s c c h
software competence center
hagenberg

FUZZY.LOGIC.LAB.LINZ

Abstracts of the FLLL/SCCH Master and PhD Seminar

Room 010, Software Park Hagenberg

February 8, 2005

Software Competence Center Hagenberg
Hauptstrasse 99
A-4232 Hagenberg
Tel. +43 7236 3343 800
Fax +43 7236 3343 888
www.scch.at

Fuzzy Logic Laboratorium Linz-Hagenberg
Hauptstrasse 99
A-4232 Hagenberg
Tel. +43 7236 3343 431
Fax +43 7236 3343 434
www.flll.jku.at

Program

Session 1 (Chair: Susanne Saminger) 9:00–10:30

9:00 Petr Cintula, Libor Behounek
Towards Logical Calculus for Fuzzy Mathematics I

9:30 Petr Cintula, Libor Behounek
Towards Logical Calculus for Fuzzy Mathematics II

10:00 Peter Sarkoci
Positive Subhomogenity and Domination of t-norms

10:30 Coffee Break

Session 2 (Chair: Thomas Natschläger) 10:45–12:45

10:45 Milan Petrik
Memory Elements in Many-valuedd Logical Circuits

11:15 Kristen A. McIntyre, Leila Muresan
Simple Rules for the Creation of Robust, Efficient Network Topologies

11:45 Bettina Heise
Introduction to DIC microscopy and DIC image reconstruction

12:15 Edwin Lughofer
Data-Driven Incremental Learning of Fuzzy Systems Part 3: Application Cases for FLEXFIS

Towards A Logical Calculus for Fuzzy Mathematics I, II

Libor Běhounek Petr Cintula
Institute of Computer Science, Academy of Sciences of the Czech Republic
{behounek|cintula}@cs.cas.cz

The aim of the talk is to give an introduction to higher-order fuzzy logic and show a sketch of a natural-deduction-style calculus for making proofs in formal fuzzy mathematics within its framework. We first reproduce the basic definitions of higher-order fuzzy logic. For details, see [1].

Definition 1 (Henkin-style second-order fuzzy logic)
Let $\mathcal{F}$ be a fuzzy logic which extends BLΔ*. The Henkin-style second-order fuzzy logic over $\mathcal{F}$ is a theory over multi-sorted first-order $\mathcal{F}$ with (crisp) equality and sorts for objects (lowercase variables) and classes (uppercase variables). Both of the sorts subsume subsorts for n-tuples, for all $n \geq 1$. Apart from the obvious necessary function symbols and axioms for tuples (tuples equal iff their respective constituents equal), the only primitive symbol is the membership predicate $\in$ between objects and classes. The axioms for $\in$ are the following:*

1. *The comprehension axioms: $y \in \{x \mid \varphi(x)\} \leftrightarrow \varphi(y)$, where φ may contain any parameters and comprehension terms.*
2. *The extensionality axiom: from $(\forall x)(x \in X \leftrightarrow x \in Y)$ infer $X = Y$.*

Though admitting non-standard models (to achieve axiomatizability), Henkin-style second-order logic is a good axiomatic approximation of Zadeh's [5] notion of fuzzy set.

Notice that unless the formula φ expresses a crisp condition, the set term $\{x \mid \varphi(x)\}$ should not be read "the set of all those x for which φ holds", but rather "the (fuzzy) set to which any x belongs in the same degree in which φ is true about x".

Convention 2 *The usual precedence of connectives is assumed. Furthermore, we introduce the following abbreviations:*

$x \in A$	Ax
$\langle x_1, \ldots, x_k \rangle \in R$	$Rx_1 \ldots x_k$
$\neg(x \in A)$	$x \notin A$
$(\forall x)(x \in A \rightarrow \varphi)$	$(\forall x \in A)\varphi$

$(\exists x)(x \in A \,\&\, \varphi)$	$(\exists x \in A)\varphi$
$\{x \mid x \in A \,\&\, \varphi\}$	$\{x \in A \mid \varphi\}$
$\Delta(\varphi \leftrightarrow \psi)$	$\varphi = \psi$
$\Delta(\varphi \rightarrow \psi)$	$\varphi \leq \psi$, *etc.*

The defined connective $=$ should not be mistaken for the identity predicate. Membership degrees and truth values are referred to solely by means of formulae; there are no variables for truth degrees in $\mathcal{F}_2$.

Definition 3 (Henkin-style higher-order fuzzy logic)
Henkin-style fuzzy logic of higher orders is obtained by repeating the previous definition on each level of the type hierarchy. Obviously, defined symbols of any type can then be shifted to all higher types as well. (Consequently, all theorems are preserved by uniform upward type-shifts.) Types may be allowed to subsume all lower types.

Henkin-style fuzzy logic $\mathcal{F}$ of order n will be denoted by $\mathcal{F}_n$, the whole hierarchy by $\mathcal{F}_\omega$. The types of terms are either denoted by a superscripted parenthesized type (e.g., $X^{(3)}$), or understood from the context.

It should be stressed that despite the name, Henkin-style higher-order fuzzy logics are *theories* over *first-order* fuzzy logics (see [4]).

All usual notions of elementary fuzzy set theory and elementary theory of fuzzy relations can be defined in $\mathcal{F}_2$. Notice that in $\mathcal{F}_2$, these defined notions are generally *graded* rather than crisp.

Definition 4 (Fuzzy class operations and relations)

$\emptyset$	$=_{\mathrm{df}}$	$\{x \mid 0\}$	*empty class*
V	$=_{\mathrm{df}}$	$\{x \mid 1\}$	*universal class*
$\mathrm{Ker}(X)$	$=_{\mathrm{df}}$	$\{x \mid Ax = 1\}$	*kernel*
$\mathrm{Supp}(X)$	$=_{\mathrm{df}}$	$\{x \mid Ax > 0\}$	*support*
$\setminus X$	$=_{\mathrm{df}}$	$\{x \mid x \notin X\}$	*complement*
$X \cap Y$	$=_{\mathrm{df}}$	$\{x \mid x \in X \,\&\, x \in Y\}$	*intersection*
$X \cup Y$	$=_{\mathrm{df}}$	$\{x \mid x \in X \veebar x \in Y\}$	*union*
$X \setminus Y$	$=_{\mathrm{df}}$	$\{x \mid x \in X \,\&\, x \notin Y\}$	*difference*

Definition 5 (Fuzzy class operations and relations)

$\mathrm{Hgt}(X)$	$\equiv_{\mathrm{df}}$	$(\exists x)(x \in X)$	*height*
$\mathrm{Norm}(X)$	$\equiv_{\mathrm{df}}$	$(\exists x)\Delta(x \in X)$	*normality*
$\mathrm{Crisp}(X)$	$\equiv_{\mathrm{df}}$	$(\forall x)\Delta(x \in X \veebar x \notin X)$	*crispness*
$\mathrm{Fuzzy}(X)$	$\equiv_{\mathrm{df}}$	$\neg\,\mathrm{Crisp}(X)$	*fuzziness*
$X \subseteq Y$	$\equiv_{\mathrm{df}}$	$(\forall x)(x \in X \rightarrow x \in Y)$	*inclusion*
$X \approx Y$	$\equiv_{\mathrm{df}}$	$(\forall x)(x \in X \leftrightarrow x \in Y)$	*equality*
$X \parallel Y$	$\equiv_{\mathrm{df}}$	$(\exists x)(x \in X \,\&\, x \in Y)$	*compatibility*

If the underlying logic $\mathcal{F}$ is strong enough (e.g., ŁΠ, see [3]), then $\mathcal{F}_\omega$ is strong enough to harbour a large part of fuzzy mathematics. (The logic ŁΠ

contains all connectives of t-norm logics PC($*$), where $*$ is a finite sum of G, Ł, and Π). Due to a metatheorem of [1], all classical theories of order $n \in \omega$ can be interpreted in $\mathcal{F}_\omega$; thus all usual classical structures (metrics, measure, numbers) are available in $\mathcal{F}_\omega$.

The usual Hilbert-style calculus is rather cumbersome for actual proofs of mathematical theorems in $\mathcal{F}_\omega$ (due to the lack of the classical deduction theorem). Therefore we develop a more convenient calculus which would enable easier proofs and would be closer to the usual proofs of classical mathematics. We give here only a sketch of the system; the set of derived rules which are useful for the work in fuzzy mathematics listed here is far from complete (the work is in progress). The presented calculus is the natural-deduction style, though not normalized to the introduction and elimination rules for all connectives (this would hardly be convenient for actual proofs, as well as hardly attainable due to the proof-theoretical properties of logics with prelinearity).

Definition 6 (Sequents) *The expression $\varphi \Rightarrow \psi$ is the sequent of our calculus which corresponds to the formula $\varphi \to \psi$. The comma sign is used in sequents as an abbreviation of the strong conjunction $\&$. The sequents $\Rightarrow \psi$ and $\varphi \Rightarrow$ mean $1 \Rightarrow \psi$ and $\varphi \Rightarrow 0$, respectively. Multisets of (sub)formulae in sequents will be denoted by uppercase letters $A, B, C \ldots$ and the sequents by the letters $\Phi, \Psi, \Xi, \ldots$ (possibly with indices).*

The proof in the calculus is defined as usual, i.e., as a sequence of sequents, each of which is an axiom or the result of an application of a valid rule of inference (listed below) to the previously proved sequents. The rules of inference are written as $\Phi_1 \ \ldots \ \Phi_n \ / \ \Psi$.

Theorem 7 (The rules of inference) *The following list gives examples of the rules of inference which are provably valid in* $\mathrm{BL}\Delta_\omega$*:*

- *Cut:* $A_1, B \Rightarrow A_2 \ A_2, C \Rightarrow A_3 \ / \ A_1, B, C \Rightarrow A_3$
- *Conjunction:* $A_1 \Rightarrow B_1 \ A_2 \Rightarrow B_2 \ / \ A_1, A_2 \Rightarrow B_1, B_2$
- *Generalization:* $A \Rightarrow B \ / \ (\forall x)A \Rightarrow (\forall x)B$
- *Existential generalization:* $A, B \Rightarrow C \ / \ (\forall x)A, (\exists x)B \Rightarrow (\exists x)C$
- *Deduction:* $A \Rightarrow B \ / \ \Rightarrow A \to B$
- *Delta necessitation* $A \Rightarrow B \ / \ \Delta A \Rightarrow \Delta B$*, etc.*

Often it is instructive to read the rules from the bottom up, i.e., a rule Φ/Ψ as "in order to prove Ψ it is sufficient to prove Φ".

The proofs using this proof system can be done in a similar manner one works in classical mathematics. The resemblance can be even enhanced if the formulae are rephrased in natural language, wherein the connectives are interpreted as the connectives of fuzzy logic (e.g., 'if φ then ψ' is to be understood as 'the more φ, the more ψ). The theorems then look like classical statements about crisp sets;

however, since the sets can be *fuzzy*, the rules of fuzzy logic (as reflected in our calculus) are applied in their proofs. This approach conforms the methodology of [2], according to which speaking of fuzzy (rather than crisp) sets requires the rules of inference of fuzzy (rather than classical) logic, but in other respects is not different from reasoning in classical logic.

References

[1] L. Běhounek and P. Cintula (2004) "Fuzzy class theory", to appear in Fuzzy Sets and Systems.

[2] Libor Běhounek and Petr Cintula. "From fuzzy logic to fuzzy mathematics: A methodological manifesto." Submitted, 2004.

[3] F. Esteva, L. Godo and F. Montagna (2001) "The ŁΠ and ŁΠ$\frac{1}{2}$ logics: Two complete fuzzy systems joining Łukasiewicz and product logics", Arch. Math. Logic, 40:39–67.

[4] P. Hájek (1998) Metamathematics of Fuzzy Logic, Kluwer, Dordercht.

[5] L. Zadeh (1965) "Fuzzy sets", Information and Control 8:338–353.

Positive Subhomogenity and the Domination of t-norms

Peter Sarkoci
STU Bratislava, Radlinského 9, 81237 Bratislava, Slovakia
E-mail: peter.sarkoci@stuba.sk

Preliminaries

Motivations to study domination comes from different branches of mathematics. The notion was introduced within the framework of probabilistic metric spaces where it plaays an important role in construction of cartesian products of probabilistic metric spaces [14]. Later the domination of t-norms was studied in connection with construction of different types of fuzzy relations [1–3, 16]. The concept of domination was further extended to the much more general class of aggregation operators [11]. The domination of aggregation operatros becomes interesting when seeking aggregation procedures which preserves T-transitivity of aggregated fuzzy relations [11] or aggregation procedures which preserves extensionality of fuzzy sets with respect to given T-equivalence relation [12]. The most general definition of domination requires arbitrary operations defined on poset [4].

Definition 1 *Let $(P, \geq)$ be a poset and let $A\colon P^m \to P$, $B\colon P^n \to P$ be two operations defined on P with arity m and n, respectively. We say that A dominates B ($A \gg B$ in symbols) if each matrix $(x_{i,j})$ of type $m \times n$ over P satisfies the inequality*

$$\begin{aligned} &A(B(x_{1,1}, x_{1,2}, \ldots, x_{1,n}), \ldots, B(x_{m,1}, x_{m,2}, \ldots, x_{m,n})) \quad \geq \\ &B(A(x_{1,1}, x_{2,1}, \ldots, x_{m,1}), \ldots, A(x_{1,n}, x_{2,n}, \ldots, x_{m,n})). \end{aligned}$$

Recall that a t-norm [9, 14] is a monotone, associative and commutative binary operation $T\colon [0,1]^2 \to [0,1]$ with neutral element 1. Important examples of t-norms are: the minimum $T_{\mathbf{M}}$, the product $T_{\mathbf{P}}$, the Łukasiewicz t-norm $T_{\mathbf{L}}$ and the drastic t-norm $T_{\mathbf{D}}$ given by

$$\begin{aligned} T_{\mathbf{M}}(x,y) &= \min(x,y), \\ T_{\mathbf{P}}(x,y) &= xy, \\ T_{\mathbf{L}}(x,y) &= \max(0, x+y-1), \\ T_{\mathbf{D}}(x,y) &= \begin{cases} xy & \max(x,y) = 1 \\ 0 & \text{otherwise} \end{cases}. \end{aligned}$$

The additive generator is each continuous and strictly decreasing function $f\colon [0,1] \to [0,\infty]$ with $f(1) = 0$. The binary operation $T\colon [0,1]^2 \to [0,1]$ given

by

$$T(x,y) = f^{-1}\left(\min(f(0), f(x) + f(y))\right) \tag{1}$$

is a continuous archimedean t-norm generated by f [9]. Continuous archimedean t-norm is strict if its additive generator is unbounded, otherwise the t-norm is nilpotent. In our talk we restrict to strict t-norms.

We say that a t-norm T_1 is stronger than a t-norm T_2 ($T_1 \geq T_2$ for short) if any $x, y \in [0,1]$ satisfy $T_1(x,y) \geq T_2(x,y)$. We use the notation $T_1 > T_2$ whenever simultaneously $T_1 \geq T_2$ and $T_1 \neq T_2$ hold. One can easily show that each t-norm is weaker than $T_{\mathbf{M}}$ and stronger than $T_{\mathbf{D}}$ and that $T_{\mathbf{M}} > T_{\mathbf{P}} > T_{\mathbf{L}} > T_{\mathbf{D}}$ holds. For two strict t-norms T_1 and T_2 with additive generators f_1 and f_2 respectively $T_1 \geq T_2$ holds if and only if the function $f_1 \circ f_2^{-1}$ is superadditive [9].

By Definition 1 we have that two t-norms T_1 and T_2 satisfy $T_1 \gg T_2$ iff for each $x, y, u, v \in [0,1]$

$$T_1(T_2(x,y), T_2(u,v)) \geq T_2(T_1(x,u), T_1(y,v)). \tag{2}$$

It is easy to show that each t-norm T satisfies $T_{\mathbf{M}} \gg T$, $T \gg T_{\mathbf{D}}$ and $T \gg T$. Moreover, by [9,13], the representative t-norms $T_{\mathbf{P}}$ and $T_{\mathbf{L}}$ satisfy $T_{\mathbf{P}} \gg T_{\mathbf{L}}$. If $T_1 \gg T_2$ then by inequality (2), the neutrality of 1 and the commutativity of t-norms we have that any $y, u \in [0,1]$ satisfy

$$\begin{aligned} T_1(y,u) = {} & T_1(T_2(1,y), T_2(u,1)) \geq \\ & \geq T_2(T_1(1,u), T_1(y,1)) = T_2(u,y) = T_2(y,u) \end{aligned}$$

so that $T_1 \geq T_2$, see [9]. Thus $T_1 \geq T_2$ is a necessary condition for $T_1 \gg T_2$; we may also say that domination is a subrelation of $\geq$. The converse implication does not hold. Since the domination is a subrelation of $\geq$ it is antisymmetric. The old open problem [14, Problem 12.11.3] is whether domination is transitive on the set of all t-norms. If it were true domination would be a partial order.

When inspecting domination, the tool of φ-transform can be helpful. Let φ be an order isomorphism of the interval $[0,1]$ and let T be an arbitrary t-norm. Define $T_\varphi \colon [0,1]^2 \to [0,1]$ by

$$T_\varphi(x,y) = \varphi^{-1}\left(T(\varphi(x), \varphi(y))\right)$$

to be the φ-transform of T. It is easy to show that T_φ is again a t-norm [9]. Moreover, for arbitrary t-norms T_1 and T_2 and for arbitrary order isomorphism φ the satisfaction of $T_1 \gg T_2$ is equivalent to $(T_1)_\varphi \gg (T_2)_\varphi$ so that φ-transforms preserve domination [11]. It is well known that a t-norm is strict (nilpotent) iff there exists φ such that $T = (T_{\mathbf{P}})_\varphi$ ($T = (T_{\mathbf{L}})_\varphi$) [9]. Moreover, it is clear that each φ-transform of a strict (nilpotent) t-norm is again strict (nilpotent). Thus in order to characterize pairs of dominated by strict (nilpotent) t-norms it suffices to characterize strict (nilpotent) t-norms dominated by $T_{\mathbf{P}}$ ($T_{\mathbf{L}}$).

Let T_1, T_2 be t-norms with additive generators f_1 and f_2 respectively. Define new mapping $h = f_1 \circ f_2^{-1}$. It can be shown [9] that T_1 dominates T_2 iff any $x, y, u, v \in \mathbb{R}_0^+$ satisfy the following inequality

$$h^{-1}(h(x+u) + h(y+v)) \leq h^{-1}(h(x) + h(y)) + h^{-1}(h(u) + h(v)) \tag{3}$$

The inequality 3 is known as Mulholland inequality. In 1950 H. P. Mulholland [10] proved that if h and $\log \circ h \circ \exp$ are convex functions, then h satisfies 3.

Later in 1984 R. M. Tardiff [15] proved that if h is differentiable, convex and such that $\log \circ h' \circ \exp$ is convex, then h satisfies 3. Recently it was shown that the Tardiff condition implies the Mulholland [8]. The question whether Mullholand condition is also necessary is still open. It can be shown [9] that the domination is transitive on the set of all archimedean t-norms iff the set of all solutions of Mulholland inequality is closed with respect to the composition of mappings.

Domination by $T_{\mathbf{P}}$

We will restrict our considerations to domination of strict t-norms. First we show one necessary condition for a t-norm dominated by $T_{\mathbf{P}}$.

Definition 2 *Let $f\colon \mathbb{R} \to \mathbb{R}$ be a real function. We say that f is positively subhomogenous if each $x \in \mathbb{R}$ and each $\alpha \in [0,1]$ satisfy $\alpha f(x) \geq f(\alpha x)$.*

Proposition 3 Let T be a strict t-norm such that $T_{\mathbf{P}} \gg T$. Then each vertical section of T is positively subhomogenous.

Next we characterize all strict t-norms which fulfill this property by means of additive generators.

