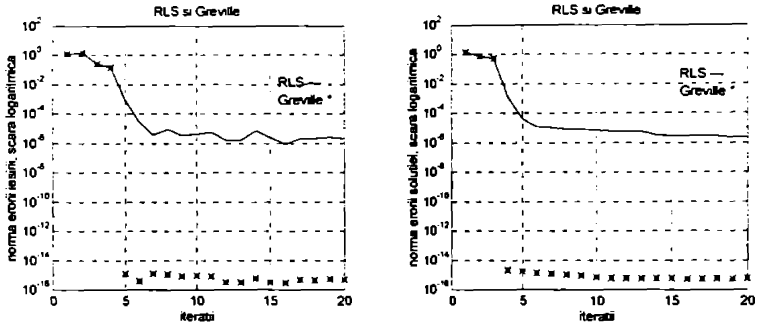


Figura 3.4.2. Eroarea datorată inițializării algoritmului RLS cu $\delta = 10^{-5}$.
a) norma erorii ieșirii b) norma erorii soluției

Trebuie însă menționat că o valoare foarte mică, precum $\delta=10^{-14}$, poate cauza instabilitatea algoritmului RLS. Într-o simulare cuprinzând 1000 de iterații, valoarea $\delta=10^{-12}$ nu a ridicat probleme. Dar eroarea datorată inițializării algoritmului RLS poate fi neglijată în prezența zgomotului. Aceasta se poate observa în figura 3.4.3, unde efectul lui $\delta=10^{-3}$ este total mascat de zgomotul $\sigma=10^{-2}$.

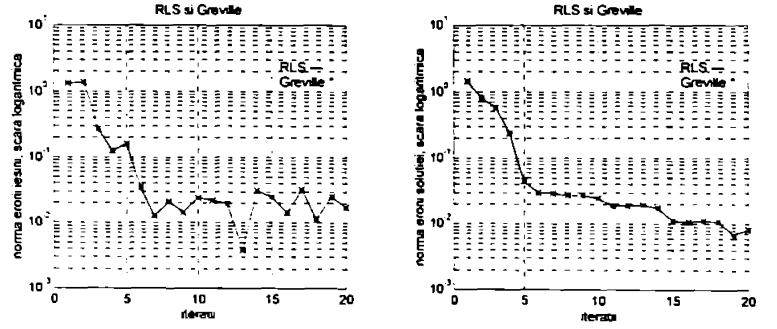
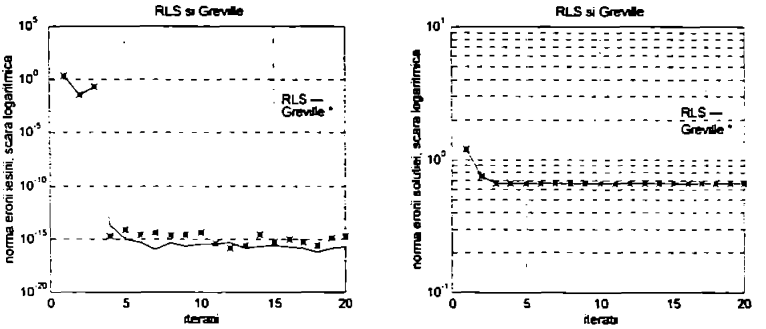


Figura 3.4.3. Mascarea erorii datorate inițializării algoritmului RLS cu $\delta = 10^{-3}$, prin zgomotul datelor de intrare. $\sigma=10^{-2}$.
a) norma erorii ieșirii b) norma erorii soluției

În fine, figura 3.4.4 prezintă comportarea celor doi algoritmi pentru vectori de intrare, de dimensiune 4, aparținând unui subspațiu de dimensiune 3. Datele sunt fără zgomot, iar pentru RLS. $\delta=10^{-16}$. Matricea A este învățată după 3 iterații.

În concluzie, algoritmul Greville și algoritmul RLS sunt echivalente dacă δ, parametrul inițializării RLS, este suficient de mic în raport cu precizia de calcul sau cu zgomotul datelor. Se poate beneficia simultan de generalitatea soluției dată de pseudoinversa

Moore-Penrose, şi de simplitatea şi eficienţa algoritmului RLS. Soluţiile obţinute de ambii algoritmi, atunci când vectorii de intrare sunt liniar dependenţi, sunt deplasate.



a. b.

Figura 3.4.4. Algoritmul Greville şi algoritmul RLS, $\delta=10^{-16}$, pentru vectori de intrare de dimensiune 4 aparţinând unui subspaţiu de dimensiune 3.
a) eroarea ieşirii b) eroarea soluţiei.

Prin această echivalenţă s-a demonstrat posibilitatea extinderii algoritmului RLS pentru cazul în care vectorii de intrare nu au componente liniar independente. În mod tradiţional algoritmul RLS este utilizat doar atunci când matricea de corelaţie a vectorilor de intare este nesingulară. adică pentru vectori de intrare liniar independenţi. În acest caz, soluţia obţinută este o estimare nedeplasată a vectorului sau matricii de regresie (Haykin, 1991). Extensia demonstrată aici a algoritmului RLS este aplicabilă şi pentru date având matricea de corelaţie singulară. În acest caz se obţine este o estimare deplasată a matricii căutate.

4. Comandă cinematică local liniară cu învăţare rapidă şi memorare adaptivă

În cele ce urmează se consideră sistemul prezentat în subcapitolul 2.1, figura 2.2.1. Acesta cuprinde braţul de robot ERICC şi sistemul vizual prezentat în §§ 2.1.1-2.1.2. Comanda braţului se face pe baza modelării local liniare (LLM, *Local Linear Mappings*), ca în § 2.2.1, figurile 2.2.4 şi 2.2.5. Toate experimentele sunt realizate prin simulare, utilizând pentru braţ modelul simplificat din § 2.1.1, figura 2.1.2 ecuaţia (2.1.1), iar pentru cele două camere, modelul din § 2.1.2, figura 2.1.4 şi ecuaţiile (2.1.9), (2.1.11). Se consideră cazul unei ţinte punctiforme, interesând doar poziţia ei şi a braţului. Acestea pot fi caracterizate prin vectori tridimensionali în spaţiul geometric. În spaţiul vizual aceştia au dimensiunea 4, dar imaginile lor variaţi aparţin unui subspaţiu liniar de dimensiune 3. Pentru a se putea pune în evidenţă argumentele unor funcţii, faţă de notaţiile din § 2.2.1-2.2.3, unde numărul iteraţiei a fost notat ca argument, în continuare numărul iteraţiei va apare ca indice. De exemplu, vectorul vizual la iteraţia i, v(i), va fi în continuare notat $\mathbf{v}_i$, iar $\mathbf{v}(\mathbf{r}_i)$ va reprezenta vectorul vizual în funcţie de poziţia carteziană la iteraţia i, $\mathbf{r}_i$. De asemenea, ţinta vizuală temporară notată în § 2.2.1 $\mathbf{t}_{VT}(i)$ va fi notată simplu $\mathbf{t}_i$, iar ţinta finală va fi notată $\mathbf{t}_f$.

În § 1.2.3 au fost prezentate realizările întâlnite în bibliografie, de reţele neuronale LLM. La toate[1], învăţarea matricii jacobiene sau a inversei sale, pentru care eroarea pătratică medie a ieşirilor este minimă, se face prin algoritmul LMS. Datorită viziunii stereoscopice, vectorii de intrare în blocul de comandă neuronală sunt liniar dependenţi. Aceasta face ca soluţia de eroare pătratică să poată fi obţinută prin algoritmul Greville prezentat în § 3.2. Aşa cum s-a arătat în § 3.3, viteza sa de adaptare este net superioară algoritmului LMS. De asemenea, în § 3.4 s-a demonstrat că în locul algoritmului lui Greville poate fi utilizat algoritmul RLS, dacă parametrul δ al iniţializării este suficient de mic. Aşadar se propune ca pentru învăţarea matricii comenzii **Jc**, în locul algoritmului LMS să fie folosit algoritmul RLS. Se propune şi actualizarea acestei matrici pe baza informaţiei câştigate pe parcursul deplasării braţului spre ţintă. De asemenea, se propune un algoritm, ce corespunde unei structuri de reţea neuronală, ce învaţă **Jc** doar acolo unde este necesar.

4.1. Învăţare totală în fiecare poziţie [2]

Modul cel mai simplu conceptual de realizare a comenzii cinematice local liniare adaptive este prin învăţarea matricii **Jc** în fiecare din poziţiile succesive ale braţului. Informaţia necesară este disponibilă prin valoarea comenzilor unghiulare date $\Delta\theta_i$ şi de valorile deplasărilor corespunzătoare $\Delta\mathbf{v}_i$ date de camerele video.

[1] Cu excepţia amestecului de experţi din (Schaal şi Atkeson, 1995), articol întâlnit după redactarea acestui capitol, unde se foloseşte algoritmul RLS cu uitare (λ<1).

[2] Rezultatele din acest subcapitol şi din următorul alcătuiesc lucrarea propusă (Cimponeriu şi Kihl, 1997).

Învățarea LLM prin RLS se poate face, în acest caz, în exact 3 iterații. În fiecare poziție se învață $\mathbf{Jc}$ prin 3 tatonări după direcții alese convenabil sau aleatoare, liniar independente. Cu aproximarea obținută $\tilde{\mathbf{Jc}}$ se obține o comandă unghiulară și se face un pas, ajungându-se în poziția următoare, unde se face din nou învățarea matricii $\mathbf{Jc}$. Algoritmul de comandă adaptivă este următorul :

a. Se cunoaște poziția țintei finale dată de sistemul vizual, $\mathbf{t}_f$. Se cunoaște poziția inițială a brațului în coordonate unghiulare θ_0 și vizuale $\mathbf{v}_0$. Se inițializează poziția curentă θ_i , $\mathbf{v}_i$ cu aceste valori.

b. Eroarea de poziționare față de ținta finală, dată de sistemul vizual, este $\mathbf{m}_i=\mathbf{t}_f-\mathbf{v}_i$. Se calculează poziția țintei temporare curente $\mathbf{t}_i$, astfel încât ea să se afle în direcția lui $\mathbf{t}_f$, la o distanță mai mică sau egală cu $\|\Delta\mathbf{v}_d\|_{max}$. Ea dă deplasarea dorită, $\Delta\mathbf{v}_{d,i} = \mathbf{t}_i - \mathbf{v}_i$.

c. Se obține o aproximare a matricii de comandă $\tilde{\mathbf{Jc}}$: se comandă succesiv 3 mici deplasări unghiulare $\Delta\theta_k$ corespunzătoare fiecăreia din articulațiile brațului, și se observă deplasările corespunzătoare $\Delta\mathbf{v}_k$, k=1...3. Cu datele obținute se calculează $\tilde{\mathbf{Jc}}$ prin algoritmul RLS.

d. Cu aproximarea disponibilă $\tilde{\mathbf{Jc}}$ se calculează comanda unghiulară pentru brațul de robot, $\Delta\theta_i= \tilde{\mathbf{Jc}}\,\Delta\mathbf{v}_{d,i}$.

e. Brațul de robot se deplasează în noua poziție $\mathbf{r}_{i+1}=\mathbf{r}(\theta_i+\Delta\theta_i)$, conform (2.1.1). Imaginea noii poziții pe ecranele CCD ale sistemului vizual este $\mathbf{v}_{i+1}=\mathbf{v}(\mathbf{r}_{i+1})$. (2.1.2). Aceasta corespunde observării unei deplasări $\Delta\mathbf{v}_i=\mathbf{v}_{i+1}-\mathbf{v}_i$. Datorită erorii de model și de comandă, ea diferă de $\Delta\mathbf{v}_{d,i}$. Noua eroare de poziționare este $\mathbf{m}_{i+1}=\mathbf{t}_f-\mathbf{v}_{i+1}$

f. Dacă eroarea de poziționare $\mathbf{m}_{i+1}$ nu este suficient de mică, se reia de la pasul b.

(4.1.1)

În figura 4.1.1 este prezentată o simulare cu acest algoritm, pentru deplasări vizuale mici, de 10 pixeli/iterație. Ca pentru toate simulările prezentate în acest paragraf și în următorul, sunt ilustrate poziția brațului, eroarea de poziționare, poziția cleștelui robotului și a țintei pe ecranele CCD, și valorile unghiurilor articulațiilor brațului. Brațul este reprezentat prin imaginea în planul vertical propriu și prin poziția acestui plan, dată prin proiecția lui pe orizontală sau prin vederea de sus a brațului. Eroarea de poziționare este reprezentată atât în coordonate liniare cât și logaritmice, pentru observarea comportării pentru erori de poziționare mai mari, respectiv mai mici decât $\|\Delta\mathbf{v}_d\|^{-1}_{max}$, cum s-a arătat în § 2.4.4. Eroarea de poziționare finală, pentru care algoritmul se oprește, este 10^{-4} pixeli valoare mult mai mică decât cea obtenabilă în mod practic, care poate fi de ordinul pixelului; aceasta se face pentru efectuarea și observarea mai multor iterații în apropierea țintei. Cât privește camerele CCD, imaginile amândurora sunt reprezentate pe aceeași figură, ele putând fi distinse prin notația B sau C plasată în dreptul pozițiilor inițiale. Pe imaginile brațului și ale camerelor, pozițiile inițiale sunt reprezentate prin o, pozițiile obținute la fiecare iterație prin x, iar țintele prin +. Pentru simularea reprezentată în figura 4.1.1 și în următoarele, cele două imagini ale țintei sunt

întâmplător aproape coincidente. Pe graficele din dreapta sus, pentru a fi mai ușor vizibile, valorile erorii de poziționare și ale unghiurilor, reprezentate prin · , sunt interpolate liniar între valorile discrete corespunzătoare iterațiilor. Se remarcă faptul că traiectoriile obținute pe ecranele CCD sunt drepte. Aceasta arată că comanda este fără erori, iar modelul local liniar este valabil cu o bună aproximare. Aceeași traiectorie se obține și dacă în fiecare punct se face o învățare RLS (4.1.1.c) cu câte 3 vectori $\Delta\theta_k$ având direcții aleatoare: și în acest mod se obține o matrice $\tilde{J}c$ fără erori. Faptul că direcțiile pot fi aleatoare permite efectuarea a doar două tatonări, informația corespunzătoare celei de-a treia fiind dată de deplasarea care tocmai a fost efectuată spre țintă.

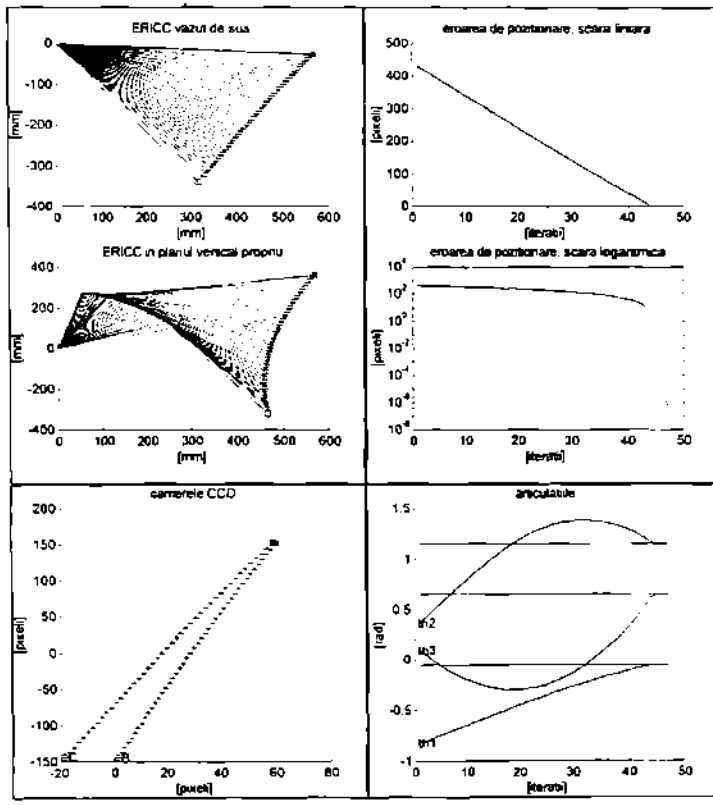


Figura 4.1.1. Simularea algoritmului (4.1.1) pentru $\theta_0 = [-0.8178\ 0.3627\ 0.0934]^T$ și $\theta_f = [-0.0473\ 1.1520\ 0.6470]^T$. Pași mici, de 10 pixeli.

În simularea prezentată în figura 4.1.2 învățarea se face ca în figura 4.1.1, dar pașii sunt mai mari, de cca. 100 de pixeli. Traiectoriile vizuale nu mai sunt drepte. Întrucât, așa cum a arătat figura 4.1.1, $\tilde{J}c$ este fără erori, aceasta înseamnă că modelarea local liniară este doar aproximativă, lungimea unui pas fiind, de această dată, mare. Poziționarea este mult mai rapidă.

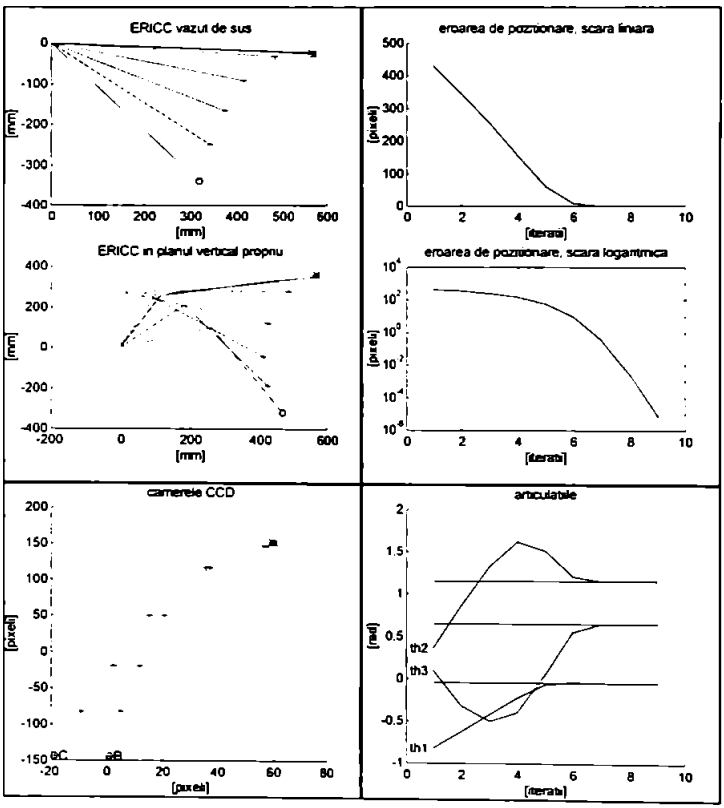


Figura 4.1.2. Învățare RLS exactă. Pași de 100 de pixeli.

Din comparațiile dintre algoritmul LMS, Greville și RLS rezultă că matricea **Jc** este învățată mult mai puțin eficient prin LMS decât prin RLS. Figura 4.1.3 ilustrează acest fapt prezentând o simulare realizată în aceleași condiții ca cea din figura 4.1.1, dar pentru care, înaintea fiecărui pas, învățarea matricii **Jc** (4.1.1 c) se face prin algoritmul LMS aplicat datelor obținute prin 10 mici deplasări aleatoare. Spre deosebire de figura 4.1.1, deplasarea vizuală obținută nu este liniară, arătând că $\tilde{\mathbf{J}}\mathbf{c}$ este eronată.

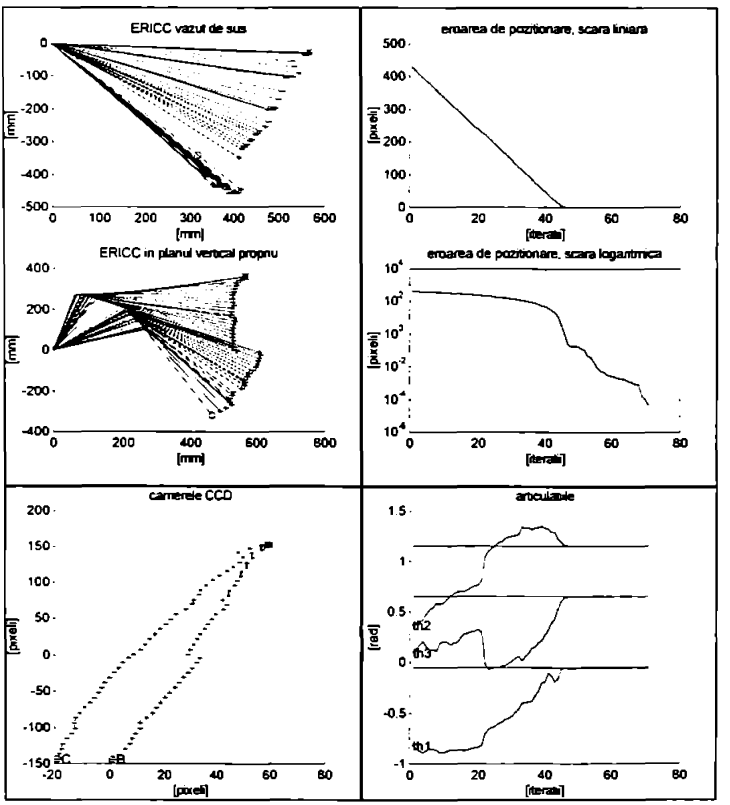


Figura 4.1.3. Matricea Jc învățată prin câte 10 mici deplasări aleatoare și LMS

În concluzie, utilizarea algoritmului RLS este net superioară algoritmului LMS prin numărul mai mic (mai exact, minim) de tatonări ale brațului, și prin precizia rezultatului obținut, ce se manifestă în liniaritatea traiectoriei în spațiul vizual.

4.2. Adaptare parțială în fiecare poziție

În paragraful precedent, în fiecare punct $\tilde{\mathbf{J}}\mathbf{c}$ a fost învățată integral, algoritmii RLS și LMS fiind inițializați cu $\tilde{\mathbf{J}}\mathbf{c} = \mathbf{0}$. Dacă se evită singularitățile[1], transformarea cinematică inversă este continuă și $\mathbf{J}\mathbf{c}$ nu diferă mult între poziții apropiate. Proprietatea de continuitate poate fi folosită pentru inițializarea matricii $\tilde{\mathbf{J}}\mathbf{c}$ cu valoarea din poziția precedentă, și actualizarea acestei valori.

Informația minimă pentru actualizare se poate obține printr-o singură deplasare a brațului. Deplasarea poate avea o direcție precizată sau aleatoare. Mai avantajos arv fi primul caz, când direcția ar fi cea a țintei: deplasarea în direcția țintei se face în orice caz, iar mișcarea robotului nu ar fi perturbată de micile deplasări necesare învățării jacobienei.

Se observă că, dacă învățarea se face pe baza unui singur vector de intrare, algoritmii RLS și LMS optimizat sunt echivalenți: Conform § 3.4.1 algoritmul RLS este echivalent cu algoritmul Greville. Pe de altă parte, dacă vectorii de intrare prezentați algoritmului LMS optimizat sunt ortogonali, conform § 3.3.1, algoritmul Greville și algoritmul LMS sunt echivalenți. Dar primul vector prezentat algoritmului LMS, $\Delta\mathbf{v}(1)$, este ortogonal pe precedenții (subspațiul generat de vectorii prezentați anterior este nul). Așadar, fiind ambii echivalenți cu algoritmul Greville, pentru $\Delta\mathbf{v}(1)$ algoritmii LMS și RLS sunt echivalenți. Actualizarea va fi făcută așadar prin algoritmul LMS, mai simplu.

În continuare se prezintă câteva simulări, în care la fiecare pas $\tilde{\mathbf{J}}\mathbf{c}$ este inițializată cu valoarea precedentă și actualizată pe baza unei singure deplasări. În poziția de pornire inițializarea se face fie cu 0, fie aleator, fie cu valoarea exactă. Reprezentările se fac în același mod ca în simulările precedente.

[1] Pozițiile pentru care jacobiana transformării cinematice devine singulară (Craig, 1986).

În figura 4.2.1 $\tilde{J}c$ este inițializată cu 0 și apoi actualizată printr-o o mică tatonare de direcție aleatoare. Pe imaginile CCD se observă că, mai ales la începutul mișcării, $\tilde{J}c$ nu este corectă, ducând la deplasări de direcție diferită de cea a țintei. Pe parcurs, chiar cu viteza scăzută de convergență a algoritmului LMS, se obține o valoare $\tilde{J}c$ destul de bună. În apropierea țintei, funcția $Jc(\theta)$ este mai neliniară și mișcarea devine din nou ușor dezordonată. Totuși, datorită direcțiilor aleatoare de tatonare (nereprezentate în figură), întrucât transformarea cinematică nu variază rapid, $\tilde{J}c$ ajunge să fie învățată suficient de bine pentru a permite poziționarea dorită.

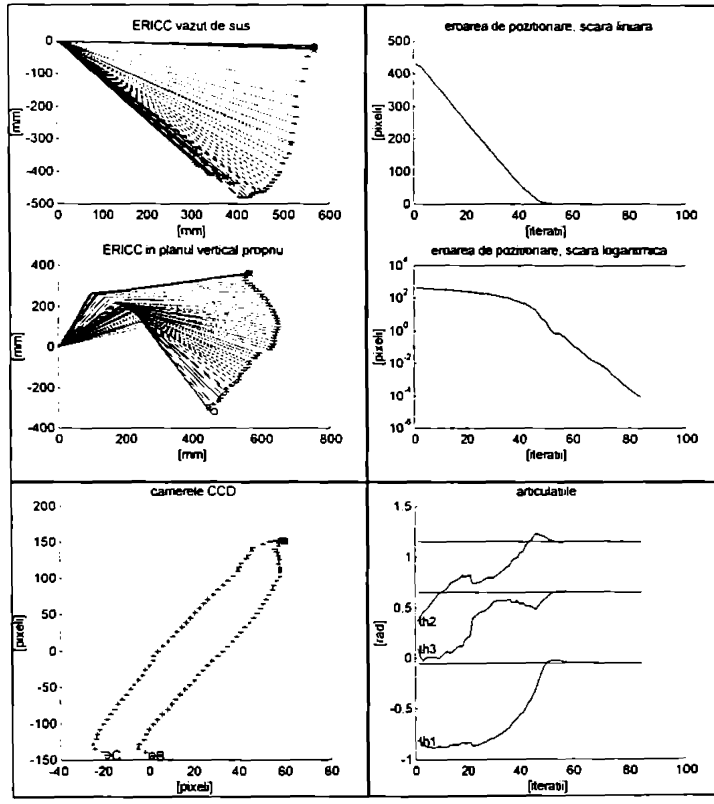


Figura 4.2.1. Actualizarea $\tilde{J}c$ printr-o direcție aleatoare la fiecare pas, la deplasarea între pozițiile date de $\theta_0 = [-0.82\ 0.36\ 0.09]^T$, $\theta_f = [-0.05\ 1.15\ 0.65]^T$

Dacă actualizarea se face doar pe baza deplasărilor spre țintă, comportarea depinde de inițializarea $\tilde{J}c$ la începutul mișcării. Pentru simularea din figura 4.2.2 $\tilde{J}c$ este inițializată cu 0, iar pentru figura 4.2.3, $\tilde{J}c$ este inițializată cu valori aleatoare. Figura 4.2.2 sugerează că în cazul inițializării $\tilde{J}c$ cu 0 este posibil ca mișcarea să se oprească. Sunt reprezentate doar 95 de iterații; în realitate, după încă câteva, brațul reîncepe să se miște, spre țintă. În figura 4.2.3, în cele din urmă poziționarea reușește, dar mișcarea este practic necontrolată.

