IMPLEMENTAREA ÎN MATLAB – SIMULINK A UNOR STRATEGII TIPICE DE IDENTIFICARE BAZATE PE MODELE NEURALE

1. Considerații generale, motivație și obiectiv

Identificarea sistemelor dinamice bazată pe modele neurale s-a conturat ca direcţie de cercetare la începutul anilor ’90, drept urmare a investigaţiilor matematice asupra proprietăţilor de aproximare ale reţelelor neuronale. Evoluţia noului domeniu a fost impulsionată de contribuţiile remarcabile ale unor nume de prestigiu în Automatică, ca, de exemplu, K. Narendra, K. Parthasarathy, P.J. Gawthorp, T.J. McAvoy, L. Ljung. Totuşi, firmele producătoare de software tehnico-ştiinţific au demarat dezvoltarea de facilităţi specifice reţelelor neuronale, astfel încât apariţia în 1992 a primei versiuni a ***Neural Network Toolbox*** încorporată în mediul MATLAB 4.2 a avut un impact major asupra interesului acordat de către automatişti acestei direcţii de cercetare.

2. Cunoștințe prealabile necesare

- Capitolul ”Aplicații ale rețelelor feedforward multistrat în identificarea şi controlul sistemelor dinamice” din cursul ”Aplicații ale rețelelor neurale în automatică”.
- Experiență de programare în MATLAB-Simulink.

3. Breviar teoretic

3.1. Identificarea sistemelor dinamice utilizând rețele neurale

Modelele generale de identificare parametrică a sistemelor dinamice (liniare sau neliniare) se pot implementa, atât hard cât şi soft, cu ajutorul reţelelor neurale artificiale. Parametrii modelului neural sunt reprezentaţi de ponderile sinaptice ale neuronilor şi deplasările acestora. Teoria generală privind identificarea sistemelor se poate aplica direct pentru obţinerea modelului neural ţinând cont de echivalenţele:

- *structura modelului = arhitectura reţelei neurale*
- *estimare = antrenare*
- *validare = generalizare*
- *set de date de estimare = set de date de antrenare*
- *set de date de validare = set de date de generalizare*

Structura modelelor neurale

Considerăm un sistem dinamic cu o intrare şi o ieşire (figura 6.1.1) pentru care se măsoară semnalele de intrare u şi de ieşire y la momentele discrete de timp $t_k = kt$ (unde T reprezintă perioada de eşantionare a semnalelor, $k \in N$), notând $u[k] = u(kT), y[k] = y(kT), k = \overline{1,N}$.

Figura 1. *Sistem dinamic SISO*

În problema identificării sistemelor dinamice, unde neliniaritatea care trebuie modelată depinde de timp, problema care se pune este cea a reprezentării explicite a timpului în structura reţelei neurale. Pentru ca o reţea neurală să fie dinamică, ea trebuie să posede o memorie. Ideea de la care s-a pornit la începutul anilor 1990 a fost cea de a se utiliza rețele neurale statice împreună cu un

sistem extern de întârziere a semnalelor de intrare în reţea, obţinând aşa numitele *reţele neurale cu dinamică externă*.

Un interes practic deosebit îl prezintă obţinerea modelelor neurale de tip ARX (NNARX) în timp discret, cu perioadă de eşantionare dată:

$$\hat{y}[k] = f(y[k-1], y[k-2], \dots, y[k-n], u[k-d], u[k-d-1], \dots, u[k-d-m]) \qquad (1)$$

unde $d, n \in N^*, m \in N$ şi

$$f: R^{n+m+1} \to R \qquad (2)$$

este o aplicaţie (liniară sau neliniară, netedă) ce descrie transferul intrare-ieşire al unei reţele neurale statice cu topologie adecvată. Această funcţie poate fi reprezentată cu ajutorul unei reţele neurale statice cu propagare înainte a semnalului cu funcţii de activare liniare (ADALINE) sau neliniare (MLP). Reţeaua are ca intrări valorile de la momentele anterioare de timp ale intrării şi ieşirii procesului. Valorile semnalelor de intrare şi de ieşire la momentele anterioare de timp formează vectorul *regresorilor* care este notat:

$$\varphi[k] = [\, y[k-1], y[k-2], \dots, y[k-n], u[k-d], u[k-d-1], \dots, u[k-d-m] \,]^T \qquad (3)$$

şi se obţine utili-zând un sistem de întârziere (*Tapped Delay Line*) plasat în afara reţelei neurale propriu-zise.

Configuraţia prezentată în formula xx este cunoscută şi sub denumirea de *modelul serie-paralel* (SPM) deoarece în raport cu intrarea reţeaua neurală este conectată paralel cu procesul, iar în raport cu ieşirea conectarea este în serie. Eroarea de predicţie , diferenţa dintre ieşirea reală a procesului şi ieşirea estimată cu ajutorul reţelei neurale , este utilizată pentru antrenarea reţelei aplicând algoritmi de antrenare uzuali, independenţi de timp. În această figură, z^{-1}reprezintă operatorul de întârziere cu un pas, $z^{-1}u[k] = u[k-1]$, iar ZOH notează extrapolatorul de ordin zero (*Zero Order Hold*) utilizat pentru discretizarea, cu perioada de eşantionare dorită, a unui proces continuu.

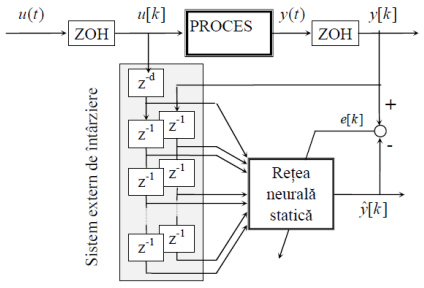


Figura 2. *Modelul neural de tip ARX(NNARX) – modelul serie-paralel*

După antrenarea reţelei neurale, modelul identificat poate fi folosit independent de proces, aducând la intrarea reţelei ieşirea acesteia, printr-o conexiune inversă externă. O astfel de configuraţie poartă numele de *modelul paralel* şi este utilizată pentru simularea proceselor dinamice. Configuraţia poate fi utilizată şi în faza de antrenare a reţelei neurale, modelul obţinut în acest fel fiind de fapt modelul neural de tip *output error* (NNOE) (figura 3)

$$\hat{y}[k] = f(\hat{y}[k-1], \hat{y}[k-2], \dots, \hat{y}[k-n], u[k-d], u[k-d-1], \dots u[k-d-m]) \qquad (4)$$

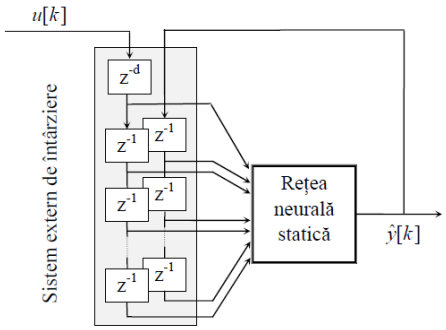


Figura 3. *Modelul neural OE (NNOE) – modelul paralel*

În mod asemănător se pot obţine şi modelele neurale de tip ARMAX (NNARMAX) sau Box-Jenkins (NNBJ). Dezavantajele modelelor recursive de tip NNOE, NNARMAX sau NNBJ sunt posibila instabilitate a modelului şi necesitatea unui efort suplimentar pentru a calcula gradientul ce intervine în procedura iterativă de antrenare a reţelei.

Pentru reprezentarea modelelor neliniare de tip NNARX cu o singură intrare şi o singură ieşire Narendra şi Parthasarathy (1990) au introdus patru modele. Modelele propuse de ei sunt descrise de ecuaţiile:

Modelul 1:	$\hat{y}[k+1] = \sum_{i=0}^{n-1} \alpha_i \cdot y[k-i] + g(u[k], u[k-1], \dots, u[k-m+1])$	(5)
Modelul 2:	$\hat{y}[k+1] = f(y[k], y[k-1], \dots, y[k-n+1]) + \sum_{i=0}^{m-1} \beta_i \cdot u[k-i]$	(6)
Modelul 3:	$\hat{y}[k+1] = f(y[k], y[k-1], \dots, y[k-n+1]) + g(u[k], u[k-1], \dots, u[k-m+1])$	(7)
Modelul 4:	$\hat{y}[k+1] = f(y[k], y[k-1], \dots, y[k-n+1], u[k], u[k-1], \dots, u[k-m+1])$	(8)

Se presupune că funcţiile *f* şi *g* sunt diferenţiabile în raport cu toate argumentele lor. Aceste funcţii pot fi aproximate cu ajutorul unor reţele neurale statice. Alegerea modelului adecvat depinde de informaţia despre procesul considerat disponibilă a priori. În modelele 1 (figura 4) şi 2 (figura 5) se presupune că neliniaritatea procesului implică numai intrarea, respectiv numai ieşirea.

