UNIVERSITATEA "POLITEHNICA" TIMISOARA
Facultatea de Automatizari si Calculatoare

Autor:
Ing. Daniel-Ioan Curiac

SOLUTII DE MODELARE A VITEZEI VÂNTULUI SI DE CONDUCERE ADAPTIVA A AGREGATELOR AEROELECTRICE

- TEZA DE DOCTORAT -

Conducator stiintific:
Prof.Dr.Ing. Nicolae Budisan

Timisoara
1995

CUPRINS

CAPITOLUL 1: INTRODUCERE.....pag.5

CAPITOLUL 2: SOLUTIE DE MODELARE SI PREDICTIE A VITEZEI VÂNTULUI UTILIZÂND TEORIA SISTEMELOR HAOTICE.....pag.10

2.1 Seria de timp - secventa de observatii ordonate in timp..pag.10
2.2 Abordarea Box-Jenkins - metodologie clasica de modelare si predictie a seriilor de timp.....pag.11
2.2.1 Modele utilizate in predictia seriilor de timp.....pag.11
2.2.2 Metodologia Box-Jenkins.....pag.13
2.2.3 Studiu de caz - modelarea unei serii reprezentând viteza vântului.....pag.16
2.2.3.1 Program Matlab pentru calculul parametriilor seriilor de timp utilizând metodologia Box-Jenkins.....pag.19
2.2.4 Dezavantajul utilizarii zgomotului alb in cadrul abordarii Box-Jenkins.....pag.20
2.3 Sisteme dinamice haotice.....pag.20
2.3.1 Prezentare generala. Definitii.....pag.21
2.3.2 Sisteme haotice remarcabile.....pag.22
2.3.2.1 Atractorul Lorenz.....pag.22
2.3.2.2 Atractorul Rössler.....pag.26
2.3.2.3 Atractorul Henon.....pag.30
2.4 Solutie de identificare on-line a modelului vântului.....pag.33
2.4.1 Estimarea parametrilor modelului seriei de timp.....pag.34
2.4.1.1 Metoda celor mai mici patrate - varianta on-line.....pag.34
2.4.1.2 Algoritmi recursivi de estimare obtinuti prin utilizarea factorizarilor matriciale.....pag.37
2.5 Pachet de programe destinat estimarii multivariabile on-line a seriilor de timp - simulatorul SERTIM 1.0.....pag.38
2.5.1 Structura si principalele componente ale pachetului SERTIM 1.0.....pag.40
2.5.2 Facilitati in utilizarea pachetului de programe SERTIM 1.0.....pag.41
2.5.3 Rezultate experimentale.....pag.45
ANEXA 2.1 Programe de simulare a sistemelor dinamice haotice.pag.59
ANEXA 2.2 Programele sursa in limbajul C pentru pachetul SERTIM 1.0.....pag.63

CAPITOLUL 3: UTILIZAREA RETELELOR NEURONALE IN ANALIZA SI PREDICTIA SERIILOR DE TIMP.....pag.97

3.1 Retelele neuronale.....pag.97
3.1.1 Modelul neuronului McCullogh-Pitts.....pag.99

3.1.2 Retele neuronale feedforward.....pag.102
3.1.2.1 Algoritmul de propagare inapoi (backpropagation)....pag.104
3.1.2.2 Metoda gradientului.....pag.107
3.1.2.3 Dificultati aparute in antrenarea retelelor neuronale folosind algoritmul backpropagation.....pag.113
3.1.2.4 Metoda de antrenare a retelelor neuronale.....pag.115
3.1.2.4.1 Structura adoptata pentru reteaua neuronala.....pag.115
3.1.2.4.2 Prezentarea metodei.....pag.116
3.2 Solutie pentru modelarea si predictia seriilor de timp utilizând retele neuronale recurente.....pag.119
3.2.1 Retele neuronale recurente.....pag.119
3.2.1.1 Studiu asupra alegerii erorii maxime in procesul de antrenare a RNR.....pag.121
3.2.2 Programe in limbajele C si Simnon. Studii de caz.....pag.122
3.2.2.1 Studiu de caz - antrenarea unei retele neuronale recurente pentru obtinerea unei serii de timp sinusoidale f(x)=sin(x) in primul cadran.....pag.122
3.2.2.2 Studiu de caz - antrenarea unei retele neuronale recurente pentru obtinerea unei serii de timp cosinusoidale f(x)=cos(x) in primele doua cadrane.....pag.132
3.2.2.3 Studiu de caz - antrenarea unei retele neuronale recurente pentru obtinerea unei serii de timp amortizate....pag.135
3.2.2.4 Studiu de caz - antrenarea unei retele neuronale recurente pentru obtinerea unei serii de timp periodice.....pag.137
3.2.3. Concluzii asupra modelarii seriilor de timp utilizând retele neuronale recurente.....pag.139
ANEXA 3.1 Programul sursa antrenam.c.....pag.140

CAPITOLUL 4: SOLUTIE DE IMPLEMENTARE A UNEI TEHNICI DE CONDUCERE ADAPTIVA CU ELEMENTE NEURONALE DESTINATA CONDUCERII AGREGATELOR AEROELECTRICE.....pag.146

4.1 Introducere in tematica sistemelor adaptive.....pag.146
4.2 Principalele tehnici de conducere adaptiva.....pag.147
4.2.1 Programarea câstigului.....pag.148
4.2.2 Sisteme adaptive cu model etalon (SAME).....pag.149
4.2.3 Regulatoare autoacordabile (RAA).....pag.150
4.3 Aspecte teoretice ale conducerii adaptive.....pag.151
4.3.1 Problema conducerii adaptive.....pag.152
4.3.2 Principii de proiectare.....pag.154
4.3.3 Analiza stabilitatii si convergentei structurilor adaptive.....pag.155
4.4 Utilizarea elementelor neuronale pentru implementarea unui regulator autoacordabil.....pag.156
4.4.1 Schema de conducere adoptata.....pag.156
4.4.2 Aspecte de implementare.....pag.159
4.4.3 Alegerea unor parametri pentru conducerea adaptiva....pag.164
4.4.4 Studiu de caz. Rezultate. Concluzii.....pag.166
ANEXA 4.1 Programul sursa in limbajul C ce implementeaza algoritmul propus.....pag.174

ANEXA 4.2: Demonstrarea convergentei algoritmului destinat conducerii adaptive.....pag.179

CAPITOLUL 5: CONCLUZII.....pag.185

BIBLIOGRAFIE.....pag.188

CAPITOLUL 1: INTRODUCERE

In secolul XX societatea umana este confruntata cu rezolvarea ingratei ecuatii a energiei, a carei solutie, prin complementaritate poate fi furnizata numai prin dezvoltarea unor surse noi, alternative, de energie.

In spatiul acestor coordonate, una din problemele actuale ale tarii este cea a asigurarii independentei din punct de vedere energetic. Odata cu trecerea timpului, va trebui sa se apeleze tot mai mult la importuri, fie ca este vorba de carbune, titei, uraniu sau alte materii prime. Romania dispune de resurse nationale inca nevalorificate carora li se va acorda cu certitudine importanta cuvenita intr-un viitor nu prea indepartat: soare, vânt, ape geotermale, etc.

Utilizarea acestora nu este o chestiune conjuncturala. Tari industrializate, bogate, dezvolta ample programe pentru promovarea lor. De exemplu, utilizarea energiei eoliene a capatat in tarile vestice (Statele Unite ale Americii, Danemarca, Germania, Olanda, Italia, Marea Britanie, etc.), un caracter industrial cu certe efecte economice. Cifra de afaceri in acest domeniu - in ultimii câtiva ani - a depasit valoarea de 10 miliarde dolari.

Astfel, actualitatea dezvoltarii energeticii eoliene in Romania este incadrata intr-un context mondial, prin cererea acuta de surse noi de energie si oferta vântului ca sursa curata, nepoluanta accesibila ca tehnologie si care se gaseste pe plan mondial intr-o faza de maturizare industriala.

O privire istorica a etapei 1930-1990, caracterizata prin redescoperirea sursei eoliene, ofera o privire de ansamblu asupra dinamicii solutiilor tehnice si a interesului economic permitând formularea unor orientari bine sedimentate pentru ultimul deceniu al secolului si primele decenii ale celui de al treilea mileniu.

Epoca 1930-1973 poate fi numita "a initiativelor locale" - unele indraznete si naive (Honnef 1932: proiect de centrala de 20MW), altele spectaculoase (1931 URSS - agregat de 100kW, 1941 SUA - 1250kW, 1950 Scotia - 100kW; 1957 Danemarca - 200kW; 1946 Franta - 132, 1000 si 800 kW). Confruntarea cu petrolul inclina balanta in defavoarea vântului. Socul crizelor petrolului din anii 1970 modifica conjunctura, vântul devenind sursa cu multe promisiuni si cu numeroase realizari tehnice de vârf: MOD 1 si 2, respectiv 2.5 MW in SUA, GROWIAN 1 de 3MW in Germania, KAMEWA de 2MW in Suedia, TWIND de 2MW si centrala experimentala NIBE in Danemarca, toate in anii 1980-1982.

Costurile si fiabilitatea ridica probleme care au dus la stagnarea tendintei de marire a puterii unitare. Au aparut solutii industriale competitive cu sursele clasice inclusiv cu cea nucleara in zona puterilor mijlocii de 100 - 300 kW, cu diametre ale turbinei de pina la 30 m, fara a fi parasita insa definitiv zona puterilor mari.

Daca in 1980 puterea instalata in lume era estimata la 30MW, in ultimii ani aceasta a crescut la peste 3000MW.

O centrala aeroelectrica amenajata este o zona in care din rezervorul de energie electrica al atmosferei terestre se extrage mai multa putere decât cea mijlocie regenerata de soare. Aceste concentrari de valorificare depind de regimul local al vântului si de calitatea tehnologiei de valorificare. Turbinele eoliene utilizate azi, in punctul lor optim de functionare, sunt capabile sa valorifice sub forma de energie electrica doar circa o treime din puterea cinetica a tubului de curent asociat turbinei.

Se considera ca limita tehnic-valorificabila de energie cinetica a vântului valoarea de 0.25W/m^2, ceea ce la nivel mondial duce la un potential de 130TW, mult mai mare decât cel hidroenergetic. Unele prognoze de valorificare pe plan mondial pentru anul 2000, prevad o putere mijlocie de circa 1TW. Unele tari industrializate conteaza in 2000 pe cote de participare a energiei vântului de peste 5% in productia de energie electrica. In privinta Romaniei rezulta un nivel teoretic de circa 500TWh/an. In aceste conditii un potential amenajabil in urmatorii 15-20 de ani de ordinul 5-7TWh/an este o prognoza realista; acestui nivel i-ar corespunde o putere instalata de circa 2000-3000MW si o suprafata amenajata pentru centrale aeroelectrice de circa 100-150 km^2.

Aceste conditii pot fi asigurate relativ usor in zonele montane si pe litoralul Marii Negre. Terenul acoperit de reteaua de agregate nu este extras de la alte valorificari, distanta dintre agregate fiind de 150-200m, in cazul agregatului de 300 kW, si de 300-350m, in cazul agregatului de 1000kW.

In continuare va fi prezentata situatia enrgeticii eoliene in câteva zone ale lumii:

- *Certitudini si perspective pe plan mondial in energetica eoliana*

○ *Europa occidentala*

Noul context economic si industrial al Europei unite subliniat prin obiectivele pietei comune vor solicita existenta unei baze energetice deosebit de puternica. Situatia energiei in cadrul Comunitatii Europene sufera inca de insecuritate, de dezechilibre regionale si de nerezolvarea completa a problemelor de protectie a mediului. Cheia rezolvarii acestor probleme este dezvoltarea si utilizarea pe scara larga a noilor tehnologii energetice. Acesta este si motivul pentru care in anul 1990 a fost lansat programul "THERMIE", un proiect destinat cu precadere dezvoltarii de tehnologii pentru obtinerea energiei din surse regenerabile. Prin acest proiect se aloca sume considerabile pentru dezvoltarea energeticii eoliene (peste 10.6 MECU anual), sume care le completeaza pe cele alocate prin programele nationale si regionale.

Un aspect important este modul de alocare, in cadrul programului "THERMIE", a fondurilor pentru agregatele aeroelectrice in functie de destinatia acestora:

CODUL "THERMIE"	DESTINATIA SUMELOR ALOCATE	NUMAR DE PROIECTE	COSTUL TOTAL (MECU)	FOND ALOCAT PRIN "THERMIE" (MECU)
10	AGREGATE EOLIENE AUTONOME	19	6.56	2.1
20	AGREGATE EOLIENE CONECTATE LA RETEA	97	185.2	53.3
30	GRUPURI DIESEL-EOLIAN	23	8.4	3.4
40	GRUPURI SOLAR-EOLIAN	3	0.6	0.2
50	GRUPURI HIDRO - EOLIAN	3	2.3	0.9
TOTAL		145	203.2	56.9

In tarile Comunitatii Economice Europene, daca se dispunea in 1989 de o putere instalata in agregate aeroelectrice de 215MW, iar in 1992 de 533MW, prognoza pentru 2000 este de 4250MW, pentru 2005 de 11500MW, iar pentru 2030, de 100000MW care vor asigura o pondere de 10% in balanta energiei electrice.

○ *Statele Unite ale Americii*

Cele 16000 agregate in functiune in California in 1989, realizate in doar 8 ani, cu colaborarea a 30 firme din diferite tari, realizeaza 1% din energia electrica a Californiei, echivalenta cu productia unei centrale nucleare. Pâna in anul 2000 cercetatorii americani prevad cresterea ponderii energiei eoliene pâna la 8% din intreaga productie de energie electrica a Californiei, adica aproximativ 400MW.

In 1994 erau oferite spre comercializare investitorilor publici si privati peste 100 de tipuri de agregate, estimându-se, pentru anii imediat urmatori, o explozie in utilizarea acestora pe intreg teritoriul Statelor Unite.

○ *Tari in curs de dezvoltare*

Tarile in curs de dezvoltare cauta sa urmareasca si ele trend-ul mondial, cautând pentru aceasta sprijin financiar. Un exemplu concludent il constituie Egiptul, care cu sprijinul PNUD si ONUDI, isi propune, pe o fâsie de 250km de-a lungul tarmului Marii Rosii, realizarea unei centrale aeroelectrice de 5000MW pâna in anul 2005, din care primele 100 turbine au si fost puse in functiune. Ponderea acestei surse in balanta energiei electrice a acestei tari va ajunge la 16%.

○ *Romania*

Studiile facute la noi demonstreaza ca potentialul eolian al Romaniei este de acelasi ordin de marime cu cel hidroenergetic, existând conditii tehnico-economice, ca pe un termen mediu sa se instaleze câteva mii de megawati eolieni cu o productie anuala de câteva miliarde kWh/an. Alinierea noastra la nivel mondial inseamna realizarea in urmatorii 20 ani a unei puteri instalate in centrale aeroelectrice de 2000-3000MW.

Preocuparile de pina acum ale exceptionalilor tehnicieni romani, ne ofera sansa reala de a putea trece in curând la utilizarea pe scara industriala a acestei surse nationale si nepoluante in conditii de competitivitate, creind in acelasi timp noi locuri de munca.

Formatiunile de cercetare stiintifica si proiectare din tara au acumulat o experienta semnificativa in domeniu, fapt ce a permis realizarea a 14 tipuri de agregate mici, un agregat Darrieus de 100kW si unul cu elice de 300kW (Semenic). Este in studiu si unul de 1000kW. Au fost intocmite studii pentru mai multe amplasamente de centrale aeroelectrice, totalizând o putere instalata de circa

2000MW.

Pe certitudinile din California si Comunitatea Economica Europeana, cu experienta valoroasa acumulata in tara, pot fi dezvoltate programe indraznete, de concretizare in etapa imediat urmatoare a rezultatelor obtinute, demonstrate prin agregatele aeroelectrice de putere medie puse in functiune.

Topics-urile agreate atât de catre programele de dezvoltare americane cât si de catre cele europene, in domeniul cercetarii stiintifice asupra agregatelor aeroelectrice sunt [The93] in ordine:

- Masurarea, modelarea si predictia vântului;
- Experimentarea si utilizarea turbinelor eoliene;
- Dezvoltarea unor strategii de conducere adaptiva si optimala;
- Utilizarea agregatelor aeroelectrice in regim autonom (necuplate la retea);
- Turbine eoliene de puteri mari.

Prezenta teza de doctorat se inscrie pe aceasta linie dezvoltând si concretizând prin solutii originale atât problematica modelarii si predictiei vântului (capitolele 2 si 3) cât si problematica strategiilor de conducere adaptiva (capitolul 4).

Autorul tezei tine sa multumeasca in mod deosebit conducatorului stiintific, domnului prof.dr.ing. Nicolae Budisan, pentru indrumarea competenta, extrem de eficienta si plina de intelegere acordata de-a lungul perioadei de pregatire a doctoratului.

Autorul multumeste de asemenea cadrelor didactice din Catedra de Automatica si Informatica Tehnica pentru discutiile purtate, pentru sugestiile si recomandarile, precum si incurajarile primite.

CAPITOLUL 2: SOLUTIE DE MODELARE SI PREDICTIE A VITEZEI VÂNTULUI UTILIZÂND TEORIA SISTEMELOR HAOTICE.

Vântul, ca fenomen aleator, poate fi asimilat unei serii de timp, si modelat ca atare. In acest capitol este prezentata abordarea Box-Jenkins (ABJ) destinata modelarii si predictiei seriilor de timp (paragraful 2.2), abordare ce sta la baza dezvoltarii unei metodologii de identificare a unor modele viabile pentru vânt utilizând teoria sistemelor haotice (paragrafele 2.3, 2.4).

Metodologia propusa in acest capitol inlatura cel mai important dezavantaj al utilizarii ABJ - folosirea in vederea modelarii a zgomotului alb. In finalul capitolului este descrisa structura pachetului de programe aferent, rezultatele obtinute, precum si programele sursa (Anexa 2.2).

2.1 SERIA DE TIMP - SECVENTA DE OBSERVATII ORDONATE IN TIMP

Modelarea, analiza si predictia seriilor de timp dezvaluie un domeniu relativ nou, cu o dezvoltare exploziva in ultimii ani si cu aplicatii in aproape toate sferele de activitate: industrie, agricultura, transporturi, economie, geologie-geofizica, meteorologie, hidrologie, biologie, medicina, demografie, etc.

Seria de timp poate fi definita ca reprezentând o secventa ordonata in timp de observatii (masuratori), efectuate asupra unor variabile.

Pentru anumite serii de timp, masurarea valorilor acestora are

loc continuu, la fiecare moment de timp, seriile de timp numindu-se, in acest caz, continue (de exemplu, viteza vântului poate fi masurata continuu).

Cu toate acestea marea majoritate a seriilor de timp din practica sunt constituite din observatii efectuate asupra unor variabile la momente de timp predeterminate, separate prin intervale de timp egale, care pot fi, de exemplu secunde, minute, ore, zile, luni, ani. Astfel de secvente sunt referite drept serii de timp discrete si pot fi obtinute in diferite moduri. Chiar daca seria de timp originala, care se studiaza este continua, aceasta poate fi transformata intr-una discreta, printr-o operatie de discretizare (esantionare). O serie de timp discreta, obtinuta in acest mod, este denumita serie de timp esantionata.

Un alt tip de serie de timp care apare frecvent in practica este acela in care variabila de interes nu poate fi masurata continuu la fiecare moment de timp, fiind posibila numai masurarea valorilor cumulate (integrate) ale acesteia, dupa anumite intervale de timp egale. Astfel de serii de timp sunt referite ca serii de timp cumulate sau serii de timp integrate (de exemplu, cantitatea de precipitatii masurata zilnic intr-o statie meteo).

Pe parcursul acestui capitol se vor face referiri doar la seriile de tip discret, cu observatii masurate la intervale de timp egale.

2.2 ABORDAREA BOX-JENKINS - METODOLOGIE CLASICA DE MODELARE SI PREDICTIE A SERIILOR DE TIMP

Prezentul paragraf isi propune o sinteza a aspectelor metodologice si practice specifice analizei si predictiei seriilor de timp prin abordarea Box-Jenkins (ABJ), una din cele mai frecvent utilizate si precise tehnici de predictie pe termen scurt.

2.2.1 MODELE UTILIZATE IN PREDICTIA SERIILOR DE TIMP

In scopul modelarii seriei de timp z(t), aceasta este considerata in abordarea Box-Jenkins ca fiind marimea de iesire a unui filtru necunoscut, la intrarea caruia se aplica o secventa de tip zgomot alb e(t), ca in figura 2.1.

Zgomotul alb este un semnal aleator ce are ca functie de autocorelatie impulsul Dirac si o putere spectrala constanta, diferita de zero, in toate frecventele spectrului (Figura 2.2).

Acest semnal nu poate fi obtinut practic, datorita imposibilitatii constructiei unor generatoare de putere infinita, fiind inlocuit in aplicatii practice cu semnale aleatoare sau deterministe cu proprietati asemanatoare.

Figura 2.1

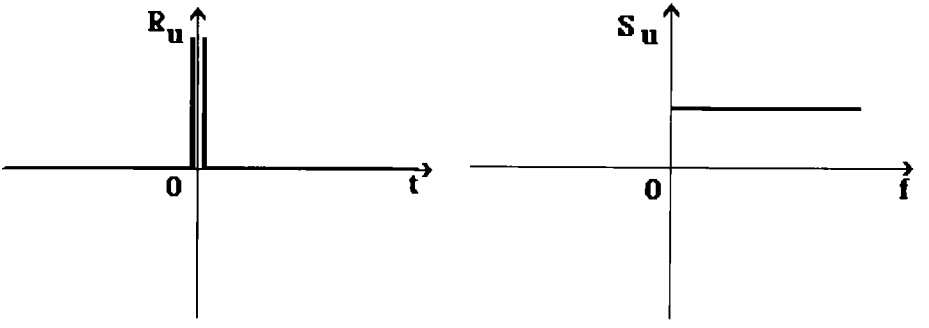


Figura 2.2: Functia de autocorelatie si functia densitate spectrala de putere a zgomotului alb

Principalele tipuri de modele utilizate in practica modelarii seriilor de timp sunt prezentate succint in cele ce urmeaza:

a) Model autoregresiv si de medie alunecatoare ARMA

Are urmatoarea structura:

$$z(t) = a_1 z(t-1) + a_2 z(t-2) + \ldots + a_{na} z(t-na) + b_0 e(t) + b_1 e(t-1) + b_2 e(t-2) + \ldots + b_{nb} e(t-nb) \qquad (2.1)$$

- unde:
  - z(t) - seria de timp (iesirea filtrului);
  - e(t) - zgomot alb (intrarea filtrului);
  - na - ordinul autoregresiei;
  - nb - ordinul mediei alunecatoare.

Modelele ARMA sunt utilizate preponderent in modelarea seriilor de timp stationare.

b) Model autoregresiv integrat si de medie alunecatoare ARIMA

Este un caz particular al modelului ARMA, pentru care polinomul:

$$1+a_1q+a_2q^2+\ldots+a_{na}q^{na}$$

are ca radacina pe 1, cu ordinul de multiplicitate d (d este ordinul de integrare).

Acest tip de model este utilizat in special pentru modelarea seriilor de timp nestationare omogene.

c) Model autoregresiv AR

Se obtine prin particularizarea modelelor ARMA considerându-se coeficientii b_1, b_2, ... , b_{nb} ca fiind zero:

$$z(t)=-a_1z(t-1)-a_2z(t-2)-\ldots-a_{na}z(t-na)+b_0e(t) \qquad (2.2)$$

d) Model de medie alunecatoare MA

Este un caz particular al modelelor ARMA in care coeficientii a_1, a_2, ... , a_{na} sunt zero:

$$z(t)=b_0e(t)+b_1e(t-1)+b_2e(t-2)+\ldots+b_{nb}e(t-nb) \qquad (2.3)$$

Ultimele doua clase de modele sunt des utilizate in modelarea seriilor de timp datorita faptului ca nu intotdeauna alegerea unui model ARMA aduce o precizie sensibil marita.

2.2.2 METODOLOGIA BOX-JENKINS [Pop91]

Metodologia de modelare Box - Jenkins reprezinta una din cele mai frecvent utilizate si precise tehnici de modelare si predictie a seriilor de timp. Metodologia consta in principal din urmatoarele etape:

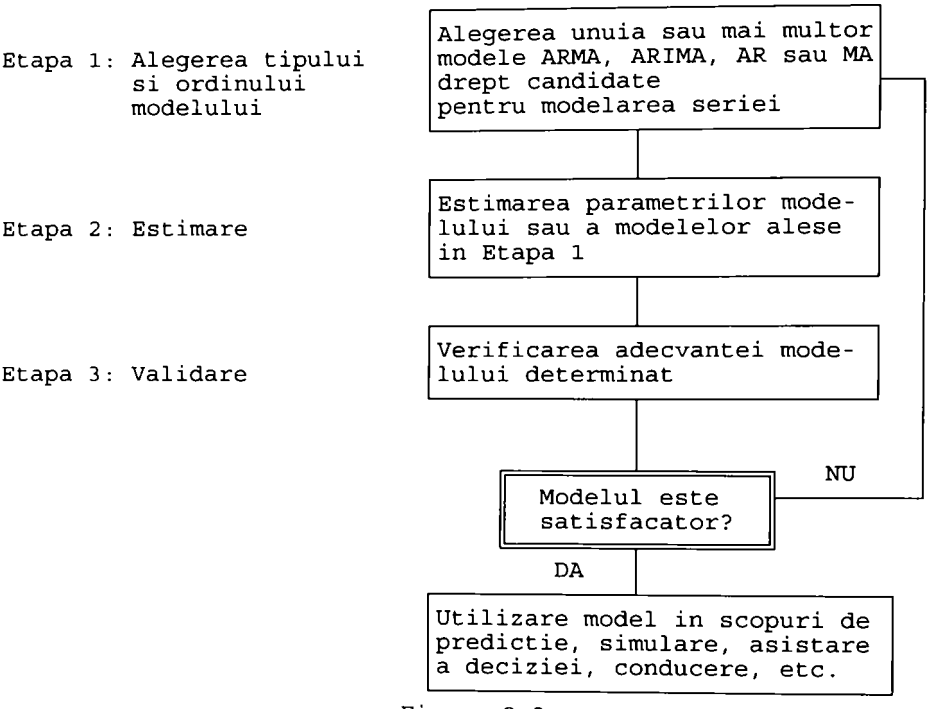


Figura 2.3

Etapa I : Alegerea tipului si ordinului modelului

- consta in alegerea unuia sau mai multor modele ARIMA, ARMA, AR sau MA ca si candidate pentru modelarea seriei, pe baza studiului alurii graficelor functiei de autocorelatie estimata si a functiei de autocorelatie partiala estimata (ABJ ofera practic un "sistem expert" pentru usurarea alegerii).

Aplicarea metodologiei Box-Jenkins face apel la utilizarea functiilor de autocorelatie estimata r_k (FAE) si a functiei de autocorelatie partiala estimata ϕ_{kk} (FAPE). Ideea de baza a analizei de autocorelatie consta in calculul coeficientului de corelatie pentru fiecare pereche ordonata (z_t, z_{t+k}).

Coeficientul de autocorelatie estimat al observatiilor separate prin k intervale de timp din cadrul aceleiasi serii de timp este notat cu r_k. Coeficientii r_k sunt valori statistice, determinate pentru un esantion de date si furnizeaza estimatii ale coeficientilor de autocorelatie teoretici. Coeficientul de autocorelatie estimat r_k reprezinta o masura a dependentei statistice dintre perechile ordonate ale observatiilor efectuate asupra a doua variabile aleatoare. Este un numar adimensional ce

poate lua valori cuprinse intre -1 si 1. Valoarea -1 are semnificatia unei corelatii negative perfecte, iar valoarea 1 a unei corelatii pozitive perfecte. Daca $r_k=0$, atunci z_{t+k} si z_t nu sunt corelate. Evident, eroarea de esantionare poate produce valori diferite de zero pentru r_k, chiar in cazul in care coeficientul de autocorelatie teoretica corespunzator este zero.

Formula de calcul a coeficientilor de autocorelatie este:

$$r_k = \frac{\sum_{t=1}^{n-k} [z(t)-E(z)]\,[z(t+k)-E(z)]}{\sum_{t=1}^{n} [z(t)-E(z)]^2} \qquad (2.4)$$

unde E este operatorul de mediere, iar n este numarul observatiilor (lungimea secventei).

Functia de autocorelatie partiala estimata FAPE este, in principiu, similara cu functia de autocorelatie estimata. Analiza de autocorelatie partiala se refera la masura gradului de corelatie dintre z_t si z_{t+k}, cu luarea in consideratie a efectelor valorilor seriei $\{z_t\}$ care intervin intre cele doua momente t si t+k. Coeficientul de autocorelatie partiala estimat, care masoara in aceste conditii dependenta dintre z_t si z_{t+k}, se noteaza cu ϕ_{kk} si reprezinta o marime statistica ce furnizeaza o estimatie a coeficientului de autocorelatie partiala teoretic.

Coeficientii de autocorelatie partiala sunt furnizati de urmatorul sistem de ecuatii recursive:

$$\phi_{11} = r_1 \qquad (2.5)$$

$$\phi_{kk} = \frac{r_k - \sum_{j=1}^{k-1} \phi_{k-1,j} r_{k-j}}{1 - \sum_{j=1}^{k-1} \phi_{k-1,j} r_j} \qquad (k=2,3,\ldots) \qquad (2.6)$$

unde:

$$\phi_{kj} = \phi_{k-1,j} - \phi_{kk}\phi_{k-1,k-j} \qquad (k=3,4,\ \ldots\ ;\ j=1,2,\ \ldots\ ,k-1) \qquad (2.7)$$

Metodologia Box-Jenkins ofera un "sistem expert" (la care s-a ajuns prin experimente) de alegere a tipului si ordinului modelului in functie de alura graficelor FAE si FAPE.

Modelul determinat in cadrul acestei etape reprezinta doar un candidat potential pentru modelul final al seriei. Pentru stabilirea modelului final urmeaza a fi parcurse etapele de estimare si de validare din cadrul metodologiei Box-Jenkins. Eventual se poate reveni la etapa de identificare daca modelul ales drept candidat, pentru seria analizata, se dovedeste inadecvat.

Etapa a II-a : Estimarea parametrilor modelului

- aceasta etapa contine determinarea estimatiilor pe baza de eficienta statistica ale coeficientilor (parametrilor) modelului ales in prima etapa. De asemenea, aceasta etapa furnizeaza informatii privind adecvanta modelului ales. In particular, daca coeficientii nu satisfac anumite conditii de tip inegalitate, pentru stationaritate si inversabilitate, modelul este respins.

In vederea estimarii parametrilor se folosesc metode clasice de estimare cum ar fi: metoda celor mai mici patrate liniare sau neliniare, metoda variabilei instrumentale, etc.

Etapa a III-a : Validarea si diagnoza modelului

- sunt parcurse câteva proceduri de validare si diagnoza (testul t, testul χ^2, reprezentarea grafica a reziduurilor, etc.) capabile sa orienteze utilizatorul la stabilirea adecvantei modelului ales si eventual la reformularea acestuia [Pop91].

Modelul final al seriei de timp este obtinut prin reluarea ciclul de identificare, estimare si validare-diagnoza pâna la obtinerea unor indicatori corespunzatori de calitate ai modelarii seriei de timp. Aplicarea iterativa a celor 3 etape ale metodologiei de modelare nu garanteaza faptul ca, in final, se va determina cel mai bun posibil model, dar sansele pentru aceasta cresc.

Modelul obtinut prin parcurgerea acestei metodologii poate fi utilizat in scopuri de predictie, conducere, simulare, asistare a deciziei, etc.

2.2.3 STUDIU DE CAZ - MODELAREA UNEI SERII REPREZENTÂND VITEZA VÂNTULUI

In cele ce urmeaza se va determina modelul vitezei vântului utilizând datele masurate zilnic pe durata a 70 de zile in Timisoara, date prezentate in Figura 2.4.

Pentru inceput a fost calculata functia de autocorelatie estimata si functia de autocorelatie partiala estimata, a caror grafice sunt reprezentate in figurile 2.5 si 2.6.

Analiza celor doua grafice releva faptul ca FAE si FAPE corespund din punct de vedere al alurii cu cele corespunzatoare

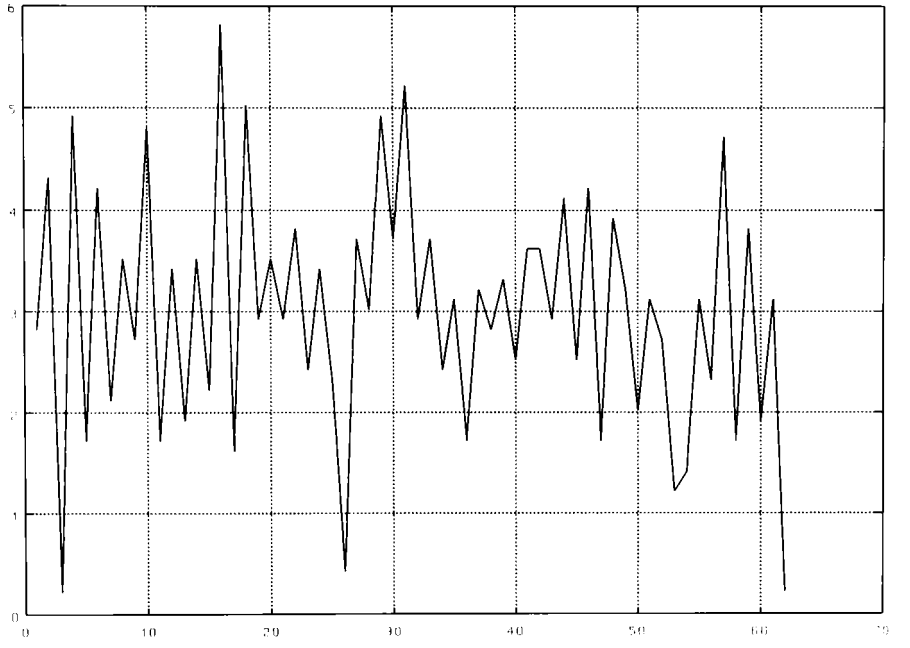


Figura 2.4

unui proces autoregresiv de ordinul 2, fapt pentru care, conform ABJ, un model AR2 ar fi indicat pentru modelare.

Pentru estimarea parametrilor modelului a fost utilizata metoda CMMP off-line liniara, ajungându-se in final la urmatorul model matematic al seriei:

$$v(t) = 5.1371 - 0.2481v(t-1) + 0.2406v(t-2) + e(t) \qquad (2.8)$$

Acest model a fost validat utilizând testele t si χ^2 prezentate in [Pop91].

Pe baza modelului determinat poate fi realizata cu usurinta predictia vitezei vântului pentru momente ulterioare de timp, predictie ce sta la baza conducerii propriu-zise a agregatelor eoliene, studiilor de amplasament si nu in ultimul rând a predictiei fenomenelor meteorologice.

In concluzie se poate afirma ca utilizarea metodologiei Box-Jenkins pentru modelarea si predictia vitezei vântului permite obtinerea unor modele de o buna precizie si rezonabile din punct de vedere dimensional.

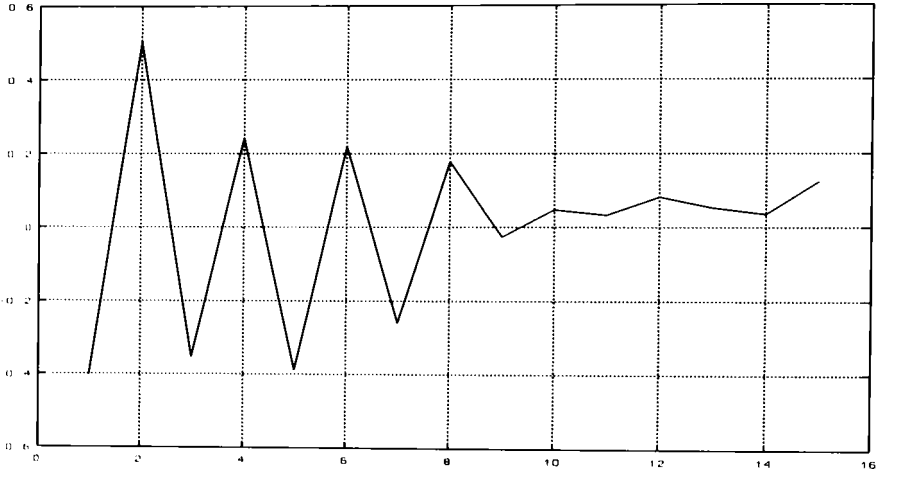


Figura 2.5

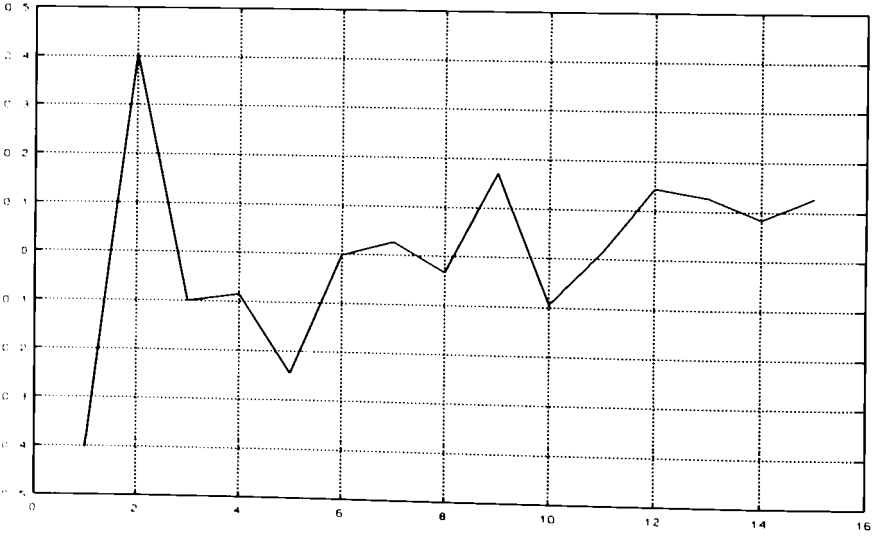


Figura 2.6

2.2.3.1 PROGRAM MATLAB PENTRU CALCULUL PARAMETRIILOR SERILOR DE TIMP UTILIZÂND METODOLOGIA BOX-JENKINS.

```
%INTRODUCEREA DATELOR MASURATE ALE VANTULUI
v=[2.82;4.32;0.22;4.92;1.72;4.22;2.12;3.52;2.72;4.82;1.72;3.42
1.92;3.52;2.22;5.82;1.62;5.02;2.92;3.52;2.92;3.82;2.42;3.42
2.32;0.42;3.72;3.02;4.92;3.72;5.22;2.92;3.72;2.42;3.12;1.72
3.22;2.82;3.32;2.52;3.62;3.62;2.92;4.12;2.52;4.22;1.72;3.92
3.22;2.02;3.12;2.72;1.22;1.42;3.12;2.32;4.72;1.72;3.82;1.92
3.12;0.22];
plot(v),grid,title('DATE MASURATE ALE VANTULUI')
meta g1
pause(5)
%CALCULUL FUNCTIEI DE AUTOCORELATIE ESTIMATA
n=length(v);
miu=mean(v);
z1=v-miu*ones(n,1);
jos=z1'*z1;
for k=1:n/4 383
sus=0;
for t=1:n-k
sus=sus+(v(t,1)-miu)*(v(t+k,1)-miu);
end
r(k)=sus/jos;
end
plot(r),grid,title('FUNCTIA DE AUTOCORELATIE ESTIMATA')
meta g2
pause(5)
%CALCULUL FUNCTIEI DE AUTOCORELATIE PARTIALA ESTIMATA
fi(1)=r(1);
a(1,1)=fi(1);
for k=2:n/4
   sum1=0;
   sum2=0;
   for j=1:k-1
     sum1=sum1+a(k-1,j)*r(k-j);
     sum2=sum2+a(k-1,j)*r(j);
   end
   fi(k)=(r(k)-sum1)/(1-sum2);
   a(k,k)=fi(k);
   for j=1:k-1
    a(k,j)=a(k-1,j)-fi(k)*a(k-1,k-j);
   end
end
plot(fi),grid,title('FUNCTIA DE AUTOCORELATIE PARTIALA ESTIMATA')
meta g3
pause(5)
%IDENTIFICAREA SERIEI DE TIMP
```

```
th=armax(v,[2 2]);
present(th)
end
```

2.2.4 DEZAVANTAJUL UTILIZARII ZGOMOTULUI ALB IN CADRUL ABORDARII BOX-JENKINS

Semnalul de tip zgomot alb, considerat a reprezenta intrarea filtrului din Figura 2.1, prezinta doua dezavantaje majore:

a) el nu poate fi obtinut practic; Problema teoretica a generarii zgomotului alb (semnal ipotetic ce contine armonici corespunzatoare tuturor frecventelor intre 0 si ∞) este echivalenta cu obtinerea unui generator de putere infinita, fapt imposibil.

b) fiind un semnal aleator, acesta nu poate fi reprodus. Aceasta inseamna ca ori de câte ori se doreste obtinerea unei secvente de tip zgomot alb, de tot atâtea ori secventele numerice obtinute vor fi diferite, dar identice din punct de vedere al proprietatilor statistice. Astfel, testul cel mai sugestiv - compararea seriei de timp reale cu cea obtinuta in urma simularii ei pe baza modelului identificat utilizând o relatie de tipul (2.1) nu poate fi utilizat.

Acest fapt a impus cautarea unor semnale deterministe care sa aproximeze zgomotul alb in domeniul frecventelor de interes. Un astfel de semnal ar putea fi semnalul pseudoaleator binar (SPAB), dar generarea unor SPAB-uri de lungime infinita, neperiodice (necesare identificarii on-line a seriei de timp) devine imposibila.

In cele ce urmeaza va fi utilizat ca semnal de intrare al filtrului un semnal haotic determinist, care prezinta proprietatea de a excita filtrul (figura 2.1) in toate frecventele cuprinse intr-o banda de interes (pentru vânt aceasta banda de frecvente ar putea fi aleasa intre 0 si 10Hz). Aceasta idee corespunde si asemanarii din punct de vedere al proprietatilor dintre semnalul haotic si vânt. Astfel, modelarea vântului devine mai "palpabila".

2.3 SISTEME DINAMICE HAOTICE

Teoria sistemelor haotice reprezinta unul din cele mai in voga domenii ale matematicii ultimilor ani.

Cu toate ca domeniul este inca la inceput, teoriile matematice in acest domeniu nefiind pe deplin inchegate, datorita revolutiei in tehnologia calculatoarelor, a fost posibil studiul unor sisteme neliniare ce modeleaza fenomene fizice, biologice, sociale, economice, dovedindu-se comportamentul lor haotic [Arn91], [Aro91], [Ber88], [Bec89], [Bed91], [Blo89], [Mar91], [Mie90], [Pas94], [Ric90],

[Shi90], [Vak90], [Wer93].

Primele studii in domeniul sistemelor cu evolutie haotica au fost prezentate in anul 1963 de catre doi cercetatori: S.Smale [Sma63_1] [Sma63_2] si E. Lorenz [Lor63]. Smale a studiat dinamica unei aplicatii f in spatiul $\mathbb{R}^2$, numita si aplicatia potcoava de cal, aratând ca aceasta aplicatie are un comportament neobisnuit, si anume pe o submultime compacta, invarianta, f are o orbita densa si o multime numarabila de orbite periodice [Hol89].

In incercarea de a prezice evolutia vremii pe o perioada indelungata de timp, Lorenz [Lor63] a analizat evolutia unui sistem in spatiul tridimensional $\mathbb{R}^3$. Datorita dependentei sensibile de conditiile initiale (erori mici in alegerea starilor initiale conduc la devieri majore in evolutia lor) Lorenz a dovedit imposibilitatea prezicerii pe termen lung a vremii. In acelasi timp a observat ca traiectoriile starilor sistemului sunt atrase de o forma geometrica bizara numita atractor straniu.

In continuare sunt prezentate caracteristicile sistemelor dinamice haotice, precum si câteva exemple remarcabile, ce pot fi utilizate in generarea unor semnale haotice cu scopul declarat al aproximarii zgomotului alb in domeniul de frecvente caracteristic fenomenului natural - vânt.

2.3.1 PREZENTARE GENERALA. DEFINITII [PeE92], [ScR91], [Fal90] [Ott90]

Fie (X,d) un spatiu metric compact si $f:X\to X$ o aplicatie continua.

Definitie.

Sistemul dinamic $f:X\to X$ este topologic tranzitiv daca exista x_0 apartinând lui X astfel incât orbita lui x_0 sa fie densa in X, sau cu alte cuvinte, exista puncte initiale in X cu orbite ce trec arbitrar de aproape de orice punct a lui X.

Definitie.

Dinamica lui f (sau simplu f) depinde sensibil de conditiile initiale daca există M>0 astfel incât pentru orice x apartinând lui X si pentru orice vecinatate V a lui x exista y apartinând lui V si n apartinând lui $\mathbb{N}^*$ cu proprietatea ca $d(f^n(x), f^n(y))>M$.

Prin $f^n(x)$ s-a notat punctul actual al orbitei pornita din punctul initial x, obtinut prin aplicarea succesiva de n ori a aplicatiei f.

Definitie.

Fie (X,d) un spatiu metric, $f:X\to X$ o aplicatie invarianta, continua si A o submultime compacta a lui X. Dinamica lui f este haotica pe A (sau mai simplu f este haotica pe A) daca sunt indeplinite urmatoarele trei conditii:

1) dinamica lui f depinde sensibil de conditiile initiale;
2) f:A→A este topologic tranzitiv;
3) multimea punctelor periodice ale aplicatiei f pe multimea A este densa in A.

Aceasta poate fi considerata o definitie completa, deoarece dupa unii autori [Wig88][Wig90], pentru ca f sa fie haotica pe A sunt suficiente a fi indeplinite primele doua conditii.

2.3.2 SISTEME HAOTICE REMARCABILE [All89][Rue89].

2.3.2.1 ATRACTORUL LORENZ

In anul 1963, E.N. Lorenz a publicat un articol remarcabil intitulat "Deterministic Non-Periodic Flow" [Lor63], articol ce a marcat inceputul unui domeniu fascinant: sistemele haotice. Lorenz era in cautarea unui sistem de ecuatii diferentiale in spatiul $\mathbb{R}^3$ capabile sa modeleze evolutia fenomenelor meteorologice. Ecuatiile pe care acesta le propune sunt prezentate in continuare:

$$\frac{dx_1}{dt} = -\sigma x_1 + \sigma x_2 \qquad (2.9)$$

$$\frac{dx_2}{dt} = r x_1 - x_2 - x_1 x_3 \qquad (2.10)$$

$$\frac{dx_3}{dt} = x_1 x_2 - b x_3 \qquad (2.11)$$

unde σ, r si b sunt parametrii reali pozitivi.

In figura 2.7 este prezentata evolutia in timp a acestui sistem intr-un spatiu al fazelor, tridimensional (x_1, x_2, x_3), pentru σ=10, b=8/3, r=28, iar in figurile 2.8, 2.9, 2.10 sunt prezentate evolutiile in timp ale fiecareia dintre cele trei variabile de stare.

Dupa cum se observa, in sistemul (2.9)-(2.11) nu apare nici o marime de intrare, sistemul evoluând liber dintr-o stare initiala data. In literatura [Vin91], acest sistem este completat prin introducerea unei marimi de comanda in relatia (2.10) in scopul studierii posibilitatilor de conducere in sisteme haotice. In acest caz, sistemul (2.9)-(2.11) se poate rescrie sub forma:

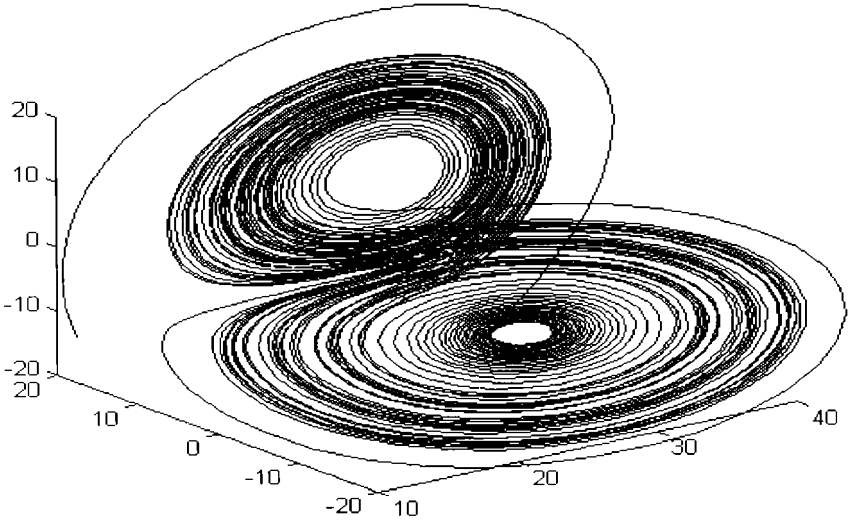


Figura 2.7: Evolutia in timp a sistemului Lorenz

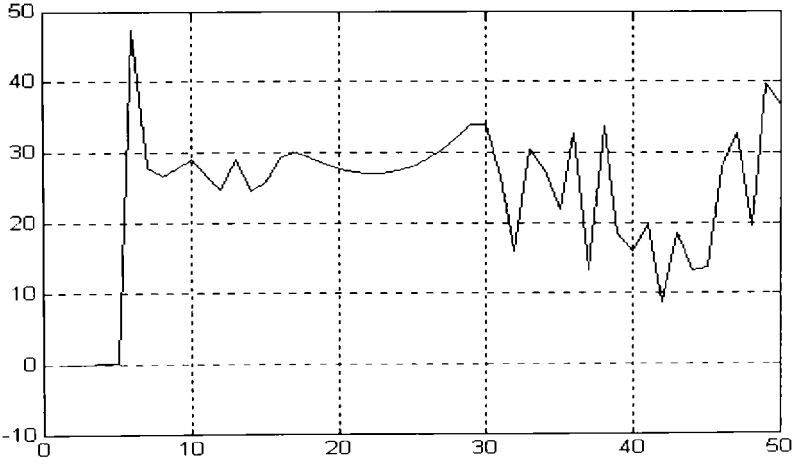


Figura 2.8

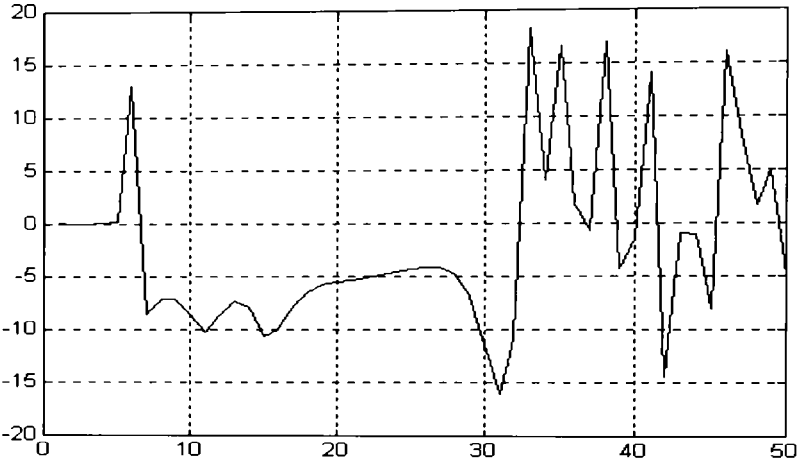


Figura 2.9

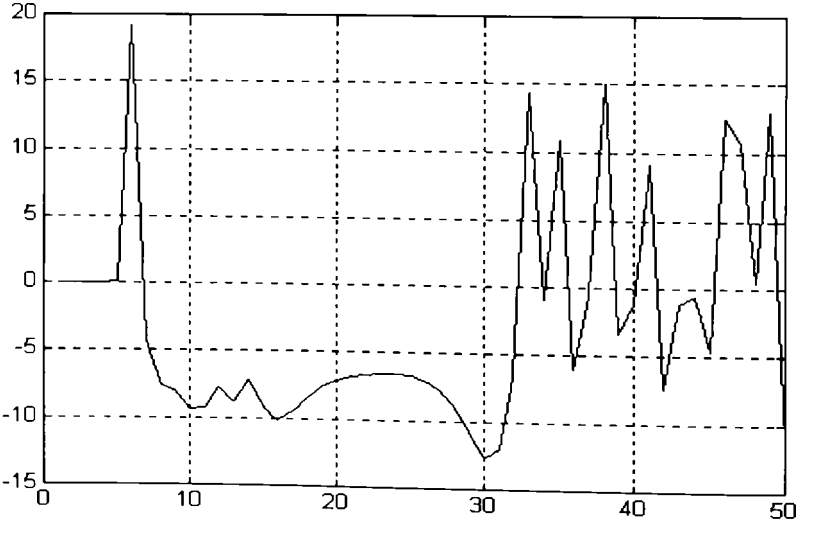


Figura 2.10

$$\frac{dx_1}{dt} = -\sigma x_1 + \sigma x_2 \qquad (2.12)$$

$$\frac{dx_2}{dt} = r x_1 - x_2 - x_1 x_3 + u \qquad (2.13)$$

$$\frac{dx_3}{dt} = x_1 x_2 - b x_3 \qquad (2.14)$$

Atractori

Observând figura 2.7 se poate aprecia existenta a doi atractori. Determinarea coordonatelor acestora poate fi realizata prin anularea derivatelor in raport cu timpul din relatiile (2.9), (2.10) si (2.11):

$$-\sigma x_1 + \sigma x_2 = 0 \qquad (2.15)$$

$$r x_1 - x_2 - x_1 x_3 = 0 \qquad (2.16)$$

$$x_1 x_2 - b x_3 = 0 \qquad (2.17)$$

Prin rezolvarea acestui sistem de ecuatii rezulta trei solutii:

a) solutia banala $x_1=x_2=x_3=0$, punctul (0,0,0) fiind un punct stationar.

b) cei doi atractori pentru cazul in care parametrul r>1:

$$x_1 = x_2 = \pm \sqrt{b(r-1)} \qquad (2.18)$$

$$x_3 = r - 1 \qquad (2.19)$$

Deci, cei doi atractori corespunzatori sistemului Lorenz, pentru valorile parametrilor considerate anterior (σ=10, b=8/3, r=28), vor avea coordonatele:

$$C1 = \left(\sqrt{b(r-1)} , \sqrt{b(r-1)} , (r-1) \right) = (8.4852 , 8.4852 , 27) \qquad (2.20)$$

$$C2 = \left(-\sqrt{b(r-1)} , -\sqrt{b(r-1)} , (r-1) \right) = (-8.4852 , -8.4852 , 27) \qquad (2.21)$$

In scopul obtinerii unui semnal haotic discret, este necesara liniarizarea modelului matematic (2.9)-(2.11), precum si discretizarea modelului liniarizat, fapt ce implica (datorita erorilor de liniarizare si discretizare) pierderea caracterului

haotic. Totusi semnale discrete cu caracteristici haotice se pot obtine prin simularea sistemului Lorenz (neliniar si continual) si esantionarea semnalului continual obtinut.

