Proceedings of the 8th IEEE Mediterranean Conference
on Control and Automation (MED 2000)
Rio, Patras, GREECE 17-19 July, 2000

ALGORITHMS FOR COMPUTING FUZZY MODEL NETWORK SYSTEMS

GANCHO VACHKOV♣, TOSHIO FUKUDA♦

♣Department of Micro System Engineering, Nagoya University, Furo-cho 1, Chikusa-ku, Nagoya 464-8603, Japan
E-mail: *vachkov@mein.nagoya-u.ac.jp*

♦Center for Cooperative Research in Advanced Science and Technology, Nagoya University, Furo-cho 1, Chikusa-ku, Nagoya 464-8603, Japan
E-mail: f*ukuda@mein.nagoya-u.ac.jp*

Abstract: In this paper several algorithms for computing the specially proposed Fuzzy Model Network Systems (FMNS) are presented and analyzed. FMNS are complex structures consisting of a number of interrelated simple fuzzy model units and linear junction units (modules). The main computation problem in FMNS is to calculate a part or the entire set of the unmeasured variables in the system with a predetermined structure and known set of measured variables. The computational algorithms presented in this paper include: 1) non-iterative computation of a feedforward type of FMNS; 2) iterative inverse calculation of one-dimensional fuzzy model units and 3) iterative calculation of closed loop (cyclic type) FMNS by use of a specially proposed fuzzy iteration block. All the proposed algorithms are explained and illustrated on numerical examples with comments about their practical application to industrial systems and plants.

Key Words. Network structures, fuzzy models, inverse fuzzy models, iterative algorithms

1. INTRODUCTION

Many industrial processes or mechanical systems can be considered from a system viewpoint as complex systems with large number of input, output and intermediate variables. Complexity of such systems is represented not only by their size, but also by the structure and the type of interrelations between the variables. Let a system has input, output and intermediate variables represented by the vectors $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{V}$, respectively and a structure given by the matrix **S**. In the classical *black box* calculation and identification approach only the inputs $\boldsymbol{X}$ and outputs $\boldsymbol{Y}$ are taken into account with the structure **S** and the intermediate variables $\boldsymbol{V}$ and relationship being neglected.

Unlike the black box approach, the *modular* system approach takes into account the internal structure **S** and the intermediate variables $\boldsymbol{V}$ of the system. Here a number of smaller models of typical system elements (units, modules) are used, each of them representing the relationship between smaller number of system variables. They are called *system modules* that use one type of a model with different settings of its parameters for the different system units.

The modular approach is very useful in many practical cases. For example these are the cases when the available set of measurable variables includes only certain part of the variables in $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{V}$ and the system structure **S** is known. Fault diagnosis problems [1,4] are typical cases of a system calculation under partial measured information. Here the computation task is to find the possible system inputs by using part of the measured outputs and/or intermediate variables.

Fuzzy models can be successfully used as system modules in the modular system approach. Especially the Takagi-Sugeno (*TS*) fuzzy models [2,3,7] are convenient modeling tools since they are considered as universal approximators [3] of wide class of nonlinear relationships. In this paper simplified one-dimensional TS fuzzy models as well as linear junctions are used as main units in the specially

proposed Fuzzy Model Network Systems (FMNS). Then some computational algorithms for FMNS with feedforward and loop structure are proposed and analysed.

2. STATEMENT OF THE PROBLEM

We assume further that the Fuzzy Model Network System under investigation is a complex interconnected structure between M process variables, represented by N one-dimensional non-linear fuzzy model units and P junction points (algebraic operator units). All M variables of the system can be classified into the following three vectors $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{V}$ of the input, output and intermediate variables, respectively. It is also assumed that all N fuzzy model units have been identified to a desired level of accuracy representing in a plausible way the nonlinear relations between a certain pairs of system variables. In order to enable any kind of connection between the system variables we assume that each junction point unit and each fuzzy model have both input and output gain coefficients, as follows:

$$y = Ky \sum_{i=1}^{n} Kx_i x_i \ , \tag{1}$$

for the junction point unit with n inputs as shown in Fig. 1. and

$$y = Kx.F(x).Ky \ , \tag{2}$$

for the fuzzy model unit, as shown in Fig. 2. $\boldsymbol{F}(x)$ is a nonlinear fuzzy mapping of the one-dimensional fuzzy model, as described in the sequel.

