System Identification Project Part.2 Non-Linear Arx

Olaru Ariana-Casandra

Index 11
Faculty of Automation and Computer Science
Technical University of Cluj-Napoca

June 7, 2024

Contents

Introduction

Introduction

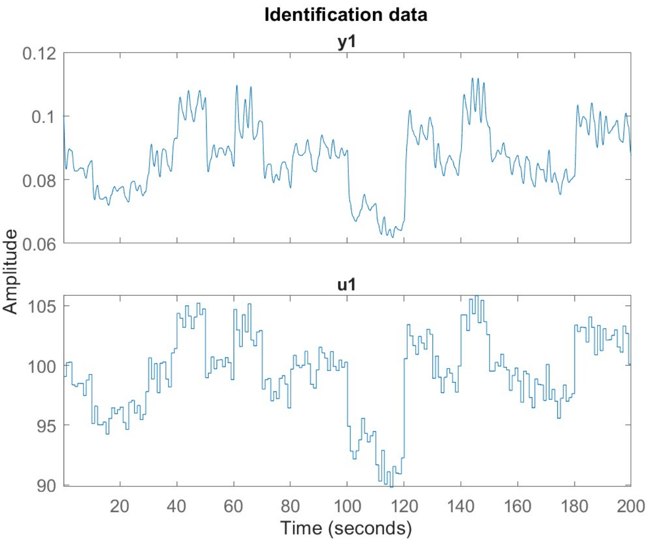
Developing a black box model for a dynamic system, using a polynomial, nonlinear ARX, for adaptive model orders, delay and polynomial degree.

na, nb – model orders
($na = nb$)

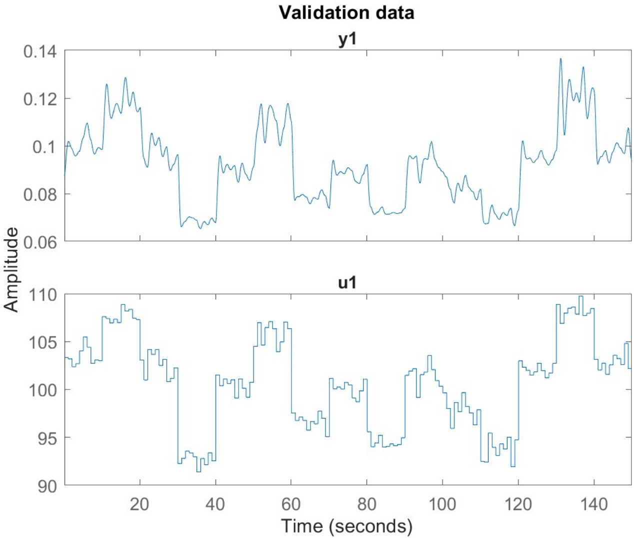
nk – delay
($nk = 1$)

m – polynomial degree

Identification Data



Validation Data



Algorithm

Algorithm

- Loading the data and extracting them
- Generating the PHY matrix (regressors) and THETA (parameters)
- Computing the mse and the finding the minimum mse for each case

Algorithm

$$
\begin{array}{cc}
1 \ldots na & na+1 \ldots 2na \\
\left[\begin{array}{c|c} y(i-j-nk) & u(i-j-nk) \end{array}\right] &
\end{array}
$$

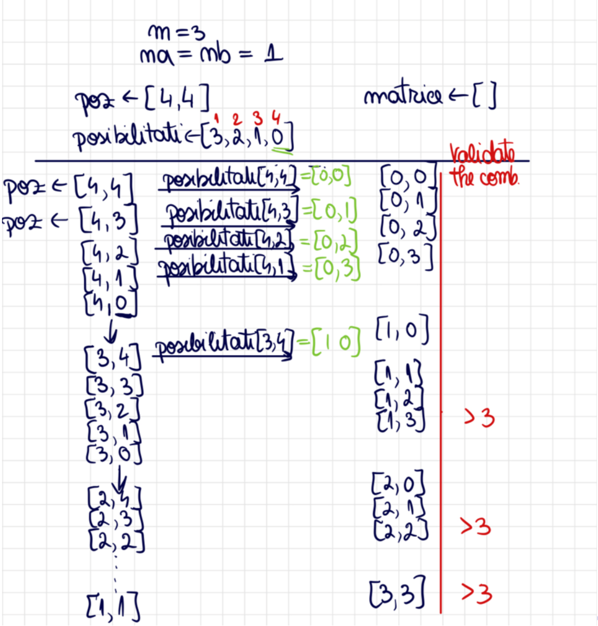
$$phy = \left[\begin{array}{c|c} y(i-j-nk) & u(i-j-nk) \end{array}\right]$$