2.3.2.2 ATRACTORUL RÖSSLER

In anul 1976, Otto E. Rössler a descoperit un sistem simplu, care permite, dupa toate probabilitatile, obtinerea cea mai facila a haosului in sisteme continue [All89][Rue89]. Ecuatiile pe care acesta le propune sunt prezentate in continuare:

$$\frac{dx_1}{dt} = -x_2 - x_3 \qquad \textbf{(2.22)}$$

$$\frac{dx_2}{dt} = x_1 + ax_2 \qquad \textbf{(2.23)}$$

$$\frac{dx_3}{dt} = x_1x_3 - cx_3 + b \qquad \textbf{(2.24)}$$

unde a,b,c sunt parametrii reali pozitivi, a fiind cuprins intre 0 si 2.

In figura 2.11 este prezentata evolutia in timp a acestui sistem intr-un spatiu al fazelor, tridimensional (x_1, x_2, x_3), pentru a=0.2, b=0.2, c=5.7, iar in figurile 2.12, 2.13, 2.14 sunt prezentate evolutiile in timp ale fiecareia dintre cele trei variabile de stare.

Dupa cum se observa, in sistemul (2.22)-(2.24) nu apare nici o marime de intrare, sistemul evoluând liber dintr-o stare initiala data. Sistemul poate fi completat prin introducerea unei marimi de comanda intr-una din cele trei relatii, de exemplu in prima in scopul studierii posibilitatilor de conducere in sisteme haotice. In acest caz, sistemul (2.22)-(2.24) se poate rescrie sub forma:

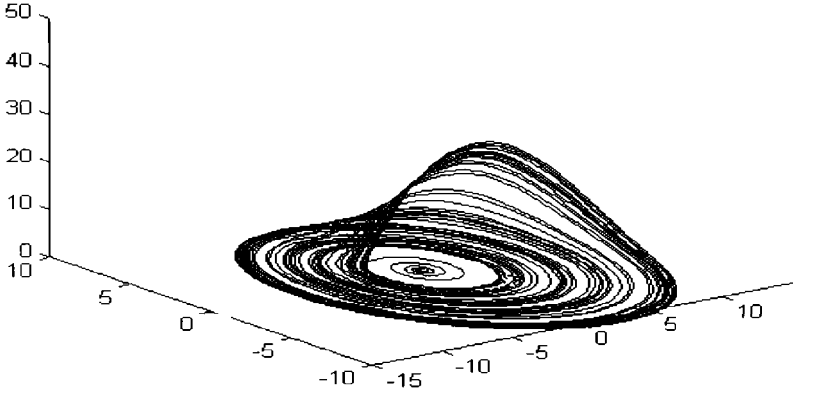


Figura 2.11: Evolutia in timp a sistemului Rössler

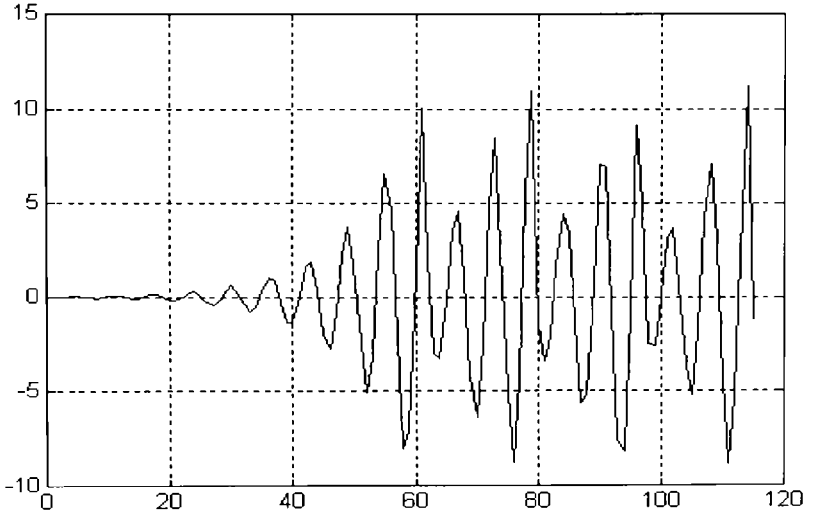


Figura 2.12

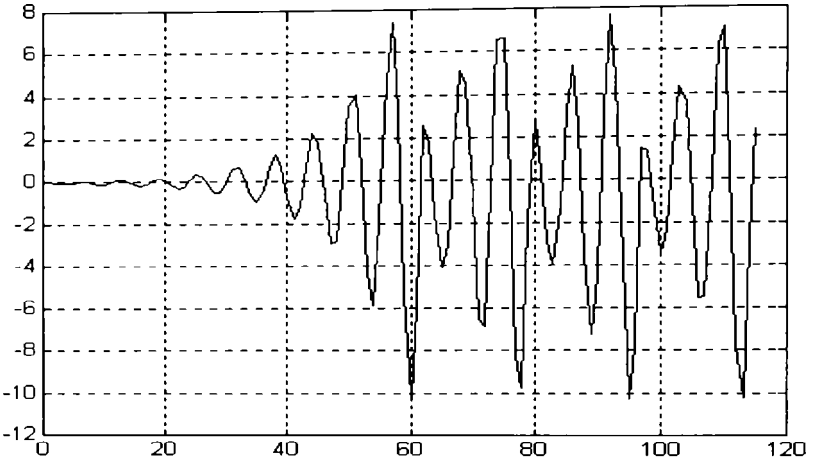


Figura 2.13

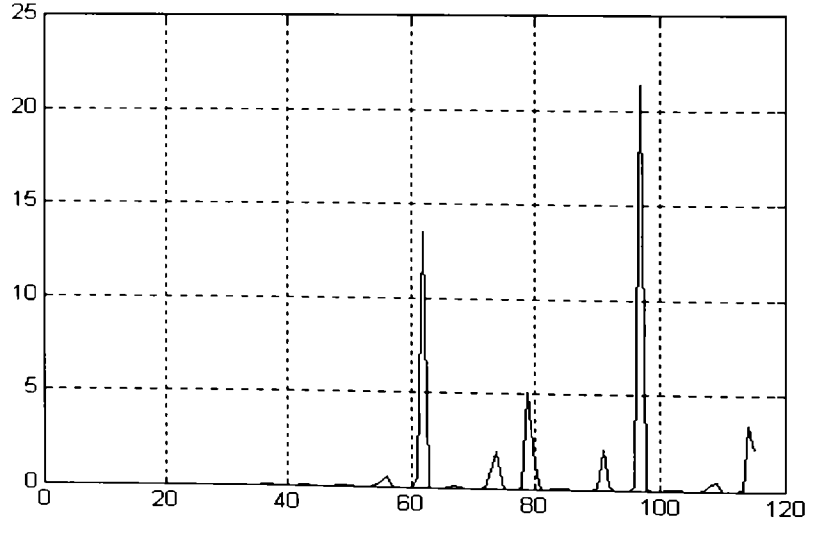


Figura 2.14

$$\frac{dx_1}{dt} = -x_2 - x_3 + u \qquad (2.25)$$

$$\frac{dx_2}{dt} = x_1 + ax_2 \qquad (2.26)$$

$$\frac{dx_3}{dt} = x_1x_3 - cx_3 + b \qquad (2.27)$$

Atractori

Observând figura 2.11 se poate aprecia existenta a doi atractori. Determinarea coordonatelor acestora poate fi realizata prin anularea derivatelor in raport cu timpul din relatiile (2.22), (2.23) si (2.24):

$$x_2 + x_3 = 0 \qquad (2.28)$$

$$x_1 + ax_2 = 0 \qquad (2.29)$$

$$x_1x_3 - cx_3 + b = 0 \qquad (2.30)$$

Prin rezolvarea acestui sistem de ecuatii rezulta doua solutii, reprezentând cei doi atractori:

$$x_1 = \frac{c \pm \sqrt{c^2-4ab}}{2} \qquad (2.31)$$

$$x_2 = -\frac{c \pm \sqrt{c^2-4ab}}{2a} \qquad (2.32)$$

$$x_3 = \frac{c \pm \sqrt{c^2-4ab}}{2a} \qquad (2.33)$$

cu conditia:

$$c^2-4ab > 0 \qquad (2.34)$$

Deci, cei doi atractori corespunzatori sistemului Rössler, pentru valorile parametrilor considerate anterior (a=0.2, b=0.2, c=5.7), vor avea coordonatele:

$$C1 = (5.69295 , -28.4647 , 28.4647) \qquad (2.35)$$

$$C2 = (0.007 , -0.0351 , 0.0351) \qquad (2.36)$$

In scopul obtinerii unui semnal haotic discret, este necesara liniarizarea modelului matematic (2.22)-(2.24), precum si discretizarea modelului liniarizat, fapt ce implica (datorita erorilor de liniarizare si discretizare) pierderea caracterului haotic. Totusi semnale discrete cu caracteristici haotice se pot obtine prin simularea sistemului Rössler (neliniar si continual) si esantionarea semnalului continual obtinut.

2.3.2.3 ATRACTORUL HENON

In anul 1976, astronomul francez Michel Henon a sugerat un model discret simplificat corespunzator dinamicii haotice a sistemului Lorenz [Hen76_1] [Hen76_2]. Datorita simplitatii modelului propus, acesta este des utilizat in simularile numerice ale haosului [Ben91] [Ott90] [War94] [Lai94] [Tar94].

Modelul Henon este descris prin urmatorul sistem de relatii recurente:

$$x_{k+1} = y_k + 1 - ax^2 \qquad (2.37)$$

$$y_{k+1} = bx_k \qquad (2.38)$$

unde valorile parametrilor sugerate de Henon sunt a=1.4 si b=0.3. Deci, sistemul Henon este:

$$x_{k+1} = y_k + 1 - 1.4x^2 \qquad (2.39)$$

$$y_{k+1} = 0.3x_k \qquad (2.40)$$

In figura 2.15 este prezentat atractorul Henon, in coordonate (x,y), iar in figurile 2.16 si 2.17 sunt prezentate variatiile in timp a marimilor x si y.

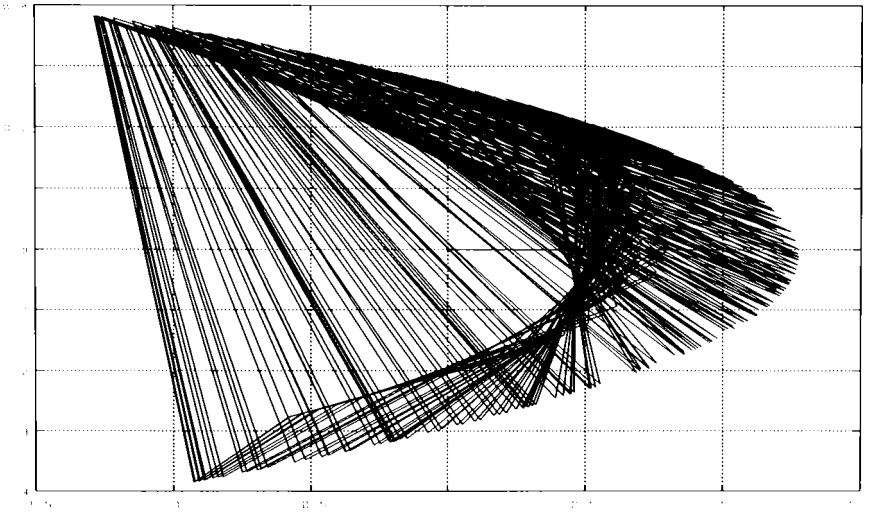


Figura 2.15: Evolutia in timp a sistemului Henon

Pentru simularea haosului poate fi utilizat fie x fie y, utilizând relatiile recursive urmatoare:

$$x_k = 1 - 1.4x_{k-1}^2 + 0.3x_{k-2} \qquad \textbf{(2.41)}$$

$$y_k = 0.3 - 4.6666y_{k-1}^2 + 0.3y_{k-2} \quad , \qquad \textbf{(2.42)}$$

relatii deduse prin eliminarea lui y, respectiv x din relatiile (2.39), (2.40).

Relatia (2.41) este utilizata pentru generarea semnalului haotic utilizat pentru identificarea modelului seriei de timp a vântului in cadrul pachetului de programe propus in acest capitol [PeV92].

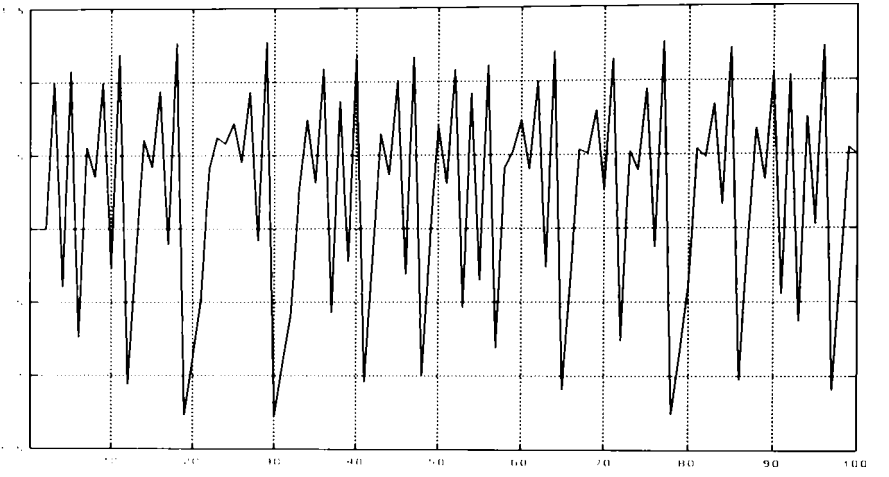


Figura 2.16

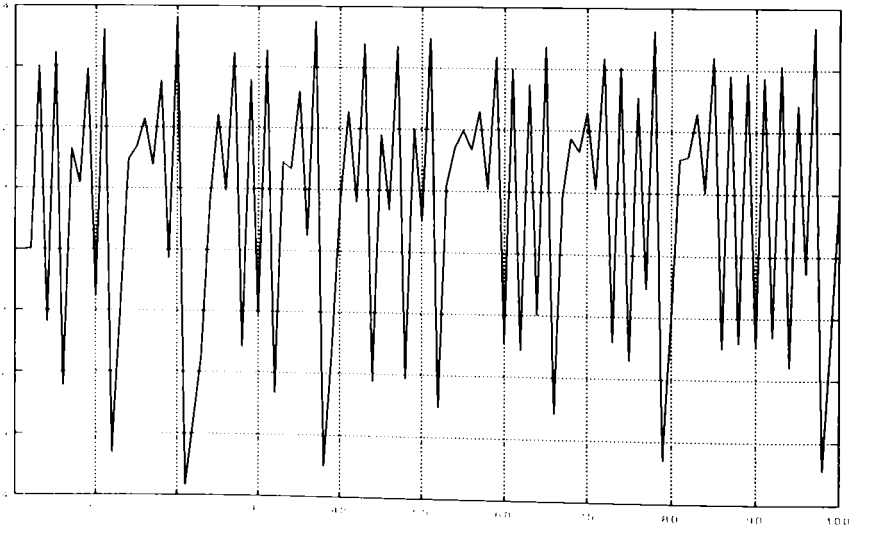


Figura 2.17

2.4 SOLUTIE DE IDENTIFICARE ON-LINE A MODELULUI VÂNTULUI

In acest paragraf este propusa o metodologie de identificare si predictie on-line a vitezei vântului utilizând un semnal haotic discret. La baza acestei metodologii sta abordarea Box-Jenkins, in care semnalul de intrare al filtrului (zgomotul alb) este inlocuit cu un semnal haotic, semnal ce indeplineste conditia de a excita filtrul in intreg domeniul frecventelor de interes (in cazul vântului acest domeniu este cuprins intre 0 si 10Hz).

Figura 2.18

Semnalul haotic discret aplicat la intrarea filtrului necunoscut din figura 2.18 este una dintre marimile de stare ale sistemului Henon, sistem discutat in paragraful 2.3.2.3:

$$x_k = 1 - 1.4x_{k-1}^2 + 0.3x_{k-2} \qquad (2.43)$$

Identificarea modelului filtrului, si implicit al modelului seriei de timp asociate vântului se desfasoara conform procedurii descrise in figura 2.3, cu urmatoarele particularitati impuse de functionarea on-line a algoritmului de identificare:

Etapa I: Structura modelului seriei de timp trebuie sa satisfaca compromisul dintre precizia de modelare - model de un ordin cât mai ridicat - si limitarile impuse de functionarea in timp real - model de un ordin cât mai redus. Se considera ca un model ARMA (relatia(2.1)) de ordinul 2, 3, maxim 4, satisface acest compromis in cazul identificarii on-line vântului.

Etapa II: In vederea estimarii parametrilor modelului a carui structura a fost aleasa in etapa anterioara se vor utiliza algoritmi robusti la impreciziile calculului matematic (datorita lungimii finite de reprezentare interna a variabilelor in memoria calculatorului), algoritmi derivati din metoda standard a celor mai mici patrate cum sunt U-D si SQ.

Etapa III: Calculele statistice, utilizate de obicei in faza de validare a modelului, nu se recomanda a fi utilizate in cazul unei identificari in timp real, deoarece acestea maresc nejustificat de mult timpul de calcul/perioada de esantionare. Din acest motiv, o

comparare vizuala a seriei de timp originale cu seria de timp simulata pe baza modelului identificat este mult mai simpla din punct de vedere al calculelor si mai "palpabila".

Modelele determinate, urmând aceasta metodologie, pot fi folosite in vederea predictiei vitezei vântului, precum si pentru implementarea unor tehnici de conducere adaptiva.

De o deosebita importanta in obtinerea unor modele viabile ale vântului este etapa a II, etapa ce va fi dezvoltata in cele ce urmeaza.

2.4.1 ESTIMAREA PARAMETRILOR MODELULUI SERIEI DE TIMP

La baza dezvoltarii unor tehnici de estimare cu o functionare "sigura" in timp real sta metoda clasica a celor mai mici patrate, varianta on-line [Ter87] [Lju87].

2.4.1.1 METODA CELOR MAI MICI PATRATE - VARIANTA ON-LINE

Se considera modelul ARMA rescris sub forma:

$$y(t) = \Phi(t)^T \Theta(t) + \epsilon(t) \qquad \textbf{(2.44)}$$

unde vectorul masuratorilor (al regresorilor) este:

$$\Phi(t) = [-y(t-1) \ \ldots \ y(t-na) \ \ u(t-1) \ \ldots \ u(t-nb)]^T \qquad \textbf{(2.45)}$$

iar vectorul parametrilor ce vor fi estimati este:

$$\theta(t) = [a_1(t) \ \ldots \ a_{na}(t) \ \ b_1(t) \ \ldots \ b_{nb}(t)]^T \qquad \textbf{(2.46)}$$

Se poate observa ca in ultima relatie, parametrii sunt functii de timp, fapt caracteristic estimatorilor on-line.

Algoritmul recursiv CMMP [Got95] [Ter80] [Lju85] [Lju86] [Pro85] [Pro94] [Sod87] [Wah86] este descris prin urmatorul set de relatii:

$$\theta(t) = \theta(t-1) + K(t)\epsilon(t) \qquad \textbf{(2.47)}$$

$$\varepsilon(t) = y(t) - \Phi(t)^T \Theta(t-1) \qquad (2.48)$$

$$K(t) = \frac{P(t-1)\Phi(t)}{\lambda + \Phi(t)^T P(t-1)\theta(t)} \qquad (2.49)$$

$$P(t) = \frac{P(t-1) - K(t)\Phi(t)^T P(t-1)}{\lambda} \qquad (2.50)$$

In aceste relatii P(.) este matricea de covarianta, λ este factorul de uitare, iar K(.) este o marime de calcul.

Implementarea acestui set de relatii recurente este facila. In esenta, urmatoarele operatii trebuiesc a fi efectuate pentru adaptarea informatiei disponibile la momentul (t-1):

a) adaptare pentru pasul actual:

$$\Phi(t-1) \rightarrow \Phi(t)$$

b) calcule:

$$x = P(t-1)\Phi(t)$$

$$\sigma = \lambda + \Phi(t)^T x$$

$$K(t) = \frac{x}{\sigma}$$

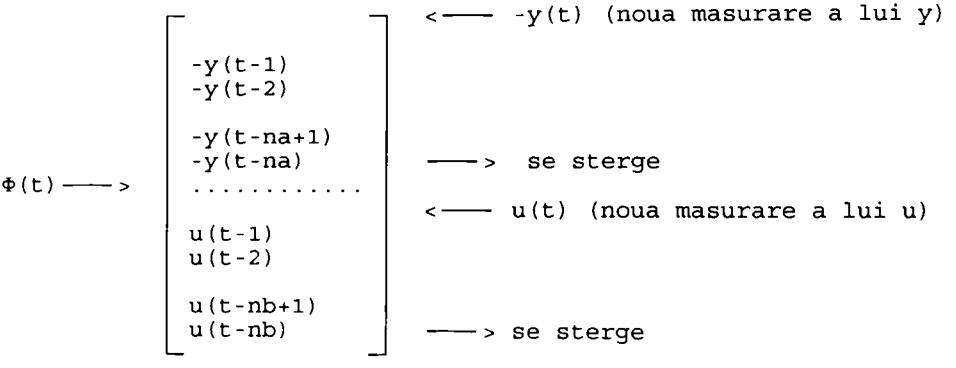
$$\varepsilon(t) = y(t) - \Phi(t)^T \Theta(t-1)$$

c) adaptare pentru pasul urmator:

$$\theta(t) = \theta(t-1) + K(t)\epsilon(t) \qquad (2.103)$$

$$P(t) = \frac{P(t-1) - K(t)x^T}{\lambda}$$

Adaptarea vectorului masuratorilor este extrem de evidenta :



Analizând relatiile (2.47)-(2.50) se observa urmatoarele:
- $\theta(t)$ este o estimatie a reziduului la momentul t al modelului având parametrii furnizati de vectorul $\theta(t-1)$;
- termenul de corectie al lui $\theta(t-1)$ este proportional cu eroarea de estimare $\epsilon(t)$;
- daca P(t-1) este "mic" atunci K(t) va fi "mic" si deci termenul de corectie din (2.47) va avea valori mici; in general, acest termen va avea valori mari daca P(.) este "mare";
- in mod aproximativ, P(.) este invers proportional cu λ. In consecinta, termenul de corectie din (2.47) poate fi, de exemplu, crescut prin micsorarea lui λ.

Aceste observatii stau la baza determinarii unor modalitati practic acceptabile pentru initializarea algoritmului CMMP on-line. Acest algoritm recursiv trebuie initializat prin precizarea marimilor $\theta(0)$, si P(0). De asemenea trebuie aleasa valoarea factorului de uitare λ. In continuare, se dau unele indicatii, mentionate in biografia de specialitate [Lju87][Pro85][Ter87], pentru alegerea acestor trei cantitati.

- *Alegerea lui $\theta(0)$:*

Cel mai frecvent, in practica nu se dispune de estimatii initiale ale parametrilor θ; in lipsa unor informatii apriorice despre θ, o alegere rezonabila pentru $\theta(0)$ este:

$$\Theta(0) = 0 \qquad (2.51)$$

Daca se dispune de informatii apriorice in ceea ce priveste parametrii atunci componentele vectorului $\theta(0)$ se aleg in conformitate cu aceste informatii.

- *Alegerea lui P(0):*

Deoarece, in general, vectorul parametrilor θ va fi departe de valoarea sa "optima", $P(0)$ trebuie ales cu elemente suficient de mari pentru a permite corectii semnificative ale estimatiei initiale $\theta(0)$. O alegere simpla si frecvent utilizata a lui $P(0)$ este urmatoarea:

$$P(0) = \alpha I \qquad (2.52)$$

unde α este un scalar ales pentru a raspunde compromisului intre convergenta lenta in cazul α "prea mic" si prezenta oscilatiilor in amplitudine ale componentelor vectorului $\theta(.)$ in cazul α "prea mare". Valori indicate pentru α sunt de ordinul sutelor, miilor sau chiar a zecilor de mii.

In cazul unei informatii apriorice relativ la valorile parametrilor, α se va alege de ordinul zecilor.

- *Alegerea lui λ:*

Daca sistemul in studiu are parametri care variaza in timp se va alege $\lambda<1$ (cu atât mai mic decât 1 cu cât parametrii procesului variaza mai rapid in timp). Totusi λ nu trebuie sa fie mult mai mic decât 1 pentru a nu produce oscilatii ale estimatiilor. In mod uzual se alege λ in plaja (0.95, 0.995).

Daca procesul investigat este invariant in timp, atunci ar fi de asteptat sa se alega $\lambda=1$. Totusi, deoarece reziduurile $\epsilon(t)$ pentru valori reduse ale lui t (t=1,2,...) corespund unor estimatii imprecise ale parametrilor (datorita alegerii destul de arbitrare a lui $\theta(0)$) aceste reziduuri ar trebui deponderate in criteriul CMMP, prin modificarea factorului de uitare dupa o schema simpla: $\lambda<1$ pentru primele câteva zeci de esantioane: t=1,2,...N1 si $\lambda=1$ t=N1+1, N1+2,...

2.4.1.2 ALGORITMI RECURSIVI DE ESTIMARE OBTINUTI PRIN UTILIZAREA FACTORIZARILOR MATRICIALE

In acest paragraf se va discuta urmatoarul aspect important referitor la procedura CMMP recursiva prezentata anterior: pentru orice procedura recursiva erorile de rotunjire (inerent asociate implementarii pe un calculator numeric) se pot cumula si deci pot avea un efect detrimental asupra rezultatelor finale. Este algoritmul (2.47)-(2.50) robust la acest tip de erori?

Diverse studii [Lju87][Sim89] au aratat ca algoritmul recursiv CMMP este instabil numeric in sensul ca erorile de rotunjire nu se acumuleaza in timpul efectuarii iteratiilor cu ecuatiile algoritmului. Aceste erori afecteaza estimatiile $\{\theta(t)\}$ furnizate de (2.47)-(2.50). Pentru robustificarea algoritmului recursiv CMMP la erori de rotunjire s-au propus o serie de variante ameliorate ale lui. Toate aceste variante sunt echivalente matematic cu (2.47)-(2.50), dar comportarea lor numerica este superioara. In continuare se prezinta doua tipuri de factorizari matriciale (U-D si SQ), eficiente si atractive, ce se aplica matricii de covarianta P(.) sau inversei acesteia numita matricea de informatie, factorizari cu efecte benefice asupra comportarii numerice ale algoritmului CMMP on-line:

- *Factorizarea U-D*

$$P(t) = U(t) \cdot D(t) \cdot U(t)^T \qquad \textbf{(2.53)}$$

unde U este o matrice superior triunghiulara, iar D o matrice diagonala.

- *Factorizarea Cholesky sau SQ (radacina patrata)*

$$P(t) = S(t) \cdot S(t)^T \qquad \textbf{(2.54)}$$

unde S este o matrice superior triunghiulara.

Algoritmii de factorizare, prezentati in bibliografia de specialitate [Sim89][Pro91_2] au la baza rotatiile plane standard sau modificate. Ei au fost testati obtinându-se rezultate bune in toate cazurile studiate. Nu s-a putut stabili experimental superioritatea din punct de vedere numeric a unui algoritm asupra altuia.

2.5 PACHET DE PROGRAME DESTINAT ESTIMARII MULTIVARIABILA ON-LINE A SERIILOR DE TIMP - SIMULATORUL SERTIM 1.0.

Pachetul SERTIM 1.0 este un mediu integrat destinat simularii procesului de estimare a seriilor de timp utilizând metodele CMMP, U-D si SQ.

In decursul fiecarei perioade de esantionare se desfasoara doua etape: etapa de simulare a seriei de timp si etapa de estimare.

- *Etapa de simulare*

Aceasta etapa inlocuieste masurarea valorii momentane a seriei

de timp, adica a vitezei vântului.

Valoarea momentana a seriei de timp ce va fi identificata (valoare care, in practica, ar trebui masurata) este obtinuta prin simularea functionarii unui filtru cunoscut (figura 2.18) când la intrarea acestuia se aplica semnalul haotic (2.43), in conditiile prezentei unui zgomot aditiv aleator cu o distributie normala la iesire.

■ *Etapa de estimare a parametrilor*

In etapa de estimare se considera drept marimi cunoscute doar valorile momentane si anterioare ale intrarii (semnalul haotic) si iesirii filtrului (parametrii filtrului se considera necunoscuti). Pe baza acestor date intrare-iesire se determina valorile parametrilor utilizând algoritmi de estimare on-line.

Aprecierea calitatii procesului de estimare se face prin compararea parametrilor filtrului folosit in etapa de simulare cu parametrii estimati ai aceluiasi filtru determinati in etapa de estimare.

Una dintre importantele facilitati oferite utilizatorului consta in estimarea unor modele multivariabile pe iesire, practic a mai multor serii de timp simultan. Astfel ar putea fi identificate modele ale vântului in diferite puncte de masura, caz intâlnit in practica in special pentru predictia vitezei vântului pentru o arie energetica eoliana.

Pachetul software este dotat cu o baza de date - fisierul model1.dta - care contine modele matematice ale unor filtre mono sau multivariabile. Aceasta baza de date poate fi modificata din program cu optiunile new si del, din fereastra 1 putând fi adaugate sau sterse cimpuri (modele).

Rezultatele estimarii sunt afisate grafic pe ecran (doar pe display SVGA pentru o calitate satisfacatoare) utilizând in acest scop driverul egavga.bgi si pot fi listate la imprimanta.

Programul sertim.exe este scris in limbajul de programare C utilizând compilatorul Turbo C 2.0 al firmei Borland.

Datorita dimensiunii sursei s-a folosit optiunea Project din meniul editorului tc.exe cu fisierul sertim.prj. Acesta contine modulele C corespunzatoare partilor functionale mai importante ale programului. Interfata cu utilizatorul este "user-friendly" bazându-se pe principiul ferestrelor.

Fisierul de help - sertim.hlp contine informatii despre fiecare dintre optiunile din meniul principal al sertim.exe si este citit in timpul rularii la apasarea tastei F1.

Pachetul SERTIM contine deci 4 fisiere:

sertim.exe
sertim.hlp
egavga.bgi
model1.dta

2.5.1 STRUCTURA SI PRINCIPALELE COMPONENTE ALE PACHETULUI SERTIM 1.0 [Cu93]

Programul executabil sertim.exe este obtinut prin compilarea unor module C cu ajutorul optiunii Project.

Modulul main.c contine functia main() si joaca rolul de manager al diferitelor taskuri implementate in celelalte module ale programului. La lansarea in executie se initializeaza mai intâi diferitele variabile globale in functia initt(). Apoi, dupa afisarea meniului principal si a background-ului in functia firma(), se intra intr-un ciclu while (in main()) pentru citire tastatura. La detectarea codurilor de tastatura corespunzatoare cursorului se schimba fereastra (functia winx()) sau se modifica portiunea in invers video din fereastra curenta (winy()).

La apasarea tasei Enter controlul este dat functiei dispune() care hotareste care task va fi lansat in executie functie de pozitia in interiorul ferestrei (kly) si de numarul acesteia (klx). Aceasta procedura este des folosita si in alte module si creaza meniul user-friendly cu ferestre.

Modulul box.c contine functiile win() si closewin() care sunt folosite in celelalte module pentru deschiderea respectiv inchiderea unei ferestre. De asemenea aici se mai gaseste setcrs() pentru stabilirea dimensiunii cursorului si out() care executa iesirea fortata din program in cazul unei memorii RAM insuficiente.

Modulul help.c este compus din functia help() care este apelata la apasarea tastei F1 in meniul principal. Functie de parametrul pe care il primeste este citit din fisierul sertim.hlp textul corespunzator meniului de unde a fost tastat F1.

Modulul box1.c realizeaza functiile din prima fereasra din meniul principal: load, view, new, del si exit. Functia load() citeste din fisierul de date modell.dta numele modelelor ce se gasesc aici si le afiseaza pe ecran pentru a putea fi ales unul dintre acestea. In acest caz este rezervat spatiu de memorie (cu instructia malloc) pentru vectorul parametrilor exacti ai modelului care vor fi cititi din modell.dta. Similar lucreaza cu modell.dta functiile new() si del() pentru adaugarea in fisierul baza de date modell.dta a unui nou model, respectiv pentru a sterge unul vechi. View() realizeaza afisarea parametrilor modelului filtrului utilizat in simulare.

Modulul box2.c realizeaza functiile estim, input si noise din a doua fereastra a meniului principal. Dupa modelul din main.c, functiile estim si input creeaza un meniu secundar (o noua fereastra) de unde se alege cu Enter optiunea dorita: selectarea tipului de estimator dorit respectiv alegerea valorii initiale ale vectorului semnalului haotic U. Foarte importanta este functia puls() apelata peste tot in program pentru citirea unui numar int, long sau float, dupa ce este deschisa o noua fereastra si este afisata vechea valoare a variabilei unde se va face citirea. Functia puls() primeste variabila ca pointer pentru a-i putea modifica valoarea.

Modulul para.c este alcatuit din functia para care impreuna cu tst_p() si dispune_p() realizeaza submeniul para din fereastra 2 a meniului principal. In caz ca este aleasa spre executie optiunea case, controlul il ia functia caz() care afiseaza pe rând intr-o fereastra matricele parametrilor A1...Ana, B1...Bnb. Trecerea de la o matrice la alta se face tastând F3. Daca se tasteaza in schimb Enter cu ajutorul functiei ask() si goo() se trece in invers video câmpul de pe ecran unde este afisat parametrul pe care se afla cursorul si este negata valoarea unei variabile dintr-un sir (par[0][][]) asociata respectivului parametru. Cursorul este reprezentat de faptul ca foreground-ul zonei ecran de sub el clipeste. Functia break este implementata absolut similar utilizând aceeasi functie caz() pentru sirul par[1][][].

Modulele udu.c si cmmp.c se ocupa de fapt de simularea unei serii de timp, estimarea parametrilor acesteia si de afisarea lor pe ecran in mod grafic. Datorita rezolutiei necesare pentru o buna vizualizare a parametrilor s-a luat hotarirea ca programul sa nu poata fi rulat decât pe calculatoare PC ce dispun de interfata SVGA.

2.5.2 FACILITATI IN UTILIZAREA PACHETULUI DE PROGRAME SERTIM 1.0 [Cu93]

Pachetul de programe de simulare a estimarii seriilor de timp SERTIM a fost realizat cu scopul de a aprecia calitatea procedurilor de estimare recursiva CMMP, U-D si SQ. El poate fi utilizat cu modificari minore si in cadrul unei structuri de estimare on-line. El a fost conceput ca sistem de programe interactiv bazat pe principiul ferestrelor. In continuare se vor prezenta câteva dintre facilitatile mai importante oferite utilizatorului.

o *Introducerea modelului matematic al unui filtru in baza de modele*

Operatii necesare:
- selectam optiunea "FILE"
- selectam optiunea "NEW". Pe ecran apare o fereastra pentru introducerea marimilor caracteristice ale modelului.
- in functie de structura si ordinul modelului matematic se introduc in ordine: NY - numarul de iesiri, NA - ordinul autoregresiei, NU - numarul de intrari, NB - ordinul mediei alunecatoare. Dupa introducerea acestor marimi se deschide o noua fereastra destinata introducerii sub forma matriciala a modelului matematic intrare-iesire mono sau multivariabil.
- se introduc parametri. Initial toti parametri sunt nuli. Pentru a selecta parametrul pe care dorim sa-l introducem sau sa-l modificam utilizam "sagetile" pentru lucrul in cadrul matricii curente sau tasta F3 pentru trecerea la o alta matrice a parametrilor. Dupa selectarea parametrului se tasteaza "ENTER", iar pe ecran va apare o fereastra destinata introducerii valorii

numerice a parametrului.
- se introduce valoarea parametrului selectat, dupa care se tasteaza "ENTER". In acest moment pe ecran se va realiza modificarea acestui parametru. Pentru introducerea unui alt parametru se repeta pasul anterior, precum si cel curent.
- dupa ce s-au realizat toate modificarile, odata cu parasirea ultimei matrici de parametrii, pe ecran se deschide o noua fereastra ce are inscris mesajul "SAVE ? (Y/N)". In cazul in care dorim salvarea modelului matematic introdus tastam Y, iar in caz contrar N.
- daca s-a optat pentru salvare, operatorul este solicitat pentru introducerea numelui modelului. Acest nume poate avea maxim 8 caractere.

Modelele matematice sunt salvate automat in fisierul modell.dta ce intra in componenta pachetului de programe SERTIM. In acest fisier pot fi salvate maxim 64 de modele, numar ce a fost prevazut din necesitatea de a incape intr-o singura fereastra, fara a se realiza "scroll".

Dupa introducerea modelului in baza de modele, acesta poate fi vizualizat utilizând optiunea "View", poate fi incarcat ca model activ cu optiunea "Load", respectiv poate fi sters cu optiunea "Del".

○ *Stergerea modelului matematic al unui proces din baza de modele*

Operatii necesare:
- selectam optiunea "FILE"
- selectam optiunea "Del". Pe ecran se deschide o fereastra ce contine toate modelele matematice aflate in baza de modele (fisierul modell.dta).
- este selectat cu ajutorul sagetilor modelul care va fi sters si se tasteaza "Enter".
- pentru a nu se realiza o stergere involuntara apare pe monitor o protectie caracterizata prin mesajul "Delete ? (Y/N)". In cazul in care stergerea modelului este dorita de utilizator se tasteaza Y, iar in caz contrar N.

○ *Selectarea unui model din baza de modele*

Operatii necesare:
- selectam optiunea "Load". Pe ecran se deschide o fereastra ce contine toate modelele matematice aflate in baza de modele (fisierul modell.dta).
- este selectat cu ajutorul sagetilor modelul care va fi declarat activ si se tasteaza "Enter".

○ *Alegerea unor marimi caracteristice estimarii*

■ *Alegerea raportului semnal-zgomot*

Operatii necesare:

- selectam optiunea "Noise". Printr-o fereastra utilizatorul este informat cu privire la valoarea curenta a raportului semnal-zgomot. Daca utilizatorul nu doreste sa modifice aceasta valoare tasteaza "ESC", iar daca nu, introduce valoarea dorita. Pot fi introduse numere supraunitare ceea ce semnifica un semnal de putere superioara zgomotului, sau o valoare subunitara ceea ce semnifica un zgomot de putere superioara semnalului util (semnalul haotic).

■ *Initializarea matricii parametrilor* θ

Operatii necesare:
- selactam optiunea "Init θ". Daca pe ecran apare un semn in dreptul acestei optiuni inseamna ca matricea initiala a parametrilor este chiar matricea parametrilor utilizata in simulare, adica cea din modelul matematic. In caz contrar matricea parametrilor este matricea nula. Selectarea sau deselectarea se face cu "ENTER".

■ *Alegerea lui* α

Alfa este un parametru utilizat in initializarea matricii P. $P=\alpha I$, unde I este matricea unitate.
Operatii necesare:
- selectam optiunea "Alfa". Printr-o fereastra utilizatorul este informat cu privire la valoarea curenta a parametrului α. Daca utilizatorul nu doreste sa modifice aceasta valoare tasteaza "ESC", iar daca nu, introduce valoarea dorita.

■ *Alegerea lui* λ

λ este cunoscut in literatura de specialitate ca "factor de uitare" si are valori cuprinse intre 0.98 si 1.
- selectam optiunea "Lambda". Printr-o fereastra utilizatorul este informat cu privire la valoarea curenta a parametrului λ. Daca utilizatorul nu doreste sa modifice aceasta valoare tasteaza "ESC", iar daca nu, introduce valoarea dorita.

○ *Alegerea valorilor initiale ale intrarii*

Modelele matematice prezente in baza de modele si utilizate in simulare necesita pentru calculul recursiv, initializarea cu valori corespunzatoare momentelor anterioare celui din care incepe simularea. Aceasta initializare trebuie sa cuprinda atât intrarea cât si iesirile. Pentru iesiri s-a adoptat o ipoteza simplificatoare, considerându-se ca având valori initiale nule. In schimb pentru valorile initiale ale intrarii (semnalul haotic) se poate alege intre: a) valori initiale alese de utilizator; b) valori initiale nule; c) valori initiale aleatoare.

Pentru aceasta este necesare selectarea optiunii "Init", dupa care se va selecta din fereastra ce se deschide una din optiunile a), b) sau c).

○ *Alegerea tipului de estimator*

Operatii necesare:
- selectam optiunea "Estim".
- selectam unul dintre estimatoarele propuse (U-D, SQ sau CMMP). Selectarea se face tastând "ENTER".

○ *Facilitati de reprezentare grafica a parametrilor*

■ *Alegerea parametrilor ce vor fi reprezentati pe ecran*

Implicit, pe ecran vor apare toti parametrii. Daca dorim sa dispunem de graficul de variatie a unui numar redus de parametrii este necesara selectia optiunii "Param". Din fereastra ce se deschide se va selecta optiunea "none" care invalideaza optiunea "all", iar mai apoi, utilizând optiunea "case" se vor selecta parametrii doriti. Aceasta selectie se face tastând "ENTER", parametrul selectat trecând in invers-video.

In cazul in care se doreste vizualizarea unui singur parametru pe ecran va apare in coltul din dreapta-sus valoarea parametrului in acel moment.

■ *Alegerea numarului de estimatii succesive ce va fi reprezentat grafic*

Se utilizeaza optiunea "Zoom x", iar in fereastra ce se deschide vom introduce numarul de estimatii ce va fi reprezentat grafic.

■ *Alegerea factorului de scala pentru axa ordonatelor*

Se utilizeaza optiunea "Zoom y", iar in fereastra ce se deschide vom introduce o valoare cuprinsa intre 0 si 230. Orice valoare introdusa mai mare decât valoarea maxima se va considera 230, si orice valoare mai mica de 1 se va considera 0.

■ *Alegerea punctului de pe axa ordonatelor prin care va trece axa mediana orizontala a ecranului*

In mod analog, se va selecta optiunea "Abscisa", iar in fereastra deschisa cu acest prilej va fi introdusa valoarea dorita.

■ *Alegerea modului de trasare al graficului*

Pachetul de programe SERTIM permite trasarea graficelor prin puncte, corespunzatoare diferitelor estimatii, si respectiv prin linii, linii ce unesc aceste puncte. Selectarea modului de trasare al graficului se face prin optiunea "Lines", utilizând tasta "ENTER".

■ *Modul de lucru "pas cu pas"*

In cazul acestui mod de lucru, estimarea, deci si reprezentarea grafica se va face numai pentru un pas de estimare, dupa care este necesara apasarea tastei "ENTER" pentru trecerea la pasul urmator.

■ *Obtinerea graficului la imprimanta*

Pentru obtinerea graficului la imprimanta trebuiesc parcurse urmatoarele etape:
- premergator lansarii in executie a programului sertim.exe vom lansa in executie graphics.com din DOS5 sau DOS6. Exista astfel necesitatea realizarii unui fisier batch care sa contina programele graphics.com si sertim.exe.
- lansam in executie programul sertim.exe.
- incarcam din baza de modele, modelul dorit.
- alegem tipul de estimator, parametrii estimarii, precum si modul de reprezentare grafica.
- lansam estimarea tastând F3 sau selectând optiunea "Go".
- tastam "Print Screen".

2.5.3 REZULTATE EXPERIMENTALE

Particularizarile si recomandarile evidentiate in cadrul algoritmilor de estimare on-line considerati au fost analizate in cazul unei aplicatii specifice, utilizand pentru filtru un model ARMA de ordinul 4:

$$y(t) = 0.1y(t-1) - 0.2y(t-2) + 0.3y(t-3) - 0.1u(t-1) + 0.23u(t-2) - 0.4u(t-3) \quad \textbf{(2.55)}$$

Programul de calcul implementând algoritmii de estimare descrisi, a fost rulat pentru diverse situatii posibile legate de modelul matematic anterior referit in scopul evidentierii unor concluzii privind calitatea estimatorilor descrisi (corectitudinea estimarii, viteza de calcul, convergenta si stabilitatea calculelor, etc.). S-au avut in vedere urmatoarele aspecte:
- influenta raportului dintre semnalul util (semnalul de intrare haotic) si semnalul perturbator, deci zgomotul;
- influenta factorului de uitare λ, reprezentând dependenta de valorile anterioare;
- influenta valorii de initializare a matricii de covarianta α.

Aceste aspecte au fost grupate in cazuri de studiu specifice, iar rezultatele din cadrul fiecarei grupe au fost evaluate cu ajutorul graficelor de variatie in timp ale parametrilor estimati

(figurile 2.19 - 2.42), rezultând urmatorul tabel.

