

# MODELUL ARX NELINIAR

Studenti: Enescu Ruxandra, Hegheş Antonia, Niculiţă Roxana  
Numar identificare: 6

# CUPRINS

## 1. Parte introductivă

1.1. Motivație

1.2. Descrierea problemei

## 2. Algoritm și proces

2.1. Prezentare date

2.2. Predictie și simulare

2.3. Găsirea parametrilor  $\theta$

2.4. Metoda implementată

## 3. Rezultate

3.1. Grafice și observații

## 4. Discuții ale problemei

4.1. Optimizare soluție

4.2. Concluzii

# PARTE INTRODUCȚIVĂ – Motivație

## Ce este un model ARX?

- AutoRegressive (depinde de valorile lui  $y$  precedente) with eXogenous inputs (dependenta de  $u$ )
- O reprezentare matematica folosita pentru a modela comportamentul unui sistem dinamic.

## Ce este un sistem dinamic?

- Un sistem a carui stare evolueaza in timp;
- Comportamentul unui sistem dinamic in orice moment este dependent de starile sale anterioare si de intrarile primite.

# PARTE INTRODUCȚIVĂ – Descrierea problemei

- Se dau următoarele date (#6): doua seturi de date masurate de la un sistem dinamic, ce cuprind intrarea, iesirea si perioada de esantionare;
- Se construiește modelul pornind de la coeficientii  $n_a$ ,  $n_b$ ,  $n_k$  si gradul  $m$ . (Pentru simplitate,  $n_k = 1$ )
- Forma generala: (exemplu pentru  $m=1$ )

$$y(k) = \theta_1 y(k-1) + \dots + \theta_{n_a} y(k-n_a) + \theta_{n_a+1} u(k-1) + \dots + \theta_{n_b} u(k-n_b)$$

- Modelul este neliniar deoarece relatia dintre intrare si iesire este descrisa printr-o functie neliniara, iar puterile de ordin superior ale acestora trebuie sa fie luate in considerare.

# ALGORITM ȘI PROCES – Prezentare date

- Identificarea sistemelor se bazează pe separarea setului de date în identificare și validare.
- Astfel, se utilizează vectorii  $u_{id}$  și  $y_{id}$  pentru construirea matricei de regresori  $\phi_{id}$ , respectiv  $u_{val}$  și  $y_{val}$  pentru datele de validare.
- Se creează vectorii  $X_{id}$  și  $X_{val}$  în care se stochează comportamentul sistemului la momente anterioare, folosite în etapele de simulare și predicție.

# ALGORITM ȘI PROCES – Predictie si simulare

## PREDICTIE

- Predictia foloseste valorile iesirilor intarziate ale sistemului.
- Se concentreaza pe estimarile cuantificabile ale viitoarelor stari sau iesiri ale sistemului, bazate pe date existente.
- Forma aproximatorului:

$$\hat{y}(k) = -a_1 y(k-1) - a_2 y(k-2) - \dots - a_{na} y(k-na) \\ b_1 u(k-1) + b_2 u(k-2) + \dots + b_{nb} u(k-nb)$$

# ALGORITM ȘI PROCES – Predictie și simulare

## SIMULARE

- Iesirile precedente ale sistemului nu sunt disponibile.
- Iesirile reale  $y(k - i)$  sunt necunoscute și trebuie adărate înlocuite cu iesirile simulate  $\tilde{y}(k - i)$  la iterații anterioare:
- Forma aproximatorului:

$$\tilde{y}(k) = -a_1\tilde{y}(k-1) - a_2\tilde{y}(k-2) - \dots - a_{na}\tilde{y}(k-na) + b_1u(k-1) + b_2u(k-2) + \dots + b_{nb}u(k-nb)$$

## ALGORITM ȘI PROCES – Găsire parametrilor $\theta$

- Este cunoscut că ieșirea dată  $y_{id}$  a sistemului este rezultatul înmulțirii dintre matricea  $\phi_{id}$ , calculată în etapa de predicție și vectorul de coeficienți  $\theta$ .

$$Y = \Phi\theta$$

- Astfel, aplicând metoda regresiei liniare, vom găsi  $\theta$ .

$$\theta = (\Phi^T \Phi)^{-1} \Phi^T Y$$

- Același vector  $\theta$  se va folosi și în cadrul etapei de simulare.



# ALGORITM ȘI PROCES – Metoda implementata

- Generarea puterilor fiecarui element al polinomului s-a realizat cu ajutorul unei functii ce calculeaza toate combinatiile de numere intre 0 si m.
- S-au eliminat toate liniile unde se depasea gradul maxim m, rezultand o matrice ce contine puterile corespunzatoare fiecarui termen al polinomului.

Ex pentru  $m_a = m_b = 2$  și  $m = 3$

$$X_{id} = [y(k-1); y(k-2); u(k-1); u(k-2)]$$

$$\Rightarrow y(k) = 1 + y^3(k-1) + y^3(k-2) + u^3(k-1) + u^3(k-2) + y^2(k-1)y(k-2) + \dots \\ \cdot [\theta_1 \quad \theta_2 \quad \dots \quad \theta_{C_{m+m_a+m_b}^m}]$$

# ALGORITM ȘI PROCES – Metoda implementata

- Numarul de linii al matricei de puteri este egal cu numarul termenilor din polinom si se calculeaza cu formula:

$$C_{m+na+nb}^m$$

- Exemplu pentru  $m=3$  si  $na=nb=2$ :

$C_7^3 = 35$  – numarul de termeni din polinomul ridicat la puterea  $m$ .