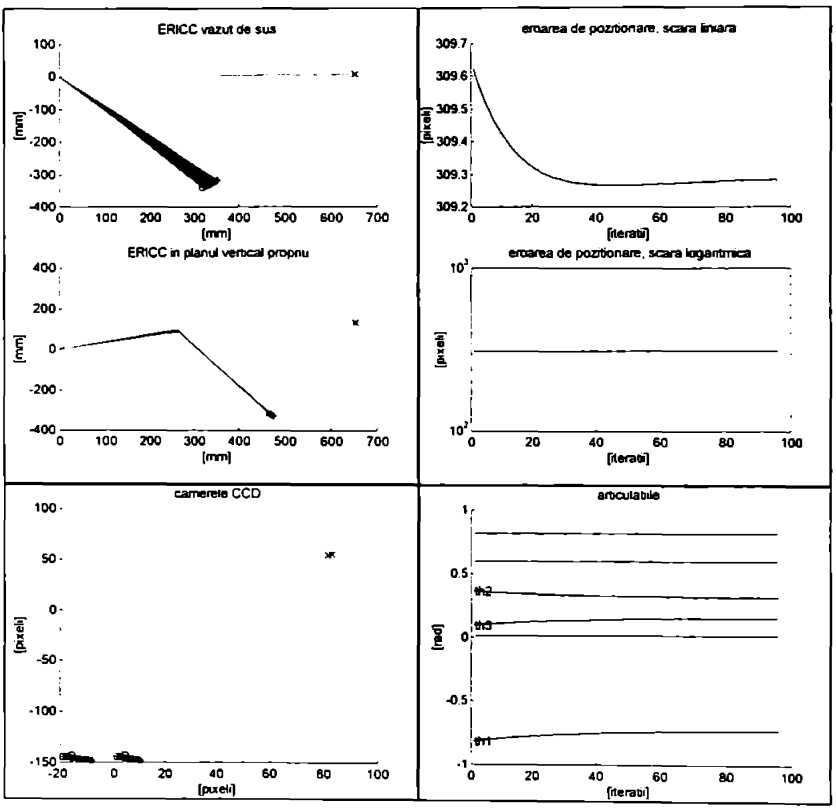


Figura 4.2.2. Actualizarea $\tilde{J}c$ pe baza deplasării în direcția țintei. Inițializare cu zero.

Dacă inițializarea matricii $\tilde{J}c$ este aleatoare, atunci mișcarea poate fi haotică, ca în figura 4.2.3.

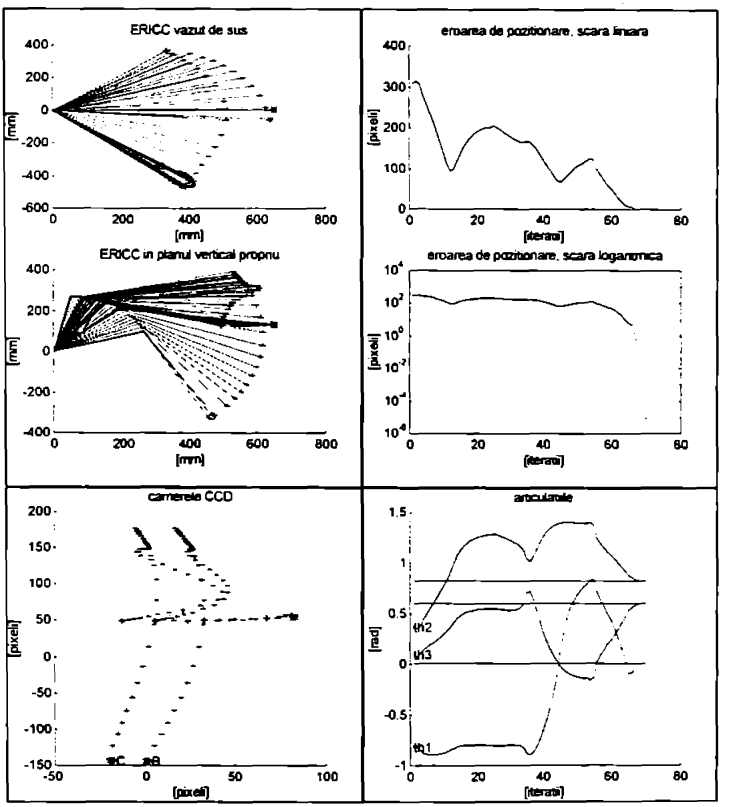


Figura 4.2.3. Actualizarea $\tilde{J}c$ pe baza deplasării în direcția țintei. Inițializare aleatoare.

Dacă la începutul mișcării $\tilde{J}c$ nu este inițializată aleator, ci cu valoarea exactă pentru punctul de pornire (obținută, spre exemplu, prin trei tatonări și algoritmul RLS), atunci poziționarea se poate efectua, mișcarea tipică la actualizarea doar după direcția deplasării fiind cea din figura 4.2.4.

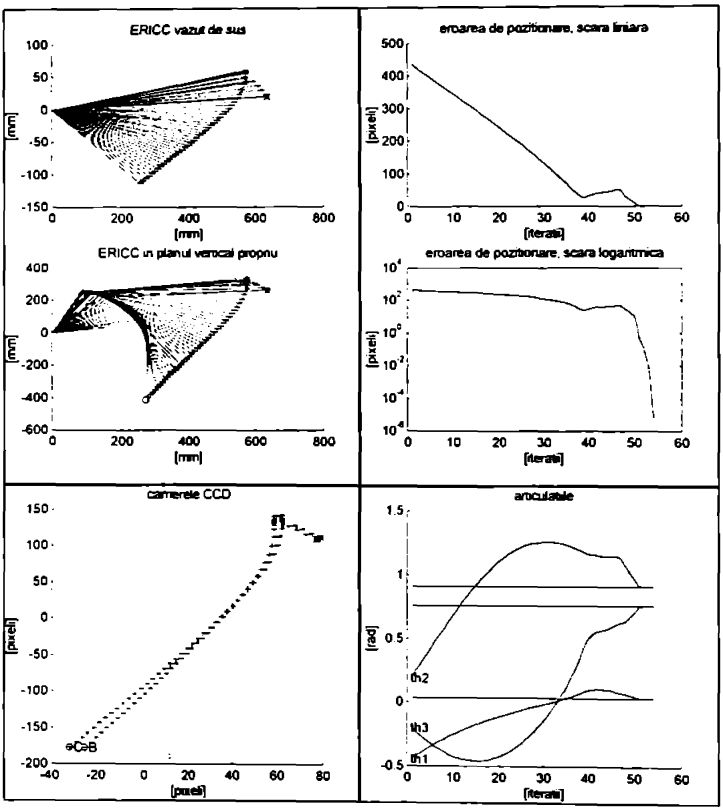


Figura 4.2.4. $\tilde{J}c$ inițializată cu valoarea exactă și actualizată după direcția deplasării. Deplasarea este dată de $\theta_0=[-0.44\ 0.19\ -0.2]^T$, $\theta_f=[0.03\ 0.91\ 0.76]^T$

Este însă important ca actualizarea $\tilde{J}c$ să fie făcută: se poate întâmpla ca, în absența actualizării, poziționarea să nu poată fi efectuată. Chiar dacă este mai degrabă un exemplu spectaculos și nu unul tipic, cazul din figura 4.2.5 ilustrează acest lucru.

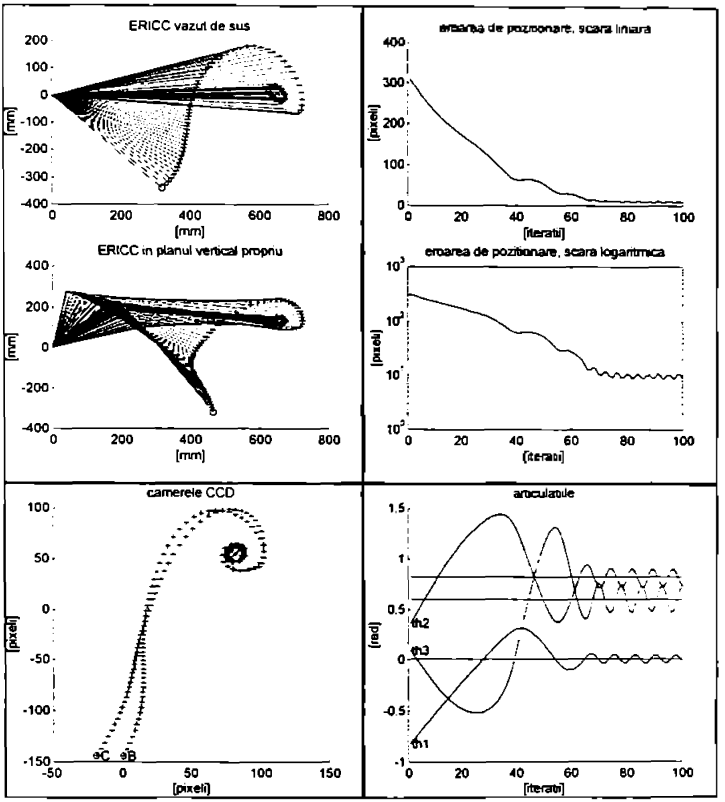


Figura 4.2.5. Matricea $\tilde{J}c$ inițializată cu valoarea exactă, dar neactualizată pe parcursul deplasării. θ_0 și θ_f ca în figura 4.2.1.

Să observăm că, pentru actualizarea de direcție dată, ceea a țintei, teoretic mișcarea se poate opri :

Conform algoritmului (4.1.1) și figurilor 2.2.6 și 2.2.7, dacă se dorește deplasarea vizuală $\Delta v_d(k)$, cu $\Delta v_d(k) \in \mathcal{R}(Jv)$ (ceea ce înseamnă că deplasarea dorită Δv_d poate fi și obținută), se dă comanda unghiulară $\Delta\theta(k+1)$ (4.2.1) și se obține deplasarea $\Delta v(k+1)$ (4.2.2):

$$\Delta\theta(k+1) = \tilde{J}c(k-1)\, \Delta v_d(k) \qquad (4.2.1)$$

$$\Delta v(k+1) \cong \mathbf{Jv}\,\mathbf{Jr}\,\Delta\theta(k+1) \tag{4.2.2}$$

unde egalitatea este doar aproximativă întrucât a fost considerat modelul liniarizat al brațului și al sistemului vizual, și $\mathbf{Jv}\in\mathbf{R}^{4\times3}$, $\mathbf{Jr}\in\mathbf{R}^{3\times3}$, $\tilde{\mathbf{Jc}}\in\mathbf{R}^{3\times4}$. Dacă se presupun deplasări suficient de mici, atunci jacobienele nu depind de θ. Fie descompunerea (4.2.3):

$$\Delta v_d(k) = \alpha\,\Delta v(k) + \beta\,\Delta v^{\perp}(k), \tag{4.2.3}$$

unde $\Delta v^{\perp}(k)$ este ortogonal pe $\Delta v(k)$, și de lungime unitate. Dacă actualizarea matricii $\tilde{\mathbf{Jc}}$ se face prin LMS optimizat, după direcția lui $\Delta v(k)$, atunci această direcție este învățată corect și se obține (4.2.4).

$$\Delta v(k+1) = \alpha\,\Delta v(k) + \beta\,\mathbf{Jv}\,\mathbf{Jr}\,\hat{\mathbf{Jc}}_{LMS}\,\Delta v^{\perp}(k) \tag{4.2.4}$$

Dacă $\hat{\mathbf{Jc}}_{LMS}$ diferă de valoarea corectă, produsul $\mathbf{Jv}\,\mathbf{Jr}\,\hat{\mathbf{Jc}}_{LMS}$ diferă de matricea proiecție pe spațiul coloanelor lui **Jv**, $\mathcal{R}(\mathbf{Jv})$, și al doilea termen din (4.2.4) diferă de al doilea termen din (4.2.3). Aceasta înseamnă că după actualizarea LMS, deplasarea obținută nu coincide cu cea dorită, sau mișcarea este dezordonată, ea nefăcându-se în direcția dorită nici după adaptarea LMS. În plus, așa cum s-a arătat în § 3.1, actualizarea LMS poate altera aproximarea deja disponibilă a matricii **Jc**.

Dacă presupunem că matricea $\tilde{\mathbf{Jc}}$ este deficientă de rang și $\Delta v^{\perp}(k)$ aparține spațiului său nul, atunci cel de-al doilea termen din (4.2.4) se anulează. Dacă Δv_d este constant, atunci

$$\Delta v(k+m) = \alpha^m \Delta v(k) = \alpha^{m-1}\Delta v_d, \tag{4.2.5}$$

și dacă $\alpha<1$, în cele din urmă mișcarea se oprește. Aceasta se întâmplă în special dacă **Jc** este inițializat cu 0.

Practic, chiar dacă mișcarea nu se va opri, datorită abaterilor de la aproximarea liniară făcută și a dependenței de θ a jacobienelor, totuși ea va fi mult încetinită, ca în figura 4.2.2.

Așadar actualizarea LMS este necesară, dar nu și suficientă, ea neputând garanta efectuarea poziționării, în special dacă $\tilde{\mathbf{Jc}}$ este deficientă de rang. Această situație poate fi eliminată în mai multe moduri :

1. Se evită ca $\tilde{\mathbf{Jc}}$ să fie deficientă de rang, de exemplu prin însumarea de zgomot care eventual poate fi proporțional cu distanța rămasă până la țintă. Acest procedeu duce însă la o mișcare neregulată, ce se dorește a se evita.
2. Pentru $\tilde{\mathbf{Jc}}$ actualizată, se testează dacă deplasarea obținută este suficient de bună: de exemplu se calculează cosinusul unghiului dintre deplasarea dorită Δv_d și cea obținută Δv; dacă acesta este mai mic decât o anumită valoare, cum ar fi 0.94 (cos 20°), se reînvață $\tilde{\mathbf{Jc}}$, de exemplu prin 3 tatonări și RLS. (4.2.6)

   Figura 4.2.6 ilustrează efectul acestui procedeu asupra deplasării pentru aceeași poziție inițială și țintă ca pentru figura 4.2.4.
3. S-ar putea ortogonaliza vectorii Δv consecutivi, prin revenirea la algoritmul RLS. Aceasta ar corespunde algoritmului RLS pentru semnale nestaționare: prin utilizarea unui factor de uitare $\lambda<1$, se poate învăța aproximarea liniară valabilă doar local: efectul

vectorilor Δv, $\Delta\theta$ corespunzând iterațiilor, deci și pozițiilor îndepărtate. scade exponențial cu λ. Această abordare ar presupune eventual adaptarea lui λ în funcție de neliniaritatea transformării de aproximat și de lungimea pasului. Însă câteva simulări efectuate cu mai multe valori pentru λ nu au dat satisfacție: mișcarea obținută nu este netedă, existând cazuri în care poziționarea spre țintă nu converge.

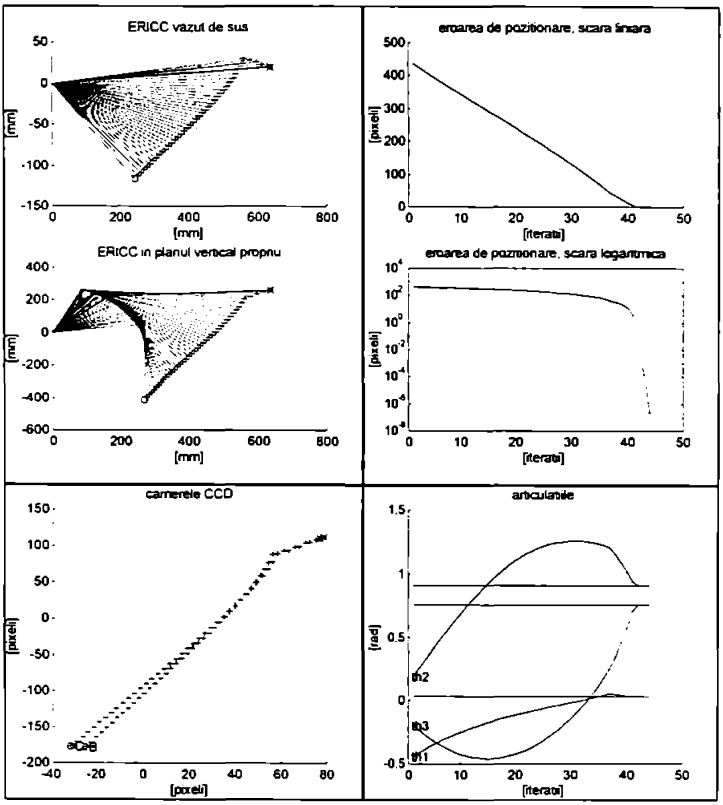


Figura 4.2.6. Învățarea $\tilde{\mathbf{J}}\mathbf{c}$ pentru $cos(\Delta v_d, \Delta v) < 0{,}94$; cu θ_0 și θ_f ca în figura 4.2.1.

Concluzii :

Continuitatea transformării cinematice inverse face ca să nu fie necesară la fiecare pas învățarea în totalitate a matricii $\tilde{\mathbf{J}}\mathbf{c}$. Efectuarea câte unei tatonări de direcție aleatoare permite învățarea chiar în lipsa oricărei inițializări, dar are ca efect obținerea unei mișcări dezordonate. Dacă $\mathbf{Jc}$ este inițializată corect, se poate folosi pentru actualizare doar informația obținută din deplasarea spre țintă. Însă în acest caz nu poate fi garantată poziționarea în ținta dorită. Aceasta impune utilizarea unor criterii de sesizare a corectitudinii mișcării, și de reînvățare totală, atunci când este cazul, a matricii comenzii.

4.3. Memorare adaptivă

Paragraful precedent a arătat că este posibilă poziționarea brațului de robot printr-o mișcare netedă pe porțiuni, în care $\tilde{J}c$ este actualizată folosind informația deplasării doar în direcția țintei. Această informație este însă, atât în principiu cât și practic, insuficientă, astfel încât uneori este necesară reinvățarea totală a matricii $\tilde{J}c$. Pentru aceasta, în aceste puncte trebuie efectuate câteva tatonări, numărul lor minim fiind dat de dimensiunea spațiului în care are loc mișcarea, în acest caz 3. Se pot imagina scheme în care să se încerce învățarea parțială a $\tilde{J}c$ pe o câte o direcție eventual ortogonală pe direcțiile deja învățate, prin RLS, și verificarea rezultatului obținut; însă această cale nu a fost urmărită în această lucrare, întrucât s-a apreciat că raportul, dat de câștigul prin numărul mai mic de mișcări / complexitatea algoritmului, nu este favorabil.

Așa cum s-a prezentat în § 1.2.3, Ritter *et al.* (1989, 1992), apoi Walter și Schulten (1993) și Hasselroth *et al.* (1994) învață și memorează $\tilde{J}c$ pentru întreg spațiul de lucru, în puncte date de o hartă Kohonen. După învățare (care necesită câteva mii de iterații, cu mișcări aleatoare), comanda este obținută exclusiv prin valorile $\tilde{J}c$ memorate. Aceeași metodă este preluată de Kuhn *et al.* (1995) și Gresser *et al.* (1996). Fritzke (1995) utilizează și el aproximări local liniare pentru aproximarea funcțiilor.

4.3.1. Algoritm de memorare adaptivă cu învățare rapidă

Se propun aici trei îmbunătățiri ale metodei lui Ritter *et al.* Prima se bazează pe observația că un braț de robot efectuează în general mișcări repetitive într-un anumit subspațiu al spațiului total de lucru. Aceasta face ca să fie suficient, ca la rețeaua CMAC (§ 1.2.3), a se învăța valorile $\tilde{J}c$ doar în zona de interes. Dacă în plus algoritmul este capabil să învețe rapid și $\tilde{J}c$ dintr-o altă zonă, atunci această abordare nu are nici un inconvenient.

A doua îmbunătățire constă în a memora $\tilde{J}c$ într-un număr cât mai restrâns de puncte, minimul necesar pentru a obține o anumită performanță. Criteriul care asigură numărul minim de astfel de puncte este cel al convergenței poziționării, studiat în capitolul 2. Un alt criteriu poate fi cel de netezime a traiectoriei, ca în (4.2.6). Amplasarea acestor puncte se face adaptiv, acolo unde este necesar.

A treia îmbunătățire constă în a utiliza, pentru calculul $\tilde{J}c$, și informația disponibilă *on-line*, prin actualizarea LMS prezentată în § 4.2.

Se obține astfel o schemă de comandă derivată din (4.1.1), în care valorile măsurate ale $\tilde{J}c$ se memorează, iar învățarea (măsurarea prin 3 mișcări de test și calcul RLS) se face doar atunci când este necesar. În plus, valorile memorate ale $\tilde{J}c$ pot fi actualizate pe parcursul deplasării, prin LMS. Este de așteptat ca prin această adaptare numărul de puncte pentru care este necesară memorarea să fie destul de restrâns.

Bazat pe schema de comandă prezentată în figurile 2.2.4 şi 2.2.5 şi cu notaţiile din § 2.2.1, se propune următorul algoritm[1] :

1. Se stabileşte ţinta (vizuală) $\mathbf{t_v}$=`vtarg`, aflată în spaţiul de lucru al robotului. Pentru simulări, ţinta se poate alege aleator.
2. Se stabileşte poziţia iniţială **v**(0)=`v0` a braţului Se ia poziţia curentă $\mathbf{v}(k)|_{k=1}$=`v1`=`v0`. În spaţiul articulaţiilor aceasta corespunde vectorului $\theta(k)|_{k=1}$=`th1`.
3. Se iniţializează $\tilde{\mathbf{J}}\mathbf{c}$=`Jc`, conform algoritmului (4.3.2).
4. Se calculează ţinta temporară, aflată, în spaţiul 4D al camerelor CCD, în direcţia ţintei, la o distanţă de cel mult $\|\Delta\mathbf{v}_d\|_{max}$=`12dv` de poziţia curentă. Ea dă deplasarea dorită $\Delta\mathbf{v}_d$=`dv1`.
5. Se calculează comanda `dth1`=$\Delta\theta=\tilde{\mathbf{J}}\mathbf{c}\,\Delta\mathbf{v}_d$.
6. Braţul efectuează deplasarea cerută $\Delta\theta$, iar camerele observă deplasarea Δ**v**=`dv2`. Noua poziţie este **v**(k+1)=`v2`.
7. $\tilde{\mathbf{J}}\mathbf{c}$ este actualizată prin LMS, folosind $\Delta\theta$ şi $\Delta\mathbf{v}$.
8. Se testează dacă deplasarea $\Delta\mathbf{v}$ este corectă (criterii A, B). Dacă da, se sare la 10.
9. Dacă $\Delta\mathbf{v}$ nu este corectă, se revine în poziţia precedentă şi se continuă după reiniţializarea matricii $\tilde{\mathbf{J}}\mathbf{c}$: salt la 3.
10. Este eroarea rămasă $\|$`vtarg-v2`$\|_2$ suficient de mică ? Dacă nu, reluare de la 4.

(4.3.1)

Observaţii.

1. Pasul 7, actualizarea LMS, poate lipsi. Efectul ei asupra numărului necesar de iniţializări ale $\tilde{\mathbf{J}}\mathbf{c}$ va fi studiat prin simulări.
2. Criteriul ce va fi notat în continuare cu A este criteriul (4.2.5): condiţia ca $\angle(\Delta\mathbf{v}_d\ ,\ \Delta\mathbf{v})$, unghiul dintre direcţia dorită şi cea obţinută să nu fie superior unei valori alese, sau cosinusul acestui unghi să nu depăşească pragul `cos_min`. Criteriul notat cu B este criteriul introdus în § 2.2.1 şi aplicat în subcapitolul 2.4, de scădere a erorii faţă de ţinta temporară de cel puţin q_{max}=`qmax` ori la fiecare iteraţie (§ 2.5). Se observă că satisfacerea criteriului B implică satisfacerea criteriului A, cu $cos(\Delta\mathbf{v}_d\ ,\ \Delta\mathbf{v}) \le \sqrt{1-q^2}$. Reciproca nu este adevărată: eroarea poate scădea de q ori, dar unghiul dintre direcţia dorită şi cea obţinută poate fi mare, de exemplu 180°, dacă braţul se apropie de ţintă printr-o mişcare în zigzag. Prin aceasta criteriul B este mai restrictiv decât criteriul A.
3. Datorită transformării realizate de camerele CCD şi a erorii lor de discretizare, norma considerată la punctul 10 are efect asupra erorii de poziţionare în spaţiul geometric. Acest efect va face eventual obiectul altui studiu. Subiectul unui studiu ulterior ar putea fi şi efectul utilizării altor trăsături asupra formei traiectoriei obţinute.4. Prin natura algoritmului propus, este posibil ca matricea $\tilde{\mathbf{J}}\mathbf{c}$ să difere sensibil de valoarea sa reală. Aceasta poate duce la obţinerea unor comenzi $\Delta\theta$ mult prea mari. Pentru a se evita

[1] Simbolurile scrise cu caractere Courier corespund rutinelor de simulare în MATLAB, date în anexa II.

supracomanda brațului, mărimea vectorului $\Delta\theta$ se limitează la valoarea `12dth`$=\|\Delta\theta\|_{max}$, aleasă conform §§ 2.5.2-2.5.3, suficient de mică pentru a nu perturba funcționarea normală, cvasiliniară.

5. În forma prezentată, dacă ultimul pas este incorect, algoritmul (4.3.1) revine la poziția precedentă. Aceasta are ca avantaj garantarea satisfacerii pe întreaga traiectorie a criteriului ales, iar ca dezavantaj o mișcare cu reveniri. În practică poate fi însă mai utilă o variantă care, la efectuarea unui pas greșit, doar inițializează $\tilde{J}\mathbf{c}$, fără a reveni la poziția precedentă. Se pierde astfel garanția satisfacerii la fiecare pas a criteriului ales, dar se câștigă în netezimea deplasării. În cele ce urmează, aceasta va fi soluția adoptată.
6. Algoritmul (4.3.1) prezintă programul de simulare în principiu, în detalii acesta este ușor diferit. Conținutul său este listat în anexa II, fișierul aqrls.m.

Inițializarea $\tilde{J}\mathbf{c}$ se face conform algoritmului următor (funcția `Jcinit`, respectiv fișierul Jcinit.m din anexa II):

Funcția este apelată cu argumentele `th`-poziția curentă, și `thmem`, `tabth`, `tabJc`, descrise în continuare, și returnează `thmem`, `Jc`, `tabth`, `tabJc`.

1. Se examinează lista `tabth` ce conține valorile `thmem`, adică valorile 0 pentru care s-a memorat matricea jacobiană a comenzii. Dacă lista e goală, ea se inițializează cu poziția curentă `th1`, iar lista asociată `tabJc`, conținând matricile memorate `Jcmem`, se inițializează cu valoarea `Jc` obținută prin măsurarea a trei mici deplasări efectuate prin comanda succesivă a fiecărei articulații, și RLS (funcția `JcRLS`, respectiv fișierul JcRLS.m din anexa II). Subrutina este încheiată, returnându-se valoarea `Jc` măsurată.
2. Dacă lista `tabth` nu e goală, se găsește vectorul `tabth`, vecinul memorat cel mai apropiat de poziția curentă `th1`.
3. Se testează dacă vecinul cel mai apropiat este același cu `thmem` folosit.
4. Dacă da, întrucât $\tilde{J}\mathbf{c}$ deja utilizată nu este bună, se măsoară și se calculează `Jc`, ca la punctul 1. Valorile `th1` și `Jc` se adaugă în cele două tabele. Se revine din subrutină cu `Jc`.
5. Dacă nu, se citește din tabelul `tabJc` matricea `Jcmem`, cu care se revine din subrutină.

(4.3.2)

Observații.