TIP ESTIMATOR	RAPORT SEMNAL / ZGOMOT	VALOARE α	VALOARE λ	NUMAR DE PASI
U-D	Noise=∞	α=10	λ=1	p=176
	Noise=∞	α=1000	λ=1	p=14
	Noise=∞	α=10	λ=0.97	p=151
	Noise=∞	α=1000	λ=0.97	p=12
	Noise=1000	α=10	λ=1	p=153
	Noise=1000	α=1000	λ=1	p=12
	Noise=1000	α=10	λ=0.97	p=72
	Noise=1000	α=1000	λ=0.97	p=10
SQ	Noise=∞	α=10	λ=1	p=198
	Noise=∞	α=1000	λ=1	p=24
	Noise=∞	α=10	λ=0.97	p=67
	Noise=∞	α=1000	λ=0.97	p=13
	Noise=1000	α=10	λ=1	p=270
	Noise=1000	α=1000	λ=1	p=25
	Noise=1000	α=10	λ=0.97	p=88
	Noise=1000	α=1000	λ=0.97	p=23
CMMP	Noise=∞	α=10	λ=1	p=170
	Noise=∞	α=1000	λ=1	p=12
	Noise=∞	α=10	λ=0.97	p=121
	Noise=∞	α=1000	λ=0.97	p=10
	Noise=1000	α=10	λ=1	p=180
	Noise=1000	α=1000	λ=1	p=20
	Noise=1000	α=10	λ=0.97	p=125
	Noise=1000	α=1000	λ=0.97	p=12

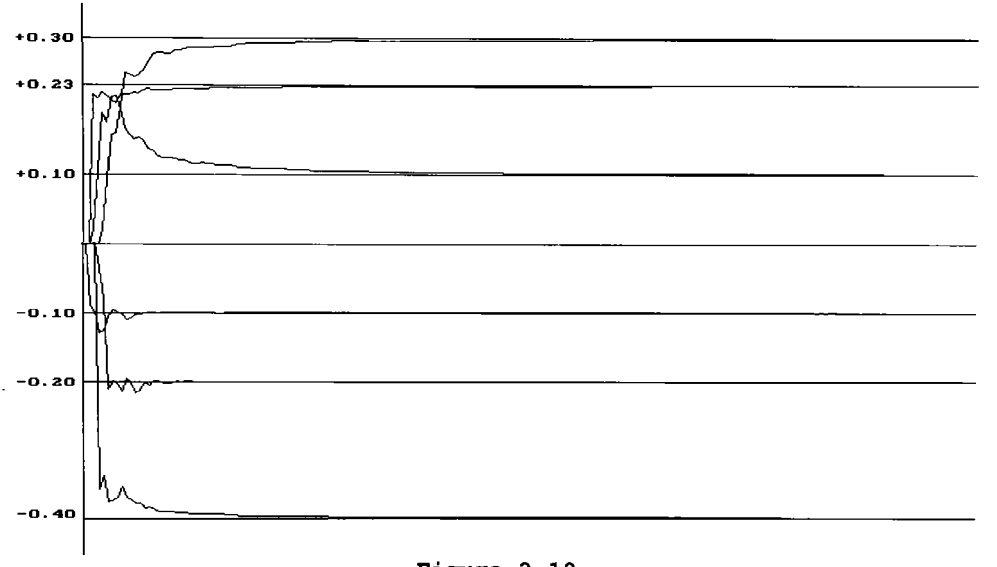


Figura 2.19

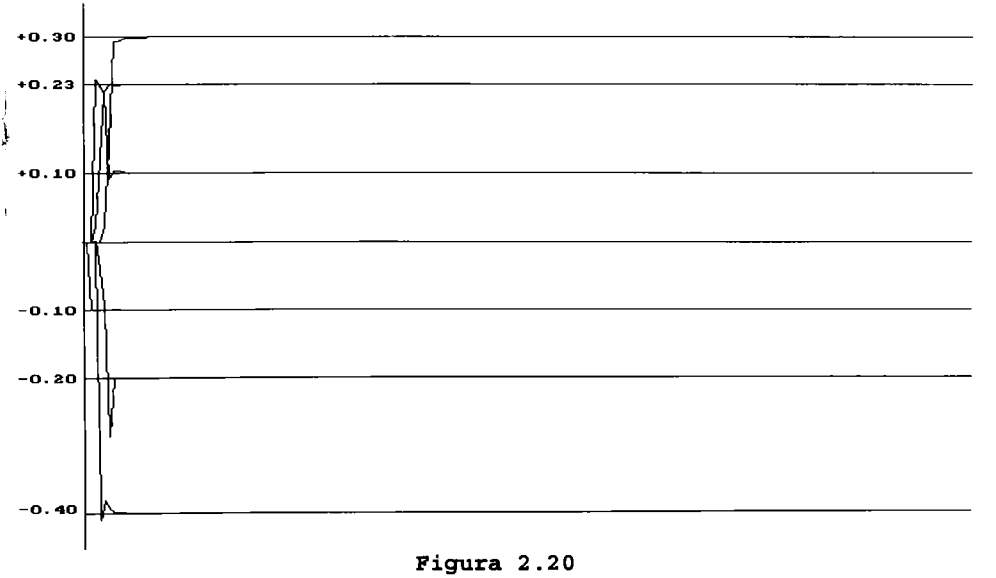


Figura 2.20

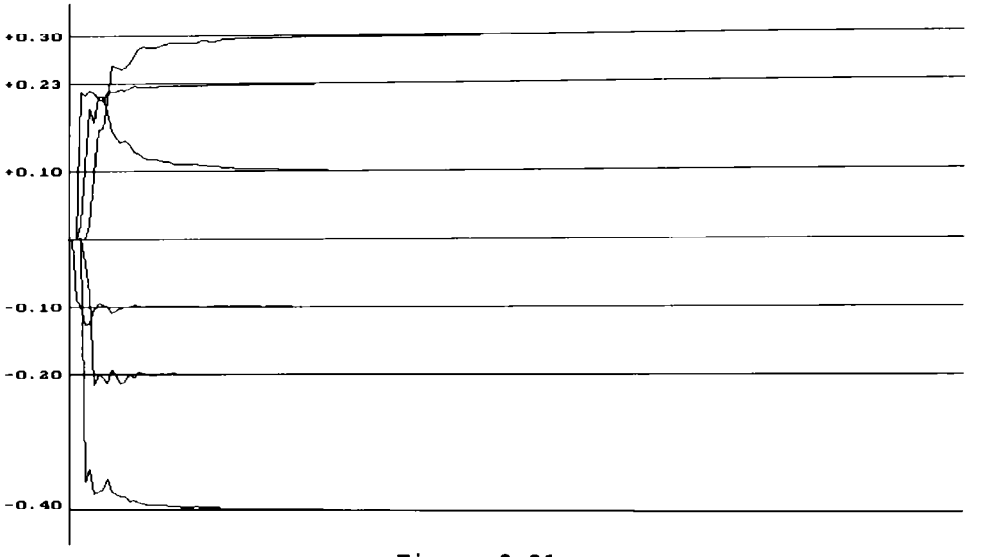


Figura 2.21

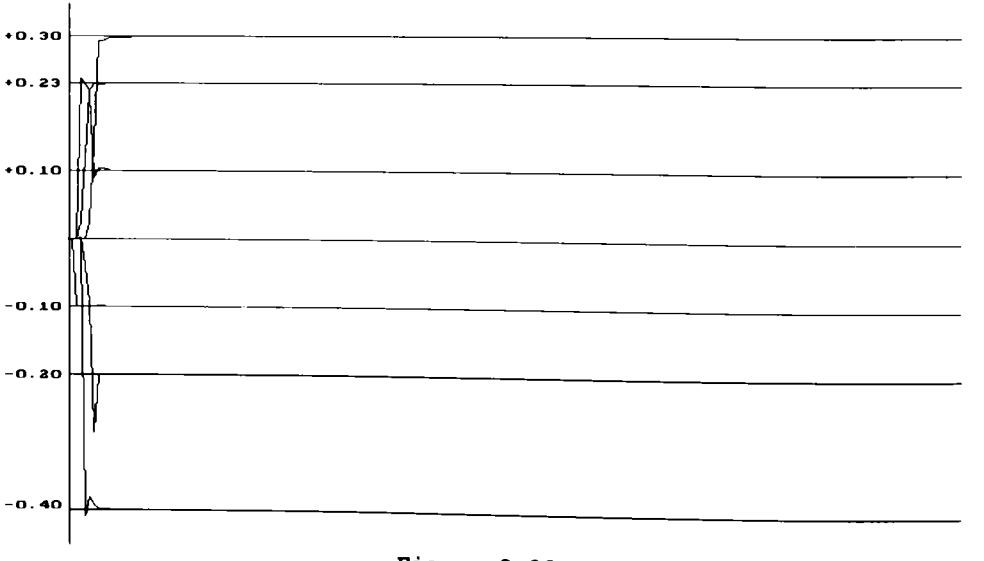


Figura 2.22

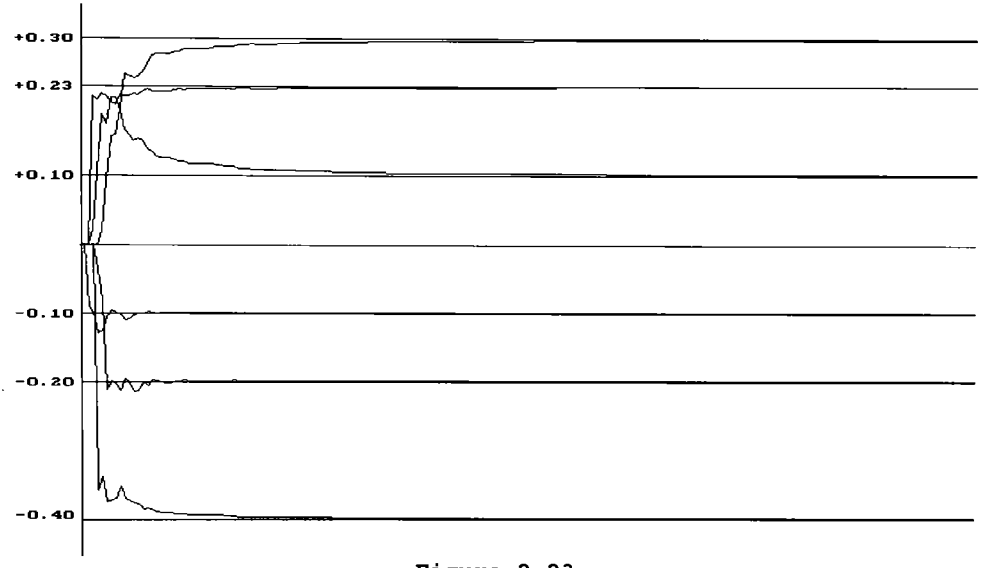


Figura 2.23

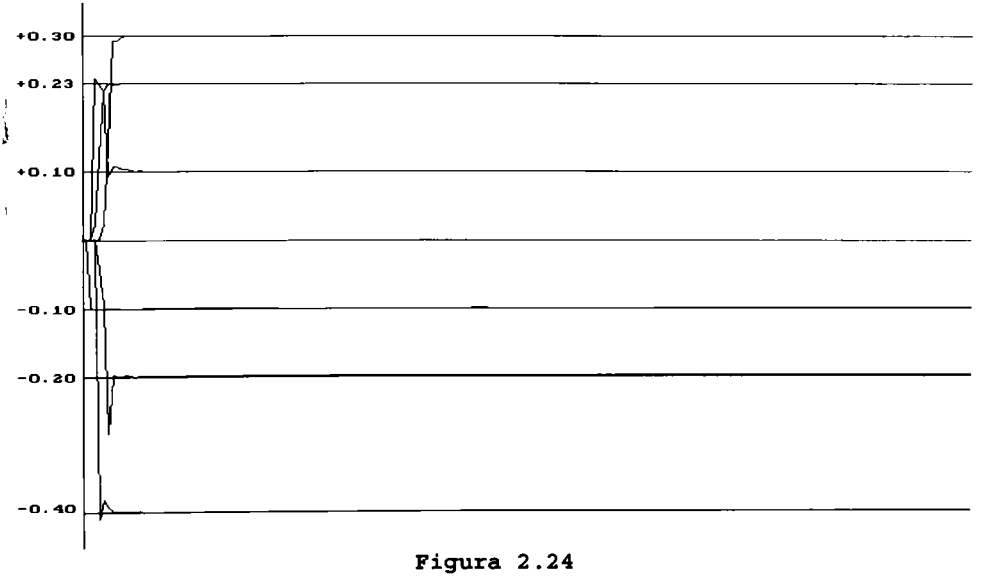


Figura 2.24

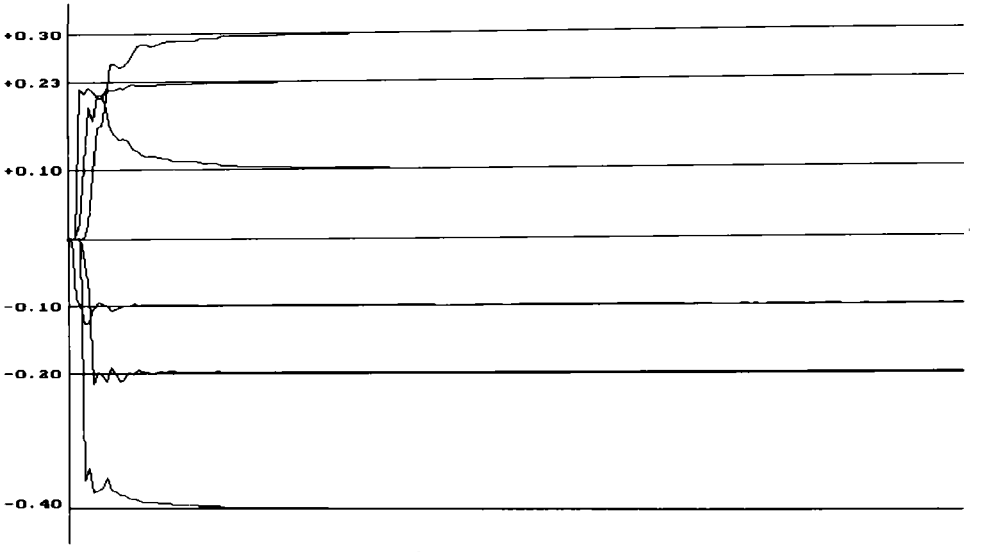


Figura 2.25

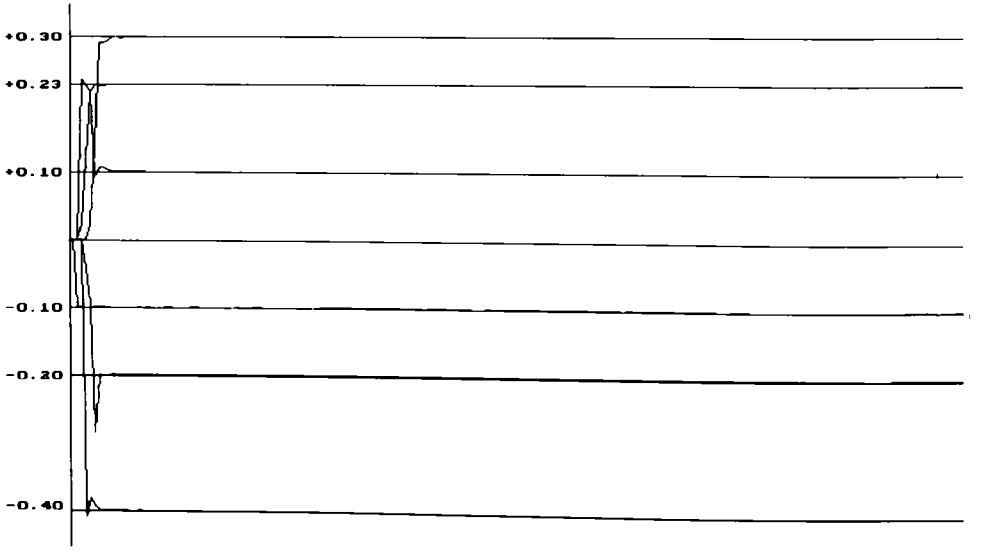


Figura 2.26

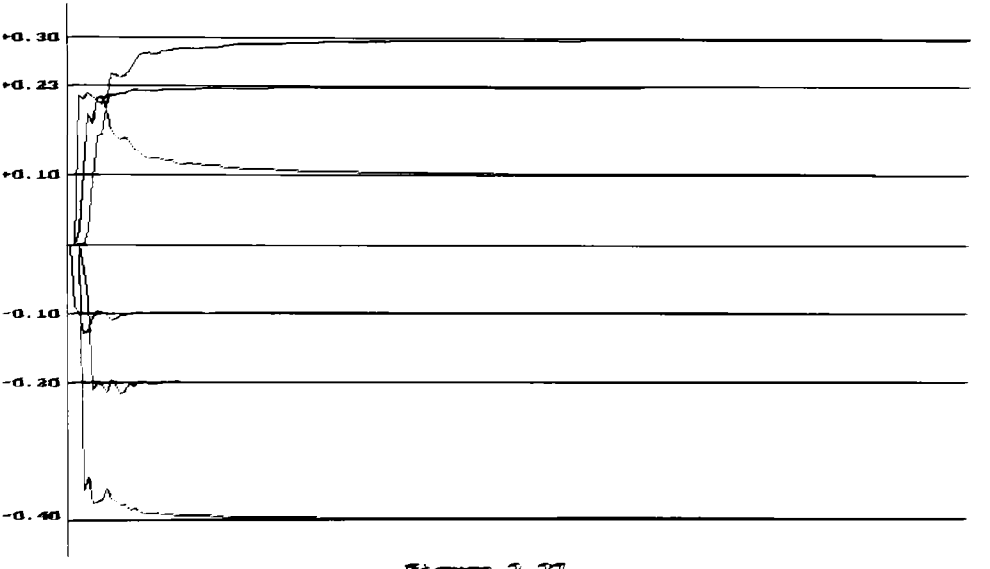


Figura 2.27

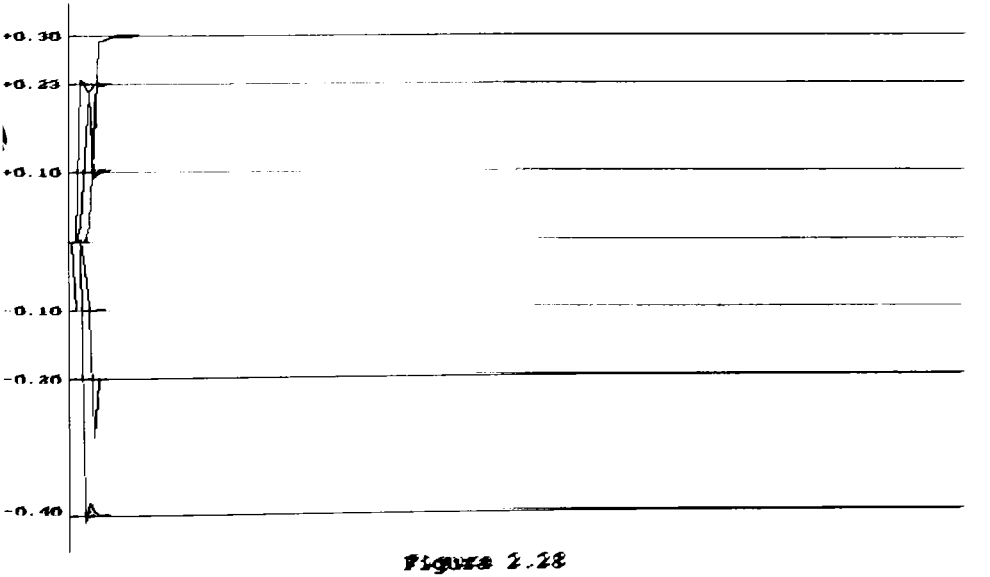


Figura 2.28

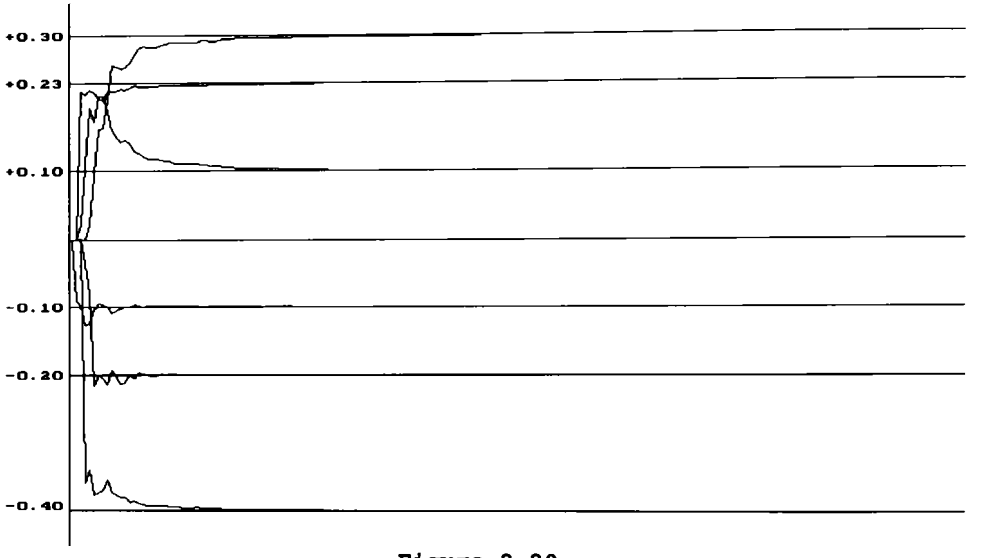


Figura 2.29

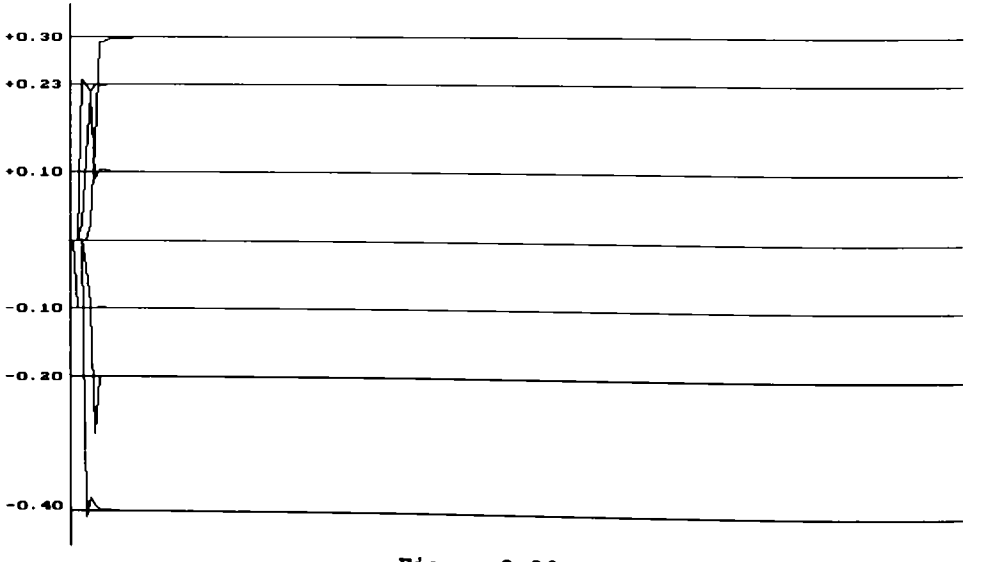


Figura 2.30

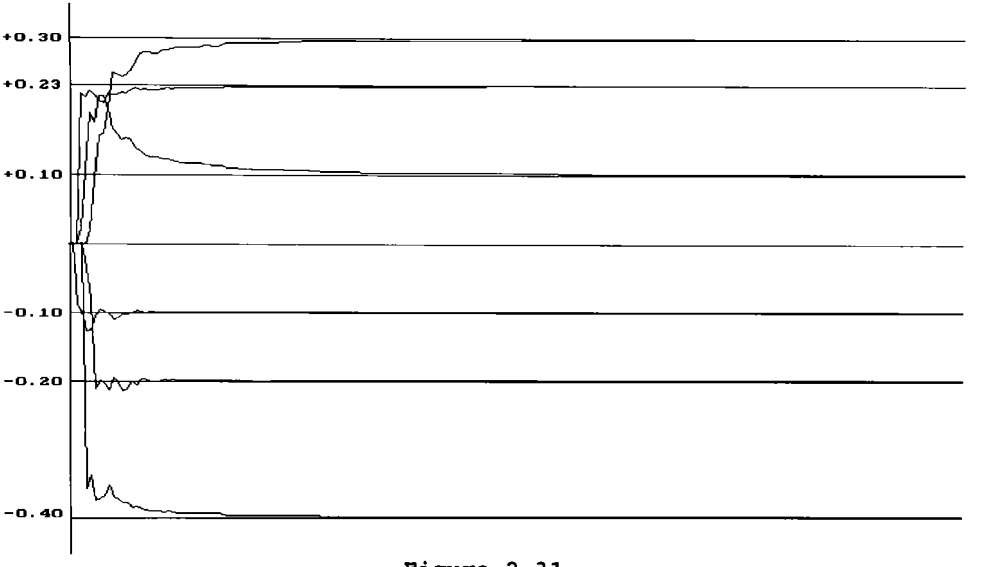


Figura 2.31

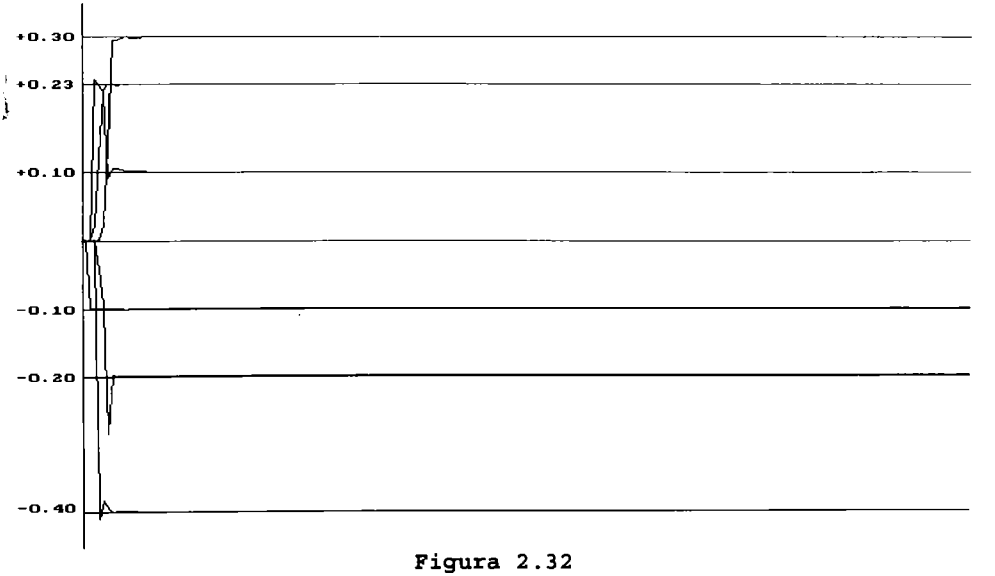


Figura 2.32

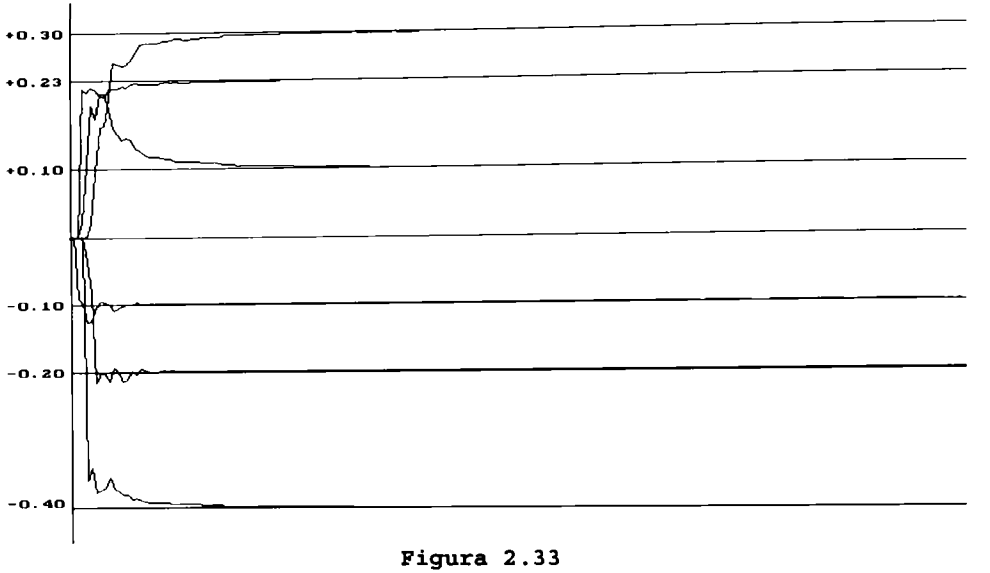


Figura 2.33

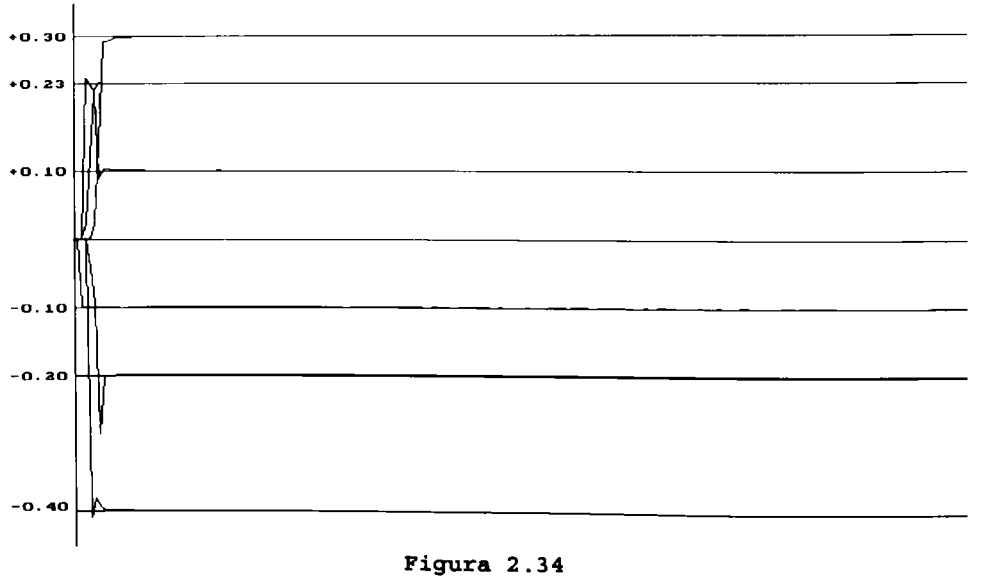


Figura 2.34

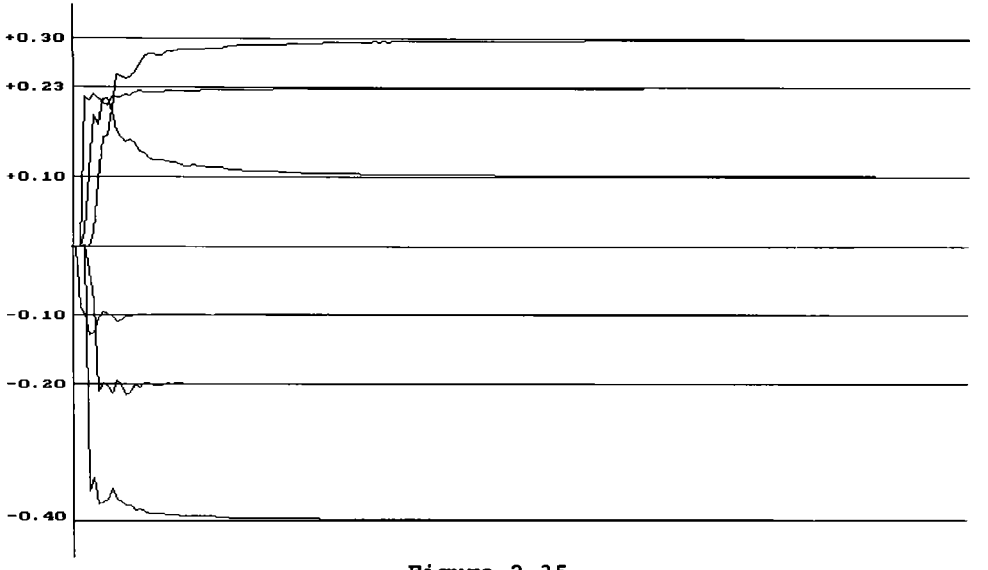


Figura 2.35

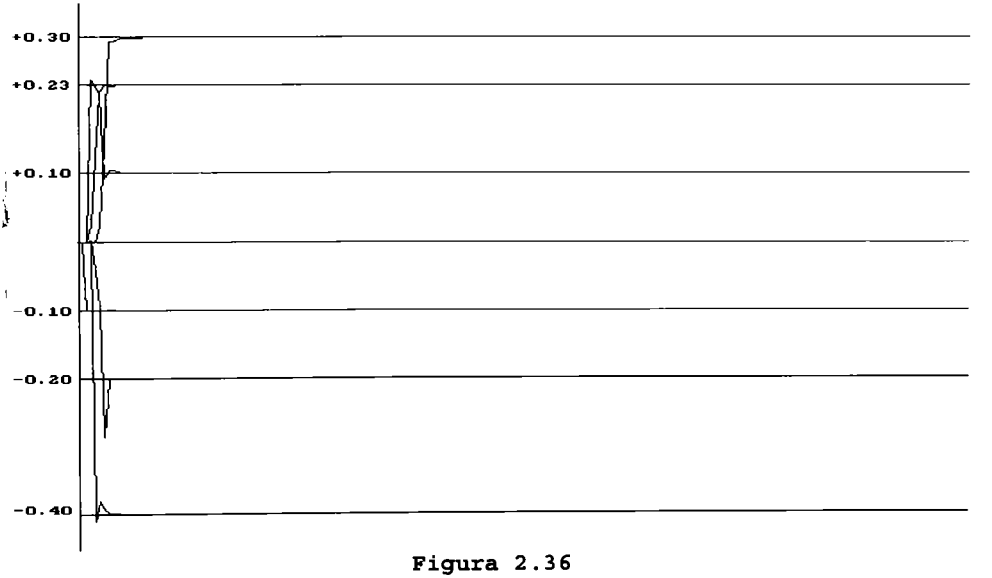


Figura 2.36

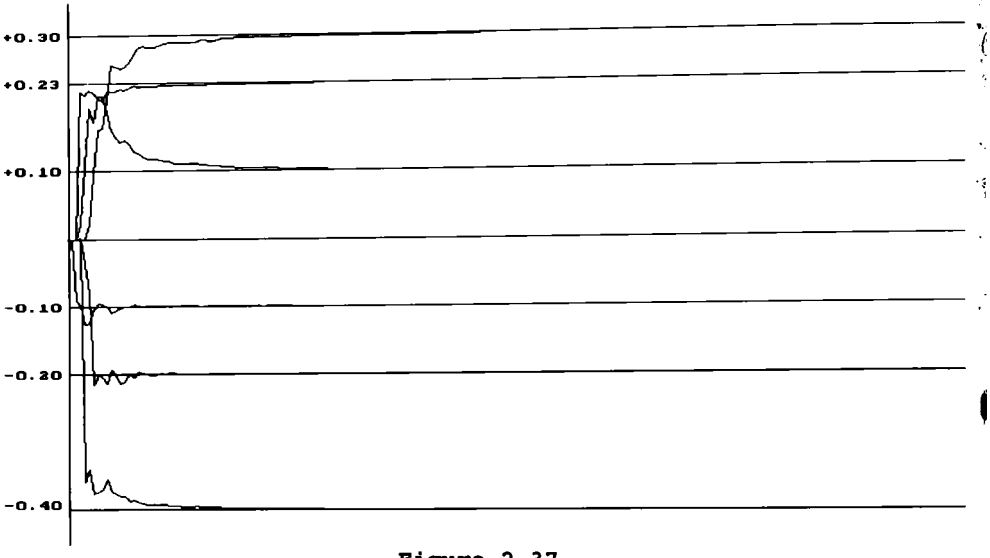


Figura 2.37

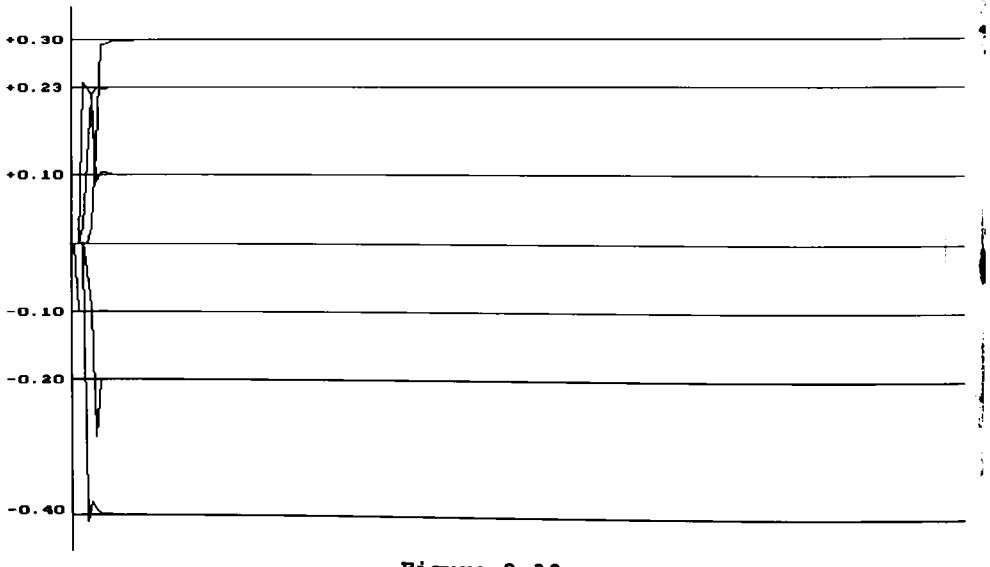


Figura 2.38

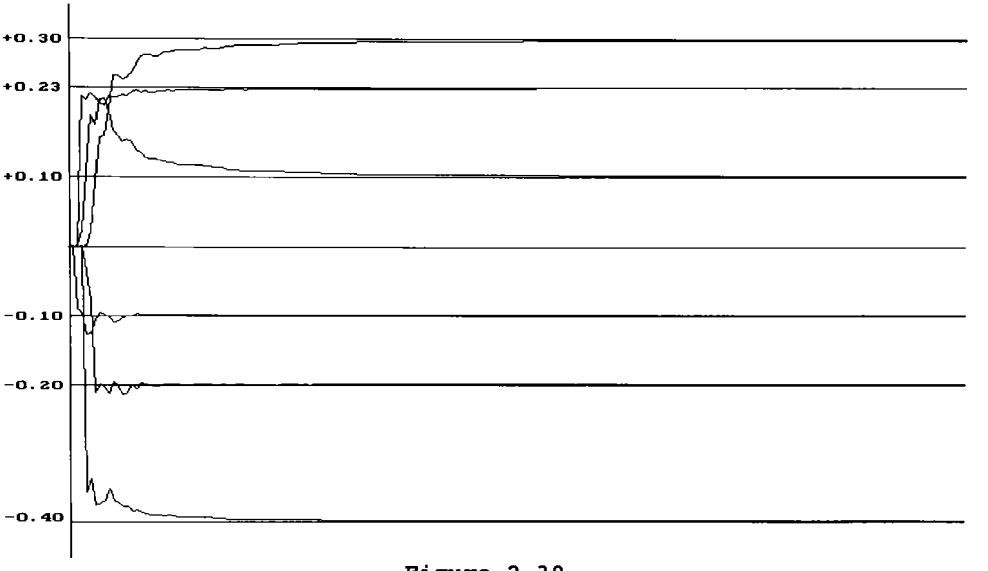


Figura 2.39

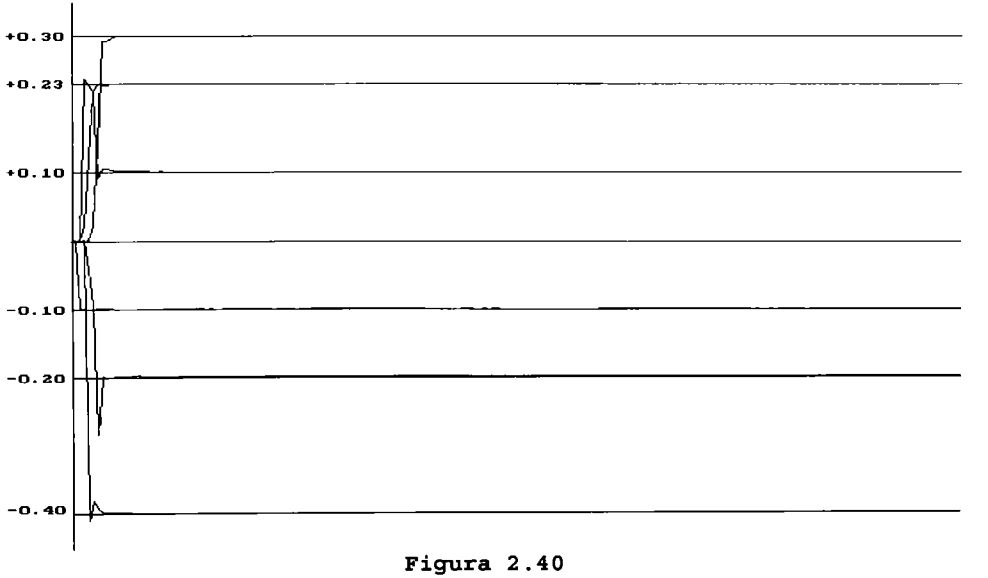


Figura 2.40

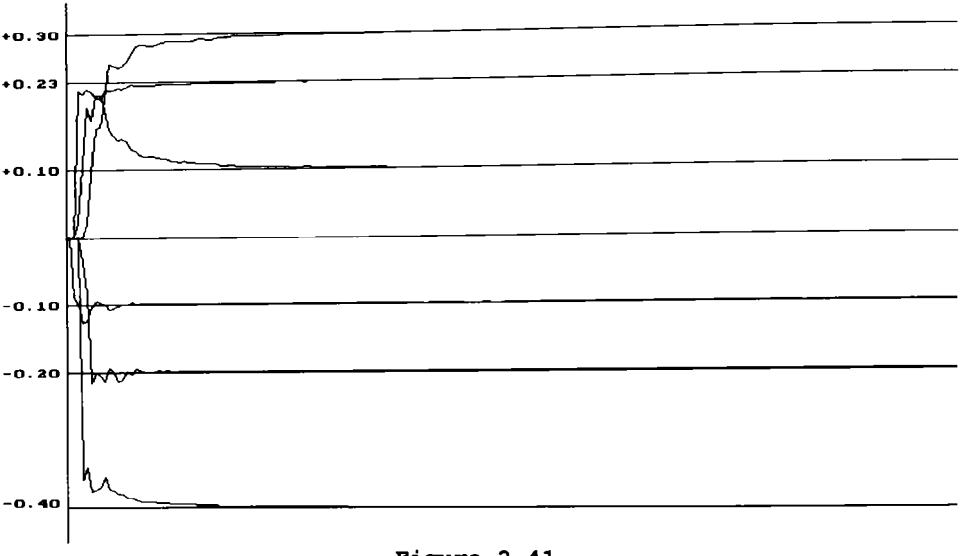


Figura 2.41

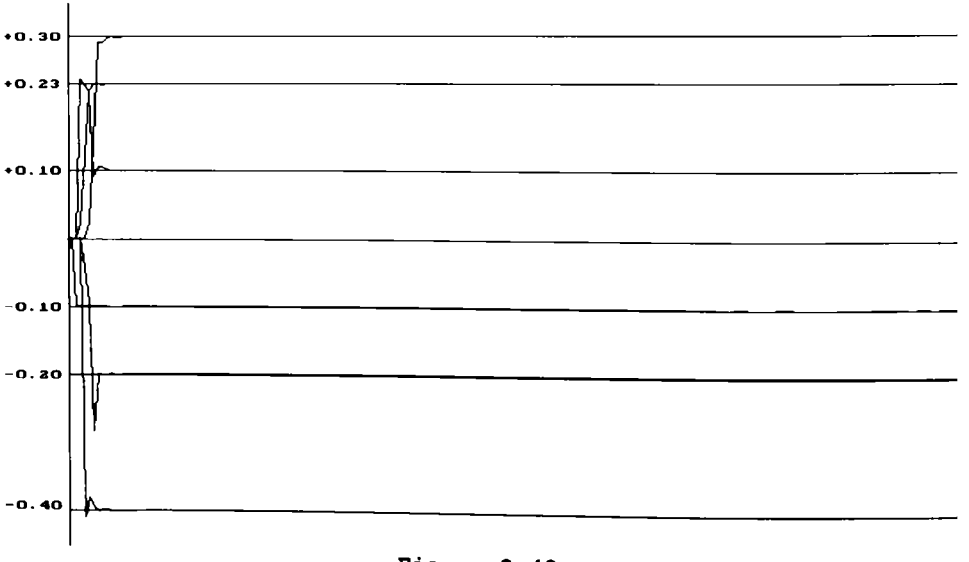


Figura 2.42

ANEXA 2.1: PROGRAME DE SIMULARE A SISTEMELOR DINAMICE HAOTICE

A. SIMULAREA SISTEMULUI LORENZ

In vederea simularii sistemului haotic continual Lorenz au fost scrise fisierele lorenz.m si lorenzeq.m in mediul MATHLAB with SIMULINK ce opereaza sub Windows 3.1.

* *fisierul lorenz.m*

```
%-----------------------------------
%SIMULAREA SISTEMULUI HAOTIC LORENZ
%-----------------------------------

clf
clc
echo on

% Rezolvarea sistemului de ecuatii diferentiale ce
% descriu atractorul Lorenz. Ecuatiile sunt definite in
% fisierul lorenzeq.m

type lorenzeq

% Valorile parametrilor globali sunt:

global SIGMA RHO BETA
SIGMA = 10.;
RHO = 28.;
BETA = 8./3.;
axis([10 40 -20 20 -20 20])
view(3)
hold on
title(' ')
clc

% Conditiile initiale

y0 = [0 0 eps];

% Se defineste orizontul de timp intre 0 si tfinal
```

```
tfinal = 100;
y = ode23p('lorenzeq',0,tfinal,y0)
echo off
```

* *fisierul lorenzeq.m*

```
%-------------------------------------
% descrierea sistemului haotic Lorenz
%-------------------------------------

function ydot = lorenzeq(t,y)

global SIGMA RHO BETA

% declararea matricii sistemului Lorenz

A = [ -BETA     0      y(2)
         0   -SIGMA    SIGMA
       -y(2)    RHO     -1   ];

ydot = A*y;
```

B. SIMULAREA SISTEMULUI RÖSSLER

In vederea simularii sistemului haotic continual Rössler au fost scrise fisierele rossler.m si rosslerq.m in mediul MATHLAB with SIMULINK ce opereaza sub Windows 3.1.

* *fisierul rossler.m*

```
%-------------------------------------
%SIMULAREA SISTEMULUI HAOTIC ROSSLER
%-------------------------------------

clf
clc
echo on

% Rezolvarea sistemului de ecuatii diferentiale ce
% descriu atractorul Rossler. Ecuatiile sunt definite in
% fisierul rosslerq.m

type rosslerq
```

```
% Valorile parametrilor globali sunt:

global a b c
a = 0.2;
b = 0.2;
c = 5.7;
axis ([-10 10 -10 10 -10 10])
view(3)
hold on
title(' ')
clc

% conditiile initiale

y0 = [eps 0 eps];

% Se defineste orizontul de timp intre 0 si tfinal

tfinal =100;
y = ode23p('rosslerq',0,tfinal,y0)
echo off
```

* *fisierul rosslerq.m*

```
%-------------------------------------
% descrierea sistemului haotic Rossler
%-------------------------------------

function ydot = rosslerq(t,y)

global a b c

% declararea matricii sistemului Lorenz

A = [    0     -1      -1
         1      a       0
         y(3)    0      -c  ];

ydot = A*y+[0;0;b];
```

C. SIMULAREA SISTEMULUI HENON

In vederea simularii sistemului haotic discret Henon a fost scris fisierul henon.m in mediul MATHLAB with SIMULINK ce opereaza sub Windows 3.1.

```
%---------------------------------
%SIMULAREA SISTEMULUI HAOTIC HENON
%---------------------------------

%conditiile initiale

x(1)=0.0;
y(1)=0.00001;

for i=2:500
 x(i)=y(i-1)+1-1.4*x(i-1)*x(i-1);
 y(i)=0.3*x(i-1);
end

% reprezentarea grafica

plot(x,'w'),grid
plot(y,'w'),grid
plot(x,y,'w'),grid
end
```

