Proiect IDENTIFICAREA SISTEMELOR

Partea 1. Modelarea unei functii necunoscute

Băhnărel Cristian-Dumitru (g 30135)

Lucaci Laurențiu (g 30135)

Data: 17 Noiembrie 2018

Cuprins

1. Introducere

Acesta este un raport pentru cursul 'Identificarea Sistemelor'. In acesta vom studia problema modelarii unei functii necunoscute. Codul folosit pentru rezolvarea acestei probleme va fi scris in MATLAB.

Acest raport contine urmatoarele sectiuni:

- Descrierea matematica a metodei de rezolvare folosite (regresia liniara)

- Explicarea functiilor folosite pentru rezolvarea problemei (MATLAB)

- Anexa: in anexa se afla intreg codul utilizat + grafice

2. Metoda de modelare - Regresia liniara

Se considera o functie neliniara fixa a carei model matematic este descris de datele de intrare y, x1 si x2.

Obiectivul este acela de a gasi valorile parametrilor a_i, astfel incat $g(x_k) \approx y_k$, $(\forall)k=1,\dots,N$, $N>>n$ (se va alege preferabil un ordin mai mare, deoarece, pentru $N=n$, va exista prea mult zgomot). Astfel, la inceput se va construi sistemul supradeterminat :

$$k=1 : \varphi_1(x_1)\,\theta_1 + \varphi_2(x_1)\,\theta_2 + \dots + \varphi_n(x_1)\,\theta_n = y_1$$

$$k=2 : \varphi_1(x_2)\,\theta_1 + \varphi_2(x_2)\,\theta_2 + \dots + \varphi_n(x_2)\,\theta_n = y_2$$

....

$$k=3 : \varphi_1(x_N)\,\theta_1 + \varphi_2(x_N)\,\theta_2 + \dots + \varphi_n(x_N)\,\theta_n = y_N$$

Acest sistem va genera o eroare: $\varepsilon_k = y_k - g(x_k) = y_k - \varphi_1(x_k)\,\theta_1 - \varphi_2(x_k)\,\theta_2 - \dots - \varphi_n(x_k)\,\theta_n$, care la randul ei va genera o eroare medie patratica (MSE- mean squared error) de:

MSE = 1/N $\sum_{k=}^{N} \boldsymbol{\varepsilon_k}$.

Sistemul sub forma matriceala va arata :

$$\underbrace{\begin{bmatrix} \varphi1(1) & \cdots & \varphi n(1) \\ \vdots & \ddots & \vdots \\ \varphi1(N) & \cdots & \varphi n(N) \end{bmatrix}}_{\phi} \underbrace{\begin{bmatrix} \theta1 \\ \vdots \\ \theta n \end{bmatrix}}_{\theta} = \underbrace{\begin{bmatrix} y(1) \\ \vdots \\ y(N) \end{bmatrix}}_{Y}$$

Obiectivul prezent va fi gasirea vectorului de parametri. Pentru aceasta se va folosi operatorul "\" (backslash) din MATLAB: $\boldsymbol{\theta = \phi \backslash Y}$. Backslash-ul va utiliza un algoritm ideal pentru realizarea operatiei matematice: $\phi^T \phi\, \theta = \phi^T Y$ -> $\underline{\boldsymbol{\theta = (\phi^T \phi)^{-1} \phi^T Y}}$, si astfel va determina θ pentru cel mai mic MSE. Deci, se va folosi metoda regresiei liniare pe datele de identificare si se va determina θ-ul , iar dupa aceea se va verifica rezultatul obtinut pe datele de validare.

3. Interpretarea rezultatelor

Pentru inceput se face memorarea datelor in variabile si se afiseaza un grafic (Anexa fig. 1) cu aceste date initiale. Matricea Y este transformata intr-un vector care va avea lungimea N*N. Acesti vectori au fost creati pentru a ne fi mai usor la partea de generare a polinomului.

Generarea polinomului de aproximare se realizeaza dupa urmatorul algoritm, care consta in creearea matricii PHI. Fiecare linie este construita prin adaugarea regresorilor, la puteri succesive pana la gradul m (**x_1 x_2 x_1^2 x_2^2 .... x_1^m x_2^m**), urmand sa fie adaugate totalitatea combinarilor perechilor de regresori la puteri diferite, cu conditia ca suma puterilor sa nu depaseasca gradul m (**$x_1^i x_2^j$, $\forall$ i,j $\in(0,m)$, i+j<=m**), pentru a se respecta forma generala a polinomului. Dupa ce o linie este construita, aceasta se pune in matricea PHI.

