LM Toolbox

For Matlab/Simulink

Contents

Theoretical background

Modelling nonlinear dynamic systems from the observed data and prior knowledge is an important area of science and engineering. Training is a key issue in the application of the Multiple Model identification because there is the added complexity of having to determine the number and the structure of sub-models as well as the parameters of the validity functions in addition to parameter identification,. There are several methods for modelling a nonlinear system and the choice of a particular modelling method depends on the aim of modelling. If the aim of modelling is control design then the identification technique should lead to simple, transparent and mathematically tractable models. In many applications it is necessary to combine the information obtained from the numerical data with heuristic knowledge. Another major requirement for nonlinear system modelling algorithm is the universality in the sense that a wide class of structurally different systems can be described. The described architecture of local models networks is capable of fulfilling these requirements and can be applied to tasks where high degree of flexibility is required. Construction of models of such structure involves a linear estimation of the regression models and nonlinear optimization of the parameters of the validity functions. A general approach to parameter estimation is to minimize some criterion that measures the difference between the output of the model and the observation data.

$$J(M,\Theta_i,\mu_i)=\frac{1}{N}\sum_{k=1}^{N}\left(\sum_{i=1}^{M}\mu_i\Theta_i\Phi(k)-y(k+1)\right)^2$$

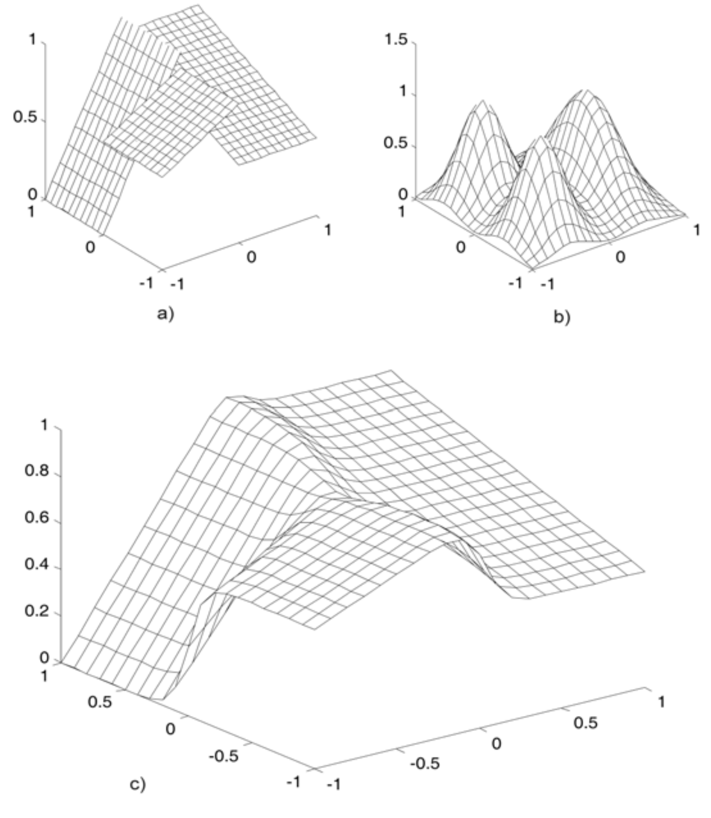

Figure 1 The nonlinear input/output approximation (c) is obtained by combining three linear models (a) with validity functions (b)

The modelling performance of the global model that consists of three local models is depicted in Figure 1, where three local models are combined with three two-dimensional validity functions to produce nonlinear approximation. The assumption for the local modelling approach is that the modelled plant has to undergo significant changes in operating conditions as it moves in the operating space. Introducing the simpler models can reduce the complexity of the nonlinear system.

Local model identification

Optimization of local model networks structure from the input-output data structure presents a challenging problem. The objective of the modeling process is to find the parameters of the validity function, parameters of the local models and also the number of local models at the same time. The modeling performance can be obtained by computation the following criterion:

(1)

where N is the number of samples, M is the number of local models, μ_i are the validity functions, $\boldsymbol{\Theta}$ are the local model parameters and regression vector $\boldsymbol{\Phi}(k)$ contains past data.

The divide-and-conquer strategy helps to improve the techniques for the design of models of nonlinear systems with the aid of computationally data-driven techniques. The modelling process involves integrating the knowledge about the system with the experimental data. The typical sources of information of the system could be:

- Experimental data, such as responses to perturbations
- A possibly incomplete nonlinear model that may be too simple or too complicated
- Qualitative knowledge, i.e. behaviours or engineer's heuristics

The model can only represent certain aspects of the system, so it is necessary to know the purpose of the model in order to decide which aspects should model capture.

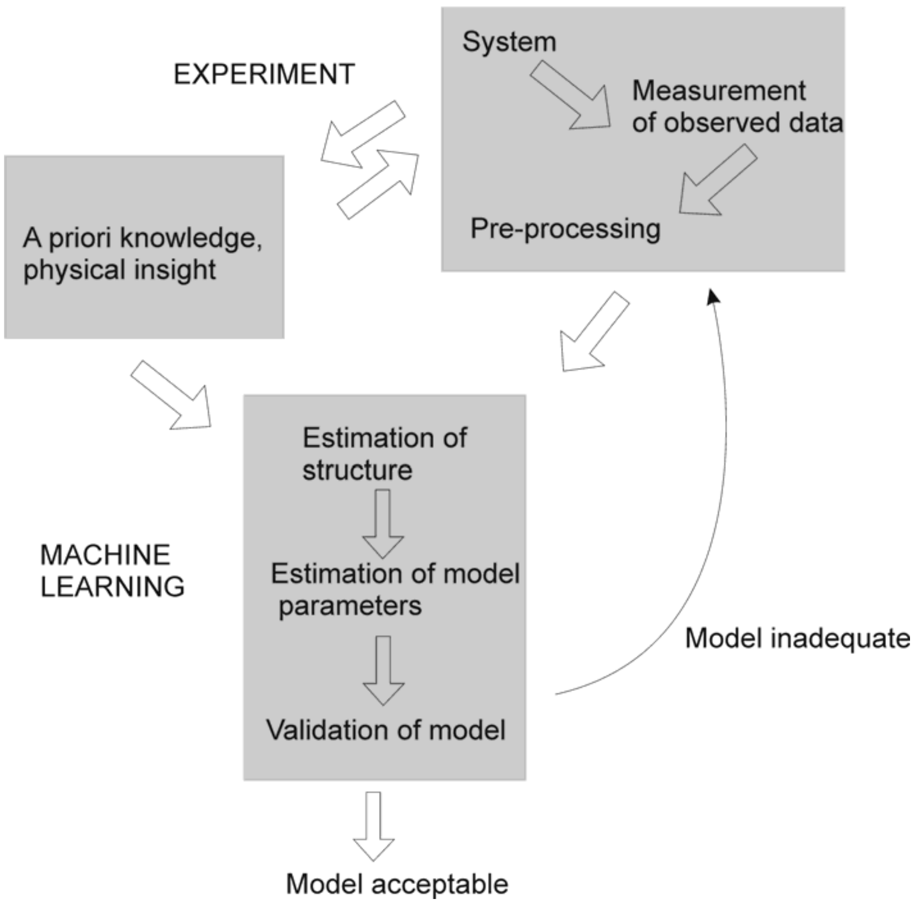

Figure 2 Engineering approach to model development

The abstract modelling cycle (Figure 2), which covers a number of further tasks that are essential in modelling:

- Experiment design and data acquisition
- Raw data processing and analysis
- Analysis of *a priori* knowledge. Physical laws and available models
- Structure and parameter optimization
- Model reduction and simplification
- Model validation, analysis and interpretation

The experimental data is used for the parameter and structure identification so it is necessary that the data covers the important aspects of the system. The input signal should exhibit enough amplitude and appropriate frequency range in order to excite all interesting modes of the plant, however these requirements are often in disagreement with the industrial practice. Once the experiments are designed, the actual input and output sequences have to be collected. In many processes it is important to have accurate models in stable areas than less accurate models throughout the whole operating range, as the system usually spends most of the time in the stable area. It is also necessary to have appropriate amount of identification data in the regions where the system is most complex or the control is most critical. Lack of data from a certain operating region can explain why the related local model parameters are not accurately identified. As it can be seen, the choice of appropriate data for structure and parameter identification is a crucial part of the modelling and requires more consideration then subsequent machine learning.

