COMPARISON OF FUZZY MODELLING APPROACHES FOR FAULT DETECTION SYSTEMS

Péter Baranyi, Ron J. Patton and Faisel J. Uppal

Control and Intelligent Systems Research Group,
University of Hull, Cottingham Road, Hull, HU6 7RX, UK.
e-mail: baranyi@alpha.ttt.bme.hu; r.j.patton@hull.ac.uk

Keywords: fault diagnosis, fuzzy modelling, parallel distributed compensation, modelling complexity and accuracy

Abstract

Fuzzy modelling together with Parallel Distributed Compensation (PDC)-based system analysis and controller/observer design techniques have emerged among the methods for developing fault detection and isolation (FDI). This paper provides a comparison of fuzzy logic modelling approaches for FDI of complex systems. The work is motivated by the modelling issues of the Takagi-Sugeno (T-S) fuzzy inference operator-based approximation structure that is essential for the implementation of a PDC system. The paper focuses on an application study of fault diagnosis for an industrial actuator system, conducted within the framework of the European study DAMADICS.

1 Introduction

Computational intelligence techniques have been recently investigated as an extension of traditional model-based fault diagnosis methods [1, 2]. The purpose of this paper is to compare two fuzzy logic-based modelling approaches, typically considered for FDI of complex (difficult to model) dynamic systems. The work is developed within the framework of a European Research Training Network study*DAMADICS* (home page: *http://diag.mchtr.pw. edu.pl/damadics/*). The comparison is based on an application study of an electro-pneumatic valve actuator applied in a sugar factory process in Lublin, Poland. These modelling approaches are termed Tensor Product Transformation (TPT) and Identification Package (IP). The detailed description of the TPT and the IP is beyond the scope of this paper and can be found in [3, 4, 5, 6]. The common primary goal of these modelling approaches is to represent a dynamic system by a TSK fuzzy model introduced by Takagi, Sugeno and Kang, see [10]. However, they have different properties in various modelling and implementation aspects that will be analysed in this paper. In the following, some aspects of TPT and IP are outlined as a basis for a comparative study.

With regard to implementation aspects, the TSK fuzzy models, have exponential computational complexity with their approximation accuracy. Furthermore, the TSK fuzzy model is "nowhere dense in the modelling space" if the number of rules is bounded. This implies that in pursuit of good modelling accuracy we should, in general face the exponential explosion of the TSK fuzzy modelling complexity. Hence, the task is to find an optimal trade-off between the modelling accuracy and processing complexity. This task plays an important role in many real systems with time-consuming implementation features. The processing time in fault diagnosis systems can be crucial in preventing and avoiding system shutdown, breakdown and even catastrophes. It is in the light of this application challenge that we here compare the TPT and IP approaches and describe how the approximation trade-off can be investigated.

Furthermore, the ability to deal with non-linearity and multiple-model complexity has resulted in an increased attention to PDC analysis and design techniques for FDI systems [4]. An important issue arising from this is the identified fuzzy-inference model structure can be linked with alternative analysis and design techniques. This paper investigates how effectively the TPT and IP approaches can be linked with/related to the PDC computational framework [7].

All the TPT, IP and PDC designs illustrated in the paper are executed numerically by computer without analytic and heuristic derivations. If we can determine an immediate links between these approaches then we may be able to design self-developing and learning FDI fault detection systems.

