# PROIECT IDENTIFICAREA SISTEMELOR

PARTEA A II-A:

ARX NELINIAR

# Cuprins


## Studentii:

Balan Andreea

Găldean Celina

Costinean Sebastian

## Grupa 30136/1

## Index:

15

# 01 Introducere

Scopul lucrării:

- identificarea unui sistem pe baza modelului tip cutie neagră

Metoda folosită:

- Arx neliniar

Structura Arx neliniar:

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

Unde: na şi nb - ordinele Arx neliniar

nk - întârzierea

d(k) - vector de intrări şi ieşiri întârziate

p - polinom de grad m de variabilele conținute de vectorul d

m ∈ {1,2,3}

# 02 Implementare

1. Implementarea acestui proiect a constat în dezvoltarea și aplicarea unui model de predicție și simulare folosind limbajul MATLAB.

2. Matricea de puteri necesară identificării a fost generată cu ajutorul unei funcții specifice, care a jucat un rol crucial în procesul de identificare.

3. Am dezvoltat o funcție specializată, 'generare', care a facilitat generarea combinată și ordonată a puterilor pentru termenii polinomiali.

# Generarea matricii de puteri

```
%functie generare
function puteri = generare(grad, na, nb)
    ultrand = zeros(1, na + nb); % Combinatia initiala
    puteri = ultrand;

    nrcomb = grad + 1; % Numarul de combinatii initiale (o singura coloana in matrice)
    nrcombnoi = 0;

    for i = 1 : na + nb
        while sum(ultrand) < grad
            ultrand(i) = ultrand(i) + 1;
            puteri = [puteri; ultrand];

            if i > 1 % Pentru i == 1 nu exista combinatii anterioare
                nrcombnoi = nrcombnoi + 1;
                randnealterat = ultrand;

                for j = 2 : nrcomb % Verifica toate combinatiile anterioare
                    randanterior = puteri(j, :);
                    randanterior = randanterior(1 : i - 1);
                end

                ultrand = randnealterat; % Continua incrementarea cu 1 de la ultimul rand

            end
        end
        nrcomb = nrcomb + nrcombnoi;
        nrcombnoi = 0;
        ultrand(i) = 0;
    end
end
```

# Matricea de regresori

1. Matricea de regresori, reprezentată de variabila **phiid**, este esențială în modelarea polinomială a sistemului, organizând puterile intrărilor și ieșirilor pentru identificarea optimă a parametrilor prin metoda celor mai mici pătrate.

2. În codul MATLAB, variabilele **yk** și **uk** sunt utilizate pentru a construi matricea de regresori pentru identificarea modelului.

3. Estimarea parametrilor modelului se realizează prin rezolvarea sistemului de ecuații liniare folosind operatorul ****, unde matricea de regresori **phiid** și vectorul de ieșire **y** sunt esențiale în procesul de ajustare a modelului la datele de identificare.

# 03 **Rezultate**

- Eroarea medie pătratică

| | m=1 | | m=2 | | **m=3** | |
|---|---|---|---|---|---|---|
| | Predicție | Simulare | Predicție | Simulare | Predicție | Simulare |
| na=nb=1 | Id:0.014<br>Val:0.003 | Id:4.878<br>Val:0.404 | Id:0.014<br>Val:0.003 | Id:46.265<br>Val:63.43 | Id:0.014<br>Val:0.003 | Id:5.112<br>Val:0.394 |
| na=nb=2 | Id:0.007<br>Val:0.001 | Id:5.048<br>Val:0.339 | Id:0.007<br>Val:0.001 | Id:4.595<br>Val:0.976 | Id:0.007<br>Val:0.001 | Id:4.840<br>Val:0.364 |
| **na=nb=3** | Id:0.000<br>Val:0.000 | Id:5.804<br>Val:0.303 | Id:0.000<br>Val:0.000 | Id:5.530<br>Val:0.145 | **Id:0.000<br>Val:0.000** | **Id:5.701<br>Val:0.079** |

# Discuție rezultate

1. Optimizare cu Gradul 3:

Alegerea cu grijă a gradului polinomial influențează puternic performanța. În special, pentru n=3 și grad=3, obținem o aproximare precisă a sistemului, ilustrând importanța acestei configurări.

