INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

**ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600**

UMI[®]

NEW METHOD AND ALGORITHM FOR INTELLIGENT SIGNAL FILTERING BASED ON FUZZY LOGIC

by

Zhi Ke

B.Eng.

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirement for the Degree of

Master of Applied Science

Department of
Mechanical and Aerospace Engineering

Ottawa-Carleton Institute
for Mechanical and Aerospace Engineering

Carleton University

Ottawa, Ontario, Canada

September 14, 2001

**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-66893-2

Canada

The undersigned recommend to
the Faculty of Graduate Studies and Research
acceptance of the thesis

**NEW METHOD AND ALGORITHM
FOR INTELLIGENT SIGNAL FILTERING BASED ON FUZZY LOGIC**

Submitted by

Zhi Ke, B.Eng.

In partial fulfillment of the requirement for
the Degree of Master of Applied Science

Thesis Supervisor

Chair, Department of Mechanical and Aerospace Engineering

Carleton University

September 14, 2001

ABSTRACT

The problem of estimating the state of a stochastic dynamical system from noisy observations taken on the state is of central importance in engineering. Noise filtering is an important part of processing a real signal sequence. The more common approach in modeling the noise is to assume that it has Gaussian distribution in addition to some low percentage of additive impulsive noise. The filtering involves the removal or reduction of impulsive noise along with the Gaussian noise while keeping the original signal.

In this thesis, we present a fuzzy filter, which is based on fuzzy logic theory and Kalman filter. It is similar to Kalman filter when a linear system with Gaussian noise is considered. However, when non-Gaussian noise is introduced, it is shown that fuzzy filter is outperforming Kalman filter, while Kalman filter does not work well. We demonstrate the performances of Kalman filter and fuzzy filter for position estimation application under different kinds of circumstances. The comparisons are made to draw conclusions.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Prof. Jurek Z. Sasiadek, who is keeping encouraging me and letting me make progress continuously. Without his patience and brilliant advice, my thesis could not have been done.

I would like to express my deepest mourning to my father, Baotai Ke's passing away in 2001, I pray for him to have peace and harmony in heaven. I thank my family, my mother, Fengluan Yang, my sister, Hui Ke. Their love and care make me have the strength to finish my study successfully. From them, I have learnt that the love of family could not be replaced by anything else.

I would like to thank my lovely girl friend, Qing Zhao, who is giving me her love, sharing my pain and success. Her continual support, encouragement and patience are highly appreciated.

I would also like to thank the faculty members and staff of the Department of Mechanical and Aerospace Engineering of Carleton University for their help and support throughout my study.

Finally, I must thank all the people who have helped me by all means.

TABLE OF CONTENTS

TITLE.....	i
ACCEPTANCE FORM.....	ii
ABSTRACT.....	iii
ACKNOWLEDGEMENTS.....	iv
TABLE OF CONTENTS.....	v
LIST OF TABLES.....	x
LIST OF FIGURES.....	xi
LIST OF SYMBOLS.....	xvi

Chapter1	INTRODUCTION.....	1
Chapter 2	KALMAN FILTER.....	12
	2.1 Introduction to Kalman Filter.....	12
	2.2 The Discrete Kalman Filter.....	14
	2.2.1 Discrete-Time Model.....	14
	2.2.2 Derivation of Discrete Kalman Filter.....	15
	2.3 Filter Parameters and Tuning.....	23
	2.4 Linearization.....	25
	2.5 Linearized Kalman Filter.....	26
	2.6 The Extended Kalman Filter (EKF)	28

Chapter 3	APPLICATION OF KALMAN FILTER	
	FOR NOISE REDUCTION.....	33
3.1	Introduction.....	33
3.2	Example of Discrete Kalman Filter Application.....	35
3.2.1	System Model.....	35
3.2.2	Simulation Results.....	38
3.2.3	Parameter Tuning and Analysis.....	40
3.2.3.1	Time Step.....	40
3.2.3.2	Signal Frequency.....	42
3.2.3.3	More complicated model.....	43
3.2.3.4	Process Noise Covariance Q	
	and Measurement Noise Covariance R	45
3.2.3.5	Correlated Process Noise and	
	Measurement Noise.....	52
Chapter 4	FUZZY LOGIC AND FUZZY CONTROL.....	60
4.1	Introduction to Fuzzy Logic.....	60
4.1.1	Fuzzy Logic and Control Engineering.....	62
4.2	Fuzzy Sets.....	63
4.2.1	Introduction to Fuzzy Sets.....	63
4.2.2	Universe.....	64
4.2.3	Membership Function.....	64
4.2.4	Singletons.....	66

4.2.5 Operations on Fuzzy Sets.....	67
4.2.5.1 Fuzzy Union and Intersection.....	67
4.2.5.2 Fuzzy Complement.....	69
4.3 Fuzzy Control.....	69
4.3.1 Advantages of Fuzzy Control.....	69
4.3.2 Elements of Fuzzy Controller.....	70
4.3.3 Rule-Base.....	72
4.3.4 Fuzzy Connectives.....	72
4.3.5 Fuzzy Implication.....	73
4.3.6 Inference Mechanism.....	74
4.3.7 Inference Step.....	74
4.3.8 Fuzzification.....	75
4.3.9 Defuzzification.....	77
4.4 Summary.....	79
Chapter 5 DESIGN OF FUZZY SYSTEM.....	80
5.1 Design Methodology.....	81
5.2 Choice of Control Variables.....	83
5.3 Input Set Design.....	83
5.4 Rule Design.....	83
5.5 Output Set Weighting.....	84
5.6 Performance Tuning.....	85
5.7 Rule-Base Tuning.....	86

	5.8 Example of Fuzzy Controller Design.....	87
	5.8.1 Fuzzy Control in Detail.....	88
	5.8.2 Building Fuzzy Controller.....	92
	5.9 Summary.....	97
Chapter 6	FUZZY FILTER AND APPLICATION.....	98
	6.1 Introduction to Fuzzy Filter.....	98
	6.2 Fuzzy Filter Construction.....	99
	6.2.1 Basic Framework.....	99
	6.2.2 Fuzzy Dynamic System Model.....	101
	6.2.2.1 Cartesian Product.....	102
	6.2.2.2 Fuzzy Composition.....	103
	6.2.2.3 Construction of Fuzzy Dynamic Model via Fuzzy Composition.....	103
	6.2.3 Fuzzy Measurement Model.....	106
	6.2.4 Measurement Update Block.....	107
	6.2.4.1 Fuzzy Bayesian Rule.....	107
	6.2.5 Defuzzification and Optimality.....	109
	6.3 Fuzzy Filter Application.....	110
Chapter 7	COMPARISON BETWEEN KALMAN FILTER AND FUZZY FILTER.....	117
	7.1 Kalman Filter.....	117
	7.2 Fuzzy Filter.....	119

7.3 Summary.....	119
Chapter 8 CONCLUSION.....	124
REFERENCES.....	126
APPENDICES.....	132
APPENDIX A.....	132
A-1 Kalman Filter.....	132
A-2 Kalman Filter with Complex Dynamic Model.....	134
A-3 Kalman Filter with Weighted Q and R , and Correlated Process and Measurement Noise.....	137
APPENDIX B.....	140
B-1 Fuzzy Filter for Two Inputs (signal error and change in the error)	140
B-2 Fuzzy Filter for Four Inputs (signal error, change in the error, difference of error covariance $\mathbf{P}_k$ and $\mathbf{P}_k^-$, change in this difference)	146

LIST OF TABLES

Table 4-1: Mathematical Characterization of Triangular	
Membership Functions.....	66
Table 5-1: Rule Table for Input and Output.....	93
Table 5-2: Graphic Form of Rule Table.....	94
Table 5-3: Rule Table for Error (e) and Change of Error (delta)	95

LIST OF FIGURES

Figure 1-1: System Process Plot.....	3
Figure 1-2: Basic Filtering Problem.....	6
Figure 1-3: Kalman Filter General Working Flow Chart.....	7
Figure 2-1: Process and Measurement Model.....	16
Figure 2-2: Kalman Filter Loop.....	23
Figure 3-1: Typical Kalman Filter Application.....	34
Figure 3-2: True Position Trajectory.....	39
Figure 3-3: Measured Position Trajectory.....	39
Figure 3-4: Estimated Position Trajectory.....	39
Figure 3-5: Innovation.....	39
Figure 3-6: Kalman Gain K	39
Figure 3-7: Measured Position Trajectory.....	40
Figure 3-8: Estimated Position Trajectory.....	40
Figure 3-9: Innovation.....	40
Figure 3-10: Kalman Gain K	40
Figure 3-11: Measured Position Trajectory.....	41
Figure 3-12: Estimated Position Trajectory.....	41
Figure 3-13: Innovation.....	41
Figure 3-14: Kalman Gain K	41
Figure 3-15: Estimated Position Trajectory.....	42

Figure 3-16: Kalman Gain $\mathbf{K}$	42
Figure 3-17: Estimated Position Trajectory.....	43
Figure 3-18: Kalman Gain $\mathbf{K}$	43
Figure 3-19: Estimated Position Trajectory.....	44
Figure 3-20: Kalman Gain $\mathbf{K}$	44
Figure 3-21: Estimated Position Trajectory.....	45
Figure 3-22: Process Noise Covariance $\mathbf{Q}$	45
Figure 3-23: Measurement Noise Covariance $\mathbf{R}$	46
Figure 3-24: Kalman Gain $\mathbf{K}$	46
Figure 3-25: Estimated Position Trajectory.....	47
Figure 3-26: Process Noise Covariance $\mathbf{Q}$	47
Figure 3-27: Measurement Noise Covariance $\mathbf{R}$	47
Figure 3-28: Kalman Gain $\mathbf{K}$	47
Figure 3-29: Estimated Position Trajectory.....	48
Figure 3-30: Kalman Gain $\mathbf{K}$	48
Figure 3-31: Estimated Position Trajectory.....	48
Figure 3-32: Kalman Gain $\mathbf{K}$	48
Figure 3-33: Estimated Position Trajectory.....	49
Figure 3-34: Kalman Gain $\mathbf{K}$	49
Figure 3-35: Estimated Position Trajectory.....	49
Figure 3-36: Kalman Gain $\mathbf{K}$	49
Figure 3-37: Estimated Position Trajectory.....	50

Figure 3-38: Kalman Gain $\mathbf{K}$	50
Figure 3-39: Estimated Position Trajectory.....	50
Figure 3-40: Kalman Gain $\mathbf{K}$	50
Figure 3-41: Kalman Gain $\mathbf{K}$	53
Figure 3-42: Estimated Position Trajectory.....	53
Figure 3-43: $\mathbf{C}_k' = 0.90\mathbf{C}_k$	54
Figure 3-44: $\mathbf{C}_k' = 0.94\mathbf{C}_k$	54
Figure 3-45: $\mathbf{C}_k' = 0.96\mathbf{C}_k$	54
Figure 3-46: $\mathbf{C}_k' = 0.99\mathbf{C}_k$	54
Figure 3-47: $\mathbf{C}_k' = 1.1\mathbf{C}_k$	55
Figure 3-48: $\mathbf{C}_k' = 1.3\mathbf{C}_k$	55
Figure 3-49: $\mathbf{C}_k' = 1.7\mathbf{C}_k$	55
Figure 3-50: $\mathbf{C}_k' = 2.0\mathbf{C}_k$	55
Figure 3-51: Kalman Gain $\mathbf{K}$	57
Figure 3-52: Estimated Position Trajectory.....	57
Figure 3-53: $\mathbf{C}_k' = 0.90\mathbf{C}_k$	57
Figure 3-54: $\mathbf{C}_k' = 0.94\mathbf{C}_k$	57
Figure 3-55: $\mathbf{C}_k' = 0.96\mathbf{C}_k$	58
Figure 3-56: $\mathbf{C}_k' = 0.99\mathbf{C}_k$	58
Figure 3-57: $\mathbf{C}_k' = 1.1\mathbf{C}_k$	58

Figure 3-58: $C'_k = 1.3C_k$	58
Figure 3-59: $C'_k = 1.7C_k$	58
Figure 3-60: $C'_k = 2.0C_k$	58
Figure 4-1: Some Typical Membership Functions.....	65
Figure 4-2 Fuzzy Union and Intersection.....	68
Figure 4-3: Fuzzy Controller Architecture.....	71
Figure 4-4: Fuzzy Set Representation.....	75
Figure 4-5: Membership Functions.....	76
Figure 4-6: Defuzzification Methods.....	79
Figure 5-1: Control System Block Diagram.....	87
Figure 5-2: Membership Functions for Temperature Input.....	88
Figure 5-3: Membership Functions for Pressure Input.....	88
Figure 5-4: Membership Functions for Throttle Setting Output.....	89
Figure 5-5: Rule 2 Evaluation.....	91
Figure 5-6: Rule 3 Evaluation.....	91
Figure 5-7: COG Defuzzification.....	92
Figure 5-8: Fuzzy Controller.....	92
Figure 6-1: Block Diagram of Fuzzy Filter.....	101
Figure 6-2: Fuzzy Dynamic System Model.....	104
Figure 6-3: Fuzzy Measurement Model.....	107
Figure 6-4: Measurement Update Block.....	108
Figure 6-5: Estimated Position Trajectory for Fuzzy Filter.....	111

Figure 6-5a: Estimated Position Trajectory for Kalman Filter.....	111
Figure 6-6: $\mathbf{e}_k$ and $\hat{\mathbf{e}}_k$ of Fuzzy Filter.....	112
Figure 6-7: $\mathbf{e}_{p_i}$ and $\hat{\mathbf{e}}_{p_i}$ of Fuzzy Filter.....	112
Figure 6-8: Estimated Position Trajectory of Fuzzy Filter with Correlated Process and Measurement Noises.....	114
Figure 6-8a: Estimated Position Trajectory of Kalman Filter with Correlated Process and Measurement Noises.....	115
Figure 6-9: $\mathbf{e}_k$ and $\hat{\mathbf{e}}_k$ of Fuzzy Filter with Correlated Process and Measurement Noises.....	115
Figure 6-10: $\mathbf{e}_{p_i}$ and $\hat{\mathbf{e}}_{p_i}$ of Fuzzy Filter with Correlated Process and Measurement Noises.....	116
Figure 7-1: Comparison of Kalman Filter and Fuzzy Filter.....	120
Figure 7-2: Comparison of Kalman Filter and Fuzzy Filter (Zoom In Part)	121
Figure 7-3: Kalman Filter.....	122
Figure 7-4: Fuzzy Filter.....	122
Figure 7-5: Kalman Filter.....	122
Figure 7-6: Fuzzy Filter.....	122
Figure 7-7: Kalman Filter.....	123
Figure 7-8: Fuzzy Filter.....	123

LIST OF SYMBOLS

$S(t)$	Signal
$N(t)$	Noise
$Z(t)$	Measurement (Signal + Noise)
A	White-noise spectral amplitude
$\mathbf{x}_k$	$(n \times 1)$ process state vector at time t_k
Φ_k	$(n \times n)$ matrix relating $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$ in the absence of a forcing function
$\mathbf{w}$	$(n \times 1)$ process vector – assumed to be a white sequence with known covariance structure
$\mathbf{z}_k$	$(m \times 1)$ vector measurement at time t_k
$\mathbf{H}$	$(m \times n)$ matrix giving the ideal (noiseless) connection between the measurement and the state vector at time t_k
$\mathbf{v}$	$(m \times 1)$ measurement vector – assumed to be a white sequence with know covariance structure and having zero crosscorrelation with the $\mathbf{w}$ sequence
$\mathbf{K}_k$	Kalman gain
$\hat{\mathbf{x}}_k$	Updated estimate

R	Measurement noise covariance
Q	Process noise covariance
P_k	Estimation error covariance
C	Covariance between process noise and measurement noise
μ	Fuzzy membership
b_i	Center of the membership function of the consequent of rule (<i>i</i>)
∫ μ_(i)	Area under the membership function μ _(i)

Chapter 1

INTRODUCTION

Experimental measurements are never perfect, even with sophisticated modern instruments. Two main types of measurement errors are recognized:

1. *Systematic error*, in which every measurement is either less than or greater than the “correct” value by a fixed percentage or amount, and
2. *Random error*, in which there are unpredictable variations in the measured signal from moment to moment or from measurement to measurement. This type of error is often called *noise*, by analogy to acoustic noise.

There are many sources of noise in physical measurements, such as building vibrations, air currents, electric power fluctuations, stray radiation from nearby electrical

apparatus, interference from radio and TV transmissions, random thermal motion of molecules, and even the basic quantum nature of matter and energy itself.

Noise may be generated within the electrical components of the input amplifier: *internal noise*. It may also be added to the signal as it travels down the input wires to the amplifier: *external noise*.

Internal noise arises from thermal effects in the amplifier and although it can be minimized, it cannot be removed completely. External noise reduction is much more of an art form. Noise is added because the signal leads act as aerials picking up environmental electrical activity. External noise has usually a nonzero correlation time and is thus colored noise. [1]

One of the fundamental problems in signal measurement is distinguishing the noise from the signal. From a typical example, we can see this problem more clearly. In Figure 1-1, we have (1) *signal*, which is a sinusoid signal source, $x = 10 \sin(t)$, (2) *noise*, which simulates normally distributed random numbers, and (3) *signal + noise*, which is the actual signal we get from the measurement device.

Figure 1-1: System Process Plot

In Figure 1-1, although we still can tell that the shape of measurement signal (bottom plot) is more or less the sinusoid, we cannot find the characteristics of the signal because of the influence of noise. We can derive the equation for this process as:

$$Z(t) = S(t) + N(t) \quad (1-1)$$

Where

$S(t)$ — Signal;

$N(t)$ — Noise;

$Z(t)$ — Measurement (Signal + Noise);

The noise can be generally divided into two types: *white noise* and *non-white (colored) noise*. White noise is defined to be a stationary process having a constant spectral density function. The term “white” is a carryover from optics, where white light is the light containing all visible frequencies. It is also called random, Gaussian or steady state noise. Denoting the white-noise spectral amplitude as A , we then have (subscript wn stands for white noise)

$$S_{wn}(j\omega) = A \quad (1-2)$$

The corresponding autocorrelation function for white noise is then

$$R_{wn}(\tau) = A\delta(\tau) \quad (1-3)$$

A *white sequence* is a sequence of zero-mean, uncorrelated random variables. That is, all members of the sequence have zero means and are mutually uncorrelated with all other members of the sequence. The covariance matrix is diagonal, i.e., each element of the diagonal is the variance of the corresponding pixel, and there is no correlation between the effect of noise on one pixel and the effect of noise on another pixel. If the

random variables are also normal, then the sequence is a Gaussian white sequence. It is the form of the autocorrelation and power spectrum that determines whether or not noise is white.

With a large number of investigations have been carried out, there has been much interest in characterization of underwater acoustic noise, urban and man-made RF noise, low-frequency atmospheric noise, and radar clutter noise. For such situations, it is well-recognized that the mathematically appealing simple white noise model is often not at all appropriate. So, non-white noise models were introduced. One feature of non-white noise model is that it specifies noise density functions which in its tail decay at rates lower than of decay of the white density tails. The essential implication of this fact is that non-white noise which tends to occur in practice is generally more likely to produce large-magnitude observations that would be predicted by a white noise model (satisfying the same constraint on some scale characteristic, such as noise percentile). [2]

For non-white noise (colored noise, time-correlated Gaussian noise), the members of the sequence are not mutually uncorrelated with all other members of the sequence, and its correlation at different times are not proportional to delta function. The covariance matrix has non-zero elements off the diagonal, there exists a correlation between the effect of noise on one pixel and effect on another. For example, Ornstein-Uhlenbeck process is a Gaussian process of zero mean and correlation [3]

$$C(t, t') = \left(\frac{D}{\tau} \right)^{\frac{|t-t'|}{\tau}} \quad (1-4)$$

where D measures the noise intensity and τ is its correlation time. The difficulty of the colored noise problem is due to the nonvanishing value of τ . If $\tau \rightarrow 0$, the stochastic process becomes a white noise.

