Identificarea Sistemelor 2018-2019

Proiect 1

Modelarea unei funcții necunoscute folosind **metoda regresiei liniare**

Student: Călina Corneliu-Dumitru

CUPRINS

1. Introducere ..... pag. 3
2. Analiza metodei
    2.1. Prelucrarea datelor de identificare și validare..... pag. 4
    2.2. Crearea vectorului de regresori (grad ‘m’ variabil)..... pag. 5
    2.3. Calcularea vectorului de parametrii..... pag. 5
3. Rezultate obținute
    3.1. Prezentarea grafică a datelor inițiale..... pag. 6
    3.2. Analiză rezultate..... pag. 6-8

4. Anexa cod MATLAB..... pag. 8-10

1. Introducere

Proiectul realizat are în vedere găsirea unei funcții care aproximează un set de date format din două variabile de intrare și una de ieșire folosind metoda regresiei liniare.

Această metodă presupune formarea unei funcții ”aproximatoare” y(x1,x2) ≈ g(x1,x2). Datele vor fi împărțite într-un set de identificare pentru care vom găsi un vector de parametrii pentru regresorii funcției și vom căuta cel mai bun grad astfel încât aproximarea să fie cât mai precisă. Apoi, pe setul de validare vom aplica funcția găsită la pasul precedent pentru a determina eficiența metodei.

Modul în care am utilizat metoda regresiei liniare pentru a găsi funcția care aproximează setul de date este prezentat în continuare printr-o analiză a metodei și a codului realizat în MATLAB.

2. Analiza metodei

2.1. Prelucrarea datelor de identificare și validare

Setul de date pus la dispoziție este format din două variabile de intrare și una de ieșire. Având două variabile de intrare, se va crea o dependență între cele două intrări și ieșire prezentată în fig. 1:

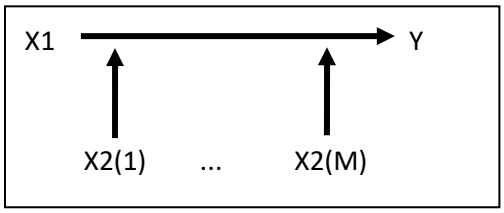

Fig. 1

- pentru fiecare valoare din vectorul X1 se vor forma asocieri cu toate cele M valori din vectorul X2. Rezultatul acestor asocieri vor contribui la realizarea ieșirii Y.

Din această cauză, vom prelucra datele astfel încât să avem această relație transpusă într-o matrice de 2x(MxM) (Xflat, fig. 2) pentru intrări și un vector de 1x(MxM) (Yflat, fig. 2) pentru ieșiri, unde M=lungimea vectorului X1. (pentru validare se procedează la fel, cu modificarea M=lungimea vectorului X1validare)

Fig. 2

```
X1flat=[];
for i=1:M
    x1linie=part1_fctforflat(M,x1(i));
    X1flat=[X1flat,x1linie];
end
X2flat=[];
for i=1:M
    for j=1:M
        X2flat=[X2flat,x2(j)];
    end
end
Xflat=[X1flat;X2flat];
Yflat=reshape(y,1,M*M);
```

Fig. 3

```
function [out] = part1_fctforflat(m, x_i )
    out=[];
    for i=1:m
        out=[out, x_i];
    end
end
```

'part1_fctforflat.m' are rolul de a crea M elemente care corespund valorii lui x1(i).

2.2. Crearea vectorului de regresori (grad 'm' variabil)

Fie g(x1,x2) aproximatorul pentru funcția y(x1,x2). Atunci:

$$y(x1,x2) \approx g(x1,x2) \quad (1)$$

Ecuația de mai sus (1) va fi simplificată cu ajutorul matricilor și modificărilor făcute precedent astfel:

$$\begin{bmatrix} Yflat(1) \\ . \\ . \\ . \\ Yflat(MxM) \end{bmatrix} = \begin{bmatrix} \rho_1(Xflat(:,1)) & \dots & \rho_k(Xflat(:,1)) \\ & . & \\ & . & \\ & . & \\ \rho_1(Xflat(:,M)) & \dots & \rho_k(Xflat(:,MxM)) \end{bmatrix} \begin{bmatrix} \theta_1 \\ \\ . \\ . \\ \theta_k \end{bmatrix}$$

unde k-numărul de regresori.

