Qualitative Modeling and Controller Design using Dynamic Fuzzy Systems

Klaus Schmid, Volker Krebs
Universität Karlsruhe (TH), Institut für Regelungs- und Steuerungssysteme
Kaiserstraße 12, D-76131 Karlsruhe Germany
Phone: +49-721-6083179, Fax: +49-721-6082707
email:{schmid,krebs}@irs.etec.uni-karlsruhe.de

ABSTRACT: Qualitative modeling may be applied when knowledge about a system is only available in linguistic form. The knowledge might be processed by a dynamic fuzzy system consisting of a rule base and an inference method modeling human reasoning. Conventional fuzzy inference methods do not consider this association to human reasoning and therefore are not suitable for the dynamic processing of linguistic knowledge. Inference has to provide both, quantitative and qualitative information about the model output. In this paper a new inference method based on the concept of interpolating rules is presented. It results in a mapping of fuzzy inputs onto fuzzy outputs with a feedback to the input, considering in particular the informational content of the fuzzy sets. Thus, a dynamic process model is built from given linguistic knowledge in form of rules. Finally, a controller design method is presented based on the qualitative process model.

KEYWORDS: qualitative reasoning, dynamic fuzzy system, inference with interpolating rules, interpolation, model-based control, inverse model control, non-linear control

1 INTRODUCTION

A qualitative model of a process may often be given in form of a rule base describing the process behavior. The rule base contains the information that relates input conditions to output responses. The rules are production rules

'If condition Then conclusion.'

A fuzzy inference method provides the mathematical means to evaluate these rules. In order to provide a formal framework for representing qualitative information, linguistic variables and values are introduced. A rule's condition and conclusion are formulated as logical operations on these variables and values. The linguistic values are represented by fuzzy sets.

The rule base consists of all rules necessary to describe the process behavior. According to this rule base the fuzzy inference method maps fuzzy input values onto fuzzy output values. Rule base and inference method represent the fuzzy system.

This contribution is based on dynamic fuzzy systems introduced by Schäfers (1997). A fuzzy system is called *dynamic fuzzy system*, if the system's fuzzy output is fed back and used as input for subsequent inference steps. Since the inference method models the human way of approximate reasoning, the output is fed back without defuzzification.

Due to the fact that the fuzzy sets characterize linguistic values, they actually have to be interpretable as such values. Therefore, their membership functions must not be of arbitrary shape. In this contribution, only fuzzy sets with triangular membership functions are considered. They are simply defined by three parameters and are easily interpretable as linguistic values. Since the system's fuzzy output may be an input value in subsequent inference steps, the inference method has to ensure that the system's output fuzzy set is interpretable, as well. Hence, the inference method is a mapping of triangular fuzzy inputs onto triangular fuzzy outputs.

An important feature of dynamic fuzzy systems is the processing of the informational content assigned to fuzzy sets. A decrease in an input's informational content must not result in an increase in the output's informational content. Therefore, it is important that the inference method is able to process the fuzziness of the input values.

A class of inference methods suitable for dynamic fuzzy systems is the *inference with interpolating rules*. In section 2 of this paper a new modified inference method is developed using this concept of interpolating rules. The method provides the means to evaluate qualitative rule based models by interpolation functions.

Interpolation functions used by Schäfers (1997) result in an exponential growth of the number of rules with the number of inputs. To cope with this problem, a useful class of interpolation functions is outlined in section 3: Piecewise linear interpolation functions require less rules for system representation. The methods presented in this paper are adopted from computational geometry.

A further advantage of the piecewise linear process model is based on the fact that it can easily be inverted. Inversion is necessary to design a controller based on the given qualitative knowledge. Model inversion is outlined in section 4 and a new controller design method is derived. Finally, in section 5 an application example will demonstrate the efficiency of the presented method.

2 INFERENCE METHOD

Conventional fuzzy inference methods such as the max-min inference (Driankov, 1993) do not compute output fuzzy sets with membership functions of a simple, interpretable shape. As a consequence, the output may only be fed back after defuzzification which results in a loss of the informational content assigned to the fuzzy sets. Therefore, in this section a new inference method will be developed based on the concept of interpolating rules.

A simple example is used to explain the inference with interpolating rules. Consider a system with one input variable **E** and one output variable **Y**. The behavior of the system may be given by a rule base consisting of two rules:

R_1: If **E** = *A* Then **Y** = K_A, (1)

R_2: If **E** = *B* Then **Y** = K_B. (2)

In contrast to conventional nomenclature, in this paper not the conditions of the rules but the fuzzy sets *A* and *B* are called *premises*. The fuzzy sets K_A and K_B are called *conclusions*. The triangular membership functions of the fuzzy sets *A* and *B*, defined over the input domain **e** of the linguistic variable **E** , as well as the membership functions of the output fuzzy sets K_A and K_B are depicted in Figure 1.

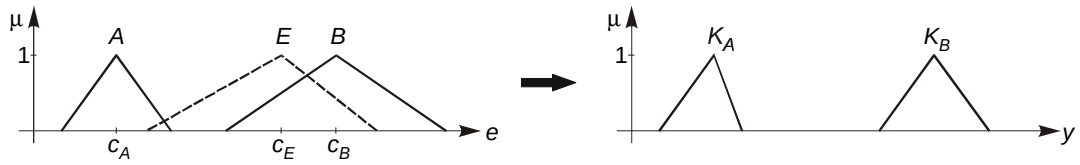

Figure 1 : Premises *A* and *B*, conclusions K_A and K_B , and input *E* .