Structure identification

Although *a priori* knowledge is important to define the model structure, in some cases the system complexity is not well understood for the model structure to be specified in advance. So it is often necessary to adapt the structure based on the information in the training data. Optimization of the model structure M is, however, a difficult non-convex optimization problem. The goal of the structure identification procedure is to relate the density and the size of operating regimes to the complexity of the system. The desirable features of the identification algorithm:

- Convergence – as the number of training points increases the algorithm should provide more accurate model of the system being modelled
- Parsimony – the model structure should be the simplest possible to achieve the required accuracy

- Robustness – the model structure produced should be robust to the noisy data
- Interpretability – the model structure should be as interpretable as possible given the local models, their validity functions and available data

The techniques for the optimization of positions and dimensions of the local regions fall into several classes:

- **Fixed Selection**
  In this approach the centres are selected randomly from the input data or distributed uniformly [1]. The widths of operating regions are calculated to some thumb rules based on *a priori* information. If the complexity of the problem is unknown a large number model is necessary for fine approximation of the nonlinear system. This is a rather bad clustering solution because even a small region can be highly nonlinear.
- **Self-organizing and clustering**
  In this approach the centres of the operating regimes are trained in an unsupervised learning fashion. Abonyi *et al*, in [2] and [3] used Expectation Maximization algorithm (EM) to identify simultaneously and directly the operating regimes and the parameters of the local models. The disadvantage of these algorithms is that the local models are clustered according to the density of the data, not according to the complexity of the problem.
- **Non-optimal construction algorithms with heuristic growing strategies**
  These techniques start with a simple structure, e.g. a global linear model, and divide the input space into smaller areas. Examples are local linear model tree (LOLIMOT) [4], Johansen and Foss algorithm [5], algorithm of Aarhus [6] that is trying to find a split point in which to carry out the decomposition of the input space to reduce the prediction error.
- **Splitting and merging**
  These algorithms try to adjust the network complexity according to the complexity of the problem. The algorithm splits an operating region into two, if the behaviour of the model is not satisfactory and two neighboring models are merged together if they have nearly identical parameters to limit the complexity of network.
- **Fine-to-course learning**
  These techniques start with a large number of local models and during training the local model are merged together to get a simple structure [7].

All the structure identification algorithms are computationally expensive. This problem becomes crucial if a number of models or input dimension of the network becomes large.

Therefore, it is always advantageous to consider the maximum possible *a priori* information about the system.

Algorithms

Johansen-Foss algorithm

For modelling the nonlinear process the Johansen-Foss algorithm [5], which incorporates an outer loop for structure optimization and inner loop for parameter identification, can be used. The scheduling variables have to be known beforehand. The Johansen-Foss algorithm starts with only one model and the weighting function is unity over the whole operating d-dimensional space Z, where d is the number of scheduling variables. Using the least-squares method, the parameters estimation of the only model can be performed. The algorithm then divides the operating space into two parts. Since an infinite number of divisions is possible it is necessary to reduce the number of divisions being investigated. This is done by only allowing the regime to be divided in a direction parallel to an axis of the box *Z1* which defines an operating regime, and at a finite number of points along each axis.

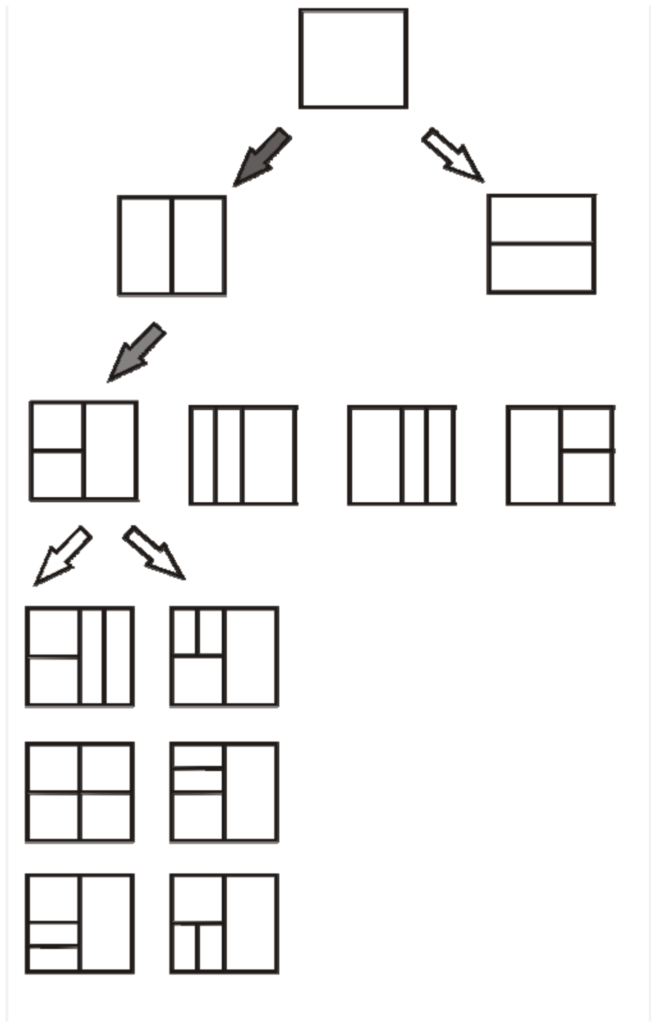

Figure 3 Operating regime decomposition

Thus for a *d*-dimensional box with *s* splitting points, a total of *ds* new decompositions are formed. New parameters of weighting functions are determined by the limits of each working regime

$$c_j = 0.5(z_j^{\max} + z_j^{\min})$$
$$F_j = \begin{bmatrix} \gamma(z_{j1}^{\max} - z_{j1}^{\min}) & 0 \\ 0 & \gamma(z_{j2}^{\max} - z_{j2}^{\min}) \end{bmatrix} \tag{1}$$

where parameter γ influences the overlapping of function and $z_j^{\max}$, $z_j^{\min}$ are limits of the *j*-th regime. For small values of γ the functions will not overlap and for large values the transition from one region to another will be smooth. Once the weighting functions parameters are obtained, the parameters of the local models can be estimated by least-squares method or weighted least-squares method.

For each split the value of cost function is calculated and the parameters of local models are determined. After considering all possible splits, the one with the lowest cost function is then chosen and the procedure is repeated. This process continues until either a maximum number, *M*, of regimes is found or until some pre-specified modelling cost criterion is satisfied. The construction algorithms can also effectively determine which variables are required to suitably decompose the operating space. If no splits are formed over a particular axis, then the variable associated with that axis can be ignored.

LOLIMOT algorithm

The term LOLIMOT stands for LOcal LInear MOdel Tree and the algorithm was developed by Nelles [4]. This method splits up the identification procedure into two parts. In the outer loop, the structure of the local model network is optimized by a tree construction algorithm. In an inner loop the parameters of the local models are estimated by local modelling technique. The construction algorithm partitions the input space by orthonormal cuts dividing the worst performance local model along the input axis, which yields the highest improvement. Consequently, the nonlinear model complexity automatically adapts to the complexity of the process.

