# EE 551 Estimation and Identification

MEET PATEL (194102507)

19Nov2019

## 1 FROLS Algorithm to fit an NARX Model for the Data

Given model

$$y(k) = -0.605y(k-1) - 0.163y^2(k-2) + 0.588u(k-1) - 0.240u(k-2) + \xi(k)$$

where $\xi(k)$ is white Gaussian noise sequence with zero mean and standard deviation 0.2. The input $u(k)$ is a uniformly random sequence between [-1 1], and the other parameters are non-linear degree $l = 3$, output delay terms $n_y = 2$, input delay terms $n_u = 2$, error terms $n_e = 0$, total time instants N=200 and ESR $\rho = 0.05$.

Let $y = [y(1),...,y(N)]^T$ be a vector of measured outputs at N time instants, and $P_m = [p_m(1),p_m(2),...,p_m(N)]^T$ be a vector formed by the $m^{th}$ candidate model term,where $m = 1,2,...,M$. Let $D = \{P_1,P_2,...,P_M\}$ be a dictionary composed of the M candidate bases.The model term selection through FROLS algorithm is equivalent to finding a full dimensional subset $D_{M_0} = \{\alpha_1,\alpha_2,...,\alpha_{M_0}\} = \{P_1,P_2,...,P_{M_0}\}$ of $M_0$ ($M_0 \leq M$) bases, from the dictionary D. Now the final Y can be approximated as,

$$Y = \theta_1\alpha_1 + \theta_2\alpha_2 + ... + \theta_{M_0}\alpha_{M_0} + e$$

or, in compact matrix form,

$$Y = X\theta + e$$

Here the parameter $\theta$ can be estimated by using least square method as,

$$\theta = (X^TX)^{-1}X^TY$$

### 1.1 Algorithm

The first-step in FROLS starts with the initial full model, in our case it is,

$$Y = c_0 + \sum_{i_1=1}^{n} c_{i_1}x_{i_1}(k) + \sum_{i_1=1}^{n}\sum_{i_2=i_1}^{n} c_{i_1i_2}x_{i_1}(k)x_{i_2}(k) + \sum_{i_1=1}^{n}\sum_{i_2=i_1}^{n}\sum_{i_3=i_2}^{n} c_{i_1i_2i_3}x_{i_1}(k)x_{i_2}(k)x_{i_3}(k)$$

where $n = n_u + n_y = 4$ and the initial full dictionary $D = \{P_1,P_2,...,P_M\}$. For $m = 1,2,\ldots,M(\frac{(n+l)!}{n!\,l!})$ let $q_m = p_m$ and $\sigma = Y^TY$, calculate

$$g_m^{(1)} = \frac{Y^Tq_m}{q_m^Tq_m}$$

$$ERR^{(1)}[m] = (g_m^{(1)})^2(q_m^Tq_m)/\sigma$$

Let

$$ERR[\,l_1\,] = \max\{ERR^{(1)}[m] : 1 \leq m \leq M\}$$

$$l_1 = arg \max_{1\leq m\leq M} \{ERR^{(1)}\}$$

The first significant basis can then be selected as $\alpha_1 = P_{l_1}$, and the first associated orthogonal vector can be chosen as $q_1 = P_{l_1}$ also set $g_1 = g_{l_1}^{(1)}$ and $err[1] = ERR^{(1)}[\,l_1\,]$. Now from the second step onwards till $s^{th}$ step, $m \neq l_1, m \neq l_2,...,m \neq l_{s-1}$. For $m = 1,2,...,M$, calculate

$$q_m^{(s)} = P_m - \sum_{r=1}^{s-1}\frac{P_m^Tq_r}{q_r^Tq_r}q_r,\ P_m \in D - D_{s-1}$$

$$g_m^{(s)} = \frac{Y^Tq_m}{q_m^Tq_m}$$

$$ERR^{(s)}[m] = (g_m^{(s)})^2(q_m^Tq_m)/\sigma$$

Let

$$ERR[l_s] = \max\{ERR^{(s)}[m] : 1 \le m \le M\}$$

$$l_s = arg \max_{1 \le m \le M} \{ERR^{(s)}\}$$

and set $q_s = q_{l_s}^{(s)}$, $g_s = g_{l_s}^{(s)}$ and $err[s] = ERR^{(s)}[l_s]$

