NOIEMBRIE-DECEMBRIE 2022

PROIECT IDENTIFICAREA SISTEMELOR

PARTEA A II-A:

ARX NELINIAR

Echipa:

STUDENȚI:

- Pîrvulescu Roberta-Alina
- Pleș Ovidiu Vasile Claudiu

GRUPA:

- 30135/2

INDECȘI PROIECT:

- 2/10

CUPRINS:

INTRODUCERE

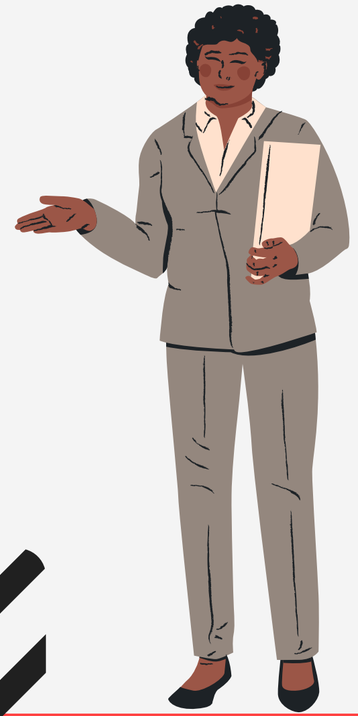
IMPLEMENTARE

REZULTATE

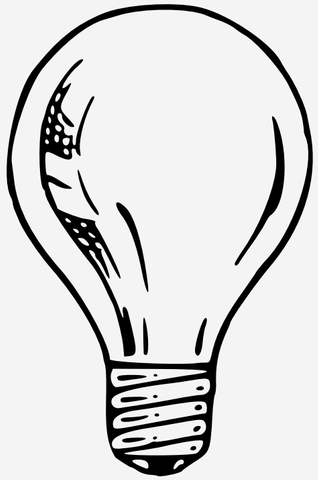
GRAFICE

CONCLUZII

1.INTRODUCERE



Introducere:



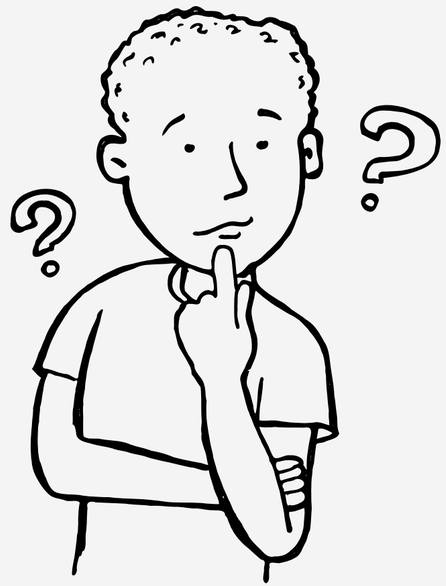
Scopul lucrării:

- identificarea unui sistem pe baza modelului tip cutie neagră

Metoda utilizată:

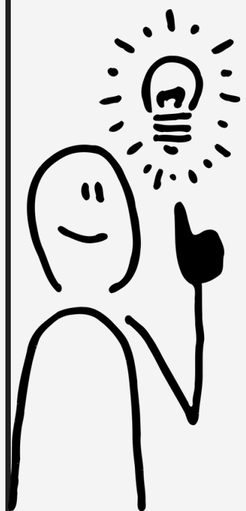
- ARX neliniar

Modelul ARX neliniar (NARX)



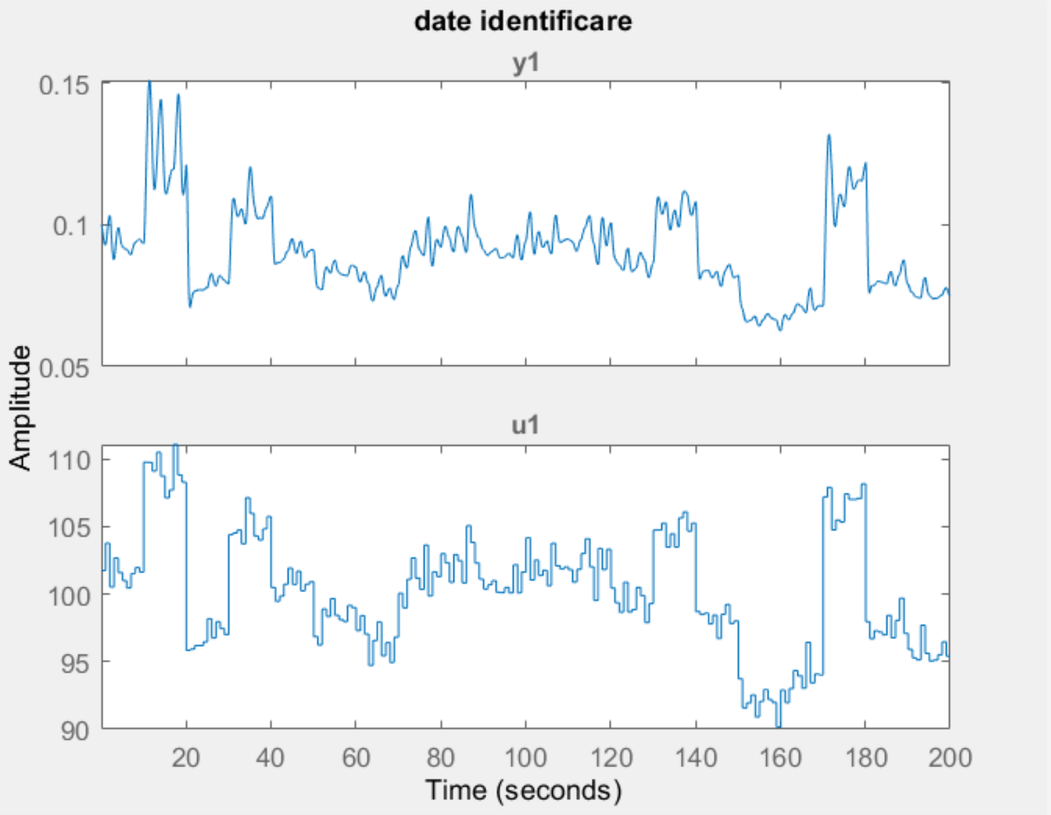
STRUCTURA:

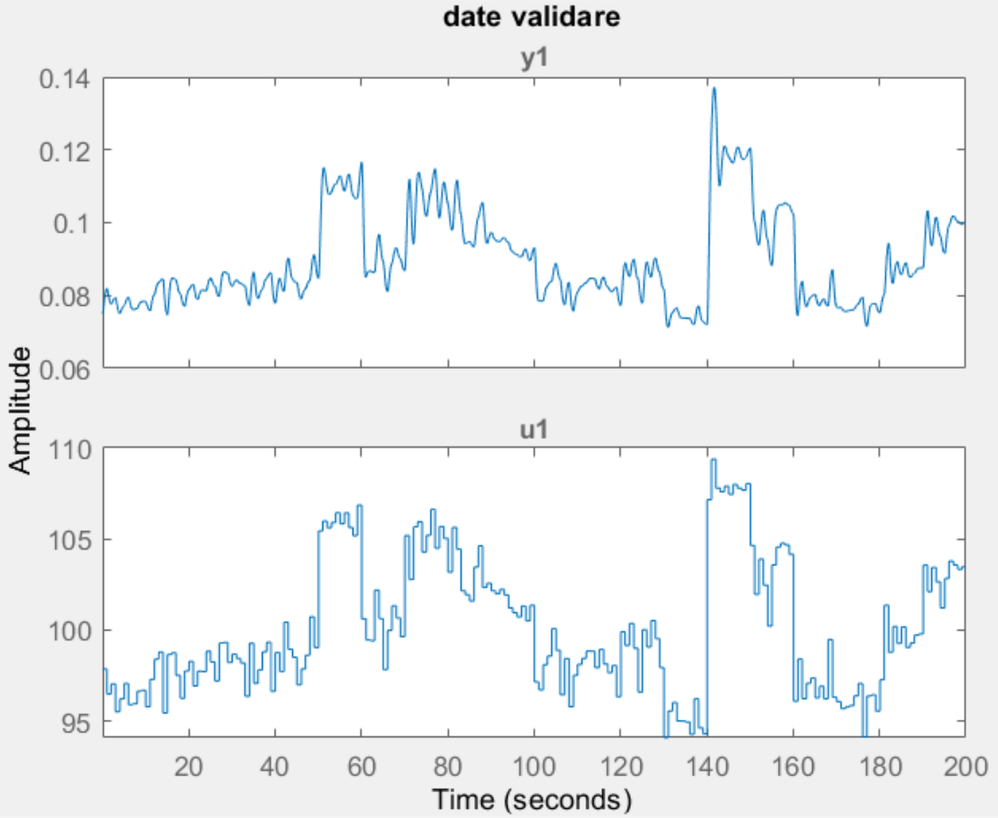
$$\begin{aligned}\hat{y}(k) &= p(y(k-1),\ldots,y(k-na),u(k-nk),u(k-nk-1),\ldots,u(k-nk-nb+1)) \\ &= p(d(k))\end{aligned}$$



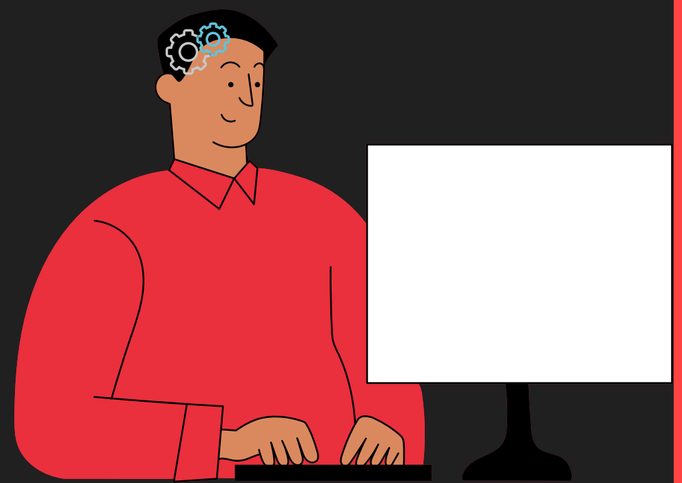
na & nb = ordinele NARX
nk = întârzierea
d(k) = vector de intrări și ieșiri întârziate
p = polinom de grad m de variabilele conținute de vectorul d
m ∈ {1 ,2 ,3}

DATE FURNIZATE





2.IMPLEMENTARE



Implementare:

- bazată pe valorile anterioare, cunoscute, ale semnalelor de intrare și ieșire
- identificare necesară => obținerea parametrilor
- ieșire determinată pe 2 fronturi:

