CREAREA UNEI BIBLIOTECI SIMULINK PENTRU EXPLOATAREA REȚELELOR NEURONALE ÎN IDENTIFICARE

Marius Kloetzer
Dan Ardelean
Octavian Păstrăvanu

Facultatea de Automatică și Calculatoare Iași

Email: mkloetzer@ac.tuiasi.ro, adan@mail.dntis.ro, opastrav@ac.tuiasi.ro

Rezumat: Este prezentată o bibliotecă de instrumente software performante, care extind utilizarea mediului Simulink în domeniul identificării bazate pe modele de tip rețea neuronală. Facilitățile nou create permit realizarea antrenării prin simpla operare cu blocuri și selectarea unui număr de opțiuni afișate în cutiile de dialog asociate acestora, după principiul "pre-designed modules". Această strategie este curent utilizată în software-ul tehnico-științific pentru a asigura simplitatea modului de definire a aplicațiilor și flexibilitatea organizării experimentelor, prin reducerea substanțială a efortului de programare în maniera clasică a scrierii de cod. Eficiența ridicată în exploatare rezultă, pe de o parte, din ușurința cu care se manipulează datele (pentru identificare off- și on-line), iar, pe de altă parte, din posibilitatea antrenării prin mai mulți algoritmi numerici. Dezvoltată pentru a veni în sprijinul activității de cercetare orientată pe aplicarea rețelelor neuronale în automatică, biblioteca permite atât construirea unor modele dinamice, bazate pe arhitecturi de tip ADALINE, MLP și RBF, cât și utilizarea acestora în simulare, oferind, totodată, un valoros material didactic, pentru diverse forme de instruire practică în cadrul învățământului universitar și postuniversitar din profilul științei sistemelor și calculatoarelor. Lucrarea include un număr de studii de caz ce ilustrează totala compatibilitate cu mediul Simulink a noilor instrumente, manevrarea comodă a bibliotecii, precum și calitatea modelelor rezultate din identificare.

Cuvinte cheie: rețele neuronale, identificare, simulare, instrumente Simulink.

1. Introducere

Identificarea bazată pe modele neuronale s-a conturat ca direcție de cercetare la începutul anilor '90, drept urmare a investigațiilor matematice asupra proprietăților de aproximare ale rețelelor neuronale de tip MLP (Multilayer Perceptron) [5], [10] și de tip RBF (Radial-Basis Function) [11]. Evoluția noului domeniu a fost impulsionată de contribuțiile remarcabile, din aceeași perioadă, ale unor nume de prestigiu în automatică, ca de exemplu [9], [13], [18], care au deschis drumul utilizării rețelelor MLP în identificarea neliniară și [3], care s-au orientat pe arhitecturi de tip RBF. În acest context, merită subliniat faptul că aplicarea în automatică a tehnicilor specifice rețelelor neuronale a fost dominată, pe întreaga durată a anilor '90, de exploatarea topologiilor MLP cu noduri sigmoidale în stratul (straturile) ascuns(e). Totuși, un număr de cercetători au continuat studierea aplicabilității rețelelor de tip RBF în identificarea și conducerea sistemelor, cum este cazul lucrărilor de referință [13], [14], [15].

Pe de altă parte, firmele producătoare de software tehnico-științific au demarat dezvoltarea de facilități specifice rețelelor neuronale, astfel încât, apariția în 1992 a primei versiuni a Neuronal Network Toolbox (NNT) încorporată în mediul MATLAB 4.2 a avut un impact major asupra interesului acordat de către automatiști acestei direcții de cercetare.

Întrucât NNT, de la prima versiune până la cea mai recentă [21], a fost conceput să acopere o arie cât mai largă de aplicații ale rețelelor neuronale, proiectanții săi nu și-au propus dezvoltarea concomitentă a unor blocuri Simulink, dedicate construcției modelelor dinamice. O inițiativă valoroasă în această direcție trebuie semnalată în studiile practice din monografia [7], care prezintă realizarea unui bloc Simulink destinat antrenării on-line, prin metoda propagării inverse, a unei configurații dinamice, bazată pe topologie MLP, destinată identificării neliniare.

Spre deosebire de abordarea restrânsă din [7], instrumentele soft, discutate în lucrarea de față, vizează un cadru mult mai larg de exploatare a rețelelor neuronale în Simulink, versiunea 4.0.1 [20], în scopul construirii modelelor dinamice, permițând utilizarea topologiilor ADALINE, MLP și RBF, selectarea a diverse strategii de organizare a datelor pentru identificare, precum și utilizarea mai multor tehnici de antrenare în conformitate cu topologia rețelei. Prin crearea unor atare instrumente, se realizează un progres notabil în ceea ce privește simplitatea manevrării modelelor neuronale în experimente de identificare, care, astfel, pot fi conduse direct în mediul Simulink (considerând inclusiv situația aplicațiilor de timp real, pentru care se poate face apel la serviciile Real-Time Workshop). Merită de amintit faptul că preocupările noastre în direcția dezvoltării de facilități Matlab-Simulink pentru utilizarea rețelelor neuronale în automatică au avut în vedere, în mod constant, și aspectul educațional, după cum reiese din [6], [12], experiența acumulată permițând crearea bibliotecii prezentate în lucrarea curentă. Validitatea acestor preocupări a primit, indirect, o confirmare importantă chiar din partea firmei producătoare, The MathWorks, care, începând cu versiunea Simulink comercializată în 2001, a

inclus un bloc pentru identificare neuronală, bazată pe arhitectura MLP. În raport cu facilităţile oferite de biblioteca noastră, implementarea furnizată de firma The MathWorks nu conţine blocuri structurate pe topologiile ADALINE şi RBF şi nici nu permite utilizarea strategiilor de antrenare on-line (pe eşantioane).

Organizarea materialului curent este concepută astfel: Secţiunea a 2a furnizează o privire de ansamblu asupra problematicii identificării bazate pe modele de tip reţea neuronală; Secţiunea a 3a este dedicată prezentării blocurilor Simulink create şi a facilităţilor de exploatare; Secţiunea a 4a ilustrează aspectele numerice semnificative ale antrenării şi comentează calitatea modelelor obţinute prin intermediul a două exemple relevante; Secţiunea a 5a prezintă o comparaţie între performanţele modelelor neurale de tip RBF şi cele de tip MLP; Secţiunea a 6a formulează câteva concluzii privitoare la eficienţa dezvoltărilor software propuse.

2. Scurtă trecere în revistă a problematicii identificării pentru modele de tip neuronal

Utilizarea reţelelor neuronale în identificare permite construirea de modele "black-box", liniare sau neliniare, care se bazează pe rezultatele teoretice privind capacitatea de aproximare a arhitecturilor feedforward, formulate în [5], [11] şi [22] pentru topologii ADALINE, MLP şi respectiv RBF. Un interes practic deosebit prezintă obţinerea modelelor discrete de tip NNARX [4], [16] cu perioada de eşantionare T:

$$\hat{y}(kT \mid \boldsymbol{\theta}) = f(y((k-1)T),\ldots,y((k-n)T),u((k-d)T),\ldots,u((k-d-m)T)); k,m,n,d \in \mathbf{N} \quad (1)$$

unde:

$$f: \mathbf{R}^{n+m+1} \rightarrow \mathbf{R} \quad (2)$$

este o aplicaţie (liniară sau neliniară netedă) ce descrie transferul intrare-ieşire al unei reţele neuronale statice, cu topologie adecvată, iar θ notează vectorul parametrilor reţelei. În relaţia (1), u şi y desemnează intrarea şi, respectiv, ieşirea procesului care constituie obiectul identificării, iar $\hat{y}$ notează predicţia realizată de NNARX pe baza regresorilor din (1). Indiferent de topologia reţelei, identificarea pe baza schemei serie-paralel (conform denumirii utilizate în [13]) din figura 1 asigură stabilitatea algoritmului iterativ de minimizare a funcţiilor obiectiv, definite cu ajutorul erorii $e(kT) = y(kT) - \hat{y}(kT|\theta)$. Formularea concretă a funcţiilor obiectiv este corelată cu modul de obţinere şi de exploatare a datelor experimentale, corespunzătoare procesului. În acest sens, literatura evidenţiază două categorii de metode de antrenare a reţelei neuronale din schema serie-paralel din figura 1:

- pe lot (batch) sau off-line, care utilizează funcţii obiectiv de forma (până la o constantă multiplicativă):

$$J = \frac{1}{N}\sum_{k=1}^{N} e^2(kT) \le \varepsilon \quad (3)$$

unde N notează numărul total de elemente disponibile în lot;

- pe eşantioane (pattern) sau on-line, care utilizează funcţii obiectiv de forma (până la o constantă multiplicativă):

$$J(iT) = \frac{1}{N_E}\sum_{k=i}^{i+N_E-1} e^2(kT) \le \varepsilon \,,\ i \in \mathbf{N} \quad (4)$$

unde N_E notează numărul de eşantioane conţinute în fereastra mobilă de selectare a elementelor şi ε eroarea dorită.

Minimizarea operează în spaţiul parametrilor reţelei (ponderi şi deplasări) şi poate fi condusă prin tehnicile clasice de optimizare fără restricţii (de exemplu [19]), adaptate la specificul organizării matriceal-vectoriale a parametrilor aplicaţiei f. În limbajul propriu reţelelor neuronale, determinarea parametrilor lui f ca rezultat al minimizării funcţiilor obiectiv (3) sau (4) este referită drept proces de antrenare sau de învăţare.

Dacă pentru arhitecturile ADALINE şi MLP pot fi aplicate ambele metode de antrenare, pentru topologia RBF numai antrenarea batch are sens. Merită de menţionat faptul că rezultatele teoretice, formulate în [18], demonstrează convergenţa (cu precizarea ordinului) pentru funcţii obiectiv de forma (4), specifice procedurilor de antrenare on-line, fapt care a încurajat dezvoltarea aplicaţiilor de timp real pentru identificarea bazată pe modele neuronale. În ceea ce priveşte alegerea unui algoritm adecvat de antrenare, trebuie făcută diferenţierea clară între cazurile ADALINE şi MLP, unde algoritmul operează pe o topologie cu un număr fixat de noduri, şi cazul RBF, unde algoritmul în sine generează numărul de noduri ascunse. Mai mult chiar, antrenarea pe topologii de tip RBF garantează întotdeauna satisfacerea condiţiei (3) în sensul erorii medii pătratice dorite ε, în timp ce antrenarea topologiilor ADALINE şi MLP se poate opri fără întrunirea condiţiilor (3) sau (4) atunci când

se depăşeşte numărul maxim de iteraţii (epoci) stabilit de utilizator. În cazul MLP, există mai mulţi algoritmi de antrenare de tip gradient şi/sau metode de tip Newton [1], [2], [8], [17], fapt ce permite confruntarea directă a performanţelor lor. Parametrii rezultaţi din identificare au înţelesuri diferite, după cum urmează: (*i*) ponderile pentru ADALINE (deplasamentul neuronului fiind nul), (*ii*) ponderile şi deplasamentele tuturor straturilor pentru topologia MLP, (*iii*) dispersiile şi centrele nodurilor pentru primul strat şi ponderea şi deplasamentul pentru neuronul din al doilea strat pentru topologia RBF.

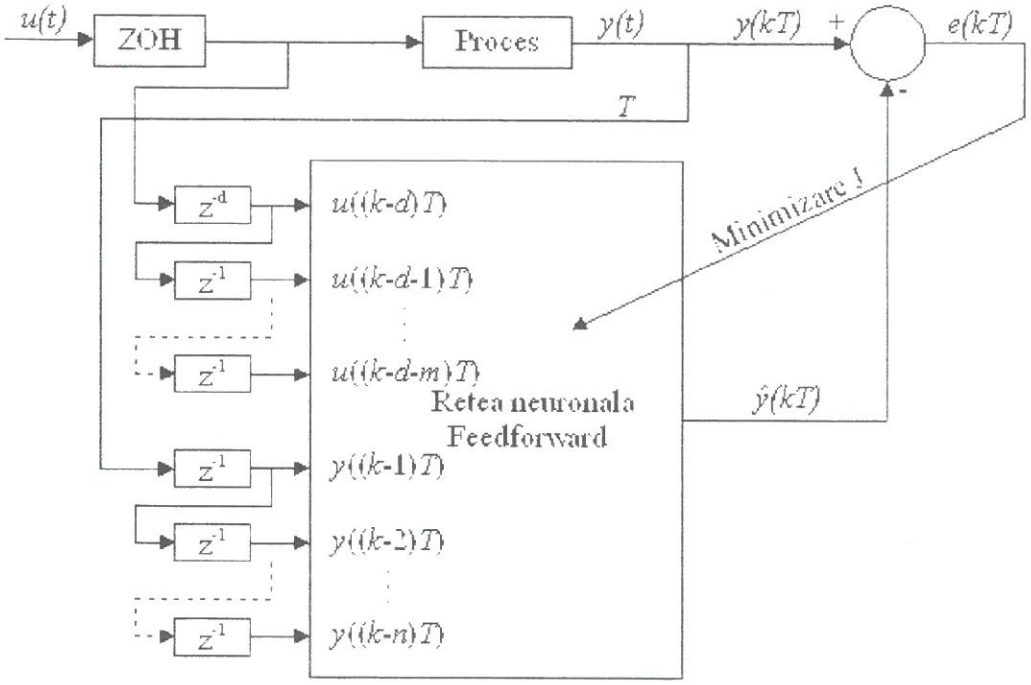


Figura 1. Schema de identificare serie-paralel corespunzătoare unui model de forma (1)

3. Blocuri Simulink pentru identificarea bazată pe modele neuronale

Metodologia de realizare a blocurilor SIMULINK se bazează pe conceptul specific de funcţie S, introdus de producătorii software-ului, ce se caracterizează printr-o standardizare a arhitecturii care asigură interconectarea modulelor [20]. În dezvoltarea blocurilor pentru identificare cu modele neuronale, construirea funcţiilor de tip S s-a bazat pe scheletul *sfuntmpl.m*.

Versiunea 4.0.1 a NNT [21] pune la dispoziţie funcţii specializate pentru definirea topologiei, realizarea antrenării şi simularea transferului intrare-ieşire a reţelelor neuronale. Astfel, pot fi descrise într-o manieră unitară (care operează cu structuri) arhitecturi cu unul sau mai multe straturi şi diferite tipuri de neuroni. Totodată, antrenarea poate fi controlată prin diverşi algoritmi, de ordinul unu sau doi, pentru care NNT furnizează implementări eficiente, cu sintaxe de apel similare.