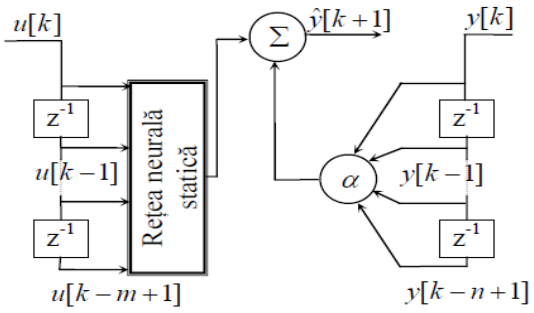


Figura 4. *Reprezentarea modelului 1*

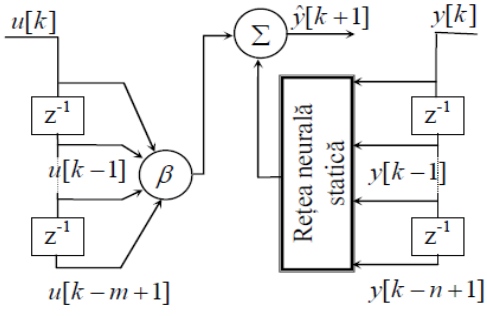


Figura 5. *Reprezentarea modelului 2.*

Modelul 3 (figura 6) rezultă ca suma a două funcţii neliniare distincte care depind numai de semnalul de intrare, respectiv ieşire. Modelul 4 (figura 7), cel mai general, reprezintă proiecţia neliniară atât a intrărilor cât şi a ieşirilor procesului de identificat.

Modelele de tip NNARX pot fi utilizate pentru probleme simple, dar utilizarea lor nu este avantajoasă pentru probleme ce implică dependenţa semnalului de ieşire al sistemului de un număr mare de valori trecute ale semnalului de intrare. Pentru modelarea unor asemenea sisteme se recomandă utilizarea modelelor recursive.

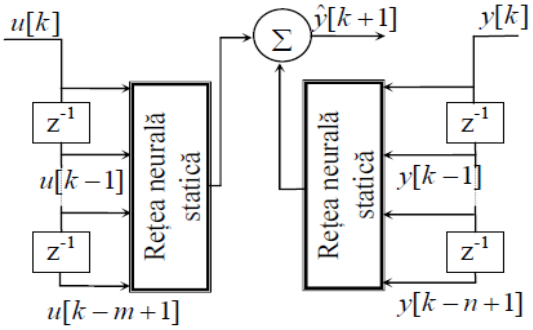


Figura 6. *Reprezentarea modelului 3*

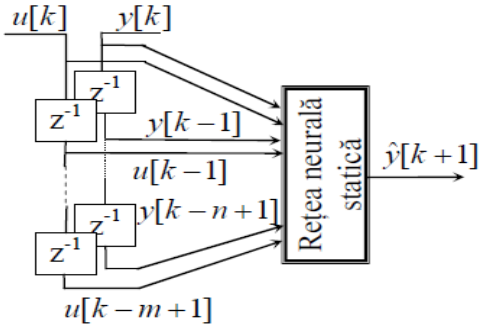


Figura 7. *Reprezentarea modelului 4*

O altă posibilitate de modelare a unui sistem dinamic neliniar revine la a construi în prima etapă un model liniar al sistemului. Reziduurile obţinute ca diferenţă între ieşirile actuale ale sistemului şi ieşirile estimate pe baza modelului liniar vor conţine evident neliniarităţile nemodelate. Pentru a modela aceste neliniarităţi se construieşte apoi un model neural având ca intrări intrările sistemului şi reziduurile.

Estimarea modelelor neurale

Aplicaţia *f* din relaţia (2) este parametrizată de ponderile şi deplasările reţelei. Indiferent de topologia reţelei, identificarea pe baza aşa-numitei scheme serie-paralel (figura 2) asigură stabilitatea algoritmului iterativ de minimizare a funcţiilor obiectiv definite cu ajutorul erorii

$$e[k] = y[k] - \hat{y}[k-1] \tag{9}$$

Formularea concretă a funcţiilor obiectiv este corelată cu modul de obţinere şi exploatare a datelor experimentale corespunzătoare procesului. În acest sens, literatura evidenţiază două categorii de metode de antrenare a reţelei neuronale din schema serie-paralel:

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

$$J = \frac{1}{N} \cdot \sum_{k=1}^{N} e^2[k] \leq \varepsilon \tag{10}$$

  unde *N* notează numărul total de elemente disponibile în lot şi ε este valoarea de prag dorită;
- pe eşantioane (*pattern*) sau *on-line*, care utilizează funcţii obiectiv de forma (până la o constantă multiplicativă):

$$J(i) = \frac{1}{N_E} \cdot \sum_{k=i}^{i+N_E-1} e^2[k] \leq \varepsilon, i \in N \tag{11}$$

  unde *NE* notează numărul de eşantioane conţinute în fereastra mobilă de selectare a elementelor, *i* este indicele iteraţiei curente şi ε este valoarea de prag dorită.

Minimizarea operează în spaţiul parametrilor reţelei (ponderi şi deplasări) şi poate fi condusă prin tehnicile clasice de optimizare fără restricţii, adaptate la specificul organizării matriceal-vectoriale a parametrilor aplicaţiei *f*. În limbajul propriu reţelelor neuronale, determinarea parametrilor funcţiei *f* ca rezultat al minimizării funcţiilor obiectiv (5.6) sau (5.7) (estimarea modelului parametric) este referită drept proces de antrenare sau învăţare.

Observaţie:

Determinarea modelului neural urmează paşii prezentaţi în capitolul 2: alegerea regresorilor modelului, a tipului neuronilor din structura reţelei neurale, alegerea numărului de neuroni din stratul intern şi apoi determinarea ponderilor reţelei (adică determinarea parametrilor modelului). Principiul care se aplică este de a porni de la simplu la complex (lama lui Occam): se încearcă întâi cu un număr redus de regresori şi de neuroni în stratul de intrare şi apoi se creşte progresiv numărul neuronilor până la un număr rezonabil. Dacă nu se determină un model suficient de bun se creşte numărul regresorilor şi se încearcă, din nou, valori crescute progresiv ale numărului de neuroni. Dacă nici în acest caz nu se determină un model corespunzător, se alege un alt tip de model pentru sistemul respectiv.

3.2. Controlul predictiv al sistemelor dinamice neliniare utilizând modele neurale

Principiul de bază al algoritmilor de control predictiv constă în prezicerea ieşirii procesului, pe baza unui model (obţinerea predictorilor), şi pe minimizarea unei anumite funcţii de cost. Diferenţele dintre diverşii algoritmi sunt date de modelul folosit de controller şi de forma funcţiei de cost. În cazul sistemelor liniare, majoritatea modelelor dinamice utilizate în controlul predictiv se obţin prin teste aplicate proceselor sau prin metode de identificare. Pentru controlul predictiv neliniar testarea şi identificarea proceselor devin mult mai dificile. Datorită proprietăţii reţelelor neurale de a fi aproximatori generali, utilizarea unui model neuronal pentru obţinerea predictorilor prezintă un foarte mare interes.

O reprezentare grafică a strategiei de control predictiv este prezentată în figura 8. La fiecare moment de eşantionare *k* se determină strategia optimă de control, fiind calculate valorile viitoare ale intrării $u[k+1], u[k+2], \dots, u[k+N_u]$ (N_u – orizontul de control), astfel încât ieşirea procesului *y* să urmărească cu eroare cât mai mică referinţa *r* pe intervalul de predicţie (N_1 – orizontul minim de predicţie, N_2 – orizontul de predicţie).

