

Generarea unui model ARX neliniar

Mocan Paula-Maria

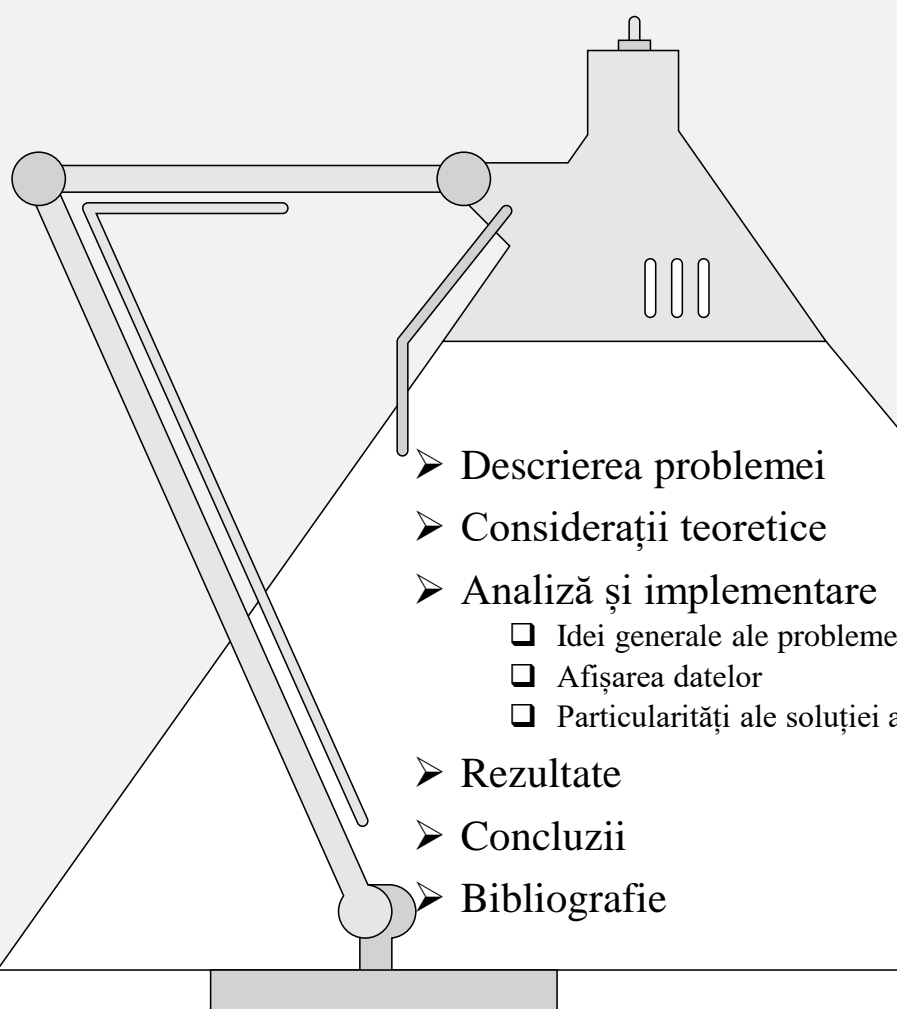
Pilug Elisei

Țicală Andreea-Irina

Pidx: 12

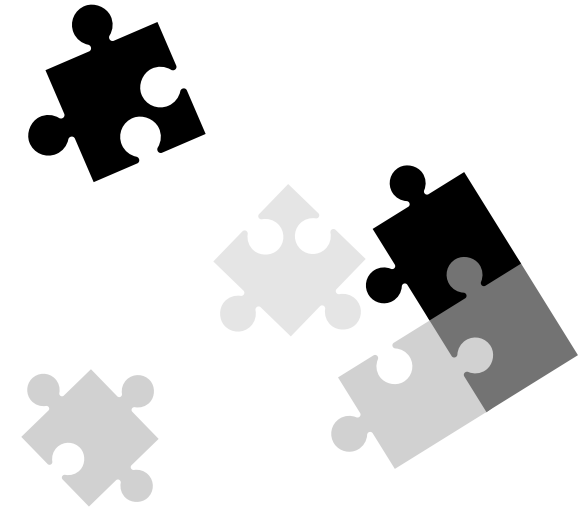
Grupe: 30134, 30136, 30135

CUPRINS

- 
- Descrierea problemei
 - Considerații teoretice
 - Analiză și implementare
 - ❑ Idei generale ale problemei
 - ❑ Afișarea datelor
 - ❑ Particularități ale soluției alese
 - Rezultate
 - Concluzii
 - Bibliografie

Descrierea problemei

- Generarea un model ARX neliniar de tip polinomial, cu ordinele n_a și n_b , respectiv gradul m configurabile.
- Scopul: găsirea valorilor pentru parametrii θ astfel încât aproximarea să fie cât mai apropiată de Y pentru orice grad m , eroarea să fie cât mai mică.



Considerații teoretice

- ARX neliniar generalizează la orice dependență neliniară:
$$y(k) = g(y(k-1), y(k-2), \dots y(k-n_a), u(k-1), u(k-2), \dots u(k-n_b); \theta) + e(k)$$
- În cazul nostru, g este un polinom de gradul m în ieșirile și intrările precedente
- Formula generală este: $Y = \phi\theta$, unde ϕ – matrice de regresori, iar θ – coeficienții regresorilor.
- Eroarea medie pătratică e calculată cu formula $MSE = \frac{1}{N} \sum (y - g)^2$, unde g este aproximarea găsită.

Analiză și implementare

- 2 seturi de date – identificare și validare, fiecare cu o intrare și o ieșire;
- Implementarea matricii de regresori prin combinarea termenilor neliniari și polinomiali în funcție de ordinele și dependențele variabilelor de intrare și ieșire;
- Calculul vectorului de coeficienți folosind formula $\theta = \phi \setminus Y$
- Realizarea predicției folosind valori reale ale ieșirilor, apoi simularea utilizând ieșirile anterioare ale modelului.
- Calculul erorilor pentru predicție și simulare.