In continuare vom antrena aproximatoare, schimband treptat gradul polinomului (m'ul) pentru a afla cea mai mica eroare medie patratica. Acest algoritm va viza doar datele de validare.

grad m	**MSE Identificare**	**MSE Validare**
3	0.178911619632224	0.179482713703208
7	0.035410826443806	0.034381402928349
11	0.008807416639480	0.008688504931161
15	0.005831504044523	0.005927652402558
18	0.004926316794313	0.005277088723184
21	***0.004556500440002***	***0.005262965886321***
26	0.003941203039659	0.006959515436563
30	0.004313911072246	0.015478423227222

Tabel 1

Conform tabelului 1, se poate observa ca pentru m=21, se obtine cea mai mica valoare a MSE-ului de validare, lucru vazut grafic si in figurile 2a si 2b din Anexa. Cu toate ca eroarea la validare este minima la gradul 21, se poate observa ca la identificare aceasta valoare inca poate sa scada deoarece apare fenomenul de supra-antrenare (ilustrat in figurile 3a si 3b din Anexa).

In fig 4a,b, 5a,b si 6a,b (gradele 3, 10 si 18) se poate observa incercarea antrenarii sistemului la diferite grade, dar aproximatorul polinomial nu modeleaza functia in modul cel mai favorabil, deoarece gradul nu este suficient de optim.

4. Anexa

4.1. codul MATLAB propriu-zis

```
clear
close all
clc
load('proj_fit_11.mat');
x=id.X;
x1=id.X{1,1};
x2=id.X{2,1};
y=id.Y;
surf(x1,x2,y); title('Datele initiale')

xflat=[];
x1flat=[];
for i=1:length(x1)
    x1flat=[x1flat,x1];
end
x2flat=[];
for i=1:length(x2)
    aux=[];
    for j=1:length(x2)
        aux=[aux,x2(i)];
    end
    x2flat=[x2flat,aux];
end

xflat=[x1flat;x2flat];
yflat=reshape(y,1,[]);

m=21;   % cel mai bun grad gasit

%---------Identificarea--------
phi=[];
for i=1:length(xflat)
x=1;
  for j=1:m
      x=[x,xflat(1,i)^j,xflat(2,i)^j];
  end
  for j=1:m
        for k=1:m
```

```
              if j+k<=m
              x=[x,xflat(1,i)^j*xflat(2,i)^k];
           end
         end
   end
ph=x;

phi=[phi;ph];
end

A=phi\yflat';
y_est= phi*A;
e=yflat'-y_est;
figure;
y_est_=reshape(y_est,length(x1),length(x2));
surf(x1,x2,y_est_);
MSE=1/length(e)*sum(e.^2);
title(['MSE Identificare=',num2str(MSE)]);

%------------Validarea-----------
xval=val.X;
x1val=val.X{1,1};
x2val=val.X{2,1};
yval=val.Y;

xflatval=[];
x1flatval=[];
for i=1:length(x1val)
    x1flatval=[x1flatval,x1val];
end
x2flatval=[];
for i=1:length(x2val)
    aux=[];
    for j=1:length(x2val)
        aux=[aux,x2val(i)];
    end
    x2flatval=[x2flatval,aux];
end
```

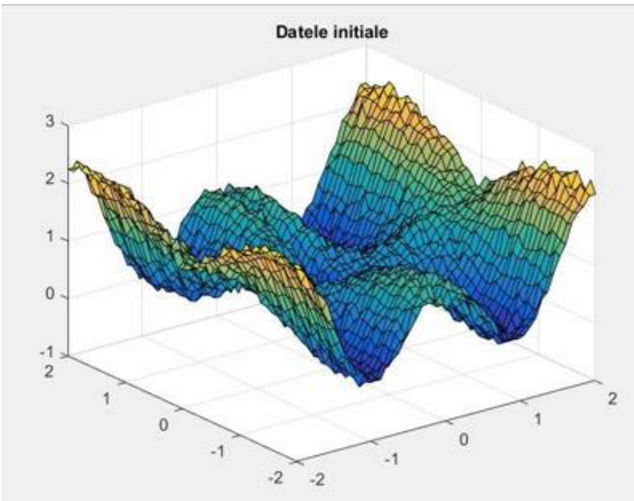
```
xflatval=[x1flatval;x2flatval];
yflatval=reshape(yval,1,[]);

phival=[];
for i=1:length(xflatval)
x=1;
  for j=1:m
          x=[x,xflatval(1,i)^j,xflatval(2,i)^j];
          end
  for j=1:m
        for k=1:m
          if j+k<=m
              x=[x,xflatval(1,i)^j*xflatval(2,i)^k];
          end
        end
  end

phval=x;
phival=[phival;phval];
end

y_estval= phival*A;
eval=yflatval'-y_estval;
figure;
y_estval_=reshape(y_estval,length(x1val),length(x2val));
surf(x1val,x2val,y_estval_);
MSEval=1/length(eval)*sum(eval.^2);
title(['MSE Validare=',num2str(MSEval)]);
```

4.2. Grafice



..

