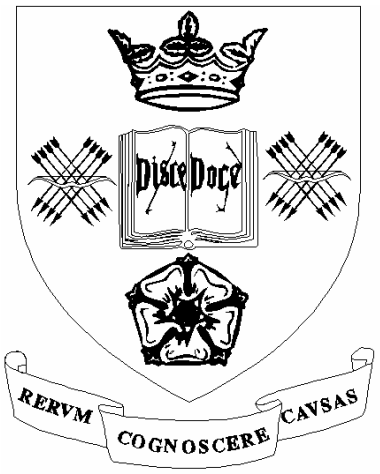


INTELLIGENT ADAPTIVE MULTISENSOR DATA FUSION USING HYBRID ARCHITECTURES

By

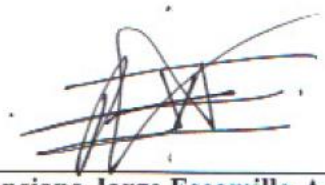
PONCIANO JORGE ESCAMILLA-AMBROSIO

A thesis submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy

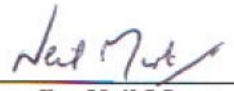
Department of Automatic Control and Systems Engineering
Faculty of Engineering
The University of Sheffield

June 2003

Unless otherwise stated in the text, the work described in this thesis was carried out solely by the candidate. None of this work has already been accepted for any degree, nor is it concurrently submitted in candidature for any degree.

Signature of the Candidate: 

Ponciano Jorge Escamilla-Ambrosio

Signature of the Supervisor: 

Dr. Neil Mort

ABSTRACT

This research work concerns the development of novel adaptive multi-sensor data fusion (MSDF) architectures which combine fuzzy logic, neuro-fuzzy and Kalman filtering techniques.

A novel adaptive scheme of the Kalman filter employing the principles of fuzzy logic is presented (referred to as fuzzy logic-based adaptive Kalman filter - FL-AKF). The adaptation is in the sense of dynamically adjusting the measurement noise covariance matrix R and/or the process noise covariance matrix Q from data as they are obtained.

An original hybrid MSDF architecture integrating the FL-AKF and a fuzzy logic performance assessment scheme is proposed (referred to as FL-AKF-FLA). This architecture merges the measurement vectors coming from multiple sensors and obtains a fused state-vector estimate which better reflects the actual value of the parameters being measured.

The MSDF architectures based on the standard Kalman filter (SKF) are extended and developed by including the FL-AKF. As a result, three novel adaptive MSDF architectures referred to as: fuzzy logic-based adaptive centralised Kalman filter (FL-ACKF), fuzzy logic-based adaptive decentralised Kalman filter (FL-ADKF), and fuzzy logic-based adaptive federated Kalman filter (FL-AFKF), are proposed.

A neuro-fuzzy-SKF system identifier and state estimator scheme is simplified and combined with the FL-AKF to develop a novel neuro-fuzzy-adaptive Kalman filter approach, referred to as neuro-fuzzy-AKF. The implementation of the FL-AKF-FLA MSDF architecture to fuse the estimates performed by multiple neuro-fuzzy-AKFs is also proposed.

The FL-ADKF is employed in a novel structure to design and auto-tune a modified hybrid PID-type fuzzy logic controller (MHPID-FLC) embedded in a multiple sensory environment. First, a new methodology for designing and tuning the scaling factors of the MHPID-FLC is presented. Second, the well-known relay auto-tuning algorithm is extended and developed for applications to the auto-tuning of the scaling factors of the MHPID-FLC. Finally, an approach which combines a low-order modelling method with the FL-ADKF MSDF architecture is proposed.

Examples developed under the MATLAB/SIMULINK simulation environment are presented to validate the proposed approaches. Good results are obtained in all cases.

ACKNOWLEDGEMENTS

I specially would like to thank my supervisor, Dr. Neil Mort, who has carefully read this manuscript and provided me with valuable comments and discussions regarding this research work. Thanks also for his supervision throughout the period of this research programme.

I am very grateful to the “Consejo Nacional de Ciencia y Tecnología” (CONACYT - México) for the financial support during this research project.

I also want to express my gratitude to my wife, Lupita, for her inexhaustible source of love, support and patience throughout this endeavour. I dedicate this thesis to her and to my parents, Ines and Primitivo.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 The multi-sensor data fusion approach and objectives of this research programme

When analysing a physical system, e. g. a chemical process, a heat exchanger, an aircraft etc., an engineer first attempts to develop a mathematical model that adequately represents some aspects of the behaviour of the system [Maybeck, 1979]. Next, he/she can investigate the system structure and modes of response by using the derived model and tools provided by system and control theories. If needed, compensators to alter these characteristics and provide appropriate inputs to generate desired system responses may be designed. However, the actual system behaviour only can be observed through measurement devices or sensors constructed to produce output data signals proportional to certain variables of interest. These sensor signals and the known inputs to the system constitute the only information that is directly discernible about the system behaviour. In most cases a single sensor cannot provide all the information required about the system under consideration. In addition, the observations made by a sensor are always subject to certain level of uncertainty and occasionally they are spurious or incorrect. In short, there is no such thing as a perfect sensor. Single sensor systems have no means of reducing this intrinsic uncertainty or in testing for erroneous measurements. Therefore, the possible failure of the sensor will result in complete system failure; single sensor systems are not robust and may fail with drastic consequences in critical systems [Durrant-Whyte, 1991]. Hence, there are several reasons to support the use of a number of different sensor devices in order to have a clearer and more robust picture of the system behaviour. These sensors yield functionally related signals, but now the problem is how to merge or fuse these signals to generate the best estimate of the variables of interest based on partially redundant and/or complementary data. This is the kernel of the Multi-Sensor Data Fusion approach.

Within the above context, "**the Multi-Sensor Data Fusion (MSDF) approach can be defined as the acquisition, processing, and synergistic combination of information gathered by various sensor devices and knowledge sources to provide a better understanding of a phenomenon under consideration**" [Varshney, 1997]. Therefore, the idea of any MSDF approach is to create a synergistic process in which the consolidation of individual data creates a combined information resource with a more accurate and reliable value than that offered by any individual sensor.

MSDF technology has undergone rapid growth since the late 1980's. At the present time, the MSDF approach is in fast development and with a broad sphere of applications. In the past, the main applications of MSDF were made in military surveillance command and control [Waltz and Buede, 1986] [Luo and Kay, 1989] [Luo and Kay, 1992]. Nowadays, the application of MSDF techniques has spread to a wide variety of fields such as robotics, aerospace engineering, image processing, medical diagnostics, and pattern recognition [Luo et al, 2002] [Grossmann, 1998] [Varshney, 1997].

Most of the proposed methods for MSDF make explicit assumptions about the nature of the sensed information. The most common assumptions include the use of a measurement model for each sensor, the incorporation of a statistically independent additive Gaussian noise term as a way of modelling the uncertainty (error) in the sensory data, and the assumption of statistical independence between the error terms for each sensor. All these assumptions are made with the

objectives of enabling the application of a broader variety of MSDF techniques and making the mathematics involved tractable.

One of the most traditional techniques to develop MSDF is Kalman filtering. The Kalman filter [Kalman, 1960] is an optimal linear estimator which incorporates all the information that can be provided to it. It processes all available measurements, regardless of their precision, to estimate the current value of the variables of interest, with use of (1) knowledge of the system and measurement device dynamics, (2) the statistical description of the system noises, measurement errors, and uncertainty in the dynamics models, and (3) any available information about initial conditions of the variables of interest [Maybeck, 1979].

Recently, non-traditional techniques also have been used to develop MSDF algorithms. In particular, fuzzy and neuro-fuzzy systems have demonstrated their capability to deal with uncertain and inexact information in control and system modelling [Harris *et al*, 2002] [Brown and Harris, 1994]. This has inspired their application in other areas, including MSDF [Kuo and Cohen, 1999] [Kobayashi *et al*, 1998]. Both of these techniques belong to the so-called "artificial intelligence" technology, usually referred to as well as the "soft computing" approach [Zadeh, 1994].

The great interest in creating machines and systems that can mimic the behaviour of humans has given origin to artificial intelligence technology. The main paradigms for generating intelligence in machines and systems include artificial neural networks, evolutionary computing techniques, fuzzy systems, intelligent agents as well as the different combinations of these approaches [Harris *et al*, 2002]. All these techniques are aimed at dealing with data driven processes subject to imprecision, uncertainty, non-linearties and with little prior knowledge. From the different possible combinations of these techniques, the most successful of them is referred to as the neuro-fuzzy systems approach.

Different techniques are being considered and investigated in the department of Automatic Control and System Engineering at the University of Sheffield to develop novel MSDF algorithms. These algorithms range from those based on traditional techniques, such as the well-established Kalman filtering methods [Mirabadi, 1999], to those based on ideas from non-traditional techniques derived from artificial intelligence technology [Prajitno, 2002]. However, little work has been made in exploring architectures considering the combination or fusion of both these approaches. **The overall aim of the research project "Intelligent Adaptive Multi-Sensor Data Fusion Using Hybrid Architectures" carried out in ACSE was especially focused to investigate the utilisation of synergistic combinations of fuzzy logic, neuro-fuzzy and Kalman filtering techniques to design novel adaptive MSDF architectures capable of dealing with uncertain and inexact information provided by imperfect sensors.** The word "hybrid" in the title of this research project refers to the actual combination of traditional techniques with non-traditional techniques to reach the intended objective.

1.2 Multi-sensor integration versus fusion

Commonly, in the literature the multi-sensor integration and multi-sensor fusion terms are used interchangeably without any distinction. However, a distinction between the two concepts is needed in order to separate the more general issues involved in the integration of multiple sensory devices at the system architecture and control level, from the more specific issues involving the actual fusion of sensory information [Luo and Kay, 1989]. This distinction is given in this section, as well as a description of the main multi-sensor integration models found in the literature, while a complete description of the most popular MSDF algorithms is reserved for chapter 2.

Multi-sensor integration is the synergistic use of the information provided by multiple sensory devices to assist in the accomplishment of a task by a system [Luo and Kay, 1989; 1992][Luo *et al*, 2002]. Multi-sensor fusion refers to any stage in the integration process where there is an actual combination or fusion of different sources of sensory information into one representational format [Luo *et al*, 2002].

Sensor integration is carried out using a hierarchical structure approach. These structures are useful in allowing an efficient representation of the different forms, levels, and resolutions of the information used for sensory processing and control [Luo *et al*, 2002]. Examples are the USA National Bureau of Standards (NBS) sensory and control hierarchy [Luo and Kay, 1989, 1992], and the Joint Directors of Laboratories (JDL) models [Hall and Llinas, 1997][Hall, 2002].

The NBS model was developed during the implementation of an experimental factory called the Automated Manufacturing Research Facility (AMRF). One of the objectives of the AMRF was the implementation of a multi-sensor interactive hierarchical robot control system. As can be seen in figure 1.1, the NBS model is a layered architecture, where complex tasks are partitioned into many progressively simpler tasks at lower levels. The main idea in this control structure is an ascending sensory processing hierarchy coupled to a descending task-decomposition control hierarchy via the world models at each level. The division in levels of processing and control is based on the observation that the complexity of a control program grows exponentially as the number of sensors and their associated processing increases. The idea is to reduce the complexity by isolating related portions of the required processing at one level.

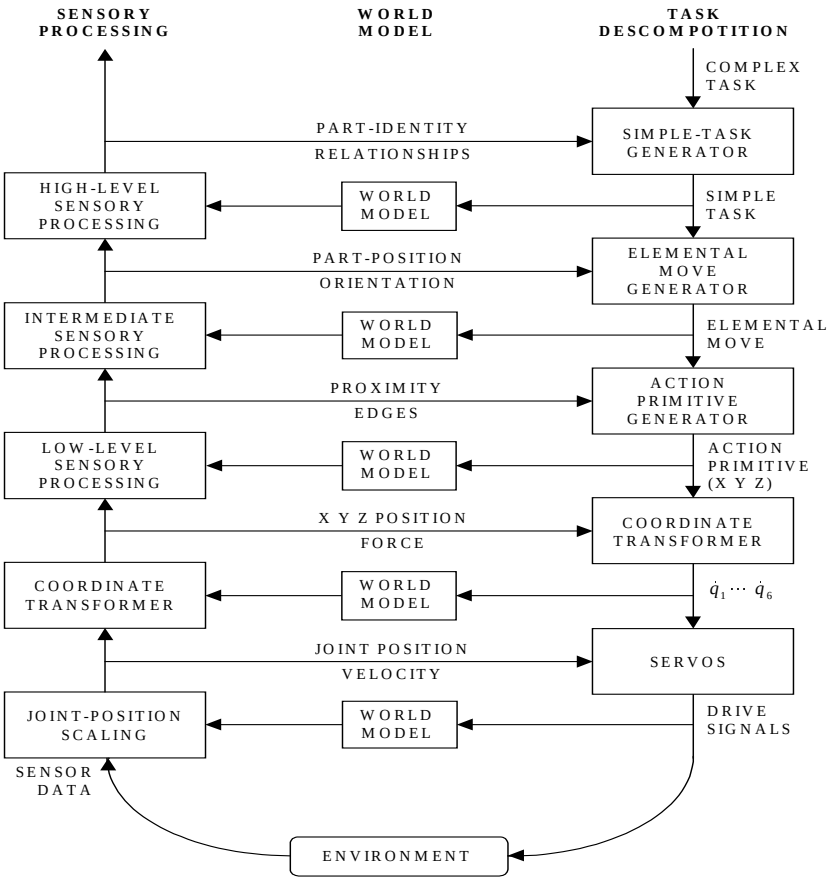


Figure 1.1 NBS sensory and control hierarchy used to control a multi-sensory robot (adapted from [Luo and Kay, 1989]).

The JDL data fusion process model is a conceptual model which identifies the processes, functions, categories of techniques, and specific techniques applicable to the data fusion process [Hall and Llinas, 1997] [Hall, 2002] [Varshney, 1997]. The basic components of this model are shown in figure 1.2. A short description of each block is given as follows:

- *Information sources.* Here is included information coming from sensors and *a priori* information from databases or human inputs.
- *Source pre-processing.* The objective of this block is to provide timely and the most pertinent data to the current situation in order to reduce processing load. This can be accomplished by data pre-screening and allocating data to appropriate processes.
- *Level one processing (object refinement).* Here, positional, parametric, and identity information of an entity are fused. This processing involves four basic functions: data alignment (transformation of data to a common set of co-ordinates and units), tracking (refinement of position, velocity and other object attributes), data association (correlation of data with objects), and identification (refinement of the object's identity estimate).
- *Level two processing (situation refinement).* Processing at this level attempts to make a contextual description of the relationship between objects and observed events. A contextual meaning to a collection of entities is assigned; here it is incorporated environmental information, *a priori* knowledge, and observations.
- *Level three processing (threat refinement).* At this level, based on the current situation, projections are made in order to evaluate future threats from an adversary. This task is fairly difficult because inferencing is based not only on results that can be obtained by computation but also on the strategies, tactics, doctrine, and political environment of the opposition.
- *Process refinement.* This is a meta-process, a process concerned about other processes. It monitors the fusion process performance, identifies information needed to improve system performance, and allocates sensors and resources to achieve the mission objectives.
- *Database management.* This is an important function required to support a successful fusion system. Functions needed are data retrieval, storage, archiving, compression, relational queries and data protection.
- *Human computer interaction.* This interface provides a mechanism for human input and communication of data fusion results to operators and users. Additionally, there are provided methods of directing human attention as well as augmenting cognition, e.g., overcoming the human difficulty in processing negative information.

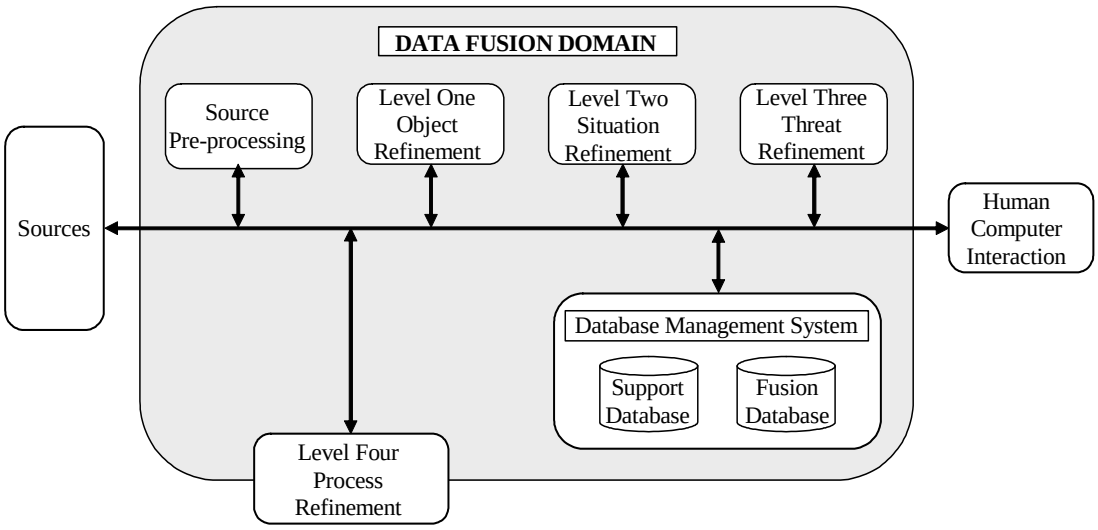


Figure 1.2 Basic components of the JDL data fusion process model (adapted from [Hall, 2002]).

Although the JDL model was originally developed for military applications, it has been used widely as a conceptual framework for the data fusion process in both military and non-military applications. However, it has its limitations, i.e., it does not handle adequately the multi-image fusion problem. But still it serves the purpose of unifying the data fusion concepts and providing a common terminology.

1.3 Motivation for MSDF

A MSDF algorithm can play a very important role in a system. This role can be evaluated with reference to the possible advantages that can be gained through the synergistic use of multi-sensory information. These advantages can be decomposed into a combination of four primary aspects [Luo and Kay, 1989] [Grossmann, 1998] [Varshney, 1997]:

a) Redundancy: Redundant information is provided from a group of sensors (or a single sensor over time) when each sensor perceives, possibly with a different accuracy, the same feature in the environment. The fusion of redundant information can reduce the overall uncertainty and thus increase the accuracy with which the features are sensed by the system. At the same time, multiple sensors providing redundant data increase the whole reliability in the case of a sensor error or failure. If one or more sensors fail or are unable to operate, then the system can continue to operate at a reduced performance level.

b) Complementarity: Multiple sensors are able to perceive features in the environment that are impossible to perceive using just a single sensor. Multiple sensors can observe a region larger than the one observable by a single sensor. In addition, different sensors can provide different types of information appropriate under different circumstances and for different tasks.

c) Timeliness: The speed of information provided by multiple sensors may be greater than that provided by a single sensor due to either the actual speed of operation of each sensor, or the processing parallelism that may be possible to achieve as part of the fusion process. Furthermore, since multiple sensors collect more data, a prescribed level of performance can be attained in a shorter time.

d) Cost of the information: Less costly information can be obtained with multiple cheap sensors compared to the cost of the equivalent information obtained from a single highly reliable but costly sensor.

More data and more sensors do not necessarily mean best information. The advantages gained through having multiple sensory data must be balanced with the possible drawbacks such as increased complexity or cost of the total system.

1.4 Overview of the thesis

The first task carried out in this investigation was to review the main existing MSDF algorithms. These algorithms can be broadly classified by the kind of techniques used in the fusion process as estimation methods, classification methods, inference methods, and artificial intelligence methods [Luo et al 2002]. In chapter 2 a review of the most popular MSDF algorithms in each class is given. The algorithms included are from estimation methods: the weighted average and the Kalman filter approaches; from classification methods: the K-means algorithm and the Kohonen feature map; from inference methods: the Bayesian inference and the Dempster-Shafer methods; and from artificial intelligence methods: the adaptive neural networks and fuzzy logic approaches.

From all the methods presented in chapter 2, Kalman filtering, fuzzy logic and neural networks are core to this thesis. Therefore, in chapter 3 a broader explanation of fuzzy inference systems (FIS), neural networks (NN) and neuro-fuzzy systems (the synergistic combination of FIS and NN) is given. A neuro-fuzzy system of particular interest in this research work is that which makes use of B-spline basis functions to implement membership functions. Thus, the reviewing of this class of neuro-fuzzy systems also is included in chapter 3.

Before proposing MSDF architectures which consider the combination of Kalman filtering with fuzzy logic techniques, in chapter 4 a more detailed explanation of the standard Kalman filter algorithm and the Kalman filter-based MSDF approaches: centralised Kalman filter, decentralised Kalman filter, and federated Kalman filter, are reviewed. A popular alternative form of the standard Kalman filter also is reviewed. An important issue in Kalman filtering is the consistency of the estimates performed by the filter. Therefore, two statistical tests to verify this aspect of the standard Kalman filter algorithm are examined in this chapter.

The problem of improving the performance of the standard Kalman filter (SKF), and in consequence the MSDF algorithms based on it, can be divided in two parts, a modelling problem and an estimation problem. The estimation problem is concerned with achieving better estimates through the proper use of the available process and measurement information. In that sense, the parameters to be adjusted to improve the performance or to maintain filter consistency are the statistical process noise and measurement noise information, the covariance matrices R and Q in the SKF. The SKF formulation assumes complete *a priori* knowledge of these process and measurement noise statistics. Whilst often they are assumed to be constant matrices, they may vary with time and, if this is so, then the nature of this variation is assumed to be known as well. However, in most practical applications these matrices are initially estimated or, in fact, are unknown. Therefore, using a SKF designed with fixed noise statistics in a changing dynamic environment is a major drawback. Thus, there is motivation for making the SKF adaptive with respect to the changing environment. In chapter 5 an on-line adaptive scheme of the Kalman filter employing the principles of fuzzy logic is presented, it is referred to as fuzzy logic-based adaptive Kalman filter (FL-AKF). The adaptation carried out is in the sense of adaptively adjusting the noise covariance matrices R and/or Q from data as it is obtained.

The SKF based MSDF architectures reviewed in chapter 4 require exact knowledge about the sensed environment and about the sensors. However, in real applications, only certain information is known about these elements. Therefore, in Chapter 6 four adaptive MSDF architectures based on the proposed FL-AKF are developed. These architectures are referred to as: fuzzy logic-based adaptive Kalman filter with fuzzy logic performance assessment scheme (FL-AKF-FLA), fuzzy logic-based adaptive centralised Kalman filter (FL-ACKF), fuzzy logic-based adaptive decentralised Kalman filter (FL-ADKF), and fuzzy logic-based adaptive federated Kalman filter (FL-AFKF).

Whilst chapters 5 and 6 deal with the problem of estimation to improve the performance of the SKF and the MSDF algorithms based on it, chapter 7 deals with the modelling problem. The modelling problem is concerned with the development of better models that more accurately describe the system under consideration. Because for linear systems generally a model exists or can readily be obtained, in this chapter only non-linear systems are considered. In that sense, first the neuro-fuzzy-SKF developed by Harris *et al* [1999, 2000, 2002] is reviewed. Then, a simplified version of the neuro-fuzzy-SKF estimator is developed. After that, a novel neuro-fuzzy adaptive Kalman filter (neuro-fuzzy-AKF) is designed based on the inclusion of the FL-AKF in the neuro-fuzzy-SKF structure. Finally, the hybrid MSDF architecture FL-AKF-FLA is used to merge the estimates obtained from several neuro-fuzzy-AKFs.

Although the developed MSDF architectures can be applied to a broad range of problems, one application which is of great interest for the author is in fuzzy control systems. In that sense, in chapter 8 first, a novel designing and tuning procedure for PID type fuzzy logic controllers (PID-FLC) is developed. Next, the auto-tuning procedure proposed by Astrom and Hagglund [1984] is extended and developed for tuning the scaling factors of PID-FLC. Then a novel procedure for auto-tuning PID-FLC by using multiple noisy sensors is presented. Here the developed MSDF architectures of chapter 6 can be applied. However, only the application of the FL-ADKF is exemplified.

Finally, in chapter 9 the conclusions and prospective future works of the research are presented.

1.5 Contributions

The following list provides a summary of the main contributions made by the author in this research programme:

- The development of a novel Fuzzy Logic-based Adaptive Kalman Filter (FL-AKF) which synergistically combines Kalman filtering and fuzzy logic techniques.
- The development of a novel MSDF architecture based on FL-AKFs and a fuzzy logic performance assessment scheme (referred to as FL-AKF-FLA architecture).
- The development of three hybrid adaptive MSDF architectures based on the proposed FL-AKF. These architectures are referred to as: fuzzy logic-based adaptive centralised Kalman filter (FL-ACKF), fuzzy logic-based adaptive decentralised Kalman filter (FL-ADKF), and fuzzy logic-based adaptive federated Kalman filter (FL-AFKF).
- The simplification of the neuro-fuzzy modelling network structure proposed by Harris *et al* [1999, 2000, 2002].
- The simplification of the neuro-fuzzy-standard Kalman filter (neuro-fuzzy-SKF) state estimator proposed by Harris *et al* [1999, 2000, 2002].
- The development of a novel neuro-fuzzy-adaptive Kalman filter (neuro-fuzzy-AKF) state estimator which synergistically combines Kalman filtering, fuzzy logic, and neuro-fuzzy techniques.
- The application of the FL-AKF-FLA MSDF architecture to merge the estimates obtained from multiple neuro-fuzzy-AKFs
- The development of a novel design and tuning procedure for PID type fuzzy logic controllers.
- The development of an auto-tuning procedure for PID type fuzzy logic controllers.
- The application of the developed FL-ADKF architecture in the auto-tuning of PID type fuzzy logic controllers using multiple noisy sensors.

CHAPTER 2

MULTI-SENSOR DATA FUSION TECHNIQUES

2.1 General multi-sensor data fusion methods

Over the years several MSDF algorithms have been developed and applied, individually and in combination, providing users with different degrees of information detail. The MSDF algorithms can be classified by levels in accordance with the amount of information they provide or by the kind of techniques used in the actual fusion process.

As shown in Table 2.1, by the kind of techniques used in the fusion process the MSDF algorithms can be broadly classified as follows: estimation methods, classification methods, inference methods, and artificial intelligence methods [Luo et al 2002]. In this chapter an overview of MSDF algorithms selected as being typical, from each class in Table 2.1, are presented as follows. From estimation methods: the weighted average and the Kalman filter approaches; from classification methods: the K-means algorithm and the Kohonen feature map; from inference methods: the Bayesian inference and the Dempster-Shafer methods; and from artificial intelligence methods: the adaptive neural networks and fuzzy logic approaches.

Table 2.1 Multi-sensor fusion algorithms classification

Estimation methods	Non recursive: Weighted average Least squares Recursive: Kalman filtering Extended Kalman filtering
Classification methods	Parametric templates Cluster analysis Learning vector quantization (LVQ) K-means clustering Kohonen feature map ART, ARTMAP, Fuzzy-ART network
Inference methods	Bayesian inference Dempster-Shafer method Generalized evidence processing
Artificial Intelligence methods	Expert systems Adaptive neural networks Fuzzy logic

2.2 Estimation methods

2.2.1 Weighted average

One of the simplest and most intuitive general methods of sensor data fusion is to take a weighted average of redundant information provided by a group of sensors and use this as the fused value [Luo and Kay, 1992].

Formally, the weighted average of N sensor measurements x_i with weights $0 \le w_i \le 1$ is,

$$\bar{x} = \sum_{i=1}^{N} w_i x_i \tag{2.1}$$

where $\sum_i w_i = 1$ and $w_i = 0$ if x_i is not within some specified thresholds. The weights can be used to account for the differences in accuracy between sensors, and a moving average can be used to fuse together a sequence of measurements from a single sensor so that the more recent measurements are given a greater weight.

If the x_i, i=1,2,…,N measurements are assumed to be independent normally distributed random variables, with distribution $\mathbf{N}(\bar{x}_i, \sigma_i^2)$, then a linear weighted mean aggregation model combining these random variables into one random variable x_f is given by [Basir and Shen, 1999]:

$$x_f = \beta_1 x_1 + \cdots \beta_N x_N \tag{2.2}$$

with variance,

$$\sigma_f^2 = \beta_1^2 \sigma_1^2 + \cdots \beta_N^2 \sigma_N^2 \tag{2.3}$$

where β_i is a positive weighting factor calculated by,

$$\beta_i = \frac{1}{\sigma_i^2 \sum_{j=1}^{N} \frac{1}{\sigma_j^2}} \tag{2.4}$$

with,

$$\sum_{i=1}^{N} \beta_i = 1 \tag{2.5}.$$

This method allows real-time processing of dynamic low-level data. However, in most cases the Kalman filter method is preferred because it provides a method that is nearly equal in processing requirements and, in contrast to a weighted average, results in estimates for the fused data that are optimal in a statistical sense.

2.2.2 Kalman filtering

The Kalman filter [Kalman, 1960] is a recursive, linear, optimal, real time data processing algorithm used to estimate the states of a dynamic system in a noisy environment. Kalman filtering is used in several multisensor systems. Mainly, it is used in those systems where it is necessary to fuse dynamic low-level redundant data in real time. If a linear model exist which describes the system under consideration, and both the system and sensor errors can be modelled as Gaussian noise, then the Kalman filter will provide unique statistically optimal estimates for the fused data. The recursive characteristic of the filter makes it appropriate for use in systems without large data storage capabilities.

The Kalman filtering can be applied in three different ways for MSDF purposes: Centralised (or standard Kalman filtering), decentralised, and federated Kalman filtering [Mirabadi *et al*, 1996]. In centralised Kalman filtering, the data from all the sensors is fed into one filter where all measurements are processed centrally to yield an optimal solution. High computational load on the processor and also lack of robustness are the disadvantages of this method.

In decentralised Kalman filtering, the standard Kalman filter is divided into one or more sensor-dedicated local filters and a master filter. Operationally, to obtain a global solution, the master filter periodically fuses outputs of local filters working in parallel. The computation load in this method is significantly reduced and the results may be locally suboptimal but globally optimal.

The federated method of Kalman filtering is sometimes recognised as a special case of the decentralised method. It employs the principle of information sharing among the local Kalman filters to improve the fault tolerance performance of the system [Gao *et al*, 1993]. The possible information to be shared includes the kinematics process noise, the initial conditions information, and common measurement information.

Due to the importance of the Kalman filter algorithm for the development of the proposed hybrid MSDF architectures, a broader description of it and the different MSDF architectures based on it are given in chapter 4.

2.3 Classification methods

2.3.1 K-means clustering

The K-means clustering algorithm partitions a collection of n vectors $\mathbf{x}_j$, j=1,…,n, into m groups (or clusters) G_i, i=1,…,m and finds a cluster center c_i in each group such that a cost function (or an objective function) of dissimilarity (or distance) measure is minimised [Jang *et al*, 1997]. If the Euclidean distance is chosen as the dissimilarity measure between a vector $\mathbf{x}_k$ in group j and the corresponding cluster center c_i, then the cost function is defined as:

$$J = \sum_{i=1}^{m} J_i = \sum_{i=1}^{m} \left(\sum_{k, \mathrm{x}_k \in G_i} \left\| \mathrm{x}_k - c_i \right\|^2 \right) \tag{2.6},$$

where $J_i = \sum_{k, \mathrm{x}_k \in G_i} \left\| \mathrm{x}_k - c_i \right\|^2$ is the cost function within group i. Thus, the value of J_i depends on the geometrical properties of G_i and the location of c_i.

Therefore the K-means clustering algorithm follows four basic steps:

(1) Initialise the cluster centers c_1,…,c_m by randomly selecting m points from among all of the data points.

(2) Define the partitioned groups by building a matrix known as membership matrix U, which is an $m \times n$ binary matrix, where the element u_{ij} is 1 if the j-th data point $\mathbf{x}_j$ belongs to group i, and 0 otherwise, this is:

$$u_{ij} = \begin{cases} 1 & \text{if} \quad \left\|\mathrm{x}_j - c_i\right\|^2 \le \left\|\mathrm{x}_j - c_k\right\|^2, \quad \text{for} \quad \text{each} \quad k \ne i \\ 0 & \text{otherwise} \end{cases} \tag{2.7},$$

which means that point $\mathbf{x}_j$ is assigned to group i if c_i is the closest center among all centers.

(3) Compute the cost function in accordance with (2.6). The procedure is stopped if either the cost function is below a certain tolerance value or its improvement over previous iteration is below a certain threshold.

(4) Update the cluster centers accordance with:

$$c_i = \frac{1}{\left|G_i\right|} \sum_{k, \mathrm{x}_k \in G_i} \mathrm{x}_k \tag{2.8},$$

where $\left|G_i\right|$ is the size of G_i, or $\left|G_i\right| = \sum_{j=1}^{n} u_{ij}$; and go to step 2.

Although the K-means algorithm is one of the most widely used clustering techniques it has several drawbacks. The K-means algorithm performance depends on the choice of the initial cluster centers [Jang *et al*, 1997]. Additionally, there is no guarantee that it will converge to an optimum solution. Also, for solving large problems, a great amount of computational effort will be required.

2.3.2 Kohonen feature maps

In this section a special type of neural network named the Kohonen feature map, also known as the self-organising map (SOM) network [Kohonen, 1990], is reviewed; although, the general artificial neural networks approach is discussed in section 2.5.2. The SOM is a two layered neural network that can learn from complex, multi-dimensional data and transform them into visually decipherable clusters [Kiang, 2001]. The SOM network performs unsupervised training based on the competitive learning paradigm [Jang *et al*, 1997]. Unsupervised learning is characterised in that it does not require the knowledge of target values. The nodes in the network converge to form clusters to represent groups of entities with similar properties. The number and composition of clusters can be visually determined based on the output distribution generated by the training process.

The SOM network has two layers of nodes, as shown in figure 2.1(a), the input layer and the Kohonen (or output) layer. The input layer is fully connected to the two-dimensional Kohonen layer. The activation of each unit in the Kohonen layer is determined by multiplying the input from each input unit by its corresponding synaptic weight and then summing for all the inputs to a particular Kohonen unit. Mathematically, this is the dot product of the input vector $\mathbf{x}=[x_1,\ldots,x_n]^{\mathrm{T}}$ and the weight vector $\mathbf{w}_i=[w_{i1},\ldots,w_{in}]^{\mathrm{T}}$.

$$a_i = \sum_{j=1}^{n} x_j w_{ij} = \mathbf{x}^T \mathbf{w}_i \tag{2.9}$$

where a_i is the activation value of Kohonen unit i, and the term w_{ij} is the weight connecting input j with Kohonen unit i. Since in most SOM networks each input unit is connected to each Kohonen unit, a single processing cycle is the computation of the dot product of the input vector

and the weight *matrix*, which is composed of all the weight vectors of all the units in the Kohonen layer. Typically, the weight vectors and the input vectors are normalized before this operation.

After calculating the activation values, the Kohonen unit with the largest activation is declared ''the winner''. Then, the weights of the winning unit are updated to more closely resemble the input vector that just stimulated it. An important concept in SOM networks is that not only the winning unit's weights are updated, but also all of the units' weights in a *neighbourhood* (see figure 2.1(b)) around the winner unit. As a result of these simple steps the network undergoes self-organization.

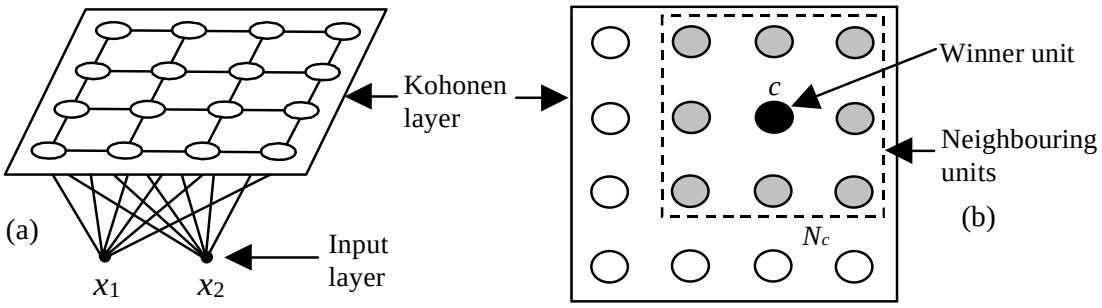


Figure 2.1 (a) A Kohonen SOM network with 2 inputs and 16 output units; (b) neighbourhood around a winner unit.

The learning procedure used in the SOM network uses a similarity measure to select a winning unit, which is the one with the largest activation. The training of the SOM network follows next procedure:

(1) A winning output unit is selected as the one with the largest similarity measure between all weight vectors $\mathbf{w}_i$ and the input vector $\mathbf{x}$. When the Euclidean distance is chosen as the dissimilarity measure the winning unit c satisfies the following equation:

$$\|\mathbf{x}-\mathbf{w}_c\| = \min_i \|\mathbf{x}-\mathbf{w}_i\| \qquad (2.10),$$

where the index c refers to the winning unit.

(2) Let Nc denote a set of index corresponding to a neighbourhood around winner c. The weights of the winner and its neighbouring units are then updated by:

$$\mathbf{w}_i(t+1) = \begin{cases} \mathbf{w}_i(t) + \eta[\mathbf{x}(t) - \mathbf{w}_i(t)] & if \quad i \in N_c(t) \\ \mathbf{w}_i(t) & if \quad i \notin N_c(t) \end{cases} \qquad (2.11),$$

where $t = 0,1,2,\ldots$ is an integer representing the discrete time co-ordinate, and η is a small positive learning rate. The neighbourhood of a winning unit can be defined by using a *neighbourhood function* $\Omega_c(i)$ around a winning unit c. One example of neighbourhood function is the Gaussian function defined as:

$$\Omega_c = exp\left(\frac{-\|p_i - p_c\|^2}{2\sigma^2}\right) \qquad (2.12),$$

where p_i and p_c are the positions of the output units i and c, respectively; σ reflects the scope of the neighbourhood. By using (2.12), (2.11) transforms to:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta\Omega_c(i)[\mathbf{x}(t) - \mathbf{w}_i(t)] \quad (2.13),$$

where i is the index for all output units.

It is suggested that in order to achieve better convergence, the learning rate η and size of neighbourhood σ should be gradually decreased with time [Kohonen, 1990].

2.4 Inference methods

2.4.1 Bayesian inference

Different sensor fusion algorithms have been devised according to the rules of probability theory [Luo and Kay, 1992, 1989] [Larsen, 1998] [Walts and Bude, 1986]. Particularly, Bayesian inference uses Bayes' rule for calculating the conditional or *a posteriori* probability of a hypothesis being true given supporting evidence.

In general terms, Bayes' rule evaluates the probability of occurrence of an arbitrary event A assuming that another event B has occurred:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (2.14),$$

where:

$P(A|B)$ = *a posteriori* probability of occurrence of event A given that event B has occurred.
$P(B|A)$ = probability of B conditioned on the occurrence of A; as well referred to as the *likelihood* function of A.
$P(A)$ = *a priori* probability of A.

Commonly, Bayes' rule is thought of in terms of updating the belief about a hypothesis A in the light of new evidence B. Thus, the *posterior* belief $P(A|B)$ is calculated by multiplying the *prior* belief $P(A)$ by the *likelihood* $P(B|A)$ that B will occur if A is true. The denominator $P(B)$ in (2.14) is just a normalising constant that ensures the posterior adds up to 1. $P(B)$ can be calculated by summing the numerator over all possible values of A,

$$P(B) = \sum_i P(B|A_i)P(A_i) \quad (2.15).$$

For multi-sensor data fusion purposes [Larsen, 1998] [Klein, 1999], Bayes' rule is generally used to support the fusion of identity information concerning some property of an object. This information is usually expressed in form of hypotheses about the identity of the object. Because of this, the field is referred to as *identity fusion*. For example, consider a set Θ containing N mutually exclusive and exhaustive hypotheses concerning the identity of some object:

$$\Theta = \{H_1, H_2, \ldots, H_N\} \quad (2.16).$$

A sensor, database, or human can express belief in these hypotheses by a set of probabilities (*a priori* probabilities), $P(\Theta)$:

$$P(\Theta) = \{P(H_1), P(H_2), \ldots, P(H_N)\} \tag{2.17}$$

where:

$$\sum_{i=1}^{N} P(H_i) = 1 \tag{2.18}.$$

Now, consider that there are *m* sensors reporting parametric data z_j (e.g. IR signatures, radar cross section, pulse repetition interval, etc) about an object whose identity is unknown. This represents new evidence about the object's identity hypothesis H_i based on the sensor-specific observations. Next, given a probability distribution $P(H_i)$, and an observation z_j (the sub-script *j* means that the data is coming from the *j*-th sensor), an updated probability distribution, $P(H_i|z_j)$, contemplating the observation can be calculated using Bayes' rule provided the conditional probability distribution, $P(z_j|H_i)$, of the measurement is known [Larsen, 1998]:

$$P(H_i \mid z_j) = \frac{P(z_j \mid H_i)P(H_i)}{\sum_{k=1}^{N} P(z_j \mid H_k)P(H_k)} \tag{2.19},$$

where a likelihood function or conditional probability, $P(z_j|H_i)$, is the probability of sensor output being z_j given H_i is true. Here, the probability of a hypothesis, $P(H_i)$, before fusion is the *a priori* probability and the updated probability, $P(H_i|z_j)$, is the *a posteriori* probability. The a priori and likelihood probability distributions are usually found using a large amount of statistical data obtained by offline experiments or by analysing the information available in the problem at hand [Klein, 1999].

Thus, the probability of having observed object *i* from the set of *N* objects given evidence z_1 from sensor 1, evidence z_2 from sensor 2, etc., is:

$$P(H_i \mid z_1 \cap z_2 \cap \cdots \cap z_m), \quad i=1,\ldots,m \tag{2.20}.$$

Finally, a joint declaration of identity can be selected by applying a decision rule. If the most probable hypothesis $P(H_i \mid z_1 \cap z_2 \cap \cdots \cap z_m)$ is selected as true, then the decision rule is referred to as the *maximum a posteriori* (MAP). Other decision rules exist, as the *maximum likelihood*, *Neyman–Pearson*, etc., an explanation of these rules can be found in [Klein, 1999]. A graphical representation of an identity fusion process using Bayesian inference is shown in figure 2.2.

Thus, the Bayesian inference method provides a mathematical structure to combine identity declarations from multiple sensors to obtain a new improved joint identity declaration. As inputs, the method requires the likelihood functions $P(z_j|H_i)$ for each sensor and entity and the a priori probabilities that the hypotheses $P(H_i)$ are true. If *a priori* information does not exist about the relative likelihood of H_i, then the principle of indifference can be used in which $P(H_i)$ for all *i* are set equal.

Bayesian theorem implementation in data fusion is limited by this technique's inability to depict the level of uncertainty in a particular sensor state, as well as its inability to ensure consistency in a collection of interrelated propositions. Other frequently cited drawbacks of a

Bayesian inference-based data fusion algorithm are its heavy computer processing and memory requirements.

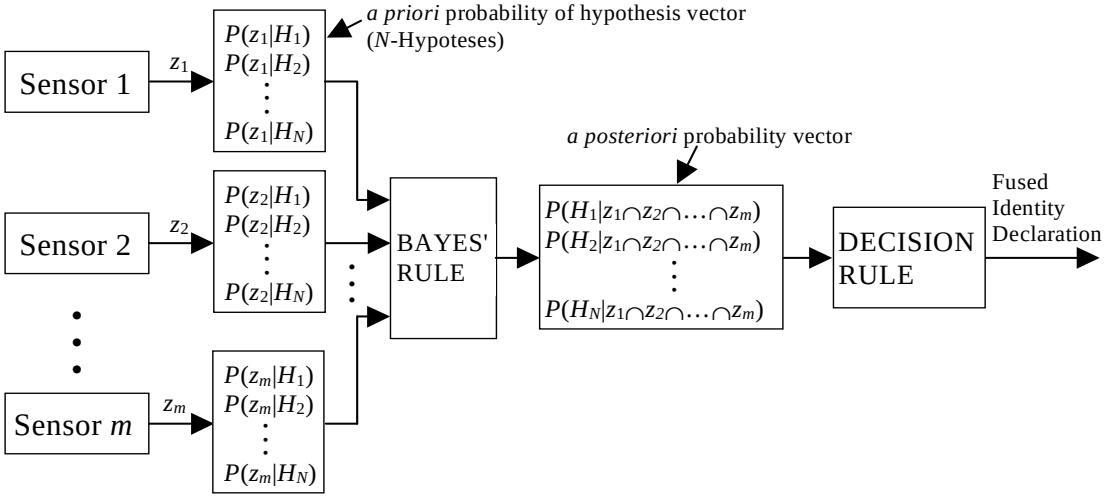


Figure 2.2 Graphical representation of an identity fusion process using Bayesian inference.

2.4.2 Dempster–Shafer evidential reasoning

Dempster–Shafer Evidential Reasoning (DSER) is an alternative to Bayesian inference. DSER is a generalisation of the Bayesian inference approach that offers a way to combine uncertain information from disparate sensor sources [Bogler, 1987].

For example, consider the same set of hypotheses as before:

$$\Theta = \{H_1, H_2, \ldots, H_N\} \tag{2.21}$$

In DSER the set of hypotheses is called the *frame of discernment*, and each hypothesis in Θ is called a *singleton*. A disjunction of singletons is referred to as a *proposition*. The set of all 2^N possible propositions, denoted by 2^Θ, is the *power set* of Θ. Thus, 2^Θ contains all possible subsets of Θ, including Θ itself, the empty set Ø, and each of the singletons, this is:

$$2^\Theta = \{\mathbf{\emptyset}, \{H_1\}, \ldots, \{H_N\}, \{H_1 \vee H_2\}, \ldots, \{H_1 \vee H_2 \vee \ldots \vee H_N\}\} \tag{2.22}$$

Conversely to the Bayesian inference where evidence has to be represented as a vector of *N* probabilities relating only to the singletons (to the hypotheses in Θ), in DSER evidence is represented as probability *masses* relating to one or more propositions.

The actual distribution of probability masses among the propositions is defined using a *mass function*:

$$m : 2^\Theta \rightarrow [0,\ 1] \tag{2.23}$$

and termed as a *basic probability assignment* (BPA). Thus, (2.23) maps a unit probability mass or belief across the elements of 2^Θ subject to next conditions:

$$m(\emptyset) = 0 \tag{2.24}$$

$$\sum_{A \in 2^{\Theta}} m(A) = 1 \tag{2.25}.$$

where $m(A)$ is called A's *basic probability number* (BPN) or simply the *mass* of A. Any subset A of Θ with $m(A) > 0$ for a particular belief function is called a *focal element* of that function.

If a given source or sensor expresses some evidence where the masses sum up to less than one, the remaining mass is assigned to the disjunction of all singletons ($H_1 \vee H_2 \vee \ldots \vee H_N$) sometimes referred to as the *general level of uncertainty*.

The total support S committed to a particular proposition A is the sum of all masses assigned to all proper subsets B of A:

$$S(A) = \sum_{B \subseteq A} m(B) \tag{2.26},$$

where $S(A)$ is called the *support* (or *belief*) *function* of A, and defines the lower probability or minimum likelihood of each proposition A. In a similar manner, another function is defined by:

$$Pl(A) = 1 - S(\overline{A}) \tag{2.27},$$

where $Pl(A)$ is known as the *plausibility function* of A, $\overline{A}$ is the complement of A, i.e. $\overline{A} = \Theta - A$; and $S(\overline{A})$ is called the *doubt function* of A:

$$Dbt(A) = S(\overline{A}) \tag{2.28}.$$

The plausibility function determines the upper probability or maximum likelihood of A and represents the mass that is free to move to the support of A as additional information becomes available. Plausibility can be though of as the extent to which the evidence does not support the negation of a proposition [Henkind and Harrison, 1988].

The difference between the plausibility and the support function is known as the *uncertainty function* of A, $u(A)$. This is:

$$u(A) = Pl(A) - S(A) \tag{2.29}.$$

where $u(A)$ represents the mass that has not been assigned for or against belief in A.

The DSER approach allows the representation of total ignorance concerning the proposition A since $S(A) = 0$ does not imply $Dbt(A) > 0$, even though $Dbt(A) = 1$ does imply $S(A) = 0$. This cannot be possible in the Bayesian approach where $u(A) = 0$ for all $A \in 2^{\Theta}$. The interval formed by combining the support with the plausibility of A is known as the *uncertainty interval*: $[S(A), Pl(A)]$. The uncertainty interval represents, by its magnitude, how conclusive the information is for proposition A. For example, the interval [0,1] represents total ignorance concerning A. Whereas the intervals [0,0] and [1,1] represent A as being false and true, respectively. A graphical representation of the above concepts is shown in figure 2.3.

Thus, DSER provides the formalism to combine the probability masses provided by multiple sensors for compatible propositions. Propositions are compatible when their intersection exists. The intersection of the propositions having the largest probability mass is selected as the output of the fusion process. Therefore, if z_i is a piece of evidence (a measurement from a sensor) that induces BPA m_i, and z_j is a piece of evidence which induces BPA m_j, then the BPA induced by

the conjunction of evidence z_i and z_j is denoted by $m_i \oplus m_j$. The DSER theory defines the following combination rule (which is known as Dempster's rule) for determining $m_i \oplus m_j$ when A≠Ø:

$$m_i \oplus m_j(A) = \frac{\sum_{X \cap Y = A; A \neq \emptyset} m_i(X) m_j(Y)}{1 - \sum_{X \cap Y = \emptyset} m_i(X) m_j(Y)} \quad (2.30),$$

where X,Y ∈ 2^{Θ}. Thus, (2.25) specifies the combined probability mass assigned to A. The combination of the two propositions is also known as taking the *orthogonal sum*.

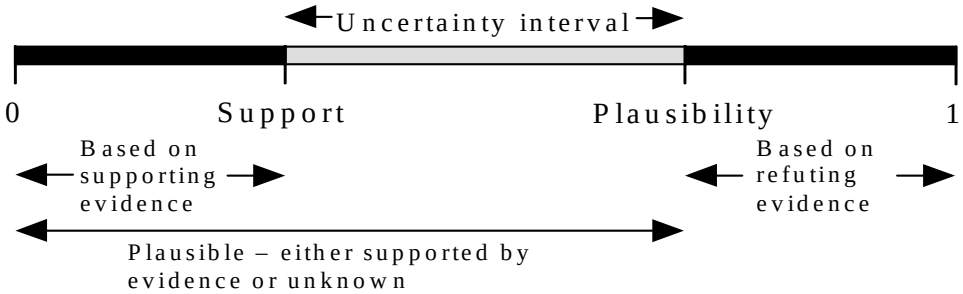


Figure 2.3 Support, plausibility, and uncertainty interval for a proposition (adapted from [Klein, 1999]).

The denominator in (2.30) is a normalisation factor that forces the new masses to sum to unity. It can be viewed as a measure of the degree of conflict or inconsistency in the information provided by the sensors. Note that if the factor is 0 the sensors are completely inconsistent and then the orthogonal sum operation is undefined. Note, as well, that by definition Ø must be assigned a mass probability of zero.

The main criticism of DSER is that it is not as theoretically rigorous as Bayesian approach. In addition, a major disadvantage of DSER is that the algorithm's computational complexity is significantly higher compared to Bayesian fusion approach (as Θ increases, 2^{Θ} increases exponentially).

2.5 Artificial intelligence methods

2.5.1 Fuzzy logic

If traditional logic (referred to here as crisp logic) is defined as the science that studies the formal principles of reasoning, then fuzzy logic can be defined as the science that studies the formal principles of approximated reasoning [Zadeh, 1988], of which crisp reasoning (traditional reasoning) is a particular case. Fuzzy logic has its origin in the theory of *fuzzy sets*, first proposed by Zadeh in 1965 as a way of dealing with the inexact nature of the human reasoning [Zadeh, 1965]. With his proposal, Zadeh offered a more appropriate conceptual scheme to represent the knowledge expressed in natural language than that provided by crisp logic.

Fuzzy logic was motivated by the necessity to find a conceptual structure adequate to manipulate the inherent vagueness and imprecision present in the representation that humans have of the world. Fuzzy logic rests in the affirmation that a concept generally has not defined

borders. In crisp logic concepts are forced to have abrupt limits. However, people do not use abrupt limits when expressing ideas about concepts.

In fuzzy logic an object is not restricted to be totally a member or totally not a member of a given set. Here an object is allowed to be partially a member of different sets at the same time, but at different degrees. This is possible because in fuzzy sets theory membership is a matter of degree [Zadeh, 1977], and thus, an object may have a grade of membership intermediate between zero (non-membership) and unity (full membership), this is in the whole range [0,1]. The grade in which an object is member of a set is defined by a *membership function*.

In addition to membership functions, in fuzzy logic different operators exist which are analogous to those found in crisp logic. Operations on fuzzy sets such as union, intersection, and complement are defined in order to manipulate fuzzy memberships. A description of the main operations on fuzzy sets is given in appendix A.

Thus, fuzzy logic offers a framework that can be used to represent processes of multi-sensor fusion [Hirota *et al*, 1992]. Each sensor reading can be viewed as a membership function of a fuzzy set. Fuzzy rules can be defined for quantifying a fused reading using the values of the membership functions. Then the information coming from different sensors can be combined by means of a fuzzy inference system (FIS), where fuzzy sets are used as adjectives in a qualitative rule base. The effect of each rule in the inference process is proportional to the degree of truth of the fuzzy sets associated with it. Therefore, by using fuzzy sets the uncertainty in multi-sensor fusion can be directly represented in the inference (i. e., fusion) process. This is possible because each proposition, as well as the actual implication operator, are allowed to be assigned a real number from 0 to 1 to indicate its degree of truth [Luo *et al*, 2002]. Consistent logical inference can take place if the uncertainty of the fusion process is modelled in some systematic fashion. A broader description of the inference process carried out in a FIS is presented in Chapter 3.

A disadvantage in fuzzy-logic based approaches is that as the number of inputs (sensors) grows, the number of rules grows as well. As a consequence, the inference process will require significant computational resources.

2.5.2 Neural networks

In section 2.3.2 the self-organising map, which is a special type of neural network, was discussed. In this section the general artificial neural networks approach is reviewed. An artificial neural network or simply a neural network (NN) is defined as a collection of processing elements (called neurons) and connection weights. The neurons and weights are structured in a network which is able to perform a mapping from an input space to an output space: $R^n \rightarrow R^m$ [Gupta and Rao, 1994]. A NN can have several layers, and each layer can have more than one neuron. The arrangement of neurons in layers or stages of processing is supposed to mimic the layered structure of a certain portion of the human brain.

The main function of each neuron in a NN is to perform a mapping from R^n to R^1. This mapping involves two distinct processing stages. The first one is a summation of the weighted inputs. While the second one involves the application of an activation function to the sum obtained in the first stage. The activation function can be one of many types, which can generate continuous or discrete outputs.

The main characteristic of a NN is its capability of storing knowledge in the connection weights. A procedure called *learning algorithm* is used to successively adjust the connection weights in order to find those values for which a better approximation is obtained to the desired

output. The learning algorithms can be either supervised or unsupervised. A supervised learning algorithm is that which involves the presentation of training input-output data and a subsequent modification of the network weights to achieve the desired mapping between inputs and outputs. An unsupervised learning algorithm involves the presentation of the input data only, followed by a self-organisation of the network through a modification of the network weights.

The network structure, which includes many neurons and connection weights, is what gives a NN its computational capabilities. NNs have been successfully applied in several areas including pattern recognition, system identification, and control systems. A broader description of how a NN works is given in Chapter 3.

One of the applications of NNs in MSDF is in what is known as feature-level fusion [Klein, 1999], as is graphically represented in figure 2.4. As is shown there, target features are extracted from several sensors, e.g. millimetre-wave radar, passive infrared sensor, and laser radar. These features are combined to form a composite vector that is used as input to a NN. The NN, which has been trained off-line to recognise the targets of interest and differentiate them from false targets, assigns observed objects to particular classes with some degree of confidence. It is necessary to remark that the training should be performed using information coming simultaneously from all the sensors. If one or more sensors are replaced for one of a different type, then the training procedure must be repeated.

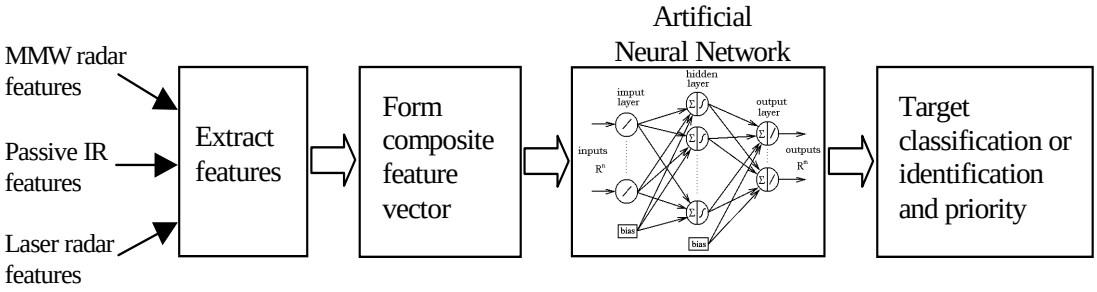


Figure 2.4 Feature-level fusion using an artificial neural network.

In general, NNs can be applied to solve three typical MSDF problems [Corcoran and Lowery, 1995]: *classification, quantification and description*. Classification problems are those where a system is required to relate the multidimensional input data to a predefined class that represents the state of the input space. An example of this kind of problem is condition monitoring, where multi-sensor information is used to indicate the condition of a process or system, and any faults that may exist in that process. Quantification problems are those involving the processing of multi-sensor information to describe the input space in order to extract the values of primary variables within that space. An example of this type of problem is that of measuring vehicle exhaust emissions using cross-sensitive sensors, where concentrations of component gases such as carbon monoxide and nitrous oxide need to be extracted and assigned a value, e. g. 25 parts per million [Corcoran and Lowery, 1995]. Description problems involve the extraction and presentation of meaningful features or concepts that are representative of the input space. An example of this kind of problem might be the calculation of the overall risk of a road-vehicle accident depending on multi-sensor data describing the vehicle speed, proximity to other vehicles, visibility and road conditions.

Due to their computational capabilities, NNs are becoming more widely used as processing tools to solve problems where multi-sensor information is involved. The reason is that NNs are able to provide a mechanism for the enhancement of the quality of information derived from multiple sensors.

2.6 Summary

MSDF algorithms can be classified by the kind of techniques used in the fusion process namely estimation methods, classification methods, inference methods, and artificial intelligence methods. In this chapter a review of the most popular MSDF algorithms in each class has been presented.

From the estimation methods, the weighted average method is a very intuitive and simple method. However, Kalman filtering is preferred because the solution it provides is optimal in a statistical sense and its computational requirements does not increase a lot compared with those of the weighted average method.

In classification methods the main task is to partition a multidimensional feature space into distinct regions, each representing an identification or identity class. Two popular methods for this kind of task are K-means clustering and the Kohonen feature map. K-means clustering algorithm is one of the commonly used unsupervised learning algorithms. While an adaptive K-means update rule forms the basis of the Kohonen feature map approach.

From inference methods, Bayesian inference allows multisensor information to be combined in accordance with the rules of probability theory. Particularly, Bayes' rule provides a relationship between the *a priori* probability of a hypothesis, the conditional probability of an observation given a hypothesis, and *a posteriori* probability of the hypothesis. Dempster-Shafer reasoning is an extension of the Bayesin inference approach that make explicit any lack of information concerning a proposition's probability by separating firm support for the proposition from just its plausibility.

Finally, from artificial intelligence methods, artificial neural networks can be trained to represent sensor information and, through associate recall, complex combinations of neurons can be activated in response to different sensory stimuli [Luo *et al*, 2002]. Fuzzy logic allows the uncertainty in multisensor fusion to be directly represented, by using fuzzy sets, in the inference process.

The selection of any algorithm in an actual application has to be made in accordance with the problem under consideration and having in mind the objective of the sensor fusion process. From all the methods presented in this chapter, Kalman filtering, fuzzy logic and neural networks are core to this thesis. Therefore, in chapter 3 a broader explanation of fuzzy systems, neural networks and neuro-fuzzy systems (the synergistic combination of fuzzy systems and neural networks) is given. While in chapter 4, a more detailed explanation of the Kalman filtering-based MSDF approaches is presented.

CHAPTER 3

NEURO-FUZZY SYSTEMS

3.1 Introduction

There are two concepts that are inherent to the human reasoning: imprecision and uncertainty. Because of that, our way of interpreting the world is generally done using vague propositions, uncertain data or appreciative judgements. However, this way of thinking is not captured in traditional logic and traditional computing. This fact has been perceived by several thinkers that in the past have tried to develop a mathematical structure capable of capturing this characteristic of the human way of thinking. As a result, several approaches have been devised and nowadays they are grouped in the so-called Soft Computing (SC) technology.

Soft computing (SC) is a term coined by Lotfi A. Zadeh [Zadeh, 1994], the father of fuzzy logic [Zadeh, 1965], to refer to systems that try to mimic the ability of the human mind to effectively employ modes of reasoning that are approximate rather than exact. In traditional (hard) computing any imprecision and uncertainty is considered as undesirable. By contrast in SC the aim is to design systems capable of exploiting the tolerance for imprecision and uncertainty, learning from examples, and adapting to changes in the operating conditions.

SC is not a technique alone, but a group of them. The principal members of SC are fuzzy logic (FL), neural networks (NN), genetic computing (GC), and probabilistic reasoning (PR) [Tsoukalas and Uhrig, 1997]. The main contributions of FL in SC are a methodology for dealing with imprecision, approximate reasoning, rule-based systems, and computing with words. NN contributes with system identification, learning, and adaptation. GC contributes with systematised random search and optimisation. PR contributes with decision making and management of uncertainty. Thus, these methodologies are synergistic and complementary rather than competitive. For this reason, it is advantageous to use them in different combinations, leading to the so-called “hybrid intelligent systems” [Jang *et al*, 1997]. Nowadays, the most visible and successful hybrid intelligent systems of this type are neuro-fuzzy systems.

Neuro-fuzzy systems integrate two complementary approaches: fuzzy logic and neural networks. On the one hand, neural networks are capable of recognising patterns and adapting themselves to cope with changing environments; if there is data available, or if it can be learned from a simulation or real task, then a neural network can be used. Secondly, fuzzy inference systems incorporate human knowledge and perform inferencing and decision making; if there is knowledge that can be expressed in rules, then a fuzzy system can be built. What neuro-fuzzy systems do is put together in a single methodology all the above characteristics. Therefore these characteristics are desirable in any MSDF architecture.

In the last chapter different techniques of MSDF were described. There were included those architectures which make use, separately, of fuzzy logic and neural networks technologies. However, as one of the objectives of this work is the development of intelligent MSDF using hybrid architectures, which include both these techniques, in this chapter fuzzy systems, neural networks, and neuro-fuzzy systems are broadly described. The concepts presented here will be used in later chapters.

3.2 Fuzzy inference systems

Imprecision and uncertainty are inherent concepts which pertain to the inexact nature of human reasoning. As a result, our way of interpreting the world is generally seen as a function of vague propositions, uncertain data and appreciative judgements. However, this way of thinking is not taken into account in traditional logic (here referred to as crisp logic), where only two fundamental premises exist: true and false, 0 and 1. Lotfi A. Zadeh noticed this and created a new logic, called fuzzy logic [Zadeh, 1973], in order to attempt to capture the uncertainty present in our reasoning when interpreting the world. This logic is based on the theory of *fuzzy sets* [Zadeh, 1965] proposed in his seminal paper of 1965.

The main purpose of fuzzy logic is to allow the use of vague concepts to characterize the variables of a system and its interrelations using words or propositions expressed in a natural or artificial language. This is possible because, in fuzzy sets theory, an object is no longer restricted to be totally a member or not a member of a set. Instead, an element may have a grade of membership intermediate between full membership and non-membership, in the whole range [0,1]. For example, if an object has a degree of membership of 0.7 to a particular set, the same element has a degree of membership of 0.3 to the complement of that set. Evidently, the theory of fuzzy sets is an extension of the traditional theory of sets, where this last case is included at the extremes. In other words, whereas in traditional logic a set has hard borders, in fuzzy logic the borders of a set are not sharply defined. Instead, the borders are soft allowing an object a smooth transition between being member or not a member of a particular set. Therefore, using fuzzy logic, systems can be designed to be able to capture, in the form of heuristic rules, the ability that all human beings possess to model a system or process using natural language.

In its origins, due to its name, fuzzy logic was considered as something obscure and without mathematical or logical foundation. Consequently, after Zadeh published his article, this new logic remained in the background. However, in 1973 Professor E. Mamdani at the University of London used, for the first time, fuzzy logic to design the automatic control of a small steam machine, giving the origin to fuzzy control systems (FCS) or fuzzy inference systems (FIS). Since the publication in 1975 of the results obtained by Mamdani [Mamdani and Assilian, 1975], FISs have had a great variety of applications ranging from industrial processes to home appliances. Nowadays, FISs are applied in a wide range of areas including automatic control, signal processing, time-series prediction, information retrieval, data classification, decision making, and so on.

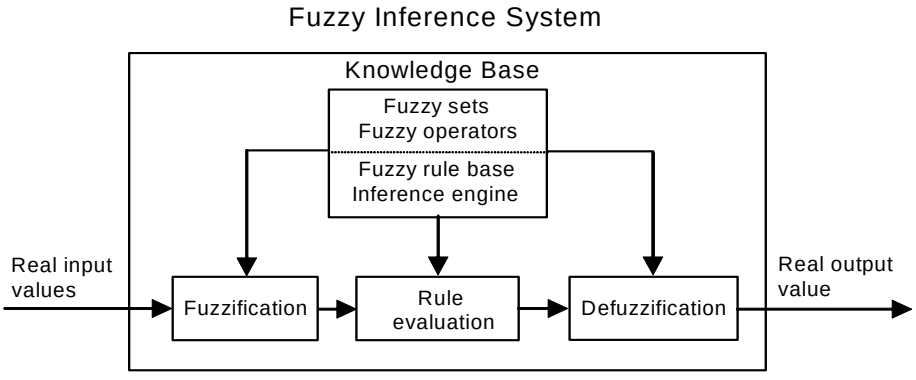


Figure 3.1 Basic structure of a FIS.

In general terms, a *fuzzy inference system* (FIS) is a computing framework based on the concepts of fuzzy logic [Jang et al, 1997]. The basic structure of a FIS is shown in figure 3.1. As can be seen, a FIS consists of 3 fundamental processes: *fuzzification, rule evaluation,* and *defuzzification.* All these processes are assisted by a *knowledge base* which comprises a fuzzy

rule base, an inference engine, fuzzy sets and fuzzy operators. In the following sections, a description of these processes is given.

3.2.1 Fuzzification process

The knowledge is represented inside a FIS through a fuzzy rule base and fuzzy sets. Thus, in order to perform inferences inside the FIS, using an inference engine and fuzzy operators, it is necessary to transform the real-valued input information into fuzzy sets. This transformation is carried out through a process known as *fuzzification*.

For simplicity, consider a multiple-input single-output (MISO) FIS: $U \subset R^n \rightarrow R$, where U is a compact universe of discourse; the process of *fuzzification* maps the real input vector $\boldsymbol{x} = (x_1, x_2,\ldots,x_n) \subset R^n$ to fuzzy sets defined in U. Where a fuzzy set F in a universe of discourse U is characterised by a membership function $\mu_F(\cdot): U \rightarrow [0,1]$, which associates with each element u of U a real number $\mu_F(u)$ that lies in the unit interval [0,1], with $\mu_F(u)$ representing the grade of membership of u in F. F may be a linguistic label such as *small, very small, big, very big*, etc. The *support* of F is the crisp set of points in U at which $\mu_F(u)>0$. A fuzzy set whose support is a single point in U with $\mu_F(\cdot) = 1.0$ is referred to as *fuzzy singleton* [Zadeh, 1965, 1973, 1977].

Specifically, if x is an input variable to the FIS, and $x = x_0 \in U$ is an input value, then the output of the process of fuzzification is a fuzzy set in U, $F = fuzzifier(x_0)$; where the operator *fuzzifier* transforms the real input value x_0 to a linguistic value or fuzzy set, F.

There are several different methods to develop the process of fuzzification, but two are the most popular:

a) Singleton fuzzification. This method maps the input x to a fuzzy singleton, F, with membership function:

$$\mu_F(x) = \begin{cases} 1 & if \;\; x = x_0 \\ 0 & in \;\; any \;\; other \;\; case \end{cases} \tag{3.1}$$

b) Approximated fuzzification:

$$\mu_F(x) \neq 0 \quad only \;\; and \;\; only \;\; if \quad |x - x_0| < \delta \tag{3.2}$$

where δ is a parameter that is determined in accordance with the context of each application.

In most applications reported in the literature the function $F = fuzzifier(x_0)$ takes the special form of $F = x_0$ for each measured value of the variable of interest [Murphy, 1991]. In other words, a crisp value at the input of a FIS is mapped to a singleton defined by the point $x_0 \in U$. Usually, if the input to the FIS is a measurement, then the fuzzification procedure used is the singleton one.

3.2.2 Process of rule evaluation

In general terms, the process of *rule evaluation* involves a fuzzy rule base and a fuzzy inference engine. A fuzzy rule base is integrated by a set of linguistic rules expressed in the form: *"**if** a set of conditions is satisfied, **then** a set of actions is taken"*. The part ***if*** of the rule is known as the antecedent, and the part ***then*** of the rule is known as the consequent.

The inference engine is an interpreter of the rule base; its task is the calculation of a fuzzy conclusion from a set of fuzzy if-then rules and one or more conditions. This fuzzy conclusion is obtained employing an inference mechanism called *approximate reasoning*, derived from fuzzy logic theory. The basic inference rule of this reasoning is the *generalised modus ponens* (GMP) [Lee, 1990] [Jang and Sun, 1995]. For example, consider a multiple-input-single-output (MISO) FIS with two input variables and fuzzy rule base of the form:

$$R_j: \text{If } x \text{ is } A_j \text{ and } y \text{ is } B_j \text{ then } z \text{ is } C_j \tag{3.3}$$

where j=1,2,...,m; m = number of rules; x, y and z are linguistic variables; A_j and B_j are linguistic values of the linguistic variables x, y and z in the universes of discourse U, V and W, and characterised by the membership functions $\mu_{A_j}(x)$, $\mu_{B_j}(y)$ and $\mu_{C_j}(z)$, respectively.

Now, suppose that the rule base (3.3) includes a single rule written as "if x is A and y is B then z is C ". Thus, the corresponding problem for GMP is expressed as:

Premise 1 (fact):	x is A' and y is B'	
Premise 2 (rule):	If x is A and y is B then z is C	(3.4).
Consequence (conclusion):	z is C'.	

where A' is close to A, B' is close to B, and C' is close to C. Note that when $A' = A$, $B' = B$, and $C' = C$ the GMP reduces to the traditional *modus ponens*.

The rule in premise 2 above can be implemented as a fuzzy relation (or implication). This relation is written as:

$$R = A \times B \rightarrow C \in U \times V \times W \tag{3.5},$$

which membership function is specified by [Jang and Sun, 1995]:

$$\begin{aligned} \mu_R(x, y, z) &= \mu_{(A\times B)\rightarrow C}(x, y, z) \\ &= \mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z) \end{aligned} \tag{3.6}$$

where the symbol $\wedge$ is used to denote the fuzzy operator of intersection, or fuzzy AND (a general description of the main operations on fuzzy sets is given in Appendix A).

Applying the compositional rule of inference [Zadeh, 1973], the fuzzy conclusion C' of the inference procedure is expressed as:

$$\begin{aligned} C' &= (A' \times B') \circ R \\ &= (A' \times B') \circ (A \times B \rightarrow C) \end{aligned} \tag{3.7},$$

where $\circ$ denotes the composition operator. Thus, using (3.6), the membership function of C' is evaluated as:

$$\begin{aligned} \mu_{C'}(z) &= \vee_{x,y}[\mu_{A'}(x) \wedge \mu_{B'}(y)] \wedge [\mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z)] \\ &= \vee_{x,y}[\mu_{A'}(x) \wedge \mu_{B'}(y) \wedge \mu_A(x) \wedge \mu_B(y)] \wedge \mu_C(z) \\ &= \{\vee_x[\mu_{A'}(x) \wedge \mu_A(x)]\} \wedge \{\vee_y[\mu_{B'}(y) \wedge \mu_B(y)]\} \wedge \mu_C(z) \end{aligned} \tag{3.8}$$

where the symbol $\vee$ is used to denote the fuzzy operator of union, or fuzzy OR.

At this point, let define:

$$w_A = \vee_x[\mu_{A'}(x) \wedge \mu_A(x)] \tag{3.9a}$$

$$w_B = \vee_y[\mu_{B'}(y) \wedge \mu_B(y)] \tag{3.9b}$$

where w_A represents the degree of compatibility between A and A'; similarly, w_B represents the degree of compatibility between B and B'. Substituting (3.9) in (3.8) results in:

$$\mu_{C'}(z) = w_A \wedge w_B \wedge \mu_C(z) \tag{3.10}$$

but, if it is defined $w = w_A \wedge w_B$, then (3.10) transforms to:

$$\mu_{C'}(z) = w \wedge \mu_C(z) \tag{3.11}.$$

In (3.11) w is called the *firing strength* or *degree of fulfilment* of this rule, and it represents the degree to which the antecedent part of the rule is satisfied [Jang and Sun, 1995]. A graphical interpretation of this result is shown in figure 3.2 when the fuzzy operators for union and intersection are selected to be the maximum (*max*) and minimum (*min*), respectively. In this case $\circ$ is called the *max-min* composition operator [Jang *et al*, 1997], and the whole inference procedure is called the *max-min compositional rule of inference* [Brown and Harris, 1994]. Observe in figure 3.2 that the resulting membership function for C' is equal to the membership function of C clipped by the firing strength w.

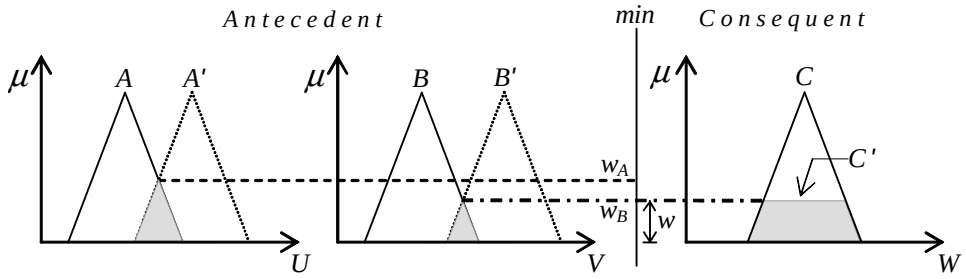


Figure 3.2 Approximate reasoning for a rule with two antecedents.

The previous development can be extended for the case of a rule base with m rules as is detailed next. For a MISO system with two inputs and one output, the problem for GMP is expressed as:

fact:	x is A' and y is B'
rule 1:	If x is A_1 and y is B_1 then z is C_1
rule 2:	If x is A_2 and y is B_2 then z is C_2
	…
	…
rule m:	If x is A_m and y is B_m then z is C_m
conclusion:	z is C'.

(3.12).

Therefore, each rule in (3.12) can be implemented as a fuzzy relation:

$$R_j = A_j \times B_j \rightarrow C_j \qquad \text{for } j = 1,2,\ldots,m \tag{3.13}$$

Applying the *max-min* compositional rule of inference, and using its characteristic of being distributive over the $\vee$ operator [Lee, 1990], the fuzzy conclusion C' of the inference procedure (3.12) is expressed as:

$$\begin{aligned} C' &= (A' \times B') \circ (R_1 \vee R_2 \vee \cdots \vee R_m) \\ &= [(A' \times B') \circ R_1] \vee [(A' \times B') \circ R_2] \vee \cdots \vee [(A' \times B') \circ R_m] \\ &= C'_1 \vee C'_2 \vee \cdots \vee C'_m \end{aligned} \tag{3.14}.$$

where C'_j is the inferred fuzzy set for rule *j*. Then, using the result given by (3.11) the membership function of each fuzzy set C'_j is obtained as:

$$\mu_{C'_j}(z) = [w_j \wedge \mu_{C_j}(z)] \qquad \text{for } j = 1,2,\ldots,m \tag{3.15}$$

Finally, the membership function of the resulting fuzzy set C' inferred from the complete set of fuzzy rules is given by the union of the resulting conclusion derived from individual rules,

$$\mu_{C'}(z) = [w_1 \wedge \mu_{C_1}(z)] \vee [w_2 \wedge \mu_{C_2}(z)] \vee \cdots \vee [w_m \wedge \mu_{C_m}(z)] \tag{3.16}$$

where w_j indicates the degree of fulfilment of the *j*-th rule; $\mu_{C_j}(z)$ is the membership function of the fuzzy set C_j (j=1,2,...,m; m = number of rules). Figure 3.3 shows a graphical representation of the operation of fuzzy reasoning for the case described. Note, that in this case the singleton fuzzification procedure has been used to transform the inputs x_0 and y_0 into fuzzy sets.

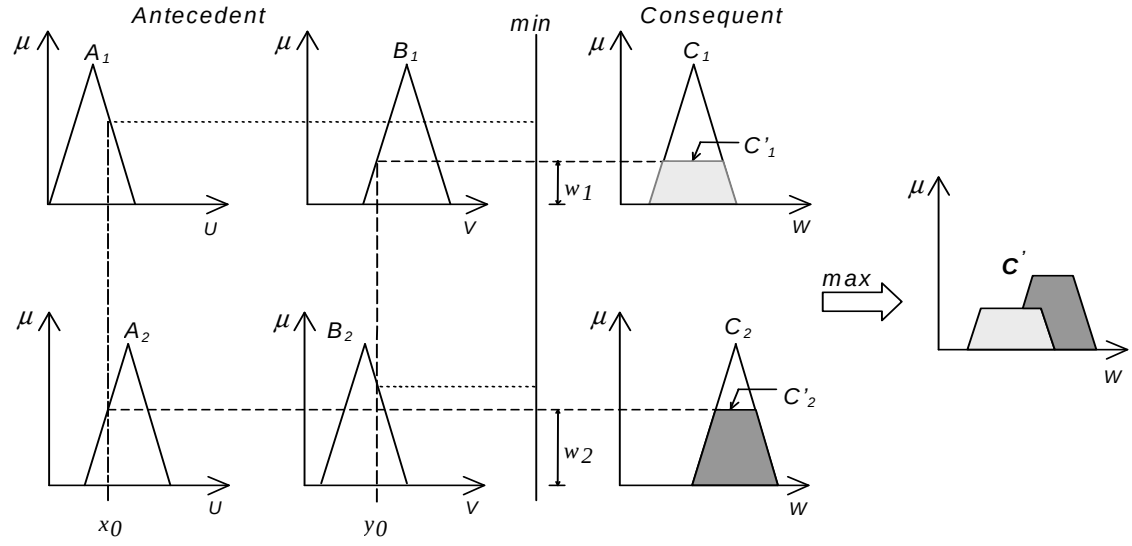


Figure 3.3 Graphical representation of the fuzzy reasoning procedure for multiple rules with multiple antecedents and *max-min compositional rule of inference* being used.

In the previous development the *max* and *min* operators were adopted for fuzzy union and fuzzy intersection, respectively. But, in fact, any S-norm and T-norm (see Appendix A) can be adopted for these fuzzy operations [Lee, 1990]. The selection of a specific S-norm and T-norm defines a specific type of fuzzy reasoning and gives its name to the compositional rule of

inference. From the possible types of fuzzy reasoning, three are the most used: *max-min*, *max-product* [Jang and Sun, 1990], and *sum-product* [Kosko, 1991], a resume is given in Table 3.1.

Table 3.1 Types of fuzzy reasoning most used.

Composition operator	Union operator	Intersection operator	Name
max-min	*max*	*min*	*Max-min compositional rule of inference*
max-product	*max*	*product*	*Max-product compositional rule of inference*
sum-product	*sum*	*product*	*Sum-product compositional rule of inference*

If the operators *max* and *product* (arithmetic product) are chosen for fuzzy union and intersection, respectively, then the composition is called the *max-product compositional rule of inference*. The calculation of the fuzzy conclusion for this case is obtained as:

$$\mu_{C'}(z) = [w_1 \cdot \mu_{C_1}(z)] \vee [w_2 \cdot \mu_{C_2}(z)] \vee \cdots \vee [w_m \cdot \mu_{C_m}(z)] \tag{3.17}.$$

Figure 3.4 shows a graphical representation of the fuzzy reasoning operation when the *max-product* compositional rule of inference is used. Note, that the singleton fuzzification procedure has been used to transform the inputs to fuzzy sets.

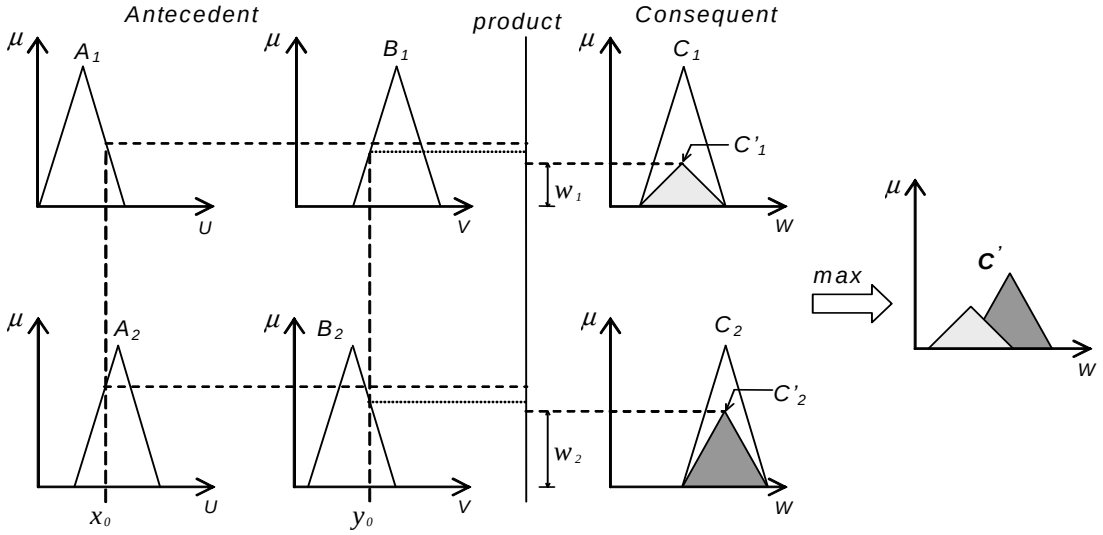


Figure 3.4 Graphical representation of the fuzzy reasoning for multiple rules with multiple antecedents and *max-product compositional rule of inference* being used.

If the operators *sum* (arithmetic summation) and *product* are chosen for fuzzy union and intersection, respectively, then the composition is called the *sum-product compositional rule of inference*. In this case the fuzzy conclusion is obtained as:

$$\mu_{C'}(z) = \sum_{j=1}^{m} w_j \cdot \mu_{C_j}(z) \tag{3.18}.$$

Figure 3.5 shows graphically the fuzzy reasoning operation when the *sum-product* compositional rule of inference is used. Note, that the singleton fuzzification procedure has been used to transform the inputs to fuzzy sets.

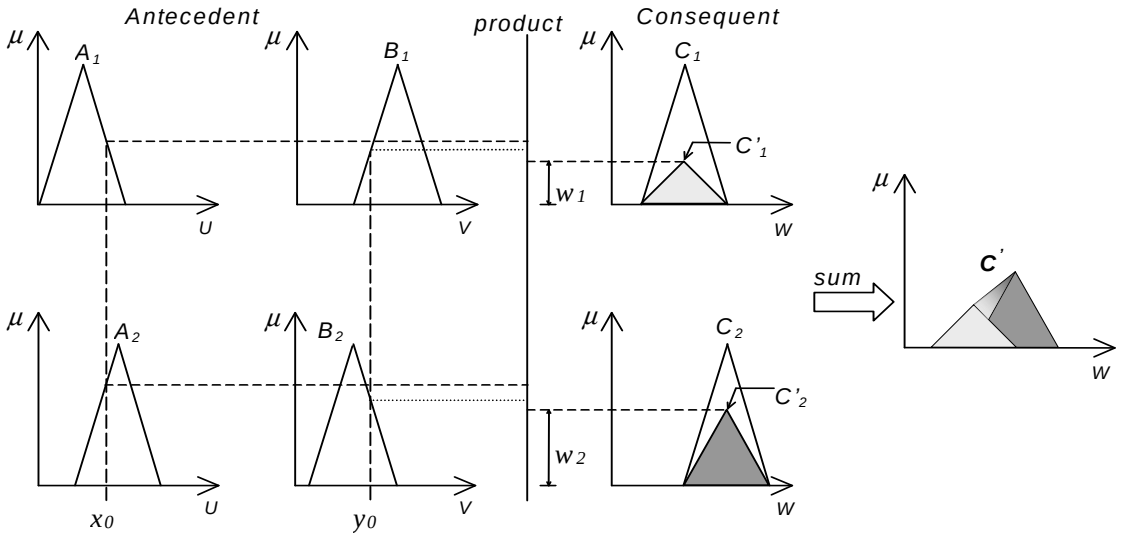


Figure 3.5 Fuzzy reasoning for multiple rules with multiple antecedents and sum-product compositional rule of inference being used.

3.2.3 Defuzzification process

Basically, the defuzzification process is a mapping from a space of fuzzy sets defined over an output universe of discourse into a space of crisp (non-fuzzy) values. In other words, the defuzzification process transforms the fuzzy conclusion *C´* into a crisp and concrete value z_0, which is given as the FIS output. In general there are several methods to perform the process of defuzzification [Lee, 1990] [Jang *et al*, 1997] [Driankov *et al*, 1993]; the most commonly used are described next.

• *Centre of Area (COA)*. The most often used of the defuzzification methods is the centre of area method. This method obtains the crisp output value applying the following formula:

$$z_0 = \frac{\int_W \mu_{C'}(z)\ z\ dz}{\int_W \mu_{C'}(z)\ dz} \tag{3.19}$$

where $\mu_{C'}(z)$ is the aggregated output membership function.

• *Bisector of area (BOA)*. The BOA defuzzification method satisfies:

$$\int_{\alpha}^{z_0} \mu_{C'}(z)dz = \int_{z_0}^{\beta} \mu_{C'}(z)dz \tag{3.20}$$

where $\alpha = \min\{z \mid z \in W\}$ and $\beta = \max\{z \mid z \in W\}$. That is, the vertical line $z = z_0$ partitions the region between $z = \alpha$, $z = \beta$, $y = 0$ and $y = \mu_{C'}(z)$ into two regions with the same area.

• *Mean of Maximum (MOM)*. The MOM method calculates a crisp output value by averaging the support values of the inferred fuzzy set *C´*, at which membership value reach a maximum μ^*. Mathematically, this is expressed as:

$$z_0 = \frac{\int_{Z'} z \; dz}{\int_{Z'} dz} \tag{3.21}$$

where $Z' = \{z \mid \mu_{C'}(z) = \mu^*\}$. In particular, if $\mu_{C'}(z)$ has a single maximum at $z = z^*$, then $z_0 = z^*$.

• *Smallest of Maximum (SOM).* The SOM defuzzification method gives as crisp output the minimum (in terms of magnitude) of the maximising *z*.

• *Largest of Maximum (LOM).* The LOM defuzzification method gives as crisp output the maximum (in terms of magnitude) of the maximising *z*. because their obvious bias the SOM and LOM defuzzification methods are not used as often as the other three methods.

Figure 3.6 shows a graphic comparison of the different FIS crisp outputs obtained with each defuzzification method for a given fuzzy set C'.

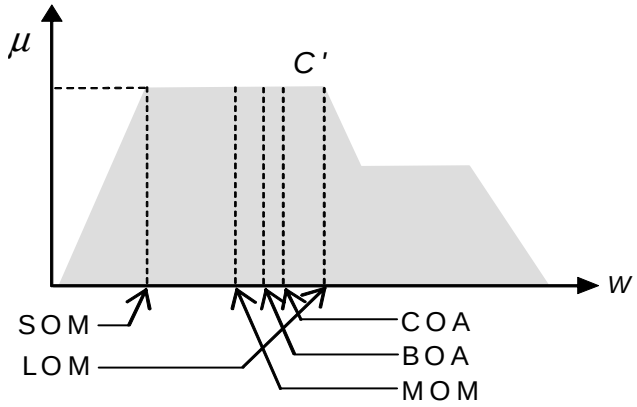


Figure 3.6 Comparison of the crisp FIS output obtained with the different defuzzification methods.

3.2.4 Types of fuzzy inference systems

In what follows, the three types of most commonly used FISs are introduced:

1) *Mamdani FIS Model:* The first FIS proposed was developed by Mamdani and Assilian in 1975 [Mamdani and Assilian, 1975]. This type of FIS was designed as a controller for a steam engine and boiler combination using a set of linguistic "if-then" control rules obtained from experienced human operators. The distinctive characteristic of this type of FIS is that in both antecedent and consequent parts of the rules, the values of the variables used are defined by membership functions, where the most commonly used are the triangular, trapezoidal, and Gaussian membership function. The type of reasoning used in this type of FIS is the *max-min* compositional rule of inference, and the type of defuzzification method used is the COA method, as it is graphically represented in figure 3.3.

The original Mamdani FIS model has been modified in different ways. Two of the most used variations are those where the *max-min* composition operator is substituted by the *max-product* and *sum-product* composition operators. These cases were previously discussed in section 3.2.2 and are graphically illustrated in figures 3.4 and 3.5, respectively.

2) *Sugeno FIS Model:* The Sugeno FIS model (also known as *TSK fuzzy model*) was proposed by Takagi, Sugeno and Kang [Sugeno and Kang, 1988] [Takagi and Sugeno, 1985] as an effort to develop a systematic approach to generate fuzzy rules from a given input-output data set. The main characteristic of this type of FIS is that the fuzzy rules used have the form:

$$\text{if } x \text{ is } A \text{ and } y \text{ is } B \text{ then } z = f(x,y) \tag{3.22}$$

where x, y and z are linguistic variables, A and B are fuzzy sets in the antecedent, and $z = f(x,y)$ is a crisp function in the consequent.

Generally, $f(\cdot)$ is defined as a polynomial in the input variables x and y, but it can be any function appropriately defined in order to describe the output of the system within the fuzzy region specified by the antecedent of the rule. If $f(\cdot)$ is a first-order polynomial then the FIS is called a *first-order Sugeno FIS model*. Whereas if $f(\cdot)$ is defined as a constant, then the FIS is called *a zero-order Sugeno FIS model*. The zero-order Sugeno FIS model can be interpreted as a special case of the Mandani FIS model, in which each rule consequent is specified by a fuzzy singleton. The zero-order Sugeno FIS model as well can be interpreted as a special case of the Tsukamoto fuzzy model (described later); in which each rule consequent is specified by a membership function given as a step function crossing at the consequent. A special characteristic of the zero-order Sugeno FIS model is that it has been proven, under certain constraints, to be functionally equivalent to a radial basis function network [Jang and Sun, 1993]. Another characteristic of the zero-order Sugeno FIS model is that the smoothness of the resulting input-output behaviour decisively depends on the existence of enough overlap between membership functions in the antecedent of the rules.

The overall output of a Sugeno FIS model is obtained via a weighted average of the crisp outputs given by the fired rules, as is graphically represented in figure 3.7 for a first-order Sugeno model. Note that, using a weighted average, the time-consuming nature of the defuzzification procedure is enormously reduced.

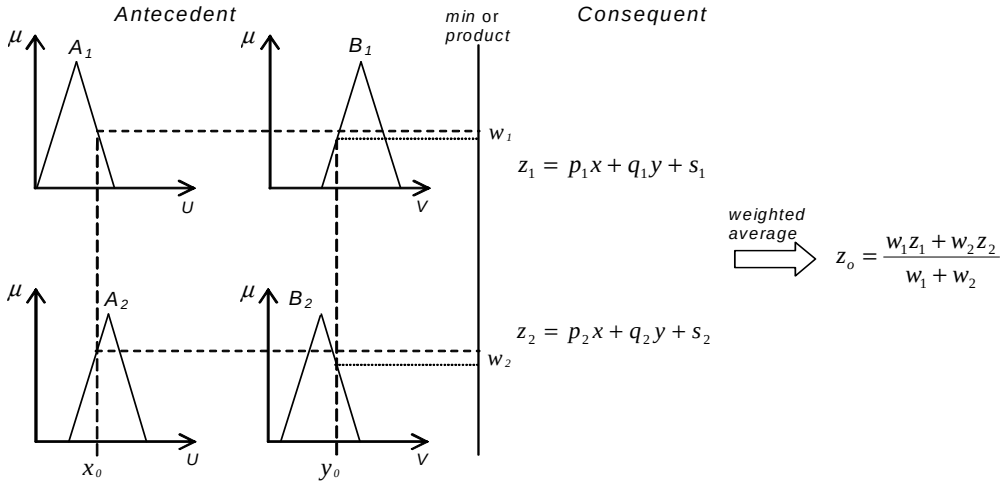


Figure 3.7 Reasoning in a first-order Sugeno fuzzy model.

3) *Tsukamoto FIS model:* In the Tsukamoto FIS model the consequent of each fuzzy rule is represented by a fuzzy set with a monotonical membership function [Tsukamoto, 1979], as is shown in figure 3.7. As a result, the inferred output of each rule is defined as a crisp value induced by the rule's firing strength. The overall output is taken as the weighted average of the outputs given by the fired rules. Figure 3.7 illustrates the whole reasoning procedure for a two-input system with two rules being firing.

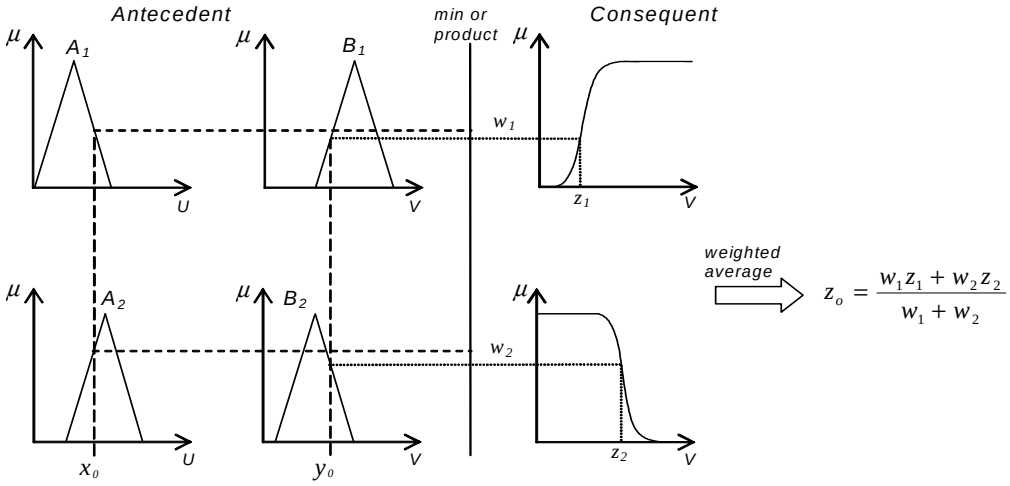


Figure 3.8 Reasoning in a Tsukamoto FIS model.

Up to this point, the processes involved in a FIS have been described, together with the three different types of FIS found in the literature. In subsequent sections the concepts of artificial neural networks, or simply neural networks, and the so-called neuro-fuzzy systems, the synergistic combination of neural networks and fuzzy systems, will be explained.

