The Pennsylvania State University

The Graduate School

College of Engineering

PRIOR KNOWLEDGE BASED IDENTIFICATION OF TAKAGI-SUGENO-KANG FUZZY MODELS FOR STATIC NONLINEAR SYSTEMS

A Dissertation in

Engineering Science and Mechanics

By

Ashutosh Tewari

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2009

The dissertation of Ashutosh Tewari was reviewed and approved* by the following:

*Signatures are on file in the Graduate School.

Abstract

Data-driven model of a physical system is a mathematical model that has been built solely using historically available input-output pairs. In such type of models we usually try to identify the mapping function (linear or nonlinear) that transforms inputs from a multivariate space to a univariate output space (Multiple-Input-Single-Output/MISO systems). The data-driven models are especially needed when constructing a physical model (based on first principles) of a process is difficult due to the lack of understanding of the underlying physical phenomenon/phenomena. Although quite popular, majority of the data-driven models have a noticeable disadvantage in terms of their poor interpretability. In other words, although such models may correctly estimate the system outputs for a given set of inputs, the manner in which the outputs are generated does not have any physical interpretation. For this reason, data-driven models are sometimes referred as black-box models. Neuro-fuzzy models are a class of data-driven models that partly alleviates the above mentioned problem of poor interpretability. These hybrid models consist of two components. The Artificial-Neural-Network (ANN) component that enables learning from historical data and Fuzzy Logic component which tries to emulate the process of human reasoning. These models are specially desired in situations when a process output can be explained by a domain expert/experts in terms of linguistic statements. In this thesis, a type of neuro-fuzzy models called Takagi-Sugeno-Kang (TSK) models are studied as a tool for static nonlinear system modeling.

In general, identification of any data-driven model involves two important steps 1) the structure identification and 2) the parameter identification. In the 1st step, a parameterized model is constructed either using the prior knowledge or a historical dataset or a combination of both. Thereafter, in the 2nd step of parameter identification the optimal values of the model parameters are estimated. The process of parameter estimation is called training and the dataset for this purpose is referred as training dataset. The general approach of parameter estimation can be summarized as follows. An error measure is defined using the training dataset and subsequently minimized with respect to the unknown model parameters. Therefore, the parameter estimation is merely an nonlinear optimization problem. In this thesis, the focus is only on the parameters estimation of TSK models under the assumption that their structure has been fixed a priori.

In order to have consistent and near optimal parameter estimates, a sufficiently large noise-free training dataset is required. However, if we have more parameters than what can be correctly estimated from a training dataset then the performance of the trained model becomes questionable. The input-output datasets generated by majority of real world systems usually have low signal to noise ratio, which further reduces their estimation capability. Hence, it becomes imperative to devise robust parameter estimation techniques that are not significantly affected by the noise in training datasets. There is a group of estimation techniques known as *regularization techniques* that are commonly used for the identification of over-parameterized models from low quality datasets. The basic idea behind regularization is to impose certain restriction on the parameter values, so that the variance in estimated parameter values can be minimized.

Contemporary regularization techniques used for the identification of neuro-fuzzy systems are similar to the ones used for the black-box models such as ANNs. To the knowledge of the author, none of these techniques explicitly uses the qualitative knowledge that can be readily obtained from a domain expert. It is certainly counterintuitive not to use existing knowledge to regularize the parameters values of neuro-fuzzy models. In an attempt to address the aforementioned problem, a regularization technique is proposed which works by incorporating prior knowledge during the parameter estimation of TSK models. The resulting TSK models are named as *A-Priori Knowledge-based Fuzzy* Models (APKFMs). The foundation of APKFMs is a hypothetical function called *knowledge function,* which is responsible for keeping a model's parameters consistent with the domain knowledge. The proposed approach has a two fold regularization effect on the TSK models. Firstly, the incorporation of knowledge function ensures that the parameters do not take physically infeasible values during their estimation. Secondly, the use of knowledge function considerably reduces the number of unknown parameters. Therefore, less number of parameters are needed to be estimated from the available training data leading to consistent parameter estimates.

The effectiveness of the APKFMs is shown using two examples of static systems. In the first example, a 2-dimensional nonlinear toy function is constructed and subsequently approximated by an APKFM. In the second example, a real world problem pertaining to the maintenance cost estimation of electricity distribution networks is taken up. In both the examples, the performance of APKFMs is benchmarked against neuro-fuzzy models identified using the three commonly used techniques namely, *Global least square estimate*, *Local least square estimate* and *Ridge regression*. The former is an unregularized parameter estimation technique while the other two represent the state of the art regularization techniques for TSK fuzzy models. The performance of all fuzzy models are evaluated and compared based on their robustness towards the noise present in the training datasets. Different training datasets of varying quality are constructed by controlling the two variables viz. the training dataset size and the signal to noise ratio. Rigorous statistical tests confirm that on an average the robustness of APKFM is superior compared to contemporary neuro-fuzzy models.

Nontechnical Abstract

Mathematical modeling of real world systems has fascinated researchers and practitioners in different disciplines for ages. A correctly built model improves our understanding of a real system and help us perform several important tasks such as cost estimation, failure prediction, system optimization etc. The most common task of a mathematical model is determining a system's output given a set of input variables. For this purpose a mathematical expression called *mapping function* is used, which converts a system's inputs to its output. There are different ways a mapping function can be constructed, each resulting in a different type of mathematical model. The most common type of model is a physical model which is based on the precise understanding of the underlying physical phenomenon/phenomena. Unfortunately, most of the real world systems are too complex to have a precise physical description. In such case, another type of mathematical models called data-driven models can prove to be extremely useful. In data-driven models the mapping function is constructed by analyzing several historically obtained input-output pairs. The resulting mapping function depends entirely on the historical data. For this reason, a good quality historical data may lead to an accurate model of a system even though its physics is poorly understood.

Neuro-Fuzzy models belong to a class of models called grey-box data-driven models. In a grey box model the construction of the mapping function is not entirely dependent on the historical data. The fuzzy logic part of the models allows them to be constructed using the knowledge of a domain expert. A typical neuro-fuzzy model consists of several unknown parameters. The optimal values these parameters are learned from the historical data enabling the trained model to mimic the input-output relationship. In an ideal scenario, when we have ample amount of error-free historical data, the neuro-fuzzy models are ideally suited to model complex systems. Unfortunately, in real world systems we rarely have a large repository of error-free data. In such cases, neuro-fuzzy models become inconsistent with domain knowledge and behave erratically.

In this thesis, I propose a novel type of models called *A-Priori Knowledge-based Fuzzy Models* (APKFMs) that can be constructed from erroneous historical datasets, typically generated by real world systems. The two main strengths of APKFMs are 1) they can be easily constructed merely needing the qualitative knowledge of an expert and 2) their remarkable robustness even in the event of training with a noisy dataset. The aforementioned strengths of APKFM is proven using two examples of nonlinear systems. In the first historical data is generated by a known non-linear function whose output is corrupted with different levels of noise. In the second example, a real world problem is taken up pertaining to the maintenance cost estimation of a city's electricity distribution network. In both the example, the advantages of APKFM is highlighted compared to the other popular neuro-fuzzy modeling techniques.

Table of Contents

List of Figures x

List of Tables xiv

List of Symbols xv

List of Acronyms xvii

Acknowledgements xviii

Chapter 1: Introduction **1**

1.1 System Modeling 2

1.1.1) Physical Models 2

1.1.2) Data-driven Models 3

1.1.3) Semi-physical Models 4

1.2 Complexity of Data-driven Models 6

1.2.1) Decomposition of Modeling Error 8

1.2.2) Bias-Variance Tradeoff 12

1.2.3) Brief Problem Statement 13

1.3 Organization of Thesis 14

1.4 Original Contributions 15

Chapter 2: Data-driven Models for Static Systems **16**

2.1 Basis function Formulation 17

2.1.1) Artificial Neural Networks (ANNs) 19

2.1.2) Radial Basis Function Networks (RBFNs) 22

2.1.3) TSK Fuzzy Models 26

2.2 Optimization of TSK Fuzzy Models 29

2.2.1) Structure Optimization of TSK Models 29

2.2.2) Parameter Optimization of TSK Models 31

2.3 Regularized Estimation of TSK Model Parameters 34

2.3.1) Regularization by Early Stopping 34

2.3.2) Regularization by Staggered Optimization 36

2.3.3) Regularization by Constraints 36

2.3.4) Regularization by Ridge Regression 38

2.3.5) Regularization by Local Optimization 40

2.4 Problem Statement 42

Chapter 3: A-Priori Knowledge-based Fuzzy Model (APKFM) **43**

3.1 Formulation of 0th order/ Singleton APKFM 44

3.1.1) Knowledge Function Formulation for a 1-Dimensional System: 45
3.1.2) Formulation of Knowledge Function for Multi Dimensional Systems 50
3.1.3) Equivalence between 0th order APKFM and 0th order TSK fuzzy model 52
3.2 Parameter Estimation of Knowledge function 52
3.2.1) Gradient-based Optimization 53
3.2.2) Derivative-free Optimization 55
3.2.3) Transformation-based Estimation of nonlinear parameters 57
3.3 Formulation of 1st order APKFM 60
3.3.1) Conversion of 1st order TSK fuzzy model to 1st order APKFM 62
3.3.2) Regularized Parameter Estimation of 1st order APKFM 65

Chapter 4: Results and Discussion 69

4.1 2-dimensional Nonlinear Function Approximation 71
4.1.1) Comparison of Prediction Accuracy 74
4.1.2) Comparison of Interpretability 78
4.2 Cost Estimation Models for the Electricity Distribution Networks 82
4.2.1) Construction of Fuzzy Models 83
4.3.2) Comparison of Fuzzy Models 84
4.3 Conclusions 89

Appendix: Fuzzy Rule Based Systems 91

1. Fuzzy Logic 91
1.1) Fuzzy sets 91
1.2) Fuzzy Rule-based Models 93
2 Classification of Fuzzy Models 93
2.1) Mamdami Fuzzy Models 94
2.2) Takagi-Sugeno-Kang (TSK) Fuzzy Models 99
2.3) RBFNs vs. Singleton/0th order TSK Models 102

References 104

List of Figures

FIGURE 1-1 *PARALLEL* CONSTRUCTION OF SEMI-PHYSICAL MODELS. IN THIS CONSTRUCTION THE EXISTING PHYSICAL MODEL IS REFINED BY A DATA-DRIVEN MODEL. B) *SERIES* CONSTRUCTION OF SEMI-PHYSICAL MODEL. IN THIS CASE, THE PHYSICAL INSIGHT ABOUT THE SYSTEM IS USED TO IMPROVE THE STRUCTURE OF THE DATA-DRIVEN MODEL. 5

FIGURE 1-2 PERFORMANCES OF TWO DATA DRIVEN MODELS THAT TRY TO SIMULATE A 3RD ORDER POLYNOMIAL. THE TRAINING DATA, AS SHOWN BY CIRCULAR DOTS, IS CORRUPTED WITH A RANDOM NOISE. THE GRAPH ON THE LEFT SHOWS THE PERFORMANCE OF A LINEAR MODEL WITH 2 PARAMETERS, WHILE ONE ON THE RIGHT CORRESPONDS TO A MORE COMPLEX NEURAL NETWORK MODEL WITH 14 PARAMETERS. 7

FIGURE 1-3 SCHEMATIC OF A GENERAL PROCESS WHOSE TRUE OUTPUT, $\boldsymbol{yT}$, IS CORRUPTED WITH A RANDOM NOISE, $\boldsymbol{e}$. AN OPTIMAL MODEL IS IDENTIFIED BY MINIMIZING THE SQUARED DIFFERENCE OF THE MEASURABLE OUTPUT, $\boldsymbol{y}$, WITH THE MODEL OUTPUT, $\boldsymbol{y}$ 8

FIGURE 1-4 THE VARIATION OF BIAS ERROR WITH INCREASING MODEL COMPLEXITY. THIS CURVE IS DRAWN WITH THE ASSUMPTION THAT IRRESPECTIVE OF THE COMPLEXITY VALUE, WE HAVE ADEQUATE DATA TO IDENTIFY THE OPTIMAL PARAMETER VALUES. 10

FIGURE 1-5 DEPENDENCE OF VARIANCE ERROR ON MODEL COMPLEXITY/NUMBER OF PARAMETERS. THE ABOVE CURVE IS DRAWN UNDER THE ASSUMPTION THAT NUMBER OF MODEL PARAMETERS IS SIGNIFICANTLY LESS THAN THE NUMBER OF TRAINING DATA POINTS. 11

FIGURE 1-6 GRAPHICAL REPRESENTATION OF THE BIAS VARIANCE TRADEOFF FOR TWO CASES. A) A MODEL WITH STEEPER VARIANCE ERROR LINE, WHICH RESULTS FROM A POOR QUALITY TRAINING DATA. B) A MODEL WITH RELATIVELY FLATTER VARIANCE ERROR LINE. 12

FIGURE 2-1 GRAPHICAL REPRESENTATION OF DATA-DRIVEN MODELS HAVING THE BASIS FUNCTION FORMULATION. THE HIDDEN LAYER OF THIS NETWORKED STRUCTURE CONSISTS OF NONLINEAR BASIS FUNCTIONS. THE OUTPUT IS CALCULATED AS THE WEIGHTED SUM OF THESE BASIS FUNCTIONS. 18

FIGURE 2-2 A SCHEMATIC SHOWING THE FUNCTIONING OF AN NEURON/BASIS FUNCTION. THE CONSTRUCTION MECHANISM COMBINES THE NONLINEAR PARAMETERS WITH THE INPUT VECTOR AND FEEDS IT TO THE BASIS FUNCTION. 18

FIGURE 2-3 A BIVARIATE BASIS FUNCTION RESULTING FROM THE RIDGE CONSTRUCTION. THE NONLINEAR PARAMETER $\boldsymbol{\theta}(\boldsymbol{nl})$ VECTOR IS TAKEN AS [1 1] AND HYPERBOLIC TANGENT IS CHOSEN AS THE ACTIVATION FUNCTION. THE DIRECTION OF INCREASE OF BASIS FUNCTION, AS SHOWN AS SHOWN IN THE CONTOUR MAP, COINCIDES WITH THE NONLINEAR PARAMETER VECTOR. 21

FIGURE 2-4 A BIVARIATE BASIS FUNCTION RESULTING FROM THE RADIAL CONSTRUCTION. THE CENTER OF THE RADIAL BASIS FUNCTION IS TAKEN AS [0.6 0.4] AND GAUSSIAN FUNCTION IS CHOSEN AS THE ACTIVATION FUNCTION. THE TOP SURFACE IS RESULTS FROM TAKING THE DISTANCE MEASURE AS THE EUCLIDEAN NORM. IN THE BOTTOM SURFACE, THE DISTANCE MEASURE IS MODIFIED ACCORDING TO A GIVEN NORM MATRIX ℕ. 24

FIGURE 2-5 DEMONSTRATION OF THE TENSOR PRODUCT CONSTRUCTION USED IN NEURO-FUZZY MODELS. THE CONICAL TRIANGULAR BASIS FUNCTIONS ARE OBTAINED FROM THE UNIVARIATE TRIANGULAR FUNCTIONS. THE CONTOUR MAPS OF MULTIVARIATE BASIS FUNCTIONS SHOWS THAT THEY ARE LOCAL IN NATURE. 27

FIGURE 2-6 GRAPHICAL DEPICTION OF THE EARLY STOPPING REGULARIZATION TECHNIQUE. IN THE TOP PLOT, WE CAN OBSERVE THE TYPICAL VARIATION IN MEAN SQUARED ERROR ON VALIDATION AND TRAINING DATASETS WITH RESPECT TO THE NUMBER OF ITERATIONS. THE ITERATIONS ARE TERMINATED WHEN THE VALIDATION ERROR REACHES THE MINIMUM (REFER THE DASHED ARROW). 35

FIGURE 2-7 CONTOUR MAPS INDICATING THE EFFECT OF RIDGE PARAMETER $\boldsymbol{\alpha}$ ON THE HESSIAN OF A QUADRATIC LOSS FUNCTION. A HIGHER VALUE OF $\boldsymbol{\alpha}$ IMPROVES THE CONDITIONING OF THE HESSIAN THEREBY REDUCING THE UNCERTAINTY IN ESTIMATED PARAMETER VALUES. 39

FIGURE 3-1 ILLUSTRATION OF THE KNOWLEDGE FUNCTION FOR A ONE DIMENSIONAL SYSTEM. THE KNOWLEDGE FUNCTION IS A GAUSSIAN FUNCTION CENTERED AT THE FAVORABLE RULE $\boldsymbol{R1}$. 48

FIGURE 3-2 ILLUSTRATION OF THE KNOWLEDGE FUNCTION WITH TWO FAVORABLE RULES ($\boldsymbol{R1}$ AND $\boldsymbol{R3}$). THE KNOWLEDGE FUNCTION IS GENERATED BY TAKING THE WEIGHTED AVERAGE OF TWO GAUSSIAN FUNCTION CENTERED AT THE TWO FAVORABLE RULES. 50

FIGURE 3-3 (A) GRAPHICAL REPRESENTATION OF A 1ST ORDER TSK MODEL FOR THE APPROXIMATION OF A 1-DIMENSIONAL SYSTEM. THE LOCAL MODEL OF 2ND RULE HAS ONLY 1 LINEAR PARAMETER I.E. $\boldsymbol{\theta 12}$. (B) GRAPHICAL REPRESENTATION OF THE 1ST ORDER APKFM FOR THE SAME SYSTEM. THE LOCAL MODEL OF 2ND RULE HAS 2 LINEAR PARAMETERS VIZ. $\boldsymbol{\theta 1-2}$ & $\boldsymbol{\theta 1+2}$. 63

FIGURE 3-4 A 2-D FUNCTION'S INPUT SPACE WITH 9 RULES AND 24 LOCAL LINEAR PARAMETERS. THE BLACK DOTS REPRESENT THE LOCATIONS OF RULE CENTERS. THE ARROWS POINT TOWARDS THE NEAREST NEIGHBORS. SINCE THE CENTERMOST RULE (RULE #5) HAS TWO NEAREST NEIGHBORS IN EACH DIRECTION, IT HAS FOUR LINEAR PARAMETERS VIZ. $\boldsymbol{\theta 1-5,\ \theta 1+5,\ \theta 2-5,\ \theta 2+5}$. 64

FIGURE 3-5 ILLUSTRATING THE EFFECT OF THE LOCATION OF AN INPUT ON THE CHOICE OF PARAMETERS USED FOR DESCRIBING A LOCAL LINEAR MODEL. 64

FIGURE 3-6 ILLUSTRATING HOW THE LINEAR PARAMETERS ON A 1ST ORDER APKFM CAN BE ESTIMATED FROM AN ALREADY IDENTIFIED 0TH ORDER APKFM. 67

FIGURE 4-1 THE TOP VIEW OF 2-DIMENSIONAL INPUT SPACE THAT HAS BEEN PARTITIONED BY 9 OVERLAPPING FUZZY SETS. THE SHADED SUBSPACES CORRESPOND TO THE FAVORABLE RULES. 73

FIGURE 4-2 PLOTS OF MEAN RMSE VS. NOISE LEVEL FOR THE FOUR TECHNIQUES OF TSK FUZZY MODEL IDENTIFICATION. THE FUZZY MODELS WERE TRAINED WITH THE DATASETS WITH 4 DIFFERENT SIZES VIZ. 40, 60, 80 AND 100. 76

FIGURE 4-3 VARIATION OF THE WIDTH OF 95% CI OF MEAN RMSE WITH THE INCREASING NOISE LEVELS FOR THE THREE COMPETING REGULARIZATION TECHNIQUES. THE SIZE OF TRAINING DATASET IS FIXED AT 40. 77

FIGURE 4-4 A) THE OUTPUT SURFACE CORRESPONDING TO THE NONLINEAR FUNCTION GIVEN IN EQUATION 4.1. B-F) THE OUTPUT SURFACE GENERATED BY FUZZY MODELS IDENTIFED USING *GLOBAL LSE* AND TRAINING DATASETS WITH VARYING NOISE LEVELS. THE ENCIRCLED REGIONS REPRESENT SOME OF THE PLACES WHERE MODEL OUTPUT IS INCOSISTENT WITH THE DOMAIN KNOWLEDGE. 79

FIGURE 4-5 A) THE OUTPUT SURFACE CORRESPONDING TO THE NONLINEAR FUNCTION GIVEN IN EQUATION 4.1. B-F) THE OUTPUT SURFACE GENERATED BY FUZZY MODELS IDENTIFED USING *RIDGE REGRESSION* AND TRAINING DATASETS WITH VARYING NOISE LEVELS. THE ENCIRCLED REGIONS REPRESENT PLACES WHERE MODEL OUTPUT IS INCOSISTENT WITH THE DOMAIN KNOWLEDGE. 80

FIGURE 4-6 A) THE OUTPUT SURFACE CORRESPONDING TO THE NONLINEAR FUNCTION GIVEN IN EQUATION 4.1. B-F) THE OUTPUT SURFACE GENERATED BY FUZZY MODELS IDENTIFED USING *LOCAL LSE* AND TRAINING DATASETS WITH VARYING NOISE LEVELS. 81

FIGURE 4-7 A) THE OUTPUT SURFACE CORRESPONDING TO THE NONLINEAR FUNCTION GIVEN IN EQUATION 4.1. B-F) THE OUTPUT SURFACE GENERATED BY *APKFM* AND TRAINING DATASETS WITH VARYING NOISE LEVELS. 82

FIGURE 4-8 PLOTS OF PREDICTED MAINTENANCE COST VS. ACTUAL MAINTENANCE COST RESULTED FROM FOUR DIFFERENT FUZZY MODELING TECHNIQUES. THE ROOT MEAN SQUARED ERROR (RMSE) DENOTES THE ERROR IN PREDICTION..85

FIGURE 4-9 OUTPUT SURFACES GENERATED BY DIFFERENT MODELS AFTER TRAINING WITH 832 DATA SAMPLES. THE INPUTS $\boldsymbol{x2}$ AND $\boldsymbol{x3}$ ARE FIXED AT A VALUE OF 0.5 WHILE $\boldsymbol{x1}$ AND $\boldsymbol{x4}$ WERE VARIED TO GENERATE THESE SURFACES. THE ENCIRCLED REGIONS SIGNIFY THAT THE OUTPUT IS NOT CONSISTENT WITH THE DOMAIN KNOWLEDGE.....86

FIGURE 4-10 OUTPUT SURFACES GENERATED BY DIFFERENT MODELS AFTER TRAINING WITH 832 DATA SAMPLES. THE INPUTS $\boldsymbol{x1}$ AND $\boldsymbol{x4}$ ARE FIXED AT 0.75 AND 0.5 RESPECTIVELY WHILE $\boldsymbol{x1}$ AND $\boldsymbol{x4}$ WERE VARIED TO GENERATE THESE SURFACES. THE ENCIRCLED REGIONS SIGNIFY THAT THE OUTPUT IS NOT CONSISTENT WITH THE DOMAIN KNOWLEDGE. 87

FIGURE 4-11 THE DISTRIBUTION OF TRAINING DATA-POINTS IN THE INPUT SPACE. THE ENCIRCLED REGIONS HAVE A SPARSE DISTRIBUTION OF POINTS. THESE REGIONS CORRESPONDS TO THOSE INPUT SUBSPACES WHERE THE MODEL OUTPUT MAY BECOME INCONSISTENT WITH THE DOMAIN KNOWLEDGE...88

FIGURE 1 A) A TRIANGULAR MEMBERSHIP FUNCTION DEFINED USING FOUR PARAMETERS $\boldsymbol{m1}, \boldsymbol{m2}, \boldsymbol{c1}$ AND $\boldsymbol{c2}$. B) A GAUSSIAN MEMBERSHIP FUNCTION DEFINED USING TWO PARAMETERS $\boldsymbol{c}$ AND $\boldsymbol{\sigma}$. 92

FIGURE 2 GRID PARTITIONED INPUT SPACE OF A 2-D SYSTEM. BOTH INPUTS $\boldsymbol{x1}$ AND $\boldsymbol{x2}$ HAVE 3 FUZZY SETS WITH GAUSSIAN MEMBERSHIP FUNCTIONS. THE INPUT SPACE GETS

PARTITIONED INTO $\mathbf{3 \times 3 = 9}$ GRIDS AS WHOSE LOCATIONS ARE SHOWN BY OVERLAPPING ELLIPSES..95

FIGURE 3 FUZZIFICATION OF INPUTS TEMPERATURE AND HUMIDITY USING TRIANGULAR MEMBERSHIP FUNCTIONS. A CRISP VALUE OF 80°F IS WARM BY A DEGREE OF 0.68 AND COLD BY 0.32. SIMILARLY, THE CRISP HUMIDITY VALUE OF 2 MG/CC IS LOW BY A DEGREE OF 0.64 AND HIGH BY 0.36. 97

FIGURE 4 GRAPHICAL ILLUSTRATION OF 3 IMPORTANT STEPS OF MAMDAMI INFERENCE. 1) IMPLICATION: TRUNCATING OUTPUT FUZZY SETS AT STRENGTH VALUES OF FIRED RULES 2) AGGREGATION: COMBINING TRUNCATED FUZZY SETS TO FORM A SINGLE AGGREGATED FUZZY SET AND 3) DEFUZZIFICATION: OBTAINING THE CRISP OUTPUT VALUE BY COMPUTING THE CENTROID OF AGGREGATED FUZZY SET. 99

FIGURE 5 A GRAPHICAL REPRESENTATION OF A 1ST ORDER TSK FUZZY MODEL FOR A ONE DIMENSIONAL SYSTEM. THE INPUT SPACE IS DIVIDED BY 3 FUZZY SETS RESULTING IN 3 RULES. THE RULE CONSEQUENTS ARE LINEAR EQUATIONS DEFINED AT THE CENTER OF CORRESPONDING RULES. 101

List of Tables

TABLE 4-1 PROPERTIES OF 1ST ORDER TSK MODEL AND APKFM THAT WERE USED FOR THE APPROXIMATION OF 2-DIMENSIONAL NONLINEAR FUNCTION. 72

TABLE 4-2 COMPARISON OF ROOT MEAN SQUARE ERRORS (RMSES) OF DIFFERENT MODELS TRAINED ON DATASETS OF DIFFERENT SIZES (40, 60, 80, 100) AND NOISE LEVELS ($\sigma L = 0.0$, $0.067R$, $0.133R$, $0.20R$, $0.267R$). THE HIGHLIGHTED ROWS INDICATE THE CASES WHEN APKFM HAD SIGNIFICANTLY LOWER RMSE COMPARED TO THE OTHER THREE TECHNIQUES. 77

TABLE 4-3 DESCRIPTION OF DIFFERENT INPUTS CONSIDERED IN THE COST ESTIMATION MODEL...83

TABLE 4-4 PROPERTIES OF 1ST ORDER TSK MODEL AND APKFM THAT WERE USED FOR BUILDING THE COST ESTIMATION MODELS. 84

List of Symbols

$\boldsymbol{X}$: Input Vector $(x_1\ x_2\ \dots \dots .\ x_n)$

$\boldsymbol{n}$: Dimension of input vector

$\boldsymbol{f}(\boldsymbol{X})$: Scalar output of a static nonlinear system

$\hat{\boldsymbol{f}}(\boldsymbol{X})$: Scalar output of a data-driven model that tries to mimic the nonlinear system $f(.)$

$\boldsymbol{\theta}_{\boldsymbol{k}}^{(\boldsymbol{l})}$: Linear parameters of the data-driven models based on basis function formulation. The subscript k represents the parameters associated with k^{th} basis function

$\boldsymbol{\theta}_{\boldsymbol{k}}^{(\boldsymbol{nl})}$: Nonlinear parameters of the data-driven models based on basis function formulation. The subscript k represents the parameters associated with k^{th} basis function

$\boldsymbol{\phi}_{\boldsymbol{k}}(.)$: k^{th} basis function

$\boldsymbol{e}$: A vector of modeling errors, where i^{th} element represents $e(i) = \left(f\left(X^i\right) - \hat{f}\left(X^i\right)\right)$

$\boldsymbol{\alpha}$: Regularization/Ridge parameter

$\boldsymbol{M}$: Number of training data points.

$\boldsymbol{\mathcal{B}}$: A set containing indices of all the rules present in the rule-base.

$\boldsymbol{R}$: Total number of rules in the rule-base/ no. of elements in set $\mathcal{B}$.

$\boldsymbol{\mathcal{F}}$: A subset of $\mathcal{B}$ containing of indices of locally favorable rules.

$\boldsymbol{J}$: Total number of locally favorable rules/ no. of elements in set $\mathcal{F}$.

$\boldsymbol{C}^{\boldsymbol{\mathcal{B}}(\boldsymbol{r})}$: Center of r^{th} rule of the rule-base $\left(c_1^{\mathcal{B}(r)}\ c_2^{\mathcal{B}(r)}\ \dots \dots .\ c_n^{\mathcal{B}(r)}\right)$

$\boldsymbol{\theta}_{\boldsymbol{o}}^{\boldsymbol{r}}$: Constant term of the r^{th} rule's local linear model.

$\boldsymbol{\theta}_{\boldsymbol{i}}^{\boldsymbol{r}}$: Coefficients of the i^{th} linear term of the r^{th} rule's local linear model

$\boldsymbol{\theta}_{\boldsymbol{i-}}^{\boldsymbol{r}}/\boldsymbol{\theta}_{\boldsymbol{i+}}^{\boldsymbol{r}}$: The two possible values of θ_i^r depending on the location of input with respect to the rule center.

$\boldsymbol{P}^{\boldsymbol{r}}(\boldsymbol{X})$: The local model corresponding to the r^{th} rule of a TSK fuzzy model

$\boldsymbol{\varkappa}(.)$: Knowledge function

$\boldsymbol{a}^{\boldsymbol{j}}$: A parameter of the Gaussian function $\left(\psi_r^j\right)$ defined at j^{th} locally favorable rule

$\boldsymbol{\sigma}_{\boldsymbol{i}}^{\boldsymbol{j}}$: Another parameter of the Gaussian function $\left(\psi_r^j\right)$ defined at j^{th} locally favorable rule

$\boldsymbol{\delta}^{(\boldsymbol{nl})}$: The nonlinear parameters of the knowledge function. It includes $a^j$and σ_i^j

$\boldsymbol{\gamma}_{\boldsymbol{r}}^{\boldsymbol{j}}$: Degree of closeness of r^{th} rule center to the center of the j^{th} favorable rule

$\boldsymbol{\theta}_{\boldsymbol{prior}}$: Prior knowledge based estimates of model parameters.

List of Acronyms

APKFM: A-Priori Knowledge-based Fuzzy Model

ANN: Artificial Neural Network

LSE: Least Square Estimate

MISO: Multiple Input Single Output

RBFN: Radial Basis Function Network

RMSE: Root Mean Squared Error

TSK: Takagi-Sugeno-Kang

Acknowledgements

First I would like to thank my thesis advisor Mirna Urquidi-Macdonald. I could not have hoped to have a better mentor for my PhD, and without her unwavering guidance and support this work would not be possible. I also thank the department of Engineering Science and Mechanics for supporting my graduate studies for three years.

I would like to thank my old roommates Amit and Vishisht for making my stay in State College enjoyable. I would also like to thank my old friends Vishal, Manish and Ronak for always being in constant touch with me despite being far away. My thanks goes to my other buddies Anuj, Varun, Doctor, Bennie, Raman, Baba, Puneet, Abhijeet, Harshad, Surinder and their girl friends/spouses for the good time that we had together. I thank my sisters Rinku and Vicky for their love and concern. I especially like to thank my parents whom I owe my present and future. Without their support I could not have achieved anything in my life. I dedicate this work to my parents, my family and friends from whom I have drawn my strength during these years of doctoral project.

Finally I would like to thank the person who has been with me in thick and thin, who always believed in me, and filled my life with love and cheerful happiness. My wife, Neha without her I cannot live.

Chapter 1: Introduction

Over the years researchers have shown great desire to build mathematical models that can simulate real world systems. In such models the relationships between different system variables are described using mathematical equations. A model, if correctly built, helps in gaining better insight about the behavior of a system. The improved understanding of a system can be useful for the prediction of its behavior in variety of situations of interest. For instance, quite often we are interested in knowing the conditions under which a system might fail. If we have a mathematical model that can correctly simulate this system, we may foresee the occurrence of a failure and avoid it by taking some corrective steps.

A large integrated system, built from several mathematical models, can be commonly found in engineering problems. Each individual model describes the behavior of a specific component of the parent system. Logically, the reliability of this integrated system should be profoundly affected by the quality of all models, which usually have different complexities. For example, modeling the flow of current through a metallic wire of fixed dimensions is a straight forward task that can be accurately done using the *Ohm's law*. On the other hand, modeling the turbulence fluid flow through a circular pipe is an exceedingly challenging problem. Nevertheless, we cannot afford an inferior model of a complex subsystem as it may have serious repercussions on the performance of the overall system. This is the reason why there is a never ending demand for the advanced modeling and simulation techniques.

In this thesis we analysis a unique type of mathematical models called *Neuro-Fuzzy* models. The neuro-fuzzy models fall in the category of *data-driven* models. Before, delving further into the details of data-drive models, it is important to understand the differences between them and other types of models. The next section, deals with system modeling in general and briefly describes three important categories of mathematical models [1]. In the later sections the focus shifts only to the data-driven models and issues related to them. Later on, the problem addressed in this thesis is discussed briefly.

1.1 System Modeling

There are no standardized techniques for modeling real world systems. It is possible to have two disparate models that describe a particular system equally well. A more likely scenario is that we have two different models that correctly describe a system under different set of conditions. The choice of modeling technique for a system depends on various factors such as its complexity, physical insight, performance requirements from the model etc. In general, mathematical models can be categorized into three classes viz. *Physical Models, Data-driven Models* and *Semi-Physical models*.

1.1.1) Physical Models

These models are based on the thorough understanding of the behavior of a system. An expert analyzes a system, understand the physical principles behind its observed behavior and expresses his knowledge mathematically. The goal of a physical model is to connect different system variables using mathematical equations, which are purely derived from the knowledge of underlying physical phenomena. A simple example of such a model is the *Gravitational* model that determines the force of attraction caused by large celestial bodies. This model was proposed by Sir Isaac Newton in the 17th century who used it to explain the planetary motion in our solar system. One characteristics of such models is that they are highly transparent i.e. there exists no ambiguity about how an output is derived from a physical model. Although the thought process that goes into building a physical model can be quite complex, the final model usually have an elegant mathematical formulation, which provides it the desired transparency. Another desirable characteristics that these models possess is their superior generalization capability. For instance, the gravitational laws are as valid on any other planet as they are on earth. For these reasons, physical models are the most desirable type of mathematical models.

Unfortunately building a physical model can prove to be a formidable task in certain cases. Sometimes a system may have a very complex behavior resulting from an intricate combination of several physical processes. In such cases, developing a physical model becomes extremely difficult and prohibitively time consuming. Here, we would like to cite an example of such a system that partly motivated the work done in this research. *Microbiologically Influenced Corrosion* (MIC) is one of the major problems that threaten the integrity of crude oil pipelines [2, 3]. Therefore, it is indispensable, both economically and environmentally, to have a tool that can help us identify sections of a pipeline network that are prone to MIC related damage. However, MIC is affected by various factors such as the crude oil's chemistry, the prevailing hydrodynamic conditions, pipeline's surface characteristics, biological interactions between different types of microorganism and the electrochemistry of the corrosion process. All these factors interact in a complex manner to yield conditions that are conducive for MIC. Clearly, building a physical model that can correctly describe the MIC problem is not easy. Therefore, for such complex systems, another modeling scheme known as data-driven modeling has been proven to be quite effective.

1.1.2) Data-driven Models

These models are fundamentally different from the physical models in their construction. For building a data-driven model of a system we are not needed to understand the physics behind its behavior. Instead, we rely on a historically generated dataset to describe its input to output mapping [4]. This type of modeling borrows heavily from the fields artificial intelligence, statistics, expert systems, approximation theory etc. The data-driven models work by first generating a flexible parameterized model, wherein the parameters do not bear any physical significance. However, these parameters make a model flexible and allow it to get molded according to the available data. The process of molding is known as *learning* and the dataset used for this purpose is called a *training* dataset. After training, an ideal data-driven model is expected to have learned the underlying behavior of a system no matter how complex it is.