Proposition 4 Let T be a t-norm with additive generator f. Each vertical section of T is positively subhomogenous if and only if $-\log \circ f$ is a convex function.

Finally we inspect what does this result mean for Mulholland's inequality.

Corollary 5 If an order isomorphism of positive real numbers solves the Mulholland's inequality, then it is convex.

References

[1] U. Bodenhofer: *A Similarity-Based Generalization of Fuzzy Orderings*, Volume C 26, Schriftereihe der Johannes-Kepler-Universität Linz, Universitätsverlag Rudolf Trauner, Linz (1999)

[2] B. De Baets, R. Mesiar: *Pseudo-metrics and T-equivalences*, J. Fuzzy Math. 5 (1997) 471–481

[3] B. De Baets, R. Mesiar: *T-partitions*, Fuzzy Sets and Systems 97 (1998) 211–223

[4] J. Drewniak, P. Drygaś, U. Dudziak: *Relation of Domination*, In: FSTA 2004 Abstracts (2004) 43–44

[5] M. J. Frank: *On the Simultaneous Associativity of $F(x,y)$ and $x + y - F(x,y)$*, Aequationes Math. 19 (1979) 194–226

[6] H. Hamacher: *Über logische Verknüpfungen unscharfer Aussagen und deren zugehörige Bewertungsfunktionen*, Progress in Cybernetics and Systems Research, Hemisphere Publ. Comp., New York (1975) 276–287

[7] H. Hamacher: *Über logische Aggregationen nicht-binär explizierter Entscheidungskriterien*, Rita G. Fischer Verlag, Frankfurt (1978)

[8] W. Jarczyk, J. Matkowski: *On Mulholland's Inequality*, Proc. Amer. Math. Soc. 130 (2002) 3243–3247

[9] E. P. Klement, R. Mesiar, E. Pap: *Triangular Norms*, Kluwer (2000)

[10] H. P. Mulhollaand: *On Generalizations of Minkowski's Inequality in the Form of a Triangle Inequality*, Proc. London Math. Soc. (2) 51 (1950) 294–307

[11] S. Saminger, R. Mesiar, U. Bodenhofer: *Domination of Aggregation Operators and Preservation of Transitivity*, Internat. J. Uncertain. Fuzziness Knowledge-Based Systems 10 (2002) 11–35

[12] S. Saminger: *Aggregation of Extensional Hulls and Domination*, In: FSTA 2004 Abstracts (2004) 92–93

[13] H. Sherwood: *Characterizing Dominates on a Family of Triangular Norms*, Aequationes Math. 27 (1984) 255–273

[14] B. Schweizer, A. Sklar: *Probabilistic Metric Spaces*, North-Holland (1983)

[15] R. M. Tardiff: *On a Generalized Minkowski Inequality and its Relation to Dominates for t-norms*, Aequationes Math. 27 (1984) 308–316

[16] L. Valverde: *On the structure of F-indistinguishability operators*, Fuzzy Sets and Systems 17 (1985) 313–328

Memory elements in many-valued logical circuits

Milan Petrík *
Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
petrikm@cmp.felk.cvut.cz

February 6, 2005

Abstract

Memory elements play a crucial role in the hardware design as parts of logical circuits which are able to remember a logical value and to hold it independently on the input for a defined period of time. Thanks to them we may work with an inner state in logical circuits, construct registers, synchronize combinational circuits, etc.

When generalizing the design of hardware to the many-valued logic, the question of many-valued memory elements becomes inevitable. In this paper we present a many-valued memory element which is based on a generalization of the R-S memory element known from the two-valued logic. We show that the circuit works correctly for finitely many logical values and this number may be arbitrary depending on the potentialities of the implementation. We also show that the structure of the memory element is independent on the number of the logical values.

The memory element we present here is "level controlled".

Keywords: many-valued logic, fuzzy logic, hardware design, logical circuit design, sequential logic, memory element, latch, flip-flop.

1 Introduction

A *memory element*[1] is a logical circuit able to remember a logical value and to hold it on its output for a defined period of time independently of its input.

In Figure 1 you can see a scheme of a general memory element. Input signal T (called usually *clock* or *time* signal) determines whether the memory element is *open* (the value of input D is "remembered" and shall appear on output Q) or it is *closed* (the old value of output Q is kept ignoring input D).

Two groups of memory elements are distinguished: *latches* and *flip-flops*. Latches (level-controlled memory elements) are open or closed depending on the value of the input T while flip-flops (edge-controlled memory elements) are

* Work supported by CEEPUS net SK-042.
[1] See e.g. [1, 3, 2].

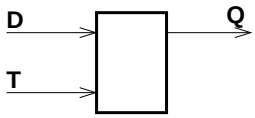


Figure 1: Scheme of a general memory element.

open when input T changes its value from one defined value to another defined value otherwise they are closed. Although flip-flops have better properties, in this paper we are interested in many-valued latches since their construction is simpler. The construction of flip-flops usually requires latches as building blocks and shall be discussed in a further paper.

In the hardware design we distinguish two categories of logical circuits: combinational logical circuits and sequential logical circuits. Combinational logical circuits are logical circuits where the value of the output depends only on the value of the input. Sequential circuits are logical circuits with history and their output value depends moreover on the inner state of the circuit; a usage of memory elements is crucial in this case.

Another important role played by memory elements in the hardware design are e.g. a construction of registers (fast on-chip memory) or a synchronization of combinational logical circuits (adding memory elements to the output of each combinational sequence to eliminate logical hazards).

2 Many-valued R-S circuit

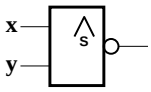


Figure 2: Gate representation of the standard many-valued Sheffer operation.

Definition 2.1 *The standard many-valued* Sheffer operation *is a binary operation* $\overline{\wedge}_S\colon [0,1]\times[0,1]\to[0,1]$ *defined as follows:*

$$\overline{\wedge}_S(a,b) = \overline{a \wedge_S b} = 1 - \min(a,b)$$

Definition 2.2 (many-valued R-S circuit) Many-valued R-S circuit*, an* R-S circuit *for short, is a logical circuit consisting of two gates implementing the standard many-valued Sheffer operation (see Figure 2b), two input ports R, S, and two output ports Q_1, Q_2, connected as shown in Figure 3.*

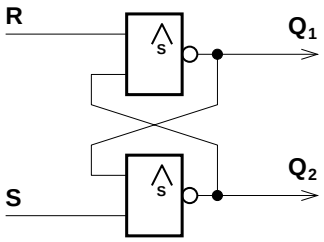


Figure 3: Scheme of a many-valued R-S circuit.

Theorem 2.3 (storing value to an R-S circuit) *The R-S circuit with input ports R, S and output ports Q_1, Q_2 satisfies: If $R \leq \overline{S}$ then $Q'_1 = \overline{R}$ and $Q'_2 = \overline{S}$. In particular, if $R = \overline{S}$ then $Q'_1 = S$ and $Q'_2 = R$. This result is independent of the initial values Q^0_1 and Q^0_2.*

Theorem 2.4 (keeping value in an R-S circuit) *The R-S circuit with input ports R, S and output ports Q_1, Q_2 satisfies: If $Q^0_1 = \overline{Q^0_2}$, $S > Q^0_1$ and $R > Q^0_2$ then $Q'_1 = Q^0_1$ and $Q'_2 = Q^0_2$.*

3 Unstable and invalid states of the many-valued R-S circuit

We have shown configurations of the values of input and output ports which allow us to store or to keep a value in an R-S circuit. Also other configurations may appear, e.g. if a logical circuit is starting and all signals have undefined, zero or other value. In this section we discuss all possible configurations of the values of the ports in an R-S circuit and its behavior as the reaction to it.

Theorem 3.1 *The R-S circuit with input ports R, S and output ports Q_1, Q_2 satisfies: If $R > \overline{S}$ then $\overline{R} \leq Q'_1 \leq S$ and $\overline{S} \leq Q'_2 \leq R$.*

Corollary 3.2 *Since $\overline{R} \leq Q'_1 \leq S$ and $\overline{S} \leq Q'_2 \leq R$ for every R and S (see Theorem 2.3 and Theorem 3.1), we can say that if we store a value in an R-S circuit and $|R - \overline{S}| = |\overline{R} - S| = \Delta$, then $Q_1 \in [S - \Delta, S + \Delta]$ and $Q_2 \in [R - \Delta, R + \Delta]$.*

Corollary 3.3 *If a value is kept in an R-S circuit, $S > Q_1$, $R > Q_2$ but $Q^0_1 \neq \overline{Q^0_2}$ then output Q_1 attains a value from an interval delimited by Q^0_1 and $\overline{Q^0_2}$. Output Q_2 attains a value from an interval delimited by Q^0_2 and $\overline{Q^0_1}$.*

Corollary 3.4 *If any of the values of R, S, Q^0_1 or Q^0_2 has an error, then an error not greater than the sum of these errors may appear on outputs Q_1 and Q_2.*

Until now we worked with gates which gave the exact result. However there is nothing ideal in the real world. Constructing a logical circuit the gates will have an error, i.e. they will give a bigger or smaller result than is the correct

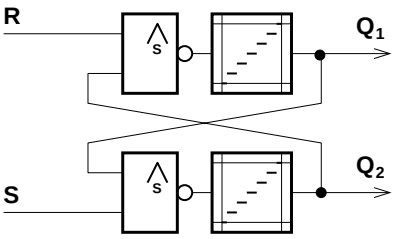


Figure 4: Scheme of a many-valued R-S circuit with added filters; the filters filtrate the output to a finite set of logical values.

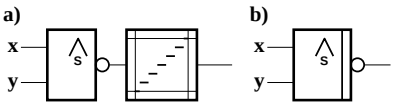


Figure 5: a - standard Sheffer operation filtered to a finite set of logical values, b - schematic symbol of a filtered standard Sheffer operation

result of the standard Sheffer operation they implement. Our R-S circuit shows robust to the cases when the gates have both the same type of defect, i.e. they both give a bigger or a smaller value then it is expected. Nevertheless the case when one gate gives a bigger value and the second a smaller value shows as fatal. In this case one output decreases and the second increases until a bound (0 or 1) is reached. The information kept in the R-S circuit is lost.

This problem is very serious concerning that the gate defect is in general unpredictable and moreover may change during the life of the circuit. A possible solution is to add a filter at the end of both gates which would restrict the values from the interval $[0, 1]$ to a finite set of discrete values spread uniformly on the interval $[0, 1]$. If the distance between two neighbouring values is at least twice the error of the gates then the impact of the gate errors will be eliminated and the functionality of the R-S circuit will not be harmed. Scheme of an R-S circuit with added filters may be seen in Figure 4.

Note that the number of logical values which may be passed out of the filter is in general arbitrary but depends on the errors of the gates.

4 Memory elements based on the many-valued R-S circuit

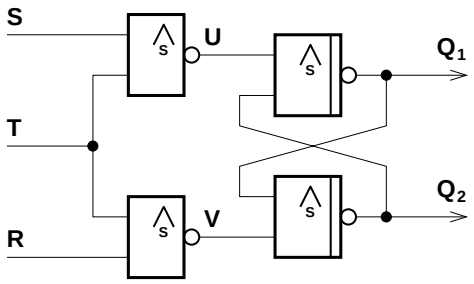


Figure 6: Scheme of a many-valued R-S latch.

Definition 4.1 (many-valued R-S latch circuit) Many-valued R-S latch circuit*, an* R-S latch *for short, is a logical circuit consisting of four gates implementing the standard many-valued Sheffer operation, two of them filtered, three input ports R, S, T and two output ports Q_1, Q_2, connected as shown in Figure 6.*

Theorem 4.2 (storing value to an R-S latch) *The R-S latch with input ports R, S, T and output ports Q_1, Q_2 satisfies: If $R \geq \overline{S}$ and $T = 1$ then $Q'_1 = S$ and $Q'_2 = R$. This result is independent of the previous values of Q_1 and Q_2.*

Theorem 4.3 *The R-S latch with input ports R, S, T and output ports Q_1, Q_2 satisfies: If R and S are constant, $R = \overline{S}$ and T is changing its value from 1 to 0, then $Q_1 = S$ and $Q_2 = R$ for the whole period of the process until $T = 0$.*

Theorem 4.4 (keeping value in an R-S latch circuit) *The R-S latch with input ports R, S, T and output ports Q_1, Q_2 satisfies: If $T = 0$ and $Q_1 = \overline{Q_2}$ then Q_1 and Q_2 keep their values, i.e. $Q'_1 = Q^0_1$ and $Q'_2 = Q^0_2$, independently of the values of inputs R and S until the value of input T is greater than 0.*

Negating input R in an R-S latch by a gate implementing the standard negation (see Figure 7) we get a *D latch* as shown in Figure 8. It passes the value of input D to the output Q (i.e. $Q'_1 = D$ and $Q'_2 = \overline{D}$) when the value of input T is 1; when the value of input T is 0, the circuit keeps its old output value (i.e. $Q'_1 = Q^0_1$ and $Q'_2 = Q^0_2$).

References

[1] J. Bokr and V. Jáneš. *Logical Systems (Logické systémy).* CTU, Prague, 1999. in Czech.

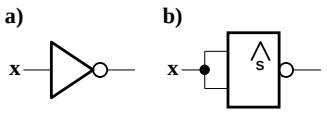


Figure 7: Gate representations of the standard negation: a - standard negation $\overline{x}$, b - standard negation implemented by the standard Sheffer operation $\overline{\underset{s}{\wedge}}(x, x) = \overline{x}$.

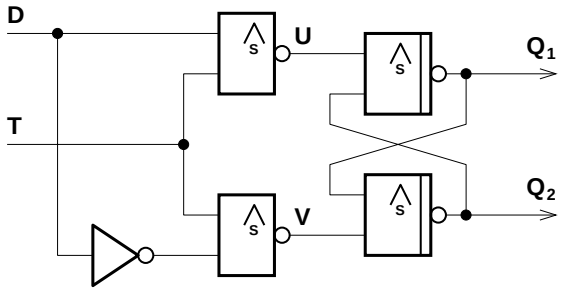


Figure 8: Scheme of a many-valued D latch.

[2] G. E. Hoernes and M. F. Heilweil. *Introduction to Boolean Algebra and Designing of Logical Circuits*. SNTL, Prague, 1969. Czech translation of English original.

[3] H. Kubátová and Z. Blažek. *Logical Systems: Exercises (Logické systémy: Cvičení)*. CTU, Prague, 1999. in Czech.

Simple Rules for the Creation of Robust, Efficient Network Topologies

Kristen A. McIntyre, Sun Microsystems Laboratories
Leila Muresan, Johannes Kepler University

Abstract - The goal of this project was to see if stable, robust, and efficient network topologies can emerge from simple local interconnection rules. To accomplish this, experiments were performed where nodes were placed semi-randomly in some space and their interactions analyzed. These nodes formed connections to neighbor nodes within the limits of a spatial 'horizon' given a limited number of connections and limited knowledge. An agent-based simulation was used to produce the sample networks to be evaluated. Resultant networks were analyzed for average and worst case diameter, clustering, and connectivity. Applications of the results include distributed computing systems, distributed sensor networks and other ad-hoc physical systems.

1. Motivation

With the advent of small devices and wireless network connectivity, the question of how to configure networks of devices without prior knowledge of the network topology has become increasingly important. The authors thought it would be interesting to explore, first, whether it was possible to create useful network graphs with only limited knowledge of the local topology and, second, to see how different algorithms operating on this local information performed relative to a set of measurements. We reasoned that if it were possible to 'evolve' networks in this fashion then there might be applications in a number of areas for networks evolved from these algorithms.

2. Self Evolving Network Simulation

The objective of this project was to see if usable network graphs could evolve from a simple set of locally applied rules. To this end the authors designed a scenario which could be expressed as a simulation. The simulation would use several different sets of local rules or algorithms to produce networks to be analyzed. The results could be explored to see what kinds of networks emerged and their quality relative to our goals of robustness, efficiency, and stability. The quality of the network would be measured along several axes relevant to those goals: minimum and average diameter, clustering coefficient, hamming distance, connectedness, and robustness.

3. Simulation Scenario

The simulation places nodes randomly on a grid. The grid, for purposes of the data presented here, is 1000 by 1000 and 125 nodes are placed for a given simulation run. Node coordinates are generated by a pseudo-random number generator with a uniform distribution (Figure 1). Each node has a degree limit which limits the number of edges it can have connecting it to other nodes. Edges are considered to be undirected. Nodes also have a horizon limit expressed as the radius of a circle. A given node can only see other nodes within the radius of this circle and can only ask nodes within that circle for information (Figure 2). A node can, however, get unique identifiers for nodes that are connected to the nodes that are within the horizon. These identifiers may be used only for purposes of equality comparison.

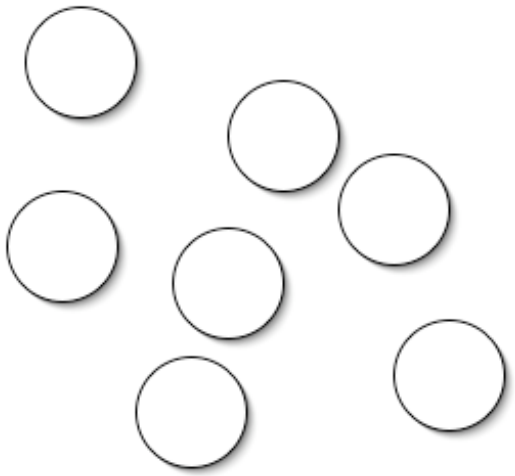


Fig.1: Randomly placed nodes

With these constraints in place nodes are given an opportunity to make a decision about what other nodes within their horizon they will connect to. A connection is defined as placing an edge in the graph connecting one node to another. The number of connections from a given node to other nodes, including connections other nodes have made to the node in question, must not exceed the degree limit for the node. This constraint is enforced during the connection decision process and thus there is no need to make a decision about disconnecting over-connected nodes.

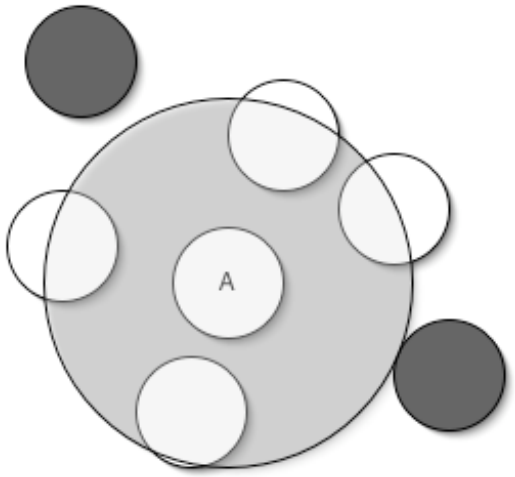


Fig. 2: Node A's horizon, shown in grey, includes only the white nodes

Each node in the simulation is equipped with a 'connector'. This term connector is not meant to evoke the image of a device that passively connects two parts of a circuit together but rather something which actively makes decisions about how connections should be made. The connector has information available to it about the nodes within the horizon limit. It can only choose connections starting from the node to which the connector is attached and ending at a node within the horizon limit. The connector can only make those choices when instructed to do so by the simulation. If a connector is invoked more than once on a node and sees connections which originated from a previous decision by the connector on that node (an 'out' edge) then it can potentially reroute those connections. If, however, another node's connector has connected to the node in question (an 'in' edge) then that connection cannot be disturbed by the node that has received the connection. A node receiving a request for connection from another node's connector can refuse the connection, in which case the connector must find somewhere else to connect. In this sense edges are directional but the directionality is only part of the simulation behavior and is not part of the analysis of the resultant networks. In addition to these rules, no node is allowed to form more than one connection to any given node. Once two nodes are connected either through an in edge or an out edge, they are considered connected and cannot form a second connection.

