

Generation of Takagi-Sugeno Fuzzy Systems with Minimum Rules in Modeling and Identification

Feng Wan and Chenglin Hu

Department of Electrical and Electronics Engineering
Faculty of Science and Technology
University of Macau
E-mail: fwan@umac.mo, ma76567@umac.mo

Abstract— An improvement is presented for the tunnel algorithm introduced in [17] for determining the minimum number of rules required by a fuzzy system for modeling and identification in the SISO case. The original tunnel algorithm can deal with only Mamdani type fuzzy systems, while the Testing Point algorithm proposed in this paper, can be used to construct Takagi-Sugeno type fuzzy systems with consequent parts of any order and has the potential to be further extended to the MISO case. Numerical examples are given to illustrate the idea and improvement.

Keywords—Fuzzy modeling and identification; Takagi-Sugeno fuzzy systems; Rule reduction

I. INTRODUCTION

How to construct fuzzy systems using as less as possible rules with guaranteed performance is a difficult but important problem, which has attracted much attention over the fuzzy society [1], [3]-[15], [17], [19]-[24]. Among them, a so-called tunnel algorithm (TA) was proposed by [17], which can partition the input domain in a non-uniform manner according to the sampling data distribution and the approximation error tolerance, when dealing with the following problem in fuzzy modeling and identification.

Given a limited number of input-output data pairs or say samples $\{x_k, g(x_k)\}$, $k = 1, 2, \dots, N$, of a single-input-single-output (SISO) system with an unknown analytical representation $g(x)$, it is expected to construct a fuzzy system $f(x)$ using M rules to approximate the unknown function $g(x)$, where M should be the minimum, and the approximation errors at each data point are less than a predefined error tolerance ε . From mathematical point of view, the task can be formulated as the following optimization problem:

$$\begin{aligned} & \text{minimize: } M, \\ & \text{subject to: } \max_{k=1,2,\dots,n} |g(x_k) - f(x_k)| \leq \varepsilon. \end{aligned} \quad (1)$$

The rest of this section briefly reviews the tunnel algorithm proposed in [17]. Restricted by the error tolerance ε , each sampling point $p_k = \{x_k, g(x_k)\}$, $k = 1, 2, \dots, N$, as shown in

Fig. 1, has its upper limit $p_k^+ = \{x_k, g(x_k) + \varepsilon\}$ and lower limit $p_k^- = \{x_k, g(x_k) - \varepsilon\}$. Connecting all the points p_k , p_k^+ and p_k^- respectively by lines in order along axis x , forms three piecewise lines PL , PL_ε^+ and PL_ε^- . PL_ε^- and PL_ε^+ are called the lower and upper boundaries of PL , respectively. Tunnel $P(\varepsilon)$ is the polygon area encircled by PL_ε^+ and PL_ε^- . The line segments $e = \overline{p_1^+ p_1^-}$ and $e' = \overline{p_N^+ p_N^-}$ are called the edges of the tunnel $P(\varepsilon)$.

Owing to the nice property of the Mamdani type fuzzy systems, the design problem can be reduced to finding as less as possible continuous line segments with guaranteed approximation accuracy. The key idea is to convert the optimization problem in (1) into an *edge-visibility* problem using the concepts and techniques from computational geometry. More specifically, “ a is visible from b ” means there exists a straight-line path connecting a and b , where a and b can be points, edges, polygons or any combination of them. If the edge $\overline{p_N^+ p_N^-}$ is invisible from $e = \overline{p_1^+ p_1^-}$, then the whole polygon $P(\varepsilon)$ can be divided into two parts: the *visible polygon* and the *invisible polygon*. The intersection of the boundaries of the visible and invisible polygons is called the *window* from the edge e in the polygon P .

The Tunnel Algorithm (TA)

The tunnel algorithm goes through five steps as follows.

Step 1: Initialization. Form the upper and lower boundaries PL_ε^+ and PL_ε^- for the given data set $\{x_k, g(x_k)\}$, $k = 1, 2, \dots, N$.

The initial tunnel is $P_1 = P(\varepsilon)$. Set the edge $e_1 = \overline{p_1^+ p_1^-}$ and $i = 1$.

Step 2: Searching.

WHILE $\overline{p_N^+ p_N^-}$ is not visible from e_i in polygon P_i DO

$e_{i+1} = \text{window from } e_i \text{ to } \overline{p_N^+ p_N^-} \text{ in polygon } P_i$;

This work is supported in part by the Macau Science and Technology Development Fund under Grant FDCT/017/2007/A and the University of Macau Research Committee under Grant RG076/07-08S/WF/FST.

P_{i+1} = invisible polygon of e_i containing $\overline{p_N^+ p_N^-}$ in polygon P_i ;

q_i = point of intersection of the line containing e_{i+1} and the edge e_i ;

$i = i + 1$.

Step 3: Output the searching results. $m = i + 1$; find a point q_{m-1} on the edge e_i and a point q_m on the edge $\overline{p_N^+ p_N^-}$ that are visible from each other within the polygon P_{m-1} .

Step 4: Use the data at the points $q_l, l = 1, 2, \dots, m$, to compute the parameters $\{a_l, b_l\}$ of the line segments $\overline{q_l q_{l+1}}$, for $l = 1, 2, \dots, m - 1$.