1. predicție
2. simulare

Regresorii - Generarea matricei de puteri

```
function matricePuteri = generarePuteri(m, na, nb)
    ultimaLinie = zeros(1, na + nb); %prima combinatie
    matricePuteri = ultimaLinie;
    NrCombinatii = m + 1; %numarul primelor combinatii (o singura coloana din matrice completata)
    NrCombinatiiNoi = 0;
    for i = 1 : na + nb
        while sum(ultimaLinie) < m
            ultimaLinie(i) = ultimaLinie(i) + 1;
            matricePuteri = [matricePuteri; ultimaLinie];
            if i ~= 1 %pentru i == 1 nu exista combinatii anterioare
                NrCombinatiiNoi = NrCombinatiiNoi + 1;
                ultimaLinieNemodificata = ultimaLinie;
                linieGeneratoare = ultimaLinie(i : end); %generare combinatii in functie de combinatiile anterioare
                for j = 2 : NrCombinatii %verifica toate combinatiile anterioare
                    linieAnterioara = matricePuteri(j, :);
                    linieAnterioara = linieAnterioara(1 : i - 1);
                    ultimaLinie = [linieAnterioara , linieGeneratoare]; %se lipeste partea prelucrata a combinatiilor anterioare cu linia generatoare
                    if sum(ultimaLinie) <= m
                        matricePuteri = [matricePuteri; ultimaLinie];
                        NrCombinatiiNoi = NrCombinatiiNoi + 1;
                    end
                end
                ultimaLinie = ultimaLinieNemodificata;%se continua incrementarea cu 1 de la ultima linie
            end
        end
        NrCombinatii = NrCombinatii + NrCombinatiiNoi;
        NrCombinatiiNoi = 0;
        ultimaLinie(i) = 0;
    end
end
```

Regresorii - Generarea matricei de puteri

matricePuteri

35x4 double

	1	2	3	4
1	0	0	0	0
2	1	0	0	0
3	2	0	0	0
4	3	0	0	0
5	0	1	0	0
6	0	2	0	0
7	0	3	0	0
8	0	0	1	0
9	0	0	2	0
10	0	0	3	0
11	0	0	0	1
12	0	0	0	2
13	0	0	0	3

generare de combinații pe baza combinațiilor anterioare

matricePuteri

35x4 double

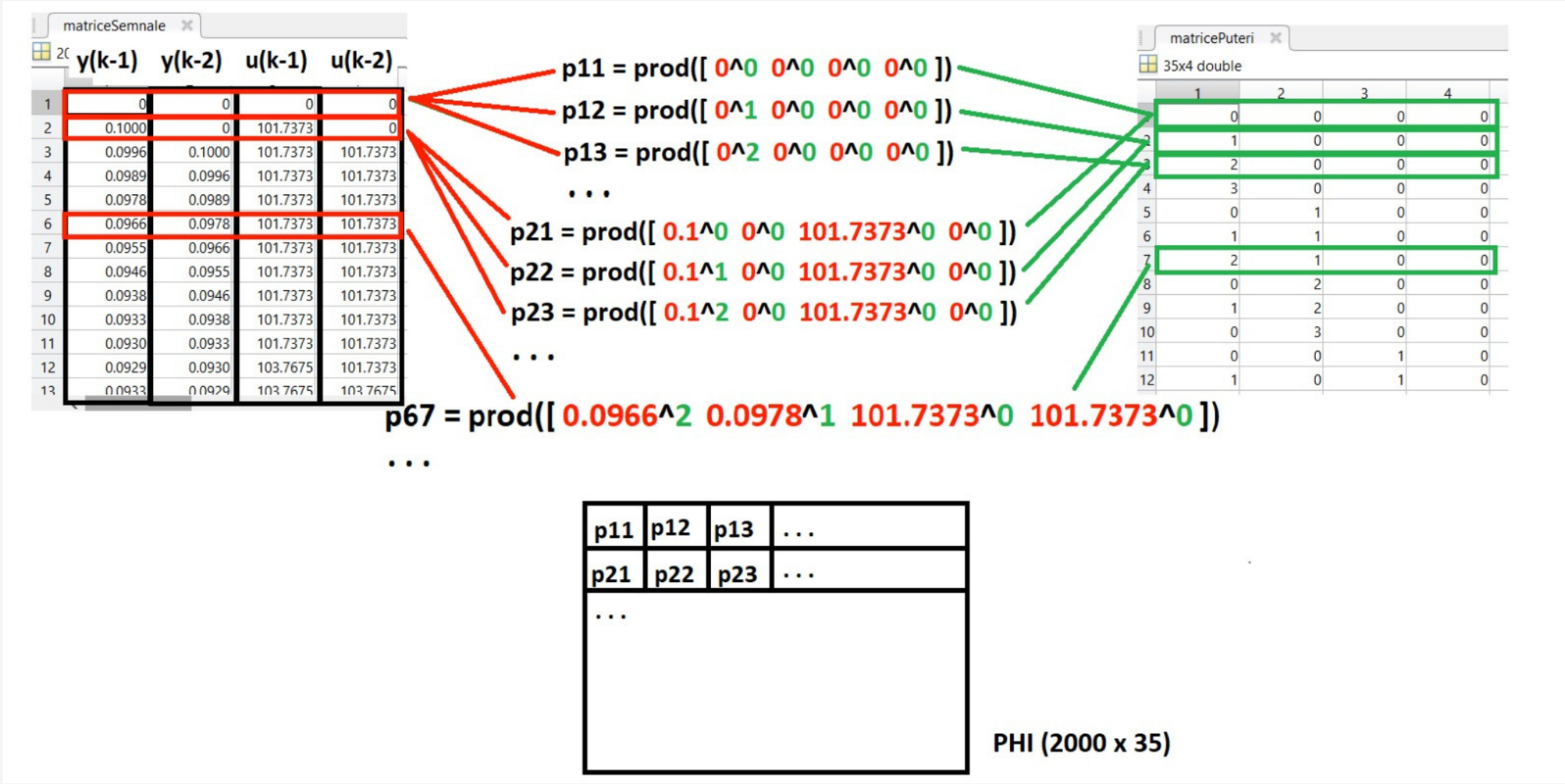
	1	2	3	4
1	0	0	0	0
2	1	0	0	0
3	2	0	0	0
4	3	0	0	0
5	0	1	0	0
6	1	1	0	0
7	2	1	0	0
8	0	2	0	0
9	1	2	0	0
10	0	3	0	0
11	0	0	1	0
12	1	0	1	0
13	2	0	1	0