ANEXA 2.2 PROGRAMELE SURSA IN LIMBAJUL C PENTRU PACHETUL SERTIM 1.0

```
/*----------------------------------------
------------ fisierul box.c -------------
----------------------------------------*/

#include<alloc.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>

extern unsigned vb;
extern struct winn
     {    int r,l,b,t,a; void *p;  } rr;

void setcrs(int su,int sd)
{
union REGS r,rr;
     r.h.ah=1; r.h.cl=su; r.h.ch=sd;
     int86(0x10,&r,&rr);
}

void out(void)
{
     window(1,1,80,24); clrscr(); setcrs(8,7);
     puts("Out of RAMM"); exit(0);
}

void fill(int x,int y,int dx,int dy,int v)
{
int i,j;
     x=x-1;dx=dx-1; y=y-1;dy=dy-1;
     for(j=y;j<=dy;j++) for(i=x;i<=dx;i++) pokeb(vb,160*j+i+i,v);
}

void frame(int x,int y,int dx,int dy)
{
int i,j;
     x=x-1;dx=dx-1; y=y-1;dy=dy-1;
     for(j=y;j<=dy;j++)
     { pokeb(vb,160*j+x*2,179); pokeb(vb,160*j+2*dx,179);}
     for(i=x;i<=dx;i++)
     { pokeb(vb,160*y+i+i,196); pokeb(vb,160*dy+i+i,196); }
     pokeb(vb,160*y+x*2,218);  pokeb(vb,160*dy+2*x,192);
     pokeb(vb,160*y+2*dx,191); pokeb(vb,160*dy+2*dx,217);
}

win(int le,int top,int rg,int bt,int at)
```

```
{
int size,sx,sy;
struct winn *p1;
     textattr(at); size=(rg+2)*(bt+2)*2;
     if((p1=malloc(sizeof(rr)))==NULL) out();
     if((p1->p=malloc(size))==NULL) out();
     p1->l=le; p1->r=rg;
     p1->t=top;  p1->b=bt; p1->a=at;
     gettext(p1->l,p1->t,p1->r+p1->l+1,p1->b+p1->t+1,p1->p);
     window(le,top,le+rg+1,top+bt+1);
     clrscr(); setcrs(0,1);
     frame(le,top,le+rg+1,top+bt+1);
     window(le+1,top+1,le+rg,top+bt);
     return p1;
}

closewin(struct winn *p1)
{
     window(1,1,80,25);
     puttext(p1->l,p1->t,p1->r+p1->l+1,p1->b+p1->t+1,p1->p);
     free(p1->p); free(p1);
}


/*-------------------------------------------
  ------------- fisierul box1.c -------------
  -------------------------------------------*/

#include <stdio.h>
#include <conio.h>
#include <io.h>
#include <fcntl.h>
#define c_r 77
#define c_l 75
#define c_u 72
#define c_d 80

extern struct winn
     {    int r,l,b,t,a; void *p;  } rr;
struct rr *l1,*l2;
extern int alb,negr,par[3][8][100],este;
extern int ok_all,ok_no,ok_ca;
extern float *b,*bb,*uini;
int n,ny,nu,na,nb;

/*************** view ********************/
void view(void)
{
int i,j,naa=0; char ch;
     l1=win(8,5,66,15,negr);
     for(naa=0;naa<na;naa++)
```

```
     {
     gotoxy(5,2);
     cprintf("Current A%d matrix :",naa+1);
     for(j=0;j<ny;j++)
          {  gotoxy(5,4+j);
             for(i=0;i<ny;i++)
             cprintf("%+3.2f  ",*(b+n*j+i+naa*ny));
           }
     if((ch=getch())=='\0') getch();
     if(ch==27) { closewin(11); return; }
     clrscr();
     }
     for(naa=0;naa<nb;naa++)
     {
     gotoxy(5,2);
     cprintf("Current B%d matrix :",naa+1);
     for(j=0;j<ny;j++)
          { gotoxy(5,4+j);
           for(i=0;i<nu;i++)
           cprintf("%+3.2f  ",*(b+n*j+i+naa*nu+na*ny));
          }
     if((ch=getch())=='\0') getch();
     if(ch==27) { closewin(11); return; }
     clrscr();
     }
     closewin(11);
}

/***************** load ***********************/

int by_lx,by_ly,ctx,cty,fis;
long int sk[54];

void lod(void)
{
int i,j;
este=0;
free(b);
free(bb);
free(uini);
n=0; ny=0; nu=0; na=0; nb=0;
ok_all=251; ok_no=32; ok_ca=32;
for(i=0;i<5;i++)  for(j=0;j<100;j++)
{ par[0][i][j]=1; par[1][i][j]=0; }
lseek(fis,sk[by_ly*6+by_lx],SEEK_SET);
read(fis,&n,2);
read(fis,&ny,2);
read(fis,&na,2);
read(fis,&nu,2);
read(fis,&nb,2);
if((b=malloc(ny*n*4))==NULL) out();
```

```
if((bb=malloc(ny*n*4))==NULL) out();
if((uini=malloc(nu*nb*4))==NULL) out();
read(fis,b,ny*n*4);
}

void cit(void)
{ int i,j,a1,a2; char fs[9];
for(i=0;i<54;i++)
 {
  if(eof(fis)) { if(i) {ctx=i%6; cty=i/6; }
  return;
 }
read(fis,fs,9);
cprintf(" %s  ",fs);
sk[i]=tell(fis);
read(fis,&a1,2);
read(fis,&a2,2);
j=a1*a2*4+6;
lseek(fis,j,SEEK_CUR);
}
ctx=6; cty=9;
}

void nui(char *bu)
{     l2=win(20,9,40,8,negr);
      cputs(bu);
      gotoxy(30,7);
      cputs("press Esc");
      while(getch()!=27);
      closewin(l2);
}

void tst_l(void)
{   int i=5,j; j=cty;
      if(by_lx<0) by_lx=i; else if(by_lx>i) by_lx=0;
      if(by_ly<0) by_ly=j; else if(by_ly>j) by_ly=0;
/*    if((by_ly==cty)&&(by_lx>ctx-1)) by_lx=ctx-1;*/
      if((by_lx>ctx-1)&&(by_ly==cty)) by_ly=cty-1;
      inv2(l1->l+1+by_lx*11,l1->t+1+by_ly,10);
}

int scan(void)
{
      if(access("model.dta",0))
      { nui("\n\n   Data file model.dta not find "); return 1; }
      fis=open("model.dta",O_RDWR | O_BINARY);
      l1=win(7,6,66,9,negr);
      by_lx=0; by_ly=0; ctx=-1; cty=-1; cit();
      if(ctx!=-1) inv2(l1->l+1,l1->t+1,10);
      else { cputs(" --- Database empty ---"); getch();
             closewin(l1); close(fis); return 1; }
```

```
      while(1)
      {
       switch(getch())
       {
       case     27 : closewin(l1); close(fis); return 1;
       case   '\r': return 0;
       case   '\0': switch(getch())
        {
        case c_u: inv2(l1->l+1+by_lx*11,l1->t+1+by_ly,10);
                  by_ly--; tst_l(); break;
        case c_d: inv2(l1->l+1+by_lx*11,l1->t+1+by_ly,10);
                  by_ly++; tst_l(); break;
        case c_r: inv2(l1->l+1+by_lx*11,l1->t+1+by_ly,10);
                  by_lx++; tst_l(); break;
        case c_l: inv2(l1->l+1+by_lx*11,l1->t+1+by_ly,10);
                  by_lx--; tst_l(); break;
        }
       }
      }
}

void load(void)
{
 if(scan()) return;
 closewin(l1);
 lod();
 close(fis);
}

void del(void)
{   int i,j; long int sz; char ch=0,*bh;
     if(scan()) return;
     l2=win(58,16,20,1,alb);
     setcrs(8,7);
     cputs("  Delete ? (Y/N) ");
     do ch=toupper(getch());
     while((ch!='N')&&(ch!='Y'));
     closewin(l2);
     closewin(l1);
     setcrs(0,1);
     if(ch=='N') { close(fis); return; }
     sz=filelength(fis);
     lseek(fis,sk[by_ly*6+by_lx],SEEK_SET);
     read(fis,&i,2);
     read(fis,&j,2);
     i=i*j*4;
     j=sz-sk[by_ly*6+by_lx]-i-10;
     sz-=i+19;
     lseek(fis,6+i,SEEK_CUR);
     if(!eof(fis))
          {
```

```
            if((bh=malloc(j))==NULL) out();
            read(fis,bh,j);
            lseek(fis,sk[by_ly*6+by_lx]-9,SEEK_SET);
            write(fis,bh,j);
            }
      i=chsize(fis,sz);
      close(fis);
}

void new(void)
{
      int mn,mny,mnu,mna,mnb,i,j,f;
      long int klp;
      char *zz,ch=1,*kr,gs[9];
      l1=win(20,7,25,8,alb);
      setcrs(8,7);
      mn=n; mny=ny; mnu=nu; mna=na; mnb=nb;
      cputs("  New modell :");
      gotoxy(11,3); cputs("NY :"); cscanf("%d",&ny);
      gotoxy(11,4); cputs("NA :"); cscanf("%d",&na);
      gotoxy(11,5); cputs("NU :"); cscanf("%d",&nu);
      gotoxy(11,6); cputs("NB :"); cscanf("%d",&nb);
      n=ny*na+nu*nb;
      if(getch()!='\r')
            {
            n=mn; ny=mny; nu=mnu; na=mna; nb=mnb;
            closewin(l1);
            return;
            }
if(!este)
{
      if((zz=malloc(mny*mn*4))==NULL) out();
      memcpy(zz,b,mny*mn*4);
}
free(b);
if((b=malloc(ny*n*4))==NULL) out();
memset(b,0,ny*n*4);
do
{
 caz(2,"s");
 window(21,8,45,15);
 textattr(alb);
 clrscr();
 cputs("Save ? (Y/N)  ");
 setcrs(8,7);
 ch=toupper(getch());
}
while((ch!='Y')&&(ch!='N'));

if(ch=='N')
{
```

```
if(!este)
  {
     n=mn; ny=mny; nu=mnu; na=mna; nb=mnb;
     free(b);
     if((b=malloc(ny*n*4))==NULL) out();
     memcpy(b,zz,ny*n*4);
     free(zz);
  }
     setcrs(0,1);
     closewin(l1);
     return;
}

if(access("model.dta",0))
     {
       nui("\n\n   Data file model.dta not find");
       free(zz);
       setcrs(0,1);
       closewin(l1);
       return;
     }
f=open("model.dta",O_RDWR | O_BINARY);
gotoxy(3,3);
cputs("Save as : ");
if((kr=malloc(11))==NULL) out();
*kr=9;
cgets(kr);
for(i=strlen(kr+2);i<8;i++) *(kr+2+i)=' ';
*(kr+10)='\0';
lseek(f,0,SEEK_END);
write(f,kr+2,9);
write(f,&n,2);
write(f,&ny,2);
write(f,&na,2);
write(f,&nu,2);
write(f,&nb,2);
write(f,b,n*ny*4);
close(f);
free(zz);
setcrs(0,1);
closewin(l1);
}

/*
ch='o';
j=1;
do
{
    gotoxy(13,3);
    clreol();
    cgets(kr);
```

```
for(i=0;i<58;i++)
{
 if(i==55) { j=0; break; }
 if(eof(fis)) { j=0; break; }
 read(f,gs,9);
 if(!strcmp(gs,kr+2))
   {gotoxy(3,5);
    clreol();
    cprintf("Ovewrite %s ? ",gs);
    while((ch!='Y')&&(ch!='N')) ch=toupper(getch());
    if(ch=='Y') { klp=tell(f); j=0; }  break;
   }
if(strcmp(gs,kr+2)>0) { klp=tell(f); j=0; break; }
}
}
 while(j);
 cputs("Ok");
*/

/*
gotoxy(3,3);
cputs("Save as : ");
if((kr=malloc(11))==NULL) out();
*kr=9;
cgets(kr);
if(access("model.dta",0))
     {
       nui("\n\n   Data file model.dta not find");
       free(zz);
       setcrs(0,1);
       closewin(11);
       return;
     }
f=open("model.dta",O_RDWR | O_BINARY); j=0;
do
for(i=0;i<54;i++)
{
if(eof(fis)) break;
read(f,gs,9);
if(!strcmp(gs,kr+2))
{
clrscr();
gotoxy(3,3);
setcrs(8,7);
cprintf("Overide %s ?",gs);
while((ch!='Y')&&(ch!='N')) ch=toupper(getch());
if(ch=='Y') { klp=tell(f); j=1; } if(j) break;
}
} while(j!=1);
*/
```

```
/*-----------------------------------------
  ----------- fisierul box 2 ----------------
  -----------------------------------------*/

#include <conio.h>
#include <stdio.h>
#include <ctype.h>

extern struct winn
     {    int r,l,b,t,a; void *p;  } rr;
struct rr *i1,*i2;
extern int alb,negr;

/*------------- citeste int, long, float ------------*/

void puls(int i, int *intr, long int *lon, float *flo)
{
    i2=win(10,5,50,15,negr);
    setcrs(8,7);
    gotoxy(2,2);
    cputs("Current value is :");
    if(i==2) cprintf(" %ld",*lon); else
    if(i) cprintf(" %d",*intr);    else cprintf(" %f",*flo);
    gotoxy(5,4);
    cprintf("New value :");
    if(i==2) cscanf("%ld",lon); else
    if(i) cscanf("%d",intr); else cscanf("%f",flo);
    getch();
    closewin(i2);
    setcrs(0,1);
}

/****************** input  ******************/
int by_i,init_fi=0,ok_i=32,ok_0=251,ok_r=32;
extern float *uini;
extern nu,nb;

void okki(void)
{
ok(i1->l,i1->t,0,&ok_i);
ok(i1->l,i1->t+1,0,&ok_0);
ok(i1->l,i1->t+2,0,&ok_r);
}

void initt(void)
{
int i,j;
     i2=win(10,5,50,15,negr);
     setcrs(8,7);
     gotoxy(3,2);
```

```
     cprintf("U [NU=%d][NB=%d]",nu,nb);
     for(i=0;i<nu;i++) for(j=nb-1;j>=0;j--)
     {
     gotoxy(3,4);
     clreol();
     if(init_fi==2)
     { cprintf("U[%d][t-%d]=%f",i+1,nb-j,*(uini+i+j*nu)); }
          else puts("No init values");
     gotoxy(3,7);
     clreol();
     cputs("New value : ");
     cprintf("U[%d][t-%d]=",i+1,nb-j);
     cscanf("%f",uini+i+j*nu);
     }
     if(getch()=='\r') init_fi=2;
     closewin(i2);
     setcrs(0,1);
     if(init_fi==2)
     { ok_i=251,ok_0=32,ok_r=32; okki(); }
}

void dispune_i(void)
{    switch(by_i)
     {
          case 0: initt(); break;
          case 1: ok_i=32,ok_0=251,ok_r=32;
                  okki(); init_fi=0; break;
          case 2: ok_i=32,ok_0=32,ok_r=251;
                  okki(); init_fi=1; break;
     }
}

void tst_i(void)
{
     if(by_i<0) by_i=2; else if(by_i>2) by_i=0;
     inv(i1->l+1,by_i+i1->t+1,i1->r);
}

void inpu(void)
{
int i;
char *zoo[]={ " Init u", " Zero", " Random" };
     i1=win(17,9,11,3,alb);
     by_i=0; okki();
     for(i=0;i<i1->b;i++) { gotoxy(2,i+1); cprintf("%s",zoo[i]); }
     inv(i1->l+1,i1->t+1,i1->r);
     while(1)  {
     switch(getch())
     {
     case 27 : closewin(i1); return;
     case   '\r': dispune_i(); break;
```

```
     case   '\0': switch(getch())
            { case 72: inv(i1->l+1,by_i+i1->t+1,i1->r);
                       by_i--; tst_i(); break;
              case 80: inv(i1->l+1,by_i+i1->t+1,i1->r);
                       by_i++; tst_i(); break;
              case 0x3b: help(150+by_i); break;
            }
     }
     }
}

/***************** esti ***********************/

int by_e,est=0;

void okye(void)
{
     int i,j=32,k=251;
     for(i=0;i<3;i++)
     {    if(i==est)  ok(i1->l,i1->t+i,0,&k);
          else ok(i1->l,i1->t+i,0,&j);  }
}

void tst_e(void)
{
     if(by_e<0) by_e=2; else if(by_e>2) by_e=0;
     inv(i1->l+1,by_e+i1->t+1,i1->r);
}

void esti(void)
{
int i=251;
char *zoo[]={ " ud", " sq", " cmmp" };
     i1=win(17,10,8,3,alb); by_e=0;
     okye();
     for(i=0;i<i1->b;i++)
     { gotoxy(2,i+1); cprintf("%s",zoo[i]); }
     inv(i1->l+1,i1->t+1,i1->r);
     while(1)
     {
     switch(getch())
     {
     case 27 : closewin(i1); return;
     case   '\r': est=by_e; okye(); break;
     case   '\0': switch(getch())
            { case 72: inv(i1->l+1,by_e+i1->t+1,i1->r);
                       by_e--; tst_e(); break;
              case 80: inv(i1->l+1,by_e+i1->t+1,i1->r);
                       by_e++; tst_e(); break;
              case 0x3b: help(160+by_e); break;
            }
```

```
     }
     }
}

/******************* noise **********************/

float noise_val;
void nois(void)
{    i1=win(10,5,50,15,negr);
     setcrs(8,7);
     if(noise_val)
      {
        gotoxy(2,2);
        cprintf("Current noise value is : %.2f",noise_val);
      }
     else
      {
        gotoxy(2,2);
        cprintf("No current noise");
      }
     gotoxy(5,4);
     cprintf("New value :");
     cscanf("%f",&noise_val);
     if(noise_val<0)     noise_val=0;
     setcrs(0,1);
     getch();
     closewin(i1);
}

/*-----------------------------------------
  ----------- fisierul sertim.c -----------
  -----------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <graphics.h>
void EGAVGA_driver(void);

/*---------------- variabile externe ----------------*/

extern int lupa,par[2][8][100];
extern int init_fi,est;
extern long int loc;
extern char tra,axa0,lin,inni;
extern float noise_val,axax,lambda,alfa,*b,*bb,*uini;
extern int n,ny,nu,na,nb;

/*------------- converteste float->sir ------------*/
```

```
char sd[10];
char *rett(float fll, int nrci)
{   int ze,sgg,dec; char sii[10];
     strcpy(sii,fcvt(fll,nrci,&dec,&sgg));
     if(sgg) strcpy(sd,"-"); else strcpy(sd,"+");
     if(dec>0) strncat(sd,sii,dec);
     else  strcat(sd,"0"); strcat(sd,".");
     if(dec<0) for(ze=0;ze<-dec;ze++)
     strcat(sd,"0");
     if(dec>0) strcat(sd,sii+dec);
     else strcat(sd,sii); return sd;  }

/******************** estim ************************/
void estim(void)
{
int g1=VGA,g2=VGAHI;
int is_break=0,memo1,memo2,pix,axa,fzz=45;
int pas=0,i,j,t,r,lp,ii,jj,ij,ct;
char colo=WHITE,ch,gro,iss=0;
float s,m,aa=1,min=0,max=0;

float u[8][8],y[10][8],theta[8][100],tt[8][100];
float w[8],k[100],err[8],fi[100],a[100],*b1;
float v[100],f[50];
/************************  UDU  ************************/
float uu[100][100];
/************************ CMMP ************************/
float p[100][100];

if((b1=malloc(ny*n*4+1))==NULL) out();
memcpy(b1,b,ny*n*4+1);

     /*------------ initializare grafica -----------*/
     registerbgidriver(EGAVGA_driver);
     initgraph(&g1,&g2,"");
     /*gro=graphresult();     if( gro != grOk )
     { printf("Error: %s\n",grapherrormsg(gro));
       puts("This program requires a VGA monitor"); exit(0); }*/

/*------ initializeaza theta, min, max, is_break, iss -----*/
     for(i=0;i<ny;i++) for(j=0;j<n;j++)
     {
       if(*(b1+n*i+j)>max) max=*(b1+n*i+j);
       if(par[1][i][j]) is_break=1;
       if(par[0][i][j])
         {
         iss++;
         ii=i;
         jj=j;
         }
       if(inni==32) theta[i][j]=0;
```

```
        else theta[i][j]=*(b1+n*i+j);
        if(*(b1+n*i+j)<min) min=*(b1+n*i+j);
      }

/*---------------------- axele ----------------------*/
      pix=(460-2*loc)/(max-min);
      axa=loc+19+max*pix;
      if(axa0==32) axa=220+axax*pix;
      memo1=axa;
      setcolor(WHITE);

/*------- scrie parametrii A,B in stinga pe verticala ------*/
      for(i=0;i<ny;i++) for(j=0;j<n;j++)     {
      r=axa-*(b1+n*i+j)*pix;   if(*(b1+n*i+j)==min)
      outtextxy(1,r-7,rett(*(b1+n*i+j),2)); else
      outtextxy(1,r-3,rett(*(b1+n*i+j),2));
      for(t=fzz;t<640;t++)      putpixel(t,r,2);     }

      /*------------ traseaza axele -----------*/
      if(axa>19) for(i=fzz;i<640;i++)
      putpixel(i,axa,WHITE);
      for(i=19;i<479;i++)
      putpixel(fzz,i,WHITE);

      /*-------------                 ------------*/
      setcolor(WHITE);
      *a=lambda/aa;
      if(lupa>600) lp=-1; else
      { lp=594/lupa; while(lp*lupa>639-fzz) lp--; }

      /*------------ initializare u,y ------------*/
      for(i=0;i<nb;i++) for(j=0;j<ny;j++) y[i][j]=0;
      for(i=0;i<na;i++) for(j=0;j<nu;j++)
      {     if(!init_fi) u[i][j]=0.00001; else
            if(init_fi==1)  u[i][j]=((float)rand()/32760);
            else memcpy(u,uini,nu*na*4);   }

switch(est)
{
case 0:         /*---------- initializare udu ------------*/
      for(i=0;i<n;i++)
      {
            *(w+i)=0;
            *(fi+i)=0;
            for(j=0;j<n;j++)     if(i==j)
            {
             uu[i][j]=1;
             p[i][j]=alfa;
            }
            else
            {
```

```
            uu[i][j]=0;
            p[i][j]=0;
           }
      }
      break;
case 1:         /*---------- initializare sq ------------*/
      for(i=0;i<n;i++)
      {
           *(w+i)=0;
           *(fi+i)=0;
           v[i]=alfa;
           for(j=0;j<n;j++) if(i==j)
           p[i][j]=sqrt(alfa); else p[i][j]=0;
         }
        break;
case 2:         /*---------- initializare cmmp ------------*/
      for(i=0;i<n;i++)
      {
           *(w+i)=0;
           *(fi+i)=0;
           for(j=0;j<n;j++) if(i==j)
           p[i][j]=alfa; else  p[i][j]=0;
      }
      break;
}
/*-------------------------------------------------------
---- inceput ciclu simulare-estimare-afisare -----------
-------------------------------------------------------*/
while(1)
{     pas++;
      /*--------- actualizare fi ---------*/
      for(i=n-nu;i>=na*ny+nu;i--) for(j=0;j<nu;j++)
                           *(fi+i+j)=*(fi+i+j-nu);
      for(i=0;i<nu;i++) *(fi+i+na*ny)=u[nb-1][i];
      for(i=na*ny-ny;i>=ny;i--)    for(j=0;j<ny;j++)
                           *(fi+i+j)=*(fi+i+j-ny);
      for(i=0;i<ny;i++) *(fi+i)=-y[na-1][i];

      /*--------- test : zgomot, break, trace ---------*/
      if(noise_val)  for(i=0;i<ny;i++)
                        *(w+i)=((float)rand()/32767)/noise_val;
      if(lin==251)  for(i=0;i<ny;i++) for(j=0;j<n;j++)
                        tt[i][j]=theta[i][j];
      if(is_break)  for(i=0;i<ny;i++) for(j=0;j<n;j++)
                  if(pas==par[1][i][j]) *(b1+n*i+j)=*(bb+n*i+j);

      /*------------ simulare ---------------*/
      for(r=0;r<ny;r++)
      {
       y[na][r]=*(w+r);
       for(i=0;i<na;i++) for(j=0;j<ny;j++)
```

```
      y[na][r]+=-*(b1+n*r+ny*i+j)*y[na-1-i][j];
      for(i=0;i<nb;i++) for(j=0;j<nu;j++)
      y[na][r]+=*(b1+n*r+na*ny+nu*i+j)*u[nb-1-i][j];
     }

     /*-------- actualizare u (aleator 0...1) ---------*/
     for(i=0;i<nb-1;i++) for(j=0;j<nu;j++) u[i][j]=u[i+1][j];
     for(i=0;i<nu;i++)
     {
       if (i==0)
       /* semnalul haotic */
       u[nb-1][i]=1-1.4*u[nb-2][i]*u[nb-2][i]+0.3*u[nb-3][i];
       else
       u[nb-1][i]=((float)rand()/32767);}
     for(i=0;i<na;i++) for(j=0;j<ny;j++) y[i][j]=y[i+1][j];
switch(est)
{
case 0:         /*---------- estimare udu ------------*/
     for(i=0;i<ny;i++)
     {
      *(err+i)=y[na][i];
      for(j=0;j<n;j++)
      *(err+i)-=theta[i][j]**(fi+j);
     }
     for(i=0;i<n;i++)
     {
      *(f+i)=0;
      for(j=0;j<n;j++) *(f+i)+=uu[j][i]**(fi+j);
     }
     for(i=0;i<n;i++)
     {
      *(v+i)=0;
      for(j=0;j<n;j++) *(v+i)+=p[i][j]*(*(f+j));
     }
     for(j=0;j<n;j++)
     {
      uu[j][j]=1;
      *(k+j)=*(v+j);
      s=-*(f+j)/(*(a+j));
      *(a+j+1)=*(a+j)+*(v+j)**(f+j);
      p[j][j]*=*(a+j)/(lambda**(a+j+1));
      if(j>0) for(i=0;i<j;i++)
       {
        m=uu[i][j];
        uu[i][j]+=s**(k+i);
        *(k+i)+=*(v+j)*m;
       }
    }
    for(i=0;i<ny;i++) for(j=0;j<n;j++)
    theta[i][j]+=*(k+j)**(err+i)/(*(a+n));
    break;
```

```
case 1:         /*---------- estimare sq ------------*/
for(i=0;i<ny;i++)
      {
      *(err+i)=y[na][i];  for(j=0;j<n;j++)
      *(err+i)-=theta[i][j]**(fi+j);
      }
      for(i=0;i<n;i++)
      {
      *(f+i)=0;
      for(j=0;j<n;j++) *(f+i)+=p[j][i]**(fi+j);
      }
      for(j=0;j<n;j++)
      {
      *(k+j)=*(f+j)*p[j][j];
      s=-*(f+j)/(*(a+j));
      *(a+j+1)=*(a+j)+*(f+j)**(f+j);
      v[j]=sqrt(*(a+j)/(lambda**(a+j+1)));
      p[j][j]*=v[j];
      if(j>0) for(i=0;i<j;i++)
              {
              m=p[i][j];
              p[i][j]+=s**(k+i);
              p[i][j]*=v[j];
              *(k+i)+=*(f+j)*m;
              }
      }
      for(i=0;i<ny;i++) for(j=0;j<n;j++)
      theta[i][j]+=*(k+j)**(err+i)/(*(a+n));
      break;
case 2:         /*---------- estimare cmmp ------------*/
      for(i=0;i<n;i++)
      {
      a[i]=0;
      for(j=0;j<n;j++) a[i]+=p[i][j]**(fi+j);
      }
      s=lambda;
      for(i=0;i<n;i++) s+=*(fi+i)*a[i];
      for(i=0;i<n;i++) k[i]=a[i]/s;
      for(i=0;i<ny;i++)
      {
      *(err+i)=y[na][i];
      for(j=0;j<n;j++)
      *(err+i)-=theta[i][j]**(fi+j);
      }
      for(i=0;i<ny;i++) for(j=0;j<n;j++)
      theta[i][j]+=k[j]**(err+i);
      for(i=0;i<n;i++) for(j=0;j<n;j++)
      p[i][j]=(p[i][j]-k[j]*a[i])/lambda;
      break;
}
      /*-------- afisare estimatie si pas --------*/
```

```
      for(i=0;i<ny;i++) for(j=0;j<n;j++) if(par[0][i][j])
      {
      if(lin==32)
      putpixel(pas+lp*pas+fzz-1,axa-theta[i][j]*pix,colo);
      else
       {
       memo1=axa-tt[i][j]*pix;
       memo2=axa-theta[i][j]*pix;
       line(pas+lp*(pas-1)+fzz-2,memo1,pas+lp*pas-1+fzz,memo2);
       }
      }
      gotoxy(70,1);
      printf("pas=%d",pas);

      /*---------- teste iesire din functie, etc -----------*/

if(iss==1)
      {
       gotoxy(50,1);
       printf("%f",theta[ii][jj]);
      }
if(tra==251)
      {
       if(getch()==27)
       {
        closegraph();
        free(b1);
        return;
       }
      }
else
{
      if(pas==lupa)
      {
      getch(); /*closegraph(); free(b1); return; */
      }
      if(kbhit())
      {
       ch=getch();
       if(ch==27)
        {
         closegraph();
         free(b1);
         return;
        }
       if(ch=='\r') getch();
       if(ch=='v')
        {
         setactivepage(1);
         cleardevice();
         setvisualpage(1);
```

```
        setcolor(10);
        for(ij=0;ij<na;ij++)
        {
          outtextxy(50,45,"A");
          outtextxy(90,45,"estimat");
          outtextxy(60,45,itoa(ij+1,sd,10));
          for(i=0;i<ny;i++) for(j=0;j<ny;j++)
          outtextxy(j*70+40,i*15+60,rett(theta[i][j+ij*ny],5));
          ct=(6+ny*2)*5; outtextxy(50,45+ct,"A");
          outtextxy(90,45+ct,"exact");
          outtextxy(60,45+ct,itoa(ij+1,sd,10));
          for(i=0;i<ny;i++) for(j=0;j<ny;j++)
          outtextxy(j*70+40,i*15+60+ct,rett(*(b+n*i+j+ij*ny),5));
          if(getch()==27)
           {
            i=-1;
           break;
           }
          cleardevice();
        }
          if(i!=-1)
          for(ij=0;ij<nb;ij++)
           {
           outtextxy(50,45,"B");
           outtextxy(60,45,itoa(ij+1,sd,10));
           outtextxy(90,45,"estimat");
           for(i=0;i<ny;i++) for(j=0;j<nu;j++)
           outtextxy(j*70+40,i*15+60,
           rett(theta[i][j+ij*nu+ny*na],5));
           outtextxy(50,50+ct,"B");
           outtextxy(90,50+ct,"exact");
           outtextxy(60,50+ct,itoa(ij+1,sd,10));
           for(i=0;i<ny;i++) for(j=0;j<nu;j++)
           outtextxy(j*70+40,i*15+60+ct,
           rett(*(b+n*i+ny*na+j+ij*nu),5));
           if(getch()==27) break;
           cleardevice();
           }
          setvisualpage(0);
          setactivepage(0);
          setcolor(colo);
          }
      }

/*-------------------------------------------------------------
----- sfirsit ciclu simulare-estimare-afisare ---------------
-------------------------------------------------------------*/
```

```
/*---------------------------------------
  ---------- fisierul help.c ------------
  ---------------------------------------*/


#include <stdio.h>
#include <io.h>
#include <conio.h>
#include <string.h>
#include <fcntl.h>

extern struct winn
     {    int r,l,b,t,a; void *p;  } rr;
struct rr *h1;
extern int alb,negr,gri;
extern vb;

void help(int jj)
{
int fi,i,kk=-1,cit;
char *buf,prf2[7]=" Help ";
     if(access("sertim.hlp",0))
     {
     nui("\n\n   Help file sertim.hlp not found !");
     return;
     }
     fi=open("sertim.hlp",O_RDONLY | O_BINARY);
     h1=win(16,5,48,13,gri);
     for(i=0;i<6;i++)
     pokeb(vb,(h1->t+h1->b)*160+(h1->l+4+i)*2,prf2[i]);
     read(fi,&kk,2);
     read(fi,&cit,2);
     while(kk<jj)
     {
      lseek(fi,cit,SEEK_CUR);
      if(eof(fi)) break;
      read(fi,&kk,2);
      read(fi,&cit,2);
     }
     if(kk!=jj)
     {
       highvideo();
       gotoxy(4,6);
       cputs("No help available here !");
       gotoxy(30,9);
       cputs("Press Esc");
       while(getch()!=27);
       close(fi);
       closewin(h1);
       return;
     }
```

```
     if((buf=malloc(cit))==NULL)
     {
       puts("Outt rammm");
       exit(0);
     }
     read(fi,buf,cit);
     i=0;
     highvideo();
     while(*(buf+i)!='\0')
     {
      if(*(buf+i)==2) gotoxy(1,wherey()+1);
      else if(*(buf+i)==9) gotoxy(wherex()+8,wherey());
             else putch(*(buf+i)); i++;
     }
     while(getch()!=27);
     close(fi);
     free(buf);
     closewin(h1);
}

/*-----------------------------------------------
  ----------- fisierul main.c -----------------
  -----------------------------------------------*/

#include <stdio.h>
#include <process.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>

#define c_r 77
#define c_l 75
#define c_u 72
#define c_d 80
#define nrbox 4
#define st_box 0
#define lat_box 1
#define ad_box 2
#define lung_msj 3

typedef struct winn
     {    int r,l,b,t,a; void *p;  } rr;
rr *w1,*w2,*w3;

unsigned vb=0xb800;
char tra=32;
int klx,kly,ct=0,alb=0x70,negr=7,gri=0x37,par[3][8][100],este=1;
int a[4][nrbox+1]=
/*    st_box     lat_box      ad_box    lung_msj  */
{ {3,11,20,29}, {18,12,14,40}, {6,9,6,12}, {4,5,4,14} };
```

```
char *sefu[]={ " (c) 1995 Curiac Software " };
char *sus[]={ "File", "Estim", "Zoom", "Something else" } ;
char *fe[]= {
 "Load", "View", "New", "Del",
 "1234567890123456789", "Exit",
 " Noise", " Init", " Alfa", " Lambda",
 " Break", " Input", " Estim", " Go    ", " Param",
 " Zoom x"," Zoom y"," Abscisa", " Calibrat", " Lines", " Trace",
 "m1", "m2", "m3", "m4", "m5", "m6",
 "m1", "m2", "m3", "m4", "m5", "m6",
          };
void winx(void);
extern int est;

/************ vip ***********/

char axa0=251,lin=251,inni=32,tet=233;
int lupa=100;
long int loc=0;
float axax,lambda=1,alfa=1000,*b,*bb,*uini;

/************** mesaje *****************/

void inita(void)
{
int i,j;
textmode(BW80);
memset(fe[4],196,a[1][0]-2);
fe[4][a[1][0]-2]='\0';
for(i=0;i<8;i++)
   for(j=0;j<100;j++)
    {
     par[0][i][j]=1;
     par[1][i][j]=0;
     par[2][i][j]=0;
    }
b=malloc(1);
bb=malloc(1);
uini=malloc(1);
}

void firma(void)
{
int i; textattr(negr); clrscr();
window(1,1,80,2); textattr(alb); clreol();
for(i=0;i<nrbox;i++)
{
 gotoxy(a[st_box][i],1);
 cprintf("  %s  ",sus[i]);
}
gotoxy(70,1);
```

```
puts("F1-Help");
window(1,25,80,25);
cputs("    Estimator multivariabil on-line");
cprintf("                %c      U. P. Timisoara",179);
clreol();
fill(1,2,80,24,176);
w1=win(2,3,1,1,2);
winx();
}

void nema(void)
{
     w2=win(20,9,40,8,negr);
     cprintf("\n\n   Sorry, not availabe yet ...");
     gotoxy(30,7);
     cputs("press Esc");
     while(getch()!=27);
     closewin(w2);
}

void nem(void)
{
     w2=win(20,9,40,8,negr);
     cprintf("\n\n   You have to load a modell first");
     gotoxy(30,7);
     cputs("press Esc");
     while(getch()!=27); closewin(w2);
}

void bye(void)
{
   closewin(w1);
   textattr(7);
   clrscr();
   puts(" Bye!");
   setcrs(8,7);
   exit(0);
}

/***************** inv & ok  ****************/

void inv(int x,int y,int z)
{
int i;
char xh,xl,xm=0;
for(i=0;i<z;i++)
     {
     xh=peekb(vb,160*(y-1)+2*(x-1)+2*i+1);
     xl=xh;
     xh>>=4;
     xl<<=4;
```

```
     xm=xl>>7;
     if(xm) xl&=0x70; else xh|=8;
     pokeb(vb,160*(y-1)+2*(x-1)+2*i+1,xh|xl);
     }
}

void ms(void)
{
     inv( a[st_box][klx], 1, a[lung_msj][klx]+4 );
}

void ok(int xok, int yok, int iff, char *ch)
{
    if(iff) if(*ch==32) *ch=251;
            else *ch=32;
     pokeb(vb,160*yok+2*(xok++),*ch);
}

/***************** disp ******************/

void winx(void)
{
int i,j=0;
closewin(w1);
if(klx>nrbox-1) klx=0;
if(klx<0) klx=nrbox-1;
w1=win(a[st_box][klx],2,a[lat_box][klx],a[ad_box][klx],alb);
ms();
for(i=0;i<klx;i++) j+=a[ad_box][i];
for(i=0;i<a[ad_box][klx];i++)
     {
     gotoxy(2,i+1);
     cprintf("%s",fe[i+j]);
     }
     if(klx==1)
     {
     ok(w1->l,3,0,&inni);
     ok(w1->l+7,3,0,&tet);
     }
     if(klx==2)
     {
     ok(w1->l,7,0,&tra);
     ok(w1->l,5,0,&axa0);
     ok(w1->l,6,0,&lin);
     }
     inv( a[st_box][klx]+1, 3, a[lat_box][klx] );
     kly=0;
}

void winy(void)
{
```

```
     if(!klx) if(kly==4) if(ct==3) kly++;
                          else kly--;
     if(kly>a[ad_box][klx]-1) kly=0;
     if(kly<0) kly=a[ad_box][klx]-1;
     inv(w1->l+1,w1->t+1+ct, a[lat_box][klx]);
     inv(w1->l+1,w1->t+1+kly,a[lat_box][klx]);
}

void dispune(void)
{
switch(klx)
{
     case 0: switch(kly)
                    {
                         case 0: load();
                                 break;
                         case 1: if(este) nem();
                                 else view(); break;
                         case 2: new();
                                 break;
                         case 3: del();
                                 break;
                         case 5: bye();
                    } break;
     case 1: switch(kly)
                    {
                         case 0: nois();
                                 break;
                         case 1: ok(w1->l,2+kly,1,&inni);
                                 break;
                         case 2: puls(0,0,0,&alfa);
                                 break;
                         case 3: puls(0,0,0,&lambda);
                                 break;
                         case 4: if(este) nem();
                                 else brea();
                                 break;
                         case 5: inpu();
                                 break;
                         case 6: esti();
                                 break;
                         case 7: if(este) nem();
                                 else
                                 {
                                  closewin(w1);
                                  estim();
                                  firma();
                                 }
                                 break;
                         case 8: if(este) nem();
                                 else para();
```

```
                                    break;
                        } break;
      case 2: switch(kly)
                        {
                              case 0: puls(1,&lupa,0,0);
                                    if(lupa<1) lupa=1;
                                    if(lupa>600) lupa=600;
                                    break;
                              case 1: puls(2,0,&loc,0);
                                    if(loc>230) loc=230;
                                    if(loc<-80000) loc=-80000;
                                    break;
                              case 2: puls(0,0,0,&axax);
                                    axa0=251;
                                    ok(w1->l,w1->t+3,1,&axa0);
                                    setcrs(0,1);
                                    break;
                              case 3: ok(w1->l,2+kly,1,&axa0);
                                    break;
                              case 4: ok(w1->l,2+kly,1,&lin);
                                    break;
                              case 5: ok(w1->l,2+kly,1,&tra);
                                    break;
                        } break;
      default   : nema(); break;
}
}

/**************** main ********************/

void main(void)
{
inita();
firma();
while(1)
{
      switch(toupper(getch()))
      {
      case    '\r': dispune();
                    break;
      case    '\0': switch(getch())
                    {
                      case      c_u: ct=kly--; winy();
                                   break;
                      case      c_d: ct=kly++; winy();
                                   break;
                      case      c_r: ms();
                                   klx++;
                                   winx();
                                   break;
                      case      c_l: ms();
```

```
                                klx--;
                                winx();
                                break;
                    case  0x3b: help(klx*10+kly);
                                break;
                    case  0x3d: if(este) nem();
                                else
                                {
                                closewin(w1);
                                estim();
                                firma();
                                }
                                break;
                    }
          }
}
}

/*-----------------------------------------
  ------------- fisierul para.c ------------
  -----------------------------------------*/

#include <conio.h>
#define c_r 77
#define c_l 75
#define c_u 72
#define c_d 80

extern struct win
     {    int r,l,b,t,a; void *p;  } rr;
struct rr *p1,*p2;
extern int alb,negr,par[3][8][100];
extern int n,ny,nu,na,nb;
extern float *b,*bb;
extern unsigned vb;

int by_p,ok_all=251,ok_no=32,ok_ca=32,xp,yp,cx,cy,naa;

/*************** para ******************/

void okk(void)
{
ok(p1->l,p1->t,0,&ok_ca);
ok(p1->l,p1->t+1,0,&ok_no);
ok(p1->l,p1->t+2,0,&ok_all);
}

void all(void)
{
int i,j;
for(i=0;i<5;i++) for(j=0;j<100;j++) par[0][i][j]=1;
```

```
ok_all=251;
ok_no=32;
ok_ca=32;
okk();
}

void none(void)
{
int i,j;
for(i=0;i<5;i++) for(j=0;j<100;j++) par[0][i][j]=0;
ok_all=32;
ok_no=251;
ok_ca=32;
okk();
}

void end(void)
{
 closewin(p2);
 ok_all=32;
 ok_no=32;
 ok_ca=251;
 okk();
}

void inv2(int x,int y,int z)
{
int i;
char xh,xl,xm;
for(i=0;i<z;i++)
 {
     xh=peekb(vb,160*(y-1)+2*(x-1)+2*i+1);
     xl=xh;
     xm=xh;
     xh>>=4;
     xl<<=4;
     xh&=7;
     xm>>=7;
     xm<<=7;
     xl|=xm;
     pokeb(vb,160*(y-1)+2*(x-1)+2*i+1,xh|xl);
 }
}

void inv3(int x,int y,int z)
{
int i;
char xh;
for(i=0;i<z;i++)
     {
     xh=peekb(vb,160*(y-1)+2*(x-1)+2*i+1);
```

```
      pokeb(vb,160*(y-1)+2*(x-1)+2*i+1,xh^0x80);
      }
}

void sujo(void)
{
      if(yp<0) yp=ny-1;
      if(yp==ny) yp=0;
      inv3(p2->l+5+7*xp,p2->t+4+yp,5);
      inv3(p2->l+5+7*xp,p2->t+4+cy,5);
}

void drst(int ab)
{
      if(!ab)
      {
      if(xp<0) xp=ny-1;
      if(xp==ny) xp=0;
      }
      else
      {
      if(xp<0) xp=nu-1;
      if(xp==nu) xp=0;
      }
      inv3(p2->l+5+7*xp,p2->t+4+yp,5);
      inv3(p2->l+5+7*cx,p2->t+4+yp,5);
}

void inn(int vl)
{
int ik;
ik=par[1][yp][vl];
p1=win(17,12,28,4,alb);
setcrs(8,7);
gotoxy(2,1);
if(ik)
  {
      cputs("Old pas break : ");
      cprintf("%d",par[1][yp][vl]);
      gotoxy(2,2); cputs("Old value      : ");
      cprintf("%f",*(bb+yp*n+vl)); }
      else cputs("No break point");
      gotoxy(2,3);
      cputs("New pas break : ");
      cscanf("%d",&par[1][yp][vl]);
      if(par[1][yp][vl]<1) par[1][yp][vl]=0;
      else
      {
      gotoxy(2,4);
      cputs("New value      : ");
      cscanf("%f",bb+yp*n+vl);
```

```
     }
     if(getch()==27) ik=-1;
     setcrs(0,1);
     closewin(p1);
     window(p2->l+1,p2->t+1,p2->l+p2->r,p2->t+p2->b);
     if(ik!=-1) if(!(par[1][yp][vl] && ik))
     inv2(p2->l+5+7*xp,p2->t+4+yp,5);
}

void rdda(void)
{
     p1=win(17,12,28,2,alb);
     setcrs(8,7);
     cputs(" New value : ");
     cscanf("%f",b+yp*n+xp+naa*ny);
     setcrs(0,1);
     closewin(p1);
     getch();
     window(p2->l+1,p2->t+1,p2->l+p2->r,p2->t+p2->b);
     gotoxy(5+7*xp,4+yp);
     textattr(negr);
     cprintf("%+3.2f",*(b+yp*n+xp+naa*ny));
     inv3(p2->l+5+7*xp,p2->t+4+yp,5);
}

void rddb(void)
{
     p1=win(17,12,28,4,alb);
     setcrs(8,7);
     cputs(" New value : ");
     cscanf("%f",b+yp*n+xp+naa*ny+na*ny);
     setcrs(0,1);
     closewin(p1);
     getch();
     window(p2->l+1,p2->t+1,p2->l+p2->r,p2->t+p2->b);
     gotoxy(5+7*xp,4+yp);
     textattr(negr);
     cprintf("%+3.2f",*(b+yp*n+xp+naa*ny+na*ny));
     inv2(p2->l+5+7*xp,p2->t+4+yp,5);
}

void goo(int pee)
{
     switch(pee)
     {
          case 0: inv2(p2->l+5+7*xp,p2->t+4+yp,5);
                  par[0][yp][xp+naa*ny]=
                  !par[0][yp][xp+naa*ny];
                  break;
          case 1: inv2(p2->l+5+7*xp,p2->t+4+yp,5);
                  par[0][yp][xp+naa*nu+na*ny]=
```

```
                    !par[0][yp][xp+naa*nu+na*ny];
                    break;
          case 2:   inn(xp+naa*ny);
                    break;
          case 3:   inn(xp+naa*nu+na*ny);
                    break;
          case 4:   rdda();
                    break;
          case 5:   rddb();
                    break;
     }
}

int ask(int i,int j)
{
int brk=0;
inv3(p2->l+5,p2->t+4,5);
while(1)
{
     switch(getch())
     {
     case    27 : return 1;
     case   '\0': switch(getch())
                  {
                   case  0x3d: brk=1;
                               break;
                   case     c_u: cy=yp--;
                                 sujo();
                                 break;
                   case     c_d: cy=yp++;
                                 sujo();
                                 break;
                   case     c_r: cx=xp++;
                                 drst(j);
                                 break;
                   case     c_l: cx=xp--;
                                 drst(j);
                                 break;
                  } break;
     case   '\r': goo(i*2+j);
                  break;
     } if(brk) break;
} return 0;
}

void caz(int wfd, char *sir)
{
int i,j;
char prf3[30]=" Press F3 for the next matrix ";
p2=win(7,5,66,15,negr);
for(i=0;i<30;i++)
```

```
     pokeb(vb,(p2->t+p2->b)*160+(p2->l+4+i)*2,prf3[i]);
for(naa=0;naa<na;naa++)
     {
     xp=0;
     yp=0;
     gotoxy(5,2);
     cprintf("Current A%d parameter",naa+1); cputs(sir);
     for(j=0;j<ny;j++)
          {
          gotoxy(5,4+j);
          for(i=0;i<ny;i++)
           {
            textattr(negr);
            if(par[wfd][j][i+naa*ny]) textattr(alb);
            cprintf("%+3.2f",*(b+j*n+i+naa*ny));
            textattr(negr); cputs("  ");
           }
         }
     if(ask(wfd,0))
        {
        if(!wfd) end();
        else closewin(p2);
        return;
        }
     textattr(negr);
     clrscr();
   }
     for(naa=0;naa<nb;naa++)
     {
     xp=0;
     yp=0;
     gotoxy(5,2);
     cprintf("Current B%d parameter",naa+1);
     cputs(sir);
     for(j=0;j<ny;j++)
          {
          gotoxy(5,4+j);
          for(i=0;i<nu;i++)
           {
           textattr(negr);
           if(par[wfd][j][i+naa*nu+na*ny]) textattr(alb);
           cprintf("%+3.2f",*(b+j*n+i+naa*nu+na*ny));
           textattr(negr); cputs("  ");
           }
          }
     if(ask(wfd,1))
          {
            if(!wfd) end();
            else closewin(p2);
            return;
          }
```

```
     textattr(negr);
     clrscr();
    }
     if(!wfd) end();
     else closewin(p2);
}

void dispune_p(void)
{
    switch(by_p)
     {
          case  0: caz(0,"s shown on screen :");
                   break;
          case 2: all();
                  break;
          case 1: none();
                  break;
     }
}

void tst_p(void)
{
     if(by_p<0) by_p=2;
     else if(by_p>2) by_p=0;
     inv(p1->l+1,by_p+p1->t+1,p1->r);
}

void para(void)
{
int i=251;
char *zoo[]={ " case", " none", " all" };
     p1=win(17,12,8,3,alb); by_p=0; okk();
     for(i=0;i<p1->b;i++)
     {
      gotoxy(2,i+1);
      cprintf("%s",zoo[i]);
     }
     inv(p1->l+1,p1->t+1,p1->r);
     while(1)
     {
     switch(getch())
     {
     case 27 : closewin(p1);
               return;
     case   '\r': dispune_p();
                  break;
     case   '\0': switch(getch())
           {
             case   72: inv(p1->l+1,by_p+p1->t+1,p1->r);
                        by_p--;
                        tst_p();
```

```
                    break;
               case   80: inv(p1->l+1,by_p+p1->t+1,p1->r);
                    by_p++;
                    tst_p();
                    break;
               case 0x3b: help(180+by_p);
                    break;
          }
     }
}
}

/**************** break **********************/

void brea(void)
{
     caz(1," break points :");
}
```

CAPITOLUL 3: UTILIZAREA RETELELOR NEURONALE IN ANALIZA SI PREDICTIA SERIILOR DE TIMP

Prezentul capitol isi propune expunerea unor solutii originale destinate analizei si predictiei seriilor de timp, solutii bazate pe utilizarea retelelor neuronale. Capitolul demareaza printr-o prezentare unitara, succinta a retelelor neuronale (paragraful 3.1), urmând ca mai apoi sa se prezinte o solutie pentru modelarea si predictia seriilor de timp (paragraful 3.2). Totodata este descrisa si o metoda originala pentru antrenarea retelelor neuronale fara interventia operatorului uman (paragraful 3.1.2.4).

3.1. RETELELE NEURONALE

Sistemul nervos uman prelucreaza mai multe miliarde de informatii in modul sau propriu. Un tânar domeniu de cercetare -Neuroinformatica- se ocupa de transpunerea modului de prelucrare a informatiei folosit de sistemul nervos uman in retele neuronale artificiale folosite in tehnica.

Cu ajutorul retelelor neuronale se incearca dezvoltarea unor sisteme cu performante "inteligente" in domenii diverse cum ar fi:

- recunoasterea formelor;
- recunoasterea si sintetizarea vorbirii (sunete si cuvinte articulate);
- diagnoza medicala, analiza radiografiilor;
- predictia fenomenelor economice;
- simularea si conducerea unor procese (procese chimice, conducerea unor vehicule autonome, supravegherea spatiului aerian, etc.);
- modelarea si predictia seriilor de timp.

Produsele acestor cercetari nu mai sunt programe uriase rulând pe supercalculatoare de tip Cray sau Supremum, ci devin programe scrise pentru hardware-uri special concepute. Aceste hardware-uri

vor fi compuse din mai multe mii de cipuri (procesoare) legate intre ele ca neuronii, si care se vor constitui intr-o retea neuronala (ca noduri si sinapse).

Argumentele esentiale in utilizarea retelelor neuronale sunt:

- capacitatea de a "invata". Prezentând retelei o serie de exemple dintr-o baza de date de antrenament ea se autoorganizeaza in conformitate cu cele prezentate;
- capacitatea de generalizare. Daca au fost antrenate corespunzator, retelele sunt capabile sa ofere raspunsuri corecte chiar si in cazul unor intrari diferite fata de cele cu care au fost antrenate, atât timp cât aceste intrari nu sunt foarte diferite;
- capacitatea de sinteza. Retelele neuronale artificiale pot lua decizii sau trage concluzii când sunt confruntate cu informatii complexe sau cu zgomote irelevante sau partiale;
- procesarea paralela, deci rapiditate fata de calculatoarele digitale seriale;
- avantajul descrierii unei probleme si a rezolvarii ei in acelasi timp de catre reteaua neuronala prin autoorganizarea acesteia si nu prin program. Acest proces de autoorganizare are loc pe parcursul unei etape de invatare prin imbinarea unei arhitecturi initiale, a unor reguli de invatare si a unui numar mare de antrenamente.

In fapt - ceea ce este important - o retea invata sa aproximeze o functie y=f(x) prin incercari ale perechilor (x,y) din baza de date de antrenament. Faptul ca functia care trebuie invatata este neliniara nu prezinta probleme.

Exista o mare varietate de retele studiate sau utilizate in aplicatii. Dintre acestea se detaseaza retelele adaptive, retelele cu memorie asociativa (folosite in recunoastere), retelele cu propagare inversa, retelele supervizoare (care generalizeza doar dupa câteva incercari). Totusi cea mai utilizata retea pentru invatare este reteaua cu propagare inversa (backpropagation net) sau BPN. Aceasta va fi prezentata succint in continuare si utilizata in unele aplicatii [Mul90] [Nar92] [Ngu92] [Pal93] [Tud93_1].

In figura 3.1 este prezentat schematic un exemplu de retea BPN pe trei nivele. Dreptunghiurile reprezinta neuroni iar liniile dintre acestia interconexiuni (sinapse).

Dupa cum se observa BPN are o structura stratificata, prezentând minim trei nivele sau straturi: de intrare (input), ascuns sau intermediar (hidden) si de iesire (output).

Neuronii din stratul de intrare au doar rolul de a memora informatia, ei nefiind activi ca ceilalti. In cazul retelelor neuronale digitale, acesti neuroni mai au rolul de elemente de esantionare.

Cybenko [Cyb89] arata ca o retea cu doua straturi ascunse si orice neliniaritate sigmoidala continua fixata este suficienta pentru a aproxima orice functie arbitrara continua pe un interval compact. Cu toate ca rezultatele lui Cybenko nu dau un indiciu a cât e necesar a fi reteaua de mare (numar de nivele intermediare si numarul de neuroni din acestea) ele arata ca structura fundamentala

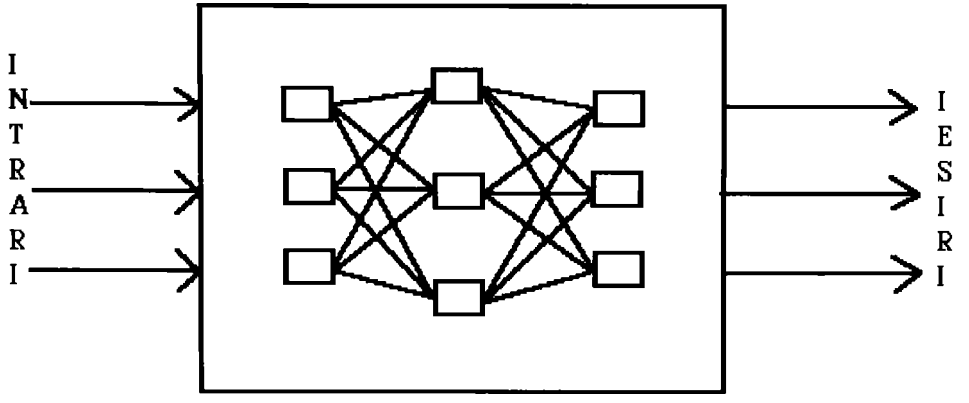


Figura 3.1. Retea neuronala pe trei straturi

a retelei BPN permite modelarea oricarei functii neliniare continue. In continuare, pe baza concluziilor lui Cybenko se vor considera doar BPN cu doua straturi ascunse.

Utilizarea unei retele presupune parcurgerea a doua faze:
- faza de invatare sau de antrenament in care retelei i se prezinta o serie de exemple din baza de date de antrenament si are loc stabilirea parametrilor acesteia (ponderi, factori de scala). Odata antrenata, reteaua poate fi folosita on-line oricând.
- faza de implementare a retelei neuronale antrenate (cu ponderile si factorii de scala fixate) in varianta hardware utilizând cipuri specializate [Atl92] [Gra92] [Mah92] [Mas92] [Sac92] sau in varianta software prin simularea retelei pe un calculator numeric.

Deoarece puterea de generalizare nu este mare, in timpul utilizarii retelei, in cazul in care apar neconcordante intre rezultatele obtinute si cele scontate este necesara reluarea fazei de invatare adaugând noi date in baza de date de antrenament.

3.1.1 MODELUL NEURONULUI McCULLOGH-PITTS [Wid92]

In anul 1943 McCulloch si Pitts au prezentat modelul matematic al neuronului artificial in articolul cu titlul "A Logical Calculus of the Ideas Immanent in Nervous Activity". Astfel, modelul McCulloch-Pitts descrie neuronul ca o celula a carei stare (net) este suma intrarilor ($inp_1, inp_2, \ldots, inp_n$) care sosesc pe cai cu diferite ponderi ($w_1, w_2, \ldots, w_n$) la care se adauga un termen de

"prepolarizare" care semnifica starea initiala a celulei, numit ulterior si bias sau factor de scala (figura 3.2).

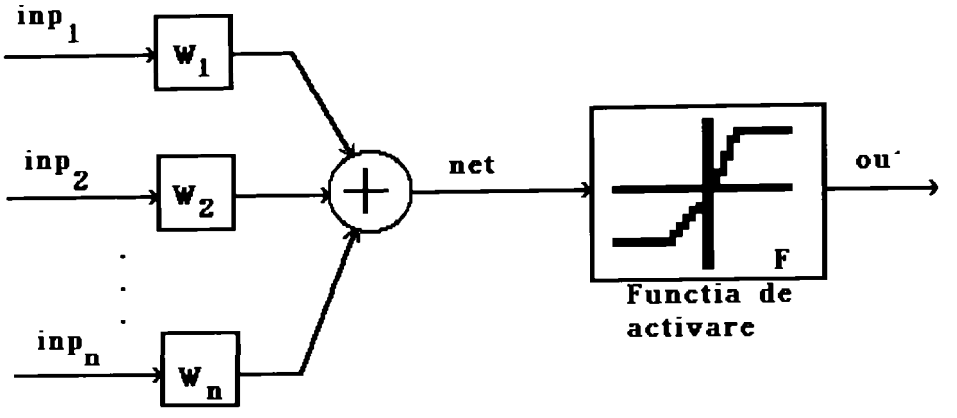


Figura 3.2: Neuronul artificial

$$net = \sum_{i=1}^{n} inp_i w_i + bias \qquad (3.1)$$

Sumei obtinute pe baza relatiei (3.1) i se aplica o functie de activare rezultând valoarea iesirii neuronului respectiv:

$$out = F(net) \qquad (3.2)$$

Functia de activare F poate sa fie o simpla functie treapta in cazul in care reteaua prelucreaza informatii binare:

$$F(x) = \begin{cases} 1 & ,x>t \\ 0 & ,altfel \end{cases} \qquad (3.3)$$

Figura 3.3

sau o functie mai complicata (o sigmoida) in cazul in care se opereaza cu semnale analogice, cu rol de a restringe domeniul de variatie al valorii de iesire a neuronului, indiferent de valoarea net. Câteva tipuri de functii de activare mai des utilizate in practica retelelor neuronale sunt explicitate in continuare:

$$F(x) = \frac{1}{1+e^{-x}} \qquad (3.4)$$

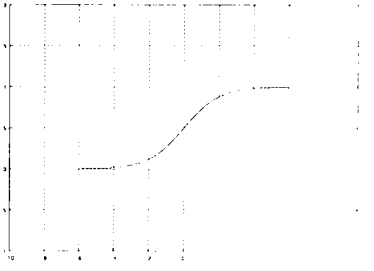


Figura 3.4

$$F(x) = \tanh(x) \qquad (3.5)$$

Figura 3.5

$$F(x) = \frac{2}{1+e^{-x}} - 1 \qquad (3.6)$$

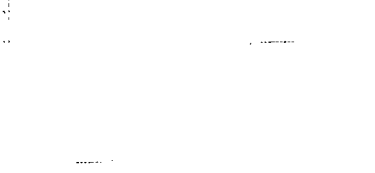


Figura 3.6

Alegerea unei sigmoide (functie continua al carei grafic prezinta doua asimptote orizontale si un singur punct de inflexiune) ca functie de activare este sustinuta teoretic de urmatoarele doua motivatii:
- necesitatea antrenarii tuturor neuronilor din retea. Daca functia de activare ar fi o functie rampa numai neuronii de pe stratul de iesire ar participa in procesul de antrenare, ei neavând nevoie de "ajutorul" neuronilor de pe straturile inferioare.
- necesitatea ca in cazul unei implementari hardware a retelei, curentii si tensiunile pe circuite sa fie limitate.

Alegerea unui anumit tip de functie de activare se realizeaza pe baza experimentelor. Dezvoltarile care urmeaza in prezentul capitol considera o functie de activare data prin relatia (3.5).

Atât ponderile cât si factorul de scala isi pot modifica valorile in timpul unui proces de antrenare, asupra caruia se va reveni pe larg in paragrafele urmatoare. Se impun anumite precizari cu privire la necesitatea si in acelasi timp avantajul introducerii unui parametru antrenabil pentru fiecare neuron din retea, numit deplasare de scala sau bias (in modelul McCullogh-Pitts acesta avea valoare constanta). Acest parametru are rolul de a deplasa originea functiei de activare producând un efect similar cu modificarea pragului de activare al neuronului. Pentru o tratare unitara a ponderilor si a factorului de scala, acest parametru - deplasare de scala se considera ca fiind ponderea unei legaturi la "+1".

3.1.2. RETELE NEURONALE FEEDFORWARD [Tod94] [Wid92]

Din multitudinea tipurilor de retele neuronale prezentate in literatura de specialitate retelele neuronale feedforward (cu reactie inainte) se detaseaza net, atât prin generalitatea structurii cât si prin gradul de utilizare in diverse aplicatii.

Intr-o retea neuronala feedforward neuronii artificiali sunt organizati pe nivele (straturi). Conexiunile dintre neuroni sunt permise doar intre neuroni apartinând unui nivel inferior cu neuroni apartinând nivelului imediat superior. Nu sunt permise conexiuni intre neuronii aceluiasi nivel si nici conexiuni dinspre nivelele superioare spre cele inferioare.

In figura (3.7) este prezentata simplificat o retea neuronala feedforward total conectata având un nivel de intrare, doua nivele intermediare si un nivel de iesire. Termenul "total conectata" din propozitia precedenta exprima faptul ca intre neuronii de pe un strat inferior si neuronii de pe stratul imediat superior sunt realizate toate conexiunile posibile. In caz contrar reteaua se numeste local conectata.

Iesirea retelelor neuronale feedforward este determinata numai de valorile curente ale intrarii si de cele ale ponderilor si factorilor de scala. Ele nu au memorie, nefiind retele recurente.

In proiectarea si implementarea unei retele neuronale

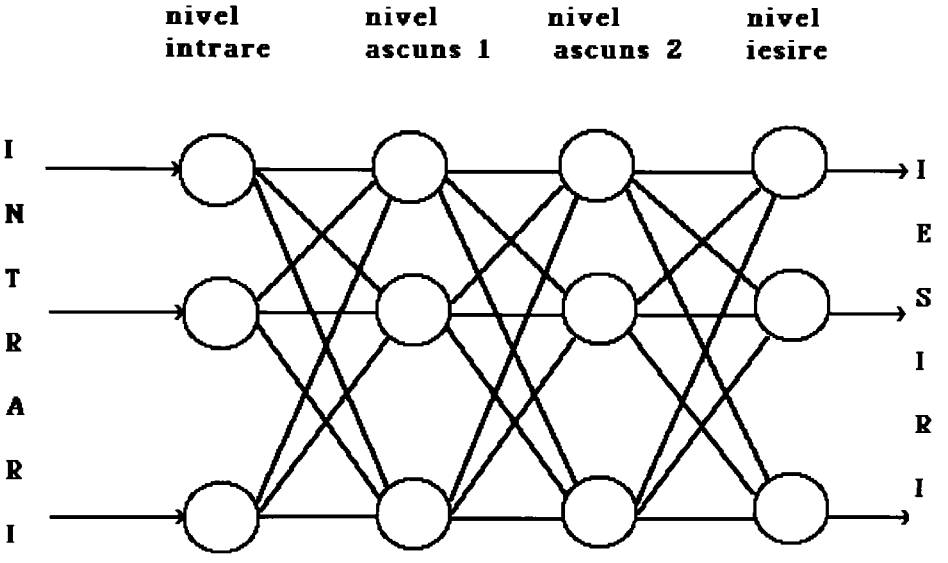


Figura 3.7: Retea neuronala total conectata

artificiale de orice tip un rol imporant il are faza de invatare (de antrenare). Performantele retelei sunt impuse in primul rând de reusita acestei faze. Algoritmul de antrenare destinat retelelor neuronale feedforward care furnizeaza cele mai bune rezultate este algoritmul de propagare inapoi (backpropagation).