Step 5: Determine the minimum rule number and compute the parameters for the Mamdani type fuzzy system. According to Theorem 1 in [17], the number of fuzzy rule M equals $m + 1$. The centers of M fuzzy sets in the IF part can be expressed as $A_l = x(q_l), l = 1, 2, \dots, M$. The corresponding centers of M fuzzy sets in the THEN part is $y_l = y(q_l), l = 1, 2, \dots, M$. With $x(q_l)$ and $y(q_l)$ standing for the coordinate values of point q_l , the l -th rule can be expressed as

$$R_l: \text{IF}(x \text{ is } A_l), \text{ THEN}(y \text{ is } y_l). \quad (2)$$

The number of the rules extracted by the tunnel algorithm has been proved to be the minimum in [17].

Remark 1: The original tunnel algorithm gives an optimal solution to the problem (1) restricted inside the tunnel; however, this solution can be further improved. According to the optimization problem (1), the built fuzzy system is constrained to the accuracy requirement on the N output values observed at the input values $x_k, k = 1, 2, \dots, N$, and no more information other than these discrete points is available, thus no further restriction is necessary. Therefore, except at these N discrete points $x_k, k = 1, 2, \dots, N$, the upper and lower boundaries of the tunnel, need not to be considered as constraints for accuracy. Next section begins with a modification aiming at breaking the tunnel and the consequent limitation aforementioned, to improve the original algorithm for a better result.

II. TESTING POINT ALGORITHM

In this section, two examples in [17] are first revisited, to show the above-mentioned restriction due to the tunnel, and the modification is proposed to break the restriction. Then, in order to extend the algorithm for the design of Takagi-Sugeno (TS) type fuzzy system, an algebraic approach is proposed which can be used to construct TS type fuzzy system with any-order consequences.

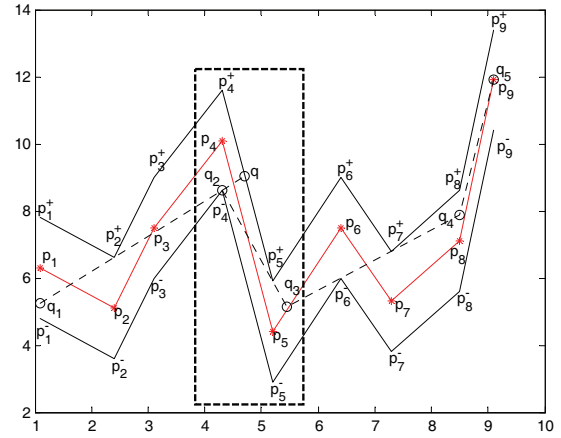


Fig. 1. Result of the tunnel algorithm. (red “*” : the available data points; dashed line: the output of the fuzzy system)

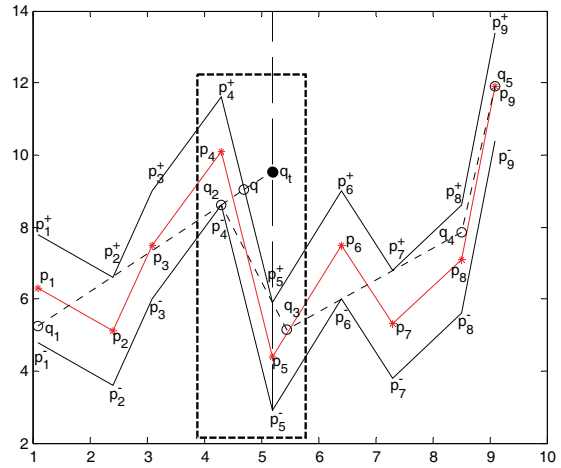


Fig. 2. Illustration of how to remove the restriction due to the tunnel.

Example 1 [17]: Construct a standard Mamdani type fuzzy system given the following nine pairs of input-output data:

$$\left\{ \begin{aligned} & (1.1, 6.3), (2.4, 5.1), (3.1, 7.5), (4.3, 10.1), (5.2, 4.4) \\ & (6.4, 7.5), (7.3, 5.3), (8.5, 7.1), (9.1, 11.9) \end{aligned} \right\}.$$

The number of rules in the fuzzy system should be minimal, while the approximation error should be less than 1.5. The result of the tunnel algorithm is shown in Fig. 1.

The initial edge is the line segment $e = \overline{p_1^+ p_1^-}$. Because $\overline{p_N^+ p_N^-}$ is invisible, the whole polygon should be divided into two parts to form a window for the next search. Tunnel algorithm finds the line segment $\overline{p_4^- q}$ as the first window. Within the square area in Fig. 1, obviously the end point q of the window determined by the tunnel degrades the optimality. If the tunnel boundaries can be broken, the new constructed window will have a wider field of vision. This improved window begins with p_4^- and ends at the point intersected by $x = x_5$ (see Fig. 2 for details, where $\overline{p_4^- q_i}$ is the improved window). As stated in Remark 1, whether the output of the

fuzzy system goes through the line segments $\overline{p_4^+ p_4^-}$ and $\overline{p_5^+ p_5^-}$ is the only concern on accuracy, since what response in between the region (x_4, x_5) does not violate the accuracy tolerance. Example 1 shows the idea of the modification, its performance is to be elaborated through a comparison study in the next example.

