PROIECT

Partea I

Modelarea unei functii necunoscute folosind un aproximator polinomial

30 Octombrie 2021

—

Identificarea Sistemelor

—

Kovács Attila-Levente

Negrea Balázs Florin

Ciordas Dragos Florin

1/1 (indexele fisierelor)

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Partea 1: Modelarea unei funcții necunoscute

În cadrul primei părti a proiectului, am dezvoltat un model pentru o funcție necunoscută cu ajutorul unui aproximator polinomial, pe baza regresiei liniare. Aproximatorul polinomial are gradul m configurabil.

a.) Regresia liniară

În cazul în care două seturi de date variază împreună, *corelația* ne spune dacă această variație este direct (pozitivă) sau inversă (negativă). Corelația indică și puterea acestei relații prin valoarea *covarianței*. Cu toate acestea, există informații folositoare pe care nu putem afla utilizând doar corelația:

- forma matematică a relației (modul în care se modifică o variabilă în raport cu o altă variabilă)
- folosirea relației de corelație pentru a prezice valoarea cea mai probabilă pentru a doua variabilă, dacă cunoaștem o valoare dată a unei variabile

Cu ajutorul *regresiei* putem afla toate aceste informații, regresia fiind *complementară* corelației. Prin intermediul regresiei se pot face predicții ale unei variabile, în funcție de valoarea altei variabile.

Predicția este procesul de estimare a valorii unei variabile cunoscând valoarea altei variabile.

Regresia liniară reprezintă cel mai simplu tip de model parametric. Această tehnică își are originile încă din anul 1809, când Gauss o folosea pentru aproximarea orbitelor planetelor.

Reprezentarea grafică a modelului este o dreaptă care se apropie cel mai mult de datele furnizate, reprezentate prin puncte. Diferența dintre modelul obținut și sistemul dat este măsurat cu ajutorul erorii medii pătratice (en. **Mean Square Error**).

Formula pentru MSE este: $J = \frac{1}{N}\sum_{k=1}^{N} e^2(k) = \frac{1}{N}\sum_{k=1}^{N}(\hat{y}(k) - y(k))^2$

Elementele problemei de regresie liniară sunt:

- Un șir de eșantioane **cunoscute** y(k) ϵ $\mathbb{R}$, indexate de k = 1, ... ,N, iar y este măsurătoarea (variabilă dependentă)
- Pentru fiecare k, avem un vector **cunoscut** $\varphi(k)\epsilon\mathbb{R}^n$, care conține regresorii $\varphi_i(k), i = 1, \dots, n;\ \varphi(k) = [\varphi_1(k), \varphi_2(k), \dots, \varphi_n(k)]^T$
- Un vector de parametri $\theta\ \epsilon\ \mathbb{R}^n$, care este **necunoscut**

Scopul problemei de regresie este de a găsi dreapta de regresie, adică linia dreaptă care se potrivește cel mai bine cu datele noastre (best-fit straight line) – această dreaptă este

rezultatul modului în care cele două seturi de date covariază (variază împreună). Reprezentarea grafică a acestei drepte se poate vedea pe *figura nr. 1*.

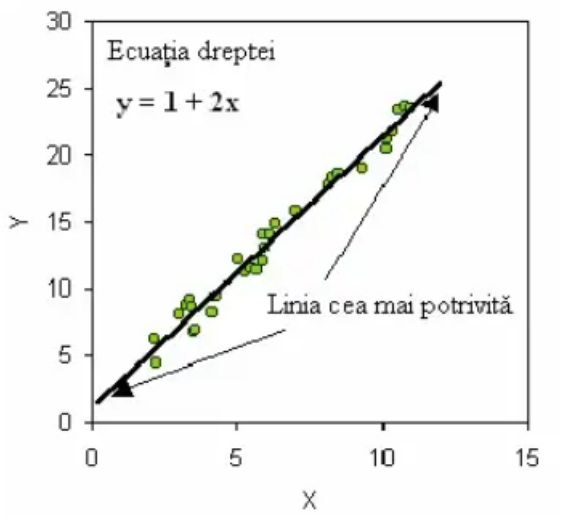

Figura 1: Reprezentarea grafică a dreptei de regresie

Există și *regresie liniară multiplă*, în care este definită relația dintre o variabilă dependentă ("*predicted*") și două sau mai multe variabile independente "*predictor*".

Pe *figura nr. 1.,* este prezentat graficul a două variabile (x,y) reprezentate în sistemul de coordinate xOy, împreună cu dreapta de regresie. Dreapta de regresie a fost determinată prin metoda celor mai mici pătrate, metodă care va fi prezentată în continuare.

În ecuația dreptei de regresie (y = 1 + 2x) din figura de mai sus, interceptul este egal cu 1, iar panta este egală cu 2. Se poate deduce ușor valoarea lui y, știind valoarea lui x. De exemplu, dacă x = 3, atunci y = 1 + 2*3 = 7.