Fig. 1

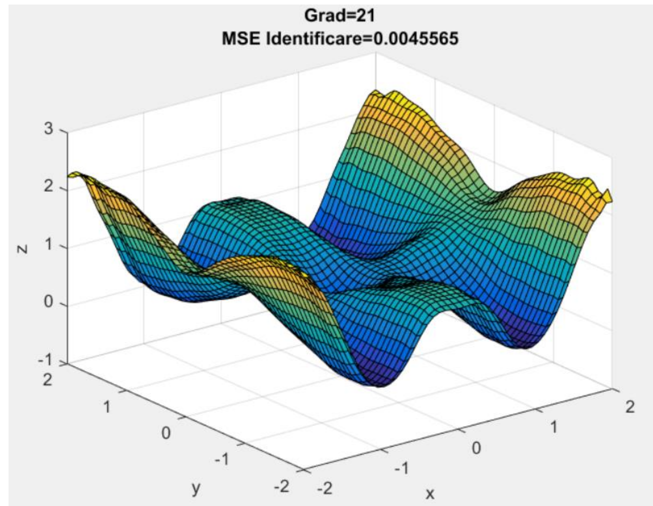


Fig. 2 a)

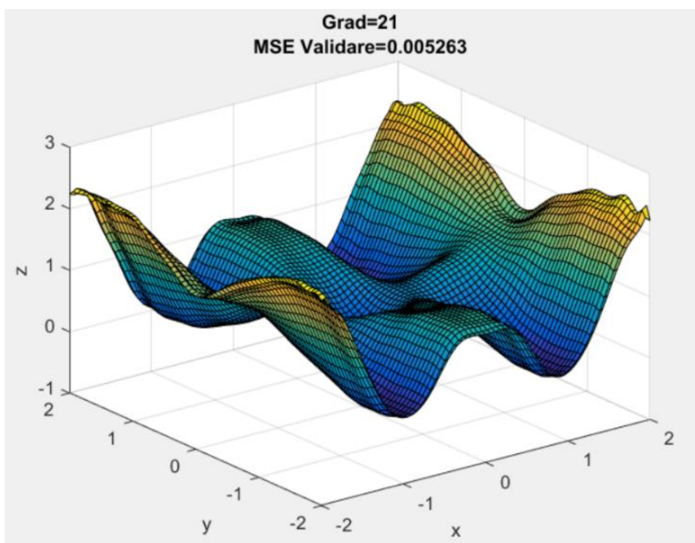


Fig. 2 b)

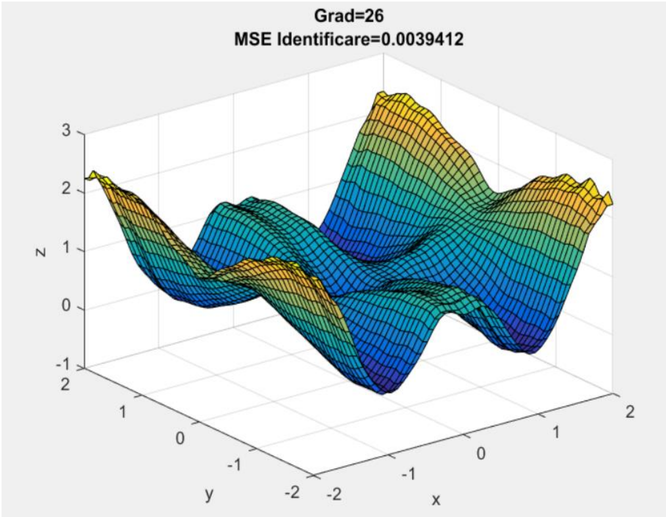


Fig 3 a)

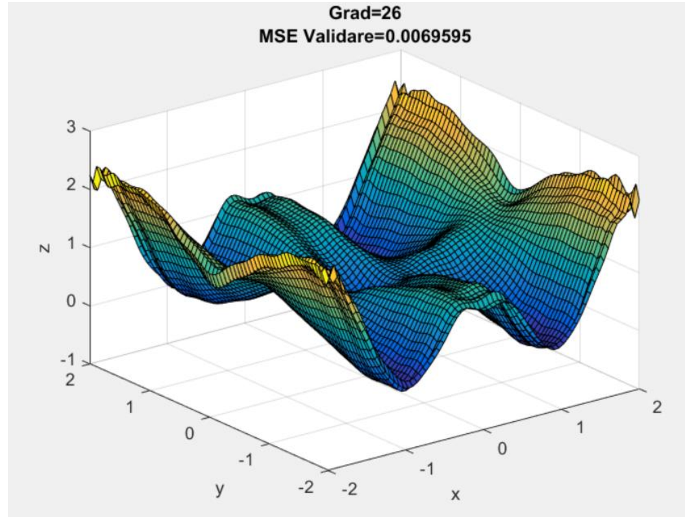


Fig 3 b)