3.3 Artificial neural networks

An artificial neural network or simply a neural network (NN) is defined as a collection of processing elements (called neurons) and connection weights (generally denoted as *w*). These neurons and weights are structured to perform a mapping from an input space to an output space $R^n \rightarrow R^m$ [Gupta and Rao, 1994]. This mapping can be very simple and linear or can be very complex and non-linear; it only depends on the structure of the NN and the functionality of each neuron. Nowadays there are a lot of morphologies of NNs and many others are being investigated. As an example, from a structural point of view, by their architecture NNs can be classified as static or dynamic, with only one level or several. By their connections, NNs can be classified as feedforward NNs, feedback NNs, laterally connected, topologically ordered and hybrids. For a more detailed description of NNs morphologies see [Gupta and Rao, 1994]. Figure 3.9 shows a typical feedforward NN, which in practice is one of the most used.

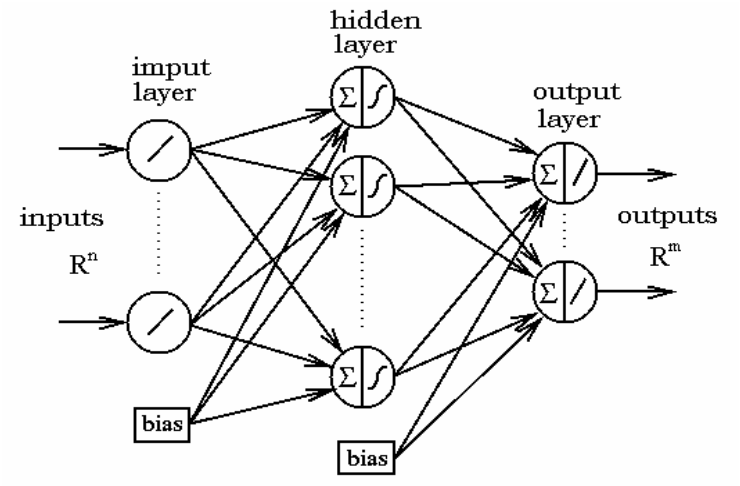


Figure 3.9 A typical feedforward NN.

As is shown in figure 3.9, a NN can have several layers, and each layer can have more than one neuron. The main function of each neuron in a NN is the collection of all its weighted inputs, an evaluation of a predefined mathematical operation, usually a dot product followed by a non-linear function, and the production of a single output. In other words, a neuron performs a mapping from R^n to R^1. Mathematically, the transformation carried out by a neuron can be described by the equation:

$$y_k = \Psi\left[\sum_{j=1}^{n} w_{kj} x_j - b_k\right] \tag{3.23}$$

where: $x_1,...,x_n$ are the inputs to the neuron; $w_{k1},...,w_{kn}$ are the connection weights for the inputs (the first subscript refers to the neuron in question and the second subscript refers to the input to which the weight is connected); y_k is the neuron output; $\Psi[\cdot]$ is some activation function [Haykin, 1999]. Generally, the activation function is non-linear and has a threshold or bias b_k. If we define $x_0 = +1$ and $w_{k0} = b_k$, then (3.23) can be rewritten as:

$$y_k = \Psi\left[\sum_{j=0}^{n} w_{kj} x_j\right] \tag{3.24},$$

and a graphical representation of the transformation carried out by a neuron is shown in figure 3.10.

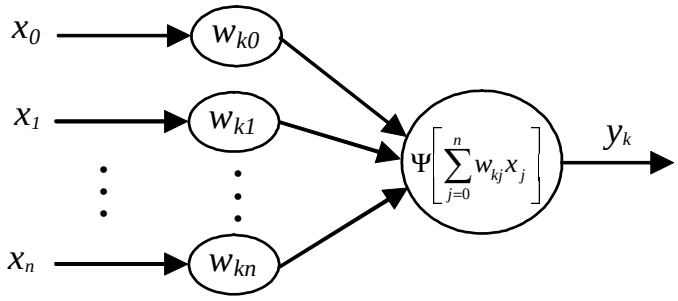


Figure 3.10 Graphical representation of the transformation carried out by a neuron.

The main characteristic of a NN is the storage of knowledge in the connection weights w_{kj}. This knowledge is acquired using a procedure called the *learning algorithm*. This algorithm successively adjusts the connection weights to find those values for which a better approximation is obtained to the desired output.

Broadly, learning algorithms can be classified as error-based (also known as supervised) or output-based (also known as unsupervised) [Gupta and Rao, 1994]. Error-based learning algorithms use an external reference signal (teacher) to generate an error signal from the comparison between the reference signal and the obtained response. Based on this error signal a NN adjusts its connection weights to improve the system performance. In this case it is assumed, a *priori*, that a desired answer is available. This desired answer is a set of training data (a pattern of input-output pairs) which is used to *train* the NN. A graphical representation of the error based learning scheme is shown in figure 3.11.

The general equation for the error-based learning algorithm is of the form:

$$w_{kj}(t+1) = w_{kj}(t) + \Delta w_{kj}(t) \tag{3.25}$$

with:

$$\Delta w_{kj}(t+1) = \eta\ x_j(t) e_k(t) \tag{3.26}$$

$$e_k(t) = y_d(t) - y_k(t) \tag{3.27}$$

where $w_{kj}(t)$ is the connection weight corresponding to the input $x_j(t)$. The parameter $\Delta w_{kj}(t)$ is the change in the connection weight $w_{kj}(t)$ (the adjustment) over an instant in time, η is a parameter called learning rate, $y_d(t)$ is the desired neural output, $y_k(t)$ is the actual neural output, and $e_k(t)$ is called the *error signal*. The back propagation algorithm [Rumelhart *et al*, 1986] is the most popular of this kind of learning algorithm.

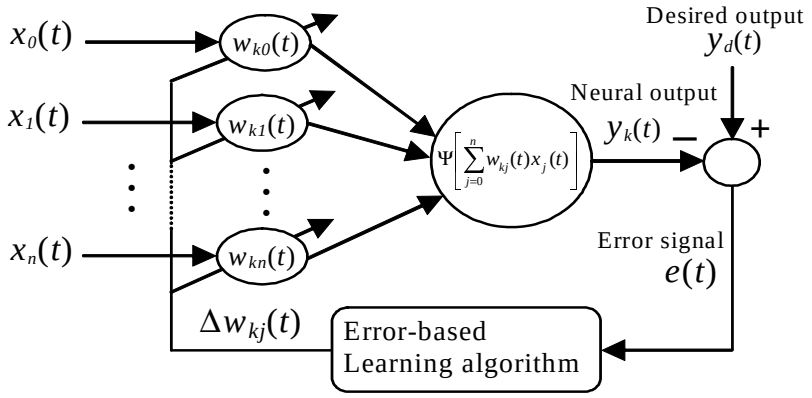


Figure 3.11 Error based (supervised) learning scheme.

On the other hand, output-based learning algorithms do not incorporate a reference signal. They generally involve a self-organising principle that relies only upon local information and internal control mechanisms in order to discover emergent collective properties. The two most important forms of this kind of learning algorithm are Hebbian learning and competitive learning [Gupta and Rao, 1994] [Haykin, 1999]. For space reasons, these algorithms are not described here, but the interested reader is referred to the cited references for a broader explanation of them.

The network structure, which includes many neurons and connection weights, is what gives a NN its computational capabilities. The arrangement of neurons in layers or stages of processing is supposed to mimic the layered structure of a certain portion of the human brain. This scheme of multilayer NN (MNN) has tested better computational capabilities than the one with a single layer. In particular, MNNs, which use the error back propagation learning algorithm, have been successfully applied in several areas including pattern recognition, system identification, and control systems.

Up to this point, NN have been described in general. In what follows next, the different combinations of neural networks and fuzzy inference systems are discussed.

3.4 Neuro-fuzzy systems

In general terms, a neuro-fuzzy system is a system in which fuzzy inference systems (FIS) and neural networks (NN) are used in combination. The main idea of this combination is to take advantage of the different characteristics that each approach has. It has been shown in distinct

ways that a feedforward NN and a special kind of FIS are universal approximators [Blum and Li, 1991][Castro, 1995][Hornik, 1989][Kosko, 1994][Wang, 1992]. Thus, both FISs and NNs solve problems by performing function approximation. On one side, if there is available knowledge expressed in rules, then a FIS can be built. On the other side, if data exists, or learning can be acquired from a simulation or real task, then a neural network can be used. Both techniques try to model expert behaviour.

A NN can be applied to a problem if there are valid training data (examples of input/output pairs). In this case supervised learning, such as back propagation in multilayer NN, is used to train the network to solve the problem. If nothing is known about valid outputs, but there is an evolution or error measure on the effects that are caused by the outputs of the neural network, then unsupervised learning can be used in order to adjust the parameters of the net and solve, in this way, the problem. A mathematical model of the problem of interest is not needed, and nor is any form of prior knowledge. However, an interpretation of the solution obtained from the learning process cannot be extracted. Thus, the neural network is a black box. Another disadvantage is the impossibility of adding or initialising a neural network with prior knowledge, if there is any. The learning process itself can take a very long time, and there is no guarantee of success.

An advantageous property attributed to neural networks is their fault tolerance regarding their inputs and changes in their structure. However, if the problem under investigation changes too much compared to the former training data, then the neural network may not be able to cope and retraining will be necessary.

On the other hand, a FIS can be used to solve a problem if there is knowledge about the solution in the form of linguistic if-then rules. In this case, suitable fuzzy sets are defined to represent linguistic terms; such as big, very big, slow, fast etc. These terms are used within the rules and the FIS is created from these rules. A formal model of the problem of interest is not needed and also training data is not needed. However, without if-then rules (maybe formulated by an expert) or data from which they can be derived, the fuzzy system cannot be created. Additionally, to make the FIS work properly, a long tuning process may be necessary. By way of contrast, fuzzy systems are also considered fault-tolerant regarding small changes in their inputs or system parameters.

FIS and NN have had, individually, an enormous success in the solution of many and varied tasks. This, taken with the characteristics mentioned above, has caused a proliferation in the engineering literature of many papers that describe or use distinct combinations of FIS and NN. Thus, in order to clarify the approach, a classification of them is needed. Nauck *et al* [1997] suggest that these combinations can be classified into four branches: 1) Fuzzy neural networks systems, 2) Concurrent neural/fuzzy systems, 3) Cooperative neuro-fuzzy models, and 4) Hybrid neuro-fuzzy models. A short description of each one of these systems is given below.

3.4.1 Fuzzy neural networks systems

In this combination fuzzy methods are used in NNs to learn faster or perform better. The main objective of this kind of system is to enhance the learning capabilities or the performance of a NN. This can be done, by using fuzzy rules to change the learning rate [Haykin, 1994] or by creating a network that works with fuzzy inputs and fuzzy logic operators [Ishibuchi *et al*., 1995]. A fuzzy NN has the same structure as a NN, but some or all of its components and parameters may be described through the mathematics of fuzzy logic theory. Thus, there are many possibilities for fuzzification of a NN and hence a variety of them have been proposed [Tsoukalas and Uhrig, 1997] [Gupta, 1994]. The obtained system cannot be interpreted in terms of fuzzy if-then rules, because the system is based on NNs with black box characteristics.

3.4.2 Concurrent neural/fuzzy systems

Generally, in this type of system a NN is used to pre-process or post-process the information coming in to, or coming out from a FIS. Both systems work together on the same task, but without influencing each other. This means that a system is not used to determine the parameters of the other and each one can be identified separately. In figure 3.12 is shown a concurrent neural/fuzzy system where a NN is used as a pre-processor for a FIS; while in figure 3.13 is shown a concurrent neural/fuzzy system where a NN is used as a post-processor for a FIS.

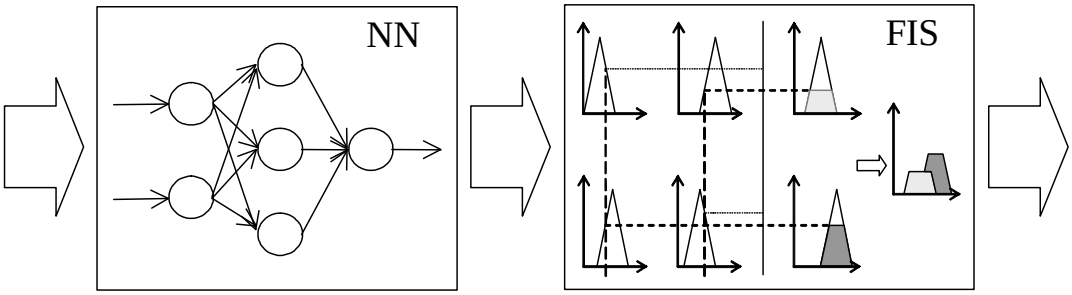


Figure 3.12 Concurrent neural/fuzzy system where a NN is used as pre-processor for a FIS.

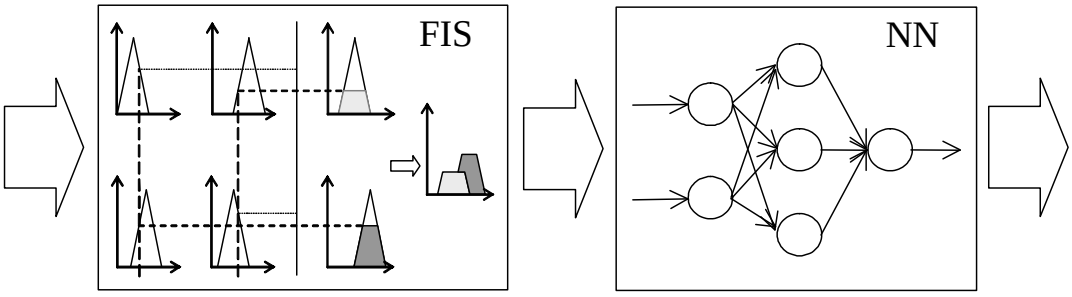


Figure 3.13 Concurrent neural/fuzzy system where a NN is used as a post-processor for a FIS.

Concurrent neural/fuzzy systems, where a NN is used as pre-processor for a FIS are suited for applications where the input variables of a FIS cannot be measured directly, so they have to be created by a combination of several values. A NN can be used as an adaptive information compressor [Nauck *et al*., 1997]. On the other hand, concurrent neural/fuzzy systems where a NN is used as a post-processor for a FIS can be used for cases where the output of a FIS cannot be applied directly to a process. In this case it may be necessary to combine the FIS output with other parameters. Then, a NN can be used to perform this combination.

3.4.3 Cooperative neuro-fuzzy models

In cooperative neuro-fuzzy systems, a NN or a NN learning algorithm has as objective the determination of certain parameters of a FIS (rules, rule weights and/or fuzzy sets). When the learning phase finishes the FIS can work without the NN. Subsequently, cooperative neuro-fuzzy models can be divided into four approaches [Nauck *et al*., 1997]: a) cooperative neuro-fuzzy systems that learn fuzzy sets offline, b) cooperative neuro-fuzzy systems that learn fuzzy rules offline, c) cooperative neuro-fuzzy systems that learn fuzzy sets online, and d) cooperative neuro-fuzzy systems that learn rule weights. A short description of each one of these approaches is given below.

3.4.3.a Cooperative neuro-fuzzy systems that learn fuzzy sets offline

In this class of cooperative systems training data is used by a NN to determine the membership functions of a FIS (see figure 3.14). The function of the NN is to find suitable parameters to define the membership functions or to perform an approximation of the fuzzy sets. The obtained fuzzy sets are used together with fuzzy rules, given separately, to build the FIS. In this case the set of training data is a set of degrees of membership corresponding to specific input values.

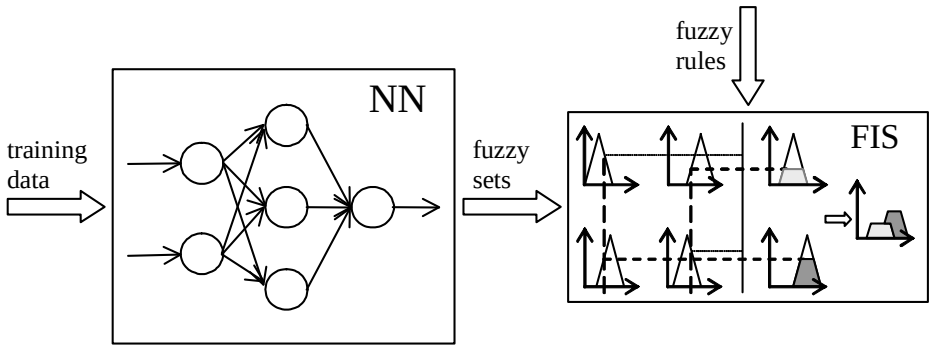


Figure 3.14 Cooperative neuro-fuzzy system that learns fuzzy sets offline.

3.4.3.b Cooperative neuro-fuzzy systems that learn fuzzy rules offline

In this case a NN is used to determine the fuzzy rules of a FIS. Training data is used by a NN to accomplish a clustering approach. A self-organising feature map or a similar architecture is generally used to learn the rules offline. Once the rules have been learnt, they are used together with fuzzy sets, provided separately, to implement a FIS. Figure 3.15 shows a system of this type.

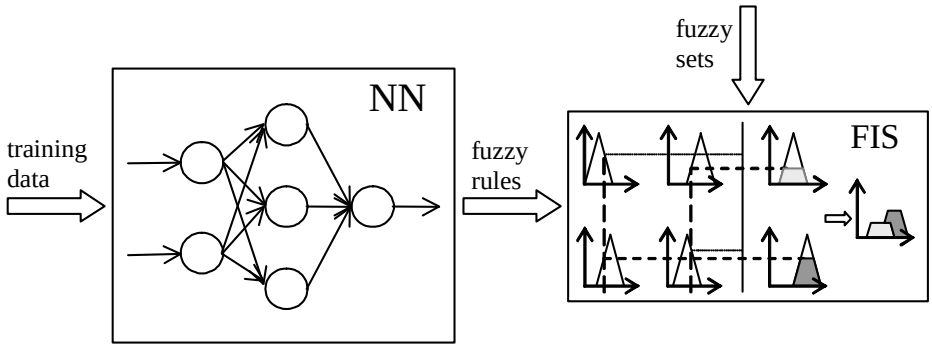


Figure 3.15 Cooperative neuro-fuzzy system that learns fuzzy rules offline.

3.4.3.c Cooperative neuro-fuzzy systems that learn fuzzy sets online

In this type of cooperative system, the parameters that define the fuzzy sets for a FIS are determined online. This can be carried out during the use of the FIS to adapt the membership functions. Initial membership functions have to be specified, and an error measure that guides the learning process of the NN is also needed. Usually, the whole NN is not present, only the neural learning algorithm is employed. Figure 3.16 shows a graphic representation of this type of cooperative neuro-fuzzy system.

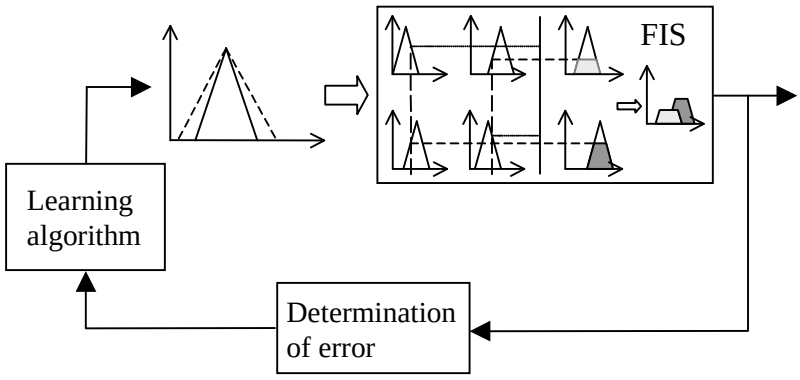


Figure 3.16 Cooperative neuro-fuzzy system that learns fuzzy sets online.

3.4.3.d Cooperative neuro-fuzzy systems that learn rule weights

In this case a NN or a learning algorithm is used in order to determine rule weights for the fuzzy rules of a FIS. This task can be performed online or offline. These weights can have several interpretations. For example Kosko [1992] defines them as rule influences, while Brown and Harris [1994] as rule confidences. Whatever the name, the main idea is to give a weight to each rule in the rule base and adjust them to obtain a better performance of the FIS. In figure 3.17 a system of this type is represented.

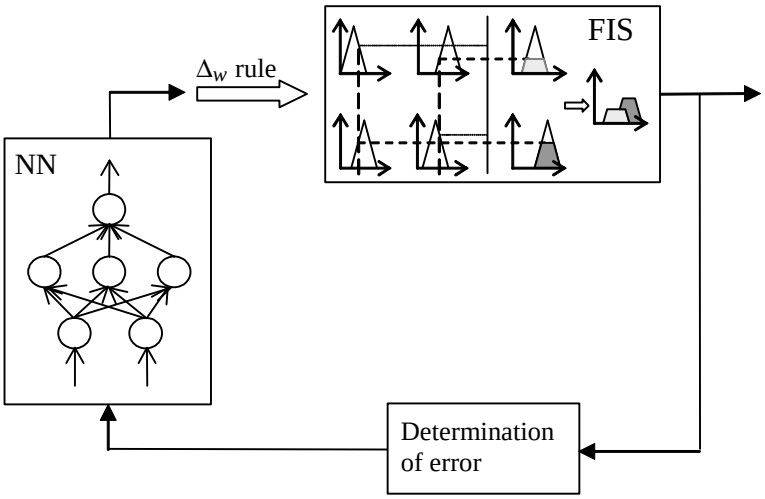


Figure 3.17 Cooperative neuro-fuzzy system that learns rule weights.

3.4.4 Hybrid neuro-fuzzy models

In these systems, a NN and a FIS are combined in a homogeneous way. This means that the obtained system cannot be divided and it can be interpreted either as a special NN with fuzzy parameters, or as a FIS arranged in a parallel distributed manner. The main idea of a hybrid neuro-fuzzy system is to represent a FIS as a special neural-network-like architecture and then apply a learning algorithm, such as back propagation or normalised least mean squares (NLMS) directly to train the system. Examples of this kind of architecture are the ANFIS system [Jang, 1993] [Jang and Sun, 1995] and the neuro-fuzzy network with B-splines used to implement fuzzy sets [Brown and Harris, 1994] [Harris *et al*, 1996]. A special characteristic of these

systems is that, once the system has been trained, the solution found can be interpreted in the form of linguistic if-then rules.

Nowadays the most successful of the four types of neuro-fuzzy systems described is the hybrid neuro-fuzzy model. However, because of the existence of many kinds of NN and the increased interest in researching in this area, it is hoped that more neuro-fuzzy models will appear in the near future. The main objective of these systems is to increase the intelligence capability of automatic systems. There, the concept of "intelligence" is defined as the capability of a system for learning and adaptation [Harris et al 2002].

A hybrid neuro-fuzzy system which is of great interest for the purposes of this work is the one which uses B-splines to implement fuzzy sets. In the next section, this hybrid neuro-fuzzy system is described and its advantages over other approaches are given. This hybrid neuro-fuzzy system will be used in Chapter 7 to design an adaptive MSDF architecture.

3.5 B-spline based hybrid neuro-fuzzy systems

The B-spline based hybrid neuro-fuzzy approach was originally proposed and developed by Brown and Harris [Brown and Harris, 1994]. In their early work, they established the first links between NNs and FISs, producing the first hybrid neuro-fuzzy adaptive system, which has the linguistic transparency of FISs coupled with the analytical tractability of NNs [Harris *et al*, 2002].

B-spline functions are local, compact, piecewise polynomials of a given order k, for which a simple recurrence relationship exists [Brown and Harris, 1994] [Zhang and Knoll, 1998]. B-spline functions have been widely used in surface fitting applications, but they also are suited to define fuzzy membership functions as is described next.

Assume x is a general input variable that is defined on the universe of discourse $[x_1,x_m]$. Given a sequence of ordered parameters, known as a *knot vector*: $[x_1,x_2,\ldots,x_m]^T$, the i-th normalised B-spline basis function $N_{i,k}$ of order k is defined as:

$$N_{i,k}(x) = \begin{cases} \begin{cases} 1 \quad \text{for} \quad x_i \le x < x_{i+1} \\ 0 \quad \text{otherwise} \end{cases} & \text{if} \quad k = 1 \\ \dfrac{x - x_i}{x_{i+k-1} - x_i} N_{i,k-1}(x) + \dfrac{x_{i+k} - x}{x_{i+k} - x_{i+1}} N_{i+1,k-1}(x) & \text{if} \quad k > 1 \end{cases} \tag{3.28}$$

with $i = 1,\ldots,m - k$. Figure 3.18 shows B-spline basis functions of order $k = 1, 2, 3, 4$, with knot vector $= [0,1,2,3,4,5,6]^T$.

Each one of the univariate B-spline basis function $N_{i,k}$ can be used to define a membership function for a corresponding fuzzy set. The selection of the order k of the B-spline basis function determines these characteristics: *degree*, *shape*, *width*, and *overlap* of the resulting fuzzy sets. For example, Table 3.2 shows these characteristics for k up to four. The width of a fuzzy set is measured by the number of knot intervals and the overlap degree by how many fuzzy sets are defined on each knot interval (see figure 3.18). Thus, by specifying only the order k and a knot vector, a set of membership functions can be implemented using the recursive relation (3.28), whose shape is determined by the order k, and where each membership function has compact support k units wide. Additionally, the membership functions can be defined to

form a partition of unity $\sum_{i=1}^{m} N_{i,k}(x) = 1$ in the universe of discourse of the corresponding linguistic variable.

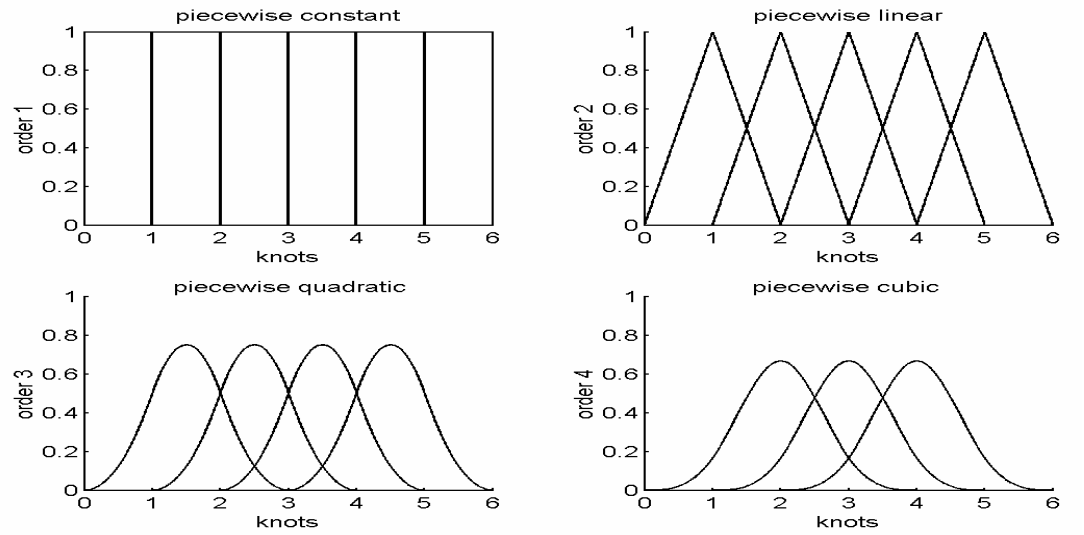


Figure 3.18 B-spline basis functions of order k = 1, 2, 3, 4, with knot vector = $[0,1,2,3,4,5,6]^T$.

Table 3.2 Fuzzy sets characteristics defined by the order of the B-spline basis function

Order k	1	2	3	4
Degree	0	1	2	3
Shape	Rectangular	Triangular	Quadratic	Cubic
Width	1	2	3	4
Overlap	1	2	3	4

In this context, by using B-splines to implement fuzzy sets an interesting class of neuro-fuzzy systems can be implemented. This class of neuro-fuzzy systems, considering a MISO system, is implemented by satisfying these conditions:

a) The real-valued inputs to the neuro-fuzzy system are represented via fuzzy singletons (a singleton fuzzification method is used),
b) B-splines are used to implement fuzzy sets in the antecedent part of fuzzy rules,
c) Fuzzy singletons are used to define the consequent part of fuzzy rules,
d) Algebraic operators are used to implement fuzzy logic functions (*product* for intersection *sum* for union, and *sum-prod* compositional rule of inference),
e) The COA defuzzification method is used.

If all the above conditions are satisfied, then the output of this class of neuro-fuzzy systems is given by:

$$y(\mathbf{x}) = \sum_{j=1}^{p} \prod_{i=1}^{n} N_i^{k_i}(x_i) a_j = \sum_{j=1}^{p} \psi_j(\mathbf{x}) a_j \tag{3.29}$$

where $N_i^{k_i}(x_i)$ are univariate B-spline basis functions (the order of the B-splines are omitted for brevity), which define the linguistic values (fuzzy sets) of input variables $x_1,\ldots x_i,\ldots,x_n$; a_j is the singleton consequent of rule j, which in this case is considered as a network weight; j = 1,…,p, p = number of rules in the fuzzy rule base; and $\psi_j(\mathbf{x})$ is the j-th multivariate B-spline basis function, $\mathbf{x} = [x_1,\ldots,x_n]^T$.

A graphical representation of a neuro-fuzzy system of this class is shown in figure 3.19. This class of neuro-fuzzy system has several important characteristics [Harris *et al*, 1999]: it (i) produces smoother interpolation, (ii) provides an equivalence between NNs and FISs, and (iii) enables FISs to be readily analysed. Additionally, this class of neuro-fuzzy system satisfies the Stone-Weirstrass theorem [Harris *et al*, 2002] and so they can approximate any continuous nonlinear function $f(\mathbf{x})$ defined on a compact domain with arbitrary accuracy.

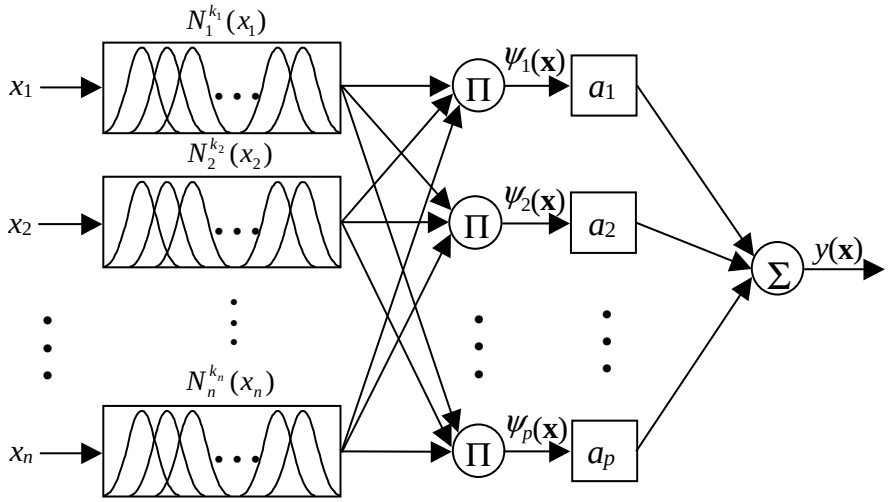


Figure 3.19 Structure of a neuro-fuzzy system using B-splines.

Equation (3.29) means that the output of the described neuro-fuzzy system is simply a weighted sum of multivariate B-spline basis functions, $\psi_j(\mathbf{x})$, for which the weights a_j can be trained by a linear optimisation algorithm. The *j*-th multivariate B-spline basis function $\psi_j(\mathbf{x})$, is generated by multiplying *n* univariate basis functions $N_i^{k_i}(x_i)$:

$$\psi_j(\mathbf{x}) = N_1^{k_1}(x_1) N_2^{k_2}(x_2) \cdots N_n^{k_n}(x_n) = \prod_{i=1}^{n} N_i^{k_i}(x_i) \qquad (3.30).$$

An example of a two dimensional multivariate B-spline basis function formed by two quadratic univariate basis functions is shown in figure 3.20.

The equivalence between the class of neuro-fuzzy systems described by (3.29) and FISs is demonstrated next. Consider a FIS with B-spline basis functions used to implement fuzzy sets in the antecedent of the fuzzy rules, it means that the fuzzy rule base is of the form:

$$R_j: \text{If } x_1 \text{ is } N_1^{k_1}(x_1) \text{ and } x_2 \text{ is } N_2^{k_2}(x_2) \text{ and } \cdots \text{ and } x_n \text{ is } N_n^{k_n}(x_n) \text{ then } y \text{ is } a_j \qquad (3.31)$$

where the above fuzzy rules have been numbered by $j = 1,2,\ldots,p$. Each j corresponds to an ordered sequence $k_1,\ldots,k_i,\ldots,k_n$; where $k_i = 1,2,\ldots,m_i$; and m_i is the number of fuzzy sets, in this case univariate B-spline basis functions $N_i^{k_i}(x_i)$, defined as linguistic values of variable x_i, $i = 1,..,n$, n is the number of input variables to the FIS.

If fuzzy singletons are used to represent the real-valued inputs and the *product* operator is used for fuzzy intersection, then the firing strength[1] of the *j*-th rule, $\mu_j(\mathbf{x})$, is given by [Brown and Harris, 1994]:

$$\mu_j(\mathbf{x}) = N_1^{k_1}(x_1)N_2^{k_2}(x_2)\cdots N_n^{k_n}(x_n) = \prod_{i=1}^{n} N_i^{k_i}(x_i) \tag{3.32}.$$

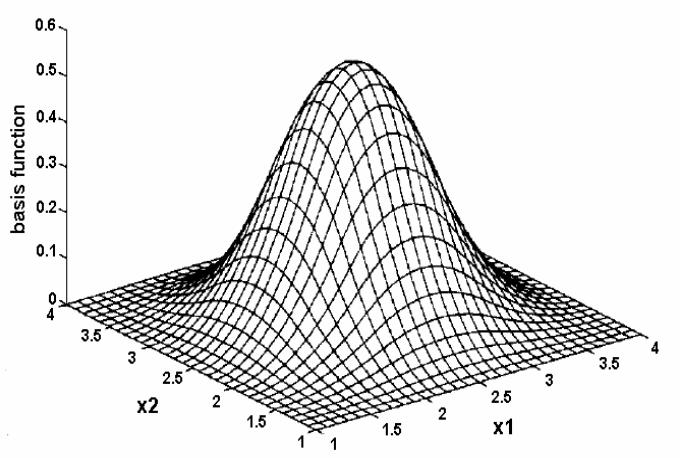


Figure 3.20 A two-dimensional quadratic multivariable B-spline basis function.

Next, using the COA defuzzification method the real output from the rule set (3.31) is given by:

$$y(\mathbf{x}) = \frac{\sum_{j=1}^{p} \mu_j(\mathbf{x})\, a_j}{\sum_{j=1}^{p} \mu_j(\mathbf{x})} = \frac{\sum_{j=1}^{p}\prod_{i=1}^{n} N_i^{k_i}(x_i)a_j}{\sum_{j=1}^{p}\prod_{i=1}^{n} N_i^{k_i}(x_i)} = \frac{\sum_{j=1}^{p}\prod_{i=1}^{n} N_i^{k_i}(x_i)a_j}{\prod_{i=1}^{n}\sum_{k_i=1}^{m_i} N_i^{k_i}(x_i)} \tag{3.33}.$$

Finally, since B-splines have a partition of unity, $\sum_{k_i=1}^{m_i} N_i^{k_i}(x_i) = 1$, (3.33) reduces to (3.29):

$$y(\mathbf{x}) = \sum_{j=1}^{p}\prod_{i=1}^{n} N_i^{k_i}(x_i)a_j = \sum_{j=1}^{p} \psi_j(\mathbf{x})a_j .$$

Thus, the neuro-fuzzy system shown in figure 3.19 can be seen either as a B-spline neural network or as a FIS with membership functions implemented using B-spline basis functions [Wu and Harris, 1997]. The training of the neuro-fuzzy system, where the free parameters are the weights $a_1,\ldots,a_p$, can be carried out using many of the traditional learning algorithms used in feed-forward neural networks. However, in the rest of this section, only the least mean squares (LMS) and normalised least mean squares (NLMS) training algorithms are described.

[1] Note that instead of using the symbol *w* to denote the firing strength, as in section 3.2.2, here the symbol $\mu_j(\mathbf{x})$, is used. This new notation reflects its dependence on the input vector **x** and will be used onwards.

First, note that if a vector of weights $\boldsymbol{a} = [a_1,\ldots,a_p]^T$ and a vector of multivariate B-spline basis functions $\psi(\mathbf{x}(t))= [\psi_1(\mathbf{x}(t)),\ldots,\psi_p(\mathbf{x}(t))]^T$ are defined, then (3.29) can be rewritten as:

$$y(\mathbf{x}(t)) = \sum_{j=1}^{p} \psi_j(\mathbf{x}(t))a_j = \psi^T(\mathbf{x}(t))\boldsymbol{a} \tag{3.34}$$

where the argument t has been included to denote the time step of an iterative process involved in adjusting the network weights. Now, if there exist a set of training pairs $\{\mathbf{x}(t)\ y_d(\mathbf{x}(t))\}$, where $y_d(\mathbf{x}(t))$ is the desired output for a given input $\mathbf{x}(t) = [x_1(t),\ldots,x_n(t)]^T$ at time step t, then the neuro-fuzzy system described by (3.34) can be trained in the weight vector $\boldsymbol{a}$ using the LMS or NLMS algorithms. Both LMS and NLMS are instantaneous training algorithms, this means that the weights are adapted on-line. These algorithms adjust the weights by using information provided by only a small subset of training pairs and by making an estimate of the mean squared error (MSE) at time step t.

The *error signal* $\varepsilon(t)$ measured at time t is defined as the difference between the desired and the actual neuro-fuzzy system output values:

$$\varepsilon(t) = y_d(\mathbf{x}(t)) - y(\mathbf{x}(t)) \tag{3.35}.$$

The objective of the LMS learning algorithm is to apply a sequence of corrective adjustments to the network weights in order to make the output signal come closer to the desired signal in a step-by-step manner [Haykin, 1999]. This objective is achieved by minimising a *cost function*, $J(\boldsymbol{a})$, defined in terms of the error signal as:

$$J(\mathrm{a}) = \frac{1}{2}\varepsilon^2(t) = \frac{1}{2}\left(y_d(\mathbf{x}(t)) - y(\mathbf{x}(t))\right)^2 \tag{3.36}.$$

The LMS algorithm is based on the use of instantaneous values for the cost function. Therefore, differentiating (3.36) with respect to the weight vector $\boldsymbol{a}$ yields:

$$\frac{\partial J(\boldsymbol{a})}{\partial \boldsymbol{a}} = \varepsilon(t)\frac{\partial \varepsilon(t)}{\partial \boldsymbol{a}} \tag{3.37}.$$

Let $\boldsymbol{a}(t)$ denote the value of the weight vector $[a_1(t),\ldots,a_p(t)]$ of the neuro-fuzzy system excited by the signal input vector $\mathbf{x}(t)$ at time step t. Thus, (3.35) can be rewritten as:

$$\varepsilon(t) = y_d(\mathbf{x}(t)) - \psi^T(\mathbf{x}(t))\boldsymbol{a}(t) \tag{3.38}.$$

Hence,

$$\frac{\partial \varepsilon(t)}{\partial \boldsymbol{a}(t)} = -\psi(\mathbf{x}(t)) \tag{3.39},$$

and from (3.39) in (3.37) gives,

$$\frac{\partial J(\boldsymbol{a})}{\partial \boldsymbol{a}(t)} = -\varepsilon(t)\psi(\mathbf{x}(t)) \tag{3.40},$$

this result is used as an estimate of the gradient vector of the cost function at time t:

$$\hat{\nabla} J(\boldsymbol{a}) = -\varepsilon(t)\psi(\mathbf{x}(t)) \tag{3.41}.$$

Finally, employing the method of steepest descent [Widrow and Stearns, 1985] successive adjustments are applied to the weight vector $\boldsymbol{a}$ in the direction opposite to the estimated gradient $\hat{\nabla} J(\boldsymbol{a})$, which results in the LMS learning algorithm formulated as:

$$\boldsymbol{a}(t+1) = \boldsymbol{a}(t) + \eta\varepsilon(t)\psi(\mathbf{x}(t)) \tag{3.42},$$

where η is the learning-rate parameter. The adjustment of the weight vector continues until the system reaches a steady state. At this point the learning process is terminated.

A possible drawback of using the LMS learning algorithm is that the reduction in the output error depends on the size of the transformed input vector. If the variance in magnitude is large, then small values of η are required for stable learning. This can greatly increase the time taken for training [Brown *et al*, 1996].

An alternative algorithm, where the dependency on the size of the transformed input is removed, is the NLMS learning algorithm. In this algorithm the weight vector is updated by

$$\boldsymbol{a}(t+1) = \boldsymbol{a}(t) + \frac{\eta\varepsilon(t)\psi(\mathbf{x}(t))}{\left\|\psi(\mathbf{x}(t))\right\|_2^2} = \boldsymbol{a}(t) + \frac{\eta\varepsilon(t)\psi(\mathbf{x}(t))}{\psi^T(\mathbf{x}(t))\psi(\mathbf{x}(t))} \tag{3.43},$$

where, if the learning-rate parameter η is in the range (0,2), then stable learning is assured [Brown *et al*, 1996].

A proof of convergence of both learning algorithms can be obtained, but these are not described here. The reader interested is referred to [Haykin, 1999] [Brown *et al*, 1996] [Widrow and Stearns, 1985].

In conclusion, as a result of using B-spline basis functions to define fuzzy membership functions in the described neuro-fuzzy system, several properties are gained:

- A simple and stable recursive relationship is used to evaluate the grade of membership for any input *x*.
- The basis functions have a compact support, which means that knowledge is stored locally across only a small number of basis functions.
- The basis functions form a partition of unity, which also implies that the corresponding fuzzy variables are complete.

Additionally, once the training has been completed, the resulting system is readily interpretable in form of if-then rules. This means that the black box aspect of a neural network is avoided, and it is also possible to obtain new knowledge from the system.

3.6 Summary

In this chapter FISs and NN have been described. Firstly, FISs form a consistent methodology to capture the way in which humans interpret the surrounding world. Thus, by using common-sense fuzzy rules a FIS is capable of characterise the variables of a system and its interrelations. Secondly, NNs are capable of learning from examples and store this knowledge in network weights distributed throughout the net.

From the aforementioned characteristics, both approaches can be synergistically combined in different ways: fuzzy neural networks, concurrent neural/fuzzy systems, cooperative neuro-fuzzy models and hybrid neuro-fuzzy systems. The most successful of these combinations are hybrid neuro-fuzzy systems.

A particular type of hybrid neuro-fuzzy system is that which makes use of B-spline basis functions to implement membership functions. These systems take advantage of the three basic properties that B-spline functions have: positivity, compact support, and partition of unity. As a result, once the training has finished, the neuro-fuzzy system is interpretable in form of if-then rules.

The concepts presented here will be used in proceeding chapters to develop novel hybrid MSDF architectures.

CHAPTER 4

KALMAN FILTERING AND MULTI-SENSOR DATA FUSION ARCHITECTURES

4.1 Introduction

The previous two chapters reviewed general MSDF techniques, and the neuro-fuzzy approaches respectively. Before proposing MSDF architectures considering hybrid traditional-non-traditional techniques, specifically combining the Kalman filter and neuro-fuzzy techniques, now it is necessary to describe the Kalman filtering approach and the MSDF architectures based on it.

The Kalman Filter algorithm was first published in 1960 by R. E. Kalman. In his famous paper Kalman described a recursive solution to the discrete data linear filtering problem [Kalman, 1960]. Since then, the Kalman filter has been the subject of extensive research and applications. With the advances in digital computing, Kalman filtering applications have diversified into many areas, but the majority of the applications are found in the areas of autonomous or assisted navigation, where, specifically, the Kalman filter has two main tasks: tracking and multisensor data fusion.

In this chapter, a broad description of the Kalman filter algorithm and the MSDF architectures based on it are given. A deeper description of the Kalman filter algorithm can be found in [Maybeck, 1979] [Billings, 1980] [Welch and Bishop, 1995] and [Brown and Hwang, 1997].

4.2 The Kalman filter algorithm

The Kalman filter is an optimal recursive data processing algorithm [Kalman, 1960] [Maybeck, 1979] [Brown and Hwang, 1997] that provides a linear, unbiased, and minimum error variance estimate of the unknown state vector $x_k \in \Re^n$ at each instant k = 1,2,…, (indexed by the subscripts) of a discrete-time process that is governed by the linear stochastic difference equation:

$$x_{k+1} = \Phi_k x_k + B_k u_k + w_k \tag{4.1}$$

with the discrete measurement vector $z_k \in \Re^m$ given by:

$$z_k = H_k x_k + v_k \tag{4.2}$$

where:

x_k = (n×1) state vector at time k.
Φ_k = (n×n) state transition matrix.
B_k = (n×l) matrix that relates the control input $u_k \in \Re^l$ to the state vector x_k.
u_k = (l×1) vector of the input forcing function.
w_k = (n×1) process noise vector.

z_k = (m×1) measurement vector at time k.
H_k = (m×n) measurement sensitivity matrix.
v_k = (n×1) vector of additive measurement noise.

Both w_k and v_k are assumed to be uncorrelated[1] zero-mean Gaussian white noise sequences with covariances:

$$E\{w_k\} = 0 \quad \text{for all } k, \tag{4.3}$$

$$E\{v_k\} = 0 \quad \text{for all } k, \tag{4.4}$$

$$E\{w_k w_i^T\} = \begin{cases} Q_k, & i = k \\ 0 & i \neq k \end{cases} \tag{4.5}$$

$$E\{v_k v_i^T\} = \begin{cases} R_k, & i = k \\ 0 & i \neq k \end{cases} \tag{4.6}$$

$$E\{w_k v_i^T\} = 0 \qquad \text{for all } k \text{ and } i \tag{4.7}$$

where $E\{\cdot\}$ is the statistical expectation operator, Q_k is the process noise covariance matrix, and R_k is the measurement noise covariance matrix. If w_k in (4.1) is replaced by $\Gamma_k w_k$, where Γ_k is a process noise distribution matrix, then Q_k becomes $\Gamma_k Q_k \Gamma_k^T$.

The objective of the Kalman filter is to estimate the value of the state vector x_{k+1} given all the information available up to the current instant of time, i. e. $z_k,\ldots,z_0$ and $u_k,\ldots,u_0$. In accordance with Welch and Bishop [1995], this objective is reached in the Kalman filter algorithm by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. In this sense, the equations for the Kalman filter can be arranged into two groups: time update equations and measurement update equations. The time update equations project forward, ahead in time, the current state and error covariance estimates to obtain the *a priori* estimates for the next step. The measurement update equations incorporate a new measurement into the a priori estimate to adjust the projected estimate and obtain an improved *a posteriori* estimate (the feedback part of the filter). From another point of view, the time update equations can also be considered as predictor equations, while the measurement equations can be considered as corrector equations.

Therefore, the specific Kalman filter equations are organized into two groups,

i) Time update (or prediction) equations:

$$\hat{x}_{k+1(-)} = \Phi_k \hat{x}_{k(+)} + B_k u_k \tag{4.8}$$

$$P_{k+1(-)} = \Phi_k P_{k(+)} \Phi_k^T + Q_k \tag{4.9}.$$

These equations project, from time step k to step k+1, the current state and error covariance estimates to obtain the *a priori* estimates, denoted by (–), for the next time step.

ii) Measurement update (or correction) equations:

$$K_k = P_{k(-)} H_k^T [H_k P_{k(-)} H_k^T + R_k]^{-1} \tag{4.10}$$

$$\hat{x}_{k(+)} = \hat{x}_{k(-)} + K_k [z_k - H_k \hat{x}_{k(-)}] \tag{4.11}$$

$$P_{k(+)} = [I - K_k H_k] P_{k(-)} \tag{4.12}.$$

[1] The Kalman filter algorithm can be extended to accommodate the cases when correlation exists between the two noise sequences or when correlated measurement noise is present but they are not considered here, the reader interested in those cases is referred to [Grewal and Andrews, 1993].

These equations incorporate a new measurement into the *a priori* estimates to obtain the improved *a posteriori* estimates, denoted by (+). A graphical representation of the Kalman filter algorithm is shown in figure 4.1.

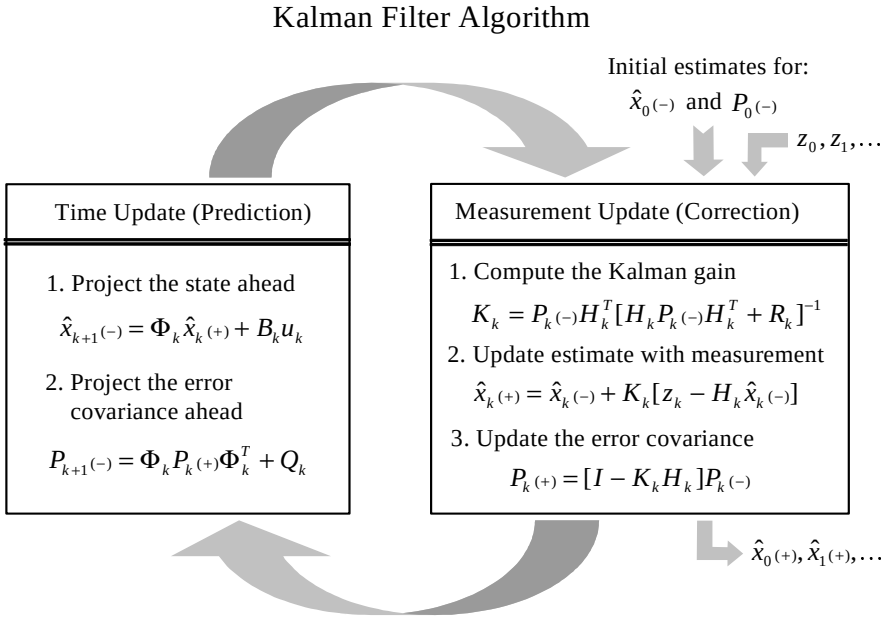


Figure 4.1 Graphical representation of the Kalman filter algorithm.

In the above equations, $\hat{x}_{k(+)}$ is an estimate of the system state vector x_k, and $P_{k(+)}$ is the covariance matrix corresponding to the state estimation error defined by

$$P_{k(+)} = E\left\{(x_k - \hat{x}_{k(+)})(x_k - \hat{x}_{k(+)})^T\right\} \tag{4.13}$$

In equation (4.11) the term $H_k \hat{x}_{k(-)}$ is referred to as the one-stage predicted measurement, which is the best prediction of what the measurement at time k will be before it is actually taken. The difference between the actual measurement z_k and its one-stage prediction is called the measurement residual r_k [Maybeck, 1979] defined as:

$$r_k = z_k - H_k \hat{x}_{k(-)} \tag{4.14}$$

Here it is necessary to mention that in the literature the term $(z_k - H_k \hat{x}_{k(-)})$ is frequently referred to as the innovation sequence and the name "residual" is reserved for the quantity $(z_k - H_k \hat{x}_{k(+)})$, which does not appear explicitly in the algorithm. Therefore, in this work the name residual will be retained.

The weighted residual, $K_k[z_k - H\hat{x}_{k(-)}]$, acts as a correction to the predicted estimate $\hat{x}_{k(-)}$ to form the estimation $\hat{x}_{k(+)}$; the weighting matrix K_k is referred to as the filter gain or the Kalman gain matrix.

In the algorithm the matrices Φ_k, B_k and H_k are assumed to be known. Q_k and R_k are nonnegative definite matrices whose values are also assumed to be known. The Kalman filter algorithm starts with initial conditions at $k = 0$ being: $\hat{x}_{0(-)}$, and $P_{0(-)}$. With the progression of time, as new measurements z_k become available, the cycle prediction-correction of states and the corresponding error covariances can follow recursively ad infinitum.

4.2.1 Alternative form of the Kalman filter algorithm

The equations of the Kalman filter algorithm, as described in last section, can be algebraically manipulated into a variety of forms [Brown and Hwang, 1997]. One of the most useful alternative forms is that which expresses the algorithm in terms of the inverse of the error covariance matrix, instead of the covariance itself. This is:

$$P_k^{-1}(+) = P_k^{-1}(-) + H_k^T R_k^{-1} H_k \tag{4.15},$$

which is known as the information matrix [Maybeck, 1979]. Similarly, an alternative expression for the Kalman filter gain is:

$$K_k = P_k(+) H_k^T R_k^{-1} \tag{4.16}.$$

The demonstration of the equivalence between the above equations and their counterparts can be found in [Brown and Hwang, 1997] and [Maybeck, 1979]. Note that the expression for the Kalman gain now involves $P_k(+)$, therefore, K_k must be computed after $P_k(+)$. This means that the order in which $P_k(+)$ and K_k are computed in the recursive algorithm is reversed from that given in the last section.

Finally, by using (4.16) and (4.12) the estimate update equation (4.11) at time k can be written in a different form as:

$$\begin{aligned} \hat{x}_k(+) &= [I - K_k H_k]\hat{x}_k(-) + K_k z_k \\ &= P_k(+) P_k^{-1}(-) \hat{x}_k(-) + K_k z_k \end{aligned} \tag{4.17},$$

hence, the complete alternative Kalman filter algorithm is summarised graphically in figure 4.2.

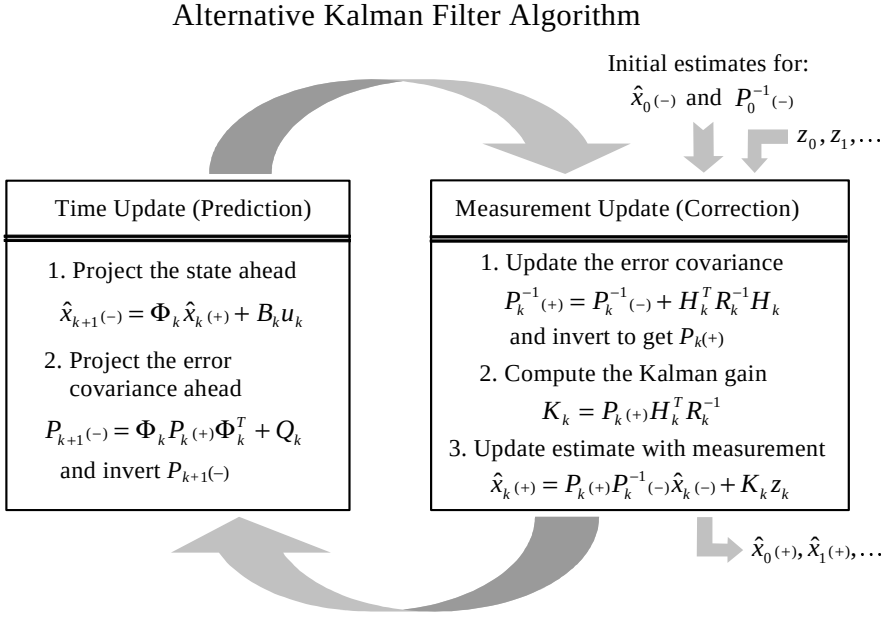


Figure 4.2 Graphical representation of the alternative Kalman filter algorithm.

In terms of information, equation (4.15) means that the updated information is equal to the prior information plus the additional information obtained from the measurement at time k. In

addition, if R_k is block diagonal, the total added information can be divided into separated components, each one representing the contribution from the respective measurement block.

The presented alternative Kalman filter algorithm possesses some special characteristics, such as allowing a start-up procedure for the case of $P_0^{-1}{}_{(-)}$ being singular. In other words, this means that the algorithm can be started with infinite uncertainty if the physical situation under consideration so dictates.

Note from figure 4.2 that two $n \times n$ matrix inversions are required for each recursive loop. This can lead to computational problems if the order of the state vector is large. Nevertheless, the alternative Kalman filter algorithm has several useful applications; one of these is in the decentralised Kalman filter as will be seen later in this chapter.

4.2.2 Consistency of the Kalman filter algorithm

When a constant parameter is estimated, the consistency of an estimator is defined as the convergence of the estimate to the actual value [Bar-Shalom and Li, 1993]. Convergence means that there is an asymptotic reduction to zero of the difference between the estimated and the actual value. In other words, this implies that the uncertainty about the true value reduces to zero asymptotically with time.

In the case of the Kalman filter, which estimates the state of a system, there are two results, the current estimate of the state, $\hat{x}_{k(+)}$, and its associated error covariance matrix $P_{k(+)}$. Thus, the definition of consistency has to be formulated in a different way.

Consider first the definition of the phenomenon known as divergence:

"If after an extended period of operation of the filter, the errors in the estimates eventually diverge to values entirely out of proportion to the rms values predicted by the equations of the filtering procedure, then the filter has diverged" [Fitzgerald, 1971].

Thus, in accordance with the above definition, the evaluation of the consistency of the Kalman filter estimates can be carried out based on its statistical characteristics. If all the requirements for optimality (given in section 4.2) are satisfied, then the first and second order moments of the state are:

$$E\{x_k - \hat{x}_{k(+)}\} = 0 \tag{4.18}$$

$$E\{[x_k - \hat{x}_{k(+)}][x_k - \hat{x}_{k(+)}]^T\} = P_{k(+)} \tag{4.19},$$

which are the conditions that the filter should satisfy in order to be consistent. Condition (4.18) is known as the *unbiasedness* requirement for the estimates (i.e. zero mean estimation error), and condition (4.19) is known as the *covariance-matching* requirement (i.e. the actual mean squared error matches the filter-calculated covariance).

It is noteworthy to say that consistency and optimality are closely related in the Kalman filter setting. Since the filter gain is based on the filter-calculated error covariances, it follows that consistency is necessary for optimality. This is why a test of consistency is essential in order to verify the optimal operation of the filter.

Therefore, in order to test the consistency of the Kalman filter, three criteria need to be satisfied:

a) The state errors should have zero mean and have magnitude commensurate with the state covariance as yielded by the filter.
b) The filter residuals should also have the above property.
c) The filter residuals should approximate a white noise process.

The last two criteria are the only ones that can be tested in real data applications. The first criterion can be tested only in simulation. For this reason, here only the real time tests for the last two criteria are described. The reader interested in the test for the first criterion is referred to [Bar-Shalom and Xiao-Rong, 1993].

The criterion b) can be tested using the *normalised residual squared*, ε_{rk}, (as well known as the *normalised innovation squared*) defined by:

$$\varepsilon_{rk} = r_k^T S_k^{-1} r_k \tag{4.20},$$

where r_k is the measurement residual, defined by equation (4.14), and S_k its predicted residual covariance calculated by the Kalman filter algorithm, see equation 4.10, as:

$$S_k = H_k P_{k(-)} H_k^T + R_k \tag{4.21}.$$

Then, the time-averaged normalised residual squared (TANRS) is defined as:

$$\bar{\varepsilon}_r = \frac{1}{n} \sum_{k=1}^{n} r_k^T S_k^{-1} r_k \tag{4.22}.$$

Thus, if the residuals are white, zero mean, and consistent with its calculated covariance S_k, then $n\bar{\varepsilon}_r$ has a chi-square distribution with nm degrees of freedom. For a large enough n, the previous statements mean that $\bar{\varepsilon}_r$ has to be equal to the dimension of the corresponding vector since it is chi-squared distributed [Bar-Shalom and Xiao-Rong, 1993], this is $\bar{\varepsilon}_r = m$, where m is the dimension of the measurement vector $\mathbf{z_k}$. This test is well known as the universally most powerful invariant test statistic (UMPITS), and is commonly used for testing the validity of process models [Stansfield, 2001].

Criterion c) can be tested using the *time-average autocorrelation* defined by:

$$\bar{\rho}(l) = \sum_{k=1}^{n} r_k^T r_{k+l} \left[\sum_{k=1}^{n} r_k^T r_k \sum_{k=1}^{n} r_{k+l}^T r_{k+l} \right]^{-\frac{1}{2}} \tag{4.23}$$

which tests the whiteness of the residuals l steps apart in a single sequence. The statistic (4.23) is, for a large enough n, in view of the central limit theorem, normally distributed. Thus, if the residuals are zero mean and white, then the mean of (4.23) is near zero and its variance is $1/n$.

The described tests are based on replacing the ensemble averages by time averages based on the ergodicity of the residual sequence.

4.3 Multi-sensor data fusion architectures based on the Kalman filter

Kalman filtering has been used in the processing of data coming from multiple sensors. These sensors are assumed to be different hardware devices, and each one with its own data stream. If

all the measurements are processed by a single filter, then it is referred to as a *centralised Kalman filter*. If the measurements coming from each sensor are processed first by a local filter and then afterwards a master filter does the fusion process, then two alternatives of processing exist, a *decentralised Kalman filter* or a *federated Kalman filter*. In the next sections, each one of these MSDF architectures based on the Kalman filtering technique are described.

4.3.1 Centralised Kalman filter

In the centralised Kalman filter (from here referred to as CKF) the measurements coming from all sensors are processed by a single filter [Gao et al, 1993], as is represented graphically in figure 4.3. The CKF yields optimal estimates in the sense of minimum mean-squared error (MMSE), subject to the usual assumptions of linear dynamics and noise statistics.

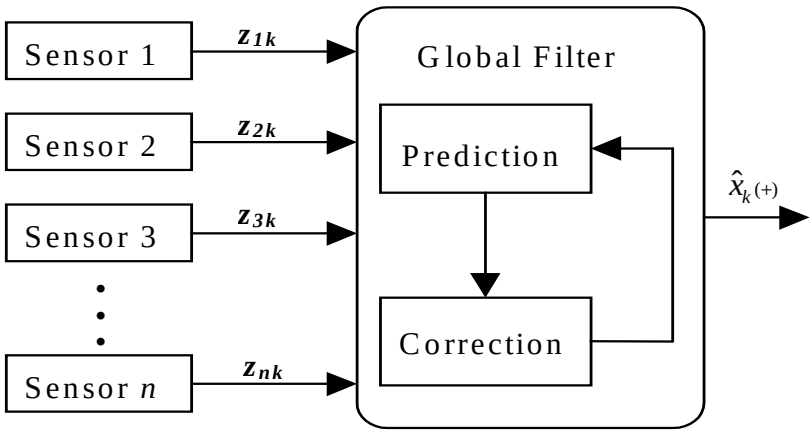


Figure 4.3 Centralised Kalman filtering architecture.

Assuming that the measurements coming in to the CKF are from different independent sensors, the global measurement vector z_k can be written as,

$$z_k = [z_{1k}^T \dots z_{nk}^T]^T \tag{4.24}$$

and the global measurement sensitivity matrix can be written as,

$$H_k = [H_{1k}^T \dots H_{nk}^T]^T \tag{4.25}$$

Thus, the corresponding measurement equations for the measurements z_{ik}, linearly related to the components of the state vector x_k, are,

$$z_{ik} = H_{ik} x_k + v_{ik} \tag{4.26}$$

where the subscript i denotes the i-th sensor and z_{ik} is the measurement vector coming from the i-th sensor at time k; and v_{ik} is the corresponding measurement noise vector with covariance matrix R_{ik}.

If it is assumed that the v_{ik} vectors are uncorrelated across all sensors, then the global measurement noise covariance matrix R_k has a block diagonal structure,

$$R_k = block\ diag[R_{1k} \dots R_{nk}] \tag{4.27}$$

Therefore, by using the global measurement vector, the global measurement sensitivity matrix, and the global measurement noise covariance matrix, given by equations (4.24), (4.25) and (4.27), respectively, the standard Kalman filter can be directly applied.

To achieve optimality, the CKF will result in a high computational load when implemented. Also there is the issue of the lack of robustness when there is spurious data in any of the sensors. For this reason several decentralised Kalman filtering approaches have been devised as is described in next sections.

4.3.2 Decentralised Kalman filter

The decentralised Kalman filter (from here referred to as DKF) is a two-stage data processing technique [Brown and Hwang, 1997] [Wei and Schwarz, 1990] [Hashemipour *et al*, 1988]. In the DKF the standard Kalman filter is divided into one or more sensor-dedicated local filters (1 to *n*) and a master filter, as is graphically represented in figure 4.4. In the first stage, the local filters process their own data in parallel to yield the best possible local estimates. In the second stage, the master filter fuses the local estimates, yielding the best global estimate. As a result, the computational load can be significantly reduced by this technique. The results of each local Kalman filter may be locally suboptimal, but when combined they are globally optimal [Brown and Hwang, 1997].

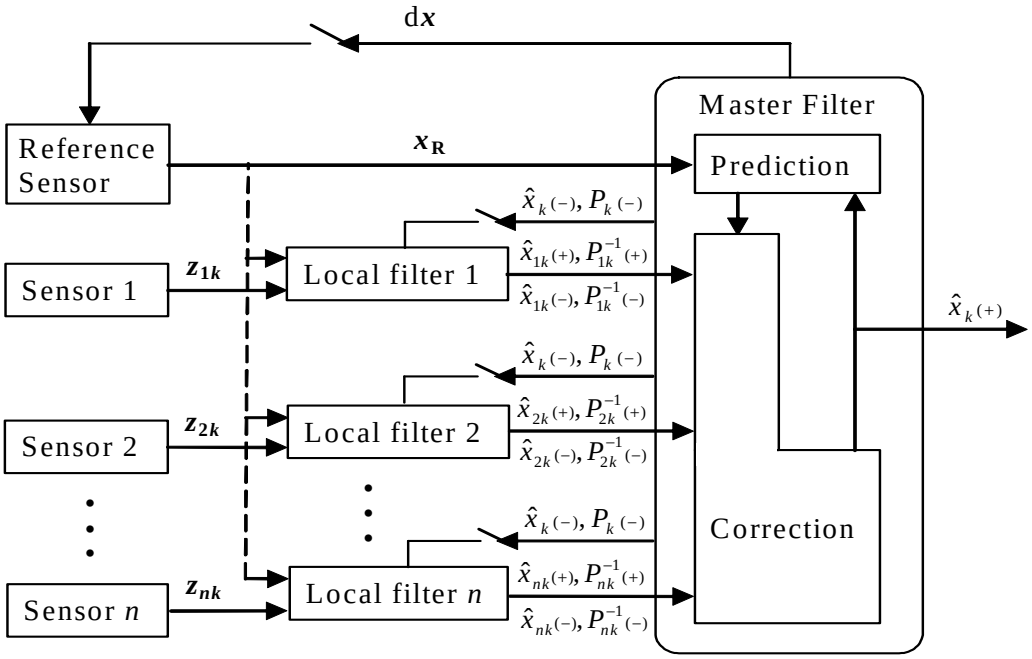


Figure 4.4 Decentralised Kalman filtering architecture.

The set of equations of the DKF are described next. First, the *i*-th updated local system can be depicted by a state space model of the form:

$$x_{i(k+1)} = \Phi_{ik} x_{ik} + B_{ik} u_{ik} + w_{ik} \tag{4.28}$$

with corresponding measurement equation:

$$z_{ik} = H_{ik} x_{ik} + v_{ik} \tag{4.29}$$

where x_{ik} is the state vector of the *i*-th local system, w_{ik} is the local system noise with covariance Q_{ik}, z_{ik} is the corresponding local measurement vector, and v_{ik} is the measurement noise with covariance R_{ik}.

With the above definitions, each *i*-th local filter can compute its estimates based strictly on its own observations and using the standard Kalman filter algorithm. However, for convenience, the alternative form of the Kalman filter algorithm is used instead. This is:

Prediction equations:

$$\hat{x}_{i(k+1)(-)} = \Phi_{ik}\hat{x}_{ik(+)} + B_{ik}u_{ik} \tag{4.30}$$

$$P_{i(k+1)(-)} = \Phi_{ik}P_{ik(+)}\Phi_{ik}^{T} + Q_{ik} \tag{4.31}$$

Correction equations:

$$P_{ik}^{-1}{}_{(+)} = P_{ik}^{-1}{}_{(-)} + H_{ik}^{T}R_{ik}^{-1}H_{ik} \tag{4.32}$$

$$K_{ik} = P_{ik(+)}H_{ik}^{T}R_{ik}^{-1} \tag{4.33}$$

$$\hat{x}_{ik(+)} = P_{ik(+)}P_{ik}^{-1}{}_{(-)}\hat{x}_{ik(-)} + K_{ik}z_{ik} \tag{4.34}.$$

Now, in order to express the global estimate $\hat{x}_{k(+)}$ in terms of the local estimates, the same assumptions made in the CKF case, expressed by equations (4.24), (4.25) and (4.27), are made here. Therefore, the global optimal estimate and associated error covariance are then calculated by:

$$\hat{x}_{k(+)} = P_{k(+)}\left[P_{k}^{-1}{}_{(-)}\hat{x}_{k(-)} + \sum_{i=1}^{n} P_{ik}^{-1}{}_{(+)}\hat{x}_{ik(+)} - \sum_{i=1}^{n} P_{ik}^{-1}{}_{(-)}\hat{x}_{ik(-)}\right] \tag{4.35}$$

$$P_{k}^{-1}{}_{(+)} = P_{k}^{-1}{}_{(-)} + \sum_{i=1}^{n} P_{ik}^{-1}{}_{(+)} - \sum_{i=1}^{n} P_{ik}^{-1}{}_{(-)} \tag{4.36}.$$

The convenience of using the alternative Kalman filter algorithm can be seen now. The local filters can pass directly their respective $\hat{x}_{ik(+)}$, $P_{ik}^{-1}{}_{(+)}$, $\hat{x}_{ik(-)}$, and $P_{ik}^{-1}{}_{(-)}$ on to the master filter, which, in turn, can then compute its global estimate. Equations (4.35) and (4.36) formulate the correction equations for the best global estimate of the state vector x_k and its covariance matrix in terms of the local estimates and their covariances [Wei and Schwarz, 1990]. The global prediction of the state vector $\hat{x}_{k(-)}$ with its respective covariance matrix $P_{k(-)}$ is computed from the prediction equations (4.8) and (4.9) given by the standard Kalman filter algorithm.

4.3.2.a Decentralised Kalman filtering with feedback

In figure 4.4 it is suggested that a feedback from the master filter to the local filters can be considered (when the switches are closed). In fact, a DKF with feedback can be formulated as is explained next. In order to allow indirect measurement sharing, the predicted state vector $\hat{x}_{k(-)}$ and its respective covariance $P_{k(-)}$ can be fed back to the local filters. This feedback enables the local filters to reset their respective prior estimates more accurately with each step than they would be able to do otherwise [Brown and Hwang, 1997]. This feedback is achieved by letting:

$$\hat{x}_{ik}(-) = \hat{x}_k(-) \tag{4.37}$$

$$P_{ik}(-) = P_k(-) \tag{4.38}.$$

With these modifications, now the local correction equations are computed using the global predicted state vector and its respective covariance, this is:

$$\hat{x}_{ik}(+) = P_{ik}(+)P_k^{-1}(-)\hat{x}_k(-) + K_{ik}z_{ik} \tag{4.39}$$

$$P_{ik}^{-1}(+) = P_k^{-1}(-) + H_k^T R_k^{-1} H_k \tag{4.40},$$

and the global state estimation and respective covariance are calculated by:

$$\hat{x}_k(+) = \sum_{i=1}^{n} P_k(+)P_{ik}^{-1}(+)\hat{x}_{ik}(+) - P_k(+)P_k^{-1}(-)x_k(-) \tag{4.41}$$

$$P_k^{-1}(+) = \sum_{i=1}^{n} P_{ik}^{-1}(+) - P_k^{-1}(-) \tag{4.42}.$$

Note that there is no direct communication between the filters. However, there is indirect information sharing by feeding back $\hat{x}_k(-)$ with each step. This is due to the fact that $\hat{x}_k(-)$ is a linear combination of past measurements in all filters.

Observe too that the master filter maintains full optimality with the feedback architecture. Moreover, the local filters improve their optimality with respect to the DKF without feedback, but they still do not have full optimality with respect to all measurements [Brown and Hwang, 1997].

4.3.3 Federated Kalman filter

A special case of DKF is the federated Kalman filter [Carlson, 1990] [Gao *et al*, 1993] (from here referred to as FKF) which employs the principle known as “information sharing” among the local Kalman filters to improve the fault-tolerant performance of the system. A representation of this scheme is shown in figure 4.5. The possible information to be shared includes the kinematic process noise, the initial condition information, and common measurement information. Usually the kinematic process noise is the information selected to be shared [Gao *et al*, 1993].

In design, the FKF is similar to the DKF. Here too a two-stage data processing algorithm is used. The difference between the two approaches is the information sharing process of the former. In consequence, the same conditions for the DKF are considered for the FKF. Then in the description of the FKF the same *n* local subsystems defined by equation (4.28) with respective measurement systems given by equation (4.29) are considered here.

Now, let the fused (full centralized) solution be represented by the covariance matrix $P_{fk}(+)$ and the state vector $\hat{x}_{fk}(+)$, the *i*-th local filter solution by $P_{ik}(+)$ and $\hat{x}_{ik}(+)$, and the master filter solution by $P_{mk}(+)$ and $\hat{x}_{mk}(+)$. Thus, if the local and master filter solutions are statistically independent, they can be optimally combined by [Carlson, 1990]:

$$P_{fk}^{-1}(+) = P_{1k}^{-1}(+) + \cdots + P_{nk}^{-1}(+) + P_{mk}^{-1}(+) \tag{4.43}$$

$$P_{fk}^{-1}(+)\hat{x}_{fk}(+) = P_{1k}^{-1}(+)\hat{x}_{1k}(+) + \cdots + P_{nk}^{-1}(+)\hat{x}_{nk}(+) + P_{mk}^{-1}(+)\hat{x}_{mk}(+) \quad (4.44).$$

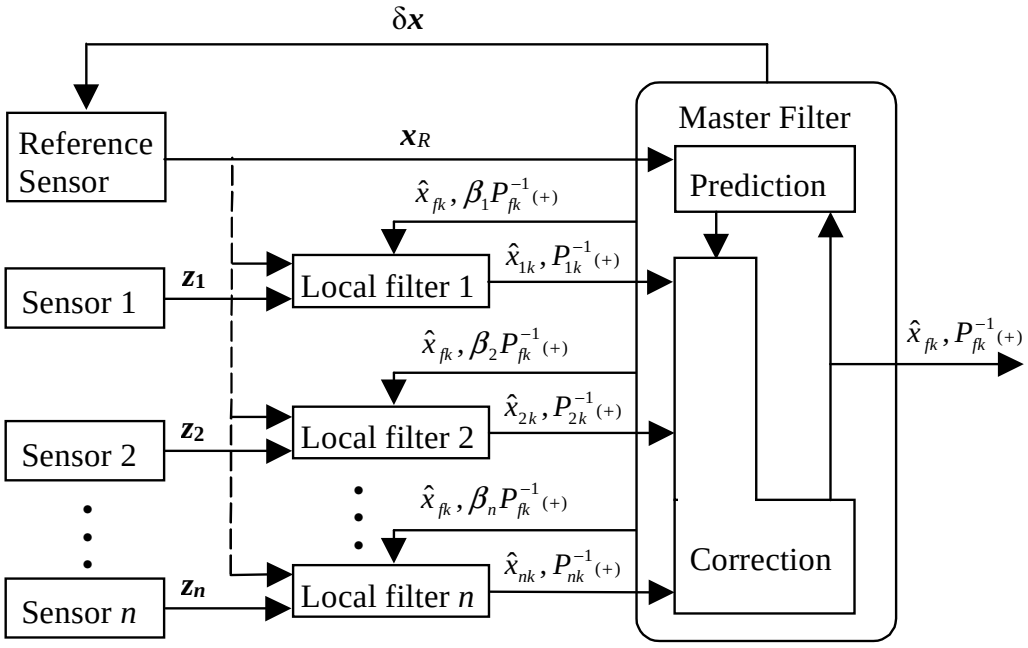


Figure 4.5 Federated Kalman filtering architecture.

The main idea of the FKF is to construct individual local and master solutions in such a way that they can be combined at any time by the above equations.

Therefore, the FKF algorithm follows the procedure now given:

1) Divide the fused (global filer) error covariance P_f, and the common process noise covariance Q_f in so way that the $i = 1,\ldots,n,m$ local filters and the master filter each receive fractions β_i, β_m of the total information, and set local state estimates to the fused (global) state estimate value $\hat{x}_k(+)$:

$$Q_{ik}^{-1} = \beta_i Q_f^{-1} \quad (4.45)$$

$$P_{ik}^{-1}(+) = \beta_i P_{fk}^{-1}(+) \quad (4.46)$$

$$\hat{x}_{ik}(+) = \hat{x}_{fk}(+) \quad (4.47)$$

where β_i, ($i = 1,\ldots,n,m$) are information-sharing factors. The conservation of information principle dictates that the information-sharing factors β_i sum to unity:

$$\sum_{i=1}^{n,m} \beta_i = 1 \quad (4.48).$$

2) Local filters and the master filter process their prediction equations independently:

$$\hat{x}_{i(k+1)}(-) = \Phi_{ik}\hat{x}_{ik}(+) + B_{ik}u_{ik} \quad (4.49)$$

$$P_{i(k+1)}(-) = \Phi_{ik}P_{ik}(+)\Phi_{ik}^T + Q_{ik} \quad (4.50)$$

with $i = 1,\ldots,n,m$.

3) Each local filter processes its own sensor measurement using, for convenience, the alternative Kalman filter algorithm:

$$P_{ik}^{-1}(+) = P_{ik}^{-1}(-) + H_{ik}^{T} R_{ik}^{-1} H_{ik} \tag{4.51}$$

$$K_{ik} = P_{ik}(+) H_{ik}^{T} R_{ik}^{-1} \tag{4.52}$$

$$\hat{x}_{ik}(+) = P_{ik}(+) P_{ik}^{-1}(-) \hat{x}_{ik}(-) + K_{ik} z_{ik} \tag{4.53}.$$

where $i = 1,2,\ldots,n$.

4) The fusion algorithm combines master filter and local filter estimates after each correction cycle by:

$$P_{fk}^{-1}(+) = \sum_{i=1}^{n} P_{ik}^{-1}(+) + P_{mk}^{-1}(-) \tag{4.54}$$

$$\hat{x}_{fk}(+) = P_{fk}(+) \left[P_{mk}^{-1}(-) \hat{x}_{mk}(-) + \sum_{i=1}^{n} P_{ik}^{-1}(+) \hat{x}_{ik}(+) \right] \tag{4.55},$$

and projects ahead the current master state and error covariance estimates. Note that in equations (4.54) and (4.55) instead of using $P_{mk}(+)$ and $\hat{x}_{mk}(+)$ they use $P_{mk}(-)$ and $\hat{x}_{mk}(-)$. This is because there is no measurement information available to update the master filter.

It is necessary to mention here that, by using the information sharing principle, local filters use partial information, subsequently the output of local filters lack optimality. However, if all conditions are met, the global solution is optimal.

An issue to be confronted in the FKF design is how the total information is to be divided among individual filters to achieve the highest improvement in performance, efficiency and fault tolerance.

4.4 Summary

In this chapter the standard Kalman filter algorithm together with a popular alternative form, have been described. An important issue is the consistency of the estimates given by the filter. Thus two statistical tests have been described to verify this aspect of the algorithm. For the purposes of this work, these tests will be of great relevance in the next chapter where a fuzzy logic-based adaptive Kalamn filter will be formulated.

Basically there are three architectures for multisensor data fusion based on the standard Kalman filter: centralised, decentralised and federated Kalman filtering. These approaches have been described in this chapter. The idea of decentralised processing will be used in chapter 6 where an hybrid multisensor data fusion architecture will be developed.

CHAPTER 5

ADAPTIVE KALMAN FILTERING THROUGH FUZZY LOGIC

5.1 Statement of the problem and motivation

The problem of improving the performance of a standard Kalman filter can be divided in two parts, a modelling problem and an estimation problem [Mohamed and Schwarz, 1999]. First, the modelling problem basically is connected with the development of better models that more accurately describe the system in question. In this case the uncertain parameters needed to be adjusted can be part of the system model (i. e. state transition matrix) and/or the measurement model (i. e. measurement design matrix). On the other hand, the estimation problem is concerned with achieving better estimates through the proper use of available process and measurement information. In this case the parameters to be adjusted are the statistical process noise and measurement noise information though the covariance matrices R_k and/or Q_k.

As described in the previous chapter, the standard Kalman filter [Kalman, 1960] formulation (from here referred to as SKF) assumes complete *a priori* knowledge of the process and measurement noise statistics, matrices Q_k and R_k. Whilst often they are assumed to be constant matrices, they may vary with time (index k) [Welch and Bishop, 1995] and, if this is so, then the nature of this variation is assumed to be known as well. However, in most practical applications these matrices are initially estimated or, in fact, are unknown. Additionally, both the measurement noise and process noise are assumed to be uncorrelated zero-mean Gaussian white noise sequences, which is reasonable in most cases. The problem here is that the optimality of the estimation algorithm in the SKF setting is closely connected to the quality of the a priori process noise and measurement noise statistics [Brown and Hwang, 1997] [Mehra, 1970]. It has been shown that inadequate initial statistics reduce the precision of the filter estimated states or introduce biases to the filter estimates [Sangsuk-Iam and Bullock, 1990]. In fact, incorrect a priori information can cause practical divergence of the filter [Fitzgerald, 1971].

From the above comments it can be argued that using a SKF designed with fixed noise statistics in a changing dynamic environment is a major drawback. Thus, there is motivation for making the SKF adaptive with respect to the exact environment. The purpose of an adaptive Kalman filter formulation (from here referred to as AKF) is to reduce or bound the errors in the estimation by modifying or adapting the Kalman filter to the real data.

In this chapter an on-line adaptive scheme of the Kalman filter employing the principles of fuzzy logic is presented. The adaptation is in the sense of adaptively adjusting the noise covariance matrices R_k and/or Q_k from data as they are obtained. The main advantages derived from the use of a fuzzy logic technique, compared to traditional adaptation schemes, are the simplicity of the approach, the possibility of including heuristic knowledge about the phenomenon under consideration, and the ability to deal with uncertain information.

The chapter is organized as follows. First a review of the traditional approaches to AKF is presented. After that, the fuzzy logic-based AKF approach is outlined. Next, in order to demonstrate the effectiveness of this approach, an illustrative example is presented and comparisons with respect to the SKF and traditional AKF approaches are given. Finally, conclusions are given which summarise the outcome of this chapter.

5.2 Traditional adaptive Kalman filter approaches

Since the development of the SKF algorithm [Kalman, 1960] different traditional AKF formulations have been devised [Mehra, 1972] [Moghaddamjoo and Kirlin, 1989] [Mohamed and Schwarz, 1999] [Jazwinski, 1969]. A common factor in all these approaches is the use of the measured data and the filter residual sequence in order to estimate the unknown noise statistics. Recently, Mohamed and Schwarz [1999] have classified these procedures into two main approaches: multiple-model-based adaptive estimation (MMAE) and innovation-based adaptive estimation (IAE). In both techniques the concept of utilising the new information available in the residual is used but they differ in their implementation.

In the SKF algorithm the residual sequence, generally denoted as r_k, is the difference between the actual measurement z_k received by the filter and its predicted value $H_k \hat{x}_{k(-)}$:

$$r_k = (z_k - H_k \hat{x}_{k(-)}) \tag{5.1}.$$

If all the assumptions for an optimal KF are met, then the residual sequence is a linear combination of independent Gaussian random variables [Jazwinski, 1970] [Maybeck, 1979]. As a result, the residual sequence is a white Gaussian sequence of mean zero and covariance $S_k = H_k P_{k(-)} H_k^T + R_k$. This means that the value of the residual r_k cannot be predicted from its previous values. For this reason the residual sequence represents the information content in the new observation and is considered the most relevant source of information for the filter adaptation. In addition, from the SKF algorithm it is known that the Kalman gain is proportional to the inverse of S_k. Thus, the residual sequence becomes a useful tool for judging the performance of the filter in actual practice. The residual sequence is available to us, as is its statistics. By checking whether residuals indeed possess their (theoretical) statistical properties, the performance of the Kalman filter can be assessed.

In the MMAE approach, a bank of Kalman filters runs in parallel with different models of the statistical filter information matrices, i.e. the process noise covariance matrix Q_k and/or the measurement noise covariance matrix R_k. In the IAE technique, the adaptation is made directly to the statistical information matrices Q_k and/or R_k based on the changes observed in the filter residual. A brief review of these approaches is given next.

5.2.1 Multiple model adaptive estimation algorithm

In the multiple model adaptive estimation (MMAE) approach, originally proposed by Magill [1965], a bank of SKFs runs in parallel [Brown and Hwang, 1997] [Caputi, 1995] [Maybeck, 1989], as shown in figure 5.1. Each SKF runs under a different realisation of the uncertain parameter vector α (the process noise covariance matrix Q_k and/or the measurement noise covariance matrix R_k).

At each time step, k, the MMAE filter does three things, as follows:

- First, each SKF in the bank of filters computes its own estimate $\hat{x}_k(\alpha_i)$, which is hypothesised assuming that $\alpha = \alpha_i$ (for i = 1,2,…, L; L is the total number of SKFs used).
- Second, the system computes the *a posteriori* probabilities for each of the hypotheses using Bayes' rule as,

$$p(\alpha_i \mid z_k) = \frac{p(z_k \mid \alpha_i)\, p(\alpha_i)}{\sum_{j=1}^{L} p(z_k \mid \alpha_j) p(\alpha_j)} \qquad i = 1, 2, \ldots, L \tag{5.2}$$

where $p(\alpha_i \mid z_k)$ is the probability that the parameter vector, α, equals the *i-th* vector α_i at time k given all the past measurements up to and including the current measurement, i.e. $z_1,\ldots, z_k$. The distribution $p(\alpha_i)$ is assumed known, although in general, $p(\alpha_i)$ is unknown, and hence a uniform distribution is assumed. The unknown parameter vector is assumed to have a finite number of possible realisations [Magill, 1965] [Chear et al, 1997], so the conditional densities $p(z_k|\alpha_i)$ in equation (5.2) are computed recursively as:

$$p(z_k \mid \alpha_i) = \frac{1}{(2\pi)^{m/2} \left| S_k \right|^{1/2}} \cdot \mathrm{e}^{-\frac{1}{2} r_k^T S_k^{-1} r_k} \cdot p(z_{k-1} \mid \alpha_i) \tag{5.3}$$

where m is the dimension of the measurement vector, r_k is the residual, and S_k its corresponding covariance matrix, both calculated in the i-th SKF. Note that the denominator in (5.2) is simply the sum of all the computed numerator terms and thus is the scale factor required to ensure that all $p(\alpha_i \mid z_k)$ sum to one.

- Finally, the scheme forms the adaptive optimal estimate of x as a weighted sum of the estimates produced by each of the individual SKFs as,

$$\hat{x}_k = \sum_{i=1}^{L} \hat{x}_k(\alpha_i)\, p(\alpha_i \mid z_k) \tag{5.4}$$

where $p(\alpha_i \mid z_k)$ is the weighting factor of the i-th SKF.

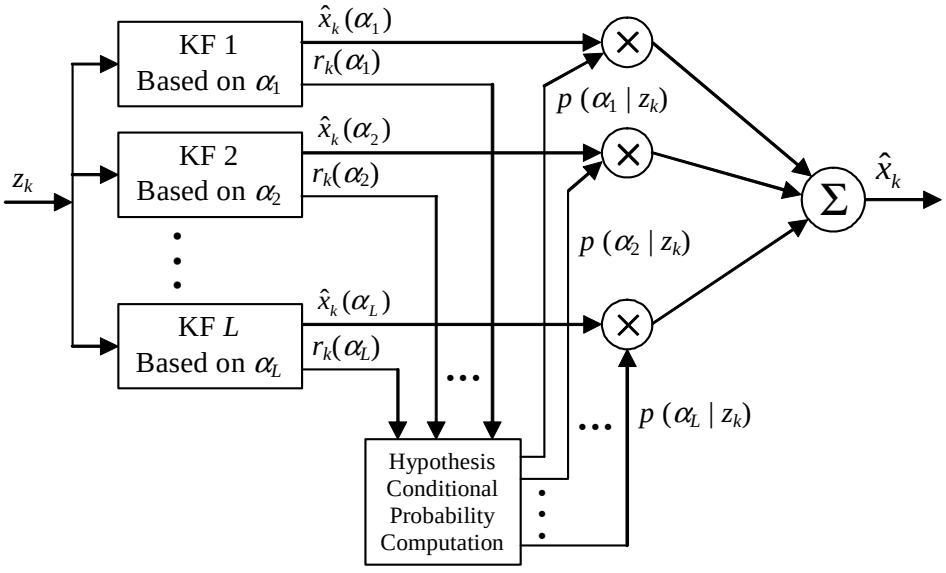


Figure 5.1 Structure of the MMAE filter.

As measurements evolve with time, the adaptive scheme learns which of the filters is the "correct" one and its weighting factor will tend to one, while the weighting factors of the remaining "mismatched" SKFs will tend toward zero. The bank of filters accomplishes this by

looking at the sums of the weighted squared residuals. The SKF with the smallest sum prevails. This is because the filter that has been modelled correctly will produce residuals with near zero mean. Of course, unless the bank contains all the possible realisations of the parameter vector, the correct filter will not necessarily be the optimal, but only the "best" one from those available.

The described MMAE approach has some drawbacks. Since the unknown parameters are assumed to be constant over time, this kind of adaptive filter cannot be readjusted if the parameter actually varies slowly with time. Some ad hoc procedures, such as periodic re-initialisation, have to be used if the scheme is to adapt to a slowly varying parameter situation. Another problem in the MMAE approach is the fact that having a bank of SKFs running in parallel increases the computational requirements when more SKFs are considered to be in the bank.

Moreover, the main problem of the MMAE approach is that its performance is dependent upon a significant difference between the residual characteristics in the correct and mismatched filters. To avoid errors in selecting the correct filter, it is important not to add too much dynamics pseudonoise during filter tuning, since this tends to mask differences between good and bad models [Maybeck, 1989] [Caputi, 1995].

5.2.2 Innovation based adaptive estimation algorithm

The innovation based adaptive estimation (IAE) approach [Mehra, 1970, 1972] [Mohamed and Schwardz, 1999] is based on the improvement of the filter performance through the adaptive estimation of the filter statistical information, the matrices R_k and/or Q_k. The adaptation mechanism is based on the whiteness of the filter residual sequence, equation (5.1). In this technique the measurement and process noise covariance matrices are adapted directly as follows:

$$\hat{R}_k = \hat{C}r_k - H_k P_{k(-)} H_k^T \tag{5.5}$$

$$\hat{Q}_k = K_k \hat{C}r_k K_k^T \tag{5.6}$$

where $P_{k(-)}$ is the state covariance matrix before update, K_k is the Kalman gain matrix, both obtained in the SKF algorithm; and $\hat{C}r_k$ is an estimate of the actual covariance matrix of the residual sequence.

Having available the residual sequence r_k, its actual covariance matrix $\hat{C}r_k$ in (5.5) and (5.6) is approximated by its sample covariance through averaging inside a sliding estimation window of size *WS*,

$$\hat{C}r_k = \frac{1}{WS}\sum_{i=i_0}^{k} r_i r_i^T \tag{5.7}$$

where $i_0 = k - WS + 1$ is the first sample inside the estimation window. The window size, *WS*, is chosen empirically to give some statistical smoothing.

The described adaptive KF algorithm implies that two additional blocks for computing the actual residual covariance matrix and both R_k and/or Q_k have to be added to the SKF formulation, as is shown in figure 5.2.

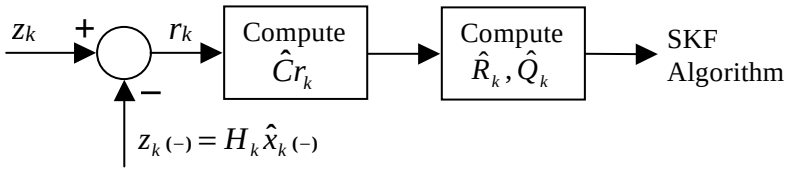


Figure 5.2 Additional blocks in the IAE filter.

The derivation of (5.5) and (5.6) were initially formulated by Mehra [1972] and lately re-derived by Mohamed [1999] using a maximum likelihood criterion (the reader interested in the whole procedure is referred to the cited references).

There is no method that is completely free of some limitations. In the described IAE care has to be taken in the choice of the sliding window size. This is particularly important if both R_k and Q_k are adapted simultaneously, which may end in a suboptimal solution or even filter divergence [Mohamed, 1999]. Also, off-line covariance propagation is not possible within the adaptive Kalman filter algorithm because of its dependency on the residual sequence that is in turn dependent on the external measurements.

5.3 Development of a fuzzy logic-based adaptive Kalman filter

In this section a novel fuzzy logic-based adaptive Kalman filter is developed. First a review of the existenting approaches is given. After that, the proposed fuzzy logic-based adaptive Kalman filter is described. Three cases are considered: adaptive estimation of the measurement noise covariance matrix R_k only, adaptive estimation of the process noise covariance matrix Q_k only, and adaptive estimation of the measurement and process noise covariance matrices, R_k and Q_k, simultaneously. Finally, some remarks about the stability of the adjusting procedure are given.

5.3.1 Previous works

In the past, some domain specific fuzzy logic-based approaches to AKF have been proposed [Lalk, 1994] [Wang and Goh, 1999] [Kobayashi *et al*, 1995, 1998]. In these works, some domain specific performance measures have been considered as input features to a fuzzy inference system (FIS) which works in supervisory mode adjusting some of the KF uncertain parameters. For example, in Kobayashi, *et al* [1998], three different FISs were used to adjust the matrices P_k, Q_k and R_k in order to reduce the effects of errors due to sensor inaccuracies in a global positioning system (GPS). R_k is adjusted through a FIS which inputs are based on the distance travelled by a vehicle between GPS updates and the geometrical dilution of precision (GDOP value) of the receiver as contributors to sensor noise. The covariance matrices P_k and Q_k are influenced by the estimation performance and hence are adjusted online through another two FISs involving such performance metrics as inputs. Thus, approaches such as these can only be applied when plant and sensor noise sources are identified and, consequently, their degradation or improvement can be monitored to tune the noise covariances online. In most real life applications such information is either not available at all or hard to obtain. Therefore, in most real situations, measurement residuals are the only available information for adaptation purposes.

Some authors have tried to derive general application AKF approaches. For example, in [Abdelnour *et al*, 1993] and later in [Sasiadek and Wang, 1999] the use of a fuzzy logic controlled exponential-weighting scheme for preventing filter divergence is proposed. The idea explored there is that of monitoring the whiteness property of the filter residual in order to

evaluate the performance of the filter. If the filter deviates from certain bounds, then corrective actions are taken through adjusting the exponential weighting factor. This factor increases or decreases the noise covariance matrices R_k and Q_k. But, as was demonstrated initially by Alspach [1972] and reaffirmed by Sangsuk-Iam and Bullock [1990], the whiteness test on the residual sequence is insufficient to adequately evaluate the KF performance. Another example is the work of Jetto *et al* [1999] where a covariance matching technique is used to develop a fuzzy logic based approach to AKF. Here the FIS is used to adjust a factor, through which, the process noise covariance matrix Q_k is tuned. However, the adjustment is carried out solely based on the size of the filter residual. The idea was to maintain the magnitude of the filter residual neither too near nor too far from zero by increasing or reducing the value of Q_k. In this case, the monitoring scheme used to evaluate the KF performance is rather involved and not easy to implement.