In [4], a filter is defined as a system that processes a signal in some desired fashion. The purpose of any filter is to separate one thing from another. In the electric filter case, this usually refers to passing signals in a specified frequency range and rejecting those outside that range; and historically, filter theory began with the problem of designing a circuit to yield the desired frequency response. Filtering reduces noise errors in the signal, so that we can have the true or desired signal. In Figure 1-2, it indicates the basic filtering problem. There are many kinds of filters could be used for noise removal purpose, such as mean filter, median filter, Gaussian filter, and so on. In this thesis, Kalman filter and fuzzy filter utilizing fuzzy logic technique have been discussed.

Figure 1-2: Basic Filtering Problem

The commonly used Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) solution of the least-squares method. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown.

The general working flow chart of Kalman filter can be shown as Figure 1-3:

Figure 1-3: Kalman Filter General Working Flow Chart

An important feature of the Kalman filter, not present in other statistical predictors, is its ability to adjust its own parameters according to the statistics of the

measurements from each image. In other filtering methods, the percentage of the error (between the measured and the expected data used to adjust the model parameters) is kept as a constant for the entire image sequence, or is adjusted by the programmer. However, the Kalman filter allows this percentage to change automatically according to the current confidence in the accuracy of the state parameters.

Fuzzy logic is an approach to computing based on “degrees of truth” rather than the usual “true or false” (1 or 0) boolean logic on which the modern computer is based. Natural language (like most other activities in life and indeed the universe) is not easily translated into the absolute terms of 0 and 1. (Whether everything is ultimately describable in binary terms is a philosophical question worth pursuing, but in practice much data we might want to feed a computer is in some state in between and so, frequently, are the results of computing.)

Fuzzy logic includes 0 and 1 as extreme cases of truth (or “the state of matters” or “fact”) but also includes the various states of truth in between so that, for example, the result of a comparison between two things could be not “tall” or “short” but “0.38 of tallness”.

Fuzzy set theory implements classes or groupings of data with boundaries that are not sharply defined, i.e., fuzzy. Any methodology or theory implementing “crisp” definitions such as classical set theory, arithmetic, and programming, may be “fuzzified” by generalizing the concept of a crisp set to a fuzzy set with blurred boundaries. The

benefit of extending crisp theory and analysis methods to fuzzy techniques is the strength in solving real-world problems, which inevitably entail some degree of imprecision and noise in the variables and parameters measured and processed for the application. Accordingly, linguistic variables are a critical aspect of some fuzzy logic applications, where general terms such as “large”, “medium”, and “small” are each used to capture a range of numerical values. While similar to conventional quantization, fuzzy logic allows these stratified sets to overlap (e.g., a 1.80m high man may be classified in both the “tall” and “medium tall” categories, with varying degrees of belonging or membership to each group). Fuzzy set theory encompasses fuzzy logic, fuzzy arithmetic, fuzzy mathematical programming, fuzzy topology, fuzzy graph theory, and fuzzy data analysis, though the term fuzzy logic is often used to describe all of these.

Fuzzy logic was conceived as a better method for sorting and handling data but has proven to be an excellent choice for many control system applications since it mimics human control logic. It can be built into anything from small, hand-held products to large computerized process control systems. It uses an imprecise but very descriptive language to deal with input data more like a human operator. It is very robust and forgiving of operator and data input and often works when first implemented with little or no tuning.

Fuzzy systems, including fuzzy logic and fuzzy set theory, provide a rich and meaningful addition to standard logic. The mathematics generated by these theories is consistent. The applications which may be generated from or adapted to fuzzy logic are wide-ranging, and provide the opportunity for modeling of conditions which are

inherently imprecisely defined, despite the concerns of classical logicians. Many systems may be modeled, simulated, and even replicated with the help of fuzzy systems, not the least of which is human reasoning itself.

The outline of this thesis is as follows:

❖ **Chapter 2 – *Kalman Filter***

In this chapter, introduction to the Discrete Kalman filter and Extended Kalman filter is presented. Including the derivation of discrete and extended Kalman filter; parameter tuning; linearization, etc.

❖ **Chapter 3 – *Application of Kalman Filter for Noise Reduction***

We use a typical example to demonstrate how the discrete Kalman filter works, and how the choices of different parameters will influence the effect of Kalman filter, mainly focus on the Kalman gain K . Also, the influence of correlated process noise and measurement noise to Kalman filter is introduced.

❖ **Chapter 4 – *Fuzzy Logic and Fuzzy Control***

A brief description of fuzzy logic, including fuzzy sets, membership functions, rule-base, etc., is illustrated. Also, the advantages of fuzzy systems are discussed, elements of fuzzy system and procedures of fuzzy control are described too.

❖ **Chapter 5 – *Design of Fuzzy System***

In addition to the discussion of general methodology of fuzzy controller design, from a typical design problem, we can make more sophisticated understanding of fuzzy control in practical use. Skills such as rule design, performance tuning, rule-base tuning are presented.

❖ **Chapter 6 – *Fuzzy Filter and Application***

In this chapter, the main construction and important elements of fuzzy filter are described, and the application of fuzzy filter is illustrated.

❖ **Chapter 7 – *Comparison Between Kalman Filter and Fuzzy Filter***

The comparison is made between Kalman filter and fuzzy filter for the purpose of noise removal, which leads us to get better idea for the utilization of each filter.

❖ **Chapter 8 – *Conclusion***

Chapter 2

KALMAN FILTER

2.1 Introduction to Kalman Filter

In 1960, R.E. Kalman published his famous paper describing a recursive solution to the discrete-data linear filtering problem. Since that time, due in large part to advances in digital computing, the Kalman filter has been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation.

The Kalman filter is a linear, recursive estimator that produces the minimum variance estimate in a least squares sense under the assumption of white, Gaussian noise processes. Because the filter is a linear estimator by definition, for navigation systems it generally estimates errors in the total navigation state. The Kalman filter also produces a measure of the accuracy of its error state vector estimate. This level of accuracy is the

matrix of second central moments of the errors in the estimate and is defined as the covariance matrix.

There are two basic processes that are modeled by a Kalman filter. The first process is a model describing how the error state vector changes in time. This model is the *system dynamics model*. The second model defines the relationship between the error state vector and any measurements processed by the filter, and it is the *measurement model*. Intuitively, the Kalman filter sorts out information and weights the relative contributions of the measurements and of the dynamic behavior of the state vector. The measurements and state vector are weighted by their respective covariance matrices. If the measurements are inaccurate (large variances) when compared to the state vector estimate, then the filter will make the measurements weight less. On the other hand, if the measurements are very accurate (small variances) when compared to the state estimate, then the filter will tend to weight the measurements heavily with the consequence that its previously computed state estimate will contribute little to the latest state estimate.

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: *time update equations* and *measurement update equations*. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update

equations are responsible for the feedback – i.e. for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate.

The Kalman filter is supplied with initial information, including the measurement error covariance, and estimates of the initial parameters and associated error, and these are used to calculate a gain matrix. The error between the parameter estimates and the measured data is determined and multiplied by a gain matrix to update the parameter estimate and estimated error. The updated error and parameters are used as input to a model of behavior, to predict the projected error and parameters at the next time instance.

Initially, when the model parameters are only rough estimates, the gain matrix ensures that the measurement data is highly influential in estimating the state parameters. Then, as confidence in the accuracy of the parameters grows with each iteration, the gain matrix values decrease, causing the influence of the measurement data in updating the parameters and associated error to lessen.

2.2 The Discrete Kalman Filter

2.2.1 Discrete-Time Model

Discrete-time processes may arise in either of two ways. [5] First, there is the situation where a sequence of events takes place naturally in discrete steps. This might be the result of a sequence of chance experiments such as the discrete random-walk

problems of statistics. Also, it may arise from sampling a continuous process at discrete times. The sampling may be intentional and may be under the control of the designer, as is the case when analog data are converted to digital form. Or sometimes the sampling may be unintentional and forced by a measurement constraint that allows observation of the process only at discrete points in time.

2.2.2 Derivation of Discrete Kalman Filter

The discrete Kalman filter is a recursive predictive update technique used to determine the correct parameters of a process model. Given some initial estimates, it allows the parameters of a model to be predicted and adjusted with each new measurement, providing an estimate of error at each update. Its ability to incorporate the effects of noise (from both measurement and modeling), and its computational structure, has made it very popular for use in computer vision tracking applications. It has been suggested that, in the right situations, it performs better than any other linear filter.

A feature of the Kalman filter, not present in other statistical predictors, is its ability to adjust its own parameters according to the statistics of the measurements from each image. In other filtering methods, such as the alpha-beta filter, the percentage of the error (between the measured and the expected data used to adjust the model parameters) is kept as a constant for the entire image sequence, or is adjusted by the designer. However, the Kalman filter allows this percentage to change automatically according to the current confidence in the accuracy of the state parameters.

In this section, the recursive equations of discrete Kalman filter are derived. We begin by the block diagram of random signal process and measurement in Figure 2-1:

Figure 2-1: Process and Measurement Model

We assume the random process to be estimated can be modeled in the form [6]

$$\mathbf{x}_{k+1} = \Phi_k \mathbf{x}_k + \mathbf{w}_k \quad (2-1)$$

The observation (measurement) of the process is assumed to occur as discrete points in time in accordance with the linear relationship

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \quad (2-2)$$

Some elaboration on notation and the various terms of the equations above is in order:

- $\mathbf{x}_k$ — $(n \times 1)$ process state vector at time t_k ;
- Φ_k — $(n \times n)$ matrix relating $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$ in the absence of a forcing function (If $\mathbf{x}_k$ is a sample of continuous process, Φ_k is the usual state transition matrix.);
- $\mathbf{w}_k$ — $(n \times 1)$ process vector – assumed to be a white sequence with known covariance structure;
- $\mathbf{z}_k$ — $(m \times 1)$ vector measurement at time t_k ;
- $\mathbf{H}_k$ — $(m \times n)$ matrix giving the ideal (noiseless) connection between the measurement and the state vector at time t_k ;
- $\mathbf{v}_k$ — $(m \times 1)$ measurement vector – assumed to be a white sequence with known covariance structure and having zero crosscorrelation with the $\mathbf{w}_k$ sequence;

The covariance matrices for the $\mathbf{w}_k$ and $\mathbf{v}_k$ vectors are given by

$$E[\mathbf{w}_k \mathbf{w}_i^T] = \begin{cases} \mathbf{Q}_k, & i = k \\ 0, & i \neq k \end{cases} \quad (2-3)$$

$$E[\mathbf{v}_k \mathbf{v}_i^T] = \begin{cases} \mathbf{R}_k, & i = k \\ 0, & i \neq k \end{cases} \quad (2-4)$$

$$E[\mathbf{w}_k \mathbf{v}_i^T] = 0, \text{ for all } k \text{ and } i \quad (2-5)$$

We assume at this point that we have an initial estimate of the process at some point in time t_k , and that this estimate is based on all our knowledge about the process prior to t_k . This prior (or *a priori*) estimate will be denoted as $\hat{\mathbf{x}}_k^-$ where the “hat” denotes estimate, and the “super minus” is a reminder that this is our best estimate prior to assimilating the measurement at t_k . (Note that super minus as used here is not related in any way to the super minus notation used in spectral factorization.) We also assume that we know the error covariance matrix associated with $\hat{\mathbf{x}}_k^-$. That is, we define the estimation error to be

$$\mathbf{e}_k^- = \mathbf{x}_k - \hat{\mathbf{x}}_k^- \quad (2-6)$$

and the associated error covariance matrix is

$$\mathbf{P}_k^- = E[\mathbf{e}_k^- \mathbf{e}_k^{-T}] = E\left[\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^-\right)\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^-\right)^T\right] \quad (2-7)$$

With the assumption of a prior estimate $\hat{\mathbf{x}}_k^-$, we now seek to use the measurement $\mathbf{z}_k$ to improve the prior estimate. We choose a linear blending of the noisy measurement and the prior estimate in accordance with the equation

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^- \right) \quad (2-8)$$

where

$\hat{\mathbf{x}}_k$ — updated estimate;

$\mathbf{K}_k$ — blending factor (yet to be determined);

The difference in equation (2-8) is called the measurement innovation, or the residual. The residual reflects the discrepancy between the predicted measurement and the actual measurement, and it is a white Gaussian sequence of mean zero and covariance $[\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k]$. This can be exploited for the practical purpose of either sensor failure detection or reasonableness checking of measurement data. The preceding is a proper characterization of the residual process provided that the mathematical model upon which the filter was based accurately depicts the real system behavior. During the operation of the filter, the actual residual sequence realization can be monitored and compared to this description. If the description appears valid up to a point in time, thereafter being violated consistently, one can deduce that something occurred in the real system to invalidate the model within the filter. If the violation occurs in only one component of the vector residual process, one can further deduce that the measuring device generating that particular residual component is the source of difficulty: a failure (soft or hard) can be declared in that sensor. A residual of zero means that the two are in complete agreement.

The problem now is to find the particular blending factor $\mathbf{K}_k$ that yields an updated estimate that is optimal in some sense. We use minimum mean-square error as the performance criterion. Towards this end we first form the expression for the error covariance matrix associated with the updated (*a posterior*) estimate.

$$\mathbf{P}_k = E[\mathbf{e}_k \mathbf{e}_k^T] = E\left[\left(\mathbf{x}_k - \hat{\mathbf{x}}_k\right)\left(\mathbf{x}_k - \hat{\mathbf{x}}_k\right)^T\right] \quad (2-9)$$

Substituting measurement equation (2-2) into (2-8) and then entering the result expression for $\hat{\mathbf{x}}_k$ into equation (2-9). The result is

$$\begin{aligned} \mathbf{P}_k = E\left\{\left[\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^-\right) - \mathbf{K}_k \left(\mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-\right)\right] \right. \\ \left. \left[\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^-\right) - \mathbf{K}_k \left(\mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-\right)\right]^T\right\} \end{aligned} \quad (2-10)$$

Now, performing the indicated expectation and noting the $(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)$ is the a prior estimation error that is uncorrelated with measurement error $\mathbf{v}_k$, we have

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T \quad (2-11)$$

The matrix $\mathbf{K}$ is chosen to be the gain or blending factor that minimizes the a posteriori error covariance in equation (2-9). This minimization can be accomplished by first substituting equation (2-8) into the above definition for $\mathbf{e}_k$, substituting that into equation (2-9), performing the indicated expectations, taking the derivative of the trace of the result with respect to $\mathbf{K}$, setting that result equal to zero, and then solving for $\mathbf{K}$. One form of the resulting $\mathbf{K}$ that minimizes equation (2-9) is given by

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \quad (2-12)$$

This particular $\mathbf{K}_k$, namely the one that minimizes the mean-square estimation error, is called the *Kalman gain*.

The covariance matrix associated with the optimal estimate may be computed.

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \mathbf{H}_k \mathbf{P}_k^- \quad (2-13)$$

or

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k) \mathbf{K}_k^T \quad (2-14)$$

or

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- \quad (2-15)$$

The updated estimated $\hat{\mathbf{x}}_k$ is easily projected ahead via the transition matrix. We ignore the contribution of $\mathbf{w}_k$ in equation (2-1) because it has zero mean and is not correlated with any of the previous $\mathbf{w}_k$'s. Thus we have

$$\hat{\mathbf{x}}_{k+1}^- = \Phi_k \hat{\mathbf{x}}_k \quad (2-16)$$

The error covariance matrix associated with $\hat{\mathbf{x}}_{k+1}^-$ is obtained by first forming the expression for the prior error

$$\begin{aligned} \mathbf{e}_{k+1}^- &= \mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1}^- \\ &= (\Phi_k \mathbf{x}_k + \mathbf{w}_k) - \Phi_k \hat{\mathbf{x}}_k \\ &= \Phi_k \mathbf{e}_k + \mathbf{w}_k \end{aligned} \quad (2-17)$$

We now note that $\mathbf{w}_k$ and $\mathbf{e}_k$ have zero crosscorrelation, because $\mathbf{w}_k$ is the process noise for the step ahead of t_k . Thus we can write the expression for $\mathbf{P}_{k+1}^-$ as

$$\begin{aligned} \mathbf{P}_{k+1}^- &= E[\mathbf{e}_{k+1}^- \mathbf{e}_{k+1}^{-T}] = E[(\Phi_k \mathbf{x}_k + \mathbf{w}_k)(\Phi_k \mathbf{x}_k + \mathbf{w}_k)^T] \\ &= \Phi_k \mathbf{P}_k \Phi_k^T + \mathbf{Q}_k \end{aligned} \quad (2-18)$$

It should be clear that once the loop is entered, it can be continued ad infinitum. The pertinent equations and the sequence of computation step are shown pictorially in Figure 2-2. This summarizes what is known as the *Kalman filter*.

Figure 2-2: Kalman Filter Loop

2.3 Filter Parameters and Tuning

In the actual implementation of the filter, the measurement noise covariance $\mathbf{R}$ is usually measured prior to operation of the filter. Measuring the measurement error covariance $\mathbf{R}$ is generally practical (possible) because we need to be able to measure the process anyway (while operating the filter) so we should generally be able to take some off-line sample measurements in order to determine the variance of the measurement noise.

The determination of the process noise covariance $\mathbf{Q}$ is generally more difficult as we typically do not have the ability to directly observe the process we are estimating. Sometimes a relatively simple (poor) process model can produce acceptable results if one "injects" enough uncertainty into the process via the selection of $\mathbf{Q}$. Certainly in this case one would hope that the process measurements are reliable.

In either case, whether or not we have a rational basis for choosing the parameters, often times superior filter performance (statistically speaking) can be obtained by tuning the filter parameters $\mathbf{Q}$ and $\mathbf{R}$. The tuning is usually performed off-line, frequently with the help of another (distinct) Kalman filter in a process generally referred to as system identification.

In closing we note that under conditions where $\mathbf{Q}$ and $\mathbf{R}$ are in fact constant, both the estimation error covariance $\mathbf{P}_k$ and the Kalman gain $\mathbf{K}_k$ will stabilize quickly and then

remain constant (see the filter update equations in Figure 2-2). If this is the case, these parameters can be pre-computed by either running the filter off-line, or for example by determining the steady-state value of $\mathbf{P}_k$.

It is frequently the case however that the measurement error (in particular) does not remain constant. Also, the process noise $\mathbf{Q}$ is sometimes changed dynamically during filter operation – becoming $\mathbf{Q}_k$ – in order to adjust to different dynamics. For example, in the case of tracking the head of a user of a 3D virtual environment we might reduce the magnitude of $\mathbf{Q}_k$ if the user seems to be moving slowly, and increase the magnitude if the dynamics start changing rapidly. In such cases $\mathbf{Q}_k$ might be chosen to account for both uncertainty about the user's intentions and uncertainty in the model.

2.4 Linearization

Some of the most successful applications of Kalman filtering have been in situations with nonlinear dynamics and /or nonlinear measurement relationships. We now examine two basic ways of linearizing the problem. One is to linearize about some nominal trajectory in state space that does not depend on the measurement data. The resulting filter is usually referred to as simply a *linearized Kalman filter*. The other method is to linearize about a trajectory that is continually updated with the state estimates resulting from the measurements. When this is done, the filter is called an *extended Kalman filter*.

2.5 Linearized Kalman Filter

The process and associated measurement relationship may be written in the form

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}_d, t) + \mathbf{u}(t) \\ \mathbf{z} &= \mathbf{h}(\mathbf{x}, t) + \mathbf{v}(t)\end{aligned}\tag{2-19}$$

where $\mathbf{f}$ and $\mathbf{h}$ are known functions, $\mathbf{u}_d$ is a deterministic forcing function, and $\mathbf{u}$ and $\mathbf{v}$ are white noise process with zero crosscorrelation. Note that the forms of these equations are somewhat restrictive in that $\mathbf{u}$ and $\mathbf{v}$ are assumed to be separate additive terms and are not included with the $\mathbf{f}$ and $\mathbf{h}$ terms.

