Takagi-Sugeno Model Identification Toolbox

March 18, 2021

Example of a NARX TS model for a Narendra function

V 1.0

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

Contents

Example of the identification of a NARX MISO TS model for given multiple inputs u and single output y.

Determine the NARX TS model

$$\hat{y}_{k+1} = \sum_{i=1}^{n_v} \phi_i(z) \cdot \left(\sum_{l=0}^{l_y} A_i \cdot y_{k-l} + \sum_{j=0}^{n_u} \sum_{l=0}^{l_u} B_{i,j} \cdot u_{k-l} + c_i \right)$$

- for given $u_j, j = 1, \ldots, n_u$ of n_u input vectors and
- input lags x_u with length l_u
- vector y of single output,
- output lags x_y with length l_y

- with FCM membership function

$$\mu_i(z) = \left(\sum_{j=1}^{n_v} \left(\frac{||z - v_i||}{||z - v_j||} \right) \frac{2}{\nu - 1} \right)^{-1}$$

- or Gauss membership function

$$\mu_i(z) = e^{-\frac{||z - v_i||^2}{2 \cdot \sigma_i^2}}$$

- norm

$$||z - v_j|| = (z - v_j)^T \cdot w_j \cdot (z - v_j)$$

- and fuzzy basis functions

$$\phi_i(z) = \frac{\mu_i(z)}{\sum_{j=1}^{n_v} \mu_j(z)}$$

- with the scheduling variable $z = u$ (for input space clustering) or $z = [u, y]$ (for product space clustering), and
- cluster centers $v_i, i = 1, \ldots, n_v$.

1 Algorithm

1. Choose number of local models: $n_v = 3$ and fuzzy parameter $\nu = 1.2$ or σ_i =???
2. Select the TS model with minimal MSE of s multi-start tries with clustering and global LS-estimation.
3. Optimize the TS model parameters (v_i, B_i, c_i) for each try or the best found model.

2 Structural parmeters

```
nv = 3;    % number of local models
nu = 1;    % number of inputs
ny = 1;    % number of outputs
nue = 1.2; % Fuzziness parameter
```

Scheduling lags z_{lag} = regressor lags $x_{lag} = [y_{k-1}, y_{k-2}, u_k]^\top$

```
z_lag_u = {0};
z_lag_y = [1,2];
x_lag_u = {0};
x_lag_y = [1,2];
```

3 Identification data

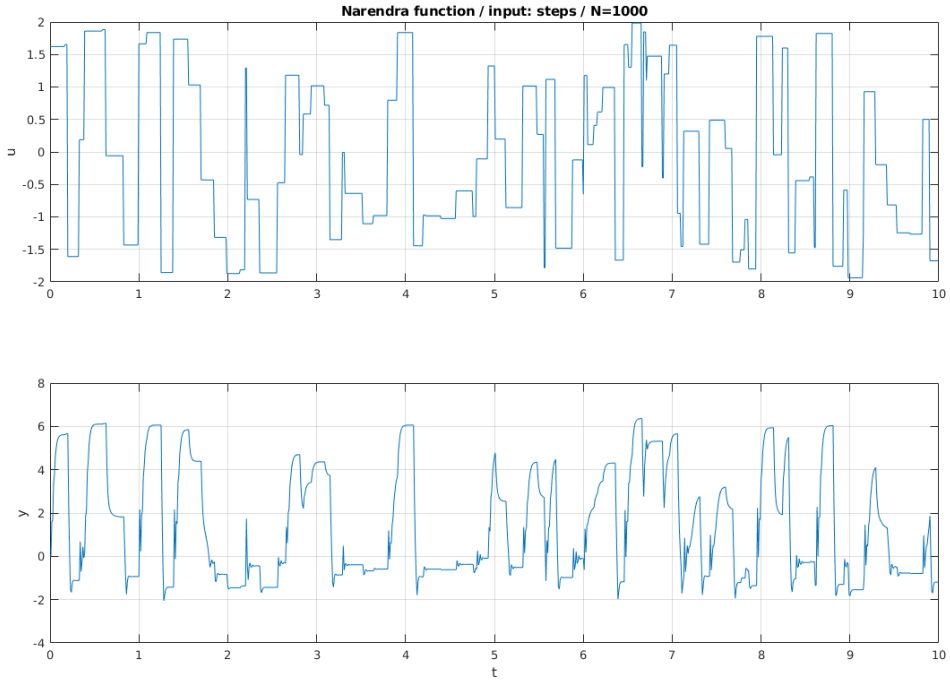
Create input u as steps with amplitude 2 and random width $l = [1, \ldots, 20]$ for $N = 1000$ time steps (sampling rate is 0.01s) and compute the output y from the Narendra function

```
N = 1000;
dt = 1e-2;                    % Sampling time
t = dt * transpose( 0:N-1 );  % time vector: $t$
rng(0);                       % Initalize random number generator
[ u, y ] = Narendra_fct( N );
```

