UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ

Proiect identificarea sistemelor-Regresii liniare

Alinei-Poiana Tudor, Ciurezu Gheorghe-Dragos, Kocis Bianca Laura

8/8

2021-2022

Cuprins

1 Descrierea problemei

Pornind de la un set de date de identificare de tip intrare-ieșire (mai exact pornind de la un model de tip cutie neagră) obținut de la un sistem dinamic neliniar, trebuie să dezvoltăm un model parametric folosind o structură ARX neliniară, care să poată aproxima cât mai exact modelul matematic din spatele datelor exeperimentale oferite. După ce obținem modelul ce aproximează setul de date de identificare, îl vom valida pe un al doilea set de date de la același sistem pentru a asigura corectitudinea modelului dezvoltat.

În practică modelul "ARX" are o utilitate foarte mare, deoarece este capabil să aproximeze ieșirea atât prin predicție cât și prin simulare. Totodată obține rezultate suficient de bune la aproximare, dacă se aleg corespunzător numărul de zerouri, de poli și întarzierea sistemului.

Predicția reprezintă etapa în care știm cum arată ieșirea și intrarea și vom folosi datele respective până la un moment k pentru a prezice cum va arăta ieșirea la urmatorul pas, k+1. În schimb, simularea reprezintă etapa în care cunoscând doar intrarea simulăm ieșirea construindu-o de la 0 și folosindu-ne de intrare si de valorile ieșirii simulate anterior (se vor considera toate valorile până în momentul 0, inclusiv, ca fiind valori nule).

2 Structura aproximatorului

Vom începe prin exemplificarea structurii unui model ARX liniar, în care vom ține cont și de întârzierea nk a intrării:

$$y(k) = -a_1y(k-1) - a_2y(k-2) - ... - a_{na}y(k-na)b_1u(k-bk-1) + b_2u(k-nk-2) + ... + \\ + b_{nb}u(k-nk-nb) + e(k)$$

Aproximatorul polinomial are forma:

$$\hat{y}(k) = p(y(k-1), ...y(k-na), u(k-nk), u(k-nk-1), ..., u(k-nk-nb+1); \theta)$$

$$\hat{y}(k) = p\left(d(k)\right)$$

$$d(k) = [y(k-1), ...y(k-na), u(k-nk), u(k-nk-1), ..., u(k-nk-nb+1)]^T$$

3 Structura matricei de regresori

Se pleacă de la ideea de ARX simplu. Considerăm vectorul de intârzieri:
$d(k) = [y(k-1), ..., y(k-na), u(k-nk), u(k-nk), u(k-nk-1), ..., u(k-nk-nb+1)]^T$ având o formă asemanătoare cu vectorul PHI de la ARX:

$$PHI = [-y(k-1), ..., -y(k-na), u(k-1-nk), ..., u(k-na-nk)]^T.$$

Vectorul de ieșiri și intrări întârziate va avea urmatoarea formă:

$$\text{d} = \begin{bmatrix} y(0) & y(-1) & \dots & y(-na) & u(-nk) & \dots & u(-nk-nb+1) \\ y(1) & y(0) & \dots & y(1-na) & u(1-nk) & \dots & u(-nk-nb+2) \\ \vdots & \vdots & \ddots & & & & \vdots \\ y(N) & y(N-1) & \dots & y(N-na) & u(N-nk) & \dots & u(-nk-nb+N+1) \end{bmatrix}$$

Pentru simplitate în explicatiile de mai jos, se vor nota elementele din vectorul de întârzieri cu X.

$$d = \begin{bmatrix} X_{11} & X_{12} & \dots & X_{1(na+nb)} \\ X_{21} & X_{22} & \dots & X_{2(na+nb)} \\ \vdots & \vdots & \vdots & \\ \vdots & \vdots & \vdots & \\ X_{N1} & X_{N2} & \dots & X_{N(na+nb)} \end{bmatrix}$$

Se va căuta un polinom de forma:

$$P = \sum x_0^{p_0} \cdot x_1^{p_1} \cdot ... \cdot x_n^{p_n}$$

De exemplu, prima linie din matricea de regresori neliniari va conține elemente de forma:

$X_{11}^{P_1} \cdot X_{12}^{P_2} \cdot \dots \cdot X_{1(na+nb)}^{P_{na+nb}}$, Unde $P_1 + P_2 + \cdots + P_{na+nb} \leq m$, iar $P_1, P_2 \dots P_{na+nb} \in \{0, 1, ..., m\}$

De exemplu, matricea de regresori neliniari va fi de forma:

$$\text{Regresori\_NARX} = \begin{bmatrix} X_{11}^0 \cdot X_{12}^0 \cdot ... \cdot X_{1(na+nb)}^0 & X_{11}^1 \cdot X_{12}^0 \cdot ... \cdot X_{1(na+nb)}^0 & \dots & X_{11}^0 \cdot X_{12}^0 \cdot ... \cdot X_{1(na+nb)}^m \\ X_{21}^0 \cdot X_{22}^0 \cdot ... \cdot X_{2(na+nb)}^0 & X_{21}^1 \cdot X_{22}^0 \cdot ... \cdot X_{2(na+nb)}^0 & \dots & X_{21}^0 \cdot X_{22}^0 \cdot ... \cdot X_{2(na+nb)}^m \\ \vdots & \vdots & \vdots & \\ \vdots & \vdots & \vdots & \\ X_{N1}^0 \cdot X_{12}^0 \cdot ... \cdot X_{N(na+nb)}^0 & X_{N1}^1 \cdot X_{N2}^0 \cdot ... \cdot X_{N(na+nb)}^0 & \dots & X_{N1}^0 \cdot X_{N2}^0 \cdot ... \cdot X_{N(na+nb)}^m \end{bmatrix}$$

Se va încerca rezolvarea sistemului de tipul $Y = \phi \cdot \theta$, unde Y este o matrice coloană reprezentată de ieșirile sistemului. Rezolvând ecuația de mai sus folosindu-ne de datele de identificare se vor obține valorile parametrilor θ. Apoi se va construi simularea si predicția sistemului.

4 Metode de implementare

4.1 Metoda 1

Pentru această primă implementare am pornit de la ideea de a genera vectorul de intrări și ieșiri intârziate de forma: $a(k) = [y(k-1), y(k-2), \dots, y(k-na), u(k-nk), u(k-nk-1), \dots, u(k-nk-nb+1)]$.

Vom folosi notația $a(k) = [x_1(k), x_2(k), \dots, x_n(k)]$, unde $n = na + nb$.

Acum problema care ramane este să determinăm toate combinațiile de tipul $x_1^{p_1} \cdot x_2^{p_2} \cdot \dots \cdot x_n^{p_n}$ cu proprietatea ca $p_1 + p_2 + \dots + p_n \leq m$, m care reprezintă gradul maxim pe care dorim să îl aibă polinomul nostru.

Soluția la care am ajuns pentru generarea tuturor acestor combinații a fost să ne folosim de operatorii punctuali din Matlab astfel ca fiecare termen din vectorul a(k) să fie ridicat la o putere corespunzatoare dintr-o matrice $M = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ p_{x1} & p_{x2} & \cdots & p_{xn} \end{bmatrix}$,unde $p_{q1}+p_{q2}+...+p_{qn} \leq m, q = 1, x \Leftrightarrow$

$\Leftrightarrow a(k)\hat{}M = \begin{bmatrix} x_1^{p_{11}} & x_2^{p_{12}} & \dots & x_n^{p_{1n}} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{p_{x1}} & x_2^{p_{x2}} & \dots & x_n^{p_{xn}} \end{bmatrix}.Regresorii x_1^{p_1} \cdot x_2^{p_2} \cdot \ldots \cdot x_n^{p_n}$ cu proprietatea ca $p_1 + p_2 + \ldots + p_n \leq m$ se vor obtine înmultind elementele de pe fiecare linie, obținând astfel un vector coloană pe care îl vom transpune și adăuga la matricea de regresori Sigma cu ajutorul căreia compunen sistemul $Y = Sigma \cdot Theta$.