4 – numarul termenilor din polinom ( $na+nb$ )

35x4 double

	1	2	3	4
1	0	0	0	0
2	0	0	0	1
3	0	0	0	2
4	0	0	0	3
5	0	0	1	0
6	0	0	1	1

Fig. 2.1

# ALGORITM ȘI PROCES – Metoda implementata

- Se aplica algoritmul corespunzator fiecarei etape si se evidentiaza grafic iesirile calculate in comparatie cu datele primite.
- Se calculeaza valorile MSE pentru fiecare y, cu ajutorul carora se vor identifica parametrii optimi na, nb si gradul m.

```
%EROARE SIMULARE IDENTIFICARE
```

```
MSE_id_sim = 0;
```

```
for p=1:N
```

```
    MSE_id_sim = MSE_id_sim + (y_id(p)-simulare_y_id(p))*(y_id(p)-simulare_y_id(p));
```

```
end
```

```
MSE_id_sim = MSE_id_sim/N;
```

# REZULTATE – Grafice și observații

## PREDICTIE IDENTIFICARE $na=nb=3 ; m=3$

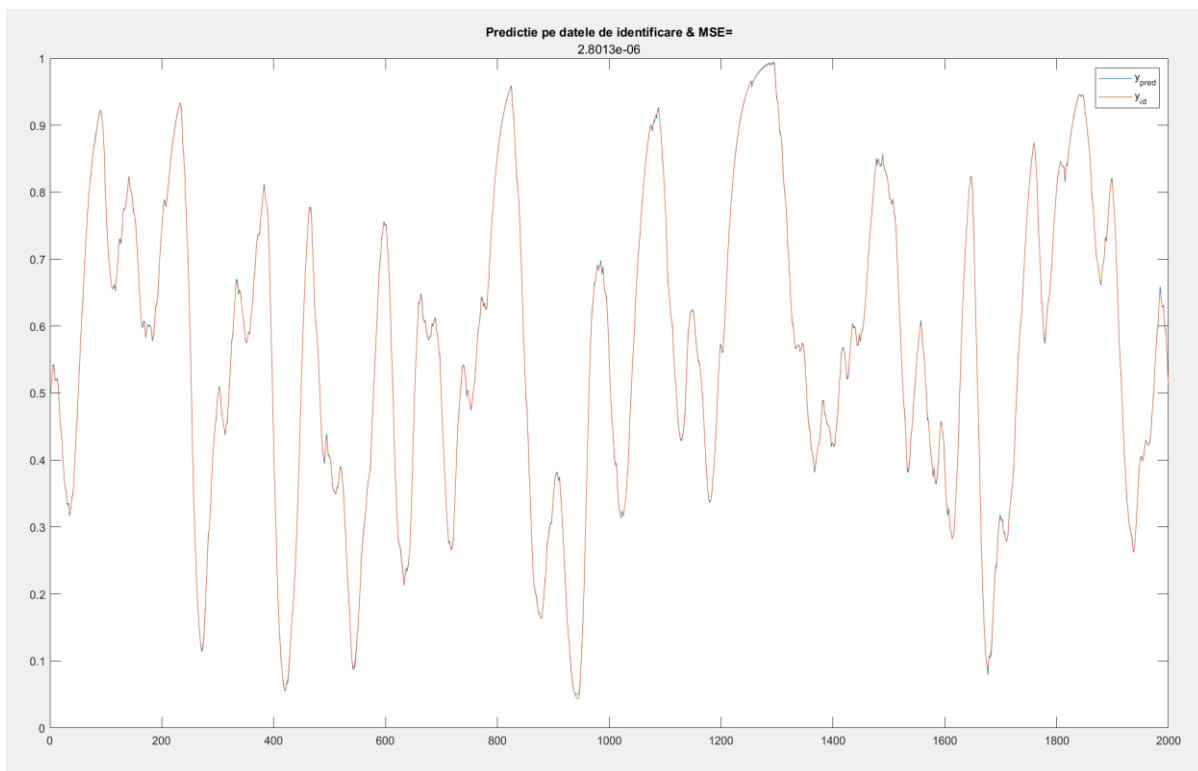


Fig. 3.2.1

