Tiago Miguel Brites Oliveira

Licenciado em Engenharia Electrotécnica e de Computadores

Recursive Neuro Fuzzy Techniques for Online Identification and Control

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores

Orientador : Luís Filipe Figueira Brito Palma, Prof. Auxiliar, FCT/UNL

Co-orientador : Paulo José Carrilho de Sousa Gil, Prof. Auxiliar, FCT/UNL

Júri:

Presidente: Prof. Doutor Luís Filipe dos Santos Gomes - FCT/UNL

Arguente: Prof. Doutor José António Barata de Oliveira - FCT/UNL

Vogais: Prof. Doutor Luís Filipe Figueira de Brito Palma - FCT/UNL
Prof. Doutor Paulo José Carrilho de Sousa Gil - FCT/UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março, 2013

Recursive Neuro Fuzzy Techniques for Online Identification and Control

Dedico a todos os meus colegas e família mas principalmente, aos meus avós e mãe pelo suporte em todos os níveis. Um carinho especial para a Cláudia, por todo o encorajamento, compreensão e apoio durante o meu percurso académico.

Acknowledgements

Quero exprimir os meus agradecimentos ao orientador da tese Luís Palma e ao co-orientador Paulo Gil, pelo suporte e orientação a nível tanto técnico como teórico.
Agradecer em geral, à Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, com principal ênfase ao Departamento de Engenharia Eletrotécnica, por todo o conhecimento transmitido. Um agradecimento especial à área de Controlo e Decisão, pela oportunidade concedida.

Abstract

The main goal of this thesis will be focused on developing an adaptative closed loop control solution, using fuzzy methodologies. A positive theoretical and experimental contribution, regarding modelling and control of fuzzy and neuro fuzzy systems, is expected to be achieved.
Proposed non-linear identification solution will use for modelling and control, a recurrent neuro fuzzy architecture. Regarding model solution, a state space approach will be considered during fuzzy consequent local models design. Developed controller will be based on model parameters, being expected not only a stable closed loop solution, but also a static error with convergence towards zero. Model and controller fuzzy subspaces, will be partitioned throughout process dynamical universe, allowing fuzzy local models and controllers commutation and aggregation.
With the aim of capturing process under control dynamics using a real time approach, the use of recursive optimization techniques are to be adopted. Such methods will be applied during parameter and state estimation, using a dual decoupled Kalman filter extended with unscented transformation.
Two distinct processes one single-input (SISO) other multi-input (MIMO), will be used during experimentation. It is expected from experiments, a practical validation of proposed solution capabilities for control and identification. Presented work will not be completed, without first presenting a global analysis of adopted concepts and methods, describing new perspectives for future investigations.

Keywords: Recursive optimization, online identification, adaptative control, self learning, Kalman filtering, Unscented Transformation, recursive neuro fuzzy.

Resumo

O objectivo primordial desta tese tem por base o desenvolvimento de uma solução de controlo adaptativo fazendo uso de metodologias difusas. Consequentemente, pretende-se com esta dissertação dar um contributo positivo para modelos e controladores descritos segundo métodos difusos e neuro difusos.
A solução de controlo passa pela criação de um modelo não linear, baseado numa arquitetura neuro-difusa recorrente com modelos locais descritos sobre a forma de espaço de estados. É também desenvolvido um controlador neuro difuso recorrente, baseado nos parâmetros do modelo, que permita uma solução estável em anel fechado com um erro estático convergente para zero. Tem-se como objectivo a criação de um modelo distribuído pelo universo de funcionamento do processo, permitindo a comutação e agregação de diversos modelos e controladores locais.
Procurando capturar as dinâmicas do processo sobre controlo, segundo uma abordagem em tempo real, é necessária a utilização de técnicas de optimização, com capacidades de recursividade. Para o efeito, recorre-se neste trabalho, a técnicas de estimação de parâmetros e de estados baseadas no filtro de Kalman dual e desacoplado, estendido com a técnica de transformação de incerteza.
É efetuada uma análise experimental utilizando dois processos distintos, visando a conciliação dos conceitos teóricos apresentados na solução proposta. Na experimentação serão demostradas as capacidades de identificação e controlo para sistemas não só de uma entrada e uma saída (SISO), como também para sistemas de múltipla entrada múltipla saída (MIMO). Irá ser realizada uma análise global da solução, incluindo novos pontos de vista para caminhos de futuras investigações.

Palavras-chave: Estimação recursiva, identificação em linha, controlo adaptativo, auto aprendizagem, filtragem Kalman, transformação de incerteza, neuronal difuso.

Contents

List of Figures

List of Tables

Acronym List

ANFIS - Adaptative-Network-Based Fuzzy Inference System
ARX - Autoregressive with Exogenous inputs
COA - Center of Area Defuzzification
EKF - Extended Kalman Filter
FIS - Fuzzy Inference System
FS - Fuzzy System
FSS - Fuzzy State-Space
GA - Genetic Algoritm
ICUKF - Interval Constrained Unscented Kalman Filter
ICUT - Interval constrained Unscented Transformation
IUKF - Interval UKF
KF - Kalman Filter
MBC - Model Based Controller
MF - Membership Function
MSE - Mean Square Error
MRAFC - Model Reference Adaptative Fuzzy Control
NFM - neuro fuzzy modeling
NFS - Neuro Fuzzy System
pdf - Probability Density Function
PSO - Particle Swarm Optimization Algorithm
PUKF - Projected UKF
RNF - Recursive Neuro Fuzzy
RNFMS - Recursive Neuro Fuzzy Modeling System
RNFCS - Recursive Neuro Fuzzy Control System
RNFS - Recurrent Neuronal Fuzzy System
RNN - Recurrent Neuronal Network
SIUKF - Sigma-point Interval UKF
SFST - Standard Fuzzy Sets Theory
SS - State-Space

SSNFS - State Space Neuro Fuzzy System
SUT- Scaled Unscented Transformation
TIUKF - Truncated Interval UKF
TSK - Takagi-Sugeno and Kang
TUKF - Truncated UKF
UD - Universe of Discourse
UKF - Unscented Kalman Filter
UPUKF - Unconstrained Projected UKF
URNDDR - Unscented Recursive Nonlinear Dynamic data Reconciliation
UT- Unscented Transformation
WAM - Weighted Average Method Defuzzification
R_k^r - Model noise covariance at instance k
R_k^e - Output noise covariance at instance k
P_k - Predicted error covariance

1

Introduction

1.1 Global motivations

In a time where systems are becoming extremely complex, and computers processing capabilities faster and cheaper, new computer algorithms demanding high processing capabilities, such as genetic algorithms and recurrent neuro networks, are becoming widely used for real-time process identification and control. Since electronics are not a precise science, components characteristics might drift during their life cycle. Process dynamics consequently, might also change for an uncertain and time dependent amount. Also processes behaviour might be correlated with unexpected external variables (for instance environment conditions), which can considerably affect process working conditions. Such effects can drive closed loop solution to divergence. To overcome this fact, most controllers implement an adaptative strategy, allowing an adaptation for process behaviour changes. Several control theories capable of real time adaptation exist, among them it can be highlighted controllers which are based on model dynamics, also known as model based controllers (MBC). For these methods, model parameters will be used during evaluation of control actions. Process modelling and identification techniques, will induce a need for real time model and controller adaptation, being possible to cover not only changes in process dynamics, but also allowing a convergence to a more accurate system identification. Most identification algorithms follow an approach of finding a global model that best fits into real process dynamics. Since during real-time identification old data needs to be rejected in order to allow new data acquisition, the dynamical range of a global model approach, might not fit process dynamics over its complete dynamical range. Is towards this fact, where it can be found fuzzy modelling approaches for model and controller design. It allows the concept of a global model being defined with local

optimum solutions.

1.2 Goals and contributions

Having in scope a model and controller design defined by several local models and controllers, two major theories have been selected i.e fuzzy modelling and neural networks. Combining both methods into a single concept, will result in a new theory known in literature by neuro fuzzy modelling (NFM). This new method for system modelling, takes the advantage of fuzzy by allowing a continuous and stable model switching during real-time evaluation. By making use of neuro networks, NFM diminishes fuzzy inference process abstractness. It also allows an augmentation of fuzzy model parameters, due to creation of weights between neuron connections. Several neuro fuzzy systems (NFS) architectures were already developed and analysed. Although, most of them use an ARX structure for model design (case of ANFIS structure), being difficult to directly implement a MBC controller using system states. Some proposals for NFS using local models structured in a state space approach exist in literature, but none featured with recursiveness. Due to adaptation requirements, learning algorithm should be able to be computed during real-time processing, fulfilling process sampling times. Among several algorithms which fits into this class, a Kalman filtering (KF) technique was adopted due to its recursive nature. An extension to standard KF method known as Unscented Transformation (UT) was considered, allowing to compute model mean and covariance in a statistical fashion. The extension of KF with UT, is known in literature by Unscented Kalman Filtering (UKF), where instead of finding a model that best fits process output dynamics, it will fit model to process output distribution. Proposed global solution for system identification and control, is described in Figure 1.1.

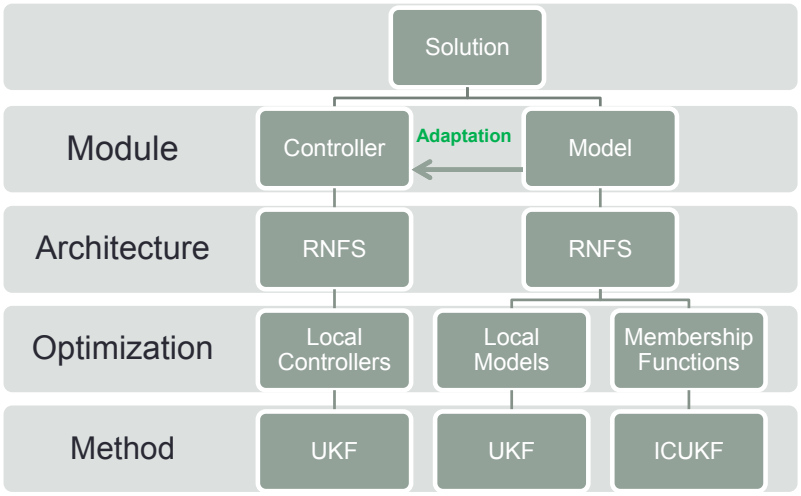


Figure 1.1: Global solution block diagram

Achieved solution is described in previous figure using a top down hierarchy. As it can be

noticed two modules were defined, a controller and a model. For both, a new Recursive Neuro Fuzzy architecture (RNF), using state space local models and controllers, was developed. Both modules will be under a real time recursive optimization, handled by UKF algorithm. Model will not only enforce optimization in rules local models, both for states and parameters, but also in membership functions (MF). Controller in opposition, will not handle MF optimization because it will use model premises. A constrained problem must be considered during MF optimization, which was addressed by other UKF extension known as Interval Constrained Unscented Kalman Filter (IUKF). Optimization of Rules aggregation and input weights is also seen as a constrained problem, although experimentation will not handle such optimization. New scientific contributions can be found in settled goals, precisely a new RNFS architecture which can be used both for controller and model design. Also, a new integration and applicability of UKF and IUKF theory was achieved, by applying it to a RNFS model and controller optimization.

(1) **Identification of non-linear systems**

The new NFS topology (as so far observed) proposed in this work, is featured with recurrence capabilities and contains eight layers handling fuzzification, fuzzy inference and defuzzification. The number of neurons for each layer depends on input and output variables, number of states and membership functions. The number of MFs depends on adopted fuzzy partitions, which should reflect process working zones. Increasing fuzzy partitions, might also increase model accurateness if enough input data is used to stimulate process throughout its working points. Although, it is also expected an augmentation of model and controller complexity, since new rules need to be considered. Complete process identification will be handled in a closed loop control solution as described in Figure 1.2.

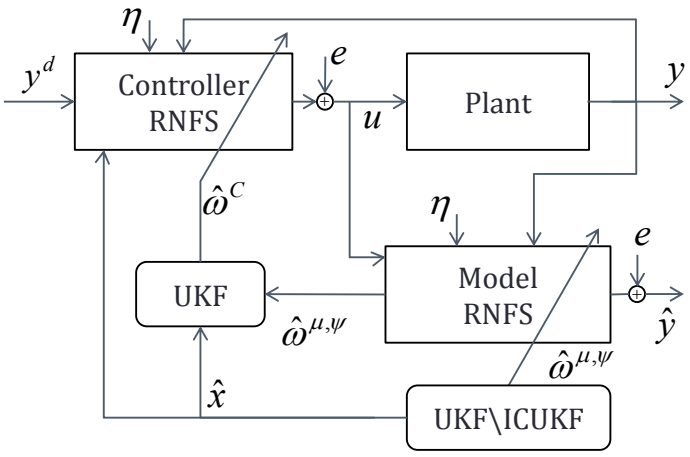


Figure 1.2: Closed loop identification block diagram

Concerning figure above, it was considered an additive noise both for model and controller. The latest, should smoothly drive process output to a desired reference value. Concerning model, it has for goal an increase of process fitness degree and

consequently, an improvement of prediction accuracy. An analytical study regarding stability analysis will not be presented, its is known that local models might suffer from zero pole cancellation and become unstable and uncontrollable. For a better understanding of Figure 1.2, it is worth a brief description regarding involved variables, which will be introduced in section 4.4. It should be considered:

- y^d is the desired process output values;
- $\hat{x}$ stands for model predicted states;
- $\hat{\omega}^C$ is the RNFS local controllers predicted parameters;
- $\hat{\omega}^{\mu;\psi}$ is the prediction of RNFS local models membership functions and consequents respectively;
- y is the process output;
- $\hat{y}$ is the model predicted output;
- e and η are additive Gaussian noise for parameters and outputs respectivelly.

(2) **Recurrent algorithm for training neuro fuzzy systems**
Regarding the use of UKF algorithm for RNFS identification, a sequential approach as defined in Figure 1.3 was considered.

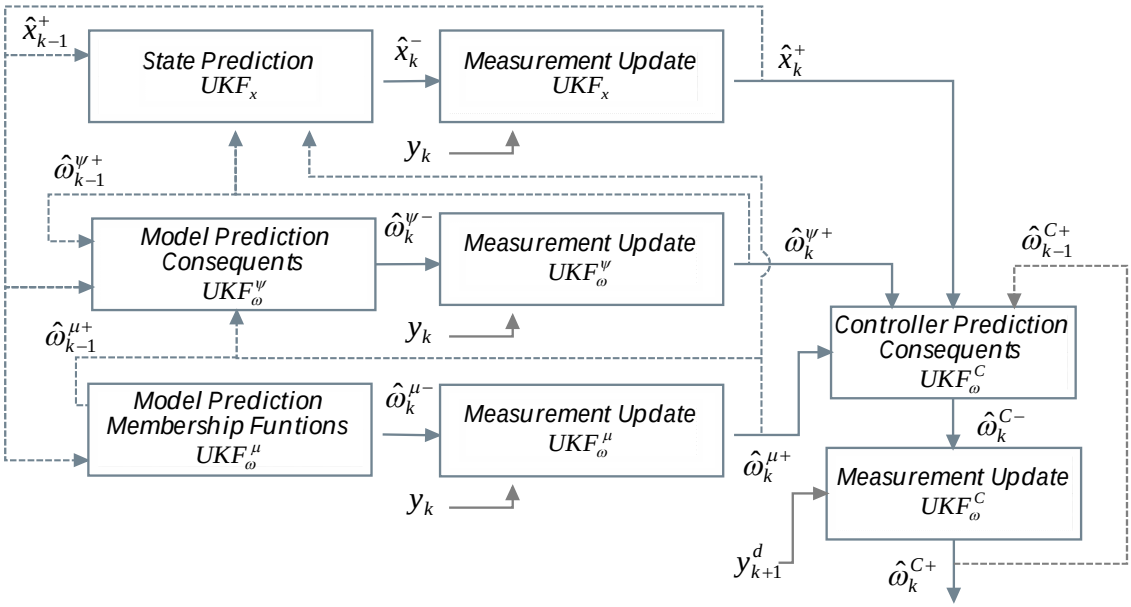


Figure 1.3: Optimization algorithm block diagram

A decoupled formulation of UKF regarding system parameters and states was adopted. Firstly, an optimization of predicted model states will be handled, followed by an optimization of predicted model parameters. Based on optimized model parameters and states, a controller parameter optimization will then be enforced, in order

to produce the next most accurate control action. It is worth mention that controller innovations, depend on error between output and reference. Regarding model, its innovations will use as cost factor the error between predicted and process outputs. Due to KF nature, it is expected for both errors a convergence to zero. Figure 1.3 variables will be introduced in chapter 4 meanwhile, a brief description for a better figure understanding should be handled. It is defined:

- $\hat{x}_{k-1}^{+}$ is the previous UKF prediction of model state;
- $\hat{x}_{k}^{-}$ is the current UKF prediction of model state after algorithm time update stage;
- $\hat{x}_{k}^{+}$ is the current UKF prediction of model state;
- y_k current process output;
- y_{k+1}^{d} is the next process output desired values;
- $\hat{\omega}_{k-1}^{\psi+}$ is the previous UKF prediction for RNFS model consequents parameters;
- $\hat{\omega}_{k}^{\psi-}$ is the current UKF prediction for RNFS model consequents parameters after algorithm time update stage;
- $\hat{\omega}_{k}^{\psi+}$ is the current UKF prediction for RNFS model consequents parameters;
- $\hat{\omega}_{k-1}^{\mu+}$ is the previous UKF prediction for RNFS model membership functions;
- $\hat{\omega}_{k}^{\mu-}$ is the current UKF prediction for RNFS model membership functions after algorithm time update stage;
- $\hat{\omega}_{k}^{\mu+}$ is the current UKF prediction for RNFS model membership functions;
- $\hat{\omega}_{k-1}^{C+}$ is the previous UKF prediction for RNFS controller parameters;
- $\hat{\omega}_{k}^{C-}$ is the current UKF prediction for RNFS controller parameters after algorithm time update stage;
- $\hat{\omega}_{k}^{C+}$ is the current UKF prediction for RNFS controller parameters;
- UKF_x UKF algorithm for RNFS model state optimization;
- UKF_{ω}^{ψ} UKF algorithm for RNFS model consequents parameter optimization;
- UKF_{ω}^{μ} UKF algorithm for RNFS model membership functions optimization (using ICUKF extension);
- UKF_{ω}^{C} UKF algorithm for RNFS controller parameter optimization

1.3 Thesis structure

Presented thesis contains six chapters including presentation chapter. Each chapter starts firstly with fundamental concepts, and a state of the art regarding what have been done and could be adopted. In chapter three a new architecture regarding model and controller design will be proposed. Chapter four will also describe the applicability of UKF in proposed RNFS architecture. A brief conclusion is included at the end of each chapter,

describing what remains to be done, future enhancements and other capabilities and applications beyond used ones.

Chapter 2: Fuzzy Logic Systems: Theory and Concepts

This chapter introduces a brief overview regarding fuzzy notions. It starts with a description about fuzzy sets and their operations, finishing with a description of fuzzy inference mechanisms.

Chapter 3: Fuzzy System Modelling

Focusing on model design phase, several modelling methodologies will be presented. Chapter starts with an overview regarding ARX, NARX and state space model structuring. Then it continues with a description of neuro networks concepts and their evolution to neuro fuzzy networks. Chapter ends with a new network structuring proposal, which can be seen as an upgrade over existing state of the art methods. Also a complete closed loop network solution as in Figure 1.2 regarding process identification and control is presented.

Chapter 4: Estimation Methods for Fuzzy Structures Parameters

This chapter provides the next step towards final system identification solution. It starts with an introduction of Kalman statistical properties and the principle of unscented transformation. Also the pervasive computation of UKF regarding state and parameter optimization is demonstrated. Searching for a better comprehension regarding UKF and RNFS integration, chapter includes an analytical description of RNFS architecture parameters, comprising variable design and dimensioning both for controller and model. Having defined RNFS variables, chapter describes the computation of ICUKF and UKF for RNFS optimization. For a better understanding of UKF intrinsic variables and dynamics, a theoretical model well known in literature, will serve as use case for a better understanding regarding algorithm convergence and stability.

Chapter 5: Implementation

Chapter five handles experimentation phase, it analyses proposed theoretical architectures and respective optimization processes. Theory presented so far, will be used to modelling and control both a SISO and MIMO plants. Model fitting ratio will be validated by making use of mean square error analysis. Controller response, will also be analysed through static error, raising time and overshot ratio. Chapter also includes an analysis regarding the effects of MF optimization i.e the advantages and disadvantages when it is considered during the optimization loop.

Chapter 6: Global Conclusions and Further Research

Presented thesis will not end without a global overview regarding proposed solution and achieved experimental results. Chapter will also include an analysis regarding further enhancements, new perspectives and vectors which could have been adopted as possible solution. The thesis will finish with an overview of what was achieved, presenting possible future investigations based on proposed solution.

2

Fuzzy Logic Systems: Theory and Concepts.

2.1 Introduction

This chapter aims to introduce fundamental notions regarding fuzzy reasoning. Presented theory will allow a complete understanding of proposed solution. Firstly, it will handle basic notions regarding fuzzy sets and their evolution from crisp sets. Having introduced fuzzy sets, it will continue with an analytical description of standard operations on fuzzy sets including their shapes known as MFs. Before starting with fuzzy inference process, it is worth defining fuzzy relations, since it contains key aspects for understanding of fuzzification, implication and defuzzification processes.

2.2 Fuzzy sets theory.

It was Zadeh [1] whom in 1965 introduced fuzzy logic, namely fuzzy sets. This theory allowed the introduction of a multi-valour logic against a boolean logic. Fuzzy logic is reflected in human methods for system analysis, which exact characteristics or properties are unknown and can differ depending on each person interpretation. This way, it is possible to handle information containing a high degree of uncertainty and accuracy, which was not possible using a "yes or no", "true or false" methodology.

2.2.1 Classic sets.

For more information regarding classic sets consider [1], since presented work will only demonstrate key notions, which will be used as starting point for the introduction of fuzzy sets. Classic or crisp sets are all sets whose elements have only two states i.e, it can either full belong or not belong to a set. Take as example the boolean logic where a variable can only be zero or one. Let **U** be a not empty set also known as *universe of discourse* (UD), it contains all possible elements or members of a given context, each of those elements are contained in **U**. The union of several elements of **U** is known as a subset U of **U**.

Definitions:

- $u \in U$ - a member u of **U** belongs to the subset U of **U**;
- $u \notin U$ - u is not member of U;
- $u \subseteq \mathbf{U}$ - U is subset of **U** or U iqual **U**;
- $\emptyset$ - empty set;

Sets with distinct properties $P_1...P_n$ for all UD are assigned with a linguistic term for instance A where:

$$A = \{a | a \text{ with properties } P_1...P_n\}$$

Each set properties can be described by their characteristic function, defined as::

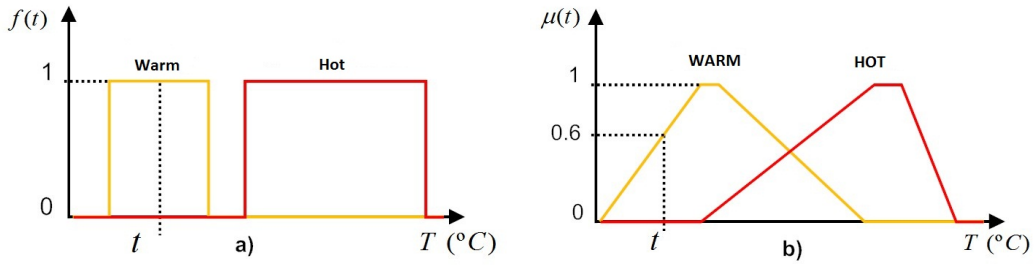
$$\mathbf{f}_A = \begin{cases} 1, & \text{if } u \in A \\ 0, & \text{if } u \notin A \end{cases} \tag{2.1}$$

In other words, characteristic function performs the mapping of elements contained in **U** to elements in domain $\{0, 1\}$ by using $\mathbf{f}_A : \mathbf{U} \rightarrow \{0, 1\}$. If $\mathbf{f}_A(u) = 1,\ u \in A$ otherwise, if $\mathbf{f}_A(u) = 0,\ u \notin A$. The characteristic function of a subset of **U**, where subset U is the aggregation of all n sets $\in$ **U**, is defined as $f(u)$, and is the projection of $f_{A_1}(u), f_{A_2}(u), ..., f_{A_n}(u)$, with $n \in \Re$ along $y - axis\ \forall u \in \mathbf{U}$.

Figure 2.1(a) displays the characteristic function of two crisp sets, with linguistic words "Warm" and "Hot", where **U** covers the temperature (T) value range of a room $T \in \Re$. In many cases $\mathbf{U} \in \Re^n$ belongs to a range $\in [-1, 1]$, becoming in this case known as "normalized universe of discourse".

2.2.2 Fuzzy sets.

Since first publication by Zadeh forty five years ago regarding fuzzy sets, a large variety of publications concerning new fuzzy set theories emerged [2] although, this section will focus on initial theory concepts provided by [1]. It includes basic concepts, the most commonly used fuzzy sets, and a theory analysis regarding fuzzy sets intersection and



(a) Crisp sets (b) Fuzzy sets

Figure 2.1: Illustration of crisp and fuzzy sets

union. For a profound theoretical analysis it is highlighted [3].

As it can be seen through 2.1, the main disadvantage of a crisp against a fuzzy set, dues to a constrained linguistic variable $u \in \Re$ characteristic function, supporting only two values. Concerning given example, the room temperatures will vary abruptly from a full "Warm" to a full "Hot" state. It was towards this direction that fuzzy sets evolved, allowing a characteristic function with a smooth transition between sets as it can be seen by 2.1(b). Fuzzy sets can be seen as an extension of crisp sets, where instead of mapping a fuzzy variable into set of values $\{0, 1\}$, it maps to a value contained in interval $[0, 1]$. The characteristic function of a fuzzy set can be defined as:

$$\begin{gathered} f_A : \mathbf{U} \rightarrow [0,1] \in \Re \\ or \\ \mu_A : \mathbf{U} \rightarrow [0,1] \in \Re \end{gathered} \tag{2.2}$$

where μ_A is the membership function of a fuzzy set A whose value represents the variable $u \in \mathbf{U}$ degree of membership on set A. For instance, through 2.1(b) it can be noticed that linguistic variable $t \in \mathbf{T}$ (where **T** is the UD), contains a degree of membership $\mu_{Warm}(t) = 0.6$ in set "Warm" and $\mu_{Hot}(t) = 0$ set "Hot". Next will be presented several definitions from the standard fuzzy sets theory (SFST):

Definition (Fuzzy Set) - A fuzzy set defined in space **U** is a set of pairs:

$$A = \{(u, \mu_A(u)), u \in \mathbf{U}\}, \ \forall x \in \mathbf{U}. \tag{2.3}$$

Definition (Empty) - A fuzzy set A is called empty if:

$$\forall\, u \in \mathbf{U}, \mu_A(u) = 0. \tag{2.4}$$

Definition (Equality) - Two fuzzy sets A and B are equal if:

$$\forall\, u \in \mathbf{U},\ \mu_A(u) = \mu_B(u) \text{ or simply } \mu_A = \mu_B. \tag{2.5}$$

Definition (Complement) - A fuzzy set A complement denoted by A' is defined as:

$$\mu_{\overline{A}} = 1 - \mu_A. \tag{2.6}$$

Definition (Containment) - A is contained in B, or A is a subset of B, if:

$$\mu_A \subset \mu_B \Longleftrightarrow A \leq B. \tag{2.7}$$

Definition (Union) - Union of two fuzzy sets A and B with respective membership functions μ_A and μ_B is a new fuzzy set C where $C = A \cup B$, whose membership function is related to μ_A and μ_B through:

$$\mu_C(u) = Max\left[\mu_A(u), \mu_B(u)\right] \ \forall\, u \in \mathbf{U} \tag{2.8}$$

or simply

$$\mu_C = \mu_A \vee \mu_B. \tag{2.9}$$

In other words, the union of A and B is the lowest fuzzy set containing A and B a s it can be seen by 2.2(b).

Definition (intersection) - Intersection of two fuzzy sets A and B with membership functions μ_A and μ_B respectively, is a new fuzzy set C such that $C = A \cap B$, where its membership function is related with μ_A and μ_B through:

$$\mu_C(u) = Min\left[\mu_A(u), \mu_B(u)\right] \ \forall\, u \in \mathbf{U} \tag{2.10}$$

or simply

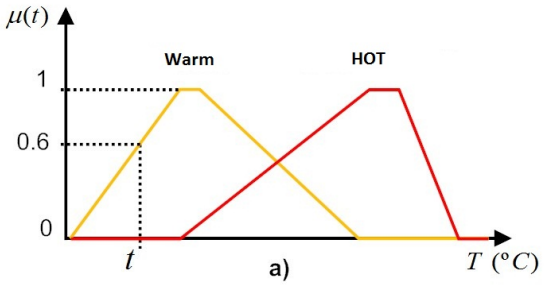
$$\mu_C = \mu A \vee \mu_B. \tag{2.11}$$

In other words, intersection of A and B is the highest fuzzy set which is contained in A and B, as it can be observed from 2.2(c). It is worth mention that for fuzzy sets, in opposition to crisp sets, is has no meaning saying that a given linguistic variable belongs to a set if $\mu_A(u) > 0$ with u $\in$ **U**. In fuzzy domain multiple intervals can be defined for instance, α and β with, $0 < \alpha < 1; 0 < \beta < 1$ and $\beta < \alpha$. Also, sentence "u belongs to A" might depend on several constraints, for instance if $\mu_A(u) > \alpha$ or $\beta < \mu_A(u) < \alpha$. Fuzzy sets are not constrained to a single value logic, they allow reasoning based on a multi value logic.

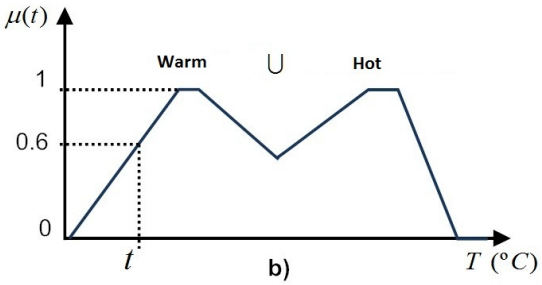
Definition (Height) - The height of a fuzzy set A is the maximum value of its membership function:

$$hgt(A) = \sup_{u \in \mathbf{U}} \mu_A(u). \tag{2.12}$$

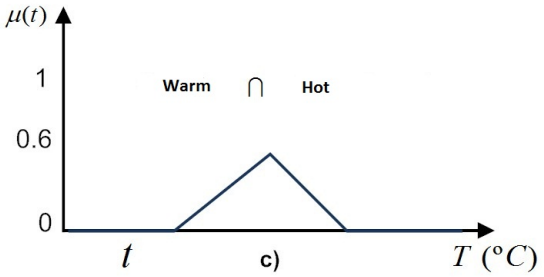
if $hgt(A) = 1$ fuzzy set A is called normal, otherwise subnormal.



(a) Fuzzy sets



(b) Union



(c) Intersection

Figure 2.2: Basic operations on fuzzy sets

Definition (Support) - Support of a fuzzy set A corresponds to all values of $u \in \mathbf{U}$ over which $\mu_A(u) > 0$, this means:

$$supp(A) = \{u \in \mathbf{U} | \mu_A(u) > 0\}. \tag{2.13}$$

Definition (Core) - Core of a fuzzy set A corresponds to all values of $u \in \mathbf{U}$ over which $\mu_A(u) = 1$ i.e:

$$core(A) = \{u \in \mathbf{U} | \mu_A(u) = 0\}. \tag{2.14}$$

A fuzzy set A is normal if its core is not empty. 2.3 demonstrates a use case of concepts "height","core","support" of a fuzzy set having linguistic word "Warm".

Definition (Convexity) - A fuzzy set A is convex if for any $u_1, u_2 \in \mathbf{U}^n$ given $\lambda \in [0, 1]$

$$\mu_A(\lambda u_1 + (1 - \lambda)u_2) \geq min\{\mu_A(u_1), \mu_A(u_1)\}. \tag{2.15}$$

as shown in 2.4.

Definition (Symmetry) - A fuzzy set is symmetric if its membership function is symmetric around a given point c

$$\mu_A(c + \Delta u) = \mu_A(c - \Delta u), \forall\, c - \Delta u \text{ e } c - \Delta u \ \in \mathbf{U}. \tag{2.16}$$

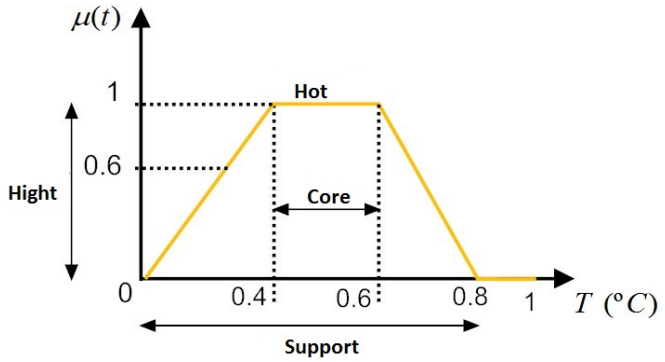


Figure 2.3: Fuzzy set with height= 1, support = $[0, 0.8]$ and core = $[0.4, 0.6]$

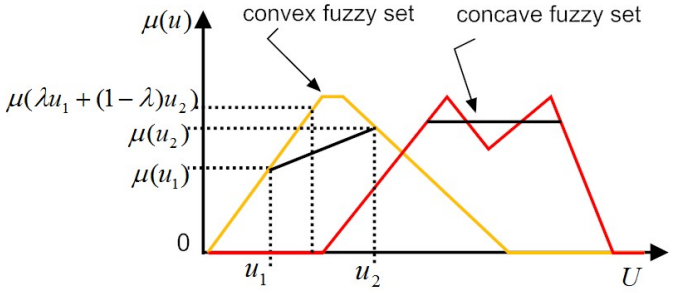


Figure 2.4: Convexity of a fuzzy set

2.2.3 Triangular norms and negation.

Several studies in the field of fuzzy set theories have been done [2] (pp. 30-63) although, section 2.2.2 only introduced classic theory from Zadeh. This section presents other theories as t-norm, t-conorm and negation, which mathematical nomenclature is based on [4](pag. 13-21).

Definition (t-norm) - The t-norm is a function T of two variables

$$T : [0, 1] \times [0, 1] \to [0, 1] \tag{2.17}$$

satisfying conditions:

1. T is monotonous

$$T\{\mu_A(u), \mu_C(u)\} \leq T\{\mu_B(u), \mu_D(u)\} \ \forall\, \mu_A(u) \leq \mu_B(u) \text{ e } \mu_C(u) \leq \mu_D(u) \tag{2.18}$$

2. T is commutative

$$T\{\mu_A(u), \mu_B(u)\} = T\{\mu_B(u), \mu_A(u)\}, \tag{2.19}$$

3. T is associative

$$T\left\{T\left\{\mu_A(u),\mu_B(u)\right\},\mu_C(u)\right\}=T\left\{\mu_A(u),T\left\{\mu_B(u),\mu_C(u)\right\}\right\}, \tag{2.20}$$

4. T satisfying boundary conditions

$$T\left\{\mu_A(u),0\right\}=0, T\left\{\mu_A(u),1\right\}=\mu_A(u), \tag{2.21}$$

with $\mu_A(u),\mu_B(u),\mu_C(u),\mu_D(u)\ \in[0,1]\ \wedge\ u\in\mathbf{U}$.

symbolically, t-norm in arguments $\mu_A(u),\mu_B(u)$ refers to the intersection of two fuzzy sets $A,B\in\ \mathbf{U}$ with membership functions μ_A e μ_B respectively,

$$T\left\{\mu_A(u),\mu_B(u)\right\}=\mu_A(u)\overset{T}{*}\mu_B(u). \tag{2.22}$$

$$\mu_{A\cap B}=\mu_A(u)\overset{T}{*}\mu_B(u) \tag{2.23}$$

Extending t-norm aggregation definition (2.20) to $n>2$ variables

$$\begin{gathered}T_{i=1}^{n}\left\{\mu_{A_i}\right\}=T\left\{T_{i=1}^{n-1}\left\{\mu_{A_i}(u)\right\},\mu_{A_n}(u)\right\}=T\left\{\mu_{A_1}(u),\mu_{A_2}(u),...,\mu_{A_n}(u)\right\}=\\=T\left\{\mu_{\mathbf{A}}(u)\right\}=\mu_{A_1}(u)*^{T}\mu_{A_2}(u)*^{T}...*^{T}\mu_{A_n}(u).\end{gathered} \tag{2.24}$$

Definition (t-conorm) - T-conorm is a function S of two variables

$$S:[0,1]\times[0,1]\to[0,1] \tag{2.25}$$

satisfying conditions:

1. S is monotonous

$$S\left\{\mu_A(u),\mu_C(u)\right\}\leq T\left\{\mu_B(u),\mu_D(u)\right\}\ \forall\,\mu_A(u)\leq\mu_B(u)\text{ e }\mu_C(u)\leq\mu_D(u), \tag{2.26}$$

2. S is commutative

$$S\left\{\mu_A(u),\mu_B(u)\right\}=S\left\{\mu_B(u),\mu_A(u)\right\}, \tag{2.27}$$

3. S is associative

$$S\left\{S\left\{\mu_A(u),\mu_B(u)\right\},\mu_C(u)\right\}=S\left\{\mu_A(u),S\left\{\mu_B(u),\mu_C(u)\right\}\right\}, \tag{2.28}$$

4. S satisfies boundary conditions

$$S\left\{\mu_A(u),0\right\}=0, S\left\{\mu_A(u),1\right\}=\mu_A(u), \tag{2.29}$$

with $\mu_A(u), \mu_B(u), \mu_C(u), \mu_D(u) \in [0,1] \ \wedge \ u \in \mathbf{U}$.

symbolically, t-conorm in arguments $\mu_A(u), \mu_B(u)$ is the union of two fuzzy sets $A, B \in \mathbf{U}$ with membership functions μ_A and μ_B respectively, and is defined as:

$$S\left\{\mu_A(u), \mu_B(u)\right\} = \mu_A(u) \overset{S}{*} \mu_B(u). \tag{2.30}$$

$$\mu_{A \cup B} = \mu_A(u) \overset{S}{*} \mu_B(u) \tag{2.31}$$

Extending t-conorm definition (2.28) to $n > 2$ variables

$$\begin{gathered} S_{i=1}^{n}\left\{\mu_{A_i}(u)\right\} = S\left\{S_{i=1}^{n-1}\left\{\mu_{A_i}(u)\right\}, \mu_{A_n}(u)\right\} = S\left\{\mu_{A_1}(u), \mu_{A_2}(u), ..., \mu_{A_n}(u)\right\} = \\ = S\left\{\mu_{\mathbf{A}}(u)\right\} = \mu_{A_1}(u) *^S \mu_{A_2}(u) *^S ... *^S \mu_{A_n}(u). \end{gathered} \tag{2.32}$$

Both t-norm and t-conorm have three general derivations:

- The family min/max already introduced in 2.2.2 defined as:

$$\begin{gathered} T_M\left\{\mu_{A_1}(u), \mu_{A_2}(u)\right\} = \min\left\{\mu_{A_1}(u), \mu_{A_2}(u)\right\} \\ S_M\left\{\mu_{A_1}(u), \mu_{A_2}(u)\right\} = \max\left\{\mu_{A_1}(u), \mu_{A_2}(u)\right\} \\ T_M\left\{\mu_{A_1}(u), \mu_{A_2}(u), ..., \mu_{A_2}(u)\right\} = \min_{i=1,...,n}\left\{\mu_{A_i}(u)\right\} \\ S_M\left\{\mu_{A_1}(u), \mu_{A_2}(u), ..., \mu_{A_2}(u)\right\} = \max_{i=1,...,n}\left\{\mu_{A_i}(u)\right\} \end{gathered} \tag{2.33}$$

- The family of algebraic triangular norms defined as:

$$\begin{gathered} T_P\left\{\mu_{A_1}, \mu_{A_2}\right\} = \mu_{A_1}\mu_{A_2} \\ S_P\left\{\mu_{A_1}, \mu_{A_2}\right\} = \mu_{A_1} + \mu_{A_2} - \mu_{A_1}\mu_{A_2} \\ T_P\left\{\mu_{A_1}, \mu_{A_2}, ..., \mu_{A_n}\right\} = \prod_{i=1}^{n} \mu_{A_i} \\ S_P\left\{\mu_{A_1}, \mu_{A_2}, ..., \mu_{A_n}\right\} = 1 - \prod_{i=1}^{n}(1 - \mu_{A_i}) \end{gathered} \tag{2.34}$$

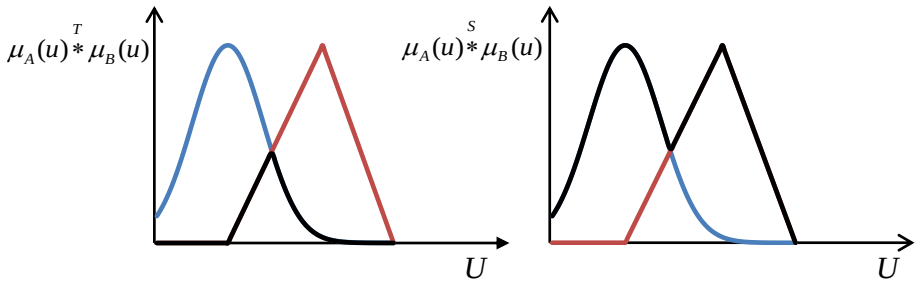
  T_P also known as t-norm product, such as S_P also known as t-conorm probabilistic sum.

- Lukasiewicz triangular norm family are defined as:

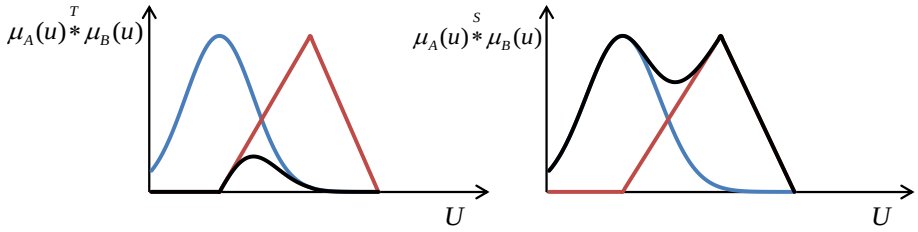
$$\begin{gathered} T_L\left\{\mu_{A_1}, \mu_{A_2}\right\} = \max\left\{\mu_{A_1} + \mu_{A_2} - 1, 0\right\} \\ S_L\left\{\mu_{A_1}, \mu_{A_2}\right\} = \min\left\{\mu_{A_1} + \mu_{A_2 1}, 1\right\} \\ T_L\left\{\mu_{A_1}, \mu_{A_2}, ..., \mu_{A_n}\right\} = \max\left\{\sum_{i=1}^{n} \mu_{A_i} - (n-1), 0\right\} \\ S_L\left\{\mu_{A_1}, \mu_{A_2}, ..., \mu_{A_n}\right\} = \min\left\{\sum_{i=1}^{n} \mu_{A_i}, \right\} \end{gathered} \tag{2.35}$$

Definition (negation)

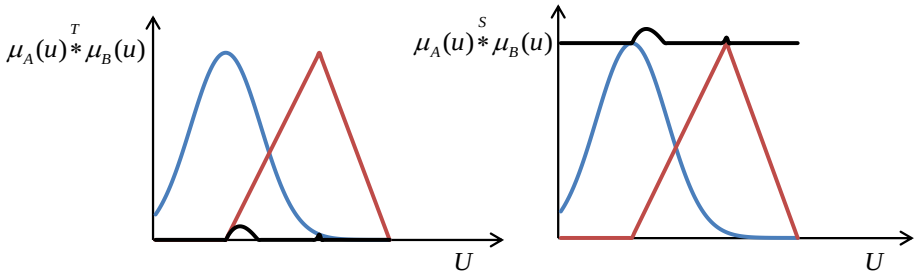
1. A non increasing function $N : [0,1] \to [0,1]$ is called negation if $N(0) = 1$ and $N(1) = 0$.



(a) Intersection and union operators using min/max method



(b) Intersection and union operators using algebric method



(c) Intersection and union operators using Lukasiewicz method

Figure 2.5: Intersection and union operators

2. A negation $N : [0,1] \rightarrow [0,1]$ is called strict negation N is continuous and strictly decreasing.

3. A strict negation $N : [0,1] \rightarrow [0,1]$ is called strong negation if it is an involution, in other words, if $N(N(\mu_A(u))) = \mu_A(u)$

Simbolically a negation or complement of two fuzzy sets $A, B \in$ **U** with respective membership functions μ_A and μ_B can be written as

$$\mu_{\overline{A}}(u) = N(\mu_A(u)) \tag{2.36}$$

where $N(\mu_A(u))$ can be any type of negation. In general, three types of negation can be defined:

- Zadeh's negation defined as

$$N(\mu_A(u)) = 1 - \mu_A(u) \tag{2.37}$$

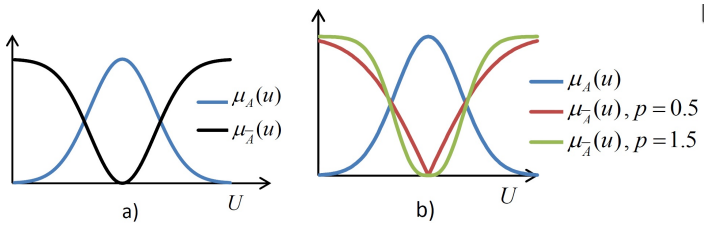
- Yager's negation defined as

$$N(\mu_A(u)) = (1 - \mu_A(u)^p)^{\frac{1}{p}},\ p > 0 \tag{2.38}$$

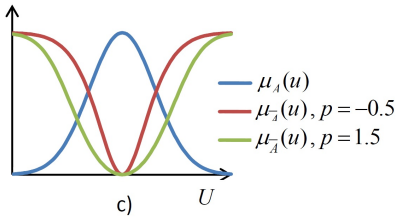
- Sugeno's negation defined as

$$N(\mu_A(u)) = \frac{1 - \mu_A(u)}{1 + p\mu_A(u)},\ p > -1 \tag{2.39}$$

2.6 illustrates an example of three negations types, and 2.5(a), 2.5(b), 2.5(c) demonstrates three types of intersection and union.



(a) Zadeh's negation (b) Yager's negation



(c) Sugeno's negation

Figure 2.6: Complement operators.