2 Nomenclature

In this section we introduce the notations being used in the paper: $\{a, b, \ldots\}$: scalar values. $\{\mathbf{a}, \mathbf{b}, \ldots\}$: vectors. $\{\mathbf{A}, \mathbf{B}, \ldots\}$: matrices. $\{\mathcal{A}, \mathcal{B}, \ldots\}$: tensors. $\mathcal{R}^{I_1 \times I_2 \times \cdots \times I_N}$:vector space of real valued $(I_1 \times I_2 \times \cdots \times I_N)$-tensors. The subscript defines the lower order: for example, an element of the matrix $\mathbf{A}$ at row-column number i, j is symbolized as $(\mathbf{A})_{i,j} = a_{i,j}$. Systematically, the ith column vector of $\mathbf{A}$ is denoted by $\mathbf{a}_i$, i.e. $\mathbf{A} = \begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \cdots\end{bmatrix}$. $\diamond_{i,j,n}, \ldots$: are indices. $\diamond_{I,J,N}, \ldots$: index upper bound: for example: $i = 1..I$, $j = 1..J$, $n = 1..N$ or $i_n = 1..I_n$. $\mathbf{A}_{(n)}$: n-mode matrix of tensor $\mathcal{A} \in \mathcal{R}^{I_1 \times I_2 \times \cdots \times I_N}$. $\mathcal{A} \times_n \mathbf{U}$: n-mode matrix-tensor product. $\mathcal{A} \otimes_n \mathbf{U}_n$: multiple product as $\mathcal{A} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 .. \times_N \mathbf{U}_N$. FDI: Fault Detection and Isolation. TPT: Tensor Product Transformation; IP: Identification Package. TSK: Takagi Sugeno Kang.

3 System Description

A detailed system description of the benchmark study and real measured training data sets of the industrial sugar factory process considered in the framework of the DAMADICS EU RTN contract are available on the home page of DAMADICS. This section gives a brief introduction to the elements of the process upon which the application study is based.

3.1 General introduction of the sugar process

The sugar factory plant produces 50,000 tonnes of sugar annually and consists of a large number of evaporation plants, boiler houses, heaters and valves. As a raw feedstock to the production of sugar, sugar beet is farmed and supplied to the factory. Raw syrup is obtained from thin sliced beets using extractors. After cleaning, decalcifying and processing the syrup, the syrup containing 14% of sugar is condensed to 70% solution using the evaporation station. Waste steam from the steam turbine is used as a heat source for the complete process. A mixture of granulated sugar and syrup is obtained after the crystallisation process in syrup boilers. For further details the reader is referred to [8].

3.2 Actuator faults in the system

The local PI controller loop that controls the syrup levels in the evaporator is throttling the syrup flow by acting on the control valve (see later). The set point syrup level in the evaporator should be controlled within the limits of a few %. The alarms are set when the syrup level is outside set limits. Both situations are dangerous for the process and process operators. In the case of too low a syrup level in the evaporator, the danger of exploding the boiler due to its overheating occurs. From knowledge of the syrup volume in the evaporator one can easily deduce that a dangerous situation could be detected by alarm analysis within 30s, giving further 30s for operator intervention. Shortening the analysis time may be the crucial point in this case. This is principally the task of advanced on-line FDI. The application of new diagnostic algorithms in smart embedded control elements may bring significant increase to overall system safety, reliability and economy [4, 9] whilst also providing a powerful tool for local performance monitoring.

3.3 FDI scheme of the electro-pneumatic valve actuator

This subsection focuses on the definition of an FDI scheme for electro-pneumatic valve actuator described above, as a crucial part of the process. The modelling techniques of the next sections are compared using this industrial actuator system example, as a basis for the FDI tasks described in [5] and in [6]. The whole valve assembly consists of 3 main parts, see Figure 1.

The PI block controls the positioner and actuator output to regulate the flow through the valve. Pneumatic pressure is applied to the servomotor diaphragm to control the stem position that changes the flow as depicted in the block labelled "internal PI". The positioner adjusts this pressure input to the servomotor to obtain the correct stem position of the actuator, see Figure 2. The valve is the final element in the assembly that alters the flow according to the valve stem position, see Figure 1.

