

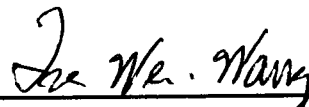
To the Graduate Council:

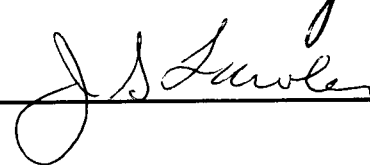
I am submitting herewith a thesis written by Sze Wei Ong entitled "Finite Impulse Response Hybrid System Modeling." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.



J. D. Birdwell, Major Professor

We have read this thesis
and recommend its acceptance:





Accepted for the Council:



Associate Vice Chancellor
and Dean of The Graduate School

**FINITE IMPULSE RESPONSE HYBRID SYSTEM
MODELING**

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Sze Wei Ong

August 1997

Copyright © Sze Wei Ong, 1997
All rights reserved

ACKNOWLEDGEMENTS

I thank my advisor, Professor J. Douglas Birdwell, for his guidance and encouragement during this research. I also thank my other committee members, Dr. Jack S. Lawler and Dr. Tse-Wei Wang, for their guidance and useful suggestions. Appreciation is also extended to Dr. Roger Horn, Mr. Kenneth Swayne and Dr. Peter Hansen for their help and advice.

This research was supported by grants from E. I. DuPont and the University of Tennessee College of Engineering Measurement and Control Engineering Center (MCEC). Process data utilized to evaluate the FIRM modeling technology were provided by a third party which agreed to its use in a manner which does not disclose the process; this data is not related to processes or technologies owned or operated by DuPont or other MCEC member companies. Software was provided by Data Refining Technologies, Inc., to generate signal object streams from raw data. This software is available at no cost for research purposes in off-line analysis only.

ABSTRACT

This research presents a new approach to process modeling which attempts to capture the dynamic behavior of object-oriented signals. The new modeling approach, finite impulse response hybrid system modeling, is a method that combines the theories of Markov chains and finite impulse response (FIR) filters. The finite impulse response hybrid system can be used to represent a special class of dynamic processes where relationships among process variables are poorly understood and hidden inside a large quantity of available data. Traditional mathematical methods of modeling do not offer the flexibility provided by the new approach since no assumptions of linearity or other restrictions of process behavior are made. A digital modeling and simulation package, implemented using Matlab is developed. The effectiveness of finite impulse response Markov chain has been demonstrated by its application to modeling a highly nonlinear exothermic reaction process. The performance of the model accurately predicts the future output events of a process based on a set of past input events.

The new modeling approach is best suited for processes which generate large and messy data in which clear cause-effect relationships are not always clearly observed or understood. The model thus developed has a high potential for it to be used in subsequent process control applications, to allow for tighter control possible than that of conventional modeling approach.

TABLE OF CONTENTS

CHAPTER	PAGE
1. Introduction	1
2. Background	5
2.1 Markov chains	6
2.1.1 Probability transition matrix	9
2.1.2 Multistep transition process	10
2.1.3 Regular Markov chain and limiting distribution	11
2.1.4 Classification of states	12
2.2 Finite impulse response filters	15
2.2.1 Properties of FIR filters	15
2.2.2 Basic network structures for FIR filters	18
2.2.3 FIR filter design	21
2.3 Object-oriented signal processing	21
2.4 Partition and refinement of time interval	26
2.5 Discrete event systems	27
2.6 Summary	29
3. FIRM Modeling	30
3.1 Condensation of raw measurement data	30

CHAPTER	PAGE
3.2 Refinement of input and output time partitions	31
3.3 State representation of object-oriented signal	44
3.3.1 Discretization and indexing of object height, duration and midpoint	44
3.3.2 Object state representation	46
3.3.3 System state representation	50
3.4 Probability transition matrix, P	51
3.4.1 Properties of the probability transition matrix	52
3.5 Simulation diagram of the model	54
3.6 Summary	55
4. Size, Complexity and Prediction Error of Model	57
4.1 Size of P	58
4.1.1 Relationship between size of P and N	58
4.1.2 Relationship between size of P and n	63
4.1.3 Relationship between size of P and s	64
4.1.4 Relationship between size of P and (N, n, s)	67
4.2 Complexity of P	72
4.2.1 Relationship between complexity of P and N	72
4.2.2 Relationship between complexity of P and n	74
4.2.3 Relationship between complexity of P and s	76
4.2.4 Relationship between complexity of P and (N, n, s)	76
4.2.5 Significance of this relationship	76

CHAPTER	PAGE
4.3 Prediction error	80
4.3.1 Prediction error at time t	80
4.3.2 Time averaged prediction error	85
4.3.3 Relationship between prediction error and N	86
4.3.4 Relationship between prediction error and n	87
4.3.5 Relationship between prediction error and s	92
4.3.6 Relationship between prediction error and (N, n, s)	98
4.4 Summary	98
5. Protocol for Application	101
5.1 2^k factorial experimental design	101
5.2 Model identification	105
5.2.1 Identification Techniques	106
5.2.2 Model validation	110
5.3 General procedure for FIRM modeling	112
5.4 Simulation	112
5.4.1 Selection of training data set	114
5.4.2 Simulation results	115
5.5 Summary	123
6. Summary and Future Research Directions	125
6.1 Research summary	126
6.2 Future research directions	127

CHAPTER	PAGE
BIBLIOGRAPHY	129
References	130
APPENDICES	132
A. Matlab code list of the refinement process	133
B. Matlab code list for encoding objects to object states	137
C. Matlab code list for calculating P	139
VITA	141

LIST OF TABLES

TABLE	PAGE
3.1 Possible combinations of i_h, i_d and i_m	47
5.1 Relationships between model features and model parameters.	107
5.2 Rules for initial estimates of parameters	111
5.3 Listing of groups of n events.	116

LIST OF FIGURES

FIGURE	PAGE
2.1 An example Markov chain.	8
2.2 $H(f)$	17
2.3 $h(t)$	17
2.4 Direct form realization of an FIR system.	19
2.5 Transposed direct form realization of an FIR system.	19
2.6 Cascade form realization of an FIR system.	20
2.7 A representative measurement signal.	24
2.8 Condensed form of signal in the figure above.	24
2.9 An interval scale and interval representations.	25
2.10 Partition and norm of $[a,b]$	28
3.1 Input signal of the training data set.	32
3.2 Condensed input signal of the training data set.	33
3.3 Output signal of the training data set.	34
3.4 Condensed output signal of the training data set.	35
3.5 An object-oriented signal and its time partition.	37
3.6 A segment of the input object-oriented signal.	38
3.7 A segment of the output object-oriented signal.	39
3.8 Refinement using $T \cup S$	41
3.9 The segment of input object-oriented signal after refinement.	42

FIGURE	PAGE
3.10 The segment of output object-oriented signal after refinement.	43
3.11 Sample signal.	48
3.12 Object state of the sample signal.	49
3.13 Sequences of system states and output object states.	53
3.14 Simulation diagram for the FIRM model.	56
4.1 Number of rows of P versus N , ($n = 10, s = 492$).	60
4.2 Number of columns of P versus N , ($n = 10, s = 492$).	61
4.3 Number of columns of P versus n , ($N = 20, s = 188$).	65
4.4 Number of rows of P versus n , ($N = 20, s = 188$).	66
4.5 Number of columns of P versus s , ($N = 40, n = 10$).	68
4.6 Number of rows of P versus s , ($N = 40, n = 10$).	69
4.7 Number of rows of P versus (N, n) , ($s = 188$).	70
4.8 Number of columns of P versus (N, n) , ($s = 188$).	71
4.9 Complexity of P versus N , ($n = 10, s = 188$).	73
4.10 Complexity of P versus n , ($N = 20, s = 188$).	75
4.11 Complexity of P versus s , ($N = 20, n = 4$).	77
4.12 Complexity versus (N, n) , ($s = 188$).	78
4.13 Complexity versus (N, n)	79
4.14 An example of true signal objects.	82
4.15 Probability density function of x_{true}	82
4.16 An example predicted signal objects.	83
4.17 Probability density function of x_{pred}	83

FIGURE	PAGE
4.18 Object sequences of true and predicted output.	84
4.19 True output.	88
4.20 Predicted output of model with $N = 1, (n = 5, s = 95)$	88
4.21 Predicted output of model with $N = 3, (n = 5, s = 95)$	89
4.22 Predicted output of model with $N = 5, (n = 5, s = 95)$	89
4.23 Predicted output of model with $N = 10, (n = 5, s = 95)$	90
4.24 Predicted output of model with $N = 50, (n = 5, s = 95)$	90
4.25 Prediction error versus $N, (n = 10, s = 188)$	91
4.26 True output.	93
4.27 Predicted output of model with $n=1, (N = 50, s = 91)$	93
4.28 Predicted output of model with $n=2, (N = 50, s = 91)$	94
4.29 Predicted output of model with $n=5, (N = 50, s = 91)$	94
4.30 Predicted output of model with $n=15, (N = 50, s = 91)$	95
4.31 Estimated 2 norm error vs. $n, (N = 30, s = 233)$	96
4.32 Estimated 2 norm error vs. size of training data set, $(N = 40, n = 10)$	97
4.33 Error versus $(N, n), (s = 188)$	99
4.34 Error versus $(N, n), (s = 80)$	100
5.1 Factorial experimental design using data set 1.	103
5.2 Factorial experimental design using data set 2.	104
5.3 A general paradigm for implementing FIRM modeling.	113
5.4 Prediction error versus $N (n = 10, s = 80)$	118
5.5 Percentage of predictable objects versus $N (n = 10, s = 80)$	118

FIGURE	PAGE
5.6 Prediction error versus n.	119
5.7 Percentage of predictable objects versus n.	119
5.8 True output of the model validation set.	120
5.9 Predicted output.	120
5.10 True output of the model validation set.	121
5.11 Predicted output.	121
5.12 True output of the model validation set.	122
5.13 Predicted output.	122
5.14 Percentage of predictable objects versus s.	124
5.15 Prediction error versus s.	124

CHAPTER 1

Introduction

This thesis presents a new approach to process modeling which attempts to capture the dynamic behavior of processes given the collected input and output object-oriented signals. The new modeling approach is based on the theories of Markov chains [1, 2] and finite impulse response (FIR) filters [3, 4]. Finite impulse response Markov chain (FIRM) models are a generalization of Markov chains. Models for several sets of data obtained from an operating industrial process have been developed and tested. This paper discusses the algorithms for developing, identifying and validating the FIRM model.

The intent of this research is to provide a protocol that can be used on industrial applications to determine, first, whether the FIRM methodology is applicable to a process, and second, model parameters which can be used to accurately represent the process.

Several categories of industrial process operations are not amenable to traditional mathematical methods of modeling, analysis and control. This is due to either poorly understood physical relationships, or overly complex process models, as with systems governed by partial differential equations. In this type of process, sequences of input and output, or time series [5, 6, 7], are available though the relationship between these

sequences are not well understood.

In most of these processes, the primary goals are to operate the process safely, to maximize profitability, and to minimize product variability. Often, process variation is caused by unobserved small perturbations which over time propagate through process dynamics and are eventually manifested by poor product quality or an event which causes shutdown. Either condition results in lost revenue which can quickly dominate any cost savings realized by traditional control algorithms.

This research develops a new approach to modeling time series by combining the theories of finite impulse response filters to Markov chains. The new process modeling approach is capable of handling applications where analytical equations of the process characteristics are nonexistent. In the model, continually changing process characteristics hidden in the time series are represented by a probability transition matrix which relates the output to a set of past inputs of the process. The developed model can be used to predict the future output of the process based on a set of system inputs. This predictive modeling method represents a new and improved version of conventional Markov chains.

A significant amount of attention has been paid to the research on Markov chain theories and its applications since its introduction in 1906 [2]. Many theories and applications, including the control of Markov chains have been developed. For the purpose of this research, particular interest is paid to the application of Markov chains in modeling industrial processes. These modeling methodologies have been widely accepted by numerous industrial communities, making Markov chains an important

branch of dynamical systems methodologies.

Markov chains can be used to model dynamic systems that evolve probabilistically. However, there are some disadvantages in using the conventional Markov chain for predicting the process output. In a conventional Markov chain, the output of the process is predicted based on a single input to the process. The model is described by a probability transition matrix, P , that probabilistically relates the current input of the process to the output. In many applications, an output event of the process is caused by a sequence of past inputs. This is prevalent in a large number of manufacturing processes where a product defect is often caused by a sequence of machine states. Since conventional Markov chain modeling only uses the present input of the process to predict the system output, the model is incapable of detecting the outcome of an event if it is caused by a sequence of past inputs. In addition, prediction based on a single input event is often less accurate for processes with repeating input events. By using a traditional Markov chain, predicted output events of the repeated input will have a stationary probability distribution even though a process' output characteristics in response to a constant input can change over time.

These disadvantages of Markov chains can be solved by introducing the concept of finite impulse response to the Markov chain. Instead of using a single input, the finite impulse response Markov chain uses a sequence of past system inputs as the input state for prediction of the system output. By using a sequence of inputs rather than a single input, repetition of system states can be reduced. This will increase the prediction accuracy of the system output. Furthermore, by relating the process output to a set of

past inputs, the model is able to detect undesired events such as product defects and variations which are often caused by a sequence of system states.

The thesis covers three main areas of FIRM modeling : development of the FIRM model, characteristics of the model, and model identification and verification techniques. Chapter 2 covers background information and contains a brief description of the traditional Markov chain as well as information on the finite impulse response filter. Also included is information on the object-oriented signal processing method used in the research to condense the raw input and output signals. Since the development of the FIRM model uses the definition of time partitions from calculus, information on time partitions and refinements is included. Chapter 3 describes the design method for developing the FIRM model from a set of raw input and output signals. Next, Chapter 4 discusses the features of the model. Three main features of the model are presented in this chapter: model size, model complexity and model prediction error. The relationships between these features and model parameters are examined. Chapter 5 covers the model identification and verification techniques for a FIRM model. A FIRM model is then developed from a set of data obtained from an industrial process using these techniques. Simulation is performed on the developed model to predict the process output. Finally, Chapter 6 summarizes FIRM model design methods and identification strategies, and discusses possible avenues for future research.

CHAPTER 2

Background

This chapter presents a survey of literature relevant to the research. A rapid review of the classical modeling approaches which are the basis for FIRM modeling is presented. Because of the large volumes of raw measurement data, data condensation is required in this research. Therefore, the data condensation technology used in this research is introduced. Some fundamental system and calculus definitions which are applied in the research are also presented.

The new modeling approach in this research is developed based upon the familiar concepts of Markov chains and finite impulse response filters. The first two sections in this chapter are devoted to these concepts. Section 2.1 reviews the theory of Markov chains. In this section, the Markov chain model is presented as a classical method for process modeling. Section 2.2 concerns the basic concepts of finite impulse response filter. The properties and design of finite impulse response filters will be discussed in this section.

The new modeling approach produces a mathematical model of process from the condensed input-output signal. Therefore, before process modeling can be done, raw measurement data from the process are condensed. A third section in this chapter describes the condensation technology used in the research.

The last two sections of the chapter provide some basic background for system and calculus definitions. These definitions are of paramount importance in the process of developing the process model.

2.1 Markov chains

A Markov chain is a process model that represents a special class of dynamic systems which evolve probabilistically. It was introduced by Markov in 1906 [2] to represent the dependence between successive random variables in a process. Since then, a wide variety of theories and applications of Markov chain have been developed. The intense growth of theory and application has made it an important branch of dynamic systems.

A Markov chain can be visualized in terms of a pointer that moves around among a set of states. However, in a Markov chain the state transitions occur probabilistically rather than deterministically. The probabilistic evolution of the states implies that future states can not be inferred from the present, except in terms of probability assessment. In general, the future evolution of a Markov process is described by a vector of probabilities.

A Markov chain process is composed of two essential components, system states and state transition probabilities. The state of the model provides a measure of the process feature of interest to the controller, whereas the state transition provides information about the process' dynamic behavior. An n^{th} order Markov chain is characterized by a set of n states $\{x^1, x^2, \dots, x^n\}$ and a set of transition probabilities p_{ji} , where p_{ji} is

the probability of the model occupying state x^j at the next time interval given that the model is in state x^i at the current time interval.

An example Markov chain models the outcomes of the games played by a certain team in a football season. The possible outcomes of the team in a football game can be characterized as win, lose or draw. Let us assume that if the team wins one game in a certain week, the chances of the team winning, losing, or tying the next game will be sixty, ten and thirty percent respectively. If the team loses the game, then there is a thirty percent chance that the team will win the next game, sixty-five percent chance that the team will lose the next game and five percent chance that the team will tie the next game. If the team ties the game, then the chances of the team winning, losing, or tying the next game will be forty-five, forty-five and ten percent respectively. Denoting the three outcomes of a game by W, L and D, respectively, the dynamic characteristics of this model can be represented by a 3×3 probability transition matrix

$$\underline{P} = \begin{bmatrix} 0.60 & 0.30 & 0.45 \\ 0.10 & 0.65 & 0.45 \\ 0.30 & 0.05 & 0.10 \end{bmatrix} \quad (2.1)$$

Alternatively, this model can be represented by a diagram as shown in Figure 2.1.

The process starts at a certain game with the outcome of that game as the state and moves, each week, to a new game with a new outcome. Exact prediction of which transition will occur is impossible. Only a probabilistic assessment of the future evolution can be made.

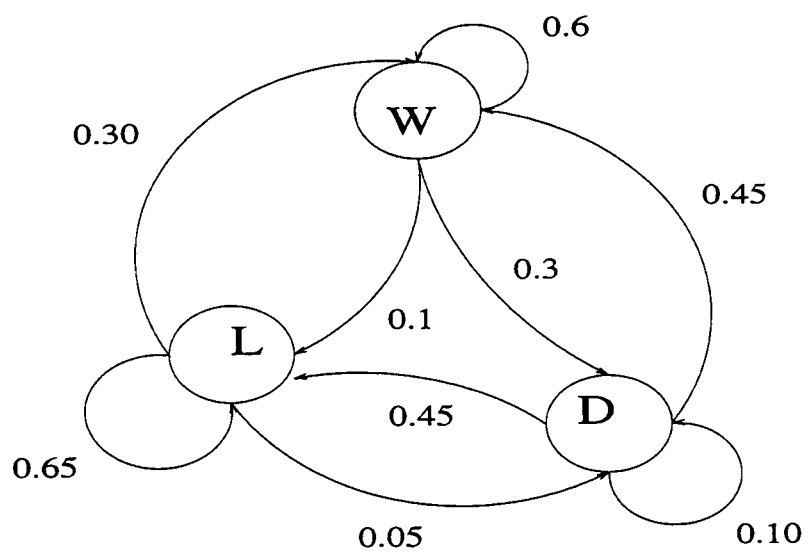


Figure 2.1: An example Markov chain.

This process is an example of a finite Markov chain where the state space of the process is a finite set. In the following sections, only finite Markov chains will be considered.

2.1.1 Probability transition matrix

The state transition dynamics of a finite Markov process are described by a $n \times n$ probability transition matrix $\underline{P} = [p_{ji}]$ where each column of $\underline{P}$ contains non-negative elements which sum to 1. This can be seen in Equation 2.1 of the example above. The transition matrix of a condensed signal can be inferred from the frequency of each transition's occurrence. For each transition from state x^i to state x^j , the transition probability p_{ji} is obtained by dividing the total number of transitions from state x^i to state x^j by the total number of state x^i occurrences. Typically, the Markov model will have many states, and the transition matrix will be sparse.

The Markov chain model can be used to predict the future behavior of a stream of signal objects, based upon a single known state. If the process occupies state x^i at the current time interval, the probability of the process occupying each of the n states at the next time interval is described by the column vector

$$p_i = [p_{1i} \ p_{2i} \ \dots \ p_{ni}]^T.$$

The state dynamics can then be written as

$$\underline{p}_{k+1} = \underline{P} * \underline{p}_k \tag{2.2}$$

The probability density

$$\underline{p}_k = [0 \cdots 1 \cdots 0]^T \quad (2.3)$$

where the 1 is in the i^{th} place, indicates certain knowledge that the state of the Markov chain after the k^{th} state transition is x^i . The output probability vector

$$\underline{p}_{k+1} = [p_{1i} \ p_{2i} \cdots p_{ni}]^T \quad (2.4)$$

can be used to estimate the probability of the process occupying various states at the next time interval. If the process is initiated in state x^i , after one transition it will be in state x^j with probability p_{ji} .

2.1.2 Multistep transition process

The matrix $\underline{P}$ gives the probability of the process occupying each of the n possible states $\{x^1, x^2, \dots, x^n\}$ after one step, given the probability of the process occupying each state at the present time instant, independent of the number of steps that have occurred. From this matrix, the probability transition matrix for a multistep transition can be derived.

To show the derivation of the multistep transition matrix, we first examine the Markov process after two steps. Starting at a given initial state x^i , the process will undergo two transitions and arrive at some state x^j . The overall transition from x^i to x^j is governed by two applications of the single step probability transition matrix $\underline{P}$. An $n \times n$ matrix $\underline{P}^2$ can be used to represent the two steps transition probabilities of the process.

To derive the matrix $\underline{P}^2$, suppose that after one step, the process is in some state x^k . The probability that the state is x^k after one step is given by p_{ki} . For the second step, the probability that the process will reach the state x^j , starting from x^k , is equal to p_{jk} . Summing over all possible first steps, the probability of reaching x^j from x^i in two transition is

$$p_{ji}^{(2)} = \sum_{k=1}^n p_{ki} p_{jk} = [P^2]_{ji} \quad (2.5)$$

Notice that the probability $p_{ji}^{(2)}$ is the ji^{th} element of the two step transition matrix $\underline{P}^2$. A more direct way of obtaining $\underline{P}^2$ is given by

$$\underline{P}^2 = \underline{P} * \underline{P} \quad (2.6)$$

Similarly, the transition probabilities for m steps are defined by the element of the matrix $\underline{P}^m$. $\underline{P}^m$ can be found using Equation 2.6.

2.1.3 Regular Markov chain and limiting distribution

A Markov chain is said to be regular [8] if $\underline{P}^m > 0$ for some positive integer m. The regularity of a Markov chain implies that over a sufficiently large number of steps every state among the n states is possible. This is because there is a strictly positive probability associated with each state. For this type of Markov chain, an unique limiting probability vector can be derived for the process.

If $\underline{P}$ is the probability transition matrix of a regular Markov chain, there is a unique probability vector $p > 0$ such that

$$\underline{P}p = p \quad (2.7)$$

The theorem above is called basic limit theorem for Markov chains. This theorem also states that for any initial state i with an initial probability vector equal to the i^{th} coordinate vector e_i , the limit vector

$$v = \lim_{m \rightarrow \infty} \underline{P}^m e_i \quad (2.8)$$

exists and is independent of i . Furthermore, v is equal to the eigenvector of p corresponding to an eigenvalue $\lambda = 1$. Alternatively, these theorems can be stated as

$$\lim_{m \rightarrow \infty} \underline{P}^m = \overline{P} \quad (2.9)$$

where $\overline{P}$ is the $n \times n$ matrix with each column equal to p . This result shows that if the Markov chain is regular, the long term probabilities are independent of the initial condition. In other words, starting at any initial state, after a large number of steps the probability of the chain reaching the state x^i is equal to the i^{th} component of p . Therefore, the limiting distribution of a regular Markov chain is determined by the limit probability vector regardless of the initial condition.

2.1.4 Classification of states

If the ji^{th} entry of the probability transition matrix of a Markov chain is zero, the transition from state x^i to state x^j in one step is impossible. However, this transition is possible in m steps if there exists an m such that ji^{th} entry of $\underline{P}^m$ is nonzero. Therefore, the presence of zero entries in a transition matrix can determine the possibility of transition from one state to another. In this section, the presence of zero entry is used to study the structure of state interconnections.

State x^j is said to be accessible from another state x^i if it is possible to begin at x^i and arrive at x^j in some finite number of steps. The accessibility can be determined from the entries of P^m . If the ji^{th} entry of P^m is nonzero for some m steps, then x^j is accessible from x^i since there is a positive probability that the Markov chain will make such a transition in m steps. Thus, x^j is accessible from x^i if and only if $p_{ji}^{(m)} > 0$ for some integer $m > 0$. Notice that the property of accessibility is nonsymmetric since $p_{ji}^{(m)}$ is not always equal to $p_{ij}^{(m)}$. This leads us to the definition of communicating states.

States x^i and x^j are said to communicate if each is accessible from the other. Communicating states divide the states of a Markov chain into distinct classes with different identities. Each of the states from the set of n states belongs to exactly one class. These classes of states are called communicating class. Therefore, the set of n states of a Markov chain can be divided into a set of communicating classes. Each state within a communicating class can only communicate with every other state in the same class. A special case is when all the states communicates with each other. In this case, there is only one communicating class in the Markov chain. If a Markov chain contains only a single communicating class, it is said to be irreducible.

There are several different categories of communicating classes. A closed communicating class is the class where no transition from any states in the class to states outside the class is possible. In other words, no state outside the closed class is accessible from any states in the class. Once the state of a Markov chain evolve into a closed class, it can never come out from the class. Therefore it is clear that the state

of a finite Markov chain will eventually enter some closed communicating class. The probability of eventually entering the closed communicating class is 1. Closed classes are sometime referred to as absorbing classes since they absorb the process state. If the closed communicating class contains only a single state, that particular state will be named the absorbing state.

A communicating class is transient if some state outside of the class is accessible from the states inside the class. A transient class has at least one path leading from one of the members inside the class to some state outside the class. Therefore, regardless of where the process starts, there is a positive probability of arriving at that exiting state in a finite number of steps. The probability of eventually leaving the transient class is 1.

From the definition of closed and transient classes, it can be seen that a Markov chain will eventually go into some closed communicating class, regardless of the initial state of the process. If the starting state is in a closed class, the process will move around inside the class and never come out from the class. If the starting state is a member of a transient class, the process will eventually exit the transient class and enter some closed class. The closed class of a Markov chain represents the equilibrium distribution of the process. More generally, the equilibrium eigenvector of $\underline{P}$ is associated with the closed communicating classes of $\underline{P}$.

2.2 Finite impulse response filters

Finite impulse response (FIR) filters [9] are linear time invariant systems in which the system outputs depend on the present input and a finite number of past input samples. This type of filter is referred to as a nonrecursive filter because of the absence of feedback loops in its implementation. It is a finite memory filter where the output is a function of input data of limited time. Equation 2.10 represents a single-output/single-input FIR filter.

$$Output_{k+1} = F(Input_k, Input_{k-1}, \dots, Input_{k-n+1}) \quad (2.10)$$

2.2.1 Properties of FIR filters

The properties of FIR filters will be illustrated by the following example. Consider a signal $x(t)$, sampled at the frequency of $f_s = 1/T$. The signal can also be represented by its samples $x(nT)$. Let the output signal $y(t)$ be a function of $x(t)$ and $x(t - T)$ as shown in Equation 2.11.

$$y(t) = \frac{1}{2}[x(t) + x(t - T)] \quad (2.11)$$

The sampled $y(t)$ can then be given by

$$y(nT) = \frac{1}{2}[x(nT) + x((n - 1)T)] \quad (2.12)$$

To examine the frequency response of this filter, let $X(f)$ and $Y(f)$ be the Fourier transforms of the signals $x(t)$ and $y(t)$ respectively. We obtain

$$Y(f) = \frac{1}{2}X(f)(1 + e^{-j2\pi fT}) \quad (2.13)$$

The transfer function $H(f)$ of the filter can be found by dividing $Y(f)$ by $X(f)$.

$$H(f) = Y(f)/X(f) = e^{-j\pi fT} \cos \pi fT \quad (2.14)$$

In Equation 2.14, the complex term $e^{-j\pi fT}$ represents a delay $\tau = T/2$ which is the propagation time of the signal through the filter. The impulse response $h(t)$ corresponding to $H(f)$ in Equation 2.14 is

$$h(t) = \frac{1}{2}[\delta(t + \frac{T}{2}) + \delta(t - \frac{T}{2})]. \quad (2.15)$$

Figures 2.2 and 2.3 show the characteristics of the system. The effect of $h(t)$ in the equation above is often called cosine filtering. From Figure 2.2, it can be seen that the filter preserves the zero frequency component and eliminates the component at frequency $f_s/2$.