Plot of the identification data

```
figure(1);clf

subplot(2,1,1)
plot(t,u)
```

```
grid on
ylabel('u')
title(sprintf('Narendra function / input: steps / N=%d',N))

subplot(2,1,2)
plot(t,y)
grid on
ylabel('y')
xlabel('t')

set(gcf,'WindowState', 'maximized' );
```



4 Create the NARX TS model

```
addpath( '../TSModel' );  % Path to TSModel class
ts = TSModel( 'ARX', nv, nu, 'Name','NARX Narendra', ...
    'Comment','Narendra function');
```

Set the identification data u, y

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

Set the data limits: u=[-2,2], y=[-5,10]

```
ts.setDataLimits( [-2,2 ; -5,10] );
```

Set the scheduling and regressor lags

```
ts.setSchedulingLags( z_lag_u, z_lag_y );
ts.setRegressorLags( x_lag_u, x_lag_y );
```

5 Clustering

Clustering in product-space $z = [u, y]$ with FCM membership functions

```
ts.clustering( 'FCM', 'nue',nue, 'tries',1, 'seed',0 );
```

Estimated cluster centers: colums y_{k-1}, y_{k-1}, u_k

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

```
v1 =
    5.0968    5.0864    1.1329
    2.1117    2.0720    0.3699
   -0.7434   -0.7214   -0.9016
```