The search is terminated at the $M_0$ step when the ESR is less than a pre-specified threshold

$$ESR = 1 - \sum_{s=1}^{M_0} err(s) \le \rho \text{ where } \rho \text{ is } 0.05$$

## 1.2 MATLAB Code

```
%% FROLS ALGORITHM TO FIT AN NARX MODEL
% Non Linear system
%                 y_k  :-0.605*y(k-1)-0.163*y^2(k-1)+0.588*u(k-1)-0.240*u(k-2)+e(k)
% where             e  ¬  Gaussian white noise
% Inputs:          ny  :  number of output delay terms
%                  nu  :  number of input delay terms
%                  ne  :  number of error terms
%                   l  :  max degree
%                   u  :  unifromily distribute input btw [-1 1]
% Outputs           c  :  selected terms (specified vector format)
%                 not  :  number of selected terms
%                 phi  :  parameter values
%
%
%% Preliminaries
clear all;
clc;
ny=2                                                  % nummber of   y terms
nu=2                                                  % nummber of   u terms
ne=0;                                                 % nummber of   e terms
n=ny+nu+ne;                                           % total terms
l=3;                                                  % max degree
ly=200;                                               % length
M= factorial(n+l)/(factorial(n)*factorial(l))         % Total number of possible terms
mu=0;                                                 % noise average
sigma=0.1;                                            % noise standard deviation
e=sigma*randn(200,1)+mu;                              % Gaussian white noise
u=-1+2*rand(200,1);                                   % uniformly distributed input
y=[0.1;0.5];                                          % initial output

for k=3:ly
    y(k)=(-0.605*y(k-1))-(0.163*y(k-2)^2)+(0.588*u(k-1))-(0.240*u(k-2))+e(k);%generating outputs
```

```
end

%% Generating all possible term sets
% C-stores all possible terms sets. --each row indicate a term
% and column values represent the powers corresponfing delay terms as u(k-1),u(k-2),y(k-1),y(k-2)
% eg: [1 0 0 2] indicate term = u(k-1)^1*y(k-2)^2

b=2;

C=zeros(M,n);                                          % dimension (35,4)
for p=1:l

    if p==1                                            % for power=1
      for i_1=1:n

          C(b,i_1)=C(b,i_1)+1;

          b=b+1;

      end

    end

    if p==2                                            % for power=2

        for i_1=1:n

          for i_2=i_1:n

              C(b,i_1)=C(b,i_1)+1;

              C(b,i_2)=C(b,i_2)+1;

            b=b+1;

          end

        end

    end


    if p==3                                            % for power=3
        for i_1=1:n

          for i_2=i_1:n

              for i_3=i_2:n

                C(b,i_1)=C(b,i_1)+1;

                C(b,i_2)=C(b,i_2)+1;

                C(b,i_3)=C(b,i_3)+1;

                  b=b+1;

              end

          end

        end
```

```
        end

end

%% creating        D matrix -- Holds the values of all possible terms for each output
nm=max(nu,ny);                                              %  picking max delay
D=ones(ly-nm,M);

%size(D)

for i=nm+1:ly                                               %  Iteraing y
       for j=1:M                                            %  Iterating       C
            k=1;

            for l=1:nu

                          D(i-nm,j)=(D(i-nm,j))*(u(i-l)^(C(j,k)));    %  combining      delay terms of input u
            k=k+1;

            end

            for m=1:ny

                          D(i-nm,j)=(D(i-nm,j))*(y(i-m)^(C(j,k)));    %  combining      delay terms of output y
            k=k+1;

            end

       end

end

size(D);


%% FROLS to Select the terms
Y=y(nm+1:ly,:);

sig=Y'*Y;                                                   %  sigma

sg=zeros(M,1);                                              %  vector to       hold selected g _m

serr=zeros(M,1);                                            %  vector to       hold selected err
%sl=zeros(M,1);

q=zeros(ly-nm,M);                                   % vector to       hold evaluated orthogonal vectors

for j=1:M
       err=zeros(M,1);

       g=zeros(M,1);

       if j==1                                              %  loop to find first          prominant term

              for  i=1:M

                    q(:,i)=D(:,i);

                    qq=q(:,i)'*q(:,i);

                    g(i)=(Y'*q(:,i))/qq;
```

```
                err(i)=(g(i)^2*qq)/sig;
            end
        else
            for  i=1:M                          % loop to find second and remaining         prominant terms
                if ¬ismember(i,sl)
                    p=D(:,i);
                        pd=zeros(ly-nm,1);

                        for r=1:size(sq,2)      % evaluating the subtracting term in         orthogonalisation
                            qr=sq(:,r);
                        pd=pd+((p'*qr)/(qr'*qr))*qr;
                    end

                    q(:,i)=p-pd;                            %  orthogonal vecctor        q_m
                    qq=q(:,i)'*q(:,i);
                    g(i)=(Y'*q(:,i))/qq;
                    err(i)=(g(i)^2*qq)/sig;                 %  evaluating err_m
                end
            end
        end
[ERR,l]=max(err);                                 % picking     the term with max err and its index
serr(j)=ERR;                                                %  store selected err value
sl(j)=l;                                                    %  store selected index of term
sg(j)=g(l);                                                 %  store selected g_m
sq(:,j)=q(:,l);                                             %  store selected orthogonal vector

ESR=1-sum(serr);                                            %  termination parameter
if ESR≤0.05                                                % check termination condition
        break                                               %  end the search if condition meets
end
end
sl;                                                         %  selected term index
c=C(sl,:)                                                   %  selected terms

%% Calculating the coefficients(parameters) using LS
for i=1:ly-nm
        X(i,:)=D(i,sl);
end
X;
    not=size(c,1)                                           %  number of selected terms
     phi=inv(X'*X)*X'*Y                                     %  parameter values
```