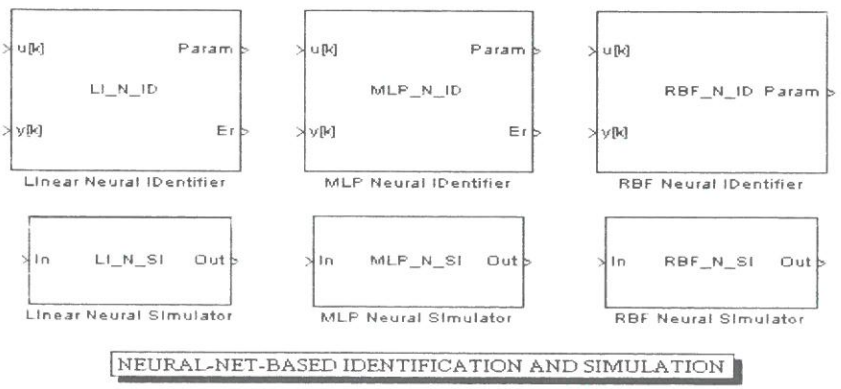


Figura 2. Organizarea bibliotecii Simulink pentru identificare bazată pe modele neuronale

În crearea bibliotecii de module Simulink [20] cu organizarea din figura 2, s-au luat în considerare separat cazurile când funcţia f din (2) este liniară, respectiv netedă, pe porţiuni, avându-se în vedere construirea a şase blocuri distincte după cum urmează:

i. pentru f liniară – ***LI_N_ID*** (LInear Neural IDentifier), bloc folosit pentru antrenarea unei reţele ADALINE şi ***LI_N_SI*** (LInear Neural SImulator), bloc care serveşte în simularea transferului intrare-ieşire a unui model neural cu topologie ADALINE;

ii. pentru f neliniară ***MLP_N_ID*** (MLP Neural IDentifier), bloc care serveşte în antrenarea reţelelor de tip MLP şi ***MLP_N_SI*** (MLP Neural SImulator) pentru simularea transferului intrare-ieşire a unui model neural cu topologie MLP;

iii. tot pentru f neliniară ***RBF_N_ID*** (RBF Neural IDentifier), bloc care serveşte în antrenarea pe lot a reţelelor de tip RBF şi ***RBF_N_SI*** (RBF Neural SImulator) pentru simularea transferului intrare-ieşire a unui model neural cu topologie RBF.

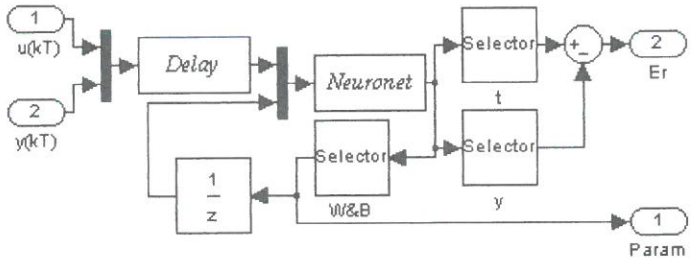


Figura 3. Arhitectura modulară a blocului *MLP_N_ID* pentru cazul (*ii*)

Conectarea blocurilor de antrenare la o diagramă Simulink este identică pentru cazurile (*i*) - (*iii*), dar casetele de dialog ataşate diferă. Figura 3 prezintă arhitectura modulară a blocului ***MLP_N_ID*** pentru cazul (*ii*), care este identică arhitecturii blocului ***LI_N_ID*** pentru cazul (*i*), excepţie făcând modulul ***Neuronet***. În cazul (*i*), ***Neuronet*** este ***ADALINE*** cu deplasament nul, în timp ce în cazul (*ii*), ***Neuronet*** are două straturi – primul sigmoidal (cu numărul de neuroni specificat în căsuţa de dialog), iar cel de-al doilea liniar. De asemenea, diferă şi algoritmii de antrenare folosiţi (algoritmi care pot fi selectaţi prin intermediul casetei de dialog) deoarece ei sunt asociaţi topologiei particulare ***Neuronet***. Modulele ***Selector*** din figura 3 organizează cozile de la ieşirea blocului. În cazul (*iii*), arhitectura modulară a ***RBF_N_ID*** conţine doar modulele ***Delay*** şi ***Neuronet***. Pentru toate blocurile de antrenare (*i*) - (*iii*), conexiunile de intrare (inports #1, #2) la momentul kT ($T>0$, $k \in \mathbb{N}$) sunt intrarea actuală $u(kT)$ (inport #1) şi ieşirea $y(kT)$ (inport #2) a procesului de identificat. Modulul ***Delay***, din figura 3, generează în mod automat regresorii necesari în modelul (1) prin implementarea unor cozi FIFO, de mărime adecvată, în concordanţă cu informaţiile din caseta de dialog. Conexiunile de ieşire (outports #1, #2) la momentul kT ($T>0$, $k \in \mathbb{N}$) furnizează pentru toate blocurile de antrenare parametrii rezultaţi din antrenare (outport #1), iar, pentru cazurile (*i*), (*ii*), blocurile furnizează eroarea actuală a modelului.

Toate blocurile prezentate în această lucrare permit antrenarea pe lot (*off-line*, în concordanţă cu (3)), iar două dintre blocurile de antrenare, (*i*) şi (*ii*), permit antrenarea pe eşantioane (*on-line*) cu numărul de eşantioane specificat (utilizând $J(iT)$ în (4), cu N_E luat din caseta de dialog).

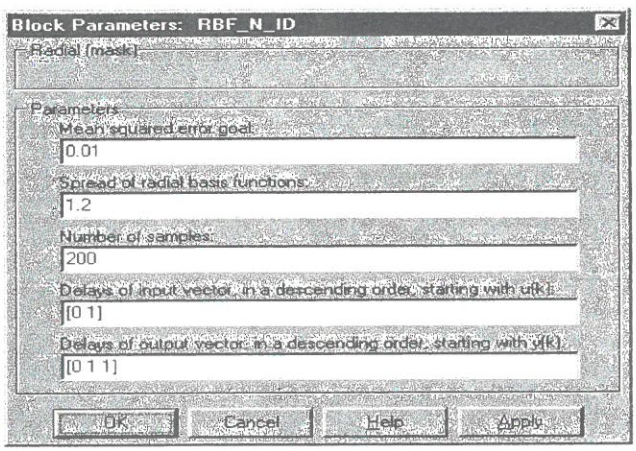


Figura 4. Caseta de dialog pentru *RBF_N_ID*

Figura 4 ilustrează principiul de comunicare cu utilizatorul prin intermediul casetelor de dialog, referirea concretă având în vedere blocul ***RBF_N_ID*** pentru cazul (*iii*). Pentru modulele ***LI_N_ID*** în cazul (*i*) şi ***MLP_N_ID*** în cazul (*ii*), casetele de dialog permit introducerea informaţiilor specifice antrenării unei reţele cu topologia respectivă.

Ca mod general de abordare, în găsirea regresorilor optimi pentru mărimea de intrare şi, respectiv, de ieşire (corespunzători modelului NNARX (1) sau, echivalent, figurii 1) este recomandată o strategie „bottom-up" în sensul că arhitectura reţelei va fi iniţializată cu un număr redus de regresori, număr ce va fi incrementat succesiv până la obţinerea rezultatului dorit.

Arhitectura internă a blocurilor de simulare pentru cazurile (*i*) - (*iii*) este bazată pe module asemănătoare cu ***Delay*** şi ***Neuronet*** prezentate mai sus. Parametrii reţelei se pot seta manual sau pot fi încărcaţi automat din spaţiul de lucru Matlab.

4. Studii de caz

Deşi utilizarea modelelor neuronale este posibilă, îndeosebi, în cazul neliniar, luăm în discuţie, pentru început, construcţia unui model liniar, deoarece acesta permite comparaţii relevante (de natură parametrică) cu reprezentările standard de tip funcţie de transfer, discretă în timp. Al doilea exemplu ilustrează construcţia unui model neliniar pentru care discuţia privind validitatea se limitează doar la analiza comparativă a răspunsurilor obţinute de la proces şi, respectiv, model, în condiţiile aceloraşi semnale de test, aplicate la intrare. În nici unul dintre cazuri nu abordăm problematica validării modelelor cu ajutorul testelor statistice.

Pentru ambele exemple, procesul supus identificării a fost simulat ca funcţionare în mediul Simulink, antrenarea reţelelor nefăcând însă uz de cunoaşterea ordinului modelelor de simulare (ipoteză absolut firească în desfăşurarea aplicaţiilor reale). Totodată, au fost considerate, separat, două situaţii pentru fiecare din exemplele ce urmează a fi detaliate mai jos:

a) ieşirea procesului nu este afectată de perturbaţii;

b) ieşirea procesului este afectată aditiv de perturbaţii de tip Gaussian, care pot atinge 1% din amplitudinea maximă a semnalului util (corespunzând, de exemplu, zgomotului de conversie al unui convertor A/D cu rezoluţie scăzută).