6 Initialize the local models

Estimate the local model parameters with global Least Squares

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

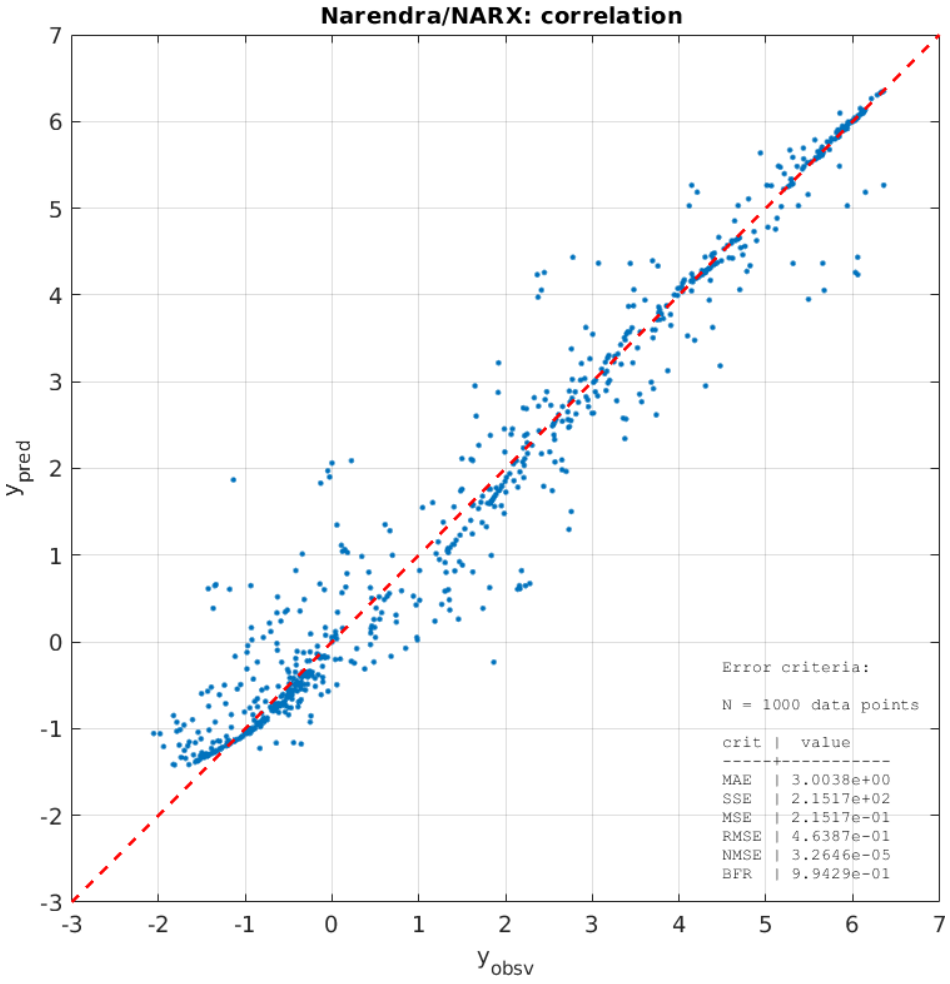
7 Predict the TS model output

for given data u: y_{pred}

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

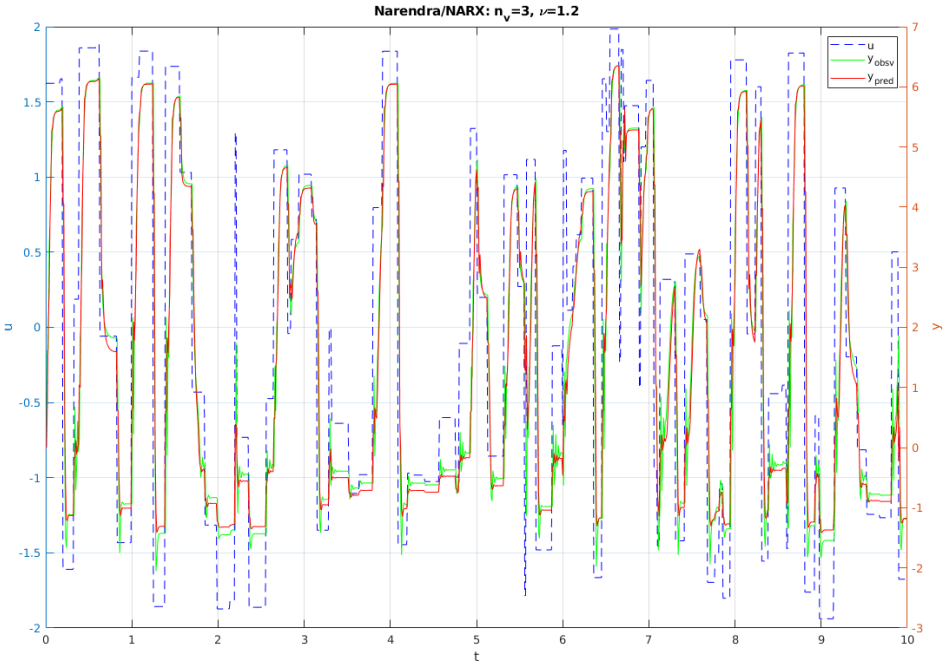
Plot the correlation

```
plotResiduals( y, y_pred, 'figure', 2, ...
    'title', 'Narendra/NARX: correlation ' );
set(gcf,'WindowState', 'maximized' );
```



Plot of identification data y and predicted data y_{pred}

```
figure(3);clf

yyaxis left
plot(t,u,'b--')
ylabel('u')

yyaxis right
plot(t,y,'g-',t,y_pred,'r-')

grid on
ylabel('y')
xlabel('t')
title( sprintf('Narendra/NARX: n_v=%d, \\nu=%g',ts.nv,ts.nue))
legend('u','y_{obsv}','y_{pred}')
set(gcf,'WindowState', 'maximized' );
```