2. Performanță Remarcabilă:

Configurația n=3 și grad=3 evidențiază o performanță excepțională. Acest rezultat susține eficacitatea acestei combinații în aproximarea și simularea sistemului.

3. **Compromis între Detalii și Generalizare:**

Gradul polinomial reprezintă un compromis crucial între detaliile complexității sistemului și capacitatea modelului de a generaliza eficient la datele de validare.

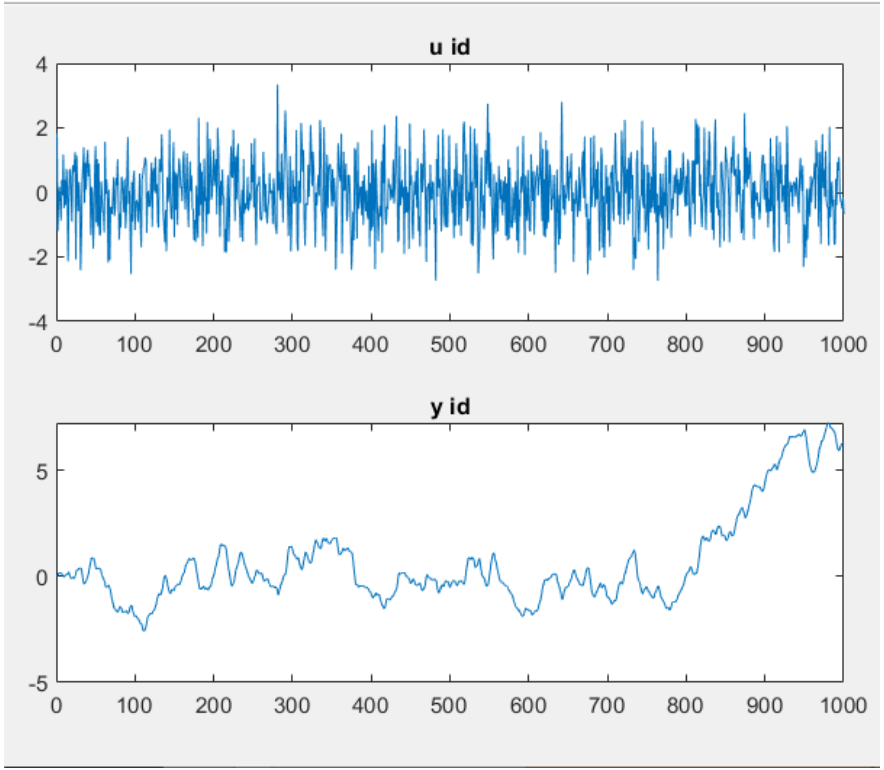
# 04 Grafice

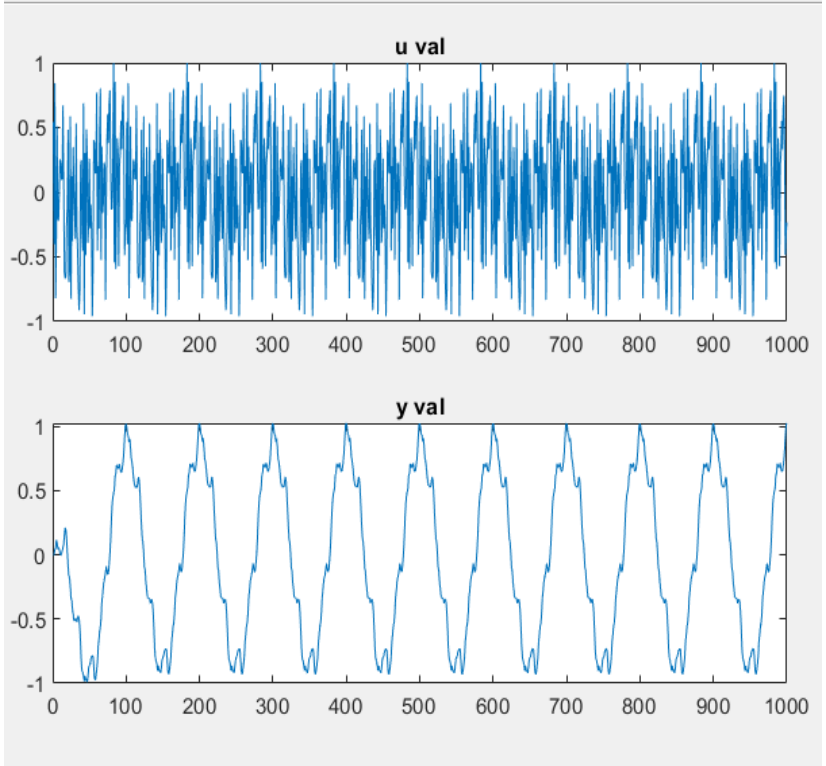
Predicţie

Simulare

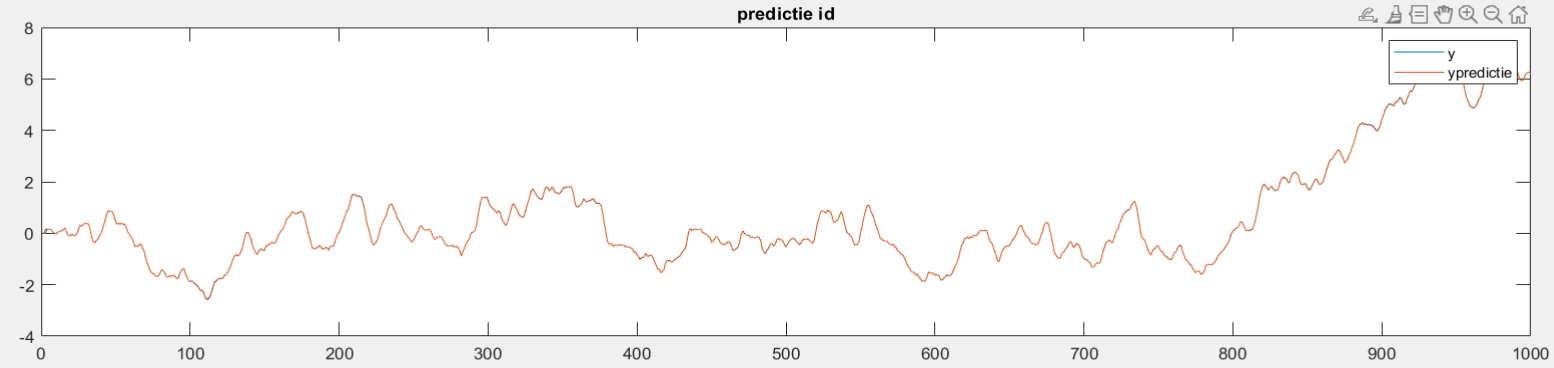