The data-driven models are primarily used for three types of problems namely *Prediction*, *Classification* and *Clustering*. Prediction, also referred as regression or function approximation, aims at predicting the behavior of a system in never-seen situations. If a model has adequately learned from the training data, it should be able to correctly predict the output for even those input conditions for which it wasn't trained. The second class of problem, known as classification, tries to group the inputs into different classes. The training data is available in the form of inputs with known class labels. Post training, a model is expected to be able to correctly classify a never-seen input to its correct class. The last type of problem is known as clustering, which is similar to classification in the sense that it tries to find naturally occurring groups in the data. The difference between clustering and classification is that clustering works in an *unsupervised* manner, i.e. we do not have information about the class labels of inputs in the training data. Hence, the groups are identified by utilizing the information about the spatial distribution of data in the input space. It should be noted that the learning process in the first two problems is called *supervised* learning because the training data has output information for every input vector. It should be mentioned here that, in this thesis, we focus only on the data-driven models pertaining to the prediction problem.

1.1.3) Semi-physical Models

Sometimes we may understand a system quite well but not well enough that a fully physically parameterized model can be developed. In such cases, one option is to completely discard the domain knowledge and build a data-driven model hoping that it will learn the system behavior from the available input-output data. This may not be the best option for two reasons, 1) the quality of training data becomes critical and 2) a more transparent model could have been made by taking into account our knowledge despite it being insufficient. By incorporating the physical insight about the system, we may come with a superior model structure. Such models lie somewhere in between the two extremes i.e. the physical models and the-data driven models and are called *semi-physical* models.

There are two general approaches of building a semi-physical models [5]. In the first approach we refine an existing physical model with a data-driven model. Often a prior physical model may be insufficient to explain the correct system behavior due to its rigid structure. In such cases, physical models can be improved by supplementing it with another data-driven model that can learn the unexplained part of the system. Such construction of semi-physical models is called the *parallel construction* and is shown in figure 1.1a. In this construction the prior physical model output is augmented by the output of a data-driven model i.e.

$$\hat{f}(X) = \hat{f}_{phys}(X) + \hat{f}_{data}(X) \tag{1.1}$$

Where $\hat{f}(X)$ is the overall output of the semi-physical model and $\hat{f}_{phys}(X)$ and $\hat{f}_{data}(X)$ are the outputs of the physical and data-driven model respectively. Therefore, the modeling error that should be minimized during the training of the data-driven model is given by equation 1.2.

$$error = f(X) - \hat{f}(X) = [f(X) - \hat{f}_{phys}(X)] - \hat{f}_{data}(X) \tag{1.2}$$

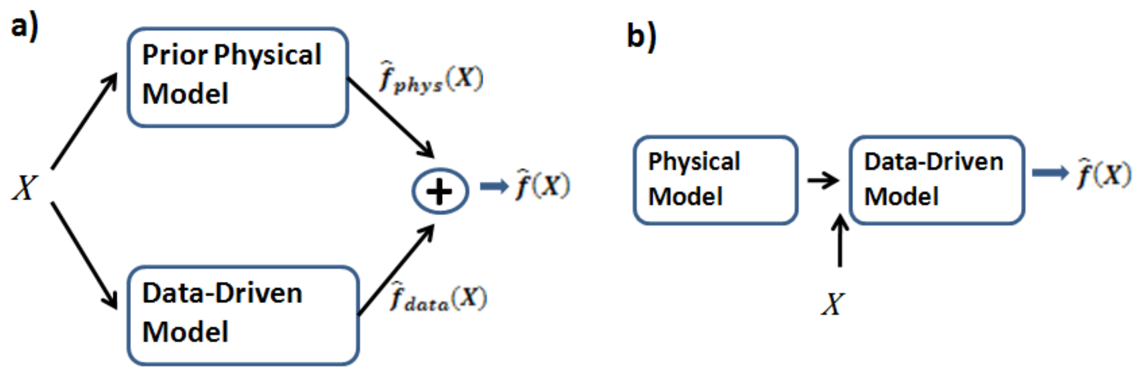

Figure 1-1 *Parallel* construction of semi-physical models. In this construction the existing physical model is refined by a data-driven model. b) ***Series* construction of semi-physical model. In this case, the physical insight about the system is used to improve the structure of the data-driven model.**

The other type of semi-physical models follows a *series construction* as shown in figure 1.1b. This approach is used when the physical model is not entirely complete. Sometimes, the system that we intend to model may consists of many subsystems of varying complexities. We may have accurate physical models for some of the simpler subsystems. In this scenario, the data-driven model is used only for the unmodeled components of the parent system. Another way in which the series construction works is by utilizing the physical insight about the system to preprocess the data. A trivial example that elaborates such models is explained as follows. Suppose we intend to model the heat dissipation from an electric heater. Clearly, it would depend on several factors like room temperature, electric current (I), aero-dynamic conditions etc. Using a set of historically obtained data, we try to identify the mapping between the inputs and the heat dissipation. It is known that heat dissipation is directly proportional to the power given to the heating element which is equal to $\mathrm{I}^2\mathrm{R}$. Therefore, instead of using the current value (I) as an input if we use its squared value (I^2), we can expect a significant improvement in the model.

Although the fundamentals are quite different, a common characteristic amongst all types of mathematical models is that they usually have certain unknown parameters. In physical models, usually only a few parameters are present which can be accurately determined by doing some control experiments. Sometimes it is even possible to make an educated guess for a physical model's parameters as they have some physical meaning. The data-driven models comparatively have very high number of parameters, hence their experimental determination is not possible. Therefore, in such models the parameter values are estimated from historically available input-output information. Clearly, a data-drive model can't have infinitely many parameters as it would require an infinitely sized historical dataset for parameter estimation. One way to find the optimal number of parameter that data-driven model should have is by optimizing its *complexity*. Complexity of parameterized models is an extremely important concept and should be understood by any person who is involved in the development of such models. In section 1.2, we define the concept of complexity in context of data-driven models and elucidate why is it so important to identify a model with optimal complexity. An important principle, known as the *bias-variance tradeoff*, is presented in section 1.2.2. A thorough understanding of this tradeoff is necessary as it has a profound influence on the complexity of a model.

1.2 Complexity of Data-driven Models

Complexity of a system is a very subjective issue. A simple model built by an engineer may seem extremely complex to a chemist simply because of his lack of knowledge in that subject area. Therefore, defining complexity based on a person's knowledge is not appropriate because in that scenario a system's complexity would change from one person to another. A better way to characterize complexity is by defining it in terms the number of parameters present in a model. As a result, a seemingly complex mathematical model with fixed number of parameters will have same degree of complexity for a mathematician as it will have for a chemist. Going by this definition, a model becomes more complex if the parameters it possesses increase. Sometimes, a model's complexity and flexibility are used synonymously because higher number of parameters in a model increases its degree of freedom and renders it more flexible.

Prior to building a data-driven model we rarely have any information about the optimal complexity that it should have. The reason why we need a model with optimal complexity is explained as follows. If we have a complex nonlinear system, a relatively simple model (with only few parameters) is not a good choice because it would not be flexible enough to capture the underlying nonlinear process. On the other hand, if we choose a model with many parameters then it would be so flexible that it would mimic even the noise present in the data. This concept would become clearer if we look at an example where we try to approximate a 3rd order polynomial with two data-driven models with different complexities. Figure 1.2 compares the output generated by these two models after training. The training data is generated from the polynomial and corrupted with a random noise. One of the models is linear with only 2 unknown parameters, whereas the other is a more flexible *Artificial Neural Network* (ANN) model with 14 parameters. The dashed and solid lines, in figure 1.2, represents the actual 3rd order polynomial and the output of trained data-driven models respectively.

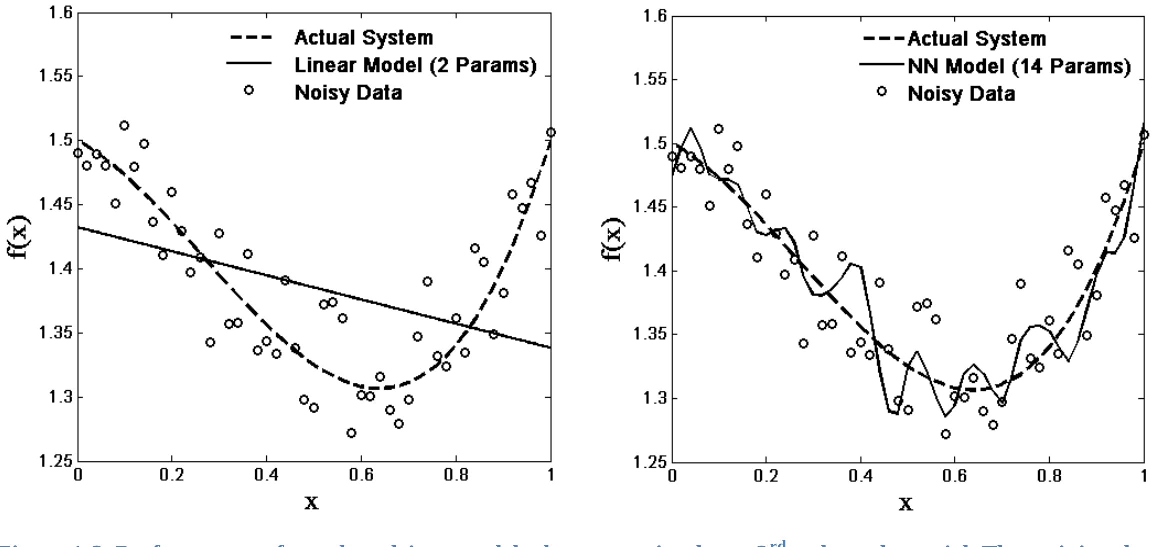

Figure 1-2 Performances of two data driven models that try to simulate a 3rd order polynomial. The training data, as shown by circular dots, is corrupted with a random noise. The graph on the left shows the performance of a linear model with 2 parameters, while one on the right corresponds to a more complex neural network model with 14 parameters.

Clearly, the linear model is too rigid to learn a higher order polynomial. Even if the noise is removed from the training dataset, the performance of this model will not change appreciably. On the contrary, the ANN model with 14 parameters is so flexible that it gets over-trained on the noisy training data. Apparently, both the models have suboptimal complexities, which results in poor approximation of the 3rd order polynomial . This the reason why, it is so important to determine the optimal complexity of a model from the available data. Determining the right complexity of a model is not a straightforward task as we usually don't have a bench mark that can allows us to conclude with 100% certainty that the optimal model complexity has been achieved. Therefore, a modeler, at best, can strive to devise strategies to get closer to the optimal complexity of a model.

1.2.1) Decomposition of Modeling Error

It is apparent from the preceding discussion that if the complexity of a model is left unnoticed it is bound to have a poor generalization capability. It can also be noted that the type of modeling error shown by an under-parameterized model is fundamentally different from that of an over-parameterized model. The reason being, *modeling error* consists of two types errors viz. *bias error* and *variance error* [6]. In the following discussion, it is shown, from a statistical viewpoint, that how the modeling error can be decomposed into its components. Consider a process shown in figure 1.3.

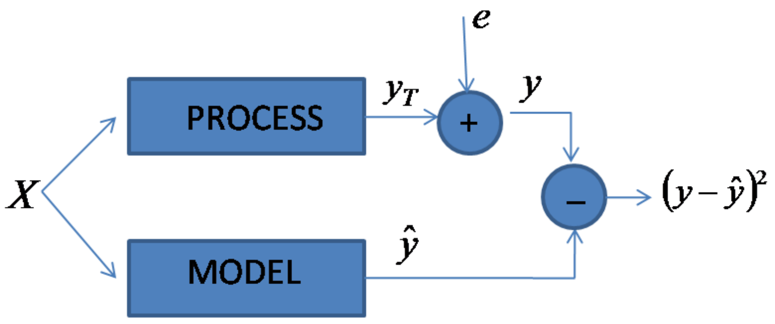

Figure 1-3 Schematic of a general process whose true output, y_T, is corrupted with a random noise, e. An optimal model is identified by minimizing the squared difference of the measurable output, y, with the model output, $\hat{y}$.

Let y_T be the true output of the process and y be the measurable output that results from the mixing of the true output with a random noise e. Also, let $\hat{y}$ be the output of a parameterized data-driven model constructed for the same system. Usually the parameters of this model are estimated by the minimization of an error function defined as the sum of squared errors. In probabilistic framework, this error function can be viewed as the expectation of the squared error i.e. $E[(y-\hat{y})^2]$. Equation 1.3 shows that how this expectation can be decomposed.

$$E[(y-\hat{y})^2] = E[(y_T+e-\hat{y})^2] = E[(y_T-\hat{y})^2] + E[e^2] - 2E[(y_T-\hat{y})]E[e] \tag{1.3}$$

The last term of the above equation results from the fact that the random noise n is correlated with both the true process output and the model output. Since the expected value of the random noise ($E[e]$) is zero, the last term vanishes. Therefore, the error function can be viewed as a combination of modeling error and the noise variance as shown in the equation below.

$$E[(y-\hat{y})^2] = E[(y_T-\hat{y})^2] + E[e^2] \tag{1.4}$$

Clearly, the noise variance can't be removed and in the best possible scenario when the model output is the same as true output, i.e. $\hat{y} = y_T$, the error function simply reduces to this noise variance. The modeling error can be further decomposed as follows:

$$\begin{aligned} E[(y_T-\hat{y})^2] &= E[(y_{T-}E[\hat{y}] - (\hat{y}-E[\hat{y}]))^2] \\ &= E[(y_{T-}E[\hat{y}])^2 + (\hat{y}-E[\hat{y}])^2 - 2(y_{T-}E[\hat{y}])(\hat{y}-E[\hat{y}])] \\ &= E[(y_{T-}E[\hat{y}])^2] + E[(\hat{y}-E[\hat{y}])^2] - 2E[(y_{T-}E[\hat{y}])(\hat{y}-E[\hat{y}])] \end{aligned} \tag{1.5}$$

Since the quantity $(y_{T-}E[\hat{y}])$ is deterministic, equation 1.5 can be further simplified as:

$$\begin{aligned} E[(y_T-\hat{y})^2] &= (y_{T-}E[\hat{y}])^2 + E[(\hat{y}-E[\hat{y}])^2] - 2(y_{T-}E[\hat{y}])E[(\hat{y}-E[\hat{y}])] \\ &= (y_{T-}E[\hat{y}])^2 + E[(\hat{y}-E[\hat{y}])^2] - 2(y_{T-}E[\hat{y}])(\cancel{E[\hat{y}]-E[\hat{y}]}) \end{aligned}$$

Hence, we have

$$E[(y_T-\hat{y})^2] = (y_{T-}E[\hat{y}])^2 + E[(\hat{y}-E[\hat{y}])^2] \tag{1.6}$$

The first component of the modeling error is deterministic and is called the bias error and the other component is the variance error. Equation 1.6 can be written linguistically in the following way.

$$(Modeling\ Error)^2 = (Bias\ Error)^2 + Variance\ Error \tag{1.7}$$

In the following discussion, a detailed discussion on these two types of errors is given.

Bias Error: This type of error occurs when a model has lower complexity than what is required. Suppose, we have such a model and also a large noise-free training dataset, which guarantees that the optimal values of the unknown parameters can be obtained. Even after the parameters are set to their optimal values, this model will exhibit significant prediction error. This error originates because of the inability of the model to mimic the underlying process due to its rigid structure. This is exactly the reason why, in figure 1.2, the linear model performed poorly when it tried to approximate a 3rd order polynomial. One noticeable characteristic of bias error is that it does not reduce with the improvement in the quality of training data. Figure 1.4, shows the qualitative dependence of bias error on the complexity of a model.

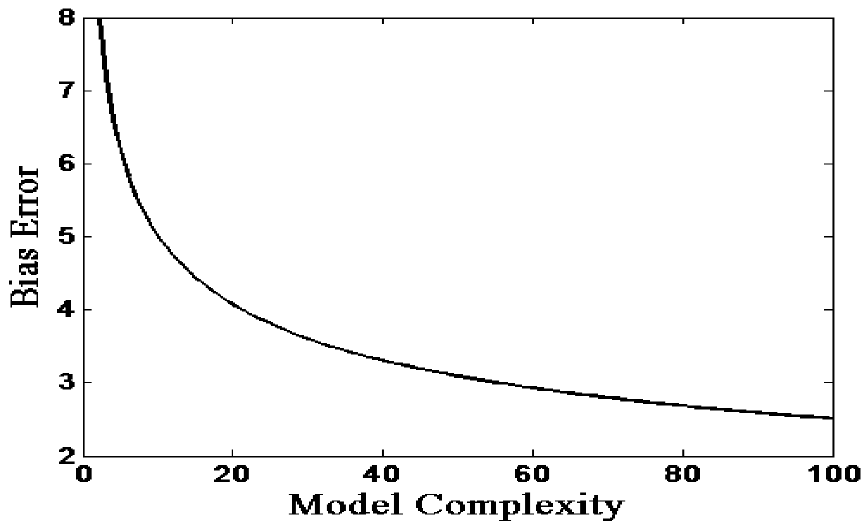

Figure 1-4 The variation of Bias Error with increasing model complexity. This curve is drawn with the assumption that irrespective of the complexity value, we have adequate data to identify the optimal parameter values.

Apparently, the bias error is large for simple models that have low flexibility. As the number of parameters increases the bias error drops sharply. The decrease in bias error becomes sluggish as the model complexity increases further. Interestingly, figure 1.4 suggests that a simple way to reduce the modeling error is to increase complexity incrementally until the desired accuracy is attained. However, the preceding statement is true only under the assumption that we have a large repository of high quality training data. This assumption is hardly true for real world problems where the data is usually limited and corrupted with some form of noise. Because of these restrictions on the historical data, the modeling error may get dominated by the second type of error known as variance error.

Variance Error: The variance error seeps into a model due to the uncertainty in the estimated parameters values. As mention earlier, in practical problems we usually do not have access to sufficient amount of good quality data. Because of this inadequacy of training data we can never be sure if the estimated parameter values are optimal or even near optimal. The deviation of estimated parameter values from their optimal values is the root cause of variance error in data-driven model. In summary, the three reasons which can contribute to a higher variance error in a data-driven model are:

1. Availability of insufficient historical data.
2. Presence of noise in the historical data.
3. Overly complex model i.e. the chosen model has high number of parameters.

It is shown in [7] that when the number of model parameters is significantly less than the number of available training data points, the variance error can be represented as:

$$variance_{error} \sim E[e^2]\frac{n}{N} \tag{1.8}$$

where, $E[e^2]$ is the noise variance, n is the number of model parameters and N is the number of available training data points. The equation 1.8 indicates that the variance error varies linearly with number of model complexity as shown in figure 1.5.

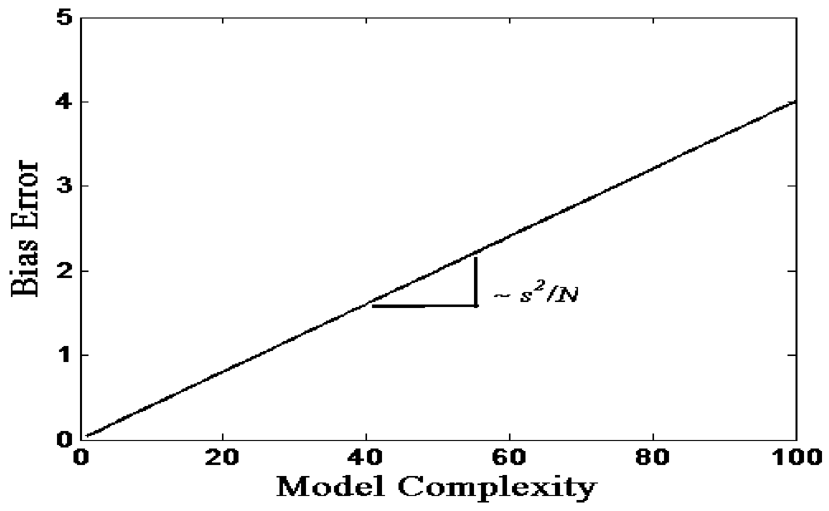

Figure 1-5 Dependence of Variance error on model complexity/number of parameters. The above curve is drawn under the assumption that number of model parameters is significantly less than the number of training data points.

It should be noted that the slope of the straight line is proportional to $E[e^2]/N$, which means that as the size of training dataset tends to infinity or the noise variance approaches a zero value, the variance error vanishes from a model. Therefore, unlike the bias error, we can make the variance error negligibly small if we have access to a large collection of noise free historical data. In such cases, the modeling error can be solely attributed to the bias error. However, in most of the real world scenarios the assumption of having large noise free historical data is far from being true. In fact, in most of the data-driven models the modeling error can be ascribed entirely to the variance error. One way to tackle this problem is to make the model less flexible. But reducing the flexibility is bound to increase the bias error sharply. Thus, in order to minimize the modeling error we need to strike a balance between these two error types [8] as discussed in the next section.

1.2.2) Bias-Variance Tradeoff

It is evident from figure 1.4 and figure 1.5 that the bias error and variance error are have conflicting characteristics. Assuming the availability of a large but finite set of training data points, the bias error decreases with the increase in model complexity while the variance error increases with the same. This contradictory behavior indicates towards the existence of an optimal model complexity at which the cumulative modeling error is minimum. This trade off is shown in figure 1.6. The optimal model complexity at which the modeling error in minimized is shown by an arrow.

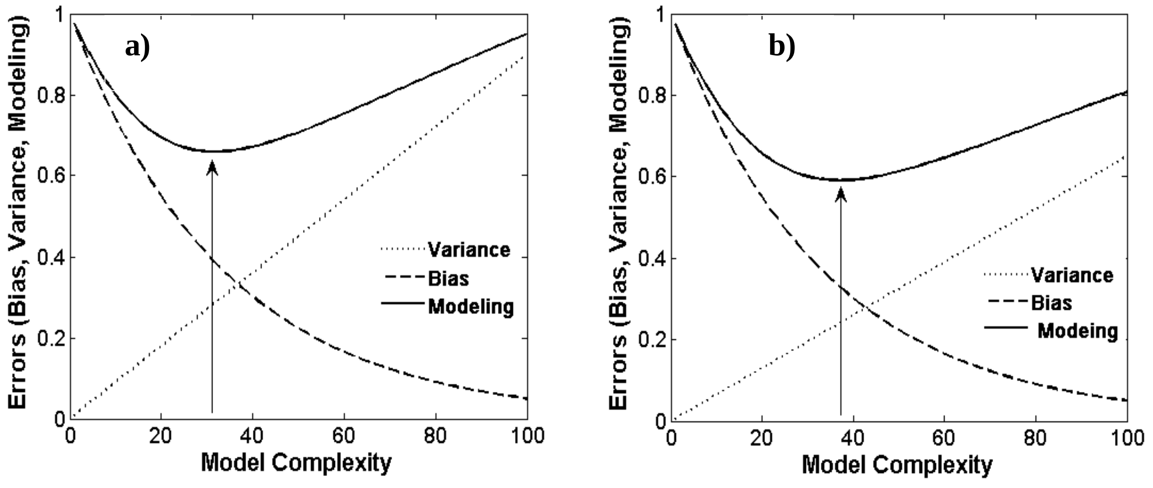

Figure 1-6 Graphical representation of the Bias Variance Tradeoff for two cases. a) **A model with steeper variance error line, which results from a poor quality training data.** b) **A model with relatively flatter variance error line.**

Figure 1.6 represents two scenarios that result in different optimal model complexities. In figure 1.6b the slope of variance error is lower than the one in figure 1.6a. As mentioned earlier, this slope can be lowered either by having a training dataset with lower noise variance or with larger size. The optimal number of model parameters is ~32 in figure 1.6a as compared to ~38 in figure 1.6b. This observation indicates that a good quality training dataset enables us to build a model with higher complexity.

It is important to mention here that the plots in figure 1.6 are hypothetical. In reality, we can't separately record the bias and variance error and draw the cumulative modeling error with a distinct minimum. Instead, we rely on certain techniques, called *regularization* techniques, that help in identifying a model with optimal complexity. Regularization, as described in section 2.3, is an extremely important concept in context of data-driven models and ignoring it can have serious repercussions of the quality of the final model.

Before concluding the discussion on bias-variance tradeoff, it is important to mention that performance of trained models in terms of modeling error is always evaluated using a *test* dataset. An ideal test dataset is chosen in such a way that it does not share any element with the training dataset. The reason behind having disjoint datasets for training and testing is that the modeling error is always minimum on the former. Therefore, it would be misleading to use a part of training dataset for a model's evaluation as it is bound to give a biased picture. As a rule of thumb, 20% of the available data is used for model validation and the rest of training. However, when the historical dataset is not large enough, its division into training and test data may cause appreciable information loss. In such cases, other methods of performance evaluation based on information criteria and statistical tests can be employed [9].

1.2.3) Brief Problem Statement

Here I briefly describe the problem that is addressed in this thesis. A more detailed problem statement is given in chapter 2. The goal of this work is to devise a novel regularization technique for the identification of *Takagi-Sugeno-Kang* (TSK) fuzzy models with the aim of attaining their optimal complexity. The main requirement from the proposed technique is its superior robustness so that it can perform well on historical process data having low signal to noise ratio, where the other contemporary regularization techniques may fail.

1.3 Organization of Thesis

The remainder of this thesis is divided into 3 chapters. In chapter 2, I start with a comprehensive review of data-driven modeling techniques for static nonlinear systems. It is shown that the foundation of such techniques exists in a generalized formulation known as *basis function formulation*. The focus is laid on three techniques viz. *Artificial Neural Networks*, *Radial Basis Function Network* and *Neuro-Fuzzy Systems*. It is shown that the primary difference between these techniques is in the manner in which their basis functions are constructed. In chapter 2, I further elaborate a special type of Neuro-fuzzy models called *Takagi-Sugeno-Kang* (TSK) fuzzy models. The two main issues related to the TSK models viz. the structure identification and the parameter identification, are discussed in detail. The need of regularized parameter estimation in TSK fuzzy models to preserve interpretability is specially emphasized. In this context, I describe the state of the art *regularization techniques* used for parameter estimation in TSK models. In the end of this chapter I formally define the problem that has been addressed in this thesis. In chapter 3, the mathematical formulation of the proposed knowledge based regularization technique for the TSK fuzzy models is presented. This technique works by modifying the basis function formulation of TSK fuzzy models by using a specialized function named as *knowledge function*. The fuzzy models built using the proposed technique are called *A Priori Knowledge-based Fuzzy Models* (APKFMs). In chapter 4, the effectiveness of APKFM as a system identification tool is proven by using it to model two static nonlinear systems. APKFM's superior robustness and interpretability is highlighted when the learning is carried out from inferior quality historical process data. Its performance is benchmarked against the other regularization techniques that are commonly used for the identification of TSK models. Finally, the analysis and observations are concluded in the end of chapter 4.

1.4 Original Contributions

There are two main contributions of this thesis that are listed in the order of increasing importance. Although these contribution may not be quite apparent at this point, listing them explicitly may help in understanding the proceeding sections and also in elucidating the importance of this work.

1. A modified version of 1st order Takagi-Sugeno-Kang (TSK) fuzzy model is proposed in section 3.3. This modification enhances the flexibility of a TSK fuzzy model by allowing a rule to have multiple local linear models. Depending on the position of an input with respect to its center, a rule chooses one of these local models during the computation of the fuzzy model output. In addition, the proposed modification improves the interpretability of TSK models and enables a modeler to obtain more consistent parameter estimates using his domain knowledge.

2. The main contribution of this thesis is the development of a technique that allows efficient incorporation of the qualitative domain knowledge in TSK fuzzy models. The fuzzy models trained by this technique are called *A-Priori Knowledge-based Fuzzy Models* (APKFM). The proposed technique can be viewed as a regularization technique that regularizes a model's parameter values during their identification from the historical process data. Unlike the existing regularization techniques, the proposed technique explicitly uses the qualitative knowledge of an expert to perform regularization. The resulting models are shown to have a superior combination of accuracy and interpretability compared to the other popular techniques.

Chapter 2: Data-driven Models for Static Systems

In static systems, there may exist a mapping, linear/nonlinear, from a n dimensional input space to a m dimensional output space, i.e.

$$Y = f(X) \tag{2.1}$$

where, $X = [x_1\, x_2 \ldots\ldots\, x_n]^T$ and $Y = [y_1\, y_2 \ldots\ldots y_m\,]^T$ are the input and output vectors respectively and $f(.)$ is the unknown function that relates the two vectors. The term static signifies that the mapping function is time invariant. The data-driven models which perform this type of input to output mapping are called *Multiple Input Multiple Output* (MIMO) models. The objective of any data-driven modeling technique is to identify an appropriate mapping function $\hat{f}(.)$ that can closely emulate the hypothetical function $f(.)$, by analyzing historical process data. Typically, a MIMO model is broken down into m independent *Multiple Input Single Output* (MISO) models. There are several advantages of this decomposition. For instance, each MISO model is simpler to understand because we can clearly identify the influence of different inputs on a single model output. Additionally, each MISO model can be build separately using different modeling schemes. For example, if we have two outputs such that one of them is expected to be linear and the other nonlinear in model inputs then we can utilize this information to build two separate data-driven models having better accuracies. Therefore, the focus of this thesis is exclusively on data-driven models pertaining to MISO systems.

In section 2.1, we present a generalized formulation of nonlinear data-driven models and discuss how such models can be identified from historically available input to output mapping information. Three modeling schemes viz. *Artificial Neural Networks*, *Radial Basis Function Networks* and *TSK Fuzzy models* are compared by highlighting their similarities and dissimilarities. We also briefly focus on the issues of structure and parameter identification in regards to the above mentioned modeling techniques.

2.1 Basis function Formulation

There are different ways of constructing a data-driven model, but practically every model can be written in a general *basis function formulation* as shown in equation 2.2:

$$\hat{f}(X) = \sum_{k=1}^{K} \theta_k^{(l)} * \phi_k\left(X, \theta_k^{(nl)}\right)$$

(2.2)

In the above equation, $\phi_k(.)$ represents a basis function, which is dependent on the input vector X and a set of nonlinear parameters $\theta_k^{(nl)}$. The model output $\hat{f}(X)$ is calculated as the weighted sum of K basis functions, where the weights are given by $\theta_k^{(l)}$. The superscript (l) signifies that the model output is linear in these parameters. A commonly used extension of equation 2.2 is obtained by replacing the linear parameters $\theta_k^{(l)}$ with a linearly parameterized function, resulting in more flexible formulation as shown in equation 2.3.

$$\hat{f}(X) = \sum_{k=1}^{K} \left(\theta_{k,0}^{(l)} + \theta_{k,1}^{(l)} x_1 + \theta_{k,2}^{(l)} x_2 \dots \dots \dots + \theta_{k,n}^{(l)} x_n\right) * \phi_k\left(X, \theta_k^{(nl)}\right)$$

(2.3)

Even in this extended formulation the model output continues to be linear in the parameters $\theta_{k,0}^{(l)}, \theta_{k,1}^{(l)}$ etc. This observation indicates that the nonlinearity in data-driven models is imparted by the nonlinear basis functions.

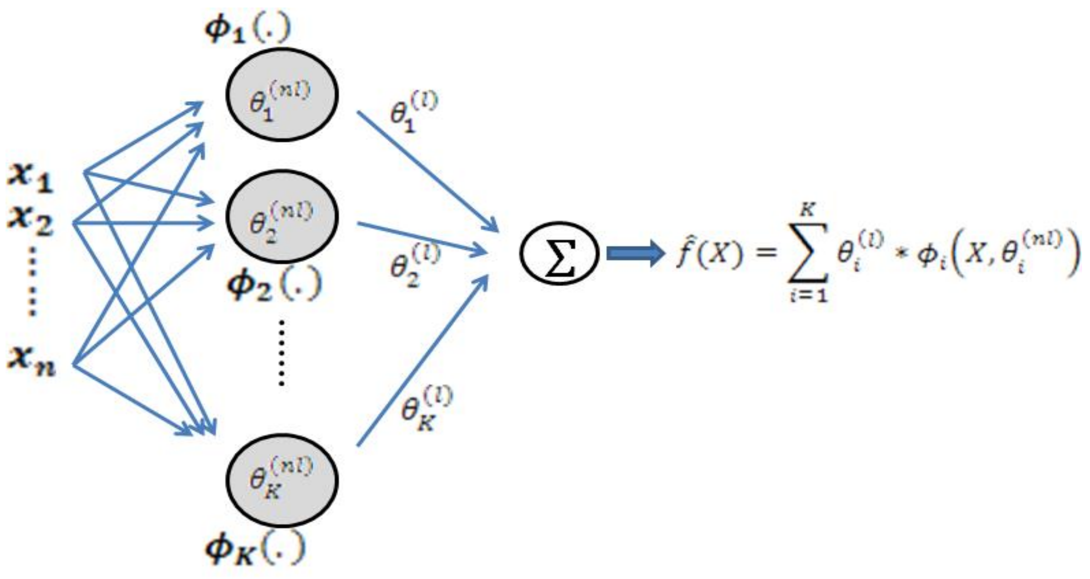

Figure 2-1 Graphical representation of data-driven models having the basis function formulation. The hidden layer of this networked structure consists of nonlinear basis functions. The output is calculated as the weighted sum of these basis functions.

Figure 2.1, shows how we can represent the basis function formulation of data-driven models in the form of a network. Clearly, this structure has 3 layers the *input layer*, the *output layer* with one node that computes the weighted sum and the *hidden layer* with K nodes. The shaded nodes in the hidden layer are the individual basis functions, which are sometimes referred as *neurons* in neural networks framework. Figure 2.2 further elaborates the functioning of these neurons. The basic idea is to combine the input vector (X) and the set of nonlinear parameters $(\theta^{(nl)})$ using a *construction mechanism*. The outcome of this combination is fed to the basis function to generate the output as $\phi(X, \theta^{(nl)})$.

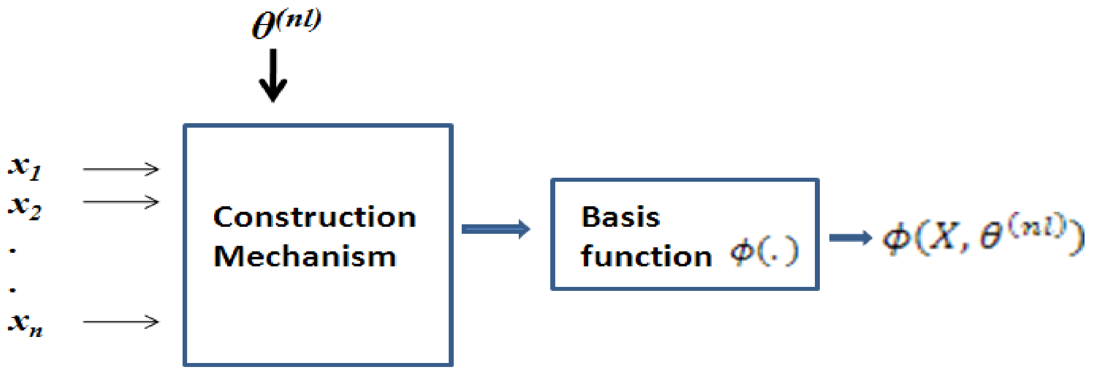

Figure 2-2 A schematic showing the functioning of an neuron/basis function. The construction mechanism combines the nonlinear parameters with the input vector and feeds it to the basis function.

There are different types of construction mechanisms of a basis function. Generally, a data driven model will have all its basis functions constructed in the same way. The construction mechanism decides the type and characteristics of a data-driven model. The three main construction mechanisms are, *Ridge* construction, which leads to *Artificial Neural Networks* (ANNs), *Radial* construction that yields *Radial Basis Function Networks* (RBFNs) and last but most relevant to this thesis is the *Tensor Product* construction, which forms the backbone of *Fuzzy Models.*

The problem of identifying an ideal data-driven model requires us to solve following three important sub-problems:

1. How many basis functions are needed to attain the desired nonlinear mapping ?
2. What should be the optimal values of nonlinear parameters $\theta^{(nl)}$?
3. What should be optimal values of linear parameters $\theta^{(l)}$?