Singura problemă care mai ramâne este generarea acestei matrice de puteri M.

Inițial am pornit de la ideea de a realiza produsul cartezian a n=na+nb mulțimi de forma 0,1,2,...,m și să îl reținem într-o matrice de dimensiune $m^{n \cdot n}$ (m^n reprezintă cardinalul unui produs cartezian a n mulțimi de m elemente) urmând ca mai apoi sa eliminăm din această matrice toate liniile care nu respectă condiția ca suma elementelor să fie ≤m. Această implementare, deși bună ca idee de bază, creează mari probleme în privința memoriei necesare pentru a stoca matricea ce păstrează produsul cartezian (programul nu ar fi rulat pentru valori ale ordinului sistemului na=nb ≥ 5 si grad mai mare decât 10, fiindcă matricea depășea o dimensiune de stocare de ordinul miilor de GB). De asemenea, această implementare ar fi durat mult fiindcă înainte de a filtra rezultatele obținute ar fi trebuit să generam mult mai multe combinații de puteri ce abia mai apoi ar fi fost eliminate, deci nici din punct de vedere al timpului de execuție nu era o metodă fiabilă pentru valori prea mari.

Optimizând ideea de mai sus am ajuns la soluția de a genera produsul cartezian a acelor n mulțimi eliminând treptat din matricea ce reține produsul cartezian liniile ce conțineau elemente a căror sumă depășeau deja valoarea maximă m (mai precis, săream peste iterațiile ce generau de la bun început linii ce nu respectau proprietatea de mai sus).

Un alt rezultat important ce a facilitat această implementare a fost determinarea numarului de elemente ale unui polinom de n variabile si grad m, rezultat a carui demonstrație este prezentată mai jos:

4.1.1 Determinarea numărului de elemente al polinomului de n variabile și grad m

$P = \sum x_0^{p_0} \cdot x_1^{p_1} \cdot ... \cdot x_n^{p_n}; \quad 0 \leq p_0 + p_1 + ... + p_n \leq m$

Câți termeni are polinomul $P(x_0, ..., x_n)$ de n+1 variabile și grad m?

Pentru început vom determina câte combinații de tipul $x_0^{p_0} \cdot x_1^{p_1} \cdot ... \cdot x_n^{p_n}$ cu $p_0 + p_1 + ... + p_n = m$ sunt.

Fie următoarea schemă: $\underbrace{\text{XXX...X}}_{\text{m+n}}$, unde m este gradul.

- Alegem oricare n X-uri și le schimbăm în 0: $\underbrace{\text{X00XXX...X0X}}_{m \cdot X \text{ și } n \cdot 0}$
- Citim câte steluțe avem între fiecare cerc, de exemplu:
  XXX00X0XX0XX0 $\rightarrow$ $x_0^3 \cdot x_1^0 \cdot x_2^1 \cdot x_3^2 \cdot x_4^2 \cdot x_5^0$
  $5 \cdot 0 \Rightarrow$ 6 variabile cu grad maxim 8

Revenind la problemă:

Avem n+m X-uri și trebuie să alegem pe poziții diferite n 0-uri. Numărul de X-uri dintre două 0-uri va reprezenta puterea unui anumit $x_i, i \leq n$. Observăm că numărul de X-uri rămase va fi n+m-n=m. Deci avem C_{m+n}^n monoame de n+1 elemente distincte de grad m.

Astfel un polinom de n+1 variabile și grad m va avea:

termeni de n+1
variabile

$$\overset{\nearrow}{C_n^n} + \overset{\nwarrow}{C_{n+1}^n} + C_{n+2}^n + C_{n+3}^n + ... + C_{n+m}^n \text{ termeni în componența sa.} \Leftrightarrow$$

$\downarrow$ *grad* 0 $\quad$ $\downarrow$ *grad* 1

$$\Leftrightarrow \frac{n!}{n! \cdot 0!} + \frac{n!}{(n+1)! \cdot 1!} + \frac{n!}{(n+2)! \cdot 2!} + ... + \frac{n!}{(n+k)! \cdot k!} + ... + \frac{n!}{(n+m)! \cdot m!} =$$

$$= \sum_{k=0}^{m} C_{n+k}^n = C_{n+m+1}^{n+1} \text{ (Identitatea Hockey-Stick)}$$

Pentru n termeni vom avea deci C_{n+m}^n termeni în polinomul $p(x_0, ..., x_{n-1})$ de grad m.

Demonstrația Identității Hockey-Stick

$$\boxed{C_n^k = C_{n-1}^{k-1} + C_{n-1}^k}$$

$$\sum_{k=0}^{m} C_{n+k}^n = C_n^n + C_{n+1}^n + C_{n+2}^n + ... + C_{n+m}^n$$

$$= C_{n+1}^{n+1} + C_{n+1}^n + C_{n+2}^n + ... + C_{n+m}^n$$

$$= C_{n+2}^{n+1} + C_{n+2}^n + C_{n+3}^n + ... + C_{n+m}^n$$

$$= C_{n+3}^{n+1} + C_{n+3}^{n+1} + ... + C_{n+m}^n$$

$$= ... = C_{n+m}^{n+1} + C_{n+m}^n$$

$$= C_{n+m+1}^{n+1}$$

Acest rezultat este important din mai multe motive:

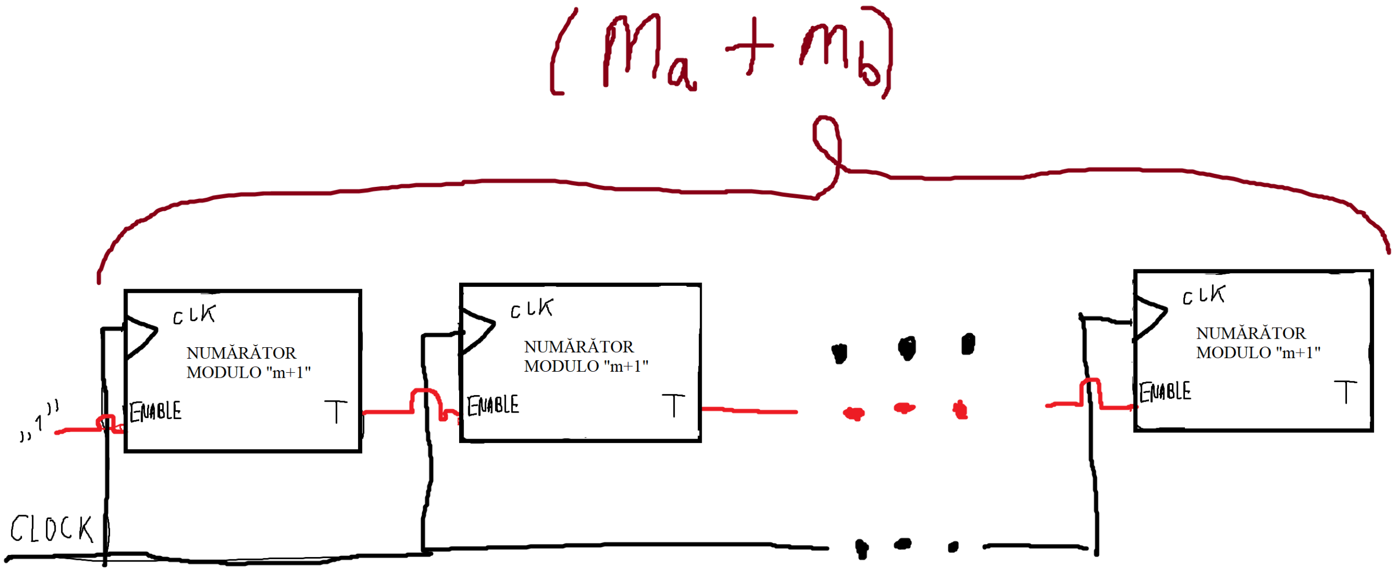
1) Se poate verifica dacă generăm numărul corespunzator de linii în matricea de puteri M.