Model+Predicție +Simulare

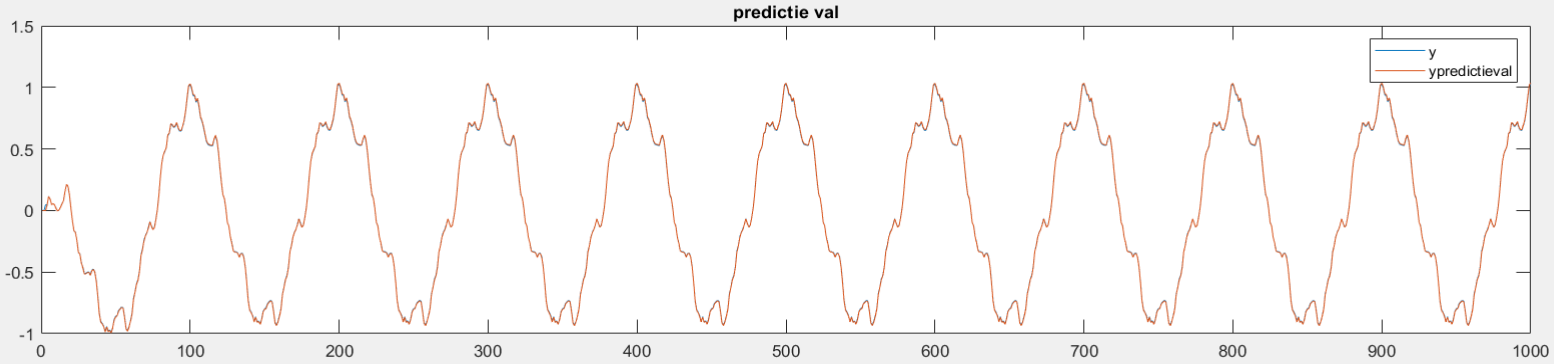
# Datele primite



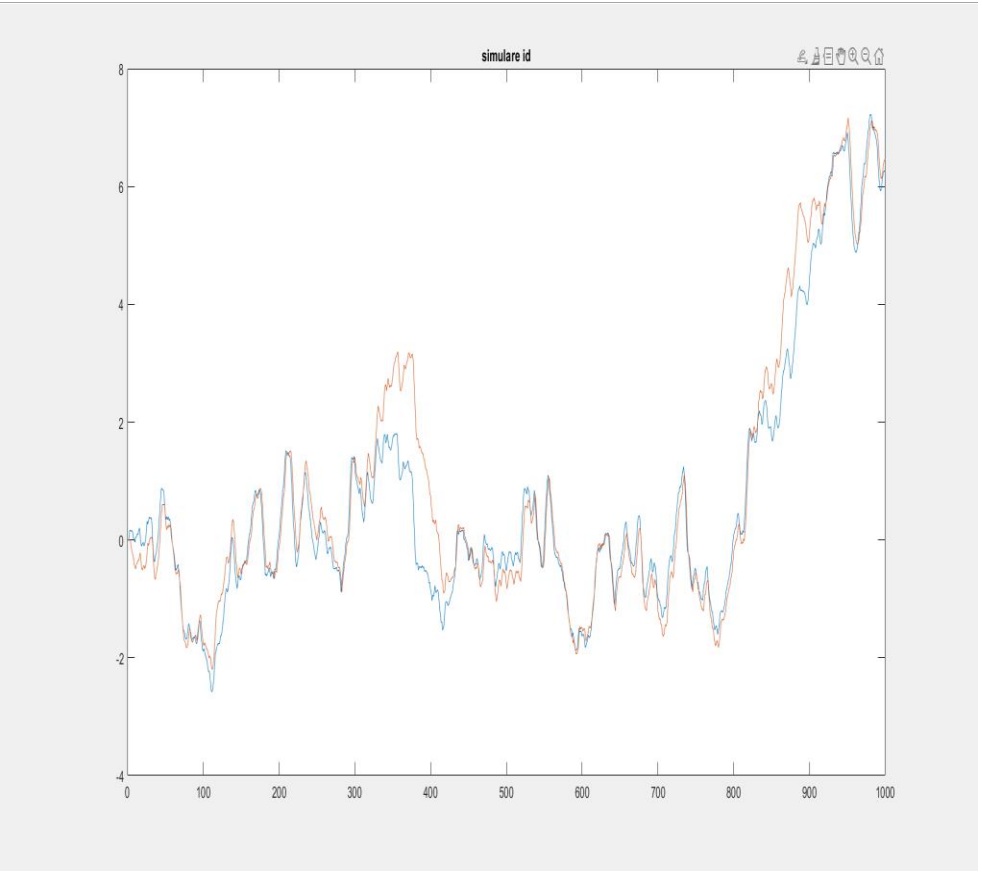



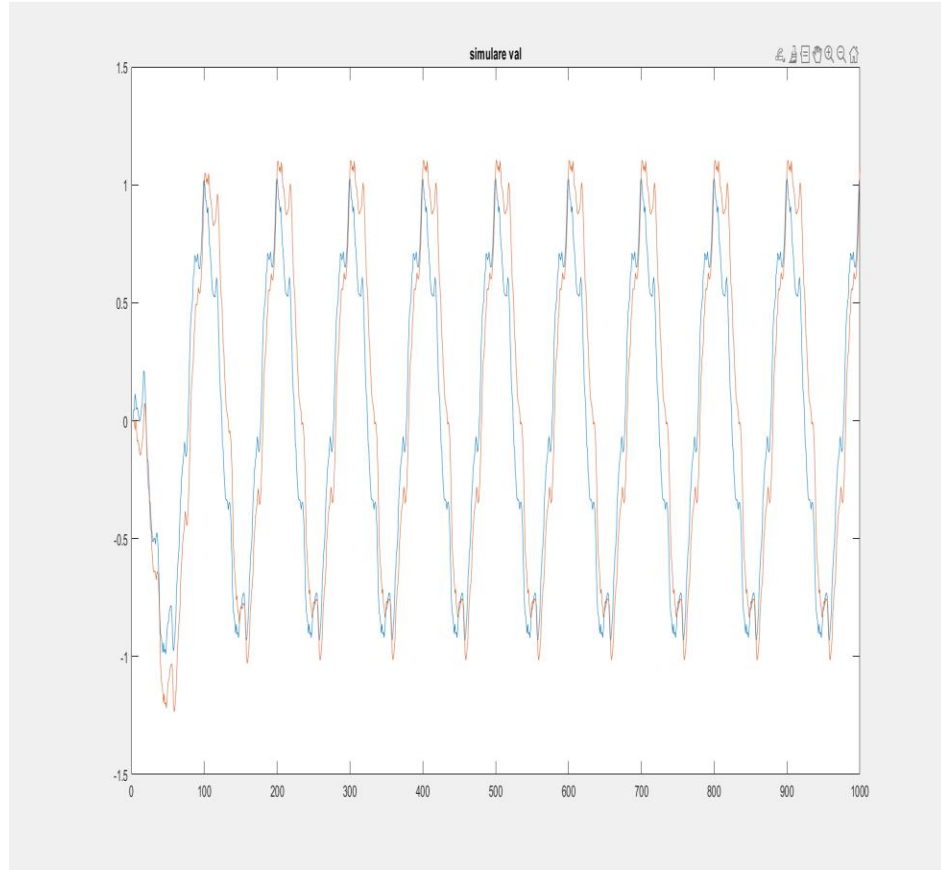
# Predicţie


predictie id
y
ypredictie
8
6
4
2
0
-2
-4
0
100
200
300
400
500
600
700
800
900
1000