1. Pentru claritate, algoritmul prezentat este ușor simplificat față de cel utilizat în simulări. De exemplu, acesta din urmă are în vedere și evitarea creșterii prea pronunțate a densității pozițiilor memorate `thmem`, prin impunerea unei distanțe minime între `thmem` și `th1`.
2. Sunt posibile scheme mai complicate, de mediere a celor mai apropiați k vecini, analog cu modul de clasificare al celor mai apropiați k vecini (*k-Nearest-Neighbor*) din recunoașterea formelor (Duda și Hart, 1973), în care clasificarea se face conform clasei indicate de cele mai multe dintre cele k prototipuri. Astfel de scheme sunt deja cunoscute (v. § 1.2.3; Littmann și Ritter, 1996).

3. Din punct de vedere practic, se poate considera că la parcurgerea unei regiuni noi, necunoscute, brațul se mișcă mai încet; din când în când se oprește, tatonează, și își continuă apoi mișcarea spre țintă. Încetinirea mișcării poate fi asociată cu necesitatea unei inițializări a jacobienei.

Algoritmul (4.3.2) corespunde unei structuri de rețea neuronală LLM (cu modelare local liniară) competitivă cu creștere. Astfel, fiecare poziție memorată corespunde unui "neuron" de tip competitiv (§ 1.2.1), în memoria căruia se găsește valoarea matricii jacobiene (§ 1.2.3). Apelul RN se face atunci când în algoritmul de comandă (4.3.1) matricea blocului de comandă nu este suficient de precisă, și corespunde necesității inițializării $\tilde{J}c$. Competiția între elementele rețelei constă în determinarea elementului al cărui vector thmem este cel mai apropiat de poziția curentă. În memoria acestui element se găsește matricea $\tilde{J}c$, care constituie răspunsul rețelei. Dacă cu această matrice se obține din nou o comandă insuficient de precisă, rețelei i se adaugă un nou element, ce învață aproximarea locală liniară din poziția curentă. Rețeaua învață aproximări LLM doar în zona în care robotul lucrează efectiv, în mod adaptiv, doar în măsura în care aproximarea deja disponibilă nu este suficient de bună. Această structură de rețea neuronală nu a fost întâlnită în literatură.

4.3.2. Simulări: deplasarea între două poziții fixe

Algoritmul (4.3.1) a fost verificat prin câteva simulări. Acestea au fost realizate în principal cu cele două criterii de la punctul 8 (observația 2). Dacă nu se specifică altfel, lungimea maximă a deplasării efectuate pentru o iterație corespunde unei distanțe vizuale de 12dv=10 pixeli. Simulările urmăresc să pună în evidență dependența numărului de puncte memorate cu neliniaritatea transformării cinematice inverse, cu criteriul de apreciere a corectitudinii pasului și cu adaptarea *on-line* LMS. Pentru aceasta, simularea lui ERICC se face fără limitarea domeniului de lucru; aplicarea limitărilor din (2.1.1) duce la obținerea, la marginea domeniului normal admisibil, a unor efecte neliniare ce maschează scopul urmărit. Astfel, întrucât deplasarea este comandată pornind de la vectori vizuali, este posibil ca, chiar pentru o comandă exactă, să se obțină o traiectorie în spațiul articulațiilor ce iese din domeniul normal de lucru. Evitarea acestei situații ține de modul de generare a traiectoriei, care depășește cadrul acestei lucrări.

Primul grup de simulări a urmărit punerea în evidență a învățării deplasării între două poziții fixe, date de θ_1=(0 0 0) și θ_2=(π/4 π/4 π/4). În fiecare simulare se efectuează șase deplasări, parcurgându-se de trei ori drumul între θ_1 și θ_2, dus-întors. Figurile prezintă traiectoriile pe ecranele CCD [3], traiectoriile în spațiul articulațiilor, reprezentat prin proiecțiile pe planurile θ_1-θ_2, θ_1-θ_3 și traiectoriile în coordonate carteziene, prin planurile x_1-x_2 și x_1-x_3. Traiectoriile sunt reprezentate prin linie continuă de nuanță deschisă, pozițiile atinse după fiecare iterație, prin puncte, pozițiile pentru care se face inițializarea lui $\tilde{J}c$, prin "o", pozițiile pentru care se memorează $\tilde{J}c$, prin "+", iar ținta finală, prin "x". Este de asemenea prezentată evoluția erorii de poziționare în coordonate liniare și logaritmice; erorile pentru cele șase

[3] Conform notațiilor din § 2.1.2, camera 1 este camera B, iar camera 2 este camera C.

traiectorii sunt concatenate, dând curbelor alura unor dinți de fierăstrău: valorile superioare reprezintă începutul unei noi poziționări, iar valorile inferioare, apropierea de țintă.

Un prim rezultat poate fi urmărit în figura 4.3.1 și în tabelul 4.3.1.

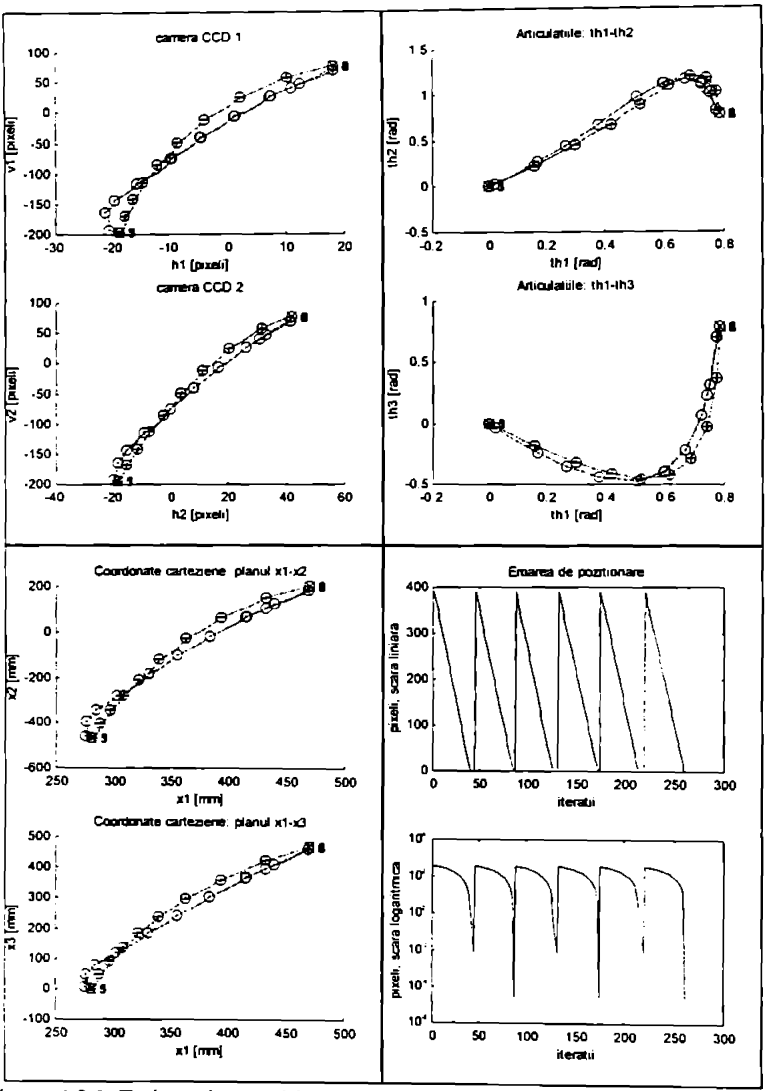


Figura 4.3.1. Traiectorie repetitivă între pozițiile date de unghiurile θ_1=(0 0 0) și θ_2=(π/4 π/4 π/4). Criteriul A, `cos_min`=0.99. Nu se face actualizare *on-line*.

Traiectoria nr.	1	2	3	4	5	6
Poziţii de iniţializare	9	11	9	10	9	10
Poziţii adăugate în memorie	9	1	0	0	0	0

Tabelul 4.3.1. Numărul poziţiilor iniţializate şi memorate pentru figura 4.3.1.

În figura 4.3.1 criteriul de evaluare a corectitudinii este cosinusul unghiului dintre direcţia dorită şi cea obţinută, care, în coordonate vizuale 4D, trebuie să fie superior lui cos_min=0,99. Nu se face actualizare LMS *on-line*. Conform figurii şi tabelului prezentat, la prima parcurgere, dus, a distanţei dintre θ_1 şi θ_2, au fost necesare 9 iniţializări şi tot atâtea memorări ale jacobienei. La întors, (traiectoria 2), valoarea $\tilde{J}c$ dată de cel mai apropiat vecin memorat a dat, pentru direcţia de întoarcere uşor diferită de cea se la dus, o comandă greşită (conform criteriului A), astfel încât a fost necesară o nouă iniţalizare cu măsurarea şi memorarea matricii de comandă. Pe parcursul întoarcerii a fost necesară de 11 ori iniţializarea, dar fără a se impune măsurarea (şi memorarea) matricii $\tilde{J}c$: valorile memorate la dus au fost suficient de bune. În continuare nu au mai fost necesare alte memorări. Traiectoria a 3-a (dus) a repetat-o întocmai pe prima. Cea de-a patra (întors) diferă uşor de a doua prin aceea că pasul greşit de la început este evitat, întrucât se dispune de un vecin mai apropiat, cu $\tilde{J}c$ suficient de bună. Mişcarea în continuare, pe traiectoriile 5 şi 6 a repetat traiectoriile 3 respectiv 4. Trebuie menţionat că programul de simulare prevede, la începutul fiecărei traiectorii, iniţializarea lui $\tilde{J}c$. În realitate, dacă braţul se mişcă pe traiectorii aflate una în continuarea celeilalte, această iniţializare nu mai este necesară. Astfel, faţă de valorile indicate în tabele, numărul real al iniţializărilor este mai mic cu 1 pentru toate traiectoriile, cu excepţia primeia.

În figura 4.3.2 sunt reproduse traiectoriile obţinute pe ecranele CCD şi în spaţiul articulaţiilor pentru aceeaşi simulare, dar cu actualizare *on-line*. Aşa cum se vede şi din tabelul 4.3.2, numărul de iniţializări şi memorări pentru $\tilde{J}c$ este sensibil mai mic (de cca 3 ori) decât

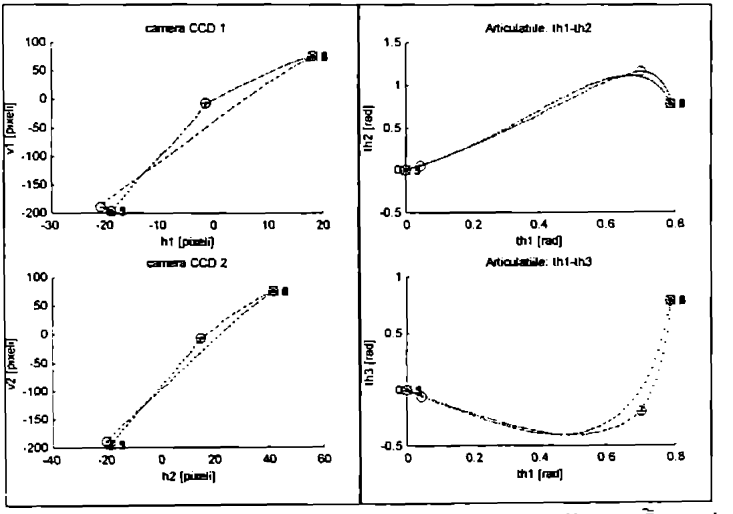


Figura 4.3.2. Aceeaşi simulare ca în figura 4.3.1, dar cu actualizarea $\tilde{J}c$ *on-line*.

Traiectoria nr.	1	2	3	4	5	6
Poziții de inițializare	3	2	3	2	3	2
Poziții adăugate în memorie	3	0	0	0	0	0

Tabelul 4.3.2. Numărul pozițiilor inițializate și memorate pentru figura 4.3.2.

în cazul precedent. De asemenea, traiectoriile sunt mai liniare pe porțiuni. După prima parcurgere dus-întors a distanței dintre pozițiile date de θ_1 și θ_2, nu mai este necesară învățarea jacobienei.

Alte două simulări au fost făcute relaxând parametrul `cos_min` la valoarea 0,8. Se permit astfel abateri mai mari ale direcției obținute față de direcția dorită a deplasării. Figura și tabelul 4.3.3 prezintă comportarea în absența actualizării *on-line* : față de cazul `cos_min`=0,99 , sunt necesare de 2,5 - 3 ori mai puține inițializări și memorări, dar traiectoriile sunt mai neliniare. Se remarcă totuși că traiectoriile CCD din figura 4.3.3 par mai neliniare decât sunt, datorită domeniilor diferite reprezentate pe orizontală (h) și verticală (v).

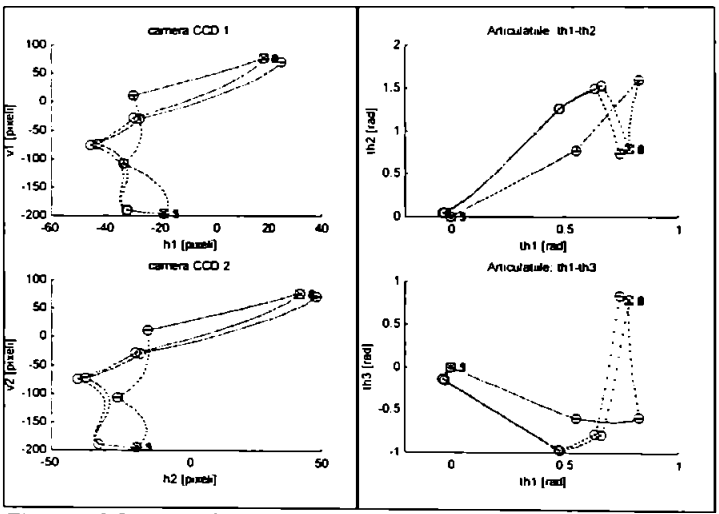


Figura 4.3.3. Aceeași simulare ca în figura 4.3.2, dar cu `cos_min`=0.8

Traiectoria nr.	1	2	3	4	5	6
Poziții de inițializare	3	5	3	4	3	4
Poziții adăugate în memorie	3	1	0	0	0	0

Tabelul 4.3.3. Numărul pozițiilor inițializate și memorate pentru figura 4.3.3.

Dacă se utilizează actualizarea LMS, figura şi tabelul 4.3.4 arată că numărul de iniţializări scade suplimentar de cca 2 ori, obţinându-se din nou liniarizarea pe porţiuni a traiectoriei. Comparaţia între figurile 4.3.4 şi 4.3.2 este ilustrativă pentru efectul parametrului `cos_min` asupra formei traiectoriei.

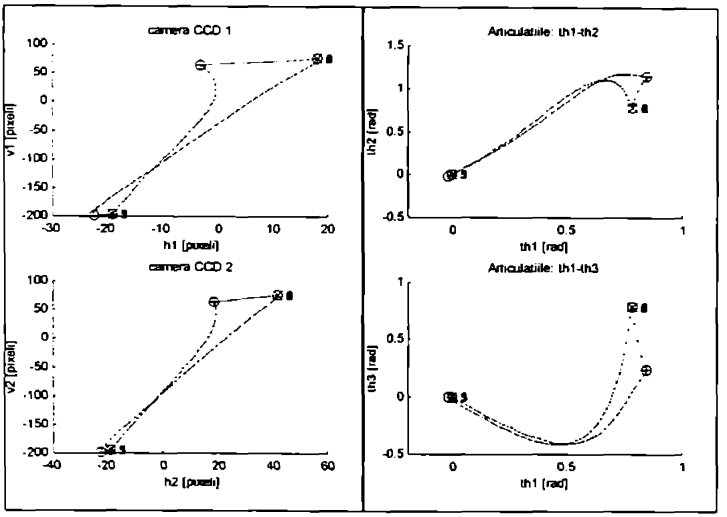


Figura 4.3.4. Aceeaşi simulare ca în figura 4.3.3, dar cu actualizarea *on-line* a $\tilde{J}c$.

Traiectoria nr.	1	2	3	4	5	6
Poziţii de iniţializare	2	2	2	2	2	2
Poziţii adăugate în memorie	2	0	0	0	0	0

Tabelul 4.3.4. Numărul poziţiilor iniţializate şi memorate pentru figura 4.3.4.

Aceleaşi simulări au fost făcute pentru criteriul B de apreciere a corectitudinii paşilor. Dacă se impune ca scăderea minimă a erorii de poziţionare faţă de ţinta temporară, qmax, să fie importantă, figurile şi tabelele 4.3.5 şi 4.3.6, obţinute pentru qmax=0.15 , arată că se obţin traiectorii aproape drepte, cu preţul mai multor iniţializări ale $\tilde{J}c$. Dacă qmax este mai mare, 0.8 , figurile 4.3.7 şi 4.3.8 arată că iniţializările se răresc, dar traiectoriile devin mai neliniare. După cum se poate constata comparând tabelele 4.3.5 cu 4.3.6 şi 4.3.7 cu 4.3.8, actualizarea *on-line* a jacobienei duce şi în acest caz la scăderea substanţială, de cca 4, respectiv 2 ori a numărului de iniţializări. Actualizarea *on-line* duce din nou la liniarizarea pe porţiuni a traiectoriilor. Se poate de asemenea remarca asemănarea dintre figurile 4.3.1 - 4.3.5, 4.3.2 - 4.3.6, 4.3.3 - 4.3.7 şi 4.3.4 - 4.3.8 : criteriile A şi B sunt practic echivalente, pentru deplasările cerute în acest caz.

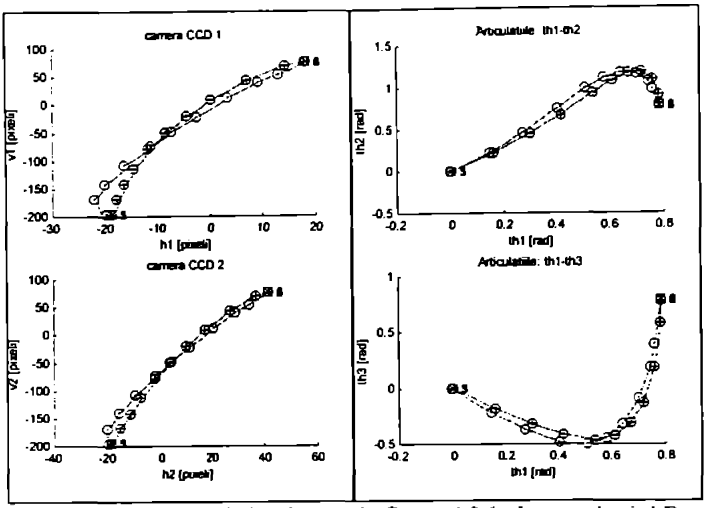


Figura 4.3.5. Aceeași simulare ca în figura 4.3.1, dar cu criteriul B, qmax=0,15 , fără actualizare *on-line*.

Traiectoria nr.	1	2	3	4	5	6
Poziții de inițializare	11	11	11	11	11	11
Poziții adăugate în memorie	11	0	0	0	0	0

Tabelul 4.3.5. Numărul pozițiilor inițializate și memorate pentru figura 4.3.5.

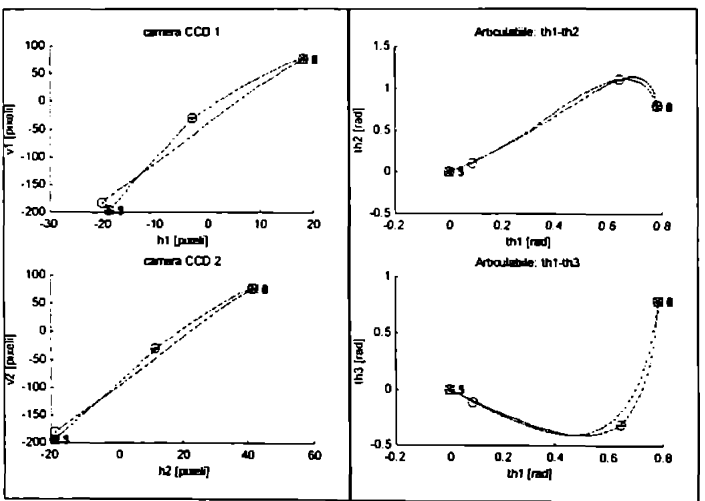


Figura 4.3.6. Aceeași simulare ca în figura 4.3.5, dar cu actualizare *on-line*.

Traiectoria nr.	1	2	3	4	5	6
Poziții de inițializare	3	2	3	2	3	2
Poziții adăugate în memorie	3	0	0	0	0	0

Tabelul 4.3.6. Numărul pozițiilor inițializate și memorate pentru figura 4.3.6.

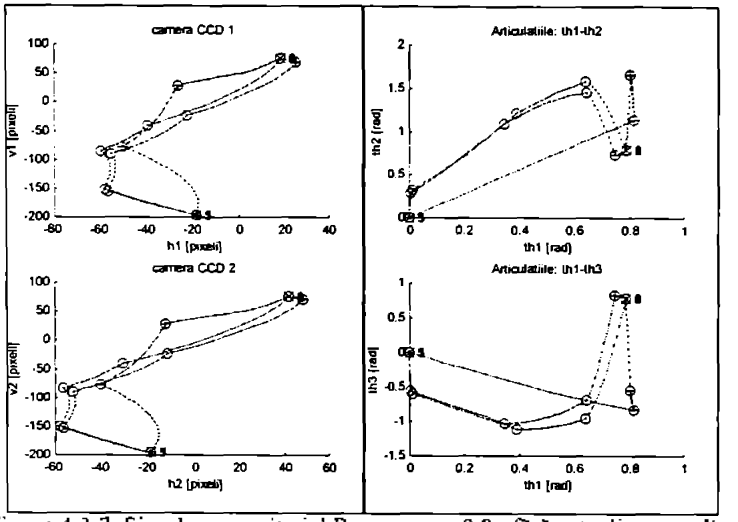


Figura 4.3.7. Simulare cu criteriul B, cu qmax=0,8 , fără actualizare *on-line*

Traiectoria nr.	1	2	3	4	5	6
Poziţii de iniţializare	4	5	4	4	4	4
Poziţii adăugate în memorie	3	1	0	0	0	0

Tabelul 4.3.7. Numărul poziţiilor iniţializate şi memorate pentru figura 4.3.7.

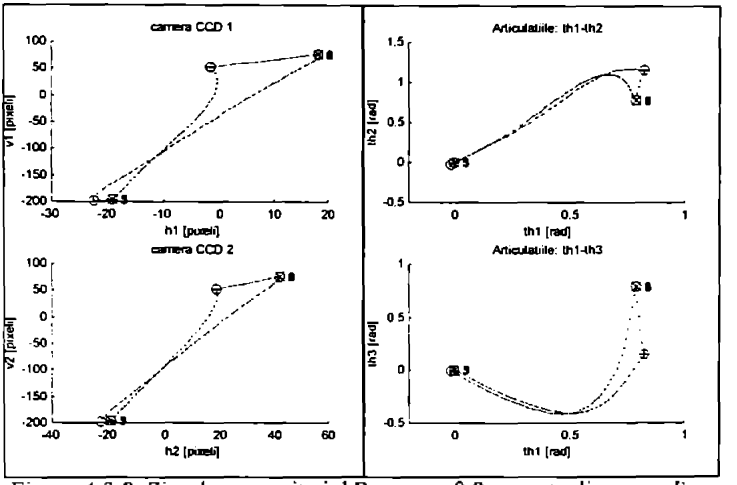


Figura 4.3.8. Simulare cu criteriul B, qmax=0,8, cu actualizare *on-line*.

Traiectoria nr.	1	2	3	4	5	6
Poziţii de iniţializare	2	2	2	2	2	2
Poziţii adăugate în memorie	2	0	0	0	0	0

Tabelul 4.3.8. Numărul poziţiilor iniţializate şi memorate pentru figura 4.3.8.

În fine, se poate menţiona că abaterile de la o dreaptă pentru traiectoriile vizuale prezentate în simulările precedente sunt datorate modului de aproximare a $\tilde{J}c$ şi principiului de poziţionare utilizat, cel al optimizării dinamice (§2.2, figura 2.2.5), şi nu aproximării local liniare utilizate: simulări suplimentare în condiţiile figurii 4.3.6 arată că creşterea sau scăderea de două ori a lungimii maxime a pasului faţă de 12dv=10 [pixeli] nu modifică sesizabil forma curbelor.

4.3.3. Simulări: deplasarea dintr-o poziţie iniţială aleatoare spre o ţintă fixă

Se doreşte a se compara numărul de iniţializări şi memorări pentru algoritmul propus, în cazul în care robotul se deplasează între mai multe puncte. Cazul cel mai general este acela în care atât poziţia iniţială cât şi ţinta, statică, au poziţii aleatoare. Simulările arată că braţul, pentru modelarea căruia nu s-au impus limitele din tabelul 2.1.1 şi ecuaţia (2.1.1), iese din domeniul de lucru. Cauza acestei probleme este impunerea traiectoriei drepte în spaţiul vizual. Aşa cum se vede în figura 4.1.1, aceasta duce la traiectorii curbe în spaţiul articulaţiilor, care pot ieşi din domeniul de lucru. Soluţia este generarea unei traiectorii vizuale mai complicate, sau simularea braţului cu limitări. Însă generarea traiectoriei este un capitol distinct al roboticii, care depăşeşte preocuparea lucrării de faţă. Pe de altă parte, impunerea limitărilor pentru braţ duce la efecte puternic neliniare la marginea domeniului, care maschează efectul neliniarităţii transformării cinematice inverse, care prezintă interes aici. De aceea a fost studiat doar cazul în care poziţionarea braţului se face spre o ţintă fixă, aflată relativ în centrul domeniului de lucru, dată de vectorul θ_f=(0 π/4 -π/6). Deplasările pornesc dintr-o poziţie aleatoare, dată de θ_0 aparţinând domeniului (-π/3 -π/4 -π/4), (π/3 π/2 π/2-π/6). Prin aceasta, marea majoritate a traiectoriilor sunt cuprinse în domeniul de lucru, şi studiul urmărit poate fi realizat cu uşurinţă.

O astfel de simulare se prezintă ca în figura şi tabelul 4.3.9. Examinându-le, se poate observa cum sunt utilizate valorile $\tilde{J}c$ memorate pe parcursul traiectoriilor precedente, la deplasarea din poziţii noi. Explicaţia figurii este aceeaşi ca pentru figura 4.3.1 şi următoarele. De această dată este vizibil şi numărul traiectoriei, notat în dreptul poziţiilor de început. Pentru prima traiectorie, datorită actualizării LMS utilizate, este suficientă o singură măsurare a jacobienei, în poziţia de start. Această valoare, actualizată pe parcurs, duce la obţinerea unei traiectorii drepte în spaţiul vizual, spre ţintă. La a doua traiectorie se face o primă iniţializare a $\tilde{J}c$ cu valoarea memorată la începutul primei traiectorii; această valoare nu satisface criteriul ales (B, qmax=0.8), şi se face măsurarea şi memorarea $\tilde{J}c$. Se mai măsoară $\tilde{J}c$ şi pe parcursul celei de-a treia traiectorii; pentru următoarele iniţializările se fac doar cu valori deja memorate. Aşa cum s-a văzut în paragraful precedent, actualizarea LMS duce la un număr mai mic de iniţializări şi la traiectorii vizuale mai liniare. Numărul de iteraţii în care ţinta este atinsă poate fi dedus din graficele erorii de poziţionare. El poate fi redus prin creşterea lungimii maxime a deplasării vizuale dorite, $\|\Delta v_d\|_{max}$=12dv=10 pixeli. Această valoare a fost aici aleasă anume relativ mică, pentru ca erorile datorate modelării local liniare să fie neglijabile faţă de erorile datorate abaterii aproximării $\tilde{J}c$ faţă de valoarea reală Jc.

