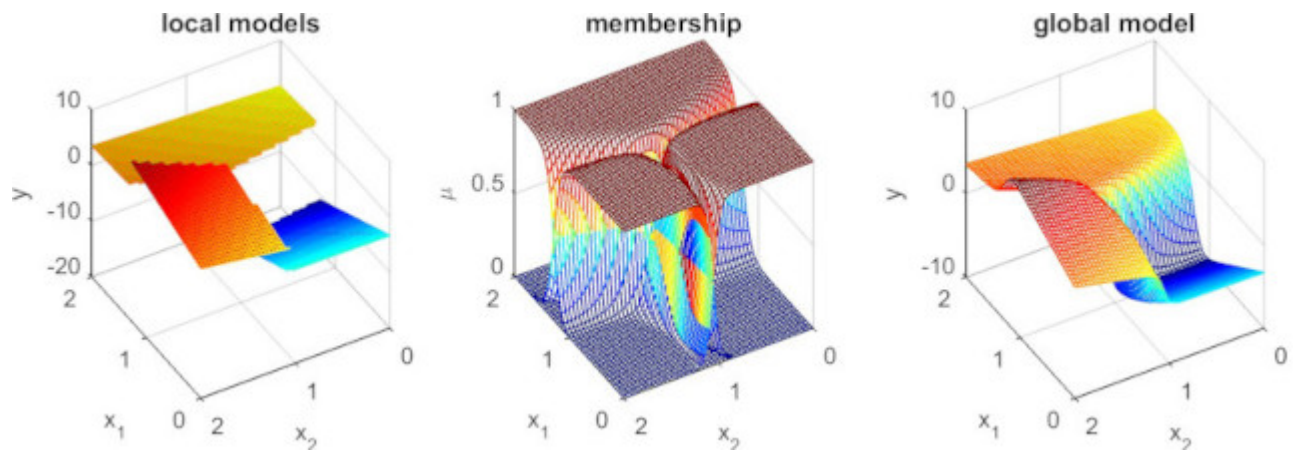


TS-Toolbox

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](#)
- [Benötigte Software](#)
- [Installation der Toolbox](#)
- [Verzeichnisse der Toolbox](#)
- [Musterprojekte](#)
- [Implementierung der Toolbox](#)
- [Verfügbare Methoden der Klasse TSModel](#)

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 * Clusterung ν, c, ϵ_{FCM} * Multistart (Anzahl) * NL-Optimierung (Abbruch-Kriterien)
- 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 den Identifikationsdaten
- Initiale Schätzung der lokalen Modelle mittels Least-Squares-Verfahren (lokal oder global)
- Optionale Optimierung der Zugehörigkeitsfunktionen ϕ_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 ($u|y$) (für statische TS-Modelle)

Implementierte Algorithmen:

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

Verfügbare Zugehörigkeitsfunktionen

- FCM-Typ-Funktionen
- Gauss-Typ-Funktionen

Lokale TS-Modelle

- Statisch: $y(t) = \sum_{i=0}^n B_i \cdot u_i(t) + C$
- 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

Benötigte Software

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

Installation der Toolbox

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

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

Verzeichnisse der Toolbox

Die Toolbox besteht aus folgenden Verzeichnissen:

- `TS_Toolbox`: Hauptverzeichnis der Toolbox
- `TS_Toolbox/TSMODEL`: Klasse für TS-Modell
- `TS_Toolbox/TSMODEL/@tsm_Base`: Basisklasse für TSMODEL
- `TS_Toolbox/TSMODEL/@tsm_Conc`: Klasse für Konklusion TSMODEL

- TS_Toolbox/TSMModel/@tsm_Data: Klasse für Datensätze u/y
- TS_Toolbox/TSMModel/@tsm_Prem: Klasse Premisse TSMModel
- TS_Toolbox/Functions: ohne Klasse TSMModel nutzbare Funktionen
- TS_Toolbox/Examples: Beispielprojekte