Dreapta de regresie intersectează axa Oy la valoarea 1 (dată de coeficientul a) și pentru fiecare creștere a lui x cu o unitate, y crește cu două unități, deci panta (gradientul) este egală cu 2. Se observă că valoarea coeficienților de regresie a și b poate să fie și negativă.

Deci, regresia liniară implică două variabile, valoarea unei variabile este **dependentă** de valoarea celeilalte. În literatura de specialitate se utilizează următoarea convenție: variabila dependentă este y, iar variabila independentă este x. Astfel, putem spune că "regresăm y pe x", adică putem folosi ecuația regresiei pentru a prezice valoarea variabilei y dacă se cunoaște valoarea variabilei x.

De notat că dacă vrem să estimăm valoarea lui x din valoarea lui y, nu putem rearanja pur și simplu ecuația. Trebuie repetate calculele pentru determinarea regresiei considerând y ca variabilă independentă și x ca variabilă dependentă, trebuie să "regresăm x pe y". În majoritatea cazurilor, acest calcul va produce valori diferite pentru noi coeficienți de regresie a și b.

Există două posibilități de calculare a coeficienților de regresie: (a și b)

1) Dacă se cunoaște valoarea coeficientului de corelație dintre cele două variabile, mediile și abaterile standard a celor două variabile – această metodă nu o vom trata în cazul nostru
2) Metoda celor mai mici pătrate: această cale nu necesită cunoașterea valorii coeficientului de corelație, a mediei sau a abaterii standard a variabilelor implicate. Metoda este utilă, iar implică găsirea coeficienților unici de regresie a și b astfel încât suma pătratelor rezidurilor să fie minimă.
Rezidurile sunt diferențele dintre valorile actuale și valorile estimate (predicted/prezise).

În cazul nostru, scopul este identificarea comportamentului variabilei dependente (necunoscute / predicted), folosind următorul model liniar (regresia liniară) cu structura:

$$\boldsymbol{y(k) = \varphi^T(k) \cdot \theta}\ , \qquad \textit{(Formula 1)}$$

unde :

$y(k)$ *este o mărime măsurabil*ă $\quad (k = 1, \dots, N) \quad$ (coeficienți)

$\varphi(k) - N$ *mărimi cunoscute* $\rightarrow$ regresorii (coeficienți)

$\theta - n$ *mărimi necunoscute* $\rightarrow$ *mărimi, care trebuie calculate*

Regresia liniară are nenumărate utilizări, iar cele care sunt folosite la Identificarea Sistemelor le putem definim ca fiind:

1) **modelarea unei serii temporale** $y(k)$ unde k reprezinta o variabilă de timp
2) **aproximarea unei funcții** (supervised learning), un concept des utilizat în Machine Learning, o ramura a Inteligentei Artificiale.
3) Alte concepte de Identificarea Sistemelor

Acest caz (2) se potrivește primei părți a proiectului, unde trebuie să identificăm modelul unei funcții necunoscute, cu ajutorul metodei regresiei liniare, aproximând cu un polinom.

Putem observa deci, că regresia liniară poate fi utililizată pentru aproximarea unei funcții oarecare. În acest caz, când regresia liniară se utilizează pentru aproximarea unei funcții, regresorii $\varphi_i(k)$ din matricea $\phi\big(x(k)\big)$ se numesc **funcții de bază**, iar

$$\phi\big(x(k)\big) = \big[\phi_1\big(x(k)\big), \phi_2\big(x(k)\big), \dots, \phi_n\big(x(k)\big)\big]^T \qquad \textit{(Formula 2)}$$

b.) Exemple de regresori – polinom în x

Polinomul în x este util pentru aproximarea funcțiilor necunoscute.

Dacă avem un polinom de gradul 2, cu două variabile de intrare

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \left(= \begin{bmatrix} x_{11}\ x_{12}\ x_{13} \dots\ x_{1N} \\ x_{21}\ x_{22}\ x_{23} \dots\ x_{2N} \end{bmatrix} \right),$$

acel polinom este:

$$g(k) = \theta_1 + \theta_2 x_1(k) + \theta_3 x_2(k) + \theta_4 x_1^2(k) + \theta_5 x_2^2(k) + \theta_6 x_1(k) x_2(k) =$$

$$= [1 \quad x_1(k) \quad x_2(k) \quad x_1^2(k) \quad x_2^2(k) \quad x_1(k)x_2(k)] \cdot [\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5 \quad \theta_6]^T =$$

$$= \phi^T\big(x(k)\big) \cdot \theta = \varphi^T(k) \cdot \theta \qquad \textbf{(Formula 3)}$$

Într-un caz general, gradul polinomului poate fi de orice grad m, iar în acel caz general, polinomul este:

$$\hat{g}(x) = [1, x_1, x_2, \dots, x_1^m, x_2^m, x_1 x_2, x_1^2 x_2, \dots, x_1^{m-1} x_2, x_1 x_2, x_1 x_2^2, \dots, x_1 x_2^{m-1}] \cdot \theta,$$

unde $\theta = [\theta_1, \theta_2, \dots, \theta_n]^T$ este vectorul de parametrii, iar gradul m este un număr natural.

