# ARX NELINIAR

ALBU CLAUDIU-VASILE

BISTRISCHI ATTILA-ROLAND

PUIA SORIN-VLAD

GRUPA 30135

# INTRODUCERE

Acest proiect implică  generarea unui model ARX (AutoRegressive with eXogenous inputs) neliniar configurabil,definirea ordinului și gradului, colectarea și preprocesarea datelor, estimarea parametrilor modelului și validarea acestuia pe seturi separate de identificare și validare.

Se va ține cont că  ordinul sistemului nu este mai mare de 3, și dinamica poate fi neliniara, iar ieșirea poate fi afectată de zgomot.

Algoritmul este aplicat în două moduri: predicție și simulare

# METODA DE REZOLVARE

Pentru obținerea modelului ARX neliniar de tip polinomial de grad configurabil vom utiliza regresia polinomială, predicția și simularea.

ARX neliniar : $y(k)=g(y(k-1),...,y(k-na),u(k-1), u(k-2),...,u(k-nb); \theta)+e(k)$

$\theta$-parametru; $\theta \in \boldsymbol{R}^n$

Predictia cu un pas înainte $\hat{y}$: $d(k)=[y(k-1),...,y(k-na), u(k-1),...,u(k-nb)]^T$

Simulare $\tilde{y}$: $\tilde{d}(k) = [\tilde{y}(k-1), \dots, \tilde{y}(k-na),u(k-1),...,u(k-nb)]^T$

# PAȘII DE REZOLVARE

1.Generarea puterilor [p1, p2, …], utilizând formula $x_1^{p_1} * x_2^{p_2} * \ldots x_n^{p_n}$,

$\sum_{i=1}^{n} p_i \leq$m, stiind ca avem $q = \frac{(na+nb+m)!}{m! * (na+nb)!}$ regresori