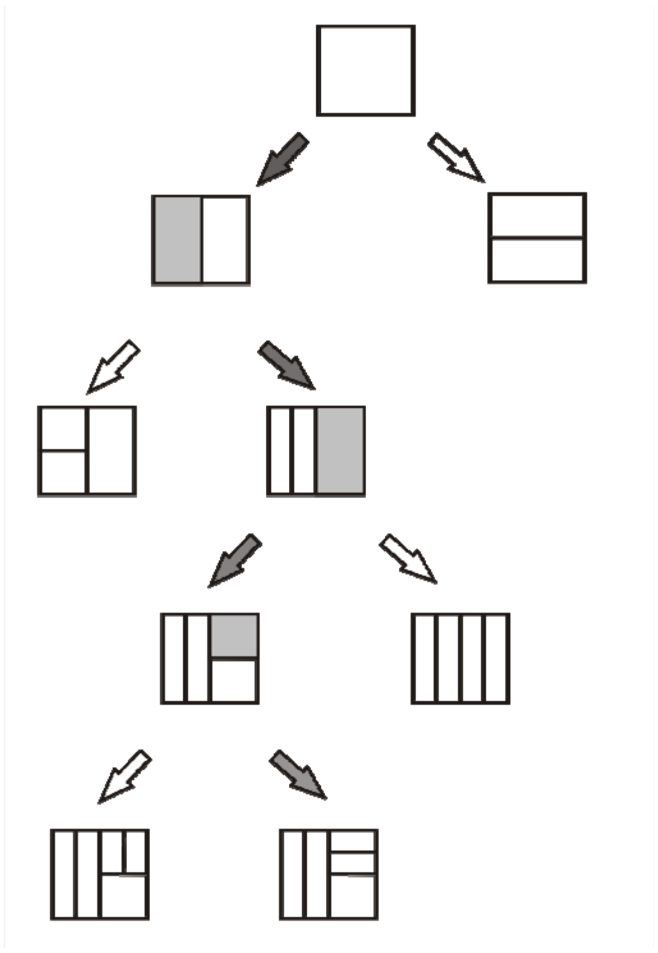

Figure 4 Operating regime decomposition

The algorithm starts with a global model that was obtained using all the input-output data. The operating space is then divided into two regimes and within each regime a local model is identified. In the next step, one of these model that fits the data worst is chosen. This model is further divided into two local models. This procedure is repeated until a stopping criterion is fulfilled.

At first sight, Johansen-Foss and LOLIMOT algorithms appear to be very similar. However, there is an obvious trade-off between computation complexity and model optimality. While the Johansen-Foss technique tries to achieve an optimal model at the expense of intensive computation, LOLIMOT minimizes the computational effort involved at the risk of producing a sub-optimal model. In general, the Johansen-Foss algorithm produces a more accurate representation than LOLIMOT but with a considerable increase in computational effort required.

Subtractive clustering

Subtractive Clustering is based on mountain clustering but uses each data point as a candidate for a cluster centre. Thus the computation is proportional to problem size instead of the problem dimension. The actual cluster centres are not necessary located at one of the data points but it is a good approximation. Subtractive clustering can be used for computation of initial cluster centres for others clustering algorithm to speed up the iteration process. A density measure at each data point is defined as:

$$D_i = \sum_{j=1}^{N} \left(-\frac{\left\| x_i - x_j \right\|^2}{\left(r_a / 2 \right)^2} \right) \tag{2}$$

where r_a is a positive constant representing the neighborhood radius. The more neighboring data points a data point has the higher value of density. After the first cluster centre is chosen as the point with the highest density value D_{c1} the density values are updated as follows:

$$D_i = D_i - D_{c1} \exp\left(-\frac{\left\| x_i - x_{c1} \right\|^2}{\left(r_b / 2 \right)^2} \right) \tag{3}$$

where r_b is a positive constant that defines a neighborhood that is reduced. Thus the density of the data points in the neighborhood of the cluster centre x_{c1} is significantly reduced. After the revision of the Density measures a new cluster centre with highest density is found. The process is repeated until the desired number of clusters is reached.

Fuzzy C-means clustering

Fuzzy c-means (FCM) clustering [8] is a method of clustering which allows one piece of data to belong to two or more clusters. This method is frequently used in pattern recognition. It is based on minimization of the following objective function:

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} \mu_{ij}^{m} \left\| x_i - c_j \right\|^2 \tag{4}$$

where *m* is the fuzziness factor greater than 1, μ_{ij} is the degree of membership of x_i in the cluster *j*, x_i is the *i*th of *d*-dimensional measured data, c_j is the d-dimension center of the cluster, and ||*|| is any norm expressing the similarity between any measured data and the center.

The degree of membership is updated through an iterative optimization of the objective function shown above, with the update of membership u_{ij} and the cluster centers c_j by:

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C}\left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|}\right)^{\frac{2}{m-1}}}, c_j = \frac{\sum_{i=1}^{N}\mu_{ij}^m x_i}{\sum_{i=1}^{N}\mu_{ij}^m} \tag{5}$$

This iteration is stop when $\max\|c^k - c^{k-1}\| < \varepsilon$, where is a termination criterion between 0 and 1, and k are the iteration steps. This procedure converges to a local minimum or a saddle point of J_m.

Gustafson-Kessel clustering algorithm

The Gustafson and Kessel algorithm [9] extends the standard fuzzy c-means algorithm by employing an adaptive distance norm, in order to detect clusters of different geometrical shapes in one data set. This algorithm has the advantage of looking for ellipsoids of variable size and orientation. Gustafson-Kessel algorithm finds clusters by minimizing the following function:

$$J_{X,m}(\boldsymbol{U},\boldsymbol{V}) = \sum_{j=1}^{N}\sum_{i=1}^{M}\mu_{i,j}^m D_{A_i}^2(x_j - c_i) \tag{6}$$

where $\boldsymbol{U}$ is a set of membership degrees , $\boldsymbol{V}$ is a set of cluster centers c, m is fuzziness factor (usually a value close to 2), X is a set of N samples x, and is a norm induced by matrix $\boldsymbol{A}$. Fuzzy covariance matrix $\boldsymbol{F}$ is defined as:

$$\boldsymbol{F}_i = \frac{\sum_{j=1}^{N}\mu_{i,j}^m (x_j - c_i)(x_j - c_i)^T}{\sum_{j=1}^{M}\mu_{i,j}^m} \tag{7}$$

The distance $D_{A_i}^2$ of a data point to the cluster centre is induced by matrix $\boldsymbol{A}_i$ as:

$$D_{A_i}^2(x) = (c_i - x)^T A_i (c_i - x) \tag{8}$$

The centers of the clusters are calculated as the weighted mean value of all membership degrees:

$$c_i = \frac{\sum_{j=1}^{N} \mu_{i,j}^m x_j}{\sum_{j=1}^{N} \mu_{i,j}^m} \tag{9}$$

The iteration is repeated until the following condition holds:

$$\max \|U(k) - U(k-1)\| < \delta \tag{10}$$

Local Model Parameteres Identification

The local models are assumed to be local auto-regressive with exogenous input (ARX) models.

$$y(k+1) = \boldsymbol{A}\begin{bmatrix} y(k-1) \\ \vdots \\ y(k-na) \end{bmatrix} + \boldsymbol{B}\begin{bmatrix} u(k-1) \\ \vdots \\ u(k-nb) \end{bmatrix} \tag{11}$$