In cazul retelelor neuronale feedforward antrenarea poate fi definita ca fiind procesul de ajustare a ponderilor si a factorilor de scala in scopul minimizarii erorilor intre baza de date de antrenament si rezultatele obtinute prin functionarea cu parametrii (ponderi si factori de scala) fixati ai retelei neuronale. Cunostintele unei retele neuronale artificiale sunt inglobate in ponderile sale, ponderi care se ajusteaza in faza de antrenare.

Algoritmii de antrenare a retelelor neuronale se impart in doua categorii, supervizati si nesupervizati [Tod94].

Invatarea supervizata, care se realizeaza prin algoritmul de propagare inapoi, presupune pentru fiecare vector de intrare câte un vector de iesire reprezentând raspunsul care se doreste sa-l dea reteaua când se prezinta la intrare vectorul de intrare. Impreuna, vectorul de intrare si de iesire asociat, formeaza o pereche de antrenament. O retea este antrenata cu un numar de perechi de antrenament ce depinde de tipul si complexitatea problemei. Se aplica un vector de intrare, se determina o anumita valoare a iesirii retelei care se compara cu vectorul de iesire asociat celui de intrare in perechea de antrenament, iar diferenta, reprezentând de fapt eroarea, se propaga inapoi in retea si ponderile se

ajusteaza conform unui algoritm care tinde sa minimizeze eroarea. Vectorii de antrenament se aplica secvential, erorile se calculeaza si se ajusteaza ponderile pentru fiecare pereche, până când eroarea corespunzatoare intregului set de antrenament se afla sub un anumit prag minim.

Invatarea nesupervizata se pare ca se apropie mai mult de modul de invatare dintr-un sistem biologic, nemainecesitând vectori de iesire asociati celor de intrare si nici comparari si propagarea erorii pentru determinarea raspunsului ideal. Setul de antrenament consta numai din vectori de intrare. In timpul unui proces de invatare nesupervizata se realizeaza gruparea vectorilor de intrare similari in clase. Adica, aplicându-se unul din vectorii de antrenament sau un vector care este suficient de apropiat de acesta, reteaua va produce, aceeasi iesire. Trebuie mentionat ca nu exista nici o posibilitate de determinare inaintea procesului de antrenament, a vectorului de iesire asociat de retea unei clase din setul vectorilor de antrenament.

3.1.2.1. ALGORITMUL DE PROPAGARE INAPOI (BACKPROPAGATION) [Tod94] [Wid92]

Acest algoritm, cel mai utilizat, face parte din categoria algoritmilor supervizati. Algoritmul de propagare inapoi poate fi aplicat retelelor feedforward cu oricâte nivele dar se poate aplica si retelelor recurente.

Algoritmul poate fi descris sintetic prin urmatorii pasi:

```
1. Declaratii variabile si constante;
2. Initializari variabile; /* ponderi si factori de scala */
3. while (eroarea obtinuta > eroarea dorita)
     {
4.       for (fiecare pereche de antrenament)
         {
5.         forward(); /* se aplica vectorul de intrare curent
                          din baza de date de antrenament la
                          intrarea retelei si are loc
                          propagarea fluxului neuronal nivel cu
                          nivel pe baza relatiilor (3.1) si
                          (3.2) pâna la nivelul de iesire. */
6.         backward(); /* ajustarea ponderilor si a factorilor
                          de scala astfel incât eroarea sa fie
                          minimizata. Eroarea este calculata
                          pe baza iesirii obtinute si a celei
                          prescrise (furnizata de baza de date
                          de antrenament) si este propagata
                          inapoi, nivel cu nivel spre
                          intrare. */
         }
```

}

Algoritmul de propagare inapoi prezinta doi pasi importanti, descrisi prin functiile forward() si backward().

Functia forward()

In acesta functie este realizata propagarea fluxului neuronal dinspre intrare spre iesire, nivel cu nivel. Astfel, la intrarea retelei este aplicat vectorul de intrare curent al perechii de antrenament, starile neuronilor fiind calculate pentru neuronii de pe primul strat intermediar pe baza relatiei (3.1).

Prin aplicarea functiei de activare conform relatiei (3.2) se obtin iesirile neuronilor de pe primul nivel intermediar. In continuare, iesirile neuronilor din primul nivel intermediar se constituie ca intrari in neuronii nivelului imediat urmator. Procesul descris mai sus se repeta nivel cu nivel pâna se ajunge la nivelul de iesire si se determina valorile iesirii retelei (iesirile neuronilor de pe stratul de iesire).

Observatie: Neuronii de pe nivelul de intrare nu sunt neuroni activi, fapt pentru care relatiile (3.1) si (3.2) nu au valabilitate in acest caz.

Functia backward()

In aceasta functie are loc ajustarea propriu-zisa a ponderilor si a factorilor de scala ale retelei. Baza de date de antrenament este constituita din perechi de antrenament (inp;tp), adica dintr-un vector de intrare (inp) si un vector de iesire prescris (tp), care se doreste a fi raspunsul retelei când acesteia i se aplica la intrare vectorul inp. Diferenta dintre tp si iesirea actuala (output) reprezinta eroarea retelei in momentul respectiv. Ponderile si factorii de scala sunt ajustate dinspre nivelul de iesire spre nivelul de intrare, nivel cu nivel pe baza unor reguli bine stabilite astfel incât eroarea sa tinda spre zero, adica la aplicarea vectorului inp din perechea de antrenament, iesirea obtinuta output sa fie cât mai apropiata (in cazul ideal identica), cu vectorul de iesire prescris (tp) din perechea de antrenament. In cele mai multe cazuri, functia backward() se rezuma la un algoritm iterativ (secvential) destinat reactualizarii ponderilor si a factorilor de scala, adica se aplica secvential fiecare pereche (inp;tp) din setul de antrenament si se reactualizeaza ponderile si factorii de scala, aceasta ori de câte ori este necesar pentru ca eroarea sa se situeze sub un anumit prag minim. Perioada de timp in care se realizeaza parcurgerea intregului set de perechi de antrenament de catre retea se numeste epoca de antrenament.

Dupa cum se poate constata, pentru o structura de retea feedforward fixata (numar de straturi si numar de neuroni pe fiecare strat fixate), vectorul iesirilor actuale ale retelei (output) depinde doar de vectorul intrarilor in retea (inp), si de parametrii retelei, adica de ponderile (w) si de factorii de scala (bias).

output=f(w,bias,inp) (3.7)

In cadrul metodei backpropagation, functia de eroare se considera a fi dependenta de vectorul iesirilor prescrise (tp) si vectorul iesirilor actuale (output):

eroarea=g(tp,output) (3.8)

Tinând cont de relatia (3.7), rezulta:

eroarea=g(tp,f(w,bias,inp)) (3.9)

Deci, global, eroarea la un moment dat este dependenta de intrarea retelei, iesirea obtinuta a retelei, iesirea dorita, ponderile si factorii de scala ai retelei. Intrucât vectorii inp si tp sunt furnizati de baza de date de antrenament, deci sunt cunoscuti, rezulta ca functia de eroare la un moment dat are drept argumente, totalitatea ponderilor si a factorilor de scala ai retelei.

Conditia realizarii unei antrenari corecte se poate exprima din punct de vedere matematic prin relatia (3.10):

eroarea=g(tp,f(w,bias,inp))=0 (3.10)

Solutiile unor ecuatii de tipul (3.10) nu pot fi determinate exact (datorita complexitatii si neliniaritatii functiilor f si g), ci doar aproximativ, prin algorimi numerici iterativi. Acesti algoritmi permit reactualizarea ponderilor si a factorilor de scala din aproape in aproape, eroarea tinzând spre zero. Acesta este practic si dezideratul antrenarii: pentru ca reteaua sa corespunda cât mai mult cerintelor pentru care a fost proiectata trebuie ca functia de eroare sa tinda spre zero, iar in cazul ideal sa fie chiar zero.

Dintre metodele iterative clasice se disting: metoda gradientului, metoda combinata Newton-gradient, metoda Fridman, ultimele doua fiind versiuni "imbunatatite" ale primei. Ghilimelele din propozitia precedenta exprima faptul ca doar in cazuri particulare, de obicei nedetectabile, rezultatele procesului de antrenare sunt mai bune in sensul unor erori mai mici. Acesta este de altfel si motivul pentru care in cadrul tezei va fi prezentata pe larg doar metoda gradientului (paragraful 3.1.2.2). Trebuie precizat faptul ca, in multe cazuri aceste metode nu conduc la rezultate satisfacatoare. Motivatiile acestei aprecieri vor fi expuse pe larg in paragraful 3.1.2.3.

Retelele neuronale feedforward pot fi implementate atât software prin simularea comportarii lor pe calculatoare numerice, cât si hardware prin utilizarea unor cipuri specializate.

In cazul implementarii lor in varianta software, unul dintre principalele avantaje ale retelelor neuronale - procesarea paralela - este practic anulat datorita faptului ca rularea programelor se realizeaza pe calculatoare numerice secventiale. Astfel, timpul de

calcul creste foarte mult, cu repercursiuni atât in ceea ce priveste timpul de calcul caracteristic unei retele antrenate (cu ponderi si factori de scala fixate), cât si in ceea ce priveste durata fazei de antrenament. Totusi, o astfel de implementare permite studiul asupra celei mai avantajoase structuri (arhitecturi) a retelei si a celei mai avantajoase metode de antrenare, uneori putându-se constitui chiar intr-o faza anterioara implementarii hardware, daca rezultatele obtinute in urma antrenarii sunt satisfacatoare.

In cazul implementarii hardware a retelelor neuronale feedforward se constata o crestere exponentiala a vitezei de calcul fata de implementarea software, cu cât reteaua contine mai multi neuroni activi. Exista cipuri specializate pentru implementarea hardware a retelelor neuronale, cu ajutorul carora se pot proiecta retele neuronale feedforward cu sute de noduri de intrare, cu 4-5 nivele ascunse, având in total sute de mii de conexiuni.

3.1.2.2 METODA GRADIENTULUI

Conform concluziilor formulate de Cybenko [Cyb89], dorindu-se a fi prezentat un studiu cât mai general, aceasta metoda va fi expusa pentru cazul unei retele neuronale total conectate cu patru nivele (doua nivele ascunse).

Figura 3.8 prezinta schematic o astfel de retea.

In esenta metoda gradientului, prezentata pe larg in diverse lucrari de specialitate [Ich92][Mul90][Tod94][Wid92], realizeaza actualizarea ponderilor w[i][j] si a factorilor de scala bias[i] prin adaugarea la acestea a unor termeni proportionali cu gradientul functiei de eroare in raport cu parametrul (pondere sau bias) respectiv.

$$\begin{cases} w[i][j] = w[i][j] + \eta * \dfrac{\delta e^2}{\delta w[i][j]} \\ b[i] = b[i] + \eta * \dfrac{\delta e^2}{\delta b[i]} \end{cases} \qquad (3.11)$$

unde η este cunoscut sub denumirea de rata de invatare, rata a carei valoare poate influenta mult convergenta algoritmului de propagare inapoi.

Fundamental, deducerea regulilor de ajustare a ponderilor si a factorilor de scala pentru retele total conectate cu patru nivele nu pune probleme deosebite. De aceea, pentru economie de spatiu si

pentru a nu complica inutil prezentarea nu se vor mai descrie detaliat pasii deductiei pentru astfel de retele a regulilor de ajustare a ponderilor. Se vor prezenta numai formulele finale si algoritmii forward si backward.

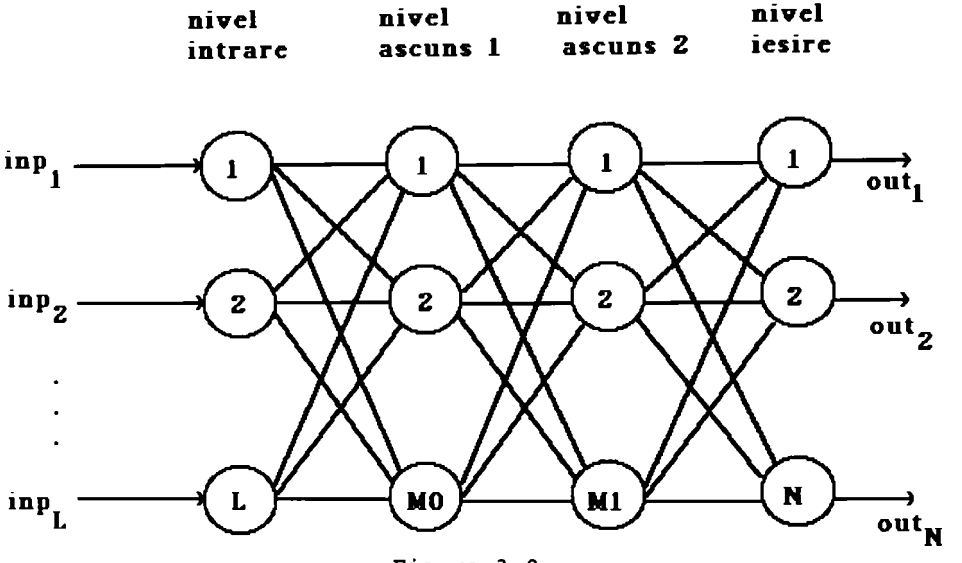


Figura 3.8

Informatiile necesare descrierii unei retele total conectate cu patru nivele (unul de intrare, doua ascunse si unul de iesire), prezentate intr-o simbolistica deja consacrata in domeniu, sunt:

L	- Numarul neuronilor de pe nivelul de intrare.
M0	- Numarul neuronilor de pe primul nivel ascuns.
M1	- Numarul neuronilor de pe al doilea nivel ascuns.
N	- Numarul neuronilor de pe nivelul de iesire.
out[N]	- Valorile iesirilor obtinute ale neuronilor de pe nivelul de iesire (vectorul iesirilor actuale sau obtinute ale retelei).
tp[N]	- Valorile dorite ale iesirilor neuronilor de pe nivelul de iesire (vectorul iesirilor prescrise ale retelei).
hidd1[M1]	- Valorile iesirilor neuronilor de pe al doilea nivel ascuns.
hidd0[M0]	- Valorile iesirilor neuronilor de pe primul nivel ascuns.

inp[L] - Valorile intrarilor neuronilor de pe stratul de intrare (vectorul intrarilor retelei). Deoarece neuronii de pe acest strat sunt inactivi ele coincid cu iesirile neuronilor de pe stratul de intrare.

wohi[N][M1] - Matricea ponderilor conexiunilor dintre nivelul de iesire si al doilea nivel ascuns.

bohi[N] - Deplasarile de scala pentru neuronii din nivelul de iesire.

wh10[M1][M0] - Matricea ponderilor conexiunilor dintre nivelele ascunse.

bh10[M1] - Deplasarile de scala pentru neuronii din al doilea nivel ascuns.

whin[M0][L] - Matricea ponderilor conexiunilor dintre primul nivel ascuns si nivelul de intrare.

bhin[M0] - Deplasarile de scala pentru neuronii din primul nivel ascuns.

η - Rata de invatare.

Valorile starilor neuronilor net (suma ponderata a tuturor intrarilor, inaintea aplicarii functiei de activare) pentru fiecare nivel sunt date de relatia generica (3.1):

Starile neuronilor de pe nivelul de iesire:

$$neto[i] = \sum_{j=0}^{M1-1} wohi[i][j]*hidd1[j]+bohi[i] \qquad (3.12)$$

Starile neuronilor de pe al doilea nivel ascuns:

$$neth1[j] = \sum_{k=0}^{M0-1} wh10[j][k]*hidd0[k]+bh10[j] \qquad (3.13)$$

Starile neuronilor de pe primul nivel ascuns:

$$neth0[k] = \sum_{l=0}^{L-1} whin[k][l]*inp[l]+bhin[k] \qquad (3.14)$$

$i=0...N-1,\ j=0...M1-1,\ k=0...M0-1$

Functia de eroare poate fi definita in diverse moduri, cea mai des intilnita fiind data de relatia urmatoare:

$$e = \sum_{i=0}^{N-1} (tp[i]-out[i])^2 \qquad (3.15)$$

Dupa cum se observa eroarea este considerata a fi una patratica medie.

Pentru deducerea relatiilor de ajustare a ponderilor si a factorilor de scala ai neuronilor de pe un anumit nivel este necesara exprimarea functiei de eroare in raport cu ponderile si factorii de scala ai nivelului neuronal respectiv. Pentru aceasta expresia lui out[i] poate fi dezvoltata ca fiind numai functie de tp, inp, ponderi si factori de scala. Astfel, pot fi obtinute formulele de calcul ale derivatei de ordinul unu (ale gradientului) functiei de eroare in raport cu oricare dintre ponderile si factorii de scala ai retelei:

$$\frac{\delta e}{\delta wohi[i][j]} = -2*(tp[i]-out[i])*F'(neto[i])*hidd1[j]$$

$$\frac{\delta e}{\delta bohi[i]} = -2*(tp[i]-out[i])*F'(neto[i]) \qquad (3.16)$$

$i=0...N-1,\ j=0...M1-1$

$$\frac{\delta e}{\delta wh10[j][k]} = -2*\sum_{i=0}^{N-1} (tp[i]-out[i])*F'(neto[i])*wohi[i][j]* *F'(neth1[j])*hidd0[k] \qquad (3.17)$$

$$\frac{\delta e}{\delta bh10[j]} = -2*\sum_{i=0}^{N-1} (tp[i]-out[i])*F'(neto[i])*wohi[i][j]* *F'(neth1[j])$$

$j=0...M1-1,\ k=0...M0-1$

$$\frac{\delta e}{\delta whin[k][l]} = -2*\sum_{i=0}^{N-1}(tp[i]-out[i])*F'(neto[i])*\sum_{j=0}^{M1-1}wohi[i][j]* *F'(neth1[j])*wh10[j][k]*F'(neth0[k])*inp[l] \qquad (3.18)$$

$$\frac{\delta e}{\delta bhin[k]} = -2*\sum_{i=0}^{N-1}(tp[i]-out[i])*F'(neto[i])*\sum_{j=0}^{M1-1}wohi[i][j]*$$

*F'(neth1[j])*wh10[j][k]*F'(neth0[k])

k=0...M0-1, l=0...L-1

Dupa cum se observa, functia de activare F trebuie sa fie derivabila si continua tocmai datorita aparitiei derivatei in formulele deduse din metoda gradientului. In cazurile prezentate prin relatiile (3.4)-(3.6) functiile de activare respective indeplinesc conditia de derivabilitate, ca de altfel orice functie sigmoidala. In continuare au fost considerate cazurile unor functii de activare mai des utilizate, pentru care pot fi deduse foarte simplu, formulele derivatelor acestora:

$$F_1(x)=\frac{1}{1+e^{-x/t}} \tag{3.19}$$

$$F_2(x)=\frac{1}{1+e^{-x/t}}-\frac{1}{2} \tag{3.20}$$

$$F_3(x)=C_1\tanh(C_{2x}) \tag{3.21}$$

cu derivatele corespunzatoare:

$$F'_1(x)=\frac{1}{t}F_1(x)(1-F_1(x)) \tag{3.22}$$

$$F'_2(x)=\frac{1}{t}[1-F_2(x)F_2(x)] \tag{3.23}$$

$$F'_3(x)=C_2\left(C_1+\frac{F_3^2(x)}{C_1}\right) \tag{3.24}$$

In dezvoltarile ce urmeaza este luata in considerare doar functia de activare F(x)=tanh(x), având derivata exprimata prin (3.24) luând $C_1=C_2=1$.

Prin considerarea relatiilor (3.16), (3.17), (3.18) pot fi dezvoltati algoritmii forward() si backward():

Algoritm forward

forward() /*descrierea fluxului neuronal dinspre intrare*/

```
            /*spre iesire                                          */
{
   /* inp --> hidd0 */
     k=0...M0-1

                 L-1
       neth0[k]=Σ whin[k][l]*inp[l]+bhin[k]
                 l=0

     hidd0[k]=F(neth0[k])
   /* hidd0 --> hidd1 */
     j=0...M1-1
                 M0-1
       neth1[j]=Σ wh10[j][k]*hidd0[k]+bh10[j]
                 k=0

     hidd1[j]=F(neth1[j])


   /* hidd1 --> out */
     i=0...N-1
                M1-1
       neto[i]=Σ wohi[i][j]*hidd1[j]+bohi[i]
                j=0

     out[i]=F(neto[i])
}
```

Algoritm backward

```
backward()    /* descrierea propagarii inapoi a erorii    */
              /* si ajustarea ponderilor si a factorilor */
              /* de scala                                 */
{
   /* out --> hidd1 */
     i=0...N-1, j=0...M1-1
     deltaout[i]=(tp[i]-out[i])*F'(neto[i])
     wohi[i][j] += η*deltaout[i]*hidd1[j]
     bohi[i]    += η*deltaout[i]
   /* hidd1 --> hidd0 */
     j=0...M1-1, k=0...M0-1

                  N-1
       delta10[j]=Σ deltaout[i]*wohi[i][j]*F'(neth1[j])
                  i=0

     wh10[j][k] += η*delta10[j]+hidd0[k]
     bh10[j]    += η*delta10[j]
   /* hidd0 --> inp */
     k=0...M0-1, l=0...L-1
```

$$deltahin[k] = \sum_{j=0}^{M1-1} delta10[j] * wh10[j][k] * F'(neth0[k])$$

```
        whin[k][l]  += η*deltahin[k]*inp[l]
        bhin[k]     += η*deltahin[k]
    }
```

3.1.2.3 DIFICULTATI APARUTE IN ANTRENAREA RETELELOR NEURONALE FOLOSIND ALGORITMUL BACKPROPAGATION

Amploarea pe care au luat-o experimentarile retelelor neuronale in cele mai diverse domenii de activitate au depasit cu mult studiul teoretic al acestora. Astfel, se constata ca aparatul matematic la care se face apel in utilizarea algoritmului de antrenare backpropagation nu este suficient de bine pus la punct, astfel ca nu pot fi diferentiate cu certitudine problemele pentru care aplicarea acestui algoritm duce la rezultate certe si problemele pentru care rezultatele sunt incerte. Datorita complexitatii arhitecturii retelelor neuronale, teorii matematice globale pentru acestea sunt greu, daca nu imposibil de dezvoltat. Se pune, totusi problema identificarii dificultatilor ce pot sa apara in antrenarea retelelor neuronale, pentru a putea fi pe cât posibil evitate.

In continuare vor fi prezentate succint principalele dificultati pe care algoritmul backpropagation le introduce in cadrul procesului de antrenare:

1) *Blocarea antrenarii retelei,* datorata formei sigmoidale a functiei de activare. Acest inconvenient mai poate fi intilnit in literatura de specialitate sub denumirea improprie de "paralizarea retelei" [Cu94][Tod94]

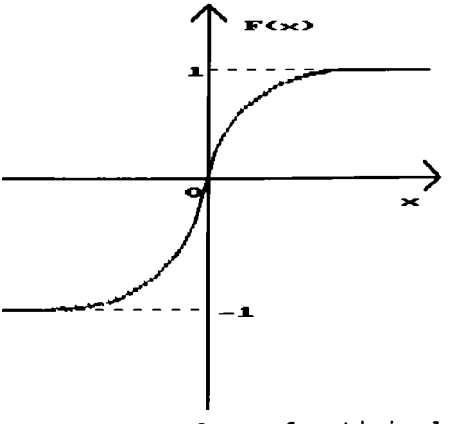


Figura 3.9: Alura functiei de activare

Pe durata procesului de antrenare, starile neuronilor (net) pot ajunge sa ia valori fie foarte mari, fie foarte mici, fapt ce determina, in urma aplicarii functiei de activare, iesiri (ou') -1- neuronilor foarte apropiate de asimptote. In aceste cazuri derivatele F' ce apar in formulele de ajustare a ponderilor si a factorilor de scala (3.16)-(3.18), vor fi practic nule. Acest fapt determina o blocare a invatarii, ponderile si

factorii de scala ramânând practic constante, cu toate ca functia de eroare nu este minimizata.

Daca in timpul antrenarii retelei neuronale apare acest fenomen, procesul de antrenare trebuie intrerupt si reluat, pornind din alte conditii initiale (ponderi si factori de scala).

2) *Minimul local [Cu94][Tod94]*

Functia de eroare descrisa de relatia (3.15) se poate considera ca o functie având ca parametrii totalitatea ponderilor si a factorilor de scala ai retelei, in numar de n. Aceasta functie poate fi reprezentata intr-un spatiu n+1 - dimensional sub forma unei suprafete.

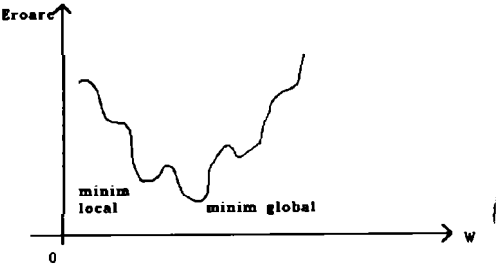


Figura 3.10

Scopul procesului de antrenare devine, intr-o astfel de reprezentare grafica, atingerea minimului global, sau, eventual, când eroarea dorita are o valoare mai mare, a unui minim local corespunzator.

Metoda gradientului ajusteaza ponderile si factorii de scala "coborând" sistemul-retea neuronala pe suprafata functiei de eroare spre un minim. Aceasta suprafata este neregulata si prezinta o multime de minime locale, in procesul antrenarii reteaua putându-se impotmoli intr-un astfel de minim local si antrenarea esueaza.

Dezvoltarea logica de mai sus prezinta si ea lucrurile foarte simplificat, intrucât aceasta suprafata nu este fixa in decursul antrenarii, ea putându-si modifica substantial alura.

Si in acest caz trebuie luata decizia repornirii procesului de antrenare in alte conditii initiale, chiar daca rezultatele vor fi si de aceasta data incerte.

3) *Utilizarea in calcule a erorii instantanee*

Metoda gradientului realizeaza o estimare on-line a ponderilor si nu una off-line cum ar fi de dorit. Acest lucru se datoreaza faptului ca este minimizata o eroare instantanee si nu una globala pentru intreaga epoca de antrenare fapt ce duce uneori la erori importante. O metoda de estimare off-line pentru antrenarea unei retele neuronale cu doua straturi ascunse este aproape imposibil de realizat. Tot bazat pe acest considerent apare fenomenul de "uitare".

Antrenarea se face pe un set de perechi de antrenament, practic in doi pasi - forward si backward. Astfel, in pasul forward se ia pe rând fiecare pereche de antrenament (inp;tp), se determina valoarile output corespunzatoare intrarilor inp, apoi se determina valoarea functiei de eroare si apoi, in pasul backward, se

realizeza ajustarea ponderilor conform metodei iterative folosite, in scopul minimizarii functiei de eroare. Procedându-se astfel, sansele obtinerii rezultatelor dorite sunt mici in cazul in care baza de date de antrenament contine un numar prea mare de perechi de antrenament, reteaua "invatând" ultimele perechi si "uitându-le" pe primele.

O solutie pentru evitarea situatiilor dezastruoase in care se poate ajunge in urma antrenarii cu algoritmul de propagare inapoi ar fi combinarea sa cu anumite metode euristice.

3.1.2.4 METODA DE ANTRENARE A RETELELOR NEURONALE

Paragraful de fata isi propune expunerea unei noi metode destinate antrenarii automate a retelelor neuronale. Metoda propune câteva imbunatatiri euristice asupra metodei gradientului, fapt ce elimina necesitatea implicarii factorului uman in procesul de antrenare. Cu toate dezvoltarile aduse in ultimul timp, teoria retelelor neuronale mai contine câteva semne de intrebare, mai ales in ceea ce priveste robusteatea antrenarii lor. In cele ce urmeaza este propus un algoritm de antrenare off-line pentru retele neuronale total conectate.

3.1.2.4.1 STRUCTURA ADOPTATA PENTRU RETEAUA NEURONALA

Metoda de antrenament propusa considera o structura a retelei neuronale destul de generala: un strat de intrare cu L neuroni, doua straturi ascunse cu M0 si respectiv M1 neuroni [Cyb89], un strat de iesire cu N neuroni (figura 3.11). Valorile corespunzatoare lui L, M0, M1 si N trebuiesc precizate de catre proiectant.

Metoda de antrenare clasica - metoda gradientului, prezinta câteva dezavantaje:

1) necesitatea precizarii unor valori initiale pentru ponderi si factori de scala, a caror alegere poate fi decisiva;
2) necesitatea precizarii coeficientului de antrenare η [Tod94];
3) necesitatea semnalizarii in cazul aparitiei unui minim local sau in cazul blocarii invatarii pentru a reporni procesul de antrenare in alte conditii initiale [Cu94] [Tod94];

Aceste dezavantaje implica, o antrenare asistata de operatorul uman, intuitia, chiar norocul acestuia, jucând un rol important in obtinerea unor rezultate satisfacatoare. Tocmai necesitatea inlaturarii interventiei operatorului uman a stat la baza dezvoltarii unei noi metode de antrenare, metoda ce se bazeaza pe urmatoarea consideratie: esuarea unui ciclu de antrenare (datorata ajungerii intr-un minim local sau a fenomenului de blocare a antrenarii) trebuie sa conduca automat la o modificarea a conditiilor initiale (ponderi, factori de scala, η) si la reluarea

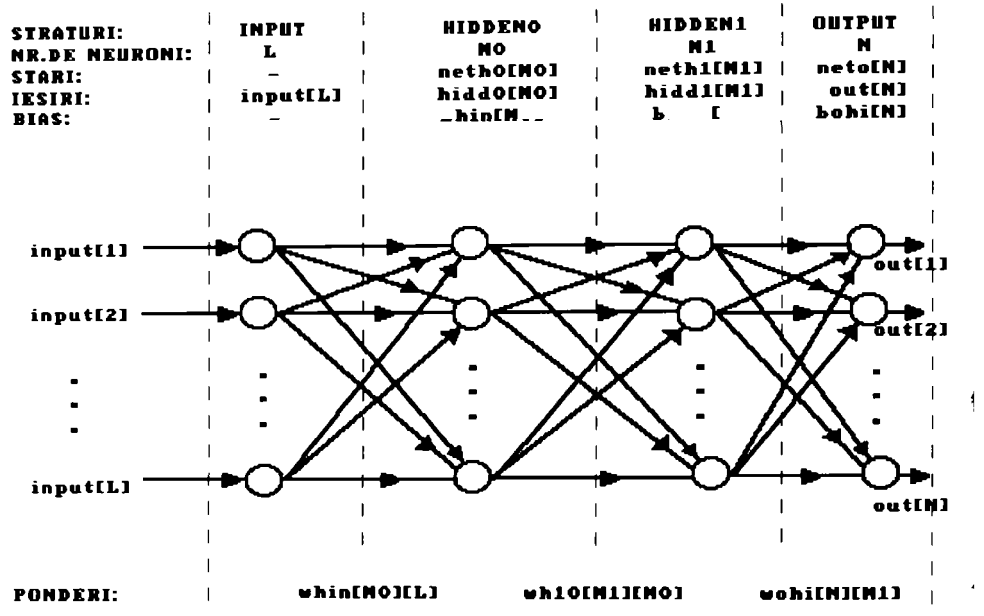


Figura 3.11

automata a procesului de antrenare.

3.1.2.4.2 PREZENTAREA METODEI

Algoritmul de antrenare ce face obiectul prezentului paragraf este descris prin urmatorii pasi:

1. Declaratii variabile;

```
/*ponderi:whin[][],wh10[][],wohi[][];*/
/*factori de scala:bhin[],bh10[],     */
/*bohi[][]; stari ale neuronilor:     */
/*neth0[],neth1[],neto[]; iesiri ale */
/*neuronilor:hidd1[],hidd0[], out[]; */
/*intrari:input[];                    */
/*iesiri prescrise:tp[]; erori        */
```

2. Initializari variabile;

```
/*L,M0,M1,N, perioada de esantionare*/
```

```
                /*h, capetele intervalului de timp  */
                /*pe care se face antrenarea tmin si*/
                /*tmax, eroarea de antrenare dorita */
                /*(er_dorita)                       */

3. while (er_obtinuta > er_dorita)
   {
4.   initializare ponderi si factori de scala;
               /* cu valori aleatoare cuprinse intre -1 si 1*/
5.   initializarea coeficientului de antrenare η;
               /* cu valori aleatoare intre 0.01 si 0.00001*/
6.   citeste eroarea minima obtinuta până la momentul actual
     (er_buna) din fisierul ponderi.bun;
7.   for(;;)                  /* bucla infinita */
     {
8.     for(t=tmin;t<tmax;t+=h) /*incepe o epoca de antrenare*/
       {
9.     actualizarea valorilor pentru intrarile RN input[];

10.    actualizarea valorilor pentru iesirile prescrise tp[];

11.    feedforward;/* calculul valorilor starilor si       */
                   /*iesirilor tuturor neuronilor din retea*/

12.      backward;/*recalcularea ponderilor si a factorilor*/
                  /*de scala (antrenarea propriu-zisa prin */
                  /*metoda gradientului)                   */
       }
13.    calculul erorii de antrenare er_obtinuta;
                  /*se realizeaza prin reluarea pasilor 8,9,*/
                  /*10 si 11, ponderile si factorii de scala*/
                  /*fiind considerate constante             */

14.    if (er_obtinuta < er_buna)
       {
15.      scrie ponderi, factori de scala si valoarea erorii
         obtinute in fisierul ponderi.bun;
       }
16.    if (er_obtinuta > er_ob_v)
17.      break;       /* parasirea buclei infinite     */
18       else
         {
19.       er_ob_v=er_obtinuta;
                  /* memorarea erorii obtinte pentru*/
                  /* un viitor pas 16               */
         }
       }             /* inchiderea buclei infinite     */
   }                 /* inchiderea buclei while        */
```

In continuare vor fi dezvoltate doar partile ce apar in plus in acest algoritm in raport cu metoda clasica a gradientului:

Pasul 4. In acest pas ponderile si factorii de scala sunt initializate ori de câte ori este necesar, cu valori cuprinse intre -1 si 1 utilizându-se o functie de generare a numerelor aleatoare uniform distribuite (functia random() din biblioteca math.h a compilatorului C al firmei Borland). Valorile limita -1 si 1 au fost alese in urma unor experimentari indelungate, tinând cont de urmatoarele considerente: a) valori mai mici decât -2 si mai mari decât 2 conduc cu o probabilitate foarte mare la fenomenul de blocare a invatarii, b) este necesara reducerea ariei de incercare a ponderilor pentru micsorarea timpului total de antrenare, tinând cont si de faptul ca in urma utilizarii metodei gradientului, aceste valori pot iesi mult din intervalul initial [-1, 1].

Pasul 5. Este analog pasului 4. cu precizarea ca tot din considerente experimentale se solicita ca η sa ia valori cuprinse in intervalul (0.01, 0.00001).

Pasul 6. Eroarea minima obtinuta este citita din fisierul ponderi.bun, fisier ce contine ponderile, factorii de scala si eroarea minima pentru cazul cel mai favorabil intilnit in decursul antrenarii. Daca ulterior se obtine o eroare mai redusa, fisierul ponderi.bun este reactualizat in pasul 15. Salvarea parametrilor retelei in cazul cel mai favorabil din punct de vedere al erorii obtinute in fisierul ponderi.bun mai are avantajul ca antrenarea retelei poate fi reluata oricând.

Pasul 16. In acest pas este verificat faptul daca s-a ajuns intr-un minim local sau a aparut fenomenul de blocare a invatarii. Cazurile in care datorita valorii destul de mari a lui η (rezulta o "aruncare") reteaua paraseste minimul local explorându-se astfel o alta zona din hiperspatiul functiei de eroare s-a considerat ca nu trebuiesc luate in considerare fiind relativ putine, aceasta si din cauza ca printr-o reluare a pasilor 4 si 5 se poate intimplator ajunge direct in acea zona a hiperspatiului functiei de eroare.

Pentru a creste gradul de automatizare a antrenarii retelei neuronale, algoritmul poate fi dezvoltat prin considerarea numarului de neuroni de pe straturile intermediare ca fiind variabil. Modificarile necesare sunt minime:

a) in pasul 2 se considera M=M1=M0=3 (considerarea aceluiasi numar de neuroni pe straturile ascunse nu micsoreaza gradul de generalitate al algoritmului);

b) se introduc intre pasii 3 si 4 ai algoritmului urmatorii pasi:

```
3.1 contor=0;    /* initializarea valorii contorului destinat*/
                 /* numararii ciclurilor de antrenament      */
                 /* realizate cu o valoare constanta a lui M */

3.2 M++;         /* incrementarea numarului de neuroni de pe */
                 /* straturile intermediare                  */

3.2 for (;;)     /* bucla infinita                           */
```

```
    {           /* aceasta bucla se inchide dupa pasul 19,   */
                /*imediat inainte de inchiderea buclei while*/
3.3   contor++; /* incrementarea contorului                  */

3.4   if (contor > f(M)) /* se testeaza valoarea contorului.*/
                         /* f(M) este o functie exponentiala*/
                         /* ce depinde de numarul de neuroni*/
                         /* de pe straturile intermediare   */

3.5     break;      /*iesirea din bucla infinita realizata*/
                    /* prin pasul 3.2                      */


19
    }
```

Prezenta metoda a fost utilizata pentru antrenarea unor retele neuronale total conectate dând rezultate foarte bune. Totusi, aceasta metoda nu poate fi utilizata in antrenarea on-line a retelelor neuronale, deoarece durata procesului de antrenare este destul de lunga (chiar zile), dar cu rezultate certe. In anexa 3.1 este prezentat programul antrenam.c ce implementeaza aceasta metoda.

3.2. SOLUTIE PENTRU MODELAREA SI PREDICTIA SERIILOR DE TIMP UTILIZÂND RETELE NEURONALE RECURENTE

Acest subcapitol isi propune prezentarea unei metodologii originale destinate modelarii si predictiei off-line a seriilor de timp utilizând retele neuronale recurente. Concluziile prezentate in finalul paragrafului sunt intarite prin studii de caz sugestive.

Analiza si predictia seriilor de timp reprezinta un domeniu relativ nou, cu o dezvoltare dinamica in ultimii ani si cu aplicatii in toate domeniile de activitate. Metodologiile care s-au impus in scopul modelarii si predictiei seriilor de timp sunt destul de numeroase, dar gradul de incredere in rezultatele obtinute depinde sensibil de natura seriilor de timp analizate.

Pentru a fi utilizata in scopul modelarii si predictiei seriilor de timp, reteaua neuronala trebuie sa tina cont de valorile temporale anterioare ale iesirilor in furnizarea noilor valori pentru iesiri. Acest tip de retea este numita Retea Neuronala Recurenta (RNR).

3.2.1. RETELE NEURONALE RECURENTE

Retelele neuronale recurente au la baza retelele neuronale feedforward cu particularitatea ca prezinta drept "intrari"

valorile intârziate ale iesirilor [Dia92][Tod94][Tsu93] (figura 3.12).

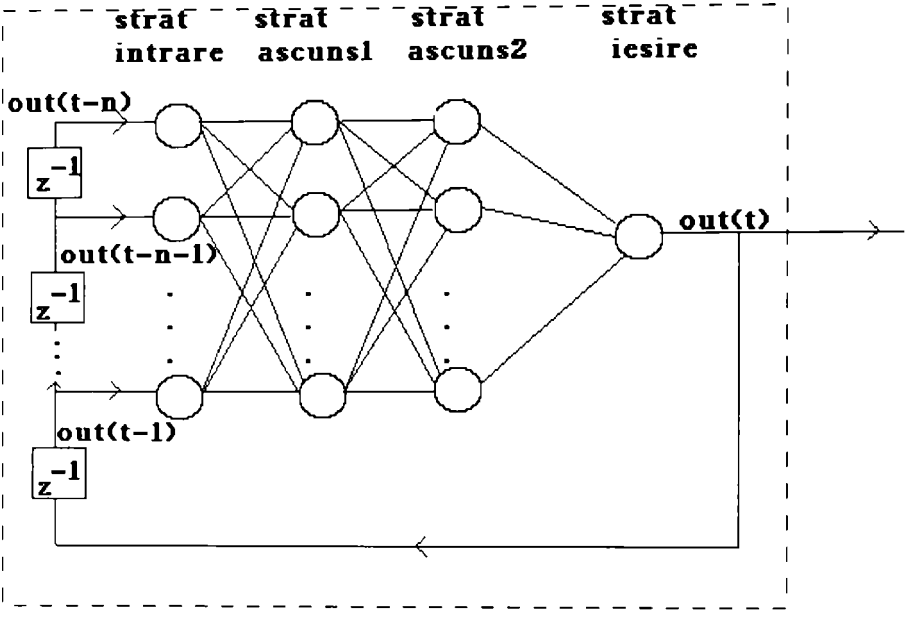


Figura 3.12

Astfel, termenul de "intrari" devine impropriu, aceste semnale devenind practic semnale interne ale RNR, fapt pentru care modelarea cu RNR a seriilor de timp devine naturala (figura 3.13). Acest tip de retea neuronala prezinta feedback, deci memorie.

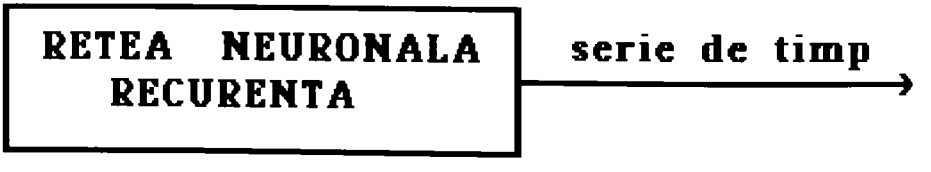


Figura 3.13

O structura generala de RNR utilizabila pentru modelarea seriilor de timp contine: un strat de intrare (input level) la care sunt conectate valori anterioare ale iesirilor, maxim doua straturi

ascunse [Cyb89] (hidden levels), un strat de iesire (output level) ce va furniza valorile curente ale marimilor de intrare. In figura 3.12 este prezentata o retea cu o singura marime (semnal) de iesire. De obicei pentru antrenare este utilizat algoritmul de propagare inapoi clasic, dar datorita neajunsurilor explicate pe larg intr-un paragraf anterior se recomanda utilizarea unui algoritm de antrenare asemanator celui propus in cadrul prezentei teze de doctorat (paragraful 3.1.2.4).

Antrenarea off-line a RNR este realizata prin simularea algoritmului de propagare inapoi pe un calculator numeric. Pentru aceasta trebuiesc a fi parcursi urmatorii pasi:

a) alegerea numarului de "intrari" a RNR, adica a numarului de valori anterioare ale iesirii aduse la intrarea retelei;
b) alegerea numarului de neuroni de pe fiecare strat ascuns tinând cont de compromisul precizie-complexitate;
c) alegerea functiei de activare;
d) alegerea ponderilor si a factorilor de scala initiale (acest punct nu se regaseste daca este utilizata metodologia din paragraful 3.1.2.4);
e) alegerea tehnicii de antrenare;
f) antrenarea propriu-zisa a RNR.

Functionarea retelei neuronale recurente in urma antrenarii ei se bazeaza pe impunerea unei stari initiale dorite prin fortarea "intrarilor" RNR la valori impuse (de care s-a tinut cont in procesul antrenarii), precum si pe existenta unor ponderi si factori de scala fixate. Astfel, la inceputul functionarii in regim normal, "intrarile" RNR sunt fortate la valorile impuse, fluxul neuronal fiind propagat automat spre iesirile RNR, unde se vor obtine valorile dorite. Aceste valori sunt aplicate mai apoi la "intrarile" RNR si procesul se repeta.

Programul, realizat in limbajul C, permite salvarea intr-un fisier pe disc (ponderi.bun) a ponderilor si a factorilor de scala, in cazul obtinerii erorii minime. Seria de timp ce se doreste a fi modelata este citita de pe disc din fisierul date.int.

3.2.1.1 STUDIU ASUPRA ALEGERII ERORII MAXIME IN PROCESUL DE ANTRENARE A RNR

Un aspect care poate produce esuarea procesului de antrenare este descrisa relativ la un exemplu:

Sa presupunem ca se doreste antrenarea unei RNR pentru a furniza la iesire functia sin(t) pentru t luând valori intre 0 si $\pi/2$.

Pentru aceasta trebuie a fi conectata la intrarea RNR o singura valoare intârziata a iesirii acesteia: sin(t-h), unde h este perioada de esantionare (figura 3.14). Daca h are o valoare redusa, atunci, practic, sin(t)=sin(t-h) si reteaua poate avea

tendinta sa "invete" functia identica y=f(u)=u. In acest caz, chiar daca in timpul antrenarii eroarea are o valoare redusa, iesirea in conditiile normale de functionare nu este sin(t).

De exemplu, daca h=0.01 eroarea patratica globala pentru domeniul de timp $[0,\pi/2]$ este:

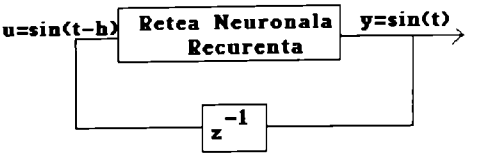


Figura 3.14

$$err=\frac{1}{N+1}\sum_{i=0}^{N}(\sin(i)-\sin(i-h))^2=0.000051 \qquad (3.25)$$

unde N este cel mai mic intreg sub $\pi/(2*h)$.

Deci, pentru acest exemplu trebuie considerata ca satisfacatoare in procesul de antrenare o eroare globala sub 0.4*err.

Daca problema mentionata mai sus apare in antrenare se recomanda modificarea fie a valoarii erorii globale dorite (prin micsorarea valorii acesteia), fie a perioadei de esantionare h (prin marirea acesteia).

3.2.2 PROGRAME IN LIMBAJELE C SI SIMNON. STUDII DE CAZ

Programele de antrenare realizate in limbajul C si programele de simulare a retelelor neuronale antrenate realizate in limbajul Simnon vor fi prezentate integral doar pentru primul studiu de caz, pentru celelalte fiind prezentate doar valorile ponderilor corespunzatoare retelelor antrenate.

3.2.2.1 STUDIU DE CAZ - ANTRENAREA UNEI RETELE NEURONALE RECURENTE PENTRU OBTINEREA UNEI SERII DE TIMP SINUSOIDALE F(X)=SIN(X) IN PRIMUL CADRAN.