2.3. Calcularea vectorului de parametri

Fiind cunoscute Yflat și matricea Ψ formată din valorile regresorilor (prezentate mai sus), va rămâne de calculat parametrii θ = Ψ \ Yflat'. Acest procedeu se repetă pentru un interval al lui 'm' (exemplu: m=1:100) pentru a determina gradul optim (cel pentru care eroarea medie pătratică este minimă). În (fig. 4)
este prezentată o secțiune de cod MATLAB în care este construită matricea Ψ pe linii respectiv funcția ('fforpsi') care construiește regresorii propriu-ziși (fig. 5).

```
PSI=[];
for i=1:M*M
    L=part1_fforpsi(m,Xflat(:,i));
    PSI=[PSI;L];
end
```

Fig. 4

```
function [out]= part1_fforpsi(m,x)
 out=1;
   for i=1:m
       out=[out, x(1,1)^i, x(2,1)^i];
       for j=1:m
           if i+j<=m
               out=[out , (x(1,1)^j)*(x(2,1)^i)];
           end
       end
   end

end
```

Fig. 5

3. Rezultate obținute

3.1. Prezentarea grafică a datelor.

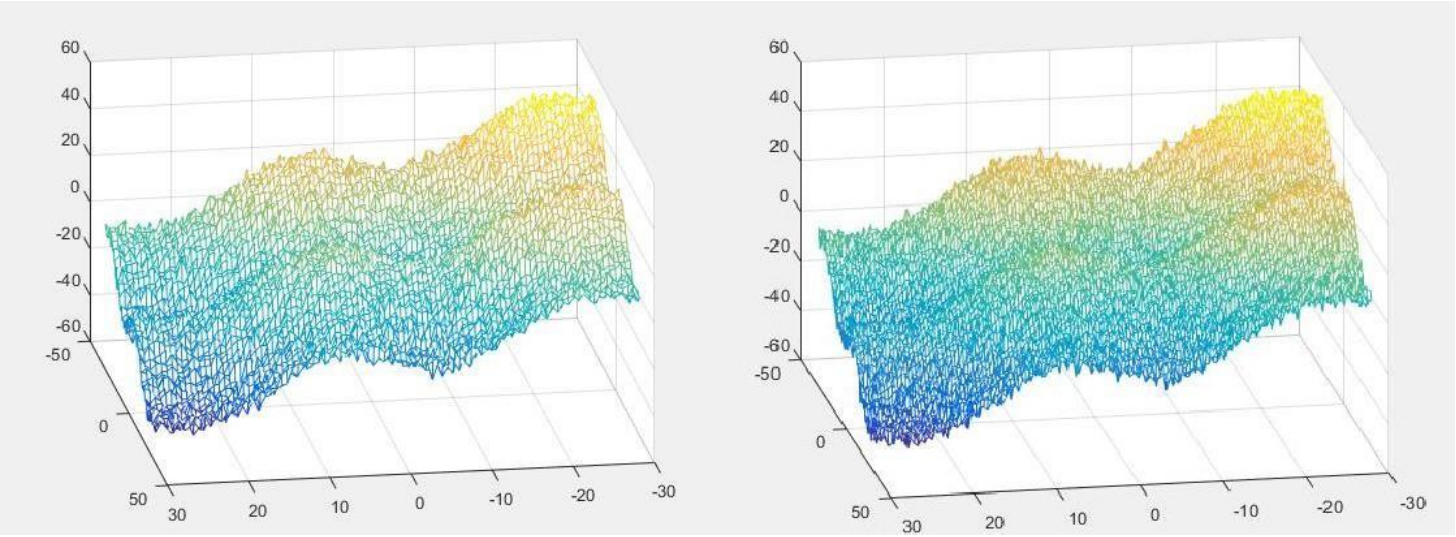

Fig. 6

În figura 6 sunt prezentate datele de identificare(stânga) și validare(dreapta).