Scriind modelul definit mai sus pentru fiecare din cele N date, se va obține un sistem de ecuații liniare:

$$\begin{aligned} y(1) &= \varphi_1(1)\theta_1 + \varphi_2(1)\theta_2 + \cdots + \varphi_n(1)\theta_n = \hat{g}(x_1) \\ y(2) &= \varphi_1(2)\theta_1 + \varphi_2(2)\theta_2 + \cdots + \varphi_n(2)\theta_n = \hat{g}(x_2) \qquad \textbf{(Formula 4)} \\ &\dots \\ y(N) &= \varphi_1(N)\theta_1 + \varphi_2(N)\theta_2 + \cdots + \varphi_n(N)\theta_n = \hat{g}(x_N) \end{aligned}$$

În aproximarea funcțiilor, $\varphi_i(k) = \phi_i\big(x(k)\big)$.

Sistemul de ecuații poate fi rescris în formă matriceală:

$$\begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} = \begin{bmatrix} \varphi_1(1) & \varphi_2(1) & \dots & \varphi_n(1) \\ \varphi_1(2) & \varphi_2(2) & \dots & \varphi_n(2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(N) & \varphi_2(N) & \dots & \varphi_n(N) \end{bmatrix} \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \qquad \textbf{(Formula 5)}$$

Matricile se vor nota ca fiind:

$$\boxed{Y = \Phi \cdot \theta} \qquad \textbf{(Formula 6)}$$

$Y \in \mathbb{R}^n, \Phi \in \mathbb{R}^{Nxn}$.

Dacă N = n, sistemul se poate rezolva cu egalitate. În realitate, însă, este mai bine să folosim N > n de exemplu datorită **prezenței zgomotului la măsurări** (nu vrem să modelăm acel zgomot). Asftel, sistemul nu mai poate fi rezolvat cu egalitate, doar cu aproximare. Cea mai bună aproximare găsim utilizând eroarea medie pătratică între ieșirea măsurată și ieșirea prezisă/calculate.

Vectorul θ este cel *necunoscut*. Problema fiind de regresie liniară, folosim notația $\hat{\boldsymbol{\theta}}$.

Vectorul $\hat{\theta}$ este soluția reală a problemei de regresie dat fiind setul datelor de intrare și ieșire, dar rămâne totuși o estimare datoriă prezenței zgomotului din date.

Pentru a rezolva problema de regresie, cu aproximare, pornim de la $Y = \phi \cdot \theta$ și înmulțim la stânga cu ϕ^T:

$$\phi^T \cdot Y = \phi^T \cdot \phi \cdot \theta$$

Continuând, se obține cu o altă înmulțire la stânga cu inversa matricii $\phi^T\phi$:

$$\hat{\theta} = (\phi^T \cdot \phi)^{-1} \cdot \phi^T \cdot Y \qquad \textbf{\textit{(Formula 7)}}$$

Această ecuație se poate rezolva direct sau cu o expresie alternativă, dar aceste metode nu le vom trata în cazul de față fiindcă ambele metode se bazează pe inversarea de matrici și se comportă prost din punct de vedere numeric. Există algoritmi mai buni, iar MATLAB alege automat algoritmul potrivit.

ϕ se stochează de obicei în variabila ***PHI*** și Y în ***Y***, comanda care rezolvă sistemul de ecuații în sensul celor mai mici pătrate este împărțirea matriceală la stânga (backslash):

$$\boldsymbol{theta = PHI \setminus Y;} \qquad \textbf{\textit{(Formula 8)}}$$

Notă: *se poate folosi și funcția* `linsolve` *în loc de* \. *(dacă se dorește un control mai detaliat al algoritmului). A se vedea doc* `linsolve` *în MATLAB.*

Alegerea modelului potrivit (cu gradul potrivit) se va face pe baza erorii minime pătratice pe setul de date pe validare.

Eroarea medie pătratică în funcție de gradul polinomului de aproximare se comportă în felul următor: *(figura nr. 2)*

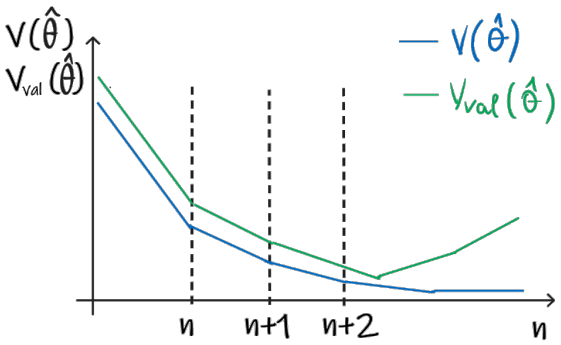

Figura 2: Comportarea valorii MSE în funcție de gradul n a polinomului de aproximare