Example 2 [17]: A standard Mamdani type fuzzy system is constructed to model an unknown plant described by the nonlinear function $g(x) = \cos((e^x)^2)$. The modeling error tolerance is $\varepsilon = 0.05$. A set of input-output pairs of the plant are provided, in which the inputs x_k are evenly distributed in the interval $[0, 2]$ with space 0.05. Fig. 3 shows the response of the final fuzzy system developed by the original tunnel algorithm while Fig. 4 shows the results from the improved tunnel algorithm. Red “*” represents the sampling data, boundaries of the tunnel are depicted by black lines, black circles are the q 's with each of which one can conclude a fuzzy rule, and the output of the fuzzy system is drawn by dashed lines. The areas marked by blue dashed ellipses show the improvement owing to the proposed algorithm. In or near the marked areas, there may be a rule reduction due to the improvement through tunnel breaking. It is clear that in most of the areas the number of q points for rules in Fig. 4 is less than that in Fig. 3. In fact, the original tunnel algorithm extracts 25 fuzzy rules while the improved algorithm finally extracts only 19.

The above example shows a large improvement of the proposed algorithm in determining the minimum number of fuzzy rules. However, the algorithm is based on the computational geometry, which is not easy for further extension to high-order or high-dimension cases. In what follows, the strategy using visibility concepts from computational geometry will be reformulated and extended to the design of TS type fuzzy systems, which are much more widely used than Mamdani type fuzzy systems in real applications.

Compared with Mamdani type fuzzy systems, the major difference is that the consequent parts of a TS type fuzzy system are polynomial functions of its inputs. Consider the simplest case, where the l -th rule of a TS type fuzzy system is in the form of

$$R_l: \text{IF}(x \text{ is } A_l), \text{ THEN}(y = f_l(x)) \quad (3)$$

with linear consequent parts $f_l(x) = \beta_1^l x + \beta_2^l$, where β_1^l and β_2^l are constant coefficients.

Replacing the rule base (2) with (3) in the fuzzy system results in a first order TS type fuzzy system, which means the consequent part of each rule is a first order polynomial function of its inputs. If for simplicity the triangular membership functions are used, the mapping defined by this first order TS type fuzzy system becomes

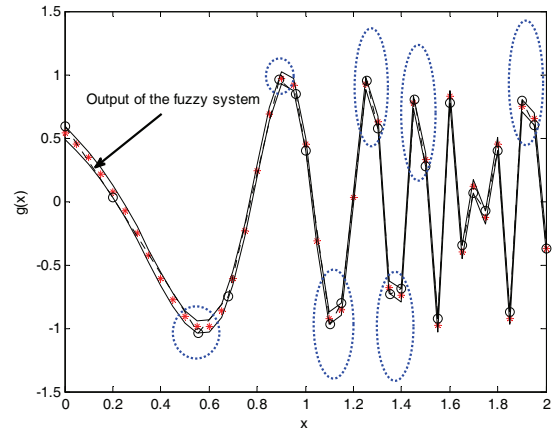


Fig. 3. Result by the tunnel algorithm in Example 2.

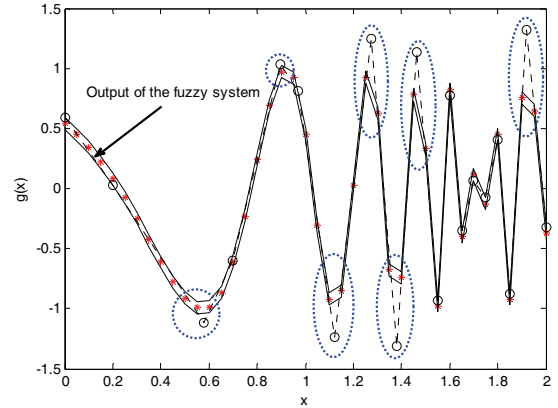


Fig. 4. Result by the improved tunnel algorithm in Example 2.

$$f(x) = \frac{\sum_{l=1}^M (f_l(x) \mu_{A_l}(x))}{\sum_{l=1}^M \mu_{A_l}(x)}. \quad (4)$$

Theorem 1: The TS type fuzzy system in (4) draws piecewise parabolas, and the number of rules used in the fuzzy system is equal to the number of the segments plus one.

Proof: Because of the localization property of the triangular membership function, among all the membership values $\mu_{A_l}(x)$ for any x , at most two but at least one value will be nonzero while the others are zero. Therefore, the input domain can be decomposed as $D = D^1 \cup \dots \cup D^{M-1}$, where $D = [x_{\min}, x_{\max}]$ and $D^l = [d_l, d_{l+1}]$, $l = 1, 2, \dots, M-1$. For any $x \in D^l$, it is known that $\mu_{A_l}(x) + \mu_{A_{l+1}}(x) = 1$, so the fuzzy system (4) in D^l can be simplified as

$$\begin{aligned} f(x) &= \sum_{j=1}^M f_j(x) \mu_{A_j}(x) = \sum_{j=l}^{l+1} f_j(x) \mu_{A_j}(x) \\ &= \left(1 - \frac{x - d_l}{\Delta d_l}\right) (\beta_1^l x + \beta_2^l) + \frac{x - d_l}{\Delta d_l} (\beta_1^{l+1} x + \beta_2^{l+1}) \\ &= \frac{\beta_1^{l+1} - \beta_1^l}{\Delta d_l} x^2 + \frac{\beta_1^l d_{l+1} - \beta_1^{l+1} d_l + \beta_2^{l+1} - \beta_2^l}{\Delta d_l} x + \frac{\beta_2^l d_{l+1} - \beta_2^{l+1} d_l}{\Delta d_l} \end{aligned} \quad (5)$$

Therefore, the fuzzy system over the whole domain D consists of $M-1$ pieces of parabolas. This proves the theorem. $\square$

The judging of visibility from geometry point of view for parabola curves is quite difficult. In the following, an algebraic approach is proposed to convert the edge-visibility problem into a Testing Point problem which can be easily extended to any order TS type fuzzy system. The following notations and theorems are necessary and straightforward.