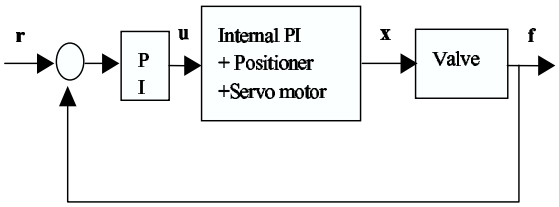

Figure 1: Main parts of the valve assembly

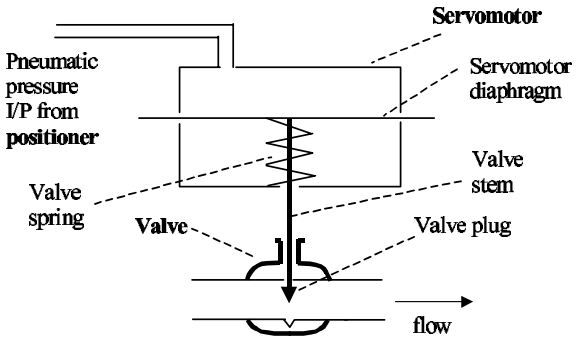

Figure 2: The servomotor and valve assembly

The following faults are detected and isolated:

- **Control valve faults** F1: Valve clogging F2: Valve plug or valve seat sedimentation F3: Valve plug or valve seat errosion F4: Internal leakage (valve tightness) F5: Medium evaporation or critical flow

- **Positioner fault** F6: Rod displacement sensor fault General faults / external faults F7: Unexpected pressure drop across valve F8: Fully or partly opened bypass valves F9: Flow rate sensor fault

The FDI scheme uses fuzzy models to identify the faulty and the fault-free behaviour of the system. The FDI residual signals are generated by the fuzzy models of the plant that approximate the non-linear electro-pneumatic valve by means of local linear models. Each local model is a linear approximation of the process in an I/P subspace and the selection of the local model is fuzzy. The optimal structure found for the fuzzy models is as follows [5,6]: Fuzzy models with a 4-input 2-output structure and fuzzy rules are constructed to generate the residuals $(r_1, r_2, , r_N)$. The inputs of the model are: the control value (u_1), inlet pressure (u_2), outlet pressure (u_3) and temperature (u_4), while the outputs are: the stem displacement of electro-pneumatic servo motor (y_1) and 1-liquid flow through the valve (y_2).

3.4 Fuzzy model applied in the FDI scheme

This subsection is intended to specify the fuzzy model form that is utilised in the FDI scheme of the electro-pneumatic valve actuator. Assume a given dynamic model:

$$s\mathbf{x}(t) = \mathbf{A}(\mathbf{p}(t))\mathbf{x}(t) + \mathbf{B}(\mathbf{p}(t))\mathbf{u}(t) \quad (1)$$

$$\mathbf{y}(t) = \mathbf{C}(\mathbf{p}(t))\mathbf{x}(t) + \mathbf{D}(\mathbf{p}(t))\mathbf{u}(t),$$

where the non-linear system matrix is:

$$\mathbf{S}(\mathbf{p}(t)) = \begin{pmatrix} \mathbf{A}(\mathbf{p}(t)) & \mathbf{B}(\mathbf{p}(t)) \\ \mathbf{C}(\mathbf{p}(t)) & \mathbf{D}(\mathbf{p}(t)) \end{pmatrix} \in \mathcal{R}^{O\times I} \quad (2)$$

and vectors $\mathbf{x}(t)$, $\mathbf{u}(t)$ and $\mathbf{y}(t)$ respectively, are the state, input and output vectors; and where $s\mathbf{x}(t) = \dot{\mathbf{x}}(t)$ is for a continuous-time system or $s\mathbf{x}(t) = \mathbf{x}(t+1)$ is for a discrete-time system, further, $\mathbf{p}(t)$ is time-varying and is bounded by the N-dimensional space $\mathbf{p}(t) \in \Omega : [a_1, b_1] \times [a_2, b_2] \times .. \times [a_N, b_N] \subset \mathcal{R}^N$. $\mathbf{P}(t)$ may, for instance, include some elements of $\mathbf{x}(t)$ or $\mathbf{u}(t)$.

