PROIECT PARTEA II

PIDX: **11/11**

Abrudan Lavinia-Maria
Malutan Alexandra

Introducere

- Dezvoltarea unui model folosind o structura ARX neliniara.
- Sistem dinamic cu o intrare si o iesire.
- Programarea procedurii de regresie pentru identificarea parametrilor.
- Utilizarea modelului cu intrari noi.

Numarul de termeni

$$C_{N+m}^{m}$$

N- nr total de variabile (na+nb)
m- gradul maxim

- na=nb=1

m=1 => $y(k) = 1 + a \cdot y(k-1) + b \cdot u(k-1)$

=> 3 termeni

m=2 => $y(k) = 1 + a \cdot y(k-1) + b \cdot u(k-1) + c \cdot y(k-1)^2 + v \cdot u(k-1)^2 + w \cdot u(k-1) \cdot y(k-1)$

=> 6 termeni

m=3 => $y(k) = 1 + a \cdot y(k-1) + b \cdot u(k-1) + c \cdot y(k-1)^2 + v \cdot u(k-1)^2 + w \cdot u(k-1) \cdot y(k-1) + d \cdot y(k-1)^3 + e \cdot u(k-1)^3 + f \cdot y(k-1)^2 \cdot u(k-1) + g \cdot u(k-1)^2 \cdot y(k-1)$

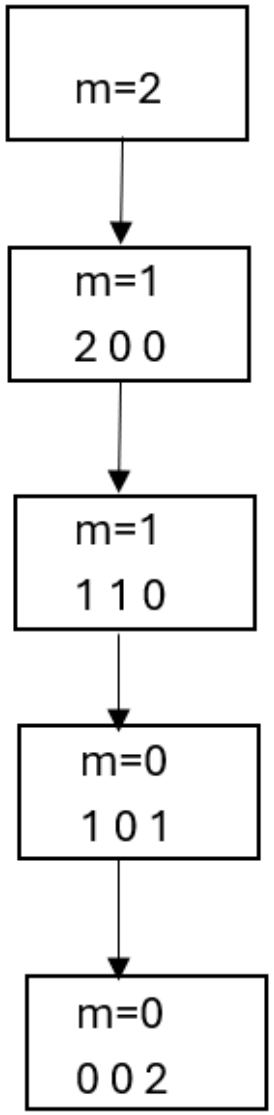
=> 10 termini

...

Procedura de gasire a parametrilor

Pentru m=3, na=nb=nk=1 :

$$\Phi=\begin{bmatrix} 1 & -y(0) & x(0) & (-\mathrm{y}(0))^2 & -y(0)x(0) & x(0)^2 & (-y(0))^3 & (-y(0))^2x(0) & -y(0)x(0)^2 & x(0)^3 \\ 1 & -y(1) & x(1) & (-\mathrm{y}(1))^2 & -y(1)x(1) & x(1)^2 & (-y(1))^3 & (-y(1))^2x(1) & -y(1)x(1)^2 & x(1)^3 \\ \vdots & & & & & & & & & \vdots \\ 1 & -y(n-1) & x(n-1) & (-\mathrm{y}(n-1))^2 & & & & & -y(n-1)x(\mathrm{n}-1)^2 & x(\mathrm{n}-1)^3 \end{bmatrix}$$



Avantajele folosirii metodei Backtracking

- Generarea simetrica a tuturor combinatiilor posibile de exponenti care respecta dimensiunea gradului maxim al polinomului, m.

- Se asigura ca nu se omit solutii posibil valide si ca fiecare combinatie de exponenti este creata.

Creearea vectorului Θ

- Parametrii theta i-am determinat prin rezolvarea ecuatiei de regresie liniara.

- L-am antrenat pe datele de identificare si l-am utilizat ulterior pentru a valida modelul si pentru simulare.

$$\theta = \phi \quad \backslash \, \mathrm{i\,dy}$$

Predictia

ϕ=polinom_generat(u,y,l,na,nb,nk);

Θ= ϕ \y;

$\hat{y}$= ϕ * Θ;

- Este un pas important pentru a verifica identificarea corecta a modelului si cu ajutorul MSE putem obtine cea mai apropiata aproximare.

Simularea

- Simularea necesita stari initiale pentru a porni, initializand aproximatoarele cu primele valori ale iesirilor reale.