Let the $(n-1)$ -th order polynomial function be $(n \geq 2)$

$$p(x) = a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_{n-1} x + a_n, \quad (6)$$

where $a_i, i = 1, \dots, n$, are the constant coefficients.

Theorem 2: To determine the coefficients of an $(n-1)$ order $(n \geq 2)$ polynomial curve characterized by $p(x)$, n distinct samples on this curve are required.

Proof: Substituting n distinct samples to the polynomial function gives n linear independent equations. Thus, the n coefficients $\{a_1, a_2, \dots, a_n\}$ can be computed by solving the linear equations. This proves Theorem 2. $\square$

Theorem 3: If a polynomial function $p(x)$ in (6) is specified by n distinct samples $(x_k, y'_k), k = 1, 2, \dots, n$, then $y = p(x_{n+1})$ that $p(x)$ can generate at the point x_{n+1} can be written as a weighted linear combination of $y'_k, k = 1, 2, \dots, n$, i.e. $y = c_1 y'_1 + c_2 y'_2 + \cdots + c_n y'_n$, where $c_i, i = 1, 2, \dots, n$, are constants which are determined by the samples.

Proof: In view of linear dependency, the conclusion is apparent. The following deduces an exact expression of the linear combination. Substituting $(x_k, y'_k), k = 1, 2, \dots, n$, to (6) results in n linear equations. Let $\mathbf{A} = [a_1, a_2, \dots, a_n]^T$,

$$\mathbf{Y} = [y'_1, y'_2, \dots, y'_n]^T \text{ and } \mathbf{X} = \begin{bmatrix} x_1^{n-1} & x_1^{n-2} & \cdots & x_1 & 1 \\ x_2^{n-1} & x_2^{n-2} & \cdots & x_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^{n-1} & x_n^{n-2} & \cdots & x_n & 1 \end{bmatrix}.$$

The linear equations can be written as

$$\mathbf{X}\mathbf{A} = \mathbf{Y}, \quad (7)$$

and $\mathbf{X}$ can be treated as a Vandermonde matrix with $\det(\mathbf{X}) = \prod_{1 \leq i < j \leq n} (x_i - x_j)$.

Since the n samples are different, i.e. $x_i \neq x_j$ when $i \neq j$, $\det(\mathbf{X}) \neq 0$. Hence, $\mathbf{X}$ is invertible.

Denote $\mathbf{IX} = \mathbf{X}^{-1}$, then matrix $\mathbf{A}$ can be obtained from $\mathbf{A} = \mathbf{IX} * \mathbf{Y}$ and its k -th element $(k = 1, 2, \dots, n)$ can be expressed as

$$a_k = \mathbf{IX}(k, 1) \cdot y'_1 + \mathbf{IX}(k, 2) \cdot y'_2 + \cdots + \mathbf{IX}(k, n) \cdot y'_n. \quad (8)$$

Therefore,

$$y = p(x_{n+1}) = a_1 x_{n+1}^{n-1} + a_2 x_{n+1}^{n-2} + \cdots + a_{n-1} x_{n+1} + a_n. \quad (9)$$

Substituting (8) to (9) and performing simplification gives

$$y = c_1 y'_1 + c_2 y'_2 + \cdots + c_n y'_n \quad (10)$$

where

$$c_k = \mathbf{IX}(1, k) \cdot x_{n+1}^{n-1} + \mathbf{IX}(2, k) \cdot x_{n+1}^{n-2} + \cdots + \mathbf{IX}(n-1, k) \cdot x_{n+1} + \mathbf{IX}(n, k), \quad k = 1, 2, \dots, n$$

are constants. $\square$

Following the edge-visibility strategy, notations $(p$ and $q)$ have the same meaning as the original tunnel algorithm, and the window $e_k, k \geq 2$ (e_1 is a line segment for initialization), which is a matrix consists with n different points, is the k -th polynomial curve found by the algorithm. Theorem 3 reveals the idea of the testing point algorithm. More specifically, equation (10) shows that the value $y = p(x_{n+1})$ at the point x_{n+1} is a weighted linear combination of y'_k 's, $k = 1, 2, \dots, n$, i.e. $y = c_1 y'_1 + c_2 y'_2 + \cdots + c_n y'_n$ where $c_i, i = 1, 2, \dots, n$ are constants already fixed by the n samples. When x_k is given and fixed, the only changeable values are the y'_k 's which are constrained by the error bands as $y'_k \in [g(x_k) - \varepsilon, g(x_k) + \varepsilon]$. Each combination of the n y'_k 's represents a normal polynomial curve, therefore all the combinations of y'_k 's form a family of curves.