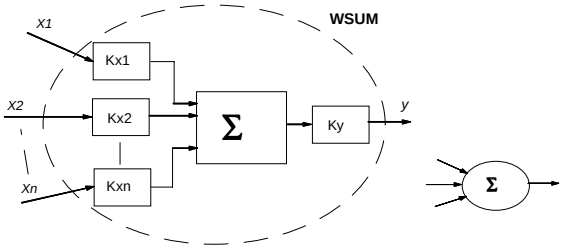

Fig. 1. Linear junction point unit

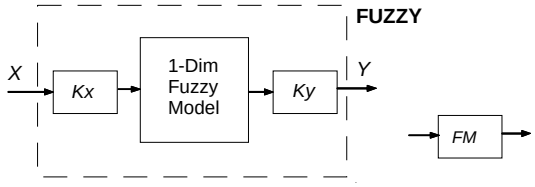

Fig. 2. One-dimensional fuzzy model unit

Further we assume that all the measured variables are contained in the vector $\boldsymbol{T}$ with

$$T \subset X \bigcup Y \bigcup V \tag{3'}$$

In the general case some of the system variables could be computed (retrieved) by using the information of the measured variables in $\boldsymbol{T}$ and the system units (models). Such variables form the vector $\boldsymbol{Z}$ of the so called *computable variables*. Finally if there are system variables that cannot be retrieved by any computation scheme, they form the vector $\boldsymbol{W}$ of the *non-computable* (non-retrievable) *variables*. The following relationship exists between the above defined vectors:

$$X \bigcup Y \bigcup V = T \bigcup Z \bigcup W \tag{3''}$$

Now the computation problem of a Fuzzy Model Network System can be formulated is as follows:
Given a FMNS with structure $\boldsymbol{S}$ *and vector* $\boldsymbol{T}$ *of measured variables, find a maximal vector* $\boldsymbol{Z}$ *of the computed variables, i.e.* $|\boldsymbol{Z}| \rightarrow max$ or $|\boldsymbol{W}| \rightarrow min.$

Additionally a *degree of computability* can be formulated as a ratio of the number of computed variables over the total number of the unmeasured variables, that is:

$$d = \frac{|Z|}{|Z| + |W|} \in [0,1] \tag{4}$$

3. ALGORITHM FOR COMPUTING FEEDFORWARD SUBSYSTEMS

The standard computing approach used in any type of complex systems, is based on a preliminary decomposition of the overall system structure into a number of sub-systems from two different types, namely: *feedforward* type (non-cyclic) subsystems and *cyclic* type (loop subsystems) as in [4,5]. Then the subsystems from the feedforward type are computed in a non-iterative way provided that the computation order of the units is preliminary analyzed as a certain sequence of units.

This standard computing approach assumes that all input variables are measured and all output and intermediate variables are to be computed. However in our more common statement of the problem the measured variables $\boldsymbol{T}$ could be located anywhere in the system structure.

An example of a feedforward subsystem with two measured variables (the bold circles •) is shown in Fig. 3.

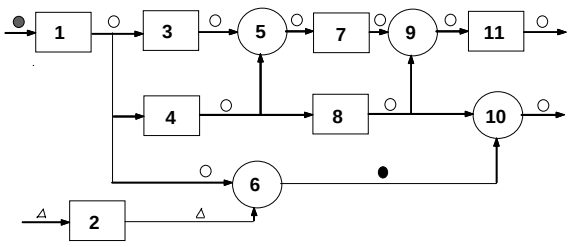

Fig. 3. Example of feedforward type of network system

The general computing algorithm of a feedforward subsystem, shown below is based on the concept of *forward* and *backward* (reverse*)* computability of the units as follows:

A system unit is called to be in a status of *forward computable* unit if its output is not measured and all of its inputs are measured or have been computed in previous computation steps.

Similarly, a system unit is in a status of *backward computable* unit if it has only one not measured input and all other inputs, including the output have been measured or computed in previous steps.

The proposed computing algorithm of the feed-forward system consists of the following main steps:

1. Search for existence of a *forward computable* unit in the system structure.

   a) If "*Yes*", perform a forward calculation of the unit. Mark the new computed output stream (variable) with ○. Go to Step 1.

   b) If "*No*", Go to Step 2.
2. Search for existence of a *backward computable* unit.

   a) If "*Yes*", perform an inverse calculation of this unit. Mark the computed output by Δ. Go to Step 1.

   b) If "*No*" Stop.]