The reasoning behind this approach to connectors was more one of convenience rather than any belief that it would produce superior results. Using these rules, connection behavior is easily understood, each node is mostly in control of its destiny, and implementation is relatively straightforward.

Connectors are allowed to act on a node multiple times and can make new decisions as the simulation progresses. This may result in stable network structures or unstable network structures depending upon the connector's behavior. To allow for stabilizing behavior, edges that connect nodes are equipped with a locking mechanism to instruct the connector in the simulation not to reroute the edge.

4. Simulation Implementation

The simulation scenario as described above was implemented in the Java language using the RePast simulation framework for creating agent based simulations. RePast provided a reasonable starting point for this development given the time constraint of producing a simulation within approximately two weeks. The RePast framework envisions a simulation as a state machine whose state is constituted by the collective states of all its components.

Java's object orientation and RePast's network support led naturally to a design where objects were used to encapsulate the state and behavior of each active component of the simulation. Each node became an object that managed edge objects connected between the nodes. Node objects also had connector objects attached to them which defined their connection behavior. The simulation framework managed the collection of node objects and provided a mechanism for progressing through the simulation run and for displaying the resultant graph.

A number for tunable parameters were designed into the simulation. Parameters included the size of the simulation space, number of nodes, horizon radius limit, edge degree limit, and connector object to use to name just a few. Other controls were available to disable the node, unlock it's edges, or cause it to move around in a random walk. Few of these were actually changed for simulation runs presented here in order to limit the scope of the investigation.

The simulator was developed to be able to output an ASCII representation of a connection matrix at any point in the simulation. The capability to take snapshots of the network graph as graphically represented by the RePast system and the capability to make movies of network evolution were also included.

Simulations that were run to compare different connection algorithms were initialized such that nodes were placed on the same grid points for each run. In each case the dimensions of the grid to place nodes was set to 1000 by 1000, 125 nodes were randomly placed in this grid, the node degree limit was set to 4 edges, and the horizon radius limit was set to 300 grid units. Later in the investigation Muresan varied the horizon limit and gathered data for connectivity vs. horizon. The size of the grid and the number of nodes were chosen to get a reasonable number of nodes in play to allow for the law of large numbers. The degree limit and horizon limit were chosen to be somewhere between the point where disconnected networks resulted and where fully connected networks resulted. In this way each connection algorithm led to detectable differences in the resultant networks.

5. Connection Algorithms

Several different connection algorithms were devised to try to understand what kinds of networks could evolve from local rules. These algorithms were expressed within the context of a connector object and the connector objects were used for simulation runs with other parameters unchanged. The algorithms were chosen on the basis of intuition gleaned from other work in the area of self assembling networks and from theoretical work on network topologies that result in what we deemed desirable properties.

The connection algorithms are summarized in Table 1:

Random
Greedy
Distant
Free Edge
Maximum Hamming Distance
Minimum Hamming Distance

Table 1: Connection algorithm summary

Most of these (excepting Minimum

Hamming Distance) has a counterpart which was designed to stabilize the network using a pseudo-simulated annealing algorithm to be described later. The above algorithms combined with the pseudo-simulated annealing are called the 'stabilizing' version of the algorithm (e.g. Greedy-Stabilizing).

The Random connector, as the name implies, attempts to connect randomly to another node. It finds nodes within the horizon radius limit and chooses a random number of those in random order up to the degree limit. It then attempts to build those connections until it runs out of neighbors or exceeds the randomly chosen degree limit.

The Greedy connector greedily tries to connect to the closest nodes it can find within the horizon radius limit. Closeness is defined as the straight line distance measured in grid squares between a node and its neighbor. It sorts the nodes in its neighborhood by distance and then attempts to connect to the closest nodes that will accept connections up to the edge degree limit.

The Distant connector is the opposite of the Greedy connector. It tries to connect to the most distant nodes it can find within the horizon radius limit. It sorts the nodes in its neighborhood by distance and then attempts to connect to the most distant nodes that will accept connections up to the edge degree limit.

The Free Edge connector is a variant of the Greedy connector. It tries to connect to the nodes with the most free edges, or greatest number of open connections, within the degree limit. If two nodes have the same number of free edges then it defaults to the Greedy connector algorithm of connecting to the closest node. It sorts the nodes in its neighborhood by free edges followed by distance and then attempts to connect to the freest and then closest nodes that will accept connections up to the edge degree limit.

The Maximum Hamming Distance connector tries to connect to the nodes with the most disjoint set of connections within the horizon radius limit. The Hamming distance is calculated by comparing a node's connected neighbors with another node's connected neighbors. The hamming distance starts at zero and is incremented for each member of one neighbor set that is not contained within the other neighbor set. The connector sorts the nodes in its neighborhood by Hamming distance and then attempts to connect to the most disjoint (maximally Hamming distant) nodes that will accept connections up to the edge degree limit.

The Minimum Hamming Distance connector was implemented as a control and test of the Hamming distance algorithm. It is the opposite of the Maximum Hamming Distance connector and tries to connect to the nodes with the most similar set of connections within the horizon radius limit. The connector sorts the nodes in its neighborhood by Hamming distance and then attempts to connect to the most similar (minimally Hamming Distant) nodes that will accept connections up to the edge degree limit.

6. Edge Stabilization

Repeatedly applying the above algorithms to all nodes in the simulation yields networks which are constantly in flux. In an effort to stabilize this behavior a stabilization algorithm was devised by McIntyre which, in retrospect, resembles simulated annealing but is not strictly simulated annealing. The stabilization results from probabilistically locking down edges while repeatedly applying the connector to the network.

In this algorithm if an edge is locked down, it is no longer eligible to be disconnected when the connector makes its decisions about new connections. If an edge remains in place for multiple applications of the connector, the probability of it being locked down is increased in proportion to its apparent stability. If an edge is appearing where there wasn't an edge on the last application of the connector then the

probability of being locked down is not increased but the probability is still non-zero. On each application of the connector to all the nodes in the network the number of locked down edges monotonically increases. Eventually all edges are locked down and at that point the network is said to be stabilized.

7. Simulation Output

Once a network simulation run is complete the final configuration of the network is output in an ASCII file format. This file contains header information indicating the settings of various simulation parameters, the node names, and the connection matrix for the network graph. The connection matrix is populated by numbers representing edges in the graph. A zero entry signifies no edge and a non-zero number represents the distance between the connected nodes in units of fractional grid squares. The distance numbers can potentially be used for latency analyses of the resultant networks. These output files were read into a MATLAB program that performed the network analysis.

8. Results

Ten networks were randomly generated for each algorithm and all edges longer than h = 300 were removed. The allowed maximum degree of a node in the resulting network was set to 4.

The resulting networks were characterized according to the following properties:

- Clustering coefficient
  The clustering coefficient of a node $v \in V$ is the percentage of neighbors of v connected to each other. The clustering coefficient of a graph is: $\gamma = \frac{\sum_{v \in V} \gamma(v)}{|V|}$.
- Diameter - the length of the shortest path between the furthest two nodes of the graph.
- SP average - The average of the shortest paths between any two nodes of the graph
- Robustness/Connectivity

For each network, the values for these properties are presented in Table 2:

Input Network	Clustering Coefficient	Diameter	SP Average
1	0.62627	1318.0596	500.2426
2	0.61608	1176.4242	486.3047
3	0.63956	1280.698	550.4126
4	0.63011	1325.6647	524.6535
5	0.64268	1255.7348	510.1326
6	0.65926	1287.4018	506.2337
7	0.62036	1278.5394	520.249
8	0.6271	1326.7616	556.0601
9	0.60669	1380.861	513.8656
10	0.62516	1238.5539	513.2025

Table 2: Characteristics of the input networks

The robustness of the resulting structures was assessed by testing the connectivity of sub-graphs of the input networks. Connectivity was assesed by removing n percent of the original nodes, $n \in \{0, 5, 10, 15, 20, 25, 30, 35, 40\}$ at random (and all edges incident to these nodes). For each graph and for each value of n the procedure was repeated 100 times and the resulting connectedness percentages are summarized in Table 5. Other data are presented in the tables that follow.

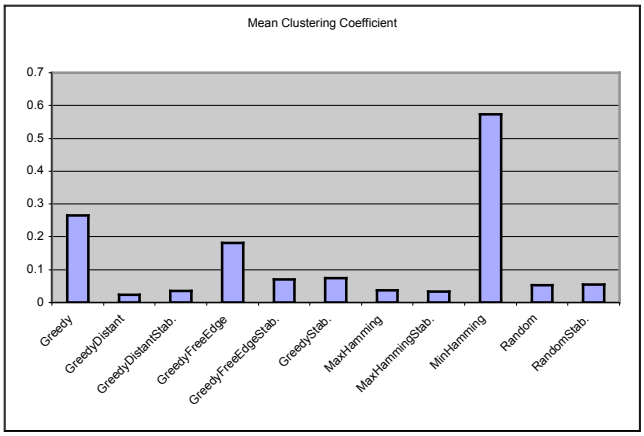


Fig. 3: Clustering coefficient averages of the resulting graphs (for the 10 input networks)

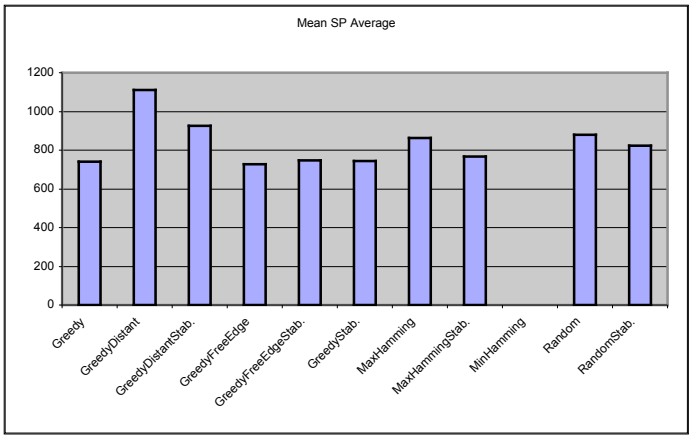


Fig. 5: Average SP of the resulting graphs

We use the diameter and SPAverage as a worst and an average-case indicator, respectively. The smallest diameters were produced by the MaxHammingStabilizing, GreedyStabilizing, and GreedyFreeEdgeStabilizing methods, while the smallest SP averages were produced by the Greedy, GreedyStabilizing, and the GreedyFreeEdge algorithm.

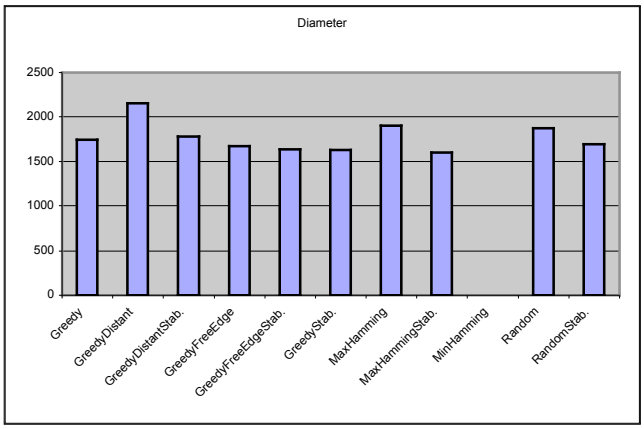


Fig. 4: Average diameters of the resulting graphs

Without claiming statistical significance, we note that almost all the methods presented performed better than the simple random algorithm (with the exception of the Greedy and MaxHamming methods from the connectedness point of view, and GreedyDistant from the diameter point-of-view). All the methods that use the stabilizing procedure (including RandomStabilizing) performed similarly well as far as connectedness was concerned.

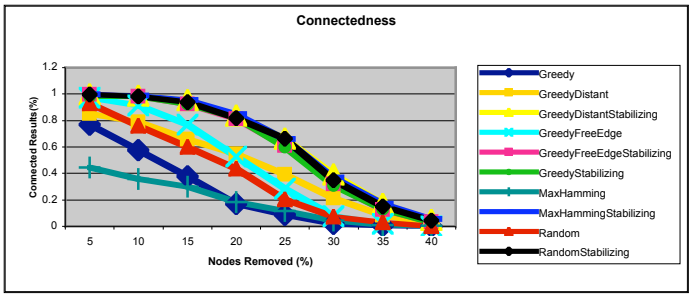


Fig. 6: Connectedness results for the tested strategies

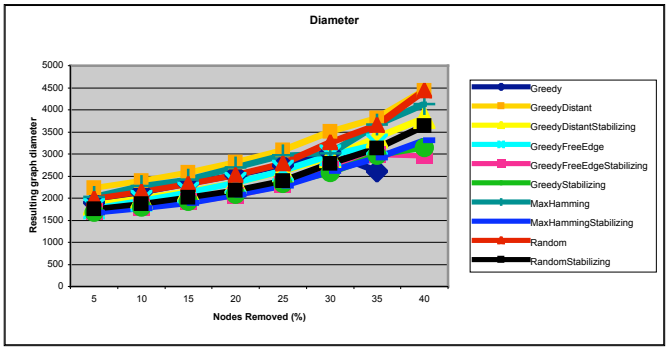


Fig. 7: Diameter averages produced by the tested strategies

The poor performance of the MaxHamming algorithm is due to the fact that for five test networks the resulting graphs were disconnected. For the remaining five test inputs the results were considerably better. The GreedyDistant and the GreedyFreeEdge strategies proved to be the best one-step methods.

8.1. Connectedness for Variable Horizons

Figure 8, shows the dependence of connectedness on the horizon (h). Each color represents a different horizon value, between 100 and 400.

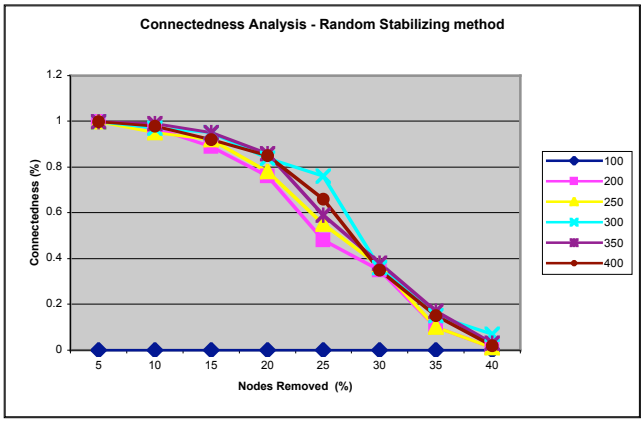


Fig. 8: Connectedness results for the Random Stabilizing method

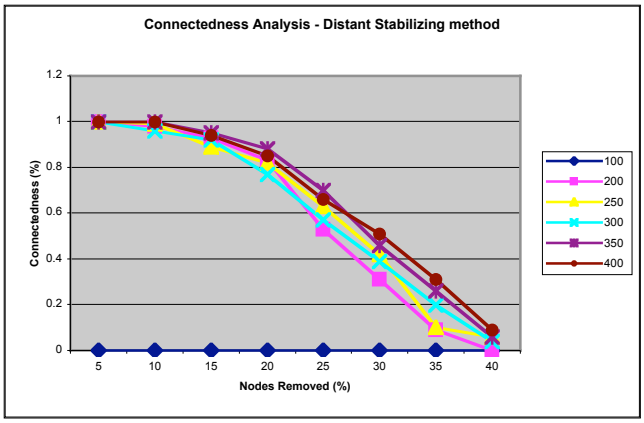


Fig. 9: Connectedness results for the Distant Stabilizing method

Apparently, the bigger the horizon, the better the DistantStabilizing method performs, compared to the Random Stabilizing one. One line of the following tables shows the connectedness of the graphs for each horizon value (after n % of the nodes were removed).

9. Conclusions

The results shown here indicate to the authors that, for the parameters we chose to measure, networks with desirable characteristics can be generated using only information local to nodes placed randomly in a graph. One implication of this is that, with attention to the algorithms employed, useful self-evolving data networks can be created for collections of devices. We believe that further investigation of unexplored areas of the current simulation framework is warranted and that it is likely to yield a better understanding of what types of connection algorithms work in what circumstances and, perhaps, why some algorithms significantly outperform others.

10. References

[1] Dekker, A. H. & Colbert D. B. (2004), Network Robustness and Graph Topology, Defence Science and Technology Organisation (DSTO) Canberra, Australia

[2] Fortuna, L. (2004), Reinforcing the Resilience of Complex Networks, Institute of Physics of São Carlos. University of São Paulo

[3] K. Sohrabi, W. Merrill, J. Elson, L. Girod, F. Newberg, and W. Kaiserswitching (2002), Scalable Self-Assembly for Ad Hoc Wireless Sensor Networks, Sensoria Corporation

[4] S. Meguerdichian, F. Koushanfar, M. Potkonjak, M. B. Srivastava (2001), Coverage Problems in Wireless Ad-hoc Sensor Networks, University of California, Los Angeles

[5] M. E. J. Newman (2003), The structure and function of complex networks, Department of Physics, University of Michigan and Santa Fe Institute

[6] S. H. Strogatz (2001), Exploring complex networks, Cornell University, Nature Vol 410, MARCH 2001

Horizon *h*/ nodes removed *n*	100	200	250	300	350	400
5	0	1	1	1	1	1
10	0	0.97	0.95	0.97	0.99	0.98
15	0	0.89	0.92	0.95	0.95	0.92
20	0	0.76	0.78	0.84	0.86	0.85
25	0	0.48	0.55	0.76	0.59	0.66
30	0	0.35	0.37	0.36	0.38	0.35
35	0	0.1	0.1	0.15	0.17	0.15
40	0	0.01	0.01	0.07	0.03	0.02

Table 3: Connectedness vs. horizon limit for the RandomStabilizing method

Horizon *h*/ nodes removed *n*	100	200	250	300	350	400
5	0	1	1	1	1	1
10	0	0.98	0.99	0.96	1	1
15	0	0.93	0.89	0.92	0.95	0.94
20	0	0.83	0.82	0.77	0.88	0.85
25	0	0.53	0.63	0.57	0.7	0.66
30	0	0.31	0.42	0.39	0.46	0.51
35	0	0.09	0.1	0.2	0.26	0.31
40	0	0	0.06	0.04	0.06	0.09

Table 4: Connectedness vs. horizon limit for the DistantStabilizing method

	Median Clustering Coefficient	Mean Clustering Coefficient	Median Diameter	Mean Diameter	Median SP Average	Mean SP Average
Greedy	0.277625	0.2657755	1662.0908	1746.2089	743.843	739.609
GreedyDistant	0.023363	0.02278178	2152.8874	2153.61682	1099.1772	1110.7228
GreedyDistantStab.	0.0320565	0.0346491	1771.10235	1780.35647	935.3276	924.91162
GreedyFreeEdge	0.1713	0.180925	1655.60955	1672.60408	732.05375	725.83881
GreedyFreeEdgeStab.	0.0679755	0.0697503	1612.96955	1632.67909	745.7608	745.88697
GreedyStab.	0.072765	0.0735597	1603.93335	1628.67186	733.81565	744.56787
MaxHamming	0.034555	0.0374186	1876.116	1904.03314	847.7668	863.61536
MaxHammingStab.	0.027383	0.0322405	1610.11585	1598.34471	765.1664	765.695
MinHamming	0.57639	0.57405				
Random	0.0540875	0.0526713	1817.5528	1876.76224	873.2156	878.73432
RandomStab.	0.054381	0.0537025	1652.87665	1693.65841	818.08415	822.02785

Table 5: Properties of the resulting networks (averages over the 10 test inputs)