Clearly, the first sub-problems is aimed at finding a model with optimal structure. Therefore, any technique that attempts to solve this problem is called a *structure optimization* technique. The remaining two sub-problems are *parameter optimization* problems. Most of the modeling techniques differ only in the manner in which the first two sub-problems are tackled. It is because, the 3rd sub-problem is simply a linear regression problem, wherein the linear parameters are invariably obtained in a least square sense i.e. by minimizing the mean squared error.

In the next few section we present a brief discussion on the three most widely used data-driven modeling techniques viz. Artificial Neural Networks, Radial Basis Function Networks, and Fuzzy Models. The focus would be on elucidating the similarities and differences that exist between these modeling techniques.

2.1.1) Artificial Neural Networks (ANNs)

The theory of Artificial Neural Networks (ANNs) draws its motivation from biological network of neurons that exists in the brains of living beings. However, unlike the extremely complex structure of a brain, ANNs have a relatively simple structure. Although ANNs can be made more complicated, it has been commonly observed that an overly complex ANN generally performs poorly. Nevertheless, both ANN and biological neural network are made from similar building blocks called *neurons*. A neuron can be defined as an entity which generates an output when excited with an input signal. The excitation of a biological neuron is caused by the imbalance in certain chemicals, whereas the excitation of an artificial neuron is purely mathematical. In ANN the neurons are arranged in a layered structure similar to the one shown in figure 2.1. The connections between the neurons generally exist from left to right so that information can travel in only one direction. Such ANNs are referred as *Feed Forward Neural Networks*. Figure 2.1 suggests that each neuron can be interpreted as a basis function $\phi(.)$ with nonlinear parameters $\theta^{(nl)}$. A neuron takes an input vector X and combines it with the nonlinear parameters using a construction mechanism and feeds it to a basis function that generates an output (refer figure 2.2).

Construction Mechanism: In ANN, we use the *Ridge Construction* mechanism which works by taking a dot product of the input vector X with the nonlinear parameter vector $\theta^{(nl)}$ i.e.

$$s = X^T \cdot \theta^{(nl)}$$

(2.4)

The resulting scalar quantity, $\boldsymbol{s}$, is given as an input to a predefined univariate basis function, which is often called an *activation function* in ANN literature. Some commonly used activation functions are:

$$\text{Logistic Function} = \frac{1}{1 + \exp{(-s)}}$$
$$\text{Hyperbolic Tangent} = \tanh{(s)}$$

(2.5)

Assuming that the basis function is chosen as hyperbolic tangent, the output of an ANN can be written as:

$$\hat{f}(X) = \sum_{k=1}^{K} \theta_k^{(l)} * \tanh{(s)}$$

(2.6)

In figure 2.3, we show the surface of a basis function generated using the ridge construction for a 2-dimensional system. The nonlinear parameter vector is chosen as $\theta^{(nl)} = [1\ 1]$ and the hyperbolic tangent function is used as the basis function. A contour map corresponding to the 3-dimensional surface is also plotted in figure 2.3. It should be noted that because of the nature of ridge construction, the basis function increases along the direction defined by the nonlinear parameter vector. This direction is shown by an arrow on the contour plot. In any other orthogonal direction the basis function is invariant. An ANN with ridge construction is specifically called a *Multiple Layer Perceptron* (MLP).

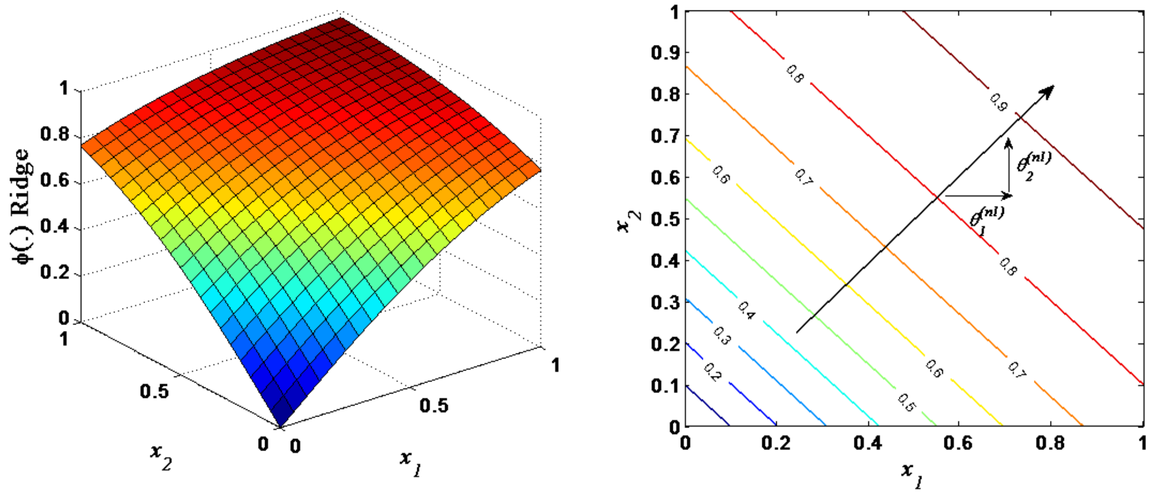

Figure 2-3 A bivariate basis function resulting from the Ridge construction. The nonlinear parameter $\theta^{(nl)}$ vector is taken as [1 1] and hyperbolic tangent is chosen as the activation function. The direction of increase of basis function, as shown as shown in the contour map, coincides with the nonlinear parameter vector.

Structure & Parameter Optimization: The optimal structure of a MLP is determined inductively, i.e. we start with a very simple structure with a few basis function/neurons in the hidden layer and train the model with the training data. The model performance is evaluated and neurons are added, incrementally, to the hidden layer until we achieve the desired accuracy. This approach generally ensures that the model attains reasonable flexibility to mimic the underlying nonlinear process without becoming over-parameterized. The drawbacks of an over-parameterized model are discussed in section 2.2. The nonlinear parameters of the basis functions are estimated using gradient-based optimization techniques. Because of the use of differentiable activation functions (equation 2.5), the output of a MLP has well defined derivatives with respect to the nonlinear parameters. State of the art training schemes of MLP involve the use of *Levenberg-Marquardt* or a *quasi-Newton* or a *conjugate-gradient* algorithm [10-13]. If a MLP has multiple hidden layers, the famous

Backpropagation algorithm is employed to estimate the nonlinear parameters of basis functions located in different hidden layers [14, 15]. Additionally, the parameter identification is usually done in a regularized manner to reduce the over-fitting on training data (refer section 3.4 for details regarding regularized parameter identification). The most popular choice of regularization technique for ANNs is the *Early Stopping* technique, which is explained in section 3.4.1.

2.1.2) Radial Basis Function Networks (RBFNs)

The theory of RBFNs has developed independently of ANNs. The radial basis functions were used as interpolation functions by mathematicians long before they were first used to build a specialized neural network [16]. Like MLPs, the RBFNs also have the neurons/basis functions arranged in a layered structure and connected in a feed forward sense. However, the number of hidden layer is always one in RBFNs. Since their invention in 1988 [17, 18], they have been successfully used to in a variety problems pertaining to prediction, classification and time series analysis [19-22].

Construction Mechanism: RBFNs employ *Radial construction* in their basis functions. The scalar s, which is dot product of input vector and parameter vector in MLPs, changes to a distance measure between the input vector (X) and the center of a radial basis function (C), i.e.

$$s = \left|\left|X - C\right|\right| = \sqrt{(X - C)^T (X - C)}$$

(2.7)

In equation 2.7 the scalar s is simply the *Euclidean* norm. A more general form of norm is preferred which transforms the distance measure according to a norm matrix $\mathbb{N}$. Therefore, equation 2.7 changes to:

$$s = \left|\left|X - C\right|\right|_{\mathbb{N}} = \sqrt{(X - C)^T \mathbb{N} (X - C)}$$

(2.8)

The distance measure given in 2.8 is computed by scaling the input dimensions according to the norm matrix $\mathbb{N}$, which results in a more flexible basis function. A typical norm matrix is a diagonal matrix, as shown in 2.9, wherein the i^{th} diagonal element represents the scaling factor $(1/\sigma_i^2)$ for i^{th} input dimension. Clearly, when all the dimensions are assigned equal scaling factors, the norm matrix transforms into an identity matrix and we obtain the Euclidean norm.

$$\mathbb{N} = \begin{bmatrix} 1/\sigma_1^2 & 0 & \cdots & 0 \\ 0 & 1/\sigma_2^2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1/\sigma_n^2 \end{bmatrix}$$

(2.9)

The most common choice of basis function in the radial construction is the *Gaussian* function. As a result the output of a RBFN can be written as:

$$\hat{f}(X) = \sum_{k=1}^{K} \theta_k^{(l)} \mathrm{e}^{-\frac{\boldsymbol{s}^2}{2}}$$

(2.10)

where, the scalar s is obtained from either equation 2.7 or 2.8. The other common choice of basis function in radial construction is the inverse multi-quadric function given in equation 2.11. This function introduces an additional nonlinear parameter a in the data-driven model.

$$\phi(.) = \frac{1}{\sqrt{s^2 + a^2}}$$

(2.11)

Figure 2.4 shows the surface of basis function generated using the radial construction for a 2-Dimensional system. The coordinate of the center of a Gaussian function is chosen as $[0.6\ 0.4]$. The surface shown in figure 2.4a corresponds to the case when Euclidean norm was chosen as the distance measure. This is evident from the corresponding contour map, which is a collection of several concentric circles. In the second case we used a modified distance measure, in which case a diagonal matrix, $\mathbb{N} = \begin{bmatrix} 1/\sigma_1^2 & 0 \\ 0 & 1/\sigma_2^2 \end{bmatrix}$, was used to compute the norm. The values of σ_1^2 and σ_2^2 were taken as 0.04 and 0.01 respectively. As a result the basis function became asymmetric as can be confirmed by its elliptical contour lines.

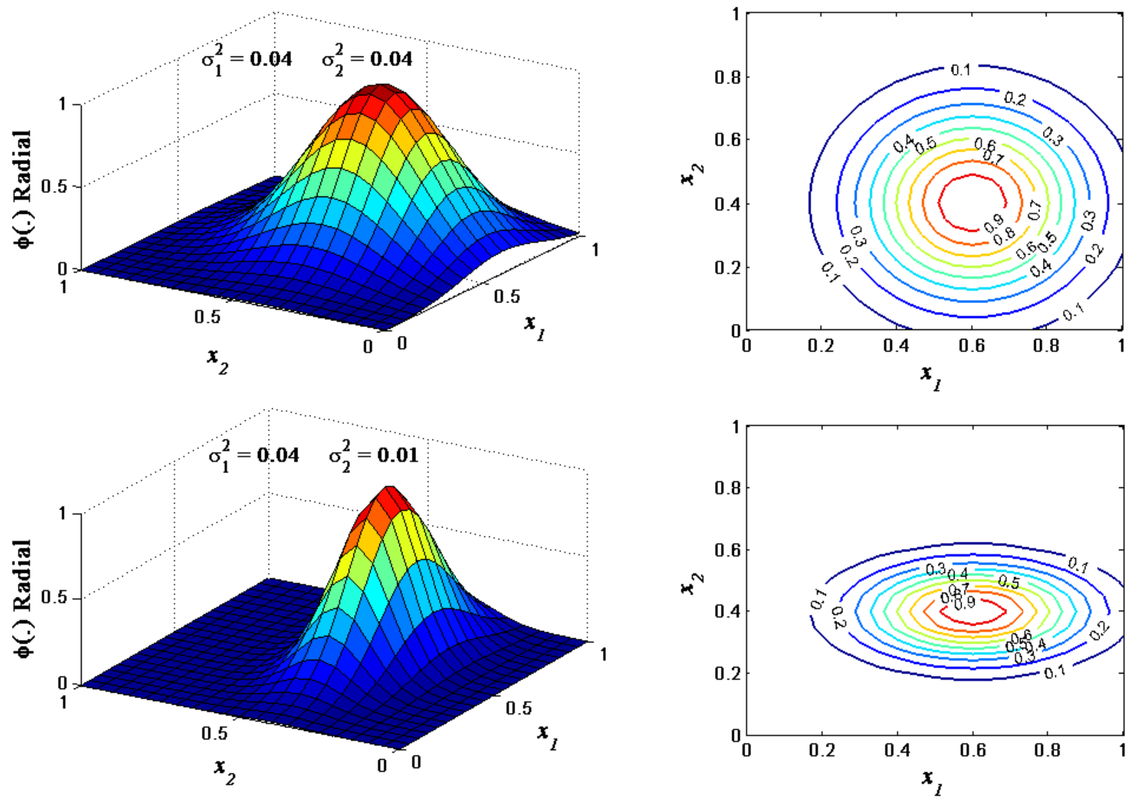

Figure 2-4 A bivariate basis function resulting from the Radial construction. The center of the radial basis function is taken as [0.6 0.4] and Gaussian function is chosen as the activation function. The top surface is results from taking the distance measure as the Euclidean norm. In the bottom surface, the distance measure is modified according to a given norm matrix ℕ.

It is worth comparing the basis function having radial construction (figure 2.4) with the one having ridge construction (figure 2.3). The later is global in the sense that it has a non-zero value in the entire input space. On the contrary, the radial construction yields a local basis function that has a limited validity region around its center. As soon as we move out of this region, the basis function output becomes zero and it does not have any impact on the model output. However, a global basis function resulting from ridge construction has a validity region that span the entire input space and hence, irrespective of the location of an input, it always has an influence on the model output. The preceding analysis suggests that in terms of interpretability RBFNs have an edge over MLPs. It is because, in RBFNs the overall model output can be considered as an outcome of the interpolation between several local models whose validity regions are defined by their respective basis function. The same interpretation cannot be extended to MLPs because of the global nature of their basis functions. Therefore, because of the poor interpretability of the MLPs, or ANNs in general, they are called *black-box* models.

Structure & Parameter Optimization: The nonlinear parameters $\theta^{(nl)}$ in RBFNs can be divided into two sets i.e. the centers of radial basis functions (C) and the variance values (σ_i^2). These parameters can be estimated either in an *Unsupervised* manner or a *Supervised* manner. In unsupervised methods we make use of superior geometric interpretability of RBFNs. The most common unsupervised approach works by performing clustering of the input data points to gain more insight about their spatial distribution in the input space. Thereafter, the basis function centers are placed over the cluster centers. This approach has an extremely desirable effect that higher number of basis functions are placed in those input subspaces where the distribution of input data points is dense. Therefore, we always have sufficient data to estimate the linear parameters $(\theta^{(l)})$ associated with the basis function, which results in model with low variance error. On the other hand, if we randomly place the centers in the input space without taking into consideration the distribution of data points, the model will have very high variance error in the sparsely populated regions of the input space. Various algorithms can be employed to identify the optimal location of cluster centers with *K-means* algorithm being the most popular and time efficient [23, 24]. The only shortcoming of K-means clustering is that it requires the prior knowledge of number of clusters needed to represent a given distribution. For this purpose, there are techniques like *subtractive clustering* that give a reasonable estimate of the number of cluster needed [25].

The biggest drawback of an unsupervised method is that it does not take into account the complexity of the system being modeled. Ideally, we should have more basis functions in those input subspaces where the system behavior is highly nonlinear and few basis functions where the behavior is relatively smooth. However, in clustering the placement of basis functions solely depends on the input data distribution and the information about the output is discarded. To rectify this drawback, a supervised approach, as proposed in [26] can be used. The basic idea is to incorporate a subset selection scheme in the identification of optimal locations of basis function centers. In this strategy, we first build large set of potential basis functions with their centers and variances fixed at some values. Thereafter, *Orthogonal Least*

Square (OLS) algorithm [27] is used to pick only the relevant basis functions while discarding the others. Since OLS algorithm works in a supervised manner, the chosen basis functions are distributed in the input space according to the complexity of the underlying system. In another supervised approach, we make use of nonlinear optimization algorithms for the estimation of optimal center and variance values. This approach gives the best estimates of nonlinear parameter values but requires more computational resources. Another problem with nonlinear local optimization is that it needs good initialization of the unknown nonlinear parameters. However, this problem can be addressed by using any of above mentioned techniques (Clustering, OLS algorithm), which usually provide reasonable parameter estimates.

2.1.3) TSK Fuzzy Models

The invention of fuzzy model was primarily aimed at developing data driven models that can also utilize a domain expert's knowledge. Unlike a MLP, the nonlinear mapping done by a fuzzy model is expected to be transparent. Since its construction is knowledge based, we can understand how does a fuzzy model calculate its output. However, unless specified by an expert, a fuzzy model does not have the ability to determine the optimal values of its parameters. For this purpose, it can be formulated as a neural network so that the optimal parameter values can be learned in the same manner as the hidden layer parameters are learned in MLP. In this section, Takagi-Sugeno-Kang (TSK) fuzzy models are compared with MLPs and RBFNs in terms of their construction and the structure/parameter optimization.

Construction Mechanism: TSK models use the *tensor product* construction, in which the multivariate basis functions are obtained from several univariate basis function [28]. This construction mechanism is fairly easy to understand and is demonstrated in figure 2.5. Each input dimension is assigned a certain number of univariate basis functions. The number, shape and spread of these functions are usually decided by the domain expert. In this example, we define two and three *Triangular* basis functions for the inputs x_1 and x_2 respectively. Thereafter, the multivariate basis functions are simply determined by forming the tensor products of these univariate functions i.e. each function of an input is multiplied with each function of the other inputs. Using this method for the current example, we get six conical functions as shown in figure 2.5 along with their contour maps. It should be noted that the a basis function resulting from tensor product construction is local in the sense that it has a limited validity region as is in the case with a radially constructed basis function.

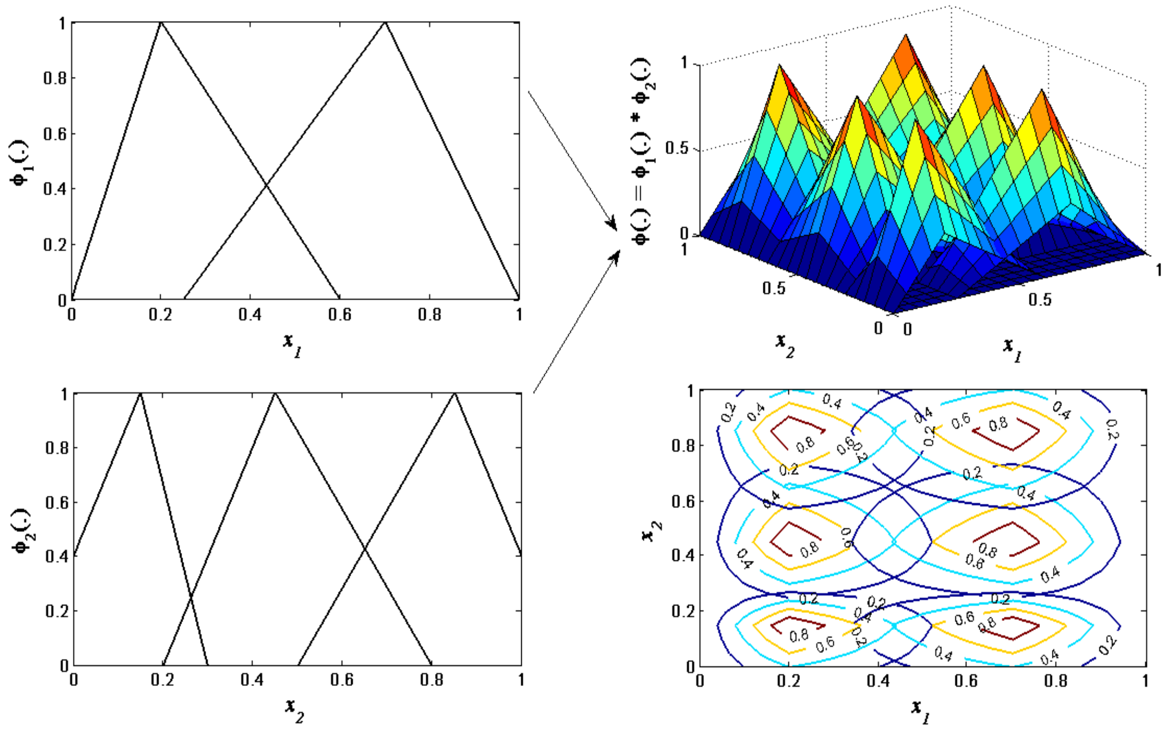

Figure 2-5 Demonstration of the tensor product construction used in neuro-fuzzy models. The conical triangular basis functions are obtained from the univariate triangular functions. The contour maps of multivariate basis functions shows that they are local in nature.

In fuzzy logic, the univariate basis function are also known as *membership functions*. These functions are used to define the membership of an element to a fuzzy set. Fuzzy sets are similar to the conventional sets except that they overlap with each other. As a result, an element can belong to two fuzzy sets with different degrees of membership. A fuzzy set is usually referred by its linguistic label such as *low*, *large*, *high* etc. Fuzzy set theory was mainly developed to emulate the manner in which humans characterize an object. We usually define an object with a some uncertainty. The overlapping nature of basis/membership functions is exploited to characterize this uncertainty about a variable/object. In fuzzy logic, the multivariate basis functions that results from tensor product of univariate basis functions are called *Rules*. The collection of all multivariate basis functions is referred to as a *Rule-base*. A more detailed discussion on fuzzy logic and TSK fuzzy models is included in the

Appendix. Here we are mainly interested in the functional representation of a TSK model, whose output can be simply written as:

$$\hat{f}(X) = \sum_{k=1}^{K} P_k(X) * \phi_k\left(X, \theta_k^{(nl)}\right)$$

(2.12)

Where, $P(X)$ is a polynomial defined on input X and $\phi_k()$ represents the k^{th} basis function resulting from tensor product construction. The order of a TSK model is decided by the order of the polynomial used in the model. A 0th and 1st order TSK models can be represented by equation 2.2 and 2.3 respectively. Equation 2.12 suggests that a TSK model's output is the weighted average of the local models, $P_k(X)$, where the weights are computed from the basis functions, $\phi_k()$. Each basis function has a local model associated with it and a validity region in the input space (figure 2.5) where its value is non-zero. Depending on the location of the input, only a few basis functions have non-zero values and contribute the net output of the system. There is another type of fuzzy models called *Mamdami* models. The output of these models does not have an intuitive functional form. Although they are not relevant to this thesis, Mamdami models are described in section 2.1 of the appendix for interested readers.

Fuzzy models are not suited for high dimensional systems as they suffer from the *curse of dimensionality*. For a general case of an n dimensional system, in which the i^{th} dimension is assigned M_i univariate basis functions, the total number of multivariate basis function obtained using the tensor product construction is equal to $\prod_{i=1}^{n} M_i$. Clear, the number of basis function grows exponentially with the input dimension. For this reason, fuzzy models, based on tensor product construction, are not very popular for systems with $n > 5$. To get rid of curse of dimensionality in fuzzy models, an interesting approach is proposed in [29] that uses input space clustering to determine the univariate basis functions for all inputs.

Structure & Parameter Optimization: The structure/parameter optimization of TSK fuzzy models is not discussed here. Instead, because of its direct relevance to the thesis, this issue is given an extensive treatment in sections 2.2.1 and 2.2.2.

2.2 Optimization of TSK Fuzzy Models

During the optimization of a TSK model, its components are tuned according to an available training dataset. A drawback of this optimization scheme is that the final model quality depends a great deal on the training dataset, which usually is corrupted with noise. During the training, if special care is not taken then the optimized model may perform erratically in never-seen conditions. In the next sections, we will discuss some popular methods for structure and parameter optimization that are used for identifying TSK models from historical data.

2.2.1) Structure Optimization of TSK Models

The structure optimization addresses the issue of determining optimal number of model parameters. The number of parameters of a TSK model is governed by the number of basis functions used in it. A model with many basis functions will be overly complex with several superfluous parameters. Although such an over-parameterized model may have the flexibility to mimic highly nonlinear behavior (low bias error), it would suffer from higher variance error at the same time. This tradeoff between the bias error and variance error is outlined in section 1.2.2. The motivation behind structure optimization is to build a TSK model with just enough basis functions such that a nonlinear system can be satisfactorily mimicked. Significant amount of work has been done in this area and there are numerous techniques available to perform structure optimization [30, 31]. These techniques can be broadly categorized into 1) *Explicit Optimization* techniques and 2) *Implicit Optimization* techniques. In explicit optimization, the structural elements are physically removed/added until the optimal model structure is identified. Some of the common explicit structure optimization schemes are discussed here briefly:

Backward Elimination: Backward elimination is an iterative technique in which we start with the most complex model structure and remove the redundant structural elements iteratively. In this context, Orthogonal Least Square (OLS) is a popular algorithm which works on linearly parameterized models [27]. Other variations of backward elimination techniques can be found in neural network literature where it is commonly referred to as *Pruning* [32, 33].

Forward Selection: As opposed to backward elimination, in forward selection we start with a very simple model. Thereafter, we iteratively add structural elements (basis functions) in it, until a desired complexity is achieved. Generally, the modeling error drops sharply in the beginning with the augmentation in model structure. However, the rate of reduction in modeling error diminishes as the model structure becomes more complex. This behavior can be explained by observing the variation of bias error with increasing model complexity as shown in figure 1.4. It would be needless to add more structural elements when the addition is not significantly reducing the modeling error as we may unnecessarily increase the model complexity. The orthogonal least square (OLS) for forward selection in commonly used for linearly parameterized models. Other popular forward selection techniques for TSK models are Adaptive Spline Modeling (ASMOD) [34, 35] and Local linear model Tree (LOLIMOT) [36, 37]. Both of these approaches have a common modus operandi wherein a basis function is added to that region of the input space which has the highest modeling error.

Genetic Algorithm Based Optimization: In another interesting approach, FUREGA [38], *Genetic Algorithm* is used for obtaining the optimal number of basis functions. Genetic algorithm is an unconventional but a very powerful optimization technique. It is primarily suitable for those systems that do not have well defined derivatives of model output with respect to its parameters. A comprehensive discussion on genetic algorithm can be found in [39, 40]. The technique FUREGA stands for *FUzzy Rule Extraction by Genetic Algorithm*. The main challenge in this approach is to code the basis functions into binary numbers so that the Genetic algorithm can be implemented. The simplest coding scheme is to represent an active basis function by 1 and an inactive one by 0. Using this scheme, a population of fuzzy models with different set of basis functions can be easily generated. Thereafter, the fittest fuzzy model with lowest modeling error can be chosen and refined by eliminating all the redundant basis functions with a label 0. These are the insignificant basis functions that have very less contribution in reducing the modeling error. The extensions and modifications of FUREGA can be in found in [41].

In contrast to the above mentioned explicit optimization techniques, the implicit optimization techniques do not require physical removal of redundant model parameters. Instead the complexity of model is manipulated by imposing certain restrictions on the values that the parameters can assume. The implicit optimization techniques are commonly known as *Regularization Techniques* because they regularize the parameter identification step of data-driven models. The entire section 2.3 is devoted to the state of the art regularization techniques and their implementation to attain optimal model complexity. However, we first discuss the basic parameter optimization techniques of TSK models, in section 2.2.2, before shifting our focus on regularization techniques.

2.2.2) Parameter Optimization of TSK Models

After the structure optimization of a model, the next logical step is to fine tune its parameter values to further improve its performance. Parameter optimization can also be carried out simultaneously with the structure optimization. The parameter optimization step can be viewed as a linear/nonlinear optimization problem requiring minimization of some predefined loss function. The most commonly used loss function is the *mean squared error*, which is derived from the training data. A TSK model has two different set of parameters namely 1) *Consequent* parameters and 2) *Premise* parameters. These parameters define different components of a TSK model. Another key difference between the two is that the model output is linear in the former and nonlinear in the later. Therefore, it is always preferred to estimate them separately rather than optimizing them in a single step.

Consequent Parameter Estimation: The consequent parameters, for TSK fuzzy models, correspond to the parameters of local models $P(X)$, as shown in equation 2.12. Since these local models are polynomials defined on input X, the net output of a TSK model is always linear in consequent parameters. Therefore, the estimation of these parameters can be simply a linear optimization problem. *Global Linear Least Square Estimate* (Global LSE) is by far the most commonly used technique for this purpose [42]. By '*global*' we mean that all the linear parameters are estimated in a single LS optimization. Here we provide a brief overview of global estimation of the consequent parameters of a 1st order TSK model that has n inputs and R basis functions. In this model the r^{th} basis function will have a local model with $n + 1$ parameters i.e. $\theta_0^r, \theta_1^r \ldots \ldots \theta_n^r$ (refer equation 2.3). Since there are R such basis functions, the total number of unknown consequent parameters are $R(n + 1)$. Hence, the vector of unknown consequent parameter, θ, can be represented as:

$$\theta = [\theta_0^1\ \theta_1^1 \ldots.\theta_n^1\ \theta_0^2\ \theta_1^2 \ldots.\theta_n^2 \ldots\ldots..\theta_0^R\ \theta_1^R \ldots.\theta_n^R]$$

(2.13)

We want to estimate the optimal consequent parameter vector, $\hat{\theta}$, using a set of M training data points. Assuming that the premise parameters are fixed at known values, each data point results in an equation, which is linear in consequent parameters. For example, the i^{th} input, $X^i = [x_1^i, x_2^i \ldots\ldots\ldots x_n^i]$, results in equation 2.14.

$$\sum_{r=1}^{R} \left(\theta_0^r + \theta_1^r x_1^i + \theta_2^r x_2^i \ldots..\theta_n^r x_n^i\right)\phi_r(X^i) = \hat{f}(X^i)$$

(2.14)

In equation 2.14, the fixed nonlinear parameters, $\theta_r^{(nl)}$, are not shown in the notation of basis functions for the sake of simplicity. The left hand side of equation 2.14 is the model output, while the right hand side, $f(X^i)$, represents the target output.

Using M training data points we can come up with a system of linear equations having the following canonical representation:

$$A\theta = b$$

(2.15)

Where, θ is the unknown parameter vector given in equation 2.12, b is a vector of target outputs given by $[f(X^1)\ f(X^2) \ldots\ldots f(X^M)]^T$ and A is a $M \times (n+1)$ regression matrix given by:

$$A = \begin{bmatrix} \phi_1(X^1) & \phi_1(X^1)x_1^1 \ldots \phi_1(X^1)x_n^1 & \cdots\cdots\cdots \phi_R(X^1) & \phi_R(X^1)x_1^1 \ldots \phi_R(X^1)x_n^1 \\ \phi_1(X^2) & \phi_1(X^2)x_1^2 \ldots \phi_1(X^2)x_n^2 & \cdots\cdots\cdots \phi_R(X^2) & \phi_R(X^2)x_1^2 \ldots \phi_R(X^2)x_n^2 \\ \ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots\ldots \\ \ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots\ldots \\ \ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots\ldots \\ \phi_1(X^M) & \phi_1(X^M)x_1^M \ldots \phi_1(X^M)x_n^M & \cdots\cdots\cdots \phi_R(X^M) & \phi_R(X^M)x_1^M \ldots \phi_R(X^M)x_n^M \end{bmatrix}$$

(2.16)

The linear system can be solved in a least square sense i.e. by minimizing the sum of squared errors. The formulated optimization problem is given by equation 2.17.

$$\arg \underset{\theta}{min} : \ \frac{1}{M}(e^T e)$$

(2.17)

Where, e is the vector of prediction error i.e. $e = A\theta - b$. The mean squared error function is quadratic in unknown consequent parameters, hence a global minimizer can be obtained using equation 2.18 [43, 44]:

$$\hat{\theta} = (A^T A)^{-1} A^T b \tag{2.18}$$

The square matrix $A^T A$ represents second derivative of the quadratic loss function with respect to the unknown parameters. This matrix is known as *Hessian* and it plays a vital role in the quality of the least square estimate given by equation 2.18. A poorly condition Hessian leads to an unstable solution, which is overly sensitive to the perturbations in the regression matrix. It should be noted that this method gives reasonable parameter estimate only when we have an over-determined linear system i.e. when the number of unknown parameters are significantly less than the number of training data points. An over-determined system usually gives a well-conditioned Hessian.

Although global LSE is a fast and efficient way to estimate consequent parameter, more often than not, in real world problems it yields an inferior TSK model with high variance error. This problem arises because the real world data generally yields an ill-conditioned linear system. Therefore, global LSE is rarely used for consequent parameter estimation. In section 2.3, we will discuss some regularization techniques that can be used to address the problem of ill-conditioned linear systems.

Premise/Antecedent Parameters Estimation: The premise parameters, $\theta_k^{(nl)}$, are the parameters associated with the basis functions. These parameters define the characteristics of a basis function such its shape and coverage in the input space. Unlike the consequent parameters, the estimation of premise parameters is not quite straight forward owing to the fact that the model output is nonlinear in them. Hence, for this purpose we rely on either nonlinear *gradient-based* or nonlinear *derivative-free* optimization techniques, both of which are usually more complex than linear optimization techniques. For detailed description of optimization schemes for premise parameters estimation, refer [45], as here we merely discuss the pros and cons of different techniques. The benefit of derivative-free techniques like genetic algorithm is that they rarely get trapped in an inferior local minimum. However, the price that you pay is a considerably higher computational demand. On the other hand, the gradient-based algorithms are computationally efficient but have a tendency to get stuck in a local minimum, if either the initial parameters values are poorly chosen or when the surface of loss function is highly contoured. As a rule of thumb, we also first try using a gradient–based technique to identify premise parameters. Only when the resulting model performs poorly on a never-seen dataset, a derivative-free technique is employed.

2.3 Regularized Estimation of TSK Model Parameters

In section 1.2 we defined the concept of complexity for parameterized data-driven models. The complexity of a model is governed by the number of its parameters. An overly complex model would be needlessly flexible and is not recommended to approximate a relatively simpler system. This was shown in figure 1.2, where a flexible neural-network model did a poor job of approximating a 3rd order polynomial. On the other hand, the linear model with fewer parameters was not sufficiently flexible to approximate the same polynomial. In data-driven modeling we rarely have an insight about the ideal model complexity. Thus, identifying a model with optimal complexity is an indispensable step in data-driven modeling. In section 1.2.2, we discussed the tradeoff between the bias error and the variance error and how this tradeoff leads to an optimally complex model. Regularization techniques are the parameter estimation techniques which attempt to strike a balance between the bias and the variance error. These techniques, also referred to as *implicit structure optimization* techniques, work by reducing the number of effective parameters of the model even though the number of nominal parameters remains the same. As a result, the model does not remain as flexible as it is apparent from the number of its nominal parameters. Here we present a comprehensive list of different regularization techniques relevant to the TSK models.

2.3.1) Regularization by Early Stopping

This is the least computationally expensive yet effective regularization technique which can be used with any iterative optimization algorithm. In an iterative optimization scheme, we generally update the parameter values until some type of convergence criteria is met. In this technique a certain percentage (usually 20%) of the training data, called validation data, is used to evaluate the performance of the fuzzy model after each iteration. The iterations are stopped when the error on validation dataset hits the minimum. The concept of early stopping is shown in figure 2.6. Starting from the initial guess we move in the direction, which decreases the loss function.

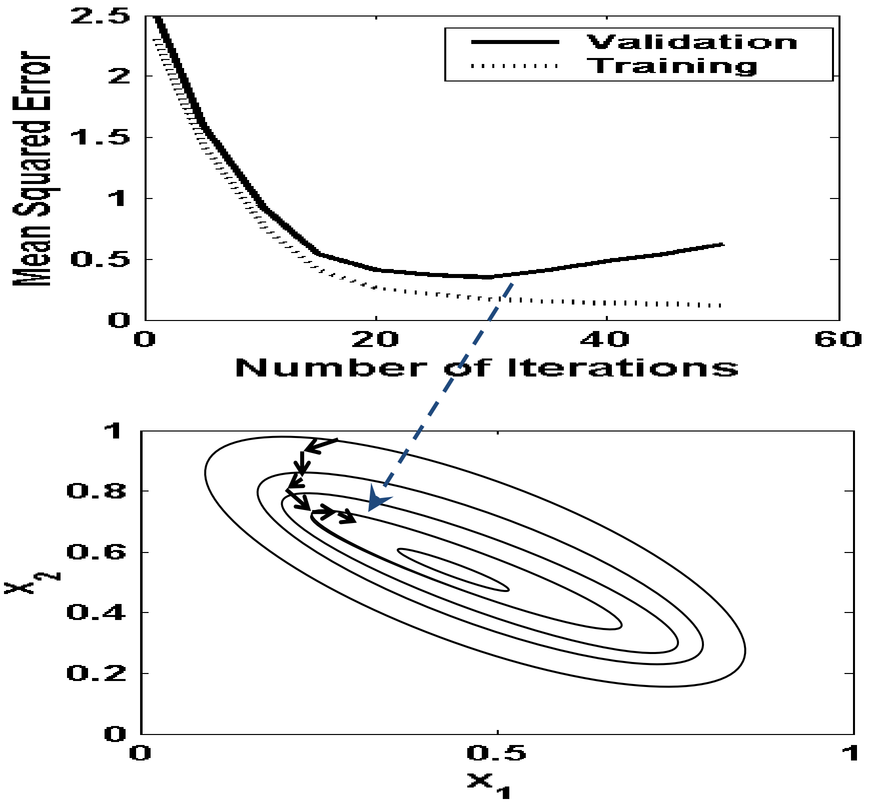

Figure 2-6 Graphical depiction of the early stopping regularization technique. In the top plot, we can observe the typical variation in mean squared error on validation and training datasets with respect to the number of iterations. The iterations are terminated when the validation error reaches the minimum (refer the dashed arrow).