Regresorii

- Matricea de regresori generată prin ridicarea la putere a matricei de semnale de intrare și ieșire
- Fiecare element din linie ridicat la puterea corespunzătoare din matricea de puteri
- Produsul acestor elemente = un element din matricea Phi de regresori

Regresorii

*exemplu pentru m=3 & na=nb=2



Identificare

```
for k = 1 : N
    for i = 1 : na + nb
         if i < na + 1
             if k - i > 0
                 matriceSemnale(k, i) = id.OutputData(k - i);
            else
                 matriceSemnale(k, i) = 0;
             end

         else
            if k - i + na > 0
                matriceSemnale(k, i) = id.InputData(k - i + na);
            else
                matriceSemnale(k, i) = 0;
            end
         end
    end
    for i = 1 : size(matricePuteri, 1)
        Phi(k, i) = prod(matriceSemnale(k, :) .^ matricePuteri(i, :));
    end
end
if m==1
    Phi(:,1)=0;
end
Theta = Phi \ id.OutputData;
```

- Bazată pe structura NARX
- Matricea de regresori realizată în funcție de intrările și ieșirile de identificare furnizate
- Succedată de aflarea parametrilor theta, necesari predicției & simulării
- Condiționată pentru gradul polinomial 1 : termenul liber nul (altfel, sistem afin)

PREDICȚIE

Metoda I

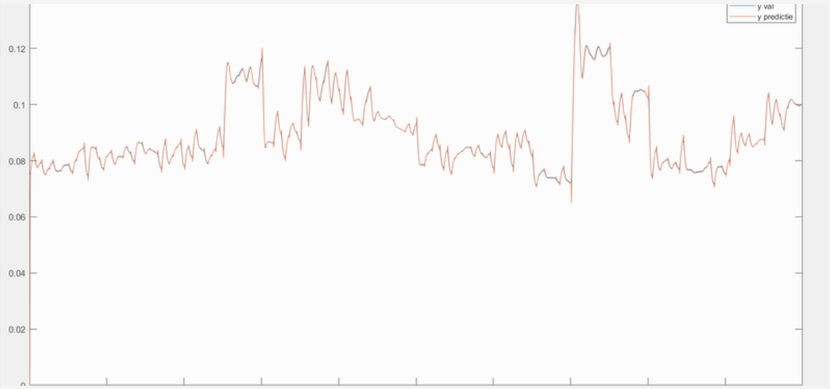
Predicție

```
for k = 1 : N
    for i = 1 : na + nb
         if i < na + 1
             if k - i > 0
                 matriceSemnale(k, i) = val.OutputData(k - i);
            else
                 matriceSemnale(k, i) = 0;
             end

         else
            if k - i + na > 0
                matriceSemnale(k, i) = val.InputData(k - i + na);
            else
                matriceSemnale(k, i) = 0;
            end
         end
    end
    for i = 1 : size(matricePuteri, 1)
        Phi(k, i) = prod(matriceSemnale(k, :) .^ matricePuteri(i, :));
    end
end
if m==1
    Phi(:,1)=0;
end
y_hat_predictie=Phi*Theta;
```

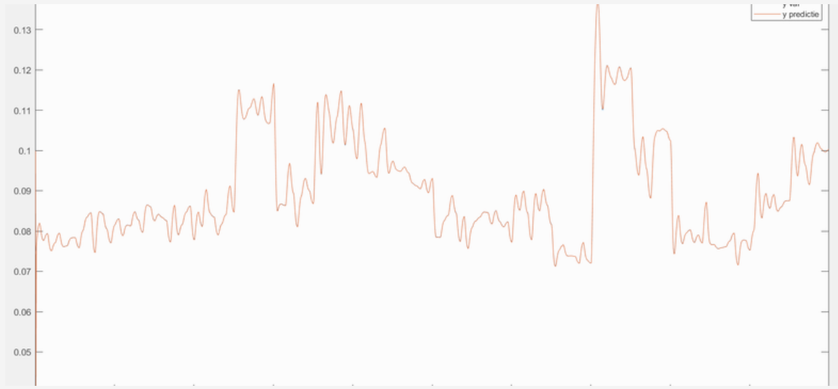
- Bazată pe structura NARX
- Matricea de regresori realizată în funcție de intrările si ieșirile de validare furnizate
- Ieșirea prezisă - utilizând regresorii și parametrii Theta aflați prin identificare
- Condiționată pentru gradul polinomial 1 : termenul liber nul (altfel, sistem afin)

PREDICȚIE

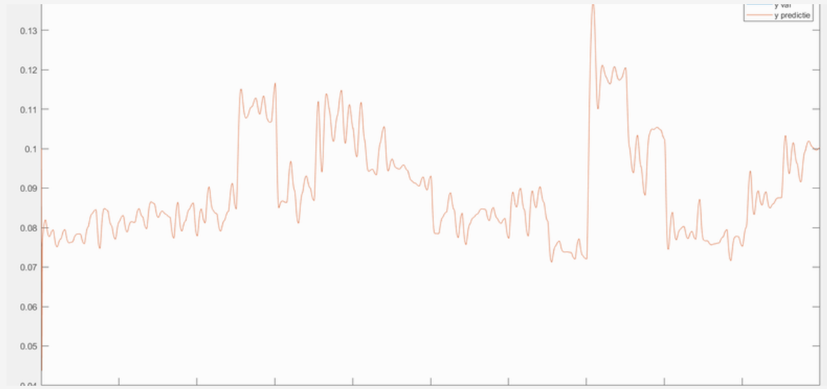
- construită pe baza valorilor deja cunoscute, de validare
- precizie sporită, eroare aproape nulă



