Takagi-Sugeno Model Identification Toolbox

March 10, 2021

Static LiP model for an academic example with extendend results.

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: Static_Acad_extended.m | Fri Feb 26 15:58:17 2021 +0100 | Axel Dürrbaum $

Contents

Example of automatic identification of a static MISO LiP TS model for given multiple inputs u and single output y and selected structural parametes from the auto-example "Static_Acad_auto.m".

1 Algorithm

1. Select the TS model with minimal MSE of m multi-start tries with clustering and LS-estimation.
2. Optimize the TS model parameters (v_i, B_i, c_i) for each try or the best found model.

2 Minimal required data

Inputs $u \in \mathbb{R}^{N \times n_u}$ and output $y \in \mathbb{R}^N$, each with N data points

3 Additional choices

Choosen are the best parameters from the "Static auto" example:

1. Number of local models $n_v = 3$
2. Fuzziness parameter $\nu = 1.2$

4 Identification data

Load data u, y generated with this model from file:

```
load( 'Data/AcadEx.mat' )
```

5 Structural parameters

Number of inputs n_u = number of columns in u

```
Par.nu = size( u, 2);
```

Number of clusters n_v = number of local models ($n_v > 1$)

```
Par.nv = 3;
```

Fuzziness parameter (FCM: $\nu = [1.05, \ldots, 2]$, Gauss: σ_i^2, 0=select range)

```
Par.fuzzy = 1.2;
```

6 Optional settings

For more control over the approximation process.

Multi-Start: number of tries s (clustering & LS), default = 10

```
Par.Tries = 10;
```

Clustering: Fuzzy C-Means (FCM) / Gustafson-Kessel (GK) / KMeans (KMeans), default = 'FCM'

```
Par.Clustering = 'FCM';
```

Clustering in product space: u and y (true) or only input space u (false)

```
Par.ProductSpace = true;
```

Norm for clustering: 'Euclidian' or 'Mahalanobis', default = 'Euclidian'

```
Par.Norm = 'Euclidan';
```

Membership functions: 'FCM' or 'Gauss' type clustering

```
Par.MSF = 'FCM';
```

Least Squares estimation of local models: 'local' or 'global', default = 'global'

```
Par.LS = 'global';
```

Optimize TS model parameters: default='both'

- no optimization: 'none',
- only v: 'cluster',
- only local models (B_i, c_i): 'model', or
- both v and B_i, c_i: 'both'

```
Par.ParOpt = 'both';
```

Optimize each try or only best try: default='each'

- each try: 'each',
- best try: 'best' (less computation time)

```
Par.IterOpt = 'each';
```

Plot clusters and residuals: 'none'/'iter'/'final', default='final'

```
Par.Plots = 'final';
```

Debug infos of algorithm progess: (0=none, 1=info, 2=detailed)

```
Par.Debug = 1;
```

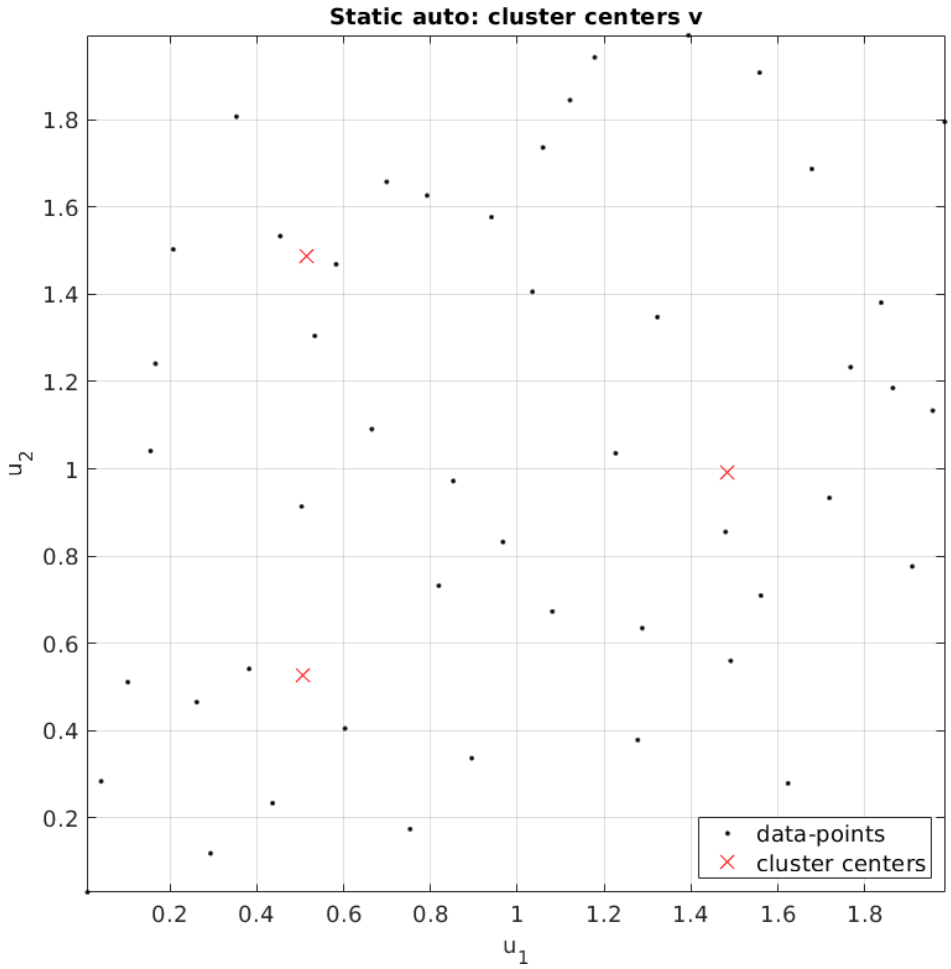
7 Estimation of Static TS model parameters