### 1.2.1 Strategy used for storing all possible terms

In the program the all possible terms of the full NARX model is stored as a matrix C. In which the columns represent the degree of a particular delay term i.e. [u(k-1) u(k-2) y(k-1) y(k-2)], and rows represents a whole term. For e.g. Row [1 0 0 0] represents the term $u(k-1)$ and [1 0 2 0] represents the term $u(k-1)y(k-1)^2$. **2.3 Output results**

| Sl No | C Matrix Entry | | | | Term | Parameter value |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | $y(k-1)$ | |
| 2 | 1 | 0 | 0 | 0 | $u(k-1)$ | |
| 3 | 0 | 1 | 0 | 0 | $u(k-2)$ | |
| 4 | 0 | 0 | 0 | 2 | $y(k-2)^2$ | |
| 5 | 1 | 0 | 0 | 2 | $u(k-1)y(k-2)^2$ | |
| 6 | 0 | 0 | 1 | 2 | $y(k-1)y(k-2)^2$ | |
| 7 | 0 | 0 | 0 | 1 | $y(k-2)$ | |
| 8 | 2 | 1 | 0 | 0 | $u(k-1)^2u(k-1)$ | |
| 9 | 0 | 0 | 0 | 3 | $y(k-2)^3$ | |
| 10 | 1 | 1 | 0 | 1 | $u(k-1)u(k-2)y(k-2)$ | |
| 11 | 3 | 0 | 0 | 0 | $u(k-1)^3$ | |
| 12 | 0 | 2 | 0 | 0 | $u(k-2)^2$ | |
| 13 | 1 | 1 | 1 | 0 | $u(k-1)u(k-2)y(k-1)$ | |
| 14 | 1 | 2 | 0 | 0 | $u(k-1)u(k-2)^2$ | |
| 15 | 1 | 0 | 0 | 1 | $u(k-1)y(k-2)$ | |
| 16 | 0 | 1 | 0 | 1 | $u(k-2)y(k-2)$ | |
| 17 | 0 | 0 | 1 | 1 | $y(k-1)y(k-2)$ | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 18 | 0 | 3 | 0 | 0 | -0.6415 | |
| | | | | | 0.5872 | |
| | | | | | -0.1585 | |
| | | | | | -0.0110 | |
| | | | | | 0.1362 | |
| | | | | | 1.6401 | |
| | | | | | -0.0809 | |
| | | | | | -0.0917 | |
| | | | | | 0.3299 | |
| | | | | | -0.2946 | |
| | | | | | 0.0637 | |
| | | | | | 0.0523 | |
| | | | | | -0.1195 | |
| | | | | | 0.1303 | |
| | | | | | -0.0309 | |
| | | | | | -0.1646 | |
| | | | | | 0.2626 | |
| | | | | | -0.0917 | |
| | | | | | 0.2682 | |
| | | | | | -0.0696 | |
| | | | | | 0.6765 | |