## SIMULARE IDENTIFICARE $na=nb=1 ; m=2$

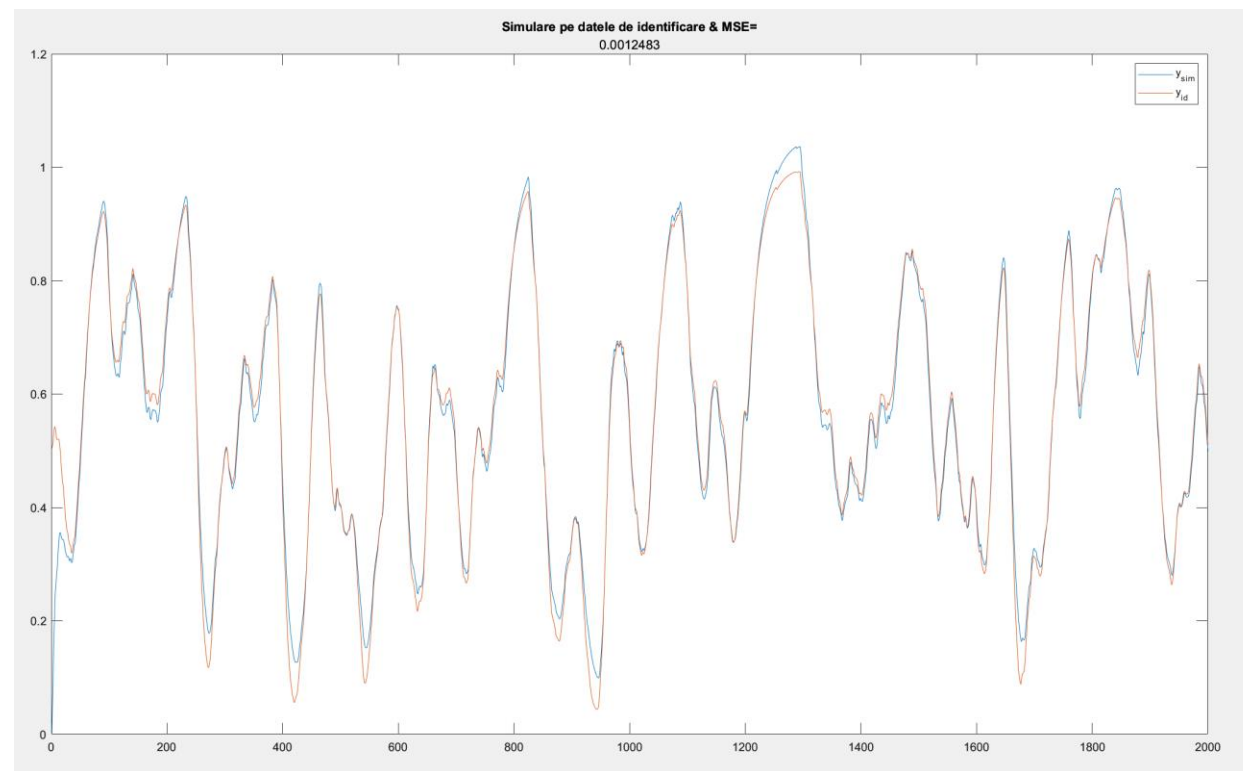


Fig. 3.2.2

# REZULTATE – Grafice și observații

## PREDICTIE VALIDARE na=nb=1 ; m=3

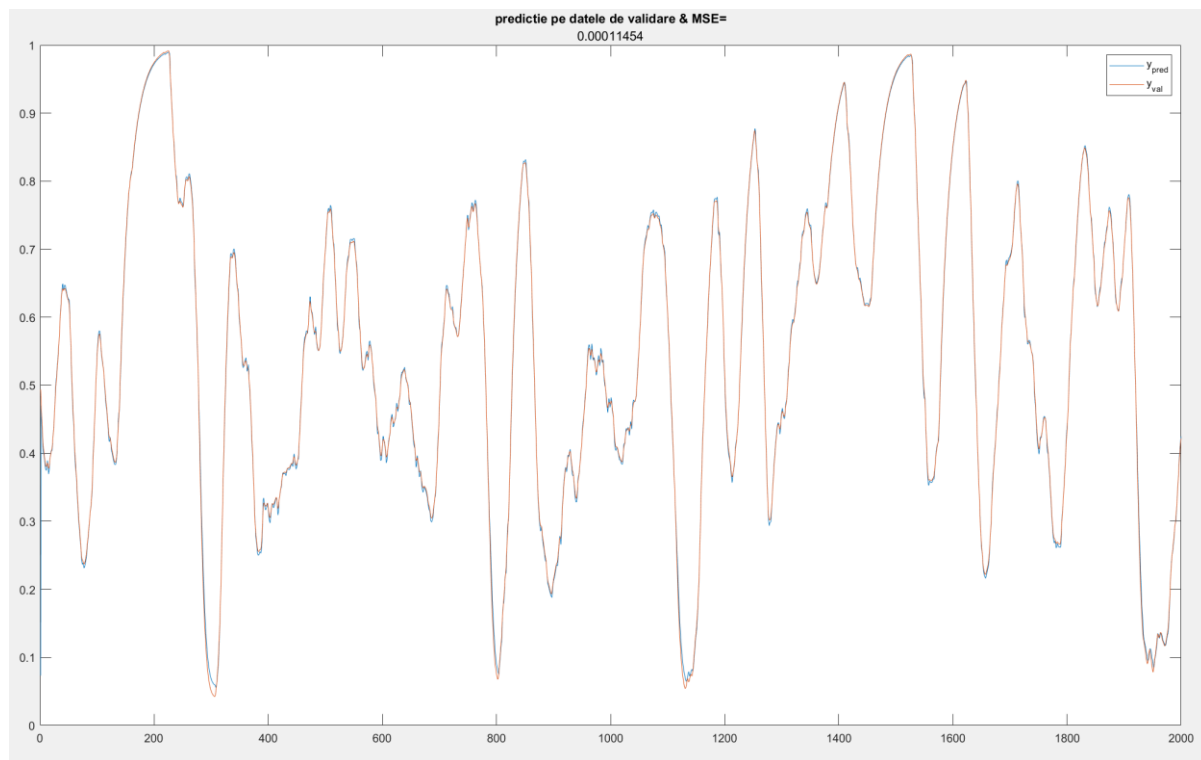


Fig. 3.2.3

## SIMULARE VALIDARE na=nb=1 ; m=2

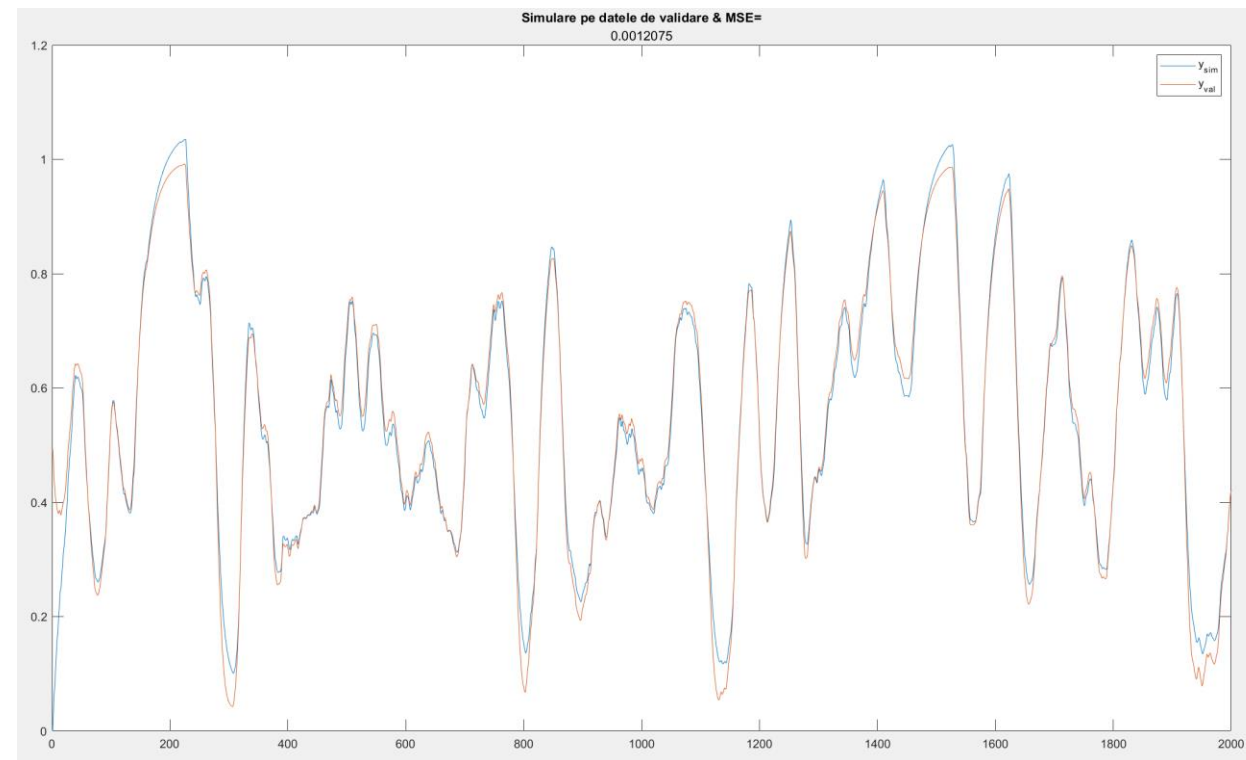


Fig. 3.2.4

# DISCUȚII ALE PROBLEMEI – Optimizare soluție

$n_a=n_b=3$  ;  $m=3$  --> model instabil pentru simulare

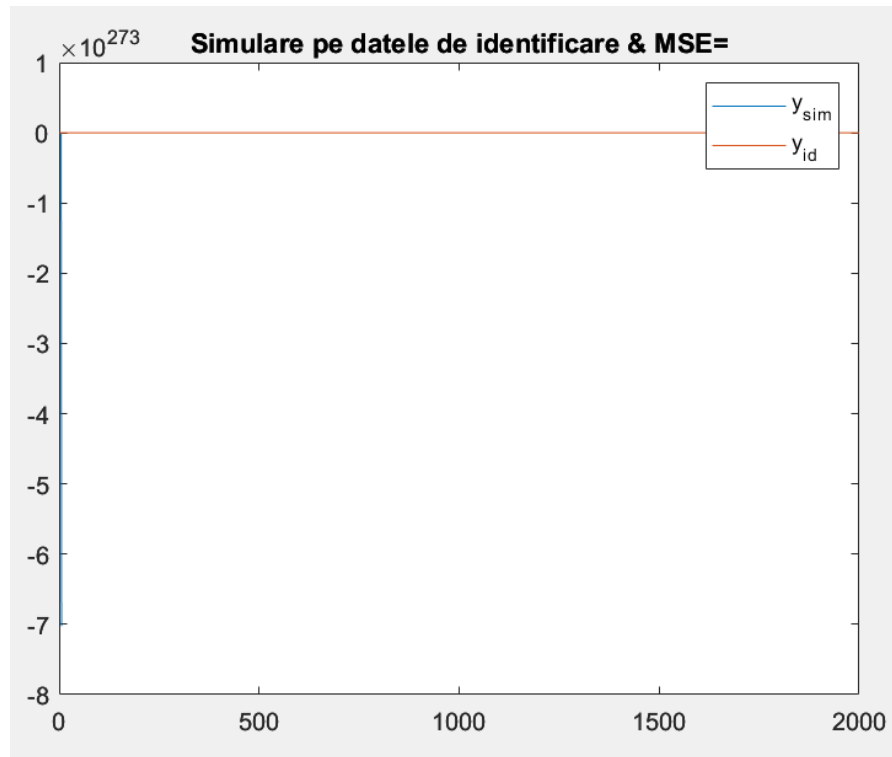


Fig. 3.2.5

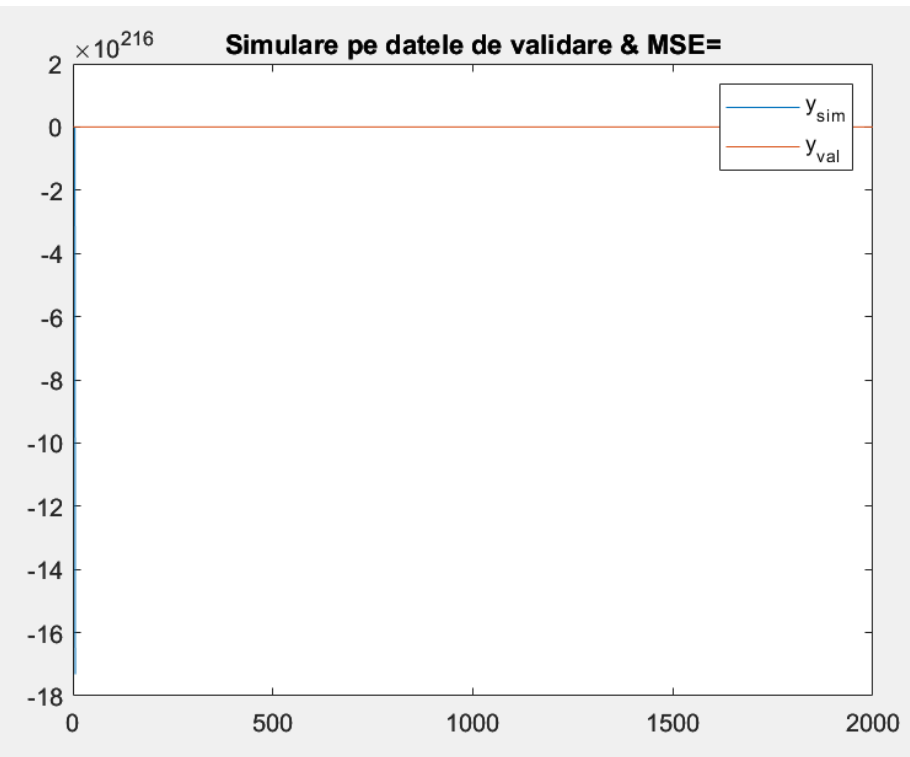


Fig. 3.2.6


## DISCUTII ALE PROBLEMEI – Concluzii

- Un grad mai mare nu inseamna o performanta mai buna, de aceea se urmareste evolutia MSE urilor pentru fiecare caz.
- Se observa ca odata cu incrementarea  $n_a$  si  $n_b$ , metoda Predictiei devine din ce in ce mai precisa.