Unlike the above approaches, it is argued that, in this work, the proposed fuzzy logic-based AKF scheme is applicable across all domains. In addition, due to the use of fuzzy logic, it is easy to understand and implement. The proposed fuzzy logic-based AKF algorithm is described next.

5.3.2 The proposed fuzzy logic-based adaptive Kalman filter

In this section a fuzzy logic-based adaptive Kalman filter (from here referred to as FL-AKF) is presented [Escamilla and Mort, 2000; 2001c]. The adaptation is in the sense of dynamically adjusting on-line the measurement noise covariance matrix R_k and/or the process noise covariance matrix Q_k using a Fuzzy Inference System (FIS). This relaxes the a priori measurement noise statistical assumptions and significantly benefits the Kalman filter performance if the noise statistics change or evolve with time. The main advantages derived from the use of fuzzy techniques, with respect to traditional adaptation schemes, are the simplicity of the approach, the possibility of including heuristic knowledge about the phenomenon under consideration, and the ability to deal with uncertain information.

The fuzzy logic adaptation scheme is based on the technique known as covariance-matching [Mehra, 1972]. The basic idea behind this technique is to make the residual sequence consistent with its theoretical covariance value. If a statistical analysis of the residual sequence shows discrepancies between its theoretical and its actual covariance, this latter measure being approximated through averaging inside a moving window, then adjustments for matrices R_k and/or Q_k are derived. The adjustments are generated on-line by a FIS based on the knowledge of the size of the discrepancy. In this way the size of the discrepancy is reduced and maintained at a minimum while at the same time the consistency between the residuals and their statistics is preserved.

5.3.2.a Adaptive estimation of the measurement noise covariance matrix R_k only

In the context of the SKF algorithm the measurement noise covariance matrix R_k represents the accuracy of the measurement instrument. Thus, the enlargement of the covariance matrix R_k for measured data means that we trust this measured data less and have more faith in the prediction. Assuming that the process noise covariance matrix Q_k is completely known, here an algorithm employing the principles of fuzzy logic is derived to adaptively adjust the matrix R_k. This is achieved in two steps; first, having available the residual sequence r_k, defined by Eq. (5.1), its theoretical covariance is,

$$S_k = H_k P_{k(-)} H_k^T + R_k \tag{5.8}$$

obtained in the SKF algorithm. Second, if it is noted that the theoretical covariance of r_k has discrepancies with its actual value, then a FIS derives adjustments for R_k based on the knowledge of the size of this discrepancy. The objective of these adjustments is to correct this mismatch as much as possible and, in this way, maintain the consistency between the theoretical and actual residual statistics.

In order to detect and monitor the size of the discrepancy between S_k and its actual value, a new variable is defined. This variable is called the Degree of Mismatch (referred to as DoM_k),

$$DoM_k = S_k - \hat{C}r_k \tag{5.9}$$

where the actual residual covariance $\hat{C}r_k$ is estimated by equation (5.7), rewritten here,

$$\hat{C}r_k = \frac{1}{WS}\sum_{i=i_0}^{k} r_i r_i^T$$

where $i_0 = k - WS + 1$ is the first sample inside the sliding estimation window. The window size, WS, is chosen empirically to give statistical smoothing.

The basic idea of adaptation used by a FIS to derive adjustments for R_k is as follows. It can be deduced from (5.8) that an increment in R_k will increment S_k, and vice versa. Thus, R_k can be used to vary S_k in accordance with the value of DoM_k in order to reduce the discrepancies between S_k and $\hat{C}r_k$. Note that all matrices S_k, $\hat{C}r_k$, R_k and DoM_k have the same dimension $m \times m$ (recall that m is the dimension of the measurement vector z_k). Thus, under the assumption that the measurement noise is an uncorrelated and Gaussian noise sequence, R_k is a diagonal matrix whose elements are the variances of the individual components of the measurement noise vector v_k. This means that the diagonal elements of R_k can be adapted in accordance with the diagonal elements of DoM_k. From here, three general rules of adaptation are defined:

1. If $DoM_k(i,i) \cong 0$ (this means $S_k(i,i)$ and $\hat{C}r_k(i,i)$ match almost perfectly) then maintain $R_k(i,i)$ unchanged.
2. If $DoM_k(i,i) > 0$ (this means $S_k(i,i)$ is greater than its actual value $\hat{C}r_k(i,i)$) then decrease $R_k(i,i)$.
3. If $DoM_k(i,i) < 0$ (this means $S_k(i,i)$ is smaller than its actual value $\hat{C}r_k(i,i)$) then increase $R_k(i,i)$.

where $S_k(i,i)$, $\hat{C}r_k(i,i)$, $R_k(i,i)$ and $DoM_k(i,i)$, $i=1, 2,\ldots, m$; are the diagonal elements of S_k, $\hat{C}r_k$, R_k and DoM_k, respectively.

Thus, a single-input-single-output (SISO) FIS can be used to sequentially generate the tuning or correction factors for the elements in the main diagonal of R_k, and this correction is made in this way,

$$R_k(i,i) = R_{k-1}(i,i) + \Delta R_k \tag{5.10}$$

where ΔR_k is the adjusting factor for $R_k(i,i)$. ΔR_k is the FIS output and $DoM_k = DoM_k(i,i)$ is the FIS input. A graphical representation of this adjusting process is shown in figure 5.3.

Following the general rules of adaptation, the FIS can be implemented considering three fuzzy sets for DoM_k: N = Negative, ZE = Zero, and P = Positive, describing the degree of mismatch; and three fuzzy sets for ΔR_k: I = Increase, M = Maintain, and D = Decrease, describing the action of correction to be taken. These membership functions are shown in figure 5.4. There, the fuzzy sets are defined by the parameters a and b, which can be selected in accordance with the system under consideration. For example, if there is knowledge about the range in which the values in R can vary, then the maximum possible value can be selected as the initial value for a. The parameter b, which defines the maximum size of adjustment to the values in R, can be selected as a percentage of a, for example the 10% will produce smooth adjustments. Obviously, these initial values can be further tuned from simulation. Hence, only three fuzzy rules are included in the FIS rule base:

1. If $DoM_k = N$, then $\Delta R_k = I$
2. If $DoM_k = ZE$, then $\Delta R_k = M$
3. If $DoM_k = P$, then $\Delta R_k = D$.

Finally, using the compositional rule of inference sum-prod and the center of area (COA) defuzzification method, R_k is adjusted in the FL-AKF as given by (5.10).

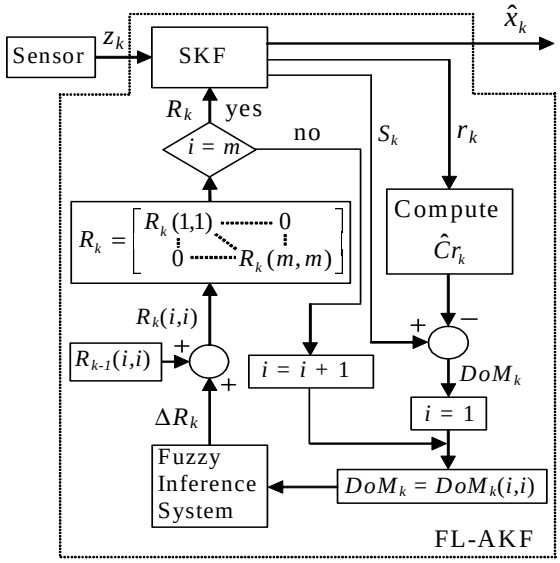


Figure 5.3 Graphical representation of the sequential algorithm to adjust R_k using a FIS.

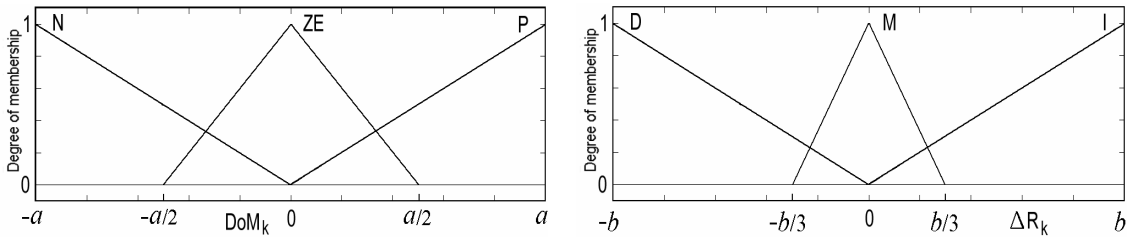


Figure 5.4 Membership functions for DoM_k and ΔR_k.

From experimentation and simulation of many systems it was found that the fuzzy sets defined in the way shown in figure 5.4 give good results in most cases. However, if necessary, the shapes and number of fuzzy sets in the membership functions can be modified to fit the requirements of the problem under consideration. This is possible thanks to the use of the fuzzy logic technique which allows capturing the knowledge that the designer has over the system under consideration.

It is necessary to remark here that, as alternatives to the sequential implementation, the adaptation algorithm can be implemented in two additional ways. In the first alternative, m parallel SISO FISs can be considered in order to adapt at once all the elements in the main diagonal of R_k, as it is shown graphically in figure 5.5(a). In the second alternative, a multiple-input-multiple-output (MIMO) FIS with $3\times m$ rules (a group of three rules for each element in the main diagonal is needed) in the rule base can be used to adjust at once all the elements in the main diagonal of R_k. This last alternative is represented graphically in figure 5.5(b). The use of any of the three ways of implementation: sequential, parallel or MIMO, depends on the computational resources and the problem under consideration.

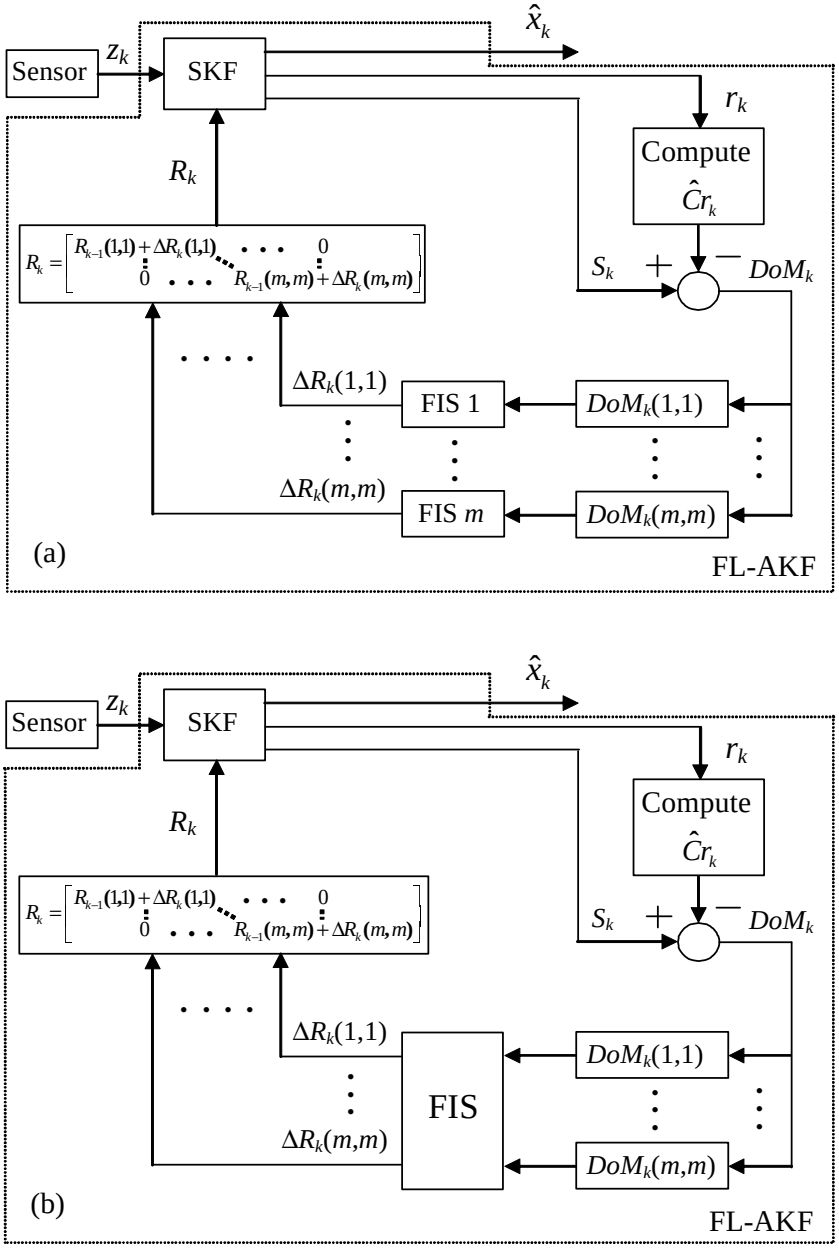


Figure 5.5 Graphical representations of the alternatives for implementing the algorithm to adjust R_k. (a) m parallel SISO FISs implementation. (b) MIMO FIS implementation.

Similarly, from experimentation it was found that a good size for the moving window in (5.7) is $WS = 15$.

5.3.2.b Adaptive estimation of the process noise covariance matrix Q_k only

The process noise covariance matrix Q_k represents the uncertainty in the process model. An increase in the covariance matrix Q_k means that we have less faith in the process model and have more confidence in the measurement. Assuming that the measurement noise covariance matrix R_k is completely known, here an algorithm employing the principles of fuzzy logic is derived to adaptively adjust the matrix Q_k. First, note that Eq. (5.8) can be rewritten as:

$$S_k = H_k(\Phi_{k-1}P_{k-1(+)}\Phi_{k-1}^T + Q_{k-1})H_k^T + R_k \qquad (5.11).$$

from the SKF algorithm. It can be deduced from Eq. (5.11) that the same basic idea of adaptation used by a FIS to derive adjustments for R_k can be used by another FIS to derive adjustments for Q_k. However, if the state and measurement vectors of the system under consideration have not the same dimension, then it will result that the matrices S_k, $\hat{C}r_k$, and DoM_k will have dimension $m\times m$, while the matrix Q_k has dimension $n\times n$. This means that the previous adjusting procedure cannot be used in this case.

In this case what is needed is to project the problem from the measurement space with dimension m, to the state space with dimension n. This projection can be done by including the Kalman gain in the calculation of DoM_k, this is:

$$DoMK_k = K_kS_kK_k^T - K_k\hat{C}r_kK_k^T \qquad (5.12a)$$

$$= K_k[H_k(\Phi_{k-1}P_{k-1(+)}\Phi_{k-1}^T + Q_{k-1})H_k^T + R_k]K_k^T - K_k\hat{C}r_kK_k^T \qquad (5.12b)$$

where the new variable $DoMK_k$, of dimension $n\times n$, is referred to as the Degree of Mismatch through the Kalman gain K_k, S_k is the theoretical covariance of the residual sequence, and $\hat{C}r_k$ is its estimated value given by Eq. (5.7). The reason for including K_k in (5.12) is basically because it is a suitable matrix, it has dimension $n\times m$, and because it is available from the SKF algorithm. Furthermore, if (5.12) is compared with (5.6), then it is realised that the factor $K_k\hat{C}r_kK_k^T$ is the estimation of the actual value of the process noise covariance matrix, $\hat{Q}_k$ [Mohamed, 1999]. Thus, (5.12) can be interpreted as the difference between the 'theoretical' value of Q_k and its actual approximated value. Hence, it can be inferred from Eq. (5.12b) that a variation in Q_k will affect the value of S_{k+1}. If Q_k is increased then S_{k+1} will be increased, and vice versa. This means that by augmenting or diminishing the value of Q_k the mismatch between S_k and $\hat{C}r_k$, detected through $DoMK_k$, can be reduced. Therefore, under the assumption that the process noise is an uncorrelated and Gaussian noise sequence, Q_k is a diagonal matrix whose elements are the variances of the individual components of the process noise vector w_k. This means that the diagonal elements of Q_k can be adapted in accordance with the diagonal elements of $DoMK_k$. From here, three general rules of adaptation are derived:

1. If $DoMK_k(i,i) \cong 0$ (this means $S_k(i,i)$ and $\hat{C}r_k(i,i)$ match almost perfectly) then maintain $Q_k(i,i)$ unchanged.
2. If $DoMK_k(i,i) > 0$ (this means $S_k(i,i)$ is greater than its actual value $\hat{C}r_k(i,i)$) then decrease $Q_k(i,i)$.
3. If $DoMK_k(i,i) < 0$ (this means $S_k(i,i)$ is smaller than its actual value $\hat{C}r_k(i,i)$) then increase $Q_k(i,i)$.

where $Q_k(i,i)$ and $DoMK_k(i,i)$, i=1, 2,…,n; are the diagonal elements of Q_k and $DoMK_k$, respectively.

Thus, a SISO FIS can be used to sequentially generate the tuning or correction factors for the elements in the main diagonal of Q_k, and this correction is made in this way,

$$Q_k(i,i) = Q_{k-1}(i,i) + \Delta Q_k \tag{5.13}$$

where ΔQ_k is the correction factor for $Q_k(i,i)$. ΔQ_k is the FIS output and $DoMK_k = DoMK_k(i,i)$ is the FIS input. A graphical representation of this adjusting process is shown in figure 5.6. Here it is necessary to remark that in this case the adaptation will be reflected in the next time step (k+1). This is because in fact Q_{k-1} is the matrix that is affecting S_k and not Q_k.

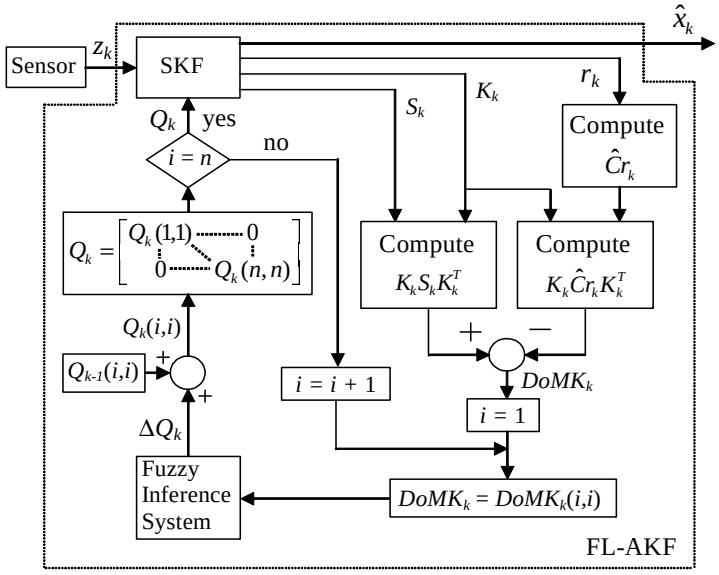


Figure 5.6 Graphical representation of the sequential algorithm to adjust Q_k using a FIS based on $DoMK_k$.

Following the general rules of adaptation, the FIS can be implemented considering three fuzzy sets for $DoMK_k$: N = Negative, ZE = Zero, and P = Positive, describing the degree of mismatch though K_k; and three fuzzy sets for ΔQ_k: I = Increase, M = Maintain, and D = Decrease, describing the action of correction to be taken. These membership functions are shown in figure 5.7. There, the fuzzy sets are defined by the parameters c and d, which can be selected in accordance with the system under consideration. For example, similar to the case for R_k, if the range in which the elements of Q_k can vary is known, then the maximum possible value can be selected as the initial value of parameter c. d can be selected as a percentage of c, e.g. 10%. Further tuning of these parameters can be performed based on simulation results. Hence, only three fuzzy rules are included in the FIS rule base:

1. If $DoMK_k = N$, then $\Delta Q_k = I$
2. If $DoMK_k = ZE$, then $\Delta Q_k = M$
3. If $DoMK_k = P$, then $\Delta Q_k = D$.

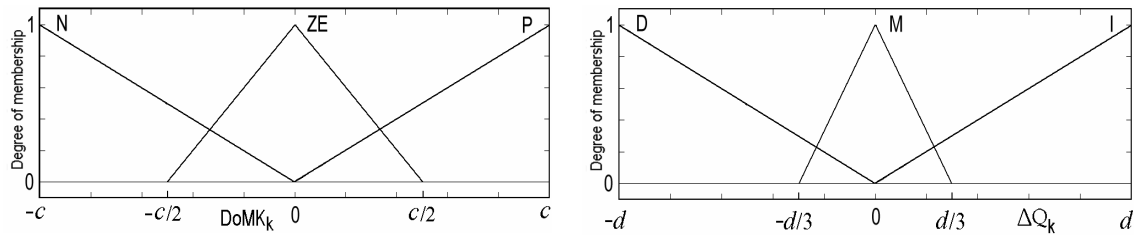


Figure 5.7 Membership functions for $DoMK_k$ and ΔQ_k.

Finally, using the compositional rule of inference sum-prod and the COA defuzzification method, Q_k is adjusted in the FL-AKF as given by Eq. (5.13).

Here also from experimentation was found that the shapes and distribution of the fuzzy sets shown in figure 5.7 give good results in most cases. Nevertheless, if required the shapes and number of fuzzy sets can be modified to fit the needs of the problem under consideration.

In a similar way as for R_k, as alternatives to the sequential implementation, the adaptation algorithm for Q_k can be implemented in two additional ways. In the first alternative, n parallel SISO FISs can be considered in order to adapt simultaneously all the elements in the main diagonal of Q_k, as it is shown graphically in figure 5.8(a). In the second alternative, a MIMO FIS with $3\times n$ rules (a group of three rules for each element in the main diagonal is required) in the rule base can be used to adjust simultaneously all the elements in the main diagonal of Q_k. This alternative is represented graphically in figure 5.8(b).

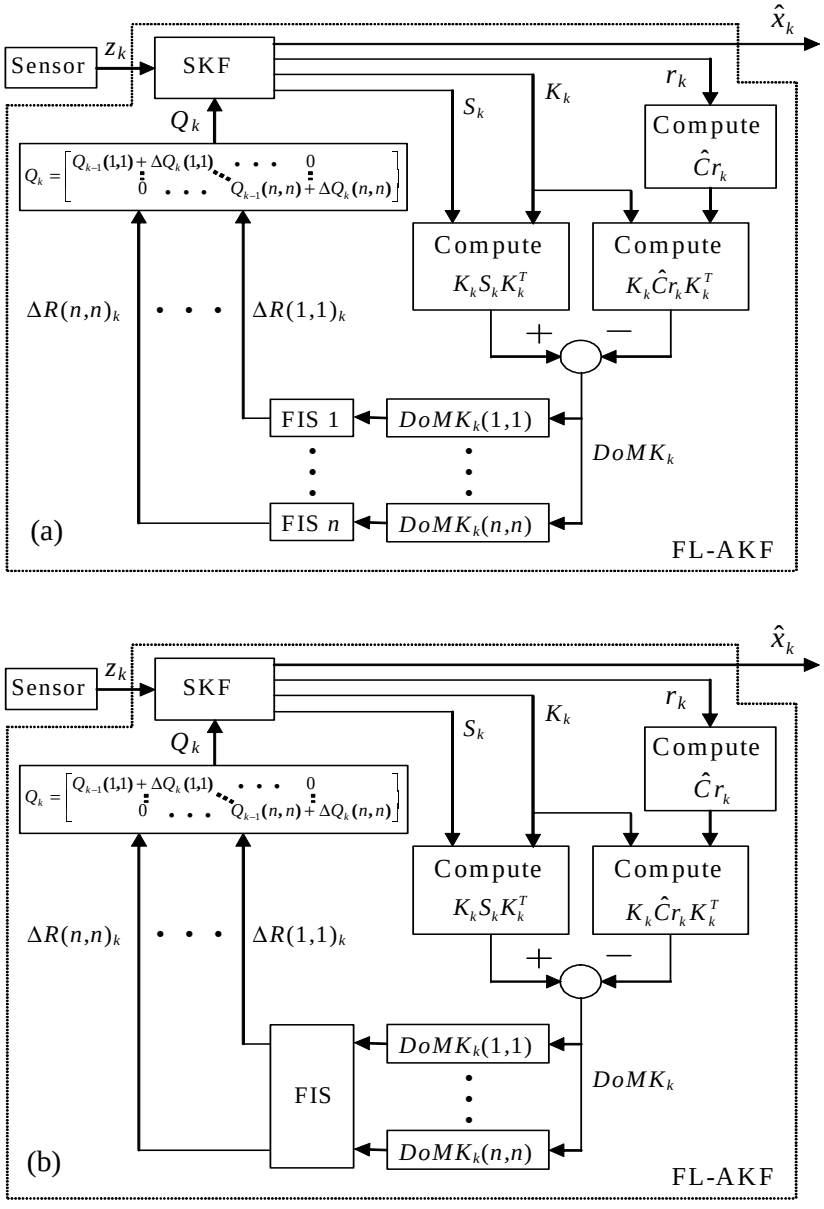


Figure 5.8 Graphical representations of the alternatives for implementing the algorithm to adjust Q_k. (a) n parallel SISO FISs implementation. (b) MIMO FIS implementation.

5.3.2.c Adaptive estimation of the measurement and process noise covariance matrices, R_k and Q_k, simultaneously

It is worth nothing here that, because the same scheme for adapting R-only and Q-only is used, in the case of simultaneously adapting R_k and Q_k numerical difficulties can be encountered in real data. Due to the approximations made, care has to be taken in the choice of the adaptive filter parameters such as the moving average window size.

In general, the simultaneous adaptation of R_k and Q_k may lead to an unstable and divergent filter estimate and should, therefore, be avoided.

5.3.3 Stability of the adjusting procedure

In the FL-AKF algorithm the measurement noise covariance matrix R_k (or the noise covariance matrix Q_k) is changed in such a way that the statistics of the filter's residuals approach that of the optimum Kalman filter. This procedure originates a kind of stabilising negative feedback in the statistics of the residual sequence of the filter. Although the stability formulation of this method is not readily accessible, due to the use of fuzzy logic techniques, it can be easily understood by the arguments given next.

The role of matrices R_k and Q_k in the SKF setting is to adjust the Kalman gain in such a way that it controls the filter bandwidth as the state and the measurement errors vary [Moghaddamjoo and Kirlin, 1989]. At steady state the filter gain remains constant, as matrices R_k and Q_k are kept constant, regardless of changes in the system dynamics or the update measurement quality. This problem is solved in the FL-AKF by adjusting the values of R_k and Q_k in an adaptive manner.

If only R_k is adapted (or only Q_k is adapted), then it can be argued that in this method R_k (or Q_k) is the only unknown parameter which controls the Kalman gain, Q_k (or R_k) is assumed to be known. Let us assume that, due to some disturbances (i.e. unknown sudden changes in v_k and/or w_k), the optimal (theoretical) covariance of the residual sequence becomes less than its actual (estimated) value. The resultant residual sequence will then become inconsistent with its covariance. Detection of this inconsistency through DoM_k (or $DoMK_k$) will demand an increment in R_k (or Q_k), which will, in turn, increase the theoretical value of the residual covariance (S_k changes in the direction which approaches its actual value). On the other hand, if the optimal (theoretical) covariance of the residual sequence is larger than its actual value, detection of this inconsistency through DoM_k (or $DoMK_k$) will demand a decrement in R_k (or Q_k). Decreasing R_k (or Q_k) will then decrease the theoretical value of the residual covariance (S_k changes again in the direction which approaches its actual value). This correction continues until S_k reaches a quasi steady-state in the vicinity of its actual value. Therefore, deviations of S_k from its actual value, due to any changes in the actual noise sequences, not only are controlled by R_k (or Q_k), but also will be reduced in time. This behaviour can be referred to as a negative feedback which has a stabilising role in the overall performance of the fuzzy logic-based adaptation algorithm.

5.3.4 Illustrative example

To demonstrate the effectiveness and accuracy of the developed FL-AKF, several experiments have been carried out and results are presented in this section. These experiments were developed under the MATLAB/SIMULINK modelling environment.

Consider the following dynamic system which models an object moving in a circular trajectory at constant speed with process noise and measurement noise [Zhu, 1999]:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \Phi \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \begin{bmatrix} w_k^1 \\ w_k^2 \end{bmatrix} \tag{5.14}$$

$$\begin{bmatrix} z_{k+1}^1 \\ z_{k+1}^2 \end{bmatrix} = H \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \begin{bmatrix} v_k^1 \\ v_k^2 \end{bmatrix} \tag{5.15}$$

where Φ is a constant rotational matrix, and H is a constant measurement matrix, given by,

$$\Phi = \begin{bmatrix} \cos(2\pi/300) & \sin(2\pi/300) \\ -\sin(2\pi/300) & \cos(2\pi/300) \end{bmatrix} \tag{5.16}$$

$$H = \begin{bmatrix} 0.7071 & 0.7071 \\ -0.7071 & 0.7071 \end{bmatrix} \tag{5.17}.$$

As the speed of the object is constant, its state vector is simply its position in the plane, a two-element vector: $\boldsymbol{x_k}=[x_k\ y_k]^T$. With initial state $\boldsymbol{x}_0=[100\ 0]^T$ and absence of noise the object will move in a circle of radius 100 about the origin of the coordinate space.

The initial conditions for Kalman filtering are defined as: $\hat{x}_{0(-)}=[100\ 0]^T$, $P_{0(-)}=[H^T H+0.1I_3]^{-1}$. The process and measurement noise vectors $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are uncorrelated zero-mean Gaussian white noise sequences with covariance matrices Q_k and R_k specified in each particular simulation.

The FISs used to adjust R_k and Q_k in the FL-AKF are specified in sections 5.3.2.a and 5.3.2.b. The parameters used to define the fuzzy membership functions are presented in table 5.1. The size of the sliding window in Eq. (5.7) was selected as 15.

Table 5.1 Parameters for the FISs used to adjust R_k and Q_k

Parameter	FIS used to adjust R_k	FIS used to adjust Q_k
a	5	
b	0.3	
c		5
d		0.3

Simulation 1: The purpose of this simulation is to investigate the performance of the developed FL-AKF under correct initial noise statistics. The performance of the FL-AKF is compared with those of a SKF, a traditional AKF using the IAE algorithm (referred to as TAKF-IAE and described in section 5.2.1), and a traditional AKF using the MMAE algorithm (referred to as TAKF-MMAE and described in section 5.2.2).

The system under consideration was simulated for 300 sec with a sample time T=0.5 sec. This means that the object completes two circles. The actual process and measurement noise covariance matrices are constant matrices given as:

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.18}$$

$$R = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.19}$$

To compare performances, the following mean squared error (MSE) measures were used:

$$MSEx = \frac{1}{n}\sum_{k=1}^{n}(x_k - \hat{x}_k)^T(x_k - \hat{x}_k) \tag{5.20}$$

$$MSEy = \frac{1}{n}\sum_{k=1}^{n}(y_k - \hat{y}_k)^T(y_k - \hat{y}_k) \tag{5.21}$$

$$MSE = \frac{1}{n}\sum_{k=1}^{n}(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k)^T(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k) \tag{5.22}$$

where x_k is the actual value of the position in the X axis and $\hat{x}_k$ its estimated value; y_k is the actual value of the position in the Y axis and $\hat{y}_k$ its estimated value; $\boldsymbol{x}_k$ is the actual value of the state vector and $\hat{\boldsymbol{x}}_k$ is its estimated value; n is the number of samples.

The consistency of the filter in each case is tested using the time-averaged normalised innovation squared (TANIS) measure [Bar-Shalom and Xiao-Rong, 1993], defined as:

$$\bar{\varepsilon}_r = \frac{1}{n}\sum_{k=1}^{n} r_k^T S_k^{-1} r_k \tag{5.23}$$

If the residuals are white, zero mean, and consistent with the calculated covariance S_k, then $\bar{\varepsilon}_r = m$, where m is the dimension of the measurement vector $\boldsymbol{z_k}$.

Results: Table 5.2 shows the Kalman gain matrix (K), residual covariance matrix (S), MSE measures, and TANIS values obtained for each case together with the initial values given to R_k and Q_k (referred to as R_0 and Q_0, which in this case are the actual correct values). Note that the results for the SKF case are optimal due to the use of the correct noise statistics. For the FL-AKF and the TAKF-IAE three results are presented, when R_k is adjusted only (R-only), when Q_k is adjusted only (Q-only) and when both R_k and Q_k are adjusted (R&Q). In the TAKF-MMAE four SKFs are considered in the bank of filters where one of them has the correct noise statistics values and the others are incorrectly specified, as indicated in the last two columns of the table. Obviously, in both the FL-AKF and TAKF-IAE approaches, the value of Q_k and/or R_k is dynamically tuned. For this reason, the values of K and S change as time progresses, and thus in Table 5.2 the values shown for K and S are those obtained averaging over all the samples. In the SKF case the values shown for K and S are those obtained once the system has reached the steady-state. In the TAKF-MMAE case the values presented for K and S are those of the SKF which the algorithm has selected as the best one.

To have a clearer picture of the differences, in Table 5.3 the percentage of degradation in performance of the FL-AKF, the TAKF-IAE, and the TAKF-MMAE algorithms with respect to the optimal SKF (considering the optimal MSE measure and TANIS value), which has the correct values of noise statistics, are presented. From Table 5.3 it can be noted that under correct noise statistics the FL-AKF performance degrades on average by only 4%, with respect to the optimal SKF. While the TAKF-IAE degrades on average by 19.4%, considering the R-only and the Q-only cases, because for the R&Q case, a divergence of the filter is observed. Since the TAKF-MMAE algorithm quickly converges to the optimal SKF, which is part of the bank of filters, the degradation in this last case is imperceptible. However, if the SKF with the correct

noise statistics is not in the bank of filters, then the degradation could be very severe, as will be shown in simulation 2.

Table 5.2 Performance under correct noise statistics: summary of results, simulation 1

Filter	**Adaptation**	***K***	***S***	**MSE*x***	**MSE*y***	**MSE**	**TANIS**	R_0	Q_0
SKF	Optimal	0.3542 -0.4384 0.3529 0.4357	4.0 -0.0099 -0.0099 2.618	0.7551	0.8532	1.608	2.002	*R*	*Q*
FL-AKF	R-only	0.3717 -0.4329 0.371 0.4307	3.74 -0.0068 -0.0068 2.78	0.7973	0.8747	1.672	2.144	*R*	*Q*
	Q-only	0.3428 -0.4232 0.3506 0.4388	3.964 0.0501 0.0501 2.593	0.7576	0.878	1.636	2.061	*R*	*Q*
	R&Q	0.3308 -0.3826 0.3419 0.4007	3.725 0.0538 0.0538 2.795	0.799	0.9122	1.711	2.175	*R*	*Q*
TAKF-IAE	R-only	0.39 -0.448 0.377 0.4311	3.904 0.0739 0.0739 2.969	0.9273	1.028	1.956	2.018	*R*	*Q*
	Q-only	0.2975 -0.4207 0.3011 0.4278	3.82 -0.0065 -0.0065 2.854	0.8567	1.028	1.884	2.331	*R*	*Q*
	R&Q	0.6023 -0.6058 0.3008 -0.0738	70.03 45.43 45.43 40.49	8.086	103.2	111.3	2.063	*R*	*Q*
TAKF-MMAE	SKF 1			0.7549	0.8531	1.608		5*R*	5*Q*
	SKF 2							2.5*R*	2.5*Q*
	SKF 3	0.3542 -0.4384 0.3529 0.4357	4.0 -0.0099 -0.0099 2.618				2.002	*R*	*Q*
	SKF 4							0.5*R*	0.5*Q*

Table 5.3 Percentage in degradation with respect to the optimal SKF: Simulation 1

Filter	Adaptation	% of degradation in performance	% of deviation TANIS test
FL-AKF	R-only	3.98	+7.09
	Q-only	1.74	+2.95
	R&Q	6.4	+8.64
TAKF-IAE	R-only	21.64	+0.8
	Q-only	17.16	+16.43
	R&Q	6821.64	+3.05
TAKF-MMAE	Correct noise statistics in the bank	0	0

In addition, the last column of table 5.3 shows the percentage of deviation (positive or negative) of the TANIS values with respect to their optimal value, 2; the greater the deviation, the greater the inconsistency of the filter. Note that in order to have a good judgement of the performances, not only it is necessary to look at the MSE values, but it is also important to look at the TANIS values too. Both must be near to their optimal values to assess that the filter is working correctly.

For a graphical view of results, in figure 5.9(a) the actual and estimated trajectories of the moving object obtained by the FL-AKF, R-only adaptation, are presented. Note that only a slightly difference in trajectories can be seen. In figure 5.9(b) the MSE for the estimated state vector is shown as a function of time. In figure 5.10(a) the adaptation of the diagonal elements of matrix R_k can be observed, this is a typical realisation. Note how these values reach a quasi steady-state very near to their actual values. In figure 5.10(b) the diagonal elements of matrix DoM_k are shown. Note how these values are well maintained, with a slight oscillation, near the value of zero. Additionally, figure 5.11(a) presents the variation observed in the elements of the Kalman gain matrix, while figure 5.11(b) shows the variation of the elements in the residual covariance matrix, both obtained by the FL-AKF, R-only case.

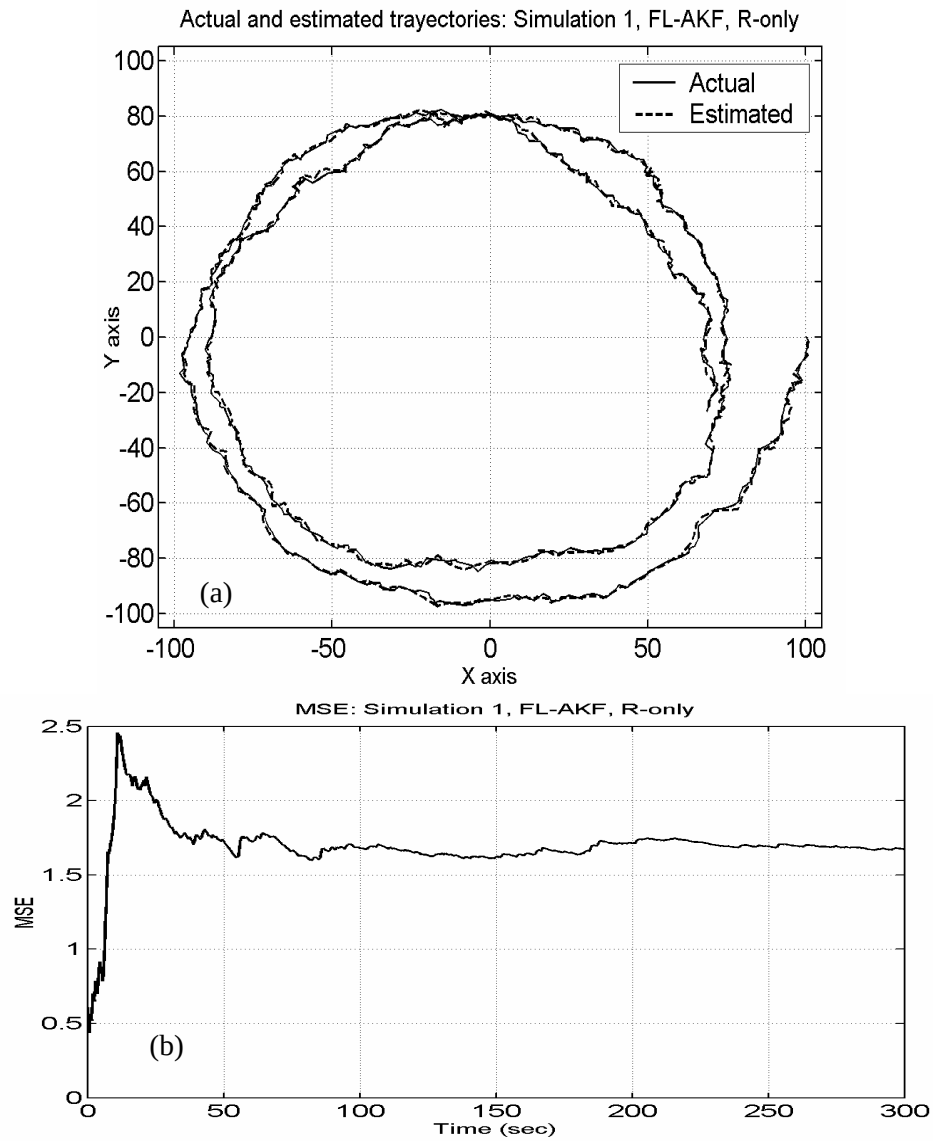


Figure 5.9(a) Actual and estimated trajectories. (b) MSE on the estimated state vector; simulation 1, FL-AKF, R-only case.

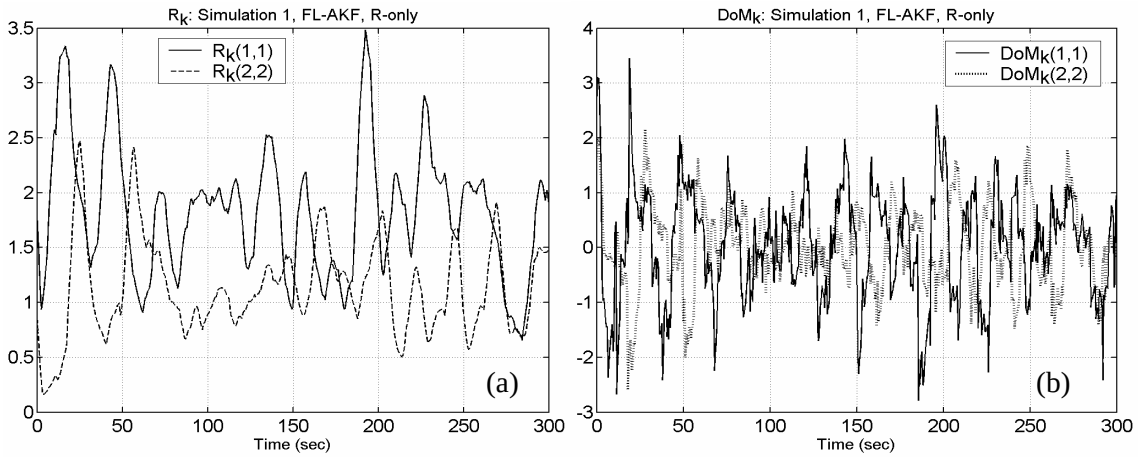


Figure 5.10(a) Estimated diagonal elements of the measurement noise covariance matrix R_k. (b) Diagonal elements of matrix DoM_k; simulation 1, FL-AKF, R-only case.

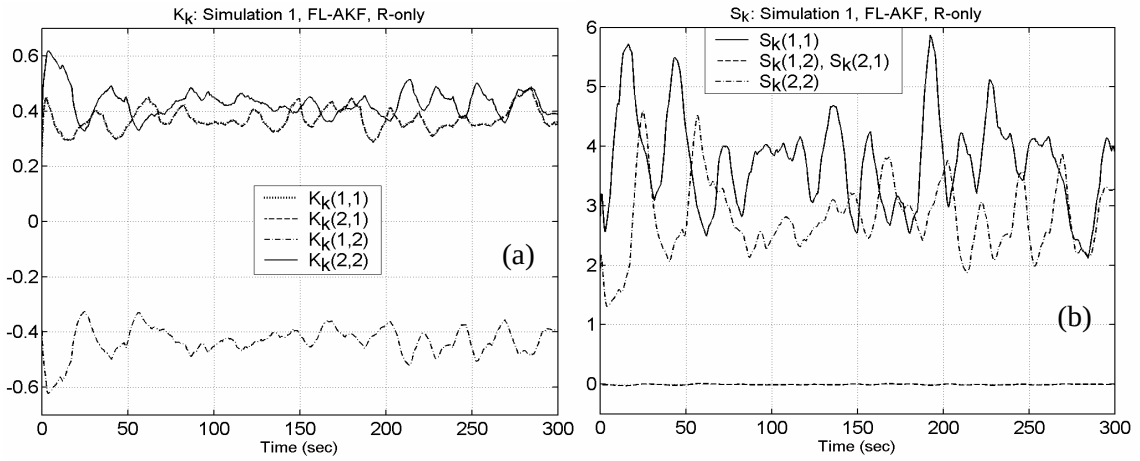


Figure 5.11(a) Elements of the Kalman gain matrix. (b) Elements of the residual covariance matrix; simulation 2, FL-AKF, R-only case.

Simulation 2: The purpose of this simulation is to investigate the performance of the FL-AKF under incorrect initial noise statistics. The performance of the proposed FL-AKF is compared with that of a SKF, a TAKF-IAE algorithm, and a TAKF-MMAE algorithm.

The system described by Eqs. (5.14) to (5.17) was simulated for 300 sec with a sample time T=0.5 sec. As in simulation 1, the actual process and measurement noise covariance matrices are constant matrices defined by Eq. (5.18) and Eq. (5.19). However, in this case the initial values for R and Q (R_0 and Q_0) are incorrectly specified, as is pointed out in the last two columns of Table 5.4. For example R_0=5R means that the initial value of R is five times bigger than its correct value.

Results: Table 5.4 shows the Kalman gain matrix, residual covariance matrix, MSE measures, and TANIS values obtained for each case together with the values given to R_0 and Q_0. For the SKF case three results are presented, when only R_0 is incorrect, when only Q_0 is incorrect, and when both R_0 and Q_0 are incorrect. For the FL-AKF and the TAKF-IAE three results are presented, when only R_k is adjusted (R-only), when only Q_k is adjusted (Q-only) and when both R_k and Q_k are adjusted (R&Q), with corresponding initial noise statistics given in the last two columns of the table. In the TAKF-MMAE case, four SKFs are considered in the bank of filters where none of them has the correct noise statistics, as is seen in the last two columns of the table. In the FL-AKF and TAKF-IAE cases the values of K and S change as time progress, thus in Table 5.4 the values shown for K and S are those obtained by averaging over all the samples. In the SKF case the values shown for K and S are those obtained once the system has reached the steady-state. In the TAKF-MMAE case the values presented for K and S are those of the SKF which the algorithm has selected as the best.

In Table 5.5 the percentage of degradation in performance of the SKF, the FL-AKF, the TAKF-IAE, and the TAKF-MMAE algorithms with respect to the optimal SKF (considering the optimal MSE measure and TANIS value), is presented. It can be seen from Table 5.5 that the performance of the SKF with incorrect initial noise statistics is severely affected. This is evident when R_0 or Q_0 is incorrect. However, note that when both matrices are incorrect, the value of K and the MSE measures are identical to those of the optimal filter; but the values of S and *TANIS* are very far from the optimal ones. This reveals a big inconsistency of the filter. This is the reason why it is argued here that the evaluation of the filter performance should be made based on both MSE and TANIS measures. This affirmation agrees with the results obtained by Alspach in his early work [1972], where it is shown that the same steady-state Kalman gain can be obtained with different values of R and Q. However, a given value of S and K is obtained with only a specific value of R and Q. This is supported by the results obtained with the FL-

AKF. Note in Table 5.4 that the values of K and S are very near to the optimal ones in the three cases considered. Thus, thanks to the fuzzy logic-based adaptation, good levels of performance and filter consistency are maintained, as is shown in Table 5.5. From this table it is obvious that the only algorithm capable of maintaining good performance as well as consistency, when initial incorrect noise statistics are given, is the proposed FL-AKF. Note, as well, how the performance of the TAKF-MMAE algorithm is severely affected when the SKF with correct noise statistics is not in the bank of filters. Moreover, this last algorithm cannot do any better than to converge to the filter with the best performance from the filters in the bank, which could be one that is very far from the optimal.

Table 5.4 Performance under incorrect noise statistics: summary of results, simulation 2

Filter	**Adaptation**	***K***	***S***	**MSE*x***	**MSE*y***	**MSE**	**TANIS**	R_0	Q_0
SKF	None	0.1921 -0.2558 0.1897 0.2511	13.7 -0.0357 -0.0357 7.793	0.9995	1.313	2.312	0.7001	5*R*	*Q*
	None	0.5415 -0.6043 0.5412 0.6036	8.531 -0.0147 -0.0147 6.854	1.041	1.046	2.087	0.9215	*R*	5*Q*
	None	0.3542 -0.4384 0.3529 0.4357	20.0 -0.0494 -0.0494 13.09	0.7549	0.8533	1.608	0.4005	5*R*	5*Q*
FL-AKF	R-only	0.3635 -0.4221 0.3627 0.4197	4.09 -0.0078 -0.0078 2.923	0.7814	0.8898	1.671	2.043	5*R*	*Q*
	Q-only	0.3581 -0.4387 0.3644 0.4513	4.192 0.0206 0.0206 2.8	0.7872	0.8609	1.648	1.958	*R*	5*Q*
	R&*Q*	0.3851 -0.4445 0.3972 0.4682	4.22 0.0835 0.0835 3.005	0.7958	0.8908	1.687	2.073	5*R*	5*Q*
TAKF-IAE	R-only	0.0356 -0.0229 -0.0035 0.0265	8.82E4 -1878 -1878 9.23E4	9.22E4	8.97E4	1.82E5	3.648	5*R*	*Q*
	Q-only	0.3036 -0.426 0.3069 0.4326	3.963 -0.0159 -0.0159 2.969	0.8777	1.023	1.901	2.295	*R*	5*Q*
	R&Q	0.9821 -1.041 0.9809 -1.067	8.755 -0.7715 -0.7715 7.541	6.277	4.785	11.06	2.069	5*R*	5*Q*
TAKF-MMAE	SKF 1			1.03	1.043	2.073		5*R*	*Q*
	SKF 2	0.5415 -0.6043 0.5412 0.6036	8.531 -0.0147 -0.0147 6.854				0.9215	*R*	5*Q*
	SKF 3							0.2*R*	0.2*Q*
	SKF 4							5*R*	5*Q*

Table 5.5 Percentage in degradation with respect to the optimal SKF: Simulation 2

Filter	Adaptation	% of degradation in performance	% of deviation TANIS test
SKF	None, incorrect R_0	43.78	-65.03
	None, incorrect Q_0	29.79	-53.97
	None, incorrect R_0 and Q_0	0	-79.99
FL-AKF	R-only	3.92	+2.05
	Q-only	2.49	-2.2
	R&Q	4.91	+3.55
TAKF-IAE	R-only	Diverge	+82.22
	Q-only	18.22	+14.63
	R&Q	587.8	+3.35
TAKF-MMAE	Incorrect noise statistics in the bank	28.92	-53.97

In order to appreciate the adaptation carried out in the FL-AKF, R-only case, in figure 5.12(a) the estimated diagonal elements of the measurement noise covariance matrix R_k are shown. Note how these values quickly converge to their actual values where they then are maintained with a slight oscillation. In figure 5.12(b) the diagonal elements of matrix DoM_k are shown and we see how these values are maintained near to zero. In figures 5.13(a) and 5.13(b) the elements of the Kalman gain matrix and the residual covariance matrix are shown, respectively. Here it can be appreciated how these values converge to quasi steady-states near to their optimal values.

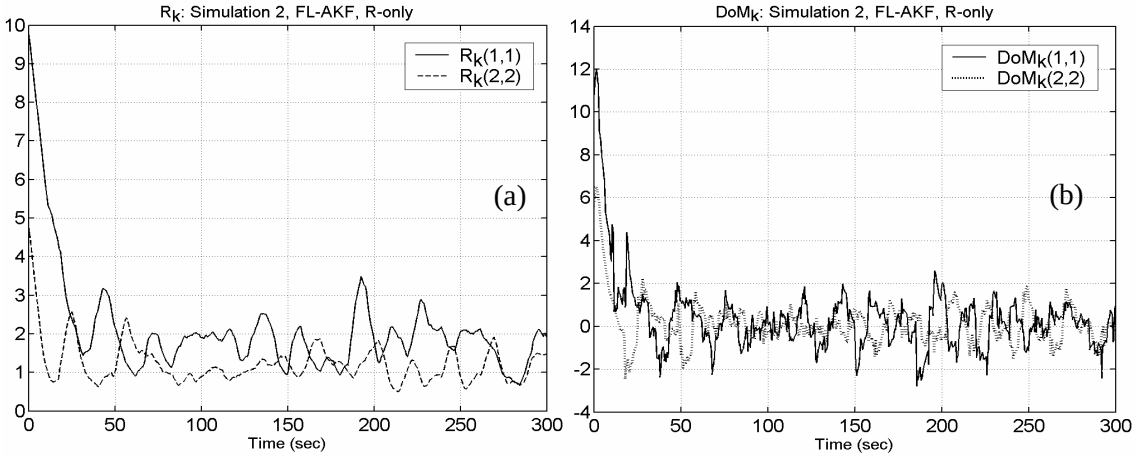


Figure 5.12(a) Estimated diagonal elements of the measurement noise covariance matrix R_k. (b) Diagonal elements of matrix DoM_k; simulation 2, FL-AKF, R-only case.

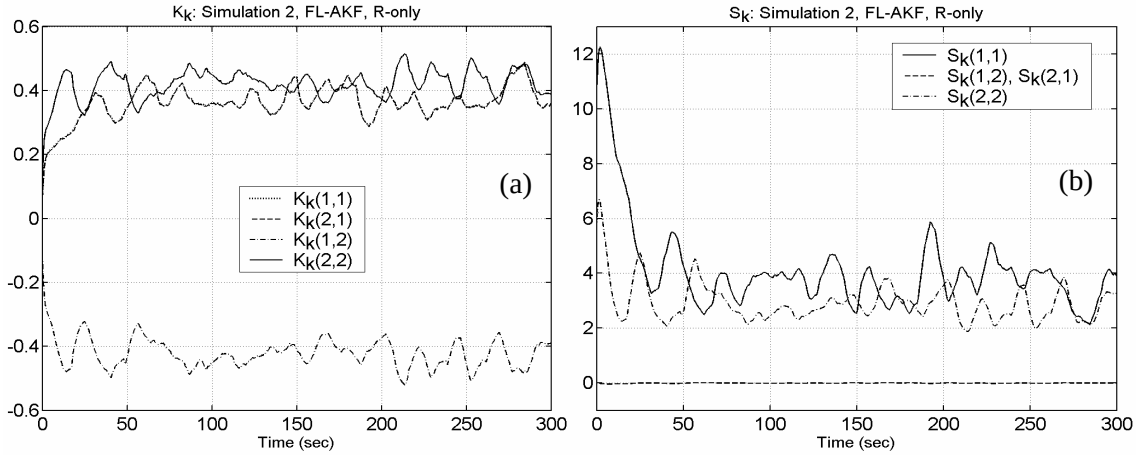


Figure 5.13(a) Elements of the Kalman gain matrix. (b) Elements of the residual covariance matrix; simulation 2, FL-AKF, R-only case.

In order to examine the adaptation performed in the FL-AKF, Q-only case, in figure 5.14(a) the estimated diagonal elements of the process noise covariance matrix Q_k are shown and they quickly converge to their actual values. In figure 5.14(b) the diagonal elements of matrix $DoMK_k$ are shown. It can be seen that these values are maintained very near to zero. In figures 5.15(a) and 5.15(b) the elements of the Kalman gain matrix and the residual covariance matrix, respectively, are shown. It can be appreciated how these values converge to values very near to their optimal.

Simulation 3: The purpose of this simulation is to investigate the performance of the developed FL-AKF under non-stationary noise profiles. The performance of the FL-AKF is compared with those of a SKF, a TAKF-IAE algorithm, and a TAKF-MMAE algorithm. The measurement noise profiles used are shown in figure 5.16; while the process noise profiles used are shown in figure 5.17.

The system under consideration was simulated for 300 sec with a sample time T=0.5 sec. The actual process and measurement noise covariance matrices are assumed as unknown. However, R_0 and Q_0 are specified as shown in Table 5.7; recall that Q and R are defined by (5.18) and (5.19), respectively. Three experiments for each filter were carried out as is detailed in Table 5.6.

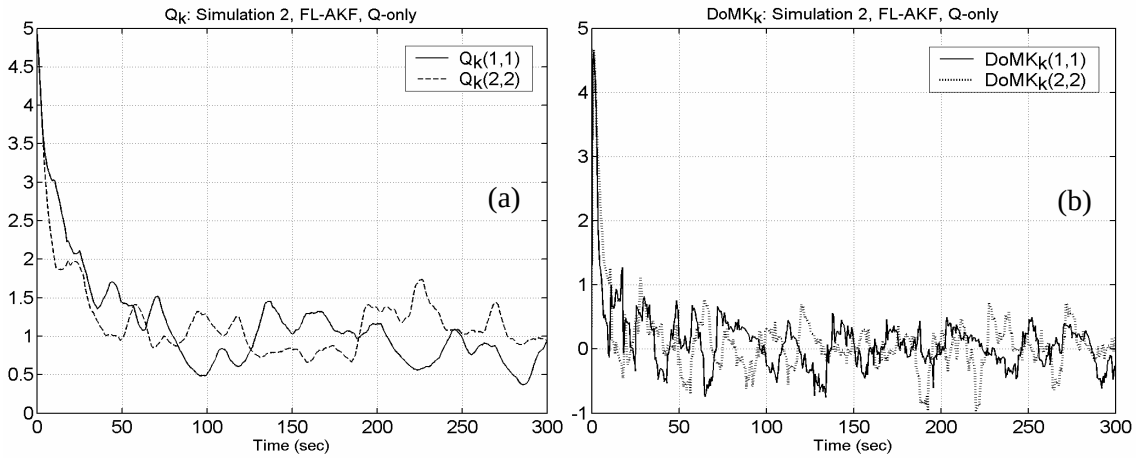


Figure 5.14(a) Estimated diagonal elements of the measurement noise covariance matrix Q_k. (b) Diagonal elements of matrix $DoMK_k$; simulation 2, FL-AKF, Q-only case.

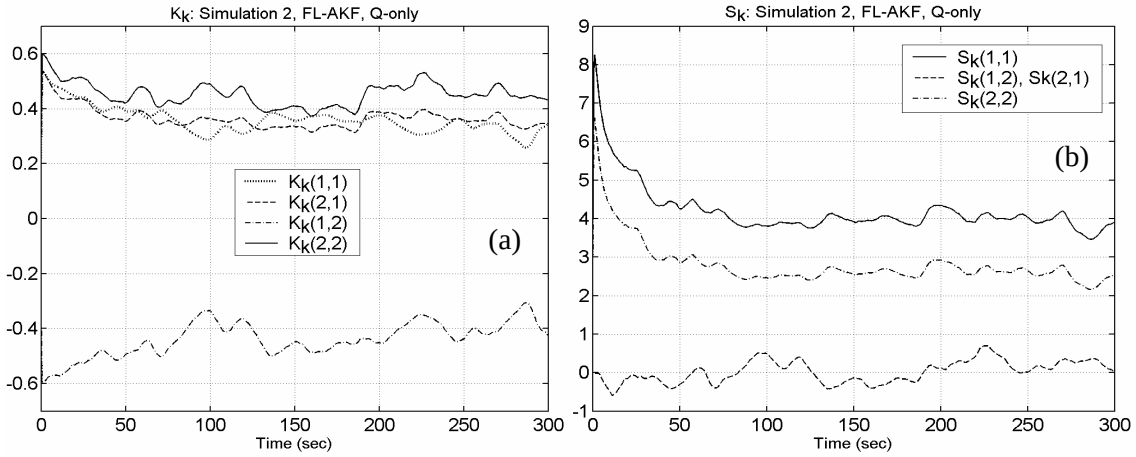


Figure 5.15(a) Elements of the Kalman gain matrix. (b) Elements of the residual covariance matrix; simulation 2, FL-AKF, Q-only case.

Table 5.6 Experiments carried out in simulation 3

Experiment	Measurement noise Profile	Process noise profile	Covariance matrices R_k	Q_k
Exp. 1	non-stationary $v_k = \begin{bmatrix} v_k^1 \\ v_k^2 \end{bmatrix}$	Stationary $Q_k = Q$	Adapted	Constant
Exp. 2	Stationary $R_k = R$	non-stationary $w_k = \begin{bmatrix} w_k^1 \\ w_k^2 \end{bmatrix}$	Constant	Adapted
Exp. 3	non-stationary $v_k = \begin{bmatrix} v_k^1 \\ v_k^2 \end{bmatrix}$	non-stationary $w_k = \begin{bmatrix} w_k^1 \\ w_k^2 \end{bmatrix}$	Adapted	Adapted

Results: Table 5.7 shows the MSE measures and TANIS values obtained for each case together with the values given to R_0 and Q_0, and the experiment performed. Note that the best results are those obtained with the proposed FL-AKF. This is thanks to the adaptation carried out, where matrices R and/or Q are adapted in such a way as to reflect, as closely as possible, the actual statistics of the noise profiles.

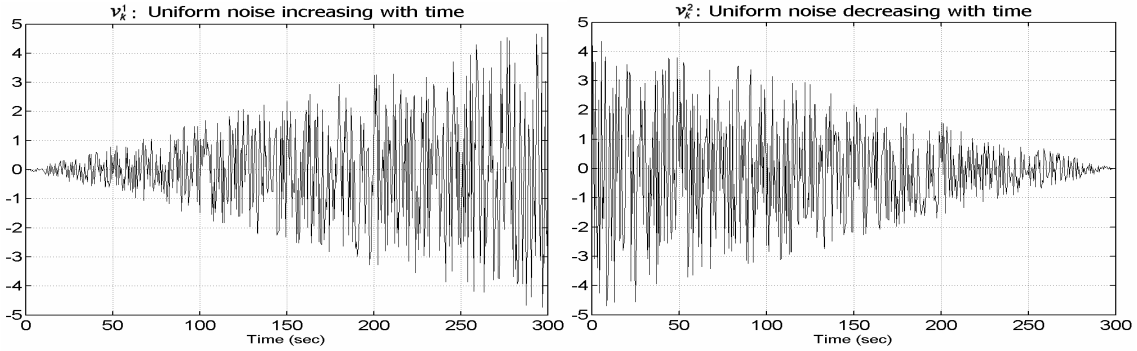


Figure 5.16 Measurement noise profiles used in simulation 3.

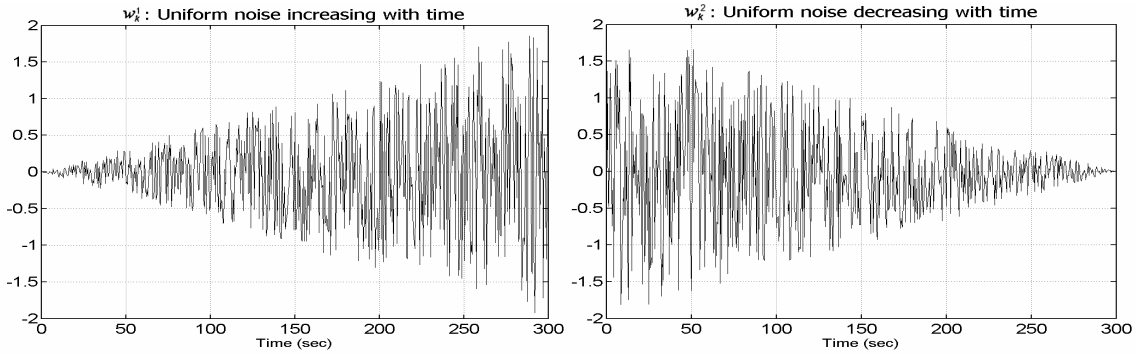


Figure 5.17 Process noise profiles used in simulation 3.

Table 5.7 Performance under non-stationary noise profiles: summary of results, simulation 3

Filter	Adaptation	MSE*x*	MSE*y*	MSE	TANIS	R_0	Q_0	Experiment
SKF	_	1.085	1.386	2.47	3.257	*R*	*Q*	Exp. 1
	_	0.6544	0.6424	1.297	1.645	*R*	*Q*	Exp. 2
	_	0.7536	1.174	1.927	2.514	*R*	*Q*	Exp. 3
FL-AKF	R-only	0.8807	1.165	2.045	2.173	*R*	*Q*	Exp. 1
	Q-only	0.5934	0.5478	1.141	1.997	*R*	*Q*	Exp. 2
	R&*Q*	0.6265	1.185	1.812	2.117	*R*	*Q*	Exp. 3
TAKF-IAE	R-only	1.12E7	1.29E7	2.4E7	3.954	*R*	*Q*	Exp. 1
	Q-only	0.7818	0.7013	1.483	2.228	*R*	*Q*	Exp. 2
	R&Q	21.97	23.62	45.59	2.148	*R*	*Q*	Exp. 3
TAKF-MMAE	SKF 1	1.26	1.816	3.076	1.048	5*R*	*Q*	Exp. 1
	SKF 2				1.516	*R*	5*Q*	
	SKF 3					0.2*R*	0.2*Q*	
	SKF 4					5*R*	5*Q*	
TAKF-MMAE	SKF 1	1.01	0.9934	2.004		5*R*	*Q*	Exp. 2
	SKF 2				0.8032	*R*	5*Q*	
	SKF 3					0.2*R*	0.2*Q*	
	SKF 4					5*R*	5*Q*	
TAKF-MMAE	SKF 1	1.015	1.896	2.911	0.7462	5*R*	*Q*	Exp. 3
	SKF 2				1.212	*R*	5*Q*	
	SKF 3					0.2*R*	0.2*Q*	
	SKF 4					5*R*	5*Q*	

In order to have a clearer picture of the adaptation being carried out, plots are presented corresponding to the R-only case under non-stationary measurement noise profiles. Figure 5.18(a) shows the diagonal estimated elements of matrix R_k and figure 5.18(b) shows the diagonal elements of matrix DoM_k. The diagonal elements of R_k are dynamically adjusted to fit as well as possible the observed measurement noise profiles. The dynamic in the noise profiles is reflected in the residual sequences as can be seen in figure 5.19 where the residual sequences are shown with their respective 2σ bounds. Finally, in figure 5.20(a) the elements of the Kalman gain matrix are shown; while in figure 5.20(b) the elements of the residual covariance matrix are shown, it can be seen how the elements of these matrices are dynamically adjusted.

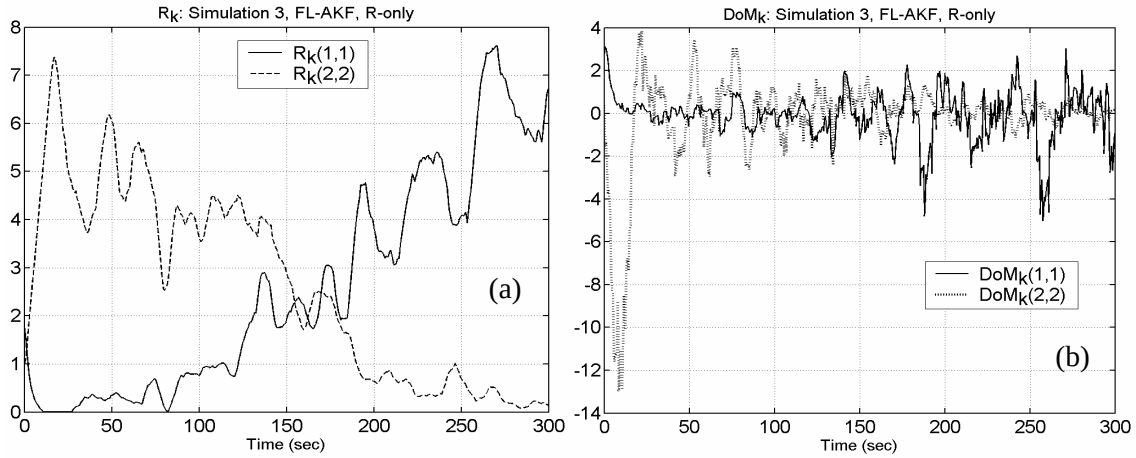


Figure 5.18(a) Diagonal estimated elements of matrix R_k. (b) Diagonal elements of matrix DoM_k; Simulation 3, R-only case, non stationary measurement noise profiles used.

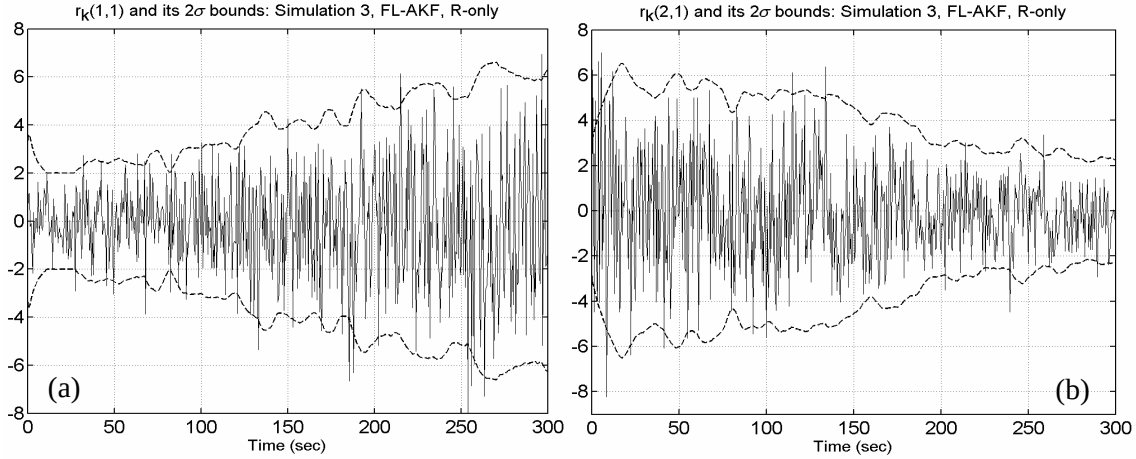


Figure 5.19 Residual sequences and their respective 2σ bounds; Simulation 3, R-only case, non stationary measurement noise profiles used.

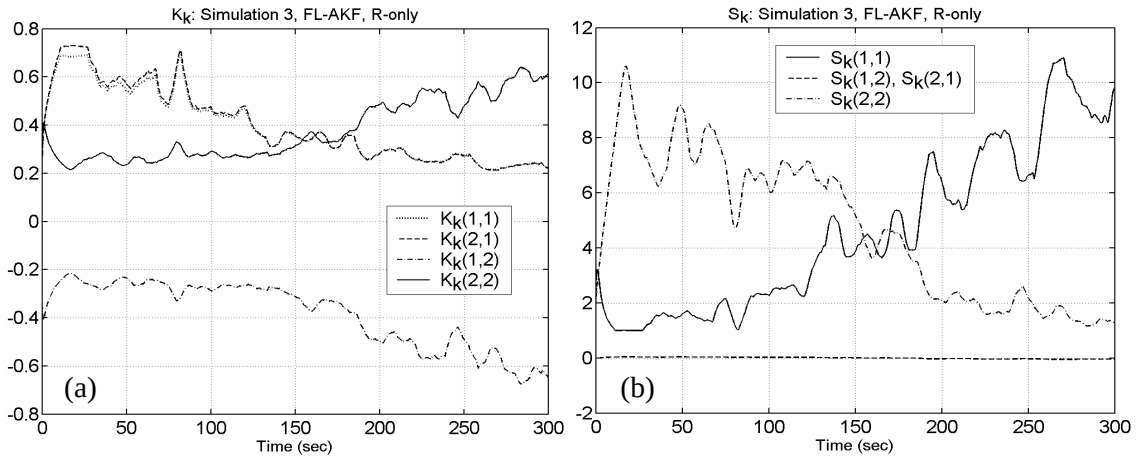


Figure 5.20(a) Elements of the Kalman gain matrix. (b) Elements of the residual covariance matrix; Simulation 3, R-only case, non stationary measurement noise profiles used.

5.4 Summary

In this chapter a fuzzy logic-based adaptive Kalman filter (FL-AKF) has been developed. The adaptation is in the sense of dynamically adjusting the measurement noise covariance matrix *R* and/or the process noise covariance matrix *Q* from data as they are obtained. This method uses the covariance-matching technique to determine if adjustments to *R* and/or *Q* are needed. An example showing the efficiency of this method has been presented. We note that superior performances were obtained with the FL-AKF than those obtained with a SKF and two different traditional adaptive Kalman filter approaches.

The role of the matrices *R* and *Q* in the SKF setting is to adjust the Kalman gain in such a way that it controls the filter bandwidth as the state and measurement errors vary. A major drawback of the SKF formulation is that at steady state its bandwidth and Kalman gain remain constant regardless of the changes in the system dynamics or the updated measurement quality. This is due to its fixed constant matrices *R* and *Q*. Conversely, the bandwidth and Kalman gain in a FL-AKF keeps changing as long as the system dynamics and statistics of the noise under which it operates change, as was particularly shown in simulation 3 in the previous section. This dynamic adaptive property of the FL-AKF is a direct result of adapting *R* and/or *Q*.

Another main characteristic of the developed FL-AKF approach is that the filter a priori statistical information is of secondary importance because it is estimated within the algorithm itself, as was shown in simulation 2. It must be remembered that the quality of these a priori noise statistics is of great importance in the SKF formulation.

The size of the sliding window over which the actual covariance of the residual is estimated has an impact on the adaptive filter performance. The smaller the window size, the faster the changes that can be captured by the FL-AKF. From numerous simulations not presented here, it was found that a good empirical value for the size of the sliding window is between 10 to 20 samples.

The numerical complexity added to the SKF in order to build a FL-AKF is marginal. From the simulations carried out it was observed that using only three simple fuzzy sets (triangular membership functions) and only three fuzzy rules for each element in the main diagonal of Q and/or R are sufficient to ensure good adaptation.

In next chapters the developed FL-AKF will be the base over which novel hybrid adaptive MSDF architectures are built. The main objective to achieve there will be that these MSDF architectures inherit the adaptive features of the proposed FL-AKF approach.

CHAPTER 6

HYBRID KALMAN FILTER-FUZZY LOGIC ADAPTIVE MULTI-SENSOR DATA FUSION ARCHITECTURES

6.1 Introduction

The Kalman filter-based MSDF architectures presented in chapter 4 require exact knowledge about the sensed environment and about the sensors. However, in real applications, only certain information is known about the sensed environment and there is no such thing as a perfect sensor. Therefore, there is scope for the development of more robust Kalman filter-based MSDF architectures. These architectures should be capable of adaptation to changes in the sensed environment and also deal with imperfect sensors.

In the MSDF literature only some approaches to adaptive MSDF are reported. From these, there are those based on the well-established Kalman filtering methods [Hong, 1991] [Zhang *et al*, 2002], and those based on recent ideas coming from soft computing technology [Kuo and Cohen, 1999] [Kobayashi *et al*, 1998]. However, little work has been done in exploring architectures that consider the combination of both these approaches. In this chapter, novel adaptive MSDF architectures, referred to as hybrid Kalman filter-fuzzy logic adaptive MSDF architectures, that combine these approaches are formulated [Escamilla and Mort, 2002, 2001a, 2001b]. The proposed architectures are designed based on the fuzzy logic-based adaptive Kalman filter developed in Chapter 5 [Escamilla and Mort, 2000, 2001c].

The general idea explored here is the combination of the advantages that both Kalman filtering and fuzzy logic techniques have. On the one hand, Kalman filtering is recognized as one of the most powerful traditional techniques of estimation: the Kalman filter provides an unbiased and optimal estimate of a state-vector in the sense of minimum error variance [Maybeck, 1979]. On the other hand, the main advantages derived from the use of fuzzy logic techniques, with respect to traditional schemes, are the simplicity of the approach, the capability of fuzzy systems to deal with imprecise information, and the possibility of including heuristic knowledge about the phenomenon under consideration [Zadeh, 1973].

In the remainder of this chapter, first a clear definition of the problem under consideration is formulated. Then the proposed hybrid adaptive MSDF architectures are described. After that, the effectiveness of the proposed MSDF approaches is demonstrated through exhaustive simulation of an illustrative example. In this study, the fault-tolerant performance of the proposed MSDF approaches is also investigated. A final discussion and a summary are given to conclude the chapter.

6.2 Problem formulation

Assume that a discrete-time process can be modelled by,

$$x_{k+1} = \Phi_k x_k + B_k u_k + w_k \tag{6.1}$$

$$z_{ik} = H_{ik} x_k + v_{ik}\,,\ i = 1,\ldots,N \tag{6.2}$$

where $x_k \in \Re^n$ is a state vector at instant of time denoted by the subscript k, $\Phi_k \in \Re^{n\times n}$ is a state transition matrix, $B_k \in \Re^{n\times l}$ relates the control input $u_k \in \Re^l$ to the state vector x_k, $w_k \in \Re^n$ represents the modelling errors characterised by the error covariance matrix Q_k. There are N measurement models described by (6.2), each of which corresponds to a local measurement. Thus, the local vector $z_{ik} \in \Re^m$ describes the measurement made by sensor i at instant of time k. $H_{ik} \in \Re^{m\times n}$ is the i-th measurement sensitivity matrix. The noise (or error) in each measurement is represented by the vector v_{ik} and specified by matrix R_{ik}. In other words, the measurement noise covariance matrix R_{ik} reflects the precision of the i-th sensor. w_k and v_{ik} are modelled as uncorrelated zero-mean Gaussian noise sequences with covariance matrices Q_k and R_{ik} satisfying:

$$E\{w_k\} = 0 \quad \text{for all } k, \tag{6.3}$$

$$E\{v_{ik}\} = 0 \quad \text{for all } k, \tag{6.4}$$

$$E\{w_k w_j^T\} = \begin{cases} Q_k, & j = k \\ 0 & j \neq k \end{cases} \tag{6.5}$$

$$E\{v_{ik} v_{ij}^T\} = \begin{cases} R_{ik}, & j = k \\ 0 & j \neq k \end{cases} \tag{6.6}$$

where $E\{\cdot\}$ is the statistical expectation operator.

It is assumed that the known information about the sensed environment and sensors is captured in the known matrices Φ_k, H_{ik}, and Q_k, while the unknown matrix R_{ik} models the uncertain and inaccurate information about the sensed environment and sensors. Hence, the objective of this chapter is to develop adaptive MSDF architectures capable of obtaining a fused estimated state vector $\hat{x}_k$ that determines the parameters being measured as precisely as possible by combining the information coming from N imperfect sensors. By adaptive we mean that the MSDF process is capable of adjusting on-line the unknown matrices R_{ik} to fit, as closely as possible, the actual statistics of the noise profiles present in the measured data. By an imperfect sensor we mean that the noise profile present in it has uncertain statistics and these statistics are not necessarily stationary. In addition, sensors can be subject to transient and persistent failures.

6.3 Hybrid adaptive MSDF architectures

In the following sections, four hybrid adaptive MSDF architectures are proposed. These architectures are referred to as: fuzzy logic-based adaptive Kalman filter with fuzzy logic performance assessment scheme (FL-AKF-FLA), fuzzy logic-based adaptive centralised Kalman filter (FL-ACKF), fuzzy logic-based adaptive decentralised Kalman filter (FL-ADKF), and fuzzy logic-based adaptive federated Kalman filter (FL-AFKF). These architectures are designed based on the fuzzy logic-based adaptive Kalman filter developed in Chapter 5 [Escamilla and Mort, 2000, 2001c].

6.3.1 Hybrid architecture FL-AKF-FLA

In this section a novel hybrid MSDF architecture combining the FL-AKF developed in Chapter 5 and a fuzzy logic performance assessment scheme is presented. Figure 6.1 shows a schematic representation of the proposed MSDF architecture. The objective of the proposed architecture, referred to as FL-AKF-FLA, is to combine the measurement-vectors coming from N disparate

sensors, each one with different measurement dynamics and noise characteristics, to obtain a fused state-vector estimate that better reflects the actual value of the parameters being measured. To reach this objective, first each measurement-vector coming from each sensor is fed to a FL-AKF. Second, a subsystem called a fuzzy logic assessor (FLA) is monitoring and assessing the performance of each FL-AKF. Thus, there are *N* sensors, *N* FL-AKFs, and *N* FLAs working in parallel as is represented in Figure 6.1 (the time-step subscript is not indicated for simplicity). The task of each FL-AKF is to obtain a state-vector estimate based on the measurement-vector coming from its own sensor. While the task of each FLA is to assess the performance of its corresponding FL-AKF through assigning to it a degree of confidence factor. Finally, the fused state-vector estimate is obtained using a weighting average scheme based on the assigned degree of confidence factors (see figure 6.1). Each FL-AKF is constructed as was specified in chapter 4. The FLA subsystem and the weighted average fusion scheme are described as follows, in this description the subscript indicating the time-step has been omitted for simplicity.

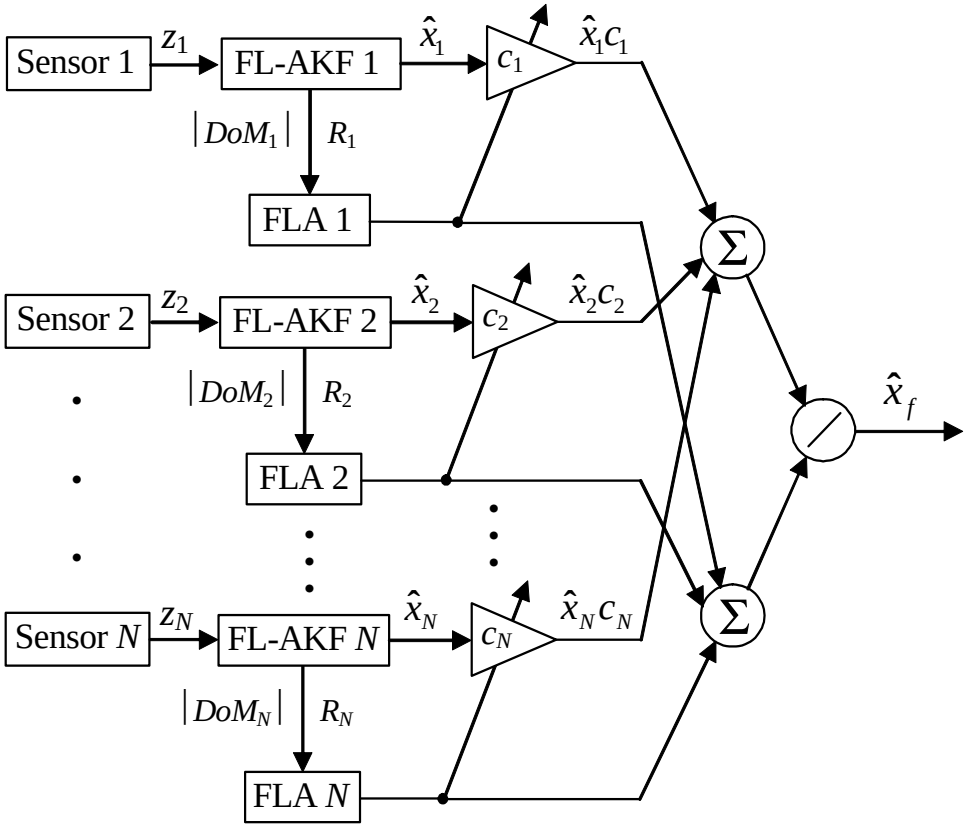


Figure 6.1 Proposed hybrid FL-AKF-FLA MSDF architecture (the time-step subscript is not indicated for simplicity).

Each FLA subsystem assigns a degree of confidence factor, or simply confidence factor, denoted as c_j, to its corresponding FL-AKF state-vector estimate $\hat{x}_j$, j=1,…,N. The degree of confidence is calculated based on the current values of the absolute value of the degree of mismatch $|DoM_j|$, which is a measure of the size of discrepancy between the theoretical value of the residual covariance matrix S_j and its estimated actual value $\hat{C}r_j$, and the adjusted measurement noise covariance matrix R_j. First, the elements of a vector of confidence values $[c_j^i \ldots c_j^m]^T$ is calculated in a recursive way by a two-inputs-one-output FIS, as is graphically represented in figure 6.2. The *i-th* degree of confidence c_j^i, a number on the interval [0, 1], is an indicator of the level of performance of the FL-AKF. This performance is evaluated through two measures, the level of consistency between the theoretical and actual residual covariance

(indicated by matrix $|DoM_j|$), and the actual level of measurement noise present in the corresponding sensor (indicated by matrix R_j). Note that both matrices $|DoM_j|$ and R_j have the same dimension $m \times m$, where m is the dimension of the measurement vector.

The FIS used in each FLA has as inputs the (i,i) elements in $|DoM_j|$ and R_j, $i=1,\ldots,m$; and as output the *i-th* degree of confidence c_j^i (see figure 6.2). The membership functions for $|DoM_j|$ and R_j are shown in figure 6.3. There the fuzzy labels mean: ZE = Zero, S = Small, and L = Large. For the output c_j^i, three fuzzy singletons are defined with the labels: G=1=Good, AV=0.5=Average, and P=0=Poor. The parameters g and h that define the fuzzy sets in the input linguistic variables can be specified in accordance with the application under consideration.

Nine rules complete the fuzzy rule base of the FIS as is given in Table 1, which is known as a decision table. Each cell in the decision table indicates the rule consequent corresponding to the rule antecedent; this last term is defined by the intersection of the linguistic values (fuzzy sets) of the FIS inputs. For example, rule 1 is defined as "if $|DoM_j|$ is ZE and R_j is ZE then c_j^i is G". The fuzzy rules are based on two simple heuristic considerations. First, if the current value of $|DoM|$ is near to zero and the current value of R is near to zero, then it means there is consistency between the theoretical and actual residual covariance, and a low level of noise is present in the sensor. Therefore, the FL-AKF filter is working almost perfectly, in consequence a degree of confidence near the maximum 1 should be assigned to it. Second, if one or both of these values increases far from zero, it means that the filter performance is degrading; thus the degree of confidence assigned by the FIS is decreased accordingly down to the minimum 0. Thus, using the compositional rule of inference sum-prod and the centre of area defuzzification method, the FIS obtains each c_j^i degree of confidence value.

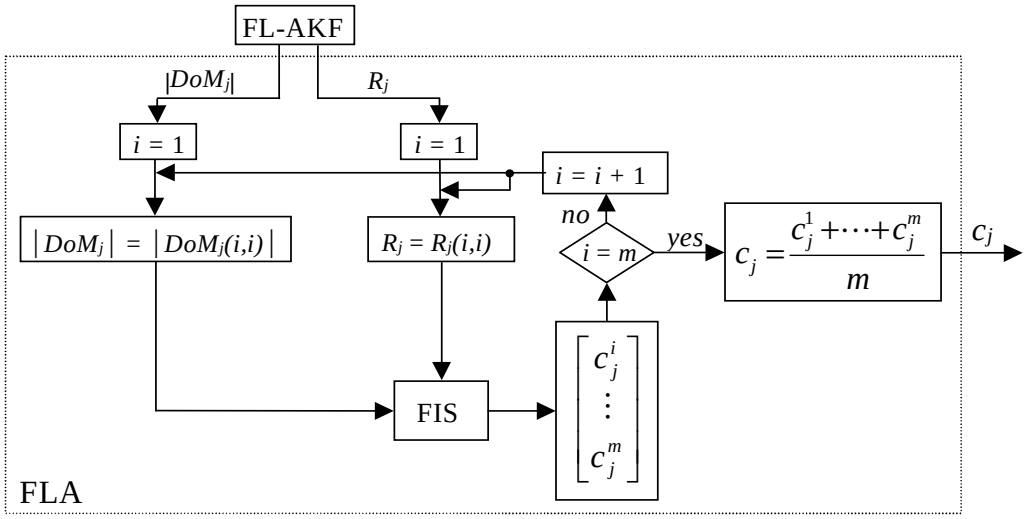


Figure 6.2 Process of calculating the degree of confidence values.

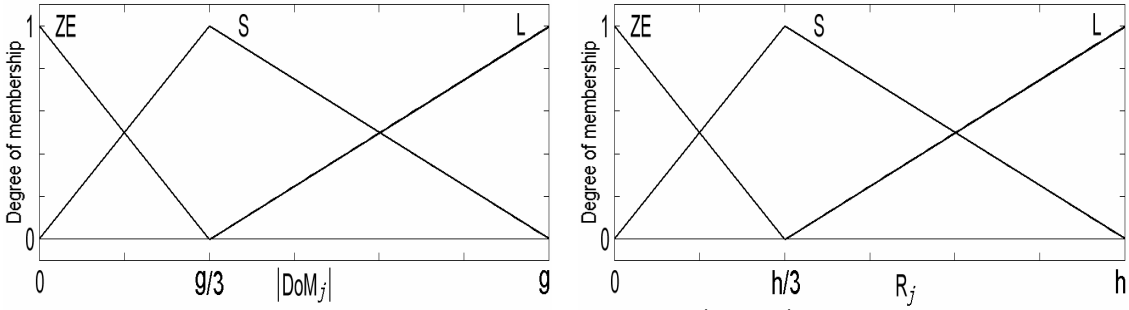


Figure 6.3 Membership functions for $|DoM_j|$ and R_j.

Once all the elements c_j^i have been obtained an averaged degree of confidence, which represents the averaged performance of the corresponding FL-AKF is calculated by,

$$c_j = \frac{c_j^1 + \cdots + c_j^m}{m} \tag{6.7}.$$

The degree of confidence c_j can be interpreted as a measure of the reliability of the *j*-th FL-AKF. A higher degree of confidence means we have a higher confidence in that the estimated state-vector represents the actual value of the parameters being measured. At the same time, the degree of confidence acts as a weighting factor used for fusion purposes.

Finally, the fused state-vector estimate is obtained using a weighted average scheme based on the assigned confidence factor values. This is,

$$\hat{x}_f = \frac{\hat{x}_1 c_1 + \cdots + \hat{x}_N c_N}{c_1 + \cdots + c_N} = \bar{c}_1 \hat{x}_1 + \cdots + \bar{c}_N \hat{x}_N \tag{6.8}$$

with

$$\bar{c}_j = \frac{c_j}{\sum_{j=1}^{N} c_j} \quad , \quad j=1,\ldots,N \tag{6.9}.$$

Table 1 Decision table of the FIS used in each FLA

$\lvert DoM_j \rvert$ \ R_j	ZE	S	L
ZE	G	G	AV
S	G	AV	P
L	AV	P	P

In order to prevent possible conflicts, one modification is incorporated. If the sum of all the degrees of confidence is equal to zero: $\sum_{j=1}^{N} c_j = 0$, then the fused output is simply the average of the *N* state-vector estimates:

$$\hat{x}_f = \frac{\hat{x}_1 + \cdots + \hat{x}_N}{N} \tag{6.10}.$$

6.3.2 Fuzzy logic-based adaptive centralised Kalman filter

The standard CKF algorithm was described in section 4.3.1 and a summary of it is given in Table 6.2. In this section the idea used in the FL-AKF, developed in the previous chapter, is extended to the CKF structure to develop a fuzzy logic-based adaptive CKF (from here referred

to as FL-ACKF) for MSDF purposes. Here the adaptation is in the sense of dynamically tuning the global measurement noise covariance matrix R_{gk}. The structure of the proposed FL-ACKF is shown in figure 6.4.

By analysing the algorithm of the CKF (see Table 6.2) it is evident that the same idea used in the FL-AKF of having a fuzzy logic-based adaptation scheme (see figure 6.5) is directly applicable to developing a FL-ACKF. First, the global residual sequence r_{gk} is defined as,

$$r_{gk} = z_{gk} - H_{gk}\hat{x}_{k(-)} \tag{6.11},$$

which has dimension $(mN)\times 1$; where m is the dimension of the measurement vector, N is the number of sensors (equal to the number of measurement vectors), subscript k indicates the instant of time, and subscript g is used to mark the global condition of the parameters.

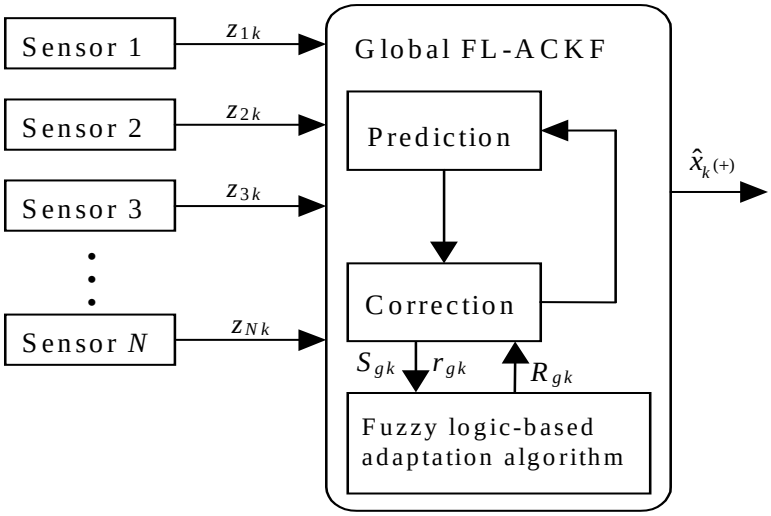


Figure 6.4 Fuzzy logic-based Adaptive Centralised Kalman Filter (FL-ACKF).

Thus, by using the global measurement sensitivity matrix H_{gk}, and the global measurement noise covariance matrix R_{gk}, the global theoretical residual covariance matrix is,

$$S_{gk} = H_{gk}P_{k(-)}H_{gk}^{T} + R_{gk} \tag{6.12}.$$

The actual global residual covariance can be estimated through averaging inside a sliding window,

$$\hat{C}r_{gk} = \frac{1}{WS}\sum_{i=i_0}^{k} r_{gi}r_{gi}^{T} \tag{6.13}$$

where $i_0 = k - WS + 1$ is the first sample inside the estimation window, WS is the window size, which is chosen empirically to give statistical smoothing, and k is the sample-time step.

Therefore, a global Degree of Mismatch (DoM_{gk}), indicative of the size of the discrepancy between the global theoretical residual covariance S_{gk} and its estimated actual value $\hat{C}r_{gk}$, can be defined as,

$$DoM_{gk} = S_{gk} - \hat{C}r_{gk} \tag{6.14}$$

Note that all matrices S_{gk}, $\hat{C}r_{gk}$, DoM_{gk} and R_{gk} have the same dimension $(mN)\times(mN)$. Because of this, the elements in the main diagonal of DoM_{gk} can be used to adjust the elements in the main diagonal of R_{gk}. Remember that under the assumption of having uncorrelated and Gaussian measurement noise sequences, R_{gk} is a diagonal matrix whose elements are the variances of the individual components of the measurement noise vectors $v_{1k},\ldots,v_{Nk}$. Therefore, from (6.6) it is deduced that by increasing the elements in the main diagonal of R_{gk} the elements in the main diagonal of S_{gk} are increased, and vice versa. With this, the size of the mismatch (DoM_{gk}) between the theoretical and actual value of the global residual covariance can be regulated through changing the value of the diagonal elements of R_{gk}. As a result, the filter consistency is maintained. Furthermore, R_{gk} is adjusted to fit the actual statistics of the noise profiles present in the sensors.