2) Putem prealoca matricea M pentru a scuti matlab-ul de a realoca matricea M pentru fiecare iteraște pe care o realizăm, număr de iterații ce poate fi considerabil de mare. Acest fapt facilitează și creșterea vitezei de execuție a algoritmului.

3) Folosind această prealocare nu vor disparea problemele legate de memoria necesară executiei algoritmului, dar aceste erori de executare (runtime error) nu vor apărea atat de repede in program (avem garantia ca vom putea executa programul și pentru valori ale ordinului sistemului mai mari decât 5 și un grad al polinomului de cel putin 10).

4.2 Metoda 2

Diferențierea față de metoda 1 este reprezentată prin construirea în mod diferit a matricei de puteri M. Implementarea are la bază cascadarea a (na+nb) număratoare modulo (m+1). În practică, fiecare numărator când ar ajunge la valoarea maximă (m), la următorul clock, va da impuls de "enable" următorului numărtor cascadat, ca să îi mărească valoarea. (vezi figura 1)



Acest cod a fost implementat astfel încât la momentul în care suma valorilor din toate număratoarele va depăși valoarea lui m se va căuta numărătorul de stare cea mai nesemnificativă, ce conține o valoare, se va reseta și va determina următorul numărător să își marească valoarea. Algoritmul își va termina executarea în momentul în care ar ajunge să distribuie un impuls numărătorului (na+nb+1), care în realitate nu există. Practic în algoritm, clockul numărătorului este de fapt fiecare interație din bucla repetitivă.

5 Media erorilor pătratice și grafice reprezentative

Zerourile din tabele se datoreaza preinițializării matricelor cu valori de 0 și nu obținerii unor erori nule.

Observatie: nu am popula toate locurile din tabel din cauza timpului de executie foarte mare în principal cauzat de problema MatLab-ului de a lucra cu valori NaN.

De asemenea, pentru urmărirea ușoară a graficelor, fiecare coloană este corespunzătoare unui grad m din intervalul [1,15], iar liniile sunt corespunzătoare ordinului sistemului na=nb din intervalul [1,5].

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.0076	0.0011	1.6796e-04	2.8204e-05	5.1450e-06	9.2034e-07	1.7397e-07	3.6793e-08	8.7060e-09	1.9509e-09	4.1050e-10	6.3789e-11	8.7566e-12	0.0032	0.4034
2	0.0030	4.6284e-04	3.6228e-05	1.2064e-06	3.5905e-08	1.1673e-09	2.4990e-11	5.7283e-13	2.4090e-14	4.4777e-16	7.3271e-18	2.1341e-15	1.6951e-06	0.1819	0.3857
3	0.0027	2.9645e-04	1.2256e-05	8.2581e-08	5.4366e-10	1.1407e-12	5.8307e-15	9.7900e-18	3.6190e-20	6.3519e-18	4.9884e-12	1.9700e-06	0.1022	0.2591	0.4949
4	0.0027	2.2320e-04	2.6059e-06	1.6699e-09	8.4633e-14	1.8394e-24	3.5084e-28	1.9313e-26	4.8192e-23	5.6362e-10	2.4523e-07	0.0660	0.1950	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1: MSE pentru predicția identificării

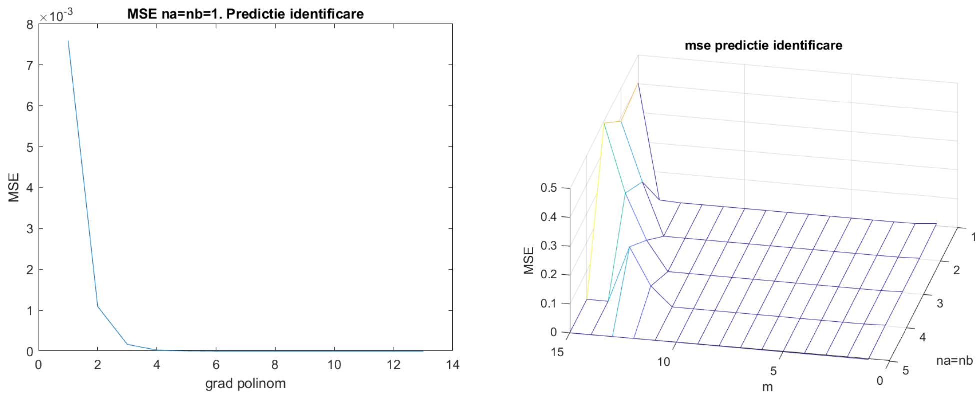


Figure 2: Graficele MSE pentru predicția identificării

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.1633	0.0178	0.0018	2.0017e-04	2.9758e-05	4.6354e-06	8.3600e-07	1.4166e-07	3.2298e-08	1.0987e-08	2.8811e-09	4.7766e-10	5.8505e-11	NaN	NaN
2	0.1763	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	0.1664	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	0.1684	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3: MSE pentru simularea identificării

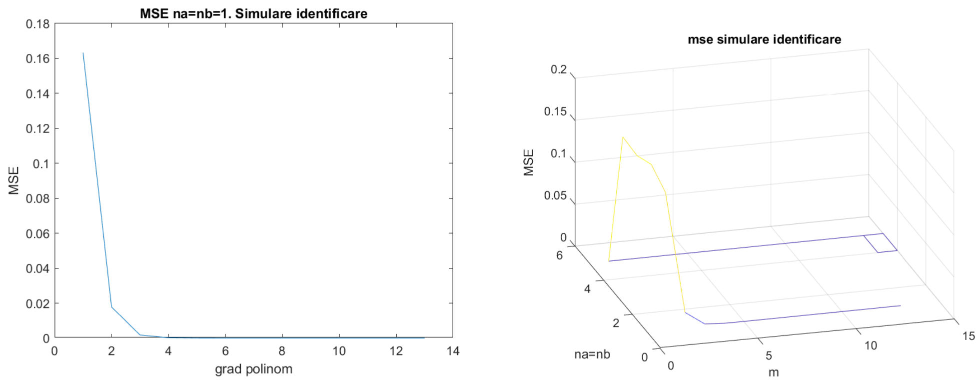


Figure 4: Graficele MSE pentru simularea identificării

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2.8301e-05	2.5132e-05	1.6180e-05	1.5946e-05	1.6071e-05	1.5874e-05	1.5854e-05	1.5916e-05	1.5891e-05	1.5881e-05	1.5889e-05	1.5890e-05	1.5889e-05	2.0594e-04	0.7293
2	3.3492e-05	0.3525	713.2259	5.3109e+05	609.2971	70.5502	0.0075	0.0311	1.8821e-04	1.7626e-05	1.6325e-05	1.5894e-05	0.1139	2.5155e+03	1.8012e+03
3	4.3825e-05	1.0731	7.7085e+05	1.2270e+06	4.5032e+04	0.3305	6.1181e-05	5.9082e-05	1.5889e-05	3.9757e-05	1.0050e-04	18.1608	550.4120	4.2840e+03	63.8909
4	4.5027e-05	0.0497	6.1370e+05	2.1139e+06	339.3165	0.0029	4.8726e-05	5.1263e-05	1.4218e-04	0.4872	4.3573	5.3301e+03	189.1837	281.1897	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5: MSE pentru predicția validării