- Metoda Simularii este mai putin precisa, insa este mai apropiata de realitate.

## DISCUTII ALE PROBLEMEI – Concluzii

<----- m ----->


na

 3x3 double

	1	2	3
1	1.5179e-04	1.3545e-04	1.1548e-04
2	1.2677e-04	6.6291e-05	4.3591e-06
3	4.8115e-06	9.7762e-06	2.8013e-06

Fig. 4.1

MSE IDENTIFICARE PREDICTIE

 3x3 double

	1	2	3
1	0.0018	0.0012	0.0017
2	0.3371	NaN	NaN
3	NaN	NaN	NaN

Fig. 4.2


MSE IDENTIFICARE SIMULARE



## DISCUTII ALE PROBLEMEI – Concluzii

<----- m ----->


na



	1	2	3
1	1.4686e-04	1.3070e-04	1.1454e-04
2	1.2333e-04	0.0018	0.5116
3	2.1598	0.0013	2.6324e+03

Fig. 4.3

MSE VALIDARE PREDICTIE



	1	2	3
1	0.0019	0.0012	0.0022
2	0.3151	NaN	NaN
3	NaN	NaN	NaN

Fig. 4.4

MSE VALIDARE SIMULARE

Va multumim!

```

close all
clear all
clc

% incarcarea setului de date
load('iddata-06.mat');

% load('c1.mat'); % m optim id pred
% load('c2.mat'); % m optim id sim
% load('c3.mat'); % m optim val pred
% load('c4.mat'); % m optim val sim
%
% load('r1.mat'); % na optim id pred
% load('r2.mat'); % na optim id sim
% load('r3.mat'); % na optim val pred
% load('r4.mat'); % na optim val sim

r1 = 3;
c1 = 3;

r2 = 1;
c2 = 2;

r3 = 1;
c3 = 2;

r4 = 1;
c4 = 2;

% alegerea parametrilor na, nb si a gradului m
na = r2;
nb = na;
m = c2;

% extragerea datelor de identificare si validare
ts = id.Ts;
u_id = id.u;
y_id = id.y;
u_val = val.u;
y_val = val.y;

N = length(y_id);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% generam toate combinatiile de puteri posibile
puteri_posibile = permutari(na+nb, m);

% eliminand toate combinatiile de puteri care adunate, dau un grad mai mare decat m
capatul_puterilor = zeros(1, na*2);