Notă: Dacă datele sunt afectate de zgomot (în practică, măsurătorile mereu sunt afectate de zgomot), creșterea exagerată a lui n (a gradului polinomului) va duce la **supraantrenare** (performanțe bune pe datele de identificare, dar proaste pe date diferite din cauza că după o anumită valoare a gradului aproximatorului, aproximarea pe setul de date de identificare devine mai bună, astfel modelăm și zgomotul). Datorită acestei fapte, **validarea pe un set separat de date este vitală în practică!**

Așadar, la aproximarea funcției în partea următoare, vom crește treptat valoarea lui n (gradul aproximatorului) până când eroarea pe datele de validare începe să crească.

c.) Modelarea unei funcții necunoscute folosind regresia liniară – aproximator polinomial

În acest caz, vor fi utilizate datele de identificare și validare din fișierul `proj_fit_01.mat` – de tip MATLAB data. Aproximarea se va face cu ajutorul programului `MATLAB`, codul va fi anexat la această lucrare de proiect.

Pentru aproximarea funcției necunoscute se va utiliza metoda descrisă mai sus (metoda polinom în x).

Primul pas este de a încărca fișierul.mat în mediul MATLAB, utilizând comanda: `load('proj_fit_01.mat').`

Se va alege gradul m al aproximatorului în funcție de valoarea erorii medii pătratice (MSE – Mean Square Error), iar pentru acest lucru va trebui ca programul să itereze pe un anumit interval a lui m.

În cazul de față se iterează de la 2 la 25. Se poate observa că codul este realizat astfel încât gradul este variabil. Utilizatorul poate specifica orice valoare pentru capătul de sus a intervalului [1; m].

Vor fi inițializate variabile pentru a păstra valoarea erorilor – 2 vectori: unul pentru datele de identificare, iar unul pe datele de validare – cu această metodă se poate afla gradul polinomului pentru aproximarea cea mai bună.

Următorul pas este încărcarea datelor din structura **id** din fișierul .mat, care conține cele 2 variabile de intrare a funcției necunoscute în structura **X**, care se extrag cu ajutorul următoarelor linii de cod: (în cazul de față am pus direct la invitarea funcției `calculeazaPHI`)

```
x1 = id.X{1};
x2 = id.X{2};
```

În mod similar se extrage și vectorul valorilor ieșirii **Y**: y = id.Y;.

Se observă că funcția necunoscută are 2 variabile de intrare, astfel se poate reprezenta în 3D. Reprezentarea funcției se poate vedea pe *figura nr. 3*, care se face cu ajutorul funcției `mesh` din MATLAB.

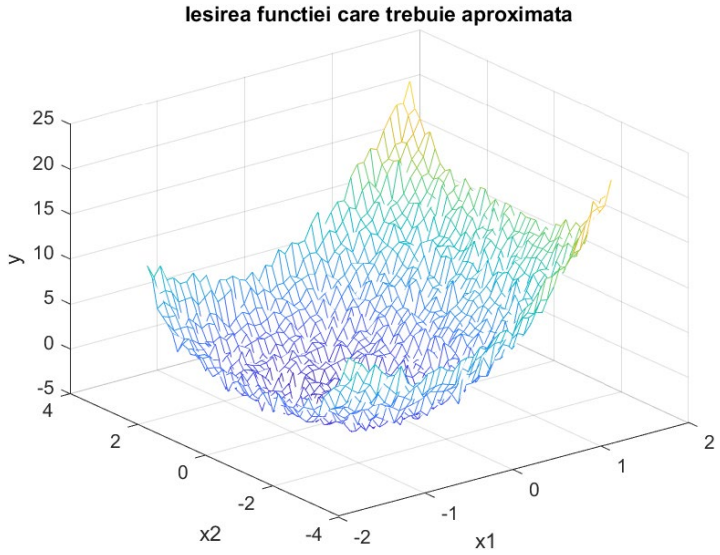

Figura 3: Reprezentarea ieșirii y al funcției necunoscute când intrările sunt x1, și respectiv x2 – datele de identificare

De pe acest grafic, se poate observa prezența zgomotului la măsurare. Ieșirea funcției necunoscute este afectată de zgomot.

Aproximarea se va face de mai multe ori, mai exact de 25 de ori, cu polinoame de gradul 2,3..., 25. Polinomul de gradul 1 nu se va lua în considerare fiindcă MSE este prea mare în acest caz și pe datele de identificare și distorsionează graficul obținut la final.

Cea mai bună aproximare este considerată cea la care eroarea medie pătratică pe datele de validare este minimă.

După încărcarea datelor și vizualizarea graficului se va calcula matricea Φ, care este compusă din valorile $\boldsymbol{\varphi_n}(\boldsymbol{N})$, unde N este lungimea datelor, iar valorile $\boldsymbol{\varphi_n}$ provin din sistemul de ecuații liniare prezentat în secțiunea anterioară *(Formula 4)*.