$$PHY = phy^{matrice}$$

$$theta = PHY \setminus Y$$

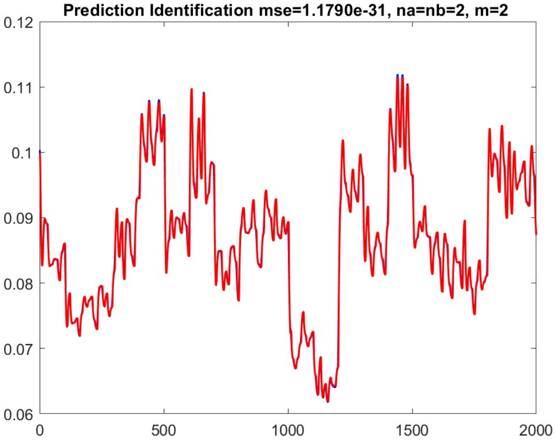
$$yhat = PHY * theta$$

Generating Combinations

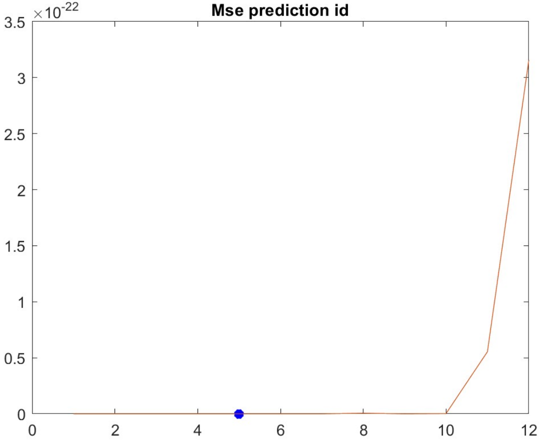


Results

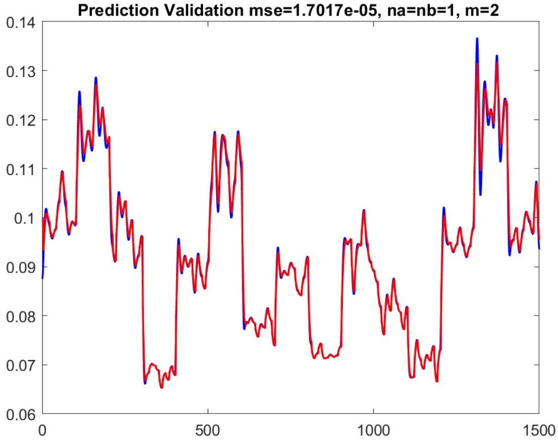
Prediction Identification



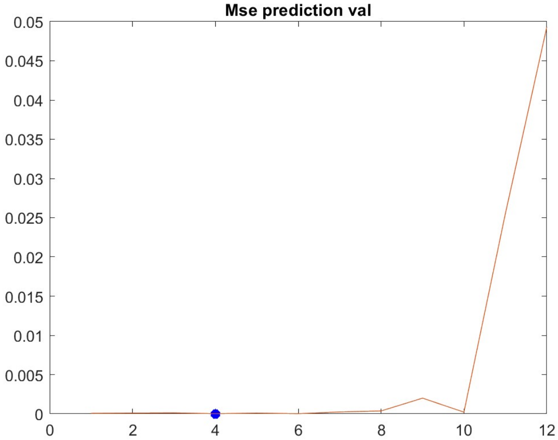
MSE Prediction Identification



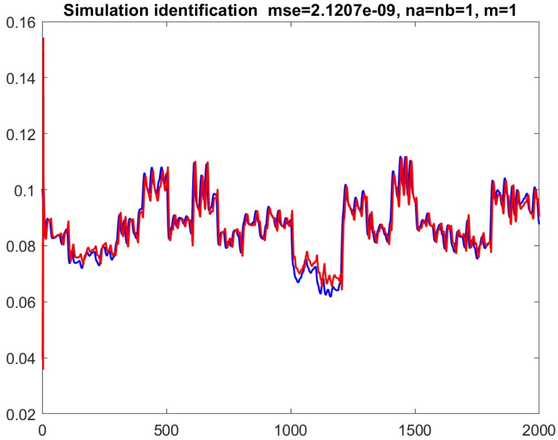
Prediction Validation



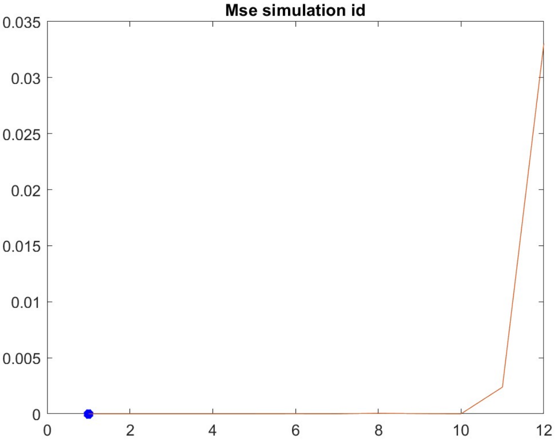
MSE Prediction Validation



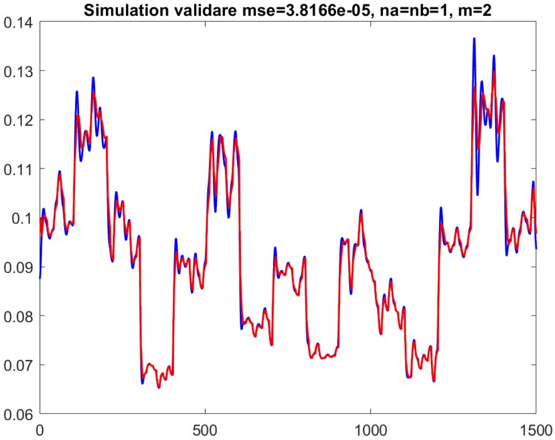
Simulation Identification



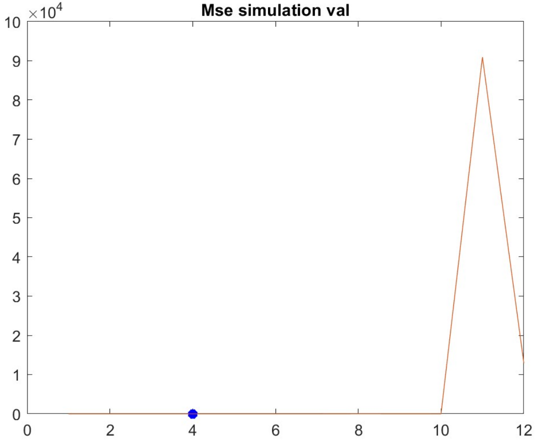
MSE Simulation Identification



Simulation Validation



MSE Simulation Validation



Conclusion

Conclusion

- Optimal values for na, nb, and m help in achieving the most accurate approximation of the graphs.
- Simulation is closer to reality, and it is more precise.
- Notably, when the parameter $M = 1$, the nonlinear ARX model reduces to a linear ARX model, making it suitable for systems with linear characteristics. This adaptability makes the ARX model a versatile tool for both linear and nonlinear system modeling.