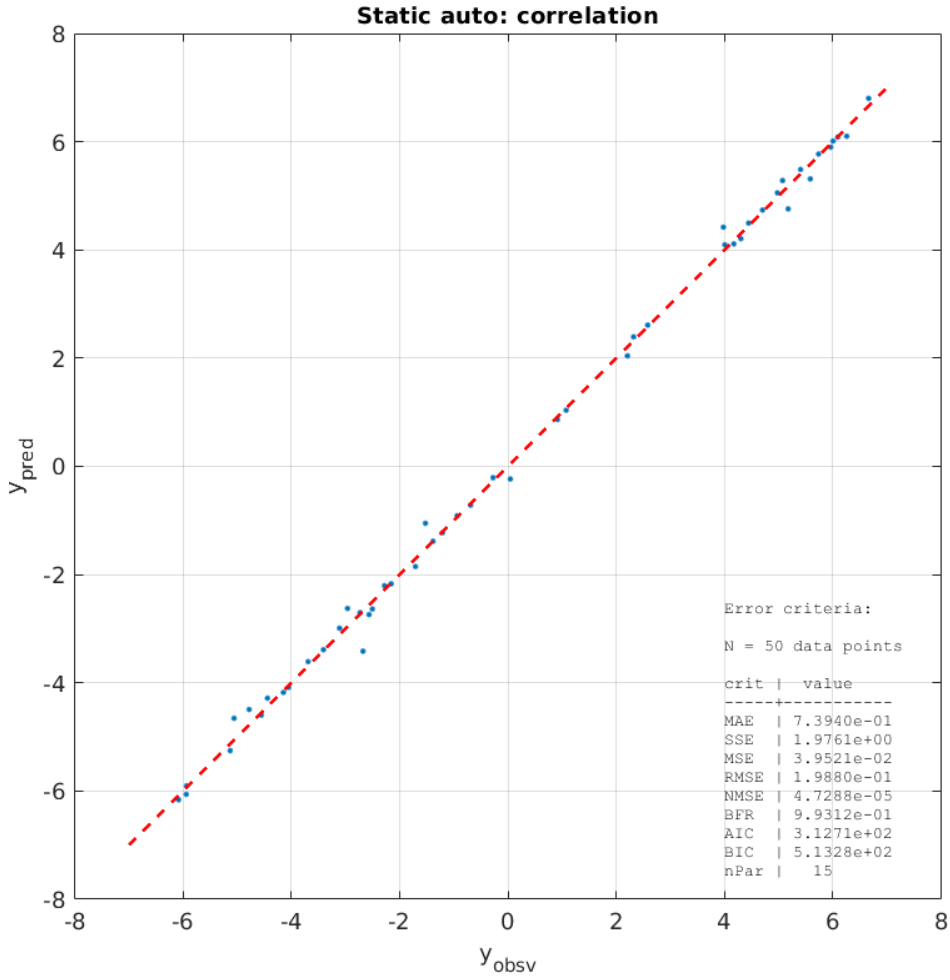
Estimate the TS model with plot of clustering and correlation:

```
model = TSM_Static_auto( u, y, Par );
```

```
Iteration: nv= 3 / fuzzy=1.20
 time = 0.037369 s

Best model: nv= 3 / fuzzy=1.20 /  mse = 3.9521e-02
```





