Proiect

Identificarea sistemelor 2018

Tirlescu Cristian-Andrei(andreitirlescu@gmail.com)

Grupa 30136

Partea 1. Modelarea unei functii necunoscute

Cuprins

1. Introducere

Se da un set de date de intrare-iesire, unde iesirea este generata de o functie necunoscuta, neliniara dar statica. Iesirea este afectata de zgomot, pe care il vom presupune aditiv, Gaussian, si de medie zero. Setul de date are doua variabile de intrare, x1 si x2, si o variabila de iesire, y, functia depinzand de cele trei variabile. Al doilea set de date este dat pentru validarea modelului.

Obiectivul nostru este dezvoltarea unui model pentru aceasta functie folosind un aproximator polinomial de grad m minim, care aproximeaza cel mai bine functia de baza si care sa obtina cea mai mica valoare a erorii medii patratice pe datele de validare. Pentru acest lucru, vom antrena modelul pe datele de identificare cu ajutorul regresiei liniare, urmand sa fie validate pe datele din setul al doilea.

2. Notiuni teoretice de implementare

2.1. Problema de regresie

Se da un set de date (x1(k), x2(k), y(k)), unde x1,x2 sunt intrari aplicate unei functii necunoscute g, y este iesirea corespunzatoare, posibil afectata de zgomot, iar k este un index de esantionare. Pentru k=1,…N, N fiind numarul de esantioane, se considera un vector $\varphi_i(k) \in R^n$, i = 1,…,n, care contine regresorii $\varphi(k) = [\varphi_1(k), \varphi_2(k), \dots, \varphi_n(k)]^T$. n este numarul de parametri regresori ai aproximatorului polinomial.

Scopul este identificarea vectorului de parametrii $\theta \in R^n$, folosind modelul liniar $y(k) = \varphi(k)^T \theta$.

Un exemplu de polinom de gradul 2 cu doua variabile de intrare x = $[x1, x2]^T$ este: $\tilde{g}(x) = \theta_1 + \theta_2 x_1 + \theta_3 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \theta_6 x_1 x_2$.

2.2. Sistem liniar

Pentru fiecare k =1,…,N se obtine un sistem de ecuatii liniare care poate fi scris sub forma matriceala:

$$\begin{matrix} y(1) \\ \dots \\ y(N) \end{matrix} = \begin{matrix} \varphi_1(1), \varphi_2(1), \dots, \varphi_{n1}(1) \\ \dots \\ \varphi_1(N), \varphi_2(N), \dots, \varphi_{n1}(N) \end{matrix} * \begin{matrix} \theta_1 \\ \dots \\ \theta_n \end{matrix}$$

$$Y = \Phi\,\theta$$

unde $Y \in R^N$, $\Phi \in R^{N\,x\,n}$.

2.3. Matricea de regresori Φ

Pentru aceasta am folosit polinomul generat aplicat pentru fiecare linie a matricii Φ. Polinomul contine toate combinatiile de puteri intre datele de intrate x1 si x2, care se bazeaza pe parcurgerea unei matrici M ∈ $R^{x1\,x\,x2}$. Dupa aplicarea acestui algoritm va rezulta matricea de regresori $\Phi \in R^{x1*x2\,x\,n}$, unde numarul N de lini este dat de numarul total de combinatii ale datelor de intrare, iar numarul n de coloane este dat de numarul de regresori ai polinomului.

In cazul de faza, avem de calculate doua matrici Φ, una pentru datele de identificare si una pentru datele de validare, urmand sa fie folosite pentru a calcula vectorul de parametrii θ si functia aproximata $\tilde{g}\_id$, respectiv $\tilde{g}\_val$.

2.4. Vectorul de parametrii θ

Calculul acestuia se face folosind algebra liniara. Plecand de la ecuatia sistemului liniar Y = Φ θ, dupa aplicarea a cativa pasi de algebra liniara se ajunge la ecuatia : $\hat{\theta} = [\Phi^T \Phi]^{-1} \Phi^T Y$.

MATLAB dispune de functia linsolve() care rezolva intr-un mod optim sistemul de ecuatii liniare, functie pe care o vom folosi in determinarea vectorului de parametrii. Apelarea ei se face cu comanda: θ = linsolve(Φ, Y).

2.5. Functia aproximata $\tilde{g}$

Dupa determinarea matricii Φ si vectorului de parametrii θ, aflarea functiei aproximate se face foarte usor, folosind produsul celor doua matrici, cu observatia ca, pe datele de validare se va folosi acelasi θ.

2.6. Eroarea medie patratica (MSE)

Caracteristic regresiei liniare, pentru a verifica ca modelul este bine determinat se calculeaza eroare medie patratica dupa formula: MSE $= \frac{1}{N^2} \sum_{k=1}^{N} (y - \tilde{g})^2$.

3. Interpretare rezultate

3.1. Alegerea gradului si fenomenul de supraantrenare

In Tabel 1 sunt prezentate valorile MSE-ului pentru anumite valori ale lui m.

m(grad polinom)	MSE-identificare	MSE-validare
1	6.0532	5.6949
2	0.6150	0.6196
3	0.6148	0.6197
4	0.5693	0.5752
5	0.5689	0.5753
6	0.4857	0.4878
7	0.4851	0.4883
8	0.4845	0.4890
9	0.4836	0.4894

10	0.2646	0.2729
11	0.2631	0.2748
12	0.1289	0.1448
13	0.1270	0.1463
14	0.1082	0.1311
15	0.1055	0.1341
16	0.1029	0.1344
17	0.1006	0.1383
18	0.0995	0.1418
19	0.0983	0.1469
20	0.1014	0.1554
21	0.1708	0.2686
22	0.2244	0.3408
23	0.4198	0.5470
24	0.9117	1.2406
25	1.3624	1.9776
26	1.5972	2.3866
27	1.7393	2.5941
28	1.9304	3.9464
29	2.5401	7.5469
30	2.8226	7.5458
31	2.9020	46.1721
32	3.1504	52.7203
33	4.0958	45.2614
34	4.3674	98.0697