Table 6.2 Summary of the standard
Centralised Kalman Filter (CKF) algorithm

Process model	Global KF
State space model: $x_{(k+1)} = \Phi_k x_k + B_k u_k + w_k$ *Measurement equation:* $z_{ik} = H_{ik} x_k + v_{ik}$ $i = 1,\ldots,N$ N is the number of sensors	*Global model:* $z_{gk} = [z_{1k}^T \ldots z_{Nk}^T]^T$ $H_{gk} = [H_{1k}^T \ldots H_{Nk}^T]^T$ $R_{gk} = block\ diag[R_{1k} \ldots R_{Nk}]$ *Prediction equations:* $\hat{x}_{(k+1)(-)} = \Phi_k \hat{x}_{k(+)} + B_k u_k$ $P_{(k+1)(-)} = \Phi_k P_{k(+)} \Phi_k^T + Q_k$ *Correction equations:* $K_k = P_{k(-)} H_{gk}^T [H_{gk} P_{k(-)} H_{gk}^T + R_{gk}]^{-1}$ $\hat{x}_{k(+)} = \hat{x}_{k(-)} + K_k [z_{gk} - H_{gk} \hat{x}_{k(-)}]$ $P_{k(+)} = [I - K_k H_{gk}] P_{k(-)}$

Therefore, mN SISO FISs, each one using three general rules of adaptation, can be implemented to adjust the diagonal elements of R_{gk}. The three general rules of adaptation are:

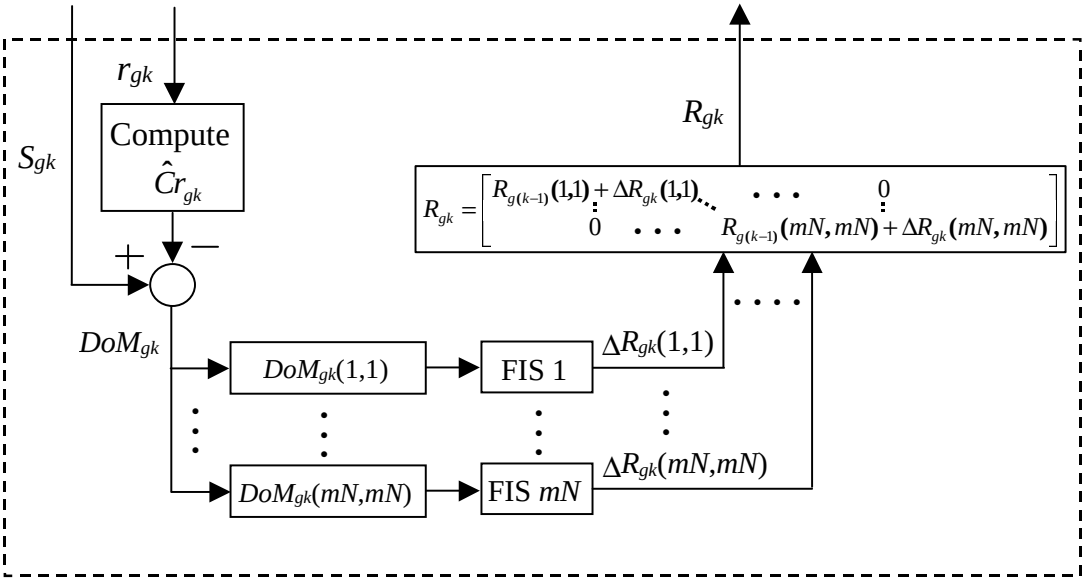
1. If $DoM_{gk}(i,i) \cong 0$ (this means $S_{gk}(i,i)$ and $\hat{C}r_{gk}(i,i)$ match almost perfectly) then maintain $R_{gk}(i,i)$ unchanged.
2. If $DoM_{gk}(i,i) > 0$ (this means $S_{gk}(i,i)$ is greater than its actual value $\hat{C}r_{gk}(i,i)$) then decrease $R_k(i,i)$.
3. If $DoM_{gk}(i,i) < 0$ (this means $S_{gk}(i,i)$ is smaller than its actual value $\hat{C}r_{gk}(i,i)$) then increase $R_{gk}(i,i)$.

where $S_{gk}(i,i)$, $\hat{C}r_{gk}(i,i)$, $R_k(i,i)$ and $DoM_{gk}(i,i)$, i=1,…,mN, are the elements in the main diagonal of S_{gk}, $\hat{C}r_{gk}$, R_{gk} and DoM_{gk}, respectively. Finally, R_{gk} is adjusted in this way:

$$R_{gk}(i,i) = R_{g(k-1)}(i,i) + \Delta R_{gk}(i,i); \qquad i=1,\ldots,mN \tag{6.15}$$

where $\Delta R_{gk}(i,i)$ is the tuning or correction factor for $R_{gk}(i,i)$. Each correction factor is the output of a different FIS, while each of the elements in the main diagonal of DoM_{gk} is the input to the corresponding FIS. A graphical representation of this adjusting process is shown in figure 6.5.

Each one of the FISs used in the fuzzy logic-based adaptation algorithm is constructed as was explained in section 5.3.2.a; thus its development is not repeated here and the reader is referred to that section. It is necessary to remark that inside the fuzzy logic-based adaptation algorithm, shown in figure 6.5, *mN* parallel SISO FISs, with three fuzzy rules each one, are considered. However, a sequential or a MIMO FIS implementation can replace the parallel implementation. In the sequential implementation a single SISO FIS is executed *mN* times until all the adjusting factors for R_{gk} are calculated. While in the MIMO FIS implementation a single FIS with 3*mN* rules is implemented to obtain at once all the adjusting factors. A broader explanation of the sequential and MIMO implementations can be found in section 5.3.2.a.



6.5 Fuzzy logic-based adaptation scheme.

6.3.3 Fuzzy logic-based adaptive decentralised Kalman filter

A summary of the standard DKF algorithm is presented in Table 6.3. A broader description of it is given in section 4.3.2. In this section a fuzzy logic-based adaptive DKF is proposed (from here referred to as FL-ADKF). The FL-ADKF is based on the FL-AKF developed in chapter 5.

The structure of the proposed FL-ADKF is shown in figure 6.6. As can be seen, this architecture is similar to the standard DKF architecture, but, in this case, instead of having *N* local SKFs, there are *N* local FL-AKFs working in parallel. Each one of the FL-AKFs is built as was described in Chapter 5. However, here solely the case where only the measurement noise covariance matrix *R* is adjusted is implemented. From Table 6.3 it can be noted that the standard DKF algorithm can work without any alteration when FL-AKFs are used as local filters. The difference is that instead of having constant matrices R_{ik}, they are dynamically adjusted to fit the actual statistics of the noise profiles present in the sensors. The master filter or fusion algorithm is applied directly using the information coming from the local FL-AKFs. This makes the whole structure adaptive.

Table 6.3 Summary of the standard Decentralised Kalman Filter (DKF) algorithm

Process model	Local KFs	Master filter
State space model: $x_{i(k+1)} = \Phi_{ik} x_{ik} + B_{ik} u_{ik} + w_{ik}$ *Measurement equation:* $z_{ik} = H_{ik} x_{ik} + v_{ik}$ $i = 1,\ldots,N$	*Prediction equations:* $\hat{x}_{i(k+1)}(-) = \Phi_{ik} \hat{x}_{ik}(+) + B_{ik} u_{ik}$ $P_{i(k+1)}(-) = \Phi_{ik} P_{ik}(+) \Phi_{ik}^T + Q_{ik}$ and invert to get $P_{i(k+1)}^{-1}(-)$ $i = 1,\ldots,N$ *Correction equations:* $K_{ik} = P_{ik}(-) H_{ik}^T [H_{ik} P_{ik}(-) H_{ik}^T + R_{ik}]^{-1}$ $\hat{x}_{ik}(+) = \hat{x}_{ik}(-) + K_{ik}[z_{ik} - H_{ik} \hat{x}_{ik}(-)]$ $P_{ik}(+) = [I - K_{ik} H_{ik}] P_{ik}(-)$ and invert to get $P_{ik}^{-1}(+)$ $i = 1,\ldots,N$	*Prediction equations:* $\hat{x}_{(k+1)}(-) = \Phi_k \hat{x}_k(+) + B_k u_k$ $P_{(k+1)}(-) = \Phi_k P_k(+) \Phi_k^T + Q_k$ and invert to get $P_{k+1}^{-1}(-)$ *Correction equations:* $P_k^{-1}(+) = P_k^{-1}(-) + \sum_{i=1}^{N} P_{ik}^{-1}(+) - \sum_{i=1}^{N} P_{ik}^{-1}(-)$ and invert to get $P_k(+)$ $\hat{x}_k(+) = P_k(+) \left[P_k^{-1}(-) \hat{x}_k(-) + \sum_{i=1}^{N} P_{ik}^{-1}(+) \hat{x}_{ik}(+) - \sum_{i=1}^{N} P_{ik}^{-1}(-) \hat{x}_{ik}(-) \right]$

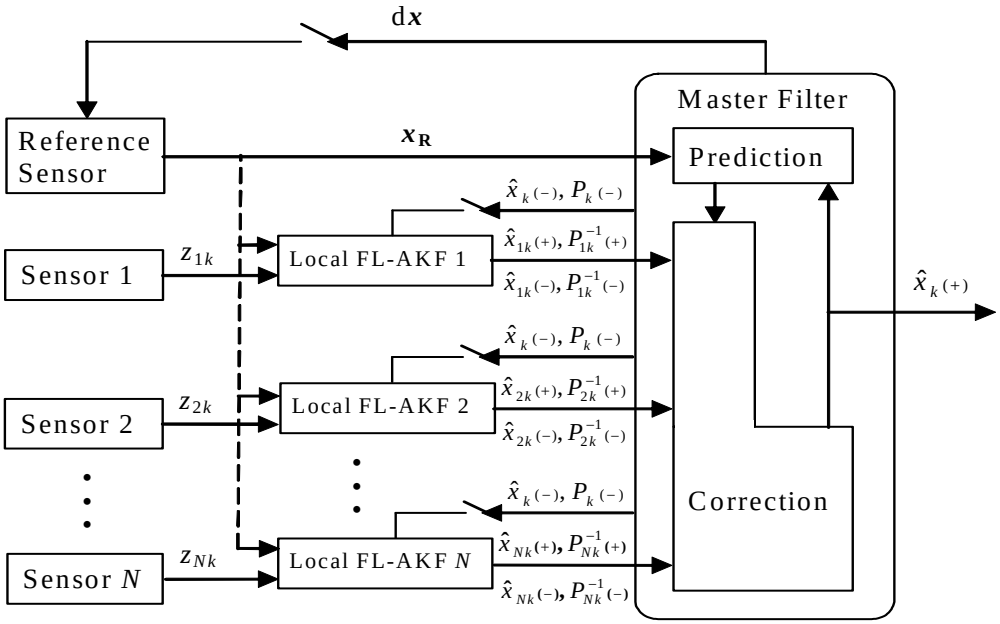


Figure 6.6 Fuzzy logic-based Adaptive Decentralised Kalman Filter (FL-ADKF).

6.3.4 Fuzzy logic-based adaptive federated Kalman filter

The standard FKF algorithm was described in section 4.3.3. A summary of it is given in Table 6.4. From an analysis of the FKF algorithm it can be deduced that the FL-AKF can be used to build a fuzzy logic-based adaptive FKF (from here referred to as FL-AFKF) algorithm.

The structure of the FL-AFKF is shown in figure 6.7. Like the FL-ADKF, the FL-AFKF is designed by substituting the local standard KFs with FL-AKFs, as can be seen in figure 6.7. However, in this case due to the use of the information sharing principle this substitution can not be directly applied. Observe in Table 6.4 that the common process noise covariance matrix Q_k and the fused error covariance matrix $P_{fk}(+)$ are affected by the information sharing factor

($1/\beta_i$) before being used in the local and master filters' prediction equations. In consequence, the local theoretical residual covariance S_{ik} does not represent the information corresponding to local filers, but reflects the global shared information. Therefore, these values cannot be used to determine the local degree of mismatch values (DoM_{ik}) for purposes of adaptation of local R_{ik} matrices. Note that the local residual sequence values ($r_{ik} = z_{ik} - H_{ik}\hat{x}_{ik(-)}$) are not affected by the information sharing factor and, consequently, neither the $\hat{C}r_{ik}$ values. Thus, what is needed is to obtain local theoretical residual covariance matrices representing the information corresponding to local filters only.

A solution to the above problem can be formulated as follows. First, from the standard FKF algorithm, the covariance residual matrix of the *i*-th filter is obtained by,

$$S_{ik} = H_{ik}P_{ik(-)}H_{ik}^T + R_{ik} \tag{6.16}$$

with

$$P_{i(k+1)(-)} = \Phi_{ik}P_{ik(+)}\Phi_{ik}^T + Q_{ik} \tag{6.17a}$$

$$P_{ik(-)} = \Phi_{i(k-1)}P_{i(k-1)(+)}\Phi_{i(k-1)}^T + Q_{i(k-1)} \tag{6.17b},$$

but by using the information sharing factor it is known that $P_{ik(+)} = (1/\beta_i)P_{fk(+)}$ and $Q_{ik} = (1/\beta_i)Q_k$, thus (6.16) transforms to,

Table 6.4 Summary of the standard Federated Kalman Filter (FKF) algorithm

Process model	Local KFs	Master filter
State space model: $x_{i(k+1)} = \Phi_{ik}x_{ik} + B_{ik}u_{ik} + w_{ik}$ *Measurement equation:* $z_{ik} = H_{ik}x_{ik} + v_{ik}$ $i = 1,\ldots,N$	*Divide global information:* $Q_{ik} = (1/\beta_i)Q_k$ $P_{ik(+)} = (1/\beta_i)P_{fk(+)}$ $\hat{x}_{ik(+)} = \hat{x}_{fk(+)}$ $i = 1,\ldots,N,M$ *Subject to*: $\sum_{i=1}^{N,M}\beta_i = 1$ *Prediction equations:* $\hat{x}_{i(k+1)(-)} = \Phi_{ik}\hat{x}_{ik(+)} + B_{ik}u_{ik}$ $P_{i(k+1)(-)} = \Phi_{ik}P_{ik(+)}\Phi_{ik}^T + Q_{ik}$ $i = 1,\ldots,N$ *Correction equations:* $K_{ik} = P_{ik(-)}H_{ik}^T[H_{ik}P_{ik(-)}H_{ik}^T + R_{ik}]^{-1}$ $\hat{x}_{ik(+)} = \hat{x}_{ik(-)} + K_{ik}[z_{ik} - H_{ik}\hat{x}_{ik(-)}]$ $P_{ik(+)} = [I - K_{ik}H_{ik}]P_{ik(-)}$ and invert to get $P_{ik}^{-1}(+)$ $i = 1,\ldots,N$	*Prediction equations:* $\hat{x}_{M(k+1)(-)} = \Phi_{Mk}\hat{x}_{Mk(+)} + B_{Mk}u_{Mk}$ $P_{M(k+1)(-)} = \Phi_{Mk}P_{Mk(+)}\Phi_{Mk}^T + Q_{Mk}$ and invert to get $P_{M(k+1)(-)}^{-1}$ *Fusion equations:* $P_{fk}^{-1}(+) = \sum_{i=1}^{N}P_{ik}^{-1}(+) + P_{Mk}^{-1}(-)$ and invert to get $P_{fk(+)}$ $\hat{x}_{fk(+)} = P_{fk(+)}\left[P_{Mk}^{-1}(-)\hat{x}_{Mk(-)} + \sum_{i=1}^{N}P_{ik}^{-1}(+)\hat{x}_{ik(+)}\right]$

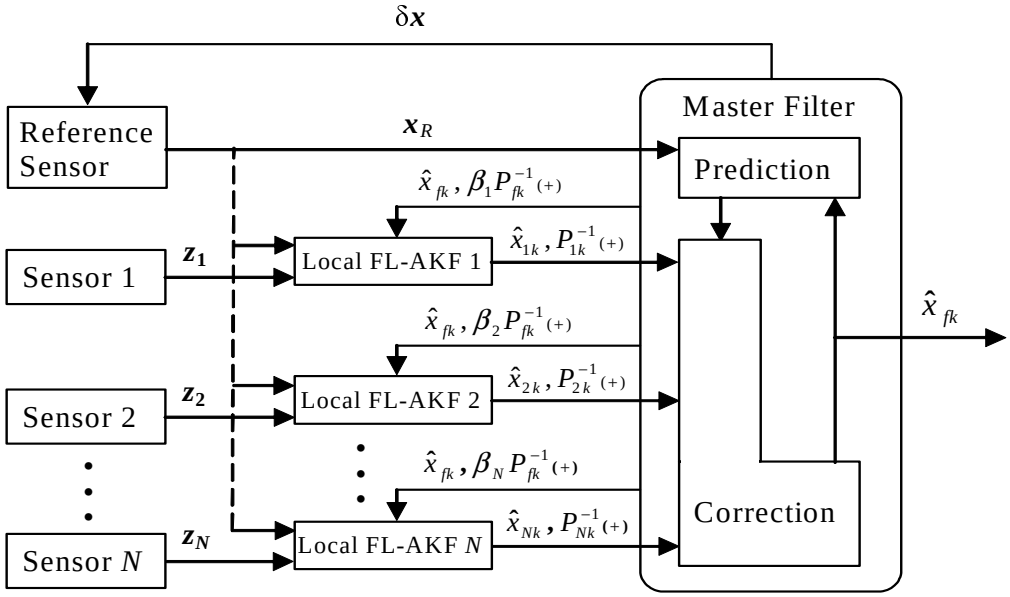


Figure 6.7 Fuzzy logic-based Adaptive Federated Kalman Filter (FL-AFKF).

$$\begin{aligned} S_{ik} &= H_{ik}[\Phi_{i(k-1)}(1/\beta_i)P_{f(k-1)(+)}\Phi^T_{i(k-1)} + (1/\beta_i)Q_{k-1}]H^T_{ik} + R_{ik} \\ &= H_{ik}(1/\beta_i)[\Phi_{i(k-1)}P_{f(k-1)(+)}\Phi^T_{i(k-1)} + Q_{k-1}]H^T_{ik} + R_{ik} \\ &= (1/\beta_i)\{H_{ik}[\Phi_{i(k-1)}P_{f(k-1)(+)}\Phi^T_{i(k-1)} + Q_{k-1}]H^T_{ik}\} + R_{ik} \\ &= (1/\beta_i)[H_{ik}P_{fk(-)}H^T_{ik}] + R_{ik} \end{aligned} \quad (6.18).$$

Equation (6.18) means that each local filter uses a fraction ($1/\beta_i$) of the factor $[H_{ik}P_{fk(-)}H^T_{ik}]$ to calculate S_{ik}. Thus, by compensating this with the factor (β_i) its effect in S_{ik} is removed, this is:

$$\begin{aligned} S^*_{ik} &= \beta_i(1/\beta_i)[H_{ik}P_{fk(-)}H^T_{ik}] + R_{ik} \\ &= H_{ik}P_{ik(-)}H^T_{ik} + R_{ik} \end{aligned} \quad (6.19),$$

where S^*_{ik} is the value used to calculate the local degree of mismatch values:

$$DoM_{ik} = S^*_{ik} - \hat{C}r_{ik} \quad (6.20),$$

and $\hat{C}r_{ik}$ is calculated using the local residual values:

$$\hat{C}r_{ik} = \frac{1}{N}\sum_{j=j_0}^{k} r_{ik}r^T_{ik} \quad (6.21),$$

where $j_0 = k - N + 1$ is the first sample inside the estimation window. Therefore, by using a fuzzy logic-based adaptation algorithm the *i*-th local measurement noise covariance matrix R_{ik} is dynamically adjusted to fit the statistics of the actual measured data. Each adaptation algorithm

is implemented as is specified in section 5.3.2.a. A graphical representation of the calculation of a local matrix S_{ik}^{*} is shown in figure 6.8.

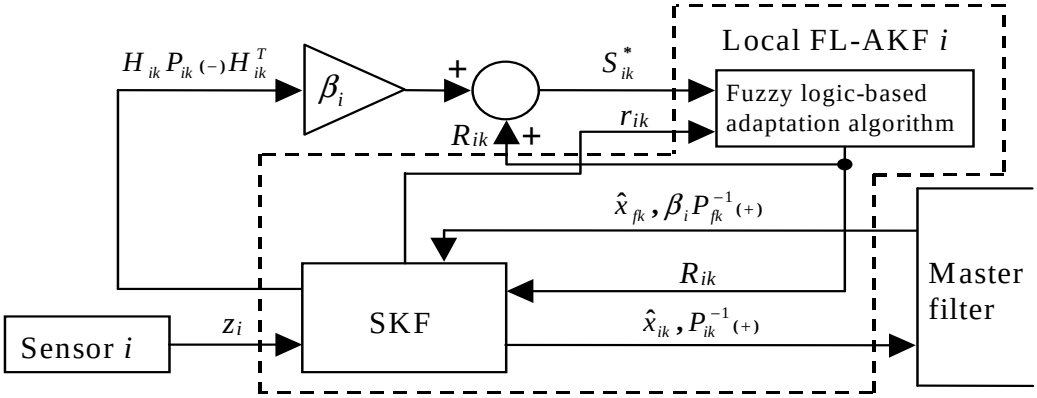


Figure 6.8 Graphical representation of the calculation of a local matrix S_{ik}^{*}.

6.4 Illustrative example

In this section an illustrative example with four noisy sensors is outlined to demonstrate the effectiveness and accuracy of the proposed hybrid Kalman filter-fuzzy logic adaptive MSDF architectures. Exhaustive simulations have been carried out and results are presented in this section. The experiments where developed under the MATLAB/SIMULINK simulation environment (a resume of the different implemented simulation models is given in Appendix B).

Consider the following linear system, which corresponds to a vehicle moving in one-dimensional co-ordinate space [Gao *et al*, 1993] [Chaer *et al*, 1997]:

$$x_{k+1} = \begin{bmatrix} p_{k+1} \\ s_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_k \\ s_k \end{bmatrix} + \begin{bmatrix} w_k^1 \\ w_k^2 \end{bmatrix} \tag{6.22}$$

$$Q_k = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.02 \end{bmatrix} \tag{6.23},$$

where p_k and s_k are the position and velocity of the vehicle, respectively. The kinematics of the vehicle is described by two Guassian white random sequences with variances of 0.2m^2 in position and $0.02\text{m}^2\text{s}^{-2}$ in velocity, as indicated by matrix Q_k. The system is assumed to have four independent navigation sensors whose measurement models are defined as follows:

$$\begin{bmatrix} z_{ik}^1 \\ z_{ik}^2 \end{bmatrix} = H_{ik} \begin{bmatrix} p_k \\ s_k \end{bmatrix} + \begin{bmatrix} v_{ik}^1 \\ v_{ik}^2 \end{bmatrix}; \qquad i = 1, 2, 3, 4 \tag{6.24},$$

$$H_{1k} = H_{2k} = H_{3k} = H_{4k} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.25},$$

where z_{ik}^1 and z_{ik}^2 are observations of the vehicle position and velocity, respectively, as measured by the *i*-th sensor; H_{ik} is the *i-th* measurement sensitivity matrix; $v_{ik} = [v_{ik}^1 \quad v_{ik}^2]^T$, i = 1, 2, 3, 4, are uncorrelated zero-mean Gaussian white noise vector sequences with covariance

matrices R_{ik} defined in each particular simulation. The process initial conditions are defined as $x_0 = [p_0 \quad s_0]^T = [0 \quad 0]^T$.

The parameters that define the fuzzy sets in the FISs used in each FL-AKF to adjust R_{ik} are a = 5 and b = 0.3. While the parameters which define the fuzzy sets in the FISs used in each FLA algorithm in the FL-AKF-FLA architecture are g = 1.5 and h = 3. The size of the sliding window in all FL-AKFs is selected as 15. In all simulations the value of the sharing factor β_i, required in the standard FKF and the FL-AFKF, is defined to be equal for all local and master filters, this is β_i = 1/5 = 0.2 (there are considered four local filters plus one master filter).

Simulation 1: The objective of this simulation is to investigate and compare the performance of the proposed adaptive MSDF architectures when initial measurement noise covariance matrices R_{i0} are correctly specified and no adaptation is performed. This means that the adaptation procedure in the local FL-AKFs is switched off in all architectures (in strict sense, standard Kalman filters (SKFs) are used as local filters). In consequence, it is expected to obtain optimal results in the FL-ACKF, FL-ADKF and FL-AFKF, which will be the base for comparison purposes.

The correct measurement noise covariance matrices are constant matrices defined as:

$$R_{1k} = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix};\ R_{2k} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix};\ R_{3k} = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix};\ R_{4k} = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} \tag{6.26}$$

Therefore, the system defined by (6.22)-(6.26) was simulated together with the four MSDF algorithms for 80 sec with a sample time of Δt = 0.2 sec. The initial conditions for Kalman filtering in all cases were specified as: $\hat{x}_{i0(-)} = [0 \quad 0]^T$ and $P_{i0(-)} = 10 I_2$.

Results: For comparison purposes, the following root mean squared error (*RMSE*) performance measures were adopted:

$$RMSE_p = \sqrt{\frac{1}{L}\sum_{k=1}^{L}(p_k - \hat{p}_k)^2} \tag{6.27}$$

$$RMSE_s = \sqrt{\frac{1}{L}\sum_{k=1}^{L}(s_k - \hat{s}_k)^2} \tag{6.28}$$

$$RMSE_x = \sqrt{\frac{1}{L}\sum_{k=1}^{L}(x_k - \hat{x}_k)^T(x_k - \hat{x}_k)} \tag{6.29}$$

where p_k is the actual position, $\hat{p}_k$ is the estimated position, s_k is the actual velocity, and $\hat{s}_k$ is the estimated velocity of the vehicle at instant of time *k*. x_k is the actual state-vector value, and $\hat{x}_k$ is the estimated state-vector value at instant of time *k*; *L* is the number of samples.

Table 6.5 shows the *RMSE* values obtained by employing each one of the four MSDF architectures, as well as the *RMSE* values obtained by local FL-AKFs. As expected, due to the use of the correct noise statistics, the *RMSE* values for the fused data using the FL-ACKF, FL-ADKF and FL-AFKF are exactly the same, and these values are optimal.

Table 6.5 Performance measures: Simulation 1

MSDF Architecture	$RMSE_p$	$RMSE_S$	$RMSE_X$	Conditions
FL-AKF-FLA	0.5376	0.3444	0.6384	Fused data
FL-AKF 1	0.8334	0.4791	0.9613	Correct R_{10} and no adaptation
FL-AKF 2	0.6892	0.3347	0.7661	Correct R_{20} and no adaptation
FL-AKF 3	0.6038	0.4890	0.7770	Correct R_{30} and no adaptation
FL-AKF 4	0.8771	0.4425	0.9824	Correct R_{40} and no adaptation
FL-ACKF	0.4720	0.2924	0.5552	Fused data, Correct R_{i0} and no adaptation **(optimal case)**
FL-ADKF	0.4720	0.2924	0.5552	Fused data **(optimal case)**
FL-AKF 1	0.8334	0.4791	0.9613	Correct R_{10} and no adaptation
FL-AKF 2	0.6892	0.3347	0.7661	Correct R_{20} and no adaptation
FL-AKF 3	0.6038	0.4890	0.7770	Correct R_{30} and no adaptation
FL-AKF 4	0.8771	0.4425	0.9824	Correct R_{40} and no adaptation
FL-AFKF	0.4720	0.2924	0.5552	Fused data **(optimal case)**
FL-AKF 1	0.8270	0.4245	0.9296	Correct R_{10} and no adaptation
FL-AKF 2	0.7708	0.3995	0.8682	Correct R_{20} and no adaptation
FL-AKF 3	0.7133	0.3603	0.7991	Correct R_{30} and no adaptation
FL-AKF 4	0.8924	0.3675	0.9651	Correct R_{40} and no adaptation

Analysing the data in Table 6.5, a comparison of performance measures based on the obtained $RMSE_x$ values can be carried out as follows. The performance measure of the FL-AKF-FLA is 15% away from the optimal value. However, this performance measure is better than the obtained with any local filter. Specifically, the FL-AKF-FLA is 20% more accurate than local FL-AKF 2, which has the best individual performance measure. The performance measure of the FL-ACKF cannot be compared with local filters, because in this case there are no local filters. But this comparison indeed can be carried out in the case of the FL-ADKF and the FL-AFKF. The fused data obtained with the FL-ADKF is 38% (considering the $RMSE_x$ value) more accurate than the obtained with local FL-AKF 2, which has the best individual performance measure in this case. Meanwhile, the performance measure obtained by using the FL-AFKF is 44% better than that obtained with local FL-AKF 3, which has the best performance measure in this case. Finally, note that on average the results obtained with local filters in the FL-AFKF are slightly less accurate than those obtained with local filters in the FL-ADKF. This is not surprising because in the FL-AFKF each local filter uses partial information in performing local estimations. Also it is noteworthy to mention that, in accordance with the theory, the results of each local filter in both the FL-ADKF and the FL-AFKF may be locally suboptimal, but when combined (fused) they are optimal, as those results obtained with the FL-ACKF. In this simulation this is possible because all conditions for optimality are present.

In order to appreciate graphically the results, in figure 6.9 the actual position and the estimated position of the vehicle, obtained with the FL-AKF-FLA algorithm and each of its local FL-AKFs, is plotted. While in figure 6.10 the actual position and the estimated position of the vehicle, obtained with the FL-ADKF and each of its local FL-AKFs, is plotted.

Simulation 2: The goal of this simulation is to investigate the performance of the proposed MSDF architectures when initial measurement noise covariance matrices R_{i0} are incorrectly specified and no adaptation is performed. This means that the adaptation procedure in the FL-AKFs is switched off in all architectures, as it is in simulation 1, but now the conditions for optimality are not present. Thus, it is expected that the performance measure values will be significantly degraded with respect to those obtained in simulation 1.

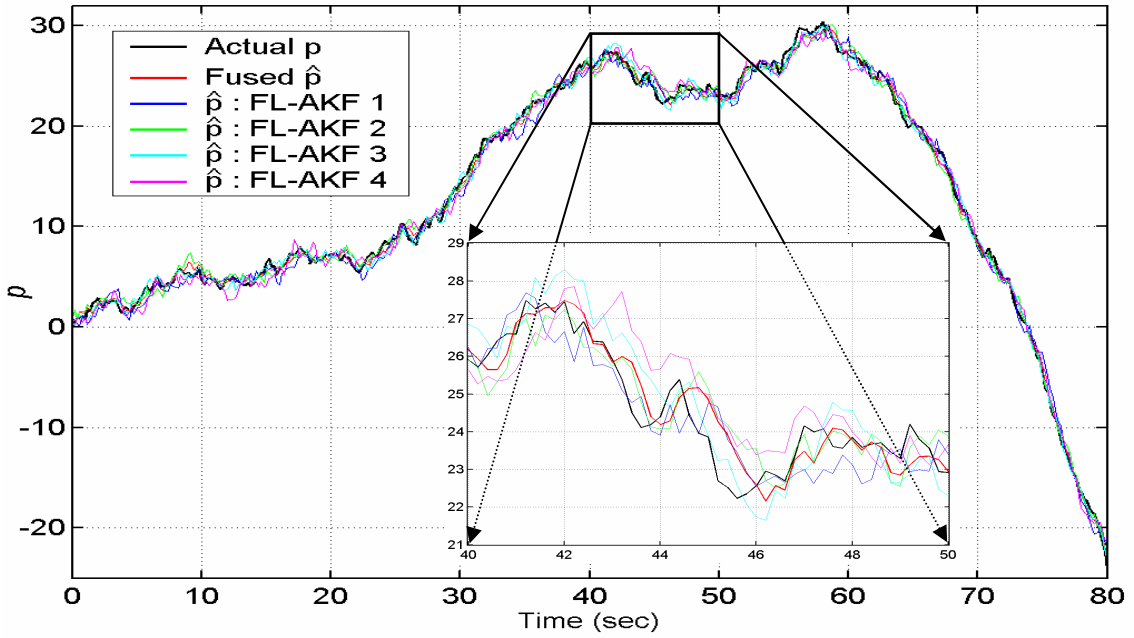


Figure 6.9 Actual position and estimated position of the vehicle obtained with the FL-AKF-FLA and each of its local FL-AKFs, simulation 1.

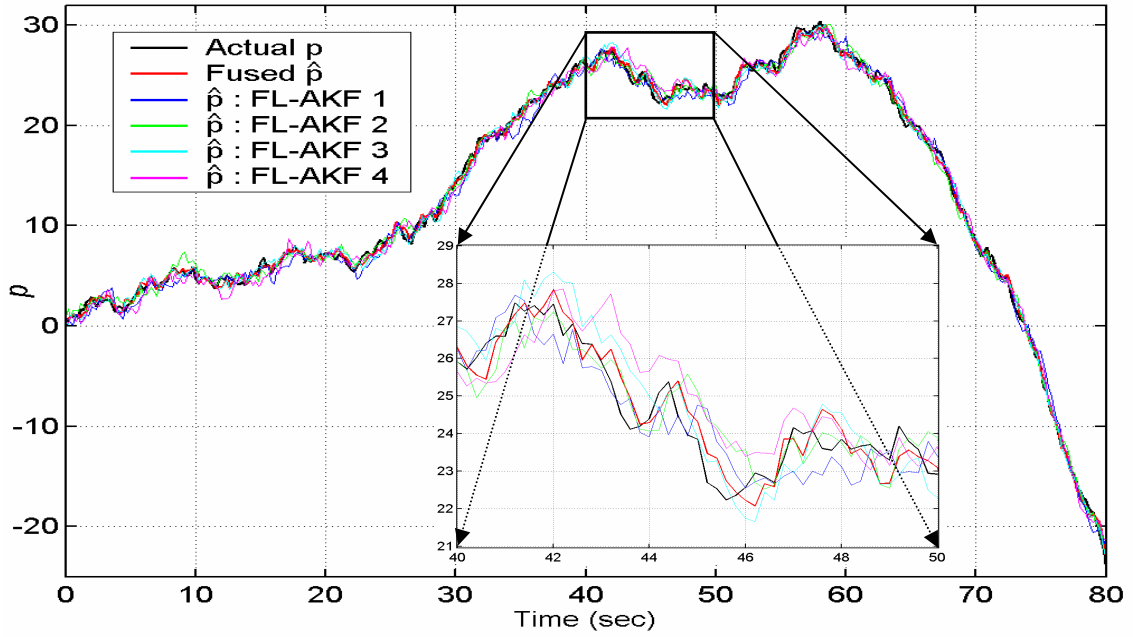


Figure 6.10 Actual position and estimated position of the vehicle obtained with the FL-ADKF and each of its local FL-AKFs, simulation 1.

The system under consideration was simulated together with the four MSDF algorithms for 80 sec with a sample time of Δt = 0.2 sec. The initial conditions for Kalman filtering are the same that those defined in simulation 1. The correct measurement noise covariance matrices are those given by (6.26). However, in this case it is assumed that these values are unknown. Therefore, an initial guess is made as:

$$R_{10} = R_{20} = R_{30} = R_{40} = \begin{bmatrix} 1 & 0 \\ 0 & 6 \end{bmatrix} \qquad (6.30).$$

Results: Table 6.6 shows the performance measures obtained with the MSDF architectures as well as those obtained with local filters. By comparing these results with the optimal performance measures (considering the $RMSE_x$ values), it is observed that the FL-AKF-FLA performance measure is degraded in 33%, while the CKF, DKF, and FKF performances are degraded in 15.6%. These results prove that by having switched off the adaptation procedure in the FL-AKFs (this is having standard KFs) and using incorrect measurement noise statistics, the MSDF performances are substantially degraded. This fact can be graphically seen in figure 6.11, where the actual position and the estimated position of the vehicle obtained with the FL-ADKF and each of its local FL-AKFs are plotted. Finally, if the FL-AKF-FLA performance is compared with the one obtained in simulation 1 then a degradation of 15.8% is observed.

Table 6.6 Performance measures: Simulation 2

MSDF Architecture	$RMSE_p$	$RMSE_s$	$RMSE_x$	Conditions
FL-AKF-FLA	0.5681	0.4729	0.7391	Fused data
FL-AKF 1	1.0010	0.5457	1.1400	Incorrect R_{10} and no adaptation
FL-AKF 2	0.7472	0.4265	0.8604	Incorrect R_{20} and no adaptation
FL-AKF 3	0.6065	0.5019	0.7872	Incorrect R_{30} and no adaptation
FL-AKF 4	0.9773	0.4892	1.0930	Incorrect R_{40} and no adaptation
FL-ACKF	0.5403	0.3464	0.6418	Fused data, incorrect R_{i0} and no adaptation
FL-ADKF	0.5403	0.3464	0.6418	Fused data
FL-AKF 1	1.0010	0.5457	1.140	Incorrect R_{10} and no adaptation
FL-AKF 2	0.7472	0.4265	0.8604	Incorrect R_{20} and no adaptation
FL-AKF 3	0.6065	0.5019	0.7872	Incorrect R_{30} and no adaptation
FL-AKF 4	0.9773	0.4892	1.0930	Incorrect R_{40} and no adaptation
FL-AFKF	0.5403	0.3464	0.6418	Fused data
FL-AKF 1	1.3160	0.4207	1.3820	Incorrect R_{10} and no adaptation
FL-AKF 2	0.9182	0.3535	0.9840	Incorrect R_{20} and no adaptation
FL-AKF 3	0.6899	0.3992	0.7971	Incorrect R_{30} and no adaptation
FL-AKF 4	1.2190	0.4024	1.2830	Incorrect R_{40} and no adaptation

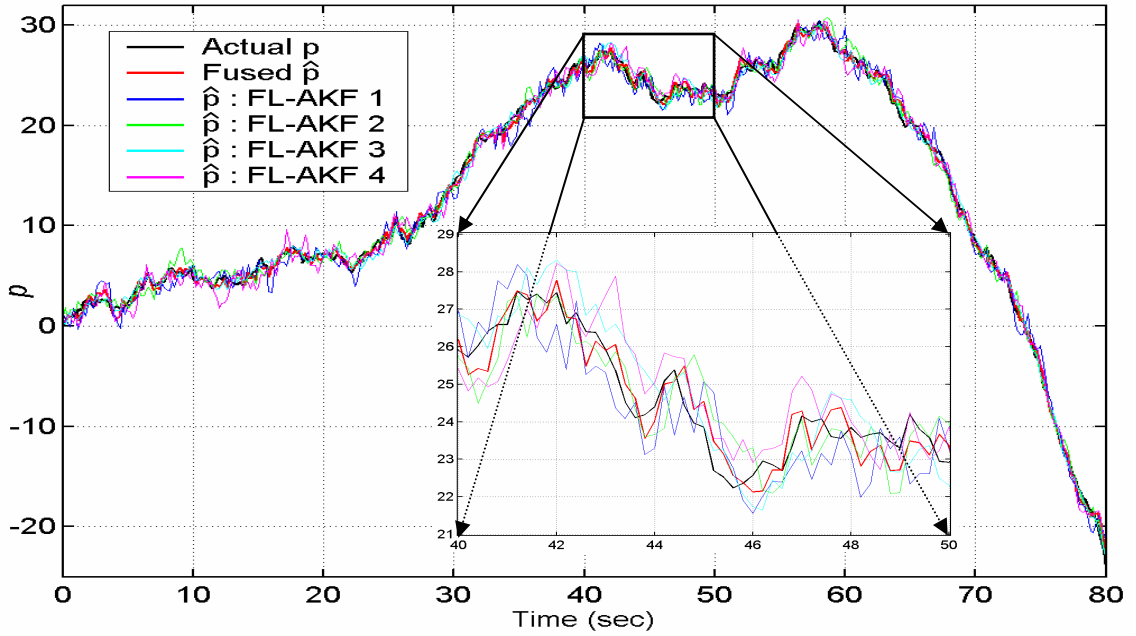


Figure 6.11 Actual position and estimated position of the vehicle obtained with the FL-ADKF and each of its local FL-AKFs, simulation 2.

Simulation 3: The purpose of this simulation is to investigate the performance of the proposed adaptive MSDF algorithms when initial measurement noise covariance matrices R_{i0} are correctly specified and adaptation is performed. Thus, in this case the adaptation procedure in the local FL-AKFs is switched on in all architectures. This simulation will show the level of degradation in performance, compared with the optimal performance measures, when carrying out adaptation under the correct noise statistics.

The system under consideration and the MSDF algorithms were simulated for 80 sec with a sample time $\Delta t = 0.2$ sec. The initial conditions for Kalman filtering are the same than those defined in simulation 1. The correct measurement noise covariance matrices are those given by (6.28).

Results: Table 6.7 shows the obtained performance measures for the local FL-AKFs and the four MSDF algorithms. Comparing these results with the optimal performance measures (considering the $RMSE_x$ values), it is observed that the FL-AKF-FLA performance is degraded by 15%, the FL-ACKF and the FL-AFKF performances are degraded only 0.3%, while the FL-ADKF performance is degraded by 0.25%. It is remarkable that the degradation in performance presented in the FL-ACKF, the FL-ADKF, and the FL-AFKF fusion algorithms is almost imperceptible. This means that, if the actual initial noise statistics are specified and the adaptation algorithm is switched on, then the performance of these three MSDF architectures remain near to the optimal. Note also that, if the FL-AKF-FLA performance is compared with the one obtained in simulation 1, then a degradation of only 0.06% is observed.

Figure 6.12 shows the actual position and estimated position of the vehicle obtained with the FL-ADKF and each of its local FL-AKFs. The way in which the elements in the main diagonal of matrices R_{1k}, R_{2k}, R_{3k}, and R_{4k} are adjusted in each of the local FL-AKFs of the FL-ADKF can be appreciated in figure 6.13. Note that, because the correct initial values are given, the performed adjustment maintains these values in a quasi steady-state around the original and correct values.

Table 6.7 Performance measures: Simulation 3

MSDF Architecture	$RMSE_p$	$RMSE_s$	$RMSE_x$	Conditions
FL-AKF-FLA	0.5433	0.336	0.6388	Fused data
FL-AKF 1	0.8429	0.4753	0.9677	Correct R_{10} and adaptation
FL-AKF 2	0.6981	0.3466	0.7795	Correct R_{20} and adaptation
FL-AKF 3	0.6112	0.4817	0.7782	Correct R_{30} and adaptation
FL-AKF 4	0.861	0.4632	0.9777	Correct R_{40} and adaptation
FL-ACKF	0.4722	0.2953	0.5569	Fused data, orrect R_{i0} and adaptation
FL-ADKF	0.4718	0.2954	0.5566	Fused data
FL-AKF 1	0.8429	0.4753	0.9677	Correct R_{10} and adaptation
FL-AKF 2	0.6981	0.3466	0.7795	Correct R_{20} and adaptation
FL-AKF 3	0.6112	0.4817	0.7782	Correct R_{30} and adaptation
FL-AKF 4	0.861	0.4632	0.9777	Correct R_{40} and adaptation
FL-AFKF	0.4722	0.2953	0.5569	Fused data
FL-AKF 1	0.8399	0.4238	0.9408	Correct R_{10} and adaptation
FL-AKF 2	0.7829	0.4036	0.8808	Correct R_{20} and adaptation
FL-AKF 3	0.7164	0.3927	0.8170	Correct R_{30} and adaptation
FL-AKF 4	0.8748	0.3917	0.9585	Correct R_{40} and adaptation

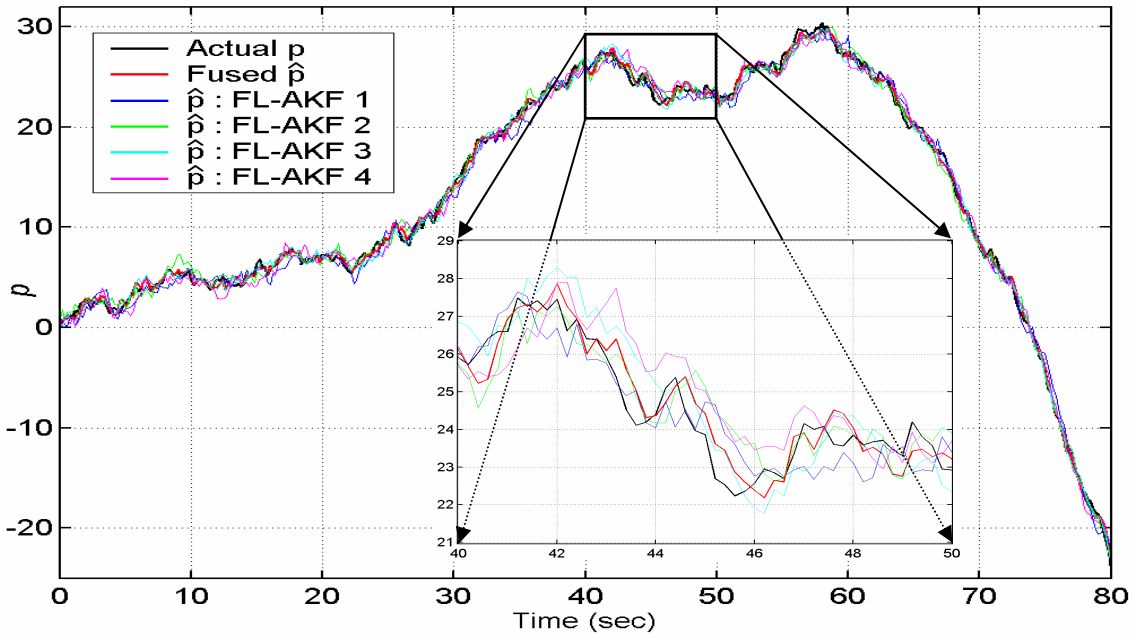


Figure 6.12 Actual position and estimated position of the vehicle obtained with the FL-ADKF and each of its local FL-AKFs, simulation 3.

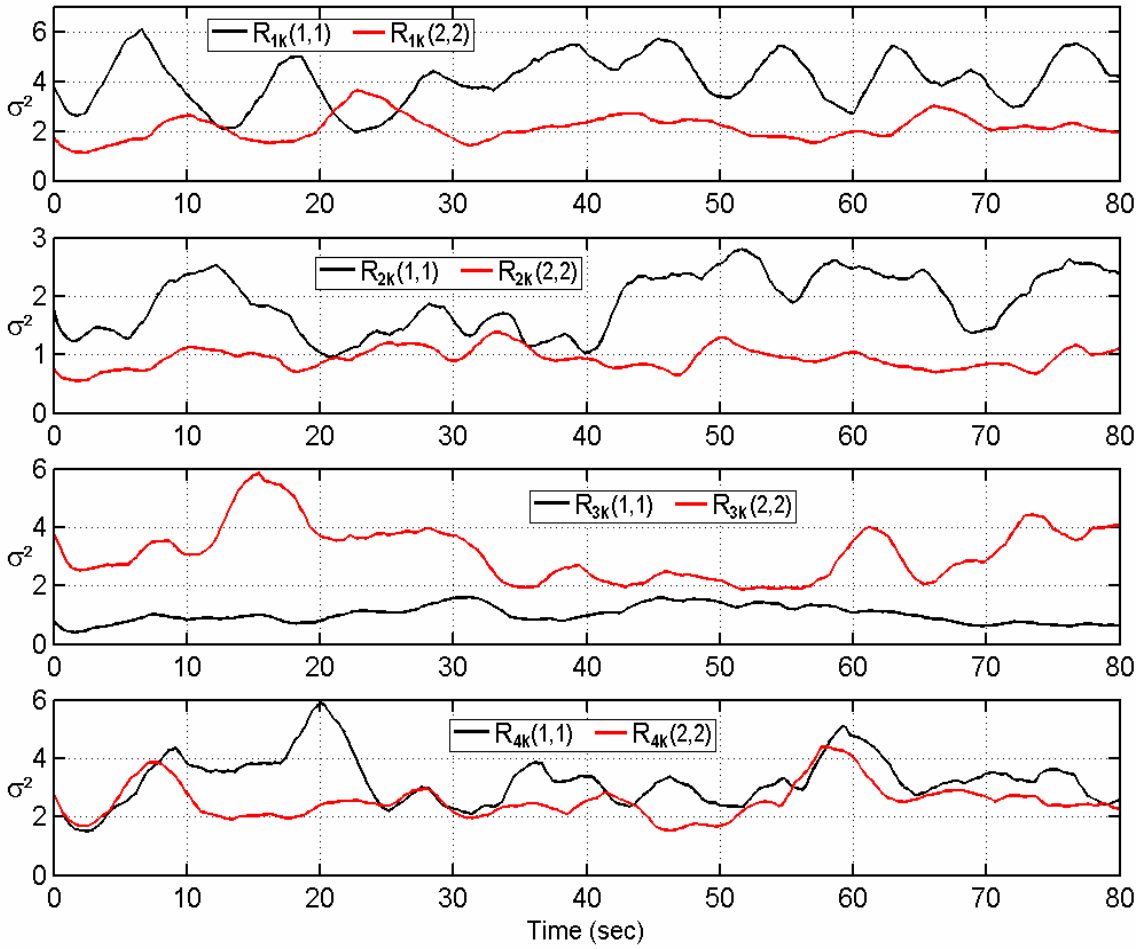


Figure 6.13 Adjustment of the elements in the main diagonal of matrices R_{1k}, R_{2k}, R_{3k}, and R_{4k} in each of the local FL-AKFs of the FL-ADKF, Simulation 3.

Simulation 4: The objective of this simulation is to investigate the performance of the proposed adaptive MSDF architectures when the initial measurement noise covariance matrices R_{i0} are incorrectly specified and adaptation is performed. Thus, the adaptation procedure in the FL-AKFs is switched on in all architectures. This simulation will show how the adaptation of the elements in the main diagonal of matrices R_{1k}, R_{2k}, R_{3k}, and R_{4k} is carried out.

The MSDF algorithms and the system under consideration were simulated for 80 sec with a sample time Δt = 0.2 sec. The initial conditions for Kalman filtering are the same than those defined in simulation 1. The incorrect measurement noise covariance matrices are those given by (6.30).

Results: Table 6.8 shows the obtained performance measures for the local FL-AKFs and the four adaptive MSDF algorithms. Comparing these results with the optimal values (considering the $RMSE_x$ values), next observations are made. The degradation in performance of the FL-AKF-FLA algorithm is 15.7%. In the case of the FL-ACKF and the FL-AFKF, the degradation is 1.6%. The minimum degradation in performance is observed in the FL-DKF, which is 1.5%. Note that, however the performance of the FL-AKF-FLA is far from the optimal, this is better than the performance observed in any local FL-AKF. Particularly, the performance of the FL-AKF-FLA is better in 20.5% with respect to the performance observed in the FL-AKF 3, which has the best individual performance measure. In figure 6.14 is shown the actual position and estimated position of the vehicle obtained with the FL-ADKF and each of its local FL-AKFs.

It is remarkable to mention that on average the degradation in performance observed in the FL-ACKF, the FL-ADKF and the FL-AFKF is less than 2%. This means that the adaptation carried out in each one of these algorithms effectively tune the value of the corresponding measurement noise covariance matrices to fit the actual noise statistics. This can be appreciated graphically in figure 6.15 where the values of the elements in the main diagonal of matrices R_{1k}, R_{2k}, R_{3k}, and R_{4k} in each of the local FL-AKFs of the FL-ADKF are plotted.

Table 6.8 Performance measures: Simulation 4

MSDF Architecture	$RMSE_p$	$RMSE_s$	$RMSE_x$	Conditions
FL-AKF-FLA	0.5435	0.3428	0.6426	Fused data
FL-AKF 1	0.8537	0.4712	0.9751	Incorrect R_{10} and adaptation
FL-AKF 2	0.7019	0.3467	0.7829	Incorrect R_{20} and adaptation
FL-AKF 3	0.6118	0.4743	0.7741	Incorrect R_{30} and adaptation
FL-AKF 4	0.8652	0.4612	0.9804	Incorrect R_{40} and adaptation
FL-ACKF	0.4753	0.3038	0.5642	Fused data, incorrect R_{i0} and adaptation
FL-ADKF	0.4745	0.3041	0.5636	Fused data
FL-AKF 1	0.8537	0.4712	0.9751	Incorrect R_{10} and adaptation
FL-AKF 2	0.7019	0.3467	0.7829	Incorrect R_{20} and adaptation
FL-AKF 3	0.6118	0.4743	0.7741	Incorrect R_{30} and adaptation
FL-AKF 4	0.8652	0.4612	0.9804	Incorrect R_{40} and adaptation
FL-AFKF	0.4753	0.3038	0.5642	Fused data
FL-AKF 1	0.8672	0.4222	0.9646	Incorrect R_{10} and adaptation
FL-AKF 2	0.7907	0.3965	0.8845	Incorrect R_{20} and adaptation
FL-AKF 3	0.7156	0.4000	0.8198	Incorrect R_{30} and adaptation
FL-AKF 4	0.8917	0.3946	0.9751	Incorrect R_{40} and adaptation

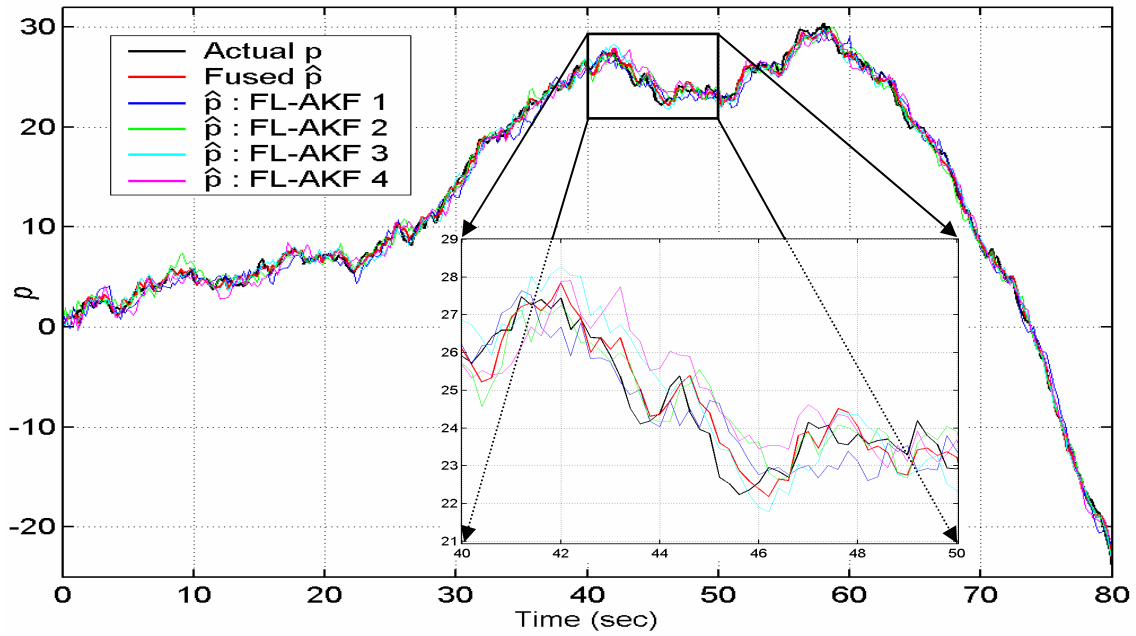


Figure 6.14 Actual position and estimated position of the vehicle obtained with the FL-ADKF and each of its local FL-AKFs, simulation 4.

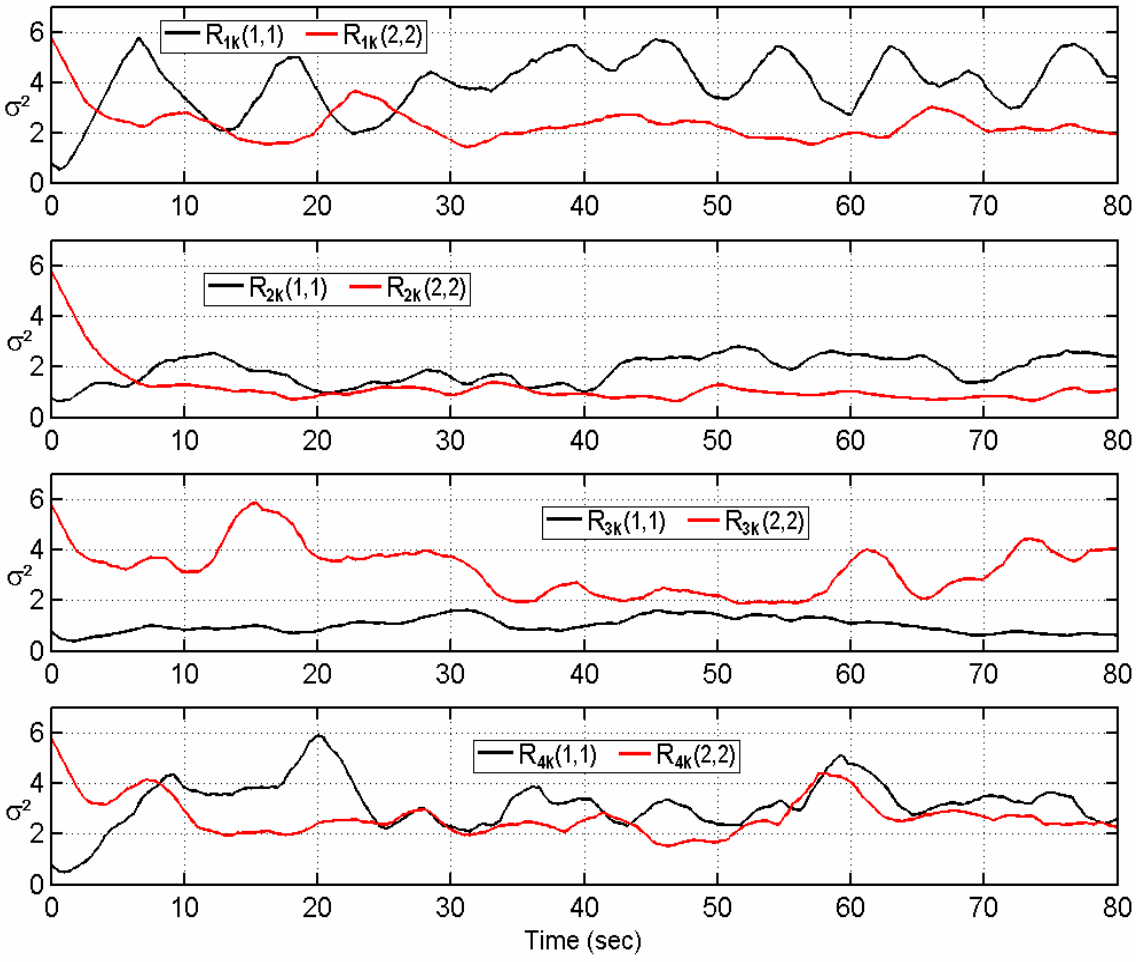


Figure 6.15 Adjustment of the elements in the main diagonal of matrices R_{1k}, R_{2k}, R_{3k}, and R_{4k} in each of the local FL-AKFs of the FL-ADKF, simulation 4.

Simulation 5: The goal of this simulation is to investigate the performance of the proposed adaptive MSDF architectures under different noise profiles in the sensors. Four different noise profiles, with different statistics, are being considered as is shown in figure 6.16.

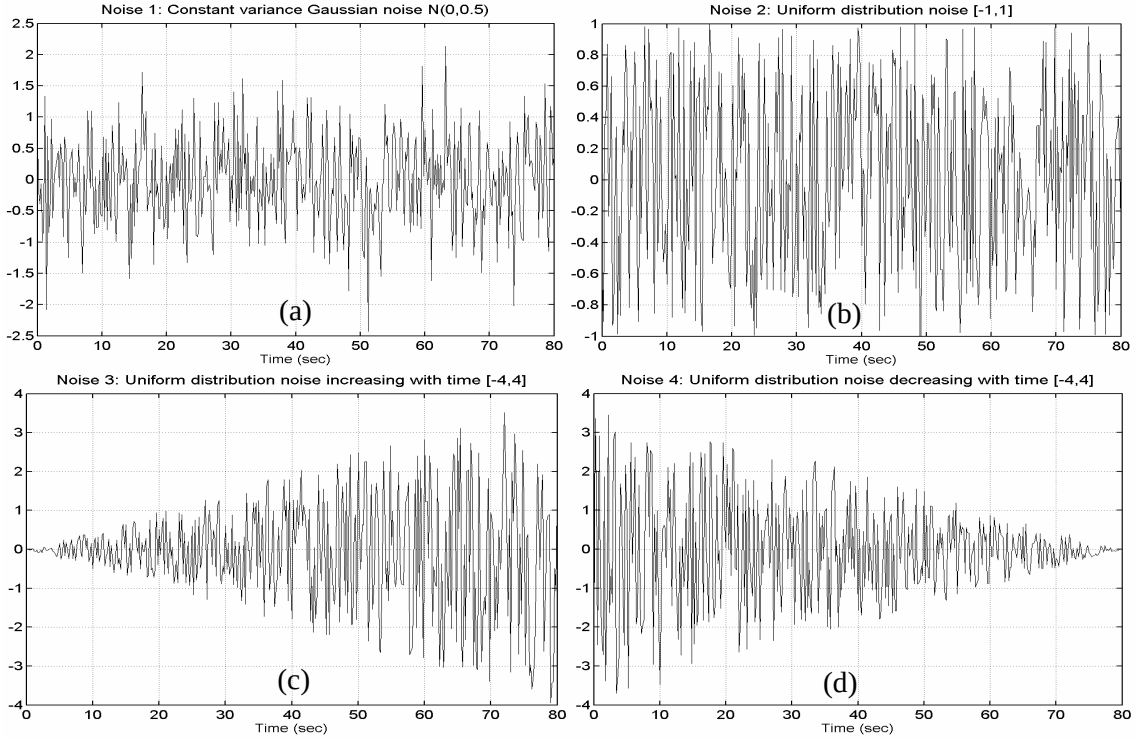


Figure 6.16 Noise profiles used in simulation 5; (a) Constant variance Gaussian noise sequence $\boldsymbol{N}(0,0.5)$; (b) Uniform distribution random noise sequence [-1,1]; (c) Uniform distribution random noise sequence increasing with time [-4,4]; (d) Uniform distribution random noise sequence decreasing with time [-4,4].

The four MSDF algorithms and the system under consideration were simulated for 80 sec with a sample time Δt = 0.2 sec. The initial conditions for Kalman filtering are the same that those defined in simulation 1. It is assumed that the actual noise statistics in all sensors are unknown, but an initial guess of the measurement noise covariance matrices is made as given by (6.30). It is expected that the adjusting procedure will tune the values of the diagonal elements of matrices R_{1k}, R_{2k}, R_{3k}, and R_{4k} to fit, as closely as possible, the actual statistics of the noise profiles. The noise profiles used in each specific sensor are those indicated in the last column of Table 6.9.

Results: Table 6.9 shows the obtained performance measures for the local FL-AKFs and the four adaptive MSDF algorithms. Comparing these results (considering the $RMSE_x$ values), the following remarks can be made. The worst performance measure is obtained with the FL-AKF-FLA algorithm, while the best performance measure is obtained with the FL-ADKF. The performance measure obtained with both the FL-ACKF and the FL-AFKF is exactly the same, and this is 2.8% worse than the obtained with the FL-ADKF. The performance measure of the FL-AKF-FLA is 20.5% worst than that obtained with the FL-ADKF algorithm. However, this performance measure, compared with local filters, is 18.6% better than that observed in local FL-AKF 2, which from this group of filters has the best performance measure.

A graphical view of the obtained results can be appreciated in figure 6.17, where the actual and estimated position of the vehicle, made by each one of the local FL-AKFs and the FL-ADKF, is plotted. The way in which the adjusting procedure tunes the values of the diagonal

element of matrices R_{1k}, R_{2k}, R_{3k}, and R_{4k} in each of the local FL-AKFs of the FL-ADKF, trying to match the actual statistics of the noise profiles, can be appreciated in figure 6.18. Note that good approximation to the actual noise statistics is achieved in all cases. It is interesting to note that if there is a non-stationary statistics (but zero-average) noise profile present in any of the sensors, as in sensors 3 and 4 in this simulation, then the adjusting procedure tunes the covariance values in the corresponding measurement covariance matrices to fit and follow, as closely as possible, the actual dynamic statistics of the noise profiles.

Table 6.9 Performance measures: Simulation 5

MSDF Architecture	$RMSE_p$	$RMSE_s$	$RMSE_x$	Conditions
FL-AKF-FLA	0.3701	0.2818	0.4652	Fused data
FL-AKF 1	0.4796	0.2833	0.5570	v_k^1 = noise 1, v_k^2 = noise 2
FL-AKF 2	0.4393	0.3334	0.5516	v_k^1 = noise 2, v_k^2 = noise 1
FL-AKF 3	0.6403	0.3534	0.7314	v_k^1 = noise 3, v_k^2 = noise 4
FL-AKF 4	0.5430	0.4009	0.6749	v_k^1 = noise 4, v_k^2 = noise 3
FL-ACKF	0.3213	0.2291	0.3946	Fused data
Sensor 1				v_k^1 = noise 1, v_k^2 = noise 2
Sensor 2				v_k^1 = noise 2, v_k^2 = noise 1
Sensor 3				v_k^1 = noise 3, v_k^2 = noise 4
Sensor 4				v_k^1 = noise 4, v_k^2 = noise 3
FL-ADKF	0.3113	0.2285	0.3862	Fused data
FL-AKF 1	0.4796	0.2833	0.5570	v_k^1 = noise 1, v_k^2 = noise 2
FL-AKF 2	0.4393	0.3334	0.5516	v_k^1 = noise 2, v_k^2 = noise 1
FL-AKF 3	0.6403	0.3534	0.7314	v_k^1 = noise 3, v_k^2 = noise 4
FL-AKF 4	0.5430	0.4009	0.6749	v_k^1 = noise 4, v_k^2 = noise 3
FL-AFKF	0.3213	0.2291	0.3946	Fused data
FL-AKF 1	0.5219	0.3137	0.6089	v_k^1 = noise 1, v_k^2 = noise 2
FL-AKF 2	0.4803	0.3155	0.5736	v_k^1 = noise 2, v_k^2 = noise 1
FL-AKF 3	0.5872	0.3364	0.6767	v_k^1 = noise 3, v_k^2 = noise 4
FL-AKF 4	0.5746	0.2879	0.6427	v_k^1 = noise 4, v_k^2 = noise 3

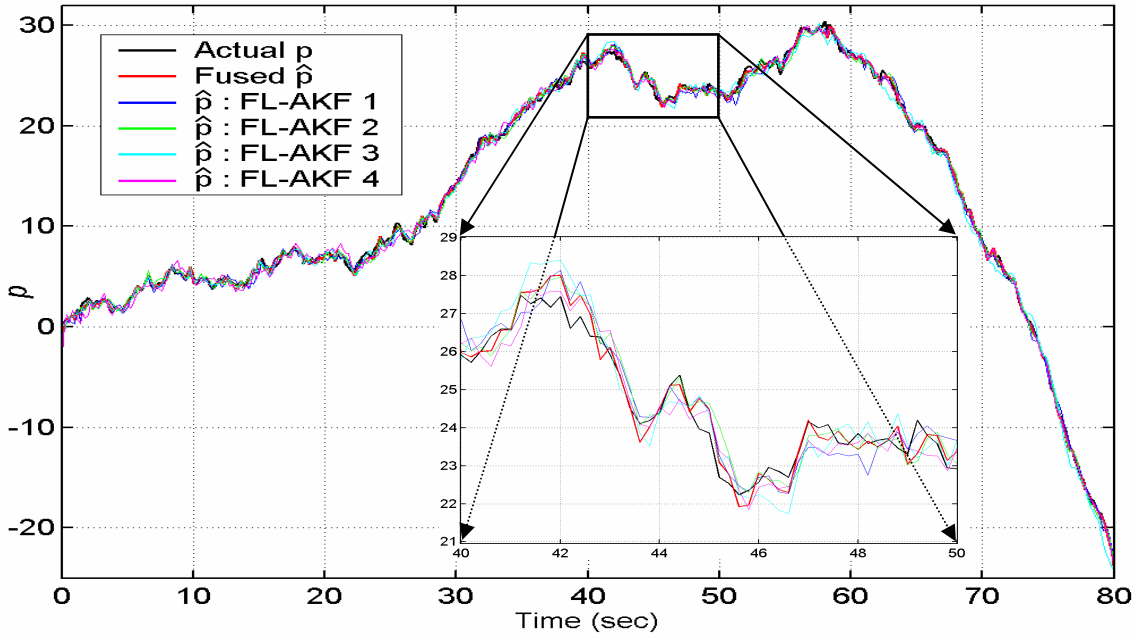


Figure 6.17 Actual position and estimated position of the vehicle obtained with the FL-ADKF and each of its local FL-AKFs, simulation 5.

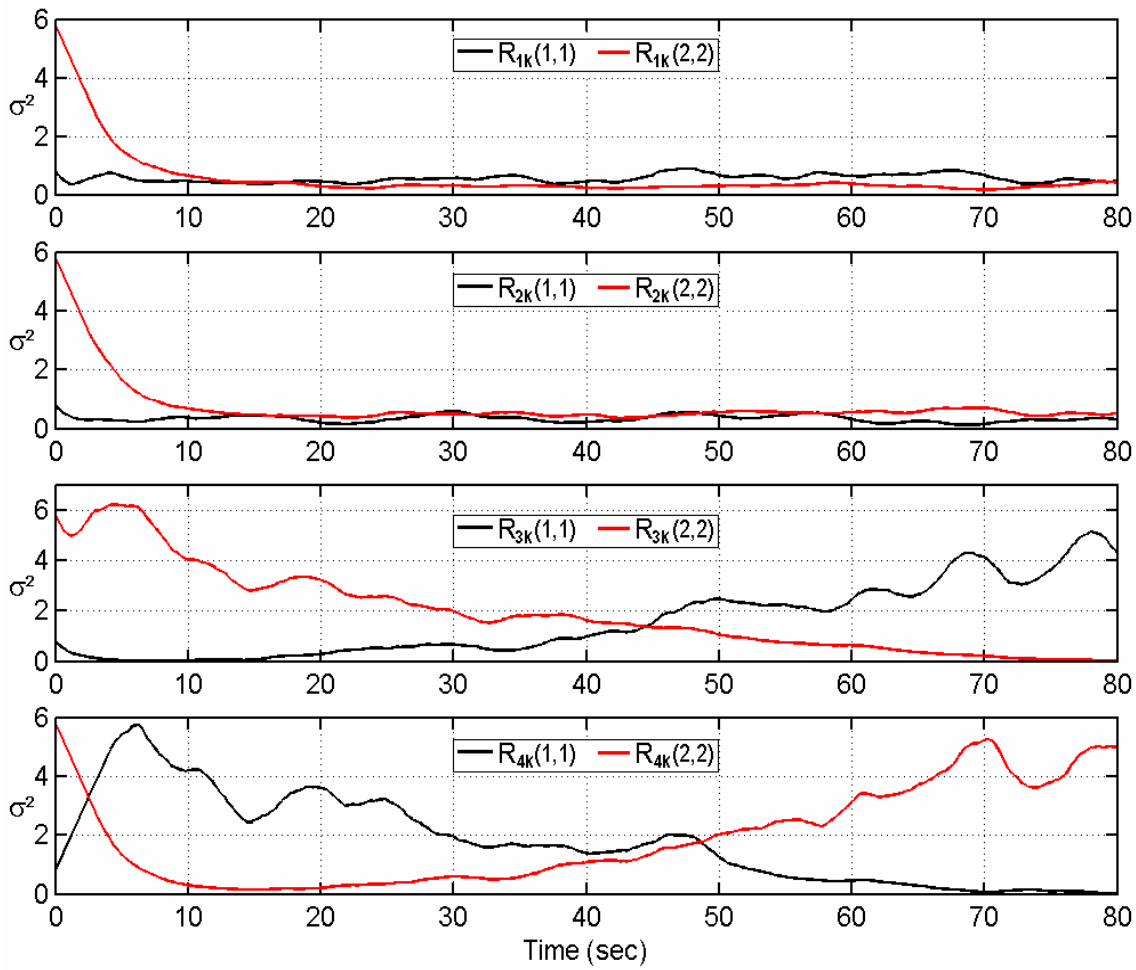


Figure 6.18 Adjustment of the elements in the main diagonal of matrices R_{1k}, R_{2k}, R_{3k}, and R_{4k} in each of the local FL-AKFs of the FL-ADKF, simulation 5.

6.4.1 Analysis and comparison of fault-tolerant characteristics

In all the simulations carried out in the previous sections it is assumed that there are no faults present in the sensors. However, in real situations there is always the possibility of having sensor failures. In the literature it has been demonstrated that the CKF approach lacks robustness when there is spurious data in any of the sensors [Gao et al, 1993] [Brown and Hwang 1997]. Conversely, it has been shown that both DKF and FKF approaches have better fault-tolerant performance than can be achieved with a CKF [Wei and Schwarz, 1990] [Gao et al, 1993]. In this section simulations are carried out in order to investigate, analyse and compare the fault-tolerant performance of the four proposed adaptive MSDF architectures. In particular, the fault-tolerant performances against three kinds of sensor failures, transient faults, persistent faults and permanent faults, are investigated. For this analysis the same process and sensors defined by (6.22)-(6.25) are used. The simulations carried out and results obtained are presented as follows.

Simulation 6: The objective of this simulation is to investigate how each one of the proposed hybrid adaptive MSDF approaches responds to the presence of transient faults, and to examine which design is the most fault-tolerant. Hence, the adaptive MSDF algorithms and the system under consideration were simulated for 80 sec with a sample time Δt = 0.2 sec. The initial conditions for Kalman filtering are the same than those defined in simulation 1. It is assumed that the correct measurement noise covariance matrices, given by (6.26), are unknown.

Therefore, an initial guess is made as given by (6.30). Transient faults are introduced in sensor 3 (which is the most accurate sensor for position measurements) in each MSDF architecture, at times t = 20, 30, 40, 50 and 60 sec, with values 20, -20, 20, -20, and 20 meters respectively, for position measurements.

Results: The performance measures defined by (6.27)-(6.29) are used to compare results. In Table 6.10 the obtained *RMSE* performance measures are tabulated. From that data it can be observed that the most fault-tolerant performance is provided by the FL-AKF-FLA, while the worst fault-tolerant performance is that seen in both the FL-ACKF and the FL-AFKF. The FL-ADKF has an intermediate fault-tolerant performance.

Table 6.10 Performance measures: Simulation 6

MSDF Architecture	$RMSE_p$	$RMSE_s$	$RMSE_x$	Conditions
FL-AKF-FLA	0.5943	0.33357	0.6826	Fused data
FL-AKF 1	0.8537	0.4712	0.9751	Incorrect R_{10} and adaptation
FL-AKF 2	0.7019	0.3467	0.7829	Incorrect R_{20} and adaptation
FL-AKF 3	1.2800	0.5093	1.3780	Incorrect R_{30} and adaptation
FL-AKF 4	0.8652	0.4612	0.9804	Incorrect R_{40} and adaptation
FL-ACKF	0.6572	0.3072	0.7255	Fused data, incorrect R_{i0} and adaptation
FL-ADKF	0.6472	0.3066	0.7161	Fused data
FL-AKF 1	0.8537	0.4712	0.9751	Incorrect R_{10} and adaptation
FL-AKF 2	0.7019	0.3467	0.7829	Incorrect R_{20} and adaptation
FL-AKF 3	1.2800	0.5093	1.3780	Incorrect R_{30} and adaptation
FL-AKF 4	0.8652	0.4612	0.9804	Incorrect R_{40} and adaptation
FL-AFKF	0.6572	0.3072	0.7255	Fused data
FL-AKF 1	0.9128	0.4237	1.0060	Incorrect R_{10} and adaptation
FL-AKF 2	0.8353	0.3966	0.9247	Incorrect R_{20} and adaptation
FL-AKF 3	1.5090	0.4111	1.5640	Incorrect R_{30} and adaptation
FL-AKF 4	0.9420	0.3966	1.0220	Incorrect R_{40} and adaptation

Analysing the performances obtained by local filters, the following remarks can be given. In both the FL-AKF-FLA and the FL-ADKF, only the performance of the local FL-AKF 3, which is processing the faulty data, is degraded. Meanwhile, the performances of the other local filters are unaffected and equal to those results shown in Table 6.8, which correspond to the results obtained under the same conditions but without the faulty data in the local FL-AKF 3. Conversely, note that the performances of all local filters in the FL-AFKF are affected by the faulty data in sensor 3. This is due to the sharing information carried out and the feedback of the fused estimated state vector to each local filter.

Figure 6.19 shows the actual position and the fused estimated position of the vehicle obtained with the FL-AKF-FLA architecture. It can be appreciated that the transient faults are practically without effect in the fused estimated position. In figure 6.20 the actual, measured and estimated position of the vehicle carried out by the FL-AKF 3 and sensor 3, in the FL-AKF-FLA architecture, is shown. Note the effects that the transient faults, introduced at times t = 20, 30, 40, 50, and 60 sec, have on the position estimates made by this local filter. In figure 6.21 the adjustment of the elements in the main diagonal of matrix R_{3k}, FL-AKF 3, FL-AKF-FLA, can be appreciated. Note that, as a result of the presence of the faults, the value of $R_{3k}(1,1)$ is increased at those times where a fault is present. With the increment of the value of $R_{3k}(1,1)$, the value of the degree of confidence factor assigned to the local FL-AKF 3, c_3, is reduced as can be seen in figure 6.22, which shows the degree of confidence factors assigned to each local FL-AKF state

vector estimation in the FL-AKF-FLA approach. Due to this increment in the measurement noise covariance value for the positioning data, the influence of the FL-AKF 3, which has the faulty data, in the fused estimated solution carried out by the FL-AKF-FLA algorithm is reduced. The performance assessment scheme of the FL-AKF-FLA algorithm does not exist in the other MSDF architectures. Therefore, the influence of the faulty data in their fused estimates is greater. As an example, the effects that the faults have in the estimates performed by the FL-ADKF can be seen in figure 6.23.

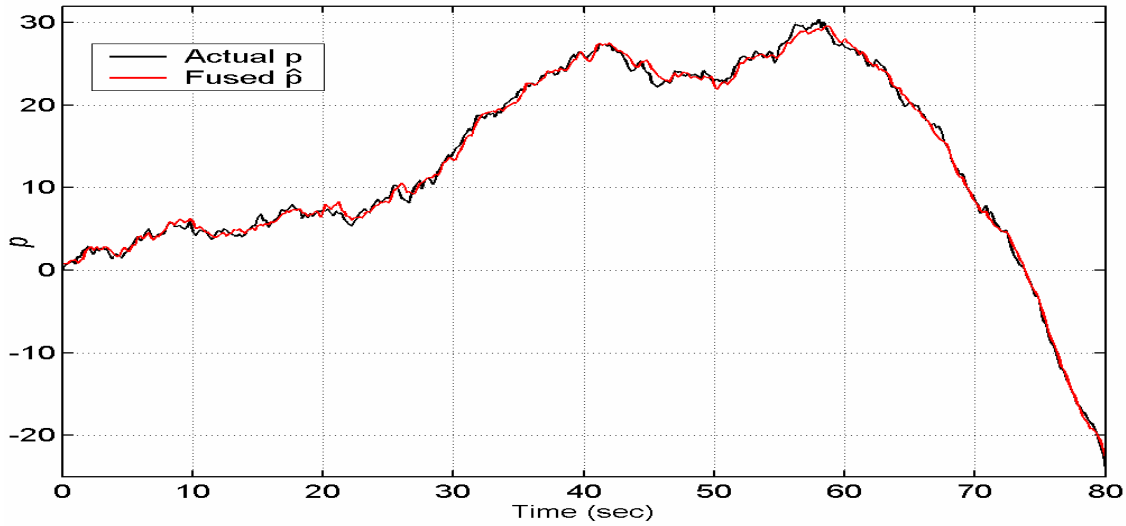


Figure 6.19 Actual position and estimated position of the vehicle obtained with the FL-AKF-FLA architecture, simulation 6.

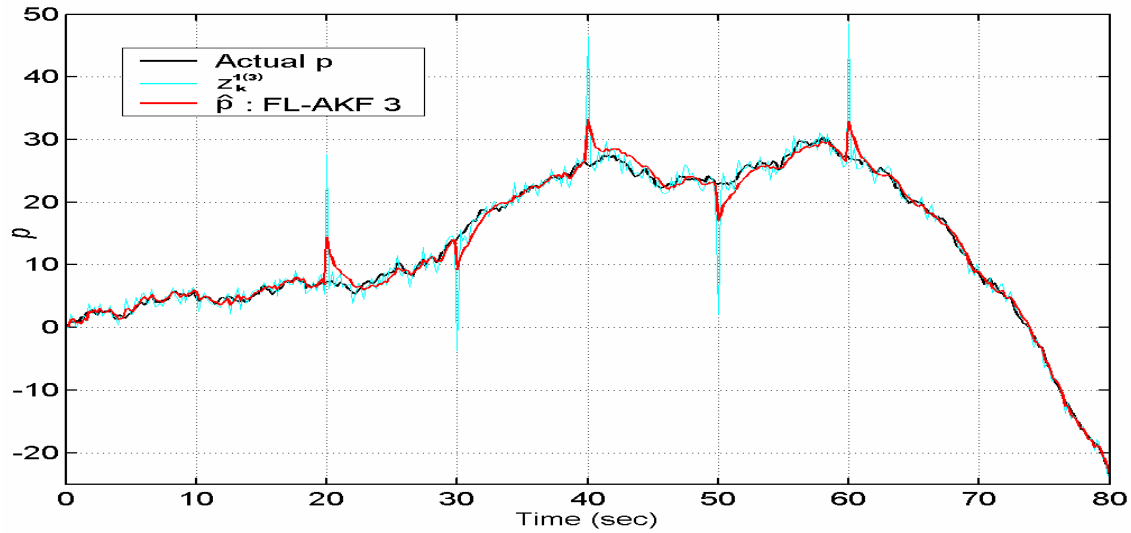


Figure 6.20 Actual position, measured position ($z_k^{1(3)}$), and estimated position of the vehicle obtained with the FL-AKF 3, FL-AKF-FLA, simulation 6.

Simulation 7: The goal of this simulation is to investigate the fault-tolerant performances of the adaptive MSDF algorithms when persistent faults (e. g. cycle slips in GPS applications [Gao et al, 1993]) are present in one of the sensors. Therefore, the adaptive MSDF algorithms and the system under consideration were simulated under the same conditions defined in the previous simulation. However, in this case three persistent faults were simulated in sensor 3 in each MSDF architecture, at times t = 20, 40, and 60 sec, with duration of 1, 1.6, and 2 sec, respectively, and all with values of 20 meters for position measurements.

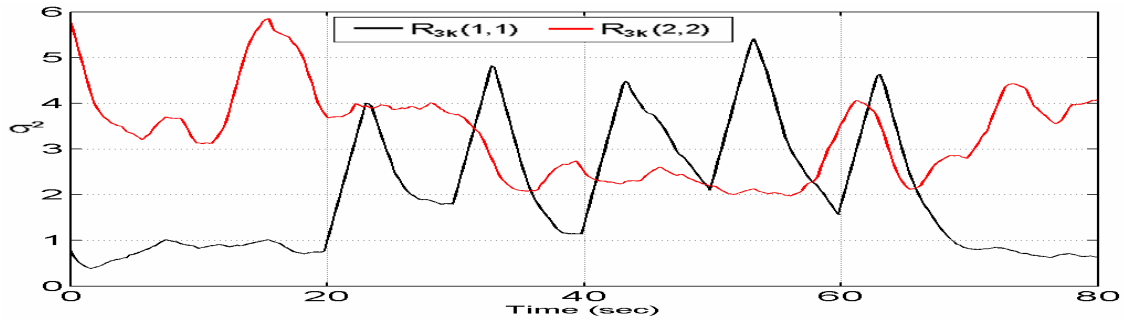


Figure 6.21 Adjustment of the elements in the main diagonal of matrix R_{3k}, FL-AKF 3, FL-AKF-FLA, simulation 6.

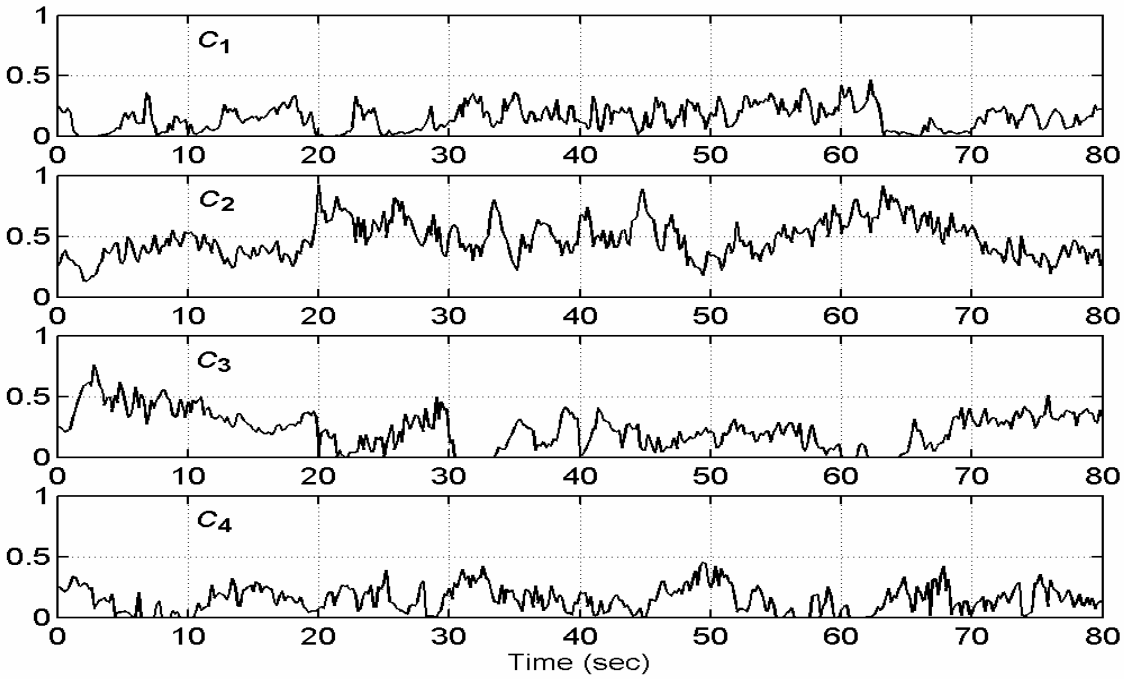


Figure 6.22 Degree of confidence factors assigned to each local FL-AKF state-vector estimate in the FL-AKF-FLA approach, simulation 6.

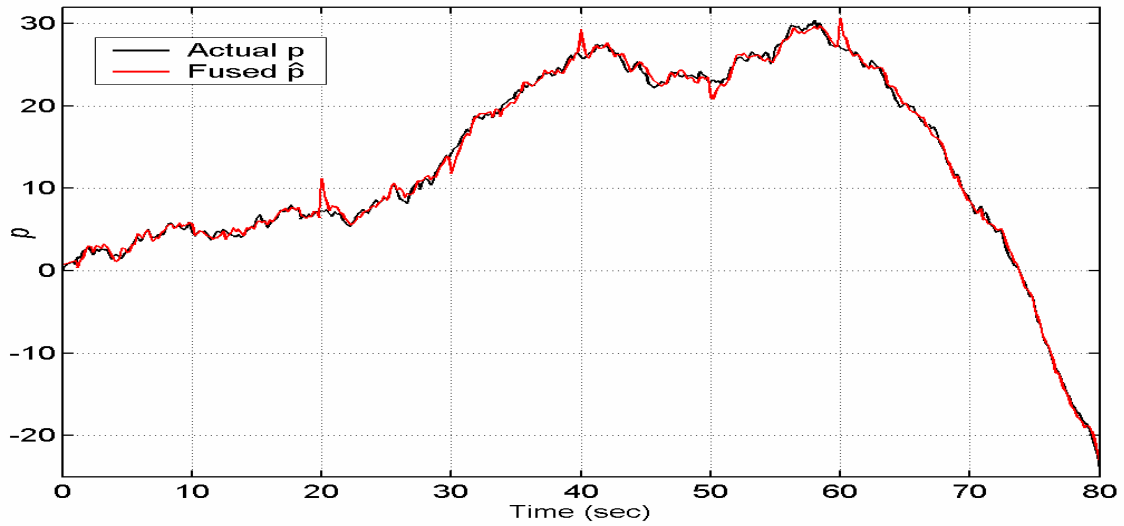


Figure 6.23 Actual position and estimated position of the vehicle obtained with the FL-ADKF, simulation 6.

Results: Table 6.11 shows the obtained performance measures for the local FL-AKFs and the hybrid adaptive MSDF algorithms. It can be noted that, like in the previous simulation, the most fault-tolerant performance is that of the FL-AKF-FLA, while the least fault-tolerant performance is that observed in both the FL-ACKF and the FL-AFKF. The FL-ADKF has an intermediate fault-tolerant performance.

Table 6.11 Performance measures: Simulation 7

MSDF Architecture	$RMSE_p$	$RMSE_S$	$RMSE_X$	Conditions
FL-AKF-FLA	0.7009	0.3406	0.7793	Fused data
FL-AKF 1	0.8537	0.4712	0.9751	Incorrect R_{10} and adaptation
FL-AKF 2	0.7019	0.3467	0.7829	Incorrect R_{20} and adaptation
FL-AKF 3	2.4420	0.5577	2.5050	Incorrect R_{30} and adaptation
FL-AKF 4	0.8652	0.4612	0.9804	Incorrect R_{40} and adaptation
FL-ACKF	0.9099	0.3081	0.9607	Fused data, incorrect R_{i0} and adaptation
FL-ADKF	0.8578	0.3071	0.9111	Fused data
FL-AKF 1	0.8537	0.4712	0.9751	Incorrect R_{10} and adaptation
FL-AKF 2	0.7019	0.3467	0.7829	Incorrect R_{20} and adaptation
FL-AKF 3	2.4420	0.5577	2.5050	Incorrect R_{30} and adaptation
FL-AKF 4	0.8652	0.4612	0.9804	Incorrect R_{40} and adaptation
FL-AFKF	0.9099	0.3081	0.9607	Fused data
FL-AKF 1	0.9787	0.4250	1.0670	Incorrect R_{10} and adaptation
FL-AKF 2	0.8603	0.3985	0.9481	Incorrect R_{20} and adaptation
FL-AKF 3	1.8470	0.4107	1.8920	Incorrect R_{30} and adaptation
FL-AKF 4	1.0140	0.3955	1.0880	Incorrect R_{40} and adaptation

From the analysis of the performances obtained by local filters, the following remarks can be given. As in the previous simulation, only the performance of the local FL-AKF 3, which is processing the faulty data, is degraded in both the FL-AKF-FLA and the FL-ADKF approaches. However, the performances of all local filters are degraded in the FL-AFKF.

The actual position and the fused estimated position of the vehicle obtained with the FL-AKF-FLA approach are shown in figure 6.24. The minor influence of the persistent faults can be noted. In figure 6.25 the actual, measured and estimated position of the vehicle carried out by the local FL-AKF 3 in the FL-AKF-FLA architecture, are shown. Note how the persistent faults, introduced at times t = 20, 40, and 60 sec, strongly affect the estimations made by this local FL-AKF. The way in which the adaptive estimation of the elements in the main diagonal of matrix R_{3k}, FL-AKF 3, FL-AKF-FLA architecture, is affected by the introduction of the faulty data can be seen in figure 6.26. As a result of the presence of the faults, the value of $R_{3k}(1,1)$ is increased at the times when a fault is present. Due to this increment in the measurement noise covariance value for the position data, the degree of confidence value assigned to the state vector estimate carried out by the FL-AKF 3 is reduced. Consequently, its influence on the fused solution is decreased accordingly. The influence of the faulty data on the fused solution obtained with the other hybrid adaptive MSDF approaches is greater. To appreciate this graphically, the actual and fused position estimates performed by the FL-DAKF are shown in figure 6.27.

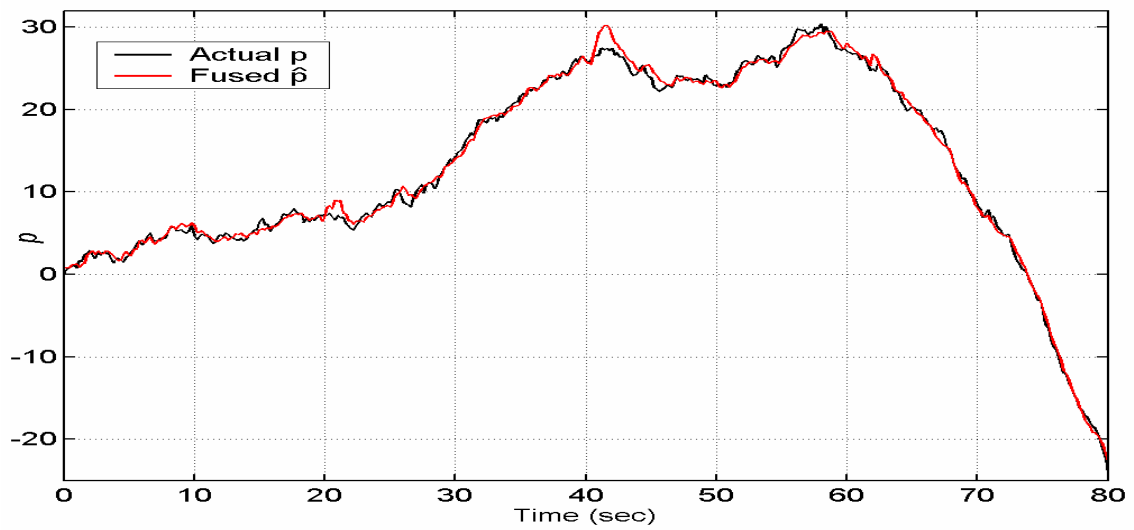


Figure 6.24 Actual position and estimated position of the vehicle obtained with the FL-AKF-FLA architecture, simulation 7.

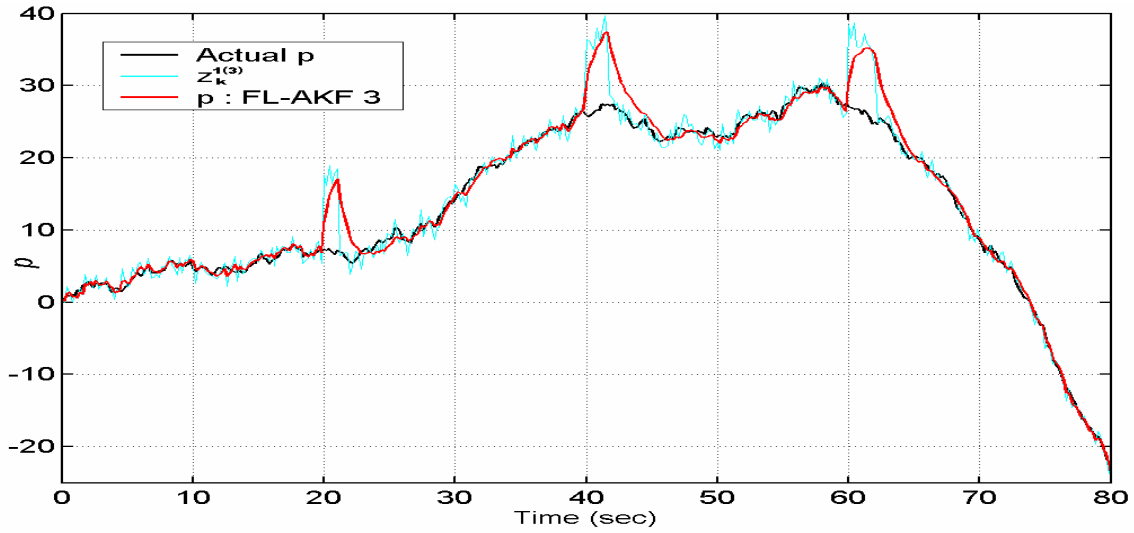


Figure 6.25 Actual position, measured position ($z_k^{1(3)}$), and estimated position of the vehicle obtained with the FL-AKF 3, FL-AKF-FLA, simulation 7.