$\tilde{\phi}$=polinom_generat(u, $\tilde{y}$, l, na, nb, nk);

$\tilde{y}$(i)= $\tilde{\phi}$ * Θ;

Determinarea celui mai bun grad m

m=2, na=nb=2

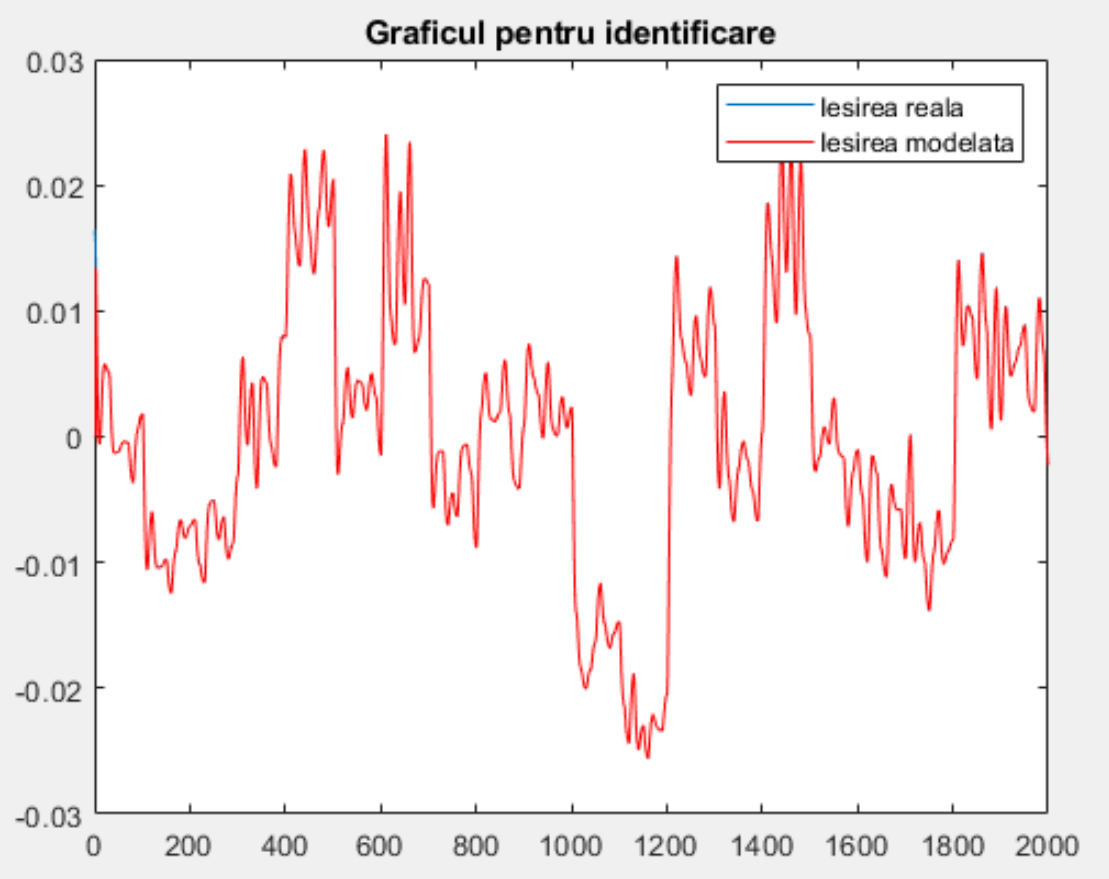
- Divergenta modelului:

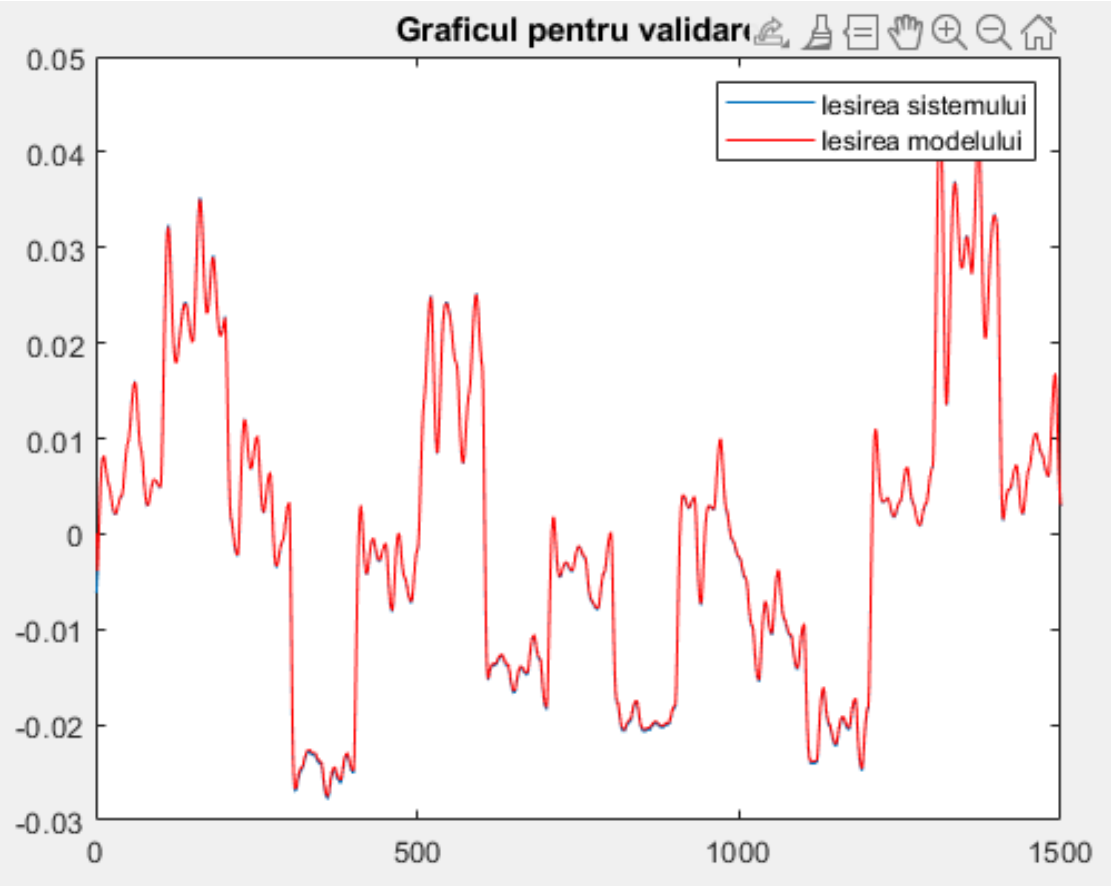
  Parametrii (θ) pot contine valori foarte mari sau necorespunzatoare, ducand la rezultate instabile (cum ar fi impartiri la zero sau operatii numerice care genereaza NaN).

- Termeni Polinomiali Instabili:

  Cand gradul m este ridicat (cum este cazul cu m=6), poate duce la depasirea limitelor numerice. Apar erori de stabilitate in procesul de regresie.

