Takagi-Sugeno Model Identification Toolbox

May 20, 2021

Example of a throttle valve as a nonlinear ARX model TS model (NARX)

V1.0

Axel Dürrbaum (`axel.duerrbaum@mrt.uni-kassel.de`)

Department of Measurement and Control (MRT)

Institute for System Analytics and Control (ISAC)

University of Kassel, Germany (`http://www.uni-kassel.de/go/mrt`)

$Id: NARX_Throttle.m | Fri Feb 26 16:25:05 2021 +0100 | Axel Dürrbaum $

Contents

Estimation of a NARX TS model from the measurement data of a throttle valve

$$y_{k+1} = \sum_{i=1}^{n_v} \phi_i(z) \cdot \left(\sum A_i \cdot x_y + B_i \cdot x_u + c_i \right)$$

with

- scheduling vector $z = [y_{k-1}, y_{k-2}, y_{k-3}, u_k]$
- FCM clustering in product-space with $s = 10$ tries
- regression vector $x = [x_y, x_u]$, $x_y = [y_{k-1}, y_{k-2}, y_{k-3}]$ and $x_u = [u_k]$
- initalization of the local models with global least squares
- optimization of the parameters: both cluster centers v and local models A_i, B_i, c_i

Path to TSModel class

```
addpath( '../TSModel' )
```

1 Structural parameters

```
nv = 3;    % number of clusters = local models
nu = 1;    % number of inputs
ny = 1;    % number of outputs
nue = 1.1; % fuzziness parameter
MSF = 'FCM';
```

2 Identification data

Load input vector u and output vector y from file:

```
load( 'Data/Throttle1.mat' );
```

Compute the time vector (for plotting)

```
dt = 1e-2; % Sampling time
t = dt * transpose( 0:size(u,1)-1 );
```

3 Creation of the TS model

```
ts = TSModel( 'ARX', nv, nu, 'Name','NARX', 'Comment','Throtte valve');
```

Set the identification data

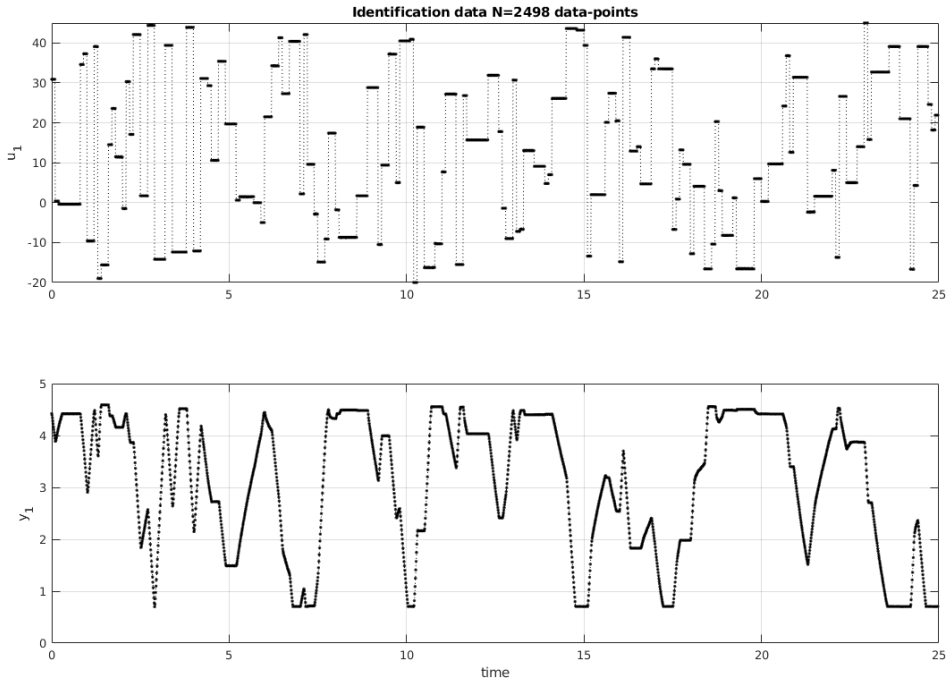
```
ts.setData( u, y, 'SampleTime',dt, 'Labels', { 'volt', 'angle' } );
```