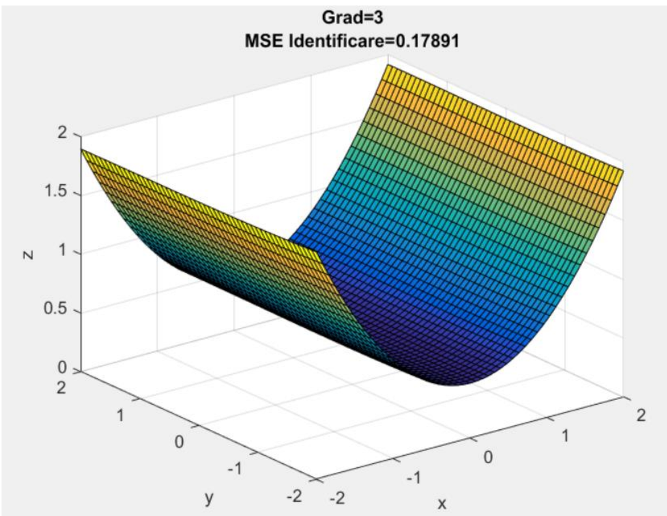


Fig 4 a)

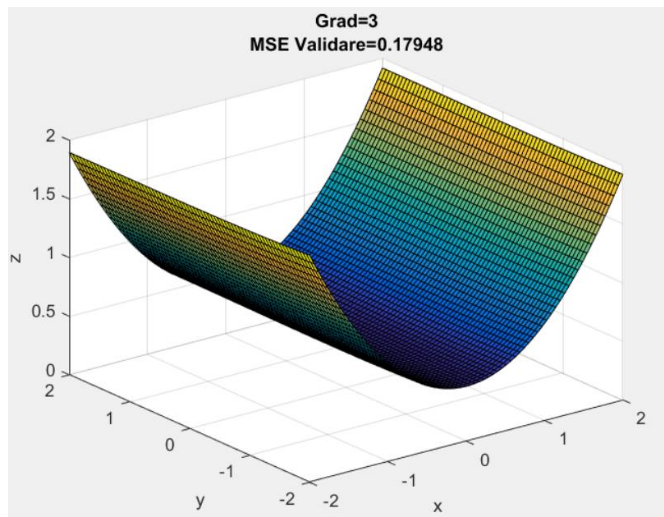


Fig 4 b)

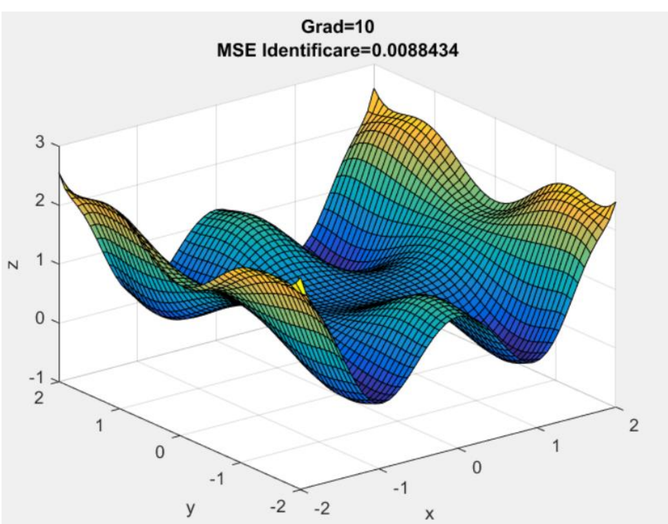


Fig 5 a)

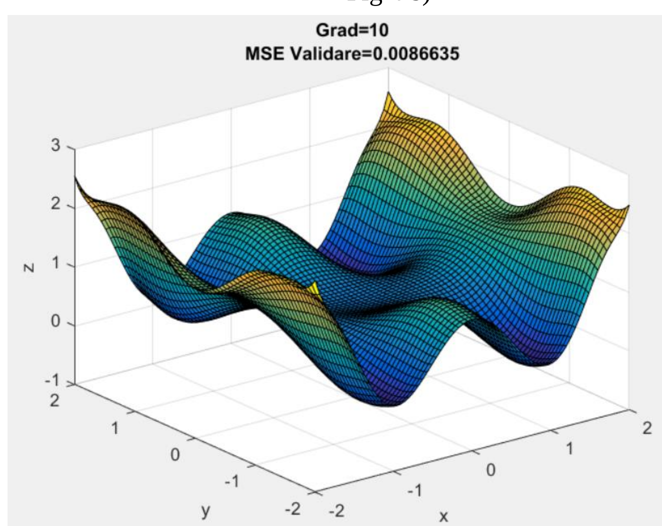


Fig 5 b)