```

```

for i = 1:length(puteri_posibile)
    sum = 0;
    for j = 1:na*2
        sum = sum + puteri_posibile(i, j);
    end

    if sum <= m
        capatul_puterilor = [capatul_puterilor; puteri_posibile(i, :)];
    end
end
capatul_puterilor = capatul_puterilor(2:end, :);

%% PREDICTIE Y ID
for i = 1:N
    Xid_pred = zeros(1, 2*na);
    for j = 1:na+nb
        if i-j+na > 0 && j > na
            Xid_pred(j) = u_id(i-j+na);
        end
        if i-j>0 && j <= na
            Xid_pred(j) = -y_id(i-j);
        end
    end

    for i1 = 1:nchoosek(m+na+nb, m)
        produs = 1;
        for j1 = 1:na+nb
            produs = produs*Xid_pred(j1)^capatul_puterilor(i1, j1);
            % X1 X2 X3 X4 ...
        end
        phi_id(i, i1) = produs;
    end
end

theta = phi_id\y_id;

predictie_y_id = phi_id * theta;

%%EROARE PREDICTIE IDENTIFICARE
MSE_id_pred = 0;

for p=1:N
    MSE_id_pred = MSE_id_pred + (y_id(p)-predictie_y_id(p))*(y_id(p)-predictie_y_id(p));
end

MSE_id_pred = MSE_id_pred/N;

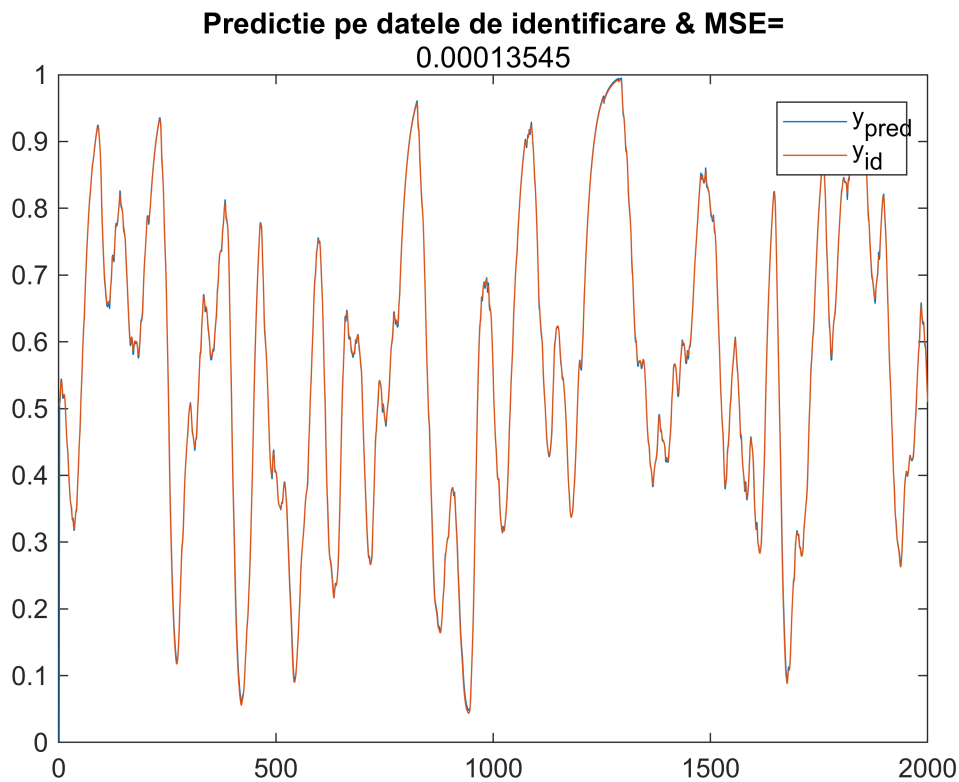
figure;
plot(predictie_y_id);
hold on;

```

```

plot(y_id);
title('Predictie pe datele de identificare & MSE=', MSE_id_pred);
legend('y_p_r_e_d', 'y_i_d');

```



```

%% PREDICTIE Y VAL
for i = 1:N
    Xval_pred = zeros(1, 2*na);
    for j = 1:na+nb
        if i-j+na > 0 && j > na
            Xval_pred(j) = u_val(i-j+na);
        end
        if i-j>0 && j <= na
            Xval_pred(j) = -y_val(i-j);
        end
    end

    for i1 = 1:nchoosek(m+na+nb, m)
        produs = 1;
        for j1 = 1:na+nb
            produs = produs*Xval_pred(j1)^capatul_puterilor(i1, j1);
            % X1 X2 X3 X4 ...
        end
        phi_val(i, i1) = produs;
    end
end
end