Nodes Removed (%) *n*	Greedy	Greedy Distant	Greedy Distant Stabilizing	Greedy FreeEdge	GreedyFree Edge Stabilizing	Greedy Stabilizing	Max Hamming	Max Hamming Stabilizing	Random	Random Stabilizing
5	0.77	0.85	0.999	0.974	0.996	0.984	0.447	0.998	0.927	0.999
10	0.578	0.788	0.988	0.915	0.981	0.984	0.359	0.989	0.759	0.98
15	0.378	0.658	0.951	0.771	0.923	0.926	0.298	0.953	0.598	0.939
20	0.167	0.546	0.837	0.527	0.814	0.825	0.186	0.847	0.434	0.818
25	0.085	0.391	0.663	0.285	0.611	0.599	0.116	0.657	0.202	0.661
30	0.02	0.211	0.394	0.077	0.317	0.31	0.045	0.37	0.072	0.349
35	0.004	0.082	0.168	0.017	0.126	0.132	0.01	0.167	0.028	0.15
40	0	0.017	0.05	0	0.035	0.029	0.004	0.054	0.001	0.043

Table 6: Connectedness percentage of resulting graphs after removing *n* percent of the nodes

Nodes Removed (%) *n*	Greedy	Greedy Distant	Greedy Distant Stabilizing	Greedy FreeEdge	GreedyFree Edge Stabilizing	Greedy Stabilizing	Max Hamming	Max Hamming Stabilizing	Random	RandomSta bilizing
5	1891.66556	2235.02058	1860.68998	1768.34165	1689.04064	1687.41167	2025.31169	1663.57466	1980.03354	1758.2301
10	2130.42531	2386.27381	2000.74805	1939.51657	1788.49428	1794.912	2274.7718	1768.22496	2145.59319	1871.69814
15	2335.6531	2583.60157	2159.84225	2131.54996	1912.38606	1924.59346	2413.12606	1887.64826	2314.55284	2009.8167
20	2521.15305	2815.12315	2352.00449	2353.40377	2057.60677	2090.55232	2694.09127	2065.43099	2517.26643	2178.42105
25	2739.27548	3081.83877	2604.67902	2623.74751	2303.53584	2327.02501	2964.27624	2276.54288	2777.69611	2394.2306
30	3033.81024	3506.81741	2933.31997	2971.46552	2637.3293	2581.35328	3017.00128	2612.58211	3260.62689	2777.23644
35	2613.03205	3814.57758	3370.5656	3222.54531	2990.16202	2964.02767	3659.5235	2917.0162	3652.89154	3130.32053
40	_	4440.89136	3813.2361	_	2952.19629	3143.21974	4127.11719	3301.19898	4429.02793	3634.93028

Table 7: Average of diameters of resulting graphs after removing *n* percent of the nodes

DIC Image Reconstruction on Large Cell Scans

Bettina Heise, Alois Sonnleitner & Erich Peter Klement

1. Introduction

Fluorescence microscopy has become the main method in imaging of cells, but differential interference contrast (DIC) microscopy is indispensable for special purposes avoiding markers. The physical background for DIC microscopy is described e.g. in (Murphy, 2001; Pluta, 1994; Holmes and Levy, 1987) and the references cited therein.
Typically DIC images appear with a bas-relief profile caused by the gradient of the optical path length θ and phase shift between the two beams. For the human eye these DIC images are easy to interpret, (*Fig.1a*), but for automatic image analysis of the DIC scans with hundreds of cells of different shapes and partially weakly identifiable contours they are complicate to analyze automatically.
In the direction of the DIC shear vector $\Delta\tau$ the intensity distribution of the cell is characterized by a transition of bright to dark, resulting in a well- defined contrast. But in the perpendicular direction to the DIC shear we have no contrast against the background and hence a lack of information about the complete cell boundary, (*Fig.1b*).

2. Mathematical algorithms for DIC image reconstruction

2.1. Non-iterative approaches

The two characteristic parameters for the DIC images are the DIC shear $\Delta\tau = (\Delta x, \Delta y)$ and the phase difference $\Delta\varphi_0$ which describes the bias retardation between the two laterally sheared DIC beams.
The intensity I_0 of the DIC image can be described generally (Cogswell and Sheppard, 1992) by

$$I_0 = 4a^2 \sin^2 \frac{\Delta\varphi_0}{2}. \tag{1}$$

The intensity change $\Delta I = I - I_0$ caused by an additional phase change $\Delta\psi$ between the two beams crossing an object regarding only the linear part is given by

$$I - I_0 = \frac{\partial I}{\partial \varphi}\Delta\psi = 4a^2 \Delta\psi \sin\Delta\varphi_0. \tag{2}$$

For nearly transparent objects we can assume that the object phase change $\Delta\psi$ is proportional to the directional derivative $\frac{\partial}{\partial r}$ of the optical path length θ in DIC shear direction $r = \Delta\tau$,

$$I - I_0 = \frac{\partial I}{\partial \varphi}\Delta\psi = 4a^2 \frac{\partial\theta}{\partial r}\sin\Delta\varphi_0. \tag{3}$$

2.1.1. Line Integration

Working in the highest dynamic range around $\Delta\varphi_0 = \pi / 2$ we can integrate *(3)* in DIC shear direction $r = \Delta\tau$ to get a dependence on the optical path length θ. For our setup with a DIC shear angle $\tau = 45°$ we have to integrate, i.e. to sum up the intensity changes along each diagonal pixel line i up to the considered pixel $r_n = (x_n, y_n)$. This integration generates a new image with an intensity $S_L(x, y)$ proportional to the optical path length θ of the object at the position (x, y) related to the background,

$$S_L(x_n, y_n) = \int_{Bound_i}^{(x_n, y_n)} (I(r) - I_{i,0})\, dr = \tilde{a}\, \theta(x_n, y_n) + c_i\,, \tag{4}$$

with the constant $\tilde{a} = 4a^2 \sin \Delta\varphi_0$. $I_{i,0}$ denotes the mean diagonal line intensity in the original DIC image, c_i are a measure for the intensities at the image boundary for each line *i*.
However, this 'integrated' image is disturbed by a randomly distributed striped structure caused by accumulation of noise along the integration path and by other effects, (*Fig. 2a*).

2.1.2. Corrected Line Integration

The correction of these artifacts can be achieved by including an additional bi-directional, exponential decay term δ into the line integration method as proposed in (Kam, 1998). We implement this decay term in the following way and get a new image with an intensity $S_{LC}(x, y)$,

$$S_{LC}(x_n, y_n) = \int_{r_1}^{(x_n, y_n)} (I(r) - I_{i,0})\, e^{-\delta |r_n - r|} dr - \int_{r_2}^{(x_n, y_n)} (I(r) - I_{i,0})\, e^{-\delta |r - r_n|} dr\,. \tag{5}$$

Compared to the straightforward line integration the artifacts can be reduced by corrected line integration method, but the intensity inside the cells is very low, (*Fig. 2b*).

2.1.3. Hilbert Transform

The Hilbert transform represents a second method to convert the bas- relief profile of the objects into symmetric appearance, (Arnison et al., 2000). Performing this transform in Fourier space gives a simple multiplication by the signum-function.

$$F_H(m) = \int_{-\infty}^{\infty} f_H(x) \exp(-i\, 2\pi\, m\, x)\, dx = -\,i\, \mathrm{sgn}(m)\, F(m)\,. \tag{6}$$

In the two-dimensional frequency space the corresponding 2D-signum function has to be perpendicular to the DIC shear direction. Since our DIC shear direction is along the diagonal of the square image, we define the edge of the signum function in the perpendicular diagonal direction. This approach differs from (Arnison et al., 2000; Kyan et al., 2001), where the Hilbert transform is performed in a DIC shear direction parallel to the image edges.
The real part of the inverse Fourier transform of (6) gives the resulting converted image, (*Fig. 2c*).

2.1.4. Deconvolution by Wiener Filtering

A further way to convert the DIC image into an image similar to one as resulting from fluorescence microscopy is the deconvolution with an appropriate 'DIC-shear function'.
Assuming as described in (van Munster et al., 1997) that the phase difference $\Delta\varphi(x, y)$ between the two laterally sheared DIC beams, causing the typical DIC bas–relief profile can be expressed as convolution of the DIC-shear function $g(x, y)$ with the phase-function $\varphi(x, y)$ which represents a measure for the optical path length of the objects, we can write

$$\Delta\varphi(x, y) = g(x, y) * \varphi(x, y) = (\delta(x - \frac{\Delta x}{2}, y - \frac{\Delta y}{2}) - \delta(x + \frac{\Delta x}{2}, y + \frac{\Delta y}{2})) * \varphi(x, y)\,, \tag{7}$$

approximating $g(x, y)$ by two $(\Delta x, \Delta y)$ - shifted δ- functions.
By a Wiener Filter (Gonzales et al., 1992) we can reconstruct $\Phi(u, v)$ getting the best estimation $\Phi'(u, v$.
The signal noise ratio approximated by a Gaussian function as proposed in (van Munster et al., 1997) seems a good assumption also for our images. The parameters of the Gaussian (amplitude κ and spread σ) have to be adapted for the specific images by testing for the highest achievable image quality.
$\Delta x, \Delta y$ must be determined for the microscope setup, then $\varphi'(x, y)$ can be calculated by the inverse Fourier transform of our estimated function $\Phi'(u, v)$. Its real part gives a measure for the optical path length, (*Fig. 2d*).

2.2. Iterative approaches

2.2.1. Iterative Line Integration

After the application of transformation methods described in (2.1.1.- 2.1.4.) the objects can be reconstructed with an brightfield-like appearance, but the cells often are incomplete and not 'filled', the intensity of the transformed objects and consequently the contrast to the background seems to be insufficient for our standard fluorescence image analysis. To improve the results we test iterative approaches for DIC image reconstruction.
We implement similar to the approach of Z. Kam the iterative line integration method by performing a damped integration as described in (2.1.2.) in the forward direction and a differentiation without any correction factor in the backward direction which gives an approximation of the original DIC image as one iteration step. The iteration is repeated with the residual given by the difference between the original and the approximated DIC image adding the result to the previous approximated DIC image. The integrated images of each iteration step are accumulated to the final reconstructed image.
The appearance of the transformed objects is improved after several iteration steps, but on the other hand the cells get shadows and fringes as a 'decay effect' with increasing number of iteration. Hence a compromise for the number of iteration has to be found by testing with different parameters. Additionally, by the repeated integration we have a low-pass filtering effect.

2.2.2. Iterative Hilbert Transform

The Hilbert transform has a transfer function of unit magnitude and hence the transformed images don't show the blurring effects as compared to the line integration. Furthermore its realization in the Fourier space is a computationally elegant solution.
In our approach for the iterative Hilbert transform (IHT) we use the Hilbert transform in forward direction and the differentiation in the backward direction as one iteration step. The difference between original and approximated image gives the residual on which the next iteration step is performed. The sum of all the resulting Hilbert transformed images gives the final image.
By the repeated differentiation in the iteration loop we have the effect of a high-pass filter and so we particularly stress the noisy parts of the image with increasing number of iterations. To reduce these effects we introduce a modified version of the iterative Hilbert transform (MITH) with the operator $\mathcal{H}_{\alpha}^{n}$,

$$\mathcal{H}_{\alpha}^{n}\{Img\} = \mathcal{F}^{-1}\{-i\,\mathrm{sgn}(u-v)\frac{\alpha}{\alpha + n(|u-v|^2 + |u+v|^2)}\mathcal{F}\{Img\}\}, \tag{8}$$

which suppresses the higher frequencies according to the chosen parameter α with increasing number of iteration.
Our MIHT-method is implemented as follows

$$
\begin{aligned}
&\mathcal{H}_{\alpha}^{0}\{Img_0\} = \tilde{T}\Delta Img_0 = \tilde{H}Img_0 = HImg_0 \\
&\mathcal{D}\{\tilde{T}\Delta Img_0\} = Img_1 \\
&\Delta Img_1 = Img_0 - Img_1 \quad \text{for } n = 0, \\
&\mathcal{H}_{\alpha}^{n}\{\Delta Img_n\} = \tilde{T}\Delta Img_n \quad . \\
&\tilde{H}Img_n = \tilde{H}Img_{n-1} + \tilde{T}\Delta Img_n \\
&\mathcal{D}\{\tilde{T}\Delta Img_n\} = Img_{n+1} \\
&\Delta Img_{n+1} = \Delta Img_n - Img_{n+1} \quad \text{for } n = 1,2,\ldots
\end{aligned}
\tag{9}
$$

$\mathcal{D}$ denotes the differentiation operator:

$$\mathcal{D}\{Img\} = Img * \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

Finally we get the resulting image

$$\tilde{H}Img_n = \sum_{k=0}^{n} \tilde{T}\Delta Img_k \ . \tag{10}$$

3. Results

Application of the different non-iterative reconstruction methods (corrected line integration, Hilbert transform and deconvolution) yielded only suboptimal results for our purposes. Iterative methods for DIC image reconstruction (iterative line integration and MIHT- method) could achieve an improvement, (*Fig. 3a*) and (*Fig. 3b*).

We applied to these reconstructed images our standard algorithms used for the fluorescence cell scan analysis, i.e. a global threshold method (Otsu algorithm (Shapiro et al., 2001) for bimodal histogram distributions and Triangle algorithm for unimodal histogram distributions (Zack, 1977)) and a local threshold method (a modification of the Niblack's Method (Trier et al., 1995)) for determining the positions and the boundaries of the cells and cell clusters.

The binary images in *Fig.4* are displayed without any correction of noise and processing artifacts to be able to compare the results. In addition we applied afterwards a special directive erosion method to reduce the fringes after binarization for both iterative methods.

Concerning the finally ascertained cell boundaries the method similarly proposed by (Kam, 1998) and our iterative method were comparable, (*Fig. 5a, 5b*), although the reconstructed objects in the first case were more blurred and hence fine details were better to recognize by the MIHT-method.

A survey of the advantages and disadvantages of the investigated methods is displayed in *Tab. 1*.

Method	Advantages	Disadvantages
Hilbert Transform (HT)	fast method, algorithm computationally easy to realize in the Fourier space	discontinuities and gaps in objects after thresholding, low contrast for thin objects
Deconvolution	smooth background	DIC shear and microscope parameters must be determined, noise distribution often not exactly known
Iterative Line Integration	smooth background	blurring effect, fringes and shadows
Modified Iterative Hilbert Transform (MIHT)	contrast improved, sharper contours than in Iterative Line Integration	slightly streaked noise pattern in the background

Tab.1: Comparison of investigated methods

4. References

Arnison MR, Cogswell CJ, Smith NI, Fekete PW, Larkin KG. 2000. Using the Hilbert transform for 3D visualization of differential interference contrast microscope images. J Microsc 199: 79-84.

Cogswell CJ, Sheppard CJR. 1992. Confocal differential interference contrast (DIC) microscopy: including a theoretical analysis of conventional and confocal DIC imaging. J Microsc 165: 81-101.

Feineigle PA. 1996. Motion Analysis and Visualization of Biological Structures Imaged via Nomarski Differential Interference Contrast Light Microscopy. PhD thesis. CMU.

Feineigle PA, Witkin AP, Stonick VL. 1996. Processing of 3-D microscopy images for data visualization. Proc.IEEE Int.Conf. Acoustics, Speech and Signal Processing 4: 2160-2163.

Gonzales RC, Woods RE. 1992, Digital Image Processing, Addison-Wesley Publishing Company: 279-282.

Gray AJ. 1999. Detecting cells in DIC microscope images using a high level Bayesian model and template matching. IEEE Colloquium on Applied Statistical Pattern Recognition.

Havlicek JP, Havlicek JW, Bovik AC. 1997. The analytic image. Proc. IEEE, Int. Conf. Image Processing: 446-449

Hesse J, Sonnleitner M, Sonnleitner A, Freudenthaler G, Jacak J, Höglinger O, Schindler H, Schütz GJ. 2004. Single molecule reader for high throughput bioanalysis. Anal.Chem **76**: 5960-5964.

Holmes TJ, Levy WJ. 1987. Signal-processing characteristics of differential-interference-contrast microscopy. Applied Optics 26: 3929-3939.

Kam Z. 1998. Microscopic differential interference contrast image processing by line integration (LID) and deconvolution. Bioimaging 6: 166-176.

Kyan MJ, Guan L, Arnison MR, Cogswell CJ. 2001. Feature extraction of chromosomes from 3-D confocal microscope images. IEEE Trans. Biomedical Engineering 48: 1306-1318.

van Munster EB, van Vliet LJ, Aten JA. 1997. Reconstruction of optical path length distributions from images obtained by a wide-field differential interference contrast microscope. J Microsc 188: 149-157.

Murphy D.2001. Differential interference contrast (DIC) microscopy and modulation contrast microscopy. Fundamentals of Light Microscopy and Digital Imaging. Wiley-Liss, New York, p. 153-168.

Pluta M.1994. Nomarski's DIC microscopy: a review. Phase Contrast and Differential Interference Contrast Imaging Techniques and Applications, SPIE Proceedings 1846, International Society for Optical Engineering: 10-25.

Preza C, Snyder D, Conchello JA. 1997. Image Reconstruction for Three-Dimensional Transmitted-Light DIC Microscopy. SPIE's BiOS97; 3D Microscopy: Image Acquisition and Processing.

Schindler H. 1998. Vorrichtung zur Visualisierung von Molekülen. WO 00/25113.

Shapiro L, Stockman G. 2001. Computer Vision. University of Washington, Prentice Hall.

Trier OD, Jain AK. 1995. Goal-Directed Evaluation of Binarization Methods.IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(12): 1191-1201.

Young D, Gray AJ. 1997. Semi-automatic boundary detection for identification of cells in DIC microscope images. IEE 6th Int. Conf. on Image Processing and its Applications, Dublin, Ireland, 1:346-350. IEEE Conference Publication No. 443.

Zack GW, Rogers WE, Latt SA. 1977. Automatic Measurement of Sister Chromatid Exchange Frequency. J. of Histochemistry and Cytochemistry, 25(7): 741-753.