Afișarea datelor

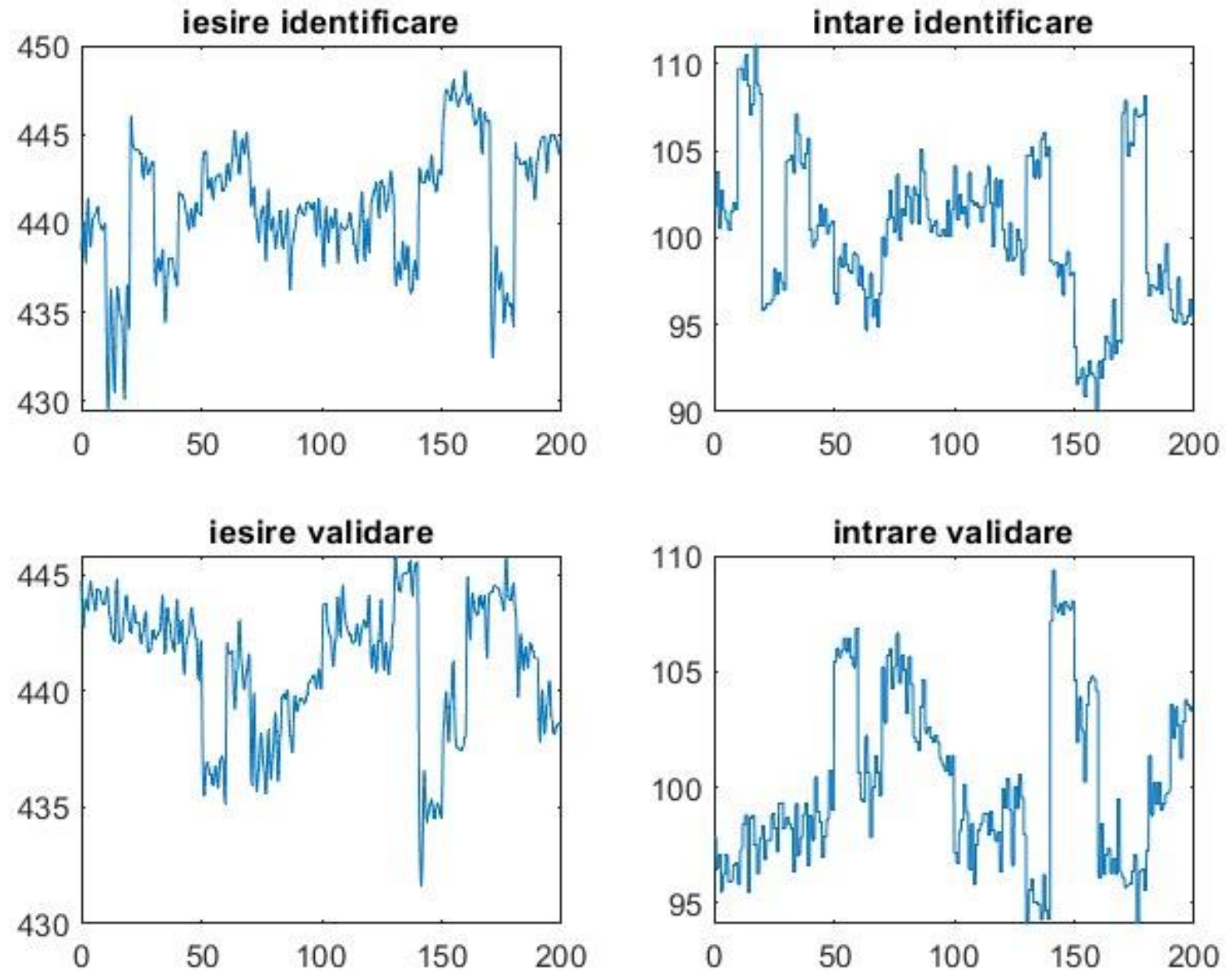


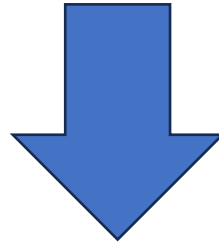
Figura 1:
Reprezentarea datelor
primate în Matlab

Particularități ale soluției noastre

- Calcul pe blocuri de matrici (coloane).
- Combinari de monoame în funcție de înmulțirea numerelor prime.
Folosind formula $C_{na+nb}^{m+na+nb}$.
- Folosirea unei matrici auxiliare pentru implementare care ne va ajuta la găsirea combinărilor următoare în funcție de proprietățile numerelor prime.

$$\begin{bmatrix} y(k-1) & y(k-2) & \dots & y(k-na) & u(k-1) & u(k-2) & \dots & u(k-nb) \\ y(k-1)^2 & y(k-2)^2 & \dots & y(k-na)^2 & u(k-1)^2 & u(k-2)^2 & \dots & u(k-nb)^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ y(k-1)^m & y(k-2)^m & \dots & y(k-na)^m & u(k-1)^m & u(k-2)^m & \dots & u(k-nb)^m \end{bmatrix}$$

Structura inițială a matricei Φ vs
matricea auxiliara



$$\begin{bmatrix} 1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 \\ 2 & 4 & & k_{na-1} & k_{na} & k_{na+1} & & k_{na+nb} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 2 & 2 & \dots & 2 & 2 & 2 & \dots & 2 \\ 2 & 4 & & k_{na-1} & k_{na} & k_{na+1} & & k_{na+nb} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ m & m & \dots & m & m & m & \dots & m \\ 2 & 4 & & k_{na-1} & k_{na} & k_{na+1} & & k_{na+nb} \end{bmatrix}$$

Exemplu: Pentru $n_a=n_b=2$ și gradul $m=2$

phi_final

2000x15 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	-438.5400	0	101.7373	0	1.9232e+05	0	1.0350e+04	0	0	-4.4616e+04	0	0	0	0
3	1	-438.7376	-438.5400	101.7373	101.7373	1.9249e+05	1.9232e+05	1.0350e+04	1.0350e+04	1.9240e+05	-4.4636e+04	-4.4636e+04	-4.4616e+04	-4.4616e+04	1.0350e+04
4	1	-438.9876	-438.7376	101.7373	101.7373	1.9271e+05	1.9249e+05	1.0350e+04	1.0350e+04	1.9260e+05	-4.4661e+04	-4.4661e+04	-4.4636e+04	-4.4636e+04	1.0350e+04
5	1	-439.2618	-438.9876	101.7373	101.7373	1.9295e+05	1.9271e+05	1.0350e+04	1.0350e+04	1.9283e+05	-4.4689e+04	-4.4689e+04	-4.4661e+04	-4.4661e+04	1.0350e+04
6	1	-439.5292	-439.2618	101.7373	101.7373	1.9319e+05	1.9295e+05	1.0350e+04	1.0350e+04	1.9307e+05	-4.4717e+04	-4.4717e+04	-4.4689e+04	-4.4689e+04	1.0350e+04
7	1	-439.7524	-439.5292	101.7373	101.7373	1.9338e+05	1.9319e+05	1.0350e+04	1.0350e+04	1.9328e+05	-4.4739e+04	-4.4739e+04	-4.4717e+04	-4.4717e+04	1.0350e+04

Matricea Φ care va avea în corespondență o matrice auxiliara numita matrice_grad formata din 2 linii, prima reprezentand gradul $y(i)$ și $u(i)$, a doua al i -lea numar prim respectiv combinările aferente dupa cum urmeaza:

- dacă elementele de pe a doua linie sunt prime se vor ridica la gradul corespunzator si se vor înmulți
- dacă doar unul dintre elementele celei de-a doua linii este prim, acesta se va ridica la gradul corespunzator și se va înmulți cu cel neprim
- dacă niciun element nu este prim se vor înmulți între ele

În condițiile în care vom face parcurgerea cu 2 variabile.

matrice_grad =

0	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2
0	2	3	5	7	2	3	5	7	6	10	14	15	21	35	

Predicție și simulare

Vom aplica același principiu și pentru simularea modelului nostru în Non Linear ARX. Doar că pentru fiecare element al șirului aproximat pe care dorim să îl obținem vom construi o linie Y (pe care am construit-o similar Φ) pe care o vom înmulți cu θ obținut anterior din formula $\theta = (\Phi^T \Phi)^{-1} \Phi^T y$ pentru a ne rezulta fiecare element în parte din datele approximate.

Rezultate predicție

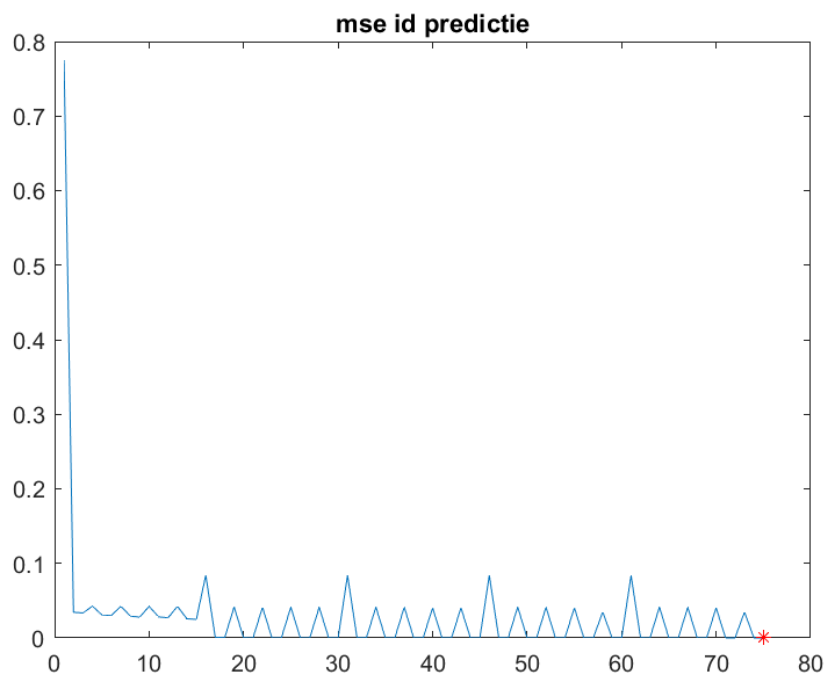


Figura 2:
Șir de erori pentru predicție, identificare

În urma calculul erorilor pentru predicție la identificare, cea mai mică eroare a rezultat pentru $n_a = n_b = 5$, iar gradul polinomului $m = 3$.