2.Generarea matricei de regresori PHI

3.Găsirea parametrilor prin rezolvarea regresiei liniare

4.Calcularea MSI

5.Găsirea celui mai bun na, nb, m

# GRAFICE PENTRU PREDICȚIE ȘI SIMULARE

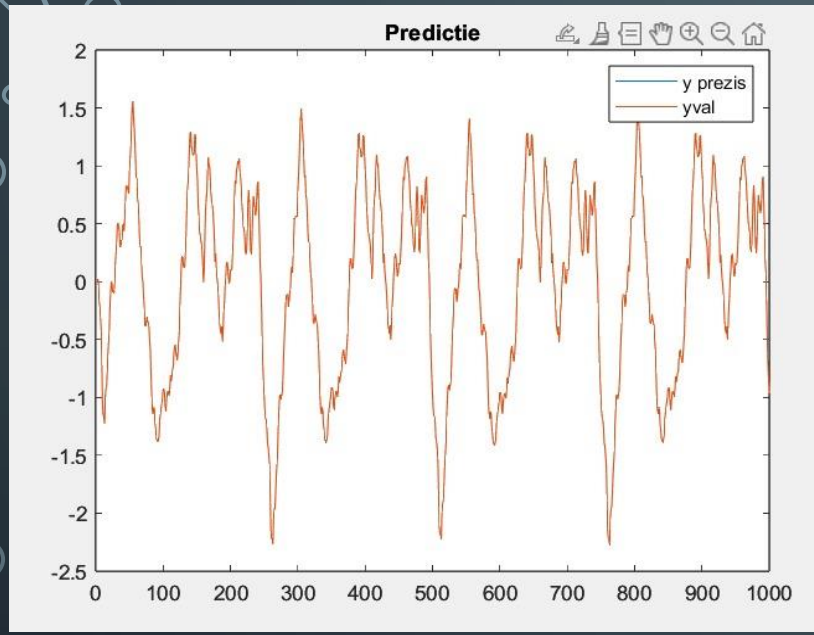


Fig.01, Predictie

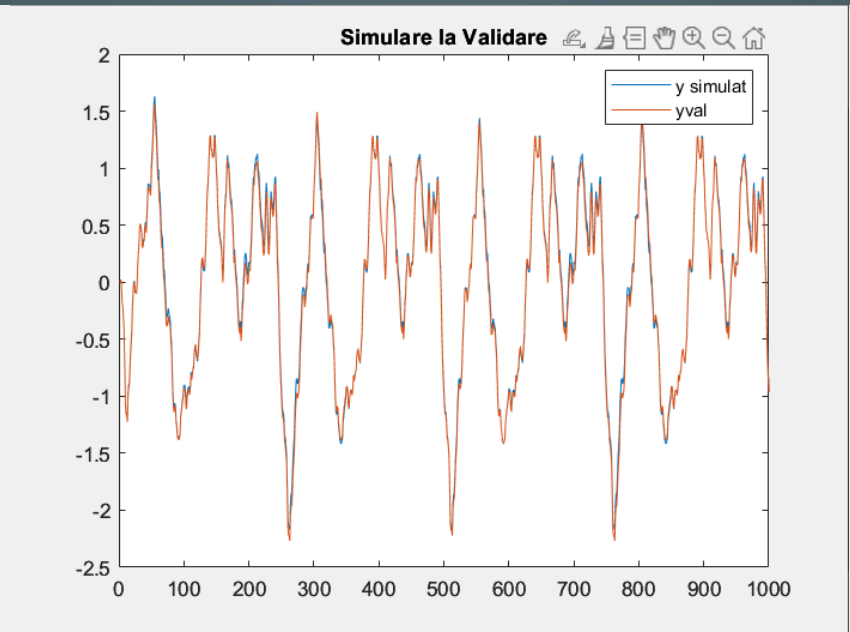


Fig.02, Simulare la Validare

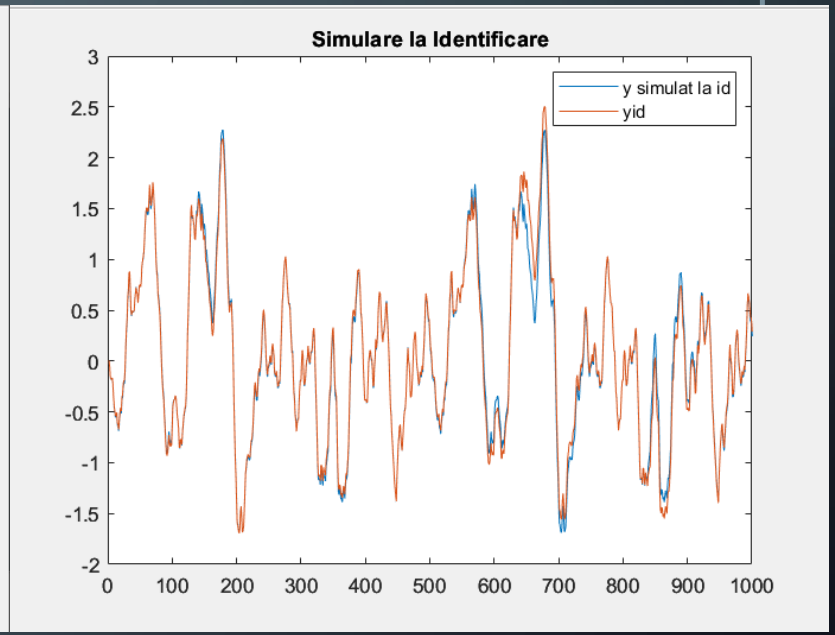


Fig.03, Simulare la Identificare

# CALCULAREA ERORIILOR

La calcularea eroriilor se poate observa că eroarea predicției și simulării au parametrii diferiți.

La predicție eroarea cea mai mica este de 0.000001 la m=3, na=nb=3

La simularea pe validare eroarea cea mai mica este de 0.002404 la m=3, na=nb=2

La simularea pe identificare eroarea ce mai mica este de 0.010440 la m=4 si na=nb=2

Erorile la predictie

| Grad M | Ordin Na si Nb | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| 1 | 0.013399 | 8.7014e-06 | 1.0943e-06 | 1.0815e-06 |
| 2 | 0.01348 | 1.3269e-05 | 1.5652e-06 | 1.6043e-06 |
| 3 | 0.013513 | 1.9416e-06 | 9.3325e-07 | 1.4541e-06 |
| 4 | 0.013404 | 2.813e-06 | 4.2305e-06 | 6.1145e-05 |

Fig.04, Erori la predictie

Erorile la simulare la validare

| Grad M | Ordin Na si Nb | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| 1 | 0.69877 | 0.023234 | 0.022011 | 0.023634 |
| 2 | 0.70218 | 0.020948 | 0.010676 | 0.0070382 |
| 3 | 0.67533 | 0.0024044 | 0.0040989 | NaN |
| 4 | 0.66511 | 0.0027186 | NaN | NaN |

Fig.05, Erori la simulare pe validare

Erorile la simulare la identificare

| Grad M | Ordin Na si Nb | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| 1 | 0.72011 | 0.016547 | 0.016547 | 0.016138 |
| 2 | 0.73132 | 0.013445 | 0.016964 | 0.02417 |
| 3 | 0.6946 | 0.011118 | 0.019399 | 0.018975 |
| 4 | 0.68989 | 0.01044 | 0.052369 | NaN |

Fig.06, Erori la simulare pe identificare

# CONCLUZIE

În  concluzie, am reusit să identificăm un model de tip cutie neagra prin metoda ARX neliniar de tip polinomial, aflând gradul si parametrii potriviți, unde erorile la predicție si simulare sunt minime. Deși este nevoie sa alegem cu atenție parametrii pentru a avea un timp de execuție bun și o acuratețe mai bună.