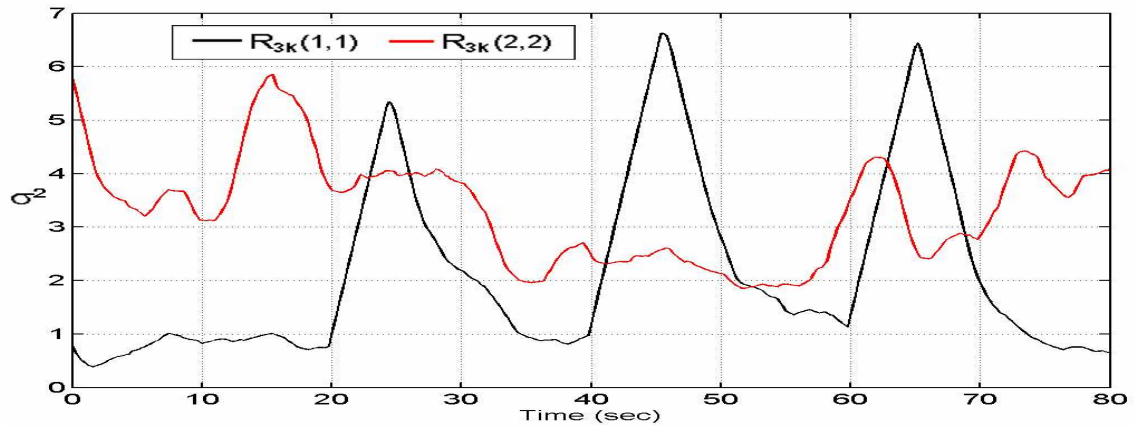


Figure 6.26 Adjustment of the elements in the main diagonal of matrix R_{3k}, FL-AKF 3, FL-AKF-FLA, simulation 7.

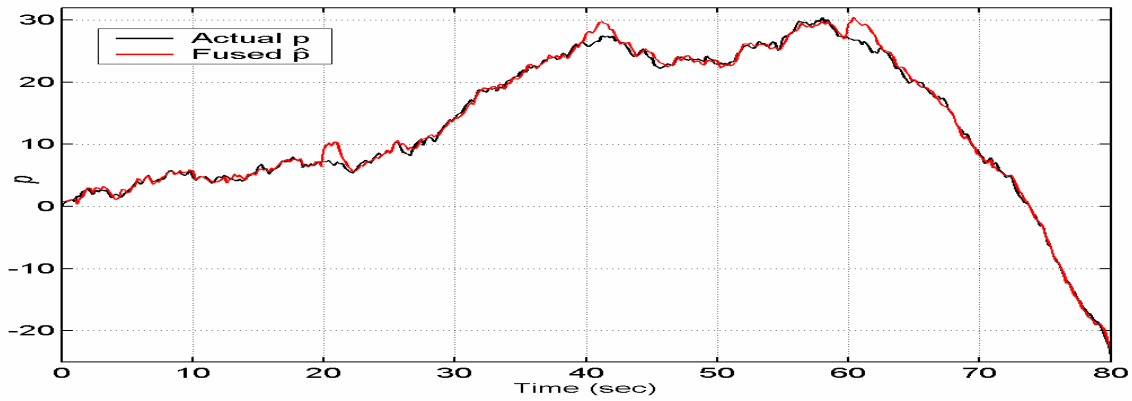


Figure 6.27 Actual position and estimated position of the vehicle obtained with the FL-ADKF, simulation 7.

Simulation 8: The goal of this simulation is to examine the fault-tolerant performances of the hybrid adaptive MSDF approaches when a permanent fault is present in one of the sensors. Therefore, the adaptive MSDF algorithms and the system under consideration were simulated under the same conditions defined in simulation 1. However, in this case a permanent fault is simulated in sensor 3 in each MSDF architecture, at time t = 40 sec, from this time the position value is stuck at a reading of 0 meters.

Results: Table 6.12 shows the obtained performance measures for the local FL-AKFs and the hybrid adaptive MSDF algorithms. From the $RMSE_x$ values of the fused data it is clear that the performances of all the adaptive MSDF algorithms are severely affected by the presence of a permanent fault in sensor 3. In fact, it can be said that none of the MSDF algorithms tolerates efficiently the presence of a permanent fault. Surprisingly, in this case the least affected approaches are the FL-ACKF and the FL-AFKF. It has been demonstrated that the traditional CKF has less fault tolerance capability than the traditional FKF [Gao *et al*, 1993] and, therefore, it was expected to observe the same characteristic in its adaptive counterpart. However, the fault tolerance capability of the FL-ACKF is similar to that observed in the FL-AFKF. This is due to the adaptation carried out in both approaches. Finally, note that the most affected approach is the FL-ADKF, while the FL-AKF-FLA has an intermediate level of affectation.

The actual and the estimated position of the vehicle obtained with the FL-ADKF and each of its local FL-AKFs are shown in figure 6.28. Note the strong effect that the permanent fault has on the fused position estimates. The effect that the permanent fault has on the estimates performed by the local FL-AKF 3 in the FL-ADKF can be seen in figure 6.29, where the actual position, the measured position ($z_k^{1(3)}$), and the estimated position of the vehicle performed by this filter are shown. Note the permanent fault introduced at time t = 40sec.

The strong influence of the fault in the fused estimates performed by the FL-ADKF can be explained by observing the effect that it has on the adjustment of $R_{3k}(1,1)$, which is shown in figure 6.30. After the fault is introduced, an initial constant increment is observed. However, due to the fact that the measured position value gets stuck at 0 meters, and with a zero level of noise, the corresponding noise covariance value is reduced accordingly to match this level of noise, as can be seen in figure 6.30. Obviously, the previous effect reflects the variations observed in the residual sequence, $r_{3k}(1,1)$, shown in figure 6.31, which corresponds to the position measured data, that after a sudden increment goes to zero after the persistent fault is introduced in sensor 3. As consequence of having a measurement noise covariance value of zero for position measurements carried out by the sensor 3, the FL-ADKF fusion algorithm takes into account more strongly the estimates performed by the FL-AKF 3 than the estimates performed by the other filters. Finally, as time progress, the FL-ADKF estimates become those of the zero measurement noise level FL-AKF, this is the number 3, as can be seen in figure 6.28.

Table 6.12 Performance measures: Simulation 7

MSDF Architecture	$RMSE_p$	$RMSE_S$	$RMSE_X$	Conditions
FL-AKF-FLA	7.1360	0.8398	7.1860	Fused data
FL-AKF 1	0.8537	0.4712	0.9751	Incorrect R_{10} and adaptation
FL-AKF 2	0.7019	0.3467	0.7829	Incorrect R_{20} and adaptation
FL-AKF 3	15.1900	1.4980	15.2600	Incorrect R_{30} and adaptation
FL-AKF 4	0.8652	0.4612	0.9804	Incorrect R_{40} and adaptation
FL-ACKF	4.2390	1.4470	4.4790	Fused data, incorrect R_{i0} and adaptation
FL-ADKF	13.0500	1.4470	13.1300	Fused data
FL-AKF 1	0.8537	0.4712	0.9751	Incorrect R_{10} and adaptation
FL-AKF 2	0.7019	0.3467	0.7829	Incorrect R_{20} and adaptation
FL-AKF 3	15.1900	1.4980	15.2600	Incorrect R_{30} and adaptation
FL-AKF 4	0.8652	0.4612	0.9804	Incorrect R_{40} and adaptation
FL-AFKF	4.2390	1.4470	4.4790	Fused data
FL-AKF 1	7.3620	1.4630	7.5060	Incorrect R_{10} and adaptation
FL-AKF 2	3.3990	1.4470	3.6940	Incorrect R_{20} and adaptation
FL-AKF 3	3.0760	1.4360	3.3950	Incorrect R_{30} and adaptation
FL-AKF 4	3.2670	1.4440	3.5720	Incorrect R_{40} and adaptation

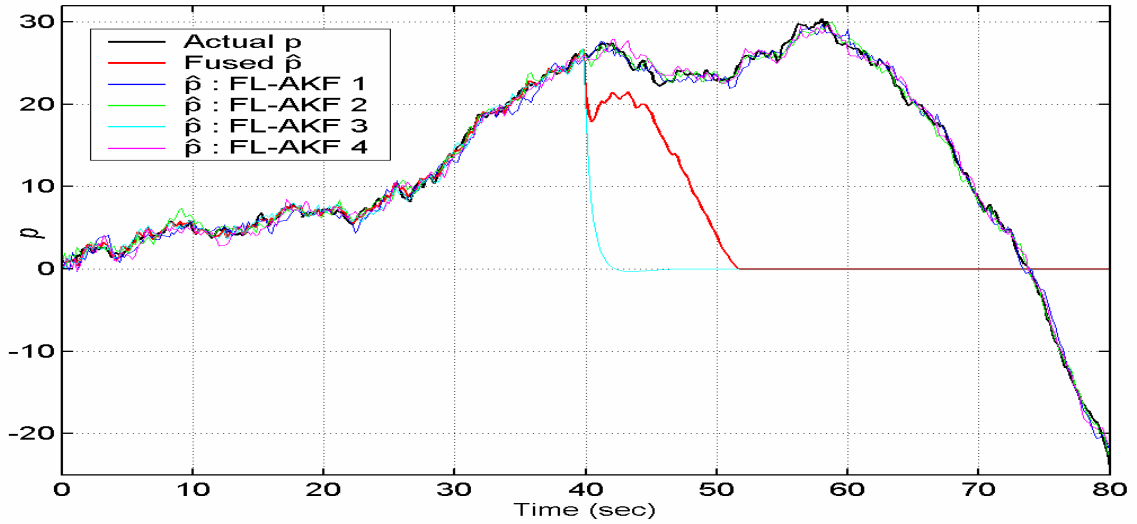


Figure 6.28 Actual and estimated position of the vehicle obtained with the FL-ADKF and each of its local FL-AKFs, simulation 8.

The position estimates performed by the FL-AFKF and each of its local FL-AKFs are shown in figure 6.32. Note that, due to the feedback carried out of the fused state-vector to the local filters, in this case the effect of the permanent fault in the fused position estimates is less drastic than that observed in the FL-ADKF. In the FL-AFKF the fused state estimates try to follow the tendency of the majority of the filters, and in this way the effect of the fault is reduced. However, due to the information sharing carried out, the effect of the fault is transmitted to all the local filters and, as a result, the local estimates are split up from the correct estimates.

In figure 6.33 the actual position, the measured position ($z_k^{1(3)}$), and the estimated position of the vehicle performed by the local FL-AKF 3 in the FL-AFKF are shown. Note the way in which the permanent fault, introduced at time t = 40 sec, affects the estimates performed by this local filter.

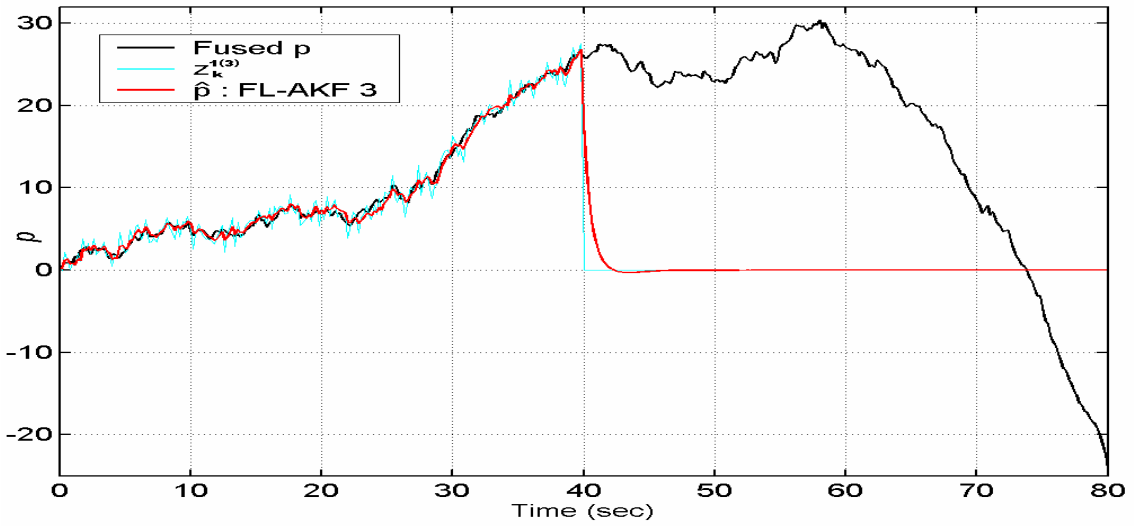


Figure 6.29 Actual position, measured position ($z_k^{1(3)}$), and estimated position of the vehicle obtained with the local FL-AKF 3, FL-ADKF, simulation 8.

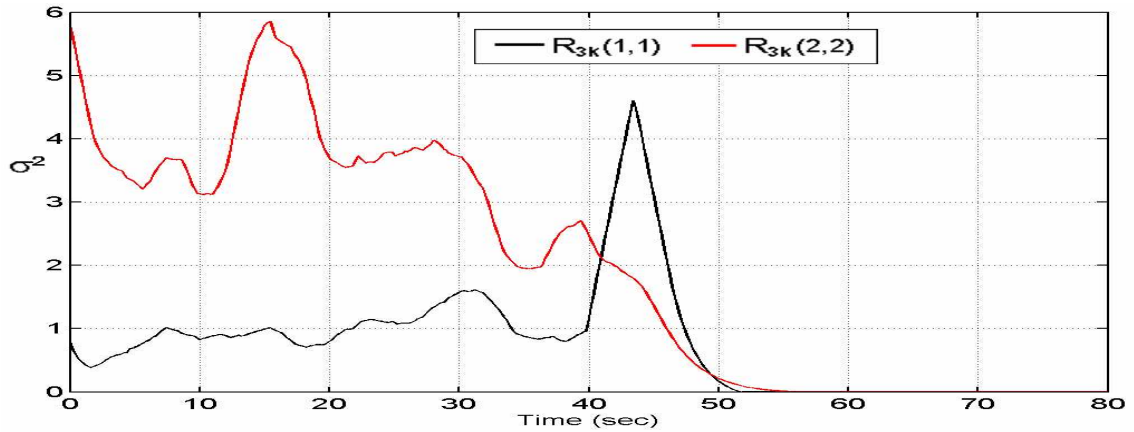


Figure 6.30 Adjustment of the elements in the main diagonal of matrix R_{3k}, FL-AKF 3, FL-ADKF, simulation 8.

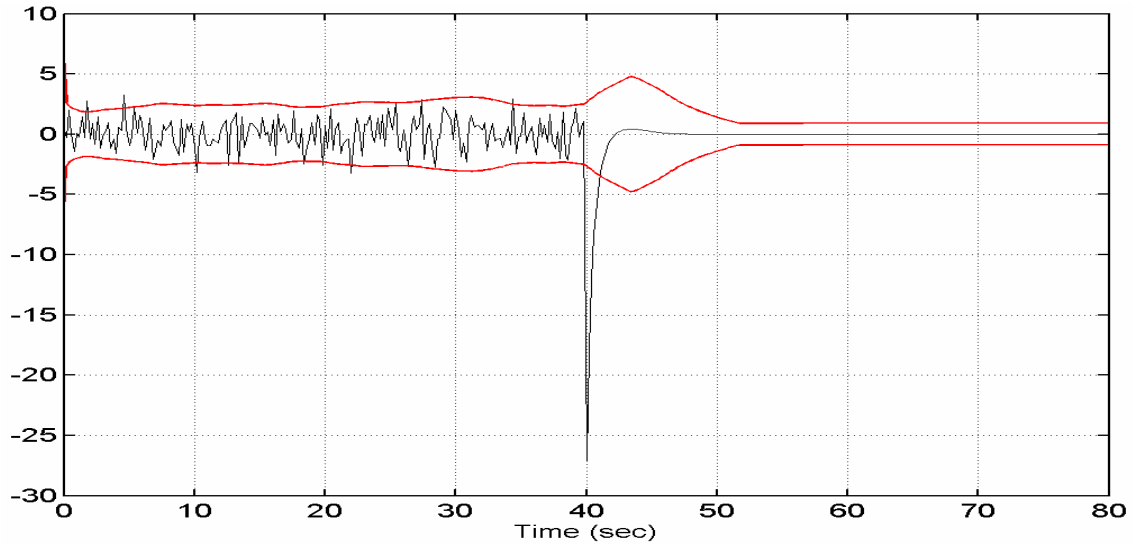


Figure 6.31 Residual sequence $r_{3k}(1,1)$ and its 2σ bounds, FL-AKF 3, FL-ADKF, simulation 8.

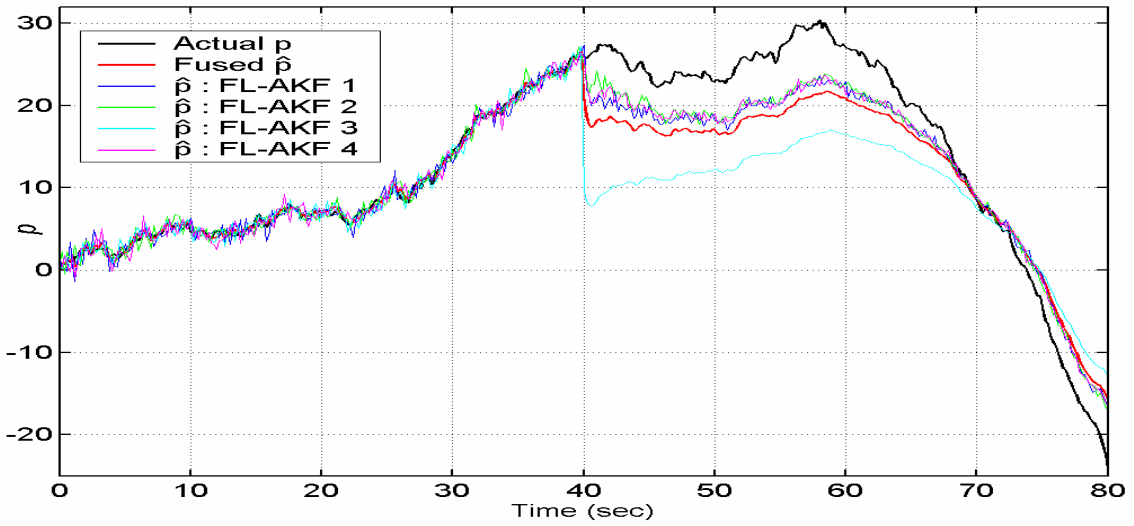


Figure 6.32 Actual and estimated position of the vehicle obtained with the FL-AFKF and each of its local FL-AKFs, simulation 8.

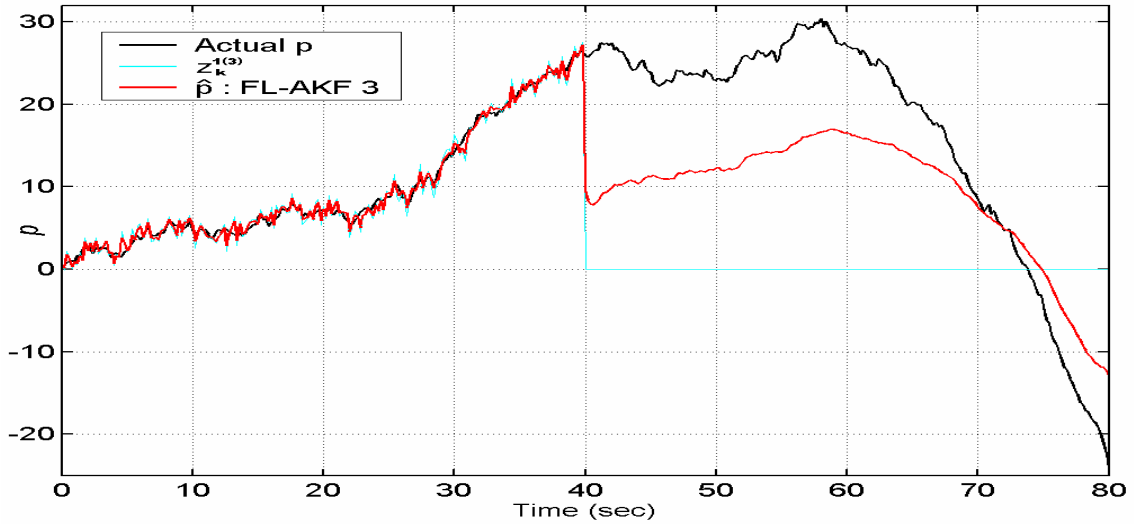


Figure 6.33 Actual position, measured position ($z_k^{1(3)}$), and estimated position of the vehicle obtained with the FL-AKF 3, FL-AFKF, simulation 8.

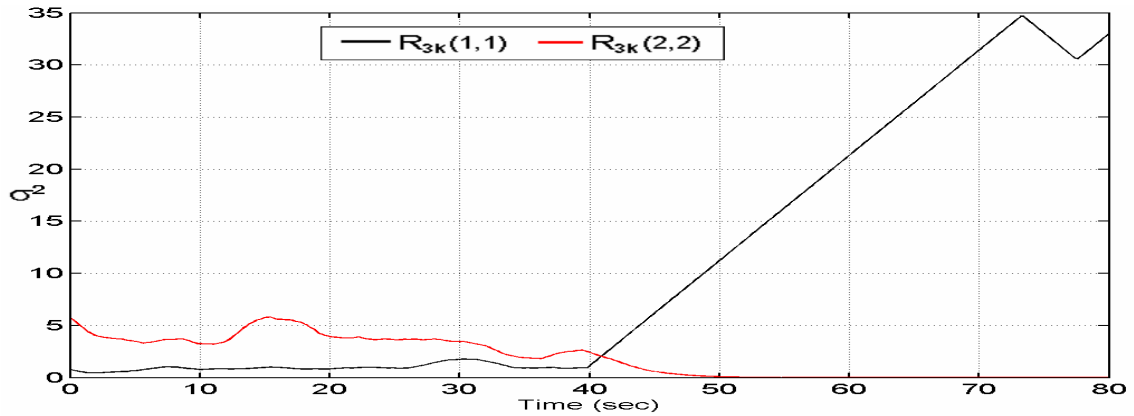


Figure 6.34 Adjustment of the elements in the main diagonal of matrix R_{3k}, local FL-AKF 3, FL-AFKF, simulation 8.

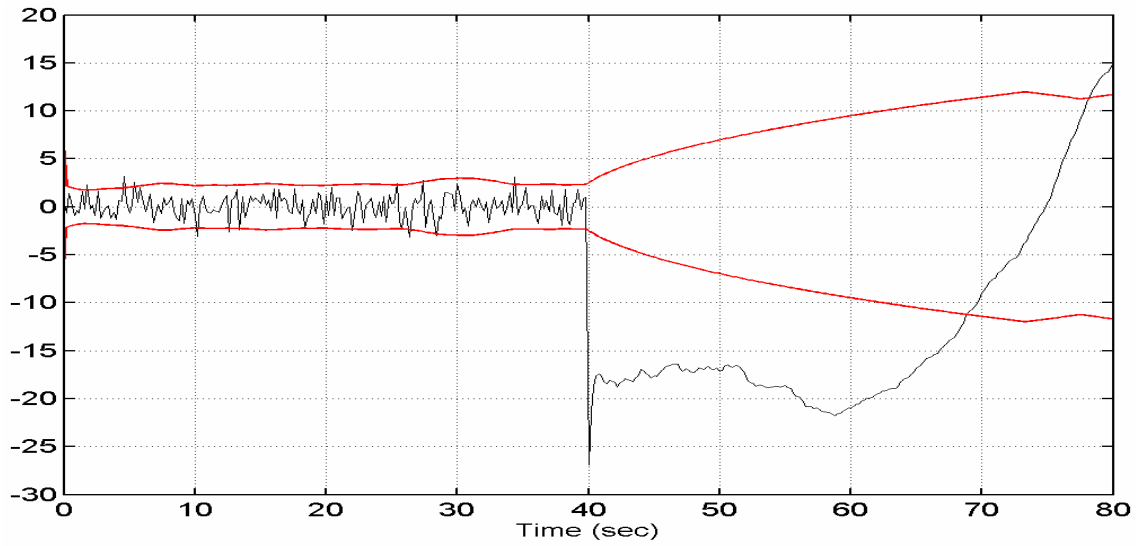


Figure 6.35 Residual sequence $r_{3k}(1,1)$ and its 2σ bounds, FL-AKF 3, FL-AFKF, simulation 8.

The effect that the permanent fault has in the adjustment of the elements in the main diagonal of matrix R_{3k} and the residual sequence $r_{3k}(1,1)$ in the FL-AKF 3 of the FL-AFKF are shown in figures 6.34 and 6.35, respectively. In this case, due to the sharing information principle and the feedback carried out, the effects are different than those observed in the FL-ADKF. In this case $R_{3k}(1,1)$ is constantly increased and $r_{3k}(1,1)$ is maintained far from its 2σ limits.

6.5 Discussion

From a general analysis of the results obtained in the last two sections several remarks can be given:

1) For the cases where faults are not present in the sensors, the fused data obtained with the FL-AKF-FLA architecture is comparatively less accurate than that obtained with the other three proposed MSDF architectures. However, this fused data is more accurate than that obtained by any of its local filters.

2) The role of the FLA in the proposed FL-AKF-FLA approach is of great importance because the fusion of the information is carried out based on the degrees of confidence generated by this component. In addition, only two variables are needed to monitor the performance of each FL-AKF and only nine 'common sense' rules are used in the FIS used in the FLA.

3) A simple FLA-weighting average structure is used to fuse the data in the FL-AKF-FLA architecture. Compared to the other architectures, this makes this structure less computationally demanding.

4) If Gaussian zero-average noise sequences are present in the sensors and the correct measurement noise covariance matrices are specified having the adaptation procedure switched on in the FL-AKFs, then the fused data obtained with the FL-ACKF, the FL-ADKF and the FL-AFKF remain very near to the optimal estimates.

5) If Gaussian zero-average noise sequences are present in the sensors and the incorrect measurement noise covariance matrices are specified having the adaptation procedure switched on in the FL-AKFs, then the measurement noise covariance values are tuned to fit, as closely as

possible, the actual statistics of the noise profiles. As a result of this, the fused data obtained with the hybrid adaptive MSDF architectures are near to the optimal estimates.

6) If there are non-stationary statistics but zero-average noise profile present in any of the sensors, then the adjusting procedure carried out in the FL-AKFs tunes the measurement noise covariance values to fit and follow, as closely as possible, the actual dynamic statistics of the noise profiles. As a result of this, the fused data obtained with all hybrid adaptive MSDF architectures is more accurate than the data obtained with any individual FL-AKF.

7) In the cases where faults are not present in the sensors the most accurate fused data is that obtained with the FL-ADKF, while the least accurate fused data is that obtained with the FL-AKF-FLA architecture. Both the FL-ACKF and the FL-AFKF have an intermediate level of accuracy.

8) If transient or persistent faults are present in any of the sensors then the most fault-tolerant architecture is the FL-AKF-FLA, while the least fault-tolerant architectures are both the FL-ACKF and the FL-AFKF. These last two architectures have similar performances (remember that in this work the sharing factors in the FL-AFKF were set to be equal for all local filters). The FL-ADKF has an intermediate fault-tolerant performance.

9) All architectures do not have good fault-tolerant performances against permanent faults. However, permanent faults are easy to detect by analysing the adjusted measurement noise covariance matrices or the residual sequences. Therefore, several fault detection techniques, e.g. voting systems [Willsky, 1976] and residual-based detection systems [Maybeck, 1979], could be used to detect the faults and implement a fault recovery algorithm. This task is out of the scope of this thesis and is left as a future work.

Therefore, the selection of one of the proposed hybrid adaptive MSDF architectures for a particular application can be made taking into account the remarks given above and the characteristics and objectives followed in the problem at hand. For example, if it is necessary to have fast processing without the requirement of a lot of computational resources, the FL-AKF-FLA approach is adequate for this task. However, if accuracy is the main concern then the FL-ADKF can be applied. If the sensors are subjected to transient or persistent faults, then both the FL-AKF-FLA and the FL-ADKF approaches are adequate. The FL-AFKF appears to be more suitable for fault detection purposes. The FL-ACKF could be applied in cases where there are only two or three sensors and the state vectors are of dimension two or three only. This is because of the computational resources needed to process all the information at the same time, which increases as the number of sensors grows.

6.6 Summary

Four hybrid adaptive MSDF architectures integrating Kalman filtering and fuzzy logic techniques have been presented. These architectures are referred to as: fuzzy logic-based adaptive Kalman filter with fuzzy logic performance assessment scheme (FL-AKF-FLA), fuzzy logic-based adaptive centralised Kalman filter (FL-ACKF), fuzzy logic-based adaptive decentralised Kalman filter (FL-ADKF), and fuzzy logic-based adaptive federated Kalman filter (FL-AFKF). These approaches exploit the advantages that both Kalman filtering and fuzzy logic techniques have: the optimality of the Kalman filter and the capability of fuzzy systems to deal with imprecise information using “common sense” rules. In this approach the linear estimations obtained by individual Kalman filters are improved through dynamically tuning the measurement noise covariance matrix R_k by means of a FIS. This prevents filter divergence and relaxes the a priori assumption about the initial value of R_k. It is particularly relevant that only three rules are needed to carry out the adaptation.

An illustrative example was presented to demonstrate the effectiveness and accuracy of the proposed adaptive MSDF architectures. Exhaustive simulations under different measurement noise conditions and with or without the presence of faults were carried out. The results from the simulations show that the proposed hybrid adaptive MSDF architectures are effective in situations where there are several sensors measuring the same parameters, but each one has different measurement dynamic and noise statistics. Thus, the general idea of exploring the combination of traditional (Kalman filtering) with non-traditional (fuzzy logic) techniques for designing adaptive MSDF architectures appears to be a good avenue of investigation.

The FL-AKF developed in Chapter V together with a neuro-fuzzy approach for non-linear process modelling and identification will be used in the next chapter to design a neuro-fuzzy-AKF state estimator. Then, the hybrid adaptive MSDF architectures developed in this chapter will be applied to merge the information coming from several neuro-fuzzy-AKF state estimators.

CHAPTER 7

HYBRID NEURO-FUZZY-KALMAN FILTER ADAPTIVE MULTI-SENSOR DATA FUSION ARCHITECTURE

7.1 Introduction

As mentioned in Chapter 5, the problem of improving the performance (reliability and accuracy) of the Kalman filter and, in consequence, the MSDF architectures based on it, can be divided into two parts, a modelling problem and an estimation problem. In the previous two chapters, only the estimation problem has been tackled. Thus, in this chapter the modelling problem is considered. In that sense, neuro-fuzzy techniques, namely those described in Chapter 3, will be used for modelling and identification purposes.

In the hybrid adaptive MSDF architectures developed in Chapter 6 it is assumed that a model of the system under consideration is available in a state-space representation. However, what happens if this model is not available, or if the system under consideration is a non-linear one? In these cases, the developed architectures cannot be applied. Nevertheless, if there is a tool through which the system under consideration can be modelled and expressed in the form needed, then the proposed MSDF architectures can be applied.

Therefore, in order to deal with the above problem, research has been carried out in the area of identification and modelling of non-linear systems. The objective of this research was to find a suitable neuro-fuzzy approach capable of modelling and expressing in a state-space representation the system under consideration. In addition, the identification process should be carried out using solely the data coming from the sensors, which can be of a different kind but should be commensurate. In this way, the FL-AKF algorithm developed in Chapter 5 can be directly applied and, in consequence, the hybrid adaptive MSDF architectures proposed in Chapter 6 can be employed to merge the data coming from the identified system.

As a result of the research carried out, it was found that the neuro-fuzzy-SKF state estimator recently developed by Harris *et al* [1999, 2000, 2002] [Wu and Harris, 1997] is adequate for the purposes followed in this chapter. Therefore, first in this chapter the neuro-fuzzy-SKF state estimator approach is described. After that, a simplified version of it is proposed. Then, the simplified version of the neuro-fuzzy-SKF state estimator is used to develop a novel hybrid neuro-fuzzy-AKF state estimator. Finally, the FL-AKF-FLA hybrid adaptive MSDF architecture proposed in Chapter 6 is used to merge the data coming from several neuro-fuzzy-AKF state estimators.

7.2 The neuro-fuzzy-SKF state estimator

Consider a general stochastic non-linear single-input-single-output (SISO) system represented by the discrete-time domain model:

$$y(t) = f\left(y(t-1),\ldots,y(t-n_y),u(t-d-1),\ldots,u(t-d-n_u)\right) + w(t) \qquad (7.1),$$

where $f(\cdot)$ is an unknown non-linear function, $u(t)$ and $y(t)$ are the system's input and output, respectively; n_y, n_u, and d are positive integers assumed *a priori* and representing the orders and

time-delay of the model; w(t) denotes a white noise sequence representing observational and modelling errors.

In order to perform state estimates by applying Kalman filtering to the system (7.1) two alternatives can be devised. In the first, after the identification and linearisation of the unknown plant has been performed, an extended Kalman filter [Brown and Hwang, 1997] can be applied. However, the resulting extended Kalman filter is non-optimal due to the linear approximation; moreover, convergence cannot be guaranteed, and even divergence may occur [Wu and Harris, 1996]. The second option is to look for another model of (7.1) that is easy to convert to an equivalent state-space form and then use a SKF. This second option is the one selected in the neuro-fuzzy-SKF design.

Hence, the non-linear system described by (7.1) can be re-expressed as:

$$\begin{aligned} y(t) &= a_1(O_t)y(t-1) + \cdots + a_{n_y}(O_t)y(t-n_y) \\ &\quad + a_{n_y+1}(O_t)u(t-d-1) + \cdots + a_n(O_t)u(t-d-n_u) + w(t) \end{aligned} \tag{7.2}$$

where $a_i(O_t)$ $i = 1,\ldots,n$, $n = n_y + n_u$, are the unknown parameters which are functions of the measurable multi-dimensional operating points O_t. O_t may be some changing environmental operating condition that causes the system's parameters to vary. The system (7.2) is an operating point dependent auto-regressive moving average (ARMA) model where the AR parameters are non-linear functions of the operating point O_t [Wu and Harris, 1996]. Also (7.2) is a special case of (7.1) when the measurable operation points O_t depend upon the past values of the system input and output [Wang *et al*, 1996a].

The model described by (7.2) is easy to convert to a state space representation. This can be achieved by considering the operating point O_t as a function of time. In such a case, (7.2) can be re-expressed in the following time-varying ARMA form:

$$\begin{aligned} y(t) &= a_1(t)y(t-1) + \cdots + a_{n_y}(t)y(t-n_y) \\ &\quad + a_{n_y+1}(t)u(t-d-1) + \cdots + a_n(t)u(t-d-n_u) + w(t) \end{aligned} \tag{7.3}$$

where $a_i(t)$ $i = 1,\ldots,n$, $n = n_y + n_u$, are the time-varying parameters. The system (7.3) can be represented in various state-space realisations, but the controllable state-space form is considered in the neuro-fuzzy-SKF as will be explained later.

The time-varying parameters, $a_i(t)$, in (7.3) can be approximated by using a neuro-fuzzy modelling network as is explained next. From (7.3) a vector of observations can be defined as:

$$x(t) = [x_1,\ldots,x_n]^{\mathrm{T}} = [y(t-1),\cdots,y(t-n_y),u(t-d-1),\ldots,u(t-d-n_u)]^{\mathrm{T}} \tag{7.4},$$

where $n = n_y + n_u$. Based on (7.4) a Sugeno-type FIS [Takagi and Sugeno, 1985] can be designed in order to approximate the system given in (7.3) [Wu and Harris, 1997]. Consider as the FIS input the vector of observations $x(t) \in \Re^n$, and as the FIS output the system value $y(t) \in \Re$. Therefore, universes of discourse for each linguistic variable x_i can be defined as $X_i \subset \Re$, and for y as $Y \subset \Re$ ($i = 1,2,\ldots,n$). Define fuzzy sets $A_i^{k_i}$, with $k_i = 1,2,\ldots,m_i$ and $i = 1,2,\ldots,n$, using B-spline functions, as values of the linguistic variables x_i. Then, fuzzy rules forming a complete rule base can be defined as:

$$j\text{-th rule: if } x_1 \text{ is } A_1^{k_1} \text{ and } x_2 \text{ is } A_2^{k_2} \text{ and } \cdots \; x_n \text{ is } A_n^{k_n} \text{ then } y(t) \text{ is } y_j(x) \tag{7.5}$$

where $y_j(x) \in Y$ and $j = 1,2,\ldots,p$; p is the number of fuzzy rules, which in order to have a complete rule base must satisfy $p = m_1 m_2 \cdots m_n$. Generally, for a task of system identification, the function $y_j(x)$ in (7.5) is selected to be a linear combination of the components of the input vector [Takagi and Sugeno, 1985], this is:

$$y_j(x) = a_1^j x_1 + a_2^j x_2 + \cdots + a_n^j x_n \tag{7.6}$$

where a_i^j ($i = 1,2,\ldots,n$; $j = 1,2,\ldots,p$) are the unknown parameters.

The FIS defined by (7.5) and (7.6) form a local linear model of (7.3) [Harris *et al*, 1999]. If the algebraic product/sum fuzzy operators (see Appendix A) are selected, B-spline functions are used to define the membership functions of the fuzzy sets $A_i^{k_i}$, and the COA defuzzification method is used, then the real (crisp) output of the FIS is given by:

$$\begin{aligned} y(t) &= \sum_{j=1}^{p} \prod_{i=1}^{n} A_i^{k_i}(x_i) y_j(x) \\ &= \sum_{j=1}^{p} \mu_j(x) y_j(x) \end{aligned} \tag{7.7}$$

where $\mu_j(x)$ represents the degree of truth value of the antecedent part of the j-th rule, and is given by:

$$\mu_j(x) = \prod_{i=1}^{n} A_i^{k_i}(x_i) \tag{7.8}$$

Equation (7.8) also can be seen as a multivariate B-spline basis function generated by multiplying the n univariate basis functions $A_i^{k_i}(x_i)$. A graphical representation of the FIS defined by the rule base (7.5) and equation (7.7) is shown in figure 7.1. This system can be seen either as a B-spline neural network or as a Sugeno-type FIS with membership functions implemented by B-spline basis functions [Wu and Harris, 1997]. A broader explanation of this special class of FIS, which characteristics makes it a hybrid neuro-fuzzy system, was given in Chapter 3, section 3.5.

The above described FIS can be reorganised to form a neuro-fuzzy modelling network [Wu and Harris, 1997] [Harris *et al*, 1999]. By substituting $y_j(x)$ defined with (7.6) into (7.7), it results in:

$$\begin{aligned} y(t) &= \sum_{j=1}^{p} \mu_j(x)(a_1^j x_1 + a_2^j x_2 + \cdots + a_n^j x_n) \\ &= \left(\sum_{j=1}^{p} \mu_j a_1^j \right) x_1 + \left(\sum_{j=1}^{p} \mu_j a_2^j \right) x_2 + \cdots + \left(\sum_{j=1}^{p} \mu_j a_n^j \right) x_n \\ &= a_1(t) y(t-1) + \cdots + a_{n_y}(t) y(t-n_y) + a_{n_y+1}(t) u(t-d-1) + \cdots + a_n(t) y(t-d-n_u) \end{aligned} \tag{7.9}$$

where:

$$a_i(t) = \sum_{j=1}^{p} \mu_j(x) a_i^j \ , \ i=1,2,\ldots,n \tag{7.10}$$

is a linear combination of the degree of truth values of the antecedent part of the fuzzy rules. Note that (7.10) is similar to (7.7) where instead of having function factors $y_j(x)$ there are singleton factors a_i^j. This means that each i-th parameter $a_i(t)$ can be approximated by a Sugeno-type FIS designed as was explained previously, but where the consequent parts of the fuzzy rules are defined using singleton fuzzy sets a_i^j. One of these FIS is represented graphically in figure 7.2. Therefore, (7.9) is equivalent to (7.3).

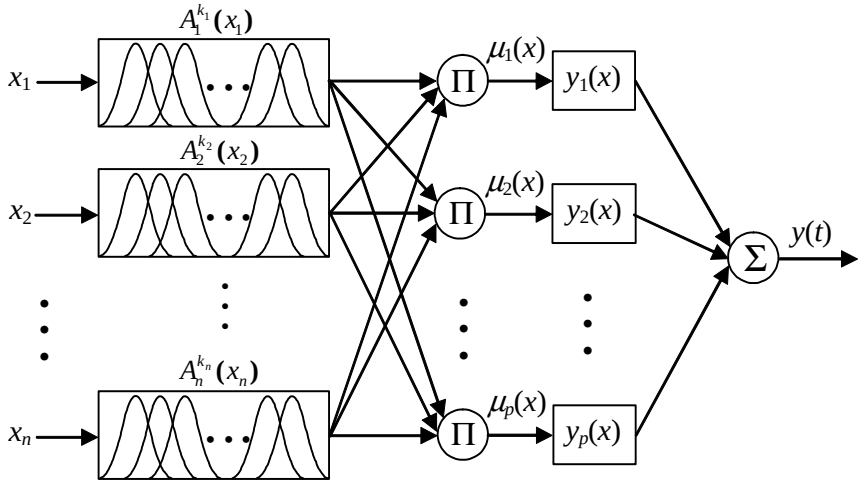


Figure 7.1 Graphical representation of the FIS defined by the rule base (7.5) and equation (7.7).

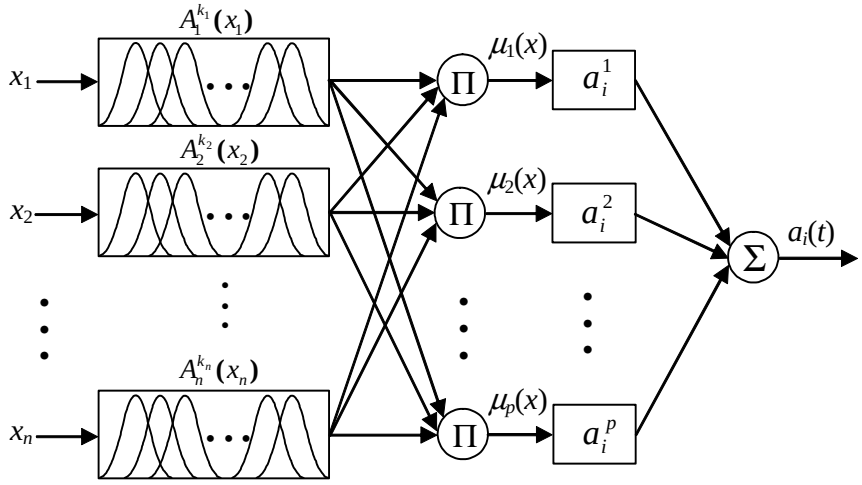


Figura 7.2 FIS used to approximate the parameters $a_i(t)$.

If each auto-regressive parameter $a_i(t)$ in (7.9) is approximated by a FIS of the kind shown in figure 7.2, then a neuro-fuzzy modelling network representing the whole equation can be formed as is shown in figure 7.3.

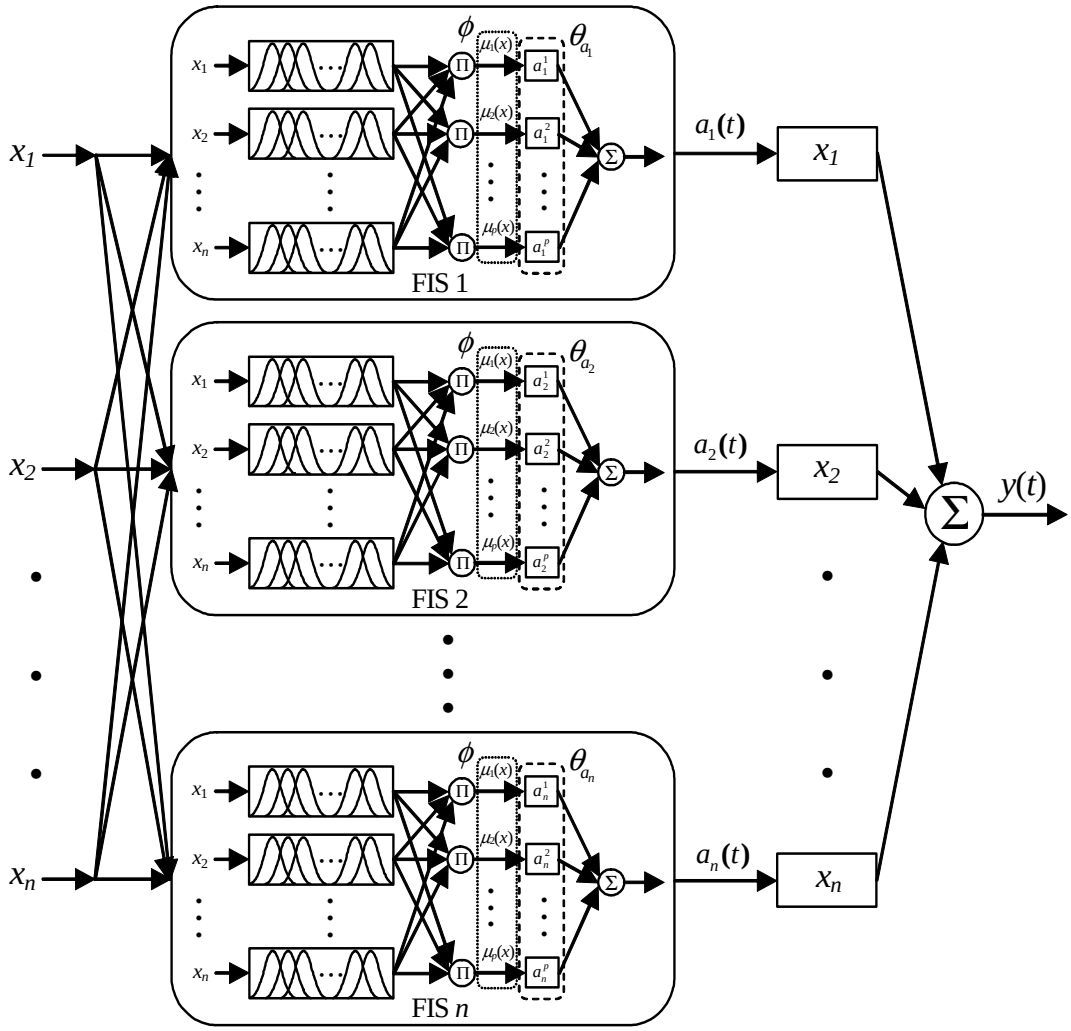


Figura 7.3 Neuro-fuzzy modelling network.

The first layer of the neuro-fuzzy modelling network in figure 7.3 is composed of *n* FISs, where each FIS output corresponds to an auto-regressive parameter $a_i(t)$. All the FISs have as input the vector *x* and use the same fuzzy rule base. Consequently, all the FISs share the same vector ϕ, which is formed by the degree of truth values, $\mu_i(x)$, of the antecedent part of the fuzzy rules:

$$\phi = [\mu_1(x), \mu_2(x), \cdots, \mu_p(x)]^T \tag{7.11}.$$

The free parameters of the network in figure 7.3 are the singletons (scalars) a_i^j, that define the consequent part of the rules in each FIS. These singletons can be arranged in *n* vectors θ_{a_i}:

$$\theta_{a_i} = [a_i^1, a_i^2, \cdots, a_i^p]^T, \quad i = 1,2,\ldots,n \tag{7.12},$$

where each vector θ_{a_i} contains the singletons corresponding to the consequent parts of the rules of the *i*-th FIS.

The neuro-fuzzy modelling network can be trained using the LMS or the NLMS algorithms, as will be shown later. Therefore, a non-linear system given in the form (7.9) can be identified by the neuro-fuzzy modelling network shown in figure 7.3.

The model (7.9) can be easily translated to a state-space representation [Harris *et al*, 1997] as is required by the SKF algorithm. If the term corresponding to the input in (7.9) is represented by,

$$\tilde{u}(t) = a_{n_y+1}(t)u(t-d-1) + \cdots + a_n u(t-d-n_u) \tag{7.13},$$

then the model (7.9) becomes:

$$y(t) = a_1(x)y(t-1) + \cdots + a_{n_y}(x)y(t-n_y) + \tilde{u}(t) + w(t) \tag{7.14},$$

Now, define:

$$z(t) = \begin{bmatrix} z_1(t) = y(t-n_y), \\ z_2(t) = y(t-n_y+1) \\ \vdots \\ z_{m-1}(t) = y(t-2) \\ z_m(t) = y(t-1) \end{bmatrix} \tag{7.15}.$$

It follows that:

$$\left.\begin{aligned} z_1(t+1) &= y(t-n_y+1) = z_2(t), \\ z_2(t+1) &= y(t-n_y+2) = z_3(t), \\ &\vdots \\ z_{m-1}(t+1) &= y(t-1) = z_m(t), \\ z_m(t+1) &= y(t) = a_1(t)y(t-1) + \cdots + a_{n_y}(t)y(t-n_y) + \tilde{u}(t) + w(t) \\ &= a_{n_y}(t)z_1(t) + \cdots + a_1(t)z_m(t) + \tilde{u}(t) + w(t) \end{aligned}\right\} \tag{7.16},$$

this leads to the following canonical controllable state-space representation:

$$z(t+1) = A(t)z(t) + B\tilde{u}(t) + \Gamma w(t),\ z(0) = \bar{z}_0 \tag{7.17}$$

$$y(t) = C(t)z(t) + D\tilde{u}(t) + v(t) \tag{7.18}$$

with:

$$A(t) = \begin{bmatrix} 0 & 1 & & & \\ 0 & 0 & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ 0 & \cdots & \cdots & & 1 \\ a_{n_y}(t) & \cdots & & a_2(t) & a_1(t) \end{bmatrix} \in \Re^{n_y \times n_y},\ B = [0 \quad \cdots \quad 1]^T \in \Re^{n_y}$$

$$\Gamma = [0 \quad \cdots \quad 1]^T \in \Re^{n_y},\ C(t) = [a_{n_y}(t) \quad \cdots \quad a_2(t) \quad a_1(t)] \in \Re^{n_y},\ \text{and } D = 1.$$

where the mean of $z(0)$ is the initial condition, given as $\bar{z}_0$; $v(t)$ represents measurement noise. It is assumed that $v(t)$ is modelled as a Gaussian zero mean white noise sequence.

Therefore, by using (7.17) and (7.18) the SKF can be directly applied:

i) Time update (or prediction) equations:

$$\hat{z}(t+1)_{(-)} = A(t)\hat{z}(t)_{(+)} + B\tilde{u}(t) \tag{7.19}$$
$$P(t+1)_{(-)} = A(t)P(t)_{(+)}A^T(t) + \Gamma Q(t)\Gamma^T \tag{7.20}$$

ii) Measurement update (or correction) equations:

$$K(t) = P(t)_{(-)}C^T(t)[C(t)P(t)_{(-)}C^T(t) + R(t)]^{-1} \tag{7.21}$$
$$\hat{z}(t)_{(+)} = \hat{z}(t)_{(-)} + K(t)[y(t) - C(t)\hat{z}(t)_{(-)} - D\tilde{u}(t)] \tag{7.22}$$
$$P(t)_{(+)} = [I - K(t)C(t)]P(t)_{(-)} \tag{7.23}$$
$$\hat{y}(t) = C(t)\hat{z}(t)_{(+)} + D\tilde{u}(t) \tag{7.24}$$

where $\hat{y}(t)$ is the filtered system output. Note that there is a slight modification in the notation of the SKF with respect to that used in previous chapters. This has been made to indicate the time-varying condition of the different matrixes included in the state space model.

Hence, a neuro-fuzzy-SKF structure can be used to produce state estimates. This structure can be arranged in two ways [Wu and Harris, 1997] [Harris *et al*, 1999]: (i) the indirect neuro-fuzzy-SKF state estimation scheme, and (ii) the direct neuro-fuzzy-SKF state estimation scheme. Both structures are described as follows.

(i) *The indirect neuro-fuzzy-SKF state estimation scheme*. In this scheme the system identification and the state estimation by the SKF are performed separately, as is represented graphically in figure 7.4. First, the neuro-fuzzy network is used to identify the non-linear system model. Once the system model has been identified, it is fed to a separate SKF to perform state estimates indirectly.

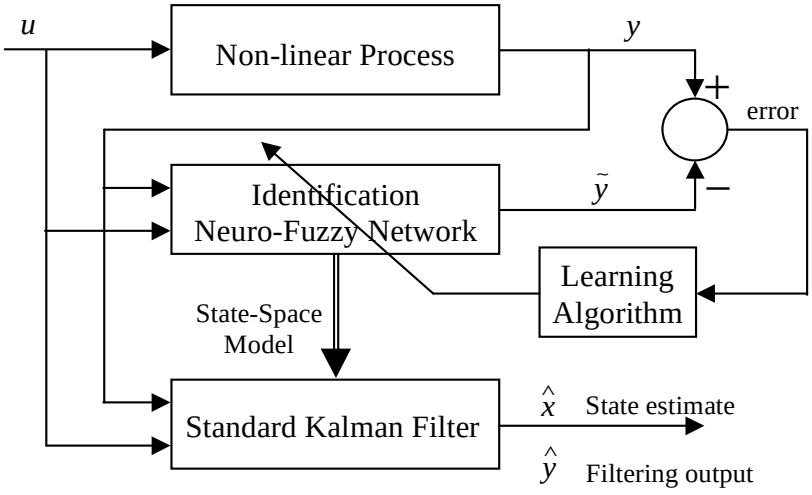


Figure 7.4 Indirect neuro-fuzzy-SKF state estimation scheme.

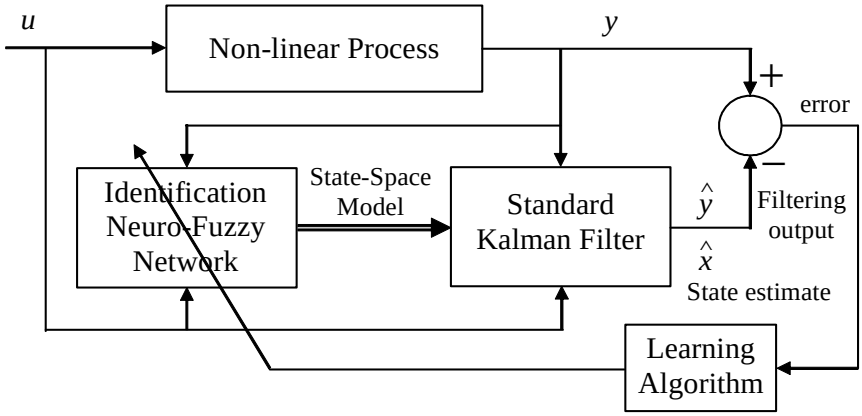


Figura 7.5 Direct neuro-fuzzy-SKF state estimation scheme.

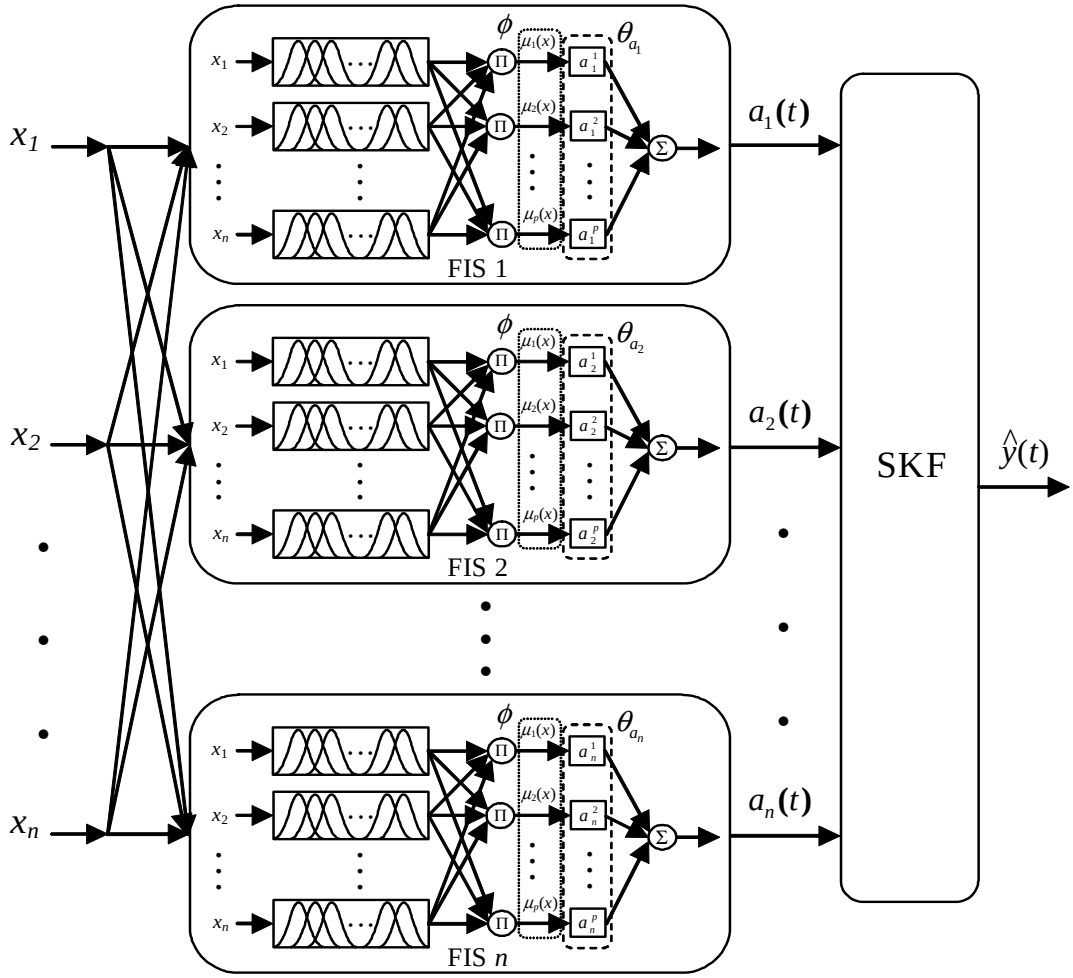


Figure 7.6 Connection between the neuro-fuzzy network of figure 7.3 and a SKF.

(ii) *The direct neuro-fuzzy-SKF state estimation scheme*. Here, the identification process and the state estimation by the SKF algorithm are combined in a bootstrap scheme, as is shown in figure 7.5, to produce state estimates directly. The connection between the neuro-fuzzy network of figure 7.3 and a SKF is implemented as is shown in figure 7.6. This structure can be viewed as a two-layered adaptive network, where the first layer is the neuro-fuzzy network shown in figure 7.3 and the second layer is a SKF which uses the auto-regressive parameters produced by the first layer to perform state estimates. Evidently, the SKF cannot be interpreted as a neural

network, thus it is not explicitly seen how the training procedure of this structure can be carried out. This matter will be clarified in the following section.

7.2.1 Training of the neuro-fuzzy-SKF state estimator

In this section the training algorithms for the two neuro-fuzzy-SKF state estimation schemes, presented in the last section, are described.

a) *Training of the indirect neuro-fuzzy-SKF state estimation scheme*. The neuro-fuzzy modelling network shown in figure 7.3 is a two-layered neural network. The first layer is composed of *n* FISs that also can be viewed as B-spline neuro-fuzzy sub-networks, whose outputs correspond to the auto-regressive parameters $a_i(t)$. The second layer is simply the regression calculation of (7.9) [Wu and Harris, 1997]. Therefore, the free parameters of the network are the singletons a_i^j (the weights of the B-spline neuro-fuzzy sub-networks), which define the consequent part of the fuzzy rules in the first layer. The weights of the second layer can be considered as fixed in each iteration of the training procedure. Several training algorithms for traditional feedforward neural networks can be used to train the neuro-fuzzy-SKF network, but the NLMS, which was reviewed in section 3.5, is employed here.

At time *t* the input

$$x(t) = [x_1, \ldots, x_n]^{\mathrm{T}} = [y(t-1), \cdots, y(t-n_y), u(t-d-1), \ldots, u(t-d-n_u)]^{\mathrm{T}} \in \Re^n,$$

is presented to the neuro-fuzzy modelling network of figure 7.3. In the forward pass the network calculates the output by (7.9), which is denoted by $\tilde{y}$ (see figure 7.4). Thus an error signal, $\varepsilon(t)$, may be defined as:

$$\varepsilon(t) = y(t) - \tilde{y}(t) \tag{7.25}.$$

The error signal needs to be propagated back through the network. But, because there are no free parameters in the second layer of the network, the error is propagated back through the second layer to the output of the first layer. Thus, the errors in $a_i(t)$, normalised by $x(t)^{\mathrm{T}}x(t)$, are given by,

$$\varepsilon_{a_i}(t) = \frac{x_i(t)\varepsilon(t)}{x(t)^T x(t)}, \quad i = 1,2,\ldots,n \tag{7.26}.$$

Then, the errors ε_{a_i} can be used to update the free parameters ($\theta_{a_i}(t)$) of the first layer. Therefore, the NLMS algorithm for the network is:

$$\theta_{a_i}(t) = \theta_{a_i}(t-1) + \eta \frac{\phi(t-1)x_i(t)\varepsilon(t)}{c + \phi(t-1)^T \phi(t-1)x(t)^T x(t)} \tag{7.27}$$

with $\theta_{a_i}(0)$ given, $0 < \eta < 2$ is the learning rate, and $c > 0$ is an arbitrarily small number which is added to avoid division by zero.

Proof of the convergence of the described training algorithm can be found in [Wang *et al*, 1996a].

b) *Training of the direct neuro-fuzzy-SKF state estimation scheme.* The neuro-fuzzy-SKF estimator shown in figure 7.6 can be viewed as a two-layered neural network. The first layer is composed of *n* FISs, whose outputs correspond to the auto-regressive parameters $a_i(t)$. The second layer corresponds to a SKF, which does not need to be trained. Nevertheless the SKF is not a neural network and information cannot pass through it inversely, the NLMS training algorithm can be used to train the network as is described as follows.

Using the definition of states given by (7.15), the state estimate $\hat{z}(t)$ is:

$$\hat{z}(t) = [\hat{y}(t-n_y)\cdots\hat{y}(t-1)]^T \tag{7.28}.$$

Substituting (7.28) and (7.13) in (7.26) results in the following regressive relation:

$$\hat{y}(t) = a_1(t)\hat{y}(t-1)+\cdots+a_{n_y}(t)\hat{y}(t-n_y)+a_{n_y+1}(t)u(t-d-1)+\cdots+a_n(t)y(t-d-n_u) \tag{7.29}.$$

Equation (7.29) can be represented as a two-layered neural network, as is shown in figure (7.7). This means that it is possible to train the neural network using the NLMS algorithm as in the previous case. However, a slight modification needs to be performed. In the forward pass the network calculates the auto-regressive parameters $a_i(t)$, which are the outputs of the first layer, according to (7.10). Then, the state-space equations (7.17) and (7.18) are formed with these parameters and the state estimation is performed using the SKF algorithm, equations (7.19)-(7.24), giving the state estimate $\hat{z}(t)$ and output estimate $\hat{y}(t)$.

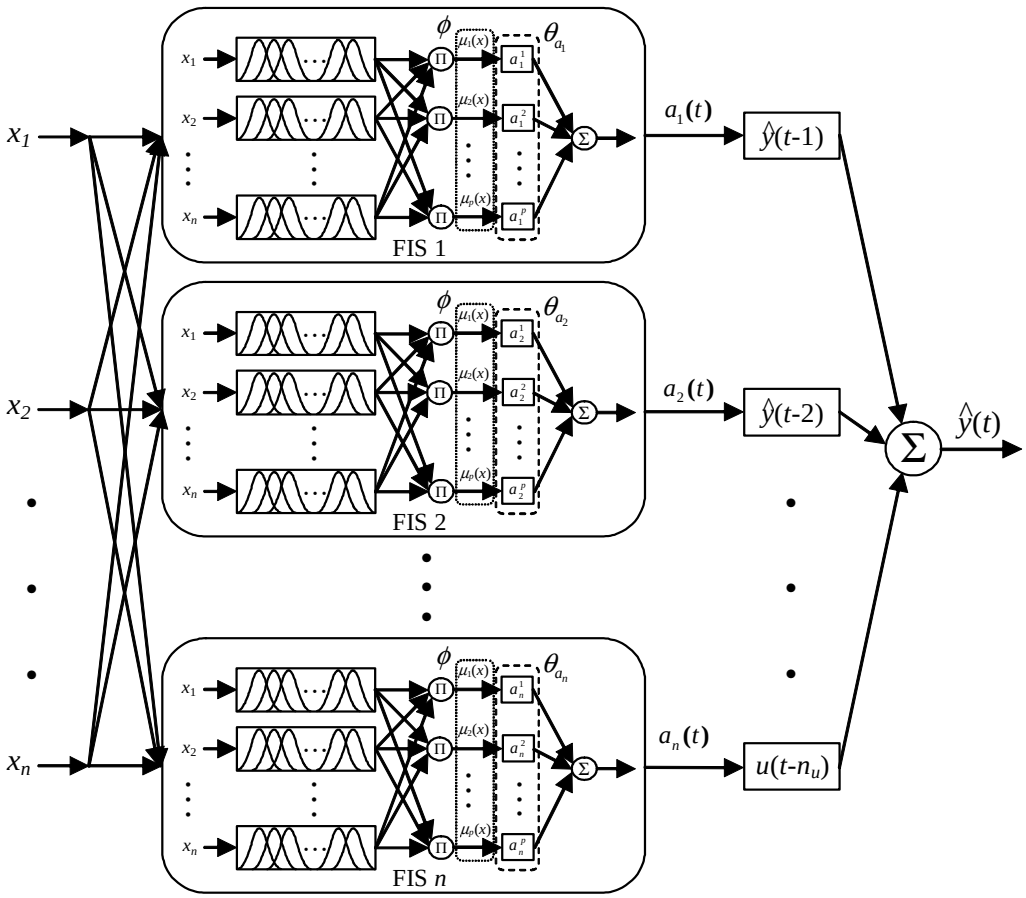


Figure 7.7 Neuro-fuzzy network representing equation (7.29).

Therefore, the NLMS training algorithm is given by:

$$\theta_{a_i}(t) = \theta_{a_i}(t-1) + \eta \frac{\phi(t-1)\xi_i(t)\varepsilon(t)}{c + \phi(t-1)^T \phi(t-1)\xi(t)^T \xi(t)} \tag{7.30}$$

where $\varepsilon(t)$ and $\xi(t)$ are defined as,

$$\varepsilon(t) = y(t) - \hat{y}(t) \tag{7.31}$$

$$\begin{aligned}\xi(t) &= [\xi_1(t) \quad \xi_2(t) \quad \cdots \quad \xi_n(t)] \\ &= [\hat{y}(t-1) \quad \cdots \quad \hat{y}(t-n_y) \quad u(t-d-1) \quad \cdots \quad u(t-d-n_u)] \in \Re^n\end{aligned} \tag{7.32}$$

and $\hat{y}(t)$ is the filter output at time t.

7.3 The simplified neuro-fuzzy-SKF state estimator

In the last section the neuro-fuzzy-SKF state estimator was described as originally proposed by Harris *et al* [1997, 1999, 2000]. However, from an analysis of the neuro-fuzzy-SKF state estimator, it is deduced that a simplified version of it can be obtained as is presented in this section.

The auto-regressive parameters $a_i(t)$ in (7.9) are non-linear functions which are approximated by FISs in the first layer of the neuro-fuzzy modelling network shown in figure 7.3. Note that all these FISs share the same vector ϕ, formed with the degree of truth values $\mu_i(x)$ of the antecedent parts of the fuzzy rules (see (7.11)). Therefore, instead of considering n complete FISs, the neuro-fuzzy network can be built considering a single antecedent rule evaluator in which the calculation of the degree of truth values $\mu_i(x)$ is performed. Then, these values are distributed among n vector blocks θ_{a_i}, which constitute the consequent parts of the n rule sets, to obtain the corresponding $a_i(t)$ parameters. This simplifies the neuro-fuzzy network structure as is shown in figure 7.8.

From the simplified neuro-fuzzy network structure of figure 7.8 the following equations can be deduced:

$$a_i(t) = \phi^T \theta_{a_i}, \quad i = 1,2,\ldots,n \tag{7.33}$$

$$\begin{aligned}y(t) &= \sum_{i=1}^{n} \phi^T \theta_{a_i} x_i = \sum_{i=1}^{n} a_i(t) x_i \\ &= a_1(t)y(t-1) + \cdots + a_{n_y}(t)y(t-n_y) + a_{n_y+1}(t)u(t-d-1) + \cdots + a_n(t)y(t-d-n_u)\end{aligned} \tag{7.34}$$

(7.34) is equivalent to (7.9) and proves that the final result obtained with the simplified neuro-fuzzy modelling network is equal to the result obtained with the original neuro-fuzzy modelling network.

Alternatively, instead of arranging the free parameters of the simplified neuro-fuzzy network (or the original neuro-fuzzy network) in vectors θ_{a_i}, these can be arranged in vectors of the form:

$$a^j = [a_1^j \quad a_2^j \quad \cdots \quad a_n^j], \quad j = 1,2,\ldots,p \tag{7.35}$$

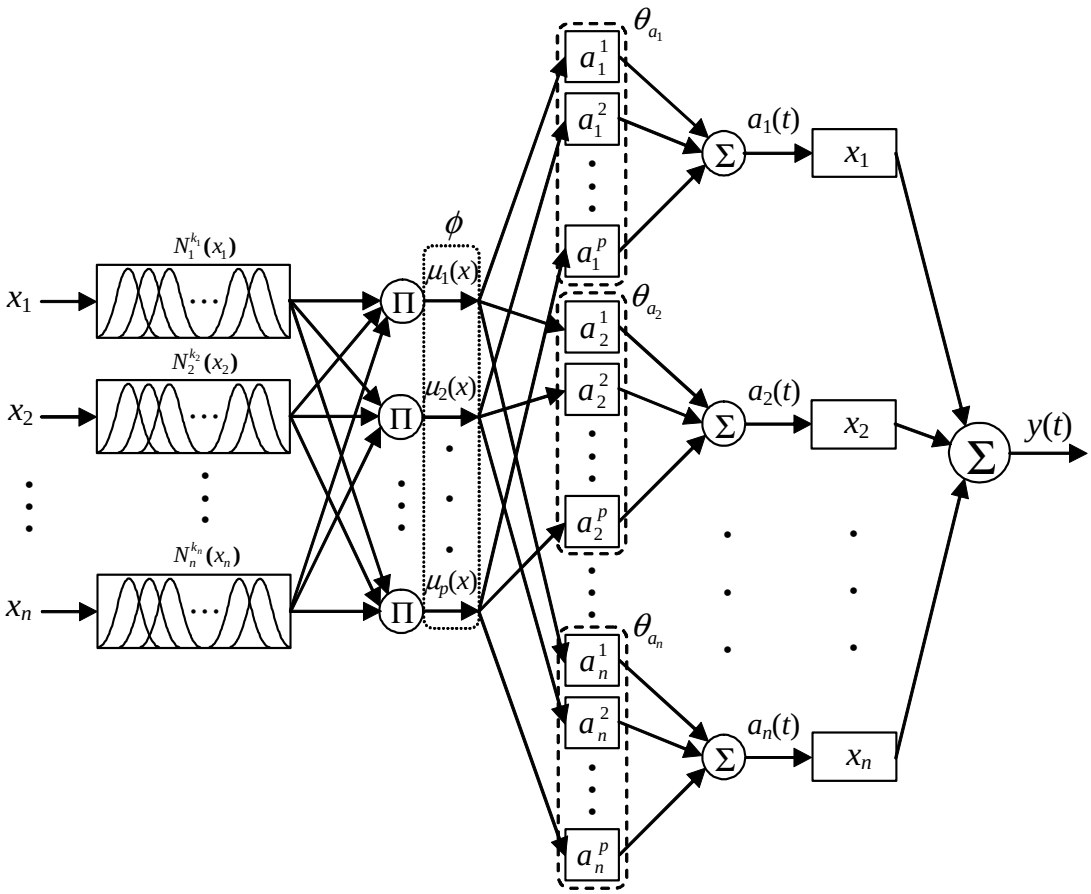


Figure 7.8 Simplified neuro-fuzzy modelling network.

Therefore, by using the vectors defined in (7.35) the output of the simplified neuro-fuzzy network can be calculated as:

$$y(t) = \sum_{j=1}^{p} \mu_j(x) a^j x(t) = \sum_{j=1}^{p} \mu_j(x) y_j(t) \tag{7.36}$$

where:

$$a^j x(t) = a_1^j x_1 + a_2^j x_2 + \cdots + a_n^j x_n = y_j(t), \qquad j = 1,2,\ldots,p \tag{7.37}.$$

Note that (7.37) is equivalent to (7.6). Therefore, the solution given by (7.36) is equivalent to the solution given by (7.9) or (7.34).

The free parameters of the simplified neuro-fuzzy modelling network (or the original neuro-fuzzy network) can be organised in a matrix form as is shown in Table 7.1. These parameters can be seen as column vectors θ_{a_i} or as row vectors a^j. Depending on which way the free parameters are taken, as column or row vectors, equation (7.34) or (7.36), respectively, is used to obtain the output of the neuro-fuzzy network.

The simplification of the neuro-fuzzy network does not alter its characteristic of being easy to translate to a state-space representation as is required by the SKF. In fact, exactly the same procedure used in the original network, and explained in section 7.2, can be followed.

Therefore, it is straightforward to include the simplified neuro-fuzzy network in the indirect and direct neuro-fuzzy-SKF schemes, shown in figures 7.4 and 7.5, to perform state estimations. In the indirect neuro-fuzzy-SKF scheme, instead of using the original neuro-fuzzy modelling network, the simplified neuro-fuzzy modelling network is employed. The connection between the simplified neuro-fuzzy network and a SKF in the direct neuro-fuzzy-SKF scheme is shown in figure 7.9.

Table 7.1
Matrix formed with the free parameters of the neuro-fuzzy network

	θ_{a_1}	θ_{a_2}	$\cdots$	θ_{a_n}
a^1	a_1^1	a_2^1	$\cdots$	a_n^1
a^2	a_1^2	a_2^2	$\cdots$	a_n^2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
a^p	a_1^p	a_2^p	$\cdots$	a_n^p

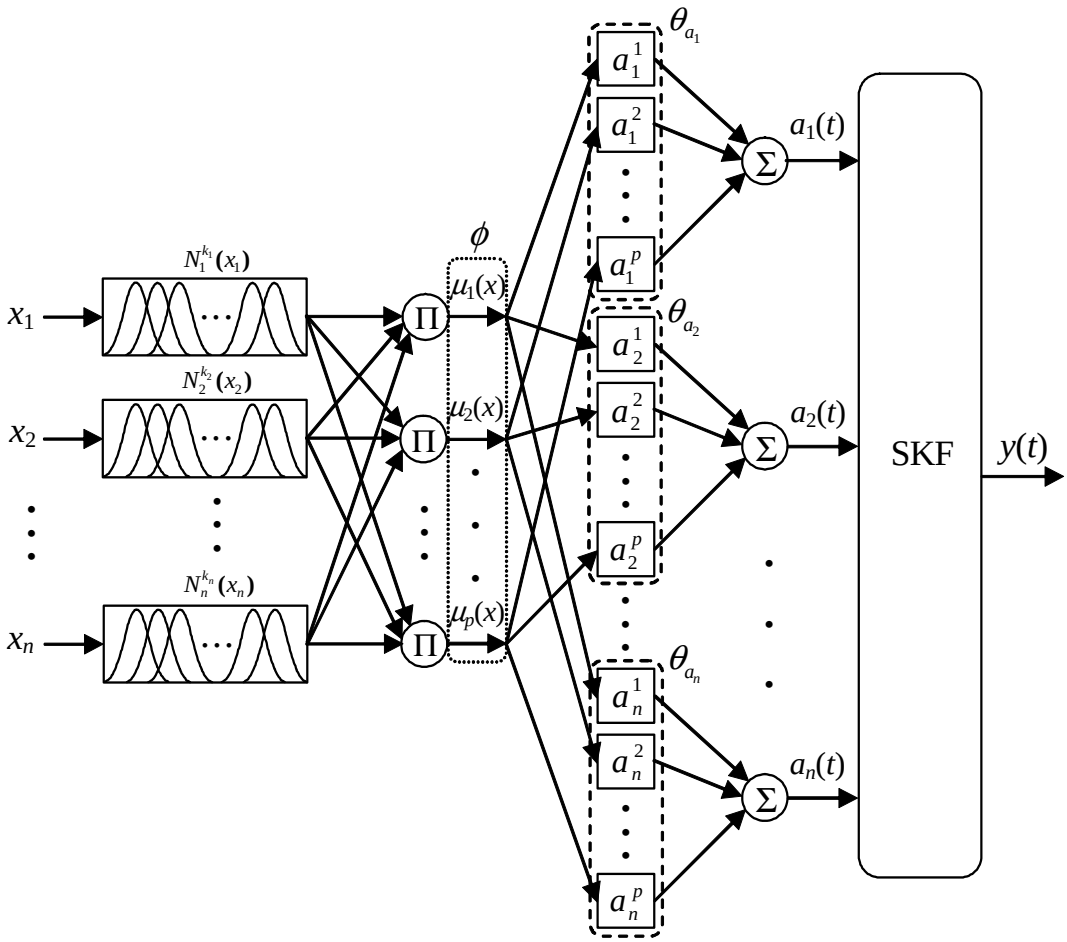


Figure 7.9 Connection between the simplified neuro-fuzzy network and a SKF.

With respect to the training of the simplified neuro-fuzzy-SKF, the procedures presented in section 7.2.1 are directly applied in both indirect and direct neuro-fuzzy-SKF state estimation schemes. Therefore, they are not repeated here.

7.4 The neuro-fuzzy-AKF state estimator

In this section the FL-AKF presented in Chapter 5 is used to develop a neuro-fuzzy-AKF state estimator (from here referred to as neuro-fuzzy-AKF). This is achieved by incorporating the FL-AKF in the direct neuro-fuzzy-SKF scheme, as is shown in figure 7.10. Note that instead of employing a SKF, a FL-AKF is used and the error signal is fed to the FL-AKF.

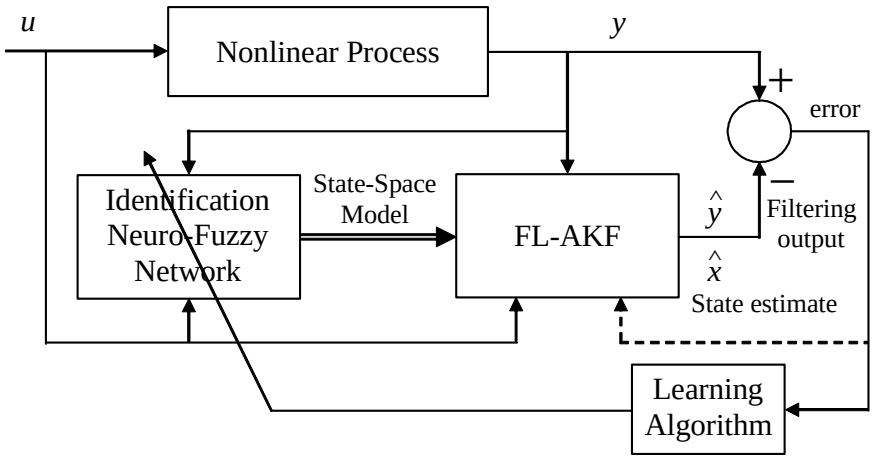


Figure 7.10 Direct neuro-fuzzy-AKF state estimation scheme.

In practice, system identification using the neuro-fuzzy-AKF can be implemented using two different approaches. The first is a series-parallel model, while the second is merely a parallel model, resembling the corresponding neural network identification approaches [Narendra and Parthasarathy, 1990] [Nelles and Isermann, 1996].

In the series-parallel model the previous process input and output are fed into the neuro-fuzzy-AKF and the error signal is used as a measurement signal for the FL-AKF as is shown in figure 7.11(a). Hence, the past values of the input and output of the plant form the input vector to the neuro-fuzzy-AKF whose output $\hat{y}(t)$ corresponds to the estimate of the plant output at any instant of time *t*. This model is similar to the direct neuro-fuzzy-SKF approach and, therefore, the same learning procedure used for that case can be applied here. The way in which the error signal is managed in the neuro-fuzzy-AKF will be clarified later.

In the parallel model the previous neuro-fuzzy-AKF output is fed back to the identification model and the error signal is used as measurement signal for the FL-AKF, as is shown in figure 7.11(b). The parallel model is recurrent and therefore can predict an arbitrary number of steps into the future. However, due to the feedback, the model inputs $\hat{y}(t-i)$ depend on the model parameters. Therefore the identification model becomes nonlinear in the parameters. This makes the gradient calculations a nonlinear optimisation problem, which requires a different learning technique. In the neural network literature, it is argued that using a parallel identification model is a difficult problem [Nelles and Isermann, 1996]. Furthermore, the parameter optimisation may become unstable. Due to this, here the series-parallel neuro-fuzzy-AKF model is used during the process of system identification. Once the system under consideration has been identified, and assuming that the output error tends to a small value asymptotically so that

$y(k) \approx \hat{y}(t)$, the series-parallel model can be replaced by a parallel neuro-fuzzy-AKF model without serious consequences [Narendra and Parthasarathy, 1990].

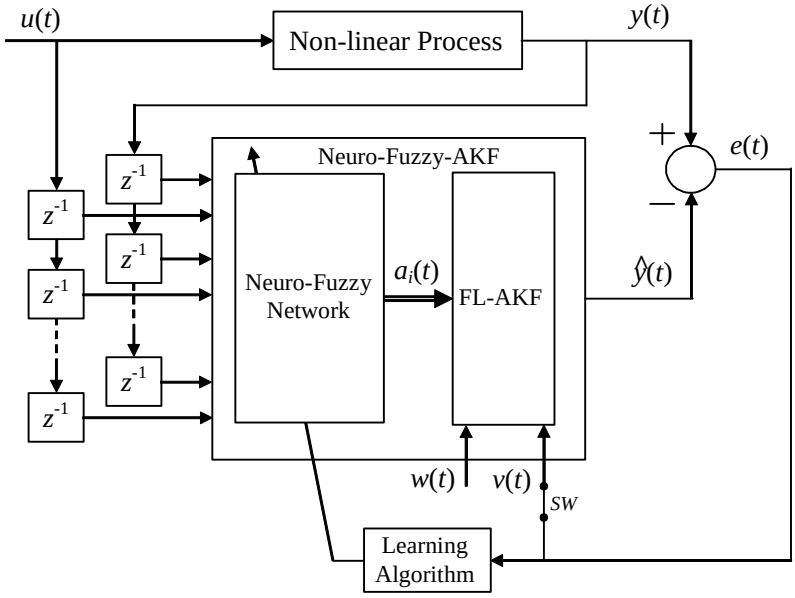


Figure 7.11(a) Series-parallel Neuro-Fuzzy-AKF model.

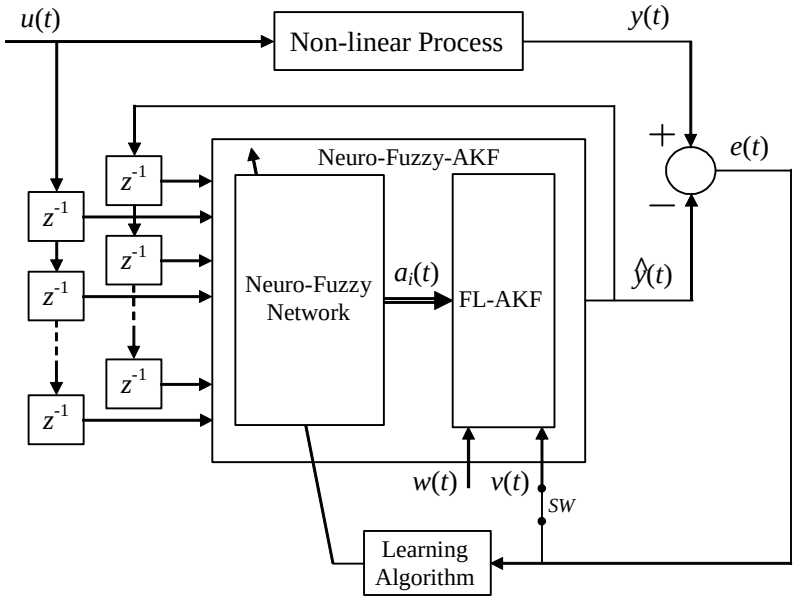


Figure 7.11(b) Parallel Neuro-Fuzzy-AKF model.

During the training process the error signal in the neuro-fuzzy-AKF scheme is used as a measurement noise signal in the state-space representation of the system (equations 7.17 and 7.18) used in the FL-AKF (see figure 7.11). By using an artificial process noise sequence [Haykin, 1999], $w(t)$, with known and fixed covariance, $Q(t)=Q \ \forall \ t$, of a low value, the covariance of the measurement noise, which in this case is the error signal, can be approximated by the adaptation algorithm in the FL-AKF. This is achieved by employing the algorithm of adaptive estimation of the measurement noise covariance matrix $R(t)$ assuming that Q is known, described in Chapter 5, section 5.3.2.a. This means that at the same time in which the identification of the process is carried out an approximation of the mismatch between the

identified model and the actual model is carried out and this mismatch is represented by the estimated measurement noise covariance matrix $R(t)$.

Therefore, at the beginning of the training of the neuro-fuzzy-AKF using the series-parallel model, a low value for Q is given, which defines an artificial process noise sequence, and an initial guessed value for $R(t)$ is defined. While Q is maintained constant over the whole training process, $R(t)$ is adaptively adjusted to match the covariance value of the error sequence. At the end of each epoch an average of $R(t)$ is obtained as:

$$R = \frac{1}{N}\sum_{t=1}^{N} R(t) \tag{7.38}$$

where N is the number of training input-output samples using for training purposes. This averaged value, R, is used as initial measurement noise covariance for the next epoch. Consequently, as the identification process progresses, and if the nonlinear system is being correctly identified, the value of R decreases with each epoch until it reaches a quasi-steady state around a small value, indicating that the identification process has converged. This utilisation of the error signal as measurement signal for the state-space representation of the system used in the FL-AKF has the effect of stabilizing the training process. In addition, it helps to avoid the possible divergence of the filter because the FL-AKF has the knowledge of the approximated mismatch between the identified model and the actual system, represented as the measurement noise covariance value, R.

Once the identification process is terminated, the switch *SW* in figure 7.11 is opened and an artificial measurement noise signal, $v(t)$, with covariance R is used to substitute the error signal.

7.5 MSDF using the neuro-fuzzy-AKF state estimator

In the last chapter four adaptive MSDF architectures based on the FL-AKF, developed in Chapter 5, were presented. In order to apply these architectures a state-space representation of the system under consideration must be available. In section 7.4 a novel neuro-fuzzy-AKF for non-linear system identification and state estimation has been developed. As an interesting characteristic, the neuro-fuzzy-AKF is capable of identifying and expressing in the form of a time varying state-space representation the non-linear system under study. As a consequence, the FL-AKF can be directly applied. Therefore, in this section the hybrid architecture FL-AKF-FLA (see section 6.3.1) is applied for MSDF of the information coming from N neuro-fuzzy-AKFs.

The implementation of the FL-AKF-FLA scheme using neuro-fuzzy-AKFs is shown in figure 7.12. This scheme is similar to that presented in section 6.3.1, but here the FL-AKFs are substituted by neuro-fuzzy-AKFs. Note also, that in this case the information that is being fused are the estimated nonlinear plant outputs $\hat{y}_i(t)$, carried out by the different neuro-fuzzy-AKFs, instead of the state vectors. The fusion process is carried out through a weighted average scheme based on the confidence values calculated by the Fuzzy Logic Assessors (FLAs). The FLAs are assessing the performance of each neuro-fuzzy-AKF, and calculate a degree of confidence value, c_i, using a fuzzy inference system (FIS). Each FIS has as inputs the absolute value of the Degree of Mismatch ($|DoM|$) and the estimated value of $R(t)$, calculated in each neuro-fuzzy-AKF (specifically, in each FL-AKF). For a complete description of the algorithm, the reader is referred to section 6.3.1. Therefore, the application of the FL-AKF-FLA for MSDF using neuro-fuzzy-AKFs is straightforward.

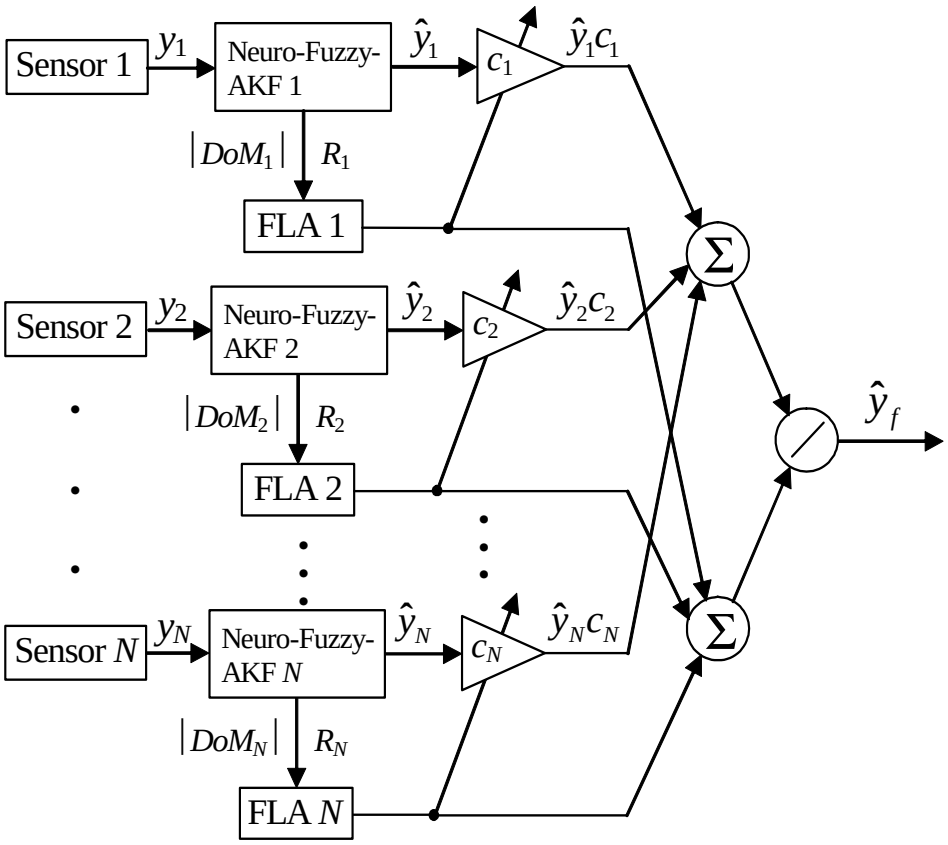


Figure 7.12 Implementation of the FL-AKF-FLA scheme using neuro-fuzzy-AKFs

7.6 Simulation results

In this section simulation results of nonlinear plant identification, state estimation, and MSDF using the neuro-fuzzy-AKF and the MSDF architecture described earlier are presented. Two benchmark processes taken from the literature are simulated. The MATLAB/SIMULINK platform for simulation was used and the developed SIMULINK models are presented in the Appendix B.

In both examples, during the identification process, a series-parallel model is used, but when the identification process is terminated the performance of the neuro-fuzzy-AKFs is evaluated using a parallel model, as is commonly reported in the neural networks literature [Narendra and Parthasarathy, 1990].

One of the problems of LMS based learning algorithms is that they have a slow rate of convergence. In order to accelerate convergence, the learning rate η in (7.30) is changed as the number of epochs increases by using the relation [Haykin, 1999]:

$$\eta(n) = \frac{\eta_0}{1 + (n + \varsigma)} \tag{7.40}$$

where $\eta(n)$ is the learning rate at the current epoch n, η_0 and ς are constants selected in order to define the decreasing rate of the learning-rate parameter. Obviously, η_0 must be inside the permissible range (0, 2) for the learning-rate parameter value. In the simulations carried out in this study the values selected are $\eta_0 = 2$, and $\varsigma = 50$.

The identification process was terminated when satisfactory correlation validity tests [Billings and Zhu, 1994]: $\Phi_{ee}(\tau)$, $\Phi_{(ye)e^2}(\tau)$, and $\Phi_{(ye)u^2}(\tau)$ were obtained.

Example 1. Consider a nonlinear system described by [Chen and Khalil, 1995] [Hong and Harris, 2001] [Liu *et al*, 1999]:

$$z(t)=\frac{2.5z(t-1)z(t-2)}{1+z^2(t-1)+z^2(t-2)}-0.3\cos[0.5(z(t-1)+z(t-2))]+1.2u(t-1) \qquad (7.41)$$

where z(t) is a state variable that has to be estimated from the measurements from two different noisy sensors:

$$y_1(t)=z(t)+\xi_1(t) \qquad (7.42)$$

$$y_2(t)=z(t)+\xi_2(t) \qquad (7.43)$$

where $\xi_1(t)$ and $\xi_2(t)$ are independent Gaussian zero–mean white noise sequences with variances 0.0025 and 0.0125 respectively. It means that sensor 1 is 5 times more accurate than sensor 2.

To train both neuro-fuzzy-AKF state estimators, a sequence of 1000 observations $y_1(t)$ and 1000 observations $y_2(t)$ were generated using as input signal $u(t)$ a chirp signal (sine wave whose frequency varies linearly with time) with initial frequency of 0.004 Hz, frequency at target time of 0.04 Hz, target time of 1000 sec, and sampling time of 1 sec. The generated input signal $u(t)$, state signal $z(t)$, and observation signals $y_1(t)$ and $y_2(t)$ are shown in figure 7.13.

Two neuro-fuzzy-AKFs were used to approximate the system. In both cases B-spline basis functions of order 2 were used as fuzzy sets for the input variables. The input vector to the neuro-fuzzy modelling network was predetermined as $x(t)=[y(t\text{-}1), y(t\text{-}2), u(t\text{-}1), u(t\text{-}2)]^T$. The knot vectors for $u(t\text{-}i)$ were defined as [-2, -1, 0, 1, 2], while the knot vectors for $y(t\text{-}i)$ were specified as [-2.75, -1, 0.75, 2.5, 4.25]. These knot vectors were defined based on the maximum range of possible values for the input and output: [-1, 1] and [-1, 2.5], respectively.

The initial conditions for the FL-AKFs inside the neuro-fuzzy-AKF structures were defined as: $\hat{z}_1(0)_{(-)}=\hat{z}_2(0)_{(-)}=[0 \quad 0.3]^T$, $P_1(0)_{(-)}=P_2(0)_{(-)}=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$, $Q_1 = Q_2 = 2\text{x}10^{-5}$, $R_1(t) = R_2(t) = 0.02$; while the initial state vectors were defined as: $z_1(0)_{(-)}=z_2(0)_{(-)}=[0 \quad 0.3]^T$ (the sub-indices refer to the number of neuro-fuzzy-AKF). As mentioned earlier, Q_1 and Q_2 were maintained as constants during the training process. However, the average of the estimated $R_1(t)$ and $R_2(t)$, obtained applying (7.38), were used as their initial values for the next epoch, whereas $P_1(1000)$ and $P_2(1000)$ were used as initial conditions for the next epoch.

The parameters that define the fuzzy sets in the FISs used in each FL-AKF to adjust $R_i(t)$, are $a = 5$ and $b = 0.3$. While the parameters which define the fuzzy sets in the FISs used in each FLA algorithm in the FL-AKF-FLA MSDF architecture are $g = 1.5$ and $h = 3$. For practical reasons, seen in chapter 5, the size of the sliding window in all FL-AKFs is selected as 15.