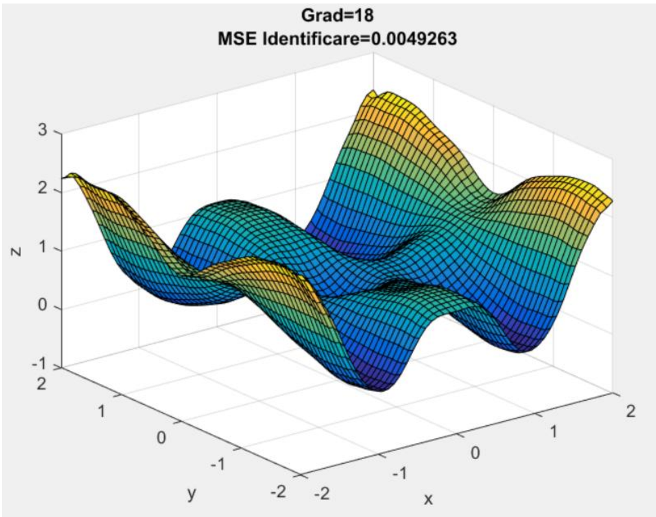

Grad=18
MSE Identificare=0.0049263
z
y
x

Fig 6 a)

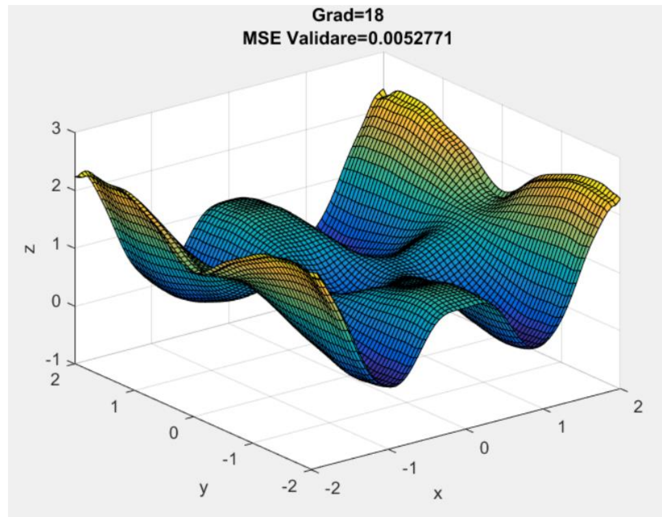

Grad=18
MSE Validare=0.0052771
z
y
x

Fig 6 b)

.

<u>Proiect IDENTIFICAREA SISTEMELOR</u>

Partea 2. ARX neliniar

Băhnărel Cristian-Dumitru (g 30135)

Lucaci Laurențiu (g 30135)

Data: 13 Ianuarie 2019

Cuprins

1. Introducere

Acesta este un raport pentru cursul 'Identificarea Sistemelor'. In acesta vom studia problema identificarii sistemelor dinamice de tip cutie neagra folosind metodat ARX neliniar. Codul folosit pentru rezolvarea acestei probleme va fi scris in MATLAB.

Acest raport contine urmatoarele sectiuni:

- Descrierea matematica a metodei de rezolvare folosite (metoda ARX)
- Explicarea functiilor folosite pentru rezolvarea problemei (MATLAB)
- Anexa: in anexa se afla intreg codul utilizat

2. Metoda de modelare - ARX neliniar

Se considera un sistem dinamic de tip SISO al carui model matematic este descris de datele de intrare u si y. Acest sistem are gradul m si ordinele na, nb si intarzierea nk (considerata 1). Metoda ARX presupune calcularea iesirii y, la un moment k, din intrarile u si iesirile la momente precedente:

$y(k) = p(y(k - 1),...,y(k - na),u(k - nk),u(k - nk - 1),...,u(k - nk - nb + 1)) = p(d(k))$

, unde p este un polinom de grad m format dintr-un vector de intrari si iesiri intarziate.