3.2. Analiza rezultatelor

Având implementată metoda în MATLAB, vom realiza și o analiză a rezultatelor obținute. Ideea principală este cea de a obține o eroare medie pătratică (MSE) cât mai mică, așa că vom executa programul pentru un grad ‘m’ variabil pentru a observa evoluția acesteia și a detecta gradul unde MSE-ul va fi minim.

În figura 7 este prezentată evoluția MSE-ului în funcție de gradul polinomului.

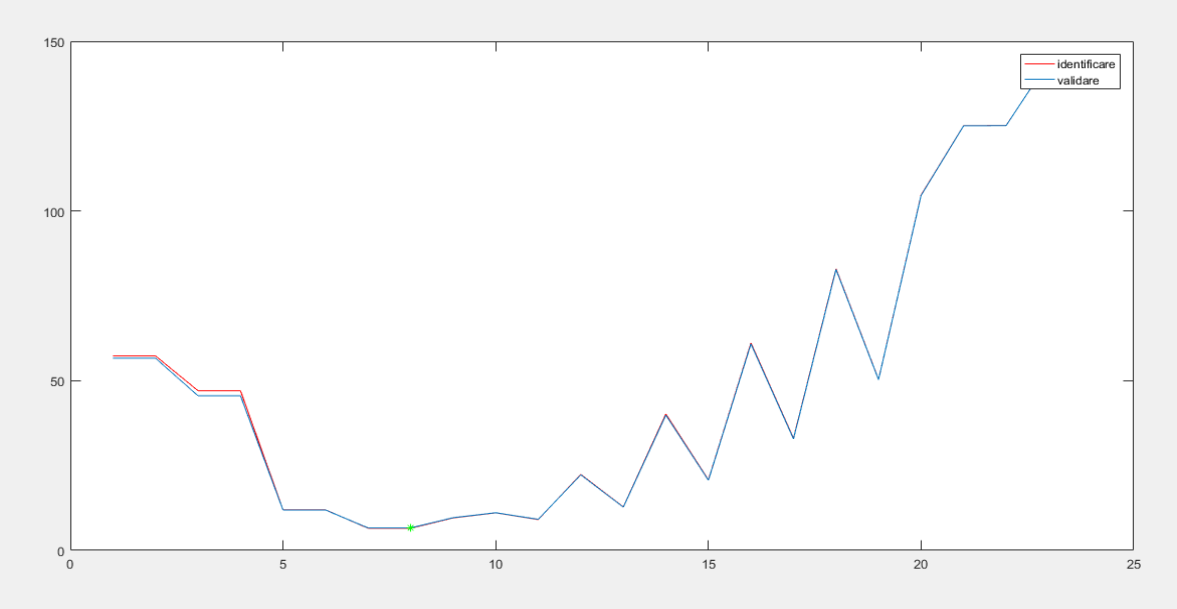

Fig. 7

Se observă că pentru gradul m=8 rezultatele sunt cele mai bune (erori mici), iar aproximatorul nostru va avea forma din figura 8.