Appendix

Appendix

```matlab
clear all;
clc;
load('iddata-11.mat')


plot(id)
title("Identification data")
u_id = id.InputData;
y_id = id.OutputData;

figure()
plot(val)
title("Validation data")
u_val = val.InputData;
y_val = val.OutputData;

matrice_erori = [];
contor_mse = 1; %index for mse
nk = 1; %delay
cazuri_bune = inf(4,3); %mse na m separat yhat

for m = 1:4
    for na = 1:3
        %pred id
        matrice = puteri(2*na,m);
        PHY_ID = PhyPred(u_id,y_id,na,matrice,nk);
```

```
        theta = PHY_ID \ y_id;
        yhat = PHY_ID*theta;
        matrice_erori(1,contor_mse) = 1/length(y_id)*sum(y_id-yhat).^2;
        if matrice_erori(1,contor_mse)< cazuri_bune(1,1)
            cazuri_bune(1,1) = matrice_erori(1,contor_mse);
            cazuri_bune(1,2) = na;
            cazuri_bune(1,3) = m;
            YHAT = yhat;
        end
        %pred val
        PHY_val = PhyPred(u_val,y_val,na,matrice,nk);
        yhat_val = PHY_val*theta;
        matrice_erori(2,contor_mse) = 1/length(y_val)*sum(y_val-yhat_val).^2;
        if matrice_erori(2,contor_mse)< cazuri_bune(2,1)
            cazuri_bune(2,1) = matrice_erori(2,contor_mse);
            cazuri_bune(2,2) = na;
            cazuri_bune(2,3) = m;
            YHAT_VAL = yhat_val;

        end
        %simu id
        [yhat_sim_id] = PhySimu(u_id,yhat,na,matrice,theta,nk);
        matrice_erori(3,contor_mse) = 1/length(y_id)*sum(y_id-yhat_sim_id).^2;
        if matrice_erori(3,contor_mse)< cazuri_bune(3,1)
            cazuri_bune(3,1) = matrice_erori(3,contor_mse);
            cazuri_bune(3,2) = na;
```

```
            cazuri_bune(3,3) = m;
            YHAT_SIM = yhat_sim_id;
        end
        %simu val
        [yhat_sim_val] = PhySimu(u_val,yhat_val,na,matrice,theta,nk);
        matrice_erori(4,contor_mse) = 1/length(y_val)*sum(y_val-yhat_sim_val).^2;
        if matrice_erori(4,contor_mse)< cazuri_bune(4,1)
            cazuri_bune(4,1) = matrice_erori(4,contor_mse);
            cazuri_bune(4,2) = na;
            cazuri_bune(4,3) = m;
            YHAT_SIMV = yhat_sim_val;
        end
        contor_mse = contor_mse+1;

    end


end
%% erori si cele mai bune cazuri (mse,na,m)
afisare(cazuri_bune(1,:),matrice_erori(1,:), YHAT, y_id, 0,'Mse prediction id', "Prediction Identification mse=%.4e, na=nb=%d, m=%d ")
afisare(cazuri_bune(2,:),matrice_erori(2,:), YHAT_VAL, y_val, 0, 'Mse prediction val', "Prediction Validation mse=%.4e, na=nb=%d, m=%d ")
afisare(cazuri_bune(3,:),matrice_erori(3,:), YHAT_SIM, y_id, 0,'Mse simulation id', "Simulation identification  mse=%.4e, na=nb=%d, m=%d")
afisare(cazuri_bune(4,:),matrice_erori(4,:), YHAT_SIMV, y_val, 0, 'Mse simulation val', "Simulation validare mse=%.4e, na=nb=%d, m=%d")

%% Functii
function [PHY] = PhyPred(u,y,na,matrice,nk)
phy = [];
```

```
PHY = [];
  for i = 1:length(u)
    for j = 1:2*na
        if j <= na
           if i-(j+nk) > 0
               phy(i,j) = y(i-(j+nk));
           else
               phy(i,j) = 0;
               break;
           end
        else
            if i-(j+nk) > 0
                phy(i,j) = u(i-(j+nk));

            else
                phy(i,j) = 0;
                break;
            end
        end
    end
  end
  for i = 1:length(u) %parcurg linii phy
      linie = phy(i,:);
     for z = 1:length(matrice) %iau linii combinari
           produs = 1;
          for j = 1:2*na %parcurg coloane
              p = linie(j)^matrice(z,j);
```

```
                produs = produs*p;
            end
            PHY(i,z) = produs;
        end
    end
    %ultima coloana termenul liber
    [PHY(:,1), PHY(:,length(matrice))] = deal(PHY(:,length(matrice)),PHY(:,1));
end

function [yhat_sim] = PhySimu(u,yhat,na,matrice,theta,nk)
linie = [];
PHY = [];
yhat_sim = [];
 for i = 1:length(u)
    for j = 1: 2*na
        if j <= na
            if i-j-nk <= 0
                linie(j)= 0;
            else
                linie(j) = yhat(i-j-nk);
            end
        else
            if i-j-nk <= 0
                linie(j)= 0;
            else
                linie(j) = u(i-j-nk);
            end
```

```
            end
        end

        for z = 1 : length(matrice)
            produs = 1;
            for j = 1 : 2*na
                p = linie(j)^matrice(z,j);
                produs = produs * p;
            end
            PHY(i,z) = produs;
        end
    end
    %termen liber ultimu
    [PHY(:,1), PHY(:,length(matrice))] = deal(PHY(:,length(matrice)),PHY(:,1));


     for i =1:length(PHY)
         yhat_sim(i) = PHY(i,:)*theta;
     end
     yhat_sim = yhat_sim';
   end



   function matrice = puteri(n,m)
   poz = (m+1)*ones(1,n);
   matrice = [];
```

```
posibilitati = m:-1:0;

 while poz ~= zeros(1,n)

    matrice = [matrice; posibilitati(poz)];
    poz(n) = poz(n)-1;

    for j = n:-1:2
        if poz(j) == 0
            poz(j) = m+1;
            poz(j-1) = poz(j-1)-1;
        end

    end
 end
 %sterg combinarile cu grad > m
 matrice = stergeCombinari(matrice,m);

end

function matriceCombinariSterse = stergeCombinari(matrice, m)
    i=1;
    while i<=length(matrice)
        if sum(matrice(i,:)) > m
            matrice(i,:) = [];
        else
            i = i+1;
```

```
            end

        end
        matriceCombinariSterse = matrice;
    end

    function afisare(cazuri,erori, yhat, y, y_verticala, mesaj, mesaj2)
    [minim_mse, poz_mse_id] = min(erori);
    figure()
    plot(poz_mse_id, y_verticala, 'b*', 'LineWidth', 3)
    hold on
    plot(erori);
    title(mesaj)
    figure()
    plot(1:length(y),y,'b','LineWidth',1.2)
    hold on
    plot(1:length(yhat),yhat,'r' ,'LineWidth',1.2)
    title(sprintf(mesaj2,cazuri(1), cazuri(2), cazuri(3)))
    end
```

Thank you!