2.2.4 Fuzzy Relations

This part describes the basic concepts of fuzzy relations used in fuzzy reasoning, which will be addressed during next section. The Cartesian product [4] is defined as:

Definition (Cartesian product) - Having two fuzzy sets $A \subseteq \boldsymbol{U}$ and $B \subseteq \boldsymbol{Y}$, then the Cartesian product between A and B is denoted by $A \times B$ and defined as:

$$\begin{gathered} \mu_{A\times B}(u, y) = min\left\{\mu_A(u), \mu_B(u)\right\} \\ \text{or} \\ \mu_{A\times B}(u, y) = \mu_A(u).\mu_B(u) \end{gathered} \tag{2.40}$$

where $u \in \boldsymbol{U}$ and $y \in \boldsymbol{Y}$ are linguistic variables. For n fuzzy sets $A_1 \subseteq \boldsymbol{U_1}$, $A_2 \subseteq \boldsymbol{U_2}, \ldots, A_n \subseteq \boldsymbol{U_n}$ the Cartesian product is denoted by $A_1 \times A_2 \times \ldots \times A_n$ and defined

as:

$$\begin{gathered}\mu_{A_1\times A_2\times\ldots\times A_n}(u_1,u_2,\ldots,u_n)=\min_{i=1\ldots n}\{\mu_{A_i}(u_i)\}\\ \text{or}\\ \mu_{A_1\times A_2\times\ldots\times A_n}(u_1,u_2,\ldots,u_n)=\prod_{i=1}^{n}\mu_{A_i}(u_i)\end{gathered}\tag{2.41}$$

In a generic concept, the Cartesian product is described by a t-norm not being restricted to the $\min$ operator. A Fuzzy relation $\mu_R(u,y)$ is a mapping from the Cartesian space $U\times Y$ to the interval $[0\ 1]$ [5] such that:

$$\begin{aligned}A\times B=R\ &\subset\ \boldsymbol{U}\times\boldsymbol{Y}\\ &=\sum_{\boldsymbol{U}\times\boldsymbol{Y}}\frac{\mu_R(\boldsymbol{U}\times\boldsymbol{Y})}{(\boldsymbol{U}\times\boldsymbol{Y})}\end{aligned}\tag{2.42}$$

with membership function

$$\begin{aligned}\mu_R(u,y)&=\mu_{A\times B}(u,y)\\ &=\min\mu_A(u),\mu_B(y)\end{aligned}\tag{2.43}$$

The Cartesian product defined in (2.42) is implemented through the cross product of two vectors, for instance consider fuzzy set

$$A=\frac{0.2}{u_1}+\frac{0.6}{u_2}+\frac{1}{u_3}$$

and fuzzy set

$$B=\frac{0.3}{y_1}+\frac{0.9}{y_2}$$

μ_R is computed through:

$$\mu_R=\frac{min(0.2,0.3)}{u_1,y_1}+\frac{min(0.2,0.9)}{u_2,y_2}+\ldots+\frac{min(1,0.9)}{u_3,y_2}$$

relation R according to (2.42) can be expressed in matrix form:

$$A\times B=R=\begin{array}{c}\\ u_1\\ u_2\\ u_3\end{array}\begin{array}{c}\begin{array}{cc}y_1 & y_2\end{array}\\ \begin{bmatrix}0.2 & 0.2\\ 0.3 & 0.6\\ 0.3 & 0.9\end{bmatrix}\end{array}$$

Definition (Fuzzy composition) - Consider fuzzy relations R and S defined in $\boldsymbol{U}\times\boldsymbol{Y}$ and $\boldsymbol{Y}\times\boldsymbol{Z}$ respectivelly. The sup-T composition of R and S is a fuzzy relation denoted

by $R \circ S \subseteq \boldsymbol{U} \times \boldsymbol{Z}$ defined as:

$$\mu_{R \circ S}(u, z) = \sup_{y \in \boldsymbol{Y}} \left\{ \mu_R(u, y) *^T \mu_S(y, z) \right\} \tag{2.44}$$

Fuzzy composition can also be applied to obtain a fuzzy set $B \subseteq \boldsymbol{Y}$ from the composition of fuzzy set $A \subseteq \boldsymbol{U}$ and fuzzy relation $R \subseteq \boldsymbol{U} \times \boldsymbol{Y}$ this means,

$$\begin{aligned} B &= A \circ R \subseteq \boldsymbol{Y} \\ &\text{with membership function} \\ \mu_B(y) &= \mu_{A \circ R}(y) \\ &= \sup_{u \in \boldsymbol{U}} \mu_A(u) *^T \mu_R(u, y) \end{aligned} \tag{2.45}$$

from above deductions $*^T$ is a T-norm previously defined.

2.2.5 Membership functions

This section will introduce most common types of membership functions (MFs), which analytical definition can be found in [6]. As it was already mentioned in section 2.2.2 by 2.2, a MF of set A respectively μ_A, does a mapping of a fuzzy variable $u \in$ **U** through fuzzy set $A \in$ **U**, where **U** is the UD defining the variable membership degree.

1. Gaussian MF "MF-G" (figure 2.7**b)**), has three parameters σ and $c \in \Re$ such that

$$f(u; \sigma, c) = e^{\frac{-(u-c)^2}{2\sigma^2}} \tag{2.46}$$

2. Generalized bell MF "MF-SG" (figure 2.7**d)**), has three parameters a, b, $c \in \Re$ such that

$$f(u; a, b, c) = \frac{1}{1 + |\frac{u-c}{a}|^{2b}} \tag{2.47}$$

3. S-shaped MF "MF-S" (much similar with figure 2.7**c)**), contains two parameters a and $b \in \Re$

$$f(u; a, b) = \begin{cases} 0, & u \leq a \\ 2\left(\frac{u-a}{b-a}\right)^2, & a \leq u \leq \frac{a+b}{2} \\ 1 - 2\left(\frac{u-a}{b-a}\right)^2, & \frac{a+b}{2} \leq u \leq b \\ 1, & u \geq b \end{cases} \tag{2.48}$$

4. Sigmoidal shape MF "MF-SIG" (figure 2.7**c)**), contains two parameters a and $b \in \Re$ such that

$$f(u; a, b) = \frac{1}{1 + e^{-a(u-c)}} \tag{2.49}$$

The sign of parameter a changes the function opening, if right or left opened. Based

on this MF-SIG, two other functions can be defined through the product of two sigmoidal functions MF-SIG $\times$ MF-SIG = MF-PSIG (figure 2.7**f)**) or by their difference MF-SIG $-$ MF-SIG = MF-DSIG (figure 2.7**a)**)

5. Z-shaped MF "FP-Z" (figure 2.7**g)**), contains two parameters a and $b \in \Re$

$$f(u;a,b) = \begin{cases} 1, & u \leq a \\ 1 - 2\left(\frac{u-a}{b-a}\right)^2, & a \leq u \leq \frac{a+b}{2} \\ 2\left(b - \frac{u}{b-a}\right)^2, & \frac{a+b}{2} \leq u \leq b \\ 0, & u \geq b \end{cases} \tag{2.50}$$

6. Triangular shaped MF "MF-TRI" (figure 2.7**h)**), contains three parameters a, b and $c \in \Re$

$$f(u;a,b,c) = \begin{cases} 0, & u \leq a \\ \frac{u-a}{b-a}, & a \leq u \leq b \\ \frac{c-u}{c-b}, & b \leq u \leq c \\ 0, & c \leq u \end{cases} \tag{2.51}$$

or in a compact form

$$f(u;a,b,c) = \max\left(\min\left(\frac{u-a}{b-a}, \frac{c-u}{c-b}\right), 0\right) \tag{2.52}$$

7. Trapezoidal shape MF "MF-TRAP" (figure 2.7**i)**), contains four parameters a, b, c and $d \in \Re$

$$f(u;a,b,c,d) = \begin{cases} 0, & u \leq a \\ \frac{u-a}{b-a}, & a \leq u \leq b \\ 1, & b \leq u \leq c \\ \frac{d-u}{d-c}, & c \leq u \leq d \\ 0, & d \leq u \end{cases} \tag{2.53}$$

8. PI(π) shaped MF "MF-PI" (figure 2.7**e)**), contains four parameters a, b, c and $d \in \Re$. This MF is built from the product between MF-S $\times$ MF-Z = MF-PI:

$$f(u;a,b,c,d) = \begin{cases} 0, & u \leq a \\ 2\left(\frac{u-a}{b-a}\right)^2, & a \leq u \leq \frac{a+b}{2} \\ 1 - 2\left(\frac{u-b}{b-a}\right)^2, & \frac{a+b}{2} \leq u \leq b \\ 1, & b \leq u \leq c \\ 1 - 2\left(\frac{u-c}{d-c}\right)^2, & c \leq u \leq \frac{c+d}{2} \\ 2\left(\frac{u-d}{d-c}\right)^2, & \frac{c+d}{2} \leq u \leq d \\ 0, & u \geq d \end{cases} \tag{2.54}$$

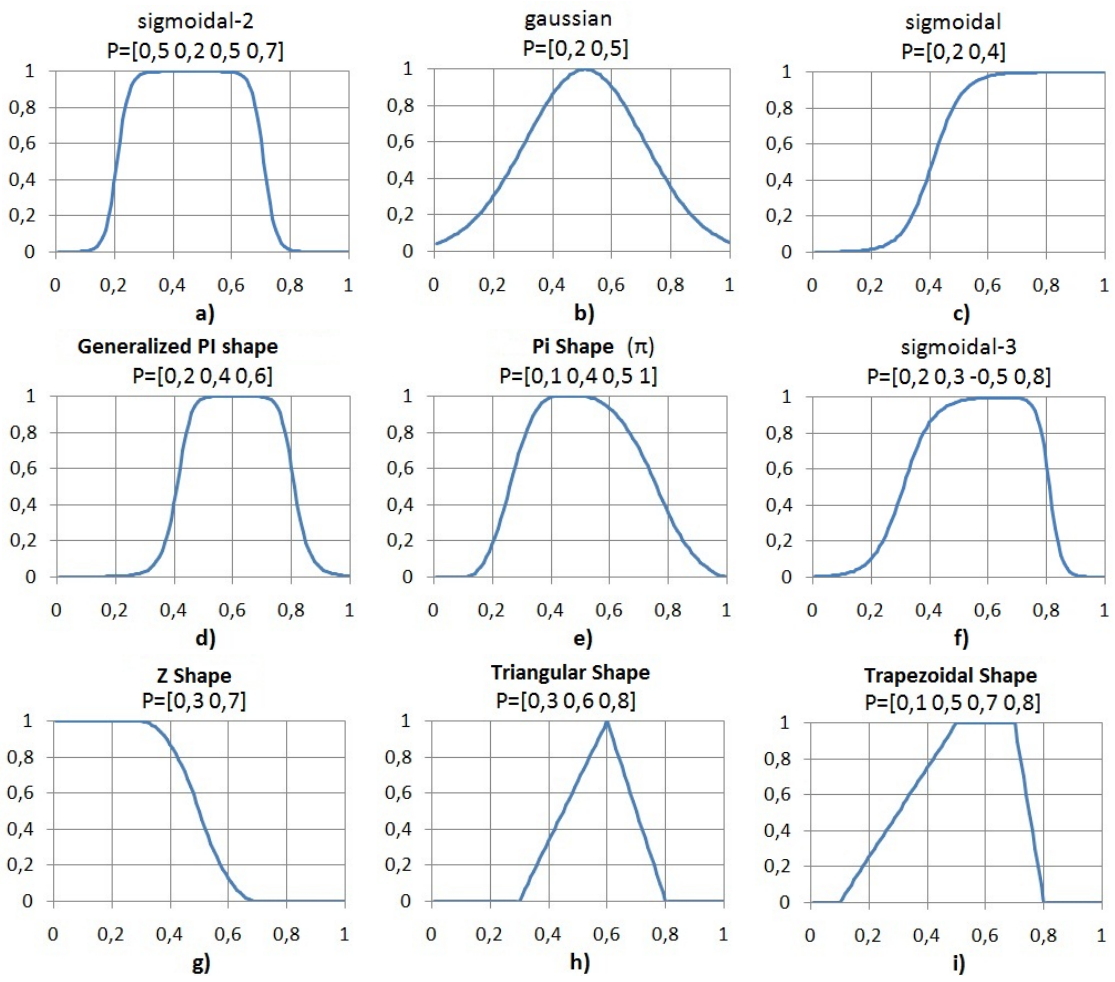


Figure 2.7: Different shapes of MFs **a)** MF-DSIG; **b)** MF-G; **c)** MF-SIG; **d)** MF-SG; **e)** MF-PI; **f)** MF-PSIG; **g)** MF-Z; **h)** MF-TRI; **i)** MF-TRAP;

2.3 Fuzzy Inference

Any fuzzy system can be comprised by four main components Figure 2.8, a Fuzzifier, Defuzzifier, Inference Engine and a rule base, independently from the used inference type. Every fuzzy system should have a finite universe of discourse **U** in order to be practically conceivable. A finite UD is achieved through a normalizer and a denormalizer layers. The normalizer projects any input linguistic variable u onto **U** by a scaling factor $\hat{u} = \frac{u}{k} \;\; k \in \Re$. In opposition, the denormalizer undoes the initial projection by the same scaling factor $u = \hat{u} \times k \; k \in \Re$.

2.3.1 Fuzzifier

This layer retains all information about fuzzy sets and their characteristic shapes i.e membership functions. The fuzzifier firstly takes as input a crisp variable $\overline{\boldsymbol{u}} \in \Re^n$, then transforms it into a linguistic variable $\boldsymbol{u}$ by mapping $\overline{\boldsymbol{u}}$ from domain $\Re^n$ to domain $\boldsymbol{U} \subset \Re^n$,

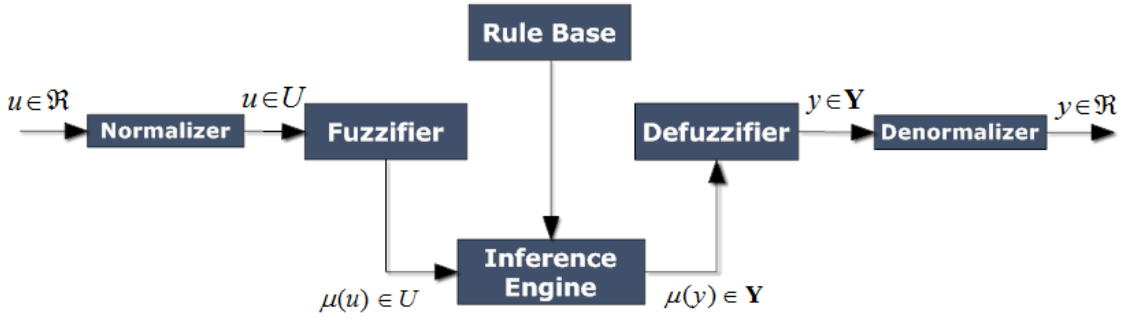


Figure 2.8: Fuzzy inference block diagram

which is a normalized UD $\in [-1, 1]$. Fuzzifier also assigns $\boldsymbol{u}$ with a grade of membership in all fuzzy sets defined in **U** [7](pp.49-51). This degree of membership can be seen as a vector whose elements are membership grades per each linguistic value. The map is defined according to (2.2). Analytically the fuzzifier computes:

$$\mu_{\overline{A}_i}(u_i) = \left[T\left\{\mu_{A_i'}(u_i), \mu_{A_i^1}(u_i)\right\} \;\; T\left\{\mu_{A_i'}(u_i), \mu_{A_i^2}(u_i)\right\} \;\; \ldots \;\; T\left\{\mu_{A_i'}(u_i), \mu_{A_i^{n_{mf}^i}}(u_i)\right\} \right] \tag{2.55}$$

where A_i', A_i^l are fuzzy sets with membership functions $\mu_{A_i'}, \mu_{A_i^l}$, $i = 1, \ldots, n$ and $l = 1, \ldots, n_m^i f$ with n the number of inputs and n_{mf}^i the number of fuzzy sets for input i. In most cases and because it is less computationally costly, $\mu_{A'}$ is defined with a singleton membership function:

$$\mu_{A'}(\boldsymbol{u}) = \begin{cases} 1 & \text{if} \quad \boldsymbol{u} = \overline{\boldsymbol{u}} \\ 0 & \text{if} \quad \boldsymbol{u} \neq \overline{\boldsymbol{u}} \end{cases} \tag{2.56}$$

for singletone fuzzifiers (2.55) can be reduced to

$$\mu_{\overline{A}_i}(u_i) = \left[\mu_{A_1^1}(u_i) \;\; \mu_{A_1^2}(u_i) \;\; \ldots \;\; \mu_{A_1^{n_{mf}^i}}(u_i) \right] \tag{2.57}$$

The fuzzifier using (2.56) is known as a singletone fuzzifier, Figure 2.10 and Figure 2.9 are examples of a singletone and non singletone fuzzifiers. Figure 2.10 shows two inputs and one output, input u_1 membership function is composed by three fuzzy sets A_{11}, A_{12} and A_{13}. Considering $u_1 = 0.4$ the fuzzifier returns the vector $\mu_A(0.4) = [\mu_{A_{11}}(0.4) \;\; \mu_{A_{12}}(0.4) \;\; \mu_{A_{13}}(0.4)] = [0.25 \;\; 0.95 \;\; 0.14]$ containing the membership grades for all defined fuzzy sets.

2.3.2 Rule Base

The rule base contains the fuzzy system reasoning knowledge, and relates system input variables to its output variables. The rule base is composed by a set of propositions which according to [3], are classified into four types. Consider variable $u_1 \in \boldsymbol{U}$ and $y \in \boldsymbol{Y}$,

fuzzy set $A \in \boldsymbol{U}$ and fuzzy set $B \in \boldsymbol{Y}$, and also variable S which is a fuzzy truth modifier belonging to a fuzzy set. Propositions can be classified as:

- unconditional and unqualified propositions;

$$p : u_1 \; is \; A$$
$$T(p) = \mu_A(V)$$

- unconditional and qualified propositions;

$$p : u_1 \; is \; A \; is \; S$$
$$T(p) = S(\mu_A(V))$$

- conditional and unqualified propositions;

$$p : If \; u_1 \; is \; A, \; Then \; y \; is \; B$$

  the proposition can also be described as a fuzzy relation subsection 2.2.4

$$p : (u_1, y) \; is \; R$$
$$R(u_1, y) = I\left\{\mu_A(u_1), \mu_B(y)\right\}, \;\; R \in \boldsymbol{U} \times \boldsymbol{Y}$$

  where I is a fuzzy implication

- conditional and qualified propositions.

$$p : If \; u_1 is A, \; Then \; y \; is \; B \; is S$$

Both first and second classes, are only characterized by an antecedent part or a premise part, there is no implication between premises and consequents, as it happens with the last two classes. Another important concept, although not considered in this work, is the use of linguistic hedges [3] where linguistic terms are modified when combined with special linguistic terms, modifying previously defined propositions. For instance, the use of linguistic hedge "very" along with linguistic value "high", can modify rule premiss:

- "Temperature is very high is true"
- "Temperature is high is very true"
- "Temperature is very high is very true"

Based on generalized modus ponens, fuzzy reasoning premises are constructed as:

$$\begin{aligned} \text{Premise} &: u\ is A' \\ \text{Implication} &: IF\ u\ is\ A\ THEN\ y\ is\ B \\ \text{Conclusion} &: y\ is\ B' \end{aligned} \tag{2.58}$$

with A, A', B, B' fuzzy sets and u a linguistic variable. Based on (2.45), fuzzy set B' can be obtained by:

$$\begin{aligned} B' &= A' \circ R \\ &= A' \circ (A \rightarrow B) \\ &\text{with} \\ \mu_{B'}(y) &= \mu_{A' \circ R}(y) \\ &= \sup_{u \in \boldsymbol{U}} \left\{ \mu_{A'}(u) *^{T} \mu_R(u, y) \right\} \end{aligned} \tag{2.59}$$

The membership function of the fuzzy relation R, having a known μ_A and μ_B, is computed as follows:

$$\begin{aligned} \mu_R(u, y) &= \mu_{A \rightarrow B}(u, y) \\ \mu_{A \rightarrow B}(u, y) &= I(\mu_A(u), \mu_B(y)) \end{aligned} \tag{2.60}$$

where I is a fuzzy implication which will be explained in next subsection. For example consider Figure 2.9 which is composed by the conditional and unqualified proposition "IF u_1 is A THEN y is B", where u_1 and y are linguistic variables and A, A', B are fuzzy sets with known membership functions. Fuzzy set B' is obtained according to (2.59) and (2.60) using a mandani inference which will be defined in next section. The previous fuzzy rules consequent are composed by fuzzy sets although, other types of consequent can be considered, not requiring the use of fuzzy sets. An example is the fuzzy reasoning according to Takagi-Sugeno (TS) [8][9]. The TS rule base shares the same premise structure as previously defined (2.58) although consequent output is a crisp variable $y = f(u, \theta)$. The generalized modus ponens of a TS rule base can be seen as a generic rule base Equation 2.58 with consequents composed by singletone membership functions [6]:

$$\begin{aligned} &\mu_B : \Re \rightarrow [0, 1] \\ &\mu_B = \begin{cases} 1 & \text{if} \quad y = f(u, \boldsymbol{\theta}) \\ 0 & \text{if} \quad y \neq f(u, \boldsymbol{\theta}) \end{cases} \end{aligned} \tag{2.61}$$

From Equation 2.61 the single tone MF is not restricted to fuzzy domain $\boldsymbol{Y}$ which is a normalized UD. A constraint must be imposed to TS defuzzification, allowing the denormalizer concept from Figure 2.8:

$$\mu_B = \max\left\{\min\left\{f(u, \boldsymbol{\theta}), \max(UD)\right\}, \min(UD)\right\} \tag{2.62}$$

As an example of a TS consequent structure see Figure 2.10.

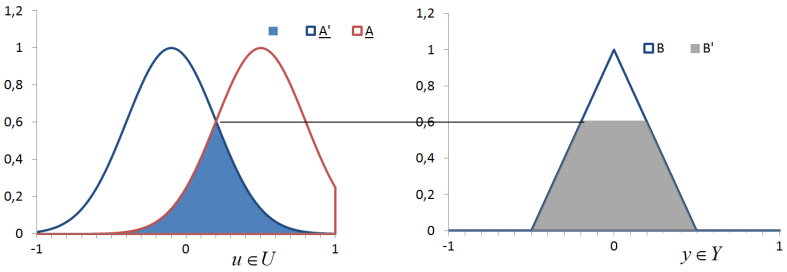


Figure 2.9: A fuzzy example using Mandani inference

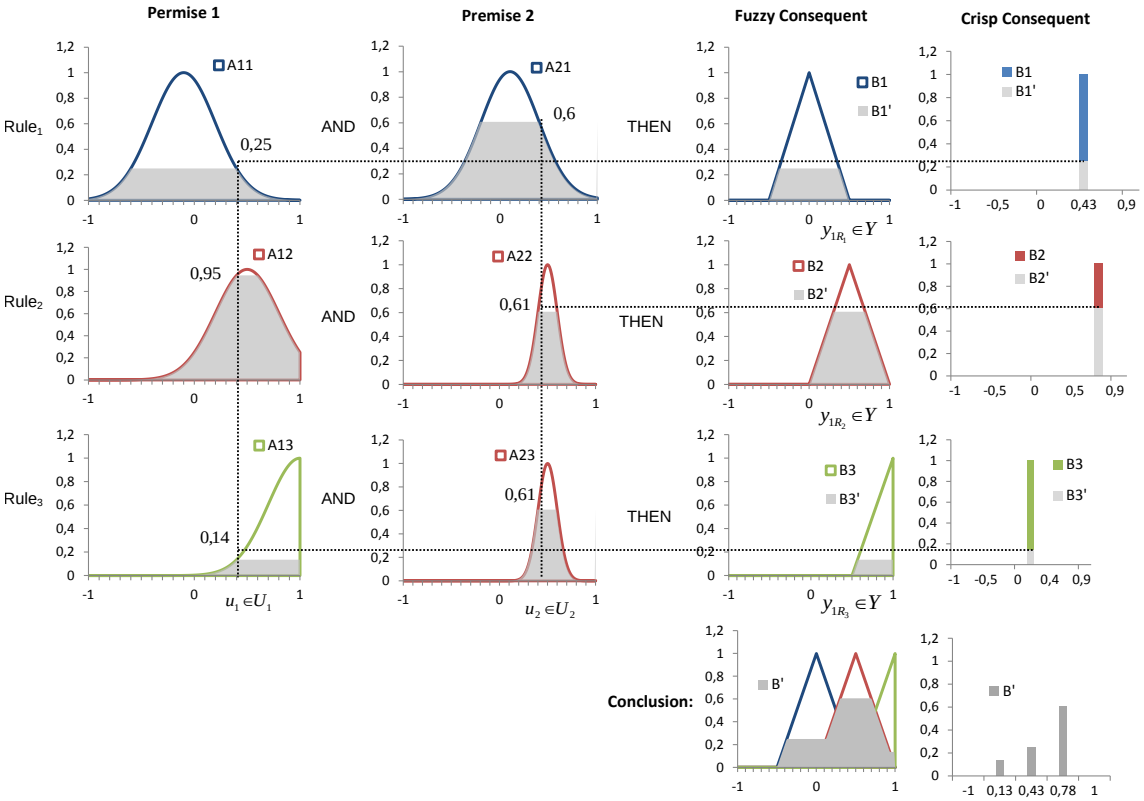


Figure 2.10: A fuzzy example using Mandani inference for a TS system and a standard fuzzy system

2.3.3 Inference Engine

Inference engine assigns trough fuzzy implications, a truth value on each fuzzy. Basic notions of fuzzy implications according to [4][3] can be defined as:

Definition (Fuzzy implication) - Fuzzy implication $I(a, b)$ is as function $I : [0, 1] \times [0, 1] \rightarrow [0, 1]$ for any a and $b \in [0, 1]$. Distinct classes of fuzzy implications can be defined:

$$I(a, b) = S(N(a), b) \tag{2.63}$$

with S a T-conorm and $N(a)$ a negation,

$$I(a,b) = \sup_u \{u \in [0,1] | T(a,u) \leq b\} \tag{2.64}$$

where T is a T-norm

$$I(a,b) = S(N(a), T(a,b)) \tag{2.65}$$

and finally

$$I(a,b) = S(T(N(a), N(b)), b) \tag{2.66}$$

with S, T, N satisfying the De Morgan Laws. Different implications can be obtained using specific T-norms, T-conorms and negations see [3]. In practice other group of implications known as mandani implications, can also be used. Mandani implications does not obey to axiomatic definition of fuzzy implications [10], they are classified as "engineering implications" according to [11][12]. This work will consider inference systems based only on mandani implications, which can be described as:

$$\begin{aligned} I(a,b) &= \min\{a,b\} \\ &= a.b \\ &= T\{a,b\} \end{aligned} \tag{2.67}$$

which is a conjunction for inference. The rule aggregation is performed by the t-conorm:

$$\begin{aligned} S\{a_1, a_2, \ldots, a_n\} &= a_1 *^S a_2 *^S *^S \ldots *^S a_n \\ &= \overset{n}{\underset{i=1}{S}} \{a_i\} \end{aligned} \tag{2.68}$$

while the antecedent aggregation of each rule is performed by the t-norm:

$$\begin{aligned} T\{a_1, a_2, \ldots, a_n\} &= a_1 *^T a_2 *^T *^T \ldots *^T a_n \\ &= \overset{n}{\underset{i=1}{T}} \{a_i\} \end{aligned} \tag{2.69}$$

The theory presented bellow will focus on MISO systems, since a MIMO model can be achieved by aggregating M MISO model. Consider a system with n inputs

$$u_1, u_2, \ldots, u_n \in \Re$$

and one output $y \in \Re$, the rule base will be composed by C conditional rules:

$$R^{(k)}: \begin{cases} \text{IF} & u_1 & \text{is} & A_1^k & \text{AND} & \\ & u_2 & \text{is} & A_2^k & \text{AND} & \ldots \\ & u_n & \text{is} & A_n^k & \text{AND} & \\ \text{THEN} & y & \text{is} & B^k & & \end{cases} \tag{2.70}$$

or in a matrix form:

$$R^{(k)} : \text{IF } \boldsymbol{u} \text{ is } A^k \text{ THEN } y \text{ is } B^k \tag{2.71}$$

with $A^k = A_1^k \times A_2^k \times \ldots \times A_n^k$, where $A_1^k, A_2^k, \ldots, A_n^k$ are fuzzy sets with membership functions $\mu_{A_i^k}(u_i),\ i = 1, \ldots, n$. Consider also input linguistic variables $\boldsymbol{u} \in \boldsymbol{U}$, output linguistic variable $y \in \boldsymbol{Y}$ and fuzzy set B^k defined with membership function $\mu_{B^k}(y)$ with $k = 1, \ldots, C$. The inference process is computed according next steps:

1. For all k compute the firing strength of each rule R^k according to (2.69):

$$\tau_k(\boldsymbol{u}) = \underset{i=1}{\overset{n}{T}} \left\{ \mu_{A_i^k(u_i)} \right\} = \mu_{A^k}(\boldsymbol{u}) \tag{2.72}$$

2. For each rule compute the fuzzy set $\overline{B}^k$ and its associated membership function according to (2.59) and (2.60). Note that when single tone fuzzifier is used equation (2.60) can be reduced to:

$$\begin{aligned} \overline{B}^k &= A' \circ (A^k \to B^k) \\ \mu_{\overline{B}^k} &= \sup_{u \in \boldsymbol{U}} \left\{ \mu_{A'}(\boldsymbol{U}) *^T \mu_{A^k \to B^k}(\boldsymbol{u}, y) \right\} \\ &= \mu_{A^k \to B^k}(\boldsymbol{u}, y) \\ &= I(\mu_{A^k}(\boldsymbol{u}), \mu_{B^k}(y)) \end{aligned} \tag{2.73}$$

   using a Mandany inference equation (2.73) is reduced to a t-norm:

$$\begin{aligned} \mu_{\overline{B}^k} &= I\text{eng}(\mu_{A^k}(\boldsymbol{u}), \mu_{B^k}(y)) \\ &= T\left\{ \mu_{A^k}(\boldsymbol{u}), \mu_{B^k}(y) \right\} \end{aligned} \tag{2.74}$$

3. Aggregate all $\overline{B}^k$ for all rules to obtain using a Mandani approach:

$$\begin{aligned} B' &= \bigcup_{k=1}^{C} \overline{B}^k \\ \mu_{B'} &= \underset{k=1}{\overset{C}{S}} \mu_{\overline{B}^k}(y) \end{aligned} \tag{2.75}$$

The previous steps are valid both for a crip consequents (TS rule base system), with $\mu_{B^k}(y)$ according to (2.56), and for fuzzy consequents as it can be see in Figure 2.10.

2.3.4 Defuzzifier

This module is responsible to produce an output crisp variable based on aggregated output membership functions μ'_B, which were obtained according to the inference module previously defined. From all existing defuzzification methods [12], solution to be

achieved will be focused on the center of area method (COA), which is defined as:

$$\overline{y} = \frac{\int\limits_{\boldsymbol{Y}} y.\mu_{B'}(y)dy}{\int\limits_{\boldsymbol{Y}} \mu_{B'}(y)dy} \tag{2.76}$$

The discrete form of COA is known as weighted average method (WAM):

$$\overline{y} = \frac{\sum\limits_{k=1}^{C} y^k.\mu_{B'}(y^k)}{\sum\limits_{k=1}^{C} \mu_{B'}(y^k)} \tag{2.77}$$

with $k = 1, \ldots, C$, y^k the center of rule k consequent membership function $\mu_{B^k}(y)$. Usually this method assumes a symmetry on output fuzzy:

$$\mu_{B^k}(y^k) = \underset{y \in \boldsymbol{Y}}{max} \{\mu_{B^k}(y)\} \tag{2.78}$$

Above method equals the defuzzification of a TS system [13] when consequent fuzzy set is a singletone (2.57):

$$\overline{y} = \frac{\sum\limits_{k=1}^{C} y^k.\tau_k(\boldsymbol{u})}{\sum\limits_{k=1}^{C} \tau_k(\boldsymbol{u})} \tag{2.79}$$

where $\tau_k(\boldsymbol{u})$ is defined as in (2.72). For above case the restriction of (2.78) has no meaning.

2.4 Conclusion

This chapter provided the reader with basic theory of fuzzy inference systems, which will allow an easily understanding of all concepts and methodologies to be adopted during the next sections. At this point reader should be capable of distinguish the main differences between fuzzy and crisp variables, understand the fuzzy reasoning process and architectures using different inference solutions. Apart from presented theory, it is worth mentioning other interesting topics which were not adopted. Among them it can be highlighted the concept of Type-2 fuzzy sets [14] precisely, interval valued type-2 fuzzy sets which allows the introduction of uncertainty variables in the degrees of membership. The given examples during the fuzzy inference section 2.3 regarding fuzzification and inference process, were restricted to singletone membership functions. However, literature [4] provides more generic inference mechanisms for fuzzy relations projection into Cartesian space.

3

Fuzzy System Modelling

3.1 Introduction

Figure 3.1: Blackbox modelling problem

Current chapter presents several fuzzy system structuring possibilities, handling the problem of "black box" system identification. Having no previous plant knowledge Figure 3.1, the identification process comprises the basic steps [15]:

1. Identification tests or experiments
2. Model order/structure selection
3. Parameter estimation
4. Model validation

In a first step, the planner must specify the sampling time, create an input signal for process excitation, and then capture its output response. Secondly, a model parametric structure must be defined, which could follow some well known parametrization structures. For instance:

ARX - The autoregressive with exogenous input model is widely used due to its applicability to least squares (LS) and recursive least squares (RLS) algorithms. The ARX is

an nth order differential equation:

$$\begin{aligned} y(k) &= -a_1 y(k-1) - \cdots - a_{n_a} y(k-n) + b_1 u(k-n_k) + \cdots + b_m u(k-m) + e(k) \\ &= \varphi(k)\theta + e(k) \end{aligned}$$

with

$$\begin{aligned} \varphi &= [-y(k-1) - \cdots - y(k-n) + u(k-n_k) + \cdots + u(k-m)] \\ \theta &= [a_1 \ \cdots \ a_{n_a} \ b_1 \ \cdots \ b_m]^T \end{aligned}$$

where

$$m = n_b - n_k + 1 \tag{3.1}$$

NARX - The nonlinear ARX is an ARX extension for the nonlinear case:

$$y(k) = f(\varphi(k), \theta) + e(k) \tag{3.2}$$

ARMAX - The ARMAX model as proposed by [16], consist on the following linear set of differential equations:

$$\begin{aligned} y(k) &= \frac{B(q)}{A(q)} u(k) + \frac{C(q)}{A(q)} e(k) \\ &\text{where} \\ A(q) &= 1 + a_1 q^{-1} + \cdots + a_n q^{-n} \\ B(q) &= \ b_1 q^{-1} + \cdots + b_n q^{-n} \\ C(q) &= 1 + c_1 q^{-1} + \cdots + c_n q^{-n} \end{aligned} \tag{3.3}$$

where $e(k)$ is white noise with zero mean and variance R. The output produced from above methods, at each instance, is correlated with information from previous samples in a "sliding window" basis. This property may turn model impractical if an enormous amount of past data is considered, resulting in an enormous number of variables and parameters.

State Space - Another method that not only allows higher numerical efficiency, but is also suitable for Kalman filtering optimization methods, is known by subspace modelling [15]. State space model design can be described by the following matrix form:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + \eta(k) \\ y(k) &= Cx(k) + Du(k) + e(k) \end{aligned} \tag{3.4}$$

where $w(k),\ v(k)$ are white noise with zero mean and variance $R^\eta,\ R^e$ respectively. Having chosen the model structure, the third step comprises parameter identification. During this stage, a model fitting problem based on a cost function minimization must be solved, using and offline, online, or both approaches. The last step comprises model validation, which can be conducted through a residual analysis and crossed validation methods. In

order to include above defined parametrization methods in a fuzzy structure, most important fuzzy architectures have to be defined firstly.

3.2 Fuzzy Modelling

3.2.1 Fuzzy NARX Structure

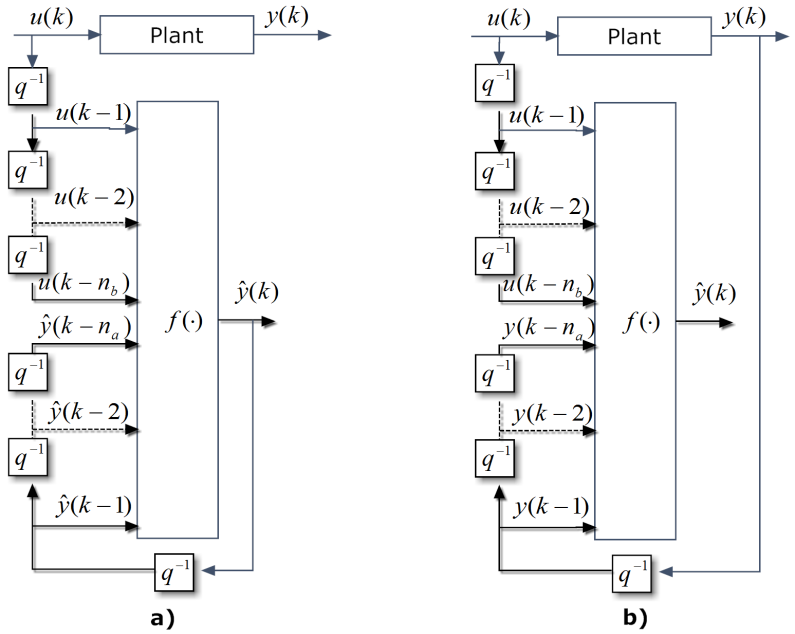


Figure 3.2: Block schema of an NARX structure for **a)** series-parallel identification **b)** series identification

Fuzzy modelling can be parametrized as a NARX structure Figure 3.2, where for each iteration, fuzzy predictor output $\hat{y}(k)$ is obtained trough a nonlinear function $f(\varphi, \theta)$, composed by a regression vector φ and a parametrization vector θ according to (3.1). An example of fuzzy NARX is the TSK approach, where rules consequents are structured as an ARX parametrization. The output of a TSK FS, is obtained through a nonlinear fuzzy combination of several ARX models. Consider the following MISO system with

two inputs $u_1; u_2$ and one output y, with a fuzzy rule based constructed as follows:

$$\begin{aligned} R^1 :\ & IF \varphi_1\ IS\ A_1^1\ AND\ \dots\ \varphi_n\ IS\ A_n^1 \\ & THEN y = \varphi\theta^1 \\ & \vdots \\ R^C :\ & IF \varphi_1\ IS\ A_1^C\ AND\ \dots\ \varphi_n\ IS\ A_n^C \\ & THEN y = \varphi\theta^C \end{aligned} \tag{3.5}$$

with

$$\begin{aligned} \varphi = [&-y(k-1) - \cdots - y(k-n) + u_1(k-1) + \cdots \\ &+u_1(k-n_{b1}) + u_2(k-1) + \cdots + u_2(k-n_{b2})] \\ \theta = [&a_1\ \cdots\ a_{n_a}\ b_1\ \cdots\ b_{n_{b1}}\ b_2\ \cdots\ b_{n_{b2}}]^T \end{aligned}$$

The total fuzzy output is computed as:

$$\begin{aligned} \hat{y}(k) &= \frac{\sum_{i=1}^{C} \tau^i(\varphi, w^i)\varphi\theta^i}{\sum_{i=1}^{C} \tau^i(\varphi, w^i)} \\ &= \sum_{i=1}^{C} \beta_i \varphi\theta^i \end{aligned} \tag{3.6}$$

where

$$\tau^i(\varphi, w^i) = \prod_{j=1}^{n} \mu_{A_j^i}(\varphi, w_j^i)$$

$$\beta_i = \frac{\tau^i(\varphi, w^i)}{\sum_{i=1}^{C} \tau^i(\varphi, w^i)}$$

with w_j^i the membership functions parameters. To compute a MIMO model there are two approaches, one is to create an augmented regression vector φ^A with the previous output values from all output variables, i.e

$$\varphi_u = \begin{bmatrix} u_{1,(k-nk_1)} \\ \dots \\ u_{1,(k-nb_1-nk_1+1)} \\ \dots \\ u_{n,(k-nk_n)} \\ \dots \\ u_{n,((k-nb_n-nk_n+1))} \end{bmatrix}^T \quad \varphi_{y_1} = \begin{bmatrix} y_{1,(k-1)} \\ \dots \\ y_{1,(k-na_1)} \end{bmatrix}^T \quad \varphi_{y_m} = \begin{bmatrix} y_{m,(k-1)} \\ \dots \\ y_{m,(k-na_m)} \end{bmatrix}^T$$

$$\varphi^A = \begin{bmatrix} \varphi_{y_1}^T \\ \cdots \\ \varphi_{y_m}^T \\ \cdots \varphi_u^T \end{bmatrix}^T$$

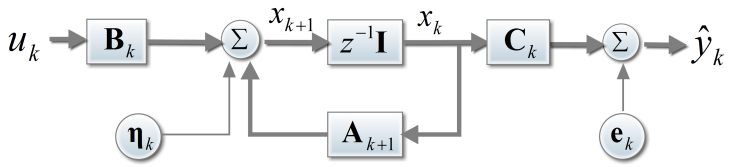
the augmented input vector is then included in the premise part. Fuzzy consequents are then increased with new output functions which will have a new coefficient matrix θ_{y_m}, e.g:

$$\begin{aligned} R^i :\ & IF \varphi_1^A \ IS \ A_1^i \ AND \ \dots \ AND \ \varphi^A{}_n \ IS \ A_n^i \\ & THEN \ y_1 = \left[\varphi_{y_1}^T \ \varphi_u^T\right] \theta_{y_1}^i \ AND \ \cdots \ AND \ y_m = \left[\varphi_{y_m}^T \ \varphi_u^T\right] \theta_{y_m}^i \end{aligned} \tag{3.7}$$

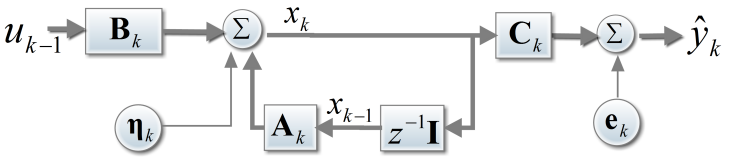
Augmenting the regression vector will increase the number of rules by a factor of $\prod_{i=1}^{n} P_i$, with P_i the number of fuzzy partitions for each element of the augmented vector. Another approach is to use only one output for each consequent and create m sets of rules, using for each set j the parametrization vector $\varphi^{S_j} = \left[\varphi_{y_j}^T \ \varphi_u^T\right]$ in the premise part, and output $y_j = f(\varphi^{S_j}, \theta^{S_j})$ in the consequent part. This will lead to a total number of rules $\sum_{j=1}^{m} \prod_{i=1}^{n^j} P_i$ with n^j the length of φ^{S_j}. For both methods the number of rules increases drastically with the number of variables. For practical cases using also a mandani inference approach it is recommended theories presented in [17] and [18]. Another method for fuzzy consequents local models design, is to use subspace methods to be introduced during next section.

3.2.2 Fuzzy State Space

This section handles the FS structuring using fuzzy state space local models. A general state space model representation is shown in Figure 3.3, where representation 3.3(b) is sometimes preferable to 3.3(a) due to a better flow chart process understanding. Several authors proposed fuzzy state space systems (FSS) for process identification and control. From [19] it was considered a model reference adaptative fuzzy control (MRAFC) based on a FSS system however, presentation only considered a phase variable in a state-space form (direct conversion from an nth-order ODE). A stability analysis of a global-state FSS



(a) State Space with input/output delay inside model



(b) State Space with input/output delay outside model

Figure 3.3: Block schema of a State Space representation

model considering a state feedback controller, was introduced by [20]. Also, [21] presented a stability analysis of fuzzy state space models in a closed (with controller) and open loop (no controller). A similar approach was taken in [22], where it was tried to incorporate a stabilization gain into a FSS model by solving a LMI problem. Presented work will not introduce fuzzy stability theory which could be applied in proposed architecture. This work will not contribute with any stability analysis although, for future investigations regarding FSS stability, it is recommended as a starting point the theory presented in [23]. The rule structure of a fuzzy system described in a state space fashion, is projected according next example. For instance consider a SISO system with a state matrix $\mathbf{x} = [x_1\ x_2\ \ldots\ x_n]$ and control input u, the FSS model is constructed as:

$$\begin{aligned} R^i:\ & IF x_1(k)\ IS\ N^i_{x_1}\ AND\ \ldots\ AND\ x_n(k)\ IS\ N^i_{x_n}\ AND\ u(k)\ IS\ N^i_u \\ & THEN\ x(k+1) = \mathbf{A}^i x(k) + \mathbf{B}^i u(k) \end{aligned} \tag{3.8}$$

the crisp state output is obtained through a WAM defuzzification introduced in previous chapters:

$$\begin{aligned}\hat{x}(k+1) &= \frac{\sum_{i=1}^{C}\tau^i(\mathbf{x},u,\mathbf{w}^i)\left(\mathbf{A}^i x(k)+\mathbf{B}^i u(k)\right)}{\sum_{i=1}^{C}\tau^i(\mathbf{x},u,\mathbf{w}^i)} \\ &= \sum_{i=1}^{C}\beta_i\left(\mathbf{A}^i x(k)+\mathbf{B}^i u(k)\right)\end{aligned}$$

where (3.9)

$$\begin{aligned}\varphi &= [\mathbf{x}\ u] \\ \tau^i(\varphi,\mathbf{w}^i) &= \prod_{j=1}^{n}\mu_{N_j^i}(\varphi_j,w_j^i) \\ \beta_i &= \frac{\tau^i(\varphi_i,w^i)}{\sum_{i=1}^{C}\tau^i(\varphi,\mathbf{w}^i)}\end{aligned}$$

using straight forward substitutions, the final FSS model becomes:

$$\begin{aligned}\hat{x}(k+1) &= \mathbf{A}x(k)+\mathbf{B}u(k) \\ &\text{with} \\ \mathbf{A} &= \sum_{i=1}^{C}\beta_i\mathbf{A}^i \\ \mathbf{B} &= \sum_{i=1}^{C}\beta_i\mathbf{B}^i\end{aligned} \tag{3.10}$$

The plant output sensors are obtained by $\hat{y}(k)=\mathbf{C}\hat{x}(k)$, note that this output is computed after fuzzy inference process. Other possibility is to include plant outputs during fuzzy reasoning as proposed by [20] in this case, fuzzy local models are augmented with a new function:

$$\begin{aligned}\hat{x}(k+1) &= \mathbf{A}x(k)+\mathbf{B}u(k) \\ \hat{y}(k) &= \mathbf{C}x(k) \\ &\text{with} \\ \mathbf{C} &= \sum_{i=1}^{C}\beta_i\mathbf{C}^i\end{aligned} \tag{3.11}$$

Because most authors presented an analysis based on a closed loop approach, it is worth presenting an overview of FSS feedback controllers. Proposed closed loop solution, was based on theory presented in [20], where it was introduced restrictions which assures not only a local but also a global asymptotically stable closed loop with no steady state error. For instance, based on model of (3.11) a controller based on a state feedback, can

be incorporated for each sub space partition:

$$\begin{aligned} \hat{x}(k+1) &= \mathbf{A}x(k) + \mathbf{B}u(k) \\ \hat{y}(k) &= \mathbf{C}x(k) \\ \hat{u}(k) &= -\mathbf{K}x(k) + gr(k) \\ &\text{with} \\ \mathbf{K} &= \sum_{i=1}^{C} \beta_i \mathbf{K}^i \end{aligned} \tag{3.12}$$

It is assumed that both controller and model shares the same dimension partition including their fuzzy set shapes. Local models from (3.12), can be designed in a canonical form, which according to referenced authors, guarantees a local asymptotic stability for a linear time-varying system. Meanwhile, achieved RNFS solution will not be designed in a canonical form. The purpose of such approach, is to easily extend model with MIMO capabilities, by just augmenting matrix C.

3.3 Flexible Neuro Fuzzy Modelling

Considering previous chapters concepts, next will be described how a FS system can evolve to a neuro fuzzy system, and how to compute the neuro fuzzy inference process. Before describing the NFS architecture, it is important to first define the basic notions of a neural network [24]:

Synapse - Represent the connections between neurons having an associated weight $w_{r,i} \in \Re$.