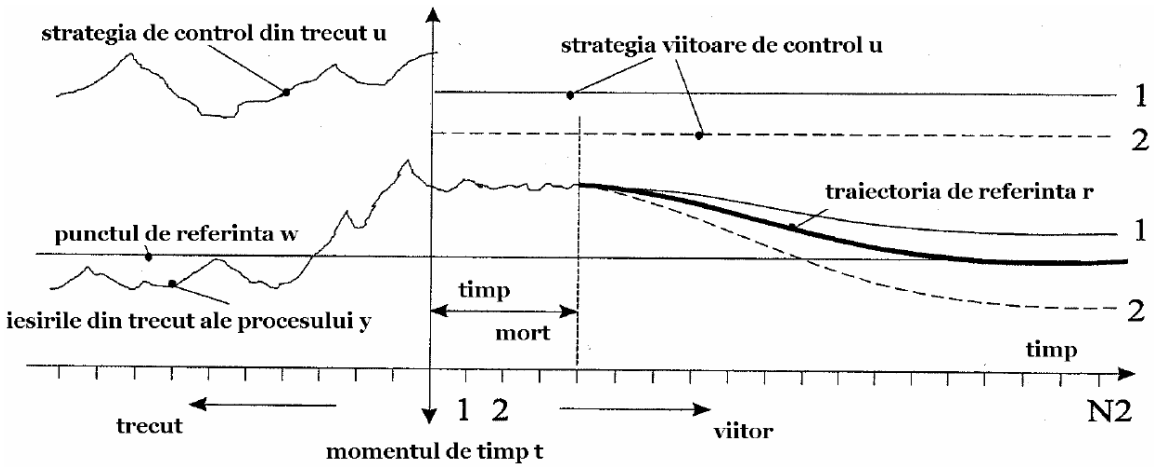


Figura 8. *Strategia controlului predictive bazat pe model*

Valorile viitoare ale ieşirii procesului sunt determinate folosind un model neural, iar pe baza erorilor între ieşirile prezise şi referinţă este calculată o funcţie de cost. Această funcţie de cost urmează a fi minimizată cu scopul obţinerii unei strategii de control optime ce vor fi aplicate procesului. Una din formele posibile ale funcţiei de cost, folosită în majoritatea algoritmilor de control predictiv, este dată de relaţia:

$$J = \sum_{i=N_1}^{N_2} (r[k+i] - y[k+i])^2 + \lambda \cdot \sum_{i=1}^{N_u} (u[k+i] - u[k+i-1])^2 \qquad (12)$$

De multe ori, în practică, ponderea λ asupra comenzii are valoarea 0, iar orizontul minim de predicţie (N_1) ia valoarea 1.

Funcţia de cost *J* urmează a fi minimizată la orice moment de timp *k* în funcţie de vectorul $= [u[k+1], u[k+2], \dots, u[k+N_u]]^T$, ce conţine valorile viitoare ale intrării.

Modelul neural folosit în calculul ieşirilor viitoare ale procesului este un model de tip ARX de forma prezentată în (figura 1). Pentru vectorii conţinând intrările, respectiv ieşirile viitoare ale procesului vom folosi următoarele notaţii:

$$u[k+i-d] = [u[k+i-d], u[k+i-d-1], \dots, u[k+i-d-m]]^T \qquad (13)$$
$$y[k+i-1] = [y[k+i-1], y[k+i-2], \dots, y[k+i-n]]^T$$

unde $i = N_1, \dots, N_2$, reprezintă ordinul predictorului, *k* este momentul curent de timp, iar *d*, *m* şi *n* reprezintă parametrii consideraţi pentru modelul NNARX antrenat: *d* - timpul mort, *m* şi *n* ordinele de întârziere ale intrării, respectiv ieşirii. Aplicând vectorii (13) modelului neural, se obţine predictorul de ordin *i* al ieşirii, $y[k+i]$, după cum rezultă şi din figura 9.

În calculul predictorilor ieşirii, daca orizontul de predicţie este mai mare decât orizontul de control sumat cu timpul mort considerat în modelul procesului ($N_2 > N_u + d$), după iteraţia $N_u + d$, intrarea va fi considerată constantă, având valoarea $u[k+N_u]$.

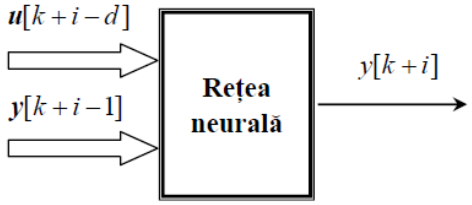


Figura 9. *Calculul predictorilor neurali*

Algoritmul iterativ de control predictiv poate fi uşor înţeles prin descrierea paşilor ce sunt parcurşi la orice moment de timp (iteraţie) *k*:

- în iteraţia anterioară (momentul de timp k-1) prin minimizarea funcţiei de cost, a fost obţinut vectorul comenzii:
$$\boldsymbol{u}_{old} = [u[k], u[k+1], \dots, u[k+N_u-1]^T \qquad (14)$$
- primul element al acestui vector, u[k], reprezintă intrarea procesului la momentul de timp curent;
- sunt calculaţi predictorii ieşirii y[k+1] de ordin $i = N_1, \dots, N_2$, folosind ca intrări ale modelului neural perechile de vectori $u[k+i-d$ şi $y[k+i-1]$
- se calculează funcţia de cost *J* la pasul curent, după formula (12);
- noua secvenţă optimă de control se obţine prin minimizarea funcţiei de cost *J* în raport cu vectorul comenzii
$$\boldsymbol{u}_{new} = [u[k+1], u[k+2], \dots, u[k+N_u]^T \qquad (15)$$

La momentul de timp următor (k+1), primul element din vectorul , $\boldsymbol{u}_{new}$, *u[k+1],* este aplicat la intrarea procesului şi algoritmul de control predictiv continuă cu iteraţia *k+1*. La startarea algoritmului, utilizatorul trebuie să aleagă valoarea iniţială *u[0]* a intrării procesului.

Intrările ce urmează a fi aplicate procesului real pot fi supuse unor restricţii ce decurg din natura fizică a acestuia. Unii algoritmi de control predictiv permit ca aceste restricţii să fie luate în consideraţie la minimizarea funcţiei de cost.

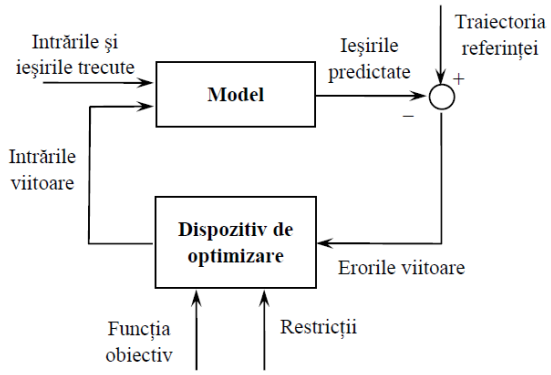


Figura 10. *Structura de bază a unui controller predictiv*

Având în vedere principiile MBPC prezentate, structura unui bloc de control ce implementează un astfel de algoritm poate fi prezentată ca în figura 10:
Viteza de calcul a unui astfel de controller trebuie să fie suficient de ridicată, pentru a permite efectuarea paşilor corespunzători unei iteraţii a algoritmului de control într-un timp mai mic decât perioada de eşantionare aleasă pentru interfaţarea cu procesul real.