$$\text{MSE} = 2.6784\text{e-}06$$

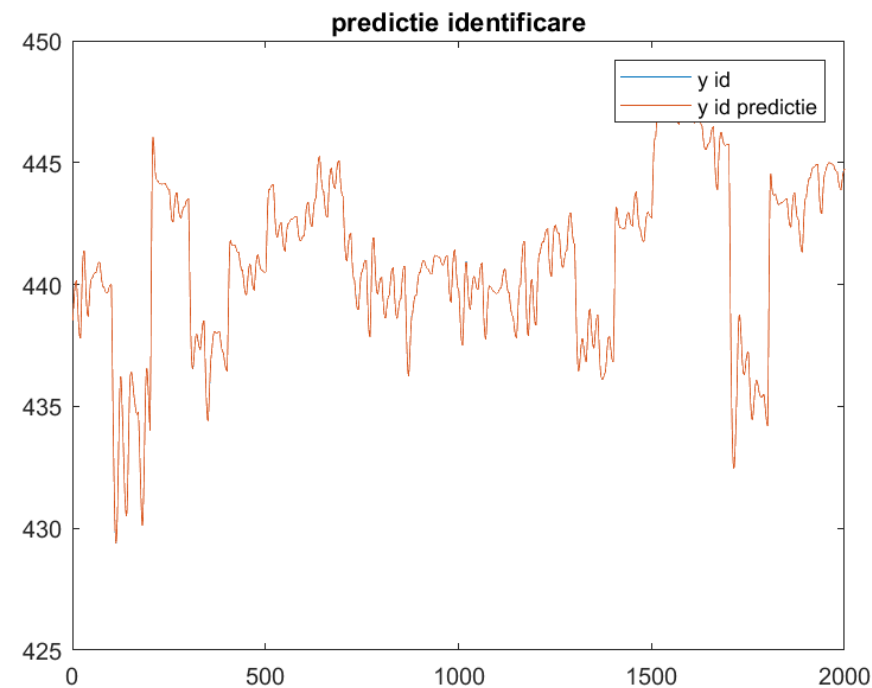
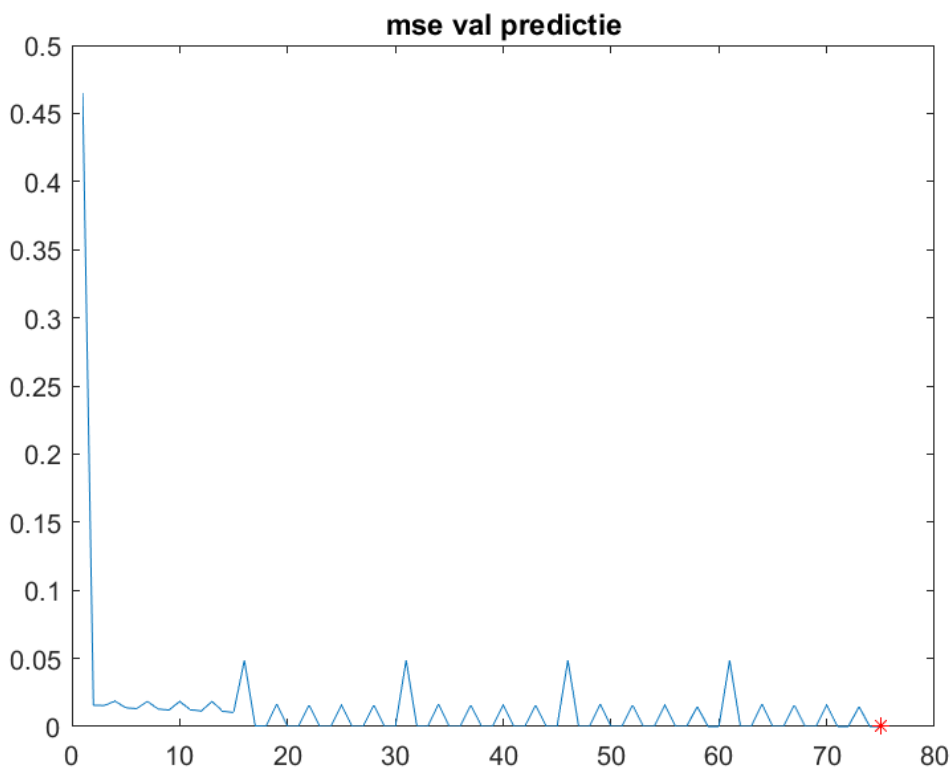


Figura 3: Predicție pentru $n_a = n_b = 5$, $m = 3$

Rezultate predicție (2)



Pentru predicția
pe setul de
validare, au reieșit
aceiași parametri
în urma calculului
erorii, $n_a = n_b = 5$,
 $m = 3$.
 $MSE = 1.4972e-06$