Limit $u = [-20; 45]$ and $y = [0; 5]$

```
ts.setDataLimits( [-20,45 ; 0,5] );
```

Plot the identification data

```
plotIdentData( ts, 1 );
set(gcf, 'WindowState', 'maximized' );
```



Set the scheduling lags: u(k), y(k-1), y(k-2), y(k-3)

```
ts.setSchedulingLags( {0}, [1,2,3] );
```

Set the regressor lags: u(k), y(k-1), y(k-2), y(k-3)

```
ts.setRegressorLags( {0}, [1,2,3] );
```

4 Clustering

Clustering is done in product-space $[u, y]$ with FCM and $\nu = 1.1$ for a multi-start of $s = 10$ tries and random number generator fixed initialized with seed 0

```
ts.clustering( 'FCM', 'nue', nue, 'tries',10, 'seed', 0 );
```

Cluster centers v_1: (columns $z = [y_{k-1}, y_{k-2}, y_{k-3}, u_k]$, rows=local models)

```
v1 = getCluster( ts )
```

```
v1 =
    3.3022    3.2939    3.2859    7.8404
    2.5556    2.5981    2.6381   34.1314
    3.5791    3.5326    3.4894  -12.1079
```

5 Initialization of the local models

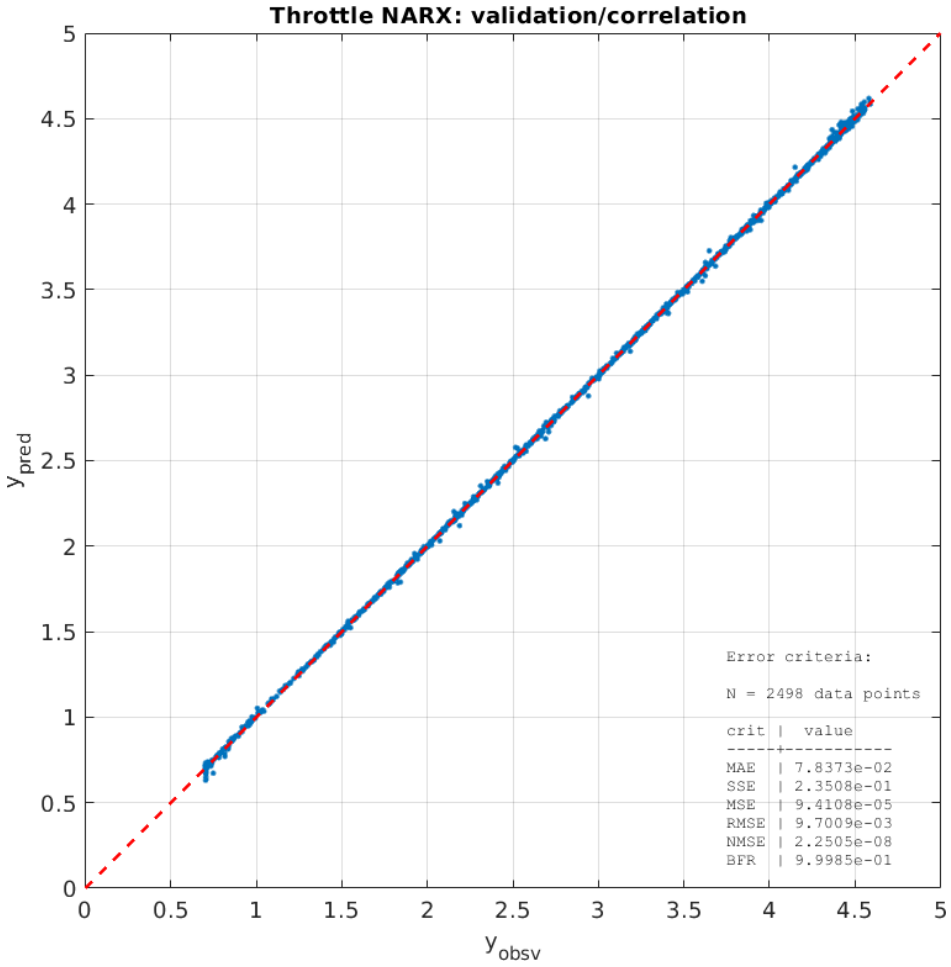
The local models are intialized with global Least-Squares and FCM membership functions ($\nu = 1.1$)

```
ts.initialize( MSF, 'nue', nue, 'method','global' );
```