Table 1. shows the performance results of this algorithm for *4* different examples of initial measured variables. The solution of Example 1 is also displayed graphically on the structure in Fig. 3. It is seen that Examples *1*,*2* and *3* correspond to a fully computable system (*d = 1*), while Example *4* has a solution with *4* non-computable variables (—) with degree of computability: *d = 4/11 = 0.364* < 1.

Table 1. Computability of the System in Fig. 5.

No	*Stream*	***Ex. 1***	***Ex. 2***	***Ex. 3***	***Ex. 4***
1	*0 - 1*	●	●	Δ	Δ
2	*0 - 2*	●	Δ	Δ	—
3	*1 - 3*	❍	❍	Δ	●
4	*1 - 4*	❍	❍	Δ	●
5	*1 - 6*	❍	❍	Δ	●
6	*2 - 6*	❍	Δ	Δ	—
7	*3 - 5*	❍	❍	❍	❍
8	*4 - 6*	❍	❍	●	●
9	*4 - 8*	❍	❍	●	●
10	*5 - 7*	❍	❍	❍	❍
11	*6 – 10*	❍	●	●	—
12	*7 - 9*	❍	❍	●	❍
13	*8 – 9*	❍	❍	❍	❍
14	*8 – 10*	❍	❍	❍	❍
15	*10 - 0*	❍	❍	❍	—
16	*11 - 0*	❍	❍	❍	❍

4. FORWARD CALCULATION OF THE FUZZY MODEL UNIT

As mentioned in the previous sections, a one-dimensional *TS* fuzzy model is used for the fuzzy unit in Fig. 1. as a non-linear approximator [3]. It possesses large flexibility in a sense that the fuzzy model parameters can be adjusted for any kind of assumed relationship between a given pair of system variables. The one-dimensional *TS* fuzzy model consisting of L fuzzy rules has the following format [2,3,7]:

$$IF\,(x\ is\ A_i)\ THEN\ \ y_i = p_{0i} + p_{1i}x\ ,\ i=1,2,...,L \qquad (5)$$

where A_i is a fuzzy set with *Gaussian* type membership function:

$$f_i(x) = \exp\left(-\frac{(x-c_i)^2}{2\sigma_i^{\,2}}\right),\ i = 1,2,...,\ L \qquad (6)$$

The tuning parameters of this fuzzy model are: centers c_i and spreads σ_i of the membership functions as well as the consequence parameters: p_{0i} and p_{1i}, $i = 1,2,\ldots,L$.

The straightforward (non-iterative) calculation of the above fuzzy model with a weighted average method for defuzzification is written as follows:

$$y = F(x) = \frac{\sum_{i=1}^{L} f_i(x)y_i}{\sum_{i=1}^{L} f_i(x)} = \frac{\sum_{i=1}^{L} f_i(x)(p_{oi} + p_{1i}x_1)}{\sum_{i=1}^{L} f_i(x)}. \qquad (7)$$

5. ITERATIVE INVERSE CALCULATION OF THE FUZZY MODEL UNIT

The proposed general computing algorithm in Section 3 is based on the ability of inverse calculation of the fuzzy model unit. The problem of inversion of the fuzzy models has been studied from analytical and numerical calculation viewpoint in [2,6]. It is an iterative procedure usually based on the observer concept as in [6] and by using different optimization techniques such as Newton-Gauss iteration scheme [2,6] or Levenberg-Marquardt algorithm as in [2], steepest descent method and possibly others.

Fig. 4. shows in a graphical way the correct inversion problem in the case of monotonously increasing function $y = f(x)$. In Fig. 5. the problem of ambiguity of the inverse solution is graphically demonstrated. Therefore in order to get a reliable and meaningful solution, the inversion procedure should be constrained in a preliminary fixed feasible boundaries.

In this paper we accept the observer concept of iterative inversion, as shown in Fig. 6. As easily noticed, this is an iterative procedure of gradually adjusting the unknown input until the output of the process model matches the real (known) output. Here we use an original fuzzy iteration control scheme as

discussed further.