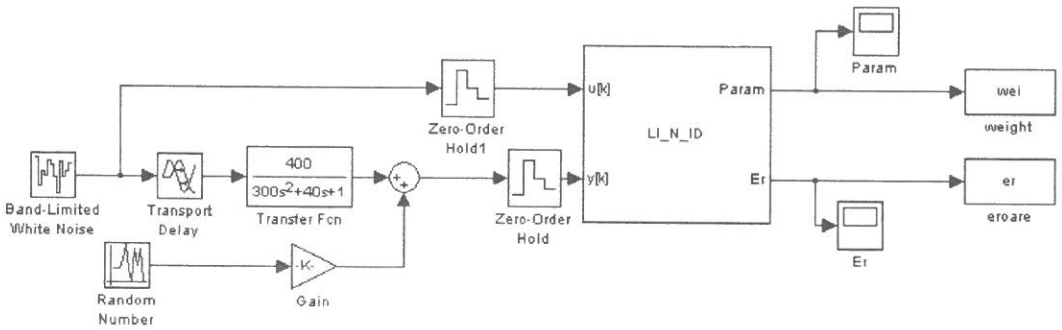


Figura 5. Schema Simulink de identificare a modelului liniar corespunzător exemplului din secţiunea 4.1

4.1. Exemplu ilustrativ pentru construcţia unui model liniar

În Figura 5, se prezintă schema Simulink (aferentă situaţiei (*b*), menţionată la începutul secţiunii curente) utilizată pentru construirea unui model liniar de forma (1) cu T=1 secundă, cu ajutorul blocului ***LI_N_ID***. Procesul ce face obiectul identificării este descris în Simulink prin funcţia de transfer continuă în timp:

$$G(s)=\frac{400e^{-15s}}{300s^2+40s+1}, \tag{5}$$

procesul fiind excitat cu o sursă de zgomot alb de bandă limitată, adecvat aleasă. Singurele informaţii presupuse cunoscute apriori sunt comportarea liniară a procesului şi valoarea timpului mort (identificabil, de exemplu, de pe un răspuns la semnal treaptă), adică d=15 în (1) pentru T=1 secundă.

Selectăm drept complet edificatoare doar o parte din rezultate, conform tabelului 1 pe care le exprimăm (dată fiind liniaritatea modelului) sub formă de funcţii de transfer discrete (ale căror coeficienţi sunt, până la un semn

cunoscut, chiar parametrii furnizați de blocul ***LI_N_ID***). Această exprimare facilitează analiza calității modelului provenit din identificare, prin comparație directă cu discretizată cu T=1 secundă a lui $G(s)$, obținută prin metoda Euler predictor:

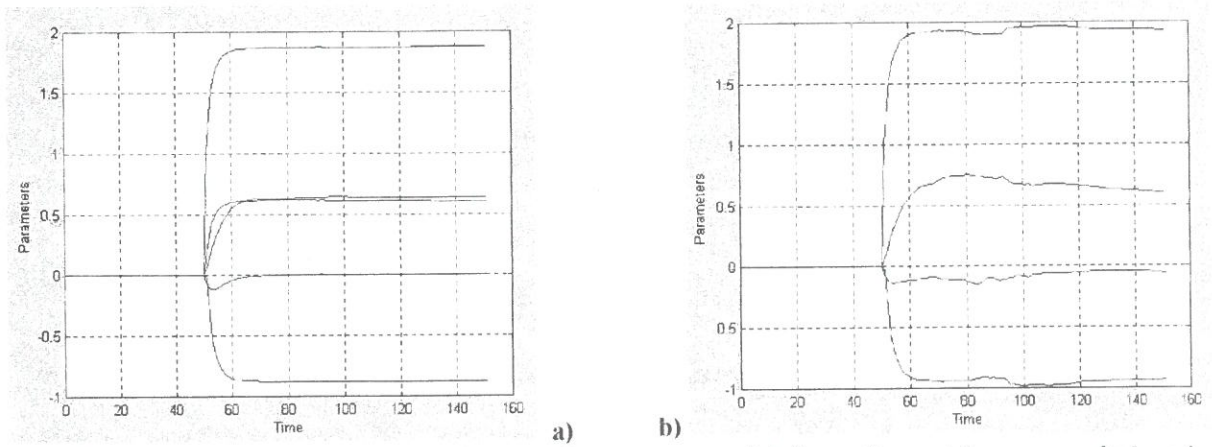
$$\widetilde{G}(z^{-1})=z^{-15}\frac{0.6378z^{-1}+0.6101z^{-2}}{1-1.872z^{-1}+0.8752z^{-2}} \tag{6}$$

Toate cazurile comentate se referă la aceleași condiții de antrenare on-line cu N_E=50 eșantioane în funcția obiectiv $J(iT)$ din (4), maxim 100 epoci/fereastră și rată de învățare variabilă.

Tabelul 1. Prezentare sintetică a rezultatelor antrenării pentru opt teste de identificare selectate ca relevante pentru secțiunea 4.1

				Cazul (*a*) - Ieșire neperturbată
d	*n*	*m*	Ordin $J(1007)$	Exprimare sub formă de funcție de transfer
15	2	1	10^{-2}	Parametrii nu converg la valori staționare
15	2	2	10^{-9}	$z^{-15}\frac{0.6377z^{-1}+0.6102z^{-2}}{1-1.872z^{-1}+0.8751z^{-2}}$
15	3	2	10^{-2}	Parametrii nu converg la valori staționare
15	3	3	10^{-9}	$z^{-15}\frac{0.6378z^{-1}+0.9392z^{-2}+0.3148z^{-3}}{1-1.356z^{-1}-0.0907^{-2}+0.4515z^{-3}}=z^{-15}\frac{0.6378(z+0.5160)(z+0.9566)}{(z+0.5159)(z-0.9665)(z-0.9054)}$
				Cazul (*b*) - Ieșire perturbată
d	*n*	*m*	Ordin $J(1007)$	Exprimare sub formă de funcție de transfer
15	2	1	10^{-2}	Parametrii nu converg la valori staționare
15	2	2	10^{-5}	$z^{-15}\frac{0.637z^{-1}+0.6158z^{-2}}{1-1.8711z^{-1}+0.8742z^{-2}}$
15	3	2	10^{-2}	Parametrii nu converg la valori staționare
15	3	3	10^{-5}	$z^{-15}\frac{0.6378z^{-1}+0.9448z^{-2}+0.3144z^{-3}}{1-1.3552z^{-1}-0.0922^{-2}+0.4521z^{-3}}=z^{-15}\frac{0.6378(z+0.5048)(z+0.9766)}{(z+0.5167)(z-0.9043)(z-0.9676)}$

Rezultatele din tabelul 1 evidențiază rolul jucat de valoarea obținută în antrenare pentru funcția obiectiv $J(iT)$ în aprecierea convergenței parametrilor rețelei și, totodată, în evaluarea calității modelului identificat. Figura 6a și figura 6b furnizează reprezentările grafice ale evoluțiilor parametrilor rețelei pentru o durată de 150 de secunde corespunzătoare cazului (*b*) pentru n=2 m=2 și, respectiv, pentru n=2 și m=1. Aceste reprezentări grafice ilustrează totodată și capacitatea blocurilor Simulink nou create de a furniza on-line informații cantitative ce caracterizează progresul antrenării. Mai subliniem ca elemente importante în construcția modelului neuronal faptul că, în toate situațiile de convergență a parametrilor, coeficientul termenului în z^0 al numărătorului (în exprimarea sub formă de funcție de transfer) rezultă numeric nul (raportat la precizia de calcul).



a) b)

Figura 6. Evoluția parametrilor rețelei neuronale pentru exemplul din secțiunea 4.1 corespunzând unui orizont de 150 secunde:
a) cazul n=2 și m=2 (parametrii converg);
b) cazul n=2 și m=1 (parametrii nu converg)

Redundanţa în stabilirea numărului de regresori conduce la apariţia perechilor pol-zero cu valori numerice identice (raportate la precizia de calcul), fapt ilustrat în tabelul 1 pentru n=m=3 pentru ambele cazuri (a) şi (b).