From this example, the properties of FIR filters can be listed as below.

1. The output $y(n)$ is a function of the present input and a set of past inputs (as shown in Equation 2.16).

$$y(n) = \sum_{i=0}^{N-1} a_i x(n-i) \quad (2.16)$$

2. The transfer function of the filter is

$$H(f) = \sum_{i=0}^{N-1} a_i e^{-j2\pi f iT} \quad (2.17)$$

3. The frequency response of the filter is periodic with period $f_s = 1/T$. The coefficients a_i (for $0 \leq i \leq N-1$) form the Fourier series expansion of this function.

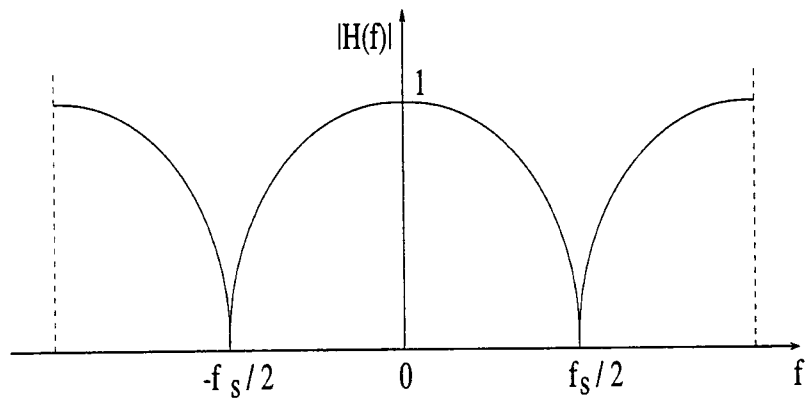


Figure 2.2: $H(f)$

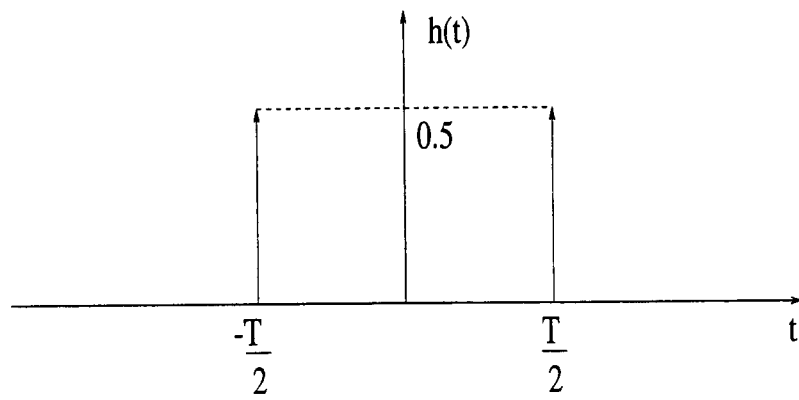


Figure 2.3: $h(t)$

2.2.2 Basic network structures for FIR filters

FIR systems can be implemented in a variety of forms. In this section, some of the most important network forms for FIR systems [10] are presented.

Direct form

The direct form realization [10] of a FIR system follows directly from Equation 2.16. This structure is the most straightforward realization of the additions and multiplications in Equation 2.16. A signal flow graph for this type of realization is shown in Figure 2.4.

Transposed direct form

The transposition theorem [10] can be applied to Figure 2.4 to obtain the transposed direct form realization structure of Figure 2.5.

Cascade form

The cascade form [10] can be realized by expressing $H(z)$ as a product of second-order factors (as in Equation 2.18)

$$H(z) = \prod_{k=1}^{n/2} (a_{0k} + a_{1k}z^{-1} + a_{2k}z^{-2}) \quad (2.18)$$

The cascade realization corresponding to Equation 2.18 is shown in Figure 2.6.

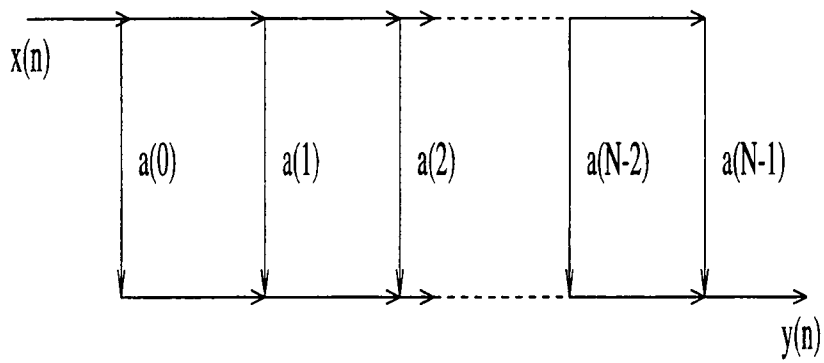


Figure 2.4: Direct form realization of an FIR system.

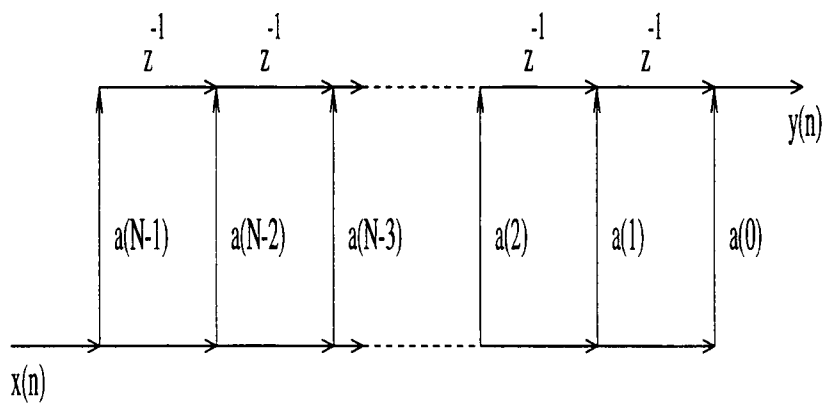


Figure 2.5: Transposed direct form realization of an FIR system.

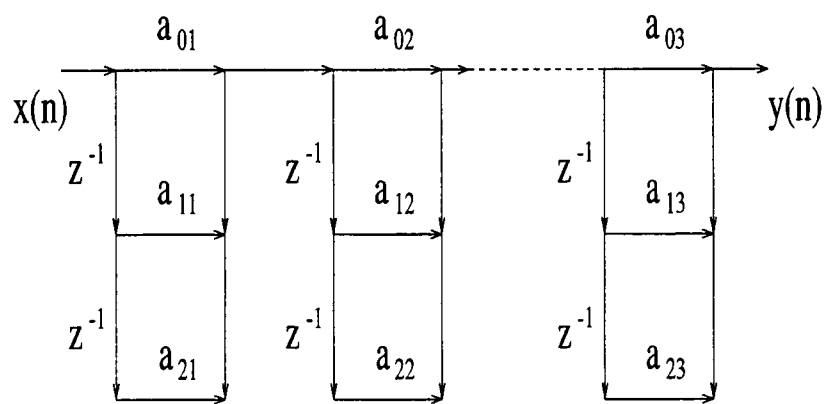


Figure 2.6: Cascade form realization of an FIR system.

2.2.3 FIR filter design

To design a FIR filter, the coefficients a_i in Equation 2.16 must be determined so that the filter satisfies all the imposed conditions. These conditions can be specified in either the time or frequency domain. Over the years, many methods for calculating these coefficients have been developed. Some of the designing techniques are listed as below.

- Design by window technique
- Design by the equiripple technique
- Design by frequency sampling
- Design by iteration
- Least square design

The detail information on these techniques can be found in the book written by Andrew Bateman [11].

2.3 Object-oriented signal processing

Data refining in this research uses technology provided by Data Refining Technologies, Inc. [12], which is available at University of Tennessee at no cost for research purposes. Using this technology, signal processing is based upon interval, rather than point set, mathematics. The advantages of this approach include [12] :

1. a hierarchical signal processing structure with decreasing resolution at higher levels of the hierarchy, yet with a common mathematical structure at all levels of the hierarchy.
2. orders of magnitude reduction in the storage and processing requirements of signals as one moves up the hierarchy, and
3. efficient localization of and access to detailed information from higher level information.

Since interval mathematics rather than point mathematics is used, this technology is referred to as Object Oriented Signal Processing.

Object-oriented signal processing reduces the volume of the raw measurement signal using an interval representation. In the interval representation of a signal, values of a variable are defined to lie within an interval over an interval of time. This interval representation is chosen such that maximum information from the original signal is retained while its volume is kept within reasonable limits. This is accomplished by using the "scrubbing" process where sequences of "similar" data are grouped within the same object and resolution in the changes of signal behavior in independent variable is kept. The manner in which this is done is propriety to Data Refining Technologies, Inc. and has not been disclosed. This, however, is not a limitation on this research project since the modeling method that is developed is independent of the scrubbing process.

The objects derived from this condensation process can be interpreted as rectangles drawn around segments of the signal. These rectangles are characterized by three attributes (height, duration and midpoint) which can be encoded into discrete signal states. With this encoding process, discrete events with attached attributes to describe the properties of signal segments are formed from the sequence of signal objects. Figures 2.7 and 2.8 show a typical process measurement signal together with its condensed form.

The mathematical basis for the scrubber's process is the theory of interval algebra. An interval algebra is composed of an interval scale, with set union and complement operators. Interval scales are used to discretize the real number into a finite set of intervals. They also allow the representation of elements of this set by integers. An interval scale with the representation of the basis intervals is shown in Figure 2.9

Triangular sets, the sets of all intervals which are members of the interval algebra, also play an important role in object oriented signal processing. Most computations in object oriented signal processing are formulated as optimizations over the elements of these sets. Furthermore, the progress of an optimization methods can also be interpreted as a set of paths through triangular sets. Design of efficient methods on object oriented signals is equivalent to design of methods optimizing problems with minimum path length. Operations upon object oriented signals can then be solved by optimizations over subsets of a triangular set which satisfy specified relations.

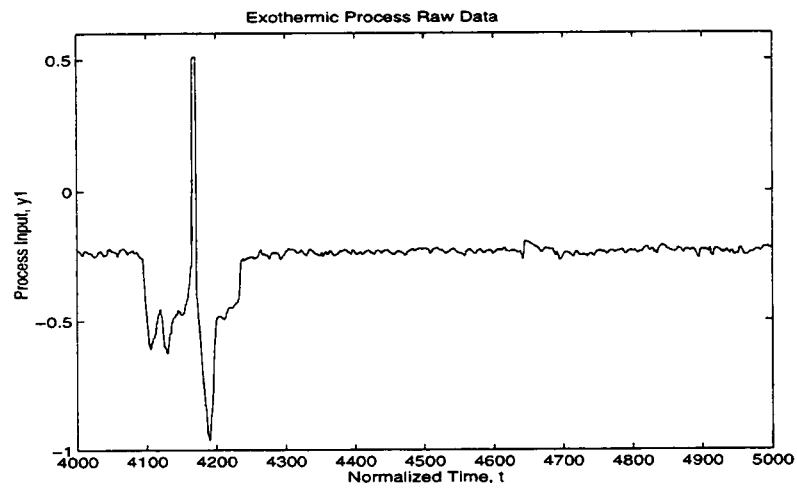


Figure 2.7: A representative measurement signal.

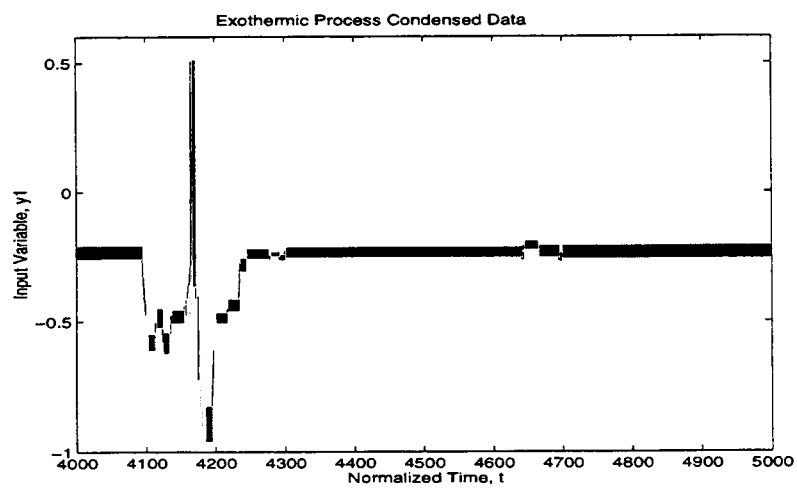


Figure 2.8: Condensed form of signal in the figure above.

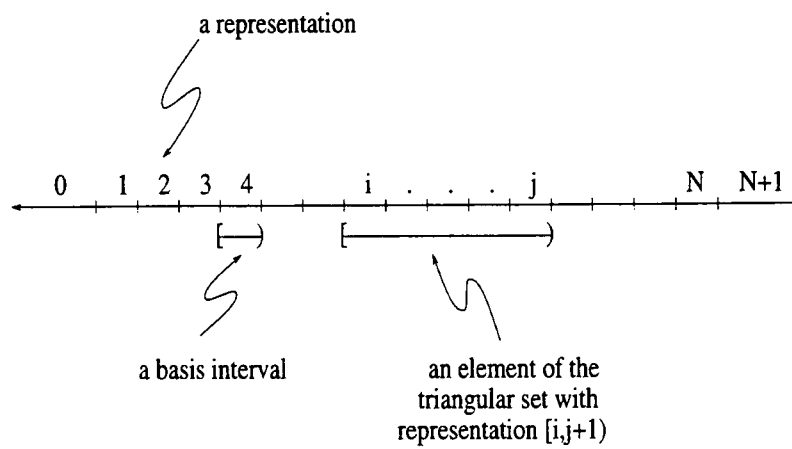


Figure 2.9: An interval scale and interval representations.

Object sequences are derived automatically from the characteristics of the measurement data. Therefore this technology can be grouped into the category of “intelligent signal processing” or “intelligent control”, with automata, fuzzy sets, rule based expert systems, and linguistic processing. The condensed signal can be understood as a discrete event system [15], with the capability of accessing the detailed information contained in the raw data. The same approach can be taken in the reverse direction. Process modeling and control strategies implemented within a discrete event, or hybrid systems, framework can be used after the signal is condensed into object-oriented sequences.

2.4 Partition and refinement of time interval

In the data condensation process, the time interval $[a, b]$ of each signal is partitioned by a set of points $P = \{x_0, x_1, \dots, x_n\}$ where each interval $[x_{j-1}, x_j]$ is the duration of the j^{th} object of the condensed signal. The theorems of partition and refinement in calculus can be applied to $P = \{x_0, x_1, \dots, x_n\}$ for the purpose of process modeling. The definitions of partition and refinement [13, 14] are stated as below.

Let $[a, b]$ be a closed bounded interval. A partition of the interval $[a, b]$ is a set of points $P = \{x_0, x_1, \dots, x_n\}$ such that

$$a = x_0 < x_1 < \dots < x_n = b.$$

The interval $[a, b]$ can be thought of as being divided by the points x_k into n subintervals

$$[x_0, x_1], [x_1, x_2], \dots, [x_{n-1}, x_n]$$

which may or may not have the same length. The norm of a partition $P = \{x_0, x_1, \dots, x_n\}$ is the number

$$\|P\| = \max_{1 \leq j \leq n} |x_j - x_{j-1}|.$$

The partition and norm of an interval $[a, b]$ are shown in Figure 2.10

A refinement of a partition $P = \{x_0, x_1, \dots, x_n\}$ is a partition Q of $[a, b]$ which satisfies $Q \supseteq P$. That is the partition Q is finer than the partition of P . Therefore by definition, if P and Q are partitions of $[a, b]$, then $P \cup Q$ is finer than both P and Q . And if Q is a refinement of P , then $\|Q\| \leq \|P\|$.

The definitions of partition and refinement will be used in the research to process the object-oriented signals before model development of the process is done.

2.5 Discrete event systems

In this section, we define discrete event, discrete event system, a sequence of discrete events and discrete event model [15, 16]. These definitions will be used when each rectangle in the object-oriented signal is encoded into discrete event based on the attributes.

Discrete events refer to the logical or symbolic, rather than numerical, states of a process. A discrete event system is a system whose states are discrete events. That is, in a discrete event system, the states have logical or symbolic values which change with the occurrence of events.

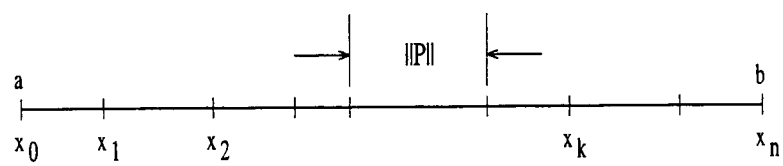


Figure 2.10: Partition and norm of $[a, b]$.

For example, the state of a manufacturing machine may be described by sentences such as `start_processing`, `finish_processing`, `machine_failed`, etc. For this type of system, the state changes only at a discrete set of points in time.

A sequence of discrete events refer to the recording of certain discrete events' occurrence. It is sometimes called trace [16]. Sequences of discrete events can be described by models called discrete event models [16]. Hence, by definition, we can say that a discrete event model is a mathematical model for representing the set of discrete event sequences that a discrete event system can generate.

2.6 Summary

The development of FIRM model uses the theories from two fields, Markov chain modeling methods and FIR filter principles. This chapter presented the background material on those areas. Due to the large volume of raw measurement data, data condensation is necessary. The data condensation technology using object-oriented signal processing was discussed. The calculus definitions of time partition and refinement were then presented. The discussion of discrete event systems completed the background material that is necessary to develop FIRM model.

CHAPTER 3

FIRM Modeling

This chapter discusses in detail the design of a FIRM model from a set of input-output signals. The condensation of raw measurement data to object-oriented signals is described in Section 1. In Section 2, the refinement of signals' time partitions is explored. Section 3 demonstrates the state representation of the object-oriented signal. Finally, the derivation of the probability transition matrix P and the simulation diagram of the model are presented in Section 4 and Section 5 of this chapter.

3.1 Condensation of raw measurement data

Raw measurements consisting of input and output signals of an industrial process are used in this research to generate the dynamic predictive hybrid model. The raw data are obtained from an operating exothermic process whose identity is masked due to the restrictions imposed by the manufacturer of the process equipment. The data are collected from the process variables at the sampling rate of 30 seconds. The measurements of these variables are recorded for the period of 2 days. These measurements, which contain the relationship among the process variables, are condensed before the process modeling step.

The data refining technology used in this research reduces the volume of raw data

measurements by one or more orders of magnitude while retaining its process dynamic characteristics for the purpose of process modeling. Using the software furnished by Data Refining Technologies, Inc. [12], a higher level interval based signal is produced. Each signal is masked to produce a sequence of signal objects which can be interpreted as rectangles drawn around segments of the signal. These rectangles are characterized by three attributes (height, duration and midpoint), which can be encoded into discrete signal states. Figures 3.1 through 3.4 show the raw and condensed input and output signals used in this research to develop the FIRM model.

The main advantage of this data refining technology is the time resolution of the raw data source, whereas the disadvantage is the variable duration of signal objects which substantially complicates the modeling and formulation of control problems.

3.2 Refinement of input and output time partitions

Although the data condensation method provides a substantial reduction in the quantity of data, it results in an event-oriented signal composed of objects with variable duration. Each object in the signal stream has an associated random time duration which does not overlap the durations of other objects within the same stream. Therefore, each input and output signal has its individual time partition which is random in nature and different from other signals. Before the condensed signals can be modeled, time partitions of the input and output signals have to be refined to produce a common time partition for both input and output signals.

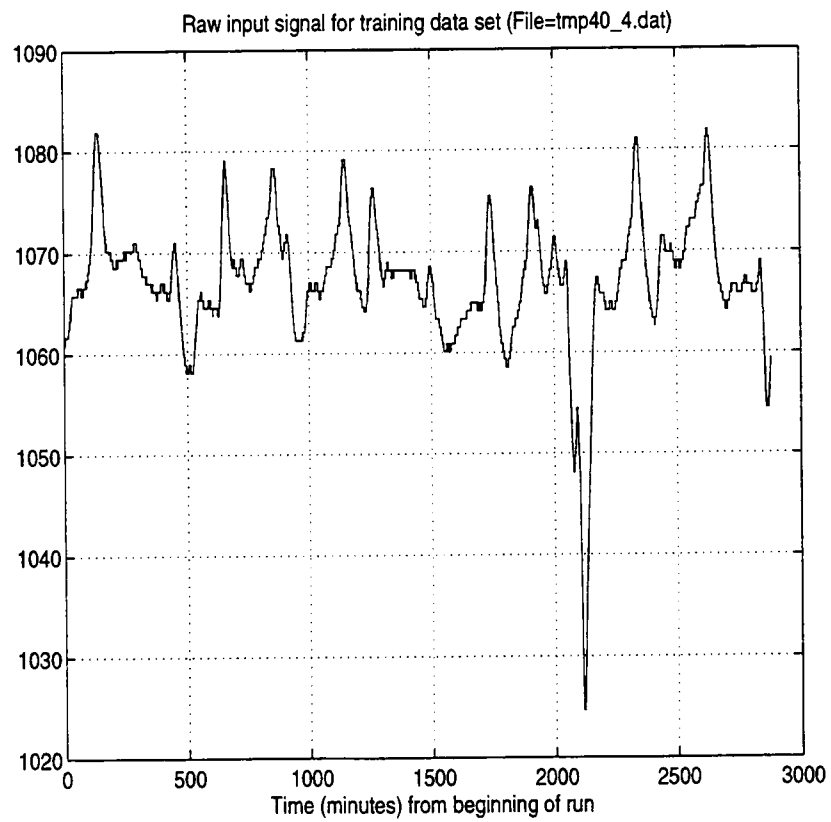


Figure 3.1: Input signal of the training data set.

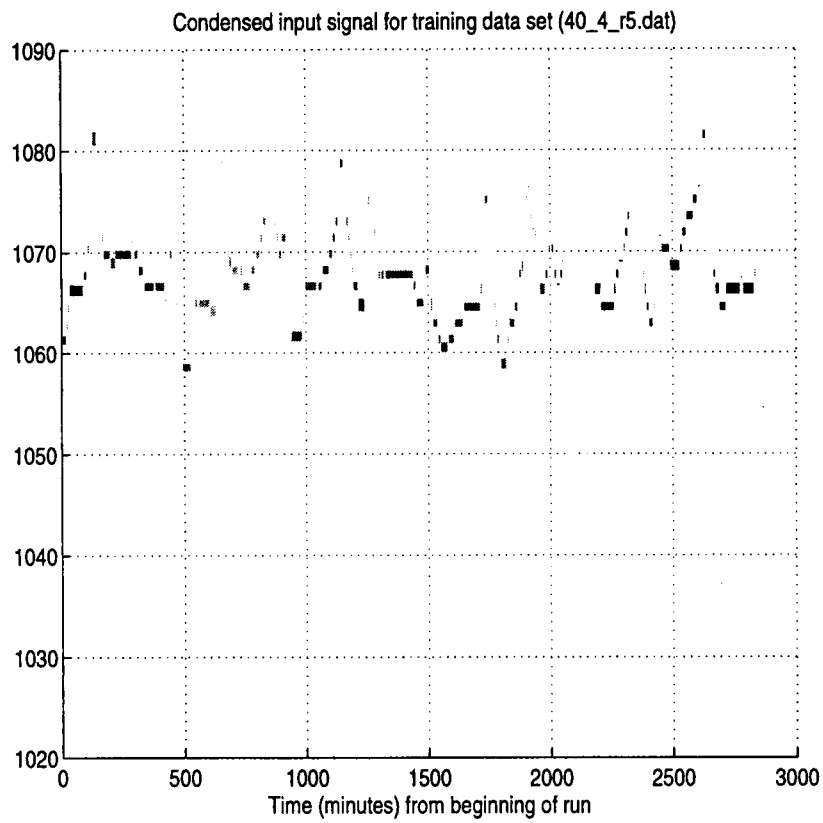


Figure 3.2: Condensed input signal of the training data set.

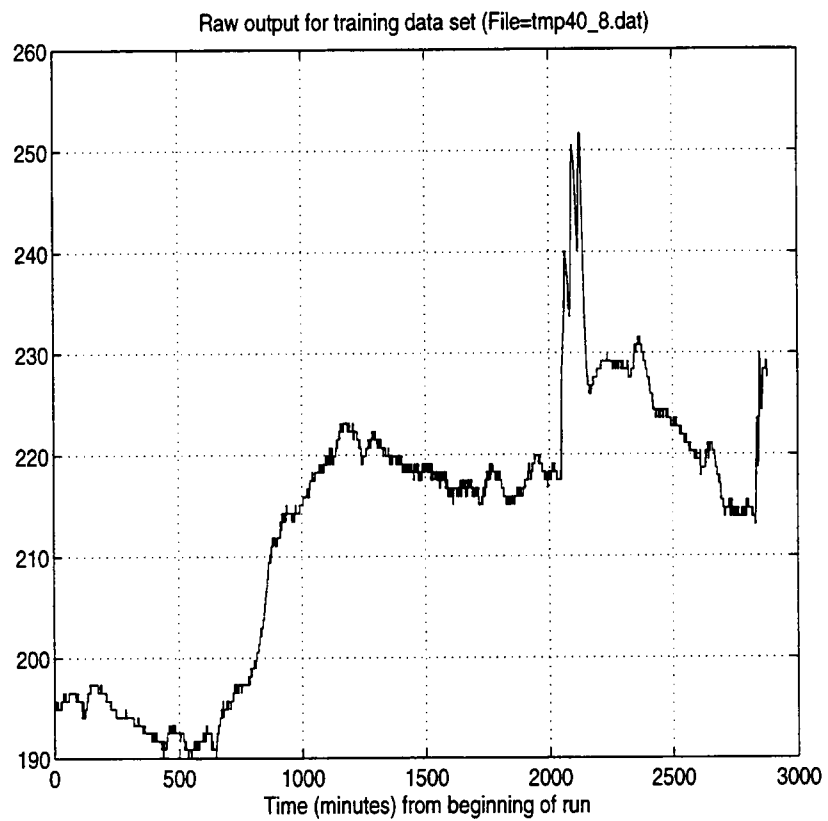


Figure 3.3: Output signal of the training data set.

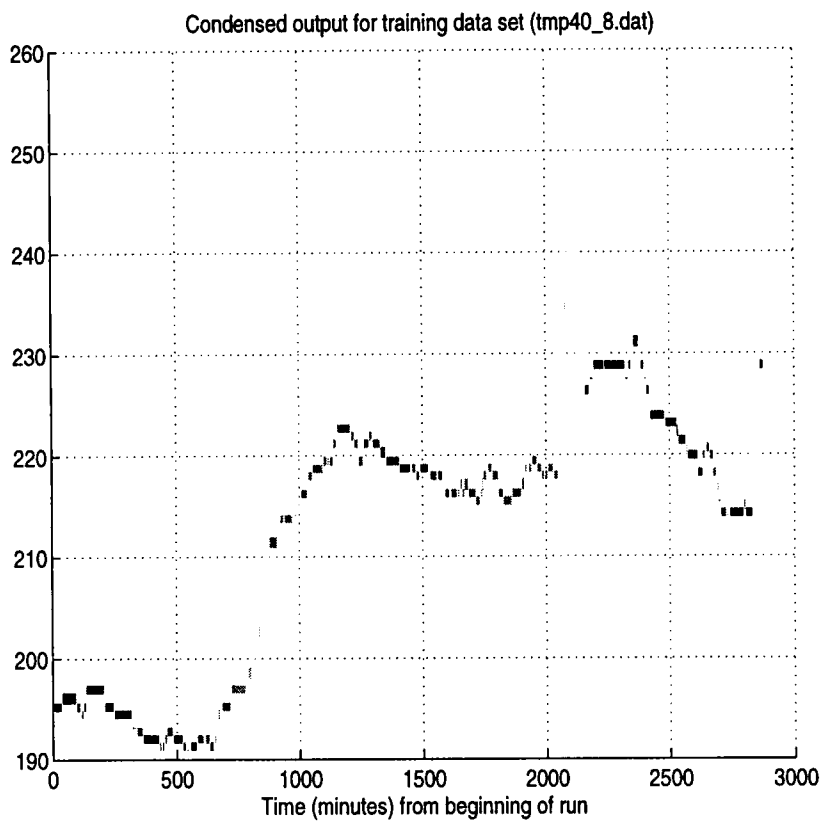


Figure 3.4: Condensed output signal of the training data set.