GRADUL 1



GRADUL 2



GRADUL 3

SIMULARE

Metoda II

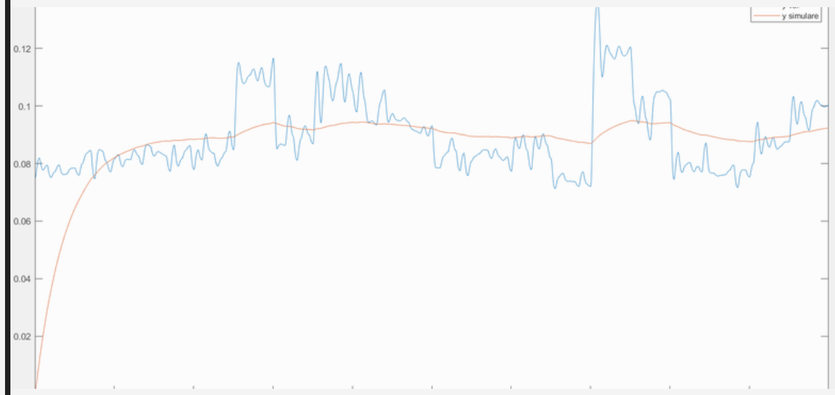
Simulare

- Bazată pe structura NARX
- Matricea de regresori realizată în funcție de intrările de validare furnizate și ieșiri simulate cu un pas înainte
- Ieșirea simulata - utilizând regresorii de pe fiecare linie construită și parametrii theta aflați prin identificare
- Condiționată pentru gradul polinomial 1 : termenul liber nul (altfel, sistem afin)

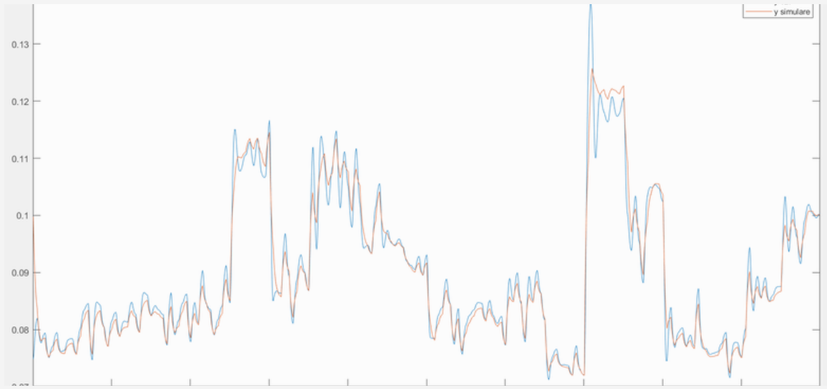
```
for k = 1 : N
    for i = 1 : na + nb
         if i < na + 1
             if k - i > 0
                 matriceSemnales(k, i) = y_hat_simulare(k - i);
            else
                 matriceSemnales(k, i) = 0;
             end

         else
            if k - i + na > 0
                matriceSemnales(k, i) = val.InputData(k - i + na);
            else
                matriceSemnales(k, i) = 0;
            end
         end
    end
    for i = 1 : size(matricePuteri, 1)
        Phis(k, i) = prod(matriceSemnales(k, :) .^ matricePuteri(i, :));
    end
    if m==1
    Phis(:,1)=0;
    end
    y_hat_simulare(k)=Phis(k,:)*Theta;
end
```

SIMULARE

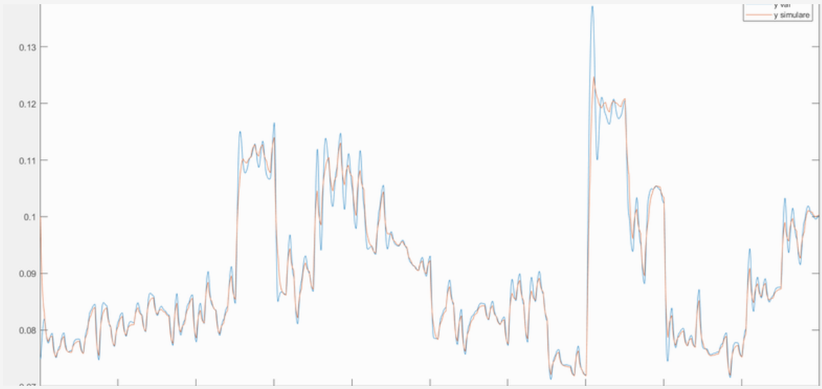
- construită pe bază de valori necunoscute ale ieșirii & cunoscute deja ale intrării , de validare
- precizie diminuată in comparație cu metoda predicției, dar și în funcție de grad



GRADUL 1



GRADUL 2



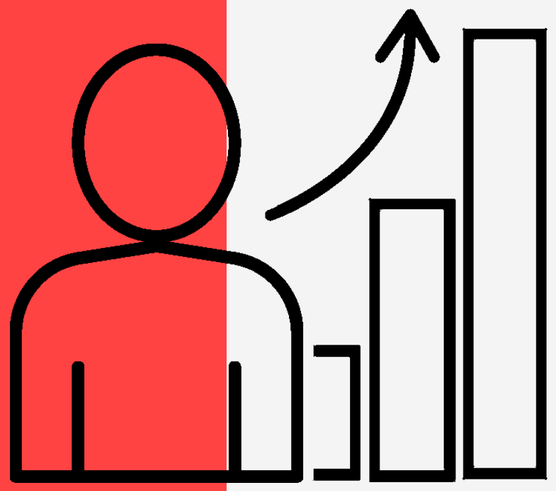
GRADUL 3

3.REZULTATE

Rezultate

Eroarea medie pătratică:

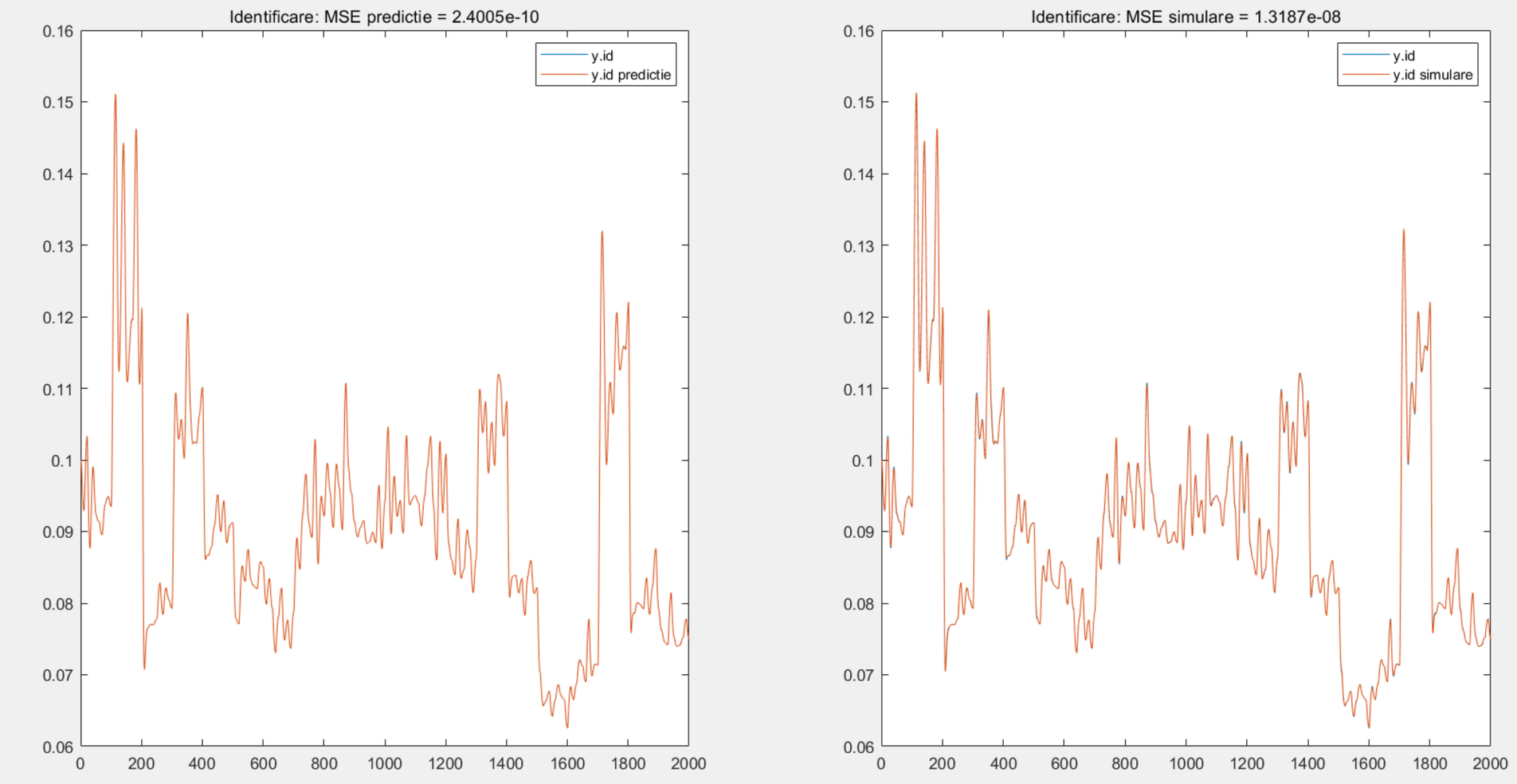
	m=1		m=2		m=3	
	Predicție	Simulare	Predicție	Simulare	Predicție	Simulare
na=nb=1	3.6412e-06	2.1011e-04	7.0993e-07	6.4266e-06	6.9317e-07	5.3739e-06
na=nb=2	3.1888e-06	4.0355e-04	9.2914e-07	NaN	8.1507e-07	1.3144e-06
na=nb=3	3.1610e-06	4.0387e-04	2.9765e-06	NaN	1.9938e-05	NaN