4.2. Exemplu ilustrativ pentru construcţia unui model neliniar utilizând modelul MLP

Exemplul prezentat în această subsecţiune ilustrează utilizarea blocului ***MLP_N_ID*** în identificarea unui model neliniar de tipul (1) cu T=1 secundă. Procesul este descris, în Simulink, de următoarea ecuaţie cu diferenţe (recomandată în literatură ca exemplu relevant pentru identificare neurală [13], [14]):

$$y(k)=2.5*\frac{y(k-2)*y(k-1)}{1+y^2(k-1)+y^2(k-2)}+0.3\cos(0.5(y(k-1)+y(k-2)))+1.2u(k-1) \qquad (7)$$

Figura 7 prezintă diagrama Simulink, utilizată pentru identificare. Semnalul de intrare este de tip „zgomot alb" adecvat ales. Toate cazurile ce vor fi comentate (extrase dintr-un set amplu de teste) se referă la aceleaşi condiţii de antrenare pe eşantioane (on-line) cu N_E=200 eşantioane în $J(iT)$ din (4) şi algoritmul Levenberg-Marquardt. Criteriile de oprire a antrenării au fost fie atingerea unui număr maxim de 100 epoci/fereastră, fie o acurateţe de aproximare $\varepsilon=10^{-9}$ pentru funcţia obiectiv (4). Stratul de intrare conţine 10 neuroni sigmoidali. Rezultatele obţinute pentru d=0, $m\geq1$, $n\geq2$, în (1) sunt comparabile, din punct de vedere al acurateţii, pentru cazurile (a) şi (b) având valori de ordinul 10^{-5} pentru $J(250T)$ în cazul (a) şi, respectiv, 10^{-3} în cazul (b). Cazurile cu d, m, n care nu îndeplinesc condiţiile menţionate anterior pot fi uşor separate, ele furnizând valori pentru $J(250T)$ de ordinul 10^{-1} – 10^{-2}.

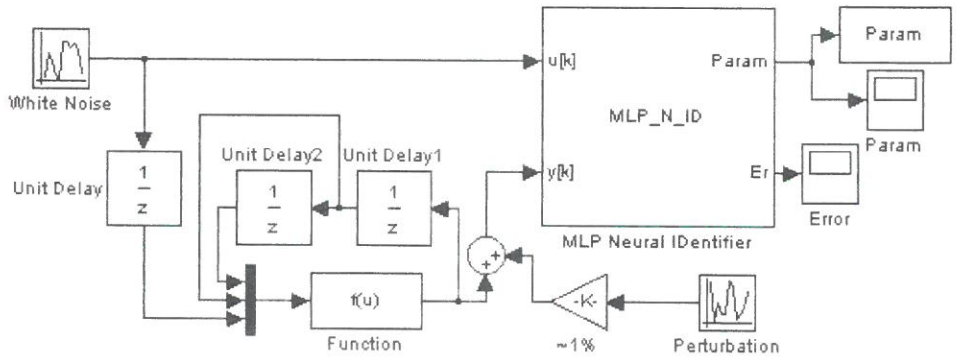


Figura 7. Diagrama pentru identificare neuronală folosind *MLP_N_ID*

Pentru a ilustra calitatea modelului neural, rezultat din identificare, figura 8 prezintă grafice comparative pentru răspunsul procesului şi al modelului pentru o intrare sinusoidală cu amplitudinea 0.7 (figura 8a) şi un semnal de intrare aleator, uniform distribuit în intervalul [-1,1] (figura 8b). Parametrii modelului neural sunt cei obţinuţi în situaţia (b) cu d=0, m=1, n=2 (adică numărul minim de regresori în (1) capabili să asigure valori favorabile pentru $J(250T)$). Semnalele de test utilizate nu au fost „văzute" în timpul identificării. Folosirea numărului exact de regresori prezenţi în ecuaţia (3) (d=1, m=0, n=2) nu a adus îmbunătăţiri vizibile în acurateţea de aproximare a modelului obţinut în comparaţie cu o serie de teste nedetaliate în această lucrare.

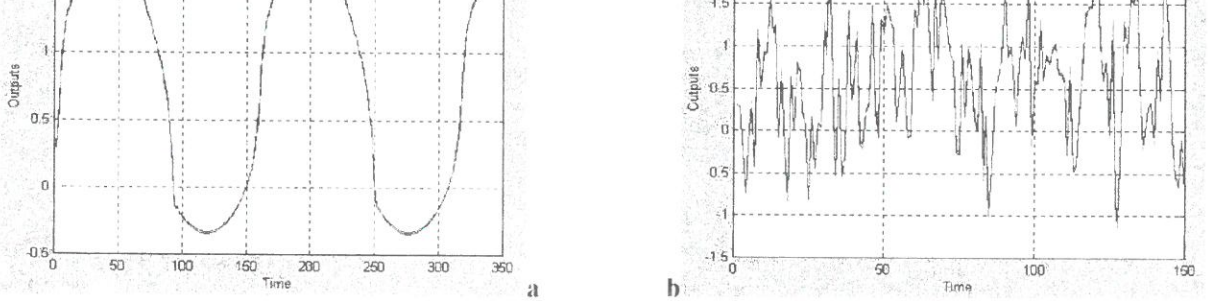


Figura 8. Grafice comparative cu răspunsurile procesului (linie continuă) şi modelului neural MLP (linie întreruptă):
a) intrare sinusoidală de amplitudine 0.7;
b) semnal aleator uniform distribuit în [-1,1]

În general vorbind, testele au arătat că modelele neurale nu sunt afectate semnificativ de folosirea regresorilor suplimentari în comparaţie cu folosirea numărului exact de regresori specificat anterior. Totuşi, redundanţa creşte complexitatea modelului neural fără nici un beneficiu real, de aceea e preferabil de căutat un model apropiat cu cel minimal.

4.3. Utilizarea identificatorului RBF

Exemplul ce urmează ilustrează utilizarea blocului ***RBF_N_ID*** pentru identificarea aceluiaşi proces neliniar (7) ca şi în subsecţiunea anterioară. Diagrama Simulink, utilizată pentru identificare, este identică cu cea din figura 7, excepţie făcând blocul ***RBF_N_ID*** care înlocuieşte blocul ***MLP_N_ID***. S-a utilizat o antrenare pe lot (batch) cu un număr N=200 de eşantioane, d=0, m=1, n=2 în (3), eroare medie pătratică dorită 0.01 şi dispersie 1.2 obţinându-se o arhitectură cu 87 neuroni de intrare. Graficele comparative ale răspunsurilor procesului şi modelului neural (rezultate obţinute din simulare cu semnalele de intrare, utilizate în subsecţiunea anterioară) au arătat aceeaşi calitate de aproximare prezentată în figura 8 (cazul MLP) şi, de aceea, prezenţa lor şi în această subsecţiune ar fi fost redundantă.