5. Figure legends

Fig.1a: DIC image of Jurkat cells (500x500 pixels subimage, totally scan size 2784x3796 pixels, scale bar 3 μm, s. Materials,
Fig.1b: Binary image after bi-level thresholding the DIC image
Fig.2a: DIC image after applying uncorrected line integration
Fig.2b: DIC image after applying corrected line integration, exp (-δ) = 0.8
Fig.2c: DIC image after applying 2D Hilbert- transform
Fig.2d: DIC image after applying deconvolution using a Wiener Filter approach with κ = 100, σ = 3
Fig.3a: DIC image after applying iterative line integration with 4 iteration steps and decay factor exp (-δ) = 0.8
Fig.3b: DIC image after applying modified iterative Hilbert transform with 4 iteration steps and α = 40
Fig.4a: Binary image after local thresholding of the corrected line integration- image (s. *Fig. 2b*)
Fig.4b: Binary image after global thresholding of the corrected line integration- image (s. *Fig. 2b*)
Fig.4c: Binary image after local thresholding of the Hilbert transform- image (s. *Fig. 2c*)
Fig.4d: Binary image after global thresholding of the Hilbert transform- image (s. *Fig. 2c*)
Fig.4e: Binary image after local thresholding of the iterative line integration- image (s. *Fig. 3a*)
Fig.4f: Binary image after global thresholding of the iterative line integration- image (s. *Fig. 3a*)
Fig.4g: Binary image after local thresholding of the modified iterative Hilbert transform- image (s. *Fig. 3b*)
Fig.4h: Binary image after global thresholding of the modified iterative Hilbert transform- image (s. *Fig. 3b*)
Fig.5a: Cell boundaries after correction (iterative line integration as reconstruction method, local threshold)
Fig.5b: Cell boundaries after correction (iterative Hilbert transform as reconstruction method, local threshold)

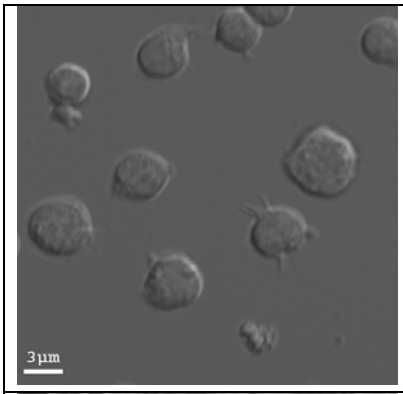


Fig. 1a)

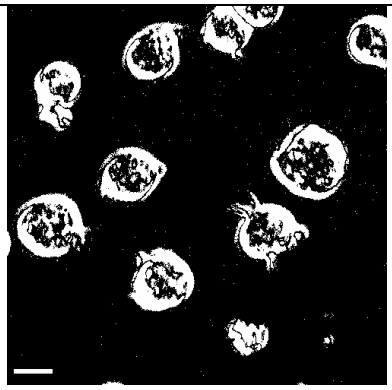


Fig. 1b)

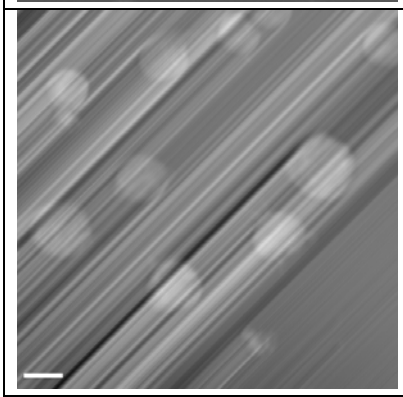
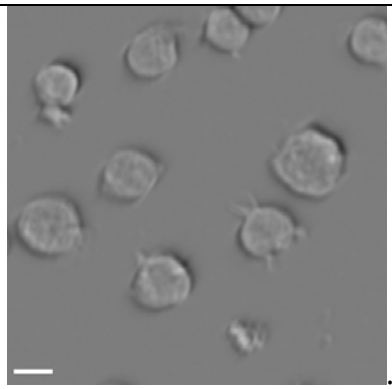


Fig. 2a)


. Fig. 2b)

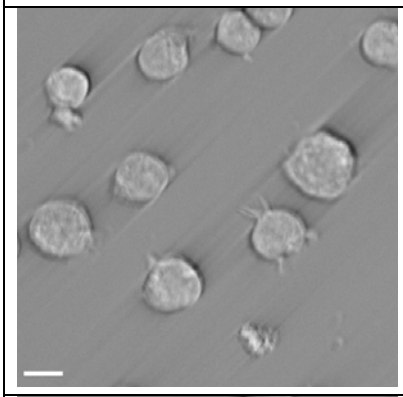


Fig. 2c)

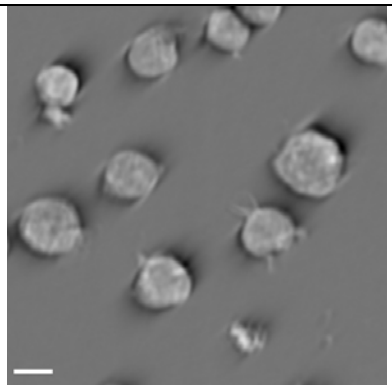


Fig. 2d)

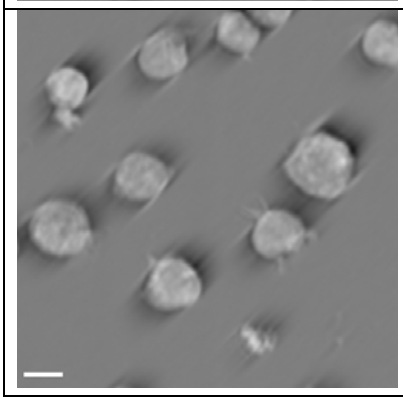


Fig. 3a)

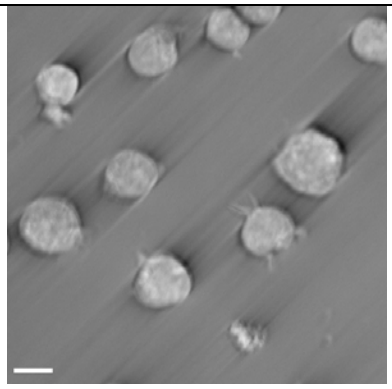


Fig. 3b)

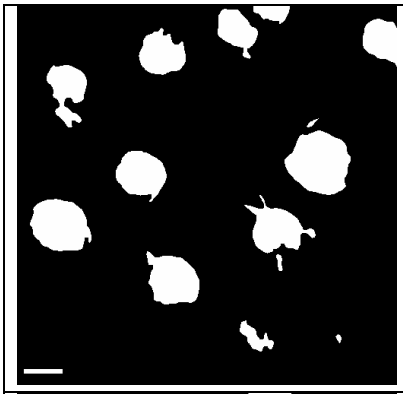


Fig.4a)

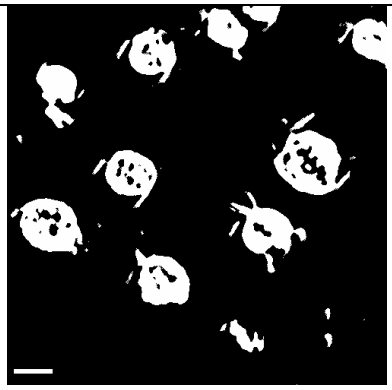


Fig.4b)

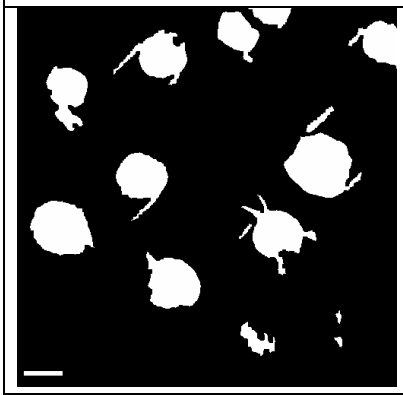


Fig.4c)

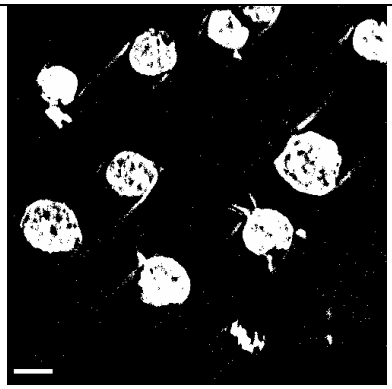


Fig.4d)

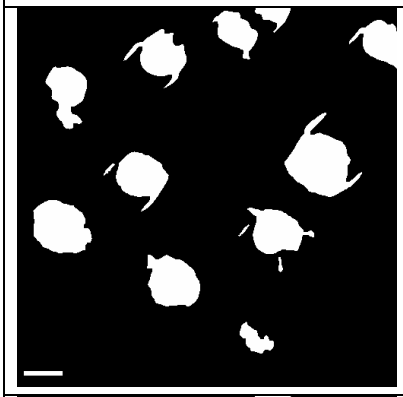


Fig.4e)

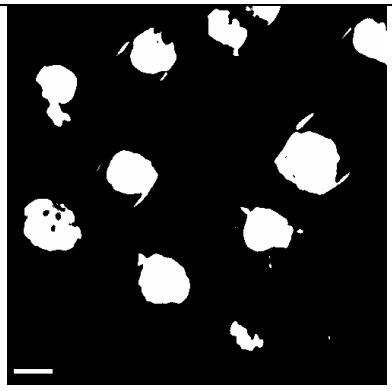


Fig.4f)

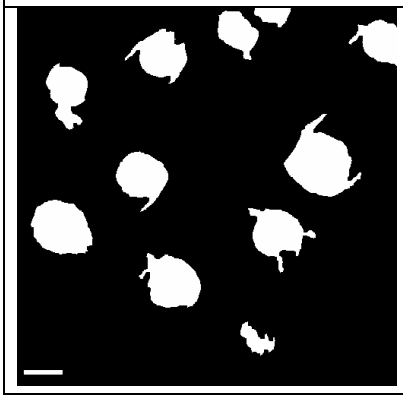


Fig.4g)

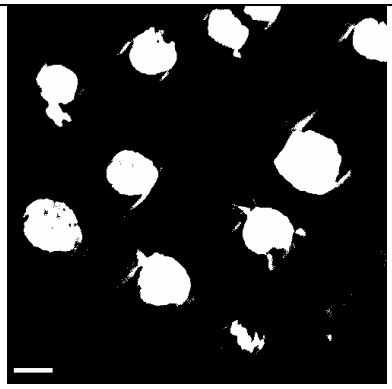


Fig.4h)

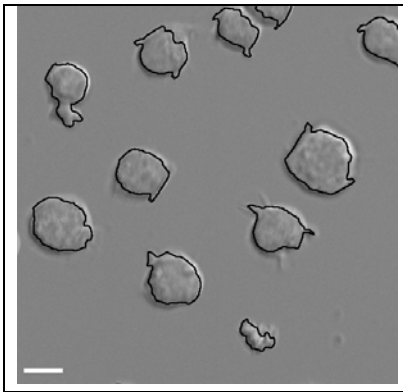


Fig. 5a)

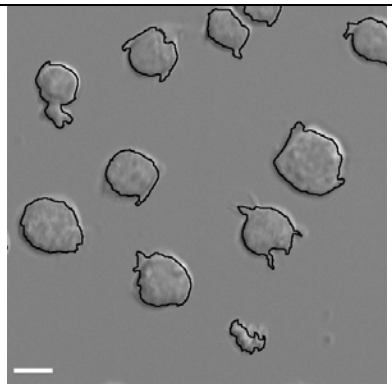


Fig. 5b)

Technical Report
FLLL–TR–0403

FUZZY.LOGIC.LAB.LINZ

Data-Driven Incremental Learning of Fuzzy Systems part 3: Application Cases for FLEXFIS

Edwin Lughofer
Fuzzy Logic Laboratorium Linz-Hagenberg
e-mail edwin.lughofer@jku.at
Erich Peter Klement
Fuzzy Logic Laboratorium Linz-Hagenberg
e-mail ep.klement@jku.at

Abstract — In this paper application cases are demonstrated, where the incremental learning variant for Takagi-Sugeno Fuzzy Systems (*FLEXFIS*) may serve and serves as an important component within their frameworks. These applications include online system identification, online fault detection, prediction, refinement of knowledge-based fuzzy models and open-loop control. Hereby the new method *FLEXFIS* is evaluated on the basis of high-dimensional stationary and dynamic data sets, which are recorded in real-life, (hence possess a certain noise level) and compared with conventional batch learning variants with respect to approximation quality, computation time, process security and interpretability and transparency of the obtained models. A conclusion is drawn where all the applied methods are rated among each other due to different criteria, finally leading to an overall rating.

Key words — *incremental learning, batch learning, online system identification, online fault detection, prediction, refinement of knowledge-based fuzzy models, open-loop control, approximation quality, process security, interpretability*

Johannes Kepler Universität Linz

Institut für Algebra, Stochastik und
wissensbasierte mathematische Systeme
Johannes Kepler Universität Linz
A-4040 Linz
Austria

Fuzzy Logic Laboratorium Linz-Hagenberg

FLLL
Softwarepark Hagenberg
Hauptstrasse 99
A-4232 Hagenberg
Austria

1 Introduction

Nowadays data-driven models become more and more an essential part in industrial systems for application tasks such as system identification and analysis, prediction, control, fault detection or simply simulation. Data driven models are mathematical models which are completely identified from data, which can be available in form of offline data sets, most commonly stored in data matrices, or in form of online measurements. Data-driven models possess the nice property that they can be built up generically in the sense that no underlying physical, chemical etc. laws about the system variables have to be known.

Whenever measurements are recorded online with a certain frequency, usually the models should be kept up-to-date from time to time, especially when tracking highly time-variant system behaviors for online identification tasks, which requires an adaptation of some model parameters in form of incremental learning steps, as a complete rebuilding from time to time with all recorded measurements would yield a too high computational effort for a complete online training. Other application cases for an incremental learning approach stem from the following requirements:

- Refinement of vague knowledge-based models in order to get a more exact representation of the underlying system behaviors
- Improving process security by preventing extrapolation to new operating conditions and by avoiding interpolation for large data holes: this is essential in the case when the first k data points are not sufficiently distributed over the whole input space and therefore the models generated from the first k data points do not contain all possible operating conditions.
- The number of data points from which data-driven models should be trained is too high in order to be able to load them into the memory at once. In this case, the only way to build data-driven models, is to perform a bufferwise loading of the data, which causes the demand of incremental learning algorithms for training the models.
- Auto-adaptation to a similar test object: models were trained for a specific test object within a system and should be adjusted a bit to a similar one in the same system, i.e. to an object possessing similar relationships between some system variables. This guarantees an early applicability of data-driven models to the new test object within an online process.

A new incremental learning variant for Takagi-Sugeno Fuzzy Systems (*FLEXFIS*) were developed in order to satisfy the above mentioned requirements, for detailed algorithms, concepts and methods see [LK03, LK04b, Lug04, LK04a]. In the subsequent sections an extensive evaluation of this new method for is demonstrated. This evaluation is based on some application examples, where data-driven modelling and adaptation yield a substantial contribution including:

- Online (System) Identification
- Fault Detection
- Prediction
- Generation of Grey Box Models (Refinement of Expert Knowledge-Based Fuzzy Systems)
- Open-Loop Control

For the first three application tasks, which are described in a more detailed way in the specific sections below, empirical tests on high-dimensional stationary and dynamic measurement data were carried out, so static fuzzy models as well as dynamic ones will be generated and adapted with the new method. A comparison of *FLEXFIS* and its batch learning variant (i.e. *extended genfis2*) will be investigated on various data sets possessing a different origin and quality with the following conventional batch learning methods for building up Takagi-Sugeno fuzzy systems (as they are all available in MATLAB's fuzzy logic toolbox):

- ANFIS: Adaptive Neuro-Fuzzy Inference Systems [Jan93, Jan92]
- FMCLUST: cluster-based learning method, see [Bab98] for a theoretical description and [Bab99] for a practical description of the usage of the method in MATLAB
- genfis2: cluster-based learning method, see [Chi94, YF94]

For the remaining two application tasks in above itemization only some aspects are discussed how and in which form the new method can be applied, but no evaluation results could be produced due to a missing implementation and missing input data.

2 Online (System) Identification

With online (system) identification it is meant to identify system behaviors and dependencies in online mode. Opposed to offline identification, where all the collected historic measurement data is sent as so-called training data into the model building algorithms, for an online identification framework measurements are recorded either bufferwise (i.e. in batch mode) or pointwise (i.e. in sample mode) with a certain frequency demanding several training steps in order to be up-to-date as soon as possible and therefore to be applicable at all (e.g. for online controlling or monitoring tasks). In principle, the online training steps can also be carried out by batch learning algorithms, but depending on the dynamics of the system the computation time can suffer in a way that the application of these methods gets unacceptable or even impossible. This fact will be outlined in this section, when comparing batch learning methods with the incremental learning method *FLEXFIS*. In all test examples a simulation of the online case could be achieved by setting the buffersize of the data processor to 1 for testing the sample mode case and to other values smaller than the number of records in the data matrices for testing various batch mode cases. In this way a pointwise or bufferwise loading of the data matrices stored on a hard-disk could be achieved.

2.1 The Framework

The generation as well as adaptation of the fuzzy models for high-dimensional measurement data is done automatically in a so-called model training job consisting of MATLAB-scripts: once having sent enough measurement data (sample rows) containing all measurement channels (columns) into the job, it tries to discover useable relationships between the channels. This is done by systematically taking each present channel as target channel and finding a good approximation for this channel by a subset of some other channels. Therefore, before the real training of the initial (for the online case) or complete (for the offline case) fuzzy models, subset selection or also called variable selection refer to [Mil02, GLK04], is applied in plain form, meaning no artificial regressors are generated from the original channels and no time shifts of the data matrix are performed. The later one is only needed for finding dynamic relationships and the data is not shifted in advance, the demand of artificial regressors is more foreseen for regression models and diminishes for fuzzy models, as they can approximate any real-occurring nonlinear relationship by using just the original channels [KKM97, Wan92]. In order to guarantee stable and correct approximations of the fuzzy models, some pre-filtering techniques for estimation and adaptation as proposed are applied. Whenever initial models are trained from the first k data points, the remaining $N - k$ data points are taken as input for adaptation in the case when applying incremental learning algorithms and are taken as input together with the first k points for re-estimation in the case when applying batch learning methods. N is the amount of data points which is available in sum within a certain test set applied for the evaluation.

Under these considerations for online identification the flowchart in Figure 1 is obtained which gives a comparison between online identification with adaptation (left path) and online identification with re-estimation (right path) as it needs to be done for the batch learning methods. Besides, it incorporates also the calculation of quality measures for the trained models. These quality measures are an important additional information about the reliability and trustability of the generated models, see also in order to be able to

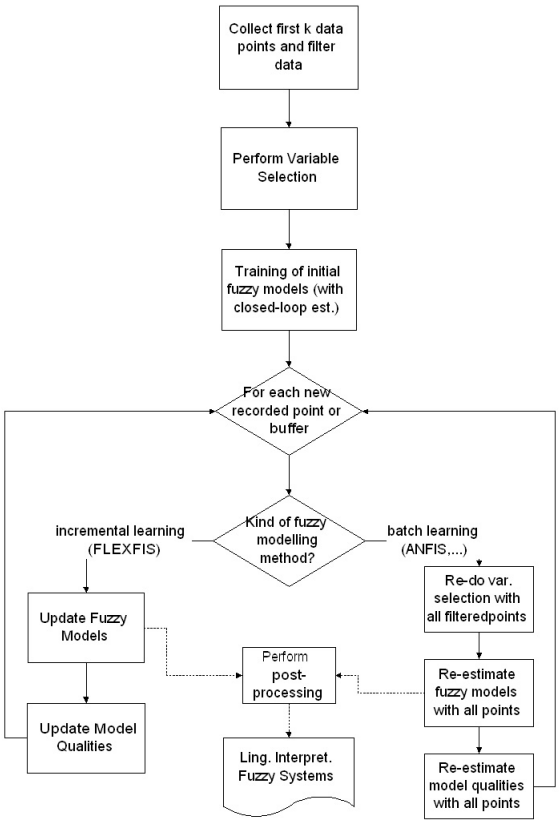


Figure 1: Online identification strategy with applying FLEXFIS as incremental learning method (left path) and applying any of the batch learning methods (right path)

distinguish between essential dependencies (in the case of a high quality) and non-usable relationships (in the case of a low quality). Indeed, after having generated the initial models, it can be considered to neglect the models with low quality for adaptation, but that would mean to exclude possible actual relationships, which come up after having recorded more measurements. Moreover, for some applications like for instance fault detection (see Section 3), they are even indispensable as they have to be incorporated into the solution for obtaining correct statements.

One weak point for the adaptation part (left path of the flowchart) may arise within the online identification approach as shown in Figure 1 and that concerns the variable selection, as it is only carried out once for the first k data points, whereas it is performed for every re-estimation step when applying the batch learning methods (right path). This pretended drawback is not a real one, because it does not spoil the quality of models significantly, as we will see in the tables below, where online identification results by using different modelling approaches will be listed and compared. Moreover, for low-dimensional systems with only a couple of measurement variables variable selection is not needed as all variables for approximating a specific target variable can be taken as input. Sometimes expert knowledge about the structures of the models exists, which also prevents the usage of variable selection.