Tabel 1. Valori MSE pentru diferite grade m

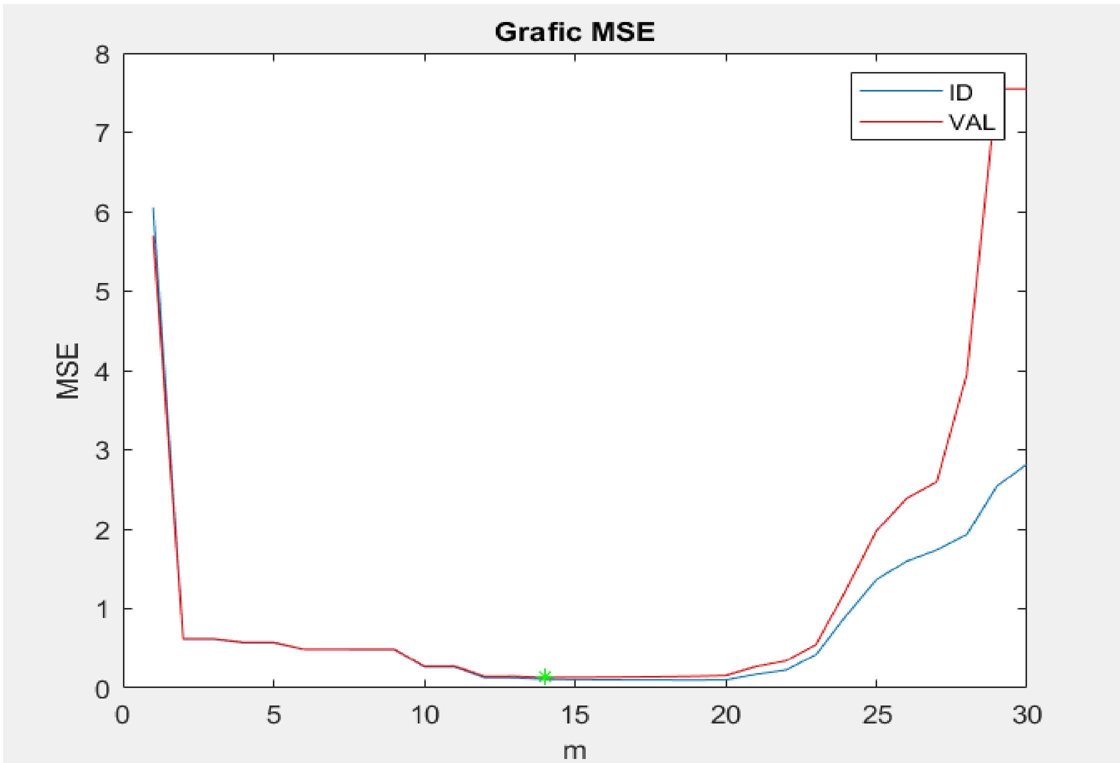

Fig.1 - Grafic MSE

Pe baza valorilolor din Tabelul 1, am identificat MSE-ul minim pe datele de validare ca fiind 0.13114 pentru m = 14. S-au plotat MSE-urile pana la gradul m=30 deoarece valoarea erorii creste exagerat dupa acest grad, iar graficul ar fi aratat ca o linie aproape de 0 urmand sa creasca brusc la niste valori foarte mari.

In Fig.1 se observa fenomenul de supraantrenare dupa un grad m > 24. Eroarea medie patratica de pe datele de validare incepe sa creasca exponential dupa gradul 24. Chiar daca pe datele de identificare, valoarea MSE-ului ramane o valoarea mica, pe datele de validare nu este asa. Acest lucru este datorat zgomotului aflat pe datele de iesire. La un grad ridicat, regresorii aproximeaza zgomotul, ceea ce va influenta valoarea vectorului de parametrii θ, care va interveni asupra functiei de aproximare a datelor de validare.

Un exemplu de zgomot aditiv, Gaussian si de medie zero este reprezentat in Fig.2. In MATLAB exista functia awgn() care genereaza acest zgomot. Graficul a fost luat din documentatia functiei.