The inference method has to calculate the output fuzzy set *Y* for the given input fuzzy set *E* by evaluating the rules describing the system behavior. If the inference method is used in a dynamic fuzzy system, it has to satisfy several conditions resulting from the connection to human reasoning, Krebs (1998). For the above example, a basic requirement is that output *Y* has to match K_A or K_B exactly, if input *E* matches the premises *A* or *B*, respectively.

If the input value lies *between* the given premises (see Figure 1), the given rules cannot be evaluated directly. Yet it is clear that output *Y* then should lie between the given conclusions. Furthermore, since input *E* is fuzzier than the premises, i.e. it contains less information, the output should be fuzzier than the conclusions, as well.

Inference with interpolating rules is performed in two consecutive steps. First, an interpolating rule is generated in order to determine the position of the output value. After that, the fuzziness of the input is mapped onto the output.

2.1 INTERPOLATING RULE

In the framework of dynamic fuzzy systems the value of the input domain belonging to the peak value of a triangular membership function is called *center* of the membership function. The center determines the position of the fuzzy set on the input domain.

Premise and conclusion of the interpolating rule only depend on the center of the input fuzzy set, but not on the shape of its membership function. Therefore, they can be calculated by interpolation with the center c_E of the input fuzzy set *E* as independent variable. The centers of the interpolating premise and the input *E* have to be identical.

The interpolating rule for the above SISO-system is then given as

IR: If **E** = *IP* Then **Y** = *IC*, (3)

where *IP* and *IC* denote interpolating premise and interpolating conclusion, respectively. If the input center matches the center of one premise the interpolating rule has to be equivalent to the corresponding given rule. Hence, the centers of the original premises serve as interpolation nodes with the given premises and conclusions as interpolation values.

Using fuzzy sets as function values at the interpolation nodes allows the definition of interpolation of triangular fuzzy sets. A triangular fuzzy set *M* is defined by three parameters, its center c_M, the left foot ℓ_M, and the right foot r_M. Plotting these parameters versus the center of the input variable and applying linear interpolation between the interpolation nodes (i.e. the centers of the premises *A* and *B*) leads to the representation shown in Figure 2.

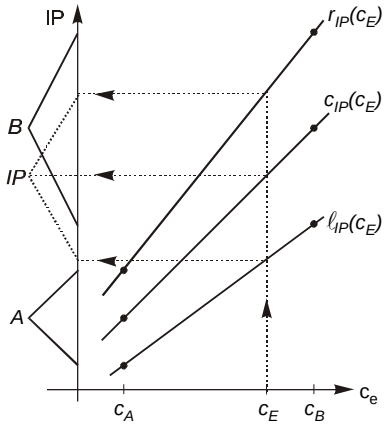

Figure 2 : Interpolation functions for the calculation of *IP*, with the centers of the given premises *A* and *B* as interpolation nodes.

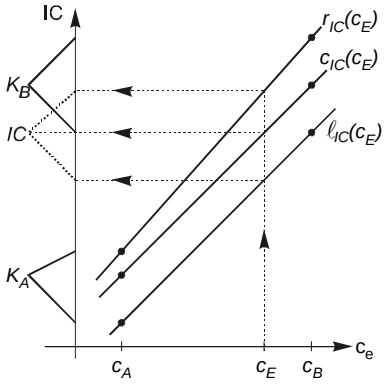

Figure 3 : Interpolation functions for the calculation of *IC*, with the centers of the given conclusions K_A and K_B as interpolation nodes.

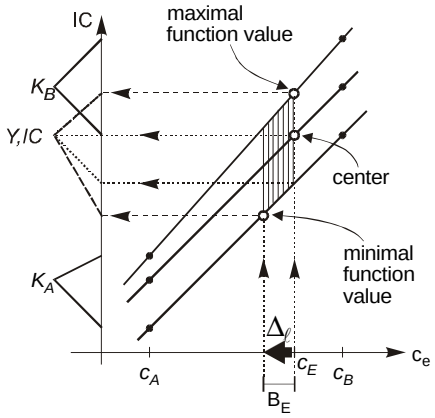

Figure 4 : Calculation of *Y* using interpolation functions of Figure 3. Maximal and minimal function value are determined in the hatched area.

In order to obtain the triangular fuzzy set representing *IP*, three interpolation functions are necessary: $\ell_{IP}(c_E)$, $c_{IP}(c_E)$, and $r_{IP}(c_E)$ for the calculation of the left foot, the center, and the right foot, respectively. Note, that the interpolation function for the center c_{IP} is the bisector of the first quadrant, and in turn the centers of *IP* and *E* are identical. If, for example, the input center c_E is equal to the center c_A of the given premise *A*, the interpolating premise is equivalent to premise *A*. If the center c_A lies between the centers of premises *A* and *B* (as illustrated in Figure 2), the interpolating premise is calculated by the interpolation functions.