Musterprojekte

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

- statisch: Akademisches Beispiel `Static_Acad_auto.m`, `Static_Acad_extending.m`
- statisch: Friedmann-Funktion 2D/3D
`Static_Friedman2D_auto.m`, `Static_Friedman3D.m`
- statisch: Kompressor-Kennlinie 3D `Satic_Kompressor.m`
- dynamisch: Narendra (SISO,NARX) `NARX_Narendra.m`
- dynamisch: Narendra (SISO,NOE) `NOE_Narendra.m`
- dynamisch: Regelkappe (SISO,NARX) `NARX_Throttle.m`
- dynamisch: Drosselkappe IAV (MISO,NARX) `NARX_MISO_Ladedruck.m`

Implementierung der Toolbox

Objektorientierte Realisierung:

- Objekt TS-Modell `TSMModel`
- Objekt Premisse `tsm_Prem`: Scheduling / Zugehörigkeitsfunktion (FBF, Gauss) (ToDo)
- Objekt Konklusion `tsm_Conc`: Regressor / lokale Modelle (Static/NARX/NOE) (ToDo)
- Objekt Daten `tsm_Data`: $u(t), y(t)$
- Objekt Basis `tsm_Base` gemeinsame Basisfunktionen der Toolbox

Verfügbare Methoden der Klasse TSMModel

Diese Funktionen greifen auf die Eigenschaften der Objekte der Klassen `TSMModel` zu:

- `TSMModel`: Konstruktor der Klasse
- `TSMModel.setName`: Name des Models
- `TSMModel.addComment`: Kommentar zum Model hinzufügen
- `TSMModel.setLags`:
- `TSMModel.setSchedulingLags`:
- `TSMModel.setRegressorLags`:
- `TSMModel.setData`: Identifikationsdaten festlegen (u,y)
- `TSMModel.setDataComment`:
- `TSMModel.setDataLabel`:
- `TSMModel.setDataLimits`:
- `TSMModel.setFuzziness`: Unschärfeparameter festlegen
- `TSMModel.clustering`: Cluserung durchführen (FBF oder Gauss)

- `TSMModel.getMSF`: Zugehörigkeitsgrad für u/y berechnen
- `TSMModel.initialize`: Parameter der lokalen Modelle initialisieren
- `TSMModel.optimize`: Parameter des TS-Modells optimieren
- `TSMModel.getCluster`: Cluster-Center lesen
- `TSMModel.setCluster`: Cluster-Center manuell festlegen
- `TSMModel.getLM`: Matrizen der lokalen Modelle A,B,C lesen
- `TSMModel.setLM`: Matrizen der lokalen Modelle A,B,C manuell festlegen
- `TSMModel.predict`: N-Schritt-Prädiktor
- `TSMModel.simulate`: 1-Schritt-Prädiktor
- `TSMModel.disp`: Einstellungen und Parameter des TS-Modells
- `TSMModel.save`: TS-Modell in Matlab mat-Datei sichern
- `TSMModel.load`: TS-Modell aus Matlab mat-Datei laden
- `TSMModel.plot`: Grafiken des TS-Modells: Identifikationsdaten und Cluster
- `TSMModel.plotIdentData`: Grafiken des TS-Modells: Identifikationsdaten u,y
- `TSMModel.plotCluster`: Grafiken des TS-Modells: Cluster v in 2D
- `tsm_Prem`:
- `tsm_Conc`:
- `tsm_Data`:

Funktionen, die nicht auf Objekten der Klasse `TSMModel` arbeiten

Diese Funktionen greifen auf keine Eigenschaften der Objekte der Klassen `TSMModel` zu und sind damit universell verwendbar

- `ErrorCriteria`:
 - `AIC`: Akaike Information Criterion
 - `BIC`: Bayesian Information Criterion
 - `MSE`: Mean Squared Error
 - `RMSE`: Root Mean Squared Error
 - `NMSE`: Normalized Mean Squared Error
 - `SSE`: Sum of Squared Errors
 - `MAE`: Maximum Absolute Error
 - `BFR`: Best Fit Rate
- `plotResiduals`: Grafik der Korrelation Beobachtung/Schätzung
- `plotResidualHist`: Histogramm der Korrelation Beobachtung/Schätzung
- `plotRuleActivation`: Regelaktivierung
- `plotMSF`: Zugehörigkeitsgrad