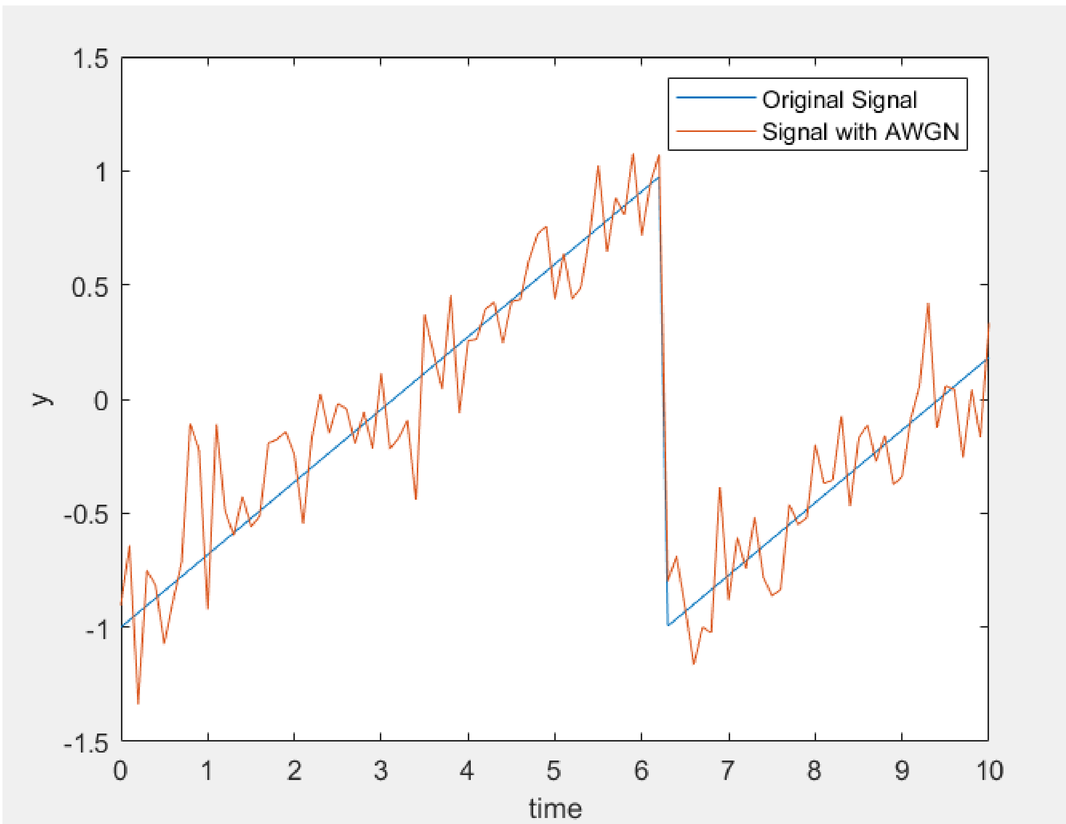

Fig.2 – Zgomot

3.2. Antrenare aproximatoare de diferite grade + Plot-uri

3.2.0. Date initiale

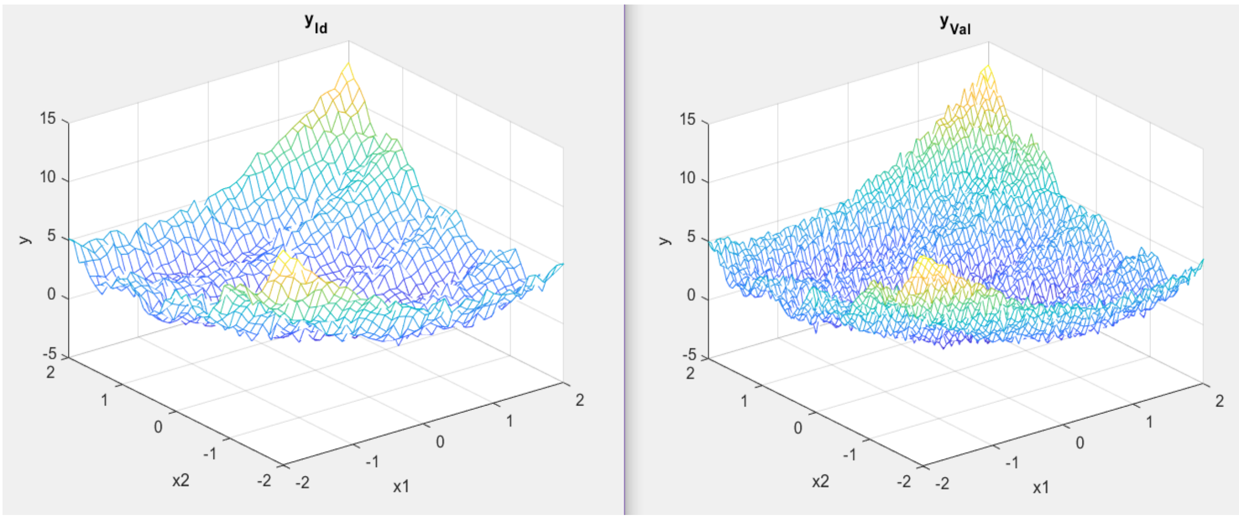

Fig.3 - Plot date initiale

3.2.1. Grad polinom m = 5

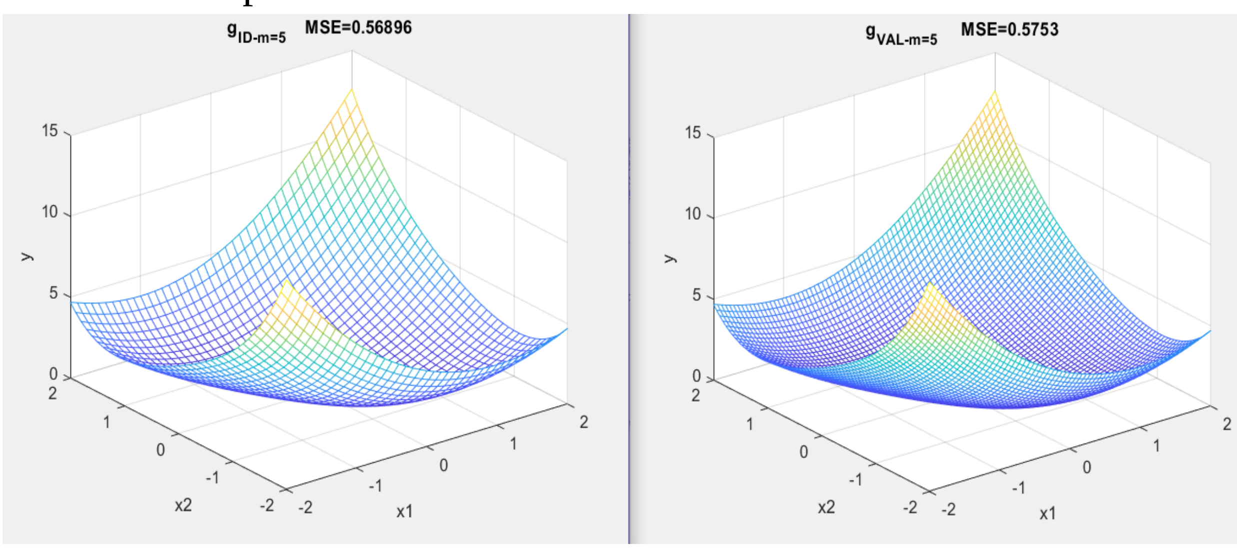

Fig.5 - Plot polinom grad 5

3.2.2. Grad polinom m = 14

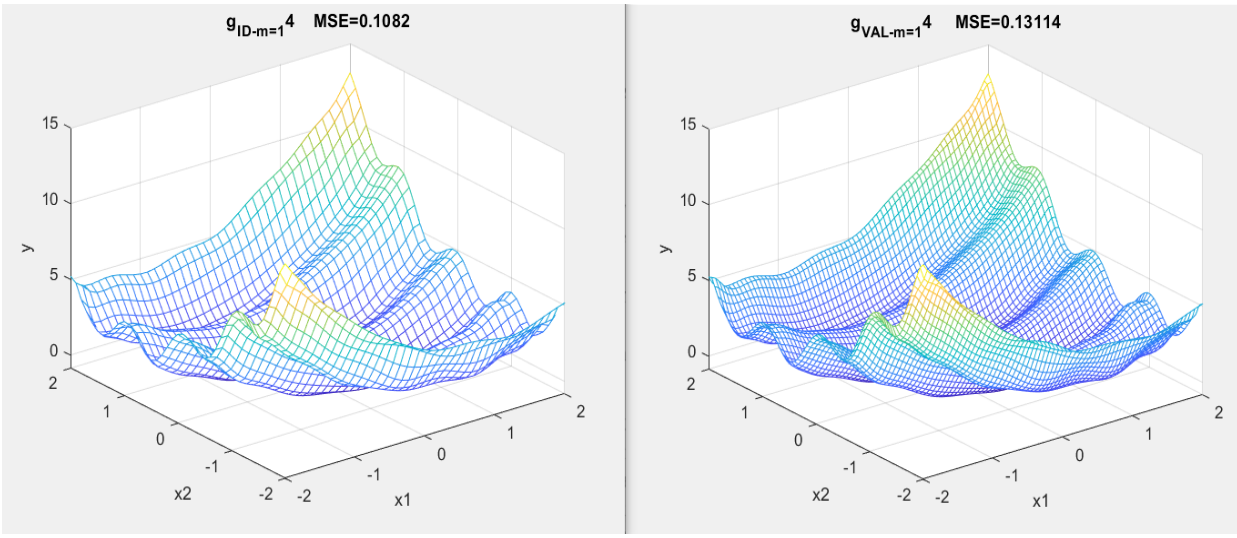

Fig.4 - Plot polinom grad 14 (optim)