In order to evaluate the performance of individual neuro-fuzzy-AKFs and the fusion algorithm, the following mean squared error measures were adopted:

$$MSEy1=\frac{1}{N}\sum_{t=1}^{N}(e_1(t))^2=\frac{1}{N}\sum_{t=1}^{N}(y_1(t)-\hat{y}_1(t))^2 \qquad (7.44a)$$

$$MSEy2 = \frac{1}{N}\sum_{t=1}^{N}(e_2(t))^2 = \frac{1}{N}\sum_{t=1}^{N}(y_2(t) - \hat{y}_2(t))^2 \tag{7.44b}$$

$$MSEz1 = \frac{1}{N}\sum_{t=1}^{N}(e_{z1}(t))^2 = \frac{1}{N}\sum_{t=1}^{N}(z(t) - \hat{y}_1(t))^2 \tag{7.44c}$$

$$MSEz2 = \frac{1}{N}\sum_{t=1}^{N}(e_{z2}(t))^2 = \frac{1}{N}\sum_{t=1}^{N}(z(t) - \hat{y}_2(t))^2 \tag{7.44d}$$

$$MSEf = \frac{1}{N}\sum_{t=1}^{N}(e_f(t))^2 = \frac{1}{N}\sum_{t=1}^{N}(z(t) - \hat{y}_f(t))^2 \tag{7.44e}$$

where $e_1(t)$ and $e_2(t)$ are the error values between the estimated and measured signals, $e_{z1}(t)$ and $e_{z2}(t)$ are the errors between the estimated and actual state (noise free) signals, and $e_f(t)$ is the error between the fused estimated signal and the actual state signal.

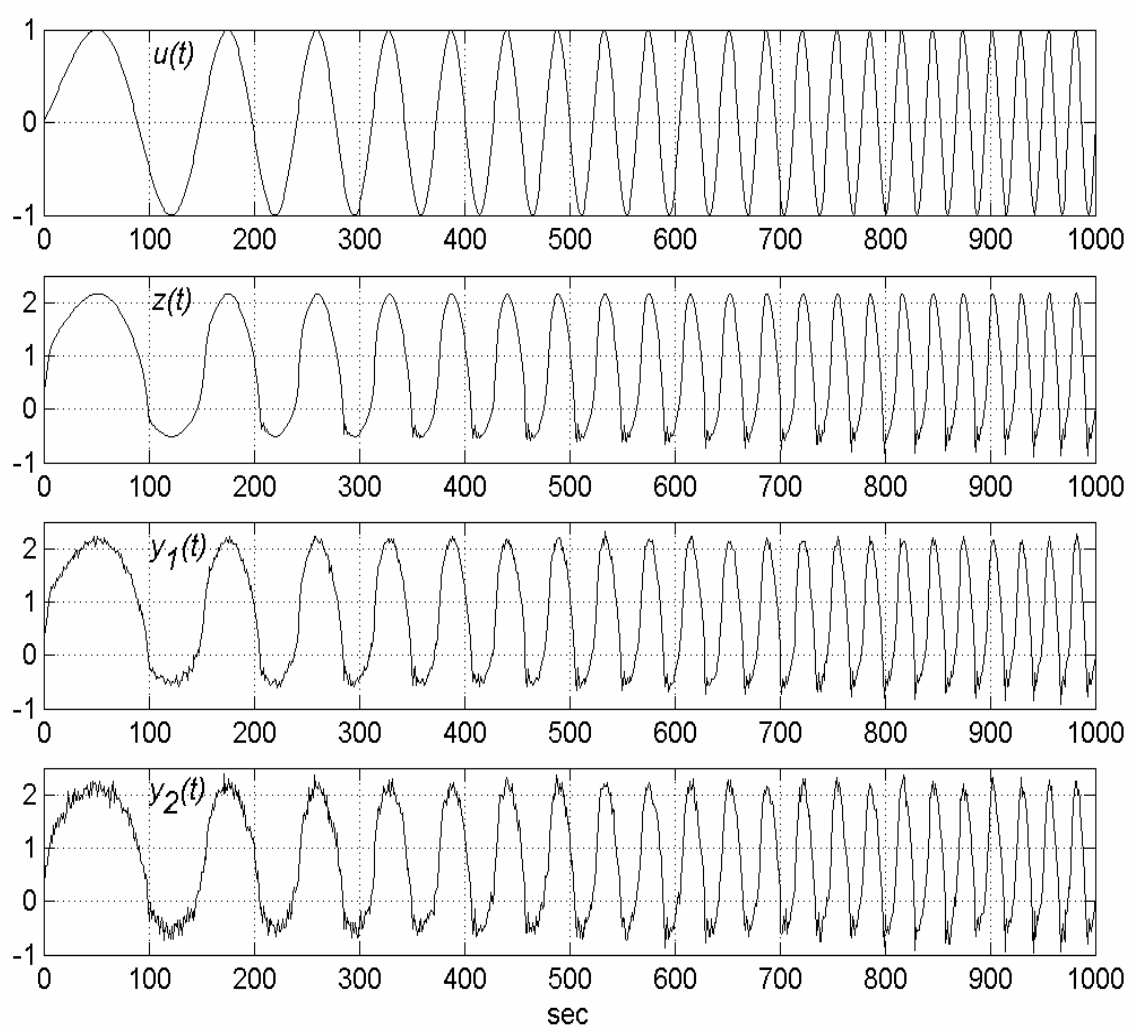


Figure 7.13 Input signal $u(t)$, state signal $z(t)$, observation $y_1(t)$, and observation $y_2(t)$.

The training process was carried out cyclically using the training data. At the end of each epoch (cycle), the validity correlation tests were carried out. The training process was maintained until acceptable correlation validity tests were obtained and the mean squared error values for both networks, given by (7.42a) and (7.42b), have converged. Figure 7.14 shows the model validity correlation tests for the neuro-fuzzy-AKF state estimators at epoch 500, which

showed satisfactory results, and thereafter the training process was stopped. The convergence of the *MSEyi* values is shown in figure 7.15. The *MSEyi* values obtained at the end of the training process are shown in Table 7.2. There also is given the *MSEf* value (given by (7.42e)) obtained by fusing the neuro-fuzzy-AKF estimates.

Figure 7.16 shows the performance of both neuro-fuzzy-AKFs at the end of the training process. From the inspection of that figure, it is obvious that the noise present in sensor 2 is greater than the noise present in sensor 1, and thus the error in the neuro-fuzzy-AKF 2 is greater. However, note that both neuro-fuzzy-AKFs perform a good approximation to the measured signal. The comparison of the estimated signals with the actual signal (the noise free signal $z(t)$) and the fused obtained signal at the end of the training process is shown in figure 7.17. Note that only a slight error exists between the fused estimate and the actual signals.

Figure 7.18 shows the approximated measurement noise covariance values obtained at epoch 500 for both neuro-fuzzy-AKFs. It can be appreciated that a quasi-steady state has been reached. The averaged measurement noise covariance values at this epoch were R_1=0.0031 and R_2=0.0129. Note that these values are very near to the actual measurement noise covariance values 0.0025 and 0.0125. These values as well support the assumption that the training process has converged.

Table 7.2 Training and validation MSE measures

Training			Validation		
MSEy1	*MSEy2*	*MSEf*	*MSEy1*	*MSEy2*	*MSEf*
$3.18\text{x}10^{-3}$	$12.74\text{x}10^{-3}$	$7.566\text{x}10^{-4}$	$4.05\text{x}10^{-3}$	$15.13\text{x}10^{-3}$	$1.929\text{x}10^{-3}$
			MSEz1	*MSEz2*	*MSEf*
			$1.381\text{x}10^{-3}$	$3.49\text{x}10^{-3}$	$1.929\text{x}10^{-3}$
Final averaged measurement noise covariance values: R_1=0.0031, R_2=0.0129					

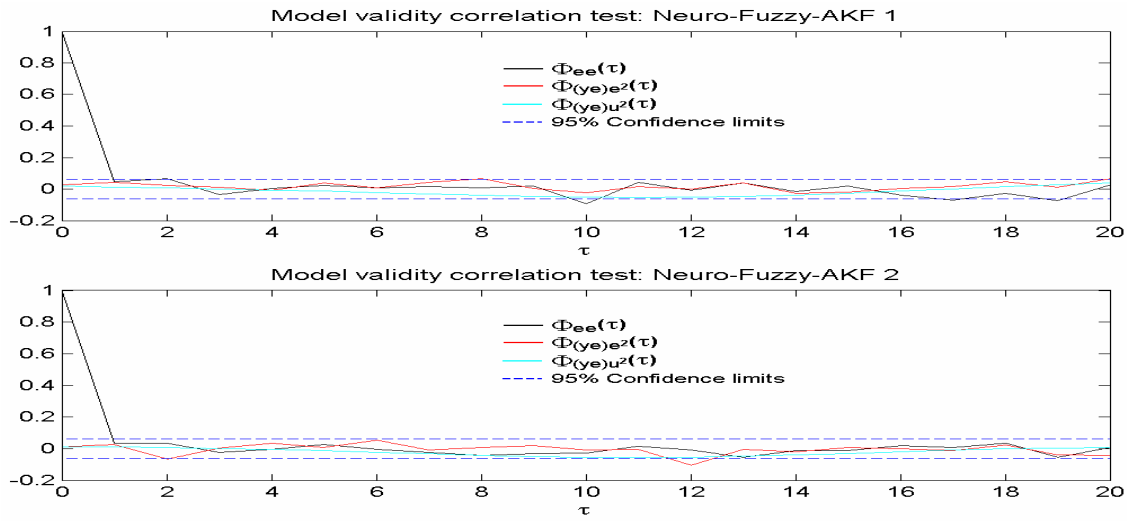


Figure 7.14 Model validity correlation test for the Neuro-Fuzzy-AKF state estimators at the end of the training process.

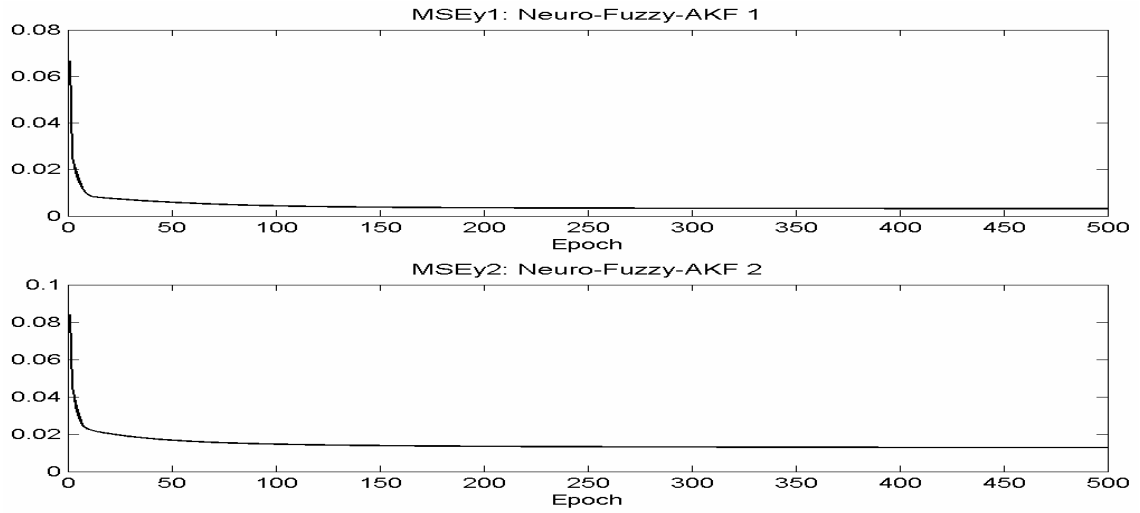


Figure 7.15 MSEy1 and MSEy2 in the training process.

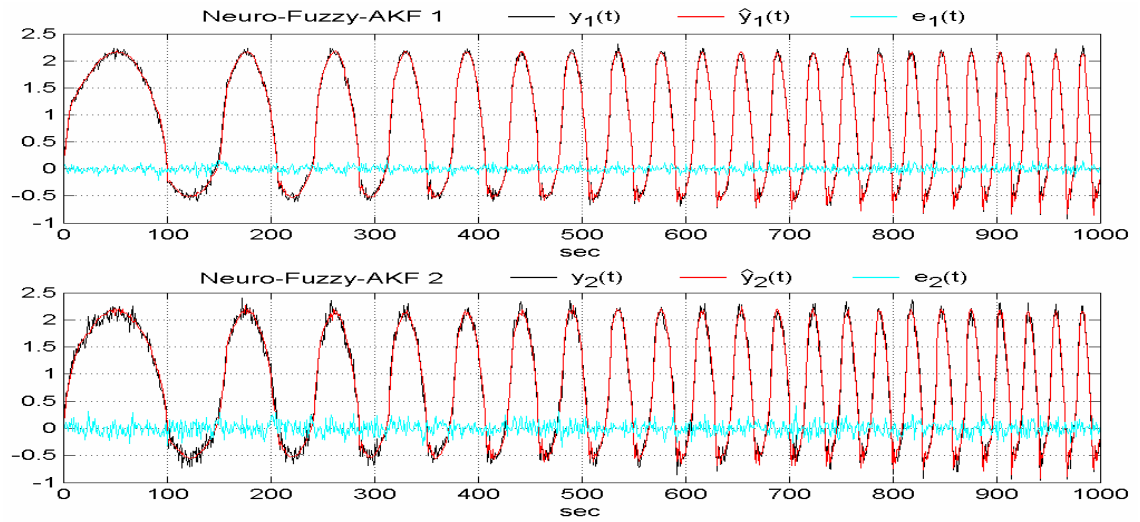


Figure 7.16 Performance of the Neuro-Fuzzy-AKF state estimators at the end of the training process.

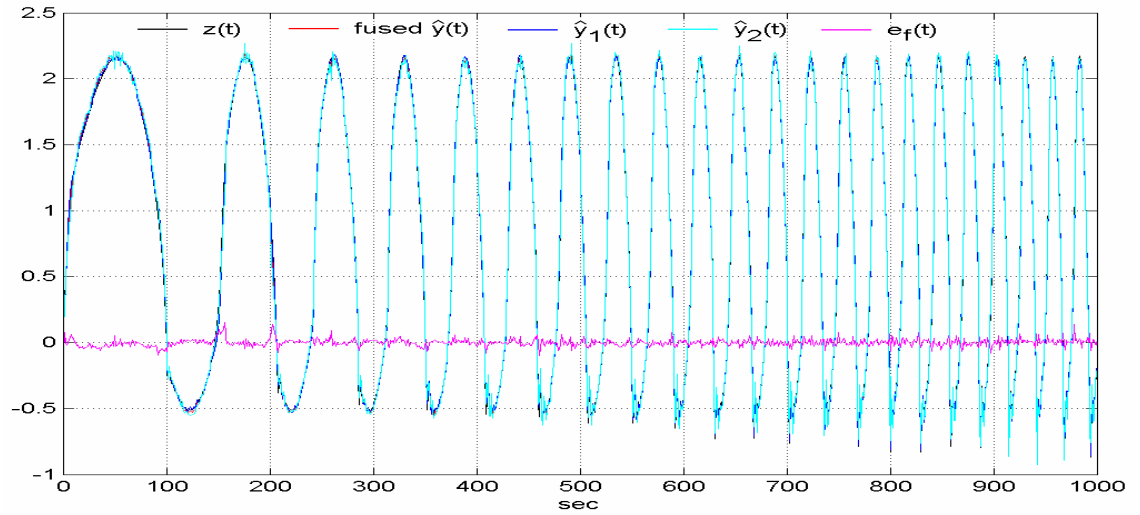


Figure 7.17 Performance of the FL-AKF-FLA fusion algorithm at the end of the training process.

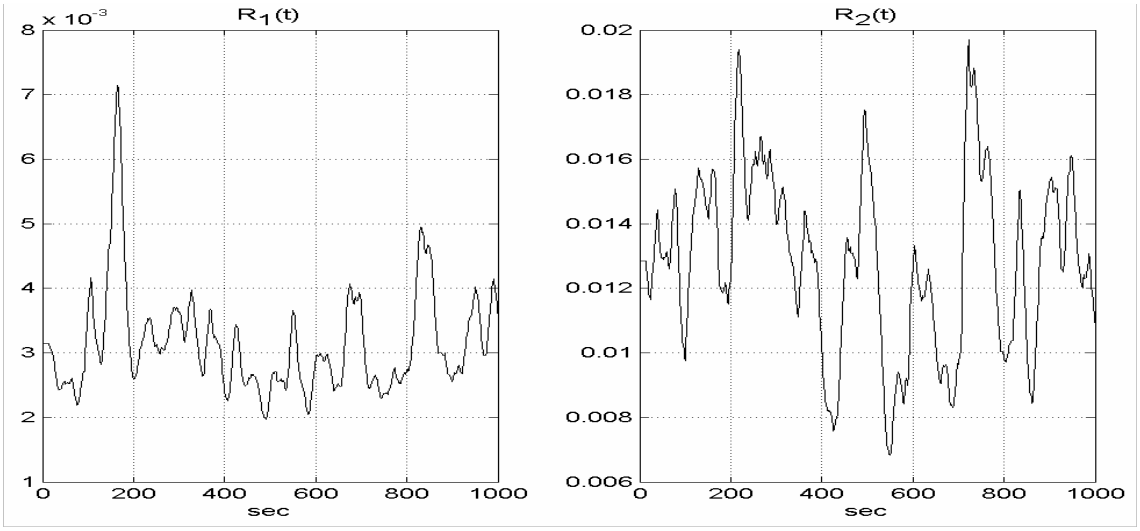


Figure 7.18 Measurement noise covariance values obtained at epoch 500.

A validation data set of 1000 samples of $z(t)$, $y_1(t)$ and $y_2(t)$ were generated using the signal $u(t)=sin(\pi t/80)$ as input with sample time of 1 sec. The generated validation signals are shown in figure 7.19.

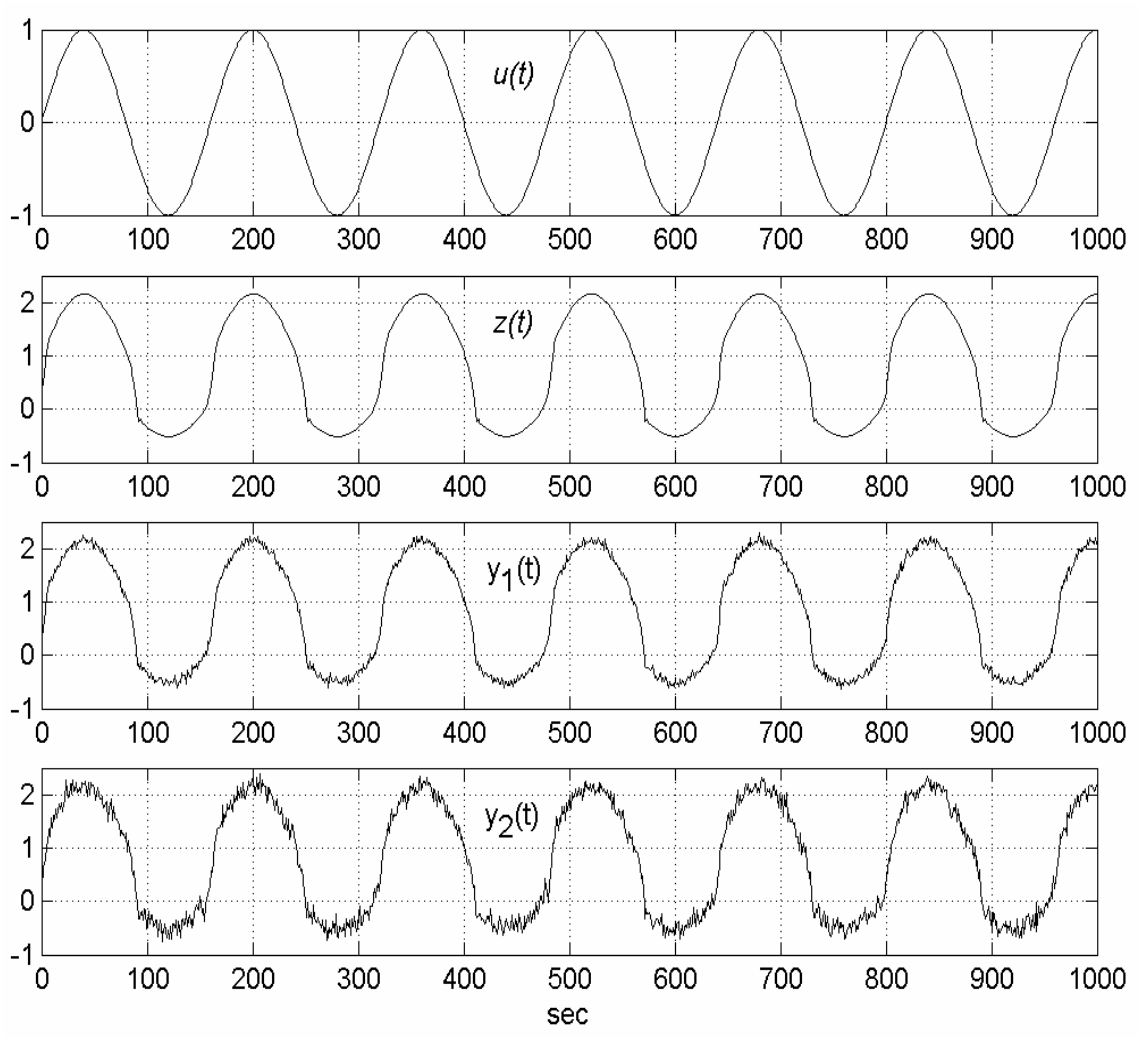


Figure 7.19 Validation signals generated with $u(t)=sin(\pi t/80)$.

Figure 7.20 shows the performance of both neuro-fuzzy-AKFs using the validation data where the estimated signals are compared with the measured signals. Note that good approximation is obtained. The MSE*yi* values obtained are shown in Table 7.2 and figure 7.21 shows the estimated signals compared with the actual noise free signal. It is relevant to note that in both filters a very good estimation of the actual signal is obtained, as is demonstrated by the *MSEzi* values shown in Table 7.2.

In order to appreciate the performance of the fusion algorithm, the fused signal, the actual signal, and both estimated signals are plotted in figure 7.22. The obtained *MSEf* value is shown in Table 7.2. Note that, for this particular case, the fused data is slightly less accurate than the data obtained with sensor 1.

Finally, figure 7.23 shows the approximated measurement noise covariance value, obtained over the validation data. Note that effectively the averaged value of R_i does not change, and the quasi-steady state is maintained.

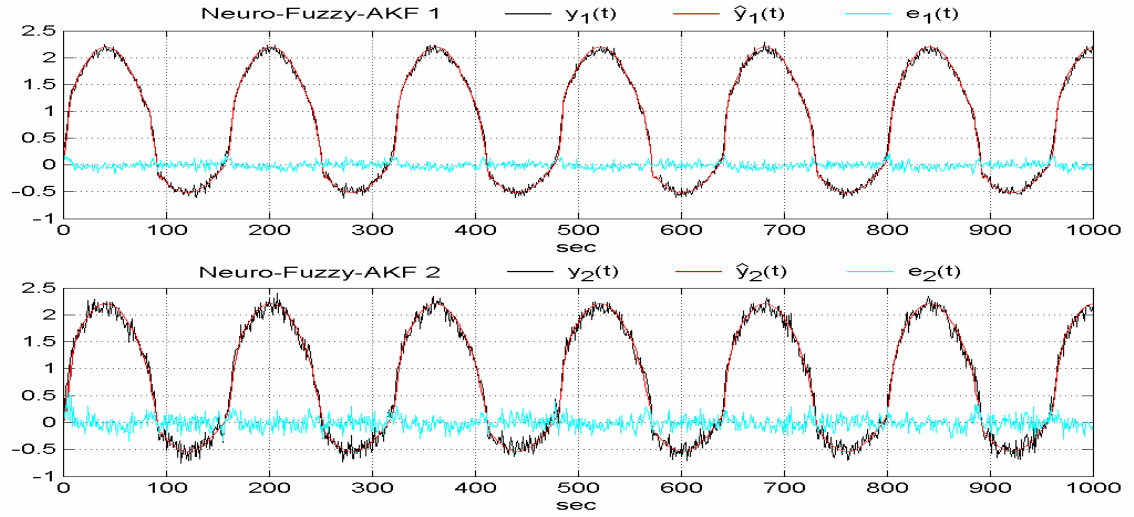


Figure 7.20 Performance of the neuro-nuzzy-AKFs over the validation data.

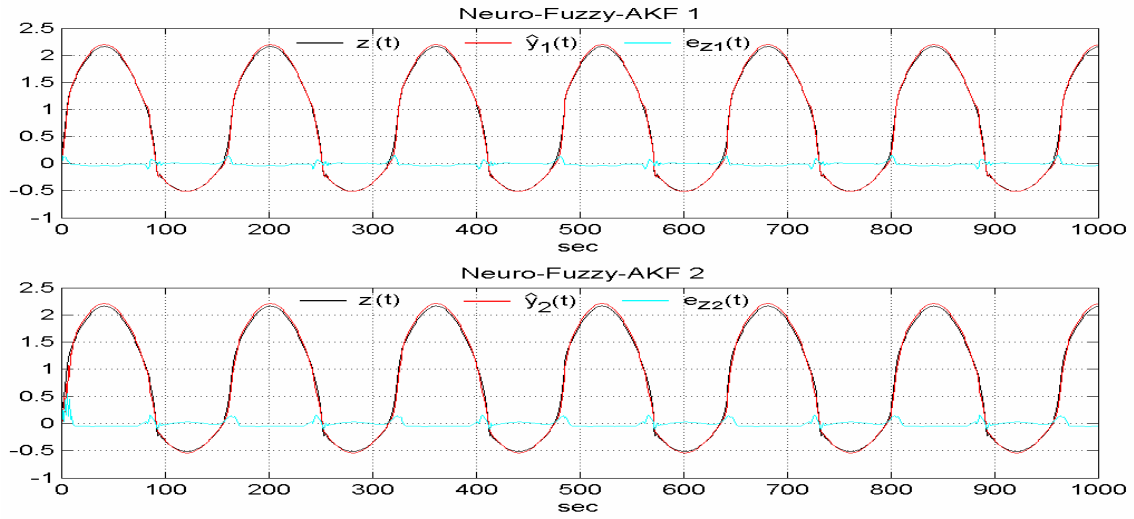


Figure 7.21 Actual and estimated signals over the validation data.

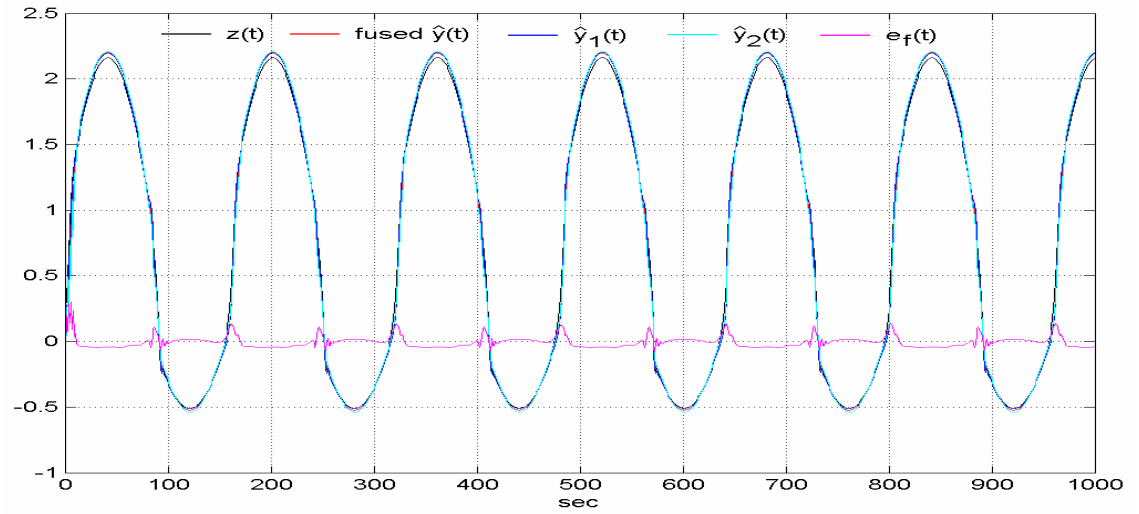


Figure 7.22 Fused and estimated signals compared with the actual noise free signal over the validation data.

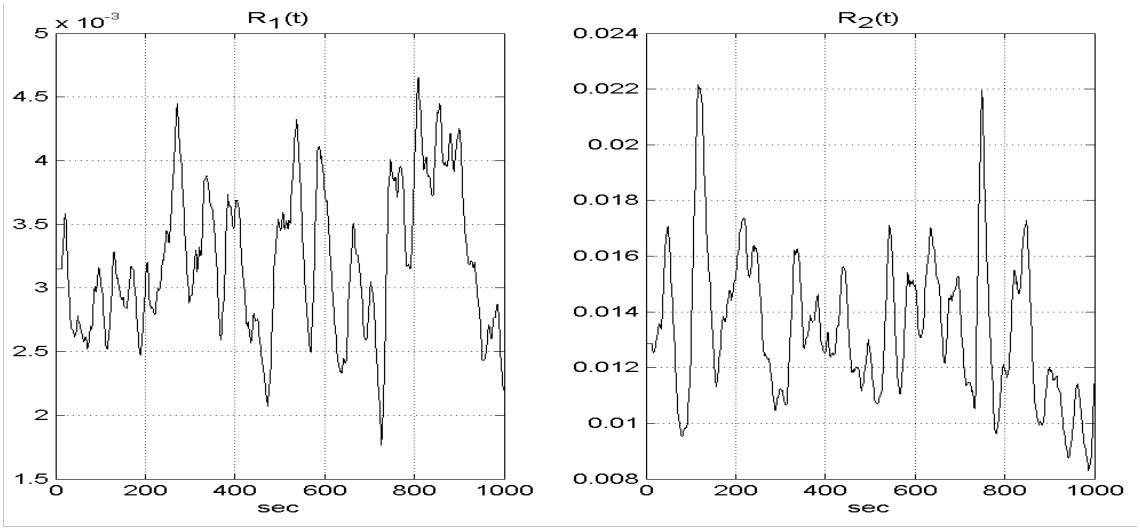


Figure 7.23 Measurement noise covariance values obtained during the validation process.

Example 2. Consider a nonlinear system described by [Harris *et al*, 2000]:

$$z(t) = \frac{z(t-1)z(t-2)}{1+z(t-1)^2} - 0.2\cos[\pi u(t-1)]z(t-2) + 1.2u(t-1) \tag{7.45}$$

where z(t) is a state variable to be estimated from two different noisy sensor measurements:

$$y_1(t) = z(t) + \xi_1(t) \tag{7.46}$$

$$y_2(t) = z(t) + \xi_2(t) \tag{7.47}$$

where $\xi_1(t)$ and $\xi_2(t)$ are independent Gaussian zero–average white noise sequences with variances 0.01 and 0.002 respectively. Note that in this case sensor 2 is 5 times more accurate than sensor 1.

Two neuro-fuzzy-AKF state estimators were considered for the task of system identification and state estimation. To train both neuro-fuzzy-AKFs, a sequence of 500 observations for $y_1(t)$ and 500 observations for $y_2(t)$ were generated using as input signal $u(t)$ a chirp signal with initial frequency of 0.004 Hz, frequency at target time of 0.04 Hz, target time of 500 sec, and sampling time of 1 sec. The generated training signals are shown in figure 7.24.

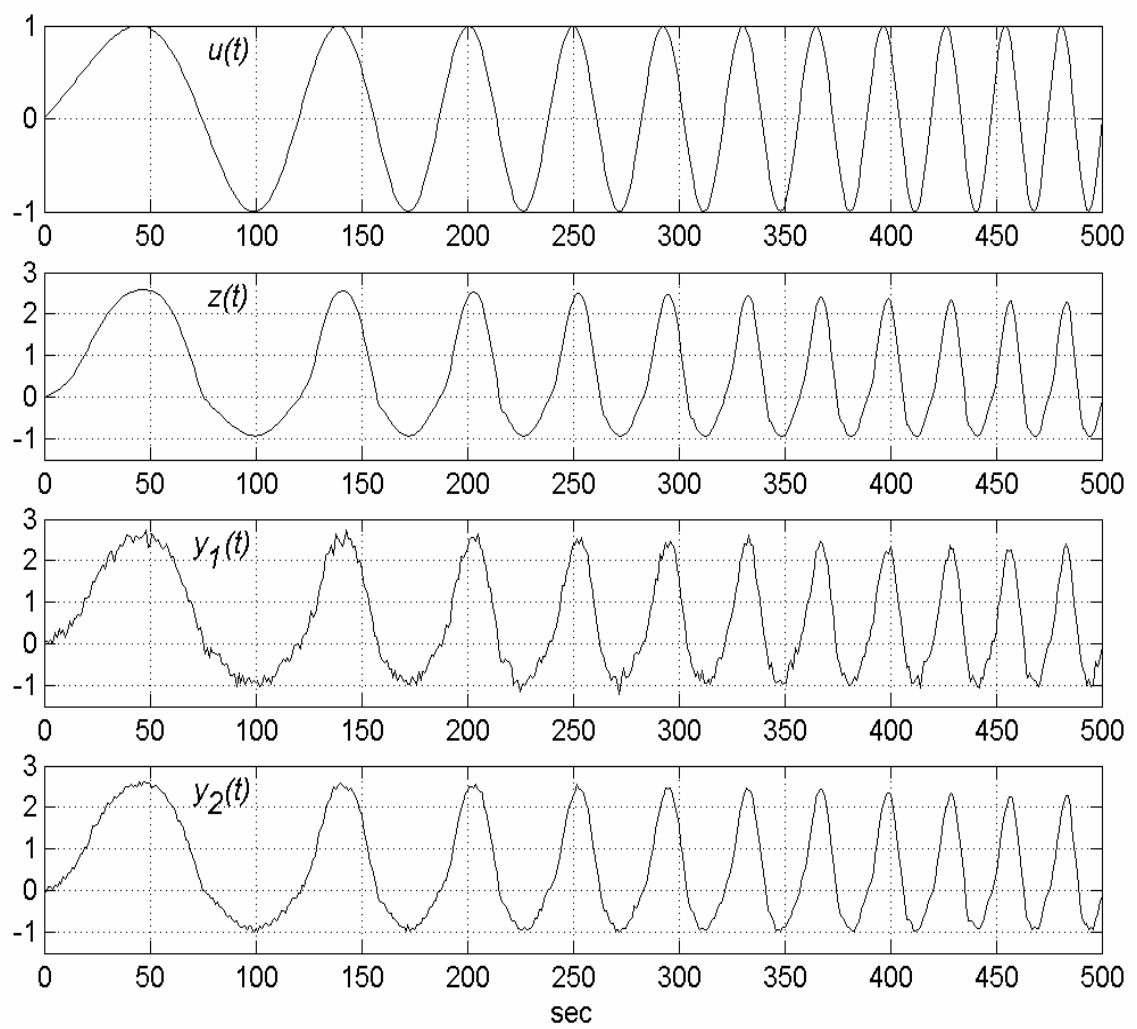


Figure 7.24 Input signal $u(t)$, output state variable $z(t)$, observation $y_1(t)$, and observation $y_2(t)$.

The input vector to the neuro-fuzzy modelling networks was predetermined as $x(t)=[y(t\text{-}1), y(t\text{-}2), u(t\text{-}1), u(t\text{-}2)]^T$. B-spline basis functions of order 2 were used as fuzzy sets for the input variables. The knot vectors for $u(t\text{-}i)$ were defined as [-2, -1, 0, 1, 2], while the knot vectors for $y(t\text{-}i)$ were specified as [-3.6 -1.4 0.8 3.0 5.2]. These knot vectors were defined based on the maximum range of possible values for the input and output: [-1, 1] and [-1.4, 3], respectively.

The initial conditions for the FL-AKFs inside the neuro-fuzzy-AKF structures were defined as: $\hat{z}_1(0)_{(-)} = \hat{z}_2(0)_{(-)} = [0 \quad 0]^T$, $P_1(0)_{(-)} = P_2(0)_{(-)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $Q_1 = Q_2 = 2\text{x}10^{-5}$, $R_1(t) = R_2(t) = 0.02$; while the initial state vectors were defined as: $z_1(0)_{(-)} = z_2(0)_{(-)} = [0 \quad 0]^T$ (the sub-indices refer to the number of neuro-fuzzy-AKFs). Q_1 and Q_2 were maintained as constants during the training process. However, the average of the estimated $R_1(t)$ and $R_2(t)$, obtained applying (7.38), were used as their initial values for the next epoch, whereas $P_1(500)$ and $P_2(500)$ were used as initial conditions for the next epoch.

The parameters that define the fuzzy sets in the FISs used in each FL-AKF to adjust $R_i(t)$, are a = 5 and b = 0.3, while, the parameters which define the fuzzy sets in the FISs used in each FLA algorithm in the FL-AKF-FLA MSDF architecture are g = 1.5 and h = 3. The size of the sliding window in all FL-AKFs is selected as 15.

The training data was used cyclically to train both neuro-fuzzy-AKFs. The validity correlation tests were carried out at the end of each epoch. The training process was maintained until acceptable correlation validity tests were obtained and the mean squared error values had approximately converged. The model validity correlation tests for the neuro-fuzzy-AKFs at epoch 400 are shown in figure 7.25, which showed satisfactory results, and thereafter the training process was stopped. The convergence of the *MSEyi* values is shown in figure 7.26. The *MSEyi* values at the end of the training process are shown in Table 7.3. The *MSEf* value obtained by fusing the neuro-fuzzy-AKF estimates also is given in Table 7.3.

The performance of both neuro-fuzzy-AKFs at the end of the training process is shown in figure 7.27. Note that the noise present in sensor 1 is greater than the noise present in sensor 2, and due to that the error in the neuro-fuzzy-AKF 1 is greater. However, both neuro-fuzzy-AKFs perform a good approximation to the measured signals. A comparison of the estimated signals with the actual signal $z(t)$ and the obtained fused signal at the end of the training process is shown in figure 7.28. Note that there is a very small error between the fused estimated signal and the actual signal.

Table 7.3 Training and validation MSE measures

Training error			Validation error		
MSEy1	*MSEy2*	*MSEf*	*MSEy1*	*MSEy2*	*MSEf*
10.94×10^{-3}	2.333×10^{-3}	4.786×10^{-4}	11.67×10^{-3}	3.281×10^{-3}	6.699×10^{-4}
			MSEz1	*MSEz2*	*MSEf*
			1.521×10^{-3}	1.223×10^{-3}	6.699×10^{-4}
Final averaged measurement noise covariance values: R1=0.011 R2=0.0023					

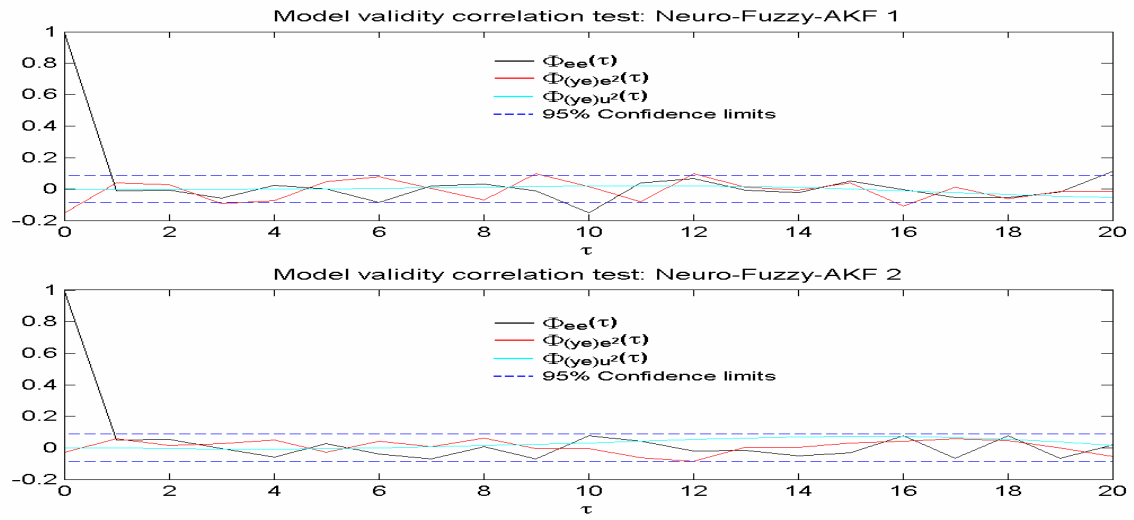


Figure 7.25 Model validity correlation test for the Neuro-Fuzzy-AKF state estimators at the end of the training process.

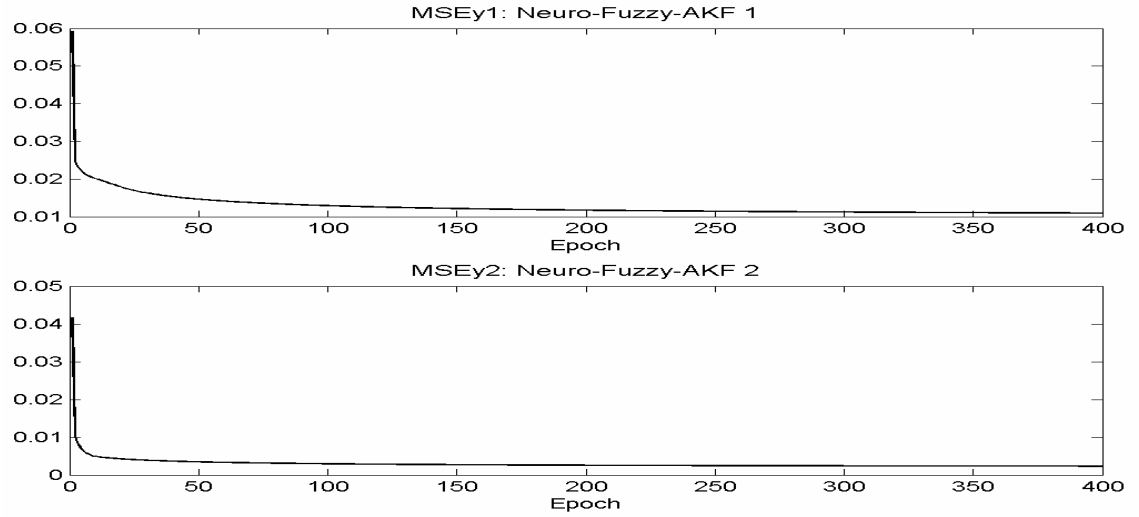


Figure 7.26 MSEy1 and MSEy2 during the training process.

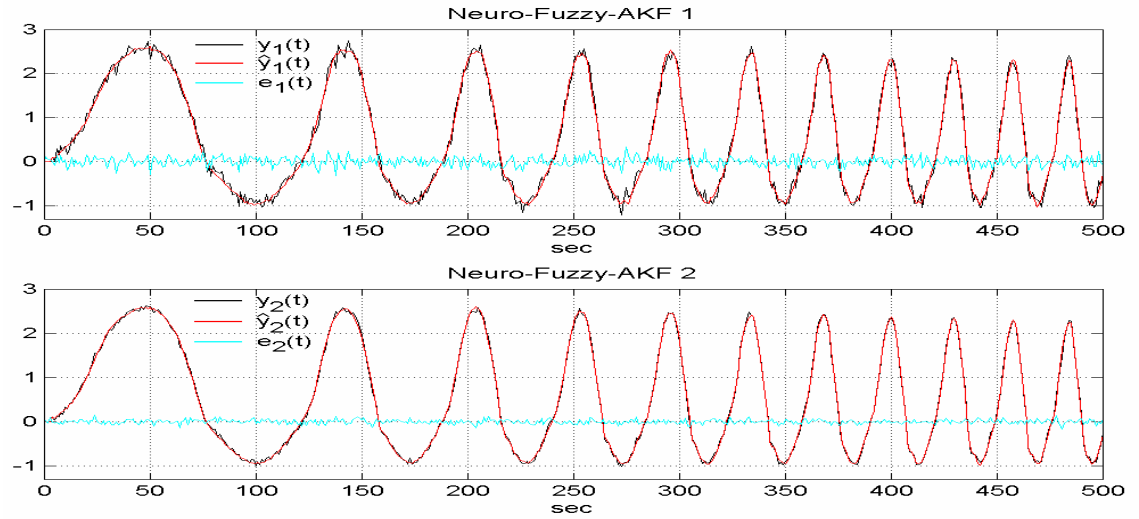


Figure 7.27 Performance of the Neuro-Fuzzy-AKF state estimators at the end of the training process.

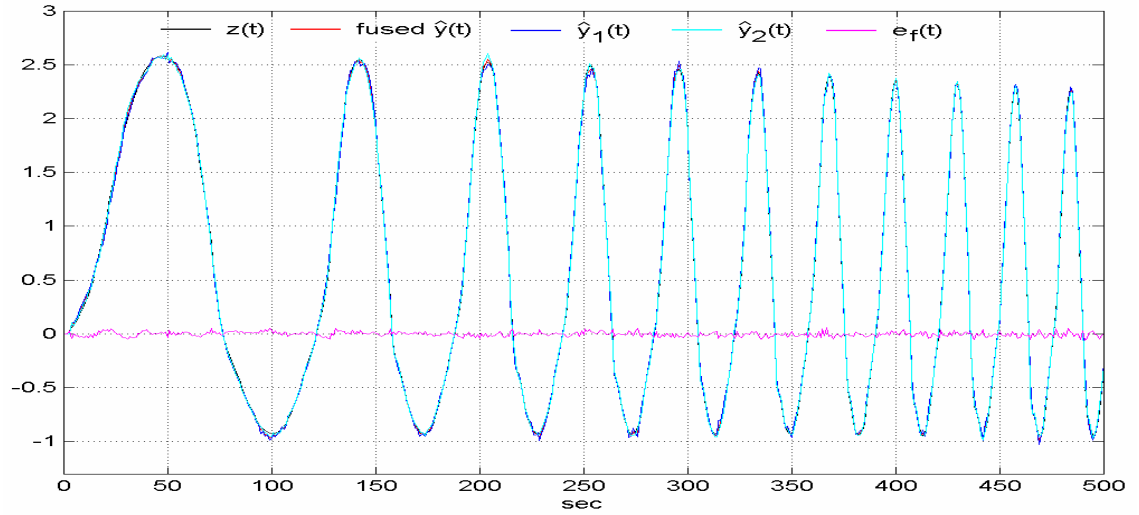


Figure 7.28 Performance of the FL-AKF-FLA fusion algorithm at the end of the training process.

The approximated measurement noise covariance values obtained at the end of the training process for both neuro-fuzzy-AKFs are shown in figure 7.29. Note that a quasi-steady state has been reached. The averaged measurement noise covariance values at epoch 500 were R_1=0.011 and R_2=0.0023, which are very near to the actual measurement noise covariance values 0.01 and 0.002, respectively. This data also supports the assertion that the training process has converged.

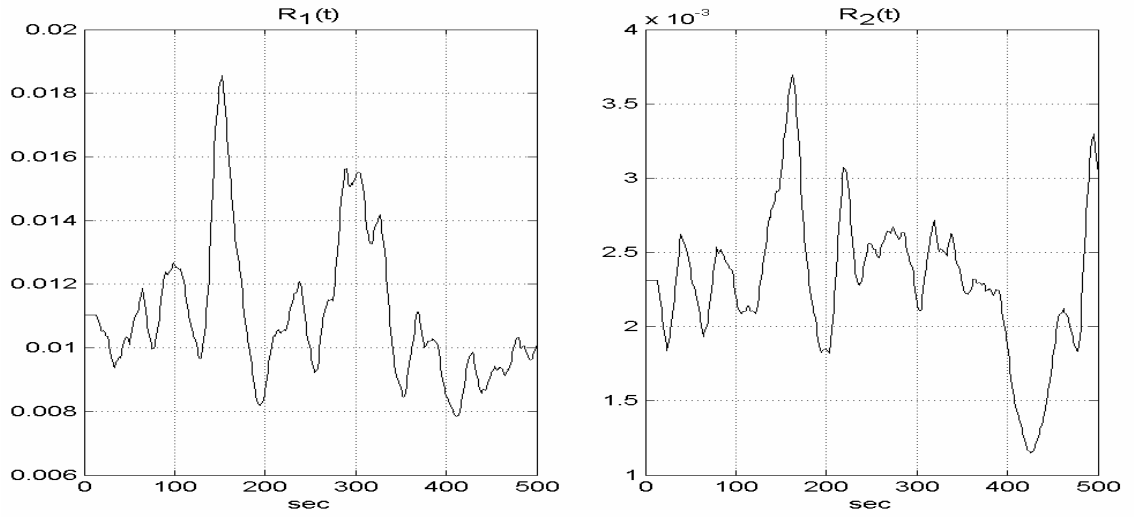


Figure 7.29 Measurement noise covariance values obtained at epoch 400.

A validation data set of 500 samples of $z(t)$, $y_1(t)$ and $y_2(t)$ was generated using as input the signal $u(t)=sin(\pi t/50)$ with sample time of 1 sec. The generated validation data is shown in figure 7.30.

The performance of both neuro-fuzzy-AKFs by using the validation data is shown in figure 7.31, where the estimated signals are compared with the measured signals. Note that good approximation is obtained. The obtained MSE*yi* values for the validation data are shown in Table 7.3. The estimated signals compared with the actual noise free signal are shown in figure 7.32. Note that in both filters a very good estimation of the actual signal is obtained, as is demonstrated by the *MSEzi* values shown in Table 7.3.

To appreciate the performance of the FL-AKF-FLA fusion algorithm, the fused signal, the actual signal, and both estimated signals are plotted in figure 7.33. The obtained *MSEf* value is shown in Table 7.3. Note that, in this case, the fused data is more accurate than the data obtained with any of the two sensors.

Finally, figure 7.34 shows the approximated measurement noise covariance value, obtained over the validation data. Note that practically the averaged value of R_i does not change, and the quasi-steady state is maintained.

Therefore, two simulated examples of neuro-fuzzy-AKF state estimation and system identification have been presented. System identification of two nonlinear systems using the series-parallel model has been performed. The identification process was carried out based on noisy signal coming from different sensors. It is worth remarking that by using a chirp signal (sine wave whose frequency varies linearly with time) as a training signal and the error signal as a measurement noise signal for the FL-AKFs inside the neuro-fuzzy-AKF structures, the training of the neuro-fuzzy-AKF using the series-parallel identification model is stable as was proved practically in the simulated examples. However, further investigation is needed to determine if this is true for a broader class of systems or to define under what conditions and for what kind of systems this is true. This task is left as future work to follow on from this research.

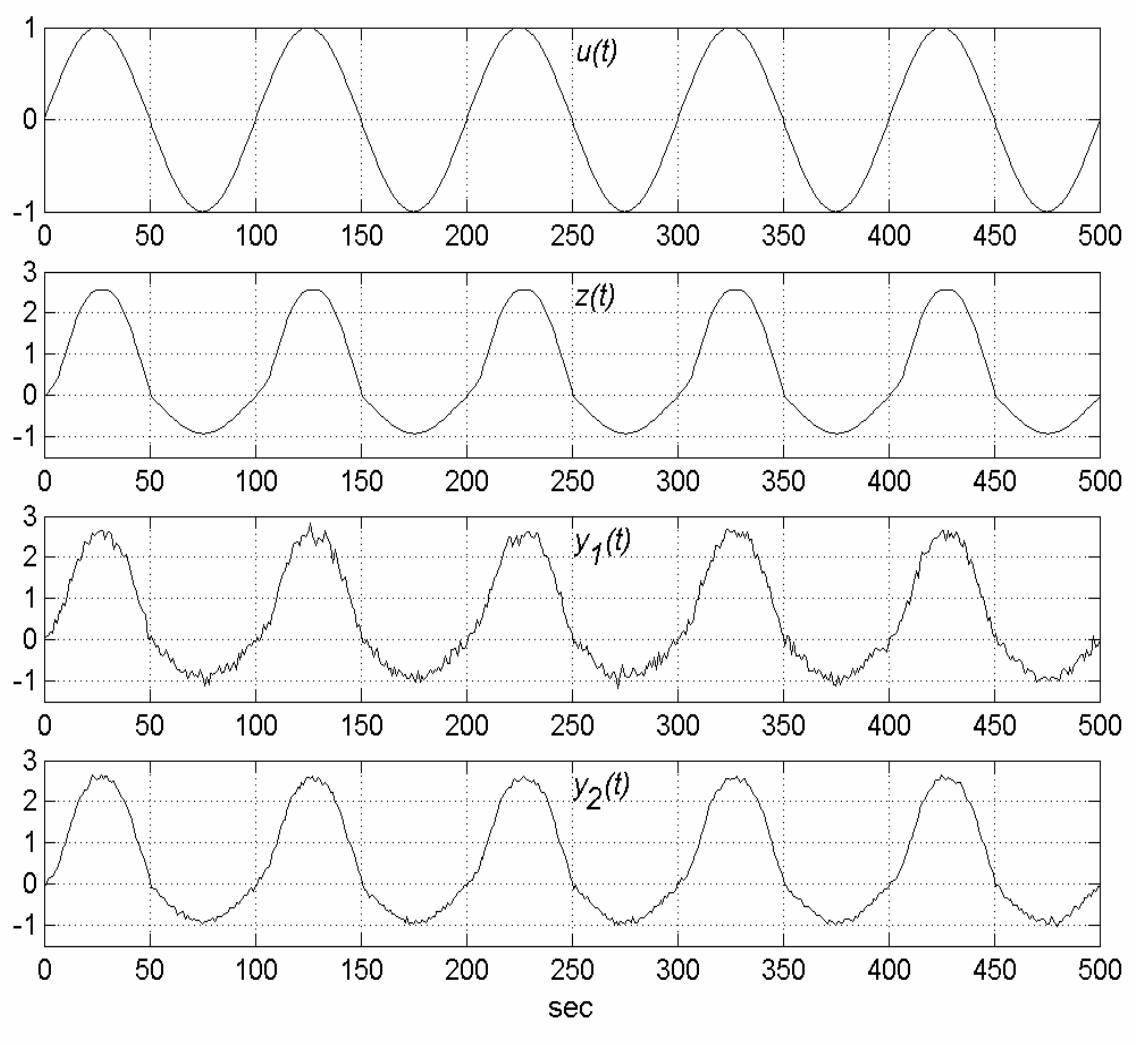


Figure 7.30 Validation data generated with $u(t)=sin(\pi t/50)$.

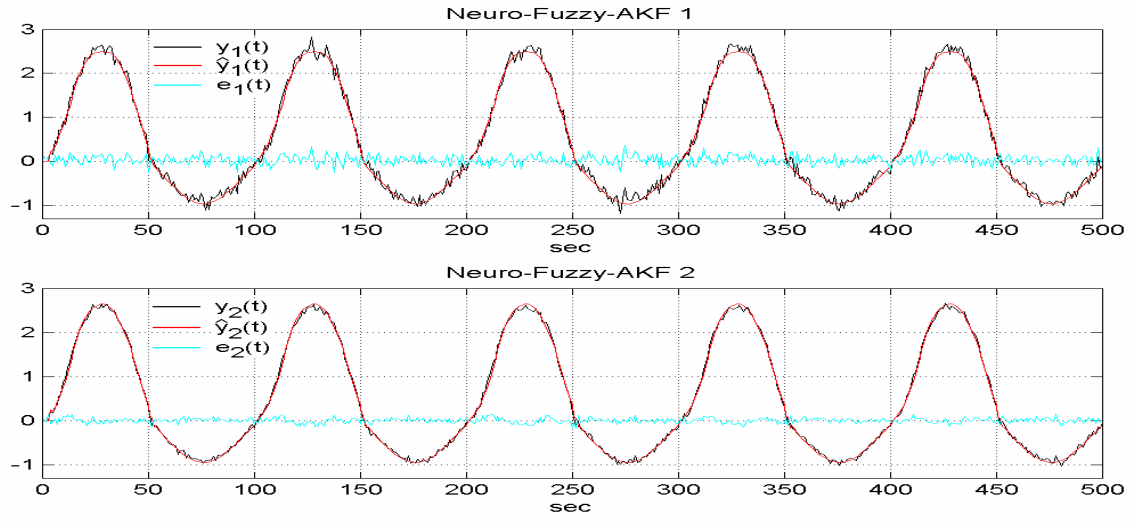


Figure 7.31 Actual and estimated signals over the validation data.

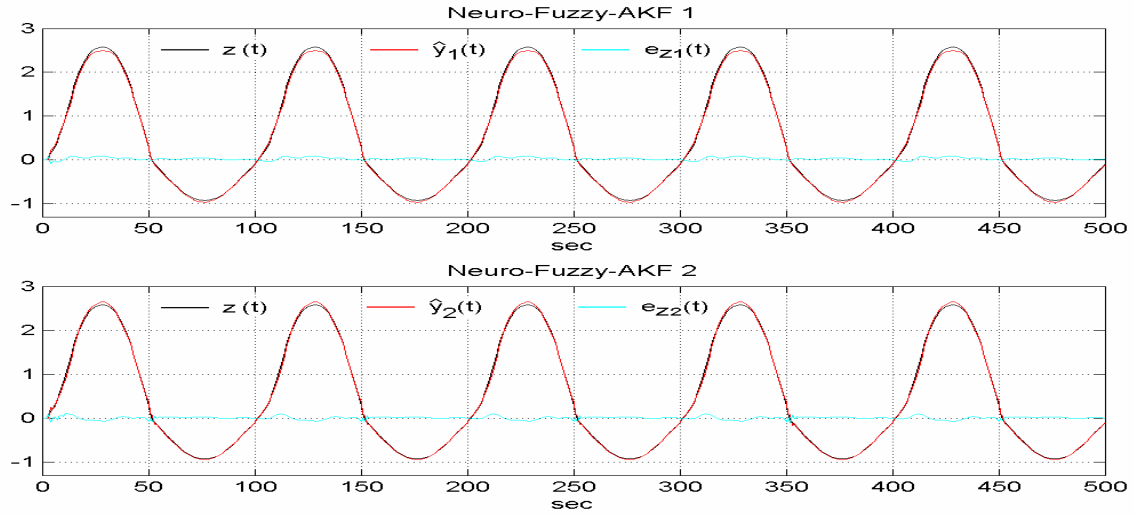


Figure 7.32 Performance of the neuro-fuzzy-AKFs compared with the actual signal.

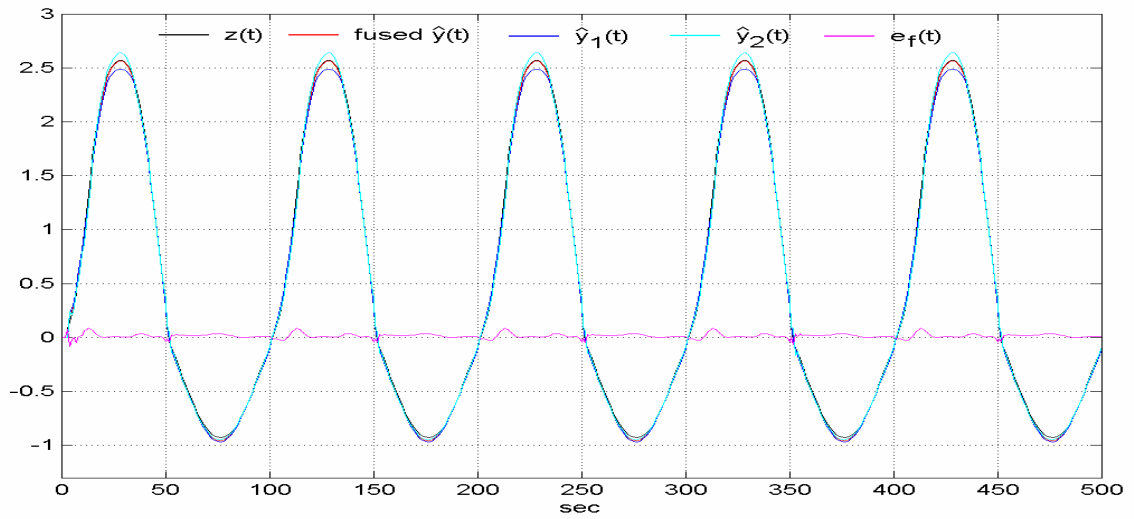


Figure 7.33 Fused and estimated signals compared with the actual noise free signal over the validation data.

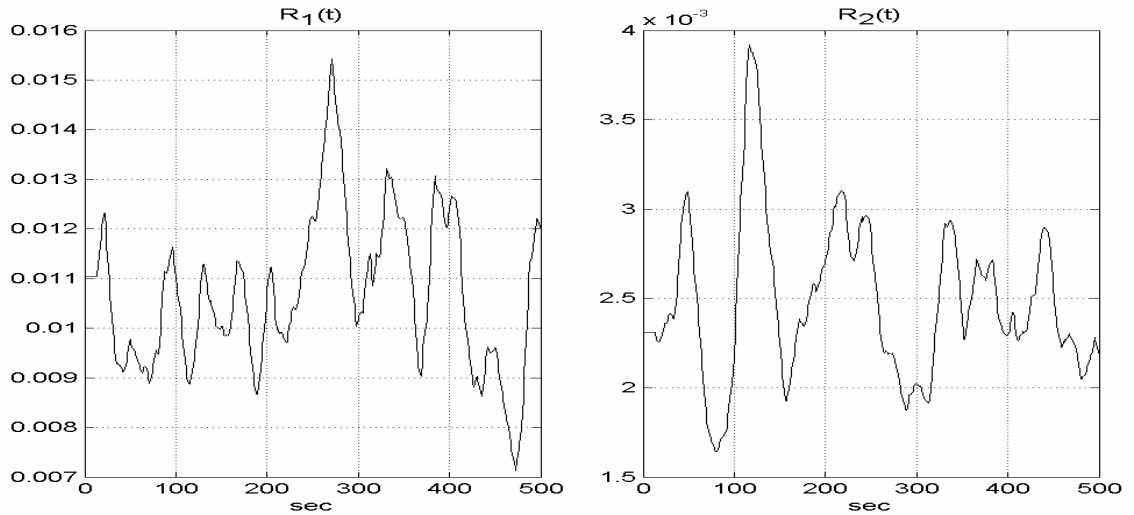


Figure 7.34 Measurement noise covariance values obtained during the validation process.

The identified systems were validated using the parallel model. In both of the examples presented, good approximation to the actual state variables were obtained. Note that, however, the identification was carried out using noisy signals and the accuracy of the estimates show that the noise is effectively filtered by the neuro-fuzzy-AKFs.

MSDF of the neuro-fuzzy-AKF state estimates was performed using the FL-AKF-FLA fusion algorithm. In example 1 the fused data, compared with the measured data, is 52% more accurate than sensor 1 which gives the most accurate measurements from the two sensors. However, if the fused data is compared with the estimated data, it is slightly less accurate than the estimates performed by the neuro-fuzzy-AKF 1, which is the more accurate of the estimators. Nevertheless, by having two sensors and two neuro-fuzzy-AKFs being fused through the FL-AKF-FLA algorithm gives the complete system the characteristic of fault tolerance against transient and persistent faults, as was found in Chapter 6. In example 2 the fused data is 79.6% more accurate than that given by sensor 2, which is the more accurate of the two sensors for this case. This data is also more accurate than the estimated data obtained with either of the neuro-fuzzy-AKF estimators. Particularly, the fused data is 45% more accurate than the data estimated by the neuro-fuzzy-AKF 2 which has the better performance of either sensor for this case.

7.7 Summary

In this chapter the neuro-fuzzy-SKF state estimator approach proposed by Harris *et al* [1999, 2000, 2002] has been reviewed. As a result of its analysis, a simplified version of the neuro-fuzzy-SKF has been proposed. A novel adaptive state estimator, referred to as neuro-fuzzy-AKF, has been proposed by substituting the SKF with a FL-AKF in the simplified neuro-fuzzy-SKF structure.

The neuro-fuzzy-AKF has as its main characteristic the possibility of using the error signal in the identification process as the measurement signal for the FL-AKF in order to estimate the modelling error at the same time in which the identification process is performed. This has the effect of stabilization during the training process.

Two simulated examples of neuro-fuzzy-AKF state estimation, system identification, and MSDF have been presented. The identification process was carried out based on noisy signals coming from different sensors and using a series-parallel model, while the identified models were validated using a parallel model. MSDF of the estimates performed by two neuro-fuzzy-AKFs were carried out using the FL-AKF-FLA algorithm presented in Chapter 6. Good results in both system identification and MSDF were obtained.

At the end of this chapter both modelling and estimation problems to improve the performance, reliability and accuracy of the Kalman filter approach and the MSDF architectures based on it have been studied. Solutions for both problems have been proposed and by simulating several examples it has been demonstrated that these solutions work very well.

In the next chapter the possibility of applying the proposed MSDF architectures in control systems will be studied. In particular, their application in the area of auto-tuning of PID type fuzzy logic controllers will be analysed.

CHAPTER 8

APPLICATION OF THE HYBRID MULTI-SENSOR DATA FUSION ARCHITECTURES IN CONTROL SYSTEMS

8.1 Introduction

Although the developed MSDF architectures can be applied to a broad range of problems, one application that is of great interest for the author is in control systems. In particular, the aim is to explore the application of the previously developed MSDF algorithms to design and tune PID-type fuzzy logic controllers when there are multiple noisy sensors measuring the process output. This chapter explores a way forward to achieve this aim. First, a novel designing and tuning procedure for PID-type fuzzy logic controllers (PID-FLC) is developed. Next, the PID auto-tuning procedure proposed by Astrom and Hagglund [1984] is extended and developed for tuning the scaling factors of a PID-FLC. Then, a novel procedure for auto-tuning the PID-FLC by using multiple noisy sensors is presented where the developed MSDF architectures of chapter 6 can be applied.

8.2 A novel design and tuning procedure for PID type fuzzy logic controllers

In recent years fuzzy logic controllers (FLC) have been widely used for industrial processes exploiting their heuristic nature associated with simplicity and effectiveness for both linear and non linear systems [Bonissone *et al*, 1995] [King and Mamdani, 1997]. In particular, several structures of PID-type FLC have been used (including PI and PD). As a consequence, research into this type of FLC has increased considerably. Lately, the research effort has been focused on the construction of an explicit link between the scaling factors of PID-type FLC (PID-FLC) and the three actions of traditional PID control (TPID). The direct result of this link would bring the possibility of applying the systematic design and tuning methods of TPID control to design and tune PID-FLC.

Several approaches have been reported in the fuzzy control literature establishing a link between TPID and PID-FLC [Mann, *et al*, 2001] [Li and Tso, 2000] [Xu, *et al*, 2000]. However, these have often resulted in complicated mathematical expressions and, moreover, some of the parameters involved are heuristically established and this heuristic is not specified. Indeed, the task of constructing the link is not an easy one. First of all, several structures of PID-FLC have been proposed, and it is necessary to select the one most suitable for the construction of the link. Second, based on the chosen structure, a clear and explicit relationship between the parameters that define this structure with the three control actions of TPID control have to be derived. And finally, the systematic design and tuning methods of TPID control have to be translated for designing and tuning the selected PID-FLC structure.

Based on the investigation of the relationship between the gains of TPID control and the scaling factors of a modified hybrid PID-FLC (MHPID-FLC), in this section a new methodology for designing and tuning PID-FLC is presented [Escamilla and Mort, 2002a]. First, in section 8.2.1, a review of the different structures of PID-FLC is carried out. Next, in section 8.2.2, the relationship between the proportional, integral and derivative actions from TPID control and the scaling factors of the MHPID-FLC is found through mathematical analysis and comparison. Then, in section 8.2.3, a design and tuning procedure for the MHPID-

FLC is proposed. Next, in section 8.2.4, the auto-tuning methodology proposed by Astrom and Hagglund [1984, 1995] is extended and developed for automatically tuning the scaling factors of the MHPID-FLC [Escamilla and Mort, 2002b]. It is shown how the scaling factors can be directly derived from the Ziegler-Nichols frequency response method. As a result, the performance of the MHPID-FLC will be better than, or at least as good as, that of its traditional counterpart. Finally, in section 8.2.4, the viability of this approach is demonstrated by simulating several benchmark processes taken from the literature.

8.2.1 Traditional PID and PID type fuzzy logic control structures

In traditional control the PI, PD and PID control algorithms are expressed in discrete time as (to avoid confusion, in this section the symbol $*$ means multiplication):

$$u_{PI_k} = K_P * e_k + K_I * \sum_{i=0}^{k} e_i * Ts = K_P * \left(e_k + \frac{1}{T_i} * \sum_{i=0}^{k} e_i * Ts \right) \tag{8.1}$$

$$u_{PD_k} = K_P * e_k + K_D * \frac{ce_k}{Ts} = K_P * \left(e_k + K_D * \frac{ce_k}{Ts} \right) \tag{8.2}$$

$$u_{PID_k} = K_P * e_k + K_I * \sum_{i=0}^{k} e_i * Ts + K_D * \frac{ce_k}{Ts} = K_P * \left(e_k + \frac{1}{T_i} * \sum_{i=0}^{k} e_i * Ts + T_d * \frac{ce_k}{Ts} \right) \tag{8.3}$$

$$e_k = y_{r_k} - y_k \tag{8.4}$$

$$ce_k = e_k - e_{k-1} \tag{8.5}.$$

where e is the error signal, ce is the change in error, y_r is the set point, y is the process output, the subscript k indicates the instant of time, $T_i = K_P/K_I$, and $T_d = K_D/K_P$. The terms K_P, K_I and K_D are referred to as the proportional, integral and derivative gains. The parameters T_i and T_d are known as the integral time and the derivative time, respectively. Ts is used to denote the sampling period of time.

As in traditional control, in fuzzy control there are the analogous structures of the PI type fuzzy logic controller (PI-FLC), PD type fuzzy logic controller (PD-FLC) and PID type fuzzy logic controller (PID-FLC). Their basic structures for discrete-time are shown in figure 1; inside these structures a fuzzy control system (FCS) develops the three well-known processes of fuzzification, rule evaluation and defuzzification [Driankov *et al*, 1993] [Lee, 1990]. The parameters GE, GCE, GCE1 and GCE2 are called the input scaling factors, while GU and GCU are called the output scaling factors. The PI-FLC and PD-FLC have been extensively studied [Lee, 1990] [Lee, 1993] [Jantzen, 1997] [Tang and Mulholland, 1987], and have achieved wide acceptance in both academic research and industrial applications. However, the PID-FLC is considered to be still at its early stage of development [Driankov, *et al*., 1993] [Li and Tso, 2000]. This is shown by the numerous recent research papers reporting the exploration of different PID-FLC structures [Jantzen, 1999] [Li and Tso 2000] [Mann, *et al*, 1999] [Woo, *et al*, 2000] [Xu, *et al*, 2000].

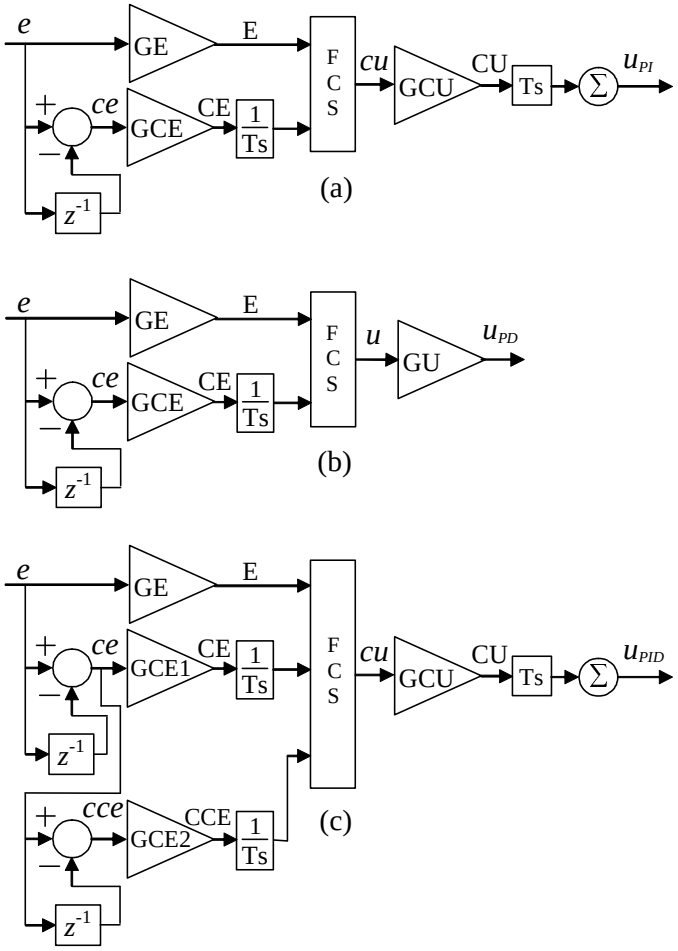


Figure 8.1 Structures for (a) PI-FLC, (b) PD-FLC, and (c) PID-FLC.

In the literature several PID-FLC structures have been proposed. Initially, the PID-FLC structures were designed considering three terms as inputs (see figure 8.1c) [Driankov, *et al*, 1993] [Abdelnour, *et al*, 1991]. Obviously, the rule base of these fuzzy controllers is three-dimensional (3-D), which makes it difficult to obtain since 3-D information is usually beyond the sensing capability of a human expert. To overcome this problem, the intuitive solution is the combination of a PI-FLC and a PD-FLC to form a PID-FLC. This idea has been developed basically in two ways, a parallel combination (PPID-FLC) and a hybrid combination (HPID-FLC).

The PPID-FLC structure was first proposed by Li and Gatland [1996], and lately has been studied by Xu *et al*, [2000]. In this structure the three-term PID-FLC is divided into two separate PI and PD parts. Thus two rule bases are used, one for a PI-FLC and one for a PD-FLC; the output is obtained by adding the respective crisp control output, as is shown in figure 8.2a. This structure has the advantage that both rule bases are two-dimensional avoiding the difficulty of designing a 3-D rule base. Consequently the design of a PID rule base becomes the design of both a PI and a PD rule base. These two rule bases share the same inputs, which reduces the tuning complexity.

The HPID-FLC structure was first proposed by Li [1997], and lately has been studied by Mann *et al*, [1999], Li and Tso [2000]. In the HPID-FLC structure a common two-dimensional rule base is employed. This rule base is shared for both the PI-FLC and the PD-FLC parts, as is shown in figure 8.2b. It means that, once appropriate scaling factors are selected, a PID control

strategy is implemented by combining a PI incremental algorithm and a PD positional algorithm using a two-term fuzzy control rule base without any increase in the number of rules. This simplifies the PID-FLC structure as it is simpler, easier to implement and faster in computation. The PI rule base is selected as the one used, because PI control is normally more important for steady state behaviour.

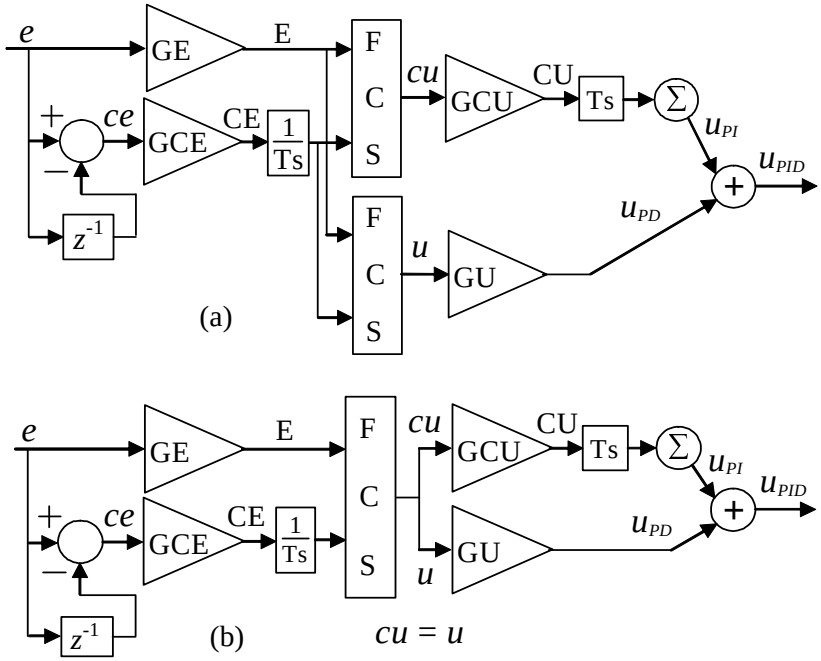


Figure 8.2 PI + PD-FLC structures for: (a) PPID-FLC, (b) HPID-FLC.

8.2.2 Mathematical analysis and comparison

First, in order to avoid derivative kick in the implementation of (8.3) a modified derivative term is used. Additionally, when the Ziegler-Nichols tuning formulae is applied generally a set point weighting factor β is employed to reduce overshoot [Astrom and Hagglund, 1995]; therefore, (8.3) is transformed as follows,

$$u_{PID_k} = K_P * \left[(\beta * y_{r_k} - y_k) + \frac{1}{T_i} * \sum_{i=0}^{k} e_i * Ts - Td * \frac{cy_k}{Ts} \right]$$
$$= K_P * \beta * y_{r_k} - K_P y_k + K_I * \sum_{i=0}^{k} e_i * Ts - K_D * \frac{cy_k}{Ts} \qquad (8.6).$$

Observe in (8.6) that the derivative term ce_k/Ts in (8.3) has been replaced by $-cy_k/Ts$. The incorporation of the above modification in the HPID-FLC structure modifies it as is shown in figure 8.3. This modified HPID-FLC (MHPID-FLC) structure is the one used in this approach.

Next, if the following assumptions are made:

1. The FCS inside the MHPID-FLC structure is a first-order Sugeno fuzzy model [Takagi and Sugeno, 1985], with fuzzy rules of the form:

   If *E* is A and *CE* is B then $u = p*E + q*CE + r$

where A and B are fuzzy sets in the antecedent, while *p*, *q*, and *r* are all constants.

2. The FCS rule base consists of four rules:

   R_1: If *E* is N and *CE* is N then $u = p_1*E + q_1*CE + r_1$
   R_2: If *E* is N and *CE* is P then $u = p_2*E + q_2*CE + r_2$
   R_3: If *E* is P and *CE* is N then $u = p_3*E + q_3*CE + r_3$
   R_4: If *E* is P and *CE* is P then $u = p_4*E + q_4*CE + r_4$

   where the coefficient constants $p_i = q_i = 1$, and $r_i = 0$; for $i = 1, 2, 3, 4$. The linguistic labels for the fuzzy sets mean P = Positive and N = Negative.

3. The universe of discourse for both FCS inputs is normalised on the range [-1, 1].

4. The membership functions of the input variables, *E* and *CE*, to the FCS are triangular complementary adjacent fuzzy sets [Escamilla, 1999] [Gravel and Mackenberg, 1995], and they are defined as shown in figure 8.4(a).

5. The product-sum compositional rule of inference [Kosko, 1992] is used in the stage of rule evaluation.

6. The weighted average is used in the defuzzification process.

then the FCS inside the MHPID-FLC structure is the simplest that can be considered, and its output is given by the sum of its inputs. This FCS is known as the normalised and linear FCS [Jantzen, 1999]; its control surface is shown in figure 8.4(b).

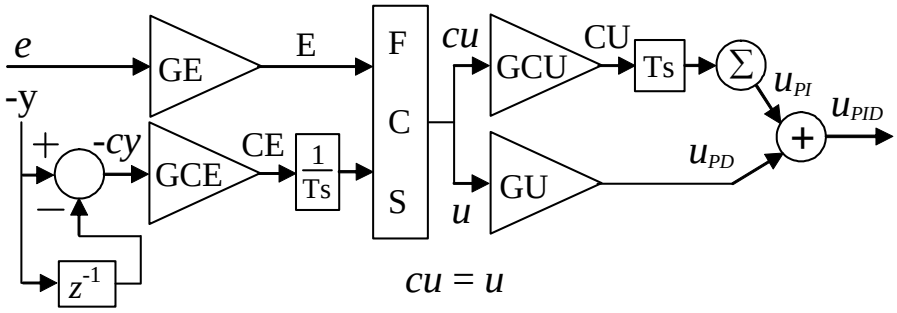


Figure 8.3 Modified HPID-FLC (MHPID-FLC) structure.

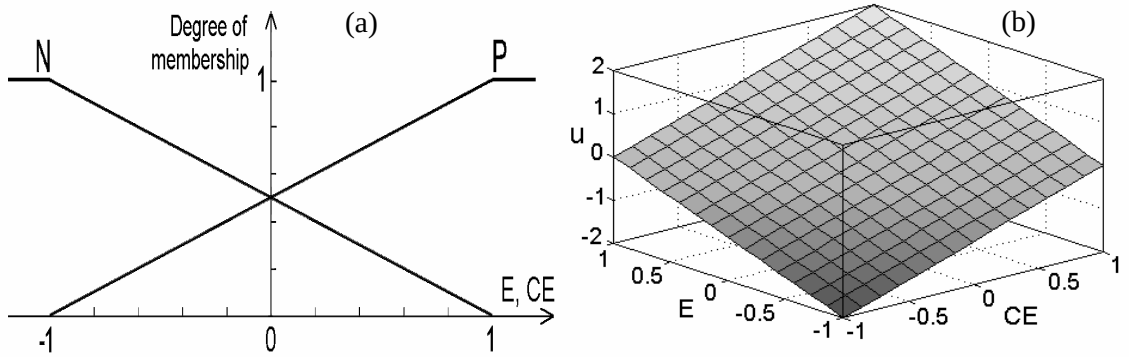


Figure 8.4 (a) Membership functions for E and CE; (b) Control surface of the normalised and linear FCS.

Thus, under the assumptions 1 to 6 the control output u_{PID} of the MHPID-FLC (see Figure 8.3) is the sum of the PI-FLC output and the PD-FLC output parts,

$$u_{PID_k} = u_{PI_k} + u_{PD_k} \tag{8.7}$$

but, each the PI and PD parts can be written as:

$$\begin{aligned} u_{PI_k} &= \sum_{i=0}^{k} \left(cu_i * GCU * Ts \right) = GCU * Ts * \sum_{i=0}^{k} cu_i \\ &= GCU * Ts * \sum_{i=0}^{k} \left[GE * e_i - GCE * \frac{cy_i}{Ts} \right] \end{aligned} \tag{8.8}$$

$$u_{PD_k} = u_k * GU = GU * \left[GE * e_k - GCE * \frac{cy_k}{Ts} \right] \tag{8.9}.$$

Substituting (8.8) and (8.9) in (8.7) results in:

$$\begin{aligned} u_{PID_k} &= u_{PI_k} + u_{PD_k} \\ &= GCU * GE * \sum_{i=0}^{k} e_i * Ts - GCU * GCE * \sum_{i=0}^{k} cy_i \\ &\quad + GU * GE * e_k - GU * GCE * \frac{cy_k}{Ts} \\ &= GCU * GE * \sum_{i=0}^{k} e_i * Ts - GCU * GCE * y_k \\ &\quad + GU * GE * (y_{r_k} - y_k) - GU * GCE * \frac{cy_k}{Ts} \\ &= GU * GE * y_{r_k} - (GCU * GCE + GU * GE) * y_k \\ &\quad + GCU * GE * \sum_{i=0}^{k} e_i * Ts - GU * GCE * \frac{cy_k}{Ts} \end{aligned} \tag{8.10}.$$

Therefore, if (8.10) and (8.6) are compared, then it is noted that the MHPID-FLC controller works like a TPID controller with set point weighting factor and modified derivative term. The equivalent set-point weight, proportional, integral and derivative gains are:

$$K_P * \beta = GU * GE \tag{8.11}$$

$$K_P = GCU * GCE + GU * GE \tag{8.12}$$

$$K_I = \frac{K_P}{T_i} = GCU * GE \tag{8.13}$$

$$K_D = K_P * T_d = GU * GCE \tag{8.14}.$$

This means that the scaling factors of the MHPID-FLC can be derived from the proportional, integral and derivative gains obtained for the traditional PID controller using well known methods, i. e. the Ziegler-Nichols tuning method [Astrom and Hagglund, 1995]. A procedure for this task is presented in next section.

8.2.3 Designing and tuning of the modified hybrid PID-FLC

If the values of K_P, K_I, and K_D or alternatively the values of K_P, T_i, and T_d are available, then the values of GE, GCE, GU and GCU in the MHPID-FLC structure (see figure 8.3) can be calculated in the following way. The proportional gain given by (8.12) can be separated in two parts:

$$\begin{aligned} K_P &= GCU * GCE + GU * GE \\ &= \alpha * K_P + (1-\alpha) * K_P \end{aligned} \tag{8.15}$$

from here it follows,

$$GCU * GCE = \alpha * K_P \tag{8.16}$$

$$GU * GE = (1-\alpha) * K_P \tag{8.17}.$$

From (8.11) and (8.17) it can be directly deduced that,

$$\beta = 1 - \alpha \tag{8.18}.$$

From assumption 3 it is clear that the possible values of E are in the range [-1, 1], thus in order to avoid saturation, GE is selected as:

$$GE = 1 \tag{8.19}.$$

In consequence, from (8.19), (8.17) becomes,

$$GU = (1-\alpha) K_P \tag{8.20}.$$

In a similar way, from (8.19), (8.13) becomes,

$$GCU = K_I \tag{8.21}.$$

Calculating GCE from (8.14) gives,

$$GCE = \frac{K_D}{GU} \tag{8.22a},$$

and from (8.20) in (8.22a) gives,

$$GCE = \frac{K_D}{(1-\alpha) * K_P} \tag{8.22b}.$$

Thus, once the parameter α is defined, the scaling factors can be calculated using Equations (8.19) to (8.22). But now the question is how should the parameter α be properly defined? First of all α has to satisfy (8.16) and (8.20), thus from (8.21) and (8.22b) in (8.16) gives,

$$K_I * \frac{K_D}{(1-\alpha) * K_P} = \alpha * K_P \tag{8.23}$$

and solving (8.23) for α gives,

$$-\alpha^2 + \alpha = \frac{K_I * K_D}{K_P^2} \tag{8.24}.$$

But, from traditional PID control,

$$K_I = \frac{K_P}{T_i}\ ; \quad K_D = K_P * T_d \tag{8.25}.$$

Thus, from (8.25) in (8.24) gives,

$$-\alpha^2 + \alpha - \frac{T_d}{T_i} = 0 \tag{8.26}$$

and applying the relation between T_i and T_d given by the Ziegler-Nichols frequency response tuning method, which formulae to calculate the PID parameter is given in Table 8.1, leads to,

$$-\alpha^2 + \alpha - \frac{1}{4} = 0 \tag{8.27}.$$

Finally, solving equation (8.27) results in,

$$\alpha_1 = \alpha_2 = \frac{1}{2} \tag{8.28}.$$

Therefore, by substituting the value of α in (8.20) and (8.22b), the solutions for GU and GCE become straightforward.

The previous development means that the MHPID-FLC is equivalent to its traditional counterpart given by (8.6) when β is selected as 0.5, calculated from (8.18), and the Ziegler-Nichols frequency response method is used to tune the controller. The formulation of the scaling factors in function of K_P, T_i, and T_d is straightforward. A summary of the relationship between the scaling factors of the MHPID-FLC and the gains of its traditional counterpart is given in Table 8.2.

Further, fine-tuning can be made based on the relationship between the scaling factors of the MHPID-FLC and the three control actions of traditional PID control. This fine-tuning, can be developed in two ways: a) by modifying the scaling factors, b) by modifying the control surface of the FCS inside the MHPID-FLC structure. These procedures are described in the following sections.

Table 8.1 PID parameters according to the Ziegler-Nichols frequency response method

K_P	T_i	T_d
$0.6*K_u$	$(1/2)*T_u$	$(1/8)*T_u$

Table 8.2 Relationship between the scaling factors of the MHPI-FLC and the gains of its traditional counterpart

GE	GCE	GU	GCU
1	$2 * \frac{K_D}{K_P}$	$\frac{K_P}{2}$	K_I
1	$2 * T_d$	$\frac{K_P}{2}$	$\frac{K_P}{T_i}$

8.2.3.a Fine-tuning the MHPID-FLC by modifying the scaling factors

The role of the scaling factors of the MHPID-FLC can be determined by analogy with the gains of the traditional PID controller. Assuming that the value of GE is fixed as 1 and using the information from Table 8.2, general guidelines for fine tuning the scaling factors of the MHPID-FLC can be given. Changing the value of GCU will affect both the proportional control term and the integral control term, see (8.12) and (8.13). Thus, increasing the value of GCU will produce a faster but less stable control. The opposite action will cause the opposite effect. Changing the value of GU affects both the proportional control term and the derivative control term, see (8.12) and (8.14). Therefore, increasing the value of GU produces both faster and more stable control. But this is only true up to a certain limit, if GU is raised above this limit then it will result in reduced stability in control. Decreasing the value of GU will produce the opposite effect. Finally, a change in the value of GCE will affect both the proportional control term and the derivative control term, see (8.12) and (8.14). Therefore, increasing the value of GCE causes a faster and more stable control. But, as for the case of GU, if GCE is raised above of certain limit the system will be destabilised. Additionally, because GCE is an input scaling factor, it has to be manipulated carefully to avoid saturation. It is recommended to first adjust the output scaling factors, and if necessary, adjust GCE afterwards. A summary of the whole analysis is presented in Table 8.3.

Table 8.3 Effects of the scaling factors on speed and stability

	Speed	Stability
GCU increases	increases	reduces
GU increases	increases	increases
GCE increases	increases	increases

8.2.3.b Fine-tuning the MHPID-FLC by modifying the control surface of the FCS inside the structure

The main advantage of considering a first-order Takagi-Sugeno FCS inside the MHPID-FLC structure is that by changing the values of the consequent parameters, *p*, *q* and *r* in the fuzzy rules, the FCS control surface is modified. This means that by changing the FCS control surface, without modifying the scaling factors found initially, the strength of the three PID control actions can be regulated. For example, figure 8.5 shows the FCS control surface obtained with modified consequent parameters: $p_1=p_4=2.5$, $q_1=q_4=3$, $r_1=r_4=0$, $p_2=p_3=0.4$, $q_2=q_3=0.4$, $r_2=r_3=0$. With these consequent parameters the strength of the control action is increased at the extremes, when E and CE are larger, and reduced when E and CE are near zero, near the steady state. Additionally, note that the transition between a stronger and a weaker control action is made smoothly. In effect, it means that a kind of gain scheduling is obtained. However, the modification of the consequent parameters makes the FCS control surface non-linear, and so they have to be manipulated carefully.

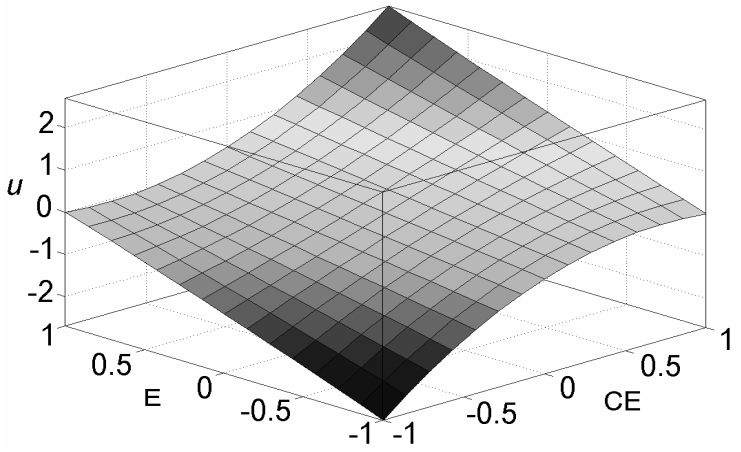


Figure 8.5 FCS control surface with modified consequent parameters.

8.2.4 Auto-tuning of the scaling factors of the MHPID-FLC

Based on the development given in section 8.2.3, the relay auto-tuning algorithm for TPID control proposed by Astrom and Hagglund [1984, 1995] can be extended and developed to auto-tune the scaling factors of the MHPID-FLC as is explained next. First of all the scaling factors have to be a function of K_u and T_u (see Table 8.1), thus from (8.20), (8.21), and (8.22) we have,

$$GU = \frac{K_P}{2} = \frac{0.6 * K_u}{2} = 0.3 * K_u \tag{8.29}$$

$$GCU = \frac{K_P}{T_i} = \frac{0.6 * K_u}{0.5 * T_u} = 1.2 * \frac{K_u}{T_u} \tag{8.30},$$

$$GCE = 2 * \frac{K_P * T_d}{K_P} = 2 * T_d = \frac{2}{8} * T_u = \frac{1}{4} * T_u \tag{8.31}.$$

The values of K_u and T_u, called the "ultimate gain" and "ultimate period" respectively, can be obtained from a relay feedback experiment as shown in figure 8.6. Therefore, the ultimate gain and the ultimate frequency can be calculated from this experiment as,

$$K_u = \frac{4 * h}{\pi * a} \quad ; \quad T_u = \frac{2 * \pi}{\omega_u} \tag{8.32}$$

where h is the relay amplitude, a is the process output amplitude, and ω_u is the oscillation frequency of the process output. It has been shown by Astrom and Hagglund [1984] that the simple estimation of K_u and T_u based on zero-crossing and peak detection works very well. Thus this method is used in this procedure and the values found are used to calculate the scaling factors of the MHPID-FLC. A summary of the relationship between the scaling factors of the MHPID-FLC and the Ziegler-Nichols frequency response tuning formulae is given in Table 8.4.

Table 8.4 Relationship between the scaling factors of the MHPID-FLC and the Ziegler-Nichols frequency response tuning formulae

G E	**G C E**	**G U**	**G C U**
1	$\frac{1}{4} * T_u$	$0.3 * K_u$	$1.2 * \frac{K_u}{T_u}$

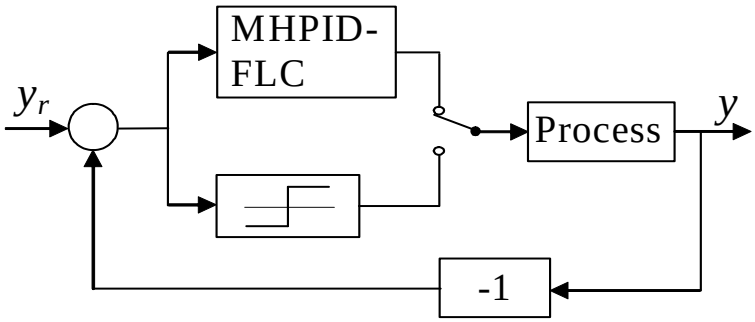


Figure 8.6 Relay feedback experiment.

8.2.5 Simulation and comparisons

In this section the viability of this approach is demonstrated by simulating several benchmark processes taken from the literature. Three auto-tuning experiments for each process have been developed in Matlab/Simulink environment, together with the Fuzzy Logic Toolbox. The first and second experiments use the relay experiment to auto-tune the gains of the TPID control given by equation (8.3), and the TPID given by equation (8.6) (referred to as TPID2) with a set-point weighting factor β = 0.5. In both these cases the tuning formulae given in Table 8.1 is applied. The third experiment is developed to simulate a relay auto-tuning procedure for the MHPID-FLC. Here the scaling factors are obtained applying the formulae given in Table 8.4.

After a relay experiment, a unit step and a unit load perturbation are introduced on the processes in order to observe their responses. The process responses under auto-tuned TPID, TPID2, and MHPID-FLC, control are plotted and compared for each case as is described as follows.