The rationale behind this technique is that, earlier during the optimization, only the significant parameters are driven to their optimal values, while the less significant parameters remain close to their initial values. It is only during the later iterations the less important parameters start to move towards their optimal values, which is when the variance error begins to accumulate. The validation dataset helps us identify the time when this transition occurs. In the beginning of optimization, the validation error decreases since the significant parameters are attaining their optimal values. When the validation error hits its minimum value we can roughly say that all the relevant information has been extracted from the training data. At this moment the training should be terminated because further iterations are merely going cause over-fitting on the training data

2.3.2) Regularization by Staggered Optimization

This technique is invariable used for the optimization of fuzzy model parameters [46, 47]. In staggered optimization we do not estimate all the parameters simultaneously. Instead, the parameters are grouped into different subsets and we optimize one set of parameters at a time, while keeping parameters of other sets constant. This technique should reduce the variance error since we estimate lesser number of parameters, using the same training data, compared to the case when all the parameters are estimated simultaneously. However, staggered optimization is effective only when the model parameters can be naturally grouped into different sets. Parameters belonging to a group should have the same physical significance and have similar influence on the outcome of a model. For example, in TSK fuzzy models two such naturally occurring groups exist in the form of premise parameters and consequent parameters. As discussed in section 2.2.2, the premise parameters define the characteristics of basis functions, while the consequent parameters are the coefficients of the local linear models. A single iteration, in the staggered optimization of TSK models, is split into two steps 1) Fix premise parameters and estimate the consequent parameters 2) Fix the consequent parameters at the value obtained in the previous step and estimate the premise parameters. Several such iterations may be needed before we fulfill some convergence criteria. There is another advantage of estimating parameters of a neuro-fuzzy model in the staggered manner. If all the parameters are to be estimated simultaneously then a nonlinear optimization technique will have to be applied to all the parameters, even though the model is linear in consequent parameters which are significant in number. Staggered optimization allows us to separate the estimation of linear and nonlinear parameters thereby reducing the overall computational demand of the estimation process.

2.3.3) Regularization by Constraints

Constraints based regularization is an extremely effective regularization technique provided appropriate constraints can be obtained [48, 49]. This technique works by constraining the parameter values thereby not allowing them to attain such values that might increase the variance error. These constraints are mostly obtained using the prior knowledge about the underlying processes. However, an expert should have high degree of confidence on his knowledge otherwise this technique may lead to significant increase in the bias error. It is important to distinguish two different types of constraints.

Hard Constraints: These constraints are in the form of either equalities or inequalities. The loss function is minimized in the presence of these constraints. Hence, the parameter estimation problem reduces to a constrained optimization problem. These constraints are called hard constraints because they must be satisfied by the estimated optimal solution. The equality and inequality constraints reduce the flexibility of a fuzzy model in different ways. The equality constraints reduce the degree of freedom of a model by reducing the number of effective parameters. A model loses one degree of freedom for each equality constraint. Therefore, less number of parameters are needed to be estimated from the available training data. On the other hand, the inequality constraints restrict the parameter space such that the final solution is only allowed to be present in a relatively smaller region defined by the constraints. This restriction on the parameter values may have a desired regularization effect on a fuzzy model. It should be noted that defining inequality constraints for premise parameters is fairly easy since they have precise physical meanings. Since the premise parameters define the shape/coverage of membership functions, we can formulate inequality constraints such that certain undesirable situations, like excessive overlapping of adjacent membership functions, can be prevented. On the contrary, defining similar constraints for consequent parameters is a formidable challenge as pointed out in section 3.3.

Soft Constraints: Regularization using soft constraints is a power tool that allows the incorporation of inexact or qualitative knowledge in the fuzzy model. Because of the vagueness of the prior knowledge it is not necessary that the estimated parameter values strictly satisfy the soft constraints. The most common way to realize soft constraints is to add a penalty function, $f_p(.)$, to the loss function as shown in equation 2.19

$$E(\theta, \alpha) = e^T e + \alpha f_p(\theta, \theta_{prior}) \qquad \textbf{(2.19)}$$

The role of this penalty term is to keep the estimated parameter values (θ) closer to the knowledge based parameter values (θ_{prior}) during the loss function minimization. In order to fulfill this role, a penalty function is formulated in such a way that its value sharply increases as the parameter values deviate from the prior values. The non-negative parameter α, in equation 2.19, determines the extent of regularization effect. For $\alpha = 0$, there is no regularization and the problem simply reduces to the unregulated minimization of the mean squared error. On the other extreme, for $\alpha = \infty$ the parameters are purely determined by prior knowledge and the learning from data does not take place. Both of these scenarios are undesirable, hence, a tradeoff is needed to strike a balance between data driven and prior knowledge driven learning. Finding the optimal value of parameter α is not very straight forward. The easiest way is to determine it heuristically by increasing its value, incrementally, until the modeling error becomes smallest on a validation dataset. Other more complex methods have been proposed in [50].

The main difficulty in this technique is to obtain prior knowledge based parameter values (θ_{prior}). Clearly, we are bound to get an inferior model if the θ_{prior} values are not chosen correctly. In the context of TSK fuzzy models, it is fairly easy to get a prior estimate of premise parameters. However, when it comes to the consequent parameters, being the coefficient of the local linear models it is difficult to assign them values using just the qualitative knowledge. This problem is a major hurdle, which prohibits the widespread application of soft constraints based regularization for TSK model parameter estimation.

2.3.4) Regularization by Ridge Regression

The equation 2.18 represents the global least square estimate of the consequent parameters of a linear TSK fuzzy model. We pointed out that the quality of this estimate largely depends on the conditioning of the Hessian (refer section 2.2.2). The Hessian, which is computed as $A^T A$, if ill-conditioned introduces significant variance error in the model. We mentioned that one way to improve the conditioning of Hessian is by having significantly larger number of training data points compared to the number of regressors. This solution, although effective, is not feasible most of the time because we might not have sufficient control over the historically generated training data. In this context, there is another popular technique known as *ridge regression* [51, 52] which works by modifying the loss function by adding a quadratic penalty term as shown in equation 2.20.

$$E(\theta, \alpha) = e^T e + \alpha\theta^2 \tag{2.20}$$

The rationale behind adding a penalty term, $\alpha\theta^2$, is that during the minimization of the loss function, $E(\theta, \alpha)$, all the parameters will be driven towards zero value. However, only those parameters which are truly reducing the loss function attain a non-zero value. As a result of the added penalty term the, least square estimate, changes from equation 2.18 to equation 2.21.

$$\underline{\hat{\theta}} = (A^T A + \alpha I)^{-1} A^T b$$

(2.21)

Here, we observe that the Hessian is modified by the addition of an scaled identity matrix αI. This addition profoundly effects the conditioning of the Hessian. The effect of ridge parameter, α, on the Hessian can be graphically observed in figure 2.7, which plots the contour maps of an arbitrary quadratic loss function given by equation 2.22.

$$E(\theta, \alpha) = 2(\theta_1 - 6)^2 + 3(\theta_2 - 4)^2 + 2.5\theta_1\theta_2 + \alpha(\theta_1^2 + \theta_2^2)$$

(2.22)

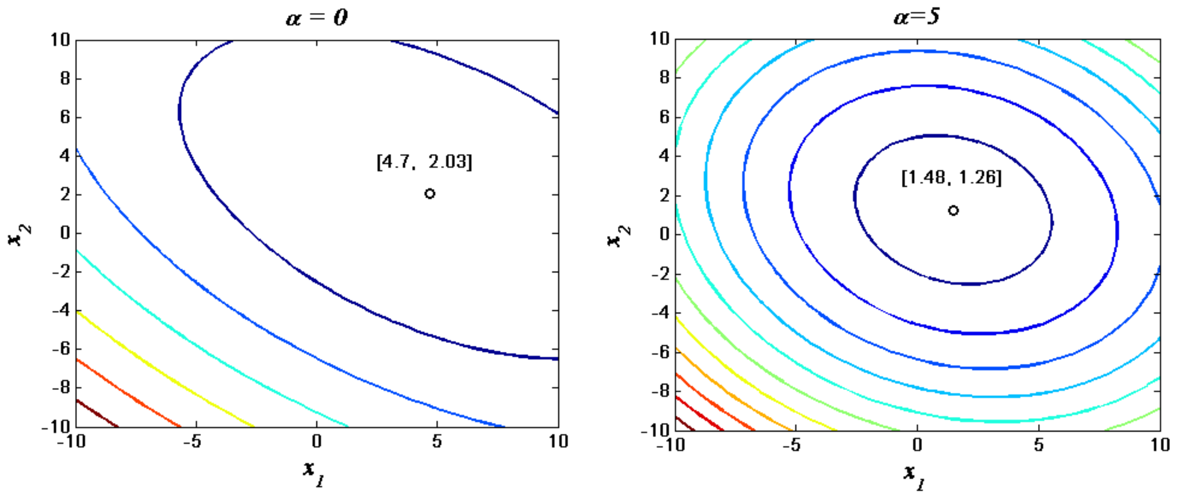

Figure 2-7 Contour maps indicating the effect of ridge parameter α on the Hessian of a quadratic loss function. A higher value of α improves the conditioning of the Hessian thereby reducing the uncertainty in estimated parameter values.

In the first case, when the value of α is zero (no regularization) the contour lines were elliptical suggesting that the Hessian is not relatively well-conditioned. As the value of α is increased to 5, the contour lines become more circular. It should also be noted that, with a non-zero ridge parameter the global minimizer is closer to the origin compared to the case when the ridge parameter was zero. Ridge regression can also be considered as a special case of soft-constraint based regularization, where all the prior parameter values, θ_{prior}, are taken as 0. Therefore, essentially what ridge regression does is convert an over parameterized model to a model with lesser but sufficient number of parameters by driving the superfluous and insignificant parameters to zero.

2.3.5) Regularization by Local Optimization

As briefly mentioned in the previous paragraph that the task of incorporating qualitative domain knowledge in the consequent parameters is a formidable challenge. Therefore, the implementation of soft constraints based regularization for TSK models identification remains questionable. Nevertheless, there are other regularization techniques which are purely computational in nature and do not rely on the availability of qualitative domain knowledge. For example, *Weighted Least Square Estimate* (Weighted/Local LSE) also known as local linear optimization has an excellent regularization effect on TSK fuzzy models [53-55]. In section 2.2.2 we saw, how global linear least square estimate can be used to estimate consequent parameters without considerable computational effort. However, global LSE suffers from high variance error if the quality of training data is inadequate. This drawback can be overcome, to some extent, by using local LSE instead of global. The idea behind local LSE is to identify the consequent parameters of each basis function separately, neglecting the overlap between them. For example, suppose that we wish to estimate the consequent parameters of an arbitrary r^{th} basis function whose local linear model is given by:

$$\theta^r(X) = \theta_0^r + \theta_1^r x_1 + \theta_2^r x_2 \ldots\ldots \theta_n^r x_n \tag{2.23}$$

The approach is similar to global LSE, wherein we estimate all the parameters simultaneously by minimizing the mean squared error with respect to the unknown parameters. However, in local LSE the model output is determined by considering only the local linear model corresponding to the r^{th} basis function, while the existence of other local linear models is neglected. In other words, we do not interpolate between different local models to compute the output of a TSK model. Instead the output is product between the basis function value at the given input and its linear model as shown in equation 2.24.

$$\hat{f}(X^i) = \phi_r(X^i) * (\theta_0^r + \theta_1^r x_1^i + \theta_2^r x_2^i \ldots\ldots \theta_n^r x_n^i) \tag{2.24}$$

Where, X^i is the i^{th} input vector $[x_1^i \; x_2^i \ldots\ldots\ldots x_n^i]$, $\phi_r(X^i)$ is the r^{th} basis function value for i^{th} input and the expression $(\theta_0^r + \theta_1^r x_1^i + \theta_2^r x_2^i \ldots\ldots \theta_n^r x_n^i)$ represents the corresponding local linear model. Using M training data points we can build a system of linear equations given by:

$$W^r \underline{X}\, \theta^r = b$$

(2.25)

Where, W is $M \times M$ weight matrix given by:

$$W^r = \begin{bmatrix} \phi_r(X^1) & 0 & 0 \ldots\ldots 0 \\ 0 & \phi_r(X^2) & 0 \ldots\ldots 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ 0 \quad 0 \quad 0 \cdots\cdots & & \phi_r(X^M) \end{bmatrix}$$

(2.26)

$\underline{X}$ is a $M \times (n+1)$ matrix of inputs values or simply the collection of input vectors:

$$\underline{X} = \begin{bmatrix} 1 & x_1^1 \; x_2^1 \ldots\ldots\ldots x_n^1 \\ 1 & x_1^2 \; x_2^2 \ldots\ldots\ldots x_n^2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 1 & x_1^M \; x_2^M \ldots\ldots\ldots x_n^M \end{bmatrix}$$

(2.27)

$\theta^r = [\theta_o^r \; \theta_1^r \; \theta_2^r \; \ldots\ldots \theta_n^r]^T$ is the vector of unknown consequent parameters of r^{th} basis function and b is a vector of target output given by $[f(X^1) \quad f(X^2) \ldots\ldots f(X^M)]^T$.

The linear system given in equation 2.25 has a straight forward solution given by equation 2.28.

$$\hat{\theta}^r = (\underline{X}^T\, W^r\, \underline{X})^{-1}\, \underline{X}^T W^r\, b$$

(2.28)

In order to estimate the consequent parameter of any other rule using the above equation, we are simply required to recomputed the weight matrix W^r. If a TSK model has total R basis functions, the local LSE requires us to use equation 2.28 R number of times making its computational complexity $O(Rn^3)$. Thus, the complexity of local LSE is linear in terms of number of basis functions which is significantly better than the cubic complexity of global LSE $O(R^3n^3)$ [44]. Hence, in addition to having regularization effect on fuzzy model, the local LSE also requires lesser resources for estimating consequent parameters.

2.4 Problem Statement

In this chapter, we presented a comprehensive review of data-driven models especially focusing on TSK fuzzy models. The need of the regularized consequent parameter estimation in TSK models is highlighted. We discussed various regularization techniques that can significantly reduce the variance error and yield a well behaved model. However, none of these regularization techniques, except the constraint based regularization, makes use of the knowledge that a domain expert possesses. Even the constraint based regularization is flawed because it requires quantification of a domain expert's knowledge which could be purely qualitative in nature. In context of TSK fuzzy models, there does not exist any technique, to the knowledge of the author, that makes use of only the qualitative domain knowledge. Therefore, the goal of this work is to develop a qualitative knowledge based regularization technique for the identification of TSK fuzzy models. The technical problem statement is provided below.

Given a 1st order TSK fuzzy model with n inputs and R basis function, a novel regularization technique is devised for the consequent parameter estimation. The strength of the proposed technique is that it simply needs the qualitative knowledge of an expert and quantifies it to reasonable consequent parameter estimates. The motivation of this work is to identify TSK models that are more robust and complaint with the expert's knowledge even when trained with an inferior quality training data.

Chapter 3: A-Priori Knowledge-based Fuzzy Model (APKFM)

In chapter 2, we discussed the issues related to the formulation and identification of TSK fuzzy models. We emphasized the necessity of regularized parameter estimation of these models owing to their inherent over-parameterization. Without regularization a TSK model tends to get over-trained on training data and loses its approximation capability. Various regularization techniques (Early stopping, Staggered Optimization, Ridge Regression, Local Optimization, Constrained Optimization) were discussed in section 2.3, which have been successfully employed in neuro-fuzzy modeling. We emphasized our special interest in soft-constrained based regularization technique (section 2.3.3) mainly because it provides a powerful tool to incorporate qualitative domain knowledge in a model. At the same time we pointed out a limitation of this technique when using it for consequent parameter estimation of 1st order TSK models. In TSK models, the consequent parameters are the coefficient of the local linear models, which cannot be easily visualized by a domain expert. Therefore, it gets difficult to translate the expert's knowledge into the consequent parameter values, which are needed for the implementation of soft-constraint based regularization. In this chapter, a computationally efficient technique is proposed that tries to overcome this limitation, thus allowing incorporation of qualitative knowledge in TSK models. Henceforth, the proposed TSK models will be referred to as *A Priori Knowledge-based Fuzzy Models* (APKFMs).

The contents of this chapter form the crux of this thesis. We begin with the formulation of 0th order APKFM in section 3.1. In section 3.2, we discuss different methods that can be used for the optimal parameter estimation of APKFM. Finally, in section 4.3 the formulation of 0th order APKFM is extended to 1st order/ Linear APKFM.

3.1 Formulation of 0th order/ Singleton APKFM.

In sections 2.1.3, we presented the general basis function formulation of TSK fuzzy models. The output of a TSK fuzzy model built on the grid partitioned input space with R basis functions can be expressed as:

$$\hat{f}(X) = \sum_{r=1}^{R} P^{r}(X) * \phi_r(X, \theta^{(nl)}) \tag{3.1}$$

where, $P^{r}(X)$ is the local model associated with the r^{th} basis function $\phi_r(X, \theta^{(nl)})$. The basis functions, which define the validity region of each local model, are formed by the tensor product of individual univariate functions (refer figure 2.5). For a 0th order fuzzy model the local models are simply constants, therefore the model output becomes:

$$\hat{f}(X) = \sum_{r=1}^{R} \theta^{r} * \phi_r(X, \theta^{(nl)}) \tag{3.2}$$

This output happens to be linear in consequent parameters (θ^{r}) and nonlinear in premise parameters $\left(\theta^{(nl)}\right)$. When it comes to estimating the optimal parameter values, these disparate parameters are identified sequentially i.e. one set is fixed while the other is updated using an appropriate optimization technique. This sequential optimization is known as *staggered optimization* and its advantages are discussed in section 2.3.2.

The goal of this work is to devise a technique that can incorporate domain knowledge during the identification of consequent parameters (θ^{r}). Such a technique is expected to have a regularization effect preventing consequent parameters from taking values that make a TSK model's behavior inconsistent with the domain knowledge. In an attempt to attain this goal, a modified formulation of 0th order TSK model is proposed, in which the consequent parameters (θ^{r}) are replaced by a function $\varkappa\left(\delta^{(nl)}\right)$. This modified formulation is shown in equation 3.3.

$$\hat{f}(X) = \sum_{r=1}^{R} \varkappa\left(\delta^{(nl)}\right) * \phi_r(X, \theta^{(nl)})$$

(3.3)

The function $\varkappa\left(\delta^{(nl)}\right)$ is named as the *knowledge function* because its formulation is based on the qualitative domain knowledge. The idea is to indirectly compute the values of the consequent parameters using a knowledge function instead of estimating them directly from the data (for example using global LSE). A knowledge function, if properly formulated, ensures that the consequent parameters always remain in accordance with the domain knowledge irrespective of the quality of data used for training. Thus, a knowledge function is intended to have a shielding effect on the consequent parameters preventing them from taking physically infeasible values. Referring back to equation 3.3, we observe that the knowledge function has its own set of nonlinear parameters $\delta^{(nl)}$, which provide it the desired flexibility. These nonlinear parameters are identified during the learning of APKFM. Typically, the number of nonlinear parameters $\delta^{(nl)}$ is considerably less than the number of consequent parameters θ^{r}. Therefore, the proposed formulation has a twofold regularization effect on fuzzy models. Firstly, because of the incorporation of domain knowledge and secondly, due to the reduced dimensionality of the model, which decreases the variance error (refer bias-variance trade off in section 1.2.2).

Apparently the performance of APKFM directly depends on how well the knowledge function is formulated. An ideal knowledge function should be able to encompass the qualitative knowledge and represent it in a parameterized mathematical expression. It should be noted that a knowledge function based on wrong domain knowledge will significantly increase the modeling error by introducing an incorrect bias to the model. Therefore, the underlying assumption of the proposed technique is that we have correct qualitative domain knowledge at our disposal. In proceeding sections, we present the mathematical formulation of a knowledge function. We start with an illustrative example of a simple one-dimensional system in section 3.1.1. Thereafter, in section 3.1.2 we extend the knowledge function formulation from one-dimensional to multidimensional systems.

3.1.1) Knowledge Function Formulation for a 1-Dimensional System:

Consider a static linear system, given by equation 3.4, that we want to approximate using a 0th order APKFM. This equation represents a straight line with a negative slope. Although, this system is simple enough to be modeled using linear regression, it serves as a good example for illustration.

$$f(x) = -4x + 12$$

(3.4)

The only information available apriori is that the *function $f(x)$ decreases as the input x increases*. This qualitative piece of domain knowledge will be used to build the APKFM. The first step in building an APKFM is to partition the input space into grids and define a basis function on every grid. For the current problem, we assign three triangular basis functions and assign linguistic labels to them viz. *low*, *medium* and *high*. Since this is a 1-dimensional system, the input space get partitioned into three regions without the need of tensor product construction. Each region corresponds to a rule, hence the rule-base consists of three rules, namely R^1, $R^2$and R^3 as given below below:

$$\boldsymbol{R^1}: IF\ x\ is\ low\ \ THEN\ f(x)\ \ is\ \theta^1$$
$$\boldsymbol{R^2}: IF\ x\ is\ med. THEN\ f(x)\ is\ \theta^2$$
$$\boldsymbol{R^3}: IF\ x\ is\ high\ THEN\ f(x)\ is\ \theta^3$$

Our aim is to replace the consequent parameters θ^1, θ^2 & θ^3 by a knowledge function $\varkappa\left(\delta^{(nl)}\right)$. Here we would like to introduce the notion of *favorable rule*. The validity region of a favorable rule coincides with that region of the input space where the model output is expected to be more than its neighboring regions. The formulation of knowledge function relies entirely on the location of such favorable rules. A domain expert should have sufficient knowledge to distinguish a favorable rules from others. In the present example, the qualitative domain knowledge that the *function $f(x)$ decreases as the input x increases*, clearly indicates that the rule R^1 is the favorable one. Lets choose the knowledge function $\varkappa\left(\delta^{(nl)}\right)$ simply to be a radial basis function centered at R^1. Such a function would monotonically decrease as we deviate from its center. If the radial basis function is chosen to be a Gaussian function, the knowledge function can be represented as:

$$\varkappa\left(\delta^{(nl)}\right) = a^1 e^{-\left(\frac{x-c^1}{\sigma^1}\right)^2}$$

(3.5)

where $\boldsymbol{c^1}$ is the center of the rule $\boldsymbol{R^1}$ and $\boldsymbol{a^1}$ & $\boldsymbol{\sigma^1}$ are the unknown parameters, $\delta^{(nl)}$, of the knowledge function. To obtain the consequent parameter, θ^r, of a rule we simply substitute x with the coordinates if its center. Therefore,

$$\theta^1 = a^1 e^{-\left(\frac{c^1 - c^1}{\sigma^1}\right)^2} = a^1$$
$$\theta^2 = a^1 e^{-\left(\frac{c^2 - c^1}{\sigma^1}\right)^2}$$
$$\theta^3 = a^1 e^{-\left(\frac{c^3 - c^1}{\sigma^1}\right)^2}$$

(3.6)

The rationale behind defining a knowledge function in this manner is quite intuitive. As we move away from the center of the favorable rule $\boldsymbol{R^1}$ to that of R^2 and R^3, the knowledge function value decreases. This is a desirable scenario because the values of constant local models (θ^r) are obtained from the very same function, as shown in equation 3.6. Hence the constant local models, determined in this manner, will be in total agreement with the qualitative domain knowledge i.e. $\boldsymbol{\theta^1} > \theta^2 > \theta^3$. This concept is further illustrated in figure 3.1.

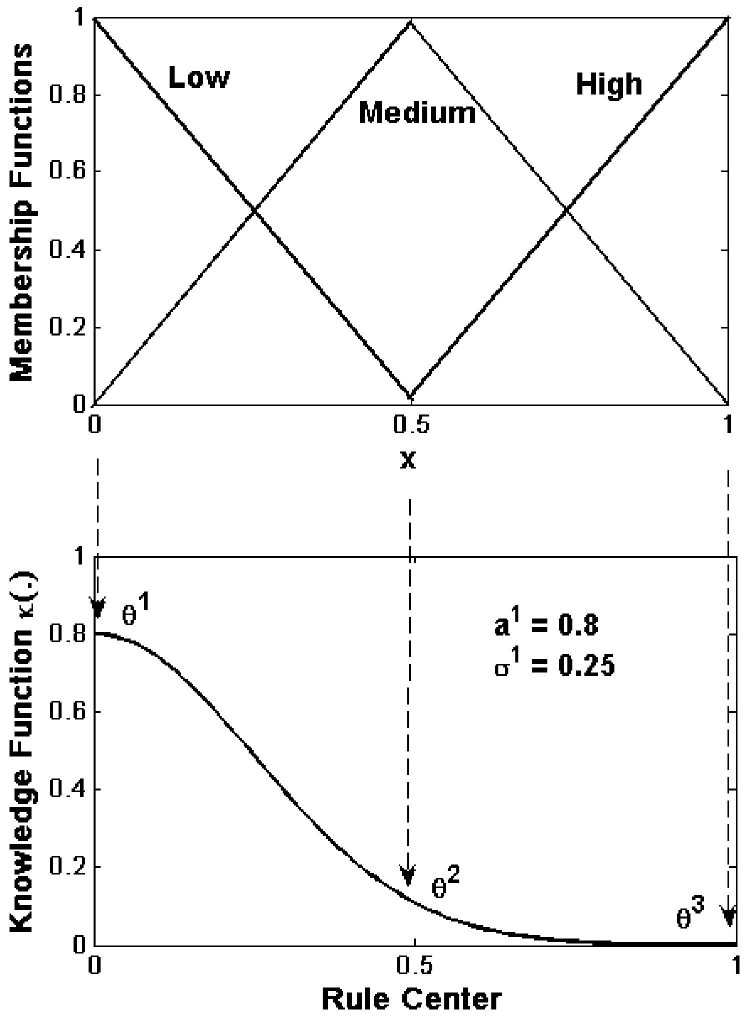

Figure 3-1 Illustration of the knowledge function for a one dimensional system. The knowledge function is a Gaussian function centered at the favorable rule R^1.

Figure 3.1 shows a Gaussian knowledge function centered at the favorable rule R^1 . This function is defined with two parameters $a^1 = 0.8$ and $\sigma^1 = 0.25$. The parameter a^1 determines the height of the Gaussian function while parameter σ^1 controls its spread in the input space. These parameters provide desired flexibility to the knowledge function and are needed to be adjusted to their optimal values during the learning phase. To obtain a consequent parameter value i.e. θ^r, we simply compute the knowledge function value at the center of the corresponding rule (refer equation 3.6). This is also shown graphically in figure 3.1 where the dotted arrows point toward the consequent parameter values of the three rules. It should be mentioned here that other type of radial basis functions, like Bell shaped, Multi-quadric etc., can be also used instead of Gaussian knowledge function provided they differentiable. The requirement of differentiability is needed during the estimation of the nonlinear parameter $\left(\delta^{(nl)}\right)$. However, this requirement becomes redundant if some derivative-free optimization technique is intended to be used.

Above example outlines a simple case when a single radial basis function was sufficient to represent the domain knowledge. However, in some cases there could be more than one favorable rule, which necessitates the use the combination of multiple radial basis functions to build the knowledge function. Let us consider a hypothetical case when the both $\boldsymbol{R}^{\mathbf{1}}$ and $\boldsymbol{R}^3$ happen to be the favorable rules. We first define two Gaussian functions $\psi^{R\mathbf{1}}$ and ψ^{R3} with parameters $(\sigma^{\mathbf{1}}, a^{\mathbf{1}})$ & (σ^3, a^3) and centered at $\boldsymbol{R}^{\mathbf{1}}$ & $\boldsymbol{R}^3$ respectively. Thereafter, the knowledge function can be obtained as the weighted arithmetic mean of $\psi^{R\mathbf{1}}$ and ψ^{R3} i.e.

$$\varkappa(\delta^n) = \gamma_r^{\mathbf{1}}\psi^{R\mathbf{1}} + \gamma_r^3\psi^{R3}$$

(3.7)

The weights in equation 3.7 are computed as follows:

$$\gamma_r^{\mathbf{1}} = \frac{D_r^{R3}}{D_r^{R\mathbf{1}} + D_r^{R3}} \quad \& \ \gamma_r^3 = \frac{D_r^{R\mathbf{1}}}{D_r^{R\mathbf{1}} + D_r^{R3}}$$

(3.8)

Where, $D_r^{R\mathbf{1}}$ and D_r^{R3} are the Euclidean distances of the r^{th} rule center to the centers of $\boldsymbol{R}^{\mathbf{1}}$ and $\boldsymbol{R}^3$ respectively. Equation 3.7 suggests that the cumulative knowledge function is obtained by simply interpolating between two independent radial basis functions. Figure 3.2 draws the resulting knowledge function. It is evident from figure 3.2 that the shape of the cumulative knowledge function relies on the parameters of individual Gaussian functions. The available domain knowledge is used only to decide the location of these Gaussian functions, with their parameters initialized to some random but feasible values. The idea is to estimate the optimal values of these parameters during the supervised learning of the model so that an optimally shaped knowledge function can be obtained.

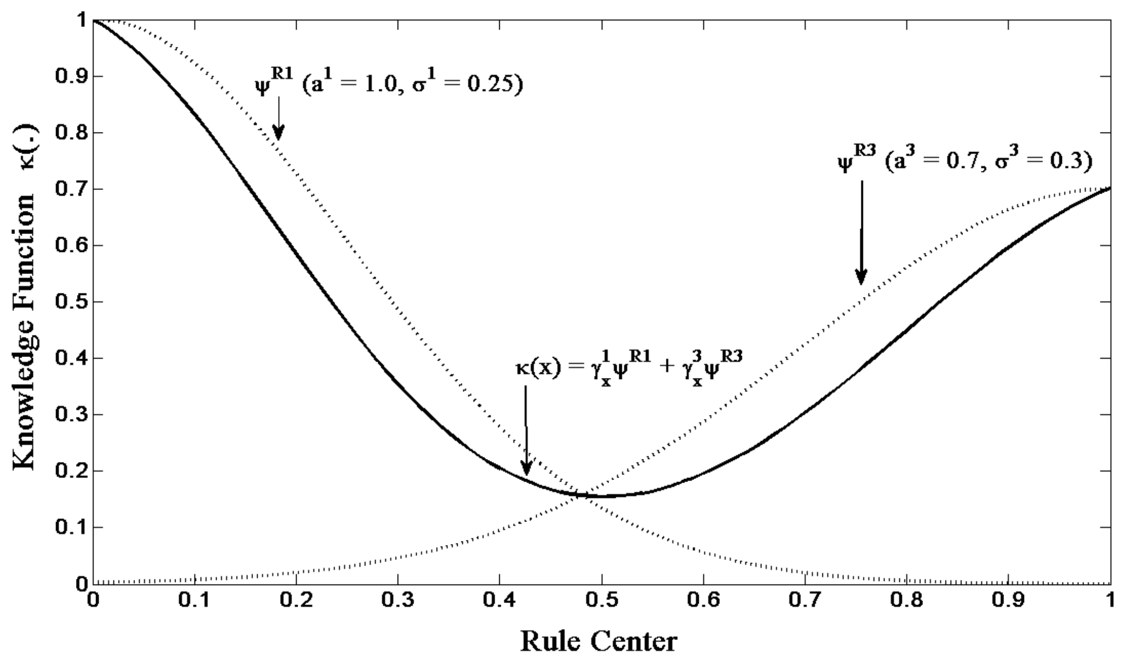

Figure 3-2 Illustration of the knowledge function with two favorable rules (R^1 and R^3). The knowledge function is generated by taking the weighted average of two Gaussian function centered at the two favorable rules.

3.1.2) Formulation of Knowledge Function for Multi Dimensional Systems

Most of the real world problems are multi-dimensional, which necessitates the translation the formulation of knowledge function from 1-dimensional to multidimensional systems. Let us consider a case of a n dimensional system whose input space is grid-partitioned into R regions. Consequently the system possesses R basis functions/rules. Suppose that from the set of R rules we can identify J favorable rules using the available domain knowledge. Let $\boldsymbol{\mathcal{B}}$ be the set of all R rules and $\boldsymbol{\mathcal{F}}$ be the set containing only favorable rules. Clearly, set $\mathcal{F}$ is a subset of $\mathcal{B}$. To generate the knowledge function for this system, we need to define J Gaussian functions centered at corresponding favorable rules. Following the same procedure as described for the 1-dimensional system, the knowledge function can be formulated as the weighted sum of J Gaussian functions i.e.

$$\varkappa\left(\delta^{(nl)}\right) = \sum_{j=1}^{J} \gamma_r^j \, \psi^j$$

(3.9)

where ψ^j is the Gaussian function defined at the center of j^{th} favorable rule having the following representation:

$$\psi^j = a^j e^{-\sum_{i=1}^{n}\left(\frac{c_i^{\mathcal{B}(r)} - c_i^{\mathcal{F}(j)}}{\sigma_i^j}\right)^2}$$

(3.10)

Where, $C^{\mathcal{B}(r)} = (c_1^{\mathcal{B}(r)}, c_2^{\mathcal{B}(r)} \ldots\ldots c_n^{\mathcal{B}(r)})$ and $C^{\mathcal{F}(j)} = (c_1^{\mathcal{F}(j)}, c_2^{\mathcal{F}(j)} \ldots\ldots c_n^{\mathcal{F}(j)})$ are the centers of r^{th} and j^{th} rules present in sets $\mathcal{B}$ and $\mathcal{F}$ respectively.

The weighting factors γ_r^j, in equation 3.9, are determined in the same manner as was done for the 1-dimensional case. A general formula to calculate γ_r^j is given by equation 3.11:

$$\gamma_r^j = \frac{1}{\sum_{i=1}^{J}\left(\frac{D_r^j}{D_r^i}\right)}$$

(3.11)

Where, D_r^j, is the Euclidian distance of the r^{th} rule center to the j^{th} favorable rule center. Clearly, as the r^{th} rule center approaches the j^{th} rule center, γ_r^j tends to 1 and other weighting factors approaches 0. A similar type of weighting factors are used in Fuzzy C-Means algorithm for clustering [56]. The final expression of the 0th order APKFM, as shown in equation 3.12, can be obtained by combining equation 3.3, 3.9, 3.10 and 3.11.

$$\hat{f}(X) = \sum_{r=1}^{R}\left(\sum_{j=1}^{J}\gamma_r^j\, a^j e^{-\sum_{i=1}^{n}\left(\frac{c_i^{\mathcal{B}(r)} - c_i^{\mathcal{F}(j)}}{\sigma_i^j}\right)^2}\right) * \phi_r(X, \theta^{(nl)})$$

(3.12)

Comparing, equation 3.2 with equation 3.12 highlights an important difference between the conventional and the knowledge function based formulation of 0th order TSK models. In equation 3.2 the model output $\hat{f}(X)$ is linear with respect to the consequent parameters (θ^r), hence these parameters can be estimated using linear optimization techniques. Clearly, equation 3.12 does not have these linear parameters because they have been replaced by the knowledge function. The model output is now nonlinear in unknown parameters of the knowledge function a^j and σ_i^j. Hence, the problem of consequent parameter estimation becomes a nonlinear optimization problem, which happens to be computationally more intensive compared to linear optimization. Therefore, the price that we pay for changing the basis function formulation to incorporate domain knowledge, is the increased computational demand. In section 3.2 we discuss some approaches that can be used to estimate these nonlinear parameters. An efficient method is presented in section 3.2.3, which transforms the expression given in equation 3.12 such that it becomes linear in parameters a^j and σ_i^j, thereby allowing the use of linear optimization techniques.