Figura 4:
Șir de erori pentru predicție, validare

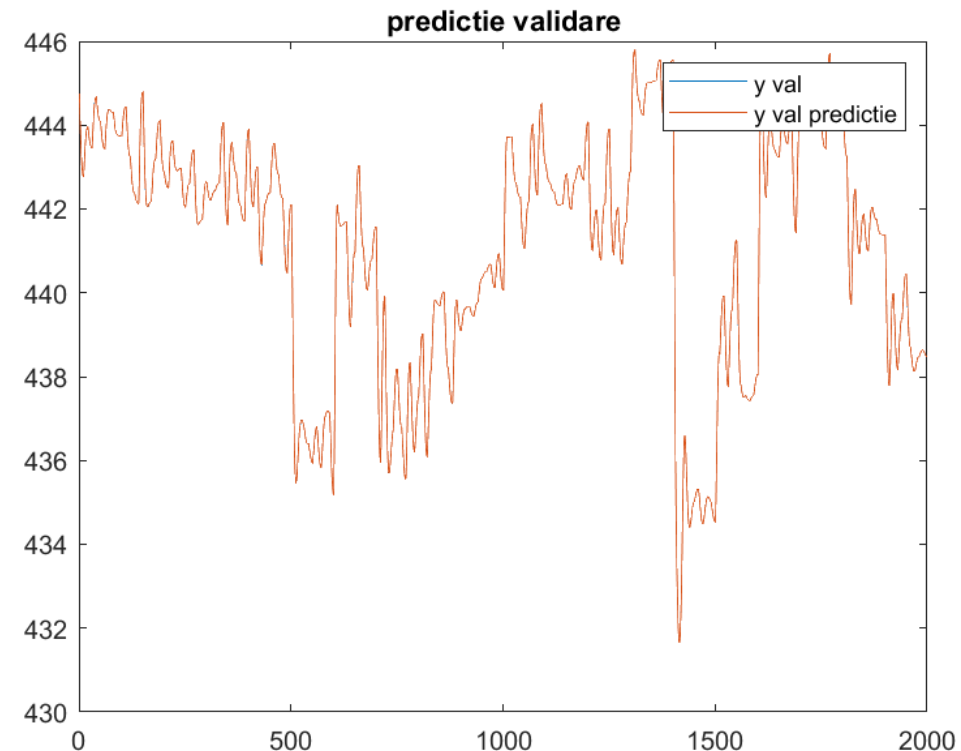


Figura 5: Predicție pentru $n_a=n_b=5$, $m=3$

Rezultate simulare

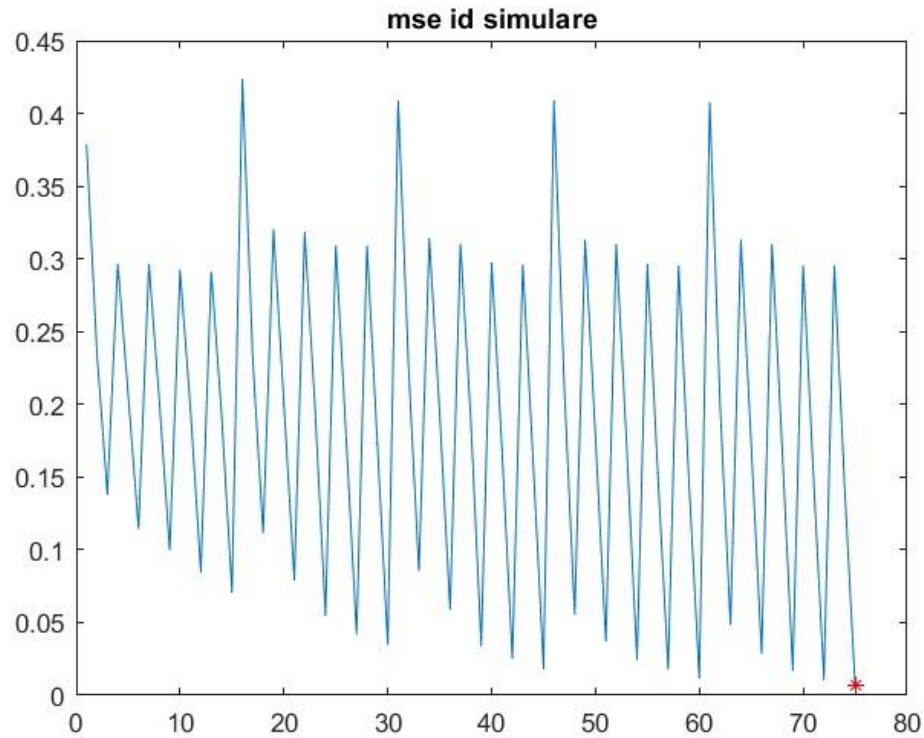


Figura 4 : Șir de erori pentru predicție, identificare
Pentru $n_a=5$, $n_b=5$, $m=3$
MSE = 0.007

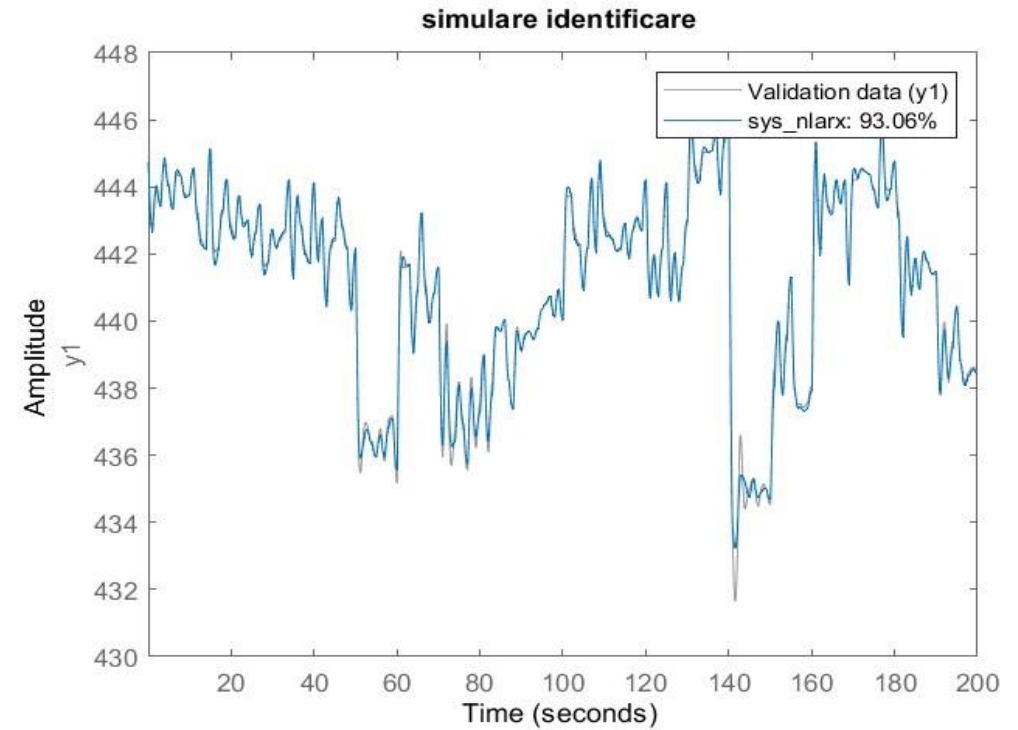


Figura 5: Rezultatele obtinute in urma simularii datelor
de identificare
Pentru $n_a=5$, $n_b=5$, $m=3$

Rezultate simulare (2)

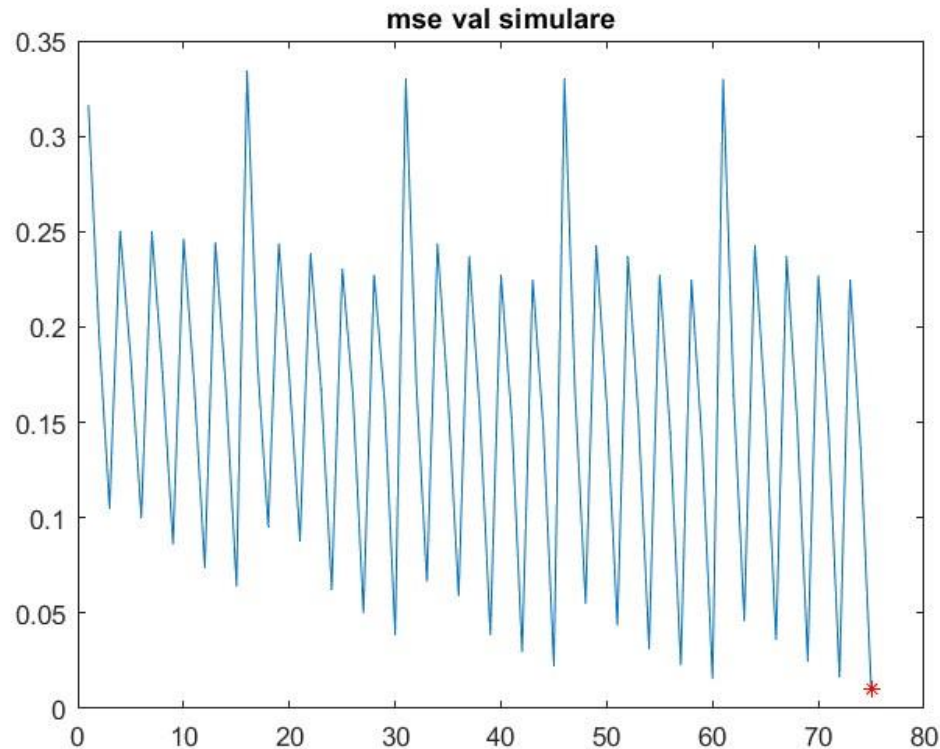


Figura 6: Șir de erori pentru simulare, validare
Pentru $n_a=5$, $n_b=5$, $m=3$
MSE = 0.01