A time partition, T , is associated with each object-oriented signal produced from the data refining process. The time partition [14] of the signal is a set of points $T = \{t_0, t_1, \dots, t_n\}$ such that

$$a = t_0 < t_1 < \dots < t_n = b.$$

where a and b are the starting and ending time of the signal. Each interval $[t_{j-1}, t_j)$ is the duration of the j^{th} object of the condensed signal. Thus, the points t_i and t_j are the endpoints of the j^{th} object of the condensed signal. An example of an object-oriented signal and the corresponding time partition of the signal is shown in Figure 3.5.

Each object-oriented signal obtained from the data refining process has its own unique time partition, in which case, a set of input and output signals will have two different sets of time partitions, one from each signal. To derive the transition probability relating each set of input and output objects, the durations for the input and output objects must be the same. This requires that both signals have the same partition. The task of obtaining the probability for the transition from each object in the input signal to a single object in the output signal becomes impossible with the different time partitions between the two signals. This difference in time partition of input and output signals is illustrated in Figures 3.6 and 3.7.

A common time partition for both signals must be obtained by relocating the endpoints of each object in both signals in order for each input object to be mapped onto a single output object. This time partition, called refinement [13], is finer than the time partitions of both the input and output signals.

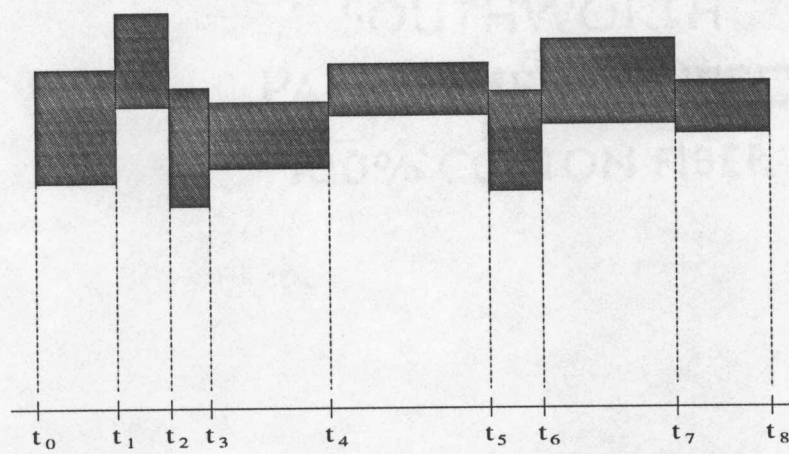


Figure 3.5: An object-oriented signal and its time partition.

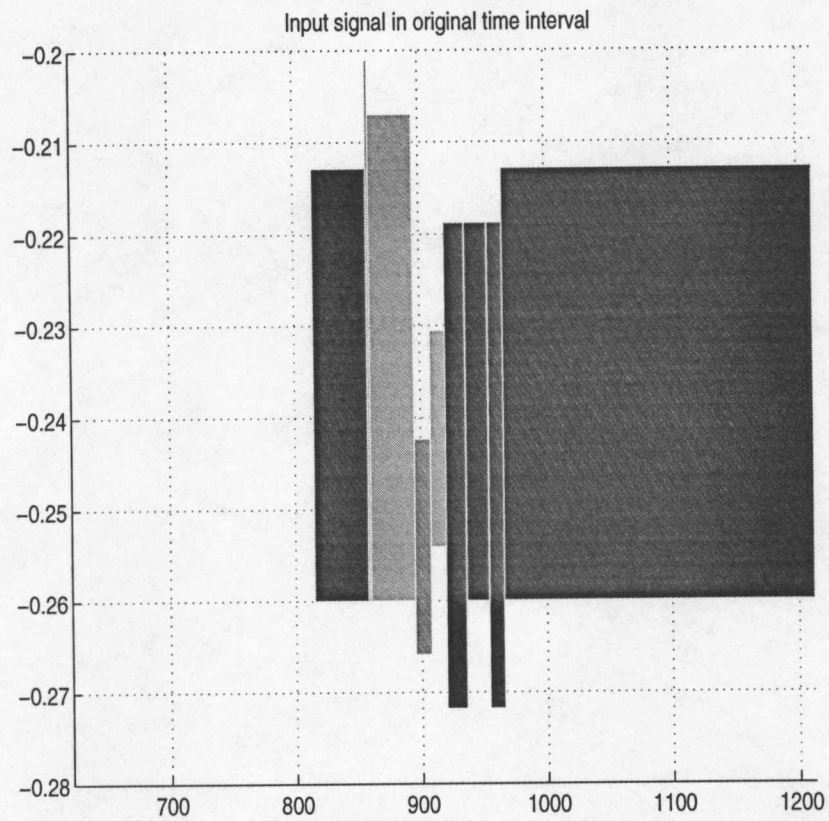


Figure 3.6: A segment of the input object-oriented signal.

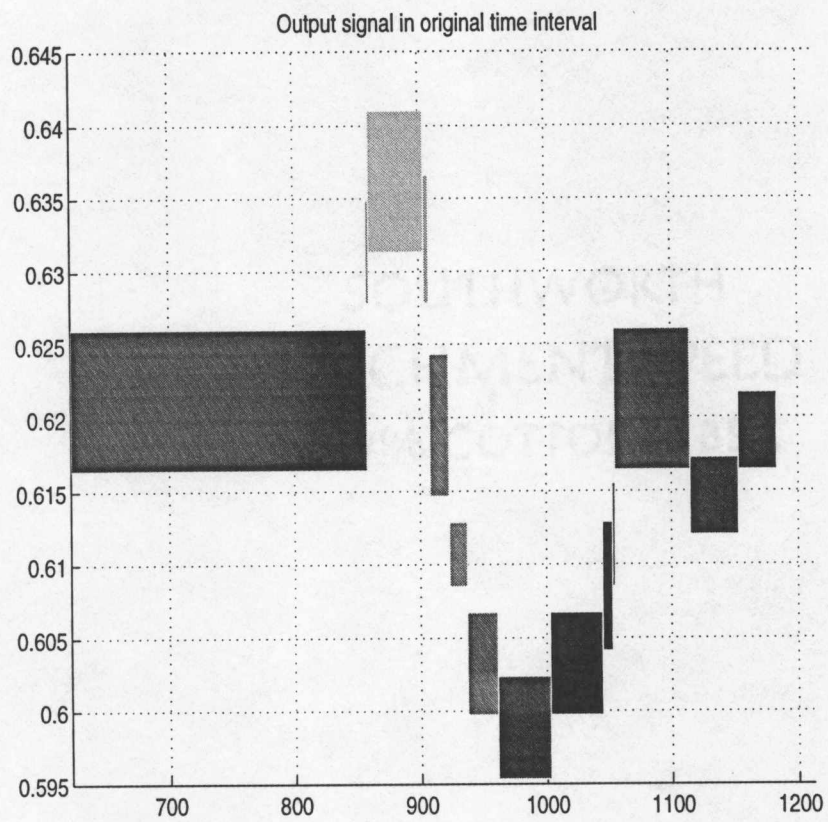


Figure 3.7: A segment of the output object-oriented signal.

There are an infinite number of refinements that can be obtained from the time partitions of a set of input and output signals. However, the refinement with the smallest set of endpoints can be derived from the equation

$$R = T \cup S \quad (3.1)$$

where R is the refinement and T and U are the time partitions of input and output signals respectively. Equation 3.1 is illustrated by Figure 3.8.

By using Equation 3.1, a refinement of the time partitions of both input and output signals is produced. The Matlab [18] function for obtaining the refinement, R , is included in Appendix A. The refinement that resulted from the use of this Matlab function is adopted as the common time partition for both input and output signals. For the segment of signals shown in Figures 3.6 and 3.7, the input and output signals after the refinement process is shown in Figures 3.9 and 3.10.

From the figure, it is noticable that some of the input and output objects with large durations are partitioned into objects with smaller durations after undergoing the refinement process. Since both the input and output objects in the new sets have the same time duration, each object in the input signal can now be associated with the corresponding output object that possesses the same time interval. If the j^{th} input object is the current input of the system, the current output of the system will be the j^{th} output object.

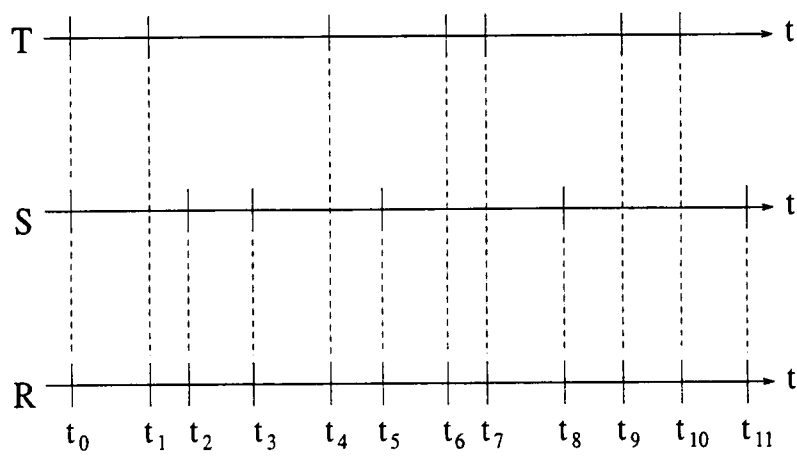


Figure 3.8: Refinement using $T \cup S$.

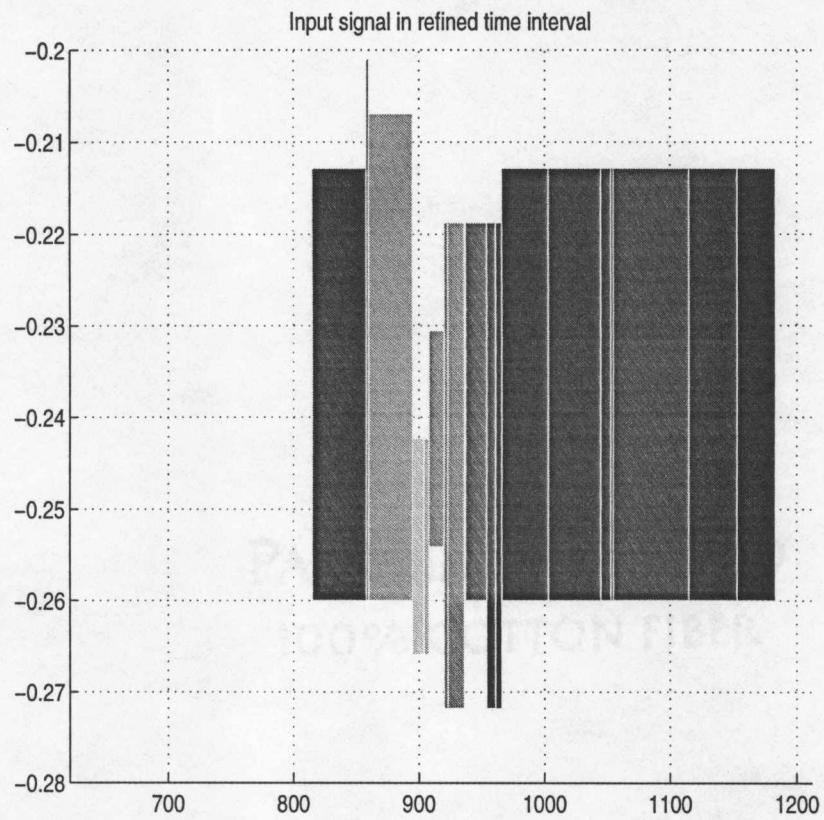


Figure 3.9: The segment of input object-oriented signal after refinement.

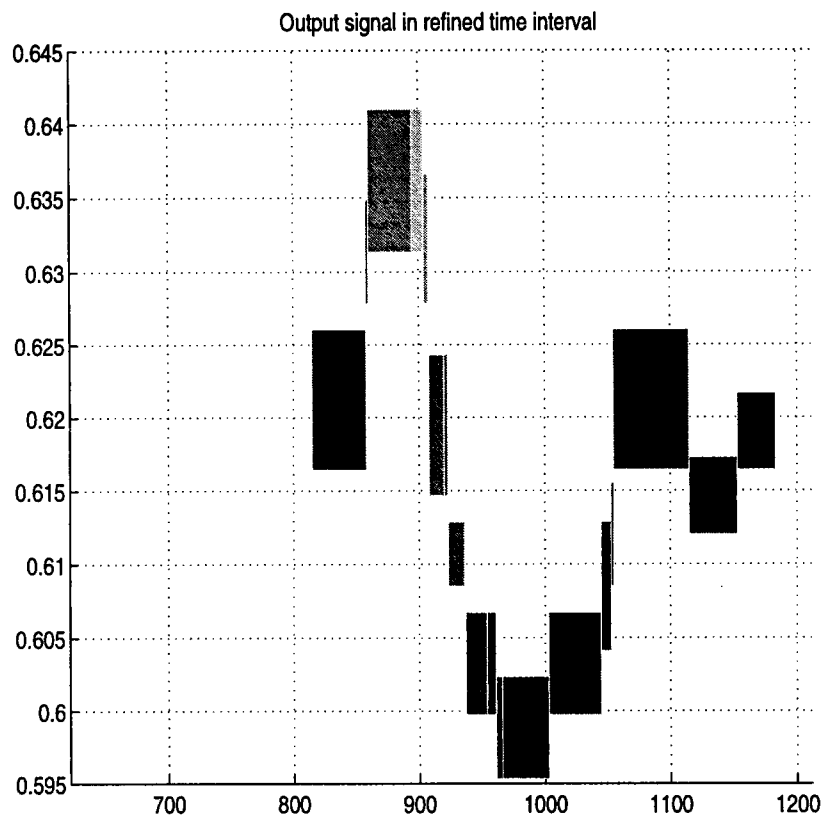


Figure 3.10: The segment of output object-oriented signal after refinement.

3.3 State representation of object-oriented signal

Each object in the refined signal is characterized by three attributes : height, duration and midpoint. The objects in the signal have to be encoded into object states and system states based on these three attributes before the probability transition matrix of the model can be derived from the set of refined input and output object-oriented signals. This section discusses the algorithm for obtaining object states and system states from the objects.

3.3.1 Discretization and indexing of object height, duration and midpoint

Since the values of height, duration and midpoint of the objects are random, the number of possible heights can range from 1, for signals with all objects of the same height, to the total number of the objects in the signal, for signal consisting of objects which are all different in height. This also holds true for the duration and midpoint of the objects. If each object is to be represented by an object state which is a number obtained based on the objects' three attributes, the number of possible states is as large as $N_1 \times N_2 \times N_3$ where N_1, N_2 and N_3 are the number of possible heights, durations and midpoints in the signal. For the input signals shown in Figure 3.2, this value amounts to $301 * 239 * 278 = 19999042$. This number is not only large but also uncontrollable since N_1, N_2 and N_3 depend on the nature of the signal.

In order to control the total number of object states and subsequently reduce it, discretization of the ranges of height, duration and midpoint is performed. The ranges

of these attributes are the intervals from the minimum to the maximum values given by

$$r_h = [\min(\text{height}), \max(\text{height})]$$

$$r_d = [\min(\text{duration}), \max(\text{duration})]$$

$$r_m = [\min(\text{midpoint}), \max(\text{midpoint})]$$

where r_h, r_d and r_m are the ranges of height, duration and midpoint respectively. The ranges r_h, r_d and r_m are partitioned into N_h, N_d and N_m numbers of equal length intervals. If the height of an object lies in the k^{th} interval of the N_h intervals, then the height index i_h of the object is assigned the value k . Similar indexing is done for the duration and midpoint of all the objects.

By indexing the objects with i_h, i_d and i_m , each object is now represented by these three indexes. Depending on the choice of N_h , objects of almost the same height can be grouped under the same index. Similarly, the objects possessing roughly the same duration and midpoint (depending on N_d and N_m) are grouped under the same duration and midpoint index. The advantage of utilizing these indexes to characterize the object is that the total number of possible object states can now be controlled. This is made possible by selecting the values for N_h, N_d and N_m since the total number of possible object states is given by $N_h \times N_d \times N_m$. Higher values of N_h, N_d and N_m may be used to achieve a better model of the process. However, the model created will be larger and more complicated. This will be explained in the next chapter together with the effect of N_h, N_d and N_m on the size and complexity of the probability transition

matrix, $\underline{P}$.

3.3.2 Object state representation

As stated above, each object characterized by the height, duration and midpoint can be represented by a number which is called an object state. This object state will now be a function of the three indexes, i_h, i_d and i_m . The object state, u_k , for each object is defined using the equation

$$u_k = i_h + N_h \times i_d + N_h \times N_d \times i_m \quad (3.2)$$

This equation is an extension of the binary encoding method for variables with different magnitude. Equation 3.2 is illustrated in Table 3.1 where all the possible combinations of i_h, i_d and i_m are listed.

Notice that the object states range from 0 to $(N_h - 1) \times (N_d - 1) \times (N_m - 1)$ in the table above. Using Equation 3.2, each input object is encoded as an input object state which is an element of the set $\{0, 1, \dots, N_h N_d N_m - 1\}$. Since many of these states do not occur in the data, they may be omitted from the model. Re-encoding the states, element from the set $0, 1, \dots, k$ of non-negative integers can be associated with each value of u_k which is associated with one or more data objects. This set is used to form the states of the Markov chain. A similar object state representation is done for the output signal resulting in a set of output object state $\{y_0, y_1, \dots, y_{y_s-1}\}$. For a sample signal in Figure 3.11, the object states are shown in Figure 3.12. The Matlab function file for converting each object to object state representation is included in Appendix B.

Table 3.1: Possible combinations of i_h, i_d and i_m .

i_h	i_d	i_m
0	0	0
0	0	1
$\vdots$	$\vdots$	$\vdots$
0	0	$N_m - 1$
0	1	0
$\vdots$	$\vdots$	$\vdots$
0	1	$N_m - 1$
$\vdots$	$\vdots$	$\vdots$
0	$N_d - 1$	$N_m - 1$
1	0	0
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$
1	$N_d - 1$	$N_m - 1$
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$
$N_h - 1$	$N_d - 1$	$N_m - 1$

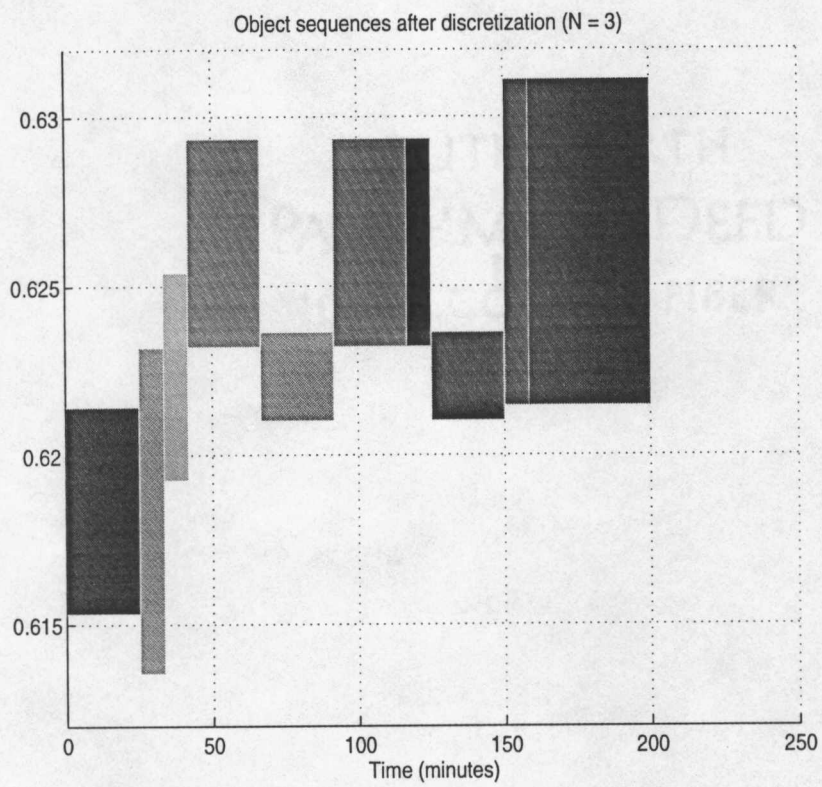


Figure 3.11: Sample signal.

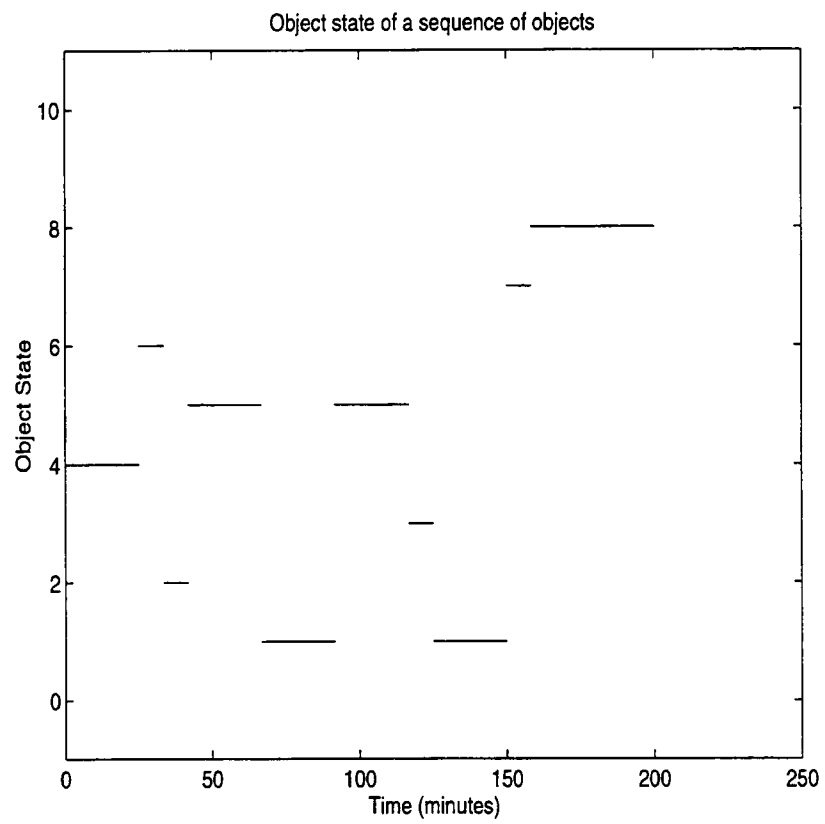


Figure 3.12: Object state of the sample signal.

3.3.3 System state representation

In a FIRM model, the system state of the model is no longer based on a single input object. Unlike the conventional Markov chain, each output object is related not only to the current input object but also to $(n - 1)$ past input objects, where n is a positive integer constant. Therefore, the system state in this model is derived from the current input object and the $(n - 1)$ past input objects.

After each object in the input signal is represented by an object state that ranges from 0 to u_s , the values of these object states are used to calculate the system state of the model. The system state is a number representing the present input object and the $(n - 1)$ past input objects. Therefore, each possible combination of n input object states is a possible system state. The following equation is used to encode these n input objects into system state representation.

$$x_k = u_k + u_k \times u_s + \cdots + u_{k-n+1} \times u_s^{n-1} \quad (3.3)$$

In Equation 3.3, u_k refers to the object state of the k^{th} object in the input signal and u_s refers to the total number of possible input object states. Each system state is an element of the set $\{x_1, x_2, \dots, x_{n_s}\}$ where n_s is the number of possible system states that exist in the input signal.

If a higher number of past inputs, n , is used to model the process, there will be more combinations of different n inputs and subsequently, a higher number of possible different system states. A model with larger n lowers the probability of repeating system states, allowing a more accurate prediction of the output. Although a larger

n improves the model of the process, the model developed will be more complex and also larger in size. The effect of n on the size and complexity of $\underline{P}$ will be covered in Chapter 4 and is a primary focus of study in this research.

3.4 Probability transition matrix, P

The dynamic behavior of the process encoded in the system states and the output object states can be modeled as an FIR model. After converting the input signal into system states and the output signal into output object states, these system states and output object states can be used to develop the probability transition matrix, $\underline{P}$, that relates the n past input objects to the output objects.

Like the conventional Markov chain model, this matrix is also developed based on the frequency of each transitions' occurrence. Since the system state of the model incorporates $(n - 1)$ past inputs with the current input, the first $(n - 1)$ system states from the input signal in the modeling process are ignored. This is because when these objects are the current input of the process, there are less than $(n - 1)$ past input objects available, making it impossible to encode these objects into system states. Therefore, modeling of the process will start at the n^{th} object of the input and output signal and proceed to the last object in the signal.

Each input object beginning from the n^{th} object is associated with a system state obtained from Equation 3.3. Each system state is mapped to an output object state obtained from the output object which lies in the same time interval of the current

input object. The probability transition matrix is then developed from this mapping based on the frequency of occurrence of each pair of system state and output object state. For each element of $\underline{P}$, p_{ji} is obtained by dividing the total number of x_i to x_j transitions by the total number of x_i occurrences. For the system state and output object state sequences shown in Figure 3.13, the probability transition matrix $\underline{P}$ is found as below. The number of transitions between each pair of states is

$$\begin{array}{ll} x^1 \longrightarrow y^1 = 1 & x^2 \longrightarrow y^1 = 2 \\ x^1 \longrightarrow y^2 = 3 & x^2 \longrightarrow y^2 = 2 \end{array}$$

The total number of occurrences for x^1 and x^2 are 4 and 3, respectively. The probabilities for each transition is occurrence are $p_{11} = \frac{1}{4}$, $p_{21} = \frac{3}{4}$, $p_{12} = \frac{2}{3}$ and $p_{22} = \frac{1}{3}$. Therefore the probability transition matrix is given by

$$\underline{P} = \begin{bmatrix} \frac{1}{4} & \frac{2}{3} \\ \frac{3}{4} & \frac{1}{3} \end{bmatrix}$$

The Matlab function for calculating $\underline{P}$ from a set of input and output signals is included in Appendix C.

3.4.1 Properties of the probability transition matrix

The probability transition matrix, $\underline{P}$, in the FIRM model exhibits most of the properties of the transition matrix of that of a conventional Markov chain [1]. The transition matrix, $\underline{P}$, is a $y_s \times n_s$ matrix where y_s and n_s are the total number of possible output object states and system states respectively. Each column of $\underline{P}$ contains non-negative

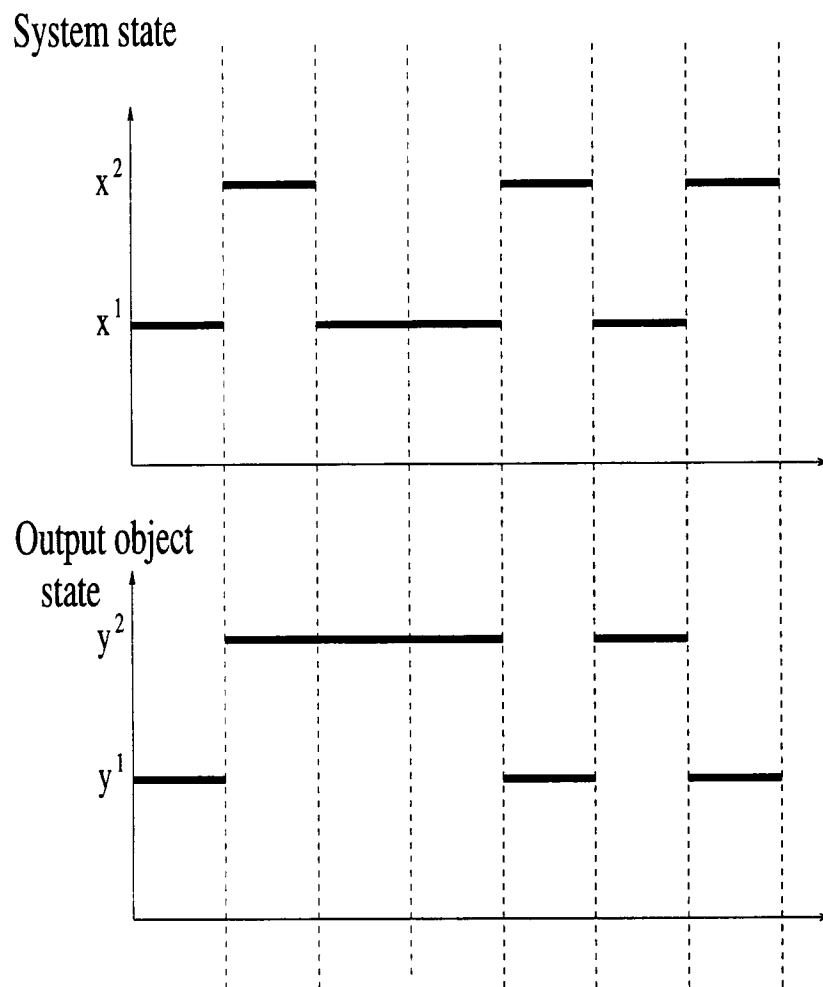


Figure 3.13: Sequences of system states and output object states.

elements which sum to 1. Also, each element, p_{ji} , is the probability of the system output object state to be y_j given that the system states is x_i .