2.2 Tests

In Table 1 a comparison between the different above itemized batch modelling methods and variants of incremental modelling is made. The comparison includes average model qualities of 62 trained up-to-5-

Table 1: Comparison of Fuzzy Model Building Methods with respect to quality and computation speed

Method	**Quality Training**	**Quality Test**	**Comp. Time offline**	**Comp. Time online 1 up-to-date every 100 p.**	**Comp. Time online 2 up-to-date each point**
FMCLUST	0.9272	0.902	6m 29s	62m 41s	Not possible
ANFIS	0.9110	0.872	5m 29s	>genfis2	Not possible
genfis2 conv.	0.9080	0.893	4m 28s	38m 31s	Not possible
genfis2 ext.	0.9110	0.904	3m 47s	34m 13s	Not Possible
genfis2 ext. conv. adapt	0.8319	0.818	3m 10s	3m 10s	3m 10s
FLEXFIS batch mode 100	0.8712	0.881	4m 36s	4m 36s	Not possible
FLEXFIS sample mode	0.8411	0.856	10m 57s	10m 57s	10m 57s

dimensional MISO (i.e. Multiple Input, Single Output) fuzzy models (i.e. maximal four inputs, always one output) on the training data set as well as a complete fresh fault-free test data set, obtained by testing a large diesel engine at an engine test bench. Indeed, originally the data set contained 1810 data points and 80 measurement channels, where 14 channels were completely neglected as input and target channels due to missing data or containing too many outliers. For each of the remaining channels, a fuzzy model was tried to be built up by a subset of the others. As different models possess different inputs and outputs with different ranges, absolute quality measures like average squared errors are not feasible for calculating an overall quality measure which comprehends the qualities of all trained models. Hence, *r-squared-adjusted* formula [LG02] normalized between 0 and 1, where near 1 means that a model with high quality could be achieved, was applied for each fuzzy model onto both, training data set and test data set. An average *r-squared-adjusted* defined by

$$R^2_{overall} = \frac{1}{m}\sum_{i=1}^{m} R^2_{adjusted\_i} \tag{1}$$

where $R^2_{adjusted\_i}$ is the *r-squared-adjusted* for the ith model, gives the overall quality of the method; overall *r-squared-adjusted* is calculated at the end of the complete identification process, hence in the case of the training data set for all 1810 data points, in the case of the test data set for 136 fresh fault-free points. This 136 fresh test points can be assumed as fault-free as they were extracted from a by an expert rated check data set. Usually, batch modelling methods achieve higher average *r-squared-adjusted* values than incremental learning methods, which is underlined in Table 1. This is because batch modelling methods always get in the complete information about the underlying dependencies to be approximated as all data points are fed into the algorithm. However, the obtained qualities on the training set of the with *FLEXFIS* adapted fuzzy models is surprisingly good, as it triggers only a 4.4% decrease when applying incremental learning in batch mode, respectively a 7.7% decrease when applying incremental learning in sample instead of batch modelling — for the four batch modelling approaches including also the *extended genfis2* approach explained in this thesis and also applied for initial fuzzy training before the incremental learning process starts the quality of trained models is nearly the same for this data set.

Moreover, for the qualities on a fresh test set (third column), adaptation and incremental learning approaches (especially in batch mode) can compete with generation approaches, which is quite a strong result for *FLEXFIS* and gets even stronger, if taking into account that it is an incremental training method and hence can deliver up-to-date models whenever they are needed. Indeed, this is also possible with the usage of batch modelling method by following the right path in the online identification scheme shown in Figure 1. But, column 5 demonstrates clearly, that when models are demanded to be up-to-date after each newly recorded 100 points, all the batch modelling methods take an almost 10 times! higher computation speed

Table 2: Comparison of *FLEXFIS* with conventional adaptation with respect to average model qualities obtained when training fuzzy models from online measurement data recorded at an engine test bench

Method	**Quality Training**	**Quality Test**	**Comp. Time**
genfis2 ext. conv. adapt	0.806	0.679	37 seconds
FLEXFIS batch mode 100	0.969	0.963	51 seconds
FLEXFIS sample mode	0.958	0.951	131 seconds

than the incremental learning method *FLEXFIS* in batch mode, which is triggered automatically when sending a buffer of points into the algorithm instead of single samples. This makes them hardly applicable in fast identification processes and even totally inapplicable for online identification tasks, where fuzzy models have to be up-to-date for each small buffer containing just a couple of points or even for each single point (see column 6). This is because, for batch modelling methods, single point update would mean training the model with k points (initial model), k+1, k+2, k+3, ..., k+N points. So, in the case of this large diesel engine data consisting of 1810 samples to perform 1710 re-estimations with 101, 102, 103, ..., 1810 data points, summing up to a computational effort of hours or days.

Comparing *FLEXFIS* in sample mode with the sample adaptation of the rules consequent parameters alone, denoted as '*genfis2 extended*, conv. adapt' in Table 1 and described in [LK03, Lug04], it has to be realized that especially for this data set the adaptation of consequent parameters is sufficient enough, as the quality of the models do not suffer significantly; moreover, it is three times faster than *FLEXFIS* in sample mode. The reason for this lies in the good distribution and density of the first k data points sent into initial model training, hence the input space for all models was covered well with data from the beginning → no data in a new region within the input space occurred for the whole adaptation process → there was no demand of shifting fuzzy sets or adjoining new sets in order to improve the quality of the models significantly. This circumstance can immediately change, when applying a different measurement plan, where the measurements are recorded in a different order. For instance, data from a BMW-diesel engine were recorded in an ascending order with respect to the two main influencing channels at an engine test bench, namely rotation speed and torque. Ascending order here means that a lattice over these two channels were laid and a stationary measurement (after reaching steady-state) were recorded for each knot point starting from point $(1200, 0)$ to $(1200, maxtorque)$, from $(1250, 0)$ to $(1250, maxtorque)$ up to $(5200, maxtorque)$, where $maxtorque$ is the maximal possible torque and depends on the actual rotation speed (rot-speed/torque-curve). This ascending measurement plan triggers more or less sorted data affecting the model qualities for the two incremental learning approaches significantly as shown in Table 2: the results obtained with *FLEXFIS* are pretty strong for both, training data set and test data set, whereas the models achieved by adaptation of the rules consequents alone are almost useless, which will be underlined even more for the application within an fault detection framework — see Section 3, where detection and overdetection rates with respect to some faulty and fault-free test data set are evaluated and compared amongst the different modelling methods. As in the case of an arbitrary online measurement process the distribution of the first k data points over the input space is usually not known a-priori, the best choice with respect to process security would be *FLEXFIS*. Indeed, in some cases the measurement plan can be elicited in a way that a good initial distribution of the operating channels is ensured, but that does not automatically ensure that all other measurement channels are affected by this strategy. So, extrapolation on newly recorded data can never be completely avoided for all fuzzy models.

The inner structure of the fuzzy models can be also quite important as an operator or expert may want to gain an insight into some system's relationships or also at least one relationship for a specific channel

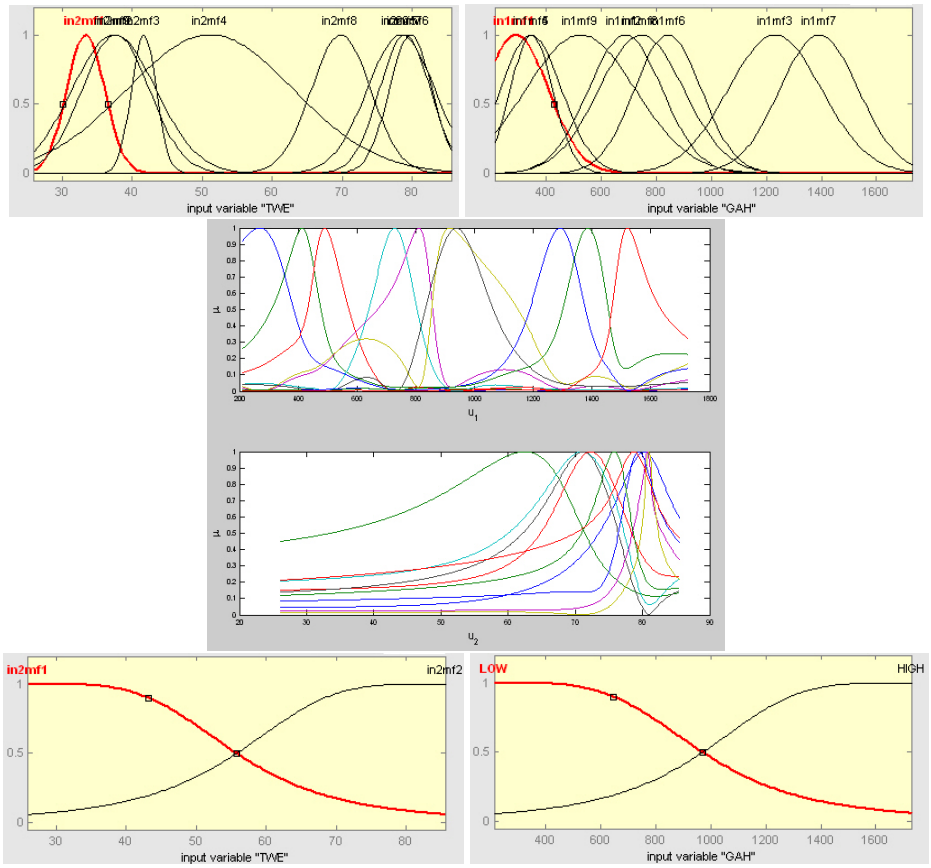


Figure 2: Fuzzy sets for two input channels for modelling air temperature after cooler when using (from top to bottom) the methods *FLEXFIS*, *FMCLUST* and *ANFIS*

he is interested in. For instance for the air temperature after the cooling unit at an engine a relationship depending on two other channels could be found with the help of the identification framework as shown in Figure 2 (Note: this is one model out of the 62). The input channels were the temperature of the water before the inflow (denoted as TWE) and the exhaust gas mass flow rate (denoted as GAH). In Figure 2 the obtained fuzzy sets for the input channels with the usage of the three methods (from top to bottom) *FLEXFIS*, *FMCLUST* and *ANFIS* are visualized. Indeed, the fuzzy sets obtained by *FLEXFIS* do not trigger a really good linguistic interpretable partition, but can be surely improved by applying an interpretability assurance method e.g. [SBV98], but was not pursued within this framework. Nevertheless compared to the fuzzy sets produced by *FMCLUST*, see middle image in Figure 2, they are wonderful. For these fuzzy sets which are not unimodal at all, a linguistic assurance method in post-processing manner can be hardly applied. *ANFIS*, however, provides the best linguistic insight into the system as shown in the lower image in Figure 2 as it produces exact fuzzy partitions i.e in each point the sum of membership degree is up to one and all the adjacent fuzzy sets overlap at the degree of 0.5. When taking into account, that *ANFIS* generates only four rules compared to nine respectively 10 rules when applying *FLEXFIS* respectively *FMCLUST*, it is no surprise that the he approximation accuracy is worst when applying *ANFIS*. However, it is a great surprise that *FLEXFIS* as incremental variant could achieve a higher approximation accuracy than (i.e. *genfis2 extended*) as batch variant by only incorporating one rule more, as it gets not all information in form of a complete data matrix in advance; a summary of achieved models structure and approximation accuracy especially for this model is given in Table 3. *ANFIS* has the nice property that it always generates real fuzzy partitions. This circumstance can trigger an explosion in the number of rules when the dimensionality of

Table 3: Characteristics of the air temperature after cooler models achieved with different fuzzy modelling methods

Method	**Fuzzy Sets**	**Number of Rules**	**Quality**
FMCLUST	unreadable	10	0.887
ANFIS	readable	4	0.851
genfis2 ext.	a bit readable	8	0.870
FLEXFIS sample mode	a bit readable	9	0.876

Table 4: Characteristics of the air/full ratio models achieved with different fuzzy modelling methods

Method	**Fuzzy Sets**	**Number of Rules**	**Quality**
FMCLUST	unreadable	10	0.996
ANFIS	readable	16	0.992
genfis2 ext.	a bit readable	5	0.979
FLEXFIS sample mode	a bit readable	6	0.936

the models gets higher. For instance in the case of a 4-dimensional fuzzy model, which was generated for modelling the air/full ratio, 16 rules appeared when using *ANFIS*, whereas *FMCLUST* and *genfis2 extended* needed only 10 respectively five rules for obtaining a similar model quality, see Table 4. *FLEXFIS* in sample mode generated six rules, but could compete with the other methods with respect to the model quality; however, the approximation accuracy is not so bad such that it could not be reasonably applied for online identification tasks. What also could be observed, was the fact, that *ANFIS* always generated exactly the same partition as shown in Figure 2 for all the 62 models, which is a precarious thing as it does not really imply a helpful linguistic information extraction from data (different data clouds should trigger different rules!). Maybe the reason for this lies in compensating the rule explosion when applying all fuzzy set combinations over the dimension. However, for a 10 dimensional modelling task *ANFIS* could not produce any result, as an 'out of memory' error occurred, see Section 4.

2 other tests were carried out based on data sets from the UCI repository [1], namely the so-called *housing* data and the *auto-mpg* data. The *housing* data, concerning housing values in suburbs of Boston, consists originally of 14 variables including one continuous class attribute, i.e. the target channel in our notations and 13 input attributes and contains 506 instances. The class attribute to be modelled from the other attributes stands for the median value of owner-occupied homes in 1000's. From the 506 instances 10% were selected randomly and used the the test data set, the remaining 90% of data were used as the training data set. For the evaluation the correlation coefficient between the predicted and the measured output was calculated as quality measure value on the test data in order to be comparable with the results in [HD04] obtained from other methods, such as *RENO*, *LAPOC* or *LAPOC-VS*. Obviously, a correlation coefficient near 1 denotes an almost identical function between the two outputs and therefore a good model accuracy, a value near 0 a useless model. It turned out, that only five input dimensions are sufficient for building up highly qualitative models, as adding more attributes did not increase the accuracy of the models significantly. In Table 5 the results obtained by the batch modelling methods as well as *FLEXFIS* are listed. From this table it can be realized that both *genfis2* variants produce the best results, which can also compete with the best methods in [HD04], whereas the results of *FLEXFIS* are not bad, when taking into account that it builds up the model sample per sample. The conventional adaptation of fuzzy systems, i.e. the adaptation of the rule consequent parameters alone, generates a completely useless model, as the first k data points do not

[1] http://www.ics.uci.edu/ mlearn/MLRepository.html

Table 5: Model accuracy and computation time of different methods when applying to *housing* data from UCI repository

Method	**Corr. Coeff.**	**Comp. Time in sec.**
FMCLUST	0.890	2.0
ANFIS	0.805	6.0
genfis2 conv.	0.922	0.8
genfis2 ext.	0.934	0.7
genfis2 ext. conv. adapt	0.450	2.2
FLEXFIS sample mode	0.903	4.0
LAPOC-VS	0.92-0.95	

cover the input range sufficiently. The slowest method of all, *ANFIS*, is also the worst of the batch learning methods with respect to the error on the test data, although on the training data the performance is quite good. This circumstance arises due to the strong overfitting effect, which is mostly caused when applying this method (probably because of the high number of rules), which is also underlined when looking at the results produced on the basis of *auto-mpg* data (see Table 6) and in Section 4. For the *auto-mpg*, which concerns city-cycle fuel consumption in miles per gallon and consists of eight attributes including one class attribute (target channel), namely the so-called 'miles per gallon', and seven input attributes, the same procedure as for the *housing* data was pursued. Again five input dimensions were enough to described the relationship significantly, the results are shown in Table 6: *FMCLUST*, *genfis2 extended* and *FLEXFIS* show almost similar results with respect to approximation accuracy and are even slightly better than the best method demonstrated in [HD04] (*LAPOC-VS*), which achieves a correlation coefficient of between 0.89 and 0.90 (this is known from due to a statement of the co-author). *ANFIS* and *genfis2 extended* with conventional adaptation are more or less forgettable.

3 Fault Detection

In the process and manufacturing industries, there has been a large push to produce higher quality products, to reduce product rejection rates, and to satisfy increasingly forceful safety and environmental regulations. Hence, the increasing complexity of measurement systems inside modern industrial processes with a rising amount of actuators and sensors demands automatic fault detection algorithms which can cope with a huge amount of variables and high-frequented dynamic data. Indeed, humans are being able to classify sensor signals by inspecting by-passing data, but this classifications are very time-consuming then and also have

Table 6: Model accuracy and computation time of different methods when applying to *auto-mpg* data from UCI repository

Method	**Corr. Coeff.**	**Comp. Time in sec.**
FMCLUST	0.917	1.3
ANFIS	0.730	5.0
genfis2 conv.	0.855	0.6
genfis2 ext.	0.916	0.5
genfis2 ext. conv. adapt	0.210	1.9
FLEXFIS sample mode	0.912	3.1
LAPOC-VS	0.89-0.90	

deficiencies because of underlying vague expert knowledge consisting of low-dimensional mostly linguistic relationships.

The IFAC Technical Committee SAFEPROCESS has defined a fault formally in the following way (also referenced in [IB96] and [CRB01]):

'Unpermitted deviation of at least one characteristic property or variable of the system from acceptable/usual/standard behaviour.'

Moreover, fault detection is formulated as:

'Determination of faults present in a system and time of detection.'

3.1 The Idea and Framework

In literature there can be found a lot of different basic approaches how to cope with the fault detection problem, which reaches from signal analysis in intelligent sensors [AA96] over a multi-classification based approach as demonstrated in [CRB01], which apply discriminant analysis functions for producing correct fault detection statements, to statistical-based fault detection applying *structural hypothesis tests* [Nyb99], which is theoretically grounded in classical hypothesis testing and propositional logic. In [HLS95, CP99, KKKC04] model-based fault detection is based on residual observer functions which entail a funded analytical theory. A model-based 'residual view' approach, where data-driven models are generated from some historic measurements or from first k online measurements and used as fault-free referrence situation for generating residuals, can be found in [SRG$^+$01] and [LPK$^+$95]. This was extended in [LKLG04] in order to take into account arbitrary many and any kind of models describing physical dependencies inside the system together with their qualities, to incorporate sensor inaccuracies for automatic thresholding and to calculate not only pure deviations from the model-based nominal case represented as a normalized residuals but also fault probabilities based on rated historic data. The framework of this approach is shown in Figure 3 and serves as the basis for the evaluation and comparison of the different fuzzy modelling methods with respect to fault detection. The fuzzy modelling methods are applied within block "Data-Driven Model Approximation", which is generally forseen for any kind of data-driven modelling algorithms. The framework provides the usage in an online application, where measurements are directly checked immediately after they are recorded. Based on the outcome of the fault detection process, data-driven and hybrid models are updated (in the case of a fault-free measurement) or not (when the measurement is faulty). For the evaluation and comparison of the fuzzy modelling methods as it will be demonstrated in Section 3.2, only the offline case of fault detection is examined, meaning fuzzy models are first generated from training data with the different batch learning and incremental learning approaches and afterwards a check data set containing fault-free points and different kinds of faults with different intensities is sent into the fault detection algorithm. From experience it is known, that this procedure is a good benchmark of the real online fault detection. The goal of fault detection can be formulated in the following way:

Goal 1. *Let $f_{1,k-1}, ..., f_{m,k-1}$ be m various multiple input single output models as describing some relationships inside an industrial process at an arbitrary point of time $k-1$ (instead of N as in the two equations), then newly recorded points $\vec{x}_{k,...,k+m}$ should be classified in a way by using these models as referrence situation, such that the number of correct classifications should be as high as possible.*

If only two classes of fault appearances occur, i.e one class representing the fault-free case and the other representing all possible faulty cases, the correct classifications can be split into correct detections of faults and into correct detection of no faults, both influencing the *detection rate* and *overdetection rate*. This splitting is done, because mostly detection and overdetection rates play different roles with different priorities (see Section 3.2). If more fault classes should be distinguished, fault patterns have to be trained from historic data and sent into the algorithm (see framework in Figure 3). Usually this fault patterns are not

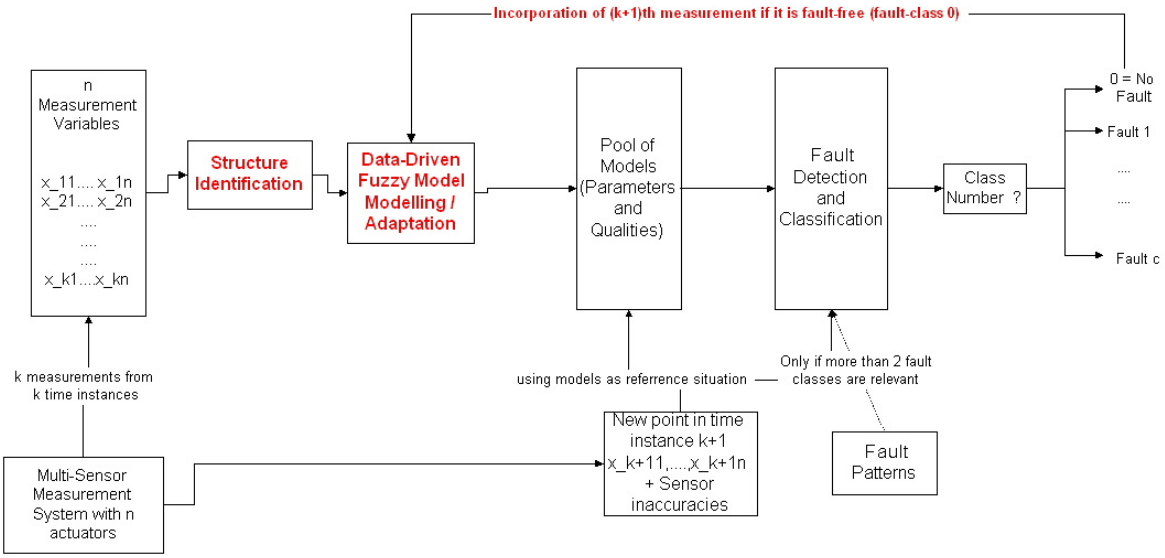


Figure 3: Fault Detection Scheme in an Online Measurement and Plausibility Check System

available and hence are neglected for the evaluation tests in this thesis. For details about the fault detection logic and also the application to analytical models refer to [LKLG04] and [GLG+04].