3.1.3) Equivalence between 0th order APKFM and 0th order TSK fuzzy model

Although their formulations appear to be quite different, there is strong link between APKFM and TSK fuzzy model. Consider a special case when no prior domain knowledge is available. In this case, we want to prove that the 0th order APKFM (equation 3.12) is equivalent to the conventional 0th order TSK model (equation 3.2). Since no domain knowledge is available, we cannot identify any favorable rules. This is equivalent to saying that each rule is favorable i.e. $\mathcal{F} = \mathcal{B}$, as a result equation 3.9 reduces to equation 3.13.

$$\varkappa\left(\delta^{(nl)}\right) = \psi^j$$

(3.13)

because, $\gamma_r^j = \begin{cases} 1 \; if \; j = r \\ 0 \; if \; j \neq r \end{cases}$

If we closely examine equations 3.3 and 3.10, we can deduce that the output of APKFM changes from equation 3.12 to 3.14.

$$\hat{f}(X) = \sum_{r=1}^{R} (a^r) * \phi_r(X, \theta^{(nl)})$$

(3.14)

Clearly, equation 3.14 is same as equation 3.2, which completes the proof. Therefore, 0th order TSK models can be considered as the special case of 0th order APKFM in the event of unavailability of any domain knowledge.

3.2 Parameter Estimation of Knowledge function

This section mainly focuses on different optimization schemes that can be applied to estimate the nonlinear consequent parameters, $\delta^{(nl)}$, of a 0th order APKFM. As discussed previously also, the first step in parameter estimation is to define a loss function that can be minimized with respect to the unknown parameters. The most common choice of loss function is the mean squared error, E, as given in equation 3.15. It is assumed that the training dataset consists of M input-output pairs.

$$E = \frac{1}{M}\sum_{m=1}^{M}\left[f(X^m) - \sum_{r=1}^{R}\left(\sum_{j=1}^{J}\gamma_r^j\, a^j e^{-\sum_{i=1}^{n}\left(\frac{c_i^{\mathcal{B}(r)}-c_i^{\mathcal{F}(j)}}{\sigma_i^j}\right)^2}\right)\times \phi_r(X^m,\theta^{(nl)})\right]^2$$

(3.15)

In the above equation, $f(X^m)$ is the target output corresponding to the m^{th}input X^m. The other term in the square brackets is the actual output given the 0th order APKFM given by equation 3.12. Since, the premise parameters, $\theta^{(nl)}$, are assumed to be fixed at known values the only unknown parameters in equation 3.15 are a^j and σ_i^j. Two fundamentally different approaches can be used to minimize this loss functions, each having their pros and cons. In the first approach, we use the information about the 1st/ 2nd order derivatives of the loss function to carry out the minimization. These methods are called *gradient-based* methods. These methods are efficient but have a tendency to get trapped in local minima. The other approach is known as *derivative-free* approach and consists of optimization techniques like Simplex search, Genetic algorithm, Taboo search etc. These techniques have higher computational demand but are far less likely to get stuck in an inferior minimizer. Both of these approaches are dealt briefly in section 3.2.1 and 3.2.2. In section 3.2.3 we present an efficient technique which works by transforming equation 3.12 such that it becomes inherently linear in previously nonlinear consequent parameters a^j and σ_i^j.

3.2.1) Gradient-based Optimization

The simplest way to carry out the minimization of the loss function given in 3.15 is to use some iterative gradient-based optimization techniques. These techniques require less computational effort if the gradients of the loss function can be determined analytically. The general procedure is to start with an initial guess of the solution, and iteratively update this guess until we converge to a local minimizer. The location of initial guess is vital because if not properly chosen, it may lead to an inferior local minimizer or may require higher number of iterations. Nevertheless, these undesirable situations can be avoided by using certain techniques, a comprehensive discussion of which can be found in [57]. The applicability of gradient-based techniques is contingent upon the differentiability of the loss function with respect to the unknown parameters. However, in case of non-differentiable loss functions the gradients can be obtained using finite difference method but at the cost of increased computational demand. Clearly, the loss function given in equation 3.15 has well defined derivatives with respect to parameters a^j and σ_i^j as given in equation 3.16a and 3.16b. Therefore, gradient-based methods are ideally suited for the estimation of these unknown parameters.

$$\frac{\partial E}{\partial a^j} = -\frac{2}{M}\sum_{m=1}^{M}\left\{\sum_{r=1}^{R}\left(f(X^m) - \sum_{r=1}^{R}\sum_{j=1}^{J}\gamma_r^j\, a^j e^{-\sum_{i=1}^{n}\left(\frac{c_i^{\mathcal{B}(r)}-c_i^{\mathcal{F}(j)}}{\sigma_i^j}\right)^2}\right)\right.$$
$$\left. * \left(\sum_{r=1}^{R}\sum_{j=1}^{J}\gamma_r^j\, e^{-\sum_{i=1}^{n}\left(\frac{\left(c_i^{\mathcal{B}(r)}-c_i^{\mathcal{F}(j)}\right)}{\sigma_i^j}\right)^2}\right)\right\}\phi_r{}^2\left(X^m,\theta^{(nl)}\right)$$

(3.16a)

$$\frac{\partial E}{\partial \sigma_i^j} = -\frac{4}{M}\sum_{m=1}^{M}\left\{\sum_{r=1}^{R}\left(f(X^m) - \sum_{r=1}^{R}\sum_{j=1}^{J}\gamma_r^j\, a^j e^{-\sum_{i=1}^{n}\left(\frac{c_i^{\mathcal{B}(r)}-c_i^{\mathcal{F}(j)}}{\sigma_i^j}\right)^2}\right)\right.$$
$$\left. * \left(\sum_{r=1}^{R}\sum_{j=1}^{J}\gamma_r^j\, a^j e^{-\sum_{i=1}^{n}\left(\frac{c_i^{\mathcal{B}(r)}-c_i^{\mathcal{F}(j)}}{\sigma_i^j}\right)^2} * \sum_{i=1}^{n}\frac{\left(c_i^{\mathcal{B}(r)}-c_i^{\mathcal{F}(j)}\right)^2}{\left(\sigma_i^j\right)^3}\right)\right\}\phi_r{}^2\left(X^m,\theta^{(nl)}\right)$$

(3.16b)

The underlying principle governing all gradient-based algorithms is to iteratively move the parameter values in a direction that decreases the loss function. This idea is shown in equation 3.17, where $\delta_{k-1}^{(nl)}$ signifies the parameter values at $(k-1)^{th}$ iteration, which are being updated to $\delta_k^{(nl)}$.

$$\delta_k^{(nl)} = \delta_{k-1}^{(nl)} - \eta_{k-1}R_{k-1}\nabla f_{k-1}$$

(3.17)

The extent of parameter update is controlled by the step size $\eta_{k-1}$and the update direction is always opposite to the gradient direction, ∇f_{k-1} rotated and scaled by some direction matrix R_{k-1}. There are two distinct steps followed in this iterative scheme 1) Determination of direction matrix R_{k-1} and 2) Determination of the optimal step size η_{k-1}. The second step is also known as *Line Search*. Typically, we first compute the direction vector ($R_{k-1}\nabla f_{k-1}$) along which we want to move the current parameter values in order to get closer to the local minimizer. Thereafter, we obtain that step size for which the reduction in loss function is highest along the chosen direction. Line search is important because if the step size is not appropriately chosen it may take numerous iterations to reach the local minimizer. Nearly all gradient-based algorithms follow these steps but they differ in the methods for calculating the step size η_{k-1} and the direction matrix R_{k-1}. For example, if we choose the direction matrix to be an identity matrix, we obtain the well known *Steepest Descent* algorithm, while the choice of the inversed Hessian matrix leads to the *Newton's method*. Similarly, some of the commonly used line search techniques are *Fibonacci search*, *Golden section search*, *Interpolation search* etc. Here we have given a very superficial discussion on gradient-based optimization even though it is an extremely advanced subject. For further reading, a good collection of nonlinear optimization techniques can be found in [57, 58].

3.2.2) Derivative-free Optimization

These methods do not require the computation of derivatives to carry out optimization. Instead, they rely on the evaluation of the loss function at different locations in the parameter space. The main advantage of a derivative-free technique is that the chance of converging to the global minimizer increases significantly. The simplest example is the *Naïve Search* scheme, in which the parameter space is divided into reasonably fine grids and the loss function is evaluated at these grid centers. Thereafter, the grid center with lowest value of loss function is chosen to be the solution. We can easily see that this technique suffers from curse of dimensionality, because if the number of unknown parameter increases, the number of places where the loss function needs to be evaluated increases exponentially. Therefore, in practical problems Naïve Search scheme is rarely used. The computational effort needed in derivative-free techniques can be significantly reduced by employing certain heuristics that limit the search in only those parts of parameter space, which have higher possibility of having the global minimizer. Nevertheless, the computational demand of derivation-free optimization techniques is higher as compared to that of gradient based techniques in general. Some of the powerful techniques which are popular among researchers and practitioners are *Genetic Algorithm*, *Simulated Annealing*, *Tabu Search* etc. [59]. It could be a confusing choice to opt between a gradient-based and derivative-free optimization technique. However, there are certain guidelines that can be followed to make an appropriate selection. Under following conditions a derivative-free optimization technique is likely to yield a better result compared to a gradient-based technique.

1. When the loss function's surface is non-differentiable at many regions such that it is not possible to analytically obtain its gradient. It should be noted that a gradient-based technique can still be used in such a situation by making use of finite difference method to compute the gradients.

2. When some parameters are not continuous but discrete in nature. Discrete parameters are very commonly encountered in practical problems. In such problem, the gradient-based techniques are not applicable. Integer programming is referred to a set of optimization techniques, wherein the parameter can only assume integer values [60].

3. When a global minimizer is strongly desired then the gradient-based techniques cannot be trusted, especially if the loss function's surface is expected to be fraught with local minima. On the other hand, if we solely use a derivative-free technique, in such cases, then the convergence to the global minimizer can be prohibitively slow. A commonly used remedy to this problem is to first use a derivative-free technique to estimate the solution. This solution is likely to be in the vicinity of the global minimizer. Afterwards, we can use the estimated solution as the initial guess for a gradient-based technique and obtain the global minimizer relatively quickly.

In the context of present problem of the estimation of the nonlinear consequent parameters ($\delta^{(nl)}$), we recommend the above mentioned combination of derivative-free and gradient-based optimization. The loss function as given in equation 3.15 can have highly contoured surface, if several locally favorable rules are present. Therefore, it would be logical to adapt a derivative-free optimization, which can lead us to a solution reasonably close the global minimizer. In addition, the loss function also has well defined derivatives which can be analytically calculated as given in equations 3.16a and 3.16b. Therefore, the approximate solution obtained by derivative-free optimization can be further refined using an efficient gradient-based algorithm.

3.2.3) Transformation-based Estimation of nonlinear parameters

So far we discussed two different approaches to estimate the nonlinear parameters of the knowledge function. Since both of them are nonlinear optimization techniques, they suffer from high computational demand. In this section, we propose a different approach which changes the estimation of nonlinear consequent parameters of an APKFM into a linear optimization problem. This approach works by transforming the output of 0th order APKFM (equation 3.12) in a logical manner such that the model output becomes inherently linear in consequent parameters a^j and σ_i^j. On careful examination, we observe that there are two places where we make use of weighted arithmetic mean in equation 3.12. Firstly, during the formulation of knowledge function which happens to be the weighed arithmetic mean of J Gaussian functions. And secondly, during the calculation of model output, which is defined as the weighted arithmetic mean of the local models of different rules. If we transform equation 3.12 by replacing these arithmetic means with geometric means, we obtain the following expression as the model output.

$$\hat{f}(X) = \prod_{r=1}^{R}\left(\prod_{j=1}^{J}\left(a^j e^{-\sum_{i=1}^{n}\left(\frac{c_i^{\mathcal{B}(r)}-c_i^{\mathcal{F}(j)}}{\sigma_i^j}\right)^2}\right)^{\gamma_r^j}\right)^{\phi_r(X,\theta^{(nl)})}$$

(3.18)

This transformation is logically sound because it keeps the physical interpretation of a fuzzy model intact. The only difference is that the model output is calculated using weighted geometric mean instead of arithmetic mean. However, it should be noted that equation 3.18 represents a different system whose optimal parameter values will not be optimal for the system defined by equation 3.12. Nevertheless, because of the similarity of the two systems we can expect their optimal values to be quite close.

One noticeable characteristic of equation 3.18 is that it is intrinsically linear in parameters $\log(a^j)$ and $\frac{1}{(\sigma_i^j)^2}$. This can be shown by taking natural logarithm of both sides resulting in equation 3.19:

$$\log(\hat{f}(X)) = \sum_{r=1}^{R} \phi_r(X, \theta^{(nl)}) \sum_{j=1}^{J} \gamma_r^j \left\{ \log(a^j) - \sum_{i=1}^{n} \frac{\left(c_i^{\mathcal{B}(r)} - c_i^{\mathcal{F}(j)}\right)^2}{(\sigma_i^j)^2} \right\}$$

(3.19)

Rearranging equation 3.19 we obtain:

$$\sum_{j=1}^{J} \left(\sum_{r=1}^{R} \phi_r(X, \theta^{(nl)}) \gamma_r^j \right) \mathbf{log}\,(\boldsymbol{a^j}) - \sum_{i=1}^{n} \sum_{j=1}^{J} \left\{ \sum_{r=1}^{R} \phi_r(X, \theta^{(nl)}) \gamma_r^j \left(c_i^{\mathcal{B}(r)} - c_i^{\mathcal{F}(j)}\right)^2 \right\} \frac{\mathbf{1}}{(\boldsymbol{\sigma_i^j})^2}$$
$$= \log(\hat{f}(X))$$

(3.20)

It can easily be verified that equation 3.20 is linear in parameters $\log(a^j)$ and $\frac{1}{\left(\sigma_i^j\right)^2}$ .This linearity of system enables us to estimate the global minimizer of the least square problem. If M training data points are available then equation 3.20 results in M simultaneous linear equations that can be represented as $A\delta = b$.

$$\begin{bmatrix} a_{1,1} & a_{1,2} \cdots\cdots a_{1,J} & a'_{1,J+1} \cdots\cdots a'_{1,J(n+1)-1} & a'_{1,J(n+1)} \\ a_{2,1} & a_{2,2} \cdots\cdots a_{2,J} & a'_{2,J+1} \cdots\cdots a'_{2,J(n+1)-1} & a'_{2,J(n+1)} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ a_{M,1} & a_{M,2} \cdots\cdots a_{M,J} & a'_{M,J+1} \cdots\cdots a'_{M,J(n+1)-1} & a'_{M,J(n+1)} \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_J \\ \delta'_{J+1} \\ \vdots \\ \delta'_{J(n+1)} \end{bmatrix} = \begin{bmatrix} \log\left(f(X^1)\right) \\ \log\left(f(X^2)\right) \\ \vdots \\ \vdots \\ \vdots \\ \log\left(f(X^M)\right) \end{bmatrix}$$

(3.21)

where, for m = 1 to M:

$$a_{m,j} = \sum_{r=1}^{R} \phi_r(X^m, \theta^{(nl)}) \gamma_{r,m}^j$$

$$a'_{m,J+j} = \sum_{r=1}^{R} \phi_r(X^m, \theta^{(nl)}) \gamma_{r,m}^j \left(c_{i,m}^{\mathcal{B}(r)} - c_{i,m}^{\mathcal{F}(j)}\right)^2$$

$\delta_j = \log(a^j)$

$\delta'_{J+j} = \frac{1}{(\sigma_i^j)^2}$

Where, A is $M \times J(n+1)$ regression matrix, δ is $J(n+1) \times 1$ parameter vector and b is $M \times 1$ target output vector. It is desirable to impose certain constraints on consequent parameters. For example parameter a^j should lie between 0 and 1 because the training data has target output values scaled between 0 and 1. If a^j falls outside this range then we may have some local models with values greater than 1 or less than 0, both of which are unacceptable. Similarly the parameter σ_i^j should be a nonnegative real number since it represents standard deviation. As a result of these constraints the parameters $\log(a^j)$ and $1/(\sigma_i^j)^2$ are bounded as:

$$\left.\begin{array}{c} -\infty < \log(a^j) \le 0 \\ 0 < 1/(\sigma_i^j)^2 < \infty \end{array}\right\}$$

(3.22)

We can easily obtain the optimal solution of the linear system (equation 3.21) with bounds (equation 3.22) using quadratic programming. The loss function to be minimized by quadratic programming is given as:

$$\underset{\delta}{min}: \ \delta^T(A^TA)\delta - (2b^TA)^T\delta$$

(3.23)

If the matrix A^TA is nonsingular then a unique solution exists for the above loss function. A variety of methods are available to perform bounded quadratic minimization [58]. We used the optimization toolbox of MATLAB for this purpose, which uses an algorithm based on the interior-reflective Newton method [61] to estimate the global minimizer.

So far we have discussed three different approaches that may be used for the estimation of consequent parameters of 0th order APKFM. The choice of the estimation method will depend on various factors like available resources, dimensionality of the system, quality of training dataset etc. However, the third method, based on output transformation, appears to be the best choice when dealing with a high dimensional system. This method will consume least amount resources for carrying out the optimization as compared to the first two methods and at the same time will provide reasonable estimates of optimal parameter values. For this reason, we have exclusively used the transformation-based estimation of consequent parameters in our analysis.

3.3 Formulation of 1st order APKFM

In section 3.1 we presented the method of constructing and identifying a 0th order APKFM. We talked about a specialized function called knowledge function that can capture qualitative domain knowledge and represent it mathematically. The next step is to devise a similar technique that can incorporate domain knowledge in 1st order/ linear TSK fuzzy models. Generally, the 1st models have higher flexibility in comparison to 0th order models. Therefore, they are better suited for modeling systems with high complexities. The formulations of 0th order and 1st order TSK models are quiet similar and have a common representation shown in equation 3.1. The only difference between the two exists in the manner the local models $P^r(X)$ are defined. In 0th order models the local models are singletons or constants resulting in the model output expression as shown in equation 3.2. On the contrary, in the 1st order models the local models are linear or 1st order polynomial of the input i.e.

$$P^r(X) = k_0^r + k_1^r x_1 + k_2^r x_2 \ldots \ldots \ldots + k_n^r x_n$$

(3.24)

where $X = (x_1\ x_2 \ldots \ldots \ldots x_n)$ is an input of a n dimensional system. Thus, the output of a 1st order TSK model can be represented by equation 3.25. It should be noted that, irrespective of the order of the model, its output remains linear in consequent parameter (k_0^r, k_1^r etc.).

$$\hat{f}(X) = \sum_{r=1}^{R} (k_0^r + k_1^r x_1 + k_2^r x_2 \ldots \ldots \ldots + k_n^r x_n) * \phi_r\left(X, \theta^{(nl)}\right)$$

(3.25)

There is another way to express the output of a 1st order models. This alternate representation is known as rule-centered form [62], which is shown in equation 3.26.

$$\hat{f}(X) = \sum_{r=1}^{R} \Big(\theta_0^r + \theta_1^r\big(x_1 - c_1^{\mathcal{B}(r)}\big) + \theta_2^r\big(x_2 - c_2^{\mathcal{B}(r)}\big) \ldots\ldots\ldots$$
$$+ \theta_n^r\big(x_n - c_n^{\mathcal{B}(r)}\big)\Big) * \phi_r\big(X, \theta^{(nl)}\big)$$

(3.26)

Where, $C^{\mathcal{B}(r)} = \big(c_1^{\mathcal{B}(r)}, c_2^{\mathcal{B}(r)} \ldots\ldots. c_n^{\mathcal{B}(r)}\big)$ is the center of r^{th}rule. The set $\mathcal{B}$ was defined earlier to be a set containing all the rules in the rule-base. It is evident that the coefficients of the local models as shown in equation 3.25 have a straightforward relationship with the coefficients of rule-centered local models appearing in equation 3.26, i.e.

$$k_i^r = \theta_i^r \ \forall\, i\colon i \in [1, n]$$
$$k_0^r = \theta_0^r - \sum_{i=1}^{n} \theta_i^r c_i^{\mathcal{B}(r)}$$

(3.27)

Although the formulations given in equation 3.25 and 3.26 are equivalent, we are interested in the later because of its enhanced interpretability. In the rule centered formulation of 1st order models, the local models $P^r(X)$ can be considered as the 1st order Taylor series expansion of the underlying nonlinear function about the rule center $C^{\mathcal{B}(r)}$, i.e.

$$\boldsymbol{P^r(X)} = f\big(C^{\mathcal{B}(r)}\big) + \left(\frac{df}{dx_1}\bigg|_{X=C^{\mathcal{B}(r)}}\right)\big(x_1 - c_1^{\mathcal{B}(r)}\big) + \left(\frac{df}{dx_2}\bigg|_{X=C^{\mathcal{B}(r)}}\right)\big(x_2 - c_2^{\mathcal{B}(r)}\big) \ldots$$
$$+ \left(\frac{df}{dx_n}\bigg|_{X=C^{\mathcal{B}(r)}}\right)\big(x_n - c_n^{\mathcal{B}(r)}\big)$$

(3.28)

Comparing equation 3.26 with equation 3.28, we notice that the parameter θ_0^rsignifies the underlying function value at the r^{th} rule center and parameter θ_i^rrepresents the gradient of the function along the i^{th} dimension about the r^{th} rule center. This relationship is shown in equation 3.29.

$$\left.\begin{aligned} \theta_o^r &= f\big(C^{\mathcal{B}(r)}\big) \\ \theta_i^r &= \left(\frac{df}{dx_i}\bigg|_{X=C^{\mathcal{B}(r)}}\right) \ \forall\, i\colon i \in [1, n] \end{aligned}\right\}$$

(3.29)

The advantage of rule-centered form is that if we have some prior information about the values of underlying function, $f(X)$, and its gradient $\nabla f(X)$ at various rule centers then a technique can be devised to incorporate this information in the consequent parameters θ_0^r and θ_i^r. It can be noted that the estimation of parameters θ_0^r is equivalent to the determination of consequent parameters of 0th order APKFM, which has already been treated extensively in sections 3.1 and 3.2. Since, the optimal values of parameters θ_0^r is already known (from the estimation of 0th order APKFM) the equation 3.26 reduces to equation 3.30.

$$\sum_{r=1}^{R}\sum_{i=1}^{n}\theta_i^r\left(x_i - c_i^{\mathcal{B}(r)}\right) * \phi_r\left(X,\theta^{(nl)}\right) = \hat{f}(X) - \sum_{r=1}^{R}\theta_0^r * \phi_r\left(X,\theta^{(nl)}\right)$$

(3.30)

The above expression is obtained simply by shifting the terms containing the known parameters θ_0^r to the right side of the equation 3.26. In equation 3.30, only the parameters θ_i^r are needed to be estimated. Before we continue our discussion on the estimation of these parameters we would like to propose a slight modification in the structure of the 1st order TSK fuzzy model that further enables the incorporation of domain knowledge. The details are given in the next section.

3.3.1) Conversion of 1st order TSK fuzzy model to 1st order APKFM

Here we would like to discuss the structure of 1st order APKFM which results from a slight modification to the structure of conventional 1st order TSK fuzzy model. In a conventional TSK model, the local linear models have n linear parameters $(\theta_1^r, \theta_2^r \ldots\ldots \theta_n^r)$ as shown in equation 3.26. On the contrary, in 1st order APKFM a local model can have as many linear parameters as the number of nearest neighbors of the corresponding rule center. For illustration, let us consider the example of a 1-dimensional system shown in figure 5 in the appendix. Figure 3.5 shows the 1st order TSK model that can be used to model this system. This model consists of 3 rules each having a local linear model defined at their centers. Since, this is 1-dimensional system each local model has one linear parameter associated with it i.e. θ_1^1, θ_1^2 or θ_1^3. As opposed to a TSK model, a 1st order APKFM allows a local model to have as many linear parameters as the number of its nearest neighbors. This difference is highlighted graphically in figure 3.3, which juxtaposes a 1st order APKFM with a 1st order TSK model. It can be observed that, in APKFM the local model corresponding to the second rule has two linear parameters viz. θ_{1-}^{2} and θ_{1+}^{2}. The position of input decides which of one of these linear parameters should be used during the computation of the model output. If the input value lies on the left of the center of the second rule, parameter θ_{1-}^{2} will be used in the local model and vice versa. On the other hand, in 1st order TSK model the linear parameters do not vary with the location of input about the rule centers.

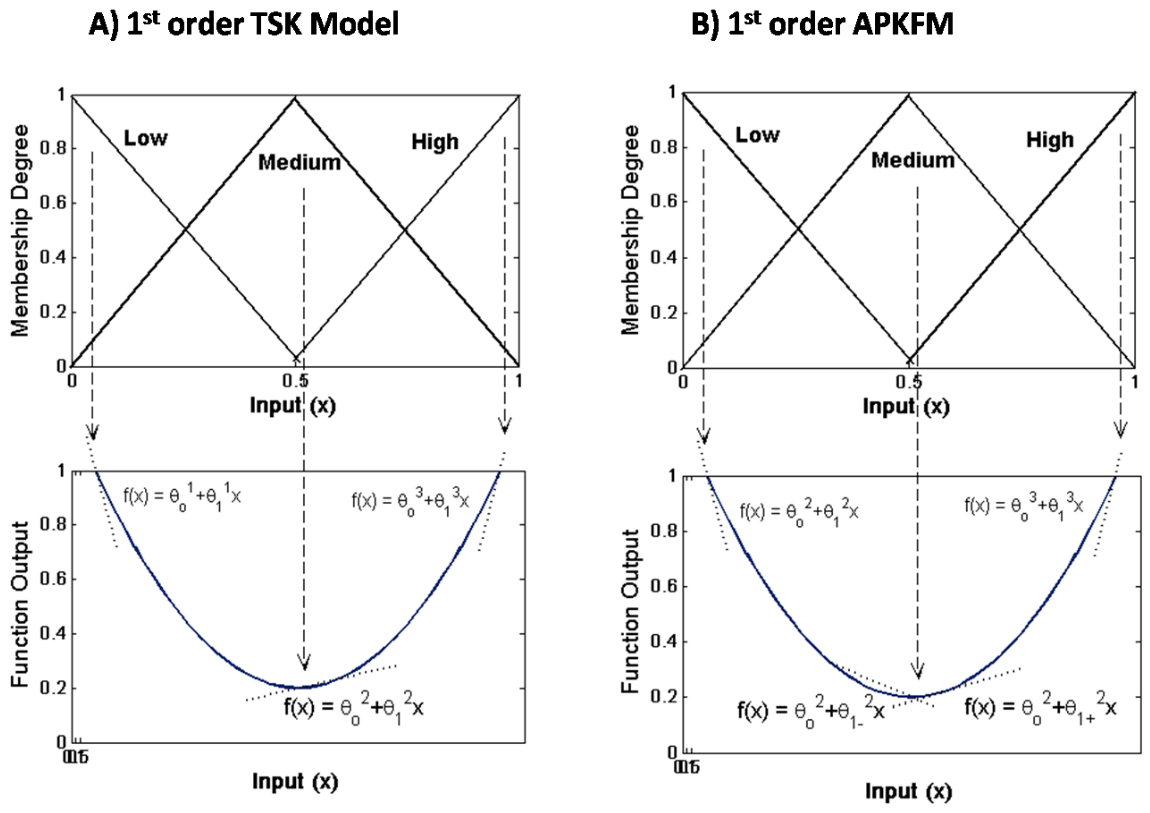

Figure 3-3 (A) **Graphical representation of a 1st order TSK model for the approximation of a 1-Dimensional system. The local model of 2nd rule has only 1 linear parameter i.e. θ_{1}^{2}.** (B) **Graphical representation of the 1st order APKFM for the same system. The local model of 2nd rule has 2 linear parameters viz. θ_{1-}^{2} & θ_{1+}^{2}.**

Consider another example of a 2-Dimensional system with two inputs x_1 and x_2. Figure 3.4 shows its grid partitioned input space with 9 rules whose centers are represented by black dots. We can observe that rule number 5 (the centermost rule) has four nearest neighbors; consequently there are four linear parameters viz. θ_{1-}^{5}, θ_{1+}^{5}, θ_{2-}^{5}, θ_{2+}^{5} associated with it.

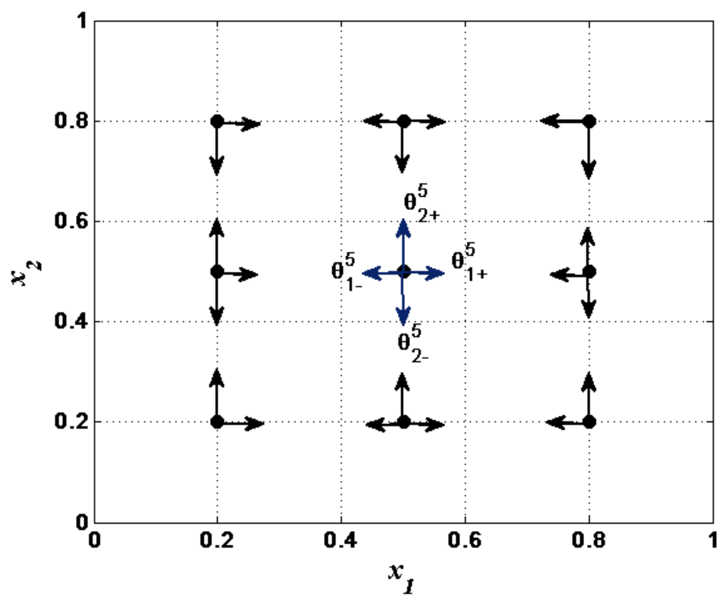

Figure 3-4 A 2-D function's input space with 9 rules and 24 local linear parameters. The black dots represent the locations of rule centers. The arrows point towards the nearest neighbors. Since the centermost rule (rule #5) has two nearest neighbors in each direction, it has four linear parameters viz. θ^5_{1-}, θ^5_{1+}, θ^5_{2-}, θ^5_{2+}.

As mentioned earlier, the position of the input decides which of the two linear parameter i.e. θ^r_{i-} or θ^r_{i+}, should be used during the output calculation. To elaborate this point further, lets refer to figure 3.5 that shows two possible location of an input. When the input corresponds to the location '*a*' then θ^5_{1+} and θ^5_{2+} are used in the output calculation. Whereas, when the input corresponds to location '*b*' then θ^5_{1-} and θ^5_{2+} are used for the same purpose.

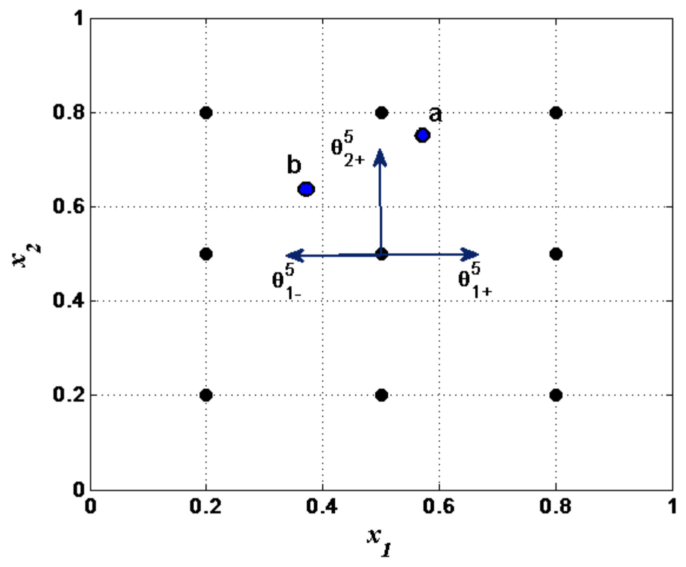

Figure 3-5 Illustrating the effect of the location of an input on the choice of parameters used for describing a local linear model.

It can be verified, visually, from figure 3.4 that there are going to be 24 linear parameters in the APKFM defined for the given 2-dimensional input space. For a generalized system with n dimensions, the total number of linear parameters can be obtained from equation 3.31:

$$\text{Total number of Linear Parameters} = \sum_{i=0}^{n} 2^{n-i} \binom{n}{i} (n+i)$$

(3.31)

The above equation is derived by considering the input space as a n dimensional hyper cube, wherein each dimension is partitioned using three input membership functions (*low*, *medium* and *high*).

3.3.2) Regularized Parameter Estimation of 1st order APKFM

In the previous section we suggested a significant modification in 1st order TSK models. As a result of the this modification the linear equation given in 3.30 changes to the form given in equation 3.32.

$$\sum_{i=1}^{n}\sum_{r=1}^{R}\left\{(\beta_i^r \theta_{i+}^r + (1-\beta_i^r)\theta_{i-}^r) * \left(x_i - c_i^{\mathcal{B}(r)}\right)\right\} * \phi_r\left(X, \theta^{(nl)}\right)$$
$$= \hat{f}(X) - \sum_{r=1}^{R} \theta_0^r * \phi_r\left(X, \theta^{(nl)}\right)$$

(3.32)

where, β_i^r is either 0 or 1 depending on the position of the input with respect to the rule center i.e.

$$\left.\begin{array}{l} \beta_i^r = 1 \; if \; x_i - c_i^{\mathcal{B}(r)} > 0 \\ \beta_i^r = 0 \; if \; x_i - c_i^{\mathcal{B}(r)} < 0 \end{array}\right\}$$

(3.33)

Equation 3.32 is linear in parameters $\theta_{i+}^r/\theta_{i-}^r$, thus if we have a training dataset with M datapoints, we can generate a system of M simultaneous linear equations. Thereafter, the optimal values of these parameters can be easily estimated in a least square sense. However, I propose that this estimation should be regulated by using the soft-constraints based regularization technique (refer section 2.3.3). This regularization technique can be used when the prior estimates of the parameter values i.e. θ_{prior} are available. To realize the soft constraints, a penalty term is added to the mean squared loss function as shown in equation

2.19. The role of this penalty function is to keep the parameter values as close to the θ_{prior} values as possible, during the minimization of the loss function. If the penalty function is chosen to be the one shown in equation 3.34 the overall loss function remains quadratic in unknown parameters θ.

$$f_p(\theta, \theta_{prior}) = (\theta - \theta_{prior})^2$$

(3.34)

As a result, the loss function continues to have a unique global minimizer, which can be easily estimated as:

$$\underline{\hat{\theta}} = (A^T A + \alpha I)^{-1}(A^T b + \alpha\theta_{\ prior})$$

(3.35)

In equation 3.34, α is the regularization parameters that govern the extent of regularization, I is an $n \times n$ identity matrix, A is the regression matrix corresponding to the linear system given by equation 3.32 and b is a vector of size M consisting of target outputs. The i^{th} element of the vector b is given as:

$$b(i) = f(X^i) - \sum_{r=1}^{R} \theta_0^r * \phi_r(X, \theta^{(nl)})$$

(3.36)

Thus, estimating the parameter values using soft constraints based regularization appears to be a fairly simple task. Unfortunately, the difficult part is to obtain the reasonable parameter values based on prior knowledge i.e. $\theta_{\ prior}$ values. As it has been mentioned earlier also, the consequent parameters of TSK fuzzy model cannot be easily visualized by an expert no matter how well he understands the system. This is the primary reason why soft-constraint based regularization cannot be easily applied for the identification of TSK fuzzy models despite having a simple formulation. However, in the rule centered formulation, the parameters θ_i^r can be interpreted as the derivative of $f(X)$ with respect to to x_i at the r^{th} rule center (refer equation 3.29). Hence, the values of parameters $\theta_{i+}^r/\theta_{i-}^r$can be roughly approximated by determining the slope of line joining r^{th} and its neighboring rule centers along the i^{th} dimension. This is illustrated in figure 3.6, which represents the same 2-dimensional system as shown in figures 3.4 and 3.5. The vertical axis plots the θ_0^r values of all the nine rules presumably obtained during the identification of 0th APKFM.