Neuron - Is the principal element of a neural network, it is also referred as the network processing element. Every neuron produces an output O_r and receives at its input the outputs from previous layers O^i_{r-1} which are connected through synapses. The output O_r is obtained through a function $O_r = f(\boldsymbol{O}_{r-1}, \boldsymbol{W}_r, \theta_r)$. In literature a neuron is composed by two sub-processing elements, one responsible to aggregate inputs through a propagation function, producing an output:

$$\begin{aligned} net_{r,j} &= f_{prop}(\boldsymbol{O}_{r-1}, \boldsymbol{W}_{r,j}) \\ &= \sum_{i=1}^{n} \left(O^i_{r-1} \times w^i_{r,j}\right) \end{aligned} \tag{3.13}$$

where i is the subscript referring to a neuron in previous layers, and j is a subscript referring to a neuron in current layer. Other authors [25] described the propagation function

as a *generalized product neuron*:

$$net_{r,j} = \prod_{i=1}^{n} \left(O_{r-1}^{i} \times w_{r,j}^{i} \right) \tag{3.14}$$

The second sub-processing element is referred as the activation function, producing output

$$O_r = f_{act}(net_{r,j}, \theta_{r,j}) \tag{3.15}$$

some well known functions which implement (3.15) exist, for instance the case of *binary threshold function*, *logistic function* or *tangent hyperbolic function*. However, other definitions can be done considering always the restriction of being differentiable, in order to allow the use of back propagation algorithms. An example of a neuron is illustrated in Figure 3.4.

Feedforward Network - In this type of network, all previous nodes are connected to posterior nodes. These architecture also defines the use of an input layer, a hidden layer which may aggregate several invisible layers, and an output layer. Connections are only permitted from one layer to the adjacent downstream layer as represented in Figure 3.5.

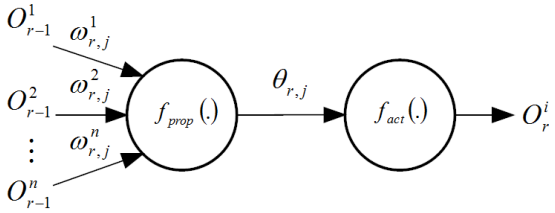


Figure 3.4: A neuron j in layer r

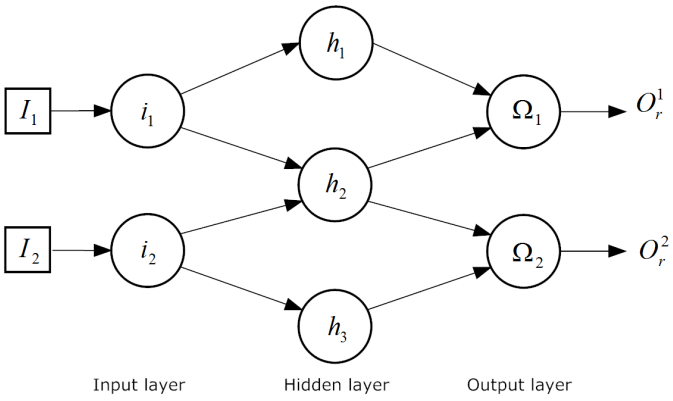


Figure 3.5: A feedforward neural network

Recurrent Network - Recurrent networks introduce the concept of cycles, where a neuron can be connected to itself over a weighted connection (*direct recurrence*), or all units

of a layer can be connected with other units of other layers (*indirect recurrence*). Units of the same layer can also be interconnected (*lateral recurrence*)[24]. Examples of recurrent networks are the Jordan network [26], Elman network [27] and Hopfield network as presented in [28].

3.3.1 Mandani-Type Neuro Fuzzy System

A generic neuro fuzzy systems was presented in [4], where apart from previous concepts, extra parameters where introduced featuring system with flexibility and robustness. Although author presented a generic representation for all types of inferences, this work will only consider the mandani case where "engineering" implications are used. Consider the weighted fuzzy system:

$$\begin{aligned} \mu_{A^k}(\boldsymbol{x}) &= \underset{i=1}{\overset{n}{T^*}} \left\{ \mu_{A_i^k}(x_i), w_{i,k}^{\tau} \right\} \\ \mu_{B'}(y) &= \underset{i=1}{\overset{C}{S^*}} \left\{ \mu_{\overline{B}^k}(y), w_k^{agr} \right\} \end{aligned} \tag{3.16}$$

where $w_{i,k}^{\tau}$ is the i-th input weight for the rule base k-th rule and T^* is a weighted t-norm defined as

$$T^* \left\{ a_1, \ldots, a_n; w_1^{\tau}, \ldots, w_n^{\tau} \right\} = \underset{i=1}{\overset{n}{T}} \left\{ 1 - w_i^{\tau}(1 - a_i) \right\} \tag{3.17}$$

where w_k^{agr} is the weight for the rule base k-th rule and S^* is a weighted s-norm defined as

$$S^* \left\{ a_1, \ldots, a_n; w_1^{agr}, \ldots, w_n^{agr} \right\} = \underset{i=1}{\overset{n}{S}} \left\{ w_i^{agr} a_i \right\} \tag{3.18}$$

for other definitions of S^* and T^* it is recommended [4]. From previous defuzzification method (2.77) flexible neuro fuzzy function is represented as:

$$\overline{y} = \frac{\sum_{r=1}^{C} y^r \cdot \underset{k=1}{\overset{C}{S^*}} \left\{ I \left(\underset{i=1}{\overset{n}{T^*}} \left\{ \mu_{A_i^k}(x_i), w_{i,k}^{\tau} \right\}, \mu_{B^k}(y^r) \right), w_k^{agr} \right\}}{\sum_{r=1}^{C} \underset{k=1}{\overset{C}{S^*}} \left\{ I \left(\underset{i=1}{\overset{n}{T^*}} \left\{ \mu_{A_i^k}(x_i), w_{i,k}^{\tau} \right\}, \mu_{B^k}(y^r) \right), w_k^{agr} \right\}} \tag{3.19}$$

Considering normal fuzzy sets where $\mu_{B^k}(y^r) = 1$ and the boundary condition of t-norms $T(a, 1) = a$ for μ_{B^k}, according to (2.74) it can be rewritten (3.16) for $y = y^r$:

$$\mu_{B'}(y^r) = S^* \left\{ \left(\underset{i=1}{\overset{n}{T^*}} \left\{ \mu_{A_i^r}(x_i), w_{i,r}^{\tau} \right\} \right), \underset{\substack{k=1 \\ k \neq r}}{\overset{C}{S^*}} \left\{ I \left(\underset{i=1}{\overset{n}{T^*}} \left\{ \mu_{A_i^k}(x_i), w_{i,k}^{\tau} \right\}, \mu_{B^k}(y^r) \right), w_k^{agr} \right\} \right\} \tag{3.20}$$

Another assumption much times taken intro practice is to consider $\mu_{B^k}(y^r) \approx 0$ for $k \neq r$ reducing (3.20) to:

$$\mu_{B'}(y^r) = \underset{i=1}{\overset{n}{T^*}} \left\{ \mu_{A_i^r}(x_i), w_{i,r}^{\tau} \right\} \cdot w_k^{agr} \tag{3.21}$$

Therefore, the crisp output for a simplified flexible mandani system is reduced to:

$$\overline{y} = \frac{\sum_{r=1}^{C} y^r \cdot \overset{n}{\underset{i=1}{T^*}} \left\{ \mu_{A_i^r}(x_i), w_{i,r}^{\tau} \right\} \cdot w_r^{agr}}{\sum_{r=1}^{C} \overset{n}{\underset{i=1}{T^*}} \left\{ \mu_{A_i^r}(x_i), w_{i,r}^{\tau} \right\} \cdot w_r^{agr}} \tag{3.22}$$

allowing the MISO neuro fuzzy representation as illustrated in Figure 3.6:

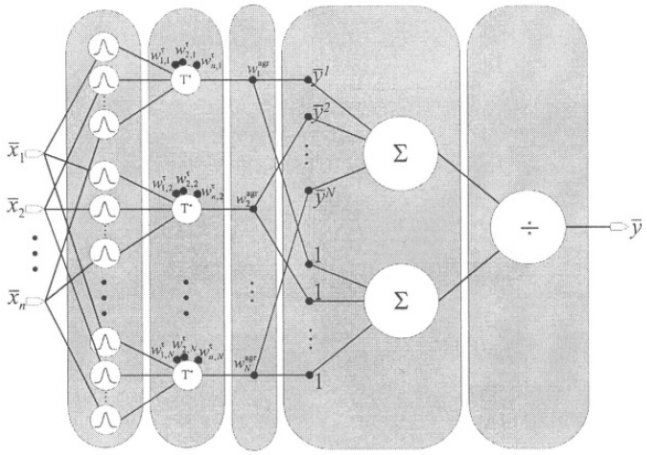


Figure 3.6: Simplified flexible neuro-fuzzy using mandani inference

3.3.2 Takagi-Sugeno Neuro Fuzzy System

TS fuzzy systems as already mention, can be seen as a mandani FS having a singletone output fuzzy set denoted by a linear function. As in mandani case, "engineering" implications are also used in a TS approach. For a TS system (3.22) can be rewritten knowing that $y^r = f\left(\boldsymbol{x}, \theta_r\right)$ and μ_{B^k} according to (2.56):

$$\overline{y} = \frac{\sum_{r=1}^{C} f(\boldsymbol{x}, \theta_r) \cdot \overset{n}{\underset{i=1}{T}} \left\{ \mu_{A_i^r}(x_i) \right\}}{\sum_{r=1}^{C} \overset{n}{\underset{i=1}{T}} \left\{ \mu_{A_i^r}(x_i) \right\}} \tag{3.23}$$

From (3.23) it can be observed that a TS system is more robust than a simplified mandani system, allowing to define for each rule a linear function rather than a simple constant (apart from any possible optimization of fuzzy set centers). As mentioned in [14] one of the most known neuro-fuzzy system using a TS Fuzzy Inference System (FIS) is the Adaptative-Network-Based FIS method (ANFIS) [29]. In general an ANFIS structure is composed by five layers three of them are hidden layers, for instance consider Figure 3.7 where: For a system with n inputs having each input P_i fuzzy partitions (membership

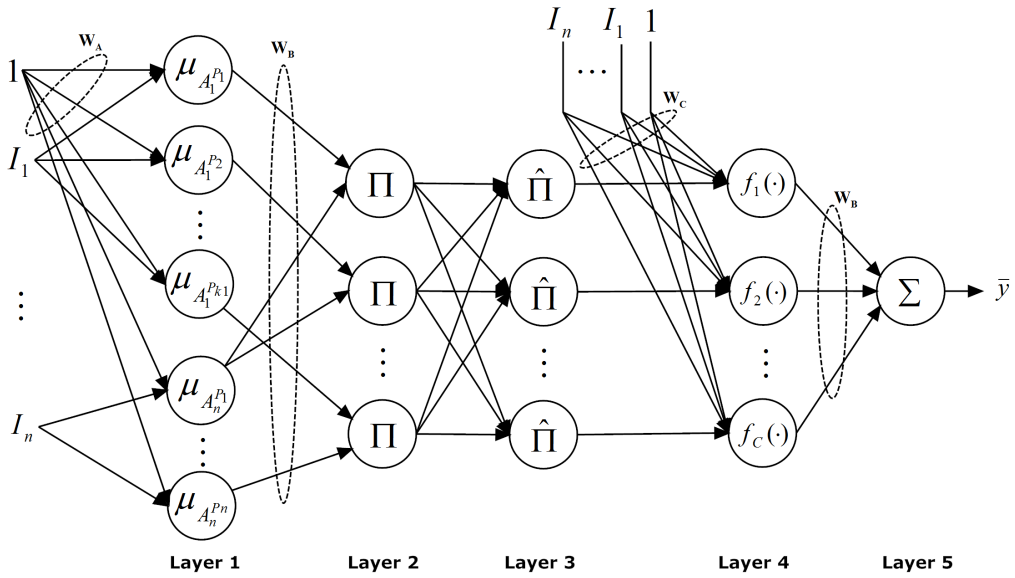


Figure 3.7: A flexible ANFIS structure

functions) the total number of rules C will be:

$$C = \prod_{i=1}^{n} P_i \tag{3.24}$$

Layer 1: Is responsible to assign a degree of membership to its inputs, the connections to this layer are weighted with $\boldsymbol{W_A}$, which corresponds to the membership functions parameters. All other connections weights are assumed to be 1. Both neuron function and the dimension of $\boldsymbol{W_A}$ depend on the type of membership functions considered.

Layer 2: This layer computes the truth value of each rule. The connections are weighted with $\boldsymbol{W_B}$ and a neuron in this layer will be denoted by following equations:

$$\begin{aligned} net_{2,j} &= \prod_{i=1}^{P_s} O_{1,i} \times W_{B^i_{2,j}} \\ O_{2,j} &= net_{2,j} \\ &\text{with} \\ P_s &= \sum_{i=1}^{n} P_i \end{aligned} \tag{3.25}$$

It is worth mention that although $net_{2,j}$ considers all connections from previous layers $i = 1, \ldots, P_s$, some of this connections does not exist in rule database, in this case planner must not create them. Another method was adopted in proposed model.

Layer 3: This layer produces a normalized truth value for each rule, the connections

weights are assumed to be 1 and a neuron in this layer will be denoted by following equations:

$$\begin{aligned} net_{3,j} &= \sum_{r=i}^{C} O_{1,i} W_{3,j}^{i} \\ O_{3,j} &= \frac{O_{2,j}}{net_{3,j}} \\ \boldsymbol{W_{3,j}} &= [1 \ \cdots \ 1]^T \end{aligned} \tag{3.26}$$

Layer 4: This layer belongs to the consequent part of reasoning assigning local models to each rule. The connections weights $\boldsymbol{W_C}$ refer to the local model parameters, other connection weights are assumed to be 1. A neuron in this layer will be denoted by following equations as described by[30]:

$$\begin{aligned} net_{4,j}^{1} &= O_{3,j} W_{C_{3,j}^{1}} \\ net_{4,j}^{2} &= O_{3,j} W_{C_{3,j}^{2}} i_1 \\ &\vdots \\ net_{4,j}^{n} + 1 &= O_{3,j} W_{C_{3,j}^{n+1}} i_n \\ O_{4,j} &= \sum_{l=1}^{n+1} net_{4,j}^{l} \end{aligned} \tag{3.27}$$

Layer 5: This layer belongs to the output layer of a standard feedforward neuro network, each neuron in this layer computes the output of a crisp variable. The connections to this neuron are weighted with $\boldsymbol{W_D}$ which assigns an aggregation weight for each rule. The neuron is defined with equations:

$$net_{5,j} = \sum_{i=1}^{C} O_{4,i} W_{3,j^i} \tag{3.28}$$

From previous definitions it can be seen that no set of equations were defined for $layer1$ as defined by (3.13). It was mainly due to complexity of some membership functions, although using sigmoidal MFs, in [31] $layer1$ was decomposed into four new layers. Another approach was taken in [30] using Gaussian MFs, meanwhile instead of using *input dimension partitioning*, author implemented *direct input space partitioning* which compared with ANFIS method, allows a reduced number of rules for the same level of accuracy. That method will not be followed in this work as Gaussian MFs doesn't satisfy a commonly used restriction (also adopted in this work):

$$\mu_A(x_j) = \sum_{i=1}^{P_j} \mu_{A_i}(x_j) = 1 \tag{3.29}$$

with μ_{A_i} a normal fuzzy set. Restriction (3.29) is very important, since it allows to discard

most fuzzy rules during online identification, this means that a constant number of rules

$$C_r = \prod_{i=1}^{n} min(P_j, 2) \tag{3.30}$$

may have a truth value different than zero. For each input at each iteration, there are only two MFs needing to be considered resulting in a membership degree greater than zero. A similar approach although using local models with a state-space formulation, was presented by [22]. It introduced an algorithm for parameter optimization based on a particle swarm optimization algorithm (PSO), although the use of a flexible inference as defined in section subsection 3.3.1 was not taken.

3.3.3 Recurrent Neuro Fuzzy System

Recurrent neuro networks (RNN) in comparison with feedforward networks, have the advantage of incorporating information about past. In order to incorporate this capability in former feedforward NFS, some authors proposed a combination of RNN with fuzzy theory, which resulted in new models known as recurrent neuro-fuzzy networks (RNFS). By analogy with the several RNN three main RNFS architectures were defined, they differ in the layer where feedback is applied, i.e [32] developed a RNFS using a TSK approach and selected the output layer as a start point of an indirect recurrence to the input layer. Since the local models were defined with an ARX parametrization, the variables used for feedback were the plant sensor outputs. Meanwhile [33] used the same layer for feedback although using a state-space approach and a mandani defuzzification method, also a structure with and without incorporation of a controller was proposed, author also presented stability analysis. A lateral recurrence in the membership layer was studied by [34] using a TSK inference, where parameter optimization was based on a back propagation (BP) algorithm. Author also presented a controller based on a MRAFC scheme including stability conditions based on restrictions imposed to the algorithm learning rate. Finally [35] demonstrated the use of a RNFS with an indirect recurrence between hidden and input layers, also a TSK inference was used and a genetic algorithm (GA) developed for optimization purpose. We will not describe authors proposed RNFS architectures since they can be seen as the standard ANFIS structure with a indirect or lateral recurrence.

3.4 Proposed Architecture for Process Identification and Control

3.4.1 Proposed RNFS Architecture for Process Identification

So far it was introduced basic theory of fuzzy sets including most commonly used inference mechanisms and their associated rule base structure. The convergence of fuzzy theory with neuro networks and their parametrization structure was also mentioned. As it can be seen, several solutions for the problem of system identification and control can

be taken. The choice of a certain structure depends on the problem to be solved and respective optimization mechanisms. The list bellow presents some useful considerations that must be taken before projecting a model and a controller:

1. Process input/output scheme, i.e MIMO/MISO/SISO/SIMO
2. Process complexity, i.e if highly nonlinear, almost linear, time varying.
3. Online vs offline identification
4. Possibility to project MRAFC controllers
5. Type of optimization procedure

The process under modelling and control is a nonlinear time varying MIMO process, this way because it is time varying, the optimization procedure to be adopted must be handled in a real time identification approach, allowing to track possible changes in process dynamics. It is also intended the design of a controller using a MRAFC approach. Based on previous considerations, the optimization procedure to be follow should be based on a Kalman filter approach, due to its capabilities of recursive estimation and the possibility to consider process noise. As will be seen later, Kalman filter was developed to solve a state-space optimization problem, despite the possibility of a non state space model be reformulated some how to this form, it is better to develop a model that is by its nature formulated in a state-space basis. Also the use of Kalman filters allows access to model states being possible to develop a MRAFC controller using a state feedback approach. This work proposes two similar structures for identification and control, both are based on a recursive neuro fuzzy system structured with a state space parametrization. A TSK inference type will be adopted due to its defuzzification simplicity in comparison with other methods e.g a Mandani inference. Proposed RNFS model architectures are illustrated in Figure 3.8,Figure 3.9 and Figure 3.10 which were based on an modified Jordan network along with an ANFIS NFS. The concept of flexibility was included by introducing input weights w^{τ} and rule weights w^{agr} as described in section subsection 3.3.1.

The major difference between architecture Figure 3.8 and Figure 3.9 is the rule base. For the first case, state variables are included in rules premise serving as criteria for subspace creation, where in the second case, the estimated outputs are used instead. Models Figure 3.9 and Figure 3.10 represents a series-parallel and a series structure as previously defined for the NARX case. In practice, the last model in comparison with the first two, may become more accurate and stable since any possible divergence in system states and respective outputs does not lead to a wrong subspace partition. The number of rules will depend on the number of output variables for the last two architectures, and on the number of states for the first architecture.
Analytically architecture of figures Figure 3.8,Figure 3.9 and Figure 3.10 can be described

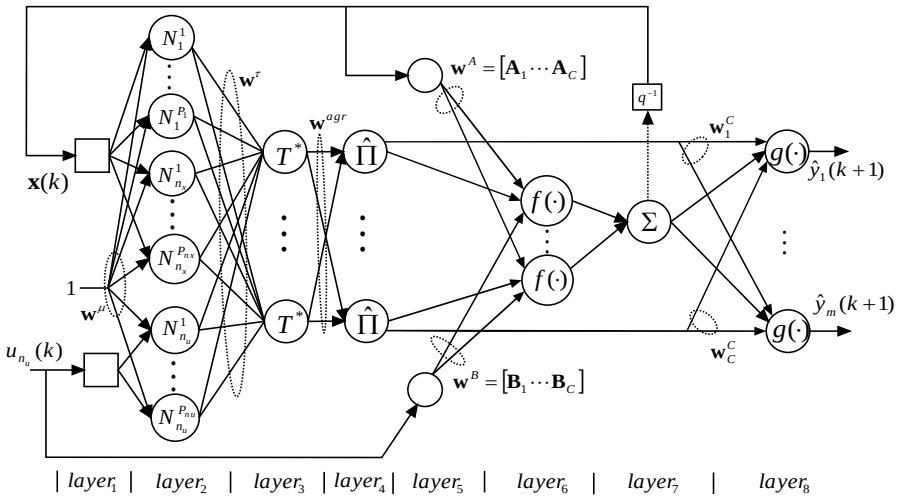


Figure 3.8: Proposed RNFS network with a state-space rule base

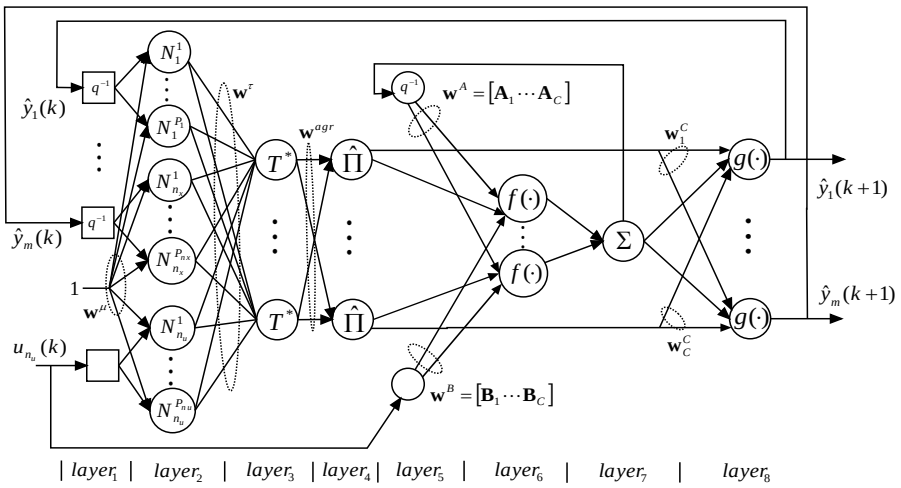


Figure 3.9: Proposed series-parallel RNFS network

as follows:

$$
\begin{aligned}
x(k) &= \sum_{i=1}^{C} \beta_i \left(w_i^A(k) x(k-1) + w_i^B(k) u(k-1) \right) \\
y(k) &= \sum_{i=1}^{C} \beta_i w_i^C(k) x(k); \qquad\qquad \beta_i = \frac{\tau^i w_i^{agr}(k)}{\sum_{r=1}^{C} \tau^r w_r^{agr}(k)} \qquad (3.31) \\
\tau^i(\varphi, w^\mu, w^\tau) &= \prod_{j=1}^{n} T^* \left\{ \mu_{N_j^i}(\varphi_j, w_j^\mu); w_{j,i}^\tau(k) \right\} \\
&= \prod_{j=1}^{n} \left(1 - w_{j,i}^\tau(k) \left(1 - \mu_{N_j^i}(\varphi_j, w_j^\mu(k)) \right) \right)
\end{aligned}
$$

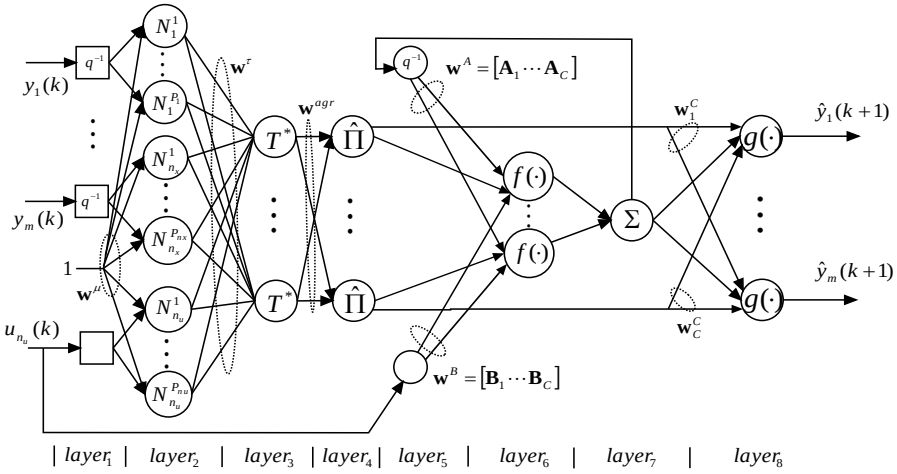


Figure 3.10: Proposed series RNFS network

$$
\begin{aligned}
&\varphi = [\mathbf{x}(k-1)\ \mathbf{u}(k-1)] \text{ using architecture Figure 3.8} \\
&\text{or} \\
&\varphi = [\hat{\mathbf{y}}(k-1)\ \mathbf{u}(k-1)] \text{ using architecture Figure 3.9} \\
&\text{or} \\
&\varphi = [\mathbf{y}(k-1)\ \mathbf{u}(k-1)] \text{ using architecture Figure 3.10}
\end{aligned}
\tag{3.32}
$$

The proposed RNFS structure contains eight layers where neurons are characterized with the following operations:

Layer 1 - Is the input layer of the RNFS network, the neurons operation are:

$$
\begin{aligned}
net_{1,j} &= \varphi_j \\
O_{1,j} &= net_{1,j}
\end{aligned}
\tag{3.33}
$$

The connections to this layer have a unitary weight value.

Layer 2 - This layer represents the fuzzification process of a FIS and belongs to the hidden layer, it is denoted by the following operations were P_k is the number of membership functions defined for input φ_k and P_s is the total number of MFs for all inputs, n_x is the number of states, n_u is the number of control inputs, C is the number of rules and index $w_{i,j}$ means the weight from neuron i in previous layer to neuron j in current

layer:

$$
\begin{aligned}
net_{2,j} &= \mu_{N_j}(O_{1,i}, w_j^{\mu}) \\
O_{2,j} &= net_{2,j}
\end{aligned}
$$

where

$$
i = \begin{cases}
1 \quad if \qquad j \leq P_1 \\
2 \quad if \quad P_1 > j \leq P_2 + P_1 \\
\quad \vdots \\
n \quad if \qquad j \leq \sum_{k=1}^{n} P_k
\end{cases} \tag{3.34}
$$

were μ_{N_j} represents the neuron associated fuzzy set shape and w_j^{μ} is the fuzzy set membership function parameters, note that although there exist only one connection having w_j^{μ} it does not constrain the number of parameters p per MF, it is assumed that $w_j^{\mu} \in \Re^{1\times p}$. Connections from previous layers are unitary (weight is one).

Layer 3 - In this layer the rules premise degrees of truth are computed, neurons in this layer are denoted with the following operation:

$$
\begin{aligned}
net_{3,j} &= \prod_{i=1}^{P_s} \left(1 - w_{i,j}^{\tau} \left(1 - O_{2,i}\right)\right) \\
O_{3,j} &= net_{3,j}
\end{aligned} \tag{3.35}
$$

This layer is similar to the one defined in (3.25), although it uses a t-norm T^* as defined in (3.17). Although figure Figure 3.8 presented all possible connections from $layer_2^{i^{th}} \underset{w_{i,j}^{\tau}}{\rightarrow} layer_3^{j^{th}}$, it is possible to make these connections neutral in order to reflect the real rules premises by considering $w_{i,j}^{\tau} = 0$.

Layer 4 - Taking the same approach as in (3.26), this layer produces a normalized degree of truth for each premise, although another weighting parameter was introduced allowing flexibility during s-norm S^* computation according to (3.18). Layer is denoted with:

$$
\begin{aligned}
net_{4,j} &= \sum_{i=1}^{C} O_{3,i} w_{i,j}^{agr} \\
O_{4,j} &= \frac{O_{3,j} w_(j,j)^{agr}}{net_{4,j}}
\end{aligned} \tag{3.36}
$$

Layer 5 - This layer can be seen as an input layer in a hidden layer, where the outputs equal inputs:

$$
\begin{aligned}
net_{5,j} &= \varphi_j \\
O_{5,j} &= net_{5,j}
\end{aligned} \tag{3.37}
$$

Layer 6 - The output of each neuron in this layer represents the local crisp models attached to each rule, the function in this layer is characterized by:

$$\begin{aligned} net_{6,j} &= \prod_{i=1}^{n_x} O_{5,i} w_{i,j}^{A} + \prod_{i=n_x+1}^{n} O_{5,i} w_{i-n_x,j}^{B} \\ O_{6,j} &= O_{4,j} net_{6,j} \\ &\text{with} \\ n &= n_x + n_u = length(\varphi) \end{aligned} \tag{3.38}$$

Layer 7 - This layer does the defuzzification of states aggregating all local models produced from previous layer. Comparing with ANFIS structure this layer functionality corresponds to its output layer (3.28), although in proposed model it is the last layer in the hidden layer:

$$\begin{aligned} net_{7,j} &= \sum_{i=1}^{C} O_{6,n_x(i-1)+j} \\ O_{7,j} &= net_{7,j} \end{aligned} \tag{3.39}$$

Layer 8 - Is the output layer of proposed RNFS model, it also belongs to a fuzzy domain although it performs a combination from previous defuzzified states producing model sensor output values:

$$\begin{aligned} net_{8,j} &= \sum_{l=1}^{C} \sum_{i=1}^{n_x} O_{7,i} w_{i,j,l}^{C} \\ O_{8,j} &= net_{8,j} \end{aligned} \tag{3.40}$$

Besides having presented three types of architectures, it will be used during implementation the architecture according to Figure 3.10. This way there is no need to constrain system states to Universe of Discourse nor the system becomes dependent of output accuracy for the case of architecture Figure 3.9.

3.4.2 Proposed MRAFC Control Architecture

This subsection aims to present a controller design which together with RNFS model previously defined, will lead plant system outputs to a desired reference. The proposed controller architecture was based on equations presented in (3.12) and on closed loop diagram according to Figure 1.2. For the problem of output tracking, an optimization algorithm Figure 1.3 with a cost function based on the error between desired and current output values is going to be adopted. A model based controller will be adopted similar to what was proposed by [36] although, this work will use a state space model approach as presented in [22] with some major differences, i.e the used rules premisses match model rules premisses and both a reference gain and a proportional error gain will be considered when computing next control action. Analytically, the proposed controller can be

described by:

$$\begin{aligned} u(k) &= \sum_{i=1}^{C} \beta_i \left(-w_i^K(k)x(k) + w_i^R(k)y^d(k) + w_i^P(k) \left(y^d(k) - y(k) \right) \right) \\ \tau^i(\varphi, w^\mu(k), w^\tau(k)) &= \prod_{j=1}^{n} T^* \left\{ \mu_{N_j^i}(\varphi_j, w^\mu(k)); w_i^\tau(k) \right\} \\ &= \prod_{i=1}^{n} \left(1 - w_i^\tau(k) \left(1 - \mu_{N_j^i}(\varphi_j, w^\mu(k)) \right) \right) \\ \beta_i &= \frac{\tau^i w_i^{agr}(k)}{\sum_{r=1}^{C} \tau^r w_r^{agr}(k)} \\ \varphi &= [\mathbf{y}(k)\ \mathbf{u}(k)] \end{aligned} \tag{3.41}$$

Controller premisses might be structured in two different ways:

- Premisses considering previous control actions $u(k-1)$ and previous plant outputs $y(k-1)$
- Premisses considering actual desired outputs $y^d(k)$ and previous plant outputs $y(k-1)$

The first method has the advantage that only the model premisses (3.31) are optimized decreasing computational effort although, controller stability might be affected by model membership function optimization. Regarding the second method, the advantage of having the reference (also known by desired outputs) instead of control actions in controller premisses, dues to the rule selection for each iteration. In this case, controller rule selection will be driven by the reference allowing controllers to be more precise and stable since any controller instability will not affect the rule selection. Meanwhile, the rule switching of reference based controllers might become sharper and can lead to abrupt changes in control actions. To overcome this effect, it is recommended a reference filtering. Combining (3.31) with (3.41) will result in (3.42) and a RNFS closed loop diagram can be represented by Figure 3.11. Figure pink area represents the premise evaluation, blue area represents the model evaluation and green area respects the controller evaluation. It should be highlighted that rule premise evaluation in case of controller, can have instead of control actions $u(k-1)$ a reference value $y^d(k)$ as already mentioned. Differences

between both approaches will be handled during experimental results.

$$
\begin{aligned}
x(k) &= \sum_{i=1}^{C} \beta_i \left(w_i^A(k) x(k-1) + w_i^B(k) u(k-1) \right) \\
y(k) &= \sum_{i=1}^{C} \beta_i w_i^C(k) x(k) \\
u(k) &= \sum_{r=1}^{C} \beta_i \left(-w_i^K(k) x(k) + w_i^R(k) y^d(k) + w_i^P(k) \left(y^d(k) - y(k) \right) \right)
\end{aligned}
\tag{3.42}
$$

From a block diagram point of view, identification process at each iteration k is repre-

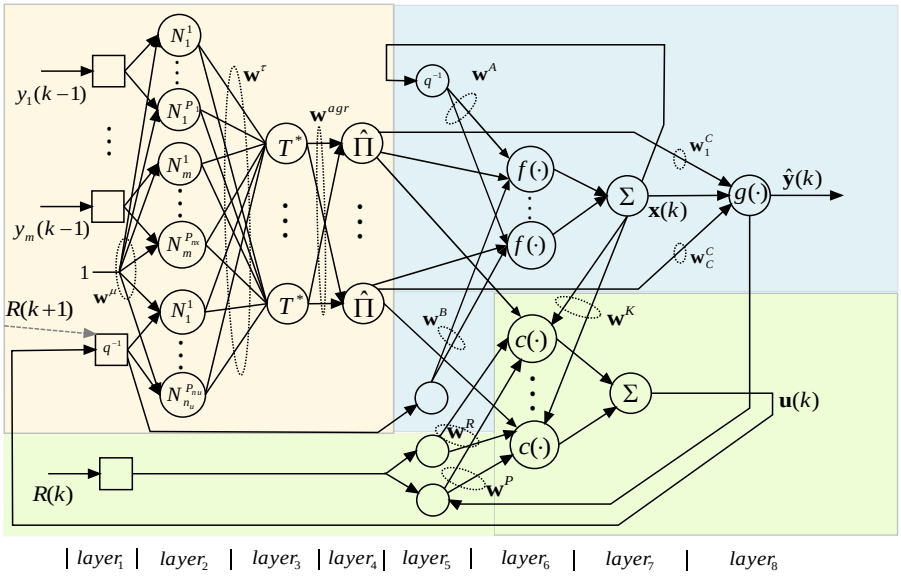


Figure 3.11: Proposed closed loop network diagram

Figure 3.12: Proposed closed loop architecture,$\hat{W}_k$ is the model parameters, $\hat{W}_k^K$ and $\hat{W}_k^R$ are controller parameters

sented in Figure 3.12, it is considered that a process output $y(k)$ due to control action $u(k-1)$ is already available. The algorithm sequence for process identification and control is described as:

1. At iteration k read the already available output sample from process sensors $y(k)$
2. Send next control action $u(k)$
3. At this time $\hat{y}(k)$ is also available since previous iteration and a new $\hat{y}(k+1)$ can be computed
4. Optimize previous estimated $\hat{x}(k-1)$ and $\hat{W}(k-1)$ to produce $\hat{x}(k)$ and $\hat{W}(k)$

5. Using optimized values of $\hat{x}(k)$ and $\hat{W}(k)$ optimize controller gains $W^K(k-1)$, $W^R(k-1)$ and $W^P(k-1)$

6. With all variables optimized compute next control action

$$u(k+1) = f\left(\hat{x}(k), \hat{W}(k), \hat{W}^{K,R,P}(k), y^d(k+1)\right)$$

Some authors i.e [36] stated that controller gains must be restricted to be $\hat{W}^k \geq 0$ in order to guarantee a negative state feedback for all states, meanwhile presented work will not enforce constraints on controller parameters. In comparison with what was done in [36], proposed controller contains an extra error gain allowing a faster controller response when changing reference set points.

3.5 Conclusion

This chapter has shown the convergence between different modelling architectures. Firstly it was presented the convergence of a standard fuzzy inference with a neuro network, which together became known in literature by Neuro Fuzzy System. One use case of RNF is the ANFIS structure which is widely used for the purpose of system identification. Next it was demonstrated the fusion of NFS with State Space theory. State space is one of the most known and studied theory with enormous applications including stable methods for identification and control, for instance the Kalman filter theory. Besides existence of Kalman based theory algorithms that does not require model under optimization to be in a state space formulation i.e UKF, it is always useful to have a state space approach being possible to make use of controlling theory based on system states feedback. Exploiting towards this direction, it was presented and proposed a new architecture that aims to converge state space and NFS into SSNFS. Having architectures defined, presented work proceeded with the identification of its structures and variables, identifying which parameters were suitable for optimization. It was defined that membership functions, membership degrees, rule truth degrees and local state space models weighting matrices were suitable for optimization. The next step towards a closed loop system, was the introduction of a controller to be applied to the plant. Making use of the Recursive Neuro Fuzzy modelling system (RNFMS) states, it was presented a controller based on a retro propagation of predicted states. Also if only an offline optimization is to be implemented, having a global controller might not work, since SSNFS dynamics are not be kept constant depending on process working points. Based on this assumption, it would be advantageous having a controller also dependent on process working points so, why not make use of RNFS model dynamics and apply it to a state space neuro fuzzy controller? Proposed Recursive Neuro Fuzzy Control System (RNFCS) used a MRAFC approach making use of model rule base knowledge for rule selecting and weighting

although, having its own local controller gains. Two different schemes were proposed regarding RNFCS premisses, one using the control actions other using the desired output values. Using the last approach a new complete RNFCS inference must be computed. Further theory regarding RNFMS and RNFCS stability, and the overall closed loop stability needs further study, since only empirical stabilization techniques and tips found in literature were presented.

4

Estimation Methods for Fuzzy Structures Parameters

4.1 Introduction

This chapter presents an overview regarding the State of the Art algorithms for real time identification. Several proposals already studied will be analysed and their usability will be inspected for the optimization of both MF weights and rules consequents. From two possible approaches i.e a Coupled Constrained Unscented Kalman Filter (**CCUCF**) and a Decoupled Constrained Unscented Kalman Filter (**DCUCF**), presented work will only follow the latest approach. Since in proposed RNFMS a difference rule subset might be considered at each iteration, the use of CCUCF have no major benefits. For real-time identification having a solution for the problem under optimization is time critical, it must satisfy process sampling time requirements. Therefore this chapter presents a study over most used on-line methods based on Kalman filter approach. Firstly it will start by formulating Kalman Filter theory [37] and proceed with its reformulation by means of an Unscented Transformation [38]. For a better understanding of Kalman concept it will be presented an introduction regarding probability and statistical properties of random variables RVs.

4.2 The Kalman Filter

4.2.1 Major Statistical Properties

The Kalman filter is a "State" estimator for a linear dynamic system described as a State-Space model perturbed by white noise, it exploits the statistical properties of Random Variables (RV) and Random Process (RP) (also called Stochastic process). Consider a RV X and x as a realization of X being a nonrandom independent variable or a constant, then
The probability distribution function (PDF) of X is

$$F_X(x) = P\left(X\left(t\right) \leq x\right)$$

The Probability density function (pdf) of X is

$$f_X(x) = \frac{dF_X\left(x\right)}{dx}$$

The moment of a random variable X is defined as:

$$\text{ith moment of } X = E(X^i)$$
$$\text{ith central moment of } X = E[(X - \overline{x})^i]$$

The first moment of X is known as mean $E[X]$ or for simplicity $\overline{x}$ and it can be computed as

$$\overline{x}(t) = \int_{-\infty}^{\infty} x f_X(x) dx$$

The variance σ_X^2 of a RV X is the second central moment of X and it can be computed as

$$\sigma_X^2(t) = \int_{-\infty}^{\infty} (x - \overline{x})^2 f_X(x) dx$$

The standard deviation of X is σ_x which is the square root of the variance. The third central moment is called skew and the forth central moment as Kurtosis. Now considering a $n-$element RV X and an m-element RV Y define:
Correlation of two column vector RV X and Y

$$\begin{aligned} R_{XY} &= E(XY)^T \\ &= \begin{bmatrix} E(X_1Y_1) & \cdots & E(X_1Y_m) \\ \vdots & \ddots & \vdots \\ E(X_nY_1) & \cdots & E(X_nY_m) \end{bmatrix} \end{aligned}$$

The covariance between X and Y

$$C_{XY} = E[(X - \overline{x})(Y - \overline{y})^T]$$

The autocorrelation of the vector RV X

$$\begin{aligned} R_X &= E(XX^T) \\ &= \begin{bmatrix} E(X_1^2) & \cdots & E(X_1 X_n) \\ \vdots & \ddots & \vdots \\ E(X_n X_1) & \cdots & E(X_n^2) \end{bmatrix} \end{aligned}$$

The autocovariance of the vector RV X

$$\begin{aligned} C_X &= E[(X - \overline{x})(X - \overline{x})^T] \\ &= \begin{bmatrix} E[(X_1 - \overline{x}_1)^2] & \cdots & E[(X_1 - \overline{x}_1)(X_n - \overline{x}_n)^T] \\ \vdots & \ddots & \vdots \\ E[(X_n - \overline{X_n})(X_1 - \overline{x}_1)^T] & \cdots & E[(X_n - \overline{x}_n)^2] \end{bmatrix} \\ &= \begin{bmatrix} \sigma_1^2 & \cdots & \sigma_{1n} \\ \vdots & \ddots & \vdots \\ \sigma_{n1} & \cdots & \sigma_n^2 \end{bmatrix} \end{aligned}$$

It is worth mentioning that both autocorrelation and auto-covariance matrix are always symmetric and positive semidefinite. A stochastic process is a random variable $X(t)$ that is time dependent, in this case the RV is continuous with time although time is discrete. For such cases, the stochastic process is defined as a continuous random sequence $X(k)$ $k = 1, 2, \ldots$. Note that $X(k_0)$ is itself a random variable being or not vector valued with $x(k_0)$ a realization of the RV $X(k_0)$. All statistical properties of a random variable can be applied to a stochastic process.

The process covariance $C_X(k)$ of $X(k)$ is defined as:

$$\begin{aligned} C_X(k) &= E\left\{[X(k) - \overline{x}(k)][X(k) - \overline{x}(k)]^T\right\} \\ &= \sum [x - \overline{x}(k)]\,[x - \overline{x}(k)]^T\, f(x, k) \end{aligned}$$

The autocorrelation of a stochastic process $X(k)$ where $X(k_1)$ and $X(k_2)$ are two distinct RVs:

$$\begin{aligned} R_X(k_1, k_2) &= E[X(k_1)X^T(k_2)] \\ &= \begin{bmatrix} E[X_1(k_1)X_1(k_2)] & \cdots & E[X_1(k_1)X_n(k_2)] \\ \vdots & \ddots & \vdots \\ E[X_n(k_1)X_1(k_2)] & \cdots & E[X_n(k_1)X_n(k_2)] \end{bmatrix} \end{aligned}$$

Define the autocovariance of a stochastic process as:

$$C_X(k_1,k_2) = E\left\{[X(k_1)-\overline{x}(k_1)][X(k_2)-\overline{x}(k_2)]^T\right\}$$
$$= \begin{bmatrix} E[(X_1(k_1)-\overline{x}_1(k_1))(X_1(k_2)-\overline{x}_1(k_2))^T] & \cdots & (X_1(k_1)-\overline{x}_1(k_1))(X_n(k_2)-\overline{x}_n(k_2))^T] \\ \vdots & \ddots & \vdots \\ E[(X_n(k_1)-\overline{x}_n(k_1))(X_1(k_2)-\overline{x}_1(k_2))^T] & \cdots & (X_n(k_1)-\overline{x}_n(k_1))(X_n(k_2)-\overline{x}_n(k_2))^T \end{bmatrix}$$

Considering two random processes $X(k)$ as an $n-$vector and $Y(k)$ as a $m-$vector it is defined correlation and cross-covariance as an $n \times m$ matrix[39]:

$$E[X(k_1)Y^T(k_2)], E\left\{[X(k_1)-\overline{x}(k_1)][Y(k_2)-\overline{y}(k_2)]\right\}$$

they are uncorrelated if:

$$E\left\{[X(k_1)-\overline{x}(k_2)][Y(k_1)-\overline{y}(k_2)]\right\} = 0,$$

and are orthogonal if their correlation matrix

$$E[X(k_1)Y^T(k_2)] = 0.$$

Finally, a random process X_k is called uncorrelated if

$$E\left\{[X(k_1)-\overline{x}_{k_1}][X_{k_2}-\overline{x}_{k_2}]^T\right\} = Q(k_1,k_2)\Delta(k_1-k_2),$$

where $\Delta(\cdot)$ is the Kronecker delta function defined as:

$$\Delta(k) = \begin{cases} 1 & \text{if } k=0 \\ 0 & \text{otherwise} \end{cases}$$

Although it was introduced the covariance definition according to [37], presented work will refer to the autocovariance C_X of a RV X or a stochastic process $X(k)$ as being simply covariance matrix, as it was done in [39] during kalman formulation.

4.2.2 Algorithm Formulation

The Kalman filter formulation was mainly derived from the RLS algorithm [37], although accounting for process noise and uncertainties. Consider the following linear stochastic system (4.1)

$$\begin{aligned} x_k &= F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + \eta_{k-1} \\ y_k &= H_k x_k + e_k \end{aligned} \tag{4.1}$$

Where η_k represents the process uncertainties and e_k the measurement noise, with covariances R_k^η and R_k^e respectively, both are uncorrelated and have a Gaussian Probability Distribution Function PDF such that:

$$\begin{aligned} \eta_k &\sim \left(0, R_k^\eta\right) \\ e_k &\sim (0, R_k) \\ E\left[\eta_k \eta_j^T\right] &= R_k^\eta \Delta(k-j) \\ E\left[e_k e_j^T\right] &= R_k \Delta(k-j) \\ E\left[e_k \eta_j^T\right] &= 0 \end{aligned} \tag{4.2}$$

Kalman filter algorithm can be used as:

$$\begin{aligned} \hat{x}_{k|k+N} &= E\left[x_k | y_1, y_2, ..., y_{k+N}\right] = \text{smoothed estimation} \\ \hat{x}_{k|k-M} &= E\left[x_k | y_1, y_2, ..., y_{k-M}\right] = \text{predicted estimation} \end{aligned}$$

and is composed by to distinct phases, a time update phase computing:

$$\begin{aligned} \hat{x}_k^- &= E\left[x_k | y_1, y_2, ..., y_{k-1}\right] = \textit{a priori} \text{ state estimate} \\ P_k^- &= E\left[(x_k - \hat{x}_k^-)(x_k - \hat{x}_k^-)^T\right] = \textit{a priori} \text{ estimation error covariance} \end{aligned}$$

and a measurement update phase computing:

$$\begin{aligned} \hat{x}_k^+ &= E\left[x_k | y_1, y_2, ..., y_k\right] = \textit{a posteriori} \text{ estimate} \\ P_k^+ &= E\left[(x_k - \hat{x}_k^+)(x_k - \hat{x}_k^+)^T\right] = \textit{a posteriori} \text{ estimation error covariance} \end{aligned}$$

This way the Discrete Kalman filter set of equations for the dynamic system (4.1) are as follows:

1. Initialization were $k = 0$:

$$\begin{aligned} \hat{x}_k^+ &= E(x_k) \\ P_k^+ &= \infty I \\ &= E[(x_k - x_k^+)(x_k - x_k^+)^T] \end{aligned} \tag{4.3}$$

2. For each time step $k = 1, 2, ...$:
   Time update equations

$$\begin{aligned} \hat{x}_k^- &= F_{k-1}\hat{x}_{k-1}^+ + G_{k-1}u_{k-1} \\ P_k^+ &= F_{k-1}P_{k-1}^+ + R_{k-1}^\eta \end{aligned} \tag{4.4}$$

Measurement update equations

$$\begin{aligned} K_k &= P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \\ \hat{x}_k^+ &= \hat{x}_k^- + K_k(y_k - H_k \hat{x}_k^-) \\ P_k^+ &= (I - K_k H_k) P_k^- \end{aligned} \tag{4.5}$$

The previous algorithm could be described as a predicted estimation problem with $M = 1$ for a stochastic process described as:

$$\begin{aligned} x_k &= F_{k-1} x_{k-1} + G_{k-1} u_{k-1} + \eta_{k-1} \\ y_{k-1} &= H_{k-1} x_{k-1} + e_{k-1} \end{aligned} \tag{4.6}$$

in this situation x_k represents a prediction of the system state having into account up to y_{k-1} output measurements. Meanwhile as it will be seen later this approach can not be done with UKF.