8 Validate with new data

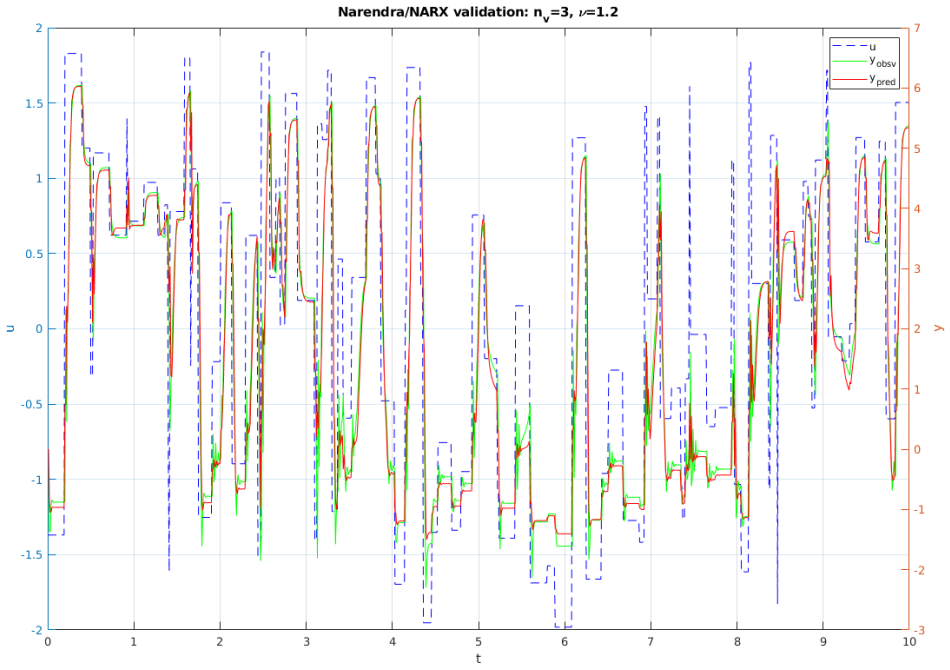
Test with new step inputs u_{val}, y_{val}

```
[u_val,y_val] = Narendra_fct( N );
y_pred_val = ts.predict( u_val,y_val );
```

Plot of observed and predicted outputs

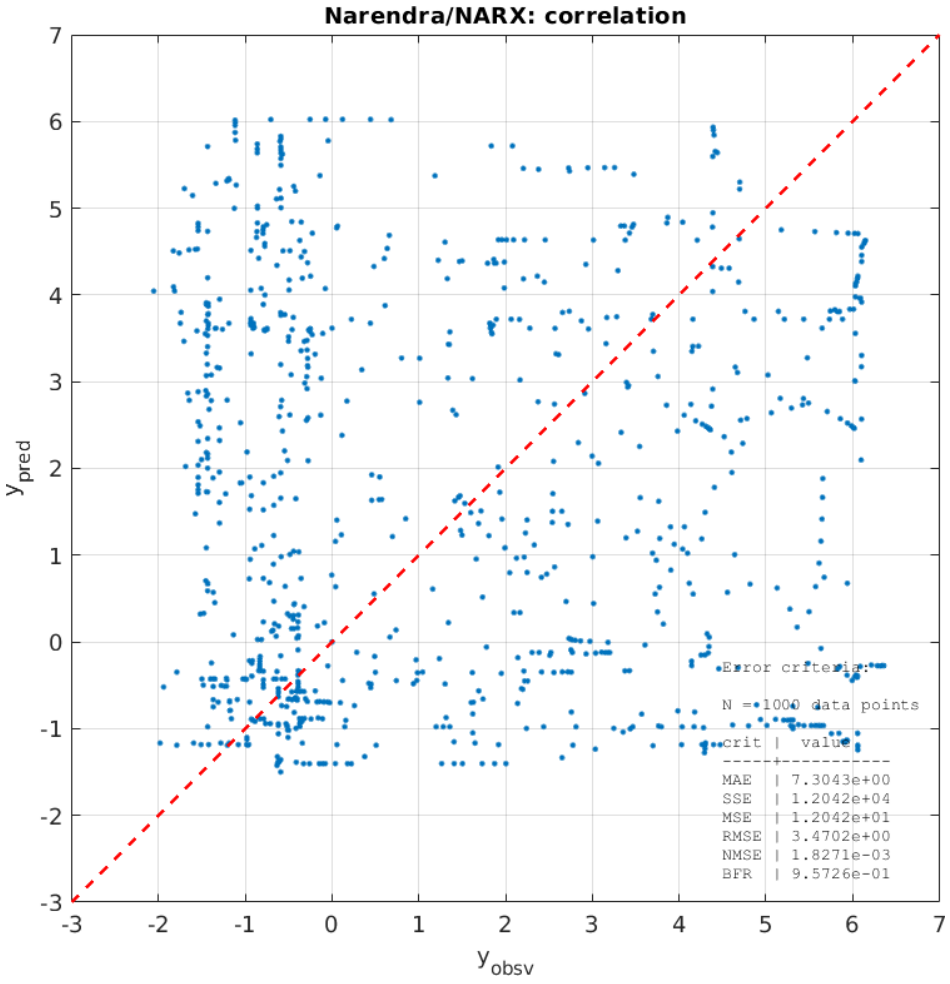
```
figure(5),clf

yyaxis left
plot(t,u_val,'b--')
ylabel('u')

yyaxis right
plot(t,y_val,'g-',t,y_pred_val,'r-')

grid on
ylabel('y')
xlabel('t')
title( sprintf('Narendra/NARX validation: n_v=%d, \\nu=%g',ts.nv,ts.nue))
```

```
legend('u','y_{obsv}','y_{pred}')
set(gcf,'WindowState', 'maximized' );
```