| | | | | | 0.2648 | |
| | | | | | 0.2110 | |
| | | | | | -0.2215 | |
| | | | | | 0.0238 | |
| | | | | | -0.0351 | |
| | | | | | -0.0351 | |
| | | | | | -1.1112 | |
| | | | | | -2.6937 | |
| | | | | | 0.8689 | |
| | | | | | 2.2821 | |
| | | | | | -1.2493 | |
| | | | | | 0.1082 | |
| | | | | | -0.0161 | |
| | | | | | 0.0089 | |
| 19 | 0 | 2 | 1 | 0 | $u(k-2)^2 y(k-1)$ | |
| 20 | 1 | 0 | 1 | 1 | $u(k-1)y(k-1)y(k-2)$ | |
| 21 | 0 | 2 | 0 | 1 | $u(k-2)^2 y(k-2)$ | |
| 22 | 2 | 0 | 1 | 0 | $u(k-1)^2 y(k-1)$ | |
| 23 | 2 | 0 | 0 | 1 | $u(k-1)^2 y(k-2)$ | |
| 24 | 1 | 0 | 2 | 0 | $u(k-1)y(k-1)^2$ | |
| 25 | 0 | 0 | 2 | 0 | $y(k-1)^2$ | |
| 26 | 0 | 1 | 1 | 0 | $u(k-2)y(k-1)$ | |
| 27 | 0 | 0 | 0 | 0 | *constant* | |
| 28 | 0 | 1 | 0 | 2 | $u(k-2)y(k-2)^2$ | |
| 29 | 0 | 1 | 1 | 1 | $u(k-2)y(k-2)y(k-2)$ | |
| 30 | 0 | 0 | 3 | 0 | $y(k-1)^3$ | |
| 31 | 0 | 0 | 2 | 1 | $y(k-1)^2 y(k-2)$ | |
| 32 | 0 | 1 | 2 | 0 | $u(k-2)y(k-1)^2$ | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 33 | 2 | 0 | 0 | 0 | $u(k-1)^2$ | |
| 34 | 1 | 1 | 0 | 0 | $u(k-1)u(k-2)$ | |
| 35 | 1 | 0 | 1 | 0 | $u(k-1)y(k-1)$ | |

Table 1: $\sigma$ = 0.2 and ESR = 0.05

| Sl No | C Matrix Entry | | | | Term | Parameter value | Error |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | $y(k-1)$ | -0.6099 | 0.0049 |
| 2 | 1 | 0 | 0 | 0 | $u(k-1)$ | 0.5848 | 0.0032 |
| 3 | 0 | 1 | 0 | 0 | $u(k-2)$ | -0.2392 | -0.0008 |
| 4 | 0 | 0 | 0 | 2 | $y(k-2)^2$ | -0.1663 | 0.0033 |

Table 2: $\sigma$ = 0.1 and ESR = 0.05

# References


[1] Stephen A Billings. *NonLinear System Identification*. Wiley, 2013.

[2] Simon Haykin. *Kalman Filtering And Neural Networks*. Awiley-Interscience Publication, 2001.

[3] mathworks.com. *Kalman Filter*. https://in.mathworks.com/, 2019.