4.2.3 Divergence Phenomenon

Kalman filter algorithm may suffer from numerical stability problems due to computer arithmetic accuracy and modelling errors which are then propagated through states [37]. From Kalman filter gain K_k equation (4.5) it is necessary to take P_k^- squared root forcing P_k^- to be positive definite. This condition may not be achieved due to round off errors or if P_k is ill-conditioned. Another issue is the covariance matrix symmetry, which may or may not become symmetric. Somme approaches have been taken to overcome this issues:

- To force symmetry implement $P_k = (P_k + P_k^T)/2$
- For a positive definiteness problem use a square root filtering approach.

4.3 Unscented Kalman Filter

$$y = h(x) \tag{4.7}$$

From previous sections it can be seen that Kalman filter formulation was developed for a linear system model time or timeless dependent (4.1). Because most applications present a non-linear system model (4.7), other algorithms based on standard Kalman filtering theory have been proposed in literature. The first Kalman filter extension was the extended Kalman filter (EKF) which extended the use of Kalman filtering by linearising the non-linear system [40], in this work EKF approach was not followed, for more details see [41][42]. An earlier approach tries to approximate the system distribution rather than its non-linear functions by means of an Unscented Transformation (UT) [38][43] as it will be seen in the next sections.

4.3.1 Principle of Unscented Transformation

Unscented transformation uses a set of so called sigma points with mean $\hat{x}$ and covariance P_k, those sigma points are then spread through the non-linear system capturing its statistical properties. The new sigma points are then evaluated to compute the new mean and covariance. Several UKF enhancements were developed, dealing with algorithm degrees of freedom like the number of sigma points l to be used, the way noise is injected into system and constraints handling. From [38] a symmetric set of $2N+1$ sigma points were present, in [44] a set of $2N$ sigma points were used and from [43] not only a set of $N+1$ sigma points were presented but also an overview over other possibilities and how to exploit process higher order information. The scaled unscented transformation (SUT) to be used in this work, was presented in [45] and an overview of most relevant SUT methods was presented in [46]. Consider a N-dimensional random variable x_k, its mean $\hat{x}_k$ and covariance P_k with a set of sigma points and their corresponding weights $S_l = \{\chi_{k,l}, w_l; l = 1, ..., L\}$, the process to compute the initial symmetric scaled sigma points with $L = 2N+1$ is as follows:

$$\begin{aligned} \chi_0 &= \hat{x} \\ w_0^{(m)} &= \frac{\lambda}{N+\lambda} \qquad\qquad i = 0 \\ w_0^{(c)} &= \frac{\lambda}{N+\lambda} + (1-\alpha+\beta) \end{aligned}$$

$$\begin{aligned} \chi_i &= \hat{x} + (\sqrt{(N+\lambda)P_x})_i \\ w_i^{(m)} &= w_i^{(c)} = \frac{1}{2(N+\lambda)} \qquad i = 1, ..., N \end{aligned} \tag{4.8}$$

$$\begin{aligned} \chi_i &= \hat{x} - (\sqrt{(N+\lambda)P_x})_i \\ w_i^{(m)} &= w_i^{(c)} = \frac{1}{2(N+\lambda)} \qquad i = N+1, ..., 2N \end{aligned}$$

with

$$\lambda = \alpha^2(N+k) - N \tag{4.9}$$

After computing the initial set of sigma points they are then propagated through the non-linear system (4.7) in order to compute the new mean and covariances:

1. Spread each sigma point through the process model to get a set of transformed samples
$$Y_i = g(\chi_i) \quad i = 1, ..., L \tag{4.10}$$

2. Compute the predicted mean as
$$\hat{y}_k = \sum_{i=1}^{L} W_i^m Y_i \tag{4.11}$$

3. Compute the predicted covariance as

$$P_y = \sum_{i=1}^{L} W_i^c (Y_i - \hat{y})(Y_i - \hat{y})^T \tag{4.12}$$

4. the predicted cross-covariance

$$P_{xy} = \sum_{i=1}^{L} W_i^c (\chi_i - \hat{x})(Y_i - \hat{y})^T \tag{4.13}$$

As it can be seen the transformation contains three degrees of freedom k, α and β. The value of k affects the fourth and higher order moments of sigma points, it can be any number in order to have $N + k \neq 0$. Meanwhile, negative numbers of k may lead to a non-positive semi-definite covariance matrix when the system distribution is assumed Gaussian, a good value is $N + k = 3$ [38]. Regarding α it controls the spread of sigma points around $\hat{x}$ and should be set to a small value in range $0 \leq \alpha \leq 1$. The value of β should be $\beta = 2$ for Gaussian distributions, it adds a correction term to the covariance concerning fourth and higher order information [43].

4.3.2 Unscented Kalman Filter Formulation

Non-linear state space model for an additive noise case:

$$\begin{aligned} x_k &= f(x_{k-1}, u_{k-1}) + \eta_{k-1} \\ y_k &= h(x_k) + e_k \end{aligned} \tag{4.14}$$

Non-linear state space model for a correlated noise case:

$$\begin{aligned} x_k &= f(x_{k-1}, u_{k-1}, \eta_{k-1}) \\ y_k &= h(x_k, e_k) \end{aligned} \tag{4.15}$$

Non-linear state space model with states and parameters for an additive noise case:
For joint estimation (augmented states)

$$\begin{aligned} x_k &= f(x_{k-1}, u_{k-1}; w_{k-1}) + \eta_{k-1} \\ w_k &= w_{k-1} + r_{k-1} \\ y_k &= h(x_k; w_k) + e_k \end{aligned} \tag{4.16}$$

For dual estimation

$$\begin{aligned} \begin{bmatrix} x_k \\ w_k \end{bmatrix} &= \begin{bmatrix} f(x_{k-1}, u_{k-1}) \\ w_{k-1} \end{bmatrix} + \begin{bmatrix} \eta_{k-1} \\ r_{k-1} \end{bmatrix} \\ y_k &= h(x_k; w_k) + e_k \end{aligned} \tag{4.17}$$

Non-linear state space model with states and parameters for a correlated noise case (same approach as before for joint and dual formulation):

$$\begin{aligned} x_k &= f(x_{k-1}, u_{k-1}, \eta_{k-1}; w_{k-1}) \\ w_k &= w_{k-1} + r_{k-1} \\ y_k &= h(x_k, e_k; w_k) \end{aligned} \tag{4.18}$$

Non-linear state space model only with parameters for a correlated noise case:

$$\begin{aligned} w_k &= w_{k-1} + r_{k-1} \\ y_k &= h(x_k, e_k; w_k) \end{aligned} \tag{4.19}$$

Non-linear state space model only with parameters for an additive noise case:

$$\begin{aligned} w_k &= w_{k-1} + r_{k-1} \\ y_k &= h(x_k; w_k) + e_k \end{aligned} \tag{4.20}$$

The incorporation of Kalman filter theory into SUT was the next step towards recursive estimation, two variants were defined for two different stochastic systems. One concerns a system (4.15) with correlated noise [45] and other a system (4.14) with additive noise used in a case study [47], both algorithms are summarized in [48] and [46]. In the first case the sigma points χ^x are augmented with the noise variables to form $\chi^a = [\chi^x \ \chi^\eta \ \chi^e]$, concerning the second case which was adopted in this work, the noise is assumed purely additive. The algorithm is computed as in Table 4.1 where an additional computation of sigma points (4.22e-4.22f) was presented which according to [45] incorporates process noise effects in model outputs. While step (4.22e) augments the sigma points with process noise covariance for a total of $2L + 1$ sigma points, the respective weights need to be recalculated. Step (4.22f) does a completely recalculation of L sigma points, although information concerning odd-moments captured by initial propagated sigma points are rejected.

Instead of state variables some systems also include parameter variables Equation 4.16 as the case of neuro networks where the estimation of both parameters and states are needed [49]. There are two methods for recursively estimate states and parameters, a joint estimation and a dual estimation which applied to UKF algorithm becomes known as joint-UKF and dual-UKF [46]. The dual-UKF computation can be seen as a two stage algorithm were, in the first stage UKF algorithm is applied to parameters estimation and the second stage UKF algorithm is applied again for states estimation. The joint-UKF is a one stage process were the system states $\hat{x}$ is augmented with the parameters $\hat{x}^a = [\hat{x} \ \hat{w}]$. This work presents a fuzzy system with a state space representation according to Equation 4.20, and will consider algorithm from Table 4.2. The complete closed loop optimization solution will be implemented according to Figure 1.3. The parameters of RNFMS

Initialization: $\hat{x}_0 = E[x_0],\ P_0 = \alpha I$
For $k \in 1, ..., \infty$:

- compute the sigma points:

$$\chi^+_{k-1} = [\hat{x}^+_{k-1}\ \ \hat{x}^+_{k-1} + \gamma\sqrt{P^+_{k-1}}\ \ \hat{x}^+_{k-1} - \gamma\sqrt{P^+_{k-1}}] \tag{4.21a}$$

- Compute time-update steps:

$$\chi^{*-}_k = f(\chi^+_{k-1}, u_{k-1}) \tag{4.22a}$$
$$\hat{x}^-_k = \sum_{i=1}^{L} W^{(m)}_i \chi^{*-}_{i,k} \tag{4.22b}$$
$$P^-_k = \sum_{i=1}^{L} W^{(c)}_i (\chi^{*-}_{i,k} - \hat{x}^-_k)(\chi^{*-}_{i,k} - \hat{x}^-_k)^T + R^\eta_{k-1} \tag{4.22c}$$
$$\text{option 1: } \chi^-_k = \chi^{*-}_k \tag{4.22d}$$
$$\text{option 2: } \chi^-_k = [\chi^{*-}_k\ \ \chi^{*-}_{0,k} + \gamma\sqrt{R^\eta_{k-1}}\ \ \hat{x}^-_k - \gamma\sqrt{R^\eta_{k-1}}] \tag{4.22e}$$
$$\text{option 3: } \chi^-_k = [\hat{x}^-_k\ \ \hat{x}^-_k + \gamma\sqrt{P^-_k}\ \ \hat{x}^-_k - \gamma\sqrt{P^-_k}] \tag{4.22f}$$
$$Y^-_k = h(\chi^-_k) \tag{4.22g}$$
$$\hat{y}^-_k = \sum_{i=1}^{L} W^{(m)}_i Y^-_{i,k} \tag{4.22h}$$

- Compute the measurement-update equations:

$$P_{\tilde{y}_k\tilde{y}_k} = \sum_{i=1}^{L} W^c_i (Y^-_{i,k} - \hat{y}^-_k)(Y^-_{i,k} - \hat{y}^-_k)^T + R^e_k \tag{4.23a}$$
$$P_{x_k y_k} = \sum_{i=1}^{L} W^c_i (\chi^-_{i,k} - \hat{x}^-_k)(Y^-_{i,k} - \hat{y}^-_k)^T \tag{4.23b}$$
$$K_k = P_{x_k y_k} P^{-1}_{\tilde{y}_k\tilde{y}_k} \tag{4.23c}$$
$$\hat{x}^+_k = \hat{x}^-_k + K_k(y_k - \hat{y^-_k}) \tag{4.23d}$$
$$P^+_k = P^-_k + K_k P_{\tilde{y}_k\tilde{y}_k} K^T_k \tag{4.23e}$$

with $\gamma = \sqrt{N + \lambda}$, $L = 2N + 1$, N is the state dimension, R^η is the process noise covariance and R^e is the measurement noise.

Table 4.1: UKF state estimation for additive noise case (system as Equation 4.14)

Initialization:

$$\begin{aligned} \hat{w}_0 &= E[w_0] \\ P_0 &= \alpha I \end{aligned} \tag{4.24}$$

For $k \in 1, ..., \infty$:

- Compute time-update equations:

$$\hat{w}_k^- = \hat{w}_{k-1}^+ \tag{4.25a}$$
$$P_{w_k}^- = P_{w_{k-1}}^+ + R_{k-1}^r \tag{4.25b}$$

- Calculate sigma-points for measurement-update:

$$\chi_k^- = [\hat{w}_k^- \;\; \hat{w}_k^- + \gamma\sqrt{P_{w_k}^-} \;\; \hat{w}_k^- - \gamma\sqrt{P_{w_k}^-}] \tag{4.26}$$

- Compute the measurement-update equations:

$$Y_k^- = g(x_k, \chi_k^-) \tag{4.27a}$$
$$\text{option 1: } \hat{d}_k^- = \sum_{i=1}^{L} W_i^{(m)} Y_{i,k}^- \tag{4.27b}$$
$$\text{option 2: } \hat{d}_k^- = g(x_k, \hat{w}_k^-) \tag{4.27c}$$
$$P_{\tilde{d}_k} = \sum_{i=1}^{L} W_i^c (Y_{i,k}^- - \hat{d}_k^-)(Y_{i,k}^- - \hat{d}_k^-)^T + R_k^e \tag{4.27d}$$
$$P_{w_k d_k} = \sum_{i=1}^{L} W_i^c (\chi_{i,k}^- - \hat{w}_k^-)(Y_{i,k}^- - \hat{d}_k^-)^T \tag{4.27e}$$
$$K_k = P_{w_k d_k} P_{\tilde{d}_k}^{-1} \tag{4.27f}$$
$$\hat{w}_k^+ = \hat{w}_k^- + K_k(d_k - \hat{d}_k^-) \tag{4.27g}$$
$$P_{w_k}^+ = P_{w_k}^- + K_k P_{\tilde{d}_k} K_k^T \tag{4.27h}$$
$$\tag{4.27i}$$

with $\gamma = \sqrt{N + \lambda}$, $L = 2N + 1$, N is the parameters dimension, R^r is the process noise covariance and R^e is the measurement noise.

Table 4.2: UKF for parameter estimation considering additive noise case. (system as in Equation 4.14)

local state space models are only driven by the process noise r_{k-1}. For a $n-$dimensional system, parameters noise covariance R^r is a $n \times n$ matrix usually constrained to a diagonal matrix assuming no noise correlation between parameters. Thus, only noise variance information will be added to the covariance matrix P_{w_k} variance terms $E[(w_{i,k} - \hat{w}_{i,k})^2]$. Both output error covariance R^e and parameter error covariance R^r both for controller and model should be predicted, allowing a convergence speed improvement of the identification algorithm Table 4.2. Although a fast convergence of R^e towards zero can lead to overfitting and instability problems. The process to estimate R^r_{k-1} can be done by three techniques [46]:

- Set to a constant value of
$$R^r_0 = \alpha_{R^r} I \tag{4.28}$$
- Use as in RLS algorithm a forgetting factor $\lambda_{RLS} \in (0, 1]$:
$$R^{r+}_k = diag(\lambda^{-1}_{RLS} P^+_{w_k}) \tag{4.29}$$
From above equation we can see that old data information will be incorporated on next iteration being the amount of concerned past data proportional to λ_{RLS}
- Another possibility is estimating the innovations using a Robbins-Monro stochastic approach:
$$R^{r+}_k = diag([1 - \alpha_r]\, R^{r+}_{k-1} + \alpha_{RM} K_k [d_k - \tilde{y}_k][d_k - \tilde{y}_k]' K'_k) \tag{4.30}$$
This method allows a fastest and more stable convergence when compared with previous methods as it will be seen during simulation results.

Regarding the prediction of output noise covariances, no explicit formulation was found in literature although, a new method was proposed an formulation can be found in attached algorithms section A.1.

As in Kalman filter, the major UKF drawback is the computation of covariance matrix square root $\sqrt{P_{w_k}}$ during sigma point calculation, such computation can be affected by divergence phenomenons due to round off errors. A more reliable procedure is making use of factorization methods to compute $\sqrt{P_{w_k}}$, for instance a popular algorithm referenced in literature is known as Cholesky factorization [37]. This method computes a unique unit lower triangular matrix S such that $P_{w_k} = SS^T$, S will always be symmetric and definite due to physical principles meanwhile, P_{w_k} must also be positive definite. A different procedure other than Cholesky factorization, is to perform a LDL factorization [50] which can be achieved if $P_{w_k} = LdL^T$ is symmetric and nonsingular. The LDL factorization also referred as UDU can also be written as $P_{w_k} = L\sqrt{D}\sqrt{D}L^T = SS^T$ with D a diagonal matrix, L a unit lower triangular matrix and L^T its complex conjugate, if all elements of D are positive then P_{w_k} is positive definite. Based on above factorization

methods other extensions to Kalman filter were developed and became known as square root filtering and U-D filtering [37]. The extension of such methods to UKF algorithm became known as SR-UKF [51] and UD-UKF [52].

4.3.3 Constrained Unscented Kalman Filter

Optimization takes an important role in system identification, not only because a complete set of input-output data may not be available but also due to process internal states variations along time. While in white box identification it might be observed that some process states must be confined to a predefined range, in black box modelling that conclusion may not be obvious since no previous knowledge is available. The need to consider a constrained optimization in black box modelling may arise from model stability conditions or from model architecture itself as the case of NFS where some intrinsic characteristics restricts certain parameters to a subset of values as mentioned in previous chapters. A variable x can be constrained by a system of linear equalities or inequalities (4.31) or by a non-linear system of equalities or inequalities. This section will present some solutions for the problem of constrained optimization using UKF when linear inequalities are considered although, before presenting them, it is worth mentioning an approach described in [30] where instead of optimizing a bounded constrained variable $0 \leq x \leq 1$, it was optimized an unconstrained variable $\alpha \in \Re$ such that $x = \frac{tanh(\alpha)+1}{2}$. Another approach is to use clipping methods [53] where outbound values are projected onto boundaries of a feasible region. This method can also be applied to UKF as described in [54]. The clipping approach in UKF [55] can lead to a sigma point symmetry lost around mean $\hat{x}^+_{k-1}$, being difficult to capture state and parameters weighted mean $\hat{x}^+_{k-1}$ and covariance P^+_{k-1}. A general overview of constraints handling in UKF was presented in [56] which besides describing seven different methods, not all are suitable for linear inequality constraints.

$$Ax \leq B \tag{4.31a}$$

$$lb \leq x \leq ub \tag{4.31b}$$

$$A_{eq}x = B_{eq} \tag{4.31c}$$

From all steps defined in algorithms Table 4.1 and Table 4.2, some of are suitable to handle constraints. Starting with the calculation of sigma points, a new method similar to clipping, although modified for UT use, was introduced in [57]. Author presented an interval constrained unscented transformation (ICUT) Table 4.3 where posteriori sigma points are projected onto a constrained surface by modifying their weights in an asymmetrical fashion such that each sigma point verifies (4.31b). The extension of ICUT to UKF is described in [56] as interval unscented Kalman filter (IUKF). Besides constraining sigma points, it is not guaranteed that time update states are kept inside a constrained subspace. To accomplish this request another set of algorithms were proposed which extends the ICUT. An unscented recursive nonlinear dynamic data reconciliation (URNDDR) was presented in

To generate constrained sigma points $\chi \in [lb, ub]$ calculate sigma steps $\theta\,(k)^{[2N\times 1]}$:

- For any instance k compute:

$$\begin{aligned}\overline{\chi}^{-}\,(k) = \big[&\hat{x}^{+}\,(k-1) \\ &\hat{x}^{+}\,\{k-1\}\,\theta_1\,(k-1)\,S_{:,1} \;\; \cdots \;\; \hat{x}^{+}\,\{k-1\}\,\theta_n\,(k-1)\,S_{:,n} \\ &-\hat{x}^{+}\,\{k-1\}\,\theta_{n+1}\,(k-1)\,S_{:,n+1} \;\; \cdots \;\; \theta_{2n}\,(k-1)\,S_{:,2n}\big]\end{aligned} \tag{4.32}$$

- For $i = 1, \ldots, N$ and $j = 1, \ldots, 2N$ compute sigma points distances

$$S = \left[\sqrt{P_x^{+}\,(k-1)} \;\; -\sqrt{P_x^{+}\,(k-1)}\right] \tag{4.33a}$$

$$\theta_j \triangleq min\,(\Theta_{:,j}) \tag{4.33b}$$

$$\Theta_{i,j} \triangleq \begin{cases} \sqrt{N+\lambda}, & \text{if } S_{i,j} = 0, \\ min\left(\sqrt{N+\lambda}, \frac{ub_i(k-1)-\hat{x}_i^{+}(k-1)}{S_{i,j}}\right), & \text{if } S_{i,j} \geq 0, \\ min\left(\sqrt{N+\lambda}, \frac{lb_i(k-1)-\hat{x}_i^{+}(k-1)}{S_{i,j}}\right), & \text{if } S_{i,j} \leq 0, \end{cases} \tag{4.33c}$$

- Compute also respective weights for $j = 1, \ldots, 2N$:

$$\gamma_0 \triangleq b, \;\; \gamma_j \triangleq a\theta_j + b \tag{4.34}$$

satisfying $\sum_{j=0}^{2N} \gamma_j = 1$, with

$$a \triangleq \frac{2\lambda - 1}{2\,(N+\gamma)\left(\sum_{j=1}\theta_j - (2N+1)\sqrt{N+\lambda}\right)} \tag{4.35a}$$

$$b \triangleq \frac{1}{2\,(N+\lambda)} - \frac{2\lambda - 1}{2\sqrt{N+\lambda}\left(\sum_{j=1}\theta_j - (2N+1)\sqrt{N+\lambda}\right)} \tag{4.35b}$$

Table 4.3: Formulation of ICUT

[57], where an optimization problem was computed for each sigma point. Such method is also known as sigma-point interval unscented Kalman filter (SIUKF) as observed by [56] where constraints were enforced in step (4.22a). A simplified version of SIUKF is known as constrained interval unscented Kalman filter (CIUKF) [56] where also an optimization problem is computed although, instead of being computed for each sigma point, it is achieved by enforcing constraints in steps (4.23d). Other UKF algorithms can be combined with ICUT where the solution of an optimization problem is not required to enforce constraints (4.31b). One method is to apply a pdf truncation as proposed in standard Kalman filter formulation [58], although applied in UKF the algorithm is known as truncated UKF (TUKF), which in combination with ICUT becomes truncated interval UKF (TIUKF) [56]. Instead of truncation, a projection of unconstrained estimates onto constrained surface such that constraints are fulfilled, can also be applied. This method is known as estimate projection [58] which applied to UKF becomes PUKF [56]. Projection is done by computing an optimization problem in step (4.23d), although with a simplified cost function which can be derived by the maximum probability method or by the mean square method as described in [59].

4.3.4 UKF Application Results from a Theoretical Model

Next will be illustrated the use of Unscented Kalman filter applied to a theoretical model used as reference by other authors more precisely in [46]. For a better understanding of UKF dynamics, it will be considered the error covariance ellipsoid which is computed according to [60]:

$$[x - m_x]^T \sum_X^{-1} [x - m_x] = K \tag{4.36}$$

Equation (4.36) is a Mahalanobis distance of vector x to the mean m_x and $\sum_X^{-1}$ is an error covariance matrix. Considering $K = 1$, the plot algorithm will be similar to proposed in[61]. For instance, consider the discrete non linear model "2-state CSTR" described as [46] with system experiencing Gaussian Noise $r_k \sim N(0, R_k^r)$ $e_k \sim N(0, R_k^e)$:

$$\begin{aligned} x_{1,k+1} &= \frac{x_{1,k}}{1 + 2k_r\Delta_t x_{1,k}} + r_{1,k} \\ x_{2,k+1} &= x_{2,k} + \frac{k_r\Delta_t x_{1,k}}{1 + 2k_r\Delta_t x_{1,k}} + r_{2,k} \\ y_k &= [1\ 1]\, x_k + e_k \end{aligned} \tag{4.37}$$

considering initial values:

$$\begin{aligned} &\Delta_t = 0.1 \quad P_0 = \begin{bmatrix} 6^2 & 0 \\ 0 & 6^2 \end{bmatrix} \quad R_k^r = \begin{bmatrix} 0.001^2 & 0 \\ 0 & 0.001^2 \end{bmatrix} \\ &R_k^e = 0.1^2 \qquad x_0 = \begin{bmatrix} 3 \\ 1 \end{bmatrix} \qquad \hat{x}_0 = \begin{bmatrix} 0.1 \\ 4.5 \end{bmatrix} \end{aligned} \tag{4.38}$$

and using algorithm parameters (4.9) $\alpha = 1$; $\beta = 2$; $K = 0$. The UKF response will be analysed using the following extensions:

- Standard UKF formulation;
- UKF with clipping on sigma points (PUKF without optimization problem) (4.26) and on predicted states, similar to Table A.5 but with clipping on sigma points (4.27)g;
- UKF with constrained interval constrained unscented transformation (CIUKF) as in Table 4.3 with a clipping on predicted states (4.27)g without solving optimization problem, see proposed Table A.2.

Also the effect of lowering parameter α will be analysed. From Figure B.1 it can be seen that for 100 samples the states slowly converge to the real states, this is due to the existence of some spread sigma points outside allowable region since system negative states (i.e pressure) are non-physical.Considering constraints in algorithm it is observed from Figure B.2 and Figure B.3 that state convergence is faster and notice an improvement using CIUKF. Also error covariance decreases rapidly in comparison with the unconstrained version, meaning a faster convergence. Concerning a lower value for α it can be noticed that convergence becomes compromised in constrained version. Besides smaller values of α resulting in a lower spread of sigma points around mean, the sigma points weights will become much more weighing, increasing the effect of neither PUKF nor CIUKF capture true mean and covariance if sigma points are outside the feasible region. The algorithm can even become unstable resulting in a non definite positive predicted error covariance. For the unconstrained case, lowering α will have no major impact, this way it is recommended the use of $\alpha \approx 1$ in order to decrease the weighing ω_0^m on χ_k^{*-}, lowering the effect of a wrong state mean prediction. The presented theoretical model has a known error distribution, although this is not the case when using a stochastic process where noise is assumed to be Gaussian with unknown covariance $r_k \sim N(0, R_k^r)$ $e_k \sim N(0, R_k^e)$. Based on this assumption noise covariances must be predicted at each iteration and an initial value must be chosen such that it allows system convergence. Next will be presented the effect on convergence when predicting those variables. The system to be used will be the same theoretical model, although it is assumed an unknown error variance. The following set of tests will be conducted using $\alpha = 0.9$, CIUKF formulation and for initial values $R_0^r = 0.1$ and $R_0^e = 1$:

- Forgetting factor $\lambda_r = 0$ for $(0, R_k^r)$ and $\lambda_e = 0$ for $(0, R_k^e)$
- Forgetting factor $\lambda_r = 0$ for $(0, R_k^r)$ and $\lambda_e > 0$ for $(0, R_k^e)$
- Forgetting factor $\lambda_r > 0$ for $(0, R_k^r)$ and $\lambda_e = 0$ for $(0, R_k^e)$
- Forgetting factor $\lambda_r > 0$ for $(0, R_k^r)$ and $\lambda_e > 0$ for $(0, R_k^e)$

From results it can be observed that system states will not converge if considering forgetting factors $\lambda_r = 0$, $\lambda_e = 0$ and a high value for R_k^{r+}, independently of chosen value for R_k^{e+} as shown by Figure B.7 and Figure B.8. Also adopting a small value of $R_k^{r+} \approx 0$ will result in a fast decrease of error covariance P_k^+, increasing the probability of convergence to a local minimum. Concerning output error covariance, the higher its value the lower the convergence speed, working as a gain smoother. Using $\lambda_e = 0$ and a low initial value might lead system to a local solution or even divergence and instability. Initial values for states and output error covariances become more insignificant with the increasing of forgetting factor values, as they will converge to computed innovations (4.30). For instance, it can be observed a convergence improvement using $\lambda_r = 0.2$ and $\lambda_e = 0.2$ as shown in Figure B.9 and Figure B.10, although from last figure state prediction becomes affected by system noise if using a higher forgetting factor λ_e.

So far it has been presented results for the case of state estimation, meanwhile parameter estimation has also interesting dynamics which will be shown in next chapter using a real stochastic process.

4.4 Proposed Algorithm for On-line Estimation of RNFMS and RNFCM

4.4.1 RNFMS Variable Design

The proposed algorithm for optimization of proposed RNFS model is based on UKF algorithm, where both membership functions and consequent parameters will be optimized reducing the error between predicted output and measured plant output. Before introducing algorithm computation, it is worth presenting a matrix formulation of adopted RNFS. Consider a state vector $x(k) = [x_1(k)\, x_2(k) \cdots x_{n_x}(k)]$, control vector $u(k) = [u_1(k)\, u_2 k \cdots u_{n_u}(k)]$ and an output vector $y(k) = [y_1(k)\, y_2(k) \cdots y_m(k)]$ for a MIMO system. The system inputs will be constructed based on a regression vector $\varphi = [y(k-1)\, u(k-1)] \in \Re^{1\times n}$. Each input φ_i is denoted with P_i membership functions each being parametrized by a vector $w_{i,j}^{\mu}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, P_i$, it is consider that all MFs parametrization vector will have the same dimension, i.e $w_{i,j}^{\mu} \in \Re^{1\times p_{max}}$. Having a total number of rules C according to (3.24) it is possible to create the following matrices:

$$M^{\xi} \in \{0,1\}^{n\times P_{max}\times C}$$
$$\text{with} \tag{4.39}$$
$$P_{max} = \max_{i=1}^{n} P_i$$

which represents a masking matrix reflecting the premises layout, this matrix is constant for all iterations and only one element equal to one is allowed per row. Also a matrix

containing all membership degrees for all inputs at each iteration:

$$M^{\sigma}(k) \in [0,1]^{n \times P_{max}}$$

$$M^{\sigma}(k) = \begin{bmatrix} \mu_1^1(\varphi(1), w_{1,1}^{\mu}) & \mu_1^2(\varphi(1), w_{1,2}^{\mu}) & \cdots & \mu_1^{P_{max}}(\varphi(1), w_{1,P_{max}}^{\mu}) \\ \mu_2^2(\varphi(2), w_{2,2}^{\mu}) & \mu_2^2(\varphi(2), w_{2,2}^{\mu}) & \cdots & \mu_2^{P_{max}}(\varphi(2), w_{2,P_{max}}^{\mu}) \\ & & \vdots & \\ \mu_n^1(\varphi(n), w_{n,2}^{\mu}) & \mu_n^2(\varphi(n), w_{n,2}^{\mu}) & \cdots & \mu_n^{P_{max}}(\varphi(n), w_{n,P_{max}}^{\mu}) \end{bmatrix} \tag{4.40}$$

if $P_i < P_{max}$ set $\mu_i^{P_j}(\varphi(i), w_{i,P_j}^{\mu}) = 0 \; \forall \; j = P_i, P_i + 1, \ldots, P_{max}$. Consider also a weighting matrix which contains input weights for all rules:

$$w^{\tau}(k) \in [0,1]^{n \times C} \tag{4.41}$$

a weighting matrix that contains rule weights:

$$w^{agr}(k) \in [0,1]^{1 \times C} \tag{4.42}$$

and the consequents parameters matrices:

$$\begin{aligned} w^A(k) &\in \Re^{n_x \times n_x \times C} \\ w^B(k) &\in \Re^{n_x \times n_u \times C} \\ w^C(k) &\in \Re^{m \times n_x} \end{aligned} \tag{4.43}$$

then the inference process can be computed according to described neuron functions in section subsection 3.4.1:

1) $M1_{i,:} = 1 - diag(M_{:,:,i}^{\xi} M^{\sigma'}(k))$

2) $M2_i = \prod \left(1 - w_{:,i}^{\tau}(k) M1_{i,:}\right)$

3) $M3_i = M2_i w_i^{agr}(k)$

4) $M4_i = \frac{M3_i}{\sum(M3)}$

5) $M5_{:,i} = w_{:,:,i}^{A}(k)x(k) + w^{B}:,:,i(k)u(k)$

6) $M6_{:,i} = M4_i M5_{:,i}$

7) $x(k+1) = \sum M6_{:,i}$

8) $y(k+1) = \sum M4_i w_{:,:,i}^{C}(k)x(k+1)$

or in a compact form:

$$\begin{aligned} x_{k+1} &= \frac{\sum_{i=1}^{C} \prod \left(1 - w^{\tau}_{:,i;k} \left(1 - diag(M^{\xi}_{:,:,i} M^{\sigma'}_{k})\right)\right) \left(w^{A}_{:,:,i;k} x_k + w^{B}_{:,:,i;k} u_k\right) w^{agr}_{i;k}}{\sum_{i=1}^{C} \prod \left(1 - w^{\tau}_{:,i;k} \left(1 - diag(M^{\xi}_{:,:,i} M^{\sigma'}_{k})\right)\right) w^{agr}_{i;k}} \\ y_{k+1} &= \frac{\sum_{i=1}^{C} \prod \left(1 - w^{\tau}_{:,i;k} \left(1 - diag(M^{\xi}_{:,:,i} M^{\sigma'}_{k})\right)\right) w^{C}_{:,:,i;k} x_{k+1} w^{agr}_{i;k}}{\sum_{i=1}^{C} \prod \left(1 - w^{\tau}_{:,i;k} \left(1 - diag(M^{\xi}_{:,:,i} M^{\sigma'}_{k})\right)\right) w^{agr}_{i;k}} \end{aligned} \tag{4.44}$$

The previous description referred to a complete inference analysis for each iteration, meanwhile it is possible to evaluate only a subset of rules. One possibility is to discard rules with firing strength lower than some predefined value (α_{rule}) although, this strategy creates a varying subset dimension of rules for $\alpha > 0$. Other strategy was followed by adopting normal fuzzy sets with membership functions verifying

$$\sum_{j=1}^{P_i} \mu_{A^j}(x_i) = 1 \tag{4.45}$$

meaning that if a fuzzy variable full belongs to a normal set, it does not belong to any other set. This restriction gives the possibility to find at each iteration a rule subset with constant dimension $C^{sub} = 2^n$, meanwhile this restriction also imposes neighbour MFs side symmetric. From a complete matrix M^{ξ} it is possible to find a subset of it by:

1. construct M^{ξ} starting from first input to last input, e.g

$$M^{\xi}_{:,:,1} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \; M^{\xi}_{:,:,2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \; \cdots \; M^{\xi}_{:,:,9} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.46}$$

2. for all inputs the most left sided MF has index 1

3. at each iteration create a matrix $M^{\mu} \in \{1, 2, \cdots\}^{n \times 2}$ containing for each input the two membership functions index whose degrees are not zero, if a membership function as degree one, take a criteria to select the second, e.g select the second which as the closest center to the first.

4. find the firing rules indexes i^{μ} that includes the above two membership functions for each input and create the rule subset masking matrix $\overline{M}^{\xi} = M^{\xi}_{:,:,i^{\mu}}$

5. create a reduced membership matrix $\overline{M}^{\sigma} = M^{\sigma}_{i,M^{\mu}_{i,:}}$

6. compute reduced inference following the same procedure as in Equation 4.4.3

This way, for five inputs (including back propagated variables) each having five membership functions, instead of evaluating 3125 rules for each iteration it will be evaluated

32, making a large system feasible. The main drawbacks using these technique during the UKF optimization procedure is the need to use a variable covariance matrix. Since estimated variables indices may vary at each iteration, it is necessary to build a new covariance matrix $\overline{P}_k$ during each algorithm cycle (will be seen in the next sections). Because for each instance k a distinct subset of rules are selected, the previous obtained covariance matrix $\overline{P}^+_{k-1}$ might not reflect variables variance and their correlation if $i^\mu_{k-1} \neq i^\mu_k$ leading to system divergence. In order to maintain system stability it is proceeded as follows:

1. consider the existence of a local covariance matrix $\overline{P}^+_{k-1} \in \Re^{\overline{n}_k \times \overline{n}_k}$ used to compute UKF algorithm, where $\overline{n}_k$ is the number of variables to estimate for a rule subset;
2. consider the existence of a global covariance matrix $P^+_{k-1} \in \Re^{n_k \times C}$ where n_k is the total number of variables for all C rules. This covariance matrix does not have to be square since only variable variance values will be considered;
3. for each iteration k if $i^\mu_{k-1} \neq i^\mu_k$, $\overline{P}^+_{k-1}$ is a diagonal matrix whose elements are obtained from P^+_{k-1} considering only parameters for current instance rule subset;
4. for each iteration k, P^+_k elements are updated from $\overline{P}^+_k$ diagonal considering only parameters for current instance rule subset.

From previous steps it can be concluded that $\overline{P}^+_{k-1}$ is a diagonal matrix every time a rule subset changes, without correlated values between variables. This approach results from the fact that setting $\overline{P}^+_{k-1}$ cross-covariance values (non diagonal values) from a possible stored global matrix may provide algorithm with a wrong correlation information. Also $\overline{P}^+_{k-1}$ may become non-definite positive. The approach of a diagonal covariance matrix was also presented in [30] where diagonal blocks were assumed to be dominant. Using these scheme not only system complexity was reduced by $\frac{1}{2}$, but also rejecting cross-correlated values lead to an assumption of uncorrelated variables, diminishing algorithm convergence speed. Besides considering only covariance diagonal values if rules subset changes, a full covariance matrix $\overline{P}^+_{k-1}$ is kept constant if subset is also kept constant. During this iterations, correlated values between all subset parameters will be considered. Meanwhile, during implementation it was found that correlating output matrix ω^C parameters with matrices ω^A and ω^B lead to a non convergence when using an excessive number of states. This effect can be reduced when computing $\overline{P}^+_k$ if no correlation is considered between output matrix parameters and other weighting matrices i.e ω^A and ω^B. A minimum number of states which resulted in an acceptable predicted error will be considered during implementation.

4.4.2 RNFMS Constrained Variable Handling

RNFS parameters concerns not only rules premise variables which involves membership parameters w^μ, input weights w^τ and rule weights w^{agr} but also consequent parameters which incorporates local state space models parameters w^A, w^B and w^C. Membership

parameters depend on the type of adopted shaping functions and will consider restriction of equation (4.45). It is recommended the use of sigmoidal (2.49), s-shaped (2.48), z-shaped (2.50) and pi-shaped due to its continuous Gaussian shape. Other first order MFs like trapezoidal or triangular could be preferable due to its simplicity although, discontinuities might compromise solution results. In order to verify imposed restriction (4.45), MFs parameters must also be accomplished with some constraints. Take as example the use of s(z)-shaped functions which in order to cover the universe of discourse, a z-shaped must be the first MF mf_1, s-shaped the last MF mf_{P_i} and pi-shaped MFs mf_i in between, such that right side of mf_{i-1} is symmetric to left side mf_i. This approach also diminishes the number of parameters under optimization being necessary to consider only two parameters for each input. For instance consider three MFs:

$$\begin{aligned} z_{MF} &= f_1(x, a_1, b_1) \subseteq \mathbf{U} \in [lb, ub] \\ \Pi_{mf} &= f_2(x, a_2, b_2, c_2, d_2) \subseteq \mathbf{U} \in [lb, ub] \\ s_{mf} &= f_3(x, a_3, b_3) \subseteq \mathbf{U} \in [lb, ub] \\ &\text{with} \\ a_1 &= a_2 \\ b_1 &= b_2 \\ c_2 &= a_3 \\ d_2 &= b_3 \end{aligned} \tag{4.47}$$

parameter optimization must be constrained to a range which can be represented by the system:

$$Ax \leq b \quad \text{with} \quad x = \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \quad A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \\ 1 & -1 \end{bmatrix} \quad b = \begin{bmatrix} cr_2 - \delta \\ -cr_1 - \delta \\ -\delta \end{bmatrix} \tag{4.48}$$

where cr_i is an extra parameter fixed along time representing the centre of i^{th} MF and $\delta > 0$ is a user defined constant value which controls the MF maximum slope. Another possibility is to consider just an upper and lower bound for parameters such that $a_1 \leq b_1 - \delta$, it is proposed:

$$\begin{aligned} cr_1 + \delta \leq & a_1 \leq \frac{|cr_1 - cr_2|}{2} - \delta \\ \frac{|cr_1 - cr_2|}{2} + \delta \leq & b_1 \leq cr_2 - \delta \end{aligned} \tag{4.49}$$

previous set of constraints simplifies algorithm, being possible to solve system using interval constrained methods as described in subsection 4.3.3. Concerning parameters w^{τ}, w^{agr} they are constrained to subset $[0, 1] \in \Re$. For each iteration the number of antecedent variables parameters regarding w^{μ}, w^{τ} and w^{agr} are $2 \times 2^n + n \times 2^n + 2^n = (3 + n)2^n$.

The optimization procedure will be based on algorithm Table 4.2 using a CIUKF, i.e using ICUT for sigma points and a clipping of measured weights as described in section subsection 4.3.4. Other approach like the projection of $\overline{\omega}_k^+$ into an unconstrained surface using function:

$$\omega_k^+ = \text{arctanh}\left(\frac{2\overline{\omega}_k^+ - ub - lb}{ub - lb}\right) \quad \forall\, ub > lb \;\wedge\; lb < lb + \epsilon \leq \overline{\omega}_k^+ \leq ub - \epsilon < ub \tag{4.50}$$

and then applying UKF algorithm to the unconstrained ω_k^+ could also be used. This method will be named as Unconstrained Projected UKF (UPUKF). Regarding consequent parameters w^A, w^B and w^C no restrictions are imposed. State estimation by itself can be constrained or not depending on adopted RNFS topology, for the case were output layer is used to feedback the input layer Figure 3.9 and Figure 3.10, no restrictions are imposed over state estimation. On the other hand if feedback is taken from hidden layer seven Figure 3.8 a restriction on states domain must be considered, the constraints to be imposed are inequality constraints with lower and upper bounds depending on fuzzy state variables $\boldsymbol{x}(x)$ universe of discourse $\boldsymbol{U}_x$ over which respective fuzzy sets A_x are defined. The constrained problem solution is equal to the previously presented regarding premise parameters, i.e an UPUKF or CIUKF approach can also be taken.

4.4.3 RNFCS Variable Design

Regarding controller variable mapping, from proposed architecture Figure 3.12 it can be observed that model rules premisses will be used also for controller if control actions are applied in premisses. Otherwise, a similar antecedent variable mapping must be used and a new inference needs to be computed. In order to proceed with controller optimization, local controllers parameters need to be defined. From (3.41) define:

$$\begin{aligned} \omega^K(k) &\in \Re^{n_u \times n_x \times C} \\ \omega^R(k) &\in \Re^{n_u \times m \times C} \\ \omega^P(k) &\in \Re^{n_u \times m \times C} \end{aligned} \tag{4.51}$$

Having defined ω^τ, ω^{agr}, M^ξ, M^σ, the inference process can be defined as:

1) $M1_{i,:} = 1 - diag(M^{\xi}_{:,:,i} M^{\sigma'}(k))$

2) $M2_i = \prod \left(1 - w^{\tau}_{:,i}(k) M1_{i,:}\right)$

3) $M3_i = M2_i w_i^{agr}(k)$

4) $M4_i = \frac{M3_i}{\sum(M3)}$

5) $M5c_{:,i} = w^R_{:,:,i}(k) y^d(k) + w^P_{:,:,i}(k)\left(y^d(k) - M8\right)$

6) $M6c_{:,i} = M4_i \left(w^K_{:,:,i}(k) x(k+1) + M5c_{:,i}\right)$

7) $M7c = \sum M6c_{:,i}$

8) $u(k+1) = M7c$

or in a compact form:

$$u_{k+1} = \frac{\sum_{i=1}^{C} \left[\prod \left(1 - w^{\tau}_{:,i;k} \left(1 - diag(M^{\xi}_{:,:,i} M^{\sigma'}_{k}) \right) \right) \right.}{\cdots} \cdots \\ \cdots \frac{\left. \left(w^{K}_{:,:,i;k} x_{k+1} + w^{R}_{:,:,i;k} y^{d}_{k} + w^{P}_{:,:,i;k} \left(y^{d}_{k} - \hat{y}_{k} \right) \right) w^{agr}_{i,k} \right]}{\sum_{i=1}^{C} \prod \left(1 - w^{\tau}_{:,i;k} \left(1 - diag(M^{\xi}_{:,:,i} M^{\sigma'}_{k}) \right) \right) w^{agr}_{i;k}} \tag{4.52}$$

One disadvantage of premise sharing between controller and model consist on the fact that model MF optimization will use a cost function based on output error, meaning a MF optimization concerning model parameters but not accounting for controller dynamics. This will result in optimized MFs only regarding model, leading to a suboptimal controlling solution. Controller proportional error terms, avoids during UKF optimization a parameter divergence, since controller weights innovations proportionally depend on error between reference and model outputs. It should be highlighted that a model divergence will also result in a controller divergence.