The interpolating conclusion *IC* is calculated in the same way. The interpolation functions are determined by the *conclusions* of the two given rules (see Figure 3). If the center of the input variable is equal to the center of premise *A*, conclusion K_A results as interpolating conclusion. If the center lies between the centers of the given premises (as depicted in Figure 3) the interpolating conclusion *IC* is determined by the interpolation functions.

Interpolation functions for interpolating premise and conclusion need not be linear. They only have to be continuous functions and must not intersect in order to obtain interpretable fuzzy sets. Intersecting interpolation functions could, for example, result in a left foot of *IP* located right of c_{IP}.

2.2 MAPPING THE FUZZINESS

The interpolating rule (3) determines the position of the output *Y*. The fuzziness of *Y* has to be computed in the second inference step. Since the rules describe the only knowledge about the behavior of the system, the output has to be at least as fuzzy as the interpolating conclusion. Interpolating premise and conclusion for the example mentioned above are depicted in Figure 5.

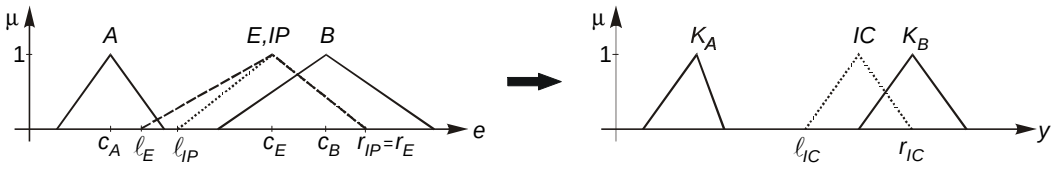

Figure 5 : Input value *E*, interpolating premise *IP*, and interpolating conclusion *IC*.

The actual input *E* is fuzzier than the interpolating premise *IP*. The fuzziness of an interpretable fuzzy set can be considered as its informational content. The area below the fuzzy set's membership function is used as a measure of informational content: the larger the area the smaller the informational content.

Applied to Figure 5, this means that input *E* contains less information than the interpolating premise *IP*. Therefore, it is quite clear that the output value *Y* should contain less information than the interpolating conclusion *IC*, as well. Hence, it follows that *Y* should be fuzzier than *IC*. In order to determine the actual output *Y*, it is necessary to compute direction and degree of its fuzziness relatively to the interpolating conclusion *IC*. As depicted in Figure 5, the left foot ℓ_E of the input is fuzzier than the left foot ℓ_{IP} of the interpolating premise. Considering membership functions as possibility distributions, this means that input values smaller than ℓ_{IP} are *possible*. Therefore, for the given rule base, output values smaller than ℓ_{IC} have to be possible, as well.

This mapping of fuzziness can be carried out by calculating the interpolation functions for the interpolating conclusion not only at the single point c_E, but over a certain region around that point. For the given example this region has to lie *left* of c_E, because the input *E* is fuzzier than the interpolating premise on the *left* side. A possible region is shown in Figure 4.

The center of the output variable only depends on the center of the input and is therefore not changed by the mapping of fuzziness. It is always equal to the center of the interpolating conclusion.

The maximal possible value of a fuzzy set is described by its right foot. Therefore, the right foot r_Y is determined as the maximal value of all three interpolation functions over the region B_E. The left foot ℓ_Y is determined by the minimal function value. Applied to input *E* of Figure 5, this mapping of fuzziness results in additionally possible values left of the interpolating conclusion. The resulting output fuzzy set *Y* is plotted in Figure 4 with dashed membership function.

One remaining degree of freedom of this mapping is the absolute value one extends the considered input domain. Denoting the extension on the left and right side by Δ_ℓ and Δ_r, respectively, B_E in Figure 4 is defined as

$$B_E = [c_E - \Delta_\ell, c_E + \Delta_r].$$

Δ_ℓ is depicted in Figure 4. For the given example, Δ_r is zero since the input fuzzy set's right side is not fuzzier than the interpolating premise. A good choice for the amount of extension may be the difference between left, respectively right foot points of interpolating premise and input fuzzy set:

$$\Delta_\ell = \begin{cases} \ell_{IP} - \ell_E & \text{for } \ell_E < \ell_{IP} \\ 0 & \text{otherwise} \end{cases},$$

$$\Delta_r = \begin{cases} r_E - r_{IP} & \text{for } r_E > r_{IP} \\ 0 & \text{otherwise} \end{cases}.$$

Whether an extension of the considered input domain to the left or right side affects the left or right foot of the output value depends on the interpolation nodes, i.e. the given premises and conclusions. If the interpolation functions in Figure 4 had negative gradients, extension Δ_ℓ to the left side would affect the *right* foot of the output. Furthermore, both left and right foot of the output can be influenced by Δ_ℓ for specific shapes of the interpolation functions.

As mentioned before, the output's informational content is bounded by the interpolating conclusion. If one foot of the input fuzzy set is *less* fuzzy than the respective foot of the interpolating premise, it has no affect on the output. This situation is depicted in Figure 6, where the input value lies at the same position as the interpolating premise, but is less fuzzy.