3.3. Blocuri simulink pentru aplicații ale rețelelor neurale în identificarea și conducerea sistemelor

Figura 11 prezintă o bibliotecă de blocuri Simulink pentru identificarea, simularea şi controlul predictiv al sistemelor pe baza modelelor neurale

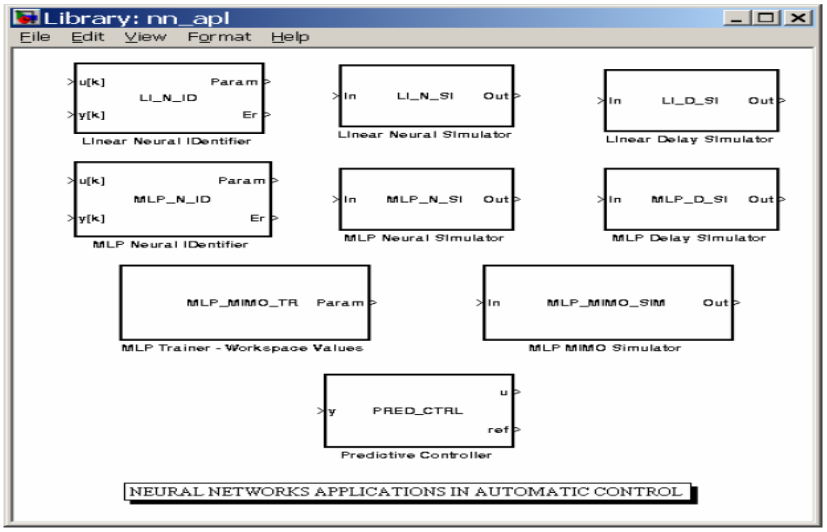


Figura 11. *Bibliotecă de blocuri Simulink pentru identificarea, simularea şi controlul predictiv al sistemelor pe baza modelelor neurale.*

Dezvoltarea acestei biblioteci de blocuri Simulink pentru identificarea, simularea şi controlul predictiv al sistemelor pe baza modelelor neurale a fost motivată de dorinţa de a uşura descrierea procesului supus identificării (aşa cum se va vedea în continuare), lăsând utilizatorului, în acelaşi timp, flexibilitate în organizarea simulării. Mai mult chiar, posibilitatea executării simulărilor în timp real, prin utilizarea Real-Time Workshops, extinde aria de aplicabilitate a procedurilor de identificare la procese reale, interfaţate adecvat prin cartele de achiziţie.

Biblioteca Simulink din figura 11 constă din blocuri ce permit identificarea şi simularea modelelor neurale cu o singură intrare şi o singură ieşire (SISO) de forma (figura 1) şi anume:
(*i*) pentru funcţia *f* liniară:

- blocul *LI_N_ID* (*LInear Neural IDentifier*) folosit pentru antrenarea unei reţele ADALINE,
- blocul *LI_N_SI* (*LInear Neural SImulator*) folosit simularea transferului intrare-ieşire a unui model neural cu topologie ADALINE, identificat anterior,
- blocul *LI_D_SI* (*LInear Delay SImulator*) folosit pentru simularea transferului intrare-ieşire a unui model neural de tip ADALINE, cu timp mort, identificat anterior;

(*ii*) pentru funcţia *f* neliniară:

- blocul *MLP_N_ID* (*MLP Neural IDentifier*) serveşte la antrenarea reţelelor de tip MLP,
- blocul *MLP_N_SI* (*MLP Neural SImulator*) serveşte pentru simularea transferului intrare-ieşire a unui model neural cu topologie MLP, identificat anterior,
- blocul *MLP_D_SI* (*MLP Delay SImulator*) utilizat pentru simularea transferului intrare-ieşire a unui model neural cu topologie MLP, cu timp mort, identificat anterior.

Pentru identificarea şi simularea modelelor neurale neliniare cu mai multe intrări şi ieşiri (MIMO), această bibliotecă Simulink mai include:

- blocul *MLP_MIMO_TR* (*MLP MIMO Trainer*) serveşte la antrenarea reţelelor de tip MLP utilizând matrice din spaţiul de lucru Matlab,
- blocul *MLP_MIMO_SI* (*MLP MIMO SImulator*) serveşte pentru simularea transferului intrare-ieşire a unui model neural cu topologie MLP, identificat anterior, cu ajutorul blocului *MLP_MIMO_TR*.

În plus, blocul *PRED_CTRL* permite implementarea unui controller predictiv pentru un sistem SISO utilizând un model neural, identificat anterior cu folosind blocul *MLP_N_ID*.

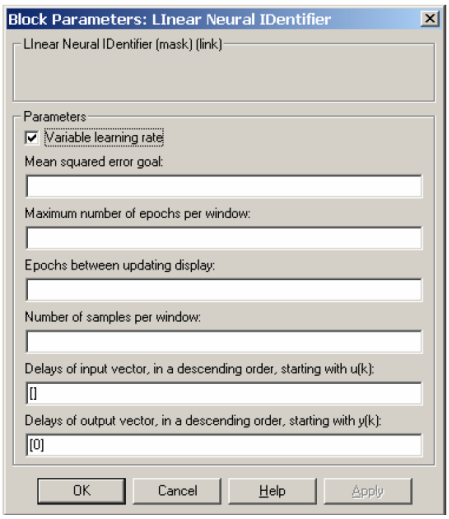


Figura 12. *Caseta de dialog a blocului* ***LIN_N_ID***

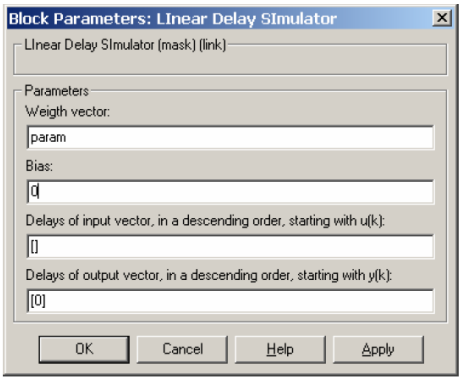


Figura 13. *Caseta de dialog a blocului* ***LIN_D_SI***

Casetele de dialog ale blocurilor *LI_N_ID* şi *MLP_N_ID* pentru identificarea modelelor SISO sunt prezentate în figurile 12 şi, respectiv, 14. Aceste blocuri permit atât antrenarea pe lot (*off-line*), în concordanţă cu (10), cât şi antrenare pe eşantioane (*on-line*), în concordanţă cu (11), cu un număr de eşantioane N_E specificat de utilizator în caseta de dialog. Utilizatorul poate seta valoarea de prag pentru eroarea pătratică medie utilizată în antrenare şi modul în care se formează vectorii regresorilor.

Casetele de dialog ale blocurilor ***LIN_D_SI*** şi ***MLP_N_SI***, utilizate în simulare, sunt prezentate în figurile 13 şi, respectiv, 15. Parametrii reţelelor pot fi setaţi manual sau pot fi încărcaţi automat din spaţiul de lucru al Matlab-ului.

În găsirea regresorilor optimi pentru vectorii de intrare şi ieşire este recomandată o strategie *bottom-up*, în sensul că arhitectura reţelei va fi iniţializată cu un număr redus de regresori. Pentru blocul de tip MLP se va porni cu un număr redus de neuroni pe stratul de intrare. Aceste valori vor incrementate succesiv până la obţinerea rezultatului dorit.

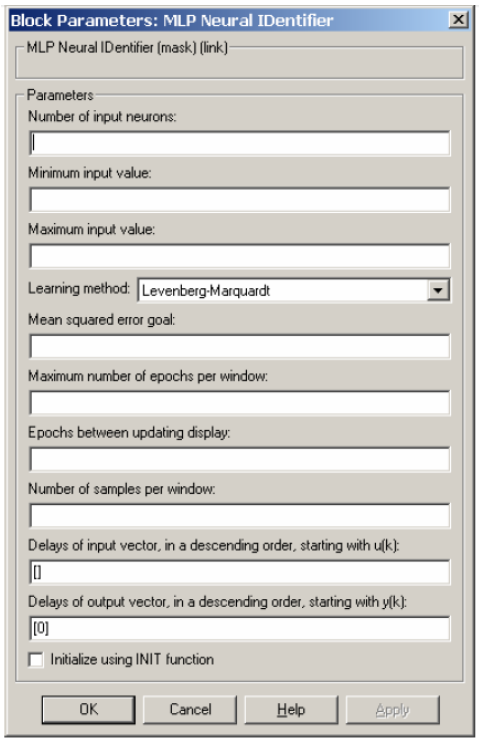


Figura 14. *Caseta de dialog a blocului* ***MLP_N_ID***

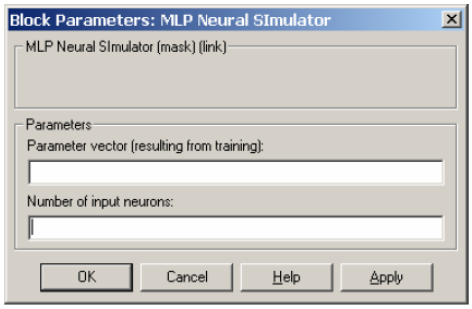


Figura 15. *Caseta de dialog a blocului* ***MLP_D_SI***

Blocul ***PRED_CTRL*** poate fi folosit pentru controlul proceselor neliniare ce au drept model neuronal o reţea de tip MLP. Caseta de dialog a acestui bloc este prezentată în figura 16, parametrii ce pot fi aleşi de utilizator privind atât reţeaua neurală (presupusă deja antrenată), cât şi algoritmul de control predictiv. Vectorul ce conţine parametrii modelului neuronal are aceeaşi organizare ca şi vectorul obţinut în urma identificării folosind blocurile ***MLP_N_ID***. Valorile întârzierilor intrării şi ieşirii, precum şi valoarea timpului mort, trebuie să fie aceleaşi cu cele considerate la crearea şi antrenarea modelului NNARX.