Definition 1: The data point $p_j = (x_j, g(x_j))$ is *reachable* by a curve family means that its range, in which this curve family and the line $x = x_j$ intersect, has a non-empty intersection with the tolerance band $[g(x_j) - \varepsilon, g(x_j) + \varepsilon]$.

The above range of the curve family can be derived directly from (10), as equation (10) is apparently continuous and its extreme points can be easily computed based on the signs of c_k 's. For instance, if c_k 's are all positive, the upper limit of the range $y_j^{\max}$ can be calculated when all the y'_k 's are equal to their upper limits $g(x_k) + \varepsilon$ and the lower limit of the range $y_j^{\min}$ is calculated when all the y'_k 's are equal to their lower limits $g(x_k) - \varepsilon$. Because of the continuity in (10), the range can be written as $[y_j^{\min}, y_j^{\max}]$ whose limits are determined by only the limits of its variables y'_k 's. With the help of the above definitions, theorems and proofs, the following testing point algorithm is thus straightforward.

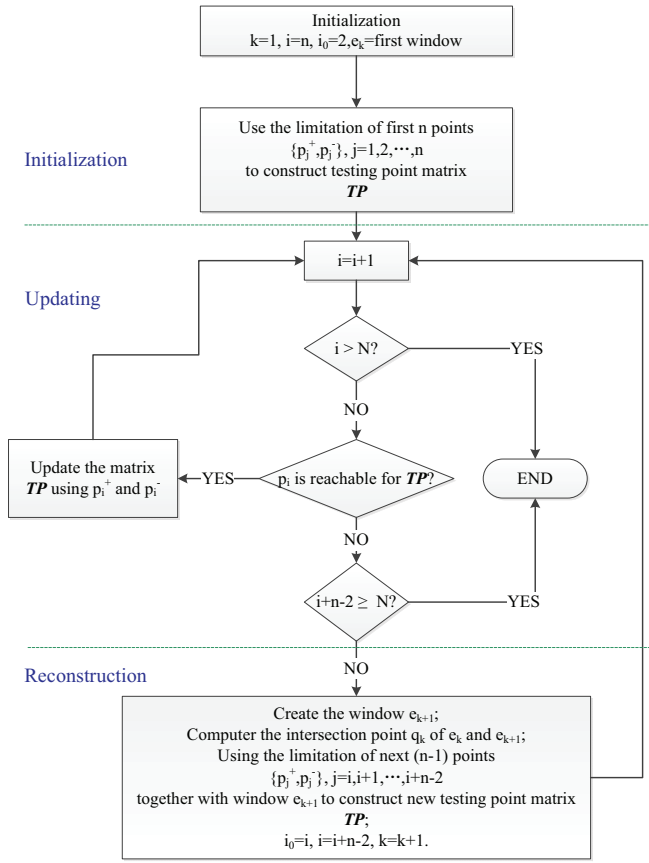


Fig. 5. Flowchart of the proposed testing point algorithm, where TP : the matrix stores the polynomials to be checked; k : the index of the created windows; i : the index of the sampling data; i_0 : the index of the first sampling data to be checked from the current window; e_k : the k -th window.

The Testing Point Algorithm (TPA)

Step 1: Initialization. Set $k = 1$, $i_0 = 2$, $i = n$, $e_k = \{p_1^+, p_1^-\}$, the array **isReachable** = $0^{N \times 1}$. Form the testing point matrix $TP \in R^{2^n \times n}$ which consists with all the combinations of the upper and lower limits of the first n points shown below.

$$TP = \begin{bmatrix} p_1^+ & p_2^+ & \cdots & p_n^+ \\ \vdots & \vdots & \vdots & \vdots \\ p_1^- & p_2^- & \cdots & p_n^- \end{bmatrix} \quad (11)$$

Step 2: Judging. $i = i + 1$. If $i > N$, go to Step 5. Otherwise, determine whether the point p_i is reachable or not. If p_i is reachable, set **isReachable**(i) = 1 and go to Step 3; otherwise, set **isReachable**(i) = 0 and go to Step 4.

Step 3: Updating. Since p_i is reachable, use $\{p_i^-, p_i^+\}$ to update TP through the following procedure.

Replacing all the elements in a column in TP with p_i^+ creates a submatrix. Therefore, the replacement of the n columns leads to n submatrices. The same number of submatrix is

obtained when using p_i^- for replacement. The updated $TP_{updated}$ are constructed by combining all of the submatrices and the original TP . As several rows may be the same in the $TP_{updated}$, it is necessary to remove the redundant rows from $TP_{updated}$. Each row of $TP_{updated}$ represents a normal polynomial curve, and all of the polynomial curves are checked one by one to see whether the points $p_{i_0}, p_{i_0+1}, \dots, p_i$ are all reachable from the selected polynomial curve. If not, the corresponding row in the $TP_{updated}$ is removed. After this procedure, set $TP = TP_{updated}$ and finish the updating step.

Step 4: Reconstruction. This step handles the TP construction when the judging step finds immediately p_i not reachable. It needs $(n-1)$ new points together with the window e_{k+1} to reconstruct the TP for testing on the next samples. If $i + n - 2 \geq N$ which means the remaining data points are just enough to form one polynomial function, go to Step 5 to finish. Otherwise, reconstruct the testing point matrix TP .