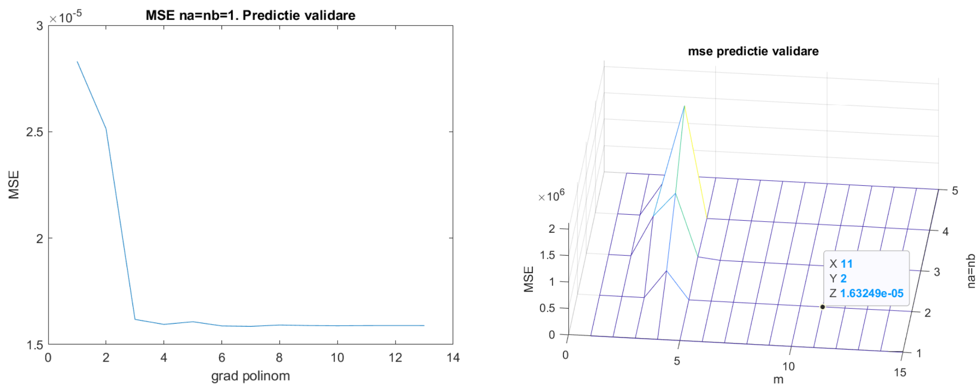


Figure 6: Graficele MSE pentru predicția validării

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.0013	0.0023	3.0100e-04	1.6165e-04	2.0297e-04	1.7034e-04	1.4864e-04	1.6303e-04	1.6088e-04	1.5799e-04	1.5876e-04	1.5949e-04	1.5925e-04	0.1001	5.8523
2	0.0043	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.5922e-04	1.5922e-04	1.5922e-04	NaN	5.8359	5.8559
3	0.0020	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.5922e-04	1.5921e-04	NaN	NaN	5.8318	5.8561	5.8577
4	0.0024	NaN	NaN	NaN	NaN	NaN	1.5930e-04	1.5935e-04	NaN	NaN	NaN	5.8040	5.8583	5.8577	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7: MSE pentru simularea validării

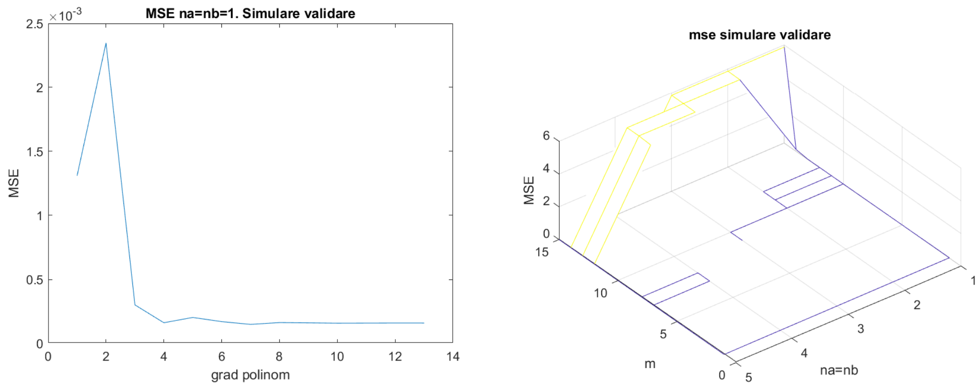


Figure 8: Graficele MSE pentru simularea validării

În urma observării atât a graficelor, cât și a tabelelor cu erori putem face trage următoarele concluzii:

-datele de identificare, la predicție, suferă același fenomen de supraantrenare, dar valorile erorilor nu sunt dezastruoase, ba chiar găsim erori de ordinul 10^{-28} cu cât creștem ordinul sistemului, deci comportarea este una bună, aproape ideală;

-asemenea predicției datelor de identificare, și predicția datelor de validare se comportă normal odată cu creșterea ordinului și a gradului, cu diferența că erorile pot avea discrepanțe foarte mari între ele în funcție de ordinul și gradul ales;

-simularea, atât pentru identificare, cât și pentru validare are un comportament mult mai pretențios, ordinul și gradul ales pentru aproximator fiind foarte importante în vederea obținerii unei aproximări bune. Spre deosebire de predicție, în cazul în care încercăm aproximarea unui sistem ce este la bază de ordin 1 cu un sistem de ordin 2, spre exemplu, erorile vor fi din ce în ce mai mari, iar ieșirea simulată poate ajunge sa tindă chiar și la infinit.

Observație

Discontinuitățile apărute in graficele realizate cu funcția mesh() sunt datorate valorilor NaN ale erorilor medii pătratice, valori ce nu sunt reprezentabile pe un grafic.

Considerând cele mai bune aproximări obținute în funcție de eroarea medie pătratică am ajuns la concluzia că pentru ordinul 1 si gradul 13 al aproximatorului vom obține cele mai bune aproximări. Aceste aproximări sunt prezentate mai jos:

5.1 Cele mai bune aproximări

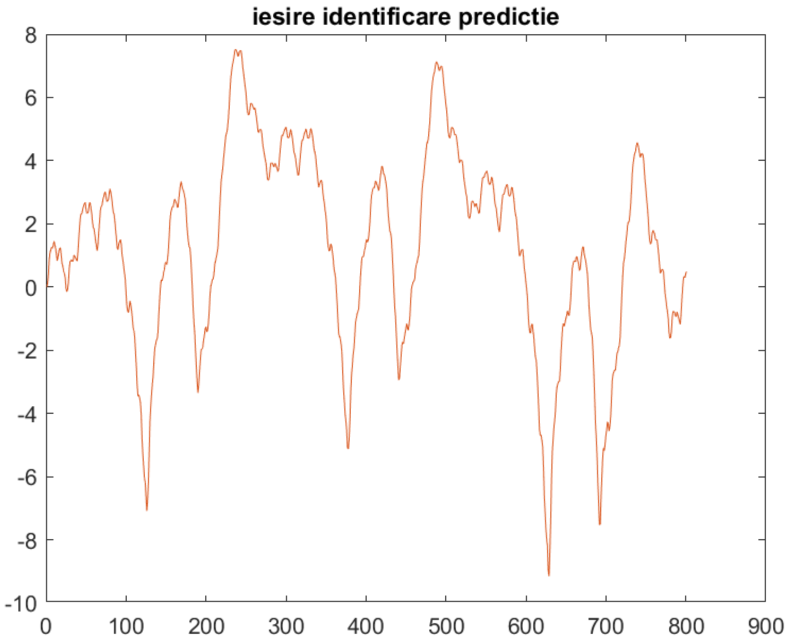


Figure 9: Predicția identificării: MSE=$8.75 \cdot 10^{-12}$

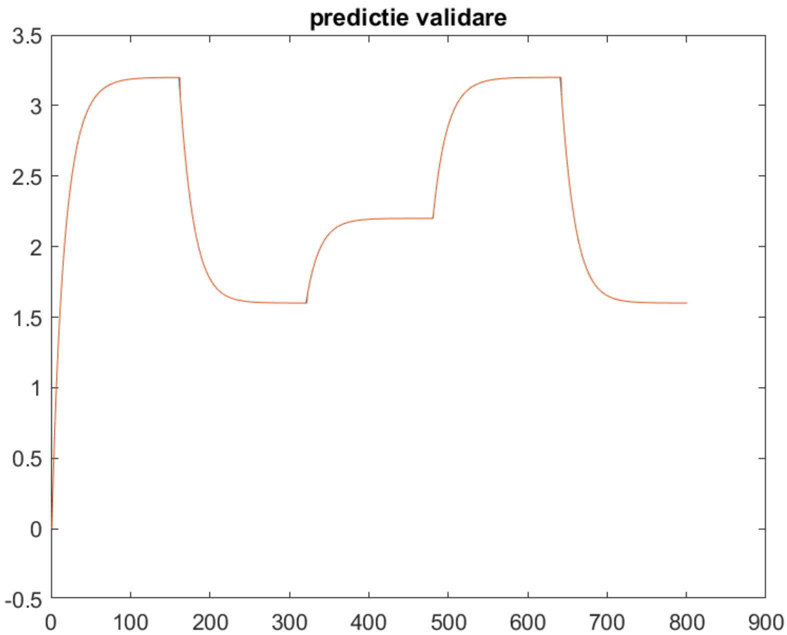


Figure 10: Predicția validării: MSE=$1.58 \cdot 10^{-5}$

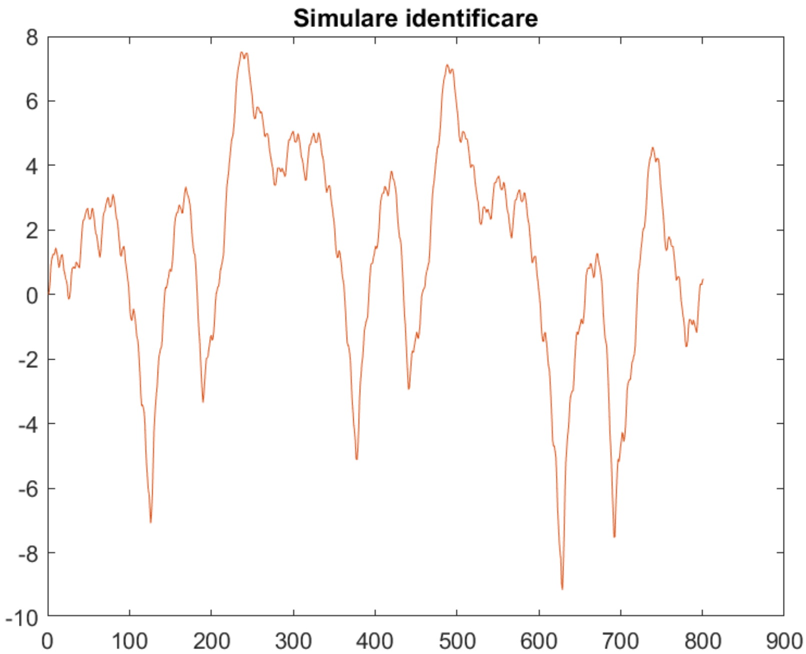


Figure 11: Simularea identificării: MSE=$5.85 \cdot 10^{-11}$

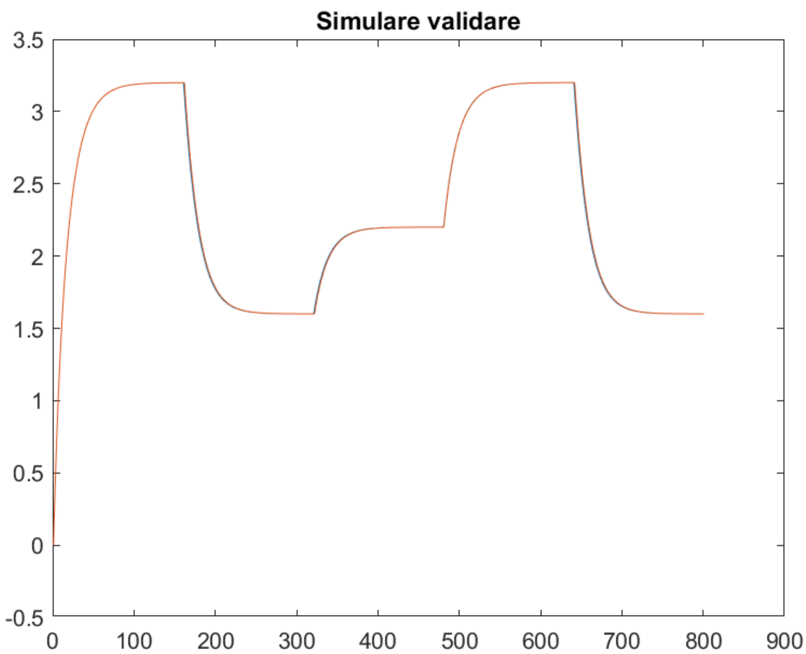


Figure 12: Simularea validării: MSE=$1.59 \cdot 10^{-4}$

6 Concluzii

- Asemenea metodei de regresie liniară, metoda ARX poate produce fenomenul de supraantrenare pentru grade prea mari ale aproximatorului (erori ce pot proveni și din erorile de calcul numeric la care pot fi supuse datele).

- Ordinul sistemului trebuie considerat cu atenție, astfel sunt șanse mari să nu putem obține atât predicțtii, dar mai ales simulări care se apropie de sistemul real.

- Metoda ARX, spre deosebire de regresie, oferă posibilitatea de a obține ieșirea sistemului fără a avea acces la ieșirea reală a acestuia cu ajutorul simulării.

- Deși este o metodă simplă, aceasta are limitările ei deoarece nu poate modela decât sisteme ale cărei intrări au Persistența Excitației suficient de mare (intrări care conțin suficientă informație $\iff$ pot pune sistemul în cât mai multe situații posibile) și în care perturbația este zgomot alb de medie zero.