Predictia sistemului si MSE

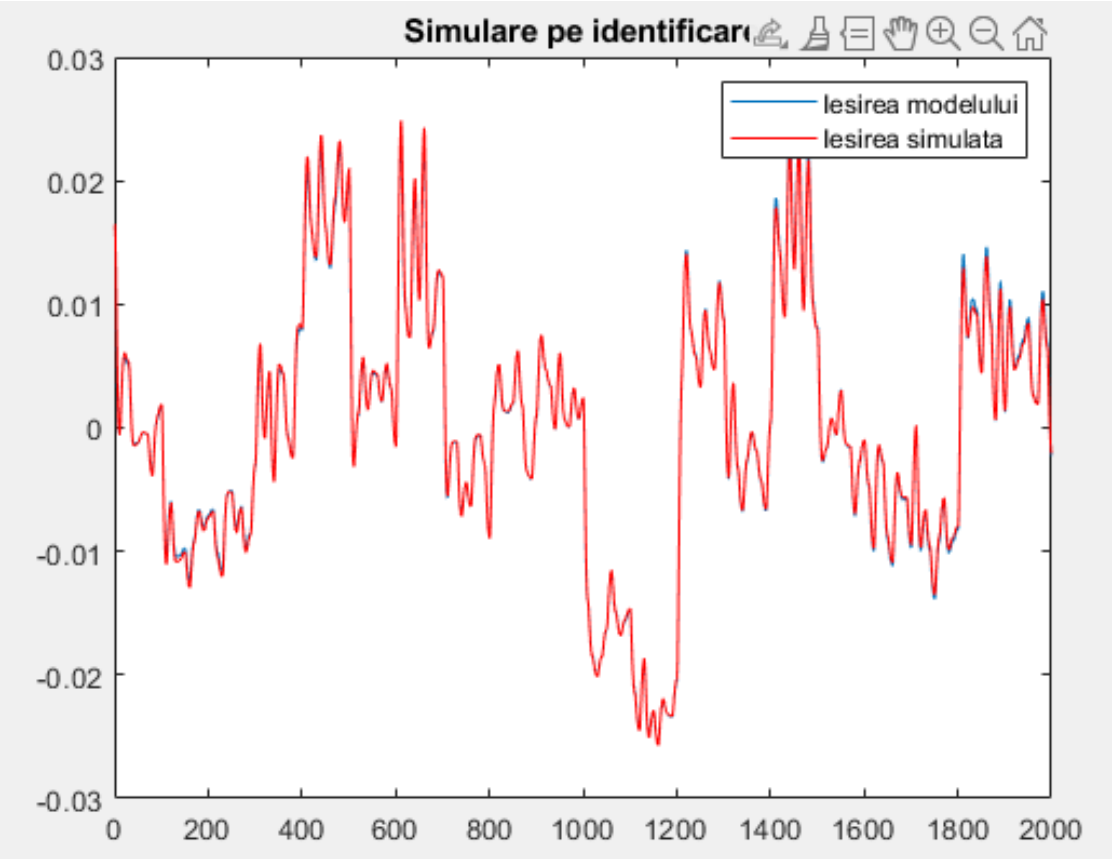


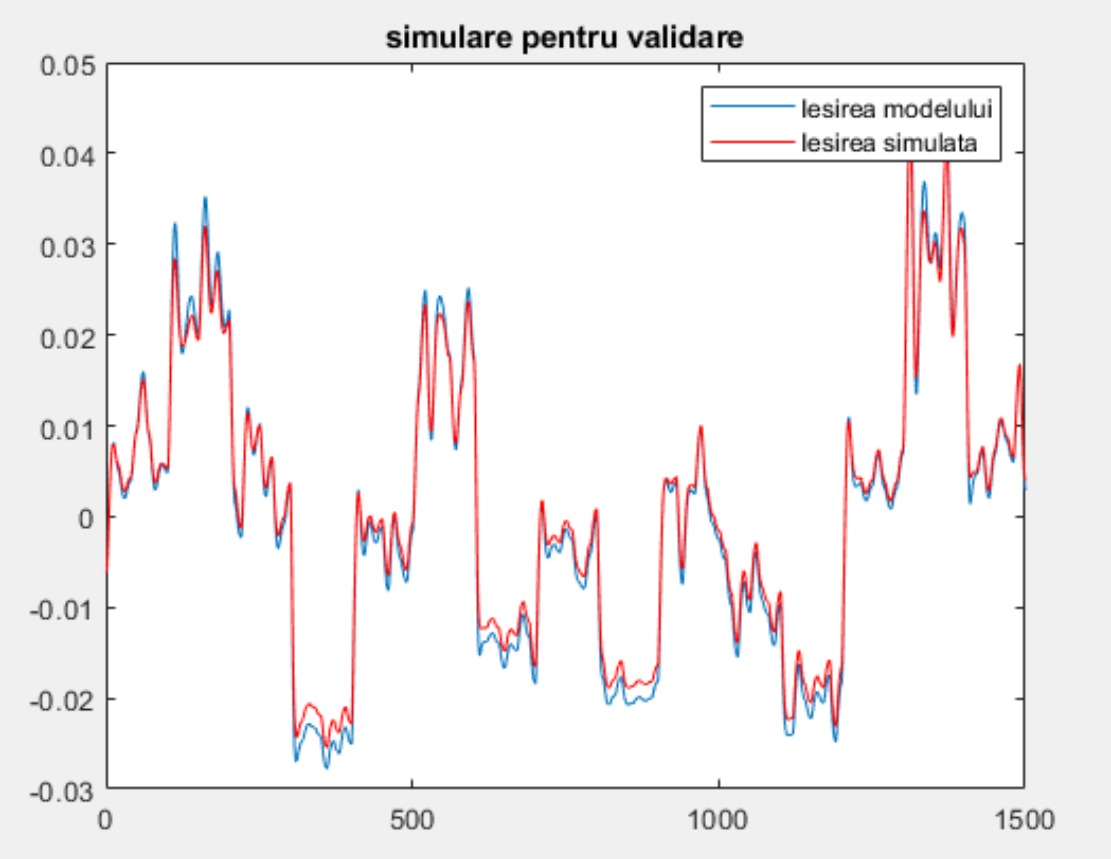



```
Eroare MSE pe datele de identificare:
    2.5689e-07
```

```
Eroare MSE pe datele de validare:
    6.7831e-08
```