Ecuațiile sunt implementate în MATLAB doar în formă matriceală *(Formula 5 și 6):*

$$\boldsymbol{Y = \phi \cdot \theta}$$

Generarea matricii a fost realizată folosind funcția `calculeazaPHI`, care este afișată pe *figura nr. 4.* și are ca parametrii de intrare:

- Vectorii x1,x2
- Constanta m, ce reprezintă gradul polinomului care va fi creată

Aceasta ne va genera matricea `phi`. ($\boldsymbol{\phi}$)

Cea mai imbricată structură repetitivă are rolul de a genera fiecare termen din cadrul acestei matrici ($\varphi_1(1), \varphi_2(1)$ ... $\varphi_N(1)$) și de a le introduce în matricea linie (vectorul) cu nume sugestiv '`rand`'.

```
function phi = calculeazaPHI(x1,x2,m)

phi = [];

for i = 1:length(x1)
    for j = 1:length(x2)
        rand = [];
        for p1 = 0:m
            for p2 = 0:m
                if(p1+p2 <= m)
                    rand = [rand x1(i)^p1*x2(j)^p2];
                end
            end
        end
        phi = [phi; rand];
    end
end

end
```

Figura 4: Funcția `calculeazaPHI` din codul asociat MATLAB

'`rand`'-urile sunt apoi adăugate în cadrul matricei coloane '`phi`', astfel obținându-se matricea $\boldsymbol{\phi}_{Nxn}$ *(Formula 5 și 6).*

Matricile '`PHI_id`' și '`PHI_val`' au fost generate prin apelul funcției '`calculeazaPHI`', prezentată anterior, având ca parametrii de intrare vectorii de identificare respectiv cei de validare, precum si gradul pentru care se dorește generarea ieșirii așteptate (en. predicted output).

Este apoi calculată matricea '`THETA`' prin împărțirea matriceală la stânga a matricelor '`PHI_id`' respectiv '`Y`'.

Desigur, acest polinom a fost generat atât pe datele de identificare cât și pentru cele de validare.

Pentru fiecare iterație a structurii repetitive, respectiv pentru gradul 1, 2, ... m, vor fi generate matricile ϕ, ieșirile y, matricea θ și cel mai important, au fost creați vectorii pentru erorile medii pătratice, reprezentați în *figura nr. 5*.

```
for grad = 1:m

PHI_id = calculeazaPHI(id.X{1},id.X{2},grad);

THETA = PHI_id\Y;    % asta o singura data se calculeaza!! doar pentru
datele de identificare
yhat_id = PHI_id*THETA;

MSE_id(1,grad) = mean((Y - yhat_id).^2);

PHI_val = calculeazaPHI(val.X{1}, val.X{2}, grad);
yhat_val = PHI_val * THETA;

MSE_val(1,grad) = mean((y_val - yhat_val).^2);

end
```

Figura 5: Structura repetitivă principală

d.) Concluzii

După 25 de iterații asupra gradului se poate vedea faptul că modelul devine supraantrenat fiindcă eroarea pe datele de validare începe să crească. Acest fapt reiese din *figura nr.6.* Eroarea pe datele de validare este minimă când gradul aproximatorului este 6. Se poate observa că forma graficului de pe *figura nr. 6* seamănă cu cea prezentată anterior, pe *figura nr.* 2 – adică comportamentul modelului este cea așteptată.

Astfel, soluția optimă în cazul de față este utilizarea unui polinom de gradul 6 pentru a aproxima funcția necunoscută, neglijând zgomotul de măsurare. De notat că pe această figură este afișată eroarea începând cu aproximatorul de gradul 2. Astfel, se poate afișa graficul mai estetic și îndepărtarea erorii medie pătratice este mai vizibilă.

De notat și faptul că în cod pentru calculul erorii am utilizat funcția `mean` din MATLAB, fiind mai convenabil codul. Cu formula generală a Erorii medii pătratice tot la aceste rezultate ajungem.

Se va afișa grafic cea mai bună aproximare a funcției necunoscute, dar și valoarea erorii. În cod, este marcat secțiunea unde se face această afișare: găsirea erorii minime și gradul corespunzător cu ajutorul funcției min, recalcularea ieșirii prezise pentru identificare și pentru validare – de notat că **la validare nu se efectuează împărțirea matricială** pentru a afla valorile dependente, fiind un calcul pentru validarea calculelor anterioare. În ultimul pas se afișează grafic funcția aproximată, cum se va vedea pe *figura nr. 7* pentru datele de identificare și *figura nr. 8* pentru validare. S-au afișat pe ambele figuri atât ieșirea prezisă, cât și ieșirea măsurată. Pe ieșirea măsurată se poate vedea prezența zgomotului la măsurare. De asemenea, cele două funcții au fost colorate diferit pentru a putea vedea diferența între ele.

Cu albastru este afișat ieșirea aproximată (prezisă), iar cu roșu, ieșirea furnizată (reală) la setul de date.

