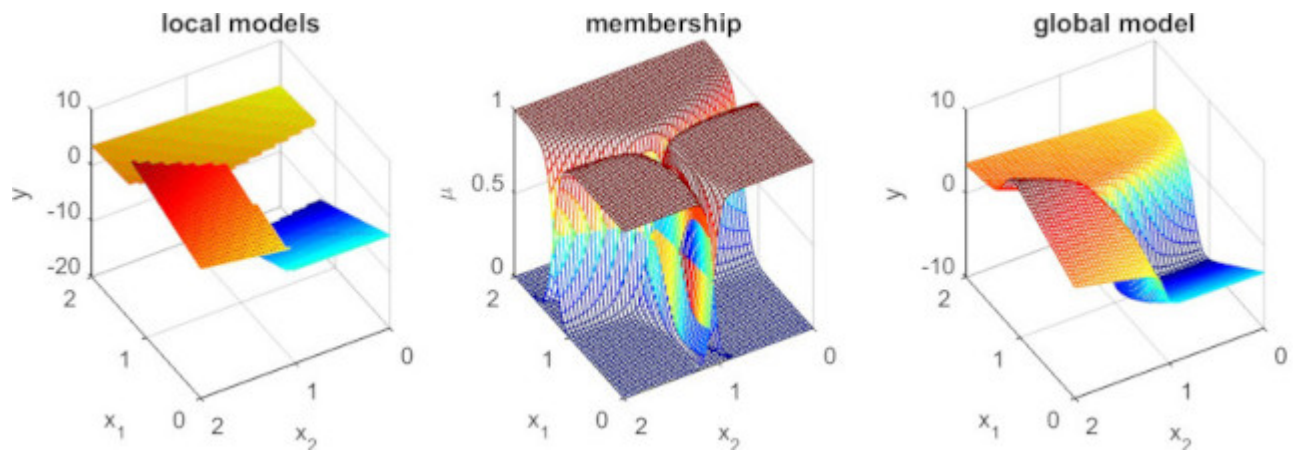


TS-Toolbox

Matlab-Toolbox zur nichtlinearen Systemidentifikation mittels lokal affiner Tagaki-Sugeno-Modelle



Version: 1.3 vom 14.9.2020

Prof. Dr.-Ing. Andreas Kroll, FG Mess- und Regelungstechnik, FB 15 Maschinenbau, Universität Kassel



URL: <http://www.uni-kassel.de/go/mrt>

Author: Axel Dürrbaum (<mailto:axel.duerrbaum@mrt.uni-kassel.de>)

Contents

- [Aufgabe: Nichtlineare Systemidentifikation und Regression](#)
- [Modellansatz: lokal affine Tagaki-Sugeno-Modelle \(TS\)](#)
- [Funktionsprinzip](#)
- [Clusterung](#)
- [Verfügbare Zugehörigkeitsfunktionen](#)
- [Lokale TS-Modelle](#)
- [Modellgütemaße](#)
- [Visualisierung](#)
- [Dokumentation](#)
- [Benötigte Software](#)
- [Installation](#)
- [Verzeichnisse](#)
- [Musterprojekte](#)
- [Implementierung](#)
- [Verfügbare Objekte \(ToDo\)](#)
- [Verfügbare Funktionen](#)

Aufgabe: Nichtlineare Systemidentifikation und Regression

- für statische MISO-Modelle

$$y(t) = f(u_1(t), \dots, u_m(t))$$

- oder dynamische MISO-Modelle

$$y(t) = f(u_1(t), \dots, u_1(t - m_1), \dots, u_m(t - 1), \dots, u_m(t - m_m), \dots, y(t - 1), \dots, y(t - n))$$

Modellansatz: lokal affine Takagi-Sugeno-Modelle (TS)

Überlagerung der c lokal affinen Teilmodelle $y_i(x)$ zu einem Gesamtmodell

$$\hat{y}(t) = \sum_{i=1}^c \mu_i(z) \cdot \hat{y}_i(x)$$

- mit den Eingangssignalen $u(t)$ und dem Ausgangssignal $y(t)$,
- der Scheduling-Variablen $z(u, y)$,
- der Zugehörigkeitsfunktionen $\mu_i(z)$,
- der Regressor-Variablen $x(u, y)$
- und den lokalen TS-Modellen $\hat{y}_i(x)$