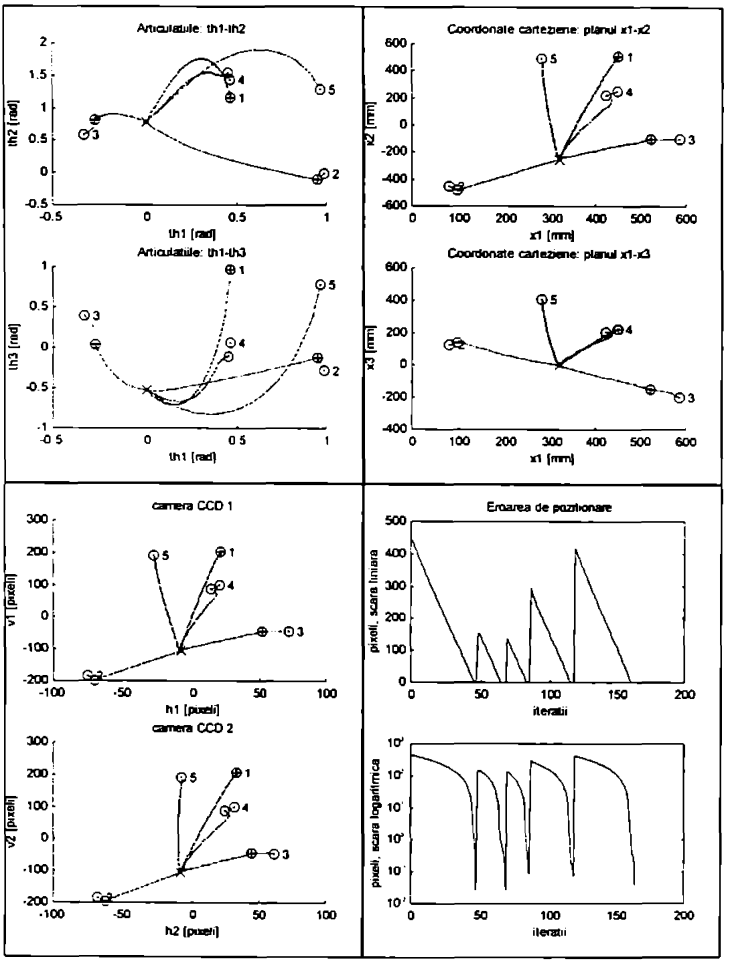


Figura 4.3.9. Cinci traiectorii pornind din poziţii aleatoare. spre ţinta dată de $\theta_f = (0\ \pi/4\ -\pi/6)$. Criteriul B, qmax=0.8 , cu actualizare LMS.

Traiectoria nr.	1	2	3	4	5
Poziţii de iniţializare	1	2	2	2	1
Poziţii adăugate în memorie	1	1	1	0	0
Poziţii iniţializate în total	1	3	5	7	8
Poziţii memorate în total	1	2	3	3	3

Tabelul 4.3.9. Numărul poziţiilor iniţializate şi memorate pentru figura 4.3.9.

Efectuarea unor simulări au arătat că criteriul A poate duce la oscilaţii pentru valori largi ale lui cos_min. precum 0,7 sau chiar 0.9. Cauza este transformarea datorată sistemului

vizual. Componenta unei deplasări în lungul axei de simetrie a celor două camere dă, mai ales pentru camere apropiate, o variație mică a vectorilor vizuali. Dacă nu se impune o valoare suficient de strânsă pentru `cos_min`, printr-o comandă insuficient de exactă poate apărea situația unei deplasări ce are o componentă mare în lungul axei comune a camerelor. Transformarea vizuală fiind relativ insensibilă la deplasări în această direcție, deplasarea se poate inversa la fiecare iterație, fără a duce la insatisfacerea criteriului de eroare și a corectării jacobienei. Acest comportament va fi eventual studiat suplimentar într-o lucrare ce va avea ca obiect erorile la modelarea local liniară a sistemului vizual. Oscilațiile pot fi evitate prin impunerea unei valori restrictive, precum `cos_min`=0,99.

O altă situație în care criteriul A poate duce la o mișcare oscilantă în jurul țintei este cea din figura 4.3.10. În acest caz, eroarea jacobienei duce la o comandă de direcție bună, dar prea mare. Direcția fiind bună, nu se cere măsurarea jacobienei, pentru a se obține o valoare care să dea o comandă fără erori, și mișcarea oscilantă persistă. Pentru evitarea acestui tip de oscilație, în apropierea țintei criteriul A, de corectitudine a direcției, poate fi înlocuit cu criteriul B, de descreștere a erorii la fiecare iterație. Nesatisfacerea acestui criteriu va impune obținerea unei comenzi corecte, ce va duce la evitarea oscilațiilor.

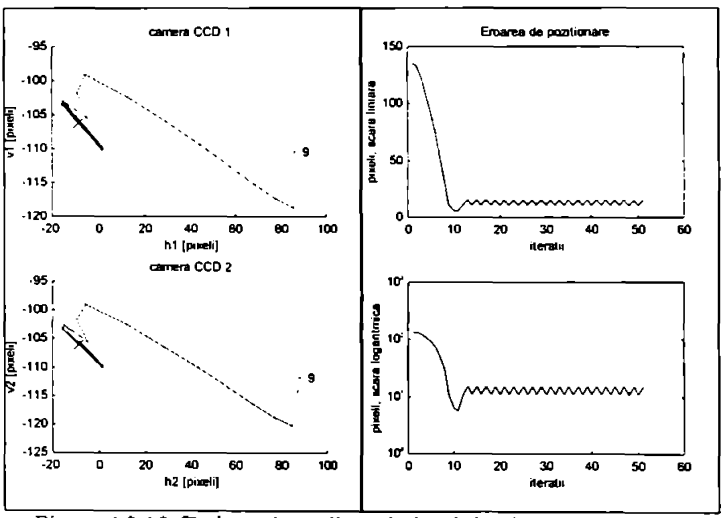


Figura 4.3.10. Traiectorie oscilantă în jurul țintei pentru criteriul A

Este interesantă compararea numărului de inițializări și memorări pentru cele două criterii, A și B, și pentru diverse valori ale parametrului de corectitudine, `cos_min`, respectiv qmax, la parcurgerea unui număr relativ mare de traiectorii pornind din poziții alese aleator. Simulările ce urmează permit această comparație. Pentru a se putea trage concluzii în condițiile alegerii aleatoare a punctelor de pornire, în fiecare simulare au fost efectuate câte 10 rulări, graficele prezentând valorile medii și deviațiile standard ale numărului total de inițializări și memorări pe parcursul a câte 400 de traiectorii pentru fiecare rulare. Graficele prezintă atât cazul în care actualizarea *on-line* este prezentă, cât și cazul în care actualizarea lipsește.

În figurile 4.3.11 -4.3.13 este utilizat criteriul B. În figura 4.3.11 este impusă valoarea maximă posibilă qmax=1. În figura 4.3.12 qmax=0.8, iar în figura 4.3.13, qmax=0.2, o valoare destul de mică. Figura 4.3.14 prezintă efectele criteriului A, cu cos_min=0.99.

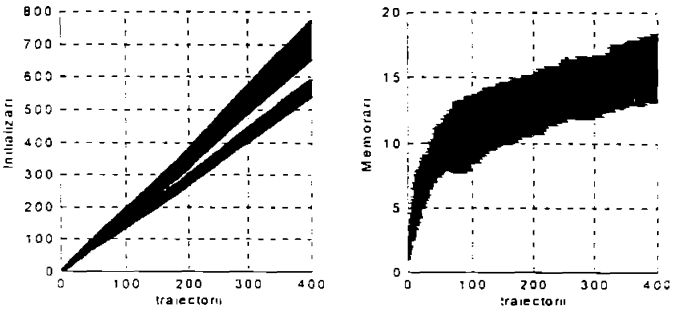


Figura 4.3.11 Numărul total de inițializări și memorări, criteriul B
qmax 1, fără (curbele superioare) și cu actualizare LMS (curbele inferioare)

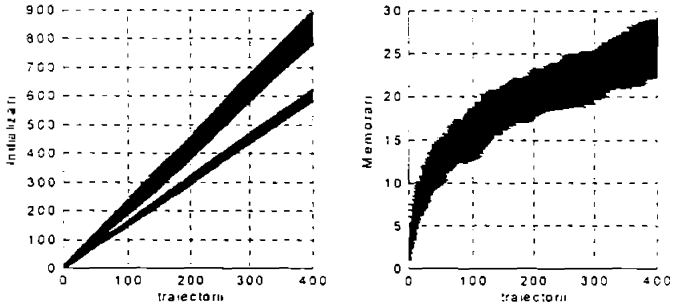


Figura 4.3.12. Numărul total de inițializări și memorări, criteriul B
qmax=0.8 , fără (curbele superioare) și cu actualizare LMS (curbele inferioare)

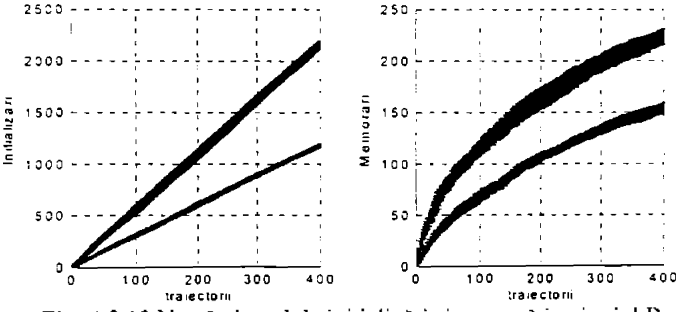


Fig. 4.3.13 Numărul total de inițializări și memorări, criteriul B.
qmax=0.2, fără (curbele superioare) și cu actualizare LMS (curbele inferioare)

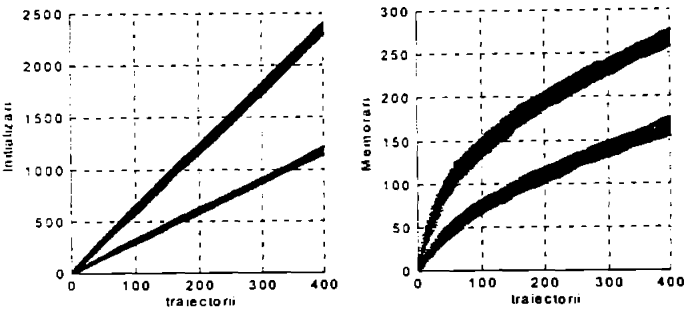


Fig. 4.3.14 Numărul total de inițializări și memorări, criteriul A, cos_min=0.99, fără (curbele superioare) și cu actualizare LMS (curbele inferioare)

Compararea acestor figuri arată că numărul inițializărilor și al memorărilor depinde mult de parametrul algoritmului sau, echivalent, de măsura în care se aproximează traiectoriile drepte. În medie, numărul inițializărilor crește liniar cu numărul traiectoriilor. Numărul memorărilor crește mai încet, având o tendință de saturare. Sunt necesare cam de 2-3 ori mai multe inițializări și 6-8 ori mai multe memorări dacă se doresc traiectorii relativ liniare (qmax=0.2 sau cos_min=0.99) față de cazul în care este suficientă poziționarea la țintă, indiferent de forma traiectoriilor (qmax=1). Actualizarea LMS duce la scăderea cu 30% - 50% a numărului de inițializări: Dacă qmax=0.8 , pentru 40 de traiectorii, sunt necesare 70 100 de inițializări, după cum actualizarea LMS este/nu este utilizată. Pentru 400 de traiectorii, circa 580/700 de inițializări sunt necesare, și circa 26 de memorări. Dacă qmax=0.2, pentru 40 de traiectorii sunt efectuate în medie 120/210 inițializări și 40 50 de memorări, iar pentru 400 de traiectorii, 1200/2100 de inițializări fac necesare 150/220 de elemente. În medie sunt necesare 3/5 inițializări și mai puțin de 0.5 memorări per traiectorie. Desigur, aceste valori depind de poziția țintei și de mărimea spațiului în care se poate afla poziția inițială.

Compararea figurilor 4.3.13–4.3.14 arată că similitudinea între criteriile A și B constatată în paragraful precedent se păstrează : pentru parametri aproximativ echivalenți ca formă a traiectoriilor (qmax=0.2, cos_min=0.99), numărul inițializărilor și memorărilor este apropiat.

Însă în raport cu concluzia ce părea că se degajă în § 4.3.2, că actualizarea LMS duce la scăderea și a numărului de memorări, figurile 4.3.11 și 4.3.12 dreapta aduc rezultate surprinzătoare. Ele arată că pentru valori mari ale parametrului qmax numărul memorărilor nu depinde de prezența sau absența actualizării *on-line*. Acest fapt poate fi explicat printr-un "volum" corespunzător fiecărei jacobiene memorate, dat de mulțimea pozițiilor pentru care $\tilde{J}c$ este corectă în limita parametrului ales, și un "volum" corespunzător actualizării LMS. Pentru valori mari ale parametrului qmax primul volum este mai mare decât al doilea, și jacobiana se măsoară în aproximativ aceleași puncte, fie că se face sau nu adaptarea LMS. Dacă qmax este mic, primul volum scade față de al doilea, și figura 4.3.13 arată că actualizarea LMS duce la scăderea numărului de memorări, cu până la 40% pentru qmax=0.2. Totuși, și pentru valori mari ale qmax, actualizarea LMS reduce numărul inițializărilor cu cca 25-50%, după cum arată figurile 4.3.11–4.3.12 stânga.

4.3.4. Concluzii, comparări cu alte metode

În § 4.3.1 s-a propus o nouă structură de RN, rețeaua neuronală LLM (cu modelare local liniară) competitivă cu creștere, și un algoritm de comandă adaptivă, care poate îmbunătăți aproximarea matricii de comandă prin utilizarea informației disponibile *on-line* pe parcursul deplasării.

Simulările efectuate cu acest algoritm, în §§ 4.3.2-4.3.3 permit desprinderea următoarelor concluzii :

- Învățarea mișcării între două puncte se poate face foarte rapid, la o singură parcurgere completă a traiectoriei.
- Cu toate că eroarea finală de poziționare poate fi arbitrar de mică, traiectoria obținută poate fi mai mult sau mai puțin apropiată de cea impusă, dreaptă. Abaterea traiectoriei de la cea impusă este funcție de parametrul de corectitudine al algoritmului. Traiectoriile mai precise necesită un număr mai mare de inițializări și memorări ale matricii jacobiene.
- Cele două criterii de evaluare a corectitudinii deplasării, propuse în (4.3.1 observația 2), duc la traiectorii similare. Criteriul A poate duce la oscilații. În varianta sa practică (4.3.1 observația 5), criteriul B funcționează fără probleme.
- Prin utilizarea informației disponibile pe parcursul deplasării actualizarea LMS *on-line* duce la scăderea numărului de inițializări și memorări, îndeosebi pentru erori mici impuse traiectoriilor.

Rețeaua neuronală LLM competitivă cu creștere este rezultatul unei abordări diferite față de abordarea clasică a aproximării prin modele local liniare (Ritter *et al.*, 1989, 1992; Walter și Schulten, 1993; Hesselroth *et al.*, 1994; Fritzke, 1995; Kuhn *et al.*, 1995; Gresser *et al.*, 1996). În toate aceste lucrări învățarea se face pe întreg spațiul, în puncte ce se dispun prin autoorganizare Kohonen (§ 1.2.1) sau adaptiv (§ 1.2.3). Pentru învățarea aproximărilor LLM brațul efectuează mișcări aleatoare, ample. Pe parcursul fiecărei traiectorii sunt găsiți neuronii ("vecinii") cei mai apropiați de poziția curentă, și valoarea corespunzătoare a matricii jacobiene ajustată prin algoritmul LMS, conform datelor Δv, $\Delta\theta$ pentru traiectoria și poziția respectivă. Se poate considera că aceasta este o învățare pasivă. Aici se propune o învățare activă : când este necesar, robotul își învață transformarea cinematică inversă, prin efectuarea unui număr minim de deplasări locale. De asemenea, învățarea se face doar în zona de lucru, eventual pentru câteva traiectorii de interes, și nu pentru întreg spațiul disponibil. Învățarea doar pentru traiectoriile de interes este proprie și rețelei CMAC, care însă învață aproximări constante pe portiuni ale funcției, și nu matrici jacobiene, printr-un algoritm total diferit (§ 1.2.3).

Față de RN prezente în literatură, rețeaua propusă se caracterizează printr-o creștere adaptivă, în funcție de precizia impusă, și doar în cazurile necesare. Prin ideea adăugării de elemente noi doar atunci când este necesar, rețeaua propusă seamănă cu rețelele ART (§ 1.2.3); aceasta este însă singura asemănare. Algoritmul lui Fritzke (1995) este și el adaptiv, bazat pe rețelele autoorganizante ce învață întreg domeniul de definiție al funcției de interes, inserând noi noduri acolo unde precizia este insuficientă. Dar prin aceasta el presupune învățarea prin prezentarea întregului set de antrenare, sau parcurgerea de mai multe ori a zonei

de lucru. Rețeaua propusă aici își alocă noi noduri pe chiar durata parcurgerii traiectoriilor, fiind mai eficace și utilizabil *on-line*.

Al doilea câștig important al rețelei propuse este adaptarea prin algoritmul RLS, cu o viteză mult mai mare de convergență față de algoritmul LMS. Cu excepția rețelei ART, care are o metodă de învățare total diferită, și a rețelei CMAC, care învață conform metodei iterative Gauss-Seidel de rezolvare a sistemelor liniare (Wong și Sideris, 1992; Golub și Van Loan, 1989), toate rețelele menționate învață prin algoritmul LMS[3]. Algoritmul LMS poate fi înlocuit cu algoritmul RLS în toate rețelele LLM, obținându-se o creștere însemnată a vitezei de învățare. Se menționează că este posibilă o economie suplimentară de mișcări de test pentru învățarea prin RLS: în locul a trei mișcări, se pot efectua doar două, cea de-a treia fiind deplasarea curentă spre țintă.

Pasul 7 al algoritmului (4.3.1) semnifică o adaptare *on-line*, pe parcursul deplasării, a ieșirii rețelei. În contextul rețelelor neuronale și în raport cu bibliografia consultată, ideea este nouă. Adaptarea nu necesită efectuarea unor mișcări suplimentare, informația folosită fiind disponibilă pe parcursul deplasării. Ea are ca efect liniarizarea pe porțiuni, uneori spectaculoasă, a traiectoriilor. Dacă parametrul criteriului de eroare este ales astfel încât să se obțină traiectorii vizuale drepte, această adaptare poate duce la diminuarea semnificativă a numărului de inițializări necesare.

Simulările efectuate cu modelul manipulatorului ERICC confirmă faptul că, față de metodele întâlnite în bibliografie, algoritmul (4.3.1)-(4.3.2) este mai economic și mai rapid.

Avantajele sunt evidente mai ales în cazul în care este necesară parcurgerea uneia sau a unui număr mic de traiectorii. Inițializarea și memorarea adaptivă, efectuată doar acolo unde este necesar, asigură învățarea unui număr minim de jacobiene. Actualizarea *on-line* poate face ca valoarea acestui minim să fie foarte redusă, de câteva elemente per traiectorie.

Avantajul învățării unui număr redus de jacobiene se păstrează și în cazul în care se dorește parcurgerea mai multor traiectorii. Metoda clasică, de plasare a punctelor după o hartă autoorganizantă Kohonen (§§ 1.2.1, 1.2.3) pretinde mai mulți neuroni. Ritter *et al.* (1992) utilizează 336 de neuroni, pentru un spațiu de lucru echivalent mai mic decât cel considerat aici. Pentru un domeniu de circa 5 ori mai restrâns (30°, 20°, -35°)-(100°, 90°, 35°), Kuhn *et al.* (1995) utilizează 700 de neuroni. Dacă este importantă doar poziționarea, rețeaua propusă poate avea doar 15 elemente pentru 400 de poziții inițiale diferite; dacă se dorește obținerea de traiectorii drepte, pentru 400 de poziții inițiale diferite sunt necesare 150 de elemente, iar pentru 40 de poziții inițiale diferite, circa 40 de elemente.

Înlocuirea algoritmului LMS cu algoritmul RLS la învățarea matricii jacobiene duce la creșterea semnificativă a vitezei de învățare. Învățarea rețelei lui Ritter *et al.*, incluzând autoorganizarea Kohonen, se face după circa 6000 de mișcări. Kuhn *et al.* are nevoie de 200 de mișcări. Aici, în fiecare punct unde este necesară învățarea jacobienei, sunt necesare (și suficiente) 3 deplasări, din care una poate fi deplasarea realizată în mișcarea dorită spre țintă.

[3] O excepție o face articolul (Schaal și Atkeson, 1995), întâlnit după redactarea acestei părți a tezei, unde este utilizat algoritmului RLS cu uitare ($\lambda<1$). Totuși, contextul lucrării, cel al unui amestec de aproximări liniare valabile pe întreg domeniul funcției și ponderate prin funcții gaussiene, prin care se obține caracterul local al aproximării, diferă de cel al lucrării de față. De asemenea, nu este scos în evidență câștigul în viteza de învățare față de algoritmul LMS.

Întrucât au moduri diferite de aproximare, rețeaua LLM nu poate fi comparată cu rețeaua CMAC. Totuși se poate menționa că rețeaua CMAC, considerată foarte performantă în privința vitezei, are teoretic o convergență exponențială (Wong și Sideris, 1992). Practic, pentru învățarea unei singure traiectorii, sunt necesare câteva zeci de parcurgeri ale acesteia (Miller, 1989; Wong și Sideris, 1992). Algoritmul (4.3.1) învață la o singură parcurgere a traiectoriei, dar efectuând, la nevoie, tatonări, pentru determinarea aproximărilor local liniare. În plus, rețeaua CMAC este o mare consumatoare de memorie, în timp ce memoria necesară pentru aproximarea local liniară adaptivă este foarte modestă.

5. Concluzii

Această lucrare a urmărit studiul şi îmbunătăţirea comenzii cinematice pentru poziţionarea unui braţ de robot cu reacţie vizuală prin reţele neuronale cu aproximare locală liniară pe porţiuni. În particular, a fost studiată în detaliu posibilitatea poziţionării exacte a braţului în condiţiile existenţei erorilor de comandă. S-a obţinut o structură originală de reţea neuronală ce permite acest lucru, într-o manieră deosebit de eficientă. Rezultatele obţinute în cazul simplu studiat pot fi generalizate pentru cazuri mai complicate.

5.1. Sinteză

Subiectul reţelelor neuronale este relativ nou. Pregătirea tezei, inclusiv prin referatele redactate şi prezentate, au permis familiarizarea celor interesaţi, cadre didactice şi studenţi, cu modul de gândire al acestui domeniu, al adaptabilităţii parametrilor unor structuri de aproximare, dar şi al construcţiei adaptive a acestor structuri. Studiul bibliografic destul de cuprinzător ce îl constituie primul referat pentru doctorat (Cimponeriu, 1994a) poate fi folosit ca material didactic. Materialul respectiv a fost reluat în mică măsură în lucrarea de faţă.

Cel de-al doilea referat pentru doctorat (Cimponeriu, 1994b) reflectă orientarea tezei spre aplicaţia practică a comenzii robotice. El este mai specializat, şi stă la baza prezentei teze.

Primul capitol al tezei face o scurtă trecere în revistă a reţelelor neuronale cu propagare înainte (*feedforward*), prezentându-le trăsăturile definitorii, şi punând accentul pe reţelele de aproximare locală. Sunt prezentate şi chestiuni ce nu sunt întâlnite sau sunt tratate sumar în literatura în limba română de reţele neuronale (Toderean, 1994; Dumitrescu şi Costin, 1996), precum proprietatea RN de a fi aproximatori universali, noţiunile legate de generalizare, reţelele de aproximare locală. Este făcut un studiu bibliografic întrucâtva mai aprofundat pe reţelele de aproximare locală. Sunt trecute în revistă realizările existente în domeniul îngust al aproximării local liniare. De asemenea, este prezentat pe scurt domeniul comenzii cinematice a roboţilor cu reacţie vizuală. Se pune accentul pe comanda în prezenţa erorilor.

În contextul primului capitol, autorul doreşte să menţioneze dificultatea considerabilă a realizării unui studiu bibliografic adus la zi, într-un domeniu aflat în plină dezvoltare, având o constantă de timp de ordinul lunilor. Se poate întâmpla, şi în teza de faţă a fost cazul în mai multe rânduri, să se constate că o anumită idee a fost deja dezvoltată şi publicată cu câteva luni sau chiar cu câţiva ani înainte. Apartenenţa la un colectiv specializat, a cărei existenţă este condiţionată de astfel de studii bibliografice, este o condiţie indispensabilă obţinerii de performanţe în cercetare.

Al doilea capitol studiază posibilitatea comenzii cinematice a unui braţ de robot cu reacţie vizuală în prezenţa erorilor inerente oricărei modelări, în particular a modelării adaptive prin reţele neuronale. Este prezentat braţul de robot şi sistemul vizual ce fac obiectul acestei lucrări. Sunt prezentate două modalităţi de comandă cinematică derivate din literatură. Este analizată partea comună iterativă a algoritmilor şi condiţia generală de convergenţă

asimptotică a modelării local liniare pentru întreaga buclă de reacție. Pentru obținerea unui suport matematic riguros în studiul buclei de reacție, este definită și utilizată transformata Z vectorială. În cadrul buclei cu reacție vizuală, analiza este apoi particularizată pentru brațul de robot. Pentru acesta, se calculează erorile maxime admisibile de aproximare a inversei matricii jacobiene, ca și domeniul de valabilitate al modelării local liniare, ce asigură convergența asimptotică a poziționării. Se obțin rezultate aplicabile în proiectarea și antrenarea de rețele neuronale pentru comanda brațului cu reacție vizuală. Rezultatele sunt validate prin simulări. Dintre concluzii, se menționează dificultatea de a se garanta convergența poziționării în orice poziție și pentru orice direcție de deplasare a brațului. Aceasta presupune oprirea antrenării prin testarea erorii pătratice maxime, și nu a mediei erorii pătratice pe setul de natrenare, care ia valori de câteva ori mai mici. Alte concluzii sunt concordanța excelentă cu rezultatele de simulare, și domeniul de valabilitate surprinzător de larg, pentru modelul local pătratic. Concluzia cea mai importantă este utilitatea unei învățări adaptive, în funcție de neliniaritatea locală a transformării cinematice inverse.

În capitolul al treilea se fac comparații între trei algoritmi de învățare a modelelor local liniare. Algoritmul LMS este simplu, aplicabil și în cazul sistemului studiat, în care vectorii de intrare pentru RN au 4 componente, dar sunt liniar dependenți, aparținând unui subspațiu liniar de dimensiune 3. Viteza de convergență a algoritmului LMS este scăzută; explicația, obținută printr-o analiză a algoritmului, constă în neortogonalitatea vectorilor de intrare. Ceilalți doi algoritmi prezentați ortogonalizează vectorii de intrare. Algoritmul Greville este aplicabil și în cazul vectorilor liniar dependenți, fiind mult mai rapid convergent. El însă presupune calcule mai complicate și mai ales memorarea tuturor vectorilor prezentați, ceea ce este nepractic. În fine, algoritmul RLS are o convergență rapidă și, deși mai complex decât algoritmul LMS, presupune un volum de calcule rezonabil și o memorie nu prea mare. Însă, în mod tradițional, el nu este aplicabil în cazul în care vectorii de intrare sunt liniar dependenți. Comparația analitică și prin simulări între algoritmul Greville și algoritmul RLS arată că, în condițiile inițializării uzuale a celui din urmă, cei doi algoritmi sunt echivalenți. Prin aceasta se demonstrează că algoritmul RLS poate fi utilizat în locul algoritmului LMS și în condițiile acestei lucrări. Se obțin astfel creșteri spectaculoase în viteza de convergență față de soluțiile prezentate în literatură, ce utilizează algoritmul LMS.