In addition to the similarity of the improved version and conventional Markov chain mentioned above, most of the properties of Markov chain presented in Chapter 2 can also be seen in this model.

3.5 Simulation diagram of the model

The Markov chain model can be utilized in such a way as to predict the future behavior of a stream of signal objects based upon a set of known input states. If the process occupies the system state x_i at the current time interval, the probability of the process occupying each of the y_s states at the next time interval is described by the column vector.

$$p_i = [p_{1i} \ p_{2i} \ \dots \ p_{ni}]^T.$$

The state dynamics can then be written as

$$\begin{bmatrix} p(y_1) \\ \vdots \\ p(y_{y_s}) \end{bmatrix} = \begin{bmatrix} p_{1,1} & \dots & p_{1,n_s} \\ \vdots & & \\ p_{y_s,1} & \dots & p_{y_s,n_s} \end{bmatrix} * \begin{bmatrix} p(x_1) \\ \vdots \\ p(x_{n_s}) \end{bmatrix} \quad (3.4)$$

$$\underline{p}_y(i+1) = \underline{P} * \underline{p}_x(i) \quad (3.5)$$

Figure 3.14 illustrates the simulation diagram for this model. The input matrix

$$\underline{p}_x(i) = [0 \dots 1 \dots 0]^T \quad (3.6)$$

is a column vector where the position of 1 indicates certain knowledge of the FIR models' state. The output probability vector $\left[\underline{p}_y(i+1) \right]$ is the probability of the next process output object assuming each symbolic value. If the current state is not precisely known, as in the attempt to predict across multiple output objects, the model's input vector is then a probability density over the set of possible input sequences.

3.6 Summary

This chapter has covered the algorithms for developing FIRM model. Raw measurement data was first condensed into object-oriented signals. Refinement of the input-output time partitions was performed to produce a common time partition for the input and output object-oriented signals. The refined objects were then encoded into object and system states. The methodology for deriving the probability transition matrix, $\underline{P}$, based on the system and output object states, was developed. The next chapter describes the features of FIRM model developed using the algorithms presented in this chapter.

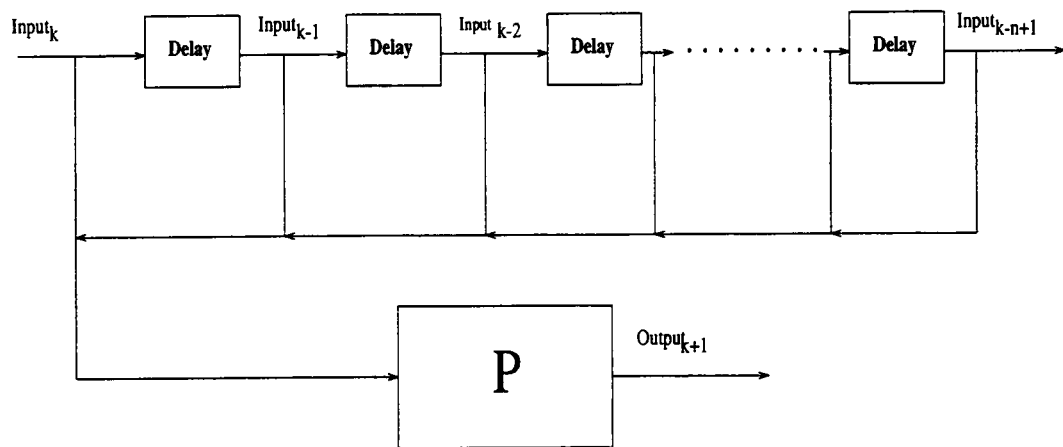


Figure 3.14: Simulation diagram for the FIR model.

CHAPTER 4

Size, Complexity and Prediction Error of Model

In the previous chapter, the new approach to modeling of object-oriented signals using FIR Markov chain (FIRM) was presented. In developing the FIRM model, the controller can vary three parameters of the model to change the model features. The model parameters are the number of discrete values used to represent values in the ranges of height, duration and midpoint of the objects, the number of past inputs used to represent the system state and the size of the training data set used in developing the model. These three parameters are denoted as N , n and s , respectively. In this chapter, the effects of these parameters on the model features are explored.

Three main features of the model are considered in this chapter. They are the size of $\underline{P}$, the complexity of $\underline{P}$ and the model prediction error. The relationships between the model features and the parameters are established graphically using the training data set. These relationships are important as they provide the guidelines for establishing the identification techniques in selecting the appropriate values for the model parameters, which will be discussed in next chapter.

4.1 Size of P

The size of a process model is usually a direct measure of the cost of developing and controlling the process model. Therefore in most of the applications, a smaller sized process model is preferred. But for most of the time, the controllers are forced to surrender some features of the model such as the accuracy of model in representing the process or predicting the output when the size of the model is reduced.

In the FIRM model, the size of the model is measured by the size of the probability transition matrix, $\underline{P}$. The size of $\underline{P}$ is determined by two numbers, the number of rows and the number of columns of the matrix. The number of rows of $\underline{P}$ reflects the total number of possible output object states in the process whereas the number of columns reflects the total number of possible system states present in the process. The number of rows and columns of $\underline{P}$ will be used as a measure of the size of the model. The effects of the model parameters N, n and s on the number of rows and columns of $\underline{P}$ are presented in the next section.

4.1.1 Relationship between size of P and N

In the FIRM model, the ranges of height, duration and midpoint, r_h, r_d and r_m respectively, are partitioned into N numbers of equal length intervals. The discretization of these ranges can directly affect the size of $\underline{P}$ as the size of $\underline{P}$ depends on the total number of possible system and output object states. If the number of discrete values, N is increased, there will be an increase in the number of equal length intervals in each

range. This in turn increases the number of possible index values for height, duration and midpoint. For example, if N is increased from 5 to 10, the number of possible values for the height index, i_h , will range from 0 to 9. As compared to model with $N = 5$, the model with $N = 10$ will have 5 additional index values, $i_h = 5$ to $i_h = 9$. Similarly, 5 new index values are also added to the existing index values of duration and midpoint if N is increased from 5 to 10 for both attributes. Although this allows better discrimination between objects, increasing the number of index values will result in a larger number of possible input and output objects states.

The total number of output object states determines the number of rows of $\underline{P}$. Therefore, an increase in N which causes a larger number of possible output object states will increase the number of rows of $\underline{P}$. This is depicted in Figure 4.1 where the number of rows of $\underline{P}$ is measured as N is increased while n and s are held constant.

The number of columns of $\underline{P}$ depends on the total number of system states. The calculation for the number of system states is given by Equation 3.3 in Chapter 3. When N is increased, the number of possible input object states, u_k , is increased. The number of possible system states, x_k , will also increase as a result of the increment of u_k . Thus, an increase in N will also increase the number of columns of $\underline{P}$. This effect is illustrated in Figure 4.2 where the number of $\underline{P}$ columns are plotted against N .

From Figures 4.1 and 4.2, it is noticable that the increases of the number of rows and columns of $\underline{P}$ are not monotonic. At certain instances, the number of rows or columns of $\underline{P}$ reduces as N increases. For example, in Figure 4.2, the number of columns in $\underline{P}$ reduces as N increase from 11 to 13. The reason behind this is when the

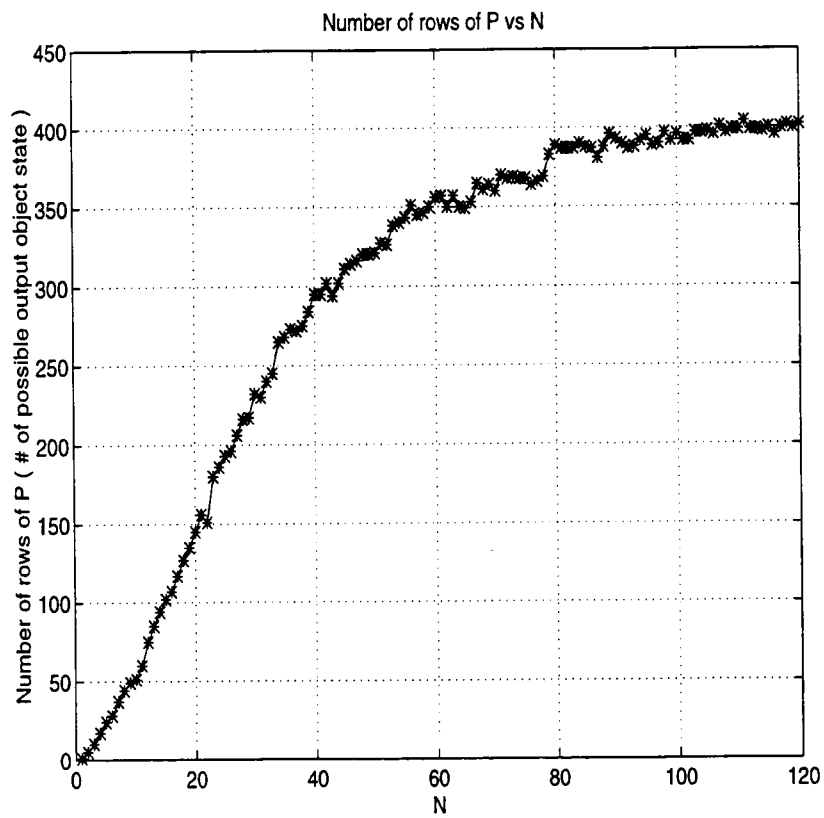


Figure 4.1: Number of rows of P versus N , ($n = 10, s = 492$).

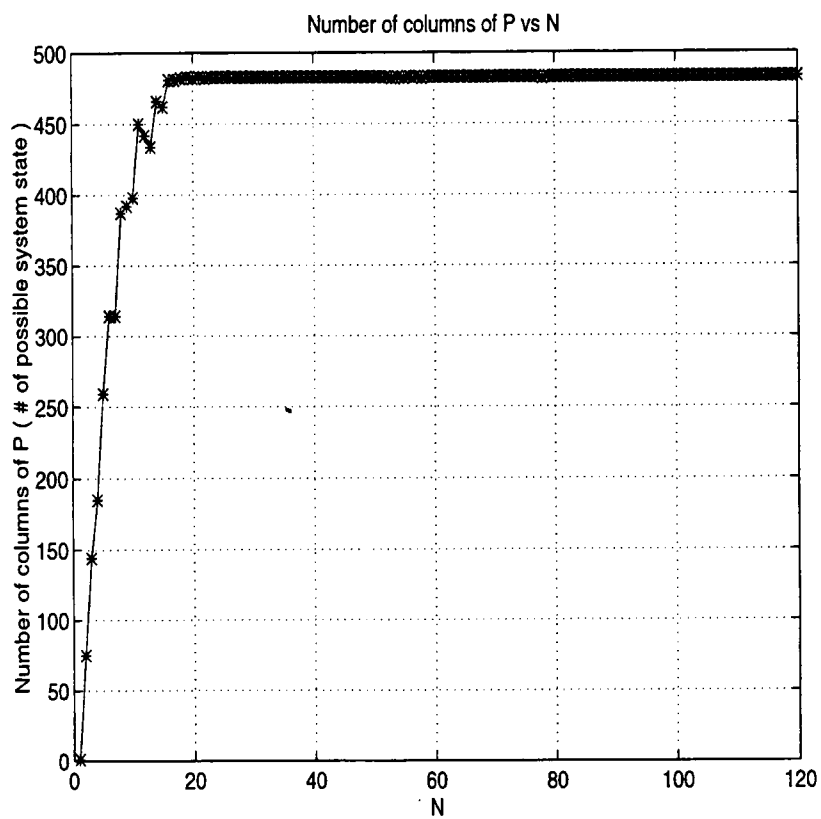


Figure 4.2: Number of columns of P versus N , ($n = 10, s = 492$).

ranges of height, duration and midpoint of input objects are discretized into 13 equal length intervals, some of the objects height, duration or midpoint which are in different intervals when $N = 11$, will be categorized in the same interval. Now that they lie in the same interval, their height, duration or midpoint will be represented by the same index, thereby reducing the number of possible indexes present in the model. This will result in less system states and therefore less columns of $\underline{P}$. A similar property is depicted in Figure 4.1 where instead of system states, less output object states are observed when N is increased to certain number.

Another property that can be seen by studying Figures 4.1 and 4.2 is that the number of rows (respectively, columns) of $\underline{P}$ approaches a constant value, r^* (respectively, c^*) as the value of N increases. Eventually, the number of rows and columns will remain constant at r^* and c^* respectively when N is sufficiently large. This is because when N is large enough, there are enough indexes to uniquely represent the height, duration and midpoint of each distinct object in the training set. That means the partition of r_h, r_d and r_m is fine enough to represent each of the objects. In this case, r^* shows the total number of different objects in the output signal while c^* shows the maximum number of different system states that can be formed from the input object state for the given n and s . This is not necessarily desirable, as the size of the training set can be very large. Rather, the prediction error, evaluated using a validation data set, is more important. This is discussed in Chapter 5. Attaining a value near r^* for an N which does not yield an acceptable error is an indication that the training data set is not sufficiently large or diverse to produce acceptable error levels. If c^* attained, either

the training data set or the number of delays (n) used in the model is not sufficient.

4.1.2 Relationship between size of P and n

In the FIR Markov chain, the system state of the model no longer depends on a single input event. Instead, the system state depends on the current input event and the past $(n - 1)$ input events of the system. As a result, a new model parameter, the number of input events, n used in representing the system state, is created in the FIRM model. Since the system state depends on n , variations of n can directly affect the size of $\underline{P}$.

In the FIRM model, each possible combination of n input object states is a possible system state. By combining the $(n - 1)$ past input with the current input to form the system state, the FIRM model reduces the possibility of system state repetitions and increases the total number of possible system states. For instance, when $n = 1$, the system state is equal to the input object state. Since many of the input object states are repeated, the total number of possible system states is small. As n is increased, more input object states are combined to form system states. This will reduce the possibility of system state repetition and, thus, increase the total number of possible system states.

As n increases, the possible number of system states increases but at a decreasing rate. Each increase in n reduces the number of repeated system states. Eventually, there will not be any repeated system states, and all the objects will be represented by a unique system state. At this point, an increase in n will no longer increase the size of $\underline{P}$.

This is illustrated in Figure 4.3 where the number of columns of $\underline{P}$ is plotted against n is increased while holding N and s constant. In Figure 4.3, the number of columns of $\underline{P}$ decreases as n takes a value larger than 7. In FIR model, the modeling of the process starts at the n^{th} event of the input and output signal. For the first $(n - 1)$ events, there are less than $(n - 1)$ past events that can be used to calculate the system state. Each time n increases, there will be an additional system state is omitted. When n is small, the increase of n will cause a large increase in the total number of possible system states. Although the increase of n causes some of the system states to be omitted, the number of possible system states are far greater than the number of omitted system states. So, there will still be a net increase in the total number of system states. After a certain point, the increase of n will no longer increase the total number of possible system states. During that period, one system state is still omitted each time n is increased. After that point, the number of columns of $\underline{P}$ decreases at the same rate n is increased.

The number of rows of $\underline{P}$ depends only on the number of possible output object states. So, a change in n will not affect the rows of $\underline{P}$ as shown in Figure 4.4 where n is increased, while N and s are held constant.

4.1.3 Relationship between size of P and s

The FIR Markov chain model is developed from a set of input-output signals called the training data set. For most of the time, it is difficult for the controller to decide how large the training data set should be in order for the model to accurately represent the

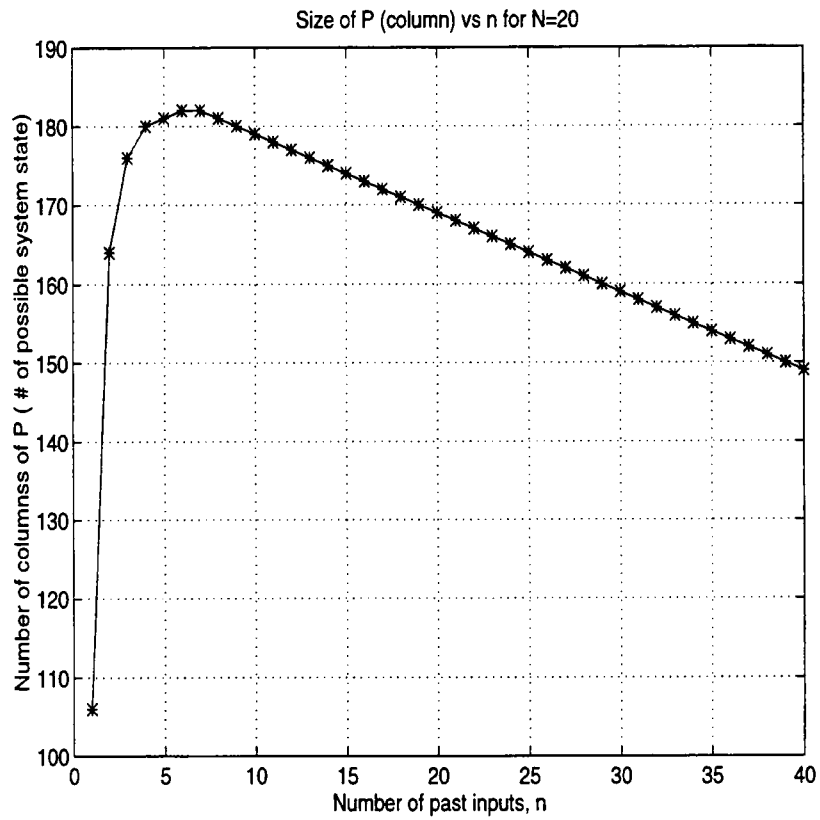


Figure 4.3: Number of columns of P versus n , ($N = 20, s = 188$).

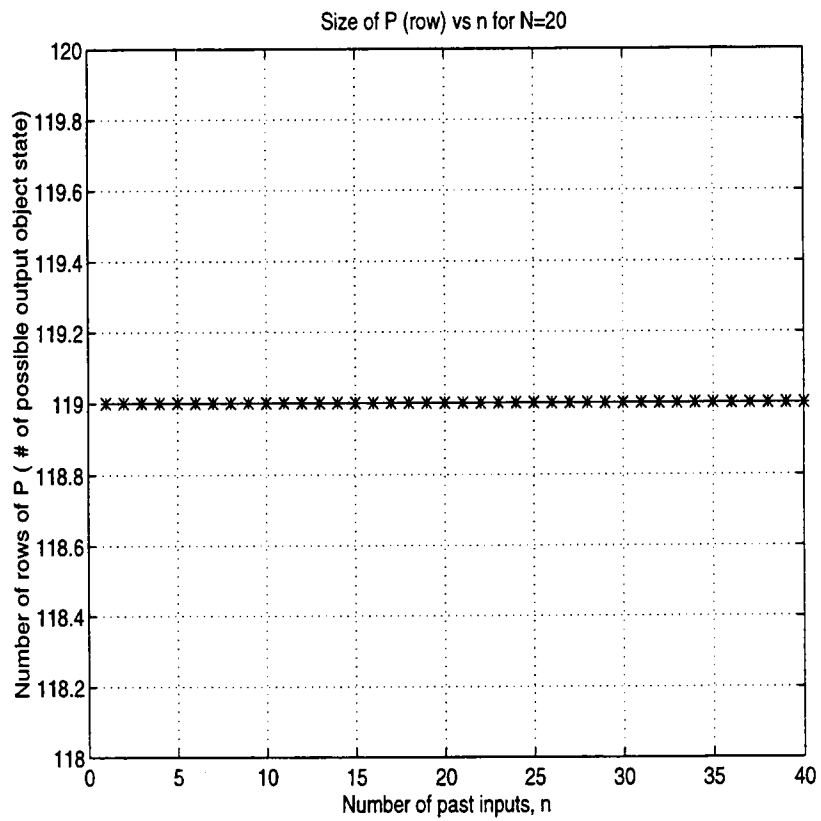


Figure 4.4: Number of rows of P versus n , ($N = 20, s = 188$).

process.

Looking at Figures 4.5 and 4.6, the graphs depicting the size of $\underline{P}$ versus the size of training data set, the size of $\underline{P}$ increases with an increase in size of the training data set. As the size of the training data set increases, more new objects are included, adding to the number of object states and system states. The curve of the size of $\underline{P}$ versus s will eventually level out once all the possible events in the process have been included in the training data set.

4.1.4 Relationship between size of P and (N, n, s)

In the previous section, the plots of the size of $\underline{P}$ versus N, n and s are obtained by changing N, n and s one at a time rather than varying them simultaneously. Therefore these plots do not completely describe the relationship between the size of and the parameters. This one-parameter-at-a-time approach may fail to elicit the combination of that would yield the true global minimum because the interactions between these parameters can not be detected.

To be able to visualize this, only two parameters are considered. A three dimensional surface plot of the number of rows and columns of $\underline{P}$ versus (N, n) with s held constant is shown in Figures 4.7 and 4.8. These figures can only show the interaction of N and n for minimization of the size of $\underline{P}$ over (N, n) graphically.

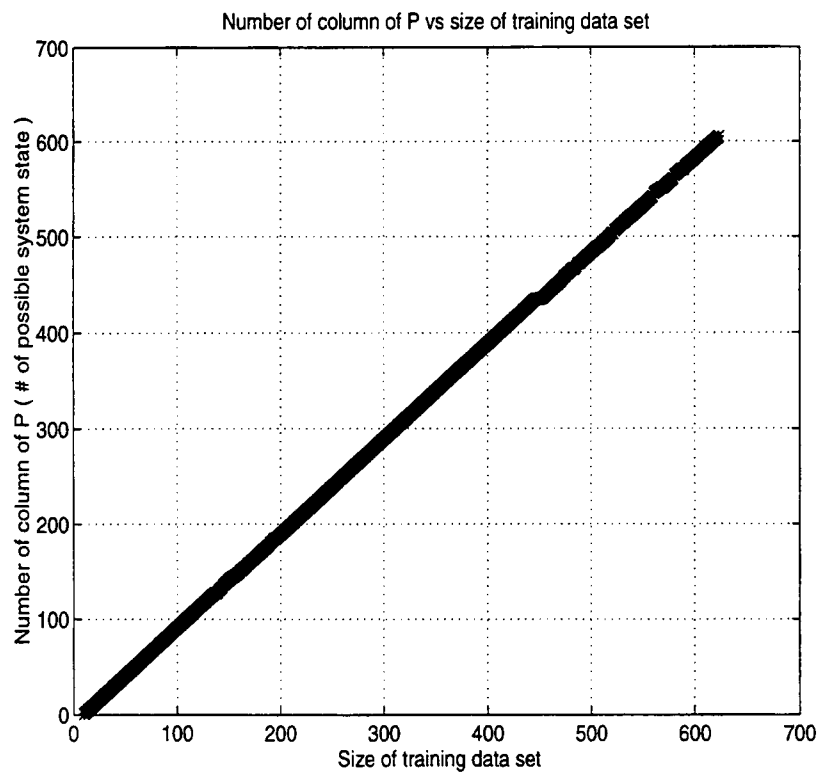


Figure 4.5: Number of columns of P versus s , ($N = 40, n = 10$).

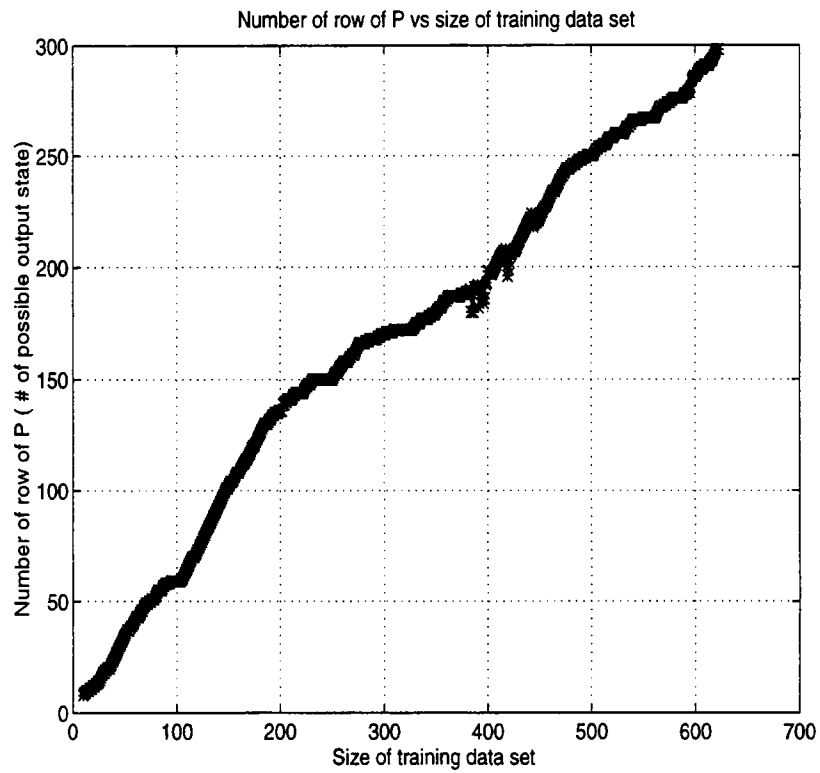


Figure 4.6: Number of rows of P versus s , ($N = 40, n = 10$).

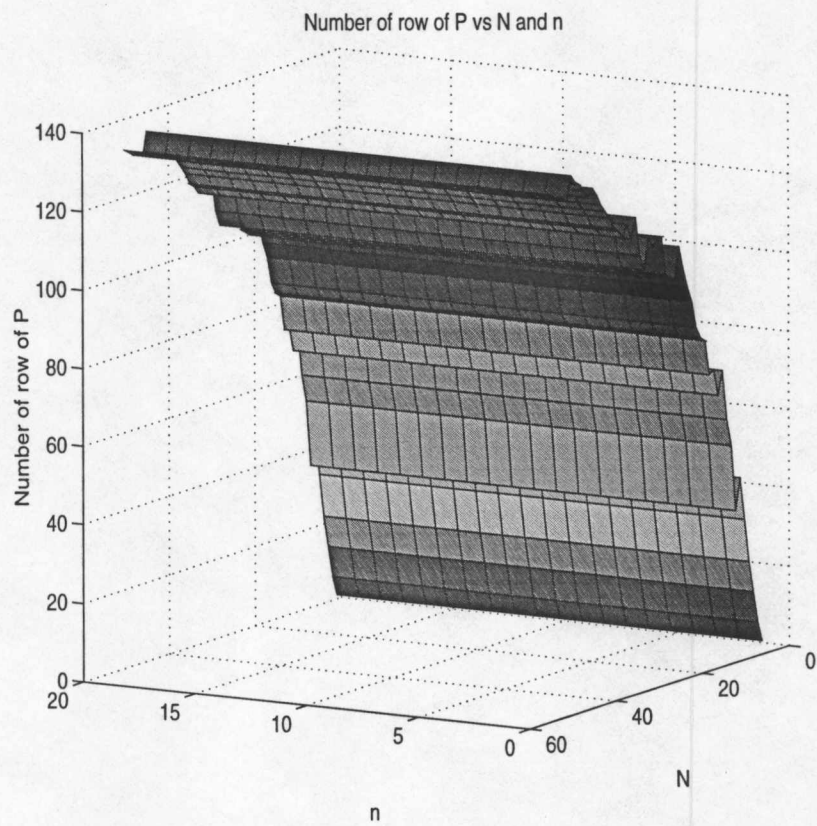


Figure 4.7: Number of rows of P versus (N, n) , ($s = 188$).