The reconstruction is very similar to Step 1. The difference is that the first point p_{fst} is the limit point from the area that the window e_{k+1} is able to cover, instead of the limit points of the line segment such as $\overline{p_1^- p_1^+}$.

Select the polynomial function e_{k+1} which is the closest one to the point p_i from TP . Theorem 3 shows that TP consists with several limit points. Pick the point p_{fst}^1 with the largest value in x -axis from e_{k+1} , then calculate the intersection point p_{fst}^{end} between e_{k+1} and the line $\overline{p_i^- p_i^+}$. The limit points of the reachable points **rP** from e_{k+1} in between the range $[p_{fst}^1, p_{fst}^{end}]$ are treated as the elements of p_{fst} . Then, the upper and lower limits of next $(n-1)$ points together with p_{fst} are used to construct the new TP in the form of (11). There may be some illegal rows constructed from which the polynomial function violates the accuracy requirement. The TP is therefore checked, by testing whether the points in **rP** are all reachable from the polynomial under checking or e_{k+1} . If not reachable, the row is considered as illegal and thus removed. This constructs a new TP for the test on the next samples.

q_k can be calculated as the intersection between e_{k+1} and e_k . Set $i_0 = i, i = i + n - 2, k = k + 1$.

Step 5: End. This is the ending process, where the last polynomial function e_{k+1} is found with its range $[q_k, q_{k+1}]$ for the fuzzy system.

If $i > N$ (jumped from Step 2), which means the last point p_N is reachable from the current TP , then find a representative one e_{k+1} from the TP which reaches p_N . A

typical choice is the one who is the closest to p_N . Then, calculate the q_k where e_k and e_{k+1} intersect. The final point q_{k+1} is the intersection between e_{k+1} and the line segment $\overline{p_N^+ p_N^-}$.

If $i \leq N$ (jumped from Step 4), which means the point p_i is not reachable but p_{i-1} is reachable from the current **TP** and the number of remaining un-judged sample points is equal or less than n , then we need to construct the e_{k+1} as $e_{k+1} = \{q_k, p_i, p_{i+1}, \dots, p_N\}$ where q_k is the intersection between e_k and the line segment $\overline{p_{i-1}^+ p_{i-1}^-}$. If the elements of e_{k+1} are not enough to determine a polynomial function, the designer can arbitrarily add some points into it (this will not affect the fuzzy rule number). A typical suggestion is to select the midpoints of line segments $\overline{p_{i+j} p_{i+j+1}}, j = 0, 1, \dots, N-i-1$.

Step 6: This step determines the minimum rule number and computes the parameters in the TS type fuzzy system. From the above five steps, q_j are used to store the partitions of the fuzzy sets and e_{j+1} to store the outputs in the range $x \in [q_j, q_{j+1}]$, $j = 1, 2, \dots, k$. For $(n-1)$ -th order TS type fuzzy system, each e_{j+1} contains $(n+1)$ points, while the response within this area is interacted by the two rules j and $(j+1)$ each of which has n parameters. In total, there are $2n$ parameters to be computed. Obviously, when $2n > (n+1)$, i.e. $n \geq 2$, the solution is not unique. The reason is that the parameters $\{\beta_i^l\}, i = 1, 2, \dots, n$ of the first rule can be only evaluated through the first area $[q_1, q_2]$. Since the degree of freedom is more than the number of equations, the solution can only show a relationship between $\{\beta_i^l\}$ and $\{\beta_i^2\}$ (Subscript “*” represents an index in $[1, 2, \dots, n]$), the solution in $[q_2, q_3]$ gives a relationship between $\{\beta_i^2\}$ and $\{\beta_i^3\}$, and similarly in all areas $[q_j, q_{j+1}]$, $j = 1, 2, \dots, k$. The sets of parameters are nesty related: if one is specified, all of the rest can be determined uniquely. A suggested choice for simplicity is to let the first rule be

$$\text{IF}(x \text{ is } A_1), \text{ THEN}(y = y(q_1)), \quad (12)$$

and then all the parameters can be determined.

The flowchart of the proposed testing point algorithm for $(n-1)$ -th order TS type fuzzy system is given in Fig. 5.

Remark 2: No matter what parameters of the final fuzzy system are determined through the above algorithm, the number of the rules can be guaranteed to be the minimum, which is of the most importance and the target of this research.

Remark 3: In most applications, TS type fuzzy systems with first order consequent parts are used, although less fuzzy rules may be needed if the order of the consequences is

increased, because higher order interpolation may increase the computational complexity and cause the so-called Runge’s phenomena (artificial oscillations with amplitude increasing with the degree of the interpolating polynomial) [17] which are not preferred. Nevertheless, in some special cases, to model highly nonlinear systems for instance, higher order TS type fuzzy system may be adopted, for which the solution to the design (based on maximal absolute error and minimum number of rules) is also provided by the proposed testing point algorithm.

III. SIMULATION RESULTS

The testing point algorithm works for the $(n-1)$ -th order TS type fuzzy system in modeling and identification. In the following, we just demonstrate the algorithm for modeling with the first order TS type fuzzy system, suggested by Remark 3.

Example 3 [17]: Consider the same problem in Example 2, the process to be modeled is characterized by $g(x) = \cos((e^x)^2)$. Fig. 6 gives the final result and Fig. 7 shows the partitions of the input domain and the membership functions obtained from q_k ’s.