În cel de-al patrulea capitol se obține o structură de rețea neuronală pentru realizarea unei comenzi adaptive a brațului de robot cu reacție vizuală. La început este examinată învățarea prin algoritmul RLS a matricii de comandă în fiecare poziție a brațului. Este apoi studiată posibilitatea adaptării matricii de comandă pe baza doar a informației disponibile pe parcursul deplasării brațului spre țintă. Cu aceste rezultate preliminare, se formulează un algoritm de comandă adaptivă, bazat pe o rețea neuronală competitivă crescătoare, ce învață aproximările liniare doar pentru traiectoriile sau zonele în care se lucrează. Învățarea se face prin algoritmul RLS, fiind posibilă și actualizarea aproximării liniare prin utilizarea informației obținute pe parcursul deplasării. Învățarea și memorarea se face la prima parcurgere a traiectoriei, în pozițiile în care nu este satisfăcut un criteriu de eroare. Parcurgerile ulterioare ale aceleiași traiectorii sau a unor zone învecinate se face folosindu-se valorile memorate. Abaterea de liniaritate ale traiectoriilor este controlabilă prin parametrul criteriului de eroare aplicat, în funcție de care rezultă și dimensiunea rețelei neuronale obținute. Performanțele rețelei propuse sunt apreciate prin simulări. În final, se face o comparație cu alte metode prezente în bibliografie, rezultatele comparației fiind favorabile rețelei propuse.

5.2. Contribuţii originale

Contribuţia autorului la primul capitol, bibliografic, se manifestă în gruparea diverselor tipuri de reţele de aproximare locală, care de obicei apar în capitole diferite. Pe parcursul întregului capitol, autorul a căutat să realizeze o expunere cât mai sintetică, selectând şi prezentând doar trăsăturile care lui i s-au părut esenţiale. Sunt uneori prezente şi aprecieri calitative ale autorului. Note critice există îndeosebi în § 1.3.3, ce prefigurează problematica capitolului 2 al tezei.

Capitolul al doilea conţine mai multe contribuţii originale:

- Modelarea matricială a vederii stereoscopice din § 2.1.2 nu a fost întâlnită sub această formă. Relaţiile (2.1.8)-(2.1.11) sunt un exerciţiu personal, necesar realizării unei subrutine eficiente pentru simularea în MATLAB.
- Schemele de comandă propuse în § 2.2.1 nu au fost întâlnite în literatură sub această formă. Criteriul (2.2.1) de apreciere a corectitudinii deplasării, ce este utilizat în întreaga teză, este original. De menţionat că el este mai riguros decât criteriul (1.3.1) întâlnit în literatură, care nu ia în considerare posibilitatea obţinerii unei comenzi de mărime bună, dar de direcţie eronată.
- Modelul iterativ general din § 2.2.2, şi cel liniarizat din § 2.2.3, figura 2.2.7 şi ecuaţia (2.2.5), sunt obţinute de autor.
- Obţinerea comenzii ideale din § 2.2.4, cu separarea celor două pseudoinverse, a jacobienei sistemului vizual şi a jacobienei braţului, nu a fost întâlnită în bibliografie.
- În subcapitolul 2.3 este definită transformata Z vectorială, şi i se studiază convergenţa. Acestea permit obţinerea unor condiţii de stabilitate şi a erorii de regim staţionar pentru sisteme multidimensionale cu reacţie. Studiul general efectual în §§ 2.3.1-2.3.4 a fost publicat în articolul (Cimponeriu şi Policec, 1995). Rezultatele obţinute sunt aplicate în § 2.4.5 la cazul particular al buclei cu reacţie vizuală. Deşi rezultatul principal, (2.3.28), este dat şi de lucrarea (Goodwin şi Sin, 1984), obţinută după elaborarea acestui subcapitol, tratarea este originală. De asemenea, alte rezultate, importante pentru această teză, precum condiţia suficientă de stabilitate (2.3.29), sau demonstrarea anulării erorii de poziţionare chiar în prezenţa erorilor de comandă, (2.3.33), sunt originale. Relaţia (2.3.34) generalizează condiţia dată de (Goodwin şi Sin, 1984), ca o stare de echilibru să fie global asimptotic stabilă, şi pentru cazul în care intrarea este o constantă nenulă.
- Conţinutul subcapitolului 2.4 este în întregime original. Sunt cuprinse calculele din § 2.4.1, de determinare a erorii pătratice maxime a matricii pentru comanda braţului de robot, ca şi calculele din § 2.4.2, pentru determinarea domeniului de validitate a modelării local liniare. De asemenea, este originală conceperea şi realizarea simulărilor din § 2.4.4, prin care aceste calcule sunt validate. După cum rezultă din § 1.3.3, prin calculele din acest subcapitol se tratează analitic o direcţie puţin dezvoltată în bibliografie. Într-o formă preliminară, conţinutul acestui subcapitol a făcut obiectul comunicării (Cimponeriu şi Gresser, 1995). Într-o formă apropiată de cea prezentată în teză, acest subcapitol a fost propus spre publicare ca articolul (Cimponeriu şi Gresser, 1996), aflat, în momentul încheierii redactării tezei, în faza de corecţie a observaţiilor recenzenţilor.

Capitolul trei conține de asemenea contribuții originale:

- Analiza algoritmului LMS din § 3.1 nu este preluată din bibliografie.
- Algoritmul din § 3.2 : (3.2.20), (3.3.1), denumit algoritmul Greville, algoritm bazat pe calculul recursiv prin teorema lui Greville al soluției dată de pseudoinversa Moore-Penrose, este un exercițiu de obținere a unui algoritm iterativ, făcut de autor.
- Stabilirea echivalenței dintre algoritmul Greville și algoritmul RLS, subcapitolul 3.4, este originală. Prin această echivalență se extinde domeniul de valabilitate al algoritmului RLS și pentru cazul în care matricea de corelație a datelor de intrare este singulară. Acest rezultat a fost publicat în articolul (Cimponeriu, 1996). Stabilirea proprietăților esențiale ale soluției extensiei algoritmului RLS, prin analiza din § 3.2 și echivalența stabilită, este de asemenea originală.

Capitolul al patrulea valorifică rezultatele obținute în capitolele precedente, aducând noi contribuții:

- Metoda adaptării matricii de comandă pe baza informației disponibile pe parcursul deplasării brațului, studiată în § 4.2, este nouă.
- Algoritmul (4.3.1), prefigurat în § 2.2.1, nu a fost întâlnit sub această formă în literatură. El este foarte general, și reprezintă aplicarea pentru comanda robotului cu reacție vizuală, a principiului optimului dinamic, întâlnit în (Bassi și Bekey, 1989).
- Structura de rețea neuronală cu modelare local liniară, competitivă cu creștere, descrisă prin algoritmul (4.3.2), este originală. Ea îmbină trăsături ale mai multor tipuri de rețele neuronale prezentate în § 1.2.3, și face obiectul comunicărilor propuse (Cimponeriu și Gresser, 1997), (Cimponeriu și Kihl, 1997). Comparația cu metodele existente și avantajele față de acestea au fost prezentate în § 4.3.4. Caracteristicile acestei rețele și ale algoritmului de comandă adaptivă sunt:
  - învățare foarte rapidă, la o singură parcurgere a traiectoriei;
  - învățarea matricilor jacobiene doar în zona de lucru, care poate cuprinde una sau mai multe traiectorii. Nu este necesară parcurgerea întregului domeniu accesibil robotului. O traiectorie sau o zonă de lucru nouă este învățată utilizându-se datele deja disponibile, dacă acestea sunt corecte și pentru noile condiții. Se obține prin aceasta exploatabilitatea imediată a brațului de robot, și o comandă adaptivă eficientă;
  - efectuarea unui număr minim de tatonări pentru calculul matricii jacobiene, prin algoritmul RLS;
  - obținerea unei erori de poziționare nulă, în prezența erorilor de comandă, mulțumită reacției vizuale;
  - controlul abaterii traiectoriei vizuale de la forma impusă (aici, dreaptă), prin parametrul criteriului de corectitudine al algoritmului;
  - creșterea și dimensiunea finală a rețelei neuronale este minimă, în funcție de precizia impusă și de zona de lucru acoperită;
  - utilizarea informației disponibile pe parcursul deplasării și actualizarea LMS *on-line* a jacobienei, care duce la scăderea numărului de apelări și a dimensiunii rețelei obținute;
  - chiar față de soluțiile cele mai performante întâlnite în bibliografie, rețeaua obținută este deosebit de eficientă, fiind superioară prin viteza de învățare, prin adaptabilitate și prin cerințele reduse de memorie.

5.3. Generalizări şi direcţii de cercetare

În legătură cu problematica şi rezultatele acestei teze, pot fi schiţate următoarele generalizări şi direcţii de cercetare :

Un studiu întrucâtva similar modelării efectuate în subcapitolul 2.4, se poate face şi pentru sistemul vizual. Prin aceasta este de aşteptat să se poată determina eroarea maximă a pseudoinversei matricii jacobiene vizuale, ce ar asigura convergenţa asimptotică a poziţionării, în condiţiile cunoaşterii jacobienei braţului de robot. S-ar putea eventual obţine şi mărimea deplasărilor pentru care modelul local liniar este valabil. În plus faţă de aceste rezultate, similare celor obţinute în teză, s-ar putea eventual calcula modul în care se reflectă eroarea de cuantizare realizată de camerele CCD, în determinarea poziţiei geometrice. Prin aceasta se pot trage concluzii asupra dispunerii celor două camere. Tradiţional, acestea se amplasează fie în planuri perpendiculare, ceea ce intuitiv pare să aducă erori minime, fie în acelaşi plan, pentru a se obţine doar trei coordonate vizuale diferite (Hager *et al.*, 1994), fie sub un unghi mic, de 15°-30° (*e.g.* Hollinghurst şi Cipolla, 1993), eventual într-o montură precum cea din figura 2.1.3. Actualmente influenţa erorilor sistemului vizual şi amplasarea optimă a camerelor este fie calculată pentru cazuri foarte particulare (Hager *et al.*, 1994), fie determinată prin simulări (Tarabanis *et al.*, 1995), fie determinată experimental (*e.g.* Hollinghurst şi Cipolla, 1993).

Modelarea local liniară a sistemului vizual permite modelarea local liniară a întregii bucle de reacţie, pornind de la cele prezentate în subcapitolul 2.3.

Legat de asemenea de sistemul vizual, o direcţie importantă de cercetare este studiul efectului unor trăsături mai complexe decât simpla poziţie în spaţiul vizual (Feddema *et al.*, 1994; Bien *et al.*, 1994), asupra preciziei poziţionării în prezenţa erorii de cuantizare a camerelor şi asupra formei traiectoriei geometrice obţinute.

O altă direcţie de cercetare este extensia algoritmul RLS rezultată din subcapitolul 3.4. În teză s-a demonstrat că, în plus faţă de ceea ce se ştia, acesta este valabil şi în cazul în care matricea de corelaţie a intrărilor este singulară. Aceeaşi proprietate o are şi algoritmul QR (Haykin, 1991). O comparaţie între eficienţele de calcul pentru cei doi algoritmi, luând în considerare şi varianta rapidă FTF (*Fast Transversal Filter*) a algoritmului RLS (Cioffi şi Kailath, 1984; Michaut, 1992), poate aduce rezultate interesante.

De asemenea, algoritmul RLS este aplicabil pentru calculul jacobienei imună la singularităţi (*singularity-robust Jacobian*, Nakamura, 1991), care în lucrarea menţionată este calculată prin descompunerea după valori singulare, sau chiar prin teorema lui Greville. Alte metode pentru calculul sau actualizarea acesteia sunt date în (Mayorga *et al.*, 1993; Telfer şi Casasent, 1994; Kreutz-Delgado şi Agahi, 1995). Faţă de acestea, algoritmul RLS pare să necesite calcule mai puţine.

În lucrarea de faţă, pentru aplicarea algoritmului RLS este esenţială existenţa buclei de reacţie, prin care se obţin variaţiile ieşirilor sistemului pentru variaţii date ale intrărilor. Utilizarea algoritmului RLS poate fi extinsă şi pentru reţele cu modelare local liniară fără reacţie, în care învăţarea se face pe baza unui set de antrenare. Astfel, pentru un vector de intrare ales x, se pot detecta cei mai apropiaţi vecini din setul de antrenare, calculate distanţele

Δx față de aceștia, ca și diferențele Δy dintre ieșirile dorite corespunzătoare. Cu acești vectori, poate fi calculată jacobiana în $\mathbf{x}$, prin RLS.

Prin numărul redus de parametri și prin învățarea locală, este de așteptat ca acest tip de rețea să aibă performanțe foarte bune de generalizare; acest fapt este dealtfel semnalat în literatură, pentru alte structuri de rețele LLM (Fritzke, 1995; Littmann și Ritter, 1996). O direcție de cercetare, în cazul confirmării acestor performanțe pe probleme test consacrate în literatură, precum predicția serici de timp Mackey-Glass (Lapedes și Farber, 1987, citat de Littmann și Ritter, 1996), este studiul teoretic al acestor performanțe, urmând abordarea lui Vapnik (1995).

Aproximarea prin modele local liniare nu este netedă. Ponderând local toate funcțiile (global) liniare, se obțin structuri echivalente cu rețelele RBF generalizate (Poggio și Girosi, 1990b, citat de Littmann și Ritter, 1996; Schaal și Atkeson, 1995), care realizează funcții netede. Aproximanți netezi se obțin și prin interpolarea valorilor funcției prin funcții *spline* cubice (Heiss, 1994). O direcție de cercetare ar putea fi căutarea unor funcții ce ar permite ponderarea doar a câtorva din modelele liniare, având ca inspirație "amestecul de experți" utilizat pentru clasificare (Jordan și Jacobs, 1994; Alpaydin și Jordan, 1996; Ripley, 1996), și prin care să se obțină funcții netede. O altă astfel de posibilitate ar putea fi interpolarea prin funcții *spline* nu a valorilor funcției, ci a valorilor jacobienei (prima derivată, în spațiul multidimensional).

Condiționat de împrejurările desfășurării acestui doctorat, obiectul tezei este cazul cel mai simplu al comenzii robotice, și anume comanda cinematică pentru un braț neredundant. O continuare necesară va fi aplicarea modelării local liniare pentru comanda a mai mult de 3 grade de libertate și pentru modelarea dinamică a brațului manipulator. În primul caz, se pot obține brațe redundante sau cu singularități, pentru care soluția utilizată este pseudoinversa Moore-Penrose, studiată aici, respectiv jacobiana imună la singularități, menționată mai sus. Cel de-al doilea caz, al modelării dinamice, este prezent în majoritatea referințelor date în subcapitolul 1.3.

În algoritmul central acestei teze traiectoria este generată conform principiul optimului dinamic, dorindu-se obținerea unei traiectorii în spațiul vizual drepte spre țintă. În § 2.2 s-a menționat că, dacă se dorește urmărirea unei traiectorii de altă formă, se pot utiliza drept ținte temporare puncte intermediare de pe traiectoria dorită (Jägersand și Nelson, 1994). Generarea traiectoriei, problemă ce apare natural, este de o cu totul altă natură, ea făcându-se pe baza definirii (și recunoașterii) obstacolelor, ce introduc restricții de mișcare, sau a definirii unui câmp potențial, a cărui valoare crește rapid în apropierea obstacolelor, și scade treptat spre țintă (*e.g.* Drăgulescu *et al.*, 1994). Această problematică este complementară tematicii acestei teze, dar se pretează și ea unei abordări prin rețele neuronale (*e.g.* Bekey și Goldberg, 1993).

Preocupările lucrării de față pot fi considerate ca situându-se în domeniul eșantionării adaptive a funcțiilor de mai multe variabile sau a cuantizării vectoriale adaptive. Așadar, este de așteptat ca ele să poată fi aplicate, într-o formă adaptată, și în prelucrarea semnalelor.

Anexa I. Noțiuni de algebră liniară : pseudoinversa Moore-Penrose, descompunerea după valori singulare a matricilor, norme matriciale [1]

(1.1) Transformările liniare dintr-un spațiu euclinian finit-dimensional $\mathcal{E}_n$ în altul $\mathcal{E}_m$ pot fi reprezentate prin matrici $\mathbf{A}^{m\times n}$.

(1.2) Produsul scalar al vectorilor reali $\mathbf{x} = (x_1,...,x_i,...x_n)^T$ și $\mathbf{y} = (y_1,...,y_i,...y_n)^T$ este $\langle \mathbf{x},\mathbf{y}\rangle = \mathbf{x}^T\cdot\mathbf{y} = \sum_{i=1}^{n} x_i y_i$. Norma lui x este $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x},\mathbf{x}\rangle}$.

(1.3) Un subspațiu liniar $\mathcal{L}$ este o submulțime nevidă a unui spațiu liniar, închisă față de adunare și înmulțirea cu scalari.

(2.1) Doi vectori sunt ortogonali, $\mathbf{x}\perp\mathbf{y}$, dacă $\langle \mathbf{x},\mathbf{y}\rangle = 0$. Un vector $\mathbf{x}$ este ortogonal pe subspațiul liniar $\mathcal{L}$, $\mathbf{x}\perp\mathcal{L}$, dacă el este ortogonal pe orice vector din $\mathcal{L}$. Mulțimea tuturor $\mathbf{x}\perp\mathcal{L}$ este subspațiu liniar, complementul ortogonal al lui $\mathcal{L}$, $\mathcal{L}^{\perp}$.

(2.2) Spațiul liniar considerat este suma directă dintre $\mathcal{L}$ și $\mathcal{L}^{\perp}$. Adică, orice vector $\mathbf{x}$ se descompune unic : $\mathbf{x} = \hat{\mathbf{x}} + \tilde{\mathbf{x}}$, cu $\hat{\mathbf{x}}\in\mathcal{L}$ și $\tilde{\mathbf{x}}\in\mathcal{L}^{\perp}$. Are loc $\|\mathbf{x}\|^2 = \|\hat{\mathbf{x}}\|^2 + \|\tilde{\mathbf{x}}\|^2$.

(2.3) Pentru orice $\mathbf{y}\in\mathcal{L}$, $\|\mathbf{x}-\mathbf{y}\|^2 \geq \|\mathbf{x}-\hat{\mathbf{x}}\|^2$, cu egalitate doar pentru $\mathbf{y}=\hat{\mathbf{x}}$. $\hat{\mathbf{x}}$ este aproximarea din $\mathcal{L}$ de eroare de normă minimă a lui x:

$$\|\mathbf{x}-\mathbf{y}\|^2 = \|(\hat{\mathbf{x}}-\mathbf{y})+\tilde{\mathbf{x}}\|^2 = \|\hat{\mathbf{x}}-\mathbf{y}\|^2 + \|\tilde{\mathbf{x}}\|^2 \geq \|\tilde{\mathbf{x}}\|^2 = \|\mathbf{x}-\hat{\mathbf{x}}\|^2.$$

(3.1) Subspațiul generat (sau acoperit) de vectorii $\mathbf{a}_i$ i=1...n. $\mathcal{L}(\mathbf{a}_1,...,\mathbf{a}_i,...,\mathbf{a}_n)$ este mulțimea tuturor combinațiilor liniare de $\mathbf{a}_i$. Unei matrici $\mathbf{A}^{m\times n}$ i se pot asocia două subspații liniare: unul generat de vectorii săi coloană $\mathbf{a}_i^{m\times 1}$, i=1...n, $\mathcal{R}(\mathbf{A})$ (*range*), și celălalt generat de vectorii săi linie $\mathbf{b}_j^{T\ n\times 1}$ j=1...m, sau vectorii coloană ai transpusei, $\mathcal{R}(\mathbf{A}^T)$. Dimensiunea subspațiilor generate este egală cu rangul matricii A, sau numărul vectorilor coloană (sau linie) liniar independenți.

(3.2) Produsul dintre vectorul $\mathbf{x}^{n\times 1}$ și matricea $\mathbf{A}^{m\times n}$, $\hat{\mathbf{z}} = \mathbf{A}\mathbf{x}$, poate fi scris ca:

$$\hat{\mathbf{z}} = \sum_{i=1}^{n} x_i\mathbf{a}_i \in \mathcal{R}(\mathbf{A}), \text{ sau}$$

$$\hat{\mathbf{z}} = \begin{pmatrix} y_1 & y_j & y_m \end{pmatrix}^T, \quad y_j = \mathbf{b}_j\cdot\mathbf{x}.$$

(3.3) Subspațiul nul al matricii A, $\mathcal{N}(\mathbf{A})$, este mulțimea vectorilor x având imaginea nulă:

[1] Dacă nu se specifică altfel, expunerea și notațiile sunt cele din (Albert, 1972).

$\mathcal{N}(\mathbf{A}) = \{\mathbf{x}: \mathbf{Ax} = \mathbf{0}\}$. Are loc relația: $\mathcal{N}(\mathbf{A}) = (\mathcal{R}(\mathbf{A}^T))^{\perp}$. (Cu notațiile (3.2), $\hat{\mathbf{z}} = \mathbf{0} \Rightarrow \hat{z}_j = 0 \Rightarrow \mathbf{x} \perp \mathbf{b}_j$, $j = 1...n \Rightarrow \mathbf{x} \quad (\mathcal{R}(\mathbf{A}^T))^{\perp}$.)

(4.1) O matrice (pătrată) este simetrică dacă este egală cu transpusa sa. Produsele de forma $\mathbf{H}^T\mathbf{H}$ și $\mathbf{H}\mathbf{H}^T$ sunt simetrice.
(4.2) Matricile $\mathbf{H}$ și $\mathbf{H}\mathbf{H}^T$ generează același subspațiu, $\mathcal{R}(\mathbf{H}) = \mathcal{R}(\mathbf{H}\mathbf{H}^T)$, sau $\mathcal{N}(\mathbf{H}^T) = \mathcal{N}(\mathbf{H}\mathbf{H}^T)$. Similar, $\mathcal{R}(\mathbf{H}^T) = \mathcal{R}(\mathbf{H}^T\mathbf{H})$, $\mathcal{N}(\mathbf{H}) = \mathcal{N}(\mathbf{H}^T\mathbf{H})$.
($\mathcal{N}(\mathbf{H}^T) = \mathcal{N}(\mathbf{H}\mathbf{H}^T)$: $\mathbf{H}^T\mathbf{x} = \mathbf{0}$, $(\forall)\,\mathbf{x} \Rightarrow \mathbf{H}\mathbf{H}^T\mathbf{x} = \mathbf{0}$, $(\forall)\,\mathbf{x}$, $\mathcal{N}(\mathbf{H}^T) \subset \mathcal{N}(\mathbf{H}\mathbf{H}^T)$; $\mathbf{H}\mathbf{H}^T\mathbf{x} = \mathbf{0}\ (\forall)\,\mathbf{x} \Rightarrow \mathbf{x}^T\mathbf{H}\mathbf{H}^T\mathbf{x} = 0$, $\|\mathbf{H}^T\mathbf{x}\|_2^2 = 0\ (\forall)\mathbf{x} \Rightarrow \mathcal{N}(\mathbf{H}\mathbf{H}^T) \subset \mathcal{N}(\mathbf{H}^T)$;
$\mathcal{N}(\mathbf{H}^T) = (\mathcal{R}(\mathbf{H}))^{\perp}$, $\mathcal{N}(\mathbf{H}\mathbf{H}^T) = (\mathcal{R}(\mathbf{H}\mathbf{H}))^{\perp} \Rightarrow \mathcal{R}(\mathbf{H}) = \mathcal{R}(\mathbf{H}\mathbf{H}^T)$.)

(5.1) O matrice pătrată este nesingulară dacă subspațiul său nul constă din doar vectorul nul. O matrice care nu e nesingulară e singulară. Dacă vectorii linie ai lui $\mathbf{H}$ sunt liniar independenți. atunci $\mathbf{H}$ e nesingulară.
(5.2) Dacă $\mathbf{H}$ e o matrice oarecare și $\delta \neq 0$. atunci $\mathbf{H}^T\mathbf{H} + \delta^2\mathbf{I}$ este nesingulară.
$\left(\left(\mathbf{H}^T\mathbf{H} + \delta^2\mathbf{I}\right)\mathbf{x} = \mathbf{0} \Rightarrow \mathbf{x}^T\left(\mathbf{H}^T\mathbf{H} + \delta^2\mathbf{I}\right)\mathbf{x} = \|\mathbf{H}\mathbf{x}\|_2^2 + \delta^2\|\mathbf{x}\|_2^2 = 0 \text{ d.d. } \mathbf{x} = \mathbf{0} \right)$

(6.1) Dacă $\mathbf{A}$ este reală și simetrică. valorile sale proprii sunt reale.
$$\left(\left(\mathbf{A} - \lambda\mathbf{I}\right)\mathbf{x} = \mathbf{0},\ \mathbf{A} \text{ reala } \Rightarrow \left(\mathbf{A} - \lambda^*\mathbf{I}\right)\mathbf{x}^* = \mathbf{0} \right.$$
$$\left. \lambda\,\mathbf{x}^{*2} = \lambda\mathbf{x}^T\mathbf{x}^* = \left(\mathbf{A}\mathbf{x}^T\right)\mathbf{x}^* = \mathbf{x}^T\mathbf{A}^T\mathbf{x}^* = \mathbf{x}^T\mathbf{A}\mathbf{x}^* = \lambda^*\mathbf{x}^T\mathbf{x}^* = \lambda^*\|\mathbf{x}\|_2^2,\ \lambda = \lambda^* \in \mathbf{R} \right)$$

(7.1) O matrice pătrată nesingulară $\mathbf{T}$ este ortogonală dacă $\mathbf{T}^T = \mathbf{T}^{-1}$. Dacă $\mathbf{T}$ este ortogonală, atunci vectorii săi coloană sunt ortonormați. La fel și vectorii săi linie.
(7.2) Dacă $\mathbf{A}$ este simetrică cu valorile proprii λ_i, i=1...n. atunci există o matrice ortogonală $\mathbf{T}$. anume cea având coloanele vectorii proprii ai lui $\mathbf{A}$. astfel încât $\mathbf{T}^T\mathbf{A}\mathbf{T} = \text{diag}(\lambda_1,...\lambda_i,...\lambda_n)$ adică matricea având primele n elemente de pe prima diagonală $\lambda_1,... \lambda_n$.

(8.1) O matrice (pătrată) $\mathbf{A}$ este idempotentă dacă $\mathbf{A}^2 = \mathbf{A}\mathbf{A} = \mathbf{A}$, aplicarea de două ori a transformării $\mathbf{A}$ nu mai modifică rezultatul obținut prin aplicarea ei o dată.
(8.2) Imaginea vectorului $\mathbf{x}$ prin transformarea liniară $\mathbf{A}$. denumită și proiecția lui $\mathbf{x}$ prin $\mathbf{A}$ este vectorul $\hat{\mathbf{x}} = \mathbf{A}\mathbf{x}$.
(8.3) O matrice (pătrată) $\mathbf{P}$ este o matrice proiecție ortogonală dacă ea este idempotentă și simetrică: $\mathbf{P}^2 = \mathbf{P}$, $\mathbf{P}^T = \mathbf{P}$.