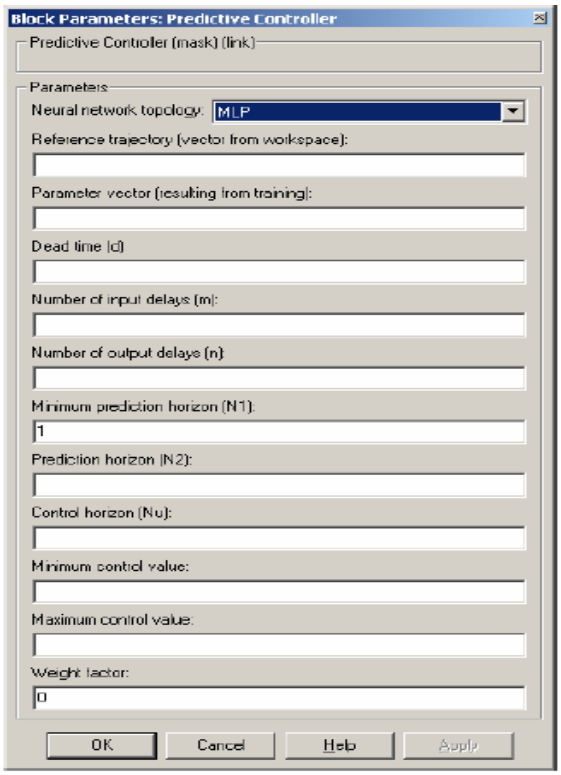


Figura 16. *Caseta de dialog a blocului* ***PRED_CTRL***

Parametrilor corespunzători algoritmului de control predictiv ce pot fi setaţi de către utilizator sunt în concordanţă cu notaţiile utilizate în paragraful 3.2. Traiectoria referinţei este presupusă a fi cunoscută apriori, fiind specificată sub forma unui vector.

Ieşirea ‘u’ a blocului de control furnizează valoarea intrării de control la fiecare pas de simulare, iar ieşirea ‘ref’ poate fi utilă pentru a aprecia modul în care ieşirea procesului urmăreşte referinţa dorită pe durata simulării.

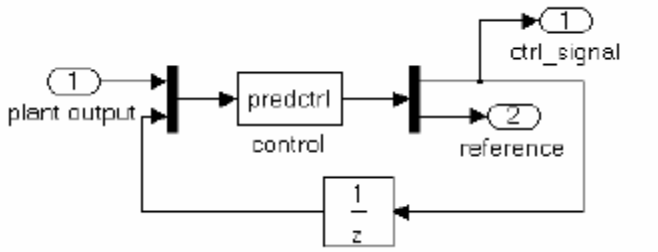


Figura 17. *Structura internă a blocului* ***PRED_CTRL***

În structura internă a blocului ***PRED_CTRL*** (figura 17) intră *funcţia-S* denumită ***predctrl*** care este responsabilă atât de organizarea datelor interne pe baza parametrilor specifici modelului neuronal şi algoritmului de control, cât şi de minimizarea funcţiei de cost *J* (12). La fiecare pas de simulare (iteraţie a algoritmului de control predictiv) această funcţie apelează o rutină ***cost.m***, care realizează calculul predictorilor neuronali şi al funcţiei de cost. Pentru crearea modelului neuronal s-au folosit funcţiile specifice din *Neural Network Toolbox*, iar pentru minimizarea funcţiei de cost s-a făcut apel la funcţia *fmincon* din *Optimal Toolbox*, care permite impunerea unor limite (restricţii) pentru variabila în raport cu care se efectuează minimizarea. La fiecare iteraţie a algoritmului este

memorat vectorul obţinut prin minimizarea lui *J*, iar acest vector va constitui punctul de start al minimizării la iteraţia următoare.

Conectarea blocului într-o schemă Simulink în vederea simulării controlului unui oarecare proces dinamic neliniar (pentru care se dispune de un model neuronal) este ilustrată în figura 18. În timpul simulării, poate fi urmărită evoluţia intrării de control şi urmărirea referinţei de către ieşirea procesului.

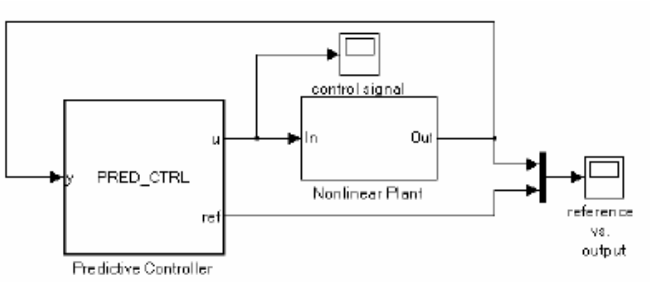


Figura 18. *Modul de conectare a blocului* ***PERD_CTRL*** *într-o schema Simulink*

3.4. Studii de caz

Deşi utilizarea modelelor neuronale este atractivă îndeosebi în cazul neliniar, luăm în discuție, pentru început, construcția unui model liniar, deoarece acesta permite comparații relevante (de natură parametrică) cu reprezentările standard de tip funcție de transfer, discretă în timp. Al doilea exemplu ilustrează construcția unui model neliniar pentru care discuția privind validitatea se limitează doar la analiza comparativă a răspunsurilor obținute de la proces și respectiv model, în condițiile aceloraşi semnale de test aplicate la intrare. În nici unul dintre cazuri nu abordăm problematica validării modelelor cu ajutorul testelor statistice.

Pentru ambele exemple, funcționarea procesului supus identificării este simulată în mediul Simulink, antrenarea rețelelor nefăcând însă uz de cunoașterea ordinului modelelor de simulare (ipoteză absolut firească în desfășurarea aplicațiilor reale). Totodată, pentru fiecare din exemple au fost considerate, separat, două situații ce urmează a fi detaliate mai jos:

(a) ieșirea procesului nu este afectată de perturbații;

(b) ieșirea procesului este afectată aditiv de perturbații 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 rezoluție scăzută).

3.4.1. Construcția unui model liniar

În figura 6.4.1 se prezintă schema Simulink (aferentă situației (b) menþionate la începutul secțiunii curente) utilizată pentru construirea unui model liniar de forma (figura 1) cu perioada de eșantionare secundă, cu ajutorul blocului LI_N_ID. Procesul ce face obiectul identificării este descris în Simulink prin funcția de transfer continuă în timp:

$$G(s) = \frac{400 \cdot e^{-15 \cdot s}}{300 \cdot s^2 + 40 \cdot s + 1} \tag{16}$$

procesul fiind excitat cu o sursă de zgomot alb de bandă limitată, adecvat aleasă. Singurele informații presupuse cunoscute apriori sunt comportarea liniară a procesului și valoarea timpul mort (identificabil, de exemplu, de pe un răspuns la semnal treaptă), adică în (figura 19) pentru secundă.