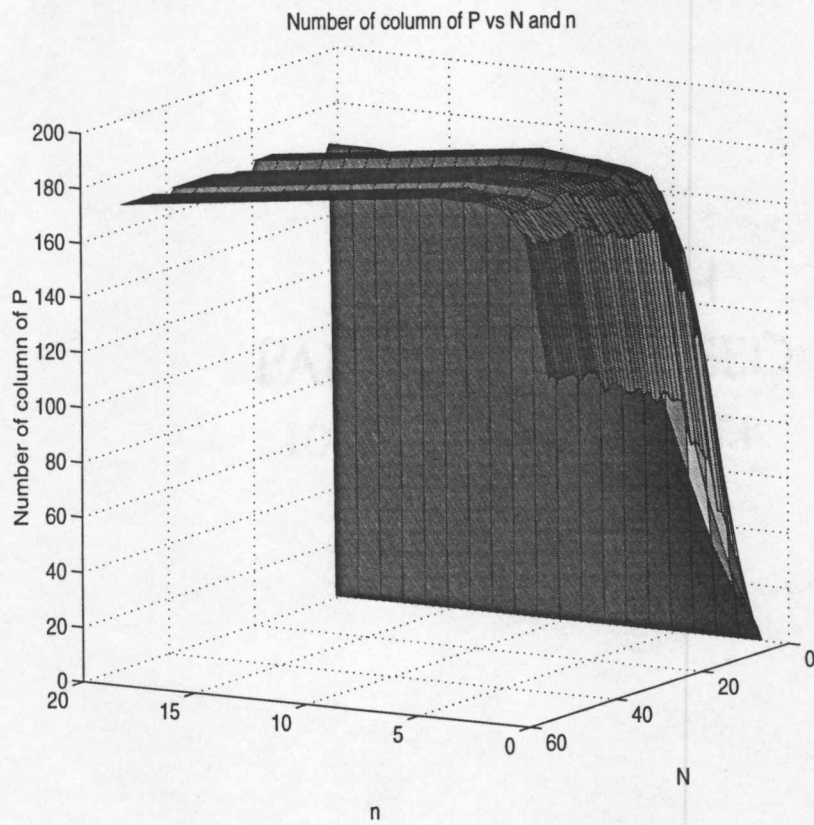


Figure 4.8: Number of columns of P versus (N, n) , ($s = 188$).

4.2 Complexity of P

Besides size, complexity of the model is also a factor influencing the cost of developing and controlling the process model. A more complex process model always requires more effort and time to control and leads to a higher implementation cost. As in the Matlab [18] simulation of the model, a complex process model requires more memory and execution time by computer system. Thus, it is always desirable to reduce the complexity of the developed model if possible.

In the FIRM model, the complexity of the model is measured by the ratio between the number of nonzero entries and the total number of entries in the probability transition matrix $\underline{P}$ as shown in Equation 4.1.

$$\text{Complexity} = \frac{\text{Number of nonzero entries in } \underline{P}}{\text{Total entries in } \underline{P}} \quad (4.1)$$

A typical FIRM model contains many states; however, the transition matrix for the model is sparse. This complexity of the matrix is affected by model parameters, N , n and s . This section explores the effect of the model parameters N , n and s on the model's complexity.

4.2.1 Relationship between complexity of P and N

The complexity of $\underline{P}$ depends on both the nonzero entries and the total entries in $\underline{P}$. Figure 4.9 shows the complexity of the model versus N . It can be seen that as N increases, the complexity of $\underline{P}$ decreases asymptotically to a small value. This is because as N increases, new system states and output object states are introduced into

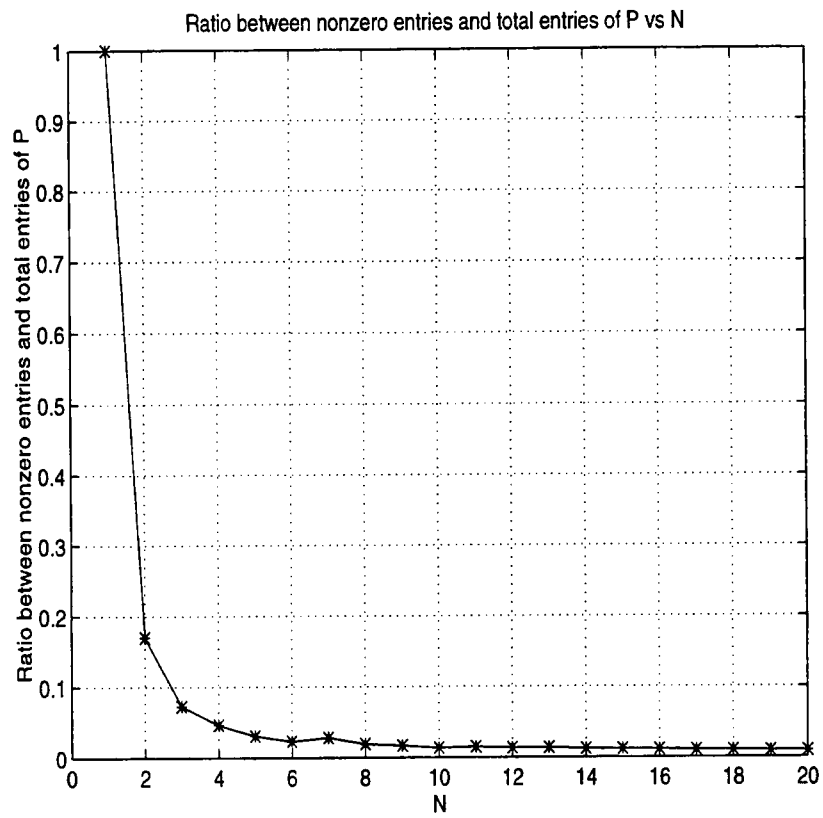


Figure 4.9: Complexity of P versus N , ($n = 10, s = 188$).

the model. Each time a new pair of system and output state is added, at least one nonzero entry will be added to $\underline{P}$. At the same time, both the number of rows and columns of $\underline{P}$ are increased by 1. Therefore the increase in the number of entries in $\underline{P}$ is much larger than the increase in the number of nonzero entries. As a result, the complexity of $\underline{P}$ decreases drastically as N is increased. It is interesting to note that the complexity appears to asymptotically approach a positive value, rather than zero. This may indicate that a minimum (and constant) level of complexity is required to represent the dynamic behavior of a given process. The significance of this is a possible subject for future research.

4.2.2 Relationship between complexity of P and n

Figure 4.10 shows the complexity of the model versus n . It can be seen that as n increases, the complexity of $\underline{P}$ decreases asymptotically to a small value. As n increases, new system states are introduced into the model. Each addition of new system states will introduce a nonzero entry and a new column into $\underline{P}$. Therefore the larger increase in the total number of entries in $\underline{P}$ causes the complexity of $\underline{P}$ to decrease drastically as n is increased.

Notice that when n is increased, the decrease of complexity is not as fast as when N is increased. This is due to the reason that the increase of n can only increase the number of system states, and therefore the number of columns of $\underline{P}$. But the increase of N can cause increases in both the system states and the output object states. Instead of increasing only the number of columns, both the number of rows and columns can

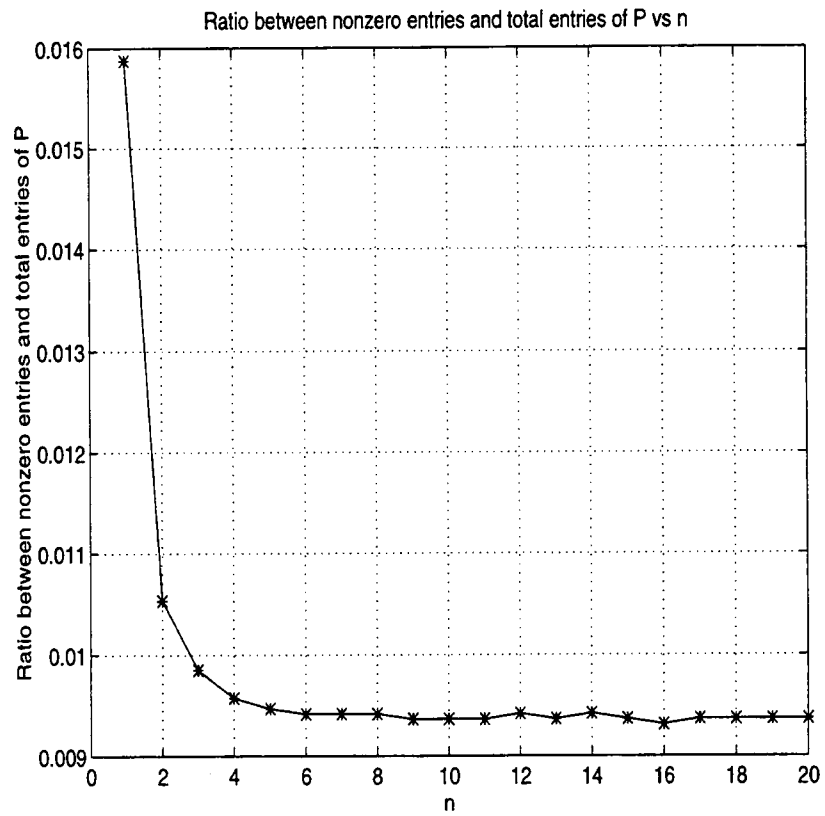


Figure 4.10: Complexity of P versus n , ($N = 20, s = 188$).

be increased. Hence, the increase in the total number of entries will be much larger and the decrease of the complexity of $\underline{P}$ will be much faster.

4.2.3 Relationship between complexity of P and s

As the size of training data set is increased, new objects from the input and output signals are added to the model. Therefore for the same reason as in the case of increasing N , the complexity of $\underline{P}$ decreases asymptotically as s increases (as shown in Figure 4.11).

4.2.4 Relationship between complexity of P and (N, n, s)

As in Section 4.1, the plots of the complexity of $\underline{P}$ versus N, n and s are obtained by changing N, n and s one at a time rather than varying them simultaneously. To be able to visualize this, three dimensional surface plots of the complexity of $\underline{P}$ versus (N, n) are shown in Figures 4.12 and 4.13 at two different view angles while holding s constant at 188.

4.2.5 Significance of this relationship

The observation that model complexity decreases asymptotically to a small value as the parameter values (N, n, s) increases is very important. This indicates that the FIRM method is capable of capturing detailed information about the dynamic behavior of a process without “blowing up” computationally. Since computational requirements remain within reasonable bounds, the models are likely to be useful in prediction and

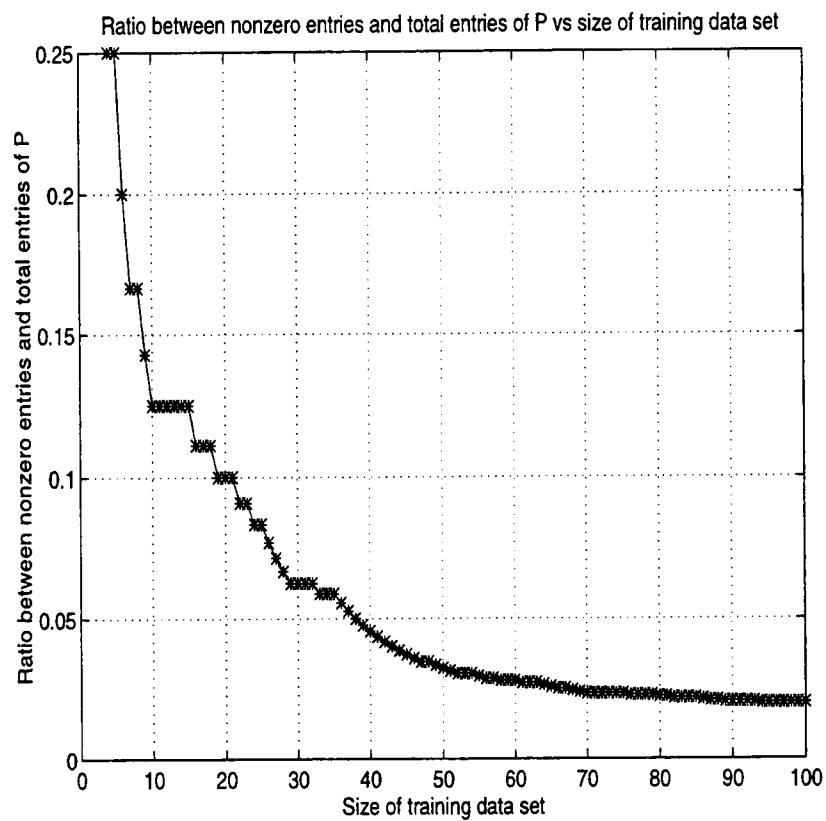


Figure 4.11: Complexity of P versus s , ($N = 20, n = 4$).

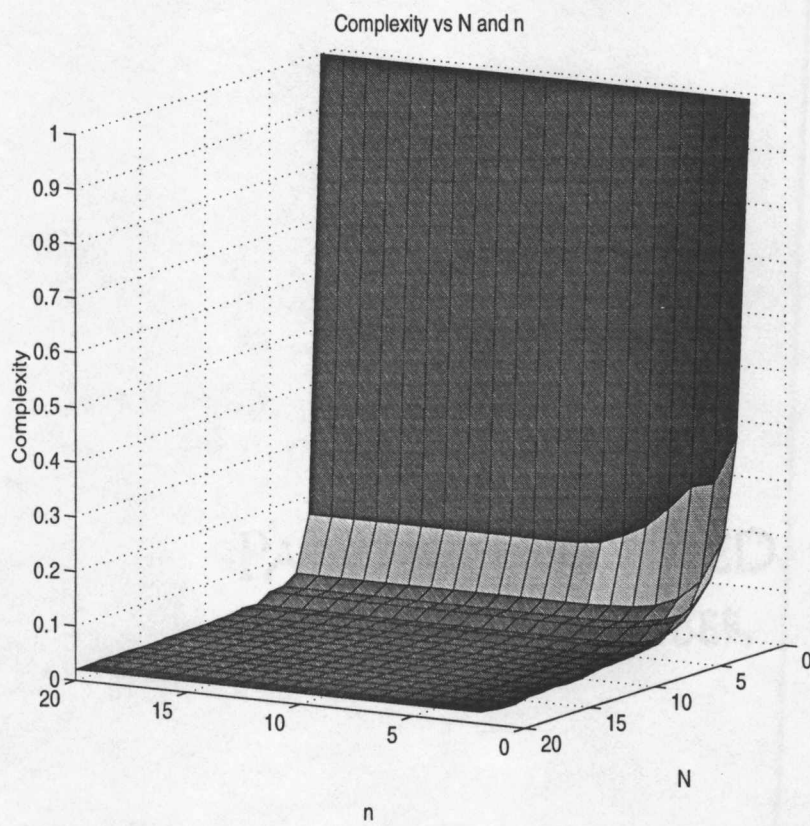


Figure 4.12: Complexity versus (N, n) , $(s = 188)$.

n

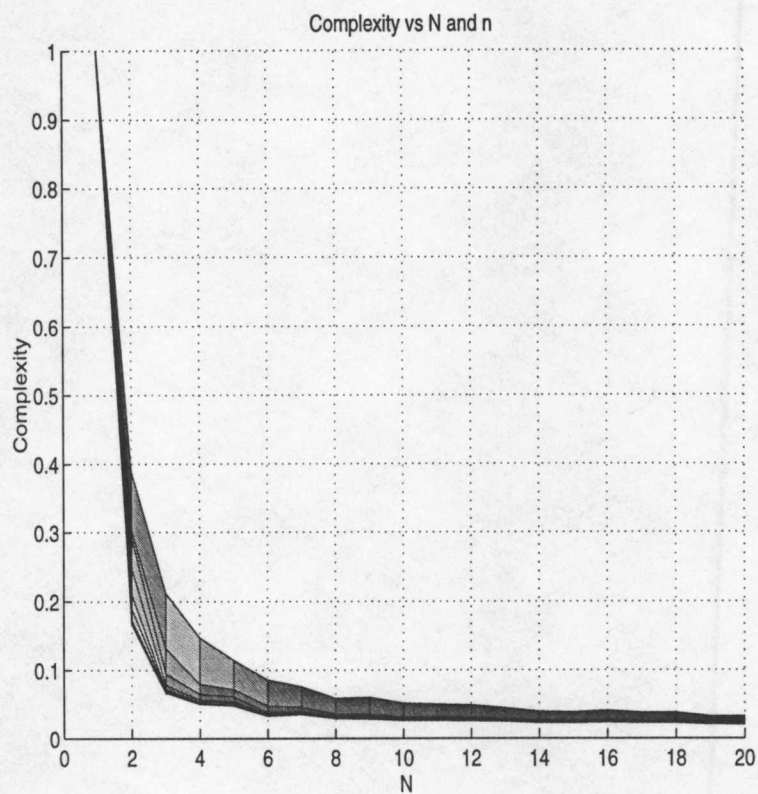


Figure 4.13: Complexity versus (N, n) .

control application especially in light of that a set of computational economical numerical tools exists specifically for sparse matrices. It is therefore reasonable to expect to obtain an acceptable description of a process' dynamic properties with a computationally implementable model.

4.3 Prediction error

In this section, the FIRM model is used to predict the output objects based on the input object signal. The same set of data which is used to develop the model is employed to illustrate the properties of the FIRM and the method used to assess modeling error. The validity of the developed model is evaluated by the accuracy of the model in predicting the future output objects. The relationship between prediction error and the model parameters N, n and s are examined. The next chapter utilizes this method to show how to access modeling error using independent training and validation data sets.

4.3.1 Prediction error at time t

The predicted output of the model is a sequence of objects. The real time output signal is embedded in these objects. Here, a random variable interpretation of the output signal is used to determine the accuracy of the model in predicting the process output. The accuracy of output prediction is measured using the probability assessment of the error which is the difference between the true output and the predicted output.

In the simulation, the predicted error is a measure of the 2-norm error between the true signal and the predicted signal in time series representation. The true output object-oriented signal shown in Figure 4.14 is used to derive the estimated 2-norm error.

According to the figure, the unknown true real time signal, x_{true} , is embedded in the sequence of objects. At each instant of time, there are two associated number, v_t^h and v_t^l which are the upper and lower boundaries of the rectangles (Figure 4.14). Assuming x_{true} is uniformly distributed in v_t^h and v_t^l at each t , the probability density function [17] of x_{true} is given by

$$p(x_{true}) = \begin{cases} \frac{1}{v_t^h - v_t^l} & x_{true} \in [v_t^l, v_t^h] \\ 0 & \text{otherwise} \end{cases} \quad (4.2)$$

The plot of Equation 4.2 is shown in Figure 4.15 Now assume that the predicted output object sequence is as shown in Figure 4.16.

From the figure, the unknown predicted real time signal, x_{pred} , is embedded in the sequence of objects. As for x_{true} , x_{pred} is assumed to be uniformly distributed. Therefore, the probability density function of x_{pred} is given by

$$p(x_{pred}) = \begin{cases} \frac{1}{v_p^h - v_p^l} & x_{pred} \in [v_p^l, v_p^h] \\ 0 & \text{otherwise} \end{cases} \quad (4.3)$$

Equation 4.3 is plotted in Figure 4.17 To find the difference between the two signals, the two sequences of objects are plotted in the same axes as in Figure 4.18.

The figure exhibits two random variables, x_{true} and x_{pred} , as a function of time. Since x_{true} and x_{pred} at each time are assumed to be independent, the joint probability

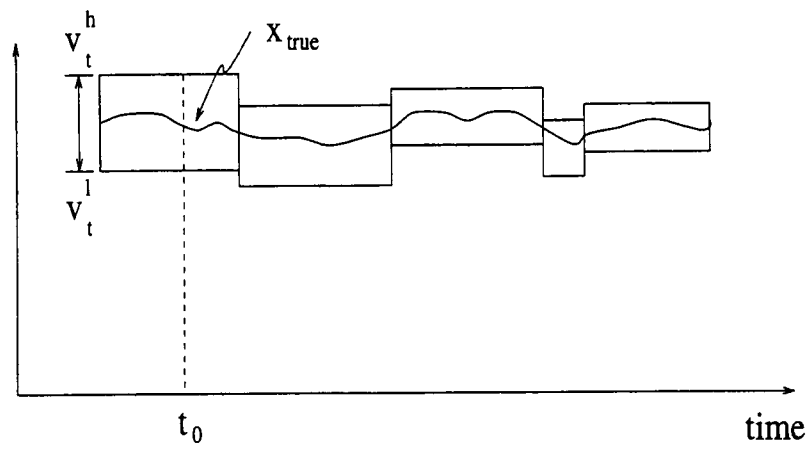


Figure 4.14: An example of true signal objects.

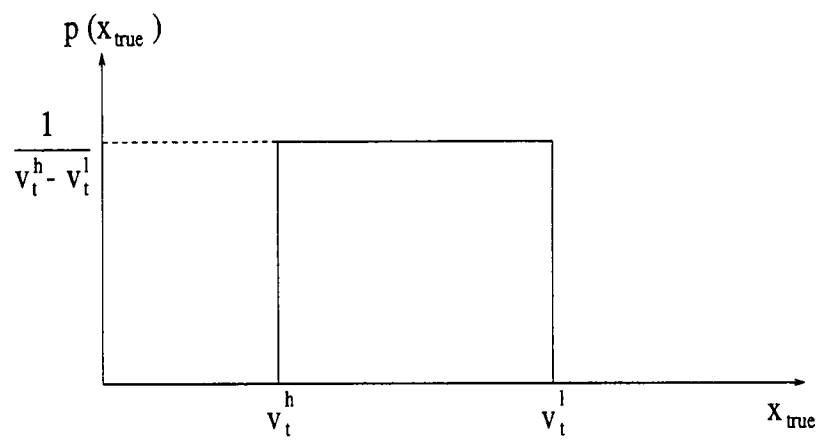


Figure 4.15: Probability density function of x_{true} .

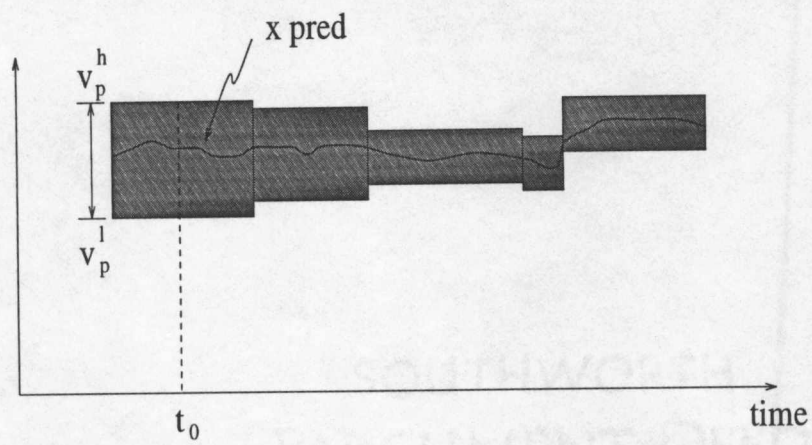


Figure 4.16: An example predicted signal objects.

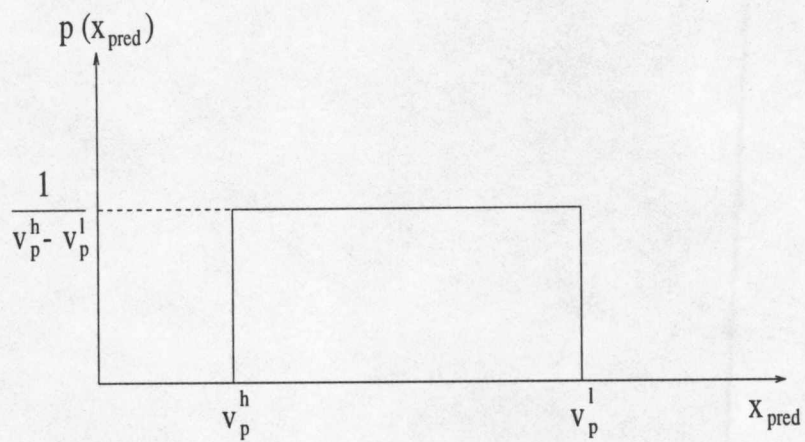


Figure 4.17: Probability density function of x_{pred} .

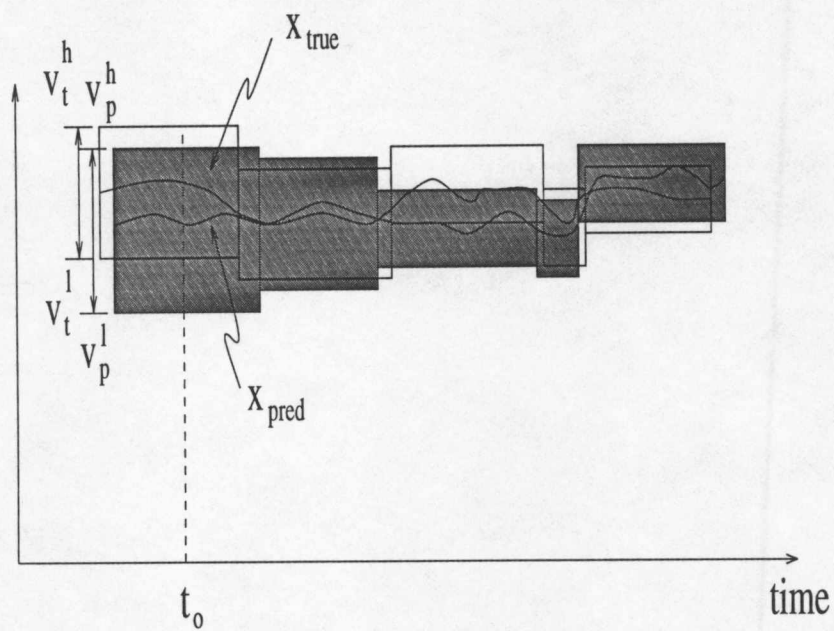


Figure 4.18: Object sequences of true and predicted output.

density function [17], $p(x_{true}, x_{pred})$ is the product of the probability density of x_{true} and x_{pred} . Thus, $p(x_{true}, x_{pred})$ is given by

$$p(x_{true}, x_{pred}) = p(x_{true})p(x_{pred}) = \begin{cases} \frac{1}{v_t^h - v_t^l} \frac{1}{v_p^h - v_p^l} & [v_t^l, v_t^h] \times [v_p^l, v_p^h] \\ 0 & \text{otherwise} \end{cases} \quad (4.4)$$

At any instant t , the 2-norm prediction error is given by $(x_{true} - x_{pred})^2$. Therefore the expected 2-norm error is

$$E[(x_{true} - x_{pred})^2] = \int_{x_{true}} \int_{x_{pred}} (x_{true} - x_{pred})^2 p(x_{true}, x_{pred}) dx_{true} dx_{pred} \quad (4.5)$$

$$= \int_{v_p^l}^{v_p^h} \int_{v_t^l}^{v_t^h} (x_{true} - x_{pred})^2 p(x_{true}, x_{pred}) dx_{true} dx_{pred} \quad (4.6)$$

$$(4.7)$$

The closed form for Equation 4.7 is

$$E[(x_{true} - x_{pred})^2] = \left(\frac{1}{v_t^h - v_t^l} \right) \left(\frac{1}{v_p^h - v_p^l} \right) \left\{ \left(\frac{(v_t^h)^3 - (v_t^l)^3}{3} \right) (v_p^h - v_p^l) + \right. \\ \left. \left(\frac{(v_p^h)^3 - (v_p^l)^3}{3} \right) (v_t^h - v_t^l) - \right. \\ \left. \left(\frac{(v_t^h)^2 - (v_t^l)^2}{2} \right) ((v_p^h)^2 - (v_p^l)^2) \right\} \quad (4.8)$$

Equation 4.8 is used to calculate the expected prediction error of the model at t_0 .