(proiecția unui vector proiecție este el însuși:
$\hat{\mathbf{x}} = \mathbf{P}\mathbf{x}$ $\mathbf{P}\hat{\mathbf{x}} = \mathbf{P}^2\mathbf{x} = \hat{\mathbf{x}} = \mathbf{P}\mathbf{x}$, $(\forall)\,\mathbf{x} \Rightarrow \mathbf{P}^2 = \mathbf{P}$;
rezultatele descompunerii prin proiecție sunt ortogonale:
$\mathbf{P}\mathbf{x} = \hat{\mathbf{x}}$, $\mathbf{x} = \hat{\mathbf{x}} + \tilde{\mathbf{x}}$, $\tilde{\mathbf{x}} = (\mathbf{I} - \mathbf{P})\mathbf{x}$, $\hat{\mathbf{x}}^T\tilde{\mathbf{x}} = (\mathbf{P}\mathbf{x})^T(\mathbf{I} - \mathbf{P})\mathbf{x} = \mathbf{x}^T\left(\mathbf{P}^T - \mathbf{P}^T\mathbf{P}\right)\mathbf{x} = 0$, $(\forall)\mathbf{x}$
$\Rightarrow \mathbf{P}^T\mathbf{P} = \mathbf{P}^T$, $\left(\mathbf{P}^T\mathbf{P}\right)^T = \mathbf{P} = \mathbf{P}^T\mathbf{P} = \mathbf{P}^T$, $\mathbf{P} = \mathbf{P}^T$.)

(9) Fie vectorii $\mathbf{z}^{mx1}$, $\mathbf{x}^{nx1}$ și matricea $\mathbf{H}^{mxn}$. Se pune problema găsirii soluției $\mathbf{x}$ de normă minimă ce minimizează $\|\mathbf{z}-\mathbf{Hx}\|^2$.

(9.1.1) Există soluții ce minimizează $\|\mathbf{z}-\mathbf{Hx}\|^2$, anume soluții ale ecuației $\mathbf{Hx}=\hat{\mathbf{z}}$, unde $\hat{\mathbf{z}}\in\mathcal{R}(\mathbf{H})$ este proiecția lui $\mathbf{z}$ pe $\mathcal{R}(\mathbf{H})$.

(9.1.2) Între acestea, $\hat{\mathbf{x}}\in\mathcal{R}(\mathbf{H}^T)$ este soluția de normă minimă, unică; ea este de forma $\hat{\mathbf{x}}=\mathbf{H}^T\mathbf{y}$.

(9.2) Întrucât $\hat{\mathbf{z}}$ este proiecția lui $\mathbf{z}$ pe $\mathcal{R}(\mathbf{H})$, este necesar și suficient ca soluțiile $\mathbf{x}$ să satisfacă $\mathbf{H}^T\mathbf{Hx}=\mathbf{H}^T\hat{\mathbf{z}}=\mathbf{H}^T\mathbf{z}$.

Observații: 1. (9.2) se poate obține și din condiția $\dfrac{\partial\|\mathbf{z}-\mathbf{Hx}\|^2}{\partial\mathbf{x}}=\mathbf{0}$.

2. (9.2) ar putea fi rezolvată ca $\mathbf{x}=\left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{z}$ dacă $\mathbf{H}^T\mathbf{H}$ ar fi inversabilă.

(9.3) Matricea $\mathbf{H}^{+nxm}=\lim_{\delta\to0}\left(\mathbf{H}^T\mathbf{H}+\delta^2\mathbf{I}\right)^{-1}\mathbf{H}^T=\lim_{\delta\to0}\mathbf{H}^T\left(\mathbf{HH}^T+\delta^2\mathbf{I}\right)^{-1}$ există pentru orice $\mathbf{H}^{mxn}$, și pentru orice $\mathbf{z}^{mx1}$, $\hat{\mathbf{x}}=\mathbf{H}^+\mathbf{z}$ este vectorul de dimensiune nx1 de normă minimă ce minimizează $\|\mathbf{z}-\mathbf{Hx}\|^2$. Matricea $\mathbf{H}^+$ este pseudoinversa Moore-Penrose a matricii $\mathbf{H}$.

$\left(\mathbf{H}^T\mathbf{HH}^T+\delta^2\mathbf{H}^T=\mathbf{H}^T\left(\mathbf{HH}^T+\delta^2\mathbf{I}\right)=\left(\mathbf{H}^T\mathbf{H}+\delta^2\mathbf{I}\right)\mathbf{H}^T;\left(\mathbf{HH}^T+\delta^2\mathbf{I}\right)\right.$ și $\left.\left(\mathbf{H}^T\mathbf{H}+\delta^2\mathbf{I}\right)\right.$ au ambele inverse (5.2), astfel că $\mathbf{H}^T\left(\mathbf{HH}^T+\delta^2\mathbf{I}\right)^{-1}=\left(\mathbf{H}^T\mathbf{H}+\delta^2\mathbf{I}\right)^{-1}\mathbf{H}^T$. Dacă $\mathbf{z}=\hat{\mathbf{z}}+\tilde{\mathbf{z}}$, $\hat{\mathbf{z}}\in\mathcal{R}(\mathbf{H})$, $\tilde{\mathbf{z}}\in\left(\mathcal{R}(\mathbf{H})\right)^{\perp}$, rezultă că $(\exists)\,\mathbf{x}_0\,|\,\mathbf{Hx}_0=\hat{\mathbf{z}}$. Întrucât $\mathbf{H}^T\mathbf{z}=\mathbf{H}^T\hat{\mathbf{z}}$, $\left(\mathbf{H}^T\mathbf{H}+\delta^2\mathbf{I}\right)^{-1}\mathbf{H}^T\mathbf{z}=\left(\mathbf{H}^T\mathbf{H}+\delta^2\mathbf{I}\right)^{-1}\mathbf{H}^T\mathbf{Hx}_0$. Se arată, utilizând (7.2) pt. matricea simetrică $\mathbf{H}^T\mathbf{H}$, că $\lim_{\delta\to0}\left(\mathbf{H}^T\mathbf{H}+\delta^2\mathbf{I}\right)^{-1}\mathbf{H}^T\mathbf{Hx}_0=\hat{\mathbf{x}}_0$, proiecția lui $\mathbf{x}_0$ pe $\mathcal{R}(\mathbf{H}^T\mathbf{H})=\mathcal{R}(\mathbf{H}^T)$, satisfăcându-se și (9.1.2).)

(9.4) Exemple:

(9.4.1) $\mathbf{H}^+=\mathbf{H}^{-1}$ dacă $\mathbf{H}$ este pătrată și nesingulară.

(9.4.2) $\mathbf{H}^+=\mathbf{H}^T(\mathbf{H}\,\mathbf{H}^T)^{-1}$ dacă liniile lui $\mathbf{H}$ sunt liniar independente ($\mathcal{R}(\mathbf{H})=\mathcal{R}(\mathbf{H}\,\mathbf{H}^T)$, $\mathbf{H}\,\mathbf{H}^T$ nesingulară).

(9.4.3) $\mathbf{H}^+=(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$ dacă coloanele lui $\mathbf{H}$ sunt liniar independente ($\mathcal{R}(\mathbf{H}^T)=\mathcal{R}(\mathbf{H}^T\mathbf{H})$, $\mathbf{H}^T\mathbf{H}$ nesingulară).

(9.4.4) Există cazuri, precum $\mathbf{H}=\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}$, în care $\mathbf{H}^+$ nu are o formulă simplă în termeni de inverse.

(9.4.5) Dacă $\mathbf{H}$ este un vector coloană $\mathbf{H}^{nx1}$ nenul, atunci $\mathbf{H}^+=\dfrac{\mathbf{H}^T}{\mathbf{H}^T\mathbf{H}}=\dfrac{\mathbf{H}^T}{\|\mathbf{H}\|^2}$.

(9.5) Proprietăți definitorii (Boullion și Odell, 1971)

(9.5.1) $\mathbf{H}\,\mathbf{H}^+\mathbf{H}=\mathbf{H}$

(9.5.2) $\mathbf{H}^+\,\mathbf{H}\,\mathbf{H}^+=\mathbf{H}^+$

(9.5.3) $(\mathbf{H}^+\,\mathbf{H})^T=\mathbf{H}^+\,\mathbf{H}$

(9.5.4) $(\mathbf{H}\,\mathbf{H}^+)^T=\mathbf{H}\,\mathbf{H}^+$

Proprietatea (9.5.1) definește inversele generalizate. Dacă în plus e satisfăcută (9.5.2), inversa generalizată este reflexivă. O inversă generalizată slabă la stânga (*left weak generalized inverse*) satisface în plus (9.5.3). O inversă generalizată slabă la dreapta (*right weak generalized inverse*) satisface (9.5.1), (9.5.2) și (9.5.4). Pseudoinversa (Moore-Penrose) satisface (9.5.1)-(9.5.4). Ea este unică.

Proprietăți importante :

(9.5.5) $\mathbf{H}\,\mathbf{H}^{+}$ este matricea proiecție pe $\mathcal{R}(\mathbf{H})$ iar $(\mathbf{I} - \mathbf{H}\,\mathbf{H}^{+})$ este matricea proiecție pe $\mathcal{N}(\mathbf{H}^{T}) = (\mathcal{R}(\mathbf{H}))^{\perp}$.

(9.5.6) $\mathbf{H}^{-}\mathbf{H}$ este matricea proiecție pe $\mathcal{R}(\mathbf{H}^{T})$ iar $(\mathbf{I} - \mathbf{H}^{+}\mathbf{H})$ este matricea proiecție pe $\mathcal{N}(\mathbf{H}) = (\mathcal{R}(\mathbf{H}^{T}))^{\perp}$.

(9.6) Câteva alte proprietăți:

$(\mathbf{H}^{T})^{-} = (\mathbf{H}^{+})^{T}$, $(\forall)\ \mathbf{H}$.

$(\mathbf{H}^{-})^{+} = \mathbf{H}$.

$(\mathbf{H}^{T}\mathbf{H})^{+} = \mathbf{H}^{+}(\mathbf{H}^{T})^{+}$ și $(\mathbf{H}\,\mathbf{H}^{T})^{+} = (\mathbf{H}^{T})^{+}\mathbf{H}^{+}$.

Dacă $\mathbf{H}$ este simetrică și $\alpha > 0$, atunci $(\mathbf{H}^{\alpha})^{+} = (\mathbf{H}^{+})^{\alpha}$.

$\mathcal{R}(\mathbf{H}^{+}) = \mathcal{R}(\mathbf{H}^{+}\mathbf{H}) = \mathcal{R}(\mathbf{H}^{T})$; $\mathcal{N}(\mathbf{H}) = \mathcal{N}(\mathbf{H}^{+}\mathbf{H}) = \mathcal{N}((\mathbf{H}^{T}\mathbf{H})^{+})$.

Dacă $\mathbf{H}$ este simetrică, $\mathbf{H}\mathbf{H}^{+} = \mathbf{H}^{+}\mathbf{H}$.

(9.7) $\mathbf{x}_0$ minimizează $\|\mathbf{z} - \mathbf{H}\mathbf{x}\|^2$ dacă și numai dacă $\mathbf{x}_0$ este de forma $\mathbf{x}_0 = \mathbf{H}^{-}\mathbf{z} + (\mathbf{I} - \mathbf{H}^{-}\mathbf{H})\mathbf{y}$, cu $\mathbf{y}$ un vector oarecare de aceeași dimensiunea cu $\mathbf{z}$.. Valoarea lui $\mathbf{x}_0$ este unică d.d. $\mathbf{H}\mathbf{H}^{-} = \mathbf{I}$. Aceasta se întâmplă doar dacă $\mathcal{N}(\mathbf{H}) = \{\mathbf{0}\}$. (cf. (9.1), (9.5))

(9.8) Ecuația $\mathbf{H}\mathbf{x} = \mathbf{z}$ are soluție d.d. $\mathbf{H}\mathbf{H}^{-}\mathbf{z} = \mathbf{z}$. Soluția este unică d.d $\mathbf{H}\mathbf{H}^{+} = \mathbf{I}$.

($\mathbf{z} \in \mathcal{R}(\mathbf{H})$; $(\mathcal{R}(\mathbf{H}))^{\perp} = \{\mathbf{0}\}$)

(9.9) Dacă $(\mathbf{A}\mathbf{B})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$, $(\mathbf{A}\mathbf{B})^{-} = \mathbf{B}^{-}\mathbf{A}^{-}$ nu e necesar adevărată.

(10) (Rao, 1965) Dacă ecuația $\mathbf{z} = \mathbf{H}\mathbf{x}$, cu $\mathbf{H}^{m\times n}$, $\mathbf{z}^{m\times 1}$, $\mathbf{x}^{n\times 1}$ are soluție, atunci o inversă generalizată sau o pseudoinversă a lui $\mathbf{H}$ este o matrice $\mathbf{H}^{-}$ pentru care $\mathbf{x} = \mathbf{H}^{-}\mathbf{z}$ este o soluție.

Orice matrice are o inversă generalizată. Aceasta nu e necesar unică.

(10.1) Soluția generală a ecuației $\mathbf{H}\mathbf{x} = \mathbf{0}$ este $\tilde{\mathbf{x}} = (\mathbf{I} - \mathbf{H}^{-}\mathbf{H})\mathbf{y}$, cu $\mathbf{y}$ oarecare. Soluția generală a ecuației $\mathbf{z} = \mathbf{H}\mathbf{x}$ este $\mathbf{x} = \hat{\mathbf{x}} + \tilde{\mathbf{x}} = \mathbf{H}\mathbf{z} + (\mathbf{I} - \mathbf{H}^{-}\mathbf{H})\mathbf{y}$, cu $\mathbf{y}$ oarecare.

(10.2) $\mathbf{H}^{-}$ există d.d. $\mathbf{H}\mathbf{H}^{-}\mathbf{H} = \mathbf{H}$, $\mathbf{H}\mathbf{H}^{-}\mathbf{H}\mathbf{H}^{-} = \mathbf{H}\mathbf{H}^{-}$; $\mathbf{H}^{-}\mathbf{H}$ este idempotentă.

(10.3) $\text{rang}(\mathbf{H}^{-}) \geq \text{rang}(\mathbf{H})$. Există $\mathbf{H}^{-}$ a.î. $\text{rang}(\mathbf{H}^{-}) = \min(m,n)$, indiferent de rangul r al lui $\mathbf{H}$.

(10.4) Pseudoinversa Moore-Penrose este o inversă generalizată, ce conține condițiile suplimentare de norme minime, echivalente cu proprietățile de proiecții (9.5); ea este unică (v. și Boullion și Odell, 1971).

(11) (Albert, 1972; Golub și Van Loan, 1989) Folosind proprietățile pseudoinversei și (7.2), se poate demonstra teorema de descompunere după valorile singulare (SVD, *Singular Value Decomposition*) a unei matrici oarecare $\mathbf{A}^{m\times n}$:

Fie $\Lambda^{r\times r}$ matricea diagonală a valorilor proprii nenule λ_j $j=1...r$ ale matricii $\mathbf{A}\mathbf{A}^{T}$, dispuse într-o ordine arbitrară. Atunci există o matrice $\mathbf{P}^{m\times r}$ și o matrice $\mathbf{Q}^{r\times n}$ astfel încât au loc (11.1)-(11.7):

(11.1) $\mathbf{P}^T\mathbf{P} = \mathbf{I}$, vectorii coloană $\mathbf{p}_j^{mx1}$ j=1...r ai matricii $\mathbf{P}$ sunt ortonormaţi.
(11.2) $\mathbf{A}\mathbf{A}^T = \mathbf{P}\Lambda\mathbf{P}^T$, vectorii $\mathbf{p}_j$ sunt vectorii proprii ai $\mathbf{A}\mathbf{A}^T$.
(11.3) $\mathbf{A}\mathbf{A}^+ = \mathbf{P}\mathbf{P}^T$, $\mathbf{P}\mathbf{P}^T$ defineşte aceeaşi proiecţie ca şi $\mathbf{A}\mathbf{A}^+$, pe $\mathcal{R}(\mathbf{A})$.
(11.4) $\mathbf{Q}\mathbf{Q}^T = \mathbf{I}$, vectorii linie $\mathbf{q}_j^{T\,1xn}$ j=1...r ai matricii $\mathbf{Q}$ sunt ortonormaţi.
(11.5) $\mathbf{A}^T\mathbf{A} = \mathbf{Q}^T\mathbf{Q}$. vectorii $\mathbf{q}_j$ sunt vectorii proprii ai matricii $\mathbf{A}^T\mathbf{A}$.
(11.6) $\mathbf{A}^+\mathbf{A} = \mathbf{Q}^T\mathbf{Q}$. $\mathbf{Q}^T\mathbf{Q}$ defineşte aceeaşi proiecţie ca şi $\mathbf{A}^+\mathbf{A}$, pe $\mathcal{R}(\mathbf{A}^T)$.
(11.7) $\mathbf{A} = \mathbf{P}\,\Lambda^{1/2}\,\mathbf{Q}$. $\mathbf{A} = \sum_{j=1}^{r} \sqrt{\lambda_j}\,\mathbf{p}_j\,\mathbf{q}_j^T$.

(12) (Albert, 1972; Golub şi Van Loan, 1989) Cu (11), pseudoinversa Moore-Penrose a matricii $\mathbf{A}$ poate fi exprimată ca

$$\mathbf{A}^+ = \mathbf{Q}^T\Lambda^{-1/2}\mathbf{P}^T = \sum_{j=1}^{r} \sqrt{\lambda_j}\,\mathbf{q}_j\,\mathbf{p}_j^T$$

(13.) Fie sistemul de ecuaţii :

$$\mathbf{y}(k) = \mathbf{W}\,\mathbf{x}(k),\ k=1, 2, \ldots, K, \tag{13.1}$$

unde $\mathbf{x}(k)\in\mathbf{R}^n$, $\mathbf{y}(k)\in\mathbf{R}^m$.

Acesta poate fi scris sub forma matricială :

$$\mathbf{Y} = \mathbf{W}\,\mathbf{X}, \tag{13.2}$$

unde $\mathbf{Y}\in\mathbf{R}^{mxK}$ şi $\mathbf{X}\in\mathbf{R}^{nxK}$ conţin cei K vectori coloană y, respectiv x.

Dacă vectorii y(k) sunt zgomotoşi, atunci (13.1) sau (13.2) admite doar o soluţii aproximative. Aproximaţia de eroare pătratică minimă este dată de (Kohonen, 1984) :

$$\mathbf{W}_*(K) = \arg\min_{\hat{\mathbf{W}}} \left\| \mathbf{Y}(K) - \hat{\mathbf{W}}\,\mathbf{X}(K) \right\|_E \tag{13.3}$$

$$\mathbf{W}_*(K) = \mathbf{Y}(K)\,\mathbf{X}^+(K), \tag{13.4}$$

unde $\|\cdot\|_E$ este norma matricială euclidiană sau Frobenius matrix norm (v. 15), iar $\mathbf{X}^+$ este pseudoinversa Moore-Penrose a matricii $\mathbf{X}$.

Dacă sistemul (13.1) sau (13.2) este nedeterminat, atunci (13.4) este soluţia de normă euclidiană minimă.

(14). Fie matricea $\mathbf{X}(k)$, ce se construieşte pe măsura acumulării vectorilor x(k) :

$$\mathbf{X}(k) = \left[\mathbf{x}(1) \quad \cdots \quad \mathbf{x}(k-1) \quad \mathbf{x}(k)\right] = \left[\mathbf{X}(k-1) \quad \mathbf{x}(k)\right] \tag{14.1}$$

Pseudoinversa $\mathbf{X}^+(k)$, poate fi calculată folosind algoritmul recursiv cunoscut sub numele de teorema lui Greville (Kohonen, 1984; Boullion şi Odell, 1971) :

$$\mathbf{X}^{+}(k)=\begin{bmatrix}\mathbf{X}^{+}(k-1)\left(\mathbf{I}-\mathbf{x}(k)\,\mathbf{p}^{T}(k)\right)\\ \mathbf{p}^{T}(k)\end{bmatrix}, \tag{14.2}$$

cu p(k) dat de :

$$\mathbf{p}(k)=\begin{cases}\dfrac{\left(\mathbf{I}-\mathbf{X}(k-1)\,\mathbf{X}^{+}(k-1)\right)\mathbf{x}(k)}{\left\|\left(\mathbf{I}-\mathbf{X}(k-1)\,\mathbf{X}^{+}(k-1)\right)\mathbf{x}(k)\right\|_1^2}, \\ \qquad\qquad \text{daca numitorul } \neq 0 \\ \dfrac{\mathbf{X}^{+T}(k-1)\,\mathbf{X}^{+}(k-1)\,\mathbf{x}(k)}{1+\left\|\mathbf{X}^{+}(k-1)\,\mathbf{x}(k)\right\|_1^2} \quad \text{în caz contrar.}\end{cases} \tag{14.3}$$

În (Albert, 1971; Boullion şi Odell, 1971) se dă modul de calcul al pseudoinversei Moore-Penrose a unei matrici partiţionate. $[\mathbf{U}\ \ \mathbf{V}]^{+}$, pe baza pseudinversei $\mathbf{U}^{+}$ şi a pseudoinversei proiecţiei lui V în $(\mathcal{R}(\mathbf{U}))^{\perp}$.

(15). Norme vectoriale şi matriciale (Golub şi Van Loan, 1989).

(15.1.1) O normă vectorială pe $\mathbf{R}^n$ este o funcţieO $f: \mathbf{R}^n \to \mathbf{R}$, $f(\mathbf{x})=\|\mathbf{x}\|$, ce satisface următoarele proprietăţi :

$f(\mathbf{x}) \geq 0, (\forall)\ \mathbf{x}\in\mathbf{R}^n ; f(\mathbf{x}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}$

$f(\mathbf{x}+\mathbf{y}) \leq f(\mathbf{x})+f(\mathbf{y})$, $(\forall)\ \mathbf{x}, \mathbf{y} \in \mathbf{R}^n$

$f(\alpha\mathbf{x}) = |\alpha| f(\mathbf{x}), (\forall)\ \alpha\in\mathbf{R}, \mathbf{x}\in\mathbf{R}^n$

(15.1.2). O clasă des utilizată o constituie normele p :

$$\|\mathbf{x}\|_p = \left(|x_1|^p + \ldots + |x_n|^p\right)^{1/p}, \quad p \geq 1.$$

(15.1.3) Dintre acestea, cele mai importante sunt normele 1, 2 şi ∞ :

$$\|\mathbf{x}\|_1 = |x_1| + \ldots + |x_n|,$$

$$\|\mathbf{x}\|_2 = \left(|x_1|^2 + \ldots + |x_n|^2\right)^{1/2} = \sqrt{\mathbf{x}^T\mathbf{x}},$$

$$\|\mathbf{x}\|_\infty = \max_{1\leq i\leq n} |x_i|.$$

(15.1.4) Toate normele pe $\mathbf{R}^n$ sunt echivalente : dacă $\|\cdot\|_\alpha$ şi $\|\cdot\|_\beta$ sunt norme pe $\mathbf{R}^n$, atunci există constantele pozitive c_1 şi c_2 astfel încât :

$c_1\|\mathbf{x}\|_\alpha \leq \|\mathbf{x}\|_\beta \leq c_2\|\mathbf{x}\|_\alpha$.

De exemplu.

$$\|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1 \leq \sqrt{n}\|\mathbf{x}\|_2,$$

$$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \sqrt{n}\|\mathbf{x}\|_\infty,$$

$$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_1 \leq n\|\mathbf{x}\|_\infty.$$

(15.2.1) O normă matricială pe $\mathbf{R}^{m\times n}$ este o funcţie $f: \mathbf{R}^{m\times n} \to \mathbf{R}$, $f(\mathbf{A})=\|\mathbf{A}\|$, ce satisface următoarele proprietăţi :

$f(\mathbf{A}) \geq 0, (\forall)\ \mathbf{x}\in\mathbf{R}^{m\times n} ; f(\mathbf{x}) = 0 \Leftrightarrow \mathbf{A} = \mathbf{0}$

$f(\mathbf{A}+\mathbf{B}) \le f(\mathbf{A})+f(\mathbf{B})$, $(\forall)\ \mathbf{A}, \mathbf{B} \in \mathbf{R}^{m\times n}$

$f(\alpha\mathbf{A}) = |\alpha| f(\mathbf{A})$, $(\forall)\ \alpha\in\mathbf{R}$, $\mathbf{A}\in\mathbf{R}^{m\times n}$

(15.2.2) Cele mai utilizate norme matriciale sunt norma Frobenius sau norma matricială euclidiană.

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2},$$

şi normele p. definite în raport cu norma vectorială p :

$$\|\mathbf{A}\|_p = \sup_{x\neq 0}\frac{\|\mathbf{Ax}\|_p}{\|\mathbf{x}\|_p} = \sup_{x\neq 0}\left\|\mathbf{A}\frac{\mathbf{x}}{\|\mathbf{x}\|_p}\right\|_p = \max_{\|x\|_p=1}\|\mathbf{Ax}\|_p.$$

(15.2.3) Observaţii :

1. Normele $\|\mathbf{A}^{m\times n}\|_p$ sunt familii de funcţii de **A**. ce depind de m. n şi p. Astfel. inegalitatea uşor verificabilă

$$\|\mathbf{AB}\|_p \le \|\mathbf{A}\|_p \|\mathbf{B}\|_p,\quad \mathbf{A}\in\mathbf{R}^{m\times n},\ \mathbf{B}\in\mathbf{R}^{n\times q}$$

este o relaţie între trei familii diferite de norme matriciale.

2. Nu toate normele matriciale au proprietatea de a fi submultiplicative. $\|\mathbf{AB}\| \le \|\mathbf{A}\|\,\|\mathbf{B}\|$. Normele p au această proprietate.