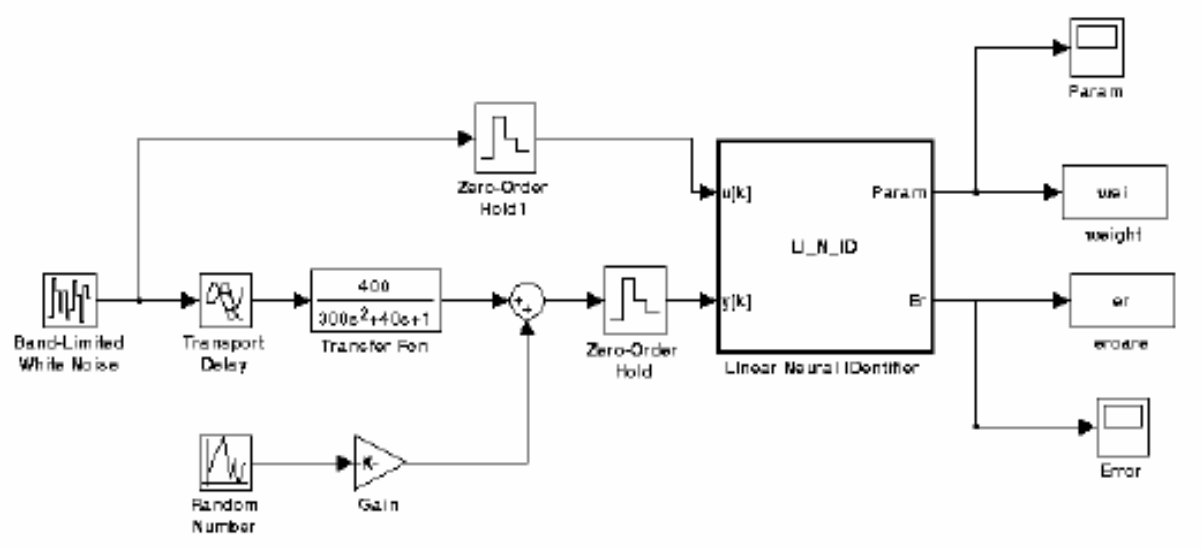


Figura 19. *Schema Simulink de identificare a modelului liniar corespunzător exemplului din secţiunea A.*

Selectăm drept complet edificatoare doar o parte din rezultate, prezentate în tabelul 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 furnizaţi de blocul LI_N_ID). Această exprimare facilitează analiza calităţii modelului provenit din identificare, prin comparaţie directă cu discretizată cu T = 1 secundă a lui G(s), obţinută prin metoda Euler predictor

$\tilde{G}(z^{-1}) = z^{-15}\dfrac{0.6378 \cdot z^{-1} + 0.6101 \cdot z^{-2}}{1 - 1.872 \cdot z^{-1} + 0.8752 \cdot z^{-2}}$	(17)

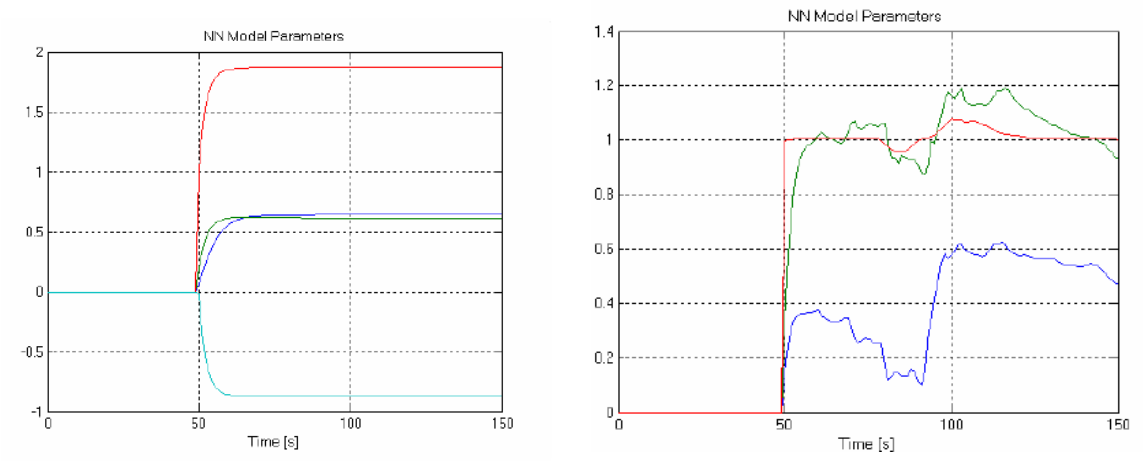
Toate cazurile comentate se referă la aceleaşi condiţii de antrenare on-line cu *NE*=50 eşantioane în funcţia obiectiv *J* din (13), maxim 100 epoci/fereastră şi rată de învăţare variabilă. Rezultatele din tabelul 6.4.1 evidenţiază rolul jucat de valoarea obţinută în antrenare pentru funcţia obiectiv *J* în aprecierea convergenţei parametrilor reţelei şi, totodată, în evaluarea calităţii modelului identificat.

Figurile 20.a şi 20.b furnizează reprezentările grafice ale evoluţiilor parametrilor reţelei pentru o durată de 150 de secunde corespunzătoare cazului (*b*) pentru *n*=2, *m*=2 şi, respectiv, pentru *n*=2 şi *m*=1. Aceste reprezentări grafice ilustrează totodată şi capacitatea blocurilor Simulink utilizate de a furniza on-line informaţii cantitative ce caracterizează progresul antrenării. Mai subliniem ca elemente importante în construcţia modelului neuronal faptul că în toate situaţiile de convergenţă a parametrilor coeficientul termenului în z^0 al numărătorului (în exprimarea sub formă de funcţie de transfer) rezultă numeric nul (raportat la precizia de calcul).

Tabelul 1. *Prezentare sintetică a rezultatelor antrenării pentru opt teste de identificare selectate cu relevanţe pentru secţiunea 3.4.1.*

				Cazul (a) – Ieșire neperturbată
d	**n**	**m**	**J(150)**	**Exprimare sub formă de funcție de transfer**
15	2	1	10^{-2}	Parametrii nu converg la valori staționare
15	2	2	10^{-9}	$z^{-15} \cdot \frac{0.6377 \cdot z^{-1} + 0.6102 \cdot z^{-2}}{1 - 1.872 \cdot z^{-1} + 0.8751 \cdot z^{-2}}$
15	3	2	10^{-2}	Parametrii nu converg la valori staționare
15	3	3	10^{-9}	$z^{-15} \cdot \frac{0.6378 \cdot z^{-1} + 0.9392 \cdot z^{-2} + 0.3148 \cdot z^{-3}}{1 - 1.356 \cdot z^{-1} + 0.0907 \cdot z^{-2} + 0.4515 \cdot z^{-3}} = z^{-15} \cdot \frac{0.6378 \cdot (\boldsymbol{z + 0.5160}) \cdot (z + 0.9566)}{(\boldsymbol{z + 0.5159}) \cdot (z - 0.9665) \cdot (z - 0.9054)}$
				Cazul (b) – Ieșire perturbată
d	**n**	**m**	**J(150)**	**Exprimare sub formă de funcție de transfer**
15	2	1	10^{-2}	Parametrii nu converg la valori staționare
15	2	2	10^{-9}	$z^{-15} \cdot \frac{0.637 \cdot z^{-1} + 0.6158 \cdot z^{-2}}{1 - 1.8711 \cdot z^{-1} + 0.8742 \cdot z^{-2}}$
15	3	2	10^{-2}	Parametrii nu converg la valori staționare
15	3	3	10^{-9}	$z^{-15} \cdot \frac{0.6378 \cdot z^{-1} + 0.9448 \cdot z^{-2} + 0.3144 \cdot z^{-3}}{1 - 1.3552 \cdot z^{-1} + 0.0922 \cdot z^{-2} + 0.4521 \cdot z^{-3}} = z^{-15} \cdot \frac{0.6378 \cdot (\boldsymbol{z + 0.5048}) \cdot (z + 0.9766)}{(\boldsymbol{z + 0.5167}) \cdot (z - 0.9043) \cdot (z - 0.9676)}$

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



a b

Figura 20. *Evoluţia parametrilor reţelei neuronale pentru exemplul din secţiunea xxx corespunzând unui interval de 150 secunde:*
a) *Cazul n=2 şi m=2 (parametrii converg),*
b) *Cazul n=2 şi m=1 (parametrii nu converg).*

3.4.2. Construcția unui model neliniar

Exemplul prezentat în continuare ilustrează utilizarea blocului ***MLP_N_ID*** în identificarea unui model neliniar de tipul (17) cu perioada de eşantionare T = 1 secundă. Procesul ce urmează a fi identificat este simulat în Simulink pe baza următoarei ecuaţii cu diferenţe (recomandată în literatură ca exemplu relevant pentru identificare neurală (Liu, *et al.*, 1998)):

$$y(k) = 2.5 \cdot \frac{y(k-2) \cdot y(k-1)}{1 + y^2(k-1) + y^2(k-2)} + \\ + 0.3 \cdot \cos(0.5 \cdot [y(k-1) + y(k-2)]) + 1.2 \cdot u(k-1) \quad (18)$$