The TSK model (1) by linguistic rules such as:

$$\textbf{IF} w_{1,i_1}(p_1) \textbf{AND} w_{2,i_2}(p_2) \textbf{AND} ... \textbf{AND} w_{N,i_N}(p_N) \quad (3)$$

$$\textbf{THEN} \mathbf{S}_{i_1,i_2,..,i_N}.$$

whose explicit form via product-sum-gravity inference operator [10] is:

$$\begin{pmatrix} s\mathbf{x}(t) \\ \mathbf{y}(t) \end{pmatrix} = \quad (4)$$

$$= \left(\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} .. \sum_{i_N=1}^{I_N} \prod_{n=1}^{N} w_{n,i_n}(p_n(t)) \mathbf{S}_{i_1,i_2,..,i_N} \right) \begin{pmatrix} \mathbf{x}(t) \\ \mathbf{u}(t) \end{pmatrix}.$$

$w_{n,i}(p_n)$ denotes the i-th membership function of the n-th variable. Matrices $\mathbf{S}_{i_1,i_2,..,i_N}$ are the system matrices of the consequent systems. To avoid complicated indexing, and facilitate further reading let us introduce the following tensor operation notation by re-writing (4) in the form:

$$\begin{pmatrix} s\mathbf{x}(t) \\ \mathbf{y}(t) \end{pmatrix} = \mathcal{S} \underset{n=1}{\overset{N}{\otimes}} \mathbf{w}_n(p_n(t)) \begin{pmatrix} \mathbf{x}(t) \\ \mathbf{u}(t) \end{pmatrix}. \quad (5)$$

Here, the row vector $\mathbf{w}_n(p_n) \in \mathcal{R}^{I_n}$ contains the antecedent membership functions $w_{n,i_n}(p_n)$, the $N+2$ -dimensional coefficient tensor $S \in \mathcal{R}^{I_1 \times I_2 \times \cdots \times I_N \times O \times I}$ is constructed from the consequent system matrices $\mathbf{S}_{i_1,i_2,..,i_N} \in \mathcal{R}^{O\times I}$. The first N dimensions of S are assigned to the dimensions of $\mathbf{p}(t)$. Let us define an important characteristic of the antecedent fuzzy sets, which is referred to as *Ruspini*-partition:

$$\forall n, i, p_n(t) : w_{n,i}(p_n(t)) \in [0, 1]; \quad (6)$$

$$\forall n, p_n(t) : \sum_{i=1}^{I_n} w_{n,i}(p_n(t)) = 1.$$

When the antecedent fuzzy sets fulfill (6) the consequent systems define a fixed polytope, where the system varies in: $\mathbf{S}(\mathbf{p}) \in \{\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_R\}$. Matrices $\mathbf{S}_r \in \mathcal{R}^{O\times I}$ are also termed vertex systems. Actually, the plolytope defines the convex hull of the vertex systems:

$$\mathbf{S}(\mathbf{p}(t)) = co\{\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_R\}_{\mathbf{w}(\mathbf{p}(t))}.$$

4 Identification by TP Transformation: TPT

The primary purpose of the TPT is to transform a state-space dynamic model, given over a bounded domain Ω, to TSK model form (5). The symbolic notation of the model is:

$$(tradeoff, \mathbf{w}_{n=1..N}(p_n(t)), \mathcal{S}) = \quad (7)$$

$$= TP\_transf(\mathbf{S}(\mathbf{p}(t)), \Omega, options),$$

where $\mathbf{S}(\mathbf{p}(t)) \in \mathcal{R}^{O\times I}$ is from the state-space model (2), and $\Omega \subset \mathcal{R}^N$ denotes the bounded domain which the transformation is performed over. "$options$" is to define constrains on the characteristics of the resulting membership functions, see later in the Section 6. The resulting functions $\mathbf{w}_{n=1..N}(p_n(t))$ and tensor $\mathcal{S}$ can be substituted immediately into (5). "tradeoff" defines the relation between the number of rules and the approximation error.

5 Identification Package: IP

IP has three components (see, [5,6]) such as product space clustering [11], which is also available in the form of MATLAB program on the internet page (*http://dutera.et.tudelft.nl/ babuska/*), and tree-like algorithms (LOLIMOT) algorithm defined in [12] and a final tuning by learning algorithms [2]. The main components of the IP serve different approximation purposes such as optimal clustering, complexity reduction and transparency.