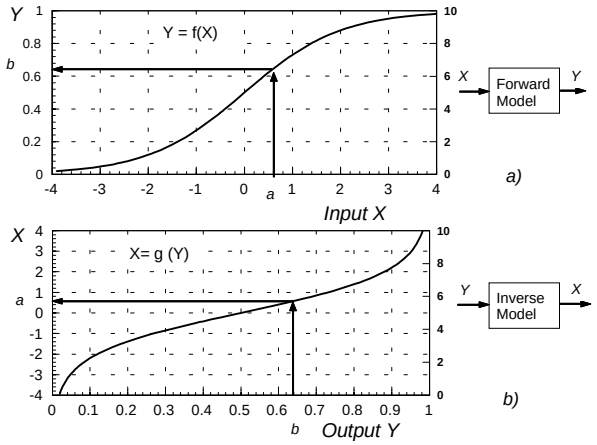

Fig. 4. Example of unique type of inversion

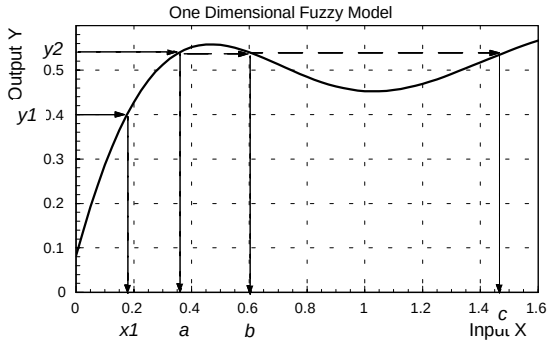

Fig. 5. Ambiguity problem in the inversion of a fuzzy model

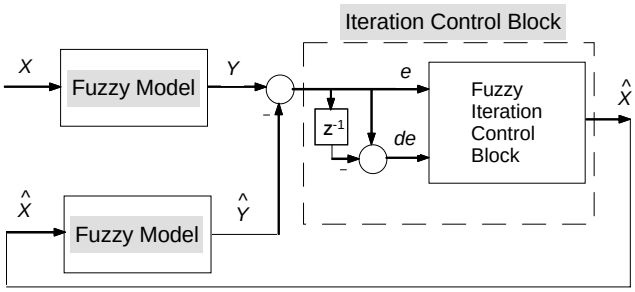

Fig. 6. The Observer concept of iterative inversion of a fuzzy model

The classical and simple Gauss-Newton iteration scheme used for fuzzy model inversion in [2,6] updates the input value $\hat{x}(k)$ with an increment $du(k)$ according to the error $e(k)$ and the function derivative $df(k)$, as follows:

$$\hat{x}(k+1) = \hat{x}(k) + \eta.du(k) = \hat{x}(k) + \eta\frac{e(k)}{df(k)}, \quad (8)$$

where k is the current iteration number and $\eta < 1$ is a predetermined learning rate. In more concrete form the Gauss-Newton iteration scheme could be written as:

$$\hat{x}(k+1) = \hat{x}(k) + \eta\frac{y - \hat{y}(k)}{f'[\hat{x}(k)]} \quad (9)$$

For the one-dimensional SISO type fuzzy model and consequences of the fuzzy rules taken as singletons $y_i = p_{oi}$, the derivative $df(k)$ can be calculated analytically as shown in [2] as:

$$\frac{dy}{dx} = f'(x) = \sum_{i=1}^{L} v_i(x)(x - c_i)(y - u_i)/\sigma_i^2 \quad (10)$$

where v_i are the normalized firing levels:

$$v_i(x) = f_i(x) / \sum_{l=1}^{L} f_l(x),\ i = 1,2,...,L \quad (11)$$

and $f_l(x)$ is the fuzzified value of the input x by the l-th *Gaussian* membership function as in (6).

It is easy to realize from (8) and (9) that the Gauss-Newton iteration scheme works well and would lead to a fast convergence if the function derivative $df(k)$ as well as the approximation error $e(k)$ are different from zero. Otherwise a quite slow learning or even undefined calculation value may easily occur and break the learning process. This calculation problem can be realized by the graphical representation of the control surface of the Gauss-Newton iteration scheme shown in Fig. 7. Here the instability of this iteration scheme becomes obvious in the areas with near zero derivatives.