3.2 Tests and Results

Two different data sets were used for the evaluation of the different fuzzy modelling methods as well as of some other widley applied methods such as correlation or regression modelling strategies (see [LG02]) to demonstrate which kind of impact they have on the correctness of fault detection statements, i.e. on detection and overdetection rates:

- Real recorded measurement data at an engine test bench for a BMW diesel engine containing sensor inaccuracies and some white noise. The data were recorded from 32 measurement channels for 940 different operating points, each describing different settings of the two main influencing channels at an engine test bench, namely rotation speed and torque. As already mentioned in Section 2 the data were recorded in an ascending order with respect to rotation speed and torque. For each setting, reaching steady state was awaited, which took about 30 seconds, and then one stationary point was measured and stored. In view of the operating conditions, these data points were assumed to be free of faults. This data set was divided into a training data set containing 760 stationary points and a test data set containing 180 stationary points. Such a division was necessary in order to obtain an equal situation for all modelling methods. Some stationary points in the test data set were artificially corrupted with faults of different intensities, which were based on descriptions of real faults. A strict requirement was that all the faults with intensities greater than 50% have to be detected.
- Simulated data for a special diesel engine, containing no noise, together with two rated check data sets one containing faults with small deviations of 10%, the other containing faults with small deviations of 5% in some channels. Hence, this data set is a good benchmark for the sensitivity of the methods with respect to tiny deviations. The training data set contained around 1000 points, while the check data sets contained in sum 2052 points each, where approximatively half of them was faulty.

When applying the different fuzzy and other data-driven modelling methods to the BMW diesel data set, it turned out that more or less all the methods only produced between zero and two overdetections, when

Table 7: Comparison of detection rates among several data-driven model building methods based on measurements recorded for an automotive diesel engine

Method	**Det. Rate**	**Comment**
FMCLUST	73.86%	all big faults
ANFIS	73.86%	all big faults
genfis2 conv.	45.15%	not all big faults
genfis2 ext.	56.82%	all big faults
genfis2 ext. conv. adapt	37.50%	not all big faults
FLEXFIS batch mode	54.55%	all big faults
FLEXFIS sample mode	47.73%	all big faults

optimizing the threshold for keeping the amount of overdetections as small as possible. This is because operators would loose confidence in the system due to a high amount of overdetections. The only exception was the *FMCLUST* method which produced an overdetection rate of 8.70%. This could not be improved by tuning in a reasonable way without loosing too much correct detections, hence *FMCLUST* was more or less useless, as an overdetection rate of maximal 2% was demanded. In Table 7 only the detection rates caused by the different methods are demonstrated, the comment relies on the intensity of the faults and should reflect another point of view of usability. From these results two essential things can be concluded:

- *ANFIS* is superior to all other fuzzy modelling methods with respect to the detection rate, which is a quite surprising fact as for online identification and also for prediction as we will see in the next section *ANFIS* does not lead to better results. Actually, for applying the trained models onto fresh test data, as it is also the case for fault detection, it was one of the worst methods.
- The weakness of adaptation of rules consequents alone when the first k data points are ordered (which is the case for this data set and which was inspected also for the online identification application task) is again confirmed here. This weakness can be compensated with *FLEXFIS* in sample mode as an increase in the detection rate of 10% can be achieved, and even more with *FLEXFIS* in batch mode, where each buffer was filled with 100 points. In this case the detection rate of the corresponding batch learning method (*genfis2 extended*) could be almost obtained, which is quite strong as this method can be also applied within an online fault detection framework.

The sensitivity with respect to spoiled training data was observed for the BMW diesel data. The outcome of this inspection was that all methods performed in the same way when adding a fault level of 5% to a selection of points within the training data: the detection rate of all methods dropped to the half, while the overdetection rate stayed the same, except for *FMCLUST* method, which showed a higher sensitivity to the faults than the other methods, as the detection rate decreased from 73.86% to 13.64%, so by a factor of 6. Luckily, often pre-filtering techniques can be applied in order to deliver more correct models and to circumvent this drop of the detection rate [LERK03].

The results obtained for the simulated data look quite similar, see Table 8: *ANFIS* is again the strongest method, *FMCLUST* produces a large amount of overdetections and the conventional adaptation of rules consequent alone fails completely (no correct detections). For this data set *FLEXFIS* in sample mode is only feasible for detecting faults which possess a 10% deviations from their correct values. More or less the results obtained with the BMW diesel data could be confirmed. For this data set also analytical models were available covering all channels in the data set, hence the data-driven fuzzy modelling approaches could be benchmarked against them. Surprisingly, neither for faults with 10% deviation nor for faults with 5% deviation, analytical models produced stronger results than all the fuzzy modelling approaches. Furthermore, in the case of faults with 10% deviation analytical models were even weaker than the incremental learning method *FLEXFIS* in batch as well as sample mode.

Table 8: Comparison of detection rates among several data-driven model building methods based on simulated engine data

Method	**Dev**	**Det. Rate**
FMCLUST	10%	85.34%
	5%	70.76%
ANFIS	10%	87.27%
	5%	74.85%
genfis2 conv.	10%	21.69%
	5%	18.20%
genfis2 ext.	10%	49.59%
	5%	24.13%
genfis2 ext. conv. adapt	10%	0.00%
	5%	0.00%
FLEXFIS batch mode	10%	46.95%
	5%	22.60%
FLEXFIS, sample mode	10%	37.58%
	5%	4.09%
Analytical models	10%	35.13%
	5%	23.92%

4 Prediction

4.1 General Information

With prediction in general or multiple step ahead prediction in particular it is meant that previous input values of a process are given at time instants $k - i$, from which the process output at time instant k should be predicted. If within the list of previous input values the minimal i is equal to 1, we speak about one step ahead prediction, if it is 2 about two step ahead prediction and so on. Prediction can be necessary, if a reaction on system input variables has not an influence on the system output(s) immediately at the next time step, but in some future time steps. Typical example of prediction tasks are short-term stock market or weather forecast. In general predictive models can be applied as time series models [aRD96] or dynamic models (e.g. ARX, NARX, ARMAX etc. models - see [Lju99]) in any industrial application. A time delay recognition for eliciting the prediction horizon as well as sufficient time delays of the underlying process in order to produce a high-qualitative predictive model can be performed with variable selection methods, see [GLK04]. For this, various shifts in the inputs has to be carried and stored as additional regressors in the regression matrix.

A prediction framework is shown in Figure 4, which should be more or less self-explainable. Once having performed down-sampling of high frequented dynamic measurement data and having generated time delayed regressors from the original set of channels appearing in the data matrix, time delay recognition is carried out with the usage of variable selection methods, which yields the input structure for the upstreamed fuzzy modelling algorithm. For the fuzzy modelling task batch learning as well as incremental learning variants can be applied. The generated fuzzy model is sent into the prediction process, where new input states are processed through the inference mechanism of the fuzzy prediction model obtaining predictive values for the output. Building up a predictive model in incremental manner can be in principal done, too, but this is not required often, because the predictive model should replace the output channel. In the next section results for two dynamic modelling tasks are presented also including tests with incremental learning methods, as they can be applied also in batch manner, of course.

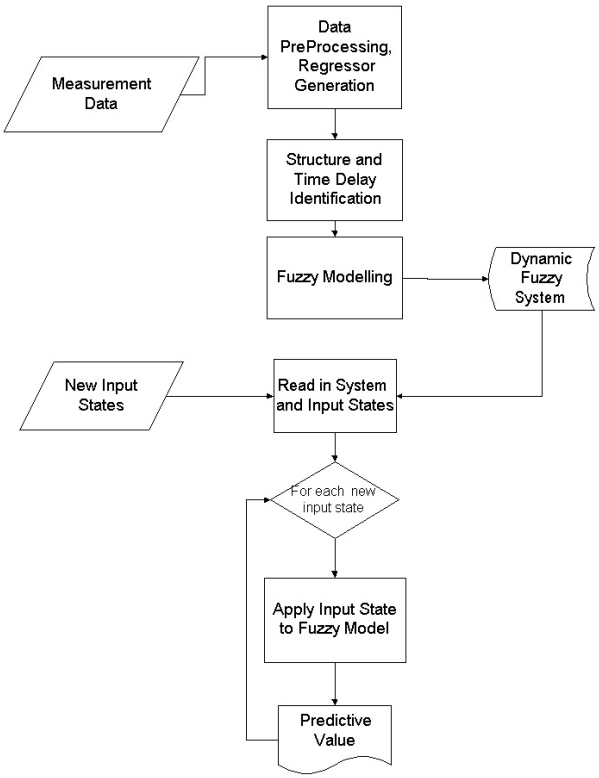


Figure 4: Prediction framework

4.2 Tests and Evaluation

The first tests were performed on the basis of the famous Wang data, which firstly appeared in his book about adaptive fuzzy system and control [Wan94] and which was quite often used by several researchers within the fuzzy community in order to benchmark their methods for fuzzy system training against some others. A summary of the outcomes of the most important methods is demonstrated in [PPPS04], where the methods are compared with respect to three criteria, namely mean squared error on the training data, mean squared error on the test data and the number of rules included in the identified Takagi-Sugeno fuzzy model. These criteria were also checked when applying the batch learning as well incremental learning methods treated in this thesis. The identification task is to build a model for the nonlinear plant, described by the following formula:

$$y(k) = f(y(k-1), y(k-2)) + u(k) \tag{2}$$

with

$$f(y(k-1), y(k-2)) = \frac{y(k-1)y(k-2)(y(k-1)-0.5)}{1+y^2(k-1)+y^2(k-2)}$$

and $u(k)$ a random input signal uniformly distributed in $[-1.5, 1.5]$. 200 data points were generated for training the Takagi-Sugeno fuzzy model, 200 additional data points with an input signal $u(k) = \sin(\frac{2\pi k}{25})$ for the testing in order to validate the overfitting effect of the methods. For being able to compare the methods treated in this thesis with the methods in [PPPS04] with respect to the mean squared error, a normalization of the data to the interval $[0, 1]$ have been applied. The performance of the methods are shown in Table 9. Obviously the best performing method on the training data is *ANFIS* as it triggers the smallest mean squared error (*MSE*). However, with eight rules and an average approximation error on the test data, which is the essential point as a strong overfitting should always be prevented, it should be not the

Table 9: Comparison of fuzzy modelling methods applied to nonlinear dynamic plant data

Method	**MSE Training**	**MSE Test**	**Number of Rules**
FMCLUST	$2.59 * 10^{-5}$	$3.4 * 10^{-3}$	10
	$9.68 * 10^{-5}$	$3.7 * 10^{-3}$	5
ANFIS	$7.7 * 10^{-6}$	$4.0 * 10^{-3}$	8
genfis2 conv.	$1.0 * 10^{-4}$	$3.2 * 10^{-3}$	9
	$3.86 * 10^{-4}$	$5.3 * 10^{-3}$	6
genfis2 ext.	$3.06 * 10^{-4}$	$3.4 * 10^{-3}$	11
	$5.53 * 10^{-4}$	$1.8 * 10^{-3}$	8
	$2.0 * 10^{-3}$	$1.5 * 10^{-3}$	4
FLEXFIS sample mode	$1.5 * 10^{-3}$	$7.8 * 10^{-3}$	8
	$1.7 * 10^{-3}$	$1.2 * 10^{-3}$	3

first choice. Surprisingly, the smallest error on the test data is achieved by the incremental learning method *FLEXFIS* in sample mode and this with only three rules, the smallest amount for all tests!

Another test was performed on building an as good as possible model for a dynamic relationship in form of a prediction model, whose target channel was defined by the emission channel NOX for an engine. This task emerged from the demand of saving expenses on a measurement sensor for NOX at an engine test bench by describing this channel through a formula of some others, which have to be measured anyway. It turned out, that at least 10 inputs (some original channels and their time delays) were needed in order to obtain an approximation quality higher than 0.9, see Table 10. The input channels for approximating NOX at time instant k consisted of the following list of channels (in the order they were selected):
Te = Engine Output Torque
P2offset = Pressure in Cylinder number 2
N = Engine Speed
Alpha = Accelerator Pedal
Tgas = Exhaust Temperature
Nd = Speed of the Dynanometer
P1offset = Pressure in Cylinder number 1
COL = CO value
together with their appropriate delays yielding a dynamic model in form of a one-step-ahead prediction of NOX described by

$$\begin{aligned} NOX(k) = & f(\text{Te}(k-5), \text{P2offset}(k-5), \text{N}(k-4), \text{Alpha}(k-6), \text{Tgas}(k-9), \text{Nd}(k-6), \\ & \text{Te}(k-4), \text{P1offset}(k-7), \text{COL}(k-1), \text{COL}(k-5)) \end{aligned} \tag{3}$$

One step ahead prediction because from the measured points at time instant $k-1$ and further back the NOX value at time instant k can be predicted. If for example the 9th channel $\text{COL}(k-1)$ would be neglected it would lead to a four-step-ahead prediction as the lowest time shift would be $k-4$. Originally, the input data matrix consisting of 6700 samples which were recorded with a certain frequency. This frequency was too high in order to obtain feasible time delays of the original channels, as shifts up to 100 steps had to be carried out producing 1600 additional channels out of the 16 original channels (for each channel 100 different shifts: $k-1, k-2, ..., k-100$) in order to get a good approximation for NOX. Thus, it turned out that a simple down-sampling by taking just each 10th point and throwing away all the others yielded a sufficient resolution. Hence, the input matrix was reduced to 670 sample, where then a time shift up to 10 was sufficient causing a manageable amount of 160 channels and finally 660 samples (due to this shift the first 10 samples needed to be cut out. It is obvious that a delay in the new data matrix of $k-l$ belongs to a delay of $k-10l$ in the original one and vice versa. With the knowledge about the chosen frequency for

Table 10: Comparison of Fuzzy Modelling Methods with Regression Strategies when approximating a dynamic high-dimensional relationship

Method	**Quality Training**	**Quality Test**
FMCLUST	0.909	0.868
ANFIS	0.911	0.842
genfis2 conv.	0.908	0.810
genfis2 ext.	0.900	0.890
genfis2 ext., conv. adapt. sample mode	0.832	0.818
FLEXFIS batch mode 100	0.871	0.836
FLEXFIS sample mode	0.841	0.829

sampling, namely 10 Hz per second, we can conclude to the real absolute delay for the impact of the input channels on NOX.

When reducing the input dimensionality to 3 for *genfis2 extended*, an readable fuzzy system for NOX was the outcome with the following four rules:

Rule 1: If Te$(k-5)$ is LOW and P2offset$(k-5)$ is HIGH and N$(k-4)$ is LOW
Then NOX$(k) = f_1$(Te$(k-5)$, P2offset$(k-5)$, N$(k-4)$)

Rule 2: If Te$(k-5)$ is MEDIUM and P2offset$(k-5)$ is MEDIUM and N$(k-4)$ is MEDIUM
Then NOX$(k) = f_2$(Te$(k-5)$, P2offset$(k-5)$, N$(k-4)$)

Rule 3: If Te$(k-5)$ is MEDIUM and P2offset$(k-5)$ is VERYHIGH and N$(k-4)$ is HIGH
Then NOX$(k) = f_3$(Te$(k-5)$, P2offset$(k-5)$, N$(k-4)$)

Rule 4: If Te$(k-5)$ is HIGH and P2offset$(k-5)$ is LOW and N$(k-4)$ is MEDIUM
Then NOX$(k) = f_4$(Te$(k-5)$, P2offset$(k-5)$, N$(k-4)$)

where the fuzzy sets for each input channel together with their linguistic meanings are visualized in Figure 5 and f_1, f_2, f_3 and f_4 are linear consequent functions, hence hyper-planes. Taking into account a sampling frequency of 10 Hz and a down-sampling rate of 10 to 1, Te$(k-5)$ denotes linguistically 'the engine output torque five seconds ago'. In the same way P2offset$(k-5)$ denotes 'Pressure in Cylinder number 2 five seconds ago' and N$(k-4)$ denotes 'Engine speed five seconds ago'. However, the model quality decreased from 0.9 to 0.81 when reducing input dimensionality from 10 to 3. When increasing the dimension to 5, an increase to 0.84 can be achieved, but the rules get less transparent, as the premise parts get more complicated. Hence, this results again demonstrates that it is always a tradeoff between linguistic interpretability and approximation accuracy.

5 Generation of Grey Box Models (Refinement of Expert Knowledge-Based Fuzzy Systems)

In this section another application task for incremental learning is treated, the so-called generation of grey box models, also called hybrid models. A white box fuzzy model is initialized from linguistic expert knowledge about some underlying relationships of a process. As usually such a white box fuzzy model is obtained from rather vague experience when working with a system, it should be refined with measurement data either in offline or in online mode, in order to ensure a higher process security. Hence an adaptation of some of the parameters from the expert knowledge-based fuzzy models should be carried out with measurements.

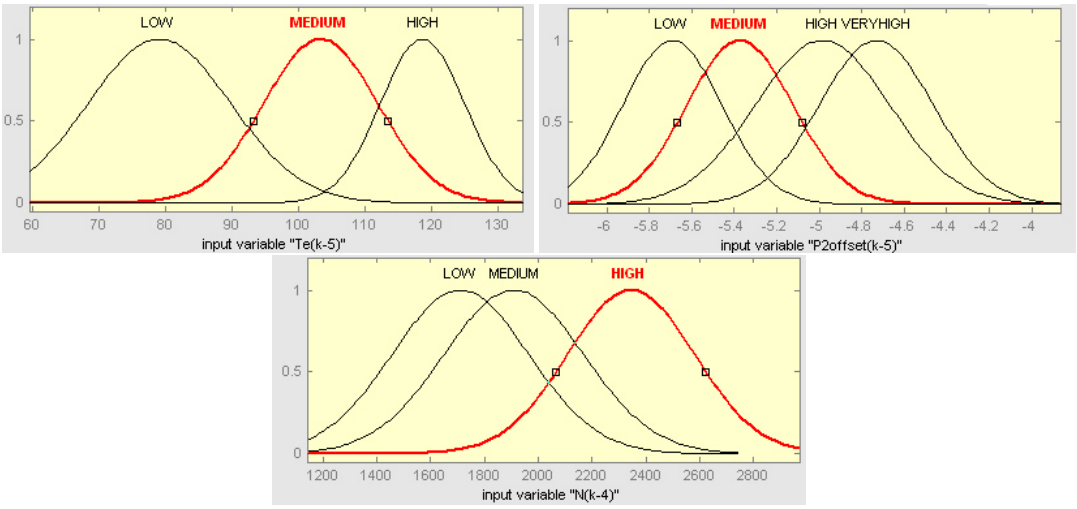


Figure 5: Fuzzy sets for the three input channels Te, P2offset and N of a fuzzy prediction model for emission channel NOX

Sometimes it is sufficient to adapt the rules consequent parameters alone, which has the favorable effect that the linguistic interpretable input partitions (as defined originally from experts) remain. However, mostly the expert has not all possible system conditions together with its correct reaction in his mind, so an adjoining of new rules for incorporating new conditions is beneficial. Here a flowchart for the generation of grey box models is demonstrated in Figure 6, which should give the basic idea of grey box generation of fuzzy models. As no concrete application case in form of both, vague linguistic expert knowledge and measurement data, was available, no tests were carried with the usage of this flowchart so far.