Let the symbolic notation of the IP be:

$$(tradeoff, \mathcal{S}) = IP(training\_data, mf\_type, R), \quad (8)$$

The input of the identification algorithm is the training data set, denoted by "$training\_data$" that are generated from the measurements of the sugar plant. "mf_type" is to define the function type of the antecedent membership functions. R is the interval of the number of the resulting rules which we are interested in. Along the same line of reasoning as above the output parameter "$tradeoff$" defines the relation between the number of antecedent fuzzy sets and the approximation error. $\mathcal{S}$ defines the fuzzy system consequents. The shape of the antecedent fuzzy sets are defined according to the selected number of rules and the set "mf_type".

6 Comparison of the identification methods

This section considers the approximation and implementation aspects of the TPT and IP methods as discussed in the introduction.

6.1 General approximation properties of TSK fuzzy model

In this subsection we refer to papers that claim that the complexity requirements of TSK systems (i.e. the number of fuzzy rules in the model) grows exponentially in terms of the number of variables. Investigtors also generally claim that multivariable real-valued continuous functions cannot be approximated arbitrarily well by TSK fuzzy model if the number of fuzzy rules is bounded. These claims lead to the necessary trade-off between approximation complexity and accuracy and hence to the trade-off between identification complexity and accuracy. This work provides the proof for the *no-where denseness* of the TSK fuzzy model when it has a bounded number of fuzzy rules.

6.2 Essential difference between TPT and IP

- The TPT is a numeric algorithm and does not require any constrains on the form of $\mathbf{S}(\mathbf{p}(t))$. This can be determined by analytic or other intelligent computational techniques. Therefore, the main purpose of the transformation is *not to extract the TSK model from the training data*, but rather to convert various different identified model forms to a common form of (5).
- IP functions with training data. Its main objective is *to approximate the sampled real system in the from of (5) from the training data set.*

6.3 Characteristic of the resulting membership functions

- In the case of TPT the shape and type of the antecedent fuzzy sets are defined by the TPT method according to the application system under investigation, it cannot be predefined by the user. However, the parameter "*option:*{*Minimal,Ruspini*}" in (7) let us set constrains on the antecedent fuzzy sets. "Minimal" means that the minimum number of basis function is applied on all dimensions in the sense that the equality:

$$\underset{n}{\mathcal{B}} \otimes \mathbf{w}_n(p_n) = \underset{n}{\mathcal{C}} \otimes \mathbf{v}_n(p_n),$$

  where $w_{n,i_n}(p_n)$ and $\mathcal{B} \in \mathcal{R}^{I_1 \times I_2 \times .. \times I_N}$ are resulted by TPT, has no solution for basis $v_{n,i_n}(p_n)$ and tensor $\mathcal{B} \in \mathcal{R}^{J_1 \times J_2 \times .. \times J_N}$ if $\exists n : J_n < I_n$. Note that the resulting basis functions $w_{n,i_n}(p_n)$ cannot be interpreted as fuzzy sets in general since they are bounded by $[-1, 1]$. "*Ruspini*"-option means that the resulting basis functions fulfill the conditions of *Ruspini*-partition (6). In this case the basis functions can, hence, be viewed as membership functions of fuzzy sets. With this option the number of basis functions may increase with one on the dimensions from the "Minimal" basis. The resulting membership functions are in numeric form, the analytic form is not provided. It is rather complicated to find the analytic form via curve-fitting techniques, for instance see Figure 5. We also should remark here that the resulting fuzzy sets do not have nice transparency of the system.
- IP results in membership functions whose analytic type is defined by the user. It hence means that the real system is represented over the transparency predefined.

Let the IP be executed on the valve actuator described in Section 3, with "mf_type"= Double Gaussian membership functions and with $R = [1, 10]$. If 10 rules are allowed then one obtains membership functions depicted on Figure 3. If 5 rules are allowed then one obtains sets depicted in 4.