The actual trajectory $\mathbf{x}(t)$ and the approximate trajectory (nominal or reference trajectory) $\mathbf{x}^*(t)$ have the relationship as:

$$\mathbf{x}(t) = \mathbf{x}^*(t) + \Delta\mathbf{x}(t)\tag{2-20}$$

So, we have the updated equations

$$\begin{aligned}\dot{\mathbf{x}}^* + \Delta\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}^* + \Delta\mathbf{x}, \mathbf{u}_d, t) + \mathbf{u}(t) \\ \mathbf{z} &= \mathbf{h}(\mathbf{x}^* + \Delta\mathbf{x}, t) + \mathbf{v}(t)\end{aligned}\tag{2-21}$$

We now assume $\Delta \mathbf{x}$ is small and approximate the $\mathbf{f}$ and $\mathbf{h}$ functions with Taylor's series expansions, retaining only first-order terms. The result is

$$\dot{\mathbf{x}}^* + \Delta \dot{\mathbf{x}} \approx \mathbf{f}(\mathbf{x}^*, \mathbf{u}_d, t) + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right]_{\mathbf{x}=\mathbf{x}^*} \cdot \Delta \mathbf{x} + \mathbf{u}(t) \quad (2-22)$$

$$\mathbf{z} \approx \mathbf{h}(\mathbf{x}^*, t) + \left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right]_{\mathbf{x}=\mathbf{x}^*} \cdot \Delta \mathbf{x} + \mathbf{v}(t)$$

where

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}; \quad \frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \dots \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} \quad (2-23)$$

It is customary to choose the nominal trajectory $\mathbf{x}^*(t)$ to satisfy the deterministic differential equation.

$$\dot{\mathbf{x}}^* = \mathbf{f}(\mathbf{x}^* + \Delta \mathbf{x}, \mathbf{u}_d, t) \quad (2-24)$$

The linearized model is

$$\begin{aligned}\Delta \dot{\mathbf{x}} &= \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right]_{\mathbf{x}=\mathbf{x}^*} \cdot \Delta \mathbf{x} + \mathbf{u}(t), & \text{(linearized dynamics)} \\ \left[\mathbf{z} - \mathbf{h}(\mathbf{x}^*, t) \right] &= \left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right]_{\mathbf{x}=\mathbf{x}^*} \cdot \Delta \mathbf{x} + \mathbf{v}(t), & \text{(linearized measurement equation)}\end{aligned}\tag{2-25}$$

The “measurement” in the linear model is the actual measurement less that predicted by the nominal trajectory in the absence of noise. Also the equivalent **F** and **H** matrices are obtained by evaluating the partial derivative matrices along the *nominal* trajectory.

2.6 The Extended Kalman Filter (EKF)

The extended Kalman filter is similar to a linearized Kalman filter except that the linearization takes place about the filter’s estimated trajectory, rather than a pre-computed nominal trajectory. That is, the partial derivative of (2-23) are evaluated along a trajectory that has been updated with the filter’s estimates; these, in turn, depend on the measurements, so that filter gain sequence will depend on the sample measurement sequence realized on a particular run of the experiment. Thus, the gain sequence is not predetermined by the process model assumptions as in the usual Kalman filter.

A general analysis of the extended Kalman filter is difficult because of the feedback of the measurement sequence into the process model. However, qualitatively it would seem to make sense to update the trajectory that is used for the linearization.

Both the regular linearized Kalman filter and the extended Kalman filter have been used in a variety of applications. Each has its advantages and disadvantages, and no general statement can be made as to which is best because it depends on the particular situation at hand. The aided inertial navigation application serves as a good example to illustrate the difference between these two forms of linearization.

Let us assume that the process has a state vector $\mathbf{x}$, and it is now governed by the *nonlinear* stochastic difference equation

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_{k-1}) \quad (2-26)$$

With a measurement $\mathbf{z}$ that is

$$\mathbf{z}_k = h(\mathbf{x}_k, \mathbf{v}_k) \quad (2-27)$$

where the random variables $\mathbf{w}_k$ and $\mathbf{v}_k$ represent the process and measurement noise.

In practice, one does not know the individual values of the noise $\mathbf{w}_k$ and $\mathbf{v}_k$ at each time step. However, one can approximate the state and measurement vector without them as

$$\bar{\mathbf{x}}_k = f\left(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k, 0\right) \quad (2-28)$$

$$\bar{\mathbf{z}}_k = h\left(\bar{\mathbf{x}}_k, 0\right) \quad (2-29)$$

where $\hat{\mathbf{x}}_k$ is some *a posteriori* estimate of the state (from a previous time step k).

To estimate a process with non-linear difference and measurement relationships, we begin by writing new governing equations that linearize an estimate about equations (2-28) and (2-29)

$$\mathbf{x}_k \approx \bar{\mathbf{x}}_k + \Phi\left(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}\right) + \mathbf{W}\mathbf{w}_{k-1} \quad (2-30)$$

$$\mathbf{z}_k \approx \bar{\mathbf{z}}_k + \mathbf{H}\left(\mathbf{x}_k - \bar{\mathbf{x}}_k\right) + \mathbf{V}\mathbf{v}_k \quad (2-31)$$

where

- $\mathbf{x}_k$ and $\mathbf{z}_k$ are the actual state and measurement vectors;
- $\bar{\mathbf{x}}_k$ and $\bar{\mathbf{z}}_k$ are the approximate state and measurement vectors from equations (2-28) and (2-29);

- $\hat{\mathbf{x}}_k$ is an *a posteriori* estimate of the state at step k ;
- random variables $\mathbf{w}_k$ and $\mathbf{v}_k$ represent the process and measurement noise;
- Φ is the Jacobian matrix of partial derivatives of f with respect to $\mathbf{x}$;

$$\Phi_{[i,j]} = \frac{\partial f_{[i]}}{\partial \mathbf{x}_{[j]}} \left(\hat{\mathbf{x}}_k, \mathbf{u}_k, 0 \right) \quad (2-32)$$

- $\mathbf{W}$ is the Jacobian matrix of partial derivatives of f with respect to $\mathbf{w}$;

$$\mathbf{W}_{[i,j]} = \frac{\partial f_{[i]}}{\partial \mathbf{w}_{[j]}} \left(\hat{\mathbf{x}}_k, \mathbf{u}_k, 0 \right) \quad (2-33)$$

- $\mathbf{H}$ is the Jacobian matrix of partial derivatives of h with respect to $\mathbf{x}$;

$$\mathbf{H}_{[i,j]} = \frac{\partial h_{[i]}}{\partial \mathbf{x}_{[j]}} \left(\bar{\mathbf{x}}_k, 0 \right) \quad (2-34)$$

- $\mathbf{V}$ is the Jacobian matrix of partial derivatives of h with respect to $\mathbf{v}$;

$$\mathbf{V}_{[i,j]} = \frac{\partial h_{[i]}}{\partial \mathbf{v}_{[j]}} \left(\bar{\mathbf{x}}_k, 0 \right) \quad (2-35)$$

The complete set of EKF equations is shown below.

EKF time update equations:

$$\hat{\mathbf{x}}_k^- = f \left(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k, 0 \right) \quad (2-36)$$

$$\mathbf{P}_k^- = \Phi_k \mathbf{P}_{k-1} \Phi_k^T + \mathbf{W}_k \mathbf{Q}_{k-1} \mathbf{W}_k^T \quad (2-37)$$

EKF measurement update equations:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{V}_k \mathbf{R}_k \mathbf{V}_k^T)^{-1} \quad (2-38)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (z_k - \mathbf{h}(\hat{\mathbf{x}}_k^-, 0)) \quad (2-39)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- \quad (2-40)$$

An important feature of the EKF is that the Jacobian $\mathbf{H}_k$ in the equation for the Kalman gain $\mathbf{K}_k$ serves to correctly propagate or "magnify" only the relevant component of the measurement information. For example, if there is not a one-to-one mapping between the measurement z_k and the state via $\mathbf{h}$, the Jacobian $\mathbf{H}_k$ affects the Kalman gain so that it only magnifies the portion of the residual $z - \mathbf{h}(\mathbf{x}^*, t)$ that does affect the state. Of course if over all measurements there is not a one-to-one mapping between the measurement z_k and the state via $\mathbf{h}$, then as we might expect the filter will quickly diverge.

Chapter 3

APPLICATION OF KALMAN FILTER FOR NOISE REDUCTION

3.1 Introduction

There are many examples in engineering where filtering is desirable. Radio communications signals are often corrupted with noise. A good filtering algorithm can remove the noise from electromagnetic signals while still retaining the useful information.

As introduced in previous chapter, the Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) solution of the least-

squares method. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown.

Figure 3-1 depicts a typical situation in which a Kalman filter could be used advantageously. A system of some sort is driven by some known control, and measuring devices provide the value of certain pertinent quantities. Knowledge of these systems inputs and outputs is all that is explicitly available from the physical system for estimation purposes. [7]

Figure 3-1: Typical Kalman Filter Application

3.2 Example of Discrete Kalman Filter Application

3.2.1 System Model

Consider the problem of estimating the variables of some system. In dynamic systems, the system variables are often denoted by the term *state variables*. Assume that the system variables, represented by the vector $\mathbf{x}$, are governed by equation (3-1), same as equation (2-1),

$$\mathbf{x}_{k+1} = \Phi_k \mathbf{x}_k + \mathbf{w}_k \quad (3-1)$$

where $\mathbf{w}_k$ is random process noise, and the subscripts on the vectors represent the time step.

Here, we use a simple dynamic system to demonstrate the application of Kalman filter for noise removal. For instance, if a dynamic system consists of a spacecraft which is accelerating with random bursts of gas from its reaction control system thrusters, the vector $\mathbf{x}$ might consist of position $\mathbf{P}$ and velocity $\mathbf{V}$. The dynamic equations are

$$\begin{aligned} \mathbf{P}_{k+1} &= \mathbf{P}_k + \mathbf{V}_k \Delta t + \frac{1}{2} \mathbf{a}_k \Delta t^2 \\ \mathbf{V}_{k+1} &= \mathbf{V}_k + \mathbf{a}_k \Delta t \end{aligned} \quad (3-2)$$

And the system equation is

$$\begin{bmatrix} \mathbf{P}_{k+1} \\ \mathbf{V}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_k \\ \mathbf{V}_k \end{bmatrix} + \begin{bmatrix} \frac{1}{2} \Delta t^2 \\ \Delta t \end{bmatrix} \mathbf{a}_k \quad (3-3)$$

where $\mathbf{a}_k$ is the random, time-varying acceleration and Δt is the time between step k and step $k+1$. Now suppose we can measure the position $\mathbf{P}$. Then our measurement at time k can be denoted $\mathbf{z}_k = \mathbf{P}_k + \mathbf{v}_k$, where $\mathbf{v}_k$ is random measurement noise.

We want to formulate an estimation algorithm such that the expected value of our estimate is equal to the expected value of the state. That is, “on average”, our estimate of the state will equal the true state, and all possible estimation algorithms, our algorithm minimizes the expected value of the square of the estimation error. That is, “on average”, our algorithm gives the “smallest” possible estimation error.

The Kalman filter is formulated as follows. Suppose we assume that the process noise $\mathbf{w}_k$ is white Gaussian noise with a covariance matrix $\mathbf{Q}$. Further assume that the measurement noise $\mathbf{v}_k$ is white Gaussian noise with a covariance matrix $\mathbf{R}$, and that it is not correlated with the process noise.

$$\text{State transition matrix } \Phi_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \quad (3-4)$$

$$\text{Measurement matrix } \mathbf{H}_k = [1 \ 0] \quad (3-5)$$

$$\text{Process noise matrix } \mathbf{w}_k = \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} * \text{process noise} \quad (3-6)$$

$$\text{Measurement noise } \mathbf{v}_k = \text{measurement noise} \quad (3-7)$$

According to equations (2-3) and (2-4),

$$\begin{aligned} \mathbf{Q}_k &= E[\mathbf{w}_k \mathbf{w}_k^T] \\ &= E\left[\begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} * \text{process noise} * \begin{bmatrix} \frac{1}{2}\Delta t^2 & \Delta t \end{bmatrix} * \text{process noise}\right] \end{aligned} \quad (3-8)$$

$$= \begin{bmatrix} \frac{1}{4}\Delta t^4 & \frac{1}{2}\Delta t^3 \\ \frac{1}{2}\Delta t^3 & \Delta t^2 \end{bmatrix} * \text{process noise}^2$$

$$\begin{aligned} \mathbf{R}_k &= E[\mathbf{v}_k \mathbf{v}_k^T] \\ &= \text{measurement noise}^2 \end{aligned} \quad (3-9)$$

3.2.2 Simulation Results

By using the equations in Figure 2-2 Kalman filter loop, we designed a simulation program in Matlab¹. The initial parameters chosen for the simulation are:

- True position trajectory: $x_k = 10 \sin(t_k)$;
- Standard deviation of position measurement noise: 10 m;
- Standard deviation of acceleration process noise: 0.5 m/s^2 ;
- Total simulation time period: 100 sec.;
- Time step Δt : 0.2 sec.;

From the plots of Figure 3-2 – 3-6, we can see that Kalman Filter works very well in spite of large measurement noise. It is able to estimate the position and filter the noises. After filtering, the estimated position trajectory matches the true position trajectory, and Kalman gain K converges to a certain value and gets stable at 0.0613 in this case.

1. Matlab is a commercially available software package that specializes in matrix computations and supports an interpretive script-like language. The package is available from The Math-Works, Inc. in Natick, Massachusetts.

Figure 3-2: True Position Trajectory

Figure 3-3: Measured Position Trajectory

Figure 3-4: Estimated Position Trajectory

Figure 3-5: Innovation

Figure 3-6: Kalman Gain K

3.2.3 Parameter Tuning and Analysis

3.2.3.1 Time Step

➤ **Change Time Step from 0.2 sec. to 0.5 sec.**

Figure 3-7: Measured Position Trajectory

Figure 3-8: Estimated Position Trajectory

Figure 3-9: Innovation

Figure 3-10: Kalman Gain K

— Notes

The figures above show that with the increment of the time step, the Kalman gain changes accordingly, in this case, $K = 0.1462$, which is larger than 0.0613 in previous state (Time step = 0.2 sec.). Also, from the figure of estimated position trajectory (Figure 3-8), it can be seen that the effect of Kalman filter is getting worse.

➤ Change Time Step from 0.2 sec. to 0.1 sec.

Figure 3-11: Measured Position Trajectory

Figure 3-12: Estimated Position Trajectory

Figure 3-13: Innovation

Figure 3-14: Kalman Gain K

— Notes

These figures give another point of view for the change of the time step, which is to reduce it, and under this circumstance, **K** becomes 0.0311, which is less than 0.0613 in time step = 0.2 sec. And, the Kalman filter is working better, we can see that from the plot of the estimated position trajectory (Figure 3-12).

— Discussion

From these two cases, enlarge the time step and reduce the time step, Kalman gain **K** is influenced respectively, and they will affect the final results of the filter too. We can see that the smaller the time step, the better the filter works.

3.2.3.2 Signal Frequency

➤ Change Signal Frequency From $1/(2\pi)$ to $1/(4\pi)$

Figure 3-15: Estimated Position Trajectory

Figure 3-16: Kalman Gain K

➤ **Change Signal Frequency From $1/(2\pi)$ to $1/\pi$**

Figure 3-17: Estimated Position Trajectory

Figure 3-18: Kalman Gain K

— **Discussion**

For these two situations, we can easily acknowledge that Kalman gain does not change with the frequency.

3.2.3.3 More complicated model

Consider a particle moving in the plane at constant velocity subject to random perturbations in its trajectory. The new position (x_1, x_2) is the old position plus the velocity (v_1, v_2) plus noise w . The new system equation is

$$\begin{bmatrix} x_1(t_k) \\ x_2(t_k) \\ v_1(t_k) \\ v_2(t_k) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t_{k-1}) \\ x_2(t_{k-1}) \\ v_1(t_{k-1}) \\ v_2(t_{k-1}) \end{bmatrix} + \begin{bmatrix} w_{x_1} \\ w_{x_2} \\ w_{v_1} \\ w_{v_2} \end{bmatrix} \quad (3-10)$$

We assume we only observe the position of the particle

$$\begin{bmatrix} y_1(t_k) \\ y_2(t_k) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(t_k) \\ x_2(t_k) \\ v_1(t_k) \\ v_2(t_k) \end{bmatrix} + \begin{bmatrix} v_{x_1} \\ v_{x_2} \end{bmatrix} \quad (3-11)$$

Figure 3-19: Estimated Position Trajectory

Figure 3-20: Kalman Gain K

— Discussion

The Kalman gain **K** in this case is 0.0437, which is smaller than 0.0613 in the simpler system model. This indicates that we have more accuracy in **K** because of the increment of the complexity of system model. The more parameters we consider, the more accuracy we can count for the Kalman gain **K**.

3.2.3.4 Process Noise Covariance $\mathbf{Q}$ and Measurement Noise Covariance $\mathbf{R}$

➤ Weighted $\mathbf{Q}$ and $\mathbf{R}$ [8][9]

Condition No. 1

$$\begin{aligned}\mathbf{Q}_k &= \mathbf{Q}\alpha^{-2(k+1)} \\ \mathbf{R}_k &= \mathbf{R}\alpha^{-2(k+1)}\end{aligned}\tag{3-12}$$

where $\alpha > 1$, $\mathbf{Q}$ and $\mathbf{R}$ are constant matrices. Since $\alpha > 1$, with the increasing of time instant k , $\mathbf{Q}$ and $\mathbf{R}$ are decreasing. This gives us more credibility to the recent data by the less influence of noise covariance, and the change is given exponentially with k . Also, from the change of $\mathbf{Q}$ and $\mathbf{R}$, we can observe the change of Kalman gain $\mathbf{K}$. If $\alpha = 1$, it's back to regular Kalman filter.

Let $\alpha = 1 + 10^{-3}$;

Figure 3-21: Estimated Position Trajectory

Figure 3-22: Process Noise Covariance $\mathbf{Q}$

Figure 3-23: Measurement Noise Covariance **R**

Figure 3-24: Kalman Gain **K**

— Notes

With the decreasing of **Q** and **R**, Kalman gain **K** becomes diverging and cannot reach a stable value. The estimated position trajectory (Figure 3-21) becomes more and more inaccurate too.

Condition No. 2

$$\mathbf{Q}_k = \frac{\mathbf{Q}}{\alpha^{-2(k+1)}} \quad (3-13)$$

$$\mathbf{R}_k = \frac{\mathbf{R}}{\alpha^{-2(k+1)}}$$

This is another situation we'd like to discuss. Similar as equation (3-12), where $\alpha > 1$, **Q** and **R** are constant matrices. Since $\alpha > 1$, with the increasing of time instant k , **Q** and **R** are increasing accordingly. This gives us less credibility to the recent data by the less influence of noise covariance, and the change is given exponentially with k .

Let $\alpha = 1+10^{(-3)}$

Figure 3-25: Estimated Position Trajectory

Figure 3-26: Process Noise Covariance Q

Figure 3-27: Measurement Noise Covariance R

Figure 3-28: Kalman Gain K

— Notes

With the increasing of Q and R , Kalman gain K goes to a peak value, then comes down quickly and gets stable at a very small number (0.00001773). The estimated position trajectory (Figure 3-25) shows better result compared with Figure 3-21.

➤ **Changing the Magnitude of Q and R**

1. Only increase **Q** by 10 times, leave **R** unaltered

Figure 3-29: Estimated Position Trajectory

Figure 3-30: Kalman Gain K

— **Notes**

K = 0.1064 (converging and getting stable at about 15 sec.)

2. Only decrease **Q** by 10 times, leave **R** unaltered

Figure 3-31: Estimated Position Trajectory

Figure 3-32: Kalman Gain K

— Notes

$K = 0.0349$ (converging but getting stable at 42 sec.)

3. Only increase R by 10 times, leave Q unaltered

Figure 3-33: Estimated Position Trajectory

Figure 3-34: Kalman Gain K

— Notes

$K = 0.0349$ (converging but getting stable at 42 sec., similar as Figure 3-32)

4. Only decrease R by 10 times, leave Q unaltered

Figure 3-35: Estimated Position Trajectory

Figure 3-36: Kalman Gain K

— Notes

$K = 0.1064$ (converging and getting stable at 15 sec., similar Figure 3-30)

5. Both increase Q and decrease R by 10 times together

Figure 3-37: Estimated Position Trajectory

Figure 3-38: Kalman Gain K

— Notes

$K = 0.1812$ (converging and getting stable at 8 sec.)

6. Both decrease Q and increase R by 10 times together

Figure 3-39: Estimated Position Trajectory

Figure 3-40: Kalman Gain K

— Notes

From Figure 3-40, Kalman gain **K** does not converge during the total time period, 100 sec. And it is showing the trend to diverge.