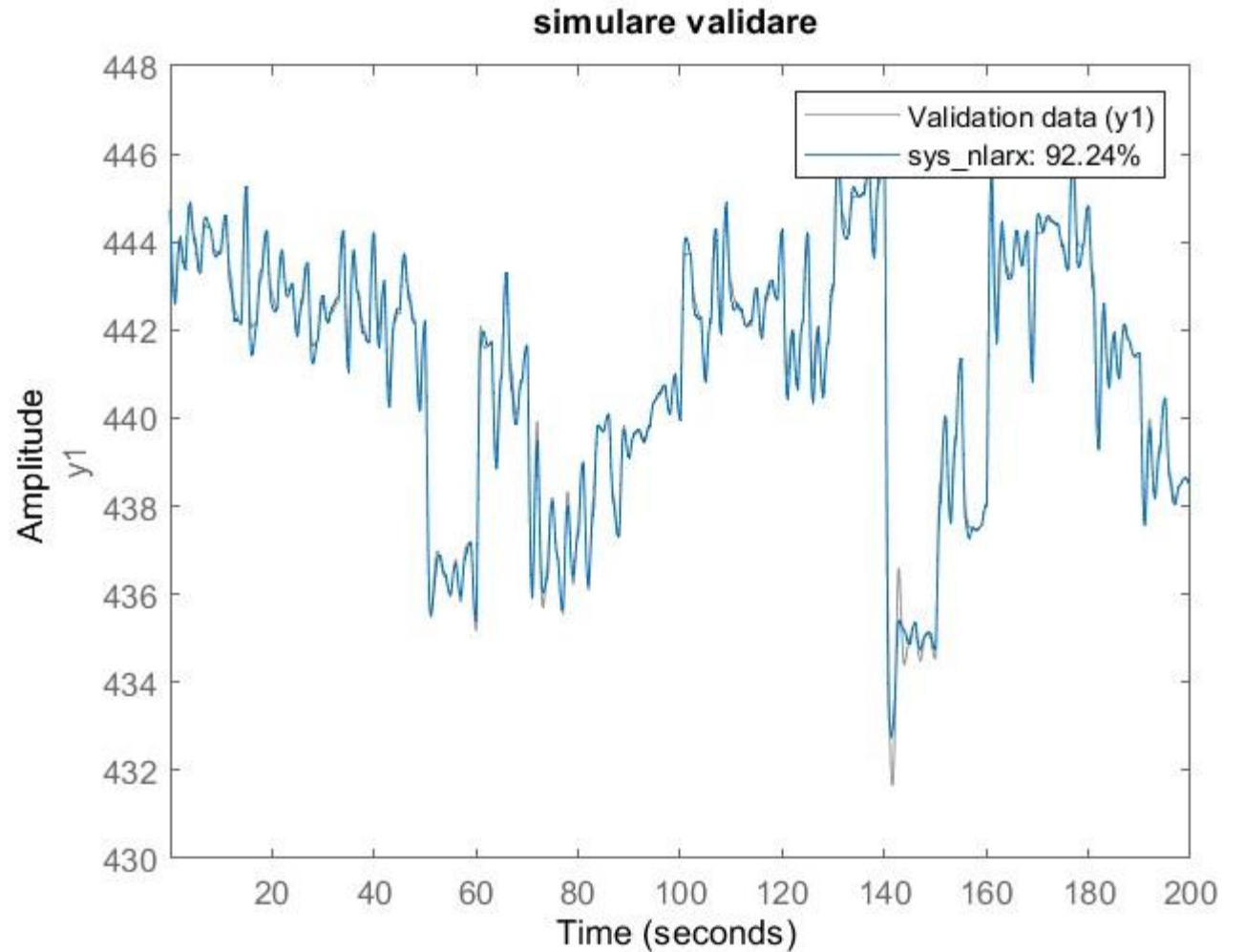


Figura 7: Rezultatele obtinute in urma simuarii datelor de validare
Pentru $n_a=5$, $n_b=5$, $m=3$

Concluzii

- În cazul predicției, modelul are eroare ușor mai mare pentru gradul minim al polinomului, $m = 1$, ajungând în subantrenare;
- Cele mai bune rezultate au fost obținute pentru $n_a=n_b=5$ și gradul $m=3$ pentru predicție.

Bibliografie

- Lucian Buşoniu
Identificarea sistemelor – Ingineria sistemelor, anul 3, Universitatea
Tehnica din Cluj-Napoca
Partea II – Baze matematice: Regresie liniară. Teoria
probabilităţilor şi statistică
Partea V – Metoda ARX
- Torsten Söderström, Petre Stoica
System Identification.
Prentice Hall, 2001.
- Multinomial theorem, Wikipedia, https://en.wikipedia.org/wiki/Multinomial_theorem ,
ultima accesare: 23/12/2023

Vă mulțumim pentru atenție !

Anexa

```
clear all
close all
clc

load('iddata-12')
y_id=id.y;
x_id=id.u;
t_id=id_array(:,1);
subplot(2, 2, 1);
plot(t_id, y_id);
title("iesire identificare")
subplot(2,2,2)
plot(t_id, x_id);
title("intare identificare")
y_val=val.y;
x_val=val.u;
t_val=val_array(:,1);
subplot(2, 2, 3);
plot(t_val, y_val);
title("iesire valide")
subplot(2,2,4)
plot(t_val, x_val);
title("intrare valide")
MSE_vector_val = [];
MSE_vector_id = [];

for na = 1:5
    for nb = 1:5
        for m = 1:3
            n=length(y_id);

            max=nchoosek(m+na+nb,
na+nb);
            phi_final=zeros(length(y_id),
max);

            phi=zeros(n, na+nb);

            matrice_grad=zeros(2, max);

            for i=1:n %pozitiile pe coloane
                k=i-1;
                for j=1:na %pozitiile pe linie
                    if k>0
                        phi(i,j)=-y_id(k);
                    end
                    k=k-1;
                end
            end

            for i=1:n %pozitiile pe coloane
                k=i-1;
                for j=(na+1):(nb+na)
                    %pozitiile pe linie

                    if k>0
                        phi(i,j)=x_id(k);
                    end
                    k=k-1;
                end
            end
        end
    end
end

phi_final(:, 1)=1; %am initializat prima coloana cu 1
aux=0;
numar_coloane_completate=1;
for i=1:m
    n=1; %nr elemente
    for j=1:(na+nb)
        a=1+j+aux;
        phi_final(:, a)=phi(:, j).^i; %initilizam urmatoarele coloane cu polinoame liniare
        matrice_grad(1, a)=i;
        matrice_grad(2, a)=nthprime(n);
        numar_coloane_completate=numar_coloane_completate+1; %a cata coloana e completata
        n=n+1;
    end
    aux=aux+(na+nb);
end
phi_final(:,end)=phi(:,1).*phi(:,2);
for i=2:max %combinatiile pe care le incercam
    for j=(i+1):max%numar_coloane_completate %pornind de la urmatorul
        if(matrice_grad(1,i)+matrice_grad(1,j)<=m && matrice_grad(1,i)>0 && matrice_grad(1,j)>0 && i~=j &&matrice_grad(2,
j)~=matrice_grad(2, i))
            phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);

            new_number=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j)^matrice_grad(1,j);
            contains = any(matrice_grad(2, :) == new_number);
            primi=isprime(matrice_grad(2,i));
            primj=isprime(matrice_grad(2,j));
            if(primi && primj)
                if(~contains)
                    phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);
                    matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
                    matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j)^matrice_grad(1,j);
                    numar_coloane_completate=numar_coloane_completate+1;
                end
            end
        end
    end
end
```

```

new_number=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j);
contains = any(matrice_grad(2, :) == new_number);
primi=isprime(matrice_grad(2,i));
primj=isprime(matrice_grad(2,j));
if(primi&&~primj)
if(~contains)
phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);
matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j);
numar_coloane_completate=numar_coloane_completate+1;
end
end

new_number=matrice_grad(2,i)*matrice_grad(2,j)^matrice_grad(1,j);
contains = any(matrice_grad(2, :) == new_number);
primi=isprime(matrice_grad(2,i));
primj=isprime(matrice_grad(2,j));
if(~primi&&primj)
if(~contains)
phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);
matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)*matrice_grad(2,j)^matrice_grad(1,j);
numar_coloane_completate=numar_coloane_completate+1;
end
end

new_number=matrice_grad(2,i)*matrice_grad(2,j);
contains = any(matrice_grad(2, :) == new_number);
primi=isprime(matrice_grad(2,i));
primj=isprime(matrice_grad(2,j));
if(~primi&&~primj)
if(~contains)
phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);
matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)*matrice_grad(2,j);
numar_coloane_completate=numar_coloane_completate+1;
end
end

end
end
end
teta=phi_final\y_id;
% y_hat_id_pred = phi_final*teta;