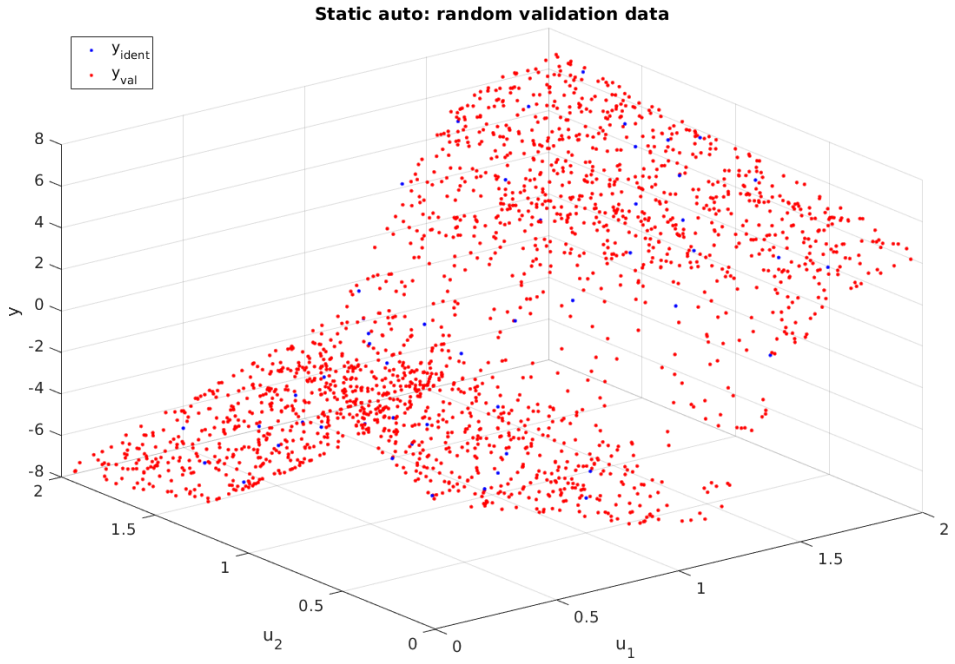
8 Validation of the TS model

As validation data, use random inputs $u_{val} \in [0,2] \times [0,2]$ with $N_{val} = 2000$ data-points

```
N_val = 2000;
u_val = 2 * rand( N_val, Par.nu );
y_val = model.predict( u_val );
```

Plot of model outputs for random test input data:

```
he = figure( 10 );
plot3( u(:,1), u(:,2),y,'b.', ...
       u_val(:,1), u_val(:,2), y_val,'r.','MarkerSize',8);
legend( 'y_{ident}', 'y_{val}','Location','NW' )
grid on
xlabel('u_1'),ylabel('u_2'),zlabel('y')
title( 'Static auto: random validation data' )
set(gca,'FontSize', 14)
set(gcf,'WindowState','maximized');
```