— Discussion

From the observation of Kalman gain **K** for weighted **Q** and **R**, and changing the magnitude of **Q** and **R**, we can see that changing **Q** and **R** does affect the transient duration and the steady state operation. Increasing **Q** would indicate either stronger noises driving the dynamics or increased uncertainty in the adequacy of the model itself to depict the true dynamics accurately. This will increase both the rate of growth of the **P_k** elements (or eigenvalues) between measurement times and their steady state values. As a result, the filter gains will generally increase, thereby weighting the measurements more heavily: this is reasonable since increased **Q** dictates that we should put less confidence in the output of the filter's own dynamics model. By similar reasoning, increased **R** would indicate that the measurements are subjected to a stronger corruptive noise, and so should be weighted less by the filter. In fact, this will decrease the gain values, the eigenvalues of the $[\mathbf{K}_k \mathbf{H}_k \mathbf{P}_k^-]$ term are smaller, so the error variances going from k to $k+1$ decrease to a lesser extent, and the steady state is quickly reached because the uncertainty involved in the state propagation is large compared to the accuracy of the measurements, so the new state estimate is heavily dependent upon the new measurement and not closely related to prior estimates.

3.2.3.5 Correlated Process Noise and Measurement Noise

So far, we have investigated the use of Kalman filter for uncorrelated process noise $\mathbf{w}_k$ and measurement noise $\mathbf{v}_k$. Now we want to consider an extension of these results, allowing correlation between the two noise processes (assumed jointly Gaussian).

Let this correlation be described by

$$E[\mathbf{w}_k \mathbf{v}_i^T] = \begin{cases} \mathbf{C}_k & i = k \\ 0 & i \neq k \end{cases} \quad (3-14)$$

A generalized derivation in [10] yields the optimal estimation algorithm with the same initial conditions and measurement update relations, i.e., leaving equations (2-8), (2-12), (2-15) unaltered, but with time propagation equations modified from equations (2-16), (2-18) to: [7]

$$\hat{\mathbf{x}}_{k+1}^- = \Phi_k \hat{\mathbf{x}}_k + \mathbf{C}_k (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \left(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^- \right) \quad (3-15)$$

$$\mathbf{P}_{k+1}^- = \Phi_k \mathbf{P}_k \Phi_k^T + \mathbf{Q}_k - \mathbf{C}_k (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \mathbf{C}_k^T - \Phi_k \mathbf{K}_k \mathbf{C}_k^T - \mathbf{C}_k \mathbf{K}_k^T \Phi_k^T \quad (3-16)$$

After applying equation (3-14), we have

$$\mathbf{C}_k = E[\mathbf{w}_k \mathbf{v}_i^T] = \begin{bmatrix} \frac{1}{2} \Delta t^2 \\ \Delta t \end{bmatrix} * \text{process noise} * \text{measurement noise} \quad (3-17)$$

Thus, by using (3-15) and (3-16), the plot for Kalman gain K is as in Figure 3-41:

Figure 3-41: Kalman Gain K

And the plot of estimated position trajectory is shown in Figure 3-42,

Figure 3-42: Estimated Position Trajectory

This can be compared to the case of no correlation between w_k and v_k . The decrease in steady state values from $P_k^- = \begin{bmatrix} 6.5286 & 1.0321 \\ 1.0321 & 0.3213 \end{bmatrix}$ to $\begin{bmatrix} 0.0353 & 0.0172 \\ 0.0172 & 0.0084 \end{bmatrix}$ and $P_k = \begin{bmatrix} 6.1285 & 0.9689 \\ 0.9689 & 0.3113 \end{bmatrix}$ to $\begin{bmatrix} 0.0287 & 0.0158 \\ 0.0158 & 0.0087 \end{bmatrix}$ is due to the exploitation of the correlation between the dynamic noise and the noise that corrupts the observable outputs: the z_k realizations reveal more about the noise process w_k .

Next, we tune the correlation C_k to see the different Kalman gain K :

Figure 3-43: $C_k' = 0.90C_k$

Figure 3-44: $C_k' = 0.94C_k$

Figure 3-45: $C_k' = 0.96C_k$

Figure 3-46: $C_k' = 0.99C_k$

Figure 3-47: $C_k' = 1.1C_k$

Figure 3-48: $C_k' = 1.3C_k$

Figure 3-49: $C_k' = 1.7C_k$

Figure 3-50: $C_k' = 2.0C_k$

— Discussion

In the above figures, Kalman gain K cannot get stable any more because the influence of correlated process noise and measurement noise. With the increasing of the power of correlation (C_k), Kalman gain K becomes more and more unstable until totally random, and the estimated precision is getting poorer. Also, we can see that for C_k which are less than the original C_k (Figure 3-43 – 3-46), Kalman gain K is getting smaller and

smaller, but it cannot get stabilized, for C_k which are larger than the original C_k , the Kalman gain K shows complete randomness.

➤ Measurement Noise Correlated With Delayed Process Noise

Now, let us consider another condition for correlated process noise and measurement noise: a model involving correlation between the dynamic noise over a sample period and the noise corrupting the measurement at the end of that interval.

The reason that we are concerned with the crosscorrelation of v_k with w_{k-1} rather than w_k is if we rewrite equation (3-1) with k retarded one step, we have

$$\mathbf{x}_k = \Phi_{k-1} \mathbf{x}_{k-1} + \mathbf{w}_{k-1} \quad (3-18)$$

Note that it is w_{k-1} (and not w_k) that represents the cumulative effect of the white forcing function in the continuous model in the interval (t_{k-1}, t_k) . Similarly, v_k represents the cumulative effect of the white measurement noise in the continuous model when averaged over the same interval (t_{k-1}, t_k) . Therefore, if we wish to have a correspondence between the continuous and discrete models for small Δt , it is the correspondence between v_k and w_{k-1} that we need to include in the discrete model.

Based on the discussion above, we have

$$C_k = E[\mathbf{w}_{k-1} \mathbf{v}_k^T] \quad (3-19)$$

and accordingly, equation (2-12) and (2-13) are changed to

$$\mathbf{K}_k = [\mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{C}_k] [\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k + \mathbf{H}_k \mathbf{C}_k + \mathbf{C}_k^T \mathbf{H}_k^T]^{-1} \quad (3-20)$$

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k [\mathbf{H}_k \mathbf{P}_k^- + \mathbf{C}_k^T] \quad (3-21)$$

Figure 3-51 indicates the Kalman gain $\mathbf{K}$ under this case:

Figure 3-51: Kalman Gain $\mathbf{K}$

Figure 3-52: Estimated Position Trajectory

Tune $\mathbf{C}_k$, we can see different Kalman gain $\mathbf{K}$

Figure 3-53: $\mathbf{C}_k' = 0.90\mathbf{C}_k$

Figure 3-54: $\mathbf{C}_k' = 0.94\mathbf{C}_k$

Figure 3-55: $C_k' = 0.96C_k$

Figure 3-56: $C_k' = 0.99C_k$

Figure 3-57: $C_k' = 1.1C_k$

Figure 3-58: $C_k' = 1.3C_k$

Figure 3-59: $C_k' = 1.7C_k$

Figure 3-60: $C_k' = 2.0C_k$

— Discussion

These figures show how Kalman gain $\mathbf{K}$ is influenced, and it's similar to the previous $\mathbf{C}_k = E[\mathbf{w}_k \mathbf{v}_k^T]$, but there is no correlation between $\mathbf{v}_k$ and $\mathbf{w}_k$, so there is a greater spreading of errors over the next interval. By being correlated with $\mathbf{w}_{k-1}$, $\mathbf{v}_k$ is also correlated with $\mathbf{x}_k$, so the measurement $\mathbf{z}_k$ is more strongly correlated with $\mathbf{x}_k$, and thus the lower $\mathbf{P}_k$ value here. $(\begin{bmatrix} -0.7779 & -0.0010 \\ -0.0010 & -0.0399 \end{bmatrix} \times 10^{-6} \text{ versus } \begin{bmatrix} 0.0287 & 0.0158 \\ 0.0158 & 0.0087 \end{bmatrix})$

These examples dealing with $\mathbf{C}_k$ reveal the improved estimation precision due to exploiting the noise correlation: observing a particular realization of $\mathbf{v}_k$ as the corruption on the measurements yields probabilistic information about the realizations of $\mathbf{w}_k$ that have driven the system dynamics. As expected, either of these formulations reduces to the original Kalman filter algorithm if $\mathbf{C}_k \equiv 0$ for all time. Moreover, if $\mathbf{v}_k$ is correlated with both $\mathbf{w}_{k-1}$ and $\mathbf{w}_k$, as

$$E \left\{ \begin{bmatrix} \mathbf{w}_{k-1} \\ \mathbf{w}_k \\ \mathbf{v}_k \end{bmatrix} \begin{bmatrix} \mathbf{w}_{k-1}^T & \mathbf{w}_k^T & \mathbf{v}_k^T \end{bmatrix} \right\} = \begin{bmatrix} \mathbf{Q}_{k-1} & 0 & \mathbf{C}_{1k} \\ 0 & \mathbf{Q}_k & \mathbf{C}_{2k} \\ \mathbf{C}_{1k}^T & \mathbf{C}_{2k}^T & \mathbf{R}_k \end{bmatrix} \quad (3-22)$$

The above results can be combined. In practical applications, a tradeoff analysis would be conducted to determine whether the performance improvement afforded by this algorithm warrants the increased computer burden beyond that of the more conventional Kalman filter formulation.

Chapter 4

FUZZY LOGIC

AND

FUZZY CONTROL

4.1 Introduction to Fuzzy Logic

The concept of Fuzzy Logic (FL) was conceived by Professor Jan Lukasiewicz, a famous Polish philosopher and logician, father of multi-value logic. Later, Lotfi Zadeh, a professor at the University of California at Berkley, adapted and applied this methodology as a way of processing data by allowing partial set membership rather than crisp set membership or non-membership. This approach to set theory was not applied to control systems until the 70's due to insufficient small-computer capability prior to that time. Professor Zadeh reasoned that people do not require precise, numerical information

input, and yet they are capable of highly adaptive control. If feedback controllers could be programmed to accept noisy, imprecise input, they would be much more effective and perhaps easier to implement. Since its introduction, fuzzy logic has been applied in many areas, some of which include: systems analysis, signal processing, pattern recognition, decision analysis, diagnostics, and control.

Fuzzy logic is a problem-solving control system methodology that lends itself to implementation in systems ranging from simple, small, embedded micro-controllers to large, networked, multi-channel PC or workstation-based data acquisition and control systems. It can be implemented in hardware, software, or a combination of both. Fuzzy logic provides a simple way to arrive at a definite conclusion based upon vague, ambiguous, imprecise, noisy, or missing input information. Fuzzy logic's approach to control problems mimics how a person would make decisions, only much faster.

Fuzzy logic incorporates a simple, "IF – THEN" rule-based approach to a solving control problem rather than attempting to model a system mathematically. The Fuzzy logic model is empirically-based, relying on an operator's experience rather than their technical understanding of the system. For example, rather than dealing with temperature control in terms such as "process temperature = 500°C", or "210°C < temperature < 220°C", terms like "IF (process is too cool) AND (process is getting colder), THEN (add heat to the process)" or "IF (process is too hot) AND (process is heating rapidly), THEN (cool the process quickly)" are used. These terms are imprecise and yet very descriptive of what must actually happen. Fuzzy logic requires some numerical parameters in order

to operate such as what is considered significant error and significant rate-of-change-of-error, but exact values of these numbers are usually not critical unless very responsive performance is required in which case empirical tuning would determine them.

4.1.1 Fuzzy Logic and Control Engineering

Fuzzy logic was first introduced to the area of control by Mamdani in 1975, with the intent of duplicating the behavior of human system operators. Imprecise descriptions of a control task that are obtained from operators can be modeled with fuzzy logic, allowing automated control without the need for a formal analysis of the system.

On the most abstract level, all controllers are simply mapping functions — they map their inputs and their state to their outputs. Standard controllers do this computationally; fuzzy logic controllers use a rule-base.

The majority of the literature on modern control methods concerns linear control. Nonlinear problems are usually dealt with by approximating them with a linear model. Those solutions that do address nonlinear problems lack generality; they are only applicable to a specific type of problem.

Fuzzy logic may be viewed as a language for describing arbitrary control surfaces. As such, it is not restricted, as classical controllers are, to the description of

linear control surfaces, and it is not restricted by the complexities of nonlinear math, as some possible solutions to the control of nonlinear systems are.

4.2 Fuzzy Sets

Fuzzy sets and fuzzy logic are used to heuristically quantify the meaning of the linguistic variables, linguistic values, and linguistic rules that are specified by the expert.

4.2.1 Introduction to Fuzzy Sets

Following Zadeh many sets have more than an either-or criterion for membership. Take for example the set of young people. A one year old baby will clearly be a member of the set, and a 100 years old person will not be a member of this set, but what about people at the age of 20, 30, or 40 years? Another example is a weather report regarding high temperatures, strong winds, or nice days. In other cases a criterion appears nonfuzzy, but is perceived as fuzzy: a speed limit of 60 kilometers per hour, a check-out time at 12 noon in a hotel, and so on. Zadeh proposed a grade of membership, such that the transition from membership to non-membership is gradual rather than abrupt.

4.2.2 Universe

Elements of a fuzzy set are taken from a universe of discourse, or universe for short. The universe contains all elements that can come into consideration.

4.2.3 Membership Function

A function that maps a set of elements to their corresponding grades of truth in a fuzzy set is called a membership function. Every element in the universe of discourse is a member of the fuzzy set to some grade, maybe even zero. The membership of an element $x, x \in X$, in the fuzzy set A is denoted:

$$\mu_A(x)$$

and X is called the universe of discourse of A .

The membership function describes the “certainty” that an element of X , with a linguistic description, may be classified linguistically. It is a graphical representation of the magnitude of participation of each input. It associates a weighting with each of the inputs that are processed, define functional overlap between inputs, and ultimately determines an output response. The rules use the input membership values as weighting factors to determine their influence on the fuzzy output sets of the final output

conclusion. Once the functions are inferred, scaled, and combined, they are defuzzified into a crisp output which drives the system.

The membership function quantifies, in a continuous manner, whether values belong to or are members of the set of values that are ruled by certain “rule-bases”, and hence it quantifies the meaning of the corresponding linguistic statement.

Clearly, many choices for the shape of the membership functions are possible (e.g., triangular and trapezoidal shapes), and these will each provide a different meaning for the linguistic values that they quantify. Figure 4-1 below gives a graphical illustration of a variety of membership functions. [5]

Figure 4-1: Some Typical Membership Functions

Here we show the mathematical characterization for the triangular membership function as an example. In Table 4-1, c^L specifies the “saturation point” and w^L specifies

the slope of nonunity and nonzero part of μ^L . Similarly, for μ^R . For μ^C , c is the center of the triangle and w is the base-width.

	Triangular Membership Functions
Left	$\mu^L(u) = \begin{cases} 1 & \text{if } u < c^L \\ \max \left\{ 0, 1 + \frac{c^L - u}{0.5w^L} \right\} & \text{otherwise} \end{cases}$
Center	$\mu^C(u) = \begin{cases} \max \left\{ 0, 1 + \frac{u - c}{0.5w} \right\} & \text{if } u < c \\ \max \left\{ 0, 1 + \frac{c - u}{0.5w} \right\} & \text{otherwise} \end{cases}$
Right	$\mu^R(u) = \begin{cases} \max \left\{ 0, 1 + \frac{u - c^R}{0.5w^R} \right\} & \text{if } u < c^R \\ 1 & \text{otherwise} \end{cases}$

Table 4-1: Mathematical Characterization of Triangular Membership Functions

4.2.4 Singletons

Strictly speaking, a fuzzy set **A** is a collection of ordered pairs:

$$\mathbf{A} = \{(x, \mu(x))\}$$

Item x belongs to the universe and $\mu(x)$ is its grade of membership in A . A single pair $\{(x, \mu(x))\}$ is called a fuzzy singleton, thus the whole set can be viewed as the union of its constituent singleton. It is often convenient to think of a set A just as a vector

$$\mathbf{a} = (\mu(x_1), \mu(x_2), \dots, \mu(x_n))$$

It is understood then, that each position i ($1, 2, \dots, n$) corresponds to a point in the universe of n points.

4.2.5 Operations on Fuzzy Sets

4.2.5.1 Fuzzy Union and Intersection

Fuzzy sets can be combined by union and intersection in a similar manner to classical sets. Zadeh defined the union of two fuzzy sets by the maximum of their membership functions:

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)) \quad (4-1)$$

And the intersection of two fuzzy sets by the minimum of their membership functions:

$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)) \quad (4-2)$$

Figure 4-2 gives an example of these operations in use:

Figure 4-2: Fuzzy Union and Intersection

(a) Two fuzzy sets **A** and **B**,

(b) their union $\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$, and

(c) their intersection $\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$.

These are not the only definitions of union and intersection, but they are the most widely used in practice.

4.2.5.2 Fuzzy Complement

The complement (“not”) of a fuzzy set **A** with a membership function $\mu_A(x)$ has a membership function given by $1 - \mu_A(x)$.

4.3 Fuzzy Control

Fuzzy control is a control method based on fuzzy logic. Just as fuzzy logic can be described simply as “computing with words rather than numbers”; fuzzy control can be described simply by “control with sentences rather than equations”. A fuzzy controller can include empirical rules, and that is especially useful in operator control plants.

4.3.1 Advantages of Fuzzy Control

When compared to classical control theory, a fuzzy logic approach to control offers the following advantages: [11][12][13]

1. It can be used in systems which cannot be easily modeled mathematically. In particular, systems with nonlinear responses that are difficult to analyze may respond to a fuzzy control approach.
2. As a rule-based approach to control, fuzzy control can be used to efficiently represent an expert's knowledge about a problem.

3. Continuous variables may be represented by linguistic constructs that are easier to understand, making the controller easier to implement and modify. For instance, instead of using numeric values, temperature may be represented as "cold, cool, nominal, warm, or hot".
4. Fuzzy controllers may be less susceptible to system noise and parameter changes, thus making them more robust.
5. Complex processes can often be controlled by relatively few logic rules, allowing a more understandable controller design and faster computation for real-time applications.

Fuzzy controllers are the most important applications of fuzzy theory. They work rather different than conventional controllers; expert knowledge is used instead of differential equations to describe a system. This knowledge can be expressed in a very natural way using linguistic variables, which are described by fuzzy sets.

4.3.2 Elements of Fuzzy Controller

The fuzzy controller is composed of the following four elements: [5]

1. A *rule-base* (a set of IF-THEN rules), which contains a fuzzy logic quantification of the expert's linguistic description of how to achieve good control.

2. An *inference mechanism* (also called an “inference engine” or “fuzzy inference” module), which emulates the expert’s making in interpreting and applying knowledge about how best to control the plant.
3. A *fuzzification interface*, which converts controller inputs into information that the inference mechanism can easily use to activate and apply rules.
4. A *defuzzification interface*, which converts the conclusions of the inference mechanism into actual inputs for the process.

Figure 4-3: Fuzzy Controller Architecture

4.3.3 Rule-Base

The fuzzy rule-base is the core part of a fuzzy logic system in the sense that all other three components – fuzzy inference mechanism, fuzzification, and defuzzification – are used to interpret these rules and make them usable for specific cases.

Specifically, the fuzzy rule-base comprises the following fuzzy IF-THEN rules:

IF x_1 is A'_1 and ... and x_n is A'_n , THEN y is B'

where A'_i and B' are fuzzy sets in $U_i \subset R$ and $V \subset R$, respectively, and $\mathbf{x} = (x_1, x_2, \dots, x_n)^T \in U$ and $y \in V$ are the input and output (linguistic) variables of the fuzzy system, respectively.

To model a rule of the form “IF A and B THEN C” (where A, B, and C are fuzzy sets) with fuzzy logic, two definitions need to be made: how the connective, “and/or,” is modeled in fuzzy logic; and how implication, “IF...THEN...” is modeled.

4.3.4 Fuzzy Connectives

The connective “and” is usually modeled with the fuzzy intersection (the min operator), and the connective “or” is usually modeled with fuzzy union (the max operator). Thus, if two propositions on the same universe are linked by a connective, the

result is a fuzzy set on that universe, and if two fuzzy propositions on differing universes are linked by a connective the result is a fuzzy relation on the product space of the two universes.