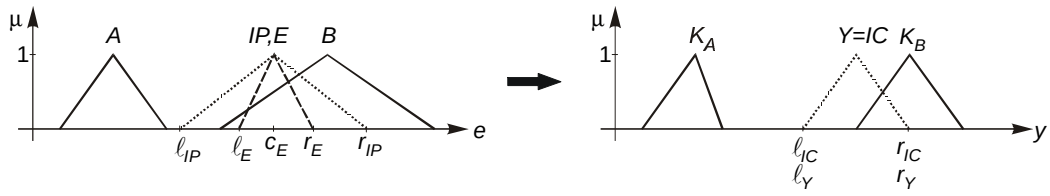

Figure 6 : If the input fuzzy set *E* is contained in the interpolating premise *IP*, the output *Y* is equivalent to the interpolating conclusion *IC*.

Again, interpolating premise and conclusion are calculated according to Figure 2 and Figure 3. Since they only depend on the center of the input value, they are equivalent to *IP* and *IC* of Figure 5. For the given fuzzy sets of Figure

6 both, Δ_ℓ and Δ_r are equal to zero. Therefore, the output fuzzy set is identical to the interpolating conclusion; the informational contents of output *Y* and interpolating conclusion *IC* are equivalent.

Extension of the presented inference method to systems with multiple inputs is simply done by defining the interpolation functions over a multidimensional input domain. For a system with n input variables, $(n+1)$ interpolations are necessary to determine n interpolating premises and the interpolating conclusion. Then, an interval B_{Ei} is calculated separately for each input variable. Finally, the resulting domain B_E is determined as the Cartesian product of n intervals, $B_E = B_{E1} \times \cdots \times B_{En}$.

3 PIECEWISE LINEAR INTERPOLATION

In the sequel, two of the three interpolation functions will be omitted. This increases the clarity of the presentation, since a special type of interpolation function is presented. Application to interpolation of triangular fuzzy sets in dynamic fuzzy systems can be performed by simply adding the interpolation functions for the additional two parameters, as done before.

In the previous section, the inference with interpolating rules was presented considering only two rules. More than two rules can be evaluated by piecewise linear interpolation. Multidimensional interpolation functions are necessary when considering MISO-systems. As mentioned before, interpolation functions may be of almost arbitrary shape. However, due to calculational simplicity, it is reasonable to apply interpolation functions as simple as possible.

The shape of interpolation function depends on the distribution of the interpolation nodes in the input domain. A lattice-like structure of the rule base leads to piecewise multilinear interpolation. Lattice-like node structure provides easy determination of the active interpolation region. However, a drawback of this structure is the exponential growth of the number of rules with the number of inputs.

Arbitrarily distributed ("scattered") interpolation nodes are employed by Berger (1997), Ulrich (1996) and Schmid (1998), applying piecewise linear interpolation. The number of rules can be kept small, and model inversion can be performed easily. In this section, piecewise linear interpolation will be outlined. Model inversion will be presented in section 4.

A first order non-linear dynamic system $x(k+1) = f\big(x(k), u(k)\big)$ is considered, its dynamic behavior being described by four rules:

R_1: If $x(k) = c_{11}$ And $u(k) = c_{21}$ Then $x(k+1) = f_1$,

$\vdots$

R_4: If $x(k) = c_{12}$ And $u(k) = c_{22}$ Then $x(k+1) = f_4$.

$x(k)$ is the state variable, $u(k)$ the control variable; c_{hj} and f_i are crisp values. Figure 7 illustrates these rules in the input-output domain.

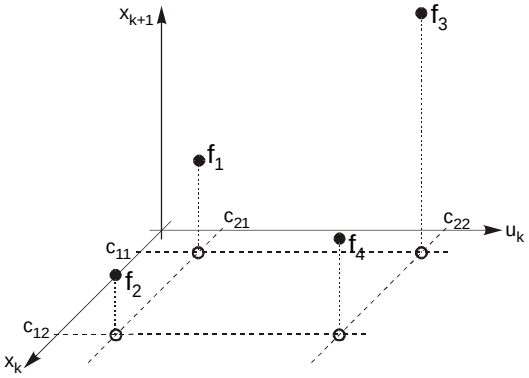

Figure 7 : Rules describing the system behavior.

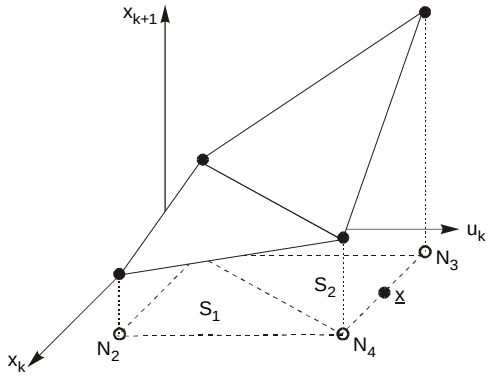

Figure 8 : Piecewise linear interpolation function.

In order to obtain a piecewise linear model of the system, regions have to be defined where the respective linear models are valid. For linear interpolation in an n-dimensional input space, $(n+1)$ interpolation nodes are necessary. In computational geometry a node is designated as *vertex,* a region as *simplex* (Preparata, 1985). Each simplex is then defined as the convex hull of $(n+1)$ vertices.