3.2.3. Grad polinom m = 25

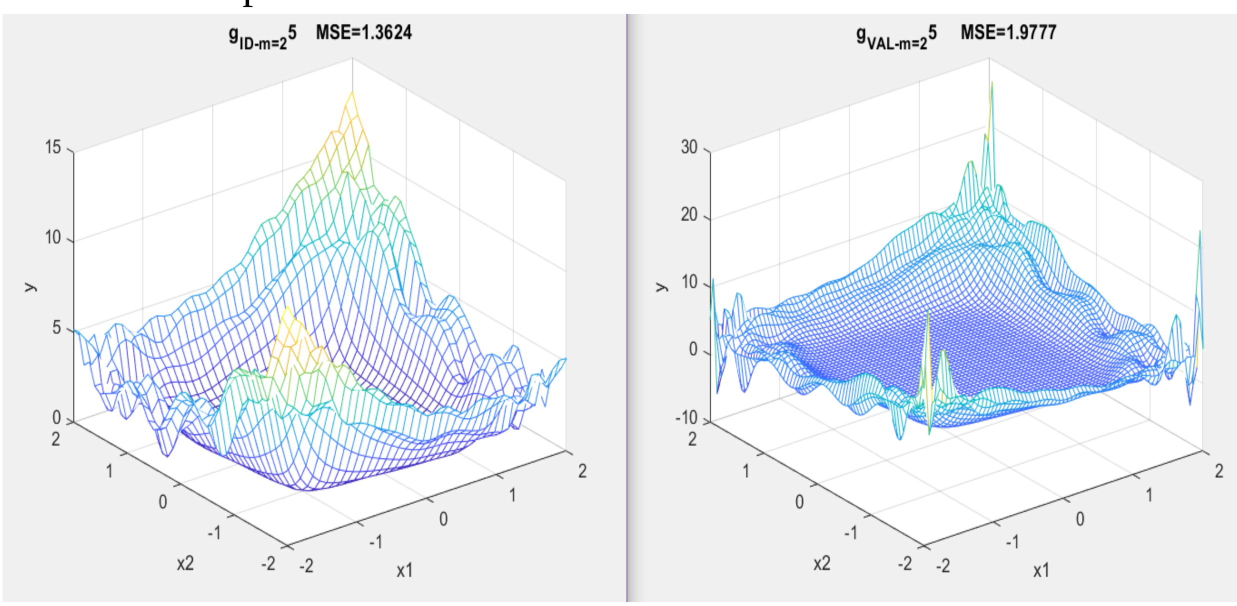

Fig.6 - Plot polinom grad 25

4. Discutie

In Fig.3 sunt prezentate plot-urile pentru datele de identificare si validare. Pe datele furnizate am antrenat polinomul, astfel in Fig.4 se observa ca pentru gradul 5, regresorii nu au gradul suficient de ridicat incat sa aproximeze curbele din datele initiale. La celalalt capat, pentru un m = 25, suficient de mare, regresorii aproximeaza zgomotul ce conduce la supraantrenare, fenomen observat in Fig6.

Folosind regresia liniara, am aflat cel mai bun grad m al polinomului care minimizeaza eroare medie patratica pe datele de validare, m = 14. In Fig.4 se observa acest lucru.

5. Script MATLAB

5.1. Functia de creare a regresorilor

```
function x = creaza_pol_n(m,k1,k2,x1,x2)
x = [1];
    for i=1:m
        for j=1:m
            if((i==j) && (i+j)<=m)
                x = [x (x1(k1).^i)*(x2(k2).^i) x1(k1).^i x2(k2).^i];
            elseif((i==j) && (i+j)>m)
                x = [x x1(k1).^i x2(k2).^i];
            elseif((i<m)&&(j<m))
                x = [x x1(k1).^i*x2(k2).^j];
            end
        end
    end
end
```

5.2. Script determinare grad m pentru MSE minim

(Script Proiect1_aflareGradOptim.m)

```
close all; clear; clc;
load('proj_fit_03.mat');
%% Date ID
x1 = id.X{1,1};
x2 = id.X{2,1};
N = length(x1);
y = id.Y; yflat = reshape(y,N*N,1); % yflat -> vector coloana, pentru a facilita calculul lui teta si implicit lui g_aprox_val
m = 1:30; % grad polinom
MSE_val_min = 1;
%% Date VAL
x1_val = val.X{1,1};
x2_val = val.X{2,1};
N_val = length(x1_val);
y_val = val.Y; y_val_flat = reshape(y_val,N_val*N_val,1);
%% Calculul matricei de regresori pentru identificare- Fi
for i=1:length(m)
    fi = [];
    for k1=1:N
        for k2=1:N
            fi = [fi;creaza_pol_n(m(i),k1,k2,x1,x2)];
        end
    end
```

```
        end
    %% Calcul vectorului de parametrii - teta
    teta = linsolve(fi,yflat);
    %% Calculul functie aproximate - g
    gflat = fi*teta;
    g = reshape(gflat,[N,N]); % transformam matricea coloana g in matrice patratica
    %% Calculul erorii medii patratice - ID
    e = sum((y-g).^2);
    MSE(i) = 1/(N^2)*sum(e);
    %% Calculul matricei de regresori pentru validare - fi_val
    fi_val = [];
    for k1=1:N_val
        for k2=1:N_val
            fi_val = [fi_val;creaza_pol_n(m(i),k1,k2,x1_val,x2_val)];
        end
    end
    %% Calculul functiei aproximate folosind vectorul teta din datele de identificare - g_val
    g_val_flat = fi_val*teta;
    g_val = reshape(g_val_flat,[N_val,N_val]);
    %% Calculul erorii medii patratice - VAL
    e_val = sum((y_val-g_val).^2);
    MSE_val(i) = 1/(N_val^2)*sum(e_val);
    clear fi_val; clear fi;
end
% Vector creat pentru a compara MSE-ul
Valori_MSE = [MSE;MSE_val]';

% Aflarea gradului m pentru care MSE este minim
for i=1:length(MSE_val)
```

```
    if(MSE_val(i) < MSE_val_min)
        MSE_val_min = MSE_val(i);
        index = i;
    end
end
fprintf('MSE_val(min) = %d\n',min(MSE_val));
fprintf('m = %d\n',index);

%% Plotam graficele erorilor medii patratice pentru datele de ID si VAL -> alegem gradum m minim
figure; plot(m,MSE); hold on; plot(m,MSE_val,'r'); hold on; plot(index,MSE_val_min,'g*'); xlabel('m'); ylabel('MSE');
title('Grafic MSE'); legend('ID','VAL');
```

5.3. Script plot-uri diferite grade

Este bazat pe aceleasi princii pe calcul ca scriptul de la 5.2. cu observatia ca acesta nu parcurge toate gradurile m, ci doar un grad m dat pentru a facilita calcul MSE-ului si plot-ul datelor dorite. (Script Proiect1_PlotGradOptim.m)

```
close all; clear; clc;
load('proj_fit_03.mat');
%% Date ID
x1 = id.X{1,1};
x2 = id.X{2,1};
N = length(x1);
y = id.Y; yflat = reshape(y,N*N,1);
mesh(x1,x2,y); title('y_I_d'); xlabel('x1'); ylabel('x2'); zlabel('y');
m = 25;
%% Date VAL
x1_val = val.X{1,1};
x2_val = val.X{2,1};
N_val = length(x1_val);
y_val = val.Y; y_val_flat = reshape(y_val,N_val*N_val,1);
figure; mesh(x1_val,x2_val,y_val); title('y_V_a_l'); xlabel('x1'); ylabel('x2'); zlabel('y');

%% Calculul vectorului teta pentru un m dat
fi = [];
for k1=1:N
    for k2=1:N
        fi = [fi;creaza_pol_n(m,k1,k2,x1,x2)];
    end
end
teta = linsolve(fi,yflat);
gflat = fi*teta;
g = reshape(gflat,[N,N]);
e = sum((y-g).^2);
MSE = 1/(N^2)*sum(e);

figure; mesh(x1,x2,g); title(['g_I_D_-_m_=_',num2str(m),'    MSE=',num2str(MSE)]);
xlabel('x1'); ylabel('x2'); zlabel('y');

%% Calcul functiei g_val pentru un m folosind vectorul teta
fi_val = [];
for k1=1:N_val
    for k2=1:N_val
        fi_val = [fi_val;creaza_pol_n(m,k1,k2,x1_val,x2_val)];
    end
end

g_val_flat = fi_val*teta;
g_val = reshape(g_val_flat,[N_val,N_val]);

e_val = sum((y_val-g_val).^2);
MSE_val = 1/(N_val^2)*sum(e_val);
figure; mesh(x1_val,x2_val,g_val); title(['g_V_A_L_-_m_=_',num2str(m),'    MSE=',num2str(MSE_val)]);
xlabel('x1'); ylabel('x2'); zlabel('y');
```

Partea 2. ARX neliniar

1. Introducere

Se considera sistemul dinamic prezentat in Fig.1 avand o intrare si o iesire. Se va dezvolta un model de tip cutie neagra pentru acest sistem, folosind o structura ARX neliniara de tip polinomial.