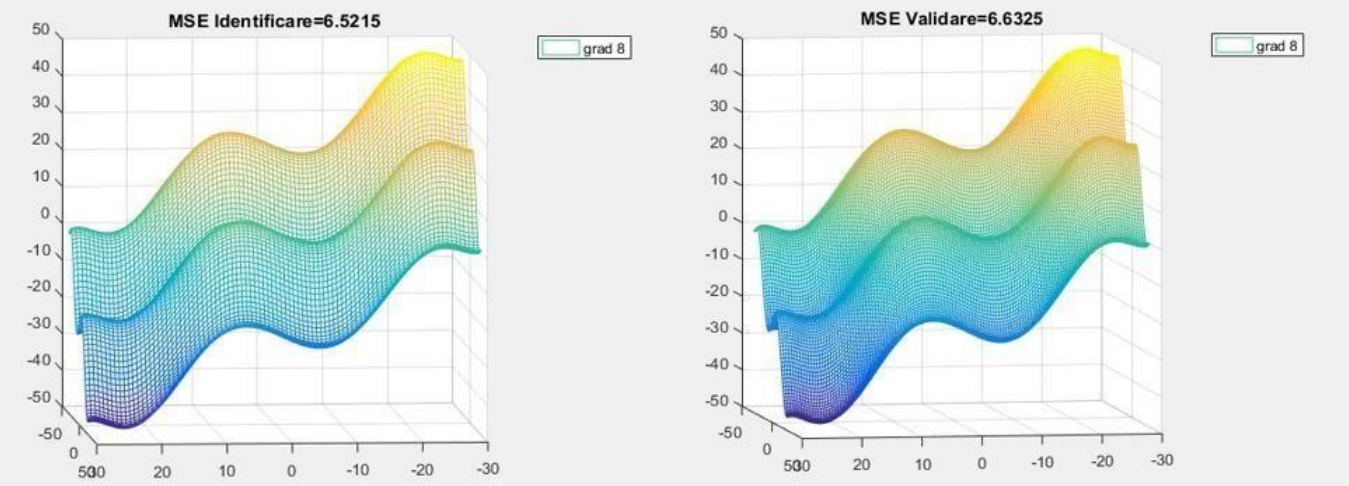

Fig. 8

Creșterea gradului nu mai este benefică din acest moment datorită inflexiunilor polinomului care nu se mai potrivesc cu cele ale setului de date. Această problemă apare din cauza supraantrenării sistemului (se aproximează și zgomotul din date). Acestea se pot observa în figura 9 de mai jos. (Grafic realizat pentru gradul m=16)