With given rules, the main task is to define the simplices, which can be done using tools of computational geometry. Ullrich (1996) describes the Delaunay triangulation, a mathematical concept constructing non-intersecting interpolation regions from a set of nodes. There are other triangulation methods, but the Delaunay triangulation exhibits several benefits when linear interpolation is applied (Ullrich, 1996). Triangulation subdivides the input space into a set of non-overlapping simplices. The piecewise linear model of a system is then defined by the function values at the vertices, given as conclusions of the rules.

One possible piecewise linear model for the described system is illustrated in Figure 8. The input space is partitioned into two simplices S_1 and S_2 defined by the vertices N_1, N_2, N_4 and N_1, N_4, N_3, respectively. In order to calculate the interpolated output for a given input vector $\underline{x} = [x_1,...,x_n]^T$, in a first step the simplex containing $\underline{x}$ has to be determined. For this purpose, Cartesian coordinates of the input are transformed to barycentric coordinates defined in relation to a specific simplex.

The nodes N_i are given as combinations of input values $\underline{x} = [x_{i,1},...,x_{i,n}]^T$ and corresponding output f_i. A simplex is defined by $(n+1)$ nodes, $\underline{x}_1,\ldots,\underline{x}_{n+1}$. The relationship between Cartesian and barycentric coordinates of input $\underline{x}$ in the considered simplex is then given by

$$\underline{x}^* = \begin{bmatrix} x_1 \\ \vdots \\ x_n \\ 1 \end{bmatrix} = \underline{V} \cdot \underline{b} = \begin{bmatrix} x_{1,1} & \cdots & x_{n,1} & x_{n+1,1} \\ \vdots & \ddots & \vdots & \vdots \\ x_{1,n} & \cdots & x_{n,n} & x_{n+1,n} \\ 1 & \cdots & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ \vdots \\ \vdots \\ b_{n+1} \end{bmatrix}, \tag{4}$$

where $\underline{x}^*$ is the input vector with an additional coordinate equal to one, $\underline{V}$ is a transformation matrix with the columns defined by the $(n+1)$ nodes of the simplex, and $\underline{b}$ is the vector of the input's barycentric coordinates. The last row of equation (4) normalizes the barycentric coordinates: $b_1 + \cdots + b_{n+1} = 1$.

Barycentric coordinates in relation to a specific simplex are non-negative if and only if this simplex contains the input vector. This fact enables determination of the simplex containing the input $\underline{x}$.

The interpolation value y for a given input is then calculated as the weighted sum of the function values in the vertices of the active simplex, with the barycentric coordinates as weighting factors. Combining the vertices' function values into a vector $\underline{f}^T = [f_1,\ldots, f_{n+1}]$ yields

$$y = \sum_{i=1}^{n+1} f_i \cdot b_i = \underline{f}^T \cdot \underline{b}.$$

With the inverse of the transformation matrix in (4) this equation can be rewritten as

$$y = \underline{f}^T \cdot \underline{V}^{-1} \cdot \underline{x}^*. \tag{5}$$

Thus,

$$y = a_0 + a_1 x_1 + \cdots + a_n x_n \tag{6}$$

describes a linear interpolation, with different coefficients a_i for each simplex, resulting in the piecewise structure of the interpolation function.

For example, with $c_{11} = c_{21} = 0$, $c_{12} = c_{22} = 1$, $f_1 = 1/2$, $f_2 = 1/4$, $f_3 = 3/2$, $f_4 = 1/2$ the transformation matrices for the two simplices shown in Figure 8 become

$$\underline{V}_1 = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \text{ and } \underline{V}_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

With equations (4) and (5) this yields the two interpolation functions

$$S_1 : \mathrm{x(k+1)} = 1/2 - 1/4\, x(k) + 1/4\, u(k), \tag{7}$$

$$S_2 : \mathrm{x(k+1)} = 1/2 - x(k) + u(k). \tag{8}$$

For given input vector $\underline{x}^* = [1/2 \quad 1 \quad 1]^T$ barycentric coordinates $\underline{b}_1$ and $\underline{b}_2$ in simplices S_1 and S_2, respectively, prove to be

$$\underline{b}_1 = [1/2 \quad -1/2 \quad 1]^T,\ \underline{b}_2 = [0 \quad 1/2 \quad 1/2]^T.$$

Since only coordinates $\underline{b}_2$ are non-negative, simplex S_2 is the active simplex containing the input vector, which can easily be seen by marking $\underline{x}$ in Figure 8.

4 MODEL INVERSION AND CONTROLLER DESIGN

In the previous two sections a new inference method was presented mapping triangular fuzzy inputs onto a triangular fuzzy output by means of piecewise linear interpolation. This method can be used to build a qualitative process model. Controller design based on this qualitative model will be considered in this section.

The controller will be piecewise linear and will be constructed by the interpolation functions of section 3, as well. Since the center of the model output only depends on the centers of the input values, it can only be affected by the center of the control variable. Therefore, it is not necessary to consider *fuzzy* control variables.

The center of the control variable for its turn is only dependent on the input centers of the controller, hence *fuzzy* inputs of the controller are not necessary either. As a consequence, the controller can be developed only considering the center equation of the model, resulting in a controller mapping crisp inputs onto a crisp output.