The local least squares cost function for the *i*-th model can be written as follows:

$$J(\theta_i) = \sum_{k=1}^{N} (y_i(k) - \hat{y}_i(k,\theta_i))^2 \tag{12}$$

wchich can be rewritten into matrix form:

$$J(\theta_i) = \left(\boldsymbol{Y} - \boldsymbol{\Psi}\boldsymbol{\Theta}_i\right)^T \left(\boldsymbol{Y} - \boldsymbol{\Psi}\boldsymbol{\Theta}_i\right) \tag{13}$$

where $\boldsymbol{Y}$ is the output vector and regression matrix $\boldsymbol{\Psi}$ is given as:

$$\boldsymbol{\Psi} = \left[\boldsymbol{\Phi}^T(1)\boldsymbol{\Phi}^T(2)\ldots\boldsymbol{\Phi}^T(N)\right]^T \tag{14}$$

Knowledge about the process gain, stability and settling time could be translated into the form of inequality constraints. Thus optimization in the form of quadratic programming (QP) can be used to obtain model parameters, instead of conventional least-squared method. If the system to be identified is assumed stable there exist several limits on the parameters of local models. For example, for the system of the second order given by

$$G(z^{-1}) = \frac{b_1 z^{-1} + b_2 z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{15}$$

the following stability margins for the parameters can be introduced if the system is stable

$$\begin{aligned} a_2 &\leq 1 \\ -a_2 + a_1 &\leq 1 \\ -a_1 - a_2 &\leq 1 \end{aligned} \tag{16}$$

which can be translated as the inequality conditions for QP as:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ b_1 \\ b_2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{17}$$

The constrained optimization problem can then be formulated as a QP:

$$\min_{\theta} \left\{ \frac{1}{2} \boldsymbol{\Theta}^T \mathbf{H} \boldsymbol{\Theta} + \mathbf{c}^T \boldsymbol{\Theta} \right\} \tag{18}$$

with matrix $\boldsymbol{H}$ and vector $\boldsymbol{c}$ given by

$$\mathbf{H} = 2\boldsymbol{\Psi}^T \boldsymbol{Q} \boldsymbol{\Psi}, \quad \mathbf{c} = -2\boldsymbol{\Psi}^T \boldsymbol{Y} \tag{19}$$

and the constraints defined as:

$$\mathbf{A}_{inq} \boldsymbol{\Theta} \leq \mathbf{b} \tag{20}$$

By using the constraints during the training process, more accurate model with improved interpretability can be identified using the input-output data.

Structure Optimization algorithm

If the global model is build using larger number of local models, the quality of fitness of the model to input-output data increases. However, the idea of 'parsimony principle' should be introduced that says that among the models which explain the data well, the model with the smallest number of independent parameters should be chosen. Since the clustering of the dynamic data can result in clusters with similar or same local model parameters, the structure can be reduced in order to obtain simpler model. Therefore strategy for model reduction using prediction error and gap metric is introduced. The distance between each pair of cluster centers is computed

$$D_{ij} = \left| M_i - M_j \right| \tag{21}$$

and the pair *(i,j)* with minimal distance D_{ij} is found. The clusters M_i, M_j are merged together and new covariance matrix F_{new} and the centre of the new cluster c_{new} are defined by the following equations:

$$\begin{aligned}
&W_{new} = W_i + W_j \\
&c_{new} = \frac{W_i}{W_{new}} c_i + \frac{W_j}{W_{new}} c_j \\
&F_{new} = \frac{W_i}{W_{new}} F_i + \frac{W_j}{W_{new}} F_j + \\
&+ \frac{W_i W_j}{W_{new}^2} \left[\left(c_i - c_j\right)\left(c_i - c_j\right)^T \right]
\end{aligned} \tag{22}$$

where W_i, W_j are the numbers of data points inside the cluster. Merging of two clusters is plotted in Figure 5.

The modeling performance of the original set of models without merging MSE_{orig} and the performance of the reduced set $MSE_{reduced}$ is computed using the following prediction performance criterion:

$$MSE_{orig}, MSE_{reduced} = \frac{1}{N-1} \sum_{k=1}^{N-1} \left(\hat{y}(k+1) - y(k+1) \right)^2 \tag{23}$$

If the following condition holds:

$$MSE_{red} - MSE_{orig} < \delta_{pred} AND \delta(M_1, M_2) < \delta_{gap} \tag{24}$$

the model merging is accepted and distances between the centers in the reduced set are recomputed. To quantify the similarity between two systems a gap metric [10] is used. The gap metric $\delta(M_1, M_2)$ is much more suitable to measure the distance between two linear systems than a metric based on norms. The gap metric between the local models associated with the clusters are computed. The gap metric for two SISO dynamic models M_1, M_2 is defined as:

$$\delta(M_1, M_2) = \sup_{\omega} \frac{\left| M_1(j\omega) - M_2(j\omega) \right|}{\sqrt{1 + \left| M_1(j\omega) \right|^2} \sqrt{1 + \left| M_2(j\omega) \right|^2}} \tag{25}$$

where $0 \le \delta \le 1$, $M_1(j\omega)$ and $M_2(j\omega)$ represent the frequency responses of the systems M_1 and M_2, respectively. Two models have similar behavior in close-loop if the value of δ is close to 0 and behave differently for value of δ close to 1.

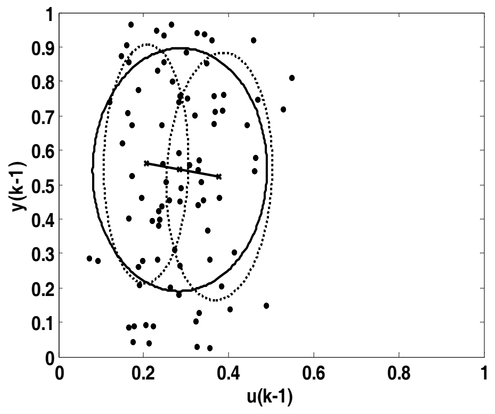

Figure 5 *Merging two clusters (dotted line – original clusters)*

If the condition (24) is false the original structure is used and new pair with minimal distance is found and algorithm repeated. The algorithm ends when no clusters can be merged without violating the condition (24).

Visualization

To reduce the dimensionality of a dataset, in order to provide visualization of the data and cluster centres the Sammon mapping [11] modified by Kovacs and Abonyi [12]. The original L-dimensional space given by $x_{ij}, i = 1,...,N, j = 1,...,L$ is transform into d-dimensional space (with d<L) with vectors $y_{il}, i = 1,...,N, l = 1,...,d$. In the modified algorithm only the distances between the points and cluster centres are considered. The distances between data points *x* and cluster centres *c* in original space are defined as:

$$d_{ij} = d(c_i, x_j) \tag{26}$$

and in d-dimensional space with cluster centres *z*:

$$d^*_{ij} = d(z_i, y_j) \tag{27}$$

The corresponding distances can be obtained by minimizing an error criterion

$$E = \frac{1}{\lambda} \sum_{i=1}^{c} \sum_{k=1}^{N} \left(\mu_{ki}\right)^m \left(d(x_k, c_i) - d^*_{ki}\right)^2 \tag{28}$$

where λ is a constant and does not influence the solution of the optimization problem. The minimization of *E* is an optimization problem with N*d variables $y_i = [y_{i1}, ..., y_{id}], i = 1, 2, ..N$. At the *t*-th iteration the values of projected data are given:

$$y_{il}(t+1) = y_{il}(t) - \alpha \left[\frac{\dfrac{\partial E(t)}{\partial y_{il}(t)}}{\dfrac{\partial E^2(t)}{\partial y_{il}^2(t)}} \right] \tag{29}$$

where α is a nonnegative scalar constant with recommended value (0.3< $0.3 \le \alpha \le 0.4$), which gives the step size for gradient search. The direction is given by:

$$\begin{aligned} \frac{\partial E(t)}{\partial y_{il}(t)} &= -\frac{2}{\lambda} \sum_{k=1,k\neq i}^{N} \left[\frac{d_{ki} - d_{ki}^*}{d_{ki} d_{ki}^*} \right] \left(y_{il} - y_{kl} \right) \\ \frac{\partial E^2(t)}{\partial y_{il}^2(t)} &= -\frac{2}{\lambda} \sum_{k=1,k\neq i}^{N} \frac{1}{d_{ki} d_{ki}^*} \left[\left(d_{ki} - d_{ki}^* \right) - \left(\frac{\left(y_{il} - y_{kl} \right)^2}{d_{ki}^*} \right) \left(1 + \frac{d_{ki} - d_{ki}^*}{d_{ki}} \right) \right] \end{aligned} \tag{30}$$