4.4.4 Decoupled RNFS Estimation using UKF

This section handles the estimation procedure of the previously described RNFS variables both for model and controller. Decoupled estimation is divided into three distinct identification processes i.e model parameters, states and controller parameters Figure 1.3. Besides identification separation between model parameters and states, other separations can be done regarding parameters. If covariance matrix P_k both for model and controller is considered to be diagonal, no correlation between stochastic variables will be considered, being indifferent any possible parameter separation. Thus, independent procedures can be taken to estimate w^{μ}, w^{τ} w^{agr}, w^{A}, w^{B}, w^{C}, w^{K}, w^{P} and w^{R}. Proposed decoupled UKF solution will only consider architecture as in Figure 3.9 and use a consequent parameter correlation defined by

$$P_{\psi;i;k} = diag\left(P^{A}_{\psi;i;k}, P^{B}_{\psi;i;k}, P^{C}_{\psi;i;k}\right)$$

where i is the sub-rule index and,

$$P^{A}_{\psi,1\cdots i;k} \in \Re^{n_x \times n_x \times C^{sub}}$$

$$P^{B}_{\psi,1\cdots i;k} \in \Re^{n_x \times n_u \times C^{sub}}$$

$$P^{C}_{\psi,1\cdots i;k} \in \Re^{m \times n_x \times C^{sub}}$$

this means that no correlation between rules are considered and matrices $W^A,\ W^B,\ W^C$ for a given local model are also not correlated between them. For controller a different correlation scheme was adopted, for a given rule subset in time k, correlate all parameters if $i^{\mu}_{k-1} \neq i^{\mu}_{k}$ meaning if current and previous subsets are equal, in this case $P_{c_t;k} = P_{c_t;k-1}$. If rule subset changes controller parameter correlation will be reset and

$$P_{c_t;k} = diag\left(P^{K}_{c_t,1\cdots i;k}, P^{R}_{c_t,1\cdots i;k}, P^{P}_{c_t,1\cdots i;k}\right)$$

where

$$P^{K}_{c_t,1\cdots i;k} \in \Re^{n_u.n_x.C^{sub}\times 1\times 1}$$

$$P^{R}_{c_t,1\cdots i;k} \in \Re^{n_u.m_u.C^{sub}\times 1\times 1}$$

$$P^{P}_{c_t,1\cdots i;k} \in \Re^{n_u.m_u.C^{sub}\times 1\times 1}$$

with m_u the plant output sensors which directly depend on control actions. Regarding membership parameters, the correlation method also take the condition $i^{\mu}_{k-1} \neq i^{\mu}_{k}$ where if not fulfilled,

$$P_{\mu;k} = diag\left(P^{\xi;n_1}_{\mu;k}, \cdots, P^{\xi;n_n}_{\mu;k}\right)$$

with $P^{\xi;n_1}_{\mu;k} \in \Re^{2\times 1\times 1}$ considering ξ the two fuzzy sets for input n used in evaluation of i^{μ}_{k}. For input and rule weights a similar approach as in MFs can be used if rule subset equality condition is not fulfilled

$$P_{\gamma;k} = diag\left(P^{\tau;i^{\mu}_{k}}_{\gamma;k}, P^{agr;i^{\mu}_{k}}_{\gamma;k}\right)$$

where $P^{agr;i^{\mu}_{k}}_{\gamma;k} \in \Re^{C^{sub}\times 1\times 1}$ and $P^{\tau;i^{\mu}_{k}}_{\gamma;k} \in \Re^{2n\times 1\times 1}$. Optimization algorithm can be seen as cascading of local optimizations defined as:

1) Consequents optimization:

$$\left[w^{\psi}(k), P_{\psi}(k), R^{r}_{\psi}(k), R^{e}_{\psi}(k)\right] = UKF\left(w^{\psi}(k-1), P_{\psi}(k-1), R^{r}_{\psi}(k-1), R^{e}_{\psi}(k-1)\right)$$

2) Membership optimization:

$$\left[w^{\mu}(k), P_{\mu}(k), R^{r}_{\mu}(k), R^{e}_{\mu}(k)\right] = CIUKF\left(w^{\mu}(k-1), P_{\mu}(k-1), R^{r}_{\mu}(k-1), R^{e}_{\mu}(k-1)\right)$$

3) Rules degrees and input weights optimization:

$$\left[w^{\gamma}(k), P_{\gamma}(k), R^{r}_{\gamma}(k), R^{e}_{\gamma}(k)\right] = CIUKF\left(w^{\gamma}(k-1), P_{\gamma}(k-1), R^{r}_{\gamma}(k-1), R^{e}_{\gamma}(k-1)\right)$$

4) State optimization:

$$[x(k), P_x(k), R_x^r(k), R_x^e(k)] = UKF\left(w^x(k-1), P_x(k-1), R_x^r(k-1), R_x^e(k-1)\right)$$

5) Controller optimization:

$$\left[w^{Ct}(k), P_{Ct}(k), R_{Ct}^r(k), R_{Ct}^e(k)\right] = PUKF\left(w^{Ct}(k-1), P_{Ct}(k-1), R_{Ct}^r(k-1), R_{Ct}^e(k-1)\right)$$

where $w^{\psi} = VEC\left(w^A, w^B, w^C\right)$, $w^{\gamma} = VEC\left(w^{\tau}, w^{agr}\right)$ and $w^{Ct} = VEC\left(w^K, w^R, w^P\right)$

Parameter estimation separation gives the possibility to have different tuning values for each process, being possible to control their convergence speed independently. Algorithm formulation for UKF and CIUKF regarding parameter and state estimation is computed according Table A.1 for consequents, Table A.2 for memberships, Table A.3 for rules and input weights and Table A.4 for state estimation $x^+(k)$. Before starting optimization algorithm, RNFS for model $RNFMS$ and RNFS for controller $RNFCS$ need to be initialized. Initialization of $RNFMS$ can be done by creating a state space model producing $\omega_0^A, \omega_0^B, \omega_0^C$ and spreading that model to all $RNFMS$ local models, using $\omega_0^{\tau} = 1$ and $\omega_0^{agr} = 1$. Having global model matrices ω_0^A, ω_0^B and ω_0^C, create a zero $RNFCS$ structure and select any local controller parametrization ω_0^K, ω_0^R, ω_0^P, then using offline data, apply PUKF algorithm to controller optimizing $\omega_0^K, \omega_0^R, \omega_0^P$. After optimizing the local controller spread that value to all local models. From table Table A.4 it can be observed that states are not constrained, although if they were used in rules premise as described by Figure 3.8, a CIUKF approach should be taken similarly to MF parameter optimization, confining states to $[lb^x, ub^x]$ UD. The approach of recomputing new sigma points as described in Table 4.1 (by option 1 and option 2) was not followed due to an increase of computational cost, which is a constraint for a real time system identification with small sampling times. Constraints handling in proposed CIUKF algorithm, are implemented by, firstly constrain sigma points and their weights according to Equation 4.8, ensuring a confined subspace during time update equations. Secondly, in order to guarantee weights inside a bounding limit, a direct projection of $w^+(k)$ is done according to (A.10)g. Meanwhile, it can be observed in case of controller optimization that computed control actions during UKF time update might be outside its allowable UD range. To overcome this restriction, it is proposed a PUKF Table A.5 using a direct projection of outside control actions during measurement update. In controller time update stage, same constraints will be applied in order to correctly evaluate model response for each controller sigma point solution, although no constraint will be defined when producing

u^-_{ref}. The disadvantage of this method in comparison with CIUKF, relies on fact that constraints will not be reflected on measured covariance. A note should be highlighted regarding controller optimization algorithm, without evaluation of an intermediate sigma point u^-_{ref}, controller weights innovation would depend on the error between reference and sensor outputs. Since no constraints are used in time update stage, controller weights would have a continuous gain until $y^d(k) - \hat{y}(k) \neq 0$ even if sigma points started producing a saturated i.e $u^-(i) > 1$ control action, once no constraints are handled in this stage. Using a projection scheme during time update would result in a gain divergence due to the fact of not being captured correct mean and covariance. A solution to this problem was achieved with the introduction of an intermediary sigma point u^-_{ref}, whose mean will be used as a reference for controller weights innovation evaluation.

4.5 Conclusion

This section presented the basic theory needed to understand the Unscented Transformation and its application to Kalman Filter theory. Due to nature of proposed RNFS architectures not all presented algorithms are suitable to be used, the set of algorithms are restricted not only by the need of constraint handling both in parameters and state estimation, but also by the loss of covariance matrix kalman continuity $P^+(k) \overset{k=k+1}{\neq} P^+(k-1)$ due to possible rules subset change for each iteration. The need to evaluate at each iteration a sub covariance matrix, makes square root methods difficult to implement since in those methods a Cholesky innovation matrix containing Cholesky update factors are estimated instead of a complete covariance matrix, being difficult to take a subspace approach and constraints using ICUT. Although a subspace of rules C^{sub} are evaluated for each iteration, its dimension is predefined and kept constant during identification process. An approach of a constant sub covariance matrix $P^+(k) \overset{k=k+1}{=} P^+(k-1)$ could be followed, such approach however, can lead system to instability because variance values of a rule subset may not reflect the accuracy of other subsets local models, resulting on inappropriate initial gains leading to a large convergence time and increasing the probability of RNFS instability. To avoid a covariance matrix whose elements do not reflect variable variances and cross-covariances leading to a wrong model confidence, a local and a global covariance method was defined. Although this technique solved the wrong parameter error variance assignment, it also required the use of rule dependent output error covariances R^e_k. Otherwise, high gains during rule transition could be observed due to a low value of R^e_k which might had being diminishing from a previous long term run rule subset. Regarding constraint handling several authors proposed the computation of quadratic problems during measurement update, however solving non linear quadratic problems is time consuming critical in a real time identification process. Instead it was proposed both an interval unscented transformation and a clipping method i.e a direct projection of outsiders into subspace boundaries.

5

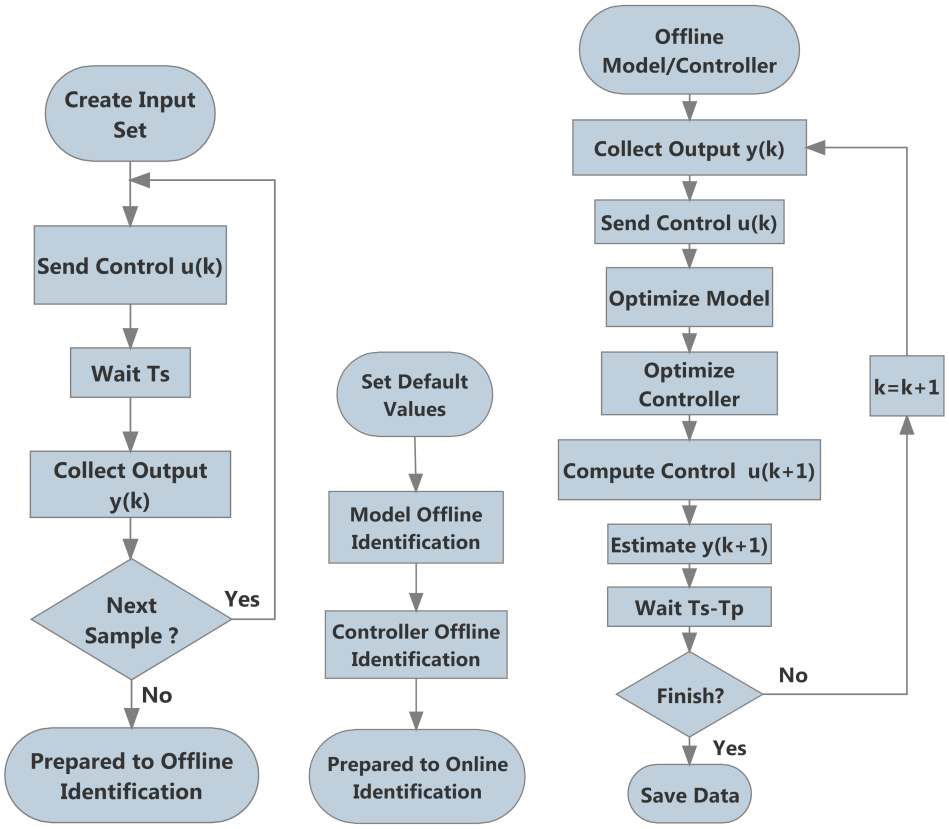
Implementation

5.1 Introduction

This section will apply the proposed identification solution and its MRAFC to two distinct nonlinear stochastic processes. Firstly, it will be presented results regarding system output prediction, its states and parameters evolution during identification process, then will be presented proposed controller architecture. Before using method for online identification, an offline approach in order to find an acceptable initial solution will be conducted. The plan can be described by the chart Figure 5.1. It is worth mention that initial process dynamics are captured using an open loop, were an initial set of data must be collected for offline estimation. To create input data vector, it is defined input set points coupled with additive Gaussian noise were system is supposed to be stable (using matlab *idinput(PRBS)*). After having an offline solution of both model and controller, chapter proceeds with an online identification where plant dynamics are captured using a closed loop architecture as shown in Figure 3.11 and Figure 1.2. Controller cost function is based on error between model output and desired reference. Regarding offline identification, it is computed a linear State Space model based on collected data using matlab identification toolbox e.g

$$n4sid(IDDATA(datOut, datIn, 1), nx, 'DisturbanceModel', 'None', 'Cov', 'None')$$

where nx is the number of states. The obtained local model is then applied to all rules consequents. The number of membership functions and their centers were designed to match input setpoints. In case of u_k and y_k inputs it will be considered the universe of discourse $UD = [-1; 1]$ and the respective variable normalization. Input weights W^{τ} and



(a) Data Collect Process (b) Offline Process (c) Online Process

Figure 5.1: Workflows for system identification and control

rule weights W^{agr} are set equal to unity. For controller, its fuzzy structure parameters will be set in a first approach equal to model parameters, and then reference values will be considered during RNFCS premise evaluation using default values. Controller rule consequents W^K, W^R and W^P will be set to zero as default value. For a best understanding of online identification workflow 5.1(c), consider the following description:

1. Collect output y_k
2. Send control action u_k that will produce output y_{k+1} to be collected in next iteration
3. Optimize model parameters to produce

$$\left[W_k^M, \hat{x}_k, ukf_k^M\right] = f\left(u_{k-1}, y_k, y_{k-1}, W_{k-1}^M, \hat{x}_{k-1}, ukf_{k-1}^M\right)$$

, where $W^M = [W^{tau,agr,\mu,A,B,C}]$ and ukf^M are the covariances of proposed algorithm Table A.2 for model optimization.

4. Based on optimized model premises and states, optimize controller parameters

$$\left[ukf_k^C, W_k^{K,R,P}\right] = h\left(ukf_{k-1}^C, W_{k-1}^{K,R,P}, W_k^M, \hat{x}_k, y_k, R_k\right)$$

5. Compute next control action $u_{k+1} = f\left(W_k^{K,R,P}, u_k, R_k, y_k, \hat{x}_k\right)$

6. At this time we can already predict $\hat{y}_{k+1} = f\left(u_{k+1}, y_k, W_k^M, \hat{x}_k\right)$

7. Wait $T_s - T_p$ to simulate process intrinsic delay respecting process sampling time, where T_s is process sampling interval and T_p is time consumed during algorithm processing.

Due to a large number of consequent variables, the display of results e.g covariance matrices could become a hard and exhaustive process. To overcome this lack of scalability it was introduced a data filter during representation keeping the number of variables constant independently on the rule number. The rule filtering method is described as follows:

1. The covariances regarding the two rules with higher membership degree will be displayed

2. The covariances regarding the two rules having largest UKF gains will also be displayed, and computed as in (5.1)

3. A plot with rule weights will include

   Two highest degrees of membership

   Two highest UKF gain rules, and they degree of membership in comparison with the two highest as described in (5.3)

The following set of equations handles the consequent variance aggregation for each rule to be used for a display analysis only.

$$
\begin{aligned}
\chi_A^{\psi-} &= reshape\left(\chi^{\psi-}(k), A\right) \rightarrow \chi_A^{\psi-} \in \Re^{n_x * n_x, C^{sub}, L} \\
\chi_A &= \sum_{i=1}^{n_x^2} \chi_A^{\psi-}(i,j,k)\,, \forall j \in \left\{1,2,...,C^{sub}\right\}; k \in \{1,2,...,L\} \rightarrow \chi_A \in \Re^{C^{sub},L} \\
R_A^{r-} &= reshape\left(R^{r-}(k), A\right) \rightarrow R_A^{r-} \in \Re^{n_x *, n_x, C^{sub}} \\
R_A^r &= \frac{1}{n_x^2} * \sum_{i}^{n_x^2} \left(R_A^{r+}(i,i,j)\right), \forall j \in \left\{1,2,...,C^{sub}\right\} \rightarrow R_A^r \in \Re^{C^{sub}} \\
\omega_A^{\psi-} &= reshape\left(\omega^{\psi-}(k), A\right) \rightarrow \omega_A^{\psi-} \in \Re^{n_x n_x, C^{sub}} \\
\omega_A^{\psi} &= \sum_{i}^{n_x^2} \omega_A^{\psi-}(i,j), \forall j \in \left\{1,2,...,C^{sub}\right\} \\
P_A^- &= \sum_{i}^{L} \left(\chi_A(:,i) - \omega_A^{\psi}\right) * W_i^{(c)} \left(\chi_A(:,i) - \omega_A^{\psi}\right)' + diag\left(R_A^r\right) \\
K_A^- &= reshape\left(K, A\right) \rightarrow K_A^- \in \Re^{n_x \times n_x, C^{sub}} \\
K_A &= \sum_{i}^{n_x^2} K_A^-(i,j), \forall j \in \left\{1,2,...,C^{sub}\right\} \\
P_A &= P_A^- - K_A P_{\tilde{d}_k} K_A' \\
\overline{P_A^a} &= P_A(i) \rightarrow i \text{ are the indices of two most relevant rules from inference} \\
\overline{P_A^r} &= P_A(j) \rightarrow j \text{ are the indices of two rules with highest gains}
\end{aligned}
\tag{5.2}
$$

Controller weights are computed in a similar process meanwhile, for membership functions ω^{μ} and states $\hat{x}$ filtering is not needed, since they are rule independent. In order to analyse system convergence, it is also needed a weighting reference regarding absolute rules β^a and relative rules β^r computed as follows:

$$
\beta(i) = \frac{\prod \left(1 - w_{:,i}^{\tau}(k)\left(1 - diag(M_{:,:,i}^{\xi} M^{\sigma'}(k))\right)\right) w_i^{agr}(k)}{\sum_{i=1}^{C} \prod \left(1 - w_{:,i}^{\tau}(k)\left(1 - diag(M_{:,:,i}^{\xi} M^{\sigma'}(k))\right)\right) w_i^{agr}(k)}
\tag{5.3}
$$

$$
\begin{aligned}
\beta_A^a &= sort\left(\beta, Ascend, 2\right), \rightarrow \beta_A^a \in \Re^2 \\
\Delta P_A &= K_A P_{\tilde{d}_k} K_A' \\
\beta_A^r &= sort\left(DeltaP_A, Ascend, 2\right), \rightarrow \beta_A^a \in \Re^2 \\
Ratio_A &= \frac{\sum \beta_A^r}{\sum \beta_A^a}
\end{aligned}
\tag{5.4}
$$

Next will be presented results using a SISO and a MIMO process during offline (plant in open loop) and online identification (plant in closed loop).

5.2 Process PT326

For the first process it was used a SISO Feedback PT326 thermal process Figure 5.2, were the goal was to control the output air temperature using a sampling time of $100ms$. Data acquisition was done using a usb DaQ NI-USB6009, where input u and output y domains $\{u|0 \leq u \leq 5\}$ and $\{y| -10 \leq y \leq 10\}$ were projected to the normalized domains $\{\overline{u}| -1 \leq \overline{u} \leq 1\}$ and $\{\overline{y}| -1 \leq \overline{y} \leq 1\}$ respectively.

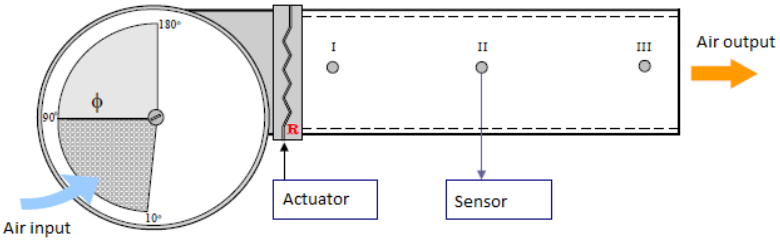


Figure 5.2: Feedback PT 326 plant

5.2.1 Offline Identification results

In order to proceed with offline identification, it was firstly created an input set using the script Table 5.1, obtaining process response according to Figure 5.3. The selected fuzzy sets for input and output is according to Figure 5.4, where selected MF centers for input were $[0; 0.5; 0.9]$ for a total of 3 fuzzy sets. Regarding output MF centers it was selected $[0; 0.2; 0.5]$ for a total of 3 fuzzy sets. The complete set of rules cover all possible fuzzy sets combinations resulting in a total of 9 rules. Consequents were initialized using a global model obtained through the application of matlab identification toolbox i.e

$$\left[\omega^{A,B,C}\right] = n4sid(datOut, datIn, nx, 'DisturbanceModel', 'None', 'Cov', 'None')$$

with $nx = 1$. The default RNFMS response to input data u is represented in Figure 5.4. Since all local models have the same parameters, rules premises does not concern the fact that either system outputs or model predicted outputs are applied, resulting in a same model response for both architectures i.e a series parallel or parallel. This way, both had a mean square error $MSE(\hat{y} - y) = \frac{1}{n}\sum(\hat{y} - y)^2 = 0.0016$.

Due to effort and time constraints, it will not be presented results regarding rules weight $\omega^{agr}(k)$ and input weights $\omega^{\tau}(k)$, they will have a constant value equal to one. Having model parameters initialized, it is proceeded with the offline identification using proposed algorithm Table A.1 for parameter estimation and Table A.2 for state estimation,

$tam = 600;$
$spread = 0.04;$
$mean = 0.1;$
$u(1 : tam, 1) = idinput(tam,' PRBS', [00.7], [mean - spreadmean + spread]);$
$mean = 0.5;$
$u(tam+1 : 2*tam, 1) = idinput(tam,' PRBS', [00.7], [mean-spreadmean+spread]);$
$mean = 0.9;$
$u(2 * tam + 1 : 3 * tam, 1) = idinput(tam,' PRBS', [00.7], [mean - spreadmean + spread]);$

Table 5.1: PT326 input data script

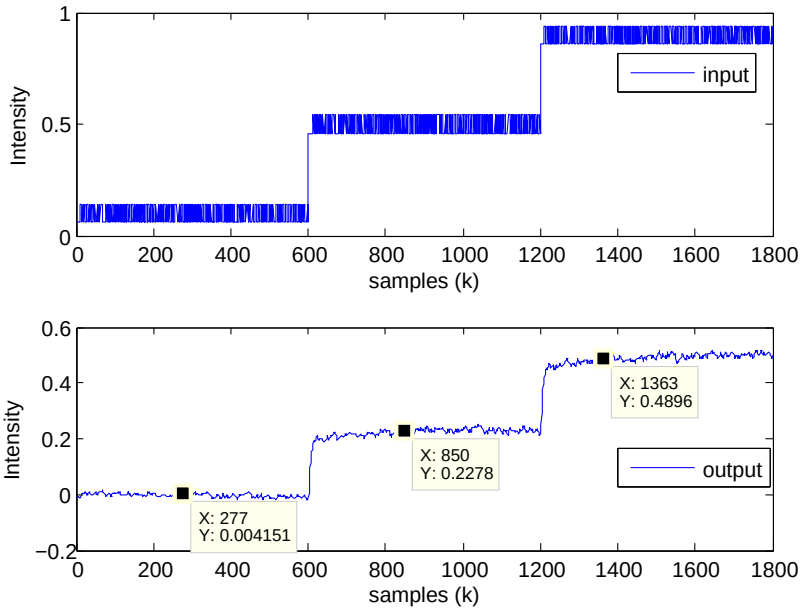


Figure 5.3: PT326 Collected data

and considering algorithm parameters which had the best results:

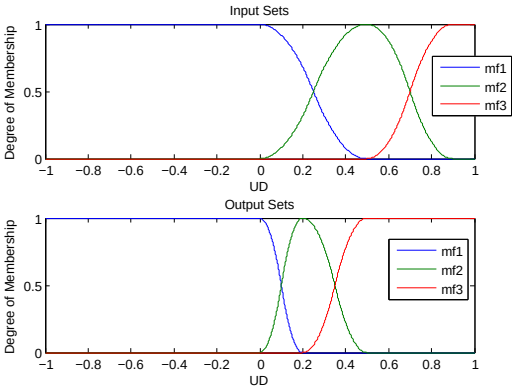
$$
\begin{array}{lllll}
C^{sub} = 9 & n = 2 & n_x = 2 & n_u = 1 \quad p = 2 & P_{max} = 3 \\
\alpha_\mu^r = 0.001 & \alpha_\psi^r = 0.05 & \alpha_x^r = 0.2 & & \\
\alpha_\mu^e = 0.1 & \alpha_{\psi;i}^e = \beta_i * 0.01 & \alpha_x^e = 0.3 & & \\
\alpha_\mu^P = 1e-2 & \alpha_\psi^P = 1e2 & \alpha_x^P = 1e-2 & & \\
\alpha_\mu^R = 1e-4 & \alpha_\psi^R = 1e-2 & \alpha_x^R = 1e-2 & & \\
\alpha_\mu^E = 1e-2 & \alpha_\psi^E = 1e2 & \alpha_x^E = 1e-2 & &
\end{array}
\tag{5.5}
$$

Where $i = 1 \cdots C^{sub}$ and β_i the rule degree.

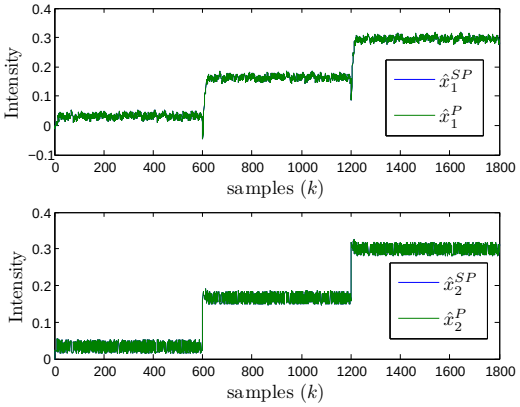
Presentation results will be split into three stages:

1) Rules consequent optimization only;

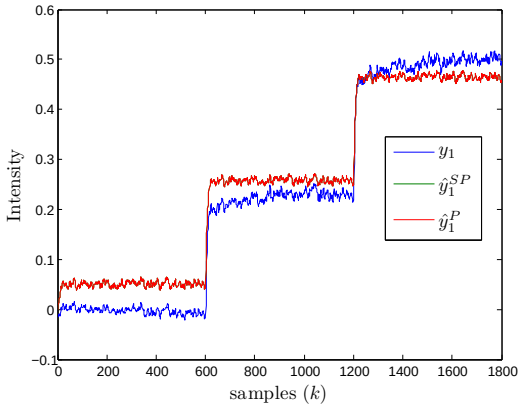
2) Rules consequent plus state optimization;



(a) RNFMS membership functions



(b) Initial RNFMS state evolution for the case of series parallel $\hat{x}^{SP}$ and parallel $\hat{x}^{P}$

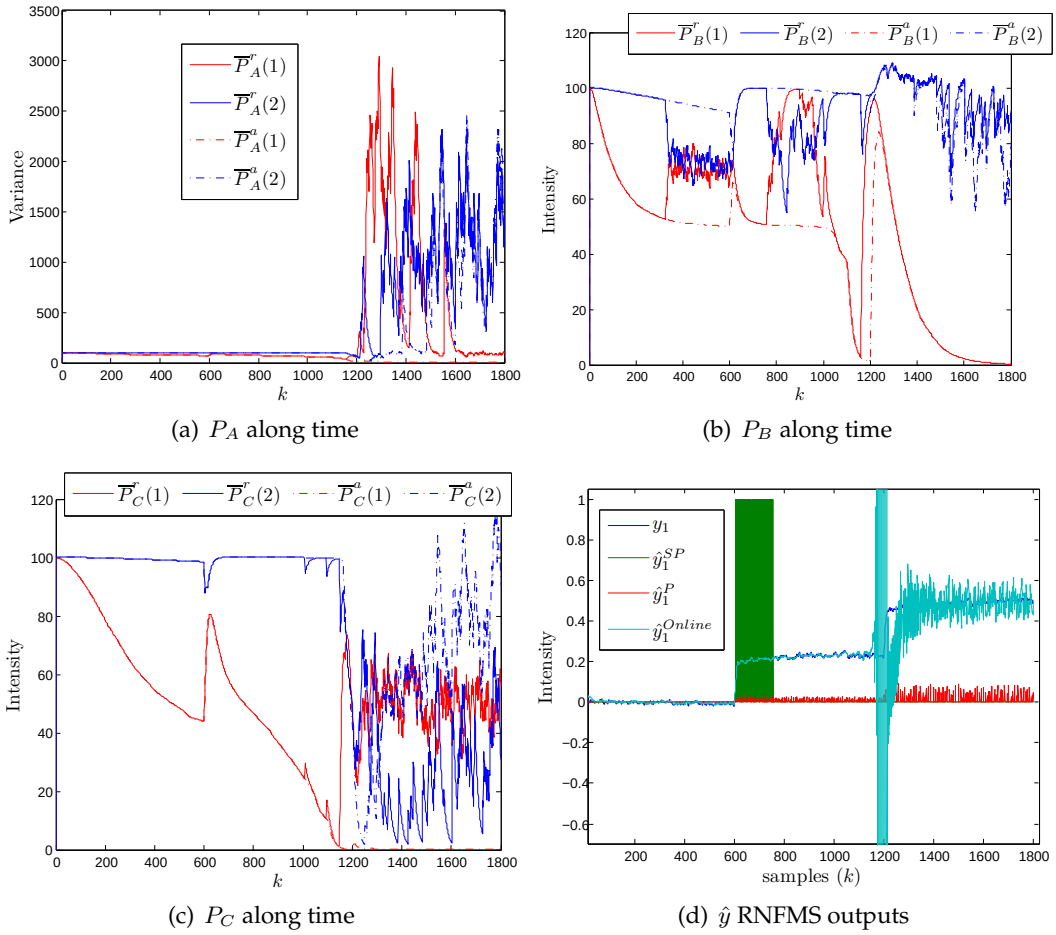


(c) Initial RNFMS dynamic output, $\hat{y}^{SP}$ concerns the series parallel architecture, $\hat{y}^{P}$ refers to parallel architecture. Mean square error between system output and model predicted output is $MSE\left(\hat{y}^{P} - \hat{y}\right) = 0.0016$

Figure 5.4: RNFMS membership functions

3) Rules consequent plus state plus membership optimization;

Proposed algorithm optimal parametrization may vary depending on plant sensitivity and dynamics, selected solution was based not only on a trial and error approach, but also accomplished with an understanding of UKF dynamics. For instance regarding this specific process it can be noticed a parameter divergence and instability resulting in a poor model Figure 5.5. The reason for this behaviour relates to the existence of high gains due to a low forgetting factor for output error covariance R_{ψ}^{e}, which will become lower than parameter covariance P_{ψ}. From all parameters, the output matrix W^{C} is specially the most critical when facing a high gain and if any correlation between its parameters are considered. This way, for some systems one possible solution is to force correlations to zero making P_C a diagonal matrix with only variance values. Other possibility would be the decrease of forgetting factor α_{ψ}^{e} although, this might on other hand slow the algorithm convergence. Using a diagonal matrix P_C when computing equations a



(a) P_A along time

(b) P_B along time

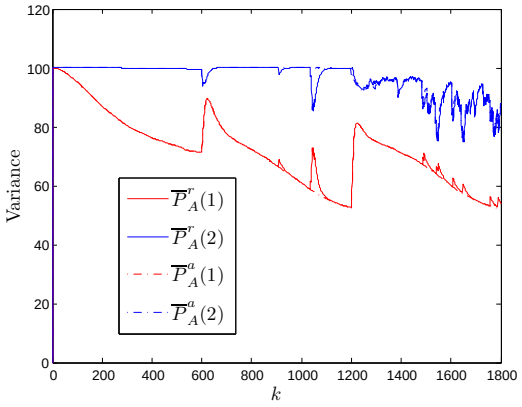
(c) P_C along time

(d) $\hat{y}$ RNFMS outputs

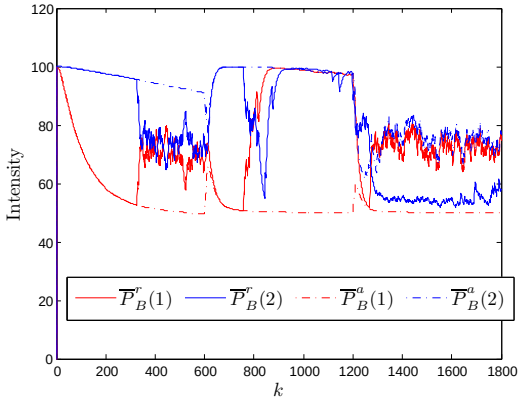
Figure 5.5: RNFMS with rules consequents optimized, W^C divergence phenomenon.

system stability improvement as shown in Figure 5.6 was achieved. Having presented model consequent optimization, it is proceeded with states estimation according to proposed algorithm. Achieved results are described in Figure 5.7 With state optimization it can be observed an output error decrease, although it increases model offline error. This was expected since with optimized states a lower prediction error will be achieved. Computed parameters innovations will also have lower gains. With states optimization, consequents stability is increased since parameters are under a lower stress. Next step towards completion of offline analysis, is the optimization of membership functions.

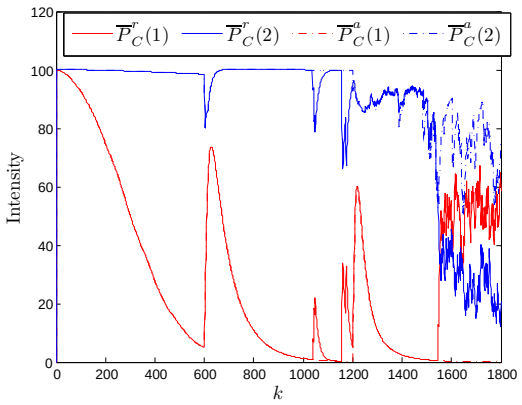
Membership function optimization will be optimized according proposed algorithm which should be capable of constraint handling as defined in previous chapter. From Figure 5.8 it is observed an error increase for all architectures except the parallel. Comparing 5.8(e) with 5.8(d), the latest has sharpened rule degrees since MF optimization increased fuzzy sets core providing linguistic variables with a higher membership degree. MFs constraint handling by proposed algorithm suffers from the fact that mean is wrongly measured in the presence of outsiders. Since default MF parameters are set equal to their centers and



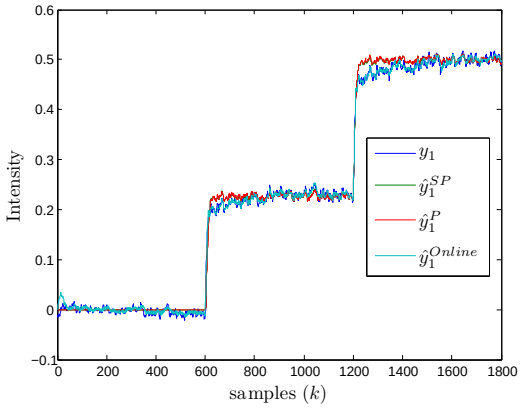
(a) P_A along time



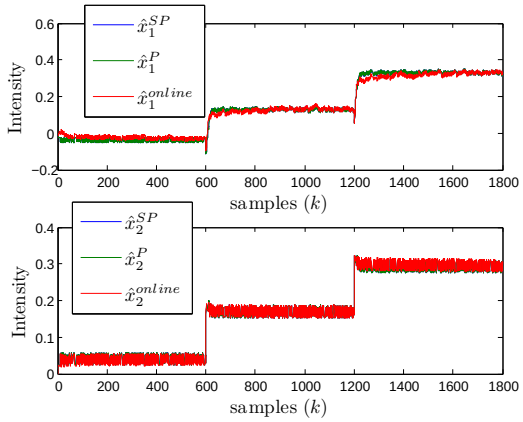
(b) P_B along time



(c) P_C along time



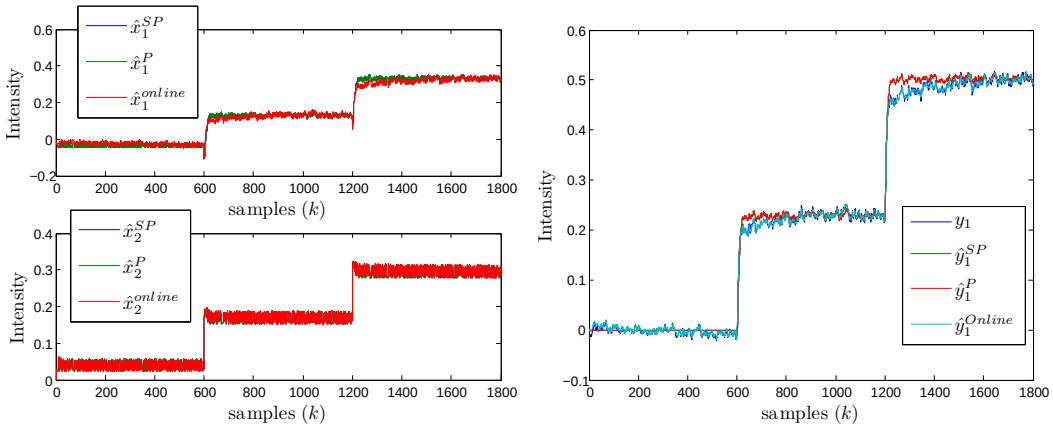
(d) $\hat{y}$ RNFMS outputs; $MSE^{online} = 3.4714e-5$; $MSE^{SP} = 1.1913e-4$; $MSE^{P} = 1.3019e-4$



(e) $\hat{x}$ predicted states

Figure 5.6: RNFMS with rules consequents optimized, no divergence.

being those a boundary, it will as can be seen by 5.8(c) enlarge fuzzy set core. The consequence of this behaviour might result in a wrong solution, although it is expected a normal behaviour as soon as no outsiders are present. Apart from any possible lack of confidence from proposed algorithm, MF optimization might have some drawbacks i.e:



(a) $\hat{x}$ along time

(b) RNFMS outputs along time

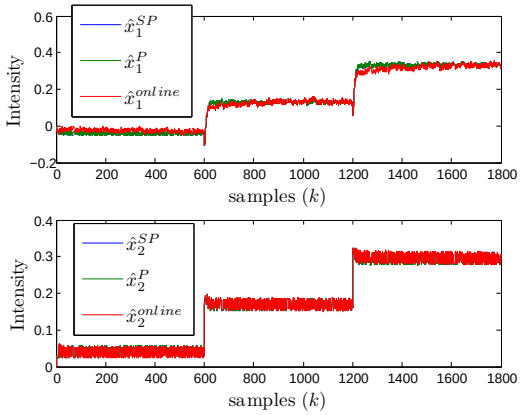
Figure 5.7: RNFMS with rules consequents and states optimized, MSE : $\hat{y}^{SP} = 1.4834e^{-4}; \hat{y}^{P} = 1.5641e^{-4}; \hat{y} = 1.3318e^{-5}$

1. New rules with a higher degree but first time fired, if their local models fitting degree is worst than a lower degree rule local model, MF optimization might wrongly increase the lower rule degree.

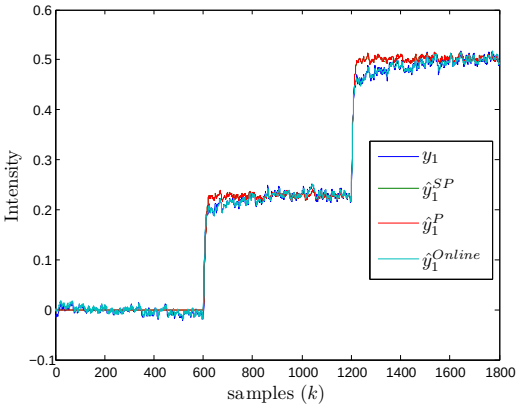
2. Learning rate decrease of new rules, and also wrong MF core enlargement.

3. Old solutions for rules subset local models can be degraded

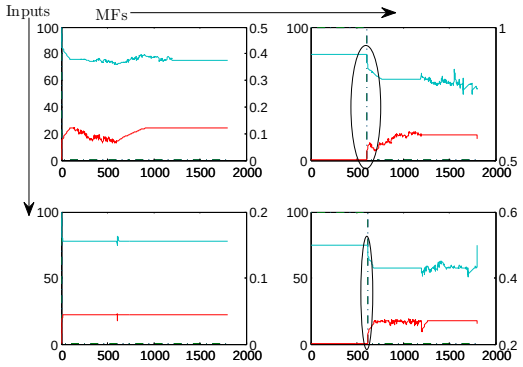
Considering previous statements regarding membership optimization, it is not advised the use such method for online identification and control.



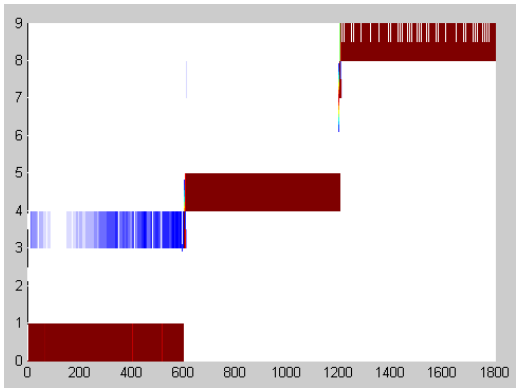
(a) $\hat{x}$ along time



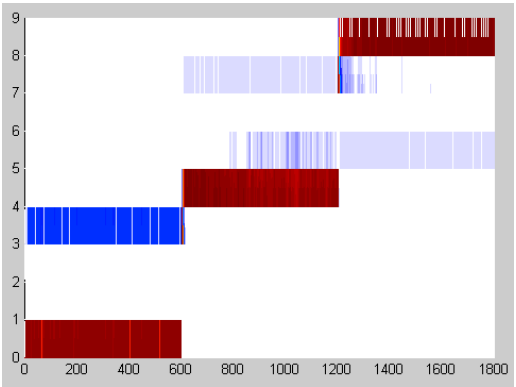
(b) RNFMS outputs along time



(c) MFs paramenters along time



(d) β_i along time after MF optimization



(e) β_i along time before MF optimization

Figure 5.8: RNFMS with rules consequents, states and MFs optimized, MSE : $\hat{y}^{SP} = 1.5297e^{-4}; \hat{y}^{P} = 1.5489e^{-4}; \hat{y} = 1.3591e^{-5}$

5.2.2 Online Identification results

NOTE:

In this subsection it will be handled UKF processing during online process identification. Due to an unexpected problem on plant under consideration, it could not be used during online experimentation. In order to maintain offline analysis it was decided to work at this stage with a plant model. The model considered to replace the real process was obtained in [62] where three state space models for three working points depending on fan speed were obtained. In order to bring some non-linear properties, models will be combined in a basic fuzzy model and an error variance with zero mean and Gaussian distribution i.e

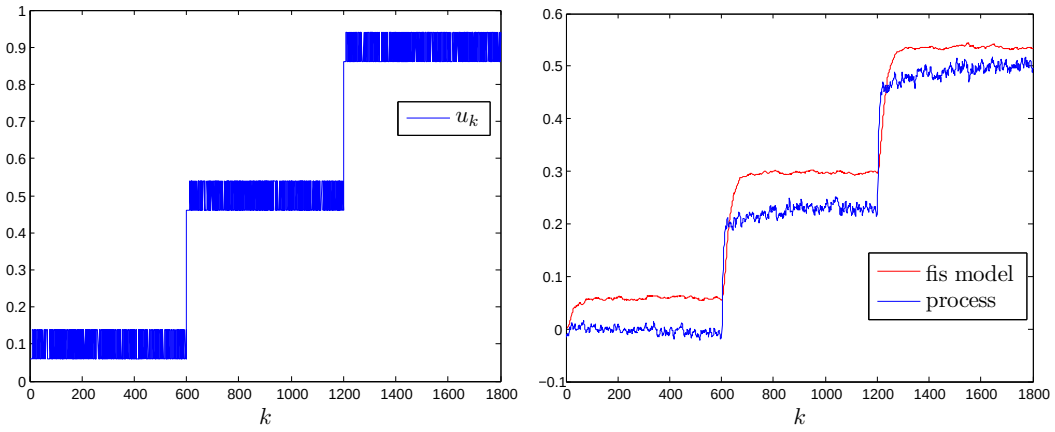
$$0.0007 * randn(Samples, 1)$$

will be considered. The fuzzy model will have three MF functions for the fan speed v_k^{fan} linguistic variable, resulting in a total of three rules, each having as consequent a SS model. The complete inference can be seen as:

$$\begin{aligned}
MF_1 &= zmf\left(v_k^{fan} + v_k^k, [0\ 0.5]\right) \\
MF_2 &= pimf\left(v_k^{fan} + v_k^k, [0\ 0.5\ 0.5\ 1]\right) \\
MF_3 &= smf\left(v_k^{fan} + v_k^k, [0.5\ 1]\right) \\
Rule_1 &= if\ v_k^{fan}\ in\ MF_1\ then\ x_k^1 = \begin{bmatrix} 0.972 & 0.023 \\ -0.040 & 0.899 \end{bmatrix} x_{k-1}^1 + \begin{bmatrix} -7.5e^{-4} \\ 0.017 \end{bmatrix} u_{k-1} + v_k^w \\
y_k^1 &= \begin{bmatrix} 19.338 & 0.123 \end{bmatrix} x_k^1 + v_k^e \\
Rule_2 &= if\ v_k^{fan}\ in\ MF_2\ then\ x_k^2 = \begin{bmatrix} 0.956 & 0.024 \\ -0.028 & 0.894 \end{bmatrix} x_{k-1}^2 + \begin{bmatrix} -1.9e^{-4} \\ 0.023 \end{bmatrix} u_{k-1} + v_k^w \\
y_k^2 &= \begin{bmatrix} 11.469 & 0.050 \end{bmatrix} x_k^2 + v_k^e \\
Rule_3 &= if\ v_k^{fan}\ in\ MF_3\ then\ x_k^3 = \begin{bmatrix} 0.973 & 0.020 \\ 4e^{-4} & 0.847 \end{bmatrix} x_{k-1}^3 + \begin{bmatrix} 3.8203e^{-5} \\ 0.052 \end{bmatrix} u_{k-1} + v_k^w \\
y_k^3 &= \begin{bmatrix} 6.7227 & -0.0860 \end{bmatrix} x_k^3 + v_k^e
\end{aligned} \tag{5.6}$$

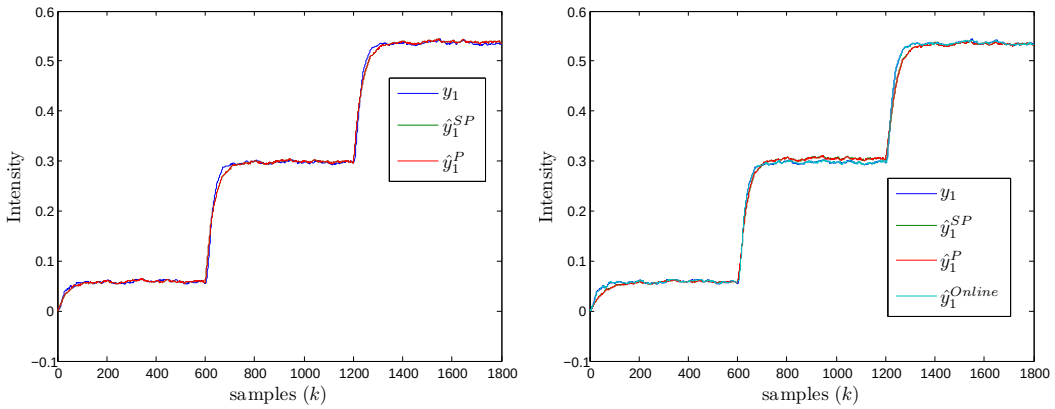
It will be considered fan with speed regulator in position 0.4 which applying same offline data produces a model response as in Figure 5.9. It can be noticed some differences between fis model and offline process captured data, this might be related to different conditions e.g environmental or even changes in setup. To overcome this differences, it is needed to initialize RNFMS with new valid default local models. After collecting a new data set from fis model, finding a new local default SS model (using ident Matlab capabilities), applying this model to all local sub models and then apply UKF for consequents and states optimization, it is expected a model response according to Figure 5.10.

An analysis regarding MF optimization during real-time will not be handled in current plant meanwhile, some results will be provided in during the process identification.



(a) Input data (b) Fuzzy model output vs Real process output

Figure 5.9: Fis model dynamics



(a) Default SS model output; $RMSE = 3.1810e^{-5}$ (b) RNFMS output; $RMSE_{online} = 9.9667e^{-7}$; $RMSE_{series} = 5.3314e^{-5}$; $RMSE_{parallel} = 5.4364e^{-5}$

Figure 5.10: RNFMS dynamics for fis model

Comparing Figure 5.9 with Figure 5.10 a small error in proposed RNFMS in comparison with offline case can be noticed. This dues the fact that nor all process dynamics nor its error distribution can be included in fis model, which will have a much linear dynamic. In order to proceed with online approach a controller must be defined since a closed loop schema will be used.