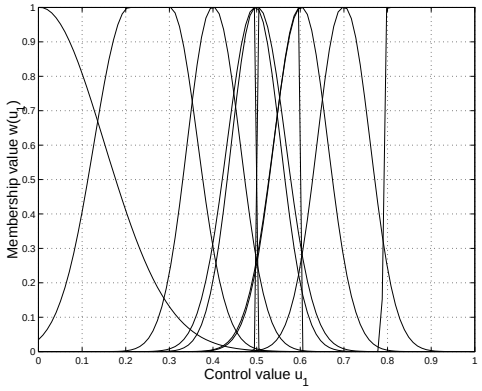

Figure 3: Result of the Identification Package for 10 rules

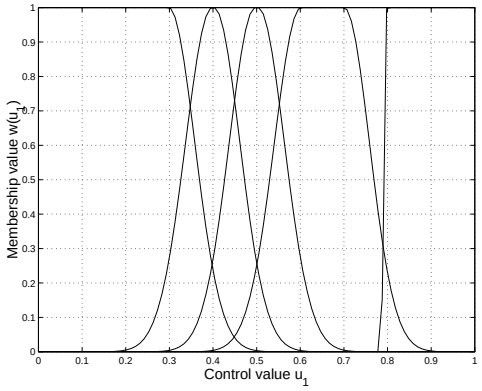

Figure 4: Result of the Identification Package for 5 rules

We can observe that the type of the antecedent fuzzy sets are kept on both Figures 3 and 4, only their parameters are modified.

Let the 10 rules resulted by the IP be converted to *Ruspini*-partition by the TPT. The result is depicted on Figure 5).

When we compare Figures 3 and 5), we can see that the antecedent fuzzy sets resulted by the TPT have not so nice transparency.

One of the most important conclusions of the comparison can also be drawn here. If there exists an exact representation by (5) of the system with a finite number of rules then the TPT certainly finds it. It is capable of revealing and extracting the tensor product structure of the system, and determining the shape of antecedents, accordingly. However, the IP can find the exact representation only if the user somehow hits upon the proper

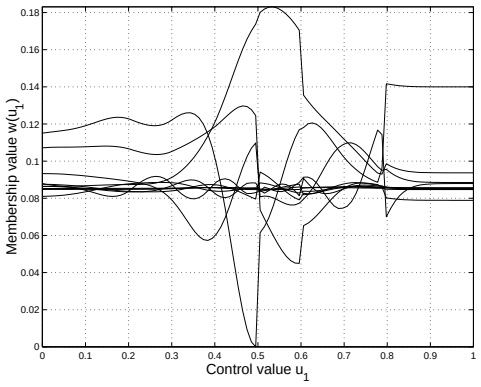

Figure 5: Result of the TPT for 10 rules

shape of the sets, which is usually by chance if we have no sufficient information about the tensor product structure of the system, which is usually the case. Therefore, the IP cannot generally find the exact representation with a finite number of rules. Consequently, TPT is capable of approximating the system without error if the system has tensor product structure, while the IP approach practically cannot. For example, the TPT is capable of exactly (without approximation error) decomposing the function $y = f(x_1, x_2) = (1 + x_1^{-2} + x_2^{-1.5})^2, \Omega : x_1 \in [1, 5], x_2 \in [1, 5]$ into 9 rules, see [3]. The IP is not able to find 9 rules with zero approximation error if we set the antecedent fuzzy sets, e.g. to be Gaussian type, since the function $f(x_1, x_2)$ does not involve Gaussian structure in its analytic form. In the case of IP the question is how to predefine the antecedent structure of the function $f(x_1, x_2)$.

6.4 Immediate link to PDC design

- PDC analysis and design techniques assumes the *"Ruspini"*-partitions of the antecedent fuzzy sets. The benefit of the "$Ruspini$"-option of TPT is that the PDC design can be immediately be executed on the resulting fuzzy rule base. The previous subsection concluded that TPT can find an exact representation. This property may play a crucial role in PDC design when an exact representation (or very small error) of the model is needed. [3] gives examples about dynamic systems that are represented exactly by the TSK model as a consequence of applying the TP transformation. The direct link between TPT and the PDC structure facilitates the design of automatic self-developing methods, see the examples of [3].