Fig1. Model sistem.

Pentru dezvoltarea sistemului dinamic se da un set de date pentru care intrarea si iesirea sunt cunoscute, iar iesirea poate fi afectata de zgomot. Al doilea set de date este utilizat pentru validarea modelului creat.

Obiectivul nostru este crearea unei functii care genereaza un model ARX neliniar de tip polinomial, avand ordinele na, nb, intarzierea nk si gradul m configurabil. De asemenea, se va folosi regresia liniara pentru a identifica parametrii. Modelul generat trebuie sa fie aplicat in doua moduri:

- Predictie (cu un pas inainte), in care se folosesc valorile reale ale iesirilor intarziate ale sistemului;
- Simulare, in care iesirile anterioare ale sistemului nu sunt disponibile, iar noi va trebui sa le aproximam cu iesirele simulate precedent ale modelului insusi.

Totodata, dorim sa folosim algoritmul regresiei liniare pentru a determina cea mai mica eroare medie patratica (MSE) pe datele de validare.

2. Notiuni teoretice si explicatii implementare

2.1. Structura NARX

Structura medelului ARX neliniar este:

$\tilde{y}(k) = p(y(k-1),\ldots,y(k-na),u(k-nk),u(k-nk-1),\ldots,u(k-nk-nb+1))$

, unde na, nb sunt ordinele sistemului si nk este intarzierea aplicata pe intrare, iar p este un polinom de grad m in aceste variabile.

2.2. Functia de generare a modelului NARX

Din structura medelului NARX se identifica iesirile si intrarile intarziate cu na si nb, pe baza carora se va crea modelul NARX de tip polinomial. Gradul polinomului m este configurabil.

Un exemplu de polinom cu na = nb = nk = 1 si m = 2 este:

$y(k) = ay(k-1) + bu(k-1) + cy(k-1)^2 + vu(k-1)^2 + wu(k-1)y(k-1) + z$

, in care a,b,c,v,w,z sunt parametrii modelului.

Se observa ca modelul contine intrari si iesire intarziate la puteri > 1 si produse intre acestea ceea ce face ca modelul sa fie neliniar.

2.3. Implementarea functiei de regresori

Algoritmul de implementare al functiei este bazat pe identificarea intrarilor si iesirilor intarziate cu na, respective nb. Acestea se vor trece intr-un vector de variabile. Se alege gradul m al polinomului care va determina puterea maxima a variabilelor de intrare si iesire.

Plecand de la dimensiunea minina de 2 a vectorului de variabile se vor face combinatii de n luate cate k, unde n este vectorul variabilelor, iar k este initial 2, deoarece dorim combinatii intre cel putin doua variabile, k fiind incrementat la fiecare pas, ajungand la dimensiunea maxima a vectorului de variabile. Folosind functia nchoosek() din MATLAB rezulta o matrice care are pe fiecare linie toate combinatiile posibile ale variabilelor luate cate k. Se impun doua variabile de lucru, L_col si L_lin, care reprezinta lungimea coloanei si a liniei matricii C, C $\in R^{L\_lin \, x \, L\_col}$. Pe baza acestora se realizeaza conditiile de executabilitate a functiei. In functie de aceste conditii, se alege un vector "de puteri" care are lungimea = L_lin si valoarea 1 pe fiecare pozitie. La fiecare pas se incrementeaza prima pozitie cu 1 pana suma elementelor din vector <= grad m. La depasirea acestei conditii se incrementeaza pozitia, se reinitializeaza vectorul. Pentru a evita reinitializarea cu vectorul unitate, se verifica daca pozitia este > 1, ceea ce conduce la incrementarea cu 1 a valorii de pe pozitia curenta. Functionarea acestei functii este evidentiata in scriptul Proiect2_verifica_regresori.m atasat fisierului ZIP in care se observa toti regresorii functiei.

2.4. Matricea de regresori Φ

2.4.1. Predictie

Pentru fiecare linie din matricea Φ se aplica functia de generare a polinomului pentru iesirile si intrarile intarziate cu na si nb la pasul i=1+na+nb:N, unde i este folosit pentru a parcurge vectorul intrari si a iesiri, Φ $\in R^{N \, x \, n}$, n = numar regresori, iar N este numarul de esatioane ale intrari sau iesiri.

2.5. Vectorul de parametrii θ

Calculul acestuia se face rezolvand sistemul de ecuatii liniare cu ajutorul functiei linsolve() din MATLAB.

De retinut ca vectorul θ se calculeaza o singura data, pe datele de identificare in cazul predictiei pentru cel mai bun MSE de pe datele de validare.

2.6. Functia aproximata ỹ

2.6.1. Predictie

Calculul iesirii aproximate se face prin produsul celor doua matrici: matricea de regresori Φ calculata anterior si vectorul de parametrii θ.

2.6.2. Simulare

Se alege un vector initial de iesiri cu na+nb zerouri. Se calculeaza produsul dintre prima linie a matricii de regresori si vectorul θ calculat in predictie rezultand un scalar. Astfel, la fiecare iteratie se va calcula produsul celor doi vectori rezultand un scalar care va fi concatenat cu iesirile simulate anterior. La final va rezulta y-ul simulat. De exemplu prima valoare simulate va fi:

$$\mathrm{y}(1) = [y(1-1), \dots, y(1-na), u(1-1), \dots, u(1-nb)]\theta$$

$$\dots$$

$$\mathrm{y}(\mathrm{N}) = [y(N-1), \dots, y(N-na), u(N-1), \dots, u(N-nb)]\theta$$

Toate aceste calcule sunt valabile pentru un θ bine determinat in urma predictiei, cel care genereaza MSE-ul cel mai mic. Ordinele na,nb si gradul polinomului m sunt aceleasi ca cele care l-au determinat pe θ.

2.7. Eroarea medie patratica (MSE)

Pentru a verifica ca modelul este bine determinat se calculeaza eroare medie patratica dupa formula: $\mathrm{MSE} = \frac{1}{N}\sum_{k=1}^{N}(y - \tilde{y})^2$.

3. Interpretare rezultate

3.1. Tabelul erorii medii patratice (MSE)

Tabelul 1 reprezinta valorile MSE-ului pentru diferite na,nb si m.

index	m(grad polinom)	na	nb	MSE-identificare	MSE-validare
1	1	1	1	11.7112	8.1744
2	1	1	2	11.7054	8.1703
3	1	1	3	11.6995	8.1662
4	1	2	1	11.7054	8.1703
5	1	2	2	11.6995	8.1662
6	1	2	3	11.6937	8.1622
7	1	3	1	11.6995	8.1662
8	1	3	2	11.6937	8.1622
9	1	3	3	11.6879	8.1581
10	2	1	1	0.0343	0.0352
11	2	1	2	0.4822	0.3136
12	2	1	3	0.3085	0.2273
13	2	2	1	0.5116	0.3670
14	2	2	2	0.2892	0.1919
15	2	2	3	0.2739	0.4374
16	2	3	1	0.4745	17.413
17	2	3	2	0.2707	13.737
18	2	3	3	0.0510	161.32
19	3	1	1	0.0334	0.03491
20	3	1	2	0.0297	2.2649
21	3	1	3	0.0288	20.659
22	3	2	1	3.4332e-04	27228.73
23	3	2	2	1.4499e-04	3069.85
24	3	2	3	6.0346e-05	1514.44
25	3	3	1	3.7081e-04	2201.88
26	3	3	2	1.8617e-05	94176.03
27	3	3	3	3.2331e-06	483115.39