A linear model is given in each model simplex and therefore linear design methods could be used for controller design. However, the great number of different local models yields a problem, as will be made clear in the sequel. Consider again the above example of a process model. The two simplices of the interpolation function for the center of the model output are depicted in Figure 9.

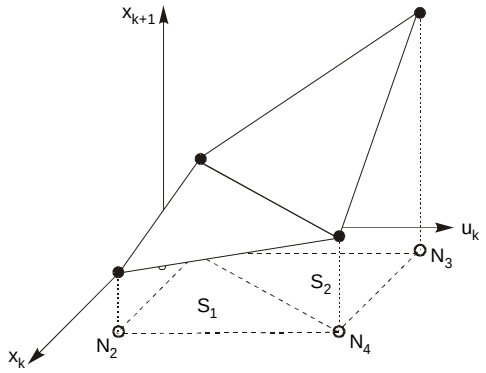

Figure 9 : Piecewise linear model.

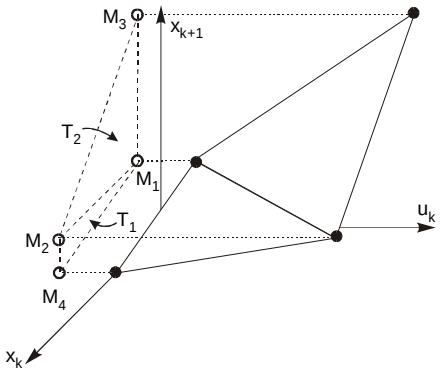

Figure 10 : The inverse of the piecewise linear model.

Each simplex represents a linear function. For example, the interpolation function of simplex S_2 may be given as

$$x(k+1) = a_0 + a_1 x(k) + a_2 u(k) .$$

One possible closed loop dynamics would be a system reaching a desired output value $w(k)$ within one time step, i.e.

$$w(k) = x(k+1) = a_0 + a_1 x(k) + a_2 u(k) . \qquad (9)$$

Solving this equation with respect to the control variable $u(k)$ yields the control law

$$u(k) = \big(w(k) - a_0 - a_1 x(k)\big)/a_2 .$$

However, this control law can not be applied, since it is not known in advance whether the resulting $u(k)$ actually leads to an input vector $\big(x(k),u(k)\big)$ lying inside the considered simplex S_2. Therefore, model inversion is necessary to determine the appropriate control $u(k)$ for given $w(k)$ and $x(k)$. Ullrich (1997) uses iterative numerical methods such as the Newton-Raphson method for non-linear root finding of piecewise linear models. Yet, this is not necessary since piecewise linear models can be inverted analytically as will be shown in the next section.

4.1 MODEL INVERSION

The model input domain is given by $\big(x(k),u(k)\big)$. Model simplices are defined by their interpolation nodes with corresponding function values, $N_i = (x_i, u_i, f_i)$, where x_i and u_i are input values and f_i are output values of the interpolation function $x(k+1) = f\big(x(k),u(k)\big)$.

To achieve the desired dynamics $w(k) = x(k+1)$, model inversion with respect to the control variable $u(k)$ is now performed by exchanging input and output variables. This means that the inverse model defines a mapping $u(k) = f_M\left(x(k), w(k)\right)$. The regions in the new input domain $\left(x(k), w(k)\right)$ are determined by the nodes $M_i = (x_i, f_i, u_i)$ depicted in Figure 10. As a consequence, both model and inverse model are defined as piecewise linear functions with simplicial input space partitioning. Note, that the M_i are formed by simply exchanging the last two coordinates of the N_i.

Due to the fact that the model describes a single-valued mapping, its input space partitioning consists of non-overlapping simplices. On the other hand, since the model needs *not* be a one-to-one mapping, input space partitioning of the inverse model may consist of overlapping simplices. However, this is not a problem, because it means that several control actions are applicable to achieve the same objective.

Figure 11 illustrates this for a simple system with only one input, being given as a non-reversible function $x(k+1) = f(u(k))$. The original model is piecewise linear on two non-overlapping intervals S_1 and S_2. Model inversion produces a multiple-valued mapping $u(k) = f_M(w(k))$ defined on two overlapping simplices T_1 and T_2. Nevertheless, the desired output $w*$ can be reached within one time step, applying either control value $u_{k,1}$ or $u_{k,2}$.

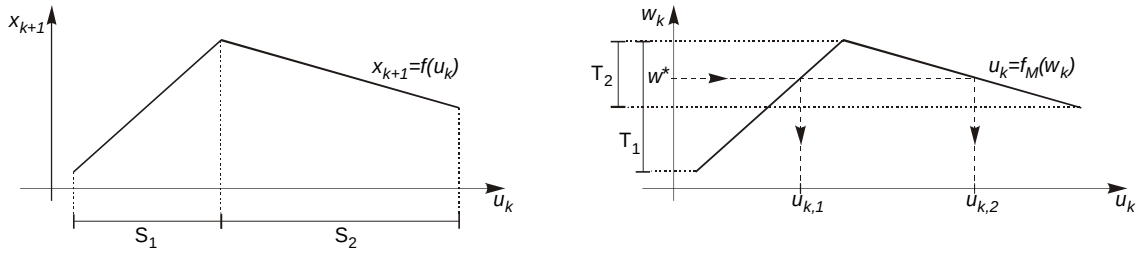

Figure 11 : Inversion of a non-monotonic mapping.