4.3.5 Fuzzy Implication

There are several models of fuzzy implication. In general, the fuzzy relation mapping the antecedent of a rule to the consequent of a rule is annotated:

$$\mathbf{R} = \mathbf{I}(\mathbf{A}, \mathbf{C}) \quad (4-3)$$

where $\mathbf{R}$ is the fuzzy relation that defines the rule, $\mathbf{A}$ is the fuzzy relation that defines the antecedent of the rule, and $\mathbf{C}$ is the fuzzy relation that defines the consequent of the rule.

The most popular definitions of $\mathbf{I}(a, c)$ are min inference, where

$$\mathbf{I}(a, c) = \min(a, c) \quad (4-4)$$

and product inference, where

$$\mathbf{I}(a, c) = ac \quad (4-5)$$

4.3.6 Inference Mechanism

The inference mechanism has two basic tasks: [5]

- (1) determining the extent to which each rule is relevant to the current situation as characterized by the inputs u_i , $i = 1, 2, \dots, n$ (we call the task “matching”); and
- (2) drawing conclusions using the current inputs u_i and the information in the rule-base (we call this task an “inference step”).

4.3.7 Inference Step

The inference process involves two steps:

1. The premises of all the rules are compared to the controller inputs to determine which rules apply to the current situation. The “matching” process involves determining the certainty that each rule applies, and typically we will more strongly take into account the recommendations of rules that we are more certain apply to the current situation.
2. The conclusion (what actions to take) are determined using the rules that have been determined to apply at the current time. The conclusions are characterized with a fuzzy set (or sets) that represent the certainty that the input to the plant should take on various values.

4.3.8 Fuzzification

The fuzzification process is the act of obtaining a value of an input variable and finding the numeric values of the membership function(s) that are defined for that variable.

Fuzzy sets are used to quantify the information in the rule-base, and the inference mechanism operates on fuzzy sets to produce fuzzy sets; hence, we must specify how the fuzzy systems will convert its numeric inputs into fuzzy sets, so that they can be used by the fuzzy system. [5] As an example, the term “tall” can be represented in Figure 4-4:

Figure 4-4: Fuzzy Set Representation

In Figure 4-4, it shows the degree of membership with which a person belongs to the category (set) “tall”. Full membership of the class “tall” is represented by a value of 1, while no membership is represented by a value of 0. At 150cm and below, a person does not belong to the class “tall”. At 210cm and above, a person fully belongs to the

class “tall”. Between 150cm and 210cm the membership increases linearly between 0 and 1.

1. The degree of belonging to the set “tall” is called the confidence factor or the membership value. The shape of the membership function curve can be non-linear.

Normally fuzzy concepts have a number of values to describe the various ranges of values of the objective term which they describe. For example, the fuzzy concept “tallness” may have the values “tall”, “medium” and “short”. The membership functions of these values are as shown in Figure 4-5:

Figure 4-5: Membership Functions

Typically, fuzzy concepts have an odd number of values: 3, 5 or 7. For example, we can extend the above values by adding very short and very tall. The real power of fuzzy logic systems, compared to crisp logic systems, lies in the ability to represent a concept using a small number of fuzzy values. This therefore reduces the number of rules required to capture the knowledge relating to that concept. To achieve the same accuracy

with crisp logic, a large number of logical values would be required resulting in a large rule-base.

4.3.9 Defuzzification

Defuzzification operates on the implied fuzzy sets produced by the inference mechanism and combines their effects to provide the “most certain” controller output.

Generally, there are two kinds of defuzzification methods.

1. **Center of Gravity (COG) method**

Kosko (1986) has suggested a modification to classical fuzzy logic which simplifies the process of finding the center of the output set. He notes that aggregation by summation (rather than the classical method of union) preserves the area of the local output membership functions. Thus, the center of an output set produced by additive aggregation is the same as that found by combining the centers of the local output sets.

Therefore, in the situation where additive aggregation and defuzzification by the center of gravity method is used, aggregation (and its accompanying calculations) may be omitted.

It's for combining the recommendations represented by the implied fuzzy sets from all the rules. (Figure 4-6 (a))

$$\mu^{crisp} = \frac{\sum_i b_i \int \mu_{(i)}}{\sum_i \int \mu_{(i)}} \quad (4-6)$$

where

b_i — center of the membership function of the consequent of rule (i)

$\int \mu_{(i)}$ — area under the membership function $\mu_{(i)}$

2. "Center Average" method (Figure 4-6 (b))

$$\mu^{crisp} = \frac{\sum_i b_i \mu_{premise_{(i)}}}{\sum_i \mu_{premise_{(i)}}} \quad (4-7)$$

Because equation is a weighted average of the center values of the output membership function centers, we call it "center-average" method.

Also, many other defuzzification schemes have been suggested with the aim of providing a better interface between the fuzzy output of a rule-base and the discrete output required by controllers (Mizmoto 1989). In practice, however, center of gravity defuzzification is used most often because it is simple and fast.

Figure 4-6: Defuzzification Methods

4.4 Summary

An introduction to the use of fuzzy logic in control has been presented, and the various aspects of fuzzy logic that effect the efficiency of fuzzy inference in fuzzy controllers discussed. It has been shown that an efficient controller can be built using the following methods for inference:

- 1) The min operator for conjunction in rules.
- 2) The product operator for implication.
- 3) Fuzzy singleton modeling of input sets.
- 4) Additive local aggregation.
- 5) Center of gravity defuzzification method.
- 6) Lumped mass modeled (singleton) output sets.

Chapter 5

DESIGN OF FUZZY SYSTEM

Fuzzy systems are very simple conceptually. They consist of an input stage, a processing stage, and an output stage. The input stage maps sensor or other inputs, such as switches, thumb wheels, and so on, to the appropriate membership functions and truth values. The processing stage invokes each appropriate rule and generates a result for each, then combines the results of the rules. Finally, the output stage converts the combined result back into a specific control output value.

5.1 Design Methodology [5]

1. Understand the behavior of the plant, how it reacts to inputs, what are the effects of the disturbance, and what fundamental limitation it presents (e.g., nonminimum phase or unstable behavior). A clear understanding comes from studying the physics of the process, developing mathematical models, using system identification methods, doing analysis, performing simulation, and using heuristic knowledge about the plant dynamics.
2. Gain a clear understanding of the closed-loop specifications (i.e., the performance objectives). These may be stated in terms of specifications on stability, rise-time, over-shoot, settling time, steady-state error, disturbance rejection, robustness, and so on.
3. Establish the basic structure of the control system ("assumed as direct (non-adaptive) controller is used"). This will establish what the plant and controller inputs and outputs should be.
4. Perform an initial control design.

The basic approaches include:

- (a) inclusion of good control knowledge,
 - (b) tuning the scaling gains,
 - (c) tuning the membership functions, and
 - (d) adding more rules or membership functions.
5. If simple initial approach to control is successful, begin working on an implementation. If it is not successful, first make sure that solid control

engineering ideas are used to pick the “nonfuzzy” part of the controller (e.g., the preprocessing of fuzzy controller inputs by choosing to use an integrator to try to remove steady-state error). If this does not work, consider the following options:

- A more sophisticated conventional control method (e.g., feedback-linearization or sliding-mode control).
 - A more sophisticated fuzzy controller. More inputs of the fuzzy controller or more rules may be needed in the rule-base. Often, the problem with tuning a fuzzy controller boils down to a basic lack of understanding of how best to control the plant and the corresponding lack of knowledge in the rule-base.
 - Try designing the fuzzy controller by using a well-designed linear control technique to specify the general shape of the control surface (especially around zero) and then tune the surface starting from there.
 - Conventional or fuzzy adaptive or supervisory control approaches.
6. Repeat the above process as often as necessary, evaluating the designs in simulation and, if possible, implementation.

The following section contains a general introduction, from the point of working with a well-conditioned rule-base.

5.2 Choice of Control Variables

First, pick the input and output variables for the controller. Some initial assumptions regarding controllability/observability may have to be made if a formal physical model has not been constructed.

5.3 Input Set Design

Cover the range of each input with overlapping triangular sets. The sets should form a fuzzy partition. The number of sets is determined by how nonlinear the control surface needs to be — two sets is sufficient for a linear control surface, five or seven sets are usually used as a starting point. An odd number of sets is usually chosen, with one set positioned at the set-point of the controller, and an equal number of sets positioned either side of it. Sets are usually concentrated around the set point to allow more precise control in that area.

5.4 Rule Design

Next, write the rules to determine the action of the controller. This procedure is usually done systematically: for every combination of input sets that may reasonably be expected to occur in practice, the necessary output value is estimated by the expert, and a rule is written expressing the relation.

This requires that the designer has sufficient knowledge of the plant to be able to predict its reaction to a given control signal (the output) in a given situation (the state of the plant for the given input region). This does not imply that a complete and accurate mathematical model of the plant is needed. However, it does imply that some form of model must exist, even if it is only a linguistic one (that is, a description based on the designers' empirical experience of the plant). Knowledge may be incorporated from many sources: from empirical models, from locally approximated mathematical models, from the observations of plant operators, and from trial and error methods.

5.5 Output Set Weighting

At this point every rule has its own unique output set. The weighting associated with the set can therefore be viewed as being associated with the rule, or with the characteristic point that the rule defines. Weightings are assigned to output sets to achieve the desired transition between characteristic points, the nature of the transition being determined by the ratio of the weights associated with adjacent characteristic points. A ratio of 1 results in a linear transition. A ratio of 3 produces a strong bias towards the more heavily weighted output. During the initial design phase it is usual to select identical weights for all output sets. This results in a linearly interpolated control surface. Set weightings are then adjusted during the tuning phase of design.

5.6 Performance Tuning

After the controller is installed (in either a real or simulated situation) performance measurements can be made. Tuning is an iterative process. The performance of the controller is measured on a region by region basis, the regions being defined by the input coverage of individual rules. The output of the rules is then adjusted with the aim of improving performance at characteristic points. The performance of the controller in regions that are covered by more than one rule must be adjusted by changing the weights associated with the output sets of the rules. Higher weights will bias the output towards the associated output.

If the desired performance cannot be obtained by adjusting the output sets, the number of input sets in the region of inadequate performance should be increased. This may be achieved by concentrating existing input sets, or by adding more input sets. Design then proceeds from the beginning, with new rules being written for the newly introduced combinations of sets.

If satisfactory control still cannot be achieved, further investigation of the physical characteristics of the plant — with particular reference to any initial assumptions of controllability/observability — will be necessary.

5.7 Rule-Base Tuning

Next, steps may be taken to simplify the rule base to make it easier to read and faster executing. This is generally referred to as *rule reduction*.

There are two basic methods of rule reduction.

1. Where output behavior does not vary between adjacent sets, the sets may be collected together under one set.

For example, for two adjacent sets a_1 and a_2 , the rules involving them and other set combinations may all result in the same output:

IF x is a_1 and y is b_j , THEN z is c_j

IF x is a_2 and y is b_j , THEN z is d_j

If $c_j = d_j$ for all j , then a_1 and a_2 may be replaced by their union. The membership function of the new set should be constructed to conform with the criteria that define a well conditioned rule base. It will be trapezoidal if a_1 and a_2 are triangular.

2. Where output remains constant for all rules that include one particular set, the rules may be represented by one rule.

For example, for some rules involving combinations of the set a_1 and the sets b_j :

IF x is a_1 and y is b_j , THEN z is k_j

If k_j does not vary with j (that is, it has the same center and weighting for all j), then the rules may be represented with the single rule:

IF x is a_1 , THEN z is k

Finally, a check should be made for redundant rules (rules that contain combinations of inputs that do not occur in practice), and any found should be removed.

5.8 Example of Fuzzy Controller Design [14][15]

In this section, a steam turbine model is used to demonstrate the design of a fuzzy controller. The block diagram of this control system appears as follows:

Figure 5-1: Control System Block Diagram

5.8.1 Fuzzy Control in Detail

We have the input temperature variable maps into the following fuzzy set:

Figure 5-2: Membership Functions for Temperature Input

The pressure input variable is mapped as follows:

Figure 5-3: Membership Functions for Pressure Input

The output variable, the throttle setting, is mapped as follows:

Figure 5-4: Membership Functions for Throttle Setting Output

where:

- N3 — Large negative.
- N2 — Medium negative.
- N1 — Small negative.
- Z — Zero.
- P1 — Small positive.
- P2 — Medium positive.
- P3 — Large positive.

The rule set includes such rules as:

Rule 1: IF temperature is cool AND pressure is weak, THEN throttle is P3.

Rule 2: IF temperature is cool AND pressure is low, THEN throttle is P2.

Rule 3: IF temperature is cool AND pressure is ok, THEN throttle is Z.

Rule 4: IF temperature is cool AND pressure is strong, THEN throttle is N2.

In practice, the controller accepts the inputs and maps them into their membership functions and truth values. These mappings are then fed into the rules. If the rule specifies an AND relationship between the mappings of the two input variables, as the examples above do, the minimum of the two is used as the combined truth value; if an OR is specified, the maximum is used. The appropriate output state is selected and assigned a membership value at the truth level of the premise. The truth values are then defuzzified.

For an example, assume the temperature is in the “cool” state, and the pressure is in the “low” and “ok” states. The pressure values ensure that only rules 2 and 3 fire:

Rule 2 evaluation:

Figure 5-5: Rule 2 Evaluation

Rule 3 evaluation:

Figure 5-6: Rule 3 Evaluation

The two outputs are then defuzzified through COG defuzzification method:

Figure 5-7: COG Defuzzification

The output value will adjust the throttle and then the control cycle will begin again to generate the next value.

5.8.2 Building Fuzzy Controller

Consider implementing with a feedback controller:

Figure 5-8: Fuzzy Controller

A fuzzy set is defined for the input error variable “e”, and the derived change in error, “delta”, as well as the “output”, as follows:

LP — Large Positive
 SP — Small Positive
 ZE — Zero
 SN — Small Negative
 LN — Large Negative

If the error ranges from -1 to $+1$, with the analog-to-digital converter used having a resolution of 0.25 , then the input variable’s fuzzy set (which, in this case, also applies to the output variable) can be described very simply as a table, with the error/delta/output values in the top row and the truth values for each membership function arranged in rows:

	-1	-0.75	-0.5	-0.25	0	0.25	0.5	0.75	1
mu(LP)	0	0	0	0	0	0	0.3	0.7	1
mu(SP)	0	0	0	0	0.3	0.7	1	0.7	0.3
mu(ZE)	0	0	0.3	0.7	1	0.7	0.3	0	0
mu(SN)	0.3	0.7	1	0.7	0.3	0	0	0	0
mu(LN)	1	0.7	0.3	0	0	0	0	0	0

Table 5-1: Rule Table for Input and Output

or, in graphical form (where each "●" has a value of 0.1):

	LN	SN	ZE	SP	LP
-1.0	oooooooo	ooo	...	...	...
-0.75	ooooooo	oooooooo	...	...	...
-0.5	ooo	oooooooooooo	ooo	...	...
-0.25	...	ooooooo	ooooooo	...	...
0	...	ooo	oooooooooooo	ooo	...
0.25	...	...	ooooooo	ooooooo	...
0.5	...	...	ooo	oooooooooooo	ooo
0.75	...	...	...	ooooooo	ooooooo
1.0	...	...	...	ooo	oooooooooooo

Table 5-2: Graphic Form of Rule Table

Suppose this fuzzy system has the following rule base:

Rule 1: IF e = ZE AND delta = ZE, THEN output = ZE

Rule 2: IF e = ZE AND delta = SP, THEN output = SN

Rule 3: IF e = SN AND delta = SN, THEN output = LP

Rule 4: IF e = LP OR delta = LP, THEN output = LN

These rules are typical for control applications in that the antecedents consist of the logical combination of the error and error-delta signals, while the consequent is a control command output.

The rule outputs can be defuzzified using a discrete COG computation:

$$\text{SUM}(I = 1 \text{ TO } 4 \text{ OF } (\mu(I) * \text{output}(I)))/\text{SUM}(I = 1 \text{ TO } 4 \text{ OF } \mu(I))$$

Now, suppose that at a given time we have:

$$e = 0.25$$

$$\text{delta} = 0.5$$

Then this gives:

	e	delta
mu(LP)	0	0.3
mu(SP)	0.7	1
mu(ZE)	0.7	0.3
mu(SN)	0	0
mu(LN)	0	0

Table 5-3: Rule Table for Error (e) and Change of Error (delta)

Plugging this into rule 1 gives:

rule 1: IF e = ZE AND delta = ZE, THEN output = ZE, $\mu(1) = \text{MIN}(0.7, 0.3) = 0.3$; output(1) = 0

where

- $\mu(1)$:

Truth value of the result membership function for rule 1. In terms of a center of gravity calculation, this is the "mass" of this result for this discrete case.

- $\text{Output}(1)$:

Value (for rule 1) where the result membership function (ZE) is maximum over the output variable fuzzy set range. That is, in terms of a center of gravity calculation, the location of the "center of mass" for this individual result.

This value is independent of the value of " μ ". It simply identifies the location of ZE along the output range.

The other rules give:

rule 2: IF $e = \text{ZE}$ AND $\text{delta} = \text{SP}$, THEN $\text{output} = \text{SN}$, $\mu(2) = \text{MIN}(0.7, 1) = 0.7$; $\text{output}(2) = -0.5$

rule 3: IF $e = \text{SN}$ AND $\text{delta} = \text{SN}$, THEN $\text{output} = \text{LP}$, $\mu(3) = \text{MIN}(0, 0) = 0$; $\text{output}(3) = 0.75$

rule 4: IF $e = \text{LP}$ OR $\text{delta} = \text{LP}$, THEN $\text{output} = \text{LN}$, $\mu(4) = \text{MAX}(0, 0.3) = 0.3$; $\text{output}(4) = -1$

The center of gravity calculation yields the crisp output value:

$$\begin{aligned} & \frac{\mu(1) \times \text{output}(1) + \mu(2) \times \text{output}(2) + \mu(3) \times \text{output}(3) + \mu(4) \times \text{output}(4)}{\mu(1) + \mu(2) + \mu(3) + \mu(4)} \\ &= \frac{(0.3 \times 0) + (0.7 \times (-0.5)) + (0 \times 0.75) + (0.3 \times (-1))}{0.3 + 0.7 + 0 + 0.3} \\ &= \frac{0 - 0.35 + 0 - 0.3}{1.3} \\ &= \frac{-0.65}{1.3} \\ &= -0.5 \end{aligned}$$

5.9 Summary

The following is a checklist of design choices that have to be made for designing a fuzzy controller:

- *Rule-base related choices:* Number of inputs and outputs, rules, universes, continuous/discrete, the number of membership functions, their overlap and width, singleton output.
- *Inference mechanism related choices:* Connectives, modifiers, activation operation, aggregation operation, and accumulation operation.
- *Defuzzification method:* COG, center-average, etc.
- *Pre- and post-processing:* Scaling, gain factors, and sampling time.

Chapter 6

FUZZY FILTER

AND APPLICATION

6.1 Introduction to Fuzzy Filter

As described in previous chapters, we have got some knowledge about Kalman filter, fuzzy logic and fuzzy systems. In control system and filter design, the establishment of a system model is quite fundamental. Many traditional approaches assume that models are available in the form of ordinary differential equations or discrete-time difference equations. In the cases where there exists uncertainty the models are generally assumed to be in the form of stochastic differential or difference equations. In many cases it is very difficult to obtain stochastic models, especially if the process is highly complex. Even when models are available it is sometimes difficult to propagate

the uncertainty through the system dynamics according to the axioms of probability theory, especially for non-linear systems. So now we introduce an “optimal” state estimator, based on fuzzy set theory, which is capable of dealing with systems with random disturbances and “uncertainty”. We will refer to this as a *fuzzy filter*, and it is a fuzzy dynamic system model of the process in the estimator [16]. This filter is similar to that of the Kalman filter when a linear system with Gaussian noises is considered.

The fuzzy dynamic system model is used to propagate uncertainty in a way analogous to that of the mean and covariance estimator of a Kalman filter. Here, the membership function is viewed as being analogous to a probability density function. In this way, the fuzzy dynamic system model lends itself directly for use in a state estimator for non-linear, non-Gaussian systems. However, this is done with significant increase in computational complexity as the fuzzy dynamic system model is propagating a function and not just two parameters of a function like mean and covariance.