6 Prediction of the initial TS model

Prediction of the NARX TS model with the identificaton input: u

```
y_pred = ts.predict( u,y );
plotResiduals( y, y_pred, 'figure', 2, 'title', ...
    'Throttle NARX: validation/correlation ' );
set(gcf, 'WindowState', 'maximized' );
```



7 Optimization of the TS model

by optimizing clusters centers v (MF) and local model parameters $A/B/c$

```
ts.optimize( 'B' );
```

Iteration	Func-count	f(x)	Norm of step	First-order optimality
0	28	0.283319		21.7
1	56	0.283319	2.00207	21.7
2	84	0.262651	0.500517	8.12
3	112	0.242779	1.00103	5.71
4	140	0.242779	0.852641	5.71
5	168	0.238057	0.21316	6.48
6	196	0.232624	0.42632	1.9
7	224	0.232188	0.182499	1.71

```
Local minimum possible.

lsqnonlin stopped because the final change in the sum of squares relative to
its initial value is less than the value of the function tolerance.
```

Cluster centers v_1: (columns: $z = [y_{k-1}, y_{k-2}, y_{k-3}, u_k]$, rows=local models)

```
v2 = getCluster( ts )
```

```
v2 =
    4.9835    3.1094    3.0130    2.2075
    0.8923    2.5174    2.8986    2.7632
    5.1885    4.0175    4.9828    2.1235
```

Local model matrices $A_i \rightarrow x_y, B_i \rightarrow x_u, c_i$

```
[A,B,c] = getLM( ts )
```

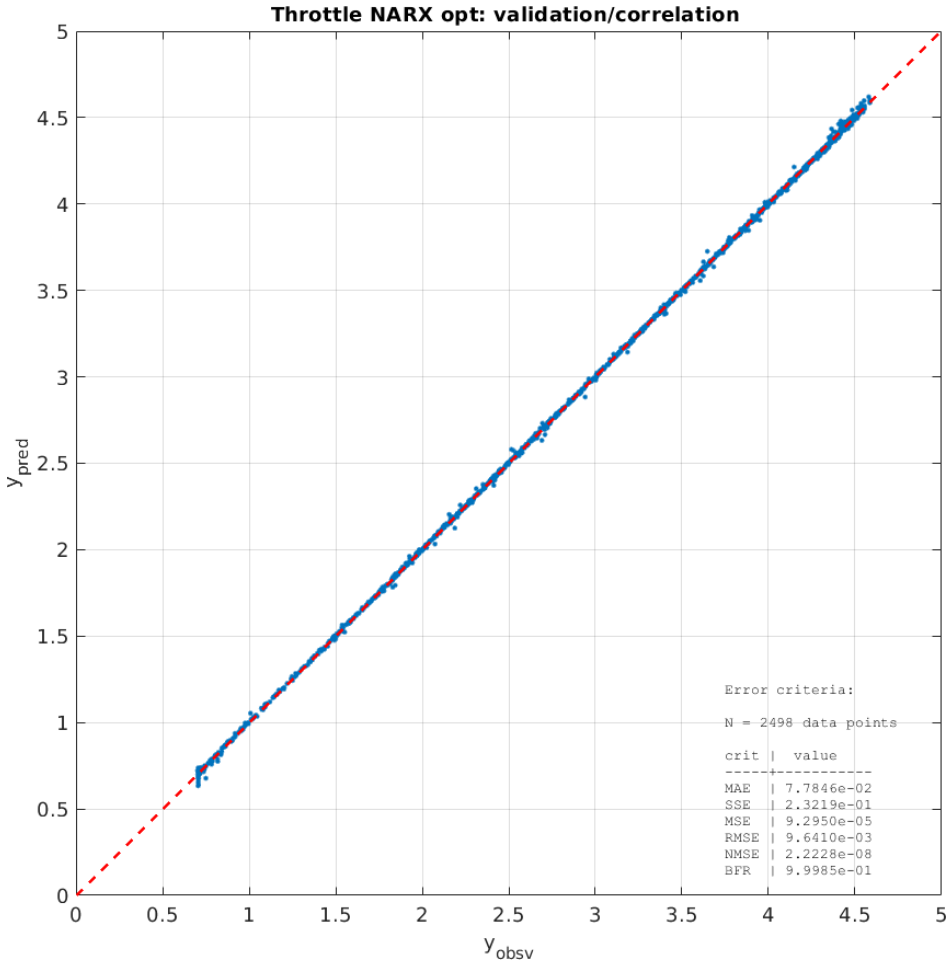
```
A =
    2.2553   -0.0007   -1.3978
   -1.6511    0.0149    0.3157
    0.3925    2.0820   -0.0011
B =
    0.0120
    2.2056
   -1.5839
c =
    0.3648
    0.0002
    0.0592
```