Simularea modelului si MSE



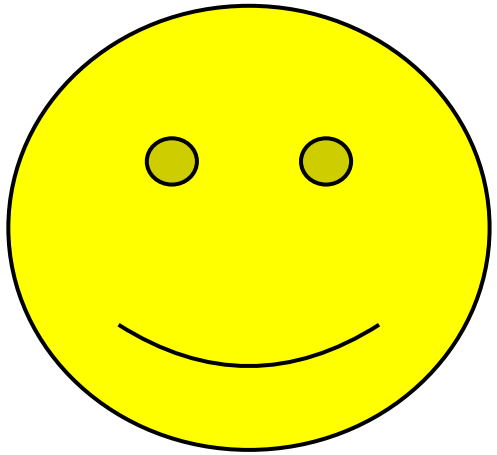



```
Eroare MSE pentru simularea pe datele de identificare:
    7.1395e-08
```

```
Eroare MSE pentru simularea pe datele de validare:
    1.9570e-06
```

Concluzie

- Proiectul realizeaza identificarea unui model ARX neliniar folosind metoda backtracking.



Anexa cu listingul codului

```
% incarcarea fisierelor cu datele primite
load('iddata-11.mat')
figure;
plot(id.InputData{1}), title("Intrarea sistemului pentru datele de identificare")
figure;
plot(id.OutputData{1}), title("Iesirea sistemului pentru datele de identificare")

figure;
plot(val.InputData{1}), title("Intrarea sistemului pentru datele de validare")
figure;
plot(val.OutputData{1}), title("Iesirea sistemului pentru datele de validare")

na=2;
nb=2;
nk=1;
m=2;

idU=id.InputData{1};
idu=detrend(idU);
valU=val.InputData{1};
valu=detrend(valU);
idY=id.OutputData{1};
idy=detrend(idY);
valY=val.OutputData{1};
valy=detrend(valY);
```

```
valoaremseid=zeros(1,m);
valorimseval=zeros(1,m);
valoaremsesimidentificare=zeros(1,m);
valoaremsesimvalidare=zeros(1,m);
%pentru simulare
Nid=length(idy);
ysimulat_id=zeros(Nid, 1);
ysimulat_id(1:na)=idy(1:na);
Nval=length(valy);
ysimulat_val=zeros(Nval,1);
ysimulat_val(1:na)=valy(1:na);
```

```
for l=1:m
    %pentru identificare
    phi=polinom_generat(idu,idy,l,na,nb,nk);
    theta=phi\idy;
    yaproximat=phi*theta;
    suma=0;
    for i=1:length(idu)
        suma=suma+(yaproximat(i)-idy(i)).^2;
    end
    mseid=(1/length(idu))*suma;
    valoaremseid(l)=mseid;

    %pentru validare
    phivalidare=polinom_generat(valu,valy,l,na,nb,nk);
    yaproximatvalidare=phivalidare*theta;
    suma=0;
    for i=1:length(valu)
        suma=suma+(yaproximatvalidare(i)-valy(i)).^2;
    end
    mseval=(1/length(valu))*suma;
    valorimseval(l)=mseval;

    %pentru simulare
    phi_simulat=zeros(Nid, size(phi, 2));
    for i=1+na:Nid
        phi_simulat=polinom_generat(idu(1:i), ysimulat_id(1:i), l, na, nb, nk);
        ysimulat_id(i)=phi_simulat(end, :)*theta;
    end
    suma=0;
    for i=1:Nid
        suma=suma+(ysimulat_id(i)-idy(i)).^2;
    end
    msesimidentificare=(1/Nid)*suma;
    valoaremsesimidentificare(l)=msesimidentificare;

    phi_simulat_val=zeros(Nval, size(phivalidare, 2));
    for i=1+na:Nval
    phi_simulat_val=polinom_generat(valu(1:i), ysimulat_val(1:i), l, na, nb, nk);
    ysimulat_val(i)=phi_simulat_val(end,:)*theta;
    end
    suma=0;
    for i=1:Nval
        suma=suma+(ysimulat_val(i)-valy(i)).^2;
    end
    msesimvalidare=(1/Nval)*suma;
    valoaremsesimvalidare(l)=msesimvalidare;
end
```

```
figure,
plot(1:m, valoaremseid, LineWidth=2),
hold on, plot(1:m,valorimseval,LineWidth=2),
hold on, plot(1:m,valoaremsesimidentificare, LineWidth=2),
hold on, plot(1:m,valoaremsesimvalidare, LineWidth=2);
title('MSE'),