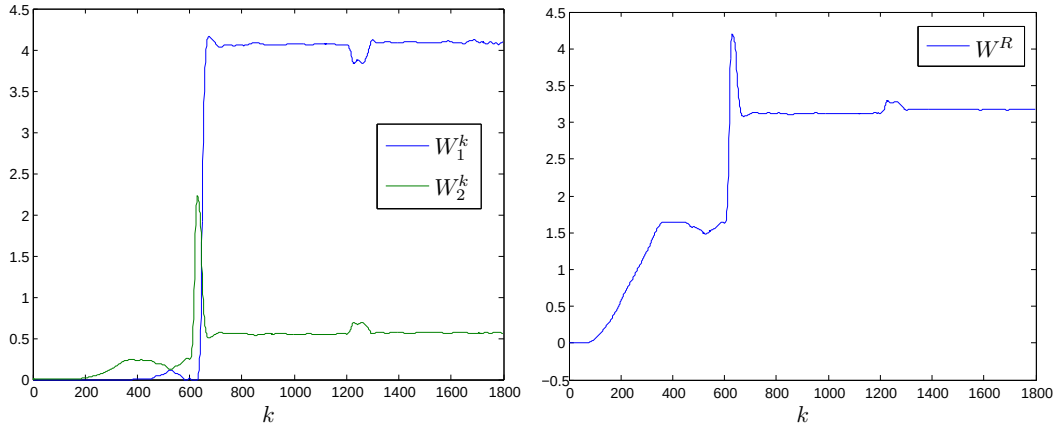
Regarding controller, the proposed method is according to subsection 3.4.2 and optimization algorithm defined by Table A.5. From proposed architecture (3.42) it will be used in this process the reference for rule selection meanwhile, next section will handle

both cases. Algorithm parametrization can be seen as follows:

$$\alpha^{P}_{Ct} = 1e^{1} \qquad \alpha^{R}_{Ct} = 1e^{-2} \qquad \alpha^{E}_{Ct;u} = 1e^{-2} \tag{5.7}$$

$$\alpha^{E}_{Ct;c} = 1e^{2} \qquad \alpha^{e}_{Ct;u} = 0.05 \qquad \alpha^{e}_{Ct;c} = 0.01 \qquad \alpha^{r}_{Ct} = 0.05 \tag{5.8}$$

Before starting online task, a suitable global controller default solution must be found and spread to all rules local controllers. A global SS controller structure will be initialized with zeros and then in an offline task, it is optimized using UKF. The optimized global SS controller is then applied to all local RNFCS controllers. Since controller equation depends on reference y^d which is not available during offline process, it must be considered as reference the model output. This approach works for this particular process since step raising time response is relatively fast in comparison with next process (DTS200). Controller RNFCS response obtained during offline UKF computation is as described in Figure 5.11. Continuing towards realtime identification, a new reference was created and



(a) W^K during UKF processing

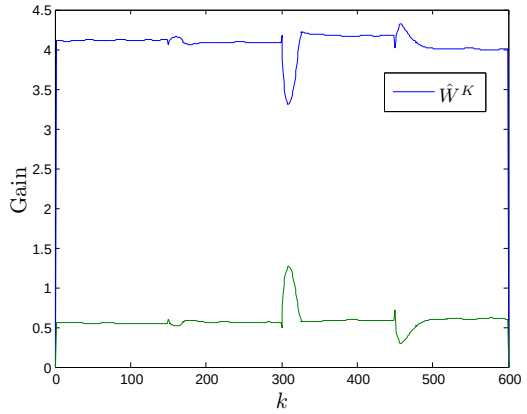
(b) W^R during UKF processing

(c) W^P during UKF processing

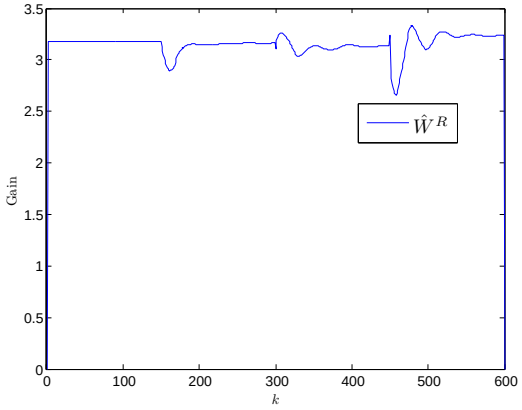
(d) u_k prediction during UKF processing

Figure 5.11: Controller initialization (reference in RNFCS premise)

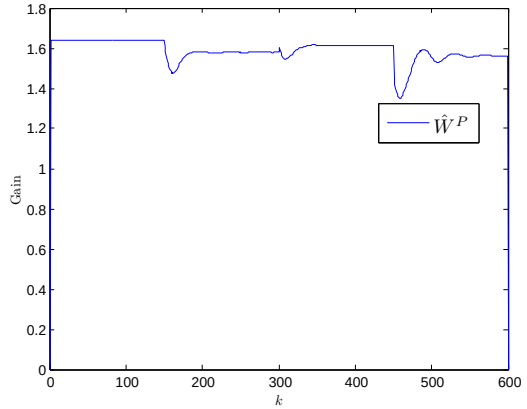
using all values achieved so far, the obtained closed loop response is described in Figure 5.11. From Figure 5.12 it can be noticed that controller successfully guided process



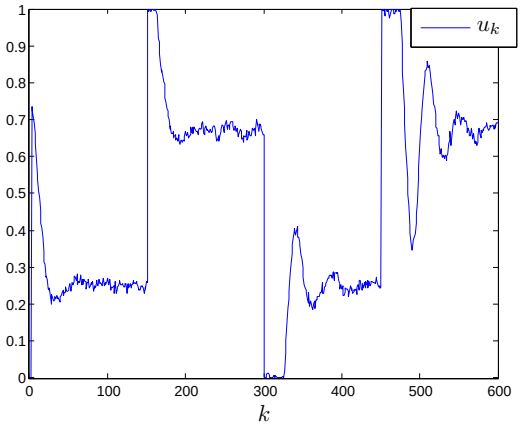
(a) W^K during UKF processing



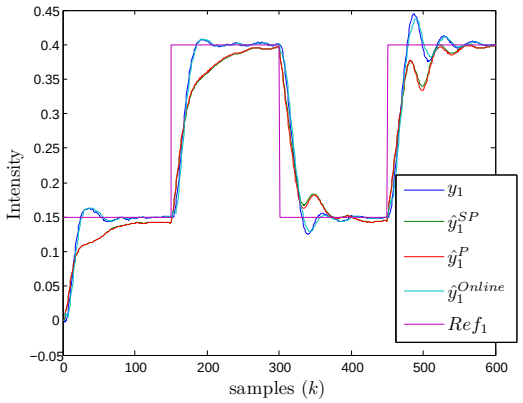
(b) W^R during UKF processing



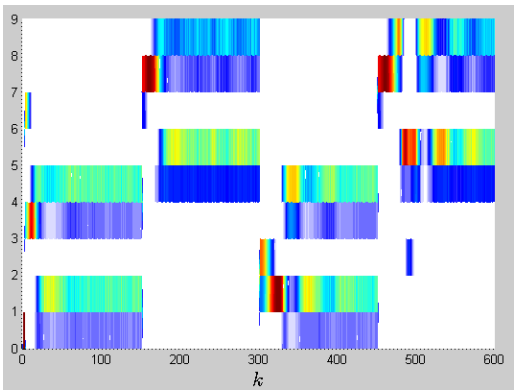
(c) W^P during UKF processing



(d) u_k prediction during UKF processing



(e) y_k prediction during UKF processing

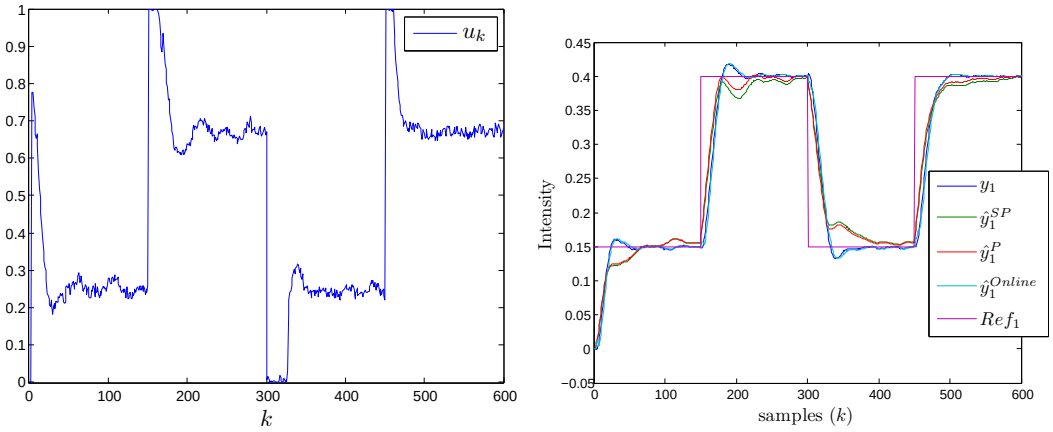


(f) Rule weights during UKF processing

Figure 5.12: Closed loop real-time experiments (reference in RNFCS premise)

output towards reference however, it shows an overshot increase when new reference set points are used. This effect dues to a lower value of $R^{e-}_{Ct}(k)$ and consequently on forgetting factor $\alpha^{e}_{Ct;u}$. A possible solution would be the decrease of $\alpha^{e}_{Ct;u}$ although, it can lead to a long raising time regarding the first samples, and would not solve the problem for a long term run simulation. Another possibility is to do a reset of $R^{e-}_{Ct}(k)$ to its initial value each time a new set point is requested. Using this latest approach an improvement can

be achieved as shown in Figure 5.13, overshot tend to decrease when new references are considered.



(a) u_k prediction during UKF processing (b) y_k prediction during UKF processing

Figure 5.13: Closed loop real-time experiments using covariance reset (reference in RN-FCS premise)

5.3 Process Amira DTS200

The second plant to be used is a MIMO process known as Amira DTS 200. The plant is composed by three tanks each having two manual valves, one to connect tank to its direct neighbour other to drain water out from system, also two pumps are used to feed tank 1 and tank 2. The default layout is described by Figure 5.14, where is only considered tank 1 and tank 2 connected to tank 3, having the latest an extra valve to drain water out from system. Regardless default layout other valves will be used in order to introduce disturbances, i.e tank 1 drain pump will be used as failure n?1 and tank 2 connection to tank 3 will be closed to introduce failure n?2. The systems is characterized by a sampling time of one second, and intensity of both sensors outputs and controller actions will be normalized to the universe of discourse $[-1; 1]$. Proposed controller will not consider tank 3, since no pump is directly used for water supply meanwhile, it will be considered during an offline identification analysis.

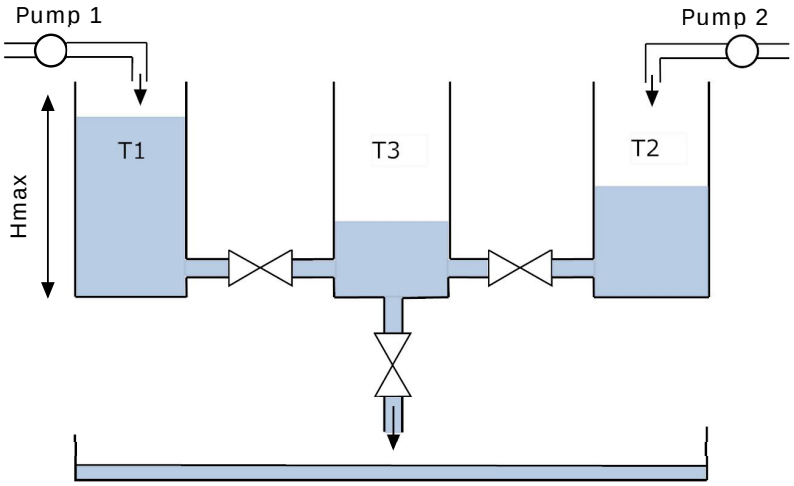


Figure 5.14: Amira DTS200 plant

5.3.1 Offline Identification results

Since from a starting point no controller is available, initial data for offline identification was collected using an open loop approach Figure 5.15. The default values used to initialize fuzzy consequents structure, were obtained through application of matlab identification toolbox i.e

$$\left[\omega^{A,B,C}\right] = n4sid(datOut, datIn, nx, 'DisturbanceModel', 'None', 'Cov', 'None')$$

with $nx = 2$, having default model response illustrated by Figure 5.16 with three fuzzy sets for each input with centers in $[0.3; 0.6; 0.9]$ for pumps and $[0.1; 0.4; 0.6]$ for three tanks

output sensors. Following same approach as in previous process:

$$\begin{array}{llllll} C^{sub} = 32 & n = 5 & n_x = 2 & n_u = 2 & p = 2 & P_{max} = 3 \\ \alpha_{\mu}^{r} = 0.05 & \alpha_{\psi}^{r} = 0.05 & \alpha_{x}^{r} = 0.2 & & & \\ \alpha_{\psi;i}^{e} = \beta_i * 0.01 & \alpha_{x}^{e} = 0.3 & & & & \\ \alpha_{\psi}^{P} = 1e2 & \alpha_{x}^{P} = 1e-2 & & & & \\ \alpha_{\psi}^{R} = 1e-2 & \alpha_{x}^{R} = 1e-2 & & & & \\ \alpha_{\psi}^{E} = 1e2 & \alpha_{x}^{E} = 1e-2 & & & & \end{array} \tag{5.9}$$

Where $i = 1 \cdots C^{sub}$ and β_i the rule degree.
Results presentation will be divided into three stages:

1) Rule's consequent optimization only;

2) Rule's consequent plus state optimization;

3) Rule's consequent plus state plus membership optimization;

Besides representation of $\hat{y}$ which is obtained during UKF computation, optimized model will also be exited for model validation purpose, using a series-parallel and series input architectures Figure 3.9 and Figure 3.10 respectively. It will also be done an analysis regarding model loss when process tanks 3 is not considered since no pump is directly connected to it. Using only consequents optimization it is observed Figure 5.17 a major

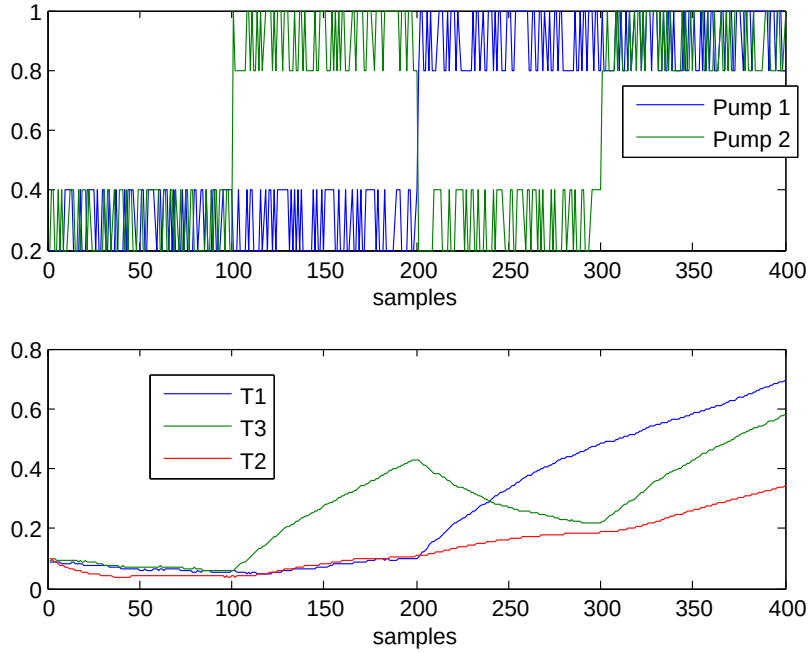
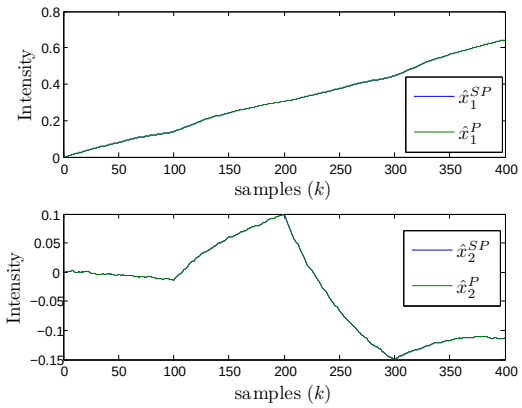


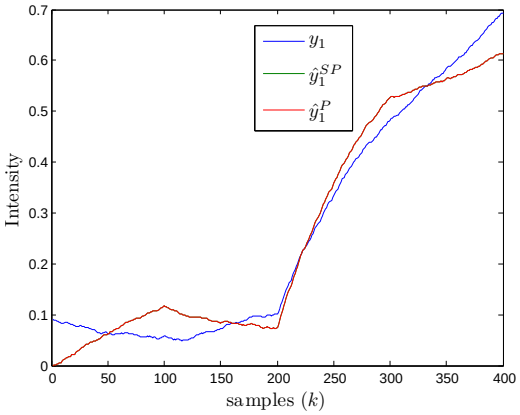
Figure 5.15: Data for offline identification

improvement regarding default SS local model. Including state optimization achieved results are according to Figure 5.18. A similar behaviour as in previous process was

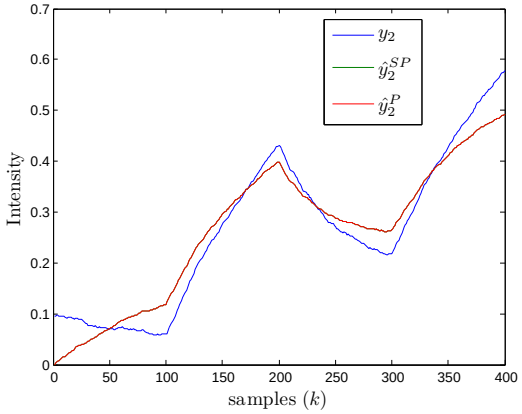
observed i.e an improvement of online prediction error at the expense of small offline model error increase. Including RNFMS membership optimization Figure 5.19, it was observed a minor improvement both for online predicted error, and for tank two an improvement also in offline model error. Similar to what was described in previous section, it was noticed a rule degree concentration in stronger rules when comparing 5.19(e) with 5.19(f). As already mentioned previously, it is expected that inclusion of tank 3 will not bring major advantages in model quality, since its information is implicitly included in sensor T1 and T2. To validate this sentence consider Figure 5.20 where a full parameter optimization was committed, showing an improvement for all tanks both of online and offline model errors.



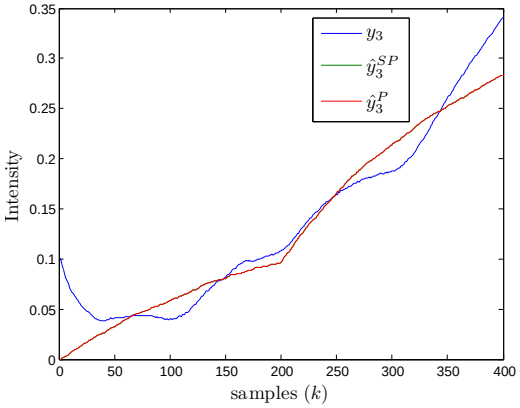
(a) $\hat{x}$ along time



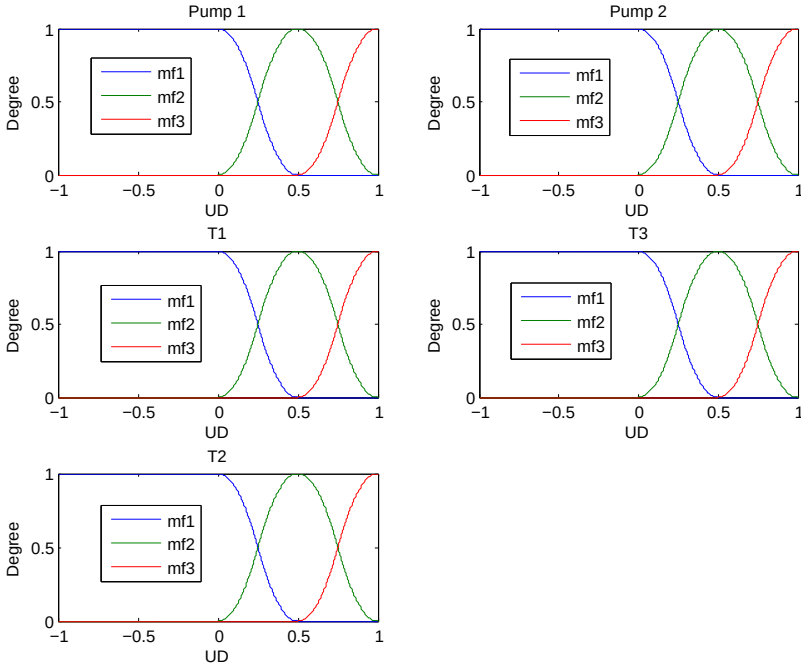
(b) $\hat{y_1}$ along time; $MSE = 0.0014$



(c) $\hat{y_2}$ along time; $MSE = 0.0014$

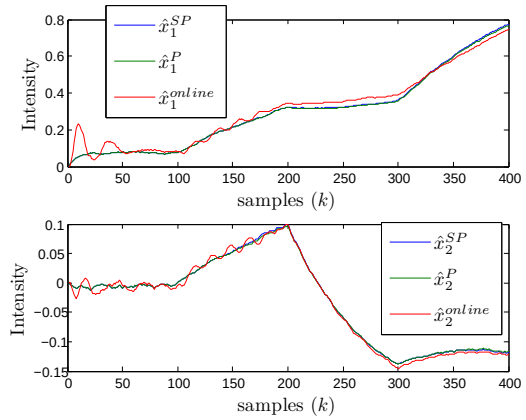


(d) $\hat{y_3}$ along time; $MSE = 0.0006$

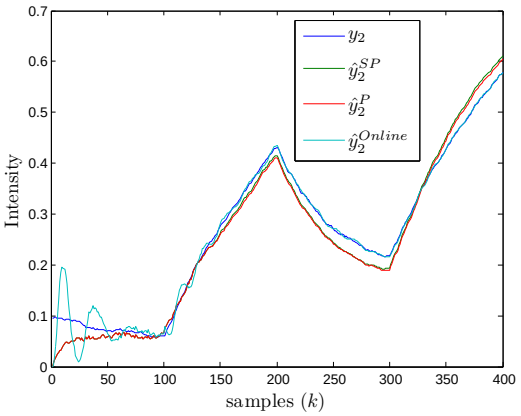


(e) Default fuzzy sets

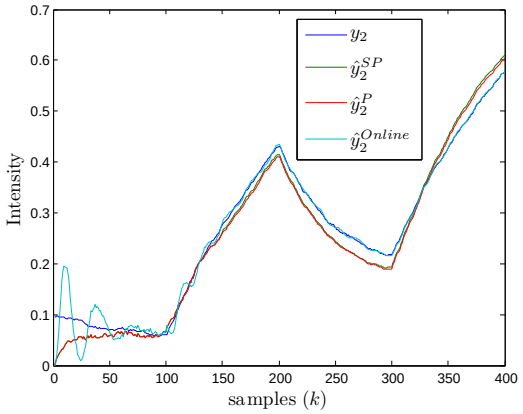
Figure 5.16: Default local model response



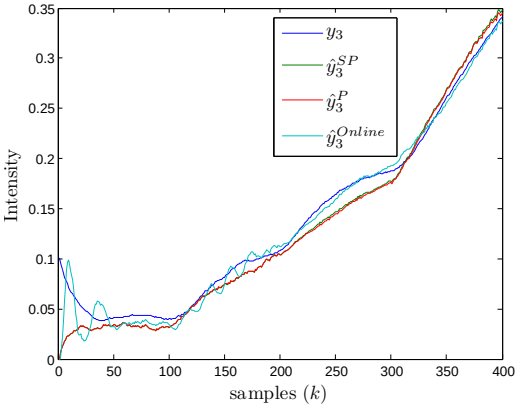
(a) $\hat{x}$ along time



(b) y_1 along time; $MSE^{online} = 0.2648e^{-3}$; $MSE^{SP} = 0.3979e^{-3}$; $MSE^{P} = 0.4036e^{-3}$

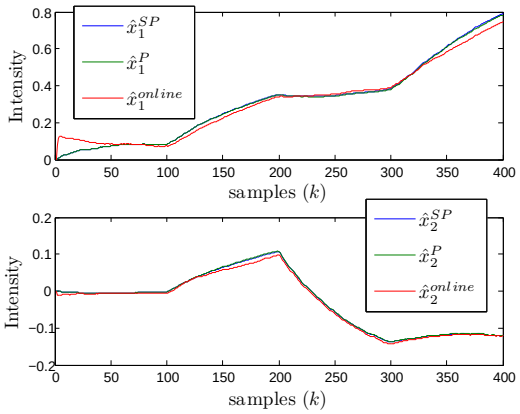


(c) y_2 along time; $MSE^{online} = 0.3968e^{-3}$; $MSE^{SP} = 0.5885e^{-3}$; $MSE^{P} = 0.5986e^{-3}$

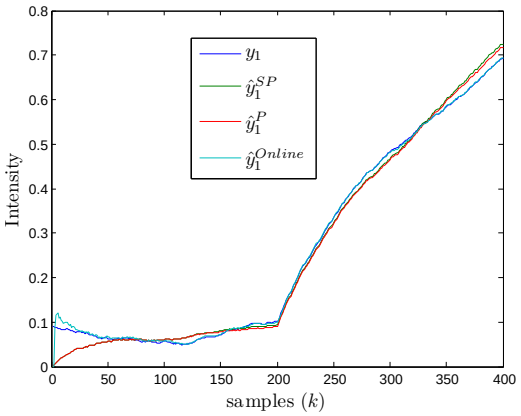


(d) y_3 along time; $MSE^{online} = 0.1522e^{-3}$; $MSE^{SP} = 0.2636e^{-3}$; $MSE^{P} = 0.2693e^{-3}$

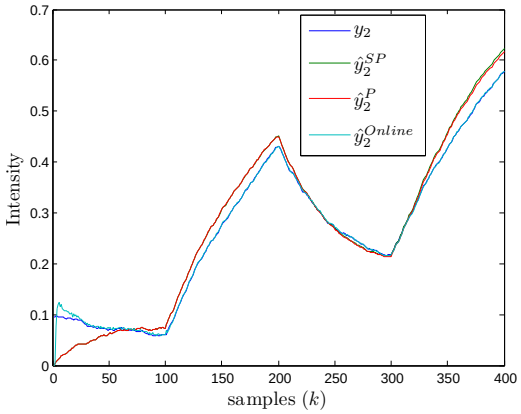
Figure 5.17: RNFMS response with optimized consequents



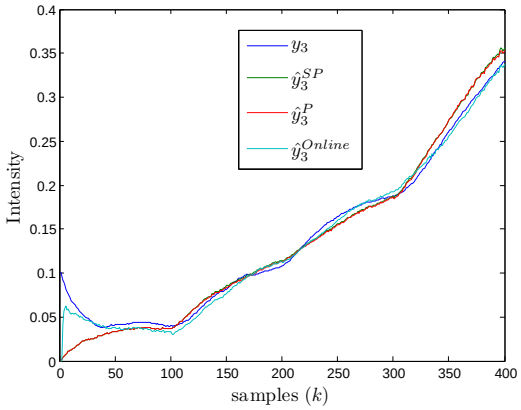
(a) $\hat{x}$ along time



(b) y_1 along time; $MSE^{online} = 0.5998e{-}4; MSE^{SP} = 0.4246e^{-3}; MSE^{P} = 0.4029e^{-3}$

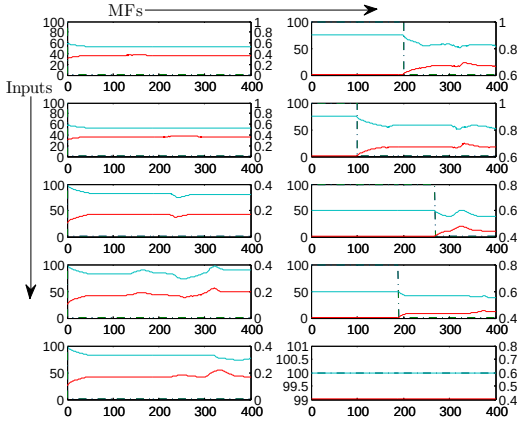


(c) y_2 along time; $MSE^{online} = 0.5809e{-}4; MSE^{SP} = 0.7828e^{-3}; MSE^{P} = 0.7328e^{-3}$

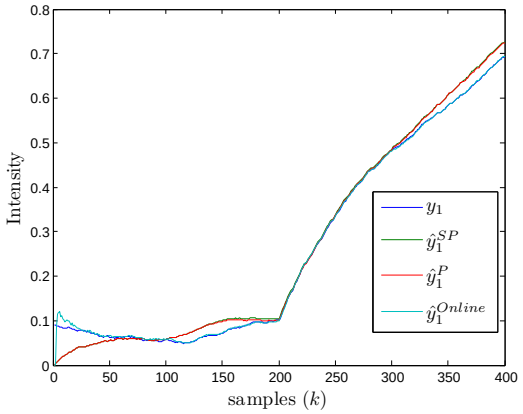


(d) y_3 along time; $MSE^{online} = 0.8975e{-}4; MSE^{SP} = 0.2905e^{-3}; MSE^{P} = 0.2876e^{-3}$

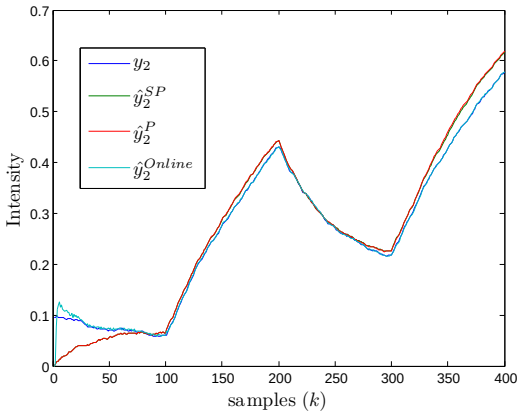
Figure 5.18: RNFMS response with optimized consequents and states



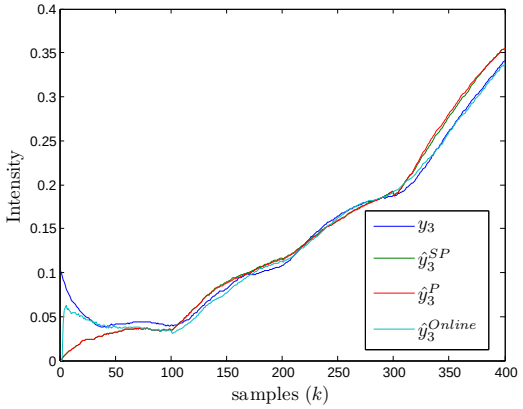
(a) MFs parameters along time



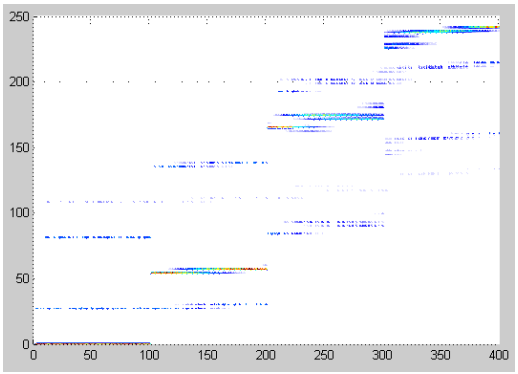
(b) y_1 along time; $MSE^{online} = 0.5920e{-}4$; $MSE^{SP} = 0.5029e^{-3}$; $MSE^{P} = 0.4719e^{-3}$



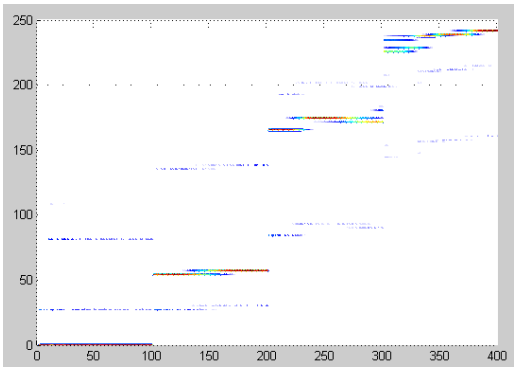
(c) y_2 along time; $MSE^{online} = 0.5785e{-}4$; $MSE^{SP} = 0.6191e^{-3}$; $MSE^{P} = 0.6594e^{-3}$



(d) y_3 along time; $MSE^{online} = 0.8263e{-}4$; $MSE^{SP} = 0.3103e^{-3}$; $MSE^{P} = 0.3270e^{-3}$

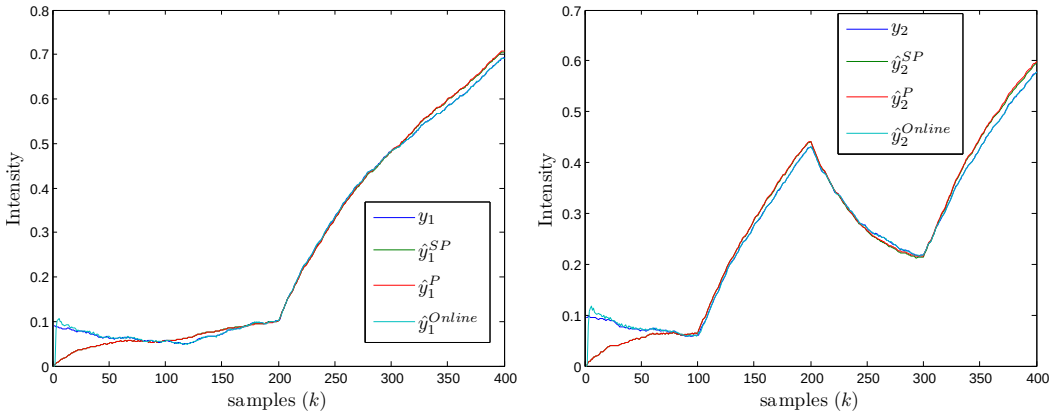


(e) Rule weights before MF optimization



(f) Rule weights after MF optimization

Figure 5.19: RNFMS response with optimized consequents, states and MFs, using all plant sensors



(a) y_1 along time; $MSE^{online} = 0.4622{-}4$; $MSE^{SP} = 0.3553^{-3}$; $MSE^{P} = 0.3642^{-3}$

(b) y_2 along time; $MSE^{online} = 0.5107{-}4$; $MSE^{SP} = 0.4759^{-3}$; $MSE^{P} = 0.4977^{-3}$

Figure 5.20: RNFMS response with optimized consequents, states and MFs, without using sensor T3

5.3.2 Online Identification results

Online identification comprises both model fitting and controller optimization. Firstly a default local controller must be defined as a starting point, apply it to all local models and then start with the identification process. Based on offline input-output data, all controller parameters were set to zero and then optimization algorithm Table A.5 was applied using the following parametrization:

$$\alpha_{Ct}^{P} = 1e^{1} \qquad \alpha_{Ct}^{R} = 1e^{-2} \qquad \alpha_{Ct;u}^{E} = 1e^{-2} \tag{5.10}$$

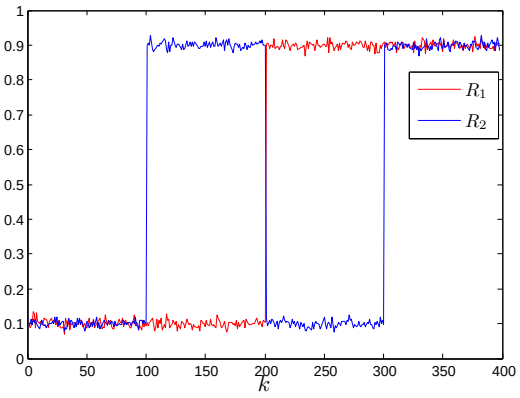
$$\alpha_{Ct;c}^{E} = 1e^{2} \qquad \alpha_{Ct;u}^{e} = 0.05 \qquad \alpha_{Ct;c}^{e} = 0.01 \qquad \alpha_{Ct}^{r} = 0.05 \tag{5.11}$$

Because estimation data was collected in an open loop, a virtual reference needs to be created since controller equation depends on it. Against what was done in previous process, a virtual reference based on offline control actions was considered. Only model outputs of tank one and two will be considered during controller optimization, since only them reflect predicted control actions during algorithm computation. Offline controller dynamics are represented in Figure 5.21. After obtaining a default local controller and spreading it to all RNFCM consequents, the closed loop online identification can be started. The initial parameters for all algorithms were considered equal to the ones already defined previously, and a new set of reference points around membership centers were created.

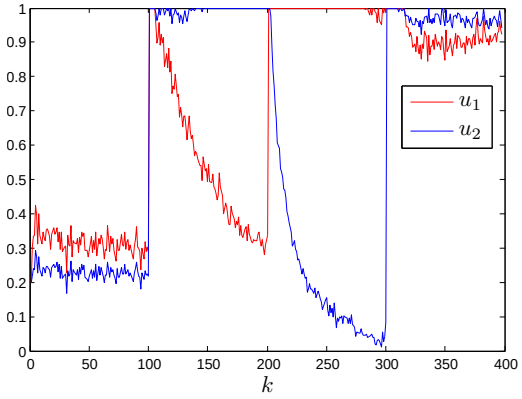
Several simulations will be done to validate both controller and model optimization. First it is considered the use of the three tanks with and without MF optimization. Also both approaches of considering firstly control and then reference in rules premises will be followed. The impact of considering all tanks against only tanks one and two will also be analysed.

First experiment (Exp.1) Figure 5.22 and Figure 5.23 shows a poor controller performance with matrix W^K strongly unstable, also model errors where significantly increased in comparison with offline solution. Next experiment (Exp.2) will not include MF optimization, where purpose is to find some negative correlation in controller behaviour when using it. From Figure 5.24 and Figure 5.25 it is noticed an improvement of controller state gain matrix and a stabilization in control actions, also the error between output and reference was decreased. It can be concluded this way, that membership parameters optimization can lead system to a non stable rule switching behaviour, having more impact if using controller in premises other than reference, since latest is usually static for a predefined time window. Several experiments where done in order to create a comparison between several possibilities. To simplify result presentation, six distinct experiments will be conducted whose results will be compared in Figure 5.26:

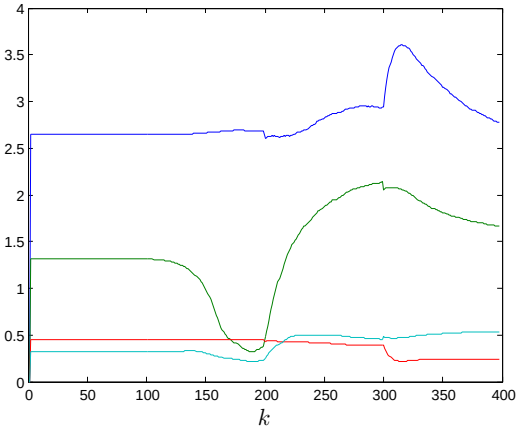
1. RNFCS with controller in premise, tank three and RNFMS with MF optimization;



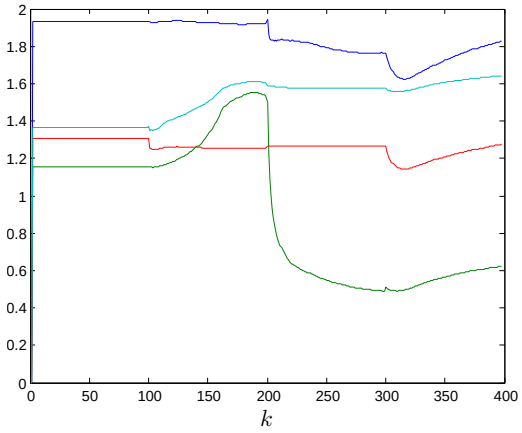
(a) Virtual reference



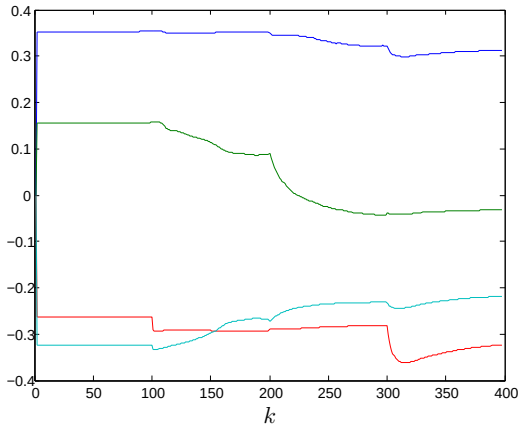
(b) Controller offline response



(c) Gain W^K evolution

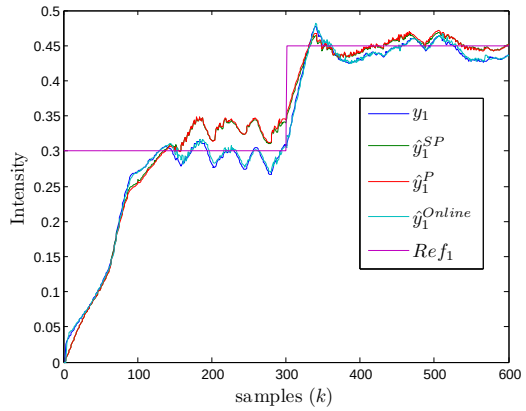


(d) Gain W^R evolution

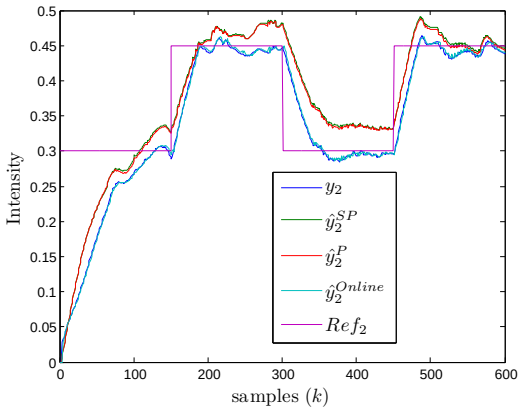


(e) Gain W^P evolution

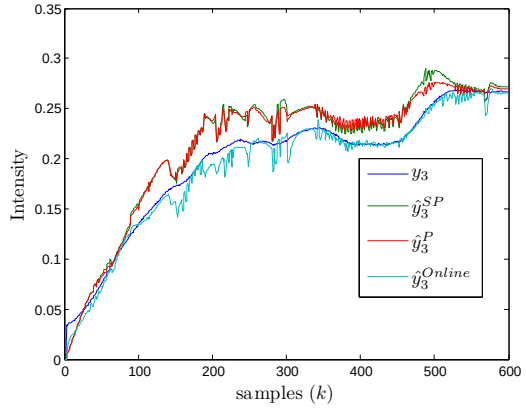
Figure 5.21: Offline local controller optimization



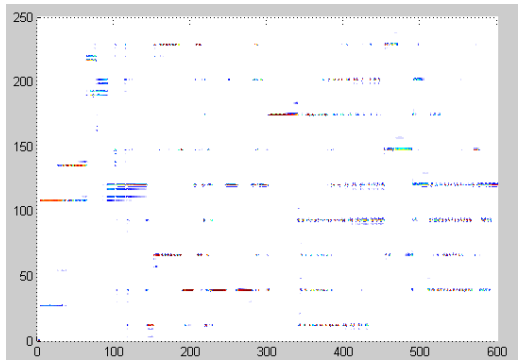
(a) y_1 along time; $MSE^{online} = 0.2026e^{-5}$; $MSE^{SP} = 0.9372e^{-4}$; $MSE^{P} = 0.1934e^{-3}$



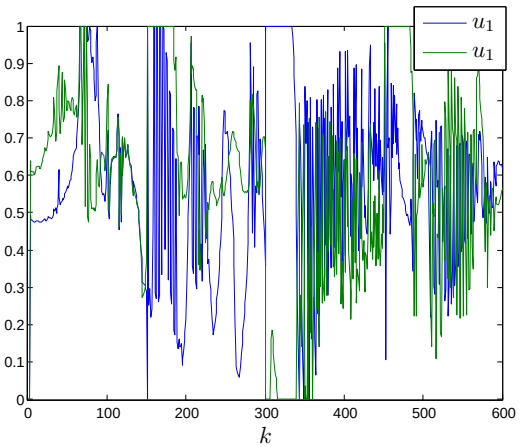
(b) y_2 along time; $MSE^{online} = 0.1655e^{-5}$; $MSE^{SP} = 0.7575e^{-4}$; $MSE^{P} = 0.2270e^{-3}$



(c) y_3 along time; $MSE^{online} = 0.1655e^{-5}$; $MSE^{SP} = 0.7575e^{-4}$; $MSE^{P} = 0.2270e^{-3}$

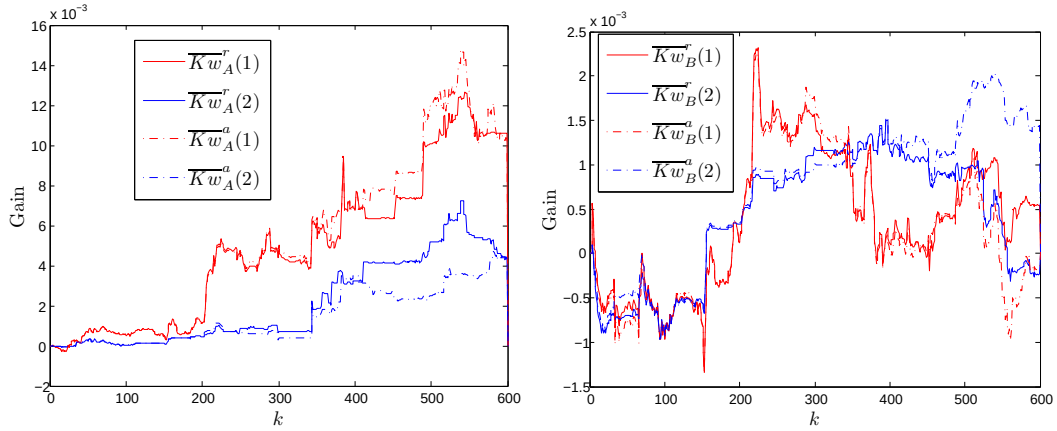


(d) Rule degrees along time



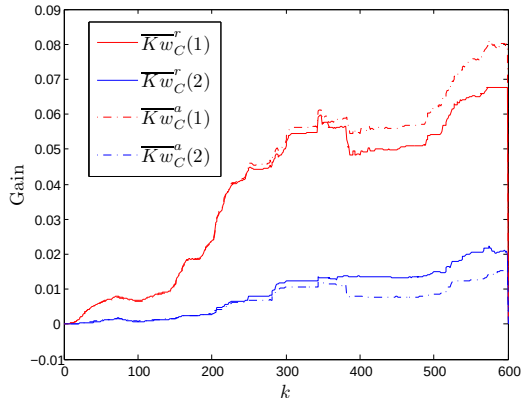
(e) u_k along time

Figure 5.22: RNFCS with controller in premise, tank three and RNFMS with MF optimization (Exp.1)