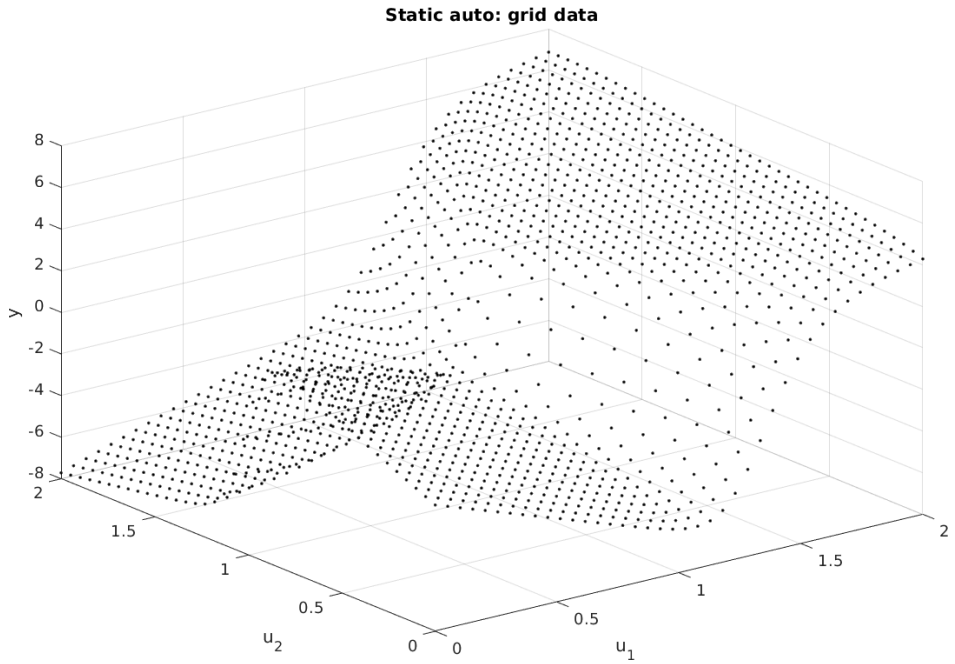


For grid type distributed test data, use $N_{grid} = 40$ data-points per input u_i

```
N_grid = 40;
[U1_grid,U2_grid] = meshgrid( linspace(0,2,N_grid), linspace(0,2,N_grid) );
u_grid = [ U1_grid(1:end)', U2_grid(1:end)' ];
y_grid = model.predict( u_grid );
Y_grid = reshape( y_grid,N_grid,N_grid );
```

Plot of model output for grid input data:

```
hg = figure( 11 );
plot3( U1_grid, U2_grid, Y_grid, 'k.')
grid on
xlabel('u_1'),ylabel('u_2'),zlabel('y')
title( 'Static auto: grid data' )
set(gca,'FontSize', 14)
hg.WindowState = 'maximized';
```



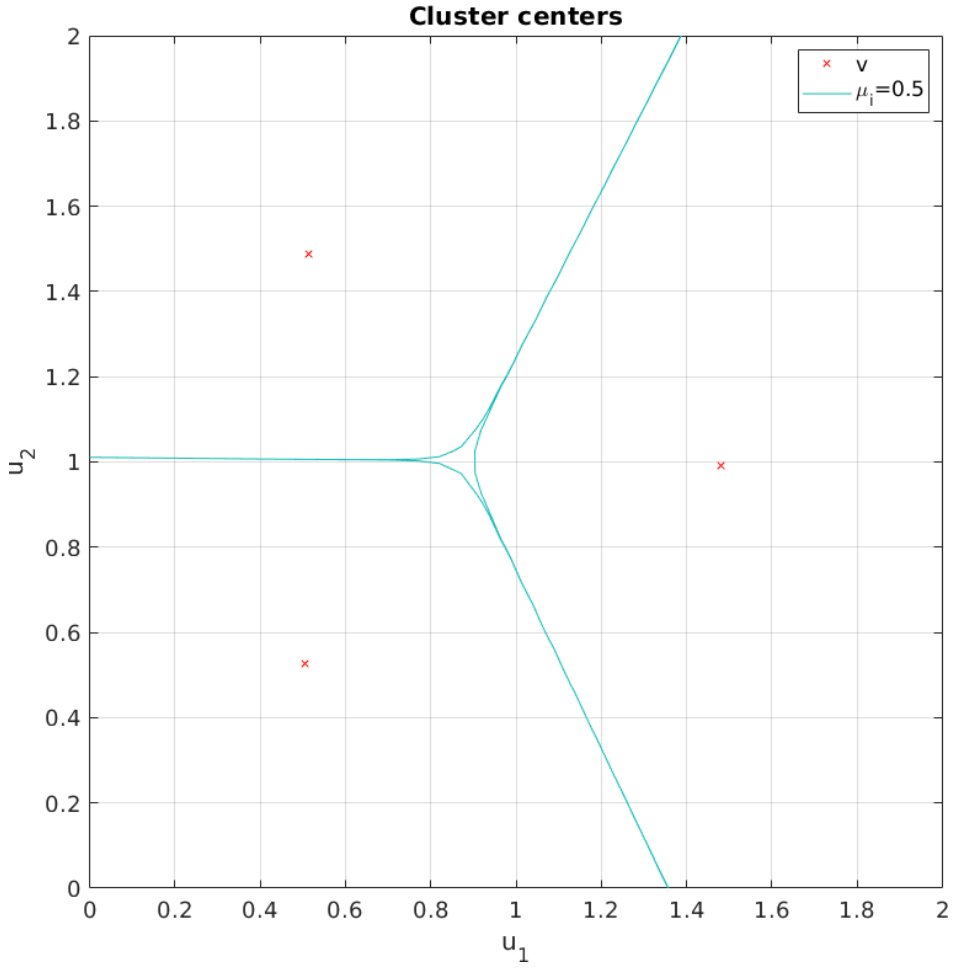
9 Plot final TS model

Compute the membership degrees for the grid data

```
mu_grid = getMSF( model, u_grid, y_grid );
```