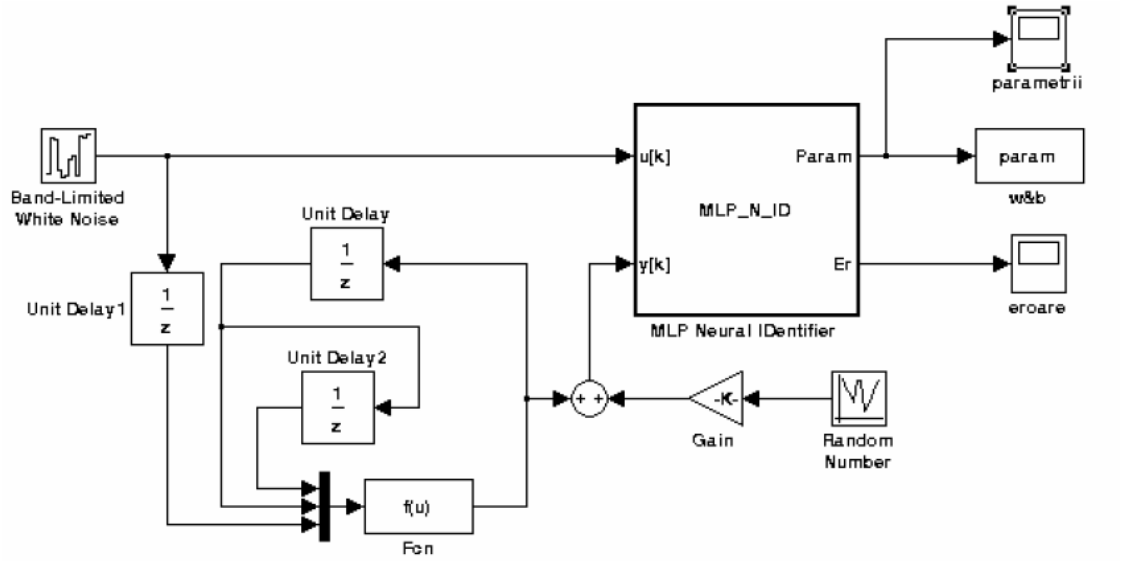
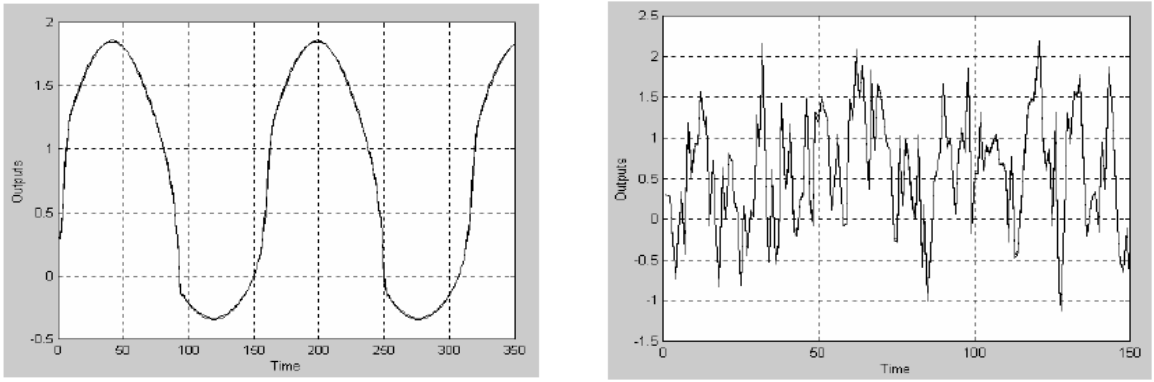


Figura 21. *Schema Simulink de identificare a modelului neliniar corespunzător exemplului din secțiunea B.*

Figura 21 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 on-line cu N_E = 200 eşantioane în funcţia obiectiv J din (10), utilizând algoritmul Levenberg-Marquardt. Criteriile de oprire a antrenării au fost fie atingerea unui număr maxim de 100 epoci/fereastră sau o acurateţe de aproximare $\varepsilon = 10^{-9}$ pentru funcţia obiectiv (10). Stratul de intrare conţine 10 neuroni sigmoidali. Rezultatele obţinute pentru d=0, m=1, n=2, în (1) sunt comparabile, din punct de vedere al acurateţei, pentru cazurile (*a*) şi (*b*) având valori de ordinul 10^{-5} pentru $J(250)$ în cazul (*a*) şi respectiv 10^{-3} în cazul (*b*). Cazurile cu *d*, *m*, *n* ce nu îndeplinesc condiţiile menţionate anterior pot fi uşor separate ele furnizând valori pentru $J(250)$ de ordinul $10^{-1} - 10^{-2}$.

Pentru a ilustra calitatea modelului neural rezultat din identificare, figura 6.4.4 prezintă grafice comparative pentru răspunsul procesului şi a modelului pentru o intrare sinusoidală cu amplitudinea 0.7 (figura 22.a) și un semnal de intrare aleator, uniform distribuit în intervalul [-1, 1] (figura 22.b). 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 $J(250)$). Semnalele de test utilizate nu au fost „văzute" în timpul identificării. Folosirea

numărului exact de regresori prezenţi în ecuaţia (21) (*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.



a b

Figura 22. *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 efectuate 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 de cel minimal.

3.4.3. Controlul predictive bazat pe model neural

Modelul neliniar identificat în secţiunea precedentă, corespunzător sistemului descris de ecuaţia cu diferenţe (3.4.2) este utilizat în continuare pentru a proiecta un controller predictiv pentru acest sistem utilizând blocul ***PRED_CTRL***. Schema Simulink utilizată este prezentată în figura 23.

Parametrii utilizaţi de algoritmul de control predictiv au următoarele valori:

- orizontul minim de predicţie $N_1 = 1$;
- orizontul de predicţie $N_2 = 3$;
- orizontul de control $N_u = 2$;
- factorul de ponderare $\lambda = 0$ şi
- valoarea iniţială a intrării $u_{init} = 0$.

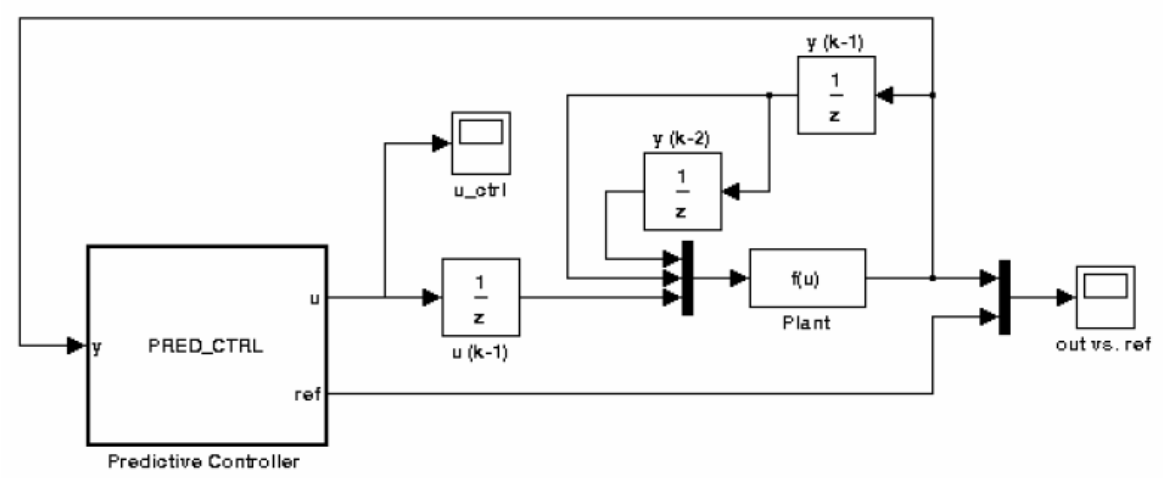


Figura 23. *Modelul Simulink utilizat pentru controlul predictiv al sistemului neliniar.*

Figura 24 ilustrează urmărirea unei referinţe sinusoidale, cu amplitudine 1, fază 0, offset 0,2 şi frecvenţă 0,005 Hz, iar figura 25 urmărirea unei referinţe de formă unghiulară, cu valori cuprinse în intervalul [-1,1].