Eventually 12 fuzzy sets $\{A_1, A_2, \dots, A_{12}\}$ are found centered at:

$$\{\alpha_1, \alpha_2, \dots, \alpha_{12}\} = \{0, 0.549, 0.876, 1.084, 1.237, 1.380, 1.543, 1.650, 1.750, 1.850, 1.950, 2.000\} \quad (13)$$

The l -th rule ($l = 1, 2, \dots, 12$) of the first order TS type fuzzy system is in the form of

$$R_l: \text{IF}(x \text{ is } A_l), \text{ THEN}(y \text{ is } \beta_1^l x + \beta_2^l), \quad (14)$$

where β_1^l and β_2^l are constants.

In the partitions $[\alpha_l, \alpha_{l+1}], l = 1, 2, \dots, k$, the output of the fuzzy system can be derived from (6), which can be re-written as

$$f_l(x) = \frac{\beta_1^{l+1} - \beta_1^l}{\alpha_{l+1} - \alpha_l} x^2 + \frac{\beta_1^l \alpha_{l+1} - \beta_1^{l+1} \alpha_l + \beta_2^{l+1} - \beta_2^l}{\alpha_{l+1} - \alpha_l} x + \frac{\beta_2^l \alpha_{l+1} - \beta_2^{l+1} \alpha_l}{\alpha_{l+1} - \alpha_l}. \quad (15)$$

Substituting all the points stored in $e_{l+1}, l = 1, 2, \dots, 11$ to (14) with corresponding l generates in total 12 linear equations. According to (12), let $\beta_1^1 = 0, \beta_2^1 = y(q_1)$. It is easy to solve the equations and finally obtain all the remaining parameters $\{\beta_1^l, \beta_2^l\}, l = 2, 3, \dots, 12$. The 12 fuzzy rules can then be concluded as

$$R_l: \text{IF}(x \text{ is } A_l), \text{ THEN}(y = 0 \cdot x + 0.5903)$$

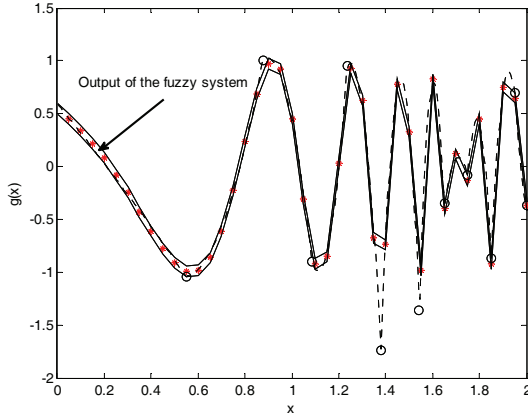


Fig. 6. First order TS type fuzzy system constructed in Example 3.

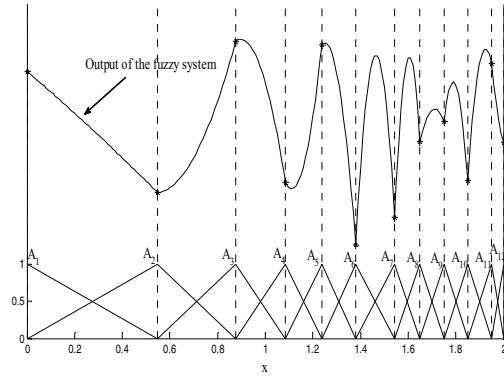


Fig. 7. The partitions of the input domain and the output of the fuzzy system in Example 3.

- R_2 : IF (x is A_2), THEN ($y = -0.2607x - 0.8981$)
 R_3 : IF (x is A_3), THEN ($y = 5.5178x - 3.8344$)
 R_4 : IF (x is A_4), THEN ($y = -5.8993x + 5.4988$)
 R_5 : IF (x is A_5), THEN ($y = 12.7912x - 14.8718$)
 R_6 : IF (x is A_6), THEN ($y = -10.0108x + 12.0781$)
 R_7 : IF (x is A_7), THEN ($y = -67.8593x + 103.3699$)
 R_8 : IF (x is A_8), THEN ($y = -128.1263x + 211.0606$)
 R_9 : IF (x is A_9), THEN ($y = -139.3971x + 243.8665$)
 R_{10} : IF (x is A_{10}), THEN ($y = -174.5493x + 322.0426$)
 R_{11} : IF (x is A_{11}), THEN ($y = -206.1204x + 402.6349$)
 R_{12} : IF (x is A_{12}), THEN ($y = -195.4124x + 390.4542$)

where $\{A_1, A_2, \dots, A_{12}\}$ are shown in Fig. 7 with centers in (13). As shown in Fig. 6, for the training data, the maximal

absolute error is less than $\varepsilon = 0.05$. Table 1 concludes the results from different algorithms for the same modeling problem and clearly shows the improvement of the proposed testing point algorithm.

Table 1: Numbers of Extracted Rules by Different Algorithms in Example 3

	Mamdani type Fuzzy System		TS type Fuzzy System
	Tunnel Algorithm	Testing Point Algorithm	Testing Point Algorithm
Number of Rules	25	19	12

IV. CONCLUDING REMARKS

An improved algorithm based on the geometric approach in [17] was proposed to build any order TS type fuzzy model with minimum number of rules and guaranteed approximation accuracy, from the input-output data of an unknown nonlinear system. Comparison study through numerical simulations demonstrates the effectiveness of the method.