8 Prediction of the optimized TS model

```
y_pred_opt = ts.predict( u,y );
```

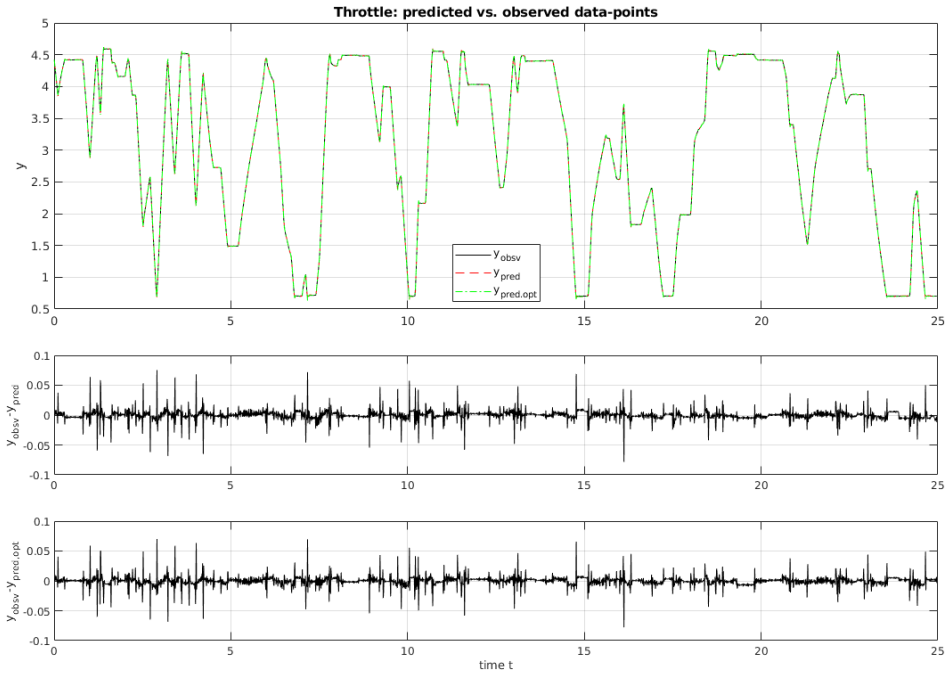
Plot the correlation

```
plotResiduals( y, y_pred_opt, 'figure', 3, 'title', ...
    'Throttle NARX opt: validation/correlation ' );
set(gcf, 'WindowState', 'maximized' );
```



Plot identification and predicted output y

```
figure(4),clf
subplot(4,1,1:2)
plot( t,y,'k-',t,y_pred,'r--',t,y_pred_opt,'g-.')
grid on
ylabel('y')
legend('y_{obsv}','y_{pred}','y_{pred.opt}','Location','best')
title( 'Throttle: predicted vs. observed data-points' )
subplot(4,1,3)
plot( t,y-y_pred,'k-')
grid on
ylabel('y_{obsv}-y_{pred}')
subplot(4,1,4)
plot( t,y-y_pred_opt,'k-')
grid on
ylabel('y_{obsv}-y_{pred,opt}')
xlabel( 'time t' )
set(gcf, 'WindowState', 'maximized' );
```




```
figure(5),clf
hold on
plot(u,y,'k.')
plot(v1(:,1),v1(:,2),'rs','MarkerSize',12)
plot(v2(:,1),v2(:,2),'bd','MarkerSize',12)
grid on
box on
xlabel( 'u' )
ylabel( 'y' )
title( 'Throttle: cluster centers v')
legend( 'data-points','v FCM', 'v opt' )
set(gcf, 'WindowState', 'maximized' );
```