Plot the membership degrees μ in 2D

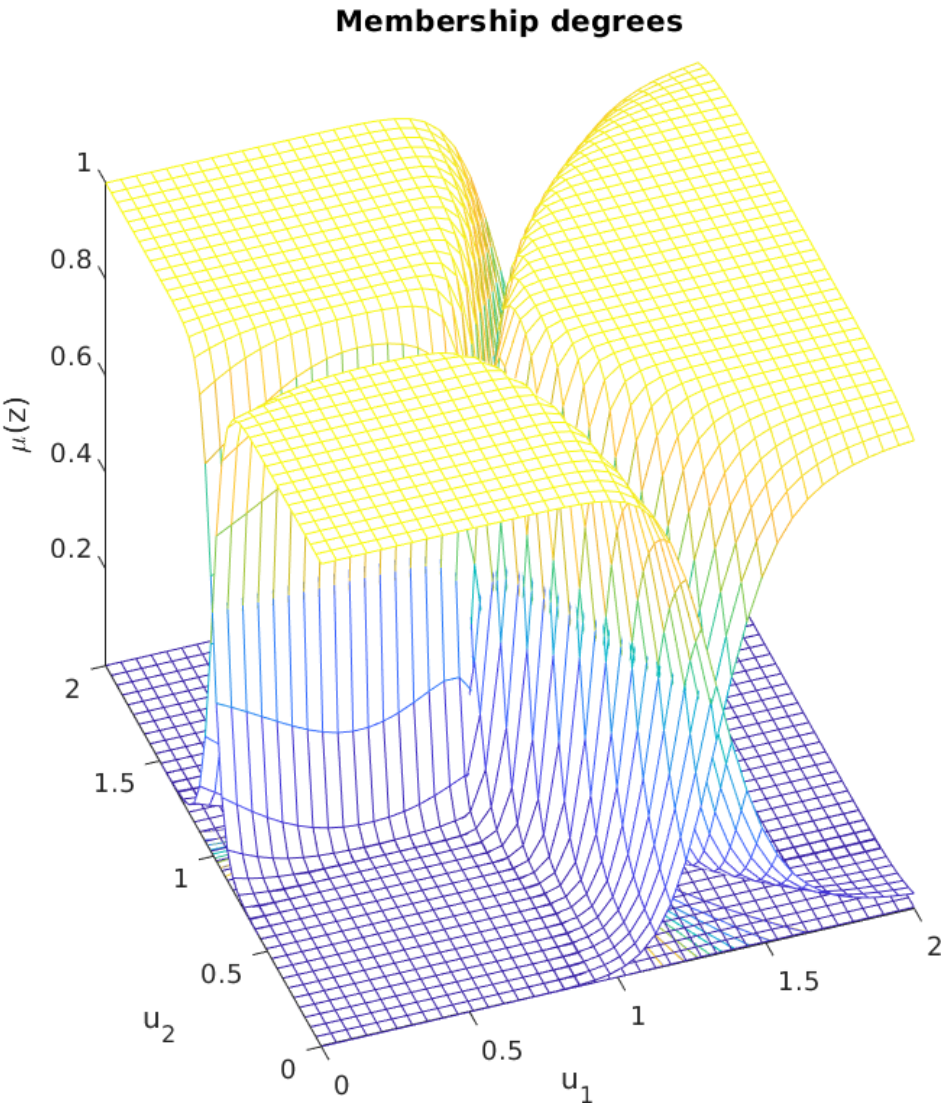
```
figure(12),clf
hold on
plot( v(:,1), v(:,2),'rx' )
for i=1:model.nv
    contour( U1_grid,U2_grid, reshape(mu_grid(:,i),N_grid,N_grid), 1 )
end
hold off
axis square
grid on, box on
view(0,90)
xlabel('u_1'),ylabel('u_2'),zlabel('\mu(z)')
title( 'Cluster centers' )
legend( 'v', '\mu_i=0.5' )
set(gca,'FontSize', 14)
set(gcf,'WindowState','maximized');
```