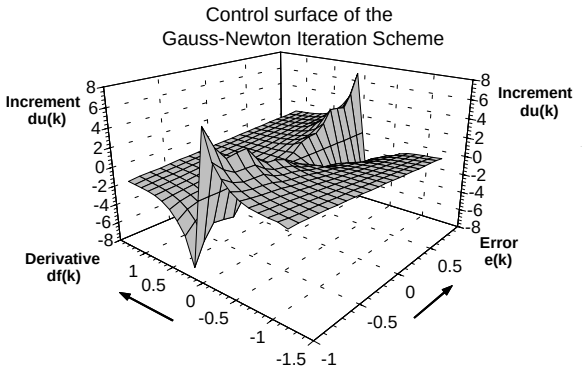

Fig. 7. Control surface of the Gauss-Newton iteration scheme

6. FUZZY ITERATION CONTROL BLOCK

In order to improve the performance of the Gauss-Newton iteration scheme, we propose the use of a fuzzy approach to approximate the original control surface from Fig. 7. by a respective two-dimensional fuzzy model.

For the approximation process we used 7 Gaussian membership functions for each input, namely the deviation (error) *E* and the function derivative *DF*, as shown in Fig. 8

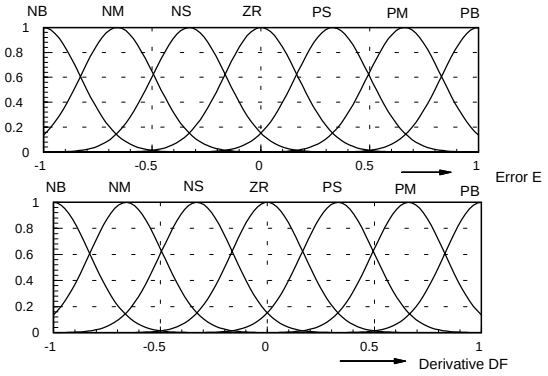

Fig. 8. Membership functions of the fuzzy iteration

The singletons of the fuzzy model have been adjusted roughly by approximate reasoning and human logic based on the shape of the original control surface in Fig. 7. Table 2 shows the Fuzzy Rule base with the singletons of the fuzzy model.

Table 2. Fuzzy rule base with singletons

PB	*-3*	*-2*	*-1*	*0*	*+1*	*+2*	*+3*
PM	*-4*	*-3*	*-2*	*0*	*+2*	*+3*	*+4*
PS	*-7*	*-5*	*-3*	*0*	*+3*	*+5*	*+7*
ZR	*-9*	*-6*	*-3*	***0***	*+3*	*+6*	*+9*
NS	*+7*	*+5*	*+3*	*0*	*-3*	*-5*	*-7*
NM	*+4*	*+3*	*+2*	*0*	*-2*	*-3*	*-4*
NB	*+3*	*+2*	*+1*	*0*	*-1*	*-2*	*-3*
▲ **e**	***NB***	***NM***	***NS***	***ZR***	***PS***	***PM***	***PB***
	→ ***df***						

The result of this kind of "soft" approximation is shown in Fig. 9. The control surface there resembles the original one in Fig. 7., but is much smoother and does not have disruptions at the zero derivative point. This fact gives a kind of preliminary confidence for the principal ability of the fuzzy control block to operate satisfactory the learning process in a wide range of derivatives and errors.

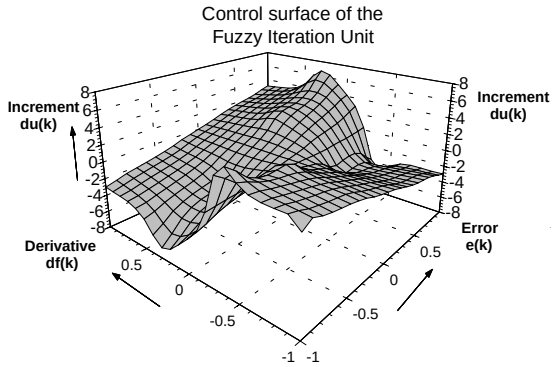

Fig. 9. Surface of the newly proposed fuzzy iteration control block

Quite often may happen that the function derivative $df(k)$ cannot be calculated analytically by use of (10) and (11). These are cases when the iteration scheme is used for other more complicated and nonlinear objects than just a single-input fuzzy model inversion. Then the function derivative $df(k)$ can be estimated numerically, by the following equations:

$$e(k) = y - \hat{y}(k) \tag{12}$$

The change-of-error $de(k)$ is evaluated as:

$$de(k) = e(k) - e(k-1) = y - \hat{y}(k) - [y - \hat{y}(k-1)] = -[\hat{y}(k) - \hat{y}(k-1)] \tag{13}$$