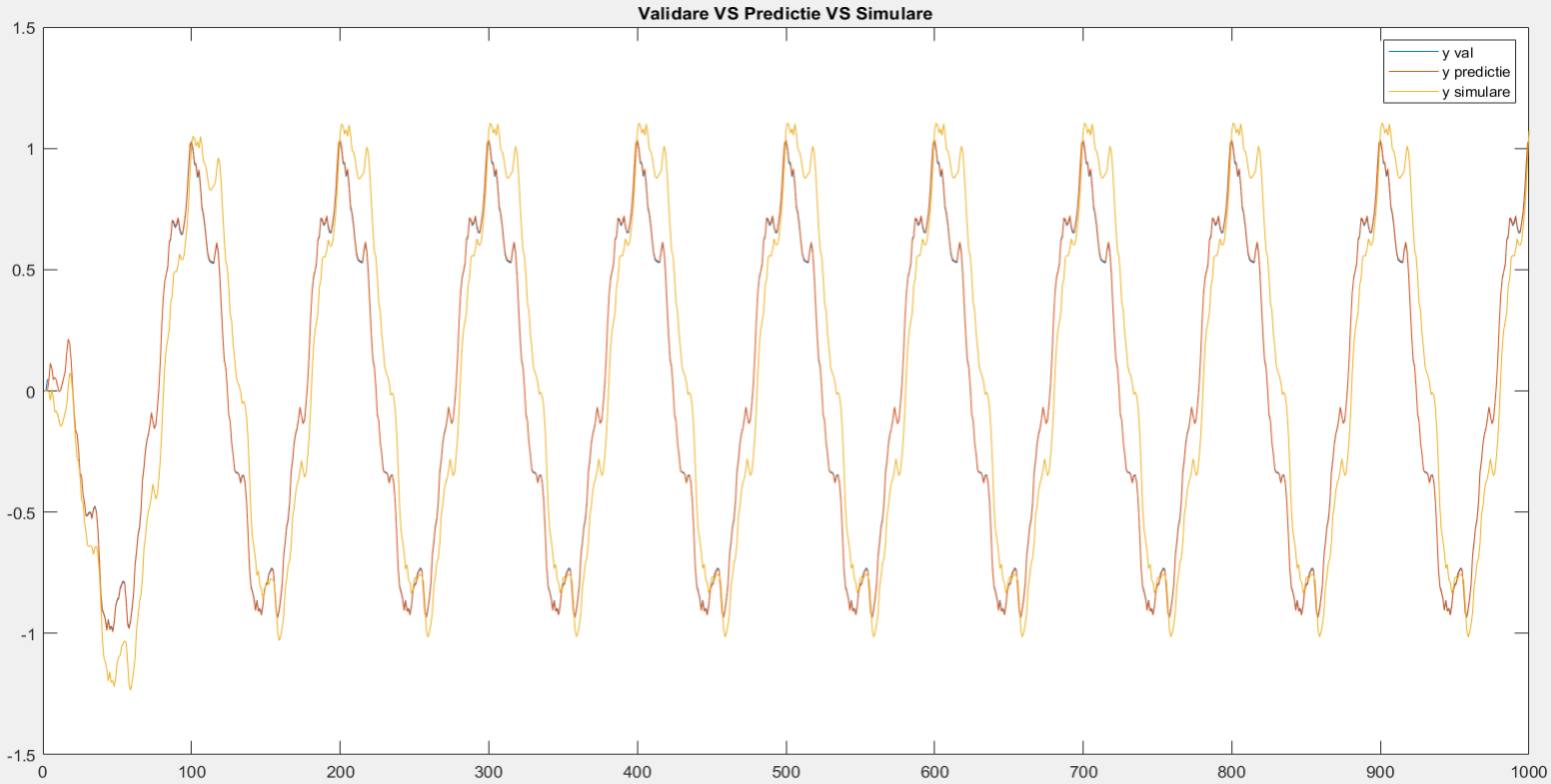

predictie val
y
ypredictieval
1.5
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600
700
800
900
1000

# Simulare


simulare id
8
6
4
2
0
-2
-4
0
100
200
300
400
500
600
700
800
900
1000


simulare val
1.5
1
0.5
0
-0.5
-1
-1.5
0
100
200
300
400
500
600
700
800
900
1000

# Model+Predicție +Simulare



# 05 **Concluzii**

În concluzie, modelul NARX dezvoltat prezintă o performanță satisfăcătoare, evidențiată prin evaluarea MSE. Cu toate acestea, influența gradului polinomial asupra preciziei modelului este un aspect semnificativ ce poate fi luat în considerare pentru optimizări ulterioare.

# Mulţumim pentru atenţie!

# Anexă

```
clear all
clc
close all
load('iddata-15')

u = id.u;
y = id.y;
vu = val.u;
vy = val.y;

figure
subplot(2,1,1);
plot(vu);
title('u val')
hold on
subplot(2,1,2);
plot(vy);
title('y val')

figure
subplot(2,1,1);
plot(u);
title('u id')
hold on
subplot(2,1,2);
plot(y);
title('y id')

NA = 3;
NB = 3;
N = length(u);
M=length(vu);
grad=3;
```

# Anexă

```
% identificare