# CODUL

```
%% arx neliniar cu na diferit de nb, eroare medie patratica
close all; clear;clc;
load('iddata-19.mat');

uid=id.u;
yid=id.y;

uval=val.u;
yval=val.y;

figure
plot(id);
figure
plot(val);

%Gradul si ordinul
na=2;
nb=3;
m=3;

nk=0;
```

```
nk=0;
N=length(uid);

% Identificare
RU=[];
for i=1:N
    for j=1:na
        if i>j
            RU(i,j)=-yid(i-j);
        else
            RU(i,j)=0;
        end
    end
end

for i=1:N
    for j=1:nb
        if i>j
            RU(i,j+na)=uid(i-j);
        else
            RU(i,j+na)=0;
        end
    end
end
```

```
% Generarea toate combinatiile
qqqqq=factorial(na+nb+m)/(factorial(m)*factorial(na+nb));
totalVariabile = na + nb;

for a = 0:((m+1)^(totalVariabile))-1
    combinatie = dec2base(a, m+1) - '0';
    suma_pe_linii(a+1) = sum(combinatie);
end

matricePuteri = dec2base(find(suma_pe_linii <= m) - 1, m+1) - '0';

phi=[];
for i=1:N
    phi(i,:)=generarePuteri(RU(i,:),matricePuteri);
end
teta=phi\yid;
```

```
% Validare
for i=1:N
    for j=1:na
        if i>j
            RUval(i,j)=-yval(i-j);
        else
            RUval(i,j)=0;
        end
    end
end

for i=1:N
    for j=1:nb
        if i>j
            RUval(i,j+na)=uval(i-j);
        else
            RUval(i,j+na)=0;
        end
    end
end

phival=[];
for i=1:N
    phival(i,:)=generarePuteri(RUval(i,:),matricePu
end

% Predictie
yhat=phival*teta;

e=yhat-yval;
emp=1/length(e)*sum(e.^2);
fprintf('Eroarea la predictie este de: %f\n',emp);
```

```
% Simulare la validare
nk=0;
phival=[];
RUval=zeros(1,na+nb);
Nval=length(uval);
y_tilt=zeros(Nval,1);
phival(1,:)=generarePuteri(RUval(1,:),matricePuteri);

for k=2:Nval

    for j=1:na
        if k>j
            RUval(k,j)=-y_tilt(k-j);
        else
            RUval(k,j)=0;
        end
    end

    for j=1:nb
        if k>j
            RUval(k,j+na)=uval(k-j);
        else
            RUval(k,j+na)=0;
        end
    end

    phival(k,:)=generarePuteri(RUval(k,:),matricePuter
    y_tilt(k)=phival(k,:)*teta;
end

e=y_tilt-yval;
emp=1/length(e)*sum(e.^2);
fprintf('Eroarea la simulare la validare: %f\n',emp)
```

```
RU=zeros(1,na+nb);
Nid=length(uid);
y_tilt_id=zeros(Nid,1);
phi(1,:)=generarePuteri(RU(1,:),matricePuteri);

for k=2:Nid

    for j=1:na
        if k>j
            RU(k,j)=-y_tilt_id(k-j);
        else
            RU(k,j)=0;
        end
    end

    for j=1:nb
        if k>j
            RU(k,j+na)=uid(k-j);
        else
            RU(k,j+na)=0;
        end
    end

    phi(k,:)=generarePuteri(RU(k,:),matricePuteri);
    y_tilt_id(k)=phi(k,:)*teta;
end

e_id=y_tilt_id-yid;
emp_id=1/length(e_id)*sum(e_id.^2);
fprintf('Eroarea la simulare la identificare:
%f\n',emp_id)
```

```
%% arx neliniar na=nb, eroare minima
clear;
close all;
load('iddata-19.mat');
uid=id.u;
yid=id.y;

uval=val.u;
yval=val.y;
nk=0;
EMP_Predictie=0;
EMP_Similare=0;
lung=1;
N=length(uid);
minim(1,:)=[1000,0,0,0];
minim(2,:)=[1000,0,0,0];
minim_id(1,:)=[1000,0,0,0];
minim_id(2,:)=[1000,0,0,0];
ysimulat=[];
yprezis=[];
```

```
% Gradul polinomului
for m=1:4
% Ordinul lui na si nb
    for na=1:4
        nb=na;

        matricePuteri=[];
        phi=[];
% Identificare
        RU=[];
        for i=1:N
            for j=1:na
                if i>j
                    RU(i,j)=-yid(i-j);
                else
                    RU(i,j)=0;
                end
            end
        end

        for i=1:N
            for j=1:nb
                if i>j
                    RU(i,j+na)=uid(i-j);
                else
                    RU(i,j+na)=0;
                end
            end
        end
```

```
% Numarul total de combinari

qqqqq=factorial(na+nb+m)/(factorial(m)*factorial(na+nb));
        suma_pe_linii=zeros(1,qqqqq);
        totalVariabile = na + nb;
        quatrice=zeros(1,qqqqq);
        i=1;
        a=-1;
        for rrr = 0:((m+1)^(totalVariabile))-1
            a=a+1;
            combinatie = dec2base(a, m+1) - '0';
            wwww= sum(combinatie);
            if wwww<=m
                quatrice(i)=a;
                i=i+1;
            else
                sum_de_skip=0;