```

```

Y=zeros(1,max);
Y(max)=1;
for k=1:length(y_val)
i=1;
for k1=1:na
if(k-k1>0)
mat_de_1(i)=y_id(k-k1);
i=i+1;

else
mat_de_1(i)=0;
i=i+1;

end
end
for k1=1:nb
if(k-k1>0)
mat_de_1(i)=x_id(k-k1);
i=i+1;
else
mat_de_1(i)=0;
i=i+1;
end
end
% % %bucle in care am introdus y_val cu intarzieri si x_val cu intarzieri
% % %valori pe care le vom folosi in determinarea lui Y
i=i-1;
aux=0;
numar_coloane_completate=1;
matrice_grad=zeros(2, max);
n=1;

```

```

for ridicare_grad=1:m
    for fiecare_element=1:i
        Y(n)=mat_de_1(fiecare_element)^ridicare_grad;
        matrice_grad(1, n)=ridicare_grad;
        matrice_grad(2, n)=nthprime(n);
        n=n+1;
    end
    %am folosit acelasi principiu ca la construirea matricei phi
end
for i=1:(max-1) %numarul coloanei
    for j=(i+1):(max-1) %pornind de la urmatorul
        if(matrice_grad(1,i)+matrice_grad(1,j)<=m && matrice_grad(1,i)>0 && matrice_grad(1,j)>0 && i~=j &&matrice_grad(2, j)~=matrice_grad(2, i))

            new_number=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j)^matrice_grad(1,j);
            contains = any(matrice_grad(2, :) == new_number);
            primi=isprime(matrice_grad(2,i));
            primj=isprime(matrice_grad(2,j));
            if(primi && primj)
                if(~contains)
                    Y(numar_coloane_completate+1)=Y(i).*Y(j);
                    matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
                    matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j)^matrice_grad(1,j);
                    numar_coloane_completate=numar_coloane_completate+1;
                end
            end
        end

        new_number=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j);
        contains = any(matrice_grad(2, :) == new_number);
        primi=isprime(matrice_grad(2,i));
        primj=isprime(matrice_grad(2,j));
        if(primi&&~primj)
            if(~contains)
                Y(numar_coloane_completate+1)=Y(i).*Y(j);
                matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
                matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j);
                numar_coloane_completate=numar_coloane_completate+1;
            end
        end
    end
end

```

```

new_number=matrice_grad(2,i)*matrice_grad(2,j);
contains = any(matrice_grad(2, :) == new_number);
primi=isprime(matrice_grad(2,i));
primj=isprime(matrice_grad(2,j));
if(~primi&&~primj)
    if(~contains)
        Y(numar_coloane_completate+1)=Y(i).*Y(j);
        matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
        matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)*matrice_grad(2,j);
        numar_coloane_completate=numar_coloane_completate+1;
    end
end

end
end
end
y_hat_id_pred(k)=Y*teta;
end

        MSE_id=sum((y_hat_id_pred-y_id).^2)/length(y_id);
        MSE_vector_id = [MSE_vector_id, MSE_id];
    end
end
end

[xid,indxid] = min(MSE_vector_id,[],'all','linear');

for na = 1:5
    for nb = 1:5
        for m = 1:3
            n=length(y_val);

            max=nchoosek(m+na+nb, na+nb);
            phi_final=zeros(n, max);

            phi=zeros(n, na+nb);

            matrice_grad=zeros(2, max);

            for i=1:n    %pozitiile pe coloane
                k=i-1;
                for j=1:na    %pozitiile pe linie
                    if k>0
                        phi(i,j)=-y_val(k);
                        % k=k-1;
                    end
                    k=k-1;
                end
            end

            end
end
end

```

```

for i=1:n %pozitiile pe coloane
    k=i-1;
    for j=(na+1):(nb+na) %pozitiile pe linie
        if k>0
            phi(i,j)=x_val(k);
        end
        k=k-1;
    end
end
phi_final(:, 1)=1; %am initializat prima coloana cu 1
aux=0;
numar_coloane_completate=1;
for i=1:m
    n=1; %nr elemente
    for j=1:(na+nb)
        a=1+j+aux;
        phi_final(:, a)=phi(:, j).^i; %initalizez urmatoarele coloane cu polinoame liniare
        matrice_grad(1, a)=i;
        matrice_grad(2, a)=nthprime(n);
        numar_coloane_completate=numar_coloane_completate+1; %a cata coloana e completata
        n=n+1;

    end
    aux=aux+(na+nb);
end
phi_final(:,end)=phi(:,1).*phi(:,2);
for i=2:max %combinatiile pe care le incercam
    for j=(i+1):max%numar_coloane_completate %pornind de la urmatorul
        if(matrice_grad(1,i)+matrice_grad(1,j)<=m && matrice_grad(1,i)>0 && matrice_grad(1,j)>0 && i~=j &&matrice_grad(2,
j)~=matrice_grad(2, i))
            phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);

            new_number=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j)^matrice_grad(1,j);
            contains = any(matrice_grad(2, :) == new_number);
            primi=isprime(matrice_grad(2,i));
            primj=isprime(matrice_grad(2,j));
            if(primi && primj)
                if(~contains)
                    phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);
                    matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
                    matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j)^matrice_grad(1,j);
                    numar_coloane_completate=numar_coloane_completate+1;
                end
            end
        end
    end
end

```

```

new_number=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j);
contains = any(matrice_grad(2, :) == new_number);
primi=isprime(matrice_grad(2,i));
primj=isprime(matrice_grad(2,j));
if(primi&&~primj)
    if(~contains)
        phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);
        matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
    end
end

```

```

matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j);
numar_coloane_completate=numar_coloane_completate+1;
end
end

```

```

new_number=matrice_grad(2,i)*matrice_grad(2,j)^matrice_grad(1,j);
contains = any(matrice_grad(2, :) == new_number);
primi=isprime(matrice_grad(2,i));
primj=isprime(matrice_grad(2,j));
if(~primi&&primj)
    if(~contains)
        phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);
        matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
    end
end

```

```

matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)*matrice_grad(2,j)^matrice_grad(1,j);
numar_coloane_completate=numar_coloane_completate+1;
end
end

```

```

new_number=matrice_grad(2,i)*matrice_grad(2,j);
contains = any(matrice_grad(2, :) == new_number);
primi=isprime(matrice_grad(2,i));
primj=isprime(matrice_grad(2,j));
if(~primi&&~primj)
    if(~contains)
        phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);
        matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
        matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)*matrice_grad(2,j);
        numar_coloane_completate=numar_coloane_completate+1;
    end
end

```

```

end
end
end

```

```

teta = phi_final\y_val;
    %y_hat_val_pred = phi_final * teta;
    Y=zeros(1,max);
Y(max)=1;
for k=1:length(y_val)
i=1;
for k1=1:na
    if(k-k1>0)
        mat_de_1(i)=y_val(k-k1);
        i=i+1;

    else
        mat_de_1(i)=0;
        i=i+1;

    end
end
for k1=1:nb
    if(k-k1>0)
        mat_de_1(i)=x_val(k-k1);
        i=i+1;
    else
        mat_de_1(i)=0;
        i=i+1;
    end
end
% % %bucle in care am introdus y_val cu intarzieri si x_val cu intarzieri
% % %valori pe care le vom folosi in determinarea lui Y
i=i-1;
aux=0;
numar_coloane_completate=1;
matrice_grad=zeros(2, max);
n=1;
%mat_de_1
for ridicare_grad=1:m
    for fiecare_element=1:i
        Y(n)=mat_de_1(fiecare_element)^ridicare_grad;
        matrice_grad(1, n)=ridicare_grad;
        matrice_grad(2, n)=nthprime(n);
        n=n+1;
    end
    %am folosit acelasi principiu ca la construirea matricei phi
end
for i=1:(max-1) %numarul coloanei
    for j=(i+1):(max-1) %pornind de la urmatorul
        if(matrice_grad(1,i)+matrice_grad(1,j)<=m && matrice_grad(1,i)>0 && matrice_grad(1,j)>0 && i~=j &&matrice_grad(2,
j)~=matrice_grad(2, i))