```

```

predictie_y_val = phi_val*theta;

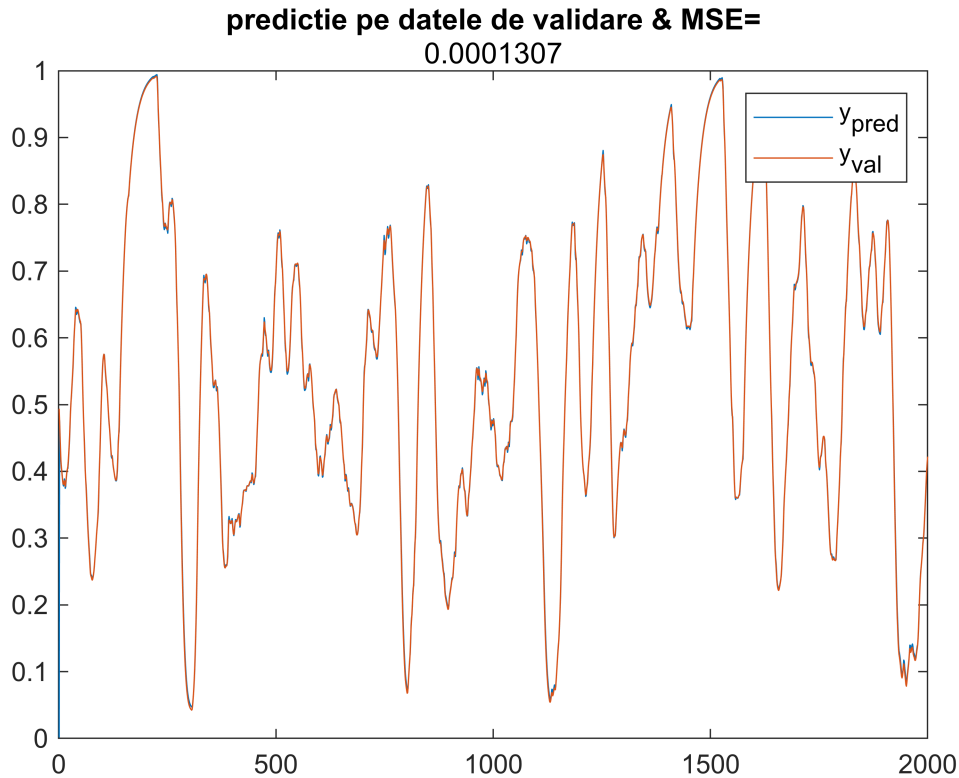
%EROARE PREDICTIE VALIDARE
MSE_val_pred = 0;

for p=1:N
    MSE_val_pred = MSE_val_pred + (y_val(p)-predictie_y_val(p))*(y_val(p)-predictie_y_val(p));
end

MSE_val_pred = MSE_val_pred/N;

figure;
plot(predictie_y_val);
hold on;
plot(y_val);
title('predictie pe datele de validare & MSE=', MSE_val_pred);
legend('y_p_r_e_d', 'y_v_a_l');

```



```

%% SIMULARE Y ID
simulare_y_id = zeros(1, N); % vectorul pe care il adunam element cu element pt a forma y tilda
Xid_sim = zeros(1, na+nb);
phi_simulare = [];

for i = 2:N

```

```

for j = 1:na+nb
    if (j<=na)
        if (j>=i)
            Xid_sim(1,j) = 0;
        end
        if (j<i)
            Xid_sim(1,j) = -simulare_y_id(1,i-j);
        end
    end

    if (j>na)
        if (j-na>=i)
            Xid_sim(1,j) = 0;
        end
        if (j-na<i)
            Xid_sim(1,j) = u_id(i-(j-na),1);
        end
    end
end

for i1 = 1:nchoosek(m+na+nb, m)
    produs = 1;
    for j1 = 1:na+nb
        produs = produs*((Xid_sim(j1))^capatul_puterilor(i1, j1));
        % X1 X2 X3 X4 ... face ridicare la putere
    end
    phi_simulare(i,i1) = produs;
end
sum = 0;
for k=1:nchoosek(m+na+nb,m)
    sum = sum + theta(k,1)*phi_simulare(i, k);
end
simulare_y_id(1, i) = sum;
end

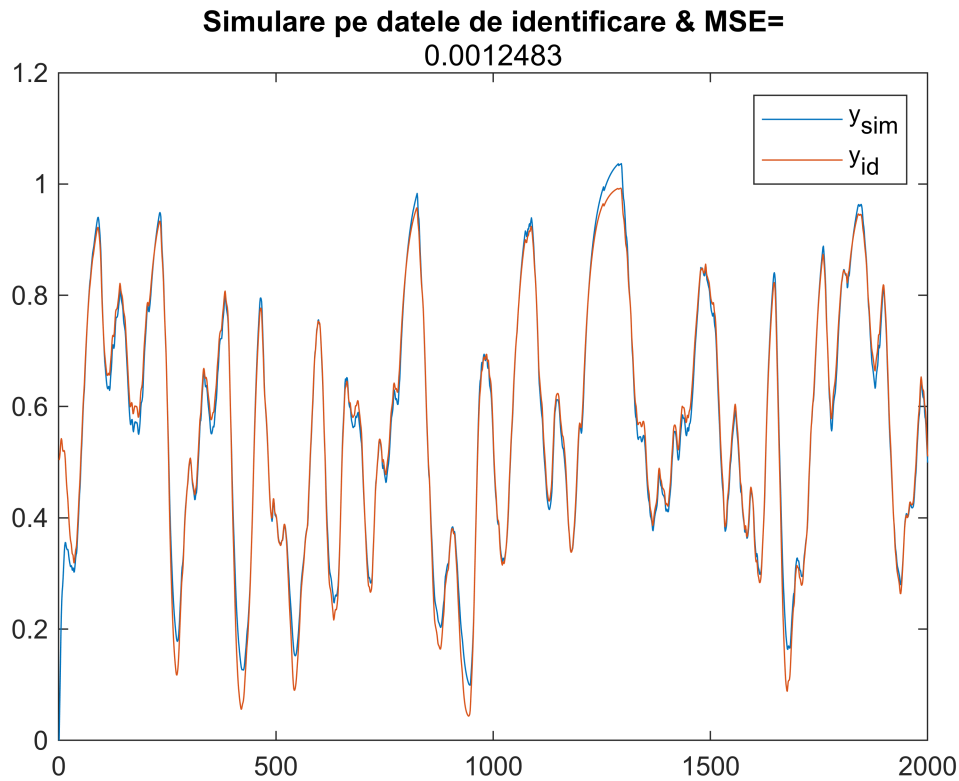
%EROARE SIMULARE IDENTIFICARE
MSE_id_sim = 0;

for p=1:N
    MSE_id_sim = MSE_id_sim + (y_id(p)-simulare_y_id(p))*(y_id(p)-simulare_y_id(p));
end

MSE_id_sim = MSE_id_sim/N;

figure;
plot(simulare_y_id);
hold on
plot(y_id);
title('Simulare pe datele de identificare & MSE=', MSE_id_sim);
legend('y_s_i_m', 'y_i_d');