4.3.2 Time averaged prediction error

In FIRM model, no output object is predicted if the system state of the process does not exist in the training data set that is used to develop the model. When there is no prediction, no output object is predicted from the model for the interval $[t_i, t_j)$ where $[t_i, t_j)$ is the time interval of the current input object. Using the calculation of

prediction error discussed in Section 4.3.1, the error caused by no prediction will not be reflected in the estimated prediction error. Percentage of predictable objects given by

$$\% \text{ predictable objects} = \frac{T_{good}}{T_{good} + T_{bad}} \times 100\% \quad (4.9)$$

where T_{good} and T_{bad} are, respectively, the sum of the total intervals over which prediction and no prediction occur, can be used to reflect the no prediction occurrence.

A time averaged prediction error can then be calculated using Equation 4.10.

$$E_{avg} = \frac{1}{T_{good}} \int_t \left(\int_{x_{true}} \int_{x_{pred}} (x_{true} - x_{pred})^2 p(x_{true}, x_{pred}) dx_{pred} dx_{true} \right) dt \quad (4.10)$$

The percentage of predictable objects and time average prediction error will be used in Section 5.4 where simulation of the model is performed using model validation data set.

4.3.3 Relationship between prediction error and N

The number of discretization, N , of the ranges of object height, duration and midpoint is one of the main factors affecting the prediction accuracy. If the number of discretization of the output object height is N , N values from the range of heights will be used to represent the heights of all output objects. Each of the output object's height is represented by one of the N values which is closest in magnitude to the actual height of the object. A similar discretization is done for the output object's duration and midpoint. Since the FIRM model uses these approximated output heights, durations and midpoints, the heights, durations and midpoints of the predicted output objects from

the model are also in term of these values. For example, if $N = 3$ is used in modeling a certain process, there are only three different heights, durations and midpoints (27 distinct objects) that can be found among the predicted output objects, as shown in Figure 4.21

The true and predicted output of the model for various values of N are shown in Figures 4.19 through 4.24. These figures show that higher value of N can make the model's predicted output more accurate, since more values of object height, duration and midpoint can be used to represent the output objects. Therefore the expected 2-norm error will decrease, on the average, when N is increased. This is shown in Figure 4.25 where the estimated 2 norm error is measured as N is increased, holding n and s constant.

From Figure 4.25, it can be seen that the estimated 2-norm error asymptotically approaches a minimum value as N becomes large. The prediction error is constant at e^* when N is sufficiently large. This is because when N is large enough, there are enough numbers to represent each of the distinct output object's height, duration and midpoint uniquely. That is, the partition of ranges of height, duration and midpoint is fine enough to represent each of the objects. In this case, e^* shows the minimum 2-norm error that can be achieved for that particular n and s .

4.3.4 Relationship between prediction error and n

In many application, most of the input states of the process are repeated. But not all the repeated input have the same output. Using a conventional Markov chain to

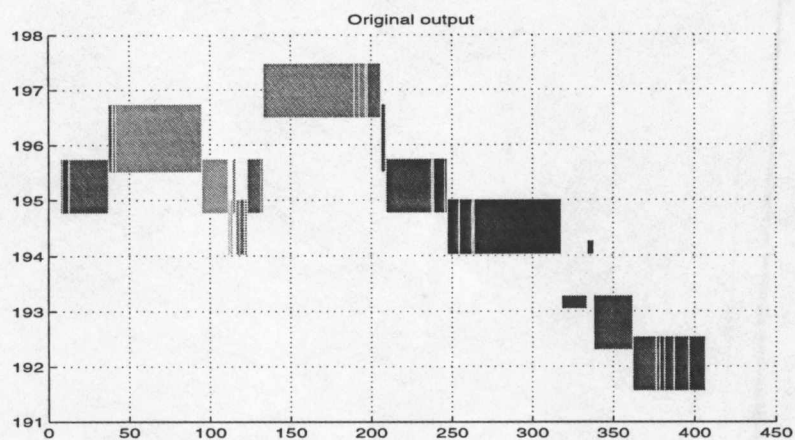


Figure 4.19: True output.

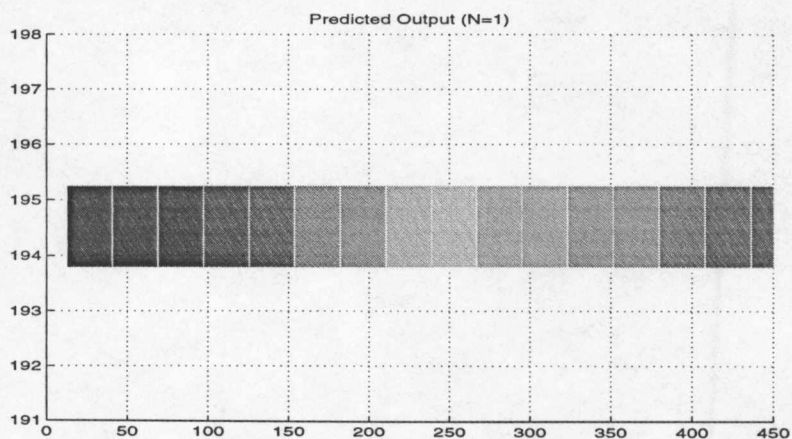


Figure 4.20: Predicted output of model with $N = 1$, ($n = 5$, $s = 95$).

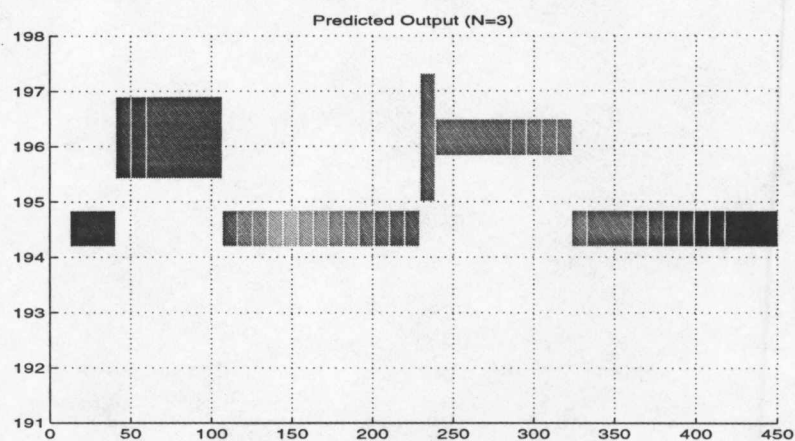


Figure 4.21: Predicted output of model with $N = 3$, ($n = 5, s = 95$).

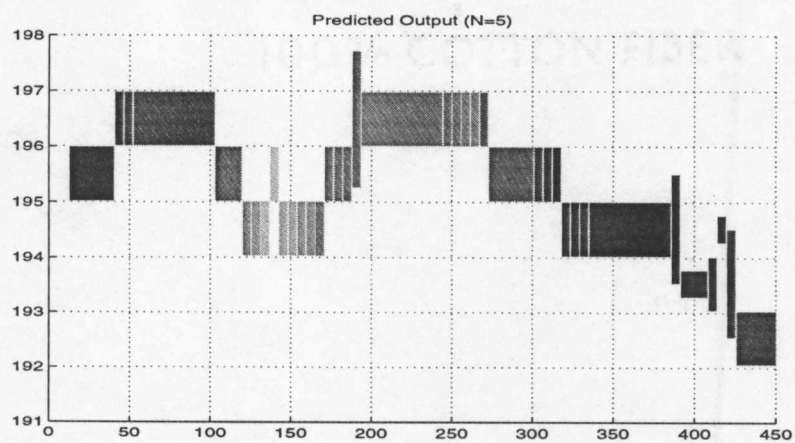


Figure 4.22: Predicted output of model with $N = 5$, ($n = 5, s = 95$).

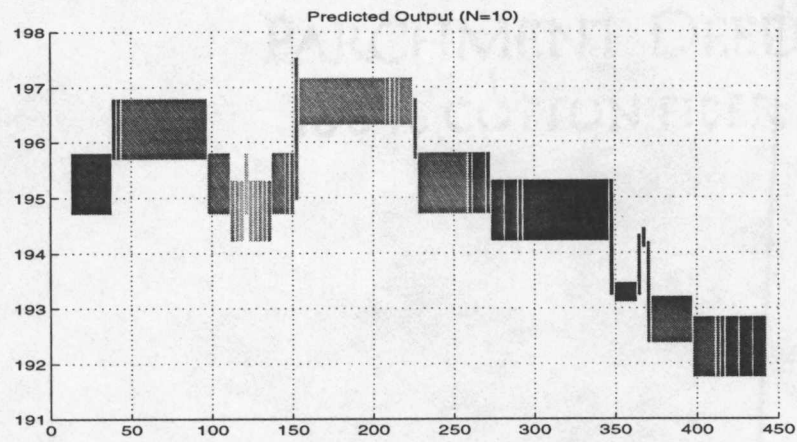


Figure 4.23: Predicted output of model with $N = 10$, ($n = 5, s = 95$).

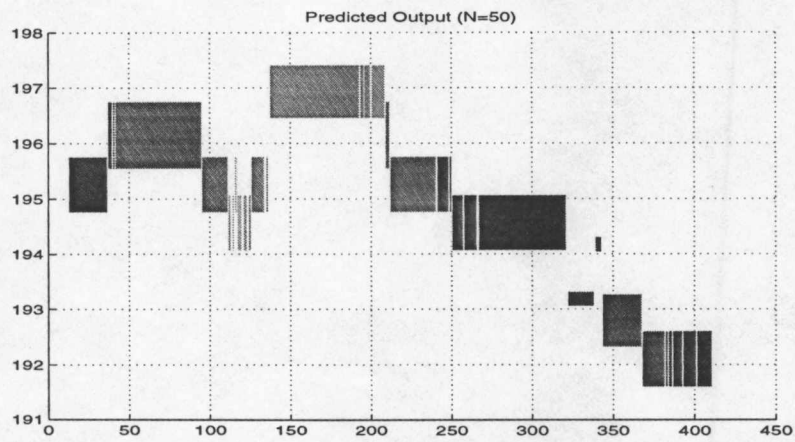


Figure 4.24: Predicted output of model with $N = 50$, ($n = 5, s = 95$).

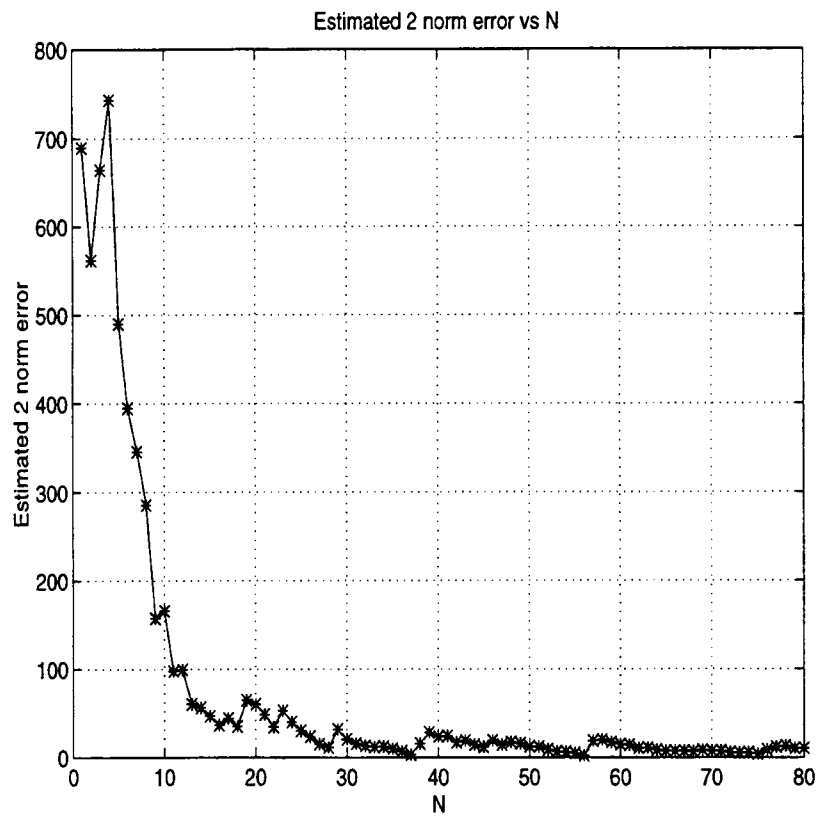


Figure 4.25: Prediction error versus N , ($n = 10, s = 188$).

predict the output of these application will cause a high error since for the same input state, the same output will be predicted.

The FIRM model solve this problem by reducing the possibility of repeated system states. The model reduces the possibility of system state repetitions by combining the system's past $(n - 1)$ inputs with the current input to form the system state. Each time n is increased, the number of repeated system states is reduced. Therefore the prediction error decreases as n increases. This can be seen in Figures 4.26 through 4.30.

When $n = 1$, the FIRM model behaves as a conventional Markov chain. Therefore, Figure 4.27 is the predicted output of a conventional Markov model of the process. The large inaccuracy of the prediction shown in Figure 4.27 suggests that FIRM model is an improved model of conventional Markov model, when n exceeds 1.

Figure 4.31 shows the plot of expected prediction error versus n . From this figure, it can be seen that as n increases, the increase in possible system states becomes smaller. This is due to the reason that each increase of n reduces the number of repeated system states. Eventually there will be no repeated system states and all the objects will be represented by unique system states. After this value of n , the error will no longer decrease.

4.3.5 Relationship between prediction error and s

Figure 4.32 shows the relationship between expected prediction error and s . From this figure, it can be seen that the estimated error decreases with the increase in the size of the training data set.

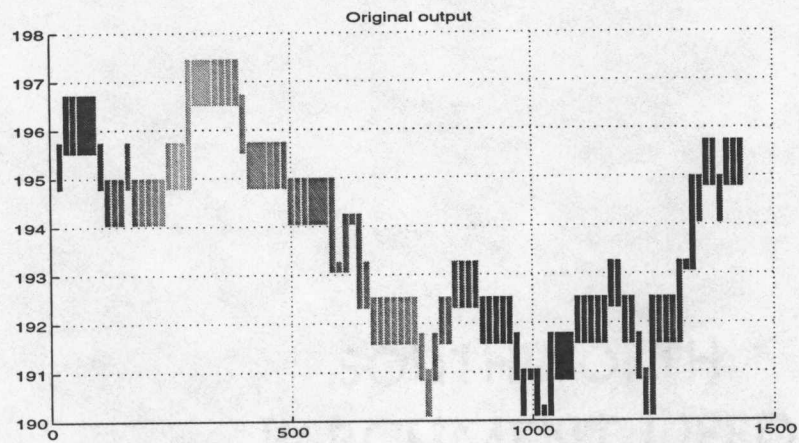


Figure 4.26: True output.

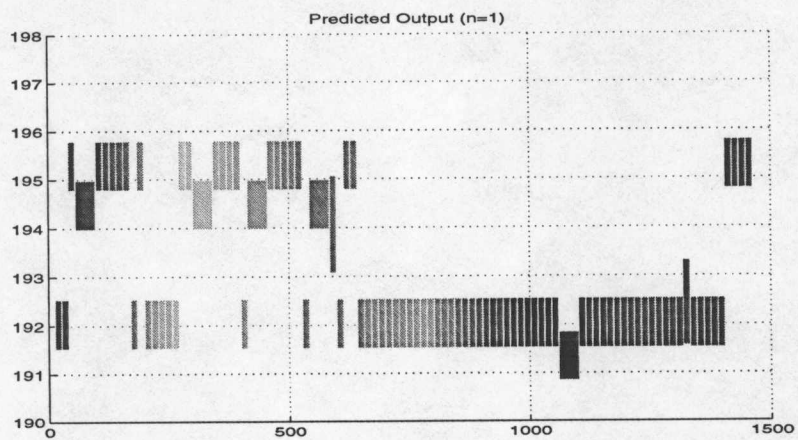


Figure 4.27: Predicted output of model with $n=1$, ($N = 50, s = 91$).

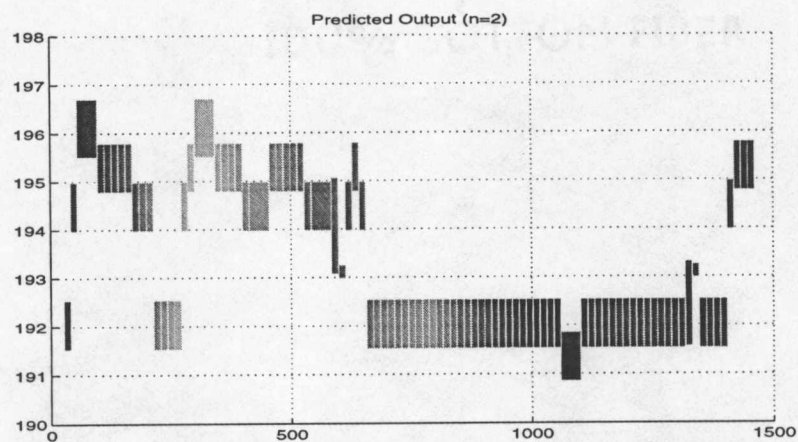


Figure 4.28: Predicted output of model with $n=2$, ($N = 50, s = 91$).

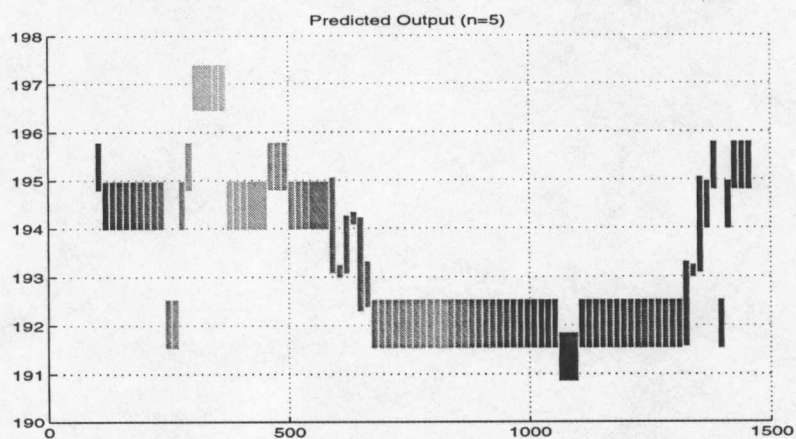


Figure 4.29: Predicted output of model with $n=5$, ($N = 50, s = 91$).

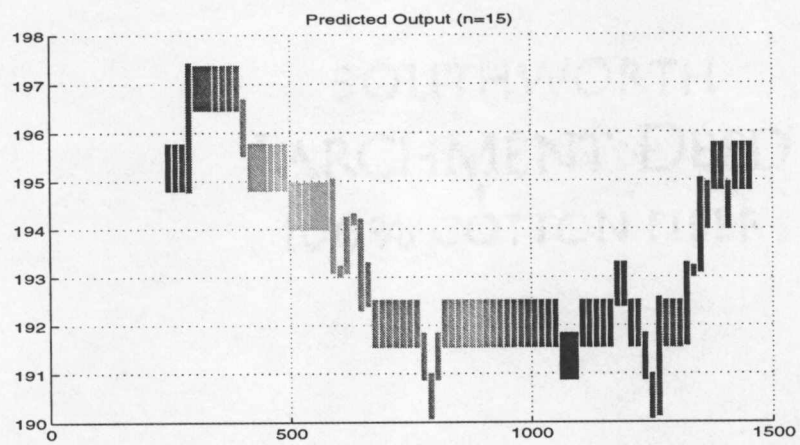


Figure 4.30: Predicted output of model with $n=15$, ($N = 50, s = 91$).

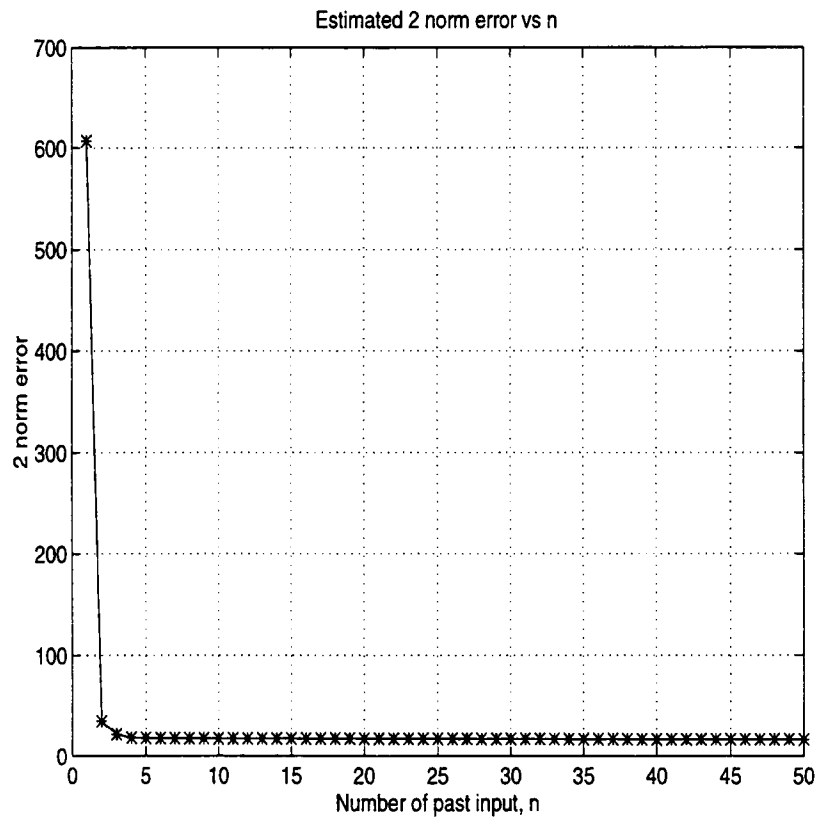


Figure 4.31: Estimated 2 norm error vs. n , ($N = 30, s = 233$).

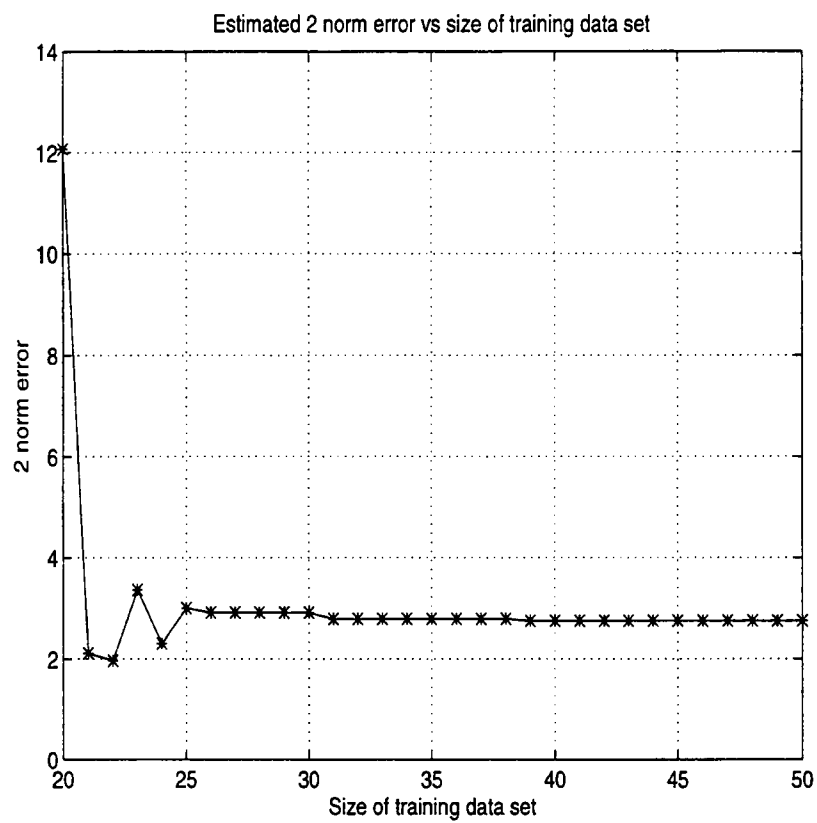


Figure 4.32: Estimated 2 norm error vs. size of training data set, ($N = 40, n = 10$).

The estimated error decreases because as the size of the training data set increases more new objects are included, and therefore the ratio of repeated object states to system states become smaller. The curve of error versus s will eventually level out once all the possible objects in the process have been included in the training data set.

4.3.6 Relationship between prediction error and (N, n, s)

Three dimensional surface plots of the expected prediction error versus (N, n) are shown in Figures 4.33 and 4.34 for different set of data to show the interaction between model parameters.

4.4 Summary

An engineer who plans to use a FIRM model of a dynamic process must determine acceptable values for the parameters N (number of discrete levels used to represent object parameters), n (number of past input objects used to calculate model state), and s (size of the training data set). The choice of these values should be based upon a selected maximum acceptable mediation error, which is in the same units as the square of the output variable. This error can be viewed as equivalent to a standard mean squared error; its square root is equivalent to a measure of this error. Once this maximum acceptable error, is chosen, and training and validation data are available, the methods discussed in this chapter can be used to graphically select values for (N, n, s) . A protocol for the modeling process is described in the next chapter.

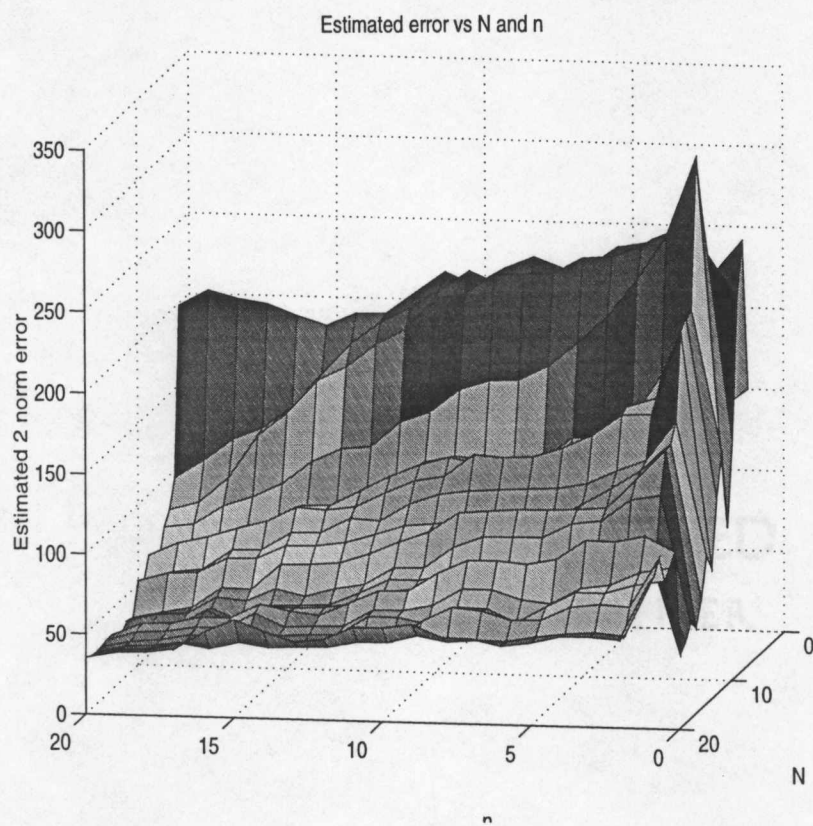


Figure 4.33: Error versus (N, n) , ($s = 188$).

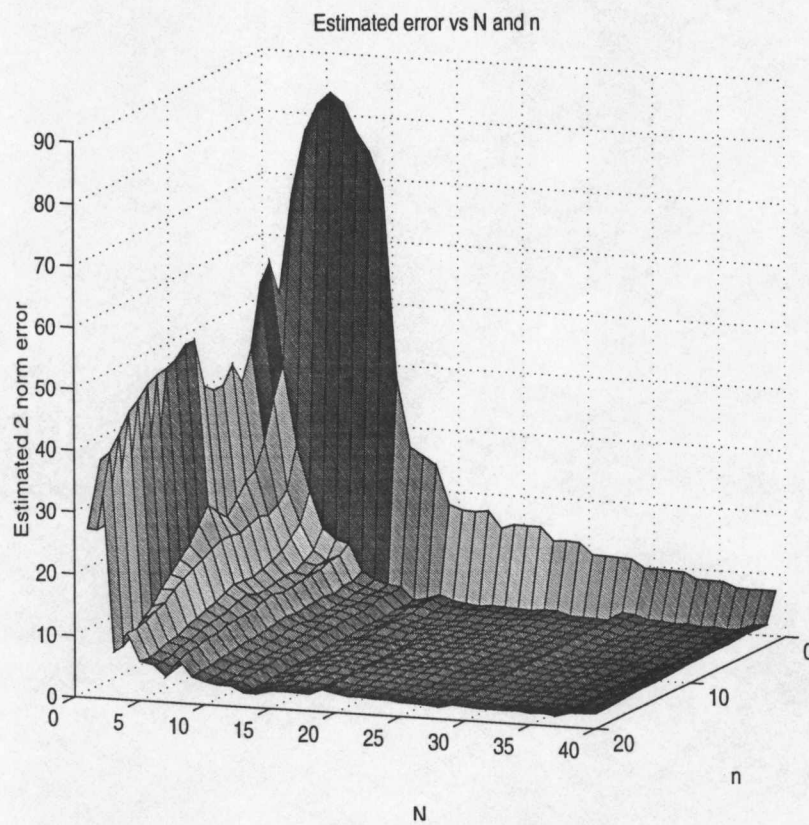


Figure 4.34: Error versus (N, n) , $(s = 80)$.