Plot the correlation

```
plotResiduals( y, y_pred_val, 'figure', 4, 'title', 'Narendra/NARX: correlation' );
set(gcf,'WindowState', 'maximized' );
```



9 Optimize the TS model parameters

Optimize both, clusters centers v (MF) and local model parameters A_i, B_i, c_i

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

```
                                         Norm of      First-order
 Iteration  Func-count     f(x)            step          optimality
     0         22         204.749                          87.1
     1         44         191.512          1.83408          172
     2         66         163.567         0.603534         19.2
     3         88         163.567          1.01328         19.2
     4        110          161.41         0.253319         14.3
     5        132         159.489         0.506638         28.2
     6        154         156.458          0.91203         29.5
     7        176         155.028         0.452131         61.6
     8        198         155.028         0.775852         61.6
     9        220         152.899         0.193963         37.7
    10        242         151.819         0.387926         9.55
    11        264         150.229         0.775852         67.5
    12        286         150.229         0.619305         67.5
    13        308         147.767         0.154826         45.9
    14        330         146.075         0.309653          109
    15        352          145.29         0.155379         6.71
    16        374         145.251         0.409002         51.1
    17        396         144.944          0.10225         3.53
    18        418         144.789         0.173526         22.1
    19        440         144.779          0.18164         63.2
    20        462          144.64       0.0454101         3.41

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.

ans =
TS-Model: Type=ARX
Name: 'NARX Narendra'
 Type: 'TSModel'
 Date: '18-Mar-2021 13:49:25'
 Comments:
  'Narendra function'
 Structural parameters: nu = 1, ny = 1, nv = 3
 Identification data: N=1000
, ts=0.01 Initial model estimation:
 lags: u_1:0, y = [1 2]
  Membership function type = FCM
  Clustering: FCM, nue=1.2 norm=Euclidian in input space
 Estimation of local models:
 lags:    u_1:0,   y = [1 2]
  Initialization of local models: global
  Optimization of model parameters: MF&LM
```

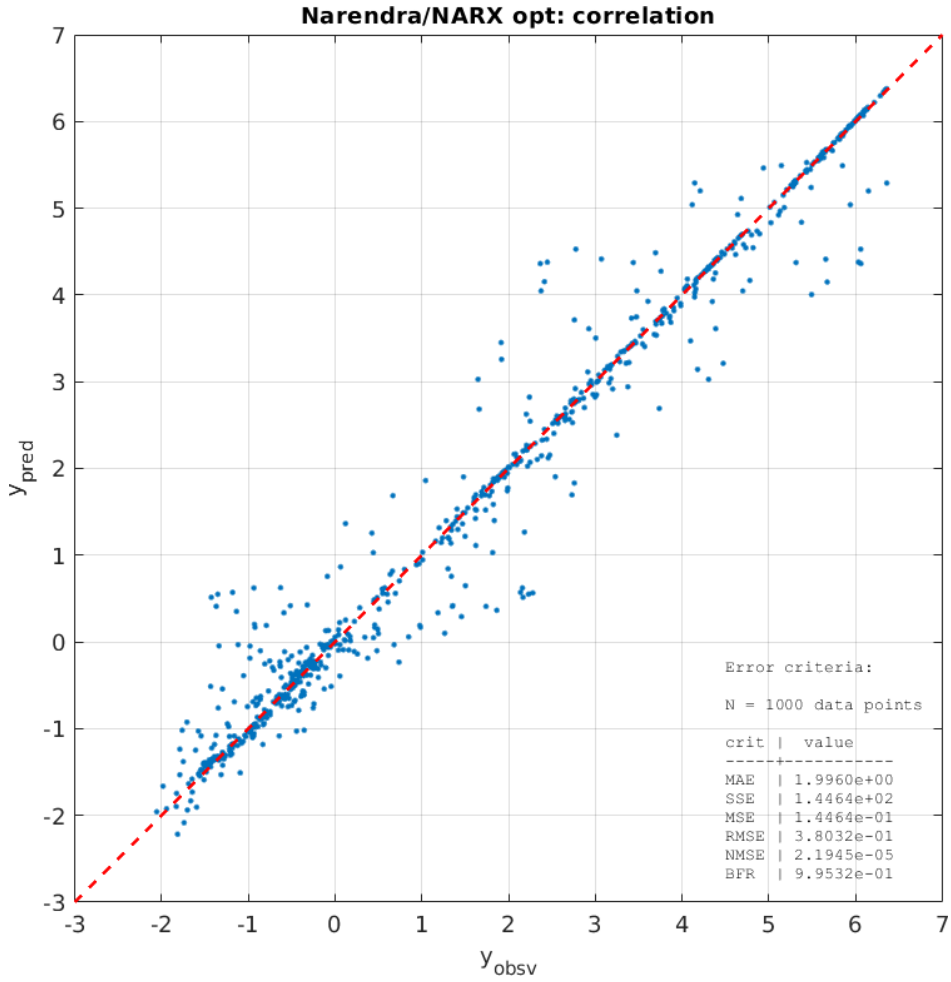
Cluster centers of the optimized TS model