4.2 CONTROLLER DESIGN

In the previous section model inversion was used to design a simple controller obtaining a desired output in one time step. Furthermore, it is possible to modify the inversion such that the closed loop dynamics will follow a reference model. In each interpolation region of a piecewise linear model $y(k+1) = f\left(\underline{x}(k), u(k)\right)$ a linear difference equation

$$y(k+1) = a_0 + a_1 x_1(k) + \cdots + a_{n-1} x_{n-1}(k) + a_n u(k) \tag{10}$$

describes the dynamic behavior, with state variables $x_i(k)$ and control variable $u(k)$. The reference model is given as

$$y(k+1) = f_R\left(\underline{x}(k), w(k)\right) = b_0 + b_1 x_1(k) + \cdots + b_{n-1} x_{n-1}(k) + b_n w(k) \tag{11}$$

where $w(k)$ is the desired output. Equating (10) and (11) produces the control law

$$u(k) = \left((b_0 - a_0) + (b_1 - a_1) x_1(k) + \cdots + (b_{n-1} - a_{n-1}) x_{n-1}(k) + b_n w(k)\right) / a_n \ . \tag{12}$$

As in the previous section, the input variables of the controller correspond to the input variables of the model, with the exception that $u(k)$ is replaced by $w(k)$. Therefore, input partitioning of the controller's input space can again be done by mapping the model's simplices S_i defined by the interpolation nodes $N_i = (x_{1,i}, \ldots x_{n-1,i}, u_i, y_i)$. Interpolation nodes M_i of the controller are produced by exchanging control and output variable of the N_i. For a specific node N_i the corresponding node M_i is then given as $M_i = (x_{1,i}, \ldots x_{n-1,i}, w_i, u_i)$, where w_i is calculated by substituting u_i and $x_{1,i}, \ldots x_{n-1,i}$ into equation (12), solving with respect to $w(k)$ and replacing $w(k)$ by w_i:

$$w_i = \left((a_0 - b_0) + (a_1 - b_1) x_{1,i} + \cdots + (a_{n-1} - b_{n-1}) x_{n-1,i} + a_n u_i\right) / b_n \ . \tag{13}$$

The M_i define a piecewise linear controller $u(k) = f_M\left(\underline{x}(k), w(k)\right)$ with appropriate linear control law in each simplex T_i. If the control output $u(k)$ is then used as model input the desired dynamics (11) is followed exactly:

$$f\left(\underline{x}(k),\, f_M\left(\underline{x}(k), w(k)\right)\right) = f_R\left(\underline{x}(k), w(k)\right).$$

An example may illustrate the design procedure for the piecewise linear model depicted in Figure 8. The interpolation functions (equations (7), (8)) are defined by the nodes $N_i = (x_i, u_i, f_i)$:

$$S_1: \text{x(k+1)} = 1/2 - 1/4\,x(k) + 1/4\,u(k) \quad (7)$$

with $N_1 = (0,0,1/2)$, $N_2 = (1,0,1/4)$, $N_4 = (1,1,1/2)$,

$$S_2: \text{x(k+1)} = 1/2 - x(k) + u(k) \quad (8)$$

with $N_1 = (0,0,1/2)$, $N_3 = (0,1,3/2)$, $N_4 = (1,1,1/2)$.

The desired reference dynamics may be given as

$$\text{x(k+1)} = -1/2\,x(k) + 3/2\,w(k). \quad (14)$$

The controller will be piecewise linear, defined by the nodes $M_i = (x_i, w_i, u_i)$. The M_i are calculated by equation (13), mapping the model's nodes. Then, the control laws are obtained by applying equation (12) for either simplex. This yields the two controller simplices

$$T_1: \text{u(k)} = -x(k) + 8/3\,w(k) - 2 \quad (15)$$

with $M_1 = (0,3/4,0)$, $M_2 = (1,9/8,0)$, $M_4 = (1,3/2,1)$,

$$T_2: \text{u(k)} = 1/2\,x(k) + 2/3\,w(k) - 1/2 \quad (16)$$

with $M_1 = (0,3/4,0)$, $M_3 = (0,9/4,1)$, $M_4 = (1,3/2,1)$.

T_1 and T_2 are non-overlapping since the model is a one-to-one mapping. It is easy to prove that these control laws produce the reference dynamics (14) by substituting (15) and (16) into (7) and (8), respectively,

5 CONTROLLER DESIGN FOR A TWO TANK SYSTEM

The design method presented above is applied to the two tank system in laboratory shape depicted in Figure 12. Influx u is the control variable, height h in the left tank is the output variable of the system. Krebs (1998) determined the system structure and Schmid (1998) outlined the generation of a piecewise linear process model. In this paper, a piecewise linear model consisting of 29 rules is used. The model was obtained by identification of measured data.