```

```

new_number=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j)^matrice_grad(1,j);
    contains = any(matrice_grad(2, :) == new_number);
    primi=isprime(matrice_grad(2,i));
    primj=isprime(matrice_grad(2,j));
    if(primi && primj)
        if(~contains)
            Y(numar_coloane_completate+1)=Y(i).*Y(j);
            matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
            matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j)^matrice_grad(1,i);
            numar_coloane_completate=numar_coloane_completate+1;
        end
    end
    new_number=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j);
    contains = any(matrice_grad(2, :) == new_number);
    primi=isprime(matrice_grad(2,i));
    primj=isprime(matrice_grad(2,j));
    if(primi&&~primj)
        if(~contains)
            Y(numar_coloane_completate+1)=Y(i).*Y(j);
            matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
            matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j);
            numar_coloane_completate=numar_coloane_completate+1;
        end
    end
    new_number=matrice_grad(2,i)*matrice_grad(2,j);
    contains = any(matrice_grad(2, :) == new_number);
    primi=isprime(matrice_grad(2,i));
    primj=isprime(matrice_grad(2,j));
    if(~primi&&~primj)
        if(~contains)
            Y(numar_coloane_completate+1)=Y(i).*Y(j);
            matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
            matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)*matrice_grad(2,j);
            numar_coloane_completate=numar_coloane_completate+1;
        end
    end

    end
end
end
y_hat_val_pred(k)=Y*teta;
end
    MSE_val = sum((y_hat_val_pred-y_val).^2)/length(y_val);
    MSE_vector_val = [MSE_vector_val, MSE_val];
end
end
end

```

```
% am observat ca ultima combinatie e cea buna dpdv al erorii, deci afisam
% predictia pt na=nb=5, m=3
figure, plot(y_id), hold on, plot(y_hat_id_pred), legend('y id', 'y id predictie'), title('predictie identificare')
figure, plot(y_val), hold on, plot(y_hat_val_pred), legend('y val', 'y val predictie'), title('predictie validate')
```

```
[xval,indxval] = min(MSE_vector_val,[],'all','linear');
figure,plot(MSE_vector_id), hold on, plot(indxid,xid,'r*'), title('mse id predictie')
figure, plot(MSE_vector_val), hold on, plot(indxval,xval,'r*'), title('mse val predictie')
```

```
for na = 1:5
    for nb = 1:5
        for m = 1:3
            n=length(y_id);

            max=nchoosek(m+na+nb, na+nb);
            phi_final=zeros(length(y_id), max);

            phi=zeros(n, na+nb);

            matrice_grad=zeros(2, max);

            for i=1:n %pozitiile pe coloane
                k=i-1;
                for j=1:na %pozitiile pe linie
                    if k>0
                        phi(i,j)=-y_id(k);
                    end
                    k=k-1;
                end
            end

            for i=1:n %pozitiile pe coloane
                k=i-1;
                for j=(na+1):(nb+na) %pozitiile pe linie

                    if k>0
                        phi(i,j)=x_id(k);
                    end
                    k=k-1;
                end
            end
        end
    end
end
```

```
phi_final(:, 1)=1; %am initializat prima coloana cu 1
aux=0;
numar_coloane_completate=1;
for i=1:m
    n=1; %nr elemente
    for j=1:(na+nb)
        a=1+j+aux;
        phi_final(:, a)=phi(:, j).^i; %initilizam urmatoarele coloane cu polinoame liniare
        matrice_grad(1, a)=i;
        matrice_grad(2, a)=nthprime(n);
        numar_coloane_completate=numar_coloane_completate+1; %a cata coloana e completata
        n=n+1;
    end
    aux=aux+(na+nb);
end
phi_final(:,end)=phi(:,1).*phi(:,2);
for i=2:max %combinatiile pe care le incercam
    for j=(i+1):max%numar_coloane_completate %pornind de la urmatorul
        if(matrice_grad(1,i)+matrice_grad(1,j)<=m && matrice_grad(1,i)>0 && matrice_grad(1,j)>0 && i~=j
            &&matrice_grad(2, j)~=matrice_grad(2, i))
                phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);

                new_number=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j)^matrice_grad(1,j);
                contains = any(matrice_grad(2, :) == new_number);
                primi=isprime(matrice_grad(2,i));
                primj=isprime(matrice_grad(2,j));
                if(primi && primj)
                    if(~contains)
                        phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);
                        matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);

                        matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j)^matrice_grad(1,j);
                        numar_coloane_completate=numar_coloane_completate+1;
                    end
                end
            end
        end
    end
end
```

```

new_number=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j);
contains = any(matrice_grad(2, :) == new_number);
primi=isprime(matrice_grad(2,i));
primj=isprime(matrice_grad(2,j));
if(primi&&~primj)
if(~contains)
phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);
matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j);
numar_coloane_completate=numar_coloane_completate+1;
end
end
new_number=matrice_grad(2,i)*matrice_grad(2,j)^matrice_grad(1,j);
contains = any(matrice_grad(2, :) == new_number);
primi=isprime(matrice_grad(2,i));
primj=isprime(matrice_grad(2,j));
if(~primi&&primj)
if(~contains)
phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);
matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)*matrice_grad(2,j)^matrice_grad(1,j);
numar_coloane_completate=numar_coloane_completate+1;
end
end

new_number=matrice_grad(2,i)*matrice_grad(2,j);
contains = any(matrice_grad(2, :) == new_number);
primi=isprime(matrice_grad(2,i));
primj=isprime(matrice_grad(2,j));
if(~primi&&~primj)
if(~contains)
phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);
matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)*matrice_grad(2,j);
numar_coloane_completate=numar_coloane_completate+1;
end
end
end
end
teta=phi_final\y_id;
% y_hat_id_sim = phi_final*teta;
Y=zeros(1,max);
y_hat_id_sim=zeros( length(y_id), 1);

```

```

Y(max)=1;
for k=1:length(y_val)
i=1;
for k1=1:na
if(k-k1>0)
mat_de_1(i)=y_hat_id_sim(k-k1);
i=i+1;

else
mat_de_1(i)=0;
i=i+1;

end
end
for k1=1:nb
if(k-k1>0)
mat_de_1(i)=x_id(k-k1);
i=i+1;
else
mat_de_1(i)=0;
i=i+1;
end
end

i=i-1;
aux=0;
numar_coloane_completate=1;
matrice_grad=zeros(2, max);
n=1;
for ridicare_grad=1:m
for fiecare_element=1:i
Y(n)=mat_de_1(foecare_element)^ridicare_grad;
matrice_grad(1, n)=ridicare_grad;
matrice_grad(2, n)=nthprime(n);
n=n+1;
end
end
for i=1:(max-1) %numarul coloanei
for j=(i+1):(max-1) %pornind de la urmatorul
if(matrice_grad(1,i)+matrice_grad(1,j)<=m && matrice_grad(1,i)>0 && matrice_grad(1,j)>0 && i~=j &&matrice_grad(2,i)~=matrice_grad(2, i))

```



```

new_number=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j)^matrice_grad(1,j);
contains = any(matrice_grad(2, :) == new_number);
primi=isprime(matrice_grad(2,i));
primj=isprime(matrice_grad(2,j));
if(primi && primj)
if(~contains)
Y(numar_coloane_completate+1)=Y(i).*Y(j);
matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);

matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j)^matrice_grad(1,j);
numar_coloane_completate=numar_coloane_completate+1;
end
end

new_number=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j);
contains = any(matrice_grad(2, :) == new_number);
primi=isprime(matrice_grad(2,i));
primj=isprime(matrice_grad(2,j));
if(primi&&~primj)
if(~contains)
Y(numar_coloane_completate+1)=Y(i).*Y(j);
matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j);
numar_coloane_completate=numar_coloane_completate+1;
end
end

new_number=matrice_grad(2,i)*matrice_grad(2,j);
contains = any(matrice_grad(2, :) == new_number);
primi=isprime(matrice_grad(2,i));
primj=isprime(matrice_grad(2,j));
if(~primi&&~primj)
if(~contains)
Y(numar_coloane_completate+1)=Y(i).*Y(j);
matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)*matrice_grad(2,j);
numar_coloane_completate=numar_coloane_completate+1;
end
end

end
end
y_hat_id_sim(k)=Y*teta;
end