S-a considerat ca serie de timp functia sinus in primul cadran, iar pentru modelarea ei a fost utilizata o RNR cu 1 neuron pe stratul de intrare, 3 neuroni pe fiecare din cele doua straturi ascunse si 1 neuron pe stratul de iesire. Starea initiala a retelei neuronale recurente antrenate mai contine pe linga ponderi si factori de scala (fixate in urma procesului de antrenare) si starea initiala a "intrarii" RNR, in acest caz 0.

```
/*************************************************************/
/* PROGRAM PENTRU ANTRENAREA UNEI RETELE NEURONALE RECURENTE*/
/*************************************************************/


#include <math.h>
#include <stdio.h>
#include <conio.h>
#include <time.h>
#include <stdlib.h>

 /* ---------declaratii variabile globale----------- */

 int i,j,k,l,n,nr_ep;
 float out[3],tp[3],hidd1[20],hidd0[20];
 float input[3];
 float wohi[3][20],bohi[3];
 float wh10[20][20],bh10[20],whin[20][3];
 float bhin[20],neth0[20],neth1[20];
 float neto[3],deltaout[3],delta10[20];
 float deltahin[20],eroarea[3];
 float t,eroarea_glob,err;
 float err_v,err_n;
 FILE *fisier,*fp,*fisdate;

  /* ---------initializari--------------- */

 int L=1;          /* numar neuroni pe stratul de intrare    */
 int M0=3;         /* numar neuroni pe primul strat ascuns   */
 int M1=3;         /* numar neuroni pe al doilea strat ascuns*/
 int N=1;          /* numar neuroni pe stratul de iesire     */
 float h=0.05;     /* perioada de esantionare                */
 float tmin=0.;    /* valoarea minima a timpului             */
 float tmax=1.5;   /* valoarea maxima a timpului             */
 float ita;        /* coeficient modificare ponderi          */
 int nr_epoci=10; /* numarul de epoci de antrenare          */
 float er_buna=1000000;
 int nr_ep=0;
 float err_satisf=0.00005;

/* ----initializarea ponderilor cu valori aleatoare---- */

void rand_param()
 {
  randomize();
  printf("\n reinitializare ponderi \n");
  ita random(1000)/100000.;
  for(i 0;i<N;i++)
   {
     for(j 0;j<M1;j++)
        wohi[i][j]=(random(10000)-5000.)/5000;
```

```
      bohi[i]=(random(10000)-5000.)/5000;
    }
   for(i=0;i<M1;i++)
    {
      for(j=0;j<M0;j++)
         wh10[i][j]=(random(10000)-5000.)/5000;
       bh10[i]=(random(10000)-5000.)/5000;
     }
   for(i=0;i<M0;i++)
     {
        for(j=0;j<L;j++)
           whin[i][j]=(random(10000)-5000.)/5000;
        bhin[i]=(random(10000)-5000.)/5000;
      }
 }

/* ---citirea valorii erorii din fisierul ponderi.bun--- */

void cit_eroare()
 {
  fisier=fopen("ponderi.bun","r");
  fscanf(fisier,"%f",&err);
  printf("\nerr=%f\n",err);
  fclose(fisier);
 }

/* ---------------Programul Principal------------------- */

void main()
 {
  clrscr();
  while (er_buna>err_satisf)
   {
    rand_param();
    cit_eroare();
    er_buna=err;
    err_v=1000000;

    for(;;)    /* bucla infinita */
     {
     input[0]=0;
     fisdate=fopen("date.int","r");
     for(t=tmin;t<tmax;t+=h)
      {
       fscanf(fisdate,"%f\n",&tp[0]);

       /*---------------- forward -----------------*/

     for (i=0;i<M0;i++)
      {
        neth0[i]=0;
```

```
    for (j=0;j<L;j++)
        neth0[i]+=whin[i][j]*input[j];
     neth0[i]+=bhin[i];
     hidd0[i]=tanh(neth0[i]);
  }
 for (i=0;i<M1;i++)
  {
    neth1[i]=0;
    for (j=0;j<M0;j++)
        neth1[i]+=wh10[i][j]*hidd0[j];
    neth1[i]+=bh10[i];
    hidd1[i]=tanh(neth1[i]);
  }
 for (i=0;i<N;i++)
  {
    neto[i]=0;
    for (j=0;j<M1;j++)
         neto[i]+=wohi[i][j]*hidd1[j];
     neto[i]+=bohi[i];
     out[i]=tanh(neto[i]);}


   /* ---------------- backward ----------------*/

 for (i=0;i<N;i++)
  {
    deltaout[i]=(tp[i]-out[i])*(1+
                tanh(neto[i])*tanh(neto[i]));
    for (j=0;j<M1;j++)
         wohi[i][j]+=ita*deltaout[i]*hidd1[j];
    bohi[i]+=ita*deltaout[i];
  }
 for (j=0;j<M1;j++)
  {
    delta10[j]=0;
    for (i=0;i<N;i++)
         delta10[j]+=deltaout[i]*wohi[i][j]
                     *(1+tanh(neth1[j])*tanh(neth1[j]));
    for (k=0;k<M0;k++)
         wh10[j][k]+=ita*delta10[j]*hidd0[k];
    bh10[j]+=ita*delta10[j];
  }
 for (k=0;k<M0;k++)
  {
     deltahin[k]=0;
     for (j=0;j<M1;j++)
         deltahin[k]+=delta10[j]*wh10[j][k]*
                      (1+tanh(neth0[k])*tanh(neth0[k]));
     for (l=0;l<L;l++)
          whin[k][l]+=ita*deltahin[k]*input[l];
      bhin[k]+=ita*deltahin[k];
```

```
    }

 input[0]=out[0];
}   /*   timp t   */
fclose(fisdate);

/* urmeaza calculul erorii in functionarea normala*/
/* (fara invatare)                                */

 for(i=0;i<N;i++)
    eroarea[i]=0.;
    eroarea_glob=0.;

 input[0]=0;
 fisdate=fopen("date.int","r");
 for(t=tmin;t<tmax;t+=h)
 {
   fscanf(fisdate,"%f\n",&tp[0]);

  /* ----------------- forward -----------------*/

  for (i=0;i<M0;i++)
   {
     neth0[i]=0;
     for (j=0;j<L;j++)
          neth0[i]+=whin[i][j]*input[j];
      neth0[i]+=bhin[i];
      hidd0[i]=tanh(neth0[i]);
   }
  for (i=0;i<M1;i++)
   {
       neth1[i]=0;
       for (j=0;j<M0;j++)
          neth1[i]+=wh10[i][j]*hidd0[j];
        neth1[i]+=bh10[i];
        hidd1[i]=tanh(neth1[i]);
    }
  for (i=0;i<N;i++)
    {
      neto[i]=0;
      for (j=0;j<M1;j++)
          neto[i]+=wohi[i][j]*hidd1[j];
       neto[i]+=bohi[i];
       out[i]=tanh(neto[i]);
       eroarea[i]+=(out[i]-tp[i])*(out[i]-tp[i]);
   }
  input[0]=out[0];
  }
 fclose(fisdate);
 for (i=0;i<N;i++)
        eroarea_glob+=eroarea[i];
```

```
      err_n=eroarea_glob;
      if (eroarea_glob<er_buna)
       {
         er_buna=eroarea_glob;
         nr_ep=n;
          fisier=fopen("ponderi.bun","w");
          /* scrierea ponderilor in fisierul ponderi.bun*/
          fprintf(fisier,"%f eroarea minima",er_buna);
          for(i=0;i<N;i++)
           {
             for(j=0;j<M1;j++)
                fprintf(fisier,"\n
                        wohi[%d][%d]=%f",i,j,wohi[i][j]);
             fprintf(fisier,"\n bohi[%d]=%f",i,bohi[i]);
           }
         for(i=0;i<M1;i++)
           {
             for(j=0;j<M0;j++)
                fprintf(fisier,"\n
                        wh10[%d][%d]=%f",i,j,wh10[i][j]);
             fprintf(fisier,"\n bh10[%d]=%f",i,bh10[i]);
           }
         for(i=0;i<M0;i++)
           {
             for(j=0;j<L;j++)
                fprintf(fisier,"\n
                        whin[%d][%d]=%f",i,j,whin[i][j]);
             fprintf(fisier,"\n bhin[%d]=%f",i,bhin[i]);
           }
       fclose(fisier);
      }   /* de la if */
      if (err_n>err_v) break;
}                /* trecerea la o noua initializare */
                 /* a parametrilor                  */
      else err_v=err_n;
   }   /* bucla infinita  */
}      /* while           */
}      /*   main          */
```

LISTAREA FISIERULUI PONDERI.BUN. Acest fisier contine ponderile pentru care eroarea medie patratica pentru o epoca este minima. Fisierul este generat automat prin programul C.

```
0.0009254 eroarea minima
wohi[0][0]=-0.016280
wohi[0][1]=-1.165363
wohi[0][2]=0.726803
bohi[0]=0.961678
wh10[0][0]=0.485785
```

```
wh10[0][1]=-0.274344
wh10[0][2]=0.067509
bh10[0]=-0.050699
wh10[1][0]=-0.062463
wh10[1][1]=-0.365259
wh10[1][2]=1.545909
bh10[1]=0.302796
wh10[2][0]=0.334588
wh10[2][1]=-0.070676
wh10[2][2]=-0.933110
bh10[2]=0.767345
whin[0][0]=-0.149939
bhin[0]=0.252094
whin[1][0]=0.669958
bhin[1]=-0.500885
whin[2][0]=-1.101163
bhin[2]=0.800268
```

PACHETUL DE PROGRAME SIMNON UTILIZAT IN SIMULAREA RETELEI NEURONALE RECURENTE FORMAT DIN FISIERELE:

- R1.T - fisier ce descrie functionarea retelei neuronale recurente, si
- RET1.T - fisier macro pentru simulare si afisare grafice

FISIERUL R1.T

```
discrete system r1
state whin11 whin21 whin31
state wh1011 wh1012 wh1013
state wh1021 wh1022 wh1023
state wh1031 wh1032 wh1033
state wohi11 wohi12 wohi13
state bohi1 bh101 bh102 bh103 bhin1 bhin2 bhin3
new nwhin11 nwhin21 nwhin31
new nwh1011 nwh1012 nwh1013
new nwh1021 nwh1022 nwh1023
new nwh1031 nwh1032 nwh1033
new nwohi11 nwohi12 nwohi13
new nbohi1 nbh101 nbh102 nbh103 nbhin1 nbhin2 nbhin3
output out1
time t
tsamp ts
initial
whin11:-0.149939
whin21:0.669958
whin31:-1.101163
```

```
wh1011:0.485785
wh1012:-0.274344
wh1013:0.067509
wh1021:-0.062463
wh1022:-0.365259
wh1023:1.545909
wh1031:0.334588
wh1032:-0.070676
wh1033:-0.933110
wohi11:-0.01628
wohi12:-1.165363
wohi13:0.726803
bohi1:0.961678
bh101:-0.050699
bh102:0.302796
bh103:0.767345
bhin1:0.252094
bhin2:-0.500885
bhin3:0.800268
sort
inp1=sin(t)
tp1=sin(t+h)
neth01=whin11*inp1+bhin1
neth02=whin21*inp1+bhin2
neth03=whin31*inp1+bhin3
hidd01=tanh(neth01)
hidd02=tanh(neth02)
hidd03=tanh(neth03)
neth11=wh1011*hidd01+wh1012*hidd02+wh1013*hidd03+bh101
neth12=wh1021*hidd01+wh1022*hidd02+wh1023*hidd03+bh102
neth13=wh1031*hidd01+wh1032*hidd02+wh1033*hidd03+bh103
hidd11=tanh(neth11)
hidd12=tanh(neth12)
hidd13=tanh(neth13)
neto1=wohi11*hidd11+wohi12*hidd12+wohi13*hidd13+bohi1
out1=tanh(neto1)
e1=tp1-out1
dout1=(tp1-out1)*(1+tanh(neto1)*tanh(neto1))
nwohi11=wohi11+ita*dout1*hidd11
nwohi12=wohi12+ita*dout1*hidd12
nwohi13=wohi13+ita*dout1*hidd13
nbohi1=bohi1+ita*dout1
delta101=dout1*nwohi11*(1+tanh(neth11)*tanh(neth11))
delta102=dout1*nwohi12*(1+tanh(neth12)*tanh(neth12))
delta103=dout1*nwohi13*(1+tanh(neth13)*tanh(neth13))
nwh1011=wh1011+ita*delta101*hidd01
nwh1012=wh1012+ita*delta101*hidd02
nwh1013=wh1013+ita*delta101*hidd03
nwh1021=wh1021+ita*delta102*hidd01
nwh1022=wh1022+ita*delta102*hidd02
nwh1023=wh1023+ita*delta102*hidd03
```

```
nwh1031=wh1031+ita*delta103*hidd01
nwh1032=wh1032+ita*delta103*hidd02
nwh1033=wh1033+ita*delta103*hidd03
nbh101=bh101+ita*delta101
nbh102=bh102+ita*delta102
nbh103=bh103+ita*delta103
a1=delta101*nwh1011+delta102*nwh1021+delta103*nwh1031
dhin1=a1*(1+tanh(neth01)*tanh(neth01))
a2=delta101*nwh1012+delta102*nwh1022+delta103*nwh1032
dhin2=a2*(1+tanh(neth02)*tanh(neth02))
a3=delta101*nwh1013+delta103*nwh1023+delta103*nwh1033
dhin3=a3*(1+tanh(neth03)*tanh(neth03))
nwhin11=whin11+ita*dhin1*inp1
nwhin21=whin21+ita*dhin2*inp1
nwhin31=whin31+ita*dhin3*inp1
nbhin1=bhin1+ita*dhin1
nbhin2=bhin2+ita*dhin2
nbhin3=bhin3+ita*dhin3
ts=t+h
ita:0.
h:0.05
end
```

FISIERUL RET1.T

```
macro ret1
syst r1
store  out1[r1] tp1[r1] inp1[r1] e1[r1]
split 1 1
simu 0 1.5 /rasp
"export exp<rasp
"save date
suspend
ashow out1[r1] tp1[r1]
text '   REFERINTA-tp1[r1] SI IESIREA RETELEI-out1[r1]'
"hcopy
suspend
ashow e1[r1]
text '   EROAREA-e1[r1]'
"hcopy
suspend
end
```

REZULTATELE OBTINUTE SUNT PREZENTATE SUB FORMA GRAFICA:

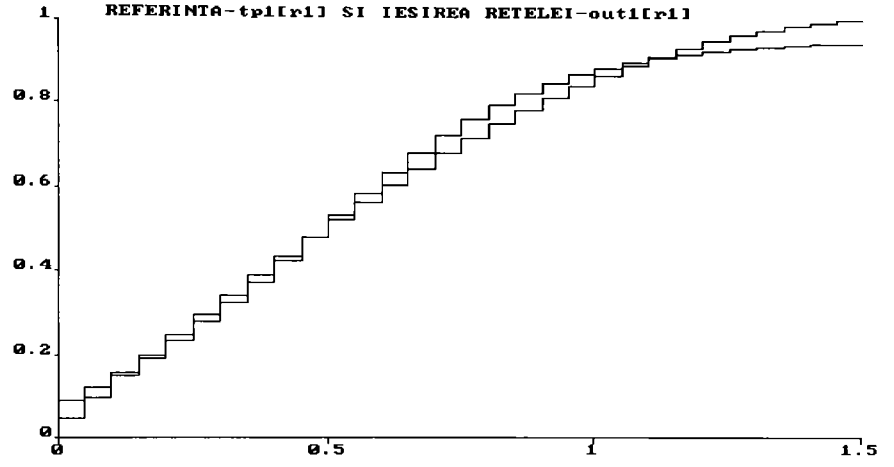


Figura 3.15

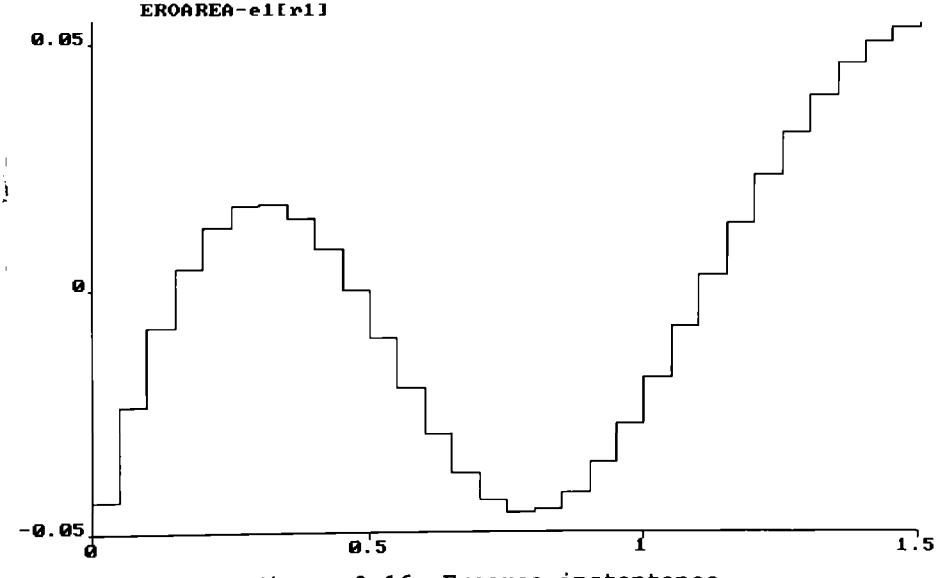


Figura 3.16: Eroarea instantanee

3.2.2.2 STUDIU DE CAZ - ANTRENAREA UNEI RETELE NEURONALE RECURENTE PENTRU OBTINEREA UNEI SERII DE TIMP COSINUSOIDALE F(X)=COS(X) IN PRIMELE DOUA CADRANE.

S-a considerat ca serie de timp functia cosinus in primele doua cadrane, iar pentru modelarea ei a fost utilizata o RNR cu 1 neuron pe stratul de intrare, 6 neuroni pe fiecare din cele doua straturi ascunse si 1 neuron pe stratul de iesire. Starea initiala a "intrarii" RNR este 1.

LISTAREA FISIERULUI PONDERI.BUN, fisier ce contine ponderile pentru care eroarea medie patratica pentru o epoca este minima. Fisierul este generat automat prin programul C.

```
0.001080 eroarea minima
 wohi[0][0]=-0.711798
 wohi[0][1]=1.173831
 wohi[0][2]=-1.048825
 wohi[0][3]=0.398274
 wohi[0][4]=-0.518770
 wohi[0][5]=-0.503454
 bohi[0]=0.740462
 wh10[0][0]=-0.233370
 wh10[0][1]=-0.200286
 wh10[0][2]=0.955808
 wh10[0][3]=-0.598142
 wh10[0][4]=0.180408
 wh10[0][5]=-0.424944
 bh10[0]=-0.168463
 wh10[1][0]=1.275047
 wh10[1][1]=0.527427
 wh10[1][2]=0.491485
 wh10[1][3]=0.654840
 wh10[1][4]=0.348748
 wh10[1][5]=-1.402770
 bh10[1]=-1.414733
 wh10[2][0]=-0.358541
 wh10[2][1]=-0.500463
 wh10[2][2]=0.398516
 wh10[2][3]=-0.268310
 wh10[2][4]=0.076379
 wh10[2][5]=0.225912
```

```
bh10[2]=-0.816472
wh10[3][0]=0.644708
wh10[3][1]=0.726405
wh10[3][2]=0.349430
wh10[3][3]=-0.661847
wh10[3][4]=0.735241
wh10[3][5]=0.116445
bh10[3]=0.606634
wh10[4][0]=-0.191750
wh10[4][1]=0.656350
wh10[4][2]=0.972625
wh10[4][3]=-1.060552
wh10[4][4]=-0.659771
wh10[4][5]=-0.578638
bh10[4]=-0.523971
wh10[5][0]=-0.944026
wh10[5][1]=-0.386125
wh10[5][2]=-0.412203
wh10[5][3]=0.127730
wh10[5][4]=-0.507248
wh10[5][5]=1.082265
bh10[5]=0.484339
whin[0][0]=3.187458
bhin[0]=-2.468420
whin[1][0]=1.369371
bhin[1]=-1.135052
whin[2][0]=-0.534147
bhin[2]=-0.304866
whin[3][0]=1.462106
bhin[3]=0.883381
whin[4][0]=0.326309
bhin[4]=-0.689669
whin[5][0]=-1.785554
bhin[5]=2.011390
```

REZULTATELE OBTINUTE SUNT PREZENTATE SUB FORMA GRAFICA:

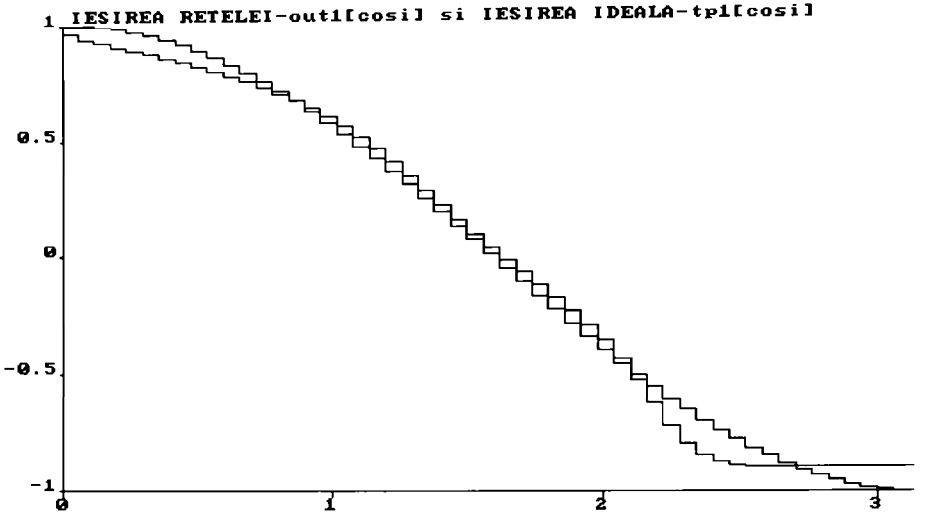


Figura 3.17

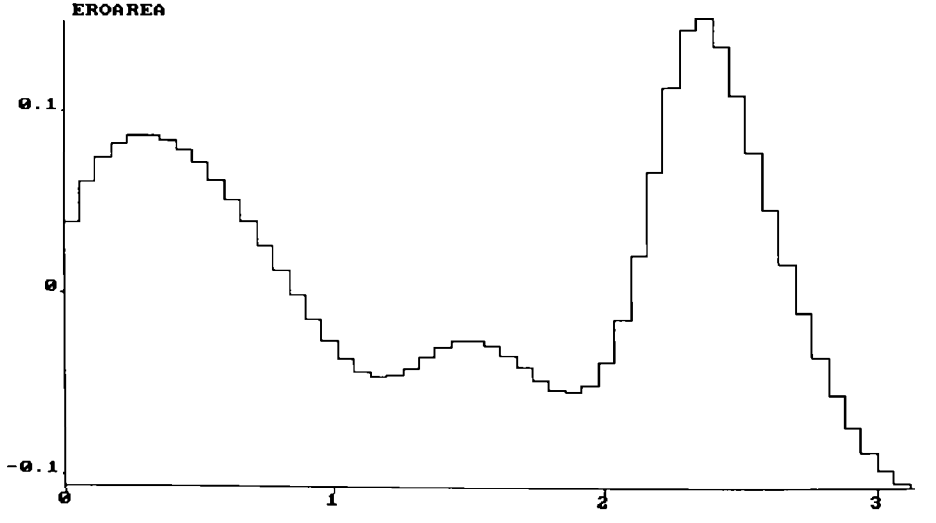


Figura 3.18: Eroarea instantanee

3.2.2.3 STUDIU DE CAZ - ANTRENAREA UNEI RETELE NEURONALE RECURENTE PENTRU OBTINEREA UNEI SERII DE TIMP AMORTIZATE

S-a considerat seria de timp din figura 3.19, iar pentru modelarea ei a fost utilizata o RNR cu 1 neuron pe stratul de intrare, 6 neuroni pe fiecare din cele doua straturi ascunse si 1 neuron pe stratul de iesire. Starea initiala a "intrarii" RNR: -1. In urma antrenarii s-a obtinut o eroare zero, adica seria initiala si cea modelata coincid.

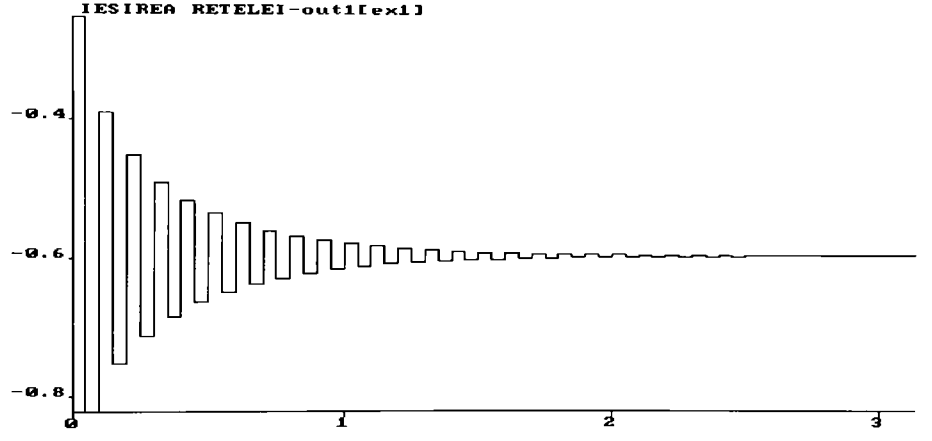


Figura 3.19

LISTAREA FISIERULUI PONDERI.BUN, fisier ce contine ponderile pentru care eroarea medie patratica pentru o epoca este minima. Fisierul este generat automat prin programul C.

```
0.000000 eroarea minima
wohi[0][0]=-0.104348
wohi[0][1]=0.501431
wohi[0][2]=-1.053551
wohi[0][3]=0.384478
wohi[0][4]=-0.531096
wohi[0][5]=-0.514386
bohi[0]=-0.224505
wh10[0][0]=-0.230586
```

```
wh10[0][1]=-0.197521
wh10[0][2]=-0.953647
wh10[0][3]=-0.599667
wh10[0][4]=0.181398
wh10[0][5]=0.422063
bh10[0]=0.165396
wh10[1][0]=1.260298
wh10[1][1]=0.512774
wh10[1][2]=-0.502942
wh10[1][3]=-0.646766
wh10[1][4]=0.343509
wh10[1][5]=-1.387504
bh10[1]=-1.398483
wh10[2][0]=-0.341976
wh10[2][1]=-0.484003
wh10[2][2]=0.411391
wh10[2][3]=-0.277373
wh10[2][4]=0.082258
wh10[2][5]=0.208764
bh10[2]=-0.834726
wh10[3][0]=0.634803
wh10[3][1]=0.716565
wh10[3][2]=0.341738
wh10[3][3]=-0.656424
wh10[3][4]=0.731721
wh10[3][5]=0.126697
bh10[3]=0.617546
wh10[4][0]=-0.179611
wh10[4][1]=0.668411
wh10[4][2]=0.982060
wh10[4][3]=-1.067193
wh10[4][4]=-0.655463
wh10[4][5]=0.566072
bh10[4]=-0.137347
wh10[5][0]=-0.133196
wh10[5][1]=0.396886
wh10[5][2]=0.420621
wh10[5][3]=0.121805
wh10[5][4]=-0.503405
wh10[5][5]=-1.093476
bh10[5]=0.472405
whin[0][0]=-0.951944
bhin[0]=-2.397623
whin[1][0]=0.346920
bhin[1]=-1.112601
whin[2][0]=0.574313
bhin[2]=-0.345032
whin[3][0]=-1.464340
bhin[3]=-0.951944
whin[4][0]=-0.355509
bhin[4]=-0.660469
```

```
whin[5][0]=-1.745496
bhin[5]=-0.051448
```

3.2.2.4 STUDIU DE CAZ - ANTRENAREA UNEI RETELE NEURONALE RECURENTE PENTRU OBTINEREA UNEI SERII DE TIMP PERIODICE.

S-a considerat seria de timp din figura 3.20, iar pentru modelarea ei a fost utilizata o RNR cu 1 neuron pe stratul de intrare, 6 neuroni pe fiecare din cele doua straturi ascunse si 1 neuron pe stratul de iesire. Starea initiala a "intrarii" RNR: -1. In urma antrenarii, la fel ca si in studiul de caz anterior s-a obtinut o eroare zero, adica seria initiala si cea modelata coincid.

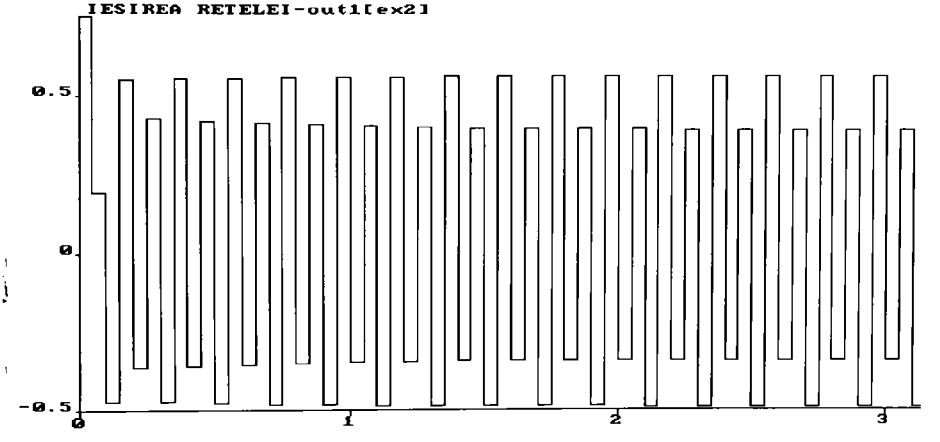


Figura 3.20

LISTAREA FISIERULUI PONDERI.BUN, fisier ce contine ponderile pentru care eroarea medie patratica pentru o epoca este minima. Fisierul este generat automat prin programul C.

```
0.000000 eroarea minima
wohi[0][0]=0.11798
wohi[0][1]=0.5173831
```

```
wohi[0][2]=-1.048825
wohi[0][3]=0.398274
wohi[0][4]=-0.518770
wohi[0][5]=-0.503454
bohi[0]=0.740462
wh10[0][0]=-0.233370
wh10[0][1]=-0.200286
wh10[0][2]=-0.955808
wh10[0][3]=-0.598142
wh10[0][4]=0.180408
wh10[0][5]=-0.424944
bh10[0]=0.168463
wh10[1][0]=1.275047
wh10[1][1]=0.527427
wh10[1][2]=-0.491485
wh10[1][3]=-0.654840
wh10[1][4]=0.348748
wh10[1][5]=-1.402770
bh10[1]=-1.414733
wh10[2][0]=-0.358541
wh10[2][1]=-0.500463
wh10[2][2]=0.398516
wh10[2][3]=-0.268310
wh10[2][4]=0.076379
wh10[2][5]=0.225912
bh10[2]=-0.816472
wh10[3][0]=0.644708
wh10[3][1]=0.726405
wh10[3][2]=0.349430
wh10[3][3]=-0.661847
wh10[3][4]=0.735241
wh10[3][5]=0.116445
bh10[3]=0.606634
wh10[4][0]=-0.191750
wh10[4][1]=0.656350
wh10[4][2]=0.972625
wh10[4][3]=-1.060552
wh10[4][4]=-0.659771
wh10[4][5]=-0.578638
bh10[4]=-0.123971
wh10[5][0]=-0.144026
wh10[5][1]=0.386125
wh10[5][2]=0.412203
wh10[5][3]=0.127730
wh10[5][4]=-0.507248
wh10[5][5]=-1.082265
bh10[5]=0.484339
whin[0][0]=3.187458
bhin[0]=-2.468420
whin[1][0]=0.369371
bhin[1]=-1.135052
```

```
whin[2][0]=0.534147
bhin[2]=-0.304866
whin[3][0]=-1.462106
bhin[3]=-0.883381
whin[4][0]=-0.326309
bhin[4]=-0.689669
whin[5][0]=-1.785554
bhin[5]=-0.011390
```

3.2.3. CONCLUZII ASUPRA MODELARII SERIILOR DE TIMP UTILIZÂND RETELE NEURONALE RECURENTE

Aceasta metodologie prezinta avantajul ca orice serie de timp, indiferent de natura ei poate fi modelata cu RNR, dezavantajul cel mai important decurgând din faptul ca pentru a se putea obtine precizii bune trebuiesc facute multe incercari asupra parametrilor retelei (ponderi initiale, factori de scala, numar de straturi ascunse, numar de neuroni pe fiecare strat, etc), incercari care coroborate cu problemele de minim local si de blocare a invatarii, pot conduce la perioade de antrenare foarte mari (ca durata). Aceste neajunsuri pot fi inlaturate, in cazul in care o antrenare off-line corespunde exigentelor practice, prin utilizarea metodei de antrenare prezentate in paragraful 3.1.2.4.

ANEXA 3.1: PROGRAMUL SURSA ANTRENAM.C

Este prezentat programul sursa in limbajul C destinat antrenarii unei retele neuronale total conectate cu trei intrari, doua straturi intermediare având câte 20 de neuroni si trei iesiri, prin algoritmul descris in paragraful 3.1.2.4. Baza de date de antrenament considerata pentru exemplificare este formata din urmatoarele functii de intrare input si de iesire prescrise tp:

```
input[0]=0.6+1.8072*t;
input[1]=(285-2228.9*t)/35;
input[2]=(148-698.79*t)/14;
tp[0]=0.6-4.9058*t-55.3571*t*t+0.0017*t*t*t;
tp[1]=0.4-4.77*t-130.8442*t*t+0.00263*t*t*t;
tp[2]=0.85+4.43*t-40.855*t*t-191*t*t*t;
```

Erorile obtinute in urma rularii programului pe durata a trei zile, fara intrerupere, pe un calculator PC-486 având o frecventa de tact de 100MHz, sunt insignifiante (0.01%).

```
/*****************************************************/
/* PROGRAM PENTRU ANTRENAREA UNEI RETELE NEURONALE */
/*****************************************************/

#include <math.h>
#include <stdio.h>
#include <conio.h>
#include <time.h>
#include <stdlib.h>

 /* ---------declaratii variabile globale----------- */

 int i,j,k,l,n,nr_ep;
 float out[3],tp[3],hidd1[20],hidd0[20];
 float input[3];
 float wohi[3][20],bohi[3];
 float wh10[20][20],bh10[20],whin[20][3];
 float bhin[20],neth0[20],neth1[20];
 float neto[3],deltaout[3],delta10[20];
 float deltahin[20],eroarea[3];
 float t,eroarea_glob,err;
 float err_v,err_n;
 FILE *fisier,*fp;

  /* ---------initializari--------------- */

 int L=3;          /* numar neuroni pe stratul de intrare    */
```

```
 int M0=20;        /* numar neuroni pe primul strat ascuns   */
 int M1=20;        /* numar neuroni pe al doilea strat ascuns*/
 int N=3;          /* numar neuroni pe stratul de iesire     */
 float h=0.002;    /* perioada de esantionare                */
 float tmin=0.;    /* valoarea minima a timpului             */
 float tmax=0.083;/* valoarea maxima a timpului              */
 float ita;        /* coeficient modificare ponderi          */
 int nr_epoci=10; /* numarul de epoci de antrenare           */
 float er_buna=1000000;
 int nr_ep=0;
 float err_satisf=0.00005;

/* ----initializarea ponderilor cu valori aleatoare---- */

void rand_param()
 {
  randomize();
  printf("\n reinitializare ponderi \n");
  ita=random(1000)/100000.;
  for(i=0;i<N;i++)
   {
     for(j=0;j<M1;j++)
        wohi[i][j]=(random(10000)-5000.)/5000;
     bohi[i]=(random(10000)-5000.)/5000;
   }
  for(i=0;i<M1;i++)
   {
     for(j=0;j<M0;j++)
        wh10[i][j]=(random(10000)-5000.)/5000;
      bh10[i]=(random(10000)-5000.)/5000;
   }
  for(i=0;i<M0;i++)
    {
       for(j=0;j<L;j++)
          whin[i][j]=(random(10000)-5000.)/5000;
       bhin[i]=(random(10000)-5000.)/5000;
    }
 }

/* ---citirea valorii erorii din fisierul ponderi.bun--- */

void cit_eroare()
 {
  fisier=fopen("ponderi.bun","r");
  fscanf(fisier,"%f",&err);
  printf("\nerr=%f\n",err);
  fclose(fisier);
 }

/* ---------------Programul Principal------------------- */
```

```
void main()
 {
  clrscr();
  while (er_buna>err_satisf)
   {
    rand_param();
    cit_eroare();
    er_buna=err;
    err_v=1000000;

    for(;;)    /* bucla infinita */
     {
     for(t=tmin;t<tmax;t+=h)
      {
       input[0]=0.6+1.8072*t;
       input[1]=(285-2228.9*t)/35;
       input[2]=(148-698.79*t)/14;
       tp[0]=0.6-4.9058*t-55.3571*t*t+0.0017*t*t*t;
       tp[1]=0.4-4.77*t-130.8442*t*t+0.00263*t*t*t;
       tp[2]=0.85+4.43*t-40.855*t*t-191*t*t*t;

       /*---------------- forward -----------------*/

     for (i=0;i<M0;i++)
      {
        neth0[i]=0;
        for (j=0;j<L;j++)
            neth0[i]+=whin[i][j]*input[j];
         neth0[i]+=bhin[i];
         hidd0[i]=tanh(neth0[i]);
      }
     for (i=0;i<M1;i++)
      {
        neth1[i]=0;
        for (j=0;j<M0;j++)
            neth1[i]+=wh10[i][j]*hidd0[j];
        neth1[i]+=bh10[i];
        hidd1[i]=tanh(neth1[i]);
      }
     for (i=0;i<N;i++)
      {
        neto[i]=0;
        for (j=0;j<M1;j++)
             neto[i]+=wohi[i][j]*hidd1[j];
         neto[i]+=bohi[i];
         out[i]=tanh(neto[i]);}

       /* ---------------- backward ----------------*/

     for (i=0;i<N;i++)
      {
```

```
      deltaout[i]=(tp[i]-out[i])*(1+
                  tanh(neto[i])*tanh(neto[i]));
      for (j=0;j<M1;j++)
           wohi[i][j]+=ita*deltaout[i]*hidd1[j];
      bohi[i]+=ita*deltaout[i];
    }
   for (j=0;j<M1;j++)
    {
      delta10[j]=0;
      for (i=0;i<N;i++)
           delta10[j]+=deltaout[i]*wohi[i][j]
                      *(1+tanh(neth1[j])*tanh(neth1[j]));
      for (k=0;k<M0;k++)
           wh10[j][k]+=ita*delta10[j]*hidd0[k];
      bh10[j]+=ita*delta10[j];
    }
   for (k=0;k<M0;k++)
    {
       deltahin[k]=0;
       for (j=0;j<M1;j++)
           deltahin[k]+=delta10[j]*wh10[j][k]*
                       (1+tanh(neth0[k])*tanh(neth0[k]));
       for (l=0;l<L;l++)
            whin[k][l]+=ita*deltahin[k]*input[l];
        bhin[k]+=ita*deltahin[k];
    }
}    /*   timp t   */

/* urmeaza calculul erorii in functionarea normala*/
/* (fara invatare)                                */

 for(i=0;i<N;i++)
    eroarea[i]=0.;
    eroarea_glob=0.;
 for(t=tmin;t<tmax;t+=h)
 {
   input[0]=0.6+1.8072*t;
   input[1]=(285-2228.9*t)/35;
   input[2]=(148-698.79*t)/14;
   tp[0]=0.6-4.9058*t-55.3571*t*t+0.0017*t*t*t;
   tp[1]=0.4-4.77*t-130.8442*t*t+0.00263*t*t*t;
   tp[2]=0.85+4.43*t-40.855*t*t-191*t*t*t;

  /* ----------------- forward -----------------*/

  for (i=0;i<M0;i++)
   {
     neth0[i]=0;
     for (j=0;j<L;j++)
          neth0[i]+=whin[i][j]*input[j];
      neth0[i]+=bhin[i];
```

```
        hidd0[i]=tanh(neth0[i]);
    }
  for (i=0;i<M1;i++)
    {
        neth1[i]=0;
        for (j=0;j<M0;j++)
           neth1[i]+=wh10[i][j]*hidd0[j];
         neth1[i]+=bh10[i];
         hidd1[i]=tanh(neth1[i]);
    }
  for (i=0;i<N;i++)
    {
      neto[i]=0;
      for (j=0;j<M1;j++)
          neto[i]+=wohi[i][j]*hidd1[j];
       neto[i]+=bohi[i];
       out[i]=tanh(neto[i]);
       eroarea[i]+=(out[i]-tp[i])*(out[i]-tp[i]);
    }
  }
for (i=0;i<N;i++)
       eroarea_glob+=eroarea[i];
err_n=eroarea_glob;
if (eroarea_glob<er_buna)
  {
    er_buna=eroarea_glob;
    nr_ep=n;
     fisier=fopen("ponderi.bun","w");
     /* scrierea ponderilor in fisierul ponderi.bun*/
     fprintf(fisier,"%f eroarea minima",er_buna);
     for(i=0;i<N;i++)
       {
         for(j=0;j<M1;j++)
            fprintf(fisier,"\n
                    wohi[%d][%d]=%f",i,j,wohi[i][j]);
         fprintf(fisier,"\n bohi[%d]=%f",i,bohi[i]);
       }
    for(i=0;i<M1;i++)
       {
         for(j=0;j<M0;j++)
            fprintf(fisier,"\n
                    wh10[%d][%d]=%f",i,j,wh10[i][j]);
         fprintf(fisier,"\n bh10[%d]=%f",i,bh10[i]);
       }
    for(i=0;i<M0;i++)
       {
         for(j=0;j<L;j++)
            fprintf(fisier,"\n
                    whin[%d][%d]=%f",i,j,whin[i][j]);
         fprintf(fisier,"\n bhin[%d]=%f",i,bhin[i]);
       }
```

```
      fclose(fisier);
     }   /* de la if */
     if (err_n>err_v) break; /*trecerea la o noua initializare */
                             /* a parametrilor                  */
     else err_v=err_n;
  }   /* bucla infinita */
}     /* while          */
}     /*   main         */
```

}

CAPITOLUL 4: SOLUTIE DE IMPLEMENTARE A UNEI TEHNICI DE CONDUCERE ADAPTIVA CU ELEMENTE NEURONALE DESTINATA CONDUCERII AGREGATELOR AEROELECTRICE.

In capitolul de fata este prezentata o structura de conducere adaptiva cu regulator autoacordabil, ce are la baza utilizarea elementelor neuronale. Aceasta solutie poate fi utilizata in conducerea unei game largi de procese, a caror modelare precisa este greu de realizat, categorie din care fac parte si agregatele aeroelectrice. Capitolul debuteaza printr-o prezentare sintetica a problematicii sistemelor adaptive (paragrafele 4.1, 4.2 si 4.3), urmând, mai apoi (paragraful 4.4) descrierea solutiei de conducere propuse.

4.1 INTRODUCERE IN TEMATICA SISTEMELOR ADAPTIVE

Dezvoltarea teoriilor uzuale de conducere a sistemelor se bazeaza in principal pe urmatoarele consideratii: a) sistemul, a carui structura este cunoscuta, poate fi modelat; si b) parametrii modelului sunt cunoscuti.

Aceste doua consideratii ridica numeroase probleme in practica conducerii agregatelor aeroelectrice.

Determinarea unui model viabil destinat conducerii unui agregat aeroelectric fixat implica eforturi sustinute in doua directii:

- determinarea modelului agregatului propriu-zis. Acest sistem dinamic este constituit din module cu un inalt grad de complexitate (rotorul, reductorul, generatorul asincron, cuplaje, frine si conectarea la retea). In literatura de specialitate [Kar93] [She91] [She93] [Nat87] [Des86] [Lei89] [Cu93_r1] au fost prezentate modele ale diverselor tipuri de agregate pentru anumite regimuri de functionare. Obtinerea unui model cu aplicabilitate generala,

pentru un agregat fixat, este aproape imposibil de realizat.
- determinarea modelului vântului. Vântul este un fenomen dinamic ce prezinta variatii relativ rapide ale vitezei si orientarii in raport cu timpul, variatii imposibil de surprins prin modele cu parametrii constanti [Cu92].

Se poate concluziona ca in conducerea agregatelor aeroelectrice apar probleme deosebite, date fiind caracteristicile procesului, care fac in multe cazuri imposibila utilizarea teoriei conducerii sistemelor liniare cu parametri constanti [Bab85] [Bel85] [Cal85] [Dra89] [IoV85] [Tud93]:
- parametri necunoscuti ai modelului agregatului si vântului;
- valoarea apreciabila a timpului mort, adesea variabila;
- comportarea puternic neliniara a procesului;
- caracteristici de transfer care variaza in timp, din cauza modificarilor parametrilor rotorului in functie de viteza vântului;
- "imbatrinirea", cu timpul, a partilor mecanice ale agregatului;
- actiunea unor perturbatii aleatoare (de exemplu, variatii ale temperaturii mediului ambiant);
- propagarea unor perturbatii cunoscute (vibratii) de-a lungul agregatului.

Aceste dificultati sunt cunoscute de mult timp, fiind propuse diferite solutii de rezolvare a problemei conducerii in situatiile mentionate anterior [And83] [Bon87] [Cha84] [Dem90] [Hoq93] [Lef85] [Mad91] [Sch84] [Sch86] [Sun83] [WaX84] [Wil90], prin utilizarea sistemelor adaptive. Aceste sisteme necesita informatie apriorica redusa si isi modifica structura si/sau parametrii in timpul functionarii utilizând informatia curenta furnizata de proces.

4.2. PRINCIPALELE TEHNICI DE CONDUCERE ADAPTIVA [Ast88] [Cal88] [Kau94]

Se considera o structura de reglare automata conventionala, cu particularitatea ca elementul de reglare (regulatorul) prezinta parametri ajustabili. Problema este de a gasi o cale convenabila de a modifica parametrii regulatorului ca raspuns la modificarile din proces si la dinamica perturbatiilor.

Conducerea adaptiva are o proprietate interesanta. Comanda nu va cauta doar sa determine ca iesirea sa urmareasca valorile dorite. Când parametrii sunt incerti, regulatorul va introduce de asemenea "perturbatii", care vor imbunatati estimarile si comenzile viitoare. Aceasta proprietate se numeste control dual.

Dificultatile asociate problemei de conducere adaptiva provin din doua surse. Prima consta in faptul ca instalatia reprezinta o "cutie neagra", la care numai intrarile si iesirile acesteia sunt disponibile in scopul conducerii. Cea de-a doua sursa generatoare de dificultati apare intr-o problema pur analitica, care consta in generarea legii de comanda adaptiva care sa asigure ca parametrii regulatorului, evoluând pentru conditii initiale arbitrare, vor converge catre valorile dorite si, in consecinta, semnalul de

eroare va tinde catre zero.

Trei metode pentru reglarea adaptiva a parametrilor, si anume, programarea câstigului, reglarea automata dupa model si regulatoare automate cu autoacordare, sunt descrise in cele ce urmeaza intru-un cadru unitar.

4.2.1. PROGRAMAREA CÂSTIGULUI

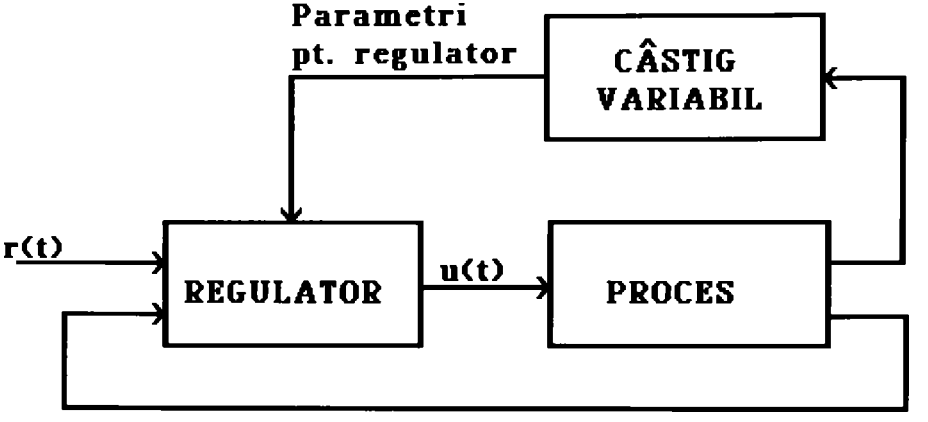


Figura 4.1 Schema bloc a unui sistem cu castig programat

Metoda poarta aceasta denumire deoarece, initial, ea a fost utilizata doar pentru a compensa modificarile coeficientului de amplificare al procesului.

La baza dezvoltarii acestei metode a stat idea ca, uneori este posibila gasirea de variabile auxiliare ale procesului care sunt intr-o buna corelatie cu modificarile din dinamica procesului. In aceste cazuri este posibila eliminarea influentelor variatiilor parametrilor prin modificarea parametrilor regulatorului in functie de variabilele auxiliare (Figura 4.1).

Programarea câstigului este realizata printr-o schema in bucla deschisa comparabila cu compensarea prin masurarea perturbatiei. Nu exista reactie negativa de compensare a programarii unui câstig incorect. Metoda are avantajul ca parametrii se pot modifica foarte rapid ca raspuns la o modificare in proces. Exista o controversa de terminologie in sensul daca programarea câstigului ar putea fi considerata sau nu ca o schema de adaptare, deoarece parametrii se modifica in bucla deschisa. Ignorând aceasta controversa, programarea câstigului este o tehnica foarte utila pentru a reduce efectele variatiilor parametrilor.

4.2.2. SISTEME ADAPTIVE CU MODEL ETALON (SAME).

In cazul conducerii adaptive cu model etalon ajustarea parametrilor regulatorului este realizata pe baza abaterii dintre iesirea procesului si iesirea modelului etalon. Modelul etalon (numit si model de referinta) este ales astfel incât sa reprezinte functionarea dorita a procesului.

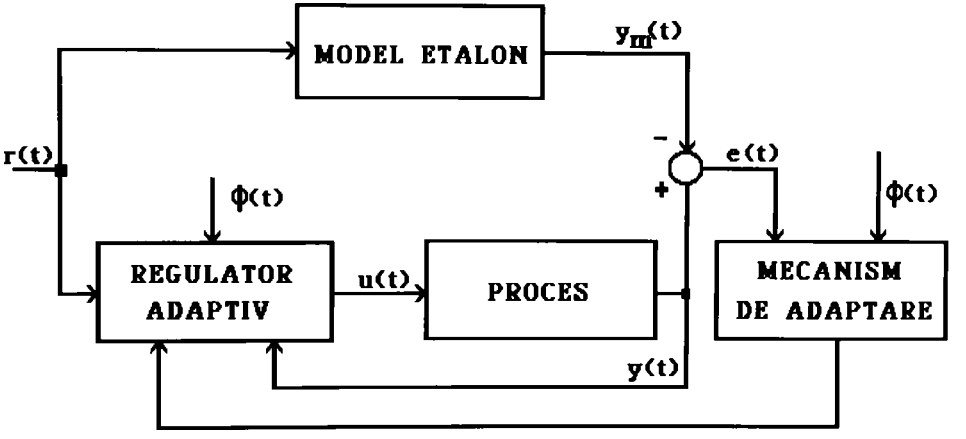


Figura 4.2 Schema bloc a unui sistem adaptiv cu model etalon

Principial, un sistem adaptiv cu model etalon are structura prezentata in figura 4.2. Intrarea de referinta r(t) este aplicata atât ansamblului regulator adaptiv - proces, cât si modelului etalon. Asupra procesului se considera ca actioneaza si unele perturbatii aditive. Regulatorul adaptiv are parametrii ajustabili, modificarea acestora fiind realizata prin intermediul mecanismului (strategiei) de adaptare astfel incât comanda u(t) rezultata sa actioneze in sensul reducerii sau chiar anularii erorii e(t) dintre iesirea procesului y(t) si iesirea modelului etalon $y_m(t)$. Mecanismul de adaptare a parametrilor regulatorului este descris prin legi sau algoritmi de ajustare, in functie de eroarea de urmarire, $e(t)=y(t)-y_m(t)$, si de alte semnale din proces disponibile, $\phi(t)$. Aceste legi sau algoritmi sunt de obicei neliniare si variante in timp.

In utilizarea SAME se presupune ca procesul are parametri constanti dar necunoscuti. Totusi, se poate admite ca parametrii sunt constanti pe intervale lungi de timp (comparativ cu dinamica procesului de adaptare), dar ca de la un interval la altul ei sufera modificari importante sub actiunea unor perturbatii

parametrice, de exemplu la schimbarea punctului de functionare. O variatie lenta in timp a parametrilor poate, de asemenea, a fi admisa.

Observatie: din categoria SAME face parte si reglarea modala alunecatoare, pentru care modelul etalon este prezentat analitic sub forma unor hipersuprafete.

4.2.3. REGULATOARE AUTOACORDABILE (RAA).

Structura de principiu a unui RAA (self-tuning regulator) este prezentata in figura 4.3. Se observa ca regulatorul adaptiv realizeaza inchiderea a doua bucle: bucla clasica, ce contine procesul si un regulator liniar cu reactie obisnuit; si bucla pentru ajustarea parametrilor regulatorului ce contine un bloc destinat identificarii on-line a parametrilor procesului si un bloc de calcul destinat actualizarii pe baza estimatiilor obtinute a coeficientilor unei legi de reglare cu o structura fixata. Regulatorul prezentat in figura 4.3 determina o comanda in sensul echivalentei certe, in terminologia conducerii stohastice, deoarece se neglijeaza faptul ca estimatiile parametrilor modelului reprezinta o aproximatie a parametrilor reali ai procesului. Se pot efectua modificari ale algoritmului de conducere, astfel incât sa se tina seama de gradul de incertitudine al parametrilor estimati (comanda prudenta), sau sa se introduca semnale externe pentru depasirea problemelor legate de identificabilitatea parametrilor modelului.

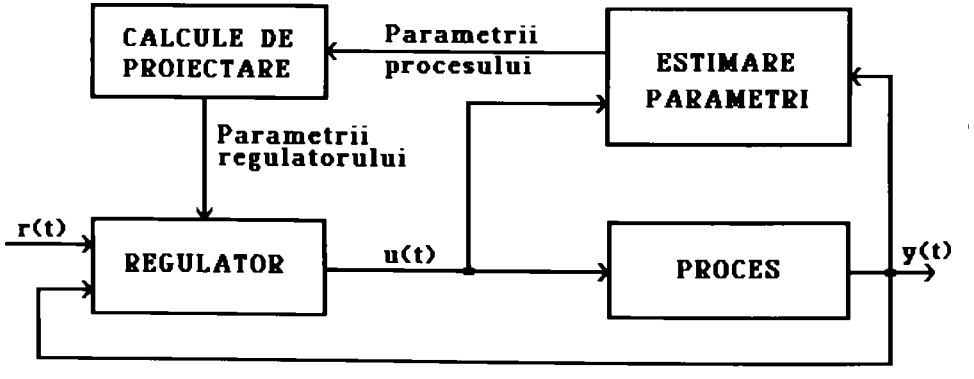


Figura 4.3 Schema bloc a unui sistem adaptiv cu regulator autoacordabil

RAA este foarte flexibil in ceea ce priveste metoda de proiectare, el determina, din datele de intrare-iesire, un model al procesului. Pâna in prezent s-au luat in considerare autoacordari bazate pe limite de faza si de amplitudine, plasarea polilor, controlul dispersiei minime si controlul linear cuadratic gaussian. Se pot folosi diferite scheme de estimare a parametrilor, ca de exemplu aproximarea stohastica, cele mai mici patrate, cele mai mici patrate extinse sau generalizate, variabile instrumentale, filtrare Kalman extinsa si metoda posibilitatii maxime, cu observatia ca trebuie asigurata robustetea estimarii in raport cu perturbatiile datorate preciziei finite cu care sunt realizate calculele. Regulatorul din figura poate fi dedus din abordarea SAME daca estimarea parametrilor se face prin actualizarea unui model de referinta.

Se poate deasemenea demonstra dualitatea sistemelor adaptive cu model etalon si cu regulatoarele autoacordabile [Cal88] [Kau94], fapt pentru care aceste structuri adaptive pot fi tratate unificat.

4.3 ASPECTE TEORETICE ALE CONDUCERII ADAPTIVE [Ast84] [Ast88] [Cal88] [Kau94] [Pro91_1] [Tao95]

In principiu, problema conducerii adaptive a unui proces insuficient cunoscut, supus actiunii perturbatiilor poate fi rezumata la determinarea marimii de comanda $u(t)$ la fiecare moment de timp t in scopul atingerii a doua obiective: stabilizarea sistemului in bucla inchisa si minimizarea cu probabilitate 1 a abaterii dintre marimea de iesire $y(t)$ si marimea de iesire dorita $y^*(t)$, indiferent de conditiile initiale. Stabilizarea sistemului in bucla inchisa se traduce prin asigurarea marginirii tuturor marimilor din sistem. Pe lânga obiectivele mentionate pot fi adaugate unele cerinte suplimentare, de implementare practica, cum ar fi:

a) pentru obtinerea lui $u(t)$ sa se utilizeze numai informatia intrare-iesire disponibila, $\{u(\tau), \tau<t\}$, $\{y(\tau), \tau<t\}$, si nu starea procesului $x(t)$. Totusi, daca starea procesului este masurabila atunci se recomanda utilizarea acesteia, atât pentru simplificarea posibila a schemei de conducere, cât si pentru eficientizarea procesului de conducere;

b) regulatorul adaptiv sa fie un sistem dinamic cauzal (neanticipativ), fara elemente de derivare, ceea ce inseamna ca, in regim stationar, regulatorul adaptiv are o matrice de transfer (strict) proprie;

c) parametrii regulatorului sa ia acele valori pentru care combinatia regulator-proces sa aiba o comportare intrare-iesire identica cu cea dorita. Aceasta cerinta este, de multe ori, intarita la aceea a consistentei estimatiilor, adica a convergentei estimatiilor spre valorile adevarate. Acest lucru necesita insa satisfacerea unor conditii de excitatie persistenta si de

identificabilitate a sistemului in bucla inchisa.

O importanta deosebita in dezvoltarea algoritmilor adaptivi o prezinta formarea vectorului semnalelor disponibile (numit uneori vectorul regresorilor sau vectorul masuratorilor), $\phi(t)$. In unele sisteme de conducere adaptiva discrete vectorul semnalelor disponibile $\phi(t)$, este constituit prin alaturarea vectorilor de comanda si de iesire anteriori momentului prezent. Alteori se utilizeaza valorile filtrate corespunzatoare. In cazul cel mai general, $\phi(t)$ reprezinta o estimatie a starii unui sistem dinamic condus de u, y si, eventual, de perturbatii. In acest caz sistemul adaptiv realizeaza estimarea simultana a parametrilor si a starii. In aceasta categorie se incadreaza observatoarele adaptive. Sistemele adaptive in care se identifica parametrii procesului $\theta(t)$ si se estimeaza starea acestuia x(t), estimatiile fiind folosite pentru generarea legii de conducere f(t), se numesc sisteme adaptive indirecte (sau explicite). In contrast, in sistemele adaptive directe (sau implicite) se estimeaza direct parametrii regulatorului (deci a lui f(t)). Pentru aceasta, se presupune cunoscuta forma modelului (tipul si ordinul) procesului si se defineste structura regulatorului care sa permita realizarea comportarii intrare-iesire dorite, in cazul in care parametrii procesului ar fi cunoscuti. Se identifica apoi parametrii regulatorului pe baza semnalului de eroare si a altor semnale disponibile ϕ si se utilizeaza parametrii identificati ca si cum ar fi exacti.

Daca se considera ca partea fixata este chiar procesul, iar sistemul ajustabil este un model ajustabil al procesului, atunci schema poate realiza identificarea procesului.

Pentru obtinerea solutiei problemei de conducere se apeleaza la anumite ipoteze simplificatoare asupra procesului si asupra zgomotului.

4.3.1 PROBLEMA CONDUCERII ADAPTIVE

Orice sistem monovariabil discret determinist, liniar si constant, finit dimensional, poate fi reprezentat printr-o relatie intrare-iesire de tipul:

$$A(q^{-1})y(t)=q^{-k}B(q^{-1})u(t) \quad , \quad t>0, \qquad (4.1)$$

unde: q^{-1} este operatorul de intirziere cu o perioada de tact; k este intirzierea ("timpul mort") exprimata in perioade de tact ($k\geq 1$) - in cazul sistemelor discretizate, datorita elementului de retinere de ordinul zero, avem $k=\tau+1$, unde τ este timpul mort (in tacte); $\{u(t)\}$ si $\{y(t)\}$ sunt sirurile valorilor marimii de intrare (comanda) si, respectiv, iesire; A(.) si B(.) sunt functii polinomiale in q^{-1}:

$$A(q^{-1}) = 1 + a_1 q^{-1} + \ldots + a_n q^{-n} \ , \qquad (4.2)$$

$$B(q^{-1}) = b_0 + b_1 q^{-1} + \ldots + b_m q^{-m} \ , \ b_0 \neq 0. \qquad (4.3)$$

Conditiile initiale ale ecuatiei (4.1) sunt exprimate prin valorile y(i), $-n<i\leq 0$, si u(i), $-k-m<i\leq -k$. Coeficientii a_i, b_j numiti parametri, vor fi in general presupusi necunoscuti.