6.2 Fuzzy Filter Construction [8][16]

6.2.1 Basic Framework

The system’s dynamics are expressed by a discrete-time stochastic equation of the form

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \quad (6-1)$$

$$\mathbf{z}_k = g(\mathbf{x}_k, \mathbf{v}_k)$$

where

- $\mathbf{x}_k$ — system state;
- $\mathbf{u}_k$ — system input;
- $\mathbf{w}_k$ — white noise process disturbance;
- $\mathbf{z}_k$ — measurement output;
- $\mathbf{v}_k$ — white noise measurement disturbance;

We shall refer to this model as the process “truth model” as it should be the best known representation of the real system behavior. Notice in the above system model that the system input is assumed to be measured perfectly. We do this without loss of generality since the “noises” or errors in measuring the process input $\mathbf{u}_k$ can be absorbed into $\mathbf{w}_k$. Also, it is not necessary that measurements are available at every time step. In many applications the measurements may become unavailable for periods of time or they may occur at a varying rate over time.

The fuzzy filter consists of three basic components, namely a fuzzy dynamic system model, a fuzzy measurement model, and a measurement update block; hence, conceptually its structure bears some similarity to the Kalman filter.

Figure 6-1: Block Diagram of Fuzzy Filter

6.2.2 Fuzzy Dynamic System Model

The fuzzy dynamic system model is designed so that it accurately characterizes the system behavior and its uncertainty. As illustrated in Figure 6-1, it measures the system input and predicts the true system output and uncertainty in the form of a fuzzy relational state variable. Notice that unlike what has been presented in the current literature, the fuzzy dynamic system modeled here is propagating the full fuzzy relation

describing the state and not a fuzzy set for each independent state variable. The reason for doing this is due to the fact that the fuzzy relation describing the state vector has a membership function that can be considered analogous to a joint probability density function. By propagating the full fuzzy relation we capture and maintain information similar to the crosscorrelation. Ignoring it is like having all the off-diagonal elements of the Kalman filter covariance matrix forced to zero.

Next we explain how to produce the fuzzy dynamic system model in Figure 6-1 from the stochastic system model of (6-1). First, we will review fuzzy composition used to generate the fuzzy dynamic system model.

6.2.2.1 Cartesian Product

If $U_1, U_2, \dots, U_r$ are fuzzy sets defined as subsets of the universes of the discourse $\mathcal{U}_1 \times \mathcal{U}_2 \times \dots \times \mathcal{U}_r$, respectively, the Cartesian product is a fuzzy set, denoted $U_1 \times U_2 \times \dots \times U_r$, with a membership function defined by

$$\mu_{U_1 \times U_2 \times \dots \times U_r}(u_1, u_2, \dots, u_r) = \min\{\mu_{U_1}(u_1), \mu_{U_2}(u_2), \dots, \mu_{U_r}(u_r)\} \quad (6-2a)$$

where $u_1, u_2, \dots, u_r$ are elements of $\mathcal{U}_1 \times \mathcal{U}_2 \times \dots \times \mathcal{U}_r$ respectively.

The Cartesian product of two or more fuzzy sets is called a *fuzzy relation*.

6.2.2.2 Fuzzy Composition

Assume that U and R are fuzzy relations defined on $\mathcal{U}_1 \times \mathcal{U}_2 \times \dots \times \mathcal{U}_r$ and $\mathcal{U}_1 \times \mathcal{U}_2 \times \dots \times \mathcal{U}_r \times \mathcal{Y}$, respectively, the “fuzzy composition” of U and R is a fuzzy relation denoted by $U \circ R$ and has a membership function defined as

$$\begin{aligned} \mu_{U \circ R}(y) = & \circ \left\{ \min \left\{ \mu_U(u_1, u_2, \dots, u_r), \mu_R(u_1, u_2, \dots, u_r, y) : \right. \right. \\ & (u_1, u_2, \dots, u_r) \in \mathcal{U}_1 \times \mathcal{U}_2 \times \dots \times \mathcal{U}_r \text{ and } y \in \mathcal{Y} \left. \right\} : \\ & (u_1, u_2, \dots, u_r) \in \mathcal{U}_1 \times \mathcal{U}_2 \times \dots \times \mathcal{U}_r \left. \right\} \end{aligned} \quad (6-2b)$$

6.2.2.3 Construction of Fuzzy Dynamic Model via Fuzzy Composition

To use fuzzy composition it is convenient to generate a list of fuzzy rules that characterizes the behavior of the stochastic dynamic behavior in (6-1). A rule base is generated so that it contains enough rules to ensure completeness (i.e., there exist enough rules that at least one is activated for all realizations of the vectors $\mathbf{x}_k$, $\mathbf{u}_k$, and $\mathbf{w}_k$). Suppose the i^{th} rule is defined so that it is activated in the region of space around the set of values $\mathbf{x}_k = \mathbf{x}^i \in \mathbb{R}^n$, $\mathbf{u}_k = \mathbf{u}^i \in \mathbb{R}^r$, and $\mathbf{w}_k = \mathbf{w}^i \in \mathbb{R}^p$. Here we define the following fuzzy relations:

$$\mathbf{X}_k^i = \mathcal{F}(\mathbf{x}^i) \quad (6-3)$$

$$\mathbf{U}^i = \mathcal{F}(u^i) \quad (6-4)$$

$$\mathbf{W}^i = \mathcal{F}(w^i) \quad (6-5)$$

$$\mathbf{X}_{k+1}^i = \mathcal{F}(f(x^i, u^i, w^i)) \quad (6-6)$$

$$\mathbf{U}(k) = \mathcal{F}(u_k) \quad (6-7)$$

where $\mathcal{F}$ represents the process of fuzzification around a single element.

Figure 6-2: Fuzzy Dynamic System Model

In Figure 6-2, we can see the procedures that how fuzzy dynamic system model works. Using the fuzzy relations defined in equations (6-3) to (6-7), we can generate a fuzzy knowledge base such that the i^{th} rule is given by

If $X(k)$ is X_k^i and $U(k)$ is U^i and $W(k)$ is W^i Then $X(k+1)$ is X_{k+1}^i

where $X(k)$ and $X(k+1)$ are fuzzy relations characterizing the current and the next state, respectively, $W(k)$ is a fuzzy relation characterizing the process distribution input, and $U(k)$ is a fuzzy relation characterizing the process input.

The rule can be denoted mathematically by a fuzzy relation of the form

$$\mathbf{R}^i = \mathbf{X}_k^i \times \mathbf{U}^i \times \mathbf{W}^i \times \mathbf{X}_{k+1}^i \quad (6-8)$$

Assuming there exist many such rules, the overall fuzzy relation describing the entire rule base is denoted by the union of these rules

$$\mathbf{R} = \bigcup_i \mathbf{R}^i \quad (6-9)$$

Thus the fuzzy dynamic system model can be expressed by the following

$$\mathbf{X}(k+1) = [\mathbf{X}(k) \times \mathbf{U}(k) \times \mathbf{W}(k)] \circ \mathbf{R} \quad (6-10)$$

The choice of the fuzzy relation $\mathbf{W}(k)$ is based on a fuzzy characterization of the process disturbance at time k . One possible choice for the membership function of $\mathbf{W}(k)$ is simply a normalized version of the joint probability density function of $\mathbf{w}_k$. In many cases, the characteristics of the process disturbance input will not change in time so often the fuzzy relation $\mathbf{W}(k)$ will be constant.

6.2.3 Fuzzy Measurement Model

The fuzzy measurement model (Figure 6-3) is designed so that it properly characterizes the measuring device and its white noise disturbances $\mathbf{v}_k$. It receives the fuzzy state relation estimate $\tilde{\mathbf{X}}(k)$ from the fuzzy dynamic system model to generate a prediction of the fuzzy relation $\tilde{\mathbf{X}}(k) \times \mathbf{Z}(k)$. Similar to the fuzzy dynamic system model, the measurement model is designed using the fuzzy extension principle or fuzzy composition. For example if $\mu_{\tilde{\mathbf{X}}(k)}(x_k)$ and $\mu_{\mathbf{V}(k)}(v_k)$ are the membership functions for fuzzy sets $\tilde{\mathbf{X}}(k)$ and $\mathbf{V}(k)$, respectively, their Cartesian product can be formed to obtain $\mu_{\tilde{\mathbf{X}}(k) \times \mathbf{V}(k)}(x_k, v_k)$. Using the fuzzy extension principle we get

$$\mu_{\tilde{\mathbf{X}}(k) \times \mathbf{V}(k)}(\mathbf{x}_k, \mathbf{v}_k) = \begin{cases} \max \left\{ \mu_{\tilde{\mathbf{X}}(k) \times \mathbf{V}(k)}(\mathbf{x}_k, \mathbf{v}_k) : \mathbf{z}_k = g(\mathbf{x}_k, \mathbf{v}_k) \right\} \\ 0 \end{cases} \quad \begin{array}{l} \text{If there does not exist } \mathbf{v}_k \\ \text{such that } \mathbf{z}_k = g(\mathbf{x}_k, \mathbf{v}_k) \\ \text{is satisfied} \end{array} \quad (6-11)$$

Figure 6-3: Fuzzy Measurement Model

6.2.4 Measurement Update Block

The measurement update block performs the function of improving the fuzzy relational state estimate $\tilde{X}(k)$ using information provided by system measurements. Here we derive the fuzzy measurement update equations in a “fuzzy Bayesian” manner by generating a new fuzzy relation for the state conditioned on system measurements.

6.2.4.1 Fuzzy Bayesian Rule

A fuzzy set, denoted A , condition on fuzzy set $B = b$ (a fuzzy singleton), denoted $A|B=b$ where b is a specific element on the universe of discourse for B , has a membership function often defined by

$$\mu_{A|B=b}(a) = \frac{\mu_{A \times B}(a,b)}{\mu_B(b)} \quad (6-12)$$

where $\mu_{A \times B}(a,b)$ is the membership function for $A \times B$, and $\mu_B(b)$ is the membership function for B . This is called “fuzzy Bayesian rule”.

Since the state equation in (6-1) is Markov, the algorithm will be recursive. If the process is Markovian, measurement updates need only be based on the current measurement since by definition previous measurements will not provide additional information.

Figure 6-4: Measurement Update Block

The fuzzy relation, $\tilde{\mathbf{X}}^+(k)$, describing the fuzzy state after the measurement update is given by

$$\tilde{\mathbf{X}}^+(k) = \tilde{\mathbf{X}}(k) \Big|_{\mathbf{Z}(k) = \mathbf{z}_k} \quad (6-13)$$

which has a membership function

$$\mu_{\tilde{\mathbf{X}}^+(k)}(x_k) = \mu_{\tilde{\mathbf{X}}(k)} \Big|_{\mathbf{Z}(k) = \mathbf{z}_k}(x_k, z_k) = N \left(\mu_{\tilde{\mathbf{X}}(k) \times \mathbf{Z}(k)}(x_k, z_k) \right) \quad (6-14)$$

where N is a normalization operation that normalizes the functions so that peak has a value of 1.

6.2.5 Defuzzification and Optimality

So far, we have described a method for generating a fuzzy estimate of the system state using a fuzzy model of the process and measurement feedback based on conditional fuzzy sets. However, in many applications the fuzzy relation estimate $\tilde{\mathbf{X}}(k)$ must be defuzzified to a specific vector $\hat{\mathbf{x}}(k)$ to be practical use. Hence, the optimality of the $\hat{\mathbf{x}}(k)$ depends greatly on the chosen defuzzification method.

In Kalman filter, the conditional mean is chosen as the optimal estimate because it is the mean, mode, and median of the joint Gaussian probability density function. When the membership function of the state relation estimate $\tilde{\mathbf{X}}(k)$ is considered analogous to a joint probability density function, computation of the mean is akin to performing a center of gravity defuzzification [48] [49]. So, the defuzzification algorithm will be either the center of gravity or mean of maximum.

6.3 Fuzzy Filter Application

Here we still use a linear stochastic model introduced in Chapter 2, which is of the form same as (2-1) and (2-2) with the same parameters. To simplify the math, we set the process input $\mathbf{u}_k$ to zero.

Let the inputs be $\mathbf{e}_k, \dot{\mathbf{e}}_k, \mathbf{e}_{p_i}, \dot{\mathbf{e}}_{p_i}$, and the output is the estimated position at step $k+1$, where $\mathbf{e}_k$ is the error between measured position value and true position value, $\dot{\mathbf{e}}_k$ is the change in $\mathbf{e}_k$, $\mathbf{e}_{p_i}$ is the difference between $\mathbf{P}_k$ and $\mathbf{P}_k^-$, and $\dot{\mathbf{e}}_{p_i}$ is the change in $\mathbf{e}_{p_i}$. Here we use a MISO (Multiple Inputs Single Output) system to accomplish the task. And the results are plotted in Figure 6-5 to Figure 6-7. From Figure 6-5, it shows the accuracy of the estimated position values is higher than what we got from Kalman filter (Figure 3-4 or Figure 6-5a). The mean value of estimated positions is -0.0620 (-1.4735 in Kalman filter), the variance is 0.2145 (6.1585 in Kalman filter).

Figure 6-5: Estimated Position Trajectory for Fuzzy Filter

Figure 6-5a: Estimated Position Trajectory for Kalman Filter

Figure 6-6: e_k and $\dot{e}_k$ of Fuzzy Filter

Figure 6-7: e_p and $\dot{e}_p$ of Fuzzy Filter

The rules being used are such as:

Rule 1: IF e_k is negative large, $\dot{e}_k$ is zero, e_{p_i} is negative large, and $\dot{e}_{p_i}$ is negative large, THEN estimated position at $k+1$ is positive large.

Rule 2: IF e_k is zero, $\dot{e}_k$ is zero, e_{p_i} is negative large, and $\dot{e}_{p_i}$ is negative large, THEN estimated position at $k+1$ is positive large.

Rule 3: IF e_k is positive large, $\dot{e}_k$ is zero, e_{p_i} is zero, and $\dot{e}_{p_i}$ is zero, THEN estimated position at $k+1$ is negative large.

Rule 4: IF e_k is zero, $\dot{e}_k$ is positive small, e_{p_i} is negative small, and $\dot{e}_{p_i}$ is negative small, THEN estimated position at $k+1$ is negative small.

-
-
-

Furthermore, we can see from the above figures and theory introduction, the simulation of fuzzy filter is done in a recursive method like Kalman filter. While for the linear Gaussian case we are able to show very close relationships between the fuzzy filter and Kalman filter, in the nonlinear Gaussian case the two methods produce quite different results. For the nonlinear Gaussian case one normally resorts to using the extended Kalman filter which relies on the use of the system model and an on-line linearization of this model. The optimality and convergence properties of the Kalman filter are lost for the nonlinear case which opens the possibility that other estimators may perform better.

The fuzzy filter does not need to perform an on-line linearization of the system model since its approach does not build on a linear estimation result like the extended Kalman filter.

In Figure 6-8 and 6-8a, we use the correlated process and measurement noises to test the filtering efficiency of Kalman filter and fuzzy filter. More figures about comparison between fuzzy filter and Kalman filter can be found in Chapter 7.

**Figure 6-8: Estimated Position Trajectory of Fuzzy Filter
with Correlated Process and Measurement Noises**

**Figure 6-8a: Estimated Position Trajectory of Kalman Filter
with Correlated Process and Measurement Noises**

**Figure 6-9: e_k and $\dot{e}_k$ of Fuzzy Filter
with Correlated Process and Measurement Noises**

**Figure 6-10: e_{p_i} and $\dot{e}_{p_i}$ of Fuzzy Filter
with Correlated Process and Measurement Noises**

Chapter 7

COMPARISON

BETWEEN

KALMAN FILTER AND FUZZY FILTER

7.1 Kalman Filter

The most widely used method for sensor fusion in mobile robot applications is the Kalman filter [8][19]. This filter is often used to combine all measurement data (e.g., for fusing data from different sensors) to get an optimal estimate in a statistical sense. If the system can be described with a linear model and both the system error and the sensor error can be modeled as white Gaussian noise, then the Kalman filter will provide a unique statistically optimal estimate for the fused data. This means that under certain

conditions the Kalman filter is able to find the best estimates based on the “correctness” of each individual measurement.

The calculation of the Kalman filter is done recursively, i.e., in each iteration, only the newest measurement and the last estimate will be used in the current calculation, so there is no need to store all the previous measurements and estimates. The measurements from a group of n sensors can be fused using a Kalman filter to provide an estimate of the current state of a system and a prediction of the future state of the system.

The inputs to a Kalman filter are the system measurements. The *a priori* information required are the system dynamics and the noise properties of the system and the sensors. The output of the Kalman filter is the estimated system state and the *innovation* (i.e., the difference between the predicted and observed measurement). The innovation is also a measure for the performance of the Kalman filter.

At each step, the Kalman filter generates a state estimate by computing a weighted average of the predicted state (obtained from the system model) and the innovation. The weight used in the weighted average is determined by the covariance matrix, which is a direct indication of the error in state estimation. In the simplest case, when all measurements have the same accuracy and the measurements are the states to be estimated, the estimate will reduce to a simple average, i.e., a weighted average with all weights equal. Note also that the Kalman filter can be used for systems with time-variant parameters.

The *extended Kalman filter* is used to place of the conventional Kalman filter if the system model is potentially numerically unstable or if the system model is not approximately linear. The extended Kalman filter is a version of the Kalman filter that can handle non-linear dynamics or non-linear measurement equations, or both.

In [8] and [9], an interesting concept of an extended Kalman filter with gain adaptation based on fuzzy logic has been shown.

7.2 Fuzzy Filter

Construction of fuzzy filter involves three elements:

- (1) fuzzy dynamic system model;
- (2) fuzzy measurement model; and
- (3) measurement update block;

And accordingly, there are three steps to accomplish the construction:

- (1) define fuzzy sets in the input space;
- (2) construct a set of IF-THEN rules by incorporating input histogram statistics to form the fuzzy membership functions, and
- (3) construct the filter based on the set of rules.

7.3 Summary

From the simulation in previous chapters, we have presented that for linear systems and triangular shaped membership functions, the fuzzy filter works similarly as Kalman filter but produces better results than the Kalman filter. In Figure 7-1 and 7-2, we show the general comparison of fuzzy filter and Kalman filter. In [47], it is shown that the convergence of the Kalman filter's estimate of the true value is guaranteed only when the system is stochastically controlled and observed. But if the conditions can not be satisfied, the convergence can not be got, and the divergence will occur.

Zoom in part, please see Figure 7-2

Figure 7-1: Comparison of Kalman Filter and Fuzzy Filter

Figure 7-2: Comparison of Kalman Filter and Fuzzy Filter (Zoom In Part)

Use the similar parameter tuning as in Kalman filter, we can compare the two filters more thoroughly.

- Both increase Q and decrease R by 10 times together

Figure 7-3: Kalman Filter

Figure 7-4: Fuzzy Filter

- Increase the process noise level and measurement noise level by 10 times

Figure 7-5: Kalman Filter

Figure 7-6: Fuzzy Filter

➤ **Weighted Q and R**

$$\mathbf{Q}_k = \mathbf{Q}\alpha^{-2(k+1)}$$

$$\mathbf{R}_k = \mathbf{R}\alpha^{-2(k+1)}$$

with $\alpha = 1+10^{-3}$;

Figure 7-7: Kalman Filter

Figure 7-8: Fuzzy Filter

Figure 7-3 to 7-8 show the estimated position trajectory for Kalman filter and fuzzy filter under different conditions, respectively. We can draw the conclusion from these figures that the fuzzy filter is outperforming the Kalman filter in position estimation for these cases.

Chapter 8

CONCLUSION

A linear system can be described by linear differential equations, and white noise means that the noise value is not correlated in time. Kalman filter is very powerful when the system is linear and contains only white noises. Kalman gain $\mathbf{K}$ converges and the result of filtering is obvious. But, when the system is non-linear, the noises become non-white, and the parameters such as covariance $\mathbf{Q}$ and $\mathbf{R}$ are being tuned, Kalman filter shows unstable, Kalman gain $\mathbf{K}$ diverges, and the filtering effect is becoming unreliable.