1) First-order plus dead time process [Hang *et al*, 1991]:

$$G_1(s) = \frac{e^{-0.2s}}{(s+1)} \tag{8.33}.$$

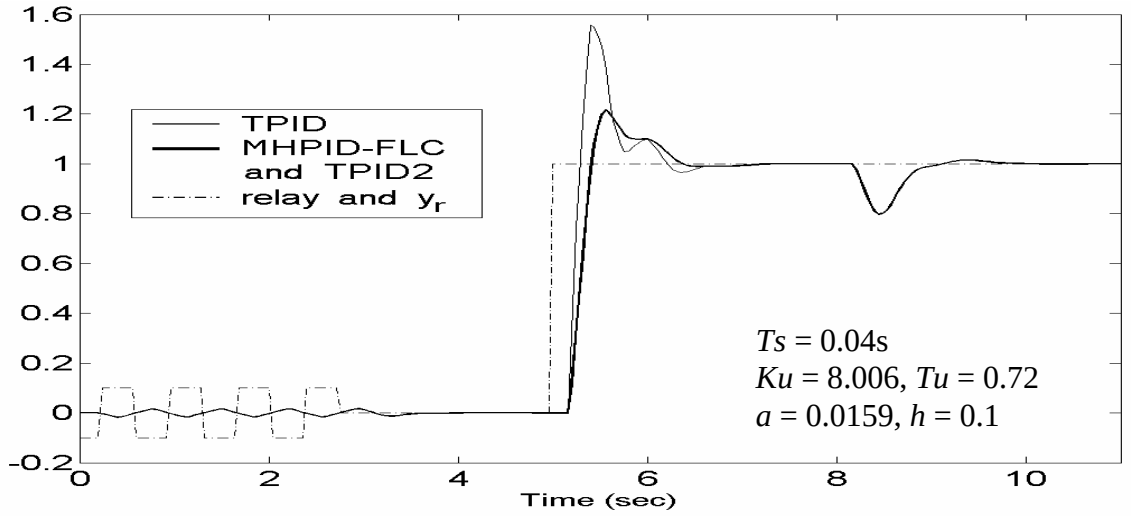


Figure 8.7 Comparison of set-point and load disturbance responses for $G_1(s)$.

2) Second-order plus dead time process (Hang, *et al*., 1991; Zhuang and Atherton, 1993):

$$G_2(s) = \frac{e^{-0.4s}}{(s+1)^2} \tag{8.34}.$$

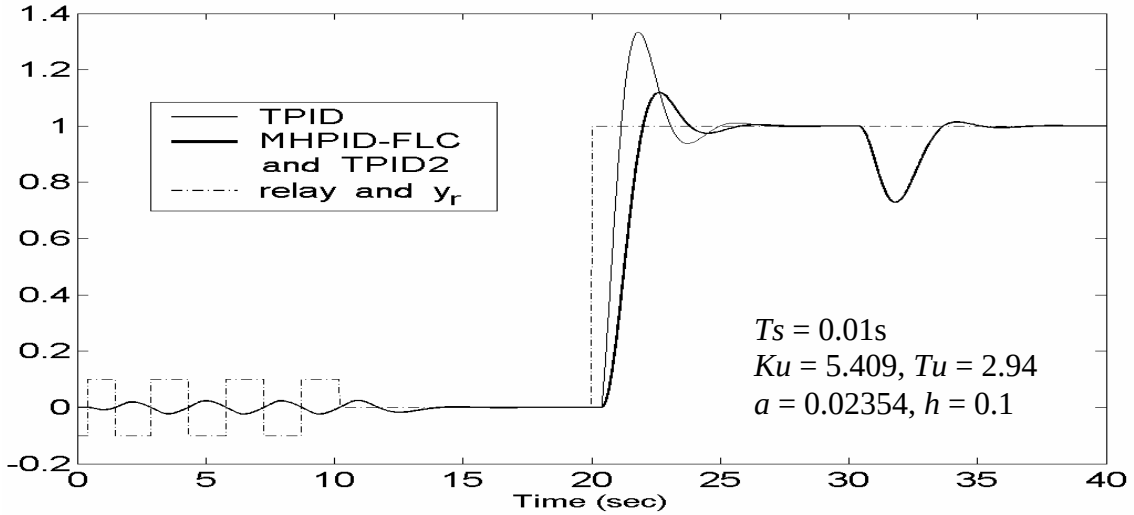


Figure 8.8 Comparison of set-point and load disturbance responses for $G_2(s)$.

3) High-order process (Zhao, *et al*., 1993):

$$G_3(s) = \frac{27}{(s+1)(s+3)^3} \tag{8.35}.$$

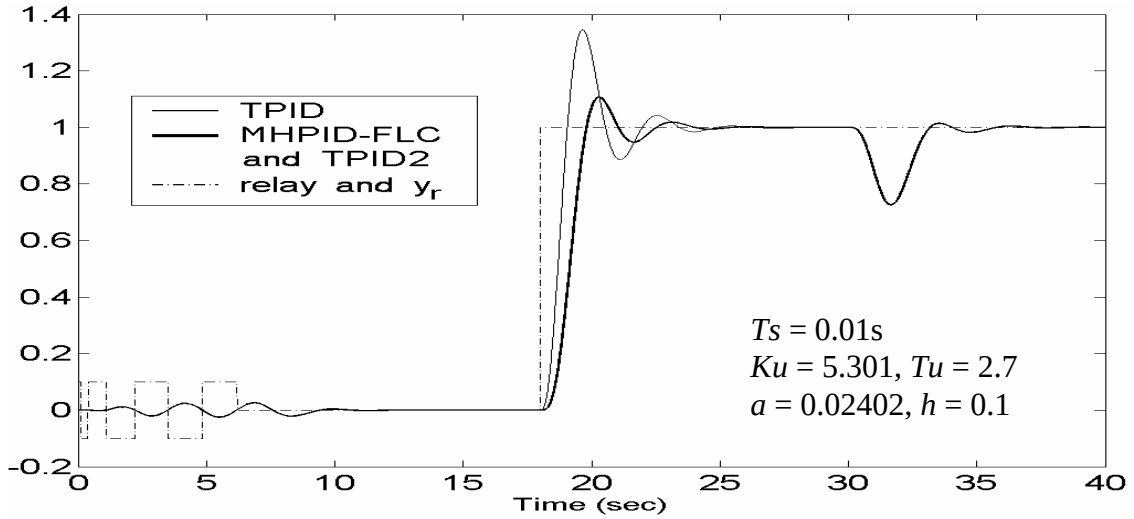


Figure 8.9 Comparison of set-point and load disturbance responses for $G_3(s)$.

4) Non-minimum phase process (Hang, *et al*., 1991):

$$G_4(s) = \frac{1-1.4s}{(s+1)^3} \tag{8.36}.$$

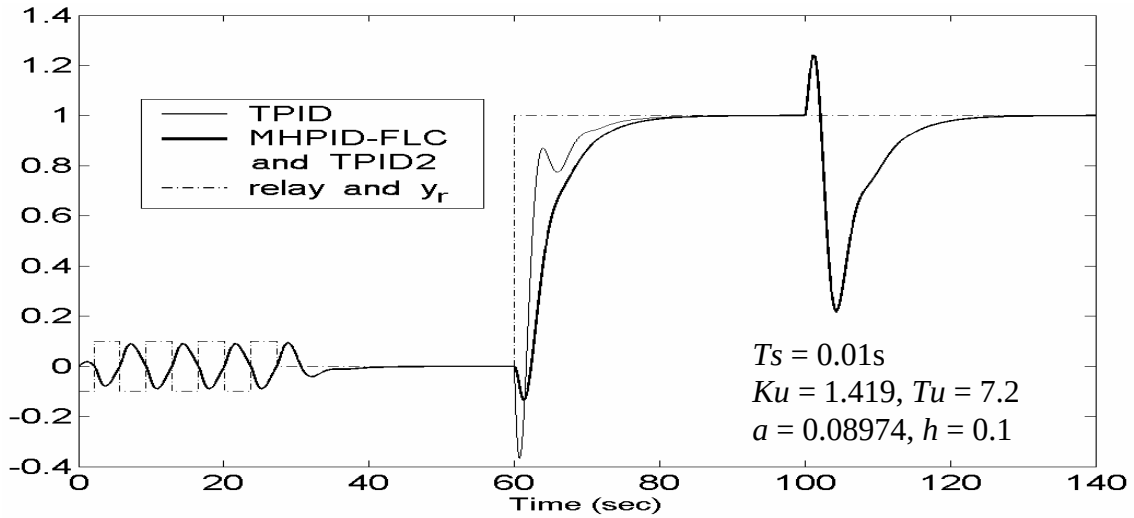


Figure 8.10 Comparison of set-point and load disturbance responses for $G_4(s)$.

From the results observed in figures 8.7 to 8.10 it is noted that the overshoot in the step-response (undershoot for process G_4) is excessive when TPID control is used. But, the overshoot is reduced by approximately 60% when MHPID-FLC and PID2 control are employed. However, this reduction in overshoot is accompanied by a small reduction in the speed of response (the rise time increases). Note that exactly the same response is obtained when MHPID-FLC and TPID2 control are used. Thus it is proved that the MHPID-FLC is equivalent to its traditional counterpart given by (8.6) when β is selected as 0.5. This means that the set-point weighting factor is embedded in the MHPID-FLC structure. Note that in all cases the load disturbance rejection is the same. Thus the MHPID-FLC and the TPID2 controllers sacrifice speed of response to a far smaller degree in order to obtain a substantial reduction in overshoot with respect to TPID control. However, this does not affect the load disturbance response.

Note that in all the processes considered, good control performances were obtained. However, if necessary, fine tuning can be performed by modifying the scaling factors or by modifying the FCS control surface inside the MHPID-FLC structure. This latter procedure will be exemplified in the next section.

8.3 Auto-tuning and MHPID-FLC using multiple noisy sensors

There are several practical problems that must be solved in order to implement an auto-tuner. In the development presented in the previous sections, some of these practical problems have not been taken into account. It is necessary, for example, to account for measurement noise. In real processes, the measurement noise in sensor devices is unavoidable so, therefore, any practical method of auto-tuning should be able to overcome measurement noise.

Measurement noise represents disturbances that distort information about the process variables obtained from the sensors. As for the measurement noise in the relay test discussed previously, these disturbances may give errors in detection of peaks and zero-crossings. As a consequence, when using an ordinary relay in the experiment, a small amount of noise can make the relay switch randomly.

In the context of system identification, noise is also a significant issue. It is apparent that in almost all identification methods a low noise-to-signal ratio is required [Ljung, 1987]. In system

identification, noise-to-signal ratio [Haykin, 1989] is usually defined as:

$$\text{noise-to-signal power spectrum ratio} = \frac{\text{mean power spectrum density of noise}}{\text{mean power of signal}} \tag{8.37}$$

denoted by N_1 or:

$$\text{noise-signal mean ratio} = \frac{\text{mean (abs(noise))}}{\text{mean (abs(signal))}} \tag{8.38}$$

denoted by N_2.

In considering the influence of noise in the controller performance, measurement noise will be fed into the system through feedback. It will generate control actions and control errors. Furthermore, high frequency components in the measurement signal might be amplified by the controller and cause wear on the actuator [Astrom and Hagglund, 1995]. Therefore, care should always be taken to reduce noise by appropriate filtering. Because the measurement noise is usually of high frequency, a low pass filter generally is used to reduce the measurement noise effects.

Several solutions have been given to the measurement noise issue in the relay feedback experiment. For example, Astrom and Hagglund [1984] pointed out that a hysteresis in the relay is a simple way to reduce the influence of measurement noise. The width of hysteresis should be bigger than the noise band [Astrom and Hagglund, 1995], and it is usually chosen as 2 times larger than the noise amplitude [Hang *et al*, 1993].

In this section a novel approach to deal with the noise issue in both the auto-tuning procedure and the control performance for the MHPID-FLC in a multi-sensor environment is proposed. The basic idea consists of combining the recent low-order modelling method proposed by Wang *et al* [1997] with the FL-ADKF approach, as is shown in figure 8.11. It is assumed that multiple sensors, which may have different accuracy levels (different measurement noise amplitudes), are used to determine the process output. The idea of using multiple sensors is to have a control system which can operate with good accuracy even when the measurement noise level is very high and with different characteristics for each sensor. The character of the noise is defined by its frequency; it may be high-frequency fluctuations or it may be low-frequency calibration errors. With several sensors it is possible to reduce calibration errors but with only one sensor nothing can be done about calibration errors [Astrom and Hagglund, 1995]. Therefore, it is desirable to develop an auto-tuning procedure and a PID controller considering multiple sensors.

The scheme shown in figure 8.11 consists of several functional blocks. A biased relay feedback experiment is used to find the process critical point information and the steady-state gain. A noise amplitude analyser and signal selector is used to estimate the noise bands and the noise covariance in each sensor. Based on the noise bands, this block selects the signal with the least noise band to perform the relay experiment with it. The data obtained from the biased relay experiment is used by a model identifier to approximate the process transfer function as a first order plus dead-time. The obtained transfer function is transformed to its discrete state-space representation. This state-space model is used by *N* FL-AKFs, which are fed by the respective noisy process output signal *yvi*, and adaptive decentralised Kalman filtering is performed. Thus, a FL-ADKF fuses and filters all the noisy measurement signals. The fused estimated process output $\hat{y}$ is used as measurement signal to compare with it the reference signal and calculate the error signal, which is fed to the MHPID-FLC. In the next section the model identifier and

translator to state-space representation is described. After that, the noise amplitude analyser and signal selector is explained. Then, the complete identification and auto-tuning procedure is summarised.

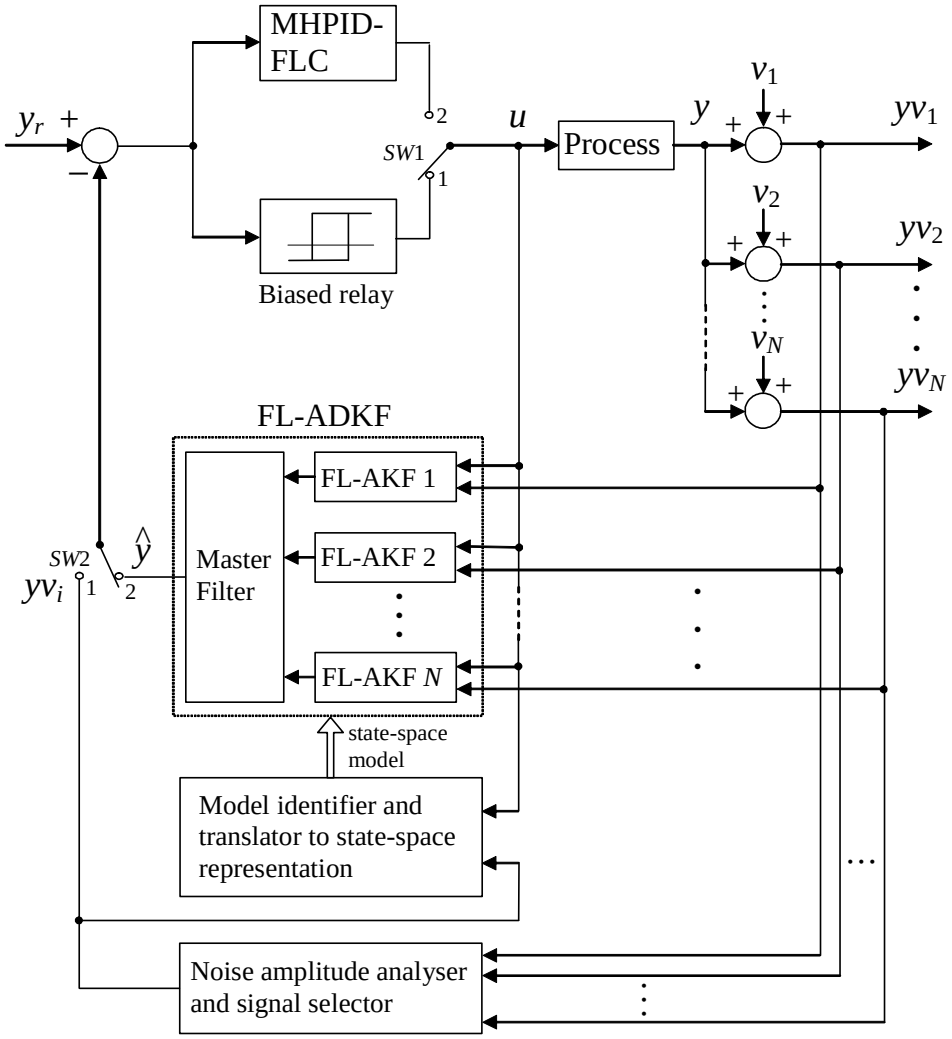


Figure 8.11 Auto-tuning the MHPID-FLC by using multiple noisy sensors.

8.3.1 Model identifier and translator to state-space representation

A large number of processes can be characterised by the first order plus dead-time model [Seborg *et al*, 1989]:

$$G(s) = \frac{Ke^{-Ls}}{Ts + 1} \tag{8.39}.$$

For these kinds of processes Wang *et al* [1997] have proposed a biased relay feedback test to derive the formulae that could precisely yield the critical point and the static gain simultaneously with a single relay test. The biased relay is shown in figure 8.12.

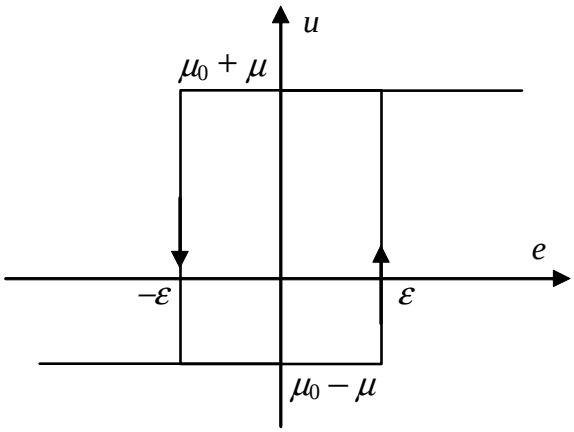


Figure 8.12 The biased relay.

Under the biased relay feedback, the process input *u* and the process output *y* is shown in figure 8.13. For the process given in (8.39) the output *y* converges to the stationary oscillation in one period (T_{u1}+ T_{u2}), and the oscillation is characterised by:

$$A_u = (\mu_0 + \mu)K(1 - e^{-L/T}) + \varepsilon\, e^{-L/T} \tag{8.40}$$

$$A_d = (\mu_0 - \mu)K(1 - e^{-L/T}) - \varepsilon\, e^{-L/T} \tag{8.41}$$

$$T_{u1} = T \ln \frac{2\mu K e^{L/T} + \mu_0 K - \mu K + \varepsilon}{\mu K + \mu_0 K - \varepsilon} \tag{8.42}$$

and

$$T_{u2} = T \ln \frac{2\mu K e^{L/T} - \mu K - \mu_0 K + \varepsilon}{\mu K - \mu_0 K - \varepsilon} \tag{8.43}.$$

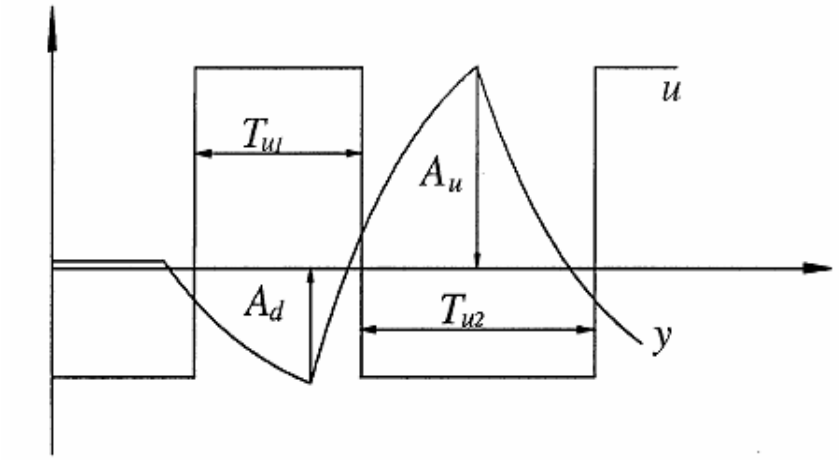


Figure 8.13 Oscillatory waveforms under a biased relay feedback (adapted from [Wang *et al*, 1997]).

The above four equations are the accurate expressions for the period and the amplitude of the limit cycle oscillation of the first order plus dead-time process. Therefore, by measuring any

three of A_u, A_d, T_{u1}, and T_{u2}, the parameters of the model K, T and L can be calculated from (8.40) to (8.43). Solving these equations is a tedious task. However, the calculations can be simplified if K is obtained by an alternative procedure. This procedure consists in calculating K as the ratio of DC components in the output and input:

$$K = \frac{\int_0^{T_{u1}+T_{u2}} y(t)\,dt}{\int_0^{T_{u1}+T_{u2}} u(t)\,dt} \tag{8.44}$$

Having available the value of K, the normalised dead-time of the process $\Theta = L/T$ can be obtained from (8.40) or (8.41) as:

$$\Theta = \ln\frac{(\mu_0+\mu)K-\varepsilon}{(\mu_0+\mu)K-A_u} \tag{8.45}$$

or

$$\Theta = \ln\frac{(\mu-\mu_0)K-\varepsilon}{(\mu-\mu_0)K+A_d} \tag{8.46}$$

It then follows from (8.42) and (8.43) that:

$$T = T_{u1}\left(\ln\frac{2\mu K e^{\Theta}+\mu_0 K-\mu K+\varepsilon}{\mu K+\mu_0 K-\varepsilon}\right)^{-1} \tag{8.47}$$

or

$$T = T_{u2}\left(\ln\frac{2\mu K e^{\Theta}-\mu K-\mu_0 K+\varepsilon}{\mu K-\mu_0 K-\varepsilon}\right)^{-1} \tag{8.48}$$

The dead time is thus calculated as:

$$L = T\Theta \tag{8.49}$$

Therefore, the previous development can be summarised as the following identification procedure:

Identification Procedure. The biased relay experiment is performed. The process input $u(t)$ and output $y(t)$ are recorded, and the periods and the amplitudes of the oscillation are measured. Then follow the next steps:

- Step 1: Compute K from (8.44).
- Step 2: Compute Θ from (8.45) or (8.46).
- Step 3: Compute T from (8.47) or (8.48).
- Step 4: Compute L from (8.49).

Wang *et al* [1997] presented several examples which demonstrated the accuracy of the method for both noisy and noisy-free sensors. If the process to be identified is of the form (8.39) and there is no measurement noise, then the parameters obtained with the biased relay method give an almost exact identification of the process parameters. Furthermore, because in practice many high-order processes can be well approximated by first-order plus dead-time models, the biased relay method can also be used to model processes of higher order. The results for some typical high-order processes also are presented in [Wang *et al*, 1997]. There, the Nyquist curves of the real processes and the obtained models are compared and it is observed that they are very close to each other over a phase range of 0 to π. Therefore, this low-order modelling is accurate enough for PID control design in most cases as will be practically demonstrated later.

For the case where there is noise in the measurements, the above method still proved to give good results. However, in this case the parameters K, A_u, and A_d have to be calculated by averaging over those values obtained over several cycles. In their paper, Wang *et al* [1997] used the values obtained over eight cycles of stationary oscillations.

Therefore, once the biased relay experiment is carried out, an approximated model of the process is available as a first-order transfer function. In order to use this model in the FL-AKFs (see figure 8.11) it is necessary to translate it to its state-space representation. This is performed in two stages. First the transfer function in continuous time is transformed to its corresponding state-space representation. Second, having available the continuous state-space representation, this is translated to its corresponding digital form. Both transformations can be directly performed by using only two commands in the MATLAB/Control Systems Toolbox environment. Thus, having available the process model in its digital state-space representation, this model can be used by the FL-ADKF to perform MSDF.

8.3.2 Noise amplitude analyser and signal selector

The noise amplitude analyser and signal selector performs several tasks. First, it determines the noise bands in each sensor. The noise band can be estimated by measuring the peak-to-peak amplitude of the output signal when the process is at steady-state [Astrom *et al*, 1993]. Second, an estimation of the measurement noise covariance values, R_i, of each sensor is performed over the data collected during a certain period of time (for this task a block from the MATLAB/DSP Block Set is employed). Finally, the signal with the minimum noise band is selected as the output signal of this block.

8.3.3 Identification and auto-tuning procedure using multiple noisy sensors

From the previous sections and referring to figure 8.11, the proposed identification and auto-tuning procedure is therefore summarised as follows:

1. *SW1* is in position 1; *SW2* is in position 1. First, in the “listening period”, 0-12 sec, the noise bands and the measurement noise covariance in each sensor are estimated. The sensor signal with the smallest noise band is selected to be fedback to the biased relay.

2. A biased relay is applied at time t = 12 sec.

3. Data is registered over five cycles of stationary oscillations. By averaging the values obtained over these five cycles, the parameter K, A_u, and A_d are calculated and the values of T_{u1} and T_{u2} are measured over the fifth cycle. With these parameters, the value of the

normalised process Θ is calculated using (8.45) or (8.46). Similarly, *T* is calculated from (8.47) or (8.48). Then, the dead time *L* is calculated from (8.49) and the process transfer function is modelled as a first-order plus dead-time. The obtained transfer function is transformed to its corresponding continuous and discrete state-space representations.

4. At the end of the fifth cycle all the FL-AKFs are activated using the state-space representation of the plant and MSDF is performed using the FL-ADKF; then the fused output is used as process output, *SW2* is switched to position 2. The initial conditions for the FL-AKFs are defined as $x_i(0) = 0$, $\hat{x}_i(0) = 0$, $i = 1,2,\ldots,N$. Because an estimation of the measurement noise covariance value R_i for each sensor has been obtained in step 1, these values are used in the corresponding FL-AKFs. Therefore, while the covariance values R_i are assumed to be known, they are not adapted in the FL-AKFs. Instead, the unknown values of the process noise covariance matrices Q_i, which represent the uncertainty in the process model, are the ones that are adaptively adjusted in the FL-AKFs. This will compensate for the modelling errors, recalling that the model used is an approximated model.

5. During the sixth cycle, the ultimate gain and the ultimate frequency are calculated as:

$$K_u = \frac{4\mu}{\pi(A_u + |A_d|)/2} \tag{8.50}$$

$$T_u = T_{u1} + T_{u2} \tag{8.51}$$

Note that (8.50) is similar to (8.32) with $h = \mu$ and $a = (A_u + |A_d|)/2$; where μ is the value of the relay amplitude when the bias is taken out, and a is the ultimate amplitude of the process output.

6. With K_u and T_u available, the scaling factors of the MHPID-FLC are calculated using the formulae given in table 8.4.

7. Finally, at the end of oscillation 6, *SW1* is switched to position 2 and the loop MHPID-FLC – Process is closed.

Afterwards, the performance of the controller can be investigated by introducing a set-point change and a load-disturbance at particular time steps. In order to test the effectiveness of the proposed approach, three examples are presented in the next section.

8.3.4 Illustrative examples

The viability of the previously described approach is demonstrated by simulating three processes taken from [Wang *et al*, 1997]. The experiments were developed under the Matlab/Simulink simulation environment (see Appendix B). It is assumed that there are two sensors in the scheme shown in figure 8.11. The measurement noise in each sensor, for all the experiments, is defined as a Gaussian zero-mean white noise sequence with variances 0.008 and 0.033 for v_1 and v_2, respectively.

Recall that the FCS inside the MHPID-FLC works with normalised inputs, in the range [-1, 1]. This normalisation is carried out by dividing the inputs between the maximum range of variation of the error signal, which in this case is assumed to be [-10, 10]. Therefore, the

normalisation factor is (1/10) applied to both inputs, *e* and *–y*. Obviously, the controller output needs to be denormalised; therefore, the controller output is multiplied by a denormalisation factor, 10 in this case.

The processes studied and the corresponding parameters obtained from the biased relay experiment are listed in Table 8.5. The scaling factors of the MHPID-FLC obtained from the auto-tuning procedure for each process are shown in Table 8.6. In order to analyse the set-point and load-disturbance responses, a step change of 10 units and a load disturbance, also of 10 units, are applied at appropriate time steps. The set-point and load-disturbance responses under MHPID-FLC for the plant in examples 1, 2 and 3 are shown in figures 8.14(a), 8.15(a), and 8.16(a), respectively. From these figures it can be noted that slightly sluggish set-point and load-disturbance responses are obtained. This is more noticeable in examples 2 and 3. However, as was mentioned in section 8.3.2.a, the control performance can be further improved by modifying the value of the consequent parameters, *p, q* and *r*, in the fuzzy rules of the FCS inside the MHPID-FLC structure. Therefore, to improve the control performance, the consequent parameters are modified as is indicated in table 8.7. The improved set-point and load-disturbance responses under MHPID-FLC for the plants in examples 1, 2 and 3 are shown in figures 8.14(b), 8.15(b), and 8.16(b), respectively. Note that the scaling factors found in the auto-tuning procedure are left unchanged.

Table 8.5 Estimated parameters from biased relay experiment

Example	Process	Biased relay test results				Model parameters		
		T_{u1}	T_{u2}	A_u	A_d	K	T	L
1	$\frac{e^{-2s}}{2s+1}$	3.35	4.0	1.714	-1.302	1.009	1.871	2.021
2	$\frac{e^{-2s}}{(2s+1)^2}$	5.55	6.45	1.543	-1.267	1.18	4.585	2.849
3	$\frac{e^{-0.5s}}{(s+1)(s^2+s+1)}$	2.95	3.35	2.106	-1.672	1.215	1.592	1.892

Table 8.6 Scaling factors obtained from the auto-tuning procedure

Example	Process	Scaling factors MHPID-FLC			
		GE	GCE	GCU	GU
1	$\frac{e^{-2s}}{2s+1}$	1	1.837	0.2757	0.5065
2	$\frac{e^{-2s}}{(2s+1)^2}$	1	3.0	0.1812	0.5437
3	$\frac{e^{-0.5s}}{(s+1)(s^2+s+1)}$	1	1.575	0.2568	0.4045

Table 8.7 Modified consequent parameters

Example	Process	Consequent parameters				Modified cons. parameters			
		Rule	p	q	r	Rule	p	q	r
1	$\frac{e^{-2s}}{2s+1}$	1	1	1	0	1	1.8	0.3	0
		2	1	1	0	2	0.4	0.4	0
		3	1	1	0	3	0.4	0.4	0
		4	1	1	0	4	1.8	0.3	0
2	$\frac{e^{-2s}}{(2s+1)^2}$	1	1	1	0	1	2.3	0.5	0
		2	1	1	0	2	0.4	0.4	0
		3	1	1	0	3	0.4	0.4	0
		4	1	1	0	4	2.3	0.5	0
3	$\frac{e^{-0.5s}}{(s+1)(s^2+s+1)}$	1	1	1	0	1	2.5	0.2	0
		2	1	1	0	2	0.1	0.1	0
		3	1	1	0	3	0.1	0.1	0
		4	1	1	0	4	2.5	0.2	0

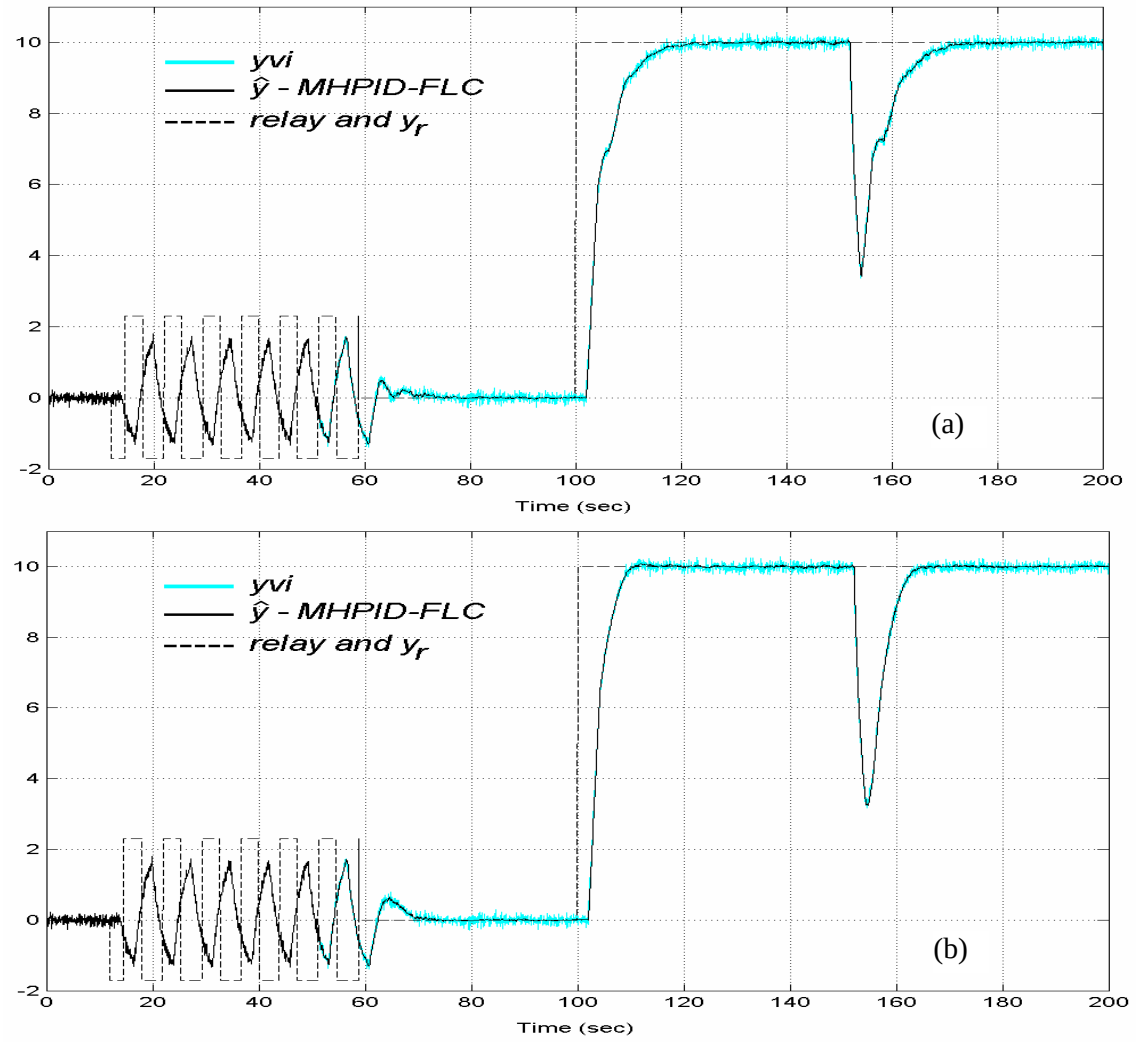


Figure 8.14 Set-point and load-disturbance responses for the plant in example 1, (a) with the original consequent parameters, (b) with the modified consequent parameters.

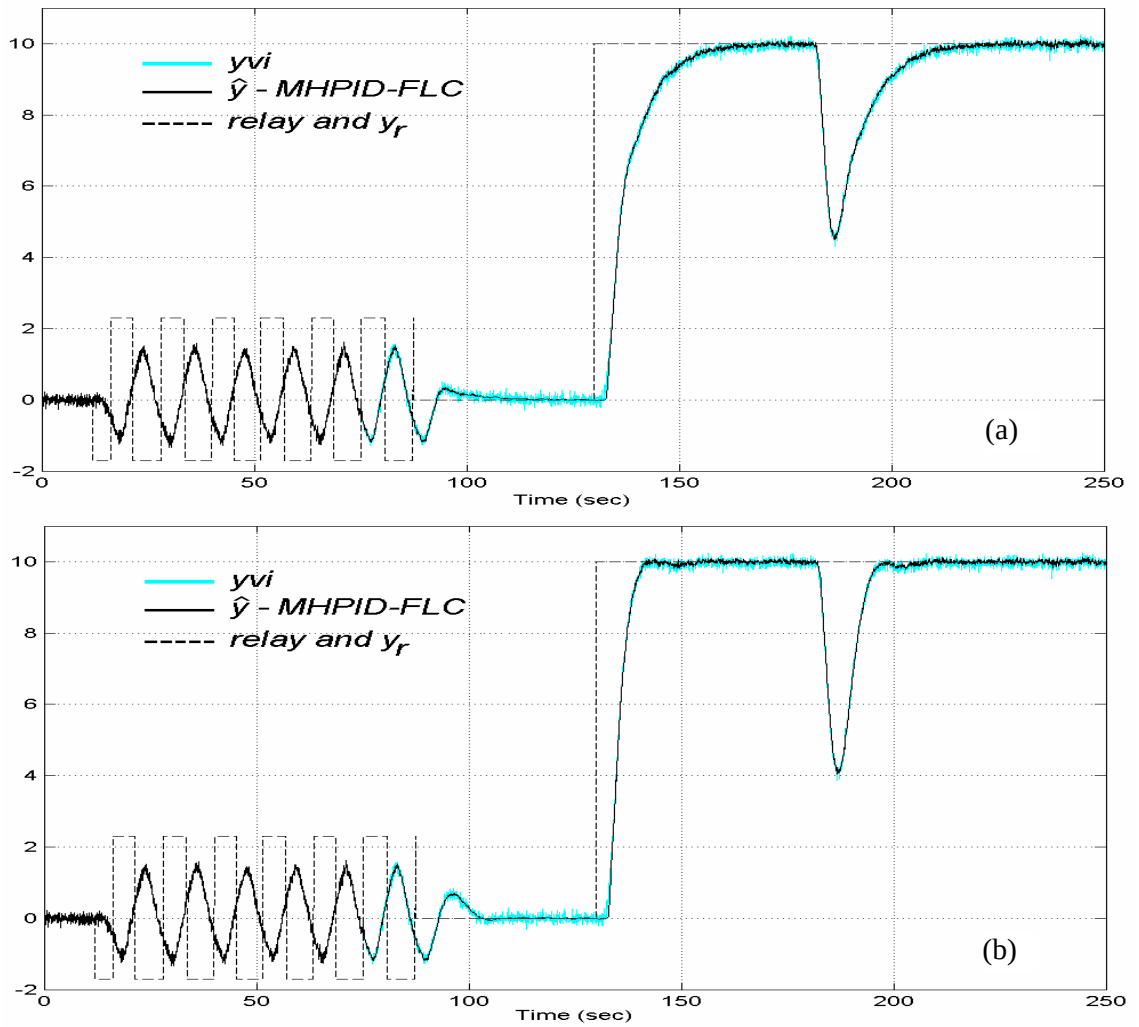


Figure 8.15 Set-point and load-disturbance responses for the plant in example 2, (a) with the original consequent parameters, (b) with the modified consequent parameters.

Note in figures 8.14 to 8.16 that as the order of the process increases, less noise is filtered by the FL-ADKF. In other words, this means that a quite accurate model is obtained when the plant is effectively of first-order. However, if the order of the plant increases, then the accuracy of the approximated model decreases. As a result, the value of the process noise covariance Q, which is adaptively adjusted, is increased to take into account this increased modelling error. This can be appreciated in figure 8.17, where the values of $R_1(t)$ and $Q_1(t)$ in the FL-AKF 1, fed by sensor 1, are plotted for each one of the examples. Remember that R and Q controls the bandwidth of the filter. Thus, while R is maintained constant, Q is constantly changing increasing or decreasing the bandwidth of the filter and, in consequence, increasing or reducing the filtering action.

Therefore, from the results obtained in the simulated examples, it was demonstrated that the described auto-tuning procedure is effective when there are multiple noisy sensors measuring the process output. Good results of MSDF and signal filtering also were obtained.

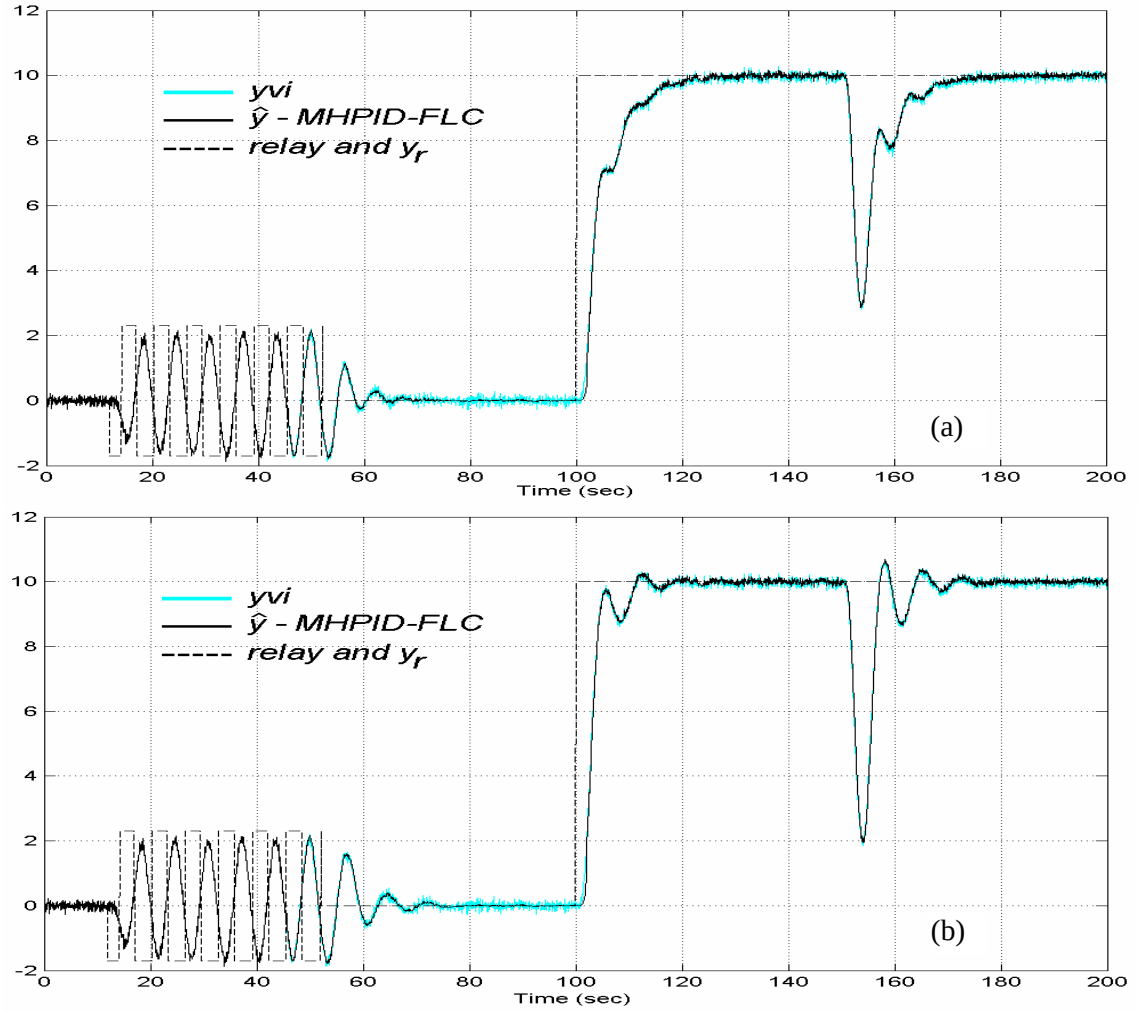


Figure 8.16 Set-point and load-disturbance responses for the plant in example 3, (a) with the original consequent parameters, (b) with the modified consequent parameters.

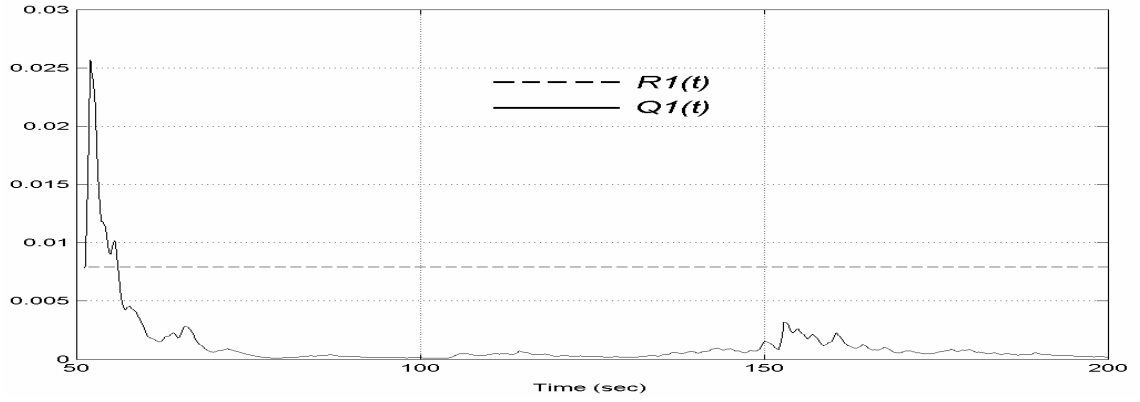


Figure 8.17 (a) Values of R_1(t) and Q_1(t) in the FL-AKF 1, fed by sensor 1, example 1.

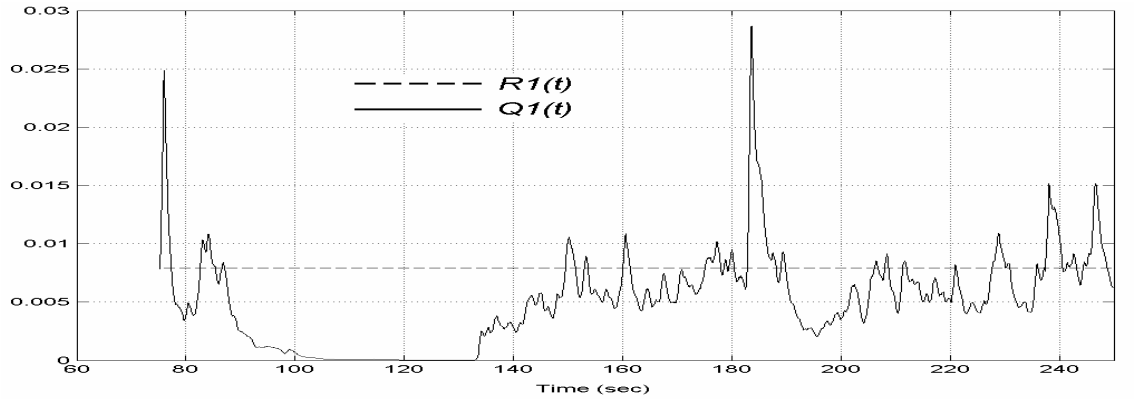


Figure 8.17 (b) Values of R_1(t) and Q_1(t) in the FL-AKF 1, fed by sensor 1, example 2.

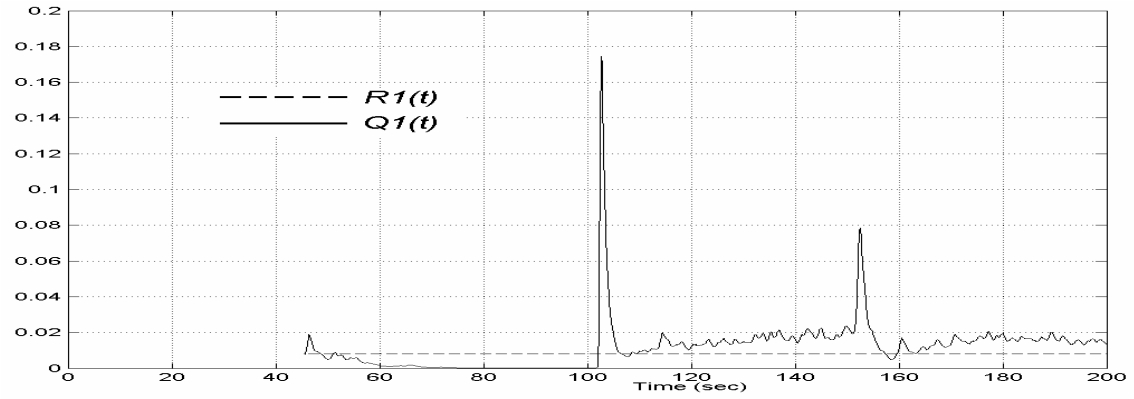


Figure 8.17 (c) Values of R_1(t) and Q_1(t) in the FL-AKF 1, fed by sensor 1, example 3.

8.4 Summary

A new methodology for designing and tuning the scaling factors of a modified hybrid PID type fuzzy logic controller (MHPID-FLC) has been presented. First, a direct relationship between the scaling factors of the MHPID-FLC and the proportional, integral and derivative actions of its traditional counterpart has been derived. Second, based on this relationship, the scaling factors are obtained using the well-known Ziegler-Nichols frequency response method. A remarkable point is that based on this relationship, the auto-tuning algorithm proposed by Astrom and Hagglund [1984] has been extended and developed for applications to the tuning of the scaling factors of the MHPID-FLC.

General guidelines for fine tuning and further improving the performance of the MHPID-FLC were given. It has been shown that this fine-tuning can be carried out in two ways: 1) modifying the scaling factors, 2) modifying the control surface of the fuzzy control system inside the MHPID-FLC structure.

The application of the proposed FL-ADKF MSDF architecture in control systems has been studied. In particular, in this chapter a novel approach to deal with the noise issue in both the auto-tuning procedure and the control performance for the MHPID-FLC, in a multi-sensor environment has been proposed. This approach combines a low-order modelling method with the FL-ADKF approach. The proposed methodology was tested in several simulated benchmark processes. Good results were obtained.

CHAPTER 9

CONCLUSIONS

9.1 Main results and conclusions of this research work

The objective of this research work was especially focused to investigate the utilisation of synergistic combinations of fuzzy logic, neuro-fuzzy and Kalman filtering techniques to design novel adaptive MSDF architectures capable of dealing with uncertain and inexact information provided by imperfect sensors. Having in mind the above objective, the main results and conclusions of this research work are summarised in this section.

The different techniques used, developments achieved, and applications studied in this research work are graphically represented in figure 9.1.

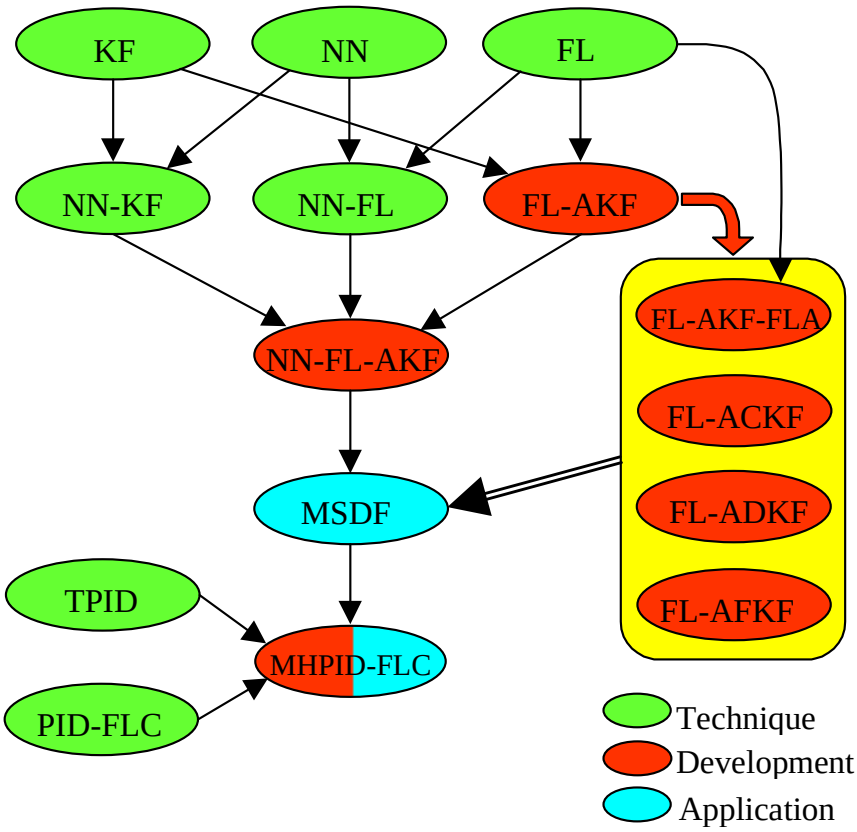


Figure 9.1 Different techniques used, developments achieved, and applications studied in this research work.

As can be seen in figure 9.1, the kernel of this research work has been the utilisation of different combinations of three main technologies: Kalman Filtering (KF), Neural Networks (NN), and Fuzzy Logic (FL). Some of these combinations already are reported in the literature and, therefore, they are indicated as existent technologies (in green). The proposed approaches, which are novel synergistic combinations of the techniques mentioned, are indicated in red.

Obviously, these developments were designed as new architectures to perform MSDF, and for this reason the MSDF approach is indicated as an application in the graphic, marked in cyan. The application of the developed MSDF approaches to MHPID-FLC, also is indicated in figure 9.1. This is a new development and a novel application of MSDF, therefore, that block is marked in both red and cyan. A short description of the main developments achieved in this research programme is given as follows.

- *The development of a novel Fuzzy Logic-based Adaptive Kalman Filter (FL-AKF) which synergistically combines Kalman filtering and fuzzy logic techniques.*

In this work, a general application adaptive Kalman filter approach was proposed. This novel development is referred to as fuzzy logic-based adaptive Kalman filter (FL-AKF). The adaptation is in the sense of dynamically adjusting the measurement noise covariance matrix R and/or the process noise covariance matrix Q from data as they are obtained. The adaptation task is carried out by a fuzzy inference system (FIS), which uses a covariance-matching technique and "common sense" rules to determine if adjustments to R and/or Q are needed.

The role of the matrices R and Q in the standard Kalman filter (SKF) setting is to adjust the Kalman gain in such a way that it controls the filter bandwidth as the state and measurement errors vary. A major drawback of the SKF formulation is that at steady state its bandwidth and Kalman gain remain constant regardless of the changes in the system dynamics or the updated measurement quality. This is due to its fixed constant matrices R and Q. Conversely, the bandwidth and Kalman gain in the FL-AKF keeps changing as long as the system dynamics and statistics of the noise under which it operates change. This dynamic adaptive property of the FL-AKF is a direct result of adapting R and/or Q.

Another main characteristic of the developed FL-AKF approach is that the filter a priori statistical information is of secondary importance because it is estimated within the algorithm itself. It must be remembered that the quality of these a priori noise statistics is of great importance in the SKF formulation.

The size of the sliding window over which the actual covariance of the residual is estimated has an impact on the adaptive filter performance. The smaller the window size, the faster the changes that can be captured by the FL-AKF. From numerous simulations it was found that a good empirical value for the size of the sliding window is between 10 to 20 samples.

The numerical complexity added to the SKF in order to build a FL-AKF is marginal. From the simulations carried out it was observed that using only three simple fuzzy sets (triangular membership functions) and only three fuzzy rules for each element in the main diagonal of Q and/or R are sufficient to ensure good adaptation.

An example showing the efficiency of the FL-AKF was presented. Superior performance was obtained with the FL-AKF over those obtained with a SKF and two different traditional adaptive Kalman filter approaches.

- *The development of a novel MSDF architecture based on FL-AKFs and a fuzzy logic performance assessment scheme (FL-AKF-FLA architecture).*

A novel hybrid MSDF architecture integrating the developed FL-AKF and a fuzzy logic performance assessment scheme was proposed (referred to as FL-AKF-FLA). This architecture merges the measurement vectors coming from N disparate sensors, each one with different measurement dynamics and noise characteristics, and obtains a fused state-vector estimate

which better reflects the actual value of the parameters being measured. This is achieved in several stages, first each measurement-vector coming from each sensor is fed to a FL-AKF. Second, a subsystem called a fuzzy logic assessor (FLA) is monitoring and assessing the performance of each FL-AKF. Thus, there are *N* sensors, *N* FL-AKFs, and *N* FLAs working in parallel. The task of each FL-AKF is to obtain a state-vector estimate based on the measurement-vector coming from its own sensor; while the task of each FLA is to assess the performance of its corresponding FL-AKF through assigning to it a degree of confidence factor, a number on the interval [0, 1]. Finally, the fused state-vector estimate is obtained using a weighting average scheme based on the assigned degree of confidence factors.

The role of the FLA in the proposed FL-AKF-FLA approach is of great importance because the fusion of the information is carried out based on the degrees of confidence generated by this component. It is noteworthy to indicate that only two variables are needed to monitor the performance of each FL-AKF and only nine 'common sense' rules are used in the FIS used in the FLA.

A simple FLA-weighting average structure is used to fuse the data in the FL-AKF-FLA architecture. Compared to the other proposed architectures, this makes this structure less demanding in computing terms to carry out the data fusion process.

- *The development of three hybrid adaptive MSDF architectures based on the proposed FL-AKF: fuzzy logic-based adaptive centralised Kalman filter (FL-ACKF), fuzzy logic-based adaptive decentralised Kalman filter (FL-ADKF), and fuzzy logic-based adaptive federated Kalman filter (FL-AFKF).*

An examination of the literature reported three MSDF architectures based on the standard Kalman filter: centralised Kalman filter (CKF), decentralised Kalman filter (DKF), and federated Kalman filter (FKF). Therefore, an obvious extension of the proposed FL-AKF approach was the development of the corresponding adaptive MSDF architectures based on it. These novel architectures are referred to as: fuzzy logic-based adaptive centralised Kalman filter (FL-ACKF), fuzzy logic-based adaptive decentralised Kalman filter (FL-ADKF), and fuzzy logic-based adaptive federated Kalman filter (FL-AFKF).

In the FL-ACKF the sensor measurements, the measurement covariance matrices and the measurement sensitivity matrices are merged to form the observation information to a central FL-AKF. Therefore, the application of the FL-AKF as the global estimator in an adaptive centralised data fusion scheme is straightforward.

The FL-ADKF processes the information in two stages. In the first stage, the local FL-AKFs process their own data in parallel to yield the best possible local estimates. In the second stage, the master filter fuses the local estimates, yielding the best global estimate. This architecture is similar to that of the standard DKF, but instead of having *N* local SKFs there are considered *N* local FL-AKFs working in parallel. In addition, instead of having local constant matrices R_{ik}, they are dynamically adjusted to fit the actual statistics of the noise profiles present in the sensors. This makes the whole FL-ADKF structure adaptive.

As the FL-ADKF, the FL-AFKF was developed by substituting the local SKFs with FL-AKFs. However, in this case, due to the use of the information sharing principle, an additional routine was added to the algorithm. The objective of the additional routine was to obtain local theoretical residual covariance matrices representing the information corresponding to local filters only. With that, the local measurement noise covariance matrices R_{ik} can be dynamically adjusted to fit the statistics of the actual measured data.

Therefore, these three approaches exploit the advantages that both Kalman filtering and fuzzy logic techniques have: the optimality of the Kalman filter and the capability of fuzzy systems to deal with imprecise information using "common sense" rules.

An illustrative example was presented to demonstrate the effectiveness and accuracy of the proposed FL-AKF-FLA, FL-ACKF, FL-ADKF, and FL-AFKF approaches. Exhaustive simulations under different measurement noise conditions and with or without the presence of faults were carried out. The results from the simulations showed that the proposed hybrid adaptive MSDF architectures are effective in situations where there are several sensors measuring the same parameters, but each one has different measurement dynamic and noise statistics. From the results of the illustrative example, it was concluded that the FL-AKF-FLA architecture is the fastest, the FL-ADKF gives the most accurate fused data, and the most fault-tolerant (for transient and persistent faults) architecture is the FL-AKF-FLA.

The selection of one of the proposed hybrid adaptive MSDF architectures for a particular application can be made taking into account their characteristics and the objectives followed in the problem at hand. For example, if it is necessary to have fast processing without the requirement of a lot of computational resources, the FL-AKF-FLA approach is adequate for this task. However, if accuracy is the main concern then the FL-ADKF can be applied. If the sensors are subjected to transient or persistent faults, then both the FL-AKF-FLA and the FL-ADKF approaches are the indicated. The FL-AFKF appears to be more suitable for fault detection purposes. The FL-ACKF could be applied in cases where there are only two or three sensors and the state vectors are of dimension two or three only. This is because of the computational resources needed to process all the information at the same time, which increases as the number of sensors grows.

- *The simplification of both the neuro-fuzzy modelling network structure and the neuro-fuzzy-standard Kalman filter (neuro-fuzzy-SKF) state estimator, proposed by Harris et al [1999, 2000, 2002].*

The first layer of the neuro-fuzzy modelling network proposed by Harris *et al* [1999, 2000, 2002] is composed of *n* FISs. All the FISs have as input the vector *x* and use the same fuzzy rule base. Consequently, all these FISs share the same vector ϕ, formed with the degree of truth values $\mu_i(x)$ of the antecedent parts of the fuzzy rules. Instead of considering *n* complete FISs, it was proposed here to build the neuro-fuzzy modelling network considering a single antecedent rule evaluator in which the calculation of the degree of truth values $\mu_i(x)$ is performed. Then, these values are distributed among *n* vector blocks θ_{a_i}, which constitute the consequent parts of the *n* rule sets, to obtain the corresponding output parameters. This simplifies the neuro-fuzzy network structure obtaining the same results, as was demonstrated in chapter 7.

The simplification of the neuro-fuzzy network does not alter its characteristic of being easy to translate to a state-space representation as is required by the SKF. In fact, exactly the same procedure used in the original network can be followed. Therefore, it is straightforward to include the simplified neuro-fuzzy network in the indirect and direct neuro-fuzzy-SKF schemes to perform state estimations. In both the indirect and direct neuro-fuzzy-SKF schemes, instead of using the original neuro-fuzzy modelling network, the simplified neuro-fuzzy modelling network is employed. Therefore a simplified neuro-fuzzy-SKF state estimator structure is obtained.

- *The development of a novel neuro-fuzzy-adaptive Kalman filter (neuro-fuzzy-AKF) state estimator which synergistically combines Kalman filtering, fuzzy logic, and neuro-fuzzy techniques.*

A novel adaptive state estimator, referred to as a neuro-fuzzy-AKF, was proposed by substituting the SKF with a FL-AKF in the simplified neuro-fuzzy-SKF state estimator structure.

The neuro-fuzzy-AKF has as its main characteristic the possibility of using the error signal in the identification process as the measurement signal for the FL-AKF in order to estimate the modelling error at the same time in which the identification process is performed. This has a stabilising effect during the training process.

In practice, system identification using the neuro-fuzzy-AKF can be implemented using two different approaches. The first is a series-parallel model, while the second is merely a parallel model. In the series-parallel model the previous process input and output are fed into the neuro-fuzzy-AKF and the error signal is used as a measurement signal for the FL-AKF. Hence, the past values of the input and output of the plant form the input vector to the neuro-fuzzy-AKF whose output $\hat{y}(t)$ corresponds to the estimate of the plant output at any instant of time *t*. This model is similar to the direct neuro-fuzzy-SKF approach and, therefore, the same learning procedure used for that case can be applied here.

In the parallel model the previous neuro-fuzzy-AKF output is fed back to the identification model and the error signal is used as measurement signal for the FL-AKF. Due to the feedback, the identification model becomes nonlinear in the parameters. This makes the gradient calculations a nonlinear optimisation problem, which requires a different learning technique. Due to this, it was proposed that a series-parallel neuro-fuzzy-AKF model be used during the process of system identification. Once the system under consideration has been identified, and assuming that the output error tends to a small value asymptotically so that $y(k) \approx \hat{y}(t)$, the series-parallel model can be replaced by a parallel neuro-fuzzy-AKF model without serious consequences [Narendra and Parthasarathy, 1990].

- *The application of the FL-AKF-FLA MSDF architecture to merge the estimates obtained from multiple neuro-fuzzy-AKFs.*

The implementation of the FL-AKF-FLA MSDF architecture using neuro-fuzzy-AKFs was proposed. This architecture is similar to that presented initially, but now the FL-AKFs are substituted by neuro-fuzzy-AKFs. Therefore, in this case the information that is being fused are the estimated nonlinear plant outputs $\hat{y}_i(t)$, performed by different neuro-fuzzy-AKFs. The fusion process is carried out through a weighted average scheme based on the confidence values calculated by the Fuzzy Logic Assessors (FLAs). The FLAs assess the performance of each neuro-fuzzy-AKF, and calculate a degree of confidence value using a fuzzy inference system (FIS). Each FIS has as inputs the absolute value of the Degree of Mismatch ($|DoM|$) and the estimated value of $R(t)$, calculated in each neuro-fuzzy-AKF (specifically, in each FL-AKF). Therefore, the application of the FL-AKF-FLA for MSDF using neuro-fuzzy-AKFs is straightforward.

Two simulated examples of neuro-fuzzy-AKF state estimation, system identification, and MSDF were presented. The identification process was carried out based on noisy signals coming from different sensors and using a series-parallel model, while the identified models were validated using a parallel model. MSDF of the estimates performed by two neuro-fuzzy-

AKFs were carried out using the FL-AKF-FLA algorithm. Good results in both system identification and MSDF were obtained.

Therefore, both modelling and estimation problems to improve the performance, reliability and accuracy of the Kalman filter approach and the MSDF architectures based on it were studied. Solutions for both problems were proposed and by simulating several examples it was demonstrated that these solutions work very well.

- *The development of a novel design and tuning procedure for PID type fuzzy logic controllers.*

A new methodology for designing and tuning the scaling factors of a modified hybrid PID type fuzzy logic controller (MHPID-FLC) was presented. This procedure was derived from the establishment of a direct relationship between the three actions of traditional PID (TPID) control and the scaling factors of the MHPID-FLC. It was proved that the MHPID-FLC works like a TPID controller with set-point weighting factor of 0.5 and modified derivative term. Based on this relationship, a set of formulae were derived to calculate the scaling factors of the MHPID-FLC employing the well-known Ziegler-Nichols frequency response method.

General guidelines for fine tuning and further improving the performance of the MHPID-FLC were given. It was shown that this fine-tuning can be carried out in two ways: 1) modifying the scaling factors, 2) modifying the control surface of the fuzzy control system inside the MHPID-FLC structure.

- *The development of an auto-tuning procedure for PID type fuzzy logic controllers.*

Based on the relationship established between TPID and the MHPID-FLC, the systematic design and tuning methods of TPID control can be extended and developed for applications in designing and tuning of the MHPID-FLC. In particular, the relay auto-tuning algorithm proposed by Astrom and Hagglund [1984] was extended and developed for applications to the auto-tuning of the scaling factors of the MHPID-FLC.

The proposed methodology was tested in several simulated benchmark processes. In all cases the MHPID-FLC performance is equivalent to its traditional counterpart. Thus, the set-point weighting factor is embedded in the MHPID-FLC structure; it is not necessary to specify it as another variable. However, in this case it is a fixed value (0.5).

- *The application of the developed FL-ADKF architecture in the auto-tuning of PID type fuzzy logic controllers using multiple noisy sensors.*

Although the developed MSDF architectures can be applied to a broad range of problems, in this work the application in the PID type fuzzy logic control approach was explored. In particular, the FL-ADKF was employed in a novel structure to design and auto-tune the MHPID-FLC embedded in a multiple sensory environment. The proposed approach combines a low-order modelling method with the FL-ADKF MSDF architecture. This approach effectively deals, as was demonstrated by simulating several examples, with the noise issue in both the auto-tuning procedure and the control performance for the MHPID-FLC.

9.2 Future work

The choice of the form and parameters that define the fuzzy sets used in the FISs inside the FL-AKF structure to adjust *R* and/or *Q* was made out using a trial and error scheme. Obviously, this process is time consuming and depends on the problem under consideration. In order to save time in an actual application, some guidelines to determine the parameters that define the fuzzy sets were given based on the experience gained through simulating many examples. However, it may be possible that for a particular application these guidelines do not work and some time must be spent in experimentation and simulation to find the correct parameters. This can be a drawback of the adjustment algorithm, and so a solution should be found. The author suggests the idea of exploring the utilisation of a neuro-fuzzy system or a genetic algorithm to automatically adjust the fuzzy sets to the requirements of the problem at hand.

From the illustrated example of the FL-AKF, it was demonstrated that the adaptation procedure is stable when *R*-only or *Q*-only are adjusted. However, this characteristic is not very clear when both *R* and *Q* are adjusted simultaneously. A deeper analysis of this case is needed to determine in what circumstances this adaptation procedure is stable. In addition, an adaptive procedure, that was not explored here, is the adaptation of *R* and *Q* in an alternating manner. That is, adapt one of these matrices for a certain period of time, and then adapt the other matrix for another certain period of time, and so on. It would be interesting to observe the performance of the FL-AKF using this procedure and determine when it can be applied.

The four developed hybrid Kalman filter-fuzzy logic adaptive MSDF architectures (FL-AKF-FLA, FL-ACKF, FL-ADKF, and FL-AFKF) demonstrated good intrinsic fault-tolerant characteristics against transient and persistent faults. This was not the case against permanent faults. However, permanent faults are easy to detect by analysing the adjusted measurement noise covariance matrices or by analysing the residual sequences. Therefore, it will not be too difficult to develop a fault detection and recovery algorithm, for example applying a voting technique or a residual-based scheme, to overcome the existence of this kind of fault.

In the two simulated examples presented of neuro-fuzzy-AKF state estimation and system identification a chirp signal (sine wave whose frequency varies linearly with time) was used to obtain the data to train the network. By using this kind of signal as a training signal and the error signal as a measurement noise signal for the FL-AKF inside the neuro-fuzzy-AKF structure, the training of the neuro-fuzzy-AKF using the series-parallel identification model is stable as was proved practically. However, further investigation is needed to determine if this is true for a broader class of systems or to define under what conditions and for what kind of systems this is the case. This task is left as future work to follow on from this research.

System identification using the neuro-fuzzy-AKF was performed in a series-parallel model configuration. In practice, a parallel model configuration also can be applied. However, due to the feedback, the identification model becomes nonlinear in the parameters. This makes the gradient calculations a nonlinear optimisation problem, which requires a different learning technique. Therefore, the determination of the kind of learning algorithms (e. g. dynamic back propagation [Narendra and Parthasarathy, 1990]) that can be applied in this case should be studied.

The neuro-fuzzy-AKF state estimator was proposed for single-input-single-output (SISO) non-linear systems. The case for multiple-input-multiple-output (MIMO) non-linear systems was not considered here and is left as a future work.

The proposed neuro-fuzzy-AKF state estimator suffers the problem of *the curse of dimensionality* [Brown and Harris, 1994] associated with medium or large input space modelling tasks. This means that the number of rules in the neuro-fuzzy modelling network (and

the associated data required for training) is an exponential function of the input space dimension. This poses a practical limitation to systems with a small input space dimension (e. g. <6). In the literature several constructional algorithms have been proposed as a solution to the above problem. Two examples are the adaptive spline modelling algorithm (ASMOD) [Harris *et al*, 1997] and the neuro-fuzzy system design and construction algorithm (NeuDec) [Hong and Harris, 2001]. The possible application of these algorithms in the neuro-fuzzy-AKF for large input space modelling is an interesting point worth investigating.

The proposed design and auto-tuning procedure for the MHPID-FLC was tested on several simulated benchmark processes. General guidelines for fine tuning and further improving the performance of the MHPID-FLC by modifying the scaling factors were given. Alternatively, fine-tuning can be carried out by modifying the control surface (by modifying the consequent parameters of the rules) of the FCS inside the MHPID-FLC structure. This was practically exemplified for the multiple sensor case. However, more research on the effects that the modification of the control surface has on the performance of the MHPID-FLC is needed. General guidelines for this type of fine-tuning procedure need to be determined and investigated. This opens another interesting avenue of investigation.

The application of the proposed FL-ADKF MSDF architecture in control systems under a multi-sensor scheme was proposed. In this case only linear systems were considered. Recently, in the literature has been reported the development of the so-called multiple model adaptive control approach [Schott and Bequette, 1997]. An extension of that approach including multiple sensors could be referred to as multiple-sensors multiple-model adaptive control. This approach may be developed for both linear and non-linear systems using the FL-AKF and the neuro-fuzzy-AKF approaches.

Finally, in the development of the MSDF approaches presented here, it has been assumed that the data being reported by the sensors is “true” information. However, in some applications (e. g. defence) artificial information may be produced to intentionally mislead the sensors. Therefore, the data produced by the sensors would be “false” information, although no indication of it is actually present in the data itself. In these cases, additional features would need be added in order to produce a MSDF system capable of effectively discriminating between true and false information before the fusion process is carried out. This area has not been investigated in this work but offers an intriguing avenue of future research.

REFERENCES

Abdelnour, G. M., Chang, C. H., Huang, F. H., and Cheung, J. Y. (1991). Design of a fuzzy controller using input and output mapping factors, *IEEE Trans. Syst., Man, Cybern.*, 21, 952-960.

Abdelnour, G., Chand, S. and Chiu, S. (1993). Applying fuzzy logic to the Kalman filter divergence problem, *Proceedings of the International Conference on Systems, Man and Cybernetics Systems: Engineering in the Service of Humans*, 1, pp. 630-635.

Abidi, M. A. and Gonzalez, R. C. (1992). *Data fusion in robotics and machine intelligence*, Academic Press, London.

Alspach, D. L. (1972). Comments "On the identification of variances and adaptive Kalman filtering", *IEEE Transactions on Automatic Control*, AC-17, pp. 843-845.

Astrom, K. J. and Hagglund, T. (1984). Automatic tuning of simple regulators with specifications on phase and amplitude margins, *Automatica*, 20, 645-651.

Astrom, K. J. and Hagglund, T. (1995). *PID Controllers: Theory, Design, and Tuning*, 2nd edition, Instrument Society of America, USA.

Astrom, K. J., Hagglund, T., Hang, C. C., and Ho, W. K. (1993). Automatic tuning and adaptation for PID controllers – A survey, *Control Eng. Practice*, 1 (4), pp. 699-714.

Bar-Shalom, Y and Li, X. R. (1993). *Estimation and Tracking: Principles, Techniques, and Software*, Artech House: Norwood, MA, USA.

Basir, O. A. and Shen, H. C. (1999). Interdependence and information loss in multi-sensor systems, *Journal of Robotic Systems*, 16 (11), pp. 597-612.

Billings, S. A. (1980). *Introduction to Kalman filters*, University of Sheffield, Dep. of Automatic Control Engineering. R. R. No. 127.

Billings, S. A. and Zhu, Q. M. (1994). Nonlinear model validation using correlation tests, *International Journal of Control*, 60, pp. 1107-1120.

Blum E. K. and Li, L. K. (1991). Approximation theory and feed forward networks, *Neural Networks*, 4, pp. 511-515.

Bogler, Philip L. (1987). Shafer–Dempster reasoning with applications to multisensor target identification systems, *IEEE Transactions on Syst, Man and Cyber.*, SMC-17 (6), pp. 968-977.

Bonissone, P., Badami, V., Chiang, K., Khedkar, P., Marcelle, K., and Schutten, M. (1995). Industrial applications of fuzzy logic at General Electric, *Proc. IEEE*, 83 (3), pp. 450-465.

Brown, M. and Harris, C. (1994). *Neurofuzzy Adaptive Modelling and Control*, Prentice Hall International (UK) Limited.

Brown, M., Mills, D. J. and Harris, C. (1996). The representation of fuzzy algorithms used in adaptive modelling control schemes, *Fuzzy Sets and Systems*, 79, pp. 69-91.

Brown, R. G. and Hwang, P. Y. C. (1997). *Introduction to random signals and applied Kalman filtering whit MATLAB exercises and solutions*, Third edition, John Wiley & Sons, New York.

Brumback, B. D. and Srinath, M. D. (1987). A chi-square test for fault-detection in Kalman filters, *IEEE Transactions on Automatic Control*, A-32 (6), pp. 552-554.

Caputi, M. J. (1995). A necessary condition for effective performance of the multiple model adaptive estimator, *IEEE Trans. Aerospace and Electronic Systems*, 31 (3), pp. 1132-1139.

Carlson, N. A. (1990). Federated square root filter for decentralized parallel processes, *IEEE Transactions on Aerospace and Electronic Systems*, 26 (3), pp. 517-525.

Castro, J. L. (1995). Fuzzy Logic Controllers are universal approximators, *IEEE Trans. Sys. man and Cyb.*, 25 (4), pp. 629-635.

Chaer, W. S., Bishop, R. H., and Ghosh, J. (1997). A mixture-of-experts framework for adaptive Kalman filtering, *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 27 (3), pp. 452-464.

Chen, F. C., and Khalil, H. K. (1995). Adaptive control of a class of nonlinear discrete-time systems using neural networks, *IEEE Transactions on Automatic Control*, 40 (5), pp. 791-801.

Chen, G. and Chui, C. K. (1991). A modified adaptive Kalman filter for real-time applications, *IEEE Trans. Aerospace and Electronic Systems*, 27 (1), pp. 149-154.

Corcoran, P. and Lowery, P. (1995). Neural network applications in multisensor systems, *Sensor Review*, 15 (4), pp. 15-18.

Dailey, D. J., Harn, P. and Li, P. –J. (1996). *ITS data fusion*, Research Report, Project T9903, Washington State Transportation Center, University of Washington, Seattle, Washington, USA.

Dall, L. (1998). *Optimal Fusion of Sensors*, PhD Dissertation, Department of Automation, Technical University of Denmark.

Driankov, D., Hellendorm, H. and Reinfrank, M. (1996). *An Introduction to Fuzzy Control*, Springer-Verlag, New York.

Durrant-Whyte, H. F. (1987). Consistent integration and propagation of disparate sensor observations, *International Journal of Robotics Research*, 6 (3), pp. 3-24.

Durrant-Whyte, H. F. (1988). Sensor models and multisensor integration. *International Journal of Robotics Research*, 7 (6), pp. 97-113.

Durrant-Whyte, H. F. (1991). Elements of sensor fusion, *IEE Colloquium on Intelligent Control*, pp. 5/1-2.

Escamilla-Ambrosio, P. J. (1999). Exposition and test of a new method of defuzzification for fuzzy control systems, *Proceedings of the IASTED International Conference on Control and Applications*, July 24-26, Banff, Alberta, Canada, pp. 503-508.

Escamilla-Ambrosio, P. J. and Mort, N. (2000). Adaptive Kalman filtering through fuzzy logic, *Proceedings of the 7th UK Workshop On Fuzzy Systems, Recent Advances and Practical Applications of Fuzzy, Neuro-Fuzzy, and Genetic Algorithm-Based Fuzzy Systems*, October 26-27, Sheffield, U.K., pp. 67-73.

Escamilla-Ambrosio, P. J. and Mort, N. (2001a). A hybrid Kalman filter-fuzzy logic multisensor data fusion architecture with fault tolerant characteristics, *Proceedings of the 2001 International Conference on Artificial Intelligence (IC-AI'2001)*, June 25-28, Las Vegas, Nevada, USA, pp. 361-367.

Escamilla-Ambrosio, P. J. and Mort, N. (2001b). A hybrid Kalman filter-fuzzy logic architecture for multisensor data fusion, *Proceedings of the 2001 IEEE Joint International Conference on Control Applications & International Symposium on Intelligent Control*, September 5-7, Mexico City, Mexico, pp. 364-369.

Escamilla-Ambrosio, P. J. and Mort, N. (2001c). Development of a fuzzy logic-based adaptive Kalman filter, *Proceedings of the European Control Conference ECC'01*, September 4-7, Porto, Portugal, pp. 1768-1773.

Escamilla-Ambrosio, P. J. and Mort, N. (2002a). A novel design and tuning procedure for PID type fuzzy logic controllers, *Proceedings of the IS'2002 First International IEEE Symposium 'Intelligent Systems'*, September 10-12, Varna, Bulgaria, pp. 36-41.

Escamilla-Ambrosio, P. J. and Mort, N. (2002b). Auto-tuning of fuzzy PID controllers, *In CD ROM Proceedings of the XV IFAC World Congress*, July 21-26, Barcelona, Spain.

Escamilla-Ambrosio, P. J. and Mort, N. (2002c). Multisensor data fusion architecture based on adaptive Kalman filters and fuzzy logic performance assessment, *Proceedings of the Fifth International Conference on Information Fusion*, July 8-11, Loews Annapolis Hotel, Annapolis, MD, U.S.A. pp.1542-1549.

Fitzgerald, R. J. (1971). Divergence of the Kalman filter, *IEEE Transactions on Automatic Control*, AC-16 (6), pp. 736- 747.

Francis, W. G. (1996). *SCAAT: Incremental Tracking with Incomplete Information*, PhD Dissertation, Department of Computer Science, The University of North Carolina at Chapel Hill.

Gao, Y., Krakiwsky, E. J., Abousalem, M. A., and McLellan, J. F. (1993). Comparison and analysis of centralized, decentralized, and federated filters, *Navigation: Journal of The Institute of Navigation*, 40 (1), pp. 69-86.

Gravel, A. and Mackenberg, H. (1995). Mathematical analysis of the Sugeno controller leading to general design rules, *Fuzzy Sets and Systems*, 85, pp. 165-175.

Grewal, M. S. and Andrews, A. P. (1993). *Kalman filtering theory and practice*, Prentice Hall, Englewood Cliffs, NJ.

Gupta, M. M. (1994). Fuzzy neural networks: Theory and applications, *Proceedings SPIE*, 2353, The International Society for Optical Engineering, pp. 303-325.

Gupta, M. M. and Rao, D. H. (1994). Neuro-Control Systems: A Tutorial, *Neuro Control Systems Theory and Applications*, Madan M, G. and Dandina H. R. Eds., IEEE Press, pp. 1-43.

Hall, D. L. (2002). *A taste of multi-sensor data fusion*, Tutorial TA1 notes, Fusion 2002 Conference, Aannapolis, Maryland, USA.

Hall, D. L. and Llinas, J. (1997). An introduction to multisensor data fusion, *Proceedings of the IEEE*, 19 (1), pp. 6-23.

Hang, C. C., Astrom, K. J. and Ho, W. K. (1991). Refinements of the Ziegler-Nichols tuning formula, *IEE Proceedings-D*, 138, pp. 111-118.

Hang, C. C., Lee, T. H., and Ho, W. K. (1993). *Adaptive Control*, Instrument Society of America.

Harris, C. J., Brown, M., Bossley, K. M., Mills, D. J., and Ming, F. (1996). Advances in neurofuzzy algorithms for real-time modelling and control, *Engng Applic. Artif. Intell*., 9 (1), pp. 1-6.

Harris, C. J., Hong, X. and Gan, Q. (2000). Neurofuzzy state estimators, in *Soft computing and intelligent systems: Theory and applications*, Edited by Naresh K. Sinha and Madam M. Gupta, Academic Press, London, UK, pp. 377-402.

Harris, C. J., Hong, X. and Gan, Q. (2002). *Adaptive modelling, estimation and fusion from data: A neurofuzzy approach*, Springer-Verlag, Berlin Heildelberg, Germany.

Harris, C. J., Wu, Z. Q. and Gan, Q. (1999). Neurofuzzy state estimators and their applications, *Annual Reviews in Control*, 23, pp. 149-158.

Harris, C. J., Wu, Z. Q., Bossley, K. and Brown, M. (1997). Intelligent neurofuzzy estimators and multisensor data fusion, in S. G. Tzafestas (ed.), *Methods and applications of intelligent control*, pp. 283-303.

Hashemipour, H. R., Roy, S. and Laub, A. J. (1988). Decentralized structures for parallel Kalman filtering, *IEEE Transactions on Automatic Control*, 33 (1), pp. 88-94.

Haykin, S. (1989). *An introduction to analog & digital communications*, John Wiley & Sons, New York, USA.

Haykin, S. (1994). *Neural Networks A Comprehensive Foundation*, Macmillan College Publishing Company, Inc., NY.

Haykin, S. (1999). *Neural networks: A comprehensive foundation*, Second Edition, Prentice Hall, Upper Saddle River, New Jersey, USA.

Henkind, S. J. and Harrison, M. C. (1988). An analysis of four uncertainty calculi, *IEEE Transactions on Systems, Man and Cybernetics*, 18 (5), pp. 700-714.

Hirota, K., Pedricz, W., and Yuda M. (1992). Fuzzy set-based models of sensor fusion, in *Fuzzy engineering toward human friendly systems*, T. Terano, M. Sugeno, M. Mukaidono, K. Shigemasu Editors, IOS press, IFES '91, Japan, pp. 623-633.

Hong, L. (1991). Adaptive data fusion, *Proceedings of the IEEE International Conference on Systems, man ad Cybernetics*, 2, pp. 767-772.

Hong, X. and Harris, C. J. (2001). Variable selection algorithm for the construction of MIMO operating point dependent neurofuzzy networks, *IEEE Transactions on Fuzzy Systems*, 9 (1), pp. 88-101.

Hong X. and Harris, C. J. (2001). Neurofuzzy design and model construction of nonlinear dynamical processes from data, *IEE Proc. Control Theory Appl.*, Vol. 148, No. 6, pp. 530-538.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators, *Neural Networks*, 2, pp. 359–366.

Ishibuchi, H., Morioka K., and Turksen, I. B. (1995). Learning by fuzzified neural Networks, *Int. J. on Approximate Reasoning*, 13 (4), pp. 327-358.

Jang, J. -S. R. (1993). ANFIS: Adaptive-Network-Based Fuzzy Inference System, *IEEE Transactions on Syst. Man, and* Cyber., 23 (3), pp. 665-685.

Jang, J. -S. R. and Sun, C. -T. (1993). Functional equivalence between radial basis function networks and fuzzy inference systems, *IEEE Transactions on Neural Networks*, 4 (1), pp. 156-159.

Jang, J. -S. R. and Sun, C. -T. (1995). Neuro-fuzzy Modeling and Control, *Proceedings of the IEEE*, 83 (3), pp. 378-406.

Jang, J. -S. R., Sun, C. -T., and Mizutani, E. (1997). *Neuro-fuzzy and soft computing: A computational approach to learning and machine intelligence*, Prentice-Hall, NJ, USA.

Jantzen, J. (1997). A robustness study of fuzzy control rules, In: EUFIT (ed.), *Proc. Fifth European Congress on Fuzzy and Intelligent Technologies*, ELITE Foundation, Promenade 9, D-52076 Aachen, pp. 1222-1227.

Jantzen, J. (1999). *Tuning of fuzzy PID controllers*, Online 98-H-871 (fpid), Technical University of Denmark: Dept. of Automation, http://www.iau.dtu.dk/#jj/pubs.

Jazwinski, A. H. (1969). Adaptive Filtering, *Automatica*, 5, pp. 475-485.

Jazwinski, A. H. (1970). *Stochastic processes and filtering theory*, Academic Press, USA.

Jetto, L., Longhi, S., and Vitali, D. (1999). Localization of a wheeled mobile robot by sensor data fusion based on a fuzzy logic adapted Kalman filter, *Control Engineering Practice*, 7, pp. 763-771.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems, *Transactions of the ASME-Journal of Basic Engineering*, 82 (Series D), pp. 35-45.

Kiang, M. Y. (2001). Extending the Kohonen self-organizing map networks for clustering analysis, *Computational Statistics &Data Analysis*, 38, pp. 161-180.

King, P. J. and Mamdani, E. H. (1997). The application of fuzzy control systems to industrial processes, *Automatica*, 13, pp. 235-242.

Klein, L. A. (1999). *Sensor and data fusion concepts and applications*, Second Edition, SPIE Optical Engineering Press, USA.

Kobayashi F., Arai, F., Fukuda, T., Shimojima, K., Onoda, M., Marui, N. (1998). Sensor fusion system using recurrent fuzzy inference, *Journal of Intelligent and Robotic Systems*, 23, pp. 201-216.

Kobayashi, K., Cheok, C. K. and Watanabe, K. (1995). Estimation of absolute vehicle speed using fuzzy logic rule-based Kalman filter, *Proceedings of the American Control Conference*, pp. 3086-3090.

Kobayashi, K., Cheok, C. K., Watanabe, K. and Munekata, F. (1998b). Accurate differential global positioning system via fuzzy logic Kalman filter sensor fusion technique, *IEEE Transactions on Industrial Electronics*, 45 (3), pp. 510-518.

Kohonen, T. (1990). The self-organizing map, *Proceedings of the IEEE*, 78 (9), pp. 1464-1480.

Kosko, B. (1992). *Neural Networks and Fuzzy Systems: A dinamical Approach to Machine Intelligence*. Prentice Hall, Englewood Cliffs, NJ, USA.

Kosko, B. (1994). Fuzzy systems as universal approximators, *IEEE Trans. Computers*, 43 (11), pp. 1329-1333.

Kuo, R. J. and Cohen, P. H. (1999). Multi-sensor integration for on-line tool wear estimation through radial basis function networks and fuzzy neural networks, *Neural Networks*, 12, pp. 355-370.

Lalk, J. (1994). Intelligent adaptation of Kalman filters using fuzzy logic, *Proceedings of the 3rd IEEE Conference on Fuzzy Systems*, 2, pp. 744-749.

Larsen, D. T. (1998). *Optimal Fusion of Sensors,* PhD Dissertation, Department of Automation, Technical University of Denmark.

Lee, C. C. (1990). Fuzzy logic in control systems: Fuzzy logic Controllers, Part I and II, *IEEE Transactions on Syst, Man and Cyber*, 20 (2), pp. 404-435.

Lee, J. (1993). On methods for improving performance of PI-type fuzzy logic controllers. *IEEE Trans. Fuzzy Syst.*, 1, pp. 298-301.

Li, H. -X. (1997). A comparative design and tuning for conventional fuzzy control, *IEEE Trans. Syst., Man, Cybern., Part B*, 27, pp. 884-889.

Li, H. -X. and Gatland, H. B. (1996). Conventional fuzzy control and its enhancement, *IEEE Trans. Syst., Man, Cybern., Part B*, 26, pp. 791-797.

Li, H. -X. and Tso, S. K. (2000). Quantitative design and analysis of fuzzy proportional-integral-derivative control—a step towards autotuning, *Int. J. Syst. Sci.*, 31, pp. 545-553.

Liu, G. P., Kadirkamanathan, V., and Billings, S. A. (1999). Neural network-based variable structure control for nonlinear discrete systems, *International Journal of Systems Science*, 30 (10), pp. 1153-1160.

Ljung, L. (1987). *Systems identification–Theory for the user*, Prentice Hall, Englewood Cliffs, New Jersey.

Luo, R. C. and Kay, M. G. (1989). Multisensor integration and fusion in intelligent systems, *IEEE Transactions on Syst., Man and Cybern.*, 19 (5), pp. 901-931.

Luo, R. C. and Kay, M. G. (1992). Data fusion and sensor integration: state-of-the-art 1990s, in *Data fusion in robotics and machine intelligence*, Edited by Abidi, M. A. and Gonzalez, R. C. Academic Press, London.

Luo, R. C., Yih, C.-C., and Su, K. L. (2002). Multisensor fusion and integration: Approaches, applications, and future research directions, *IEEE Sensors Journal*, 2 (2), pp. 107-119.

Magill, D. T. (1965). Optimal adaptive estimation of sampled stochastic processes, *IEEE Trans. Automatic Control*, AC-10, pp. 434-439.

Mamdani, E. and Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller, *International Journal of Man-Machine Studies*, 7 (1), pp. 1-13.

Mann, G. K. I., Hu, B. G. and Gosine, R. G. (1999). Analysis of direct action fuzzy PID controller structures, *IEEE Trans. Syst., Man, Cybern., Part B*, 29, pp. 371-388.

Mann, G. K. I., Hu, B. G. and Gosine, R. G. (2001). Two-level tuning of fuzzy PID controllers, *IEEE Trans. Syst., Man, Cybern., Part B*, 31, pp. 263-269.

Maybeck, P. S. (1979). *Stochastic models estimation and control*, Volume 1, Academic Press, New York.

Maybeck, P. S. (1989). Moving-bank multiple model adaptive estimation and control algorithm: an evaluation, *Control and dynamic systems*, 31, pp. 1-31.

Mehra, R. K. (1970). On the identification of variances and adaptive Kalman filtering, *IEEE Trans. Automatic Control*, AC-15 (2), pp. 175-184.

Mehra, R. K. (1972). Approaches to adaptive filtering, *IEEE Trans. Automatic Control*, AC-17, pp. 693- 698.

Mirabadi, A. (1999). *Fault tolerant train navigation systems using a multisensor integration approach*, PhD Dissertation, Department of Automatic Control and Systems Engineering, University of Sheffield.

Mirabadi, A., Mort, N., and Schmid, F. (1996). Application of sensor fusion to railway systems, *Proceedings of the 1996 IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 185-192.

Moghaddamjoo, A. and Kirlin, L. R. (1989). Robust adaptive Kalman filtering with unknown inputs, *IEEE Trans. Acoustics, speech, and signal processing*, 37 (8), pp. 1166-1175.

Mohamed, A. H. (1999). *Optimizing the estimation procedure in INS/GPS integration for kinematic applications*, PhD dissertation, Department of Geomatics Engineering, University of Calgary, Alberta, Canada.

Mohamed, A. H. and Shwarz, K. P. (1999). Adaptive kalman filtering for INS/GPS, *Journal of Geodesy*, 73 (4), pp. 193-203.

Murphy, P. (1991). Fuzzy logic and its application in control systems, *ISA*, Paper No. 91-0474, pp. 1231-1245.

Narendra, K. S., and Mukhopadhyay, S. (1997). Adaptive control using neural networks and approximated models, *IEEE Transactions on Neural networks*, 8 (3), pp. 475-485.

Narendra, K. S., and Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks, *IEEE Transactions on Neural networks*, 1 (1), pp. 4-27.

Nauck, D., Klawonn, F., Kruse, R. (1997). *Foundations of Neuro-Fuzzy Systems*, John Wiley & Sons, Inc., UK.

Nelles, O. and Isermann, R. (1996). Basis function networks for interpolation of local linear models, *Proceedings of the 35th Conference on Decision and Control*, Kobe, Japan, pp. 470-475.

Paik, B. S. and Oh, J. H. (2000). Gain fusion algorithm for decentralised parallel Kalman filters, *IEE Proc. Control Theory Appl.*, 147 (1), pp. 97-103.

Prajitno, P. (2002). *Neuro-fuzzy methods in multisensor data fusion*, PhD Dissertation, Department of Automatic Control and Systems Engineering, University of Sheffield.

Rumelhart, David E. Hinton, G. E & Williams, Ronald J. Learning representations by back-propagating errors, *Nature*, 323, pp. 533-536.

Sangsuk-Iam, S., and Bullock, T. E. (1990). Analysis of discrete-time Kalaman filtering under incorrect noise covariances, *IEEE Trans. Automatic Control*, 35 (12), pp. 1304-1308.

Sasiadek, J. Z. and Wang, Q. (1999). Fuzzy adaptive Kalman filtering for INS/GPS data fusion, *AIAA Guidance, Navigation and Control Conference*, pp. 1911-1918.

Schott, K. D., and Bequette, B. W. (1997). Multiple model adaptive control, in *Multiple model approaches to modelling and control*, Edited by R. Murray-Smith and T. A. Johansen, Taylor & Francis Ltd., U. K.

Seborg, D. E., Edgar, T. F., and Mellichamp, D. A. (1989). *Process dynamics and control*, Wiley, New York.

Stansfield, E. V. (2001). Introduction to Kalman filters, Paper presented at Kalman Filter Day, Kalman filters – Applications and Pitfalls, Thales Research Ltd, Heckfield Place, UK. Accessed via the IEE Signal Processing Professional Network.

Sugeno, M. and Kang, G. T. (1988). Structure identification of fuzzy model, *Fuzzy Sets and Systems*, 28, pp. 15-33.

Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its applications to modelling and control, *IEEE Transactions on Syst. Man and Cyber.*, 15, pp. 116-132.

Tang, K. L. and Mulholland, R.J. (1987). Comparing fuzzy logic with classical controller designs, *IEEE Trans. Syst., Man, Cybern.*, SMC-17, pp. 1085-1087.

Tsoukalas, L. H. and Uhrig, R. E. (1997). *Fuzzy and Neural Approaches in Engineering*, John Wiley & Sons, Inc., NY, USA.