CHAPTER 5

Protocol for Application

In order to model a given set of data with a FIRM model parameterized by (N, n, s) , a general approach is to be used to select the appropriate values for N, n and s . In this chapter, a set of model identification procedures will be developed based on the model features and parameter relationships discussed in Chapter 4. Before establishing the procedures for selecting the values of the parameters, N, n and s , the effects of the parameters and interactions between parameters on model features are further examined using 2^k factorial experimental design [19, 20]. The results from the experimental design are then combined with the discussion in Chapter 4 to form the identification procedures. Once the identification procedures are established, a set of general guidelines for implementing the model will be presented. Simulation and validation of the model is then performed using these procedures.

5.1 2^k factorial experimental design

In a FIRM model, the model features, size of $\underline{P}$, complexity of $\underline{P}$ and model prediction error are functions of the parameters, N, n and s . The study of the effect on these features of N, n and s by varying one parameter at a time does not take the joint effect of the parameters into account. Therefore, an experimental design method was used

to study the simultaneous effects of the parameters in order to take into consideration the interactions between the parameters.

2^k factorial experimental design [19, 20] is used in the research. In this factorial design, more than one parameters are varied to study the effects of these parameters and their interactions on the model features. This is accomplished by examining all possible combinations of two levels of the parameters in each complete trial of the experiment. In the 2^k design, two levels from each of the three parameters are selected to study the parameter effects, and denotes the number of parameters. Thus, in each complete trial, there are a total of eight treatment combinations. The two parameter levels in the experiment are selected randomly from N, n and s in each trial. Therefore, the random effects model is applied to this experiment whereby the conclusions of the experiment can be extended to levels of the parameter which are not considered in the experiment.

The effects of interest in this design are A, B, C, AB, AC, BC and ABC where A, B and C refer to the effects of N, n and s respectively while AB, AC, BC and ABC refer to the effects of interactions between N and n , N and s , n and s , and N, n and s respectively. Detailed information on the method to calculate these values can be obtained from [19, 20].

This 2^k factorial design is implemented using Matlab. The resulting outputs from applying the Matlab function on two different sets of data are shown in the tables in Figures 5.1 and 5.2.

Results shown in Figures 5.1 and 5.2 indicate that the parameters N, n and s have

Jul 31 1997 02:13:27

ed2.log

Page 1

>> error_design

N_lh =

10 40

n_lh =

5 10

S_lh =

15 80

1.2.3.4.5.6.7.8.

Analysis of Variance

Source of variation	Sum of Squares	d.o.f	Mean Square	Effect
A	4096.1	1	4096.1	-45.26
B	22465.0	1	22465.0	-105.98
AB	31.1	1	31.1	3.94
C	30215.5	1	30215.5	-122.91
AC	147.2	1	147.2	-8.70
BC	673.5	1	673.5	11.13
ABC	0.3	1	0.3	-0.36
Total	57628.7	7		

>> diary off

Figure 5.1: Factorial experimental design using data set 1.

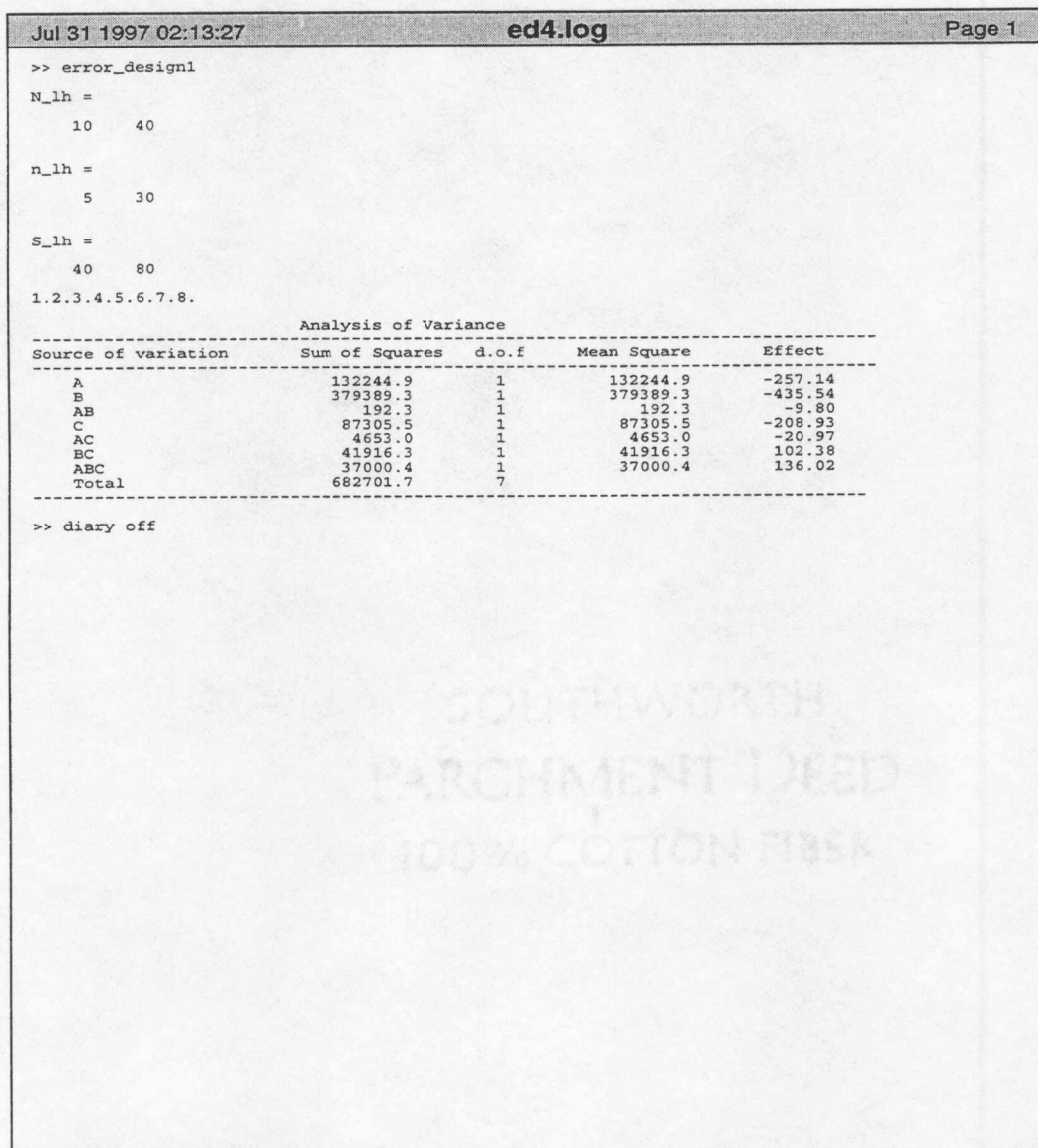


Figure 5.2: Factorial experimental design using data set 2.

negative effects on the model prediction error. The larger these parameters become, the smaller the prediction error. This matches the results shown in Chapter 4 where the prediction error decreases asymptotically as N , n and s are increased. Further, it is noted that the effects of the interactions between parameters (AB , AC , BC and ABC) are small compared to the effect of the parameters taken individually (A , B and C). This shows that the effect of interactions between model parameters is not significant compared to the effect of the choice of model parameter values. This finding will be used in the next section to establish identification techniques for FIRM model.

5.2 Model identification

The main concern in the identification of a model is to select an appropriate FIRM model to fit a given set of data. In this process, an appropriate subclass of models from the general FIRM family will be identified to represent the given data. The primary reason for this model selection is to allow the controller to systematically create a model and take account of the main features of the model.

Identification methods presented in this section are rough procedures for obtaining some idea of the values of N , n and s appropriate for the FIRM model. The identification methods are necessarily inexact because of the uncertainty of the types and nature of the physical process which exist in practice, and therefore the model parameters cannot be decided purely by mathematical arguments. The choice should be guided by the modeler's operating experience and judgement. Estimation and graphical meth-

ods are the principal tools employed in the process of identifying appropriate model parameters.

5.2.1 Identification Techniques

The previous chapter presented the relationships between model features and parameters. These relationships are summarized in Table 5.1.

This table indicates that the values of N , n and s should be as large as possible in order to obtain a good model for the process. However, after a certain point, further increases in the values of N , n and s only cause a very slight or almost no improvement to the model features. This is illustrated in the graphs on the relationships between model features and parameters in Chapter 4. In these graphs, the curves level out after the parameters become sufficiently large. In addition, increases in the values of N , n and s require larger training sets to obtain a sufficiently high percentage of successful predictions of future objects. Therefore, values of the parameters which are sufficiently large to render adequate model complexity and prediction error but not too large to effect an increase in the model size, execution time and memory needed for model implementation should be used.

Based on the observations in Chapter 4 and the conclusions of the 2^k factorial design, it is possible to develop several rules of thumb guiding the selection of the values of N , n and s . Most of these rules are derived using the data sets available to the researches. These rules may have to be modified when applied to data from other processes under consideration, and as experience accumulates.

Table 5.1: Relationships between model features and model parameters.

Model Features	$N \uparrow$	$n \uparrow$	$s \uparrow$
Size of $\underline{P}$	inc. asymptotically	inc. asymptotically	inc. monotonely
Complexity of $\underline{P}$	dec. asymptotically	dec. asymptotically	dec. asymptotically
Prediction error	dec. asymptotically	dec. asymptotically	dec. asymptotically

The rules for determining N , n and s are as follows :

1. The values of N , n and s can be chosen independently.

The 2^k factorial experimental design shows that the effects of interactions between parameters on model features are small compared to the main effects of the model parameters. This negligible interactions effects allow independent selection of the values of N , n and s .

2. Choose n in the range $6 \leq n \leq 10$

In FIRM model, the modeling of the input and output signal begins at the n^{th} object. If a large n is used, a large number of the object states will be omitted in the modeling process. This will cause a loss of some dynamic characteristics of the signal which may be important to the process. For the process considered here, $6 \leq n \leq 10$ appears to be sufficient to represent the correlation between the objects. This value may vary for different applications depending on the nature of process. The values of n should be increased if the output events of the process are known to be caused by a large sequence of input events. For example, in the effort to predict product defects, a value of n which is larger than k but smaller than $2k$ should be used if it is known that the defects are often caused by a k sequence of machine state. Therefore, it is necessary to apply some operational experience and information about the nature of the process in determining the value of n .

3. Choose $s \gg n$ and N

As mentioned above, a FIRM model omits the first $(n - 1)$ objects when forming the system states of the model. If the size of the training data set, s , were chosen to be close in magnitude to n , then a large portion of the objects in the training set would be omitted. The training data set would not be sufficiently large and varied enough to describe the dynamic behavior of the process. It is advisable that $s > 50n$ be used. Again, this value is obtained based on the data set available to the research.

The total number of different objects that can be represented by a FIRM model is nN^3 . The number of objects which are actually produced by a process is typically a small fraction, $q \ll 1$, of this number. In order to obtain a reasonably accurate probability distribution of output states for each input state on the order of 5-20 instances of each input combination, $(n \text{ objects})$ should be observed in the training set. If 10% of the set of possible input combinations are produced by the process, the training set size, s , should be at least nN^3 .

4. The value of N depends on the total number of objects in the input signal after refinement

In Section 4.1.1, it was noted that the maximum number of rows and columns of $\underline{P}$, r^* and c^* are equal to the total number of unrepeated objects in the input or output signal after refinement, s_R , for a specific n and s . Therefore, the value of N should be selected from $1 \leq N \leq N^*$ where N^* is the value of N that corresponds to r^* . The graph depicting the size of $\underline{P}$ versus N with n and s held

constant at the values chosen using rules 2 and 3 can be used to determine the appropriate value N . The value of N should be selected corresponding to the point where the curve begins to level off. For the various data sets used in this research, this value of N lies in the range of $\frac{1}{6}s_R \leq N \leq \frac{1}{4}s_R$.

The four rules stated above provide an initial estimate for the parameters, N, n and s . The initial estimates of the parameters using these rules are summarized in Table 5.2. After selecting the values of these parameters, model validation (topic of next section) needs to be performed to examine the adequacy of the developed model.

5.2.2 Model validation

Once the values of N, n and s are selected, model validation is performed to examine the adequacy of the developed model in representing the training data set. In a FIRM model, adequacy can be checked using the following steps.

1. While holding N and n at the selected values, plot the relationship between prediction error and training set size, s . The prediction error should decrease asymptotically with s . If the prediction error does not decrease with s , a larger value of N should be adopted. This is because N , the number of discretization of the ranges of object height, duration and midpoint, is not fine enough to represent the new objects. Therefore, the new objects introduced to the model by increasing the size of training data set are encoded as object states which have already been observed.

Table 5.2: Rules for initial estimates of parameters .

Model Parameters	Initial Estimates
n	$6 \leq n \leq 10$
s	$s > \max(50n, 5qnN^3)$
N	$\frac{1}{6}s_R \leq N \leq \frac{1}{4}s_R$

2. While holding N and s at the selected values, plot the relationship between prediction error and n . The prediction error should decrease asymptotically with n . If the prediction error does not decrease with n , a larger value of N should be used for the same reason described above.
3. While holding n and s at the selected values, plot the relationship between prediction error and N . The prediction error should decrease asymptotically with N . Check the prediction errors for values of N larger than s_R , where s_R is the number of objects in the refined input signal. If these prediction errors are not acceptable, it shows that the training data set is not sufficiently large or diverse. A larger value of s should be used to resolve the problem.

If the model is found to be inappropriate, rules 1 to 4 in Section 5.2.1 should be re-examined, a different set of values for the parameters can be selected.

5.3 General procedure for FIRM modeling

The general procedure for implementing a FIRM model is shown in Figure 5.3. This procedure is derived from the basic work performed in this research.

5.4 Simulation

Based on the procedures discussed above, a FIRM model is developed from data obtained from an operating industrial process. Two sets of input-output data from the

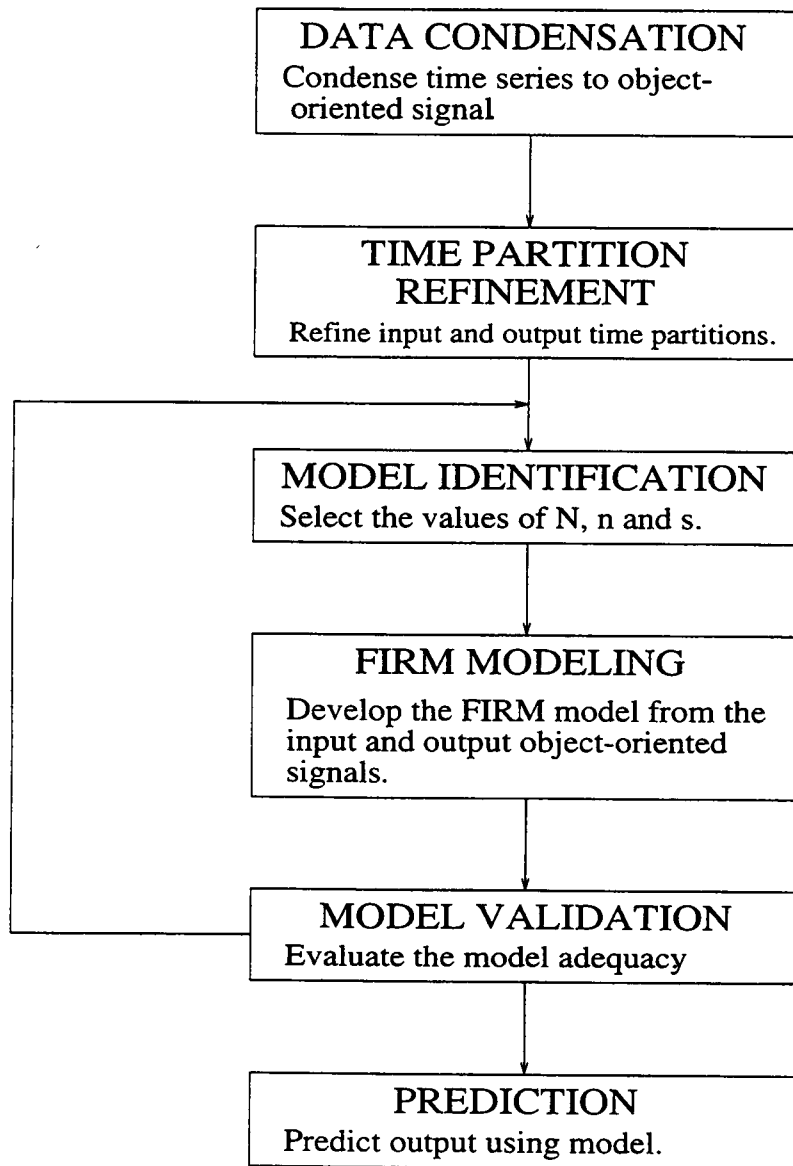


Figure 5.3: A general paradigm for implementing FIRM modeling.

process are used in the simulation process. The first set of data is used as the training set to develop the FIRM model. A data selection technique is introduced to provide a systematic way of selecting the appropriate training data set. The second set of data is used as the model validation data set for verifying the model features (prediction error and percentage of predictable objects) and parameters relationships.

In this section, the technique for selecting the training data set is first presented. This technique is used to select a set of data for developing the FIRM model. The model is then used to predict the output of the model validation data set. The prediction error and percentage of predictable objects are studied by varying the model parameters.

5.4.1 Selection of training data set

Selection of the appropriate training data set has been an important issue in FIRM modeling. A suitable training data set must be used in developing the FIRM model to ensure high accuracy of the model in predicting future output events. The training data set must be large and diverse enough to include the important dynamic behavior of the process, but not overly large so that unnecessary computational resources are required. In most applications, it is advantageous to include the sequences of input events that cause the output events of interest and exclude those sequences that do not. Improper selection of a training data set can result in effectiveness of model in predicting output behaviors of interest.

Due to the nature of the FIRM model in relating output events to sequences of past input events, the training data can not be selected by choosing a single event at

a time. Selection of the training data must be done by choosing sequences of events rather than single events. Since system states of the FIRM model are derived from groups of n events of the input signal, identification of the training data set must be done by choosing n events at each selection. For each value of n , a listing of groups of n events as shown in Table 5.3 can be created to enable the selections of the training data. Sequences of events possessing important dynamic behavior of the process should be included while others can be ignored. Note that similar tables can be derived from this table for all smaller values of n by excluding appropriate columns.

5.4.2 Simulation results

Using the method presented in Section 5.4.1, training data sets are selected to develop the FIRM models. All data available from operation of the process were used to form a table as in Table 5.3, and training sets were formed by randomly selecting rows from the table. A similar approach was used to select the model validation data set. FIRM models are developed from these training data sets. Input of the model validation sets are used to predict the process output. The predicted outputs are compared to the actual outputs in the model validation data, and the prediction error and percentage of predictable objects are calculated. The following sections explore the characteristics of the prediction error and percentage of predictable objects.

Table 5.3: Listing of groups of n events.

Seq.	Input objects					Output objects
no.	u_{k-n+1}	u_{k-n+2}	$\cdots$	u_{k-1}	u_k	y_{k+1}
1	u_0	u_1	$\cdots$	$\cdots$	u_n	y_{n+1}
2	u_1	u_2	$\cdots$	$\cdots$	u_{n+1}	y_{n+2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Effect of N on prediction error and percentage of predictable objects

Figure 5.4 shows that the prediction error decreases asymptotically as N is increased. This agrees with the results in Section 4.3.3 which show that larger discrete values used to represent the values in the ranges of object height, duration and midpoint can increase the prediction accuracy.

As mention above, the training data set in the simulation process is selected randomly. For each value of N , a different training set is used to develop the FIRM model. Similarly a different model validation set is randomly selected each time N is increased. Due to the random selection of training and model validation data sets, the percentage of predictable objects varies randomly with the increase of N . This is illustrated in Figure 5.5.

Effect of n on prediction error and percentage of predictable objects

As discussed in Section 4.3.4, prediction error of the model decreases with increasing n . When n is increased, less repeated system states due to desensitization occur in the input signal. Therefore the prediction error decreases as n is increased (Figure 5.6). The percentage of predictable objects varies randomly with the increased of n (Figure 5.7) due to the random selection of training and model validation data sets.

Effect of s on prediction error and percentage of predictable objects

An increase of the size of the training data set will decrease the number of unpredictable system states in the model. This can be seen in Figures 5.8 through 5.13.

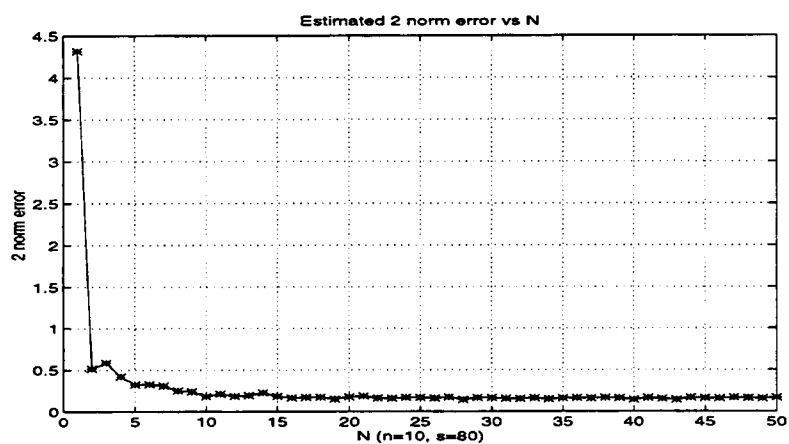


Figure 5.4: Prediction error versus N ($n = 10, s = 80$).

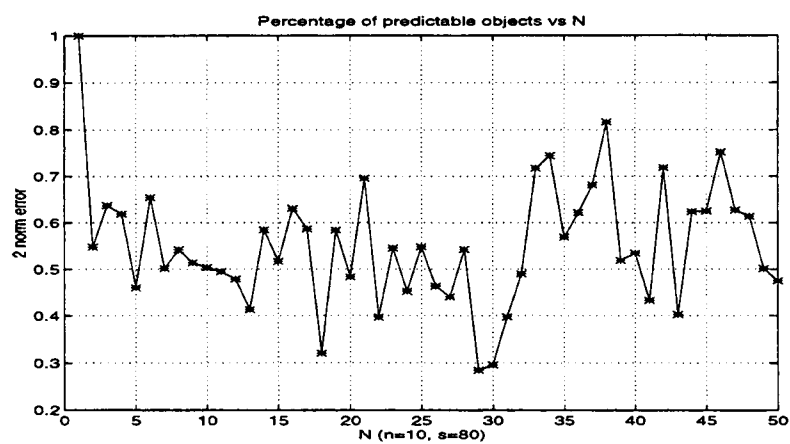


Figure 5.5: Percentage of predictable objects versus N ($n = 10, s = 80$).

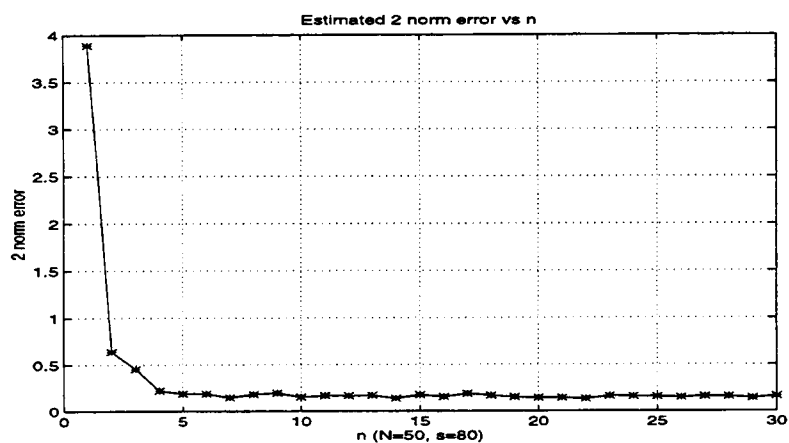


Figure 5.6: Prediction error versus n.

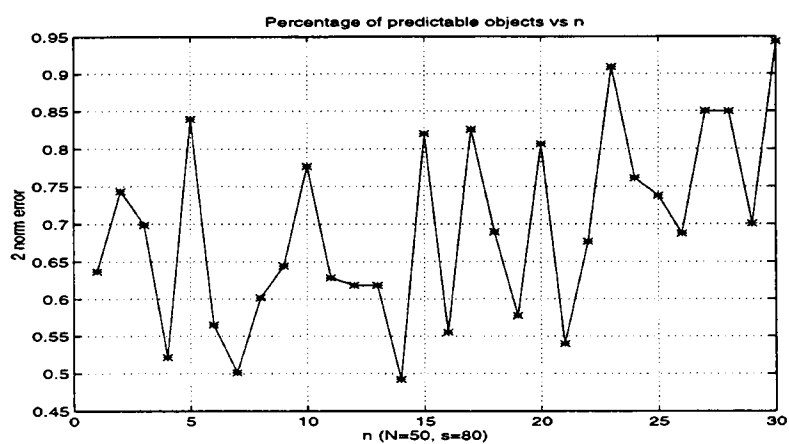


Figure 5.7: Percentage of predictable objects versus n.

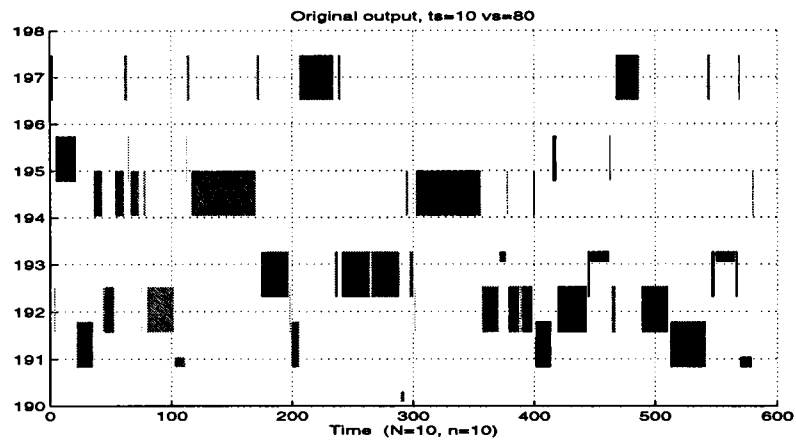


Figure 5.8: True output of the model validation set.

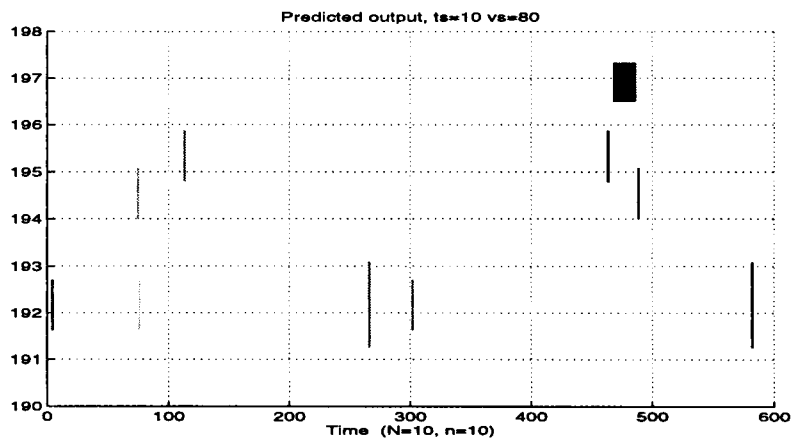


Figure 5.9: Predicted output.