7 Anexă

7.1 Metoda 1

7.1.1 Funcția care implementează ARX-ul

```
function [y_id_predictie,y_val_predictie,y_id_sim,y_val_sim] = arx_proiect(id,val,na,nb,nk,m)
%Am realizat o functie care calculeaza predictia si simularea prin metoda
%ARX neliniar atat pentru datele de identificare cat si pentru cele de
%validare.
%Functia are ca parametri de intrare datele de identificare, validare,
%numarul de zerouri na, numarul de poli nb, intarzierea nk si gradul
%aproximatorului polinomial m.
y_id_sim=[];%vecorul pentru simularea identificarii
y_val_sim=[];%vector pentru simularea datelor de validare
y_id_predictie=[];%vector pentru predictia identificarii
y_val_predictie=[];%vectorul pentru predictia datelor de validare
N=length(id.U);
a=[];
b=[];

puteri=0:m;%construim un vectori cu puteri de la 0 la m: [0 1 2....m]
x=cell(1,na+nb);%construim un cell array (o matrice cu matrici) de dimensiune 1x(na+nb).
%Avem na+nb astfel de matrici, deoarece cu ajutorul functiei polnvar2() vom
%genera un produs cartezian (din care vom scoate liniile de exponenti a caror suma este mai mare decat gradul maxim m)
%pentru na+nb seturi egale cu [0 1 2...m], care
%vor reprezenta puterile la care vom ridica fiecare dintre regresorii de
%baza [y(k-1),y(k-2),....,y(k-na),u(k-nk),u(k-nk-1),...,u(k-nk-nb+1)].
for k=1:na+nb
    x{k}=puteri;
end

puteri=polnvar2(x,m);%Liniile acestei matrice vor fi de forma [p1 p2 p3 ... p_(na+nb)], unde
%p1+p2+...+p_(na+nb)<=m;
%PREDICTIA DATELOR DE IDENTIFICARE
for k=1:N%k reprezinta indicele liniei din matricea de regresori pe care o construim.
    %Vom avea atatea linii cate esantioane vom avea in datele de
    %intrare/iesire.
    a=[];
    b=[];
    %Construim vectorul a ce va contine intrarile intarziate de forma
```

```
    %[y(k-1),y(k-2),....,y(k-na)]
    for i=1:na
        if(k-i<=0)%daca valoarea indexului k-i este mai mica sau egala decat 0,
            %vom considera ca iesirea sistemului este 0.
            a=[a,0];
        else
            a=[a,id.Y(k-i)];
        end
    end
    %La vectorul mai sus construit vom adauga intrarile intarziate ale
    %sistemului:[u(k-nk),u(k-nk-1),...,u(k-nk-nb+1)].Astfel vom obtine urmatorul vector cu
    %iesiri si intrari intarziate: [y(k-1),y(k-2),....,y(k-na),u(k-nk),u(k-nk-1),...,u(k-nk-nb+1)].
    for i=0:nb-1
        if(k-i-nk<=0)%vom considera intrarile la momentul 0 si inainte de momentul 0 ca fiind egale cu 0.
            a=[a,0];
        else
            a=[a,id.U(k-i-nk)];
        end
    end

    b1=a.^puteri;%ridicam fiecare termen din vectorul a la puterea pi corespunzatoare din matricea de puteri
    %construita mai sus
    b=prod(b1');%inmultim elementele de pe fiecare linie din din matricea b1 si astfel obtinem regresorii x1^p1*x2^p2*...*xn^pn.
    %A fost nevoie de transpusa matricei b1, fiindca functia prod()
    %realizeaza produsul elementelor de pe aceeasi coloana de pe o matrice iar elementele noastre sunt plasate pe linii.
    S1(k,:)=b;%Adaugam vectorul b mai sus construit la linia curenta k din matricea de regresori S1
end
T=S1\id.Y(:);%Folosim operatorul "\" pentru a rezolva sistemul si a afla parametrii T ai modelului ARX.
y_id_predictie=S1*T;%Realizam predictia pentru datele de identificare.

%PREDICTIA DATELOR DE VALIDARE
%Reluam acelasi algoritm explicat mai sus pentru predictia identificarii,
%doar ca de aceasta data vectorul de iesiri folosit va fi cel pentru datele de validare.
N2=length(val.U);
for k=1:N2
    a=[];
    b=[];
```

```
    for i=1:na
        if(k-i<=0)
            a=[a,0];
        else
            a=[a,val.Y(k-i)];
        end
    end
    for i=0:nb-1
        if(k-i-nk<=0)
            a=[a,0];
        else
            a=[a,val.U(k-i-nk)];
        end
    end

    b2=a.^puteri;
    b=prod(b2');
    S2(k,:)=b;
end

y_val_predictie=S2*T;%calculam predictia pentru datele de validare folosindu-ne de parametrii T calculati anterior
%prin procedeul de regresie.

%SIMULAREA DATELOR DE IDENTIFICARE
a=[];
b=[];
%Realizam simularea sistemului necunoscand valoarea iesirilor,
%determinandu-le pornind doar de la intrari, iar apoi folosindu-ne si de
%iesirile anterior determinate.
for k=1:length(id.U)
    a=[];
    b=[];
    %Asemenea predictiei, in vectorul a vom avea iesirile intarziate de
    %forma [y_sim(k-1),...,y_sim(k-na),u(k-nk),...,u(k-nb-nk+1)]. De observat
    %ca acum nu ne mai folosim de valoarea iesirii oferite in datele
    %initiale, ci de valorile determinate de noi la pasii anteriori.
     for i=1:na
        if(k-i<=0)
            %Consideram iesirile dinainte de momentul 0 si la momentul 0
            %egale cu 0, altfel alipim vectorului a[] valorile anterior simulate pentru iesire y_sim.
            a=[a,0];
        else
            a=[a,y_id_sim(k-i)];%alipim lui a iesirile simulate anterior : [y_sim(k-1),...,y_sim(k-na)].
        end
```

```
        end
        for i=0:nb-1
            if(k-i-nk<=0)
                a=[a,0];
            else
                a=[a,id.U(k-i-nk)];%Concatenam la vectorul de regresori de baza intrarile intarziate [u(k-nk),...,u(k-nb-nk+1)].
            end
        end

      b3=a.^puteri;%reluam pasul de creare a regresorilor explicat in partea de simulare.
      b=prod(b3');
      y_id_sim=[y_id_sim,b*T];%construim treptat vectorul de iesire simulat y_sim
      %inmultind vectorul ce contine combinatiile de regresori de tip
      %polinomiali x1^p1*...*xn^pn de grad p1+p2+...+pn<=m cu vectorul de
      %parametri T determinati in prima parte a functiei.

end


%SIMULAREA DATELOR DE VALIDARE
%Pentru validare, aplicam acelasi algoritm explicat mai sus pentru partea
%de simulare a identificarii.
a=[];
b=[];

for k=1:length(val.u)
      a=[];
      b=[];
      for i=1:na
          if(k-i<=0)
              a=[a,0];
          else
              a=[a,y_val_sim(k-i)];
          end
      end
      for i=0:nb-1
          if(k-i-nk<=0)
              a=[a,0];
          else
              a=[a,val.U(k-i-nk)];
          end
      end

      b4=a.^puteri;
```

```
    b=prod(b4');
    y_val_sim=[y_val_sim,b*T];
end

end
```

7.1.2 Funcția care realizează matricea de puteri

```
function [mat_puteri] = polnvar2(x,m)%Functie care genereaza toate puterile unui polinom de n variabile (n=numarul de cell-uri ale unui cell-array x)
%ce contine in fiecare x{i} puterile disponibile pentru a alcatui acel
%polinom de grad maxim m.
%In acest caz, vom folosi in fiecare cell x{i} acelasi vecor.
%Algoritmul se bazeaza pe realizarea produsului cartezian a celor n
%cell-uri (care sunt vectori linie) ale lui x, eliminand din acest produs
%cartezian elemente in momentul in care suma puterilor ajunge mai mare
%decat m;
dim=size(x);%retinem in dim dimensiunea cell-array-ului x. Ne intereseaza dim(2), care reprezinta na+nb.
n_linii=factorial(dim(2)+m)/factorial(dim(2))/factorial(m);%acesta este numarul de termeni ai unui polinom de grad m si n variabile.
mat_puteri=zeros(n_linii,dim(2));%alocam spatiul necesar matricii de puteri.
aux=mat_puteri;%Vom considera si o matrice auxiliara pentru a realiza produsul cartezian.
l2=1;%Cu ajutorul lui l2 vom sti lilia curenta pana la care am ajuns sa construim perechile de puteri.
for k=1:dim(2)
    l=1;%reactualizam contorul l cu care vom parcurge liniile matricei cu puteri construite pana la un moment dat.
    for n=1:l2%parcurgem matricea auxiliara aux de puteri pana la indicele ultimei linii construite la iteratia k-1.
        for j=1:length(x{k})%parcurgem vectorul x{i} element cu element
            if(sum([aux(n,1:k-1),x{k}(j)])<=m)%verificam daca prin concatenarea liniei n a matricei aux cu un element din vectorul x{k}
                %nu depasim conditia ca suma tuturor puterilor sa fie mai
                %mica sau egala decat m.
                mat_puteri(l,1:k)=[aux(n,1:k-1),x{k}(j)];%daca conditia e respectata alipim elementul vectorului x{k} la matricea de puteri
                %aux(n,1:k-1) reprezinta faptul ca pana la iteratia curenta
                %k am construit k-1 perechi de puteri [p1,p2..p_(k-1)].
                l=l+1;%odata gasita o noua pereche trecem mai departe, pe urmatoarea linie in matricea pe care dorim sa o
```

```
construim.
                end
            end
        end
        l2=l-1;%indicele liniei din matricea aux[][] pana la care este nevoie sa iteram este indicele ultimei linii construite in matricea mat_puteri[][].
        aux=mat_puteri;%cu noua matrice obtinuta mat_puteri[][], reactualizam matricea auxiliara.
end
%Pe scurt matricea aux[][] ne permite sa reactualizam si sa suprascriem linii din
%matricea de puteri mat_puteri[][] fara sa pierdem perechile construite anterior.

end
```