Unele efecte importante ale dimensiunii lotului (considerat la antrenare) asupra calităţii modelului identificat sunt prezentate în figura 9. Figura 9a prezintă dependenţa numărului de neuroni din stratul ascuns de numărul de eşantioane utilizate în antrenare, arătând că o dimensiune mai mare a lotului duce la un număr mai mare de neuroni şi, deci, la o sporire a timpului de calcul. Figura 9b şi 9c prezintă dependenţa erorii pătratice medii, rezultată din simulare (pentru semnal sinusoidal şi, respectiv, aleator uniform distribuit), de numărul de eşantioane din lot. Condiţiile de antrenare au fost identice pentru toate dimensiunile loturilor considerate în experimente (adică dispersie = 1.2, eroare medie pătratică dorită ε = 0.01). Ultimele două grafice arată că simpla creştere a numărului de eşantioane din lot, fără o corelare cu dispersia, nu garantează calitatea modelului neural identificat.

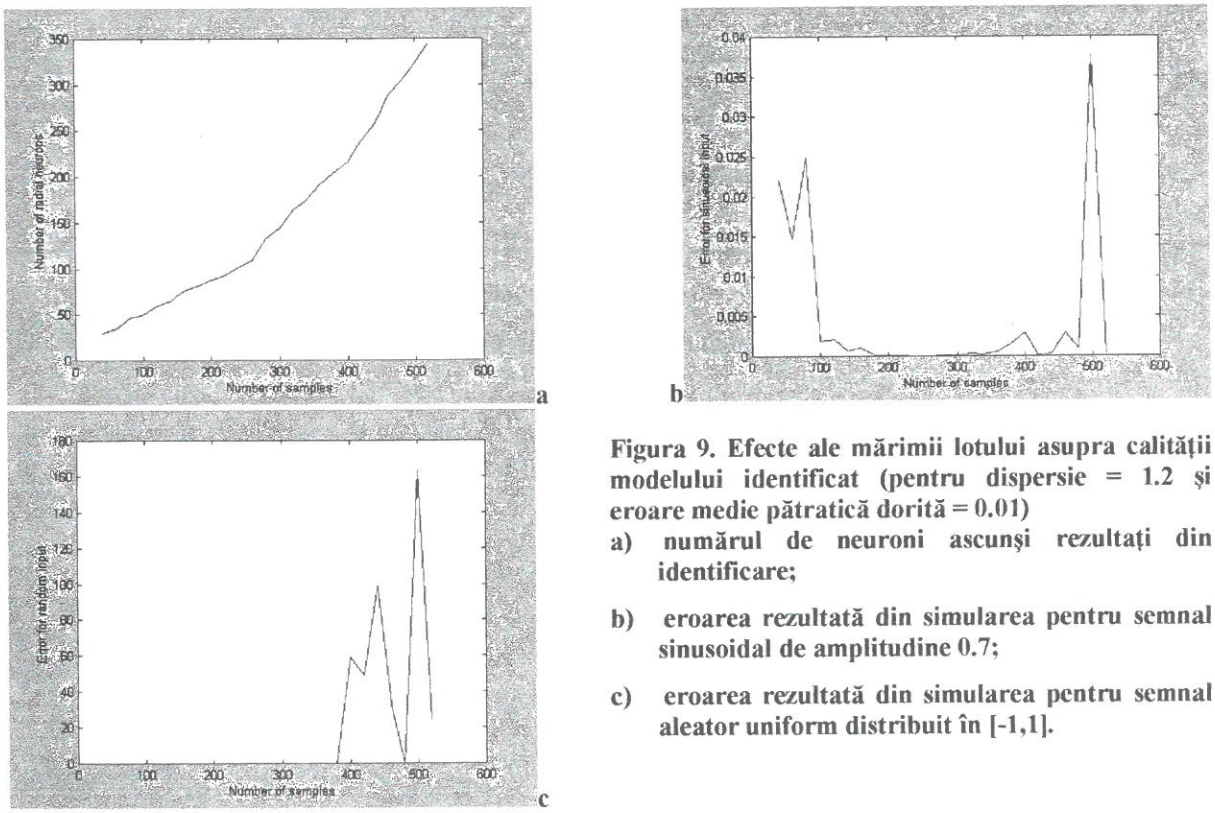


Figura 9. Efecte ale mărimii lotului asupra calităţii modelului identificat (pentru dispersie = 1.2 şi eroare medie pătratică dorită = 0.01)

a) numărul de neuroni ascunşi rezultaţi din identificare;

b) eroarea rezultată din simularea pentru semnal sinusoidal de amplitudine 0.7;

c) eroarea rezultată din simularea pentru semnal aleator uniform distribuit în [-1,1].

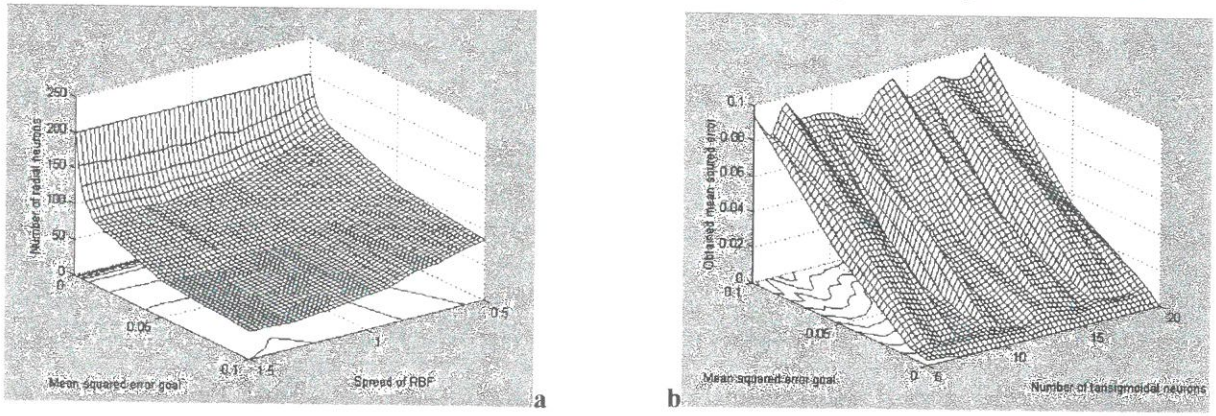
5. Comparaţie între modelele neurale de tip RBF şi cele MLP

Reţelele de tip MLP şi RBF sunt aproximatori universali, motiv pentru care este interesantă realizarea unui studiu comparativ cu privire la utilizarea lor în identificarea proceselor utilizând reţele neurale. De aceea, am construit modele de tip RBF şi MLP pentru acelaşi proces neliniar, adică cel considerat în secţiunea anterioară, ecuaţia (6). Toate rezultatele ce vor fi comentate au fost obţinute pentru antrenare pe lot (batch) cu d=0, m=1, n=2, în (1) (care diferă de valorile exacte ce apar în ecuaţia (7)). Pentru a asigura relevanţa comparaţiilor pentru reţelele MLP, s-a utilizat o valoare unică (1000) pentru a limita numărul de epoci de antrenare şi o valoare unică

(1) pentru inițializarea ponderilor și deplasamentelor neuronilor. Ne vom concentra, mai întâi, asupra rezultatelor antrenării și apoi asupra calității modelului identificat.