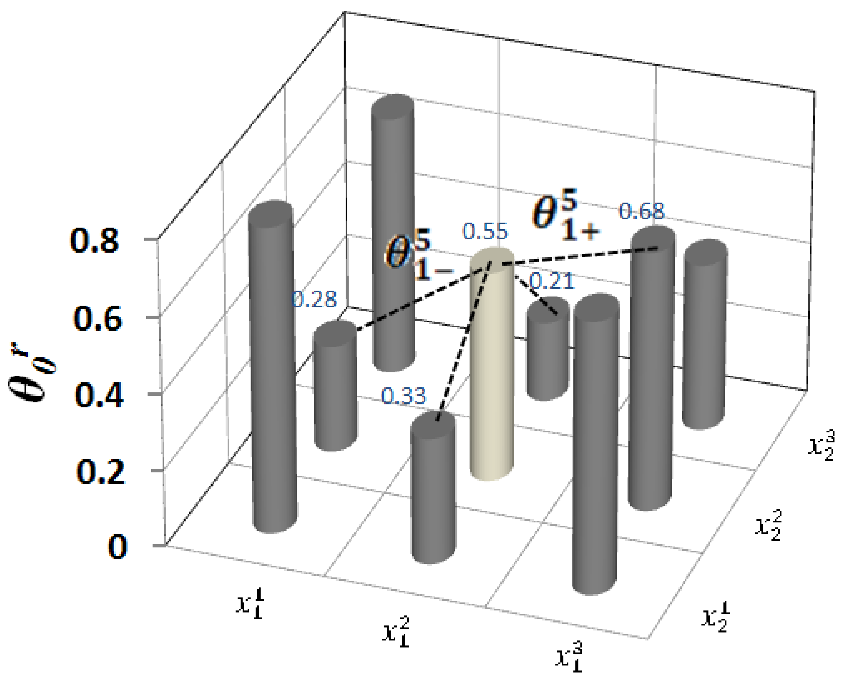

Figure 3-6 Illustrating how the linear parameters on a 1st order APKFM can be estimated from an already identified 0th order APKFM.

Let's say that we want to use the information present in figure 3.6 to estimate the consequent parameter values associated with rule number 5 (the centermost rule). Clearly, because of its center position it has four linear parameters θ_{1-}^5, θ_{1+}^5, θ_{2-}^5, θ_{2+}^5, two along each dimension. The prior estimate of these parameters will simply be the slope of line joining the 5^{th} rule center with its neighboring rules. For example,

$$\theta_{1-}^5 = \frac{\theta_0^5 - \theta_0^4}{c_1^{\mathcal{B}(5)} - c_1^{\mathcal{B}(4)}} = \frac{0.55 - 0.28}{0.5 - 0.25} = 1.08$$

(3.37)

Similarly, the prior knowledge based values of other three parameters can be found as, $\theta_{1+}^5 = 0.52$, $\theta_{2-}^5 = 0.88$ and $\theta_{2+}^5 = -1.36$. The same approach can be used to get the rough estimates of the linear parameters of other rules. This simple technique allows a modeler to obtain reasonable estimates of consequent parameter. Thereafter, these values can be used to obtain the optimal parameter values using equation 3.35.

Before concluding this chapter, it would be worth summarizing whole procedure of the identification of 1st order APKFM in a stepwise manner:

Step 1) We first identify the local model values of 0th order APKFM i.e. θ_0^r values. In this step the qualitative domain knowledge is incorporated into the fuzzy model in the form of a flexible knowledge function. The optimally shaped knowledge function is identified using the training dataset. The entire procedure is outlined in section 3.1 and 3.2.

Step 2) Once we have local model values of 0th order APKFM, we can obtain rough estimates of the linear parameters (θ_i^r) of 1st order APKFM, using the technique described in the previous section. This assignment is based on the observation that the linear parameters of APKFM can be interpreted as the gradients of underlying nonlinear function.

Step 3) Finally, the values of linear parameters obtained in previous step can be considered as the prior knowledge based parameter values i.e. θ_{prior}. These values are used to implement soft constrained based regularization technique so as to obtain the optimal parameter values of 1st order APKFM (refer section 3.3.2).

Chapter 4: Results and Discussion

In chapter 3, we laid the foundation of *A Priori Knowledge-based Fuzzy Models* (APKFMs). We presented detailed formulations of both 0th and 1st order APKFM. The underlying concept, which forms the basis of APKFM, is the notion of *favorable rules*. A favorable rule belongs to that region of the input space where the model output is expected to be more than its neighboring regions. More often than not, an expert is able to identify such rules from a grid partitioned rule-base using his insight about the system. With the help of such favorable rules, a *Knowledge Function* can be constructed, from which the parameters of the local models are determined. This feature of APKFM makes it consistent with the domain knowledge and improves its robustness in the event of training with a poor quality dataset. In addition, the APKFM also has sufficient flexibility that enables it to learn and adapt. This property is desirable because the initial formulation of APKFM is based entirely on the qualitative domain knowledge. As a result, majority of the model parameters are far from being optimal. Therefore, these unknown parameters of APKFM are needed to be identified using a training data. When it comes to the 1st order APKFM, the prior knowledge is incorporated by interpreting the local linear models as the Taylor's series expansion of the underlying nonlinear function about the rules centers. This allows easy computation of the coefficients of linear terms of local linear models from the 0th order APKFM (refer section 3.3). Thereafter, soft constraints based regularization can be implemented to obtain a robust 1st order APKFM.

In this chapter we present an in-depth analysis of 1st order APFKM as a static nonlinear system identification tool. The advantage of APKFM is elucidated by drawing comparisons with three other techniques:

1. A 1st order TSK fuzzy model whose consequent parameters are identified using *Global Least Square Estimate (LSE)*.

2. A 1st order TSK fuzzy model whose consequent parameters are identified using *Local Least Square Estimate (LSE)*.

3. A 1st order TSK fuzzy model whose consequent parameters are identified using *Ridge Regression*.

These techniques are most suitable for comparison with APKFM because they are widely used and represent the state of the art for TSK model identification. The last two techniques are particularly known for their robustness in a variety of situations. The first technique estimates the unknown parameters by simply minimizing the mean squared error in a least square sense (refer section 2.2.2). Since the parameter value are unregulated, this technique is expected to perform poorly with inferior quality training data. Whereas, the second and third techniques perfrom exceedingly well in similar scenarios because the parameter values are regularized by Local LSE and Ridge Regression respectively. For detailed discussions on these techniques refer to sections 2.3.4 and 2.3.5.

This chapter is divided into two parts. In section 4.1, we compare the function approximation capability of APKFM with the other techniques using a 2-dimensional nonlinear toy function. The rationale behind choosing a toy function is that its behavior would be exactly known in a given input domain. This is a desirable scenario when it comes to evaluating and comparing the performances of different fuzzy models. Additionally, we will not be limited by the amount of training data because we can generate as much data as we require from the analytical expression of the nonlinear function. In section 4.2, we deal with a practical problem of cost estimation that has been addressed, previously, using different data-driven modeling techniques. The performances of different models, in both the examples, are adjudged based on two properties 1) the *Prediction Accuracy* on a never seen dataset and 2) the overall *Interpretability* in the input space. An ideal fuzzy model is the one having a good balance between the above mentioned properties.

4.1 2-dimensional Nonlinear Function Approximation

The primary motivation behind the formulation of APKFM is to devise a knowledge driven regularization technique, which can be used to identify robust TSK models for static nonlinear systems. Here we present an illustrative example which highlights the superior function approximation capability of APKFM as compared to other popular techniques. We chose a 2-dimensinal nonlinear function, given by equation 4.1, for this purpose.

$$f(x_1, x_2) = (2x_2 - 0.75{x_1}^2)^2 + 0.5(1 + x_1)^2 + \boldsymbol{\varepsilon} \tag{4.1}$$

The function output $f(x_1, x_2)$ was corrupted by adding a Gaussian noise, $\boldsymbol{\varepsilon}$, with zero mean and some nonzero variance. As mentioned earlier, for benchmarking this nonlinear function was also approximated using three other techniques viz. 1) Global LSE, 2) Local LSE and 3) Ridge Regression. The basic structure of the 1st order TSK models, that were used in the above mentioned techniques, was the same. The domain of each input was partitioned into 3 univariate basis functions with linguistic labels *low*, *medium* and *high*. Thereafter, a grid partitioned rule-base was generated by taking a tensor product of individual basis functions resulting in a total $3 \times 3 = 9$ rules. The conseuquent of each rule had an associatd linear model. Since, this was a a 2-dimensional system, each linear model had three unknown parameters viz. θ_0^r, θ_1^r and θ_2^r. Therefore, the total number of unkown consequent parameters was $9 \times 3 = 27$. The structure of 1st order APKFM was similar except that structure of a rule consequent changed according to the scheme proposed in section 3.3.1. According to this scheme, a rule may have multiple linear models but only one of them is used during calculation of the output. This choice of local model is governed by the location of the input with respect to the rule's center. Table 4.1 lists few properties of 1st order TSK model and 1st order APKFM pertaining to this problem.

Table 4-1 Properties of 1st order TSK model and APKFM that were used for the approximation of 2-dimensional nonlinear function.

	1st Order TSK Model	**1st order APKFM**
Number of fuzzy sets / input	3	3
Membership function type	Gaussian	Gaussian
Number of rules	9	9
Number of Antecedent parameters	12	12
Number of Consequent parameters	27	33

There are two noticeble things in table 4.1. First, the number of consequent parameters for 1st order APKFM is greater than that of 1st order TSK model. The higher number of consequent parameters in APKFM was a direct consequence of the fact that it had more than one local model per rule. As a result, the number of consequent parameters increased marginally. The exact number of parameters was obtained using equation 3.31. Secondly, since I have not proposed any modification to the antacedent structure, the number of associated parameters in the TSK model and the APKFM was the same. Here I would like to mention that, in this comparitive study, the antecedent parameter values are kept constant. Since, the proposed regularization technique is devised only for the consequent parameters it is necessary to isolate their effect on the model output from the effect of antecedent parameters. For this reason, the antecedent parameters are not identified during the training of fuzzy models. Nevertheless, all the techniques which have been conventionally used for antecedent parameter identification in TSK models (for ex. Backpropagation algorithm), are also applicable to the APKFM.

The formulation of APKFM also requires the knowledge of favorable rules, which are identified based on the expert's knowledge about a system. Let's assume that the following domain information was available to us prior to the formulation of APKFM:

System output is <u>High</u> WHEN x_1 is <u>High</u> AND x_2 is <u>Low</u> OR vice versa.

The above statement enabled us to distinguish two favorable rules, whose locations in the input space are highlighted in figure 4.1. Figure 4.1 is the top view of the input space, which has been partitioned into 9 overlapping fuzzy sets. The qualitative information clearly indicated that amongst the 9 rules, rule number 3 and 7 were the favorable ones.

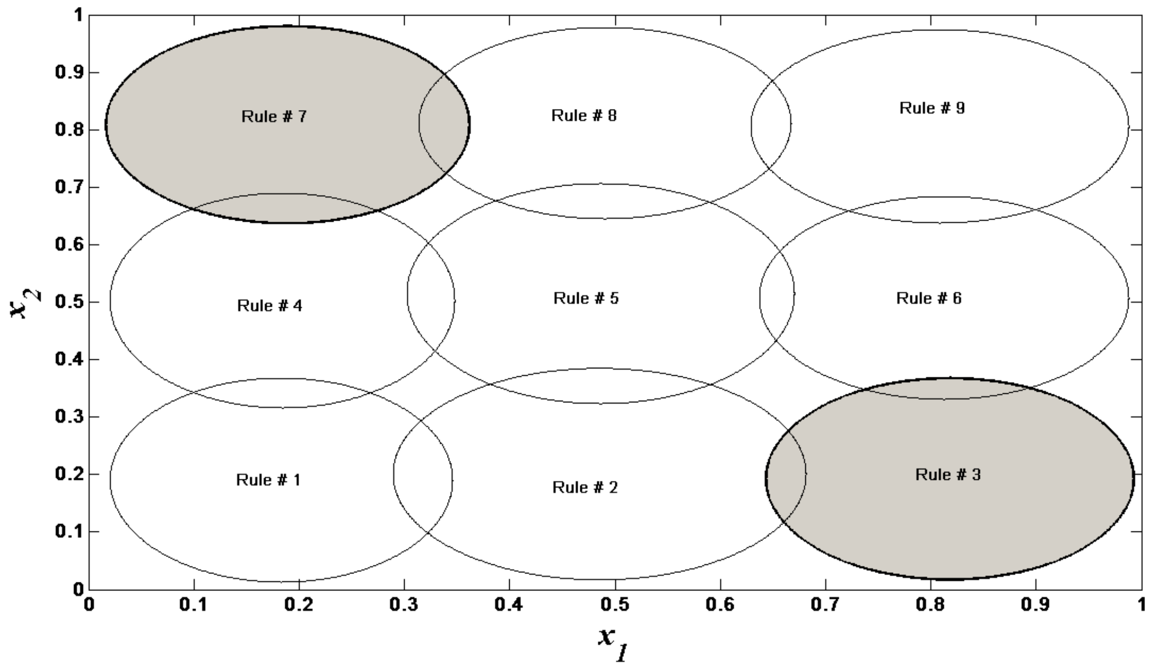

Figure 4-1 The top view of 2-dimensional input space that has been partitioned by 9 overlapping fuzzy sets. The shaded subspaces correspond to the favorable rules.

The input-output data needed for the training of fuzzy models was generated from equation 4.1 by choosing the range of both inputs as [0 3]. The model output is obtained by uniformly sampling the data points from these input ranges. Additionally, the output is corrupted by adding a Gaussian noise to it as shown by symbol $\boldsymbol{\varepsilon}$ in equation 4.1. A noise with high variance yielded an training data with poor singal to noise ratio. Another variable that had a profound effect on the quality of training data was the number of training datapoints. Clearly, a training dataset with fewer datapoints contains less information and therefore is poor in quality. The idea was to generate different training datasets with varying degree of noise level (σ_L) and number of datapoints (M) and use them to compare the performances of competing modeling techniques. Five different standard deviations values were chosen viz. 0, $0.067R$, $0.133R$, $0.2R$ and $0.267R$, where R represents the range of model output (roughly 0-30 in this example). These standard deviation values were used to generate the Gaussian noise. For each noise level we created datasets of four different sizes i.e. with 40, 60, 80 and 100 data points. In this way, we could create training datasets with 20 possible combinations of noise level and size. A dataset with noise level $0.267R$ and size 40 was the worst in terms of information content, while opposite is true for a dataset with noise level 0 and size 100.

To compare the performance of different techniques, I took into consideration two different criteria namely 1) ***Prediction Accuracy*** and 2) ***Interpretability***. Prediction accuracy measures how well a trained model is able to perform on a never-seen/test dataset. It can be quantified using Root Mean Squared Error (RMSE) . A model with low RMSE on test dataset is considered to have a good prediction accuracy. On the other hand, a model's interpretability is a measure of the consistency of a trained model with respect to the domain knowledge. Unfortunately, there is no standard method of measuring the interpretability. However, a common way to evaluate interpretability is to observe the output surface generated by a trained model and look for inconsistent regions. Historically, the fuzzy models have always been graded in terms of their prediction accuracy, while little attention was paid to their interpretability. Only in past few years, the focus has shifted from accuracy to interpretability of fuzzy models. This shift in focus is justified because an uninterpretable fuzzy model defeats the purpose of building a knowledge-based fuzzy model. In sectin 4.1.1, we first compare the prediction accuracies of different techniques while the model interpretabilities are compared in section 4.1.2.

4.1.1) Comparison of Prediction Accuracy

Here we present an indepth analysis of the prediction accuracies of the fuzzy models identified by 1) Global LSE, 2) Local LSE, 3) Ridge Regression and 4) APKFM. The datasets used for training had different combinations of noise level and size. As mentioned earlier, 20 such combinations were possible with $\sigma_L = 0.267R$ & $M = 40$ representing the worst combination and $\sigma_L = 0$ & $M = 100$ representing the best. In figure 4.2, the variation of *mean RMSE* for all the four modeling techniques are plotted against noise level, σ_L. These results are also tabulated in table 4.2. The mean RMSE value was computed by training a fuzzy model with 50 randomly generated datasets with a certain combination of noise level and size and taking the average of RMSEs on a test dataset with 200 data points. It should be mentioned that the test datasets were devoid of any noise because a noise-free dataset would provide a true performance measure of trained models. However, in real world problems we rarely have a noise free dataset to evaluate a model's performance because the training and test data are derived from a common pool of error ridden measurements. For this reason, a known function (equation 4.1) was used to truly compare the performaces of different models. There are several pertinent observations that can be made by closely examining figure 4.2.

1. It is evident from all the four graphs that Global LSE was extremely sensitive towards the noise present in training datasets. As the noise level increased, its mean RMSE value shot up remarkably. However, its sensitivity towards noise diminished as the number of training data points increased. For datasets of size 80 and 100, the increase in mean RMSE value with the noise level was relatively subdued. This behaviour of Global LSE is expected because it did not regularize the parameter values during their estimation. The lack of an explicit regularization scheme allows a fuzzy model to pickup the noise present in the training data. Whereas, the other three techniques showed a significant regularization effect on the model despite having entirely different formulations. The low mean RMSE values at higher noise levels is a clear evidence of their superior robustness as compared to Global LSE method.

2. Amongst the three regularization techniques, APKFM clearly outperformed the other two when dealing with training datasets having a higher noise levels and smaller sizes as evident from the encircled regions in figure 4.1. This observation is a clear indication that the regularization by APKFM is more effective as compared to both local LSE and ridge regression techniques. However, it should be noted that at lower noise levels ($\sigma_L = 0$ and $\sigma_L = 0.067R$) ridge regression consistently performed better than the other techniques, which implies that both Local LSE and APKFM limits the flexibility of fuzzy models to a greater extent than the ridge regression. The limited flexibility of these models increased the bias error, which did not reduce even when the quality of training datasets was improved.

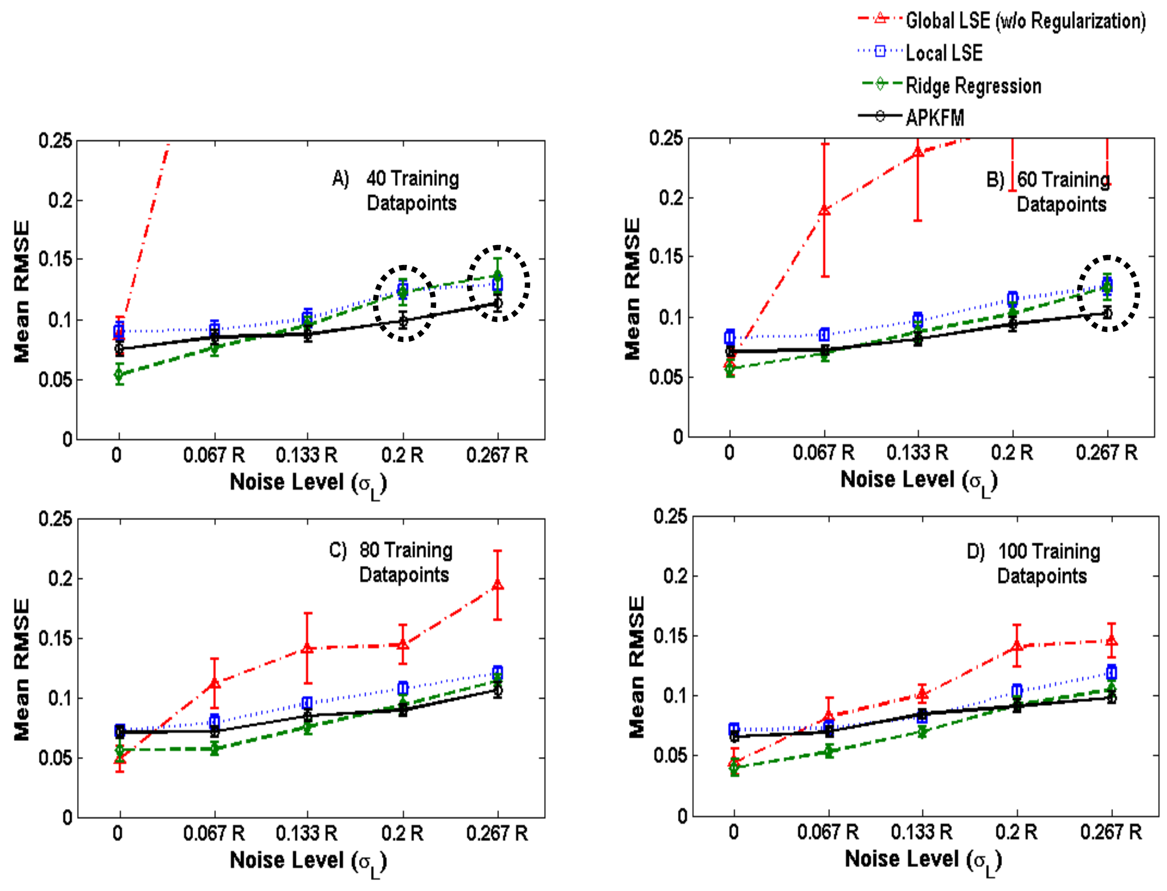

Figure 4-2 Plots of mean RMSE vs. Noise Level for the four techniques of TSK fuzzy model identification. The fuzzy models were trained with the datasets with 4 different sizes viz. 40, 60, 80 and 100.

3. The error bars plotted in figure 4.2 represent the 95% confidence interval of mean RMSE values. Wider error bars signify that the corresponding models have higher variance error. The global LSE technique, understandably, had significantly wider error bars compared to the other techniques. In the absence of regularization, the variance error increased sharply as the quality of training data deteriorated. From figure 4.2, it is difficult to comment on the variance errors of other three identification techniques in comparison to each other. For this purpose, in figure 4.3, we plot the width of 95 % CI of mean RMSE values vs. the noise levels for size 40 training datasets. It can be noted that irrespective of the noise level APKFM resulted in a fuzzy model with lowest variance error. Also, fuzzy models identified by local LSE performed better than the ones identified by ridge regression in this regard. Nevertheless, the variance error of all the modeling techniques decreased as the quality of training data improved.

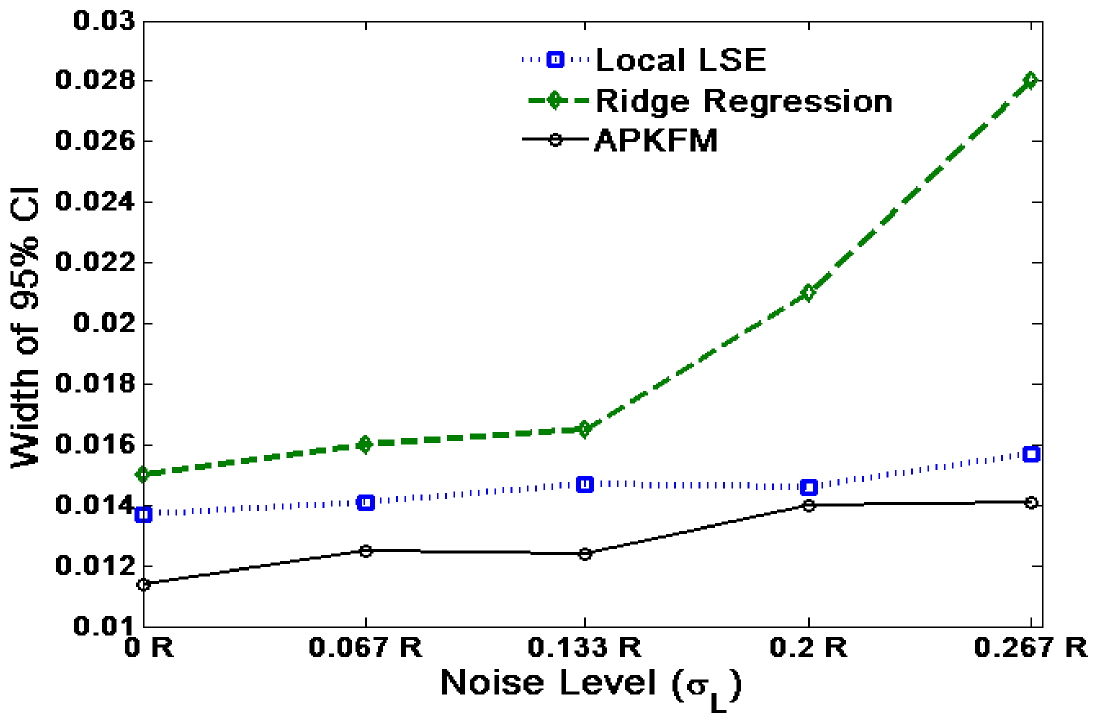

Figure 4-3 Variation of the width of 95% CI of mean RMSE with the increasing noise levels for the three competing regularization techniques. The size of training dataset is fixed at 40.

Table 4-2 Comparison of Root Mean Square Errors (RMSEs) of different models trained on datasets of different sizes (40, 60, 80, 100) and noise levels ($\sigma_L = 0.0,\ 0.067R,\ 0.133R,\ 0.20R,\ 0.267R$). The highlighted rows indicate the cases when APKFM had significantly lower RMSE compared to the other three techniques.

	Global LSE	Ridge Regression	Local LSE	APKFM
Dataset size=40				
σ_L = 0.0	0.0864	0.054	0.089	0.075
σ_L = 0.067R	0.396	0.076	0.091	0.084
σ_L = 0.133R	0.468	0.095	0.10	0.087
σ_L = 0.2R	**0.681**	**0.122**	**0.124**	**0.099**
σ_L = 0.267R	**0.797**	**0.137**	**0.129**	**0.113**
Dataset size=60				
σ_L = 0.0	0.062	0.056	0.082	0.074
σ_L = 0.067R	0.189	0.069	0.085	0.072
σ_L = 0.133R	0.237	0.087	0.096	0.081
σ_L = 0.2R	0.259	0.102	0.114	0.093

σ_L = 0.267R	0.313	0.125	0.125	0.103
Dataset size=80				
σ_L = 0.0	0.048	0.068	0.072	0.071
σ_L = 0.067R	0.112	0.057	0.079	0.071
σ_L = 0.133R	0.141	0.075	0.095	0.085
σ_L = 0.2R	0.144	0.094	0.107	0.081
σ_L = 0.267R	0.196	0.144	0.120	0.106
Dataset size=100				
σ_L=0	0.045	0.039	0.071	0.066
σ_L = 0.067R	0.082	0.053	0.073	0.069
σ_L = 0.133R	0.101	0.07	0.082	0.084
σ_L = 0.2R	0.141	0.092	0.103	0.091
σ_L = 0.267R	0.146	0.105	0.119	0.098

4.1.2) Comparison of Interpretability

In the preceding section, we highlighted the strength of APKFM and showed that the regularization based on prior knowledge is more effective as compared to local LSE and ridge regression in terms of prediction accuracy. However, another important property that is desired in an ideal fuzzy model is its interpretability. Interpretability is a desirable property because it improves the transparency of a fuzzy model. It enables a user to understand how the model output is being calculated. Accuracy and interpretability are known to be two conflicting properties and it is well established that we cannot attain the best accuracy and the best interpretability simultaneously [63-67]. Therefore, an ideal fuzzy model is the one that balances the tradeoff between accuracy and interpretability.

In this section I intend to compare the interpretability of the same four techniques that were dealt in the previous section. As we mentioned earlier, there is no standard measure of the interpretability of a system and a common way to do it is by observing the output surface of the final model. In figure 4.4, I plot the output surfaces of TSK models identified using global LSE. Figure 5.4A shows the actual surface of the 2-Dimensional system given by equation 4.1. The subsequent plots correspond to the models trained with increasingly erroneous datasets. The size of training datasets was fixed at 60. When the training dataset was error-free (figure 4.4B) the global LSE generated an output surface with an excellent resemblance with the actual surface. However, as the data became noisier I started to see some imperfections on the output surface (see encircled regions). These imperfections were in the form of some crests and troughs, which were clearly manifested by the over-fitting on erroneous training datasets. Understandably, the resemblance of output surface with the actual surface became poorer as the noise level increased further.

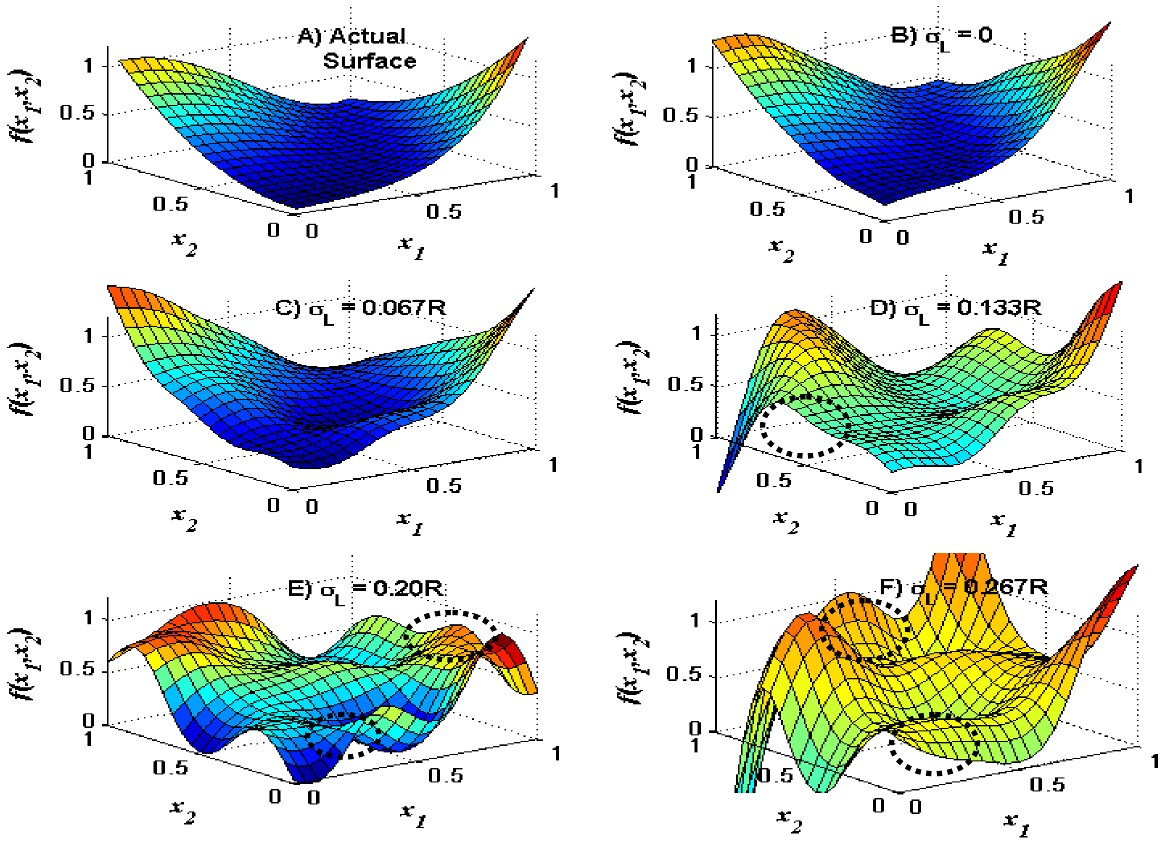

Figure 4-4 A) **The output surface corresponding to the nonlinear function given in equation 4.1.** B-F) **The output surface generated by fuzzy models identifed using *Global LSE* and training datasets with varying noise levels. The encircled regions represent some of the places where model output is incosistent with the domain knowledge.**

Figure 4.5 shows similar plots for the TSK models identified using ridge regression. It should be noted that, the datasets used for training are identical for all the techniques so that we can draw fair comparison between them. Clearly, the output surface in case of ridge regression improved markedly as compared to the global LSE. Even for higher noise levels the surface was well behaved and consistent with the domain knowledge. This observation is expected because ridge regression or, as a matter fact, any regularization technique renders a model less flexible thereby significantly reduces its over-fitting on training data. However, for very high noise levels $\sigma_L = 0.2R$ & $\sigma_L = 0.267R$, we could still observe certain input subspaces where the model behavior was not in accordance with the domain knowledge. Such regions are encircled in figure 4.5E and 4.5F.

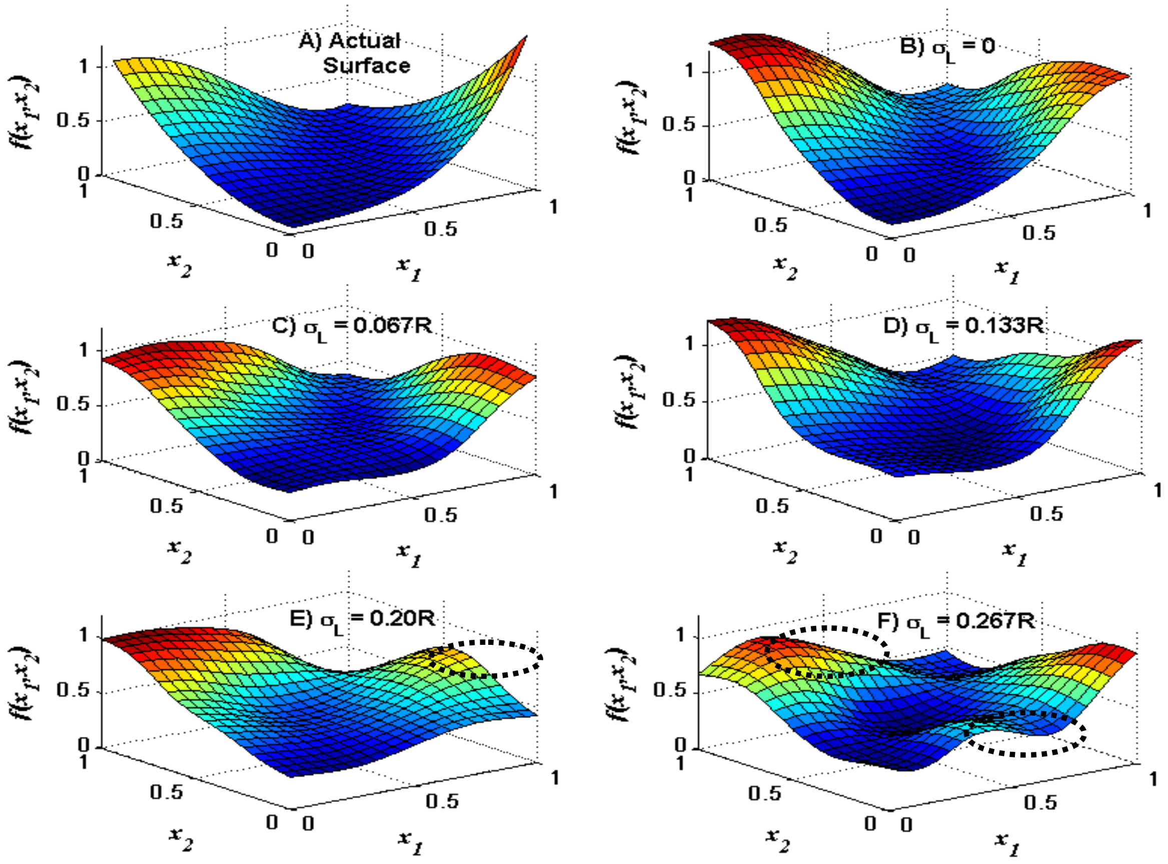

Figure 4-5 A) **The output surface corresponding to the nonlinear function given in equation 4.1.** B-F) **The output surface generated by fuzzy models identifed using *Ridge Regression* and training datasets with varying noise levels. The encircled regions represent places where model output is incosistent with the domain knowledge.**

Figure 4.6 and 4.7 show the similar plots for the TSK models trained with local LSE and APKFM. Both of them did an excellent job in terms of keeping the interpretability of the model intact even at high noise levels. The output surfaces were smooth and remained consistent with the domain knowledge in the entire input space. The advantage of APKFM over local LSE in terms of the interpretability of the output surface is not quite apparent in this example. However, the problem discussed in section 4.2 distinctly brings out the strength of APKFM over its competing regularization techniques.