$\varphi(k) = [-y(k -1),...,-y(k -na),u(k -1),...,u(k -nb)]^T$ si reprezinta vectorul de regresori, adica elementele pe baza caruia se genereaza polinomul mentionat anterior, iar coeficientii polinoamelor se gasesc in matricea teta.

$$\begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} = \begin{bmatrix} -y(0) & \cdots & -y(1-na) & u(0) & \cdots & u(1-nb) \\ -y(1) & \cdots & -y(2-na) & u(1) & \cdots & u(2-nb) \\ \vdots & \vdots & \vdots & \vdots & & \\ -y(N-1) & \cdots & -y(N-na) & u(N-1) & \cdots & u(N-nb) \end{bmatrix} \cdot \theta$$

Scopul nostru este de afla parametrii teta pe baza datelor u si y.

$y(k)=\varphi^T(k)*\theta+ \varepsilon(k)$. $\varepsilon(k)$ - reprezinta eroarea.

Aceasta genereaza o eroare medie patratica. Obiectivul urmator este de a minimiza eroarea medie patratica

3. Interpretarea rezultatelor

3.1 Explicarea codului MATLAB

In prima parte a codului MATLAB are loc memorarea datelor si afisarea unui grafic pe datele initiale (fig 1).

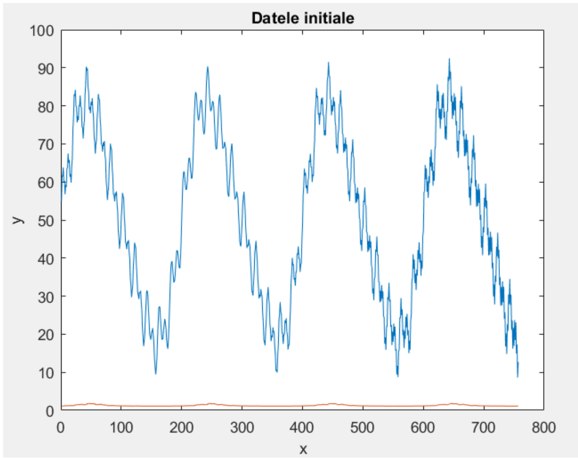


fig 1.

Urmatorul pas este crearea matricei de regresori ai polinomului. Aceasta se realizeaza prin adaugarea datelor y si a datelor u, la puteri succesive pana la gradul maxim, pe o linie, urmand ca linia sa fie alipita intr-o matrice numita "matrice_grad".

Dupa ce matricea_grad a fost construita in totalitate, se realizeaza o matrice "GR", in care se afla gradul fiecarui element de pe fiecare pozitie din matricea_grad. In continuarea programului se incearca conceperea unei matrici cu totalitatea combinarilor de elemente dintr-o linie, fara cele la gradul maxim. Acest algoritm consta in extragerea fiecarei linii din matricea de grade, dar se omit elementele aflate la gradul maxim. Dupa extragerea liniei, are loc generearea unei matrici de combinari ce contine totalitatea combinarilor de grad 2, pana la gradul m. Pentru inceput se iau combinarile de ordin 2, urmand ca acest ordin sa creasca. Aceste combinari se fac cu ajutorul functiei ***combntns***. Fiecare linie din matricea combinarilor ne da combinarea ce trebuie sa se faca. Apoi, se creeaza toate combinarile posibile dintre elementele liniei extrase.

Combinarile se realizeaza prin inmultirea tuturor elementelor din linie aflate pe pozitiile indicate in matricea de combinari si se extrage gradul elementului din matricea de grade si se adauga intr-o variabila numita grad. Scopul acestei variabile este de a verifica daca gradul elementelor combinarii sa nu fie mai mare decat gradul m. Daca aceasta conditie se indeplineste, combinarea se adauga in matricea de elemente combinate. Matricea finala se formeaza prin lipirea matricei ce contine elementele la diferite grade fara elementele la grad maxim, cu matricea elementelor combinate.

3.1 Predictia

Pentru partea de predictie, s-a creat matricea de regresori pe baza datelor initiale ale sistemului. Vectorul de predictie a fost creat cu ajutorul matricei de regresori si a vectorului de coeficienti calculati la pasii precedenti.

La metoda predictiei, s-au facut calculele pentru setul de date de identificare (fig 2. a)), iar apoi in acelasi mod, pentru setul de date de validare (fig 2. b)).