Tsukamoto, Y. (1979). An approach to fuzzy reasoning method, in *Advances in fuzzy set theory and applications*, M. M.Gupta, R. K. Ragade, and R. R. Yager, Eds. Amsterdam: North-Holland, pp. 137-149.

Varshney, P. K. (1997). Multisensor data fusion, *Electronics and Communication Engineering Journal*, 9 (6), pp. 245-253.

Waltz, E. L. and Buede, D. M. (1986). Data fusion and decision support for command and control, *IEEE Transactions on Syst., Man and Cybern.*, SMC-16 (6), pp. 865-879.

Wang, H. and Goh, C. T. (1999). Fuzzy logic Kalman filter estimation for 2-wheel steerable vehicles, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Human Environment Friendly Robots with High Intelligence and Emotional Quotients*, 1, pp. 88-93.

Wang, H., Brown, M., and Harris, C. J. (1996a). Modelling and control of nonlinear, operating point dependent systems via associative memory networks, *International Journal of Dynamics and Control*, 6, pp. 199-218.

Wang, H., Wang, A. P., Brown, M. and Harris, C. J. (1996b). One-to-one mapping and its application to neural networks based control systems design, *International Journal of Systems Science*, 27 (2), pp. 161-170.

Wang, L. X. (1992). Fuzzy systems are universal approximators, *Proc. IEEE Int. Conf. on Fuzzy Syst.*, San diego, CA.

Wang, Q. –G., Hang, C. –C., and Zou, B. (1997). Low-order modeling from relay feedback, *Ind. Eng. Chem. Res.*, 36, pp. 375-381.

Wei, M. and Schwarz, K. P. (1990). Testing a decentralized filter for GPS/INS integration, *Proceedings of the IEEE Position Location and Navigation Symposium*, pp. 429-435.

Welch, G. and Bishop, G. (1995). *An Introduction to the Kalman Filter*, The University of North Carolina at Chapel Hill, TR95-045.

Widrow, B. and Stearns, S. D. (1985). *Adaptive signal processing*, Prentice-Hall, Englewood Cliffs, NJ, USA.

Willsky, A. S. (1976). A survey of design methods for failure detection in dynamic systems, *Automatica*, 12, pp. 601-611.

Woo, Z. -W., Chung, H. -Y. and Lin, J. -J. (2000). A PID type fuzzy controller with self-tuning scaling factors, *Fuzzy Sets and Systems*, 115, pp. 321-326.

Wu, Z. Q. and Harris, C. J. (1996). An adaptive neurofuzzy Kalaman filter, *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems*, 2, pp. 1344-1350.

Wu, Z. Q. and Harris, C. J. (1997). A neurofuzzy network structure for modelling and state estimation of unknown nonlinear systems, *International Journal of Systems Science*, 28 (4), pp. 335-345.

Xu, J. X., Hang, C. C. and Liu, C. (2000). Parallel structure and tuning of a fuzzy PID controller, *Automatica*, 36, pp. 673-684.

Zadeh L. A. (1965). Fuzzy Sets, *Information and Control*, 8, pp. 338-353.

Zadeh L. A. (1973). Outline of a New Approach to the analysis of Complex Systems and Decision Processes, *IEEE Transactions on Syst, Man and Cybern.*, SMC-3 (1), pp. 28-44.

Zadeh L. A. (1977). *Theory of fuzzy sets*, Memorandum No. UCB/ERL M77-1, Electronics Research Laboratory, College of Engineering, University of California, Berkeley.

Zadeh, L. A. (1988). Fuzzy Logic, *Computer*, 21 (4), pp. 83-93.

Zadeh, L. A. (1994). Fuzzy Logic, Neural Networks, and Soft Computing, *Communic. ACM*, 37 (3), pp. 77-84.

Zhang, H., Lennox, P., Goulding, Y., and Wang, Y. (2002). Adaptive information sharing factors in federated Kalman filtering, *Proceedings of the 15th IFAC World Congress (in CD)*, Barcelona, Spain.

Zhang, J. and Knoll, A. (1998). Constructing fuzzy controllers with B-spline models – principles and applications, *International Journal of Intelligent Systems*, 13 (2/3), pp. 257-286.

Zhao, Z. -Y., Tomizuka, M. and Isaka, S. (1993). Fuzzy gain scheduling of PID Controllers, *IEEE Trans. Syst., Man, Cybern.*, 23, pp. 1392-1398.

Zhu, Y. (1999). Efficient recursive state estimator for dynamic systems without knowledge of noise covariances, *IEEE Trans. Aerospace and Electronic Systems*, 35 (1), pp. 102-113.

Zhuang, M. and Atherton, D. P. (1993). Autotuning of optimum PID controllers, *IEE Proceedings-D*, 140, pp. 216-224.

APPENDIX A

THEORY OF FUZZY SETS: NOTATION, TERMINOLOGY AND BASIC OPERATIONS

A.1 Fuzzy sets and terminology

Let U be a collection of objects, concepts or mathematical constructions denoted generically by $\{u\}$. U is called the universe of discourse and u represents the generic element of U [Zadeh, 1977]. For example, U may be the set of all real numbers; the set of integers 0, 1, 2,…, 100; the set of all residents in a city; the set of all students in a course; the set of objects in a room; the set of all names in a telephone directory, etc. Universes of discourse are usually denoted by the symbols $U, V, W,$…, with or without subscripts and/or superscripts.

Definition A.1 Fuzzy set and membership function
A fuzzy subset A of a universe of discourse U is characterised by a membership function $\mu_A : U \rightarrow [0,1]$ which associates with each element u of U a number $\mu_A(u)$ in the interval [0,1], with $\mu_A(u)$ representing the grade of membership of u in A [Zadeh, 1965, 19977]. A fuzzy set in U or, equivalently, a fuzzy subset of U, is usually denoted by one of the uppercase symbols A, B, C, D, E, F, G, H, with or without subscripts and/or superscripts.

A fuzzy set can be denoted as follows:

$$A = \begin{cases} \sum_{u_i \in U} \mu_A(u_i)/u_i, & \text{if } U \text{ is a collection of discrete objects.} \\ \int_U \mu_A(u)/u, & \text{if } U \text{ is a continuous space (usually the real line } \Re). \end{cases} \quad \text{(A.1).}$$

The summation and integration signs en (A.1) stand for the union of $(u, \mu_A(u))$ pairs; they do not indicate summation or integration. Similarly, the symbol "/" is only a marker and does not imply arithmetic division.

Definition A.2 Support
The *support* of A is the set of points in U at which $\mu_A(u)$ is positive,

$$\text{support}(A) = \{u \mid \mu_A(u) > 0\} \quad \text{(A.2).}$$

Definition A.3 Height
The *height* of A is the supremum of $\mu_A(u)$ over A,

$$\text{hgt}(A) = \sup_{u \in U} \{\mu_A(u)\} \quad \text{(A.3).}$$

Definition A.4 Crossover point
A *crossover point* of A is a point in U whose grade of membership in A is 0.5,

$$\text{crossover}(A) = \{u \mid \mu_A(u) = 0.5\} \quad \text{(A.4).}$$

Definition A.5 Normality
A is *normal* if its height is unity and *subnormal* if this is not the case.

Definition A.6 Fuzzy singleton
A fuzzy set whose support is a single point in U with $\mu_A(u) = 1.0$ is called a *fuzzy singleton*.

Definition A.7: α-level set
If A is a fuzzy subset of U, then an *α-level set* of A is a non-fuzzy set denoted by A_α which comprises all elements of U whose grade of membership in A is greater than or equal to α. In symbols:

$$A_\alpha = \{u \mid \mu_A(u) \geq \alpha\} \tag{A.5}.$$

Definition A.8 Convex and concave fuzzy sets
A fuzzy set A is convex if and only if for all $\lambda \in [0,1]$ and all u_1, u_2 in U:

$$\mu_A(\lambda u_1 + (1-\lambda)u_2) \geq \min(\mu_A(u_1), \mu_A(u_2)) \tag{A.6}.$$

In terms of the α-level set of A, A is convex if and only if the A_α are convex for all $\alpha \in (0,1]$. Dually, A is concave if and only if:

$$\mu_A(\lambda u_1 + (1-\lambda)u_2) \leq \max(\mu_A(u_1), \mu_A(u_2)) \tag{A.7}.$$

A.2 Operations on fuzzy sets

Assume that A and B are fuzzy subsets of U. Among the basic operations which can be performed on fuzzy sets are the following:

1. The *complement* of A is denoted by A' and is defined by,

$$A' \underline{\underline{\Delta}} \int_U (1 - \mu_A(u))/u \tag{A.8}.$$

2. The *union* of fuzzy sets A and B is denoted by $A \cup B$ and is defined by,

$$A \cup B \underline{\underline{\Delta}} \int_U (\mu_A(u) \vee \mu_B(u))/u \tag{A.9},$$

where $\vee$ is the symbol for max.

3. The *intersection* of A and B is denoted by $A \cap B$ and is defined by:

$$A \cap B \underline{\underline{\Delta}} \int_U (\mu_A(u) \wedge \mu_B(u))/u \tag{A.10},$$

where $\wedge$ is the symbol for min.

4. The *product* of A and B is denoted by AB and is defined by,

$$AB \underline{\underline{\Delta}} \int_U \mu_A(u)\mu_B(u)/u \tag{A.11}.$$

5. The *involution* or A^{α}, where α is any positive number, is defied as:

$$A^{\alpha} \underline{\underline{\Delta}} \int_U (\mu_A(u))^{\alpha} / u \tag{A.12}.$$

As a special case of (A.12), the operation of *concentration* (*CON*) is defined as:

$$CON(A) \underline{\underline{\Delta}} A^2 \tag{A.13},$$

while that of *dilation* (*DIL*) is expressed by,

$$DIL(A) \underline{\underline{\Delta}} A^{0.5} \tag{A.14}.$$

6. The *bounded sum* of A and B is denoted by $A \oplus B$ and is defined by,

$$A \oplus B \underline{\underline{\Delta}} \int_U 1 \wedge (\mu_A(u) + \mu_B(u)) / u \tag{A.15},$$

where + is the arithmetic sum.

7. The *bounded difference* of A and B is denoted by $A \ominus B$ and is defined by,

$$A \ominus B \underline{\underline{\Delta}} \int_U 0 \vee (\mu_A(u) - \mu_B(u)) / u \tag{A.16}$$

where – is the arithmetic difference.

8. The *left-square* of A is denoted by ${}^2A$ and is defined by,

$${}^2A \underline{\underline{\Delta}} \int_V \mu_A(u) / u^2 \tag{A.17},$$

where $V \underline{\underline{\Delta}} \{u^2 \mid u \in U\}$. More generally,

$${}^{\alpha}A \underline{\underline{\Delta}} \int_V \mu_A(u) / u^{\alpha} \tag{A.18},$$

where $V \underline{\underline{\Delta}} \{u^{\alpha} \mid u \in U\}$.

9. If $A_1,\ldots,A_n$ are fuzzy subsets of U, and $w_1,\ldots,w_n$ are non-negative weights adding up to unity, then a *convex combination* of $A_1,\ldots,A_n$ is a fuzzy set A whose membership function is expressed by,

$$\mu_A = w_1 \mu_{A_1} + \cdots + w_n \mu_{A_n} \tag{A.19},$$

where + denotes the arithmetic sum.

10. If $A_1,\ldots,A_n$ are fuzzy subsets of $U_1,\ldots,U_n$, respectively, the *cartesian product* of $A_1,\ldots,A_n$ is denoted by $A_1 \times \cdots \times A_n$ and is defined as a fuzzy subset of $U_1 \times \cdots \times U_n$ whose membership function is expressed by,

$$\mu_{A_1} \times \cdots \times \mu_{A_n}(u_1,\ldots,u_n) = \mu_{A_1}(u_1) \wedge \cdots \wedge \mu_{A_n}(u_n) \quad \text{(A.20).}$$

Equivalently,

$$A_1 \times \cdots \times A_n = \int_{U_1 \times \cdots \times U_n} (\mu_{A_1}(u_1) \wedge \cdots \wedge \mu_{A_n}(u_n)) / (u_1,\ldots,u_n) \quad \text{(A.21).}$$

A.3 T-norm and S-norm

Definition A.9 T-norm

A *T-norm* operator [Jang *et al*, 1997] is a two-place function $T(\cdot,\cdot)$ satisfying:

$$\left.\begin{array}{ll} T(0,0) = 0,\ T(a,1) = T(1,a) = a & \text{(boundary)} \\ T(a,b) \le T(c,d) \text{ if } a \le c \text{ and } b \le d & \text{(monotonicity)} \\ T(a,b) = T(b,a) & \text{(commutativity)} \\ T(a,T(b,c)) = T(T(a,b),c) & \text{(associativity)} \end{array}\right\} \quad \text{(A.22).}$$

The first requirement imposes the correct generalization to crisp sets. The second requirement implies that a decrease in the membership values in A and B cannot produce an increase in the membership value in $A \cap B$. The third requirement indicates that the operator is indifferent to the order of the fuzzy sets to be combined. Finally, the fourth requirement allows us to take the intersection of any number of sets in any order of pair-wise groupings. Four of the most frequently used T-norm operators are:

$$\left.\begin{array}{ll} \textit{Minimum}: & T_{min}(a,b) = \min(a,b) = a \wedge b. \\ \textit{Algebraic product}: & T_{ap}(a,b) = ab. \\ \textit{Bounded product}: & T_{bp}(a,b) = 0 \vee (a + b - 1). \\ \textit{Drastic product}: & T_{dp}(a,b) = \begin{cases} a, & \text{if } b = 1. \\ b, & \text{if } a = 1. \\ 0, & \text{if } a,b < 1. \end{cases} \end{array}\right\} \quad \text{(A.23).}$$

Definition A.10 S-norm (T-conorm)

A *S-norm* (or *T-conorm*) operator [Jang *et al*, 1997] is a two-place function $S(\cdot,\cdot)$ satisfying:

$$\left.\begin{array}{ll} S(1,1) = 1,\ S(0,a) = S(a,0) = a & \text{(boundary)} \\ S(a,b) \le S(c,d) \text{ if } a \le c \text{ and } b \le d & \text{(monotonicity)} \\ S(a,b) = S(b,a) & \text{(commutativity)} \\ S(a,S(b,c)) = S(S(a,b),c) & \text{(associativity)} \end{array}\right\} \quad \text{(A.24).}$$

The justification of these basic requirements is similar to that of the requirements for T-norm operators. Four of the most frequently used S-norm operators are:

$$\left.\begin{array}{ll} \textit{Maximum}: & S_{max}(a,b) = \max(a,b) = a \vee b. \\ \textit{Algebraic sum}: & S_{as}(a,b) = a + b - ab. \\ \textit{Bounded sum}: & S_{bs}(a,b) = 1 \wedge (a + b). \\ \textit{Drastic sum}: & S_{ds}(a,b) = \begin{cases} a, & \text{if } b = 0. \\ b, & \text{if } a = 0. \\ 1, & \text{if } a,b > 0. \end{cases} \end{array}\right\} \quad \text{(A.25).}$$

APPENDIX B

SIMULINK MODELS

B.1 Main SIMULIK models used in Chapter 5

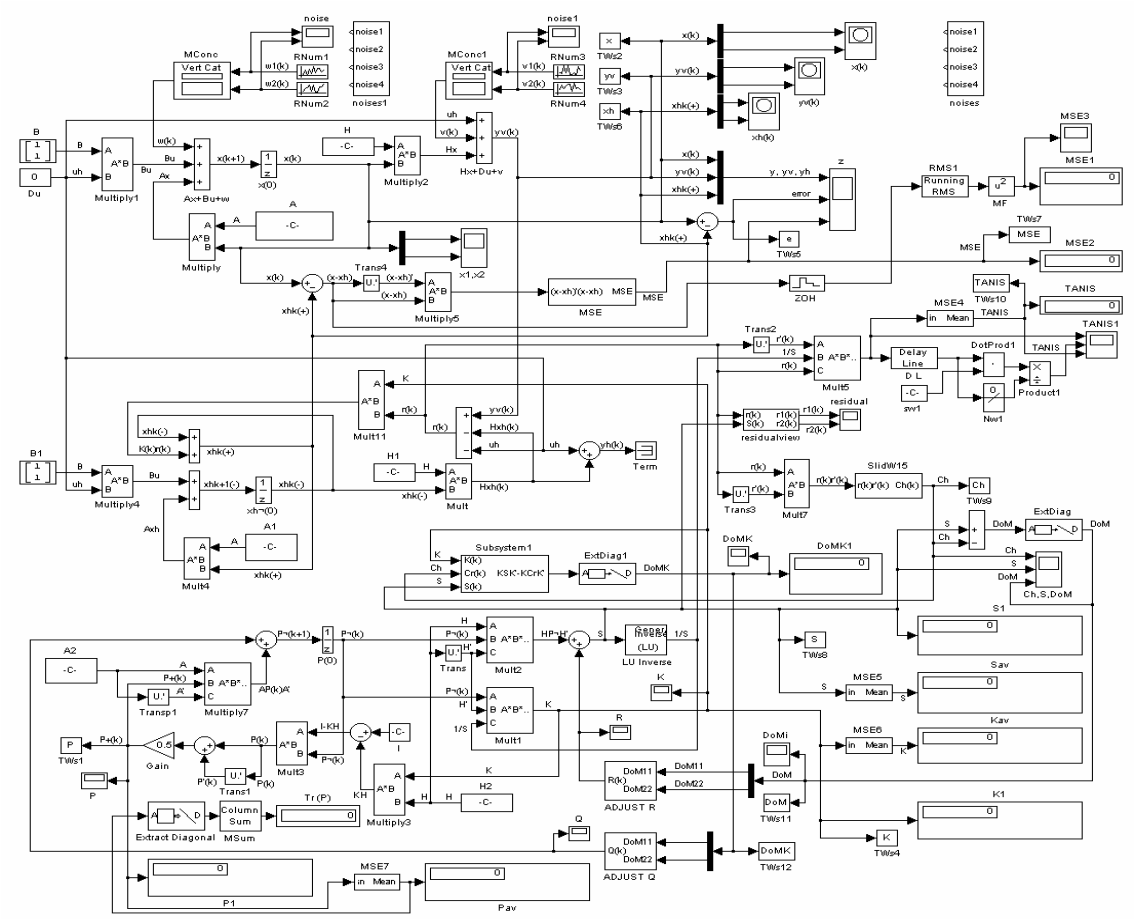


Figure B.1 The FL-AKF model.

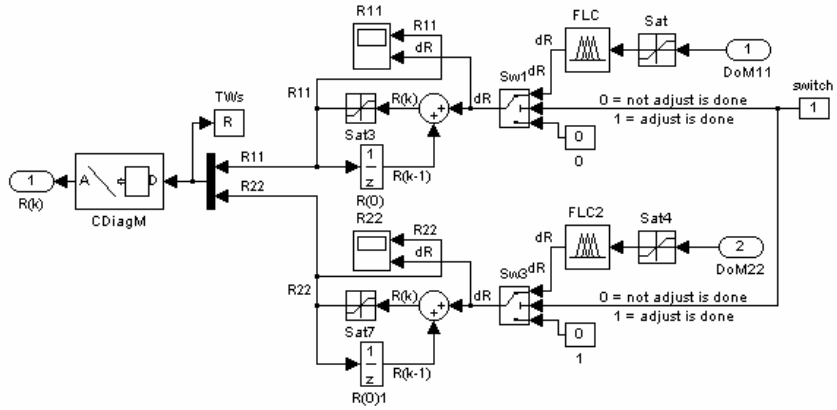


Figure B.2 Subsystem ADJUST R: FL-AKF model.

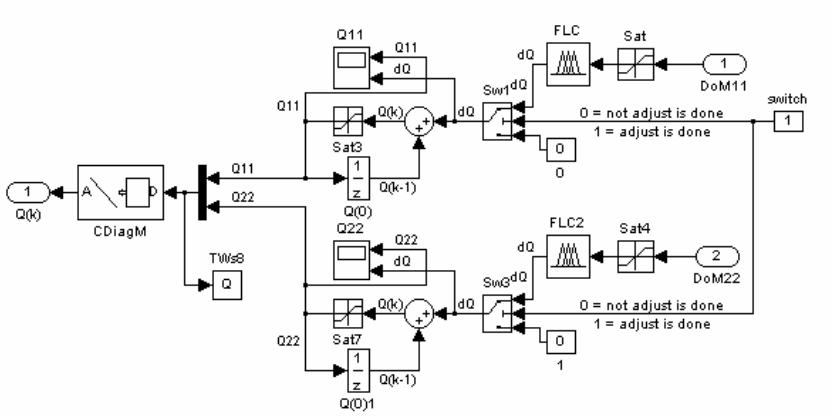


Figure B.3 Subsystem ADJUST Q: FL-AKF model.

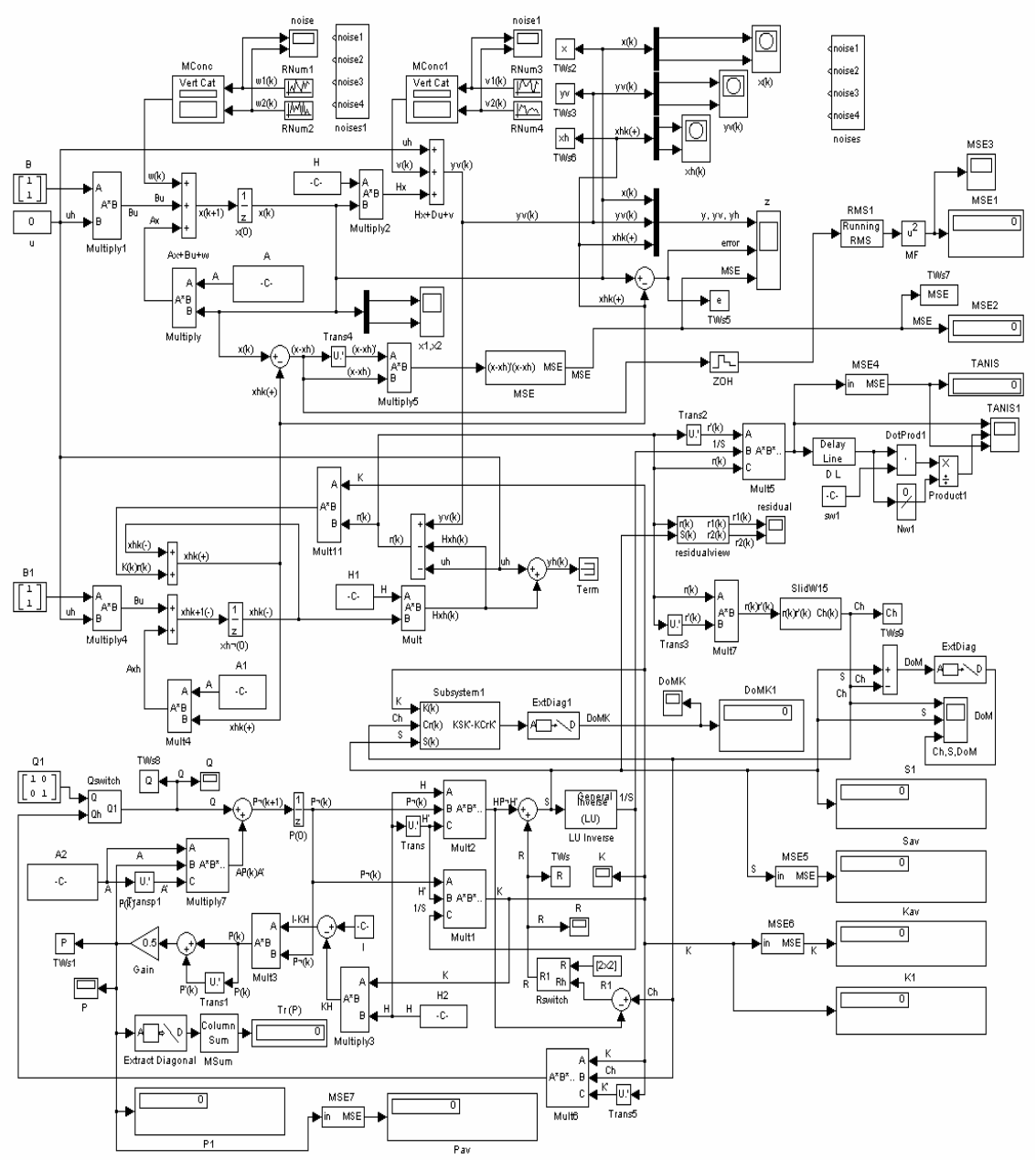


Figure B.4 The TAKF-IAE model.

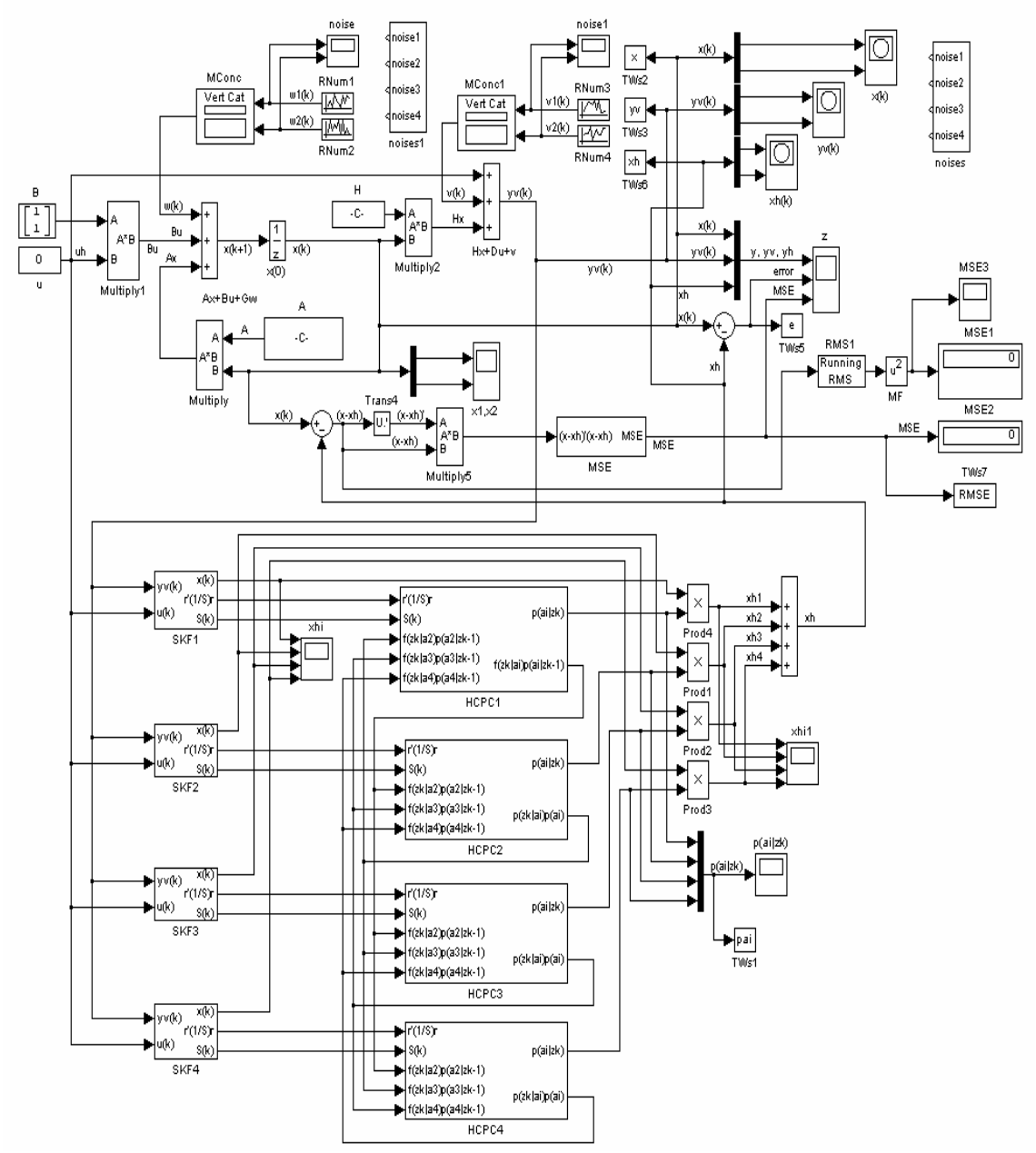


Figure B.5 The TAKF-MMAE model.

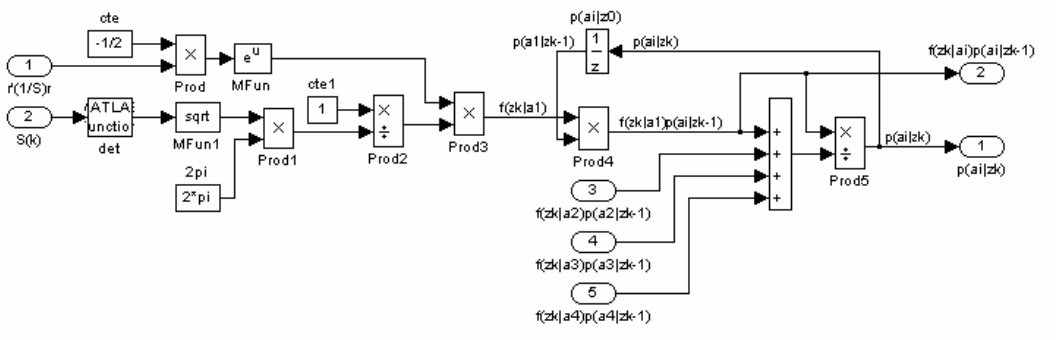


Figure B.6 Subsystem HCPC1: TAKF-MMAE model.

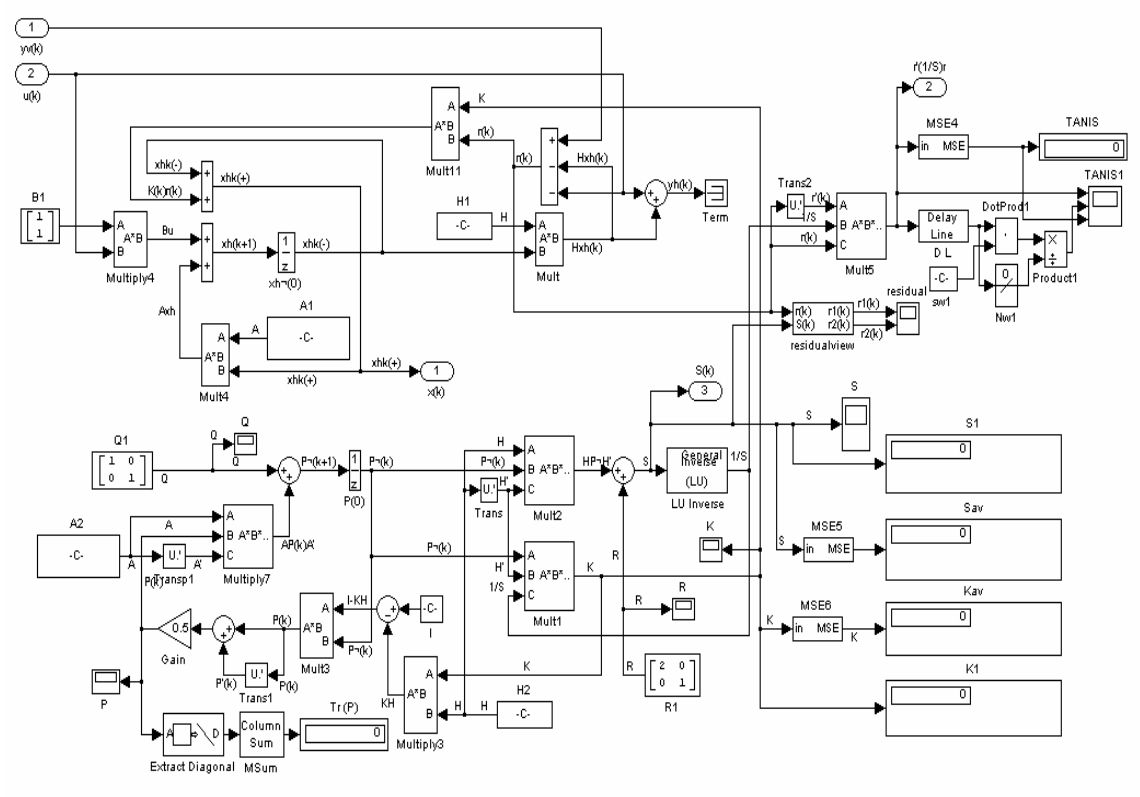


Figure B.7 Subsystem SKF1: TAKF-MMAE model.

B.2 Main SIMULIK models used in Chapter 6

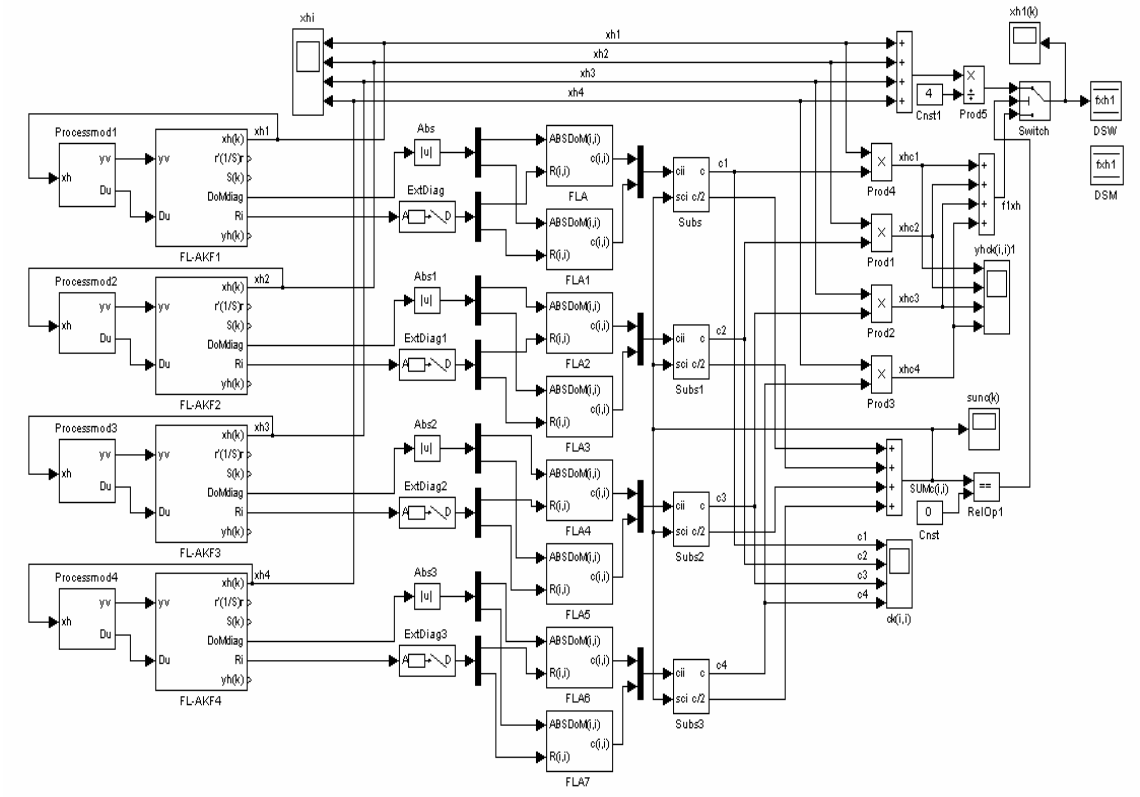


Figure B.8 The FL-AKF-FLA model.

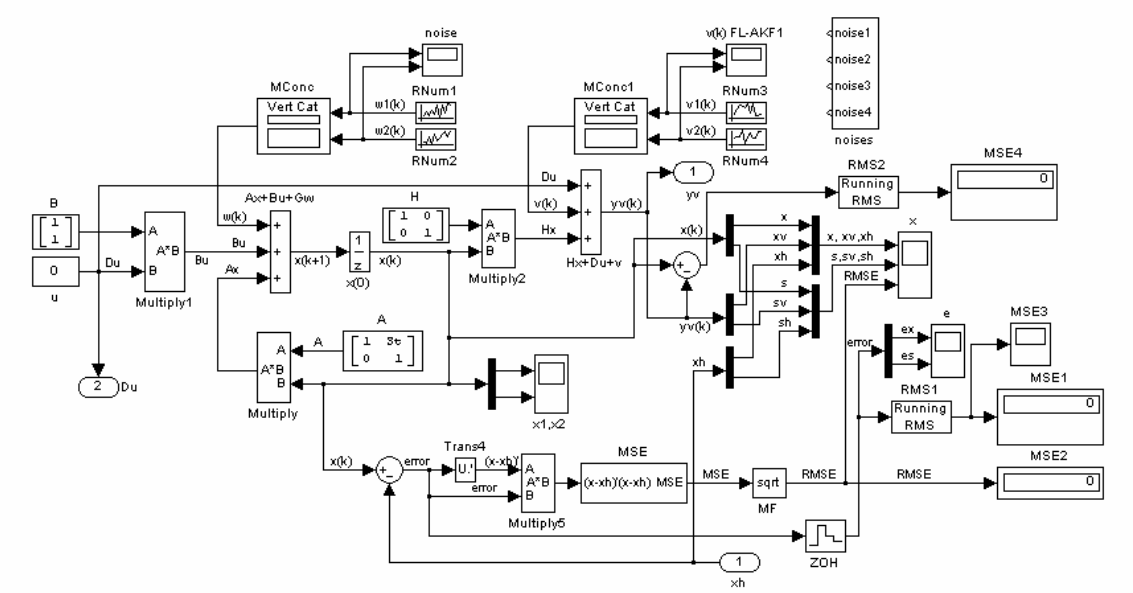


Figure B.9 Subsystem Processmod1: FL-AKF-FLA model.

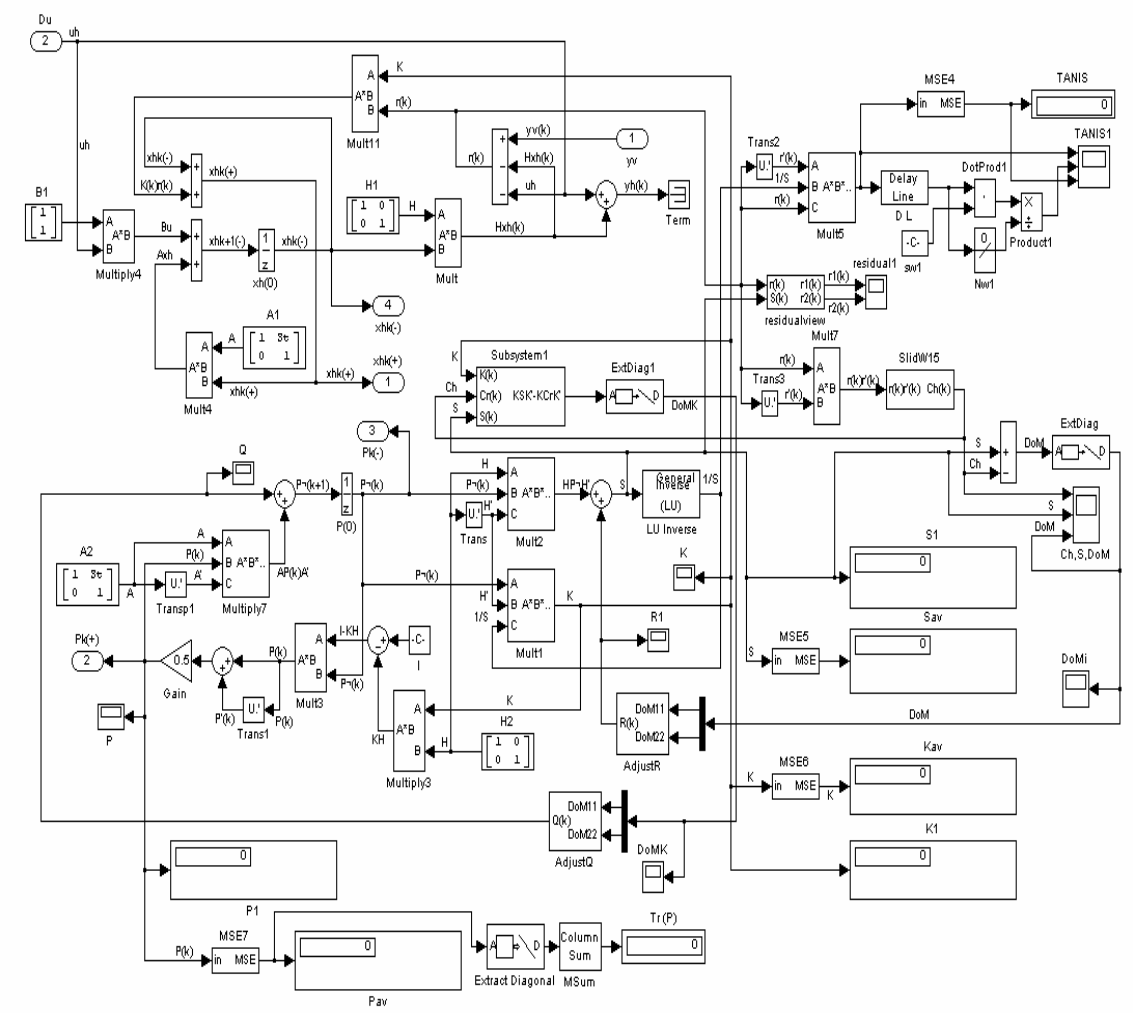


Figure B.10 Subsystem FL-AKF1: FL-AKF-FLA model.

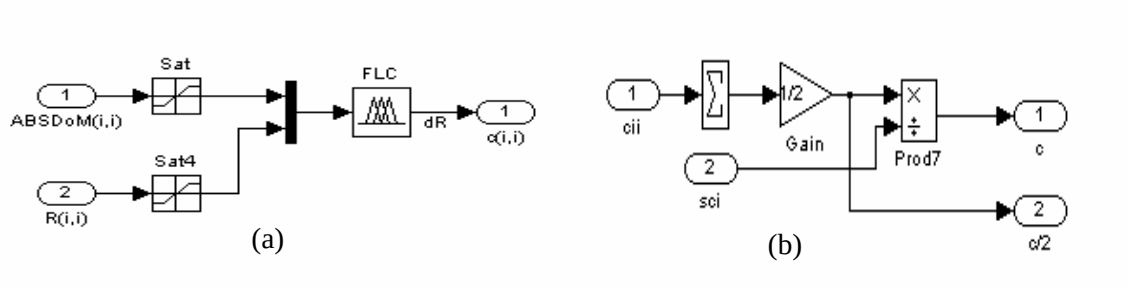


Figure B.11 (a) Subsystem FLA; (b) subsystem Subs: FL-AKF-FLA model.

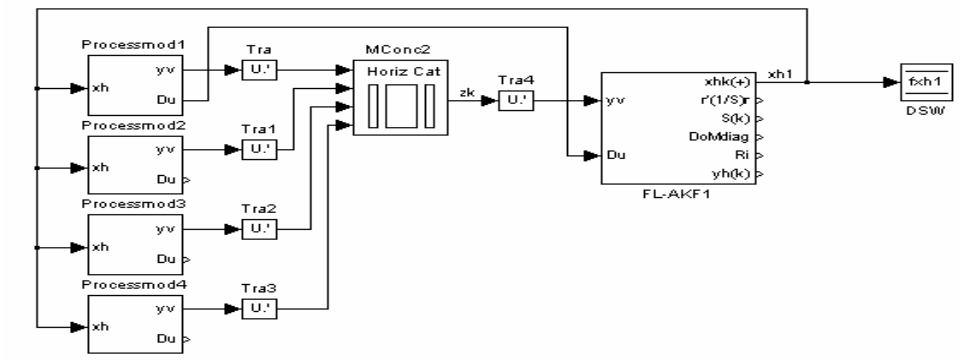


Figure B.12 The FL-ACKF model.

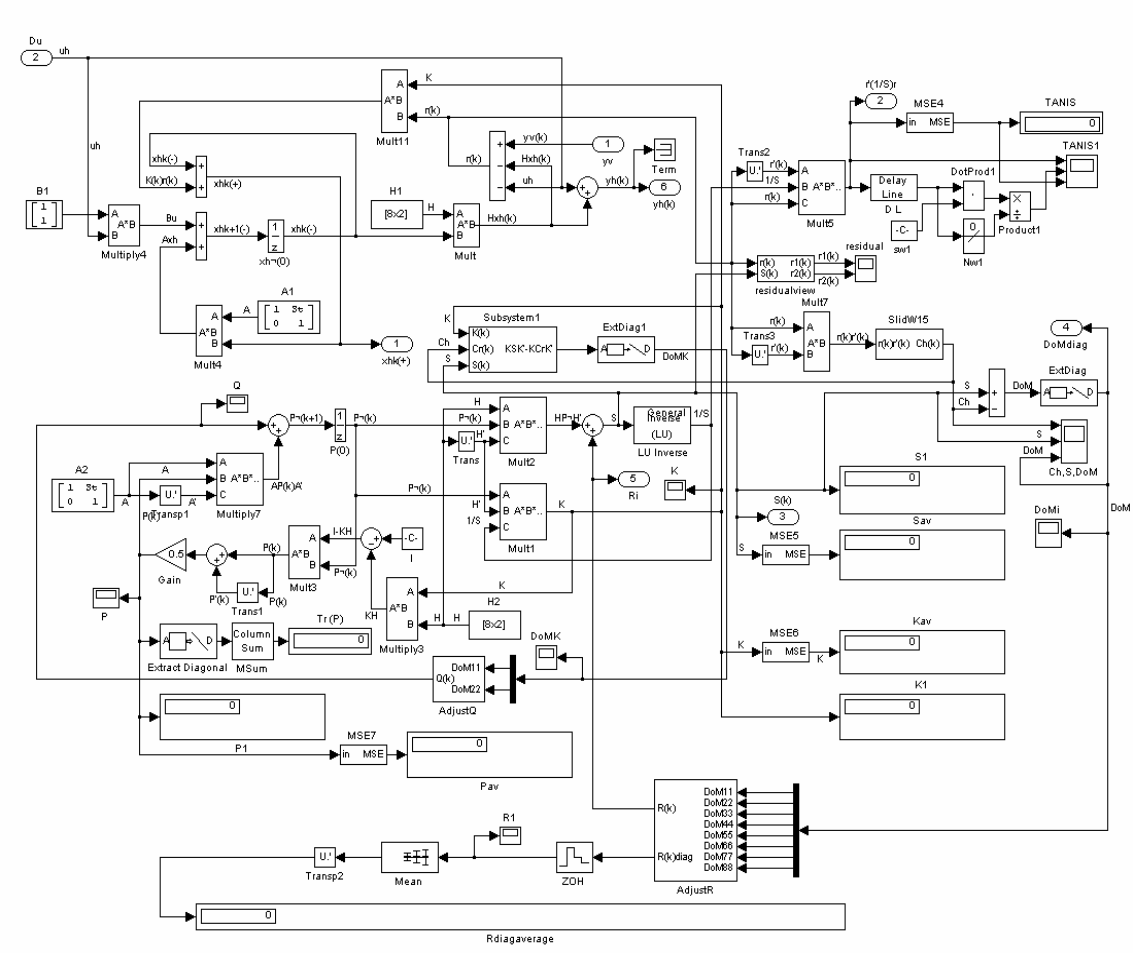


Figure B.13 Subsystem FL-AKF1: FL-ACKF model.

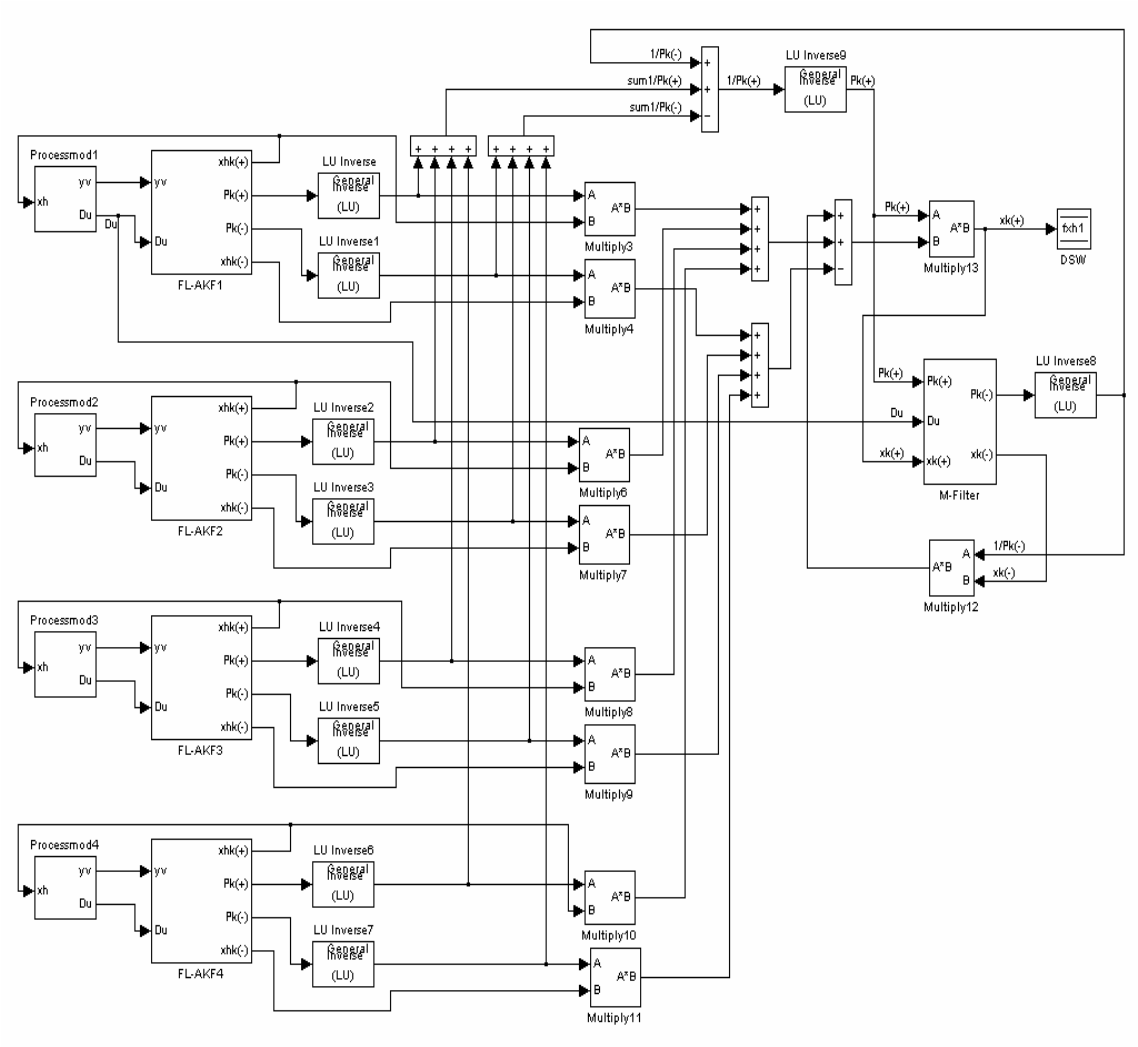


Figure B.14 The FL-ADKF model.

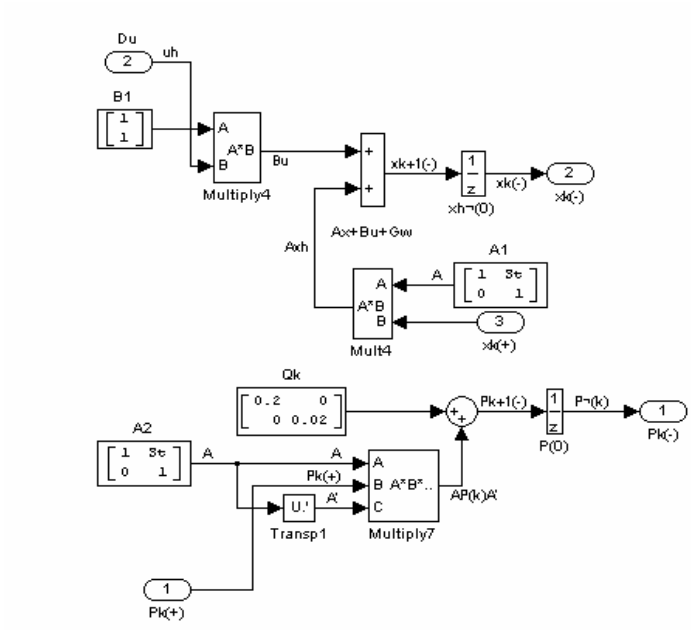


Figure B.15 Subsystem M-Filter: FL-ADKF model.

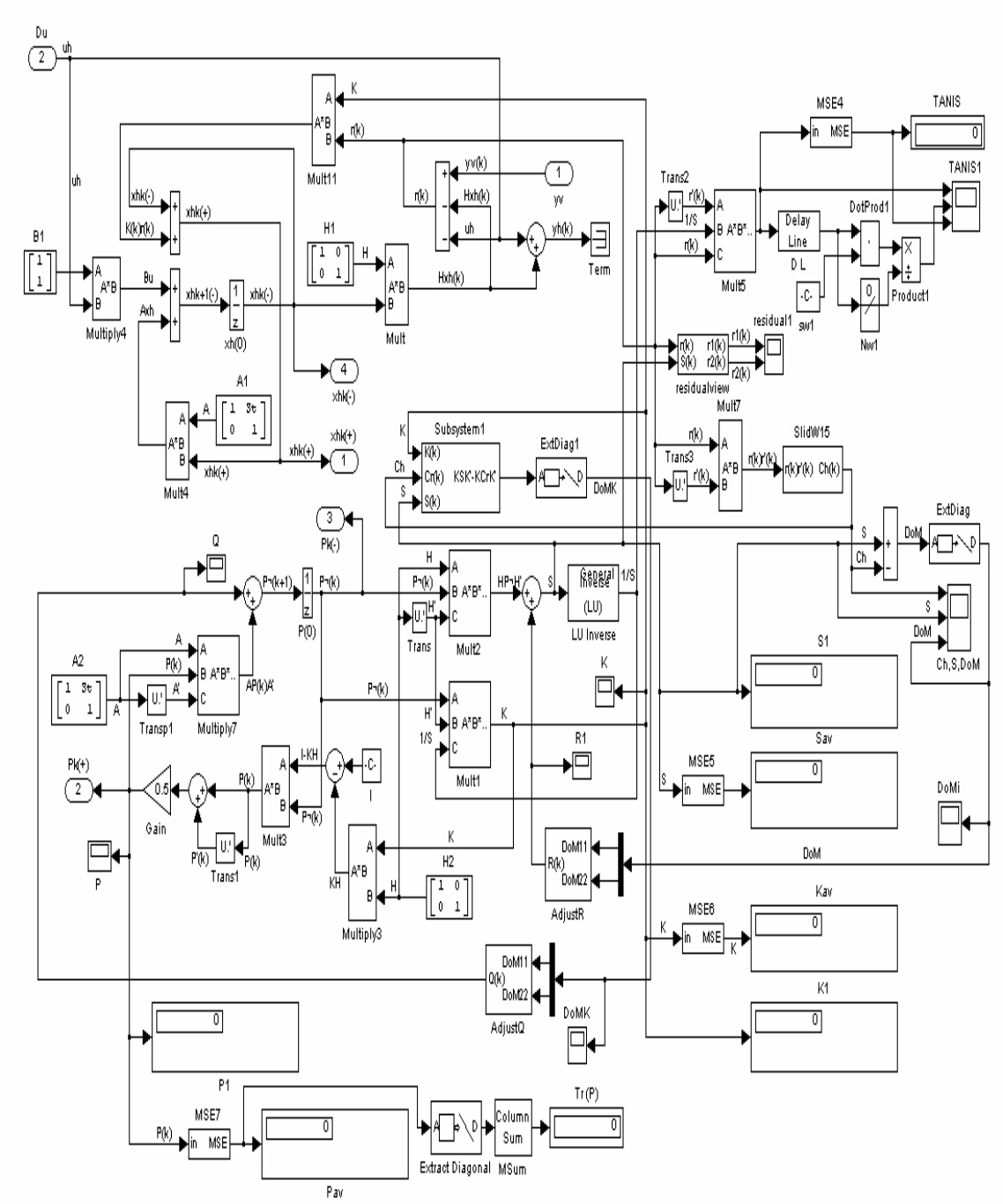


Figure B.16 Subsystem FL-AKF1: FL-ADKF model.

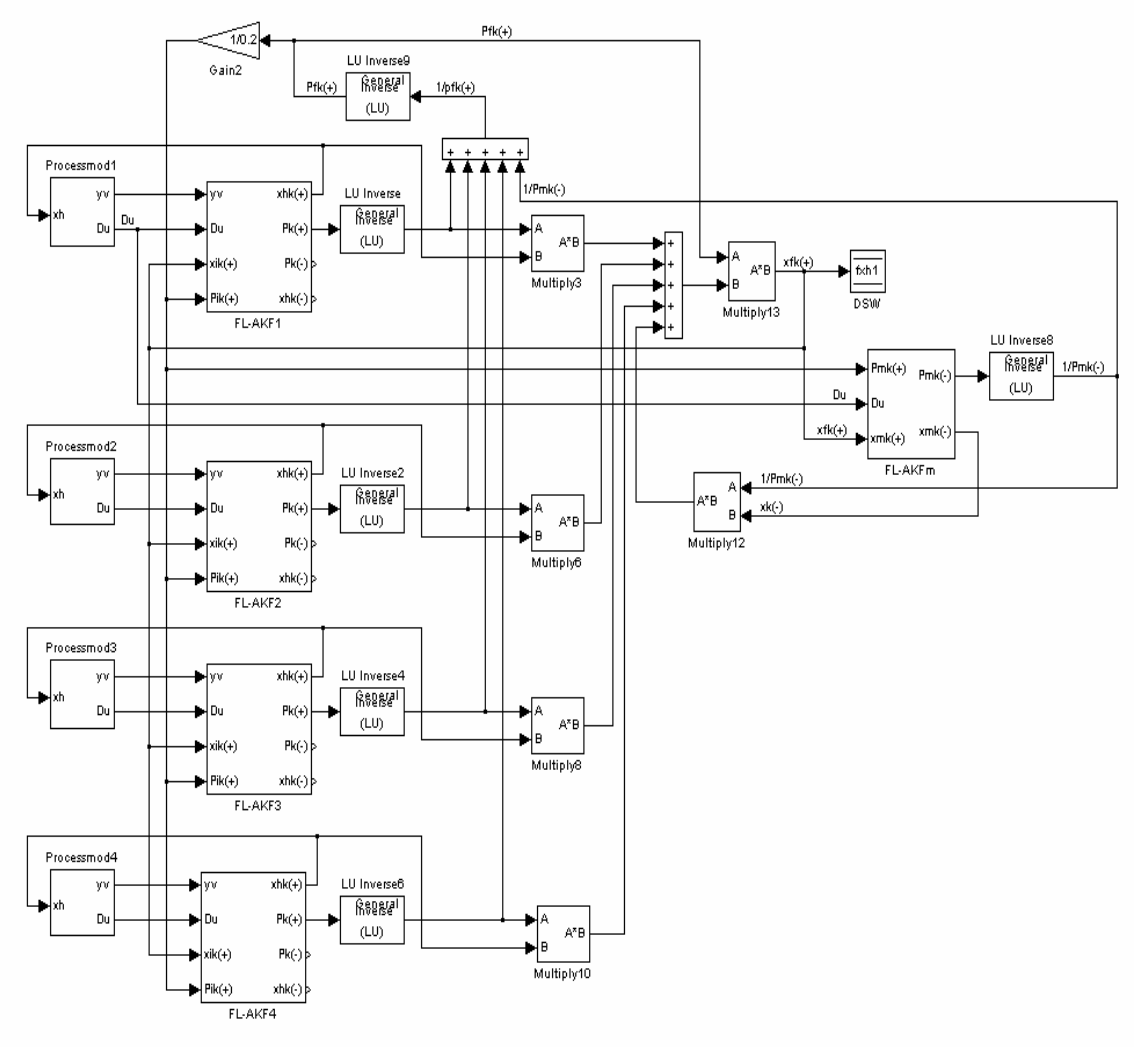


Figure B.17 The FL-AFKF model.

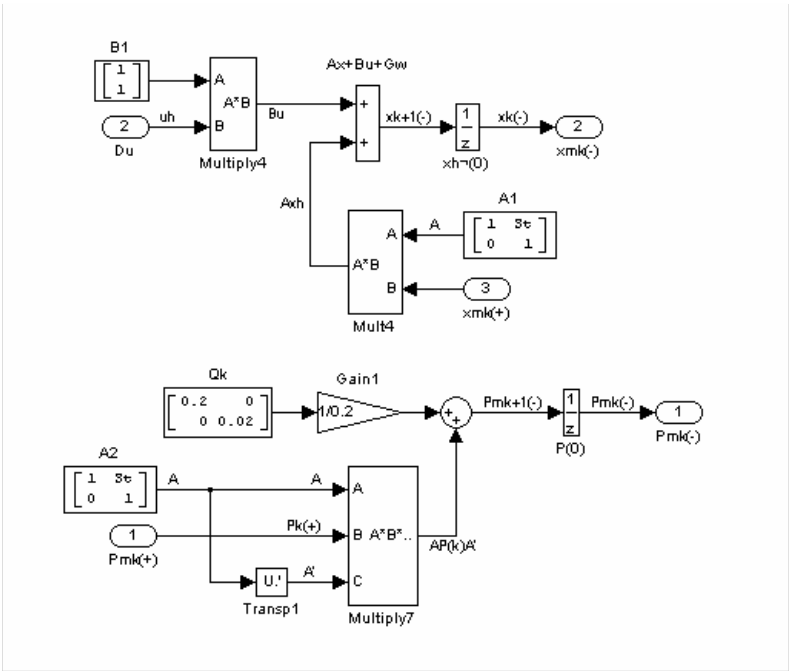


Figure B.18 Subsystem FL-AKFm: FL-AFKF model.

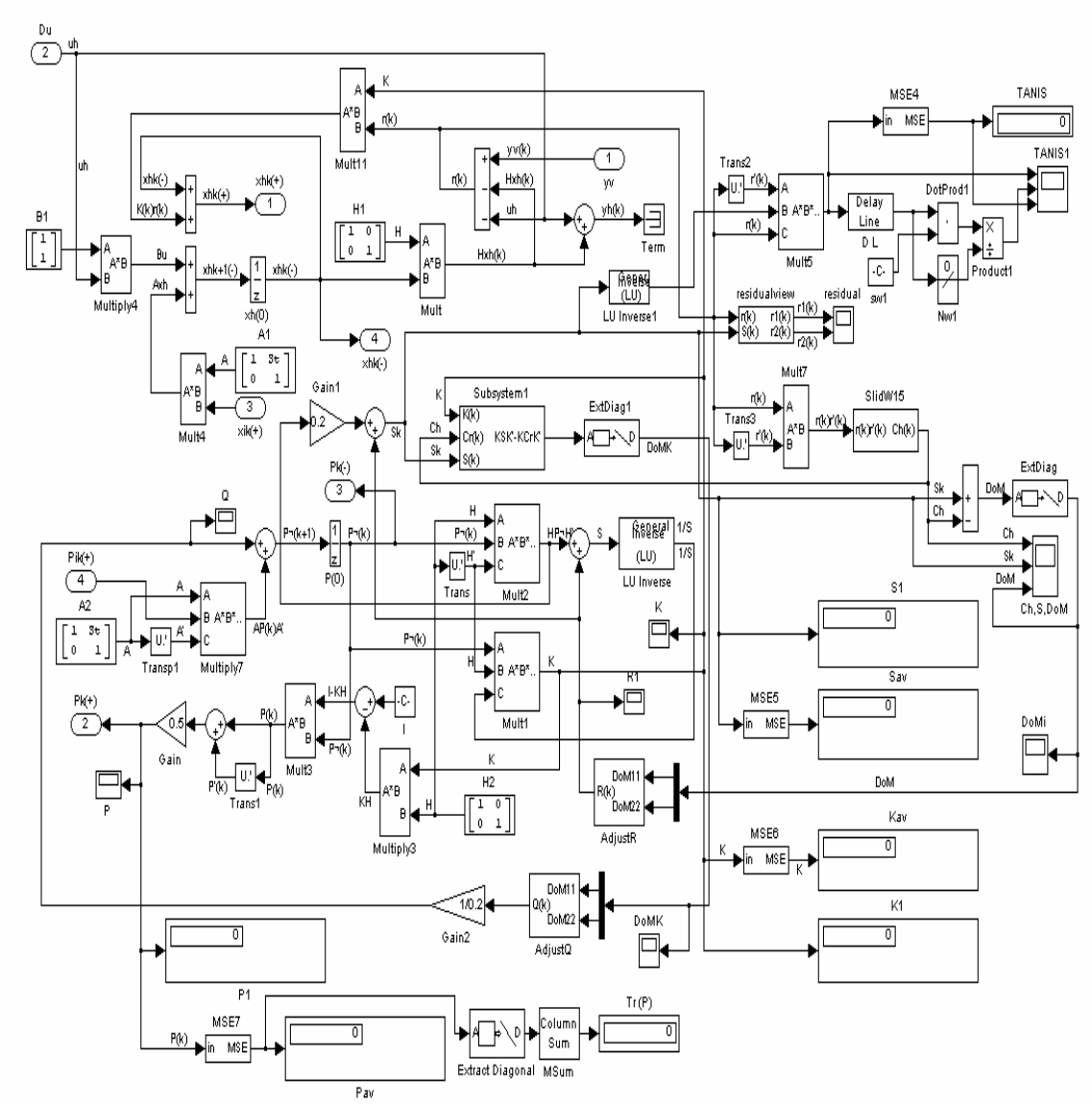


Figure B.19 Subsystem FL-AKF1: FL-AFKF model.

B.3 Main SIMULIK models used in Chapter 7

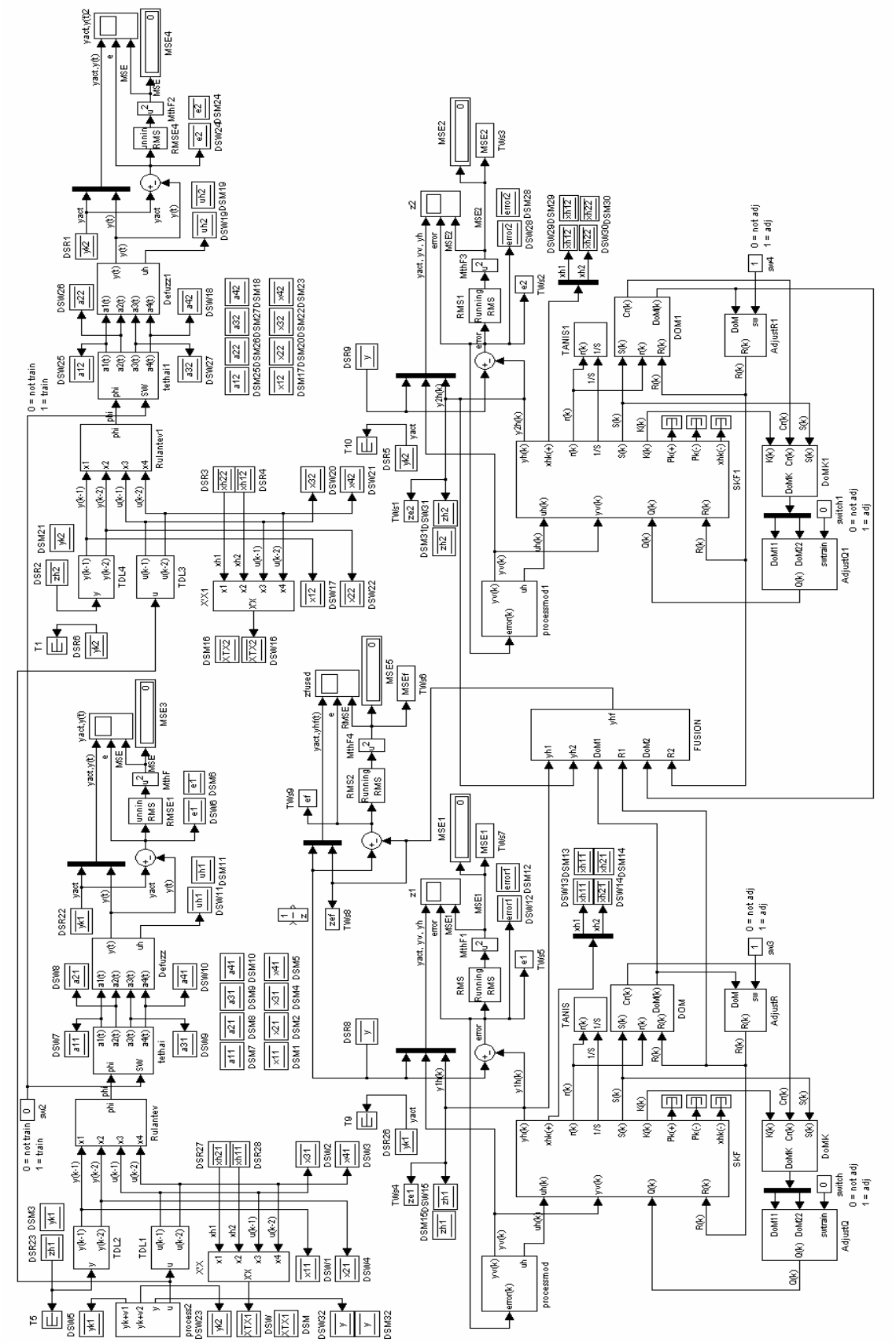
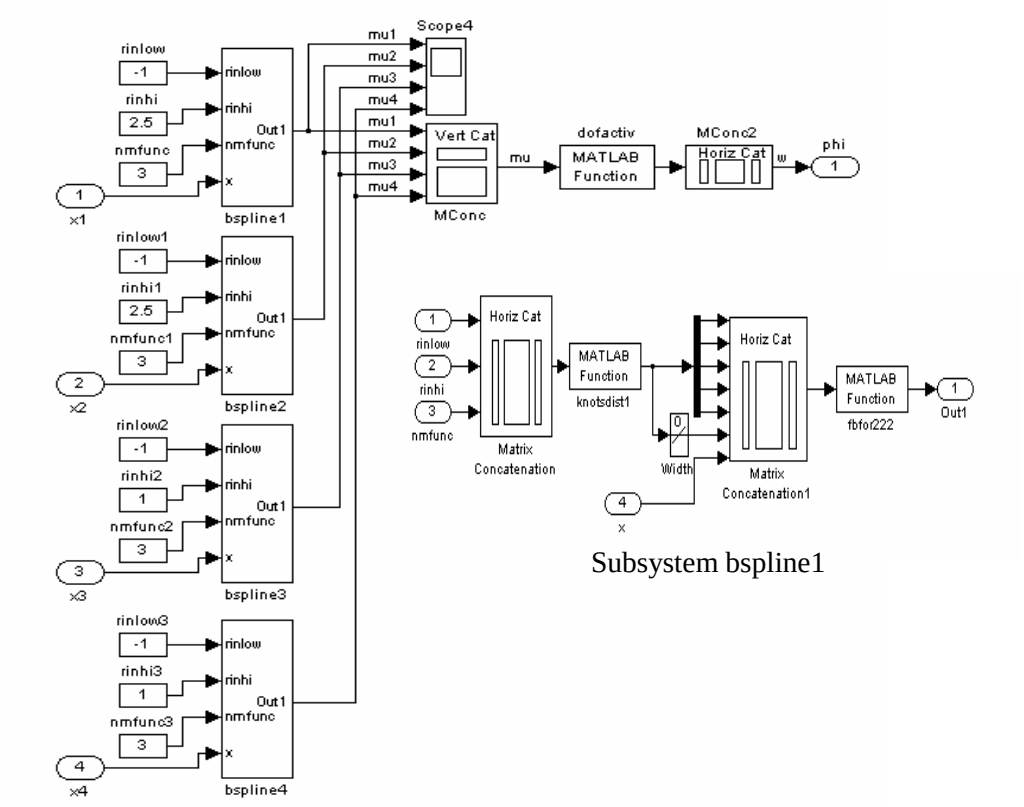
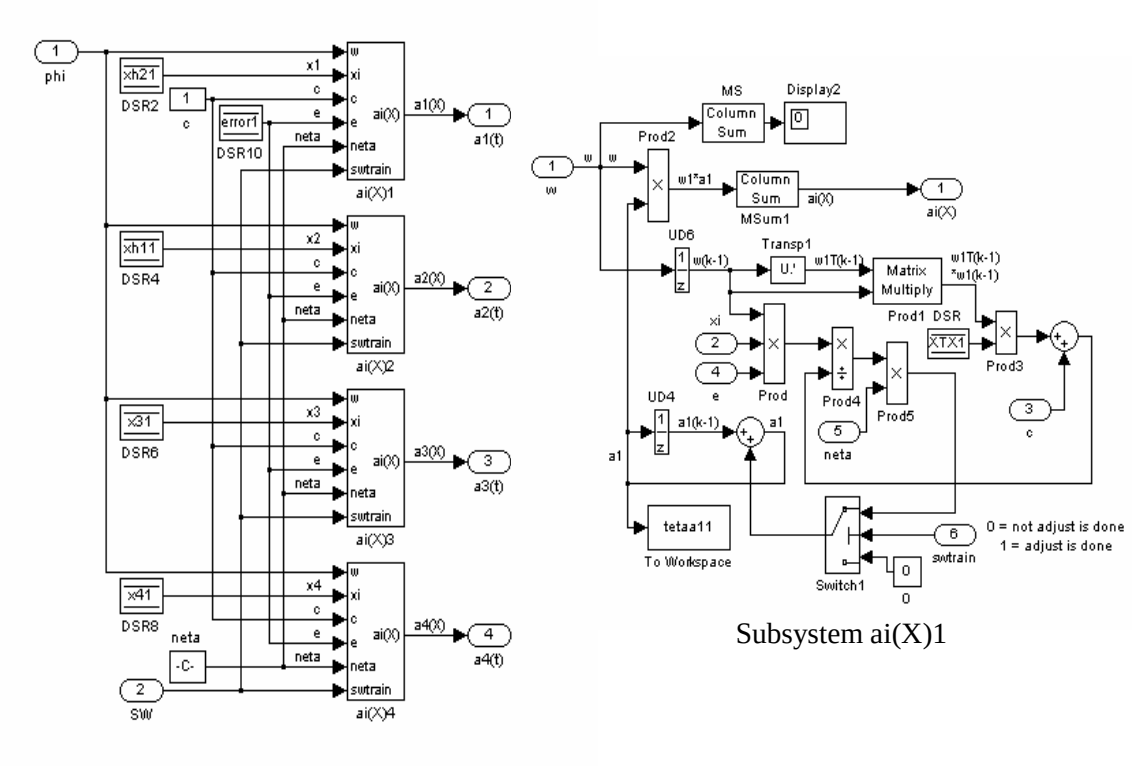


Figure B.20 The Neuro-Fuzzy-AKF and FL-AKF-FLA fusion model.



Subsystem bspline1

Figure B.21 Subsystem Rulantev: Neuro-Fuzzy-AKF, FL-AKF-FLA model.



Subsystem ai(X)1

Figure B.22 Subsystem tethai: Neuro-Fuzzy-AKF, FL-AKF-FLA model.

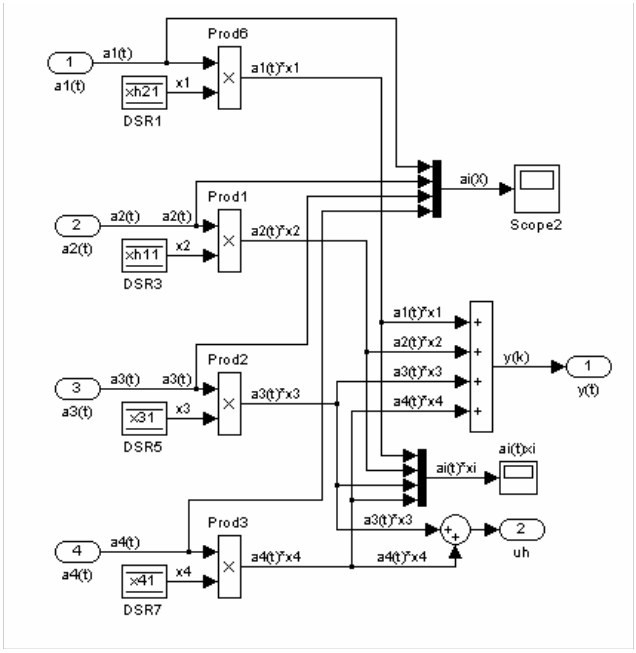


Figure B.23 Subsystem Defuzz: Neuro-Fuzzy-AKF, FL-AKF-FLA model.

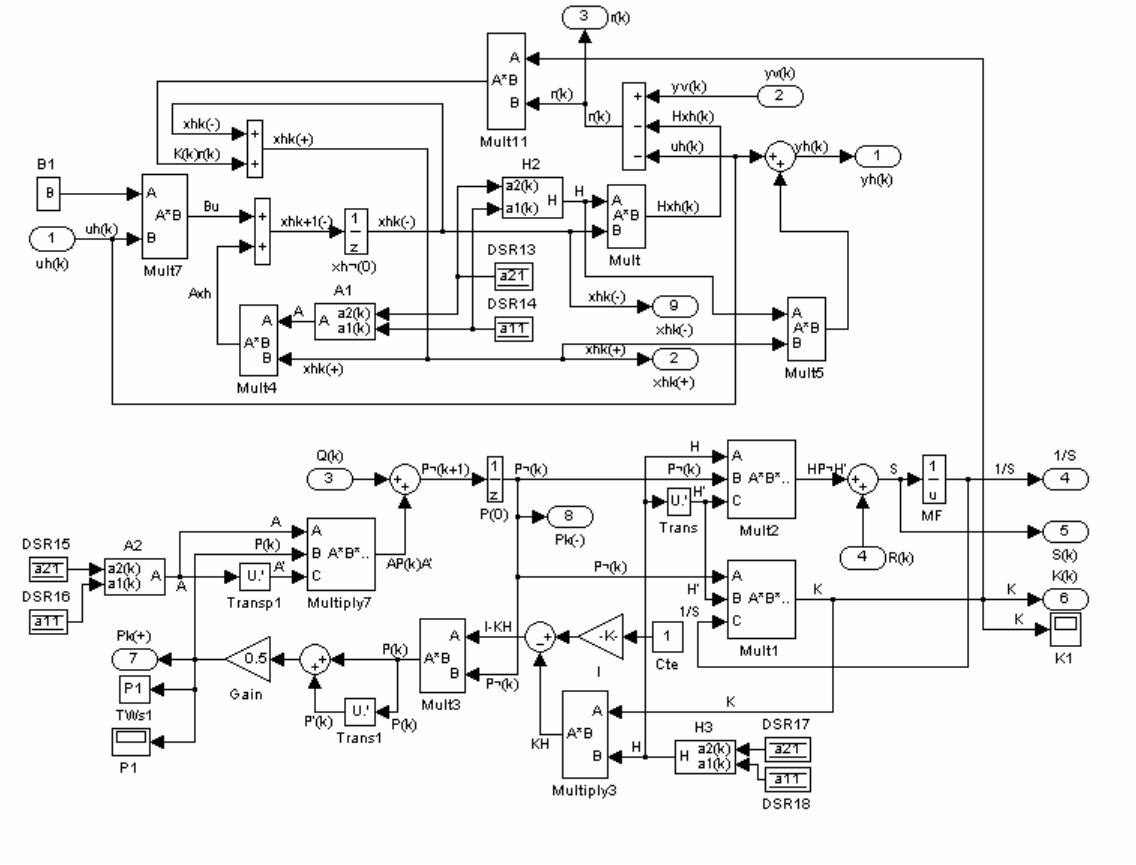


Figure B.24 Subsystem SKF: Neuro-Fuzzy-AKF, FL-AKF-FLA model.

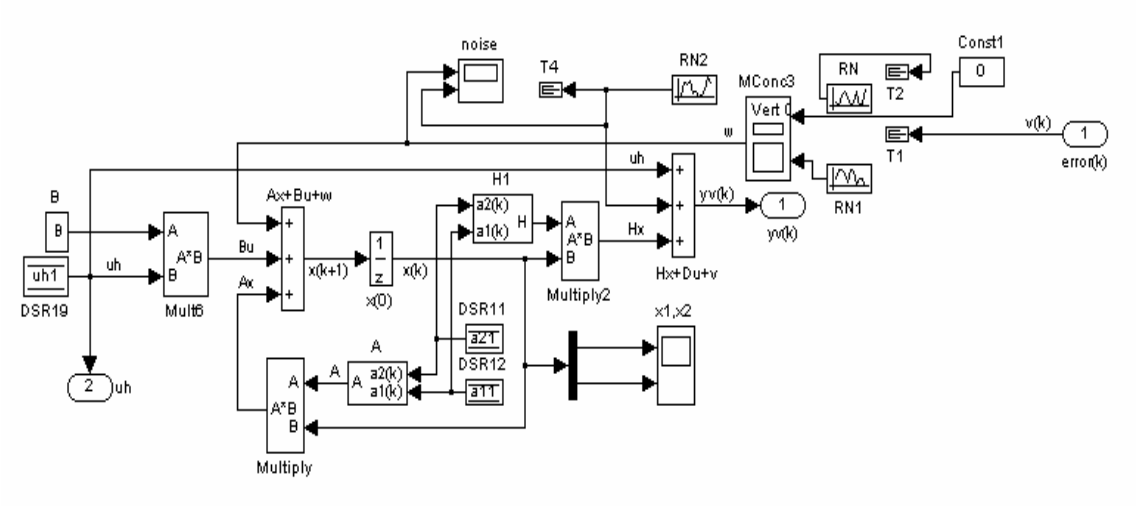


Figure B.25 Subsystem processmod: Neuro-Fuzzy-AKF, FL-AKF-FLA model.

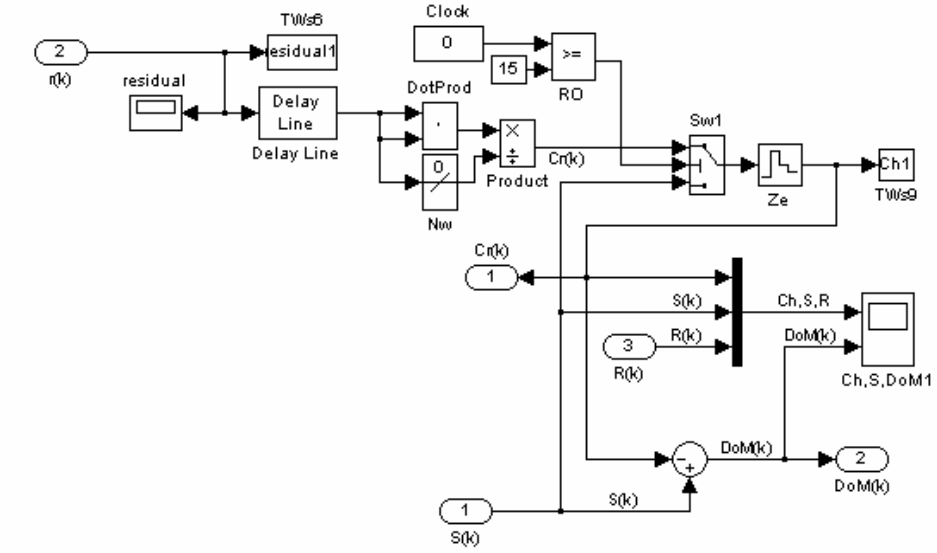


Figure B.26 Subsystem DOM: Neuro-Fuzzy-AKF, FL-AKF-FLA model.

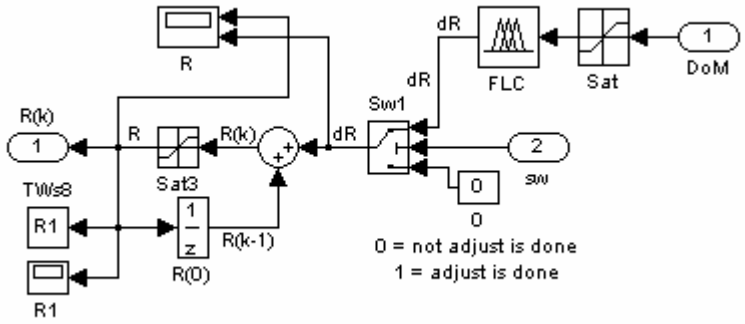


Figure B.27 Subsystem AdjustR: Neuro-Fuzzy-AKF, FL-AKF-FLA model.

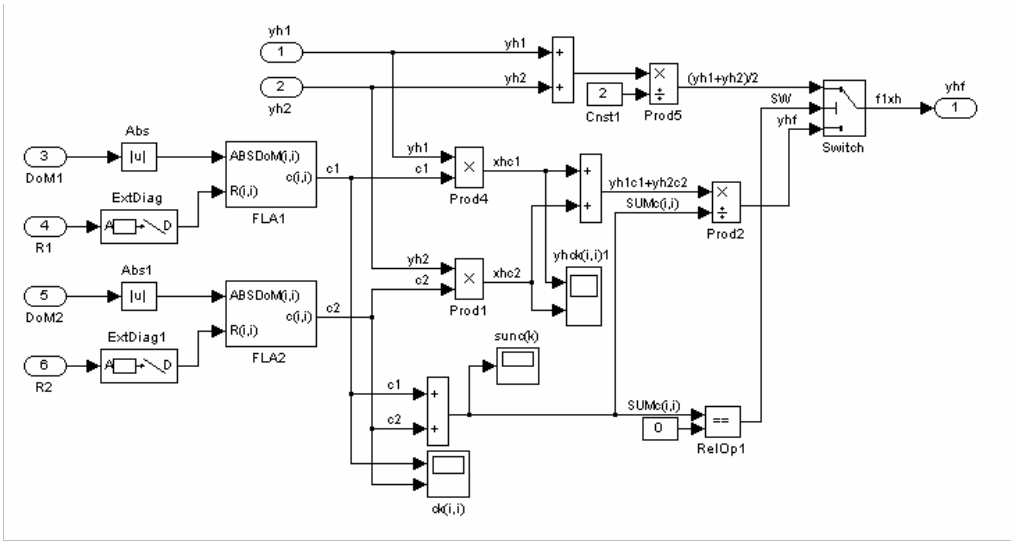


Figure B.28 Subsystem FUSION: Neuro-Fuzzy-AKF, FL-AKF-FLA model.

B.4 Main SIMULIK models used in Chapter 8

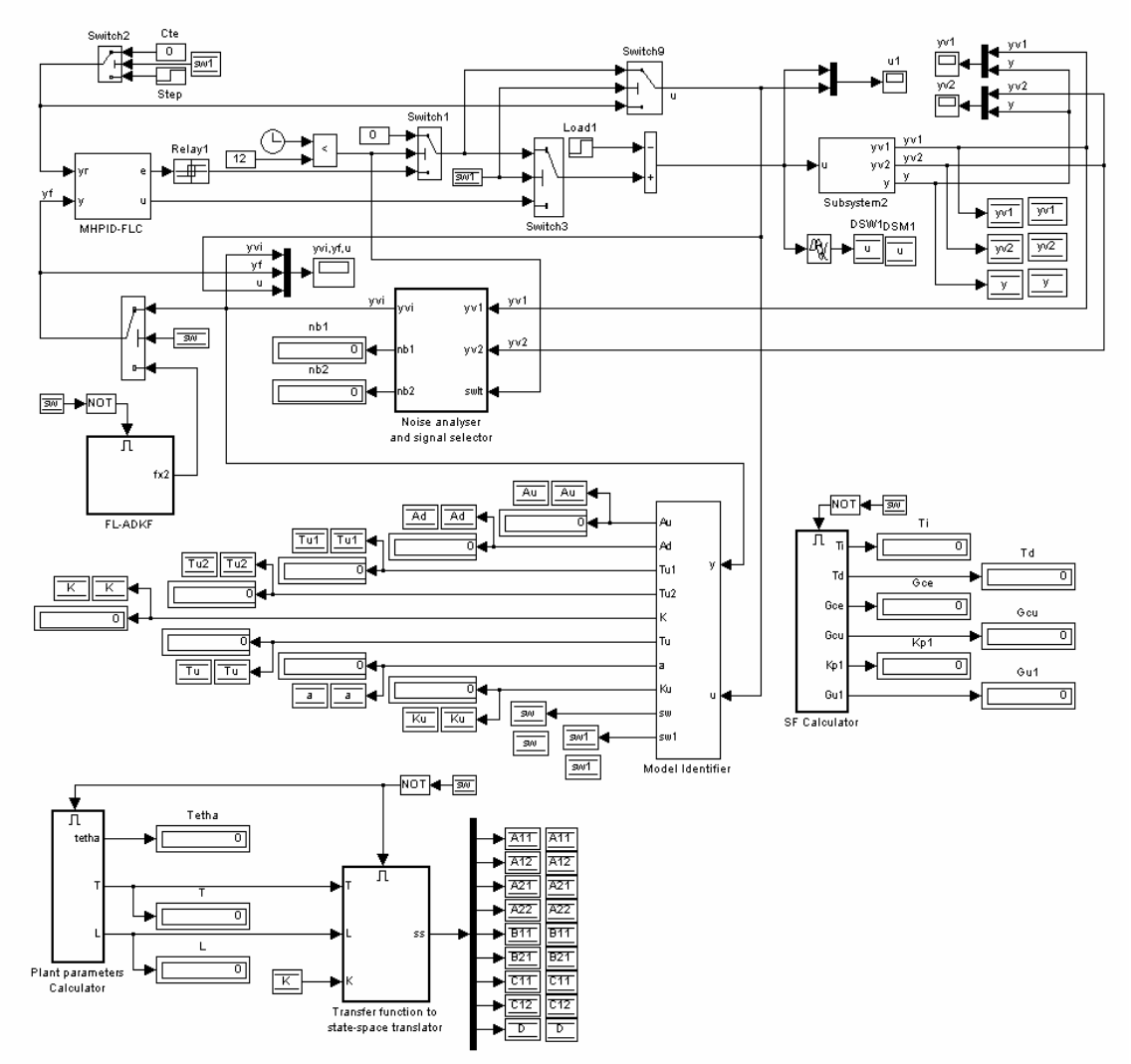


Figure B.29 The MHPID-FLC and MSDF model.

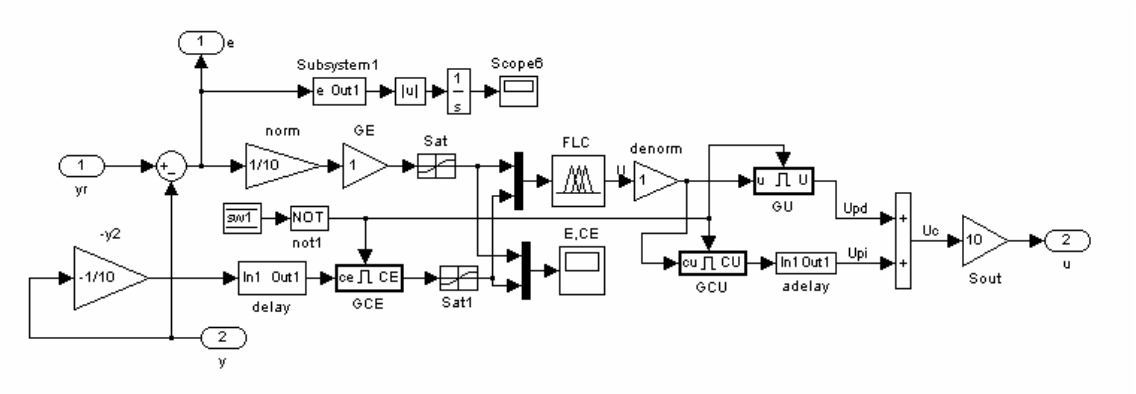


Figure B.30 Subsystem MHPID-FLC: MHPID-FLC and MSDF model.

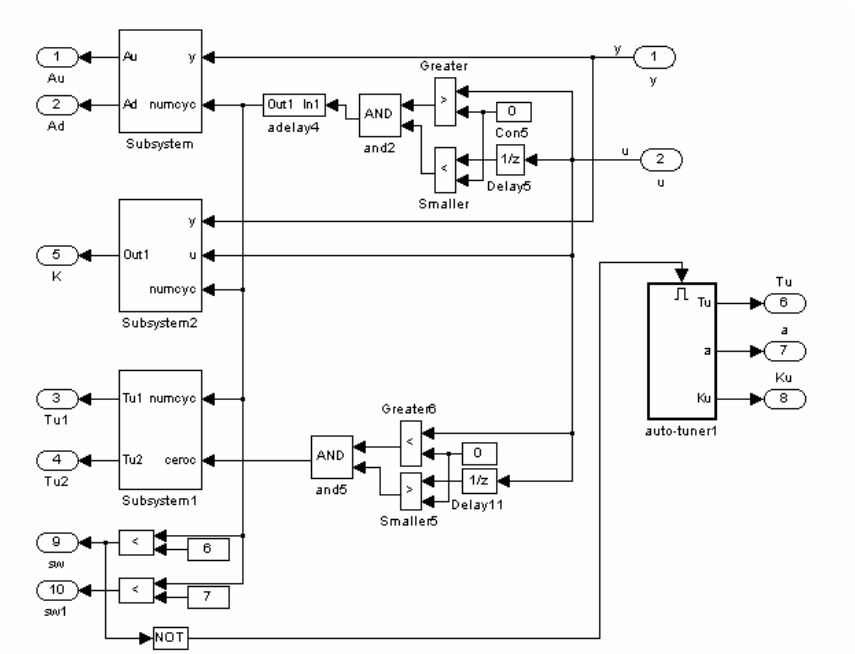


Figure B.31 Subsystem Model Identifier: MHPID-FLC and MSDF model.

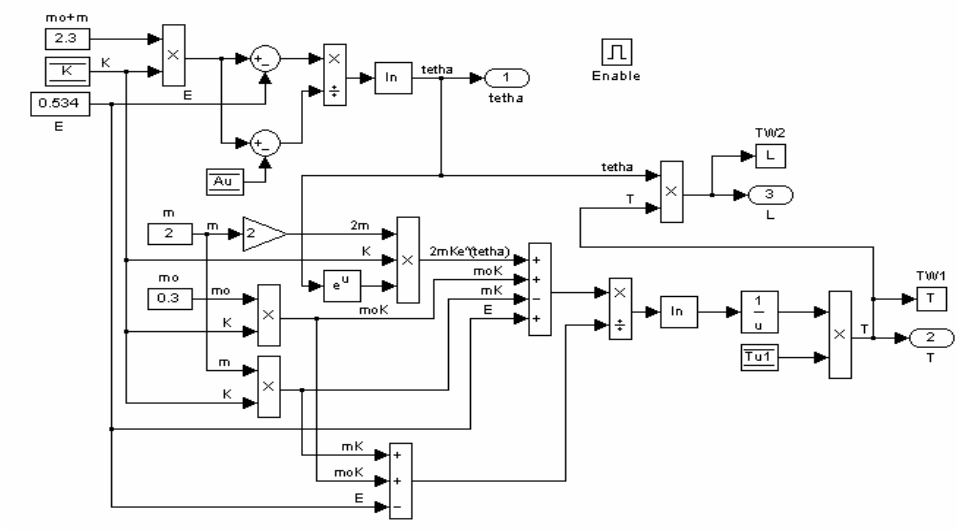


Figure B.32 Subsystem Plant Parameter Calculator: MHPID-FLC and MSDF model.