```



```
%% SIMULARE Y VAL
```

```
simulare_y_val = zeros(1, N); % vectorul pe care il adunam element cu element pt a forma y til
```

```
Xval_sim = zeros(1, na+nb);
```

```
phi_simulare = [];
```

```
for i = 2:N
```

```
    for j = 1:na+nb
```

```
        if (j<=na)
```

```
            if (j>=i)
```

```
                Xval_sim(1,j) = 0;
```

```
            end
```

```
            if (j<i)
```

```
                Xval_sim(1,j) = -simulare_y_val(1,i-j);
```

```
            end
```

```
        end
```

```
    if (j>na)
```

```
        if (j-na>=i)
```

```
            Xval_sim(1,j) = 0;
```

```
        end
```

```
        if (j-na<i)
```

```
            Xval_sim(1,j) = u_val(i-(j-na),1);
```

```
        end
```

```
    end
```



```

end

for i1 = 1:nchoosek(m+na+nb, m)
    produs = 1;
    for j1 = 1:na+nb
        produs = produs*((Xval_sim(j1))^capatul_puterilor(i1, j1));
        % X1 X2 X3 X4 ... face ridicare la putere
    end
    phi_simulare(i,i1) = produs;
end
sum = 0;
for k=1:nchoosek(m+na+nb,m)
    sum = sum + theta(k,1)*phi_simulare(i, k);
end
simulare_y_val(1, i) = sum;
end

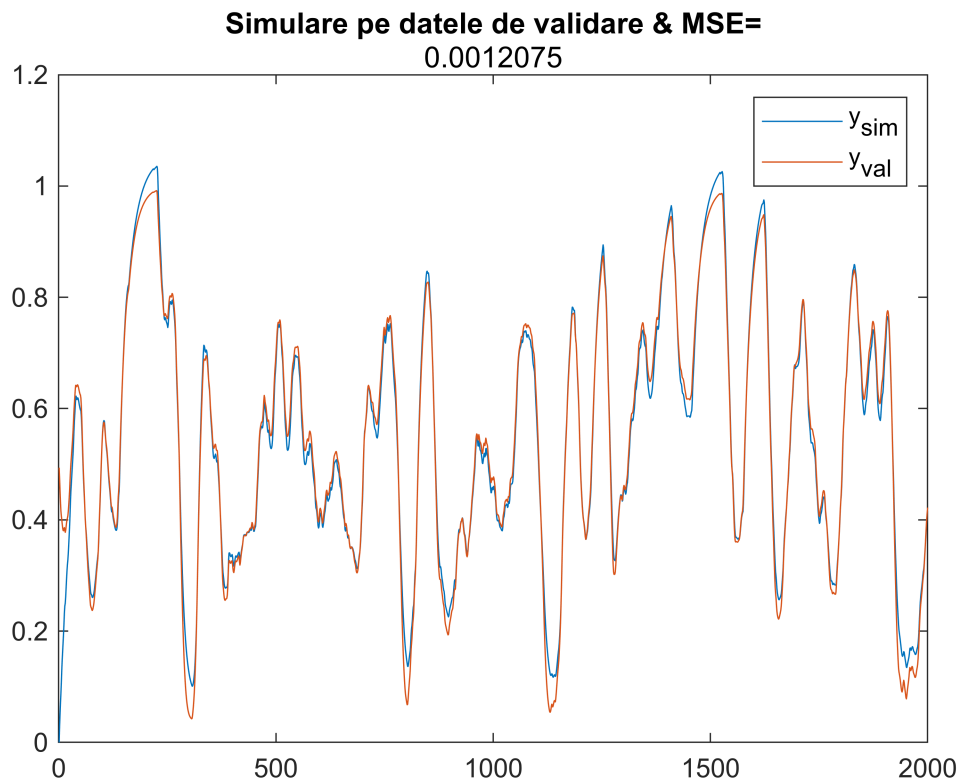
%EROARE SIMULARE VALIDARE
MSE_val_sim = 0;

for p=1:N
    MSE_val_sim = MSE_val_sim + (y_val(p)-simulare_y_val(p))*(y_val(p)-simulare_y_val(p));
end

MSE_val_sim = MSE_val_sim/N;

figure;
plot(simulare_y_val);
hold on
plot(y_val);
title('Simulare pe datele de validare & MSE=', MSE_val_sim);
legend('y_s_i_m', 'y_v_a_l');

```



```
%% functia care genereaza toate combinatiile de numere intre 0 si m
```

```
function puteri = permutari(na, m)

puteri = [];
indici = zeros(1, na);

while true
    puteri = [puteri; indici];
    i = na;
    while i > 0 && indici(i) == m
        indici(i) = 0;
        i = i - 1;
    end
    if i == 0
        break;
    end
    indici(i) = indici(i) + 1;
end
end
```