Relatia (4.1) poate modela un proces fizic. Din punct de vedere teoretic se accepta ipoteza ca procesul are parametri constanti. Practic insa, sistemele adaptive admit si invarianta doar pe intervale limitate (dar "lungi") de timp, sau permit chiar variatia lenta a parametrilor, daca aceasta variatie este mult mai lenta decât "constantele de timp" ale procesului de adaptare.

Intrucât nu se presupune minimalitatea sistemului (4.1), acesta poate include poli necontrolabili, de exemplu poli proveniti de la perturbatii masurabile descrise de ecuatii de forma:

$$E(q^{-1})v(t) = 0 \ , \qquad (4.4)$$

adica perturbatii, inclusiv referinte, de tip polinomial, sinusoidal, etc. Este necesar sa se admita ca sistemul (4.1) este stabilizabil.

Se fac urmatoarele ipoteze:

Ipoteza 1 : Intârzierea k este cunoscuta.

Ipoteza 2 : Sunt cunoscute margini superioare pentru gradele n si m.

Ipoteza 3 : Polinomul B(z) are toate radacinile in afara discului unitate inchis din planul complex C.

Primele doua ipoteze sunt destul de generale si sunt introduse pentru ca in legea de comanda sa poata apare k, m si n. Ipoteza 3 reprezinta conditia de faza minima si inseamna ca sistemul invers (cu intrarea y si iesirea u) este stabil; aceasta asigura marginirea intrarii, daca iesirea este marginita. Renuntarea la ultima ipoteza este posibila si conduce la o clasa de sisteme adaptive pentru care exista doar rezultate de stabilitate locala. Se spune ca un polinom indeplinind conditia din ipoteza 3 este asimptotic stabil.

PROBLEMA DE CONDUCERE se formuleaza astfel: Sa se determine o lege de comanda cu reactie care sa stabilizeze sistemul (4.1) in sensul ca functiile u(t) si y(t) sa fie uniform marginite, si sa asigure atingerea obiectivelor de urmarire si de reglare de mai jos:

1. Obiectivul de urmarire - y(t) trebuie sa satisfaca relatia:

$$A_r(q^{-1})y(t) = q^{-k}B_r(q^{-1})r(t) \ , \ t>0, \qquad (4.5)$$

unde r(t) este un sir uniform marginit (intrarea de referinta), iar $A_r(.)$ si $B_r(.)$ sunt polinoame date, $A_r(q^{-1})$ fiind asimptotic stabil

si monic (adica $A_r(0)=1$).

2. Obiectivul de reglare - pentru $r(t)\equiv 0$ si in conditii initiale nenule, $y(t)$ trebuie sa satisfaca relatia:

$$C(q^{-1})y(t+k)=0 \ , \ t>0, \tag{4.6}$$

unde $C(q^{-1})$ este un polinom monic asimptotic stabil dat.

Daca parametrii sistemului nu sunt cunoscuti se foloseste un algoritm adaptiv.

PROBLEMA DE CONDUCERE ADAPTIVA se formuleaza astfel: Sa se determine o lege de comanda cu reactie astfel incât functiile $u(t)$ si $y(t)$ sa fie uniform marginite si sa se asigure obiectivul

$$\lim_{t\to\infty} C(q^{-1})e(t)=0 \tag{4.7}$$

oricare ar fi conditiile initiale ale procesului si ale algoritmului adaptiv, unde $e(t)$ este eroarea de urmarire:

$$e(t)=y(t)-y^*(t). \tag{4.8}$$

Se constata ca se pretinde ca sistemul adaptiv sa se comporte asimptotic ca un sistem de reglare proiectat in ipoteza parametrilor cunoscuti. Daca polinomul $C(q^{-1})$ este asimptotic stabil, relatia (4.7) este echivalenta cu:

$$\lim_{t\to\infty} e(t)=0$$

4.3.2 PRINCIPII DE PROIECTARE

Proiectarea sistemelor de conducere adaptiva, trebuie sa rezolve simultan doua probleme:

a) problema asigurarii stabilitatii sistemului in bucla inchisa. Pentru studiul stabilitatii sunt folosite rezultate ale teoriei Liapunov, ale teoriei hiperstabilitatii, sau, mai general, ale teoriei pasivitatii;

b) problema convergentei cu probabilitate 1 (sau aproape sigura) a algoritmilor, deoarece ne situam intr-un caz stohastic. Pentru analiza algoritmilor stohastici recursivi sunt utilizate metoda sistemului de ecuatii diferentiale ordinare (EDO) asociat, sau rezultate ale teoriei martingalelor. Aceste instrumente de analiza se constituie insa si in principii de proiectare a sistemelor adaptive pentru asigurarea obiectivelor urmarite.

In sinteza se porneste de la structura cunoscuta a modelului procesului si de la performantele impuse sistemului adaptiv si se obtine structura adecvata a regulatorului adaptiv si a sistemului ecuatiilor de eroare (fie de urmarire, fie de predictie, fie a estimatiilor parametrilor). In functie de starea acestui sistem se stabileste un candidat de functie Liapunov (eventual stohastica) si se cerceteaza conditiile in care aceasta devine functie Liapunov. Frecvent se utilizeaza o conditie de real (strict) pozitivitate.

Des utilizata este si teoriei hiperstabilitatii care ofera solutii de proiectare mai sistematice decât teoria Liapunov. Principalele proprietati ale sistemelor hiperstabile sunt:

- stabilitatea (asimptotica);
- intrarea marginita implica iesirea marginita;
- conexiunea in paralel sau cu reactie a doua sisteme hiperstabile constituie un sistem hiperstabil.

Ideea de baza a tehnicii de proiectare este obtinerea, plecând de la ecuatiile ce descriu sistemul adaptiv, a unei structuri echivalente cu reactie, in care blocul de pe calea directa este descris de o matrice de transfer strict pozitiva, iar blocul de pe reactie - continând elementele ajustabile, variante in timp, neliniaritatile si elementele de memorie - este hiperstabil. Aceasta asigura hiperstabilitatea ansamblului.

4.3.3 ANALIZA STABILITATII SI CONVERGENTEI STRUCTURILOR ADAPTIVE

Datorita neliniaritatii sistemelor in bucla inchisa obtinute in scopul asigurarii unei conduceri adaptive (prin diverse tehnici de proiectare), analiza acestora devine dificila in special, daca exista perturbatii aleatoare. Acesta este si motivul pentru care dezvoltarile teoretice au fost lente si migaloase. Teoria actuala ofera rezultate certe doar pentru unele probleme speciale. Mai sunt de studiat si rezolvat multe probleme inainte de a dispune de o teorie completa. Principalele probleme spre care proiectantul trebuie sa-si indrepte atentia sunt: analiza stabilitatii, convergentei si a performantelor. Deoarece estimarea parametrilor este o parte esentiala a sistemului, prezinta interes de asemenea cunoasterea comportarii in timp a blocului destinat estimarii parametrilor.

Analiza stabilitatii. Sunt utilizate extensiv teoremele de stabilitate ale lui Liapunov si Popov pentru comanda adaptiva. Principalele dezvoltari ale sistemelor adaptive cu model etalon au fost toate inspirate de dorinta de a construi mecanisme de ajustare care sa conduca la solutii stabile.

Pentru a asigura stabilitatea trebuie demonstrat ca vectorul ϕ este limitat. Acest lucru este simplu pentru sisteme care au doar un câstig (coeficient de amplificare) variabil. Componentele

vectorului ϕ sunt insa in general functie de intrarile si iesirile procesului. Deci nu este o problema prea simpla de a arata ca ϕ este limitat.

Analiza convergentei. Problemele esentiale ale analizei convergentei trebuiesc de asemenea investigate daca estimarile de parametri converg precum si in scopul de a determina viteza de convergenta. Pentru algoritmii expliciti, problema este echivalenta cu analiza convergentei estimarii recursive a parametrilor. Aceasta problema este tratata extensiv in teoria identificarii. Exista complicatii in cazul comenzii adaptive, deoarece intrarea procesului este generata prin reactie negativa.

Excitarea procesului depinde in cazul aplicatiilor practice de perturbatii. In dezvoltarile teoretice, de obicei se presupune ca sistemul este supus unor perturbatii aleatoare. Devine astfel posibila utilizarea teoriei ergodice si teoriei martingalelor. O demonstratie foarte generala a convergentei pentru algoritmul celor mai mici patrate a fost data de Sternby (1977) prin aplicarea unei teoreme martingale de convergenta.

4.4. UTILIZAREA ELEMENTELOR NEURONALE PENTRU IMPLEMENTAREA UNUI REGULATOR AUTOACORDABIL [Bha90] [Che95] [Cu94_r3] [Moh95]

4.4.1 SCHEMA DE CONDUCERE ADOPTATA

In vederea conducerii adaptive am implementat un regulator autoacordabil, estimarea parametrilor realizându-se prin actualizarea unui model de referinta dupa cum este prezentat in figura 4.4. Atât regulatorul adaptiv cât si blocul pentru estimarea parametrilor modelului procesului sunt realizate cu elemente neuronale EN.

Schema propusa realizeaza un RAA indirect deoarece parametrii regulatorului sunt actualizati indirect, in urma estimarii parametrilor modelului procesului si efectuarii calculelor de proiectare.

In continuare se vor prezenta detaliat blocurile ce apar in schema de principiu:

- *Blocul model estimat*

Modelul procesului este identificat prin utilizarea unei retele neuronale cu un strat activ cu un singur neuron pentru care ponderile sunt actualizate cu ajutorul algoritmului de proiectie pentru a minimiza eroarea dintre iesirea modelului $y_m=\hat{y}$ si iesirea procesului $y_p=y$.

Aceasta retea invata on-line dinamica procesului si este

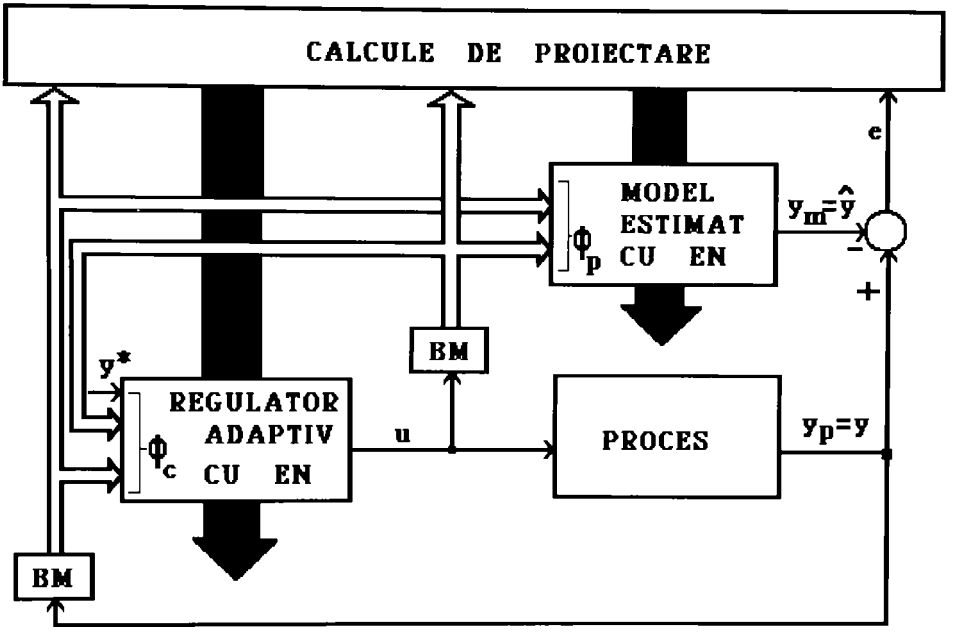


Figura 4.4 Schema bloc adoptata pentru conducerea adaptiva

folosita pentru ajustarea ponderilor celei de a doua retele neuronale (tot cu un strat activ) folosita ca regulator.

Blocul model estimat este prezentat in detaliu in figura 4.5.

La intrarea elementului neuronal este aplicat vectorul regresorilor (vectorul masuratorilor) ϕ_p:

$$\begin{aligned}\phi_p(k-1) &= [-y(k-1),\ -y(k-2),\ \ldots,\ -y(k-n),\\ &\qquad u(k-1),\ u(k-2),\ \ldots,\ u(k-m)]^T =\\ &= [\phi_p^1,\ \phi_p^2,\ \ldots, \phi_p^{n+m-1},\ \phi_p^{n+m}]^T \end{aligned} \qquad (4.9)$$

format din valori anterioare (memorate) ale comenzii u si ale iesirii procesului $y=y_p$. Ponderile corespunzatoare vectorului de intrare ϕ_p sunt ajustate on-line conform algoritmului de proiectie si au fost notate prin vectorul θ_p, iar functia de activare a fost aleasa sub forma $F(x)=C_1 tg(x)$, unde C_1 este o constanta data de nivelurile maxim si minim al iesirii procesului y.

◆ *Blocul regulator adaptiv*

Acest bloc este constituit dintr-o retea neuronala cu un neuron activ si este prezentat in detaliu in figura 4.6.

Comanda la momentul actual u se obtine prin prelucrarea

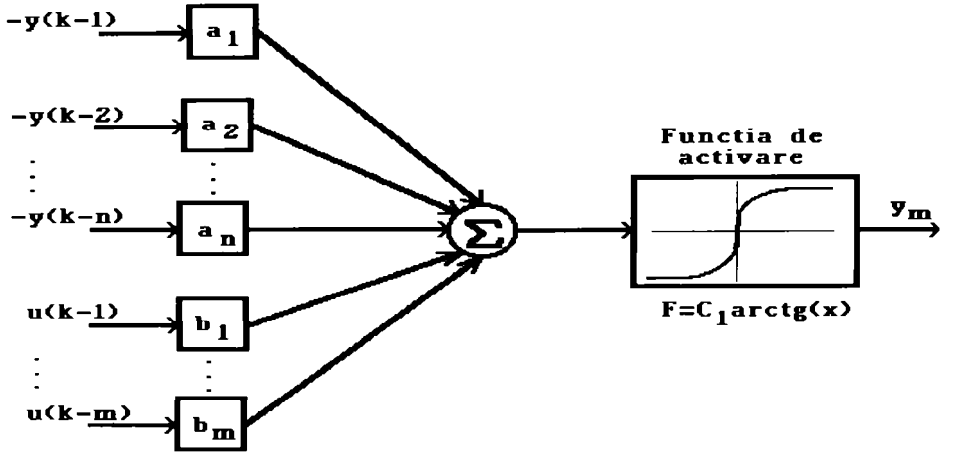


Figura 4.5 Schema blocului model estimat

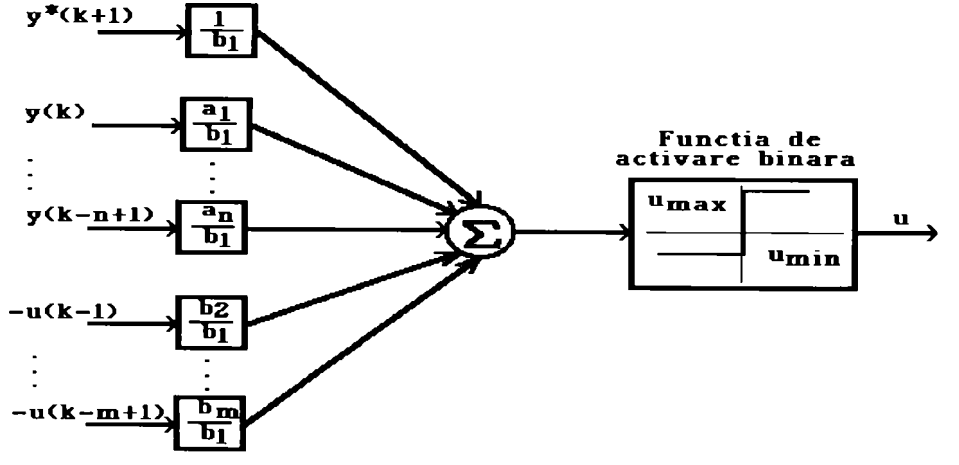


Figura 4.6 Schema blocului regulator adaptiv

vectorului de intrare ϕ_c compus din iesirea prescrisa y* si din valorile intârziate ale iesirii procesului $y=y_p$ si ale comenzii u.

$$\begin{aligned}\phi_c(k) &= [y^*(k+1),\ y^*(k),\ y(k-1),\ y(k-2),\ \ldots,\ y(k-n+1),\\ &\quad -u(k-1),\ -u(k-2),\ \ldots,\ -u(k-m+1)]^T =\\ &= [\phi_c^1,\ \phi_c^2,\ \ldots, \phi_c^{n+m-1},\ \phi_c^{n+m}]^T \end{aligned} \qquad (4.10)$$

Ponderile corespunzatoare acestui vector de intrare sunt reunite in vectorul θ_c care este ajustat printr-un artificiu de calcul pe baza ponderilor θ_p.

De aceasta data functia de activare aleasa a fost una binara.

◆ *Blocul de memorare a valorilor anterioare BM*

Cele doua blocuri prezente in schema de principiu sunt realizate sub forma:

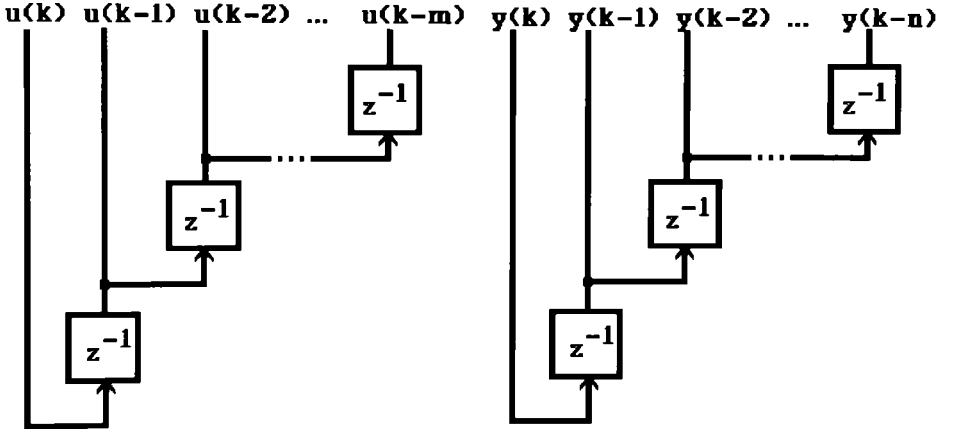


Figura 4.7 Schemele blocurilor BM

Ele furnizeaza marimile (regresorii) ce intra in componenta vectorilor ϕ_p si ϕ_c.

◆ *Blocul calcule de proiectare*

Acest bloc are rolul de a ajusta on-line ponderile celor doua elemente neuronale astfel incât iesirea procesului $y=y_p$ sa urmareasca semnalul de referinta y^*, indiferent de conditiile de functionare ale procesului.

4.4.2 ASPECTE DE IMPLEMENTARE

Se considera procesul ca fiind descris de ecuatia intrare-iesire discretizata:

$$A(q^{-1})y(t) = B(q^{-1})u(t) \quad (4.11)$$

unde $$A(q^{-1}) = 1+a_1q^{-1}+\ldots+a_nq^{-n} \quad (4.12)$$

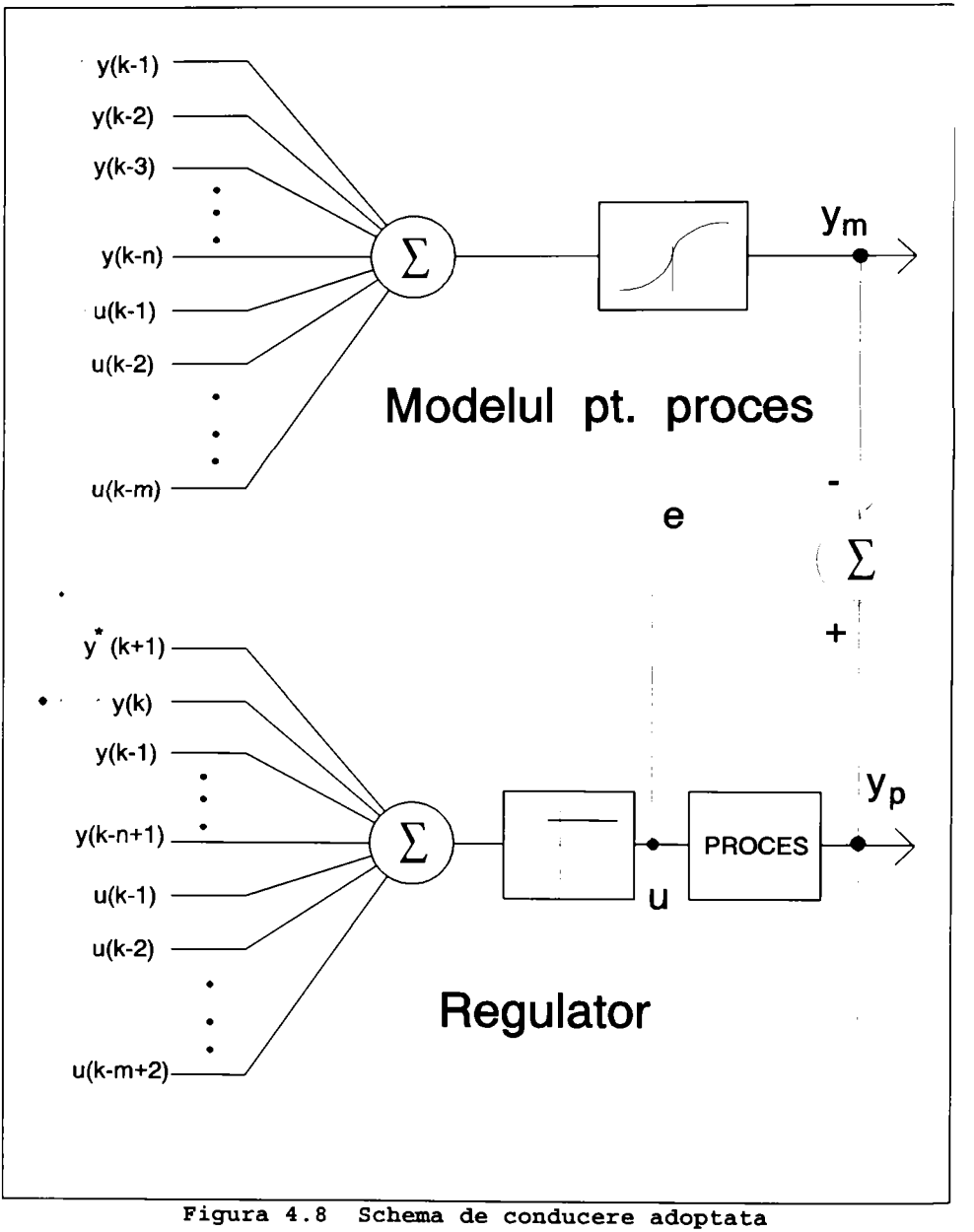


Figura 4.8 Schema de conducere adoptata

$$B(q^{-1}) = 1+b_1q^{-1}+\ldots+b_mq^{-m} \quad , \tag{4.13}$$

si care satisface urmatoarele cerinte:

1. Sunt cunoscute marginile superioare pentru gradele n si m;
2. Polinomul $B(q^{-1})$ este stabil polinomial;
3. Coeficientul $b_1 \neq 0$.

Pornind de la ecuatia procesului:

$$\begin{aligned} y(k)+a_1y(k-1)+a_2y(k-2)+\ldots+a_ny(k-n) &= \\ b_1u(k-1)+b_2u(k-2)+\ldots+b_mu(k-m) \quad , & \end{aligned} \tag{4.14}$$

separând in membrul stâng iesirea actuala, se obtine:

$$\begin{aligned} y(k) &= b_1u(k-1)+b_2u(k-2)+\ldots+b_mu(k-m) - \\ &\quad -a_1y(k-1)-a_2y(k-2)-\ldots-a_ny(k-n) \quad , \end{aligned} \tag{4.15}$$

Aceasta relatie a stat la baza utilizarii unei retele neuronale feedforward cu un singur strat activ, având ca intrare vectorul regresorilor:

$$\begin{aligned} \phi_p(k-1) &= [-y(k-1),\ -y(k-2),\ \ldots,\ -y(k-n), \\ &\qquad u(k-1),\ u(k-2),\ \ldots,\ u(k-m)]^T = \\ &= [\phi_p^{\,1},\ \phi_p^{\,2},\ \ldots, \phi_p^{\,n+m-1},\ \phi_p^{\,n+m}\ ]^T, \end{aligned} \tag{4.16}$$

ca ponderi vectorul estimat:

$$\theta_p(k) = [a_1\ a_2\ \ldots\ a_n\ b_1\ b_2\ \ldots\ b_m]^T, \tag{4.17}$$

iar ca iesire, valoarea estimata a iesirii procesului $y_m=\hat{y}$.

Aceasta retea neuronala invata dinamica procesului folosind pentru actualizarea (estimarea on-line) ponderilor algoritmul de proiectie:

$$\theta_p^i(k+1) = \theta_p^i(k) + \frac{\eta * \phi_p^i(k)}{\epsilon + \rho(k)} C(q^{-1})\, e(k) \tag{4.18}$$

$$\rho(k) = \lambda \rho(k-1) + \phi_p^T(k-1)\, \phi_p(k-1) \tag{4.19}$$

Algoritmii de proiectie sunt cei mai simpli algoritmi propusi pentru rezolvarea problemei de conducere adaptiva a sistemelor discrete deterministe. Acesti algoritmi asigura stabilitatea globala in conditii foarte largi. De asemenea, in implementarea pe calculator sunt necesare resurse de calcul (memorie, timp unitate centrala) foarte modeste. Utilizarea acestor algoritmi are insa

dezavantajul ca regimurile tranzitorii de adaptare pot fi semnificativ mai lungi decât cele corespunzatoare algoritmilor de tip CMMP.

In relatii apar urmatoarele marimi:

λ - este un factor de ponderare exponentiala a datelor in criteriu, a carui valoare redusa (de exemplu 0.98) la inceputul procedurii de autoacordare permite "uitarea" informatiilor de proasta calitate din faza initiala de operare. Factorul de "uitare" trebuie crescut imediat, dupa depasirea etapei initiale, de acordare bruta, dar mentinut la un nivel destul de scazut pentru a permite reacordarea regulatorului atunci când este necesar. Valorile tipice ale acestui parametru sunt:

$\lambda = 1$, memorie infinita;
$\lambda = 0.995$, memorie medie;
$\lambda = 0.98$, memorie foarte redusa.

Daca $\lambda(t) \equiv 1$ algoritmul pondereaza egal (in calcule) toate masuratorile din istoria trecuta a sistemului. Din aceasta cauza, in anumite situatii comportarea sistemului adaptiv poate fi nesatisfacatoare. De exemplu, daca dupa un interval lung de timp, in care a avut loc adaptarea, se produce o modificare semnificativa a parametrilor procesului (la schimbarea punctului de functionare), atunci procesul de adaptare va fi foarte lent, desi abaterile sunt importante. Acelasi lucru se poate intimpla, in conditii similare, la schimbarea semnificativa a marimii de referinta. O comportare nesatisfacatoare are loc si in cazul proceselor cu parametri lent variabili in timp. Pentru astfel de situatii se recomanda algoritmi cu factori de uitare $\lambda(t)<1$. Strategiile de selectie permitând $\lambda(t) \leq 1$ sunt mult mai flexibile si deci recomandabile.

η - constanta care poate fi privita atât din punctul de vedere al teoriei reglarii adaptive ca o constanta intre 0 si 1 cu influenta asupra regimului tranzitoriu cât si din punctul de vedere al teoriei retelelor neuronale ca si constanta de invatare intre 0 si 1 cu influenta asupra modului de invatare al retelei neuronale. O rata de invatare constanta si mare poate transforma algoritmul de invatare convergent intr-unul oscilant.

ϵ - este un scalar strict pozitiv.

$C(q^{-1})$ este un filtru de eroare care poate asigura variatii mai mici in amplitudinea marimilor de intrare si de iesire. $C(q^{-1})$ impune dinamica dorita a sistemului condus (in bucla inchisa). Filtrarea erorii diminueaza variatiile, dar determina cresterea duratei regimului tranzitoriu de adaptare.

$e(k) = y_p(k) - y_m(k)$, este eroarea la momentul k egala cu diferenta dintre iesirea procesului si iesirea prezisa de model.

Functia de activare folosita este arctangenta.

Modelul prezice iesirea procesului dupa formula:

$$y_m(k) = \arctan\left(\sum_{i=1}^{m+n} \phi_p^i(k-1)\,\theta_p^i(k-1)\right) \qquad \textbf{(4.20)}$$

Prin separarea in relatia (4.14) a valorii curente a comenzii u(k) se obtine:

$$u(k) = (1/b_1)\,[y(k+1)+a_1y(k)+a_2y(k-1)+\ldots+a_ny(k-n+1)- -b_2u(k-1)+b_3u(k-2)+\ldots+b_mu(k-m+1)]\ , \qquad (4.21)$$

In relatia (4.21) valoarea y(k+1) nefiind cunoscuta la momentul k se va inlocui cu valoarea ei prescrisa $y^*(k+1)$, rezultând:

$$u(k) = (1/b_1)\,[y^*(k+1)+a_1y(k)+a_2y(k-1)+\ldots+a_ny(k-n+1)- -b_2u(k-1)+b_3u(k-2)+\ldots+b_mu(k-m+1)]\ , \qquad (4.22)$$

Din ultima relatie apare ca naturala implementarea si a regulatorului adaptiv cu retea neuronala artificiala feedforward cu un singur strat având ca intrare vectorul regresorilor:

$$\phi_c(k) = [y^*(k+1),\ y(k),\ y(k-1),\ y(k-2),\ \ldots,\ y(k-n+1), -u(k-1),\ -u(k-2),\ \ldots,\ -u(k-m+1)]^T = = [\phi_c^1,\ \phi_c^2,\ \ldots, \phi_c^{n+m-1},\ \phi_c^{n+m}\ ]^T, \qquad (4.23)$$

ca ponderi, vectorul

$$\theta_c(k) = (1/b_1)\,[1\ a_1\ a_2\ \ldots\ a_n\ b_2\ \ldots\ b_m]^T, \qquad (4.24)$$

iar ca iesire, comanda la momentul actual u(k).

Din relatiile (4.17) si (4.24) se observa ca ponderile pentru regulator $\theta_c(k)$ se obtin din ponderile modelului $\theta_p(k)$ prin operatii simple de impartire.

Comanda se obtine la iesirea regulatorului dupa formula:

$$u(k) = \sum_{i=1}^{n+m} \phi_c^i(k)\,\theta_c^i(k) \qquad (4.25)$$

In urma celor prezentate rezulta ca modul de calcul al comenzii la pasul k, adica u(k) este:

1. se masoara iesirea procesului $y_p(k)$ si se determina urmatoarea valoare a iesirii prescrise $y^*(k+1)$;
2. cu ajutorul modelului neuronal al procesului se prezice iesirea procesului $y_m(k)$ folosind vechile ponderi $\theta_p(k-1)$ si se calculeaza eroarea $e(k)=y_p(k)-y_m(k)$;
3. folosind algoritmul de proiectie se calculeaza noile ponderi pentru model $\theta_p(k)$ si folosind relatia de transformare (4.24) se actualizeaza ponderile regulatorului $\theta_c(k)$;
4. se genereaza semnalul de comanda u(k) folosind regulatorul neuronal.

Demonstratia convergentei algoritmului destinat conducerii adaptive este prezentata in Anexa 4.2.

Regulatorul prezentat se autoadapteaza functie de eroarea de iesire a procesului. Cu cât aceasta este mai mare cu atât va fi mai

rapid procesul de adaptare. Pe durata regimului tranzitoriu de adaptare, legea de comanda neadaptata va produce comenzi necorespunzatoare si in consecinta, erori mari. Aceasta perioada de "acordare tranzitorie" este tipica tuturor regulatoarelor adaptive.

Alegerea ordinului sistemului si a timpului mort este de mare importanta. Regulatorul trebuie sa fie de ordin suficient de mare, pentru a putea faca fata dinamicii sistemului, fara ca acesta sa fie crescut in mod artificial. Regulatoarele având ordinul mare necesita mult timp pentru acordare, produc regimuri tranzitorii necorespunzatoare si sunt mai greu de acordat, in regim on-line, decât regulatoarele de ordin redus. Realizarea unei esantionari rapide a marimilor din proces conduce la regulatoare de ordin mare, in a caror functionare initiala apar comenzi violente pentru a face fata polilor rapizi si zgomotelor de masurare, inevitabile.

Limitele comenzii trebuie fixate astfel incât sa egaleze nivelele de saturatie ale elementelor de executie si sa nu necesite modificari on-line.

4.4.3 ALEGEREA UNOR PARAMETRI PENTRU CONDUCEREA ADAPTIVA

Denumirea de regulator autoacordabil poate conduce la concluzia falsa ca regulatoarele din aceasta clasa pot fi utilizate direct in conducerea unui proces, fara cunoasterea apriori a dinamicii acestuia, sau specificarea unor parametri de acordare. Regulatoarele autoacordabile utilizeaza legi de comanda destul de complexe. O acordare atenta a acestora implica specificarea de catre proiectant a unor parametri care apar in cadrul algoritmilor de estimare si proiectare. Alegerea corecta a acestora necesita cunoasterea dinamicii procesului si a principiilor de proiectare a regulatoarelor autoacordabile. De asemenea, se impune luarea unor masuri de siguranta pentru evitarea unor fenomene nedorite ce pot apare in timpul functionarii de durata a regulatoarelor adaptive.

Pentru etapa de estimare a parametrilor procesului sau ai regulatorului, utilizatorul poate decide asupra alegerii parametrilor expusi in continuare:

Ordinul sistemului

Ordinul sistemului este dat de ordinele autoregresiilor iesirii si intrarii. Stabilirea valorilor acestora se face in urma determinarii structurii modelului procesului folosind tehnici de identificare experimentala. Alegerea unor valori mai mici decât cele reale conduc la obtinerea unor regulatoare autoacordabile suboptimale, in timp ce alegerea unor valori mai mari poate mari inadmisibil durata procesului de estimare. In cazul in care nu apar restrictii de timp alegerea unor valori mai mari este indicata.

Timpul mort

Intregul k reprezentând numarul de perioade de esantionare cuprins in valoarea timpului mort al procesului, este specificat apriori. Acest parametru reprezinta constanta cea mai critica in cazul strategiilor de conducere adaptiva; o valoare a lui k mai mica decât valoarea reala poate conduce la instabilitate, in timp ce o valoare mai mare decât cea reala va conduce la o comportare suboptimala a sistemului. Alegerea pentru k a unor valori mai mari decât valoarea reala constituie o cale de obtinere a unor strategii de comanda suboptimale in cazul sistemelor de faza neminima.

Valorile initiale ale ponderilor $\theta(0)$

Se constata ca influenta lui $\theta(0)$ asupra statisticilor erorii in etapa finala este neesentiala. In etapa initiala insa, o valoare $\theta(0)$ necorespunzatoare poate provoca variatii mari in amplitudinea marimilor $\{u(t)\}$ si $\{y(t)\}$.

Factorul de uitare λ

Algoritmul de estimare poate utiliza un factor de ponderare exponentiala a datelor in criteriu având valori cuprinse intre 0.98 si 1. Factorul de uitare va avea in etapa initiala o valoare redusa (de exemplu 0.98) astfel incât regulatorul sa se poata adapta rapid la sistem. O valoare prea mica a acestui factor va conduce la un regim tranzitoriu de acordare de durata. Valoarea 0.98 este indicata a fi utilizata in perioada initiala de functionare, daca s-a utilizat un model numeric brut. O valoare initiala mai mare a factorului de uitare, de exemplu 0.99, indica certitudinea utilizatorului ca simularea anterioara a fost destul de precisa. Pe masura realizarii autoacordarii regulatorului, factorul de uitare poate fi crescut fie utilizând o procedura automata, fie prin interventia utilizatorului.

Perioada de esantionare T

Functie de valoarea timpului mort si de alegerea perioadei de esantionare rezulta un nou parametru, Δk - valoarea fractionara a timpului mort ($\Delta k<T$), care influenteaza comportarea sistemului intre perioadele de esantionare si care constituie o sursa de generare a unor modele discrete de faza neminima.

Alegerea perioadei de esantionare trebuie sa se bazeze pe dinamica principala a procesului condus, iar in absenta altor informatii perioada de esantionare se va alege egala cu valoarea partii reale a celui mai rapid pol important al sistemului. Evident, timpul de calcul reprezinta o limita superioara pentru alegerea vitezei de esantionare.

Tipul de regulator si detaliile legate de modul de utilizare a acestuia sunt determinate, in mare masura de procesul condus. Utilizatorul trebuie sa cunoasca bine procesul si obiectivele problemei de conducere in scopul selectarii parametrilor specifici etapei de conducere.

Limitele semnalului de comanda

Este esential ca limitele de amplitudine ce se impun prin software, semnalului de comanda, sa se situeze in interiorul limitelor de saturare ale sistemului. Aceste limite se determina experimental. Interfata hardware dintre calculator si instalatie trebuie sa asigure utilizarea unei zone cât mai largi de conversie a convertorului numeric analogic, astfel incât efectele de cuantizare sa nu domine semnalele de comanda.

Limitele vitezei de variatie a semnalului de referinta.

Semnalele de referinta fie ca sunt externe sau interne, generate de calculator sau nu, trebuie sa aiba o viteza de variatie sub viteza de raspuns reala sau estimata a sistemului condus, in scopul evitarii saturarii semnalelor de comanda.

4.4.4 STUDIU DE CAZ. REZULTATE. CONCLUZII

In acest paragraf se prezinta rezultatele unor experimente in simulare pentru conducerea adaptiva a unui sistem de ordinul trei ce isi modifica parametrii in functie de regimul de functionare.

Astfel s-a considerat ca in regimul de functionare A sistemul prezinta functia de transfer discreta:

$$H(q^{-1}) = \frac{0,0306q^{-1}+0,0712q^{-2}+0,0091q^{-3}}{1-1,67q^{-1}+0,871q^{-2}-0,0872q^{-3}} \qquad \textbf{(4.26)}$$

iar pentru regimul de functionare B, aceasta devine:

$$H(q^{-1}) = \frac{0,0431q^{-1}+0,139q^{-2}+0,0283q^{-3}}{1-2,184q^{-1}+1,662q^{-2}-0,0872q^{-3}} \qquad \textbf{(4.27)}$$

Trecerea de la un regim de functionare la altul se considera a se realiza brusc, astfel ca parametii vor fi variati conform unui grafic dreptunghiular cu factorul de umplere 0.5.

Rezultatele au fost obtinute in urma scrierii algoritmului de conducere in limbajul C al firmei Borland, program prezentat in Anexa 4.1 (adaptiv.c) si operational pe calculatoare IBM-PC compatibile.

Regulatorul autoacordabil implementat pentru conducerea aceastui proces realizeaza o buna urmarire a marimii de referinta de catre iesirea procesului. Pentru o functionare optima este necesara specificarea parametrilor de acordare precizati in paragraful 4.4.3. La concluziile asupra influentei acestora asupra procesului de reglare s-a ajuns prin studierea nu numai a acestui exemplu ci si a altor procese, insa toate graficele au fost trasate pentru acest caz particular.

Daca nu exista disponibil un model al procesului sau atunci când exista el reprezinta o aproximare grosiera a comportarii

procesului rezulta dificultati in precizarea ordinului sistemului. Am constatat ca alegerea pentru sistem a unui ordin mai mare decât cel real nu influenteaza performantele algoritmului de conducere. Prin urmare ordinul sistemului nu este o constanta critica si se poate alege mai mare decât ordinul real al sistemului. Totusi acesta nu trebuie ales prea mare deoarece timpul necesar calculului comenzii poate creste inadmisibil de mult.

Deasemenea influenta valorilor initiale ale ponderilor $\theta(0)$ asupra statisticilor erorii in etapa finala este neesentiala. Initializarea cu 0 a valorilor parametrilor modelului procesului si regulatorului este comoda si ofera rezultate satisfacatoare. O alta modalitate de initializare ar putea fi realizata cu valori aleatoare cuprinse intre -0.1 si 0.1.

Coeficientul ϵ ce apare la numitorul relatiei de ajustare a ponderilor se poate alege fara probleme 0.01 sau 1.

Strategiile de comanda permitând un factor de uitare subunitar $\lambda<1$ si variabil sunt mai flexibile si permit o mai buna adaptare. O astfel de strategie se poate implementa alegând un factor de uitare redus la inceputul perioadei de acordare, iar dupa depasirea fazei initiale de acordare bruta este crescut putând fi scazut ori de câte ori apar variatii bruste in marimea de referinta sau se modifica punctul de functionare.

Filtrul de eroare $C(q^{-1})$ are o mare influenta asupra convergentei procesului de invatare, o alegere necorespunzatoare putând duce la algoritmi divergenti.

Valori mici pentru constanta de invatare η duc la algoritmi de invatare oscilanti sau provoaca salturi in perioada initiala. Spre deosebire de cazul retelelor neuronale artificiale care ajusteaza ponderile prin metoda gradientului fiind posibil cazul in care nu se ajunge la nici o solutie, pentru neuronii propusi este demonstrat ca sirul erorilor in estimarea ponderilor este convergent (Anexa 4.2).

Datorita functiei de activare pe iesirea neuronului (care restrânge domeniul de iesire) comanda este amplificata cu un factor dependent de procesul reglat C_1.

Viteza de convergenta este de aproximativ 7 perioade. In continuare se prezinta sub forma tabelara timpul de calcul pentru comanda pe un pas in functie de marimea ordinului sistemului:

n=3	n=5	n=9
16.5 ms	25 ms	38.5 ms

Rezultatele au fost obtinute pe un PC - 286 la 16 MHz.

In concluzie RAA propus este stabil, nu necesita interventia operatorului decât in faza initiala pentru initializarea unui numar

redus de parametrii. Algoritmul utilizat, prin relatiile de ajustare folosite, asigura convergenta estimatiilor si marginirea comenzii si a iesirii. Acest algoritm de invatare convergent poate fi utilizat si in rezolvarea unor alte tipuri de probleme (de exemplu modelarea on-line a seriilor de timp).