De asemenea, pentru a aduce un suport practic fenomenului de supraantrenare, pe *figura nr. 9* se poate vedea efectul supraantrenării: când aproximatorul are **gradul 28**, de exemplu (gradul 28 a fost ales pentru că eroarea este mai accentuată în acest caz), ieșirea prezisă diferă foarte mult pe datele de validare față de ieșirea reală. Acest fenomen, cum a fost discutat și anterior, este datorită supraantrenării, fiindcă pe datele de identificare, aproximatorul va modela și zgomotul adăugat la ieșirea funcției.

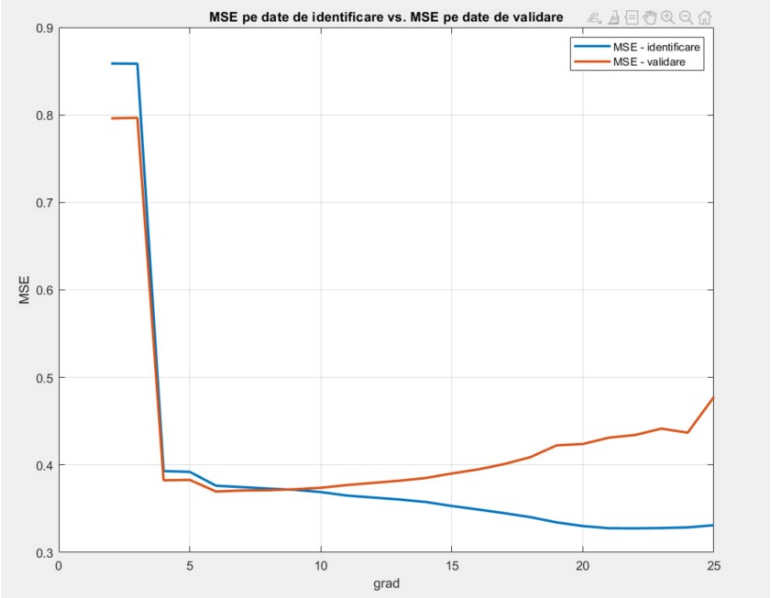

Figura 6: Reprezentarea grafică a erorii în funcție de gradul aproximatorului

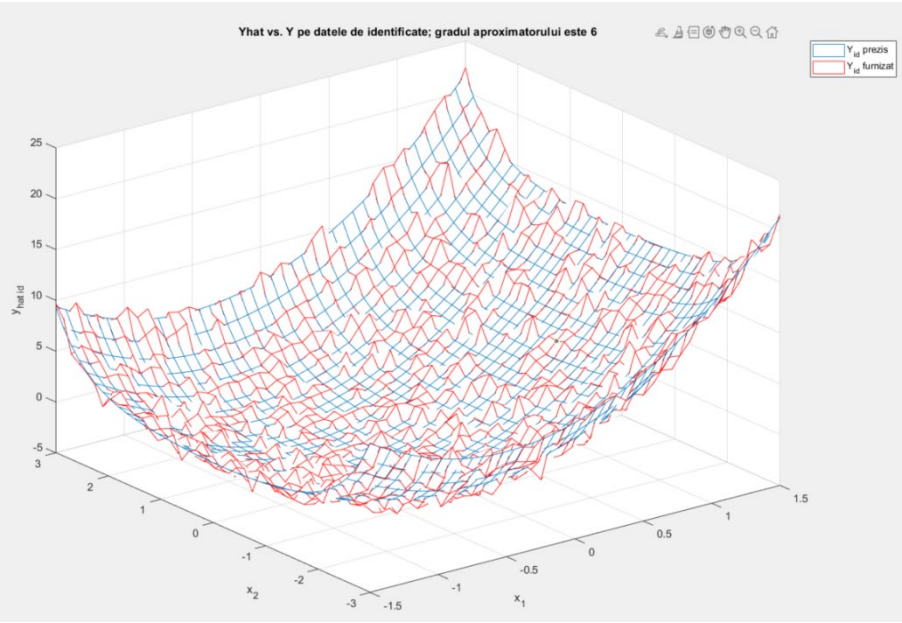

Figura 7: Reprezentarea grafică a funcției la aproximarea datelor de identificare