legend('MSE pentru identificare', 'MSE pentru validare', ...
  'MSE pe simulare identificare','MSE pe simulare validare');

disp("Eroare MSE pe datele de identificare:")
disp(mseid)
figure
plot(1:m, valoaremseid), title("MSE pentru identificare");

disp("Eroare MSE pe datele de validare:")
disp(mseval)
figure;
plot(1:m,valorimseval), title("MSE pentru validare");

disp("Eroare MSE pentru simularea pe datele de identificare:")
disp(msesimidentificare)
figure;
plot(1:m,valoaremsesimidentificare), title("MSE pe simulare identificare");


disp("Eroare MSE pentru simularea pe datele de validare:")
disp(msesimvalidare)
figure;
plot(1:m,valoaremsesimvalidare), title("MSE pe simulare validare");
```

```
figure;
plot(idy)
hold on;
plot(yaproximat, 'r-')
legend('Iesirea reala','Iesirea modelata')
title('Graficul pentru identificare');

figure;
plot(valy)
hold on;
plot(yaproximatvalidare, 'r-')
legend('Iesirea sistemului','Iesirea modelului')
title('Graficul pentru validare');

figure;
plot(idy)
hold on;
plot(ysimulat_id, 'r-')
legend('Iesirea modelului','Iesirea simulata')
title('Simulare pe identificare');

figure;
plot(valy)
hold on;
plot(ysimulat_val, 'r-')
legend('Iesirea modelului','Iesirea simulata')
title('simulare pentru validare');
```

```
function phi = polinom_generat(x, y, m, na, nb, nk)
    dim=length(y);
    combinatii=generare_combinatii(na, nb, m);
    nrtermeni=size(combinatii, 1);
    phi=zeros(dim, nrtermeni);

    %incepem sa construim matricea phi, pe fiecare rand
    for i=1:dim

        for j=1:nrtermeni
            exponent=combinatii(j, :);
            termen=1;

            %calculam termenii pentru iesirile intarziate pana la ordinul na
            for k=1:na
                if (i-k>0)
                    termen=termen*(-y(i-k)^exponent(k));
                else
                    termen=0;
                end
            end

            %calculez termenii pentru intrarile intarziate
            for k=1:nb
                if (i-nk-k+1>0)
                   termen=termen*(x(i-nk-k+1)^exponent(na+k));
                else
                   termen=0;
                end
            end
            phi(i, j)=termen;
         end
     end
end
```

```
function combinatii=generare_combinatii(na, nb, m)
    numar_total_variabile=na+nb;
    combinatii=[];
    combinatie_curenta=zeros(1,numar_total_variabile);

    function backtracking(combinatie_curenta, pozitie, suma_ramasa)

        if pozitie>numar_total_variabile
            if suma_ramasa>=0
                combinatii=[combinatii; combinatie_curenta];
            end
            return;
        end

        for i=0:suma_ramasa
            combinatie_curenta(pozitie)=i;
            backtracking(combinatie_curenta, pozitie+1, suma_ramasa-i);
        end
    end
    backtracking(combinatie_curenta, 1, m);
end
```