Funktionsprinzip

- Datensatz $\{u(t), (y(t))\}$, ggf. Normierung und Split in Identifikations- und Validierungsdaten
- Ggf. Anpassung der Standard-Einstellungen der Hyperparameter * Clustering ν , c , ϵ_{FCM} * Multistart * NL-Optimierung
- Vorgabe der Anzahl der lokalen Modelle c und des Unschärfeparameters ν
- Clustering zur Ermittlung der Partitionierung bzw. Lage der Teilmodelle im Scheduling-Raum, Multistartstrategie mit Auswahl des besten Ergebnisses auf Basis des Modellfehlers auf Identifikationsdaten
- Initiale Schätzung der lokalen Modelle mittels Least-Squares-Verfahren (lokal oder global)
- Optionale Optimierung der Zugehörigkeitsfunktionen μ_i und/oder der lokalen Teilmodelle \hat{y}_i mittels nichtlinearer Optimierung der Simulation (Matlab-Funktion `lsqnonlin`)
- Unterschiedliche Wahl der Scheduling- und Regressor-Variablen möglich
- Validierung auf neuen Daten

Clustering

Eingangs- (u) oder Produktraum ($u|y$)

Implementierte Algorithmen:

- Abstandsnormen: Euklid, Mahalanobis
- Fuzzy C-Means (FCM)
- Gustafson-Kessel (GK)

Verfügbare Zugehörigkeitsfunktionen

- FCM-Type-Funktionen
- Gauss-Funktionen

Lokale TS-Modelle

- Linear: $y(t) = \sum_{i=0}^n a_i \cdot u_i(t) + a_0$
- ARX: $y(t) = A \cdot y(t - dt) + B \cdot u(t) + C$
- OE: $y(t) = A \cdot y(t - dt) + B \cdot u(t) + C + e(t)$

Modellgütemaße

auf Identifikations- und Validierungsdaten

- Maximum Absolute Error (MAE)
- Sum of Squared Errors (SSE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- Normalized Mean Squared Error (NMSE)
- Best Fit Rate (BFR)
- ??? Akaike Information Criterion (AIC)
- ??? Bayesian Information Criterion (BIC)

Visualisierung

- Clusterung (2D, n-dimensional als mehrfache 2D)
- ??? Zugehörigkeitsmaße / Regelaktivierung
- Residuen
- ??? Residualhistogramm
- Simulation oder 1-Schritt-Prädiktion auf Identifikations- oder Validierungsdaten

Dokumentation

Benötigte Software

- Matlab R2019a oder höher (Windows/Linux/macOS)
- Matlab Fuzzy Toolbox (Funktion fcm)
- Matlab Optimization Toolbox (Funktion lsqnonlin)

Installation

1. Das Archiv `TS_Modell-<datum>-dist.zip` in ein beliebiges Verzeichnis entpacken
2. Das Verzeichnis mit der Klasse `TSModel` muss in den Matlab-Suchpfad aufgenommen werden:

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

Verzeichnisse

- `TS_Toolbox`: Hauptverzeichnis der Toolbox

- TS_Toolbox/TSMModel: Klasse für TS-Modell
- TS_Toolbox/Functions: ohne Klasse TSMModel nutzbare Funktionen
- TS_Toolbox/Examples: Beispielprojekte

Musterprojekte

im Verzeichnis **Examples** befinden sich einige Projekte, die den typischen Workflow bei der Arbeit mit der Toolbox zeigen:

- statisch: Akademisches Beispiel **LiP_Akad**
- statisch: Friedmann-Funktion 2D/3D **LiP_Friedman**
- statisch: Kompressor-Kennlinie 3D **LiP_Kompressor**
- dynamisch: Narendra (SISO) **NARX_Narendra.m**
- dynamisch: Narendra (SISO) **NOE_Narendra.m**
- dynamisch: Regelkappe (SISO) **NARX_Throttle.m**
- dynamisch: Drosselkappe IAV (MISO) **NARX_Ladedruck.m**

Implementierung

Objektorientierte Realisierung:

- Objekt Daten **tsData**: $u(t), y(t)$
- Objekt Modell **tsModel**: Daten, Premisse, Konklusion
- Objekt Premisse|tsPrem|: Scheduling / Zugehörigkeitsfunktion (ToDo)
- Objekt Premisse|tsConcl|: Regressor / lokale Modelle (LiP/ARX/OE) (ToDo)

Verfügbare Objekte (ToDo)

- Daten Parameter, Methoden
- Modell Parameter, Methoden

Verfügbare Funktionen

Funktionen, die nicht auf Objekten arbeiten

\$Id: tsm_Manual.m | Fri Dec 4 14:38:44 2020 +0100 | Axel Dürrbaum \$