The change-of-input is:

$$dx(k) = \hat{x}(k) - \hat{x}(k-1) \tag{14}$$

Then the function derivative will be:

$$df(k) = \frac{dy(k)}{dx(k)} \approx \frac{\hat{y}(k) - \hat{y}(k-1)}{\hat{x}(k) - \hat{x}(k-1)} = -\frac{de(k)}{dx(k)} \tag{15}$$

In order to make a kind of standardization of the performance of the fuzzy iteration block, the following normalization procedures with saturation are introduced for the error $e(k)$, function derivative $df(k)$ and the increment (control action) $du(k)$:

$$-1 \le E(k) \ge +1 \tag{16}$$

$$E(k) = \begin{cases} e(k)/SFe, & if \ |e(k)| \le SFe \\ 1.\mathrm{sgn}[e(k)], & otherwise \end{cases} \tag{17}$$

$$-1 \le DF(k) \ge +1 \tag{18}$$

$$DF(k) = \begin{cases} df(k)/SFdf, & if \ |df(k)| \le SFdf \\ 1.\mathrm{sgn}[df(k)], & otherwise \end{cases} \tag{19}$$

$$-1 \le DU(k) \ge +1 \tag{20}$$

$$DU(k) = \begin{cases} du(k)/SFdu, & if \ |du(k)| \le SFdu \\ 1.\mathrm{sgn}[du(k)], & otherwise \end{cases} \tag{21}$$

Here *SFe*, *SFdf* and *SFdu* are predetermined scaling factors for the error, function derivative and the control action (increment) of the fuzzy iteration block. They are the tuning parameters of the whole fuzzy iteration scheme. Actually they change the saturation level of the respective variable thus changing the performance of the iteration block. Therefore the careful selection of *SFe*, *SFdf* and *SFdu* is very important for the total iteration and computation process.

The final iterative inversion scheme based on the proposed fuzzy iteration block can be summarized as follows:

$$\hat{x}(k+1) = \hat{x}(k) + \eta . DU(k) \tag{22}$$

where $DU(k)$ is the output of the fuzzy model from Table 2, and Fig. 8. that produces the smooth control surface shown in Fig. 9.:

$$DU(k) = F[E(k), DF(k)] \tag{23}$$

Extensive simulation experiments have been carried out to compare the performance the Newton Gauss iteration block and the proposed Fuzzy Iteration Control block. In the case of inversion of a single one-dimensional fuzzy model unit it has been found that the fuzzy iteration control block is more

sensitive to the tuning of the scaling factor for the output *SFdu* than to the tuning of the other two scaling factors: *SF*e and *SFdf*. This block also exhibits some degree of robustness as shown in Fig. 11. where even with inaccurate tuning, (*SFdu = 0.35*), it does not oscillate significantly. The reason is in the implemented fuzzy logic similar to the switching control type fuzzy controller that puts upper and lower levels on the oscillation process..

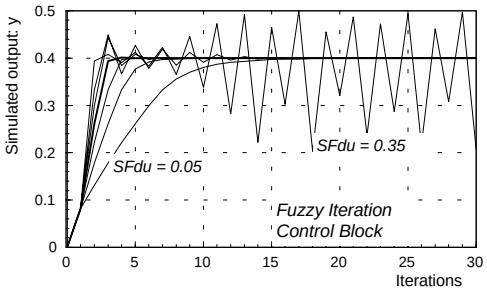

Fig. 10. Convergence and oscillations of the fuzzy iteration block with different scaling factors *SFdu*

7. ITERATIVE COMPUTATION OF CLOSED LOOP SUBSYSTEMS

The fuzzy model network structure may have also a cyclic type (closed loop) subsystems, as the one shown in Fig. 11. Generally the computational scheme of such systems requires iterative calculations where a set of so called "torn" (disconnected) streams [4,5] is used to control the convergence of the iteration process, as the "torn" stream *5* in Fig. 11.