To measure the quality of the transformation the maximal value of the mean squared error between the original and projected membership function can be computed:

$$P = \max \left\| U - U^* \right\| \tag{31}$$

Function parameters

The syntex of calling a function is

[results] = function(Data, Additional Parameters)

or

[results] = function(Data, ClusterSet, Additional Parameters)

Structures Data and ClusterSet are used within Toolbox. They have the following parameters

Data.X	N x n	Matrix containing the data
Data.min	1 x n	Minimum of each data column
Data.max	1 x n	Maximum of each data column
Data.Y	N x 1	System Output
Data.ModelOrder	1 x 2	Polynomial orders of the affine ARX model
ClusterSet.c	M x n	Cluster centres
ClusterSet.F	n x n x M	Covariance matrices for each cluster
ClusterSet.U	N x M	
ClusterSet.D	N x M	Distances between data point and cluster centres
ClusterSet.A	M x na	ARX parameters

ClusterSet.B	M x nb	ARX parameters
ClusterSet.C	M x 1	ARX parameters

Toolbox functions

ClusterMerge

Syntax:

result=ClusterMerge(*ClusterSet*, *Data*)

Description

The objective of the function ClusterMerge is to join two clusters in the *ClusterSet* into a single cluster. The parameters *Cluster1* and *Cluster2* define which clusters will be merged together. Clusters are merged according their size, orientation also density of other points around their centres. The merging is described by Equation (22).

Example

The data from the pH neutralization process model were divided into 9 clusters using the GKclassification function. Then the 4th and 5th clusters were merged together using the ClusterMerge function. The original and resulting degree of membership for each cluster is depicted on Figure 6. The plot was created using the ClusterProjection function.

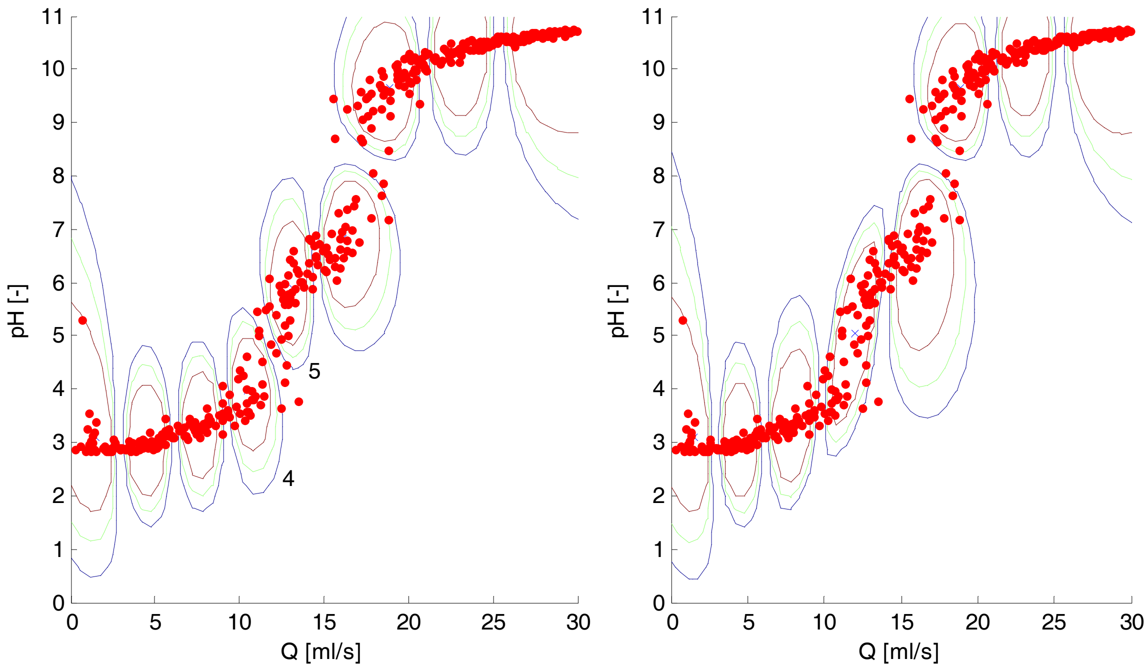

Figure 6. Merging two clusters

ClusterProjection

Syntax:

result=ClusterProjection(*ClusterSet*, *Data, Limits*)

Description

The function ClusterProjection draws Cluster centres *ClusterSet.c* and *Data.X* in the 2-d operating space defined by *Limits* [xmin xmax; ymin ymax] and also areas with degree of membership higher than 90, 80, 70 percent.

ClusterReduction

Syntax:

[results]=ClusterReduction(*Data,ClusterSet,d_gap,d_p*)

Description:

The number of clusters in the ClusterSet can be reduced using the optimiyation algorithm where similar clusters with similar local model parameters are merged together. Algorithm:

1. Create table of cluster pairs in ascending order in terms of distances between cluster centers $\|c_i - c_{i+1}\|$
2. Merge the clusters with minimal distance and create NewClusterSet with number of models decreased by 1
3. Compute the Performance indices for NewClusterSet and original ClusterSet
4. Compute the gap metric between the models being merged
5. Test the merging criterion

$$MSE_{red} - MSE_{orig} < \delta_{pred} AND \delta(M_1, M_2) < \delta_{gap} \tag{32}$$

If the condition holds the merging is accepted and a reduced ClustersSet is obtained. Algorithm returns to Step 1 and new table with cluster pair is created. If the condition does not hold a new pair with minimal distance is found and algorithm returns to Step 2. The algorithm ends when no merging is possible without violating the condition (32).

ComputeValidity

Syntax:

[results]=ComputeValidity(*Data,ClusterSet*)

Description:

The function ComputeValidity returns the membership functions *results.U* given the centres and/or covariance matrice in *ClusterSet* and *Data.X* matrix.

DataNormalize

Syntax:

result=DataNormalize(*Data*)

Description:

With function DataNormalize the data are normalized between [0,1] with algorithm:

$$X = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{33}$$

The vector of maxima and minima (*Data.max*, *Data.min*) is also computed.

DrawDivision

Syntax:

[results]=DrawDivision(*Data,ClusterSet*)

Description:

The function DrawDivision draws the rectangular operating regions that were identified by LOLIMOT or Johnasen-Foss algorithm.