This proposed testing point algorithm in its current format works only for SISO cases. Extension to higher dimension, i.e. MIMO (multiple-input-multiple-output) cases will greatly expand the applicability of the strategy and is a major concern for the future work.

REFERENCES

- [1] S. Aja-Fernandez and C. Alberola-Lopez, "Matrix modeling of hierarchical fuzzy systems," IEEE Transactions on Fuzzy Systems, vol. 16, no. 3, pp. 585-599, 2008.
- [2] R. Babuška, Fuzzy Modeling for Control. Kluwer Academic Publishers, Boston, 1998.
- [3] P. Baranyi, A. R. Varkonyi-Koczy, Y. Yam, and R. J. Patton, "Adaptation of TS fuzzy models without complexity expansion: HOSVD-based approach," IEEE Transactions on Instrumentation and Measurement, vol. 54, no. 1, pp. 52-60, 2005.
- [4] P. Baranyi, Y. Yam, A. R. Varkonyi-Koczy, R. J. Patton, P. Michelberger, and M. Sugiyama, "SVD-based complexity reduction to TS fuzzy models," IEEE Transactions on Industrial Electronics, vol. 49, no. 2, pp. 433-443, 2002.
- [5] Y. S. Ding, H. Ying, and S. H. Shao, "Necessary conditions on minimal system configuration for general MISO Mamdani fuzzy systems as universal approximators," IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 33, no. 6, pp. 857-864, 2000.
- [6] K. Cpalka., "A new method for design and reduction of neuro-fuzzy classification systems," IEEE Transactions on Neural Networks, vol. 20, no. 4, pp. 701-714, 2009.
- [7] A. Gegov, Complexity Management in Fuzzy Systems: a Rule Base Compression Approach, Springer, Berlin Heidelberg, 2007.
- [8] M. K. Güven and K. M. Passino, "Avoiding exponential parameter growth in fuzzy systems," IEEE Transactions on Fuzzy Systems, vol. 9, no. 1, pp. 194-199, 2001.
- [9] H. Ishibuchi and T. Yamamoto, "Fuzzy rule selection by multi objective genetic local search algorithms and rule evaluation measures in data mining," Fuzzy Sets Systems, vol. 141, pp. 59-88, 2004.

- [10] Y. C. Jin, "Fuzzy modeling of high-dimensional systems: Complexity reduction and interpretability improvement," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 2, pp. 212-221, 2000.
- [11] M. G. Joo and T. Sudkamp, "A method of converting a fuzzy system to a two-layered hierarchical fuzzy system and its run-time efficiency," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 1, pp. 93-103, 2009.
- [12] A. Riida and E. Rüsternb, "Identification of transparent, compact, accurate and reliable linguistic fuzzy models," *Information Sciences*, in press.
- [13] M. Setnes, R. Babuška, U. Kaymak, and H. R. van Nauta Lemke, "Similarity measures in fuzzy rule base simplification," *IEEE Transactions on System, Man, and Cybernetics B*, vol. 28, no. 3, pp. 376-386, 1998.
- [14] M. Setnes and R. Babuška, "Rule base reduction: some comments on the use of orthogonal transforms," *IEEE Transactions on System, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 31, no. 2, pp. 199-206, 2001.
- [15] T. Sudkamp, A. Knapp, and J. Knapp, "Model generation by domain refinement and rule reduction," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 33, no. 1, pp. 45-55, 2003.
- [16] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, no. 1, pp. 116-132, 1985.
- [17] F. Wan, H. L. Shang, L. X. Wang, and Y. X. Sun, "How to determine the minimum number of fuzzy rules to achieve given accuracy: a computational geometric approach to SISO case," *Fuzzy Sets and Systems*, vol. 150, no. 2, pp. 199-209, 2005.
- [18] L. X. Wang, "Fuzzy systems are universal approximators," *Proc. IEEE International Conf. on Fuzzy Systems*, San Diego, pp. 1163-1170, 1992.
- [19] L. X. Wang, "Analysis and design of hierarchical fuzzy systems," *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 5, pp. 617-624, 1999.
- [20] Y. Yam, P. Baranyi, and C. T. Yang, "Reduction of fuzzy rule base via singular value decomposition," *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 2, pp. 120-132, 1999.
- [21] J. Yen and L. Wang, "Simplifying fuzzy rule-based models using orthogonal transformation methods," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 29, no. 1, pp. 13-24, 1999.
- [22] J. Yen, L. Wang, and C. W. Gillespie, "Improving the interpretability of TSK fuzzy models by combining global learning and local learning," *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 4, pp. 530-537, 1998.
- [23] H. Ying, "Sufficient conditions on general fuzzy systems as function approximations," *Automatica*, vol. 30, no. 3, pp. 521-525, 1994.
- [24] H. Ying, "Sufficient conditions on uniform approximation of multivariate functions by general Takagi-Sugeno fuzzy systems with linear rule consequent," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 28, no. 4, pp. 515-520, 1998.
- [25] H. Ying, *Fuzzy Control and Modeling: Analytical Foundations and Applications*, IEEE Press, New York, 2000.