4.GRAFICE

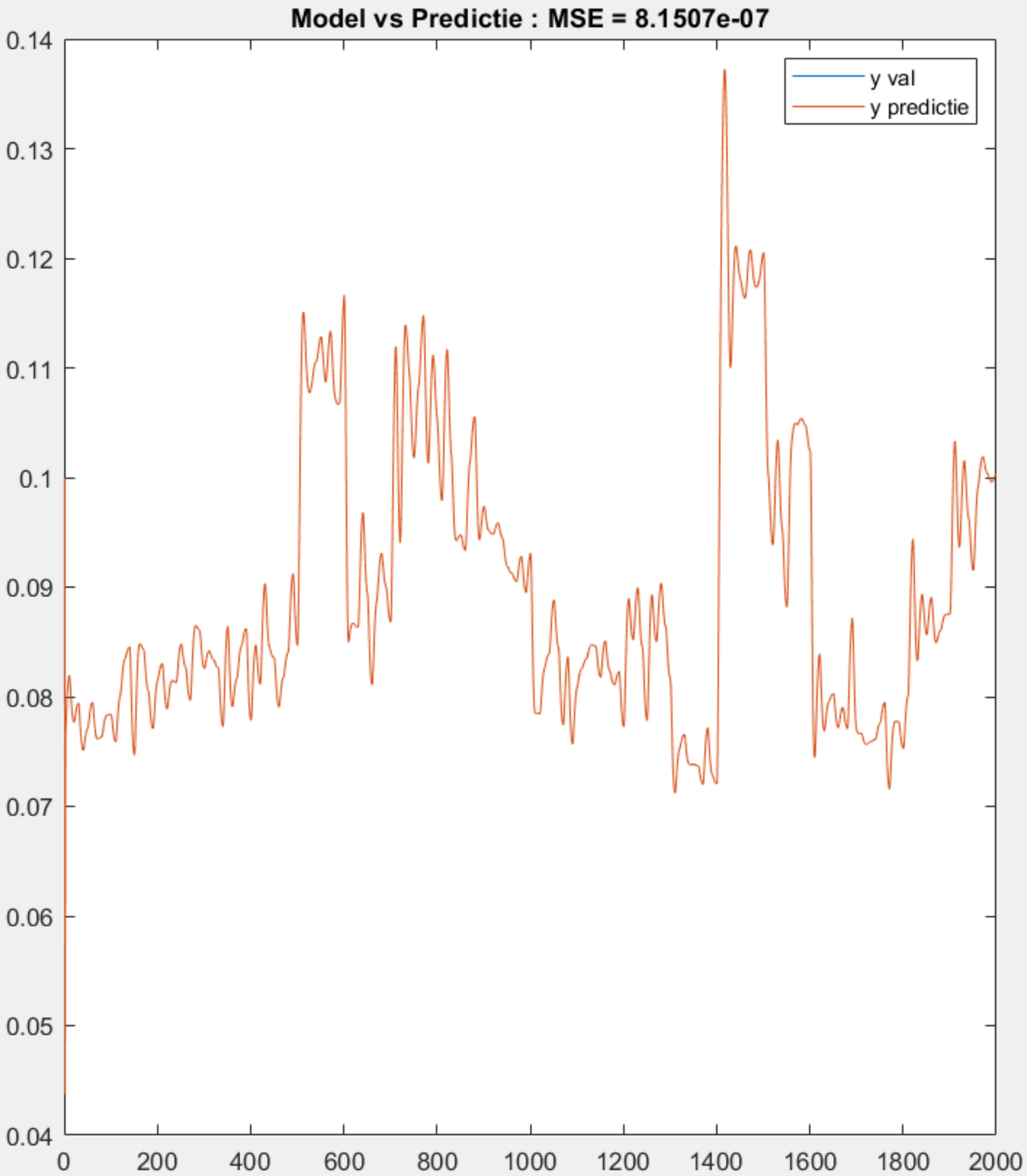
Identificare

m=3;
na=nb=2;
nk=1;



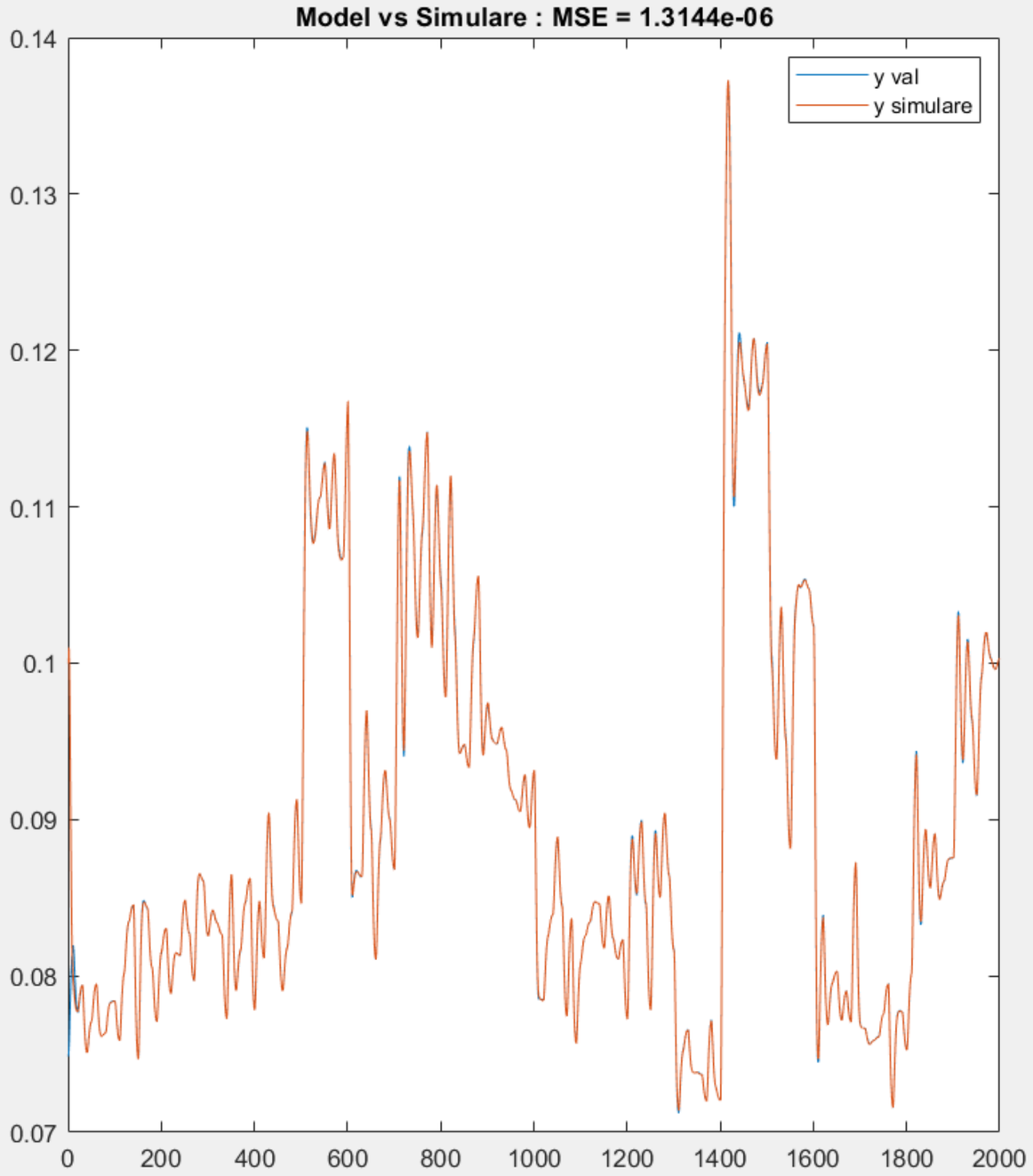
Predicție

m=3;
na=nb=2;
nk=1;