FCMclassification

Syntax:

result=ClusterProjection(*Data, NumClust, InitialC, MaxDif*)

Description:

The algorithm divides the data points from *Data.X* matrix into *NumClust* clusters. The iterative Fuzzy C-means algorithm is finished when maximal difference between the Cluster centres in step *k* and *k-1* is less then *MaxDif.* Initial position of the cluster centers *InitialC* can be provided to speed up the iterative process. The algorithm:

1. If initial centres are not provided initialize matrix $U = u_{ij}, i = 1,..., samples, j = 1,..., NumClust$ randomly
2. Calculate the centers vector $C = c_j$

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^m x_i}{\sum_{i=1}^{N} u_{ij}^m} \tag{34}$$

3. Update $U^{(k)}$

$$u_{ij} = \frac{1}{\sum_{k=1}^{NumClust} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}} \tag{35}$$

4. If $\left\|U^{(k+1)} - U^{(k)}\right\| < MaxDif$ then Stop; otherwise return to Step 2.

Example

The data from the pH neutralization process model were divided into 5 clusters using the FCMclassification function. The Figure 7 showing the degree of membership was created using the ClusterProjection function.

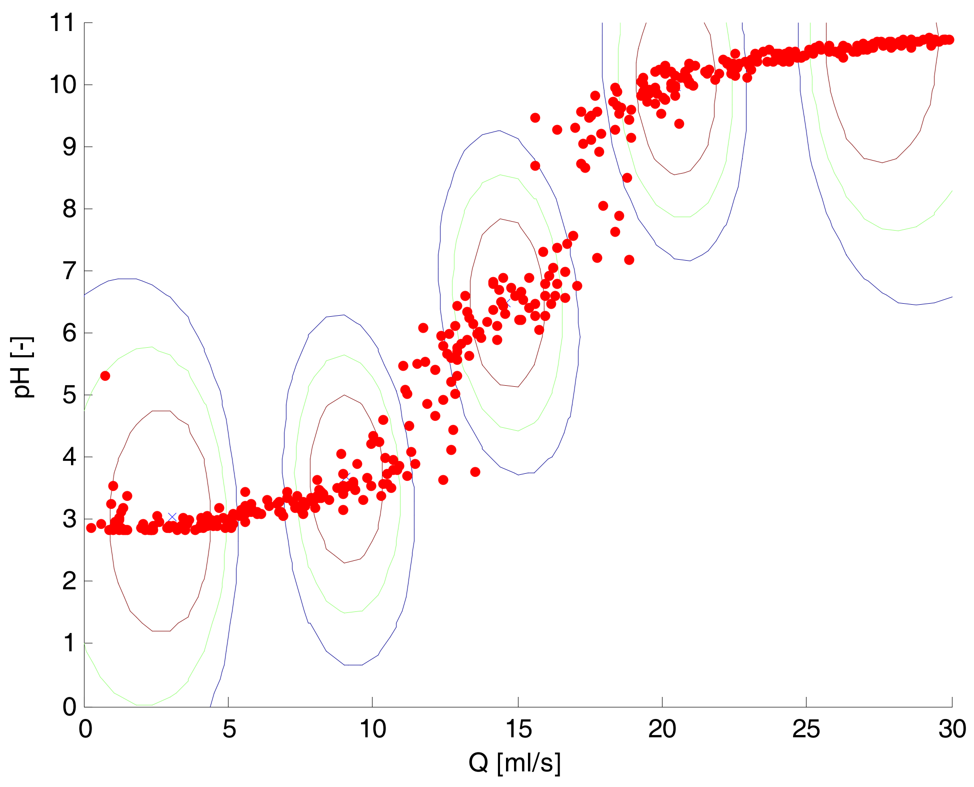

Figure 7. Classification using FCM algorithm

GKclassification

Syntax:

[results]=GKclassification(*Data,NumClust, MaxDif*)

Description:

The modified Gustafson-Kessel algorithm developed in [13] to reduce the numerical drawbacks of Gustafson-Kessel algorithm was implemented in the function. For a given dataset *Data.X*, and number of clusters *NumClust* and termination tolerance the algorithm finds the cluster centres *result.c* and covariance matrices *result.F* that minimizes the cost function (6). The initial degree of membership $U^{(l)}$ is random. The algorithm is then:

1. Calculate cluster centres

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^m x_i}{\sum_{i=1}^{N} u_{ij}^m}, 1 \leq j \leq NumClust \tag{36}$$

2. Compute the covariance matrix for each cluster

$$F_j = \frac{\sum_{i=1}^{N} \mu_{ij}^m \left(x_i - c_j\right)\left(x_i - c_j\right)^T}{\sum_{i=1}^{N} \mu_{ij}^m} \tag{37}$$

1. Identity matrix I is added to each covariance matrix F_j:

$$\boldsymbol{F}_j = (1-\gamma)\boldsymbol{F}_j + \gamma(\boldsymbol{F}_0)^{1/n}\boldsymbol{I} \tag{38}$$

Eigenvalues λ_{jk} and eigenvectors ϕ_{jk} are extracted from F_j and maximum is found

$\lambda_{j,\max} = \max_k \lambda_{jk}$

$$\lambda_{j,\max} = \lambda_{jk} / \beta, \forall k \; for \; which \; \lambda_{i,\max} / \lambda_{ij} \geq \beta \tag{39}$$

And covariance matrices are is reconstructed using:

$$F_j = \left[\phi_{j,1} \ldots \phi_{j,n}\right] diag\left(\lambda_{j,1} \ldots \lambda_{j,n}\right)\left[\phi_{j,1} \ldots \phi_{j,n}\right]^{-1} \tag{40}$$

2. Distances between the centres and datapoints are computed:

$$D_{ij}^2 = \left(x_i - c_j\right)^T \left[\det\left(\boldsymbol{F}_j\right)^{1/n} \boldsymbol{F}_j^{-1}\right]\left(x_i - c_j\right) \tag{41}$$

3. The partition matrix is updated:

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{NumClust} \left(\frac{D_{ij}}{D_{kj}}\right)^{2/(m-1)}} \tag{42}$$

4. Check the termination condition:

$$\left\| U^{(l)} - U^{(l-1)} \right\| < MaxDif \tag{43}$$

If the condition is false the algorithm is repeated.

Example:

The data from the pH neutralization process model were divided into 5 clusters using the GKclassification function. The Figure 8 showing the degree of membership was created using the ClusterProjection function.

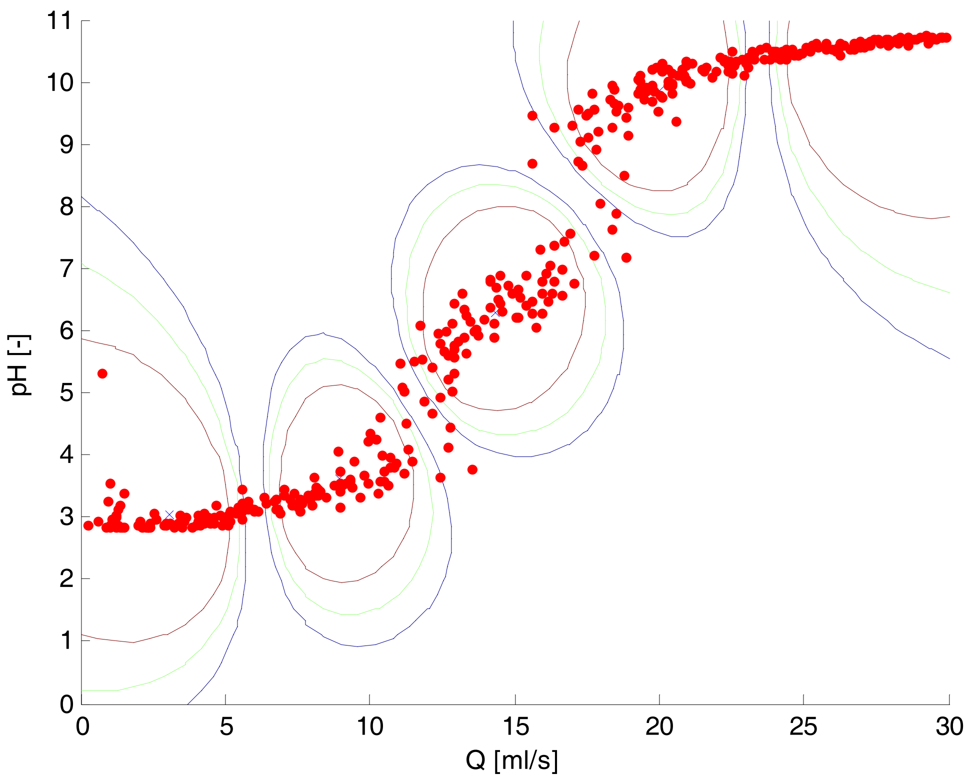

Figure 8. Clustering the data using Gustafson-Kessel algorithm

JohansenFoss

Syntax:

[results Performance]=JohansenFoss(*Data, Limits, NumClust, CuttingPoints, gama*)

Description:

The algorithm divides the *d*-dimensional operating space given by the *Limits* $Limits = \begin{bmatrix} x_{1\min} & x_{1\max} \\ \vdots & \vdots \\ x_{d\min} & x_{d\max} \end{bmatrix}$ into *NumClust* regions. Each operating regime is divided in a number of points provided by the parameter *CuttingPoints.* The overlapping of the weighting functions is controlled using the *gama* parameter.

Example:

Data from the nonlinear system with the outputs given as:

$$\begin{aligned} y(k) &= 0.7788y(k-1) + 1.106u(k-1), \quad y(k-1) < 7.5 \\ &= 0.86y(k-1) + 0.1331u(k-1) + 0.8504, \quad 7.5 \le y(k-1) \le 10.3525 \\ &= 0.7788y(k-1) + 1.106u\left(k-1\right), \quad y(k-1) > 10.3525 \end{aligned} \tag{44}$$

which was excited by the growing input signal were used to identify the global model with 5 local models using the Johansen-Foss algorithm. The data for identification were stored in *Data.X* and *Data.Y* matrices, where Data.X consists of past input and output values. The position of the clusters was developed using *Clusters* = JohansenFoss(*Data.X*, [0 6 0 18], 5, 1) and then operating regimes were drawn using DrawDivision function (Figure 9).

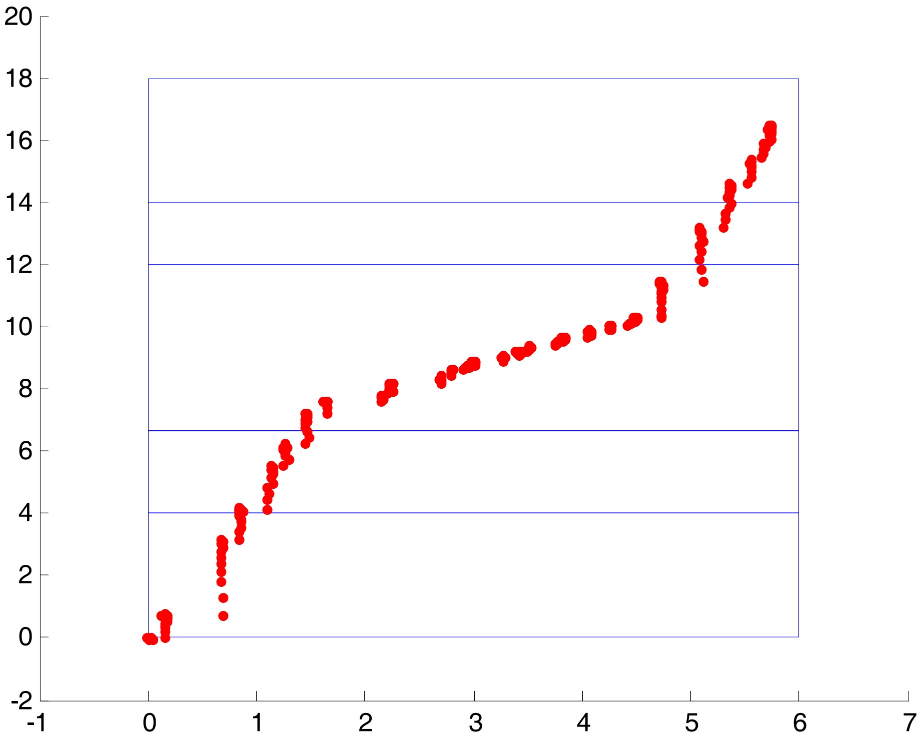

Figure 9. Divided operating space using the Johansen-Foss algorithm

LOLIMOT

Syntax:

[results Performance]=LOLIMOT(*Data, Limits, NumClust, gama*)

Description:

The algorithm divides the *d*-dimensional operating space given by the *Limits* $Limits = \begin{bmatrix} x_{1\min} & x_{1\max} \\ \vdots & \vdots \\ x_{d\min} & x_{d\max} \end{bmatrix}$ into *NumClust* regions. The overlapping of the weighting functions is controlled using the *gama* parameter.

The algorithm:

5. Divide the initial operating space into two regions and identify the local model parameters for each model
6. Find the worst submodel by evaluating I_j

$$I_j = \sum_{k=1}^{N} \mu_j(x(k))\left(y(k+1) - \hat{y}(k+1)\right) \tag{45}$$

where $\mu_j(x(k))$ is the validity of model j at k sample.

7. Check all possible division. Split the region in axis-ortogonal cut. Compute new membership functions and local model parameters.

Find the best division. The division with the best overall model error is selected. If the number of local models is less than *NumClust* return to step 2; otherwise Stop.

Description: Data from the nonlinear system with the outputs given as:

$$\begin{aligned} y(k) &= 0.7788y(k-1) + 1.106u(k-1), \quad y(k-1) < 7.5 \\ &= 0.86y(k-1) + 0.1331u(k-1) + 0.8504, \quad 7.5 \le y(k-1) \le 10.3525 \\ &= 0.7788y(k-1) + 1.106u\left(k-1\right), \quad y(k-1) > 10.3525 \end{aligned} \tag{46}$$

which was excited by the growing input signal were used to identify the global model with 5 local models using the LOLIMOT algorithm. The data for identification were stored in *Data.X* and *Data.Y* matrices, where Data.X consists of past input and output values. The position of the clusters was developed using *Clusters* = LOLIMOT(*Data.X*, [0 6 0 18], 5, 1) and then the degree of membership for clusters were drawn using ClusterProjection function (Figure 10).