7.1.3 Script-ul care apeleaza funcția ARX

```
clear
%Incarcam datele de identificare si de validare si le reprezentam grafic
%forma datelor de validare ne va indica posibilul ordin al sistemului
%asupra caruia trebuie sa aplicam metoda ARX (deoarece intrarea de validare a sistemului este o treapta si pentru acest
%semnal de intrare cunoastem formele raspunsului unui sistem).
load('iddata-08.mat');
figure(1)
subplot(2,2,1),plot(id.U),title('intrare de identificare');
subplot(2,2,2),plot(id.Y),title('iesire de identificare');
subplot(2,2,3),plot(val.U),title('intrare de validare');
subplot(2,2,4),plot(val.Y),title('iesire de validare');
%Dupa observarea formei datelor de validare am identificat ordinul
%sistemului ca fiind 1.
% na=1;%numarul de zerouri
% nb=1;%numarul de poli
nk=1;%intarzierea
m_max=15;%gradul maxim pana la care am considerat aproximatorul
n=5;%ordinul maxim considerat pentru sistem
MSE_predictie_val=zeros(n,m_max);
MSE_simulare_val=zeros(n,m_max);
MSE_predictie_id=zeros(n,m_max);
MSE_simulare_id=zeros(n,m_max);
%Calculam erorile patratice pentru diferite combinatii ale ordinelor si
%gradelor.
for grad=1:n
    y_id_predictie=[];y_val_predictie=[];y_id_sim=[];y_val_sim=[];
    for m=1:m_max
        [y_id_predictie,y_val_predictie,y_id_sim,y_val_sim]=arx_proiect(id,val,grad,grad,nk,m);
```

```
            MSE_simulare_val(grad,m)=mean((val.Y-y_val_sim').^2);
            MSE_predictie_val(grad,m)=mean((val.Y-y_val_predictie).^2);
            MSE_predictie_id(grad,m)=mean((id.Y-y_id_predictie).^2);
            MSE_simulare_id(grad,m)=mean((id.Y-y_id_sim').^2);
      end
end

%%
%Apelarea functiei pentru ARX folosind datele cu cele mai mici erori:
%na=nb=1, nk=1, m=13
clear
load('iddata-08.mat');

[y_id_predictie,y_val_predictie,y_id_sim,y_val_sim]=arx_proiect(id,val
    ,1,1,1,13);
figure(2)
plot(id.Y),hold on,plot(y_id_sim),title('Simulare identificare');

figure(3)
plot(val.Y),hold on,plot(y_val_sim),title('Simulare validare');

figure(4)
plot(id.Y),hold on,plot(y_id_predictie),title('iesire identificare
    predictie'),hold off;

figure(5)
plot(val.Y),hold on,plot(y_val_predictie),title('predictie validare'),
    hold off;
```

7.2 Metoda 2

7.2.1 Funcția care generează afișează graficele

```

clc
clear
format long
%Citire
load('iddata-08.mat')

u_id = id.u; y_id = id.y; u_val = val.u; y_val = val.y;

%Alegere parametrii: n nk m
n= 3; nk = 1;
m_max = 14;

%Genare matrice cu erori medii p tratice
MSE_pred_val=zeros(n,m_max);
MSE_sim_val=zeros(n,m_max);
```

```
MSE_pred_id=zeros(n,m_max);
MSE_sim_id=zeros(n,m_max);

for ordin = 1:n
    for grad = 1:m_max
        %Pentru un anumit ordin n  i  un anumit grad m se genereaz
        %aproxim rile
        [y_pred_id, y_pred_val, y_sim_id, y_sim_val] = calcul_ARX_Neliniar(ordin,ordin,nk,grad,id,val);

        %Adaug m erorile medii p tratices
        MSE_sim_val(ordin,grad)=mean((val.Y-y_sim_val(:)).^2);
        MSE_pred_val(ordin,grad)=mean((val.Y-y_pred_val(:)).^2);
        MSE_pred_id(ordin,grad)=mean((id.Y-y_pred_id(:)).^2);
        MSE_sim_id(ordin,grad)=mean((id.Y-y_sim_id(:)).^2);
    end
end


%% Afi are aproxim ri pentru un anumit set ini iat na,nb,nk,mkk

clc
clear
format long
%Citire
load('iddata-08.mat')

u_id = id.u; y_id = id.y; u_val = val.u; y_val = val.y;

%Alegere parametrii: na nb nk m
na = 1; nb = 1; nk = 1;
m = 13;

[y_pred_id, y_pred_val, y_sim_id, y_sim_val] = calcul_ARX_Neliniar(na,nb,nk,m,id,val);


% Predictie identificare
figure(1)
plot(y_pred_id); hold on; plot(y_id); hold off;
xlabel('index');ylabel('y'); legend('y_{prezis}','y');
title('Predictie identificare');

% Simulare identificare
figure(2)
plot(y_sim_id); hold on; plot(y_id); hold off;
xlabel('index');ylabel('y'); legend('y_{simulate}','y');
title('Predictie identificare');
```

```

% Predictie validare
figure(3)
plot(y_pred_val); hold on ; plot(y_val); hold off;
xlabel('index');ylabel('y'); legend('y_{prezis}','y');
title('Predictie validare');

% Simulare validare
figure(4)
plot(y_sim_val); hold on ; plot(y_val); hold off;
xlabel('index');ylabel('y'); legend('y_{simulat}','y');
title('Simulare Validare');

% Afisare erori
disp(strcat('MSE predictie identificare: ',num2str(mean((id.y-y_pred_id(:)).^2))));
disp(strcat('MSE simulare identificare: ',num2str(mean((id.y-y_sim_id(:)).^2))));
disp(strcat('MSE predictie validare: ',num2str(mean((val.y-y_pred_val(:)).^2))));
disp(strcat('MSE simulare validare: ',num2str(mean((val.y-y_sim_val(:)).^2))));
```

7.2.2 Funcția care creecază matricea de puteri

```
function MatricePutere = Calculare_matrice_putere(na,nb,m)

% Algoritmul are la baza implementarea solutiei
% numaratorului in baza m, practic el va oferi
% rezultatele numararii in baza m, doar ca el este optimizat
% astfel nct s nu se poat produce liniile n care
% suma valorilor sa fie mai mare dec t m

MatricePutere = []; % declaram variabila de iesire ca fiind o matrice

vector_puteri = zeros(1,na+nb); % Cream vectorul in care vom numara

pozitie_max = 1; % Aceasta este pozitia in vector unde se va
% gasii ultima valoare, in momentul in care ea va depasi
% pozitie_max > na+nb, algoritmul va returna
% ultima valoare, fiind elementul ce contine doar 0

N_vector_puteri = length(vector_puteri); % variabila in care stocam lungimea
% elementelor

    while pozitie_max <= N_vector_puteri

        % Se pot adauga valori celui mai nesemnificativ bit
```

```
        % Astfel se vor adauga valori bitului nesemnificativ cat timp
        % suma valorilor va fi mai mic  decat ”m”
        if sum(vector_puteri) < m
            vector_puteri(1) = vector_puteri(1) + 1;

        %in cazul in care adaugarea ar depasii valoare m,
        %se va reinitializa la 0 continutul pozitiei unde exista o valoare, cautand de
        %la cel mai mic index si marindu-se valoarea imediat urmatoare
        %fat  de indexul g sit
        else
            for increment = 1:pozitie_max

                %Daca se gaseste o valoare, la un anumit index, cautat crescator
                %se va initializa la 0
                if vector_puteri(increment) > 0
                    vector_puteri(increment) = 0;

                    %in cazul in care indexul gasit este la pozitia maxim
                    %pana unde s-au pus valori, el se va m ri
                    if increment == pozitie_max
                        pozitie_max = pozitie_max+1;
                    end

                    %Se va incrementa pozitia urm toare fat  de indexul
                    %unde s-a g sit valoarea
                    if increment+1 <= N_vector_puteri
                        vector_puteri(increment+1) = vector_puteri(increment+1)+1;
                    end

                    %Se va inchide forul deoarece s-a incrementat o dat o
                    %valoare
                    break;
                end
            end
        end
            %se adauga la matrice, linia corespunz toare valorilor curente.
            MatricePutere = [MatricePutere;vector_puteri];
    end
end
```