```

```

MSE_id=sum((y_hat_id_sim-y_id).^2)/length(y_id);
MSE_vector_id = [MSE_vector_id, MSE_id];
end
end
end

```

```

[xid,indxid] = min(MSE_vector_id,[],'all','linear');

```

```

for na = 1:5
for nb = 1:5
for m = 1:3
n=length(y_val);

max=nchoosek(m+na+nb, na+nb);
phi_final=zeros(n, max);

phi=zeros(n, na+nb);

matrice_grad=zeros(2, max);

for i=1:n %pozitiile pe coloane
k=i-1;
for j=1:na %pozitiile pe linie
if k>0
phi(i,j)=-y_val(k);
% k=k-1;
end
k=k-1;

end
end

for i=1:n %pozitiile pe coloane
k=i-1;
for j=(na+1):(nb+na) %pozitiile pe linie
if k>0
phi(i,j)=x_val(k);
end
k=k-1;
end
end
end

```

```

phi_final(:, 1)=1; %am initializat prima coloana cu 1
aux=0;
numar_coloane_completate=1;
for i=1:m
    n=1; %nr elemente
    for j=1:(na+nb)
        a=1+j+aux;
        phi_final(:, a)=phi(:, j).^i; %initilizam urmatoarele coloane cu polinoame liniare
        matrice_grad(1, a)=i;
        matrice_grad(2, a)=nthprime(n);
        numar_coloane_completate=numar_coloane_completate+1; %a cata coloana e completata
        n=n+1;

    end
    aux=aux+(na+nb);
end
phi_final(:,end)=phi(:,1).*phi(:,2);
for i=2:max %combinatiile pe care le incercam
    for j=(i+1):max%numar_coloane_completate %pornind de la urmatorul
        if(matrice_grad(1,i)+matrice_grad(1,j)<=m && matrice_grad(1,i)>0 && matrice_grad(1,j)>0 && i~=j
            &&matrice_grad(2, j)~=matrice_grad(2, i))
                phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);
                new_number=matrice_grad(2,i)*matrice_grad(1,i)*matrice_grad(2,j)^matrice_grad(1,j);
                contains = any(matrice_grad(2, :) == new_number);
                primi=isprime(matrice_grad(2,i));
                primj=isprime(matrice_grad(2,j));
                if(primi && primj)
                    if(~contains)
                        phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);
                        matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
                        matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j)^matrice_grad(1,j);
                        numar_coloane_completate=numar_coloane_completate+1;
                    end
                end
            end
        new_number=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j);
        contains = any(matrice_grad(2, :) == new_number);
        primi=isprime(matrice_grad(2,i));
        primj=isprime(matrice_grad(2,j));
        if(primi&&~primj)
            if(~contains)
                phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);
                matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
                matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j);
                numar_coloane_completate=numar_coloane_completate+1;
            end
        end
    end
end

```

```

new_number=matrice_grad(2,i)*matrice_grad(2,j)^matrice_grad(1,j);
contains = any(matrice_grad(2, :) == new_number);
primi=isprime(matrice_grad(2,i));
primj=isprime(matrice_grad(2,j));
if(~primi&&primj)
    if(~contains)
        phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);
        matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
        matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)*matrice_grad(2,j)^matrice_grad(1,j);
        numar_coloane_completate=numar_coloane_completate+1;
    end
end

new_number=matrice_grad(2,i)*matrice_grad(2,j);
contains = any(matrice_grad(2, :) == new_number);
primi=isprime(matrice_grad(2,i));
primj=isprime(matrice_grad(2,j));
if(~primi&&~primj)
    if(~contains)
        phi_final(:, numar_coloane_completate+1)=phi_final(:, i).*phi_final(:,j);
        matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
        matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)*matrice_grad(2,j);
        numar_coloane_completate=numar_coloane_completate+1;
    end
end

end
end
teta = phi_final\y_val;
%y_hat_val_sim = phi_final * teta;
Y=zeros(1,max);
y_hat_val_sim=zeros(length(y_val), 1);
Y(max)=1;
for k=1:length(y_val)
    i=1;
    for k1=1:na
        if(k-k1>0)
            mat_de_1(i)=y_hat_val_sim(k-k1);
            i=i+1;
        else
            mat_de_1(i)=0;
            i=i+1;
        end
    end
end

```

```

end
for k1=1:nb
    if(k-k1>0)
        mat_de_1(i)=x_val(k-k1);
        i=i+1;
    else
        mat_de_1(i)=0;
        i=i+1;
    end
end
i=i-1;
aux=0;
numar_coloane_completate=1;
matrice_grad=zeros(2, max);
n=1;
%mat_de_1
for ridicare_grad=1:m
    for fiecare_element=1:i
        Y(n)=mat_de_1(fiecare_element)^ridicare_grad;
        matrice_grad(1, n)=ridicare_grad;
        matrice_grad(2, n)=nthprime(n);
        n=n+1;
    end
end
for i=1:(max-1) %numarul coloanei
    for j=(i+1):(max-1) %pornind de la urmatorul
        if(matrice_grad(1,i)+matrice_grad(1,j)<=m && matrice_grad(1,i)>0 && matrice_grad(1,j)>0 && i~j &&matrice_grad(2,
j)~=matrice_grad(2, i))

            new_number=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j)^matrice_grad(1,j);
            contains = any(matrice_grad(2, :) == new_number);
            primi=isprime(matrice_grad(2,i));
            primj=isprime(matrice_grad(2,j));
            if(primi && primj)
                if(~contains)
                    Y(numar_coloane_completate+1)=Y(i).*Y(j);
                    matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
                    matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j)^matrice_grad(1,j);
                    numar_coloane_completate=numar_coloane_completate+1;
                end
            end
        end
    end
end
matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j)^matrice_grad(1,j);
    numar_coloane_completate=numar_coloane_completate+1;
end
end

```

```

new_number=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j);
    contains = any(matrice_grad(2, :) == new_number);
    primi=isprime(matrice_grad(2,i));
    primj=isprime(matrice_grad(2,j));
    if(primi&&~primj)
        if(~contains)
            Y(numar_coloane_completate+1)=Y(i).*Y(j);
            matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
            matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)^matrice_grad(1,i)*matrice_grad(2,j);
            numar_coloane_completate=numar_coloane_completate+1;
        end
    end
new_number=matrice_grad(2,i)*matrice_grad(2,j);
    contains = any(matrice_grad(2, :) == new_number);
    primi=isprime(matrice_grad(2,i));
    primj=isprime(matrice_grad(2,j));
    if(~primi&&~primj)
        if(~contains)
            Y(numar_coloane_completate+1)=Y(i).*Y(j);
            matrice_grad(1,numar_coloane_completate+1)=matrice_grad(1,i)+matrice_grad(1,j);
            matrice_grad(2,numar_coloane_completate+1)=matrice_grad(2,i)*matrice_grad(2,j);
            numar_coloane_completate=numar_coloane_completate+1;
        end
    end
end
end
end
y_hat_val_pred(k)=Y*teta;
end
MSE_val = sum((y_hat_val_pred-y_val).^2)/length(y_val);
MSE_vector_val = [MSE_vector_val, MSE_val];
end
end
end
sys_id=iddata(y_id, x_id, t_id(2)-t_id(1));sys_val=iddata(y_val, x_val, t_val(2)-t_val(1));
sys_nlarx=iddata(y_hat_id_sim,x_id, t_id(2)-t_id(1));
compare(sys_nlarx, sys_val);title("simulare identificare")
sys_nlarx=iddata(y_hat_id_sim,x_val, t_val(2)-t_val(1));
compare(sys_nlarx, sys_val);title("simulare validare")

```

```

[xval,indxval] = min(MSE_vector_val,[],'all','linear');
figure,plot(MSE_vector_id), hold on, plot(indxid,xid,'r*'), title('mse id simulare')
figure, plot(MSE_vector_val), hold on, plot(indxval,xval,'r*'), title('mse val simulare')

```