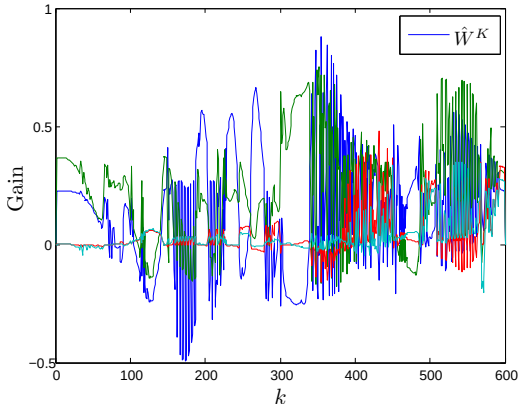


(a) Kw^A evolution

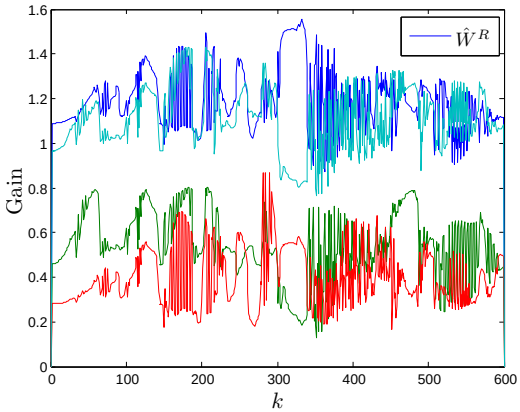
(b) Kw^B evolution



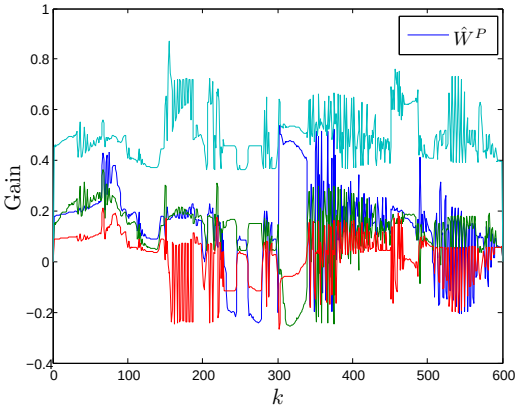
(c) Kw^C evolution



(d) W^K evolution

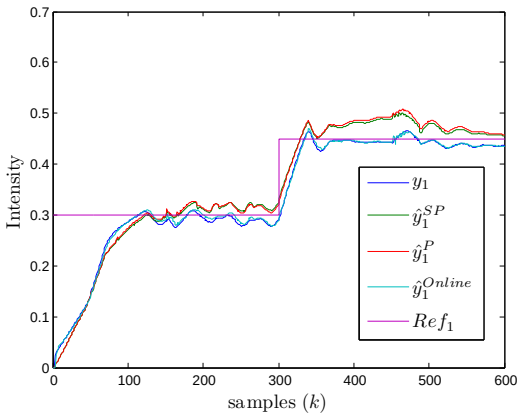


(e) W^R evolution

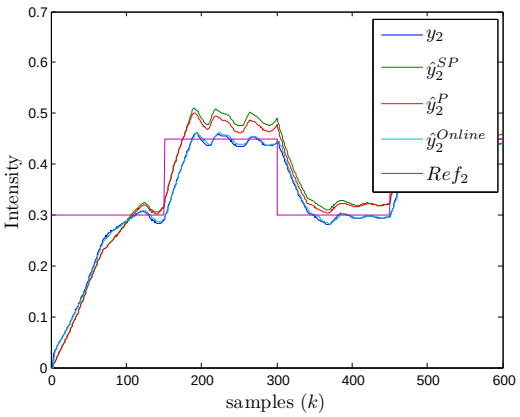


(f) W^P evolution

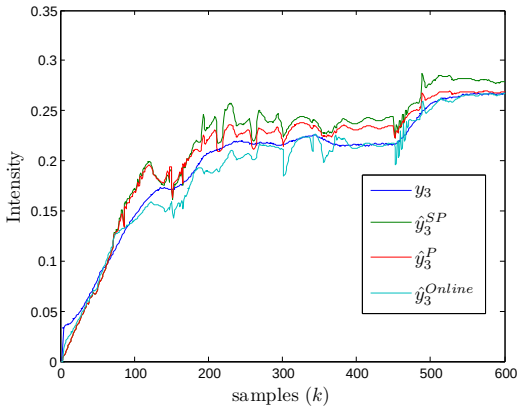
Figure 5.23: Online RNFMS and RNFCM gain evolution. RNFCS with controller in premise, tank three and RNFMS with MF optimization (Exp.1)



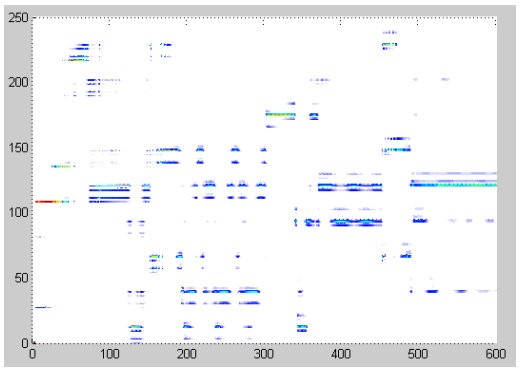
(a) y_1 along time; $MSE^{online} = 0.2026e^{-5}; MSE^{SP} = 0.9372e^{-4}; MSE^{P} = 0.1934e^{-3}$



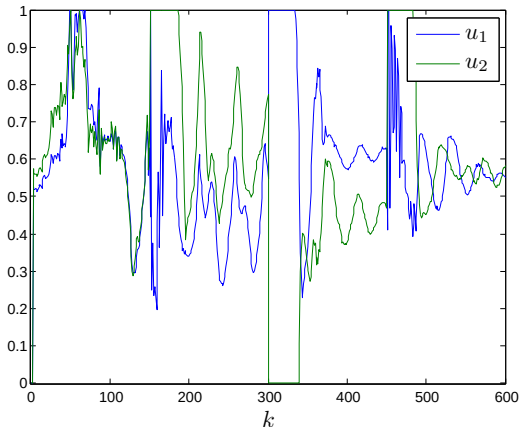
(b) y_2 along time; $MSE^{online} = 0.1655e^{-5}; MSE^{SP} = 0.7575e^{-4}; MSE^{P} = 0.2270e^{-3}$



(c) y_3 along time; $MSE^{online} = 0.1655e^{-5}; MSE^{SP} = 0.7575e^{-4}; MSE^{P} = 0.2270e^{-3}$

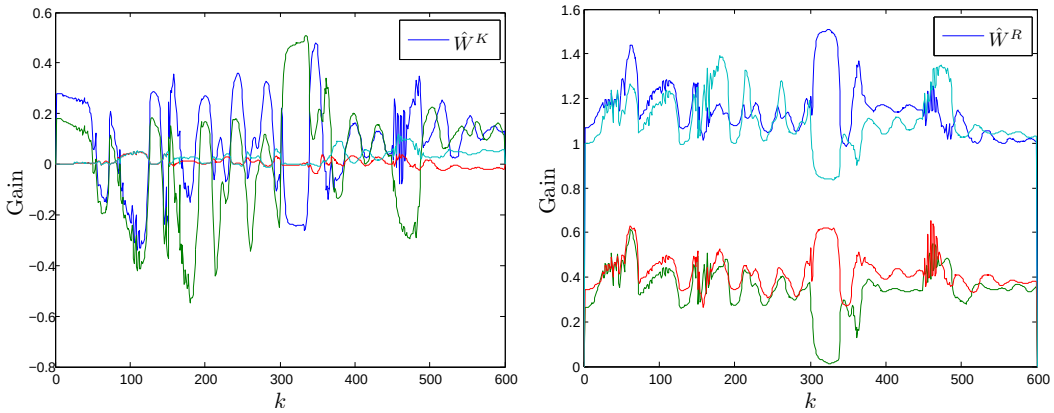


(d) Rule degrees along time

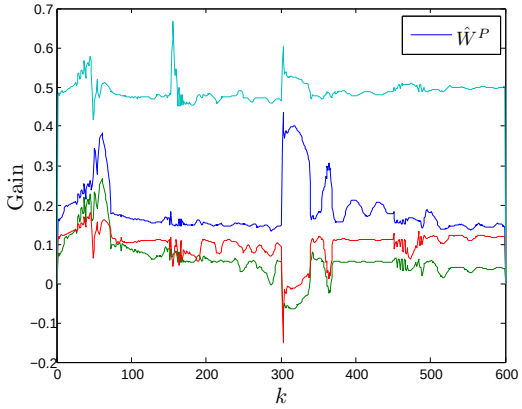


(e) u_k along time

Figure 5.24: RNFCS with controller in premise, tank three and RNFMS without MF optimization (Exp.2)



(a) W^K evolution (b) W^R evolution



(c) W^P evolution

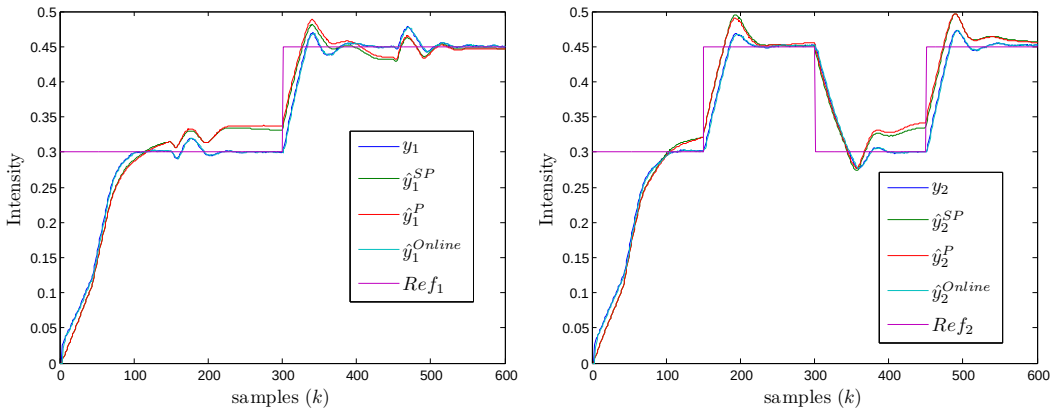
Figure 5.25: Online RNFMS and RNFCS gain evolution. RNFCS with controller in premise, tank three and RNFMS without MF optimization (Exp.2)

Exp. Nº	Using Ref	Using control	3º tank	Using MF	RMSE online	RMSE Series	RMSE Parallel	RMSE Ref	RMSE Prediction	RMSE Series	RMSE Parallel	RMSE Ref	RMSE online	RMSE Series	RMSE Parallel
1	no	yes	yes	yes	0.1500e-4	0.0005	0.5137e-3	0.0064	0.1112e-4	0.0011	0.9507e-3	0.0057	0.6649e-4	0.0005	0.9507e-3
2	no	yes	yes	no	0.1464e-4	0.6908e-3	0.7950e-3	0.0054	0.1150e-4	0.9014e-3	0.6158-3	0.0058	0.9918e-4	0.4188e-3	0.1916e-3
3	yes	no	yes	yes	0.1246e-4	0.7946e-3	0.7491e-3	0.0068	0.0913e-4	0.8062e-3	0.5991e-3	0.0060	0.6951e-4	0.4685e-3	0.3105e-3
4	yes	no	yes	no	0.1166e-4	0.0001	0.2755e-3	0.0055	0.1028e-4	0.0012	0.7800e-3	0.0059	0.7579e-4	0.0002	0.0905e-3
5	yes	no	no	yes	0.6213e-5	0.6383e-3	0.6834e-3	0.0064	0.7830e-5	0.4475e-3	0.4364e-3	0.0063			
6	yes	no	no	no	0.5931e-5	0.3181e-3	0.3900e-3	0.0052	0.8035e-5	0.3209e-3	0.3770e-3	0.0060			

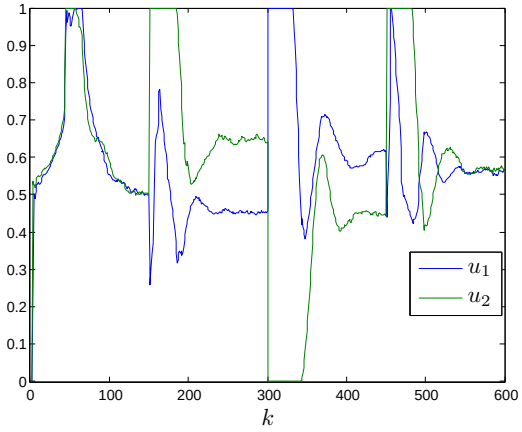
Figure 5.26: Experimental results comparison

2. RNFCS with controller in premise, tank three and RNFMS without MF optimization;
3. RNFCS with reference in premise, tank three and RNFMS with MF optimization;
4. RNFCS with reference in premise, tank three and RNFMS without MF optimization;
5. RNFCS with reference in premise, no tank three and RNFMS with MF optimization;
6. RNFCS with reference in premise, no tank three and RNFMS without MF optimization;

From table Figure 5.26 it can be concluded that using controller in premises drives worst results and considering tank three will increase static and model errors. Also it is noticed that controller performance is dependent on model performance. System response for the best solution is depicted in Figure 5.27, where it is observed a static error convergence to zero. Also the overshot increase was expected according to analysis done in previous process, since it was not enforced a reset on covariance $R_{Ct}^{e-}(k)$ when rule subset changes. Presented work will finish the experimentation with a simulation according to experimental layout six, and accounting for system failures. In a first approach it will be introduced a failure on tank one by half opening its water drain valve, being expected a system online prediction convergence and a static error evolution towards zero. Failure one will be introduce in sample 150 and will remain until end of experiment. Next system will be reverted to its original layout and then, it is introduced a new failure on tank two by half closing its connection to tank three. This latest failure will be introduced in sample 150 and then be removed in sample 250, where it is expected both for predicted and static errors a convergence to zero after failure insertion and removal. Observing closed loop response under failures Figure 5.28 the expected behaviour was achieved. In both situations, the static and predicted error converged towards zero although, for the first failure the static error convergence seemed to stop. Offline model errors where completely degraded, the explanation dues the fact that not all rules were affected with the new failure dynamics, and for those who have changed, when system was restored remained with the wrong plant failure behaviour. This conclusion was only rational having as pillars all acquired empirical understanding on proposed solution dynamics.

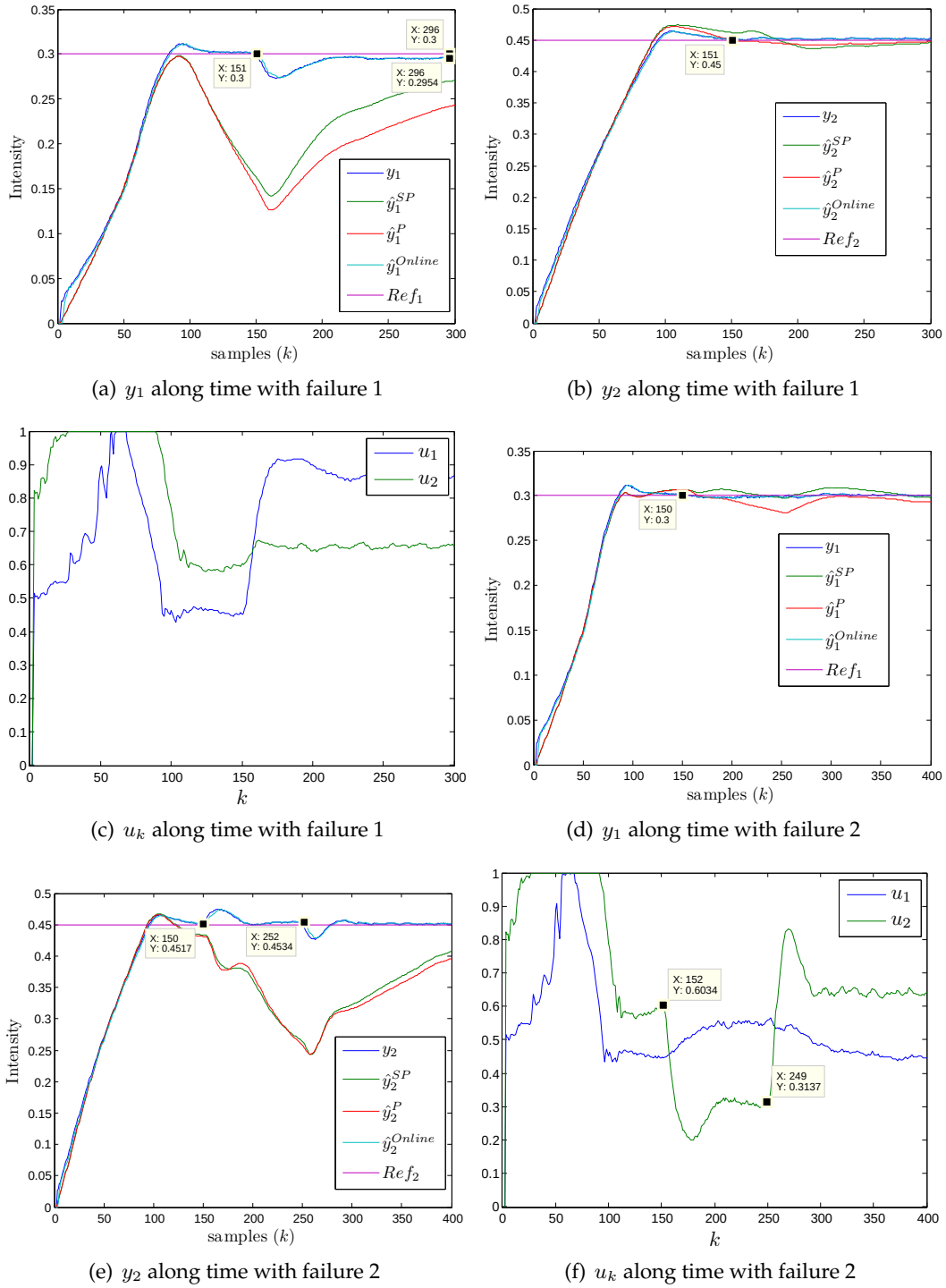


(a) y_1 along time (b) y_2 along time



(c) u_k along time

Figure 5.27: RNFCS with reference in premise, no tank three and RNFMS without MF optimization (Exp.6)



(a) y_1 along time with failure 1

(b) y_2 along time with failure 1

(c) u_k along time with failure 1

(d) y_1 along time with failure 2

(e) y_2 along time with failure 2

(f) u_k along time with failure 2

Figure 5.28: Online closed loop response with plant failures. RNFCS with reference in premise, no tank three and RNFMS without MF optimization

5.4 Conclusion

Current chapter had successfully validated all major points proposed in this thesis, with the exception of rule degrees and input weights which algorithm could not be validated. However, from membership functions and from a basic reasoning, there are more disadvantages other than advantages when optimizing rule premise parameters regarding fuzzy sets shape, rule degrees and input weights. Changing rule coverage, will also change subspace partitioning whose local models might have been already optimized for the old partitions. The subspace change for a given set of rules kept constant for a given time window, will result in a wide number of local models, proportional to the input vector dimension, becoming inaccurate. However, it was seen that increasing a rule weight will increase its local model convergence speed lowering the damage on its neighbours. It remains for a future investigations a deeper analysis of advantages and disadvantages when optimizing premise parameters. Focusing on controller, it was observed during experiments the need to consider during optimization a correlation between all rules of a given subset. Otherwise, controller would become unstable due to a constant gain switch resultant from rule subset changes. Through proposed method such effect was reduced but not fully eliminated being possible future occurrences. A study on theories regarding controller switching and dwell times might be useful for future algorithm enhancements and stability analysis.

Model increased complexity due to an excessive number of states, can increase the probability of zero pole cancellation decreasing controllability and observability. It is advised a minimal number of state on which an acceptable estimation error is achieved. Also, the correlation of RNFMS output layer parameters with the hidden layers parameters, might also increase the non observability and divergence if a wrong gain is set to W^C overriding previous layers dynamics.

6

Global Conclusions and Further Research

With the need to control systems facing a complexity increase, white box modelling is becoming to complex and sometimes impossible. The growth of grey box modelling theories focusing on building a theoretical model as accurate as possible considering only process inputs and outputs, allows a complete new wave of control and identification methods. Although grey box modelling will never be as accurate as white box techniques, for instance input and output data usually contains noise which creates drifts on model dynamics. Also if process output is correlated with some form of non Gaussian noise, it will produce undesired model biasing and wrong dynamics to be identified. Also a model divergence or biasing will lead to a model based controller divergence and consequently to a static error increase. The excessive model fitting will also increase the effects of model and noise correlation, leading to so called over-fitting problems.

6.1 Global Conclusions

It is expected from this thesis not a precise control solution but a new promising method for online adaptative control and identification capable of self tuning. From presented proposals it is expected the provisioning of new vectors for recursive fuzzy modelling and control, including a proof concept regarding Unscented Kalman Filter optimization capabilities when using a RNFS architecture.

Better results could be achieved by fine tuning algorithm parameters, meanwhile to find

a precise and robust solution based not only on empirical results but also on mathematical facts, a deep analysis of a global solution integration between UKF, recursive neuro fuzzy and state space methodology needs to be done. Also stability and convergence over long term simulations could not be assured, since only a small number of samples were considered.

One of the major drawback regarding UKF robustness relates to the handling of constraints. For cases where the model parameter solution that minimises a constrained cost function has a gradient towards a boundary, proposed algorithm will fail by constantly moving parameters away from the boundary region. New methods need to be investigated in order to better handle constraints without making use of real-time quadratic problems.

Achieved architecture and its solution followed a step by step approach, acquiring shape throughout the state of the art investigation. Other solutions where on sight for instance, it was tried to combine genetic algorithms for consequent optimization, where it was realized a constraint regarding high processing times. Also, it was under and initial scope the use of spline shaped membership function and clustering methods for fuzzy sets optimization. Although such theories would lead to an increase of system parameters to be optimized during a real time basis. Regarding presented algorithms precisely UKF, it was noticed a lack of stability analysis for instance the system behaviour when facing non Gaussian noise.

To finalize, regardless non sleeping hours and for some times the feeling of "no possible solution", it was a great pleasure to work with all my colleagues, sharing their commitment and belief to create something innovative and new.

6.2 Further Research

The aim of proposed solution was to obtain a model whose statistical properties best fit process statistical properties. Presented solution, apart from model fitting, should be capable of handling the following problems:

- Patern recognition and classification, which is a well known capability from neuro networks;
- Fault detection and diagnosis by creating rules to monitor model parameter evolution;
- Multi layer reasoning, by means of using low level model outputs to feed high level models inputs.

Apart from new applications, some work was left undone:

- Rules and input weights optimization analysis;
- Stability analysis analytical study;
- Study regarding impact of forgetting factors in UKF convergence;
- Long term simulation stability analysis;
- Use of different inference mechanisms, fuzzy relations and defuzzification methods;
- Improvement of constraint handling during constrained parameter optimization;
- Analysis regarding the use of different membership functions;
- Methods to discard useless rules from rule subset, turning real-time inference faster;
- Study the advantages of using local states instead of global states, i.e each local model has its own state vector.

Above listed items relies only theoretical ideas raised during implementation and brainstorming sessions. Although, some might open new windows towards computer reasoning enhancements.

Bibliography

[1] Lotfi A. Zadeh. Fuzzy Sets. *Information and Control. 8*, pages pp 338–353, 1965.

[2] Etienne E. Kerre (Eds.) Paul P. Wang, Da Ruan. *Studies in Fuzziness and Soft Computing*, volume 215. Springer, 2007.

[3] Bo Yuan George J. Klir. *Fzzy Sets and Fuzzy Logic - Theory and Applications*. Prentice Hall, 1995.

[4] Leszek Rutkowski. *Structures, Learning And Performance Evaluation, Flexible Neuro-Fuzzy Systems*. Kluwer Academic Publishers, 2004.

[5] Timothy J. Ross. *Fuzzy Logic With Engeneering Applications*. John Wiley and Sons, INC, 2004.

[6] MathWorks. Fuzzy Logic Toolbox User's Guide. 1995-2008.

[7] William Siler and James J. Buckley. *Fuzzy Expert Systems and Fuzzy Reasoning*. John Wiley and Sons, INC, 2005.

[8] T. Takagi and M. Sugeno. Fuzzy Identification of Systems and its Applications to modeling and control, vol. smc-15. *IEEE Trans. Syst., Man, Cybern.*, pages 116–132, 1985.

[9] T. Takagi and M. Sugeno. A New Tsk Fuzzy Modeling approach. *IEEE Trans. Syst., Man, Cybern.*,, pages 116–132, 1985.

[10] J. C. Fodor. On Fuzzy Implications Operators, fuzzy sets and fuzzy systems, vol.42. *Proc. of the IEEE*, pages 293–300, 1991.

[11] J. M. Mendel. Fuzzy Locic Systems for Engineering: a tutorial. *Proc. of the IEEE*, pages 83(3):345–377, 1995.

[12] J. M. Mendel. *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*. Prentice All PTR, 2001.

[13] Michio Sugeno. An introductory survey of fuzzy control. *Inf. Sci.*, pages 59–83, 1985.

[14] I. Burhan Türksen Asli Celikyilmaz. *Studies in Fuzziness and Soft Computing*, volume 240. Springer, 2009.

[15] Yucai Zhu. *Multivariate System Identification for Process Control*. Elsevier Science and Technology Books, 2001.

[16] Karl-Johan Astrom and Torsten Bohlin. Numerical identification of linear dynamic systems from normal operating records. *IFAC Syposium on Self-Adaptive Systems, Teddington, England.*, 1965.

[17] Shuqing Wang Yin Wang, Gang Rong. Hybrid fuzzy modeling of chemical processes. *Fuzzy Sets and Systems*, 130:265 – 275, 2002.

[18] J.A.M. Felippe de Souza L. Schnitman and T. Yoneyama. Takagi-sugeno-kang fuzzy structures in dynamic system modeling. *Instituto Tecnológico de Aeronáutica*, 2001.

[19] *Indirect model reference adaptive fuzzy control of dynamic fuzzy-state space model*, volume 148, 2001.

[20] Neville W. Rees Shu-Guang Cao and Gang Feng. Analysis and design of fuzzy control systems using dynamic fuzzy-state space models. *IEEE TRANSACTIONS ON FUZZY SYSTEMS*, 7(2), 1999.

[21] Trung Tat Pham Guanrong Chen. *Introduction to Fuzzy Sets, Fuzzy Logic, and Fzzy Control Systems*. CRC Press LLC, 2001.

[22] *Augmented Stable Fuzzy Control for Flexible Robotic Arm Using LMI Approach and Neuro-Fuzzy State Space Modeling*, volume 55, 2008.

[23] Sang Chul Ahn Won Chul Kim and Wook Hyun Kwon. Stability analysis and stabilization of fuzzy state space models. *Fuzzy Sets and Systems*, 71:131 – 142, 1995.

[24] David Kriesel. *A Brief Introduction to Neural Networks*. 2007. available at http://www.dkriesel.com.

[25] Mario Köppen, Nikola K. Kasabov, and George G. Coghill, editors. *Advances in Neuro-Information Processing, 15th International Conference, ICONIP 2008, Auckland, New Zealand, November 25-28, 2008, Revised Selected Papers, Part II*, volume 5507 of *Lecture Notes in Computer Science*. Springer, 2009.

[26] M. I. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. *Cognitive Science*, 1986.

[27] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179 – 211, 1990.

[28] Ben Kröse, Patrick van der Smagt, and Patrick Smagt. An introduction to neural networks, 1996.

[29] J.-S. R. Jang. Anfis: adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23:665–685, 1993.

[30] Gang Feng Chu Kwong Chak and Jian Ma. An adaptative fuzzy neural network for mimo system model approximation in high-dimensional spaces. *IEEE Transactions on Systems, Man, and Cybernetics - PartB*, 28(3), 1998.

[31] Takeshi Furuhashi Shin-ichi Horikawa and Yoshiki Uchikawa. On fuzzy modeling using neural networks with the back-propagation algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(5), 1992.

[32] Jie Zhang and A. Julian Morris. Recurrent neuro-fuzzy networks for nonlinear process modeling. *IEEE TRANSACTIONS ON NEURAL NETWORKS*, 10(2), 1999.

[33] Wen Yu. State-space recurrent fuzzy neural networks for nonlinear system identification. *Neural Processing Letters*, 2005.

[34] *Identification and Control of Dynamic Systems Using Recurrent Fuzzy Neural Networks*, volume 8, 2000.

[35] *A TSK-Type Recurrent Fuzzy Network for Dynamic Systems Processing by Neural Network and Genetic Algorithms*, volume 10, 2002.

[36] Trung Tat Pham Guanrong Chen. *Introduction to Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems*. CRC Press, 2001.

[37] Dan Simon. *Optimal State Estimation*. Wiley-Interscience, 2006.

[38] Hugh F. Durrant-Whyte Simon Julier, Jeffrey Uhlmann. A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators. 2008.

[39] Angus P. Andrews Mohinder S. Grewal. *Kalman Filtering Theory and Practice Using Matlab*. Wiley-Interscience, 2008.

[40] Simon Haykin. *Kalman Filtering and Neural Networks*. Wiley-Interscience, 2008.

[41] A.H. Jazwinski. *Stochastic Processes and Filtering Theory*. New York: Academy Press, 1970.

[42] P.S. Maybeck. *Stochastic Models, Estimation and Control, Vol. 2*. New York: Academy Press, 1982.

[43] Simon Julier and Jeffrey Uhlmann. Unscented Filtering and Nonlinear Estimation. 2004.

[44] Jeffrey K. Uhlmann Simon J. Julier and Hugh F. Durrant-Whyte. A New Approach for Filtering Non Linear Systems. 1995.

[45] Rudolph van der Merwe. Sigma-point Kalman Filters for Probabilistic Inference in Dynamic State-space Models. 2004.

[46] T.S. Schei S. Kolas, B.A. Foss. Constrained Nonlinear State Estimation Based on Ukf Approach. 2009.

[47] J. H. Gove and D. Y. Hollinger. Aplication of a Dual Unscented Kalman Filter for Sumltaneous State and Prameter Estimation in Problems of Surface-atmosphere Exchange. 2006.

[48] Simon Haykin. *Kalman Filtering and Neural Networks*. John Wiley and Sons, INC, 2001.

[49] H. Duarte-Ramos P. Gil, J. Henriques and A. Dourado. Unscented Kalman Filter in Adaptative Neural Mode-based Predictive Control. 2002.

[50] Charles F. Van Loan Gene H.Golub. *Matrix Computations, Third Edition*. The Johns Hopkins University Press, 1996.

[51] Rudolph van der Merwe and Eric A. Wan. The Square-root Unscented Kalman Filter for state and parameter-estimation. 2001.

[52] H. Ghanbarpour Asl and S. H. Pourtakdoust. Ud Covariance Factorization for unscented kalman filter using sequential measurements update. 2007.

[53] D. Simon and Tien Li Chia. Kalman filtering with state equality constraints. *Aerospace and Electronic Systems, IEEE Transactions on*, 38(1):128 –136, jan 2002.

[54] R Kandepu, B Foss, and L Imsland. Applying the unscented kalman filter for nonlinear state estimation. *Journal of Process Control*, 18(7-8):753–768, 2008.

[55] R. Kandepu, L. Imsland, and B.A. Foss. Constrained state estimation using the unscented kalman filter. In *Control and Automation, 2008 16th Mediterranean Conference on*, pages 1453 –1458, june 2008.

[56] B. Teixeira, L. Torres, L.A. Aguirre, and D.S. Bernstein. Unscented filtering for interval-constrained nonlinear systems. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 5116 –5121, dec. 2008.

[57] Pramod Vachhani, Shankar Narasimhan, and Raghunathan Rengaswamy. Robust and reliable estimation via unscented recursive nonlinear dynamic data reconciliation. *Journal of Process Control*, 16(10):1075 – 1086, 2006.

[58] D. Simon. Kalman filtering with state constraints: a survey of linear and nonlinear algorithms. *Control Theory Applications, IET*, 4(8):1303 –1318, august 2010.

[59] Dan Simon and Donald L. Simon. Aircraft turbofan engine health estimation using constrained kalman filtering. 2003.

[60] M. Isabel Ribeiro. Gaussian Probability Density functions: Properties and error characterization. *Institute for Systems and Robotics Instituto Superior Técnico*, 2004.

[61] Paul Michael Newman. C4b - Mobile Robots. 2003.

[62] Fernando Jorge Martins Almeida Costa. *Controlo Supervisionado*. Faculdade De Ciências e Tecnologia - Universidade Nova de Lisboa; Tese de Mestrado, 2011.

A

Proposed Algorithms

A.1 RNFMS consequents optimization algorithm

Initialization:

$$\begin{aligned}
&P_A(0) = \alpha_\psi^P eye\,(n_x * n_x, n_x * n_x, C) \quad R_A^r(0) = \alpha_\psi^R eye\,(n_x * n_x, n_x * n_x, C)\\
&R_\psi^e(0) = \alpha_\psi^E eye\,(m, m, C)\\
&P_B(0) = \alpha_\psi^P eye\,(n_x * n_u, n_x * n_u, C) \quad R_B^r(0) = \alpha_\psi^R eye\,(n_x * n_u, n_x * n_u, C)\\
&P_C(0) = \alpha_\psi^P eye\,(m * nx, m * nx, C) \quad R_C^r(0) = \alpha_\psi^R eye\,(m * nx, m * nx, C)
\end{aligned} \tag{A.1}$$

For each iteration k:

- Get next i_k^μ and set $\varphi = [y(k-1)\ \ u(k-1)]$
- Compute all layer $M4$ neurons output $\overline{M}^\sigma = M_4\left(w_{i_k^\mu}^{\tau,agr}(k-1), w_{i_k^\mu}^{\mu-}(k-1), u(k-1), i_k^\mu\right)$
- For each rule j in i_k^μ

  Compute time-update equations:

$$\hat{w}^\psi(0) = RNFMS_{[A,B,C]}^{i_k^\mu}(k-1) \tag{A.2a}$$

$$\Rightarrow P_\psi^-(k) = blkdiag\left(\left[\begin{array}{ccc} P_A^{i_{j;k}^\mu}(k-1) & P_B^{i_{j;k}^\mu}(k-1) & P_C^{i_{j;k}^\mu}(k-1) \end{array}\right]\right) \tag{A.2b}$$

$$\Rightarrow R_\psi^{r-}(k) = blkdiag\left(\left[\begin{array}{ccc} R_A^{r,i_{j;k}^\mu}(k-1) & R_B^{r,i_{j;k}^\mu}(k-1) & R_C^{r,i_{j;k}^\mu}(k-1) \end{array}\right]\right) \tag{A.2c}$$

$$\Rightarrow R_\psi^{e-}(k) = R_\psi^{e,i_{j;k}^\mu}(k-1) \tag{A.2d}$$

- Calculate sigma-points:

$$\chi^{\psi-}(k) = \left[\hat{w}^{\psi-}(k) \quad \hat{w}^{\psi-}(k) + \gamma\sqrt{P_{\psi}^{-}(k)} \quad \hat{w}^{\psi-}(k) - \gamma\sqrt{P_{\psi}^{-}(k)}\right] \tag{A.3}$$

For $i = 1 : L$

$$x(k) = f\left(\chi_{A,B,i}^{\psi-}(k), \hat{x}^{+}(k-1), \varphi, \overline{M}^{\sigma}, i_k^{\mu}\right) \tag{A.4a}$$

$$Y_i = g\left(\chi_{C,i}^{\psi-}(k), x(k), \overline{M}^{\sigma}, i_k^{\mu}\right) \text{ (4.44)} \tag{A.4b}$$

- Compute the measurement-update equations:

$$\hat{w}^{\psi-}(k) = \sum_{i=1}^{L} W_i^{(m)} \chi_i^{\psi-} \tag{A.5a}$$

$$\hat{d}^{-}(k) = \sum_{i=1}^{L} W_i^{(m)} Y_j \tag{A.5b}$$

$$P_{\psi}^{-}(k) = \sum_{i=1}^{L} W_i^{(c)} \left(\hat{w}_i^{\psi-}(k) - \chi_i^{\psi-}(k)\right)\left(\hat{w}_i^{\psi-}(k) - \chi_i^{\psi-}(k)\right)^T + R_{\psi}^{r-}(k) \tag{A.5c}$$

$$P_{\tilde{d}_k} = \sum_{i=1}^{L} W_i^{(c)} \left(Y_i - \hat{d}^{-}(k)\right)\left(Y_i - \hat{d}^{-}(k)\right)^T + R_{\psi}^{e-}(k) \tag{A.5d}$$

$$P_{w_k d_k} = \sum_{i=1}^{L} W_i^{(c)} \left(\hat{w}_i^{\psi-}(k) - \chi_i^{\psi-}(k)\right)\left(Y_i - \hat{d}^{-}(k)\right)^T \tag{A.5e}$$

$$K(k) = P_{w_k d_k} P_{\tilde{d}_k}^{-1} \tag{A.5f}$$

$$\hat{w}^{\psi+}(k) = \hat{w}^{\psi-}(k) + K(k)\left(d(k) - \hat{d}^{-}(k)\right) \tag{A.5g}$$

$$P_{\psi}^{+}(k) = P^{\psi-}(k) + K(k) P_{\tilde{d}_k} K(k)^T \tag{A.5h}$$

$$R_{\psi}^{r+}(k) = \left(1 - \alpha_{\psi}^{r}\right) R_{\psi}^{r-}(k) + \alpha_{\psi}^{r} K(k)\left(d(k) - \hat{d}^{-}(k)\right)\left(d(k) - \hat{d}^{-}(k)\right)^T K(k)^T \tag{A.5i}$$

$$R_{\psi}^{r+}(k) = diag\left(diag\left(R_{\psi}^{r+}(k)\right)\right) \tag{A.5j}$$

$$\Delta = \left(d(k) - \hat{d}^{-}(k)\right)\left(d(k) - \hat{d}^{-}(k)\right)^T \tag{A.5k}$$

$$R_{\psi}^{e+}(k) = diag\left(diag\left(\left(1 - \alpha_{\psi}^{e}\right) R_{\psi}^{e-}(k) + \alpha_{\psi}^{e} K(k) K(k)^T \Delta\right)\right) \tag{A.5l}$$

- Update global Model parameters $RNFMS_{[A,B,C]}^{i_k^{\mu}}(k)$ with new local models.
- Save diagonals of local covariance matrices P_{ψ} and R_{ψ}^{r+} into global covariance matrices $P_{A,B,C}(k)$ and $R_{A,B,C}^{r}(k)$. In some cases considering only $diag(P_C)$ can return best results.

with $\gamma = \sqrt{N+\lambda}$, $L = 2N+1$, parameter dimension N, process noise covariance R^r, measurement noise R^e and a low pass filter poles α^e_ψ and α^r_ψ. Sigma points $W^{(m)}$ and $W^{(c)}$ are computed according (4.8).

Table A.1: RNFMS decoupled UKF consequents parameter estimation considering additive noise case. (system as in Equation 4.16)

A.2 RNFMS membership optimization algorithm

Initialization:

$$P_\mu(0) = \alpha^P_\mu ones\left(2, (P_{max}-1), nu+m\right) \quad R^r_\mu(0) = \alpha^R_\mu ones\left(2, (P_{max}-1), nu+m\right)$$
$$R^e_\mu(0) = \alpha^E_\mu ones\left(m, m\right) \tag{A.6}$$

For each iteration k:

- Compute matrix M^μ containing the rules subset membership indexes
- Get next i^μ_k and set $\varphi = [y(k-1) \;\; u(k-1)]$
- Get lb and ub bounding matrices for memberships M^μ
- Compute time-update equations:

$$\hat{\overline{w}}^\mu(0) = w^{\mu, M^\mu}(k-1) \tag{A.7a}$$
$$\textbf{if } i^\mu_k \neq i^\mu_{k-1} \tag{A.7b}$$
$$\Rightarrow \overline{P}^-_\mu(k) = diag\left(P^{M^\mu}_\mu(k-1)\right) \tag{A.7c}$$
$$\Rightarrow \overline{R}^{r-}_\mu(k) = diag\left(R^{r, M^\mu}_\mu(k-1)\right) \tag{A.7d}$$
$$\Rightarrow R^{e-}_\mu(k) = R^{e+}_\mu(k-1) \tag{A.7e}$$
$$\textbf{else} \tag{A.7f}$$
$$\Rightarrow \overline{P}^-_\mu(k) = \overline{P}^+_\mu(k-1) \tag{A.7g}$$
$$\Rightarrow \overline{R}^{r-}_\mu(k) = \overline{R}^{r+}_\mu(k-1) \tag{A.7h}$$
$$\Rightarrow R^{e-}_\mu(k) = R^{e+}_\mu(k-1) \tag{A.7i}$$

- Calculate sigma-points:

$$\chi^{\mu-}(k) = ICUT\left(lb, ub, \overline{P}^-_\mu(k)\right) \tag{A.8}$$

For $j = 1 : L$

$$\overline{M}^\sigma = M_4\left(w^{\tau, agr}_{i^\mu_k}(k-1), \chi^{\mu-}_j(k), \varphi, i^\mu_k\right) \tag{A.9a}$$
$$x\left(k\right) = f\left(w^\psi_{A,B}(k-1), \hat{x}^+(k-1), \varphi, \overline{M}^\sigma, i^\mu_k\right) \tag{A.9b}$$
$$Y_j = g\left(w^{\psi-}_C(k-1), x(k), \overline{M}^\sigma, i^\mu_k\right) \text{ (4.44)} \tag{A.9c}$$

- Compute the measurement-update equations:

$$\hat{\overline{w}}^{\mu-}(k) = \sum_{i=1}^{L} W_i^{(m)} \chi_i^{\mu-} \tag{A.10a}$$

$$\hat{d}^{-}(k) = \sum_{i=1}^{L} W_i^{(m)} Y_j \tag{A.10b}$$

$$\overline{P}_{\mu}^{-}(k) = \sum_{i=1}^{L} W_i^{(c)} \left(\hat{\overline{w}}_i^{\mu-}(k) - \chi_i^{\mu-}(k)\right) \left(\hat{\overline{w}}_i^{\mu-}(k) - \chi_i^{\mu-}(k)\right)^T + \overline{R}_{\mu}^{r-}(k) \tag{A.10c}$$

$$P_{\tilde{d}_k} = \sum_{i=1}^{L} W_i^{(c)} \left(Y_i - \hat{d}^{-}(k)\right) \left(Y_i - \hat{d}^{-}(k)\right)^T + R_{\mu}^{e-}(k) \tag{A.10d}$$

$$P_{w_k d_k} = \sum_{i=1}^{L} W_i^{(c)} \left(\hat{\overline{w}}_i^{\mu-}(k) - \chi_i^{\mu-}(k)\right) \left(Y_i - \hat{d}^{-}(k)\right)^T \tag{A.10e}$$

$$K(k) = P_{w_k d_k} P_{\tilde{d}_k}^{-1} \tag{A.10f}$$

$$\hat{\overline{w}}^{\mu+}(k) = max\left(min\left(\hat{\overline{w}}^{\mu-}(k) + K(k)\left(d(k) - \hat{d}^{-}(k)\right), ub\right), lb\right) \tag{A.10g}$$

$$\overline{P}_{\mu}^{+}(k) = \overline{P}^{\mu-}(k) + K(k) P_{\tilde{d}_k} K(k)^T \tag{A.10h}$$

$$\overline{R}_{\mu}^{r+}(k) = \left(1 - \alpha_{\mu}^{r}\right) \overline{R}_{\mu}^{r-}(k) + \alpha_{\mu}^{r} K(k) \left(d(k) - \hat{d}^{-}(k)\right) \left(d(k) - \hat{d}^{-}(k)\right)^T K(k)^T \tag{A.10i}$$

$$\overline{R}_{\mu}^{r+}(k) = diag\left(diag\left(\overline{R}_{\mu}^{r+}(k)\right)\right) \tag{A.10j}$$

$$\Delta = \left(d(k) - \hat{d}^{-}(k)\right) \left(d(k) - \hat{d}^{-}(k)\right)^T \tag{A.10k}$$

$$R_{\mu}^{e+}(k) = diag\left(diag\left(\left(1 - \alpha_{\mu}^{e}\right) R_{\mu}^{e-}(k) + \alpha_{\mu}^{e} K(k) K(k)^T \Delta\right)\right) \tag{A.10l}$$

- Update membership function parameters with $\hat{\overline{w}}^{\psi+}(k)$.
- Save diagonals of local covariance matrices $\overline{P}_{\mu}$ and $\overline{R}_{\mu}^{r+}$ into global covariance matrices $P_{\mu}(k)$ and $R_{\mu}^{r}(k)$

with $\gamma = \sqrt{N + \lambda}$, $L = 2N + 1$, parameter dimension N, process noise covariance R^r, measurement noise R^e and a low pass filter poles α_{μ}^{e} and α_{μ}^{r}. Sigma points $W^{(m)}$ and $W^{(c)}$ are computed as in Table 4.3

Table A.2: RNFMS decoupled CIUKF membership parameter optimization considering additive noise case. (system as in Equation 4.16)

A.3 RNFMS rule and input weights degree optimization algorithm

Initialization:

$$\begin{aligned} P_\tau(0) &= \alpha_\gamma^P ones\,(nu+m, C) & R_\tau^r(0) &= \alpha_\gamma^R ones\,(nu+m, C) \\ P_{agr}(0) &= \alpha_\gamma^P ones\,(C) & R_\gamma^r(0) &= \alpha_\gamma^R ones\,(C) \\ R_\gamma^e(0) &= \alpha_\gamma^E ones\,(m, m) \end{aligned} \tag{A.11}$$

For each iteration k:

- Get next i_k^μ and set $\varphi = [y(k-1)\ \ u(k-1)]$
- Get lb and ub bounding matrices for $\hat{\overline{w}}^{\tau,agr}$
- Compute time-update equations:

$$\hat{\overline{w}}^{\gamma}(0) = \left[w_{i_k^\mu}^{\tau}(k-1)\ \ w_{i_k^\mu}^{agr}(k-1)\right] \tag{A.12a}$$

$$\textbf{if } i_k^\mu \neq i_{k-1}^\mu \tag{A.12b}$$

$$\Rightarrow \overline{P}_\gamma^-(k) = diag\left(\ P_\tau^{i_k^\mu}(k-1)\quad P_{agr}^{i_k^\mu}(k-1)\ \right) \tag{A.12c}$$

$$\Rightarrow \overline{R}_\gamma^{r-}(k) = diag\left(\ R_\tau^{r,i_k^\mu}(k-1)\quad R_{agr}^{r,i_k^\mu}(k-1)\ \right) \tag{A.12d}$$

$$\Rightarrow R_\gamma^{e-}(k) = R_\gamma^{e+}(k-1) \tag{A.12e}$$

$$\textbf{else } \Rightarrow \overline{P}_\gamma^-(k) = \overline{P}_\gamma^+(k-1) \tag{A.12f}$$

$$\Rightarrow \overline{R}_\gamma^{r-}(k) = \overline{R}_\gamma^{r+}(k-1) \tag{A.12g}$$

$$\Rightarrow R_\gamma^{e-}(k) = R_\gamma^{e+}(k-1) \tag{A.12h}$$

Calculate sigma-points:

$$\chi^{\gamma-}(k) = ICUT\left(lb, ub, \overline{P}_\gamma^-(k)\right) \tag{A.13}$$

For $j = 1 : L$

$$\overline{M}^\sigma = M_4\left(\chi_j^{\gamma-}(k), w_{i_k^\mu}^\mu(k-1), \varphi, i_k^\mu\right) \tag{A.14a}$$

$$x(k) = f\left(w_{A,B}^\psi(k-1), \hat{x}^+(k-1), \varphi, x^+(k-1), \overline{M}^\sigma, i_k^\mu\right) \tag{A.14b}$$

$$Y_j = g\left(w_C^{\psi-}(k-1), x(k), \overline{M}^\sigma, i_k^\mu\right) \text{ (4.44)} \tag{A.14c}$$