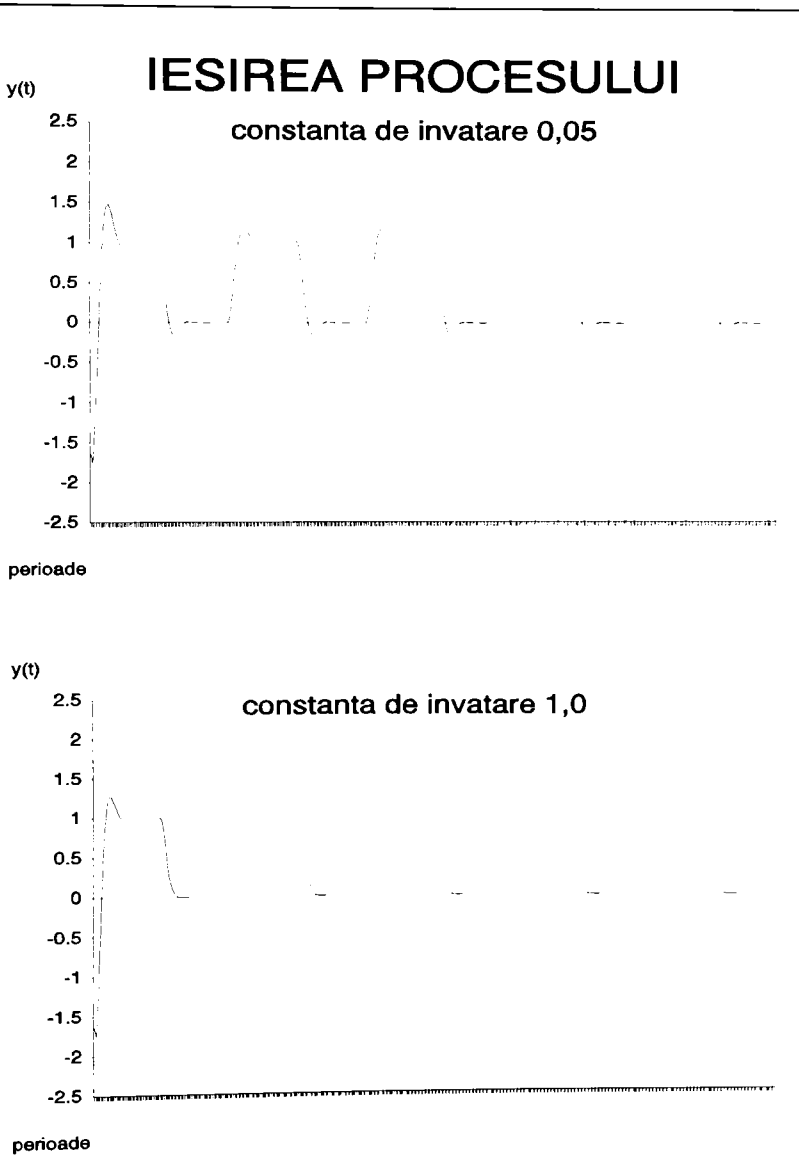


Figura 4.9

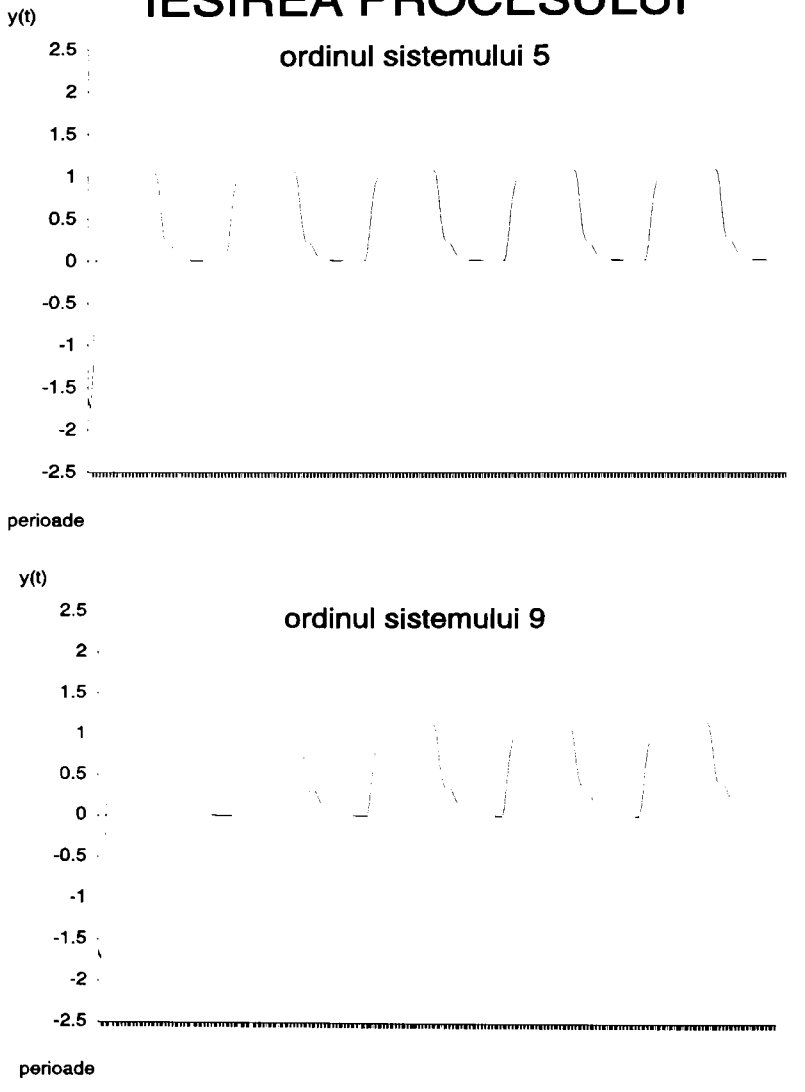


Figura 4.10

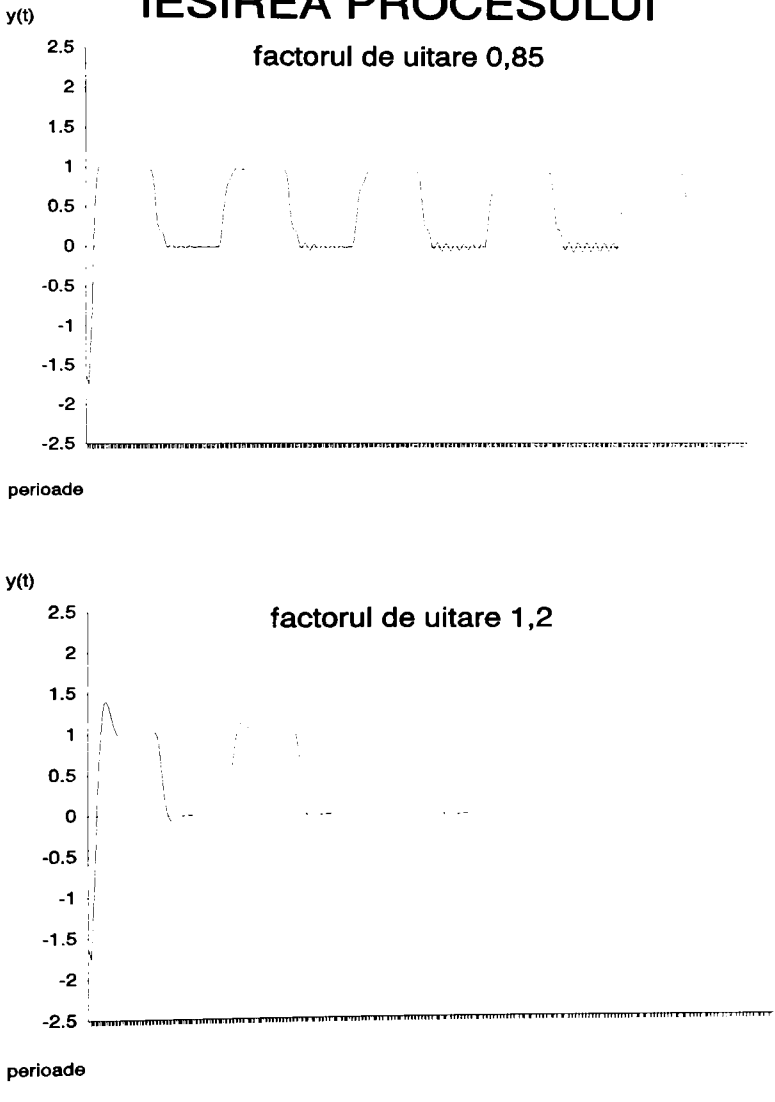


Figura 4.11

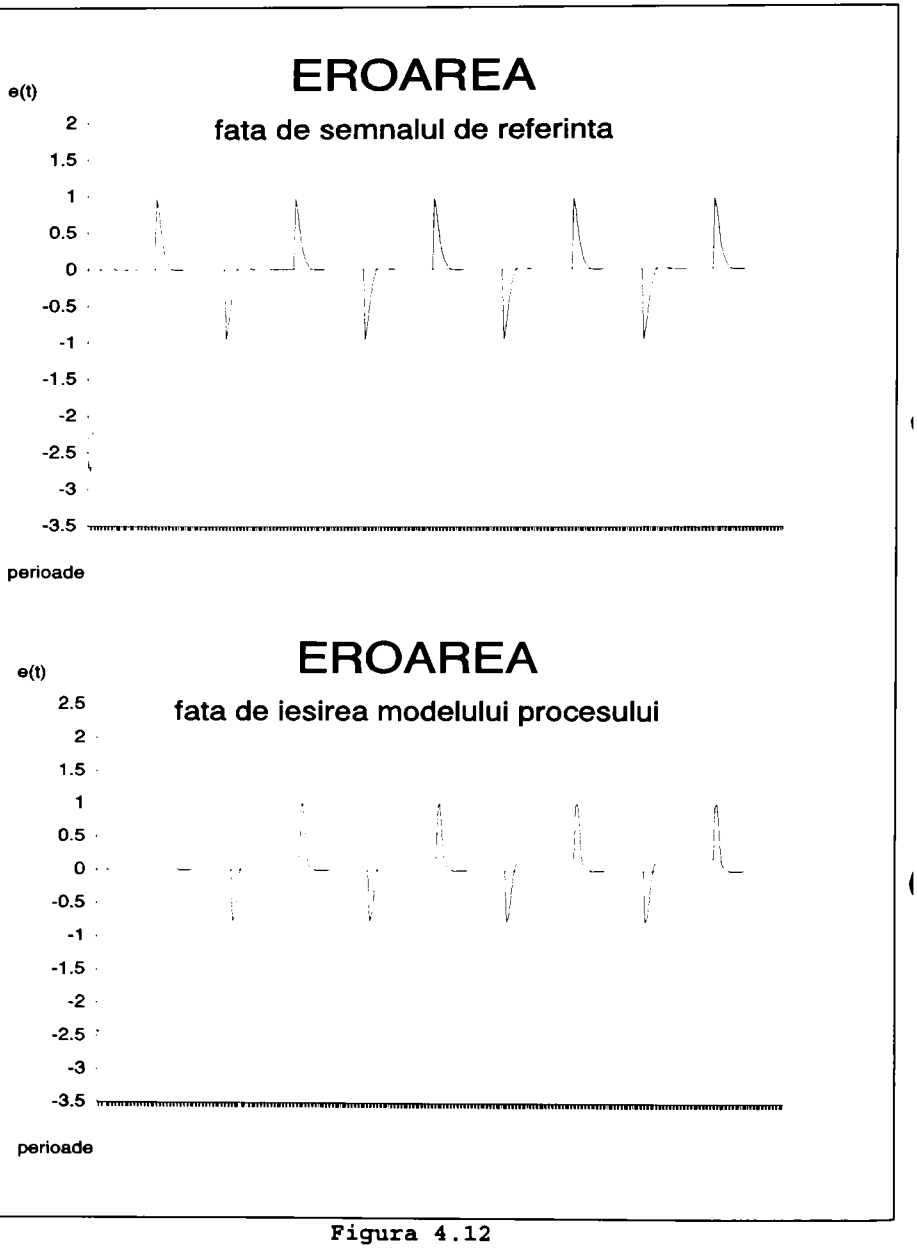


Figura 4.12

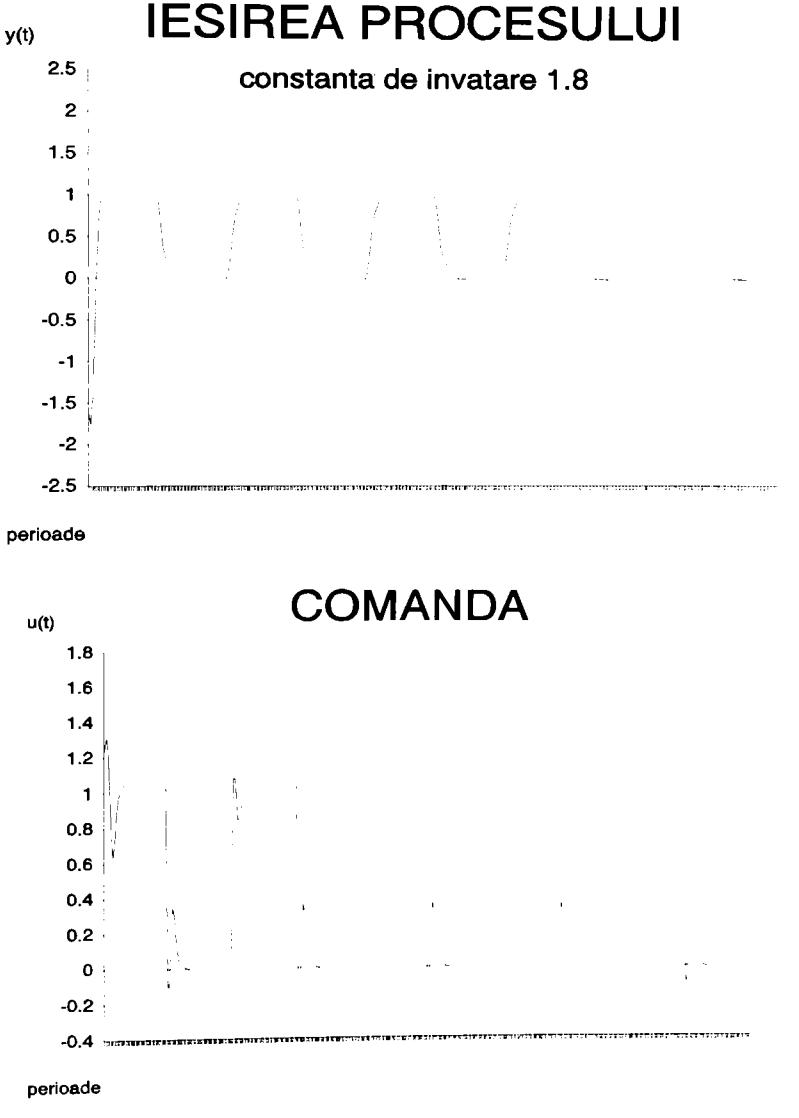


Figura 4.13

ANEXA 4.1 PROGRAMUL SURSA IN LIMBAJUL C CE IMPLEMENTEAZA ALGORITMUL PROPUS

```
//************************************************************
// PROGRAM PT. CONDUCEREA ON-LINE A UNUI PROCES FOLOSIND UN
// SISTEM ADAPTIV REALIZAT CU RETELE NEURONALE ADAPTIV.C
//************************************************************
// Copyright  April, 1995  Curiac, V 2.0
//************************************************************

#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <graphics.h>
#include <conio.h>
#include <stdarg.h>

#define N 5
#define M 5
#define NM N+M
#define UMAX 10
#define UMIN -10

double y_prescris(int k)
{
  return ((k/40)%2) ? 0.0 : 1.0;
}

int gprintf( int *xloc,int *yloc,char *fmt,...)
                /*tiparire in mod grafic*/
     {
     va_list argptr;
     char str[140];
     int cnt;
     va_start(argptr,fmt);
     cnt=vsprintf(str,fmt,argptr);
     outtextxy( *xloc, *yloc, str);
     *yloc+=textheight("H")+2;
     va_end(argptr);
     return( cnt );
     }


void main(void)
{
 register int i;
 int k,nr_epoci;
```

```
int m,n;
int x,y;
int x1,y11,y12,y21,y22;
double c0,c1,c2;
double er,err,err1,emp,yp,yob,eroarea,e,x2,comanda;
double thetap[NM],fip[NM],fi[NM],thetamodel[NM];
double thetac[NM],fic[NM];
double alfa,epsilon,lambda;
FILE *fisier;
double com;
/* request auto detection */
int gdriver = DETECT, gmode, errorcode;
int xmax,ymax;

randomize();
for(i=0;i<NM;i++)
  {thetap[i]=0.0;
  fip[i]=0;
  fi[i]=0;
  thetamodel[i]=0;
  thetac[i]=0;
  fic[i]=0;}

//initializari
n=3;  m=3;  nr_epoci=600;

thetamodel[0]=-1.67;    // a1
thetamodel[1]=0.871;    // a2
thetamodel[2]=-0.0872;
thetamodel[3]=0.0306;
thetamodel[4]=0.0712;
thetamodel[5]=0.0091;


alfa = 1.0;      // constanta de invatare
epsilon = 0.01; //ct. numitorului rel. de ajustare ponderi
thetap[n] = 1.0;
fip[n] = 1.0;
fip[0] = 1.0;
fic[0]=y_prescris(2);
c0=1.0; c1=-1.0; c2=0.0;  // filtrul de eroare
lambda = 0.99;            // factorul de uitare
yob=0.0; com=1.335;
err=0.0; x2=0.0; err1=0.0; emp=0.0;

x1=10;
y11=200;
y12=200;

/* initialize graphics mode */
initgraph(&gdriver, &gmode, "c:\\borland.c\\bgi");
```

```
/* read result of initialization */
errorcode = graphresult();

if (errorcode != grOk)  /* an error occurred */
{
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Press any key to halt:");
    getch();
    exit(1);             /* return with error code */
}
xmax=getmaxx();
ymax=getmaxy();
setfillstyle(SOLID_FILL,LIGHTGREEN);
bar(0,0,xmax,ymax);
setcolor(BLACK);
rectangle(1,1,xmax-1,ymax-1);
rectangle(3,3,xmax-3,ymax-3);
line(5,200,xmax-5,200);
line(10,50,10,350);
fisier=fopen("adaptiv.txt","w");
fprintf(fisier,"Valorile initiale ale starii modelului sunt:\n");
for(i=0;i<m+n;i++)
  fprintf(fisier,"%f \t",thetap[i]);
fprintf(fisier,"\n");
for(k=1;k<nr_epoci;k++)      // bucla de reglare
 {
 if(k==200){
     thetamodel[0]=-2.184; // a1
     thetamodel[1]=1.662;  // a2
     thetamodel[2]=-0.432;
     thetamodel[3]=0.043;
     thetamodel[4]=0.139;
     thetamodel[5]=0.0283;
     com=0.275;
     x2=0;
     for(i=0;i<NM;i++)
       {thetap[i]=0.0;
       thetac[i]=0;
       thetap[n] = 1.0;}
     }
 if(k%40<15)
     lambda=0.99;
 else
     lambda=0.995;

 yp=0;
 for(i=0;i<m+n;i++)
     yp+=thetap[i]*fip[i];//iesirea prezisa folosind vechile
                          // ponderi
 yp=0.1*atan(yp);
 yob=0;
```

```
for(i=0;i<m+n;i++)
yob+=thetamodel[i]*fip[i];  //valoarea iesirii procesului
   //in cazul real se obtine de la iesirea procesului
yob+=(random(1000)-500)/1000;

x2=9+k;
y21=floor(200-100*y_prescris(k));
setcolor(WHITE);
line(x1,y11,x2,y21);
y22=floor(200-100*yob);
setcolor(RED);
line(x1,y12,x2,y22);
x1=x2;
y11=y21;
y12=y22;
eroarea=c0*(yob-yp)+c1*err+c2*err1;
err1=err;
err=yob-yp;
er=yob-10*yp;
emp+=er*er;
for(i=0;i<m+n;i++)
    x2=lambda*x2+fip[i]*fip[i];
for(i=0;i<m+n;i++)  //se calculeaza noile ponderi pt.
                    // modelul procesului
    thetap[i]+=(alfa*eroarea*fip[i])/(epsilon+x2);
thetac[0]=1/thetap[n]; //calcul ponderile pt. controler
for(i=0;i<n;i++)
    thetac[i+1]=thetap[i]/thetap[n];
for(i=n+1;i<m+n;i++)
    thetac[i]=thetap[i]/thetap[n];
fic[0]=y_prescris(k+1);
for(i=1;i<n+1;i++)
    fic[i]=-fip[i-1];
for(i=n+1;i<m+n;i++)
    fic[i]=-fip[i];
comanda=0.0;  //se calculeaza comanda
for(i=0;i<m+n;i++)
    {comanda+=thetac[i]*fic[i];fi[i]=fip[i];}
comanda=com*atan(comanda);
if(comanda>UMAX) comanda=UMAX;
if(comanda<UMIN) comanda=UMIN;
fip[0]=-yob;         //se fac trecerile la un nou pas
for(i=1;i<n;i++)
    fip[i]=fi[i-1];
fip[n]=comanda;
for(i=n+1;i<m+n;i++)
    fip[i]=fi[i-1];
e=yob-y_prescris(k);
fprintf(fisier,"%lf ",e);
fprintf(fisier,"Epoca= %d \t eroarea= %f \t iesirea=%f
                  \n",k,err,yob);
```

```
  fprintf(fisier,"Modelul procesului este : \n");
  for(i=0;i<m+n;i++)
     fprintf(fisier,"%f \t",thetap[i]);
  fprintf(fisier,"\n");
  fprintf(fisier,"Vectorul corespunzator modelului : \n");
  for(i=0;i<m+n;i++)
     fprintf(fisier,"%f \t",fip[i]);
  fprintf(fisier,"\n");
  fprintf(fisier,"Vectorul corespunzator controlerului : \n");
  for(i=0;i<m+n;i++)
     fprintf(fisier,"%f \t",fic[i]);
  fprintf(fisier,"\n \n");
 } //for nr_epoci
 fclose(fisier);
 /* clean up */
 emp=emp/nr_epoci;
 x=100; y=350;
 gprintf(&x,&y,"%f",emp);
 getch();
 closegraph();
} //main
```

ANEXA 4.2: DEMONSTRAREA CONVERGENTEI ALGORITMULUI DESTINAT CONDUCERII ADAPTIVE

Se considera structura simplificata a sistemului de conducere adaptiva dezvoltat pe parcursul acestui capitol:

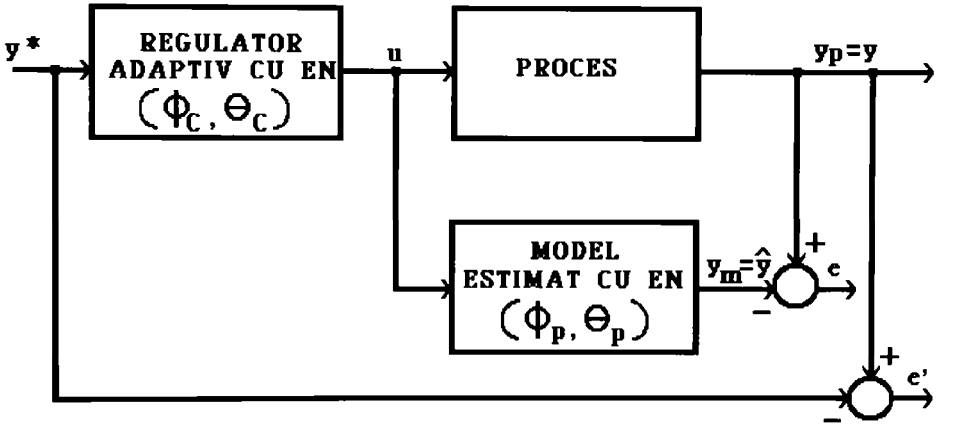


Figura 4.14 Schema bloc a sistemului adaptiv

PROBLEMA 4.1: Sa se demonstreze egalitatea celor doua marimi de eroare precizate in figura, adica e si e'.

Demonstratie:

Iesirea reala a procesului poate fi exprimata pe baza modelului sau matematic intrare-iesire discret sub forma:

$$y(k)=y_p(k)=-a_{01}y(k-1)-\ldots-a_{0n}y(k-n)+b_{01}u(k-1)+\ldots+ +b_{0m}u(k-m)=\theta_{0p}(k-1)\phi_p(k-1)^T=\theta_{0p}\phi_p(k-1)^T \quad (4.28)$$

unde:

$\theta_{0p}=[a_{01}\ \ldots\ a_{0n}\ \ b_{01}\ \ldots\ b_{0m}]^T$ - vectorul parametrilor reali ai procesului;

$\phi_p(k-1)=[-y(k-1)\ \ldots\ -y(k-n)\ \ u(k-1)\ \ldots\ u(k-m)]^T$ - vectorul masuratorilor.

Iesirea prezisa a procesului poate fi exprimata pe baza modelului matematic intrare-iesire discret corespunzator blocului

de estimare model sub forma:

$$\hat{y}(k)=y_m(k)=-a_1y(k-1)-\ldots-a_ny(k-n)+b_1u(k-1)+\ldots+b_mu(k-m)=\\ =\theta_p(k-1)\,\phi_p(k-1)^T \qquad (4.29)$$

unde:

$\theta_p(k-1)=[a_1\ \ldots\ a_n\ \ b_1\ \ldots\ b_m]^T$ - vectorul parametrilor estimati ai procesului.

Pentru exprimarea iesirii prescrise y*(k) se recurge la un artificiu de calcul: se exprima u(k-1) din relatia (4.29):

$$u(k-1)=\frac{1}{b_1}[\hat{y}(k)+a_1y(k-1)+\ldots+a_ny(k-n)-b_2u(k-2)-\ldots-b_mu(k-m)]$$

se inlocuieste valoarea prezisa $\hat{y}(k)$, care nu este disponibila la momentul (k-1) cu valoarea iesirii prescrise la momentul k, y*(k) si se separa in membrul stâng aceasta valoare, rezultând:

$$y^*(k)=-a_1y(k-1)-\ldots-a_ny(k-n)+b_1u(k-1)+\ldots+b_mu(k-m)=\\ =\theta_p(k-1)\,\phi_p(k-1)^T \qquad (4.30)$$

In continuare se va explicita eroarea e'(k), pe baza relatiilor (4.28) si (4.30):

$$e'(k)=y(k)-y^*(k)=\theta_{0p}\phi_p(k-1)^T-\theta_p(k-1)\,\phi_p(k-1)^T=(\theta_{0p}-\theta_p(k-1))\,\phi_p(k-1)^T=\\ =-\tilde{\theta}_p(k-1)\,\phi_p(k-1)^T \qquad (4.31)$$

si eroarea e(k) pe baza relatiilor (4.28) si (4.29):

$$e(k)=y(k)-\hat{y}(k)=\theta_{0p}\phi_p(k-1)^T-\theta_p(k-1)\,\phi_p(k-1)^T=\\ =(\theta_{0p}-\theta_p(k-1))\,\phi_p(k-1)^T=-\tilde{\theta}_p(k-1)\,\phi_p(k-1)^T \qquad (4.32)$$

Observând egalitatea expresiilor situate in membrul drept ale relatiilor (4.31) si (4.32) se poate conchide ca:

$$e'(k)=e(k) \qquad (4.33)$$

PROBLEMA 4.2: Sa se demonstreze convergenta algoritmului destinat conducerii adaptive, adica sa se arate ca:

$$\lim_{k\to\infty} e(k)=0$$

Demonstratie:

Se considera algoritmul de proiectie sub forma urmatorului sistem de doua ecuatii recursive:

$$\theta_p(k)=\theta_p(k-1)+\frac{\eta*\phi_p(k-1)}{\epsilon+\rho(k-1)}e(k) \qquad (4.34)$$

$$\rho(k-1)=\lambda\rho(k-2)+\phi_p^T(k-1)\phi_p(k-1) \qquad (4.35)$$

Fiecarui membru al relatiei (4.34) i se scade vectorul parametrilor reali ai procesului, adica θ_{0p}:

$$\theta_p(k)-\theta_{0p}=\theta_p(k-1)-\theta_{0p}+\frac{\eta*\phi_p(k-1)}{\epsilon+\rho(k-1)}e(k) \qquad (4.36)$$

si se noteaza erorile in estimarea parametrilor la momentul k si k-1 utilizând simbolul tilda:

$$\theta_p(k)-\theta_{0p}=\tilde{\theta}_p(k) \qquad (4.37)$$

$$\theta_p(k-1)-\theta_{0p}=\tilde{\theta}_p(k-1) \qquad (4.38)$$

Astfel relatia (4.36) devine:

$$\tilde{\theta}_p(k)=\tilde{\theta}_p(k-1)+\frac{\eta*\phi_p(k-1)}{\epsilon+\rho(k-1)}e(k) \qquad (4.39)$$

Pentru a continua demonstratia trebuie facut apel la urmatoarea lema de calcul vectorial, a carei demonstratie fiind imediata nu se va detalia:

Lema: Fie A, B si C trei vectori. Daca A=B+C atunci:

$$A^TA=B^TB+C^TC+2B^TC \tag{4.40}$$

Facând apel la aceasta lema, relatia (4.39) devine:

$$\tilde{\theta}_p(k)^T\tilde{\theta}_p(k)=\tilde{\theta}_p(k-1)^T\tilde{\theta}_p(k-1)+\left[\frac{\eta\phi_p(k-1)\,e(k)}{\epsilon+\rho(k-1)}\right]^T\left[\frac{\eta\phi_p(k-1)\,e(k)}{\epsilon+\rho(k-1)}\right]+ +2\frac{\eta\tilde{\theta}_p(k-1)^T\phi_p(k-1)\,e(k)}{\epsilon+\rho(k-1)} \tag{4.41}$$

Pentru simplificarea scrierii se fac urmatoarele notatii:

$$\tilde{\theta}_p(k)^T\tilde{\theta}_p(k)=\|\tilde{\theta}_p(k)\|^2 \tag{4.42}$$

$$\tilde{\theta}_p(k-1)^T\tilde{\theta}_p(k-1)=\|\tilde{\theta}_p(k-1)\|^2 \tag{4.43}$$

si se rescrie relatia (4.41) sub forma:

$$\|\tilde{\theta}_p(k)\|^2-\|\tilde{\theta}_p(k-1)\|^2=\left[\frac{\eta\phi_p(k-1)\,e(k)}{\epsilon+\rho(k-1)}\right]^T\left[\frac{\eta\phi_p(k-1)\,e(k)}{\epsilon+\rho(k-1)}\right]+ +2\frac{\eta\tilde{\theta}_p(k-1)^T\phi_p(k-1)\,e(k)}{\epsilon+\rho(k-1)} \tag{4.44}$$

In continuare ne vom opri asupra membrului drept al relatiei (4.44), care va fi notat cu E. Deoarece marimea e(k) este scalara, deci este egala cu transpusa ei, si tinând cont de relatia (4.31) rezulta pentru E expresia:

$$E=\eta\left[\frac{\eta\phi_p(k-1)^T\phi_p(k-1)}{\epsilon+\rho(k-1)}-2\right]\frac{e(k)^2}{\epsilon+\rho(k-1)} \tag{4.45}$$

Se va demonstra ca E<0 daca η apartine intervalului (0,1).

Pe baza relatiei (4.35), tinând cont ca valoarea initiala a lui ρ se alege pozitiva, adica $\rho(0)\geq 0$, rezulta ca $\rho(k-1)\geq 0$ (se obtine din valoarea sa anterioara prin adunarea unei cantitati

pozitive). Totodata ϵ este un scalar strict pozitiv. Pe baza acestor consideratii se poate demonstra ca:

$$\frac{e(k)^2}{\epsilon+\rho(k-1)} \geq 0 \qquad \textbf{(4.46)}$$

Daca η apartine intervalului (0,1), pe baza relatiei (4.35) se poate demonstra simplu ca:

$$\frac{\eta \phi_p(k-1)^T \phi_p(k-1)}{\epsilon+\rho(k-1)} < 1 \qquad \textbf{(4.47)}$$

deci valoarea parantezei din relatia (4.45) este sub -1, si in consecinta E<0.

Pe baza relatiei (4.44) se poate conchide ca:

$$\|\tilde{\theta}_p(k)\|^2 < \|\tilde{\theta}_p(k-1)\|^2 \qquad \textbf{(4.48)}$$

Observând relatia (4.42) rezulta ca:

$$\|\tilde{\theta}_p(k)\|^2 \geq 0 \qquad \textbf{(4.49)}$$

$\|\tilde{\theta}_p(k)\|^2$ este un sir monoton descrescator (4.48) si marginit inferior (4.49), deci este convergent.

S-a obtinut astfel un rezultat important: eroarea globala in calculul ponderilor descreste la fiecare pas, nemaiaparând dificultatile discutate in cazul retelelor neuronale cu mai multe straturi active.

In continuare, pe baza acelorasi considerente asupra expresiei E (in special (4.47)) se poate conchide ca:

$$\|\tilde{\theta}_p(k)\|^2 - \|\tilde{\theta}_p(k-1)\|^2 < -\frac{e(k)^2}{\epsilon+\rho(k-1)} \qquad \textbf{(4.50)}$$

sau:

$$\|\tilde{\theta}_p(k-1)\|^2 - \|\tilde{\theta}_p(k)\|^2 > \frac{e(k)^2}{\epsilon+\rho(k-1)} \qquad \textbf{(4.51)}$$

Prin trecere la limita in relatia (4.51) tinând cont de

relatiile (4.46) si (4.48) se obtine:

$$\lim_{k\to\infty}\frac{e(k)^2}{\epsilon+\rho(k-1)}=0 \qquad \textbf{(4.52)}$$

si, deci:

$$\lim_{k\to\infty} e(k)=0 \qquad \textbf{(4.53)}$$

Cum e'(k)=e(k) rezulta:

$$\lim_{k\to\infty} e'(k)=0 \qquad \textbf{(4.54)}$$

demonstrând convergenta algoritmului propus.

CAPITOLUL 5: CONCLUZII

Lucrarea de fata se incadreaza in domeniul dezvoltarii unor tehnici de conducere adaptive si optimale a agregatelor aeroelectrice. Sunt prezentate solutii originale pentru predictia seriilor de timp corespunzatoare vitezei vintului utilizind teoria sistemelor haotice si a retelele neuronale recurente, precum si o solutie de conducere adaptiva a agregatelor aeroelectrice bazata pe utilizarea elementelor neuronale.

Modelarea vitezei vintului se constituie intr-o etapa absolut necesara implementarii unor tehnici de conducere optimale a agregatelor aeroelectrice. Astfel, au fost implementate doua tehnici de predictie a vitezei vintului:

a) - una on-line destinata predictiei pe termen scurt (zecimi de secunda) utilizind rezultate ale teoriei sistemelor haotice;

b) - una off-line destinata predictiei pe termen lung (ore, zile, etc.) utilizind retele neuronale recurente.

In conducerea agregatelor aeroelectrice apar probleme deosebite, date fiind caracteristicile procesului, care fac in multe cazuri imposibila utilizarea teoriei conducerii sistemelor liniare cu parametri constanti: parametri necunoscuti ai modelului agregatului si vintului; valoarea apreciabila a timpului mort, adesea variabila; comportarea puternic neliniara a procesului; caracteristici de transfer care variaza in timp, din cauza modificarilor parametrilor rotorului in functie de viteza vintului; propagarea unor perturbatii cunoscute (vibratii) de-a lungul agregatului. Din acest motiv se impune dezvoltarea unor tehnici de conducere adaptive cu o functionare sigura. Solutia pe care o propune prezenta lucrare se refera la utilizarea a doua retele

neuronale cu un singur strat activ, una destinata estimarii on-line a parametrilor agregatului si una avind rolul de regulator adaptiv.

Rezultatele investigatiilor autorului au fost la nivelul asteptarilor, si au confirmat buna orientare a directiei de aprofundare impusa de conducatorul stiintific. Contributiile autorului se rezuma la urmatoarele:

1. Prezentarea sintetica a notiunii de serie de timp precum si a modelelor utilizate in practica modelarii acestora.
2. Analiza critica a metodologiei Box-Jenkins destinata modelarii si predictiei pe termen scurt a seriilor de timp. Evidentierea dezavantajelor utilizarii semnalului tip zgomot alb in cadrul abordarii Box-Jenkins.
3. Abordarea sintetica a problematicii sistemelor dinamice haotice privita prin prisma necesitatii generarii unor semnale de test haotice atit in sistemele continuale cit si in sistemele discrete.
4. Solutie de predictie on-line a vitezei vintului bazata pe utilizarea in etapa de identificare a unui semnal haotic discret. Aceasta solutie elimina dezavantajele utilizarii semnalului tip zgomot alb din cadrul abordarii Box-Jenkins.
5. Elaborarea unui pachet de programe (SERTIM 1.0) destinat simularii fazei de estimare multivariabila on-line a parametrilor seriilor de timp. Acest mediu de simulare are la baza solutia enuntata la punctul anterior.
6. Prezentarea sintetica a retelelor neuronale total conectate, a retelelor neuronale recurente si a algoritmului backpropagation de antrenare al acestora.
7. Evidentierea principalelor dezavantaje ale utilizarii metodei gradientului in antrenarea retelelor neuronale total conectate.
8. Metoda originala de antrenare a retelelor neuronale total conectate, metoda ce inlatura necesitatea interventiei operatorului uman in etapa de antrenament.
9. Elaborarea unui program de antrenare a retelelor neuronale (antrenam.exe), avind la baza metoda de antrenare propusa.
10. Solutie de modelare si predictie pe termen lung (ore, zile, etc.) a vitezei vintului utilizind retele neuronale recurente.
10. Studiu asupra alegerii erorii maxime admisibile in procesul de antrenare a retelelor neuronale recurente.
11. Verificarea prin simulare (sunt prezentate patru studii de caz) a metodologiei de predictie a vitezei vintului utilizind retelele neuronale recurente.
12. Abordarea sintetica a principalelor tehnici de conducere adaptiva si evidentierea avantajelor utilizarii acestora in raport cu tehnicile de conducere considerate clasice.
13. Solutie de implementare a unui regulator adaptiv utilizind retele neuronale cu un singur nivel activ. Prezentarea structurii regulatorului adaptiv, precum si a strategiei de modificare a parametrilor acestuia.
14. Studiu privind alegerea unor parametri pentru conducerea adaptiva.
15. Demonstrarea convergentei algoritmului destinat conducerii adaptive.

16. Elaborarea unui program de simulare (adaptiv.exe) a functionarii unui proces condus de catre un regulator adaptiv bazat pe utilizarea a doua retele neuronale cu un singur strat activ.

In concluzie se poate afirma ca solutiile propuse in lucrare, solutii ce pledeaza pentru utilizarea unor tehnici de conducere adaptiva si optimala a agregatelor aeroelectrice, conduc la rezolvarea problemelor de predictie a vitezei vintului atit pe termen scurt, cit si pe termen lung, cit si a problemei conducerii adaptive in conditii de siguranta in functionare.

Teza de doctorat are o extindere de 195 de pagini, materialul scris fiind ilustrat cu 76 figuri si 3 tabele, iar bibliografia contine 125 titluri.

BIBLIOGRAFIE

[All89] K.Alligood, J.Yorke - "Fractal Basin Boundaries and Chaotic Attractors, in Chaos and Fractals" - Proc.Symp.Appl. Math.39, A.M.S, pp.41-55,1989.

[And83] P.M.Anderson, A.Bose - "Stability Simulation of Wind Turbine Systems" - IEEE Transactions on Power Apparatus and Systems, Vol. PAS-102, No.12, pag.3791-3795, december 1983.

[Arn91] A.Arneodo, F.Argoul, s.a - "Homoclinic Chaos in Chemical Systems" - Tecnical Report 91-53, Center of Excellence in Mathematical Sciences, Cornell University 1991.

[Aro91] J.Aronson - "Chaos: A SUN-Based Program for Analyzing Chaotic Systems" - National Institute of Standards and Technology, Maryland 1991.

[Ast84] K.J. Astrom - "Teoria si aplicatiile reglarii automate adaptive" - A.M.C,vol.38, 1984

[Ast89] K.J.Astrom, B. Wittenmark - "Adaptive Control" - Addison-Wesley, Reading Mass,1989.

[Atl92] L.E.Atlas, Y.Suzuki - "Digital Systems for Artificial Networks" - Artificial Neural Networks, IEEE Press, 1992.

[Baa93] G.E.van Baars - "Experimental and Theoretical Results of the UNIWEX Wind Turbine" - European Community Wind Energy Conference, Lubeck-Tavemunde, Germany, pag.518-521, 8-12 March, 1993.

[Bab85] I.Babutia, T.L.Dragomir, I.Muresan, O.Prostean - "Conducerea automata a proceselor" - Editura Facla 1985.

[Bak89] A.G. Bakirtzis, P.S. Dokopoulos - "A Probabilistic Costing Method for the Evaluation of the Performance of Grid" - IEEE Transaction on Energy Conversion, Vol. 4, No. 1, pag. 34-40, March 1989.

[Bel85] C.Belea - "Teoria Sistemelor. Sisteme Neliniare" - Editura Didactica si Pedagogica, 1985.

[Ber88] A.L.Bertozzi - "Heteroclinic Orbits and Chaotic Dynamics in Planar Fluid Flow" - SIAM J.Math.Anal.19, pp.1271-1294, 1988.

[Bey93] H.G.Beyer, T.Degner - "Experimental Wind-Diesel System at the German Institute for Wind Energy - Layout of the Storage Unit and First Operational Experience" - European Community Wind Energy Conference, Lubeck-Tavemunde, Germany, pag. 339-342, 8-12 March, 1993.

[Bec89] K.H.Becker, M.Dorfler - "Dynamical Systems and Fractals. Computer Graphics Experiments in Pascal, Cambridge University Press, 1989.

[Bed91] T.Bedford, T.Keane - "Ergodic Theory, Symbolic Dynamics and Hyperbolic Spaces" - Oxford University Press, 1991.

[Ben91] M.Benediks, L.Carleson - "The Dynamics of the Henon Map" - Annals of Math.133,pp.73-169,1991.

[Bha90] V.N.Bhat, Th.McAvoy - "Modeling Control Process Systems via Neural Computation" IEEE Control System Magazine, April 1990.

[Blo89] A.M.Bloch, J.E.Marsden - "Controlling Homoclinic Orbits" - Center for Pure and Applied Mathematics, University of California, Berkeley, PAM-445, Feb.1989.

[Bon87] P.M.M. Bongers, T.G.van Engelen - "A Theoretical Model and Simulation of a Wind Turbine" - Wind Engineering Vol.11 No.6, pag. 344-350,1987.

[Bon89] P.M.M. Bongers, T.G.van Engelen - "Optimal Control of a Wind Turbine in a Full Load" - EWEA, pag. 345-349, 1989.

[Bud82] N.Budisan - "Sistem de conversie a energiei mecanice la turatie variabila in energie electrica de frecventa constanta, cu generator asincron, pentru centrale eoliene si alte centrale neconventionale" - Electrotehnica, Electronica si Automatica, anul 30, august 1982.

[Bud85_1] N.Budisan, V.Coifan s.a - "Consideratii privind un sistem de teleinvestigare cu calculator pentru agregatul aeroelectric MD-2/300 Semenic" - Sesiunea de comunicari tehnico-stiintifice in domeniul automatizarilor, IPA, 1985.

[Bud85_2] N.Budisan, D.Marchis s.a - "Automat pentru comanda si controlul agregatului aeroelectric AEROTIM-L1" - Conferinta Masini Hidraulice, Timisoara, 18-19.10.1985.

[Bud92_1] N.Budisan - "Concepts Concerning the Electrical Equipment for Autonomous Wind Generators With Capacitive Excited Asynchronous Generator" - Symposium and Demonstration of the Use of Wind Energy in Romania, Timisoara, 3-6 mai, 1992.

[Bud92_2] N.Budisan - "A New Concept About Electroenergetic Equipment Organization and Automatic Control Strategy of Wind-Diesel Systems" - Symposium and Demonstration of the Use of Wind Energy in Romania, Timisoara, 3-6 mai, 1992.

[Bud95] N.Budisan, T.Hentea - "Autonomous Hydro, Diesel, Biogas Single or Mixed Energetical Systems With Asynchronous Generators" -30th Intersociety Energy Conversion Engineering, Proceedings, Canada, 1995.

[Cal85] S.Calin, I.Dumitrache, s.a - "Regulatoare automate" - Editura Didactica si Pedagogica 1985.

[Cal88] S.Calin, Th.Popescu, B.Jora, V.Sima - "Conducerea adaptiva si flexibila a proceselor industriale" - Editura Tehnica, 1988

[Cha84] S.M. Chan, R.L. Cresap - "Wind Turbine Cluster Model" - IEEE Transaction on Power Apparatus and Systems, Vol. PAS-103, No.7, pag. 1692-1698, July 1984.

[Che95] F-C.Chen, H.K.Khalil - "Adaptiv Control of a Class of Nonlinear Discrete-Time Systems Using Neural Networks" - IEEE Transactions on Automatic Control, vol.40, no.5, May 1995.

[Chu93] L.O.Chua - "Global Unfolding of Chua's Circuit" - IEICE Trans.Fundamentals, vo.E76-A, No.5 May 1993.

[Cyb89] G.Cybenko - "Continuous Value Neural Networks with Two Hidden Layers Are Sufficient" - Math. Contr. Signal and Sys. vol.2, pp.303-314, 1989.

[Cu92] D.I.Curiac, N.Tudoroiu, O.Prostean - "Studiu privind predictia vitezei vintului utilizat in producerea energiei pentru o ministatiune montana" - Simpozionul international - "Instalatii pentru constructii si confort ambiental"

- vol.2 pag.178-182, Timisoara 2-3 aprilie 1992.

[Cu93] D.I.Curiac - "Facilities of the Estim 1.0 Package Used in the On-Line Multivariable Estimation of Systems" - Acta Universitatis Cibiniensis Technical series A.Electronics, Electrotechnics and Computer Science vol.X(1) pag.44-47, 1993.

[Cu93_r1] D.I.Curiac - "Modele ale agregatelor aeroelectrice" - referat de doctorat, 1993.

[Cu94_r2] D.I.Curiac - "Estimarea multivariabila on-line a parametrilor sistemelor" - referat de doctorat, 1994.

[Cu94_r3] D.I.Curiac - "Retele neuronale in conducerea proceselor" - referat de doctorat, 1994.

[Cu94] D.I.Curiac - "Reference Input Devices Based on Reccurent Neural Networks" - Buletinul Stiintific si Tehnic al Universitatii Tehnice din Timisoara, Automatica si Calculatoare, Tomul 39(53), 1994.

[Dem90] C.S. Demoulias, P.S. Dokopoulos - "Transient Behaviour and Self-excitation of Wind-driven Induction Generator after its Disconnection from the Power Grid" - IEEE Transactions on Energy Conversion, Vol. 5, No.2, pag. 272-278, June 1990.

[Des86] L.Dessaint, H. Nakra - "Propagation and Elimination of Torque Ripple in a Wind Energy Conversion System" - IEEE Tranzactions of Energy Conversion, Vol. EC-1, No.2, pag.104-112, June 1986,

[Dra89] T.L.Dragomir - "Regulatoare automate" - vol.2, Lito IPT, 1989.

[Dum93] I.Dumitrache, I.Mihu, s.a. - "Automatizari electronice" - Editura Didactica si Pedagogica 1993.

[Diaz92] J.E.Diaz, A.Peinado - "Recurrent Neural Network for Speech Recognition" - DETC Report, p.361-369, 1992.

[Dra85] I.Dragu, I.M.Iosif - "Prelucrarea Numerica a Semnalelor Discrete in Timp" - Editura Militara, 1985.

[Eng93] T.G.van Engelen, I.G.Kamphuis - "A Case Study on an Easy Definable Operating High Speed Control and Safety System" - European Community Wind Energy Conference, Lubeck- Tavemunde, Germany, pag. 538-544, 8-12 March, 1993.

[Fal90] K.L.Falconer - "Fractal Geometry: Mathematical Foundations and Applications" - Wiley & Sons, 1990.

[Gra92] H.P.Graf, L.D.Jackel - "Analog Electronic Neural Network Circuits" - Artificial Neural Networks, IEEE Press, 1992.

[Got95] S.Goto, N.Nakamura, K.Vasak - "On-Line Spectral Estimation of Nonstationary Time Series Based on AR Model Parameter Estimation and Order Selection With Forgetting Factor" - IEEE Transactions on Signal Processing, vol.43, no.6, June 1995.

[Hen76_1] M.Henon, Y.Pomeau - "Two Strange Attractors With a Simple Structure, in Turbulence and Navier Stokes Equations" - Orsay pp.29-68, Lect.Notes in Math.565, Springer Verlag 1976.

[Hen76_2] M.Henon - "A Two Dimensional Mapping With a Strange Attractor" - Comm.Math.Phys. 50, pp.69-77, 1976.

[Hin82] E.N. Hinrichsen, P.J. Nolan - "Dynamics and Stability of Wind Turbine Generators" - IEEE Transactions on Power Apparatus and Systems, Vol. PAS-101, No. 8, pag. 2640-2648, August 1982.

[Hol89] P.Holmes - "Oscillations and the Smale Horseshoe Map, in Chaos and Fractals" - Proc.Symp.Appl.Math.39, A.M.S, pp.25-39, 1989.

[Hoq93] A.Hoque, Q.Ahsan, W.C. Beattie - "A Statistical Investigation of the Behaviour of an Insolated Wind Turbine" - European Community Wind Energy Conference, Lubeck-Tavemunde, Germany, pag. 355-358, 8-12 March, 1993.

[Ich92] Y.Ichikowa, T.Sawa - "Neural Network Application for Direct Feedback Controllers" - IEEE Transactions on Neural Networks, vol.3, no.2, March 1992.

[IoV85] V.Ionescu - "Teoria sistemelor liniare" - Editura Didactica si Pedagogica 1985.

[IoT82] T.Ionescu - "Sisteme si echipamente pentru conducerea proceselor" - Editura Didactica si Pedagogica 1982.

[Kar93] G.Kariniotakis, G.Stavrakakis, E.Nogaret, M.Bordier - "Advanced Modelling and Identification of Automous Wind-Diesel Power System. Application on the French Island Desirade" - European Community Wind Energy Conference, Lubeck-Tavemunde, Germany, pag. 347-350, 8-12 March, 1993.

[Kau94] H.Kaufman, I.Bar-Kana, K.Sobel - "Direct adaptive control algorithms" - Springer-Verlag, 1994

[Lai94_1] Y-C.Lai, C.Grebogi - "Converting Transient Chaos into Sustained Chaos by Feedback Control" - The Academical Physical Society, Physical Review E, vol.49, no.2, feb.1994.

[Lai94_2] Y-C.Lai - "Controlling Chaos" - Computers in Physics, vol.8,no.1, feb.1994.

[Lai94_3] Y-C.Lai, C.Grebogi - "Synchronization of Spatiotemporal Chaotic Systems by Feedback Control" - The Academical Physical Society, Physical Review E, June 1994.

[Lef85] S.Lefebvre, L.Dessaint - "Simulator Study of a Vertical Axis Wind Turbine Generator Connected to a Small Hydro Network" - IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, No.5, may 1985, pag.1095-1101.

[Lei89] W.E.Leithead, J.Wilkie - "Simulation of Wind Turbine by Simples Models" - EWEAC, pag.336-340, 1989.

[Lju87] L. Ljung - "System Identification - Theory for the User"- Prentice Hall Inc. Englewood Cliffs. New Jersey. 1987.

[Lju85] L. Ljung - "On the estimation of transfer functions" - Automatica, vol.21, pp.677-696, 1985

[Lju86_1] L. Ljung - "Parametric methods for identification of transfer functions of linear systems" - Advances in Control, vol.24, Academic Press, New York, 1986.

[Lju86_2] L. Ljung - "System Identification Toolbox: Manual, The Math Works" - Inc., Sherborn, Mass, 1986

[Lor63] E.N.Lorenz - "Deterministic Nonperiodic Flow" - J.Atmos Sci. 20, pp.130-141, 1963.

[Mad91] P.H.Madsen, G.M.McNerney - "Frequency Domain Modeling Of Free Yaw Response of Wind Turbines to Wind Turbulence" - Journal of Solar Energy Engineering, vol.113, pag.102-111, may 1991.

[Mah92] M.A.C.Maher, S.P.DeWeerth, s.a - "Implementing Neural Architectures Using Analog VLSI Circuits" - Artificial Neural Networks, IEEE Press, 1992.

[Mar91] M.Marek, I.Schreiber - "Chaotic Behaviour of deterministic Dissipative Systems" - Cambridge Univ. Press, 1991.

[Mas92] A.Masaki, Y.Hirai, M.Yamada - "Neural Networks in CMOS: A Case Study" - Artificial Neural Networks, IEEE Press, 1992.

[Mie90] A.Mielke, P.Holmes, O.O'Reilly - "Cascades of Homoclinic Orbits to, and Chaos Near a Hamiltonian Saddle-Center" - Math.Sci.Inst. Cornell Univ., Technical Rep. no.85, 1990.

[Moh95] O.Mohammed, D.Park, s.a - "Practical Experiences With Adaptive Neural Network Short-Term Load Forecasting System" - IEEE Transactions on Power Systems, vol.10, no.1, Feb.1995.

[Mul90] Muller B., Reinhardt J., "Neural Networks" Spinger-Verlag 1990.

[Mur83] A. Murdoch, J.R. WinKelman - "Control Design and Performance Analysis of a 6 MW Wind Turbine-Generator" - IEEE Transaction on Power Apparatus and Systems, Vol. PAS-102, No. 5, pag. 1340-1347, May 1983.

[Nar92] K.S.Narendra, K.Parthasarathy - "Identification and Control of Dynamical Systems Using Neural Networks" - Artificial Neural Networks, IEEE Press, 1992.

[Nat87] K.Natarajan, A.M.Sharaf - "Modeling and Control Design for Wind Energy Power Conversion Scheme Using Self-Excited Induction Generator" - IEEE Transactions on Energy Conversion, Vol.EC-2, No.3, pag.506-512, september 1987.

[Ngu92] D.H.Nguyen, B.Widrow - "Neural Networks for Self-Learning Control Systems" - Artificial Neural Networks, IEEE Press, 1992.

[Ott90] E.Ott, C.Grebogi, J.A.Yorke - "Controlling Chaos" - The Academical Physical Society, Physical Review Letters, vol.64, no.11, March 1990.

[Pal93] SK.Pal, S.Mitra, "Multilayer Perceptron, Fuzzy Sets, and Clasification", IEEE Trans. on Neural Network, Vol.3, No.3, May 1993 pag 498-504.

[Pas94] M.Paskota, A.I.Mees, K.L.Teo - "Stabilizing Higher Periodic Orbits" - International Journal of Bifurcation and Chaos, vol.4, No.2, 1994.

[PeE92] E.Petrisor - "Sisteme dinamice haotice" - Litografia Universitatii din Timisoara 1992.

[PeV92] V.Petrov, B.Peng, K.Showalter - "A Map-Based Algorithm for Controlling Low-Dimensional Chaos" - J.Chem.Phys., vol.96, No.10,15 may 1992.

[Pop91] Th. Popescu, S. Demetriu - "Practica modelarii si predictiei seriilor de timp" - Editura Tehnica Bucuresti 1991.

[Pro85] O.Prostean, I.Muresan - "Tehnici de identificare si modelare" - vol.1 si 2, Lito IPTVT 1985.

[Pro91_1] O.Prostean, B.Lustrea, D.I.Curiac - "Some Aspects of Hydrogenerator Excitation Control Using an Adaptive Regulator" - Buletinul Institutului Politehnic Iasi Tomul XXXVII(XLI), Fasc.1-4, Sectia VI Automatica si

Calculatoare, pag.39-43, 1991.

[Pro91_2] O.Prostean, B.Lustrea, D.I.Curiac - "Asupra estimarii recursive a parametrilor generatorului sincron utilizat in reglajul autoacordabil al excitatiei" - Lucrarile celui de al III-lea simpozion "Structuri, Algoritmi si Echipamente de conducere a proceselor industriale" vol.1 pag.313-318, Iasi 25-26 octombrie 1991.

[Pro94] O.Prostean, D.I.Curiac, I.Filip - "Experience with Adaptive Control in a Stochastic Environment with Application for the Power Systems" - Simpozionul international - ConTI'94, vol.3, pag.13-22, Timisoara noiembrie 1994.

[Ric90] P.Richter, H.J.Scholtz, A.Wittek - "A Breathing Chaos" - Nonlinearity 3, pp.59-67, 1990.

[Rue89] D.Ruelle - "Chaotic Evolution and Strange Attractors" - Cambridge Univ. Press, 1989.

[Sac92] E.Sackinger, B.E.Boser - "Application of the ANNA Neural Network Chip to High - Speed Character Recognition" - IEEE Trans. on Neural Network, p.498-504, Vol.3,No.3, May.1992.

[Sav95] A.V.Savkin, I.R.Petersen - "Recursive State Estimation for Uncertain Systems With an Integral Quadratic Constraint" - IEEE Transactions on Automatic Control, vol.40, no.6, June 1995.

[Sch83] R.A.Schlueter, G.L.Park - "Methods of Reducing Wind Power Changes from Large Wind Turbine Arrays" - IEEE Transactions on Power Apparatus and Systems, Vol. PAS-102, No.6,pag.1642-1650, june 1983.

[Sch84] R.A.Schlueter, G.L.Park - "Simulation and assessment of Wind Array Power Variations Based on Simultaneous Wind Speed Measurements" - IEEE Transactions on Power Apparatus and Systems, Vol. PAS-103, No.5, pag.1008-1016, may 1984.

[Sch86] R.A.Schlueter, G.Sigari - "Wind Array Power Prediction for Improved Operating Economics and Reliability" - IEEE Transactions on Power Systems, Vol. PWRS-1, No.1, pag.137-142, february 1986.

[ScR91] R.M.Schori - "Chaos: An Introductions to Some Topological Aspects" - Continuum Theory and Dynamical Systems pp.142-162, Contemporar Mathematics 117, A.M.S, 1991.

[She91] Y.Sheinman, A.Rosen - "A Dynamic Model for Performance Calculations of Grid-Connected Horizontal Axis Wind Turbines" - Wind Engineering Vol.15 No.4, pag.211-239,1991.

[She93] Y.Sheinman, A.Rosen - "Modeling a Wind Turbine in a Turbulent Wind" - European Community Wind Energy Conference, Lubeck-Tavemunde, Germany, pag. 495-498, 1993.

[Shi90] T.Shinbrot, E.Ott, C.Grebogi, J.A.Yorke - "Using Chaos to Direct Trajectories to Targets" - The Academical Physical Society, Physical Review Letters, vol.65, no.26, Dec 1990.

[Sim89] V.Sima - "Tehnici de factorizare in algoritmii de conducere adaptiva" - A.M.C, vol.51, 1989.

[Sma63_1] S.Smale - "Stable Manifolds for Differential Equations and Diffeomorphism" - Ann.Scuola Norm.Sup.Pisa,1, pp.97-116,1963.

[Sma63_2] S.Smale - "Diffeomorphisms With Many Periodic Points, in Differential

and Combinatorial Topology" - Priceton Univ.Press, pp.63-80, 1963.

[Sod87_1] T. Soderstrom - Model structure determination. In Encyclopedia of Systems and Control, Pergamon Press Elmsford, New York, 1987

[Sod87_2] T. Soderstrom, P. Stoica - System Identification. Prentice-Hall, Englewood Cliffs, New Jersey, 1987

[Sun83] R.M.Sundar, J.P.Sullivan - "Performance of Wind Turbines in Turbulent Atmosphere" - Solar Energy Vol.31 No.6, pag.567- 575, 1983.

[Sym83] D.P. Symanski - "The Operating Experiences and Performance Characteristics During the First Year of Operation of the Crotched Mt. New Hampshire Windfarm" - IEEE Transaction on Power Apparatus and Systems, Vol. PAS-102, No. 6, pag. 1637-1641, June 1983.

[Tao95] G.Tao, P.V.Kokotovic - "Continuous-Time Adaptive Control of Systems With Unknown Backlash" - IEEE Transactions on Automatic Control, vol.40, no.6, June 1995.

[Tar94] Z.Taroczkai - "Geometric Method for Stabilizing Unstable Periodic Orbits" - The Academical Physical Society, Physical Review Letters, vol.51, no.2, feb.1995.

[Ter80] M.Tertisco, P.Stoica - "Identificarea si Estimarea Parametrilor Sistemelor" - Editura Academiei,1980.

[Ter87] M.Tertisco, P.Stoica, Th.Popescu - "Identificarea asistata de Calculator a Sistemelor" - Editura Tehnica,1987

[The93] *** - "THERMIE - Wind Energy Technology Projects" - Directorate - General for Energy (DG XVII), Commission of the European Communities, 1993.

[Tod94] Toderean G., Costeiu M., Giurgiu M. - "Retele Neuronale" - Microinformatica Cluj 1994.

[Tud90] Tudoroiu N, "Optimizarea proceselor neliniare de la Combinatul Chimic Craiova" - Teza de doctorat, U.Craiova 1990.

[Tud93_1] Tudoroiu N, Curiac D, Prostean O. - "Automatizari complexe" - Editura Mirton Timisoara 1993.

[Tud93_2] N.Tudoroiu, D.I.Curiac - "Teoria Sistemelor de Reglare Automata Liniare - Abordare Aplicativa" - Editura Mirton Timisoara 1993.

[Tud93_3] N.Tudoroiu, O.Prostean, D.I.Curiac, I.Filip - Teoria Sistemelor de Reglare Automata Neliniare, Discrete si Optimale - Abordare Aplicativa" - Editura Mirton Timisoara 1993.

[Tsu93] T.Tsubata, H.Kawabata, s.a. - "Intermittency of Recurrent Neuron and Its Network Dynamics" - IEICE Trans.Fundamentals, vo.E76-A, No.5 May 1993.

[Vak90] A.Vakakis, J.Burdick - "Chaotic Motions in the Dynamics of a Hopping Robot" - Proc.of IEEE Int.Conf.Robotics and Automation, Cincinnati Ohio, pp.1464-1469, 1990.

[Vin91] T.Vincent, J.Yu - "Control of a Chaotic System" - Dynamics and Control 1, pp.35-52, 1991.

[Wah86_1] B. Wahlberg - "On model reduction in system identification" - Proc.Americ. Control Conf. Seattle, Wash., 1986

[Wah86_2] B. Wahlberg, L.Ljung - Design variables for bias distribution in transfer function estimation. IEEE Trans. Automatic Control, vol.AC-31.pp. 134-144, 1986

[WaX84] W.Xifan,D.Hui-Zhu - "Reability Modeling of Large Wind Farms and Associated Electric Utility Interface Systems" - IEEE Transactions on Power Apparatus and Systems, Vol. PAS-103, No.3, pag. 569-575, March 1984.

[WaH94] H.Wang, M.Brown, C.J.Harris - "Neural Network Modelling of Unfnown Nonlinear Systems Subject to Immeasurable Disturbances" - IEEE Proc.- Control Theory Appl., Vol.141, No.4, July 1994.

[War94] J.Warncke, M.Bauer, W.Martienssen - "Multiparameter Control of High-Dimensional Chaotic Systems" - Europhysics Letters, 25(5), pp.323-328, 1994.

[Wer93] P.J.Werbos - "Quantum Theory, Computing and Chaotic Solitons" - IEICE Trans.Fundamentals, vo.E76-A, No.5 May 1993.

[Wid92] B.Widrow, M.A.Lehr - "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation" - Artificial Neural Networks, IEEE Press, 1992

[Wig88] S.Wiggins - "Global Bifurcation and Chaos: Analytic Methods" - Appl.Math.Sci. 73, Springer Verlag 1988.

[Wig90] S.Wiggins - "Introduction to Applied Nonlinear Dynamical Systems and Chaos" - Springer Verlag 1990.

[Wil90] J.Wilkie, W.E.Leithead - "Simulation of Wind Turbines by Simple Models" - Wind Engineering Vol.14, No.4, pag.247- 274,1990.

[Yeg86] S.S. Yegna Narayanan, V.J. Johnny - "Contributions to the Steady State Analysis of Wind-turbine Driver Self-excited Induction Generators" - IEEE Transactions on Energy Conversion, vol.EC-1, No.1, pag. 169-175, March 1986.