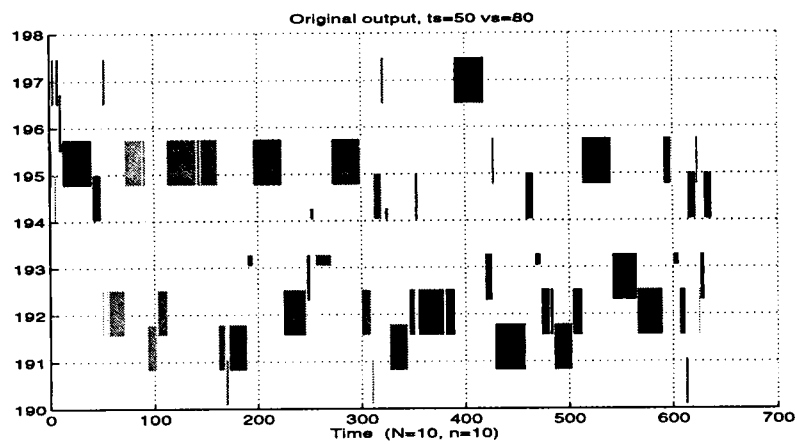


Figure 5.10: True output of the model validation set.

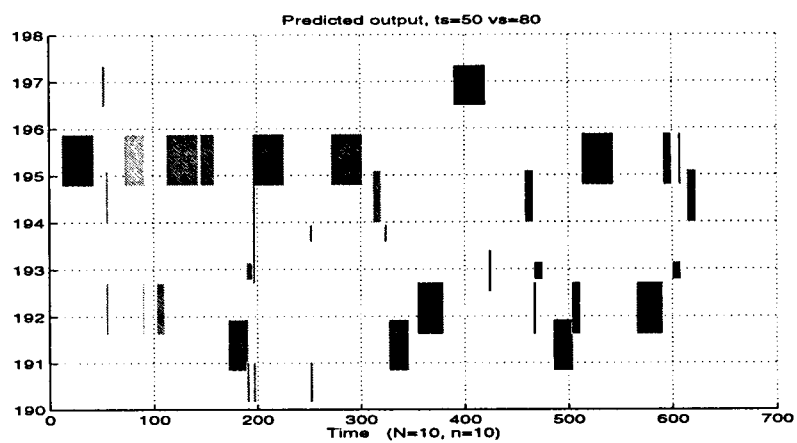


Figure 5.11: Predicted output.

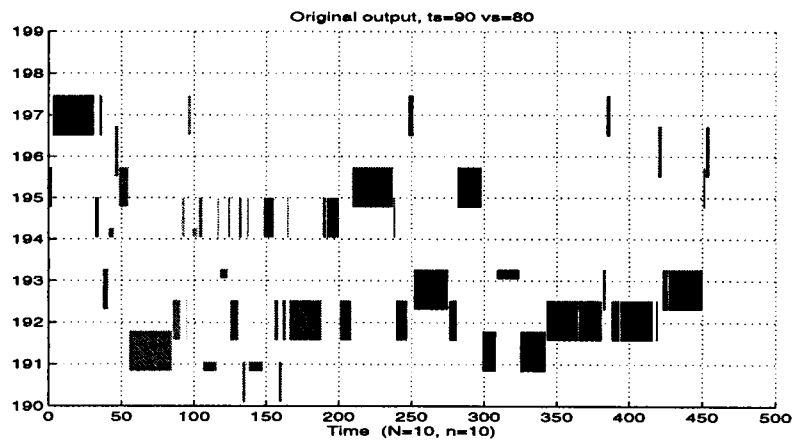


Figure 5.12: True output of the model validation set.

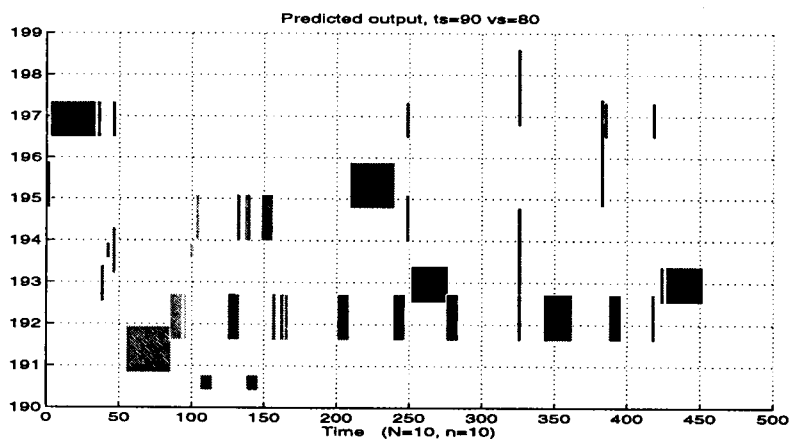


Figure 5.13: Predicted output.

These figures show that as the size of training data set is increased, more system states can be predicted. Therefore the percentage of predictable objects will increase as s is increased. This is illustrated in Figure 5.14.

The decreasing occurrence of unpredictable states with increasing training set size, s does not affect the graph of prediction error versus training set size since the calculation of prediction error does not depend upon unpredictable state. Therefore the prediction error varies randomly as s is increased (Figure 5.15) due to the random selection of data set.

5.5 Summary

This chapter has described the procedures for implementing FIRM model. Using the discussions in Chapter 4 and results from 2^k factorial experimental design, a set of model identification techniques were developed to select the values of model parameters, N , n and s . The procedure for validating the adequacy of FIRM model was also presented.

An FIRM model was developed based on the implementation procedure. The model was developed from data obtained from an operating industrial process. The developed model was then validated in the simulation process using model validation data set. Results of the simulation agreed with the discussions presented in Chapter 4.

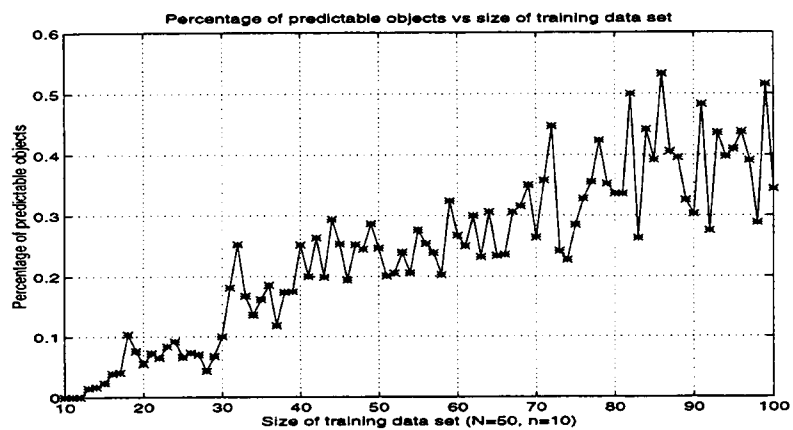


Figure 5.14: Percentage of predictable objects versus s .

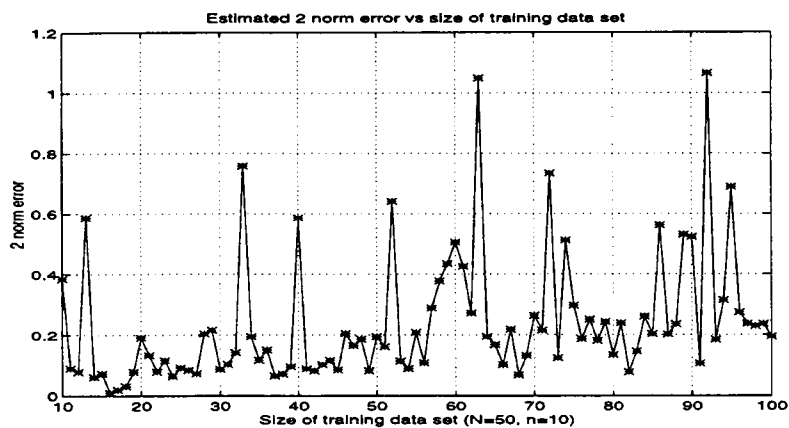


Figure 5.15: Prediction error versus s .

CHAPTER 6

Summary and Future Research Directions

This research explored and evaluated the FIRM modeling approach for representing dynamic behavior of object-oriented signals. The basic strategy for developing the FIRM model has been to encode object signal and sets of objects to object states and system states, respectively. A probability transition matrix can then be derived to relate the system states and output states, and this matrix can be used to predict the process future output objects.

Although this research is the beginning stage of development of the FIRM modeling approach, the result from the simulation demonstrates its potential in representing highly nonlinear and complicated industrial process. The research shows that the FIRM model is capable of representing process where the relationship between input and output signals is impossible or overly complex to be modeled by traditional modeling approach. Examples of this category of processes include waste stream treatment, polymer and combustion processes. Tests performed on the model revealed that its performance exceeds that of a traditional Markov chain.

The main characteristic of the FIRM model is its capability of predicting future output events based on a sequence of past input events. This characteristic of the model is particularly useful in application where product defects or variations in product

quality can occur due to complex or poorly understood chains of events. Therefore the application potential of the FIRM model includes event detection, reduction of product variation and control of product transitions.

This chapter presents the main result of the development of the FIRM model and discusses several further research prospects.

6.1 Research summary

A new process modeling approach using FIRM model has been presented in this research. The model is developed from a set of input and output object-oriented signals. The data condensation technology which condenses time series into object-oriented signals has also been presented. The methodology for encoding objects into object and system states is discussed along with the derivation of the probability transition matrix relating these states.

Main features of the model are explored to study the effects of model parameters on improving model performance. The relationships between model features and parameters are investigated. These relationships are used as the guidelines for selecting the appropriate model to fit a given set of data.

Model identification techniques are generated from the study of parameters effects on model features using experimental design. Procedures for selecting the values of model parameters are presented. General procedures for using the FIRM model is developed. A FIRM model is then constructed from data obtained from an industrial

process. Simulation of the model is done and superior performance of the model is achieved compared to what traditional modeling methods allow.

6.2 Future research directions

Several important questions have surfaced during the development of FIRM modeling approach. One of the most interesting questions to answer is the relationships between model features and probability transition matrix features. There are two features of the probability transition matrix, $\underline{P}$, which are closely related to the model features : rank of $\underline{P}$ and the fundamental subspaces [21] of $\underline{P}$. These two features may provide some practical mathematical guidelines in selecting the values of model parameters. Future research can consider the relationships between these features and the features of model, as well as the proper interpretation of these two attributes

Control applications of the FIRM model is another direction of further research. Control problem can be formulated to find the optimal future input signal object sequence which minimizes a given expected future cost-to-go function [22, 24, 25]. The fact that each signal object occupies a random time interval causes the control problem to be somewhat unique and complicated. It is envisioned that dynamic programming algorithms [23] can be used to solve the control problem with moving time horizon. Once an admissible control law to the optimal control problem is found, a mapping must be chosen to generate control signals from the signal objects.

FIRM modeling approach offers many advantages over traditional modeling meth-

ods. It provides a hybrid predictive model which is capable of predicting process future output accurately. FIRM modeling approach can be used to develop model for process with highly nonlinear and complex dynamic characteristics. With the simple implementation procedure, FIRM model provides an important modeling approach for understanding and controlling of extremely complex industrial processes. However, suitable modifications of well-developed control approaches need to be developed to accommodate the new FIRM modeling paradigms.

BIBLIOGRAPHY

BIBLIOGRAPHY

- [1] Revuz, D., *Markov Chains*, North-Holland, 1984.
- [2] Adke, S. R., and S. M. Manjunath, *An Introduction to Finite Markov Chain Processes*, John Wiley & Sons, 1984.
- [3] DeFatta, D. J., J. G. Lucas, and W.S. Hodgkiss, *Digital Signal Processing : A System Design Approach*, John Wiley & Sons, pp. 49-54, 1988.
- [4] Beauchamp, K., and C. Yuen, *Digital Methods for Signal Analysis*, George Allen & Unwin, pp.221-233, 1979.
- [5] Kendall, M., and J. K. Ord, *Time Series*, Oxford University Press, 1990.
- [6] Brockwell, P. J., and R. A. Davis, *Introduction to Time Series and Forecasting*, Springer, 1996.
- [7] Hamilton, J. D., *Time Series Analysis*, Princeton University Press, 1994.
- [8] Dynkin, E. B., *Markov Processes and Related Problems of Analysis*, Cambridge University Press, pp. 187-218, 1982.
- [9] Bellanger, M., *Digital Processing of Signals*, John Wiley & Sons, pp. 111-144, 1984.
- [10] Oppenheim, A. V., and R. W. Schafer, *Digital Signal Processing*, Prentice-Hall, pp. 155-163, 1975.
- [11] Bateman, A., and W. Yates, *Digital Signal Processing Design*, Computer Science Press, pp. 190-202, 1989.
- [12] Birdwell, J. D., and B. C. Moore, "Condensation of information from signals for process modeling and control", *Hybrid Systems II*, Springer, pp. 45-63, 1995.
- [13] Wade, W. D., *An Introduction to Analysis*, Prentice-Hall, pp. 99-100, 1995.
- [14] Purcell, J. E., *Calculus, with Analytic Geometry*, Appleton-Century-Crofts, pp. 207-209, 1965.
- [15] Banks, J., J. S. Carson II, and B. L. Nelson, *Discrete-Event System Simulation*, Prentice-Hall, pp. 9-13, 1996.
- [16] Varaiya, P., A. B. Kurzhanski, "Discrete event systems : models and applications", *Lecture Notes in Control and Information Sciences*, Springer-Verlag, 1988.
- [17] Cooper, G. R., and C. D. McGillem, *Probabilistic Methods of Signal and System Analysis*, Harcourt Brace Jovanovich College Publishers, pp. 48-141, 1986.

- [18] The Math Works Inc., *Matlab Reference Guide*, Van Nostrand Reinhold Co., 1992.
- [19] Montgomery, D. C., and G. C. Runger, *Applied Statistics and Probability for Engineers*, John Wiley & Sons, pp. 126-177, 1994.
- [20] Walpole, R. E., and R. H. Myers, *Probability and Statistics for Engineers and Scientists*, Macmillan Publishing Company, pp. 503-529, 1985.
- [21] Strang, G., *Linear Algebra and Its Application*, Harcourt Brace Jovanovich, pp. 1-128, 1988.
- [22] Smallwood, R. D., and E. J. Sondik, "The optimal control of partially observable Markov processes over a finite horizon", *Operations Research*, Vol. 21, pp.1071-1088, 1973.
- [23] Bertsekas, D. P., "Dynamic Programming and Stochastic Control", *Math. in Science and Engineering*, Vol. 125, pp.111-144, Academic Press, 1976.
- [24] Sondik, E. J., "The optimal control of partially observable Markov processes over the infinite horizon : discounted costs" *Operations Research*, Vol. 26, pp.283-305, 1978.
- [25] Lovejoy, W. S., "A survey of algorithmic methods for partially observed Markov decision processes", *Annals of Operations Research*, Vol. 28, pp.47-66, 1991.

APPENDICES

APPENDIX A

Matlab code list of the refinement process

```
% REFINE Refine the time partitions of two object oriented
% sequences to produce a common time partition
% (refinement) for both sequences. Refinement, R,
% is given by
%  $R = P \cup Q$ 
% where P and Q are the time partition of the
% two sequences.
% [objs] = refine(A,B) produce the matrix objs
% which contains
% the refinement and both signals in the form
%  $objs = [ \text{ref}_l \text{ref}_r \text{bot}_A \text{top}_A \text{bot}_B \text{top}_B ]$ 
% The matrices A and B must be in the form
%  $[ \text{left right} \mid \text{bot top} ]$ .
%
% See also CONVERT, MODEL and PREDICT
function [objs] = refine(A,B)

[m_A,n_A] = size(A) ;
[m_B,n_B] = size(B) ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Convert the time interval into numerical numbers
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t_l_A = zeros(m_A,1) ;
t_r_A = zeros(m_A,1) ;
day_i = A(1,3) ;
K = 0 ;
for i = 1:m_A
    day_l = A(i,3) ;
    day_r = A(i,9) ;
    if day_l ~= day_i
        day_i = A(i,3) ;
        K = K + 24*60 ;
    end
    t_l_A(i) = A(i,5)*60 + A(i,6) + A(i,7)/60 + K ;
    if day_r ~= day_i
        day_i = A(i,9) ;
        K = K + 24*60 ;
    end
    t_r_A(i) = A(i,11)*60 + A(i,12) + A(i,13)/60 + K ;
end
s = A(1,5)*60 + A(1,6) + A(1,7)/60 ;
t_l_A = t_l_A - s*ones(size(t_l_A)) ;
t_r_A = t_r_A - s*ones(size(t_r_A)) ;
```

```

t_l_B = zeros(m_B,1) ;
t_r_B = zeros(m_B,1) ;
day_i = B(1,3) ;
K = 0 ;
for i = 1:m_B
    day_l = B(i,3) ;
    day_r = B(i,9) ;
    if day_l ~= day_i
        day_i = B(i,3) ;
        K = K + 24*60 ;
    end
    t_l_B(i) = B(i,5)*60 + B(i,6) + B(i,7)/60 + K ;
    if day_r ~= day_i
        day_i = B(i,9) ;
        K = K + 24*60 ;
    end
    t_r_B(i) = B(i,11)*60 + B(i,12) + B(i,13)/60 + K ;
end
s = B(1,5)*60 + B(1,6) + B(1,7)/60 ;
t_l_B = t_l_B - s*ones(size(t_l_B)) ;
t_r_B = t_r_B - s*ones(size(t_r_B)) ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Sort the numbers to refine the time intervals %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Y_1 = [ t_l_A ; t_r_A(m_A,1) ] ;
Y_2 = [ t_l_B ; t_r_B(m_B,1) ] ;
Y = [ Y_1 ; Y_2 ] ;
Y = sort(Y) ;

K = 0 ;
for i = 1:size(Y,1)-1
    if Y(i) == Y(i+1)
        Y(i) = NaN ;
        K = K + 1 ;
    end
end

Y = sort(Y) ;
Y = Y(1:size(Y,1)-K,1) ;

start_pt = max(Y_1(1),Y_2(1)) ;
for i = 1:size(Y,1)
    if Y(i) == start_pt
        s = i ;
    end
end

end_pt = min(Y_1(m_A+1),Y_2(m_B+1)) ;
for i = 1:size(Y,1)
    if Y(i) == end_pt
        e = i ;
    end
end

```

```

end

objs(:,1) = Y(s:e-1)           ;
objs(:,2) = Y(s+1:e)           ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Matching the time intervals with the data %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1:size(objs,1)
    for j = 1:size(Y_1)-1
        if objs(i,1) >= Y_1(j)
            if objs(i,1) <= Y_1(j+1)
                objs(i,3) = A(j,14)      ;
                objs(i,4) = A(j,15)      ;
            end
        end
    end
end

for i = 1:size(objs,1)
    for j = 1:size(Y_2)-1
        if objs(i,1) >= Y_2(j)
            if objs(i,1) <= Y_2(j+1)
                objs(i,5) = B(j,14)      ;
                objs(i,6) = B(j,15)      ;
            end
        end
    end
end

[m_objs,n_objs] = size(objs)      ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot the original and refined object oriented signal %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
min_A = min(t_l_A)                ;
max_A = max(t_r_A)                ;
min_B = min(t_l_B)                ;
max_B = max(t_r_B)                ;

figure(1)
clg
for i = 1:m_A
    x_A = [ t_l_A(i) ; t_l_A(i) ; t_r_A(i) ; t_r_A(i) ] ;
    y_A = [ A(i,14)  ; A(i,15)  ; A(i,15)  ; A(i,14)  ] ;
    patch(x_A,y_A,i) ;
end
title('Input signal in original time interval') ;
grid on
axis_A = axis ;
axis([min([min_A,min_B]) max([max_A,max_B]) axis_A(3:4)]);

figure(2)
clg
for i = 1:m_B
    x_B = [ t_l_B(i) ; t_l_B(i) ; t_r_B(i) ; t_r_B(i) ] ;
    y_B = [ B(i,14)  ; B(i,15)  ; B(i,15)  ; B(i,14)  ] ;
    patch(x_B,y_B,i) ;
end

```

```

end
title('Output signal in original time interval')      ;
grid on
axis_B = axis
axis([min([min_A,min_B]) max([max_A,max_B]) axis_B(3:4)]);
figure(3)
clg
for i = 1:m_objs
    x_u = [ objs(i,1) ; objs(i,1) ; objs(i,2) ; objs(i,2) ] ;
    y_u = [ objs(i,3) ; objs(i,4) ; objs(i,4) ; objs(i,3) ] ;
    patch(x_u,y_u,i)
end
title('Input signal in refined time interval')      ;
grid on
axis([min([min_A,min_B]) max([max_A,max_B]) axis_A(3:4)]);
figure(4)
clg
for i = 1:m_objs
    x_y = [ objs(i,1) ; objs(i,1) ; objs(i,2) ; objs(i,2) ] ;
    y_y = [ objs(i,5) ; objs(i,6) ; objs(i,6) ; objs(i,5) ] ;
    patch(x_y,y_y,i)
end
title('Output signal in refined time interval')      ;
grid on
axis([min([min_A,min_B]) max([max_A,max_B]) axis_B(3:4)]);

```

APPENDIX B

Matlab code list for encoding objects to object states

```
% CONVERT Encode each object in an object-oriented signal
% into a number using the equation :
%      index = index_m + Nm*index_d + Nm*Nd*index_h
% The function CONVERT produces matrices "state",
% which contains the encoded numbers representing each
% object in the signal(using the equation above),
% "state_new", which contains encoded number from 1
% to n and "state_unrep", which contains the list
% of all possible encoded number. CONVERT also
% produces the discretization matrices, x_hist_h,
% x_hist_d and x_hist_m, used in discretize the
% ranges of h, d and m. CONVERT is used to encode
% objects in the training data set. To encode objects in
% the model validation data set, the function CONVERT1
% is used.
%
% See also CONVERT1, CONVERT2, REFIN, MODEL, MODEL1, MODEL2.
function [state,state_new,state_unrep,x_hist_h,x_hist_d,x_hist_m]
    = convert(A,Nhdm)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute the height, duration and midpoints of each object %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[m_A,n_A] = size(A) ;
Nh = Nhdm(1) ;
Nd = Nhdm(2) ;
Nm = Nhdm(3) ;
objs = A ;
heights = objs(:,4)-objs(:,3) ;
durations = objs(:,2)-objs(:,1) ;
midpoints = 0.5*(objs(:,3)+objs(:,4)) ;
[n_hist_h,x_hist_h] = hist(heights,Nh) ;
[n_hist_d,x_hist_d] = hist(durations,Nd) ;
[n_hist_m,x_hist_m] = hist(midpoints,Nm) ;
if Nh == 1
    indx_h = ones(1,m_A) ;
else
    [trash,indx_h] = min(abs(x_hist_h'*ones(1,m_A)-ones(Nh,1)
        *heights')) ;
end
indx_h = indx_h - 1 ;
if Nd == 1
    indx_d = ones(1,m_A) ;
```

```

else
    [trash,indx_d] = min(abs(x_hist_d'*ones(1,m_A)-ones(Nd,1)
        *durations')));
end
indx_d = indx_d - 1 ;
if Nm == 1
    indx_m = ones(1,m_A) ;
else
    [trash,indx_m] = min(abs(x_hist_m'*ones(1,m_A)-ones(Nm,1)
        *midpoints')));
end
indx_m = indx_m - 1 ;
index = indx_m+Nm*indx_d+Nm*Nd*indx_h ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reorganize the state assignments in sequential numbers from 1 to n. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
state = index' ;
[state_sort state_index] = sort(state) ;
S = 0 ;
for i = 1:(m_A-1) ;
    if(state_sort(i,1)<state_sort((i+1),1)) ;
        S = S + 1 ;
        state_unrep(S) = state_sort(i,1) ;
    end ;
end ;
state_unrep = [state_unrep' ; state_sort(m_A)] ;
[a b] = size(state_unrep) ;
state_rename = zeros(m_A,1) ;
for i = 1:a ;
    for j = 1:m_A ;
        if(state_sort(j,1)==state_unrep(i)), ;
            state_rename(j) = i ;
        end ;
    end ;
end ;
state_new = zeros(m_A,1) ;
for i = 1:m_A ;
    state_new(state_index(i))=state_rename(i) ;
end ;

```

APPENDIX C

Matlab code list for calculating P

```
% MODEL This function find the transition matrix P for a FIR markov
% chain model. The input is the unrefined signal in the form
% A = [ left_1 right_1 | bot_1 top_1 ]
% and the output is the unrefined signal in the form
% B = [ left_2 right_2 | bot_2 top_2 ].
% The transition matrix P is a M by Nk matrix, where M is the
% number of possible output and Nk is the number of possible
% n past input combinations that exist in the input signal.
% P shows the probabilities of occurrence of each output given the
% n past inputs combination. The variable, ns, is the number of
% possible states of the input signal A.
% The difference between MODEL and MODEL1 is the input
% arguments. For MODEL1 the input argument must be refined
% signals.
%
% See also REFINE, CONVERT, MODEL1 and MODEL2

function [P,system,y_table,x_hist_h,x_hist_d,x_hist_m]
    = model(A,B,n,Nhdm,ns) ;

[m_A,n_A] = size(A) ;
[m_B,n_B] = size(B) ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Refine the interval of A and B, and find the state index %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

objs = refine(A,B) ;
[m_objs,n_objs] = size(objs) ;
u = objs(:,1:4) ;
y = [objs(:,1:2) objs(:,5:6)] ;
[state_u state_new_u,state_unrep_u,x_hist_h,x_hist_d,x_hist_m]
    = convert(u,Nhdm);
[state_y state_new_y,state_unrep_y,y_table] = convert2(y,Nhdm) ;
stored_temp = zeros(1,n) ;
state = zeros(m_objs,1) ;
for i = 1:m_objs
    stored_temp = [ stored_temp(2:n) state_new_u(i) ] ;
    stored_input(i,:) = stored_temp ;
    for j = 1:n
        state(i) = state(i) + stored_temp(j)*(ns^(j-1)) ;
    end
end
state = state(n:m_objs,:) ;
```

```

state_y      = state_y(n:m_objs)      ;
state_new_y  = state_new_y(n:m_objs)  ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reorganize the state from 1 to total number of states %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m_state = size(state,1)                ;
[state_sort state_index] = sort(state)  ;
S = 0                                  ;
for i = 1:(m_state-1)                  ;
    if(state_sort(i,1)<state_sort((i+1),1))
        S = S + 1                      ;
        state_unrep(S) = state_sort(i,1) ;
    end                                ;
end                                    ;
state_unrep = [state_unrep' ; state_sort(m_state)] ;
[a b] = size(state_unrep)              ;
state_rename = zeros(m_state,1)        ;
for i = 1:a                            ;
    for j = 1:m_state                  ;
        if(state_sort(j,1)==state_unrep(i)),
            state_rename(j) = i        ;
        end                            ;
    end                                ;
end                                    ;
state_new = zeros(m_state,1)           ;
for i = 1:m_state                      ;
    state_new(state_index(i))=state_rename(i) ;
end                                    ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute P %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
system = [ state_new state_new_y ]      ;
P_acc = zeros(size(state_unrep_y,1),size(state_unrep,1));
for i = 1:m_objs-n+1                  ;
    P_acc(system(i,2),system(i,1))
        = P_acc(system(i,2),system(i,1)) + 1 ;
end                                    ;
[m_P,n_P] = size(P_acc)                ;
if m_P == 1                            ;
    column_sum = P_acc                  ;
else                                    ;
    column_sum = sum(P_acc)             ;
end                                    ;
P = P_acc*diag(ones(1,size(P_acc,2))./column_sum) ;

```

VITA

Sze Wei Ong was born on May 17, 1974 in Segamat, Johor, Malaysia. He received a Bachelor of Science degree in Electrical Engineering from the University of Tennessee, Knoxville in 1995. In January, 1996, Sze Wei Ong enrolled in the master program offered by the Department of Electrical Engineering in the University of Tennessee, Knoxville. He received the Master of Science degree in Electrical Engineering in August, 1997.