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

```
v2 =
   -1.1413    3.3203   -0.2958
    0.0847   -0.0034    1.7030
    0.2234   -0.7193    1.5269
```

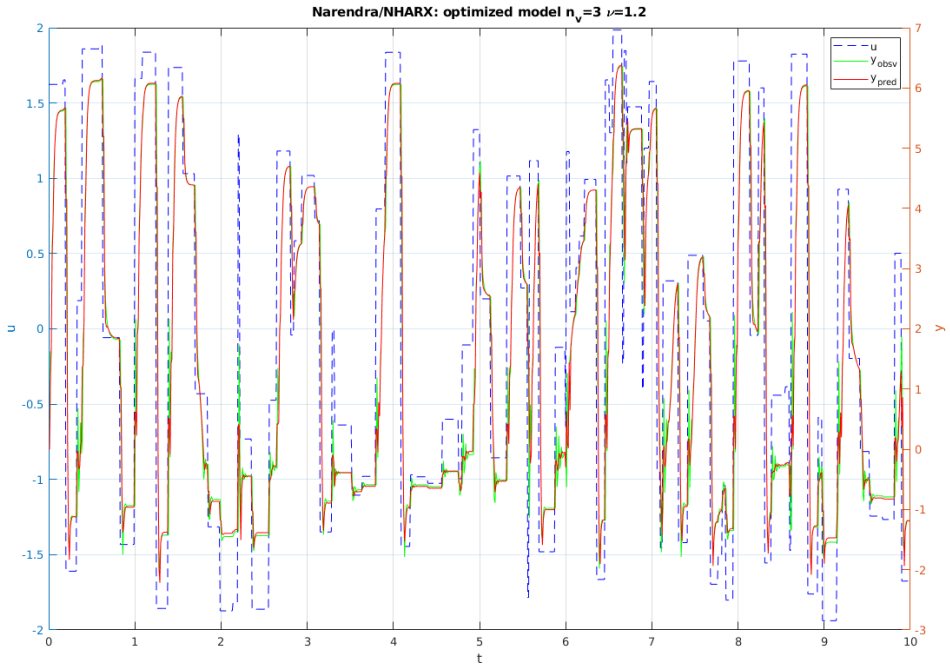
Plot the correlation

```
y_pred_opt = ts.predict( u,y );
plotResiduals( y, y_pred_opt, 'figure', 5, ...
    'title', 'Narendra/NARX opt: correlation' );
set(gcf,'WindowState', 'maximized' );
```



Plot of observed and predicted outputs for optimized TS model

```
figure(6);clf

yyaxis left
plot(t,u,'b--')
grid on
ylabel('u')
title( sprintf('Narendra/NHARX: optimized model n_v=%d \\nu=%g',ts.nv,ts.nue))

yyaxis right
plot(t,y,'g-',t,y_pred_opt,'r-')
grid on
ylabel('y')
xlabel('t')
legend('u','y_{obsv}','y_{pred}')
set(gcf,'WindowState', 'maximized' );
```


Narendra/NHARX: optimized model n_v=3 ν=1.2
u
y_obsv
y_pred
t
u
y