Tabel 1. Valori MSE

3.2. Plot-uri Predictie si Simulare

3.2.1. Grafice predictie pe date ID si VAL

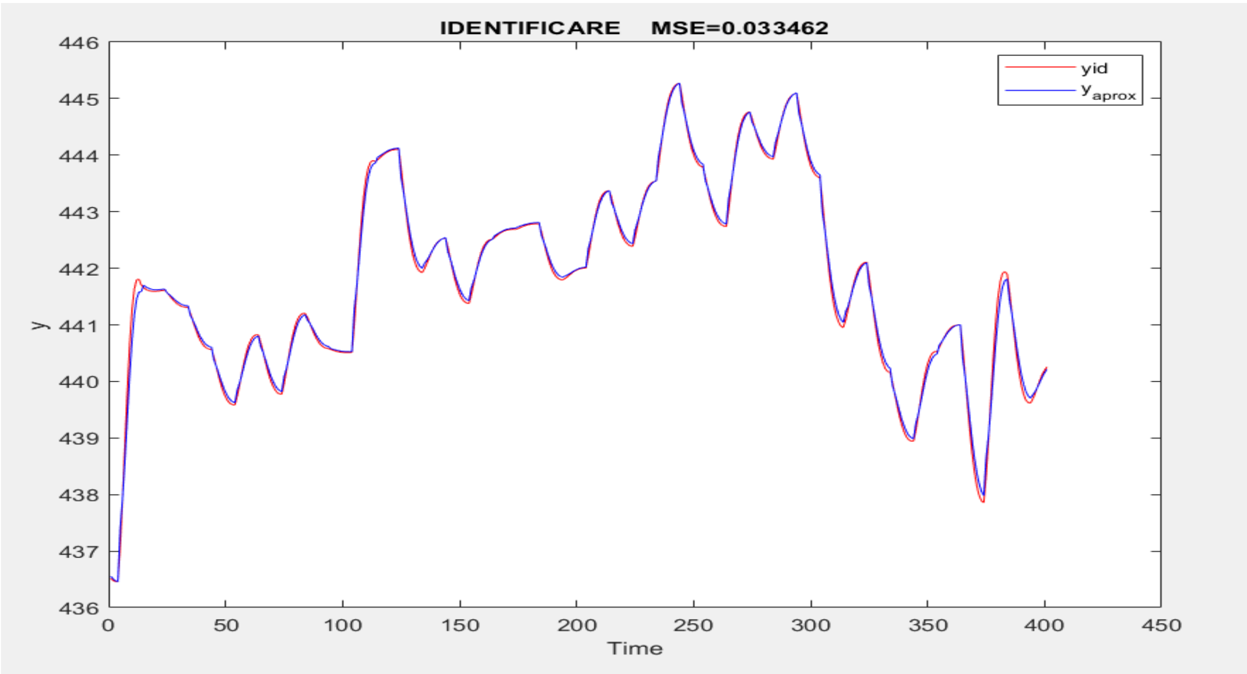

Fig2. Grafic predictie ID

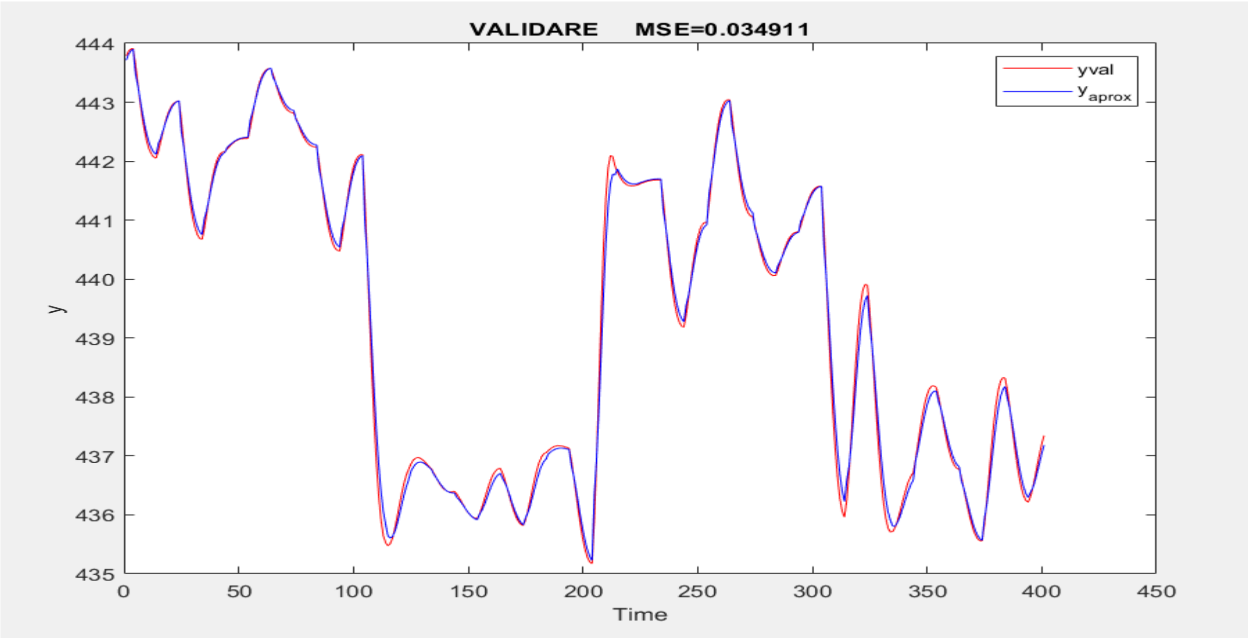

Fig3. Grafic predictie VAL

3.2.2. Grafice simulare pe date ID si VAL

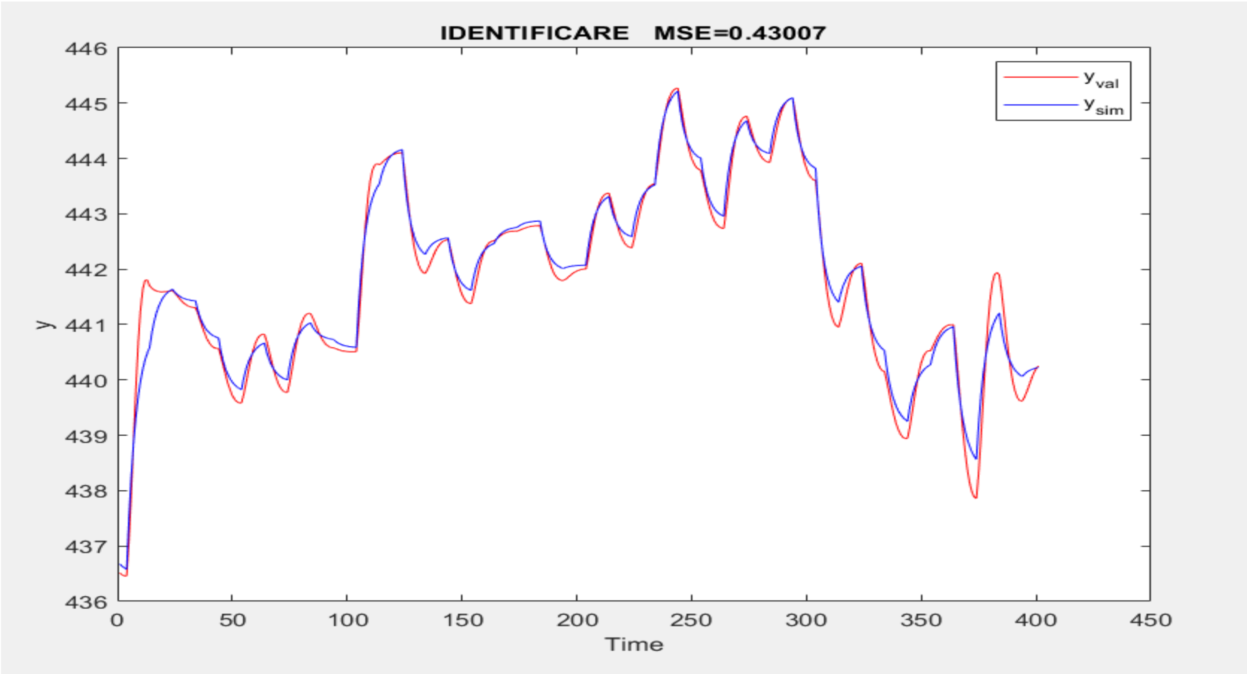

Fig4. Grafic simulare ID

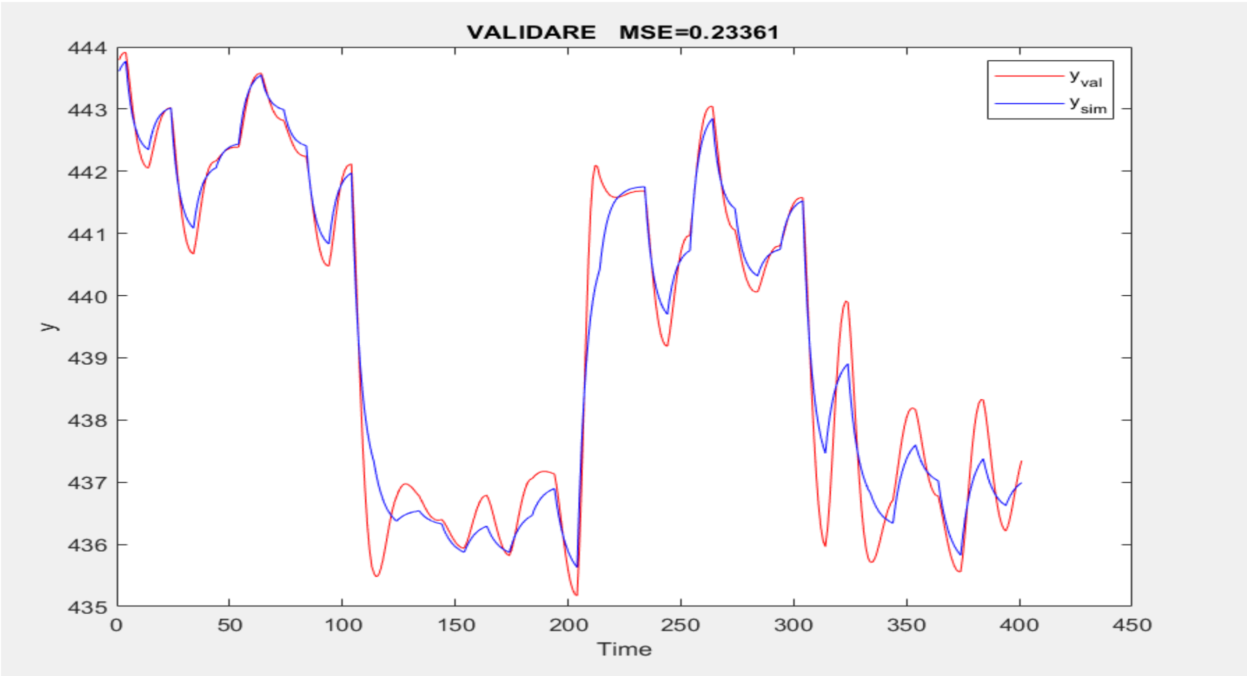

Fig5. Grafic simulare VAL

4. Discutie

Pe baza Tabelului 1 am descoperit MSE-ul minim = 0.03491 pentru na = 1, nb = 1 si m =3, index 19. Regresorii functiei sunt [1, y1, u1, y1^2, u1^2, y1^3, u1^3, u1*y1, u1*y1^2, u1^2*y1], unde y1 = y(i-1) si u1 = u(i-1).

Fenomenul de supraantrenare se ragaseste pentru datele din Tabelul 1 corespunzator indecsilor 18 si >= 21. Se observa ca valorile de pe identificare ale MSE-ului sunt cu mult mai mici decat cele de pe validare. Acesta fenomen este caracterizat prin modelarea zgomotului de la iesire care influenteaza calculul vectorului de parametrii θ responsabil pentru restul calculelor, adica regresorii aproximeaza foarte bine zgomotul de pe identificare.

Predictie:

In Fig.2 si Fig.3 sunt reprezentate grafic iesire y de pe datele de identificare si validare peste care este suprapusa iesirea aproximata ỹ. Se observa ca, functia de generare a regresorilor reuseste sa identifice vectorul de parametrii θ care sa aproximeze cat mai bine functia de iesire reala.

Simulare:

In Fig.4 si Fig.5 sun reprezentate grafic iesirile y de pe datele de identificare si validare peste care este suprapusa iesirea simulata calculata pe baza iesirilor aproximate ỹ simulate anterior. Se observa ca, graficele nu se suprapun la fel de bine ca in cazul predictiei, chiar daca, in cazul validarii, valoarea MSE-ului este mai mica decat in cazul predictiei. Acest lucru este datorat erorilor de calcul care se fac la fiecare iteratie pe datele de simulare. Nestiind datele reale se foloseste vectorul de parametrii θ care incearca sa aproximeze cat mai bine iesirea, dar acesta deja contine mici erori de calcul de la predictie care sunt apoi adaugate la calculul iesirilor simulate.