- The antecedent fuzzy sets by IP fits the requirement of PDC only if we set "mf_type" accordingly to (6). For instance the antecedent fuzzy sets on Figure 3 and 4 do not fulfill (6), they need further transformations, which are generally not trivial and may unnecessarily increase the number of rules, see [3].

6.5 Trade-off between approximation complexity and accuracy

Both of the identification techniques serve the approximation trade-off via determining a relation between the number of rules and the approximation error. This helps us with reducing the fuzzy rule base complexity via discarding rules according to an acceptable reduction error.

- TPT assigns one weighting value to each antecedent fuzzy set. When we discard the antecedent fuzzy sets, the reduction error is bounded by the sum of the weighting values assigned to the discarded antecedents.

- IP also indicates the relation between the number of rules and the modelling accuracy.

Figure 6 shows the "trade-off relation" offered by the IP. By the help of Figure 6 we can easily optimize the approximation trade-off. In order, to perform the comparison on the same approximated model let the TPT be executed on the 10 rules resulted by the IP. Figure 7 presents the "trade-off" relation offered by the TPT. Figure 7 shows two curves. The upper one is the estimated error bound during the transformation, whilst the lower one is the actual reduction error. We can conclude that the two "trade-off relations" defined by the two identification techniques significantly differ. **In the present case the IP is rather advantageous.**

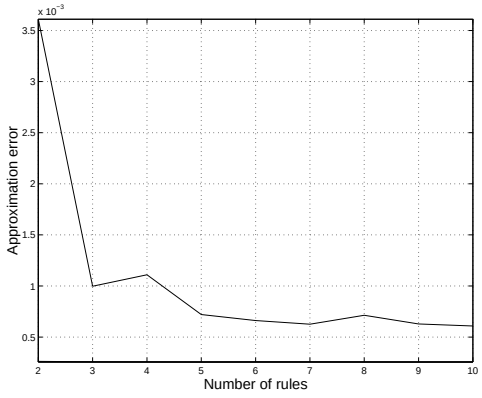

Figure 6: Trade-off by the Identification Package

In the following, we outline the reason of this significant difference. The core of the TPT is the Higher Order Singular Value Decomposition (HOSVD). The rank of a given tensor can be decreased via HOSVD. It is simply done by discarding singular values. The error, in a least-squares sense, between the given tensor and its lower ranked variant is bounded by the sum of the discarded singular values. This property is utilised in a large variety of complexity reduction techniques as well as in the TPT. One important point should be noted here. It is not guaranteed by the HOSVD that the lower ranked variant is the possible best approximation of the given tensor subject to the lower rank constrain even in the case when the smallest singular value is discarded, except in the two-dimensional case. This

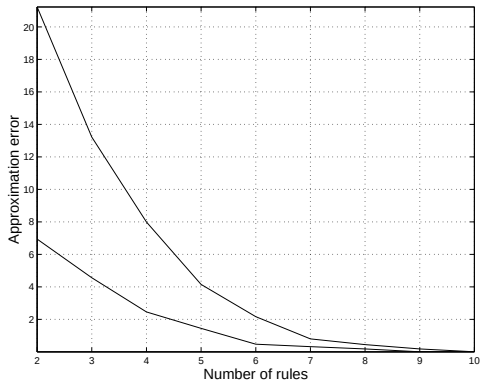

Figure 7: Trade-off by the TPT

means that we can find a tensor with the same lower rank that is closer to the given tensor in the sense of least-squares error. This fact explains why the trade-off relation of the TPT differs. The retraining of the TP transformed rule base yields an approximation error that is numerically equivalent to the error of the IP.