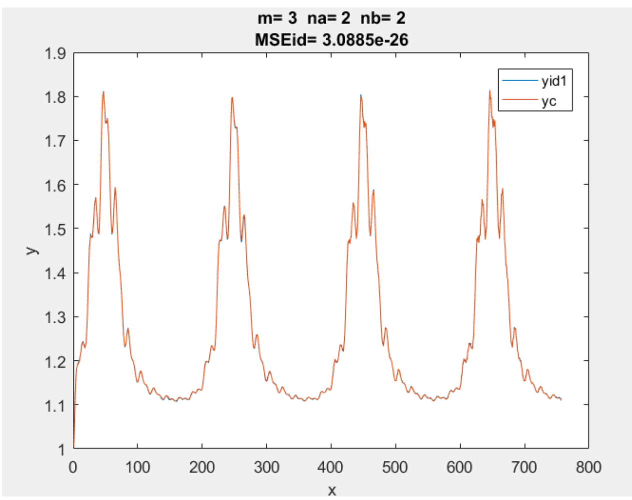


fig 2 a)

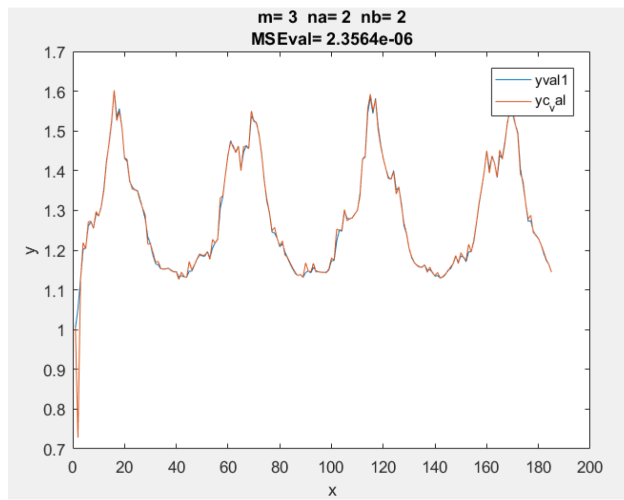


fig 2 b)

In urmatoarele figuri se poate observa incercarea antrenarii sistemului cu mai multe intrari si un grad mai mare. In aceasta situatie, apare fenomenul de supraantrenare, iar la modelarea datelor, datorita zgomotului, aproximarea nu va fi exacta (fig 3 a), b)).

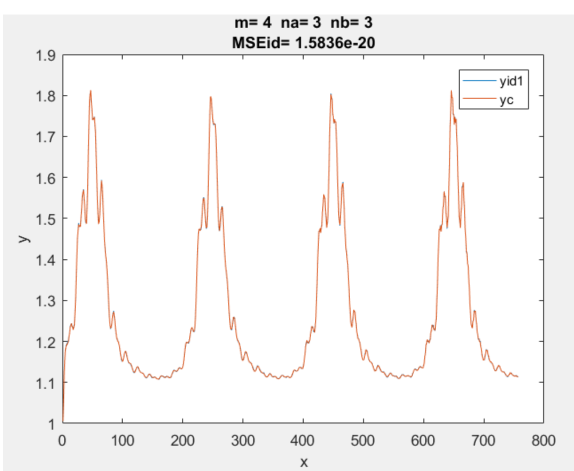


fig 3 a)

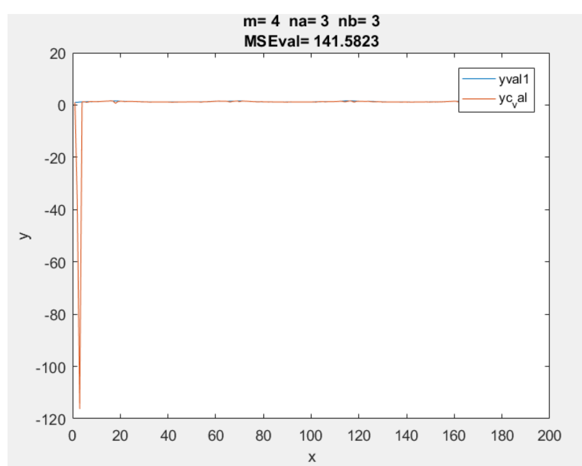


fig 3 b)

3.2 Simularea

La simulare, datele de iesire sunt necunoscute si ele trebuiesc inlocuite cu iesirile calculate la simulare la pasii anteriori. Se foloseste aceeasi metoda ca la simulare, doar ca acum vectorul de intarzieri contine iesirile calculate anterior si intrarile sistemului. Acest algoritm se aplica, la fel ca cel mentionat la punctul 3.1, atat pentru setul de date de identificare, cat si pentru setul de date de validare.