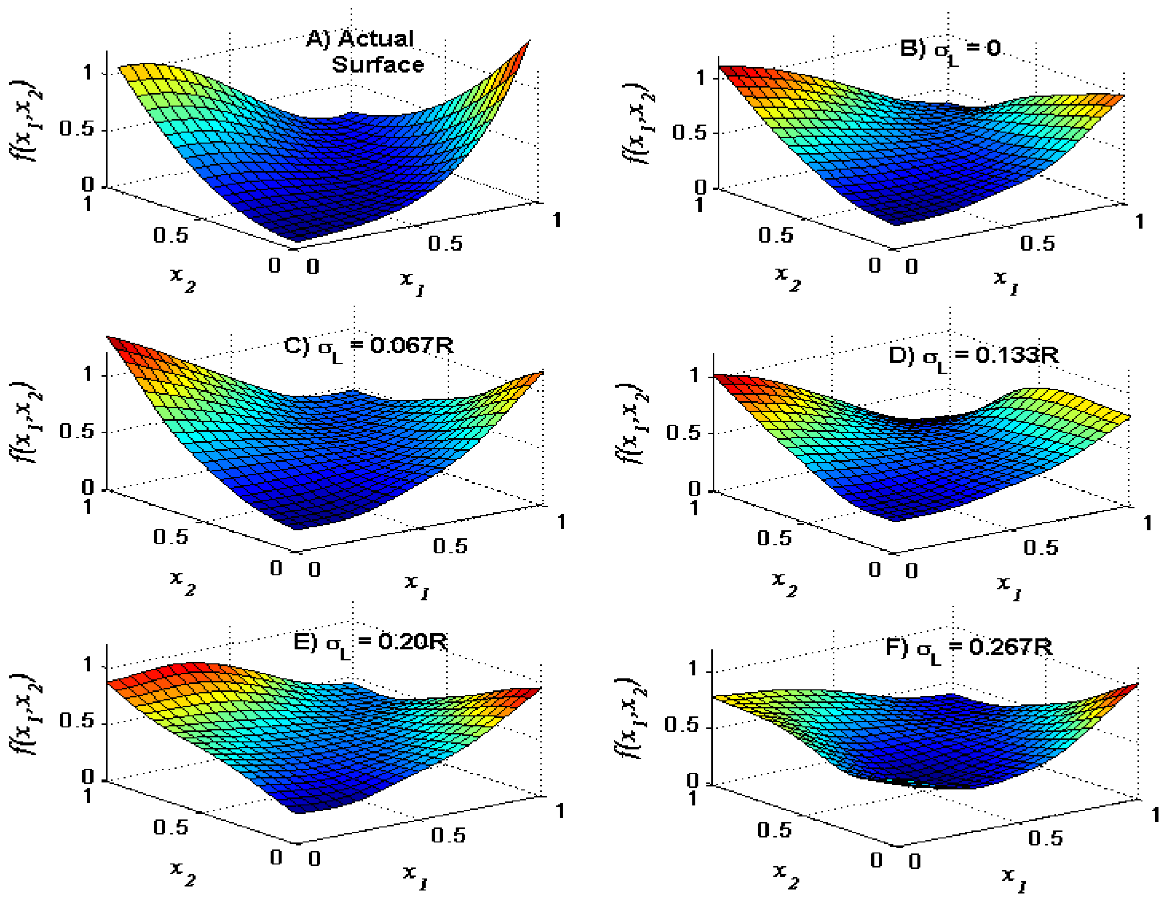

Figure 4-6 A) **The output surface corresponding to the nonlinear function given in equation 4.1.** B-F) **The output surface generated by fuzzy models identifed using *Local LSE* and training datasets with varying noise levels.**

In conclusion, the comprehensive study of a 2-dimensional nonlinear function proves the distinct advantages of APKFM over other regularization techniques in terms of robustness. When trained with low quality data, the prediction accuracy of APKFM was significantly better than that of local LSE and ridge regression regularization techniques. Additionally, both APKFM and local LSE techniques yielded more interpretable models as compared to ridge regression.

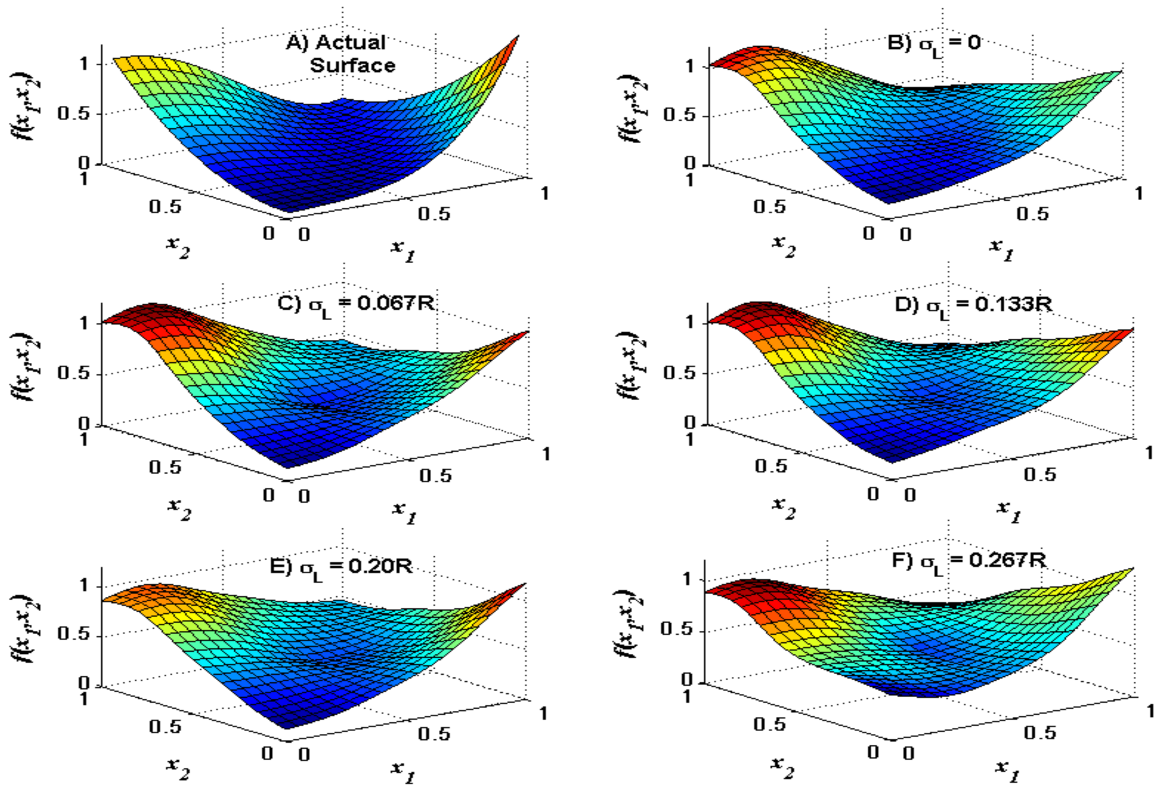

Figure 4-7 A) **The output surface corresponding to the nonlinear function given in equation 4.1.** B-F) **The output surface generated by *APKFM* and training datasets with varying noise levels.**

4.2 Cost Estimation Models for the Electricity Distribution Networks

To further corroborate the advantage of APKFM in a real world scenario, we picked up a practical problem requiring estimation of the maintenance cost of the electricity distribution networks in Spanish towns. The government of Spain allocates funds to several companies to maintain the medium and low voltage electric lines. Therefore, it is important for the government to be able to come up with reasonable estimates of the maintenance cost of electricity networks in different towns. These estimates are required for the appropriate allocation of funds to different companies. Over the years, the maintenance cost was estimated based on the exact measurement of the total length of the distribution lines. This method provided excellent estimates but suffered from the requirement of extensive manpower. Moreover, especially in developing towns, several kilometers of lines are added, every year, in a highly convoluted manner, which makes the task of measuring their exact lengths quite difficult. Therefore, an indirect method is needed which could provide reasonable estimates without being overly exorbitant. An approach proposed by Cordon et al

[68] proved to be quite effective and highly economical in this regard. In this approach, the maintenance cost was estimated from fours easily measurable characteristic of a town. The idea was to develop a data driven model that takes these four characteristics as inputs and computes the maintenance cost as the output. These four town characteristics are listed in table 4.3.

Table 4-3 Description of different inputs considered in the cost estimation model

Symbol	Input Description
x_1	Sum of the lengths of streets in the town
x_2	Total area of the town
x_3	Area Occupied by buildings
x_4	Energy supply to the town

The data was available in the form of these four characteristics of a town and the corresponding maintenance cost as estimated from the conventional method based on the direct measurement of the distribution network's length. In [68] this problem is modeled by different techniques viz. linear regression, 2nd order polynomial regression, neural networks, Mamdami fuzzy model, TSK fuzzy model etc. Out of these techniques, it is shown that the TSK model provided the best prediction accuracy.

In this analysis, I used the same dataset as used in the above mentioned study and analyzed the performances of 1st order TSK model identified by Global LSE, Local LSE and Ridge regression. At the same time, an APKFM was trained in order to demonstrate its superior interpretability compared to the above mentioned techniques.

4.2.1) Construction of Fuzzy Models

The structures of TSK model and APKFM was built in the same manner as was done in the previous example of nonlinear function approximation. Each input domain was divided into 3 triangular univariate basis functions with linguistic labels *low*, *medium* and *high*. Eighty one multivariate basis functions were formed by taking the tensor product of these univariate fuzzy sets. The APKFM's structure was similar to that of TSK model except for the modified linear models that were used in rule consequents. Table 4.4 lists the important properties of these models.

Table 4-4 Properties of 1st order TSK model and APKFM that were used for building the cost estimation models.

	1st Order TSK Model	1st order APKFM
Number of fuzzy sets / input	3	3
Membership function type	Triangular	Triangular
Number of rules	81	81
Number of Antecedent parameters	36	36
Number of Consequent parameters	405	437

Additionally, for the formulation of APKFM, we needed some qualitative information that could help us identify favorable rules. Let that qualitative information be in the form of following statement:

"*The model output increases as the inputs* x_1, x_2, x_3 *and* x_4 *increase*"

The validity of this statement can be easily verified if we look at the description of our model inputs as given in table 4.3. Clearly, we can expect the maintenance cost of the distribution network to rise if any of these inputs value increases. This qualitative information enabled us to identify the most favorable region of the input space having the following linguistic representation:

$$x_1 \text{ } is\ high\ AND\ x_2\ is\ high\ AND\ x_3\ is\ high\ and\ x_4\ is\ high$$

This region of the input space coincides with the validity region of the rule number 81. Thereafter, following the methodology presented in chapter 3, a 1st order APKFM was built.

4.3.2) Comparison of Fuzzy Models

The training of all the models was carried out using 832 randomly chosen data samples. A separate set containing 210 data samples was used for the validation of trained models. Again the performance of different models were adjudged based on two criteria viz. *accuracy* and *interpretability*. The prediction accuracies are compared in Figure 4.8, which plots the actual maintenance cost vs. predicted maintenance cost for the four techniques.

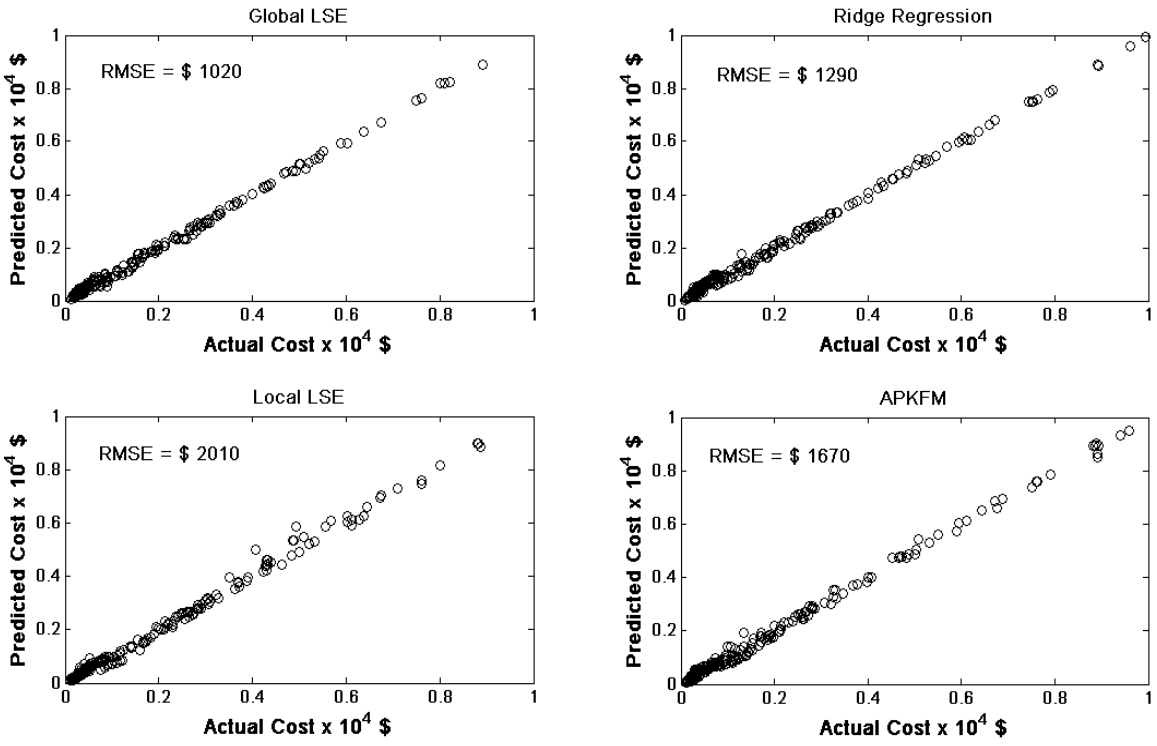

Figure 4-8 Plots of predicted maintenance cost vs. actual maintenance cost resulted from four different fuzzy modeling techniques. The root mean squared error (RMSE) denotes the error in prediction.

Clearly, all the techniques yielded models with excellent accuracy. The TSK model identified by global LSE had the lowest RMSE of \$ 1020 on a test dataset, while the model identified by local LSE was least accurate with an RMSE of \$ 2010. The APKFM prediction accuracy was acceptable at \$ 1670, which is not far behind the other techniques. However, the main strength of APKFM proved to be its superior interpretability in the entire input space. This is illustrated in figures 4.9 and 4.10, which compares the output surface generated by the four techniques. In figure 4.9, the inputs x_2 and x_3 were fixed at a value of 0.5, while the surfaces were generated by spanning the remaining two inputs in the entire input space. One thing that stands out in figure 4.9 is the effect of regularization on the output surface. All the three regularization techniques resulted in surfaces which were much smoother compared to the one generated by unregularized global LSE technique. This is a desirable effect of regularization because most of the real world process show smooth transitions from one state to another and the output seldom changes abruptly with the change in the input conditions.

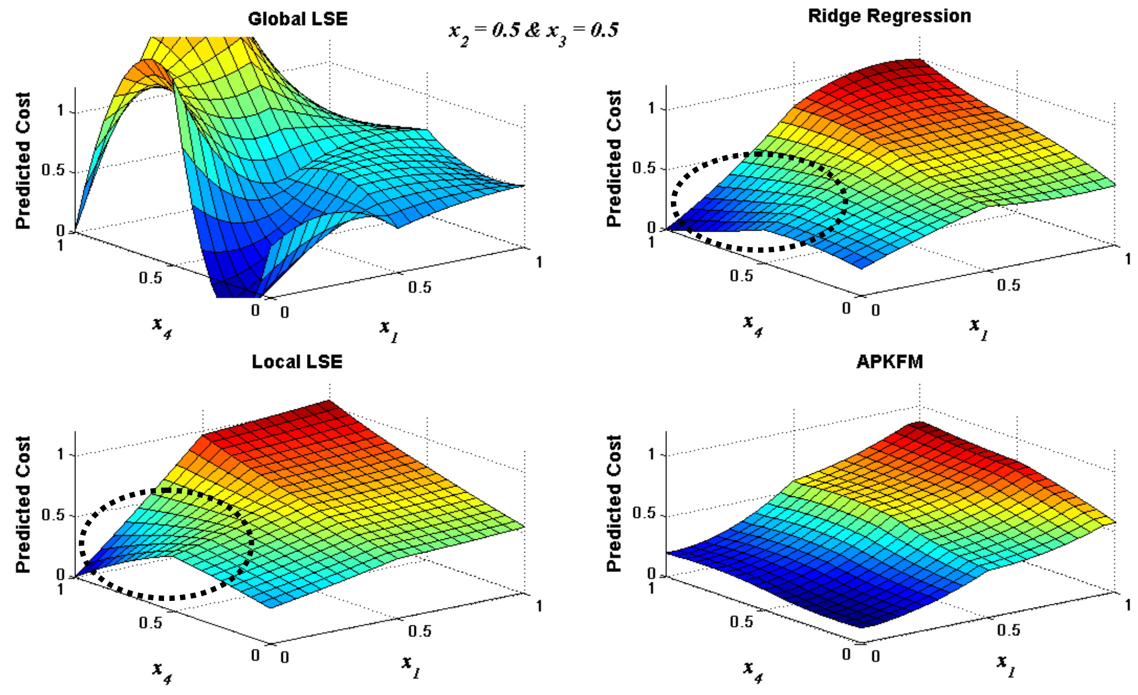

Figure 4-9 Output surfaces generated by different models after training with 832 data samples. The inputs x_2 and x_3 are fixed at a value of 0.5 while x_1 and x_4 were varied to generate these surfaces. The encircled regions signify that the output is not consistent with the domain knowledge.

Among the three regularization techniques, APKFM resulted in an output surface, which was most consistent with domain knowledge. If we observe the encircled regions in figure 4.9, we see that the output is decreasing with the increase in the input x_4. These regions belong to the surfaces generated by local LSE and ridge regression techniques. According to the domain knowledge the model output should never be decreasing with the input x_4 because it would suggest that we require less resources to maintain a town that has a higher energy supply to it. Therefore, despite having smoother surfaces the model behavior was inconsistent with the domain knowledge for local LSE and ridge regression techniques. However, this inconsistency was not shown by APKFM as evident from its output surface. Figure 4.10 further bolsters this point by juxtaposing the output surfaces generated by varying inputs x_2 and x_3. The inputs x_1 and x_4 were fixed at values 0.7 and 0.5 respectively. Even in this case, the output surface of APKFM turned out to be more consistent with the domain knowledge as compared to the other techniques. There were certain regions in the surface generated by ridge regression and local LSE, that out-rightly negated the domain knowledge.

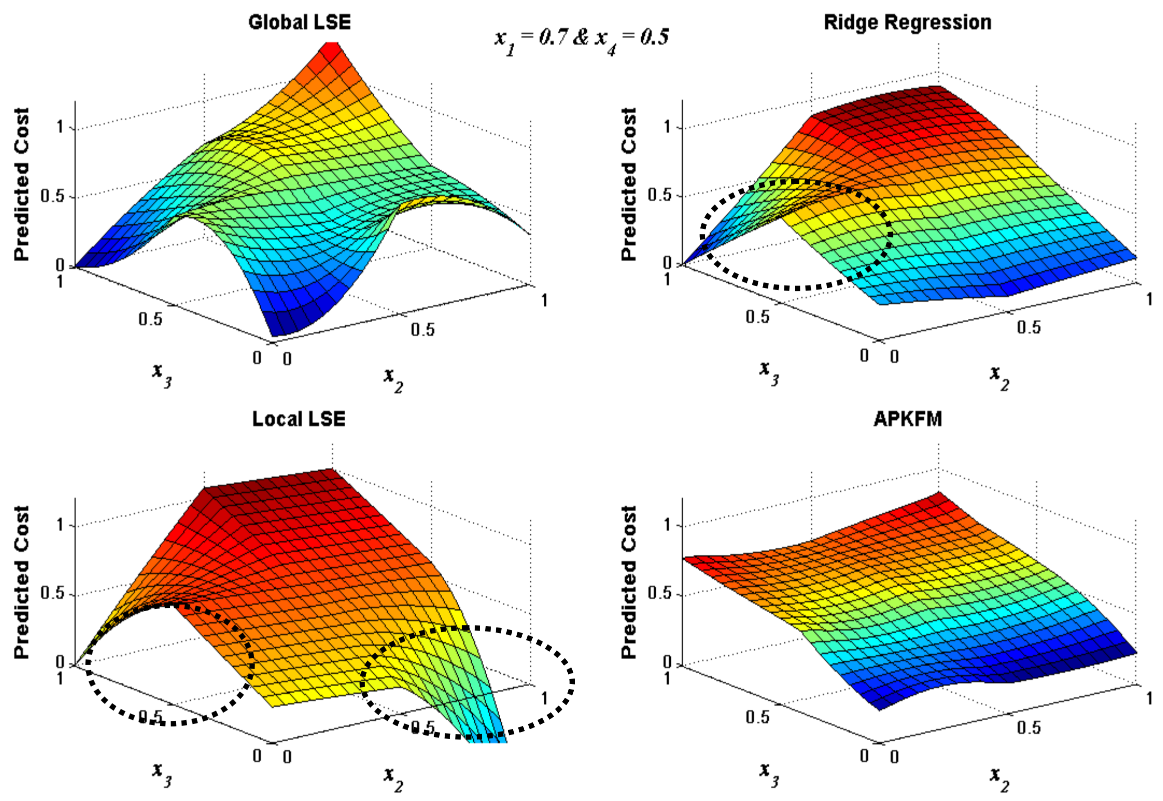

Figure 4-10 Output surfaces generated by different models after training with 832 data samples. The inputs x_1 and x_4 are fixed at 0.75 and 0.5 respectively while x_1 and x_4 were varied to generate these surfaces. The encircled regions signify that the output is not consistent with the domain knowledge.

One question that is worth investigating here is that why were the ridge regression and local LSE techniques unable to capture the domain knowledge despite being known for their robustness. In fact, in the previous example of 2-dimensional nonlinear function approximation, the surface generated by local LSE (figure 4.6) was as robust to noisy training dataset as was APKFM (figure 4.7). The answer to this question lies in the distribution of training data points in the input space. Figure 4.11 shows the spread of training data points in the input space for the current problem.

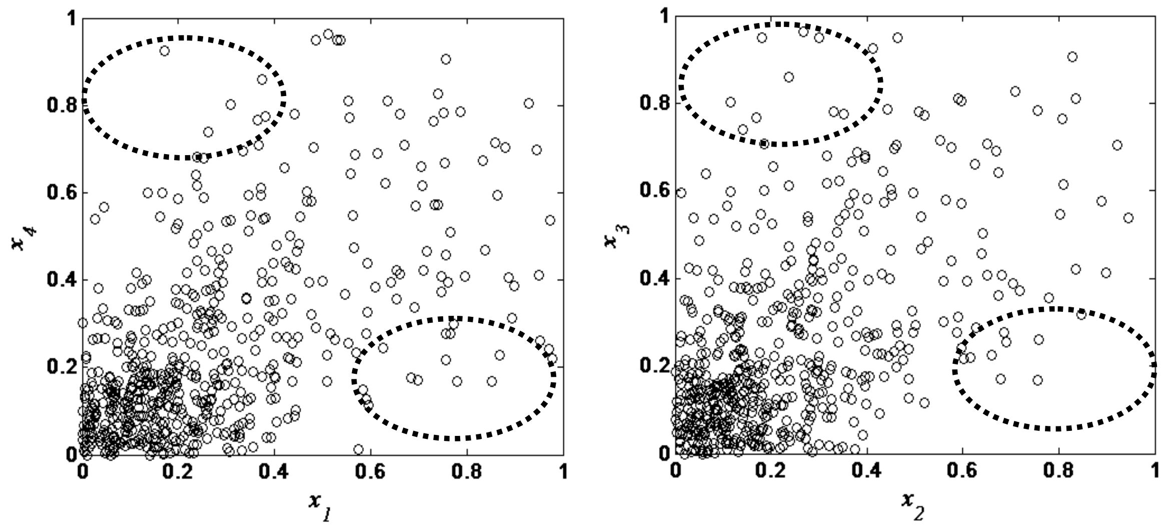

Figure 4-11 The distribution of training data-points in the input space. The encircled regions have a sparse distribution of points. These regions corresponds to those input subspaces where the model output may become inconsistent with the domain knowledge.

Clearly, there were certain regions of the input space where the distribution is sparse, as shown by the encircled regions in figure 4.11. If we carefully examine the figures 4.9 and 4.10, we observe that the inconsistent regions of output surface correspond to these sparsely populated input subspaces. As a consequence, both local LSE and ridge regression techniques performed poorly because they did not have sufficient data points to learn from. This observation indicates towards the poor extrapolation capability of these two regularization techniques, which has also been highlighted by several other authors [69-71]. In the 2-dimensional function approximation example, this limitation was not evident because the training data-points were uniformly sampled from the input space. Coming back to the current problem, the APKFM showed excellent extrapolation capability and was able to maintain the consistency of the model even in the regions with few data-points. This ability of APKFM is the direct manifestation of its knowledge function based formulation. In APKFM, unlike the other techniques, the consequent parameters of different rules are not identified independently from each other. Instead a knowledge function, $\varkappa(.)$, is used for this purpose, which assigns values to the parameters of even those rules where the distribution of data is sparse.

Therefore, the two examples of static nonlinear function approximation discussed in this section clearly indicates towards the better robustness of the APKFM compared to the other TSK model identification techniques. The superiority of APKFM is clearly evident from its ability to yield an accurate and interpretable fuzzy model from a historical process data having low signal to noise ratio. For this reason, APKFM is ideally suited for modeling real world processes that generate limited quantity of low quality data.

4.3 Conclusions

The neuro-fuzzy modeling is a unique system identification technique which, unlike other data-driven techniques, allows an expert to construct a model utilizing his physical insight about the system. Once constructed, such models are fine-tuned using a set of historically available input-output pairs in order to compensate for any form of deficiency in the expert's knowledge. However, if special care is not taken during their tuning, these models may end up behaving like black-box models thus losing their interpretability. This tendency of neuro-fuzzy models is especially true when the data used for training is either has low signal to noise ratio. The *Takagi-Sugeno-Kang* (TSK) is unarguably the most widely used type of fuzzy model but at the same time is more likely to become a black-box model compared to other types of fuzzy models (example: Mamdami models). In order to mitigate the undesirable affect of inferior quality training data, certain techniques known as *regularization techniques* are employed to estimate the parameter values of TSK models. These regularization techniques works by reducing the flexibility of a model, thereby not allowing it to get over-trained on the noise.

In this thesis, we have proposed a regularization technique which is unique in the sense that it utilizes qualitative knowledge of an expert to carry out regularization. The proposed technique is devised for the consequent parameter estimation of 1st order TSK fuzzy models. The resulting fuzzy models are called *A-Priori Knowledge-based Fuzzy Models* (APKFMs). A fundamental component of an APKFM is a specialized function known as *knowledge function*, which is formulated using the qualitative knowledge of an expert. The primary role of knowledge function is to keep the model parameters consistent with the domain knowledge even when they are being estimated from a poor quality training dataset.

The superiority of APKFM compared to the models identified by other regularization techniques, was showed using two examples of static nonlinear system identification. In the first example, the true system was known in the form of a 2-dimensional nonlinear function. The performances of fuzzy models were rated on two conflicting properties namely *accuracy* and *interpretability*. It was apparent that APKFM resulted in a best combination of these two desirable properties compared to the other regularization techniques. Moreover, the performance of APKFM got relatively better as the quality of training data deteriorated further. In the second example, we took a practical cost estimation problem related to electricity distribution networks in Spain. This was ideal problem for benchmarking as it has been already addressed using several other data-driven methodologies. It was shown that the APKFM not only maintained its consistency with the domain knowledge in the entire input space but also did not compromise with its prediction accuracy. This ideal combination of accuracy and interpretability was not found in the competing regularization techniques.

Appendix: Fuzzy Rule Based Systems

1. Fuzzy Logic

The theory of fuzzy logic was proposed by Zadeh in 1965 [72], which paved a new way to characterize non-probabilistic uncertainties. The fuzzy logic can be considered as an extension of Boolean logic wherein the value of a variable is limited to either 0 or 1. On the contrary, fuzzy logic allows a variable to take a continuum of values between 0 and 1. The invention of fuzzy logic is motivated by the observation that humans rarely think in a precise manner such that a value of 0 (false) or 1 (true) can be assigned to a variable (event). Quite often the human logic is based on vague and uncertain statements, which cannot be captured using classical Boolean logic. There exists many ways to deal with the imprecision in the statements; the probability theory is the obvious choice. However, to deal with imprecision in rule-based form, the theory of fuzzy logic had to be developed [73]. Over the years, the theory of fuzzy logic has advanced significantly and some of the important contributions can be found in [47, 74-76]. In this appendix, the fuzzy logic theory is explained only to a level that would enable a reader to understand the contents of this thesis. For detailed discussion on fuzzy logic and its application, refer to [28, 66, 77, 78] .

1.1) Fuzzy sets

Fuzzy sets are the building blocks of a fuzzy system. To understand the characteristics of a fuzzy set, first consider the definition of a set based on the classical set theory. Let there be a classical (crisp) set $A_c$defined in some finite dimensional space. The following function $f_{crisp}(X)$ defines the membership of an element X to the crisp set A_c.

$$f_{crisp}(X) = \begin{cases} 1 & if\ X \in A_c \\ 0 & if\ X \notin A_c \end{cases}$$

(1)

In other words, an element is either absent or present in a crisp set and there is no uncertainty in this regard. On the other hand, membership of an element X to a fuzzy set A_f is represented using the function $f_{fuzzy}(X)$.

$$f_{fuzzy}(X) = \begin{cases} \mu(X) & if\ X \in A_f \\ 0 & if\ X \notin A_f \end{cases}$$

(2)

Where, $\mu(X)$ is called the *membership function* of the fuzzy set A_f. This membership function is continuous and is defined such that $\mu(x) \to [0,1]$. Clearly, the membership degree of an element to a fuzzy set may vary depending on its location in the finite dimensional space. This feature of a fuzzy set enables an element to have different degrees of membership to itself. In other words the degree of membership of an element can take continuum of logic values between 0 and 1, which is not possible in two-valued Boolean logic. A value closer to 1 signifies high level of confidence that an element belongs to a given fuzzy set and vice versa.

A membership function can assume variety of shapes provided it remains continuous. Figure 1 shows two such functions defined on a normalized scale. The shape and size of these membership functions are characterized by certain parameters. For example the *Gaussian* membership function, shown in figure 1b, is defined using two parameters c and σ that denote its center and standard deviation respectively.

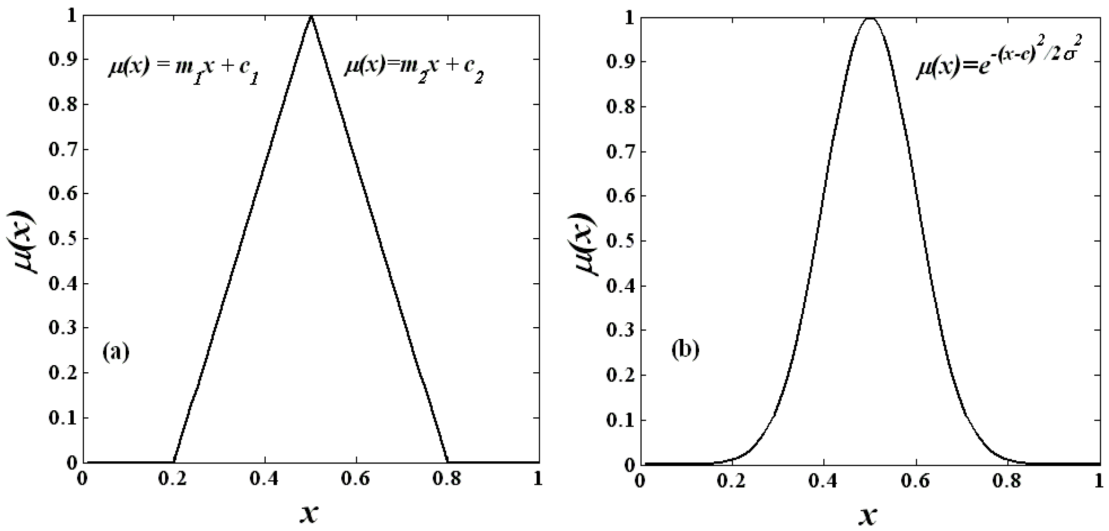

Figure 1 a) **A triangular membership function defined using four parameters m_1, m_2, c_1 and c_2.** b) **A Gaussian membership function defined using two parameters c and σ.**

Most of the day to day decisions made by humans are based on the qualitative description of a variable. We commonly associate a linguistic tag (high, low, few, often etc.) to a variable in order to define it qualitatively. Our assessment of a variable never follows a black or white pattern. Therefore, when it comes to emulating human decision making process, the use of crisp sets may not be appropriate. Fuzzy sets provide an excellent way to model this imprecision involved in the human decision making process. When it comes to building a fuzzy model of a complex system, we first need a human expert who has sufficient domain knowledge so that a comprehensive knowledge-base can be built. Thereafter, fuzzy logic theory is used to translate this qualitative knowledge-base into a mathematical fuzzy model. The next section provides a detailed discussion on the mathematical formulation of qualitative domain knowledge using fuzzy logic.

1.2) Fuzzy Rule-based Models

As mentioned earlier, a fuzzy model tries to emulate a human expert's thought process to simulate a system. So far, there does not exist any theory, based on first principles, that can quantify the human thought process. Nevertheless, human knowledge can always be expressed in the form of IF-THEN production rules. The set of all such rules is called a *rule-base* and it forms the backbone of a fuzzy model. The syntax of a production rule is:

IF <*Antecedent*> **THEN** <*Consequent*>

Clearly, a production rule consists of two parts: 1) the *IF* part, called antecedent, which signifies the input condition and 2) the *THEN* part, called consequent, which represents the resulting output when the input conditions are met. Generally, the antecedent consists of multiple linguistic statements joined together by either conjunction (AND) or disjunction (OR). Since, we are dealing with only *Multiple-Input-Single-Output* (MISO) systems, the consequent is either a single linguistic statement or a linearly parameterized function. A linguistic statement in the consequent of a production rules yields a *Mamdami* fuzzy model, which a mathematical function leads to a *Takagi-Sugeno-Kang* (TSK) model.

2 Classification of Fuzzy Models

The Fuzzy rule-based models are classified into 1) *Mamdami* Fuzzy models 2) *Takagi-Sugeno-Kang* (TSK) Fuzzy models. As mentioned earlier, the difference between the two schemes lies in the type of consequent used in their production rules. Mamdami models are historically important because they were the first type of fuzzy models that gained immense popularity in 1970s and 80s. However, their popularity and applications started to diminish by 1990s with the advent of TSK fuzzy models. The primary reason for the reduced interest in Mamdami models among practitioners was that they did not yield efficient neuro-fuzzy models. The Mamdami neuro-fuzzy models suffer from high computational demand involved in their parameter learning process. In proceeding section, we discuss the construction of both Mamdami and TSK fuzzy models.

2.1) Mamdami Fuzzy Models

The production rules used for the construction of MISO Mamdami models can be represented in the following form:

IF x_1 is Low AND x_2 is $High$ AND………………… x_n if $Medium$ **THEN** $f(X)$ is $High$.

Where, $X = (x_1, x_2 \dots\dots x_n)$ is the input vector and $f(X)$ is the fuzzy model's output. Each input is assigned certain linguistic labels (low, high, very high etc.), which represent fuzzy sets with predefined membership functions. The fuzzy sets are usually defined in a manner that partitions the entire input space into grids. For illustration, A grid partitioned input space for a two dimensional system is shown in figure A.2. We can observe that each input's domain is divided into three fuzzy sets with overlapping Gaussian membership functions. As a result the 2-D input space is partitioned into nine grids. Figure 2 shows the location of these 9 grids, which are more or less uniformly distributed in the input space. Figure 2.5, in chapter 2, shows a similar grid partitioned 2-D input space when triangular membership functions are used. It also shows the 3-D surface generated by bivariate fuzzy sets resulting from the tensor product construction.

