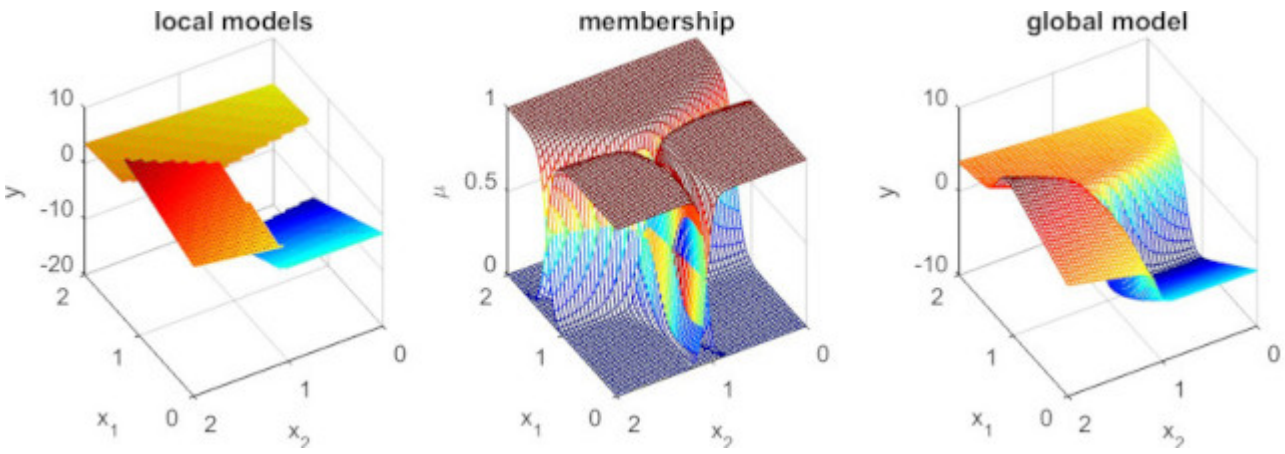
TS-Toolbox

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

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

- oder dynamische MISO-Modelle

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

Modellansatz: lokal affine Tagaki-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 Emittlung der Partitionierung bzw. Lage der Teilmodelle im Scheduling-Raum, Multistartstrategie mit Auswahl des besten Ergebnisses auf Basis des Modellfehlers auf den Identtifikationsdaten
- 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 Regeressor-Variablen möglich
- Validierung auf neuen Daten

Clusterung

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 Optimzation 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 Toobox
- 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/TSModel/@tsm_Data: Klasse für Datensätze u/y
- TS_Toolbox/TSModel/@tsm_Prem: Klasse Premisse TSModel
- TS_Toolbox/Functions: ohne Klasse TSModel 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_extendend.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 `TSModel`
- 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 Basisifunktiononen der Toolbox

Verfügbare Methoden der Klasse TSModel

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

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

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

- `tsm_Prem`:

- `tsm_Conc`:

- `tsm_Data`:

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

Diese Funktionen greifen auf keine Eigenschaften der Objekte der Klassen TSModel 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

$Id: tsm_Manual.m | Wed Feb 24 16:12:09 2021 +0100 | Axel DÃŒrrbaum $

Published with MATLAB® R2021a