puteri=generare(grad,NA,NB);

yk=zeros(N,NA);
uk=zeros(N,NB);
ykval=zeros(M,NA);
ukval=zeros(M,NB);
phiid=zeros(N,size(puteri,1));
phival=zeros(N,size(puteri,1));

% d predictie
for n = 1:N
    for m = 1:NA
        if n - m <= 0
            yk(n,m) = 0;
            ykval(n,m)=0;
        else
            yk(n,m) = y(n-m);
            ykval(n,m)= vy(n-m);
        end
    end
    for m = 1:NB
        if n-m <= 0
            uk(n,m) = 0;
            ukval(n,m)=0;
        else
            uk(n,m)= u(n-m+1);
            ukval(n,m)=vu(n-m+1);
        end
    end
end
```

# Anexă

```
d=[yk uk];
dval=[ykval ukval];

for n = 1:N
 for k = 1 : size(puteri,1)
       phiid(n,k) = prod(d(n,:) .^ puteri(k,:));
       phival(n,k) = prod(dval(n,:) .^ puteri(k,:));
 end
end

if grad==1
    phiid(:,1)=0;
    phival(:,1)=0;
end

theta = phiid \ y;


phiidsim=zeros(N,size(puteri,1));
phivalsim=zeros(N,size(puteri,1));
dids = zeros(N,NA+NB);
dvals = zeros(N,NA+NB);
ysimulareid = zeros(N,1);
ysimulareval = zeros(N,1);
regl = 0.01*norm(theta(1:end))^2;
```