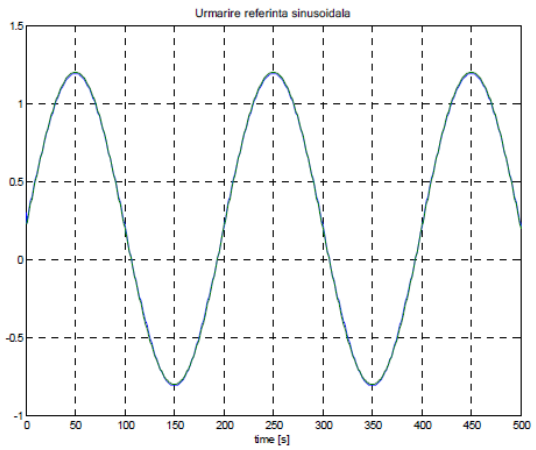


Figura 24. *Grafice comparative ale unei referinţe sinusoidale şi a ieşirii sistemului neliniar controlat de un regulator predictiv.*

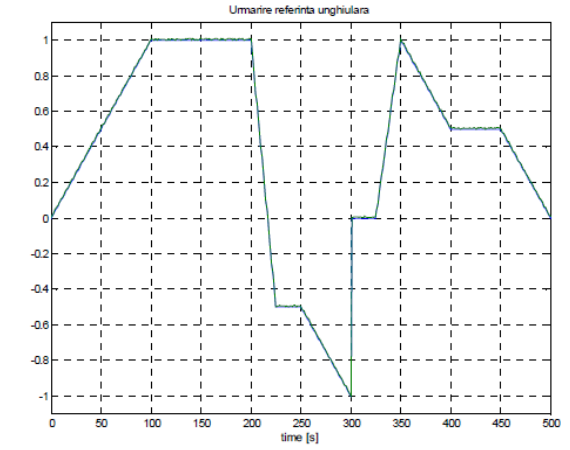


Figura 25. *Grafice comparative ale unei referinţe unghiulare şi a ieşirii sistemului neliniar controlat de un regulator predictiv.*

4. Problematica propusă pentru studiu

Problema 1.

Pe baza exemplului prezentat în secţiunea xxxx să se construiască modelul Simulink pentru identificarea sistemului liniar descris de funcţia de transfer

$$G(s) = \frac{125 \cdot e^{-3s}}{100 \cdot s^2 + 40 \cdot s + 3}$$

considerând perioada de eşantionare a semnalelor de 1 secundă.

Să se ruleze experimentele de identificare off-line şi on-line în următoarele condiţii:

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).

Corespunzător ecuaţiei (6.1.1), se vor lua în discuţie următoarele cazuri:
(*i*) $d = 4$, $n = 2$, $m = 1$;
(*ii*) $d = 4$, $n = 2$, $m = 2$;
(*iii*) $d = 4$, $n = 3$, $m = 2$;
(*iv*) $d = 4$, $n = 3$, $m = 3$;

Pentru fiecare din cazurile (*i*)-(*iv*) se va fixa valoarea de prag a erorii pătratice medii utilizată în antrenarea reţelei (*mean squared error goal*) la valorile 10^{-2} şi apoi 0; în fiecare caz în parte se va comenta legătura dintre parametrii reţelei neurale şi coeficienţii funcţiei de transfer discrete corespunzătoare sistemului, obţinută prin apelarea funcţiei Matlab c2d.

După rularea unui experiment de identificare, se va rula, pentru validarea modelului, un experimentul de simulare, considerând ca semnal de intrare în sistem şi în modelul neural obţinut atât un semnal de tip zgomot alb, cât şi un semnal determinist, de tip treaptă unitară sau sinusoidal.

Problema 2.

Să se reprezinte următoarele sisteme discrete sub forma unei reţele neuronale dinamice constituită dintr-o ADALINE şi întârzierile adecvate:

a) $G_1(z) = 10 \cdot \dfrac{(z+1)\cdot(z-1)}{(z-\frac{1}{2})\cdot(z-\frac{1}{3})}$

b) $G_2(z) = \dfrac{z^{-5} - 0.3 \cdot z^{-6}}{1 - 0.5 \cdot z^{-1} + 0.06 \cdot z^{-2})}$

c) $G_3(z) = \dfrac{b_2 \cdot z^2 + b_1 \cdot z + b_0}{b_3 \cdot z^3 + b_2 \cdot z^2 + b_1 \cdot z + b_0}$, $a_i, b_j \neq 0, i = \overline{0,3}, j = \overline{0,2}$

d) $G_4(z) = z^{-3} \cdot G_3(z)$

Problema 3.

Se consideră reţeaua neuronală cu dinamică externă din figura 7.18.1, unde reţeaua ADALINE are parametrii:

$$w = [2 \quad 0 \quad -18 \quad \eta \quad -1] \qquad b = 0$$

Iar q^{-1} notează operatorul de întârziere în domeniul timp.

a. Să se determine funcţia de transfer a reţelei.

b. Să se discute stabilitatea IMEM în funcţie de valoarea parametrului $\eta \in \mathbf{R}$.

c. Pentru valorile lui η pentru care sistemul este stabil IMEM să se arate că există reţele dinamice cu o configuraţie mai simplă care realizează acelaşi transfer intrare-ieşire şi să se reprezinte grafic aceste configuraţii.

Problema 4.

Se consideră sistemul neliniar în timp discret descris de ecuaţia cu diferenţe (pentru o perioadă de eşantionare T=1 secundă):

$$y(k) = \frac{y[k-1] + y[k-2]}{1 + 2y[k-1] \cdot y[k-2]} - 1.5 \cdot u[k-1] + u[k-2]$$

1. Folosind blocurile Simulink prezentate în secţiunea 6.3, să se obţină un model neuronal de tip MLP pentru procesul dat, efectuând experimente de identificare off-line şi on-line în următoarele condiţii:
(*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).

Corespunzător modelului NNARX, se va lua în discuţie atât cazul când întârzierile reţelei neuronale sunt egale cu întârzierile reale ale procesului, cât şi cazul introducerii unor regresori suplimentari în modelul neuronal.

2. Calitatea modelelor neuronale obţinute va fi apreciată din punct de vedere al analizei comparative a răspunsurilor obţinute de la proces şi respectiv model, pentru următoarele cazuri:
(*a*) semnal de intrare aleator cu distribuţie uniformă în intervalul [-1, 1];
(*b*) semnal de intrare aleator binar de nivele 1 şi -1.

3. Modelele obţinute vor fi folosite pentru a realiza controlul predictiv al procesului considerat în următoarele cazuri:
(*a*) referinţă sinusoidală, cu amplitudine 1, fază iniţială 0 şi pulsaţie 2250π rad/sec.;
(*b*) referinţă în formă de dinţi de ferăstrău de amplitudine 1, offset 0.5 şi perioadă 100 secunde.

Pentru algoritmul de control predictiv, se vor considera următorii parametrii:

- orizont de control N_u=2;
- orizont minim de predicţie N_1=1;
- orizont de predicţie N_2=3 şi
- ponderea asupra comenzii λ=0,2.

Problema 5.

Se consideră o instalaţie având funcţia de transfer

$$G(s) = \frac{5 \cdot e^{-20 \cdot s}}{10 \cdot s + 1}$$

pentru care se determină un model discret folosind o rețea neurală dinamică constituită dintr-o rețea ADALINE și întârzierile necesare.

Să se reprezinte grafic structura rețelei neuronale dinamice și să se precizeze parametrii rețelei ADALINE considerând că discretizarea lui G(s) se realizează exact, adică:

$$G_d(s) = (1 - z^{-1}) \cdot Z\{\frac{G(s)}{s}\}$$

pentru:

a) perioada de eșantionare de 1 sec;
b) perioada de eșantionare de 2 sec.

Problema 6.

Se consideră un proces având funcția de transfer

$$G(s) = \frac{30 \cdot s + 1}{35 \cdot s^2 + 12 \cdot s + 1}$$

Pentru care se determină un model discret folosind o rețea neuronală dinamică alcătuită dintr-o rețea ADALINE și întârzierile adecvate. Să se reprezinte grafic structura rețelei neuronale dinamice și parametrii rețelei ADALINE presupunând că discretizarea lui G(s) se realizează pentru o perioadă de eșantionare T = 1 sec, folosind:

a. metoda dreptunghiului în avans, care folosește $s \cong \frac{z-1}{T}$
b. metoda dreptunghiului în întârziere, care folosește $s \cong \frac{z-1}{T \cdot z}$
c. metoda trapezului, care folosește $s \cong \frac{2}{T} \cdot \frac{z-1}{z+1}$