3. Normele p au proprietatea importantă :

$$\|\mathbf{Ax}\|_p \le \|\mathbf{A}\|_p \|\mathbf{x}\|_p,\ (\forall)\ x\in\mathbf{R}^n,\ \mathbf{A}\in\mathbf{R}^{m\times n}.$$

4. Pentru orice α, $\|\cdot\|_\alpha : \mathbf{R}^n$ şi β, şi $\|\cdot\|_\beta : \mathbf{R}^m$, se poate defini :

$$\|\mathbf{A}\|_{\alpha\beta} = \sup_{x\neq 0}\frac{\|\mathbf{Ax}\|_\beta}{\|\mathbf{x}\|_\alpha},$$

cu proprietatea

$$\|\mathbf{Ax}\|_\beta \le \|\mathbf{A}\|_{\alpha\beta}\|\mathbf{x}\|_\alpha.$$

(15.2.4) Toate normele pe $\mathbf{R}^{m\times n}$ sunt echivalente. În particular. pentru $\mathbf{A}\in\mathbf{R}^{m\times n}$. au loc relaţiile :

$$\|\mathbf{A}\|_1 = \max_{1\le j\le n}\sum_{i=1}^{m}|a_{ij}|,$$

$$\|\mathbf{A}\|_\infty = \max_{1\le i\le m}\sum_{j=1}^{n}|a_{ij}|,$$

$$\|\mathbf{A}\|_2 \le \|\mathbf{A}\|_F \le \sqrt{n}\,\|\mathbf{A}\|_2,$$

$$\max_{i,j}|a_{ij}| \le \|\mathbf{A}\|_2 \le \sqrt{mn}\,\max_{i,j}|a_{ij}|,$$

$$\frac{1}{\sqrt{n}}\|\mathbf{A}\|_\infty \le \|\mathbf{A}\|_2 \le \sqrt{m}\,\|\mathbf{A}\|_\infty,$$

$$\frac{1}{\sqrt{m}}\|\mathbf{A}\|_1 \le \|\mathbf{A}\|_2 \le \sqrt{n}\,\|\mathbf{A}\|_1.$$

(15.2.5) Spre deosebire de normele 1 şi ∞. norma 2 se calculează mai dificil. Anume. ea este rădăcina pătrată a celei mai mari valori proprii a matricii $\mathbf{A}^T\mathbf{A}$. O estimare a ordinului său de mărime poate fi obţinută cu relaţia :

$$\|\mathbf{A}\|_2 \le \sqrt{\|\mathbf{A}\|_1\|\mathbf{A}\|_\infty}.$$

Anexa II. Listingul unor fişiere utilizate în simulările MATLAB

În această anexă sunt listate fişierele utilizate în simulările MATLAB corespunzătoare subcapitolului 4.3. Acestea sunt, în ordinea crescătoare a complexităţii :
- fişierul ERICCR.m, corespunzând modelării din § 2.1.1, (2.1.1) ;
- fişierul view2m.m, corespunzând modelării din § 2.1.1, (2.1.9), (2.1.11) ;
- fişierele JcRLS.m şi Jcinit.m- § 4.3.1, corespunzând algoritmului (4.3.2) ;
- fişierul aqRLS.m - § 4.3.1, corespunzând algoritmului (4.3.1).

fişierul ERICCR.m

```
function [r,Jr,e, th]=ericcr(th)
% functie ce modeleaza manipulatorul ERICC,
% nenormalizat cu L2+L3
%   th-vectorul unghiurilor articulatiilor (3x1) [rad]
%   r-pozitia euclidiana (3x1) [mm]
%  Jr-matricea jacobiana a robotului [dri/dthj] (3x3)
% e-cod de eroare:
%           0: OK
%           1:th1<th1min
%           2:th1<th1max
%           4:th1<th2min
%           8:th1<th2max
%           16:th1<th1min
%           32:th1<th1max
%           64:x1<x1min

% A.C. 29/06/95

l1=0; l2=280; l3=317+150; %mm, l1=340
th1m=-3*pi/4; th1M=3*pi/4;
th2m=-pi/4; th2M=pi/2;
th3m=-pi/4; th3M=pi/2-pi/6;
x1min=140;

e=0;
 if th(1)<th1m; e=1; end; % th(1)=th1m;
 if th(1)>th1M; e=2; end; % th(1)=th1M;
 if th(2)<th2m; e=4; end; % th(2)=th2m;
 if th(2)>th2M; e=8; end; % th(2)=th2M;
 if th(3)<th3m; e=16; end; % th(3)=th3m;
 if th(3)>th3M; e=32; end; % th(3)=th3M;

sin1=sin(th(1)); cos1=cos(th(1));
sin2=sin(th(2)); cos2=cos(th(2));
sin3=sin(th(3)); cos3=cos(th(3));
sin23=sin(th(2)+th(3)); cos23=cos(th(2)+th(3));
x1=l2*cos2+l3*sin23;
```

```
y1=l2*sin2-l3*cos23;

% 1/(l2+l3)*
x=x1*cos1;
z=x1*sin1;
y=y1+l1;

r= [x y z]';

    dx1dt2=-l2*sin2+l3*cos23;
    dx1dt3=l3*cos23;
    dy1dt2=l2*cos2+l3*sin23;
    dy1dt3=l3*sin23;
    dxdt1=-x1*sin1;
    dxdt2=dx1dt2*cos1;
    dxdt3=dx1dt3*cos1;
    dydt1=0;
    dydt2=dy1dt2;
    dydt3=dy1dt3;
    dzdt1=x1*cos1;
    dzdt2=dx1dt2*sin1;
    dzdt3=dx1dt3*sin1;

    Jr=[dxdt1 dxdt2 dxdt3; dydt1 dydt2 dydt3; dzdt1 dzdt2 dzdt3];

 if x1<=x1min; e=64; end;
```

fişierul view2m.m

```
function [v,Jv,e]=view2m(r)
% functie ce modeleaza vederea celor doua camere
% in forma matriciala
% intoarce nr. echivalent al pixelului activat
% r=[x1 x2 x3]' pozitia euclidiana a punctului [mm]
% v=[h1,v1,h2,v2]' vectorul vizual [pixeli]
% Jv=matricea jacobiana a transformarii vizuale (dv1/drj), (4x3)
% e=cod de eroare:
%           0:  OK
%           1:  h1<h1min
%           2:  h1>h1max
%           4:  v1<v1min
%           8:  v1>v1max
%           16: h2<h2min
%           32: h2>h2max
%           64: v2<v2min
%           128:v2>v2max
%

% A.C. 29/06/95 revazut 24/11/95

% camerele B & C
w=500; % distanta dintre ele este 2w
e=5000; % distanta dintre linia ce le uneste la axa x a robotului
D=17.5; % distanta focala
qh=12.7e-3; % cuantizarea pe orizontala [mm]
qv=8.3e-3; % cuantizarea pe verticala [mm]
H=6.4/2/qh; % aria sensibila [mm]
```

```
V=4.8/2/qv;  % aria sensibila [mm]

Jv=zeros(4,3);
qq=[1/qh 1/qv]';
Q=[1/qh 1/qv 1/qh 1/qv]'; % scari CCD (cuantizare)
I46=[eye(2,3) zeros(2,3);zeros(2,3) eye(2,3)];
I43=[eye(2,3); eye(2,3)];
al=atan(w/e);

oB=[350-w 0 -e]'; % centrul camerei B
u1B=[cos(al) 0 -sin(al)]'; % orientarea
u2B=[0 1 0]';
u3B=[sin(al) 0 cos(al)]';
RB=[u1B u2B u3B];

oC=[350+w 0 -e]'; % centrul camerei C
u1C=[cos(al) 0 sin(al)]'; % orientarea
u2C=[0 1 0]';
u3C=[-sin(al) 0 cos(al)]';
RC=[u1C u2C u3C];

% proiectiile optice
rB=RB'*(r-oB)/(u3B'*(r-oB));
rC=RC'*(r-oC)/(u3C'*(r-oC));

v=(D*diag(Q)*I46*[rB; rC]);

kB=1/(u3B'*(r-oB));
JB=kB*RB'*(eye(3) - kB*(r-oB)*u3B');
kC=1/(u3C'*(r-oC));
JC=kC*RC'*(eye(3) - kC*(r-oC)*u3C');

Jv=D*diag(Q)*I46*[JB;JC];

e=0;
```

fişierul JcRLS.m

```
function [Jc]=JcRLS(th1)
% invata Jc prin comanda separata a fiecarei axe si RLS

l2dth=0.001; % lungimea vectorului comanda dth [rad]
delta=1e-12; % initializari RLS
Jc=zeros(3,4); Ri=1/delta*eye(4,4);

%[r1,Jr1,e,th3]=ericc2(th1);
[r1,Jr1,e,th3]=ericcr(th1);
[v1,Jv1,e]=view2m(r1);

for k=1:3
%  dth2=ones(3,1)-2*rand(3,1);
%  if(norm(dth2)>l2dth), dth2=l2dth/norm(dth2)*dth2; end; % normalizare

  dth2=zeros(3,1); dth2(k,1)=l2dth;

  % deplasare si privire
  th2=th1+dth2; % noua pozitie unghiulara
```

```
  [r2,Jr2,e,th]=ericcr(th2); dr2=r2-r1;
%  [r2,Jr2,e,th]=ericc2(th1,th2); dth2=th-th1, dr2=r2-r1;
  [v2,Jv2,e]=view2m(r2); dv2=v2-v1;

  % algoritmul RLS
  h=Ri*dv2; G=h/(1+dv2'*h);
  Riv=Ri; Ri=Ri-G*h';
  Jc=Jc+(dth2-Jc*dv2)*G';
end % for k

% comparare cu valoarea adevarata
%  C=pinv(Jv1*Jr1); errelJc=norm(Jc-C)/norm(C)
%  condC=cond(C)
```

fişierul Jcinit.m

```
function [thmem,Jc,tabth,tabJc,er]=Jcinit(th,thmem,tabth,tabJc)
% functie ce initializeaza Jc
% argumente :
%       th - pozitia curenta
%       thmem - pozitia vecinului utilizat precedent
%            thmem(:) = inf : primul punct de pe traiectorie
%            thmem(:) = -inf : se cere esantionare in punctul respectiv
%       tabth, tabJc - tabelele de memorare
% valori returnate :
%       thmem - pozitia vecinului celui mai apropiat
%       Jc - matricea jacobiana citita sau calculata
%      er - cod de eroare :
%            0 : ok
%            1 : pozitie curenta prea apropiata de vecin

[v,puncte]=size(tabth);
dmin=0.01; % densitatea maxima de puncte 0.2 [rad] = 11.5 [deg]
er=0;

% tabel gol ?
if puncte==0
  thmem=th;
  Jc=JcRLS(th);
  tabth=[tabth th];
  tabJc=[tabJc Jc];
'O'
  return
end

% cel mai apropiat punct memorat
distmin=inf;
for i=1:puncte
  dist=norm(tabth(:,i)-th);
  if(dist <= distmin)
    distmin=dist;
    indice=i;
  end
end

% inceputul traiectoriei curente ?
if thmem(1) == inf
```

```
'A'
  thmem=tabth(:,indice); % se citeste
  Jc=tabJc(:,4*(indice-1)+1:4*indice);
  return
end

% se cere esantionare ?
if thmem(1) == -inf
  if distmin<=dmin % e prea apropiat ?
'E'
    er=1;
    thmem=tabth(:,indice);
    Jc=tabJc(:,4*(indice-1)+1:4*indice);
    return
  else % se masoara punctul cerut
'F'
    tabth=[tabth th];
    Jc=JcRLS(th);
    thmem=th;
    tabJc=[tabJc Jc];
    return
  end
end

% e punctul tocmai utilizat ?
if(norm(tabth(:,indice)-thmem) <= 100*eps)
  % da:
  if distmin<=dmin % e prea apropiat ?
'B'
    er=1;
    thmem=tabth(:,indice);
    Jc=tabJc(:,4*(indice-1)+1:4*indice);
    return
  else % necesar punct nou
'C'
    tabth=[tabth th];
    Jc=JcRLS(th);
    thmem=th;
    tabJc=[tabJc Jc];
    return
  end
else % nu: se citeste
'D'
  thmem=tabth(:,indice);
  Jc=tabJc(:,4*(indice-1)+1:4*indice);
  return
end
```

fişierul aqRLS.m

```
% aqrls.m
% cuantizare adaptiva cu RLS
% jacobienele sunt calculate prin RLS
%  si actualizate prin LMS
% memorare doar la nevoie
% 4.11.96
```

```
clear all

% variabile de rulare
N_traiectorii=6; % numarul de traiectorii
N_rulari=1; % numarul de rulari

criteriu='B'; % A: netezime, B: scaderea erorii
varianta='2'; % 1: de principiu, 2: practica
cos_min=0.7; % cosinusul unghiului 4D minim pt. criteriul 'A'
qmax=0.8; % rata maxima de descrestere a erorii pt. 'B'
LMS=1; % se face (1) sau nu (0) adaptare LMS

figuri=1; % se deseneaza (1) sau nu (0) figurile
l2dv=10; % lungimea vectorului dv dorit [pixeli]
l2dth=0.25; % lungimea maxima a vectorului comanda dth [rad]
errend=1e-1; % eroarea maxima la sfirsitul pozitionarii
FontSize=8; % marimea caracterelor din ferestrele grafice
rand('seed',17); % initializarea generatorului aleator

% limitele lui ERICC
thm=[-pi/3 -pi/4 -pi/4]';
thM=[pi/3 pi/2 pi/2-pi/6]';

% matricile ce memoreaza rularile
N_p_mem=[]; N_p_adaug=[]; % istoria nr. de memorari
N_p_inite=[]; N_p_inite_tot=[]; % istoria nr. de initializari
Err=[]; % eroarea, memorata pentru toate traiectoriile si rularile

% initializarea figurilor
if N_rulari==1
  h1=figure(1); clf; hold on % spatiul articulatiilor
%  set(h1,'Position', [884 454 356 512])
  h2=figure(2); clf;% hold on % coordonate carteziene
%  set(h2,'Position', [884 454 356 512])
  h3=figure(3); clf; hold on % imaginile CCD
%  set(h3,'Position', [884 454 356 512])
  h4=figure(4); clf; % hold on % evolutia erorii
%  set(h4,'Position', [884 454 356 512])
  h5=figure(5); clf; hold on % evolutia initializarilor si memorarilor
%  set(h5,'Position', [884 454 356 512])
  h6=figure(6); clf; hold on % punctele initializate si memorate
%  set(h5,'Position', [884 454 356 512])
end
tic % initializarea duratei de executie

% nr. de rulari
for rulare=1:N_rulari;
  rulare

  % memorari
  tabth=[]; tabJc=[]; % tabelele de memorare pt. th si Jc
  p_mem=[]; p_adaug=[]; % istoria nr. de memorari
  p_inite=[]; p_inite_tot=[]; % istoria nr. de initializari
  tab_init=[];

  % traiectorii
  for t=1:N_traiectorii
    traiectorie=t
```

```
    [l p_initiale]=size(tabth);

    % genereaza o pozitie initiala aleatoare admisibila
    % dintr-un vector theta propagat inainte
    e0=1; while e0,

      % simularea 1
      if t==1, th0 =[0 0 0]'; end
      if t==2, th0 =[pi/4 pi/4 pi/4]'; end
      if t==3, th0 =[0 0 0]'; end
      if t==4, th0 =[pi/4 pi/4 pi/4]'; end
      if t==5, th0 =[0 0 0]'; end
      if t==6, th0 =[pi/4 pi/4 pi/4]'; end

      % simularea 2 si 3
%      th0=thm+rand(3,1).*(thM-thm);

      [r0,Jr1,e0,th0]=ericcr(th0);
    end; % while e0
    [v0,Jv1,e]=view2m(r0);

    % genereaza o tinta aleatoare admisibila
    e0=1; while e0,
      thtarg=thm+rand(3,1).*(thM-thm);

      % simularea 1
      if t==1, thtarg =[pi/4 pi/4 pi/4]'; end
      if t==2, thtarg =[0 0 0]'; end
      if t==3, thtarg =[pi/4 pi/4 pi/4]'; end
      if t==4, thtarg =[0 0 0]'; end
      if t==5, thtarg =[pi/4 pi/4 pi/4]'; end
      if t==6, thtarg =[0 0 0]'; end

      % simularea 2
%      thtarg =[0 pi/4 -pi/6]';

      % simularea 3
%      thtarg=thm+rand(3,1).*(thM-thm);

      [rtarg,Jr2,e0,thtarg]=ericcr(thtarg);
    end; % while e0
    [vtarg,Jv2,e]=view2m(rtarg);
    err0=norm(vtarg-v0); % eroarea initiala

    % matrici pentru memorarea traiectoriei curente
    Th1=[th0]; R1=[r0]; V1=[v0];
    Err=[Err err0]; % grafice serie
%     Err=[err0]; % grafice paralel

    % initializarea variabilelor din bucla
    v1=v0; th1=th0; r1=r0; v1=v0; err1=err0; iter=0;
    thmem=inf*ones(3,1);  initializari=1; errmin=inf;

    % start
    [thmem,Jc,tabth,tabJc,er]=Jcinit(th1,thmem,tabth,tabJc); % invata Jc
initial
    [rmem,Jr,e,th]=ericcr(thmem);
    [vmem,Jv,e]=view2m(rmem);
```

```
%   Jc=pinv(Jv*Jr); % valoarea adevarata, pt. test
    initth=th1; initr=r1; initv=v1; % memorarea pozitiilor initializate

    % bucla de reactie iterativa
    while err1>errend

      % deplasarea de efectuat
      dv1=(vtarg-v1);
      if(norm(dv1)>l2dv), dv1=l2dv/norm(dv1)*dv1; end; % normalizare
      dth1=Jc*dv1;  % comanda
      if(norm(dth1)>l2dth), dth1=l2dth/norm(dth1)*dth1; end; % normalizare

      % deplasare si privire
      th2=th1+dth1;
      [r2,Jr2,eth,th]=ericcr(th2); dth2=dth1; dr2=r2-r1;
      [v2,Jv2,er]=view2m(r2); dv2=v2-v1;
      err2=norm(vtarg-v2);

      % adaptarea prin LMS optimizat
      eLMS=dth2-Jc*dv2;
      if LMS==1, Jc=Jc+eLMS*dv2'/(dv2'*dv2); end

      % criterii de test
      if criteriu=='A'
        cos=dv2'*dv1/norm(dv2)/norm(dv1) % netezimea
        if cos>cos_min
          pas_corect=1;
        else
          pas_corect=0;
        end
      elseif criteriu=='B'
        q1=norm(dv1-dv2)/norm(dv1); % scaderea erorii
        if q1<qmax
          pas_corect=1;
        else
          pas_corect=0;
        end
      end % criteriu 'A'/'B

      if varianta=='1'
        if pas_corect==1
          % actualizarea pozitiei
          th1=th2; r1=r2; v1=v2; err1=err2;
          imin=iter; %errmin=err2;
        else % pas gresit
          if iter-imin>1 % al doilea pas gresit
            thmem=-inf*ones(3,1); % esantioneaza aici
            imin=iter;
          end % if iter-imin
        % initializeaza Jc
          [thmem,Jc,tabth,tabJc,er]=Jcinit(th1,thmem,tabth,tabJc);
          [rmem,Jr,e,th]=ericcr(thmem);
          [vmem,Jv,e]=view2m(rmem);
          initth=[initth th1]; % memorarea pozitiilor initializate
          initr=[initr r1]; initv=[initv v1];
          initializari= initializari+1;
        end % pas_corect/gresit
        % end varianta 1
```

```
    elseif varianta=='2'
      % actualizarea pozitiei
      th1=th2; r1=r2; v1=v2; err1=err2;
      imin=iter; %errmin=err2;
      if pas_corect~=1 % pas gresit
      % initializeaza Jc
        [thmem,Jc,tabth,tabJc,er]=Jcinit(th1,thmem,tabth,tabJc);
        thmem'
        [rmem,Jr,e,th]=ericcr(thmem);
        [vmem,Jv,e]=view2m(rmem);
        initth=[initth th1]; % memorarea pozitiilor initializate
        initr=[initr r1]; initv=[initv v1];
        initializari= initializari+1;
      end % if pas_corect
    end % varianta '1'/'2'

    %pas_corect
    C=pinv(Jv*Jr);
    errelJc=norm(Jc-C)/norm(C);
    erreldth=norm(dth2-C*dv2)/norm(dth2);

    % memorarea traiectoriei
    Th1=[Th1 th1]; R1=[R1 r1]; V1=[V1 v1]; Err=[Err err1];
    iter=iter+1;

  end % while err1

  % numarul de initializari si memorari
  [l p_fin]=size(tabth);
  p_mem=[p_mem p_fin];
  p_adaug=[p_adaug p_fin-p_initiale];
  p_inite=[p_inite  initializari];
  p_inite_tot=[p_inite_tot  sum(p_inite)];

  % punctele memorate in coordonate ERICC si CCD
  tabr=zeros(3,p_fin);
  tabv=zeros(4,p_fin);
  for i=1:p_fin
    tabr(:,i)=ericcr(tabth(:,i));
    tabv(:,i)=view2m(tabr(:,i));
  end

  % grafice

  if N_rulari==1
   if figuri==1
    figure(1);
    h11=subplot(211); hold on;
    set(h11,'FontSize',FontSize)
    title('Articulatiile: th1-th2')
    xlabel('th1 [rad]'), ylabel('th2 [rad]')
    plot(thtarg(1),thtarg(2),'wx') % tinta
    plot(Th1(1,:),Th1(2,:),'b-') % traiectoria curenta
    plot(Th1(1,:),Th1(2,:),'w.') % traiectoria curenta
    plot(initth(1,:),initth(2,:),'wo') % punctele initializate
    plot(tabth(1,:),tabth(2,:),'w+') % punctele memorate
    h111=text(th0(1),th0(2),['  ' num2str(traiectorie)]); % pozitia
initiala
```

```
set(h111,'FontSize',FontSize,'Color','g','HorizontalAlignment','Left')

      h12=subplot(212); hold on;
      set(h12,'FontSize',FontSize)
      title('Articulatiile: th1-th3')
      xlabel('th1 [rad]'), ylabel('th3 [rad]')
      plot(thtarg(1),thtarg(3),'wx') % tinta
      plot(Th1(1,:),Th1(3,:),'b-') % traiectoria curenta
      plot(Th1(1,:),Th1(3,:),'w.') % traiectoria curenta
      plot(initth(1,:),initth(3,:),'wo') % punctele initializate
      plot(tabth(1,:),tabth(3,:),'w+') % punctele memorate
      h121=text(th0(1),th0(3),['  ' num2str(traiectorie)]); % pozitia
initiala

set(h121,'FontSize',FontSize,'Color','g','HorizontalAlignment','Left')

      figure(2)
      h21=subplot(211); hold on
      set(h21,'FontSize',FontSize);
      title('Coordonate carteziene: planul x1-x2')
      xlabel('x1 [mm]'), ylabel('x2 [mm]')
      plot(rtarg(1),rtarg(2),'wx') % tinta
      plot(R1(1,:),R1(2,:),'b-') % traiectoria curenta
      plot(R1(1,:),R1(2,:),'w.') % traiectoria curenta
      plot(initr(1,:),initr(2,:),'wo') % punctele initializate
      plot(tabr(1,:),tabr(2,:),'w+') % punctele memorate
      h211=text(r0(1),r0(2),['  ' num2str(traiectorie)]); % pozitia
initiala

set(h211,'FontSize',FontSize,'Color','g','HorizontalAlignment','Left')

      h22=subplot(212); hold on
      set(h22,'FontSize',FontSize);
      title('Coordonate carteziene: planul x1-x3')
      xlabel('x1 [mm]'), ylabel('x3 [mm]')
      plot(rtarg(1),rtarg(3),'wx') % tinta
      plot(R1(1,:),R1(3,:),'b-') % traiectoria curenta
      plot(R1(1,:),R1(3,:),'w.') % traiectoria curenta
      plot(initr(1,:),initr(3,:),'wo') % punctele initializate
      plot(tabr(1,:),tabr(3,:),'w+') % punctele memorate
      h121=text(r0(1),r0(3),['  ' num2str(traiectorie)]); % pozitia
initiala

set(h121,'FontSize',FontSize,'Color','g','HorizontalAlignment','Left')

      figure(3)
      h31=subplot(211);hold on
      set(h31,'FontSize',FontSize)
      title('camera CCD 1')
      xlabel('h1 [pixeli]')
      ylabel('v1 [pixeli]')
      plot(vtarg(1),vtarg(2),'wx') % tinta
      plot(V1(1,:),V1(2,:),'b-') % traiectoria curenta
      plot(V1(1,:),V1(2,:),'w.') % traiectoria curenta
      plot(initv(1,:),initv(2,:),'wo') % punctele initializate
      plot(tabv(1,:),tabv(2,:),'w+') % punctele memorate
```

```
      h311=text(v0(1),v0(2),['  ' num2str(traiectorie)]); % pozitia
initiala

set(h311,'FontSize',FontSize,'Color','g','HorizontalAlignment','Left')

      h32=subplot(212);hold on
      set(h32,'FontSize',FontSize)
      title('camera CCD 2')
      xlabel('h2 [pixeli]')
      ylabel('v2 [pixeli]')
      plot(vtarg(3),vtarg(4),'wx') % tinta
      plot(V1(3,:),V1(4,:),'b-') % traiectoria curenta
      plot(V1(3,:),V1(4,:),'w.') % traiectoria curenta
      plot(initv(3,:),initv(4,:),'wo') % punctele initializate
      plot(tabv(3,:),tabv(4,:),'w+') % punctele memorate
      h321=text(v0(3),v0(4),['  ' num2str(traiectorie)]); % pozitia
initiala

set(h321,'FontSize',FontSize,'Color','g','HorizontalAlignment','Left')


      figure(4)
      h41=subplot(211);
      set(h41,'FontSize',FontSize)
      plot(Err,'w-'), hold on
      title('Eroarea de pozitionare')
      xlabel('iteratii')
      ylabel('pixeli, scara liniara')
      h42=subplot(212);
      set(h42,'FontSize',FontSize)
      semilogy(Err,'w-'), hold on
      xlabel('iteratii')
      ylabel('pixeli, scara logaritmica')

      figure(5),clf
      h51=subplot(211);
      set(h51,'FontSize',FontSize)
      hold on
      plot(p_adaug,'w+'); plot(p_inite,'wo'); plot(0,-1,'b.')
      title('Numarul punctelor initializate si adaugate')
      xlabel('traiectorii')

      h52=subplot(212);hold on
      set(h52,'FontSize',FontSize)
      plot(p_mem,'w+'); plot(p_inite_tot,'wo'); plot(0,-1,'b.')
      title('Numarul total al punctelor initializate si memorate')
      xlabel('traiectorii')

     end % if figuri
      % afisarea punctelor initializate si memorate
      figure(6);hold on
      h61=subplot(121); hold on;
      set(h61,'FontSize',FontSize)
      title('Articulatiile: th1-th2')
      xlabel('th1 [rad]'), ylabel('th2 [rad]')
      plot(initth(1,:),initth(2,:),'w.') % punctele initializate
      for i=1:p_fin
        plot(tabth(1,i),tabth(2,i),'wo') % punctele memorate
```

```
        h611=text(tabth(1,i),tabth(2,i),['   ' num2str(i)]); % pozitia
initiala

set(h611,'FontSize',FontSize,'Color','g','HorizontalAlignment','Left')
      end

      h62=subplot(122); hold on;
      set(h62,'FontSize',FontSize)
      title('Articulatiile: th1-th3')
      xlabel('th1 [rad]'), ylabel('th3 [rad]')
      plot(initth(1,:),initth(3,:),'w.') % punctele initializate
      for i=1:p_fin
        plot(tabth(1,i),tabth(3,i),'wo') % punctele memorate
        h621=text(tabth(1,i),tabth(3,i),['   ' num2str(i)]); % pozitia
initiala

set(h621,'FontSize',FontSize,'Color','g','HorizontalAlignment','Left')
      end

    end % N_rulari

  end % for traiectorii

  N_p_inite=[N_p_inite; p_inite]; % istoria nr. de initializari
  N_p_adaug=[N_p_adaug; p_adaug]; % istoria nr. de memorari
  N_p_inite_tot=[N_p_inite_tot; p_inite_tot]; % istoria nr. total de
initializari
  N_p_mem=[N_p_mem; p_mem]; % istoria nr. total de memorari

  % rezultate
  if rulare==1 % listare pentru tabel
    traiectorii=1:N_traiectorii
    N_p_inite
    N_p_adaug
    N_p_inite_tot
    N_p_mem
  else
    % numarul punctelor initializate si memorate
    h6=figure(1), clf,hold on, grid on
    set(h6,'Position',[644 628 616 322])
    traiectorii=1:N_traiectorii;
    h61=subplot(121); hold on, grid on
    set(h61,'FontSize',FontSize,'XLim',[0 N_traiectorii]) %,'YLim',[-2 20]
    set(gca,'XColor','b','YColor','b')

errorbar(traiectorii,mean(N_p_inite_tot),std(N_p_inite_tot),std(N_p_inite_t
ot),'w-')
    xlabel('traiectorii')
    ylabel('Initializari')
    h62=subplot(122); hold on, grid on
    set(gca,'FontSize',FontSize,'XLim',[0 N_traiectorii])
    set(gca,'XColor','b','YColor','b')
    errorbar(traiectorii,mean(N_p_mem),std(N_p_mem),std(N_p_mem),'w-')
    xlabel('traiectorii')
    ylabel('Memorari')
  end % if rulare==1

end % for rulare
```

```
toc % durata de executie [s]

break

%save fig43141 N_traiectorii N_p_inite N_p_adaug N_p_inite_tot N_p_mem
```

Bibliografie

Abu-Mostafa, Y. (1995). Hints. *Neural Computation*, 7(4), 639-671.