7 Conclusion

In this paper we have compared two TSK fuzzy model identification techniques in readiness for application to an FDI task in a system which has model-complexity. We can first conclude that although these two approaches have some common modelling aims and shows similar features they have significantly different modelling roles and cannot be mutually replaced. They can however be linked, since the purpose of IP is to extract the identified model from the training data, whilst the TPT has the role of transforming an already identified model into the TSK fuzzy model form with various advantageous properties. It has been pointed out that if we utilise the TSK fuzzy model-based approach to FDI we should face the approximation trade-off in the modelling phase. It has also been concluded that the IP generally offers advantageous approximation trade-off. The TPT may need further training and it has also been shown that it can certainly find the exact representation with a finite number of rules, whenever the TPT exists. If we would like to extract transparency from the rule base then the use of the IP technique is suggested in this paper. A significant conclusion in the sense of FDI design is that the use of further PDC-based analysis is well supported by TPT. Having these conclusions, we may be motivated to extend the TPT with the clustering and the retraining phase of IP, thus bringing together their joint advantages. Future work may focus on the development of an identification method that starts with training data and results in a fuzzy rule base whereupon PDC can readily be executed. This approach would be of value, for example for the multiple-model strategy required for FDI of a very non-linear actuator system as typified by the case of the sugar factory study.

Acknowledgements: Funding support for this work under the EU Fifth Framework Programme Research Training Network contract HPRTN-CT-2000-00110 DAMADICS. Fruitful discussions with DAMADICS partners also gratefully acknowledged.

References

[1] R. J. Patton, F. J. Uppal, and C. J. Lopez-Toribio. Soft computing approaches to fault diagnosis for dynamic systems: A survay. *Proc. of 4th* IFAC *Symposium* SAFEPROCESS *2000, Budapest*, 1:298–311, 2000.

[2] J. M. F. Calado, J. Korbicz, K. Patan, R. J. Patton, and J. S da Costa. Soft computing approaches to fault diagnosis for dynamic systems. *European Journal of Control*, 7(2-3):248–286, july 2001.

[3] P. Baranyi. HOSVD based TP model transformation as a way to LMI based controller design. *IEEE Transaction on Industrial Electronics (in press 2003)*.

[4] J. Chen and R. J. Patton. *Robust model-based ault diagnosis for dynamic systems*. ISBN: 0-7923-8411-3. Kluwer Academic, 1999.

[5] F. J. Uppal, R. J. Patton, and V. Palade. Neuro fuzzy based fault diagnosis applied to an electro-pneumatic valve. *15th Triennial World Congress of the international Federation of Automatic Control*, 2002.

[6] F. J. Uppal, R. J. Patton, and M. Witczak. A hybrid neuro-fuzzy and de-coupling approach applied to the damadics benchmark problem. *Proc. of the 5th IFAC Symposium* SAFEPROCESS *2003, Washington DC*, 2003.

[7] K. Tanaka and H. O. Wang. *Fuzzy Control Systems Design and Analysis - A Linear Matrix Inequality Approach*. Hohn Wiley and Sons, Inc., 2001, 2001.

[8] M. Syfert, R. J. Patton, M. Bartys, and J. Quevedo. Development and application of methods for actuator diagnosis in industrial control systems (damadics). *Proc. of the 5th IFAC Symposium*SAFEPROCESS *2003, Washington DC*, 2003.

[9] S. Simani, C. Fantuzzi, and R. J. Patton. *Model-based Fault Diagnosis in Dynamic Systems using Identification Techniques*. Advances in Industrial Control, ISBN: 1-85233-685-4. Springer, 2002.

[10] M. Mizumoto. *Fuzzy controls by Product-sum-gravity method*. Eds. Liu and Mizumoto, Advancement of Fuzzy Theory and Systems in China and Japan, International Academic Publishers, 1990. c1.1-c1.4.

[11] R. Babuska. *Fuzzy modelling in control*. Kluwer Academic Publishers, 1998.

[12] O. Nelles and R. Isermann. Basis function networks for interpolation of local linear models. *IEEE Conference on Decision and Control (CDC)*, pages 470–475, 1996.