5.1. Comentarii asupra antrenării rețelelor

Rezultatele antrenării sunt analizate pentru condiții variate, utilizate în procesul de învățare. De aceea, pentru modelele RBF, numărul de neuroni din stratul ascuns este privit ca un rezultat al antrenării (care depinde de doi parametri cheie, adică eroarea medie pătratică dorită ε și dispersie). Similar, pentru modelele MLP, eroarea medie pătratică obținută a fost privită ca rezultat al antrenării (care depinde de doi parametri cheie, adică eroarea medie pătratică dorită ε și numărul de neuroni sigmoidali). Acest mod de a privi rezultatele antrenării permite o interpretare bazată pe grafice tridimensionale, date în figura 10 (a – model RBF, b – model MLP). Pentru ambele tipuri de modele, dacă se alege o anumită valoare pentru eroarea medie pătratică dorită (adică o secțiune în graficele tridimensionale figura 10a și 10b), rezultatele antrenării (numărul de neuroni în cazul RBF și eroarea medie pătratică obținută pentru cazul MLP) nu sunt mult influențate de al doilea parametru utilizat în antrenare (dispersia pentru cazul RBF și numărul de neuroni sigmoidali în cazul MLP). Trebuie precizat că utilizarea unei erori medii pătratice dorite nule (adică $\varepsilon = 0$ în (3)) are înțelesuri diferite pentru modelele de tip RBF și cele de tip MLP: (*i*) numărul de neuroni ascunși este egal cu dimensiunea lotului pentru cazul RBF; (*ii*) algoritmul de optimizare pentru modelele MLP va fi forțat să utilizeze numărul maxim de epoci alocate pentru antrenare.



a b

Figura 10. Interpretarea grafică a elementelor cheie ce caracterizează rețeaua la construcția modelului neliniar:

a) numărul de neuroni în funcție de dispersie și eroarea medie pătratică dorită pentru modelul RBF;

b) eroarea medie pătratică obținută în funcție de numărul de neuroni sigmoidali și eroarea medie pătratică dorită pentru modelul MLP

Pentru a compara timpii necesari antrenării modelelor de tip RBF și cele de tip MLP, figura 11 prezintă numărul de operații în virgulă mobilă (flops) funcție de dimensiunea lotului, pentru următoarele condiții de antrenare (considerate favorabile în concordanță cu rezultatele furnizate de figura 10): eroare medie pătratică dorită $\varepsilon = 0.01$ pentru ambele modele, dispersie = 1.2 pentru modelul RBF și număr de neuroni sigmoidali pentru modelul MLP = 19. Deși modelul MLP conține un număr mare de neuroni sigmoidali, arhitectura de tip RBF devine dezavantajoasă, din punct de vedere al timpului de calcul, față de MLP atunci când dimensiunea lotului depășește 300 eșantioane, deoarece creșterea numărului de operații în virgulă mobilă este liniară în cazul MLP (linie întreruptă) și exponențială pentru cazul RBF (linie continuă).

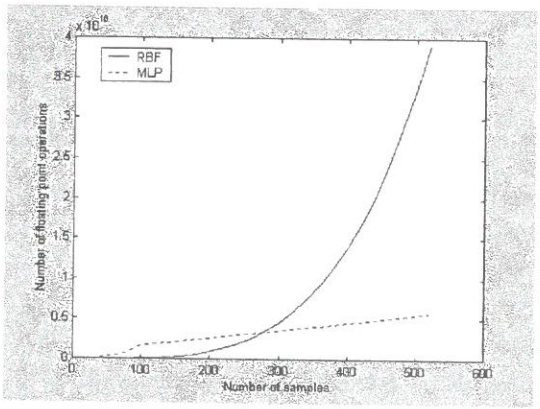
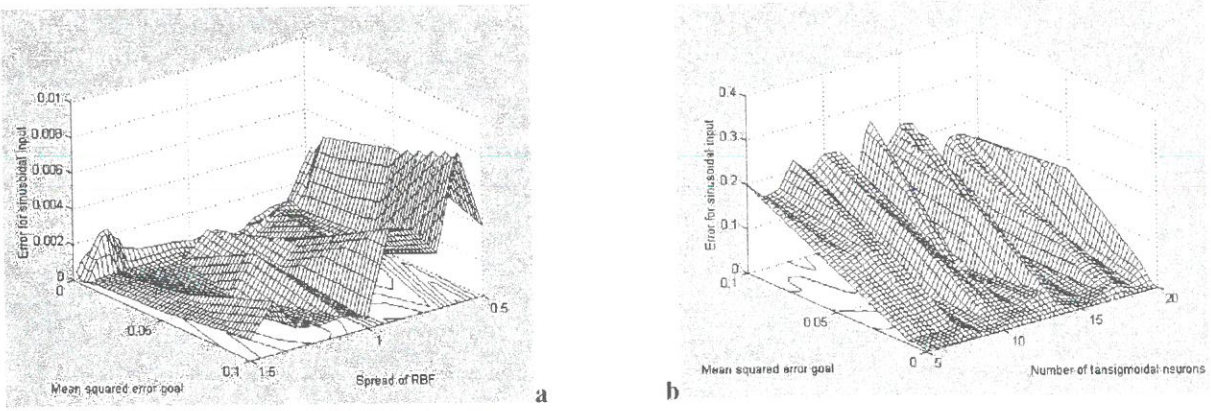


Figura 11. Comparare între timpii de calcul necesari antrenării modelelor RBF și MLP cu aceeași eroare medie pătratică dorită $\varepsilon = 0.01$:

- **cazul RBF (linie continuă);**
- **cazul MLP (linie întreruptă)**

5.2. Comentarii asupra calității modelului obținut

Calitatea modelelor identificate este evaluată din punct de vedere al rezultatelor simulării modelelor cu semnale ce nu au fost „văzute" în timpul antrenării. O interpretare grafică a acestor rezultate se poate da folosind aceleași grafice tridimensionale, ca în subsecțiunea anterioară, prezentate în figurile 12 și 13. Suprafețele trasate în aceste grafice reflectă dependența calității modelului obținut (exprimată pentru ambele modele, RBF și MLP, de eroarea medie pătratică obținută din simulare) de condițiile de antrenare (investigate în subsecțiunea 5.1). Graficele permit compararea calității modelelor RBF și MLP obținute prin simulare cu semnal sinusoidal de amplitudine 0.7 (figura 12) și, respectiv, semnal aleator uniform distribuit în intervalul [-1,1] (figura 13).



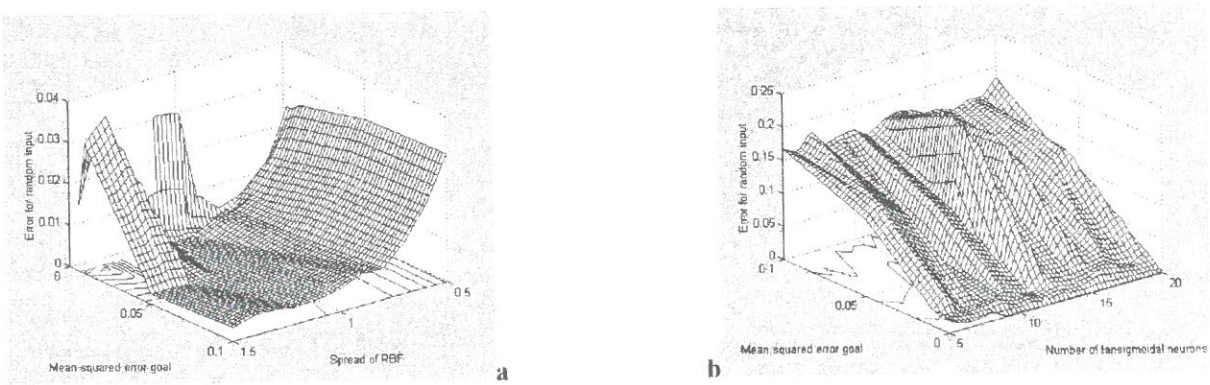
a b

Figura 12. Interpretarea grafică a elementelor cheie ce caracterizează calitatea modelului identificat pentru simularea cu semnal sinusoidal de amplitudine 0.7:

a) calitatea modelului RBF funcție de dispersie și eroarea medie pătratică dorită;

b) calitatea modelului MLP funcție de numărul de neuroni sigmoidali și eroarea medie pătratică dorită.