In this thesis we have presented a fuzzy filter that was developed using fuzzy logic technique. We have shown that for linear systems with white process and measurement noise, the fuzzy filter produces the similar result as the Kalman filter. The example we used is a dynamic system that consists of a spacecraft which is accelerating with random bursts of gas from its reaction control system thrusters. It is possible to find

other classes of systems whose solution is difficult to obtain with regular Kalman filter methods but easily obtained using the fuzzy filter approach. For example, for systems with colored process and measurement noise the Kalman filter does not perform properly and its gain diverges. In that case such filter has to be replaced by other filters, for example, fuzzy filter. In this work a procedure of fuzzy filter design was presented. The design of fuzzy logic inference engine was performed on the basis of general experience and results from multiple simulations. The rules could be also extracted on purely experimental basis. In this case, by tuning parameters, weighted Q and R , and introducing correlated process and measurement noises, we investigated and compared simulation results. From the experimental results, plots and data, we found that in many cases, especially linear systems with white noise, the fuzzy filter could be considered as an equivalent to the Kalman filter and for colored noise it could be considered as extension of such filter. The fuzzy filter could also be used for non-linear systems.

Also, it has been shown that convergence of the Kalman filter's estimate to the true value is guaranteed, as could be expected, only when the system is controllable and observable.

REFERENCES

- [1] Edited by: Moss, Frank; McClintock, P. V. E., "Noise in Nonlinear Dynamical Systems", Vol. 2, "Theory of Noise Induced Processes In Special Applications", Cambridge University Press, 1989
- [2] Kassam, Saleem A., "Signal Detection in Non-Gaussian Noise", Dowden & Culver, Inc. 1988
- [3] Wio, H. S., Colet, P., Miguel, M. San, Pesquera, L., and Rodriguez, M. A., "Stochastic Processes Driven By Colored Noise: A Path Integral Point of View", pp. 157-169, Edited by: Tirapegui, E. and Zeller, W., "Instabilities and Nonequilibrium Structures III", Kluwer Academic Publishers, 1991
- [4] Cunningham, Edward P., "Digital Filtering, An Introduction", Houghton Mifflin Company, 1992
- [5] Passino, Kevin M., and Yurkovich, Stephen, "Fuzzy Control", Addison Wesley Longman, Inc., 1998
- [6] Brown, Robert Grover, and Hwang, Ptarick Y. C., "Introduction to Random Signals and Applied Kalman Filtering", 2nd edition, John Wiley & Sons, Inc., 1992
- [7] Maybeck, Peter S., "Stochastic Model, Estimation and Control", Vol. 1, Academic Press, Inc. 1979
- [8] Sasiadek, J.Z., Wang, Q., and Zaremba, M.B., "INS/GPS Navigation Data Fusion Using Fuzzy Adaptive Kalman Filtering", pp. 543-547, Proc. of IFAC, Symposium on Robot Control (SYROCO '00), Sept. 21-23, 2000 Vienna, Austria

- [9] Sasiadek, J, and Wang, Q., (1999), "Sensor Fusion Based on Fuzzy Kalman Filtering for Autonomous Robot Vehicle", Proc. IEEE Int. Conf. On Robotics and Automation
- [10] Kalman, R. E., "New Methods in Wiener Filtering Theory", Proc. Symp. Eng. Applications of Random Function Theory and Probability, 1st. Wiley, New York, 1963
- [11] Mamdani, E. H., Ostergaard, J. J., Lembessis, E. In "Advances in fuzzy sets, possibility theory, and applications", P. P. Wang, Ed., Plenum Press, 1983
- [12] Klement, E. P., "Fuzzy logic in artificial intelligence" – 8th Austrian artificial intelligence conference FLAI '93. Proceedings, Slany, W., Ed., Springer-Verlag, 1993, Vol. 695
- [13] Mamdani, E. "Twenty years of fuzzy control: experiences gained and lessons learnt." Proceedings of the Second IEEE International Conference on Fuzzy Systems, 1993, pp.339-344
- [14] Self, Kevin, "Designing With Fuzzy Logic", IEEE SPECTRUM, November 1990, pp. 42-44, 105
- [15] Cox, Earl, "Fuzzy Fundamentals", IEEE SPECTRUM, October 1992, pp. 58-61
- [16] Layne J.R., Passino K.M., "A Fuzzy Dynamic Mode Based State Estimator", Fuzzy Sets and Systems, To appear in 2001
- [17] Lee, C., "Fuzzy Logic In Control Systems: Fuzzy Logic Controller – Part I", IEEE, Trans. On Systems, Man. And Cybernetics, Vol. 20, pp. 404-418, Mar/Apr, 1990

- [18] Lee, C., "Fuzzy Logic In Control Systems: Fuzzy Logic Controller – Part II", IEEE, Trans. On Systems, Man. And Cybernetics, Vol. 20, pp. 419-435, Mar/Apr, 1990
- [19] Borenstein, Johann; Everett, H. R.; Feng, Liqiang, "Navigating Mobile Robots", A. K. Peters, Ltd., 1996
- [20] Hovanessian, S.A. "Introduction to Sensor Systems", Artech House, Inc., 1988
- [21] Alfred, Leick, "GPS satellite surveying", New York, Wiley, 1990
- [22] Hager, Gregory D., "Task-directed sensor fusion and planning – A Computational Approach", Boston, Mass., Kluwer Academic Publishers, 1990
- [23] Hartana, Pande, "Comparison of Linearized and Extended Kalman Filter in GPS Aided Inertial Navigation System", M. Eng, Thesis, Carleton University, Ottawa, 2000
- [24] Edited by: Rondeau, S., "Space Robotics (SPRO'98)" – A Proceeding volume from the IFAC Workshop St-Hubert, Quebec, Canada, 19-22 October 1998, Pergamon, 1999
- [25] Madsen, Christi K.; Zhao, Jian H., "Optical Filter Design and Analysis", John Wiley & Sons, Inc, 1999
- [26] Parks, T. W.; Burrus, C. S., "Digital Filter Design", John Wiley & Sons, Inc., 1987
- [27] Edited by: R.R. Yager, "Fuzzy Sets and Applications: Selected Papers by L.A. Zadeh", John Wiley, New York, 1987
- [28] Kallianpur, Gopinath, "Stochastic Filtering Theory", Springer-Verlag New York Inc., 1980

- [29] Edited by: Germani, Alfredo, "Stochastic Modeling and Filtering", Proceedings of the IFIP-WG 7/1 Working Conference, Rome, Italy, December 10-14, 1984, M. Thoma and A. Wyner, "Lecture Notes in Control and Information Sciences", Springer-Verlag Berlin, Heidelberg, 1987
- [30] Childers, Donald, and Durling, Allen, "Digital Filtering and Signal Processing", West Publishing Company, 1975
- [31] Edited by: Hamza, M. H., "Applied Control Filtering and Signal Processing" – A Publication of The International Association of Science and Technology for Development – IASTED, ACTA Press, 1987
- [32] A. Leon-Garcia, "Probability and Random Processes for Electrical Engineering", 2nd ed., Addison-Wesley, 1994
- [33] Lynn, Paul A., "Digital Signals, Processors and Noise", The MacMillan Press Ltd. 1992
- [34] Jacobs, O. L. R. "Introduction to control theory", Clarendon Press, Oxford, 1974
- [35] Wang, Qi, "Control and Navigation System for Autonomous Vehicles and Robots Using Fuzzy Logic and Kalman Filtering", M. Eng. Thesis, Carleton University, Ottawa, 1999
- [36] Miguel, M. San, Colet, P., Wio, H. S., Pesquera, L., and Rodriguez, M. A., "Path Integral Approach to The Colored Noise Problem", pp. 172-186, Fox, Ronald F., "Testing Approximate Theories of Colored Noise", pp. 207-227, Cambridge University Press, 1990
- [37] Wang, Li-Xin, "A Course in Fuzzy Systems and Control", Prentice-Hall, Inc., 1997

- [38] Edited by: Moss, Frank; McClintock, P. V. E., "Noise in Nonlinear Dynamical Systems", Vol. 1, "Theory of Continuous Fokker-Planck Systems", Cambridge University Press, 1989
- [39] Kosko, Bart, "Neural Networks And Fuzzy Systems", A Dynamical Systems Approach to Machine Intelligence, Prentice-Hall, Inc., 1992
- [40] Edited by Abraham Kandel, Gideon Langholz, With foreword by Lofti A. Zadeh, "Fuzzy Control Systems", CRC Press, Inc., 1994
- [41] Edited by: Moss, Frank; McClintock, P. V. E., "Noise in Nonlinear Dynamical Systems", Vol. 3, "Experiments and Simulations", Cambridge University Press, 1989
- [42] Sasiadek, Jerzy Z, and Wojcik, Piotr J., "Design of A Discrete Kalman Filter for A Tactile Sensor in A Feedback Control System", Vol., 45, No. 4, Int. J. Control, pp. 1211-1228, 1987
- [43] Haykin, Simon, "Adaptive Filter Theory", 2nd edition, Prentice-Hall, Inc., New Jersey, 1991
- [44] Gibson, Jerry D., Koo, Boneung, and Gray, Steven D. "Filtering of Colored Noise for Speech Enhancement and Coding", IEEE Transactions on Signal Processing, Vol. 39, No. 8, pp. 1732-41, August 1991.
- [45] Lewis, Frank L., "Optimal Estimation, With an Introduction to Stochastic Control Theory", John Wiley & Sons, 1986
- [46] Babuska, Robert, "Fuzzy Modeling for Control", Kluwer Academic Publishers, 1998

- [47] Maybeck, P., "Stochastic Models, Estimation, and Control", Vol. 1-3, San Diego, California, Academic Press, 1979
- [48] Jazwinski, Andrew H., "Stochastic Processes and Filtering Theory", New York, Academic Press, 1970
- [49] Edited by: Dubois, Didier; Prade, Henri; and Yager, Ronald R., "Fuzzy Information Engineering – A Guided Tour of Applications", John Wiley & Sons, Inc., 1997
- [50] Yager, Ronald R., and Filev Dimitar P., "Essentials of Fuzzy Modeling and Control", John Wiley & Sons, Inc., 1994
- [51] Ott, Henry W. "Noise Reduction Techniques in Electronic Systems" second edition, John Wiley & Sons, Inc., 1988

APPENDICES

APPENDIX A

In this appendix, the simulation programs written in Matlab for Kalman filter are presented.

A.1 Kalman Filter

```
% Simulation Program For Regular Kalman Filter

% alpha = forgetting factor (alpha >= 1)
% duration = length of simulation (seconds)
% dt = step size (seconds)

clear all

measnoise = 10; % position measurement noise (m)
accelnoise = 0.5; % acceleration noise (m/sec^2)
dt = 0.2;
duration = 100;
alpha = 1;

a = [1 dt; 0 1]; % transition matrix
c = [1 0]; % measurement matrix
x = [0; 0]; % initial state vector
xhat = x; % initial state estimate

Q = accelnoise^2 * [dt^4/4 dt^3/2; dt^3/2 dt^2];
% process noise covariance
Pest = Q; % initial estimation covariance
R = measnoise^2; % measurement error covariance

% set up the size of the innovations vector
Inn = zeros(size(R));

pos = []; % true position array
poshat = []; % estimated position array
posmeas = []; % measured position array
Processnoise = []; % process noise array
Measnoise = []; % measurement noise array
```



```

gaink = []; % Kalman filter gain
Innavation = [];

for t = 0 : dt: duration,

    % Simulate the process
    ProcessNoise = accelnoise * [(dt^2/2)*randn; dt*randn];
    x = [10*sin(t); 0];
    xest = x;
    x = a * x + ProcessNoise;

    % Simulate the measurement
    MeasNoise = measnoise * (randn);
    z = c * x + MeasNoise;

    % Innovation
    Inn = (z - c * xhat);

    % Covariance of Innovation
    s = c * Pest * c' + R;

    % Gain matrix
    K = Pest * c' * inv(s);

    % State estimate
    xhat = xest + K * Inn;

    % Covariance of prediction error
    P = Pest - K * c * Pest;
    Pest = a * P * a' + Q;
    xest = a * xhat;

    % Save some parameters in vectors for plotting later
    pos = [pos; x(1)];
    posmeas = [posmeas; z];
    poshat = [poshat; xest(1)];
    Processnoise = [Processnoise; ProcessNoise(1)];
    Measnoise = [Measnoise; MeasNoise(1)];
    gaink = [gaink; K(1)];
    Innavation = [Innavation; Inn(1)];
end

% Plot the results
t = 0 : dt : duration;
diff = pos - poshat;

figure(1)
plot(t,pos,'r',t,poshat,'k',t,posmeas,'b')
grid on
title('Kalman Filter Performance')
ylabel('Positions (m)');
xlabel('Time (sec)');

figure(2)
plot(t,pos)
grid on

```

```

ylabel('True Position Trajectory (m)')
xlabel('Time (sec)');

figure(3)
plot(t,posmeas)
grid on
ylabel('Measured Position Trajectory (m)');
xlabel('Time (sec)');
figure(4)
plot(t,poshat);
grid on
ylabel('Estimated Position Trajectory (m)');
xlabel('Time (sec)');

figure(5)
plot(t,Innavation)
grid on
ylabel('Innovation (m)');
xlabel('Time (sec)');

figure(6)
plot(t, Measnoise)
grid on
ylabel('Measurement Noise')
xlabel('Time (sec)');

figure(7)
plot(t, gaink)
grid on
ylabel('Kalman Filter Gain K');
xlabel('Time (sec)');

```

A.2 Kalman Filter with Complex Dynamic Model

```

% Kalman Filter with Complex Dynamic Model

% alpha = forgetting factor (alpha >= 1)
% duration = length of simulation (seconds)
% dt = step size (seconds)

clear all

measnoise = 10; % position measurement noise (m)
accelnoise = 0.5; % acceleration noise (m/sec^2)
dt = 0.2;
duration = 100;
alpha = 1;
ss = 4; % state size
os = 2; % observation size

a = [1 0 1 0; 0 1 0 1; 0 0 1 0; 0 0 0 1]; % transition matrix

```

```

c = [1 0 0 0; 0 1 0 0];
x = [0; 0; 0; 0];
xhat = x; % initial state estimate

Q = 0.0001*eye(ss); % process noise covariance
Pest = Q; % initial estimation covariance
R = 100*eye(os);

% set up the size of the innovations vector
Inn = zeros(size(R));

pos = []; % true position array
poshat = []; % estimated position array
posmeas = []; % measured position array
Processnoise = []; % process noise array
Measnoise = []; % measurement noise array
gaink = []; % Kalman filter gain
Innovation = [];

for t = 0 : dt: duration,

    % Simulate the process
    ProcessNoise = accelnoise * [(dt^2/2)*randn; (dt^2/2)*randn;
    dt*randn; dt*randn];
    x = [10*sin(t); 0; 0; 0];
    xest = x;
    x = a * x + ProcessNoise;

    % Simulate the measurement
    MeasNoise = measnoise * [randn; randn];
    z = c * x + MeasNoise;

    % Innovation
    Inn = z - c * xhat;

    % Covariance of Innovation
    s = c * Pest * c' + R;

    % Gain matrix
    K = Pest * c' * inv(s);

    % State estimate
    xhat = xest + K * Inn;

    % Covariance of prediction error
    P = Pest - Pest * c' * inv(s) * c * Pest;
    Pest = a * P * a' + Q;
    xest = a * xhat;

    % Save some parameters in vectors for plotting later
    pos = [pos; x(1)];
    posmeas = [posmeas; z(1)];
    poshat = [poshat; xest(1)];
    Processnoise = [Processnoise; ProcessNoise(1)];
    Measnoise = [Measnoise; MeasNoise(1)];
    gaink = [gaink; K(1)];

```

```

        Innavigation = [Innavation; Inn(1)];
    end

    % Plot the results
    t = 0 : dt : duration;
    diff = pos - poshat;

    figure(1)
    plot(t, pos, 'r', t, poshat, 'k', t, posmeas, 'b')
    grid on
    title('Kalman Filter Performance')
    ylabel('Positions (feet)');
    xlabel('Time (sec)');

    figure(2)
    plot(t, pos)
    grid on
    ylabel('True Position (feet)')
    xlabel('Time (sec)');

    figure(3)
    % plot(t, posmeas)
    plot(t, Innavigation)
    grid on
    % ylabel('Measured Position (feet)');
    ylabel('Innavation');
    xlabel('Time (sec)');

    figure(4)
    plot(t, poshat);
    grid on
    ylabel('Estimated Position Trajectory (m)');
    xlabel('Time (sec)');

    figure(5)
    plot(t, Processnoise)
    grid on
    ylabel('Process Noise');
    xlabel('Time (sec)');

    figure(6)
    plot(t, Measnoise)
    grid on
    ylabel('Measurement Noise');
    xlabel('Time (sec)');

    figure(7)
    plot(t, gaink)
    grid on
    ylabel('Kalman Filter Gain K');
    xlabel('Time (sec)');

```

A.3 Kalman Filter with Weighted Q and R, and Correlated Process

Noise and Measurement Noise

```
% Simulation Program For Kalman Filter with Weighted Q and R, and
% Correlated Process Noise and Measurement Noise

% alpha = forgetting factor (alpha >= 1)
% duration = length of simulation (seconds)
% dt = step size (seconds)

clear all

measnoise = 10; % position measurement noise (feet)
accelnoise = 0.5; % acceleration noise (feet/sec^2)
dt = 0.2;
duration = 100;
alpha = 1;
% alpha = 1 + 10^(-3);

a = [1 dt; 0 1]; % transition matrix
c = [1 0]; % measurement matrix
x = [0; 0]; % initial state vector
xhat = x; % initial state estimate

Q = accelnoise^2 * [dt^4/4 dt^3/2; dt^3/2 dt^2];
% process noise covariance
Pest = Q; % initial estimation covariance
R = measnoise^2; % measurement error covariance
C = accelnoise * measnoise * [dt^2/2; dt];
% covariance between process noise and measurement noise

% set up the size of the innovations vector
Inn = zeros(size(R));

pos = []; % true position array
poshat = []; % estimated position array
posmeas = []; % measured position array
Processnoise = []; % process noise array
Measnoise = []; % measurement noise array
gaink = []; % Kalman filter gain
Innovation = [];

% k = 0;
for t = 0 : dt: duration,

    % Simulate the process
    ProcessNoise = accelnoise * [(dt^2/2)*randn; dt*randn];
    x = [10*sin(t); 0];
    xest = x;
    x = a * x + ProcessNoise;

    % Simulate the measurement
```

```

MeasNoise = measnoise * (randn);
z = c * x + MeasNoise;

% Innovation
Inn = z - c * xhat;

% Covariance of Innovation
% s = c * Pest * c' + R;

% Gain matrix
% K = (a * Pest * c' + C) * inv(s);
K = (Pest * c' + C) * inv(c * Pest * c' + R + c * C + C' *
c');

% State estimate
xhat = a * xest + K * Inn;

% Covariance of prediction error
% P = a * P * a' + Q - a * P * c' * inv(s) * c * P * a';
P = Pest - K * (c * Pest + C');
% P = Pest - K * c * Pest;
Pest = a * P * a' + Q;
% Pest = a * P * a' + Q - C * inv(s) * C' - a * K * C' - C *
% K' * a';
xest = a * xhat;
% xest = a * xhat + C * inv(s) * Inn;
% k = k + 1;
% Q = Q / (alpha ^ (-2 * (k)));
% R = R / (alpha ^ (-2 * (k)));

% Save some parameters in vectors for plotting later
pos = [pos; x(1)];
posmeas = [posmeas; z];
poshat = [poshat; xhat(1)];
Processnoise = [Processnoise; ProcessNoise(1)];
Measnoise = [Measnoise; MeasNoise(1)];
gaink = [gaink; K(1)];
Innavation = [Innavation; Inn(1)];
end

% Plot the results
t = 0 : dt : duration;
diff = pos - poshat;

figure(1)
plot(t,pos,'r',t,poshat,'k',t,posmeas,'b')
grid on
title('Kalman Filter Performance')
ylabel('Positions (m)');
xlabel('Time (sec)');

figure(2)
plot(t,pos)
grid on
ylabel('True Position Trajectory (m)')
xlabel('Time (sec)');

```

```

figure(3)
plot(t, posmeas)
grid on
ylabel('Measured Position Trajectory (m)');
xlabel('Time (sec)');

figure(4)
plot(t, poshat);
grid on
ylabel('Estimated Position Trajectory (m)');
xlabel('Time (sec)');

figure(5)
plot(t, Innovation)
grid on
ylabel('Innovation (m)');
xlabel('Time (sec)');

figure(6)
plot(t, Measnoise)
grid on
ylabel('Error covariance');
xlabel('Time (sec)');

figure(7)
plot(t, gaink)
grid on
ylabel('Kalman Filter Gain K');
xlabel('Time (sec)');

```

APPENDIX B

In this appendix, we present the simulation programs for fuzzy filter in Matlab.