Albert, A. (1972). *Regression and the Moore-Penrose Pseudoinverse*. New York : Academic Press.

Albus. J.S. (1975). A new approach to the manipulator control: the Cerebellar Model Articulation Controller (CMAC). *Transactions of the ASME, Journal of Dynamic Sistems, Measurement and Control*, 37(3), 220-227.

Alpaydin, E. şi Jordan, M.I., (1996). Local linear perceptrons for classification. *IEEE Transactions on Neural Networks*, 7(3), 788-792.

Arimoto, S. şi Miyazaki, F. (1985). Sensory feedback for robot manipulators. *Journal of Robotic Systems*, 2(1), 53-71.

Atkeson, C.G (1989). Learning arm kinematics and dynamics, *Annual Review of Neuroscience*, 12, 157-183.

Atkeson. C.G (1991). Using locally weighted regression for robot learning. *Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, CA*, 958-963.

Bailleul. J. (1986). Avoiding obstacles and resolving kinematic redundancy. *Proceedings of IEEE International Conference on Robotics and Automation*, 1698-1704.

*** Barras Provence. Robot ERICC - Descriptif technique, guide d'exploitation et d'utilisation, Manosque, France.

Bassi, D. F. şi Bekey, G. A. (1989). High precision position control by cartesian trajectory feedback and connectionist inverse dynamics feedforward. *Proceedings of the 1989 International Joint Conference on Neural Networks*, Wasington DC, II, 341-347.

Beale, M. (1992) *Neural Network Toolbox for MATLAB*. The MathWorks.

Bekey. G. A. şi Goldberg, K. J., editors (1993). *Neural Networks in Robotics*. Boston: Kluwer Academics.

Bellanger, M. G. (1987). *Adaptive Digital Filters and Signal Analysis*. New York: Marcel Dekker.

Bien, Z., Jang W. şi Park J. (1994). Characterization and use of feature-Jacobian matrix for visual servoing. În Hashimoto (1994), 317-363.

Blum. E. K. şi Li, L.K. (1991). Approximation theory and fedforward networks. *Neural Networks*, 4(4), 511-515.

Boullion, T.L. şi Odell, P.L. (1971). *Generalized Inverse Matrices*. New York: Wiley-Interscience.

Breton, S. (1994). Contribution a l'approche connexioniste de la commande d'un système robot-vision, Rapport de stage RR 100, Laboratoire TROP, Université de Haute-Alsace, Mulhouse.

Brown, M., Harris, C.J. şi Parks. P.C. (1993). The interpolation capabilities of the binary CMAC. *Neural Networks*, 6(3), 429–440.

Cardaliaguet, P. şi Evrard, G. (1992). Approximation of a function and its derivative with a neural network, *Neural Networks*, 5(2), 207-220.

Carpenter, G. A. şi Grossberg, S. (1987 a). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics and Image Processing*, **37**, 54-115.

Carpenter, G. A. şi Grossberg, S. (1987 b). ART2: self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, **26**, 4919-4930.

Castaño, A. şi Hutchkinson, S. (1994). Visual compliance: task-directed visual servo control. *IEEE Transactions on Robotics and Automation*, **10**(3), 334-342.

Cherkassky, V. şi Lari-Najafi, H. (1991). Constrained topological mapping for nonparametric regression analysis. *Neural Networks*, **4**(1), 27-40.

Cichocki, A. şi Unbehauen, R. (1992). *Neural Networks for Optimization and Signal Processing*. New York: John Wiley and Sons.

Cimponeriu, A. (1994 a). *Modele de reţele neuronale artificiale nerecursive statice (feedforward) pentru recunoaşterea semnalelor*, referat I (bibliografic) pentru doctorat, Timişoara, Universitatea Tehnică, Facultatea de Electronică şi Telecomunicaţii, ianuarie 1994.

Cimponeriu, A. (1994 b). *Comandă robotică utilizând reţele neuronale*, referatul II pentru doctorat, Timişoara, Universitatea Tehnică, Facultatea de Electronică şi Telecomunicaţii, noiembrie 1994.

Cimponeriu, A. şi Gresser, J. (1995). Precision Requirements for Closed-Loop Kinematic Robotic Control Using Neural Networks. *International Conference on Neural Networks and Their Applications NEURAP'95*, Marseille.

Cimponeriu, A. şi Policec, A. (1995). The Z transform for vector systems. *Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara*, tom 40/54 Electrotehnică. Electronică şi Telecomunicaţii, 81-88.

Cimponeriu, A. (1996). An extension of the RLS algorithm. *Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara*, tom 41/55, Electrotehnică. Electronică şi Telecomunicaţii. (acceptată)

Cimponeriu, A. şi Gresser, J. (1996). Precision Requirements for Closed-Loop Kinematic Robotic Control Using Linear Local Mappings. *Neural Networks*. (propusă spre publicare).

Cimponeriu, A. şi Gresser, J. (1997). Adaptive Learning with the Growing Competitive Linear Local Mapping Network for Robotic Hand-Eye Coordination. 1997 International Conference on Neural Networks, Houston, Texas (propusă).

Cimponeriu, A. şi Kihl, H. (1997). Fast, Efficient Learning for Linear Local Mapping Networks for Robotic Hand-Eye Coordination. 1997 International Conference on Neural Networks, Houston, Texas (propusă).

Cioffi, J.M. şi Kailath, T. (1984). Fast, recursive-least-squares transversal filters for adaptive filtering. *IEEE Transactions on Acoustics, Speech and Signal Processing*, **32** (2), 304-337.

Corke, P. (1994). Visual Control of Robot Manipulators - A Review. În K. Hashimoto (1994), 1-31.

Cotter, N. E. şi Guillerm, T. J. (1992). The CMAC and a theorem of Kolmogorov. *Neural Networks*, **5**(2), 221-228.

Craig, J. J. (1986). *Introduction to Robotics*. Reading, MA : Addison-Wesley.

(DEX) *** (1984). Dicţionarul explicativ al limbii române. Bucureşti, Editura Academiei.

Drăgulescu, D., Toth-Taşcău, M. şi Moldovan, F. (1994). Planificarea mişcării roboţilor industriali. Timişoara: Helicon.

Duda, R. O. şi Hart, P.E. (1973). *Pattern Classification and Scene Analysis*. New York: Wiley.

Dudgeon R.E. şi Mersereau R.M. (1984). *Multidimensional Digital Signal Processing*. Englewood Cliffs, NJ: Prentice Hall.

Duflo, M. (1990). *Méthodes récursives aléatoires*. Paris: Masson.

Dumitrescu. D. şi Costin, H. (1996). *Reţele neuronale, teorie şi aplicaţii*. Bucureşti: Teora.

Fahlman, S. E. şi Lebiere, C. (1990). The cascade-correlation learning architecture. În D.S.Touretzky (ed.), *Advances in Neural Information Processing Systems 2*, 524-532. Los Altos, CA: Morgan Kaufmann.

Feddema J.T., Lee C. S. G. şi Mitchell O.R. (1994). Feature-Based Visual Servoing of Robotic Systems. În Hashimoto (1994.), pp. 105-138.

Fritzke, B. (1994 a). Supervised learning with growing cell structures. În J.D. Cowan, G. Tesauro şi J. Alspector, *Advances in Neural Information Processing Systems 6* (Proceedings of the 1993 NIPS Conference), 255-262, San Francisco: Morgan Kaufmann.

Fritzke, B. (1994 b). Growing cell structures - a self-organizing network for unsupervised and supervised learning, *Neural Networks*, 7(9), 1441-1460.

Fritzke, B. (1995). Incremental Learning of Linear Local Mappings. *Proceedings of the International Conference on Artificial Neural Networks*, Paris, 1995.

Fukushima, K. (1988). Neocognitron: a hierarchical neural network capable of visual pattern recognition. *Neural Networks*, **1**, 119-130.

Funahashi, K.-I. (1989). On the approximate realization of continuous mappings by neural networks, *Neural Networks*, **2**, 183-192.

Gallant, A. R. şi White, H. (1992). "On learning the derivatives of an unknown mapping with multilayer feedforward networks", *Neural Networks*, **5**(1), 129-138.

Geman, S., Bienenstock, E. şi Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, **4**, 1-58.

Gersho, A. şi Gray, R.M. (1992). *Vector Quantization and Signal Compression*. Boston: Kluwer Academic Publishers

Gill, P. E., Murray, W. şi Wright, M. H. (1981). *Practical Optimization*. London: Academic Press.

Girosi, F., Jones, M. şi Poggio, T. (1995). Regularization theory and neural networks architectures. *Neural Computation*, 7(2), 219-269.

Girosi, F. şi Poggio, T. (1990). Networks and the best approximation property. *Biological Cybernetics*, **63**, 169-176.

Golub, G. H. şi Van Loan, C. F. (1989). *Matrix Computations*. 2nd edition. Baltimore, MD : The Johns Hopkins University Press.

Goodwin, G.C. şi Sin, K.S. (1984). *Adaptive Filtering, Prediction and Control*. Englewood Cliffs, NJ: Prentice-Hall.

Gorinevsky, T. şi Konnoly, H. (1994). Comparison of some neural network and scattered data approximation: The inverse manipulator kinematics example. *Neural Computation*, **6**(3), 521-542.

Graf, H.P. (1996). Recent developments in Neural Net hardware. Prezentare la Montreal Workshop and Spring School on artificial Neural Networks, April 15-30 1996. Centre de Recherche Mathematique, Universite de Montreal. http://www.iro.umontreal.ca /labs/neuro/ spring96/english.html

Gresser, J., Urban, J. P., Buessler, J. L. şi Kihl, H. (1996). A neural approach to visual servoing in robotics. *Proceedings of the Symposium on Elelctronics and Telecommunications*, Timişoara, "Politehnica" University, September 1996, 2-7.

Hager, G. D., Chang, W.-C. şi Morse, A. S. (1994). Robot hand-eye coordination based on stereo vision. *1994 IEEE International Conference on Robotics and Automation*.

Hartman, E.J. Keeler, J.D. şi Kowalski, M. (1990). Layered neural networks with gaussian hidden units as universal approximators. *Neural Computation*, **2**, 210-215.

Hashimoto, K., editor (1994). *Visual servoing - Real-time control of robot manipulators based on visual sensory feedback*. London: World Scientific.

Hashimoto, K. şi Kimura, H. (1994). LQ optimal and nonlinear approaches to visual servoing. în Hashimoto (1994).

Haykin, S. (1991). Adaptive filter theory, 2nd edition. Englewood Cliffs, NJ : Prentice-Hall.

Haykin, S. (1994). *Neural Networks, A Comprehensive Foundation*. New York: Macmillan Publishing.

Hecht-Nielsen, R. (1990). *Neurocomputing*. Reading, MA:Addison-Wesley, 1990.

Heiss, M. (1994). Inverse passive learning of an input-output-map through update-spline-smoothing. IEEE Transactions on Automatic Control, **39**(2), 259-268.

Hertz, J. A., Krogh, J. A. şi Palmer, R. G. (1991). *Introduction to the theory of neural computation*. Redwood City, CA: Addison-Wesley.

Hesselroth, T., Sarkar, K., van der Smagt, P. şi Schulten, K. (1994). Neural network control of a pneumatic robot arm. IEEE Transactions on Systems, Man, and Cybernetics, **24**(1), 28-38. Rezumatul în http://www.ks.uiuc.edu /Papers/abstracts/HESS94.html

Hollinghurst, N. şi Cipolla, R. (1993). Uncalibrated stereo hand-eye coordination. Technical Report TR 126, Cambridge University, Dept. of Engineering, September 1993.

Hornik, K., Stinchcombe, M. şi White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, **2**(5), 359-366.

Hornik, K., Stinchcombe, M. şi White, H. (1990). Universal approximation of an unknown mapping and its derivatives using feedforward networks. *Neural Networks*, **3**, 661-560.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks* , **4**(2), 251-257.

Hunt, K.J., Haas, R. şi Murray-Smith, M. (1996). Extending the functional equivalence of radial basis functions networks and fuzzy inference systems. *IEEE Transactions on Neural Networks*, **7**(3), 776-781.

Hush, D. R. şi Horne, B.G. (1993). Progress in supervised neural networks. *IEEE Signal Processing Magazine*, **10**(1), 8-39.

Jacobs, R. A. şi Jordan, M. I. (1993). Learning piecewise control strategies in a modular neural network architecture. *IEEE Transactions on Systems, Man and Cybernetics*, **23**(2), 337-345.

Jägersand, M. (1996). Visual servoing using trust region methods and estimation of the full coupled visual-motor Jacobian. Proceedings of the IASTED Applications of Robotics and Control. ftp:// ftp.cs.rochester.edu /pub/u/jag/ iastedARC96.ps.Z

Jägersand, M. şi Nelson, R. (1994). Adaptive differential visual feedback for uncalibrated hand-eye coordination and motor control. Technical Report TR # 579, University of Rochester.

Jägersand, M. şi Nelson, R. (1995). Visual space task specification, planning and control. În Proceedings of the 1995 IEEE International Symposium on Computer Vision, 521-526. http://www.cs.rochester.edu/u/{jag, nelson}/ .

Jägersand, M., Fuentes, O. şi Nelson, R. (1997). Experimental evaluation of uncalibrated visual servoing for precision manipulation. *Proceedings of the 1996 International Conference on Robotics and Automation* (propusă).

Jordan, M.I. (1994). Computational aspects of motor control and motor learning. MIT Computational Cognitive Science Report 9206, ftp://psyche.mit.edu/pub/jordan/control-intro.ps.Z. şi în H. Heuer şi S. Keele (editori), *Handbook of perception and action: Motor skills*. New York: Academic Press. ftp:// psyche.mit.edu /pub/jordan/control-intro.ps.Z

Jordan, M.I. şi Jacobs, R.A. (1994). Hierarchical mixture of experts and the EM algorithm. *Neural Computation*, **6**(2), 181-214.

Khadraoui, D., Motyl, G., Martinet, P., Gallice, J. şi chaumette, F. (1995). Visual servoing in robotics scheme using a camera/laser-stripe sensor. IRISA, Institut de Recherche en Systèmes Aléatoires, Publication interne n° 898, janvier 1995. Propus spre publicare la *IEEE Transactions on Robotics and Automation*.

Kohonen, T. (1989). *Self-organization and associative memory*, ed. a 3-a. New York: Springer Verlag.

Kohonen, T., Barna, G. şi Chrisley R. (1988). Statistical pattern recognition with neural networks : benchmarking studies. În IEEE International Conference on Neural Networks (San Diego 1988), vol. I, 61-68. New York: IEEE.

Kohonen, T., Kangas, J., Laaksonen, J. şi Torkkola, K. (1992). LVQ PAK. The Learning Vector Quantization Package, version 2.1, Laboratory of Computer and Information Science, The Helsinki University of Technology, ftp://cochlea.hut.fi (190.299.168.48) /pub/lvq-pak.

Kreutz-Delgado, K., Agahi, D. (1995). A Recursive Sigularity-Robust Jacobian Generalized Inverse. *IEEE Transactions on Robotics and Automation*, **11**(6), 887-892.

Kuhn, D., Buessler, J.L. şi Urban, J.-P. (1995). Neural approach to visual servoing for robotic hand eye coordination. *Proceedings of the 1995 International Conference on Neural Networks*. Perth, Australia, **5**, 2364-2369.

* * * - MICAM X, Notice d'utlilisation, Digital Vision Technologies, Toulouse.

Lane, S.H., Handelman, D.A. şi Gelfand, J.J. (1992). Theory and development of higher-order CMAC neural networks. *IEEE Control Systems*, **12**(2), 23-30.

Lapedes, A., şi Farber, R. (1987). *Nonlinear Signal Processing Using Neural Networks: prediction and System Modeling*. Technical Report TR LA-UP-87-2662, Los Alamos National Laboratory, Los Alamos, NM.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. şi Jackel, W.D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, **1**, 541-551.

Lee, S. şi Kil, R. M. (1991). A gaussian potential function network with hierarchically self-organization learning. *Neural Networks*, **4**(2), 207-224.

Lee, S. şi Kil, R. M. (1994). Redundant Arm Kinematic Control With Recurrent Loop. *Neural Networks*, 7(4), 643-659.

Leighton, R. (1992). *The Aspirin/MIGRAINES Neural Network Software User's Manual, Release V6.0*, The MITRE Corporation, USA.

Leonard, J. A., Kramer, M. A. şi Ungar, L.H. (1992). Using radial functions to approximate a function and its error bounds, *IEEE Transactions on Neural Networks*, **3**(4), 624-627.

Lin, Y. şi Song, S.-M. (1993). A CMAC neural network for the kinematic control of walking machine. În Bekey şi Goldberg (1993).

Lippmann, R. P. (1989). Pattern classification using neural networks. *IEEE-Communications Magazine*, **27**(11), 47-64.

Littmann, E. şi Ritter, H. (1992). Cascade LLM networks. În I. Aleksander şi J. Taylor (editori). *Artificial Neural Networks*. **2**, 253-257, Elsevier Science.

Littmann, E. şi Ritter, H. (1996). Learning and generalization in cascade network architectures. *Neural Computation*, **8**(7), 1521-1539.

Lu, C.-P., Mjolsness, E. şi Hager, G. D. (1994). Online computation of exterior orientation with application to hand-eye calibration. Research Report YALEU/DCS/RR-1046. August 1994.

Maru, N., Kase, H., Nishikawa, A. şi Miyazaki, F. (1993). Manipulator control by visual servoing with the stereo vision. În *Proceedings of The IEEE International Conference on Inteligent Robots and Systems*, 1866-1870, IEEE Computer Society Press.

Martinetz, T. M., Berkovitz, S. G. şi Schulten, K. J. (1993). Neural-gas network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, 4(4), 558-569.

Mayorga, R. V., Milano, N., şi Wong, A. K. C. - A Fast Procedure for Manipulator Inverse Kinematics Computation and Singularities Prevention. *Journal of Robotic Systems*, **10**(1), 45-72, 1993.

*** Micam X. Digital Vision Technologies. MICAM X. Notice d'utilisation, Toulouse.

Michaut, F. (1992). *Méthodes adaptatives pour le signal*. Paris: Hermès.

Micula, Gh. (1978). *Funcţii spline şi aplicaţii*. Bucureşti: Editura Tehnică.

Miller, W.T. (1989). Real-time application of neural networks for sensor-based control od robots with vision. *IEEE Transactions on Systems, Man, and Cybernetics*. **19**(4), 825-831.

Miller, W. T., Sutton, R. S. şi Werbos, P. J., editors (1990). *Neural Networks for Control*. Cambridge, MA: MIT Press.

Molina, C. şi Niranjan, M. (1996). Pruning with replacement on limited resource allocating networks by F-projections. *Neural Computation* **8**(4), 855-868.

Moody, J. (1989). Fast learning in multi-resolution hierarchies. În D.S. Touretzky (ed.). *Advances in Neural Information Processing Systems 1*, 29-39, San Mateo, CA: Morgan Kaufmann.

Moody, J. şi Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, **1**, 281-294.

Moore, K. L. (1992). *Iterative Learning Control for Deterministic Systems*. Springer Verlag London Ltd.

Najafi, H. L. şi Cherkassky, V. (1994) Adaptive knot placement for nonparametric regression. În J. D. Cowan, G. Tesauro şi J. Alspector. *Advances in Neural Information Processing Systems 6*, 247-254, San Francisco: Morgan Kaufmann.

Nakamura, Y. (1991). *Advanced Robotics : Redundancy and Optimization*. Reading, MA : Addison-Wesley.

Niyogi, P. şi Girosi, F. (1995). On the relation between generalization error, hypothesis complexity, and sample complexity, for radial basis functions. *Neural Computation*. **8**, 819-842.

Pellionisz, A. şi Llinas, R. (1979). Brain modeling by tensor network theory and computer simulation. The cerebellum: Distributed processor for predictive coordination. *Neuroscience*, **4**, 323-348.

Pellionisz, A. şi Llinas, R. (1980). Tensorial approach to the geometry of brain function: Cerebellar coordination via a metric tensor. *Neuroscience*, 5, 1125-1136.

Penrose, R. (1955). A generalized inverse for matrices. *Proc. Cambridge Philos. Soc.* **51**, 406-413.

Platt, J. (1991). A resource-allocating network for function interpolation. *Neural Computation*, **3**, 213-225.

Ploix, J.-L., Dreyfus, G., Corriou, J.-P., şi Pascal, D. (1994). From knowledge-based models to recurrent networks: an application to an industrial process. În J. Hérault (ed.), *Neural Networks and Their Applications*.

Poggio, T. şi Girosi, F. (1990 a). Networks for approximation and learning. *Proceedings of the IEEE* **78**(9).

Poggio, T. şi Girosi, F. (1990 b). Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, **247**.

Policarpou, M. M. şi Ioannou, P. A. - A neural-type parallel algorithm for fast matrix inversion. *Proc. of The Fifth International Parallel Processing Symposium*. IEEE, pp.108-113, 1991.

Pop E., Naforniţă I., Tiponuţ V., Mihăescu A. şi Toma L. (1989). *Metode în prelucrarea numerică a semnalelor*, vol. I, II. Facla, Timişoara.

Press, W. H., Flannery, B. P., Teukolsky, S. A. şi Vetterling, W. T. (1988). *Numerical Recipes in C*. Cambridge : Cambridge University Press.

Rao, C. R. (1965). *Linear Statistical Inference and Its Applications*. New York: Wiley.

Reed, R. (1993). Pruning algorithms - a survey. *IEEE Transactions on Neural Networks*, **4**(5), 740-747.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge UK: Cambridge University Press.

Ritter H. J., Martinetz, T. M. şi Schulten, K. J. (1989). Topology-conserving maps for learning visuo-motor coordination. *Neural Networks*, **2**, 159-168.

Ritter H. J., Martinetz, T. M. şi Schulten, K. J. (1992). *Neural Computation and Self-Organizing Maps*. Reading, MA: Addison-Wesley.

Rivoire M. şi Ferrier J.-L. (1992). *Cours d'automatique, tome 1 - Signaux et systèmes*. Paris: Eyrolles.

Roger Jang, J.-S. şi Sun, C.-T., (1993). Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Transactions on Neural Networks*, **4**(1), 156-158.

Rumelhart, D. E. şi McClelland, J. L., editori (1986). *Parallel Distributed Processing I, II*. Cambridge MA: MIT Press.

Samson, C., Le Borgne, M. şi Espiau, B. (1991). *Robot control: The task function approach*. Oxford: Claredon Press.

Schaal, S. şi Atkeson, C.G. (1995). From isolation to cooperation: An alternative view of a system of experts. Propusă pentru conferinţa Neural Information Processing Systems 1995.

Sontag, E. D. (1990). Feedback Stabilization Using Two-Hidden-Layer Nets. Report SYCON-90-11, Rutgers Center for Systems and Control.

Spătaru, A. (1990). *Elemente de analiză pentru probabilişti*. Bucureşti: Ed. Academiei.

Specht, D.F. (1990). Probabilistic neural networks. *Neural Networks*, **3**(1), 109-118.

Suga, N. (1990). Cortical computational maps for auditory imaging. *Neural Networks*, **3**(1), 3-22.

Tarabanis, K. A., Allen, P. K. şi Tsai, R. Y. (1995). A survey of sensor planning in computer vision. *IEEE Transactions on Robotics and Automation*, 11(1), 86-104.

Telfer, B. A. şi Casasent, D. P. (1994). Fast method for updating robust pseudoinverse and Ho-Kashyap associative processors. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(9), 1387-1390.

Toderean, G., Coşteiu, M. şi Giurgiu, M. (1994). *Reţele Neuronale*. Ed. Microinformatica, Cluj-Napoca.

van der Smagt, P. (1994). Minimisation methods for training feed-forward networks. *Neural Networks*, 7(1), 1-11.

van der Smagt, P. P. (1995). Visual robot arm guidance using neural networks. PhD Thesis (summary), University of Amsterdam, January 1995.

van der Smagt, P. P., Groen, F. C. A. şi Kröse, B. J. A. (1995). A monocular robot arm can be neurally positioned. 1995 International Conference on Intelligent Autonomous Systems (IAS-4), în curs de tipărire. http://www.ks.uiuc.edu /Papers/Neural_Network_Control_Pneumatic_Robot_Arm/ Neural_Network_Control_Pneumatic_Robot_Arm.html

Van Huffel, S. şi Vanderwalle, J. (1991). *The Total Least Squares Problem, Computational Aspects and Analysis. Frontiers in Applied Mathematics*, 9. Philadelphia : SIAM.

Vapnik, V. (1992). Principles of risk minimization for learning theory. În Moody, J. E., Hanson, S. J. şi Lippmann, R. P. (editori), *Advances in Neural Information Processing Systems 4*, San Mateo, CA: Morgan Kaufmann. 831-838.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. New York: Springer.

Vapnik, V. (1996). Theory of consistency of learning process. Prezentare la Workshop on Artificial Neural Networks, Université de Montreal. 15-30 aprilie 1996.

Waibel, A., Hanazawa, T., Hinton, G. şi Lang, K. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics Speech and Signal Processing*, 37(3). 328-339.

Walter, J. A. şi Schulten, K. J. (1993). Implementation of Self-Organizing Neural Networks for Visuo-Motor Control of an Industrial Robot. *IEEE Transactions on Neural Networks*, 4(1). 86-95.

Weiss, S. M. şi Kulikowski, C. A. (1991). *Computer Systems that Learn*. San Mateo, CA: Morgan Kaufmann.

Weiss, L.E., Sanderson, A.C. şi Newman, C.P. (1987). Dynamic sensor-based control of robots with visual feedback. *IEEE Journal of Robotics and Automation*, 3(5), 404-417.

White, H. (1989). Learning in artificial neural networks: a statistical perspective. *Neural Computation*, 1, 425-464.

Weymaere, N. şi Martens, J.P. (1991). A fast and robust learning algorithm for feedforward neural networks. *Neural Networks*, 4(3), 361-369.

Wettschereck, D., şi Dietterich, T. 1994. Locally adaptive nearest neighbor algorithms. În J.D. Cowan, G. Tesauro şi J. Alspector, *Advances in Neural Information Processing Systems 6* (Proceedings of the 1993 NIPS Conference), 184-191, San Francisco: Morgan Kaufmann.

Widrow, B. şi Hoff, M. E. (1960). Adaptive Switching Circuits. În *1960 IRE WESCON Convention Record*, part 4, 96-104. New York: IRE.

Widrow, B. şi Stearns, S. D. (1985). *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall.

Williams, T. şi Kelley, C. (1993). Gnuplot 3.5. ftp:// ftp.dartmouth.edu/ pub/gnuplot' gnuplot3.5.tar.Z

Wong, Y. (1991). How radial basis functions work. Proceedings of the IJCNN (International Joint Conference on Neural Networks) San Diego 1991, **II**, 133-138.

Wong, Y. şi Sideris, A. (1992). Learning convergence in the Cerebellar Model Articulation Controller. *IEEE Transactions on Neural Networks*, **3**(1), 115-120.

Yeung, D.-Y. şi Bekey, G. A. (1993). On reducing learning time in context-dependent mappings. *IEEE Transactions on Neural Networks*, **4**(1), 31-42.

Yoshimi, B. H. (1995). Visual control of robotic tasks, PhD Dissertation, Columbia University (yoshimi@cs.columbia.edu).

Zador, P. l. (1982). Asymptotic quantization error of continuous signals and the quantization dimension. *IEEE Transactions on Information Theory*, **28**(2), 139-149.