4. Anexa

```
clc; clear all; close all
load('iddata-03');
uid=id.u;
yid=id.y;
yval=val.y;
uval=val.u;
% plot([uid,yid]);
% title('Datele initiale');
% xlabel('x');
% ylabel('y');
na=1;
nb=2;
m=3;
yid1=yid;
yid=[zeros(1,na+nb),yid'];
uid=[zeros(1,na+nb),uid'];
psi=[];
L=[];
Matr_y=[];
Matr_u=[];
linie=[];
matrice_grad=[];
  for i=1+na+nb:length(yid)
    linie=[];
    Matr_y=[];
    Matr_u=[];
    for j=1:na
      Matr_y=[Matr_y yid(i-j)];
    end
    for j=1:nb
      Matr_u=[Matr_u uid(i-j)];
    end
    linie=[Matr_y Matr_u];
    matrice_grad=[matrice_grad;linie];
  end
  aux=matrice_grad;
  if m>1
    for i=2:m
      z=aux.^i;
      matrice_grad=[matrice_grad z];
```

```
      end
   end
aux_final=matrice_grad;
matrice_combinari=[];
GR=ones((na+nb)*m,1);
for i=2:m
   GR((na+nb)*(i-1)+1:(na+nb)*i)=i;
end
MATR_comb=[];
for i=1:length(yid1)
      x=aux_final(i,1:length(aux_final(1,:))-length(linie));
      linie_combinari=[];
      for var=2:m
         COMB_gri=combntns(1:length(x),var);
         clc
         for k=1:length(COMB_gri)
            produs=1;
            grad=0;
            for j=1:var
               produs=produs*x(COMB_gri(k,j));
               grad=grad+GR(COMB_gri(k,j));
            end
            if grad<=m
               linie_combinari=[linie_combinari, produs];
            end
         end
      end
      MATR_comb=[MATR_comb;linie_combinari];
end
matr_grad=[];
for i=1:length(matrice_grad)
   matr_grad = [matr_grad; matrice_grad(i,1:(na+nb)*(m-1))];
end

matrice_finala=[matr_grad MATR_comb];
q=ones(length(yid1),1);
matrice_finala=[q matrice_finala];

%------------------identificarea------------------

psi=matrice_finala;
teta=psi\yid1;
yc=psi*teta;
eid=yid1-yc;
```

```
MSEid=1/length(eid)*sum(eid).^2;
plot([yid1,yc]);
legend('yid1','yc')
title({['m= ',num2str(m),'  na= ',num2str(na),'  nb= ',num2str(nb)];['MSEid= ',num2str(MSEid)]});
xlabel('x');
ylabel('y');
figure;

%--------------------validarea-------------------

yval1=yval;
yval=[zeros(1,na+nb),yval'];
uval=[zeros(1,na+nb),uval'];

psi_val=[];
L_val=[];
Matr_y_val=[];
Matr_u_val=[];
linie_val=[];
matrice_grad_val=[];
  for i=1+na+nb:length(yval)
    linie_val=[];
    Matr_y_val=[];
    Matr_u_val=[];
    for j=1:na
      Matr_y_val=[Matr_y_val yval(i-j)];
    end
    for j=1:nb
      Matr_u_val=[Matr_u_val uval(i-j)];
    end
    linie_val=[Matr_y_val Matr_u_val];
    matrice_grad_val=[matrice_grad_val;linie_val];
  end
  aux_val=matrice_grad_val;
  if m>1
    for i=2:m
      z=aux_val.^i;
      matrice_grad_val=[matrice_grad_val z];
    end
  end

aux_final_val=matrice_grad_val;
matrice_combinari_val=[];
GR_val=ones((na+nb)*m,1);
```

```
for i=2:m
   GR_val((na+nb)*(i-1)+1:(na+nb)*i)=i;
end

MATR_comb_val=[];
for i=1:length(yval1)
      x=aux_final_val(i,1:length(aux_final_val(1,:))-length(linie_val));
      linie_combinari_val=[];
      for var=2:m
         COMB_gri_val=combntns(1:length(x),var);
         clc;
         for k=1:length(COMB_gri_val)
            produs=1;
            grad=0;
            for j=1:var
               produs=produs*x(COMB_gri_val(k,j));
               grad=grad+GR_val(COMB_gri_val(k,j));
            end
            if grad<=m
               linie_combinari_val=[linie_combinari_val, produs];
            end
         end
      end
      MATR_comb_val=[MATR_comb_val;linie_combinari_val];
end

matr_grad_val=[];
for i=1:length(matrice_grad_val)
   matr_grad_val = [matr_grad_val; matrice_grad_val(i,1:(na+nb)*(m-1))];
end
matrice_finala_val=[matr_grad_val MATR_comb_val];
q=ones(length(yval1),1);
matrice_finala_val=[q matrice_finala_val];

psi_val=matrice_finala_val;
yc_val=psi_val*teta;
eval=yval1-yc_val;
MSEval=1/length(eval)*sum(eval).^2;
plot([yval1,yc_val]);
legend('yval1','yc_val')
title({['m= ',num2str(m),'  na= ',num2str(na),'  nb= ',num2str(nb)];['MSEval=
',num2str(MSEval)]});
xlabel('x');
ylabel('y');
```