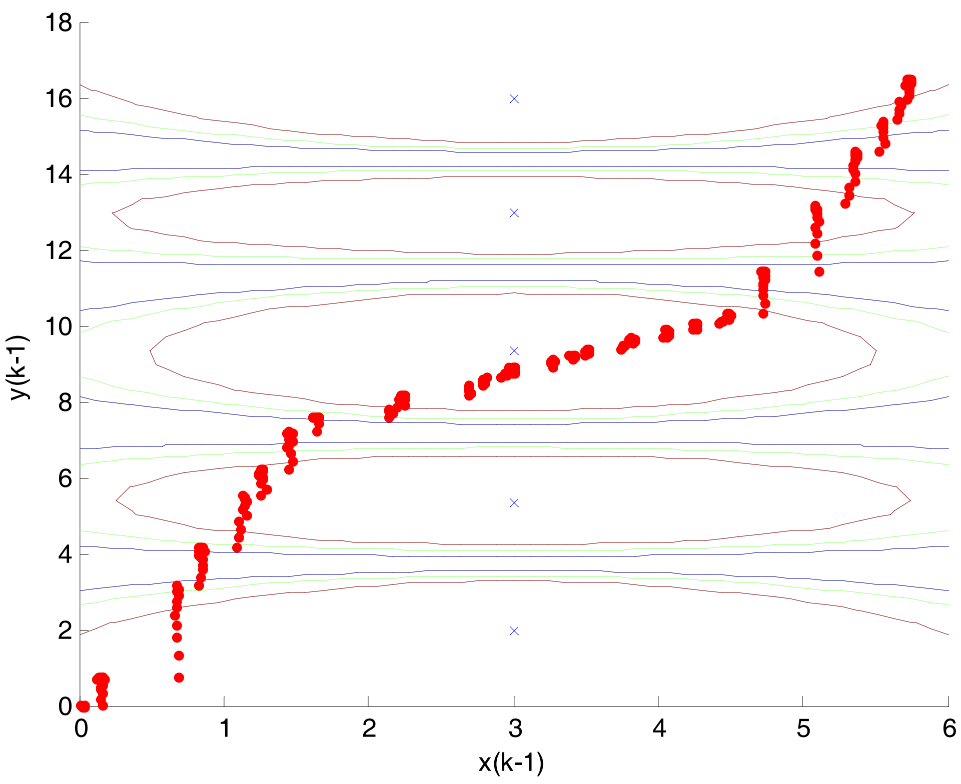

Figure 10. Division of the operating space using the LOLIMOT algorithm

MultipleIdent

Syntax:

[results]=MultipleIdent(*Data,ClusterSet, Constraints*)

Description:

The function MultipleIdent identifies the local model parameters of all the clusters in the ClusterSet by solving the quadratic programming problem. The constrained optimization problem for *j*-th model can then be formulated as a QP:

$$\min_{\theta_j} \left\{ \frac{1}{2} \boldsymbol{\Theta}_j^T \mathbf{H} \boldsymbol{\Theta}_j + \mathbf{c}^T \boldsymbol{\Theta}_j \right\} \tag{47}$$

with matrix ***H*** and vector ***c*** given by

$$\mathbf{H} = 2\boldsymbol{\Psi}^T \boldsymbol{Q}_j \boldsymbol{\Psi}, \quad \mathbf{c} = -2\boldsymbol{\Psi}^T \boldsymbol{Y} \tag{48}$$

For *j*-model only data with maximal membership function among the models at sample *i* are used for identification. Matrix

$\boldsymbol{\Psi}_i = [-y(i),\cdots,-y(i-na+1),u(i),\cdots,u(i-nb+1),1]$ contains the past values of output and input, matrix $\boldsymbol{Q}_j = \begin{bmatrix} \mu_{1j} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mu_{Nj} \end{bmatrix}$. The constraints for optimization can be provided as in quadprog function.

$$\begin{aligned} &\boldsymbol{A\Theta} < \boldsymbol{b} \\ &\boldsymbol{A}_{eq}\boldsymbol{\Theta} = \boldsymbol{b}_{eq} \\ &LB < \boldsymbol{\Theta} < UB \end{aligned} \tag{49}$$

For i-th model

Constraints.A=A

Constraints.b=b

Constraints.Aeq=Aeq

Constraints.beq=beq

Constraints.LB=LB

Constraints.UB=UB

PerfValidate

Syntax:

[results]=PerfValidate(*Data,ClusterSet*)

Description:

The function PerfValidate evaluates the unseen data *Data.X* using the centres, covariance matrices and local model parameters *(ClusterSet.c, ClusterSet.F, ClusterSet.A, ClusterSet.B, ClusterSet.C)* identified through any method. Returns the value of mean-squared error:

$$MSE = \frac{1}{samples}\sum_{i=1}^{samples}\left(y(i) - \sum_{j=1}^{NumClust}\mu_{ij}y_j(i)\right)^2 \tag{50}$$

PredictControl

Syntax:

[results]=PredictControl(*ClusterSet,PastData,Reference, Constraints, lambda*)

Description:

The function PredictControl returns the value of control increment obtained by minimization of the control error on the future trajectory *Reference*. The *ClusterSet* parameters are used as a model of the process. *PastData* vector contains the past inputs and outputs of the process and are also used for computation the membership degree of each local model. *Constraints.A, Constraints.b, Constraints.Aeq, Constraints.beq, Constraints.LB, Constraints.UB* can be specified as in quadprog function.

Sammon

Syntax:
[results]=Sammon(*ClusterSet, Data, MaxStep, Alpha*)

Description:

The modified Sammon function projects the Data.X in the n-dimensional space into a 2-dimensional space. The MaxStep specifies the maximum iteration number and Alpha the step size of the gradient method.

Subtractive

Syntax:
[results]=Subtractive(*Data, NumClust, ra*)

Description:

The cluster centers are iteratively found based on a density measure given by Equation (2). The resulting centers can be used for initialization of the FCM algorithm. *Ra* is a positive constant representing the neighboring algorithm.

Reference

[1] GOLLEE, H. HUNT, K.J. Nonlinear modelling and control of electrically stimulated muscle: a local model network approach. *International Journal of Control*. 1997, Vol. 68, no. 6, pages 1259-1288.

[2] ABONYI, J. CHOVAN, T., SZEIFERT, F. Identification of Nonlinear Systems using Gaussian Mixture of Local Models. *Hungarian Journal of Industrial Chemistry*. 2001, Vol. 29, pages 134-139.

[3] ABONYI, J. TAR, J., SZEIFERT, F. Identification of MIMO Processes by Fuzzy Clustering. In: Proceedings of IEEE International Conference on Intelligent Systems. 2001, Helsinki, Finland.

[4] NELLES, O. Orthonormal basis functions for nonlinear system identification with local linear model trees (LOLIMOT), In: *Proceeding of 11th IFAC Symposium on System Identification.*1997, Kitakyushu, Fukuoka, Japan.

[5] JOHANSEN, T.A., FOSS, B.A. Identification of nonlinear system structure and parameters using regime decomposition, *Automatica*. 1995, Vol. 31.

[6] AARHUS, L. Nonlinear empirical modelling using local PLS models. Ph. D. Thesis. 1994, University of Oslo, Norway.

[7] OMOHUNDRO, S.M. Bumptrees for efficient function, constraints and classification learning. In: Advances in Neural Information Processing Systems 3. Morgan Kaufmann Publishers. 1991, San Francisco, CA, pages 693-699.

[8] DUNN, J. C. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters, *Journal of Cybernetics*. 1973. Vol. 3, pages 32-57.

[9] GUSTAFSON, D.E., KESSEL, W.C. Fuzzy clustering with a fuzzy covariance matrix, in Proc. of IEEE Conference on Decision and Control, 1978, pp.761–766.

[10] GEORGIOU, T.T, SMITH, M.C. Optimal robustness in the gap metric. *IEEE Transactions on Automatic Control*, vol. 35, 1990, pages 673-686.

[11] SAMMON, J.W. Jr. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 1969. Vol. 18. pages 401-409.

[12] KOVACS, A., ABONYI, J. Vizualization of fuzzy clustering results by modified sammon mapping. Proceedings of the 3rd International Symposium of Hungarian Researchers on Computational Intelligence, 2002. pages 177-188.

[13] BABUSKA, R. VAN DER VEEN, P.J., KAYMAK, U. Improved covariance estimation for Gustafson-Kessel clustering. In Proceedings of 2002 IEEE International Conference on Fuzzy Systems, Honolulu, 2002, pages 1081-1085.