# Anexă

```
%simulare
for n = 1:N
    for m = 1:NA
        if n - m <= 1
            dids(n,m) = 0;
            dvals(n,m)= 0;
        else
            dids(n,m) = ysimulareid(n-m);
            dvals(n,m)= ysimulareval(n-m);
        end
    end
    for m = 1:NB
        if n-m <= 1
            dids(n,m+NA) = 0;
            dvals(n,m+NA)=0;
        else
            dids(n,m+NA)= u(n-m);
            dvals(n,m+NA)= vu(n-m);
        end

     end

    for k=1:size(puteri,1)
        phiidsim(n,k)=prod(dids(n, :) .^ puteri(k, :));
        phivalsim(n,k)=prod(dvals(n, :) .^ puteri(k, :));
    end

    ysimulareid(n) =phiidsim(n,:) * theta;
    ysimulareval(n) =phivalsim(n,:) * theta;

    if grad==1
    phiidsim(:,1)=0;
    phivalsim(:,1)=0;
    end
end
```

# Anexă

```
% predictie pt val si id
ypredictieid = phiid * theta;
figure
subplot(2,1,1);
plot(y); hold on; plot(ypredictieid); title('predictie id');
legend('y', 'ypredictie');

ypredictieval = phival * theta;
subplot(2,1,2);
plot(vy); hold on; plot(ypredictieval); title('predictie val');
legend('y', 'ypredictieval');

% Mse pt predictie id
err_predid = y - ypredictieid;
MSE_predid = mean(err_predid.^2);

% Mse pt simulare val
err_simid = y - ysimulareval;
MSE_simid = mean(err_simid.^2);

%afisare mse
fprintf('MSE for prediction id: %f\n', MSE_predid);
fprintf('MSE for simulation id: %f\n', MSE_simid);

% afisare simulare id
figure
plot(y); hold on; plot(ysimulareid); title('simulare id');

% Mse pt predictie val
err_predval = vy - ypredictieval;
MSE_predval = mean(err_predval.^2);

% Mse pt simulare val
err_simval = vy - ysimulareval;
MSE_simval = mean(err_simval.^2);
```

# Anexă

```
%afisare mse
fprintf('MSE for prediction val: %f\n', MSE_predval);
fprintf('MSE for simulation val: %f\n', MSE_simval);

% afisare simulare val
figure
plot(vy); hold on; plot(ysimulareval); title('simulare val');

% Plots
% Identificare:
figure;
subplot(121) % Predictie
plot(y);
hold on;
plot(ypredictieid);
legend('y.id', 'y.id predictie');
title("Identificare: MSE predictie");

subplot(122) % Simulare
plot(y);
hold on;
plot(ysimulareid);
legend('y.id', 'y.id simulare');
title("Identificare: MSE simulare");


% Validare:
figure;
subplot(121); % Predictie
plot(vy);
hold on;
plot(ypredictieval);
legend('y val', 'y predictie');
title("Model vs Predictie: MSE ");
```

# Anexă

```
subplot(122); % Simulare
plot(vy);
hold on;
plot(ysimulareval);
legend('y val', 'y simulare');
title("Model vs Simulare: MSE");

% Validare predictie + simulare
figure;
plot(vy);
hold on;
plot(ypredictieval);
hold on;
plot(ysimulareval);
legend('y val', 'y predictie', 'y simulare');
title('Validare VS Predictie VS Simulare');
```

# Anexă

```
%functie generare
function puteri = generare(grad, na, nb)
    ultrand = zeros(1, na + nb); % Combinatia initiala
    puteri = ultrand;

    nrcomb = grad + 1; % Numarul de combinatii initiale (o singura coloana in matrice)
    nrcombnoi = 0;

    for i = 1 : na + nb
        while sum(ultrand) < grad
            ultrand(i) = ultrand(i) + 1;
            puteri = [puteri; ultrand];

            if i > 1 % Pentru i == 1 nu exista combinatii anterioare
                nrcombnoi = nrcombnoi + 1;
                randnealterat = ultrand;

                for j = 2 : nrcomb % Verifica toate combinatiile anterioare
                    randanterior = puteri(j, :);
                    randanterior = randanterior(1 : i - 1);
                end

                ultrand = randnealterat; % Continua incrementarea cu 1 de la ultimul rand

            end
        end
        nrcomb = nrcomb + nrcombnoi;
        nrcombnoi = 0;
        ultrand(i) = 0;
    end
end
```