Examinarea vizuală directă a graficelor din figura 12 și figura 13 arată că, pentru alegeri adecvate ale condițiilor de antrenare, calitatea modelului RBF obținut este mai bună ca aceea a modelului MLP. Această remarcă generală este bazată pe informații numerice precise (utilizate în construcția graficelor), care dau magnitudinea erorilor rezultate din simulare pentru ambele tipuri de modele. Totuși, este important de precizat faptul că suprafețele obținute pentru modelele de tip MLP (figura 12b și figura 13b), spre deosebire de cele corespunzătoare modelelor RBF, prezintă o anumită regularitate geometrică ceea ce garantează o mai ușoară exploatare pentru utilizatorii mai puțin experimentați. Mai mult, ambele suprafețe trasate în figurile 12b și13b păstrează caracteristicile principale ale suprafeței trasate în figura 10b din subsecțiunea anterioară, acest lucru arătând că, pentru modelele de tip MLP, rezultatele unei antrenări adecvate sunt capabile să furnizeze informații credibile despre calitatea modelului identificat ce urmează a fi exploatat pentru o varietate de semnale de intrare. Din păcate, o asemenea regulă nu poate fi formulată în cazul modelelor RBF, figurile 10a, 12a și 13a prezentând diferențe majore în ceea ce privește forma suprafeței. Această ultimă remarcă subliniază necesitatea acordării unei atenții suplimentare validării modelului identificat atunci când se utilizează arhitecturi de tip RBF.



a b

Figura 13. Interpretarea grafică a elementelor cheie ce caracterizează calitatea modelului identificat pentru simularea cu semnal aleator uniform distribuit în intervalul [-1,1]:

a) calitatea modelului RBF funcție de dispersie și eroarea medie pătratică dorită;

b) calitatea modelului MLP funcție de numărul de neuroni sigmoidali și eroarea medie pătratică dorită.

6. Concluzii

Prin crearea bibliotecii de module vizuale prezentate în lucrare, proiectate şi implementate în spiritul tehnologiei Simulink, s-a urmărit extinderea posibilităţilor de exploatare a acestui mediu în domeniul identificării bazate pe reţele neuronale. Prin utilizarea modulelor software aparţinând acestei biblioteci, utilizatorul poate să se concentreze strict pe problematica identificării, fiind în totalitate degrevat de activitatea laborioasă a scrierii de cod în maniera programării clasice. Astfel, instrumentele realizate permit accesul la toate arhitecturile standard de reţele neuronale vizate în identificare (ADALINE, MLP şi RBF), oferind variante de antrenare de tip off- şi on-line, prin diverşi algoritmi, în conformitate cu specificul topologiei selectate.

Studiile de caz considerate ilustrează aspectele fundamentale ale antrenării, acoperind atât situaţia dinamicii liniare, cât şi a celei neliniare. Un accent deosebit este pus pe calitatea modelelor rezultate din identificare şi pe posibilitatea acestora de a reproduce, prin simulare, comportări noi, necunoscute din etapa de antrenare. Alocarea unei secţiuni separate pentru analiza comparativă a identificării neliniare bazată pe arhitecturi MLP şi, respectiv, RBF permite comentarea amplă a avantajelor şi dezavantajelor privind atât procesul de antrenare, cât şi eficienţa în sine a modelării.

Elaborarea unor module care să pună la dispoziţie strategii (legi) de conducere ce folosesc modelele neuronale, rezultate din identificare, nu a putut fi discutată în lucrarea de faţă datorită volumului mare de informaţii ce ar excede cadrul tradiţional. Abordarea unei atare problematici va face obiectul tratării distincte pe parcursul unui viitor articol.

Bibliografie

1. **BATTITI, R.**: First and Second Order Methods for Learning: Between Steepest Descent and Newton's Method. În: Neural Computation, nr. 4, 1992, pp. 141 – 166.
2. **CHARALAMBOUS, C.**: Conjugate Gradient Algorithm for Efficient Training of Artificial Neural Networks. În: IEEE Proc., nr. 139, 1992, pp. 301 – 310.
3. **CHEN, S., S.A. BILLINGS, C.F.N. COWAN, P.M. GRANT**: Practical Identification of NARMAX Models Using Radial Basis Functions. În: International Journal of Control, nr. 52, 1990, pp. 1327 – 1350.
4. **CHEN, S., S.A. BILLINGS**: Neural Networks for Nonlinear Dynamic System Modelling and Identification. În: International Journal of Control, nr. 56(2), 1992, pp. 319 – 346.
5. **CYBENKO, G.**: Approximation by Superpositions of a Sigmoidal Function. În: Mathematics of Control, Signals and Systems, nr. 2, 1989, pp. 303 – 314.
6. **DE KEYSER, R., O. PĂSTRĂVANU, D. ONU**: An Approach to Neural Network Based Nonlinear Identification and Control Using MATLAB. În: Prep. 3rd IFAC Symposium on Advances in Control Education, ACE '94, Tokyo, 1994, pp. 115 – 118.
7. **DUMITRACHE, I., N. CONSTANTIN, M. DRAGOICEA**: Reţele neuronale, MatrixRom, 1999.
8. **HAGAN, M.T., M. MENHAJ**: Training Feedforward Networks with the Marquardt Algorithm. În: IEEE Trans. on Neural Networks, nr. 5, 1994, pp. 989 - 993.
9. **HUNT, K.J., D. SBARBARO, R. ZBIKOWSKI, P.J. GAWTHORP**: Neural Networks for Control – A Survey. În: Automatica, nr. 28, 1992, pp. 1083 – 1112.
10. **HVONIK, K., M. SINCHCOMBE, H. WHITE**: Multilayer Feedforward Network are Universal Approximators. În: Neural Networks, nr. 2, 1989, pp. 359 – 366.
11. **PARK, J., I.W. SANDBERG**: Universal Approximation Using Radial-Basis-Function Networks. În: Neural Computation, vol. 3, 1991, pp. 246 - 257.
12. **KLOETZER , M., D. ARDELEAN, O. PĂSTRĂVANU**: Developing Simulink Tools for Teaching Neural-Net-Based Identification. În: 9th Mediterranean Conference on Control and Automation, 2001, p. 63.
13. **LIU, G.P., V. KADIRKAMANATHAN, S.A. BILLINGS**: Stable Sequential Identification of Continuous Nonlinear Dynamical Systems by Growing RBF Networks. În: International Journal of Control, nr. 65, 1996, pp. 53 – 60.

14. **LIU, G.P., V. KADIRKAMANATHAN, S.A. BILLINGS**: Predictive Control for Non-Linear Systems Using Neural Networks. În: International Journal of Control, nr. 71, 1998, pp. 1119 – 1132.

15. **LIU, G.P., V. KADIRKAMANATHAN, S.A. BILLINGS**: Variable Neural Networks for Adaptive Control of Nonlinear Systems. În: IEEE Transactions on Systems, MAN and Cybernetics, nr. 29, 1999, pp. 34 – 43.

16. **LJUNG, L., J. SJÖBERG, H. HJALMARSON**: On Neural Network Model Structres in System Identification. În: Bittanti, S., Picci, G. (Editori) Identification, Adaptation, Learning, NATO ASI Series, Springer, 1996.

17. **MOLLER, M.F.**: A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning. În: Neural Networks, nr. 6, 1992, pp. 525 – 533.

18. **QIN, S.Z., H.T. SU, T.J. MCAVOY**: Comparison of Four Neural Net Learning Methods for Dynamic System Identification. În: IEEE Trans. on Neural Networks, nr. 2, 1992, pp. 52 – 262.

19. **SIMA, V., A. VARGA**: Practica optimizării asistate de calculator, Editura Tehnică, 1986.

20. **The MathWorks Inc.**: Simulink 4.1 (Release 12.1), 2001.

21. **The MathWorks Inc.**: Neural Networks Toolbox 4.0.1 (Release 12.1), 2001

22. **WIDROW, B., M.E. HOFF**: Adaptive Switching Circuits. În: IRE WESCON Convention Record, New York IRE, 1960, pp. 96 – 104.