Plot the membership degrees μ in 3D

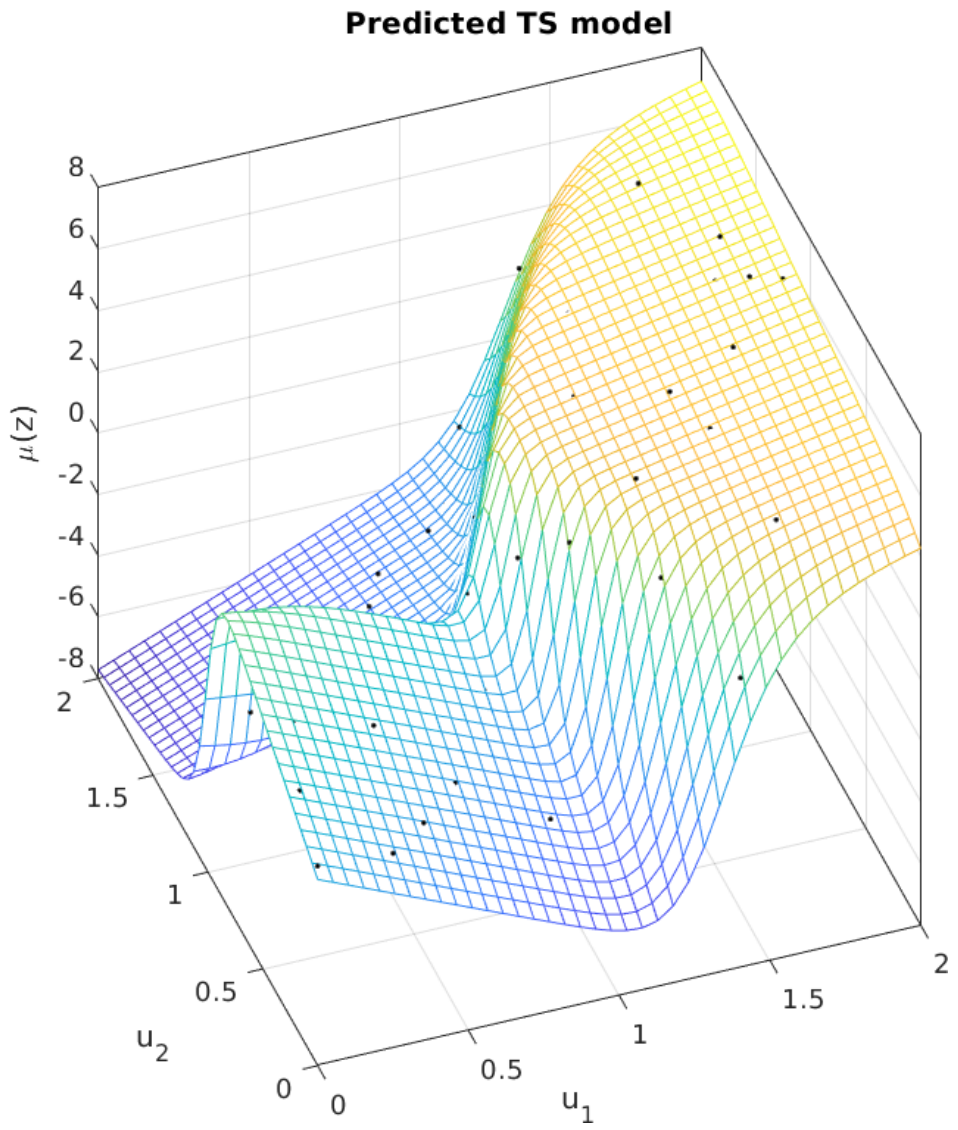
```
figure(13),clf
hold on
for i=1:model.nv
    meshc( U1_grid,U2_grid, reshape(mu_grid(:,i),N_grid,N_grid) )
```

```
end
hold off
axis square
view(-20,40)
xlabel('u_1'),ylabel('u_2'),zlabel('\mu(z)')
title( 'Membership degrees' )
set(gca,'FontSize', 14)
set(gcf,'WindowState','maximized');
```



Plot the TS model output y in 3D

```
figure(14),clf
hold on
plot3( u(:,1),u(:,2),y,'k.' )
mesh( U1_grid,U2_grid, reshape(Y_grid,N_grid,N_grid) )
hold off
title( 'Predicted TS model' )
xlabel('u_1'),ylabel('u_2'),zlabel('y')
grid on, box on
axis square
view(-20,40)
xlabel('u_1'),ylabel('u_2'),zlabel('\mu(z)')
set(gca,'FontSize', 14)
set(gcf,'WindowState','maximized');
```


Predicted TS model
μ(z)
8
6
4
2
0
-2
-4
-6
-8
2
1.5
1
0.5
0
u2
0
0.5
1
1.5
2
u1