The first two blocks, i.e. collecting expert knowledge and coding these expert knowledge into a fuzzy system, denote the offline part of this flowchart, where the highest development effort lies in collecting this expert knowledge, as this is usually carried out within a lot of discussion and meetings.

In the block "Build a Fuzzy Model by Coding Expert Knowledge" generally every kind of fuzzy model can be built up, hence Mamdani-type or Takagi-Sugeno-type models also with the usage of different t-norms and fuzzy sets. This eventually causes some necessary modifications to the *FLEXFIS* approach, which will be also discussed at the end of this thesis in Section 7. However, if choosing Takagi-Sugeno type fuzzy models for the coding of linguistic expert knowledge, *FLEXFIS* can be directly applied within the grey box generation framework.

The block "Read in Fuzzy Model, Extract Parameters to Form Clusters, ..." requires a software technical implementation step which was not needed in the case of online identification, as all was completely generated out of data from scratch. This performs first the back transferring of the parameters in the fuzzy sets and the rule structure to form the partition in the cluster space, which from the mathematical point of view more or less straightforward as

- Amount of clusters = amount of rules
- Each rule's premise form one cluster where
- Each premise part corresponds to a fuzzy set whose center can be assigned to the corresponding entry in cluster center vector
- The width of the fuzzy sets can also be extracted and stored together with cluster centers

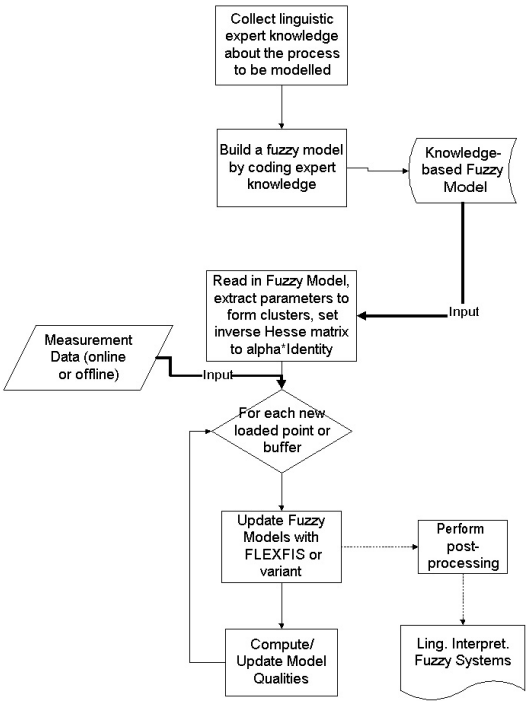


Figure 6: Grey box fuzzy modelling strategy with applying *FLEXFIS* (or some of its variants) as incremental learning algorithm

and second the initial setting of the inverse Hesse matrix for the *RWLS*, as obviously no inverse Hesse matrix will be set initially by an expert or operator. The best initial setting is αI with α a big positive integer, see [Lug04].

The block "Update Fuzzy Models with *FLEXFIS* or a Variant" incorporates the correct adaptation strategy with respect to the type of the knowledge-based fuzzy system and also the wished kind of adaptation (rules consequent only, rule structure learning only or combined, etc.).

6 Open-Loop Control

Most advanced control design schemes rely on a model of the process, and many even require information such as uncertainties of the model, i.e. an estimation of the model quality. This is because due to the understanding and identification of the underlying process, a sufficient insight into the process can be obtained. This also means that the process models should be interpretable either in physical sense or at least in linguistic sense. For analytical modelling approaches this is mostly guaranteed anyway, in the case of the usage of data or expert knowledge fuzzy models are a good option for accomplishing that. Whenever no analytical description and no expert knowledge are available at all or whenever expert knowledge is available only for some special operating cases, the usage of measurement data for identifying the process model is favorable. As usually for highly time-variant systems the behavior of some dependencies between variables can be not described completely and uniquely from collected historic measurements, an adaptation of the process model with newly recorded measurements is mandatory. Additionally, it guarantees an improvement of the process security. If an automatic fuzzy model inversion is downstreamed leading to a *feed-forward controller* as shown in Figure 7, the controller design does not need to be carried out manually.

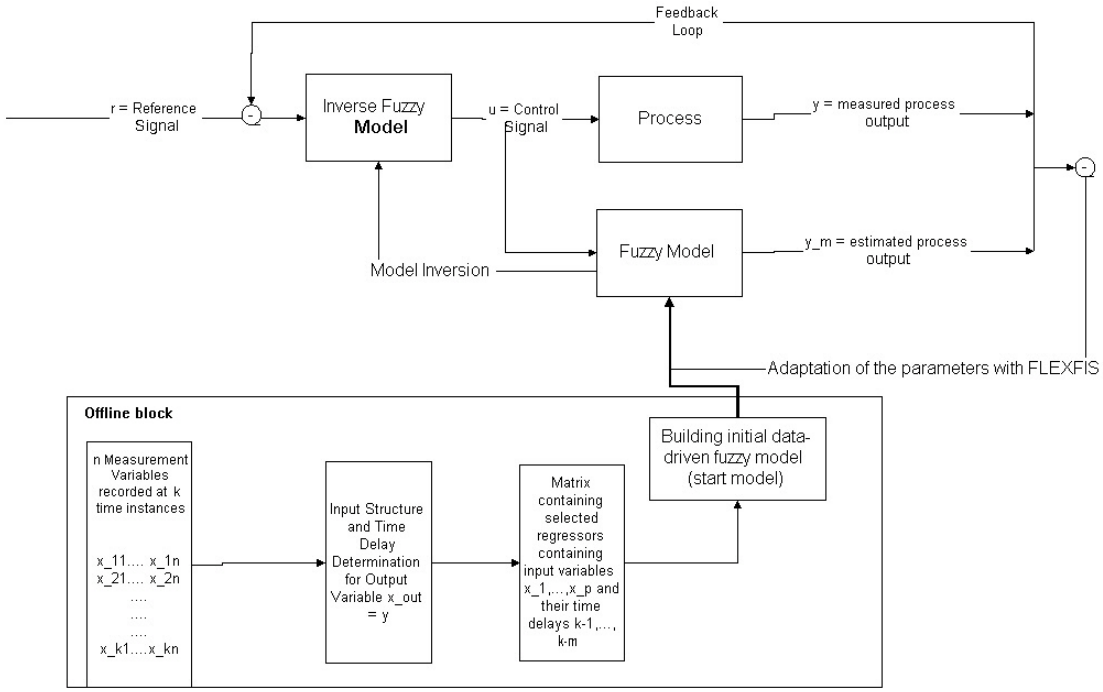


Figure 7: Open-loop control framework by using *FLEXFIS* for updating the process model and by applying fuzzy model inversion

In this approach of a control system fuzzy models or rule bases are not used as batch building components for conventional P, PI or PID controllers as proposed in [KWHW00] or incorporated into an objective function which characterizes the impact of future control signal values onto values of state variables and whose minimization is intended — see [FFNI00, LP97]. Besides the online adaptation concept of fuzzy models, however, it includes the possibility of generating the fuzzy models from high dimensional dynamic data — as a structure and time delay identification procedure is upstreamed — and to perform control operations directly by inverted fuzzy models. The last point omits the need of a complex objective function to be minimized and triggers a linguistically interpretable and understandable fuzzy controller. Moreover, it is well known that the use of an inverse model possesses the advantages of open-loop control, i.e. inherent stability and 'perfect' control with zero error both in dynamic transients and in steady states [EMP86]. In the case, when linguistic expert knowledge is available, the offline block in Figure 7 can be omitted by coding directly this expert knowledge into a fuzzy model, which can be refined through the adaptation process with new incoming measurements, see Section 5. Of course, *FLEXFIS* can be also applied to other controlling techniques (when no model inversion is possible at all), whenever a process model is incorporated into the whole control loop and should be updated and extended (e.g. for predictive control).

7 Conclusion

For all application tasks, an evaluation of the method was carried out based on real-life measured data or simulated data from already well-known models. The evaluation results were compared with those of well-known batch learning approaches for training Takagi-Sugeno fuzzy systems due to some criteria such as approximation accuracy or computational effort and should underline the strengths and weaknesses of *FLEXFIS* (also opposed to other methods). In table 11 a summary is given, which states a kind of rating for each method, where three entries are possible for each row: + (=good), ○ (=average) and -

Table 11: Comparison of fuzzy model building methods with respect to some important applicability criteria

Criteria	*FMCLUST*	*ANFIS*	*genfis2 conv*	*genfis2 ext*	*FLEXFIS*
Approximation Quality	+	-	○	+	○
Computational Effort (online)	-	-	-	-	+
Process Security	-	-	○	+	+
Interpretability and Transparency	-	+	○	○	○
Applicability in Online Identification	○	-	○	○	+
Applicability in Fault Detection	+	+	○	○	○
Applicability in Prediction	+	+	○	+	+
Overall	0	-1	-1	2	4

(=bad). Additionally, an overall score for each method is stated in the last row of the table, which has to be relativized a bit, because the ratings of all properties are simply summed up with equal weighting by taking into account that a + is counted as +1, a ○ counted as 0 and - counted as -1. In this sense, no priority is given, which should be done when some demands are more important than others: for instance in the case of pure fault detection the insight and also the quality is less important, the most important thing for this task is that the fault detection statements themselves are correct. In the case of the computation effort the online mode is meant where models should be updated from time to time, for the offline case the computational effort is mostly not an essential criterium for a selection of the method. *ANFIS* gets a minus regarding to approximation quality because of the strong overfitting effect it usually possesses, which can be seen from all the approximation accuracies results based on fresh test data. Regarding process security, some runtime problems occur for *ANFIS* and also *FMCLUST* during evaluation, whereas for *genfis2 conventional* indeed no problems occurred, but no ridge regression for numerical stabilization and a pre-filtering scheme for deleting faults as it is done for *genfis2 extended* and *FLEXFIS* are integrated into the algorithm, hence the rating is maximal 0. When looking at the applicability for the different application tasks, it can be concluded that for online identification in most cases *FLEXFIS* should be chosen, whereas in the some special case, namely whenever the distribution of the first k data points is well spread over the input space alternatively *genfis2 extended* together with the adaptation of rule consequent parameters alone should be chosen, as it is much more faster than *FLEXFIS*. However, an entire covering of the input space with the first k data points can be guaranteed only very rarely, namely in the case if a very good expert knowledge about the involved measurement variables is available ($\rightarrow$ the measurement plan can then be constructed on the basis of this knowledge in an appropriate way). For offline identification *genfis2 extended* seems to be the strongest variant. Moreover, for offline fault detection the best choice is *ANFIS*, whereas for online fault detection the best choice is *FLEXFIS* in batch mode, as *ANFIS* is not capable incremental learning steps. In the case of prediction, so the application of the modelling variants to dynamic models, obviously *genfis2 extended* or *FLEXFIS* in sample mode are favorable.

References

[AA96] S. Alag and A.M. Agogino. A methodology for intelligent sensor measurement, validation, fusion, and fault detection for equipment monitoring and diagnostics, 1996.

[aRD96] P.J. Brockwell abd R.A. Davis. *Introduction to Time Series and Forecasting*. Springer Verlag, Heidelberg Berlin New York, 1996.

[Bab98] R. Babuska. *Fuzzy Modeling for Control*. Kluwer Academic Publishers, Boston, 1998.

[Bab99] R. Babuska. Fuzzy modeling and identification toolbox - for use with matlab, 1999.

[Chi94] S. Chiu. Fuzzy model identification based on cluster estimation. *Journal of Intelligent and Fuzzy Systems*, 2(3), 1994.

[CP99] J. Chen and R.J. Patton. *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061, 1999.

[CRB01] L.H. Chiang, E.L. Russell, and R.D. Braatz. *Fault Detection and Diagnosis in Industrial Systems*. Springer Verlag London Berlin Heidelberg, London, Great Britain, 2001.

[EMP86] C. Economou, M. Morari, and P. Palsson. Internal model control 5. extension to nonlinear systems. *Ind. Eng. Chem. Process Des. Dev.*, 25:403–411, 1986.

[FFNI00] A. Fink, M. Fischer, O. Nelles, and R. Isermann. Supervision of nonlinear adaptive controllers based on fuzzy models. *Journal of Control Engineering Practice*, 8:1093–1105, 2000.

[GLG+04] J. Galindo, J.M. Lujan, C. Guardiola, E. Lughofer, and E.P. Klement. Fault detection in engine measurement systems by a model-based approach. In *Proc. SAE 2004*. SAE, July 2004.

[GLK04] W. Groißböck, E. Lughofer, and E.P. Klement. A comparison of variable selection methods with the main focus on orthogonalization. In M. Lopéz-Díaz, M.Á. Gil, P. Grzegorzewski, O. Hryniewicz, and J. Lawry, editors, *Soft Methodology and Random Information Systems*, Advances in Soft Computing, pages 479–486. Springer, Berlin, Heidelberg, New York, 2004.

[HD04] J. Himmelbauer and M. Drobics. Regularized numerical optimization of fuzzy rule bases. In *Proceedings of FUZZ-IEEE 2004*, Budapest, Hungary, 2004.

[HLS95] P. Hsu, K. Lin, and L. Shen. Diagnosis of multiple sensor and actuator failures in automotive engines. *IEEE Transactions on Vehicular Technology*, 44(4):779–789, November 1995.

[IB96] R. Isermann and P. Ball. Trends in the application of model-based fault detection and diagnosis of technical processes. In *Proc. of the 13th IFAC World Congress, volume N*, pages 1–12. IEEE Press, 1996.

[Jan92] J.-S.R. Jang. Self-learning fuzzy controllers based on temporal backpropagation. *IEEE Trans. on Neural Networks*, 3:714–721, 1992.

[Jan93] J.-S.R. Jang. ANFIS: Adaptive-network-based fuzzy inference systems. *IEEE Trans. Syst. Man Cybern.*, 23:665–685, 1993.

[KKKC04] J. Korbicz, J.M. Koscielny, Z. Kowalczuk, and W. Cholewa. *Fault Diagnosis - Models, Artificial Intelligence and Applications*. Springer Verlag, Berlin Heidelberg, 2004.

[KKM97] E.P. Klement, L. Koczy, and B. Moser. Are fuzzy systems universal approximators? *General Systems*, 28:259–282, 1997.

[Koz92] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992.

[Koz94] J.R. Koza. *Genetic Programming II*. The MIT Press, Cambridge, Massachusetts, 1994.

[KWHW00] S.J. Kang, C.H. Woo, H.S. Hwang, and K.B. Woo. Evolutionary design of fuzzy rule base for nonlinear system modelling and control. *IEEE Trans. on Fuzzy Systems*, 8(1):37–45, 2000.

[LERK03] E. Lughofer, H. Efendic, L. Del Re, and E.P. Klement. Filtering of dynamic measurements in intelligent sensors for fault detection based on data-driven models. In *Proceedings IEEE CDC — IEEE CDC Conference, Maui, Hawaii*, pages 463–468, December 2003.

[LG02] E. Lughofer and W. Groißböck. Generating correlation and regression models from high dimensional measurement data - advanced aspects, strategies and validation. Technical Report FLLL-TR-0212, FLLL, A-4232 Hagenberg, Austria, September 2002.

[Lju99] L. Ljung. *System Identification: Theory for the User*. Prentice Hall PTR, Prentic Hall Inc., Upper Saddle River, New Jersey 07458, 1999.

[LK03] E. Lughofer and E.P. Klement. Online adaptation of Takagi-Sugeno fuzzy inference systems. In *Proceedings of CESA'2003—IMACS Multiconference*, Lille, France, 2003. CD-Rom, paper S1-R-00-0175.

[LK04a] E. Lughofer and E.P. Klement. Flexfis: A variant for incremental learning of takagi-sugeno fuzzy systems. Technical Report FLLL-TR-0402, Fuzzy Logic Laboratorium Linz-Hagenberg, A-4232 Hagenberg, Austria, 2004.

[LK04b] E. Lughofer and E.P. Klement. Premise parameter estimation and adaptation in fuzzy systems with open-loop clustering methods. In *Proceedings of FUZZ-IEEE 2004*, Budapest, Hungary, 2004.

[LKLG04] E. Lughofer, E.P. Klement, J.M. Lujan, and C. Guardiola. Model-based fault detection in multi-sensor measurement systems. In *Proceedings of IEEE IS 2004*, pages 184–189, Varna, Bulgaria, 2004.

[LP97] R.K. Lim and M.Q. Phan. Identification of a multistep-ahead observer and its application to predictive control. *Journal of Guidance, Control and Dynamics*, 20(6):1200–1206, 1997.

[LPK+95] E.G. Laukonen, K.M. Passino, V. Krishnaswami, G.-C. Lub, and G. Rizzoni. Fault detection and isolation for an experimental internal combustion engine via fuzzy identification. *IEEE Trans. on Control Systems Technology*, 3(9):347–355, September 1995.

[Lug04] E. Lughofer. Data-driven adaptation of fuzzy models in open-loop part 1: Adaptation of rules consequent parameters. Technical Report FLLL-TR-0401, Fuzzy Logic Laboratorium Linz-Hagenberg, A-4232 Hagenberg, Austria, 2004.

[Mil02] A. Miller. *Subset Selection in Regression Second Edition*. Chapman and Hall/CRC, Boca Raton, Florida 33431, 2002.

[Nyb99] M. Nyberg. *Model Based Fault Diagnosis, Methods, Theory, and Automotive Engine Application*. PhD thesis, Department of Electrical Engineering Linköping University, SE-581 83 Linköping, Sweden, May 1999.

[PPPS04] P.C. Panchariya, A.K. Palit, D. Popovic, and A.L. Sharma. Nonlinear system identification using takagi-sugeno type neuro-fuzzy model. In *Proc. 2nd International IEEE Conference Intelligent Systems 2004*, pages 76–81. IEEE Press, 2004.

[SBV98] M. Setnes, R. Babuska, and H.B. Verbruggen. Complexity reduction in fuzzy modeling. *Mathematics and Computers in Simulation*, 46(5-6):509–518, 1998.

[SRG+01] A. Schrempf, L. Del Re, W. Groißböck, E. Lughofer, E.P. Klement, and G. Frizberg. Automatic engine modelling for failure detection. In *Proc. 2001 ASME International Mechanical Engineering Congress and Exposition*, March 2001.

[Wan92] L.X. Wang. Fuzzy systems are universal approximators. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, pages 1163–1169, 1992.

[Wan94] L.X. Wang. *Adaptive Fuzzy Systems and Control*. Prentice-Hall, Englewood Cliffs NJ, 1994.

[YF94] R. Yager and D. Filev. Generation of fuzzy rules by mountain clustering. Technical Report MII-1318R, Machine Intelligence Institute, Iona College, New Rochelle, NY 10801, 1994.