B-1 Fuzzy Filter for Two Inputs (signal error and change in the error)

```
% Fuzzy Filter for Two Inputs (signal error and change in the
% error)

clear all

t = 0;
index = 1; % Integration step size
step = 0.2;
duration = 100; % length of simulation (seconds)
counter = 10;
% This counter will be used to count the number of integration
% steps that have been taken in the current sampling interval.
% Set it to 10 to begin so that it will compute a fuzzy
% controller
% output at the first step.
eold = 0; % Initialize the past value of the error (for use
          % in computing the change of the error, c). Notice
          % that this is somewhat of an arbitrary choice since
          % there is no last time step. The same problem is
          % encountered in implementation.
measnoise = 10; % position measurement noise
accelnoise = 0.5; % acceleration noise
a = [1 step; 0 1]; % transition matrix
d = [1 0]; % measurement matrix
Q = accelnoise^2 * [step^4/4 step^3/2; step^3/2 step^2];
% process noise covariance
Pest = Q;
R = measnoise^2; % measurement error covariance

% Initialize parameters for the fuzzy filter
nume = 11; % Number of input membership functions for the e
          % universe of discourse
numc = 11; % Number of input membership functions for the c
          % universe of discourse

% Next, we define the scaling gains for tuning membership
% functions for
% universes of discourse for e, change in e (what we call c) and
```



```

% delta. These are g1, g2, and g0, respectively
% These can be tuned to try to improve the performance.

% g1 = 0.2; g2 = 0.014; g0 = 10;
g1 = 0.2; g2 = 0.01; g0 = 20;

% Next, define some parameters for the membership functions
we = 0.2 * (1/g1);
    % we is half the width of the triangular input membership
    % function bases
wc = 0.2 * (1/g2);
base = 0.4 * g0;

% Place centers of membership functions of the fuzzy controller:

% Centers of input membership functions for the e universe of
% discourse of fuzzy controller (a vector of centers)

ce = [-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1]*(1/g1);

% Centers of input membership functions for the c universe of
% discourse of fuzzy controller (a vector of centers)
cc = [-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1]*(1/g2);

% This next matrix specifies the rules of the fuzzy filter.
% The entries are the centers of the output membership functions.
% This choice represents just one guess on how to synthesize
% the fuzzy controller. Notice the regularity
% of the pattern of rules. Notice that it is scaled by g0, the
% output scaling factor, since it is a normalized rule base.
% The rule base can be tuned to try to improve performance.

rules=[1    1    1    1    1    1    0.8  0.6  0.3  0.1    0;
       1    1    1    1    1    0.8  0.6  0.3  0.1    0   -0.1;
       1    1    1    1    0.8  0.6  0.3  0.1    0   -0.1  -0.3;
       1    1    1    0.8  0.6  0.3  0.1    0   -0.1  -0.3  -0.6;
       1    1    0.8    0.6  0.3  0.1    0   -0.1  -0.3  -0.6  -0.8;
       1  0.8  0.6  0.3  0.1    0    0   -0.1  -0.3  -0.6  -0.8  -1;
       0.8  0.6  0.3  0.1    0   -0.1  -0.3  -0.6  -0.8  -1   -1;
       0.6  0.3  0.1    0   -0.1  -0.3  -0.6  -0.8  -1   -1   -1;
       0.3  0.1    0   -0.1  -0.3  -0.6  -0.8  -1   -1   -1   -1;
       0.1    0  -0.1  -0.3  -0.6  -0.8  -1   -1   -1   -1   -1;
       0 -0.1  -0.3  -0.6  -0.8  -1   -1   -1   -1   -1  -1]*g0;

x = 0;

while t <= duration

    ProcessNoise = accelnoise * [(step^2/2)*randn; step*randn];
    noise(index) = ProcessNoise(1);
    psi_r(index) = 10*sin(t);
    psi_a(index) = psi_r(index) + noise(index);
    MeasNoise(index) = measnoise * (randn);
    z(index) = psi_a(index) + MeasNoise(index);
    psi(index) = psi_r(index) + x;

```

```

w(index) = x;

% Fuzzy Filter

if counter == 10,
    counter = 0;
    c_count = 0; e_count = 0;
    % These are used to count the number of
    % non-zero mf certainties of e and c

    % First, for the given fuzzy filter inputs we determine
    % the extent at which the error membership functions
    % of the fuzzy controller are on (this is the
    % fuzzification
    % part).

    e(index) = (psi_r(index) - z(index));
    % Calculates the error input for the fuzzy filter

    c(index) = (e(index)-eold)/step;
    eold = e(index);
    % Save the past value of e for use in the
    % above
    % computation the next time around the loop

    % The following if-then structure fills the vector mfe
    % with the certainty of each membership function of e for
    % the
    % current input e. We use triangular membership
    % functions.

    if e(index) <= ce(1)
        % Takes care of saturation of the left-most mf
        mfe = [1 0 0 0 0 0 0 0 0 0];
        % i.e., the only one on is the
        % left-most one
        e_count = e_count+1; e_int = 1; % One mf on, it is the
        % left-most one.

    elseif e(index) >= ce(numc) % Takes care of saturation
        % of the right-most mf
        mfe = [0 0 0 0 0 0 0 0 0 1];
        e_count = e_count+1; e_int = numc;
        % One mf on, it is the
        % right-most one

    else
        % In this case the input is on the middle part of the
        % universe of discourse for e
        % Next, we are going to cycle through the mfs to
        % find all that are on

        for i = 1:numc
            if e(index) <= ce(i)
                mfe(i) = max([0,1+(e(index)-ce(i))/we]);
                % In this case the input is to the

```

```

        % left of the center ce(i) and we compute
        % the value of the mf centered at ce(i)
        % for this input e

if mfe(i) ~= 0
    % If the certainty is not equal to zero then say
    % that have one mf on by incrementing our count

    e_count = e_count+1;
    e_int = i; % This term holds the index last
               % entry with a non-zero term

end
else
    mfe(i) = max([0,1+(ce(i)-e(index))/we]);
    if mfe(i) ~= 0
        e_count = e_count+1;
        e_int = i;
    end
end
end
end

% The following if-then structure fills the vector mfc
% with the
% certainty of each membership function of the c

if c(index) <= cc(1)
    mfc = [1 0 0 0 0 0 0 0 0 0];
    % Takes care of saturation of left-most mf

    c_count = c_count+1;
    c_int = 1;
elseif c(index) >= cc(numc)
    % Takes care of saturation of the right-most mf

    mfc = [0 0 0 0 0 0 0 0 0 1];
    c_count = c_count+1;
    c_int = numc;
else
    for i = 1:numc
        if c(index) <= cc(i)
            mfc(i) = max([0,1+(c(index)-cc(i))/wc]);
            if mfc(i) ~= 0
                c_count = c_count+1;
                c_int = i; % This term holds last entry
                           % with a non-zero term

            end
        else
            mfc(i) = max([0,1+(cc(i)-c(index))/wc]);
            if mfc(i) ~= 0
                c_count = c_count+1;
                c_int = i; % This term holds last entry
                           % with a non-zero term

            end
        end
    end
end
end

```

```

        end
    end
end

% The next two loops calculate the crisp output using only the
% non-
% zero premise of error,e, and c. This cuts down computation
% time
% since we will only compute the contribution from the rules that
% are on (i.e., a maximum of four rules for our case). The
% minimum
% is used for the premise (and implication for the center-of-
% gravity
% defuzzification case).

    num = 0;
    den = 0;
    for k = (e_int-e_count+1):e_int
        % Scan over e indices of mfs that are on

        for l = (c_int-c_count+1):c_int
            % Scan over c indices of mfs that are on

            prem = min([mfe(k) mfc(l)]);
            % Value of premise membership function

% This next calculation of num adds up the numerator for the
% center of gravity defuzzification formula. rules(k,l) is the
% output center for the rule. base*(prem-(prem)^2/2) is the area
% of a symmetric
% triangle that peaks at one with base width "base" and that is
% chopped off at
% a height of prem (since we use minimum to represent the
% implication). Computation of den is similar but without
% rules(k,l).

            num = num + rules(k,l)*base*(prem-(prem)^2/2);
            den = den + base*(prem-(prem)^2/2);
        end
    end
    delta(index) = num/den;
    x = delta(index)/g0;
else
    % This goes with the "if" statement to check if the
    % counter=10
    % so the next lines up to the next "end" statement are
    % executed whenever counter is not equal to 10

    e(index) = e(index-1);
    c(index) = c(index-1);
    delta(index) = delta(index-1);
    x = delta(index)/g0;
end

time(index)=t;

```

```

        if delta(index) >= 20, delta(index) = 20; end
        if delta(index) <= -20, delta(index) = -20; end

        t = t + step;
        index=index+1;
        counter=counter+1;
    end

% Plot the results

figure(1)
clf
subplot(211)
plot(time,psi_r,time,z,'k--')
grid on
title('Desired Signal (solid) vs Noised Signal (dashed)')
xlabel('Time (sec)');
subplot(212)
plot(time,psi_r,time,psi,'k--')
grid on
title('Desired Signal (solid) vs Filtered Signal (dashed)')
xlabel('Time (sec)');

figure(2)
clf
subplot(211)
plot(time,e,'k-')
grid on
xlabel('Time (sec)')
title('Error between noised signal and desired signal')
subplot(212)
plot(time,c,'k-')
grid on
xlabel('Time (sec)')
title('Change in error')
zoom

figure(3)
clf
%subplot(211)
%plot(time,z)
hist(w)
grid on
%title('Crisp outputs of fuzzy filter');
%xlabel('Time (sec)');
%subplot(212)
%plot(time,noise)
%grid on
%title('Noise');
%xlabel('Time (sec)');

```

B-2 Fuzzy Filter for Four Inputs (signal error, change in the error, difference of error covariance P_k and P_k^- , change in this difference)

```
% Fuzzy Filter for Four Inputs (Signal error, change in the
% error, difference of error covariance P and Pest, change in
% this difference)

clear all

t = 0;
index = 1;
step = 0.2;
duration = 100; % length of simulation (seconds)
counter = 10;
eold = 0;
epold = 0;
measnoise = 10; % position measurement noise
accelnoise = 0.5; % acceleration noise
a = [1 step; 0 1]; % transition matrix
d = [1 0]; % measurement matrix
Q = accelnoise^2 * [step^4/4 step^3/2; step^3/2 step^2];
% process noise covariance
Pest = Q;
R = measnoise^2; % measurement error covariance
D = accelnoise * measnoise * [step^2/2; step];
alpha = 1 + 10^(-3);
% alpha = 1;

% Initialize parameters for the fuzzy filter
nume = 11;
numc = 11;
numep = 11;
numepc = 11;
g1 = 0.05; g2 = 0.005; g3 = 1; g4 = 5; g0 = 40;
%g1 = 5; g2 = 5; g3 = 10; g4 = 10; g0 = 40;
%g1 = 0.2; g2 = 0.014; g0 = 20;

% Next, define some parameters for the membership functions
% we = 0.2*(1/g1);
we = (0.2/g1);
    % we is half the width of the triangular input membership
    % function bases
% wc = 0.2*(1/g2);
wc = (0.2/g2);
    % Similar to we but for the c universe of discourse
% wep = 0.2*(1/g3);
wep = (0.1/g3);
wepc = (0.1/g4);
base = 0.4*g0;

% Place centers of membership functions of the fuzzy controller:
```

```

ce = [-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1]*(1/g1);
cc = [-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1]*(1/g2);
cep= [-1 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1 0]*(1/g3);
cepc= [-1 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1 0]*(1/g4);

rules=[1 1 1 1 1 1 0.8 0.6 0.3 0.1 0;
1 1 1 1 1 0.8 0.6 0.3 0.1 0 -0.1;
1 1 1 1 0.8 0.6 0.3 0.1 0 -0.1 -0.3;
1 1 1 0.8 0.6 0.3 0.1 0 -0.1 -0.3 -0.6;
1 1 0.8 0.6 0.3 0.1 0 -0.1 -0.3 -0.6 -0.8;
1 0.8 0.6 0.3 0.1 0 -0.1 -0.3 -0.6 -0.8 -1;
0.8 0.6 0.3 0.1 0 -0.1 -0.3 -0.6 -0.8 -1 -1;
0.6 0.3 0.1 0 -0.1 -0.3 -0.6 -0.8 -1 -1 -1;
0.3 0.1 0 -0.1 -0.3 -0.6 -0.8 -1 -1 -1 -1;
0.1 0 -0.1 -0.3 -0.6 -0.8 -1 -1 -1 -1 -1;
0 -0.1 -0.3 -0.6 -0.8 -1 -1 -1 -1 -1]*g0;

x = 0;
C = 0;

%for t = 0 : dt: duration,
while t <= duration

    ProcessNoise = accelnoise * [(step^2/2)*randn; step*randn];
    noise(index) = ProcessNoise(1);
    psi_r(index) = 10*sin(t);
    psi_a(index) = psi_r(index) + noise(index);
    MeasNoise(index) = measnoise * (randn);
    z(index) = psi_a(index) + MeasNoise(index);
    psi(index) = psi_r(index) + x;
    w(index) = x;
    % Covariance of Innovation
    s = d * Pest * d' + R;
    % Covariance of prediction error
    K = Pest * d' * inv(s);
    u(index) = K(1);
    P = (Pest - K * d * Pest);
    % Pest = a * P * a' + Q - D * inv(s) * D' - a * K * D' - D *
    % K' * a';
    Pest = (a * P * a' + Q);

    % Fuzzy Filter
    if counter == 10,
        counter = 0;
        c_count = 0; e_count = 0;
        ep_count = 0; epc_count = 0;
        e(index) = (psi_r(index) - z(index));
        c(index) = (e(index)-eold)/10;
        eold = e(index);
        ep(index) = P(1) - Pest(1);
        epc(index) = (ep(index)-epold)/10;
        epold = ep(index);

        if e(index) <= ce(1)
            mfe = [1 0 0 0 0 0 0 0 0 0];
            e_count = e_count+1; e_int = 1;
        elseif e(index) >= ce(numel(ce))

```

```

        mfe = [0 0 0 0 0 0 0 0 0 0 1];
        e_count = e_count+1; e_int = nume;
    else
        for i = 1:nume
            if e(index) <= ce(i)
                mfe(i) = max([0,1+(e(index)-ce(i))/we]);
                if mfe(i) ~= 0
                    e_count = e_count+1;
                    e_int = i;
                end
            else
                mfe(i) = max([0,1+(ce(i)-e(index))/we]);
                if mfe(i) ~= 0
                    e_count = e_count+1;
                    e_int = i;
                end
            end
        end
    end
end

if c(index) <= cc(1)
    mfc = [1 0 0 0 0 0 0 0 0 0 0];
    c_count = c_count+1;
    c_int = 1;
elseif c(index) >= cc(numc)
    mfc = [0 0 0 0 0 0 0 0 0 0 1];
    c_count = c_count+1;
    c_int = numc;
else
    for i = 1:numc
        if c(index) <= cc(i)
            mfc(i) = max([0,1+(c(index)-cc(i))/wc]);
            if mfc(i) ~= 0
                c_count = c_count+1;
                c_int = i;
            end
        else
            mfc(i) = max([0,1+(cc(i)-c(index))/wc]);
            if mfc(i) ~= 0
                c_count = c_count+1;
                c_int = i;
            end
        end
    end
end
end

if ep(index) <= cep(1)
    mfep = [1 0 0 0 0 0 0 0 0 0 0];
    ep_count = ep_count+1; ep_int = 1;
elseif ep(index) >= cep(numep)
    mfep = [0 0 0 0 0 0 0 0 0 0 1];
    ep_count = ep_count+1; ep_int = numep;
else
    for i = 1:numep
        if ep(index) <= cep(i)
            mfep(i) = max([0,1+(ep(index)-cep(i))/wep]);
            if mfep(i) ~= 0

```



```

        ep_count = ep_count+1;
        ep_int = i;
    end
else
    mfep(i) = max([0,1+(cep(i)-ep(index))/wep]);
    if mfep(i) ~= 0
        ep_count = ep_count+1;
        ep_int = i;
    end
end
end
end
end

if epc(index) <= cepc(1)
    mfepec = [1 0 0 0 0 0 0 0 0 0];
    epc_count = epc_count+1;
    epc_int = 1;
elseif epc(index) >= cepc(numepc)
    mfepec = [0 0 0 0 0 0 0 0 0 1];
    epc_count = epc_count+1;
    epc_int = numepc;
else
    for i = 1:numepc
        if epc(index) <= cepc(i)
            mfepec(i) = max([0,1+(epc(index)-cepc(i))/wepc]);
            if mfepec(i) ~= 0
                epc_count = epc_count+1;
                epc_int = i;
            end
        else
            mfepec(i) = max([0,1+(cepc(i)-epc(index))/wepc]);
            if mfepec(i) ~= 0
                epc_count = epc_count+1;
                epc_int = i;
            end
        end
    end
end
end
end

num = 0;
den = 0;
%num1 = 0;
%den1 = 0;
for k = (e_int-e_count+1):e_int
    for l = (c_int-c_count+1):c_int
        for m = (ep_int-ep_count+1):ep_int
            for n = (epc_int-epc_count+1):epc_int
                prem = min([mfe(k) mfc(l) mfep(m) mfepec(n)]);
                num = num + rules(k,l) * base * (prem-
                    (prem)^2/2);
                den = den + base * (prem-(prem)^2/2);
            end
        end
    end
end
end
delta(index) = num/den;

```

```

        %for m = (ep_int-ep_count+1):ep_int
        %   for n = (epc_int-epc_count+1):epc_int
        %       preml = min([mfep(m) mfepc(n)]);
        %       numl = numl + rules1(m,n)*base1*(preml-
        %           (preml)^2/2);
        %       denl = denl + base1*(preml-(preml)^2/2);
        %   end
        %end
        %deltal(index) = numl/denl;

        %x = (delta(index)/g0 + deltal(index)/g01)/2;
        x = delta(index)/g0;

    else
        e(index) = e(index-1);
        c(index) = c(index-1);
        ep(index) = ep(index-1);
        epc(index) = epc(index-1);
        delta(index) = delta(index-1);
        %deltal(index) = deltal(index-1);
        %x = (delta(index)/g0 + deltal(index)/g01)/2;
        x = delta(index)/g0;
    end

    time(index)=t;

    if delta(index) >= 40, delta(index) = 40; end
    if delta(index) <= -40, delta(index) = -40; end

    t = t + step;
    index = index+1;
    counter = counter+1;
    C = C + 1;
    Q = Q * (alpha ^ (-2 * (C)));
    R = R * (alpha ^ (-2 * (C)));

end

% Plot the results
%t = 0 : dt : duration;
figure(1)
clf
%subplot(211)
%plot(time,psi_r,time,z,'k--')
%grid on
%title('Desired Signal (solid) vs Noised Signal (dashed)')
%xlabel('Time (sec)');
%subplot(212)
% plot(time,psi_r,'k',time,psi, '--')
plot(time,psi)
grid on
ylabel('Estimated Position Trajectory')
xlabel('Time (sec)');

figure(2)
clf

```

```

subplot(211)
plot(time,e,'k-')
grid on
xlabel('Time (sec)')
title('Error between noised signal and desired signal')
subplot(212)
plot(time,c,'k-')
grid on
xlabel('Time (sec)')
title('Change in above error')
zoom

figure(3)
subplot(211)
plot(time,ep,'k-')
grid on
%title('Crisp outputs of fuzzy filter');
title('Error between P and Pest')
xlabel('Time (sec)');
subplot(212)
plot(time,epc,'k-')
grid on
title('Change in above error');
xlabel('Time (sec)');

figure(4)
% subplot(211)
plot(time,u,'k-')
grid on
%title('Crisp outputs of fuzzy filter');
title('Gain K')
xlabel('Time (sec)');

```