                zzz=length(combinatie);
                iesi=0;
                for x=1:zzz
                    sum_de_skip=sum_de_skip+combinatie(x);
                    if(sum_de_skip>m)
                        iesi=1;
                        carry = 1;

                        for c = x:-1:1
                            suma = combinatie(c) + carry;
                            combinatie(c) = mod(suma, m+1);
                            carry = floor(suma / (m+1));
                            if carry == 0
                                break;
                            end
                        end
```

```
if carry > 0
                            combinatie = [carry, combinatie];
                        end
                        combinatie(x+1:zzz)=0;
                        a=polyval(combinatie, m+1)-1;
                    end
                    if iesi==1
                        break;
                    end
                end
            end

            if i>qqqqq
                break;
            end
        end
        matricePuteri=dec2base(quatrice , m+1) - '0';
% Generare matrice puteri
        phi=[];
        for i=1:N
            phi(i,:)=generarePuteri(RU(i,:),matricePuteri);
        end

        teta=phi\yid;
```

```
% Validare
        for i=1:N
            for j=1:na
                if i>j
                    RUval(i,j)=-yval(i-j);
                else
                    RUval(i,j)=0;
                end
            end
        end

        for i=1:N
            for j=1:nb
                if i>j
                    RUval(i,j+na)=uval(i-j);
                else
                    RUval(i,j+na)=0;
                end
            end
        end

        phival=[];
        for i=1:N
            phival(i,:)=generarePuteri(RUval(i,:),matricePuteri);
        end
%Predictie
        yhat=phival*teta;

        e=yhat-yval;
        emp=1/length(e)*sum(e.^2);
        EMP_Predictie(m,na)=emp;
        if(emp<minim(1,1))
            minim(1,:)=[emp,m,na,nb];
            yprezis=yhat;
        end
```

```
% Simulare validare
        nk=0;
        phival=[];
        RUval=zeros(1,na+nb);
        Nval=length(uval);
        y_tilt=zeros(Nval,1);
        phival(1,:)=generarePuteri(RUval(1,:),matriceP

        for k=2:Nval

            for j=1:na
                if k>j
                    RUval(k,j)=-y_tilt(k-j);
                else
                    RUval(k,j)=0;
                end
            end

            for j=1:nb
                if k>j
                    RUval(k,j+na)=uval(k-j);
                else
                    RUval(k,j+na)=0;
                end
            end

            phival(k,:)=generarePuteri(RUval(k,:),matr
            y_tilt(k)=phival(k,:)*teta;
        end

        e=y_tilt-yval;
        emp=1/length(e)*sum(e.^2);
        EMP_Similare(m,na)=emp;

        if(emp<minim(2,1))
            minim(2,:)=[emp,m,na,nb];
            ysimulat=y_tilt;
        end
```

```
% Simulare identificare

        phi=[];
        RU=zeros(1,na+nb);
        Nid=length(uid);
        y_tilt_id=zeros(Nid,1);
        phi(1,:)=generarePuteri(RU(1,:),matricePuteri);

        for k=2:Nid

            for j=1:na
                if k>j
                    RU(k,j)=-y_tilt_id(k-j);
                else
                    RU(k,j)=0;
                end
            end

            for j=1:nb
                if k>j
                    RU(k,j+na)=uid(k-j);
                else
                    RU(k,j+na)=0;
                end
            end

            phi(k,:)=generarePuteri(RU(k,:)
            y_tilt_id(k)=phi(k,:)*teta;
        end

        e_id=y_tilt_id-yid;
        emp_id=1/length(e_id)*sum(e_id.^2);
        EMP_Similare_id(m,na)=emp_id;

        if(emp_id<minim_id(2,1))
            minim_id(2,:)=[emp_id,m,na,nb];
            ysimulat_id=y_tilt_id;
        end

        lung=lung+1;
    end
end
```

```
% Functia pentru generarea puterilor
function puteri=generarePuteri(X,matricePuteri) %returneza liniile din
matricea phi

for i=1:length(matricePuteri(:,1))
    q=X.^matricePuteri(i,:);
    puteri(i)=prod(q);
end
end
```