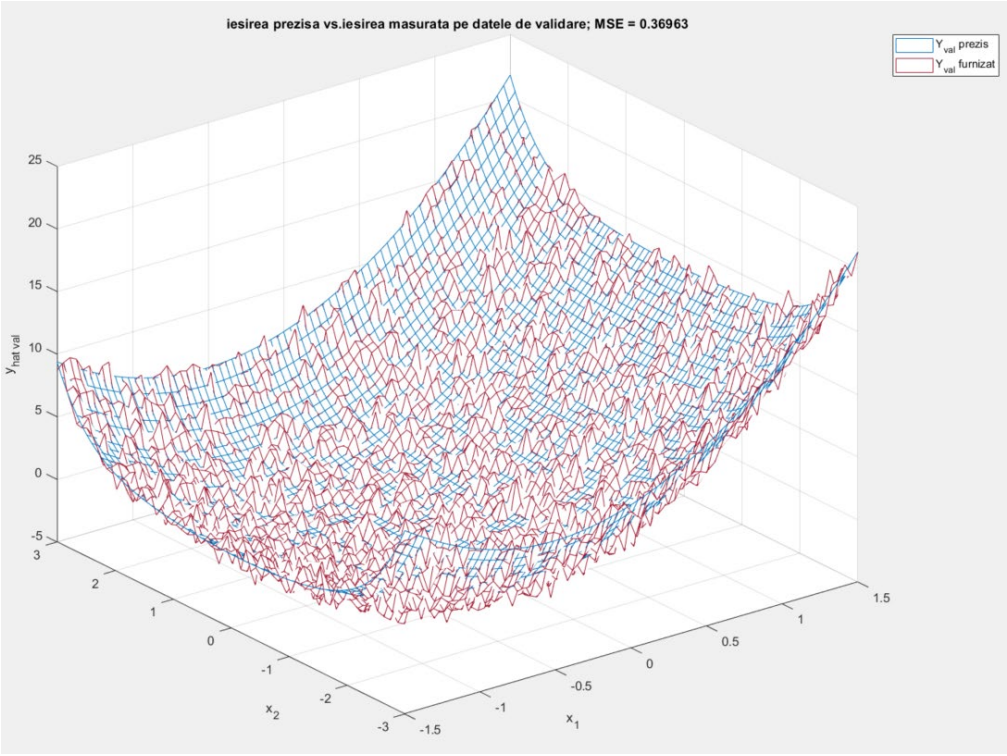

Figura 8: Reprezentarea funcției aproximate utilizând datele de validare

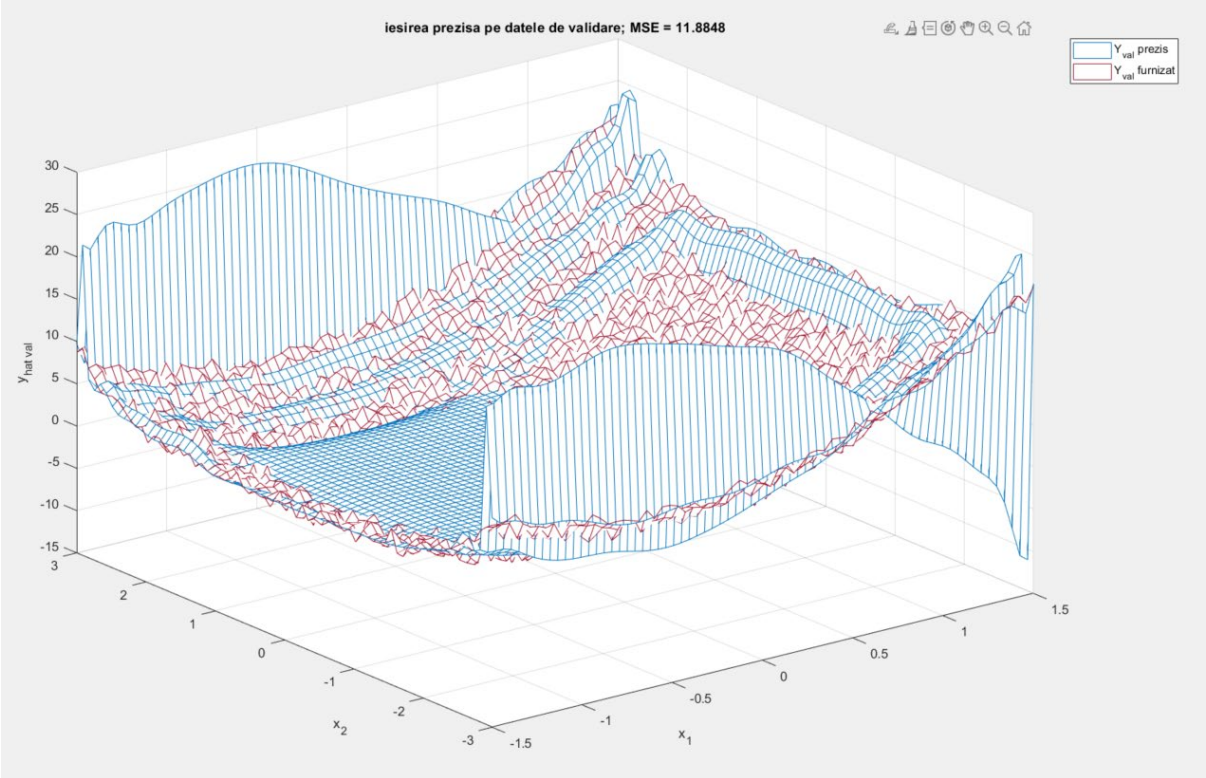

Figura 9: Fenomenul de supraantrenare, aproximator de gradul 28 – eroare mare

e.) ANEXA – codul MATLAB

```
clear all;
close all;
load('proj_fit_01.mat');

Y = id.Y;
Y = Y(:);
y_val = val.Y;
y_val = y_val(:);

m = 28;
MSE_id = zeros(1,m);
MSE_val = zeros(1,m);

for grad = 1:m

PHI_id = calculeazaPHI(id.X{1},id.X{2},grad);

THETA = PHI_id\Y;   % asta o singura data se calculeaza!! doar pentru datele de
identificare
yhat_id = PHI_id*THETA;

MSE_id(1,grad) = mean((Y - yhat_id).^2);

PHI_val = calculeazaPHI(val.X{1}, val.X{2}, grad);
yhat_val = PHI_val * THETA;

MSE_val(1,grad) = mean((y_val - yhat_val).^2);

end

figure();
grad_vect = 1:1:m;
plot(grad_vect, MSE_id), hold on
plot(grad_vect, MSE_val), hold off
title("MSE pe date de identificare vs. MSE pe date de validare");
xlabel("grad"), ylabel("MSE");
grid, legend("MSE - identificare", "MSE - validare");

%% se cauta gradul aproximatorului unde eroarea este cea mai mica
[msemin, grad] = min(MSE_val)
PHI_id = calculeazaPHI(id.X{1},id.X{2},grad);

THETA = PHI_id\Y;
yhat_id = PHI_id*THETA;
mse_id = mean((Y - yhat_id).^2);
PHI_val = calculeazaPHI(val.X{1}, val.X{2}, grad);
yhat_val = PHI_val * THETA;
mse_val = mean((y_val - yhat_val).^2);

figure(),
mesh(id.X{1},id.X{2},reshape(yhat_id, [length(id.X{1}), length(id.X{2})]),
'EdgeColor',[0 0.4470 0.7410])
xlabel("x_1"), ylabel("x_2"), zlabel("y_{hat id}");
title(strcat("Yhat vs. Y pe datele de identificate; gradul aproximatorului este ",
num2str(grad)));
hold on, mesh(id.X{1},id.X{2},id.Y, 'EdgeColor',[1 0 0]);
hold off;
```

```
legend("Y_{id} prezis", "Y_{id} furnizat");

figure();