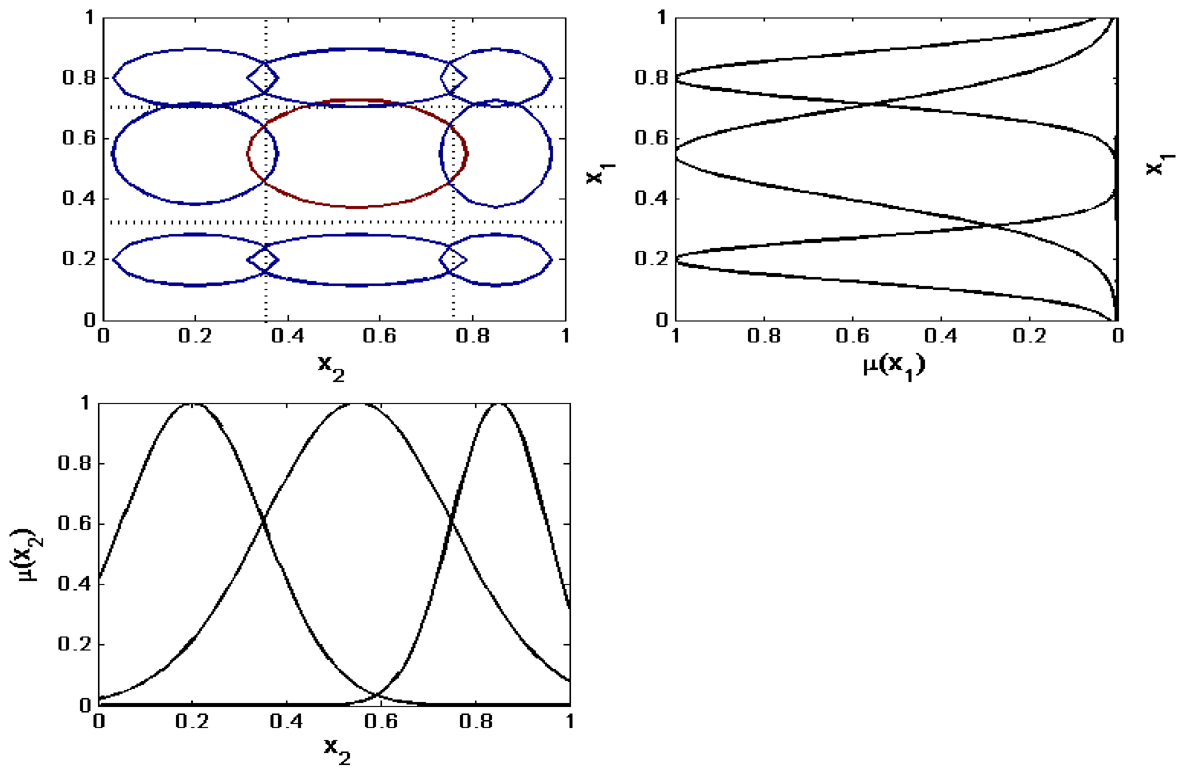

Figure 2 Grid partitioned input space of a 2-D system. Both inputs x_1 and x_2 have 3 fuzzy sets with Gaussian membership functions. The input space gets partitioned into $3 \times 3 = 9$ grids as whose locations are shown by overlapping ellipses.

In Mamdami model, the unidimensional output space is also partitioned using univariate fuzzy sets with associated linguistic labels. Thereafter, a rule-base is generated by a domain expert by translating his knowledge into linguistic IF-THEN production rules. In the current example of 2-D system, there could be maximum 9 rules corresponding to the 9 grids. However, a domain expert may not include all these 9 rules in the rule-base because some of them might be redundant in his opinion. Generally, a rule base generated by an expert is significantly smaller than the rule-base generated solely by grid partitioning. As elaborated in section 2.1.3, a grid partitioned rule-base suffers from the curse of dimensionality as it becomes prohibitively large if the number of inputs increases.

Once a comprehensive rule-base is generated by establishing all the pertinent input-output relationships, fuzzy inference can be carried out. Since we are only dealing with MISO systems, fuzzy inference can be considered as a technique of mapping a multidimensional input to a unidimensional output. To carry out the fuzzy inference, certain steps are needed to be followed. These steps are common for both Mamdami and TSK fuzzy models barring the last few steps. The chronological order of these steps is given as follows:

Fuzzification → Rule Evaluation → Implication → Aggregation → Defuzzification

To illustrate the fuzzy inference mechanism for Mamdami models, we take an example of a fuzzy controller embedded in an air-conditioning (AC) unit, which regulates the temperature and humidity of a room. The controller should be able to assess the prevailing interior conditions and regulate the operation of AC unit such that temperature and humidity are maintained at the preset values. Let there be two inputs to the fuzzy controller i.e. *temperature* and *humidity*. The output is the operating *power* of the AC unit. Our goal is to build a simple Mamdami model which could be used as the aforementioned controller. We begin with assigning two fuzzy sets to both inputs. The fuzzy sets assigned to the inputs temperature and humidity are given linguistic labels cold/warm and low/high respectively. Additionally, each fuzzy set is represented by a triangular membership function. The output space is also partitioned using two univariate Gaussian membership functions with linguistic label low and high. Since each input domain is divided into two fuzzy sets, we can have maximum four rules in the rule base. However, let's assume that, as per the domain knowledge, the rule-base consists of the following three rules:

Rule 1: IF **Temperature** is *warm* AND **Humidity** is *high* THEN **Power** is *high*
Rule 2: IF **Temperature** is *cold* AND **Humidity** is *low* THEN **Power** is *low*
Rule 3: IF **Temperature** is *warm* AND **Humidity** is *low* THEN **Power** is *high*

Given this rule-base, the mechanism of Mamdami Inference is explained as follows. Let the current value of room temperature and humidity be 80°F and 2 mg/cc respectively. The different steps of Mamdami inference are outlined in the same order as mentioned above.

Fuzzification: The first step, known as *fuzzification*, converts the crisp input values to the corresponding fuzzy values. The fuzzification is carried out using the input membership functions. For this case, the crisp values of temperature and humidity (80 °F and 2 mg/cc) are converted to the fuzzy values given below (refer figure 3):

$$\mu_{temp}^{cold}(80) = 0.32 \quad \& \quad \mu_{humid}^{low}(2) = 0.64$$
$$\mu_{temp}^{warm}(80) = 0.68 \qquad \mu_{temp}^{warm}(2) = 0.36$$

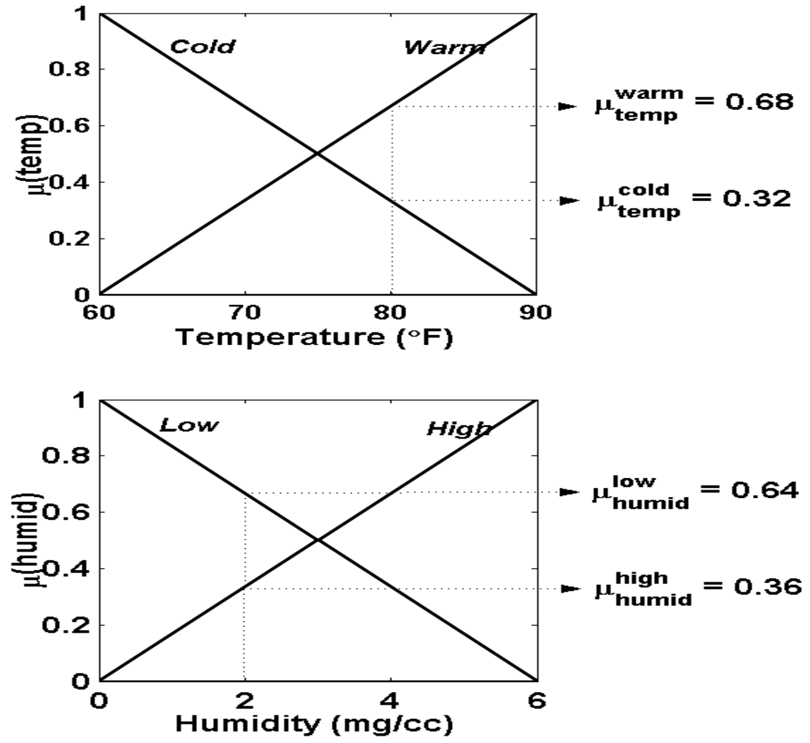

Figure 3 Fuzzification of inputs temperature and humidity using triangular membership functions. A crisp value of 80°F is warm by a degree of 0.68 and cold by 0.32. Similarly, the crisp humidity value of 2 mg/cc is low by a degree of 0.64 and high by 0.36.

The fuzzified values signify the degree of membership of an input to different fuzzy sets. For example, the temperature value of 80°F belongs to the fuzzy sets *cold* and *warm* to a degree of 0.32 and 0.68 respectively. Similar inference can be made for the crisp humidity value of 2 mg/cc.

Rule evaluation: When an input is given to a fuzzy model, generally, a small fraction of the total number of rules are fired. The dormant rules have no influence of on the model output because their firing strengths are zero. In rule evaluation, we determine the firing strength (τ) of every rule in the rule base. The firing strength of a rule is obtained by performing conjunction between the fuzzy values of the inputs appearing in that rule. The operators used for performing the conjunction are called τ-norms [79]. The most commonly used τ-norms operators are *min* and *product*. If the product operator is used as τ-norm, then the firing strength is simply the product of input fuzzy values. The firing strengths of three rules, in this example would be:

Rule 1: $\tau_1 = \mu_{temp}^{warm}(80) \times \mu_{humid}^{high}(2) = 0.68 \times 0.36 = 0.245$

Rule 2: $\tau_2 = \mu_{temp}^{cold}(80) \times \mu_{humid}^{low}(2) = 0.32 \times 0.64 = 0.205$

Rule 3: $\tau_3 = \mu_{temp}^{warm}(80) \times \mu_{humid}^{low}(2) = 0.68 \times 0.64 = 0.435$

Implication: By implication we mean the effect of a rule's firing strength on its output fuzzy set. The implication is done by truncating the output membership function at a value equal to the firing strength (τ). The result of implication is a new fuzzy set with a truncated membership function. This is shown in figure 4, where the truncated output fuzzy sets, for the three rules, are shown by the shaded regions. It can be noted that the truncation is carried out at a value equal to a rule's firing strength.

Aggregation: After performing implication on all the rules with non-zero firing strength, all the truncated fuzzy sets are combined into a single fuzzy set. The unification is done by applying an aggregation method (max, probor, sum etc.) [80]. In this case we use *max* as the aggregation operator, which results in an aggregated fuzzy set also shown in figure 4.

Defuzzification: The final step is to perform the defuzzification of the aggregated fuzzy set to obtain a crisp output value. There are several methods that can be employed for defuzzification, which may yield substantially different output values. Perhaps the most popular defuzzification method is *center of gravity* (COG) method [80, 81], in which the output corresponds to the centroid of the area of aggregated fuzzy set. Using COG method, we obtain the output power values of 0.66 W as shown in figure 4. In this way Mamdami inference leads to an output power value of 0.66 W when input values are 80°F (temperature) and 2 mg/cc (humidity).

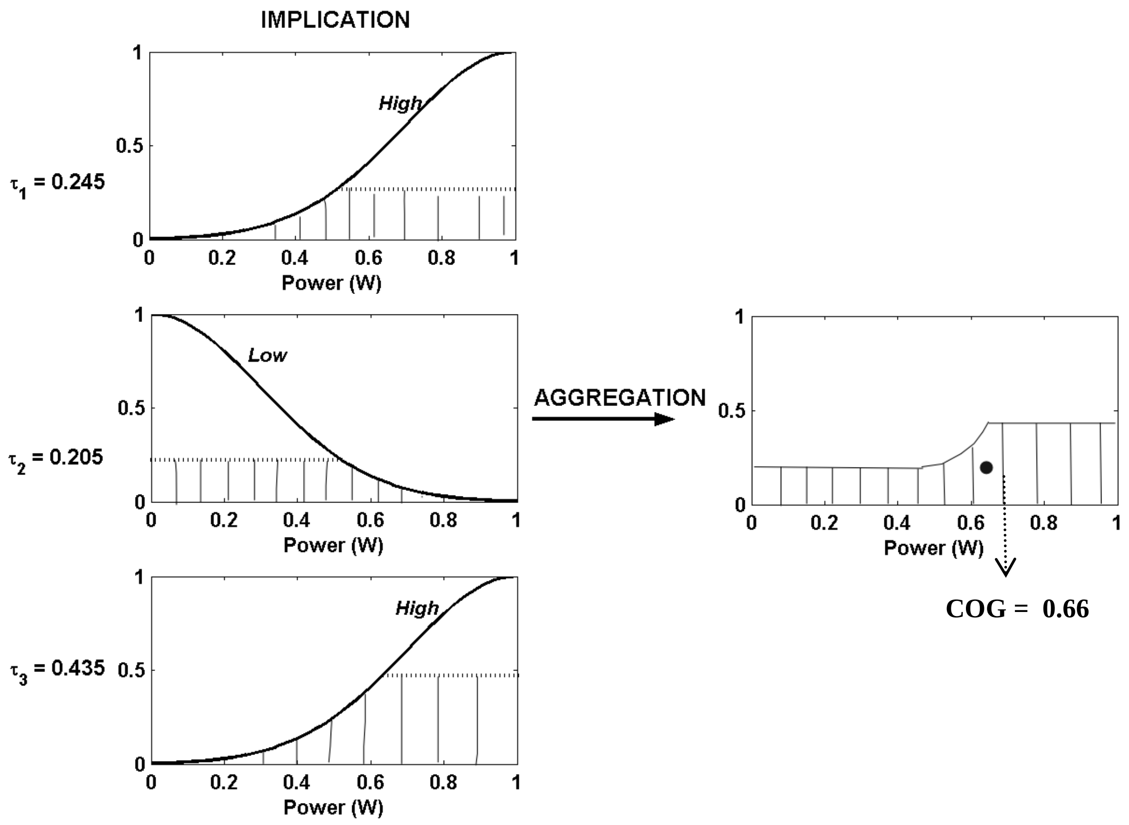

Figure 4 Graphical illustration of 3 important steps of Mamdami inference. 1) Implication: **truncating output fuzzy sets at strength values of fired rules 2)** Aggregation: **Combining truncated fuzzy sets to form a single aggregated fuzzy set and 3)** Defuzzification: **Obtaining the crisp output value by computing the centroid of aggregated fuzzy set.**

There are some inherent difficulties involved in the implementation of Mamdami fuzzy models. Overall, the Mamdami fuzzy inference is quite complicated and computationally demanding. To be specific, the defuzzification step is significantly complex because it requires time consuming numerical computation of the centroid. Additionally, the learning schemes of Mamdami fuzzy models are more intricate than those of Takagi-Sugeno-Kang fuzzy models, which make the later a better choice for building adaptive neuro-fuzzy models.

2.2) Takagi-Sugeno-Kang (TSK) Fuzzy Models

The TSK models are similar to the Mamdami models in their construction with the only exception in the representation of a rule's consequent part. Instead of being a fuzzy set, the rule consequent in a TSK model is a polynomial function of the input variables. Therefore, unlike Mamdami models, in TSK models the rule consequents have a functional form, which cannot be represented in linguistic terms. A rule from a 1st order TSK model has the following representation:

IF x_1 is *Low* AND x_2 is *High* AND.............. x_n if *Medium* **THEN** $f(X)$ is $P(X)$

where $P(X) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \ldots\ldots \theta_n x_n$ **(3)**

The order of a TSK fuzzy model is same as the order of the polynomial used in its rule consequents. Generally, there is no restriction on the order of this polynomial. However, as we increase the polynomial order the complexity of a model also increases, which need not yield a better TSK model. An overly complex model suffers from high variance error as emphasized in the discussion of bias-variance tradeoff presented in section 1.2.2. In fact, for most of the static and dynamic modeling problems either a 0th order or 1st order TSK model is sufficient.

The TSK inference is similar to the Mamdami inference for the first few steps. The input fuzzification and rule evaluation are carried out in the same manner as described for Mamdami models. However, because of different consequents, the inference mechanism deviates from Mamdami inference after the rule evaluation step. Firstly, the defuzzification is not needed in TSK inference, because fuzzy sets are not used to represent the model output. Secondly, the implication and aggregation steps are combined together and the model output is calculated as:

$$\hat{f}(X) = \frac{\sum_{r=1}^{R} P^r(X)\,\tau^r}{\sum_{r=1}^{R} \tau^r} = \sum_{r=1}^{R} P^r(X)\,\bar{\tau}^r$$

(4)

where, R is the total number rules in the rule-base and $\bar{\tau}^r = {\tau^r}/{\sum_{r=1}^{R} \tau^r}$ is the normalized firing strength of the r^{th}rule. This normalization provides an extremely desirable addition to unity property to the fired rules. If we carefully examine equation 4, we can observe its resemblance with the general basis function formulation given in equation 2.12. The TSK models can be interpreted as a collection of several local linear models, where every local model has a validity region as defined by a rule's antecedent. These validity regions are visible in both figure 2 and figure 2.5. The model output is obtained by first weighting each local model by the firing strength of the corresponding rule and then summing them together. The TSK inference would become clearer if we consider an example of a 1-dimensional function approximation, where the function to be approximated is a parabola centered at $x = 0.5$:

$$f(x) = a(x - 0.5)^2 + b \tag{5}$$

A simple 1st order TSK fuzzy model with following rules can be built to approximate this function:

Rule 1: IF x is low THEN $f(x) = \theta_o^1 + \theta_1^1 x$

Rule 2: IF x is medium THEN $f(x) = \theta_o^2 + \theta_1^2 x$

Rule 3: IF x is high THEN $f(x) = \theta_o^3 + \theta_1^3 x$

This rule base is generated by partitioning the one-dimensional input space in three fuzzy sets with linguistic labels low, medium and high and triangular membership functions. Figure 5 shows the schematic of the resulting TKS fuzzy model.

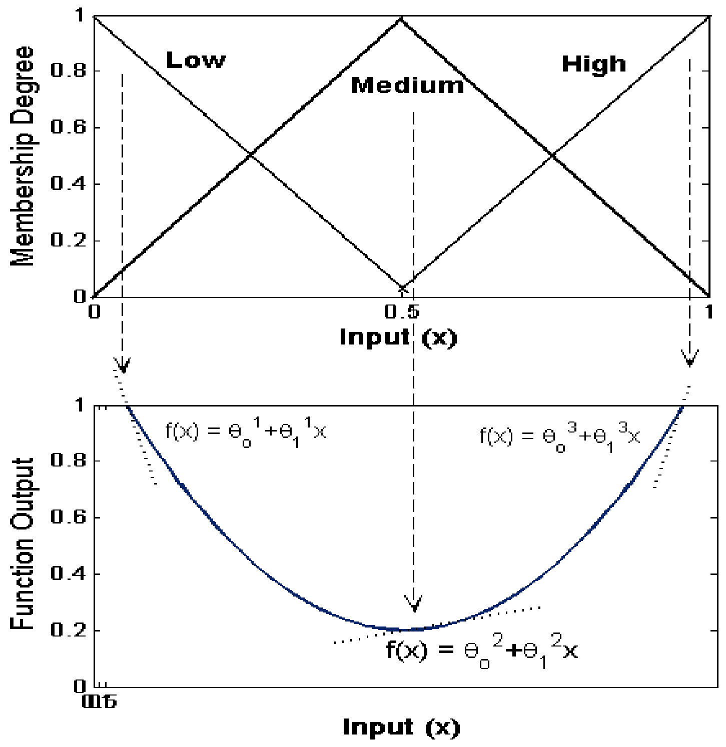

Figure 5 A graphical representation of a 1st order TSK fuzzy model for a one dimensional system. The input space is divided by 3 fuzzy sets resulting in 3 rules. The rule consequents are linear equations defined at the center of corresponding rules.

Here we see that linguistic antecedent of each rule has a triangular validity region in the input space. For each rule we define a local model which approximates the underlying nonlinear function $f(x)$ by a linear equation. As mentioned earlier, the model output is nothing but the weighted arithmetic mean of these local linear models. Therefore, a TSK model output is obtained by interpolating between various local models that have different validity regions in the input space. Only those local models are used during the interpolation, which have an active validity region for a given input. Apparently unlike the Mamdami models, which require computationally intensive centroid calculation, the output calculation in TSK fuzzy models is fairly straightforward owing to the functional form of its rule consequents.

2.3) RBFNs vs. Singleton/0th order TSK Models

In this section we would like to briefly highlight the similarities between the TSK fuzzy models with the Radial Basis Function Networks (RBFNs). Although the theories of TSK models and RBFN were developed independently, there exists a strong resemblance between the two. Under following restrictions on the TSK fuzzy models, they are equivalent to normalized RBFNs.

1. The TSK model should be of 0th order i.e. the local models should simply be constants (singletons).
2. Gaussian functions should be used as the membership functions for fuzzy sets and the product operator should to perform the conjunction in a rule's antecedent.

When the above two conditions are met the output of a TSK fuzzy model is given by:

$$\hat{f}(X) = \sum_{r=1}^{R} \theta_o^r \left(\prod_{i=1}^{n} e^{-\frac{1}{2}\frac{(x_i - c_i^r)^2}{\sigma_i^r}} \right) \tag{6}$$

Where, θ_o^r is the singleton/constant local model, c_i^r and σ_i^r are the center and standard deviation of the Gaussian membership function of the i^{th} input appearing in the r^{th} rule. The expression in the bracket represents the firing strength of r^{th} rule obtained by taking the product of individual Gaussian membership functions. Equation 6 can be rewritten in the form shown in equation 7 by replacing the product of univariate Gaussian membership functions by a single n dimensional Gaussian function.

$$\hat{f}(X) = \sum_{r=1}^{R} \theta_o^r \, e^{-\frac{1}{2}\sum_{i=1}^{n}\frac{(x_i - c_i^r)^2}{\sigma_i^r}}$$

(7)

The equation 7 essentially represents the output of a RBFN (refer equation 2.10) with the basis function placed on a grid partitioned input space. For more in-depth discussion on the equivalence of TSK fuzzy models and RBF networks refer [82].

References

[1] A. Rutherford, *Mathematical Modeling Techniques*, New York: Dover, 1994.

[2] E. Heitz, H. C. Flemming, and W. Sand, *Microbially influenced corrosion of materials*, Berlin, Heidelberg: Springer, 1996.

[3] R. Javaherdashti, *Microbiologically influenced corrosion- An engneering insight*, UK: Spinger, 2008.

[4] D. P. Solomatine, "Data-driven modeling: Paradigm, methods, experiences ". pp. 757-763.

[5] P. Lindskog, "Methods, algorithms and tools for system identification based on prior knowledge.," Department of Electrical Engineering, Linkoping University, Linkoping, Sweden, 1996.

[6] G. M. James. "Variance and bias for general loss functions," http://www-rcf.usc.edu/~gareth/research/bv.pdf

[7] L. Ljung, *System Identification- Theory for User*, 2 ed., Upper Saddle River, NJ, 1999.

[8] G. Stuart, B. Elie, Ren *et al.*, "Neural networks and the bias/variance dilemma," *Neural Comput.*, vol. 4, no. 1, pp. 1-58, 1992.

[9] N. R. Draper, and H. Smith, *Applied Regression Analysis*, New York: John Wiley & Sons, 1981.

[10] M. T. Hagan, and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *Neural Networks, IEEE Transactions on*, vol. 5, no. 6, pp. 989-993, 1994.

[11] J. Y. F. Yam, and T. W. S. Chow, "Extended least squares based algorithm for training feedforward networks," *Neural Networks, IEEE Transactions on*, vol. 8, no. 3, pp. 806-810, 1997.

[12] J.-S. R. Jang, and E. Mizutani, "Levenberg-Marquardt method for ANFIS learning," Berkeley, CA, USA, 1996, pp. 87-91.

[13] T. Feuring, J. J. Buckley, and Y. Hayashi, "Gradient descent learning algorithm for fuzzy neural networks," Anchorage, AK, USA, 1998, pp. 1136-1141.

[14] P. J. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," Harvard, Boston, 1974.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533-536, 1986.

[16] M. J. D. Powell, "Radial basis functions for multivariable interpolation: a review," *Algorithms for approximation*, pp. 143-167: Clarendon Press, 1987.

[17] D. S. Broomhead, and D. Lowe, "Multivariable function interpolation and adaptive networks," *Complex Systems*, vol. 2, pp. 321-355, 1988.

[18] J. A. S. Freeman, and D. Saad, "Learning and generalization in Radial Basis Function Network," *Neural Computation*, vol. 7, no. 5, pp. 1000-1020, 1995.

[19] J. Park, and I. W. Sandberg, "Universal approximation using radial-basis-function-networks," *Neural Computation*, vol. 3, no. 2, pp. 246-257, 1991.

[20] J. M. Hutchinson, "A Radial Basis function approach to financial time series analysis," MIT, Boston, 1994.

[21] Z. Shi, Y. Tamura, and T. Ozaki, “Nonlinear time series modelling with the radial basis function-based state-dependent autoregressive model,” *International Journal of Systems Science,* vol. 30, no. 7, pp. 717-727, 1999.

[22] S. Haralambos, D. Philip, and A. Alex, “A classification technique based on radial basis function neural networks,” *Adv. Eng. Softw.,* vol. 37, no. 4, pp. 218-221, 2006.

[23] G. A. Wilkin, and H. Xiuzhen, "K-means clustering algorithms: Implementation and comparison," *Proceedings - 2nd International Multi-Symposiums on Computer and Computational Sciences, IMSCCS'07.* pp. 133-136.

[24] A. M. Bagirov, “Modified global K-means algorithm for minimum sum-of-squares clustering problems,” *Pattern Recognition,* vol. 41, no. 10, pp. 3192-3199, 2008.

[25] S. L. Chiu, "Cluster estimation method with extension to fuzzy model identification," *IEEE International Conference on Fuzzy Systems.* pp. 1240-1245.

[26] O. Nelles, and R. Isermann, "A new technique for the determination of hidden layer parameters in RBF Networks."

[27] S. Chen, S. A. Billings, and W. Luo, “Orthogonal least squares methods and their application to non-linear system identification,” *International Journal of Control,* vol. 50, no. 5, pp. 1873 - 1896, 1989.

[28] O. Nelles, “Nonlinear System Identification,” *Measurement Science and Technology,* vol. 13, no. 4, pp. 245-250, 2002.

[29] A. Krone, and T. Slawinski, "Data-based extraction of unidimensional fuzzy sets for fuzzy rule generation." pp. 1032-1037 vol.2.

[30] I. Hideyuki, F. Toshio, S. Takanori *et al.*, “Structure optimization of fuzzy neural network by genetic algorithm,” *Fuzzy Sets Syst.,* vol. 71, no. 3, pp. 257-264, 1995.

[31] Z. Świątnicki, and V. Olej, "Generation and Optimization of Fuzzy Neural Network Structure," *PRICAI 2002: Trends in Artificial Intelligence*, pp. 57-67, 2002.

[32] J. Ni, Q. Song, and M. J. Grimble, "Robust pruning of RBF network for neural tracking control systems," *Proceedings of the IEEE Conference on Decision and Control.* pp. 6331-6336.

[33] N. R. Pal, and T. Pal, “On rule pruning using fuzzy neural networks,” *Fuzzy Sets and Systems,* vol. 106, no. 3, pp. 335-347, 1999.

[34] T. Kavli, and E. Weyer, "ASMOD (Adaptive Spline Modelling of Observation Data) - some theoretical and experimental results," *IEE Colloquium (Digest).* pp. 3-1.

[35] E. Weyer, and T. Kavli, “Theoretical properties of the ASMOD algorithm for empirical modelling,” *International Journal of Control,* vol. 67, no. 5, pp. 767-789, 1997.

[36] A. S. Jamab, and B. N. Araabi, "Piecewise linear model tree: A modified combination of two learning algorithms for neuro-fuzzy models," *Proceedings of the IEEE International Conference on Control Applications.* pp. 2155-2159.

[37] M. Fischer, O. Nelles, and R. Isermann, “Predictive control based on local linear fuzzy models,” *International Journal of Systems Science,* vol. 29, no. 7, pp. 679-697, 1998.

[38] O. Nelles, M. Fischer, and B. Mueller, "Fuzzy rule extraction by a genetic algorithm and constrained nonlinear optimization of membership functions," *IEEE International Conference on Fuzzy Systems.* pp. 213-219.

[39] C. Harpham, C. Dawson, and M. Brown, “A review of Genetic Algorithm applied to training of Radial Basis Function Network,” *Neural Computing & Applications,* vol. 13, no. 3, pp. 193-201, 2004.

[40] M. Mitchell, *An Introduction to Genetic Algorithms*, p.^pp. 221: MIT Press, 1998.

[41] O. Nelles, "GA based generation of fuzzy rules" *Fuzzy Evolutionary Computation*, Boston: Kluwer Academic 1997.

[42] J. Wolberg, *Data analysis using the method of least squares: Extracting the most information from experiments* Springer, 2005.

[43] J. S. R. Jang, "Least-Sqaure Methods for System Identification," *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, pp. 95-128: Prentice Hall, 1997.

[44] O. Nelles, "Linear Optimization," *Nonlinear System Identification*, pp. 35-77: Springer, 2001.

[45] R. Fuller, "Fuzzy Neural Networks," *Introduction to Neuro-Fuzzy Systems*, Advances in Soft Computing, p. 289: Springer, 2000.

[46] J. Botzheim, E. Lughofer, E. P. Klement *et al.*, "Separated antecedent and consequent learning for Takagi-Sugeno fuzzy systems," *IEEE International Conference on Fuzzy Systems*. pp. 2263-2269.

[47] J.-S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 665-685, 1993.

[48] J. Abonyi, R. Babuska, M. Setnes *et al.*, "Constrained parameter estimation in fuzzy modeling," *IEEE International Conference on Fuzzy Systems*. pp. 951-956.

[49] A. Fiordaliso, "Constrained Takagi-Sugeno fuzzy system that allows for better interpretation and analysis," *Fuzzy Sets and Systems*, vol. 118, no. 2, pp. 307-318, 2000.

[50] K. M. Bossley, *Regularization Theory Applied to Neuro-Fuzzy Modeling*, Universty of Southampton, 2003.

[51] J. E. Ketz, and W. E. Leininger, "Ridge regression as a cost estimation technique," *Minutes of the Meeting - Pennsylvania Electric Association, Engineering Section*, vol. 1, pp. 19-21, 1979.

[52] D. W. Marquardt, "Ridge regression in practice," *The American Statistician* vol. 29, no. 1, pp. 3-20, 1975.

[53] J. Yen, L. Wang, and W. Gillespie, "Global-local learning algorithm for identifying Takagi-Sugeno-Kang fuzzy models," *IEEE International Conference on Fuzzy Systems*. pp. 967-972.

[54] J. Yen, W. Liang, and C. W. Gillespie, "Improving the interpretability of TSK fuzzy models by combining global learning and local learning," *Fuzzy Systems, IEEE Transactions on*, vol. 6, no. 4, pp. 530-537, 1998.

[55] R. Murray-Smith, and T. A. Johansen, "Local learning in local model networks." pp. 40-46.

[56] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy C-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2-3, pp. 191-203, 1984.

[57] L. E. Scales, *Introduction to non-linear optimization*: Springer-Verlag New York, Inc., 1985.

[58] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, p.^pp. 872: Wiley-Interscience, 2006.

[59] O. Nelles, "Nonlinear Global Optimization," *Nonlinear System Identification*, pp. 113-135: Spriger, 2001.

[60] J. K. Karlof, *Integer Programming: Theory and Practice*, p.^pp. 316: CRC Press, 2006.

[61] T. F. Coleman, and Y. Li, "A Reflective Newton Method for Minimizing a Quadratic Function Subject to Bounds on some of the Variables," *SIAM Journal on Optimization,* vol. 6, no. 4, pp. 1040-1058, 1996.

[62] M. Bikdash, "Highly interpretable form of sugeno inference systems," *IEEE Transactions on Fuzzy Systems,* vol. 7, no. 6, pp. 686-696, 1999.

[63] M. Setnes, R. Babuska, and H. B. Verbruggen, "Rule-based modeling: Precision and transparency," *IEEE Transactions on Systems, Man & Cybernetics Part C: Applications and Reviews,* vol. 28, no. 1, pp. 165-167, 1998.

[64] A. Riid, and E. Rustern, "Transparent fuzzy systems in modeling and control," *Interpretability Issues in Fuzzy Modeling*, Studies in Fuzziness and Soft Computing 128, J. Casillas, O. Cordon, F. Herrera *et al.*, eds., pp. 452-476: Springer, 2003.

[65] L. J. Herrera, H. Pomares, I. Rojas *et al.*, "TaSe, a Taylor series-based fuzzy system model that combines interpretability and accuracy," *Fuzzy Sets and Systems,* vol. 153, no. 3, pp. 403-427, 2005.

[66] R. Babuska, and H. Verbruggen, "Neuro-fuzzy methods for nonlinear system identification," *Annual Reviews in Control,* vol. 27 I, pp. 73-85, 2003.

[67] J. Abonyi, R. Babuska, H. B. Verbruggen *et al.*, "Incorporating prior knowledge in fuzzy model identification," *International Journal of Systems Science,* vol. 31, no. 5, pp. 657-667, 2000.

[68] O. Cordon, F. Herrera, and L. Sanchez, "Solving electrical distribution problems using hybrid evolutionary data analysis techniques," *Applied Intelligence,* vol. 10, no. 1, pp. 5-24, 1999.

[69] J. S. R. Jang, "Input selection for ANFIS learning," *IEEE International Conference on Fuzzy Systems.* pp. 1493-1499.

[70] K. Kosanovich, A. Gurumoorthy, E. Sinzinger *et al.*, "Improving extrapolation capability of neural networks," in IEEE International Symposium on Intelligent Control, Dearborn, MI, 1996.

[71] P.-Q. Li, and X.-R. Li, "Interpolation and extrapolation ability of fuzzy neural network load modeling," *Gaodianya Jishu/High Voltage Engineering,* vol. 34, no. 6, pp. 1155-1160, 2008.

[72] L. A. Zadeh, "Fuzzy sets," *Information and Control,* vol. 8, no. 3, pp. 338-353, 1965.

[73] L. A. Zadeh, "Fuzzy-algorithmic approach to the definition of complex or imprecise concepts," *International Journal of Man-Machine Studies,* vol. 8, no. 3, pp. 249-291, 1976.

[74] D. Dubois, and H. Prade, "Management of Uncertainty in Fuzzy Expert System and Some Applications," pp. 39-58: CRC Press Inc, Boca Raton, FL, USA, 1987.

[75] T. Takagi, and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on Systems, Man and Cybernetics,* vol. SMC-15, no. 1, pp. 116-132, 1985.

[76] J. M. Mendel, *Uncertain rule-based fuzzy logic systems introduction and new directions*, Upper Saddle River, NJ: Prentice Hall PTR, 2001.

[77] R. Babuska, and H. B. Verbruggen, "Overview of fuzzy modeling for control," *Control Engineering Practice,* vol. 4, no. 11, pp. 1593-1606, 1996.

[78] N. Detlef, and K. Rudolf, "Neuro-fuzzy systems for function approximation," *Fuzzy Sets Syst.,* vol. 101, no. 2, pp. 261-271, 1999.

[79] J. Fodor, "Binary operations on fuzzy sets: Recent advances," *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science).* pp. 16-29.

[80] D. Nauck, F. Klawonn, and R. Kruse, *Foundations of neuro-fuzzy systems*: John Wiley & Sons, 1997.

[81] M. Negnevitsky, *Artificial Intelligence: A guide to intelligent systems*, 2 ed.: Addison Wesley, 2005.

[82] J. S. R. Jang, and C. T. Sun, "Functional equivalence between radial basis function networks and fuzzy inference systems," *Neural Networks, IEEE Transactions on*, vol. 4, no. 1, pp. 156-159, 1993.

VITA

Ashutosh Tewari was born in Hoshangabad, Madhya Pradesh, India on April 14, 1980, the son of Maya Tewari and Tara Chandra Tewari. After completing his higher secondary education at Kendriya Vidhyalaya, Andrews Ganj, New Delhi, India in 1997, he entered the Indian Institute of Technology, Bombay, India, from where he received the degrees of Bachelor of Technology and Master of Technology in 2003. In August, 2003, he entered the graduate School at the Pennsylvania State University to pursue a doctoral degree from the department of Engineering Science and Mechanics.

Permanent Address:
317C Regal
Shipra Sun City
Indirapuram, Ghaziabad
UP: 201010, India

Local Address:
118 Downey Drive
Manchester, CT: 06040
USA

This thesis was typed by the author.