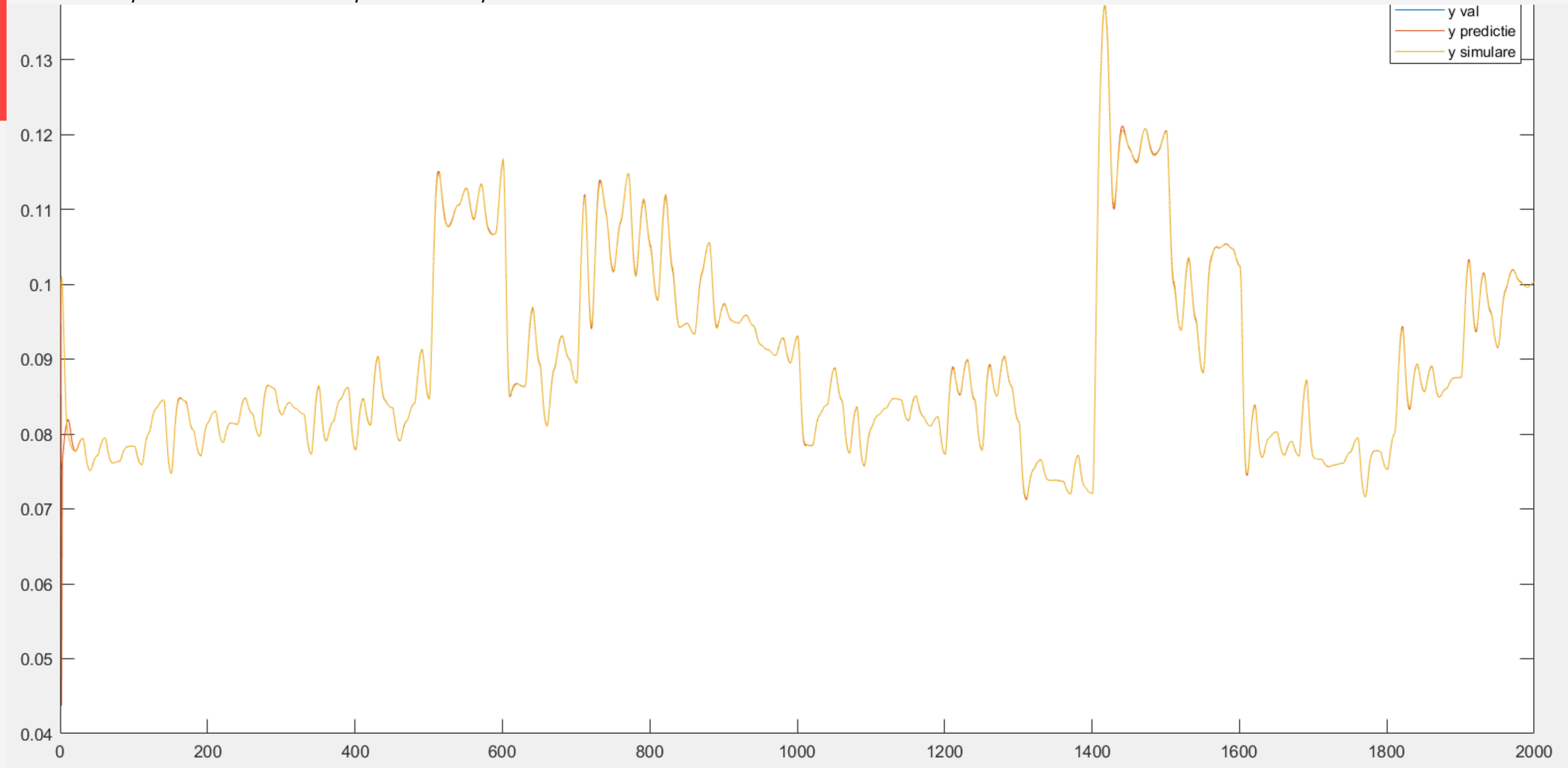
Simulare

m=3;
na=nb=2;
nk=1;



Model+Predicție +Simulare

m=3; na=nb=2; nk=1;



5.CONCLUZII

CONCLUZII GRADUL I

- na & nb
  - eroarea de predicție invers proporțională cu ordinele NARX
  - eroarea de simulare direct proporțională cu ordinele mici NARX

- predicție VS simulare
  - eficiența predicției crescută comparativ cu aceea a simulării

CONCLUZII GRADUL II

na & nb

- eroarea de predicție direct proporțională cu ordinele NARX
- sistem stabil doar pentru na=nb=1

predicție VS simulare

- eficiența predicției crescută comparativ cu aceea a simulării

CONCLUZII GRADUL III

- na & nb
  - eroarea de predicție direct proporțională cu ordinele NARX
  - eroarea de simulare invers proporțională cu ordinele NARX, pentru na=nb=1 și na=nb=2
  - simulare instabila pentru ordine NARX >=3
- predicție VS simulare
  - eficiența predicției crescută comparativ cu aceea a simulării

CONCLUZII GENERALE

na & nb

- eroarea de predicție direct proporțională cu ordinele NARX
- eroarea de simulare invers proporțională cu ordinele NARX mici
- simulare instabila pentru ordine mari NARX

predicție VS simulare

- eficiența predicției crescută comparativ cu aceea a simulării

m - gradul polinomial

- cele mai mici erori pentru m maximal
- cele mai mari erori pentru minimal

MULȚUMIM PENTRU ATENȚIE!