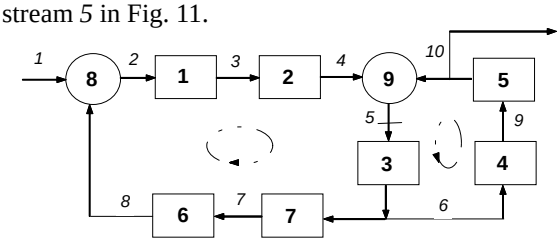

Fig. 11. A closed loop subsystem with one "torn" stream: *5*.

Then the computational strategy of this system consists of a repetitive calculation of all the units in a specified computation sequence, for example: (*3 – 7 – 6 – 8 – 1 – 2 – 4 – 5 – 9*) for the system in Fig. 11. Fig. 12. shows some different possible cases of nonlinear relationships during the iteration process until convergence with gain *G=1* is achieved.

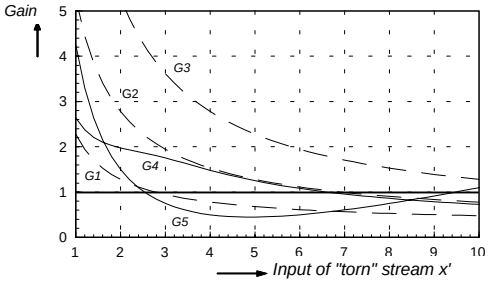

Fig. 12. Nonlinear relationship between the input and the gain of the closed loop system.

Usually the non-linearity in the iteration loop here cannot be expressed analytically and numerical evaluation of the derivatives should be used. Therefore the proposed above fuzzy control iteration block could be an appropriate tool for efficient conversion in computing the closed loop systems since it does not require analytical calculation of the derivative and relies on its numerical evaluation. Some preliminary numerical simulation carried out on the system in Fig. 11. have snown that the fuzzy iteration block reaches convergence faster than the widely used simple iteration (substitution) method. Still the problem of proper tuning of the scaling factors of this block needs to be further analysed.

8. CONCLUSIONS

In this paper some computational algorithms for feedforward and closed loop fuzzy model network systems FMNS have been proposed and analysed. The standard structure of such systems consists of *two types* of system units: one dimensional fuzzy models and linear junction points that are able to represent arbitrary relationships between any system variables.

A special fuzzy iteration block is proposed to carry out the iteration process in the fuzzy model inversion as well as in the loop system calculation. This block exhibits a kind of robustness and works with numerical evaluated derivatives. By use of the proposed computational algorithms maximum degree of computability of the total system can be achieved. Future research is aimed at a feasible algorithm of learning all the model units in the FMNS.

9. REFERENCES

[1] I-Cheng Chang, Cheng-Ching Yu, Ching-Tien Liou, Model-Based Approach for Fault Diagnosis. 1. Principles of Deep Model Algorithm, *Industrial & Engineering Chemistry Research*, 33, 1994, 1542-1555.

[2] Filev D. Inversion of Fuzzy Models – Practical Issues, Proc. of the *FUZZ-IEEE'98*, Anchorage, Alaska, USA, May 4-9, 1998, pp. 1658-1663.

[3] Hao Ying, General SISO Takagi-Sugeno Fuzzy Systems with Linear Rule Consequent Are Universal Approximators, IEEE Trans. on Fuzzy Systems, Vol. 6, No. 4, 1998, pp. 582-587.

[4] Shiozaki, J., Matsuyama, H., O'Shima E., Iri M., An Improved Algorithm for Diagnosis of System Failures in the Chemical Process, *Comput. Chem. Eng*., 9, 1985, 285.

[5] Swamy M., Thulasiraman K. Graphs, Networks and Algorithms, *Wiley*, New York, 1981.

[6] Varkonyi-Koczy A., Peceli G., Dobrowiecki T., Kovacshazy T. Iterative Fuzzy Model Inversion, Proc. of the, *FUZZ-IEEE'98*, Anchorage, Alaska, USA, May 4-9, 1998, pp. 561-566.

[7] R. R. Yager, On the Construction of Hierarchical Fuzzy Systems Models, *IEEE Trans. on Systems, Man and Cybernetics*, v. 28, No. 1, 1998, 55-66.