- Compute the measurement-update equations:

$$\hat{\overline{w}}^{\gamma-}(k) = \sum_{i=1}^{L} W_i^{(m)} \chi_i^{\gamma-} \tag{A.15a}$$

$$\hat{d}^{-}(k) = \sum_{i=1}^{L} W_i^{(m)} Y_j \tag{A.15b}$$

$$\overline{P}_{\gamma}^{-}(k) = \sum_{i=1}^{L} W_i^{(c)} \left(\hat{\overline{w}}_i^{\gamma-}(k) - \chi_i^{\gamma-}(k)\right) \left(\hat{\overline{w}}_i^{\gamma-}(k) - \chi_i^{\gamma-}(k)\right)^T + \overline{R}_{\gamma}^{r-}(k) \tag{A.15c}$$

$$P_{\tilde{d}_k} = \sum_{i=1}^{L} W_i^{(c)} \left(Y_i - \hat{d}^{-}(k)\right) \left(Y_i - \hat{d}^{-}(k)\right)^T + R_{\gamma}^{e-}(k) \tag{A.15d}$$

$$P_{w_k d_k} = \sum_{i=1}^{L} W_i^{(c)} \left(\hat{\overline{w}}_i^{\gamma-}(k) - \chi_i^{\gamma-}(k)\right) \left(Y_i - \hat{d}^{-}(k)\right)^T \tag{A.15e}$$

$$K(k) = P_{w_k d_k} P_{\tilde{d}_k}^{-1} \tag{A.15f}$$

$$\hat{\overline{w}}^{\gamma+}(k) = max\left(min\left(\hat{\overline{w}}^{\gamma-}(k) + K(k)\left(d(k) - \hat{d}^{-}(k)\right), ub\right), lb\right) \tag{A.15g}$$

$$\overline{P}_{\gamma}^{+}(k) = \overline{P}^{\gamma-}(k) + K(k) P_{\tilde{d}_k} K(k)^T \tag{A.15h}$$

$$\overline{R}_{\gamma}^{r+}(k) = \left(1 - \alpha_{\gamma}^{r}\right) \overline{R}_{\gamma}^{r-}(k) + \alpha_{\gamma}^{r} K(k) \left(d(k) - \hat{d}^{-}(k)\right) \left(d(k) - \hat{d}^{-}(k)\right)^T K(k)^T \tag{A.15i}$$

$$\overline{R}_{\gamma}^{r+}(k) = diag\left(diag\left(\overline{R}_{\gamma}^{r+}(k)\right)\right) \tag{A.15j}$$

$$\Delta = \left(d(k) - \hat{d}^{-}(k)\right) \left(d(k) - \hat{d}^{-}(k)\right)^T \tag{A.15k}$$

$$R_{\gamma}^{e+}(k) = diag\left(diag\left(\left(1 - \alpha_{\gamma}^{e}\right) R_{\gamma}^{e-}(k) + \alpha_{\gamma}^{e} K(k) K(k)^T \Delta\right)\right) \tag{A.15l}$$

- Update rules and inputs degrees with $\hat{\overline{w}}^{\gamma+}(k)$.
- Save diagonals of local covariance matrices $\overline{P}_{\gamma}^{+}(k)$, and $\overline{R}_{\gamma}^{r+}(k)$ into global covariance matrices $P_{\tau,agr}(k)$ and $R_{\tau,agr}^{r}(k)$

with $\gamma = \sqrt{N + \lambda}$, $L = 2N + 1$, parameter dimension N, process noise covariance R^r, measurement noise R^e and a low pass filter poles α_{γ}^{e} and α_{γ}^{r}. Sigma points $W^{(m)}$ and $W^{(c)}$ are computed as in Table 4.3

Table A.3: RNFMS decoupled CIUKF rules and inputs degree optimization considering additive noise case. (system as in Equation 4.16)

A.4 RNFMS state optimization algorithm

Initialization:

$$\begin{aligned} \hat{x}(0) &= RNFMS_{x0}^{i_k^\mu}(k-1) \quad & P_x(0) &= \alpha_x^P eye(nx, nx) \\ R_x^r(0) &= \alpha_x^r eye(nx, nx) & R_x^e(0) &= \alpha_x^e eye(m, m) \end{aligned} \tag{A.16}$$

For each iteration k:

- Set $\varphi = [y(k-1) \;\; u(k-1)]$
- Compute $\overline{M}^\sigma = M_4\left(w^\tau(k-1), w^{agr}(k-1), w_{i_k^\mu}^\mu(k-1), u(k-1), i_k^\mu\right)$
- Compute time-update equations:

$$\hat{x}^-(k) = \hat{x}^+(k-1) \qquad P_x^-(k) = P_x^+(k-1) \tag{A.17a}$$

$$R_x^{r-}(k) = R_x^{r+}(k-1) \qquad R_x^{e-}(k) = R_x^{e+}(k-1) \tag{A.17b}$$

- Calculate sigma-points:

$$\chi^-(k) = \left[\hat{x}^+(k-1) \;\; \hat{x}^+(k-1) + \gamma\sqrt{P_x^+(k-1)} \;\; \hat{x}^+(k-1) - \gamma\sqrt{P_x^+(k-1)}\right] \tag{A.18}$$

For $j = 1 : L$

$$\begin{aligned} \chi_j^+(k) &= f\left(w_{A,B}^\psi(k-1), \hat{x}^+(k-1), \varphi, \chi_j^-(k), \overline{M}^\sigma, i_k^\mu\right) \\ Y_j &= g\left(w_C^{\psi-}(k-1), \chi_j^+(k), \overline{M}^\sigma, i_k^\mu\right) \text{ (4.44)} \end{aligned} \tag{A.19}$$

- Compute the measurement-update equations:

$$\hat{x}^-(k) = \sum_{i=1}^{L} W_i^{(m)} \chi_i^{x+}(k) \tag{A.20a}$$

$$\hat{d}^-(k) = \sum_{i=1}^{L} W_i^{(m)} Y_i \tag{A.20b}$$

$$P_x^-(k) = \sum_{i=1}^{L} W_i^{(c)} \left(\hat{x}_i^-(k) - \chi_i^{x+}(k)\right)\left(\hat{x}_i^-(k) - \chi_i^{x+}(k)\right)^T + R_x^{r-}(k) \tag{A.20c}$$

$$P_{\tilde{d}_k} = \sum_{i=1}^{L} W_i^{(c)} \left(Y_i - \hat{d}^-(k)\right)\left(Y_i - \hat{d}^-(k)\right)^T + R_x^{e-}(k) \tag{A.20d}$$

$$P_{w_k d_k} = \sum_{i=1}^{L} W_i^{(c)} \left(\hat{x}_i^-(k) - \chi_i^{x+}(k)\right)\left(Y_i - \hat{d}^-(k)\right)^T \tag{A.20e}$$

$$K(k) = P_{w_k d_k} P_{\tilde{d}_k}^{-1} \tag{A.20f}$$

$$\hat{x}^{+}(k) = \hat{x}^{-}(k) + K(k)\left(d(k) - \hat{d}^{-}(k)\right) \tag{A.21a}$$

$$P_x^{+}(k) = P^{x-}(k) + K(k)P_{\tilde{d}_k}K(k)^T \tag{A.21b}$$

$$R_x^{r+}(k) = (1-\alpha_x^r)\,R_x^{r-}(k) + \alpha_x^r K(k)\left(d(k) - \hat{d}^{-}(k)\right)\left(d(k) - \hat{d}^{-}(k)\right)^T K(k)^T \tag{A.21c}$$

$$R_x^{r+}(k) = diag\left(diag\left(R_x^{r+}(k)\right)\right) \tag{A.21d}$$

$$\Delta = \left(d(k) - \hat{d}^{-}(k)\right)\left(d(k) - \hat{d}^{-}(k)\right)^T \tag{A.21e}$$

$$R_x^{e+}(k) = diag\left(diag\left((1-\alpha_x^e)\,R_x^{e-}(k) + \alpha_x^e K(k)K(k)^T\Delta\right)\right) \tag{A.21f}$$

with $\gamma = \sqrt{N+\lambda}$, $L = 2N+1$, parameter dimension N, process noise covariance R_x^r, measurement noise R_x^e and a low pass filter poles α_x^e and α_x^r. Sigma points $W^{(m)}$ and $W^{(c)}$ are computed as in (4.8)

Table A.4: RNFMS decoupled UKF state estimation considering additive noise case. (system as in Equation 4.16)

A.5 RNFCS controler optimization algorithm

Initialization:

$$\begin{aligned} P_K(0) &= \alpha_{Ct}^{P} ones\,(n_x, n_u, C) & R_K^r(0) &= \alpha_{Ct}^{R} ones\,(n_x, n_u, Ct) \\ R_{Ct;u}^e(0) &= \alpha_{Ct;u}^{E} ones\,(m, m) & R_{Ct;c}^e(0) &= \alpha_{Ct;c}^{E} ones\,(m, m) \\ P_R(0) &= \alpha_{Ct}^{P} ones\,(n_u, m) & R_R^r(0) &= \alpha_{Ct}^{R} ones\,(n_u, m) \\ P_P(0) &= \alpha_{Ct}^{P} ones\,(n_u, m) & R_P^r(0) &= \alpha_{Ct}^{R} ones\,(n_u, m) \end{aligned} \tag{A.22}$$

For each iteration k:

- Get next i_k^μ based on $u(k-1)$ and $RNFMS(k-1)$
- Compute time-update equations:

$$\hat{w}^{Ct}(0) = RNFCS_{[K,R]}^{i_k^\mu}(k-1) \tag{A.23a}$$

$$\textbf{if } i_k^\mu \neq i_{k-1}^\mu \tag{A.23b}$$

$$\Rightarrow P_{Ct}^-(k) = diag\left(P_K^{i_k^\mu}(k-1),\ P_R^{i_k^\mu}(k-1),\ P_P^{i_k^\mu}(k-1)\right) \tag{A.23c}$$

$$\Rightarrow R_{Ct}^{r-}(k) = diag\left(R_K^{r,i_k^\mu}(k-1),\ R_R^{r,i_k^\mu}(k-1),\ R_P^{i_k^\mu}(k-1)\right) \tag{A.23d}$$

$$\Rightarrow R_{Ct;u}^{e-}(k) = R_{Ct;u}^{e+}(k-1) \tag{A.23e}$$

$$\Rightarrow R_{Ct;c}^{e-}(k) = R_{Ct;c}^{e+}(k-1) \tag{A.23f}$$

$$\textbf{else} \tag{A.23g}$$

$$\Rightarrow P_{Ct}^-(k) = P_{Ct}^+(k-1) \tag{A.23h}$$

$$\Rightarrow R_{Ct}^{r-}(k) = R_{Ct}^{r+}(k-1) \tag{A.23i}$$

$$\Rightarrow R_{Ct;u}^{e-}(k) = R_{Ct;u}^{e+}(k-1) \tag{A.23j}$$

$$\Rightarrow R_{Ct;c}^{e-}(k) = R_{Ct;c}^{e+}(k-1) \tag{A.23k}$$

- Calculate sigma-points:

$$\chi^{Ct-}(k) = \left[\hat{w}^{Ct-}(k) \;\; \hat{w}^{Ct-}(k) + \gamma\sqrt{P^{-}_{Ct}(k)} \;\; \hat{w}^{Ct-}(k) - \gamma\sqrt{P^{-}_{Ct}(k)}\right] \tag{A.24}$$

For $j = 1 : L$

$$\chi^{u+}_{j} = h_1\left(\chi^{Ct-}_{j}, x^{+}(k), Ref(k), \overline{M}^{\sigma}, i^{\mu}_{k}\right) \text{ (3.42)} \tag{A.25a}$$

$$u = min\left(max\left(\chi^{u+}_{j}, UD^{-}\right), UD^{+}\right) \tag{A.25b}$$

$$\varphi = [y(k) \;\; u] \tag{A.25c}$$

$$i^{\mu+}_{k} = h_2\left(\varphi, RNFMS(k)\right) \tag{A.25d}$$

$$\overline{M}^{\sigma} = M_4\left(w^{\tau}(k), w^{agr}(k), w^{\mu}(k), \varphi, i^{\mu+}_{k}\right) \tag{A.25e}$$

$$x^{-}(k+1) = f\left(w^{\psi}_{A,B}(k), \hat{x}^{+}(k), \varphi, \overline{M}^{\sigma}, i^{\mu+}_{k}\right) \tag{A.25f}$$

$$Y_j = g\left(w^{\psi}_{C}(k), x^{-}(k+1), \overline{M}^{\sigma}, i^{\mu+}_{k}\right) \text{ (4.44)} \tag{A.25g}$$

- Compute the measurement-update equations:

$$\hat{w}^{Ct-}(k) = \sum_{i=1}^{L} W^{(m)}_{i} \chi^{Ct-}_{i} \tag{A.26a}$$

$$\hat{u}^{-}_{ref}(k) = \sum_{i=1}^{L} W^{(m)}_{i} \chi^{u+}_{i} \tag{A.26b}$$

$$\hat{d}^{-}(k) = \sum_{i=1}^{L} W^{(m)}_{i} Y_j \tag{A.26c}$$

$$P^{ux}_{\tilde{d}_k} = \sum_{i=1}^{L} W^{(c)}_{i} \left(Y_i - \hat{d}^{-}(k)\right)\left(Y_i - \hat{d}^{-}(k)\right)^{T} \tag{A.26d}$$

$$P^{u}_{\tilde{d}_k} = P^{ux}_{\tilde{d}_k} + R^{e-}_{Ct;u}(k) \tag{A.26e}$$

$$P^{u}_{w_k d_k} = \sum_{i=1}^{L} W^{(c)}_{i} \left(\hat{u}^{-}_{ref}(k) - \chi^{u+}_{i}(k)\right)\left(Y_i - \hat{d}^{-}(k)\right)^{T} \tag{A.26f}$$

$$K^{u}(k) = P^{u}_{w_k d_k} P^{u^{-1}}_{\tilde{d}_k} \tag{A.26g}$$

$$u^{+}_{ref} = min\left(max\left(\hat{u}^{-}_{ref}(k) + K^{u}(k)\left(Ref(k+1) - \hat{d}^{-}(k)\right), 0\right), 1\right) \tag{A.26h}$$

$$R^{e+}_{Ct;u}(k) = \left(1 - \alpha^{e}_{Ct;u}\right) R^{e-}_{Ct}(k) + \alpha^{e}_{Ct;u} K^{u^T}(k) K^{u}(k) diag\left(diag\left(P^{ux}_{\tilde{d}_k}\right)\right) \tag{A.26i}$$

$$R^{e+}_{Ct;u}(k) = diag\left(diag\left(R^{e+}_{Ct;u}(k)\right)\right) \tag{A.26j}$$

$$P^{-}_{Ct}(k) = \sum_{i=1}^{L} W_i^{(c)} \left(\hat{w}_i^{Ct-}(k) - \chi_i^{Ct-}(k)\right) \left(\hat{w}_i^{Ct-}(k) - \chi_i^{Ct-}(k)\right)^T + R^{r-}_{Ct}(k) \tag{A.27a}$$

$$P_{\tilde{d}_k} = \sum_{i=1}^{L} W_i^{(c)} \left(\chi_i^{u+} - \hat{u}^{-}_{ref}(k)\right) \left(\chi_i^{u+} - \hat{u}^{-}_{ref}(k)\right)^T + R^{e-}_{Ct;c}(k) \tag{A.27b}$$

$$P_{w_k d_k} = \sum_{i=1}^{L} W_i^{(c)} \left(\hat{w}_i^{Ct-}(k) - \chi_i^{Ct-}(k)\right) \left(\chi_i^{u+} - \hat{u}^{-}_{ref}(k)\right)^T \tag{A.27c}$$

$$K(k) = P_{w_k d_k} P^{-1}_{\tilde{d}_k} \tag{A.27d}$$

$$\hat{w}^{Ct+}(k) = \hat{w}^{Ct-}(k) + K(k) \left(u^{+}_{ref} - \hat{u}^{-}_{ref}(k)\right) \tag{A.27e}$$

$$P^{+}_{Ct;c}(k) = P^{Ct-}(k) + K(k) P_{\tilde{d}_k} K(k)^T \tag{A.27f}$$

$$R^{r+}_{Ct;c}(k) = (1 - \alpha^r_{Ct})\, R^{r-}_{Ct}(k) + \tag{A.27g}$$

$$+\alpha^r_{Ct} K(k) \left(u^{+}_{ref} - \hat{u}^{-}_{ref}(k)\right) \left(u^{+}_{ref} - \hat{u}^{-}_{ref}(k)\right)^T K(k)^T \tag{A.27h}$$

$$R^{r+}_{Ct}(k) = diag\left(diag\left(R^{r+}_{Ct}(k)\right)\right) \tag{A.27i}$$

$$R^{e+}_{Ct;c}(k) = \left(1 - \alpha^e_{Ct;c}\right) R^{e-}_{Ct}(k) + \alpha^e_{Ct;c} diag\left(diag\left(K(k)^T K(k) P_{\tilde{d}_k}\right)\right) \tag{A.27j}$$

$$R^{e+}_{Ct;c}(k) = diag\left(diag\left(R^{e+}_{Ct;c}(k)\right)\right) \tag{A.27k}$$

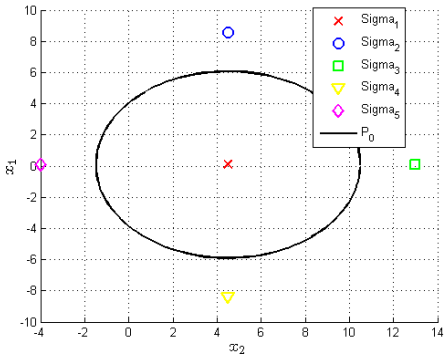
- Update global controller parameters $RNFCS^{i^{\mu}_k}_{[K,R]}(k)$ with new local models.
- Save diagonals of local covariance matrices P_{Ct} and R^{r+}_{Ct} into global covariance matrices $P_{K,R}(k)$ and $R^{r}_{K,R}(k)$
- New control action $u(k+1)$ can be computed.

with $\gamma = \sqrt{N + \lambda}$, $L = 2N+1$, parameter dimension N, process noise covariance R^r, measurement noise R^e and a low pass filter poles α^e_{Ct} and α^r_{Ct}. Sigma points $W^{(m)}$ and $W^{(c)}$ are computed according (4.8).

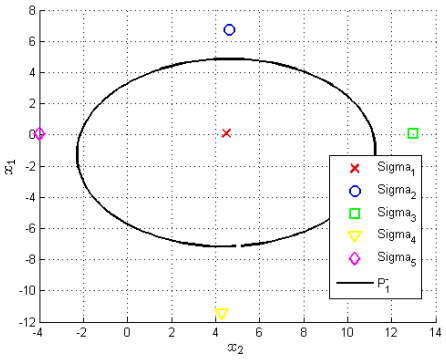
Table A.5: RNFCS decoupled UKF parameter estimation considering additive noise case. (system as in Equation 4.16)

B

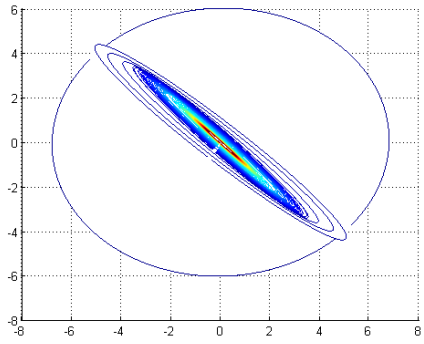
UKF application results



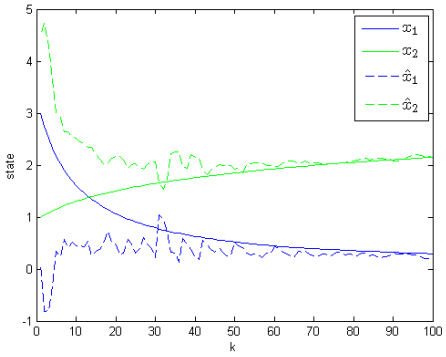
(a) Initial sigma points and covariance



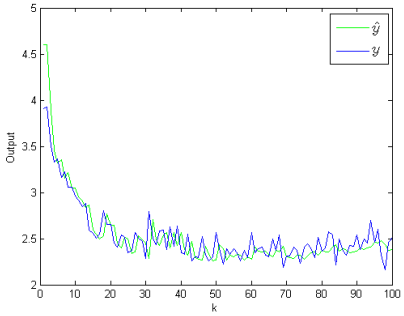
(b) Spread sigma points and resulting covariance



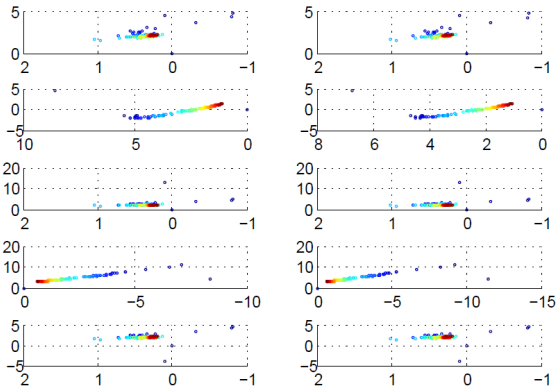
(c) P_k^- along time



(d) Predicted state evolution

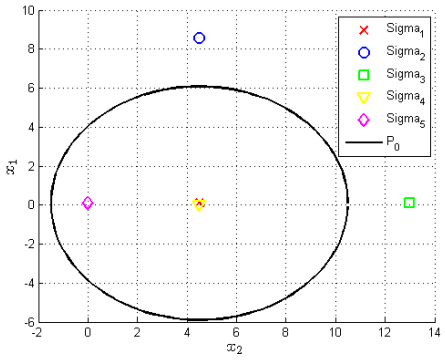


(e) Predicted output

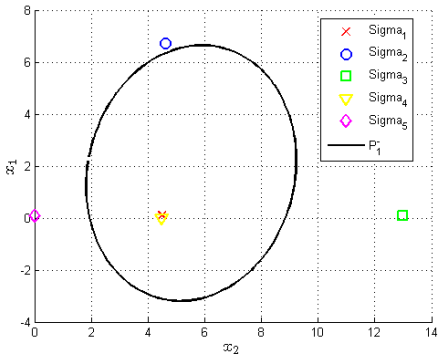


(f) Sigma points evolution, on left χ_{k-1}^{+}, on right χ_k^{*-}

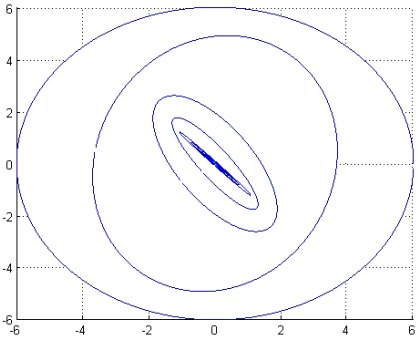
Figure B.1: 2-state CSTR simulation using standard UKF, $\alpha = 1$



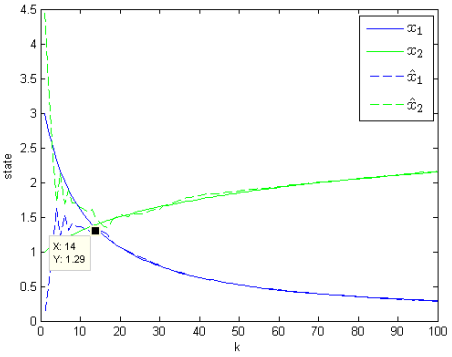
(a) Initial sigma points and covariance



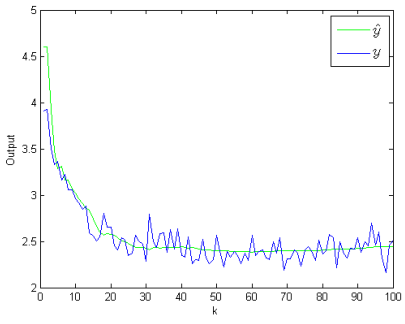
(b) Spread sigma points and resulting covariance



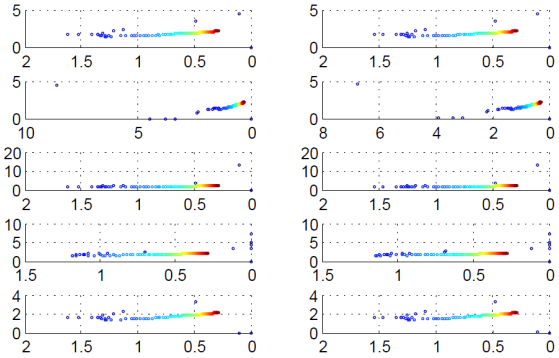
(c) P_k^- along time



(d) Predicted state evolution

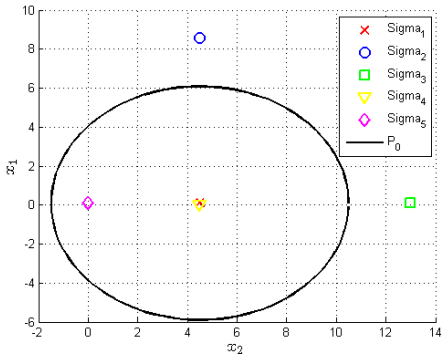


(e) Predicted output

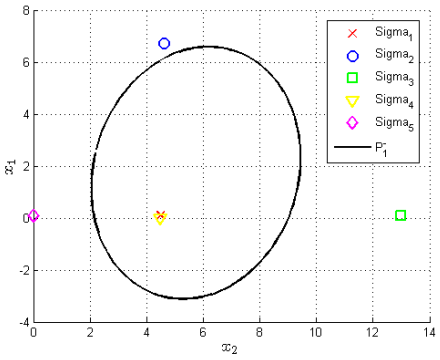


(f) Sigma points evolution, on left χ_{k-1}^{+}, on right χ_k^{*-}

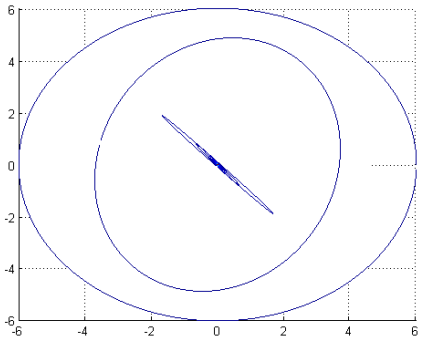
Figure B.2: 2-state CSTR simulation using PUKF, $\alpha = 1$



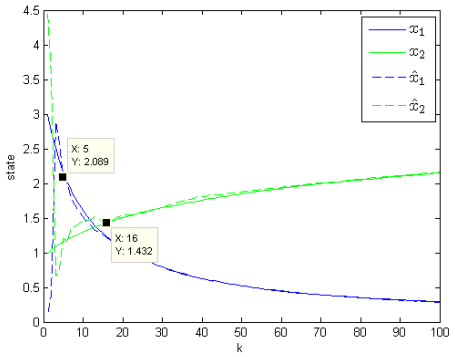
(a) Initial sigma points and covariance



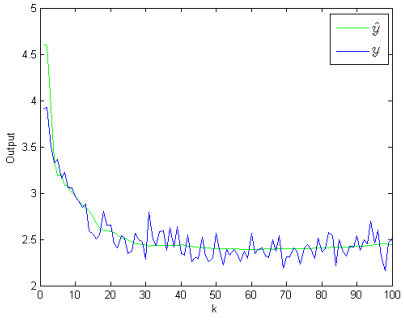
(b) Spread sigma points and resulting covariance



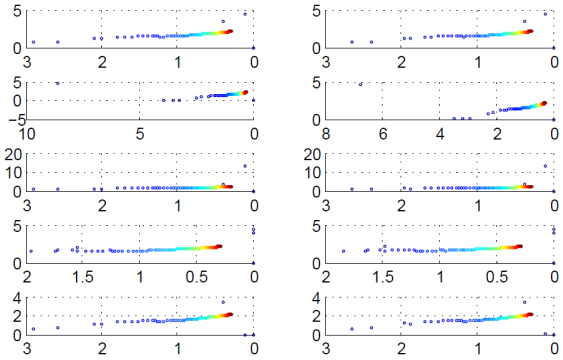
(c) P_k^- along time



(d) Predicted state evolution

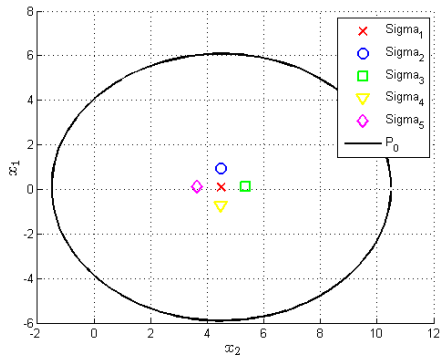


(e) Predicted output

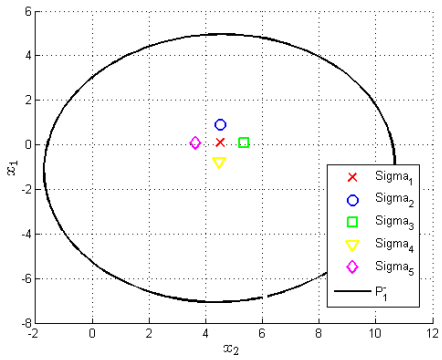


(f) Sigma points evolution, on left χ_{k-1}^+, on right χ_k^{*-}

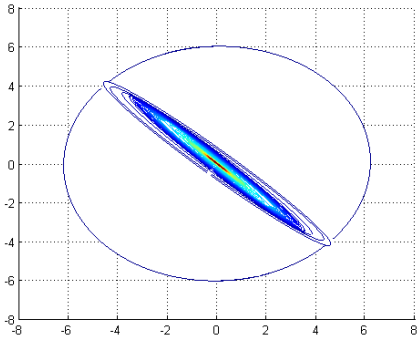
Figure B.3: 2-state CSTR simulation using CIUKF, $\alpha = 1$



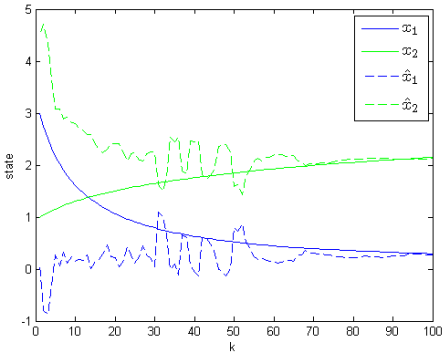
(a) Initial sigma points and covariance



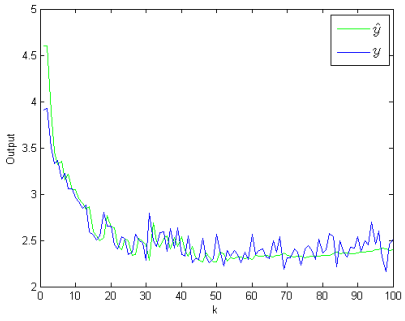
(b) Spread sigma points and resulting covariance



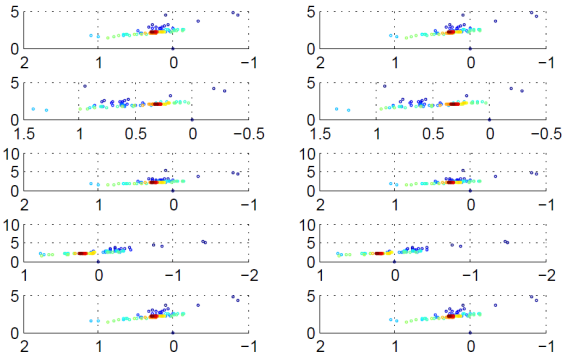
(c) P_k^- along time



(d) Predicted state evolution

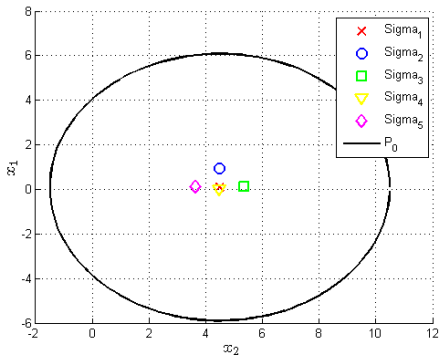


(e) Predicted output

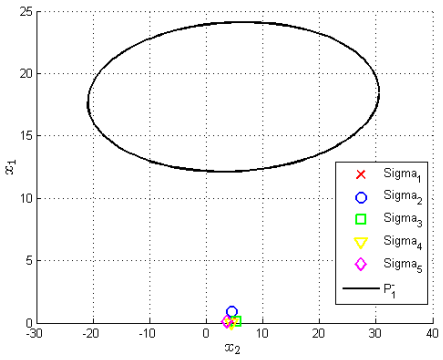


(f) Sigma points evolution, on left χ_{k-1}^{+}, on right χ_k^{*-}

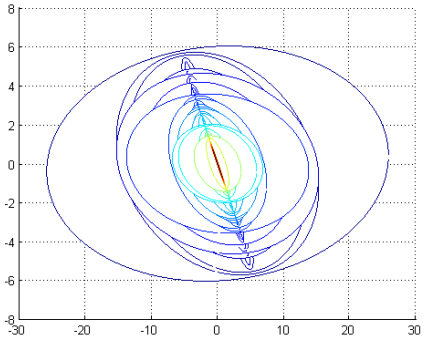
Figure B.4: 2-state CSTR simulation using standard UKF, $\alpha = 0.1$



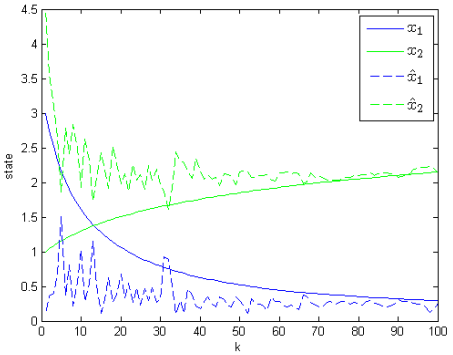
(a) Initial sigma points and covariance



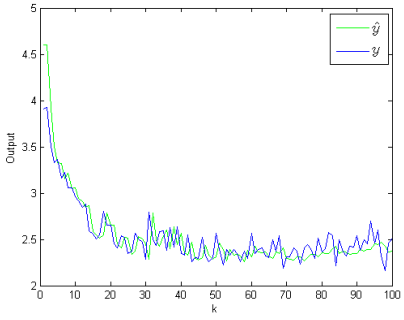
(b) Spread sigma points and resulting covariance



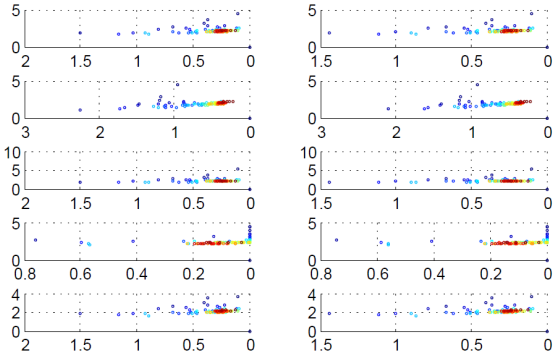
(c) P_k^- along time



(d) Predicted state evolution

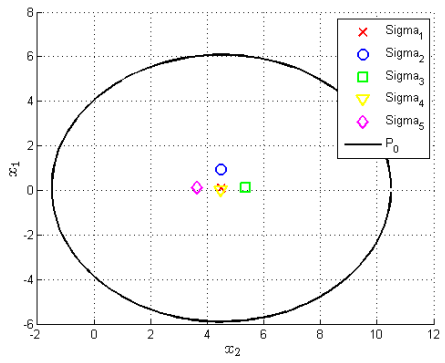


(e) Predicted output

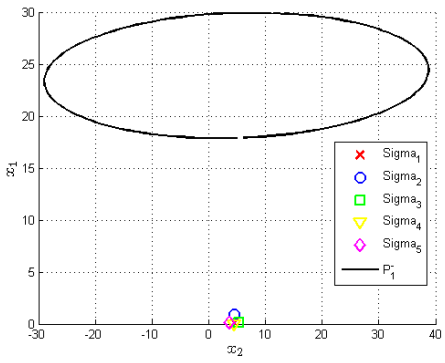


(f) Sigma points evolution, on left χ_{k-1}^{+}, on right χ_k^{*-}

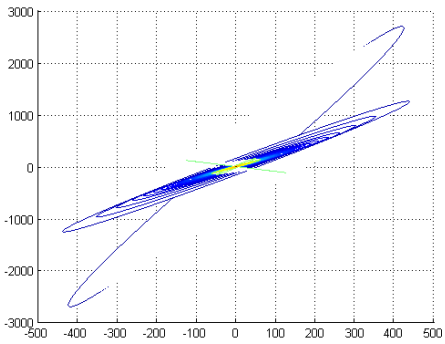
Figure B.5: 2-state CSTR simulation using PUKF, $\alpha = 0.1$



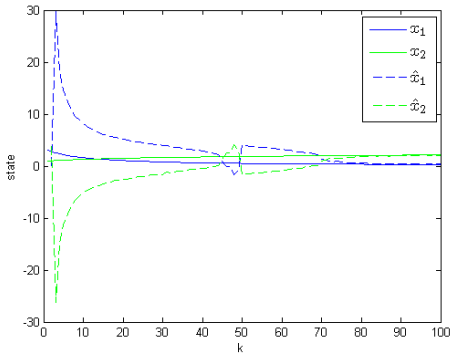
(a) Initial sigma points and covariance



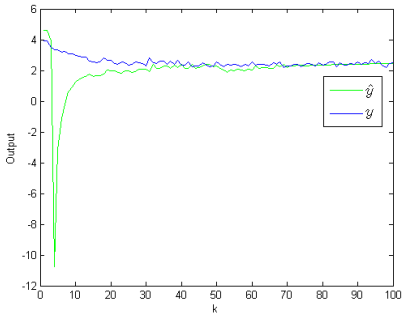
(b) Spread sigma points and resulting covariance



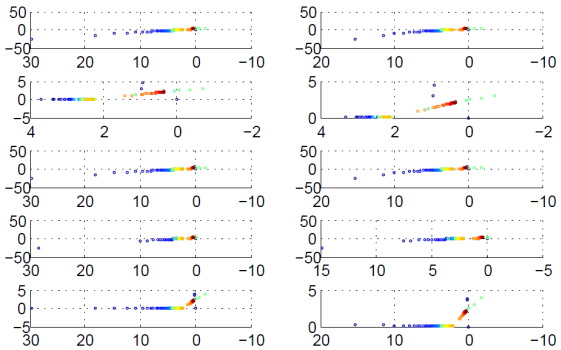
(c) P_k^- along time



(d) Predicted state evolution

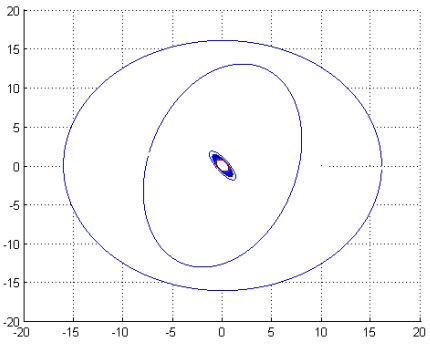


(e) Predicted output

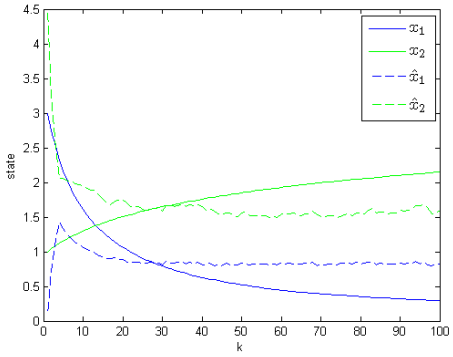


(f) Sigma points evolution, on left χ_{k-1}^{+}, on right χ_k^{*-}

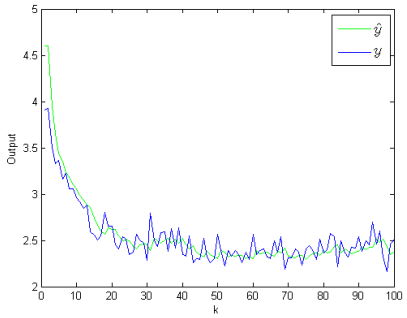
Figure B.6: 2-state CSTR simulation using CIUKF, $\alpha = 0.1$



(a) P_k^- along time

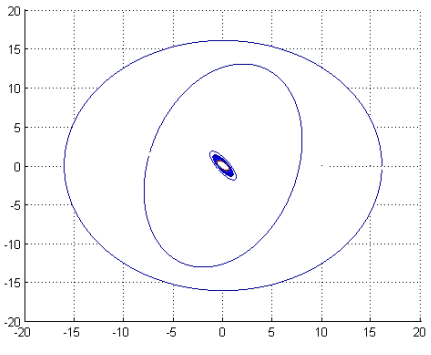


(b) Predicted state evolution

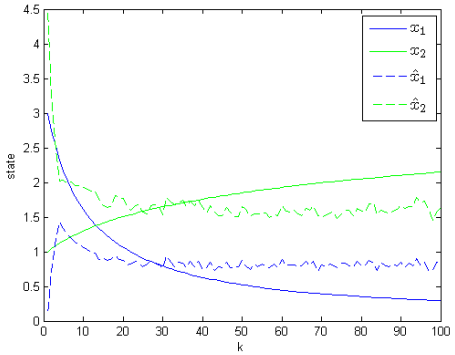


(c) Predicted output

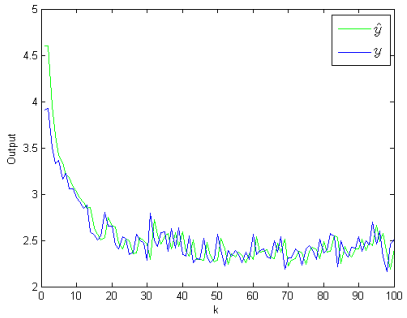
Figure B.7: 2-state CSTR simulation using CIUKF, $\alpha = 0.9$, $\lambda_r = 0$, $\lambda_e = 0$, $R_0^r = 0.1$ and $R_0^e = 1$



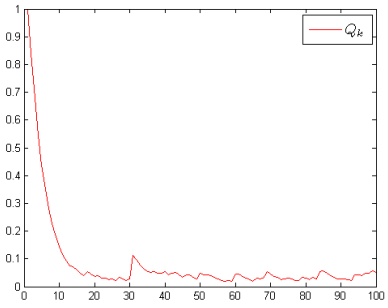
(a) P_k^- along time



(b) Predicted state evolution

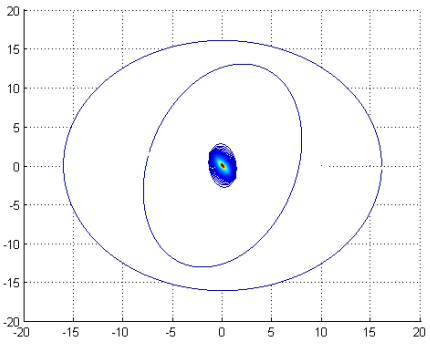


(c) Predicted output

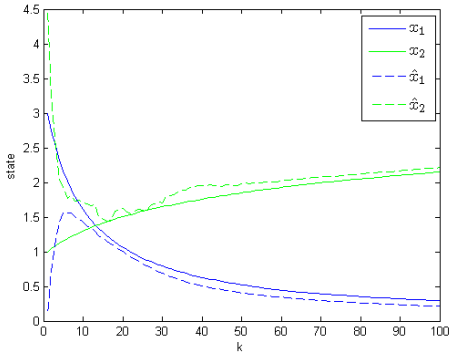


(d) R_k^{e-} along time

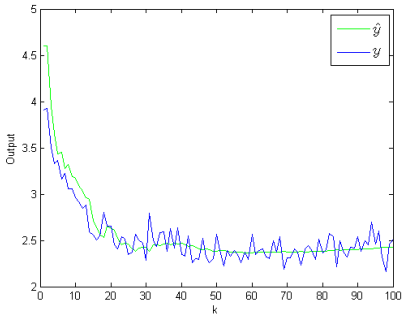
Figure B.8: 2-state CSTR simulation using IUKF, $\alpha = 0.9$, $\lambda_r = 0$, $\lambda_e = 0.2$, $R_0^r = 0.1$ and $R_0^e = 1$



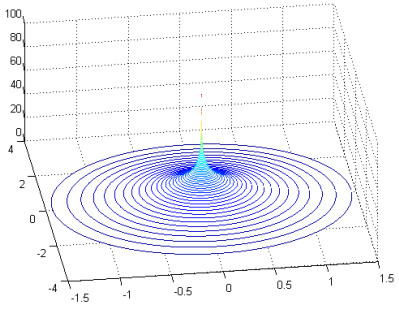
(a) P_k^- along time



(b) Predicted state evolution

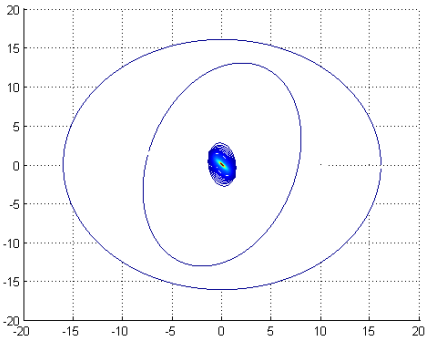


(c) Predicted output

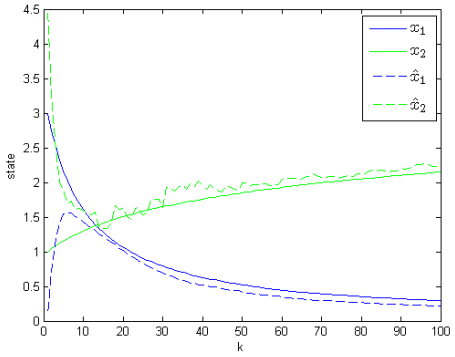


(d) R_k^{r-} along time

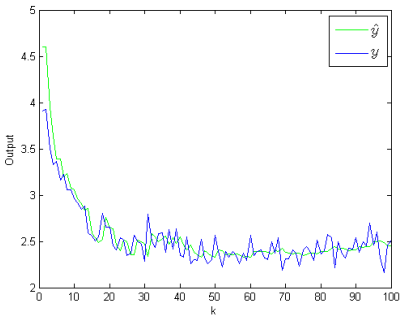
Figure B.9: 2-state CSTR simulation using IUKF, $\alpha = 0.9$, $\lambda_r = 0.2$, $\lambda_e = 0$, $R_0^r = 0.1$ and $R_0^e = 1$



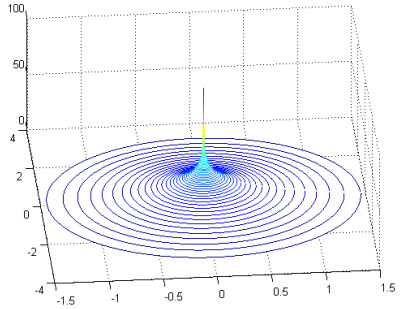
(a) P_k^- along time



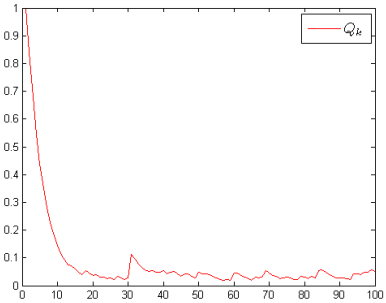
(b) Predicted state evolution



(c) Predicted output



(d) R_k^{r-} along time



(e) R_k^{e-} along time

Figure B.10: 2-state CSTR simulation using IUKF, $\alpha = 0.9$, $\lambda_r = 0.2$, $\lambda_e = 0.2$, $R_0^r = 0.1$ and $R_0^e = 1$