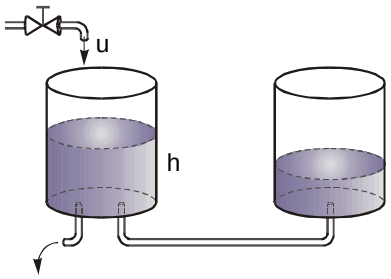

Figure 12 : Two tank system.

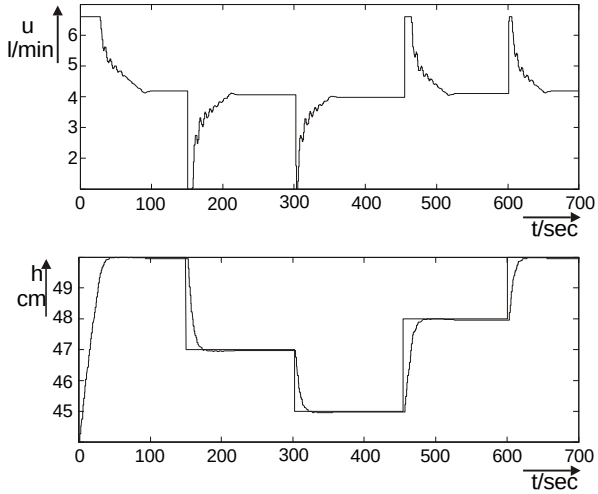

Figure 13 : Dynamic response of the controlled two tank system (bottom) and history of the control variable (top).

Input-output description of the system yields three state variables, actual and last height $h(k)$ and $h(k-1)$, and last control $u(k-1)$. The state variables as well as the actual control $u(k)$ are inputs of the model. The model output is an estimate for the height $h(k+1)$. Controller design according to the previous section yields a controller with the state variables and the command variable $w(k)$ as inputs. The reference dynamics to be followed is given as

$$y_{k+1} = y_k - 1/4\,y_{k-1} + 1/4\,w_k.$$

The performance of the piecewise linear controller with 29 rules is shown in the lower part of Figure 13 for several set points. The upper part of Figure 13 shows the history of the control variable. The control variable is limited to the interval [1l/min; 6,6 l/min]. The controller does not yield steady state accuracy since deviations between model and

system are not considered. For that purpose, the controller may be embedded in an internal model control structure, as proposed by Ullrich (1997). Further improvements may be achieved by adaptive strategies described in Brown (1994).

6 CONCLUSIONS

This paper presented a powerful new fuzzy inference method that particularly considers the informational content of inputs and output. Inference is based on the concept of interpolating rules, first calculating an interpolating rule, and in a second step mapping the fuzziness of the inputs onto the outputs. The inference method is the core of a class of qualitative process models called dynamic fuzzy systems.

The model output is calculated by means of interpolation functions. Lattice-like interpolation functions suffer from the curse of dimensionality, the number of rules exponentially growing with the number of inputs. The proposed piecewise linear interpolation copes with that problem, reducing the number of necessary rules. However, evaluation of the piecewise linear interpolation functions needs advanced mathematical methods, e.g. a transformation to barycentric coordinates.

The major benefit of using piecewise linear functions lies in the way a model based controller may be derived. Analytical model inversion is possible and a desired reference dynamics may be adjusted. The resulting controller compensates the system's nonlinearities. The presented control design is applicable to systems where compensation is possible and does not lead to unstable internal dynamics.

Further investigation will focus on more accurate interpolation functions and application of advanced control design methods.

REFERENCES

Berger, C.S., 1997, "Modelling dynamic systems using finite elements", Int. J. of Control, vol.68, no.3, pp.431-448.

Brown, Martin; Harris, Chris, 1994, "Neurofuzzy Adaptive Modelling and Control", Prentice Hall, New York.

Driankov, Dimiter; Hellendoorn, Hans; Reinfrank, Michael, 1993, "An Introduction to Fuzzy Control", Springer-Verlag, Berlin.

Krebs, Volker; Schäfers, Elmar, 1998, "Dynamische Fuzzy-Systeme zur qualitativen Prozeßmodellierung", VDI/VDE-GMA-Tagung Computational intelligence: neuronale Netze, evolutionäre Algorithmen, fuzzy control im industriellen Einsatz; VDI-Berichte 1381, Berlin, Germany, pp. 115-135.

Preparata, Franco P.; Shamos, Michael Ian, 1985, "Computational Geometry: An Introduction", Springer-Verlag, New York.

Schäfers, Elmar; Krebs, Volker; Schmid, Klaus, 1997, "Inferenzverfahren für dynamische Fuzzy-Systeme", Forschungsbericht 7. Workshop Fuzzy Control des GMA-UA 1.4.2, Dortmund, Germany.

Schmid, Klaus; Krebs, Volker, 1998, "Ein Inferenzverfahren für unvollständige Regelbasen", 8. Workshop Fuzzy-Control des GMA-FA 5.22, Dortmund, Germany.

Ullrich, Thorsten; Tolle, Henning, 1996, "Delaunay Networks for Modelling of Non-Linear Processes", Proceedings of the IASTED Conference on Modelling and Simulation, Pittsburgh, USA.

Ullrich, Thorsten; Brown, Martin, 1997, "Delaunay-based nonlinear internal model control", Int. Conference on Application of Computer Systems (ACS '97), Szczecin, Poland.