mesh(val.X{1},val.X{2},reshape(yhat_val, [length(val.X{1}), length(val.X{2})]),
'EdgeColor',[0 0.4470 0.7410]);
xlabel("x_1"), ylabel("x_2"), zlabel("y_{hat val}")
title(strcat("iesirea prezisa vs.iesirea masurata pe datele de validare; MSE = ",
num2str(mse_val)));
hold on, mesh(val.X{1}, val.X{2}, val.Y, 'EdgeColor',[0.6350 0.0780 0.1840]);
hold off;
legend("Y_{val} prezis","Y_{val} furnizat" )

%% vizualizare cu grad maxim - cazul de supraantrenare\

PHI_id = calculeazaPHI(id.X{1},id.X{2},28);
THETA = PHI_id\Y;
yhat_id = PHI_id*THETA;
mse_id = mean((Y - yhat_id).^2);
PHI_val = calculeazaPHI(val.X{1}, val.X{2}, 28);
yhat_val = PHI_val * THETA;
mse_val = mean((y_val - yhat_val).^2);

figure(),
mesh(id.X{1},id.X{2},reshape(yhat_id, [length(id.X{1}), length(id.X{2})]),
'EdgeColor',[0 0.4470 0.7410])
xlabel("x_1"), ylabel("x_2"), zlabel("y_{hat id}");
title(strcat("Yhat pe datele de identificate; gradul aproximatorului este ",
num2str(grad)));
hold on, mesh(id.X{1},id.X{2},id.Y, 'EdgeColor',[1 0 0]);
hold off;
legend("Y_{id} prezis", "Y_{id} furnizat");

figure();
mesh(val.X{1},val.X{2},reshape(yhat_val, [length(val.X{1}), length(val.X{2})]),
'EdgeColor',[0 0.4470 0.7410]);
xlabel("x_1"), ylabel("x_2"), zlabel("y_{hat val}")
title(strcat("iesirea prezisa pe datele de validare; MSE = ", num2str(mse_val)));
hold on, mesh(val.X{1}, val.X{2}, val.Y, 'EdgeColor',[0.6350 0.0780 0.1840]);
hold off;
legend("Y_{val} prezis","Y_{val} furnizat" )

function phi = calculeazaPHI(x1,x2,m)

phi = [];
for i = 1:length(x1)
    for j = 1:length(x2)
        rand = [];
        for p1 = 0:m
            for p2 = 0:m
                if(p1+p2 <= m)
                    rand = [rand x1(i)^p1*x2(j)^p2];
                end
            end
        end
        phi = [phi; rand];
    end
end

end
```