7.2.3 Funcția care identifică parametrii

```
function Theta = identificare_Theta_ARX(id,Matrice_puteri,na,nb,nk)
```

```
    %Citire valori
    u_id = id.u;
    y_id = id.y;

    N = length(u_id);
    % Calculare Matrice de regresori
    Regresori_NARX = [];

    for k = 1:N

        %Declaram linia curent  ca fiind un vector sterg nd valorile
        %anterioare
        linie_curenta = [];

        %Se adauga elementele corespunz toare ie irii
        for pozitie_na = 1:na
            %consider m ini ial valoarea 0
            valoare_curenta = 0;
            index_y = k-pozitie_na;

            %in cazul in care indexul este unul valid(>0)
            %se va extrage acea valoare
            if index_y > 0
                valoare_curenta = y_id(index_y);
            end

           %se adauga la linia curent  valoarea curent
           linie_curenta = [linie_curenta,valoare_curenta];
        end

        %Se adauga elementele corespunz toare intr rii
        for pozitie_nb = 0:nb-1
            %consider m ini ial valoarea 0
            valoare_curenta = 0;
            index_u = k - nk - pozitie_nb ;

            %in cazul in care indexul este unul valid(>0)
            %se va extrage acea valoare
            if index_u > 0
                valoare_curenta = u_id(index_u);
            end

            %Se adaug  la linia crent  valoarea curent  obtinut
            linie_curenta = [linie_curenta,valoare_curenta];
        end

        %Se va ridica fiecare element din linie la toate puterile
```

```
        %obtinute. Practic fiecare linie va avea o combinatie de puteri
        %o putere pentru un element
        termeni_puteri = linie_curenta.^Matrice_puteri;

        %Matricea ob inut  se va transpune astfel   nct   s   existe
        %elementele ridicate la combinatiile de puteri pe coloan
        termeni_puteri = termeni_puteri';

        %Se va aplica func ia prod, ea va inmul ii toate elementele de
            pe o
        %linie   ntre  ele, rezultatul va fi c   se va ob ine un vector
            linie,
        % n  care la fiecare pozi ie se vor gasii termenii inmultiti
            intre
        %ei la cate o combina ie de puteri
        termeni_puteri = prod(termeni_puteri);

        %Adaugam noua linie ob inut  la matricea de regresori
            neliniaris
        Regresori_NARX = [Regresori_NARX;termeni_puteri];
    end

    % Creare matrice Y
    Y = y_id(:);

    % Aflare Parametrii Theta
    Theta = Regresori_NARX\Y;

end
```

7.2.4 Funcția care calculează aproximarea prin predicție sau simulare

```
function y_aproximat = calcul_y(u,y,na,nb,nk,Theta,Matrice_puteri)
%u-intrarea

%y-iesirea de validare. Daca exista se va folosi predictia, altfel
%simularea
%na - numarul de zerorui
%nb - numarul de poli
%nk - intarzierea
%Theta - Parametrii functiei
%Matrice_puteri - matricea ce contine toate puterile


    y_aproximat(1) = 0; %variabila in care se v-a construi simularea

    N_val = length(u); %variabila pentru stocarea lungimii intr rii

    goSimulate = 0; % variabil  care va interpreta dac  se doreste
```

```
        simulare
    %sau predictie. 0 - predictie, 1-simulare. initial consideram ca se
    %doreste predic tia

    %in cazul in care nu s-a dat ca parametru y, o iesire valida, se
    %considera c  se dore te s  se ob in  simularea.
    if isempty(y)
       goSimulate = 1; %
    end

     for index = 1 : N_val

         if goSimulate == 1
             y = y_aproximat; %Practic la fiecare itera ie , in cazul
             %in care se dore te simularea, se va reactualiza vectorul
             %de valori aproximate.
         end

         %Creare  vector cu valori anterioare
         val_anterioare = [];

         % Se adauga elementele corespunz toare ie irii
         for pozitie_na = 1:na

             %consider m ini ial valoarea 0
             valoare_curenta = 0;
             index_y = index - pozitie_na;

             %in cazul in care indexul este unul valid(>0)
             %se va extrage acea valoare
             if index_y > 0
                 valoare_curenta =  y(index_y);
             end

             val_anterioare = [val_anterioare,valoare_curenta];
         end

         for pozitie_nb = 0:nb -1
             %consider m ini ial valoarea 0
             valoare_curenta = 0;
             index_u = index -nk - pozitie_nb;

             %in cazul in care indexul este unul valid(>0)
             %se va extrage acea valoare
             if index_u > 0
                 valoare_curenta = u(index_u);
             end

```

```
            val_anterioare = [val_anterioare,valoare_curenta];
        end

        %Se va ridica fiecare element din linie la toate puterile
        %obtinute. Practic fiecare linie va avea o combinatie de puteri
        %o putere pentru un element
        linie_regresori = val_anterioare.^Matrice_puteri;

        %Matricea ob inut  se va transpune astfel  nct  s  existe
        %elementele ridicate la combinatiile de puteri pe coloan
        linie_regresori = linie_regresori';

        %Se va aplica func ia prod, ea va inmul ii toate elementele de pe o
        %linie  ntre  ele, rezultatul va fi c  se va ob ine un vector linie,
        % n  care la fiecare pozi ie se vor gasii termenii inmultiti intre
        %ei la cate o combina ie de puteri
        linie_regresori = prod(linie_regresori);


        %Aflare aproximare y, la un anumit index
        y_aproximat(index) = linie_regresori * Theta;
    end


end
```

7.2.5 Funcția care returnează aproximările prin simulare și predicție

```
function [y_pred_id,y_pred_val,y_sim_id,y_sim_val] = calcul_ARX_Neliniar(na,nb,nk,m,id,val)

    u_id = id.u;
    y_id = id.y;
    u_val = val.u;
    y_val = val.y;

    % Aflare lungime iesire, intrare
    N = length(u_id);
    N_val = length(u_val);

    % Calculare toate combinatiile de puteri;
    Matrice_puteri = Calculare_matrice_putere(na,nb,m);

    %Aflare Theta
    Theta = identificare_Theta_ARX(id,Matrice_puteri,na,nb,nk);

```

```
    % Calculare y aproximat pe identificare cu predictie
    y_pred_id = calcul_y(u_id,y_id,na,nb,nk,Theta,Matrice_puteri);

    % Calculare y aproximat pe identificare cu simulare
    y_sim_id = calcul_y(u_id,[],na,nb,nk,Theta,Matrice_puteri);

    % Calculare y aproximat pe validare cu predictie
    y_pred_val = calcul_y(u_val,y_val,na,nb,nk,Theta,Matrice_puteri);

    % Calculare y aproximat pe validare cu simulare
    y_sim_val = calcul_y(u_val,[],na,nb,nk,Theta,Matrice_puteri);
end
```