5. Script MATLAB

5.1. Functia de generare regresori arx

```
function [fi_l] = Proiect2_creeaza_reg(u,y,na,nb,m,index)
    y_aux = []; u_aux = [];
    % iesirile si intrarile intarziate cu na, nb
    for k1=1:na
        y_aux = [y_aux y(index-k1)];
    end
    for k2=1:nb
        u_aux = [u_aux u(index-k2)];
    end
    % vectorul de intrari si iesiri intarziate
    var = [u_aux y_aux];
    grad = m; dim_var = length(var); reg = 1;

    for k=2:dim_var
        C = nchoosek(var,k);
        L_col = length(C(:,1));
        L_lin = length(C(1,:));
        q = 2; % puterea pentru combinatiile de variabile care inmultite dau gradul <= gradul impus
        v = ones(1,L_lin); % vectorul de puteri
        poz = 1; % pozitia in vectorul de puteri

        if(L_lin <= grad)
            if(L_lin == dim_var)
                % toate puterile
                for i=1:grad
                    reg = [reg C(1,:).^i];
                end
            end
            % Combinatii cu aceleasi puteri
            if(L_lin <= grad/L_lin)
                while(L_lin*q <= grad)
                    C_aux = C;
                    for i=1:L_lin
                        C_aux(:,i) = C(:,i).^q; % ridicam toate coloanele la puterea q
                    end
                    for i=1:L_col
                        reg = [reg prod(C_aux(i,:))]; % inmultim toate liniile
                    end
                    q = q+1;
                end
            end
```

```
            % Combinatii puteri diferite
            while(poz+length(v)-1 <= grad && poz <= L_lin)
                v_aux = v;
                if(poz > 1) % sa nu luam iara vectorul ones
                    v_aux(poz) = v_aux(poz)+1;
                end
                while(sum(v_aux) <= grad)
                    C_aux = C;
                    for j=1:L_col
                        % ridicam toata linia la vectorul de puteri v
                        C_aux(j,:) = C_aux(j,:).^v_aux;
                        reg = [reg prod(C_aux(j,:))];
                    end
                    v_aux(poz) = v_aux(poz)+1;
                end
                poz = poz+1;
            end
        end
    end
    fi_l = reg;
end
```

5.2. Script aflare na,nb si m (Proiect2_Aflare_Ordine_Grad.m)

```
close all; clear; clc;
load('iddata-12.mat');

%% Date de lucru
grad_maxim = 3; MSE_id = []; MSE_val = []; i=1;
MSE_val_min = 1;

%% Determinare model sistem pentru MSE minim
for m=1:grad_maxim
    for na=1:grad_maxim
        for nb=1:grad_maxim
            %% ID
            % Date initiale
            uid = id.u; yid = id.y;
            uid = [zeros(1,na+nb) uid']; % adaugam zerouri pentru a putea parcurge intarzirile la inceput ale vectorilor u si y
            yid = [zeros(1,na+nb) yid'];
            N = length(yid);
            % Calculul matricii fi
            for j=1+na+nb:N
                fi(j,:) = Proiect2_creeaza_reg(uid,yid,na,nb,m,j);
            end
```

```
            % Calculul vectorului de parametrii prin rezolvarea sistemului liniar
            teta = linsolve(fi,yid');
            % Calculul iesiri aproximate
            y_aprox = fi*teta;
            % Calculul erorii medii patratice
            MSE_id(i) = 1/N*sum((yid-y_aprox').^2);
            % Vectori pentru a retine ordinele sistemului in vectorul MSE
            Na(i) = na; Nb(i) = nb; M(i) = m;

            %% VAL
            % Date initiale
            uval = val.u; yval = val.y;
            uval = [zeros(1,na+nb), uval'];
            yval = [zeros(1,na+nb), yval'];
            N = length(yval);
            % Calculul matricii fi
            for j=1+na+nb:N
                fi_val(j,:) = Proiect2_creeaza_reg(uval,yval,na,nb,m,j);
            end
            % Calculul iesiri aproximate
            y_aprox_val = fi_val*teta;
            % Calculul erorii medii patratice
            MSE_val(i) = 1/N*sum((yval-y_aprox_val').^2);
            % Aflarea ordinelor sistemului na si nb si gradului m pentru cel mai mic MSE
            if(MSE_val(i) < MSE_val_min)
                MSE_val_min = MSE_val(i);
                na_min_val = na;
                nb_min_val = nb;
                m_min_val = m;
            end
            % Index de retinere a pozitiei vectorului MSE
            i = i+1;
            clear fi fi_val;
        end
    end
end
% Vectorul asociat MSE-ului si ordinelor sistemului
Ordin = [MSE_id; MSE_val; Na; Nb; M]';
fprintf('na_min=%d, nb_min=%d, m_min=%d\n',na_min_val,nb_min_val,m_min_val);
```

5.3. Script Predictie si Simulare (Proiect2_Predictie_Simulare.m)

```
close all; clear; clc;
load('iddata-12.mat');

%% Predictie
    %% ID
    uid = id.u; yid = id.y;
    na=1; nb=1; m = 3; % determinate in scriptului Proiect2_Aflare_Ordin_Grad
    uid = [zeros(1,na+nb) uid'];
    yid = [zeros(1,na+nb) yid'];

    N = length(uid); Ts = id.Ts;
    % Alegem un inverval mai mic pentru a plota datele si a vedea mai bine graficele
    interval = 400:800;

    for i=1+na+nb:N
        fi(i,:) = Proiect2_creeaza_reg(uid,yid,na,nb,m,i);
    end
    teta = linsolve(fi,yid');
    y_aprox = fi*teta;

    MSE = 1/N*sum((yid-y_aprox').^2);
    plot(yid(interval),'r'); hold on; plot(y_aprox(interval),'b');
    xlabel('Time'); ylabel('y');
    legend('yid','y_a_p_r_o_x'); title(['IDENTIFICARE    MSE=',num2str(MSE)]);

    %% VAL

    uval = val.u; yval = val.y;
    uval = [zeros(1,na+nb), uval'];
```

```
    uval = [zeros(1,na+nb), uval'];
    yval = [zeros(1,na+nb), yval'];
    N = length(uval);

    for i=1+na+nb:N
        fi_val(i,:) = Proiect2_creeaza_reg(uval,yval,na,nb,m,i);
    end
    y_aprox_val = fi_val*teta;
    MSE_val = 1/N*sum((yval-y_aprox_val').^2);
    figure; plot(yval(interval),'r'); hold on; plot(y_aprox_val(interval),'b');
    xlabel('Time'); ylabel('y');
    legend('yval','y_a_p_r_o_x'); title(['VALIDARE    MSE=',num2str(MSE_val)]);
```

```
%% Simulare
    %% ID
    u_sim_id = uid;
    y_sim_id = zeros(1,na+nb);
    N = length(u_sim_id);

    for i=1+na+nb:N
        % la fiecare iteratie se calculeaza y(i) cu valorile aproximate anterior
        y_l = Proiect2_creeaza_reg(u_sim_id,y_sim_id,na,nb,m,i);
        % se actualizeaza vectorul de iesire simulat
        y_sim_id = [y_sim_id y_l*teta];
    end

    MSE_sim_id = 1/N*sum((yid-y_sim_id).^2);
    figure; plot(yid(interval),'r'); hold on; plot(y_sim_id(interval),'b');
    xlabel('Time'); ylabel('y');
    legend('y_v_a_l','y_s_i_m'); title(['IDENTIFICARE   MSE=',num2str(MSE_sim_id)]);

    %% VAL
    u_sim_val = uval;
    y_sim_val = zeros(1,na+nb); % vector initial de valori
    N = length(u_sim_val);

    for i=1+na+nb:N
        y_l = Proiect2_creeaza_reg(u_sim_val,y_sim_val,na,nb,m,i);
        y_sim_val = [y_sim_val y_l*teta];
    end

    MSE_sim_val = 1/N*sum((yval-y_sim_val).^2);
    figure; plot(yval(interval),'r'); hold on; plot(y_sim_val(interval),'b');
    xlabel('Time'); ylabel('y');
    legend('y_v_a_l','y_s_i_m'); title(['VALIDARE   MSE=',num2str(MSE_sim_val)]);
```