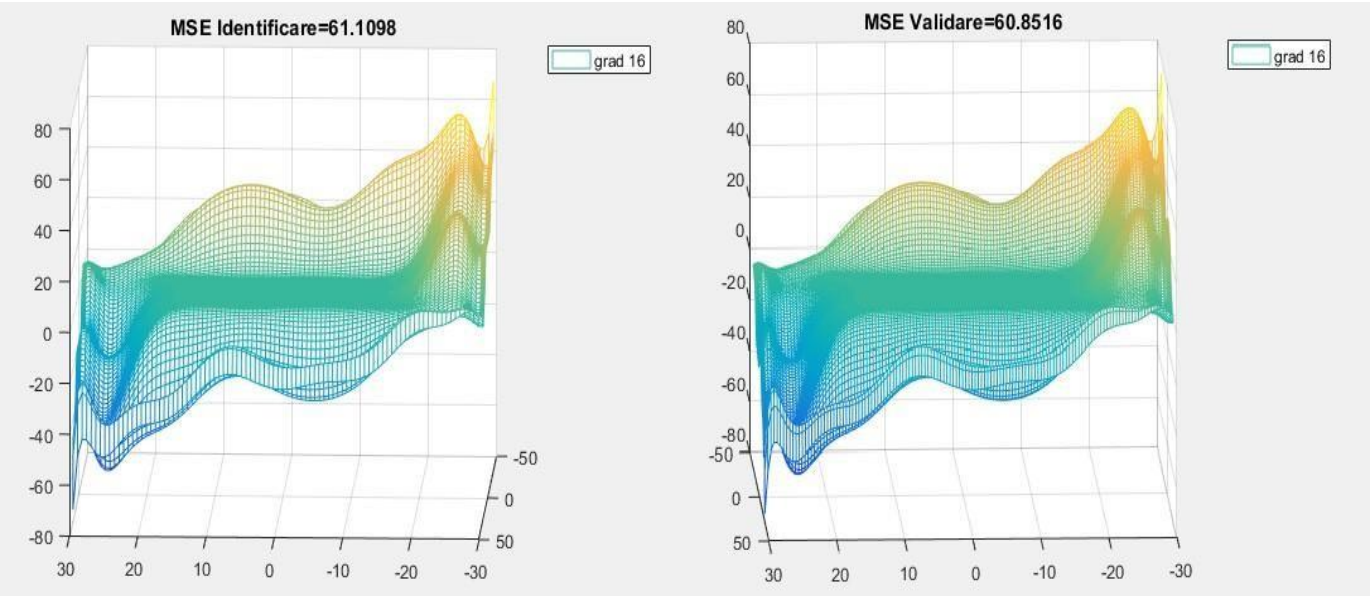

Fig. 9

4. Anexa cod MATLAB

part1_solution.m:

```
clc
clear all
close all
warning('off','all');
load('proj_fit_08.mat');
x1=id.X{1,1};
x2=id.X{2,1};
y=id.Y;
x1v=val.X{1,1};
x2v=val.X{2,1};
yv=val.Y;
M=length(x1);
mesh(x1,x2,y)
mesh(x1v,x2v,yv)
Nv=length(yv);
% =============IDENTIFICARE VALORI ===========
X1flat=[];
for i=1:M
    x1linie=part1_fctforflat(M,x1(i));
    X1flat=[X1flat,x1linie];
```

```
end
X2flat=[];
for i=1:M
    for j=1:M
        X2flat=[X2flat,x2(j)];
    end
end
Xflat=[X1flat;X2flat];
Yflat=reshape(y,1,M*M);

%==============VALIDARE VALORI============
X1val=[];
for i=1:Nv
    x1vallinie=part1_fctforflat(Nv,x1v(i));
    X1val=[X1val,x1vallinie];
end
X2val=[];
for i=1:Nv
    for j=1:Nv
        X2val=[X2val,x2v(j)];
    end
end
Xvalflat=[X1val;X2val];

%========== variabile pentru a salva informatiile la gradul optim==========
errid_minim=9999;
errval_minim=0;
grad_optim=1;
yidentificare_optim=[];
yvalidare_optim=[];
m_max=5;
for m=1:m_max
    %====================IDENTIFICARE ============
    PSI=[];
    for i=1:M*M
        L=part1_fforpsi(m,Xflat(:,i));
        PSI=[PSI;L];
    end
    A=PSI\Yflat';
    y_sim=PSI*A;
    y_est=reshape(y_sim,M,M);
    e=sum((y-y_est).^2);
    MSE(m)=1/length(e)^2*sum(e);


    %=============== VALIDARE =============
    PSIv=[];
    for i=1:Nv*Nv
        L=part1_fforpsi(m,Xvalflat(:,i));
        PSIv=[PSIv;L];
    end

    y_simVAL=PSIv*A;
    y_estVAL=reshape(y_simVAL,Nv,Nv);
    eVAL = sum((yv-y_estVAL).^2);
    MSEval(m)= 1/(length(eVAL)^2)*sum(eVAL);
```

```
    %Determinare grad optim (cea mai mica eroare)
    if MSE(m) < errid_minim
        errid_minim=MSE(m);
        errval_minim=MSEval(m);
        grad_optim=m;
        yidentificare_optim=y_est;
        yvalidare_optim=y_estVAL;
    end

end

mesh(x1,x2,y)
figure;
mesh(x1,x2,yidentificare_optim);
title(['MSE identificare =',num2str(errid_minim)]);
figure;
mesh(x1v,x2v,yvalidare_optim);
title(['MSE validare =',num2str(errval_minim)]);
figure;
plot(1:m_max,MSE,'r',1:m_max,MSEval,grad_optim,MSE(grad_optim),'*g');
legend('identificare','validare');
```

part1_fforpsi.m:

```
function [out]= part1_fforpsi(m,x)
 out=1;
   for i=1:m
       out=[out, x(1,1)^i, x(2,1)^i];
       for j=1:m
           if i+j<=m
               out=[out , (x(1,1)^j)*(x(2,1)^i)];
           end
       end
   end

end
```

part1_fctforflat.m:

```
function [out] = part1_fctforflat(m, x_i )
    out=[];
    for i=1:m
        out=[out, x_i];
    end
end
```